PHILIPS

Integrated circuits

Part 10 May 1983

Integrated Fuse Logic

signetics

INTEGRATED CIRCUITS

PART 10 - MAY 1983

SIGNETICS INTEGRATED FUSE LOGIC (IFL)

GENERAL CONTENTS

SELECTION GUIDE

INTRODUCTION

IFL SERIES 20

IFL SERIES 28

IFL PROGRAMMING

MILITARY PRODUCTS

PACKAGE OUTLINES

APPLICATIONS

DATA HANDBOOK SYSTEM

Our Data Handbook System is a comprehensive source of information on electronic components, sub-assemblies and materials; it is made up of four series of handbooks each comprising several parts.

ELECTRON TUBES	BLUE
SEMICONDUCTORS	RED
INTEGRATED CIRCUITS	PURPLE
COMPONENTS AND MATERIALS	GREEN

The several parts contain all pertinent data available at the time of publication, and each is revised and reissued periodically.

Where ratings or specifications differ from those published in the preceding edition they are pointed out by arrows. Where application information is given it is advisory and does not form part of the product specification.

If you need confirmation that the published data about any of our products are the latest available, please contact our representative. He is at your service and will be glad to answer your inquiries.

May 1980

ELECTRON TUBES (BLUE SERIES)

The blue series of data handbooks is comprised of the following parts:

T1 **Tubes for r.f. heating**

T2 **Transmitting tubes for communications**

T3 **Klystrons, travelling-wave tubes, microwave diodes**

ET3 **Special Quality tubes, miscellaneous devices** (will not be reprinted)

T4 **Magnetrons**

T5 **Cathode-ray tubes**
Instrument tubes, monitor and display tubes, C.R. tubes for special applications

T6 **Geiger-Müller tubes**

T7 **Gas-filled tubes**
Segment indicator tubes, indicator tubes, dry reed contact units, thyratrons, industrial rectifying tubes, ignitrons, high-voltage rectifying tubes, associated accessories

T8 **Picture tubes and components**
Colour TV picture tubes, black and white TV picture tubes, colour monitor tubes for data graphic display, monochrome monitor tubes for data graphic display, components for colour television, components for black and white television and monochrome data graphic display

T9 **Photo and electron multipliers**
Photomultiplier tubes, phototubes, single channel electron multipliers, channel electron multiplier plates

T10 **Camera tubes and accessories, image intensifiers**

T11 **Microwave components and assemblies**

January 1982

SEMICONDUCTORS (RED SERIES)

The red series of data handbooks is comprised of the following parts:

S1 **Diodes**
Small-signal germanium diodes, small-signal silicon diodes, voltage regulator diodes(< 1,5 W), voltage reference diodes, tuner diodes, rectifier diodes

S2 **Power diodes, thyristors, triacs**
Rectifier diodes, voltage regulator diodes (> 1,5 W), rectifier stacks, thyristors, triacs

S3 **Small-signal transistors**

S4 **Low-frequency power transistors and hybrid IC modules**

S5 **Field-effect transistors**

S6 **R.F. power transistors and modules**

S7 **Microminiature semiconductors for hybrid circuits**

S8 **Devices for optoelectronics**
Photosensitive diodes and transistors, light-emitting diodes, displays, photocouplers, infrared sensitive devices, photoconductive devices.

S9 Taken into handbook T11 of the blue series

S10 **Wideband transistors and wideband hybrid IC modules**

January 1982

INTEGRATED CIRCUITS (PURPLE SERIES)

The purple series of data handbooks is comprised of the following parts:

IC1 **Bipolar ICs for radio and audio equipment**

IC2 **Bipolar ICs for video equipment**

IC3 **ICs for digital systems in radio, audio and video equipment**

IC4 **Digital integrated circuits**
LOCMOS HE4000B family

IC5 **Digital integrated circuits — ECL**
ECL10 000 (GX family), ECL100 000 (HX family), dedicated designs

IC6 **Professional analogue integrated circuits**

IC7 **Signetics bipolar memories**

IC8 **Signetics analogue circuits**

IC9 **Signetics TTL logic**

IC10 **Signetics Integrated Fuse Logic (IFL)**

IC11 **Microprocessors, microcomputers and peripheral circuitry**

May 1983

COMPONENTS AND MATERIALS (GREEN SERIES)

The green series of data handbooks is comprised of the following parts:

C1 **Assemblies for industrial use**
PLC modules, PC20 modules, HNIL FZ/30 series, NORbits 60-, 61-, 90-series, input devices, hybrid ICs

C2 **Television tuners, video modulators, surface acoustic wave filters**

C3 **Loudspeakers**

C4 **Ferroxcube potcores, square cores and cross cores**

C5 **Ferroxcube for power, audio/video and accelerators**

C6 **Electric motors and accessories**
Permanent magnet synchronous motors, stepping motors, direct current motors

C7 **Variable capacitors**

C8 **Variable mains transformers**

C9 **Piezoelectric quartz devices**
Quartz crystal units, temperature compensated crystal oscillators, compact integrated oscillators, quartz crystal cuts for temperature measurements

C10 **Connectors**

C11 **Non-linear resistors**
Voltage dependent resistors (VDR), light dependent resistors (LDR), negative temperature coefficient thermistors (NTC), positive temperature coefficient thermistors (PTC)

C12 **Variable resistors and test switches**

C13 **Fixed resistors**

C14 **Electrolytic and solid capacitors**

C15 **Film capacitors, ceramic capacitors**

C16 **Piezoelectric ceramics, permanent magnet materials**

January 1983

GENERAL CONTENTS

Preface
Contents
Ordering information
Product status definitions

PREFACE

The continuing trend of system integration has created new challenges for the design engineer. He must strive to consolidate higher complexity, more feature laden circuits into his designs without sacrificing flexibility. Today's competitive electronic marketplace has created the need for logic devices which can provide cost effective methods of reducing random logic requirements and interface with fixed LSI custom logic, yet retain the flexible features required to modify the system prior to production. Signetics has developed IFL (Integrated Fuse Logic) as a programmable logic family to address that need.

The IFL family consists of a relatively few devices which are designed to address logic needs from random gates in the case of the Field Programmable Gate Arrays to highly sophisticated state machines in the case of the Field Programmable Logic Sequencers. Signetics has pioneered since 1975 the concept of total AND/OR/INVERT programmability. IFL can encompass these wide ranging levels of integration without the necessity of providing a multitude of devices; each dedicated to a specific application or I/O structure. The flexible programming structure allows the designer to "mold" the individual device's architecture to the complete range of requirements typically found in system design.

Since the production process utilized to manufacture IFL is identical to the process used on the Signetics Bipolar PROM Family, it is possible to provide a highly manufacturable, high performance, cost effective product family. The 1983 IFL Data Manual contains information the designer will require in order to effectively utilize IFL. For information on the Signetics PROM Family refer to the Signetics 1982 Bipolar Memory Data Manual. For the convenience of the designer, a Product Selection Guide has been included in this publication.

Integrated Fuse Logic Marketing

Except for the application section, this handbook contains the Signetics Integrated Fuse Logic Data manual issued January 1983.

CONTENTS

		page
Ordering information		5
Product status definitions		6
Bipolar memory selection guide		9
Integrated fuse logic		
Introduction		13
IFL data sheets Series 20		
82S150/151	Field Programmable Gate Array (FPGA) (18 x 15 x 12)	47
82S152/153	Field Programmable Logic Array (FPLA) (18 x 42 x 10)	53
82S152A/153A	Field Programmable Logic Array (FPLA) (18 x 42 x 10)	53
82S154/155	Field Programmable Logic Sequencer (FPLS) (16 x 45 x 12) 4-bit register	61
82S156/157	Field Programmable Logic Sequencer (FPLS) (16 x 45 x 12) 6-bit register	61
82S158/159	Field Programmable Logic Sequencer (FPLS) (16 x 45 x 12) 8-bit register	61
IFL data sheets Series 28		
82S100/101	Field Programmable Logic Array (FPLA) (16 x 48 x 8)	85
82S102/103	Field Programmable Gate Array (FPGA) (16 x 9 x 9)	91
82S104/105	Field Programmable Logic Sequencer (FPLS) (16 x 48 x 8)	97
82S104A/105A	Field Programmable Logic Sequencer (FPLS) (16 x 48 x 8)	107
IFL programming information		
82S100/101	Field Programmable Logic Array (FPLA) (16 x 48 x 8)	119
82S102/103	Field Programmable Gate Array (FPGA) (16 x 9 x 9)	123
82S104/105	Field Programmable Logic Sequencer (FPLS) (16 x 48 x 8)	125
20 pin series	Field Programmable Logic	129
82S152/153	Field Programmable Logic Array (FPLA) (18 x 42 x 10)	131
82S158/159	Field Programmable Logic Sequencer (FPLS) (16 x 45 x 12)	133
Military products		
Military products/process levels		137
Military products selection guides		
Logic		141
Memory		146
Linear		148
Military products linear industry cross reference		149

CONTENTS (continued)

page

Package outlines

General information 153

Packages
- N: plastic dual in-line 155
- F: hermetic cerdip 156
- G: hermetic leadless chip carriers 157

Application section IFL handbook

Introduction and contents 161

New developments in integrated fuse logic 163

LOGMIN users manual 177

Field-programmable arrays
- part 1: powerful alternatives to random logic 181
- part 2: sequencers and arrays transform truth tables into working systems 187

Controllers using FPLAs and chapter contents 195

Single-chip multiprocessor arbiter using the Signetics 82S105 FPLS 223

Expandable memory system using the Signetics field programmable gate array 231

8-bit parallel CRC generator/checker using field programmable logic 235

A simple keyboard encoder using the FPLS 239

The FPLA adds display flexibility to CRT controllers 245

Field programmable logic arrays 251
- Table of contents (chapter) 252

Programming services bipolar memory/LSI products 361

ORDERING INFORMATION

Signetics Bipolar Memory integrated circuit products may be ordered by contacting either the local Signetics sales office, Signetics representatives and/or Signetics authorized distributors. A complete listing is located in the back of this manual.

The table shown provides part number definition for Signetics memory products. The Signetics part number system allows complete definition for ordering a device. The part number itself and the product description is defined on each data sheet. The suffix is a single letter defining a package type (as shown in the table on this page). Additional or special processing is defined by adding the processing indicator when required.

The military qualification, Full MIL Signetics or Full JAN slash sheet status, can be determined by referring to the military section of this manual. Verification of the military status can be made by contacting your local Signetics sales office.

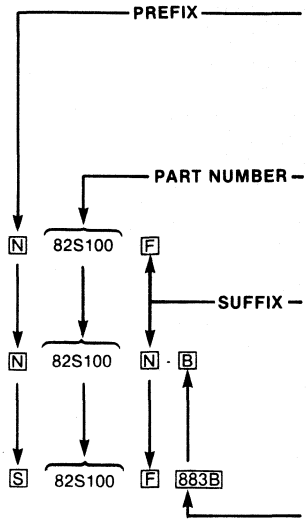


Table 1 PART NUMBER DESCRIPTION

USED TO DEFINE OPERATING TEMPERATURE RANGE	
PREFIX	**DEVICE TEMPERATURE RANGE**
N	0°C to +75°C
S	−55°C to +125°C

USED TO DEFINE PART ITSELF

USED TO DEFINE PACKAGE TYPE	
PACKAGE CODE	**PACKAGE DESCRIPTION**
F	20, 24, 28-Lead DIL CERDIP
G	20, 28-Lead Square Leadless
R	28-Lead CERAMIC-Beryllia Flat Pack
N	20, 24, 28-Lead DIL PLASTIC

PROCESSING INDICATOR

MILITARY		
883C		Full MIL-STD-883 Class C Processing
883B		Full MIL-STD-883 Class B Processing including 168 hrs. Burn-In
Commercial		
A	SUPR II A	Tightened AQLs
B	SUPR II B	Tightened AQLs with Operating High Temp. Burn-In

DEFINITION OF TERMS

Data Sheet Identification	Product Status	Definition
Preview	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Advance Information	Sampling or Pre-Production	This data sheet contains advance information and specifications are subject to change without notice.
Preliminary	First Production	This data sheet contains preliminary data and supplementary data will be published at a later date. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

SELECTION GUIDE

SELECTION GUIDE

DEVICE	ORGANIZATION	OUTPUT CIRCUIT[1]	OUTPUT LOGIC[2]	ACCESS TIME[4]	TEMPERATURE[3]	PACKAGE[5]	PINS	MAX I_{CC}
CAMs								
10155	8 x 2	OE	—	13	C	F, N	18	140
RAMs								
3101A	16 x 4	OC	B	35	C	F, N	16	105
54/74S189	16 x 4	TS	B	35	M, C	F, N	16	110
82S21	32 x 2	OC	T	50	C	F, N	16	130
82S16	256 x 1	TS	T	50	M, C	F, N	16	115
82S17	256 x 1	OC	T	50	M, C	F, N	16	115
74S301	256 x 1	OC	B	50	M, C	F, N	16	115
82LS16	256 x 1	TS	T	40	M, C	F, N	16	70
82LS17	256 x 1	OC	T	40	M, C	F, N	16	70
74LS301	256 x 1	OC	B	40	M, C	F, N	16	70
82S09	64 x 9	OC	T	45	M, C	F, N	28	190
82S09A	64 x 9	OC	T	35	C	F, N	28	190
82S19	64 x 9	OC	B	35	M, C	F, N	28	190
82S210	256 x 9	TS	B	60	M, C	F, N	24	185
82S212	256 x 9	TS	B	45	M, C	F, N	22	185
82S212A	256 x 9	TS	B	35	C	F, N	22	185
8X350	256 x 8	TS	B	N/A	M, C	F, N	22	185
ECL								
10415	1024 x 1	OE	B	20	C	F	16	150
10415A	1024 x 1	OE	B	15	C	F	16	150
10415B	1024 x 1	OE	B	10	C	F	16	150
100415	1024 x 1	OE	B	20	C	F	16	150
100415A	1024 x 1	OE	B	15	C	F	16	150
100415B	1024 x 1	OE	B	10	C	F	16	150
100422	256 x 4	OE	B	20	C	F	24	210
10422A	256 x 4	OE	B	15	C	F	24	210
10422B	256 x 4	OE	B	10	C	F	24	210
10470*	4096 x 1	OE	B	25	C	F	18	150
10470A*	4096 x 1	OE	B	15	C	F	18	150
100470*	4096 x 1	OE	B	25	C	F	18	150
100470A*	4096 x 1	OE	B	15	C	F	18	150
10474*	1024 x 4	OE	B	30	C	F	24	195
10474A*	1024 x 4	OE	B	20	C	F	24	195
100474*	1024 x 4	OE	B	30	C	F	24	195
100474A*	1024 x 4	OE	B	20	C	F	24	195
FPLAs								
82S100	16 x 48 x 8	TS	—	50	M, C	F, N	28	170
82S101	16 x 48 x 8	OC	—	50	M, C	F, N	28	170
82S152	18 x 32 x 10	OC	I/O	40	M, C	F, N	20	155
82S152A	18 x 32 x 10	OC	I/O	30	M, C	F, N	20	155
82S153	18 x 32 x 10	TS	I/O	40	M, C	F, N	20	155
82S153A	18 x 32 x 10	OC	I/O	30	M, C	F, N	20	155
FPGAs								
82S150*	18 x 12	OC	I/O	20	M, C	F, N	20	155
82S151*	18 x 12	TS	I/O	20	M, C	F, N	20	155
82S102	16 x 9	OC	—	35	M, C	F, N	28	170
82S103	16 x 9	TS	—	35	M, C	F, N	28	170
FPLSs								
82S104	16 x 48 x 8	OC	R	60	M, C	F, N	28	180
82S104A	16 x 48 x 8	OC	R	50	M, C	F, N	28	180
82S105	16 x 48 x 8	TS	R	60	M, C	F, N	28	180
82S105A	16 x 48 x 8	TS	R	50	M, C	F, N	28	180
82S154*	16 x 32 x 12	OC	I/O, R	65	M, C	F, N	20	155
82S155*	16 x 32 x 12	TS	I/O, R	65	M, C	F, N	20	155
82S156*	16 x 32 x 12	OC	I/O, R	65	M, C	F, N	20	155
82S157*	16 x 32 x 12	TS	I/O, R	65	M, C	F, N	20	155
82S158	16 x 32 x 12	OC	I/O, R	65	M, C	F, N	20	155
82S159	16 x 32 x 12	TS	I/O, R	65	M, C	F, N	20	155

NOTES on following page

SELECTION GUIDE

DEVICE	ORGANIZATION	OUTPUT CIRCUIT[1]	OUTPUT LOGIC[2]	ACCESS TIME[4]	TEMPERATURE[3]	PACKAGE[5]	PINS	MAX. I_{CC}
PROMs								
82S23	32 x 8	OC	—	50	M, C	F, N	16	77
82S23A*	32 x 8	OC	—	25	M, C	F, N	16	100
82S123	32 x 8	TS	—	50	M, C	F, N	16	77
82S123A*	32 x 8	TS	—	25	M, C	F, N	16	100
82S126	256 x 4	OC	—	50	M, C	F, N	16	120
82S126A*	256 x 4	OC	—	35	M, C	F, N	16	120
82S129	256 x 4	TS	—	50	M, C	F, N	16	120
82S129A*	256 x 4	TS	—	35	M, C	F, N	16	120
10149	256 x 4	OE	—	20	C	F	16	150
100149	256 x 4	OE	—	20	C	F	16	150
82S130	512 x 4	OC	—	50	M, C	F, N	16	140
82S130A*	512 x 4	OC	—	35	M, C	F, N	16	140
82S131	512 x 4	TS	—	50	M, C	F, N	16	140
82S131A*	512 x 4	TS	—	35	M, C	F, N	16	140
82S115	512 x 8	TS	—	60	M, C	F, N	24	175
82S140	512 x 8	OC	—	60	M, C	F, N	24	175
82S141	512 x 8	TS	—	60	M, C	F, N	24	175
82S137	1024 x 4	TS	—	60	M, C	F, N	18	140
82S137A	1024 x 4	TS	—	45	M, C	F, N	18	140
82S137B	1024 x 4	TS	—	35	C	F, N	18	140
82S147	512 x 8	TS	—	60	M, C	F, N	16	155
82S147A	512 x 8	TS	—	45	M, C	F, N	20	155
82LS181	1024 x 8	TS	—	150	M, C	F, N	24	80
82S180	1024 x 8	OC	—	70	M, C	F, N	24	175
82S181	1024 x 8	TS	—	70	M, C	F, N	24	175
82S181A	1024 x 8	TS	—	50	M, C	F, N	24	175
82S181B	1024 x 8	TS	—	45	C	F, N	24	175
82S183	1024 x 8	TS	—	60	M, C	F, N	24	175
82S2708	1024 x 8	TS	—	90	M	F, R, G	24	185
82S185	2048 x 4	TS	—	100	M, C	F, N	18	120
82S185A	2048 x 4	TS	—	50	M, C	F, N	18	155
82S185B	2048 x 4	TS	—	45	C	F, N	18	155
82S191	2048 x 8	TS	—	80	M, C	F, N	24	175
82S191A	2048 x 8	TS	—	60	M, C	F, N	24	175
82HS195*	4096 x 4	TS	—	35	M, C	F, N	20	155
82S321	4096 x 8	TS	—	80	M, C	F, N	24	175
82HS321*	4096 x 8	TS	—	40	M, C	F, N	24	175
82HS641*	4096 x 8	TS	—	45	M, C	F, N	24	175

*To be announced.

NOTES:

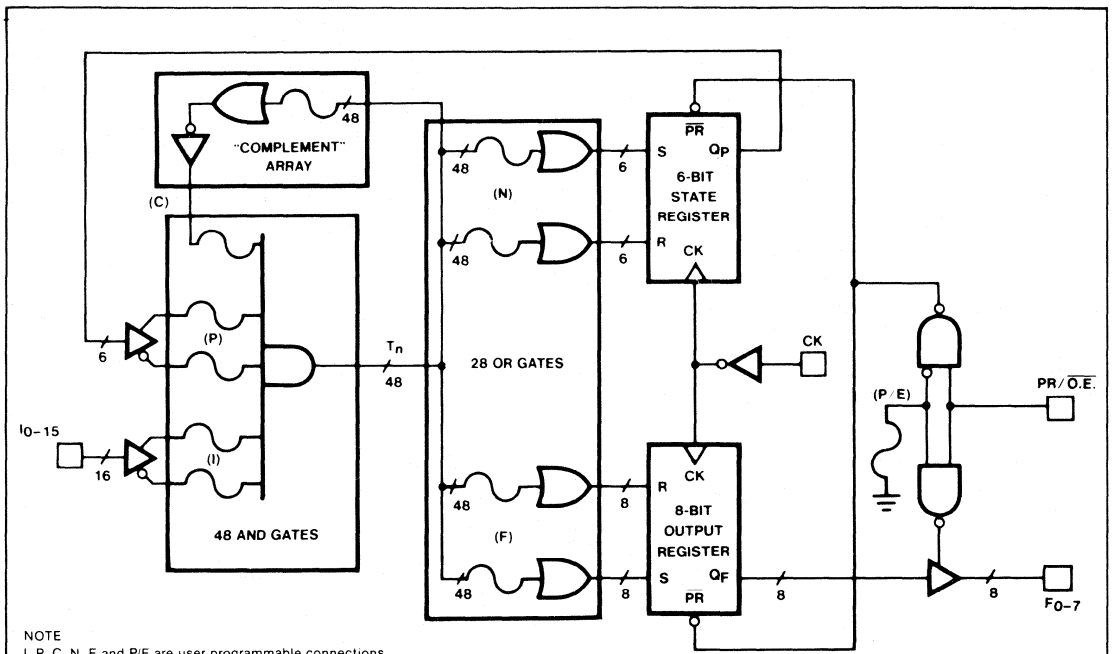
1. Output circuit:
   OE = Open emitter
   OC = Open collector
   TS = 3-State
2. Output logic:
   T = Transparent—input data appears on output during Write
   B = Blanked—output is blanked during Write
   R = Output logic
   I/O = Input/output option
3. Temperature range:
   C = Commercial (0°C to +75°C)
   M = Military (−55°C to +125°C)
4. Commercial (0°C to +75°C)
5. Packages:
   F = Hermetic Cerdip Dual In Line
   N = Plastic Dual In Line
   R = Ceramic Flat Pack
   G = Ceramic Square Leadless Chip Carrier

INTRODUCTION

WHAT IS INTEGRATED FUSE LOGIC?

In 1975, Signetics Corporation developed a new product family by combining its expertise in semi-custom gate array products and fuse-link Programmable Read Only Memories (PROM). Out of this marriage came Integrated Fuse Logic (IFL). The 82S100 Field Programmable Logic Array (FPLA) was the first member of this family. The FPLA was an important industry first in two ways. First the AND/OR/INVERT architecture allowed the custom implementations of Sum of Product logic equations. Second, the three level fusing allows complete flexibility in the use of this device family. All logic interconnections from input to output are programmable.

Development of this family did not stop with the 82S100. In 1977, the 82S103, Field Programmable Gate Array (FPGA), and the 82S107, Field Programmable ROM Patch (FPRP) were introduced. The 82S105, Field Programmable Logic Sequencer (FPLS) was announced in 1979. This device represents a significant step forward for IFL. The FPLS is a fully implemented Mealy State Machine on a chip. Incorporated into its architecture are 48 T-Terms, an 8-bit output register, and a 6-bit internal state register.



NOTE
I, P, C, N, F and P/E are user programmable connections.

Figure A. FPLS

The FPLS can synchronously perform sequential routines at 20MHz. All of these products are now known as the IFL Series 28, Programmable Logic Family.

20-PIN		SERIES 20
PART NUMBER	**TYPE**	**CONFIGURATION**
82S150/151	FPGA	18 Input/12 Output
82S152/153	FPLA	18 Input-10 Output-42 Term
82S154-159	FPLS	16 input-12 Output-45 Term
154/155	FPLS	4 Registered Outputs
156/157	FPLS	6 Registered Outputs
158/159	FPLS	8 Registered Outputs
28-PIN		**SERIES 28**
82S100/101	FPLA	16 Input-8 Output-48 Term
82S102/103	FPGA	16 Input-9 Output-
82S104/105	FPLS	16 Input-8 Output-48 Term
		6 Bit State Register
		8 Output Registers

Figure B. IFL Product Family

Signetics latest innovation in this area is the Series 20 Programmable Logic Family. All members of this family are contained in 20-pin packages. While reducing the number of pins in this family Signetics has utilized controlled I/O in order to maintain the utility of Series 20. Appropriate control terms have been included to allow active control of pin direction.

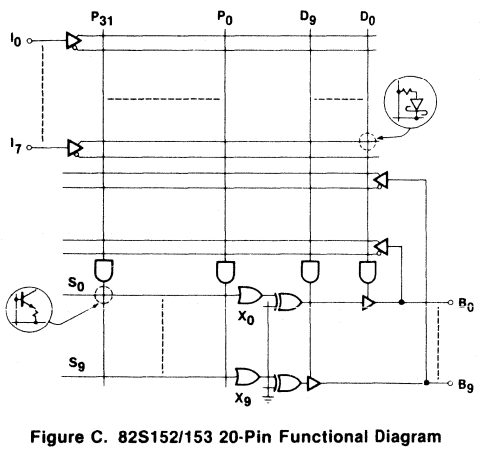
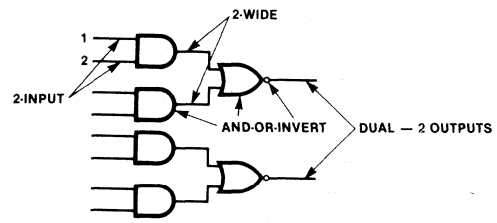


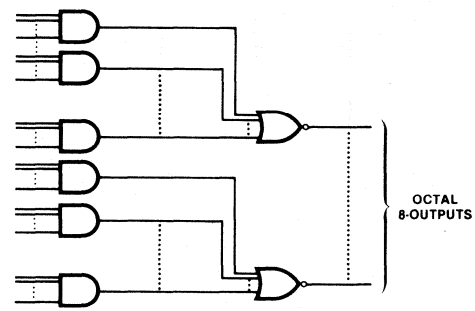
Figure C. 82S152/153 20-Pin Functional Diagram

PRODUCT DESCRIPTION

While all IFL devices are architecturally similar, the FPLA is most representative of the concepts involved. The FPLA is an AND/OR/INVERT device, with all internal interconnections programmable via fuse links. If we compare the 82S100 FPLS to the familiar 7450 AOI gate the similarity of function becomes apparent.



7450
2 x 2 x 2
FIXED INTERCONNECTIONS
DUAL, 2-WIDE, 2-INPUT, AND-OR-INVERT



82S100 16 x 48 x 8
OCTAL, 48-WIDE, 16-INPUT, AND-OR-INVERT

Figure 1

Package-Gate Replacement Potential

All Signetics IFL devices are capable of replacing multiple discrete logic devices. The number of packages is dependent on the application and the device used.

If all product terms and inputs on the 82S100 are utilized a total of 272 gates and 140 I.C.s can theoretically be replaced. Since the 82S100 is made up of logic structures which are not available as commercial logic devices; such as 32 input AND gates and 48 input OR gates, these numbers are based on breakdowns of these high complexity logic structures into structures which are available as discrete logic.

Table 1 GATE REPLACEMENT (*All* Terms Used)

	Gates	I.C.s
Each P-Term = 2 8-Input AND Gates & 1 2-Input AND Gate =	144	108
OR Matrix = 16 4-Input OR Gates =	128	32
	272	140

The typical application does not fully utilize the FPLA however. We can assume that the typical application is as defined in Table 2.

Table 2 TYPICAL APPLICATION

4 Variables/P-Term	40 Gates
4 P-Terms/Output	15-20 I.C.s

The replacement of 15-20 I.C.s with one 82S100 is not unusual.

IFL ECONOMICS

The reason any product line exists is because it solves a problem in a cost effective manner.

IFL is no different in this respect. Let's assume that one 82S100 is replacing 15 TTL logic packages.

If we assume that the average selling price of an SSI level device is $0.25, then we have a basis for comparision.

15 I.C.s @ $0.25 = $3.75 1 82S100 FPLA = $7.00

Considering piece part cost alone does not fare well for the 82S100 FPLA. *But,* part cost is one of the smaller costs in systems manufacturing. Let's next consider PC board costs.

PC BOARD

The purchase price of a PCB is calculated on a per square inch basis. Typical costs at the 5K level are $0.23/in^2, the cost for LS' is 1.2 in^2 = $0.23/in^2 = $0.28. For a 28-pin FPLA device, the packing density is 1-82S100 per 2.5 in^2, giving a cost of 2.5 in^2 x $0.23/in^2 = $0.58.

PCB FABRICATION:

Now let's consider the cost of building the board. The primary cost for PCB fabrication is parts preparation and stuffing.

Estimates by PCB manufacturers indicate a nominal cost of $0.14 per device for LS TTL and $0.16 for 28-pin FPLA. There is also a basic cost of $0.02 per in^2 for handling, soldering, and inspecting.

The cost of incoming inspection and inventory/usage must also be considered in any fair comparison.

INCOMING INSPECTION

The cost of incoming inspection can be calculated on a per-pin basis. Assuming an equal mix of 14 and 16-pin SSI parts and a testing cost of $0.01 per pin, the LS TTL cost is $0.15 per IC. The corresponding cost for the 82S100 would be $0.01 x 28 or $0.28 total.

COMPONENT INVENTORY/USAGE:

The inventory cost for LS TTL parts is dominated by inventory maintenance, while the LSI type parts, the piece price dominates. Assuming a 2% usage rate due to parts breakage, burnout, etc., in the manufacturing process, the parts cost for LS TTL is 2% x $0.25 = $0.005. Adding $0.02 per device for inventory maintenance, the total LS cost $0.03. For FPLA, 2% = $7.00 + $0.02 = $0.16.

Consideration must also be given to testing the completed PCB and Rework costs.

PCB TESTING

Testing costs are generally figured on a PCB basis and are usually go-no-go tests using a "bed of nails". For this comparison, assume a PCB of 5" x 7". The cost for testing the completed PCB is estimated to be $3.60 for 20 I.C.s. This breaks down as $0.18 per LS TTL device or about $0.45 per FPLA device.

PCB REWORK

The main causes of rework are poorly plated holes, solder bridges, and parts inserted incorrectly. The "bed of nails" test can usually detect all of the faults. Rework costs about 50% of test costs. The result is $1.80 per PCB or $0.09 per device for LS TTL and $0.18 for the 28-pin FPLA.

System manufacturing costs are harder to define. For this comparison we estimated $0.28 for each LS device and about $0.65 for the FPLA. These costs include backplane wiring, final system assembly and test costs.

Now using these figures, the following comparison can be made:

COST ANALYSIS

	LS TTL	FPLA
IC COMPONENTS	.25	7.00
PCB MANUFACTURING		
Incoming Inspection	.15	.28
Component Inventory/Usage	.03	.16
PC Board	.28	.58
Fabrication	.14	.16
Test	.18	.45
Rework	.09	.18
ASSEMBLY LABOR	.28	.65
COST PER COMPONENT	1.40	9.46
NUMBER OF COMPONENTS	x 15	x 1
TOTAL	21.00	9.46

CONCLUSION:
THE FPLA is a 55% savings over the LS TTL DESIGN.

IFL-LOGIC SYNTHESIS

28-PIN

IFL is capable of logic Synthesis. No intermediate step is required to implement Boolean Logic Equations with IFL. Each term in each equation simply becomes a direct entry into the Logic Program Table. The following example illustrates this straightforward concept.

$X_0 = AB + \overline{C}D + B\overline{D}$

$\overline{X}_1 = \overline{A}B + \overline{C}D + EFG$

BIPOLAR FIELD PROGRAMMABLE LOGIC ARRAY (16×48×8) 82S100 (O.C.)/82S101 (T.S.)

INTEGRATED FUSE LOGIC
SERIES 28

FPLA PROGRAM TABLE (LOGIC)

P-Terms
$P_0 = A\overline{B}$
$P_1 = \overline{C}\overline{D}$
$P_2 = B\overline{D}$
$P_3 = \overline{A}B$
$P_4 = EFG$

$X_0 = P_0 + P_1 + P_2$
$\overline{X}_1 = P_3 + P_1 + P_4$

	AND: INPUT (I_m)																POLARITY							
																					L		H	
																	OR							
TERM	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT F_P 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	H	H	–	–	–	–	–							•	A
1	–	–	–	–	–	–	–	–	–	–	–	L	H	–	–	–							A	A
2	–	–	–	–	–	–	–	–	–	–	H	–	L	–	–	–							•	A
3	–	–	–	–	–	–	–	–	–	L	H	–	–	–	–	–							A	•
4	–	–	–	–	–	–	–	–	–	–	–	–	–	H	H	H							A	•
44																								
45																								
46																								
47																								
PIN NO.	20	21	22	23	24	25	26	27	2	3	4	5	6	7	8	9	10	11	12	13	15	16	17	18
VARIABLE NAME										A	B	C	D	E	F	G							X_1	X_0

Figure 2. Bipolar Field Programmable Logic Array (16 x 48 x 8) 82S100 (O.C.)/82S101 (T.S.)

Figure 3. Equivalent Fixed Logic Diagram

In the previous example the two Boolean Logic equations were broken into Product Terms. Each P-Term was then programmed into the P-term section of the FPLA Program Table. This was accomplished in the following manner:

Step 1

Select which input pins I_0-I_{15} will correspond to the input variables. In this case A-G are the input variable names. I_6 through I_0 were selected to accept inputs A-G respectively.

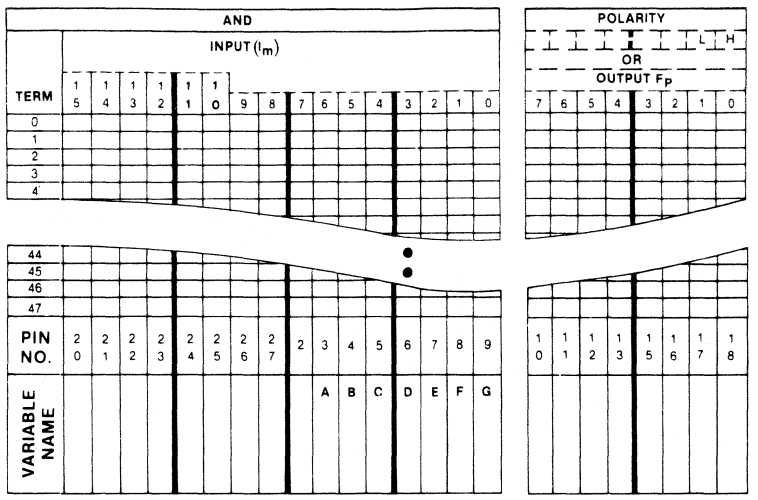


Figure 4

Step 2

Transfer the Boolean Terms to the FPLA Program Table. This is done simply by defining each term and entering it on the Program Table.

e.g., $P_0 = AB$

This P-term translates to the Program Table by selecting $A = I_6 = H$ and $B = I_5 = H$ and entering the information in the appropriate column.

$$P_1 = \overline{C}D$$

This term is defined by selecting $C = I_4 = L$ and $D = I_3 = H$, and entering the data into the Program Table. Continue this operation until all P-terms are entered into the Program Table.

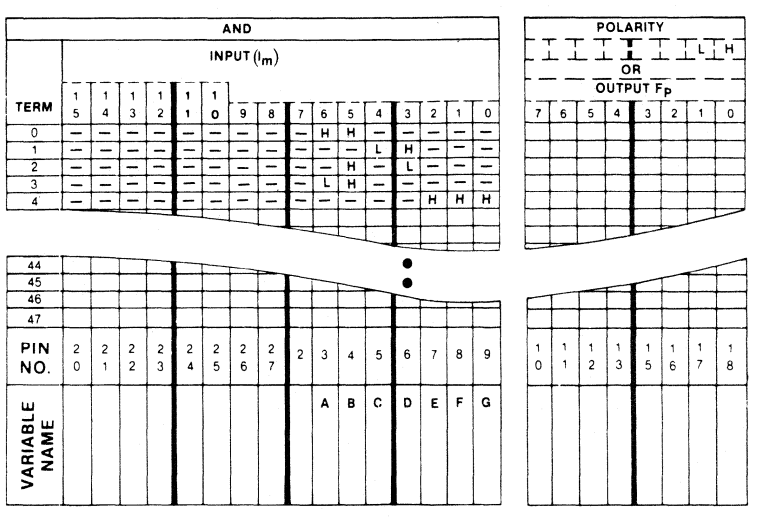


Figure 5

Step 3
Select which output pins correspond to each output function. In this case F_0 = Pin 18 = X_0, and F_1 = Pin 17 = X_1.

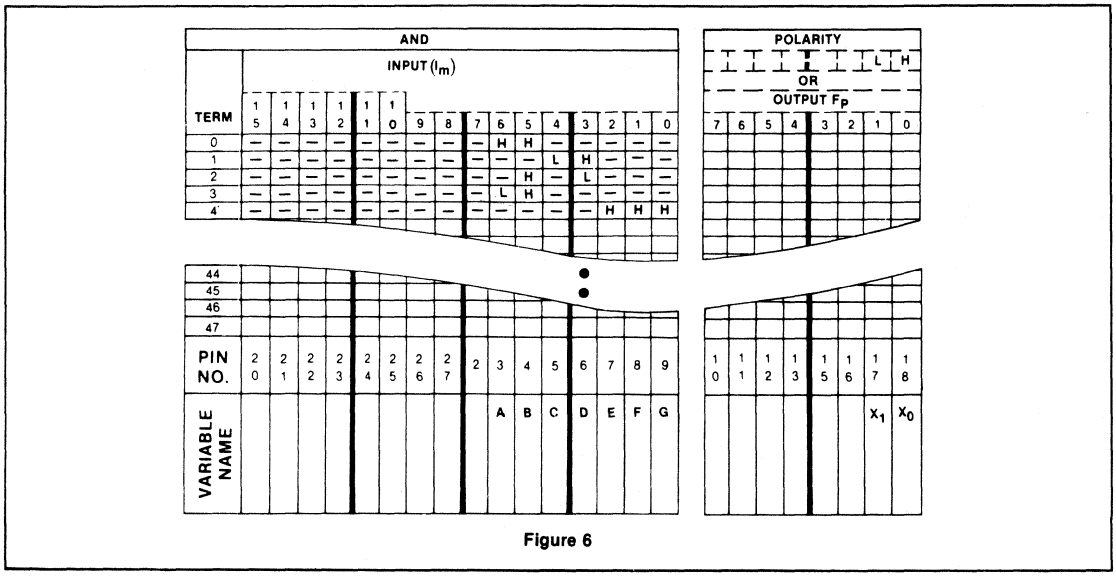


Figure 6

Step 4
Select the Output Active Level desired for each Output Function. For X_0 the active level is high for a positive logic expression of this equation. Therefore it is only necessary to place an (H) in the Active Level box above Output Function 0, (F_0). Conversely, X_1 can be expressed as X_1 by placing an (L) in the Active Level box above Output Function 1, (F_1).

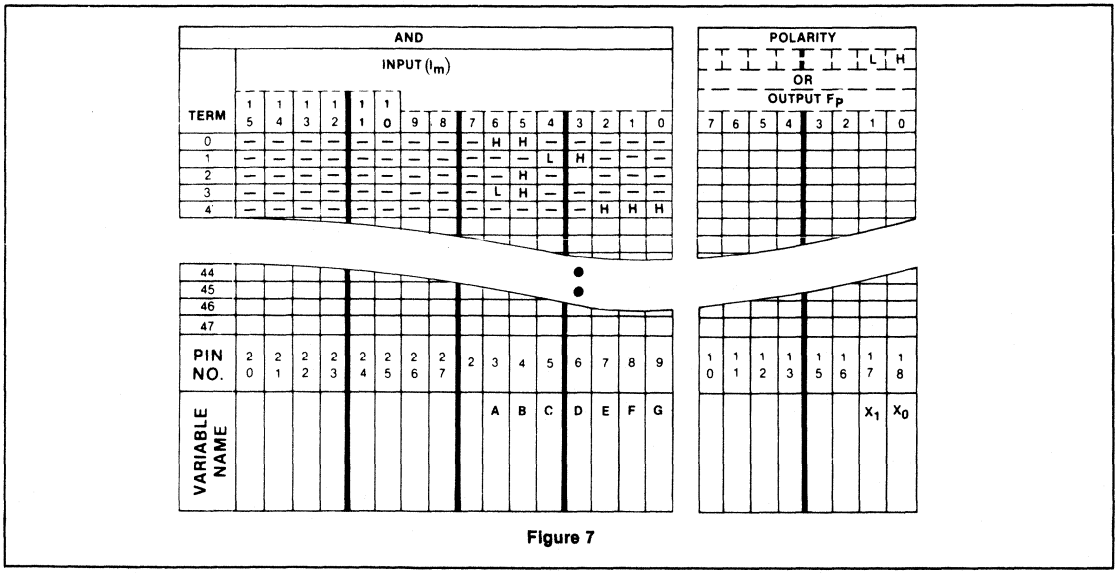


Figure 7

Step 5

Select the P-Terms you wish to make active for each Output Function. In this case $X_0 = P_0 + P_1 + P_2$, so an A has been placed in the intersection box for P_0 and X_0, P_1 and X_0 and P_2 and X_0.

Terms which are not active for a given output are made inactive by placing a (•) in the box under that P-term. Leave all unused P-terms unprogrammed.

Continue this operation until all outputs have been defined in the Program Table.

Step 6

Enter the data into a Signetics approved programmer. The input format is identical to the Signetics Program Table. You specify the P-Terms, Output Active Level, and which P-Terms are active for each output exactly the way it appears on the Program Table.

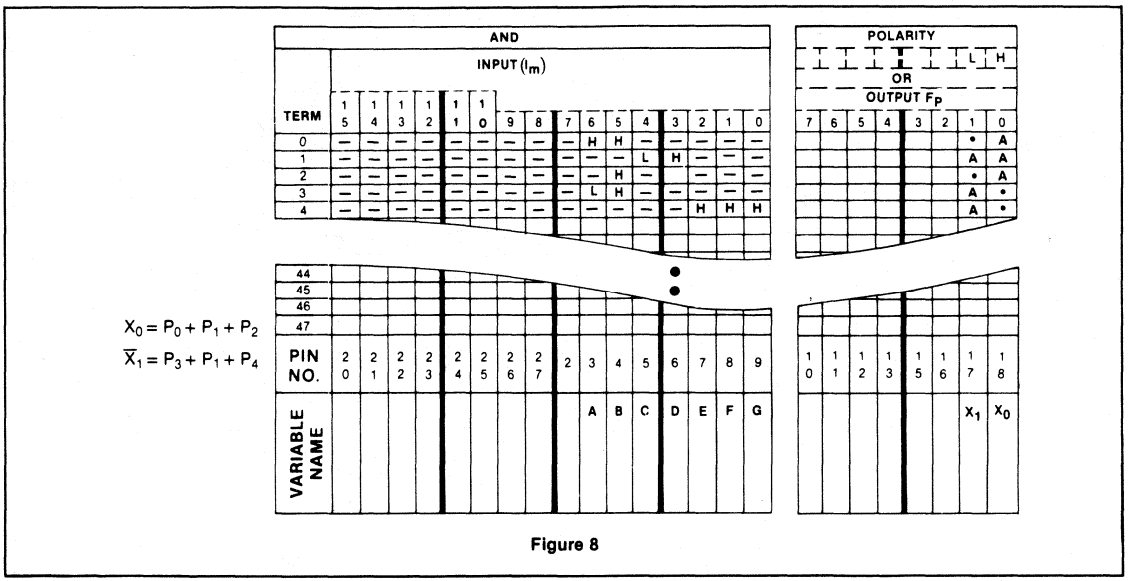


Figure 8

IFL LOGIC SYNTHESIS

20-PIN

When fewer inputs and outputs are required in a logic design and low cost is most important, the Signetics 20-pin IFL should be considered first choice. The 82S153 is an FPLA with 8 inputs, 10 I/O pins, and 42 product terms. The user can configure the device by defining the direction of the I/O pins. This is easily accomplished by using the direction control terms D_0-D_9 to establish the direction of pins B_0-B_9. The D-terms control the tri-state buffers found on the outputs of the EX-OR gates. Figures 10 and 11 show how the D-term configures each B_X pin.

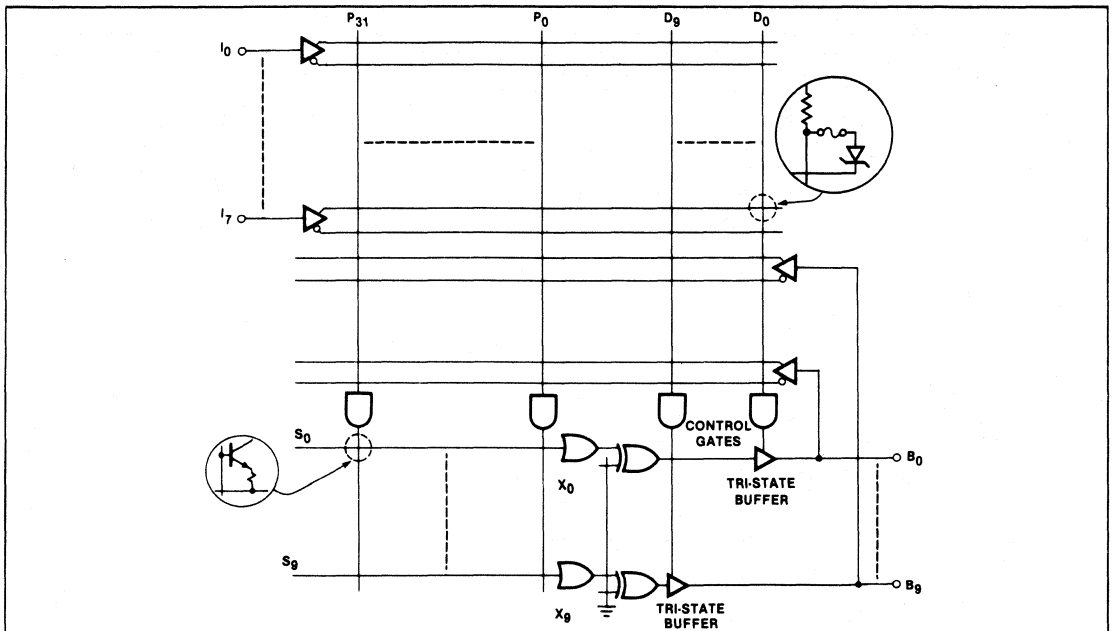
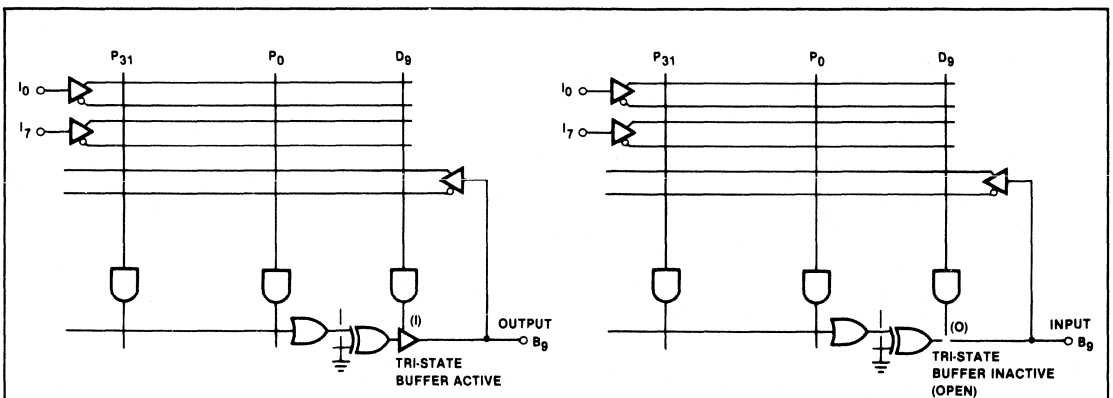


Figure 10. 82S153 Functional Diagram



a) D_9 Active Makes B_9 Appear as an Output With Feedback

b) D_9 Inactive Makes B_9 Appear as an Input

Figure 11.

To control each D-term, it is necessary to understand that each control gate is a 36-input AND gate. To make the Tri-State buffer active (B_X pin an output), the output of the control gate must be at logic HIGH (1). This can be accomplished in one of two ways. A HIGH can be forced on all control gate input nodes or fuses can be programmed. When a fuse is programmed that control gate input node is internally pulled up to HIGH (1). See Figure 12 and Figure 13.

Programming the fuse permanently places a HIGH (1) on the input to the control gate. The input pin no longer has any effect on that state.

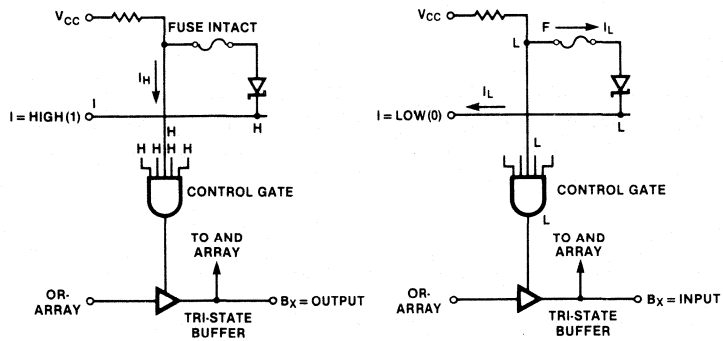


Figure 12. Input Effect on Control Gates (Fuse Intact)

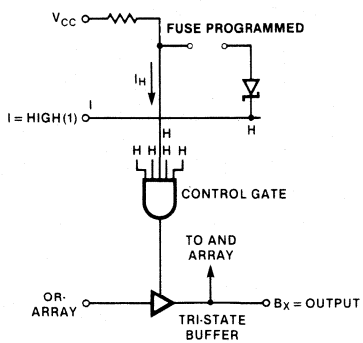


Figure 13. Effect on Control Gate if Fuse is Programmed

DEDICATING B_X PIN DIRECTION

Since each input to the D-terms is true and complement buffered (see Figure 11), when the device is shipped with all fuses intact, all control gates have half of the 36 input lines at logic low (0). The result of this is all Control Gate outputs are low (0) and the tri-state buffers are inactive. This results in all B_X pins being in the input condition. The resultant device is therefore an 18-input, 0-output FPLA. While useful as a bit bucket or Write-Only-Memory (WOM) most applications require at least one output. Clearly, the first task is to determine which of the B_X pins are to be outputs. The next step is to condition the control gate to make the tri-state buffer for those gates active. To dedicate B_0 and B_1 as outputs it is necessary to program all fuses to the inputs to Control Gates D_0 and D_1. This internally pulls all inputs to those gates to HIGH (1) permanently. Since all inputs to the Control Gates are HIGH (1), the output is HIGH (1) and the tri-state buffers for B_0 and B_1 are active. This permanently enables B_0 and B_1 as outputs. Note that even though B_0 and B_1 are outputs, the output data is available to the AND array and via the internal feedback (see Figure 11a).

To program this data, the 82S153 Program Table is used as follows:

FPLA PROGRAM TABLE

BIPOLAR FIELD PROGRAMMABLE LOGIC ARRAY (18×42×10) 82S152 (O.C.)/82S153 (T.S.)

INTEGRATED FUSE LOGIC
SERIES 20

PROGRAM TABLE ENTRIES

OR — B (O): A — ACTIVE; • — INACTIVE | CONTROL (POL.): H — HIGH; L — LOW

AND — I, B (I): U — INACTIVE; H — I, B; L — I, B; — — DON'T CARE

TERM	AND I 7	6	5	4	3	2	1	0	AND B(I) 9	8	7	6	5	4	3	2	1	0	POLARITY OR B(O) 9	8	7	6	5	4	3	2	1	0
0																												
1																												
2																												
3																												
4																												
•																												
•																												
•																												
30																												
31																												
D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–										
D0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–										
PIN	8	7	6	5	4	3	2	1	19	18	17	16	15	14	13	12	11	9										
VARIABLE NAME																												

NOTES
1. The FPLA is shipped with all links intact. Thus a background of entries corresponding to states of virgin links exists in the table, shown BLANK for clarity.
2. Unused I and B bits in the AND array are normally programmed Don't Care (–).
3. Unused product terms can be left blank.

Figure 14. Dedicating B_0 and B_1 as Outputs and B_2 through B_9 as Inputs

By placing a (—) don't care in each input box you are specifying that the true and complement fuses are programmed on each Control Gate thus permanently dedicating the B_0 and B_1 pins as outputs. By placing a (0) in all input boxes for B_2-B_9, you are specifying that both true and complement fuses are intact. This causes a low (0) to be forced on half of the Control Gate inputs, guaranteeing the output of the Control Gate will be low (0). When the Control Gate outputs are low (0), the tri-state buffer is inactive and the B_2-B_9 pins are enabled as inputs. All B_X pin directions can be controlled in this manner.

ACTIVE DIRECTION CONTROL

Sometimes it is necessary to be able to actively change the direction of the B_X pins without permanently dedicating them. Some applications which require this include tri-state bus enable, multi-function decoding, etc. This can easily be done by programming the control gate to respond to one or more input pins. It is only necessary to select which I_X and B_X pins will control the pin directions and the active level HIGH (H) or LOW (L) that will be used. The 82S153 Program Table in Figure 15 shows the method of controlling B_0-B_9 with I_7. When I_7 is LOW (L), pins B_0-B_9 are outputs when I_7 is HIGH (H), pins B_0-B_9 are inputs. Note that by programming all other I_X and B_X pins as DON'T CARE (—) they are permanently disconnected from control of B_X pin direction.

FPLA PROGRAM TABLE

BIPOLAR FIELD PROGRAMMABLE LOGIC ARRAY (18×42×10) 82S152 (O.C.)/82S153 (T.S.)

INTEGRATED FUSE LOGIC
SERIES 20

FPLA PROGRAM TABLE (LOGIC)

PROGRAM TABLE ENTRIES

OR: B(0) A = ACTIVE, • = INACTIVE; CONTROL (POL.) H = HIGH, L = LOW

AND: I, B(I) 0 = INACTIVE, H = I, B, L = I, B, — = DON'T CARE

TERM	AND I 7	6	5	4	3	2	1	0	AND B(I) 9	8	7	6	5	4	3	2	1	0	POLARITY OR B(0) 9	8	7	6	5	4	3	2	1	0
0																												
1																												
2																												
3																												
4																												
•																												
•																												
•																												
30																												
31																												
D9	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D8	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D7	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D6	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D5	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D4	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D3	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D2	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D1	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
D0	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—										
PIN	8	7	6	5	4	3	2	1	19	18	17	16	15	14	13	12	11	9										
VARIABLE NAME	ENABLE																											

NOTES
1. The FPLA is shipped with all links intact. Thus a background of entries corresponding to states of virgin links exists in the table, shown BLANK for clarity.
2. Unused I and B bits in the AND array are normally programmed Don't Care (–).
3. Unused product terms can be left blank.

Figure 15. Active Control of B_0-B_9 Using I_7 Active Low (L)

The previous 28-pin logic synthesis example (page 2-4) could be done on the 82S153 as follows:

$$X_0 = AB + \overline{C}D + B\overline{D}$$
$$\overline{X}_1 = \overline{A}B + \overline{C}D + EFG$$

Note that B_0 was used as a CHANGE input. When B_0 is HIGH (H) the outputs appear on B_8 and B_9. When B_0 is low (L), the outputs appear on B_6 and B_7. B_1 through B_5 are not used and therefore left unprogrammed.

FPLA PROGRAM TABLE

BIPOLAR FIELD PROGRAMMABLE LOGIC ARRAY (18X32X10) 82S152 (O.C.)/82S153 (T.S.)

PROGRAM TABLE ENTRIES

AND: INACTIVE (I) | I, B | I, B | DON'T CARE (—)

OR: ACTIVE (A) | INACTIVE (•) — B(O)

CONTROL (POL.): HIGH (H) | LOW (L)

TERM	AND I 7	6	5	4	3	2	1	0	AND B(I) 9	8	7	6	5	4	3	2	1	0	OR B(O) 9	8	7	6	5	4	3	2	1	0
POLARITY																			L	H	L	H						
0	H	H	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	•	A	•	A						
1	—	—	L	H	—	—	—	—	—	—	—	—	—	—	—	—	—	—	A	A	A	A						
2	—	H	—	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—	•	A	•	A						
3	L	H	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	A	•	A	•						
4	—	—	—	—	H	H	H	—	—	—	—	—	—	—	—	—	—	—	A	•	A	•						
•																												
•																												
•																												
30																												
31																												
D9	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	H										
D8	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	H										
D7	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	L										
D6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	L										
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
PIN	8	7	6	5	4	3	2	1	19	18	17	16	15	14	13	12	11	9	19	18	17	16	15	14	13	12	11	9
VARIABLE NAME	A	B	C	D	E	F	G											CHANGE	X_1 CHANGE	X_0 CHANGE	X_1	X_0						

Figure 16. 82S153 Example

SEQUENTIAL LOGIC CONSIDERATIONS

The 82S104/105 and 82S154-82S159 represent significant increases in complexity when compared to the combinational logic devices previously discussed. By combining the AND-OR combinational logic with clocked output flip-flops and appropriate feedback, SIGNETICS has created the first family of totally flexible sequential logic machines.

The 82S105 FPLS (Field Programmable Logic Sequencer) is an example of a high order machine whose applications are manifold. Application areas for this device include high speed data controllers, microprocessor and minicomputer bus arbitration, industrial controls, timing generation, multi-function counters and shift registers and microprocessor driven microcontrollers. The 82S105 is fully capable of performing fast sequential operations in relatively low speed microprocessor systems. By placing repetitive sequential operations on the 82S105, microprocessor overhead is reduced. Each 82S105 can be viewed as a high-speed 48 state subroutine.

The following pages summarize the 82S105 architecture and features.

FPLS ARCHITECTURE

The 82S104/105 Logic Sequencer is a programmable state machine of the Mealy type, in which the output is a function of the present state and the present input.

With the FPLS a user can program any logic sequence expressed as a series of jumps between stable states, triggered by a valid input condition (I) at clock time (t). All stable states are arbitrarily assigned and stored in the State Register. The logic output of the machine is also programmable, and is stored in the Output Register.

CLOCKED SEQUENCE

A synchronous logic sequence can be represented as a group of circles interconnected with arrows. The circles represent stable states, labeled with an arbitrary numerical code (binary, hex, etc.) corresponding to discrete states of a suitable register. The arrows represent state transitions, labeled with symbols denoting the jump condition and the required *change* in output. The number of states in the sequence depends on the length and complexity of the desired algorithm.

STATE JUMPS

The state from which a jump originates is referred as the present state (P), and the state to which a jump terminates is defined as the next state (N). A state jump always causes a change in state, but may or may not cause a change in machine output (F).

State jumps can occur only via "transition terms" T_n. These are logical AND functions of the clock (t), the present state (P), and a valid input (I). Since the clock is actually applied to the State Register, $T_n = I \bullet P$. When T_n is "true", a control signal is generated and used at clock time (t) to force the contents of the State Register from (P) to (N), and to change the contents of the Output Register (if necessary). The simple state jump below, involving 2 inputs, 1 state bit, and 1 output bit, illustrates the equivalence of discrete and programmable logic implementations.

FPLS LOGIC STRUCTURE

The FPLS consists of programmable AND and OR gate arrays which control the Set and Reset inputs of a State Register, as well as monitor its output via an internal feedback path. The arrays also control an independent Output Register, added to store output commands generated during state transitions, and to hold the output constant during state sequences involving no output changes. If desired, any number of bits of the Output Register can be used to extend the width of the State Register, via external feedback.

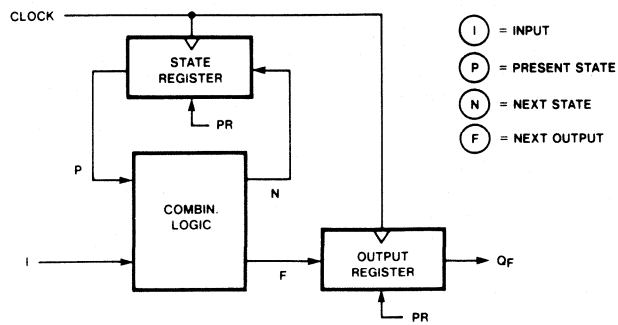


Figure 1. Basic architecture of 82S104/105 FPLS. I, P, N, and F are multi-line paths denoting groups of binary variables programmed by the user.

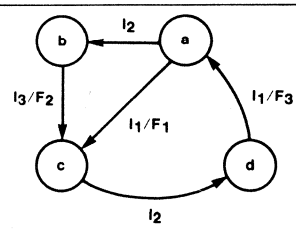


Figure 2. Typical state diagram. I_{1-3} are jump conditions which must be satisfied before any transitions take place. F_r are changes in output triggered by I_m, and stored in the Output Register. State transitions a→b and c→d involve no output change.

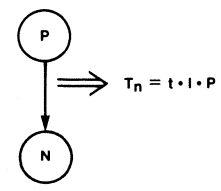


Figure 3. Typical state transition between any two states of Figure 2. The arrow connecting the two states gives rise to a transition term T_n. I is the jump condition.

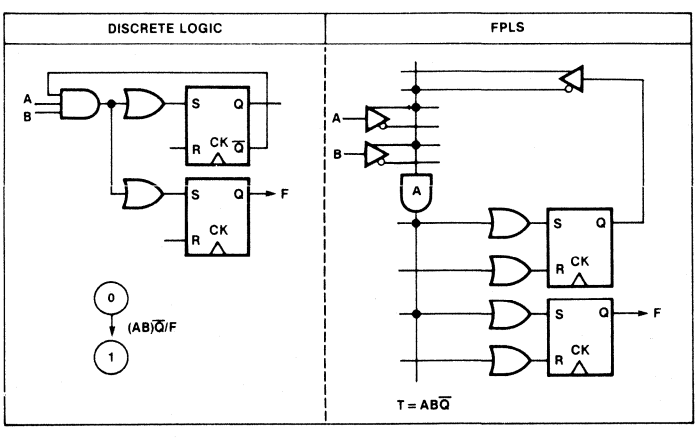


Figure 4. Typical state jump from state (0) to state (1), if inputs A=B="1". The jump also forces F="1", as required.

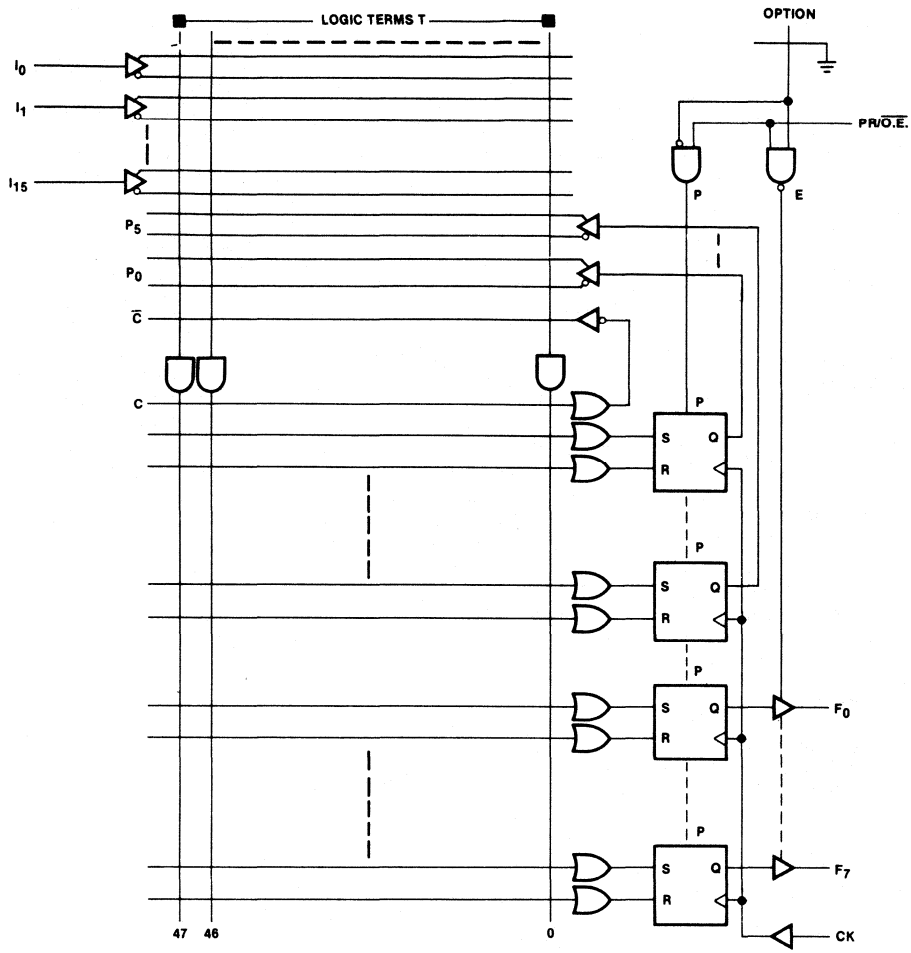


Figure 5. Simplified Logic Diagram of 82S104/105 FPLS

INPUT BUFFERS

16 external inputs (I_m) and 6 internal inputs (P_S), fed back from the state register, are combined in the AND array through two sets of True/Complement (T/C) buffers. There are a total of 22 T/C buffers, all connected to multi-input AND gates via fusible links which are initially intact.

Selective fusing of these links allows coupling either True, Complement, or Don't Care values of (I_m) and (P_S).

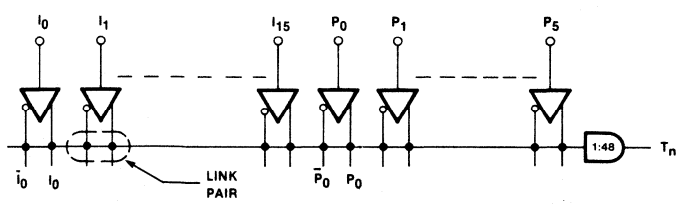


Figure 6. Typical AND gate coupled to (I) and (P) inputs. If at least one link *pair* remains intact, T_n is unconditionally forced Low.

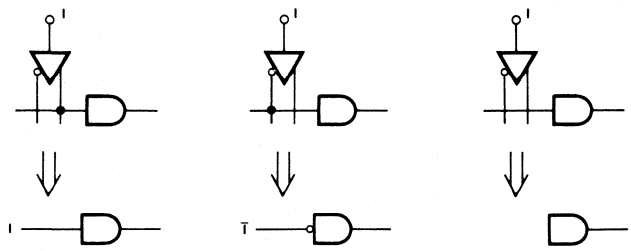


Figure 7. Choice of input polarity coupling to a typical AND gate. With both links open, (I) is logically Don't Care.

"AND" ARRAY

State jumps and output changes are triggered at clock time by valid transition terms T_n. These are logical AND functions of the present state (P) and the present input (I).

The FPLS AND array contains a total of 48 AND gates. Each gate has 45 inputs—44 connected to 22 T/C input buffers, and 1 dedicated to the Complement Array. The outputs of all AND gates are propagated through the OR array, and used at clock time (t) to force the contents of the State Register from (P) to (N). They are also used to control the Output Register, so that the FPLS 8-bit output F_r is a function of the inputs and present state.

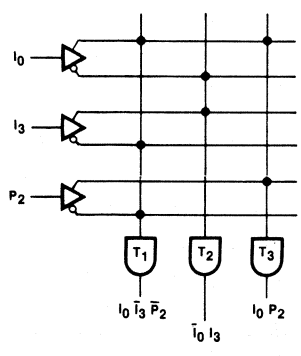


Figure 8. Typical transition terms involving arbitrary inputs and state variables. All remaining gate inputs are programmed Don't Care. Note that T_2 output is state independent.

"OR" ARRAY

In general, a clocked sequence will consist of several stable states and transitions, as determined by the complexity of the desired algorithm. All state and output changes in the state diagram imply changes in the contents of state and output registers.

Thus, each flip-flop in both registers may need to be conditionally set or reset several times with T_n commands. This is accomplished by selectively ORing through a programmable OR array all AND gate outputs T_n necessary to activate the proper flip-flop control inputs.

The FPLS OR array consists of 14 pairs of OR gates, controlling the S/R inputs of 14 state and output register stages, and a single OR gate for the Complement Array. All gates have 48 inputs for connecting to all 48 AND gates.

COMPLEMENT ARRAY

The COMPLEMENT array provides an asynchronous feed back path from the OR array back to the AND array.

This structure enables the FPLS to perform both direct and complement sequential state jumps with a minimum of transition (AND) terms.

Typically direct jumps, such as T_1 and T_2 in Figure 11 require only a single AND gate each.

But a complement jump such as T_3 generally requires many AND gates if implemented as a direct jump. However, by using the complement array, the logic requirements for this type of jump can be handled with just one more gate from the AND array.

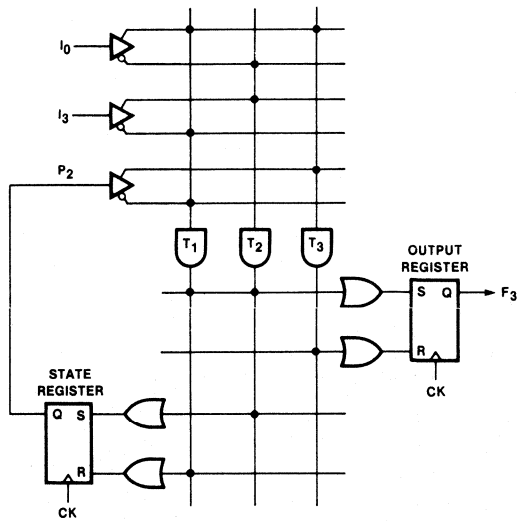


Figure 9. Typical OR array gating of transition terms $T_{1,2,3}$ controlling arbitrary state and output register stages.

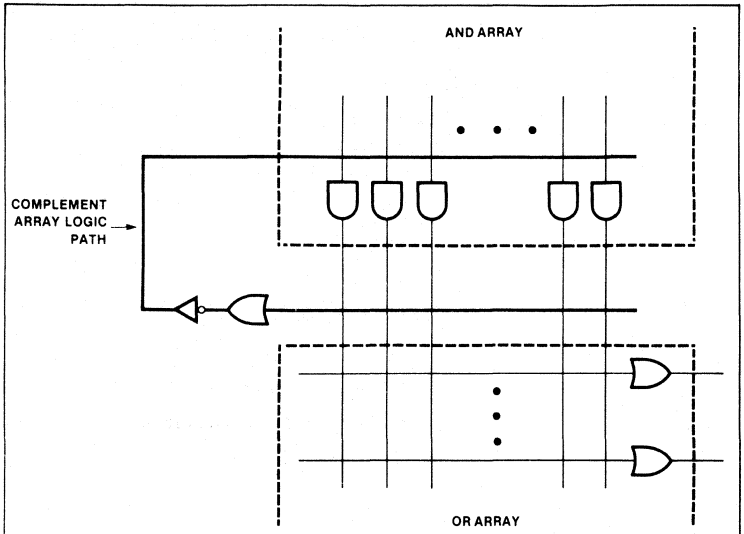
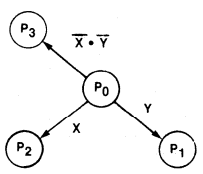


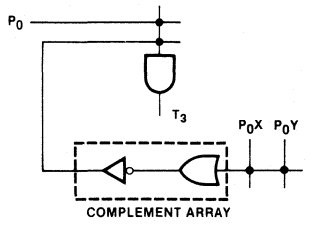
Figure 10. The COMPLEMENT array is logically constructed from a 48-input programmable OR gate followed by an inverter. All AND terms coupled to the OR gate are complemented at the inverter output, and can be fed back as inputs to the AND array.

As indicated in Figure 12, the single complement array gate may be used for many states of the state diagram. This happens because all transition terms linked to the OR gate include the present state as a part of their conditional logic. In any particular state only those transition terms which are a function of that state are enabled; all other terms coupled to different states are disabled and do not influence the output of the complement array. As a general rule of thumb, the complement array can be used as many times as there are states.

(A) TYPICAL STATE SEQUENCE **(B) COMPLEMENT JUMP**



TRANSITION TERMS



DIRECT

$T_1 = P_0X$

$T_2 = P_0Y$

$T_3 = P_0(\overline{P_0X + P_0Y})$

$T_3 = P_0[\overline{P_0(X + Y)}]$

$T_3 = P_0(\overline{P}_0 + \overline{(X + Y)})$

$T_3 = 0 + P_0\overline{(X + Y)}$

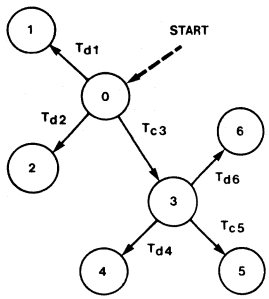
$T_3 = P_0(\overline{X} \cdot \overline{Y})$

COMPLEMENT

$T_3 = P_0(\overline{X} \cdot \overline{Y}) = P_0(\overline{T_1 + T_2})$

Figure 11. (A) X and Y specify the conditional logic for direct jump transition terms T_1 and T_2. The complement jump term T_3 is true only when both T_1 and T_2 are false. (B) Note that the complementary logic expression for T_3, $\overline{T_1 + T_2}$, corresponds exactly to the logic structure of the complement array.

(A) STATE DIAGRAM



(B) LOGIC DEFINITION

$T_{d1} = I_0\bar{I}_1P_0$

$T_{d2} = I_2P_0$

$T_{c3} = \overline{(T_{d1} + T_{d2})P_0} = \overline{(I_0\bar{I}_1 + I_2)P_0}$

$T_{d4} = \bar{I}_2P_3$

$T_{d6} = I_0I_1P_3$

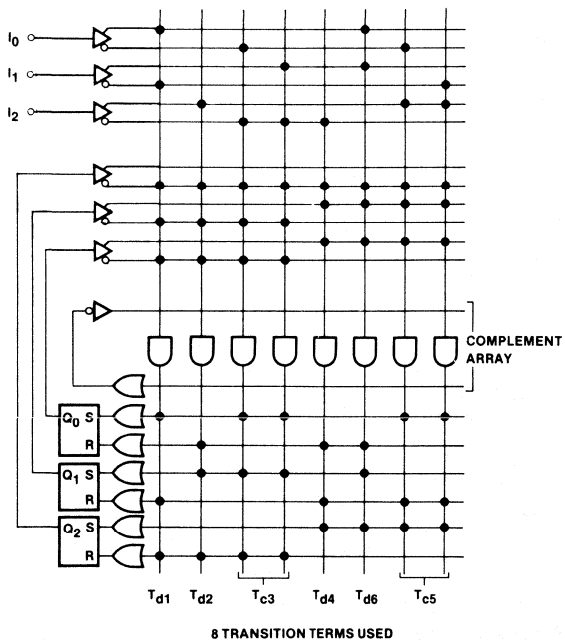
$T_{c5} = \overline{(T_{d4} + T_{d6})P_3} = \overline{(I_0I_1 + \bar{I}_2)P_3}$

T_{cn} = COMPLEMENT STATE TRANSITION TERM

T_{dn} = DIRECT STATE TRANSITION TERM

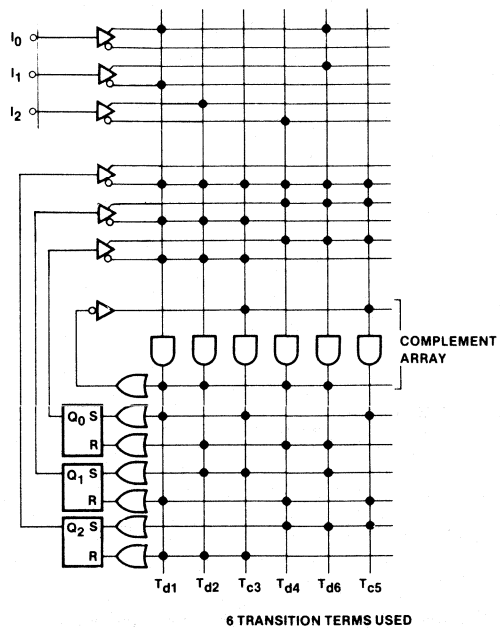
P_s = PRESENT STATE

(C) STATE LOGIC WITHOUT USING THE COMPLEMENT ARRAY



8 TRANSITION TERMS USED

(D) STATE LOGIC USING THE COMPLEMENT ARRAY



6 TRANSITION TERMS USED

Figure 12. Logic reduction with the complement array. The logic state diagram in (a) includes complement jumps T_{C3} and T_{C5} defined in (b). When using the complement array a savings of 2 transition terms results, as shown in (c) and (d).

IFL-20 SEQUENTIAL DEVICES

The 20-pin IFL family also includes sequential devices. These devices are all similar in architecture. The major difference consists of the number of outputs which are registered. The following table summarizes the IFL-20 sequential family.

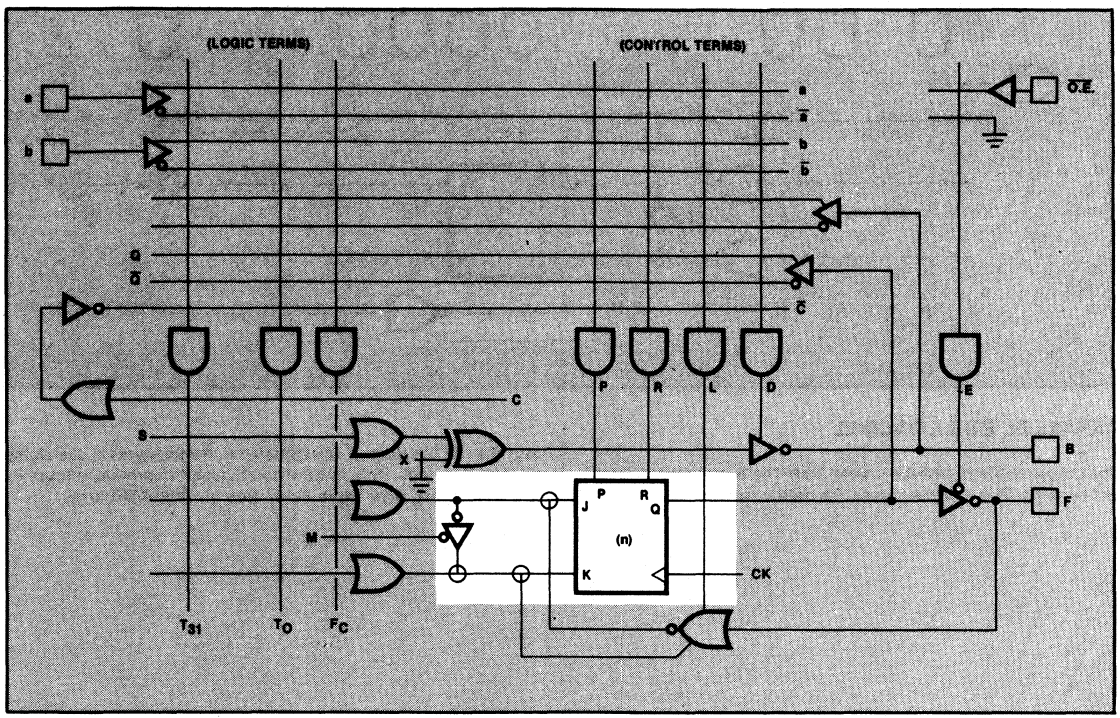
Device	Registered Outputs	Output Type
82S154	4-bit Register	O.C.
82S155	4-bit Register	T.S.
82S156	6-bit Register	O.C.
82S157	6-Bit Register	T.S.
82S158	8-bit Register	O.C.
82S159	8-bit Register	T.S.

IFL-20 SEQUENTIAL FAMILY

FEATURES

The IFL-20 sequencers have been designed with a maximum of flexibility in mind. Each element of the architecture contributes to the ease of use the IFL-20 family provides. Each part has the features listed:

- J/K flip-flop Output Register
- Dynamic control of flip-flop type (J/K or D)
- Parallel bus load
- Asynchronous preset and reset capability
- Combinational I/O pins
- Programmable enable pins
- Output feedback to AND array available



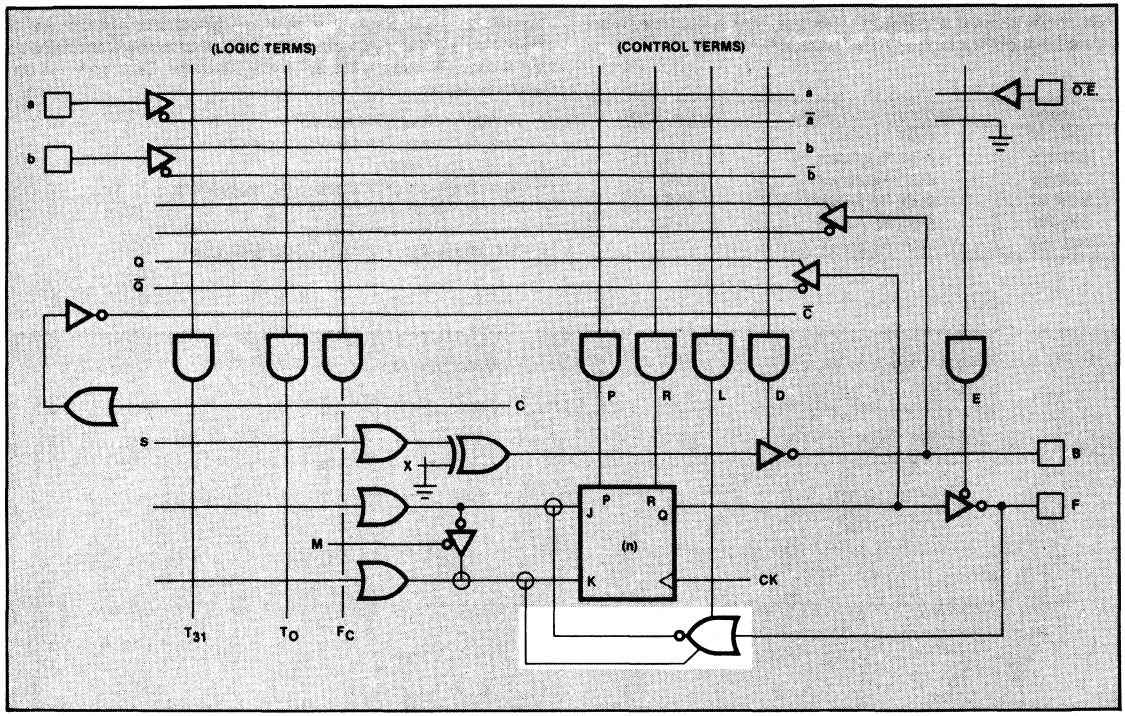
OUTPUT REGISTER

The output register of the IFL-20 sequential devices are comprised of fully implemented J/K flip-flops. Each flip-flop is positive edge-triggered from a common clock.

In addition, a dynamically controllable "foldback" buffer between the J and K inputs to the flip-flop allows the designer to change the function of the flip-flop from J/K to D type under the control of the flip-flop control term (F). A fuse allows the designer to dedicate the flip-flop as a permanent D-type by programming the fuse.

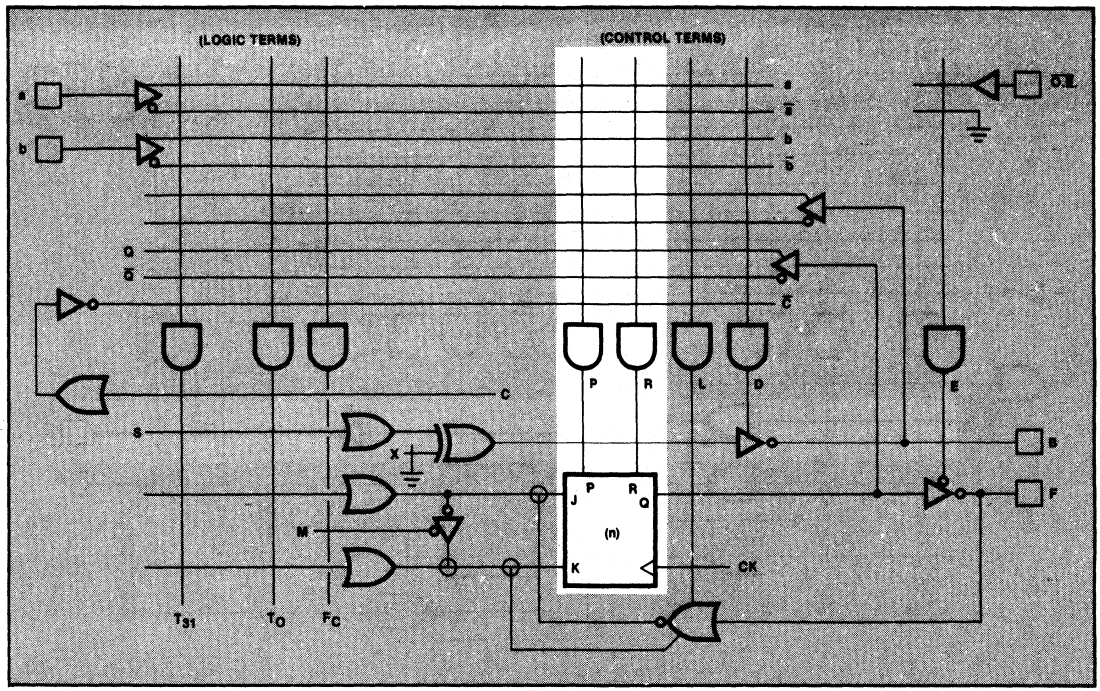
By leaving the fuse intact, the flip-flop control line (F) is maintained as the active mode control. If the output of F is logic Low (0), then the flip-flop is configured as a D-type. If the output of F is a logic High (1), then the flip-flop is configured as a J/K type. Term F is programmed in the same manner as the 32 logic transition terms (T). It is well to note at this time that when the flip-flop is configured as a D-type, the OR-term driving the K input must not be active.

During all modes of operation, the output register data is fed back to the AND array. This feedback is used to establish present-state to next-state jumps.



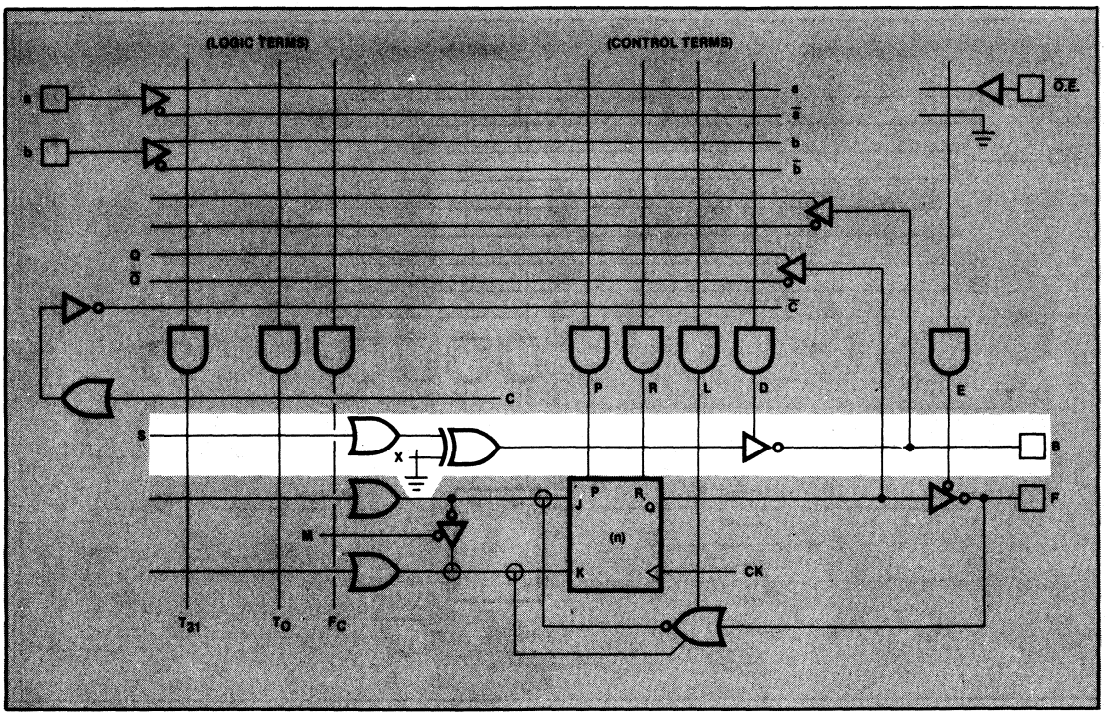
PARALLEL BUS LOADING

The output register may be loaded from the bus via the load control term (L). This feature forces the data contained on the F_0 pin into the J/K flip-flop. It operates synchronously with the rising edge of the clock. This feature can be used to preload a state into the machine, or to latch input data into the AND array.



PRESET AND RESET

Asynchronous preset and reset capability has been provided on the IFL-20 sequential devices. This feature is controlled in the AND-array on the 82S158 and 82S159. The remainder of the IFL-20 devices are controlled via the OR-array. See the individual data sheet for details.



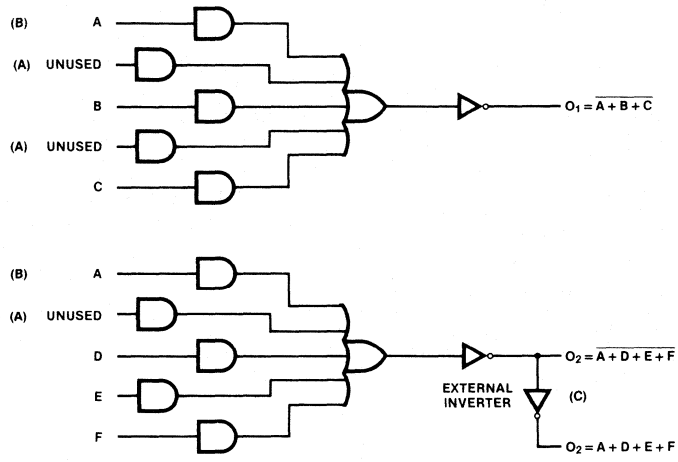
COMBINATIONAL I/O PINS

Each IFL-20 sequential device has a number of combinational pins (B_0) which can be programmed and used exactly as the B pins on the 82S152 and 82S153 FPLA. The direction control term (D) establishes the data flow on these pins. The individual data sheet should be consulted for the quantity and pin number assignment for each device.

PLA vs. PAL™

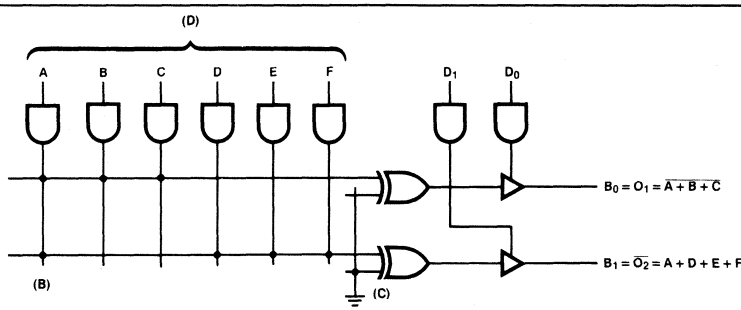
The PLA architecture provides the most efficient means of implementing logic. The 28-pin devices, 82S100/101 and 82S102/103 are unique in their ability to directly implement logic. They offer the most useable P-terms, the highest number of inputs and outputs, and the most straightforward programming of any logic device in the industry. The following diagrams illustrate the relative ease of programming the flexible PLA structure against the fixed OR array of the PAL.

PAL is a trademark of Monolithic Memories, Inc.



NOTES:
(A) Unused P-Terms lost to designer.
(B) Common P-Term to 2 or more outputs requires 2 or more P-Terms; one on each output.
(C) External inverter required to change active level of output.
(D) 10 P-Terms used.

Figure 17. Application Example PAL Approach



NOTES:
(A) All P-Terms usable.
(B) Common P-Terms available to all outputs.
(C) Programmable outputs eliminate need for external component.
(D) 6 P-Terms used.

Figure 18. IFL Approach

FPLA 82S152/153	COMPATIBLE WITH PAL™ LOW/ MEDIUM COMPLEXITY PRODUCT, i.e., 14H/L4 16H/L2 16L8 10H/L8 16C1, 12H/L6 ADDITIONAL FEATURES: 1. PROGRAMMABLE OUTPUT POLARITY 2. PROGRAMMABLE OR ARRAY 3. 10 OUTPUTS vs. 8 FOR PAL 4. 32 PRODUCT TERMS vs. 16 FOR LOW AND MEDIUM PALs
FPLS 82S154/6/8 (O/C) 82S155/7/9 (T/S)	COMPATIBLE WITH PAL REGISTERED TYPES 16R8, 16R6, 16R4, etc. ADDITIONAL FEATURES: 1. PROGRAMMABLE AND/OR, ALSO CONTROLLABLE FLIP FLOPS; AS D, TOGGLE, OR JK

PAL is a trademark of Monolithic Memories, Inc.

Figure 19. IFL-20 Product Comparison

The Signetics approach to programmable logic gives the designer the flexibility he needs where he needs it — in the device itself. The combination of totally flexible architecture, more useable product terms, and ease of logic implementation make Signetics Integrated Fuse Logic the obvious choice.

IFL FAMILY SUMMARY

FIELD PROGRAMMABLE GATE ARRAYS

82S102	16 × 9	28-pin	Open Collector
82S103	16 × 9	28-pin	Tri-state
82S150	18 × 12	20-pin	Open Collector
82S151	18 × 12	20-pin	Tri-state

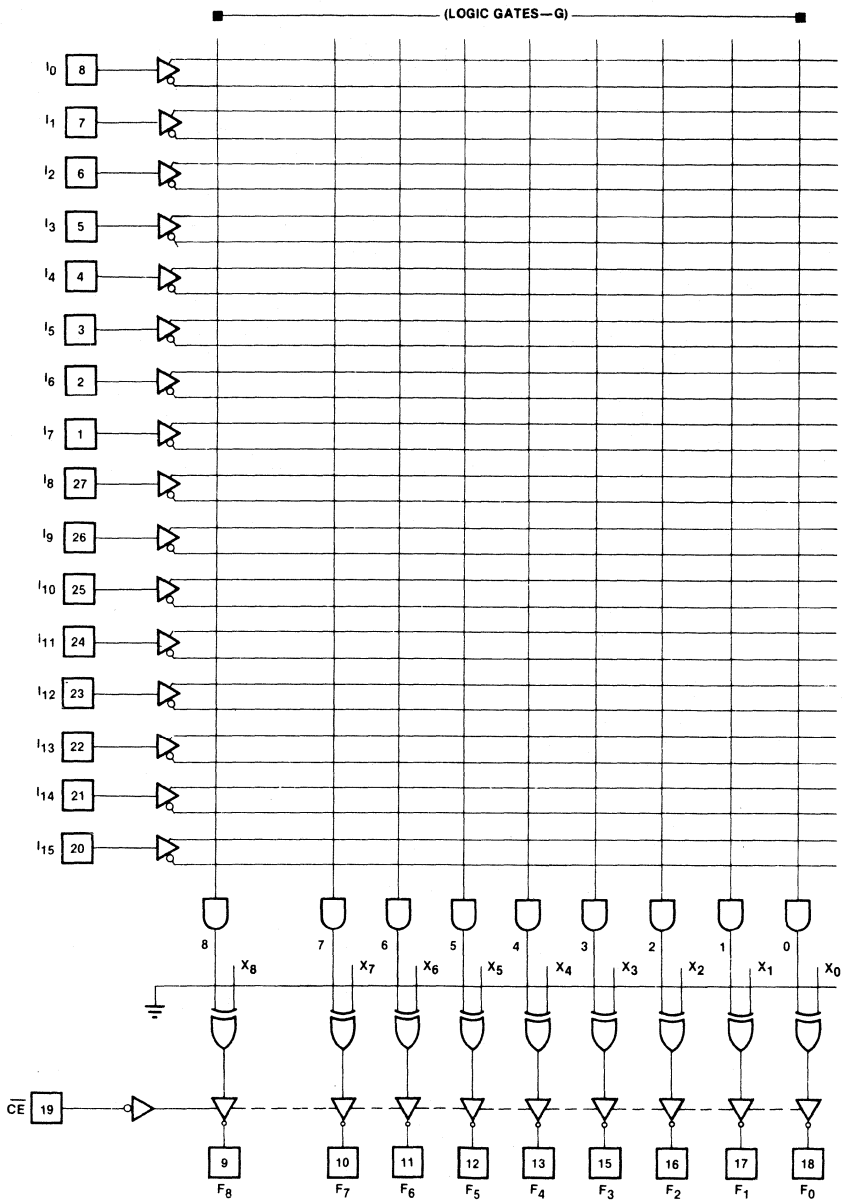
FIELD PROGRAMMABLE LOGIC ARRAYS

82S100	16 × 48 × 8	28-pin	Tri-state
82S101	16 × 48 × 8	28-pin	Open Collector
82S152	18 × 32 × 10	20-pin	Open Collector
82S153	18 × 32 × 10	20-pin	Tri-state

FIELD PROGRAMMABLE LOGIC SEQUENCERS

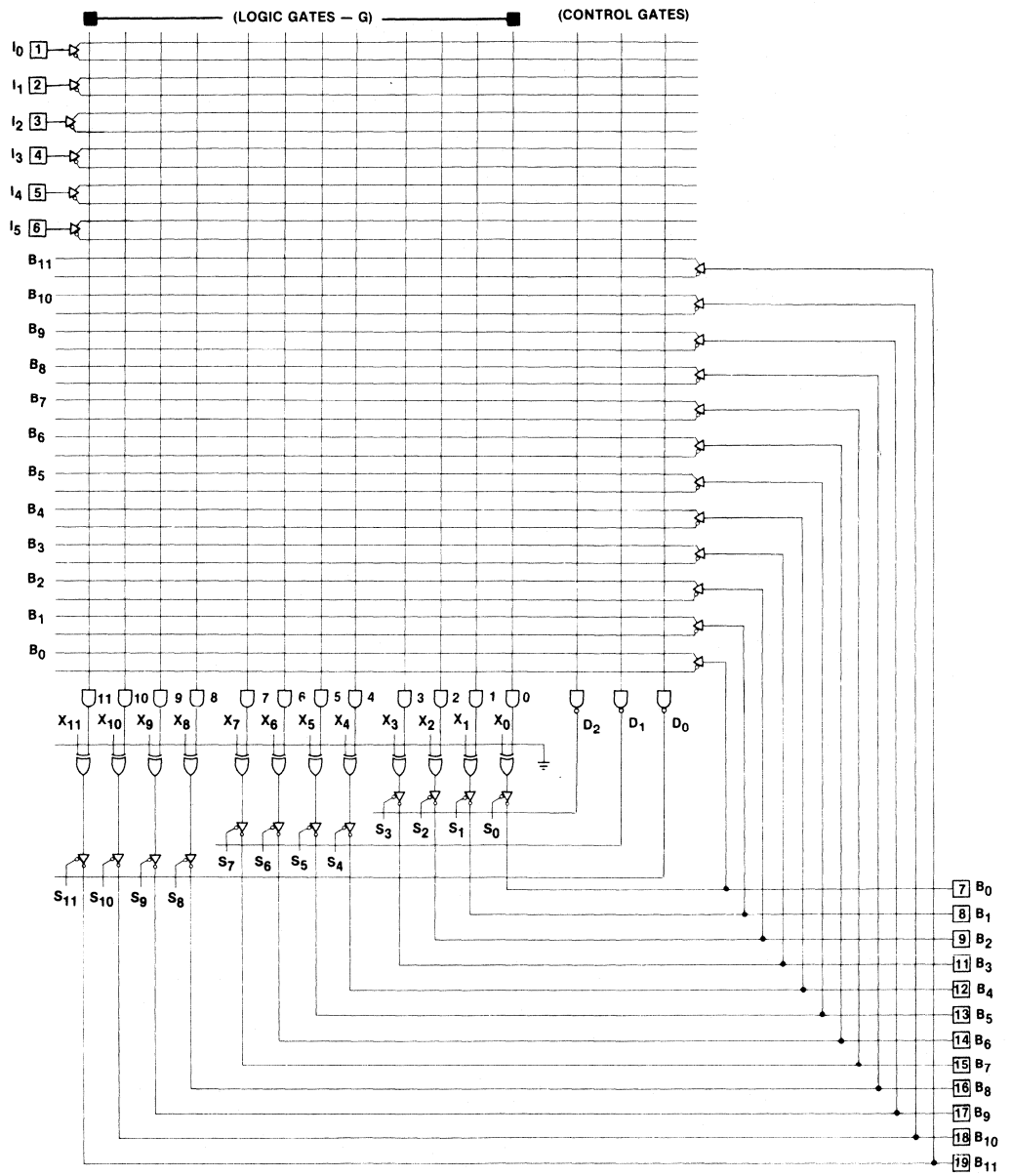
82S104	16 × 48 × 8	28-pin	O.C. Mealy State Machine
82S105	16 × 48 × 8	28-pin	T.S. Mealy State Machine
82S154	16 × 48 × 8	20-pin	O.C. 4-bit Sequencer
82S155	16 × 32 × 12	20-pin	T.S. 4-bit Sequencer
82S156	16 × 32 × 12	20-pin	O.C. 6-bit Sequencer
82S157	16 × 32 × 12	20-pin	T.S. 6-bit Sequencer
82S158	16 × 32 × 12	20-pin	O.C. 8-bit Sequencer
82S159	16 × 32 × 12	20-pin	T.S. 8-bit Sequencer

FPGA LOGIC DIAGRAM



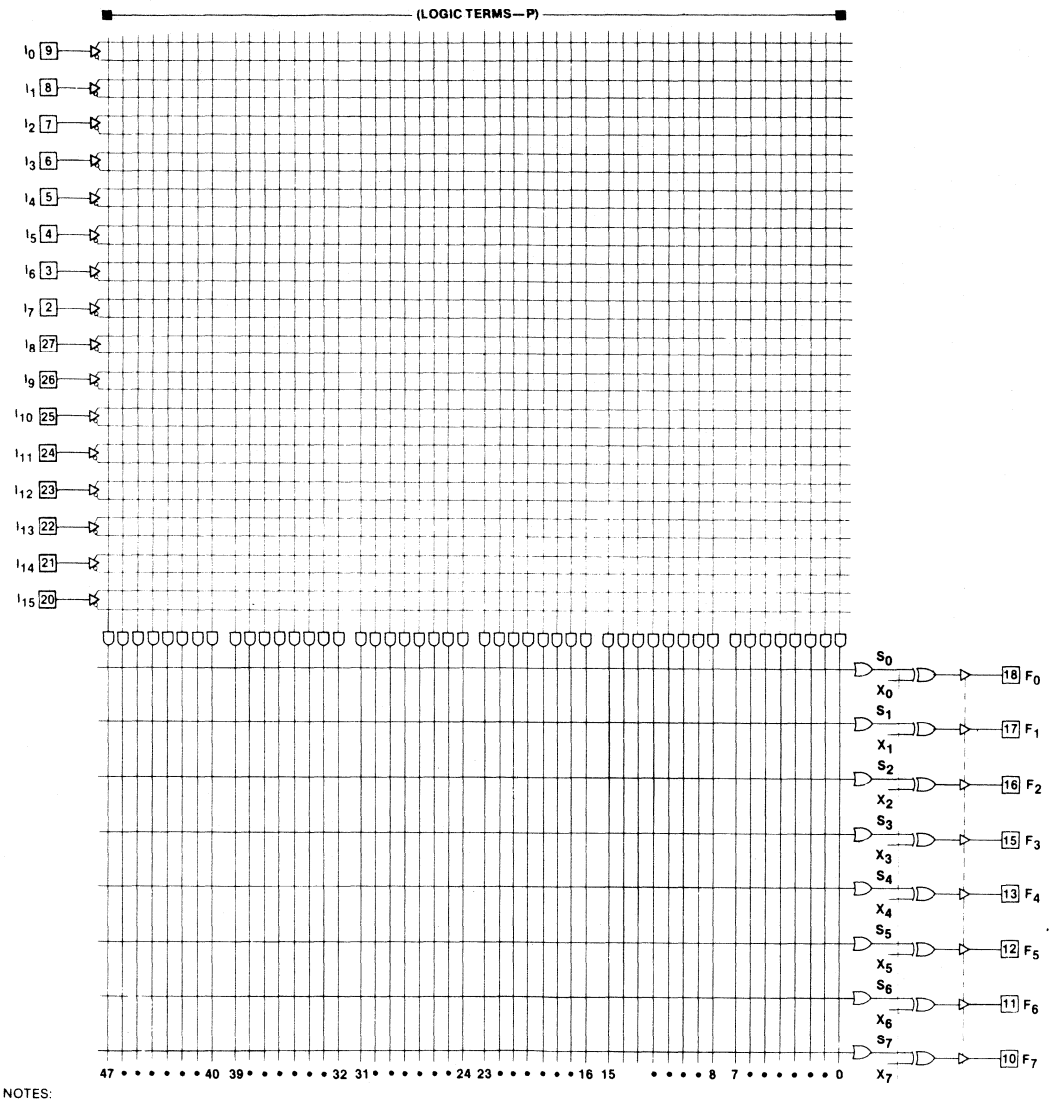
NOTES:
1. All gate inputs with a blown link float to a logic "1".
2. Programmable connection.

FPGA LOGIC DIAGRAM



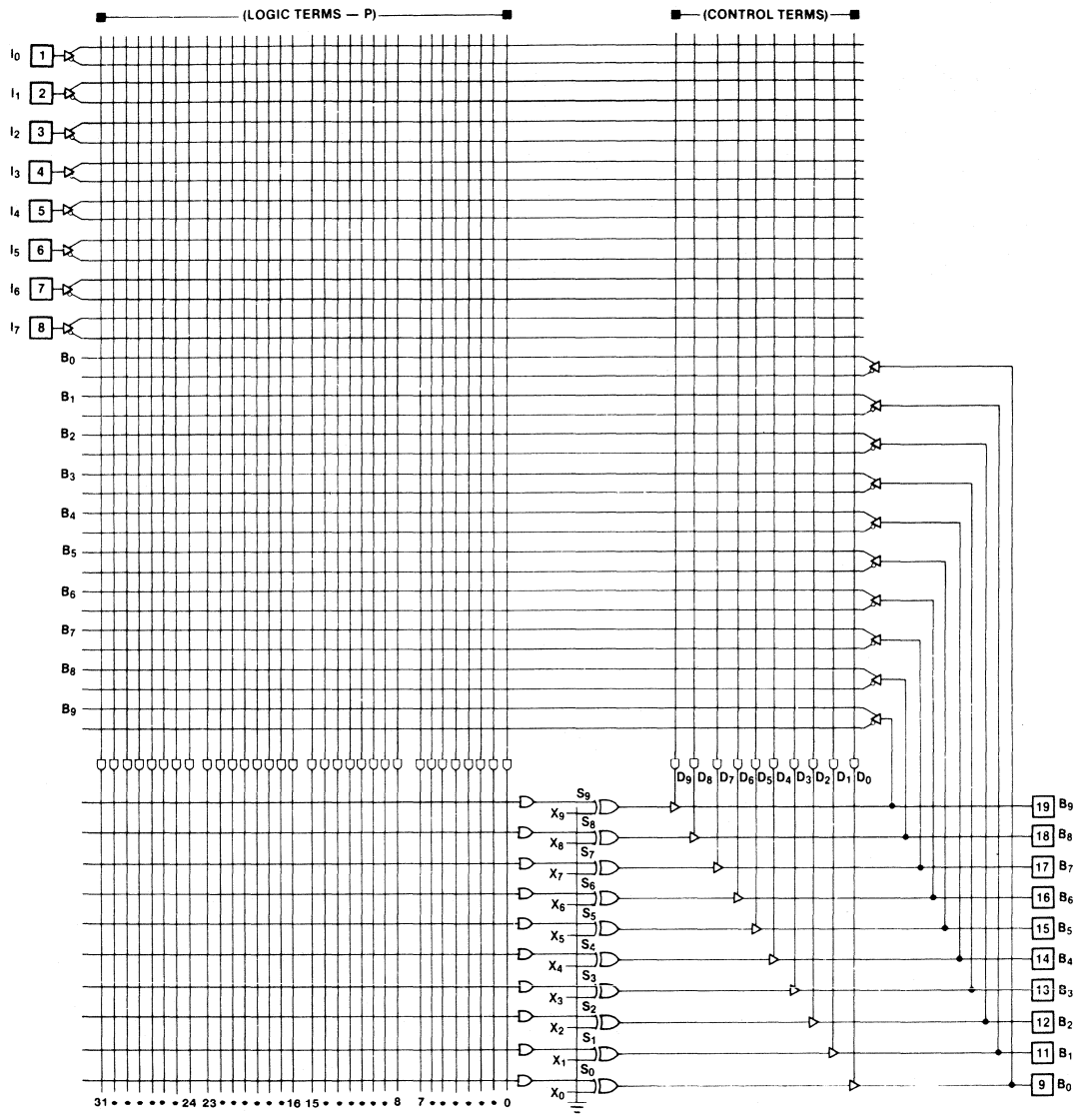
NOTES:
1. All gate inputs with a blown link float to a logic "1".
2. Programmable connections.

FPLA LOGIC DIAGRAM



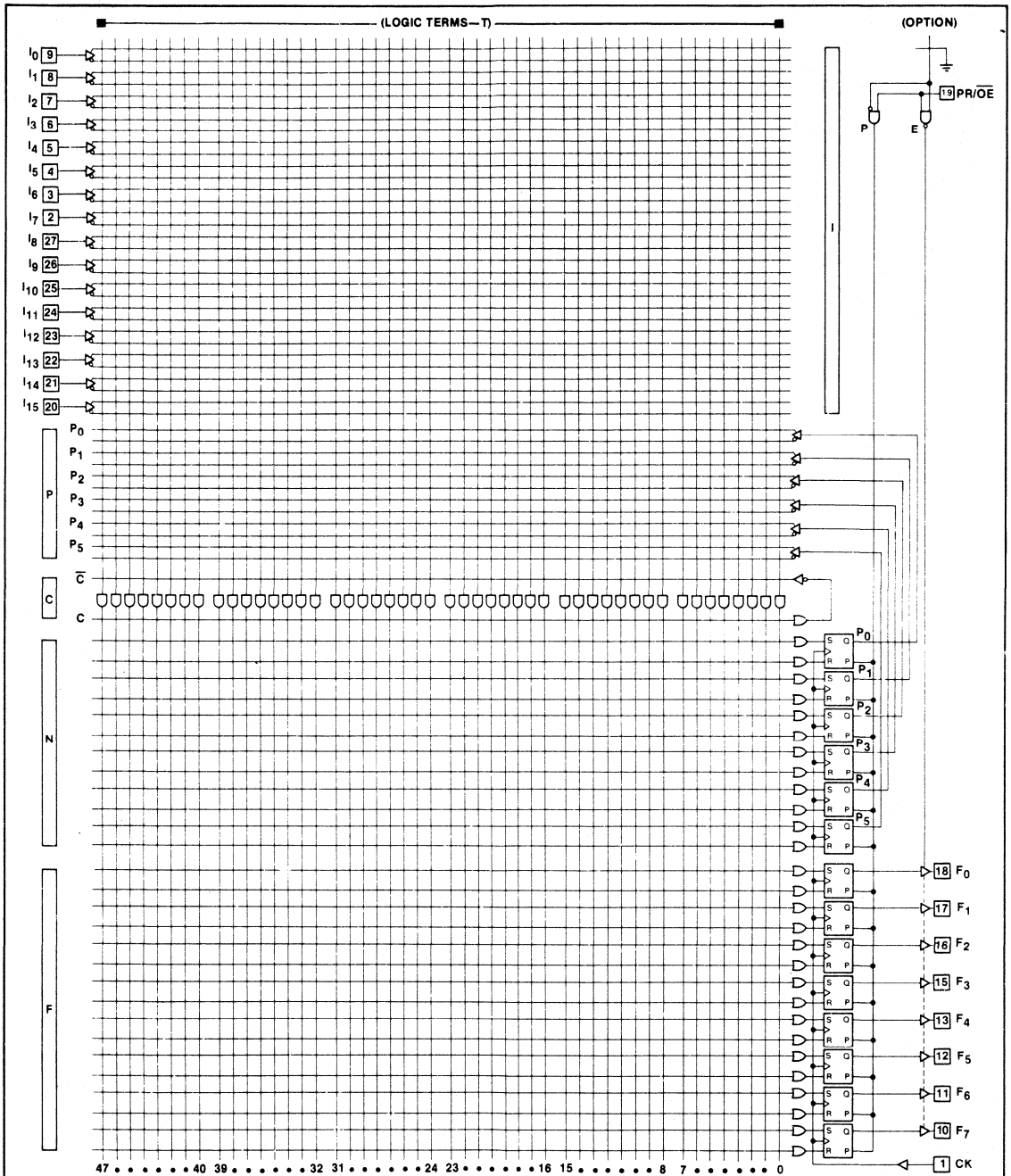
NOTES:
1. All AND/EX-OR gate inputs with a blown link float to a logic "1".
2. All OR gate inputs with a blown link float to a logic "0".
3. Programmable connections.

FPLA LOGIC DIAGRAM



NOTES:
1. All AND/EX-OR gate inputs with a blown link float to a logic "1".
2. All OR gate inputs with a blown link float to a logic "0".
3. Programmable connections.

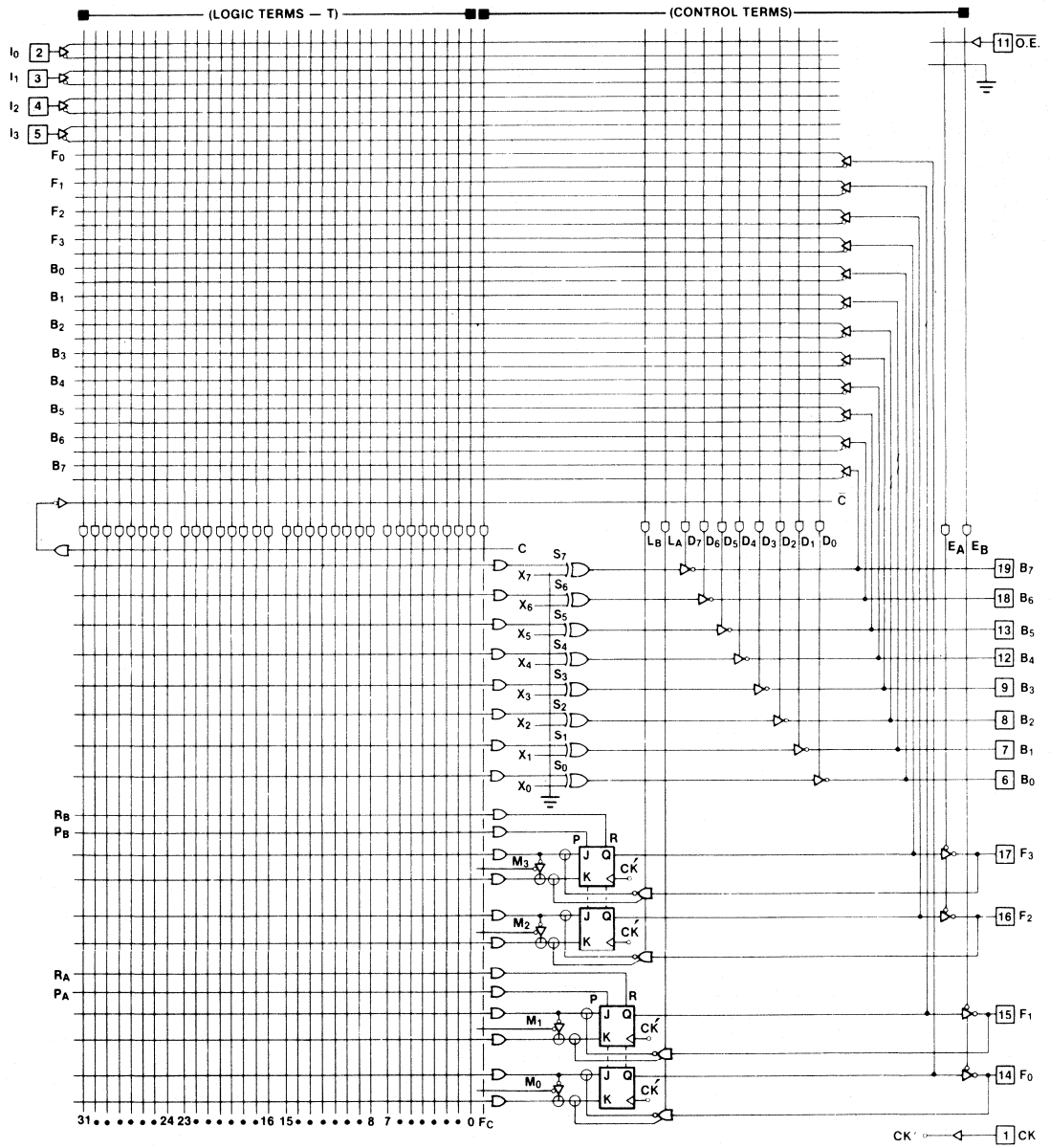
FPLS LOGIC DIAGRAM



NOTES
1. All AND gate inputs with a blown link float to a logic "1".
2. All OR gate inputs with a blown link float to a logic "0".
3. Programmable connection.

FPLS LOGIC DIAGRAM

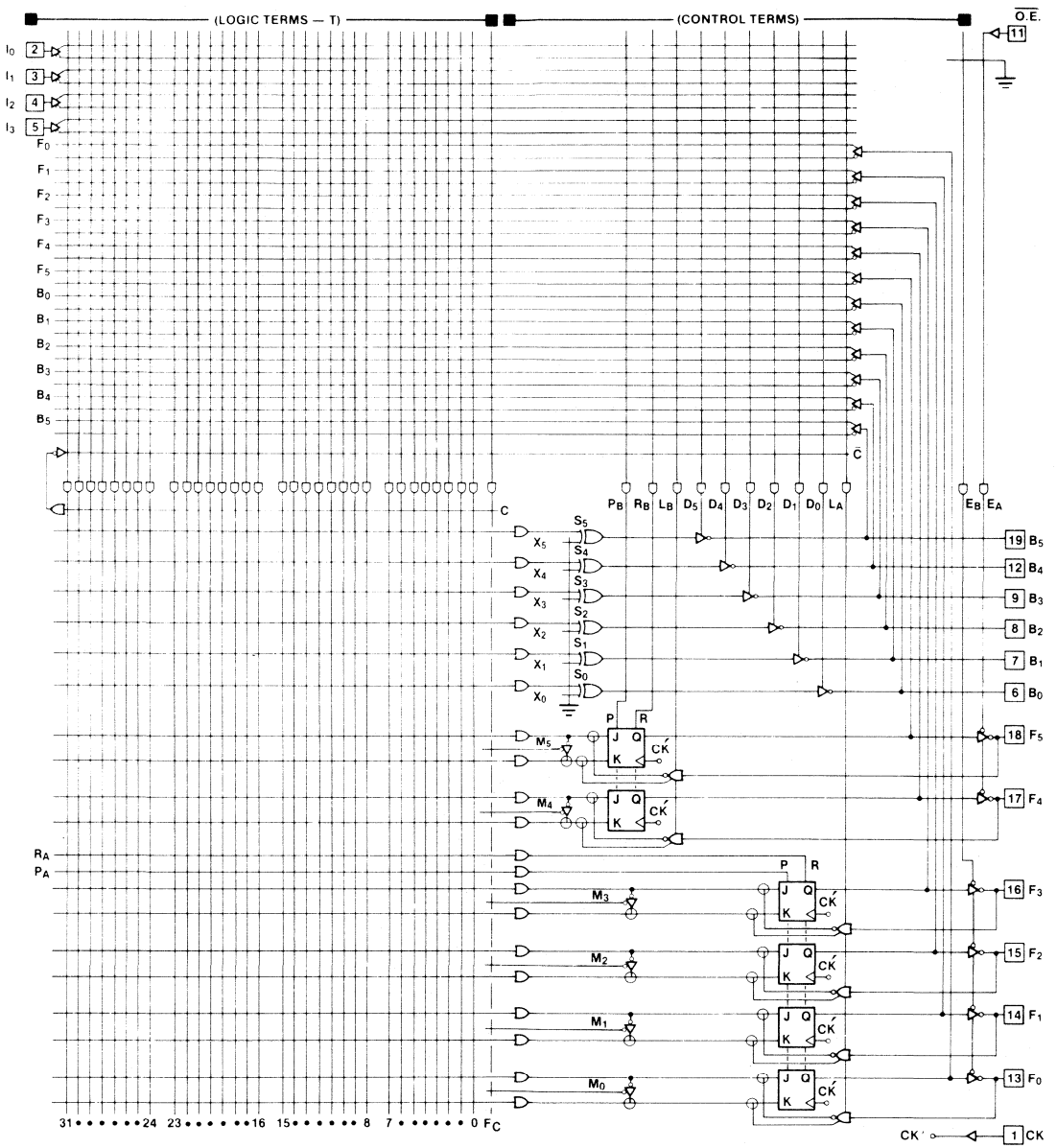
82S154/155



NOTES:
1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inptus with a blown link float to logic "1".
3. Programmable connection.
4. ⏚ denotes WIRE-OR.

FPLS LOGIC DIAGRAM

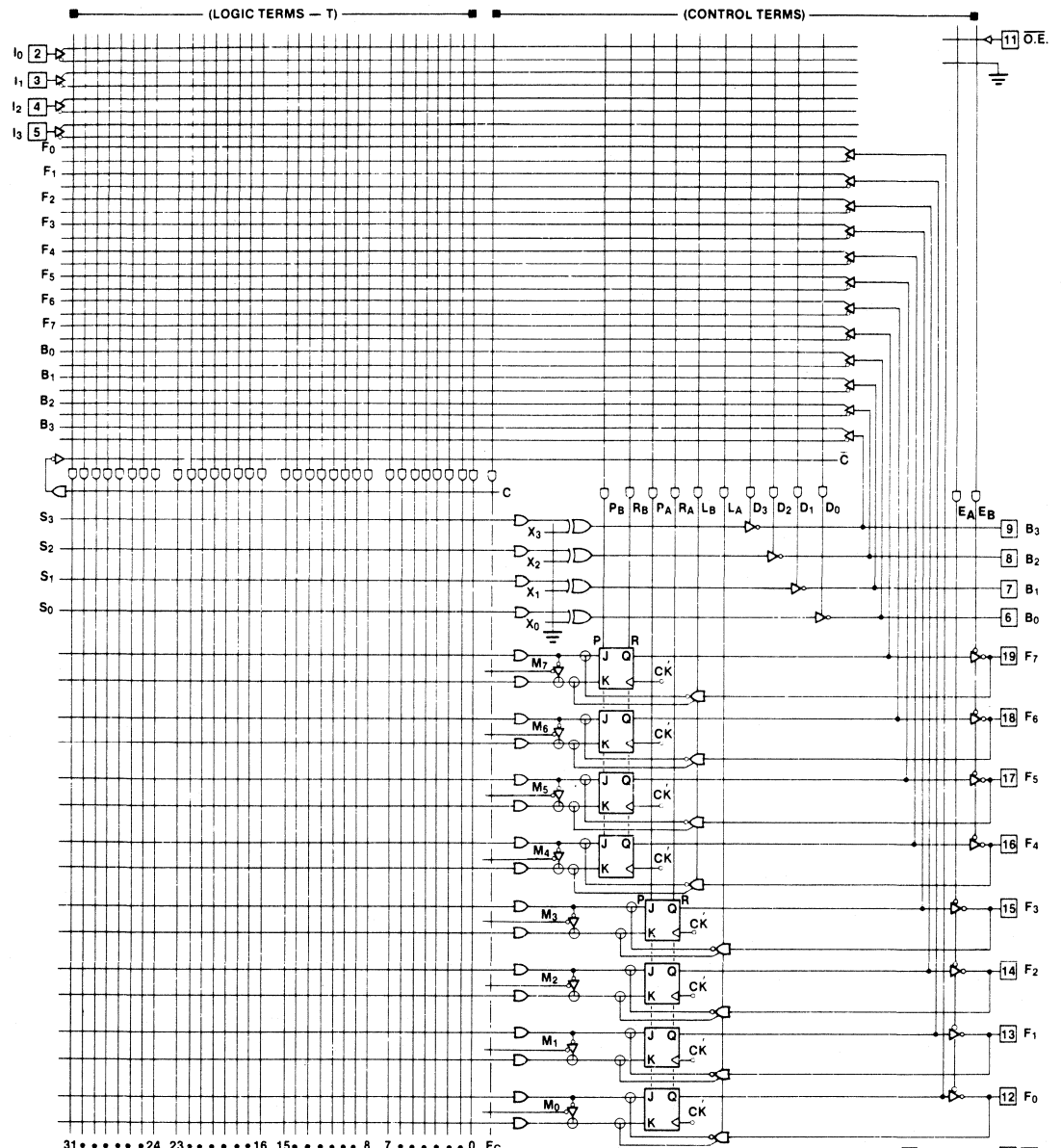
82S156/157



NOTES:
1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inputs with a blown link float to logic "1".
3. ● Programmable connection.
4. ⊕ denotes WIRE-OR.

FPLS LOGIC DIAGRAM

82S158/159



NOTES:
1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inputs with a blown link float to logic "1".
3. Programmable connection.
4. ⊕ denotes WIRE-OR.

IFL SERIES 20

FIELD PROGRAMMABLE GATE ARRAY (18×15×12) 82S150 (O.C.)/82S151 (T.S.)

Preview

INTEGRATED FUSE LOGIC
SERIES 20

DESCRIPTION

The 82S150 and the 82S151 are single level logic elements, consisting of 12 AND gates with fusible link connections for programming I/O polarity, I/O direction and output enable control.

All gates are linked to 6 inputs (I) and 12 bidirectional I/O lines (B). These yield variable I/O gate configurations via 3 direction control gates (D), ranging from 18 inputs to 12 outputs.

On chip T/C buffers couple either True (I, B) or Complement ($\overline{I}$, $\overline{B}$) input polarities to each AND gate. The polarity of all gate outputs is individually programmable through a set of EX-OR gates for implementing AND/NAND logic functions. Alternately, if desired, OR/NOR logic functions can also be realized by programming for each gate the complement of its inputs and output (DeMorgan's Theorem).

The 82S150 and the 82S151 are field programmable, enabling the user to quickly generate custom patterns using standard programming equipment.

Both devices are available in a 20-pin slim line package. For the commercial temperature range (0°C to +75°C) specify N82S150/151 N or F. For the military temperature range (−55°C to +125°C) specify S82S150/151 F only.

FEATURES

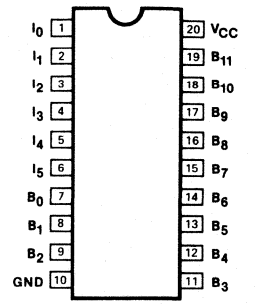
- Field Programmable (Ni-Cr link)
- 6 inputs
- 15 Product Terms:
  - 12 Logic Terms
  - 3 Control Terms
- 12 bidirectional I/O lines
- Active high or low outputs
- Programmable output enable
- Power dissipation: 650mW (typ)
- I/O propagation delay: 25ns (max)
- Input loading
  - N82S150/151: −100μA (max)
  - S82S150/151: −150μA (max)
- Output options
  - 82S150: open collector
  - 82S151: three-state
- TTL compatible

APPLICATIONS

- Random gating functions
- Address decoding
- Code detectors
- Memory mapped I/O
- Fault monitors
- I/O port decoders

PIN CONFIGURATION

F,N PACKAGES



F = Cerdip
N = Plastic

LOGIC FUNCTIONS

Typical Output Functions:

Active-High

$X = A \cdot \overline{B} \cdot C \cdot \ldots\ldots$

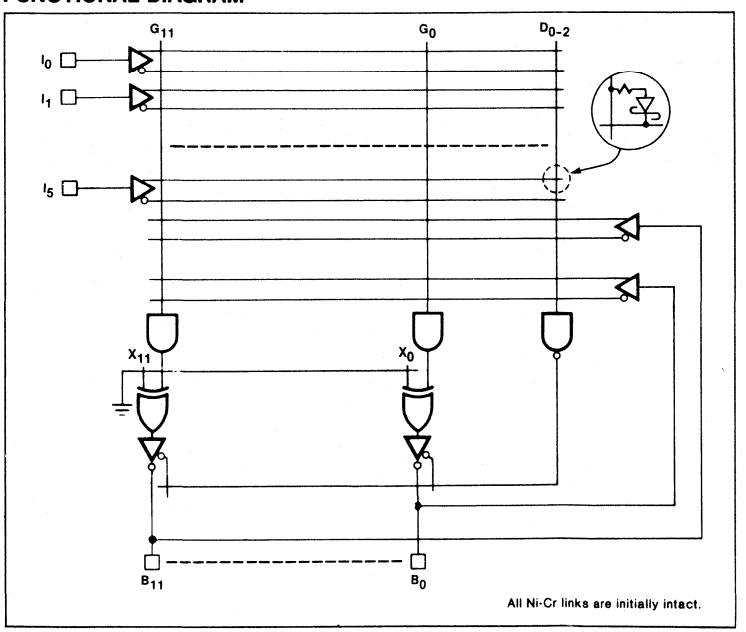
Active-Low

$X = A \cdot \overline{B} \cdot C \cdot \ldots\ldots$

$X = \overline{A} + B + \overline{C} + \ldots\ldots$

NOTES:
1. For each of the 12 outputs, either function X (active-high) or $\overline{X}$ (active-low) is available, but not both. The desired output polarity is programmed via the EX-OR gates.
2. X, A, B, C, etc. are user defined connections to fixed inputs (I) and bidirectional pins (B).

FUNCTIONAL DIAGRAM

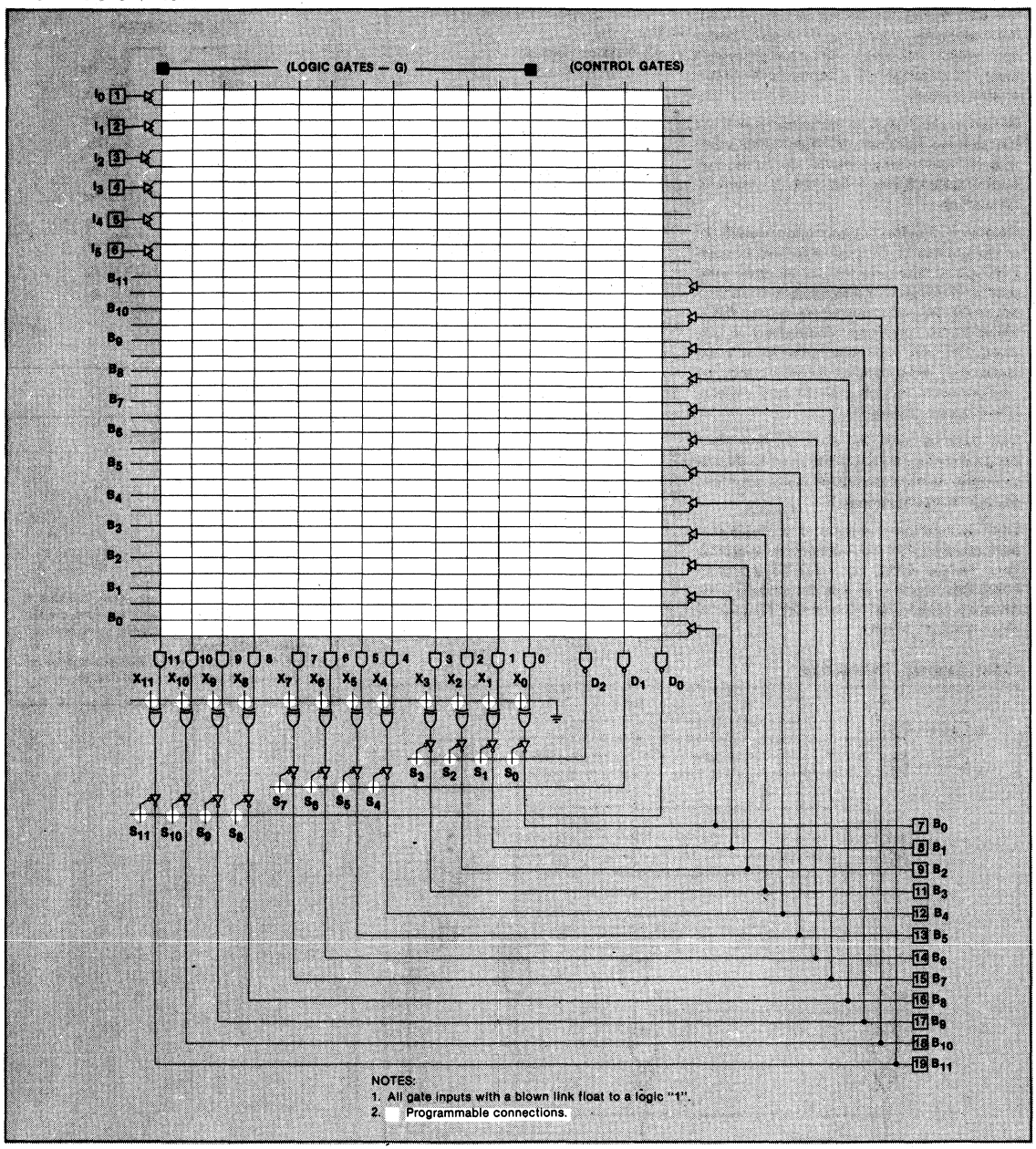


All Ni-Cr links are initially intact.

Preview

INTEGRATED FUSE LOGIC
SERIES 20

FPGA LOGIC DIAGRAM



Preview

INTEGRATED FUSE LOGIC
SERIES 20

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING Min	RATING Max	UNIT
V_{CC}	Supply voltage		+7	Vdc
V_{IN}	Input voltage		+5.5	Vdc
V_{OUT}	Output voltage		+5.5	Vdc
I_{IN}	Input currents	−30	+30	mA
I_{OUT}	Output currents		+100	mA
	Temperature range			C°
T_A	Operating			
	N82S150/151	0	+75	
	S82S150/151	−55	+125	
T_{STG}	Storage	−65	+150	

THERMAL RATINGS

TEMPERATURE	Military	Commercial
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

DC ELECTRICAL CHACTERISTICS

N82S150/151: $0°C \le T_A \le +75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S150/151: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TEST CONDITION	N82S150/151 Min	N82S150/151 Typ[2]	N82S150/151 Max	S82150/151 Min	S82150/151 Typ[2]	S82150/151 Max	UNIT
	Input voltage[3]								V
V_{IL}	Low	V_{CC} = Min			.85			.80	
V_{IH}	High	V_{CC} = Max	2.0			2.0			
V_{IC}	Clamp[3,4]	V_{CC} = Min, I in = − 18mA		−.8	−1.2		−.8	−1.2	
	Output voltage[3]	V_{CC} = Min							V
V_{OL}	Low	I_{OL} = 10mA			.5				
V_{OL}	Low	I_{OL} = 8mA						.5	
V_{OH}	High	I_{OH} = −2mA	2.4			2.4			
	Input Current								μA
I_{IL}	Low	V_{IN} = 0.45V			−100			−150	
I_{IH}	High	V_{IN} = 5.5V			40			50	
	Output Current	V_{CC} = max							
I_{OLK}	Leakage (82S150)	V_{OUT} = 5.5V			40			60	μA
$I_{O(OFF)}$	Hi-Z state (82S151)	V_{OUT} = 5.5V			40			60	μA
		V_{OUT} = .45V			−40			−60	
I_{OS}	Short circuit (82S151)[4,5]	V_{OUT} = 0V	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current	V_{CC} = max		130	155		130	155	mA
	Capacitance	V_{CC} = 5V							pF
C_{IN}	Input	V_{IN} = 2.0V		8			8		
C_B	I/O	V_B = 2.0V		15			15		

AC ELECTRICAL CHACTERISTICS

$R_1 = 470\Omega$, $R_2 = 1K\Omega$
N82S150/151: $0°C \le T_A \le +75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S150/151: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TO	FROM	TEST CONDITIONS	N82S150/151 Min	N82S150/151 Typ[2]	N82S150/151 Max	S82S150/151 Min	S82S150/151 Typ[2]	S82S150/151 Max	UNIT
T_{PD}	Progagation delay	Output ±	Input ±	C_L = 30pF		20	25		20		ns
T_{OE}	Output enable	Output−	Input ±			20	25		20		ns
T_{OD}	Output disable[7]	Output+	Input ±	C_L = 5pF		20	25		20		

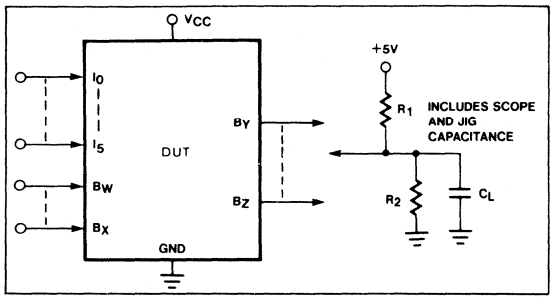
NOTES

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device of these or any other condition above those indicated in the operation of the device specifications is not implied.
2. All typical values are at V_{CC} = 5V, T_A = 25°C.
3. All voltage values are with respect to network ground terminal.
4. Test one at a time.
5. Duration of short circuit should not exceed 1 second.
6. Measured at $V_T = V_{OL} + 0.5V$.

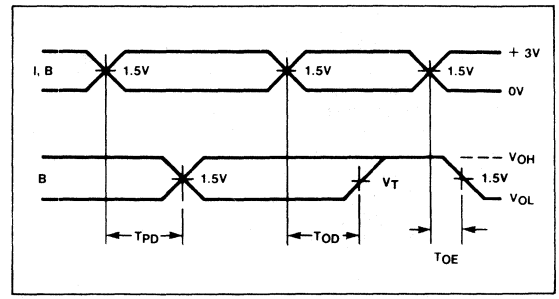
Preview

INTEGRATED FUSE LOGIC
SERIES 20

TEST LOAD CIRCUIT



TIMING DIAGRAM



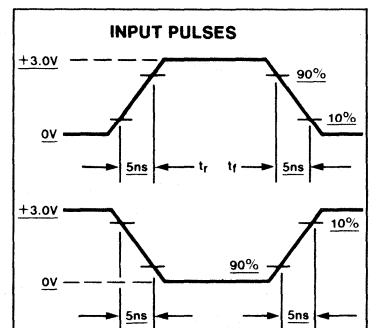
TIMING DEFINITIONS

TPD Propagation delay between input and output.

TOD Delay between input change and when output is off (Hi-Z or High).

TOE Delay between input change and when output reflects specified output level.

VOLTAGE WAVEFORM



Measurements: All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

FIELD PROGRAMMABLE GATE ARRAY (18×15×12) 82S150 (O.C.)/82S151 (T.S.)

Preview

INTEGRATED FUSE LOGIC
SERIES 20

LOGIC PROGRAMMING

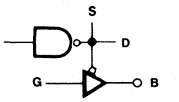
In a virgin device all Ni-Cr links are intact.

The FPGA can be programmed by means of Logic programming equipment.

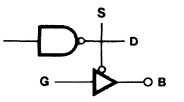
With Logic programming, the AND/EX-OR gate input connections necessary to implement the desired logic function are coded directly from logic equations using the Program Table on the following page.

In this Table the logic state of variables I and B associated with each gate Gn, Dn is assigned a symbol which results in the proper fusing pattern of corresponding links defined as follows:

I/O DIRECTION—(B)

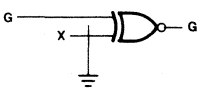


DIRECTION	CODE
I/O	A

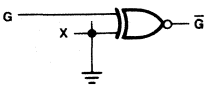


DIRECTION	CODE
INPUT	●

OUTPUT POLARITY — (B)

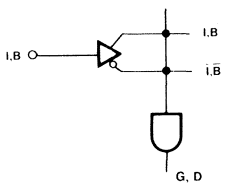


ACTIVE LEVEL	CODE
HIGH	H

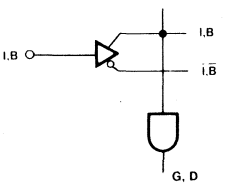


ACTIVE LEVEL	CODE
LOW	L

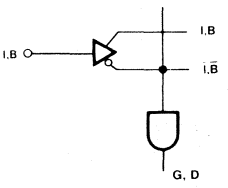
"AND" ARRAY—(I,B)



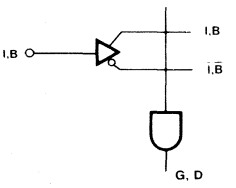
STATE	CODE
INACTIVE[1,2]	O



STATE	CODE
I,B	H



STATE	CODE
$\bar{I},\bar{B}$	L



STATE	CODE
DON'T CARE	—

NOTES:
1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates Gn, Dn.
2. Any gate Gn, Dn will be unconditionally inhibited if any one of the (I, B) link pairs is left intact.

Preview

INTEGRATED FUSE LOGIC
SERIES 20

FPGA PROGRAM TABLE (Logic)

CUSTOMER NAME _____

PURCHASE ORDER # _____

SIGNETICS DEVICE # _____ CF (XXXX) _____

CUSTOMER SYMBOLIZED PART # _____

TOTAL NUMBER OF PARTS _____

PROGRAM TABLE # _____ REV _____ DATE _____

D2 =
G0 =
G1 =
G2 =
G3 =
D1 =
G4 =
G5 =
G6 =
G7 =
D0 =
G8 =
G9 =
G10 =
G11 =

NOTES

1. The FPGA is shipped with all links intact. Thus a background of entries corresponding to states of virgin links exists in the table, shown BLANK for clarity.
2. Unused I and B bits are normally programmed Don't Care (—).
3. Unused Gates can be left blank.
4. Disregard Polarity for Gates used only as inputs.

PROGRAM TABLE ENTRIES

AND: INACTIVE 0; I, B H; I, B L; Don't Care —

I, B

CONTROL: (POL.) HIGH H, LOW L; (DIR.) I/O A, INPUT •

PIN	GATE	DIR.	POL.	AND I 5	I 4	I 3	I 2	I 1	I 0	B (I) 11	10	9	8	7	6	5	4	3	2	1	0
	D2																				
6	00																				
7	01																				
8	02																				
9	03																				
	D1																				
11	04																				
12	05																				
13	06																				
14	07																				
	D0																				
15	08																				
16	09																				
17	10																				
18	11																				
		PIN		5	4	3	2	1	19	18	17	16	15	14	13	12	11	9	8	7	6
VARIABLE NAME																					

FIELD PROGRAMMABLE LOGIC ARRAY (18×42×10)

82S152 (O.C.)/82S153 (T.S.)
82S152A (O.C.)/82S153A (T.S.)

INTEGRATED FUSE LOGIC
SERIES 20

DESCRIPTION

The 82S152 and 82S153 are two-level logic elements, consisting of 32 AND gates and 10 OR gates with fusible link connections for programming I/O polarity and direction.

All AND gates are linked to 8 inputs (I) and 10 bidirectional I/O lines (B). These yield variable I/O gate configurations via 10 direction control gates (D), ranging from 18 inputs to 10 outputs.

On chip T/C buffers couple either True (I, B) or Complement ($\overline{I}$, $\overline{B}$) input polarities to all AND gates, whose outputs can be optionally linked to all OR gates. Their output polarity, in turn, is individually programmable through a set of EX-OR gates for inplementing AND-OR or AND-NOR logic functions.

The 82S152 and the 82S153 are field programmable, enabling the user to quickly generate custom patterns using standard programming equipment.

Both devices are available in a 20 pin slim line package. For the commercial temperature range (0°C to +75°C) specify N82S152/153 N or F and N82S152A/153A N or F. For the military temperature range (−55°C to +125°C) specify S82S152/153 F only and S82S152A/153A F only.

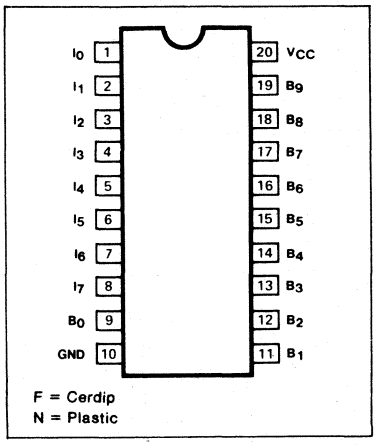
FEATURES

- Field programmable (Ni-Cr links)
- 8 inputs
- 32 AND gates
- 10 OR gates
- 10 bidirectional I/O lines
- Active high or low outputs
- 42 Product Terms:
  - 32 Logic terms
  - 10 Control terms
- I/O propagation delay:
  - N82S152/153: 40ns (max)
  - N82S152A/153A: 30ns (max)
  - S82S152/153: 60ns (max)
  - S82S152A/153A: 45ns (max)
- Input loading
  - N82S152/153: −100μA (max)
  - S82S152/153: −150μA (max)
- Power dissipation:
  - 650mW (typ)
- Output options:
  - 82S152: open collector
  - 82S153: tri-state
- TTL compatible

APPLICATIONS

- Random logic
- Code converters
- Fault detectors
- Function generators
- Address mapping
- Multiplexing

PIN CONFIGURATION



F = Cerdip
N = Plastic

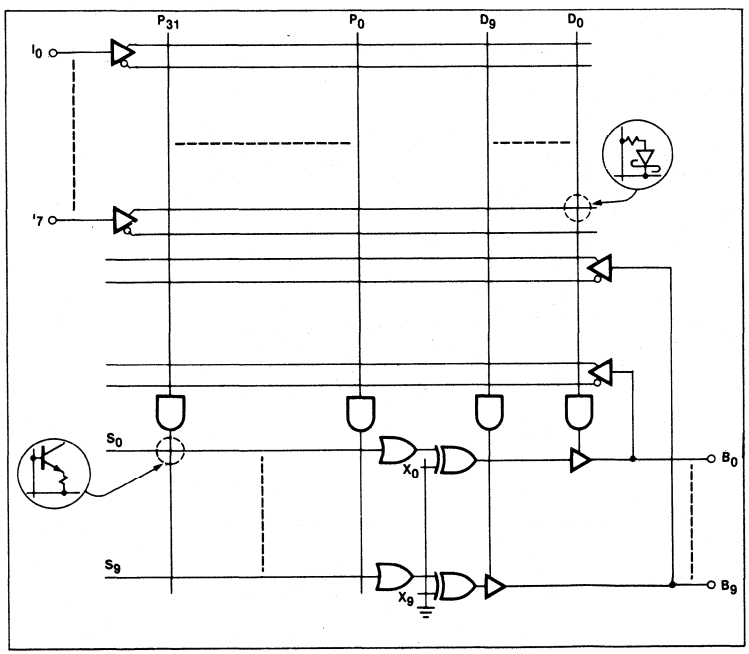
LOGIC FUNCTION

Typical product term:
$Pn = A \cdot \overline{B} \cdot C \cdot D \cdot \ldots\ldots$
Typical logic function:
At Output Polarity = H
$Z = P0 + P1 + P2 \ldots\ldots$

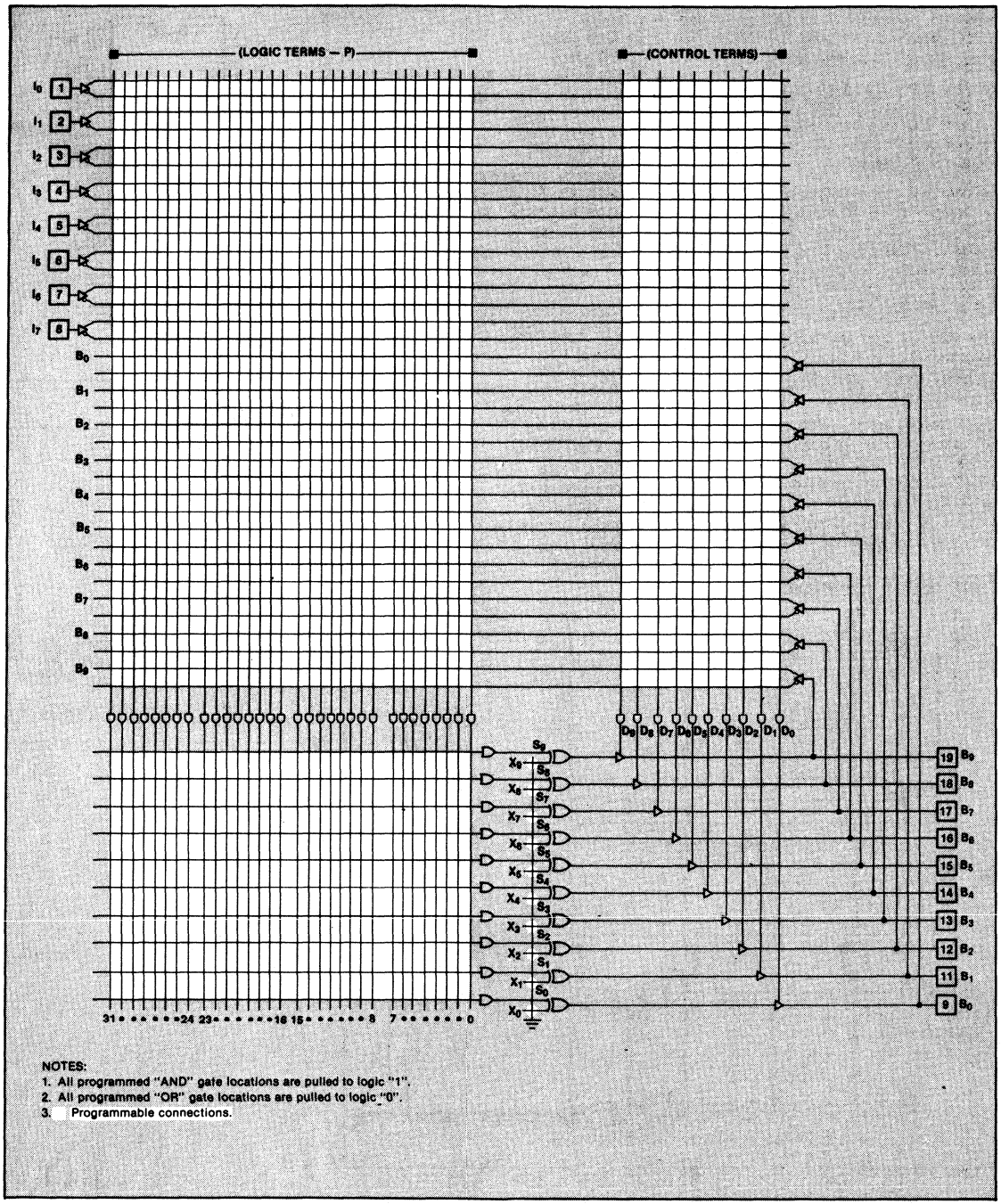
At Output Polarity = L
$\overline{Z = P0 + P1 + P2 + \ldots\ldots}$
$Z = P0 \cdot P1 \cdot P2 \ldots\ldots$

NOTES:
1. For each of the 10 outputs, either function Z (active-high) or Z (active-low) is available, but not both. The desired output polarity is programmed via the EX-OR gates.
2. Z, A, B, C, etc. are user defined connections to fixed inputs (I) and bidirectional pins (B).

FUNCTIONAL DIAGRAM



FPLA LOGIC DIAGRAM



NOTES:
1. All programmed "AND" gate locations are pulled to logic "1".
2. All programmed "OR" gate locations are pulled to logic "0".
3. Programmable connections.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER		RATING Min	RATING Max	UNIT
V_{CC}	Supply voltage		+7	Vdc
V_{IN}	Input voltage		+5.5	Vdc
V_{OUT}	Output voltage		+5.5	Vdc
I_{IN}	Input currents	−30	+30	mA
I_{OUT}	Output currents		+100	mA
	Temperature range			C°
T_A	Operating			
	N82S152/153/152A/153A	0	+75	
	S82S152/153/152A/153A	−55	+125	
T_{STG}	Storage	−65	+150	

THERMAL RATINGS

TEMPERATURE	Military	Commercial
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

DC ELECTRICAL CHARACTERISTICS

N82S152/153, N82S152A/153A: $0°C \le T_A \le +75°C$, $4.75 \le V_{CC} \le 5.25V$
S82S152/153, S82S152A/153A: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TEST CONDITION	N82S152/153 Min	N82S152/153 Typ[2]	N82S152/153 Max	S82S152/153 Min	S82S152/153 Typ[2]	S82S152/153 Max	UNIT
	Input voltage[3]								V
V_{IL}	Low	V_{CC} = min			.85			.80	
V_{IH}	High	V_{CC} = max	2.0			2.0			
V_{IC}	Clamp[3,4]	V_{CC} = min, Iin = −18mA		−.8	−1.2		−.8	−1.2	
	Output voltage	V_{CC} = min							V
V_{OL}	Low[3,5]	I_{OL} = 15mA			.5				
V_{OL}	Low[3,5]	I_{OL} = 12mA						.5	
V_{OH}	High[3,6]	I_{OH} = −2mA	2.4			2.4			
	Input current								μA
I_{IL}	Low	V_{IN} = 0.45V			−100			−150	
I_{IH}	High	V_{IN} = 5.5V			40			50	
	Output current	V_{CC} = max							
I_{OLK}	Leakage (82S152)	V_{OUT} = 5.5V			40			60	μA
$I_{O(OFF)}$	Hi-Z state (82S153)	V_{OUT} = 5.5V			40			60	μA
		V_{OUT} = .45V			−40			−60	
I_{OS}	Short circuit (82S153)[4,6,7]	V_{OUT} = 0V	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current[8]	V_{CC} = max		130	155		130	165	mA
	Capacitance	V_{CC} = 5V							pF
C_{IN}	Input	V_{IN} = 2.0V		8			8		
C_B	I/O	V_B = 2.0V		15			15		

AC ELECTRICAL CHACTERISTICS

$R_A = 470\Omega$, $R_2 = 1K\Omega$
N82S152/153, N82S152A/153A: $0°C \le T_A \le +75°C$, $4.75 \le V_{CC} \le 5.25V$
S82S152/153, S82S152A/153A: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TO	FROM	TEST CONDITIONS	N82S152/153 Min	N82S152/153 Typ	N82S152/153 Max	S82S152/153 Min	S82S152/153 Typ	S82S152/153 Max	UNIT
T_{PD}	Progagation delay	Output ±	Input ±	C_L = 30pF		30	40		30	55	ns
T_{OE}	Output enable	Output−	Input ±			25	35		25	45	
T_{OD}	Output disable[9]	Output+	Input ±	C_L = 5pF		25	35		25	45	ns

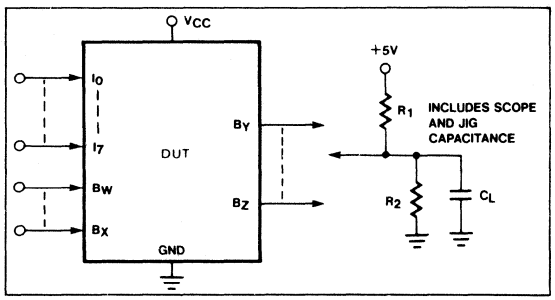
PARAMETER		TO	FROM	TEST CONDITIONS	N82S152A/153A Min	N82S152A/153A Typ	N82S152A/153A Max	S82S152A/153A Min	S82S152A/153A Typ	S82S152A/153A Max	UNIT
T_{PD}	Progagation delay	Output ±	Input ±	C_L = 30pF		20	30		20	45	ns
T_{OE}	Output enable	Output−	Input ±			20	30		20	40	
T_{OD}	Output disable[9]	Output+	Input ±	C_L = 5pF		20	30		20	40	ns

Notes on following page.

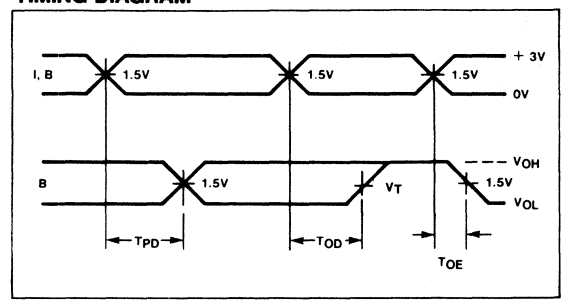
NOTES

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other condition above those indicated in the operation of the device specifications is not implied.
2. All typical values are at V_{CC} = 5V, T_A = 25°C
3. All voltage values are with respect to network ground terminal.
4. Test one at a time.
5. Measured with +10V applied to I_7.
6. Measured with +10V applied to I_{0-7}. Output sink current is supplied thru a resistor to V_{CC}.
7. Duration of short circuit should not exceed 1 second.
8. I_{CC} is measured with I_{0-7} and B_{0-9} at 4.5V.
9. Measured at $V_T = V_{OL} + 0.5V$.

TEST LOAD CIRCUIT



TIMING DIAGRAM



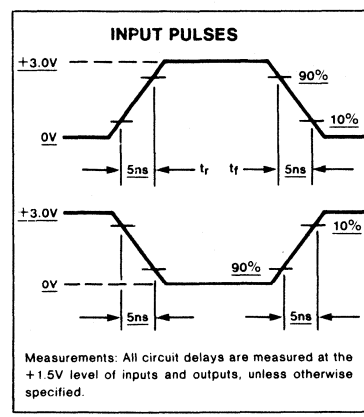
TIMING DEFINITIONS

T_{PD} Propagation delay between input and output.

T_{OD} Delay between input change and when output is off (Hi-Z or High).

T_{OE} Delay between input change and when output reflects specified output level.

VOLTAGE WAVEFORM



Measurements: All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

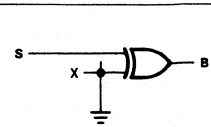
LOGIC PROGRAMMING

The FPLA can be programmed by means of Logic programming equipment.

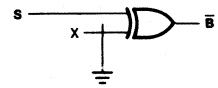
With Logic programming, the AND/OR/EX-OR gate input connections necessary to implement the desired logic function are coded directly from logic equations using the Program Table on the following page.

In this Table the logic state of variables I, P, and B associated with each Sum Term S is assigned a symbol which results in the proper fusing pattern of corresponding link pairs, defined as follows:

OUTPUT POLARITY — (B)

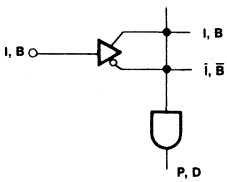


ACTIVE LEVEL	CODE
HIGH	H

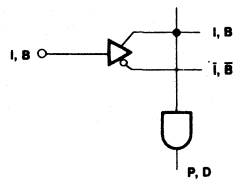


ACTIVE LEVEL	CODE
LOW	L

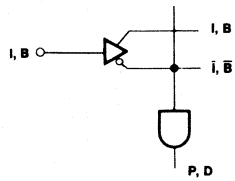
"AND" ARRAY — (I, B)



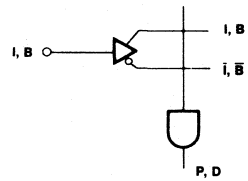
STATE	CODE
INACTIVE [1, 2]	O



STATE	CODE
I, B	H

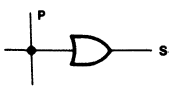


STATE	CODE
Ī, B̄	L

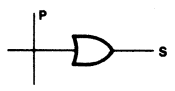


STATE	CODE
DON'T CARE	—

"OR" ARRAY — (B)



P_n STATUS	CODE
ACTIVE	A



P_n STATUS	CODE
INACTIVE	•

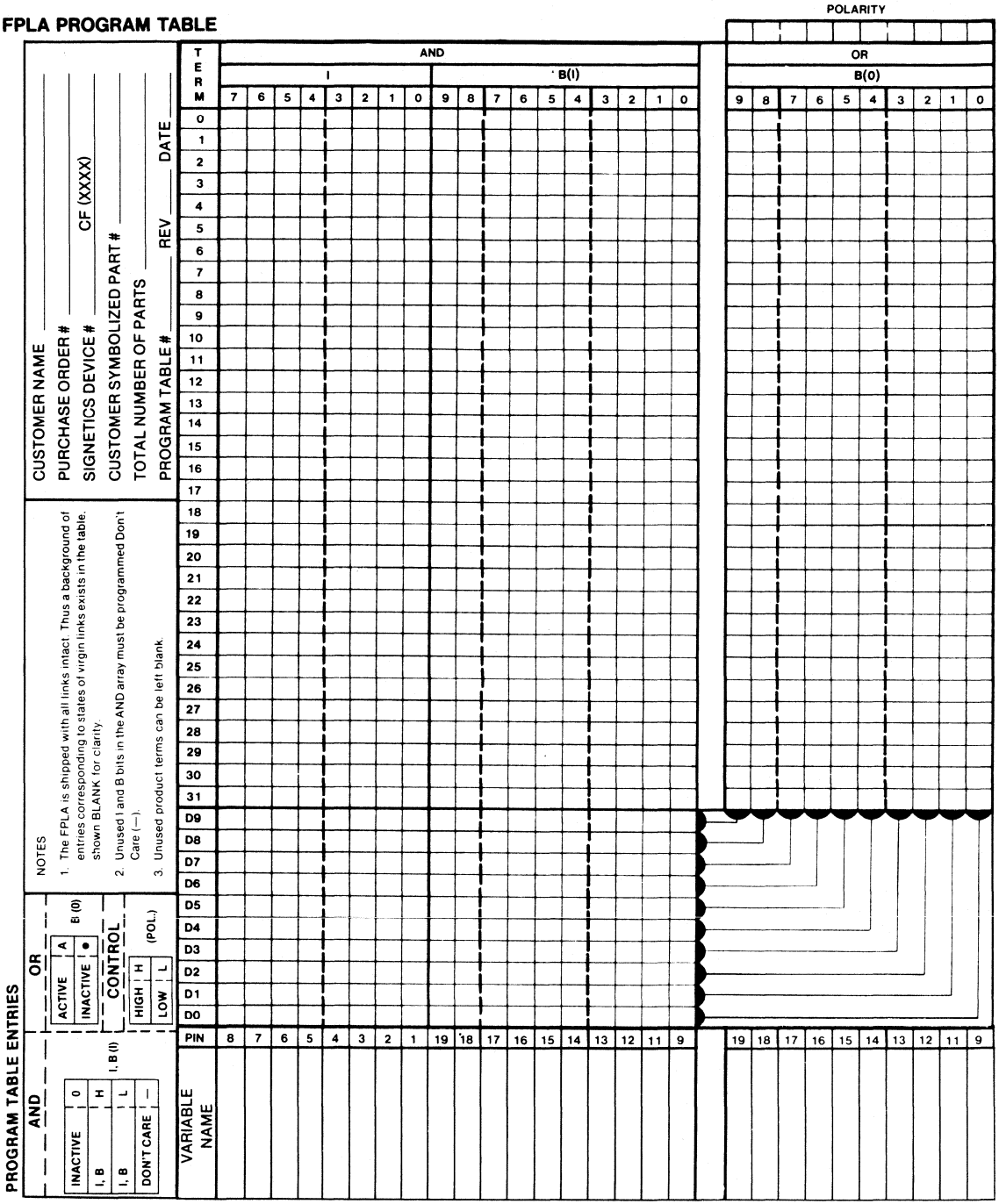
NOTES
1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates P_n, D_n.
2. Any gate P_n, D_n will be unconditionally inhibited if any one of its (I, B) link pairs is left intact.

VIRGIN STATE

A factory shipped virgin device contains all fusible links intact, such that:

1. All outputs at "H" polarity.
2. All P_n terms are disabled.
3. All P_n terms are active on all outputs.

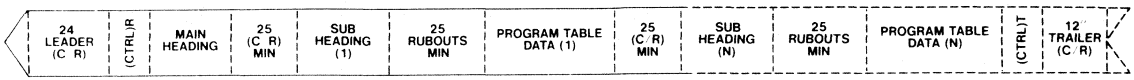
FPLA PROGRAM TABLE



TWX TAPE CODING (LOGIC FORMAT)

The FPLA Program Table can be sent to Signetics in ASCII code format via airmail using any type of 8-level tape (paper, mylar, fanfold, etc.), or via TWX: just dial (910) 339-9283, tell the operator to turn the paper puncher on, and acknowledge. At the end of transmission instruct the operator to send tape to Signetics Order Entry.

A number of Program Tables can be sequentially assembled on a continuous tape as follows, however,limit tape length to a roll of 1.75 inch inside diameter and 4.25 inch outside diameter.



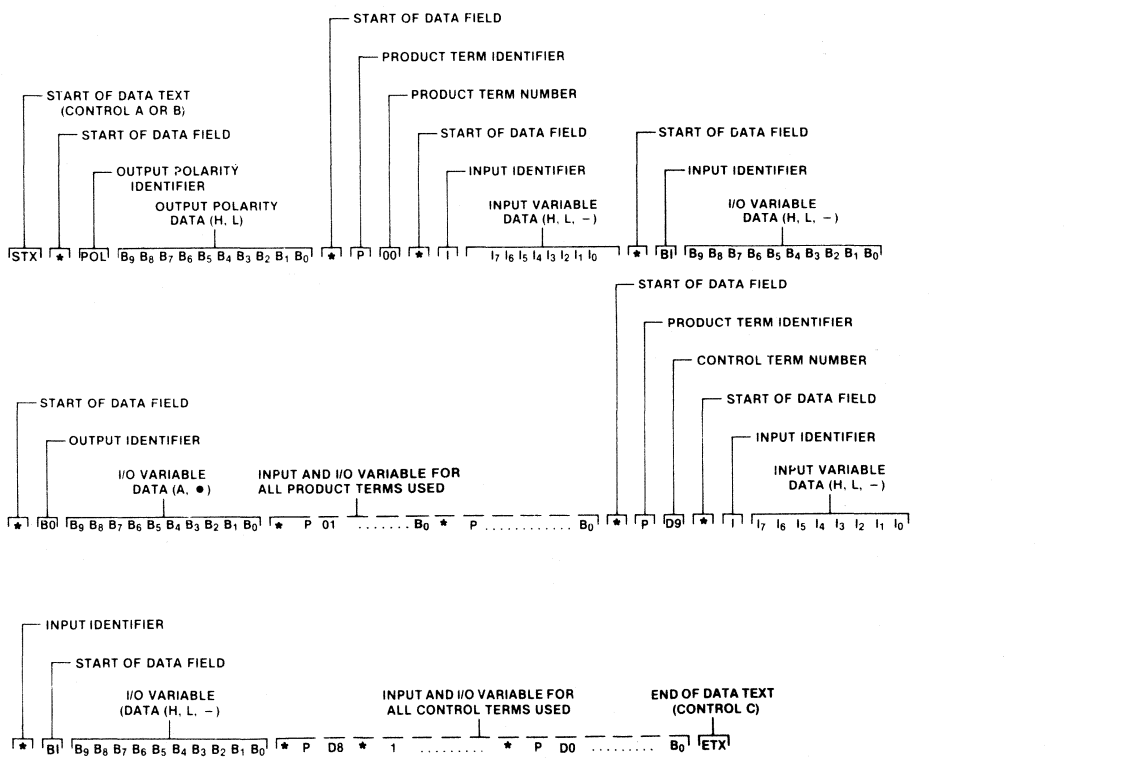
A. The MAIN HEADING at the beginning of tape includes the following information, with each entry preceded by a ($) character, whether used or not:

1. Customer Name _____
2. Customer TWX No. _____
3. Date _____
4. Purchase Order No. _____
5. Number of Program Tables _____
6. Total Number of Parts _____

B. Each SUB HEADING should contain specific information pertinent to each Program Table as follows, with each entry preceded by a ($) character, whether used or not:

1. Signetics Device No. _____
2. Program Table No. _____
3. Revision _____
4. Date _____
5. Customer Symbolized Part No. _____
6. Number of Parts _____

C. Program Table data blocks are initiated with an STX character, and terminated with an ETX character. The body of the data consists of output polarity, product term, and output information separated by appropriate identifiers in accordance with the following format. Entries for the data fields correspond to those defined in the Logic PROGRAM TABLE:



DESCRIPTION

The 82S154/155/156/157/158/159 are Open Collector and Tri-state registered logic elements combining AND/OR gate arrays with clocked J/K flip-flops, dynamically convertible to D-type via a "foldback" inverting buffer and control gate F_C. They all have similar organization, featuring respectively 4, 6, or 8 registered I/O outputs (F), in conjunction with 8, 6, or 4 bidirectional I/O lines (B). These yield variable I/O gate and register configurations via control gates (D, L) ranging from 16 inputs to 12 outputs.

The AND/OR arrays consist of 32 logic AND gates, 13 control AND gates, and 21 OR gates with fusible link connections for programming I/O polarity and direction. All AND gates are linked to 4 inputs (I), bidirectional I/O lines (B), internal flip-flop outputs (Q), and Complement Array output ($\overline{C}$). The Complement Array consists of a NOR gate optionally linked to all AND gates for generating and propagating complementary AND terms.

On chip I/C buffers couple either True (I, B, Q) or Complement ($\overline{I}$, $\overline{B}$, $\overline{Q}$, $\overline{C}$) input polarities to all AND gates, whose outputs can be optionally linked to all OR gates. One group of OR gates drives bidirectional I/O lines (B), whose output polarity is individually programmable through a set of EX-OR gates for implementing AND-OR or AND-NOR logic functions. Another group drives the J-K inputs of all flip-flops, as well as asynchronous Preset and Reset lines (P, R), (except the 82S158/159, where AND functions are provided).

All flip-flops are positive edge trigger and can be used as input, output, or I/O (for interfacing with a bidirectional data bus) in conjunction with load control gates (L), steering inputs (I), (B), (Q) and programmable output select lines (E).

The 82SXXX are field programmable, enabling the user to quickly generate custom patterns using standard programming equipment.

All devices are available in a 20-pin, slim line package. For the commercial temperature range (0°C to +75°C) specify N82SXXX N or F. For the military temperature range (−55°C to +125°C) specify S82SXXX F only.

FEATURES

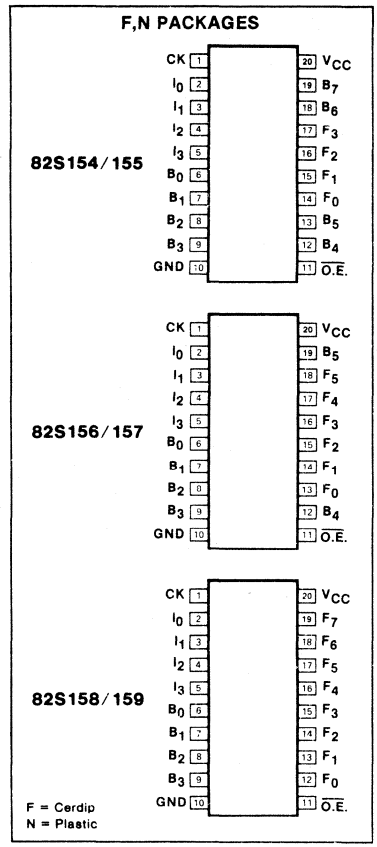
- **Field programmable (Ni-Cr link)**
- **4 Inputs**
- **13 control gates**
- **32 AND gates**
- **21 OR gates**
- **45 product terms:**
  32 logic terms
  13 control terms
- **Bidirectional I/O lines:** 82S154/155—8
  82S156/157—6
  82S158/159—4
- **Bidirectional Registers:** 82S154/155—4
  82S156/157—6
  82S158/159—8
- **J/K, T, or D-type flip-flops**
- **Asynchronous Preset/Reset**
- **Complement Array**
- **Active high or low outputs**
- **Programmable $\overline{O.E.}$ control**
- **Positive edge trigger clock**
- **Power-on reset on all flip-flops (F_n = "1")**
- **Clock frequency:** N82SXXX: 15 MHz (max)
  S82SXXX: MHz (max)
- **Input loading:** N82SXXX: −100μA (max)
  S82SXXX: −150μA (max)
- **Power dissipation: 650mW (typ)**
- **TTL Compatible**

APPLICATIONS

- **Random sequential logic**
- **Synchronous up/down counters**
- **Shift registers**
- **Bidirectional data buffers**
- **Timing function generators**
- **System controllers/synchronizers**
- **Priority encoder/registers**

PIN CONFIGURATION

F,N PACKAGES



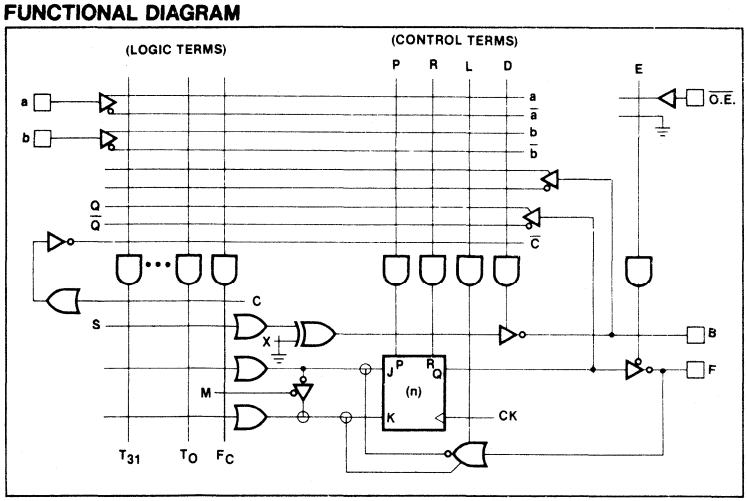
FLIP-FLOP TRUTH TABLE

V_{CC}	$\overline{O.E.}$	L	CK	P	R	J	K	Q	F
	H								H/HI-Z
↑	L	X	X	L	X	X	X	L	H
+5	L	X	X	H	L	X	X	H	L
	L	X	X	L	H	X	X	L	H
	L	L	↑	L	L	L	L	Q	$\overline{Q}$
	L	L	↑	L	L	L	H	L	H
	L	L	↑	L	L	H	L	H	L
	L	L	↑	L	L	H	H	$\overline{Q}$	Q
	H	H	↑	L	L	L	H	L	H*
	H	H	↑	L	L	H	L	H	L*
	+10V	X	↑	X	X	L	H	L	H**
		X	↑	X	X	H	L	H	L**

NOTES

1. Positive Logic:
   $J/K = T_0 + T_1 + T_2 \ldots\ldots T_{47}$
   $T_n = \overline{C} \bullet (I_0 \bullet I_1 \bullet I_2 \ldots) \bullet (Q_0 \bullet Q_1 \ldots) \bullet (B_0 \bullet B_1 \bullet \ldots)$
2. ↑ denotes transition from Low to High level.
3. X = Don't Care
4. * = Forced at F_n pin for loading J/K flip-flop in I/O mode. L must be enabled, and other active T_n disabled via steering input(s) I, B, or Q.
5. At P = R = H, Q = H. The final state of Q depends on which is released first.
6. * * = Forced at F_n pin to load J/K flip-flop independent of program code (Diagnostic mode).

FUNCTIONAL DIAGRAM



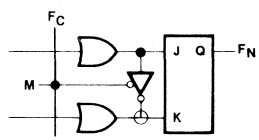
PIN DESIGNATION

PIN NO.	SYMBOL	NAME AND FUNCTION	POLARITY
1	CK	**CLOCK** The clock input to all flip-flops. A Low-to-High transition on this line is necessary to update the contents of flip-flops.	Active-High
2-5	I_{0-3}	**INPUTS** Fixed logic inputs to the AND array	Active-High/Low (user defined)
6-9	B_{0-3}	**STATIC I/O PINS** Bidirectional external inputs to the AND array, or outputs from the OR array, programmable via control gates D_{0-3}.	Active-High/Low (user defined)
11	$\overline{O.E.}$	**OUTPUT ENABLE** Provides output enable functions E_A and E_B to flip-flop banks F_{0-3} and F_{4-7} respectively. May be programmed for external control, enable, or disable of flip-flop outputs.	Active-Low
12, 13	$(B_n/F_n)^*$	**STATIC OR REGISTERED I/O PINS** Bidirectional static or registered I/O pins, dependent on device configuration.	Active-High/Low (user defined)
14-17	$(F_n)^*$	**REGISTERED I/O PINS** Bidirectional flip-flop outputs or direct load inputs, selected via control gates L_{A-B} in conjunction with E_{A-B} output enables.	Active-Low
18, 19	$(B_n/F_n)^*$	**STATIC OR REGISTERED I/O PINS** Same as 12, 13.	Active-High/Low (user defined)

(*) Value of (n) is device dependent. Refer to circuit diagrams.

FLIP-FLOP FUNCTION

The output flip-flops (F_N) are programmable via the Flip-Flop Control Term (F_C). A tri-state inverter is positioned between the J-K inputs with a wire-OR connection at the K terminal. If (F_C) is programmed active (A) as shown, then F_C controls the output state of the tri-state inverter.



When F_C = High, the flip-flop functions as J-K type. When F_C = Low, the flip-flop functions as D-type.

If F_C is programmed as a (•), control M = High unconditionally, and the flip-flop is permanently defined as a J-K.

LOGIC FUNCTION

COMBINATORIAL

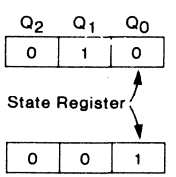
Typical Product Term: $P_n = A \bullet \overline{B} \bullet \overline{C} \bullet D \bullet \overline{E} \bullet \ldots.$

Typical Output Function: Active High at D_Y = "1": $Y = P_0 + P_1 + P_2 + \ldots.$

Active Low: $Y = \overline{P_0 + P_1 + P_2 + \ldots.} = \overline{P}_0 \bullet \overline{P}_1 \bullet \overline{P}_2 \bullet \ldots.$

SEQUENTIAL (J/K type)

Typical State Transition:



SET Q_0: $J_0 = (\overline{Q}_2 \bullet Q_1 \bullet \overline{Q}_0) \bullet \overline{A} \bullet \overline{B} \bullet C \ldots$
$K_0 = 0$

RESET Q_1: $J_1 = 0$
$K_1 = (\overline{Q}_2 \bullet Q_1 \bullet \overline{Q}_0) \bullet \overline{A} \bullet \overline{B} \bullet C \ldots$

HOLD Q_2: $J_2 = 0$
$K_2 = 0$

NOTES

1. For each of the combinatorial outputs, either function Y (active-high) or $\overline{Y}$ (active-low) is available, but not both. The desired output polarity is programmed via the EX-OR gates.
2. Y, A, B, C, etc. are user defined connections to fixed inputs (I), bidirectional pins (B), "foldback" register outputs (Q), and Complement Array ($\overline{C}$).
3. Sequential state transitions occur on the positive edge of clock. External flip-flop outputs are given by $F_n = \overline{Q}_n$.
4. For D-type flip-flops, $K_n = \overline{J}_n$. For T-type flip-flops, $K_n = J_n$.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER		RATING Min	RATING Max	UNIT
V_{CC}	Supply voltage		+7	Vdc
V_{IN}	Input voltage		+5.5	Vdc
V_{OUT}	Output voltage		+5.5	Vdc
I_{IN}	Input currents	−30	+30	mA
I_{OUT}	Output currents		+100	mA
	Temperature range			°C
T_A	Operating			
	N82S154/5/6/7/8/9	0	+75	
	S82S154/5/6/7/8/9	−55	+125	
T_{STG}	Storage	−65	+150	

THERMAL RATINGS

TEMPERATURE	MILITARY	COMMERCIAL
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

DC ELECTRICAL CHARACTERISTICS

N82S154/5/6/7/8/9: $0° \leq T_A \leq +75°C$, $4.75V \leq V_{CC} \leq 5.25V$
S82S154/5/6/7/8/9: $-55°C \leq T_A \leq +125°C$, $4.5V \leq V_{CC} \leq 5.5V$

PARAMETER		TEST CONDITIONS	N82S154/5/6/7/8/9 Min	N82S154/5/6/7/8/9 Typ[2]	N82S154/5/6/7/8/9 Max	S82S154/5/6/7/8/9 Min	S82S154/5/6/7/8/9 Typ[2]	S82S154/5/6/7/8/9 Max	UNIT
	Input voltage[3]								V
V_{IH}	High	V_{CC} = Max	2			2			
V_{IL}	Low	V_{CC} = Min			0.85			0.8	
V_{IC}	Clamp	V_{CC} = Min, I_{IN} = −18mA		−0.8	−1.2		−0.8	−1.2	
	Output voltage[3]	V_{CC} = Min							V
V_{OH}	High (82S155/7/9)	I_{OH} = −2mA	2.4			2.4			
V_{OL}	Low	I_{OL} = 10mA		0.35	0.5				
V_{OL}	Low	I_{OL} = 10mA					0.35	0.5	
	Input current								μA
I_{IH}	High	V_{IN} = 5.5V		<1	40		<1	50	
I_{IL}	Low	V_{IN} = 0.45V		−10	−100		−10	−150	
I_{IL}	Low (CK input)	V_{IN} = 0.45V		−50	−250		−50	−350	
	Output current	V_{CC} = Max							
I_{OLK}	Leakage[5]	V_{OUT} = 5.5V		1	40		1	60	μA
$I_{O(OFF)}$	Hi-Z state (82S155/7/9)[7]	V_{OUT} = 5.5V		1	40		1	60	μA
		V_{OUT} = 0.45V		−1	−40		−1	−60	
I_{OS}	Short circuit (82S155/7/9)[4,6]	V_{OUT} = 0V	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current[7]	V_{CC} = Max		150	190		150	190	mA
	Capacitance	V_{CC} = 5.0V							pF
C_{IN}	Input	V_{IN} = 2.0V		8			8		
C_{OUT}	Output	V_{OUT} = 2.0V		15			15		

NOTES

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device of these or any other condition above those indicated in the operation of the device specifications is not implied.
2. All typical values are at V_{CC} = 5V, T_A = 25°C.
3. All voltage values are with respect to network ground terminal.
4. Test one at a time.
5. Measured with V_{IH} applied to $\overline{O.E.}$.
6. Duration of short circuit should not exceed 1 second.
7. I_{CC} is measured with the $\overline{O.E.}$ input grounded, all other inputs at 4.5V and the outputs open.

Advance Information

INTEGRATED FUSE LOGIC
SERIES 20

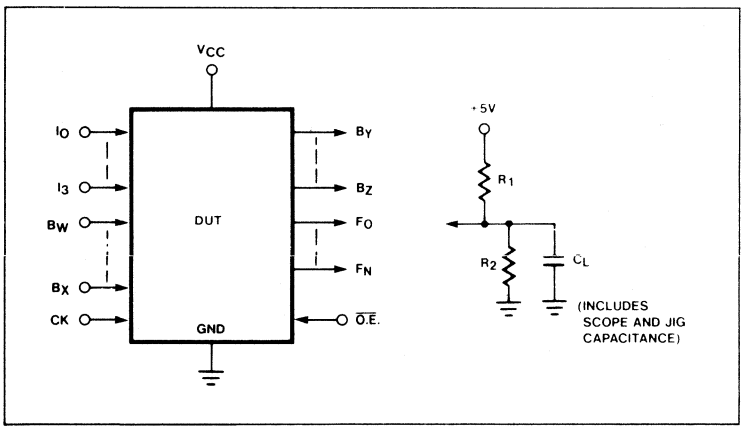
AC ELECTRICAL CHARACTERISTICS

$R_1 = 470\Omega$, $R_2 = 1k\Omega$
N82S154/5/6/7/8/9: $0°C \leq T_A \leq +75°C$, $4.75V \leq V_{CC} \leq 5.25V$
S82S154/5/6/7/8/9: $-55°C \leq T_A \leq +125°C$, $4.5V \leq V_{CC} \leq 5.5V$

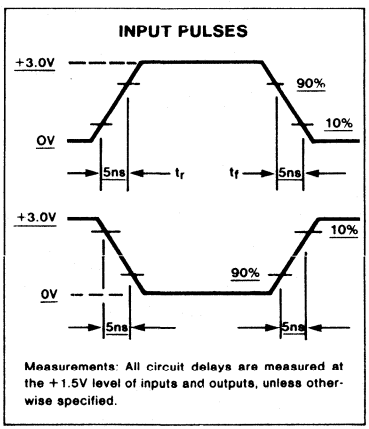
	PARAMETER	TO	FROM	TEST CONDITIONS	N82S154/5/6/7/8/9/ Min	N82S154/5/6/7/8/9/ Typ[1]	N82S154/5/6/7/8/9/ Max	S82S154/5/6/7/8/9 Min	S82S154/5/6/7/8/9 Typ[1]	S82S154/5/6/7/8/9 Max	UNIT
	Pulse width										ns
T_{CKH}	Clock[2] high	CK−	CK+			20			20		
T_{CKL}	Clock low	CK+	CK−			20			20		
T_{CKP}	Period	CK+	CK+			50			50		
T_{PRH}	Preset/Reset pulse	(I,B)+	(I,B)−			30			20		
	Set up time										ns
T_{IS1}	Input	CK+	(I,B)±			30			30		
T_{IS2}	Input (through F_n)	CK+	F±	C_L = 30pF		10			5		
T_{IS3}	Input (through Complement array)[4]	CK+	(I,B)±			40			40		
	Hold time										ns
T_{IH1}	Input	CK+	(I,B)±			−10			−10		
T_{IH2}		CK+	F±			0			−5		
	Propagation delay										ns
T_{CKO}	Clock	F±	CK+			25			25		
T_{OE1}	Output enable	F−	$\overline{O.E.}$−			20			20		
T_{OD1}	Output disable[3]	F+	$\overline{O.E.}$+	C_L = 5pF		20			20		
T_{PD}	Output	B±	(I,B)±	C_L = 30pF		40			35		
T_{OE2}	Output enable	B±	(I,B)+			35			35		
T_{OD2}	Output disable[3]	B+	(I,B)−	C_L = 5pF		35			35		
T_{PRO}	Preset/Reset	F±	(I,B)+	C_L = 30pF		50			50		
T_{PPR}	Power-on preset	F−	V_{CC}+			0			0		

NOTE
1. All typical values are at V_{CC} = 5V, T_A = 25°C.
2. To prevent spurious clocking, clock rise time (10%-90%) ≤ 10ns.
3. Measured at $V_T = V_{OL} + 0.5$ V.
4. When using the Complement Array T_{CKP} = 75ns (min).

TEST LOAD CIRCUIT

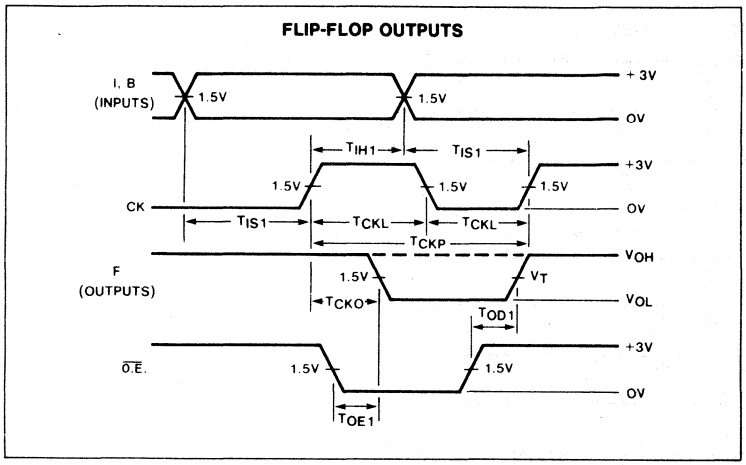


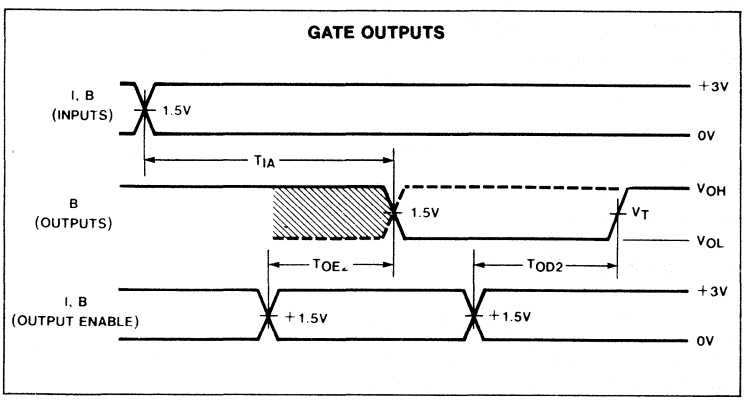
VOLTAGE WAVEFORM

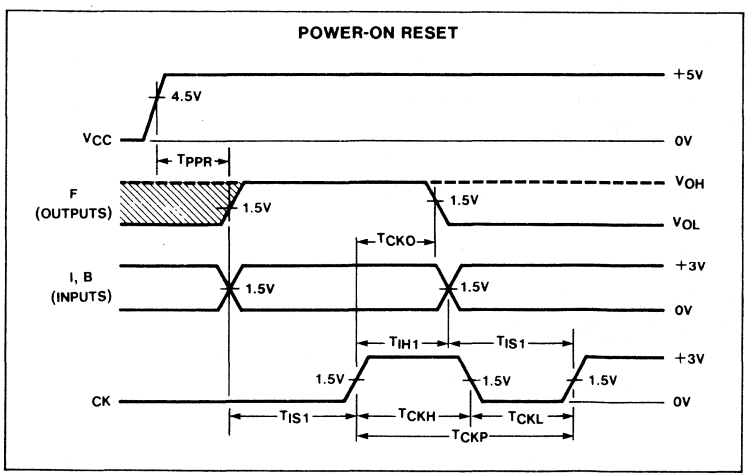


Measurements: All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

TIMING DIAGRAMS





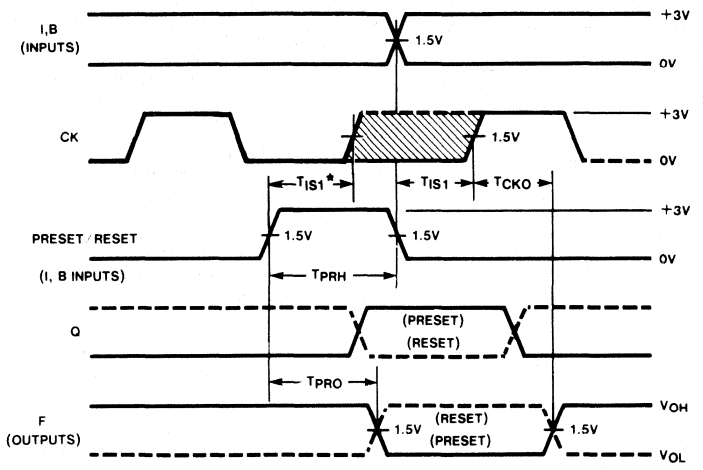


MEMORY TIMING DEFINITIONS

T_{CKH}	Width of input clock pulse.
T_{CKL}	Interval between clock pulses.
T_{CKP}	Clock period.
T_{PRH}	Width of preset input pulse.
T_{IS1}	Required delay between beginning of valid input and positive transition of clock.
T_{IS2}	Required delay between beginning of valid input forced at flip-flop output pins, and positive transition of clock.
T_{IH1}	Required delay between positive transition of clock and end of valid input data.
T_{IH2}	Required delay between positive transition of clock and end of valid input data forced at flip-flop output pins.
T_{CKO}	Delay between positive transition of clock and when Outputs become valid (with $\overline{O.E.}$ low).
T_{OE1}	Delay between beginning of Output Enable Low and when Outputs become valid.
T_{OD1}	Delay between beginning of Output Enable High and when Outputs are in the off state.
T_{PD}	Propagation delay between combinational inputs and outputs.
T_{OE2}	Delay between predefined Output Enable High, and when combinational Outputs become valid.
T_{OD2}	Delay between predefined Output Enable Low and when combinational Outputs are in the off state.
T_{PRO}	Delay between positive transition of predefined Preset/Reset input, and when flip-flop outputs become valid.
T_{PPR}	Delay between V_{CC} (after power-on) and when flip-flop outputs become preset at "1" (internal Q outputs at "0").

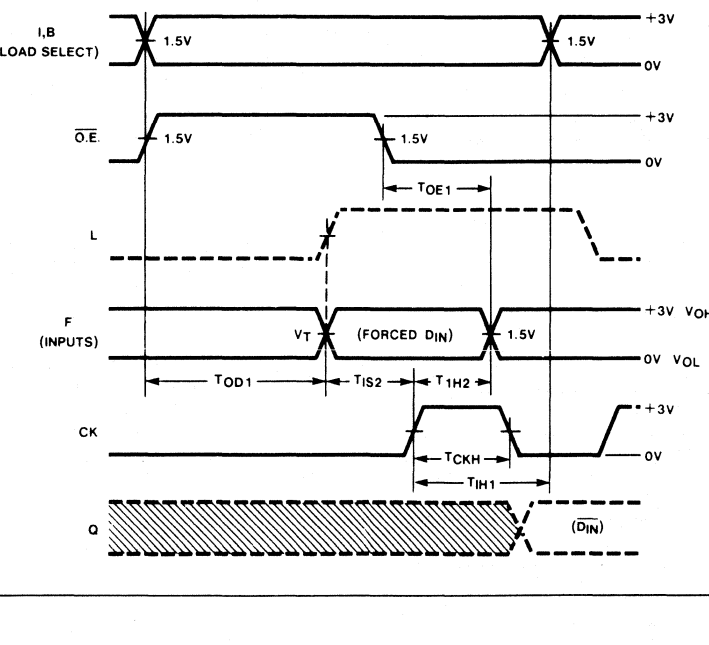
TIMING DIAGRAMS (Cont'd)

ASYNCHRONOUS PRESET/RESET



*Preset and Reset functions override Clock. However, F outputs may glitch with the first positive Clock Edge if T_{IS1} cannot be guaranteed by the user.

FLIP-FLOP INPUT MODE

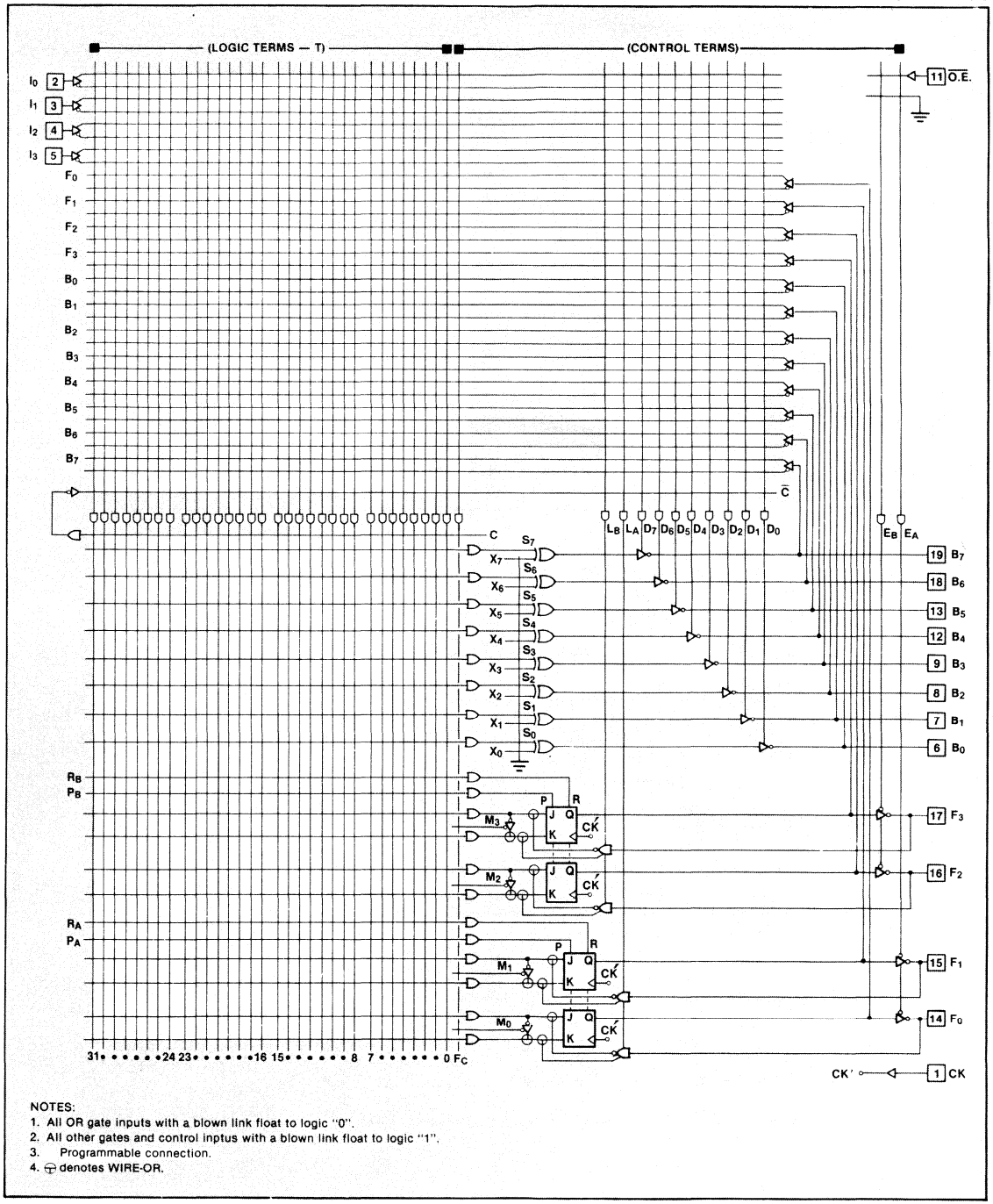


Preview

INTEGRATED FUSE LOGIC
SERIES 20

FPLS LOGIC DIAGRAM

82S154/155



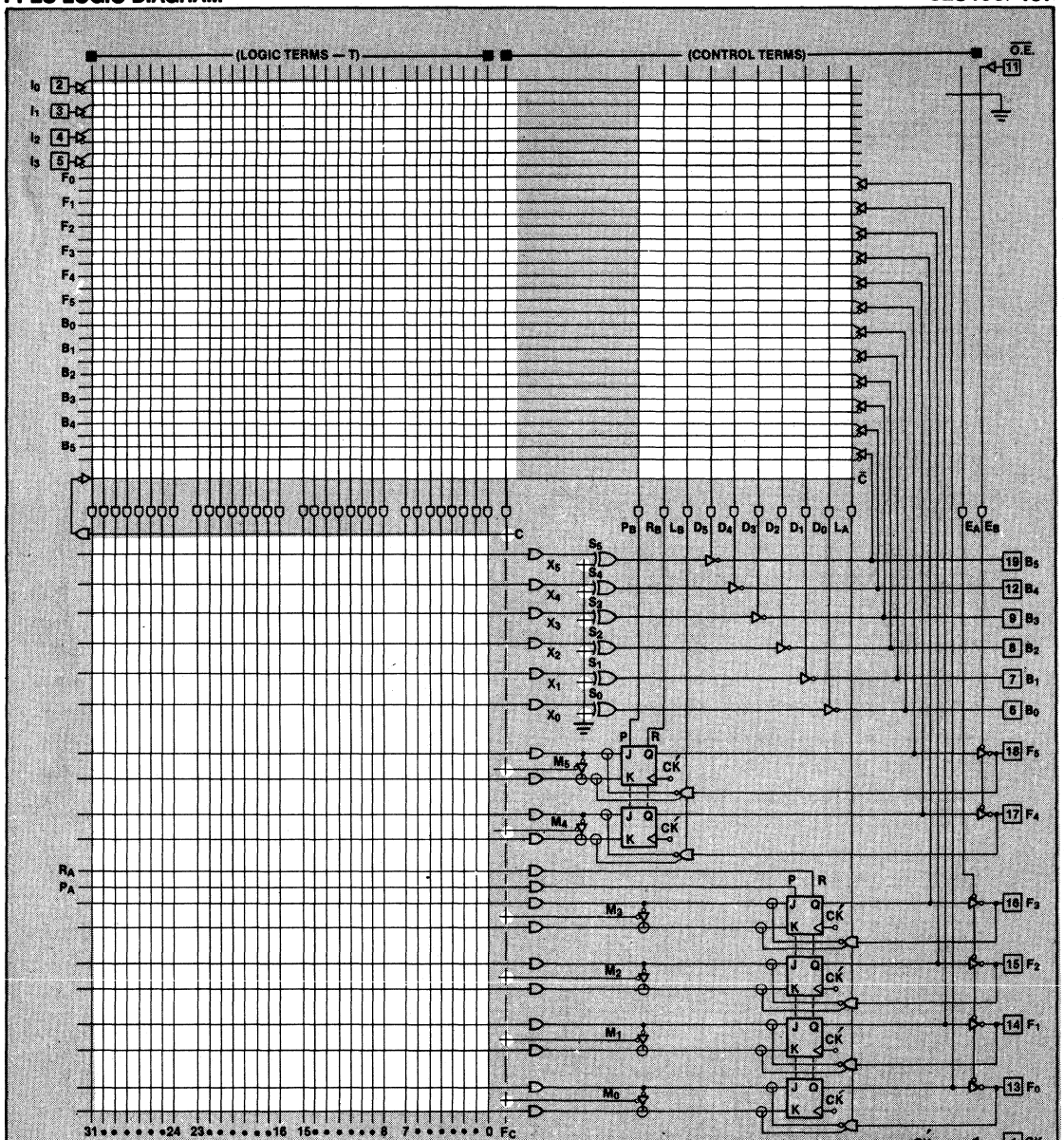
NOTES:
1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inptus with a blown link float to logic "1".
3. ■ Programmable connection.
4. ⊕ denotes WIRE-OR.

Preview

INTEGRATED FUSE LOGIC
SERIES 20

FPLS LOGIC DIAGRAM

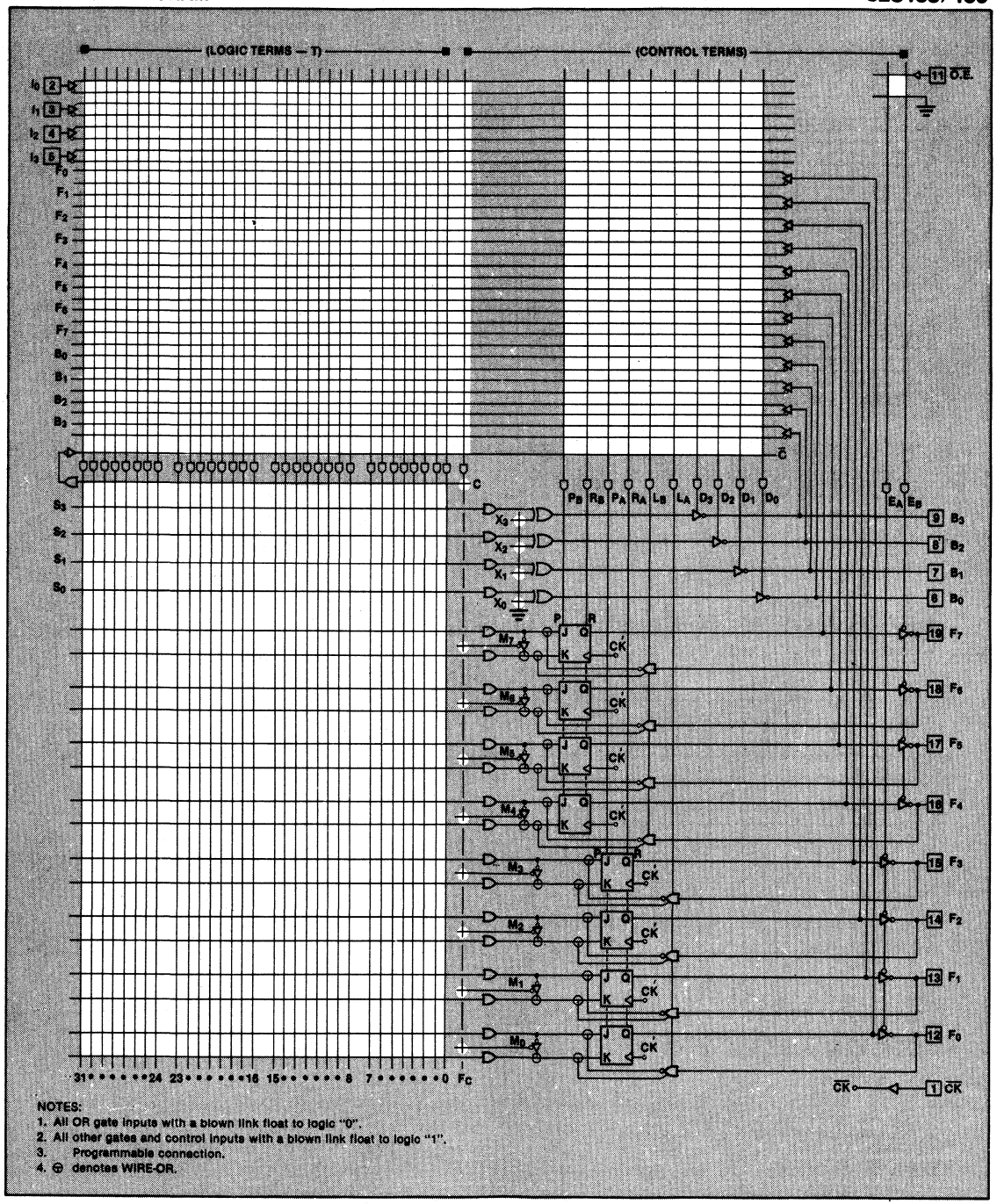
82S156/157



NOTES:
1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inputs with a blown link float to logic "1".
3. Programmable connection.
4. ⊖ denotes WIRE-OR.

FPLS LOGIC DIAGRAM

82S158/159



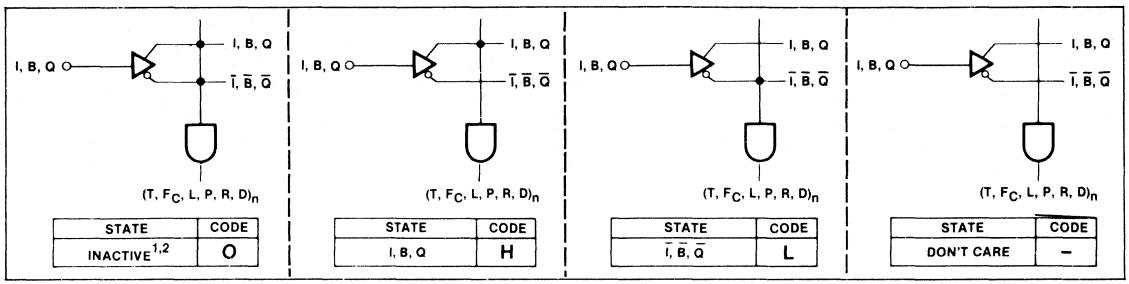
NOTES:
1. All OR gate inputs with a blown link float to logic "0".
2. All other gates and control inputs with a blown link float to logic "1".
3. ▪ Programmable connection.
4. ⊕ denotes WIRE-OR.

The FPLS can be programmed by means of Logic Programming equipment.

With Logic programming, the AND/OR-EX-OR input connections necessary to implement the desired logic function are coded directly from the State Diagram using the Program Tables on the following pages.

In these Tables, the logic state or action of all I/O, control, and state variables is assigned a symbol which results in the proper fusing pattern of corresponding links defined as follows:

"AND" ARRAY – (I), (B), (Qp)



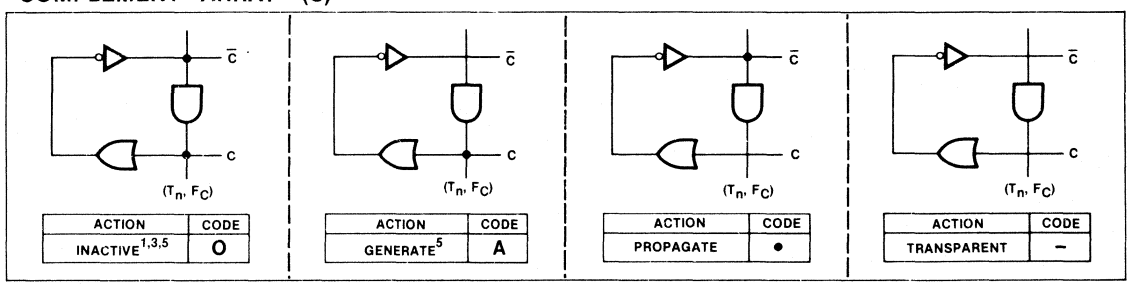
STATE	CODE
INACTIVE[1,2]	O

STATE	CODE
I, B, Q	H

STATE	CODE
$\overline{I}$, $\overline{B}$, $\overline{Q}$	L

STATE	CODE
DON'T CARE	–

"COMPLEMENT" ARRAY – (C)



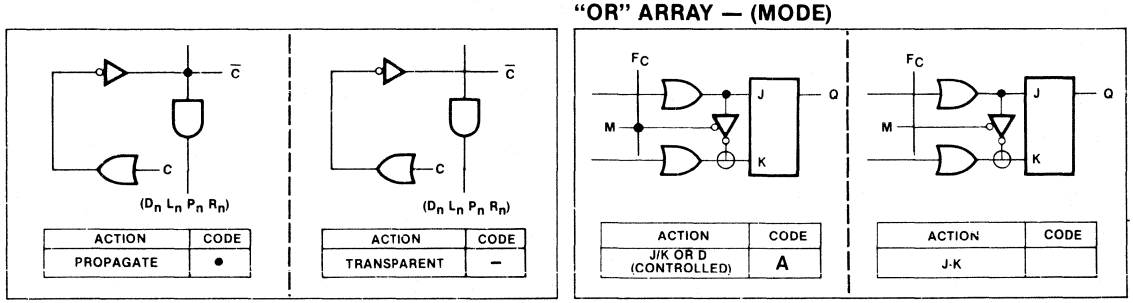
ACTION	CODE
INACTIVE[1,3,5]	O

ACTION	CODE
GENERATE[5]	A

ACTION	CODE
PROPAGATE	•

ACTION	CODE
TRANSPARENT	–

"OR" ARRAY — (MODE)



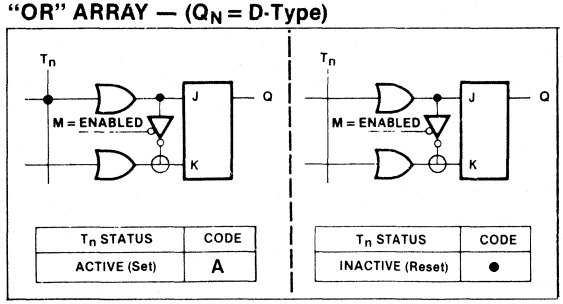
ACTION	CODE
PROPAGATE	•

ACTION	CODE
TRANSPARENT	–

ACTION	CODE
J/K OR D (CONTROLLED)	A

ACTION	CODE
J-K	

"OR" ARRAY — (Q_N = D-Type)

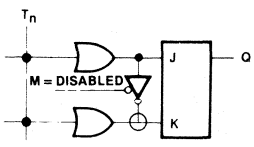


T_n STATUS	CODE
ACTIVE (Set)	A

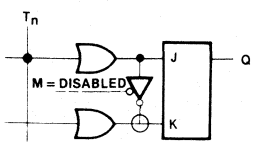
T_n STATUS	CODE
INACTIVE (Reset)	•

Notes on following page.

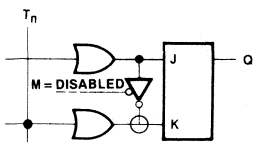
"OR" ARRAY — (Q_N = J-K Type)



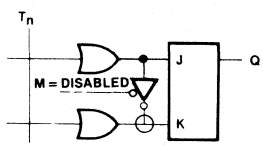
ACTION	CODE
TOGGLE	O



ACTION	CODE
SET	H

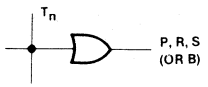


ACTION	CODE
RESET	L

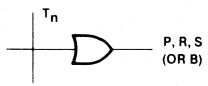


ACTION	CODE
HOLD	–

"OR" ARRAY — (S or B), (P), (R)

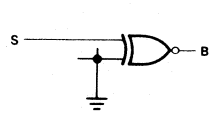


T_n STATUS	CODE
ACTIVE	A

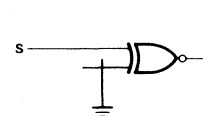


T_n STATUS	CODE
INACTIVE	•

"EX-OR" ARRAY — (B)

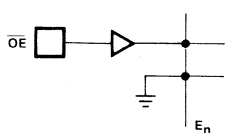


POLARITY	CODE
LOW	L

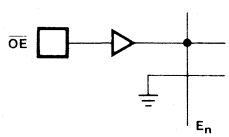


POLARITY	CODE
HIGH	H

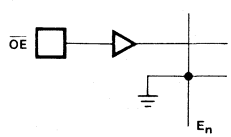
"$\overline{O.E.}$" ARRAY — (E)



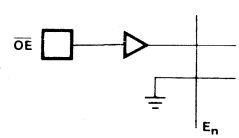
ACTION	CODE
IDLE[4]	O



ACTION	CODE
CONTROL	A



ACTION	CODE
ENABLE[4]	•



ACTION	CODE
DISABLE	–

NOTES
1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates.
2. Any gate $(T, F_C, L, P, R, D)_n$ will be unconditionally inhibited if any one of the I, B, or Q link pairs is left intact.
3. To prevent oscillations, this state is not allowed for C link pairs coupled to active gates T_n, F_C.
4. E_n = O and E_n = • are logically equivalent states, since both cause F_n outputs to be unconditionally enabled.
5. These states are not allowed for control gates $(L, P, R, D)_n$ due to their lack of "OR" array links.

Preview

INTEGRATED FUSE LOGIC
SERIES 20

FPLS PROGRAM TABLE

82S154/155

AND		OR		CONTROL			
INACTIVE O	I, B (I), Q (P)	ACTIVE A	P, R, B (O); (Q = D)	J/K •	F/F MODE	IDLE O	$E_{A, B}$
I, B, Q H		INACTIVE •		J/K or D (controlled) A		CONTROL A	
I, B, Q L						ENABLE •	
DON'T CARE –						DISABLE –	
INACTIVE O	C	TOGGLE O	(Q = J/K)	HIGH H	(POL.)		
GENERATE A		SET H		LOW L			
PROPAGATE •		RESET L					
TRANSPARENT –		HOLD –					

NOTES
1. The FPLS is shipped with all links intact. Thus a background of entries corresponding to states of virgin links exists in the table, shown BLANK for clarity.
2. Program unused C, I, B, and Q bits in the AND array as (—). Program unused Q, B, P, and R bits in the OR array as (—) or (A), as applicable.
3. Unused Terms can be left blank.
4. Q (P) and Q (N) are respectively the present and next states of flip-flops Q.

THIS PORTION TO BE COMPLETED BY SIGNETICS

CF (XXXX) _____

CUSTOMER SYMBOLIZED PART # _____

DATE RECEIVED _____

COMMENTS _____

CUSTOMER NAME _____

PURCHASE ORDER # _____

SIGNETICS DEVICE # _____

TOTAL NUMBER OF PARTS _____

PROGRAM TABLE # _____ REV ____ DATE ____

TERM	C	AND I				AND B(I)								AND Q(P)				F/F MODE		E_B	E_A	OR Q(N)				OR P		OR R		OR B(O)								POLARITY							
		3	2	1	0	7	6	5	4	3	2	1	0	3	2	1	0					3	2	1	0	B	A	B	A	7	6	5	4	3	2	1	0								
0																																													
1																																													
2																																													
3																																													
4																																													
5																																													
6																																													
7																																													
8																																													
9																																													
10																																													
11																																													
12																																													
13																																													
14																																													
15																																													
16																																													
17																																													
18																																													
19																																													
20																																													
21																																													
22																																													
23																																													
24																																													
25																																													
26																																													
27																																													
28																																													
29																																													
30																																													
31																																													
F_C																																													
L_B																																													
L_A																																													
D7																																													
D6																																													
D5																																													
D4																																													
D3																																													
D2																																													
D1																																													
D0																																													
PIN		5	4	3	2	19	18	13	12	9	8	7	6	17	16	15	14																														

Preview

INTEGRATED FUSE LOGIC
SERIES 20

82S156/157

FPLS PROGRAM TABLE

AND		OR		CONTROL			
INACTIVE O I, B, Q H I, B, Q L DON'T CARE —	I, B (I), Q (P)	ACTIVE A INACTIVE •	P, R, B (O), (Q = D)	J/K • J/K or D (controlled) A	F/F MODE	IDLE O CONTROL A ENABLE • DISABLE —	$E_{A, B}$
INACTIVE O GENERATE A PROPAGATE • TRANSPARENT —	C	TOGGLE O SET H RESET L HOLD —	(Q = J/K)	HIGH H LOW L	(POL.)		

NOTES
1. The FPLS is shipped with all links intact. Thus a background of entries corresponding to states of virgin links exists in the table, shown BLANK for clarity.
2. Program unused C, I, B, and Q bits in the AND array as (—). Program unused Q, B, P, and R bits in the OR array as (—) or (A), as applicable.
3. Unused Terms can be left blank.
4. Q (P) and Q (N) are respectively the present and next states of flip-flops Q.

F/F MODE | E_B | E_A | POLARITY

THIS PORTION TO BE COMPLETED BY SIGNETICS

CF (XXXX) _____

CUSTOMER SYMBOLIZED PART # _____

DATE RECEIVED _____

COMMENTS _____

CUSTOMER NAME _____

PURCHASE ORDER # _____

SIGNETICS DEVICE # _____

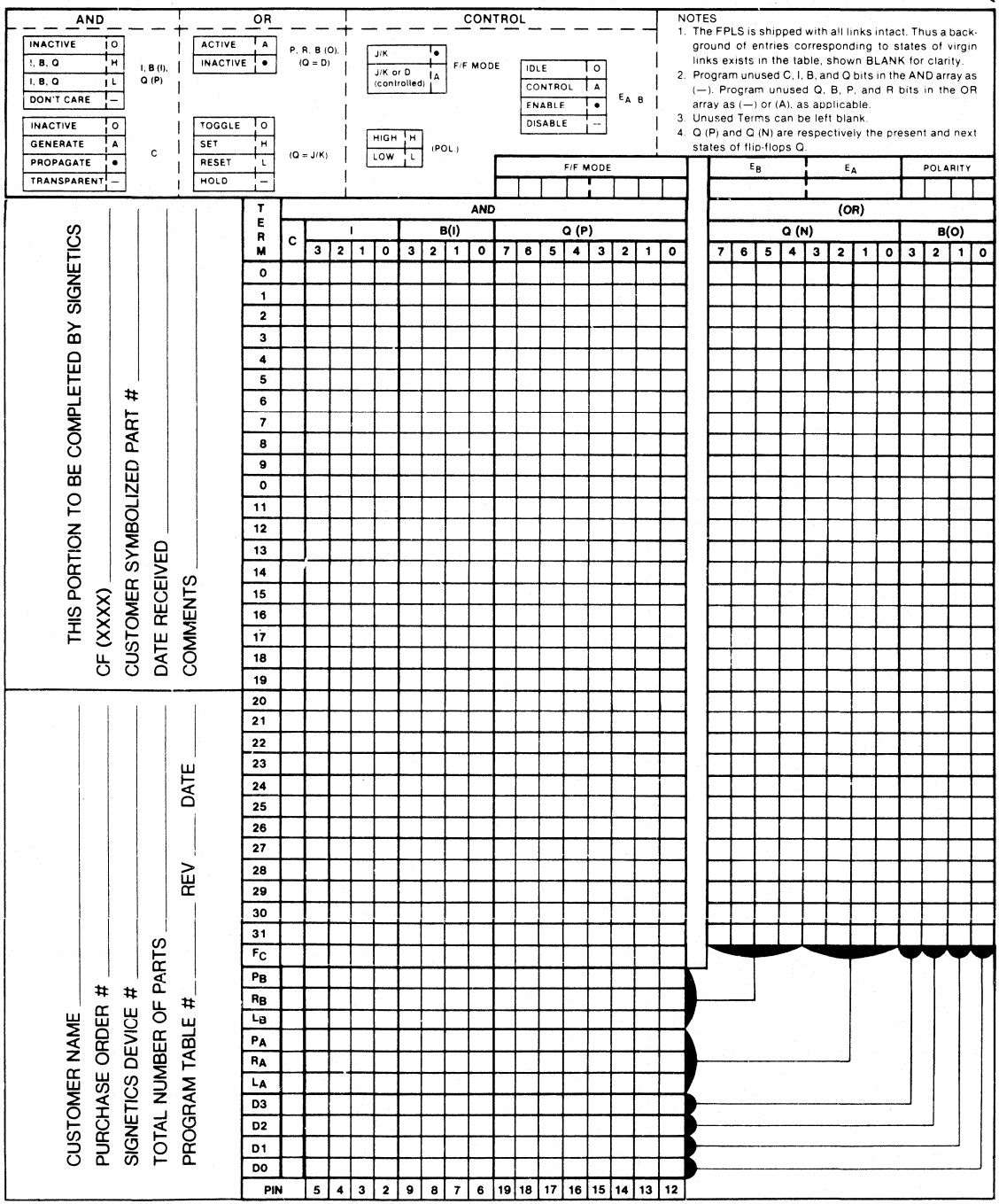
TOTAL NUMBER OF PARTS _____

PROGRAM TABLE # _____ REV ____ DATE ____

TERM	AND: C	I: 3 2 1 0	B(I): 5 4 3 2 1 0	Q (P): 5 4 3 2 1 0	OR: Q (N): 5 4 3 2 1 0	P: A	A	B(O): 5 4 3 2 1 0
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
F_C								
P_B								
R_B								
L_B								
L_A								
D5								
D4								
D3								
D2								
D1								
D0								
PIN		5 4 3 2	19 12 9 8 7 6	18 17 16 15 14 13				

FPLS PROGRAM TABLE

82S158/159



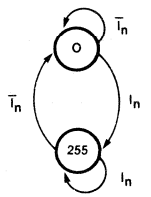
TEST ARRAY (82S158/159)

The FPLS may be subjected to AC and DC parametric tests prior to programming via an on chip test array.

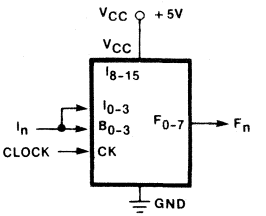
The array consists of test transition terms 32 and 50, factory programmed as shown below for the 82S15819.

Testing is accomplished by clocking the FPLS and applying the proper input sequence to I_{0-3} and B_{0-3} as shown in the test circuit timing diagram.

STATE DIAGRAM



FPLS UNDER TEST

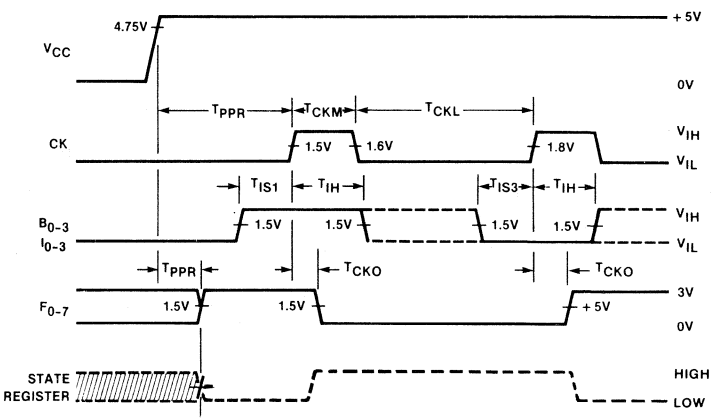


TEST ARRAY PROGRAM

TERM	C	I 3	I 2	I 1	I 0	B (I) 3	B (I) 2	B (I) 1	B (I) 0	Q (P) 7	Q (P) 6	Q (P) 5	Q (P) 4	Q (P) 3	Q (P) 2	Q (P) 1	Q (P) 0	Q (N) 7	Q (N) 6	Q (N) 5	Q (N) 4	Q (N) 3	Q (N) 2	Q (N) 1	Q (N) 0	B (O) 3	B (O) 2	B (O) 1	B (O) 0
		MODE A	A	A	A	A	A	A	A	E_B O								E_A O								(POLARITY) L	L	L	L
		AND																(OR)											
31																													
32	•	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	A	A	A	A
50	A	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	A	A	A	A

Both terms 32 and 50 must be deleted during user programming to avoid interfering with the desired logic function. This is accomplished automatically by any Signetics' qualified programming equipment.

TEST CIRCUIT TIMING DIAGRAM



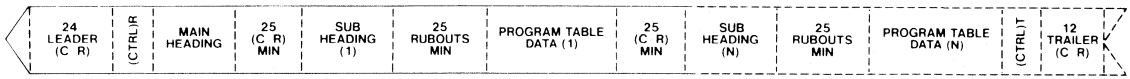
TEST ARRAY DELETED

TERM	C	I 3	I 2	I 1	I 0	B (I) 3	B (I) 2	B (I) 1	B (I) 0	Q (P) 7	Q (P) 6	Q (P) 5	Q (P) 4	Q (P) 3	Q (P) 2	Q (P) 1	Q (P) 0	Q (N) 7	Q (N) 6	Q (N) 5	Q (N) 4	Q (N) 3	Q (N) 2	Q (N) 1	Q (N) 0	B (O) 3	B (O) 2	B (O) 1	B (O) 0
		MODE A	A	A	A	A	A	A	A	E_B O								E_A O								(POLARITY) L	L	L	L
		AND																(OR)											
31																													
32	•	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	—	—	—	—	—	—	—	—	•	•	•	•
50	A	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	—	—	—	—	—	—	—	—	•	•	•	•

TWX TAPE CODING (LOGIC FORMAT)

The FPLA Program Table can be sent to Signetics in ASCII code format via airmail using any type of 8-level tape (paper, mylar, fanfold, etc.), or via TWX: just dial (910) 339-9283, tell the operator to turn the paper puncher on, and acknowledge. At the end of transmission instruct the operator to send tape to Signetics Order Entry.

A number of Program Tables can be sequentially assembled on a continuous tape as follows, however, limit tape length to a roll of 1.75 inch inside diameter and 4.25 inch outside diameter.



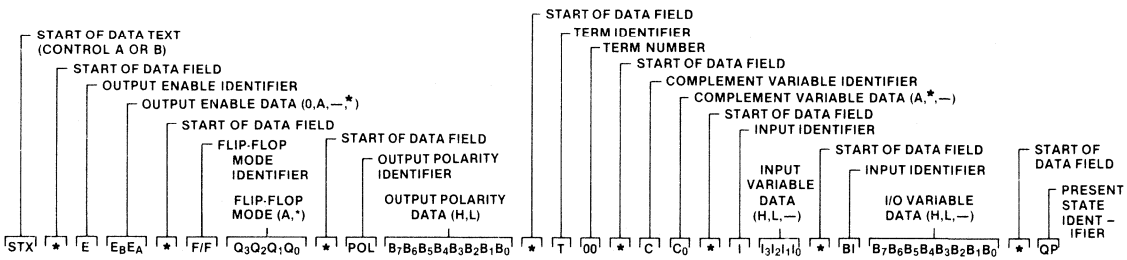
A. The MAIN HEADING at the beginning of tape includes the following information, with each entry preceded by a ($) character, whether used or not:

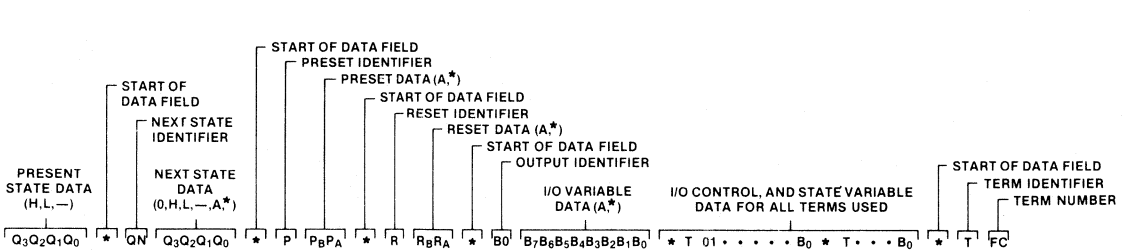
1. Customer Name _____
2. Customer TWX No. _____
3. Date _____
4. Purchase Order No. _____
5. Number of Program Tables _____
6. Total Number of Parts _____

B. Each SUB HEADING should contain specific information pertinent to each Program Table as follows, with each entry preceded by a ($) character, whether used or not:

1. Signetics Device No. _____
2. Program Table No. _____
3. Revision _____
4. Date _____
5. Customer Symbolized Part No. _____
6. Number of Parts _____

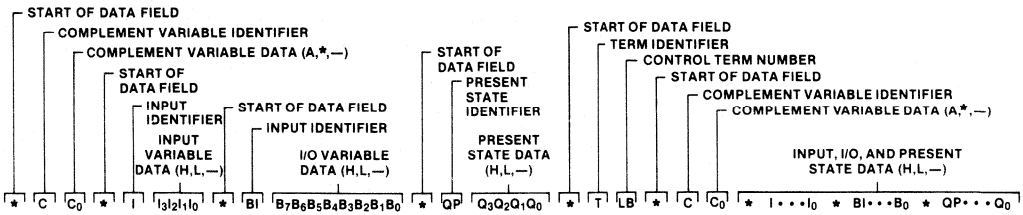
C. Program Table data blocks are initiated with an STX character, and terminated with an ETX character. The body of the data consists of output polarity, product term, and output information separated by appropriate identifiers in accordance with the following format. Entries for the data fields correspond to those defined in the Logic PROGRAM TABLE:

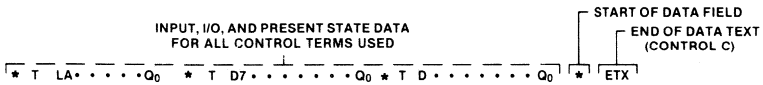




(Continued on next page)

TWX TAPE CODING (Continued)

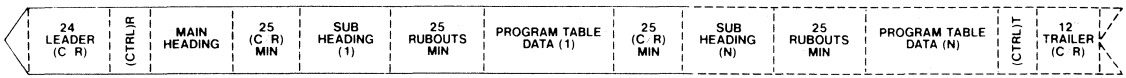




TWX TAPE CODING (LOGIC FORMAT)

The FPLA Program Table can be sent to Signetics in ASCII code format via airmail using any type of 8-level tape (paper, mylar, fanfold, etc.), or via TWX: just dial (910) 339-9283, tell the operator to turn the paper puncher on, and acknowledge. At the end of transmission instruct the operator to send tape to Signetics Order Entry.

A number of Program Tables can be sequentially assembled on a continuous tape as follows, however, limit tape length to a roll of 1.75 inch inside diameter and 4.25 inch outside diameter.



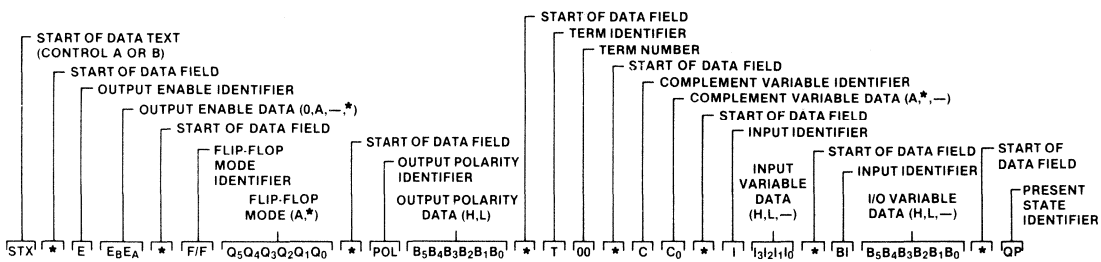
A. The MAIN HEADING at the beginning of tape includes the following information, with each entry preceded by a ($) character, whether used or not:

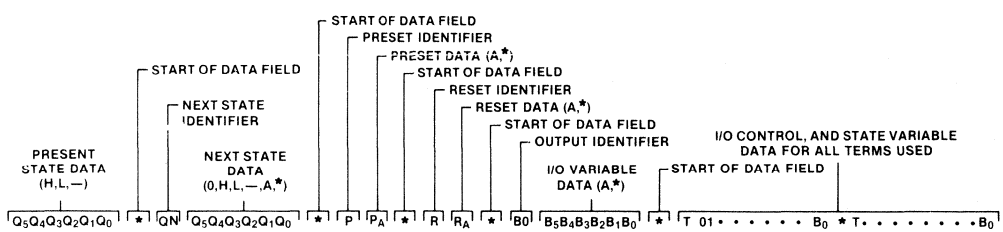
1. Customer Name _____
2. Customer TWX No. _____
3. Date _____
4. Purchase Order No. _____
5. Number of Program Tables _____
6. Total Number of Parts _____

B. Each SUB HEADING should contain specific information pertinent to each Program Table as follows, with each entry preceded by a ($) character, whether used or not:

1. Signetics Device No. _____
2. Program Table No. _____
3. Revision _____
4. Date _____
5. Customer Symbolized Part No. _____
6. Number of Parts _____

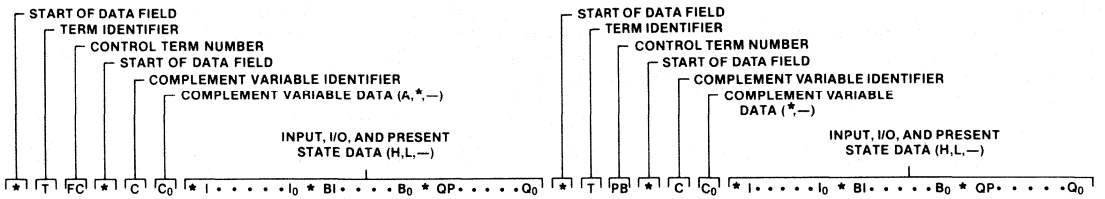
C. Program Table data blocks are initiated with an STX character, and terminated with an ETX character. The body of the data consists of output polarity, product term, and output information separated by appropriate identifiers in accordance with the following format. Entries for the data fields correspond to those defined in the Logic PROGRAM TABLE:

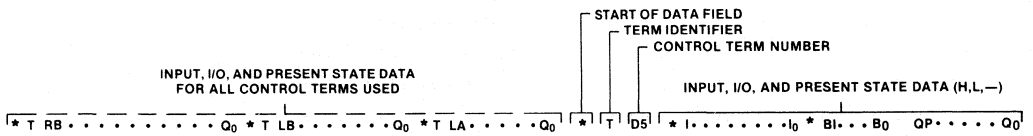


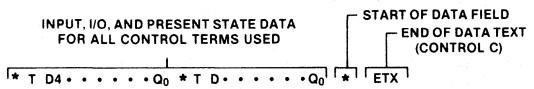


(Continued on next page)

TWX TAPE CODING (Continued)



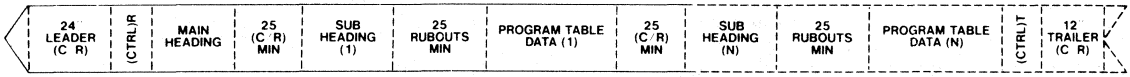




TWX TAPE CODING (LOGIC FORMAT)

The FPLA Program Table can be sent to Signetics in ASCII code format via airmail using any type of 8-level tape (paper, mylar, fanfold, etc.), or via TWX: just dial (910) 339-9283, tell the operator to turn the paper puncher on, and acknowledge. At the end of transmission instruct the operator to send tape to Signetics Order Entry.

A number of Program Tables can be sequentially assembled on a continuous tape as follows, however,limit tape length to a roll of 1.75 inch inside diameter and 4.25 inch outside diameter.



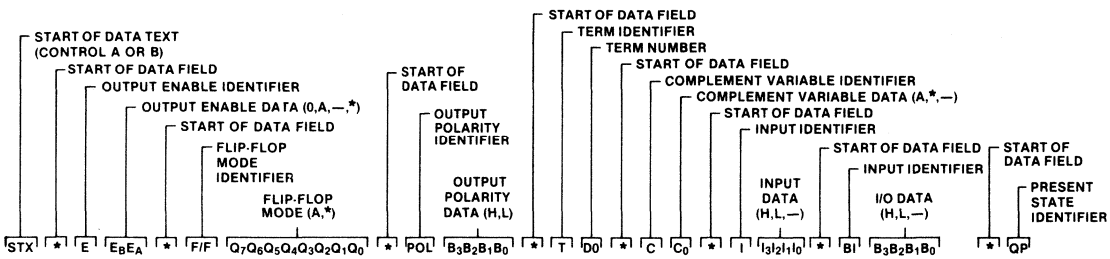
A. The MAIN HEADING at the beginning of tape includes the following information, with each entry preceded by a ($) character, whether used or not:

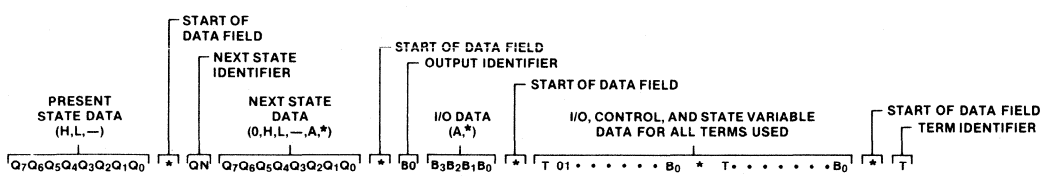
1. Customer Name _____
2. Customer TWX No. _____
3. Date _____
4. Purchase Order No. _____
5. Number of Program Tables _____
6. Total Number of Parts _____

B. Each SUB HEADING should contain specific information pertinent to each Program Table as follows, with each entry preceded by a ($) character, whether used or not:

1. Signetics Device No. _____
2. Program Table No. _____
3. Revision _____
4. Date _____
5. Customer Symbolized Part No. _____
6. Number of Parts _____

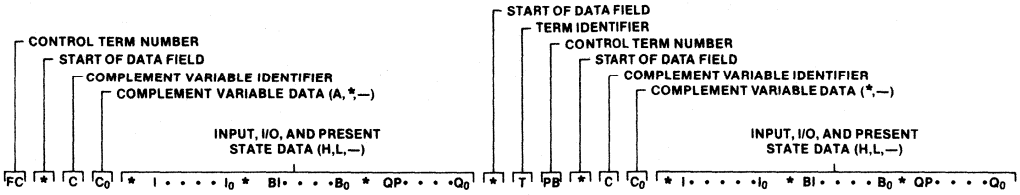
C. Program Table data blocks are initiated with an STX character, and terminated with an ETX character. The body of the data consists of output polarity, product term, and output information separated by appropriate identifiers in accordance with the following format. Entries for the data fields correspond to those defined in the Logic PROGRAM TABLE:

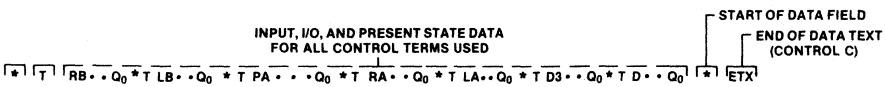




(Continued on next page)

TWX TAPE CODING (Continued)





IFL SERIES 28

FIELD PROGRAMMABLE LOGIC ARRAY (16×48×8) 82S100 (T.S.)/82S101 (O.C.)

INTEGRATED FUSE LOGIC
SERIES 28

DESCRIPTION

The 82S100 (tri-state) and 82S101 (open collector) are Bipolar, Fuse-Link Programmable Logic Arrays (FPLA). Each device utilizes the standard AND/OR/Invert architecture to directly implement custom sum of product logic equations.

Each device consists of 16 dedicated inputs and 8 dedicated outputs. Each output is capable of being actively controlled by any or all of the 48 product terms. The true, complement, or don't care condition of each of the 16 inputs ANDed together comprise one P-term. All 48 P-terms are then ORed to each output. The user must then only select which P-terms will activate an output by disconnecting terms which do not affect the output. In addition each output can be fused as active-high$^{(H)}$ or active-low$^{(L)}$.

The 82S100 and 82S101 are fully TTL compatible, and include chip-enable control for expansion of input variables, and output inhibit. They feature either open collector or tri-state outputs for ease of expansion of product terms and application in bus-organized systems.

Both devices are available in commercial and military temperature ranges. For the commercial temperature range (0°C to +75°C) specify N82S100/101, F or N, and for the military temperature range (−55°C to +125°C) specify S82S100/101, F or G, I, R.

FEATURES

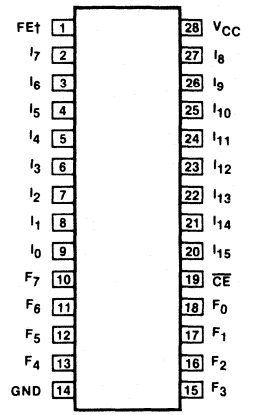
- **Field programmable (Ni-Cr link)**
- **Input variables: 16**
- **Output functions: 8**
- **Product terms: 48**
- **Address access time:**
  S82S100/101 — 80ns max
  N82S100/101 — 50ns max
- **Power dissipation: 600mW typ**
- **Input loading:**
  S82S100/101: −150μA max
  N82S100/101: −100μA max
- **Chip enable input**
- **Output option:**
  82S100: Tri-state
  82S101: Open collector
- **Output disable function:**
  Tri-state — Hi-Z
  Open collector — Hi
- **Separate I/O architecture**

APPLICATIONS

- **CRT display systems**
- **Random logic**
- **Code conversion**
- **Peripheral controllers**
- **Function generators**
- **Look-up and decision tables**
- **Microprogramming**
- **Address mapping**
- **Character generators**
- **Data security encoders**
- **Fault detectors**
- **Frequency synthesizers**
- **16 bit-to-8 bit bus interface**
- **Random logic replacement**

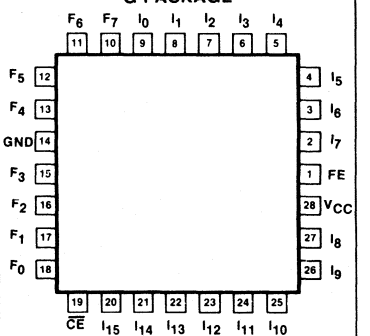
PIN CONFIGURATION

F,I,N,R PACKAGE*



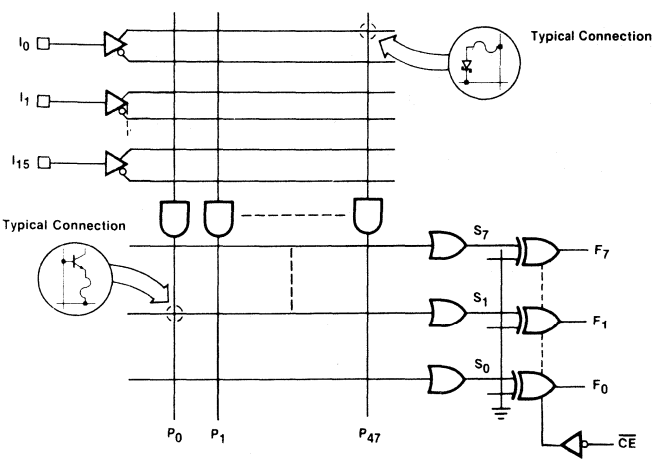
*F = Cerdip I = Ceramic N = Plastic
†Open or grounded during normal operation.

G PACKAGE



G = Leadless TOP VIEW

LOGIC DIAGRAM



Fusible Ni-Cr links are initially intact at all array cross-points.

LOGIC FUNCTION

Typical Output Term:

$P_0 = I_0 \cdot I_1 \cdot \overline{I_2} \cdot I_5 \cdot \overline{I_{13}}$

Typical Output Functions: @ $\overline{CE} = 0$:

$F_0 = (P_0 + P_1 + P_2)$ @ active high

$F_0 = (\overline{P_0} \cdot \overline{P_1} \cdot \overline{P_2})$ @ active low

NOTE
For each of the 8 outputs, either the function F_p (active high) or $\overline{F_p}$ (active low) is available but not both.

INTEGRATED FUSE LOGIC
SERIES 28

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER		RATING Min	RATING Max	UNIT
V_{CC}	Supply voltage		+7	Vdc
V_{IN}	Input voltage		+5.5	Vdc
V_{OUT}	Output voltage		+5.5	Vdc
I_{IN}	Input currents	−30	+30	mA
I_{OUT}	Output currents		+100	mA
	Temperature range			°C
T_A	Operating			
	N82S100/101	0	+75	
	S82S100/101	−55	+125	
T_{STG}	Storage	−65	+150	

THERMAL RATINGS

TEMPERATURE	MILITARY	COMMERCIAL
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

DC ELECTRICAL CHARACTERISTICS

N82S100/101: $0°C \le T_A \le +75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S100/101: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TEST CONDITIONS	N82S100/101 Min	N82S100/101 Typ[2]	N82S100/101 Max	S82S100/101 Min	S82S100/101 Typ[2]	S82S100/101 Max	UNIT
	Input voltage[3]								V
V_{IH}	High	V_{CC} = Max	2			2			
V_{IL}	Low	V_{CC} = Min			0.85			0.8	
V_{IC}	Clamp[3,4]	V_{CC} = Min, I_{IN} = −18mA		−0.8	−1.2		−0.8	−1.2	
	Output voltage	V_{CC} = Min							V
V_{OH}	High (82S100)[3,5]	I_{OH} = −2mA	2.4			2.4			
V_{OL}	Low[3,6]	I_{OL} = 9.6mA		0.35	0.45		0.35	0.50	
	Input current								μA
I_{IH}	High	V_{IN} = 5.5V		<1	25		<1	50	
I_{IL}	Low	V_{IN} = 0.45V		−10	−100		−10	−150	
	Output current	$\overline{CE}$ = High, V_{CC} = Max							
I_{OLK}	Leakage[7]	V_{OUT} = 5.5V		1	40		1	60	μA
$I_{O(OFF)}$	Hi-Z state (82S100)[7]	V_{OUT} = 5.5V		1	40		1	60	μA
		V_{OUT} = 0.45V		−1	−40		−1	−60	
I_{OS}	Short circuit (82S100)[4,8]	$\overline{CE}$ = Low, V_{OUT} = 0V	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current[9]	V_{CC} = Max		120	170		120	180	mA
	Capacitance[7]	$\overline{CE}$ = High, V_{CC} = 5.0V							pF
C_{IN}	Input	V_{IN} = 2.0V		8			8		
C_{OUT}	Output	V_{OUT} = 2.0V		17			17		

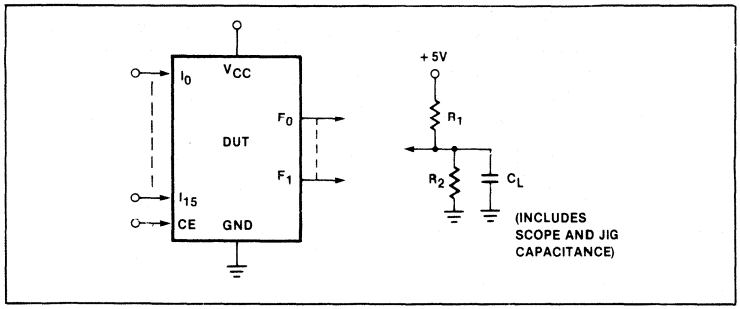
AC ELECTRICAL CHARACTERISTICS

$R_1 = 470\Omega$, $R_2 = 1k\Omega$, $C_L = 30pF$
N82S100/101: $0°C \le T_A \le +75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S100/101: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

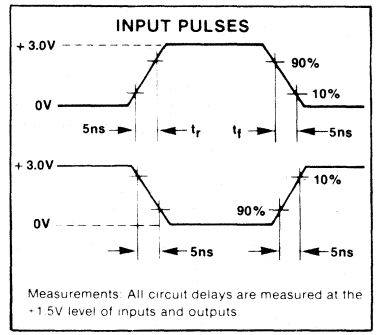
PARAMETER		TO	FROM	N82S100/101 Min	N82S100/101 Typ[2]	N82S100/101 Max	S82S100/101 Min	S82S100/101 Typ[2]	S82S100/101 Max	UNIT
	Propagation delay									ns
T_{IA}	Input	Output	Input		35	50		35	80	
T_{CE}	Chip enable	Output	Chip enable		15	30		15	50	
	Disable time									ns
T_{CD}	Chip disable	Output	Chip enable		15	30		15	50	

NOTES:
1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and only functional operation of the device at these or any other conditions above those indicated in the operation of the device specifications is not implied.
2. All voltages are at V_{CC} = 5V, T_A = 25°C.
3. All voltage values are with respect to network ground terminal.
4. Test one at a time.
5. Measured with V_{IL} applied to $\overline{CE}$ and a logic high stored.
6. Measured with a programmed logic condition for which the output test is at a low logic level. Output sink current is applied through a resistor to V_{CC}.
7. Measured with V_{IH} applied to $\overline{CE}$.
8. Duration of short circuit should not exceed 1 second.
9. I_{CC} is measured with the chip enable input grounded, all other inputs at 4.5V and the outputs open.

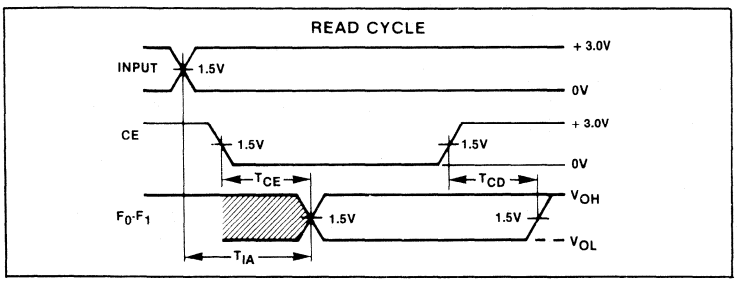
TEST LOAD CIRCUIT



VOLTAGE WAVEFORM



TIMING DIAGRAM



TIMING DEFINITIONS

T_{CE} Delay between beginning of Chip Enable low (with Input valid) and when Data Output becomes valid.

T_{CD} Delay between when Chip Enable becomes high and Data Output is in off state (Hi-Z or high).

T_{IA} Delay between beginning of valid input (with Chip Enable low) and when Data Output becomes valid.

VIRGIN DEVICE

The 82S100/101 are shipped in an unprogrammed state, characterized by:

1. All internal Ni-Cr links are intact.
2. Each product term (P-term) contains both true and complement values of every input variable I_m (P-terms always logically "false").
3. The "OR" Matrix contains all 48-P-terms.
4. The polarity of each output is set to active high (F_P function).
5. All outputs are at a low logic level.

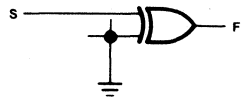
LOGIC PROGRAMMING

The FPLA can be programmed by means of Logic programming equipment.

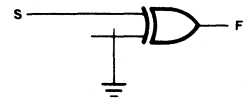
With Logic programming, the AND/OR/EX-OR gate input connections necessary to implement the desired logic function are coded directly from logic equations using the Program Table.

In this Table the logic state of variables I, P, and F, associated with each Sum Term S_n, is assigned a symbol which results in the proper fusing pattern of corresponding link pairs, defined as follows:

OUTPUT POLARITY—(F)

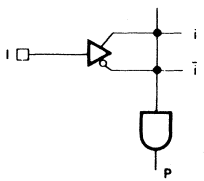


ACTIVE LEVEL	CODE
HIGH	H

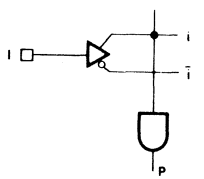


ACTIVE LEVEL	CODE
LOW	L

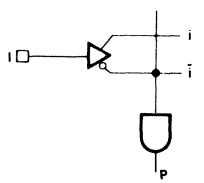
"AND" ARRAY - (I)



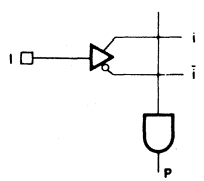
STATE	CODE
INACTIVE[1,2]	O



STATE	CODE
I	H

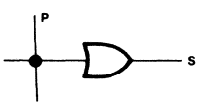


STATE	CODE
$\bar{I}$	L

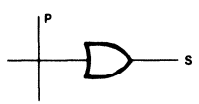


STATE	CODE
DON'T CARE	—

"OR" ARRAY — (F)



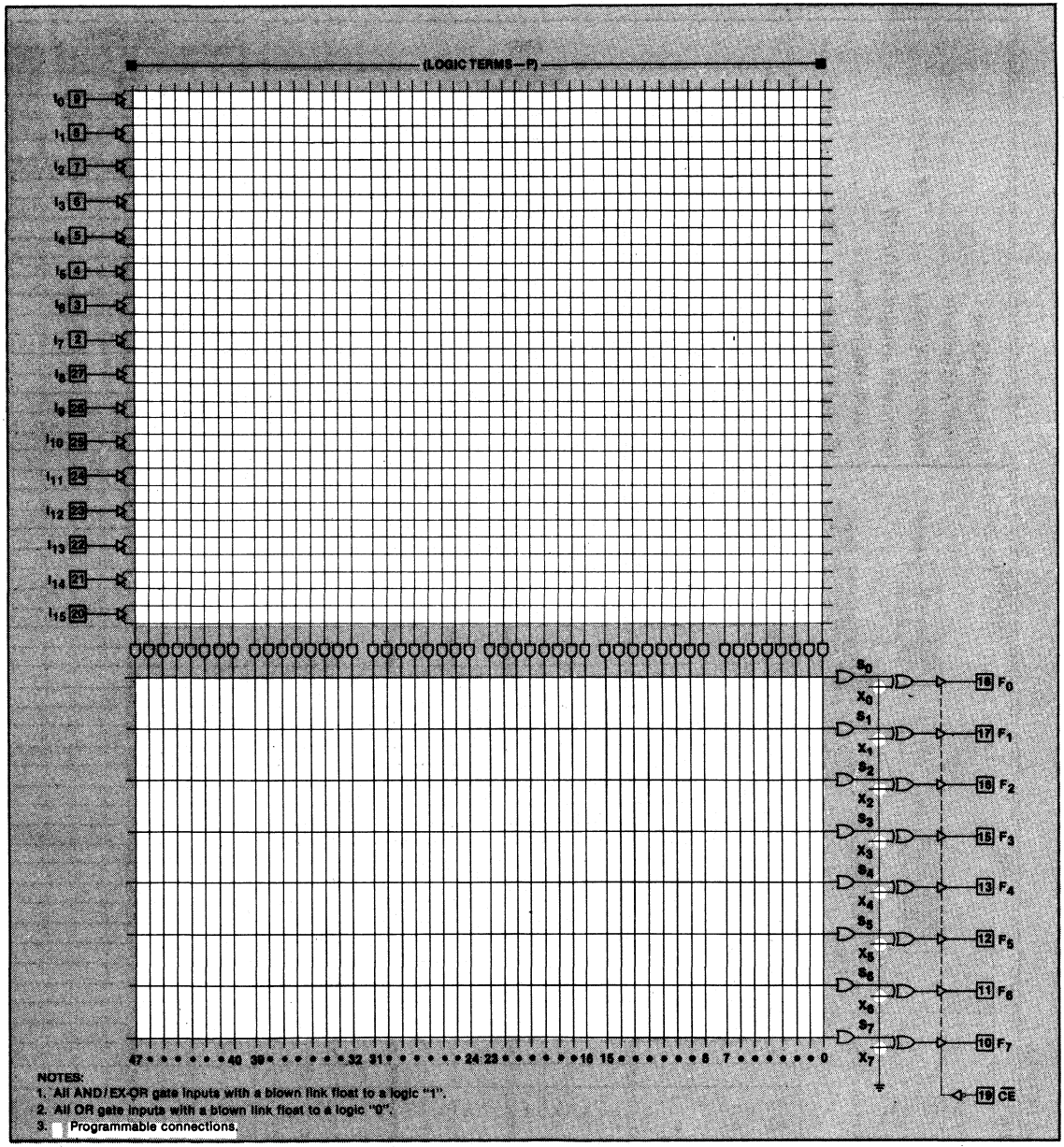
Pn STATUS	CODE
ACTIVE	A



Pn STATUS	CODE
INACTIVE	•

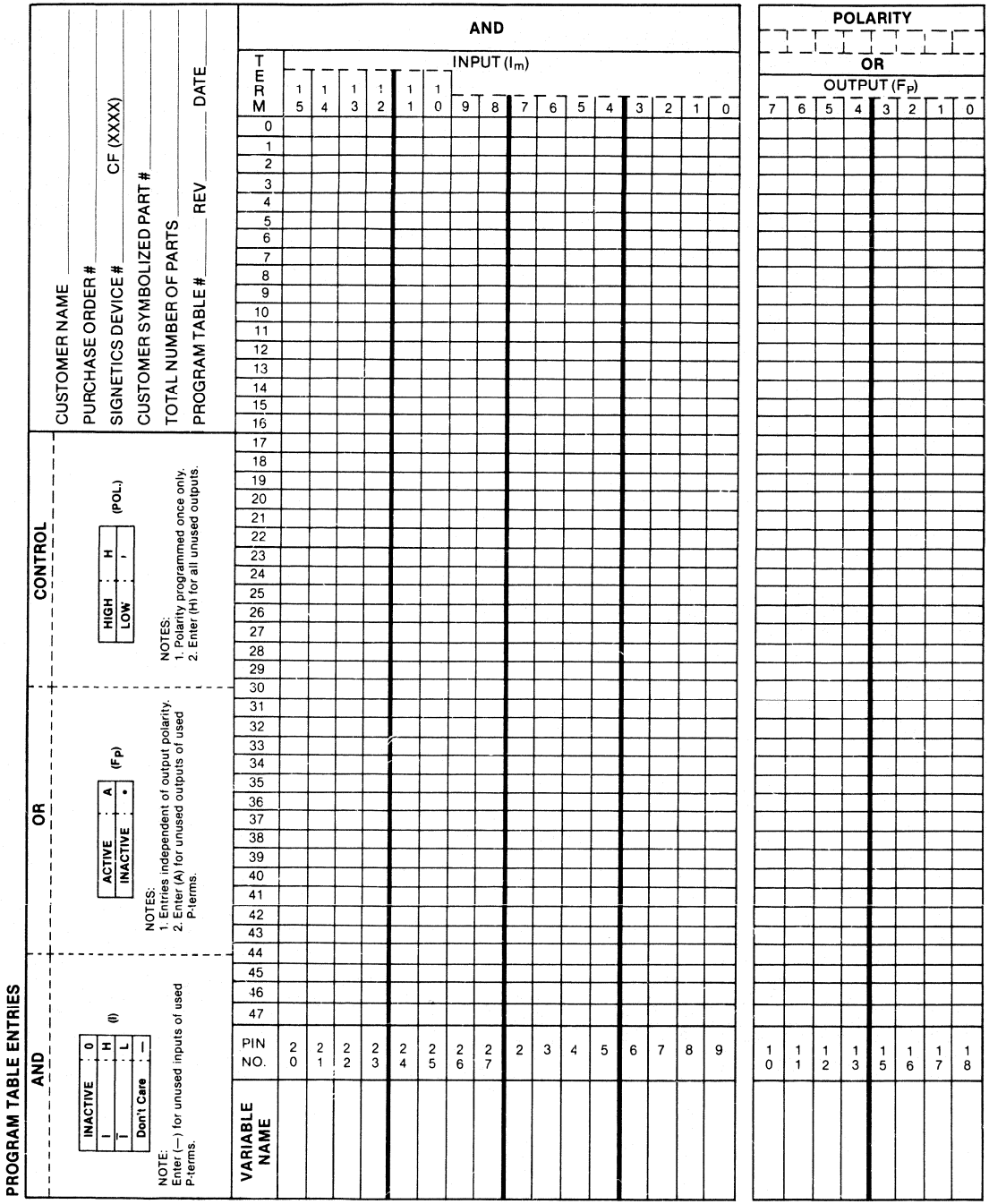
NOTES
1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates Pn.
2. Any gate Pn will be unconditionally inhibited if any one of its (I) link pairs is left intact.

FPLA LOGIC DIAGRAM



NOTES:
1. All AND/EX-OR gate inputs with a blown link float to a logic "1".
2. All OR gate inputs with a blown link float to a logic "0".
3. ■ Programmable connections.

FPLA PROGRAM TABLE



INTEGRATED FUSE LOGIC
SERIES 28

DESCRIPTION

The 82S102 and 82S103 are Bipolar, fuse programmable, gate arrays. The device consists of 9 AND/NAND Gates which share 16 common inputs. The type of gate is selected by programming the output as active-high(H) or active-low(L). Each of the 16 inputs I_0-I_{15} can be programmed to provide the True (H), Complement (L), or Don't Care (—) state to each of the 9 AND/NAND gates. OR/NOR logic functions can also be implemented by complementing the inputs and outputs via on-chip inverting buffers.

Both devices are field-programmable, which means that custom patterns are immediately available.

The 82S102 and 82S103 include chip-enable control for output strobing and inhibit. They feature either open collector or tri-state outputs for ease of expansion of input variables and application in bus-organized systems.

Both devices are available in the commercial and military temperature ranges. For the commercial range (0°C to +75°C) specify N82S102/103, F or N, and for the military range (−55°C to +125°C) specify S82S102/103, F, G, I, and R.

FEATURES

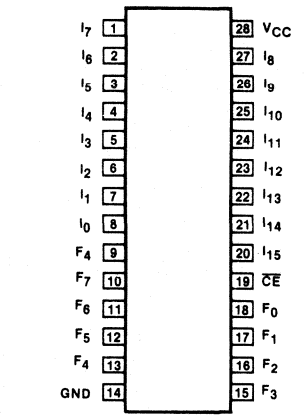
- **Field programmable (Ni-Cr link)**
- **16 input variables**
- **9 output functions**
- **Chip enable input**
- **I/O propagation delay:**
  N82S102/103: 35ns max
  S82S102/103: 50ns max
- **Power dissipation: 600mW typ**
- **Input loading:**
  N82S102/103: -100μA max
  S82S102/103: -150μA max
- **Output options:**
  82S102: Open collector
  82S103: Tri-state
- **Output disable function:**
  82S102: Hi
  82S103: Hi-Z
- **Fully TTL compatible**

APPLICATIONS

- **Random logic**
- **Address decoders**
- **Code detectors**
- **Peripheral selectors**
- **Fault monitors**
- **Machine state decoders**

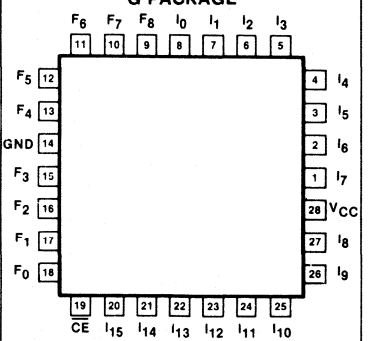
PIN CONFIGURATION

F,I,N,R PACKAGE*



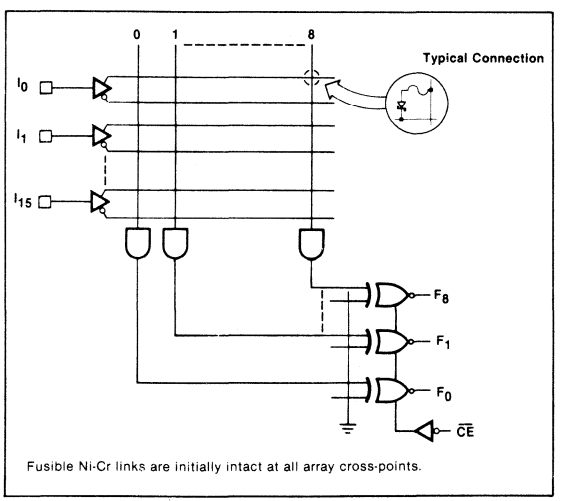
*F = Cerdip N = Plastic
I = Ceramic R = Flat Pack

G PACKAGE



G = Leadless TOP VIEW

LOGIC DIAGRAM



Fusible Ni-Cr links are initially intact at all array cross-points.

LOGIC FUNCTION

Typical Output Functions @ $\overline{CE} = 0$:

Active High (H)

$F_0 = (I_0 \cdot I_1 \cdot I_2 \cdot \ldots \cdot \overline{Im})$

Active Low (L)

$F_0 = (\overline{I_0} + \overline{I_1} + \overline{I_2} + \ldots Im)$

$m = 0, 1, 2, \ldots . 15$

NOTES

For each of the 9 outputs, either the function F_p (active high) or $\overline{F}_p$ (active low) is available but not both. The required output polarity is programmed in the EX-OR array.

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
	Output voltage		Vdc
V_{OH}	High (82S102)	+5.5	
V_O	Off-state (82S103)	+5.5	
I_{IN}	Input current	±30	mA
I_{OUT}	Output current	+100	mA
	Temperature range		°C
T_A	Operating		
	N82S102/103	0 to +75	
	S82S102/103	−55 to +125	
T_{STG}	Storage	−65 to +150	

THERMAL RATINGS

TEMPERATURE	MILITARY	COMMERCIAL
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

DC ELECTRICAL CHARACTERISTICS

N82S102/103: $0°C \le T_A \le +75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S102/103: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER[1]		TEST CONDITIONS	N82S102/103			S82S102/103			UNIT
			Min	Typ[2]	Max	Min	Typ[2]	Max	
	Input voltage								V
V_{IL}	Low[1]	V_{CC} = Min			0.85			0.8	
V_{IH}	High[1]	V_{CC} = Max	2.0			2.0			
V_{IC}	Clamp[1,3]	V_{CC} = Min, I_{IN} = −18mA		−0.8	−1.2		−0.8	−1.2	
	Output voltage	V_{CC} = Min							V
V_{OL}	Low[1,4]	I_{OL} = 9.6mA		0.35	0.45		0.35	0.50	
V_{OH}	High (82S103)[1,5]	I_{OH} = −2mA	2.4			2.4			
	Input current								μA
I_{IL}	Low	V_{IN} = 0.45V		−10	−100		−10	−150	
I_{IH}	High	V_{IN} = 5.5V		<1	25		<1	50	
	Output current	V_{CC} = Max							
I_{OLK}	Leakage (82S102)[6]	V_{OUT} = 5.5V		1	40		1	60	μA
$I_{O(OFF)}$	Hi-Z state (82S103)[6]	V_{OUT} = 5.5V		1	40		1	60	μA
		V_{OUT} = 0.45V		−1	−40		−1	−60	
I_{OS}	Short circuit (82S103)[3,7]	V_{OUT} = 0V	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current[8]	V_{CC} = Max		120	170		120	180	mA
	Capacitance	V_{CC} = 5.0V							pF
C_{IN}	Input	V_{IN} = 2.0V		8			8		
C_{OUT}	Output[6]	V_{OUT} = 2.0V		15			15		

AC ELECTRICAL CHARACTERISTICS

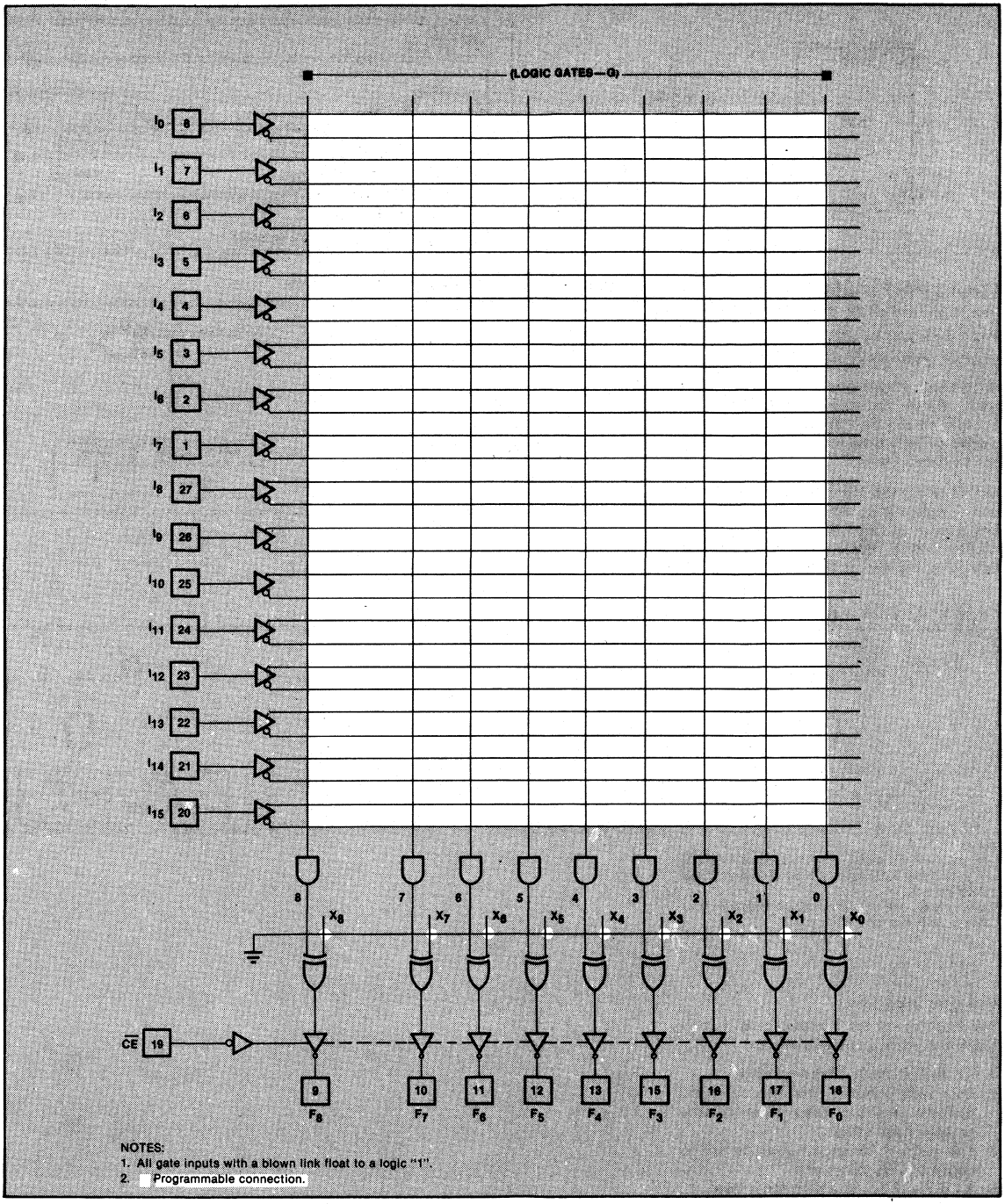
$R_1 = 470\Omega$, $R_2 = 1k\Omega$, $C_L = 30pF$
N82S102/103: $0°C \le T_A \le +75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S102/103: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TO	FROM	N82S102/103			S82S102/103			UNIT
				Min	Typ[2]	Max	Min	Typ[2]	Max	
	Progagation delay									ns
T_{IA}	Input	Output	Input		20	35		20	55	
T_{CE}	Chip enable	Output	Chip enable		15	30		15	45	
	Disable time									ns
T_{CD}	Chip disable	Output	Chip enable		15	30		15	45	

NOTES

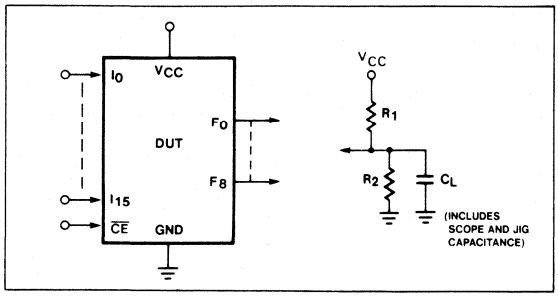
1. All voltage values are with respect to network ground terminal.
2. All typical values are at V_{CC} = 5V, T_A = 25°C.
3. Test each output one at a time.
4. Measured with a programmed logic condition for which the output under test is at a low logic level. Output sink current is supplied through a resistor to V_{CC}.
5. Measured with V_{IL} applied to $\overline{CE}$ and a logic high at the output.
6. Measured with V_{IH} applied to $\overline{CE}$.
7. Duration of short circuit should not exceed 1 second.
8. I_{CC} is measured with the chip enable input grounded, all other inputs at 4.5V and the outputs open.

FPGA LOGIC DIAGRAM

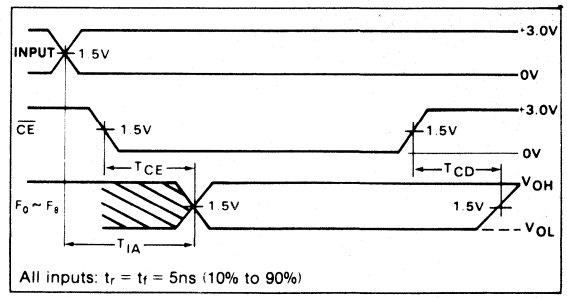


NOTES:
1. All gate inputs with a blown link float to a logic "1".
2. ■ Programmable connection.

TEST LOAD CIRCUIT



VOLTAGE WAVEFORM



All inputs: $t_r = t_f = 5ns$ (10% to 90%)

LOGIC PROGRAMMING

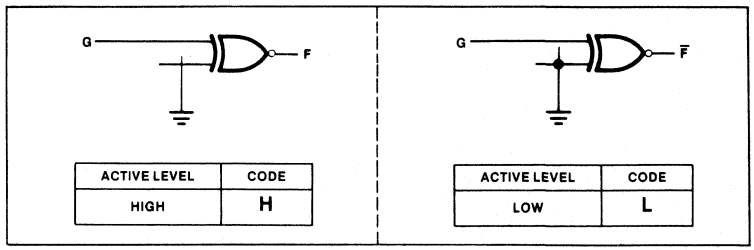
In a virgin device all Ni-Cr links are intact.

The FPGA can be programmed by means of Logic programming equipment.

With Logic programming, the AND/EX-OR gate input connections necessary to implement the desired logic function are coded directly from logic equations using the Program Table.

In this table, the logic state of variables I and F associated with each gate G_n is assigned a symbol which results in the proper fusing pattern of corresponding link pairs, defined as follows:

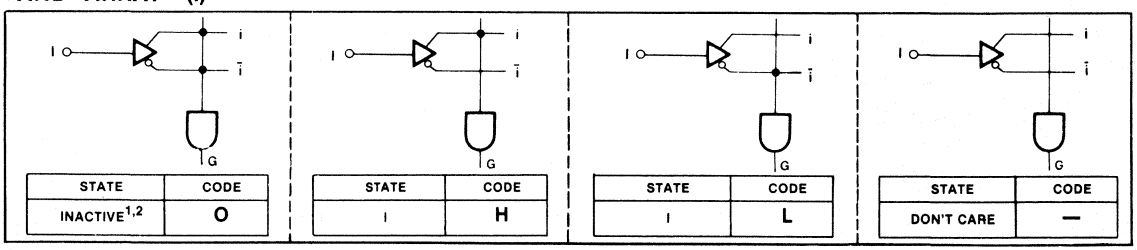
OUTPUT POLARITY—(F)



ACTIVE LEVEL	CODE
HIGH	H

ACTIVE LEVEL	CODE
LOW	L

"AND" ARRAY—(I)



STATE	CODE
INACTIVE[1,2]	O

STATE	CODE
I	H

STATE	CODE
I	L

STATE	CODE
DON'T CARE	—

NOTES
1. This is the initial unprogrammed state of al link pairs. It is normally associated with all unused (inactive) AND gates G_n.
2. Any gate G_n will be unconditionally inhibited if any one of its (I) link pairs is left intact.

VIRGIN DEVICE

The 82S102/103 are shipped in an unprogrammed state, characterized by:

1. All internal Ni-Cr links are intact.
2. Each gate contains both true and complement values of every input variable I_m (logic Null state).
3. The polarity of each output is set to active low ($\overline{F}_p$ function).
4. All outputs are at a high logic level.

INTEGRATED FUSE LOGIC
SERIES 28

FPGA PROGRAM TABLE

CUSTOMER NAME _____	*THIS PORTION TO BE COMPLETED BY SIGNETICS*
PURCHASE ORDER # _____	*CF (XXXX)* _____
SIGNETICS DEVICE # _____	*CUSTOMER SYMBOLIZED PART #* _____
TOTAL NUMBER OF PARTS _____	*DATE RECEIVED* _____
PROGRAM TABLE # _____	*COMMENTS* _____

F_0 (18) _____ =
F_1 (17) _____ =
F_2 (16) _____ =
F_3 (15) _____ =
F_4 (13) _____ =
F_5 (12) _____ =
F_6 (11) _____ =
F_7 (10) _____ =
F_8 (9) _____ =

GATE		INPUT															
POLARITY		I_{15}	I_{14}	I_{13}	I_{12}	I_{11}	I_{10}	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
F_0																	
F_1																	
F_2																	
F_3																	
F_4																	
F_5																	
F_6																	
F_7																	
F_8																	
PIN NO.		20	21	22	23	24	25	26	27	1	2	3	4	5	6	7	8
VARIABLE NAME																	

NOTES
1. The FPGA is shipped with all links intact. Thus a background of entries corresponding to states of virgin links exists in the table, shown BLANK for clarity.
2. Unused I bits are normally programmed Don't Care (—).
3. Unused Gates can be left blank.

PROGRAM TABLE ENTRIES

AND		CONTROL	
INACTIVE	0	HIGH	H
I	H	LOW	L
$\overline{I}$	L		
Don't Care	—		
(I)		(POL.)	

DESCRIPTION

The 82S104 (open collector outputs) and the 82S105 (tri-state outputs) are bipolar, programmable state machines of the Mealy type. They contain logic AND-OR gate arrays with user programmable connections which control the inputs of on-chip State and Output registers. These consist respectively of 6 Q_P, and 8 Q_F edge triggered, clocked S/R flip-flops, with an asynchronous Preset option. All flip-flops are unconditionally preset to "1" during power turn on.

The AND array combines 16 external inputs $I_{0\text{-}15}$ with 6 internal inputs $P_{0\text{-}5}$ fed back from the State register to form up to 48 Transition terms (AND terms). All Transition terms can include True, False, or Don't Care states of the controlling variables, and are merged in the OR array to issue next-state and next-output commands to their respective registers on the Low to High transition of the Clock pulse. Both True and Complement Transition terms can be generated by optional use of the internal input variable (C) from the Complement array. Also, if desired, the Preset input can be converted to $\overline{\text{Output-Enable}}$ function, as an additional user programmable option.

Both devices are available in commercial and military temperature ranges. For the commercial temperature range (0°C to +75°C) specify N82S104/105, F or N, and for the military temperature range (−55°C to +125°C) specify S82S104/105 F, G or R.

FEATURES

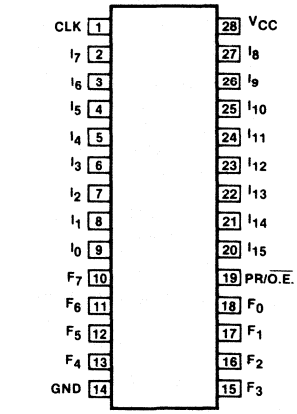
- Field programmable (Ni-Cr link)
- 16 input variables
- 8 output functions
- 48 transition terms
- 6-bit state register
- 8-bit output register
- Transition complement array
- Positive edge trigger clock
- Programmable asynchronous preset or output enable
- Power-on preset to all "1" of internal registers
- f(max) = 14.28MHz
- 650mW power dissipation (typical)
- TTL compatible
- Single +5V supply
- Open collector and tri-state versions

APPLICATIONS

- Interface protocols
- Sequence detectors
- Peripheral controllers
- Timing generators
- Sequential circuits
- Elevator controllers
- Security locking systems
- Counters
- Shift registers

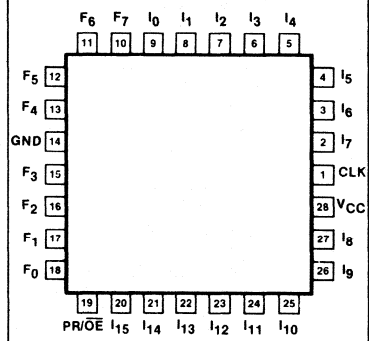
PIN CONFIGURATION

R,F, N PACKAGE*



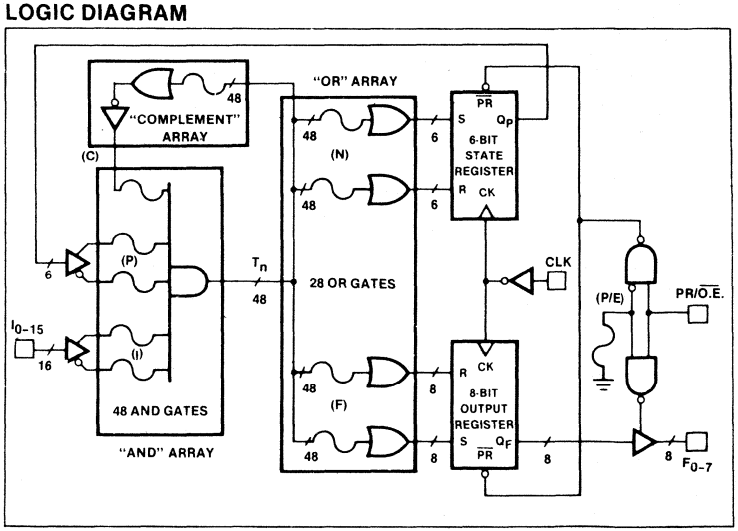
*F = Cerdip R = Flat Pack
N = Plastic

G PACKAGE



G = Leadless

LOGIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER		RATING Min	RATING Max	UNIT
V_{CC}	Supply voltage		+7	Vdc
V_{IN}	Input voltage		+5.5	Vdc
V_{OUT}	Output voltage		+5.5	Vdc
I_{IN}	Input currents	−30	+30	mA
I_{OUT}	Output currents		+100	mA
	Temperature range			°C
T_A	Operating			
	N82S104/105	0	+75	
	S82S104/105	−55	+125	
T_{STG}	Storage	−65	+150	

THERMAL RATINGS

TEMPERATURE	MILITARY	COMMERCIAL
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

DC ELECTRICAL CHARACTERISTICS

N82S104/105: $0°C \le T_A \le 75°C$, $4.75V \le V_{CC} \le 5.25V$
S82S104/105: $-55°C \le T_A \le +125°C$, $4.5V \le V_{CC} \le 5.5V$

PARAMETER		TEST CONDITIONS	N82S104/105 Min	N82S104/105 Typ[2]	N82S104/105 Max	S82S104/105 Min	S82S104/105 Typ[2]	S82S104/105 Max	UNIT
	Input voltage[3]								V
V_{IH}	High	V_{CC} = Max	2			2			
V_{IL}	Low	V_{CC} = Min			0.85			0.8	
V_{IC}	Clamp[3,4]	V_{CC} = Min, I_{IN} = −18mA		−0.8	−1.2		−0.8	−1.2	
	Output voltage	V_{CC} = Min							V
V_{OH}	High (82S105)[3,5]	I_{OH} = −2mA	2.4			2.4			
V_{OL}	Low[3,6]	I_{OL} = 9.6mA		0.35	0.45		0.35	0.50	
	Input current								μA
I_{IH}	High	V_{IN} = 5.5V		<1	25		<1	50	
I_{IL}	Low	V_{IN} = 0.45V		−10	−100		−10	−150	
I_{IL}	Low (CK input)	V_{IN} = 0.45V		−50	−250		−50	−350	
	Output current	V_{CC} = Max							
I_{OLK}	Leakage[7]	V_{OUT} = 5.5V		1	40		1	60	μA
$I_{O(OFF)}$	Hi-Z state (82S105)[7]	V_{OUT} = 5.5V		1	40		1	60	μA
		V_{OUT} = 0.45V		−1	−40		−1	−60	
I_{OS}	Short circuit (82S105)[4,8]	V_{OUT} = 0V	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current[9]	V_{CC} = Max		120	180		120	185	mA
	Capacitance[7]	V_{CC} = 5.0V							pF
C_{IN}	Input	V_{IN} = 2.0V		8			8		
C_{OUT}	Output	V_{OUT} = 2.0V		10			10		

NOTES

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device of these or any other condition above those indicated in the operation of the device specifications is not implied.
2. All typical values are at V_{CC} = 5V, T_A = 25°C.
3. All voltage values are with respect to network ground terminal.
4. Test one at a time.
5. Measured with V_{IL} applied to $\overline{O.E.}$ and a logic high stored, or with V_{IH} applied to PR.
6. Measured with a programmed logic condition for which the output is at a low logic level, and V_{IL} applied to PR/$\overline{O.E.}$ Output sink current is supplied thru a resistor to V_{CC}.
7. Measured with V_{IH} applied to PR/$\overline{O.E.}$.
8. Duration of short circuit should not exceed 1 second.
9. I_{CC} is measured with the PR/$\overline{O.E.}$ input grounded, all other inputs at 4.5V and the outputs open.

COMMERCIAL

AC ELECTRICAL CHARACTERISTICS $R_1 = 470\Omega$, $R_2 = 1k\Omega$, $C_L = 30pF$

N82S104/105: $0°C \leq T_A \leq +75°C$, $4.75V \leq V_{CC} \leq 5.25V$

PARAMETER		TO	FROM	N82S104/105			UNIT
				Min	Typ[1]	Max	
	Pulse width						ns
T_{CKH}	Clock[2] high	CK−	CK+	25	15		
T_{CKL}	Clock low	CK+	CK−	25	15		
T_{CKP1} B	Period (w/o c-array)	CK+	CK+	80	40		
T_{CKP2} B	Period (w/c-array)	CK+	CK+	120	60		
T_{PRH}	Preset pulse	PR+	PR−	25	15		
	Set-up time[3]						ns
T_{IS1} A	Input	CK+	Input±	60			
T_{IS1} B	Input	CK+	Input±	50			
T_{IS1} C	Input	CK+	Input±	42			
T_{IS2} A	Input (through Complement array)	CK+	Input±	90			
T_{IS2} B	Input (through Complement array)	CK+	Input	80			
T_{IS2} C	Input (through Complement array)	CK+	Input	72			
T_{VS}	Power-on preset	CK−	V_{CC}+	0	−10		
T_{PRS}	Preset	CK−	PR−	0	−10		
	Hold time						ns
T_{IH}	Input	Input±	CK+	5	−10		
	Propagation delay						ns
T_{CKO}	Clock	Output±	CK+		15	30	
T_{OE}	Output enable	Output−	O.E.−		20	30	
T_{OD}	Output disable	Output+	O.E.+		20	30	
T_{PR}	Preset	Output+	PR+		18	30	
T_{PPR}	Power-on preset	Output+	V_{CC}+		0	10	
	Frequency of operation						MHz
f_{MAX}	w/o c-array B					14.28	
f_{MAX}	w/c-array B					8.33	

MILITARY

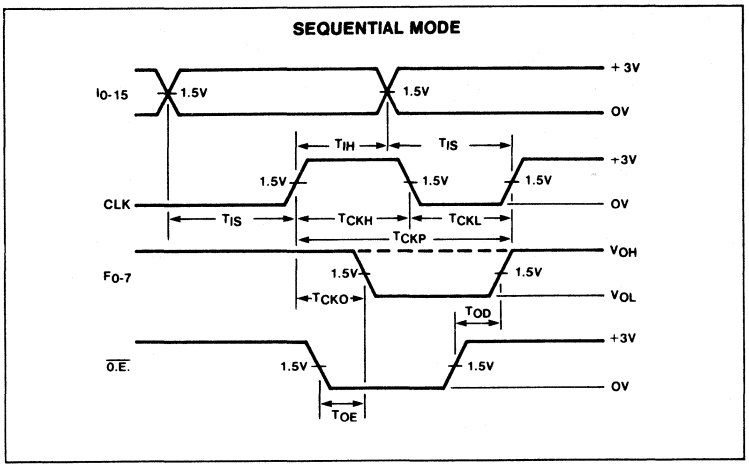
AC ELECTRICAL CHARACTERISTICS $R_1 = 470\Omega$, $R_2 = 1k\Omega$, $C_L = 30pF$

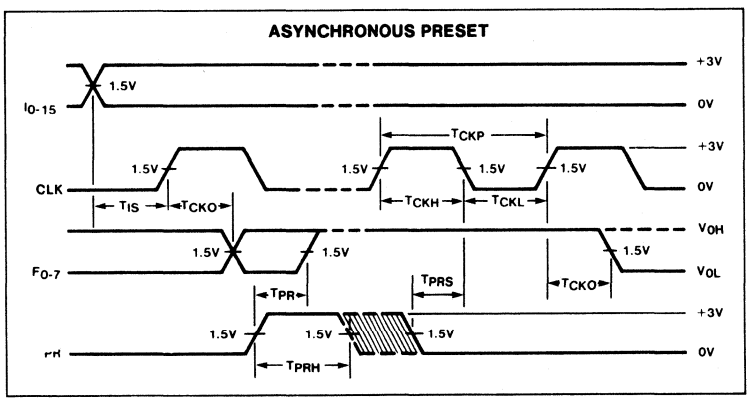
S82S104/105: $-55°C \leq T_A \leq +125°C$, $4.5V \leq V_{CC} \leq 5.5V$

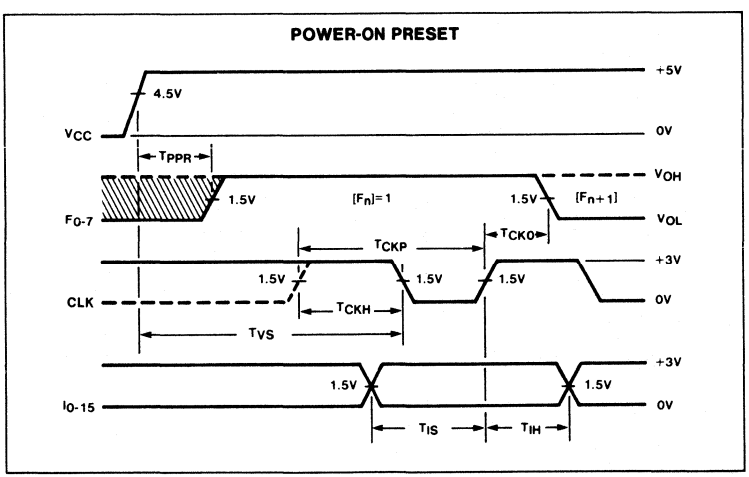
PARAMETER		TO	FROM	S82S104/105			UNIT
				Min	Typ[1]	Max	
	Pulse width						ns
T_{CKH}	Clock[2] high	CK−	CK+	40	15		
T_{CKL}	Clock low	CK+	CK−	40	15		
T_{CKP} A	Period (w/o c-array)	CK+	CK+	120	65		
T_{PRH}	Preset pulse	PR+	PR−	40	15		
	Set-up time						ns
T_{IS1} A	Input	CK+	Input±	80	40		
T_{IS2} A	Input (through Complement array)	CK+	Input±	120	70		
T_{VS}	Power-on preset	CK−	V_{CC}+	5	10		
T_{PRS}	Preset	CK−	PR−	5	−10		
	Hold time						ns
T_{IH}	Input	Input±	CK+		−10	10	
	Propagation delay						ns
T_{CKO}	Clock	Output±	$\overline{CK}$+		25	40	
T_{OE}	Output enable	Output−	$\overline{O.E.}$−		20	40	
T_{OD}	Output disable	Output+	$\overline{O.E.}$+		20	40	
T_{PR}	Preset	Output+	PR+		25	45	
T_{PPR}	Power-on preset	Output+	V_{CC}+		0	20	

NOTES
1. All typical values are at $V_{CC} = 5V$, $T_A = 25°C$.
2. To prevent spurious clocking, clock rise time (10% -90%) ≤ 10ns

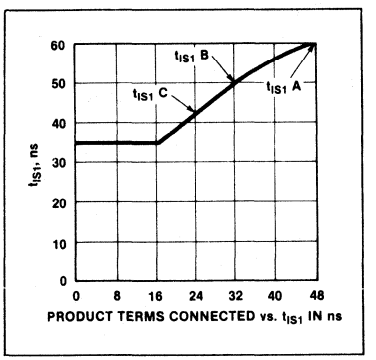
TIMING DIAGRAMS



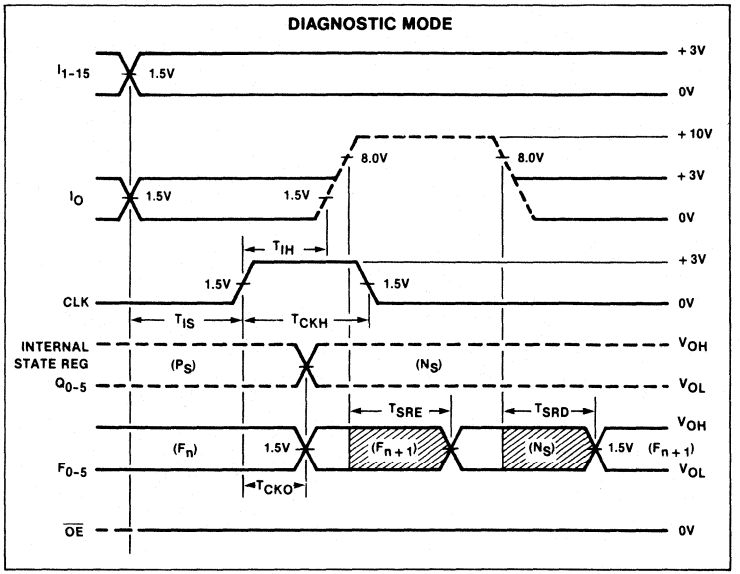




The speed performance of the Logic Sequencer depends upon the number of product terms connected to the sum term lines. The effect of using all 48 product terms, i.e., having them connected in the AND matrix, does not have an appreciable effect on the speed performance of the device. The major effect is the capacitive loading of each product term on the sum term lines. The effect of this loading can be seen in the figure below titled *Product Terms Connected vs. t_{IS1} in ns.* The AC characteristics table contain three performance limits for the parameters; t_{IS1} and t_{IS2}. The first, t_{IS1} A is representative of a device having up to 48 product terms connected to any sum term line. The t_{IS1} B is representative of a device having up to 32 product terms connected to any sum term line. And the t_{IS1} C represents a device having up to 24 product terms connected to any sum term line. The three entries of t_{IS2} A, B and C are the same 48, 32 and 24 connected product terms as the t_{IS1} notation. Product term lines are lines denoted by the numbers T0 to T47, and the sum term lines are denoted by the letter blocks N and F, on the FPLS logic diagram, page 4-17. This input setup time vs. the number of product term connections to sum term lines is shown in the figure. When other than 48, 32 or 24 product terms are connected to one sum term line the speed performance can be estimated from the figure.



TIMING DIAGRAMS (Cont'd)



TIMING DEFINITIONS

T_{CKH} Width of input clock pulse.

T_{CKL} Interval between clock pulses.

T_{CKP} Clock period—when not using Complement array.

T_{IS1} Required delay between beginning of valid input and positive transition of clock.

T_{CKP_2} Clock period—when using Complement array.

T_{IS2} Required delay between beginning of valid Input and positive transition of clock, when using optional Complement Array (two passes necessary through the AND array).

T_{VS} Required delay between V_{CC} (after power-on) and negative transition of clock preceding first reliable clock pulse.

T_{PRS} Required delay between negative transition of asynchronous Preset and negative transition of clock preceding first reliable clock pulse.

T_{IH} Required delay between positive transition of clock and end of valid Input data.

T_{CKO} Delay between positive transition of clock and when Outputs become valid (with PR/$\overline{O.E.}$ low).

T_{OE} Delay between beginning of Output Enable Low and when Outputs become valid.

T_{OD} Delay between beginning of Output Enable High and when Outputs are in the off state.

T_{SRE} Delay between input I_O transition to Diagnostic mode and when the Outputs reflect the contents of the State Register.

T_{SRD} Delay between input I_O transition to Logic mode and when the Outputs reflect the contents of the Output Register.

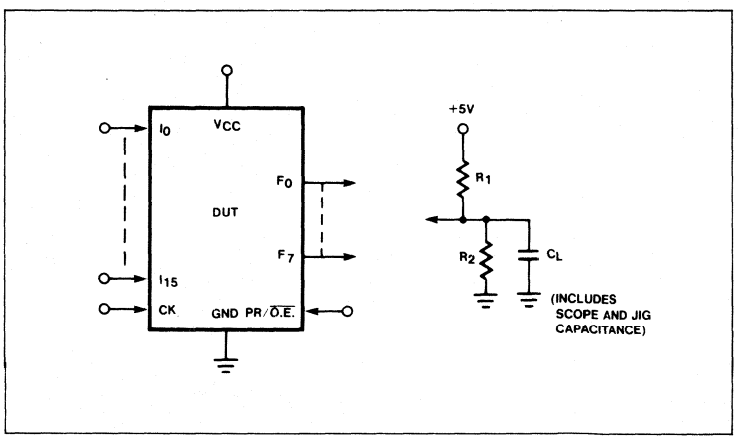
T_{PR} Delay between positive transition of Preset and when Outputs become valid at "1".

T_{PPR} Delay between V_{CC} (after power-on) and when Outputs become preset at "1".

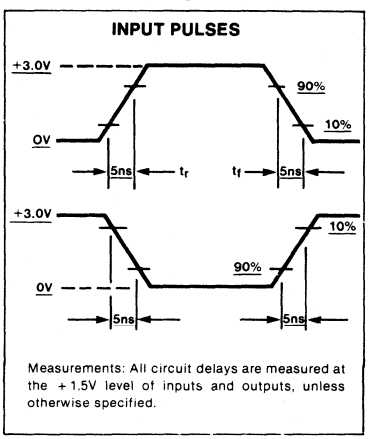
T_{PRH} Width of preset input pulse.

f_{MAX} Maximum clock frequency.

TEST LOAD CIRCUIT

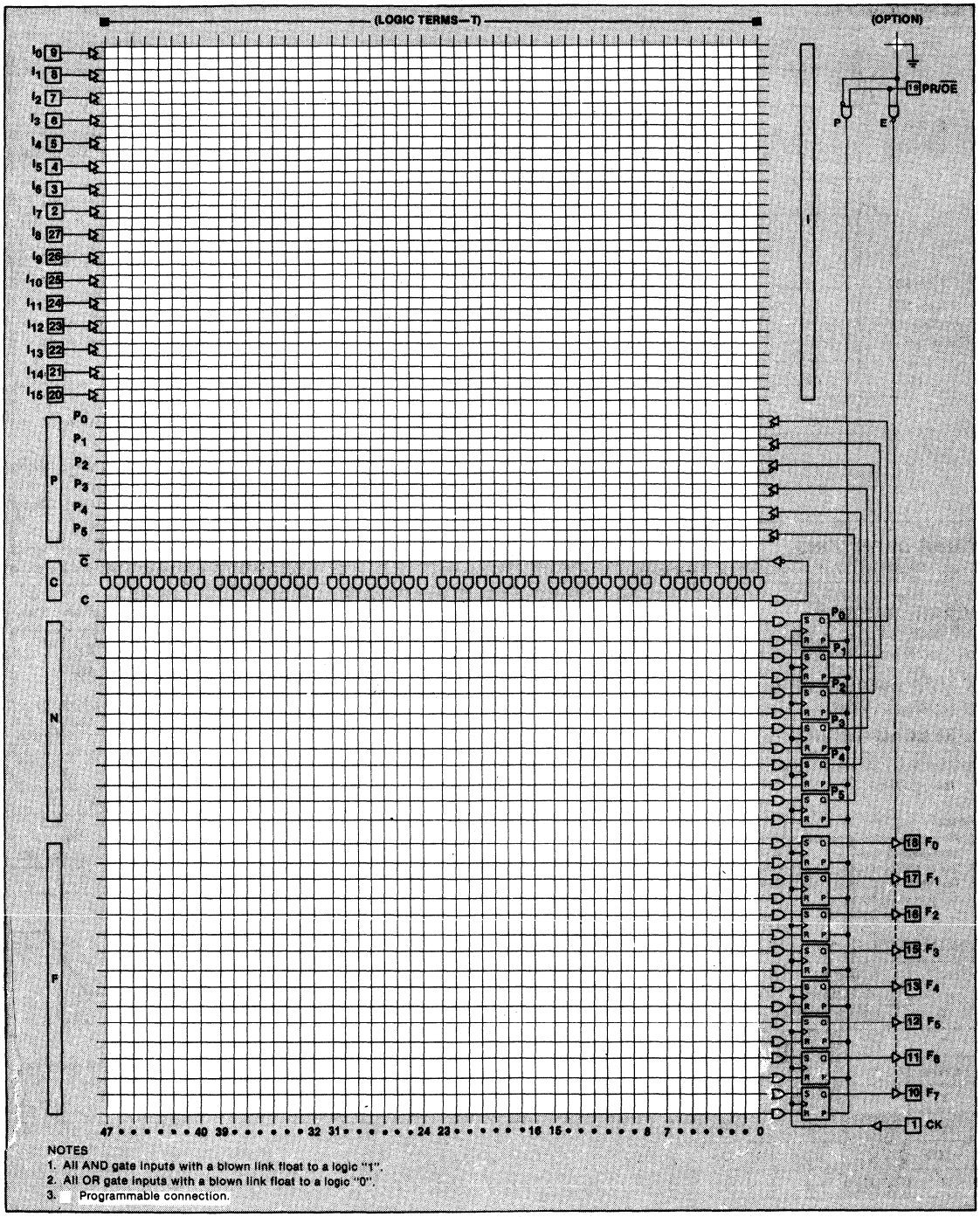


VOLTAGE WAVEFORM



Measurements: All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

FPLS LOGIC DIAGRAM



NOTES
1. All AND gate inputs with a blown link float to a logic "1".
2. All OR gate inputs with a blown link float to a logic "0".
3. ▪ Programmable connection.

PIN DESIGNATION

PIN NO.	SYMBOL	NAME AND FUNCTION	POLARITY
1	CK	**CLOCK** The clock input to the State and Output Registers. A Low-to-High transition on this line is necessary to update the contents of both registers.	
[2-8 20-27]	I_{1-15}	**LOGIC INPUTS** The 15 external inputs to the AND array used to program jump conditions between machine states, as determined by a given logic sequence.	
9	I_0	**LOGIC/DIAGNOSTIC INPUT** A 16th external logic input to the AND array, as above, when exercised with standard TTL levels. When I_0 is held at +10V, device outputs F_{0-5} reflect the contents of State Register bits P_{0-5}. The contents of the Output Register remain unaltered.	
[10-13 15-18]	F_{0-7}	**LOGIC/DIAGNOSTIC OUTPUTS** Eight device outputs which normally reflect the contents of Output Register bits Q_{0-7}, when enabled. When I_0 is held at +10V, $F_{0-5} = (P_{0-5})$, and $F_{6,7}$ = Logic "I".	
19	PR/$\overline{O.E.}$	**PRESET OR $\overline{\text{OUTPUT ENABLE}}$ INPUT** A user programmable function: • **PRESET** Provides an asynchronous preset to logic "1" of all State and Output Register bits. Preset overrides Clock, and when held High, clocking is inhibited and F_{0-7} are High. Normal clocking resumes with the first full clock pulse following a High-to-Low clock transition, after Preset goes Low. • **$\overline{\text{OUTPUT ENABLE}}$** Provides an output enable function to buffers F_{0-7} from the Output Register.	Active-High (H) Active-Low (L)

VIRGIN STATE[1,2,3]

A factory shipped virgin device contains all fusible links intact, such that:

1. PR/$\overline{OE}$ option is set to PR.
2. All product terms are disabled.
3. All S/R flip-flop inputs are disabled.
4. Test array is programmed with standard test pattern.

NOTES
1. All outputs will be at "1", as preset by initial power-up procedure.
2. Device can be clocked via test array function.
3. Test array function MUST be deleted before incorporating user program.

TRUTH TABLE (All flip-flops)

V_{CC}	INPUT OPTION		CK	S	R	ALL F/F
	PR	$\overline{O.E.}$				Q_P/Q_F
	H		X	X	X	H
	L	X	↑	L	L	Q_P/Q_F
+5V	L	X	↑	L	H	L
	L	X	↑	H	L	H
	L	X	↑	H	H	INDET.
Power On	X	X	X	X	X	H

TRUTH TABLE (Output Control)

I_0	INPUT OPTION(1)		F_N
	PR	$\overline{O.E.}$	
*	H		H
+10V	L		Q_P
X	L		Q_F
*		H	H/Hi-Z
+10V		L	Q_P
X		L	Q_F

* H, L or +10V
X Don't Care — $V_{IN} \leq 5.5V$
↑ L to H transition
(1) User programmable option. Either preset or O.E. function may be selected.

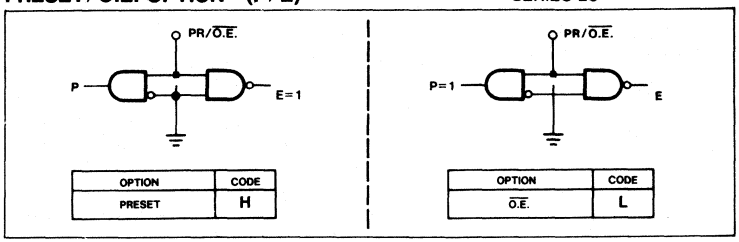
LOGIC PROGRAMMING

The FPLS can be programmed by means of Logic programming equipment.

With Logic programming, the AND/OR gate input connections necessary to implement the desired logic function are coded directly from the State Diagram using the Program Table on the following page.

In this Table, the logic state or action of control variables C, I, P, N, and F, associated with each Transition Term T_n, is assigned a symbol which results in the proper fusing pattern of corresponding link pairs, defined as follows:

PRESET/$\overline{O.E.}$ OPTION - (P/E)



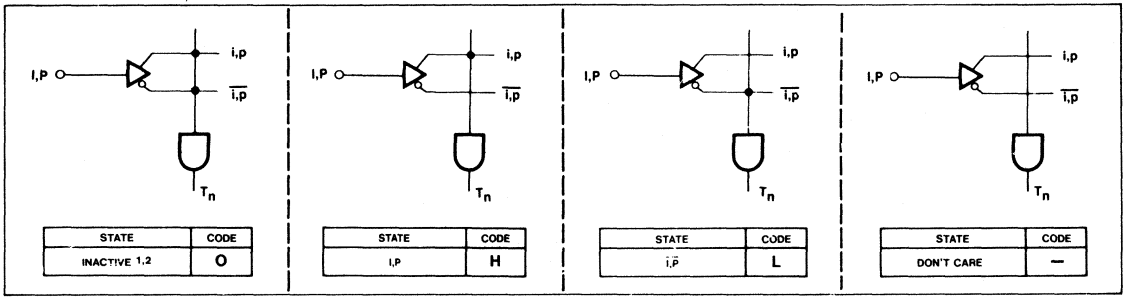
OPTION	CODE
PRESET	H

OPTION	CODE
$\overline{O.E.}$	L

PROGRAMMING THE 82S104/105

The 82S104/105 has a power-up preset feature. This feature insures that the device will power-up in a known state with all register elements (state and output register) at a logic high (H). When programming the device it is important to realize this is the initial state of the device. You *must* provide a next state jump if you do not wish to use all highs (H) as the present state.

"AND" ARRAY - (I), (P)



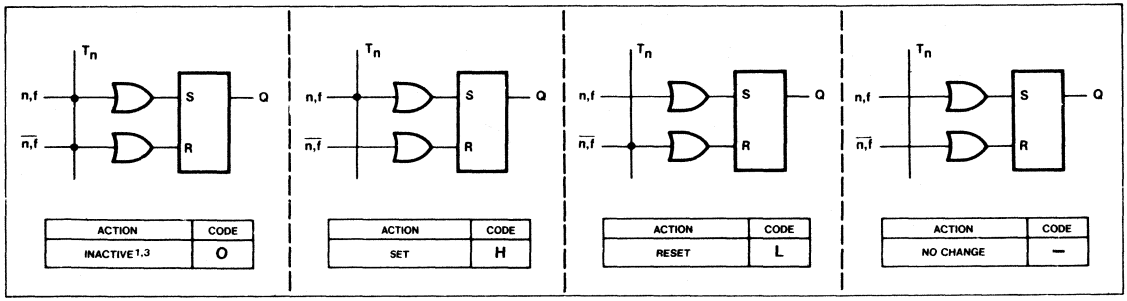
STATE	CODE
INACTIVE 1,2	O

STATE	CODE
I,P	H

STATE	CODE
$\overline{i,p}$	L

STATE	CODE
DON'T CARE	—

"OR" ARRAY - (N), (F)



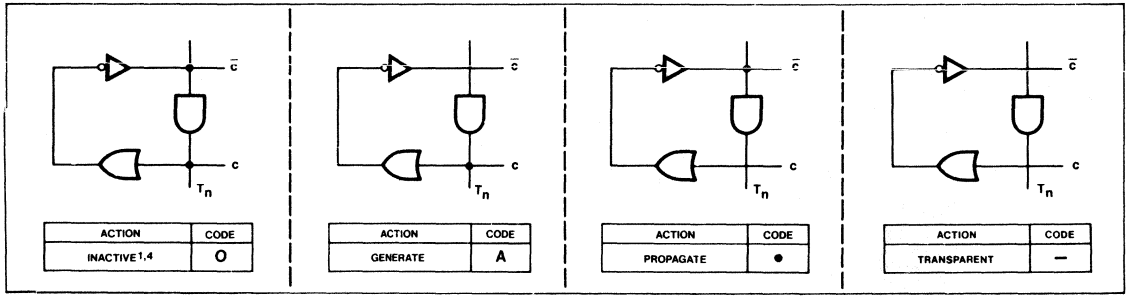
ACTION	CODE
INACTIVE 1,3	O

ACTION	CODE
SET	H

ACTION	CODE
RESET	L

ACTION	CODE
NO CHANGE	—

"COMPLEMENT" ARRAY - (C)



ACTION	CODE
INACTIVE 1,4	O

ACTION	CODE
GENERATE	A

ACTION	CODE
PROPAGATE	•

ACTION	CODE
TRANSPARENT	—

NOTES

1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates T_n.
2. Any gate T_n will be unconditionally inhibited if any one of its I or P link pairs is left intact.
3. To prevent simultaneous Set and Reset flip-flop commands, this state is not allowed for N and F link pairs coupled to active gates T_n (see flip-flop truth tables).
4. To prevent oscillations, this state is not allowed for C link pairs coupled to active gates T_n.

FPLS PROGRAM TABLE

CUSTOMER NAME _____

PURCHASE ORDER # _____

SIGNETICS DEVICE # CF (XXXX)

CUSTOMER SYMBOLIZED PART # _____

TOTAL NUMBER OF PARTS _____

PROGRAM TABLE _____

REV _____

DATE _____

PROGRAM TABLE ENTRIES

AND

GENERATE	A	Cn
PROPAGATE	*	
TRANSPARENT	—	

I, P	H	Im, Ps
I, $\overline{P}$	L	
DON'T CARE	—	

OR

SET	H	Ns, Fr
RESET	L	
NO CHANGE	—	

OPTION

PRESET	H	P/E
$\overline{OE}$	L	

TERM	AND: C_n	AND: INPUT (Im) 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	AND: PRESENT STATE (Ps) 5	4	3	2	1	0	REMARKS	OPTION (P/E) / OR: NEXT STATE (Ns) 5	4	3	2	1	0	OR: OUTPUT (Fr) 7	6	5	4	3	2	1	0
0																																						
1																																						
2																																						
3																																						
4																																						
5																																						
6																																						
7																																						
8																																						
9																																						
10																																						
11																																						
12																																						
13																																						
14																																						
15																																						
16																																						
17																																						
18																																						
19																																						
20																																						
21																																						
22																																						
23																																						
24																																						
25																																						
26																																						
27																																						
28																																						
29																																						
30																																						
31																																						
32																																						
33																																						
34																																						
35																																						
36																																						
37																																						
38																																						
39																																						
40																																						
41																																						
42																																						
43																																						
44																																						
45																																						
46																																						
47																																						
PIN NO.		20	21	22	23	24	25	26	27	2	3	4	5	6	7	8	9													10	11	12	13	15	16	17	18
VARIABLE NAME																																						

NOTES

1. The FPLS is shipped with all links initially intact. Thus, a background of "0" for all Terms, and an "H" for the P/E option, exists in the table, shown BLANK instead for clarity.
2. Unused C_n, Im, and Ps bits are normally programmed Don't Care (—).
3. Unused transition and output Terms can be left blank.
4. Letters in variable fields are used as identifiers by logic type programmers.

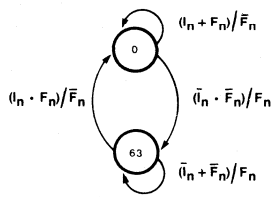
TEST ARRAY

The FPLS may be subjected to AC and DC parametric tests prior to programming via an on chip test array.

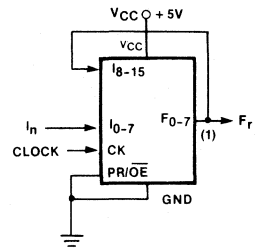
The array consists of test transition terms 48 and 49, factory programmed as shown below.

Testing is accomplished by clocking the FPLS and applying the proper input sequence to I_{0-7} as shown in the test circuit timing diagram.

STATE DIAGRAM



FPLS UNDER TEST



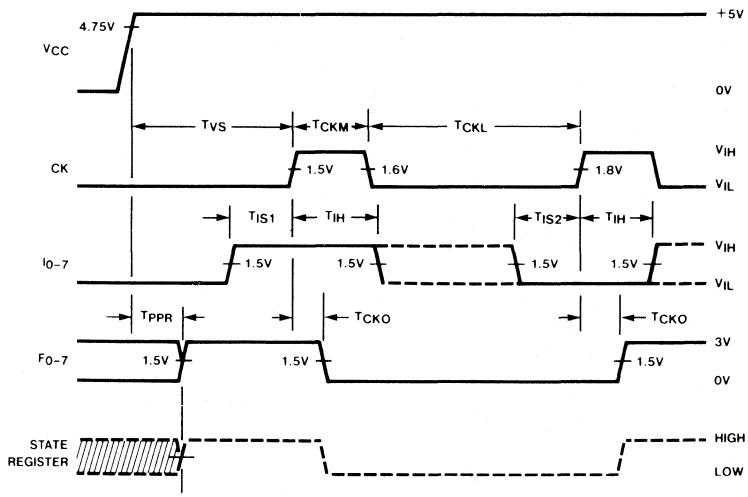
TEST ARRAY PROGRAM

TERM	C	AND: INPUT (Im) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	AND: PRESENT STATE (Ps) 5 4 3 2 1 0	OR: NEXT STATE (Ns) 5 4 3 2 1 0	OR: OUTPUT (Fr) 7 6 5 4 3 2 1 0
48	A	H H H H H H H H H H H H H H H H	H H H H H H	L L L L L L	L L L L L L L L
49	●	L L L L L L L L L L L L L L L L	L L L L L L	H H H H H H	H H H H H H H H

OPTION (P/E): H

Both terms 48 and 49 must be deleted during user programming to avoid interfering with the desired logic function. This is accomplished automatically by any Signetics' qualified programming equipment.

TEST CIRCUIT TIMING DIAGRAM



TEST ARRAY DELETED

TERM	C	AND: INPUT (Im) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	AND: PRESENT STATE (Ps) 5 4 3 2 1 0	OR: NEXT STATE (Ns) 5 4 3 2 1 0	OR: OUTPUT (Fr) 7 6 5 4 3 2 1 0
48	—	H H H H H H H H H H H H H H H H	H H H H H H	— — — — — —	— — — — — — — —
49	●	L L L L L L L L L L L L L L L L	L L L L L L	— — — — — —	— — — — — — — —

OPTION (P/E): H

DESCRIPTION

The 82S104 (open collector outputs) and the 82S105 (tri-state outputs) are bipolar, programmable state machines of the Mealy type. They contain logic AND-OR gate arrays with user programmable connections which control the inputs of on-chip State and Output registers. These consist respectively of 6 Q_P, and 8 Q_F edge triggered, clocked S/R flip-flops, with an asynchronous Preset option. All flip-flops are unconditionally preset to "1" during power turn on.

The AND array combines 16 external inputs $I_{0\text{-}15}$ with 6 internal inputs $P_{0\text{-}5}$ fed back from the State register to form up to 48 Transition terms (AND terms). All Transition terms can include True, False, or Don't Care states of the controlling variables, and are merged in the OR array to issue next-state and next-output commands to their respective registers on the Low to High transition of the Clock pulse. Both True and Complement Transition terms can be generated by optional use of the internal input variable (C) from the Complement array. Also, if desired, the Preset input can be converted to $\overline{\text{Output-Enable}}$ function, as an additional user programmable option.

Both devices are available in commercial temperature ranges. For the commercial temperature range (0°C to +75°C) specify N82S104A F or N; N82S105A F or N.

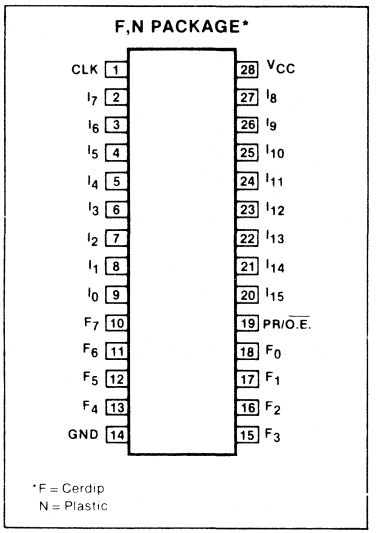
FEATURES

- Field programmable (Ni-Cr link)
- 16 input variables
- 8 output functions
- 48 transition terms
- 6-bit state register
- 8-bit output register
- Transition complement array
- Positive edge trigger clock
- Programmable asynchronous preset or output enable
- Power-on preset to all "1" of internal registers
- f(max) = 20MHz
- 650mW power dissipation (typical)
- TTL compatible
- Single +5V supply
- Open collector and tri-state versions

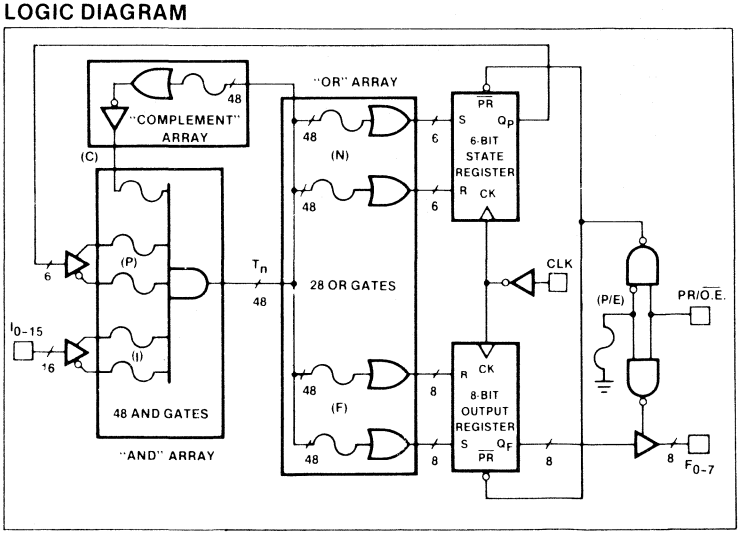
APPLICATIONS

- Interface protocols
- Sequence detectors
- Peripheral controllers
- Timing generators
- Sequential circuits
- Elevator controllers
- Security locking systems
- Counters
- Shift registers

PIN CONFIGURATION



LOGIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER		RATING Min	RATING Max	UNIT
V_{CC}	Supply voltage		+7	Vdc
V_{IN}	Input voltage		+5.5	Vdc
V_{OUT}	Output voltage		+5.5	Vdc
I_{IN}	Input currents	−30	+30	mA
I_{OUT}	Output currents		+100	mA
	Temperature range			°C
T_A	Operating			
	N82S104/105	0	+75	
T_{STG}	Storage	−65	+150	

THERMAL RATINGS

TEMPERATURE	COMMERCIAL
Maximum junction	150°C
Maximum ambient	75°C
Allowable thermal rise ambient to junction	75°C

DC ELECTRICAL CHARACTERISTICS $0°C \le T_A \le +75°C, 4.75V \le V_{CC} \le 5.25V$

PARAMETER		TEST CONDITIONS	N82S104A/105A Min	Typ[2]	Max	UNIT
	Input voltage[3]					V
V_{IH}	High	V_{CC} = Max	2			
V_{IL}	Low	V_{CC} = Min			0.85	
V_{IC}	Clamp[3,4]	V_{CC} = Min, I_{IN} = −18mA		−0.8	−1.2	
	Output voltage	V_{CC} = Min				V
V_{OH}	High (82S105)[3,5]	I_{OH} = −2mA	2.4			
V_{OL}	Low[3,6]	I_{OL} = 9.6mA		0.35	0.45	
	Input current					μA
I_{IH}	High	V_{IN} = 5.5V		<1	25	
I_{IL}	Low	V_{IN} = 0.45V		−10	−100	
I_{IL}	Low (CK input)	V_{IN} = 0.45V		−50	−250	
	Output current	V_{CC} = Max				
I_{OLK}	Leakage[7]	V_{OUT} = 5.5V		1	40	μA
$I_{O(OFF)}$	Hi-Z state (82S105)[7]	V_{OUT} = 5.5V		1	40	μA
		V_{OUT} = 0.45V		−1	−40	
I_{OS}	Short circuit (82S105)[4,8]	V_{OUT} = 0V	−20		−70	mA
I_{CC}	V_{CC} supply current[9]	V_{CC} = Max		120	180	mA
	Capacitance[7]	V_{CC} = 5.0V				pF
C_{IN}	Input	V_{IN} = 2.0V		8		
C_{OUT}	Output	V_{OUT} = 2.0V		10		

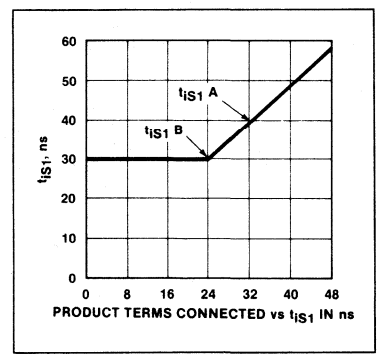
NOTES

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device of these or any other condition above those indicated in the operation of the device specifications is not implied.
2. All typical values are at V_{CC} = 5V, T_A = 25°C.
3. All voltage values are with respect to network ground terminal.
4. Test one at a time.
5. Measured with V_{IL} applied to $\overline{O.E.}$ and a logic high stored, or with V_{IH} applied to PR.
6. Measured with a programmed logic condition for which the output is at a low logic level, and V_{IL} applied to PR/$\overline{O.E.}$ Output sink current is supplied thru a resistor to V_{CC}.
7. Measured with V_{IH} applied to PR/$\overline{O.E.}$.
8. Duration of short circuit should not exceed 1 second.
9. I_{CC} is measured with the PR/$\overline{O.E.}$ input grounded, all other inputs at 4.5V and the outputs open.

AC ELECTRICAL CHARACTERISTICS $R_1 = 470\Omega$, $R_2 = 1k\Omega$, $C_L = 30pF$ $0°C \leq T_A \leq +75°C$, $4.75V \leq V_{CC} \leq 5.25V$

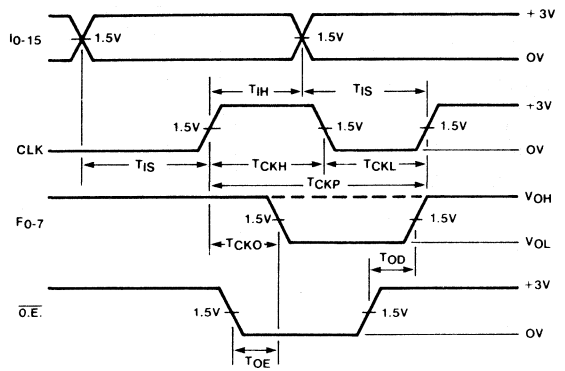
PARAMETERS		TO	FROM	N82S104A/105A			UNIT
				Min	Typ[1]	Max	
	Pulse width						ns
T_{CKH}	Clock high[2]	CK−	CK+	25	15		
T_{CKL}	Clock low	CK+	CK−	25	15		
T_{CKP1} B	Period (w/o c-array)	CK+	CK+	50	40		
T_{CKP2} B	Period (w/c-array)	CK+	CK+	80	50		
T_{PRH}	Preset pulse	PR+	PR−	25	15		
	Set-up time[3]						ns
T_{IS1} A	Input ≤32 P-terms	CK+	Input±	40			
T_{IS1} B	Input ≤24 P-terms	CK+	Input±	30			
T_{IS2} A	Input (through Complement array)	CK+	Input±	70			
T_{IS2} B	Input (through Complement array)	CK+	Input±	60			
T_{VS}	Power-on preset	CK−	V_{CC}+	0	−10		
T_{PRS}	Preset	CK−	PR−	0	−10		
	Hold time						ns
T_{IH}	Input	Input±	CK+	5	−10		
	Propagation delay						ns
T_{CKO}	Clock	Output±	$\overline{CK}$+		15	20	
T_{OE}	Output enable	Output−	$\overline{O.E.}$−		20	30	
T_{OD}	Output disable	Output+	$\overline{O.E.}$+		20	30	
T_{PR}	Preset	Output+	PR+		18	30	
T_{PPR}	Power-on preset	Output+	V_{CC}+		0	10	
	Frequency of operation						MHz
f_{MAX} A	w/o c-array					16.6	
f_{MAX} B	w/o c-array					20	

NOTES:
1. All typical values are at $V_{CC} \pm 5V$, $T_A = 25°C$.
2. To prevent spurious clocking, clock rise time (10%-90%) ≤ 10ns.
3. The speed performance of the Logic Sequencer depends upon the number of product terms connected to the sum term lines. The effect of using all 48 product terms, i.e., having them connected in the AND matrix, does not have an appreciable effect on the speed performance of the device. The major effect is the capacitive loading of each product term on the sum term lines. The effect of this loading can be seen in the figure to the right titled *Product Terms Connected vs. t_{iS1} in ns*. The AC characteristics table contain two performance limits for each of the parameters; t_{is1} and t_{is2}. The first, t_{is1} A is representative of a device having up to 32 product terms connected to any sum term line. The t_{is1} B represents a device having up to 24 product terms connected to any sum term lin. The two entries of t_{is2} A and B are the same 32 and 24 connected product terms as the t_{is1} notations. Product term lines are lines deonted by the numbers T0 to T47, and the sum term lines are deonted by the letter blocks N and F, on the FPLS logic diagram, page 4-27. This input setup time vs. the number of product term connections to sum term lines is shown in the figure. When other than 32 or 24 product terms are connected to one sum term line the speed performance can be estimated from the figure.

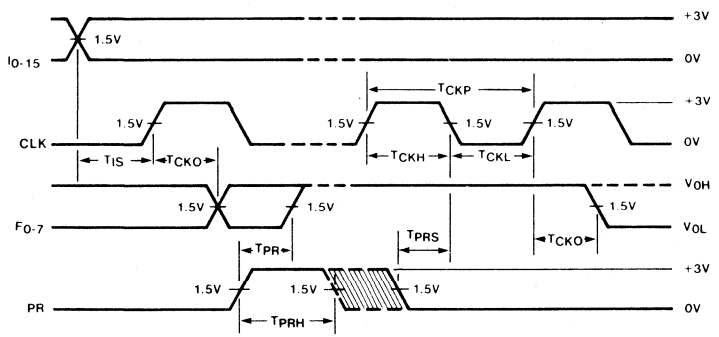


TIMING DIAGRAMS

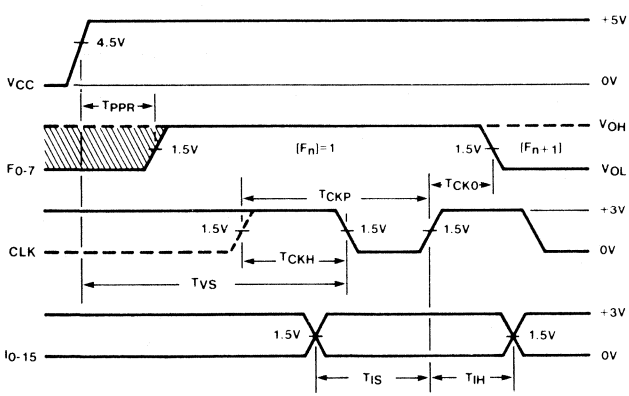
SEQUENTIAL MODE



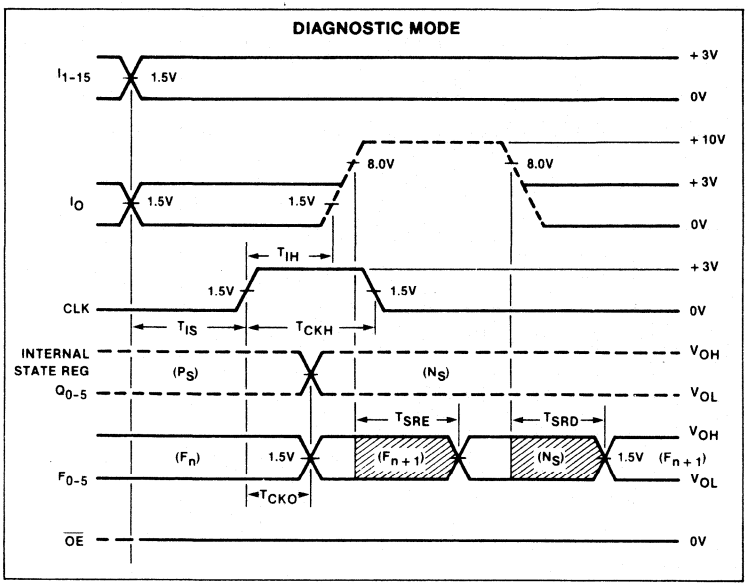
ASYNCHRONOUS PRESET



POWER-ON PRESET



TIMING DIAGRAMS (Cont'd)



TIMING DEFINITIONS

T_{CKH} Width of input clock pulse.

T_{CKL} Interval between clock pulses.

T_{CKP} Clock period—when not using Complement array.

T_{IS1} Required delay between beginning of valid input and positive transition of clock.

T_{CKP_2} Clock period—when using Complement array.

T_{IS2} Required delay between beginning of valid Input and positive transition of clock, when using optional Complement Array (two passes necessary through the AND array).

T_{VS} Required delay between V_{CC} (after power-on) and negative transition of clock preceding first reliable clock pulse.

T_{PRS} Required delay between negative transition of asynchronous Preset and negative transition of clock preceding first reliable clock pulse.

T_{IH} Required delay between positive transition of clock and end of valid Input data.

T_{CKO} Delay between positive transition of clock and when Outputs become valid (with PR/$\overline{O.E.}$ low).

T_{OE} Delay between beginning of Output Enable Low and when Outputs become valid.

T_{OD} Delay between beginning of Output Enable High and when Outputs are in the off state.

T_{SRE} Delay between input I_O transition to Diagnostic mode and when the Outputs reflect the contents of the State Register.

T_{SRD} Delay between input I_O transition to Logic mode and when the Outputs reflect the contents of the Output Register.

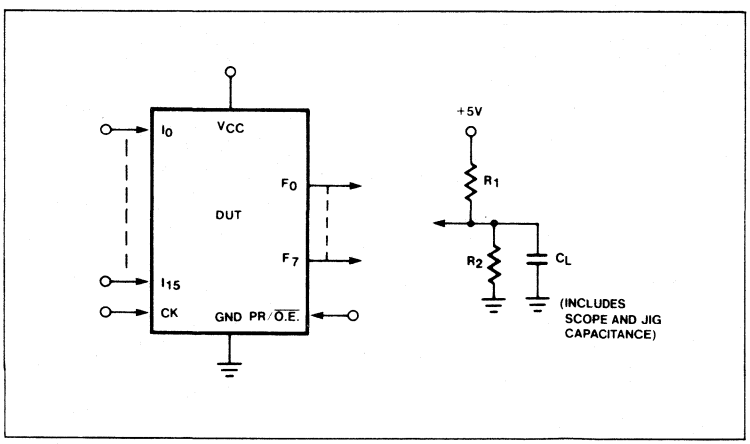
T_{PR} Delay between positive transition of Preset and when Outputs become valid at "1".

T_{PPR} Delay between V_{CC} (after power-on) and when Outputs become preset at "1".

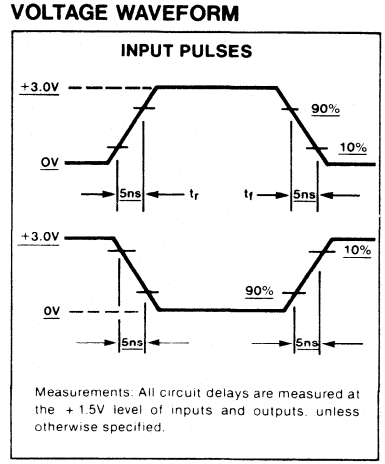
T_{PRH} Width of preset input pulse.

f_{MAX} Maximum clock frequency.

TEST LOAD CIRCUIT

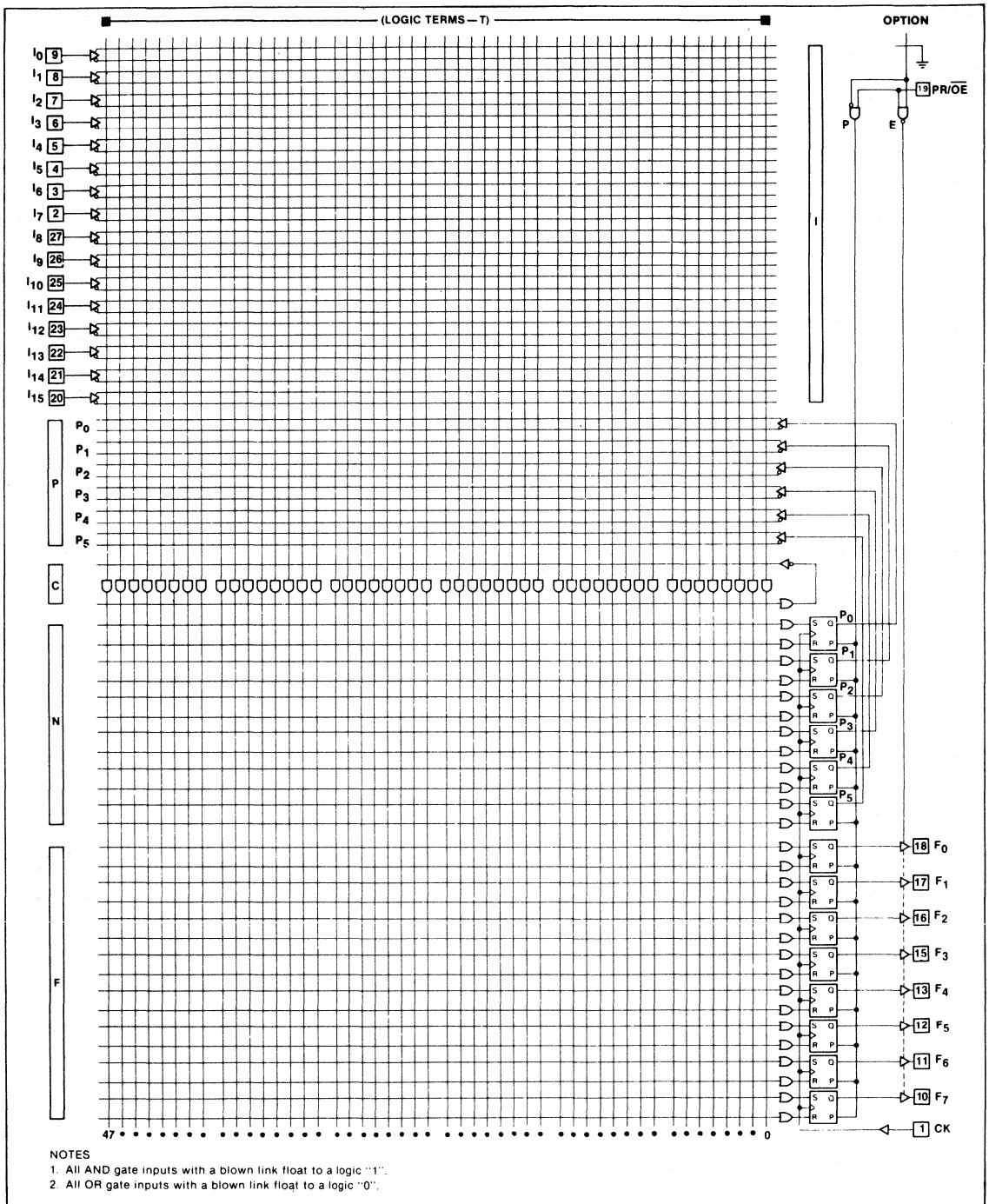


VOLTAGE WAVEFORM



Measurements: All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

FPLS LOGIC DIAGRAM



NOTES
1. All AND gate inputs with a blown link float to a logic "1".
2. All OR gate inputs with a blown link float to a logic "0".

PIN DESIGNATION

PIN NO.	SYMBOL	NAME AND FUNCTION	POLARITY
1	CK	**CLOCK** The clock input to the State and Output Registers. A Low-to-High transition on this line is necessary to update the contents of both registers.	
[2-8] [20-27]	I_{1-15}	**LOGIC INPUTS** The 15 external inputs to the AND array used to program jump conditions between machine states, as determined by a given logic sequence.	
9	I_0	**LOGIC/DIAGNOSTIC INPUT** A 16th external logic input to the AND array, as above, when exercised with standard TTL levels. When I_0 is held at +10V, device outputs F_{0-5} reflect the contents of State Register bits P_{0-5}. The contents of the Output Register remain unaltered.	
[10-13] [15-18]	F_{0-7}	**LOGIC/DIAGNOSTIC OUTPUTS** Eight device outputs which normally reflect the contents of Output Register bits Q_{0-7}, when enabled. When I_0 is held at +10V, $F_{0-5} = (P_{0-5})$, and $F_{6,7}$ = Logic "I".	
19	PR/$\overline{\text{O.E.}}$	**PRESET OR $\overline{\text{OUTPUT ENABLE}}$ INPUT** A user programmable function: • **PRESET** Provides an asynchronous preset to logic "1" of all State and Output Register bits. Preset overrides Clock, and when held High, clocking is inhibited and F_{0-7} are High. Normal clocking resumes with the first full clock pulse following a High-to-Low clock transition, after Preset goes Low. • **$\overline{\text{OUTPUT ENABLE}}$** Provides an output enable function to buffers F_{0-7} from the Output Register.	Active-High (H) Active-Low (L)

VIRGIN STATE[1,2,3]

A factory shipped virgin device contains all fusible links intact, such that:

1. PR/$\overline{\text{OE}}$ option is set to PR.
2. All product terms are disabled.
3. All S/R flip-flop inputs are disabled.
4. Test array is programmed with standard test pattern.

NOTES
1. All outputs will be at "1", as preset by initial power-up procedure.
2. Device can be clocked via test array function.
3. Test array function MUST be deleted before incorporating user program.

TRUTH TABLE (All flip-flops)

V_{CC}	INPUT OPTION		CK	S	R	ALL F/F
	PR	$\overline{\text{O.E.}}$				Q_P/Q_F
	H		X	X	X	H
	L	X	↑	L	L	Q_P/Q_F
+5V	L	X	↑	L	H	L
	L	X	↑	H	L	H
	L	X	↑	H	H	INDET.
Power On	X	X	X	X	X	H

TRUTH TABLE (Output Control)

I_0	INPUT OPTION[1]		F_N
	PR	$\overline{\text{O.E.}}$	
*	H		H
+10V	L		Q_P
X	L		Q_F
*		H	H/Hi-Z
+10V		L	Q_P
X		L	Q_F

* H, L or +10V
X Don't Care — V_{IN} < 5.5V
↑ L to H transition
(1) User programmable option. Either preset or O.E. function may be selected.

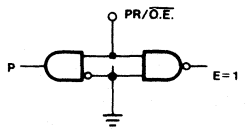
LOGIC PROGRAMMING

The FPLS can be programmed by means of Logic programming equipment.

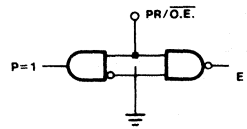
With Logic programming, the AND/OR gate input connections necessary to implement the desired logic function are coded directly from the State Diagram using the Program Table on the following page.

In this Table, the logic state or action of control variables C, I, P, N, and F, associated with each Transition Term T_n, is assigned a symbol which results in the proper fusing pattern of corresponding link pairs, defined as follows:

PRESET/$\overline{O.E.}$ OPTION - (P/E)



OPTION	CODE
PRESET	H

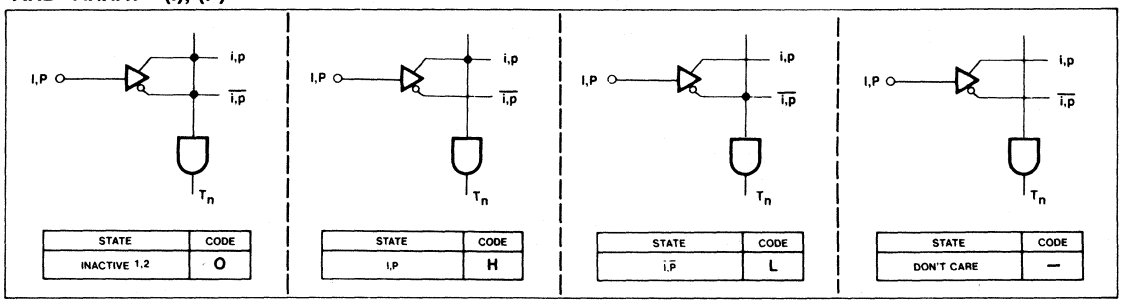


OPTION	CODE
$\overline{O.E.}$	L

PROGRAMMING THE 82S104/105

The 82S104/105 has a power-up preset feature. This feature insures that the device will power-up in a known state with all register elements (state and output register) at a logic high (H). When programming the device it is important to realize this is the initial state of the device. You *must* provide a next state jump if you do not wish to use all highs (H) as the present state.

"AND" ARRAY - (I), (P)



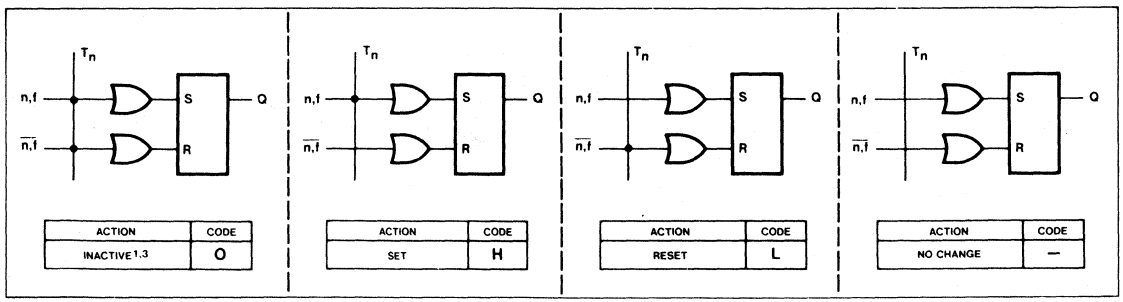
STATE	CODE
INACTIVE 1,2	O

STATE	CODE
I,P	H

STATE	CODE
$\overline{I,P}$	L

STATE	CODE
DON'T CARE	—

"OR" ARRAY - (N), (F)



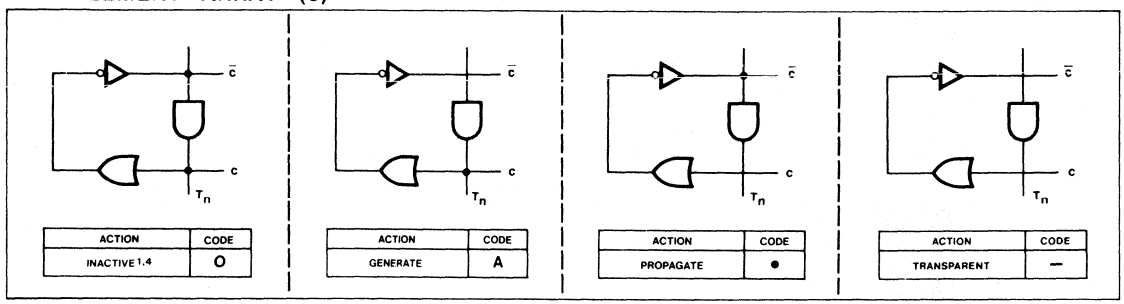
ACTION	CODE
INACTIVE 1,3	O

ACTION	CODE
SET	H

ACTION	CODE
RESET	L

ACTION	CODE
NO CHANGE	—

"COMPLEMENT" ARRAY - (C)



ACTION	CODE
INACTIVE 1,4	O

ACTION	CODE
GENERATE	A

ACTION	CODE
PROPAGATE	•

ACTION	CODE
TRANSPARENT	—

NOTES

1. This is the initial unprogrammed state of all link pairs. It is normally associated with all unused (inactive) AND gates T_n.
2. Any gate T_n will be unconditionally inhibited if any one of its L or P link pairs is left intact.
3. To prevent simultaneous Set and Reset flip-flop commands, this state is not allowed for N and F link pairs coupled to active gates T_n (see flip-flop truth tables).
4. To prevent oscillations, this state is not allowed for C link pairs coupled to active gates T_n.

INTEGRATED FUSE LOGIC
SERIES 28

PROGRAM TABLE ENTRIES:

CUSTOMER NAME _____

SIGNETICS DEVICE # _____ CF (XXXX)

CUSTOMER SYMBOLIZED PART # _____

AND			OR		
GENERATE	A	Cn	I, P	H	Ns, Fr
PROPAGATE	•		I, $\overline{P}$	L	
TRANSPARENT	—		DON'T CARE	—	
SET	H	Im, Ps	OPTION		
RESET	L		PRESET	H	P/E
NO CHANGE	—		$\overline{OE}$	L	

NO.	C_n	INPUT (Im)																PRESENT STATE (Ps)						REMARKS	OPTION (P/E) NEXT STATE (N)						OUTPUT (Fr)							
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	5	4	3	2	1	0		5	4	3	2	1	0	7	6	5	4	3	2	1	0
0																																					
1																																					
2																																					
3																																					
4																																					
5																																					
6																																					
7																																					
8																																					
9																																					
10																																					
11																																					
12																																					
13																																					
14																																					
15																																					
16																																					
17																																					
18																																					
19																																					
20																																					
21																																					
22																																					
23																																					
24																																					
25																																					
26																																					
27																																					
28																																					
29																																					
30																																					
31																																					
32																																					
33																																					
34																																					
35																																					
36																																					
37																																					
38																																					
39																																					
40																																					
41																																					
42																																					
43																																					
44																																					
45																																					
46																																					
47																																					
PIN NO.		20	21	22	23	24	25	26	27	2	3	4	5	6	7	8	9														10	11	12	13	15	16	17	18
VARIABLE NAME																																					

NOTES

1. The FPLS is shipped with all links initially intact. Thus, a background of "0" for all Terms, and an "H" for the P/E option, exists in the table, shown BLANK instead for clarity.
2. Unused C_n, Im, and Ps bits are normally programmed Don't Care (—).
3. Unused transition and output Terms can be left blank.
4. Letters in variable fields are used as identifiers by logic type programmers.

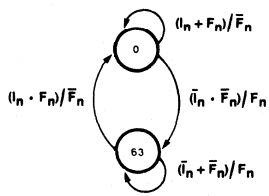
TEST ARRAY

The FPLS may be subjected to AC and DC parametric tests prior to programming via an on chip test array.

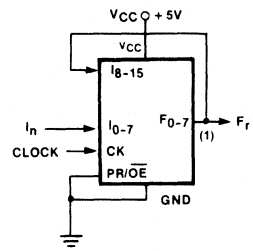
The array consists of test transition terms 48 and 49, factory programmed as shown below.

Testing is accomplished by clocking the FPLS and applying the proper input sequence to I_{0-7} as shown in the test circuit timing diagram.

STATE DIAGRAM



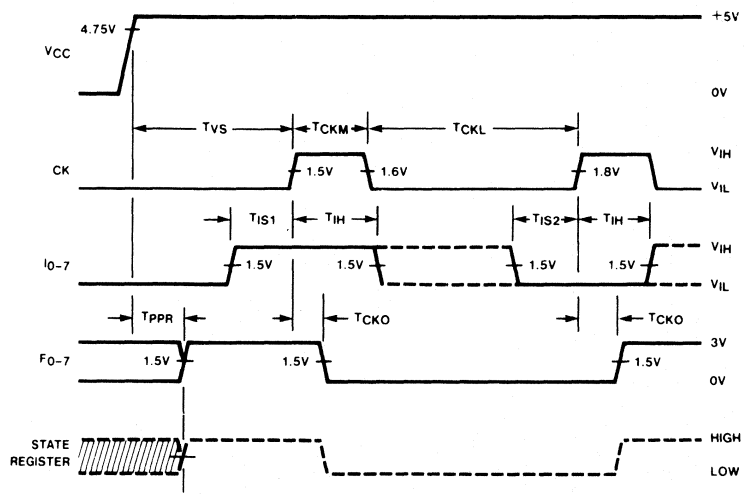
FPLS UNDER TEST



TEST ARRAY PROGRAM

TERM	C	AND: INPUT (Im) 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	PRESENT STATE (Ps) 5	4	3	2	1	0	OPTION (P/E) H · OR · NEXT STATE (Ns) 5	4	3	2	1	0	OUTPUT (Fr) 7	6	5	4	3	2	1	0
48	A	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L
49	●	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H

TEST CIRCUIT TIMING DIAGRAM



Both terms 48 and 49 must be deleted during user programming to avoid interfering with the desired logic function. This is accomplished automatically by any Signetics' qualified programming equipment.

TEST ARRAY DELETED

TERM	C	AND: INPUT (Im) 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	PRESENT STATE (Ps) 5	4	3	2	1	0	OPTION (P/E) H · OR · NEXT STATE (Ns) 5	4	3	2	1	0	OUTPUT (Fr) 7	6	5	4	3	2	1	0
48	—	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	—	—	—	—	—	—	—	—	—	—	—	—	—	—
49	●	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	—	—	—	—	—	—	—	—	—	—	—	—	—	—

VIRGIN DEVICE

The 82S100/101 are shipped in an unprogrammed state, characterized by:

1. All internal Ni-Cr links are intact.
2. Each product term (P-term) contains both true and complement values of every input variable I_n (P-terms always logically "false").
3. The "OR" Matrix contains all 48-P-terms.
4. The polarity of each output is set to active high (F_P function).
5. All outputs are at a low logic level.

RECOMMENDED PROGRAMMING PROCEDURE

To program each of 8 Boolean logic functions of 16 true or complement variables, including up to 48 P-terms, follow the Program/Verify procedures for the "AND" matrix, "OR" matrix, and output polarity outlined below. To maximize recovery from programming errors, leave all links in unused device areas intact.

SET-UP

Terminate all device outputs with a 10K resistor to +5V. Set GND (pin 14) to 0V.

Output Polarity

PROGRAM ACTIVE LOW ($\overline{F_P}$ FUNCTION)

Program output polarity before programming "AND" matrix and "OR" matrix. Program 1 output at the time. (L) links of unused outputs are not required to be fused.

1. Set FE (pin 1) to GND.
2. Set V_{CC} (pin 28) to V_{CCL} (0.4V).
3. Set $\overline{CE}$ (pin 19), and I_0 through I_{15} to V_{IH} (3.0V).
4. Apply V_{OPF} (17.5V) to the appropriate output, and remove after a period PW_P.
5. Repeat step 4 to program other outputs.

VERIFY OUTPUT POLARITY

1. Set FE (pin 1) to GND; set V_{CC} (pin 28) to V_{CCP} (8.5V).
2. Enable the chip by setting $\overline{CE}$ (pin 19) to V_{IL}.
3. Address a non-existent P-term by applying V_{IH} to all inputs I_0 through I_{15}.
4. Verify output polarity by sensing the logic state of outputs F_0 through F_7. All outputs at a high logic level are programmed active low (F_P function), while all outputs at a low logic level are programmed active high ($\overline{F_P}$ function).
5. Return V_{CC} to GND.

"AND" Matrix

PROGRAM INPUT VARIABLE

Program one input at the time and one P-term at the time. All input variable links of unused P-terms are not required to be fused. However, unused input variables must be programmed as Don't Care for all programmed P-terms.

1. Set FE (pin 1) to V_{FEL}, and V_{CC} (pin 28) to V_{CCP}.
2. Disable all device outputs by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.
4. Address the P-term to be programmed (No. 0 through 47) by forcing the corresponding binary code on outputs F_0 through F_5 with F_0 as LSB. Use standard TTL logic levels V_{OHF} and V_{OLF}.
5a. If the P-term contains neither I_0 nor $\overline{I_0}$ (input is a Don't Care), fuse both I_0 and $\overline{I_0}$ links by executing both steps 5b and 5c, before continuing with step 7.
5b. If the P-term contains I_0, set to fuse the $\overline{I_0}$ link by lowering the input voltage at I_0 from V_{IX} to V_{IH}. Execute step 6.
5c. If the P-term contains $\overline{I_0}$, set to fuse the I_0 link by lowering the input voltage at I_0 from V_{IX} to V_{IL}. Execute step 6.
6a. After t_D delay, raise FE (pin 1) from V_{FEL} to V_{FEH}.
6b. After t_D delay, pulse the $\overline{CE}$ input from V_{IH} to V_{IX} for a period PW_P.
6c. After t_D delay, return FE input to V_{FEL}.
7. Disable programmed input by returning I_0 to V_{IX}.
8. Repeat steps 5 through 7 for all other input variables.
9. Repeat steps 4 through 8 for all other P-terms.
10. Remove V_{IX} from all input variables.

VERIFY INPUT VARIABLE

1. Set FE (pin 1) to V_{FEL}; set V_{CC} (pin 28) to V_{CCP}.
2. Enable F_7 output by setting $\overline{CE}$ to V_{IX}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.
4. Address the P-term to be verified (No. 0 through 47) by forcing the corresponding binary code on outputs F_0 through F_5.
5. Interrogate input variable I_0 as follows:
   A. Lower the input voltage at I_0 from V_{IX} to V_{IH}, and sense the logic state of output F_7.
   B. Lower the input voltage at I_0 from V_{IH} to V_{IL}, and sense the logic state output F_7.

The state of I_0 contained in the P-term is determined in accordance with the following truth table:

INTEGRATED FUSE LOGIC
SERIES 28

I_0	F_7	INPUT VARIABLE STATE CONTAINED IN P-TERM
0 1	1 0	$\overline{I_0}$
0 1	0 1	I_0
0 1	1 1	Don't Care
0 1	0 0	(I_0), ($\overline{I_0}$)

Note that 2 tests are required to uniquely determine the state of the input variable contained in the P-term.

6. Disable verified input by returning I_0 to V_{IX}.
7. Repeat steps 5 and 6 for all other input variables.
8. Repeat steps 4 through 7 for all other P-terms.
9. Remove V_{IX} from all input variables.

"OR" MATRIX

PROGRAM PRODUCT TERM

Program one output at the time for one P-term at the time. All P_n links in the "OR" matrix corresponding to unused outputs and unused P-terms are not required to be fused.

1. Set FE (pin 1) to V_{FEL}.
2. Disable the chip by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. After t_D delay, set V_{CC} (pin 28) to V_{CCS}, and inputs I_6 through I_{15} to V_{IH}, V_{IL}, or V_{IX}.
4. Address the P-term to be programmed (No. 0 through 47) by applying the corresponding binary code to input variables I_0 through I_5, with I_0 as LSB.
5a. If the P-term is contained in output function F_0 ($F_0 = 1$ or $\overline{F_0} = 0$), go to step 6, (fusing cycle not required).
5b. If the P-term is **not** contained in output function F_0 ($F_0 = 0$ or $\overline{F_0} = 1$), set to fuse the P_n link by forcing output F_0 to V_{OPF}.
6a. After t_D delay, raise FE (pin 1) from V_{FEL} to V_{FEH}.
6b. After t_D delay, pulse the $\overline{CE}$ input from V_{IH} to V_{IX} for a period PW_P.
6c. After t_D delay, return FE input to V_{FEL}.
6d. After t_D delay, remove V_{OPF} from output F_0.
7. Repeat steps 5 and 6 for all other output functions.
8. Repeat steps 4 through 7 for all other P-terms.
9. Remove V_{CCS} from V_{CC}.

PROGRAMMING SYSTEM SPECIFICATIONS[1]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Rec	Max	
V_{CCS}	V_{CC} supply (program/verify "OR", verify output polarity)[2]	I_{CCS} = 550mA, min, Transient or steady state	8.25	8.5	8.75	V
V_{CCL}	V_{CC} supply (program output polarity)		0	0.4	0.8	V
I_{CCS}	I_{CC} limit (program "OR")	V_{CCS} = +8.5 ± .25V	550		1,000	mA
	Output voltage					V
V_{OPF}	Program output polarity[3]	I_{OPH} = 300 ± 25mA	17.0	17.5	18.0	
V_{OPL}	Idle		0	0.4	0.8	
I_{OPH}	Output current limit (Program output polarity)	V_{OPH} = +17 ± 1V	275	300	325	mA
	Input voltage					V
V_{IH}	High		2.4	3.0	5.5	
V_{IL}	Low		0	0.4	0.8	
	Input current					μA
I_{IH}	High	V_{IH} = +5.5V			50	
I_{IL}	Low	V_{IL} = 0V			-500	
	Forced output voltage					V
V_{OHF}	High		2.4		5.5	
V_{OLF}	Low		0	0.4	0.8	
	Output current					
I_{OHF}	High	V_{OHF} = +5.5V			100	μA
I_{OLF}	Low	V_{OLF} = 0V			-1	mA
V_{IX}	Program enable level		9.5	10	10.5	V
I_{IX1}	Input variables current	V_{IX} = +10V			10	mA
I_{IX2}	$\overline{CE}$ input current	V_{IX} = +10V			10	mA
V_{FEH}	FE supply (program)[3]	I_{FEH} = 300 ± 25mA, Transient or steady state	17.0	17.5	18.0	V
V_{FEL}	FE supply (idle)	I_{FEL} = -1mA, max	1.25	1.5	1.75	V
I_{FEH}	FE supply current limit	V_{FEH} = +17 ± 1V	275	300	325	mA
V_{CCP}	V_{CC} supply (program/verify "AND")	I_{CCP} = 550mA, min, Transient or steady state	4.75	5.0	5.25	V
I_{CCP}	I_{CC} limit (program "AND")	V_{CCP} = +5.0 ± .25V	550		1,000	mA
I_{OPF}	Output current (program)				10	mA
T_R	Output pulse rise time	10% to 90%	10		50	μs
PW_P	$\overline{CE}$ programming pulse widths		10	10	25	μs
t_D	Pulse sequence delay		1		10	μs
T_{PR}	Programming time			60		μs
	Programming duty cycle				25	%
F_L	Fusing attempts per link				1	cycle
V_S	Verify threshold[4]		1.4	1.5	1.6	V

NOTES

1. These are specifications which a Programming System must satisy in order to be qualified by Signetics.
2. Bypass V_{CC} to GND with a 0.01μf capacitor to reduce voltage spikes.
3. Care should be taken to ensure that the voltage is maintained during the entire fusing cycle. The recommended supply is a constant current source clamped at the specified voltage limit.
4. V_S is the sensing threshold of the FPLA output voltage for a programmed link. It normally constitutes the reference voltage applied to a comparator circuit to verify a successful fusing attempt.
5. These are new limits resulting from device improvements, and which supersede, but do not obsolete the performance requirements of previously manufactured programming equipment.

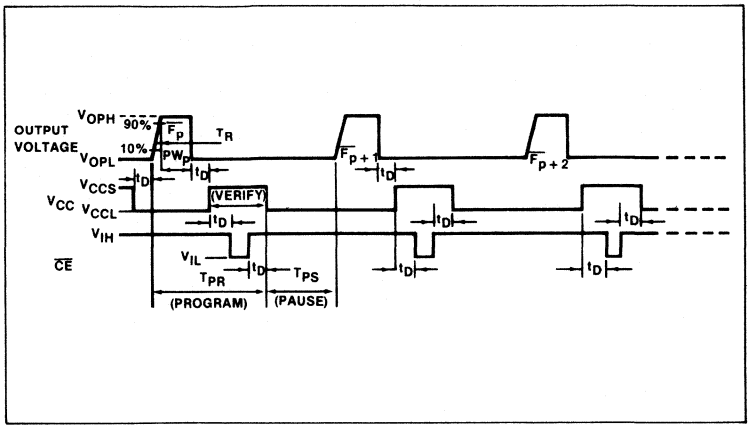
VERIFY PRODUCT TERM

1. Set FE (pin 1) to V_{FEL}.
2. Disable the chip by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. After t_D delay, set V_{CC} (pin 28) to V_{CCS}, and inputs I_0 through I_{15} to V_{IH}, V_{IL}, or V_{IX}.
4. Address the P-term to be verified (No. 0 through 47) by applying the corresponding binary code to input variables I_0 through I_5.
5. After t_D delay, enable the chip by setting $\overline{CE}$ (pin 19) to V_{IL}.
6. To determine the status of the P_n link in the "OR" matrix for each output function F_p or $\overline{F_p}$, sense the state of outputs F_0 through F_7. The status of the link is given by the following truth table:

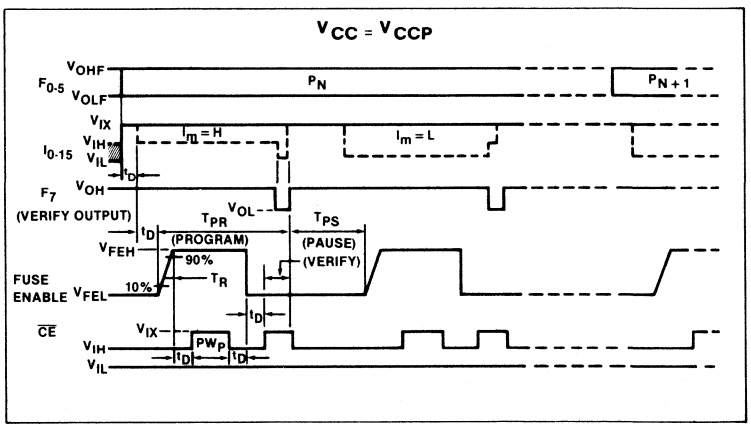
OUTPUT		P-TERM LINK
Active High (F_p)	Active Low ($\overline{F_p}$)	
0	1	Fused
1	0	Present

7. Repeat steps 4 through 6 for all other P-terms.
8. Remove V_{CCS} from V_{CC}.

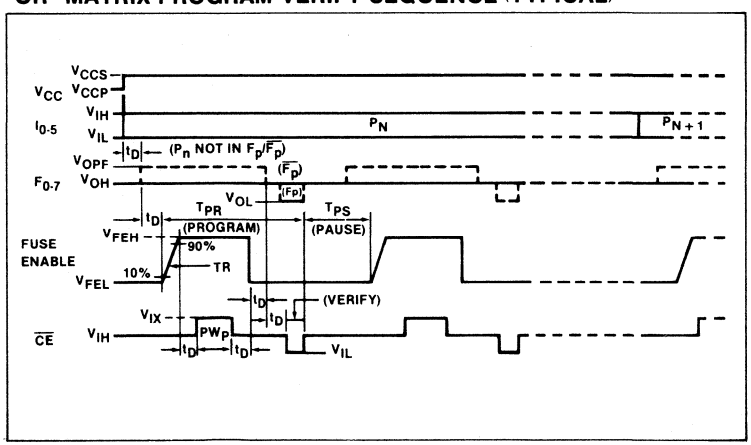
OUTPUT POLARITY PROGRAM-VERIFY SEQUENCE (TYPICAL)



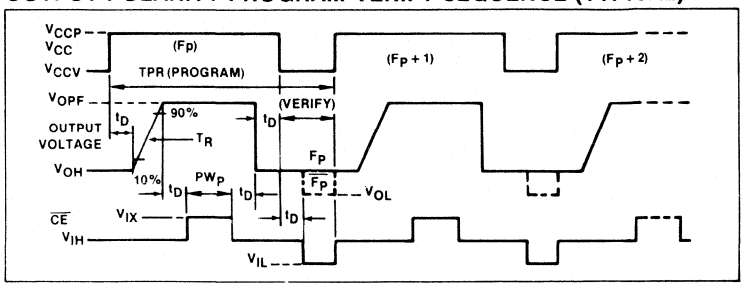
"AND" MATRIX PROGRAM-VERIFY SEQUENCE (TYPICAL)



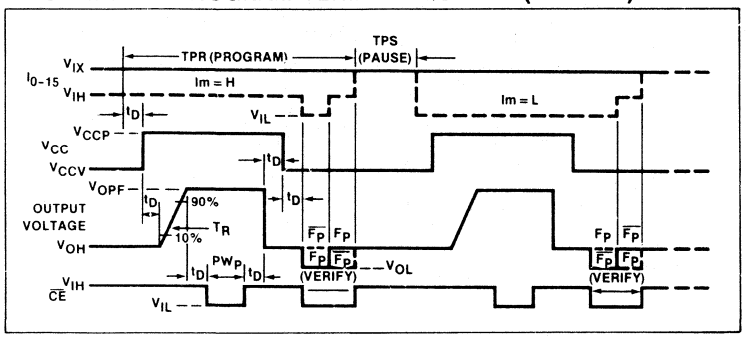
"OR" MATRIX PROGRAM-VERIFY SEQUENCE (TYPICAL)



OUTPUT POLARITY PROGRAM-VERIFY SEQUENCE (TYPICAL)



INPUT MATRIX PROGRAM-VERIFY SEQUENCE (TYPICAL)



VIRGIN DEVICE

The 82S102/103 are shipped in an unprogrammed state, characterized by:

1. All internal Ni-Cr links are intact.
2. Each gate contains both true and complement values of every input variable I_m (logic Null state).
3. The polarity of each output is set to active low ($\overline{F_p}$ function).
4. All outputs are at a high logic level.

RECOMMENDED PROGRAMMING PROCEDURE

To program each of 9 Boolean logic functions of 16 True, Complement, or Don't Care input variables follow the program/verify procedures for the Input Matrix and Output Polarity outlined below. To maximize recovery from programming errors, leave all links of unused gates intact.

SET-UP

Terminate all device outputs with a 10KΩ resistor to +5V.

Output Polarity

PROGRAM ACTIVE HIGH (Fp FUNCTION)

Program output polarity before programming inputs (for convenience). Program one output at a time. (L) links of unused outputs are not required to be fused.

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCV}.
2. Disable all device outputs by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.

A. Raise V_{CC} (pin 28) from V_{CCV} to V_{CCP}.
B. After t_D delay, force output to be programmed to V_{OPF}.
C. After t_D delay, pulse the $\overline{CE}$ input from V_{IH} to V_{IX} for a period PW_P.
D. After t_D delay, remove V_{OPF} voltage source from output being programmed.
E. After t_D delay, return V_{CC} (pin 28) to V_{CCV}, and verify.
F. Repeat steps A through E for any other output.

VERIFY OUTPUT POLARITY

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCV}.
2. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.

A. After t_D delay, set the $\overline{CE}$ input to V_{IL}.
B. Verify output polarity by sensing the logic state of outputs F_0 through F_8. All outputs at a low logic level are programmed active low ($\overline{F_p}$ function), while all outputs at a high logic level are programmed active high (F_p function).

Input Matrix

PROGRAM INPUT VARIABLE

Program one input at a time for one gate at a time. Input variable links of unused gates are not required to be fused. However, unused input variables must be programmed at Don't Care for all used gates.

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCV}.
2. Disable all device outputs by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.

A-1. If a gate contains nether I_0 nor $\overline{I_0}$ (input is a Don't Care), fuse both links by executing both steps A-2 and A-3, before continuing with step C.
A-2. If a gate contains I_0, set to fuse link by lowering the input voltage at I_0 from V_{IX} to V_{IL}. Execute step B.
A-3. If a gate contains $\overline{I_0}$, set to fuse link by lowering the input voltage at I_0 from V_{IX} to V_{IL}. Execute step B.
B-1. After t_D delay, raise V_{CC} from V_{CCV} to V_{CCP}.
B-2. After t_D delay, force output of gate to be programmed to V_{OPF}.
B-3. After t_D delay, pulse the $\overline{CE}$ input from V_{IH} to V_{IL} for a period PW_P.
B-4. After t_D delay, remove V_{OPF} voltage source from output of gate being programmed.
B-5. After t_D delay, return V_{CC} (pin 28) to V_{CCV}, and verify.
C. Disable programmed input by returning I_0 to V_{IX}.
D. Repeat steps A through C for all other input variables.
E. Repeat steps A through D for all other gates to be programmed.
F. Remove V_{IX} from all input variables.

VERIFY INPUT VARIABLE

Unambiguous verification of the logic state programmed for the inputs of each gate requires prior knowledge of its programmed output polarity. Therefore, the output polarity verify procedure must precede input variable verify.

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCV}.
2. Enable all outputs by setting $\overline{CE}$ (pin 19) to V_{IL}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.

A. Interrogate input variable I_0 as follows:
Lower the input voltage to I_0 from V_{IX} to V_{IL}, and sense the logic state of outputs F_{0-8}.

Raise the input voltage to I_0 from V_{IL} to V_{IH} and sense the logic state of outputs F_{0-8}.

The state of I_0 contained in each gate is determined in accordance with the given truth table. Note that 2 tests are required to uniquely determine the state of the input variable contained in each gate.

B. Disable verified input by returning I_0 to V_{IX}.

C. Repeat steps A and B for all other input variables.

D. Remove V_{IX} from all input variables.

TRUTH TABLE FOR INPUT VERIFICATION

I_0	Fp	$\overline{Fp}$	INPUT VARIABLE STATE
0	1	0	$\overline{I_0}$
1	0	1	
0	0	1	I_0
1	1	0	
0	1	0	Don't care
1	1	0	
0	0	1	(I_0), $(\overline{I_0})$
1	0	1	

PROGRAMMING SYSTEMS SPECIFICATIONS[1]

	PARAMETER	TEST CONDITIONS	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT
	V_{CC} supply					V
V_{CCP}	Program[2]	I_{CCP} = 550mA, min Transient or steady state	8.25	8.5	8.75	
V_{CCV}	Verify		4.75	5.0	5.25	
I_{CCP}	I_{CC} limit (program)	V_{CCP} = +8.5 ± .25V, Transient or steady state	550		1,000	mA
V_{OPF}	Forced output voltage[3] (program)	I_{OP} = 300 ± 25mA, Transient or steady state	17.0	17.5	18.0	V
I_{OPF}	Output current (program)	V_{OP} = +17 ± 1V, Transient or steady state	275	300	325	mA
	Input voltage					V
V_{IH}	High		2.4		5.5	
V_{IL}	Low		0	0.4	0.8	
	Input current					μA
I_{IH}	High	V_{IH} = +5.5V			50	
I_{IL}	Low	V_{IL} = 0V			-500	
V_{IX}	$\overline{CE}$ program enable level		9.5	10	10.5	V
I_{IX1}	Input variables current	V_{IX} = +10V			10.0	mA
I_{IX2}	$\overline{CE}$ input current	V_{IX} = +10V			10.0	mA
T_R	Output pulse rise time	10% to 90%	10		50	μs
PW_P	$\overline{CE}$ programming pulse width		10	10	25	μs
t_D	Pulse sequence delay		1		10	μs
T_{PR}	Programming time			60		μs
	Programming duty cycle				100	%
F_L	Fusing attempts per link				1	cycle
V_S	Verify threshold[4]		1.4	1.5	1.6	V

NOTES

1. These are specifications which a Programming System must satisfy in order to be qualified by Signetics.
2. Bypass V_{CC} to GND with a 0.01μF capacitor to reduce voltage spikes.
3. Care should be taken to ensure that the voltage is maintained during the entire fusing cycle. The recommended supply is a constant current source clamped at the specified voltage limit.
4. V_S is the sensing threshold of a gate output voltage for a programmed link. It normally constitutes the reference voltage applied to a comparator circuit to verify a successful fusing attempt.

82S105 PROGRAMMING SEQUENCE

SETUP

1. Set Enable to GND
2. Apply address to Input set
3. Set Strobe to V_{IH} (5.5V)
4. Set P/V to V_{IL} (0V).
5. Wait t_d (1μs), set V_{CC} at V_{CCP} (8.5V).

PROGRAM

1. Wait t_D, raise P/V to V_{IH}
2. Wait t_D, raise Enable to V_{OPF} (17V)
3. Wait t_D, pulse Strobe to V_{IL} for PW_P (10μs)
4. Wait t_D, return Enable to GND
5. Wait t_D, return P/V to V_{IL}

VERIFY[1]

1. Wait t_D, lower Strobe to V_{IL}
2. After PW_V (5μs), read Sense: A V_{IH} level indicates a blown fuse
3. Raise Strobe to V_{IH}

NEXT VARIABLE SELECT

1. After t_D, lower V_{CC} to GND[2]
2. Same term
   2.1. Wait t_D, input new variable address
3. Different term
   3.1. Wait t_D, input new variable address
   3.2. Input new term address
4. Wait t_D, set V_{CC} at V_{CCP}
5. Continue with program or verify sequence.

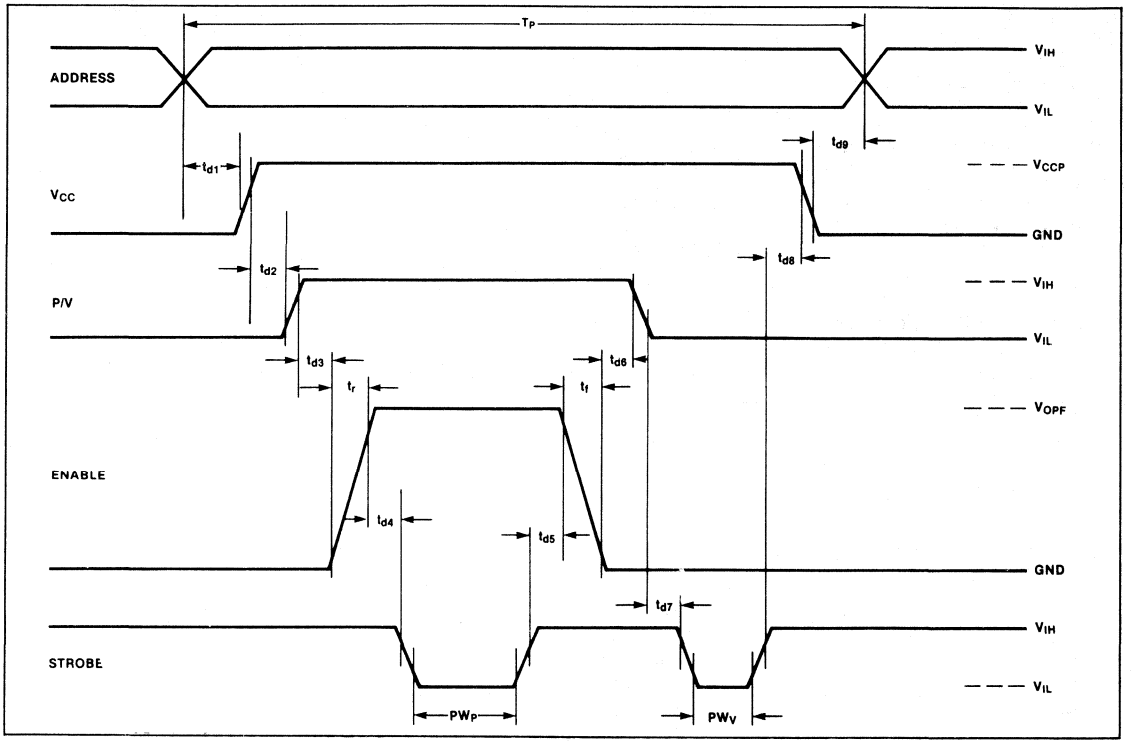
*All references in programming the 82S104/105 also apply to the 82S104A/105A.

PROGRAMMER SPECIFICATION[4, 7]

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Rec	Max	
V_{CCP}	I_{CCP} = 550mA[5]	8.25	8.5	8.75	V
I_{CCP}	V_{CCP} = 8.5V	550		1000	mA
V_{CCP} Duty Cycle				25	%
Input voltage					
V_{IH}		2.4	3.0	5.5	V
V_{IL}		0	0.4	0.8	V
Input current					
I_{IH}	V_{IH} = 5.5V			50	μA
I_{IL}	V_{IL} = 0V			– 500	μA
V_{IX}	I_{IX} = 10mA	9.5	10.0	10.5	V
V_{OPF}	I_{OPF} = 325mA[6]	17.0	17.5	18.0	V
I_{OPF}	V_{OPF} = 17.5V	275		325	mA
PW_V	10-10% Strobe	5.0	5.0	10.0	μs
PW_P	10-10% Strobe	10.0	10.0	25.0	μs
t_{d1}	50% Add CH-90% V_{CC}	10.0	10.0	25.0	μs
t_{d2}	90% V_{CC}-10% P/V	5.0	5.0	10.0	μs
t_{d3}	90% P/V-10% Enable	1.0	1.0	5.0	μs
t_r	10%-90% Enable	17.0	20.0	25.0	μs
t_{d4}	90% Enable-10% Strobe	1.0	1.0	5.0	μs
t_{d5}	10% Strobe-90% Enable	1.0	1.0	10.0	μs
t_f	90%-10% Enable	4.0	4.0		μs
t_{d6}	10% Enable-90% P/V	1.0	1.0	10.0	μs
t_{d7}	10% P/V-90% Strobe	1.0	1.0	10.0	μs
t_{d8}	90%-Strobe-90% V_{CC}	1.0	1.0	10.0	μs
t_{d9}	10% V_{CC}-50% Add CH	1.0	1.0	10.0	μs
Slew rates					
Rising edge					
V_{CC} and Enable				1.0	V/μs
Strobe and P/V				1.0	V/ns
Falling edge					
V_{CC} and Enable				10.0	V/μs
Strobe and P/V				10.0	V/ns
T_P Period/Fuse			60		μs
F_L Attempts/Fuse				1	cycle

NOTES:
1. All fuses may be programmed and then all verified rather than the recommended program-verify-change address cycle.
2. V_{CC} is turned off during each address change.
3. Numbers in parentheses are recommended values.
4. These specifications which a Programming System must satisfy in order to be qualified by Signetics.
5. Bypass V_{CC} to GND with a 0.01μF capacitor to reduce voltage spikes.
6. Care should be taken to ensure that the voltage is maintained during the entire fusing cycle. The recommended supply is a constant current source clamped at the specified voltage limit.
7. These are new limits resulting from device improvements and which supersede, but do not obsolete the performance requirements of previously manufactured programming equipment.

82S105 PROGRAMMING WAVEFORMS



PROGRAM PR/$\overline{OE}$ OPTION

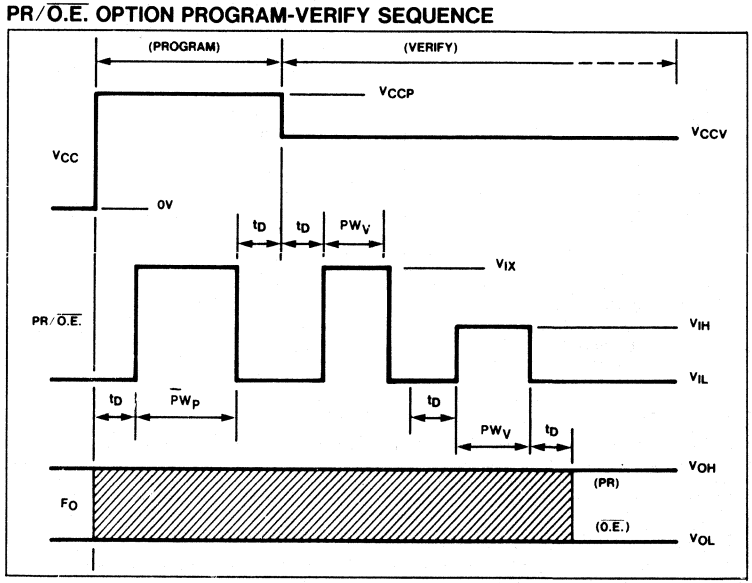
1. With PR/$\overline{OE}$ (pin 19) at GND, raise V_{CC} to V_{CCP} (8.5V).
2. After t_D (10μs) delay, pulse PR/$\overline{OE}$ to V_{IX} (10.0V) for a duration of PW_P (10μs).
3. t_D delay after PR/$\overline{OE}$ has returned to GND, lower V_{CC} to V_{CCV} or GND.

VERIFY PR/$\overline{OE}$ OPTION

1. With PR/OE at GND, set V_{CC} to V_{CCV}.
2. After a delay of t_D raise PR/$\overline{OE}$ to V_{IX} for a minimum duration of PW_V (5μs).
3. Return PR/$\overline{OE}$ to GND with a fall time less than T_f (4μs).
4. After t_D delay, pulse PR/$\overline{OE}$ to V_{IH} (3.0V) for a duration of PW_V (5μs).
5. After t_D delay, F_O (pin 18) indicates V_{OH} if the PR option is selected and V_{OL} if the $\overline{OE}$ option is programmed.

NOTES
1. All outputs will be at "1", as preset by initial power-up procedure.
2. Device can be clocked via test array function.
3. Test array function MUST be deleted before incorporating user program.

PR/$\overline{O.E.}$ OPTION PROGRAM-VERIFY SEQUENCE



PROGRAM CYCLE ROW/COLUMN FUSE ADDRESSING

VARIABLE SELECT TABLE[1]

ROW HEX ADDRESS I_{13} I_{12} I_{11}	ROW HEX ADDRESS I_{10} I_9 I_8 I_7	SELECTED VARIABLE		ROW HEX ADDRESS I_{13} I_{12} I_{11}	ROW HEX ADDRESS I_{10} I_9 I_8 I_7	SELECTED VARIABLE	
0	0	N_0	SET	4	0	AND Array	I_0
0	1		RESET	4	1		$\overline{I_0}$
0	2	N_1	SET	4	2		I_1
0	3		RESET	4	3		$\overline{I_1}$
0	4	N_2	SET	4	4		I_2
0	5		RESET	4	5		$\overline{I_2}$
0	6	N_3	SET	4	6		I_3
0	7		RESET	4	7		$\overline{I_3}$
0	8	N_4	SET	4	8		I_4
0	9		RESET	4	9		$\overline{I_4}$
0	A	N_5	SET	4	A		I_5
0	B		RESET	4	B		$\overline{I_5}$
0	C	F_0	SET	4	C		I_6
0	D		RESET	4	D		$\overline{I_6}$
0	E	F_1	SET	4	E		I_7
0	F		RESET	4	F		$\overline{I_7}$
1	0	F_2	SET	5	0		I_8
1	1		RESET	5	1		$\overline{I_8}$
1	2	F_3	SET	5	2		I_9
1	3		RESET	5	3		$\overline{I_9}$
1	4	F_4	SET	5	4		I_{10}
1	5		RESET	5	5		$\overline{I_{10}}$
1	6	F_5	SET	5	6		I_{11}
1	7		RESET	5	7		$\overline{I_{11}}$
1	8	F_6	SET	5	8		I_{12}
1	9		RESET	5	9		$\overline{I_{12}}$
1	A	F_7	SET	5	A		I_{13}
1	B		RESET	5	B		$\overline{I_{13}}$
1	C	Complement Array	C	5	C		I_{14}
				5	D		$\overline{I_{14}}$
1	D	Empty Address Space		5	E		I_{15}
				5	F		$\overline{I_{15}}$
				6	0		P_0
				6	1		$\overline{P_0}$
				6	2		P_1
				6	3		$\overline{P_1}$
				6	4		P_2
				6	5		$\overline{P_2}$
				6	6		P_3
				6	7		$\overline{P_3}$
				6	8		P_4
				6	9		$\overline{P_4}$
				6	A		P_5
				6	B		$\overline{P_5}$
3	F			6	C	Complement Array	$\overline{C}$

TRANSITION TERM SELECT TABLE[2]

COLUMN HEX ADDRESS I_5 I_4	COLUMN HEX ADDRESS I_3 I_2 I_1 I_0	SELECTED TRANSITION TERM
0	0	0
0	1	1
0	2	2
0	3	3
0	4	4
0	5	5
0	6	6
0	7	7
0	8	8
0	9	9
0	A	10
0	B	11
0	C	12
0	D	13
0	E	14
0	F	15
1	0	16
1	1	17
1	2	18
1	3	19
1	4	20
1	5	21
1	6	22
1	7	23
1	8	24
1	9	25
1	A	26
1	B	27
1	C	28
1	D	29
1	E	30
1	F	31
2	0	32
2	1	33
2	2	34
2	3	35
2	4	36
2	5	37
2	6	38
2	7	39
2	8	40
2	9	41
2	A	42
2	B	43
2	C	44
2	D	45
2	E	46
2	F	47
3	0	Test Col 48
3	1	Test Col 49

NOTES
1. A row address identifies a particular variable coupled to all transition terms.
2. With a variable selected by the row address the column address further selects a coupling fuse for each term.

IFL20 PROGRAMMING SEQUENCE

SETUP

1. Set Enable to GND
2. Apply address to Input set
3. Set Strobe to V_{IH} (5.5V)
4. Set P/V to V_{IL} (0V).
5. Wait t_d (1μs), set V_{CC} at V_{CCP} (8.5V).

PROGRAM

1. Wait t_D, raise P/V to V_{IH}
2. Wait t_D, raise Enable to V_{OPF} (17V)
3. Wait t_D, pulse Strobe to V_{IL} for PW_P (10μs)
4. Wait t_D, return Enable to GND
5. Wait t_D, return P/V to V_{IL}

VERIFY[1]

1. Wait t_D, lower Strobe to V_{IL}
2. After PW_V (5μs), read Sense: A V_{IH} level indicates a blown fuse
3. Raise Strobe to V_{IH}

NEXT VARIABLE SELECT

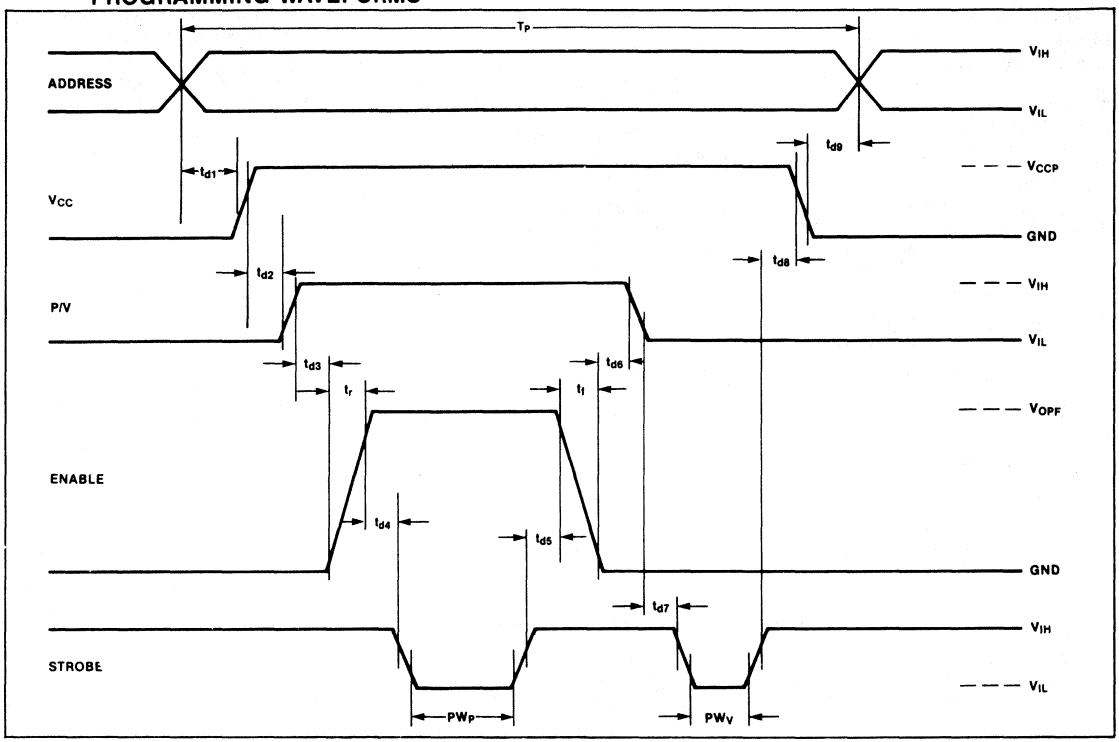
1. After t_D, lower V_{CC} to GND[2]
2. Same term
   2.1. Wait t_D, input new variable address
3. Different term
   3.1. Wait t_D, input new variable address
   3.2. Input new term address
4. Wait t_D, set V_{CC} at V_{CCP}
5. Continue with program or verify sequence.

PROGRAMMER SPECIFICATION[4, 7]

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Rec	Max	
V_{CCP}	I_{CCP} = 550mA[5]	8.25	8.5	8.75	V
I_{CCP}	V_{CCP} = 8.5V	550		1000	mA
V_{CCP} Duty Cycle				25	%
Input voltage					
V_{IH}		2.4	3.0	5.5	V
V_{IL}		0	0	0.8	V
Input current					
I_{IH}	V_{IH} = 5.5V			50	μA
I_{IL}	V_{IL} = 0V			– 500	μA
V_{IX}	I_{IX} = 10mA	9.5	10.0	10.5	V
V_{OPF}	I_{OPF} = 325mA[6]	17.0	17.5	18.0	V
I_{OPF}	V_{OPF} = 17.5V	275		325	mA
PW_V	10-10% Strobe	1	5.0	50	μs
PW_P	10-10% Strobe	10.0	10.0	110	μs
t_{d1}	50% Add CH-90% V_{CC}	1	5	50	μs
t_{d2}	90% V_{CC}-10% P/V	1	5.0	50	μs
t_{d3}	90% P/V-10% Enable	1.0	1.0	50	μs
t_r	10%-90% Enable	17.0	20.0	40	μs
t_{d4}	90% Enable-10% Strobe	1.0	1.0	50	μs
t_{d5}	10% Strobe-90% Enable	1.0	1.0	50	μs
t_f	90%-10% Enable	2	7	20	μs
t_{d6}	10% Enable-90% P/V	1.0	1.0	50	μs
t_{d7}	10% P/V-90% Strobe	1.0	1.0	50	μs
t_{d8}	90%-Strobe-90% V_{CC}	1.0	1.0	50	μs
t_{d9}	10% V_{CC}-50% Add CH	1.0	1.0	50	μs
Slew rates					
Rising edge					
V_{CC} and Enable				1.0	V/μs
Strobe and P/V				1.0	V/ns
Falling edge					
V_{CC} and Enable				10.0	V/μs
Strobe and P/V				10.0	V/ns
T_P Period/Fuse			60		μs
F_L Attempts/Fuse				1	cycle

NOTES:
1. All fuses may be programmed and then all verified rather than the recommended program-verify-change address cycle.
2. V_{CC} is turned off during each address change to achieve a 25% duty cycle whenever any recommended time is exceeded.
3. Numbers in parentheses are recommended values.
4. These specifications which a Programming System must satisfy in order to be qualified by Signetics.
5. Bypass V_{CC} to GND with a 0.01μF capacitor to reduce voltage spikes.
6. Care should be taken to ensure that the voltage is maintained during the entire fusing cycle. The recommended supply is a constant current source clamped at the specified voltage limit.
7. These are new limits resulting from device improvements and which supersede, but do not obsolete the performance requirements of previously manufactured programming equipment.

PROGRAMMING WAVEFORMS



PROGRAM CYCLE ROW/COLUMN FUSE ADDRESSING

VARIABLE SELECT TABLE[1]

ROW HEX ADDRESS		SELECTED VARIABLE	
$I_6I_7B_0$	$B_2B_3B_4B_5$		
0	0	AND Array	$\frac{I_0}{I_0}$
0	1		
0	2		$\frac{I_1}{I_1}$
0	3		
0	4		$\frac{I_2}{I_2}$
0	5		
0	6		$\frac{I_3}{I_3}$
0	7		
0	8		$\frac{I_4}{I_4}$
0	9		
0	A		$\frac{I_5}{I_5}$
0	B		
0	C		$\frac{I_6}{I_6}$
0	D		
0	E		$\frac{I_7}{I_7}$
0	F		
1	0		$\frac{B_9}{B_9}$
1	1		
1	2		$\frac{B_8}{B_8}$
1	3		
1	4		$\frac{B_7}{B_7}$
1	5		
1	6		$\frac{B_6}{B_6}$
1	7		
1	8		$\frac{B_5}{B_5}$
1	9		
1	A		$\frac{B_4}{B_4}$
1	B		
1	C		$\frac{B_3}{B_3}$
1	D		
1	E		$\frac{B_2}{B_2}$
1	F		
2	0		$\frac{B_1}{B_1}$
2	1		
2	2		$\frac{B_0}{B_0}$
2	3		
2 ↓ 2	4 ↓ F	Empty Address Space	

ROW HEX ADDRESS		SELECTED VARIABLE	
$I_6I_7B_0$	$B_2B_3B_4B_5$		
3	0	OR Array	9
3	1		8
3	2		7
3	3		6
3	4		5
3	5		4
3	6		3
3	7		2
3	8		1
3	9		0
3	A	Polarity Enable	
3 ↓ 3	B ↓ F	Empty Address Space	

TERM SELECT TABLE[2]

COLUMN HEX ADDRESS		SELECTED PRODUCT TERM	
I_0I_1	$I_2I_3I_4I_5$		
0	0	Logic Terms	0
0	1		1
0	2		2
0	3		3
0	4		4
0	5		5
0	6		6
0	7		7
0	8		8
0	9		9
0	A		10
0	B		11
0	C		12
0	D		13
0	E		14
0	F		15
1	0		16
1	1		17
1	2		18
1	3		19
1	4		20
1	5		21
1	6		22
1	7		23
1	8		24
1	9		25
1	A		26
1	B		27
1	C		28
1	D		29
1	E		30
1	F		31
2	0	Control Terms	0
2	1		1
2	2		2
2	3		3
2	4		4
2	5		5
2	6		6
2	7		7
2	8		8
2	9		9
2	A	Polarity Terms	0
2	B		1
2	C		2
2	D		3
2	E		4
2	F		5
3	0		6
3	1		1
3	2		8
3	3		9

NOTES
1. A row address identifies a particular variable coupled to all product terms.
2. With a variable selected by the row address the column address further selects a coupling fuse for each term.

PROGRAM CYCLE ROW/COLUMN FUSE ADDRESSING

VARIABLE SELECT TABLE[1]

ROW HEX ADDRESS			SELECTED VARIABLE
$B_1B_2B_3$	$F_0F_1F_2F_3$		
0	0	AND ARRAY	$\overline{B}_3$
0	1		B_3
0	2		$\overline{B}_2$
0	3		B_2
0	4		$\overline{B}_1$
0	5		B_1
0	6		$\overline{B}_0$
0	7		B_0
0	8		$\overline{I}_3$
0	9		I_3
0	A		$\overline{I}_2$
0	B		I_2
0	C		$\overline{I}_1$
0	D		I_1
0	E		$\overline{I}_0$
0	F		I_0
2	0		$\overline{F}_0$
2	1		F_0
2	2		$\overline{F}_1$
2	3		F_1
2	4		$\overline{F}_2$
2	5		F_2
2	6		$\overline{F}_3$
2	7		F_3
2	8		$\overline{F}_4$
2	9		F_4
2	A		$\overline{F}_5$
2	B		F_5
2	C		$\overline{F}_6$
2	D		F_6
2	E		$\overline{F}_7$
2	F		F_7
4	F		C_N

ROW HEX ADDRESS			SELECTED VARIABLE
$B_1B_2B_3$	$F_0F_1F_2F_3$		
5	0	FLIP-FLOP CONTROL TERMS	F_7 J K
5	1		
5	2		F_6 J K
5	3		
5	4		F_5 J K
5	5		
5	6		F_4 J K
5	7		
5	8		F_3 J K
5	9		
5	A		F_2 J K
5	B		
5	C		F_1 J K
5	D		
5	E		F_0 J K
5	F		
6	0	J-K/D OPTION	F_7 F_6 F_5 F_4 F_3 F_2 F_1 F_0
6	1		
6	2		
6	3		
6	4		
6	5		
6	6		
6	7		
7	1	OR ARRAY	B_3 B_2 B_1 B_0
7	2		
7	3		
7	4		
7	5	C-ARRAY	C_N
7	6	OE	EA
7	8	POLARITY TERMS	B_0 B_1 B_2 B_3
7	A		
7	D		
7	F		

TERM SELECT TABLE[2]

COLUMN HEX ADDRESS			SELECTED TERM	
CLK I_0	$I_1I_2I_3B_0$			
0	0	TRANSITION TERM	0	
0	1		1	
0	2		2	
0	3		3	
0	4		4	
0	5		5	
0	6		6	
0	7		7	
0	8		8	
0	9		9	
0	A		10	
0	B		11	
0	C		12	
0	D		13	
0	E		14	
0	F		15	
1	0		16	
1	1		17	
1	2		18	
1	3		19	
1	4		20	
1	5		21	
1	6		22	
1	7		23	
1	8		24	
1	9		25	
1	A		26	
1	B		27	
1	C		28	
1	D		29	
1	E		30	
1	F		31	
2	1	CONTROL TERM		PA
2	2			PB
2	3			RA
2	4			RB
2	5			LA
2	6			LB
2	7			FC
2	8	OUTPUT ENABLE	•	EA
2	9			EB
2	C		A	EA
2	A			EB
2	D		POLARITY	
2	E	DIRECTION CONTROL		D_0
2	F			D_1
3	0			D_2
3	1			D_3
2	0		TEST Col. 2	
3	2		TEST Col. 1	

NOTES
1. A row address identifies a particular variable coupled to all product terms.
2. With a variable selected by the row address the column address further selects a coupling fuse for each term.

MILITARY PRODUCTS

MILITARY PRODUCTS/ PROCESS LEVELS

The Signetics MIL 38510/883 Program is organized to provide a broad selection of processing options, structured around the most commonly requested customer flows. The program is designed to provide our customers:

- Fully compliant 883 flows on all products.
- Standard processing flows to help minimize the need for custom specs.
- Cost savings realized by using standard processing flows in lieu of custom flows.
- Better delivery lead times by minimizing spec negotiation time, plus allows customer to buy product off-the-shelf or in various stages of production rather than waiting for devices started specifically to custom specs.

The following explains the different processing options available to you. Special device marking clearly distinguishes the type of screening performed. Refer to Tables 2, 3, 4 and 5.

JAN QUALIFIED (JB)

JAN Qualified product is designed to give you the optimum in quality and reliability. The JAN processing level is offered as the result of the government's product standardization programs, and is monitored by the Defense Electronic Supply Center (DESC), through the use of industry-wide procedures and specifications.

JAN Qualified products are manufactured, processed and tested in a government certified facility to Mil-M 38510, and appropriate device slash sheet specifications. Design documentation, lot sampling plans, electrical test data and qualification data for each specific part type has been approved by the Defense Electronic Supply Center (DESC) and products appear on the DESC Qualified Products List (QPL-38510).

Group B testing, per Mil-Std-883 Method 5005, is performed on each six weeks of production on each slash sheet for each package type. Group C, per Mil-Std-883 Method 5005, is performed every ninety days for each microcircuit group. Group D testing, per Mil-Std-883 Method 5005, is performed every six months for each package type.

In addition to the common specs used throughout the industry for processing and testing, JAN Qualified products also possess a requirement for a standard marking used throughout the IC industry.

Table 1 MILITARY PACKAGE AVAILABILITY

JAN CASE OUTLINE AND LEAD FINISH	SIGNETICS MILITARY PACKAGE TYPES						
	CAN		DUAL-IN-LINE				
	8-PIN	10-PIN	8-PIN	14-PIN	16-PIN	18-PIN	24-PIN
PB	—	—	FE	—	—	—	—
CB	—	—	—	F	—	—	—
EB	—	—	—	—	F	—	—
JB	—	—	—	—	—	—	F
DB	—	—	—	W	—	—	—
FB	—	—	—	—	W	—	—
ZC	—	—	—	—	—	—	Q
GC	H	—	—	—	—	—	—
IC	—	H	—	—	—	—	—
VB	—	—	—	—	—	I	—

All products listed are also available in Die form.

Table 2 MILITARY SUMMARY

	JB	RB	RC
	Jan Qualified	883B	883C
54	X	X	X
54LS	X	X	X
54S	X	X	X
82	X	X	X
8T	—	X	X
93XX	X	X	X
96XX	—	X	X
Analog	X	X	X
Bipolar Memory	X	X	X
Microprocessor	—	X	X

MIL-STD-883, LEVEL B

Processing to this option is ideal when no JAN slash sheets are released on devices required. Product is processed to Mil-Std-883 Method 5004, and is 100% electrically tested to industry data sheets. Devices are selectively available as custom processed parts with electricals screened to the JAN Slash Sheets.

MIL-STD-883, LEVEL C

If you need a Military temp, range device, but do not require burn in screening performed, our 883C product is ideal. 883C parts are the standard full Mil-Temperature range product to the Signetics data sheet parameters and screened to MIL-STD-883, Class C.

MILITARY GENERIC DATA

Signetics has a new program for those customers who require quality conformance data on their products. This program allows our customers to obtain reliability information without the necessity or running Groups B, C and D inspections for their particular purchase order. It provides for the customer something that has not been readily available before in the semiconductor industry in that all Military Generic Data is controlled and audited by both Government Inspection in the case of JAN data and Signetics Quality Assurance.

Signetics Military Generic Data is compiled by the Military Products Division based on data from 1) JAN quality conformance lots, and 2) Data generated by quality conformance lots run for other reliability programs. Refer to Table 4.

A Military Generic family is defined as consisting of die function and package type families.

Military Generic Data

- Allows our customers to qualify Signetics products based on existing quality conformance data performed at Signetics.
- Allows our customers to reduce costs and improve deliveries.
- Provides assurance that all Signetics die function families and packages meet Mil-M-38510 and customer reliability requirements.
- Provides an attributes summary to the customer backed by lot identity and traceability.

Table 3 MILITARY PRODUCTS PROCESSING MATRIX

PROCESS LEVEL AND MARKING	PRE-CAP VISUAL	BURN IN	FUNCTIONAL TEST	DC/AC @25°C	DC/AC @TEMP	QPL	OFFSHORE
JB JM38510XXXXX	2010, Cond. B	Yes	100%	100%	100%	Yes	No
RB SXXXX883B	2010, Cond.B	Yes	100%	100%	100%	No	Yes
RC SXXXX883C	2010, Cond. B	No	100%	100% dc Sample ac	Sample dc only	No	Yes

Table 4 DEFINITION AND QUALIFYING MANUFACTURING PERIODS FOR GENERIC DATA

QUALIFIED SUB-GROUPS	QUALIFIES	OPTION 1	OPTION 2
A*	Electrical Test		
B	Package—Same package construction and lead finish.	Data selected from devices manufactured within 6 weeks of the manufacturing period on the same production line through final seal.	Data selected from devices manufactured within 24 weeks of manufacturing period.
C	Die/Process—Devices representing the same process families.	Data selected from representative devices from the same microcircuit group and sealed within 12 weeks of the manufacturing period.	Data selected from the representative devices from the same microcircuit group and sealed within 48 weeks of the manufacturing period.
D	Package—Same package construction and lead finish.	Data selected from the devices representing the same package construction and lead finish manufactured within the 24 weeks of manufacturing period. If specific data not available, Option 2 will be supplied	Data selected from the devices representing the same package construction and lead finish manufactured within the 52 weeks of manufacturing period.

NOTE*
Group A is performed on each lot or sublot of Signetics devices.

Table 5 REQUIREMENTS AND SCREENING FLOWS FOR STANDARD PRODUCTS

DESCRIPTION OF REQUIREMENTS AND SCREENS	MIL-M-38510 AND MIL-STD-883 REQUIREMENTS, METHODS AND TEST CONDITIONS	REQUIREMENT	PROCESSING LEVELS			
			CLASS S	JAN QUALIFIED (JB)	883B (RB)	883C (RC)
General Mil-M-38510	The Manufacturer shall establish and implement a Products Assurance Program Plan and provide for a manufacturer survey by the qualifying activity, Para. 3.4.1.1	—	X	X	N/A	N/A
1. Pre-Certification						
A. Product Assurance Program Plan						
B. Manufacturer's Certification						
2. Certification	Received after manufacturer has completed a successful survey, Para. 3.4.1.2	—	X	X	N/A	N/A
3. Device Qualification	Device qualification shall consist of subjecting the desired device to groups A, B, C & D of method 5005 to tightened LTPD, Para. 3.4.1.2	—	X	X	N/A	N/A
4. Traceability	Traceability maintained back to a production lot Para. 3.4.6	—	X	X	X	X
5. Country of Origin	Devices must be manufactured, assembled, and tested within the U.S. or its territories, Para. 3.2.1	—	X	X	N/A	N/A
Screening Per Method 5004 of Mil-Std-883						
6. Internal Visual (Precap)	2010, Cond. A or B	100%	XA	XB	XB	XB
7. Stabilization Bake	1008, Cond. C Min; (24 Hrs @ 150°C)	100%	X	X	X	X
8. Temperature Cycling*	1010, Cond. C; (10 cycles, −65°C to +150°C)	100%	X	X	X	X
*For Class B and C devices thermal shock may be substituted, 1011, Cond. A; (15 cycles, 0° to +100°C)						
9. Constant Acceleration	2001, Cond. E; (30kg in Yl Plane)	100%	X	X	X	X
10. Visual Inspection	There is no test method for this screen; it is intended only for the removal of "Catastrophic Failures" defined as "Missing Leads, Broken Packages or Lids Off."	100%	X	X	X	X
11. Seal (Hermeticity)	1014					
A. Fine	Cond. A or B (5.0 X 10−⁸CC/Sec)	100%	X	X	X	X
B. Gross	Cond. C2 Min.	100%	X	X	X	X
12. Interim Electricals (Pre Burn-In)	Per applicable device specification	100% Optional	100% Read & Record	Slash Sheet	Data Sheet	N/A
13. Burn-In	1015, Cond. as specified (160 hrs. Min. at 125°C)	100%	100% 240 hrs.	X	X	N/A
14. Final Electricals	Per applicable Device Specification	100%	100% Read & Record	Slash Sheet	Data Sheet	Data Sheet
A. Static Tests @ 25°C	Sub Group 1		X	X	X	X
B. Static Tests @ +125°C	Sub Group 2		X	X	X	N/A
C. Static Tests @ −55°C	Sub Group 3		X	X	X	N/A

Table 5 REQUIREMENTS AND SCREENING FLOWS FOR STANDARD PRODUCTS (Continued)

DESCRIPTION OF REQUIREMENTS AND SCREENS	MIL-M-38510 AND MIL-STD-883 REQUIREMENTS, METHODS AND TEST CONDITIONS	REQUIREMENT	PROCESSING LEVELS			
			CLASS S	JAN QUALIFIED (JB)	883B (RB)	883C (RC)
D. Dynamic Test @25°C	Sub Group 4 (for Linear Product Mainly)		X	X	X	X
E. Functional Test @25°C	Sub Group 7		X	X	X	X
F. Switching Test @25°C	Sub Group 9		X	X	X	N/A
15. Percent Defective allowable (PDA)	A PDA of 10% is a normal requirement applied against the static tests @25°C (A-1). This is controlled by the slash sheets for JB products. For RB 10% is standard	10%	10% 3% Funct"l	X	X	N/A
16. Marking	Fungus Inhibiting Paint	100%	As Req'd	JM38510/ XXXX Slash Sheet #	S X X X X 883B	SXXXX 883C
17. X-Ray	2012		100%	N/A	N/A	N/A
18. External Visual	2009	100%	X	X	X	X
Quality Conformance Inspection per Method 5005 of Mil-Std 883						
19. Group A	Electrical Tests-Final Electricals (#14 above) repeated on a sample basis. (Sub Groups 1 thru 12 as specified.)	Each Lot	X	X	X	X
20. Group B	Package functional and constructional related test i.E. package dimensions, resistance to solvents, internal visual & mechanical, bond strength & solderability.	Every 6 week per pkg. group	X	X	Generic Data Available	
21. Group C	Die related tests I.E. 1,000 hr. operating life, temperature cycling, & constand acceleration.	Every 3 months per μcircuit type	X	X	Generic Data Available	
22. Group D	Package related tests I.E. physical dimensions, lead fatigue, thermal shock, temperature cycle, moisture resistance, mechanical shock, vibration variable frequency constant acceleration, & salt atmosphere.	Every 6 months per package type	X	X	Generic Data Available	

LOGIC—5400 SERIES

DEVICE	DESCRIPTION	JM38510 SLASH SHEET	JAN QUALIFIED		MIL-STD 883	
			DIP	FLAT-PACK	DIP	FLAT-PACK
5400	Quad 2-Input NAND Gate	/00104	1	1	F	W
5404	Hex Inverter	/00105	1	1	F	W
5420	Dual 4-Input NAND Gate	/00102	1	1	F	W
5426	Quad 2-Input NAND Gate with o/c	/00805	1	—	F	—
5432	Quad 2-Input OR Gate	/16101	1	1	F	W
5473	Dual J-K Master-Slave Flip-Flop	/00202	1	1	F	W
5474	Dual D-Type Edge Triggered Flip-Flop	/00205	1	1	F	W
5475	Quad Bistable Latch	/01501	1	1	F	W
5476	Dual J-K Master-Slave Flip-Flop	/00204	1	1	F	W
5477	Quad Bistable Latch	—	—	—	—	W
5485	4-Bit Magnitude Comparator	/15001	1	1	F	W
5493	4-Bit Binary Counter	/01302	1	1	F	W
5496	5-Bit Shift Register	/00902	1	1	F	W
54121	Monostable Multivibrator	/01201	1	1	F	W
54123	Retriggerable Monostable Multivibrator	/01203	1	1	F	W
54126	Quad 3-State Buffer	/15302	1	1	F	—
54151	8-Line to 1-Line Mux	/01406	1	1	F	W
54153	Dual 4-Line to 1-Line Mux	/01403	1	1	F	W
54161	Synchronous 4-Bit Binary Counter	/01306	1	1	F	W
54163	Synchronous 4-Bit Binary Counter	/01304	1	1	F	W
54164	8-Bit Parallel-Out Serial Shift Register	/00903	1	—	F	—
54165	Parallel-Load 8-Bit Shift Register	—	•	•	F	W
54174	Hex D-Type Flip-Flop with Clear	/01701	1	1	F	W
54175	Quad D-Type Edge-Triggered Flip-Flop	/01702	1	1	F	W
54180	8-Bit Odd/Even Parity Checker	/01901	1	1	F	W
54190	Synchronous Up/Down Counter (BCD)	—	—	—	•	•
54191	Synchronous Up/Down Counter (Binary)	—	—	—	•	•
54193	Synchronous 4-Bit Binary Up/Down Counter	/01309	1	1	F	W
54279	Quad S-R Latch	—	—	—	F	W
54365A	Hex Buffer w/Common Enable (3-State)	/16301	1	—	F	—
54366A	Hex Buffer w/Common Enable (3-State)	/16302	1	—	F	—
54367A	Hex Buffer, 4-Bit and 2-Bit (3-State)	/16303	1	—	F	—
54368A	Hex Buffer, 4-Bit and 2-Bit (3-State)	/16304	1	—	F	—
9309	Dual 4-Input Multiplexer	/01404	1	1	F	W

NOTE
1 QPLI 2 QPLII • In process

LOGIC—54LS SERIES

DEVICE	DESCRIPTION	JM38510 SLASH SHEET	JAN QUALIFIED		MIL-STD 883		
			DIP	FLAT-PACK	DIP	LCC	FLAT-PACK
54LS00	Quad 2-Input NAND Gate	/30001	1	1	F	—	W
54LS02	Quad 2-Input NOR Gate	/30301	1	1	F	—	W
54LS04	Hex Inverter	/30003	1	1	F	—	W
54LS08	Quad 2-Input AND Gate	/31004	1	1	F	—	W
54LS10	Triple 3-Input NAND Gate	/30005	1	1	F	—	W
54LS14	Hex Schmitt Trigger	/31302	1	1	F	—	W
54LS20	Dual 4-Input NAND Gate	/30007	1	1	F	—	W
54LS28	Quad 2-Input NOR Buffer	—	—	—	F	—	W
54LS32	Quad 2-Input OR Gate	/30501	1	1	F	—	W
54LS37	Quad 2-Input NAND Buffer	/30202	1	1	F	—	W
54LS42	BCD-to-Decimal Decoder	/30703	1	1	F	G	W
54LS73	Dual J-K Master-Slave Flip-Flop	/30101	1	1	F	—	W
54LS74	Dual D-Type Edge-Triggered Flip-Flop	/30102	1	1	F	—	W
54LS75	Quad Bistable Latch	/31601	1	1	F	—	W
54LS83A	4-Bit Binary Full Adder	—	—	—	F	—	W
54LS85	4-Bit Magnitude Comparator Gate	/31101	1	1	F	G	W
54LS86	Quad 2-Input Exclusive-OR Gate	/30502	1	1	F	—	W
54LS90	Decade Counter	/31501	1	1	F	—	W
54LS93	4-Bit Binary Counter	/31502	1	1	F	—	W
54LS95	4-Bit Left-Right Shift Register	/30603	1	1	F	—	W
54LS96	5-Bit Shift Register	/30604	1	1	F	—	W
54LS107	Dual J-K Master-Slave Flip-Flop	/30108	1	1	F	—	W
54LS109	Dual J-K Positive Edge-Triggered Flip-Flop	/30109	1	1	F	—	W
54LS112	Dual J-K Negative Edge-Triggered Flip-Flop	/30103	1	1	F	—	W
54LS113	Dual J-K Negative Edge-Triggered Flip-Flop	/30104	1	1	F	—	W
54LS125	Quad Bus Buffer Gate w/3-State Outputs	/32301	1	1	F	—	W
54LS138	3-to-8 Line Decoder/Demux	/30701	1	1	F	G	W
54LS139	Dual 2-to-4 Line Decoder/Demux	/30702	—	—	F	G	W
54LS151	8-Line to 1-Line Mux	/30901	•	•	•	—	•
54LS153	Dual 4-Line to 1-Line Mux	/30902	1	1	F	G	W
54LS154	4-Line to 16-Line Decoder/Demux	—	—	—	1	—	—
54LS157	Quad 2-Input Data Selector (non-inv.)	—	—	—	F	—	W
54LS161A	Synchronous 4-Bit Binary Counter	/31504	1	1	F	G	W
54LS163A	Synchronous 4-Bit Binary Counter	/31512	1	1	F	G	W
54LS164	8-Bit Parallel-Out Serial Shift Register	/30605	1	1	F	—	W
54LS173	Quad D-Type Flip-Flop (3-State) (8T10)	—	—	—	F	—	W
54LS174	Hex D-Type Flip-Flop with Clear	/30106	1	1	F	G	W
54LS175	Quad D-Type Edge-Triggered Flip-Flop	/30107	1	1	F	G	W
54LS181	4-Bit Arithmetic Logic Unit	/30801	1	—	F	—	W
54LS190	Synchronous Up/Down Counter (BCD)	/31513	1	1	F	—	W
54LS191	Synchronous Up/Down Counter (Binary)	/31509	1	1	F	—	W

NOTE
1 = Level 1 Qualification 2 = Level 2 Qualification • = In process
LCC = Leadless Chip Carrier
JEDEC standard 20-Pin

LOGIC—54LS SERIES (Cont'd)

DEVICE	DESCRIPTION	JM38510 SLASH SHEET	JAN QUALIFIED		MIL-STD 883		
			DIP	FLAT-PACK	DIP	LCC	FLAT-PACK
54LS192	Synchronous Decade Up/Down Counter	/31507	1	1	F	—	W
54LS193	Synchronous 4-Bit Binary Up/Down Counter	/31508	1	1	F	G	W
54LS194A	4-Bit Bidirectional Universal Shift Register	/30601	•	•	F	G	W
54LS195A	4-Bit Parallel-Access Shift Register	/30602	1	1	F	—	W
54LS197	Presettable Binary Counter/Latch (8291)	/32002	1	1	F	—	W
54LS221	Dual Monostable Multivibrator	—	—	—	•	—	•
54LS240	Octal Inverter Buffer 3-State	/32401	•	—	F	G	—
54LS241	Octal Buffer 3-State	/32402	•	—	F	G	—
54LS242	Quad Inverting TCRS 3-State	/32801	•	•	F	—	W
54LS243	Quad TCRS 3-State	/32802	•	•	F	—	W
54LS244	Octal Buffer 3-State	/32403	•	—	F	G	—
54LS245	Octal TCRS 3-State	/32803	—	—	F	G	—
54LS251	Data Selector/Mux with 3-State Outputs	—	—	—	F	G	W
54LS253	Dual 4-Line to 1-Line Data Selector/Mux	—	•	•	F	—	W
54LS257A	Quad 2-Line to 1-Line Data Selector/Mux	/30906	1	1	F	G	W
54LS258A	Quad 2-Line to 1-Line Data Selector/Mux	/30907	1	1	F	—	W
54LS261	2X4 Parallel Binary Multiplier	—	—	—	F	—	W
54LS266	Quad Exclusive-NOR Gate	/30303	1	1	F	—	W
54LS273	Octal D Flip-Flop	/32501	1	•	F	G	—
54LS283	4-Bit Adder	/31202	•	•	F	G	W
54LS290	Decade Counter	/32003	1	1	F	—	W
54LS293	4-Bit Binary Counter	/32004	1	1	F	—	W
54LS295B	4-Bit Right-Shift Left-Shift Register	/30606	1	1	F	—	W
54LS298	Quad 2-Input Mux with Storage	—	—	—	F	—	W
54LS363	Octal Latch for MOS Interface (3-State)	—	—	—	F	G	—
54LS364	Octal Flip-Flop for MOS Interface (3-State)	—	—	—	F	G	—
54LS365A	Hex Buffer w/Common Enable (3-State)	/32201	1	1	F	—	W
54LS367A	Hex Buffer, 4-Bit and 2-Bit (3-State)	/32203	1	1	F	G	W
54LS368A	Hex Buffer, 4-Bit and 2-Bit (3-State)	/32204	1	1	F	—	W
54LS373	Octal Transparent Latch (3-State)	/32502	1	—	F	G	—
54LS374	Octal D Flip-Flop (3-State)	/32503	1	—	F	G	—
54LS375	Quad Latch	/31604	1	1	F	G	W
54LS377	Octal D Flip-Flop Clock Enable	—	—	—	F	—	—
54LS378	Hex D Flip-Flop with Enable	—	—	—	F	—	W
54LS393	Dual Binary Ripple Counter	—	—	—	F	—	W
54LS395A	4-Bit Cascadeable Shift Register (3-State)	/30607	1	1	F	—	W
54LS490	Dual Decade Ripple Counter	/32703	1	1	F	—	W
54LS670	4X4 Register File (3-State)	/31901	•	•	F	G	W

NOTE
1 = Level 1 Qualification 2 = Level 2 Qualification • = In process
LCC = Leadless Chip Carrier JEDEC standard 20-Pin

LOGIC—8T INTERFACE SERIES

DEVICE	DESCRIPTION	MIL-STD 883	
		DIP	FLAT PACK
8T05	7-Segment Decoder Display Driver (Active-Hi Outputs)	F	W
8T09	Quad Bus Driver with 3-State Outputs	F	W
8T13	Dual Line Driver	F	W
8T16	Dual Communications EIA/MIL Receiver with Hysteresis	F	—
8T18	Dual 2-Input NAND (High Voltage to TTL Interface)	F	W
8T26A	Quad Bus Driver/Receiver (3-State Outputs)	F	R
8T28	Quad Non-Inverting Bus Driver/Receiver (3-State Outputs)	F	W
8T30	Dual TTL/DTL to MOS Transceiver/Port Controller	F	—
8T80	Quad 2-Input NAND Gate (High Voltage)	F	W
8T90	Hex Inverter (High Voltage)	F	W
8T97	High Speed Hex Buffers/Inverters (74367/DM8097)	F	R
8T98	High Speed Hex Buffers/Inverters (74368/DM8098)	F	R
8T126	Quad 3-State Transceivers	F	W
8T127	Quad 3-State Transceivers	F	W
8T128	Quad 3-State Transceivers	F	W
8T129	Quad 3-State Transceivers	F	W

LOGIC—82XX SERIES

DEVICE	DESCRIPTION	MIL-STD 883	
		DIP	FLAT PACK
8202	Buffer Register	F	W
8250	Binary-to-Octal Decoder	F	W
8266	2-input 4-Bit Digital Multiplexer	F	W
8273	10-Bit Serial-In Parallel-Out Shift Register	F	W
8280	BCD Decade Counter/Storage Element	F	W
8281	4-Bit Binary Counter/Storage Element	F	W

LOGIC—54S SERIES

DEVICE	DESCRIPTION	JM38510 SLASH SHEET	JAN QUALIFIED		MIL-STD 883		
			DIP	FLAT-PACK	DIP	LCC	FLAT-PACK
54S00	Quad 2-Input NAND Gate	/07001	1	1	F	—	W
54S02	Quad 2-Input NOR Gate	/07301	1	1	F	—	W
54S04	Hex Inverter	/07003	1	1	F	—	W
54S08	Quad 2-Input AND Gate	/08003	1	1	F	—	W
54S10	Triple 3-Input NAND Gate	/07005	1	1	F	—	W
54S11	Triple 3-Input NAND Gate	/08001	1	1	F	—	W
54S20	Dual 4-Input NAND Gate	—	—	—	F	—	W
54S40	Dual 4-Input NAND Buffer	/07201	1	1	F	—	W
54S51	Dual 2-Wide 2-Input A01 Gate	/07401	1	1	F	—	W
54S74	Dual D-Type Edge-Triggered Flip-Flop	/07101	1	1	F	—	W
54S85	4-Bit Magnitude Comparator	/08201	1	—	F	—	—
54S86	Quad 2-Input Exclusive-OR Gate	/07501	1	1	F	G	W
54S112	Dual J-K Negative Edge-Triggered Flip-Flop	/07102	1	1	F	—	W
54S113	Dual J-K Negative Edge-Triggered Flip-Flop	/07103	1	1	F	—	W
54S133	13-Input NAND Gate	/07009	1	1	F	—	W
54S135	Quad Exclusive-OR/NOR Gate	—	—	—	F	—	—
54S138	3-to-8 Line Decoder/Demux	—	—	—	—	—	—
54S139	Dual 2-to-4 Line Decoder/Demux	/07702	—	—	F	—	W
54S140	Dual 4-Input NAND Line Driver	/08101	1	1	F	—	W
54S151	8-Line to 1-Line Mux	/07901	1	1	F	G	W
54S153	Dual 4-Line to 1-Line Mux	/07902	1	1	F	G	W
54S157	Quad 2-Input Data Selector (non-inv.)	/07903	1	1	F	G	W
54S158	Quad 2-Input Data Selector (inv.)	/07904	1	1	F	G	W
54S174	Hex D-Type Flip-Flop with Clear	—	—	—	F	G	W
54S181	4-Bit Arithmetic Logic Unit	/07801	1	—	F	—	—
54S253	Dual 4-Line to 1-Line Data Selector/Mux	—	—	—	F	—	W
54S260	Dual 5-Input NOR Gate	—	—	—	F	G	W
54S273	Octal D-Type Flip-Flop with Reset	—	—	—	F	—	—
54S258	Quad 2-Line to 1-Line Data Selector/Mux	—	—	—	—	—	—
54S280	9-Bit Odd/Even Parity Generator/Checker	—	—	—	—	—	—
54S374	Octal D-Type Flip-Flop (3-State)	—	—	—	F	G	—

NOTE
1 = Level 1 Qualification 2 = Level 2 Qualification • In process
LCC = Leadless Chip Carrier
JEDEC standard 20-Pin

BIPOLAR MEMORY

DEVICE	ORGANIZATION	PACKAGE		OUTPUT CIRCUIT	NUMBER OF PINS
PROMs					
82S23	32 x 8	F	R	OC	16
82S115	512 x 8	I	R	TS	24
82S123	32 x 8	F	W	TS	16
82S126	256 x 4	F	W	OC	16
82S129	256 x 4	F	R	TS	16
82S130	512 x 4	F	R	OC	16
82S131	512 x 4	F	R	TS	16
82S137	1024 x 4	F	R	TS	18
82S137A	1024 x 4	F	R	TS	18
82S141	512 x 8	F	R	TS	24
82S147	512 x 8	F	R	TS	20
82S181	1024 x 8	F	R,G	TS	24
82S181A	1024 x 8	F	R,G	TS	24
82LS181	1024 x 8	F	R	TS	24
82S185	2048 x 4	I	R	TS	18
82S185A	2048 x 4	I	R	TS	18
82S2708	1024 x 8	F	R	TS	24
82S191	2048 x 8	F,I	R,G	TS	24
82S191A	2048 x 8	F,I	R,G	TS	24
82HS195	4096 x 4	I	R	TS	20
82S321	4096 x 8	I	R	TS	24
FPLF					
82S100	16 x 48 x 8	F,I	R,G	TS	28
82S101	16 x 48 x 8	F,I	R,G	OC	28
82S102	16 x 9	F,I	R	OC	28
82S103	16 x 9	F,I	R	TS	28
82S104	16 x 48 x 8	F,I	R	OC	28
82S105	16 x 48 x 8	F,I	R	TS	28
82S152	18 x 32 x 10	F	R,G	OC	20
82S153	18 x 32 x 10	F	R,G	TS	20
82S157	16 x 32 x 12	F	R,G	OC	20
82S158	16 x 32 x 12	F	R,G	TS	20
RAMs					
54S189	16 x 4	F	R	TS	16
54S301	256 x 1	F	R	OC	16
82S09	64 x 9	I	R	OC	28
82S19	64 x 9	I	R	OC	28
82S16	256 x 1	F	R	TS	16
82LS16	256 x 1	F	W	TS	16
82LS17	256 x 1	F	W	OC	16
54LS301	256 x 1	F	W	OC	16
82S210	256 x 9	F	R	TS	24
82S212	256 x 9	F	R	TS	22
8X350	256 x 8	F	R	TS	22

BIPOLAR MEMORY — JAN

JAN M-38510			
DEVICE	SLASH SHEET	PKG	QUAL STATUS
82S23	/20701	F	QPL I
82S123	/20702	F	QPL I
82S126	/20301	F	QPL I
82S129	/20302	F	QPL I
82S130	/20401	F	QPL I
82S131	/20402	F	QPL I
82S137	/20602	F	QPL I
82S141	/20802	F	QPL I
82S115	/20803	F	In Process
82S181	/20904	F	QPL I
82S185	/20902	I	QPL I
82S2708	/20905	I	In Process
82S191	/21002	I	QPL I
82S100	/1501	I	In Process

Per QPL M38510-52 dated 11 Feb '82.

LINEAR DEVICES

DEVICE	DESCRIPTION	PACKAGE*	
		DIP	CAN
	OPERATIONAL AMPLIFIERS		
LF155	JFET Op Amp		H
LF156	JFET Op Amp		H
LH2101A	Dual Op Amp	F	
LM101/A	Hi Perf Op Amp	F	H
LM124	Quad Op Amp	F	
LM158	Dual Op Amp		H
MC1556	Hi Perf Op Amp	F	H
MC1558	Dual Op Amp	F	H
SE530	Hi Slew Op Amp	F	H
SE532	Dual Op Amp		H
SE5512	Dual Op Amp	FE	H
SE5532	Dual Op Amp	F,FE	H
SE5532A	Dual Op Amp	FE	
SE5534	Lo Noise Op Amp	F,FE	H
SE5534A	Lo Noise Op Amp	FE	H
SE5537	Sample and Hold Amp	FE	H
SE5539	High Freq Op Amp	F	—
μA747	Dual Op Amp	F	H
	COMPARATORS		
SE521	Dual Differential Comparator	F	
SE522	Dual Differential Comparator	F	
SE527	Voltage Comparator	F	H
SE529	Voltage Comparator	F	H
LH2111	Dual Voltage Comparator	F	
LM111	Voltage Comparator	F	H
LM139/A	Quad Voltage Comparator	F	
LM193/A	Dual Voltage Comparator		H
	DIFFERENTIAL AMPLIFIERS		
SE510	Dual Differential Amplifier	F	
SE511	Dual Differential Amplifier	F	
μA733	Video Amplifier	F	H
	PHASE LOCKED LOOPS		
SE567	Tone Decoder PLL	F	H
SE564	Phase Locked Loop	F	H
	TIMERS		
SE555	Timer	F,FE	H
SE556-1	Dual Timer	F	
SE558	Quad Timer	F	

DEVICE	DESCRIPTION	PACKAGE*	
		DIP/ CAN	CAN
	VOLTAGE REGULATORS		
SE5553	Dual Track Reg	F	H
SE5554	Dual Track Reg	F	H
μA723	Adj Volt Reg	F	H

DEVICE	DESCRIPTION	PACKAGE	
		DIP	CAN
	D to A CONVERTERS		
DAC-08	8-Bit Mult DAC	F,Q	H
MC1508-8	8-Bit Mult DAC	F	—
SE5008	8-Bit Mult DAC	F	—
SE5009	8-Bit Mult DAC	F	—
SE5018	8-Bit μP-Comp DAC	F	—
SE5019	8-Bit μP-Comp DAC	F	—
SE5118	8-Bit μP-Comp DAC	F	—
SE5119	8-Bit μP-Comp DAC	F	—

DEVICE	DESCRIPTION	PACKAGE	
		DIP	CAN
	DUAL LINE RECEIVERS		
DS7820/A	Dual Line Receiver	F	—
DS7830/A	Dual Diff Line Driver	F	—

DEVICE	DESCRIPTION	PACKAGE	
		DIP	CAN
	MOS FET SWITCH		
SD210	Switch N-Channel Enhance	EE	—
SD211	Switch N-Channel Enhance	EE	—
SD5002	Quad Analog Switch	I	

DEVICE	DESCRIPTION	PACKAGE	
		DIP	CAN
	SMPS CONTROL CIRCUITS		
SE5560	SMPS Controller	F	—
SG1524	Reg Pulse Width Mod	F	—

JAN M – 38510			
DEVICE	SLASH SHEET	PKG	QUAL STATUS
SE555	10901BGC	H	QPL I
SE556-1	10902BCB	F	QPL 1
LH2101A	10105BEB	F	QPL 1
LM101A	10103BCB	F	QPL 1
LM101A	10103BPB	FE	QPL 1
LM101AH	10103BGC	H	QPL 1
μA74 i	10101BGC	H	QPL 1
μA747	10102BGC	H	QPL 1
DAC-08A	11302BCB	F	IN PROCESS

NOTES
F = Cerdip
H = TO-5

MILITARY PRODUCTS LINEAR INDUSTRY CROSS REFERENCE

LINEAR INDUSTRY CROSS REFERENCE

FAIRCHILD	SIGNETICS
μA111	LM111
μA139	LM139
μA733	μA733
μAF155/156	LF155/156
μA101	LM101
μA101A	LM101A
MC1556	MC1556
μA1558	MC1558
μA747	μA747
MC1555	SE555
μA556	SE556
μA109	LM109
μA79XX	79XX(7)
μA723	μA723

MOTOROLA	SIGNETICS
MLM111	LM111
MC1733	μA733
LF155/56	LF155/156
MLM101	LM101
MLM101A	LM101A
MC1558	MC1558
MC1747	μA747
MC3556	SE556
MLM109	LM109
MC78XX	78XX(7)
MC79XX	79XX(7)
MC1723	μA723
MC1508	MC1508-8

NATIONAL	SIGNETICS
LM161	SE527
LH2111	LH2111
LM111	LM111
LM119	LM119
LM139	LM139
LM193	LM193/193A
LM733	μA733
LF155/56	LF155/156
LH2101A	LH2101A
LH2108A	LH2108A
LM101A	LM101
LM101	LM101A
LM124	LM124
LM158	LM158
LM1558	MC1558
LM1581	SE532
LM747	μA747
LM567	SE567
DM7820	DM7820
DM7830	DM7830
LM555	SE555
LM109	LM109
LM723	μA723

PMI	SIGNETICS
SSS1508	MC1508-8
DAC-08	SE5008

RAYTHEON	SIGNETICS
LM111	LM111
LM139	LM139
RM733	μA733
LF155/56/57	LF155/156
LM101	LM101
LM101A	LM101A
LM124	LM124
RM1556	MC1556
RM1558	MC1558
RM747	μA747
RM555	SE555
LM109	LM109
RM723	μA723

T.I.	SIGNETICS
LM111	LM111
SN52733	μA733
LF155/56	LF155/156
SN52101A	LM101A
SN55182	DM7820
SN55183	DM7830
SN52555	SE555
SE556	SE556
LM109	LM109
μA79XX	μA79XX(7)
SN52723	μA723

PACKAGE OUTLINES

PACKAGES—GENERAL INFORMATION

INTRODUCTION

The following information applies to all packages unless otherwise specified on individual package outline drawings.

General

1. Dimensions shown are metric units (millimeters), except those in parentheses which are English units (inches).
2. Lead spacing shall be measured within this zone.
   a. Shoulder and lead tip dimensions are to centerline of leads.
3. Tolerances non-cumulative
4. Thermal resistance values are determined by utilizing the linear temperature dependence of the forward voltage drop across the substrate diode in a digital device to monitor the junction temperature rise during known power application across V_{CC} and ground. The values are based upon 120 mils square die for plastic packages and a 90 mils square die in the smallest available cavity for hermetic packages. All units were solder mounted to P.C. boards, with standard stand-off, for measurement.

Plastic Only

5. Lead material: Alloy 42 (Nickel/Iron Alloy) Olin 194 (Copper Alloy) or equivalents, solder dipped.
6. Body material: Plastic (Epoxy)
7. Round hole in top corner denotes lead No. 1.
8. Body dimensions do not include molding flash.
9. SO Packages-microminature packages.
   a. Lead material: Alloy-42.
   b. Body material: Plastic (Epoxy).

Hermetic Only

10. Lead material
    a. ASTM alloy F-15 (KOVAR) or equivalent—gold plated, tin plated, or solder dipped.
    b. ASTM alloy F-30 (Alloy 42) or equivalent—tin plated, gold plated or solder dipped.
    c. ASTM alloy F-15 (KOVAR) or equivalent—gold plated.
11. Body Material
    a. Eyelet, ASTM alloy F-15 or equivalent—gold or tin plated, glass body.
    b. Ceramic with glass seal at leads.
    c. BeO ceramic with glass seal at leads.
    d. Ceramic with ASTM alloy F-30 or equivalent.
12. Lid Material
    a. Nickel or tin plated nickel, weld seal.
    b. Ceramic, glass seal.
    c. ASTM alloy F-15 or equivalent, gold plated, alloy seal.
    d. BeO Ceramic with glass seal.
13. Signetics symbol, angle cut, or lead tab denotes Lead No. 1.
14. Recommended minimum offset before lead bend.
15. Maximum glass climb .010 inches.
16. Maximum glass climb or lid skew is .010 inches.
17. Typical four places.
18. Dimension also applies to seating plane.

PLASTIC PACKAGES

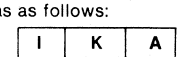
NO. OF LEADS	PACKAGE CODE	Θ_{ja}/Θ_{jc}(°C/W)	DESCRIPTION[1]
Standard Dual-In-Line			
8	NE	162/65	
14	NH	150/65	TO-116/MO-001
16	NJ	137/53	MO-001
18	NK	135/53	
20	NL	135/53	
22	NM	120/53	
24	NN	116/53	MO-015
24	NNE NNF	120/60	Slim Line
28	NQ	116/53	MO-015
40	NW	110/50	MO-015
Power Dual-In-Line[2]			
14	NHA	95/33	Butterfly
16	NJA	95/33	Butterfly
18	NKA	90/26	Butterfly
20	NLA	90/26	Butterfly
24	NNA	60/23	Heatsink
28	NQA	56/21	Heatsink
SO Packages			
8	DE	110	SO-8
14	DH	100	SO-14
16	DJ	100	SO-16

NOTES
1. Dual-in-Line packages unless otherwise described.
2. Package outline is the same as corresponding standard Dual-in-Line package with identical number of leads.

HERMETIC PACKAGES

NO. OF LEADS	PACKAGE CODE	Θ_{ja}/Θ_{jc}(°C/W)	DESCRIPTION[1]
Metal Headers			
3	HBA	100/20	TO-5 Header
4	EC	100/20	TO-46 Header
4	EE	150/25	TO-72 Header
8	HEA/HEB	150/25	TO-5 Header
10	HFB/HFA	150/25	TO-5/TO-100 Header, Short Can
10	HFD/HFC	150/25	TO-5/TO-100 Header, Tall Can
Flat Packs			
10	QF	230/55	Flat Ceramic
10	WF	240/50	Flat Ceramic
14	QHA	185/45	Flat Ceramic Laminate
14	WH	205/50	Flat Ceramic
16	QJA	170/45	Flat Ceramic Laminate
16	RJA	133/30	Flat Ceramic, BeO
16	WJ	200/50	Flat Ceramic
18	RKA	107/22	Flat Ceramic, BeO
24	QNA	155/44	Flat Ceramic Laminate
24	RNA	107/22	Flat Ceramic, BeO
24	WN	155/40	Flat Ceramic
28	RQA	107/22	Flat Ceramic, BeO
40	RWA	95/20	Flat Ceramic, BeO
Cerdip Family			
8	FE	110/30	Dual-in-Line Ceramic
14	FH	110/30	Dual-in-Line Ceramic
16	FJ	100/30	Dual-in-Line Ceramic
18	FK	93/27	Dual-in-Line Ceramic
20	FL	90/25	Dual-in-Line Ceramic
22	FM	75/27	Dual-in-Line Ceramic
24	FN	60/26	Dual-in-Line Ceramic
28	FQ	57/27	Dual-in-Line Ceramic
Laminated Ceramic, Side Brazed Lead			
8	IEA	100/30	Dip Laminate
14	IHA	95/25	Dip Laminate
16	IJA	90/25	Dip Laminate
18	IKA/IKB	88/25	Dip Laminate
22	IMA	80/25	Dip Laminate
24	INC/INH	65/25	Dip Laminate
28	IQA	60/25	Dip Laminate
28	GQ*	90/35	Chip Carrier
40	IWA	55/25	Dip Laminate
44	GX*	75/30	Chip Carrier
48	JY*	55/25	Dual-in-Line Ceramic
50	IZA	42/20	Dip Laminate

Memory Package Codes consist of two or three alphas as follows:



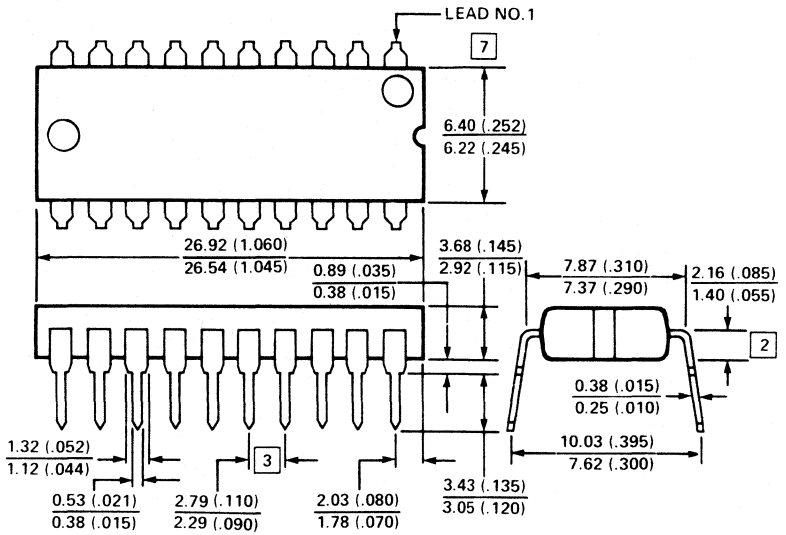
Package Type
D = Small Outline Plastic DIP
F = Cerdip
G = Hermetic Leadless
I = Ceramic Side-Brazed DIP
N = Plastic DIP
R = Ceramic Flat Pack
W = Cerpack Flat Pack

Revision or other Signetics internal designator

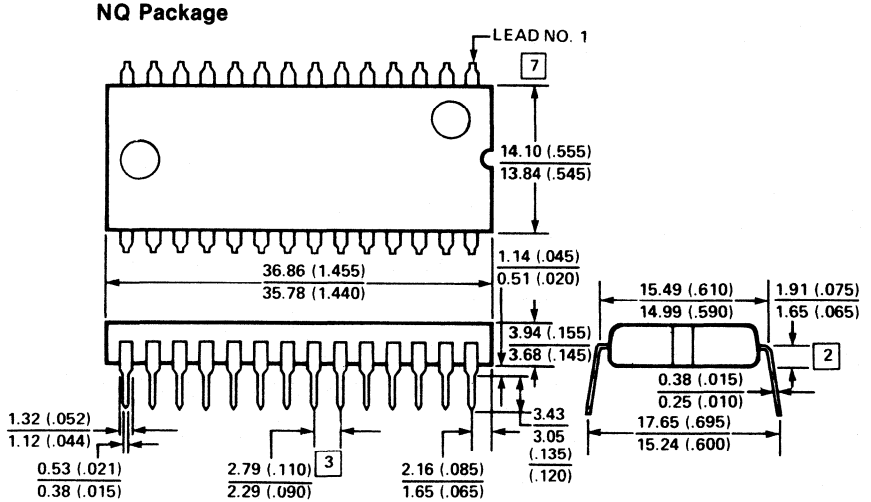
Number of Pins
J = 16
K = 18
L = 20
M = 22
N = 24
Q = 28

PLASTIC: Standard and Power Dual-In-Line

NL Package

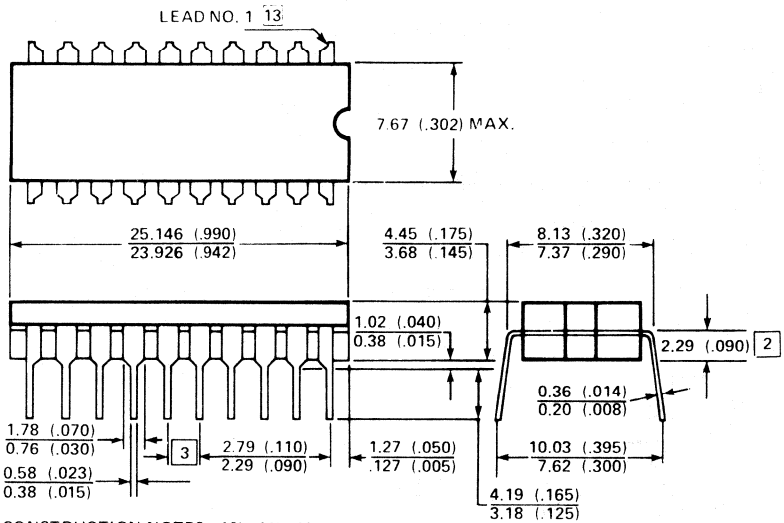


NQ Package



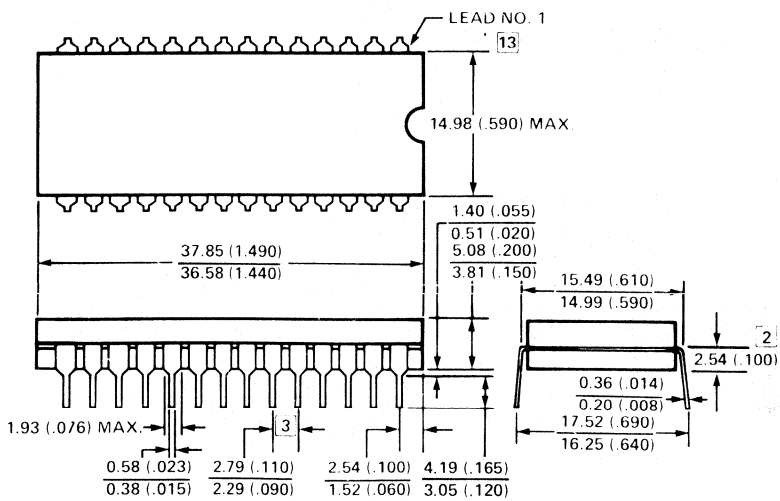
HERMETIC: Cerdip Family

FL Package



CONSTRUCTION NOTES: 10b, 11b, 12b

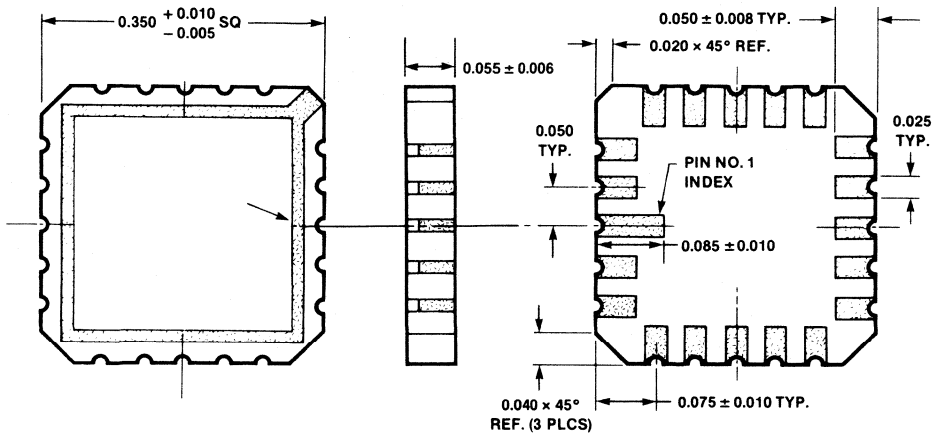
FQ Package



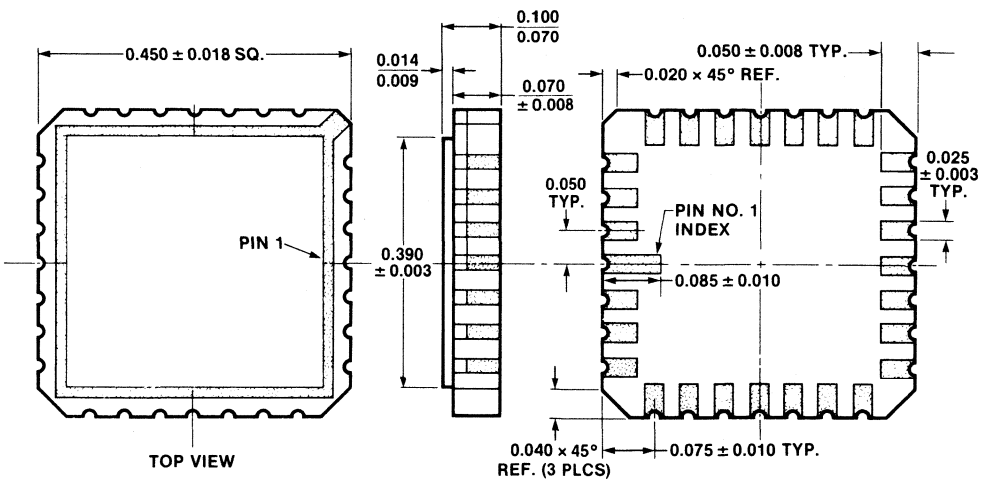
CONSTRUCTION NOTES: 10b, 11b, 12b

Leadless Chip Carriers

GL



GQ



Applications section IFL handbook

Note: This section contains most of the IFL applications data and technical articles which have been published in the past. The sole purpose of this applications section is to *guide* the engineer as to how problems could be solved using Integrated Fuse Logic. The reader should realize that some data is not correct anymore in the sense that formats have been changed and new IFL types have been developed, making more cost effective solutions possible.

CONTENTS

New developments in integrated fuse logic 163

LOGMIN users manual 177

Field-programmable arrays
part 1: powerful alternatives to random logic 181
part 2: sequencers and arrays transform truth tables into working systems 187

Controllers using FPLAs and chapter contents 195

Single-chip multiprocessor arbiter using the Signetics 82S105 FPLS 223

Expandable memory system using the Signetics field programmable gate array 231

8-bit parallel CRC generator/checker using field programmable logic 235

A simple keyboard encoder using the FPLS 239

The FPLA adds display flexibility to CRT controllers 245

Field programmable logic arrays application manual 251
Table of contents (chapter) 252

Programming services bipolar memory/LSI products 361

New developments in integrated fuse logic

Custom logic is expensive – too expensive if your production run is short. 'Random logic' is cheaper but occupies more sockets and board space. Integrated Fuse Logic bridges the gap. Using IFL, you can configure an off-the-shelf chip to perform just the logic functions you need. Design and development times are much shorter, and risk much lower, than for custom logic. Connections are fewer than for random logic and, for all but the simplest functions, propagation delay is usually shorter. Yet another advantage that IFL has over custom logic is that it allows you to redesign the functions without redesigning the chip – giving you an invaluable margin not only for cut-and-try during system development but also for later revision of system design. You're not tied down by the need to recover capital invested in a custom chip.

An IFL chip is an array of logic elements – gates, inverters, and flip-flops, for instance. In the virgin state everything is connected to everything else by nichrome fuses and, although the chip has the capacity to perform an extensive variety of logic functions, it doesn't have the ability to. What gives it that is programming: selectively blowing undesired fuses so that those that remain provide the interconnections necessary for the required functions.

Signetics Series 20 IFL, named for the number of pins, supplements the well-known Series 28. The package is smaller – little more than a third the size, in fact – but the improved architecture, with user-programmable shared I/O, compensates for the fewer pins. The series comprises the following members, in order of increasing complexity:

- 82S150/82S151 – field-programmable gate array
- 82S152/82S153 – field-programmable logic array
- 82S154/82S159 – field-programmable logic sequencer

The even-numbered types have open-collector, and the odd-numbered types three-state outputs.

Entry to all the devices is via a product matrix, an array of input and shared I/O lines fuse-connected to the multiple inputs of an array of AND gates (see Fig. 1, 2 and 6). To exploit the capacity of any device it is important to make the most economical use of the AND gates it has available. Application of the Morgan's theorem can help in this. For example, inputs for the function

$$F = A + B + C + D$$

would occupy four of the AND gates of the product matrix. However, the same function rewritten as

$$\overline{F} = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

would occupy only one. Moreover, the second function could be done on the simplest of the Series 20 devices (*and* leave eleven gates over for other functions), whereas the first could not. The fact that all inputs of the Series 20 devices, including the shared ones, incorporate double buffers that make the true and complement forms of all input variables equally accessible, greatly facilitates the use of de Morgan's theorem for logic minimisation.

To convert the minimised logic equations to the pattern of fuses to be blown you can use either a programming sheet (see e.g. Fig.5) or boolean equation program-entry software that lets you enter the equations via the keyboard of a terminal. The direct programmability of logic equations makes system design with IFL simple and sure. Functional changes can be made by replacing one IFL chip by another, differently programmed. In many cases you can even remove the original one, reprogram it on the spot, and re-insert it. Programming machines qualified for the Series 20 are at present available from DATA I/O, KONTRON, and STAG.

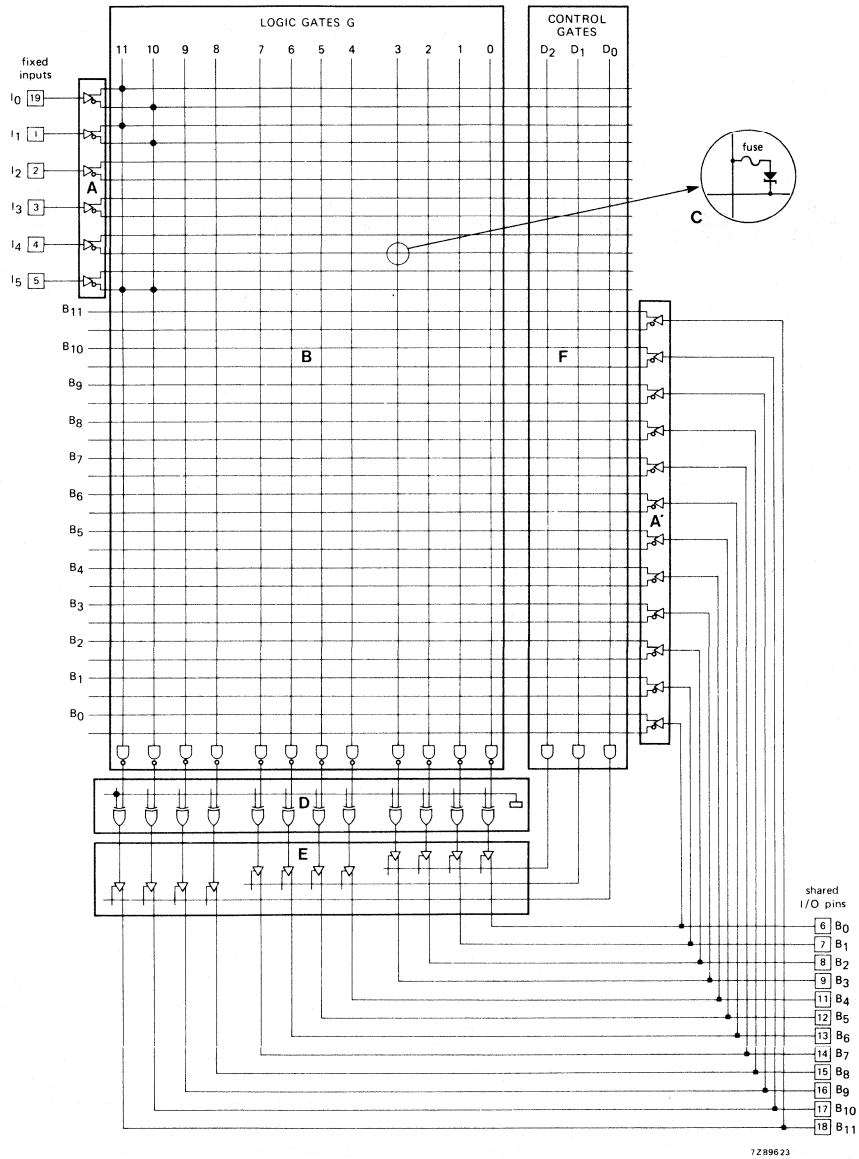


Fig.1 Field-programmable gate array 82S151. A, dedicated inputs; A', programmable I/O. B, product (NAND) matrix with fused connections C; each of the vertical lines in the matrix represents 36 inputs to the terminating NAND gate. D, exclusive-OR array with inputs grounded via fuses for polarity control. E, programmable three-state output buffers. F, fuse-programmable control matrix. Square dots (■) represent permanent connections, round dots (●) intact fuse connections. Connected as shown, the array is programmed for the functions

$$\overline{B}_{11} = I_0 I_1 \overline{I}_5 \quad \text{and} \quad B_{10} = \overline{I}_0 \overline{I}_1 \overline{I}_5$$

FPGA 82S150/82S151

The field-programmable gate array is the simplest of the Series 20 IFL devices; Fig.1 shows the functional diagram. The array can accept up to 18 inputs. There are six dedicated input pins (A) and twelve (A′) that can be programmed as inputs, outputs, or bidirectional I/O. All input variables, whether on dedicated or programmed input pins, are available in both true and complement form in the product matrix (B), and both forms are buffered: either form can drive all twelve product lines if required. In the virgin state, all the input variables and their complements are connected to all the product lines via a diode and a fuse (C), and the product matrix is effectively inoperative. To enable it to generate the required functions, unrequired connections between individual input lines and product lines are severed by blowing the connecting fuses.

At the output of the product matrix are twelve NAND gates, each with 36 inputs to accommodate the 18 possible input variables and their complements. Each of the product terms is normally active-Low, but a unique feature of Signetics IFL is that any or all of them can be independently programmed active-High. This is done by means of an array of exclusive-OR gates (D) at the NAND-gate outputs; when the fuse that grounds the second input of each OR gate is blown, the output of that gate is inverted.

The product matrix and exclusive OR-gate connections shown in Fig.1 illustrate the flexibility conferred by having buffered complements of all input variables internally available, together with independently programmable output polarities. Output B_{11}, shown with its exclusive OR-gate fuse intact, is programmed

$$\overline{B_{11}} = I_0\, I_1\, \overline{I_5}$$

At the same time, and without using any additional inputs, output B_{10} (fuse blown) is programmed

$$B_{10} = \overline{I_0}\, \overline{I_1}\, \overline{I_5}$$

Each of the exclusive-OR gates drives a three-state output buffer. In the virgin state all the buffers (E) are disabled and therefore in the high-impedance state. The function of the programmable I/O pins (A′) is then determined by the I/O control matrix (F). The three AND gates at the control-matrix output are active-High, and when one of them is in the High state the four output buffers it controls are enabled; the corresponding I/O pins then act as outputs. Conversely, when a control-matrix AND-gate output is Low and the control fuse for the corresponding three-state buffer is intact, the pins controlled by that gate act as inputs. Thus, these pins can be programmed in groups of up to four to act as inputs or outputs according to the state of selected input variables. If required, any of the programmable I/O pins can be made a dedicated output by blowing the control fuse of the output buffer associated with it.

The speed of the FPGA compares favourably with TTL, although its propagation delay (currently specified as 30 ns) is longer than the individual gate delay of TTL. When the number of inputs required is large, however, the FPGA more than makes up for this. When more than eight inputs are required, for example, the FPGA has a distinct advantage. Then, the overall propagation delay of TTL often amounts to two or three gate delays, but that of the FPGA to only one (see Table 1).

TABLE 1
Comparison of propagation delays for FPGA and TTL

number of inputs	TTL gates in series	t_{pd} (ns)			
		FPGA	TTL		
			74	74LS	74S
≤8	1	30	20	17	7
>8	2	30	40	35	16

FPLA 82S152/82S153

Architecture

With two levels of logic, embodied in a product matrix terminating in 32 AND gates coupled to a ten-output OR matrix (Fig.2), the FPLA is a step up in complexity from the FPGA. Again, there is provision for 18 input variables, internally complemented and buffered, but here divided between eight dedicated input pins and ten individually programmable I/O pins. As before, exclusive-OR gates grounded by fuses provide output polarity control, and any of the programmable I/O pins can be made a dedicated output by blowing the control fuse of the output buffer associated with it.

Programming

The first step in programming the FPLA is to define the required functions in boolean equations. These do not necessarily have to be minimised, but a good principle is: the fewer terms you use, the more capacity you will have left for implementing other function on the same FPLA or for altering the program during prototype development. Karnaugh maps will yield sums of products suitable for the FPLA. Logic minimisation methods (Quine-McCluskey, for instance, Ref.1,2) are particularly useful for complicated functions and are often necessary for reducing them to minimum-sums-of-products (MSP). A program called LOGMIN, accessible via data link to Signetics' computer, derives MSPs for an array of up to 16 inputs and 8 outputs.

When the required functions have been defined, corresponding programming instructions are entered in a programming table the layout of which reflects the FPLA

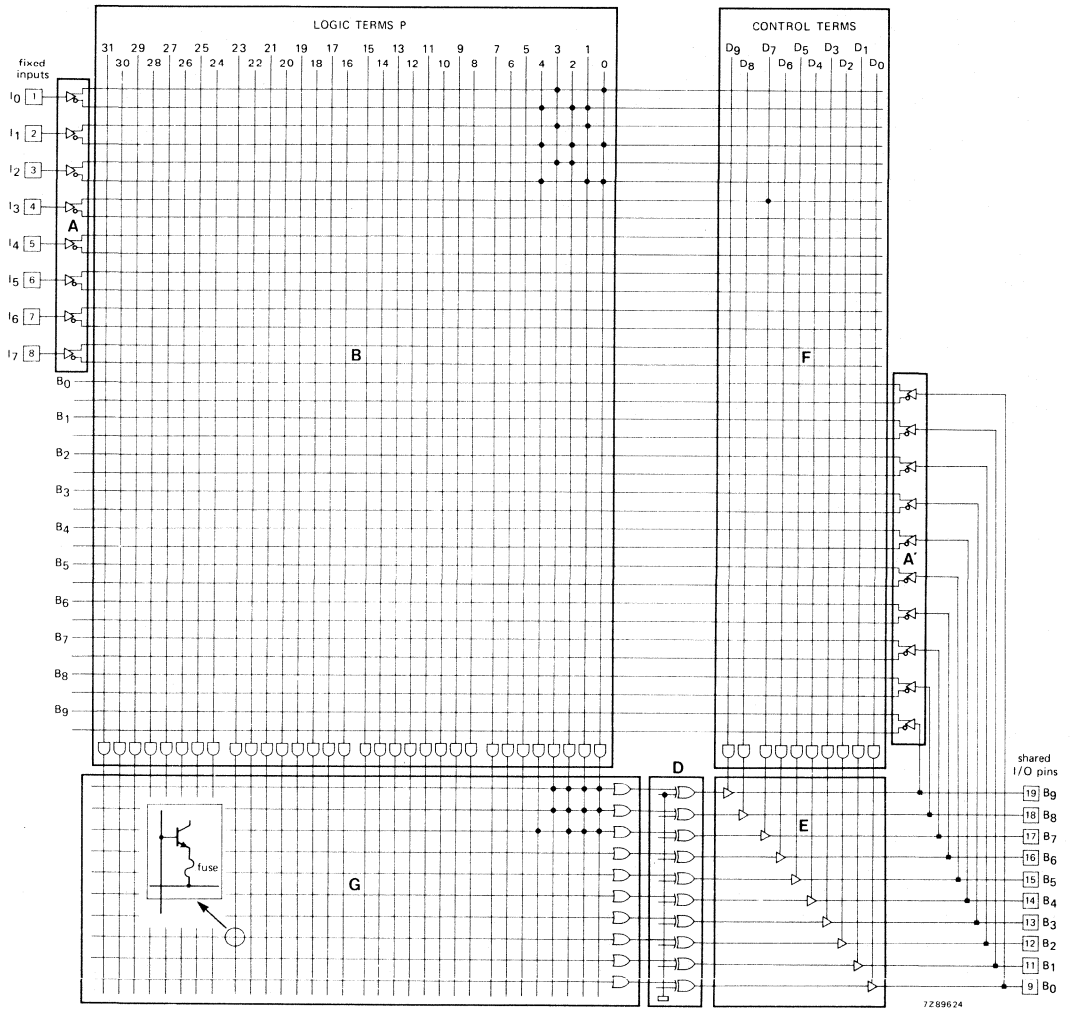


Fig.2 Field-programmable logic array 82S153. A to F, as in Fig.1. G, sum (OR) matrix. Connected as shown, the array is programmed as a single-bit full adder with Carry Enable

architecture. (A computer program that accepts boolean equations as input and generates an FPLA programming table as output is also available.) The programming machine blows the FPLA fuses in the pattern prescribed by the table.

As an illustration of FPLA programming, consider a full adder. Figure 3 shows a TTL version (74LS80) and the corresponding logic equations. Note that the feedback of $\overline{C}_{n+1}$ introduces a second propagation delay. In the FPLA this is eliminated by redefining Σ in terms of A, B and C_n, as shown in Fig.4, and using the right-hand side of the equation for $\overline{C}_{n+1}$ instead of the term itself. At first glance this would appear to require a minimum of three product terms for $\overline{C}_{n+1}$ plus four for Σ, or a total of seven. The Karnaugh maps, however, show considerable overlap between the two functions: the map for $\overline{C}_{n+1}$ differs from that for Σ only by having $A\,B\,C_n$ instead of $\overline{A}\,\overline{B}\,\overline{C}_n$. Rewriting the equation for $\overline{C}_{n+1}$ to introduce $\overline{A}\,\overline{B}\,\overline{C}_n$ and eliminate $A\,B\,C_n$,

$$\overline{C}_{n+1} = A\,\overline{B}\,\overline{C}_n + \overline{A}\,B\,\overline{C}_n + \overline{A}\,\overline{B}\,C_n + \overline{A}\,\overline{B}\,\overline{C}_n$$

increases the number of product terms by one, but now $\overline{C}_{n+1}$ and Σ have three terms in common. Therefore, since the FPLA allows multiple use of product terms, it is sufficient to program each of the common terms only once; thus, the original seven product terms are effectively reduced to five.

To fill in the programming table (Fig.5), first allocate inputs and outputs.

Inputs		Outputs:	
A	$= I_0$	$\overline{C}_{n+1}$	$= B_7$
B	$= I_1$	$\overline{\Sigma}$	$= B_8$
C_n	$= I_2$	Σ	$= B_9$

Next, enter the product terms of Σ in the product-matrix (AND) part of the table, using H to indicate a true input and L a false one.

- Term 0 is $A\,\overline{B}\,\overline{C}_n$: mark H, L, L in columns I_0, I_1, I_2 of row 0
- Term 1 is $\overline{A}\,B\,\overline{C}_n$: mark L, H, L in columns I_0, I_1, I_2 of row 1
- Term 2 is $\overline{A}\,\overline{B}\,C_n$: mark L, L, H in columns I_0, I_1, I_2 of row 2
- Term 3 is $A\,B\,C_n$: mark H, H, H in columns I_0, I_1, I_2 of row 3.

Fill the rest of rows 0, 1, 2, and 3 with dashes to indicate that all other inputs are to be disconnected from Terms 0, 1, 2, and 3 (fuses blown).

The product terms of Σ must be added to form the sum-of-products required at output B_9. Indicate the required addition by putting an A (for Attached, i.e. fuse unblown) in the Term 0, 1, 2 and 3 spaces of column $B(O)_9$; Term 4 is not required for Σ, so put a dot in the Term 4 space to indicate that it is to be disconnected (fuse blown). To indicate that the output is to be active-High, put an H in the polarity square above the $B(O)_9$ column. Finally, fill row D_9 with

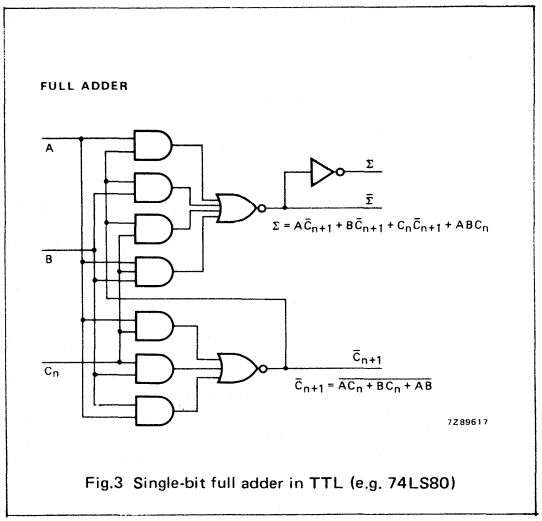


Fig.3 Single-bit full adder in TTL (e.g. 74LS80)

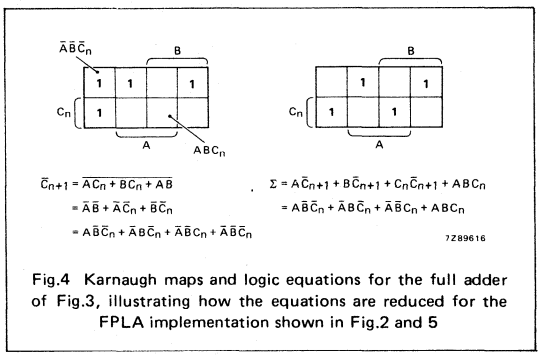


Fig.4 Karnaugh maps and logic equations for the full adder of Fig.3, illustrating how the equations are reduced for the FPLA implementation shown in Fig.2 and 5

dashes to indicate that all fuses on line D_9 of the control matrix are to be blown and B_9 is to be a dedicated output. This completes the programming of Σ.

The $\overline{\Sigma}$ output on B_8 is programmed in just the same way, except that the polarity square above the $B(O)_8$ column is marked L to indicate active-Low. (Note that in the FPLA the Σ and $\overline{\Sigma}$ outputs change simultaneously, because all output signals traverse the exclusive-OR array (D), whether they are active-Hight or active-Low. In the TTL full adder shown in Fig.3 the output inverter delays the change of Σ with respect to $\overline{\Sigma}$.)

The output C_{n+1} on B_7 contains three of the same terms as Σ, plus the term $\overline{A}\,\overline{B}\,\overline{C}_n$. Only this last term needs to be additionally programmed in the product matrix: mark L, L, L in columns I_0, I_1, I_2 of the Term 4 row. Indicate the addition

$$A\,\overline{B}\,\overline{C}_n + \overline{A}\,B\,\overline{C}_n + \overline{A}\,\overline{B}\,C_n + \overline{A}\,\overline{B}\,\overline{C}_n$$

by putting an A in rows 0, 1, 2 and 4 of column $B(O)_7$, and show that Term 3 ($A\,B\,C_n$) is not required by putting a dot

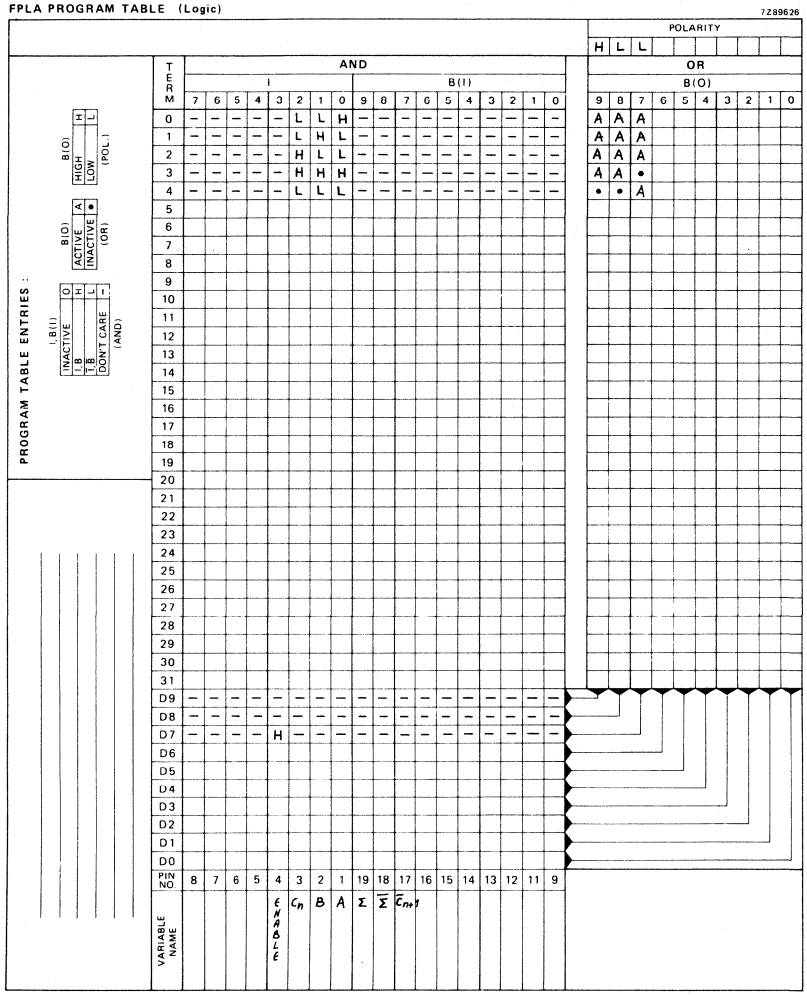


Fig.5 FPLA programming table filled in for the full adder of Fig.2

in the Term 3 row to indicate disconnection (fuse blown). Put an L in the $B(O)_7$ polarity square to indicate active-Low.

Identifying B_7 as a dedicated output by indicating that all the fuses to control term D_7 are to be blown, would now complete the programming of the full adder. However, a useful supplementary feature would be a Carry Enable function to keep the B_7 output buffer in the high-impedance state except when the enable input I_3 is true. The output buffer is enabled when both the fuses of a control term are blown, or when one is blown and the term that controls the

output buffer is true. Thus, a Carry Enable can be provided via the I_3 input by leaving the fuse for active-High operation of the enable signal to control term D_7 intact. To indicate this, put an H in the I_3 column of row D_7 and fill the rest of the row with dashes.

The full adder with output Carry Enable uses only four of the eight dedicated inputs, three of the ten programmable I/O pins, and five of the 32 AND gates. The remaining capacity can be used for programming other functions which may, if required, also make use of AND-gate outputs already programmed for the full adder.

All fuses not indicated as blown in the programming table are normally left intact to preserve capacity for later program revisions or the addition of supplementary functions. If it is essential to minimise propagation delay, however, the finalised program should include instructions for blowing all unused fuses to minimise load capacitance.

Table 2 compares the propagation delay of the FPLA and TTL; the value given for the FPLA is a worst-case value with all fuses intact.

TABLE 2
Comparison of two-level logic propagation delays for FPLA and TTL

number of of inputs	TTL gates in series	t_{pd} (ns)			
		FPLA	TTL		
			74	74LS	74S
≤4	1	40	40	33	14
5–8	2	40	43	35	16
>8	3	40	63	53	25

FPLS 82S154 – 82S159

Architecture

The FPLS (Fig.6, page 170) is the most complex of the Series 20 IFL devices. Like the FPLA, it has a 32-term product matrix followed by an OR matrix. In the FPLS, however, the OR matrix is larger and comprises three distinct parts, with architecture differing in detail from type to type. In the 82S154 and 82S155, for instance, the first part consists of eight 32-input gates coupled, like those of the FPLA, to an output-polarity-controlling exclusive-OR array. The second consists of twelve additional gates which control four flip-flops. These are what give the FPLS its sequential character, enabling it to dictate its next state as a function of its present state. The third part is the deceptively simple complement array (I in Fig.6): a single OR gate with its output inverted and fed back into the product matrix. This enables a chosen sum-of-products to become a common factor of any or all the product terms and makes it possible to work factored sum-of-products equations. It is also useful for handshaking control when interfacing with a processor and for altering the sequence of a state machine without resorting to a large number of product terms.

The 82S154 and 82S155 have four dedicated inputs and eight programmable I/O pins that can be allocated in the same way as in the FPLA. It also has four shared I/O pins (L) whereby the flip-flops can be interfaced with a bi-directional data bus. Two product terms, L_A and L_B in the control matrix F, control the loading of the flip-flops, in pairs, synchronised with the clock.

Figure 7 shows the architecture of the flip-flop circuitry in the 82S154 and 82S155. The flip-flops are positive-edge triggered and can be dynamically changed to J-K, T, or D-types according to the requirements of the function being performed; this considerably lessens the demands on the logic. The three-state inverter between the J and K inputs governs the mode of operation, under the control of the product term F:

- When the inverter is in the high-impedance state the flip-flop is a J-K type, or a T type when J = K.
- When the inverter is active, $K = \overline{J}$ and the flip-flop is a D type; the K input must then be disconnected from the OR matrix.

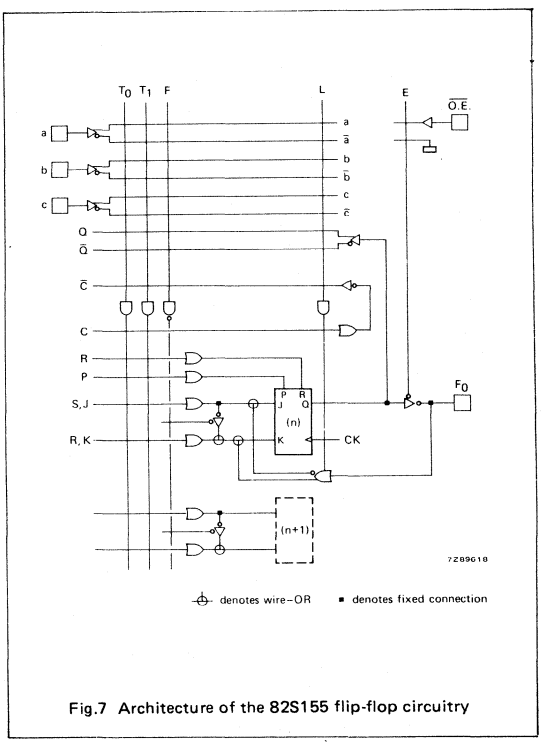


Fig.7 Architecture of the 82S155 flip-flop circuitry

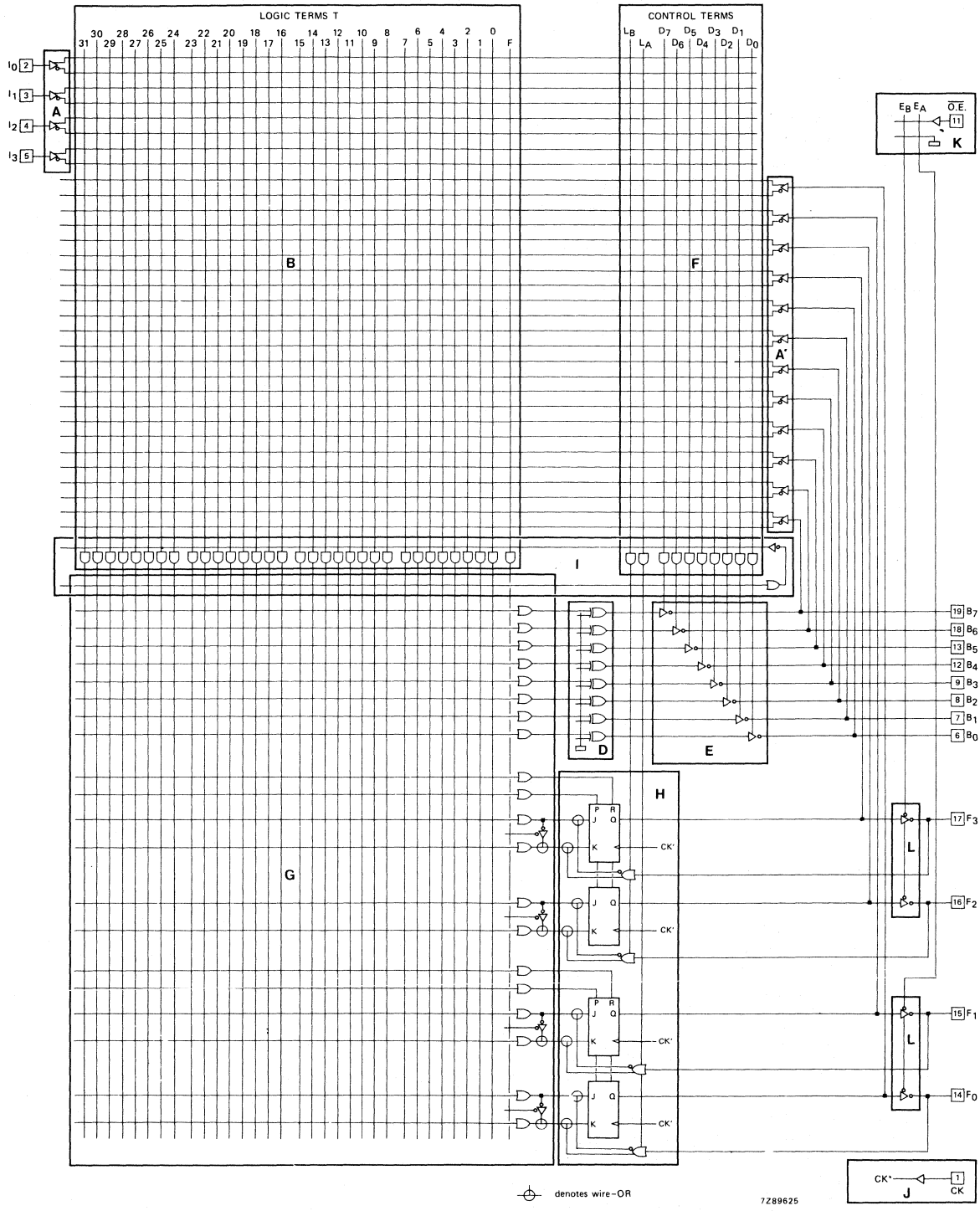


Fig.6 Field-programmable logic sequencer 82S155. A to G, as in Fig.3. H, flip-flops and bus-load buffer. J, clock input. K, output-buffer enable. L, three-state flip-flop output buffers

All the product terms from the product matrix (T_0 to T_{31} in Fig.6) are fuse-connected to the J and K input OR gates. If both fuses of any one product term are left intact, J = K and the flip-flop is a T type.

The flip-flops of the 82S154 and 82S155 have asynchronous Preset and Reset controlled by terms in the OR matrix that take priority over the clock. When power is applied all flip-flops are set to a defined state. Their three-state output buffers can be controlled from the enable pin OE or permanently enabled or disabled by blowing fuses or leaving them intact in the enable array (K in Fig.6).

The 82S156/7 and 82S158/9 sequencers have, respectively, six and eight flip-flops. The architecture differs in detail but is similar in principle to that of 82S154 and 82S155.

Programming

The FPLS is programmed in much the same way as the FPLA, using a table to instruct the machine that blows the undesired fuses. It is not necessary to work with a circuit diagram; in fact, it is even undesirable to do so, since applying the necessary logic reduction techniques would in most cases make the diagram difficult to read and more a hindrance than a help. An example of how to program the FPLS as a universal counter/shift-register is given in the Appendix.

DEVELOPMENT AND PRODUCTION ECONOMY WITH IFL

Underlying the design philosophy of the Signetics Series 20 IFL is the concept of programmable arrays whose architecture emulates logic equation formats rather than mere aggregations of gates. The unique combination of features which support this philosophy includes:

- double-buffered true and complement inputs
- programmable-polarity outputs
- programmable I/O for internal feedback and maximum freedom in allocating inputs and outputs
- truth-table programming format.

These features are common to all the IFL devices. In the field-programmable logic sequencers they are further supported by:

- flip-flops with dynamically alterable operating modes
- a complement array for simplified handshaking control.

From the development engineer's point of view an important advantage of IFL is that it eliminates breadboarding. Once he has worked out the functions required in terms of minimised logic equations, he can program an IFL device accordingly. Once programmed, it will perform those functions.

Loading the instructions into the programming machine usually takes no more than a couple of hours; after that, the machine can program the devices at a rate of 100 an hour. Moreover, since any IFL device can be programmed in many different ways, IFL has considerable potential for simplifying purchasing and stock control. One type of device can be programmed to perform a diversity of tasks for which it would otherwise be necessary to purchase and stock many different devices.

Series 20 IFL is second-sourced by Harris Semiconductor.

APPENDIX

Programming an FPLS as a counter/shift-register

Objective: to program an 82S155 FPLS as a count-up, count-down, shift-right, shift-left machine governed by three control terms – COUNT/$\overline{\text{SHIFT}}$, RIGHT/UP, LEFT/DOWN. Direct implementation would result in a machine with 64 state transitions (see Table A1), which is beyond the scope of the 82S155 or even the 28-pin 82S105. Logic reduction is therefore necessary.

As there are only four feedback variables (D, C, B, A), you can do the reduction by hand, one mode at a time; the control terms need not be included till the summary equations are written. Using the transition mapping method suggested here, you can examine the excitation equations for all types of flip-flops (R-S, J-K, D, T) and choose those types that will perform the required functions using the fewest product terms. Table A2 summarises the rules for flip-flop implementation using transition maps; the transition symbols used in the table mean:

present state	next state	transition symbol
0	0	0
0	1	α
1	0	β
1	1	1

Using these symbols, construct Table A3 from Table A1 to enable you to examine the excitation equations for all types of flip-flops. Proceeding one mode at a time, transfer the state conditions from Table A3 to Karnaugh maps, as in Fig.A1. Following the rules in Table A2, derive the excitation equations for the different types of flip-flops (the examples shown in Fig.A1 omit the T type because it is the same as the J-K type when J = K). In deciding which types of flip-flop to use, remember that logic minimisation with IFL is different from logic minimisation with 'random logic':

TABLE A1
Present-state/next-state table for counter/shift-register

state no.	present state				next state															
					count down				count up				shift left				shift right			
	D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A
0	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0
2	0	0	1	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0	1
3	0	0	1	1	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	1
4	0	1	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	1	0
5	0	1	0	1	0	1	0	0	0	1	1	0	1	0	1	0	1	0	1	0
6	0	1	1	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	1	1
7	0	1	1	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	1	1
8	1	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0
9	1	0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0
10	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0	1	0	1	0	1
11	1	0	1	1	1	0	1	0	1	1	0	0	0	1	1	1	1	1	0	1
12	1	1	0	0	1	0	1	1	1	1	0	1	1	0	0	1	0	1	1	0
13	1	1	0	1	1	1	0	0	1	1	1	0	1	0	1	1	1	1	1	0
14	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1
15	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
control terms																				
COUNT/$\overline{\text{SHIFT}}$					1				1				0				0			
RIGHT/UP					0				1				0				1			
LEFT/DOWN					1				0				1				0			

TABLE A3
Transition table for counter/shift-register

state no.	present state				transition															
					count down				count up				shift left				shift right			
	D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A
0	0	0	0	0	α	α	α	α	0	0	0	α	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	β	0	0	α	β	0	0	α	β	α	0	0	β
2	0	0	1	0	0	0	β	α	0	0	1	α	0	α	β	0	0	0	β	α
3	0	0	1	1	0	0	1	β	0	α	β	β	0	α	1	β	α	0	β	1
4	0	1	0	0	0	β	α	α	0	1	0	α	α	β	0	0	0	β	α	0
5	0	1	0	1	0	1	0	β	0	1	α	β	α	β	α	β	α	β	α	β
6	0	1	1	0	0	1	β	α	0	1	1	α	α	1	β	0	0	β	1	α
7	0	1	1	1	0	1	1	β	α	β	β	β	α	1	1	β	α	β	1	1
8	1	0	0	0	β	α	α	α	1	0	0	α	β	0	0	α	β	α	0	0
9	1	0	0	1	1	0	0	β	1	0	α	β	β	0	α	1	1	α	0	β
10	1	0	1	0	1	0	β	α	1	0	1	α	β	α	β	α	β	α	β	α
11	1	0	1	1	1	0	1	β	1	α	β	β	β	α	1	1	1	α	β	1
12	1	1	0	0	1	β	α	α	1	1	0	α	1	β	0	α	β	1	α	0
13	1	1	0	1	1	1	0	β	1	1	α	β	1	β	α	1	1	1	α	β
14	1	1	1	0	1	1	β	α	1	1	1	α	1	1	β	α	β	1	1	α
15	1	1	1	1	1	1	1	β	β	β	β	β	1	1	1	1	1	1	1	1

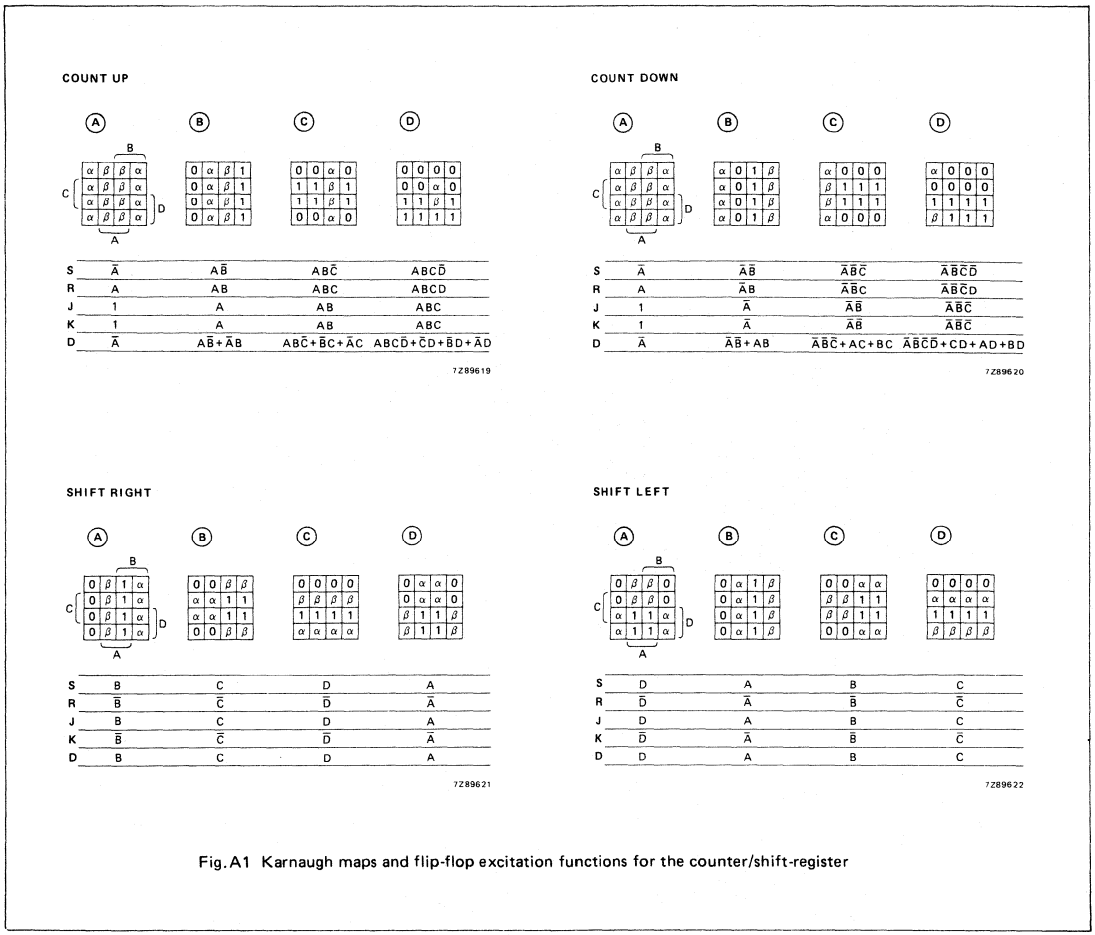


Fig. A1 Karnaugh maps and flip-flop excitation functions for the counter/shift-register

TABLE A2
Rules for flip-flop implementation using transition maps

flip-flop type	input	must include	must exclude	redundant
R-S	S	α	β,0	1,x
	R	β	α,1	0,x
D	D	α,1	β,0	x
T	T	α,β	0,1	x
J-K	J	α	0	1,β,x
	K	β	1	0,α,x

TABLE A4
Number of product terms required for counter/shift-register flip-flop excitation

flip-flop type	count up	count down	shift right	shift left	total
S R only	8	8	8	8	32
J K only	4	4	8	8	24
D only	10	10	4	4	28
FPLS	4(J-K)	4(J-K)	4(D)	4(D)	16

with random logic you seek to reduce the number of standard packages required; with IFL you seek to reduce the number of product terms.

From Fig.A1 it is evident that you should choose J-K or T flip-flops for the counter mode and D flip-flops for the shift mode, for you then require only one product term per flip-flop per mode. Table A4 summarises the number of product terms per mode the various types of flip-flops would require.

Figure A2 shows the completed programming table for

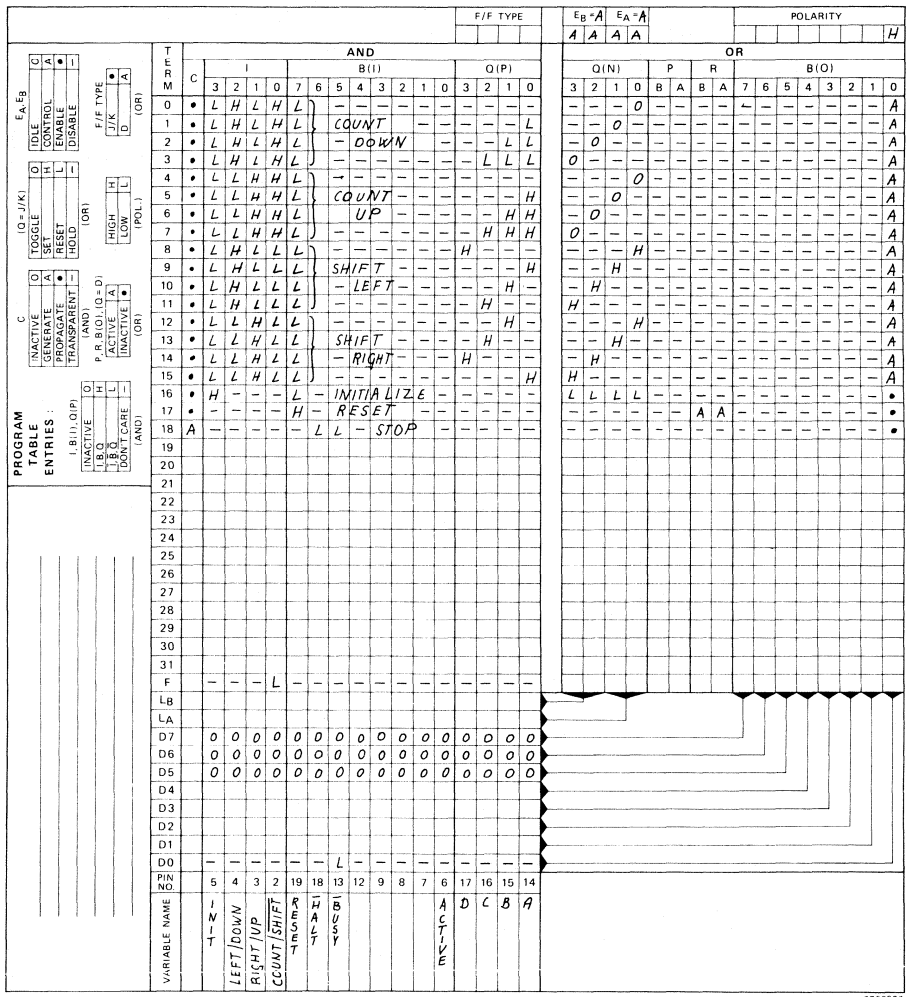


Fig.A2 82S155 FPLS programming table for the counter/shift register

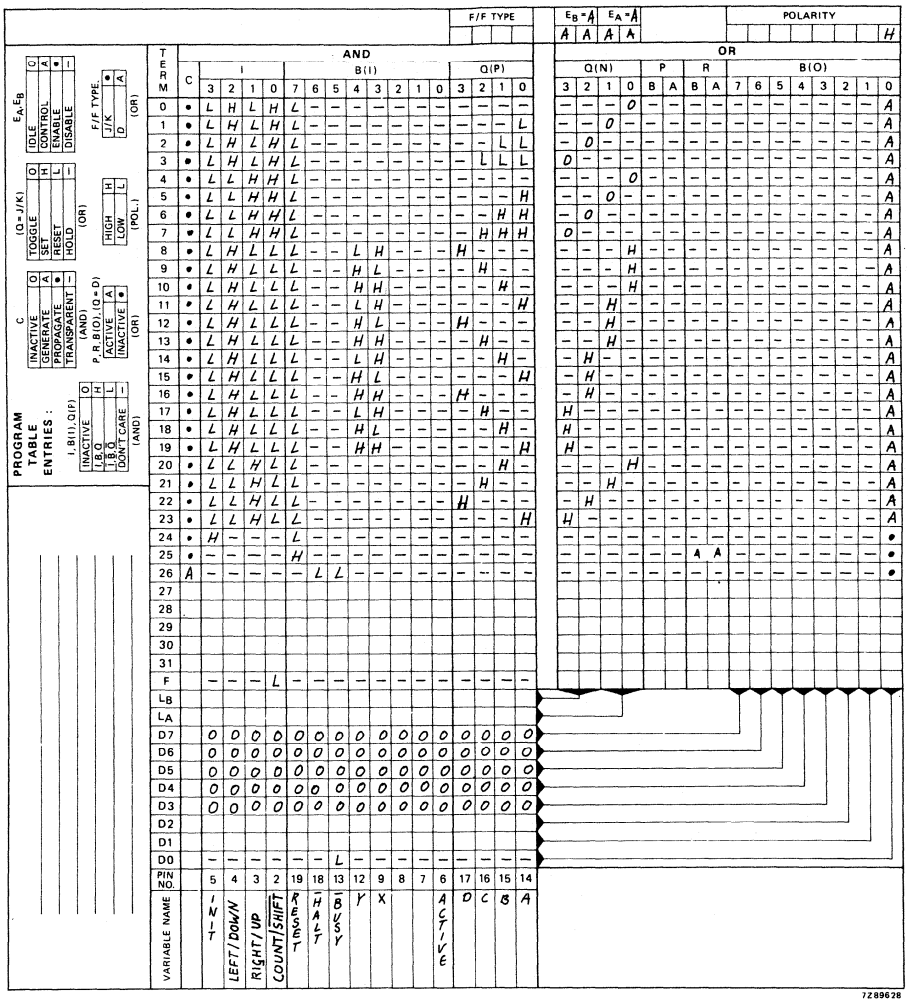


Fig.A3 Modified 82S155 FPLS programming table for the counter/shift-register with the addition of a binary multiplier

the counter/shift-register. The programming of Terms 0 to 15 reflects the flip-flop excitation equations and illustrates the value of being able to switch the flip-flops dynamically from one type of operation to another. Terms 16, 17 and 18, respectively, provide for INITIALIZE, asynchronous RESET, and STOP functions.

The programming of the two additional inputs $\overline{\text{HALT}}$ and $\overline{\text{BUSY}}$ illustrates the value of the complement array, which is made active when $\overline{\text{HALT}}$ and $\overline{\text{BUSY}}$ are Low (A in the Complement square of Term 18) and propagated into all the other terms (dot in the Complement squares of Terms 0 to 17). This means that unless the $\overline{\text{HALT}}$ and $\overline{\text{BUSY}}$ inputs are High none of the product terms will be true and the state of the machine will not change. If the complement array were not used, twice the number of product terms would be required, even if one of the additional inputs were omitted.

As it is, the design uses only 19 of the 32 product terms available, so there is ample capacity for extending its capabilities. For example, the shift-left function can be augmented by a binary multiplication capability, using a D type flip-flop to make it shift one, two, or three places according to the state of two extra inputs, X and Y. Figure A3 shows the revised programming table. The binary multiplication function occupies nine additional product terms and is an example of why initial product-term minimisation is always worthwhile: it affords design flexibility and frees capacity for additional features or refinements that might not otherwise be feasible.

The IFL counter/shift-register has a set-up time of 50 ns – just half what it would be if LS TTL were used.

REFERENCES

1. QUINE, W. V. 1952. 'The problem of simplifying truth functions' Am. Math. Monthly 59. pp. 521-531.
2. MC CLUSKEY, E. J. 1956. 'Minimization of Boolean functions' Bell System Technical Journal 35. pp. 1417-1444.

ACKNOWLEDGEMENT

The author wishes to thank Bill Wiley Smith, bipolar memory product applications manager at Signetics Corporation, Sunnyvale, for his creative contribution to a more widespread understanding of integrated fuse logic.

1.0 LOGMIN INTRODUCTION

LOGMIN is a computer program which performs multiple output AND-OR logic minimization. It can find a nearly minimal AND-OR network for any problem with 16 or fewer inputs and eight or fewer outputs.

Its main purpose is to minimize program tables for the Signetics' 82S100/101 Field Programmable Logic Array. Minimization can be a key concern to the array designer, since it allows him to compress more logic functions in each FPLA, thereby reducing the number of logic elements, power, and cost required to implement any given function.

2.0 WHAT IS AN FPLA?

Signetics' FPLAs are fast, user programmable, TTL logic elements with memory, which can streamline logic system design by integrating the equivalent of 528 TTL gates in 196 packages into a *single* IC package.

In terms of logic, the FPLA is a two level AND-OR, AND-NOR combinational logic element, consisting of a system of logic gates with programmable inputs and outputs as shown in Figure 1. These, by means of on-chip programmable connectors, enable the user to quickly implement eight logic functions with a maximum of 48 product (AND) terms, involving up to 16 input variables.

FPLA AS A TWO LEVEL PROGRAMMABLE LOGIC ELEMENT

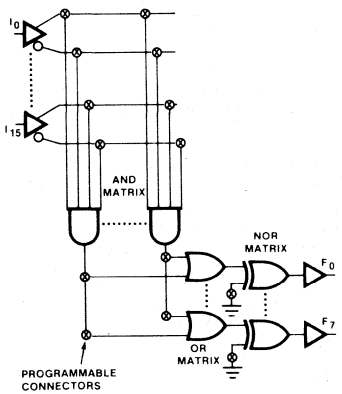


Figure 1

A more detailed organization of the FPLA is shown in Figure 2. The device consists of an upper resistor-diode AND matrix containing 48 product term columns (P-terms), and a lower emitter-follower OR matrix containing eight sum term rows (S-terms), one for each output function. Each P-term in the AND matrix is initially coupled to 16 true and complement input variables via 32 fusible Ni-Cr links for programming any desired input combination.

Each P-term is also coupled to each S-term in the OR matrix through an emitter fuse, for pulling the summing node to a high level when the P-term is activated. Each S-term in turn is coupled to its respective output via an Ex-OR gate, which has programmable transmission polarity by means of an input to ground fusible link.

Selective programmable of the internal links allows the user to create specific logic paths for producing any logic functions as a sum of products, defined in the typical Karnaugh Map of Figure 3.

LOGIC STRUCTURE OF FPLA ILLUSTRATING AND, OR, AND EX-OR ARRAYS

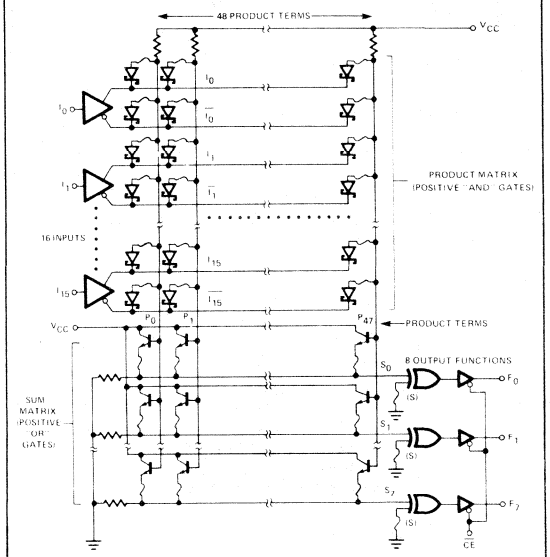
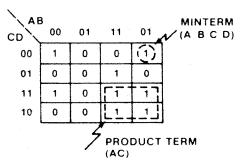


Figure 2

TYPICAL KARNAUGH MAP



Minterms enclose a single cell.
Product terms enclose 2 or more cells.

Figure 3

The transmission through the FPLA can be traced along the equivalent logic path shown in Figure 4. From this figure, it is apparent that Signetics' FPLA is basically a two level logic element. The first level produces 48 internal AND functions, P_0 through P_{47}, of up to 16 logic input variables, or their complement. The second level produces eight OR output functions, F_0 through F_7, each involving up to 48 of the internally generated AND terms. Alternately, if desired, this second logic level can be programmed to provide 8 NOR output functions F_0^* through F_7^*. However, for each of the 8 outputs, either the function F_p (active-high), or F_p^* (active-low) is available, but not both. The required output polarity is programmed by the user via link (S). The overall logic function provided by each FPLA output is summarized in Figure 5.

When viewed strictly as a logic element, the FPLA can be used to implement sets of logic equations of the type:

$$F_0 = I_0 + I_1 \overline{I}_5 + I_2 \overline{I}_3 I_7 \ldots$$

$$F_1 = \overline{I}_0 + I_1 \overline{I}_5 + I_6 I_7 I_8 \ldots$$

$$\overline{F}_2 = \overline{I}_2 + I_7 I_5 + \ldots \text{etc.}$$

EQUIVALENT LOGIC PATH OF SIGNETICS' FPLA

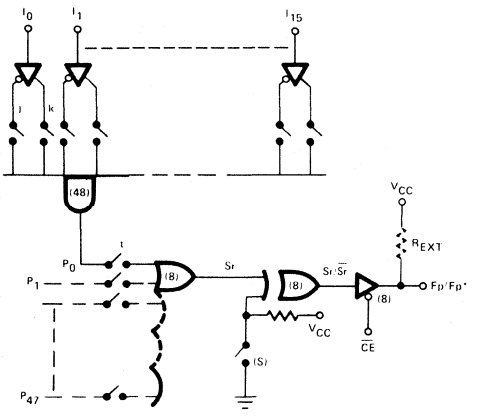
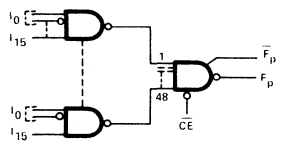


Figure 4

TWO LEVEL LOGIC FUNCTION PROVIDED BY EACH OF THE 8 FPLA OUTPUTS



All input NANDs have 16 pairs of inputs for gating either true or complement states of logic variables. Fp or $\overline{Fp}$ are obtained with $\overline{CE}$ = "0."

Figure 5

or, by use of De Morgan's theorem, their equivalent as for F_0:

$$F_0 = \overline{(\overline{I_0})\,(\overline{I_1} + I_5)\,(\overline{I_2} + I_3 + \overline{I_7})\ldots}$$

This is readily shown in the logic equivalence of Figure 6.

Generally FPLAs are effectively used in design situations involving many input variables and few active logic states; and, with a maximum access time of 50ns, the FPLA is a practical alternative to the long logic chains necessary when dealing with several input variables.

The following example is a brief, but concise, illustration of how to integrate random logic with discrete gates into a Signetics' FPLA. Given the set of logic equations F_{1-4} below:

$$F_1 = X_1 + \overline{X}_2\overline{X}_3$$
$$F_2 = X_2 + \overline{X}_1\overline{X}_3$$
$$F_3 = X_3 + \overline{X}_1\overline{X}_2$$
$$F_4 = \overline{X}_1X_3 + \overline{X}_1\overline{X}_2$$

These can be implemented with discrete gates as in the AND-OR-NOT logic network of Figure 7.

This method is practical for simple systems; but in more complex applications, it soon produces a distributed logic network with many IC packages and types, difficult to design, troubleshoot and modify.

LOGIC EQUIVALENCE OF F_0

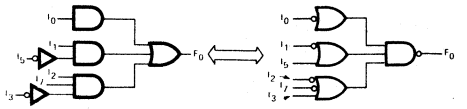


Figure 6

DISCRETE GATE LOGIC NETWORK FOR EQUATION SET F_{1-4}

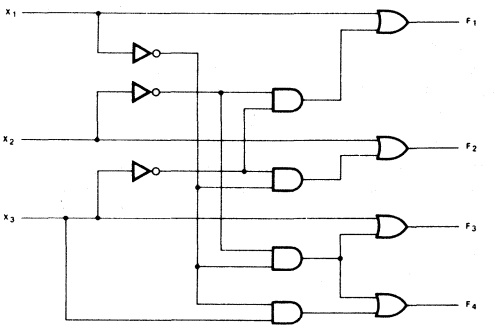


Figure 7

On the other hand, the same set of equations can be easily coded in an FPLA Program Table and programmed in a device using inexpensive field equipment.

Typically, F_1 would require the FPLA to contain the fused link pattern shown in Figure 8, as specified in the accompanying Program Table slice. Overall, all four logic functions would use three inputs, four outputs and seven product terms of the FPLA, leaving remaining resources spare for late modifications.

For example, if it becomes necessary to change the X_1 product term in F_1 to $\overline{X}_1$, deleting the wrong product term and adding the new one becomes a trivial task, as indicated in the modified pattern and revised Program Table of Figure 9.

These modifications can be made at any time in the field by the user, usually within the same device (as long as spare resources are available), by means of inexpensive programming equipment (as low as $350).

2.1 FPLA RESOURCES

Signetics' family of bipolar Field Programmable Logic Arrays includes both tri-state (82S100), and open collector devices (82S101), featuring the following characteristics:

- Field programmable (Ni-Cr link)
- 16-input variables
- 8 output functions
- 48 product terms
- 50ns max. access time (0-75°C)
- 600mW power dissipation (typical)
- TTL compatible
- 28-pin package
- $\overline{CE}$ input for expansion or inhibit
- Outputs individually programmable active "high" or "low"
- Single +5V power supply

INTEGRATING LOGIC WITH FPLAs

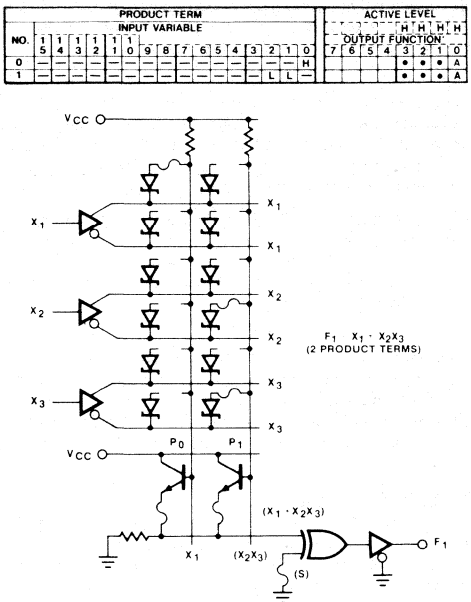
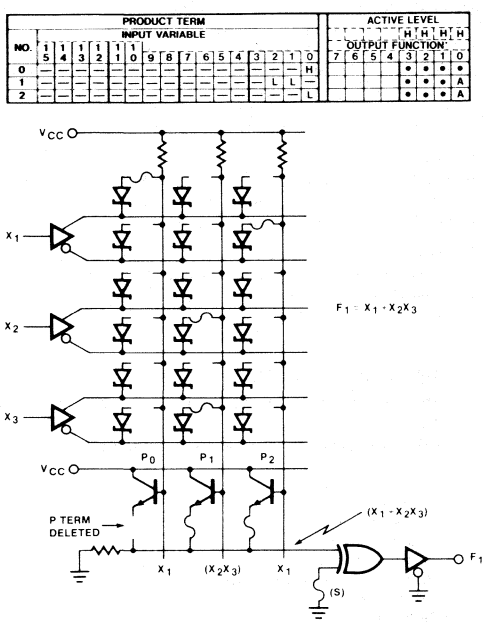


Figure 8

MODIFYING LOGIC WITH FPLAs



F_1 modified by deleting term X_1 in the OR matrix and adding new term $\bar{X}_1$.

Figure 9

These features and organization combine into an easy to use, high performance device, affording distinct user benefits:

A. 16-input variables
The 16X8 I/O configuration permits direct byte manipulations required by intelligent terminals, peripherals, microprocessor based emulators, minicomputers, and all the way up to the larger mainframes. Also, in address mapping applications, it provides the capability to scan an address field 65,536 words deep.

B. Chip Enable input
The Chip Enable input is a major improvement over alternate devices:

- Eases expansion of input variables and/or product terms.
- Permits application of tri-state device in bus organized systems.
- Provides logic inhibit or preconditional decoding functions.
- Provides a unique "default" logic state for all outputs, regardless of programmed output polarity.

C. Fastest access time
50ns maximum over the commercial temperature range renders the replacement of random logic feasible.

D. Fully buffered devices
All product terms can be utilized as many times as required, without affecting device speed and power dissipation.

E. 48 product terms (P-terms)
Allow the user to store in the FPLA 48 distinct words of eight bits each. These 48 words can be addressed by a minimum of 48 input address combinations, chosen by the user among a total available pool of 2^{16} (65,536).

F. Polarity of all outputs individually programmable active-high or active-low
This feature is particularly useful in achieving further product term minimization in cases where the complement of an output function can be implemented with fewer product terms.

Example:
As shown in Figure 10, a 50% reduction in P-terms is obtained when the output of the logical structure of $\overline{Fy}$ is inverted by means of a gate external to the elementary FPLA. The desired function Fy is then realized with penalties in hardware, and circuit delays (however small). These are eliminated when using an FPLA with output polarity programmed active-low to realize the function zero's, rather than one's.

3.0 LOGIC MINIMIZATION TECHNIQUES

Three Logic minimization approaches are useful with the FPLA.

As shown in Figure 10, the use of the output polarity exclusive-or gate can reduce required FPLA resources. It will be shown in FPLA EXPANSION how this gate can also be used to directly parallel "N" FPLA's such that they perform as a single FPLA but with "N" times the product term resources.

The bit slice technique of minimization involves solving each output function separately in terms of its input variables. A popular method of accomplishing this is through the use of Karnaugh Maps. Although bit slice minimization provides a minimal solution for each output function, the overall system realization with all outputs viewed together may not be minimum.

The third minimization approach is Simultaneous Minimization. Whereas bit slice minimization provides a minimum single output solution but not necessarily a minimum system solution, simultaneous minimization provides a minimum system realization but not necessarily a minimum solution for each output. Systematic methods such as the Quine-McClusky method yield good results for problems smaller than eight or nine inputs with five or six outputs, but extension to larger problems requires laborious hand manipulations which inevitably are prone to error.

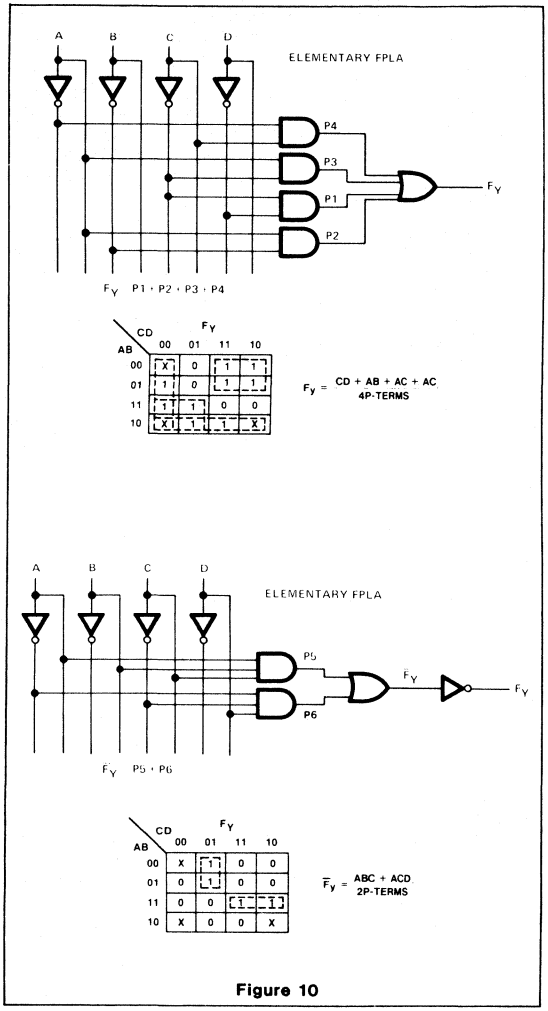


Figure 10

To illustrate bit slice and simultaneous minimization methods, the results of an FPLA code conversion problem will be examined in Figure 11. The unminimized code conversion table shown in 11a shows that 83 product terms are required to map the 16-bit input code into a six bit output code. Through multiple use of Karnaugh Maps and traditional Boolean Algebra, the bit slice method is able to implement the same code conversion in 52 P-terms. This particular minimization will require two FPLA's to implement just like the unminimized table. The only benefit that one has gained from this effort is perhaps program space to perform future code additions or modifications. The simultaneous minimization in 11c provides a conversion in just 44 product terms; eight less than the bit slice method, but more important in this application, it has enabled the designer to implement the entire code conversion in one FPLA instead of two!

4.0 GENERAL DESCRIPTION OF SIGNETICS' SIMULTANEOUS MINIMIZATION PROGRAM

Most all existing AND-OR minimization algorithms fall into two basic categories. The first category is successive approximation. It includes algorithms which start with a realization and improve it iteratively through random permutations of the realization and localized application of classical ideas from switching theory. IBM's "MINI" algorithm is an example in this category. Algorithms of this type have not been very effective on problems which have a significant proportion of ones in their truth tables.

The other category includes various implementations of the Quine-McClusky method. Algorithms in this class fail on large problems because of the size of the multiple output prime implicant covering table. An upper bound on the table size is $3^N \times 2^N \times (2^M - 1)$ bits, where N and M are the number of inputs and outputs, respectively. For 16 inputs and eight outputs, that number is over 10^{14} bits!

Signetics' program fits into neither category. It is based on two new and proprietary logic synthesis algorithms. It is much faster than the current state-of-the-art because it does not use the enormous prime implicant covering table of the Quine-McClusky method. The AND-OR realizations produced are not absolutely minimal solutions, but usually are very good.

The algorithm quickly discovers all essential multiple output prime implicants in the truth table. Since most truth tables are not completely covered by essential prime implicants, the second algorithm completes the realization by calling a subroutine called CLEANUP. CLEANUP takes most of the time consumed by the minimization algorithm.

5.0 USE OF LOGMIN

LOGMIN (Logic Minimization) is a preprocessing program that assembles AND-OR input data into a file suitable for use as a minimization source file. There are two data entry modes presently available, keyboard and paper tape.

Keyboard AND-OR data entry is done from the user's terminal in a conversational manner. LOGMIN will direct the user to key in the necessary AND-OR data in the correct format and will also provide error diagnostics for the case of incorrect data entry.

Paper tape entry is done via a teletype tape reader. LOGMIN will ask a few preliminary questions and then request that the user's paper tape be sent. The paper tape format is that of a standard FPLA programming tape. As before, LOGMIN assembles the tape entry into a minimization source file. The minimized output file generated may be listed on the user's terminal and/or a paper tape can be punched on the user's teletype. The output paper tape is of suitable format to program an FPLA.

Field-programmable arrays: powerful alternatives to random logic

Bridging design gap, TTL-compatible logic family is described in Part 1 of a two-part article

by Napoleone Cavlan and Stephen J. Durham, *Signetics Corp., Sunnyvale, Calif.*

☐ With the steady growth of integrated circuit technologies, hardly a day goes by without the news that yet another chip has made scores of discrete TTL packages obsolete. Yet, though large-scale integration is packing entire system architectures onto a few chips, it is still impossible to complete a design without some discrete logic to hold the framework together.

The increase of LSI has thus created the need for efficient ways to bridge the gaps between large functional islands. Because of complexity, performance, or uniqueness, these bridges have evolved into nontrivial random-logic configurations that still rely on clusters of small- and medium-scale integrated circuits, whose fixed functions never quite fit the problem. Now Signetics Corp. has attempted to meet the need with a field-programmable logic family.

The family spans three ranges of complexity: at the low end are the field-programmable gate arrays (FPGAs); covering the middle range are the more complex logic arrays (FPLAs); and finally there are the logic sequencers (FPLSs). These last, most complex elements have built-in registers and enable the designer to proceed from state diagram directly to hardware. The family, summed up in the table on p.182, is compatible with TTL and operates from a +5-volt supply.

The devices provide a powerful and compact alternative to random logic, replacing discrete gates, wires, and connectors, with significant savings in board space,

FIELD-PROGRAMMABLE LOGIC FAMILY									
Device	Organization	Device	Inputs	Outputs[1]	Chip enable ($\overline{CE}$)	I_{CC} (max)	Delay (max)	Availability	Package[2]
FPGA	• AND/NAND	82S102 82S103	16	9 OC 9 TS	yes	170 mA	35 ns	now	N
FPLA	• AND-OR/NOR	82S100 82S101		8 TS 8 OC			50 ns	now	N, F
	• AND-OR • Self-enable output	82S106 82S107		8 OC 8 TS	no		70 ns	4Q79	N
FPLS	• AND-OR • Complement array • 6-bit state register • 8-bit output register	82S104 82S105		8 OC 8 TS	yes[3]		90 ns	4Q79	N, F

[1] OC = open collector TS = three-state
[2] N = plastic F = Cerdip
[3] $\overline{CE}$ input may be optionally programmed as preset.

power, and cost. Moreover, since all devices can be programmed and modified in the field (as programmable read-only memories can) using readily available programming equipment, the logic can be changed to meet new customer requirements or specifications, or to recover quickly from design errors—after delivery to the field—without expensive printed-circuit-board retooling.

Programming options

Depending on their complexity, members of the programmable logic family have internal AND gates, OR gates, S-R flip-flops, true or complement buffers, and exclusive-OR (EXOR) gates. Those elements can be combined to perform single-level, double-level, and sequential logic functions—all by blowing fuse links.

There are other fuse options in output structures for the entire logic family, too, since either active-high or active-low functions can be generated without additional hardware or signal delay. Finally, the family is well suited to bus-organized environments such as microprocessor systems, since all its members offer, in addition to open-collector outputs, three-state outputs whose signals are in the high-impedance state until activated by a chip-enable input.

All the logic elements perform standard logic functions that can be represented by augmenting conventional logic symbols with a few new definitions so that they can represent multiple-input gates (see "How the FPLF defines logic," p.183).

The gate arrays

The simplest member in the family is the field-programmable gate array, which performs single-level logic functions. The equivalent logic diagram for the FPGA is shown in Fig. 1. The two gate arrays currently available are open-collector (82S102) and three-state output (82S103) versions of the same array, which comprise nine NAND gates fuse-selectively connected to 16 common inputs by true/complement buffers.

Fuses in the FPGAs allow individual outputs to be complemented to AND, so that by proper manipulation of the input polarities, and by using De Morgan's theorem, AND, OR, NAND, and NOR logic functions can be easily implemented. The parts thus serve as universal logic elements that can be tailored to applications requiring random logic, as in fault monitors, code detectors, and address decoders for microcomputer systems with memory-mapped I/O.

The logic arrays

Devices performing two-level combinational logic functions are grouped into the field-programmable logic array (FPLA) category. These elements are a step up in complexity from gate arrays, capable of generating AND-OR, AND-NOR, and their De Morgan equivalents. There are at present two array types in the FPLA family, each with either open-collector or three-state outputs.

The equivalent logic diagram of the first array type, the open-collector output 82S101 (or three-state output 82S100), is shown in Fig. 2. The first level of logic in the device is made up of 48 AND gates fuse-connectable to any of 16 common inputs by true/complement buffers. The second logic level consists of eight OR gates—one per device output—each capable of being selectively coupled to any of the 48 gates. Finally, fusing options are included for generating true or complementary outputs.

The second logic array type, the 82S106/107, has nearly the same organization as the first. The exception is that an additional OR gate with fixed inputs has been added to generate an internal enable command for the output structure. That self-enable is generated whenever any of the AND gates become logically true, which occurs when the external input code matches the internal AND-gate program. In the absence of such a match, all device outputs are unconditionally disabled. The self-enable signal is available externally—the chip-enable input ($\overline{CE}$) pin on the 82S100/101 becomes an open-collector output called $\overline{FLAG}$. Because of this feature, the 82S106/107 can be viewed as a content-addressable programmable read-only memory, ideally suited to modifying data in large ROMs, as will be shown in the second part of this article.

Shared gates

Both array types benefit from the second level of logic. The advantage here is that the AND gates can be shared—OR gates can couple with up to 48 AND gates. Also, a key advantage of this arrangement over single-level logic is that it allows editing—disconnecting invalid AND terms from the OR array and replacing them with spare AND gates (Fig. 3).

Open-collector versions of both gate-array and logic-array devices can form wired-AND outputs in order to expand the number of AND gates available on a single chip. This solves the problem that is posed by applications exceeding the resources of a single device. The only restriction is that the expanded outputs have to be programmed to be active-low.

By far the most powerful members of the family are the field-programmable logic sequencers (FPLS), which add on-chip registers to arrays of AND and OR gates. The

How the FPLF defines logic

For the most part, schematic representation of logic in the field-programmable logic family follows conventional notation—the devices include AND, OR, and exclusive-OR (EXOR) gates, as well as set-reset (S-R) flip-flops and true or complement buffers. To simplify the representation of fuse-link programmabiity, however, the FPLF schematics use a matrix arrangement with cross-point coupling to represent intact fuse links.

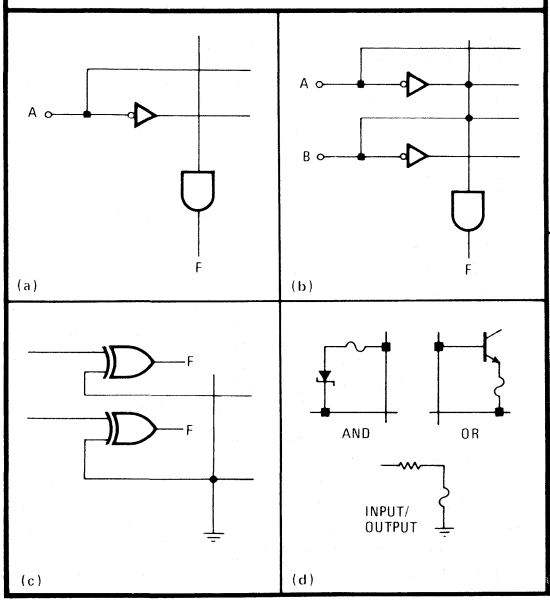
For example, (a) in the figure shows a typical input and AND gate of a gate array. The square "solder dot" represents a fixed internal connection. Both the line from input A and the line from the output of the inverter intersect the vertical input line of the AND gate; in actuality, fuse links make both connections. An intact fuse link is represented by a round solder dot. Blowing either of the fuse links will determine whether the input to the AND gate is A, or its complement, $\overline{A}$. (Leaving both fuses intact holds the output of the AND low, whereas blowing both fuses results in a "don't care" situation, an output that is independent of either input.)

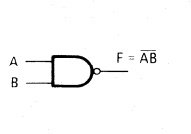
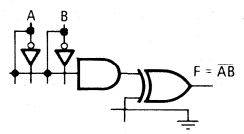
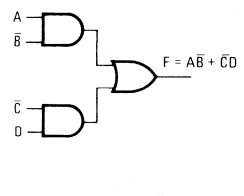
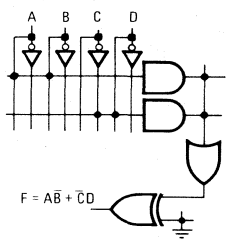
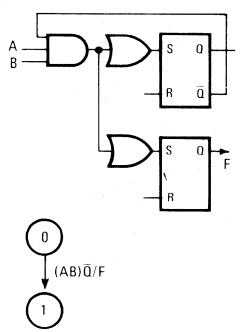
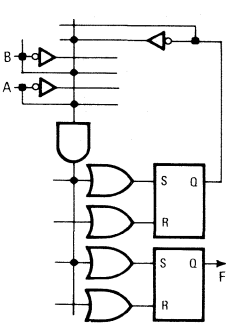
Extending the matrix and cross-point coupling approach a step further, (b) shows the configuration of a two-input AND gate. Since the input to the AND gate crosses the four lines of inputs A and B as well as their complements, the gate serves as a four-input AND while appearing to be a single-input gate. Since the members of the field-programmable logic family have 16 inputs intersecting each AND gate, the gates are actually 32-input devices; the number of inputs used is determined finally by the number of fuses left intact. Thus, in (b), solder dots (or intact fuse links) create the logical equation $F = \overline{A}B$.

The exclusive-OR gates on all outputs of the logic-family devices allow programming for either active-high or active-low output signals. As shown in (c), a fuse link grounding one of the two inputs of an EXOR gate results in an active-high output; blowing the fuse results in an output that is active-low.

The details of the fusing mechanisms are shown in (d). AND gates have a fuse in series with a Schottky diode, while OR gate fusing uses an npn transistor. The fuses for the true/complement input buffers and active-high/active-low outputs are in series with resistors.

The analogy between fixed and programmed logic is best shown by the examples in the table. The first example is typical of the single-level logic to which the gate array is applicable. The two-level logic of the second example is satisfied by the logic arrays. Finally, the registered state machine that executes the state transition of the third example is a candidate for the field-programmable logic sequencers.

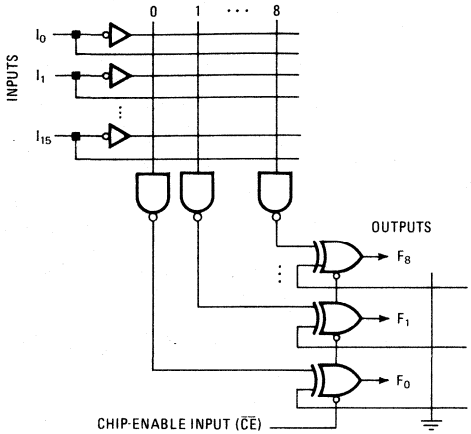


FIXED LOGIC	PROGRAMMED LOGIC
	
	
	

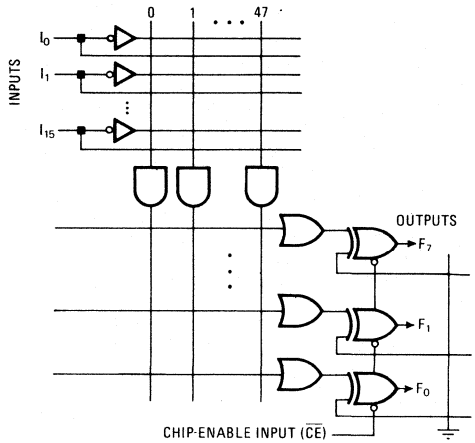
sequencers are actually self-contained state machines, since they can be programmed to perform any synchronously clocked logic sequence.

State machines, whose general structure is shown in Fig. 4a, usually take two forms: Moore machines, in which the output is a function of the present state only; and Mealy machines, whose output is a function of both the present state and the present input.

Figure 4b shows the basic architecture of the open-collector output 82S104 (or three-state output 82S105),



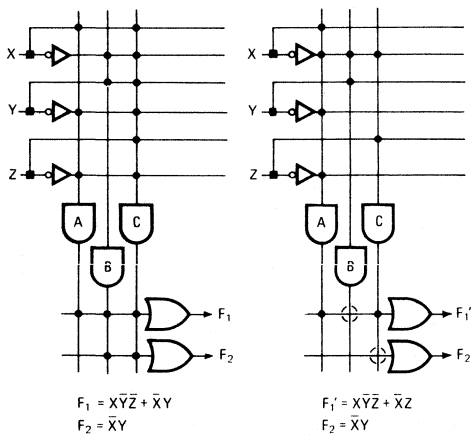
1. Gate array. The simplest device in Signetics' field-programmable logic is the gate array, capable of single-level logic. Any of 16 inputs can connect to nine NAND gates by true/complement buffers. Since outputs can be complemented to AND, manipulating De Morgan's theorem makes the device a universal logic element.



2. Double deep. The field-programmable logic array carries out two-level combinational logic. The 16 inputs couple to 48 AND gates, which in turn connect to any of nine OR gates. Either true or complement outputs are provided.

the first members of the FPLS family. With the FPLS, a user may program any logic sequence that can be expressed as a series of jumps between stable states triggered by a valid input condition I at clock time t. The number of states in the sequence depends on the length and complexity of the desired algorithm.

A typical state diagram is shown in Fig. 5. The state from which a jump originates is called the present state P, and that at which it terminates is the next state N. A jump always causes a change in state, but may or may not cause a change in the machine's output F.



3. Editing. The logic array's programmable OR gates allow sharing of AND gates, as with gate B at left. The OR array also allows easy editing of logic statements when design changes are made; note how spare gate C at right was used to modify output F_1 to F_1'.

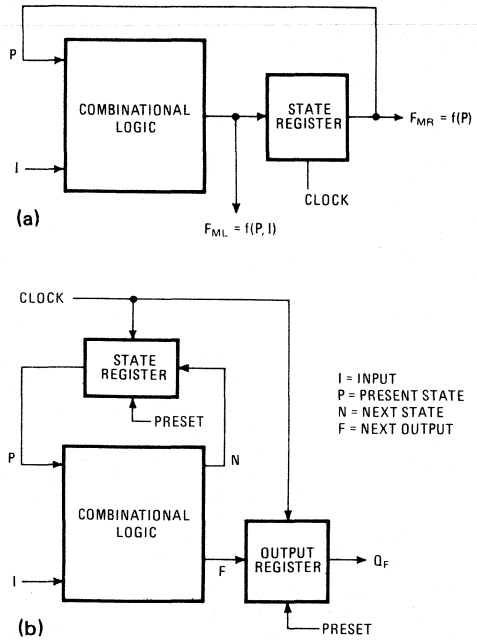
All states are arbitrarily assigned and stored in the state register, where the clock and next-state information from the combinational logic are the inputs. State jumps can occur only when transition terms are true. A transition term is, by definition, the logical AND function of the clock, present state, and valid inputs; hence, $T_n = t \cdot I \cdot P$. However, since the clock is actually applied to the state register, it may be removed from the equation. When T_n is true, a control signal is generated that, at clock time t, forces the contents of the state register from P to N and, if necessary, changes the contents of the output register.

FPLS organization

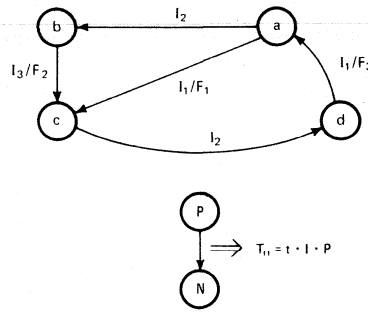
The architecture of the 82S104/105 is a natural extension of the static logic structure of the FPLA. It accepts 16 input variables and provides eight output functions. It has a 6-bit state register and an 8-bit output register; all the internal registers are automatically preset to logic 1 when power is applied. The FPLS provides for 48 transition terms, which can be selected to be either true or complementary.

A look at the equivalent logic diagram of the FPLS (Fig. 6) shows its extension of the static FPLA. The AND and OR gate arrays of the latter have been expanded to control the set and reset (S and R) inputs of six flip-flops (the state register) and to monitor the register's contents over an internal feedback path. Also, an independent 8-bit output register has been added to store output commands generated during state transitions and to hold the output constant during state sequences involving no output changes.

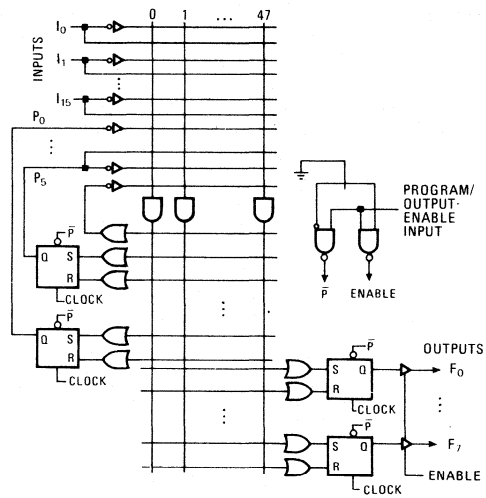
The AND array comprises 48 positive AND gates, each with 44 input connections from a set of true/complement buffers. The AND gates are used to form logic products of 16 external inputs (I_0 to I_{15}) with six present-state (P) inputs fed back from the state register. The gates are



4. State machine. A state machine (a) takes either a Mealy or Moore form. The architecture of the field-programmable logic sequencer (b) is that of a self-contained Mealy machine, where the output is a function of both the present state and the present input.



5. State diagram. Example of a state diagram (a) with four states—A, B, C, and D. I_1-I_3 are jump conditions, which trigger output changes F_1-F_3. A state change (b) gives rise to transition term T_n, which is logical AND of clock t, input I, and present state P.



6. Sequencer. The field-programmable logic sequencer has 16 outputs, 48 AND gates, and 28 OR gates, plus 14 flip-flops that serve as state and output registers. Either an asynchronous preset input or an output-enable input is available as a programming option.

therefore called transition terms because, like the transition terms in state diagrams, they issue next-state commands.

The OR array contains 28 positive OR gates, each with 48 input connections to all 48 AND gates. The outputs of the ORs drive the set and reset inputs of the 14 S-R flip-flops that are state and output registers.

The FPLS is made still more flexible by a complement array comprising a single 48-input OR gate that drives an inverter, which then feeds back into the AND array. The complement array forms a bridge between the AND and OR arrays for generating NAND functions of input-jump conditions; the user programs it in such a way as to suit each transition term.

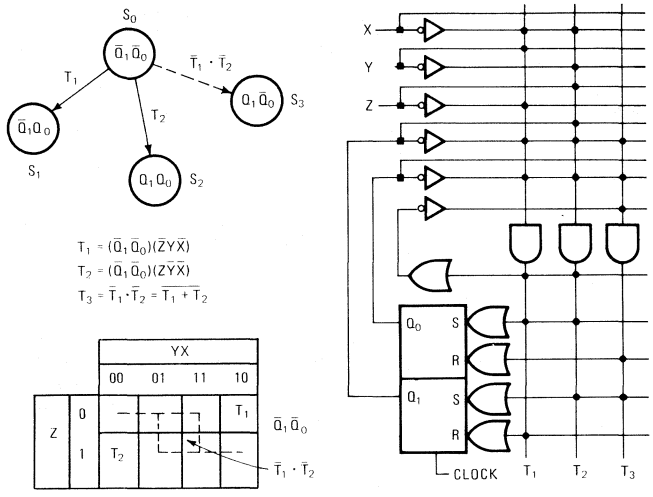
De Morgan's theorem

De Morgan's theorem to reduce logic terms can be easily implemented with the complementary array so that the most use is made of the AND gates. For example, if the transition term is $T = (Q)(\overline{X}+\overline{Y}+\overline{Z})$, where Q is the output of the state register and $\overline{X}$, $\overline{Y}$, and $\overline{Z}$ are inputs, three AND gates in the FPLS are required. However, De Morgan's theorem changes the transition term to $T = (Q)\,\overline{XYZ}$, which requires only two AND gates.

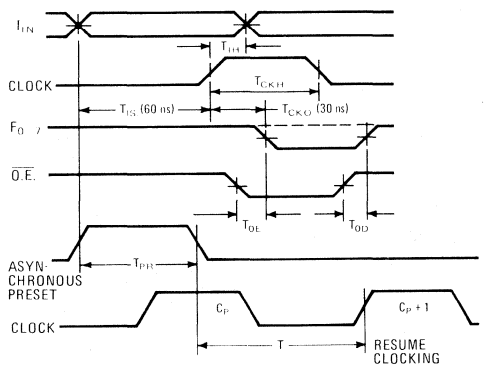
The complementary array is also an efficient means of aborting a clocked sequence in the absence of valid jump conditions. As Fig. 7 shows, considerable minimization of AND gates is possible when the detection of valid jumps involves many complements of jump functions, especially as the number of variables increases.

All clocked S-R flip-flops that make up the state and output registers offer the option of asynchronous presetting to all 1s. The 64-state total that can be represented by the state register is adequate in most cases to chart algorithms involving fewer than 48 nonredundant transitions. The register accepts next-state commands (N) from the OR array and supplies present-state information (P) to the AND array.

The output register is similar to the state register, except it has eight states for servicing eight output functions. It accepts the next-output commands F_0–F_7



7. Complementary. Use of AND gates in the FPLS is greatly reduced by the complement array, a single 48-input OR gate driving an inverter that feeds back into the AND array. In this example, the default jump from state S_0 to S_3 is reduced from three AND gates to a single gate T_3.



8. Timing. Minimum clock duration for the FPLS is 20 nanoseconds. Minimum width of preset input, which overrides clock, is 40 ns. Normal clocking resumes with the first full clock pulse following a negative clock transition after the trailing edge of the present signal.

from the OR array and then reflects its contents to the device outputs through the buffered Q outputs of each of the flip-flops. Also, as an added feature to enhance fault isolation, driving input I_0 to +10 volts will route the contents of the state register (P_0–P_5) directly to outputs F_0–F_5 without any alteration of the contents of the output register. However, the feature is not recommended for use in a normal mode of operation (as in a Moore machine). This is because it increases the device's maximum current by 5 to 10 milliamperes and thereby lowers the maximum ambient temperature rating of the package by approximately 5°C.

As a final programming option in the 82S104/105, a pin can function as either an active-high asynchronous preset (pr) or an active-low output enable ($\overline{OE}$). The output-enable function forces all outputs to logic 1 (or to high impedance in the 82S105) and is normally used when the device is sharing a bus. It does not inhibit clocking of the internal registers. The asynchronous preset option, on the other hand, is useful when the logic sequence requires an immediate state-independent return to initial conditions. The state register and output register can also be synchronously preset independently of one another by dedicating that function to one of the input variables in conjunction with a single transition term and a clock pulse.

Timing constraints

The maximum clock rate of the 82S104/105 can be inferred from its timing diagram (Fig. 8), which shows worst-case delays and setup requirements during a typical I/O cycle. Using stable external inputs as a reference, the device can be clocked after a minimum setup time of 60 nanoseconds. The next output (as well as the next internal state) will be valid 30 ns after the positive edge of the clock, giving a total I/O delay of 90 ns. Since both output enable and disable delays are also 30 ns, when the $\overline{OE}$ pin is used its signal's edge should occur prior to or coincidentally with the clock in order to avoid increasing I/O delays.

The asynchronous preset option includes a clock lockout feature that eliminates the potential hazard of spurious clocking. But, as the timing diagram shows, when using the lockout feature it is possible to miss one clock pulse, which may be prohibitive in some applications.

Applications

The second part of this article, to appear in the next issue of *Electronics*, will provide examples of applications for the gate- and logic-array devices. It will also describe in detail the development of a full-blown cartridge-tape drive controller built with a single logic-sequencer chip. The design example proceeds from flow chart to state-sequence diagram to hardware. □

Sequencers and arrays transform truth tables into working systems

by Napoleone Cavlan and Stephen J. Durham
Signetics Corp., Sunnyvale, Calif.

☐ Because of its power and flexibility, the Signetics field-programmable logic family is ideal for replacing the discrete logic normally used to interface large-scale integrated devices, as shown in Part 1 [July 5, 1979, p.181]. The examples of applications that follow show how to exploit its special features.

In designing with these gate and logic arrays and logic sequencers, the user need concern himself only with generating truth tables associated with the state diagrams or sets of Boolean logic equations that define his function. The one restriction is that he must use logic symbols corresponding to the status of fuse links.

As indicated in Fig. 1, an extra set of symbols is needed to describe all the states of FPLF gates corresponding to all combinations of blown and unblown fuse links. Once ordered into truth tables, the user-defined functions are then directly mapped onto standard program tables furnished with FPLF elements, whose fuses are then blown by a logic-type programmer. As the user gains experience, he can manipulate logic variables intuitively and can eventually implement algorithms directly on the program tables with only the device schematics for reference. (The formal step of deriving state diagrams and logic equations will not be considered here.)

Because of their simple and uncommitted structure, FPLF elements are suited to a wide variety of applications, several of them already well documented. The following examples illustrate the typical use of each logic element and match devices with applications.

Bus translator

Signetics' Instructor 50 microcomputer system is built around the 2650 microprocessor; but for compatibility with other systems and peripheral devices in the hobbyist market, it interfaces to the S100 bus, which is based mainly on 8080 microprocessor signals. Yet to carry out the seemingly unwieldy task of bus translation, only a single FPGA is needed. The gate array translates the logical combinations of timing, enable, and control signals supplied by the 2650 and its I/O hardware into control signals entirely compatible with the S100 bus definitions, as shown in Fig. 2.

The programmable feature of the FPGA is strategically invaluable in this case since the S100 bus is not yet totally standardized. The FPGA permits easy adaptation of the interface to changes in specifications, which are subject to arbitrary manipulation by manufacturers in the hobby arena.

Two-level logic

The logic arrays add a second level of combinational logic to the gate arrays, and thus another level of versatility. AND/OR combinations of the FPLAs are well suited to carrying out polynomial equations and the like, as shown in the next example.

In systems that transfer large blocks of data, a cyclic redundancy check (CRC) scheme can significantly improve data integrity. The technique appends a check word to a transmitted sequence of data, and the receiving end uses that word to check for errors. A cyclical division of the transmitted data by an industry-standard polynomial generates the CRC word; the remainder from the division forms the check word.

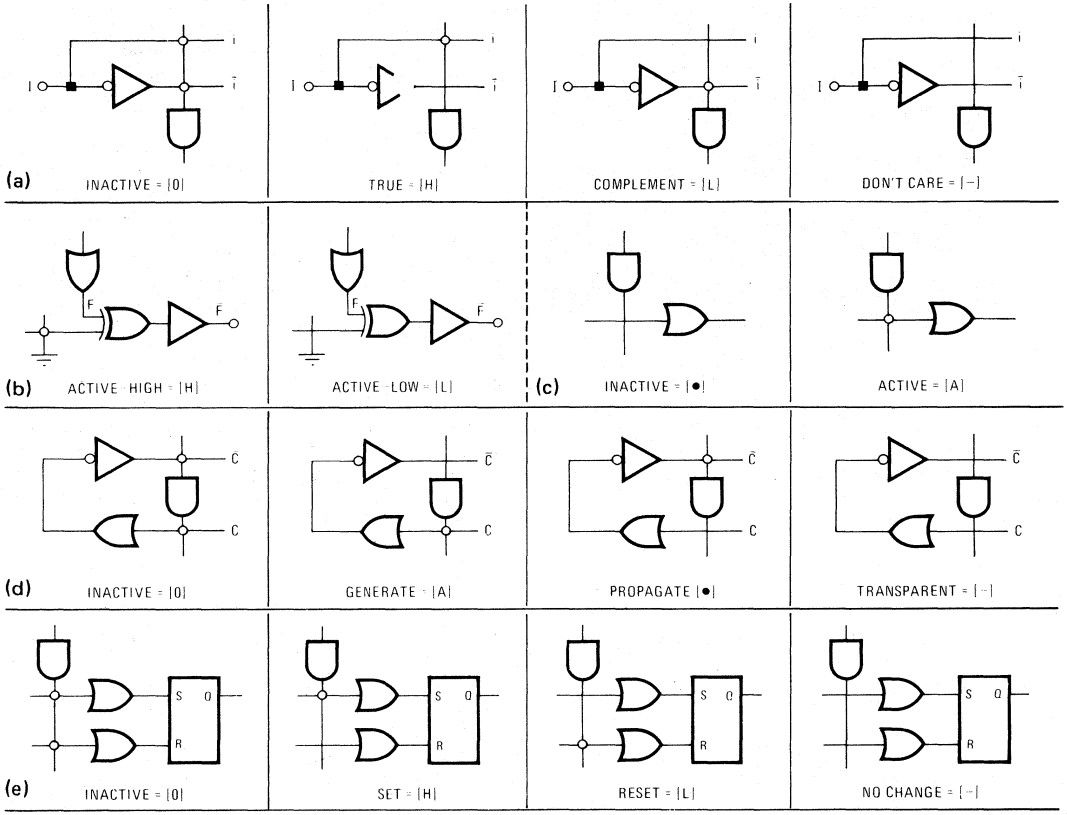
Polynomials lend themselves to serial manipulation, and serial CRC generation and checking are easy to implement. But in a multiple-line data system with parallel organization, a considerable amount of hardware may be needed for parallel-to-serial conversion. Moreover, the multiple-bit clocking for each word carries an inherent speed loss—a factor of 8 for a byte-oriented system. A parallel CRC generator-checker circuit is the answer, developed from the set of logic equations describing the function of the circuit in the form of a state machine.

The general design of the CRC circuit is shown in Fig. 3a, along with the logic equation set for the popular CRC polynomial $P(x) = x^{16}+x^{15}+x+1$. Figure 3b shows that the entire byte-wide parallel CRC generator-checker circuit can be implemented with only five chips: two 8-bit latches, two FPLAs, and an FPGA. The FPLAs contain the set of logic equations controlling the flip-flop inputs, which are expanded from EXOR form to sum-of-products form. In Fig. 3a, variables N_0–N_{15} represent the next CRC word after clocking, based on the current word B_0–B_{15} and the present input byte D_0–D_7.

CRC generation begins by driving the $\overline{\text{RESET}}$ line low to initialize the latches to zero. Pulsing the clock line then transfers the first byte of the data block in at D_0–D_7. Subsequent bytes are clocked in the same way. The cyclic nature of this design places no limit on the size of the data block that can be processed. During data transmission, the 16-bit CRC word is available at outputs B_0–B_{15} after the last data byte has been clocked in; it is appended as two check bytes to the data in the block.

Checking

The circuit is used in the check mode when receiving data containing CRC characters. The last 2 bytes in the data block received are CRC send characters. They too are clocked in and contribute to form a final receive pair of CRC characters, which, for error-free transmission, must both be zero. If an error has occurred, B_0–B_{15} will be nonzero. The FPGA will detect the nonzero condition and generate an error signal. This parallel CRC format can operate on data blocks at speeds in excess of 5.7



1. New notation. The many combinations of blown and unblown fuse links in the field-programmable logic family require new notation. The four possibilities for AND gates are shown in (a), while those for exclusive-OR outputs are in (b). The combinations for OR gates are in (c). The complement array in the logic sequencers is detailed in (d). Finally, OR gates controlling the flip-flops in sequencers are in (e).

megabytes per second.

An interesting use for the FPLA is in changing data at a few locations of a read-only memory (see "How to patch a read-only memory," p. 192).

The abilities of the field-programmable logic sequencer are well demonstrated by its use as a controller for a cartridge-tape transport. In this example, one chip replaces many—a distinct advantage if the controller is to be packed on a single-board microcomputer. Although the chip's function is complex, it can be programmed methodically and worked directly from a flow chart.

Controller routines

The controller executes fixed routines in response to status and input commands that may originate from an input/output bus or a monitoring station. Its outputs operate the velocity servo that drives the cartridge, form I/O status signals, and enable writing of data. The input and output signals of the one-chip controller are shown in detail in Fig. 4.

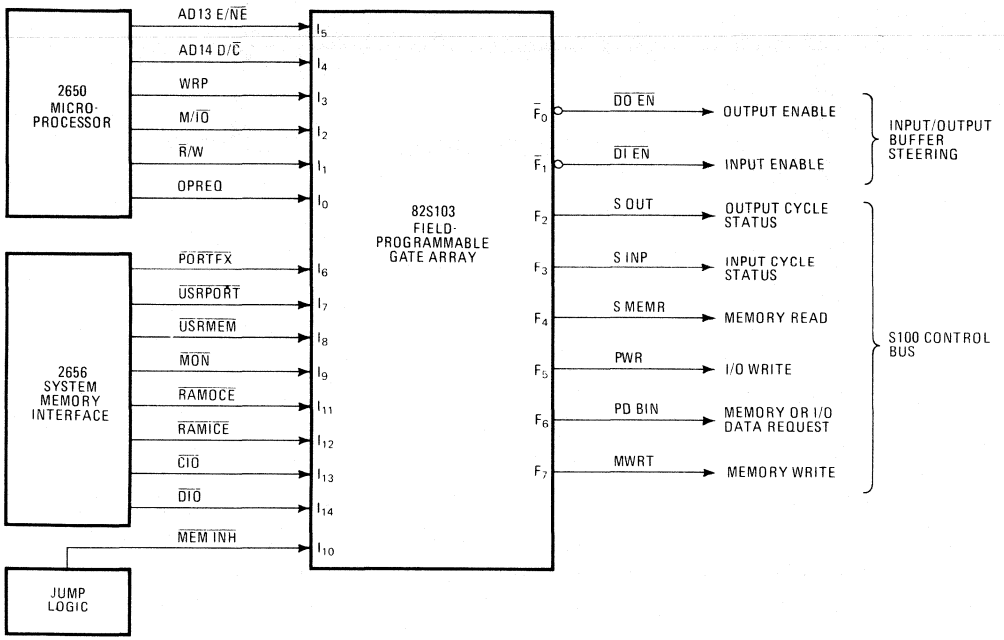
The controller carries out these eight routines:

- Move tape fast-forward.
- Move tape slow-forward.
- Move tape fast-reverse.
- Move tape slow-reverse.
- Bring tape to load point when cartridge is inserted.
- Rewind tape to load point.
- Rewind tape to beginning and eject cartridge in response to unload command.
- Rewind tape to beginning and eject cartridge in response to auto-unload true condition.

The routines could be represented concisely in a conventional Mealy state diagram, but that often obscures the actual machine function. Flow charts are more easily understood, where input variables, machine states, and output functions are given variable names. Such a chart is shown in Fig. 5.

Diagramming the flow

What would be transition terms in a Mealy state machine become true/false statements regarding the system inputs (taken one at a time) in the chart. The correlation is most obvious in the simple example in Fig. 6. The flow chart in (a) shows a conditional change from



2. Translator. Getting S100 bus signals, which are mostly 8080 microprocessor signals, out of a 2650 microprocessor calls for a field-programmable gate array. One 82S103 translates signals from the 2650 and its companion 2656 interfacing chip to the hobby bus.

state A to state B. The conditions in the flow chart's diamonds must be simultaneously satisfied for the state change to occur. The conditions take on variable names, and for this example, which arbitrarily assumes a 4-bit state register, three inputs, and two outputs, the corresponding state diagram is shown in Fig. 6b.

The transition from A to B denotes a jump from 10 (1010_2) to 13 (1101_2) and an output transition to 2 (10_2) at the next clock pulse if the combination $X_n = 4$ (100_2) is true. The transition is synthesized by forming a transition term $T = P_3\overline{P}_2P_1\overline{P}_0I_2\overline{I}_1\overline{I}_0$ and using term T at the next clock pulse to generate next-state and next-output commands for the state and output registers, respectively. For the state register, flip-flops N_0 and N_2 are set by connecting T to set lines S_o and S_2, and flip-flop N_1 is reset by coupling T to the R_1 reset line. Similarly, for the output register bit F_o is reset and bit F_1 is set by connecting T to corresponding flip-flop reset (R_0) and set (S_1) lines.

Controller conditions

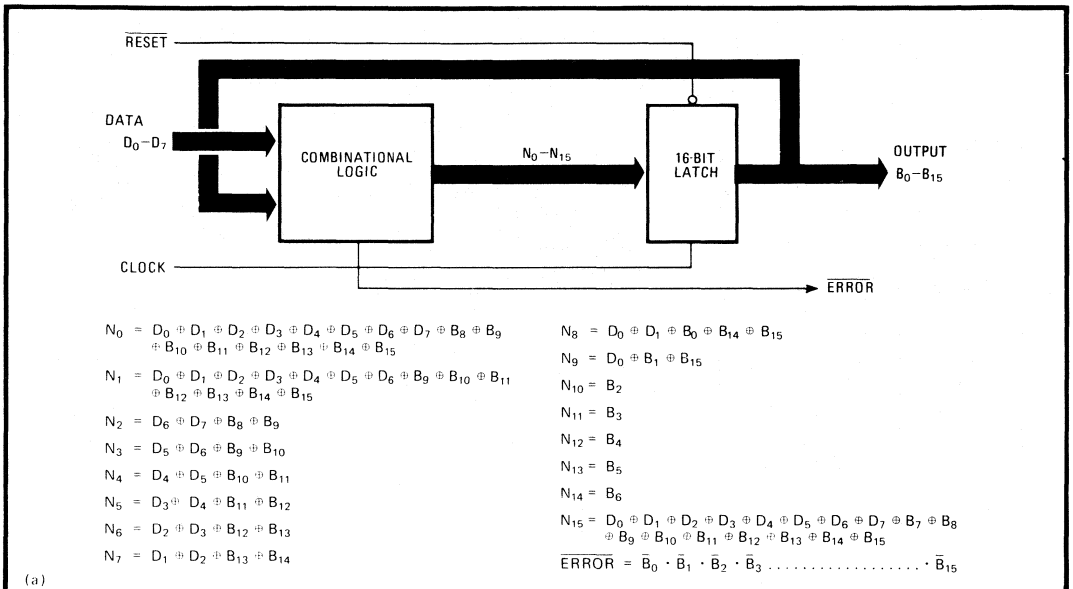
Referring again to the controller flow chart, it can be seen that whenever the tape-drive power is turned on, or when an interlock is opened, the transport must be stopped. That is achieved by an input signal to the controller called $\overline{\text{INTRDY}}$ that resets the state register with an unconditional jump to state 1 or STOP. When that occurs, all outputs on the FPLS chip become inactive, WRITE is inhibited, and speed and direction are arbitrarily set to SLOW and REVERSE. From the STOP state, operation into any mode ocurs by state and output jumps when all of the intervening conditions are simultaneously satisfied.

As an example, writing at normal speed will occur with a jump from state 1 to state 3, which requires that the following criteria be satisfied:

- The data cartridge is in place; therefore CIP is true.
- The drive has been addressed; SEL is true.
- The tape has been commanded to run; TR is true.
- The controller is not in state 6; state 6 is false.
- The tape should move slowly; therefore $\overline{\text{FAST}}$ is true (an active-low signal).
- The tape should move forward; $\overline{\text{FWD}}$ is true.

In tracing the jump between these states two things must be noted. First, the commands RWD, UNL, and TR are mutually exclusive, so that when either is true the others can be considered false or "don't care." Second, after TR = true, the condition (State = 6?) is inserted to indicate invalid jumps to states 2 and 3, which could originate from state 6 with an AUTO UNL false. Clearly, these should be avoided to inhibit honoring requests for read slow (or fast) forward while stopped at the end of the tape. So, the (State = 6?) condition is a reminder to avoid programming 6 → 2 and 6 → 3 state jumps in the FPLS. A similar argument holds for (State = 7?) and (State = 11?) conditions.

After data has been either written or read, the tape drive is commanded to stop by TR false, which causes a



$N_0 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus B_8 \oplus B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} \oplus B_{13} \oplus B_{14} \oplus B_{15}$

$N_1 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} \oplus B_{13} \oplus B_{14} \oplus B_{15}$

$N_2 = D_6 \oplus D_7 \oplus B_8 \oplus B_9$

$N_3 = D_5 \oplus D_6 \oplus B_9 \oplus B_{10}$

$N_4 = D_4 \oplus D_5 \oplus B_{10} \oplus B_{11}$

$N_5 = D_3 \oplus D_4 \oplus B_{11} \oplus B_{12}$

$N_6 = D_2 \oplus D_3 \oplus B_{12} \oplus B_{13}$

$N_7 = D_1 \oplus D_2 \oplus B_{13} \oplus B_{14}$

$N_8 = D_0 \oplus D_1 \oplus B_0 \oplus B_{14} \oplus B_{15}$

$N_9 = D_0 \oplus B_1 \oplus B_{15}$

$N_{10} = B_2$

$N_{11} = B_3$

$N_{12} = B_4$

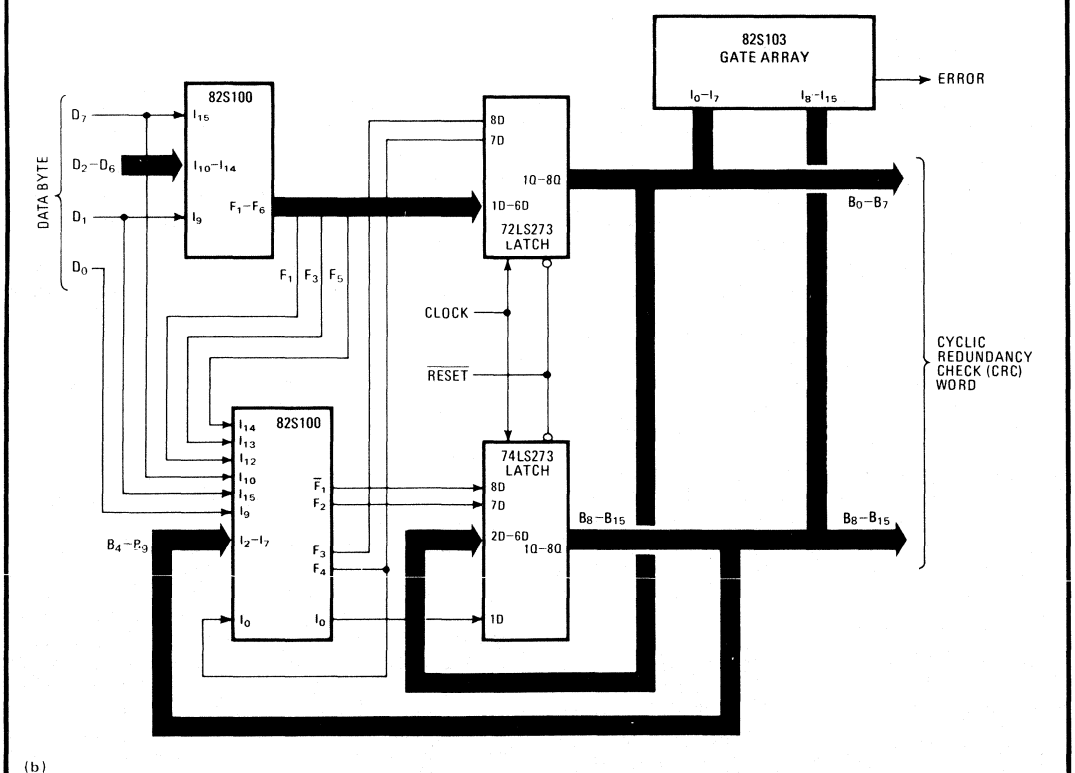
$N_{13} = B_5$

$N_{14} = B_6$

$N_{15} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus B_7 \oplus B_8 \oplus B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} \oplus B_{13} \oplus B_{14} \oplus B_{15}$

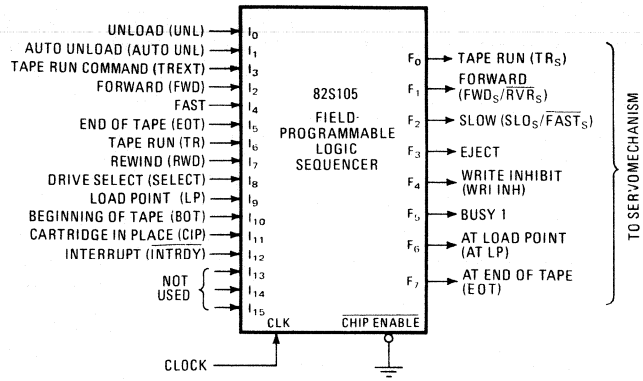
$\overline{ERROR} = \bar{B}_0 \cdot \bar{B}_1 \cdot \bar{B}_2 \cdot \bar{B}_3 \ldots\ldots\ldots\ldots\ldots\ldots \cdot \bar{B}_{15}$

(a)

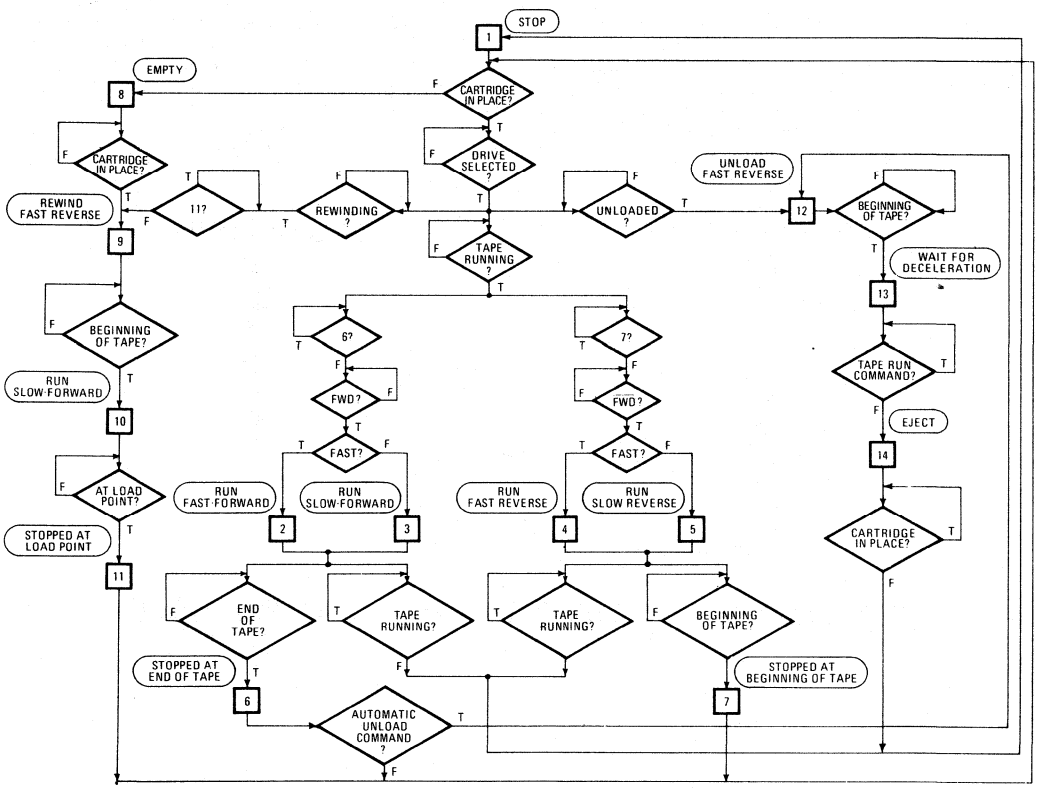


(b)

3. Error-free. The technique of using a cyclic redundancy check (CRC) word for error-free data transmission requires complex logic to generate the word (a). A pair of logic arrays, two latches, and a gate array (b) do the job, which usually requires a boardful of chips.



4. Tape controller. A field-programmable logic sequencer like this tape controller can perform extremely complex tasks. The 82S105 receives commands from an input/output bus or monitor, and provides all the necessary signals for driving the tape-transport servo-motor mechanism.



5. Goes with the flow. The first step in designing the controller is preparing a flow chart of the operation. The chart is much easier to understand than a state diagram or Mealy machine, yet provides all the information needed for programming the logic-sequencer chip.

jump from state 3 (RUN SLOW FORWARD) to state 1. By similar arguments, the tape drive can be run either fast or slow in either forward or reverse directions by jumping to states 2, 4, and 5.

When the end of tape is reached (EOT true), the tape drive is stopped. That is implemented by jumps 2 →6 or 3 →6. Once in state 6, the tape drive can no longer move in the forward direction because of the State 6 false condition preceding states 2 and 3. If AUTO UNL is true, the drive will automatically rewind (state 12), wait for tape to decelerate (state 13), eject the tape cartridge (state 14) and stop. If AUTO UNL is false, the drive must

How to patch a read-only memory

It is a shame to throw away read-only mem ɔs. But often firmware-based systems must commit control programs to large mask-programmed ROMs, only to have a design revision requiring a new program—and a new ROM. If no pin-compatible, user-programmable ROM is available, the customer could end up waiting out the 5-to-10-week turnaround time for the new mask parts—and throwing away his inventory of old ROMs.

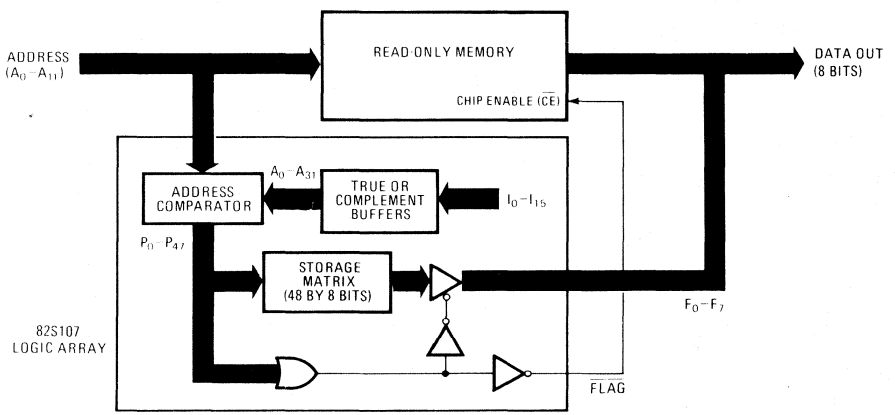
One way to save an obsolete ROM (or even PROMs—it hurts to throw them away, too) is by patching, which redirects certain addresses to an adjunct smaller memory. This can be done most efficiently with an 82S107 field-programmable logic array.

As a ROM patch (FPRP), the FPLA becomes a programmable, content-addressable PROM that continually monitors the address bus. As shown in the figure, when the FPRP encounters a match that signals a correction of data, its flag output ($\overline{FL}$) disables the ROM, and new data from the FPRP is put on the output bus. If, for example, address 750 were to be given new data A9, address 5FE were to be given 7F, and addresses OA4–OA7 were all to be reassigned B4, the FPRP would be programmed as in the table. For a 12-bit address, only inputs I_0–I_{11} are used, and the remaining four, I_{12}–I_{15} then become "don't care." (Incidentally, inputs I_0 and I_1 in the second product term are also "don't care" because they define an address block of four locations.)

The address comparator can patch up to 48 non-overlapping addresses anywhere within a memory field of 64 kilobytes. Block addressing is possible, too, using the FPRP's true or complement input buffers. Moreover, the number of addresses can be expanded by hooking several devices in parallel and wire-ANDing their flag outputs.

Since the outputs of the ROM patch primarily define a byte of memory data rather than a set of logic functions, output polarity is not controlled. Also, to maintain compatibility with the gate array, the FPRP generates its self-enable signal with a fixed multiple-input OR gate; the only disadvantage of that method is addresses (AND terms), once programmed, may no longer be deleted.

The ROM patch affords a recovery strategy effective in several design situations, including modifications of dedicated application programs, operating systems, assemblers, and monitor routines. It also permits on-site optimization of system parameters, in accordance with, say, environmental variables, and allows custom function options and product-line diversification. The customer need only allot board space next to the mask ROM for an FPRP; no parts are actually used until program changes are required after the product is in the field.



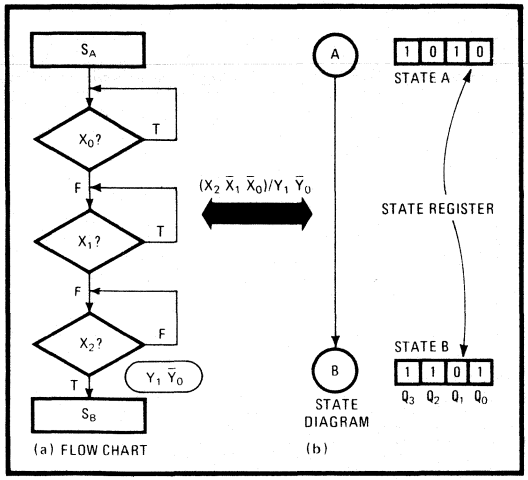
PROGRAM TABLE FOR ROM PATCH

Input address	Product term No.	Comparator input 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Active level H H H H H H H H / Output function 7	6	5	4	3	2	1	0	Output data
750	0	–	–	–	–	L	H	H	H	L	H	L	H	L	L	L	L	A	•	A	•	A	•	•	A	A9
5FE	1	–	–	–	–	L	H	L	H	H	H	H	H	H	H	H	L	•	A	A	A	A	A	A	A	7F
OA4–OA7	2	–	–	–	–	L	L	L	L	H	L	H	L	L	H	–	–	A	•	A	A	•	A	•	•	B4
		A_{11}															A_0	D_7							D_0	

wait for either a rewind command (RWD), an unload command (UNL), or reverse command (FWD).

If the tape is moved in the reverse direction until the beginning (BOT), the drive is stopped. This is implemented by a jump from states 4 or 5 to state 7. Once in state 7, the tape drive can no longer move in the reverse direction because of the state 7 false condition preceding states 4 and 5. The tape will remain stopped at the beginning until TWD, UNL, or FWD commands are given.

If no cartridge is in place (CIP false) when the tape drive is turned on, the controller will jump from state 1 to 8, and signal EMPTY. When a cartridge is installed,



6. Flow chart to state diagram. Simple transition from state A to state B is shown in flow chart (a). Three inputs (X_0,X_1,X_2) and two outputs (Y_0,Y_1) are assumed. The contents of a four-bit state register show the transition from state A (1010_2) to state B (1101_2).

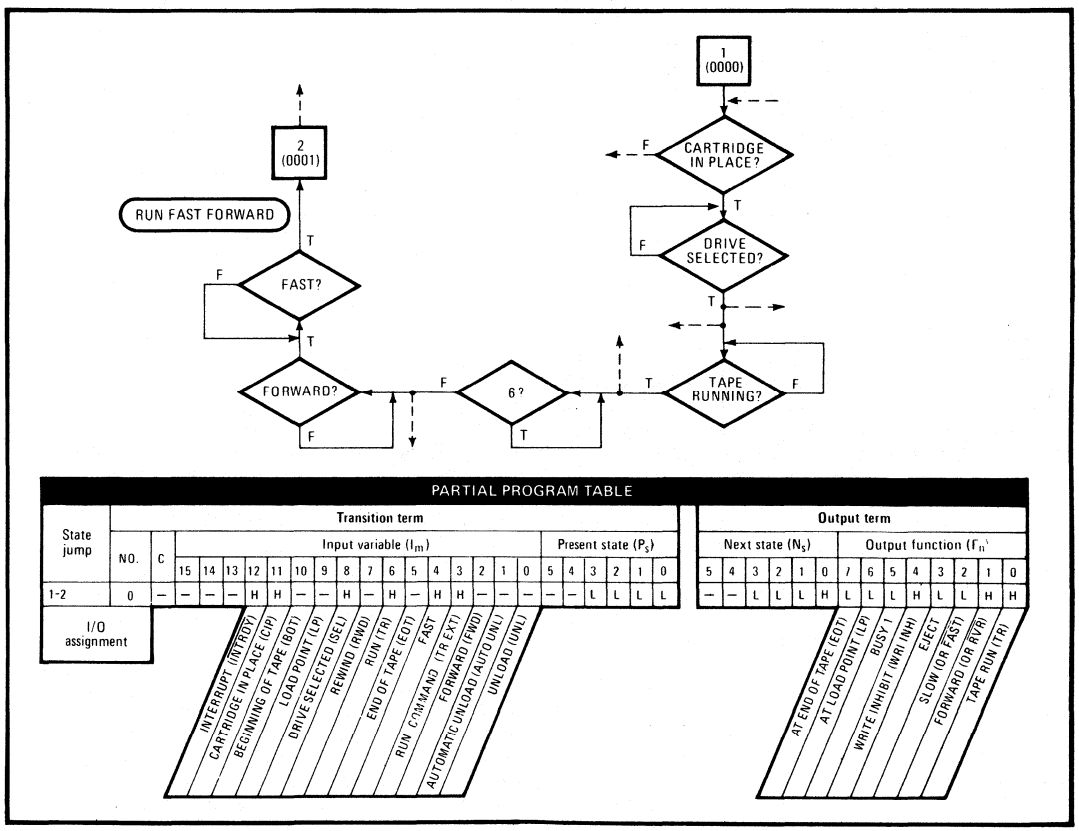
CIP = true implements a jump from state 8 to 9. In state 9 the tape will rewind in fast reverse until a BOT mark is reached. BOT true implements a jump from state 9 to 10. The tape now runs at slow speed in the forward direction until the load point (LP) is reached. LP true implements a jump from state 10 to 11 indicating STOPPED AT LP. From state 11, forward, reverse, or unload commands can be executed, but not rewind, because of the state 11 condition preceding state 9. That keeps RWD from being needlessly repeated.

State jump

A single state jump is shown in detail in Fig. 7. The transition is from state 1 to 2. In the latter, the controller is required to enter the READ FAST FORWARD routine from STOP when:

- CIP is true.
- SEL is true.
- TR is true.
- State 6 is false.
- FWD is true.
- FAST is true.

In response to this jump, the controller outputs that must change to issue the appropriate commands are run (TR),



7. Detailed state jump. The transition from state 1 (STOP) in the tape cartridge controller to state 2 (READ FAST FORWARD) is shown in flow-chart form. That part of the logic sequencer coding of the state jump is shown below, including transition and output terms.

TABLE 1. Comparison of design alternatives for tape controller.

Parameter	Field-programmable logic sequencer	Discrete logic	Monolithic Memories Inc.'s Programmable Array Logic
Chip count	1 chip	6 chips	14 chips
Circuit board area	0.84 in.^{2}	2.13 in.^{2}	3.78 in.^{2}
Power (typical)	0.60 W	1.36 W	4.8 W
Speed	90 ns/state	132 ns/state	105 ns/state
Voltage	+5 V	+5 V	+5 V
Cost (high-volume production)	$12	$14	$48

TABLE 2. Programming equipment for the field-programmable logic family.

Type	Manufacturer	Model	Field-programmable device				Availability
			Gate array	Logic array	ROM patch	Logic sequencer	
Logic	Signetics	FP-103	•				now
		FP-104		•	•	•	
	Curtis	PR-100		•			
		PR-100A		•	•		
	Data I/O	10		•	•		
Memory	Data I/O	17,19	•	•	•	•	3Q79
	Sunrise Electronics	SM100		•	•		now
			•			•	in development
Hybrid	Stag	PPX-Plus		•	•		now
			•			•	in development

forward (FWD), and fast (FAST).

The flow chart of the controller routines is complete with 14 states and 36 state jumps (including synchronous reset). As such, four state-register flip-flops sufficiently represent all states. All state jumps can be directly programmed into the chip from the flow chart. All state jumps occur on the leading edge of the clock.

The advantage of a controller built with the FPLS is best shown by a comparison to discrete logic, which would comprise PROMs, latches, and gates, using the same state diagram as for the FPLS. Table 1 compares the FPLS controller with a discrete implementation as well as with Monolithic Memories Inc.'s Programmable Array Logic chips, in several aspects.

Programming

The key to design flexibility with programmable logic is the availability of programming equipment. The need for PROMs in this equipment has led to a large number of memory programmers being offered by several manufacturers. Generally, they operate with personality card sets that meet the requirements of various PROM technologies. Suppliers have already begun developing sets compatible with memory programmers for logic devices. Hardware is expected to be available by the end of the third quarter of this year.

For the concept to work, the logic devices must be manipulated as memory chips are—by defining the desired fusing pattern in terms of an address-data relationship. Although this tends to obscure the logic function of the device, which is not visible on the program table, it is sure to provide low-cost programming equipment that can be manned by low-skilled labor.

Logic programming is another possibility, and low-cost equipment is already available from Signetics. Logic programmers allow direct entry of the logic function from the program table; no reference to the device logic diagram is necessary, and the user need not specify the status of each individual link in a device. Such programmers are more convenient for engineering use during the initial design phase, but with their high programming speed—about 10 seconds per device—can also be effective in production. Their only drawback is that they are dedicated machines and cannot program PROMs.

Some manufacturers offer a hybrid type of PROM programmer that can also be configured to do logic programming. Table 2 shows the various options available to prospective users now, or in the near future. □

Controllers using FPLAs

CONTENTS

1	Structure	197
2	Signetics FPLAs	199
3	Control systems using FPLAs	205

1 STRUCTURE

An FPLA can be regarded as a set of buffers, inverters, an AND matrix and an OR matrix with many programmable nodes, see Fig. 1. Each AND gate (with buffers and inverters) generates a product term which is a logic product of a fixed number of inputs. The latter may be true, false or don't care. For example, $A\bar{B}C\bar{D}$ and $\bar{A}BCX$ are product terms in a 4-input device, (X = don't care).

The active logic levels of inputs to the AND gates are specified by the connections on the AND matrix. The output of an AND gate is activated when the logic input combination is identical to that programmed in the product term. Outputs of the AND gates are connected to the OR gates as specified in the OR matrix. The output of an OR gate is only activated when the output of an activated AND gate is connected to an input of that OR gate.

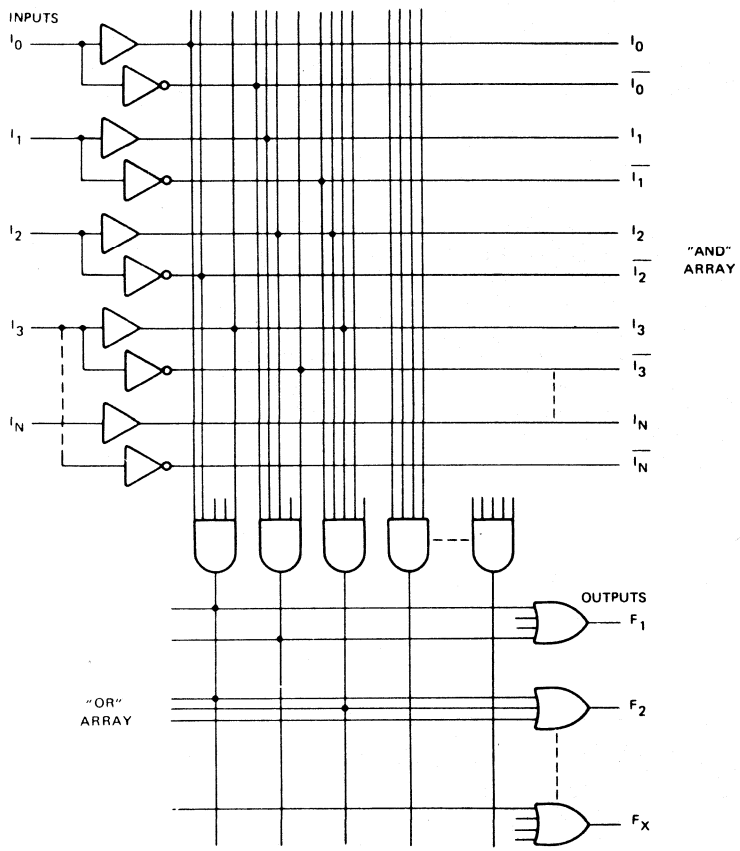


Fig. 1 Structure of an FPLA.

To increase the flexibility of the FPLA, the active output level can be programmed either HIGH or LOW. To achieve this, the outputs of the OR gates are connected to the output buffers via EX-OR gates (see Fig. 2). If a logic HIGH level is offered to the other input of this gate, the EX-OR has an inverting function and the output is active LOW. Alternatively, if a LOW level is applied to that input, the EX-OR is non-inverting and the output is active HIGH. The output buffers are activated by the chip enable input $\overline{CE}$.

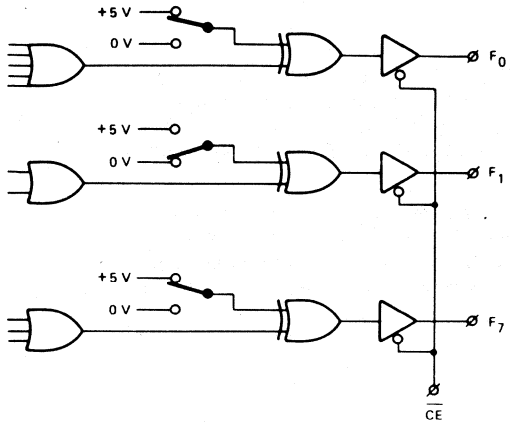


Fig. 2 Exclusive OR gates and output buffers.

Figure 3 shows a simple example of translating a logic expression into FPLA hardware. In the program table, each horizontal line represents one of the available product terms. These terms describe which input connections will activate the AND gates. In product term 0 of the example in Fig. 3, inputs 4 and 2 are connected to the AND gate via inverters (L); inputs 3 and 0 are connected via buffers (H). None of the other inputs is connected to the AND gate (–). The output of AND gate 0 is connected to OR gates 7 and 5 (A). When the input combination is identical to the programmed logic combination of product term 0, outputs 7 and 5 are activated. The active level of output 6 is programmed LOW, so output 6 is LOW if the input combination of the FPLA corresponds to one of the product terms which activates the output. The Boolean expressions for the programmed functions are

$$F_7 = I_0\bar{I}_2I_3\bar{I}_4 + I_1I_3$$

$$\bar{F}_6 = I_1I_3 + \bar{I}_0I_3$$

$$F_5 = I_0\bar{I}_2I_3\bar{I}_4\ .$$

PROGRAM TABLE ENTRIES

INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
I_m	$\bar{I}_m$	DON'T CARE	PROD. TERM PRESENT IN F_n	PROD. TERM *NOT* PRESENT IN F_n	ACTIVE HIGH	ACTIVE LOW
H	L	– (dash)	A	• (period)	H	L
NOTE Enter L, H for *unused* inputs of *used* P terms			NOTES 1) Entries independent of output polarity 2) Enter (A) for *unused* outputs of *used* P terms		NOTES 1) Polarity programmed once only 2) Enter (H) for all *unused* outputs	

NO.	PRODUCT TERM* INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL H L H L L L L L	OUTPUT FUNCTION* 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	L	H	L	–	H		A	•	A	•	•	•	•	•
1	–	–	–	–	–	–	–	–	–	–	–	–	H	–	H	–		A	A	•	•	•	•	•	•
2	–	–	–	–	–	–	–	–	–	–	–	–	H	–	–	L		•	A	•	•	•	•	•	•
3																									
4																									
5																									
45																									
46																									
47																									

Fig. 3 Example of an FPLA program table.

2 SIGNETICS FPLAs

The 82S100 (three-state outputs) and 82S101 (open collector outputs) are bipolar FPLAs which contain 48 programmable product terms (AND gates) and 8 programmable sum terms (OR gates). For each product term a logic combination of 16 input variables (HIGH, LOW or don't care) can be programmed. Each OR gate supplies an output function which can be programmed either active HIGH or active LOW.

Figure 4 shows the block diagram of a Signetics FPLA, illustrating the basic structure of buffers/inverters, AND and OR matrices, EX-OR gates and output buffers. Each of the 16 input lines of the FPLA is connected via an inverter and a buffer to the 48 AND gates. These connections are made via fusible links which form the AND matrix. An input level can be programmed by fusing the appropriate link of the AND gate. Each product term can be programmed within a field of 32 fuses; see Fig. 4, Product Terms.

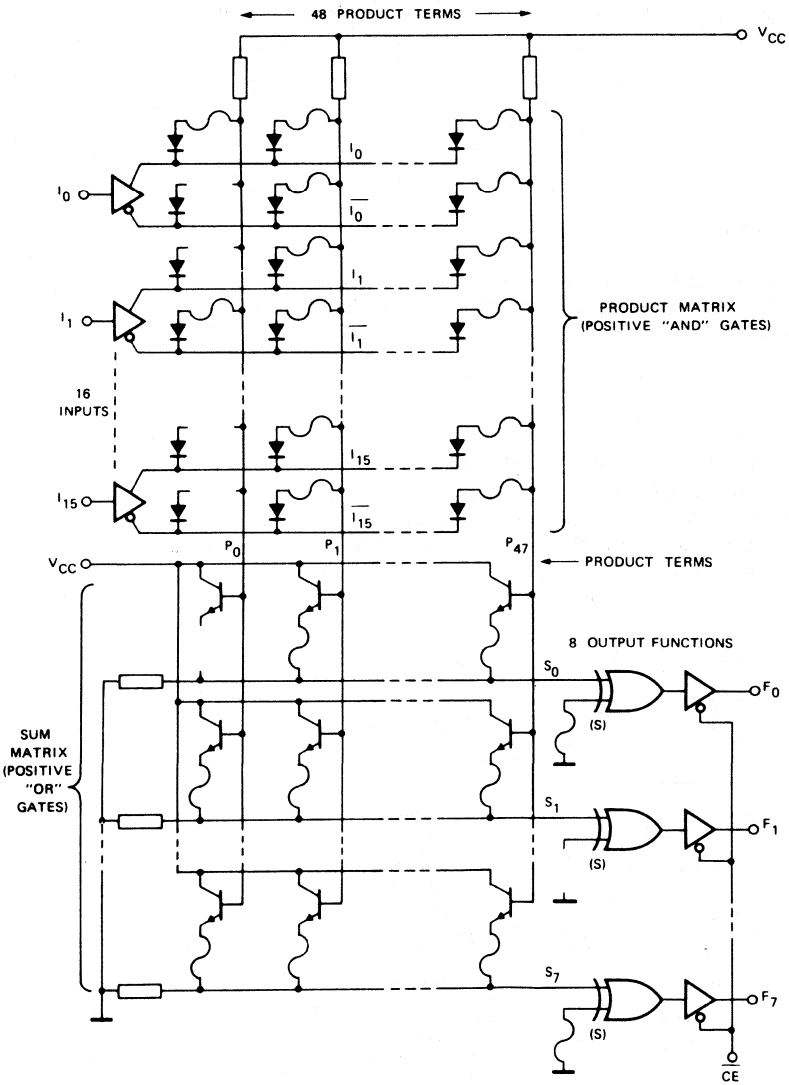


Fig. 4 Block diagram – note where links have been fused.

In product term 0, input I_0 has been programmed as true (HIGH), I_1 as false (LOW) and I_{15} as don't care. To program an input as true, the link of the inverted input must be fused. To program an input as false, the other input's link must be fused. For a don't care input, both links must be fused.

All outputs of the AND gates are connected to each of the 8 OR gates via emitter followers with fusible links. This field of fusible links is called the OR matrix. Each sum term can be programmed from a row of 48 fuses. A product term (AND gate) is disconnected from the OR gate by fusing the appropriate link. In Fig. 4, the product term P_0 has been disconnected from the OR gate of the output Function F_0.

In virgin devices all output functions have an active HIGH output level. To program an output to active LOW, the earth link of the EX-OR gate must be fused. In Fig. 4 output function F_1 has been programmed as active LOW. Unused product terms need not be programmed.

Two versions of FPLA are available. The 82S100 has three-state outputs. When enabled ($\overline{CE}$ LOW), the outputs are HIGH or LOW depending on the state of the function. In the inhibit condition ($\overline{CE}$ HIGH), all outputs have high impedance (floating). The 82S101 has open collector outputs. When enabled ($\overline{CE}$ LOW), the outputs are either LOW or floating, depending on the state of the function. When inhibited, all outputs are floating. With the open collector output there is no difference between the active HIGH and inhibit states.

2.1 EXPANSION OF VARIABLES

The FPLA has 16 inputs, 8 outputs and 48 product terms. In some applications, however, these may not be sufficient, in which case they can be expanded, generally by dividing the original program table into smaller ones.

2.1.1 Product term expansion

If the 48 product terms of a single FPLA are insufficient, more devices can be used.

When open collector FPLAs 82S101 are used, all inputs and all outputs of the required number of FPLAs should be connected in parallel. All $\overline{CE}$ inputs should be connected to ground; see Fig. 5. The outputs must be programmed active LOW to realize the wired OR function. The total number of product terms available is $48n$, where n is the number of devices in parallel.

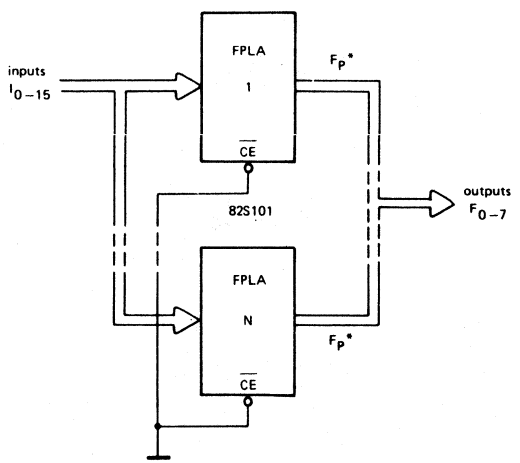


Fig. 5 Product term expansion with open collector output FPLAs.

When three-state devices are used, product term expansion cannot be achieved in the same way. As the three-state devices have an active pull-up, the outputs may be paralleled provided that only one FPLA is enabled at a time. An alternative is to divide the program table into two parts, each containing less than 48 product terms. This can be accomplished by examining one input variable and separating the product terms by placing those with HIGH levels of that variable into one table (A) and LOW levels into a second table (B). All product terms containing a don't care level for this variable must be programmed in both tables; see Fig. 6. The $\overline{CE}$ inputs of the FPLAs are controlled by the input which has been chosen to divide the program table. Figure 7 shown an example where input I_2 controls the $\overline{CE}$ line. When I_2 is HIGH the FPLA containing table A is enabled; when I_2 is LOW table B's FPLA is enabled. It should be noted that I_2 should not be programmed in any of the FPLA product terms. Product terms with up to 17 inputs can be programmed. In this way up to 96 product terms can be programmed.

P_n	INPUTS				OUTPUTS							
	I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	X	X	X	1	0	0	0	0	0	0	0	1
1	X	X	1	0	0	0	0	0	0	1	0	0
2	X	1	0	1	0	0	0	0	1	0	0	0
3	X	0	1	1	0	0	0	0	1	0	0	0
4	X	1	0	0	0	0	0	1	0	0	0	0
5	0	1	X	1	0	0	0	1	0	0	0	0
6	1	0	X	1	0	0	0	1	0	0	0	0
7	1	0	1	X	0	0	1	0	0	0	0	0
8	1	1	X	1	0	0	1	0	0	0	0	0
9	0	1	1	X	0	0	1	0	0	0	0	0
10	1	0	X	X	0	1	0	0	0	0	0	0
11	1	X	1	X	0	1	0	0	0	0	0	0
12	1	1	X	X	1	0	0	0	0	0	0	0

P-terms		INPUTS				OUTPUTS							
P_n	P_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0a	0	X	0	X	1	0	0	0	0	0	0	0	1
1a	1	X	0	1	0	0	0	0	0	0	1	0	0
3	2	X	0	1	1	0	0	0	0	1	0	0	0
6	3	1	0	X	1	0	0	0	1	0	0	0	0
7	4	1	0	1	X	0	0	1	0	0	0	0	0
10	5	1	0	X	X	0	1	0	0	0	0	0	0
11a	6	1	0	1	X	0	1	0	0	0	0	0	0

Subtable A to be stored in FPLA #1 with I_2 removed. P_{11a} can be eliminated since it is "covered" by P_7.

P-terms		INPUTS				OUTPUTS							
P_n	P_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0b	0	X	1	X	1	0	0	0	0	0	0	0	1
1b	1	X	1	1	0	0	0	0	0	0	1	0	0
2	2	X	1	0	1	0	0	0	0	1	0	0	0
4	3	X	1	0	0	0	0	0	1	0	0	0	0
5	4	0	1	X	1	0	0	0	1	0	0	0	0
8	5	1	1	X	1	0	0	1	0	0	0	0	0
9	6	0	1	1	X	0	0	1	0	0	0	0	0
11b	7	1	1	1	X	0	1	0	0	0	0	0	0
12	8	1	1	X	X	1	0	0	0	0	0	0	0

Subtable B to be stored in FPLA #2, with I_2 also removed.

Fig. 6 Dividing a 13 product term program table into two smaller tables.

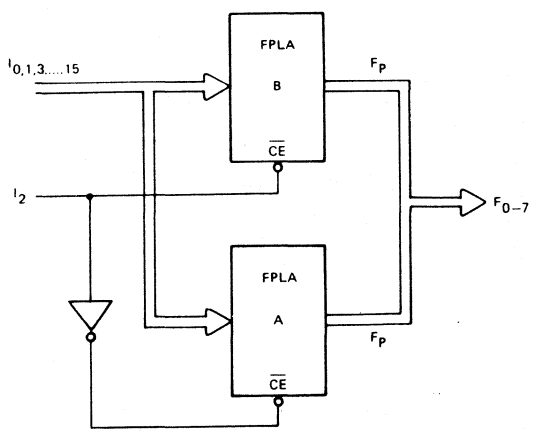


Fig. 7 Product term expansion with three-state output FPLAs.

2.1.2 Input expansion

Within one FPLA it is difficult to expand the input variables. In principle, it is possible to multiplex the input variables in order to reduce the field of input variables to 16. In most applications it is more practical to expand the input variables by using more FPLA devices as shown in Figs. 8 and 9.

In the configuration in Fig. 8, the number of input variables can be expanded to 24. FPLA C reduces the 16 inputs $I_0 - I_{15}$ to the 8 input lines $I_0 - I_7$ of FPLA E. Inputs $I_{16} - I_{23}$ are connected directly to input lines $I_8 - I_{15}$ of FPLA E. Note that there is a difference in delay time for inputs $I_0 - I_{15}$ and $I_{16} - I_{23}$.

In the configuration of Fig. 9, the number of input variables can be expanded to 32. Here the delay times of all inputs are the same but twice that of a single FPLA. For these configurations, both the three-state and open collector devices can be used.

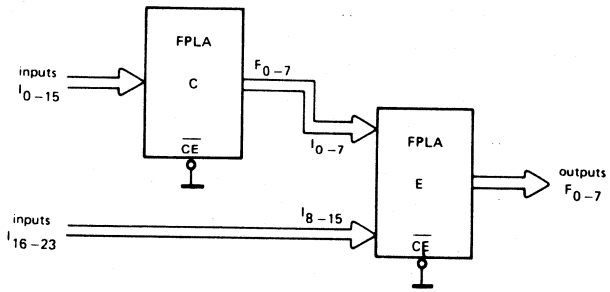


Fig. 8 Input expansion – 24 inputs.

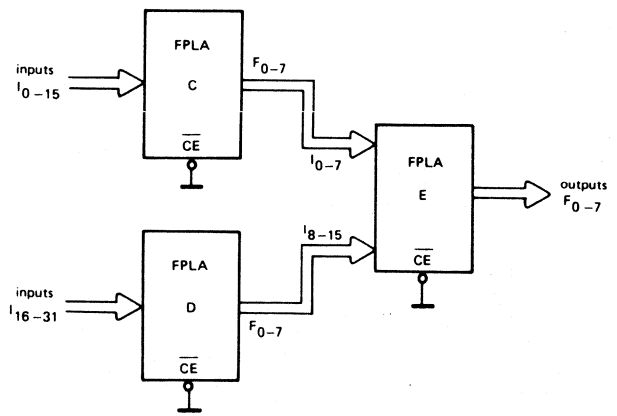


Fig. 9 Input expansion – 32 inputs.

2.1.3 Output expansion

The simplest way to expand the number of outputs is to use more FPLAs as shown in Fig. 10. The inputs of these FPLAs are connected in parallel and each FPLA controls 8 outputs. The product terms of outputs $F_0 - F_7$ are programmed in FPLA A and those of outputs $F_8 - F_{15}$ are programmed in FPLA B.

In appropriate applications, a more economical method for increasing the number of outputs is to use a 1 out of 16 decoder; see Fig. 11. The FPLA output lines $F_4 - F_7$ are decoded into the output lines $F_4 - F_{19}$. Note that only one of these outputs can be activated at a time and that their activated levels are LOW. In the inhibit state of the decoder N74154 all outputs are HIGH. When more than one of the expanded outputs must be activated simultaneously, normal decoders cannot be used. However, a PROM could be used as shown in Fig. 12. Here the number of outputs can be increased to 11. The inexpensive 32 x 8 PROM 82S123 decodes the outputs $F_3 - F_7$ from the FPLA and generates the outputs $F_3 - F_{10}$. Note that when using PROMs and decoders there is some difference in delay time between the direct outputs and the decoded ones.

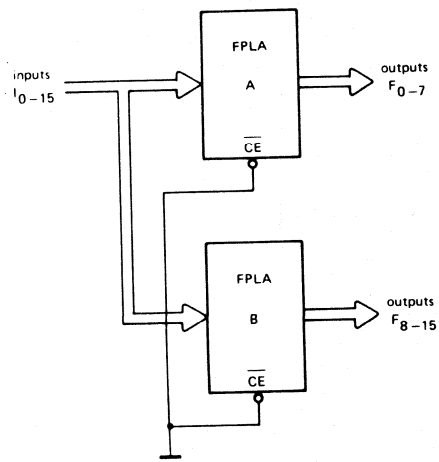


Fig. 10 Output expansion using an extra FPLA.

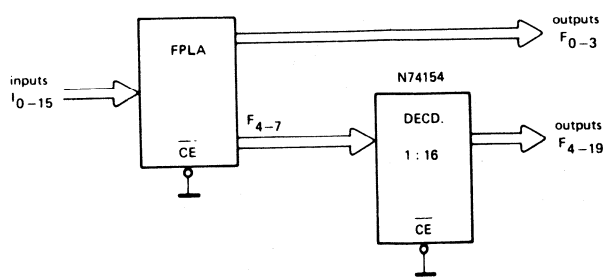


Fig. 11 Output expansion with a 1 : 16 decoder.

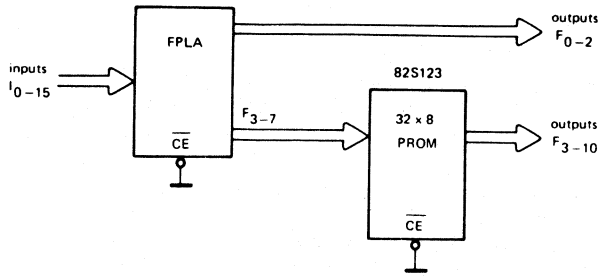


Fig. 12 Output expansion with a PROM.

2.2 EDITING FPLAs

FPLAs have inherent program editing capabilities. After programming, a number of program modifications can be incorporated as indicated in Fig. 13.

ADD	P-Term To F_p/F_p^*	YES	Program desired logic function into any unused P-Term. Blow S-Term link(s) coupling P-Term to undesired output functions.
	$I_m/\overline{I_m}$ To P-Term	NO	Delete erroneous P-Term. Add new, corrected P-Term.
DELETE	P-Term From F_p/F_p^*	YES	Blow S-Term link coupling P-Term to F_p/F_p^*.
	$I_m/\overline{I_m}$ From P-Term	YES	Blow both links coupling the input variable to the P-Term.
CHANGE	$F_p \rightarrow F_p^*$	YES	Blow EX-OR link of output to be inverted.
	$I_m/\overline{I_m} \rightarrow X$	YES	Delete $\overline{I_m}/I_m$ from P-Term.
	$I_m \leftrightarrow \overline{I_m}$	NO	Delete erroneous P-Term, and add a new P-Term.
	$F_p^* \rightarrow F_p$	NO	Use spare active-HIGH output.

Fig. 13 Editing possibilities of an FPLA.

3 CONTROL SYSTEMS USING FPLAs

In small machine control systems, an FPLA is a very powerful device by virtue of its programmability and flexibility. The FPLA can be used separately or in a structure of high level logic functions. To give some indication of the possibilities of FPLAs in control systems, a number of controllers increasing in complexity and flexibility are described in this section.

3.1 COMBINATIONAL CONTROLLERS

The basic structure of a combinational controller is a set of AND, OR and NOT functions. The outputs of such a controller are only determined by its input variables as with an FPLA. Because the FPLA has to be programmed as a sum of product terms, the logic expressions to be realized must also be written in this way. This is demonstrated with the aid of some examples.

A simple AND-OR logic function is shown in Fig. 14. The Boolean expression for this function is

$$F_7 = \bar{I}_0 I_1 0 \bar{I}_2 + \bar{I}_0 \bar{I}_1 0 \bar{I}_1 5 + I_0 I_1 .$$

This is programmed in the FPLA program table in Fig. 15; note that the unused inputs are programmed as don't care.

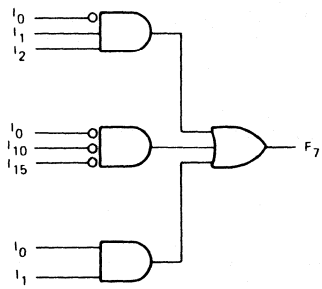


Fig. 14 Logic diagram of $F_7 = \bar{I}_0 I_1 I_2 + \bar{I}_0 \bar{I}_1 0 \bar{I}_1 5 + I_0 I_1$.

PRODUCT TERM*																ACTIVE LEVEL							
	INPUT VARIABLE															H							
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	A	•	•	•	•	•	•	
1	L	–	–	–	–	L	–	–	–	–	–	–	–	–	–	L	A	•	•	•	•	•	•	
2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	H	A	•	•	•	•	•	•	

Fig. 15 Program table for the logic diagram of Fig. 14.

A second example is the n-bit exclusive OR function. The output of an EX-OR is activated when only one input condition is true. EX-OR functions with up to 16 bits can be programmed in a single FPLA. The equation for a three-bit exclusive OR function is

$$F_7 = I_0 \bar{I}_1 \bar{I}_2 + \bar{I}_0 I_1 \bar{I}_2 + \bar{I}_0 \bar{I}_1 I_2$$

The logic diagram for this function is shown in Fig. 16. This is represented in the FPLA program table, Fig. 17. A 16-bit EX-OR function is realized by using 16 product terms and one output.

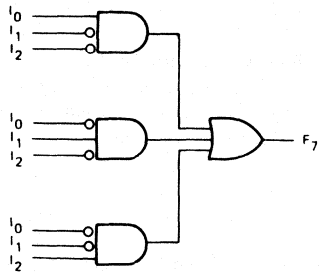


Fig. 16 Three-bit exclusive-OR function.

	PRODUCT TERM* INPUT VARIABLE																ACTIVE LEVEL H OUTPUT FUNCTION*							
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	—	—	—	—	—	—	—	—	—	—	—	—	—	L	L	H	A	•	•	•	•	•	•	•
1	—	—	—	—	—	—	—	—	—	—	—	—	—	L	H	L	A	•	•	•	•	•	•	•
2	—	—	—	—	—	—	—	—	—	—	—	—	—	H	L	L	A	•	•	•	•	•	•	•

Fig. 17 Program table for 3-bit EX-OR function.

Besides simple AND-OR functions, more complex functions such as the example shown in Fig. 18 can also be programmed in an FPLA. The logic equation of this function is

$$F7 = I7(I15 + I10 + I1I2\bar{I}3)$$

In order to program this function into an FPLA, it must be rewritten as a sum of products

$$F7 = I7I15 + I7I10 + I1I2\bar{I}3I7$$

Figure 19 shows the corresponding FPLA program table.

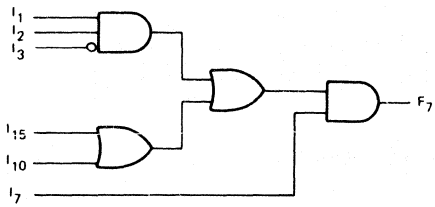


Fig. 18 Logic diagram of $F7 = I7(I15 + I10 + I1I2\bar{I}3)$.

	PRODUCT TERM* INPUT VARIABLE																ACTIVE LEVEL H OUTPUT FUNCTION*							
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	H	—	—	—	—	—	—	—	H	—	—	—	—	—	—	—	A	•	•	•	•	•	•	•
1	—	—	—	—	—	H	—	—	H	—	—	—	—	—	—	—	A	•	•	•	•	•	•	•
2	—	—	—	—	—	—	—	—	H	—	—	—	L	H	H	—	A	•	•	•	•	•	•	•

Fig. 19 Program table for $F7 = I7I15 + I7I10 + I1I2\bar{I}3I7$.

In some applications, product terms can be saved by programming the output level to be active LOW. As an example, Fig. 20 shows a function in the form of a Karnaugh map. This can be reduced into Boolean algebra in two ways

$$F_7 = \bar{I}_0\bar{I}_1 + I_2\bar{I}_3 + I_0\bar{I}_2 + \bar{I}_1 I_2$$

or

$$F_7 = \bar{I}_0 I_1\bar{I}_2 + I_0 I_2 I_3 .$$

These are represented in the logic diagrams in Figs. 21*a* and 21*b* and in the program tables Figs. 22*a* and 22*b*.

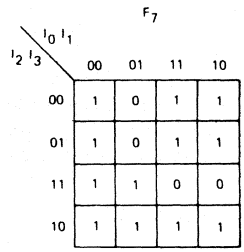


Fig. 20 A Karnaugh map.

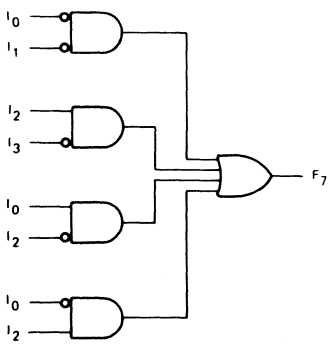


Fig. 21*a* $F_7 = \bar{I}_0\bar{I}_1 + I_2\bar{I}_3 + I_0\bar{I}_2 + \bar{I}_0 I_2$.

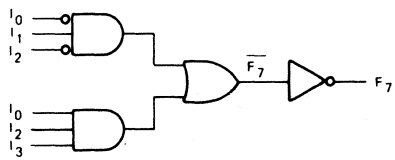


Fig. 21*b* $F_7 = \bar{I}_0 I_1\bar{I}_2 + I_0 I_2 I_3$.

NO.	PRODUCT TERM* INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL H — OUTPUT FUNCTION* 7	6	5	4	3	2	1	0
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	L	L	A	•	•	•	•	•	•	•
1	—	—	—	—	—	—	—	—	—	—	—	—	L	H	—	—	A	•	•	•	•	•	•	•
2	—	—	—	—	—	—	—	—	—	—	—	—	—	L	—	H	A	•	•	•	•	•	•	•
3	—	—	—	—	—	—	—	—	—	—	—	—	—	H	—	L	A	•	•	•	•	•	•	•

Fig. 22*a* Program table for Fig. 21*a*.

NO.	PRODUCT TERM* INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL L — OUTPUT FUNCTION* 7	6	5	4	3	2	1	0
0	—	—	—	—	—	—	—	—	—	—	—	—	—	L	H	L	A	•	•	•	•	•	•	•
1	—	—	—	—	—	—	—	—	—	—	—	—	H	H	—	H	A	•	•	•	•	•	•	•

Fig. 22*b* Program table for Fig. 21*b*.

3.2 SEQUENTIAL CONTROLLERS

In a sequential controller, the outputs are not only a function of the input variables at a given time, they are also dependent on previous input levels. Besides the AND, OR and NOT functions, the time element is introduced into sequential control systems. This time element is represented by the memory function. There are two types of sequential controller: the synchronous controller which uses clock pulses to synchronize the time when the outputs may be changed; and the asynchronous controller, the outputs of which can change independent of a clock pulse.

3.2.1 Asynchronous controllers

In simple asynchronous systems, the memory function can be obtained by a direct feedback of the output lines to the inputs. When using this configuration, it is important to avoid states in which the controller can start to oscillate, if several memory functions are used and a combination of outputs has to be changed into another combination, it is possible that one output can be faster than others. The intermediate state which appears can then influence the next state of the control system. Whenever there is a possibility of oscillations occurring, a synchronous system should be chosen. However, if there is no chance of oscillation an asynchronous system can be used successfully.

Figure 23 shows a simple memory configuration which can be expressed as

$$F7 = I_1 I_2 \bar{I}_3 + F7 I_4 .$$

To program this function into an FPLA, there must be feedback from the output $F7$ to one of the inputs, for example I_0; see Fig. 24.

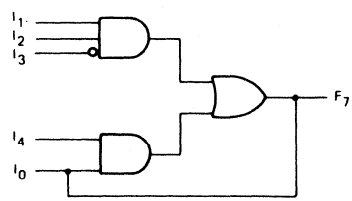


Fig. 23 A simple memory configuration.

PRODUCT TERM*																ACTIVE LEVEL							
	INPUT VARIABLE															H							
NO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	F7 0	OUTPUT FUNCTION* 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	–	A	•	•	•	•	•	•	•
1	–	–	–	–	–	–	–	–	–	–	–	H	–	–	–	H	A	•	•	•	•	•	•	•

Fig. 24 Program table for the logic diagram of Fig. 23.

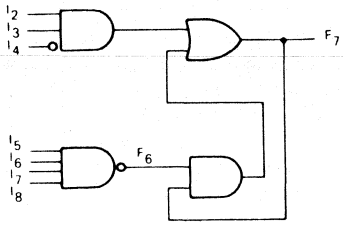


Fig. 25 $F7 = I_2I_3I_4 + F7(I_5I_6I_7I_8)$.

A more complex memory function is shown in Fig. 25. This is represented by

$$F7 = I_2I_3I_4 + F7\ (I_5I_6I_7I_8)$$

There are two ways in which this can be programmed:

(*a*) as a sum of products with one output activated;
(*b*) as a sum of products using an extra output.

In (*a*) the function can be rewritten as

$$F7 = I_2I_3I_4 + F7\bar{I}_5 + F7\bar{I}_6 + F7\bar{I}_7 + F7\bar{I}_8 .$$

The program of this function is given in Fig. 26. This method requires many product terms, especially when function F_6 (Fig. 25) has more inputs.

In (*b*), by using an extra input, function F_6 can be programmed in one product term and the whole function can be programmed in three terms. The expressions are

$$F6 = \overline{I_5I_6I_7I_8} \quad \text{and} \quad F7 = I_2I_3I_4 + F7F6 .$$

In the program table of Fig. 27, $F7$ is connected to I_0 and F_6 to I_1. Note that the active level of F_6 is programmed as LOW.

PRODUCT TERM*																	ACTIVE LEVEL							
	INPUT VARIABLE																H	L	L	L	L	L	L	L
NO.	I5	I4	I3	I2	I1	I0	I9	I8	I7	I6	I5	I4	I3	I2	I1	F7	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0	—	—	—	—	—	—	—	—	—	—	—	L	H	H	—	—	A	•	•	•	•	•	•	•
1	—	—	—	—	—	—	—	—	—	—	L	—	—	—	—	H	A	•	•	•	•	•	•	•
2	—	—	—	—	—	—	—	—	—	L	—	—	—	—	—	H	A	•	•	•	•	•	•	•
3	—	—	—	—	—	—	—	—	L	—	—	—	—	—	—	H	A	•	•	•	•	•	•	•
4	—	—	—	—	—	—	—	L	—	—	—	—	—	—	—	H	A	•	•	•	•	•	•	•

Fig. 26 Program table for the logic diagram of Fig. 25.

PRODUCT TERM*																		ACTIVE LEVEL							
	INPUT VARIABLE																	H	L	L	L	L	L	L	L
NO.	I5	I4	I3	I2	I1	I0	I9	I8	I7	I6	I5	I4	I3	I2	I1	F6	F7	OUTPUT FUNCTION*							
																		7	6	5	4	3	2	1	0
0	—	—	—	—	—	—	H	H	H	H	—	—	—	—	—	—		•	A	•	•	•	•	•	•
1	—	—	—	—	—	—	—	—	—	—	L	H	H	—	—			A	•	•	•	•	•	•	•
2	—	—	—	—	—	—	—	—	—	—	—	—	—	H	H			A	•	•	•	•	•	•	•

Fig. 27 Saving product terms by rewritting the function of Fig. 25 as $F7 = I_2I_3\bar{I}_4 + F7F6$ and $F6 = \overline{I_5I_6I_7I_8}$.

In an asynchronous control system in which several memory functions are required, the number of inputs and/or outputs may be insufficient. This problem is solved by using more FPLAs in a configuration such as Fig. 28, for example. In order to make memory functions, four outputs of both FPLAs A and B are fed back to the inputs. A total of 8 memory functions can be obtained in this way. All outputs of FPLAs A and B are connected to the inputs of FPLA C. The eight outputs of FPLA C are the outputs of the control system. The remaining 12 inputs of both FPLAs A and B are the 24 inputs of the system.

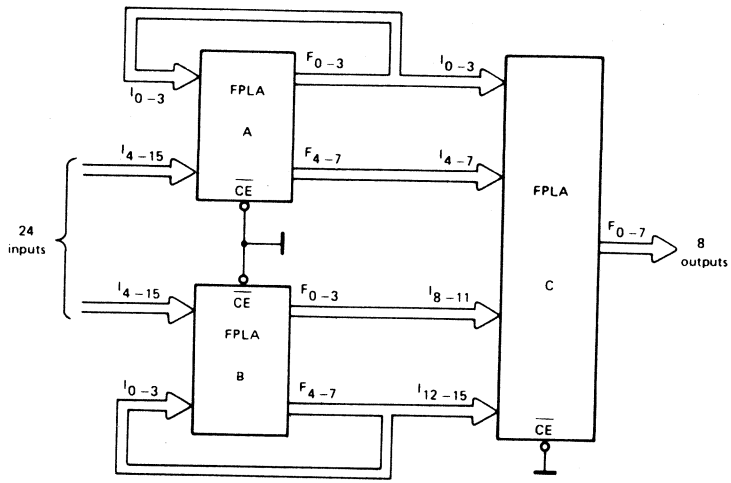


Fig. 28 Input expansion for an asynchronous control system with several memory functions.

3.2.2 Synchronous controllers

More complicated systems must be synchronized by clock pulses. Figure 29 shows a simple synchronous system. The outputs of the FPLA are clocked into the latches by the system clock. another approach to making a sequential controller is to feed back a number of FPLA outputs to the inputs via triggered flip-flops; see Fig. 30. This is the basic principle for all the synchronous sequential control systems described in this publication.

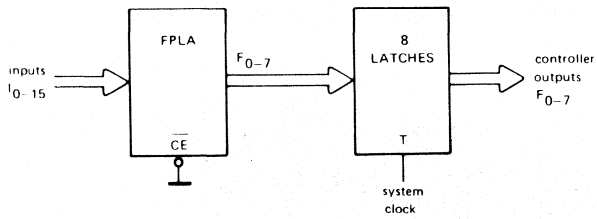


Fig. 29 A simple synchronous system.

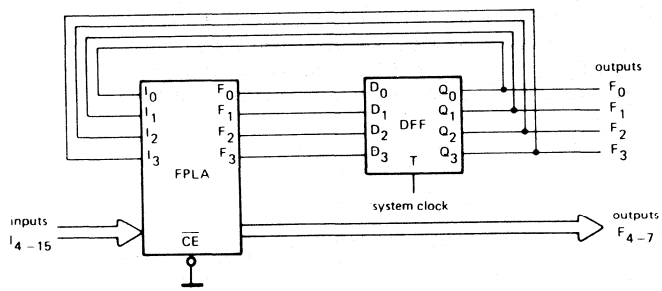
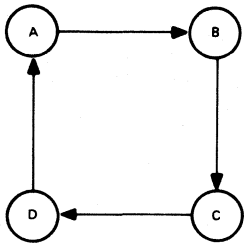


Fig. 30 Synchronous system with triggered flip-flops in the feedback paths.

The first step in designing a synchronous sequential control system is to make a state diagram or a flow chart. The whole controller function is divided into separate states, each represented by a defined combination of the levels of the fedback outputs. The state of the controller at a given time is determined by the previous state and the new input conditions. A simple example is a synchronous four-counter. The state diagram is given in Fig. 31 and program table in Fig. 32.

Each state of the control function is programmed into the product terms. When the controller is in a defined state, the next state is determined by the levels of outputs F_0, F_1, F_2 and F_3. These are connected to the inputs via clocked D flip-flops; see Fig. 30. If the controller is in an undefined state – i.e. the levels of the inputs do not correspond to a combination programmed in a product term – after one clock pulse, it is reset to the first state. Hence product term 3 need not be programmed.



	Q_3	Q_2	Q_1	Q_0
A =	L	L	L	L
B =	L	L	L	H
C =	L	L	H	L
D =	L	L	H	H

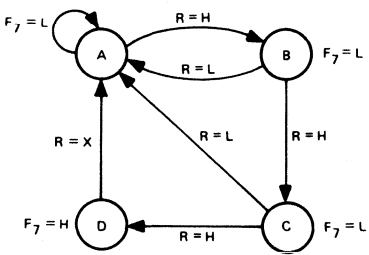
present state	next state
A	B
B	C
C	D
D	A

Fig. 31 Synchronous four-counter.

	PRODUCT TERM*																D_3 D_2 D_1 D_0 ACTIVE LEVEL							
	INPUT VARIABLE											Q_3	Q_2	Q_1	Q_0		H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L	•	•	•	•	•	•	•	A
1	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	•	•	•	•	•	•	A	•
2	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	L	•	•	•	•	•	•	A	A
3	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	•	•	•	•	•	•	•	•

Fig. 32 Program table for synchronous four-counter.

A more complex example is the same counter with decoding of the fourth state and synchronous reset to the first state. A state diagram of this function is given in Fig. 33. This function is represented in the program table Fig. 34. The conditions for R = L do not have to be programmed because if the input combination does not correspond to a product term, the controller always returns to the first state (LLLL) at the next clock pulse. In this example, the output functions only depend on the state of the controller. This is the principle of the Moore model which is a sequential synchronous circuit where the output functions are only a function of the state of that circuit.



	Q_3	Q_2	Q_1	Q_0
A =	L	L	L	L
B =	L	L	L	H
C =	L	L	H	L
D =	L	L	H	H

F_7 = D

R	present state	next state
H	A	B
H	B	C
H	C	D
X	D	A
L	X	A

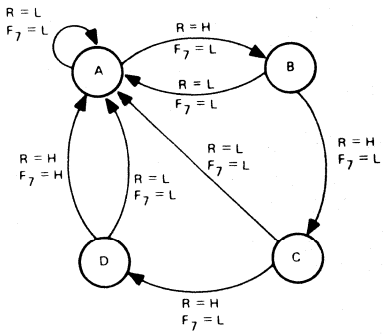
Fig. 33 Synchronous counter with reset to the first state.

	PRODUCT TERM*																	D_3 D_2 D_1 D_0 ACTIVE LEVEL							
	INPUT VARIABLE											R	Q_3	Q_2	Q_1	Q_0		H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	H	–	L	L	L	L	•	•	•	•	•	•	•	A
1	–	–	–	–	–	–	–	–	–	–	H	–	L	L	L	H	•	•	•	•	•	•	A	•
2	–	–	–	–	–	–	–	–	–	–	H	–	L	L	H	L	•	•	•	•	•	•	A	A
3	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	A	•	•	•	•	•	•	•

Fig. 34 Program table for Fig. 33.

In the last example, the output $F7$ is activated when the controller is in state D (LLHH) irrespective of the level of the reset input R. If output $F7$ is also a function of R, which is activated when HIGH, the state diagram should be changed. The level of $F7$ must be defined for each level of R; see Fig. 35.

This function is represented in program table Fig. 36. Note in particular that in state D, the level of output $F7$ depends on R – in all other states, the level of $F7$ is LOW. In this example, the outputs are functions of the state of the controller and of the levels of the inputs. This is the principle of the Mealy model.



	Q_3	Q_2	Q_1	Q_0
A =	L	L	L	L
B =	L	L	L	H
C =	L	L	H	L
D =	L	L	H	H

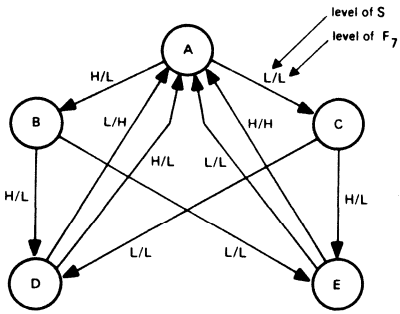
R	present state	state state	output $F7$
H	A	B	L
H	B	C	L
H	C	D	L
H	D	A	H
L	X	A	L

Fig. 35 Synchronous counter where the output is also dependent on the reset level.

PRODUCT TERM*																ACTIVE LEVEL							
	INPUT VARIABLE										R	Q_3	Q_2	Q_1	Q_0	H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	H	–	L	L	L	L	•	•	•	•	•	•	•	A
1	–	–	–	–	–	–	–	–	–	–	H	–	L	L	L	H	•	•	•	•	•	•	A	•
2	–	–	–	–	–	–	–	–	–	–	H	–	L	L	H	L	•	•	•	•	•	•	A	A
3	–	–	–	–	–	–	–	–	–	–	H	–	L	L	H	H	A	•	•	•	•	•	•	•

Fig. 36 Program table for Fig. 35.

A fourth example is a serial parity check according to the Mealy model. The parity output F_7 must be activated (HIGH level) when a three-bit serial information has an even parity. The flow and state diagrams are given in Fig. 37 and program table in Fig. 38.



	Q_3	Q_2	Q_1	Q_0
A =	L	L	L	L
B =	L	L	L	H
C =	L	L	H	L
D =	L	L	H	H
E =	L	H	L	L

present state	next state S = H	next state S = L	output F_7 S = H	output F_7 S = L
A	B	C	L	L
B	D	E	L	L
C	E	D	L	L
D	A	A	L	H
E	A	A	H	L

Fig. 37 Three-bit serial parity check.

NO.	PRODUCT TERM* INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	S 5	4	Q_3 3	Q_2 2	Q_1 1	Q_0 0	D_3	D_2	D_1	D_0 ACTIVE LEVEL H	H	H	H	H	H	H	H	OUTPUT FUNCTION* 7	6	5	4	3	2	1	0	
0	—	—	—	—	—	—	—	—	—	H	—	L	L	L	L												•	•	•	•	•	•	A	STATE A
1	—	—	—	—	—	—	—	—	—	L	—	L	L	L	L												•	•	•	•	•	A	•	STATE A
2	—	—	—	—	—	—	—	—	—	H	—	L	L	L	H												•	•	•	•	•	A	A	STATE B
3	—	—	—	—	—	—	—	—	—	L	—	L	L	L	H												•	•	•	•	A	•	•	STATE B
4	—	—	—	—	—	—	—	—	—	H	—	L	L	H	L												•	•	•	•	A	•	•	STATE C
5	—	—	—	—	—	—	—	—	—	L	—	L	L	H	L												•	•	•	•	•	A	A	STATE C
6	—	—	—	—	—	—	—	—	—	L	—	L	L	H	H												A	•	•	•	•	•	•	STATE D
7	—	—	—	—	—	—	—	—	—	H	—	L	H	L	L												A	•	•	•	•	•	•	STATE E

Fig. 38 Program table for 3-bit serial parity check.

The previous examples have described sequential controllers with only one external input variable. When more input variables are used, the state diagrams become rather complicated and so flow diagrams are more useful. Each block of these diagrams represents one or more function states, each having to be programmed as product terms as shown in the following examples.

Figure 39 shows a simple example where only unconditional functions are used with an unconditional jump from state 2 to state 0. The reset condition of the feedback latches can initiate (START) the program. This function can be programmed in three product terms (Fig. 40).

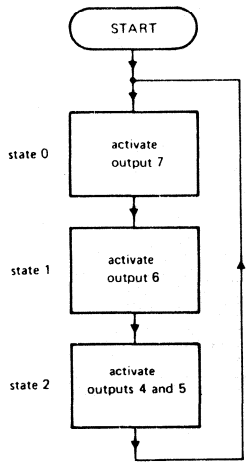


Fig. 39 Flow diagram with unconditional jump.

PRODUCT TERM*																	D_3 D_2 D_1 D_0
	INPUT VARIABLE											a_3 a_2 a_1 a_0				ACTIVE LEVEL	
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	H H H H H H H H
																	OUTPUT FUNCTION*
																	7 6 5 4 3 2 1 0
0	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L	A • • • • • • A
1	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	• A • • • • A •
2	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	L	• • A A • • • •

Fig. 40 Program table for Fig. 39.

Conditional functions can also be used. In Fig. 41 a conditional jump of one state to itself is given until a defined input combination (code) is presented.

It should be noted that when the input combination corresponds to more than one product term, the respective active output levels of all those product terms are activated. The function is programmed into product terms in Fig. 42. In this example the succeeding state is state 5, but any state could be used provided that output 2 has an active output level. In one state of the function, a conditional jump to one of several states can be made. The conditional *n*-way jump only needs one clock period. Figure 43 shows a four-way jump. If none of the three codes is presented to the inputs, the controller is switched to state 4 at the next clock pulse. If one of the codes A, B or C is presented, the next state is 5, 6 or 7, respectively. The program table for this function is given in Fig. 44.

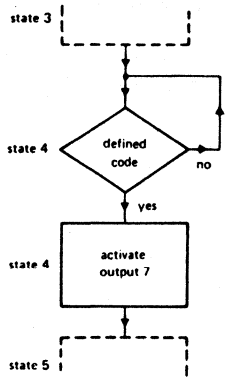


Fig. 41 Flow diagram with a conditional jump.

	PRODUCT TERM* INPUT VARIABLE																D_3 D_2 D_1 D_0 ACTIVE LEVEL							
NO	15	14	13	12	11	10	9	8	7	6	5	4	Q_3	Q_2	Q_1	Q_0	H	H	H	H	H	H	H	H
																	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0																								
1																								
2																								
3	—	—	—	—	—	—	—	—	—	—	—	—	L	H	L	L	•	•	•	•	•	A	•	•
4	—	—	H	—	L	H	H	L	H	H	—	—	L	H	L	L	A	•	•	•	•	A	•	A

defined code

Fig. 42 Program table for Fig. 41.

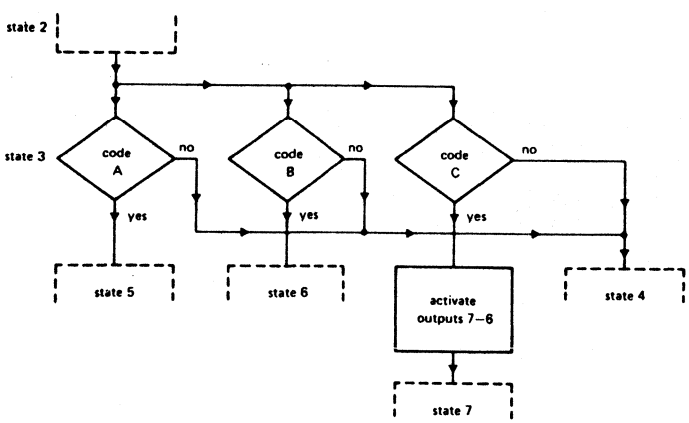


Fig. 43 Flow diagram with a 4-way jump.

	PRODUCT TERM* INPUT VARIABLE																D_3 D_2 D_1 D_0 ACTIVE LEVEL							
NO.	15	14	13	12	11	10	9	8	7	6	5	4	Q_3	Q_2	Q_1	Q_0	H	H	H	H	H	H	H	H
																	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0																								
1																								
2																								
3	—	—	—	—	—	—	—	—	—	—	—	—	L	L	H	H	•	•	•	•	•	A	•	•
4	—	—	—	—	H	H	—	L	H	L	—	—	L	L	H	H	•	•	•	•	•	A	•	A
5	—	—	—	—	L	L	H	L	H	H	—	—	L	L	H	H	•	•	•	•	•	A	A	•
6	—	—	—	—	H	L	L	H	L	H	—	—	L	L	H	H	A	A	•	•	•	A	A	A

Fig. 44 Program table for Fig. 43.

If the outputs have to be activated conditionally in a state of the controller, a conditional execute can be used as shown in the flow diagram Fig. 45. This function can be programmed into product terms as shown in Fig. 46.

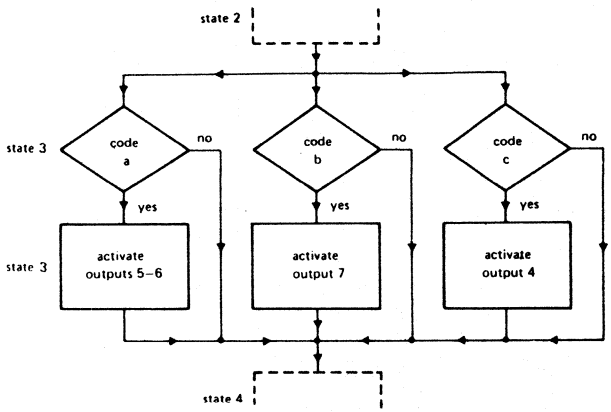


Fig. 45 Flow diagram with conditional execute.

	PRODUCT TERM*																D_3 D_2 D_1 D_0							
	INPUT VARIABLE											Q_3 Q_2 Q_1 Q_0				ACTIVE LEVEL							
NO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	H	H	H	H	H	H	H	H
																	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0																								
1																								
2																								
3	—	—	—	—	—	—	—	—	—	—	—	—	L	L	H	H	•	•	•	•	•	A	•	•
4	—	—	—	—	H	H	H	—	L	H	—	—	L	L	H	H	•	A	A	•	•	A	•	•
5	H	—	—	H	—	—	—	—	H	—	—	—	L	L	H	H	A	•	•	•	•	A	•	•
6	L	—	—	L	—	—	—	—	H	—	—	—	L	L	H	H	•	•	•	A	•	A	•	•

Fig. 46 Program table for Fig. 45.

The next example is a simple program. A man has the choice of going to work by car or by bicycle. Usually he smiles when he goes by bicycle, but sometimes he prefers to go by car. The flow diagram of this program is given in Fig. 47 and the program table in Fig. 48. Product term 0 need not be programmed because with an undefined input combination the next state is always state 0.

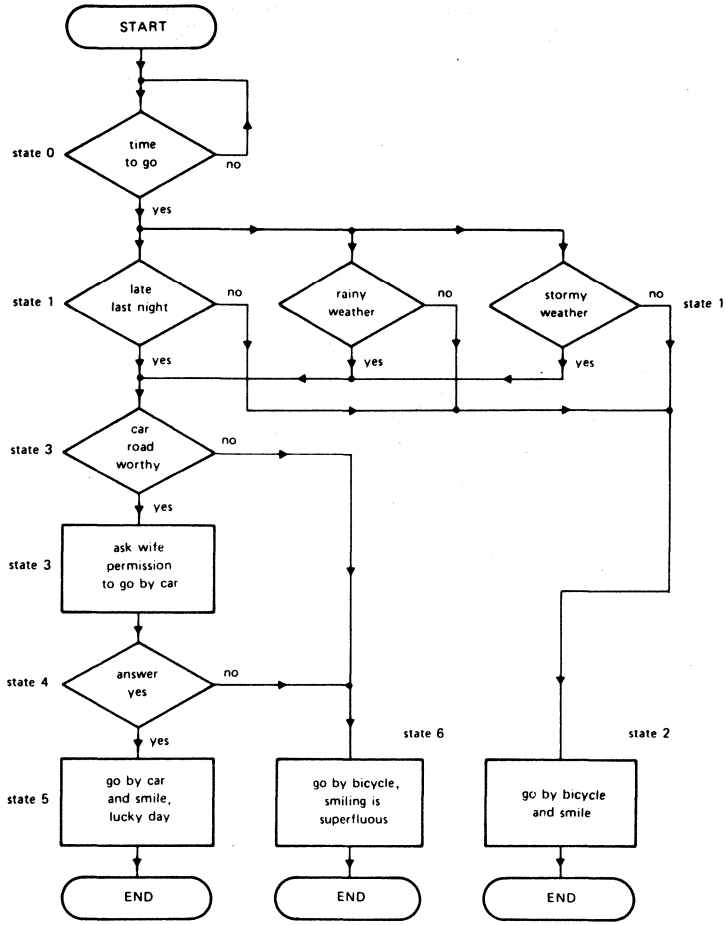


Fig. 47 Flow diagram of a simple program.

NO.	permission	road worthy	stormy weather	rainy weather	late night	time to go	Q_3	Q_2	Q_1	Q_0	bicycle	car	smile	ask wife	D_3	D_2	D_1	D_0	
	PRODUCT TERM*										ACTIVE LEVEL								
	INPUT VARIABLE										OUTPUT FUNCTION*								
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
0	–	–	–	–	–	–	–	–	–	–	L	–	L	L	L	L	•	•	•	•	•	•	•	•	
1	–	–	–	–	–	–	–	–	–	–	H	–	L	L	L	L	•	•	•	•	•	•	•	A	
2	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	•	•	•	•	•	•	A	•	STATE 1
3	–	–	–	–	–	–	–	–	–	H	–	–	L	L	L	H	•	•	•	•	•	•	A	A	
4	–	–	–	–	–	–	–	–	H	–	–	–	L	L	L	H	•	•	•	•	•	•	A	A	
5	–	–	–	–	–	–	–	H	–	–	–	–	L	L	L	H	•	•	•	•	•	•	A	A	
6	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	L	A	•	A	•	•	•	A	•	STATE 2
7	–	–	–	–	–	–	L	–	–	–	–	–	L	L	H	H	•	•	•	•	•	A	A	•	STATE 3
8	–	–	–	–	–	–	H	–	–	–	–	–	L	L	H	H	•	•	•	A	•	A	•	•	
9	–	–	–	–	–	L	–	–	–	–	–	–	L	H	L	L	•	•	•	•	•	A	A	•	STATE 4
10	–	–	–	–	–	H	–	–	–	–	–	–	L	H	L	L	•	•	•	•	•	A	•	A	
11	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	•	A	A	•	•	A	•	A	STATE 5
12	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	L	A	•	•	•	•	A	A	•	STATE 6

Fig. 48 Program table for Fig. 47.

3.2.3 Practical circuits

The diagram of a basic practical synchronous sequential control system is shown in Fig. 49. Hex edge-triggered D-type flip-flops N74174 are used in all the input lines to the FPLA including the feedback lines representing the state of the controller. Using this configuration, all output lines are only changed at the positive edge of the system clock. After this edge, all input and output lines of the FPLA are stable and so spurious pulses cannot effect the system when the clock signal is at either the HIGH or LOW level. Hence the system is fully synchronous and easy to test. In this case, four of the eight output lines are used in the feedback circuit. In general, five lines are used for this purpose and so in most applications output expansion is necessary.

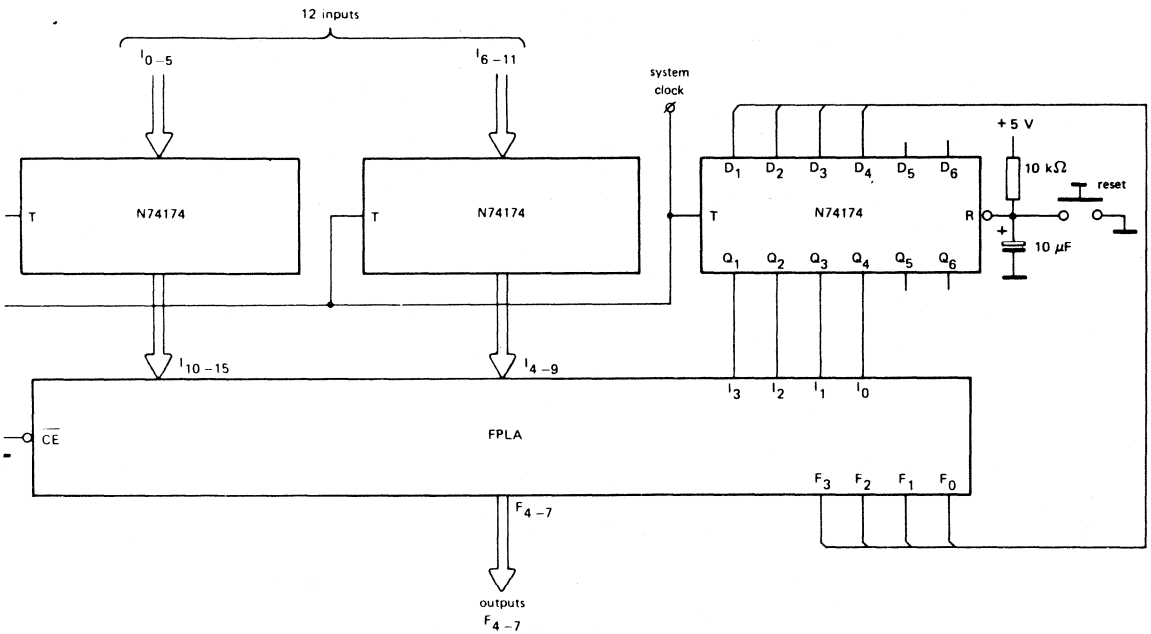


Fig. 49 Basic practical synchronous sequential controller.

Figure 50 shows an example of a controller in which 32 states can be programmed. There are 11 control inputs C_0 to C_{10} and 16 outputs, only one of which is activated at a time. In the states 0 to 15 one of the outputs O_0 to O_7 is activated and in the states 16 to 32 one of the outputs O_8 to O_{15} is activated. To avoid spurious pulses, the outputs are only activated when the clock input is LOW.

The example shown in Fig. 51 also has 32 programmable states and 11 control inputs. The output circuit comprises two inexpensive PROMs type 82S123, one for generating outputs O_0 to O_7, the other for O_8 to O_{15}. Outputs O_8 to O_{15} are only functions of the state of the controller, whereas outputs O_0 to O_7 are functions of the controller and the input conditions of the FPLA. So in any state of the controller, depending on the PROMs' programs, up to 16 outputs can be activated simultaneously.

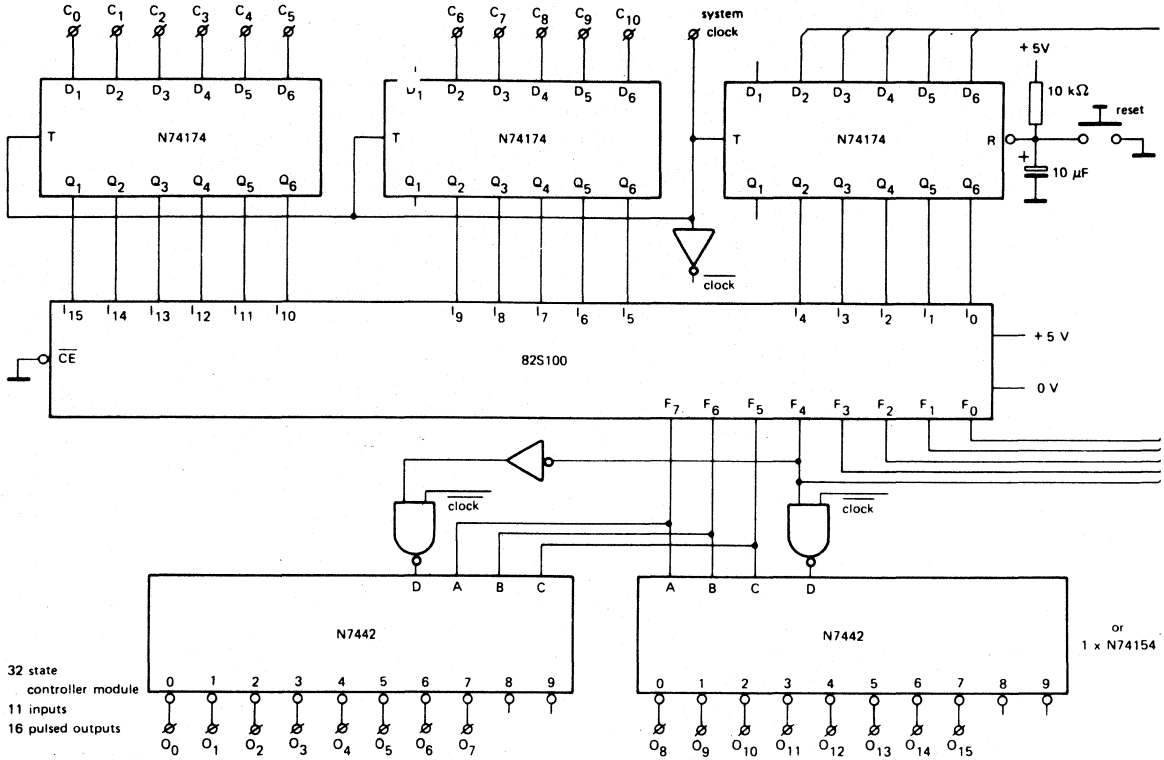

N74174
82S100
N7442
system clock
clock
+5V
10 kΩ
reset
10 μF
+ 5 V
0 V
or
1 x N74154
32 state
controller module
11 inputs
16 pulsed outputs

Fig. 50.

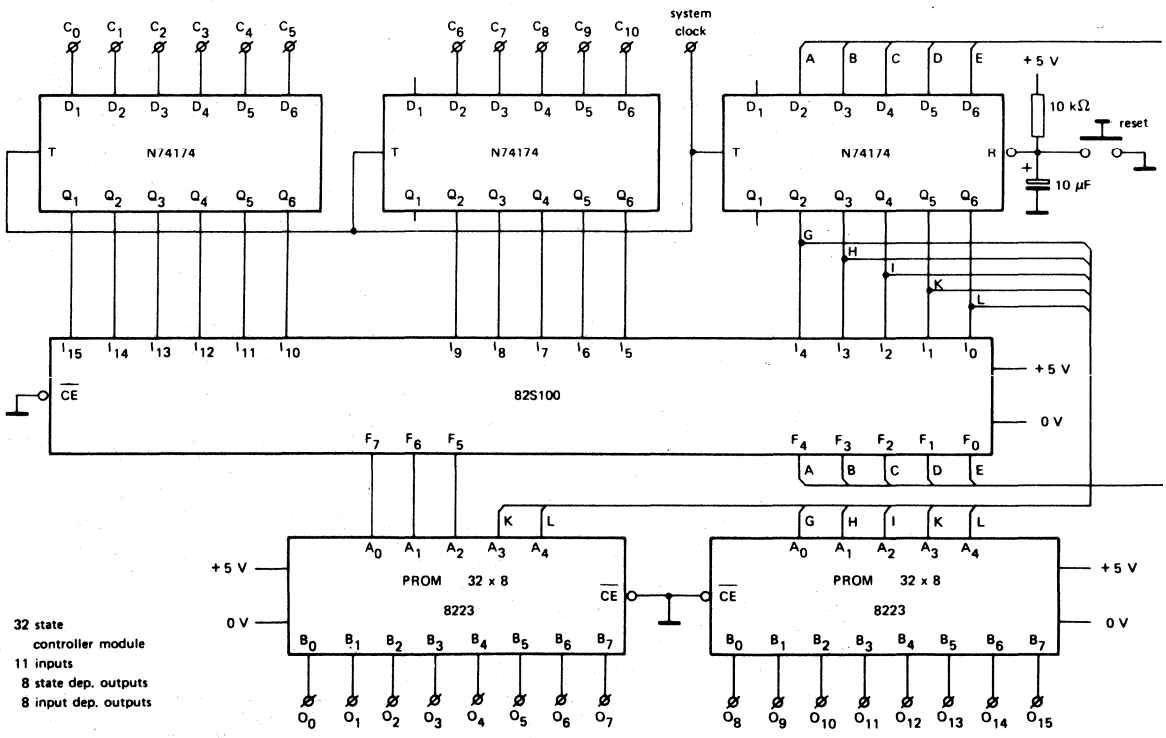
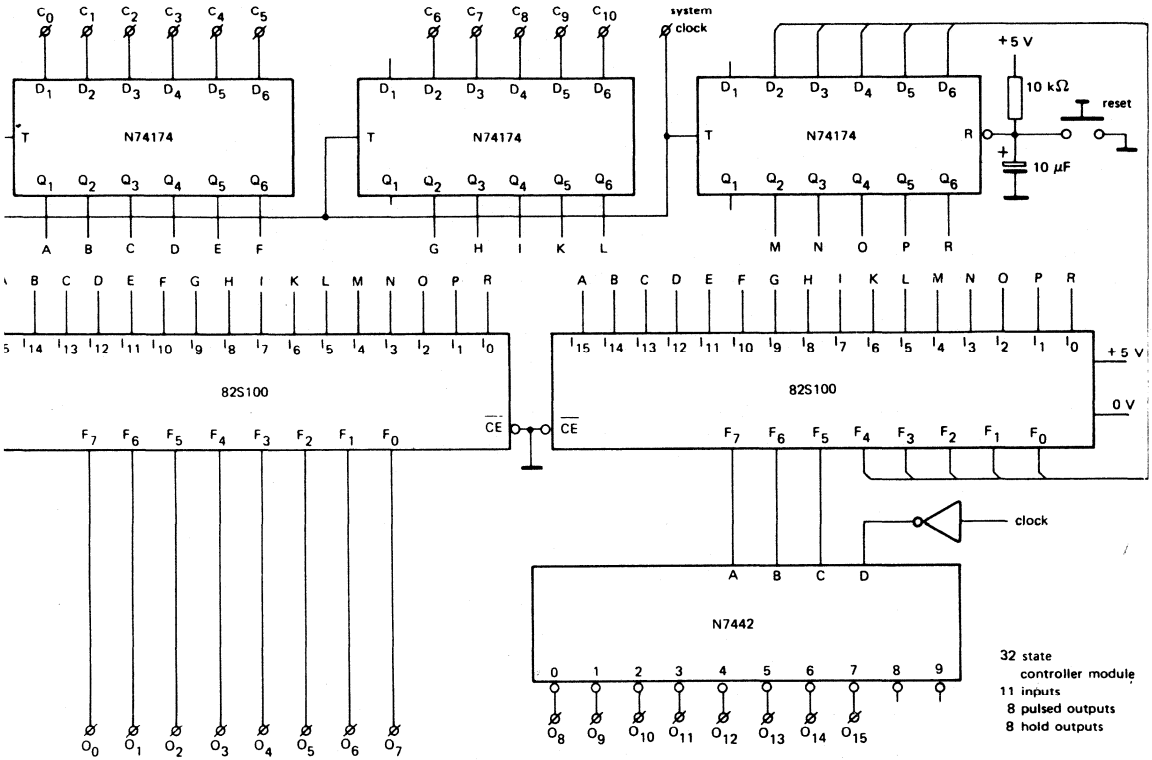

N74174
82S100
PROM 32 x 8
8223
system clock
+5 V
10 kΩ
reset
10 μF
0 V
32 state
controller module
11 inputs
8 state dep. outputs
8 input dep. outputs

Fig. 51.

In the example shown in Fig. 52, which has the same input circuit as Figs. 50 and 51, two FPLAs with the same input conditions are used. In one FPLA the states of the controller are programmed. The outputs of this FPLA are decoded and generate the outputs $O8$ to $O15$, only one of which can be activated at a time. All outputs of the other FPLA are a function of the state of the controller and the input conditions of the FPLA. These outputs can all be activated simultaneously.



52.

Single chip multiprocessor arbiter using the Signetics 82S105 FPLS

In multiprocessor environments there are considerable savings to be made through sharing system resources. If each processor must support its own bus structure, I/O devices and bulk storage medium, system cost can be very high. In the configuration shown in Fig. 1, all processors share a common communication bus plus a number of system resources.

Since every processor must use the common system bus to communicate with its peripherals, a priority structure that resolves simultaneous processor bus requests into a single bus grant is required. In addition to making request-grant transactions, transient bus contention due to grant switching must be removed by inserting precise guard times between bus grants.

Signetics' Field Programmable Logic Sequencer provides a convenient and cost effective means of implementing a synchronous arbiter to perform these tasks, within a single chip.

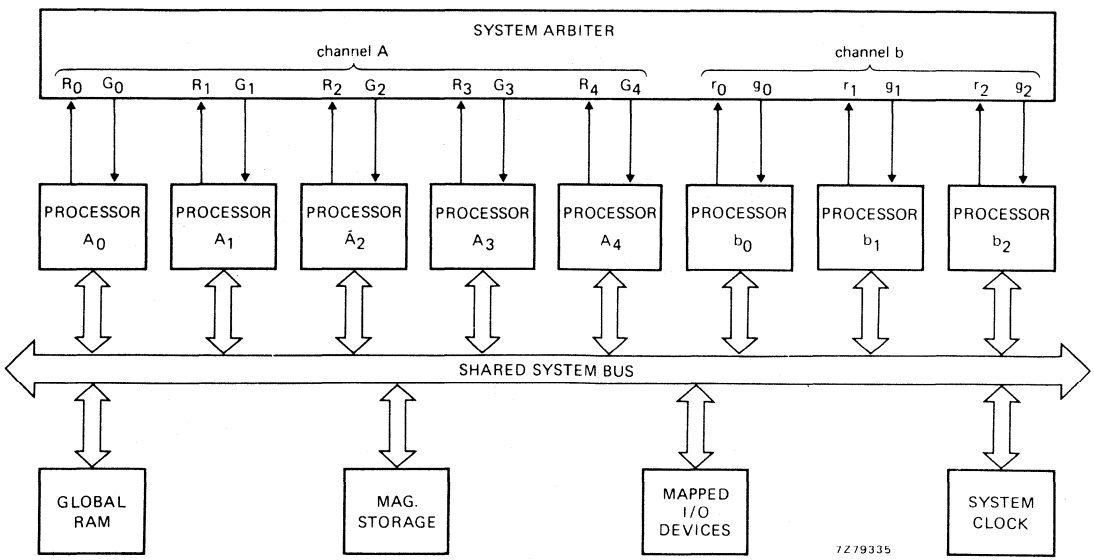


Fig. 1. Typical multiprocessor system structure.

Arbiter structure

Within a multiprocessor system, two general classes of processors may be recognized: priority A and priority b. Priority A processors have the highest request priority and must only compete with other priority A processors for bus control. The arbiter must issue A grants in a manner that prevents any priority A processor from locking out another priority A processor. To enable this, the priority A rules implemented here use a Last Granted Lowest Priority (LGLP) ring structure. After an A processor has completed a bus related task, his next arbitrated request priority will be lowest in the A request group. The previously second highest priority A processor will then have the highest priority. The net effect is that every priority A processor will take a turn at having highest priority. Priority A processors are typically ones that perform real time operations or vital system tasks. Priority b processors are lower in priority than the As and may only be granted system control when no A requests are pending; b processors usually perform background tasks. Within the b group, further ordering exists such that each b processor has a fixed priority position.

Plumber (Ref. 1), Pearce (Ref. 2), Hojberg (Ref. 3) present asynchronous techniques of arbiter implementation. These methods all have hard-wired priority rules and imprecise guard times during grant switching. As pointed out in Hojberg (Ref. 4), a synchronous state machine can be configured as a Mealy type controller to provide not only precise guard times and programmable priority rules, but also programmable input/output polarity. The state machine in Fig. 2 is made from a control PROM array and an edge triggered latch. The A and b requests and the machine's present state are used by the control PROM to determine the next A and b grants and the next state.

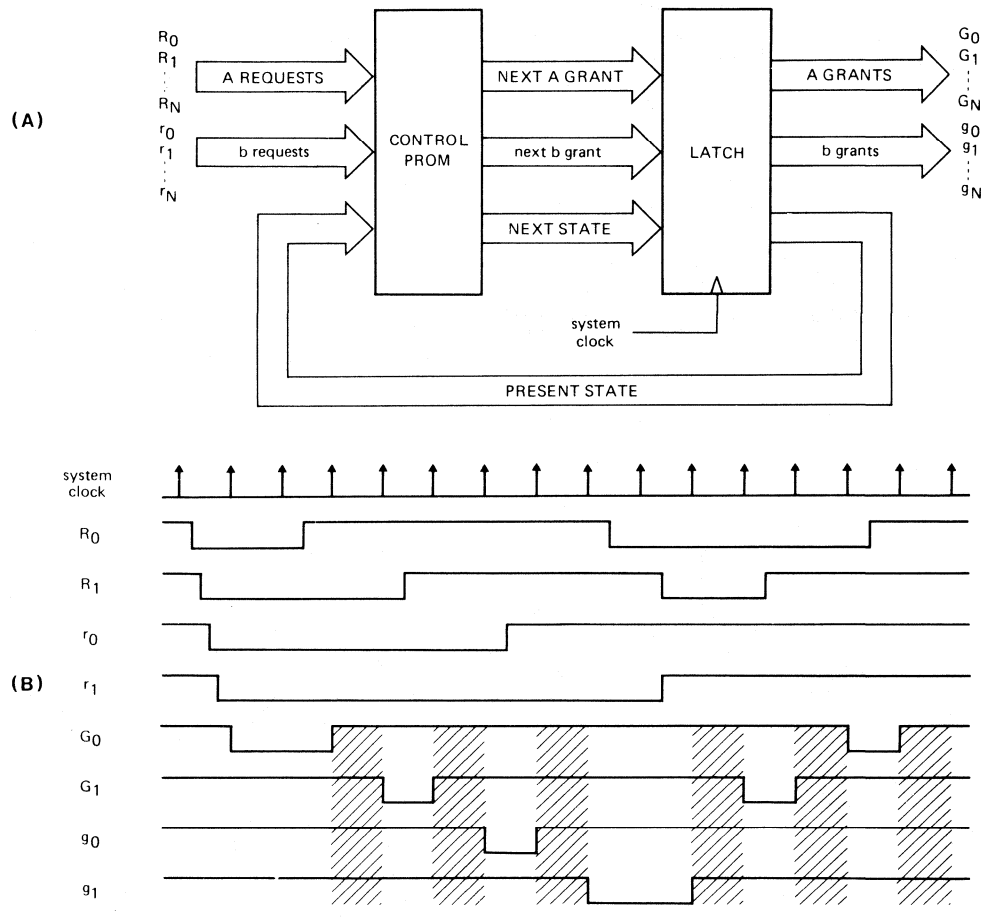


Fig. 2. Arbiter constructed from a Mealy-type state machine.
(a) A and b service requests (R_N, r_N) plus the present state determine, through the control PROM, the next grant outputs (G_N, g_N).
(b) Requests R_0, R_1, r_0, and r_1 are asserted low in the same clock sample period. The priority rules that determine the order in which the grants are issued, and the shaded guard areas are programmed into the control PROM. Note that the A and b request lines and the present state input to the PROM must have a set-up time equal to or greater than the latch set-up time plus the PROM access time.

System operation

Two machine states can be identified by inspection: a wait state and a grant state. The state machine enters a grant state as a response to a system request on either R_N or r_N. The machine will remain in this state with a single grant line asserted as long as the request remains asserted. Upon releasing the request line, the machine will pass through a single wait state before considering other pending requests. This provides a single state guard time. The requests received must meet the set-up requirement of the edge triggered latch after propagating through the control PROM. If these time considerations do not fit within a given multiprocessor structure, an input latch may be added such that the R_N and r_N lines are clocked through the latch by the system clock, thereby removing asynchronous set-up time considerations. On the basis of a state machine approach, two techniques of implementation are feasible:

1. using an architecturally advanced single IC controller, the FPLS;
2. a traditional PROM/latch configuration.

FPLS arbiter implementation

A five priority A and three priority b arbiter will be constructed such that all grant outputs will be asserted low for grants and all request inputs will be asserted low for system requests.

Brief FPLS description

The FPLS block diagram shown in Fig. 3(a) consists of a control PROM and 14 clocked S/R flip-flops. The control PROM is actually an AND-OR logic array that functions as a Content Addressable PROM: CAPROM. The CAPROM is organized as 48 words of 28 bits with 16 external input lines, and 6 internal inputs fed back from the State Register. The 28 CAPROM outputs drive the S/R inputs of the 6 bit State Register and 8 bit Output Register. Note that the state feedback path is made inside the FPLS.

The I_N and present state inputs P_S represent 2^{22} possible CAPROM input codes; 48 of these codes may be mapped in the CAPROM to provide a 28 bit register control word. As shown in Fig. 4, each input code may be specified by assigning to the variables either, Low "L", High "H", or Don't Care "–" logic states. If any input code falls logically outside the programmed codes, the CAPROM asserts a Low on all its 28 internal outputs, thereby issuing a "no change" command to the R/S flip-flops.

This is an important architectural feature because it requires that only state or output transition terms be programmed. Looping terms that neither change state nor output need not be programmed in the FPLS owing to the "no-change" characteristic of S/R flip-flops when S = R = 0. An example of this is shown in Fig. 5.

The S/R inputs of both state and output registers are specified by CAPROM outputs ("AND" functions of request inputs and present state) in the program table of Fig. 4. The corresponding next state of each bit will be set to 0 for L, 1 for H, and No Change for "–". The FPLS $\overline{PR}/\overline{OE}$ line may be assigned either Asynchronous Preset or Output Enable functions, via a user programmable option.

The entire function is integrated into a single 28 pin package designated as 82S104 (open collector outputs) or 82S105 (three-state outputs).

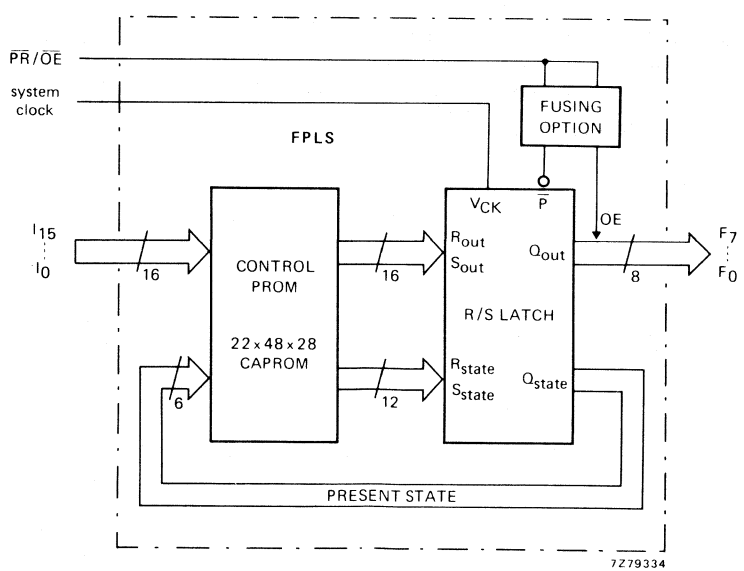


Fig. 3 Block diagram of single-chip FPLS.

No.		TRANSITION TERM																						OUTPUT TERM														
		input variable (I_N)																present state (P_s)						next state (N_s)						output function (F_n)								
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
0		–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	L	L	L	H	H	L	H	H	H	H	H	H	H	L	
1		–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	L	L	L	H	H	H	H	H	H	H	H	H	L	H	
2		–	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	H	L	L	H	H	H	L	H	H	H	H	H	L	H	H	W_0
3		–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	H	H	L	L	H	H	H	H	H	H	H	H	L	H	H	H	
4		–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	H	H	H	L	H	L	H	H	L	H	H	H	L	H	H	H	H	
5		–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	H	H	H	H	H	L	L	L	L	H	H	H	H	H	H	H	H	H	L	H	
6		–	–	–	–	–	–	–	–	–	–	–	–	–	L	H	–	H	H	H	H	H	L	L	L	H	H	H	L	H	H	H	H	H	L	H	H	
7		–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	–	H	H	H	H	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	W_1
8		–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	–	H	H	H	H	H	L	L	H	L	H	H	L	H	H	H	L	H	H	H	H	
9		–	–	–	–	–	–	–	–	–	–	–	H	H	H	H	L	H	H	H	H	H	L	L	L	L	H	H	L	H	H	H	H	H	H	H	L	
10		–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	–	H	H	H	H	L	H	L	L	H	H	H	L	H	H	H	H	H	L	H	H	
11		–	–	–	–	–	–	–	–	–	–	–	–	L	H	–	–	H	H	H	H	L	H	L	L	H	H	H	H	H	H	H	H	L	H	H	H	
12		–	–	–	–	–	–	–	–	–	–	–	L	H	H	–	–	H	H	H	H	L	H	L	H	L	H	H	L	H	H	H	L	H	H	H	H	W_2
13		–	–	–	–	–	–	–	–	–	–	–	H	H	H	–	L	H	H	H	H	L	H	L	L	L	H	H	L	H	H	H	H	H	H	H	L	
14		–	–	–	–	–	–	–	–	–	–	–	H	H	H	L	H	H	H	H	H	L	H	L	L	L	H	H	H	H	H	H	H	H	H	L	H	
15		–	–	–	–	–	–	–	–	–	–	–	–	L	–	–	–	H	H	H	H	L	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	
16		–	–	–	–	–	–	–	–	–	–	–	L	H	–	–	–	H	H	H	H	L	L	L	H	L	H	H	L	H	H	H	L	H	H	H	H	
17		–	–	–	–	–	–	–	–	–	–	–	H	H	–	–	L	H	H	H	H	L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	L	W_3
18		–	–	–	–	–	–	–	–	–	–	–	H	H	–	L	H	H	H	H	H	L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	H	
19		–	–	–	–	–	–	–	–	–	–	–	H	H	L	H	H	H	H	H	H	L	L	L	L	H	H	H	L	H	H	H	H	H	L	H	H	
20		–	–	–	–	–	–	–	–	–	–	–	L	–	–	–	–	H	H	H	L	H	H	L	H	L	H	H	L	H	H	H	L	H	H	H	H	
21		–	–	–	–	–	–	–	–	–	–	–	H	–	–	–	L	H	H	H	L	H	H	L	L	L	H	H	L	H	H	H	H	H	H	H	L	
22		–	–	–	–	–	–	–	–	–	–	–	H	–	–	L	H	H	H	H	L	H	H	L	L	L	H	H	H	H	H	H	H	H	H	L	H	W_4
23		–	–	–	–	–	–	–	–	–	–	–	H	–	L	H	H	H	H	H	L	H	H	L	L	H	H	H	L	H	H	H	H	H	L	H	H	
24		–	–	–	–	–	–	–	–	–	–	–	H	L	H	H	H	H	H	H	L	H	H	L	L	H	H	H	H	H	H	H	H	L	H	H	H	
25		–	–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	H	–	–	–	H	L	H	–	–	–	H	H	L	H	H	H	H	H	
26		–	–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	H	H	–	–	–	H	H	L	–	–	–	H	L	H	H	H	H	H	H	$W_{0\text{-}4}$
27		–	–	–	–	–	–	–	–	L	H	H	H	H	H	H	H	H	H	H	–	–	–	H	L	L	–	–	–	L	H	H	H	H	H	H	H	
28		–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	L	L	L	H	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	G_0
29		–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	–	L	L	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	G_1
30		–	–	–	–	–	–	–	–	–	–	–	–	–	H	–	–	L	L	H	H	H	L	H	H	H	H	L	L	H	H	H	H	H	H	H	H	G_2
31		–	–	–	–	–	–	–	–	–	–	–	–	H	–	–	–	L	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	G_3
32		–	–	–	–	–	–	–	–	–	–	–	H	–	–	–	–	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	G_4
33		–	–	–	–	–	–	–	–	–	–	H	–	–	–	–	–	H	L	H	–	–	–	H	H	H	–	–	–	H	H	H	H	H	H	H	H	g_0
34		–	–	–	–	–	–	–	–	–	H	–	–	–	–	–	–	H	H	L	–	–	–	H	H	H	–	–	–	H	H	H	H	H	H	H	H	g_1
35		–	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–	H	L	L	–	–	–	H	H	H	–	–	–	H	H	H	H	H	H	H	H	g_2
36																																						
										$\bar{r}_2$	$\bar{r}_1$	$\bar{r}_0$	$\bar{R}_4$	$\bar{R}_3$	$\bar{R}_2$	$\bar{R}_1$	$\bar{R}_0$													$\bar{g}_2$	$\bar{g}_1$	$\bar{g}_0$	$\bar{G}_4$	$\bar{G}_3$	$\bar{G}_2$	$\bar{G}_1$	$\bar{G}_0$	

Fig. 4 FPLS program table for priority arbiter.

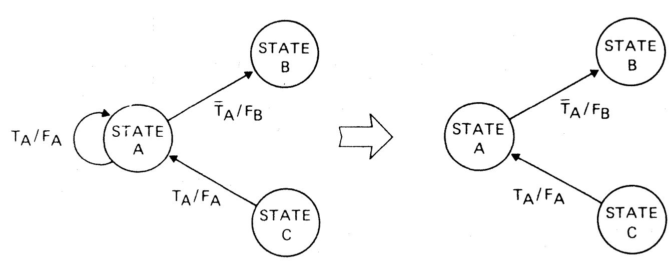


Fig. 5 FPLS state and output transitions.

State algorithm

Figure 6 displays the circular state form and all possible state transitions of the LGLP priority structure. Hex states 3B, 3C, 3D, 3E, and 3F are arbiter wait states $W_{0\text{-}4}$. In these states, processor A and b requests are monitored. Figure 6 illustrates a typical grant to processor A1 in state 07_{16}. As long as A1 asserts its request line low, the next state will be 07_{16} and the next output will remain with G_1 asserted low and all the other grant outputs asserted high. Since no change in state or grant output results from this transition, no CAPROM resources are required.

As soon as processor A1 returns his request line, R_1, to 1, a state transition is made to 3D and an output transition is made to set all outputs to 1. Since processor A1 was the last to be granted system resources, it will now have the lowest A level request priority (LGLP). In wait state W_2, the highest priority processor will be A2, second A3, third A4, and fourth A0. To maintain the LGLP rule, grant transitions must follow the state rule $G_N \rightarrow W_{(N+1)}$, and wait states, W_M, must set their A priorities so that processor A_M is highest priority. Priority decreases as one proceeds clockwise around the state ring to the lowest priority processor, A(m – 1).

When no A requests are pending, b requests may be granted. To avoid upsetting the LGLP priority rule, b grants must leave and return to the same wait state. Since the b priority structure is the same regardless of the wait state, only a single set of b transition terms are required.

For example, a grant transition to g_2 (Hex 20 – 25) can be issued only if there are no A, b0, or b1 requests pending. Given the binary wait state code 111XXX, where X represents Don't Care, a request code of 01111111 will transfer the arbiter to the grant state g_2 from any of the wait states $W_{0\text{-}4}$.

It is important to realize that in making this transition, the lower 3 state bits will not be changed – they provide the wait state return address. When r_2 returns high, 1XXXXXXX, a transition back to the previously exited wait state is made by forcing ones in the 3 most significant state bits and leaving the lower 3 state bits unchanged.

All output and state bits are initially preset to 1 through the use of the optional preset function. Grant output lines are only forced low when transitions are made to grant states, and returned to 1 when jumping back to a wait state.

The complete arbiter circuit is shown in Fig. 7(a).

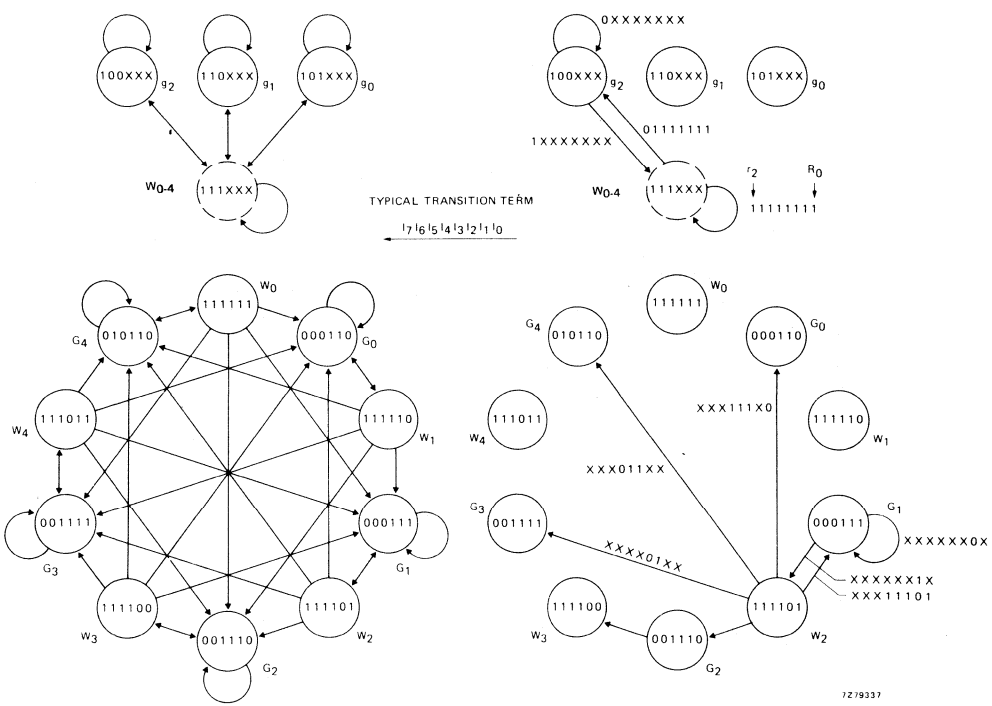


Fig. 6 State transition diagram for the priority arbiter.

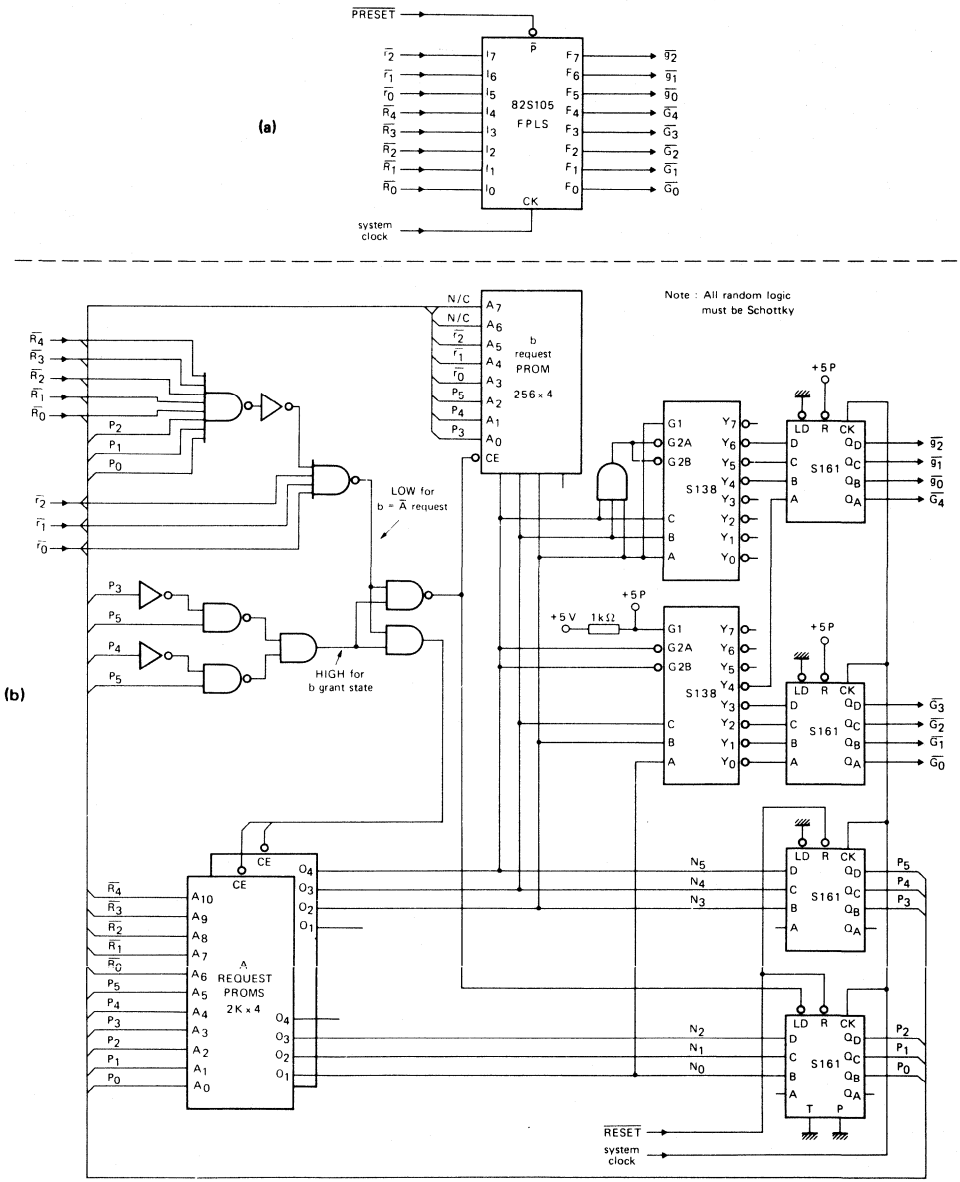


Fig. 7 Arbiter logic diagram (a) using FPLS (b) using PROMs and latches.

PROM/latch implementation

The same arbiter can be implemented with discrete PROMs and latches using the same state diagrams as for the FPLS, except that looping transition terms must now be programmed. Coding of all state and output transitions requires programming of 2 memory fields: the A request PROMs (2K x 6) and the b request PROM (64 x 3). The complete circuit diagram is shown in Fig. 7(b).

The A request PROMs determine the next machine state ($N_{0\text{-}5}$) at all times, except when there are no A requests pending and there is a b request, or if the machine is presently in a b grant state. In these cases, the b request PROM controls the machine's next state.

The grant control lines are decoded from the next state lines and latched in 2 quadruple output latches. This PROM/LATCH organization is conceptually the same as that shown in Fig. 2.

Evaluation of the two designs

As can be seen from the logic diagrams, the FPLS can offer significant advantages over discrete MSI arrays in the design of state machines. The comparison of both design alternatives for the priority arbiter is shown in Table 2. Clearly, the FPLS approach uses fewer parts, with savings in PC board space and power requirements. Furthermore, as many logic elements must be placed in the input-to-latch propagation path, the speed of the PROM/LATCH version is about 50% slower than that of the FPLS.

Table 2 Comparison of the two designs of arbiter.

parameter	FPLS	PROM/LATCH
parts count	1 IC	≈ 19 ICs
PCB space	51 mm²	542 mm²
power	0,65 W	2,85 W
speed	90 ns state	159 ns state
voltage	+5 V	+5 V

REFERENCES

1. W.W. Plumber: "Asynchronous Arbiters", IEEE Transactions on Computers, January 1972, pp. 37-42.
2. R.C. Pearce, J.A. Field and W.D. Little: "Asynchronous Arbiter Module", IEEE Transactions on Computers, September 1975, pp. 931-933.
3. K. Soe Hojberg: "An Asynchronous Arbiter Resolves Resource Allocation Conflicts on a Random Priority Basis", Computer Design, August 1977, pp. 120-123.
4. K. Soe Hojberg: "One-Step Programmable Arbiter for Multiprocessors", Computer Design, April 1978, pp. 154-158.

Expandable memory system using the Signetics Field Programmable Gate Array

The Signetics FPGA is a programmable array of nine bipolar AND/NAND gates, having sixteen common inputs. Input buffers generate both the true and complement forms of the input signals, avoiding the need for inversion of signals outside the chip. The true/complement forms of each of the 16 inputs are connected by two nichrome fuses to each of the AND gates; blowing one or both fuses allows the user to program an input as true, false or don't care. The output of each AND gate is programmable, through a fuse and an exclusive-OR gate, to provide either an AND or NAND function. Both open-collector output (82S102) and three-state output (82S103) versions are available. The block diagram of the FPGA is shown in Fig. 1.

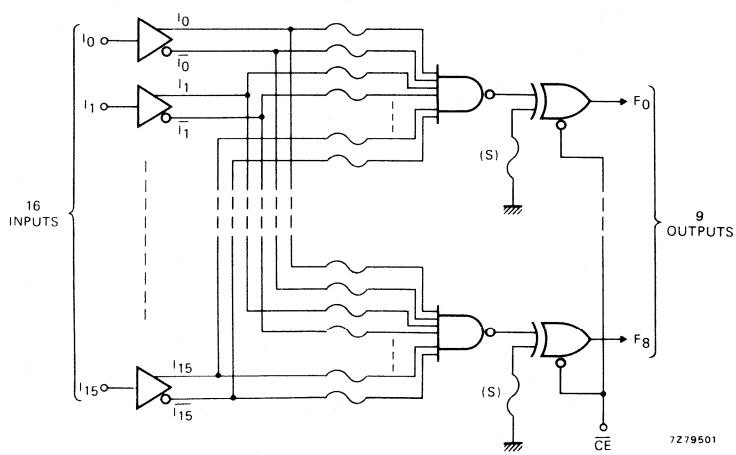


Fig. 1 Logic diagram of FPGA. Each NAND gate is initially connected to true and complement forms of the 16 input signals. The exclusive-OR gates allow programming of active-high or active-low outputs.

Expansion of a memory system at some time during its life is a common requirement. Unfortunately, replacing existing memory chips with larger capacity ones will not work; the chip-enable and address signals must be changed. Normally this will require changes to the printed wiring, resulting in a complete new print board. However, use of an FPGA to generate the chip-enable and some address signals, combined with a generic series of memory chips, greatly simplifies the problem. Figure 2 shows the pin allocation for the Signetics 4K PROM (82S141) and the 16K PROM (82S191). These devices are pin-compatible except for the substitution of pins $\overline{CE}_1$ and NC (not connected) by the pins A_9 and A_{10} in the 16K device. In a similar manner, the Signetics 8K PROM (82S181) uses A_9 in place of NC. Future additions to the range, with larger capacities are expected to use remaining chip-enable inputs as higher-order address lines. This method allows PROMs from 512 x 8 to 8K x 8 to be made pin compatible.

The FPGA is used as a programmable address and chip-enable decoder as shown in Fig. 3. By changing the FPGA program, the system can accept PROMs in the range 4K, 8K, 16K, 32K or 64K, resulting in a memory of 2½ Kbytes to 40 Kbytes. All that need be changed is the FPGA and the PROMs. FPGA outputs F_0 to F_3 provide the appropriate chip-enable or address line according to the PROM in use. Table 1 shows the different signal requirements of the PROM range. Outputs F_4 to F_7 provide the chip-select signals for the individual PROMs. Note that the start address of the memory system can be programmed anywhere in the 64K address field of the 16 inputs. Another feature of the FPGA decoder is its simple compatibility with the system bus control mechanism: the control signal $\overline{RD}$ can be connected to the FPGA chip-enable input so that when $\overline{RD}$ is low the memory system is enabled, when $\overline{RD}$ is high the PROM outputs are held disabled.

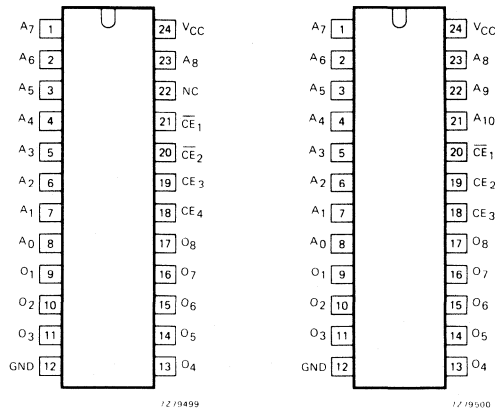


Fig. 2 Pin allocation for (a) 82S141 4K PROM, and (b) 82S191 16K PROM.

TABLE 1 Address/chip-enable signals for PROM range.

PROM	size	F_3	F_2	F_1	F_0
82S141	4K	CE_4	CE_3	$\overline{CE}_1$	NC
82S181	8K	CE_4	CE_3	$\overline{CE}_1$	A_9
82S191	16K	CE_3	$\overline{CE}_2$	A_{10}	A_9
*	32K	$\overline{CE}_2$	A_{11}	A_{10}	A_9
*	64K	A_{12}	A_{11}	A_{10}	A_9

* Yet to be announced.

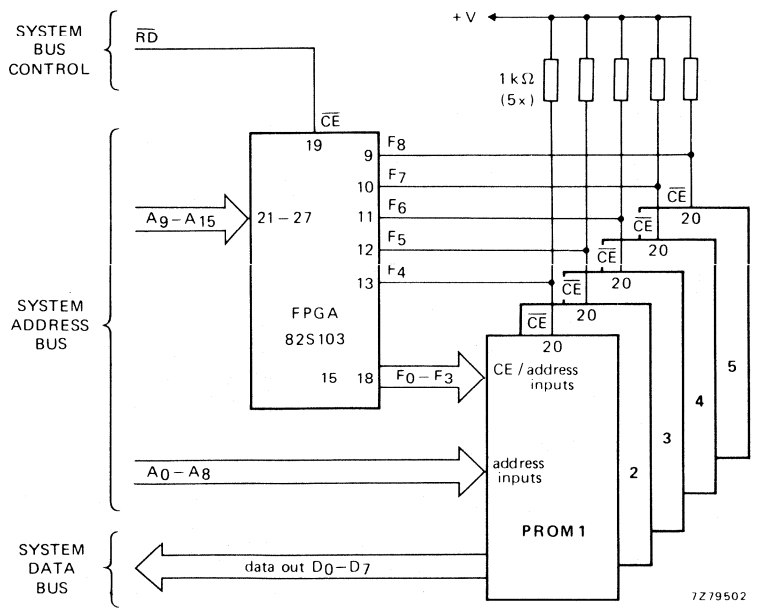


Fig. 3 Expandable memory system using an FPGA.

Tables 2 and 3 show the FPGA program required for a 2½ Kbyte memory using 4K PROMs and a 40 Kbyte memory using 64K PROMs, using the circuit of Fig. 3 in each case.

TABLE 2 Program for 2½ Kbyte memory using 4K PROMs.

OUTPUT POLARITY		INPUT VARIABLE															
		I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_A	I_B	I_C	I_D	I_E	I_F
F_0	H 0	; 0	— 1	— 2	— 3	— 4	— 5	— 6	— 7	— 8	— 9	— 10	— 11	— 12	— 13	— 14	— 15
F_1	H 16	— 16	— 17	— 18	— 19	— 20	— 21	— 22	— 23	— 24	— 25	— 26	— 27	— 28	— 29	— 30	— 31
F_2	L 32	— 32	— 33	— 34	— 35	— 36	— 37	— 38	— 39	— 40	— 41	— 42	— 43	— 44	— 45	— 46	— 47
F_3	L 48	— 48	— 49	— 50	— 51	— 52	— 53	— 54	— 55	— 56	— 57	— 58	— 59	— 60	— 61	— 62	— 63
F_4	L 64	— 64	— 65	— 66	— 67	— 68	— 69	— 70	— 71	— 72	L 73	L 74	L 75	L 76	L 77	L 78	L 79
F_5	L 80	— 80	— 81	— 82	— 83	— 84	— 85	— 86	— 87	— 88	H 89	L 90	L 91	L 92	L 93	L 94	L 95
F_6	L 96	— 96	— 97	— 98	— 99	— 100	— 101	— 102	— 103	— 104	L 105	H 106	L 107	L 108	L 109	L 110	L 111
F_7	L 112	— 112	— 113	— 114	— 115	— 116	— 117	— 118	— 119	— 120	H 121	H 122	L 123	L 124	L 125	L 126	L 127
F_8	L 128	— 128	— 129	— 130	— 131	— 132	— 133	— 134	— 135	— 136	L 137	L 138	H 139	L 140	L 141	L 142	L 143
Active-high = H Active-low = L		I_m = H $\overline{I_m}$ = L Don't Care = —															

The number in each cell in the table denotes its address for programmers with a decimal address display.

TABLE 3 Program for 40 Kbyte memory using 64K PROMs.

OUTPUT POLARITY		INPUT VARIABLE															
		I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_A	I_B	I_C	I_D	I_E	I_F
F_0	H 0	— 0	— 1	— 2	— 3	— 4	— 5	— 6	— 7	— 8	H 9	— 10	— 11	— 12	— 13	— 14	— 15
F_1	H 16	— 16	— 17	— 18	— 19	— 20	— 21	— 22	— 23	— 24	— 25	H 26	— 27	— 28	— 29	— 30	— 31
F_2	H 32	— 32	— 33	— 34	— 35	— 36	— 37	— 38	— 39	— 40	— 41	— 42	H 43	— 44	— 45	— 46	— 47
F_3	H 48	— 48	— 49	— 50	— 51	— 52	— 53	— 54	— 55	— 56	— 57	— 58	— 59	H 60	— 61	— 62	— 63
F_4	L 64	— 64	— 65	— 66	— 67	— 68	— 69	— 70	— 71	— 72	— 73	— 74	— 75	— 76	L 77	L 78	L 79
F_5	L 80	— 80	— 81	— 82	— 83	— 84	— 85	— 86	— 87	— 88	— 89	— 90	— 91	— 92	H 93	L 94	L 95
F_6	L 96	— 96	— 97	— 98	— 99	— 100	— 101	— 102	— 103	— 104	— 105	— 106	— 107	— 108	L 109	H 110	L 111
F_7	L 112	— 112	— 113	— 114	— 115	— 116	— 117	— 118	— 119	— 120	— 121	— 122	— 123	— 124	H 125	H 126	L 127
F_8	L 128	— 128	— 129	— 130	— 131	— 132	— 133	— 134	— 135	— 136	— 137	— 138	— 139	— 140	L 141	L 142	H 143
Active-high = H Active-low = L		I_m = H $\overline{I_m}$ = L Don't Care = —															

The number in each cell in the table denotes its address for programmers with a decimal address display.

8-bit parallel CRC generator/checker using Field Programmable Logic

The cyclic redundancy check (CRC) is a method of error detection that is widely used in data transmission and recording systems. The CRC method produces a unique check word that is appended to the end of the data sequence. Should any bit in the data or check word become corrupted, subsequent decoding will indicate an error. The check word is generated by the division of the data, in a serial stream, by a selected polynominal expression. The remainder generated by this division forms the unique check word. The CRC16 Reverse polynominal [$P(X) = X^{16} + X^{15} + X + 1$] is one of the more common expressions to be used, and is the one considered in this Technical Note.

Conventional generator/checker

The conventional CRC circuit uses a serial data input and a shift register with feedback loops to simulate the code polynominal. A typical circuit is shown in Fig. 1. The sequence starts with all the flip-flops being reset via the MR line. Each bit of the data is then sequentially clocked into the shift register via the D line. When the last bit of data has been clocked in, The CRC check word is available on the Q outputs of the flip-flops.

When the data is checked, it is entered into the the shift register, in the same way as for check-word generation, followed by the check word. If any bit of the data has been corrupted, the shift register contents are non-zero.

This serial method of CRC generation/checking is simple to implement providing that the data is already in a serial bit stream. However, when the data is in a parallel format, not only are parallel-to-serial and serial-to-parallel conversion circuits required, but the serial part of the system must work at several times the speed of the parallel data system. In the case of 8-bit bytes, the serial logic must be clocked 8 times faster than the parallel logic.

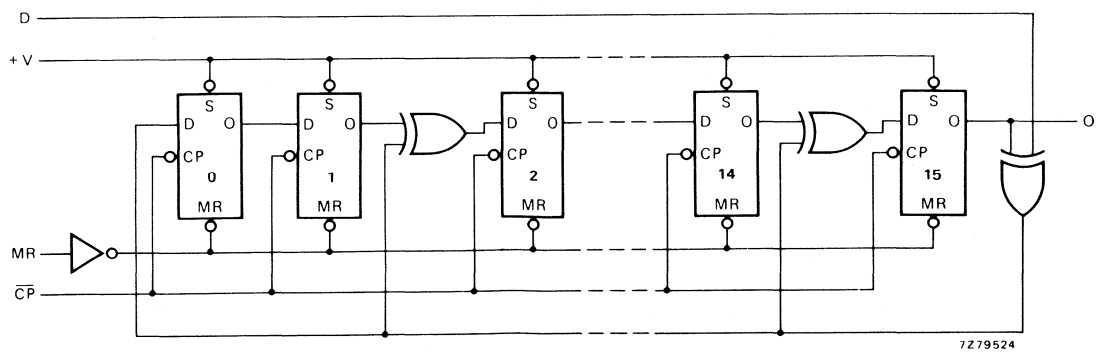


Fig. 1 Conventional serial CRC generator/checker circuit.

Parallel CRC generator/checker

A parallel CRC circuit can be constructed from clocked combinational logic to perform the same functions as the circuit of Fig. 1, and thus avoid the problem of format conversion and high speeds. However, the resulting logic circuit is complex and only becomes a practical proposition when it can be implemented in an IC such as the FPLA. Figure 2 shows the schematic diagram, while Table 1 gives the logic equations defining the next state of each output. A practical circuit is shown in Fig. 3: it uses only five ICs, one of which can be omitted if the checking function is not required. It has a data throughput rate of over 5,7 Mbytes/s.

Circuit operation begins with resetting the latches and then entering the first data byte at inputs D_0 to D_7. The output of the FPLAs is then clocked into the latches before the next data byte is presented at D_0 to D_7. In the generation mode, the check words are available on outputs B_0 to B_{15} after clocking the last data byte. In the check mode, the two check words must be entered after the data block. If an error has occurred, the $\overline{\text{ERROR}}$ line is set low.

Tables 2, 3 and 4 show the programs for the two FPLAs and the FPGA.

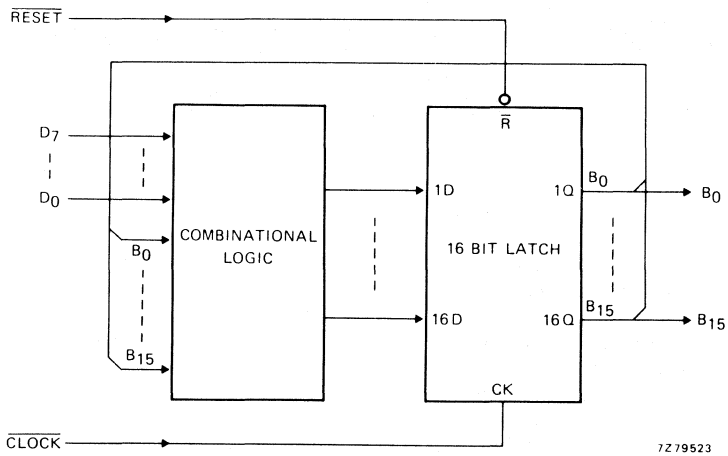


Fig. 2 Schematic diagram of parallel CRC generator/checker circuit.

TABLE 1 Logic equations defining the next state of each output of the circuit of Fig. 2.

$NB_0 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus B_8 \oplus B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} \oplus B_{13} \oplus B_{14} \oplus B_{15}$

$NB_1 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} \oplus B_{13} \oplus B_{14} \oplus B_{15}$

$NB_2 = D_6 \oplus D_7 \oplus B_8 \oplus B_9$

$NB_3 = D_5 \oplus D_6 \oplus B_9 \oplus B_{10}$

$NB_4 = D_4 \oplus D_5 \oplus B_{10} \oplus B_{11}$

$NB_5 = D_3 \oplus D_4 \oplus B_{11} \oplus B_{12}$

$NB_6 = D_2 \oplus D_3 \oplus B_{12} \oplus B_{13}$

$NB_7 = D_1 \oplus D_2 \oplus B_{13} \oplus B_{14}$

$NB_8 = D_0 \oplus D_1 \oplus B_0 \oplus B_{14} \oplus B_{15}$

$NB_9 = D_0 \oplus B_1 \oplus B_{15}$

$NB_{10} = B_2$

$NB_{11} = B_3$

$NB_{12} = B_4$

$NB_{13} = B_5$

$NB_{14} = B_6$

$NB_{15} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus B_7 \oplus B_8 \oplus B_9 \oplus B_{10} \oplus B_{11} \oplus B_{12} \oplus B_{13} \oplus B_{14} \oplus B_{15}$

NB_n: next state of B_n

TABLE 2 Program table for $FPLA_1$ in Fig. 3.

NO	PRODUCT TERM: INPUT VARIABLE 1	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: OUTPUT FUNCTION 7	6	5	4	3	2	1	0
	5																H	H	H	H	H	H	H	H
0	L	L	–	–	–	–	–	–	–	–	–	–	L	H	–	–	·	·	·	·	·	A	·	·
1	H	H	–	–	–	–	–	–	–	–	–	–	L	H	–	–	·	·	·	·	·	A	·	·
2	L	L	–	–	–	–	–	–	–	–	–	–	H	L	–	–	·	·	·	·	·	A	·	·
3	–	–	–	–	–	L	L	H	L	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
4	–	–	–	–	–	H	H	H	L	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
5	–	–	–	–	–	L	L	L	H	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
6	–	–	–	–	–	H	H	L	H	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
7	–	–	–	–	–	L	H	L	L	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
8	–	–	–	–	–	L	H	H	H	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
9	–	–	–	–	–	H	L	L	L	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
10	–	–	–	–	–	H	L	H	H	–	–	–	–	–	–	–	·	·	·	·	A	·	·	·
11	–	–	–	–	L	L	–	–	H	L	–	–	–	–	–	–	·	·	·	A	·	·	·	·
12	–	–	–	–	H	H	–	–	H	L	–	–	–	–	–	–	·	·	·	A	·	·	·	·
13	–	–	–	–	L	L	–	–	L	H	–	–	–	–	–	–	·	·	·	A	·	·	·	·
14	–	–	–	–	H	H	–	–	L	H	–	–	–	–	–	–	·	·	·	A	·	·	·	·
15	–	–	–	–	L	H	–	–	L	L	–	–	–	–	–	–	·	·	·	A	·	·	·	·
16	–	–	–	–	L	H	–	–	H	H	–	–	–	–	–	–	·	·	·	A	·	·	·	·
17	–	–	–	–	H	L	–	–	L	L	–	–	–	–	–	–	·	·	·	A	·	·	·	·
18	–	–	–	–	H	L	–	–	H	H	–	–	–	–	–	–	·	·	·	A	·	·	·	·
19	–	–	–	L	L	–	–	–	–	L	H	–	–	–	–	–	·	·	A	·	·	·	·	·
20	–	–	–	H	H	–	–	–	–	L	H	–	–	–	–	–	·	·	A	·	·	·	·	·
21	–	–	–	L	L	–	–	–	–	H	L	–	–	–	–	–	·	·	A	·	·	·	·	·
22	–	–	–	H	H	–	–	–	–	H	L	–	–	–	–	–	·	·	A	·	·	·	·	·
23	–	–	–	L	H	–	–	–	–	L	L	–	–	–	–	–	·	·	A	·	·	·	·	·
24	–	–	–	L	H	–	–	–	–	H	H	–	–	–	–	–	·	·	A	·	·	·	·	·
25	–	–	–	H	L	–	–	–	–	L	L	–	–	–	–	–	·	·	A	·	·	·	·	·
26	–	–	–	H	L	–	–	–	–	H	H	–	–	–	–	–	·	·	A	·	·	·	·	·
27	–	–	L	L	–	–	–	–	–	–	L	H	–	–	–	–	·	A	·	·	·	·	·	·
28	–	–	H	H	–	–	–	–	–	–	L	H	–	–	–	–	·	A	·	·	·	·	·	·
29	–	–	L	L	–	–	–	–	–	–	H	L	–	–	–	–	·	A	·	·	·	·	·	·
30	–	–	H	H	–	–	–	–	–	–	H	L	–	–	–	–	·	A	·	·	·	·	·	·
31	–	–	L	H	–	–	–	–	–	–	L	L	–	–	–	–	·	A	·	·	·	·	·	·
32	–	–	L	H	–	–	–	–	–	–	H	H	–	–	–	–	·	A	·	·	·	·	·	·
33	–	–	H	L	–	–	–	–	–	–	L	L	–	–	–	–	·	A	·	·	·	·	·	·
34	–	–	H	L	–	–	–	–	–	–	H	H	–	–	–	–	·	A	·	·	·	·	·	·
35	–	L	L	–	–	–	–	–	–	–	–	L	H	–	–	–	A	·	·	·	·	·	·	·
36	–	H	H	–	–	–	–	–	–	–	–	L	H	–	–	–	A	·	·	·	·	·	·	·
37	–	L	L	–	–	–	–	–	–	–	–	H	L	–	–	–	A	·	·	·	·	·	·	·
38	–	H	H	–	–	–	–	–	–	–	–	H	L	–	–	–	A	·	·	·	·	·	·	·
39	–	L	H	–	–	–	–	–	–	–	–	L	L	–	–	–	A	·	·	·	·	·	·	·
40	–	L	H	–	–	–	–	–	–	–	–	H	H	–	–	–	A	·	·	·	·	·	·	·
41	–	H	L	–	–	–	–	–	–	–	–	L	L	–	–	–	A	·	·	·	·	·	·	·
42	–	H	L	–	–	–	–	–	–	–	–	H	H	–	–	–	A	·	·	·	·	·	·	·
43	H	H	–	–	–	–	–	–	–	–	–	–	H	L	–	–	·	·	·	·	·	A	·	·
44	L	H	–	–	–	–	–	–	–	–	–	–	L	L	–	–	·	·	·	·	·	A	·	·
45	L	H	–	–	–	–	–	–	–	–	–	–	H	H	–	–	·	·	·	·	·	A	·	·
46	H	L	–	–	–	–	–	–	–	–	–	–	L	L	–	–	·	·	·	·	·	A	·	·
47	H	L	–	–	–	–	–	–	–	–	–	–	H	H	–	–	·	·	·	·	·	A	·	·

TABLE 3 Program table for $FPLA_2$ in Fig. 3.

NO	PRODUCT TERM: INPUT VARIABLE 1	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: OUTPUT FUNCTION 7	6	5	4	3	2	1	0
	5																H	H	H	H	H	H	H	H
0	–	L	H	L	–	–	L	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
1	–	L	H	H	–	–	H	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
2	–	H	L	L	–	–	L	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
3	–	H	L	H	–	–	H	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
4	–	L	L	L	–	–	H	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
5	–	H	H	L	–	–	H	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
6	–	L	L	H	–	–	L	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
7	–	H	H	H	–	–	L	–	–	–	–	–	L	–	–	–	·	·	·	A	·	·	·	·
8	–	L	L	L	–	–	L	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
9	–	L	L	H	–	–	H	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
10	–	L	H	L	–	–	H	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
11	–	L	H	H	–	–	L	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
12	–	H	H	L	–	–	L	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
13	–	H	H	H	–	–	H	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
14	–	H	L	L	–	–	H	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
15	–	H	L	H	–	–	L	–	–	–	–	–	H	–	–	–	·	·	·	A	·	·	·	·
16	–	–	–	–	–	H	–	–	–	–	–	L	–	–	–	L	·	·	A	·	·	·	·	·
17	–	–	–	–	–	L	–	–	–	–	–	H	–	–	–	L	·	·	A	·	·	·	·	·
18	–	–	–	–	–	L	–	–	–	–	–	L	–	–	–	H	·	·	A	·	·	·	·	·
19	–	–	–	–	–	H	–	–	–	–	–	H	–	–	–	H	·	·	A	·	·	·	·	·
20	–	–	–	–	–	L	–	–	–	–	–	L	–	H	–	L	·	·	·	·	·	·	·	A
21	–	–	–	–	–	H	–	–	–	–	–	L	–	H	–	H	·	·	·	·	·	·	·	A
22	–	–	–	–	–	L	–	–	–	–	–	H	–	L	–	L	·	·	·	·	·	·	·	A
23	–	–	–	–	–	H	–	–	–	–	–	H	–	L	–	H	·	·	·	·	·	·	·	A
24	–	–	–	–	–	L	–	–	–	–	–	L	–	L	–	H	·	·	·	·	·	·	·	A
25	–	–	–	–	–	L	–	–	–	–	–	H	–	H	–	H	·	·	·	·	·	·	·	A
26	–	–	–	–	–	H	–	–	–	–	–	L	–	L	–	L	·	·	·	·	·	·	·	A
27	–	–	–	–	–	H	–	–	–	–	–	H	–	H	–	L	·	·	·	·	·	·	·	A
28	–	–	–	–	–	–	H	–	–	–	L	–	L	–	–	–	·	A	·	·	·	·	·	·
29	–	–	–	–	–	–	L	–	–	–	L	–	H	–	–	–	·	A	·	·	·	·	·	·
30	–	–	–	–	–	–	L	–	–	–	H	–	L	–	–	–	·	A	·	·	·	·	·	·
31	–	–	–	–	–	–	H	–	–	–	H	–	H	–	–	–	·	A	·	·	·	·	·	·
32	L	–	–	–	–	–	L	–	H	L	–	–	L	–	–	–	A	·	·	·	·	·	·	·
33	L	–	–	–	–	–	H	–	H	L	–	–	H	–	–	–	A	·	·	·	·	·	·	·
34	L	–	–	–	–	–	L	–	L	H	–	–	L	–	–	–	A	·	·	·	·	·	·	·
35	L	–	–	–	–	–	H	–	L	H	–	–	H	–	–	–	A	·	·	·	·	·	·	·
36	L	–	–	–	–	–	H	–	L	L	–	–	L	–	–	–	A	·	·	·	·	·	·	·
37	L	–	–	–	–	–	H	–	H	H	–	–	L	–	–	–	A	·	·	·	·	·	·	·
38	L	–	–	–	–	–	L	–	L	L	–	–	H	–	–	–	A	·	·	·	·	·	·	·
39	L	–	–	–	–	–	L	–	H	H	–	–	H	–	–	–	A	·	·	·	·	·	·	·
40	H	–	–	–	–	–	L	–	L	L	–	–	L	–	–	–	A	·	·	·	·	·	·	·
41	H	–	–	–	–	–	H	–	L	L	–	–	H	–	–	–	A	·	·	·	·	·	·	·
42	H	–	–	–	–	–	H	–	H	L	–	–	L	–	–	–	A	·	·	·	·	·	·	·
43	H	–	–	–	–	–	L	–	H	L	–	–	H	–	–	–	A	·	·	·	·	·	·	·
44	H	–	–	–	–	–	L	–	H	H	–	–	L	–	–	–	A	·	·	·	·	·	·	·
45	H	–	–	–	–	–	H	–	H	H	–	–	H	–	–	–	A	·	·	·	·	·	·	·
46	H	–	–	–	–	–	H	–	L	H	–	–	L	–	–	–	A	·	·	·	·	·	·	·
47	H	–	–	–	–	–	L	–	L	H	–	–	H	–	–	–	A	·	·	·	·	·	·	·

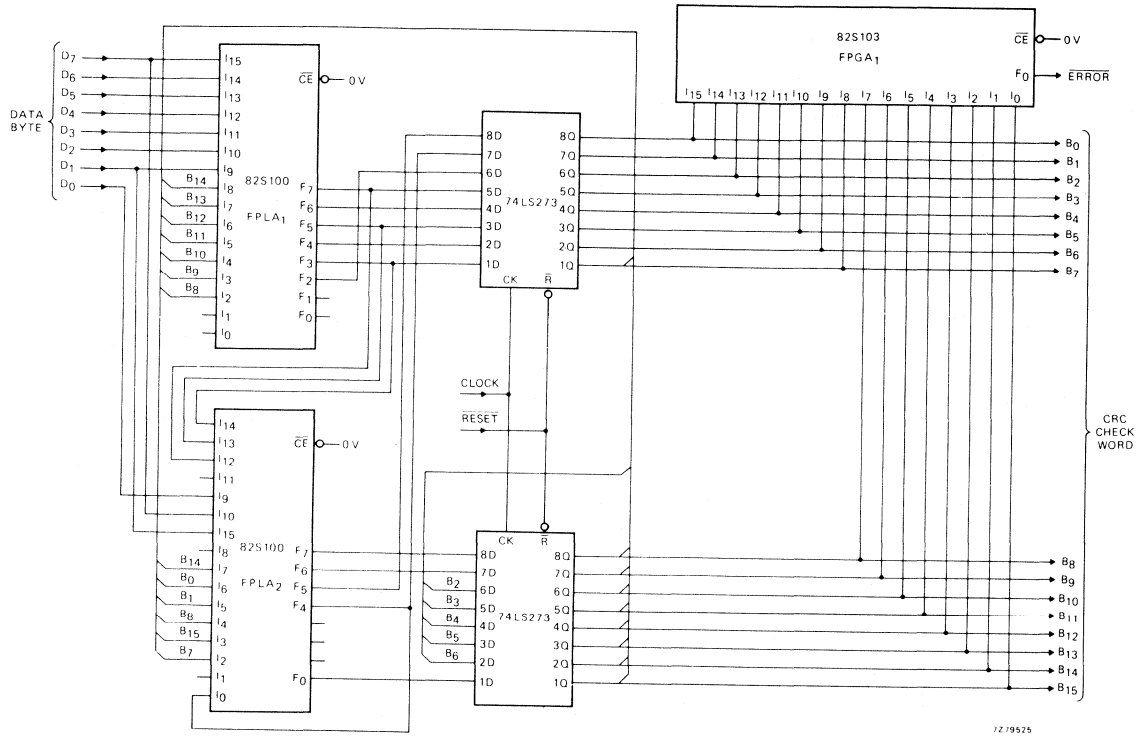


Fig. 3 Practical circuit for parallel CRC circuit.

TABLE 4 Program table for FPGA in Fig. 3.

OUTPUT	POLARITY	INPUT VARIABLE															
		I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
F_0	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
F_1																	
F_2																	
F_3																	
F_4																	
F_5																	
F_6																	
F_7																	
F_8																	

A simple keyboard encoder using the FPLS

A keyboard encoding device must be able to perform four basic functions:

- keyboard scanning;
- contact bounce suppression;
- key encoding;
- signalling availability of valid output data.

The maximum size of keyboard that can be used with a given encoder depends largely on the time taken to scan the keyboard and the time taken to suppress the effects of key contact bounce. The longer that these times are, the slower the maximum data input rate becomes.

Contact bounce suppression can be performed by a low-pass filter consisting of an RC combination. A filter with a cut-off around 100 Hz is usually sufficient to remove the high-frequency bounce signals. However, the design of some encoders makes it more attractive to suppress the bounce signals in a digital manner. This can be done by regularly sampling the contact signal to ascertain its stability over a defined period. Figure 1 shows the flow chart of a simple bounce suppression system. Once a depressed key has been detected, there must be n consecutive 'no key pressed' samples before the key is considered released and the scanning routine can be re-started. The number of samples required to suppress contact bounce depend on the sampling frequency, f_s, and low-pass cut-off frequency, f_c:

$$n = f_s/f_c.$$

For a 1 kHz sampling rate and a 100 Hz cut-off, there must be 10 samples with the key released before the system can continue scanning.

The key encoding mechanism may be fixed in the hardware, or may be programmable, allowing the user to select the code for each key. Fixed encoders normally conform to national or international standards for keyboards.

Once a depressed key is detected and encoded, the code must be presented at the output, accompanied by a signal to indicate the presence of a valid code. The form of this signal will depend upon the requirements of the application.

Obviously, the ideal keyboard encoder will be a single chip, providing the user with the facility to program all the operational parameters just mentioned. The simplest approach to this problem is to use the Signetics Field Programmable Logic Sequencer (FPLS). The FPLS is simple and quick to program and does not need the tedious assembly processes of a microprocessor. It is available in both open collector (N82S104) and three-state (N82S105) versions.

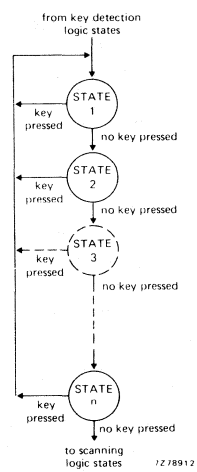


Fig. 1 State diagram of a digital filter to eliminate the effects of key bounce.

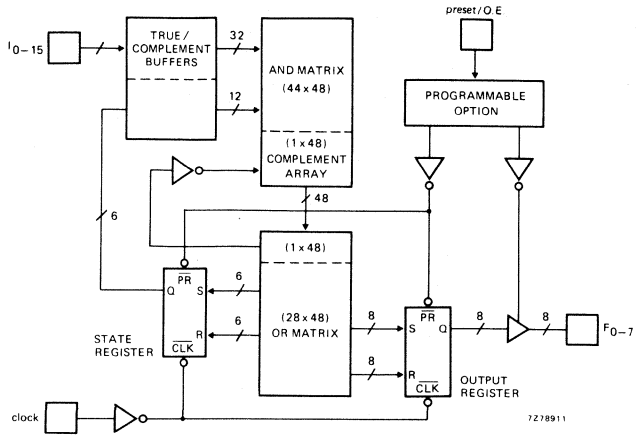


Fig. 2 Block diagram of the Signetics FPLS 82S105/6.

Design of a keyboard encoder using the FPLS

The combination of a synchronous state machine and an asynchronous switch matrix requires a unique state machine algorithm.

Every synchronous circuit that combines combinational logic and a latch, as shown in Fig. 3(a), must ensure a sufficient set-up time to allow for the propagation delay of the logic and latch. Failure to do this can lead to undesired changes in the state or outputs. Figure 3(b) illustrates this effect: when I_0 becomes high, the next state of both F_0 and F_1 should be high, but if the input delay to F_1 is such that the clock pulse occurs before the input reaches F_1, an error will occur.

With a device like a keyboard, there is no way to guarantee set-up time, as the user can press a key at any instant. One way of avoiding this problem is to use a latch between the keyboard and the encoder. However, there is another solution that does not require the extra latch: segregated logic techniques.

Logic segregation separates state machine logic such that any external asynchronous event affects only one state or output register bit. Thus, set-up time violations will not result in errors, but merely a delay of one clock cycle. Figure 4 shows the complete circuit, excluding the clock, for an encoder for a 60 key keyboard operating under the principle described above.

The FPLS scans the key matrix in rows of 15 keys. Outputs F_0 to F_3 are used to select a row by making one output low at a time. If a key in the selected row is depressed, one of the inputs I_1 to I_{15} will be low, causing the FPLS to leave the scanning mode and enter the key-detected mode. The code programmed for this key is then output on F_0 to F_6, and $\overline{\text{key Strobe}}$, F_7, goes low for one clock period. The key data should be clocked into the receiving logic by the rising edge of $\overline{\text{Key Strobe}}$. After output of the key data, the system performs a bounce signal suppression routine before returning to the scan mode.

The tasks performed by the FPLS can be divided into three areas: key scanning, key code output strobing, and bounce suppression. In construction of the FPLS program, care has been taken to ensure that only one internal register bit is changed as a result of a change in the external inputs (I_0 to I_{15}), complying with the requirement of segregated logic. The program chart for the FPLS is shown in Fig. 5.

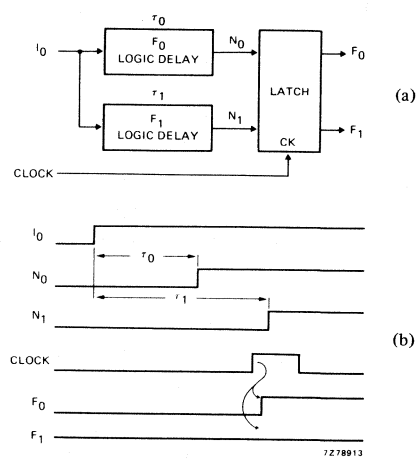


Fig. 3 Combinational logic and latches. (a) general circuit, and (b) timing diagram showing violation of set-up time requirements and the result.

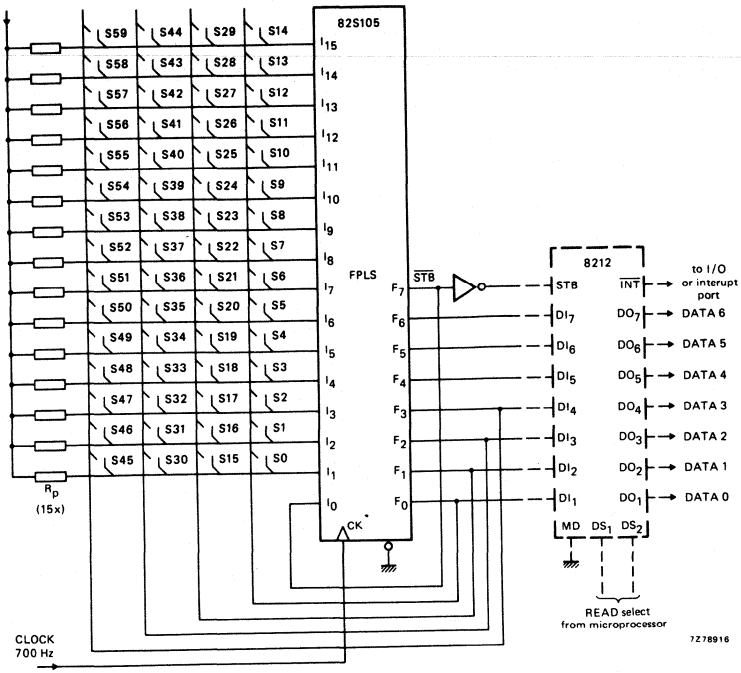


Fig. 4 Circuit diagram of a 60 key encoder using the FPLS. The 8212 I/O port, shown in broken lines, is only required for direct connection to a microprocessor.

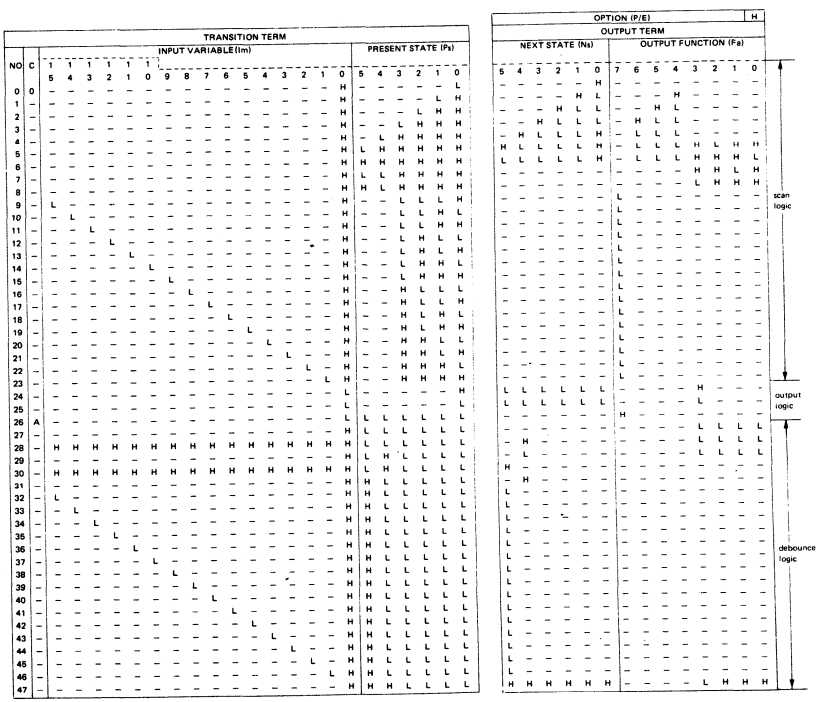


Fig. 5 Program chart for the FPLS in the circuit of Fig. 4.

Key scanning

The keys are scanned in four columns of 15 keys each. The six present state bits are used to represent a four-bit row counter, counting from 1 to 15, and a two-bit column counter, counting 0 to 3. The row counter is automatically incremented by lines 0 to 3 as long as no key is pressed. When the count reaches 15, lines 4 to 6 ensure that the column counter is incremented and the row counter reset to 1. At the same time as the state counter is being incremented, the output register is incremented to hold the 3 most significant bits of the row counter. The least significant bit of the row counter cannot be held in the output register because output functions 0 to 3 are used to select the columns of keys. Figure 6 shows the state diagram of the scanning logic.

When a key is pressed, the row and column counters are incremented as described above until the correct column is selected and the row counter reaches the value corresponding to the pressed key. At this point, one of lines 9 to 23 is active, causing the $\overline{\text{Strobe}}$ output to go low. The state and output registers remain unchanged. The $\overline{\text{Strobe}}$ output, connected to input I_0 causes the system to enter the output mode on the next clock pulse. Figure 7 shows the interruption of the scanning sequence when a pressed key is detected.

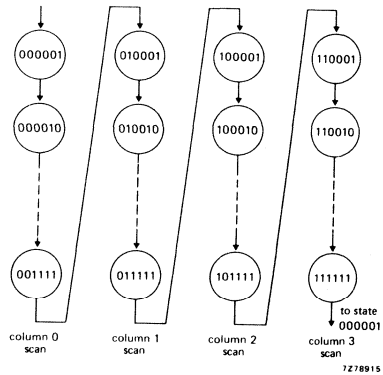


Fig. 6 State diagram for key scanning operation, no key pressed.

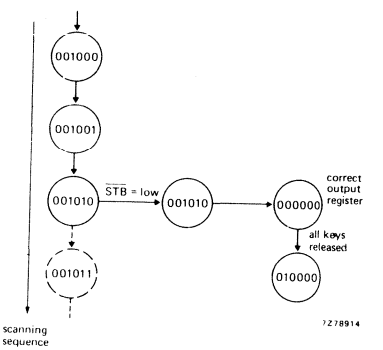


Fig. 7 Detection of pressed key (001010) during scanning mode.

Output strobing

With input I_0 ($\overline{\text{Strobe}}$) low, a transition occurs from the last counter state to state zero. In this state, output F_3 is set high or low, corresponding to the least significant bit of the row counter.

On the next cycle, output F_7 goes high to complete the output cycle. With F_7 high in state zero, the system enters the bounce suppression mode.

The code generated for each key is determined by the counting action. However, these codes are programmable because any key can be allocated to any state of the counter. The program linking the keys to the counter states can be found in lines 9 to 23 of the FPLS program.

Bounce suppression

While the key is still pressed, the system remains in state zero with bits F_0 to F_3 being set low. When the key is lifted, the system enters the bounce suppression sequence, lines 27 to 47. Detection of any key pressed during the bounce suppression sequence results in the sequence being started anew. By ensuring that no other keys are pressed, instance of simply the key just detected, the system overcomes the problems of two or more keys being detected at the same time: the first key to be scanned, when two or more are pressed, is the one detected.

Interface to a microprocessor

A keyboard encoder can be connected to a microprocessor in either a scanned or interrupt mode. In the scanned mode, the microprocessor regularly interrogates the encoder to see if a key has been pressed, and if so, reads the key code. In the interrupt mode, the encoder generates an interrupt signal to the microprocessor when a pressed key is detected. The interrupt signal causes the microprocessor to execute an interrupt service routine which reads and processes the key code.

Addition of an 8-bit register, shown in broken lines in Fig. 4, allows the FPLS encoder to be used in either mode with a microprocessor. The 8212 is a special interface register incorporating a service request latch, which is set by the STB input. The output $\overline{\text{INT}}$ is then set low until the data latches are read by the microprocessor.

Specifications of the encoder

Maximum number of keys	60
Key codes	programmable
Maximum keyboard scan time	86 ms
Bounce suppression time	8,6 ms
Power supply	5,0 ± 0,25 V, 650 mW typical.

Larger capacity keyboard

The circuit shown in Fig. 8 can be used to expand the encoder's capability up to 120 keys. Two open-collector FPLSs (82S104) with the same program as for the 60-key encoder are used. A Signetics Field Programmable Gate Array (82S103) provides the gated interconnections.

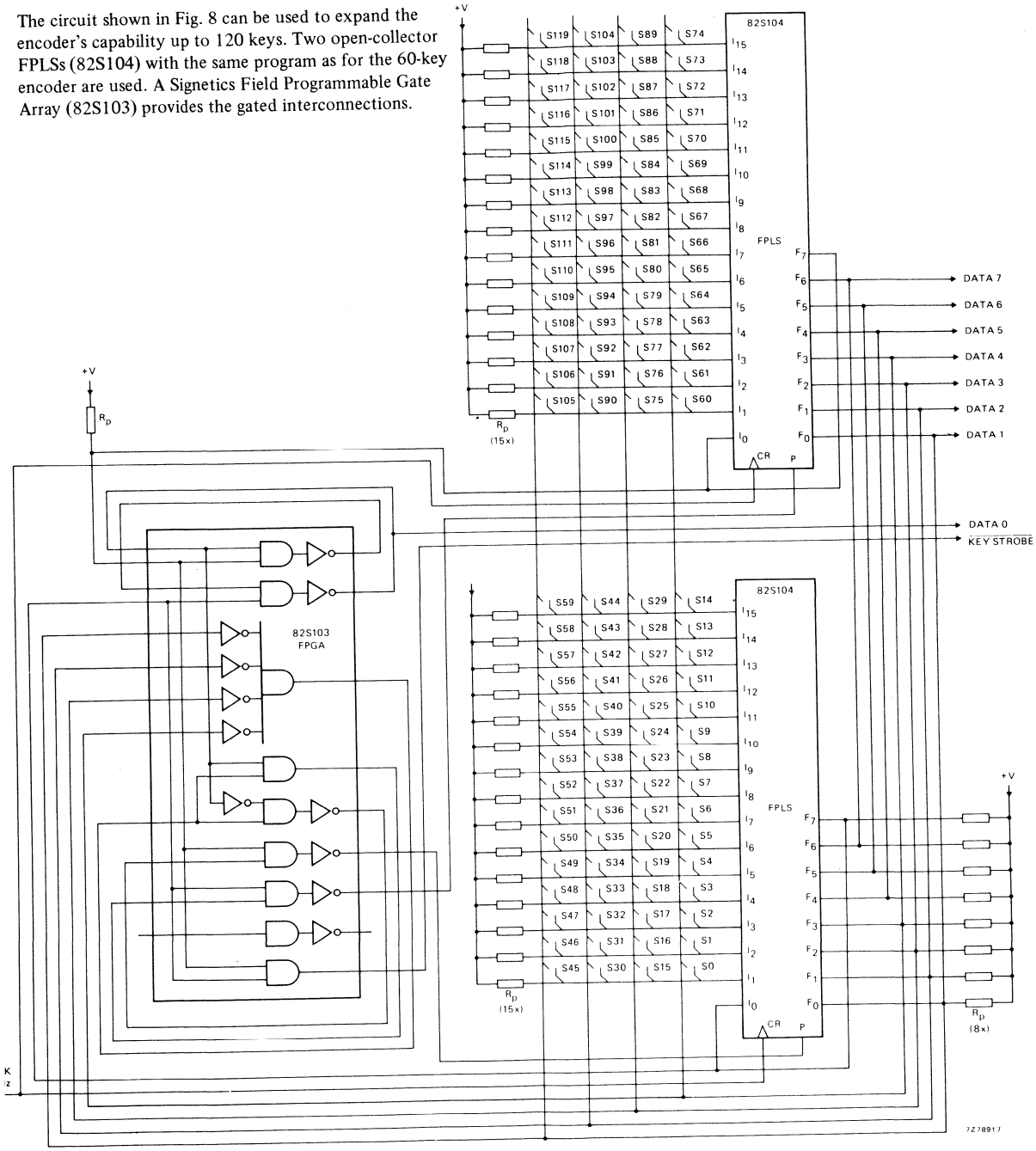


Fig. 8 120 key keyboard encoder.

THE FPLA ADDS DISPLAY FLEXIBILITY TO CRT CONTROLLERS

INTRODUCTION

The FPLA may be used to add display flexibility in many raster scan CRT applications.

Most CRT controllers are designed to scan a screen a line at a time from the lefthand screen side to the right, and proceed line by line from the top of the screen to the last displayed line at the bottom of the screen.

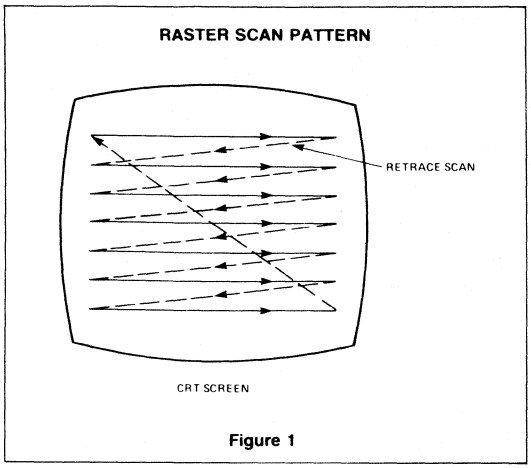


Figure 1

This scan pattern is ideally suited for displaying text that is written in the same scan pattern, such as English. This allows a CRT refresh memory to store lines of text in a linear, contiguous manner. For instance, a typical 72 character wide by 64 line (72X64) CRT monitor could arrange its refresh memory such that it maps directly to the CRT scan pattern.

This allows the CRT scan address to be used as the refresh memory address.

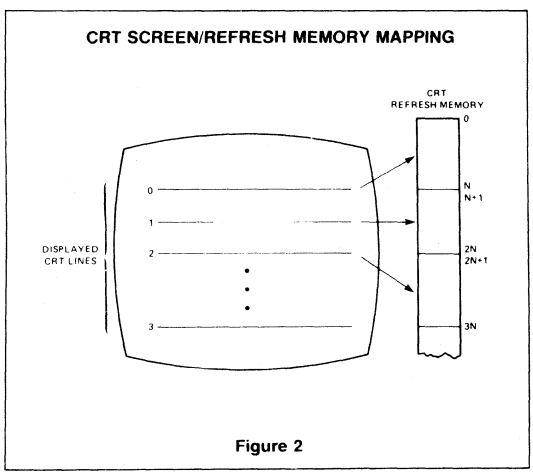


Figure 2

Although this works well for English text, the same CRT controller would have a difficult time displaying foreign text, such as Japanese which is written in lines from top to bottom and proceeds from right to left, or Arabic which is written in lines from right to left and proceeds from top to bottom. This discussion will show how the FPLA can add text display flexibility to a typical CRT controller; the 6845.

CRT CONTROLLER FUNCTIONS

It is the task of the CRT controller to configure and control the raster scan of the CRT in a manner that allows text to be easily displayed.

Towards this end, the 6845 uses the same line scan pattern shown in Figure 1. It combines many scan lines together to form a single character row. Figure 3 illustrates this technique. Also under its control is the number of characters displayed per character row and thus the horizontal dot pattern spacing.

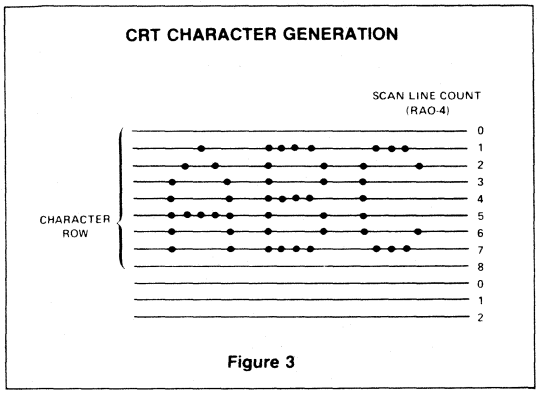


Figure 3

As can be seen in Figure 4, lines RA0-4 address the scan lines within a character row. They sequence from a count of 0 to last character row scan line as shown in Figure 3.

Refresh memory address lines, MA0-13 sequence from a programmed start address on a character by character basis to the end of the last character displayed in the last character row. Figure 5 illustrates this sequential count for a 72 by 64 CRT display with a start address of 0.

The CRT controller has two primary operational modes; refresh memory update, and CRT refresh.

During the refresh memory update cycle, the refresh memory address multiplexer is switched towards the processor address bus. At this time the processor can insert new or updated data to be displayed on the CRT screen. Since this cycle requires refresh RAM address control, CRT refresh must be interlaced around the processor to RAM updates. A good time to update the RAM is during scan line retrace time (see Figure 1). At this time no CRT refresh operations are taking place and is, therefore, a perfect time window. The 6845 indicates this window via its DISPLAY ENABLE line.

During the CRT refresh cycle the refresh memory address multiplexer is switched towards the 6845, where lines MA0-13 provide addressing. Addressed refresh data is then latched and translated by the Character Generator ROM into a corresponding row dot pattern. This word is then shifted serially to the video buffer which provides the CRT display interface.

For a complete description of the operation of the 6845, refer to the 6845 data sheet. At this point the discussion shall focus on the way this typical CRT controller application may be adapted with an FPLA to provide foreign text display.

CRT REFRESH MEMORY MAPPING

As discussed earlier, English text is conveniently stored in refresh memory, since its written pattern directly corresponds to the CRT scan pattern.

For the same controller to display foreign text with a different written pattern than the given scan pattern an alteration is required in either; 1.) the way the data is stored in the refresh RAM, or 2.) the way the refresh data is accessed. For instance, Japanese text displayed on a 72X64 line CRT requires that the first character in the first text line be

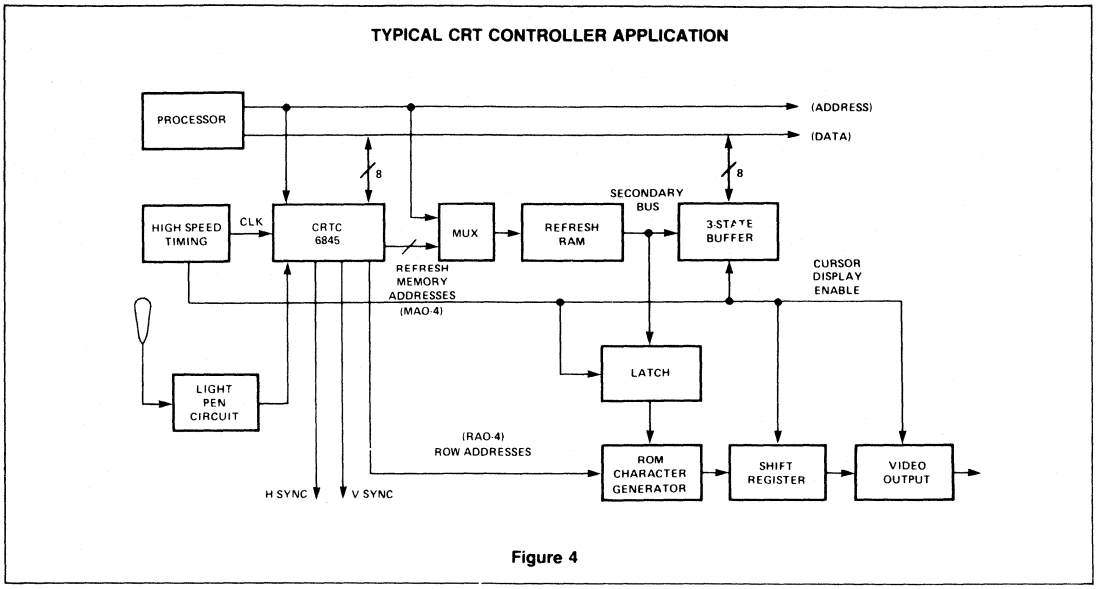


Figure 4

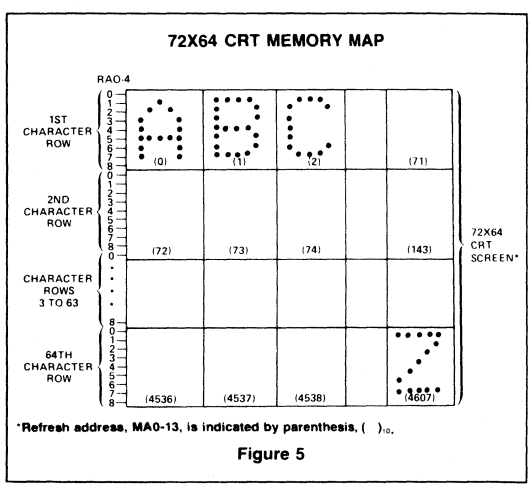


Figure 5

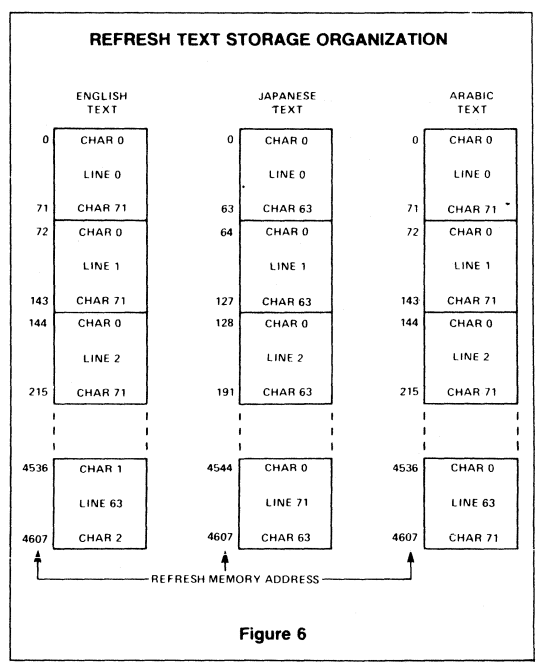


Figure 6

displayed in CRT character position 71 (MA0-13 = 71_{10}); the second in 143; and so on, to the last character in the last displayed vertical text line, where MA0-13 = 4536.

If the processor assumed this scrambling operation in a software map a great deal of overhead program memory and processor time would be consumed. This fact is extenuated when it is necessary to display multiple text patterns on the same monitor screen such as English, Japanese, and Arabic format.

The alternate method of altering the access sequence allows the processor to stack text in the refresh memory in the same way regardless of text. The first text character of the first text line will be placed in refresh memory address 0, the second in address 1, and so on until the end of the text line. The next text line will be stored in a contiguous manner directly after the first line and so on. Figure 6 shows the processor stored refresh text memory for English, Japanese, and Arabic formats.

Given this storage organization, a modified refresh memory address, M0-13, must be supplied to the refresh address multiplexer that follows the sequence shown in Figure 7. This modified address sequence, if synchronized with the correspondingly shown MA0-13 count, arranges the stored text on the CRT monitor in a manner correct for each language format. Japanese or Chinese text will be displayed from top to bottom and right to left; Arabic will be right to left, top to bottom; and English will be from left to right, top to bottom.

MULTI-LANGUAGE REFRESH SEQUENCES

TEXT CHARACTER POSITION (MA0-13)*	ENGLISH TEXT ADDRESS SEQUENCE*	JAPANESE TEXT ADDRESS SEQUENCE*	ARABIC TEXT ADDRESS SEQUENCE*	COUNT SEQUENCE CONDITIONS
0	0	4544	71	
1	1	4480	70	
2	2	4416	69	
3	3	4352	68	
•	•	•	•	RA0-4 $\overline{\text{MAX}}$.
•	•	•	•	
•	•	•	•	
71	71	0	0	RA0-4 MAX.
72	72	4545	143	
73	73	4481	142	
74	74	4417	141	
•	•	•	•	RA0-4 $\overline{\text{MAX}}$.
•	•	•	•	
•	•	•	•	
143	143	1	72	RA0-4 MAX.
•	•	•	•	
•	•	•	•	RA0-4 $\overline{\text{MAX}}$.
•	•	•	•	
4464	4464	4606	4535	RA0-4 MAX.
4465	4465	4542	4534	
4466	4466	4478	4533	
•	•	•	•	RA0-4 $\overline{\text{MAX}}$.
•	•	•	•	
•	•	•	•	
4535	4535	62	4464	
4536	4536	4607	4607	RA0-4 MAX.
4537	4537	4543	4606	
4538	4538	4479	4605	
•	•	•	•	RA0-4 $\overline{\text{MAX}}$. RA0-4 MAX.
•	•	•	•	
•	•	•	•	
4607	4607	63	4536	

* All entries are in decimal

Figure 7

FPLA PRIMER

The FPLA is a programmable logic array that consists of 48 16-input AND gates, and eight 48-input OR gates with user programmable inputs and output polarity. It is functionally equivalent to 528 TTL gates in 196 I.C. packages. The AND and OR gate architecture is shown in Figure 8; a more detailed FPLA description maybe found in the FPLA, 82S100/101 data sheet.

FPLA ADDRESS SEQUENCER IMPLEMENTATION

An FPLA based sequencer is used to perform the three address sequences indicated in Figure 7. As can be seen in Figure 9, the Refresh Memory Sequencer circuit diagram, two FPLAs are used. Three sources of data are required to provide the address sequence; 1.) scan position at the end of a character row (MA0-13 = N(72)-1), 2.) last scan line in a character row (RA0-4 max. = 8), 3.) text address sequence (Text 1, Text 0).

The Address Map Monitor monitors the refresh address bus, MA0-13, and provides an active low output each time Equation 1 is satisfied.

MA0-13 = N(72)-1 Equation: 1

where: N = 1,2,3, ... 64

Careful examination of Equation 1 reveals that the active low output corresponds to the last displayed character position in each character row; MA0-13 = 71, 143, . . . 4607. A straight forward FPLA program requires 64 product terms to implement Equation 1, but with the aid of Signetics' Simultaneous Logic Minimization Program, the required number of product terms is reduced to just 48, the size of a single FPLA!

It can be seen from Figure 7 that this information plus the row address, RA0-4, is enough information to define a single text address sequence. For this particular application the maximum number of scan lines per character row is 9, therefore, the single row address line RA_3 provides the "RA0-4 max (imum count)." logic.

FPLA ARCHITECTURE

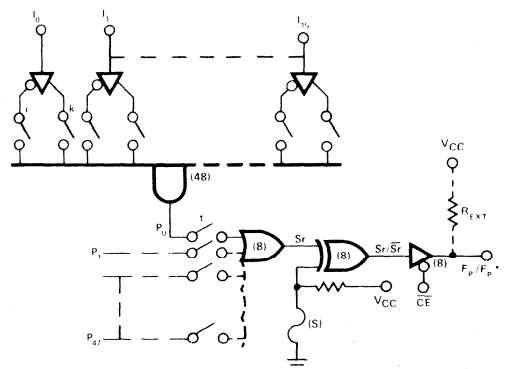


Figure 8

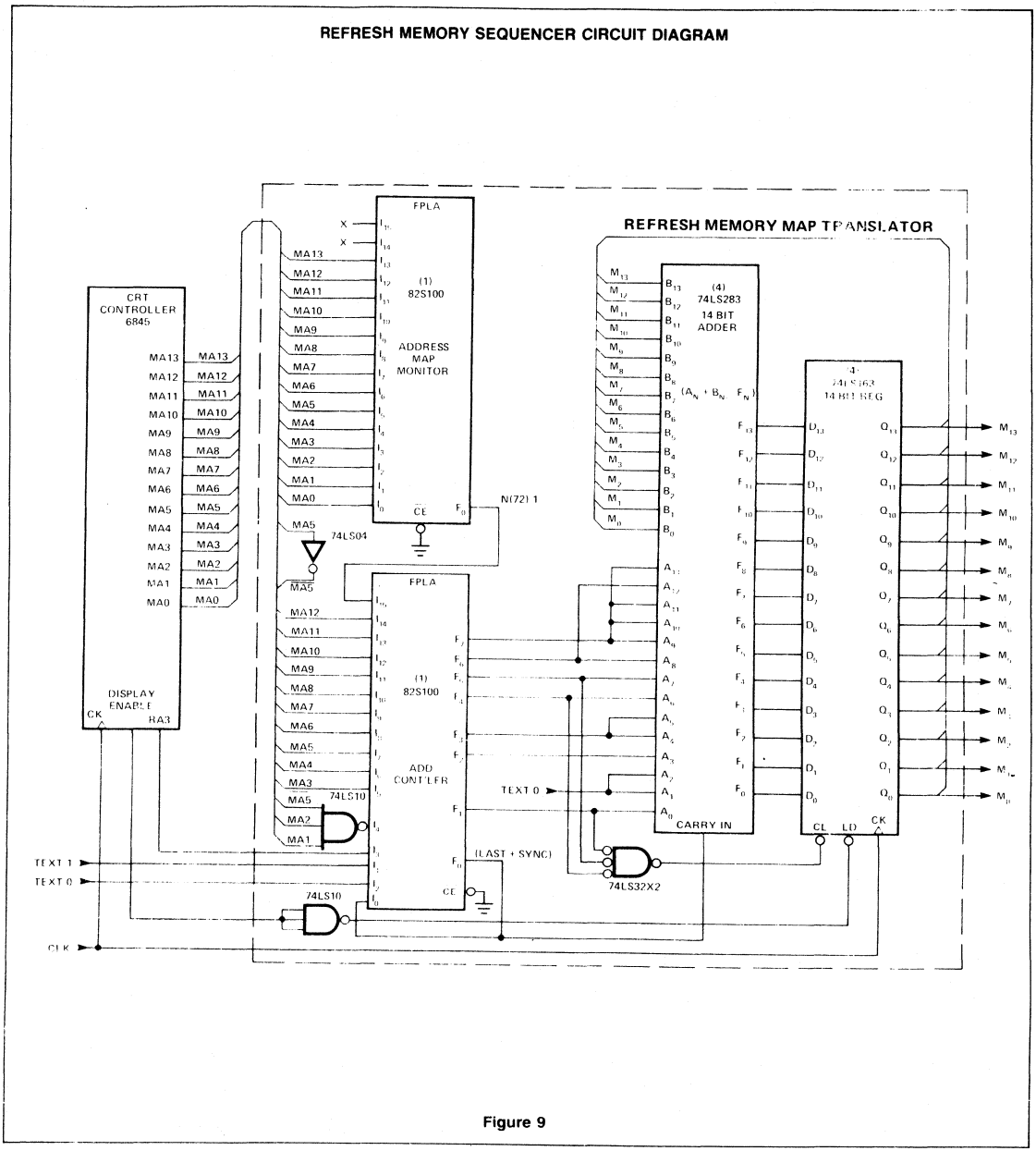


Figure 9

Lines Text 0 and Text 1 represent a binary text selection code where:

DISPLAY FORMAT SELECTION

TEXT 1	TEXT 0	DISPLAY FORMAT
0	0	Japanese
0	1	Arabic
1	0	English

Table 1

The selection code should also be used to select the appropriate Character Generator ROM also.

The actual sequence operation takes place through a process of controlled addition. The four 74LS283's provide the means by which the Add Controller FPLA sequences the modified refresh address, M0-13. Add Controller outputs F1-7 supply the addition increment word. The sequential condition logic indicated in Figure 7 is programmed in this FPLA as can be seen in the Add Controller programming Table; Figure 11.

ADDRESS MAP MONITOR PROGRAM

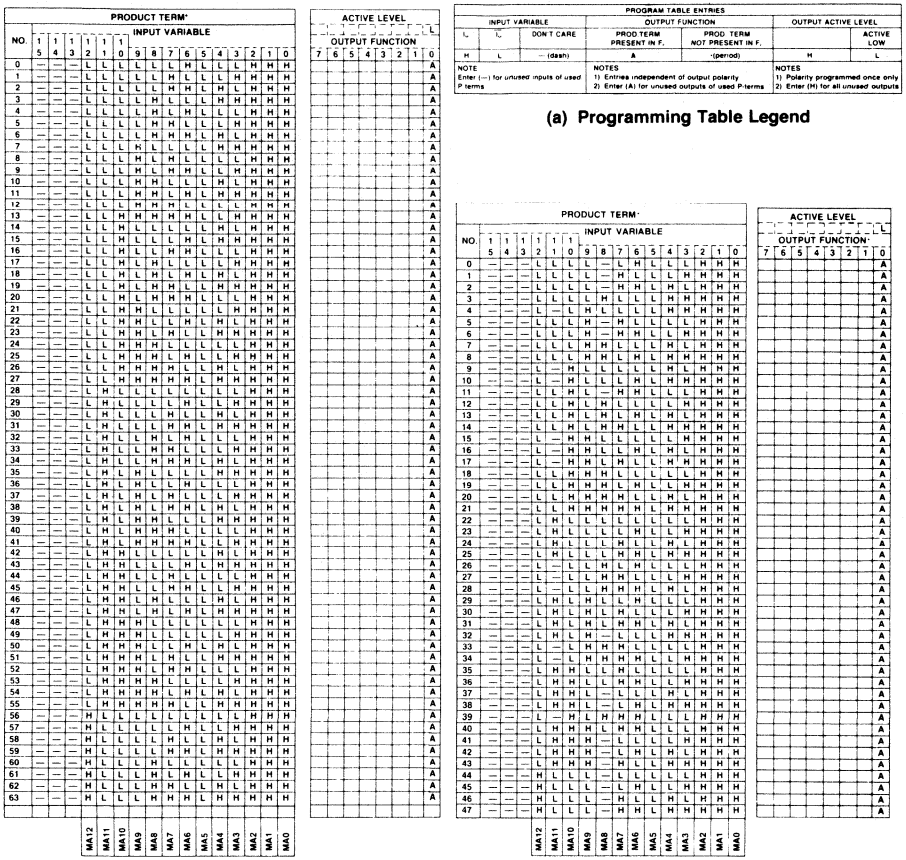


Figure 10

ADD CONTROLLER PROGRAMMING TABLE

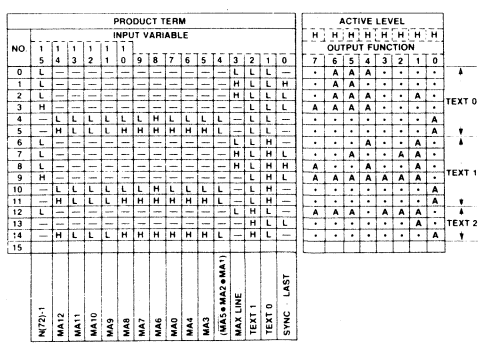


Figure 11

Figure 12 illustrates in flow chart form the conditional logic process the Add Controller follows in a Japanese address sequence. All subtract operations are implemented through twos compliment addition. A similar flow chart maybe constructed for English an(rabic texts.

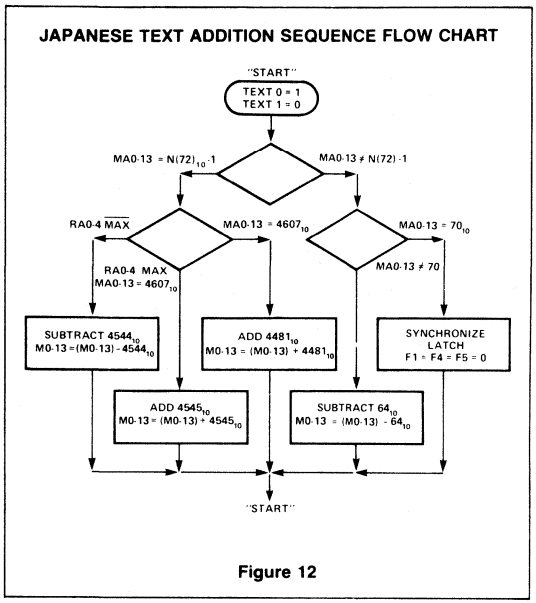


Figure 12

Add Controller output, F_0, provides synchronization logic by monitoring MA0-13, such that M0-13 and MA0-13 track each other as shown in Figure 7. To force synchronization, Add Controller outputs F1, F4, and F5 are set to 0, thereby resetting the 14-bit register to, M0-13 = 0, when the following refresh memory states are detected.

SYNCHRONIZATION CODES

DISPLAY FORMAT	MA0-13*
Japanese	**70_{10}**
Arabic	**70_{10}**
English	**4607_{10}**

*Synchronization takes place only when RA0-4 is maximum.

Table 2

SUMMARY

Where CRT display flexibility is indicated, either as a present design requirement or as a future possibility, the FPLA can provide an economic solution. Alternative comparable solutions require many more parts, power, and P.C. board real estate. For example, a PROM address sequencer implementation would require 64 8K PROMs, 26 Watts, and 17.6 square inches of P.C. board space; whereas the FPLA solution yields 2 FPLAs, 1.2 Watts, and 1.92 square inches of P.C. board space.

REFERENCES

1. National Application Note An-199, National Semiconductor, Santa Clara, California.
2. Motorola MC6845 Data Sheet, Motorola Semiconductors, Austin, Texas.
3. Charles Carinalli, "Slash CRT-Terminal Component Count," Electronic Design 14, July 5, 1978, pp. 88-95.

Field programmable logic arrays

TABLE OF CONTENTS

1. Data Cartridge Tape Drive Controller. 257
2. Serial Impact Printer Controller 261
3. Distributed Controller 263
4. Pneumatic Valve Controller 265
5. Sequential Traffic Controller 266
6. Sequential Control System Using an FPLA .. 267
7. "N" Expandable System Status and Control Logic Unit..... 268
8. Ignition Timing For I/C Engine Using Multi-Dimensional Curve Fit 269
9. Using FPLAs and SRGs to Store Huffman Codes 270
10. N-Out-Of-8 Decode 274
11. 16-Bit Shift and Logic Unit 278
12. FPLA Modifies Sequential Circuit Design ... 283
13. Bandswitching Code Converter..... 284
14. RF-Channel Control Selector 285
15. Serial Data Filter 289
16. 8-Channel Pulse Width Discriminator..... 291
17. Timing Decoder & Sequence Controller For Data Acquisition System 292
18. Controller for Single TDM Data Channel 297
19. ATC Transponder Encoder 300
20. Mobile FM-VHF Transceiver Uses FPLA for Quick Channel Selection 301
21. Programmable Divider for Frequency Synthesizer 303
22. Programmable Frequency Synthesizer 306
23. An IEEE-488 to Speech Synthesizer Interface 306
24. Function Decoding in Camac Modules..... 308
25. Programmable Waveform Generator..... 309
26. Programmable Function Generator 311
27. Register Port Selector for 4-Bit Slice Microprocessors..... 312
28. OP-Code Decoder for PDP-11 MPCU 314
29. Target Instruction Decoder for Micro-programmed Emulator 317
30. Single or Block Address Decoding for M6800μP..... 319
31. Interrupt Priority Control for M6800μP 319
32. Unibus Address Monitor..... 321
33. Hardware-Macro Generator 322
34. FPLA Enhances Pipelined Process Architecture 322
35. I/O Port Decoder 323
36. Peripheral Address Decoder 324
37. Memory Address Trap 324
38. Maskable Interrupt Vector Generator 324
39. 15-Input Priority Encoder and I.D. Generator..... 326
40. FPLA Simplifies Alpha-Numeric Display Operation..... 326
41. An Economical Digital Display Conditioner.. 329
42. Rate Measurement of Low Frequency Events 335
43. Logic Analyzer 337
44. Combination Sync & Video Generator 338
45. Court and Scoreband Generator for Ping Pong T.V. Game 342
46. Intelligent Tape Reader to T.V. Video Display Interface..... 343
47. Digital Message Decoder 344
48. Calculator Chip Interface..... 346
49. Microprocessor to Calculator-Chip Interface . 349
50. Hand-Held ASCII Keyboard..... 351
51. 7-Segment to BCD Converter..... 353
52. Decimal Point & Legend Display Driver..... 354
53. One to Sixteen Digit Combination Lock 355
54. Data Security Encoder..... 356
55. Calendar Clock Date Reminder..... 357
56. Real Time Preventive Maintenance Monitor.. 358

Signetics' FPLAs are logic elements with memory, and can be viewed in two basic ways.

In terms of logic, the FPLA is a two level AND-OR, AND-NOR *combinatorial* logic element, consisting of a system of logic gates with programmable inputs and outputs as shown in Figure 1. These, by means of on-chip programmable connectors, enable the user to quickly implement 8 logic functions with a maximum of 48 product (AND) terms, involving up to 16 input variables.

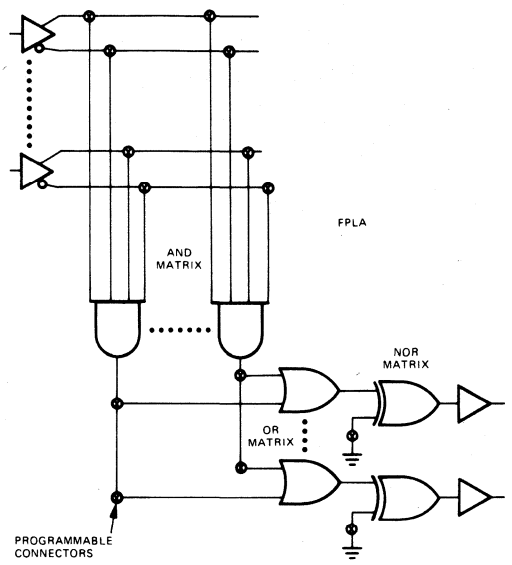


Figure 1: FPLA as a two-level, programmable logic element.

A more detailed organization of the FPLA is shown in Figure 2. The first logic level, the AND matrix, consists of 48 resistor-diode AND gates each connected to 16 True and Complement inputs via 32 fusible nichrome links. The second logic level, the OR matrix, consists of 8 emitter follower OR gates each connected by fusible links to all 48 outputs from the AND matrix. The output of each of these 8 OR gates is in turn buffered through an EX-OR gate which allows changing the logic to NOR via a fusible link to ground.

If your design involves a random logic structure, the FPLA can be used in place of discrete gates and wires. You gain

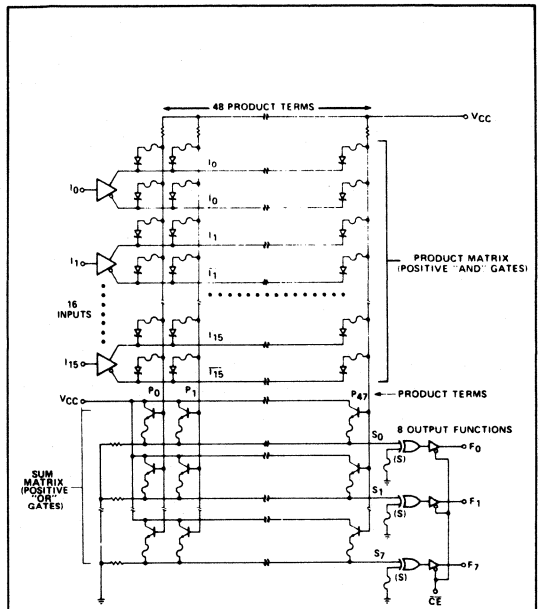


Figure 2: Logic structure of FPLA, illustrating AND, OR, and EX-OR arrays.

total flexibility from the immediate availability of your logic function for which you need only use as much of the FPLA as necessary for your particular application, with the rest available for later expansion or modification. But, if your design needs to be structured more like a memory, then the FPLA can be used in another, perhaps more effective way.

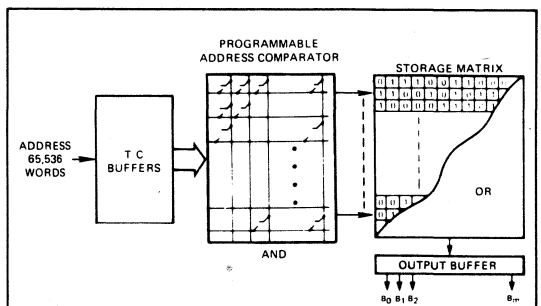


Figure 3: FPLA as a conditionally addressable 48 x 8 memory.

From the partitioning shown in Figure 3, the FPLA can also be viewed as a conditionally addressable memory; one in which the AND matrix functions as a programmable address comparator, recognizing only the preprogrammed address combinations used to activate the system, and ignoring all others. Plus, an OR matrix which then functions as a storage array for 48 output words, each containing 8 bits representing active or idle system commands. Or, to put it another way, you have a memory system that can logically scan a total address field up to 65,536 words deep, to linearly select down to 48 replies randomly scattered within that address space.

No matter how you look at it, generally FPLAs are effectively used in design situations where you have many input variables and few active logic states; and, with a maximum access time of 50ns, the FPLA is a practical alternative to the long logic chains necessary when dealing with several input variables.

The following example is a brief, but concise illustration of how to integrate random logic with discrete gates into a Signetics' FPLA.

Given the set of logic equations $F_{1\text{-}4}$ below:

$$F_1 = X_1 + \bar{X}_2\bar{X}_3 \qquad F_3 = X_3 + \bar{X}_1\bar{X}_2$$

$$F_2 = X_2 + \bar{X}_1\bar{X}_3 \qquad F_4 = \bar{X}_1X_3 + \bar{X}_1\bar{X}_2$$

These can be implemented with discrete gates as in the AND-OR-NOT logic network of Figure 4.

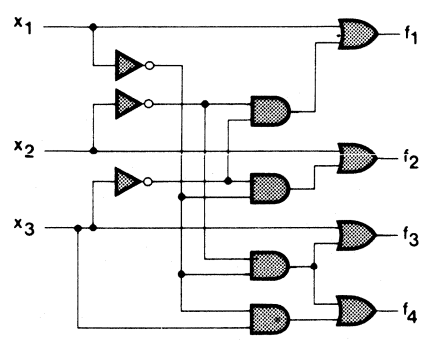


Figure 4: Discrete gate logic network for equation set $F_{1\text{-}4}$.

This method is practical for simple systems; but in more complex applications, it soon produces a distributed logic network with many I.C. packages and types, difficult to design, troubleshoot and modify.

On the other hand, the same set of equations can be easily coded in an FPLA Program Table and programmed in a device using inexpensive field equipment. Typically, F_1 would require the FPLA to contain the fused link pattern shown in Figure 5, as specified in the accompanying Program Table slice. Overall, all four logic functions would use 3 inputs, 4 outputs, and 7 product terms of the FPLA, leaving remaining resources spare for later modifications.

FPLA PROGRAM TABLE

PRODUCT TERM NO	INPUT VARIABLE (I_m) 11 5	10 4	9 3	8 2	7 1	6 0	5 9	4 8	3 7	2 6	1 5	0 4	3	2	1	0	ACTIVE LEVEL H H H H	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H						•	•	•	A
1	–	–	–	–	–	–	–	–	–	–	–	–	L	L	–							•	•	•	A

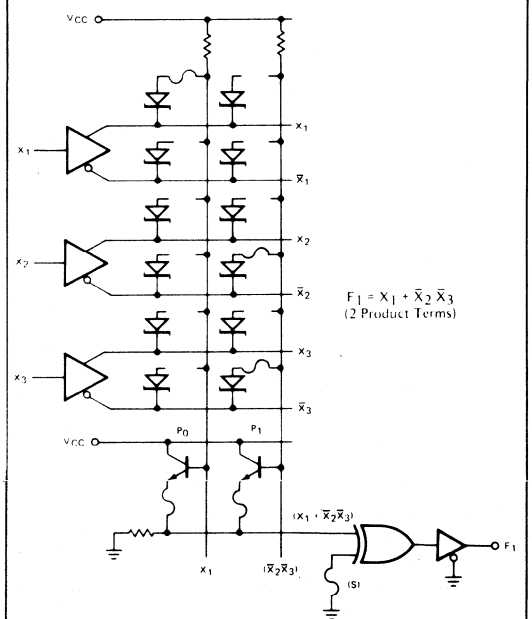


Figure 5: Integrating logic with FPLAs.

For example, if it becomes necessary to change the X_1 product term in F_1 to $\overline{X}_1$, deleting the wrong product term and adding the new one becomes a trivial task, as indicated in the modified pattern and revised Program Table of Figure 6.

FPLA PROGRAM TABLE

PRODUCT TERM															ACTIVE LEVEL							
	INPUT VARIABLE (I_m)																					
NO.															OUTPUT FUNCTION							
	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H					•	•	•	
1	–	–	–	–	–	–	–	–	–	–	–	–	–	L	L	–					•	•	•	A
2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L					•	•	•	A

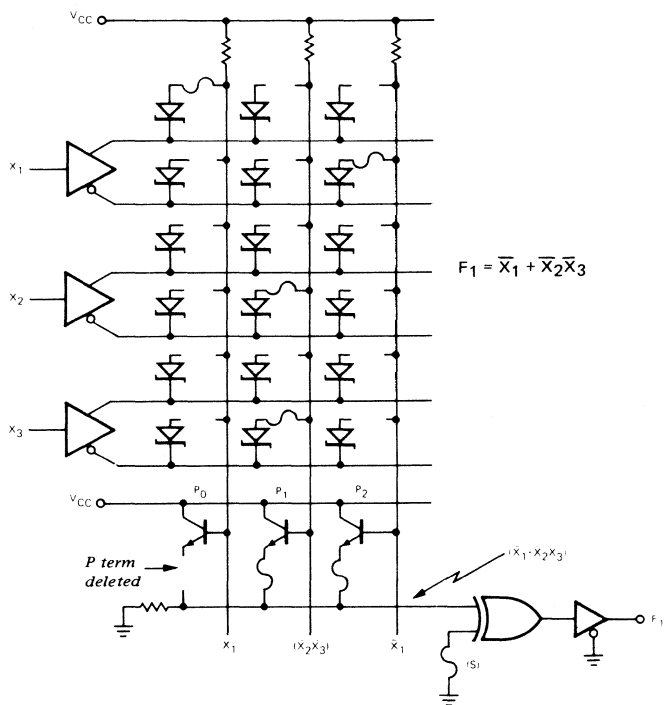


Figure 6: Modifiying random logic with FPLAs. F_1 modified by deleting term X_1 in the OR matrix and adding new term $\overline{X}_1$.

The circuit and associated state diagram in Figures 1 and 2 illustrate a 14-state logic machine which controls the tape motion of a Data Cartridge Tape Drive (Qantex Model 600). It allows the tape drive to be operated externally by a computer and internally according to the routines programmed into the FPLA. The FPLA contains all of the combinational logic required to implement the external commands and internal routines. And, by reducing total IC count, it contributed to reduce from 2 to 1 the number of circuit boards used in the tape drive.

In the circuit of Figure 1, all inputs are derived from I/O commands, the status of the tape drive, and the output of the four J-K flip-flops. The states of the flip-flops determine which state the logic machine is in as defined in Figure 3. The outputs are used to operate the velocity servo which drives the Data Cartridge to form I/O status signals, and to enable the writing of data.

The current state of the logic machine (Q1, Q2, Q3, Q4,) is decoded by the FPLA and combined with the other input signals to determine which state the machine will be switched to next. The possible sequences are described by the Flow Diagram of Figure 2. A jump to a following state can occur when the intervening conditions are true. All desired jumps are programmed in the FPLA Program Table of Figure 4. Jumps occur on the trailing edge of the clock pulse. The state codes (Figure 3) are decoded by the PROM

SCHEMATIC OF TAPE DRIVE CONTROLLER

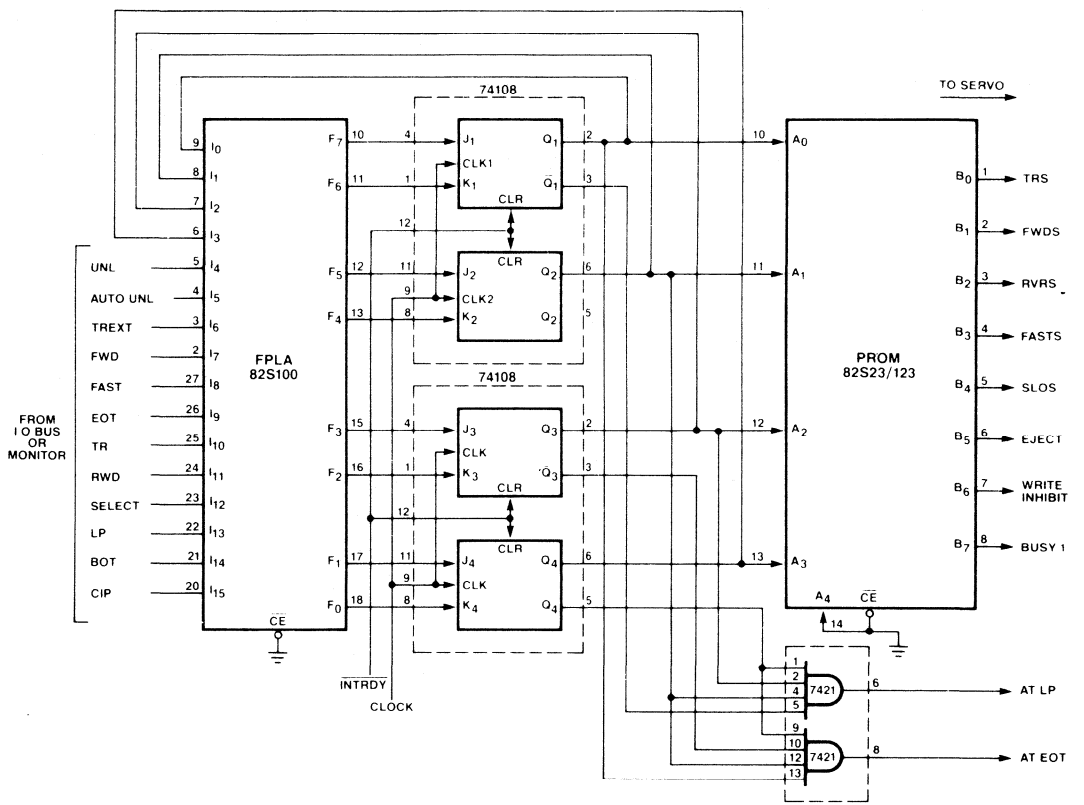


Figure 1.

FLOW DIAGRAM OF CONTROLLER

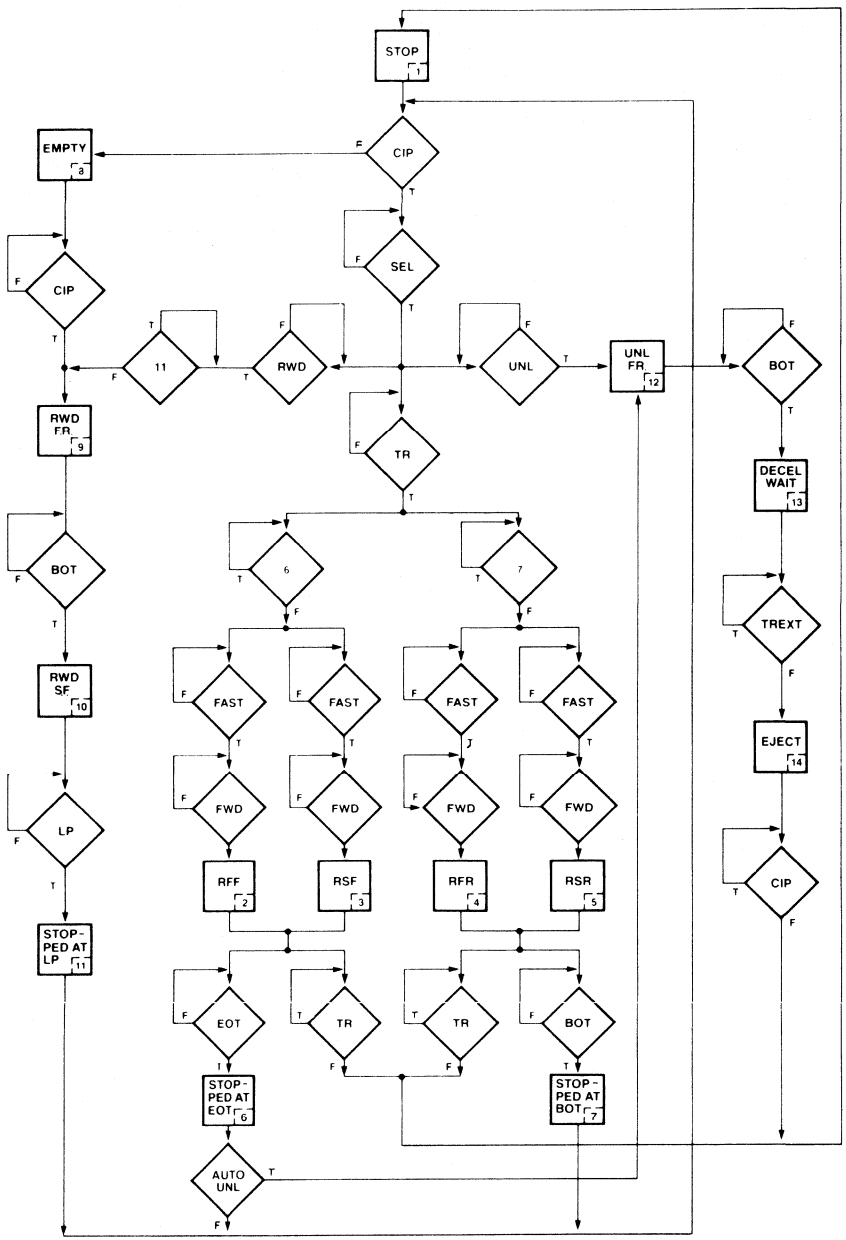


Figure 2.

STATE CODE ASSIGNMENTS

STATE		CODE			
NO.	NAME	Q1	Q2	Q3	Q4
1	STOP	0	0	0	0
2	RFF	1	0	0	0
3	RSF	0	1	0	0
4	RFR	0	0	1	0
5	RSR	0	0	0	1
6	STOPPED AT EOT	1	1	0	0
7	STOPPED AT BOT	0	0	1	1
8	EMPTY	1	0	1	0
9	RWD FR	1	1	1	0
10	LDSF	0	1	1	1
11	STOPPED AT LP	0	1	1	0
12	UNLFR	1	1	0	1
13	DECEL WAIT	1	0	1	1
14	EJECT	1	0	0	1

Figure 3.

and the two AND gates with the assignment tabulated in Figure 5. The decoded outputs provide the various signals needed to operate the tape drive.

Whenever the tape drive power is turned on, or an interlock opened, the tape drive is required to be stopped. This defines state 1, STOP, which has been assigned the code 0000. The state is achieved by using $\overline{\text{INTRDY}}$ to clear the four J-K flip-flops. Once set to STOP, operation at normal write speeds can occur when the following set of conditions are simultaneously satisfied, as diagrammed in Figure 2:

a. Data cartridge is in place: CIP true
b. Tape Drive has been addressed: SEL true
c. Tape has been commanded to run: TR true
d. The logic machine is *not* in state 6: 6 false
e. Tape should move at slow speed: $\overline{\text{FAST}}$ true (=slow)
f. Tape should move forward: FWD true

Then the logic machine will jump from state 1 to 3 (RSF). The preceding conditions are programmed in jump 1-3 of Figure 4. the outputs implemented by the new state are described in Figure 5 under state 3. Reading across the line for state 3 of Figure 5, notice that the servo will be driven (TRS true) in the forward direction (FWDS true) at the slow speed (SLOS true).

After data has been either written or read, the tape drive is commanded to stop (TR false). This allows a jump from state 3 (run-slow-forward) to state 1 (STOP). The jump from state 3 to 1 is charted in Figure 2 and programmed in Figure 4. Figure 5 shows that the only output active in state 1 is Write Inhibit.

By similar agruments, the tape drive can be run either fast or slow in either forward or reverse direction (states 2, 4 and 5).

When the end of tape is reached (EOT true), the tape drive is stopped. This is implemented by a jump from 2-6 or 3-6. Once in state 6, the tape drive can no longer move in the forward direction because of the state 6 false condition preceding states 2 and 3. If auto-unload is true, the drive will automatically rewind (state 12), wait for tape to decelerate (state 13), eject the tape cartridge (state 14) and stop (state 1). If auto-unload is false, the tape drive must wait for either a rewind command (RWD), an unload command (UNL), or a reverse command.

If the tape should be moved in the reverse direction until the beginning of tape is reached, the tape drive is stopped. This is implemented by a jump from 4 to 6 to 7. Once in state 7, the tape drive can no longer move in the reverse direction because of the state 7 false condition preceding states 4 and 5. The tape drive will remain at state 7 (STOPPED AT BOT) until RWD, UNL, or FWD command is given.

If no Data Cartridge is in place (CIP false) when the tape drive is turned on, the logic machine will jump from 1 to 8 (EMPTY). When a cartridge is installed, CIP goes true implementing a jump to 8 to 9 (RWD FR). In state 9, the tape will be rewound in fast reverse until a BOT mark is reached. BOT true implements a jump from 9 to 10 (LDSF). Tape now runs at slow speed in the forward direction until load point (LP) is reached. LP true implements a jump from 10 to 11 (STOPPED AT LP). From state 11, a forward, reverse, or UNL command can be implemented. RWD cannot be implemented from state 11 because of the state 11 condition preceding state 9. This keeps RWD from being needlessly repeated.

The logic machine implements and controls the following tape drive operations and routines:

a. move tape fast forward
b. move tape slow forward
c. move tape fast reverse
d. move tape slow forward
e. bring tape to load point when Data Cartridge installed
f. rewind tape to load point
g. rewind tape to BOT and eject cartridge in response to UNL command
h. rewind tape to BOT and eject cartridge in response to an AUTO UNL true condition.

NOTES:

1) The rewind and unload routines are self-completing and need no further implementation once started.
2) BOT, EOT, LP, are pulses.

3) The ROM output line pairs of FWRS and RVRS, and FASTS and SLOS are required to operate latches which remember tape speed and direction during deceleration.

4) Whenever a Data Cartridge is removed, a CIP false pulse is generated which temporarily resets the logic machine to STOP.

FPLA PROGRAM TABLE FOR IMPLEMENTING ALL STATE JUMPS IN THE FLOW-CHART

STATE JUMP	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1-2	0	H	–	–	H	–	H	–	H	H	–	–	–	L	L	L	L
1-3	1	H	–	–	H	–	H	–	L	H	–	–	–	L	L	L	L
1-4	2	H	–	–	H	–	H	–	H	L	–	–	–	L	L	L	L
1-5	3	H	–	–	H	–	H	–	L	L	–	–	–	L	L	L	L
1-8	4	L	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L
1-9	5	H	–	–	H	H	–	–	–	–	–	–	–	L	L	L	L
1-12	6	H	–	–	H	–	–	–	–	–	–	–	H	L	L	L	L
2-1	7	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H
2-6	8	–	–	–	–	–	–	H	–	–	–	–	–	L	L	L	H
3-1	9	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	L
3-6	10	–	–	–	–	–	–	H	–	–	–	–	–	L	L	H	L
4-1	11	–	–	–	–	–	L	–	–	–	–	–	–	L	H	L	L
4-7	12	–	H	–	–	–	–	–	–	–	–	–	–	L	H	L	L
5-1	13	–	–	–	–	–	L	–	–	–	–	–	–	H	L	L	L
5-7	14	–	H	–	–	–	–	–	–	–	–	–	–	H	L	L	L
6-4	15	H	–	–	H	–	H	–	H	L	–	L	–	L	L	H	H
6-5	16	H	–	–	H	–	H	–	L	L	–	L	–	L	L	H	H
6-9	17	H	–	–	H	H	–	–	–	–	–	L	–	L	L	H	H
6-12	18	H	–	–	H	–	–	–	–	–	–	L	H	L	L	H	H
6-12	19	–	–	–	–	–	–	–	–	–	–	H	–	L	L	H	H
7-2	20	H	–	–	H	–	H	–	H	H	–	–	–	H	H	L	L
7-3	21	H	–	–	H	–	H	–	L	H	–	–	–	H	H	L	L
7-9	22	H	–	–	H	H	–	–	–	–	–	–	–	H	H	L	L
7-12	23	H	–	–	H	–	–	–	–	–	–	–	H	H	H	L	L
8-9	24	H	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H
9-10	25	–	H	–	–	–	–	–	–	–	–	–	–	L	H	H	H
10-11	26	–	–	H	–	–	–	–	–	–	–	–	–	H	H	H	L
11-2	27	H	–	–	H	–	H	–	H	H	–	–	–	L	H	H	L
11-3	28	H	–	–	H	–	H	–	L	H	–	–	–	L	H	H	L
11-4	29	H	–	–	H	–	H	–	H	L	–	–	–	L	H	H	L
11-5	30	H	–	–	H	–	H	–	L	L	–	–	–	L	H	H	L
11-12	31	H	–	–	H	–	–	–	–	–	–	–	H	L	H	H	L
12-13	32	–	H	–	–	–	–	–	–	–	–	–	–	H	L	H	H
13-14	33	–	–	–	–	–	–	–	–	–	L	–	–	H	H	L	H
14-1	34	L	–	–	–	–	–	–	–	–	–	–	–	H	L	L	H
I/O ASSIGNMENT		CIP	BOT	LP	SEL	RWD	TR	EOT	FAST	FWD	TR-EXT	AUTO UNL	UNL	O_4	O_3	O_2	O_1

ACTIVE LEVEL	H	H	H	H	H	H	H	H
OUTPUT FUNCTION	7	6	5	4	3	2	1	0
	K4	J4	K3	J3	K2	J2	K1	J1
1-2	A	●	A	●	A	●	●	A
1-3	A	●	A	●	●	A	A	●
1-4	A	●	●	A	A	●	A	●
1-5	●	A	A	●	A	●	A	●
1-8	A	●	●	A	A	●	●	A
1-9	A	●	●	A	●	A	●	A
1-12	●	A	A	●	●	A	●	A
2-1	A	●	A	●	A	●	A	●
2-6	A	●	A	●	●	A	●	A
3-1	A	●	A	●	A	●	A	●
3-6	A	●	A	●	●	A	●	A
4-1	A	●	A	●	●	A	●	A
4-7	●	A	●	A	A	●	A	●
5-1	A	●	A	●	A	●	A	●
5-7	●	A	●	A	A	●	A	●
6-4	A	●	●	A	A	●	A	●
6-5	●	A	A	●	A	●	A	●
6-9	A	●	●	A	●	A	●	A
6-12	●	A	A	●	●	A	●	A
6-12	●	A	A	●	●	A	●	A
7-2	A	●	A	●	A	●	●	A
7-3	A	●	A	●	●	A	A	●
7-9	A	●	●	A	●	A	●	A
7-12	●	A	A	●	●	A	●	A
8-9	A	●	●	A	●	A	●	A
9-10	●	A	●	A	●	A	A	●
10-11	A	●	●	A	●	A	A	●
11-2	A	●	A	●	A	●	●	A
11-3	A	●	A	●	●	A	A	●
11-4	A	●	●	A	A	●	A	●
11-5	●	A	A	●	A	●	A	●
11-12	●	A	A	●	●	A	●	A
12-13	●	A	●	A	A	●	●	A
13-14	●	A	A	●	A	●	●	A
14-1	A	●	A	●	A	●	A	●

Figure 4.

SERVO COMMANDS GENERATED BY DECODING STATE REGISTER WITH PROM AND GATES

INPUT					OUTPUT									
STATE					TRS	FWDS	RVRS	FASTS	SLOS	EJECT	WRITE INH	BUSY 1	AT LP	AT EOT
#	CODE													
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	1	0	0	0	1	1	0	1	0	0	1	0	0	0
3	0	1	0	0	1	1	0	0	1	0	0	0	0	0
4	0	0	1	0	1	0	1	1	0	0	1	0	0	0
5	0	0	0	1	1	0	1	0	1	0	1	0	0	0
6	1	1	0	0	0	0	0	0	0	0	1	0	0	1
7	0	0	1	1	0	0	0	0	0	0	1	0	0	0
8	1	0	1	0	0	0	0	0	0	0	1	0	0	0
9	1	1	1	0	1	0	1	1	0	0	1	1	0	0
10	0	1	1	1	1	1	0	0	1	0	1	1	0	0
11	0	1	1	0	0	0	0	0	0	0	1	0	1	0
12	1	1	0	1	1	0	1	1	0	0	1	1	0	0
13	1	0	1	1	0	0	0	0	0	0	1	0	0	0
14	1	0	0	1	0	0	0	0	0	1	1	0	0	0
PROM PIN #'s	10	11	12	13	1	2	3	4	5	6	7	8	AND GATE 1	AND GATE 2

Figure 5.

SERIAL IMPACT PRINTER CONTROLLER 2

The circuit shown in Figure 1 is a sequential controller which implements all functions needed to control a dot matrix serial printer, and provides a simple, inexpensive, and reliable control circuit useful in many design environments.

The controller consists of an 82S100 FPLA as a program decoder, and two 74175 quad D-type flip-flops as feedback elements.

Parallel ASCII data is presented to the inputs of the FPLA along with an associated Data Strobe. If data detected by the FPLA is a printable character (as opposed to a control function) it will be entered into the buffer memory.

The buffer memory consists of two 3351's configured as an 80 x 9 F1F0, and will hold one full line of 80 characters.

When the memory is full, or if the FPLA detects an ASCII "carriage return," the controller will begin a print cycle, and print out the data stored in memory. The first character in memory will be presented to the inputs of the MK2302P Character Generator. The controller will strobe the Output Enable of the character generator, increment the column counter clock, step the print head, test the Counter Output of the character generator, and shift the next piece of data from the memory if the Counter Output is true. This sequence will continue until the memory is empty, as signalled by the Output Ready line of the second 3351.

The controller will then generate Carriage Return, Line Feed, and Ribbon Advance signals via the 7442 decoder. The print cycle is then complete and the controller awaits new data.

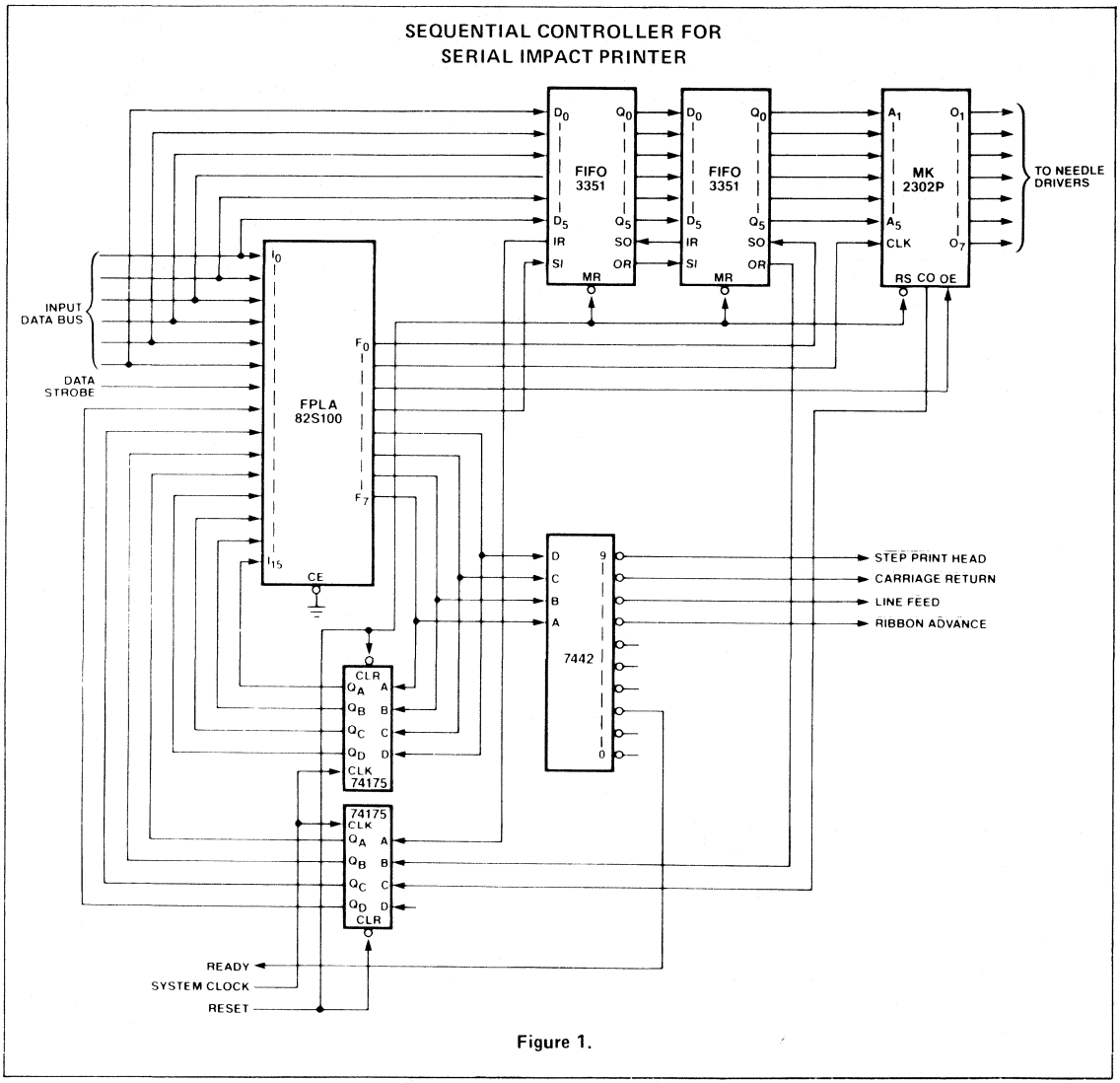

SEQUENTIAL CONTROLLER FOR
SERIAL IMPACT PRINTER
FIFO 3351
FIFO 3351
MK 2302P
TO NEEDLE DRIVERS
INPUT DATA BUS
DATA STROBE
FPLA 82S100
7442
STEP PRINT HEAD
CARRIAGE RETURN
LINE FEED
RIBBON ADVANCE
74175
74175
READY
SYSTEM CLOCK
RESET

Figure 1.

In industrial or commercial processes it is often necessary to provide many contact closure type control points and/or contact sense points. A simple control system of this type can be implemented using the circuit of Figure 1, and any serial compatible send/receive terminal such as a Teletype or CRT controller.

The circuit has the following features:

A. Up to 56 jumper selectable addressed units may be connected in series to a single control circuit. Address are in octal, excluding 00,11,22,33,44,55,66,77.
B. Up to 16 contact closure output points (TTL level) for each unit. Points A thru P functions ("S"et and "R"eset).
C. Up to 16 contact closure input points (TTL level) for each unit. Points A thru P function ("?" Read).

The serial IN(SI) and serial OUT(SO) drivers are not defined in this circuit. A typical output communications sequence is (#34AS), where:

"#" = Attention character that resets the address function in all units on the serial communications circuit.
"3" = The 1st address character
"4" = The 2nd address character
"A" = The control point
"S" = The control function

When sensing a contact point, replace the "R" or "S" function with a "?" function to read the sense point. The unit responds (in ASCII) with a "1" for a closed contact and a "0" for an open contact point.

Several different modifications of this basic circuit are possible to send BCD or ASCII data using the same or a modified decoding scheme. The decoding function for this circuit is shown in the FPLA program table of Figure 2.

DISTRIBUTED PROCESS CONTROLLER

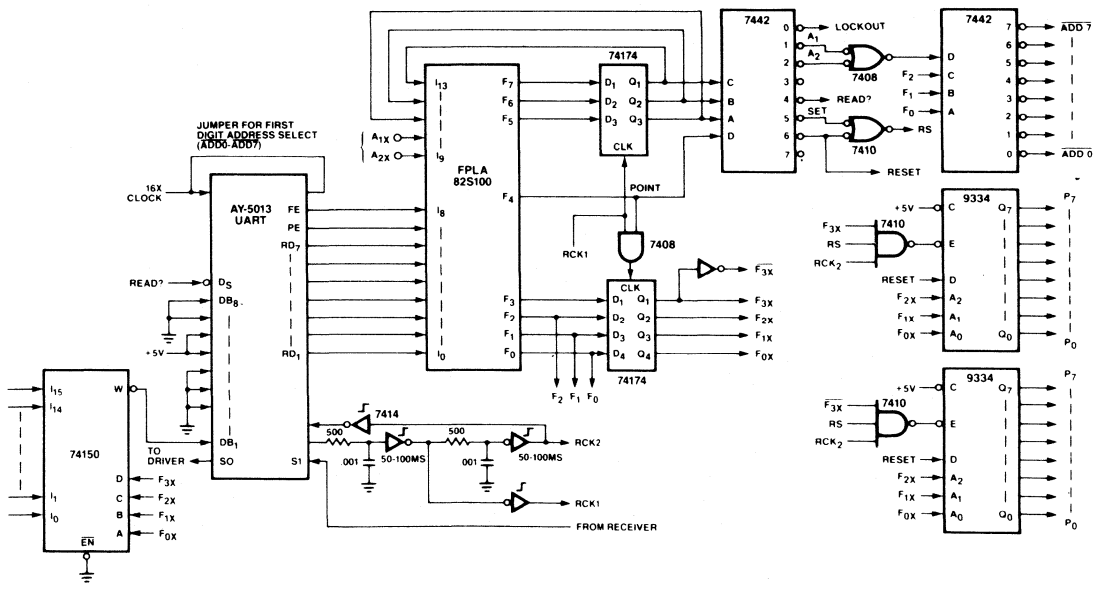


Figure 1.

FPLA PROGRAM TABLE

COMMENT	NO	PRODUCT TERM INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL L L L H H H H H OUTPUT FUNCTION 7	6	5	4	3	2	1	0
=	0	–	–	–	–	–	–	–	L	L	L	H	L	L	L	H	H	A	A	•	•	•	•	•	•
TRANS. ERR.	1	–	–	–	–	–	–	–	H	H	–	–	–	–	–	–	–	A	A	A	•	•	•	•	•
A1 ERR.	2	–	–	L	L	H	–	L	–	–	–	–	–	–	–	–	–	A	A	A	•	•	•	•	•
A2 ERR.	3	–	–	L	H	L	L	–	–	–	–	–	–	–	–	–	–	A	A	A	•	•	•	•	•
"S" ET	4	–	–	H	–	–	H	H	L	L	L	H	H	L	L	H	H	•	A	•	•	•	•	•	•
"R" ESET	5	–	–	H	–	–	H	H	L	L	L	H	H	L	L	H	L	•	•	A	•	•	•	•	•
READ "?"	6	–	–	H	–	–	H	H	L	L	H	H	H	H	H	H	H	•	A	A	•	•	•	•	•
A	7	–	–	H	–	–	H	H	L	L	L	L	L	L	L	L	H	•	•	•	A	•	•	•	A
B	8	–	–	H	–	–	H	H	L	L	L	L	L	L	L	H	L	•	•	•	A	•	•	A	•
C	9	–	–	H	–	–	H	H	L	L	L	L	L	L	L	H	H	•	•	•	A	•	•	A	A
D	10	–	–	H	–	–	H	H	L	L	L	L	L	L	H	L	L	•	•	•	A	•	A	•	•
E	11	–	–	H	–	–	H	H	L	L	L	L	L	L	H	L	H	•	•	•	A	•	A	•	A
F	12	–	–	H	–	–	H	H	L	L	L	L	L	L	H	H	L	•	•	•	A	•	A	A	•
G	13	–	–	H	–	–	H	H	L	L	L	L	L	L	H	H	H	•	•	•	A	•	A	A	A
H	14	–	–	H	–	–	H	H	L	L	L	L	L	H	L	L	L	•	•	•	A	A	•	•	•
I	15	–	–	H	–	–	H	H	L	L	L	L	L	H	L	L	H	•	•	•	A	A	•	•	A
J	16	–	–	H	–	–	H	H	L	L	L	L	L	H	L	H	L	•	•	•	A	A	•	A	•
K	17	–	–	H	–	–	H	H	L	L	L	L	L	H	L	H	H	•	•	•	A	A	•	A	A
L	18	–	–	H	–	–	H	H	L	L	L	L	L	H	H	L	L	•	•	•	A	A	A	•	•
M	19	–	–	H	–	–	H	H	L	L	L	L	L	H	H	L	H	•	•	•	A	A	A	•	A
N	20	–	–	H	–	–	H	H	L	L	L	L	L	H	H	H	L	•	•	•	A	A	A	A	•
O	21	–	–	H	–	–	H	H	L	L	L	L	L	H	H	H	H	•	•	•	A	A	A	A	A
P	22	–	–	H	–	–	H	H	L	L	L	H	H	L	L	L	L	•	•	•	A	•	•	•	•
I CHARACTER ADDRESS DECODE	23	–	–	L	L	H	–	H	L	L	L	H	H	L	L	L	L	•	•	•	•	•	•	•	•
	24	–	–	L	L	H	–	H	L	L	L	H	H	L	L	L	H	A	•	A	•	•	•	•	A
	25	–	–	L	L	H	–	H	L	L	L	H	H	L	L	H	L	A	•	A	•	•	•	A	•
	26	–	–	L	L	H	–	H	L	L	L	H	H	L	L	H	H	A	•	A	•	•	•	A	A
	27	–	–	L	L	H	–	H	L	L	L	H	H	L	H	L	L	A	•	A	•	•	A	•	•
	28	–	–	L	L	H	–	H	L	L	L	H	H	L	H	L	H	A	•	A	•	•	A	•	A
	29	–	–	L	L	H	–	H	L	L	L	H	H	L	H	H	L	A	•	A	•	•	A	A	•
	30	–	–	L	L	H	–	H	L	L	L	H	H	L	H	H	H	A	•	A	•	•	A	A	A
II CHARACTER ADDRESS DECODE	31	–	–	L	H	L	H	–	L	L	L	H	H	L	L	L	L	•	A	A	•	•	•	•	•
	32	–	–	L	H	L	H	–	L	L	L	H	H	L	L	L	H	•	A	A	•	•	•	•	A
	33	–	–	L	H	L	H	–	L	L	L	H	H	L	L	H	L	•	A	A	•	•	•	A	•
	34	–	–	L	H	L	H	–	L	L	L	H	H	L	L	H	H	•	A	A	•	•	•	A	A
	35	–	–	L	H	L	H	–	L	L	L	H	H	L	H	L	L	•	A	A	•	•	A	•	•
	36	–	–	L	H	L	H	–	L	L	L	H	H	L	H	L	H	•	A	A	•	•	A	•	A
	37	–	–	L	H	L	H	–	L	L	L	H	H	L	H	H	L	•	A	A	•	•	A	A	•
	38	–	–	L	H	L	H	–	L	L	L	H	H	L	H	H	H	•	A	A	•	•	A	A	A
ILLEGAL CODES	39	–	–	–	–	–	–	–	–	L	L	L	–	–	–	–	–	A	A	A	•	•	•	•	•
	40	–	–	–	–	–	–	–	–	L	L	H	–	–	–	–	–	A	A	A	•	•	•	•	•
	41	–	–	–	–	–	–	–	–	L	H	L	–	–	–	–	–	A	A	A	•	•	•	•	•
	42	–	–	–	–	–	–	–	–	H	H	L	–	–	–	–	–	A	A	A	•	•	•	•	•
	43	–	–	–	–	–	–	–	–	H	H	H	–	–	–	–	–	A	A	A	•	•	•	•	•

Figure 2.

The output states of solenoid-controlled, air-operated process valves, are a function of logical combinations of input variables, which are gated by contacts to signify various process modes, sequence steps, other valve positions, flow, temperature, pressure, level, etc.

Occasionally it is desirable to change the function or equation of process valves. The FPLA provides an elegant yet cost effective solution for reprogramming these valves and/or other process devices. In the circuit of Figure 1, the set of 4N2B couplers senses the status of input contacts and will command the 82S101 to operate the valve when the proper logical combination of inputs occurs. The PS101 will energize the solenoid valve when the output of the FPLA goes LOW.

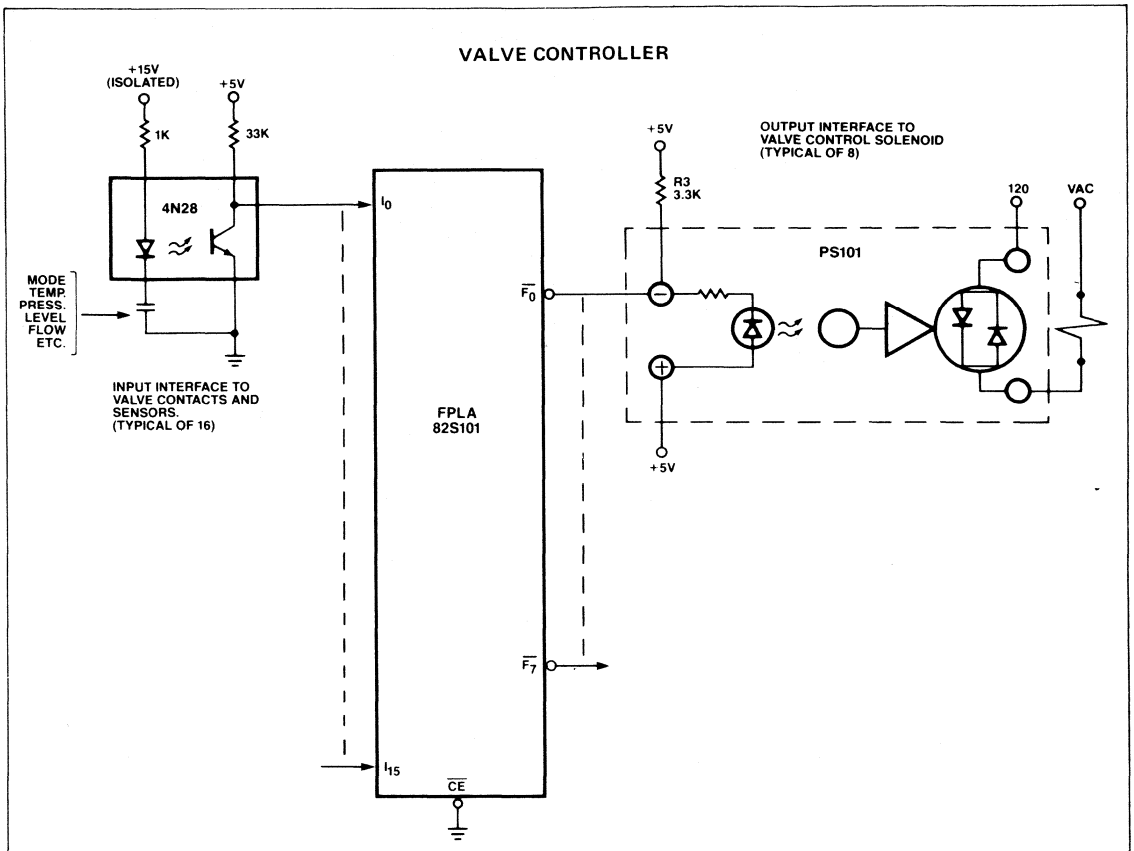


Figure 1.

This controller breaks the time of traffic flow into the four basic parts as shown in Figure 1. Notice that each crossroad is assigned a traffic light during each time frame, and that each time frame has a certain length in seconds. As shown

TIMING OF TRAFFIC LIGHTS

TRAFFIC #1	GREEN	YELLOW	RED	RED
TRAFFIC #2	RED	RED	GREEN	YELLOW
TIME	45 SEC	15 SEC	45 SEC	15 SEC
STATE	00	01	10	11

Figure 1.

in Figure 2, in order for the circuit to recognize the separate states, each state is assigned a state variable to be stored by FFI. The other flip-flops (#2 thru #4) will store the output information for the traffic lights.

Because of the limited number of outputs, the information is time multiplexed out of the FPLA over a 2 bit bus.

The two 7493's, and the 555 timer form a binary counter for counting up to 64 second in 0.5 second intervals.

The FPLA determines the state of the machine by I_{16} and I_{15} and the time elapsed by I_7 thru I_0. When the proper input conditions are met, the 7493 counter is reset, the state in FFI is updated, and the rest of the flip-flops are updated as needed. The time counter continues until a new change in state is indicated by the state of the inputs. At this time the process is repeated.

By programming the FPLA the designer can determine the length of the time segments, and the lights to be activated. Since several inputs are still available there is room to add vehicles and/or people sensors which would instruct the FPLA to change state and continue from there.

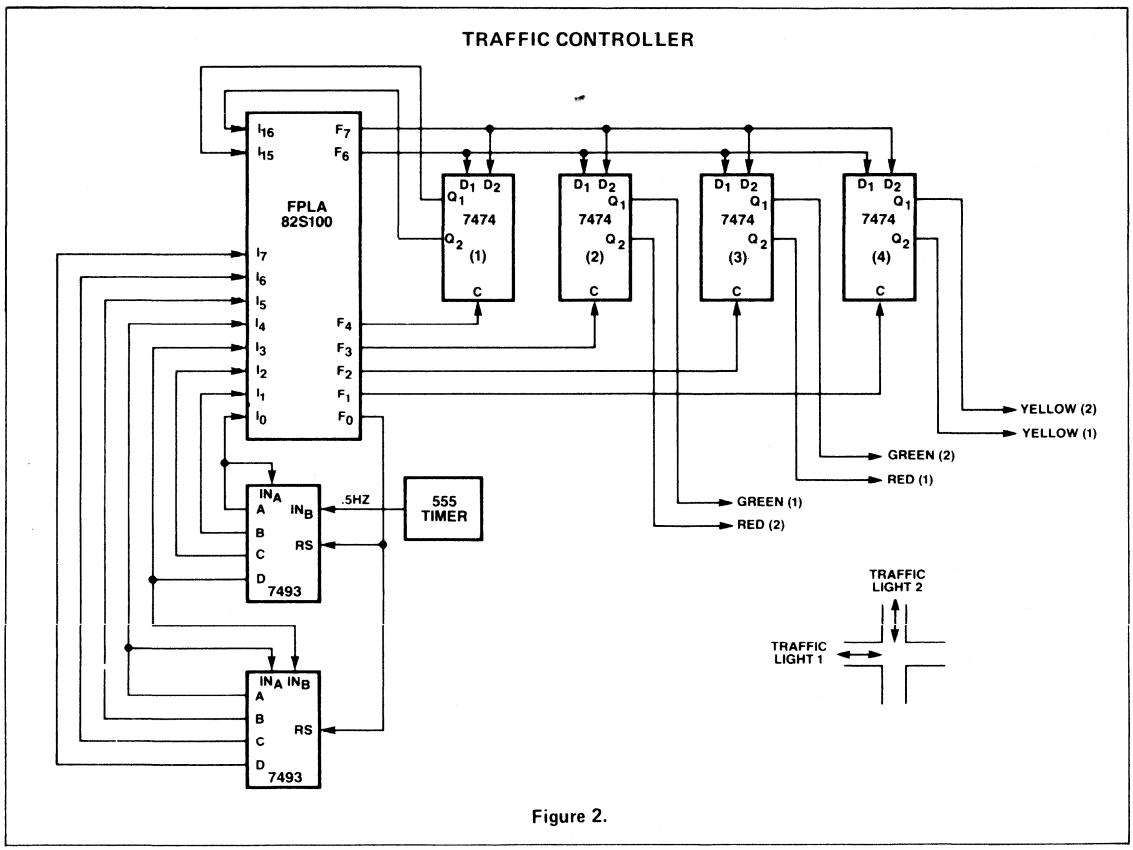


Figure 2.

In the circuit of Figure 1 an 82S101 FPLA is used together with a few other components to build a complete sequential control system on a single 5" x 7" PC card. In this example the circuit is used as a machine controller which recognizes specific starting conditions on its TTL and 120 VAC inputs, and drives the 120 VAC air solenoids in a predetermined sequence, interrupted by time delays as needed. Position feedback from limit switches is used to ensure that each step in the sequence is properly completed, and that the FPLA can detect jams and abort the sequence if any step requires too much time for completion. Logic level outputs are available for such functions as resetting external counters or starting A/D converters.

The system has six inputs and five outputs for interfacing with the machine. In this case, three inputs and two outputs are TTL-compatible, while the remainder are used with 120 VAC. The FPLA analyzes the inputs and internal states of the system, and uses this information to control the desired status of each output, the operation of the time delay circuit, and the advancement of the step counter. Resistor LED indicators can be used for troubleshooting or status checking, with some being driven directly by the FPLA. All 120 VAC interfaces are optically isolated. The 120 VAC inputs come into the system through AC input modules, which provide TTL compatible signals and also drive LED status indicators. These inputs are coupled through debouncing gates before being taken to the FPLA. One of the debouncer sections, with a capacitor tied to its input, provides a startup delay signal, and the debouncer clock is used as the system clock.

An integrated counter/decoder/latch/display provides the sequential step number. The FPLA can either increment or reset the count, and the step number is made available both as a 7 segment display and as BCD data fed back to the FPLA. Time delays are generated by combining a programmable timer/counter with the FPLA. Some of the timer outputs are taken to the FPLA which combines them as required to build time delays of different lengths (50 ms to 12 sec, in this case).

The logic level outputs of the system are taken directly from the FPLA. Outputs to 120 VAC devices are provided by DIP solid state relays driven by the FPLA. LED indicators may be driven by the same FPLA outputs, to show which AC outputs are on. Fuses and varistors for protecting the AC outputs are also mounted on the card.

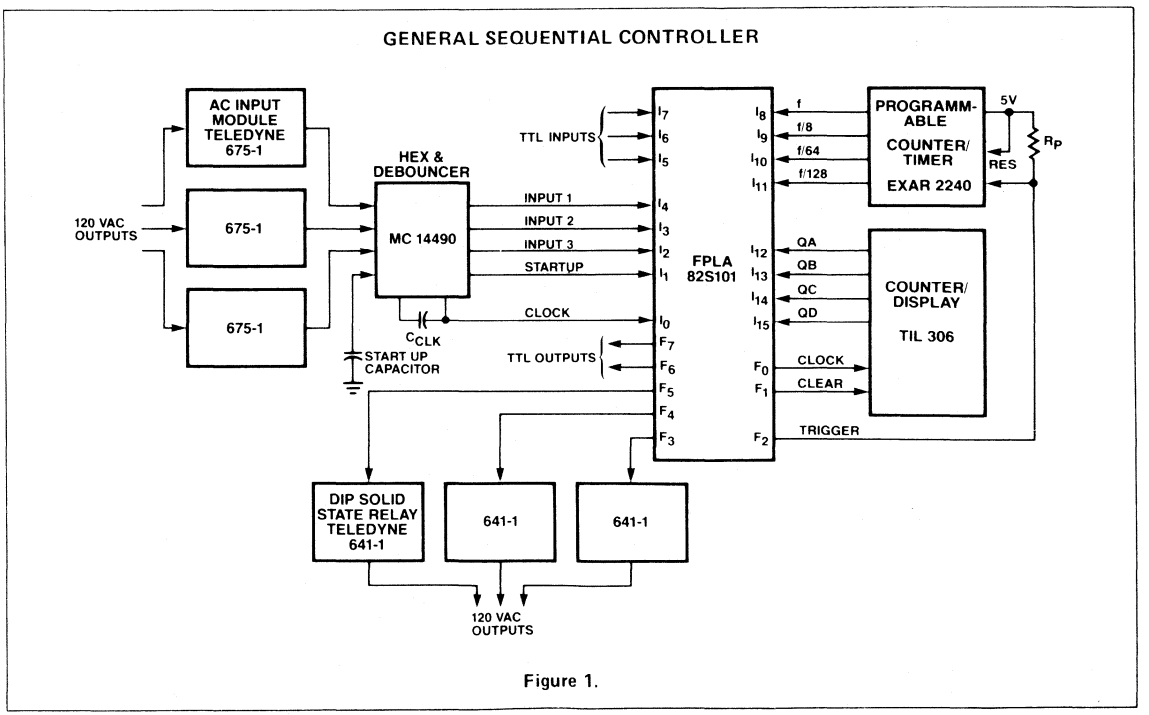


Figure 1.

It is common practice to provide redundant signal paths in todays high reliability military communication systems in order to ensure adequate mission availability. In these systems, it is necessary to monitor the performance of the various subsystems' display status in a concise manner, and provide controls for switching over to standby equipment in either manual or automatic modes. Customarily, a unique System Status and Control Logic Unit (SSCLU) is designed for a system which integrates these control and status functions at a central console. Random T^2L Logic, mounted on a rack full of wire wrap boards, is typically used to implement the SSCLU functions. This approach is inherently costly, inflexible, and unreliable.

The introduction of the Field Programmable Logic Array (FPLA) provides a powerful tool which makes possible the general purpose "n" Expandable SSCLU circuit card set. These two unique circuit card types are all that is normally required to realize an SSCLU. The first circuit card type, shown in Figure 1, is referred to as Subsystem Circuit Card. The second, shown in Figure 2, is designated the Summary Circuit Card. One Subsystem Circuit Card is normally required to monitor and configure each of the subsystems making up a total communication system. Thus, a Satellite Communication Terminal can be divided into the

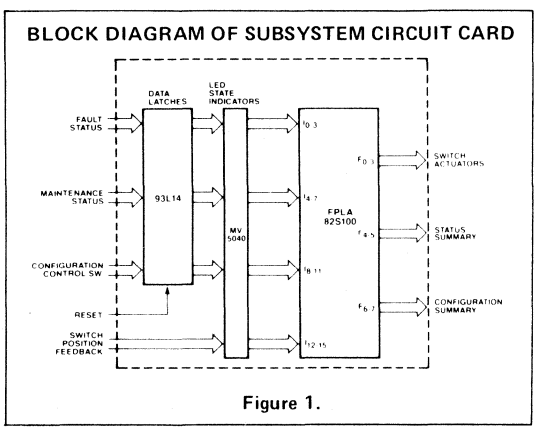


Figure 1.

following dual redundant subsystems: X-Band Downlink, X-Band Uplink, Tracking, and Up/Down Conversion. Each of these subsystems can normally be monitored and controlled by one Subsystem Circuit Card. Two or more simple subsystems could be accommodated by a single Subsystem Circuit Card, or conversely, two cards could possibly be required to service one complex subsystem. The function of the Subsystem Circuit Card is three-fold. It displays the incoming subsystem status and configuration data, reduces the data to develop configuration and status summary messages, and generates control signals to configure the subsystems. The operation of the Subsystem Circuit Card is straightforward. As shown in Figure 1, digital status, configuration and control data, which is normally in the form of contact closures from various subsystem fault sensors, enters the circuit card where it is latched with the exception of the configuration switch feedback signals. The switch feedback is routed directly to the FPLA inputs. The latches are all cleared simultaneously through the reset bus. All inputs to the FPLA have LED state indicators which constantly display the data being fed into it. This feature allows immediate verification of status and facilitates rapid fault isolation. The FPLA scans its input data and generates 3 output data groups based on its micro-program. Its first four outputs activate solid state relays which in turn result in configuration switching of the subsystems equipment. The next two FPLA outputs contain a 2-bit binary code which indicates the subsystem status summary as tabulated in Figure 3. The final two outputs carry a second 2-bit binary code representing the configuration of the subsystem also tabulated in Figure 3.

The SSCLU Summary Circuit Card is illustrated in Figure 2. Its primary purpose is to consolidate and interpret the 2-bit binary status and configuration data from up to 4 Subsystem Circuit Cards. It then generates the outputs based on its micro-program to drive visual and aural system level status and configuration indicators. The inputs to both of its FPLAs are paralleled, and all 16 of them have LED input state indicators. Each FPLA output feeds an open collector lamp driver. Based on the monitoring requirements of the particular system, the outputs of both FPLAs could also be paralleled, resulting in only eight status outputs, but allowing up to 96 product terms to be micro-programmed per Summary Circuit Card.

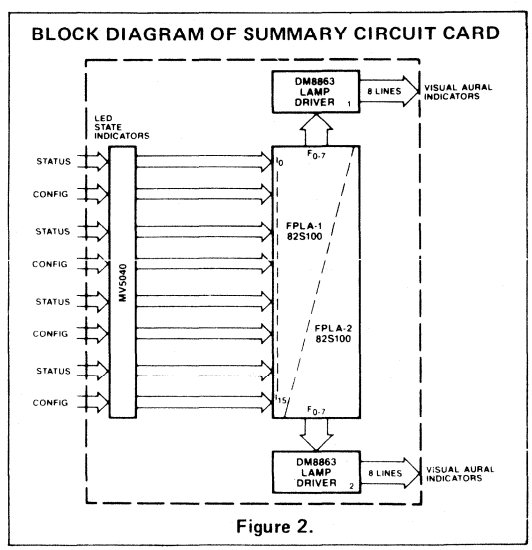


Figure 2.

STATUS AND CONFIGURATION SUMMARY CODE EXAMPLES

	CODE	EQUIPMENT ON LINE	COMMENTS
CONFIGURATION CODE	00	PARAMP I - IFLA I	NORMAL DOWN LINK I
	11	PARAMP II - IFLA II	NORMAL DOWN LINK II
	10	PARAMP I - IFLA II	DOWNLINK I II CROSS PATCH
	01	PARAMP II - IFLA I	DOWNLINK I II CROSS PATCH
	CODE	MEANING	
SUMMARY CODE	00	DN LINK FULLY AVAILABLE	
	10	DN LINK MINOR FAULT	
	01	DN LINK MAJOR FAULT	
	11	DN LINK SWITCH CONFIRMATION FAULT	

Figure 3.

IGNITION TIMING FOR I/C ENGINE USING MULTI-DIMENSIONAL CURVE FIT 8

The diagram of Figure 1 shows a system for optimizing ignition timing of an internal combustion engine in terms of engine speed, temperature, and manifold vacuum. The greatest contribution to the correct spark advance comes from the speed of the engine. For this measurement two clocks are needed: a constant time base from a crystal oscillator and a full cycle from the output of a finely slotted disc in the distributor.

Next is the vacuum at the intake manifold. This is derived from a pressure transducer and an analog to digital converter. Only six bits of resolution should be necessary. It would help keep the cost down and the conversion time below the 20 microsecond limit that would be encountered at high engine speeds and with a 360° slotted angular clock source.

A third transducer measuring coolant temperature uses comparators to look at only one warm-up setting and one threshold at which the engine has overheated to the extent it should be shut down.

Finally a strobe, PISTON CLK, is provided for system syn-

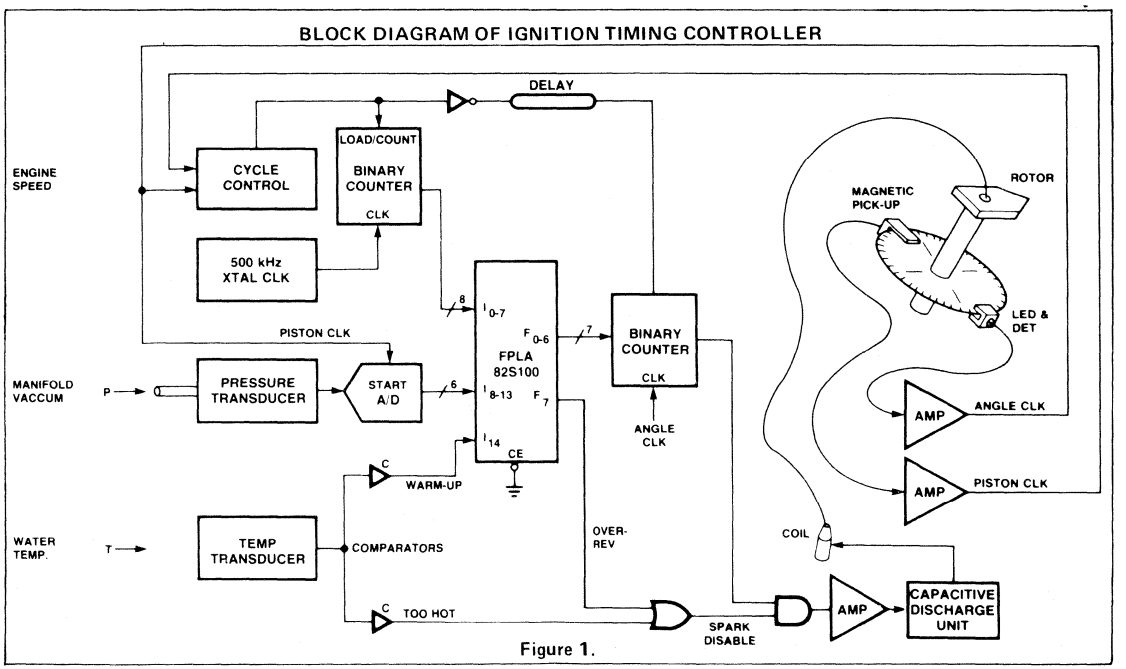


Figure 1.

chronization by adding a magnetic pick-up to the angular clock disc at a point just ahead of the maximum advance angle. This type of sensor would allow us to program for contingency operation in the event dirt fouled the LED source or detector.

FPLA computed spark advance values are loaded into the output counter via a short delay element as soon as the engine speed has been determined. At the same time, the angular clock input is enabled and the counter is incremented to its maximum value. This will trigger a capacitive discharge unit which in turn ignites the mixture in the combustion chamber. A gate has been added to disable the spark under FPLA determined stress conditions such as excessive engine speed.

Although there are only 48 terms available in the 82S100, this should be more than adequate resolution for an automotive ignition. Effective resolution of input and output values, however, are not limited to one part in 48. Because of functional non-linearities and the involvement of more than one variable input dimension, there may be numerous paths to any of the 48 solutions. An increasing input slope will have a decreasing effect on an output and lower order bits can be programmed to a "Don't Care" state. Furthermore, as one input parameter becomes more dominant in determining output, the significance of the other may be diminished with the same technique.

While working out hypothetical solutions, it became apparent that it is preferable to keep input values in direct proportion to output. This way fewer bits are needed to define a new range where a new set of "Don't Care" positions can be programmed. This rule was violated only in the tachometer circuit in favor of response time. Here, several bit positions were necessary to determine the small time interval that represents an over-reved engine.

USING FPLAs AND SRGs TO STORE HUFFMAN CODES 9

In modern serial data transmission systems, the thruput of a communications channel can be improved by using data compression techniques to reduce data redundancies. One method, developed by Huffman, encodes the set of symbols required to convey various messages into a variable length code set. The frequency of use of each symbol dictates the length of the code that is to be associated with that symbol.

Typically, variable length codes are stored in fixed length read only memory look-up tables with a delimiter bit to allow stripping away the extraneous bits stored with the codes. The original uncompressed symbols are then used to address the look-up table as each symbol occurs in the message. As each code is retrieved from the table, some special equipment must remove extraneous bits and compact the codes into byte size chunks for processing by a device such as a USART.

The circuit described here shows an alternative method of storing, retrieving and compacting variable length Huffman codes. This method uses some of the properties of Huffman codes and the properties of maximal length shift register generators (SRGs) to implement an addressable, variable length, storage device.

A typical but fictitious Huffman code set of 34 codes used to demonstrate the circuit implementation is shown in Figure 1. The codes, as shown, must be transmitted left-most bit first. The reason for this leads to a very important property of Huffman codes: no code may be coded in such a way to appear, digit for digit, as the first part of any other code of greater length. This characteristic allows a receiver

TYPICAL HUFFMAN CODE SET

SYMBOL #	LENGTH	CODE	SYMBOL #	LENGTH	CODE
0	2	00	17	7	0100111
1	3	101	18	7	0111010
2	3	110	19	7	0111011
3	4	0101	20	7	0111111
4	4	0110	21	7	0111110
5	4	1000	22	7	1001000
6	5	10011	23	7	1001001
7	5	11101	24	7	1001010
8	6	010001	25	7	1001011
9	6	010010	26	7	1110000
10	6	011100	27	7	1100001
11	6	011110	28	7	1110000
12	6	111110	29	7	1110001
13	6	111100	30	7	1111111
14	6	111101	31	7	1111110
15	7	0100001	32	8	01000000
16	7	0100110	33	8	01000001

Figure 1.

to uniquely decode an incoming message made up of a continuous stream of serial data by knowing only the code set and the starting point of the message. Two other properties of Huffman codes concern each possible sequence of digits of length one less than the longest code in the set, so that they must 1) be used either as a code, or must have one of its prefixes used as a code and that 2) every binary Huffman code set must have exactly two codes of length L, where L is the length of the longest code in the set.

Linear sequential shift register generators (SRGs) are a special class of non-binary counters whose operation can be described and analyzed by application of linear algebra techniques. These counters are made up of a number of delay stages which, except for the first stage, each receive as their input the output from the previous stage (just as

ordinary shift registers). The input to the first stage is derived from a Modulo-2 addition (EX-or function) of selected outputs of the delay stages. These counters repeatedly cycle through a defined sequence of states that are a function of the feedback used. The SRGs used here have a maximal length sequence of states, and thus generate all 2^k-1 states required for Huffman code sets, where k is the number of delay stages. The "all-zero" state is always missing in these counters since the exclusive-or feedback would cause the counter to remain in this state forever.

Since the longest code in Figure 1 is 8 bits long, an 8 stage maximal length SRG will be used in the generation of the code set, since only the last two codes are this long. The other codes in the set will be made up from combinations of less than 8 digits. Every combination of less than 8 digits (2^n, where n = 1, 2, 3, 4, 5, 6, 7) is available from an 8 stage maximal length SRG.

The maximal length SRG used here is defined by $I(X) = X^8 + X^6 + X^5 + X^4 + 1$, which is the characteristic polynominal of an SRG with the outputs of the 8^{th}, 4^{th}, 3^{rd}, and 2^{nd} stages summed Modulo-2 and fed back to the 0^{th} stage, the input. The connections for any SRG can be found directly from the characteristic polynomial associated with that SRG by subtracting from the order k of the polynomial each power of X present in the polynomial.

The selected SRG will produce all the sequences necessary to make up the code set; so all that is needed is circuitry to allow addressing the SRG to access the codes. FPLAs are ideally suited for this application since they can be used as programmable address decoders. This can be accomplished by using code addresses in combination with the outputs of each delay stage of the SRG as inputs to the FPLA.

In the circuit diagram of Figure 2, the SRG (8,4,3,2,0) is made up of U3, U4, U8-A, U8-B, and U8-C and is normally sequencing through its 255 states. An input code address latch is provided by U1, which could be an output port of a

VARIABLE LENGTH STORAGE SYSTEM FOR A TYPICAL HUFFMAN CODE SET

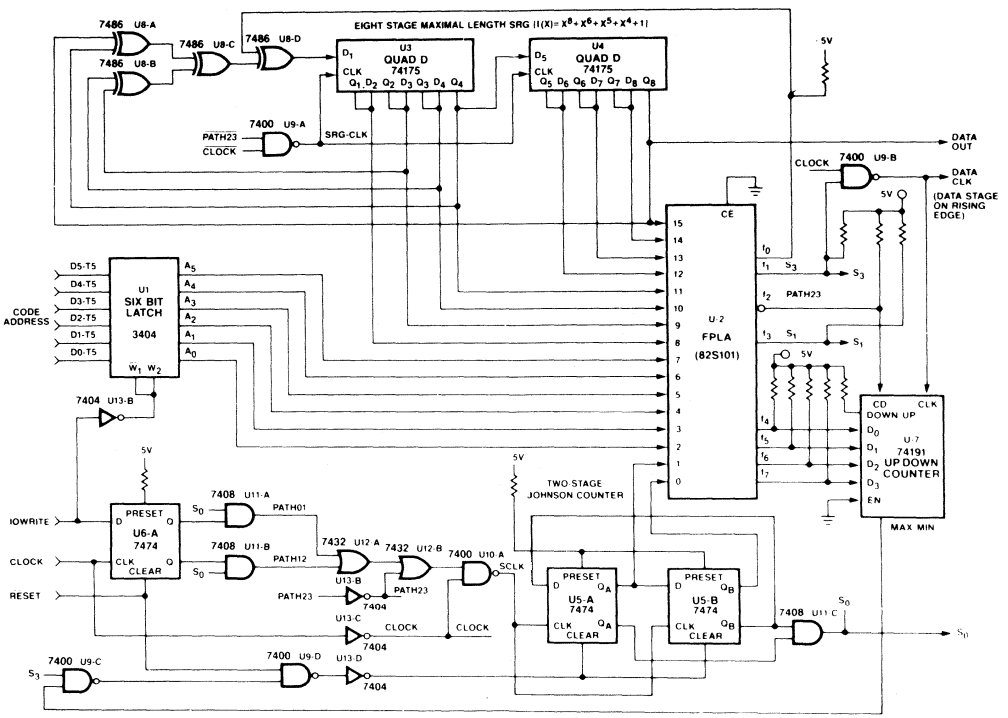


Figure 2.

processor controlled system where the circuitry described here serves as a peripheral. An indication of the latch availability is provided with the output S_0. The FPLA is used for address decoding and some state control functions. The D flip-flop, U6-A, serves to re-clock the "IOWRITE" signal for use by the state controller, which is implemented with a 2-stage Johnson counter by U-5. The up/down counter U-7 is used in conjunction with the FPLA to provide the proper number of data out clocks for each code addressed. With reference to Figure 3, initially the system is in state S_0, and is waiting to be addressed. The S_0 output could be used in either an interrupt driven or a polled system to determine when the circuit is ready for the next code address. When the 6 bit input address latch is serviced the "IOWRITE" signal goes HIGH and is latched by U6-A. This causes the state controller to advance to S_1 on the next system clock. State S_1 is a do nothing state that is just waiting for "IOWRITE" to return back to a logic "0" condition, indicating that the address in the input latch is not stable. When "IOWRITE" goes to logic "0", $\overline{Q}$ of U6-A will go HIGH on the next system clock, enabling the state counter to advance to S_2 on the next clock to find the code that has been addressed.

Since the SRG is free running through its 255 states, the Q_8 through Q_1 inputs to the FPLA are continuously changing. The FPLA is programmed in such a way that it looks for a match between the desired code (the input latch address) and the condition of the last L bits of the SRG, where L is the length of the desired code. For example, as tabulated in Figure 4, the address of code #0 is $\bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4\bar{A}_5$ and code #0 is 00, so the FPLA contains the Product Term ($\bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4\bar{A}_5\bar{Q}_8\bar{Q}_7\bar{Q}_A\bar{Q}_B$). The Q_A and Q_B factors serve to ensure that this condition is met only in S_2. When a match is found, the FPLA outputs $\overline{Path23}$ and a code length word of f_4, f_5, f_6, and f_7. The signal $\overline{Path23}$ does three things: 1) loads the code length word into the up/down counter, 2) disables the clock to the SRG so the desired code is held in the SRG, 3) enables the state counter to advance to S_3. With the system in S_3, the data clock is enabled and the SRG clock is re-enabled. The addressed code can now be collected by a device such as a high speed synchronous receiver tied to the data and data clock lines as the data is shifted out of the SRG in S_3. Notice that the data clock is also connected to the clock input of the up/down counter. This will allow shifting out the L bits corresponding to the length of the code addressed. When all bits of the code have been shifted out the max/min signal from the up/down counter will go HIGH and reset the state counter back to S_0 to wait for the next code address.

The access time for the codes is directly proportional to the code length since the combination of bits to make the shorter codes occur more often in the 255 bit long SRG sequence.

Longer codes could be accommodated with the use of longer SRGs. Longer code sets could also be implemented using this method by using more FPLAs.

STATE CONTROLLER TIMING DIAGRAM

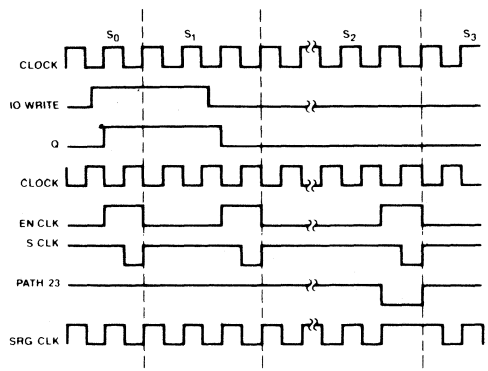


Figure 3.

FPLA PROGRAM TABLE FOR HUFFMAN CODE SET AND STATE CONTROL FUNCTIONS

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	L	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	L	L	–	–	–	–	–	–	L	L	L	L	L	L	H	H	•	•	A	•	•	A	•	•
1	H	L	H	–	–	–	–	–	L	L	L	L	L	H	H	H	•	•	A	A	•	A	•	•
2	H	H	L	–	–	–	–	–	L	L	L	L	H	L	H	H	•	•	A	A	•	A	•	•
3	L	H	L	H	–	–	–	–	L	L	L	L	H	H	H	H	•	A	•	•	•	A	•	•
4	L	H	H	L	–	–	–	–	L	L	L	H	L	L	H	H	•	A	•	•	•	A	•	•
5	H	L	L	L	–	–	–	–	L	L	L	H	L	H	H	H	•	A	•	•	•	A	•	•
6	H	L	L	H	H	–	–	–	L	L	L	H	H	L	H	H	•	A	•	A	•	A	•	•
7	H	H	H	L	H	–	–	–	L	L	L	H	H	H	H	H	•	A	•	A	•	A	•	•
8	L	H	L	L	L	H	–	–	L	L	H	L	L	L	H	H	•	A	A	•	•	A	•	•
9	L	H	L	L	H	L	–	–	L	L	H	L	L	H	H	H	•	A	A	•	•	A	•	•
10	L	H	H	H	L	L	–	–	L	L	H	L	H	L	H	H	•	A	A	•	•	A	•	•
11	L	H	H	H	H	L	–	–	L	L	H	L	H	H	H	H	•	A	A	•	•	A	•	•
12	H	H	H	H	H	L	–	–	L	L	H	H	L	L	H	H	•	A	A	•	•	A	•	•
13	H	H	H	H	L	L	–	–	L	L	H	H	L	H	H	H	•	A	A	•	•	A	•	•
14	H	H	H	H	L	H	–	–	L	L	H	H	H	L	H	H	•	A	A	•	•	A	•	•
15	L	H	L	L	L	L	H	–	L	L	H	H	H	H	H	H	•	A	A	A	•	A	•	•
16	L	H	L	L	H	H	L	–	L	H	L	L	L	L	H	H	•	A	A	A	•	A	•	•
17	L	H	L	L	H	H	H	–	L	H	L	L	L	H	H	H	•	A	A	A	•	A	•	•
18	L	H	H	H	L	H	L	–	L	H	L	L	H	L	H	H	•	A	A	A	•	A	•	•
19	L	H	H	H	L	H	H	–	L	H	L	L	H	H	H	H	•	A	A	A	•	A	•	•
20	L	H	H	H	H	H	H	–	L	H	L	H	L	L	H	H	•	A	A	A	•	A	•	•
21	L	H	H	H	H	H	L	–	L	H	L	H	L	H	H	H	•	A	A	A	•	A	•	•
22	H	L	L	H	L	L	L	–	L	H	L	H	H	L	H	H	•	A	A	A	•	A	•	•
23	H	L	L	H	L	L	H	–	L	H	L	H	H	H	H	H	•	A	A	A	•	A	•	•
24	H	L	L	H	L	H	L	–	L	H	H	L	L	L	H	H	•	A	A	A	•	A	•	•
25	H	L	L	H	L	H	H	–	L	H	H	L	L	H	H	H	•	A	A	A	•	A	•	•
26	H	H	H	L	L	L	L	–	L	H	H	L	H	L	H	H	•	A	A	A	•	A	•	•
27	H	H	H	L	L	L	H	–	L	H	H	L	H	H	H	H	•	A	A	A	•	A	•	•
28	H	H	H	L	L	L	L	–	L	H	H	H	L	L	H	H	•	A	A	A	•	A	•	•
29	H	H	H	L	L	L	H	–	L	H	H	H	L	H	H	H	•	A	A	A	•	A	•	•
30	H	H	H	H	H	H	H	–	L	H	H	H	H	L	H	H	•	A	A	A	•	A	•	•
31	H	H	H	H	H	H	L	–	L	H	H	H	H	H	H	H	•	A	A	A	•	A	•	•
32	L	H	L	L	L	L	L	L	H	L	L	L	L	L	H	H	A	•	•	•	•	A	•	•
33	L	H	L	L	L	L	L	H	H	L	L	L	L	H	H	H	A	•	•	•	•	A	•	•
34	L	L	L	L	L	L	L	L	L	–	–	–	–	–	–	–	•	•	•	•	•	•	•	A
35	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	L	•	•	•	•	A	•	•	•
36	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	H	•	•	•	•	•	•	A	•
INPUT ASSIGNMENT	Q_8 ——————— Q_1								A_5 ——————— A_0						Q_A	Q_B								

Figure 4.

Judicious partitioning of a set of input variables allows an FPLA to easily generate a 4-bit output N, where N equals the number of one's in the 8-bit input I_N. The FPLA Program Table implementing this function is shown in Figure 1, in which positive logic is assumed. The correct output appears after two passes have occurred in the array. A first pass is used to calculate two interim values R_L and R_H. They are partial expressions of the number of one's over two 4-bit sections of input I_N called I_L and I_H respectively. R_L and R_H are fed back to inputs 12 through 15 in the input section, and together with I_N generate the final result N in a second array pass through product terms 20-33.

FINAL FPLA PROGRAM TABLE FOR N-OUT OF -8 DECODE FUNCTION. R_H, R_L ARE FOLDBACK CONNECTIONS

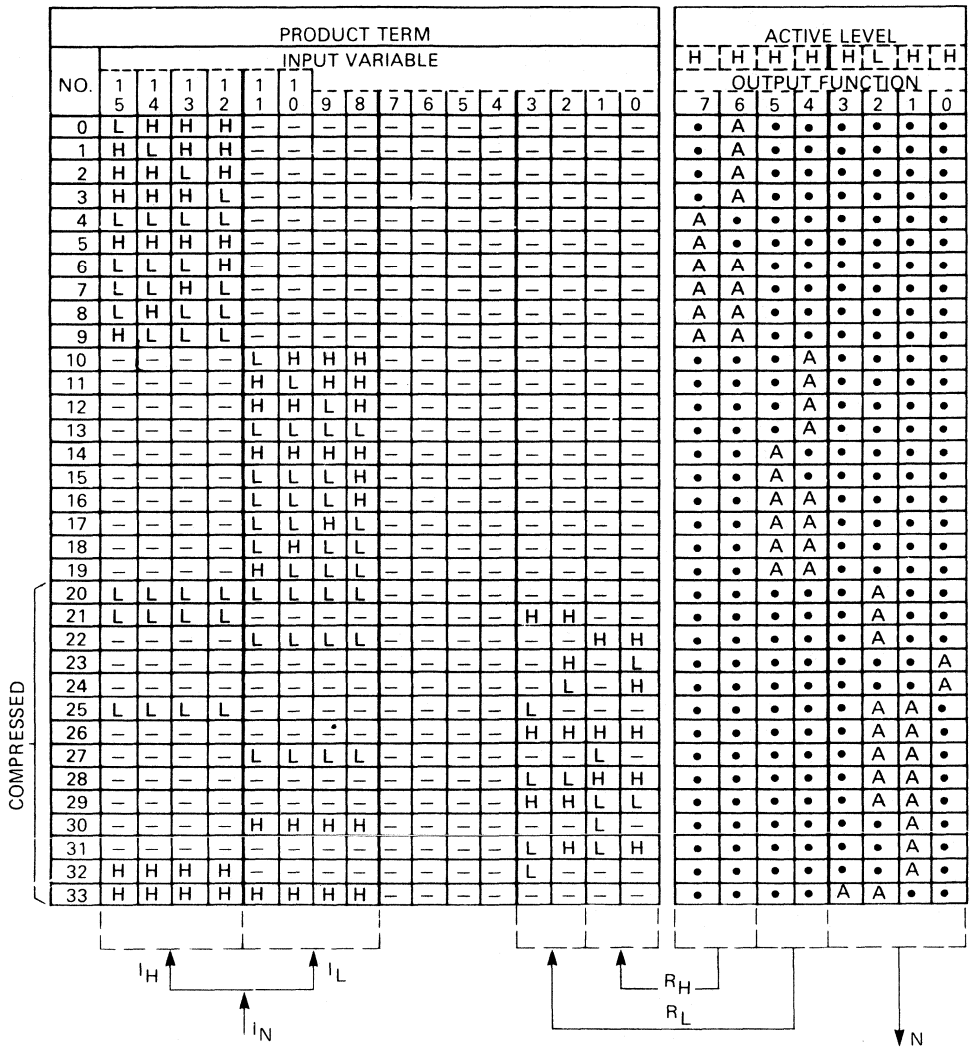


Figure 1.

Minimization of product terms has been achieved by careful selection of the definition of R_L and R_H.

A 2-bit number can denote four different cases, yet the number of one's in I_L (or I_H) can range from zero to four (5 cases) with the following frequencies of occurence: 1, 4, 6, 4, 1. It thus makes sense to define R_L (and R_H) in such a way that:

a) the default code 00 is used to express the case of the highest occurrence.

b) the two cases of lowest occurrence are chosen to be the cases expressed by a common code.

As a result, the following definition can be adopted for both design cases:

$$R_L = \begin{cases} 00 & \text{when there are 2 one's in } I_L \\ 01 & \text{when there are 3 one's in } I_L \\ 11 & \text{when there is 1 one in } I_L \\ 10 & \text{for either no one's or all one's in } I_L \end{cases}$$

The same definition is adopted for R_H in relation to I_H. Notice that the righthand bit of R is a "1" for an odd number of one's and a "0" for an even number. Based on this definition a first design pass resulted in 44 product terms, with the first 20 shown in Figure 1. The other 24 product terms are shown in Figure 2. Addition of a product term for the case of all zero's in I_N, and simultaneous inversion of active level for output 5 transforms the table in Figure 2 to that in Figure 3, indicating only 20 product terms are required for array pass 2. Visual inspection of

PART OF FPLA TABLE FROM INITIAL DESIGN PASS.

NOTE F_2 ASSIGNED ACTIVE-HIGH

	PRODUCT TERM															
	INPUT VARIABLE															
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																
1																
2																
20	L	L	L	L	–	–	–	–	–	–	–	–	H	H	–	–
21	–	–	–	–	L	L	L	L	–	–	–	–	–	–	H	H
22	L	L	L	L	–	–	–	–	–	–	–	–	L	L	–	–
23	–	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H
24	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	L
25	L	L	L	L	–	–	–	–	–	–	–	–	L	H	–	–
26	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H
27	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	L
28	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	H
29	L	L	L	L	H	H	H	H	–	–	–	–	–	–	–	–
30	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H
31	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L
32	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	H
33	H	H	H	H	L	L	L	L	–	–	–	–	–	–	–	–
34	–	–	–	–	H	H	H	H	–	–	–	–	–	–	H	H
35	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	L
36	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H
37	H	H	H	H	–	–	–	–	–	–	–	–	H	H	–	–
38	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	L
39	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H
40	H	H	H	H	–	–	–	–	–	–	–	–	L	L	–	–
41	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	H
42	H	H	H	H	–	–	–	–	–	–	–	–	L	H	–	–
43	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–	–

	ACTIVE LEVEL							
	H	H	H	H	H	H	H	H
	OUTPUT FUNCTION							
	7	6	5	4	3	2	1	0
0								
1								
2								
20	•	•	•	•	•	•	•	A
21	•	•	•	•	•	•	•	A
22	•	•	•	•	•	•	A	•
23	•	•	•	•	•	•	A	•
24	•	•	•	•	•	•	A	•
25	•	•	•	•	•	•	A	A
26	•	•	•	•	•	•	A	A
27	•	•	•	•	•	•	A	A
28	•	•	•	•	•	•	A	A
29	•	•	•	•	•	A	•	•
30	•	•	•	•	•	A	•	•
31	•	•	•	•	•	A	•	•
32	•	•	•	•	•	A	•	•
33	•	•	•	•	•	A	•	•
34	•	•	•	•	•	A	•	A
35	•	•	•	•	•	A	•	A
36	•	•	•	•	•	A	•	A
37	•	•	•	•	•	A	•	A
38	•	•	•	•	•	A	A	•
39	•	•	•	•	•	A	A	•
40	•	•	•	•	•	A	A	•
41	•	•	•	•	•	A	A	A
42	•	•	•	•	•	A	A	A
43	•	•	•	•	A	•	•	•

Figure 2.

Figure 3 for commonalities resulting in "don't care" conditions allows a further reduction to only 14 product terms as shown in the bottom part of the table in Figure 1. A variant of this method yields the function incorporated in the table of Figure 4. Here, a single output labeled "yes" is activated when the number of one's in input I_N equals the binary number of 4-bit control input M. The function could have been obtained using an FPLA programmed as in Figure 1 followed by an MSI compare module (SN7485). However, this would have increased the total circuit delay. As Figure 4 shows, it is still possible to perform the entire function within one FPLA with two array passes.

PART OF FPLA TABLE AFTER SECOND PASS.

NOTE F_2 CHANGED TO ACTIVE – LOW

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	L	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0																								
1																								
2																								
20	L	L	L	L	–	–	–	–	–	–	–	–	H	H	–	–	•	•	•	•	•	A	•	A
21	–	–	–	–	L	L	L	L	–	–	–	–	–	–	H	H	•	•	•	•	•	A	•	A
22	L	L	L	L	–	–	–	–	–	–	–	–	L	L	–	–	•	•	•	•	•	A	A	•
23	–	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H	•	•	•	•	•	A	A	•
24	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	L	•	•	•	•	•	A	A	•
25	L	L	L	L	–	–	–	–	–	–	–	–	L	H	–	–	•	•	•	•	•	A	A	A
26	–	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	•	•	•	•	•	A	A	A
27	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	L	•	•	•	•	•	A	A	A
28	–	–	–	–	L	L	L	L	–	–	–	–	–	–	L	H	•	•	•	•	•	A	A	A
29	L	L	L	L	H	H	H	H	–	–	–	–	–	–	–	–	•	•	•	•	•	•	•	•
30	–	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	•	•	•	•	•	•	•	•
31	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	L	•	•	•	•	•	•	•	•
32	–	–	–	–	–	–	–	–	–	–	–	–	H	H	L	H	•	•	•	•	•	•	•	•
33	H	H	H	H	L	L	L	L	–	–	–	–	–	–	–	–	•	•	•	•	•	•	•	•
34	–	–	–	–	H	H	H	H	–	–	–	–	–	–	H	H	•	•	•	•	•	•	•	A
35	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	L	•	•	•	•	•	•	•	A
36	–	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	•	•	•	•	•	•	•	A
37	H	H	H	H	–	–	–	–	–	–	–	–	H	H	–	–	•	•	•	•	•	•	•	A
38	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	L	•	•	•	•	•	•	A	•
39	–	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	•	•	•	•	•	•	A	•
40	H	H	H	H	–	–	–	–	–	–	–	–	L	L	–	–	•	•	•	•	•	•	A	•
41	–	–	–	–	H	H	H	H	–	–	–	–	–	–	L	H	•	•	•	•	•	•	A	A
42	H	H	H	H	–	–	–	–	–	–	–	–	L	H	–	–	•	•	•	•	•	•	A	A
43	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–	–	•	•	•	•	A	A	•	•
44	L	L	L	L	L	L	L	L	–	–	–	–	–	–	–	–	•	•	•	•	•	A	•	•

Figure 3.

FPLA PROGRAM TABLE FOR FUNCTION "YES" = "1" WHEN NUMBER OF 1's IN I_N AND M ARE THE SAME

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	L	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
1	H	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
2	H	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
3	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	•
4	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	•
5	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	•
6	L	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
7	L	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
8	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
9	H	L	L	L	–	–	–	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•
10	–	–	–	–	L	H	H	H	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
11	–	–	–	–	H	L	H	H	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
12	–	–	–	–	H	H	L	H	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
13	–	–	–	–	H	H	H	L	–	–	–	–	–	–	–	–	•	•	•	A	A	A	A	•
14	–	–	–	–	L	L	L	L	–	–	–	–	–	–	–	–	•	•	A	•	A	A	A	•
15	–	–	–	–	H	H	H	H	–	–	–	–	–	–	–	–	•	•	A	•	A	A	A	•
16	–	–	–	–	L	L	L	H	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
17	–	–	–	–	H	L	H	L	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
18	–	–	–	–	L	H	L	L	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
19	–	–	–	–	H	L	L	L	–	–	–	–	–	–	–	–	•	•	A	A	A	A	A	•
20	L	L	L	L	L	L	L	L	L	L	L	L	–	–	–	–	•	•	•	•	A	A	A	A
21	L	L	L	L	–	–	–	–	L	L	L	H	H	H	–	–	•	•	•	•	A	A	A	A
22	–	–	–	–	L	L	L	L	L	L	L	H	–	–	–	–	•	•	•	•	A	A	A	A
23	L	L	L	L	–	–	–	–	L	L	H	L	L	L	–	–	•	•	•	•	A	A	A	A
24	–	–	–	–	–	–	–	–	L	L	H	L	H	H	H	H	•	•	•	•	A	A	A	A
25	–	–	–	–	L	L	L	L	L	L	H	L	–	–	L	L	•	•	•	•	A	A	A	A
26	L	L	L	L	–	–	–	–	L	L	H	H	L	H	–	–	•	•	•	•	A	A	A	A
27	–	–	–	–	–	–	–	–	L	L	H	H	L	L	H	H	•	•	•	•	A	A	A	A
28	–	–	–	–	–	–	–	–	L	L	H	H	H	H	L	L	•	•	•	•	A	A	A	A
29	–	–	–	–	L	L	L	L	L	L	H	H	–	–	L	H	•	•	•	•	A	A	A	A
30	L	L	L	L	H	H	H	H	L	H	L	L	–	–	–	–	•	•	•	•	A	A	A	A
31	–	–	–	–	–	–	–	–	L	H	L	L	L	H	H	H	•	•	•	•	A	A	A	A
32	–	–	–	–	–	–	–	–	L	H	L	L	L	L	L	L	•	•	•	•	A	A	A	A
33	–	–	–	–	–	–	–	–	L	H	L	L	H	H	L	H	•	•	•	•	A	A	A	A
34	H	H	H	H	L	L	L	L	L	H	L	L	–	–	–	–	•	•	•	•	A	A	A	A
35	–	–	–	–	H	H	H	H	L	H	L	H	–	–	H	H	•	•	•	•	A	A	A	A
36	–	–	–	–	–	–	–	–	L	H	L	H	L	H	L	L	•	•	•	•	A	A	A	A
37	–	–	–	–	–	–	–	–	L	H	L	H	L	L	L	H	•	•	•	•	A	A	A	A
38	H	H	H	H	–	–	–	–	L	H	L	H	H	H	–	–	•	•	•	•	A	A	A	A
39	–	–	–	–	H	H	H	H	L	H	H	L	–	–	L	L	•	•	•	•	A	A	A	A
40	–	–	–	–	–	–	–	–	L	H	H	L	L	H	L	H	•	•	•	•	A	A	A	A
41	H	H	H	H	–	–	–	–	L	H	H	L	L	L	–	–	•	•	•	•	A	A	A	A
42	–	–	–	–	H	H	H	H	L	H	H	H	–	–	L	H	•	•	•	•	A	A	A	A
43	H	H	H	H	–	–	–	–	L	H	H	H	L	H	–	–	•	•	•	•	A	A	A	A
44	H	H	H	H	H	H	H	H	H	L	L	L	–	–	–	–	•	•	•	•	A	A	A	A

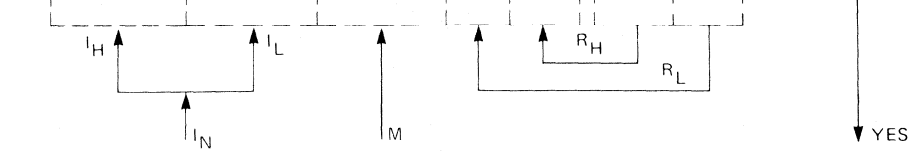


Figure 4.

A shift/logic unit (SLU) with FPLA's can provide high-speed field and bit manipulations for data packing in fast I/O controllers, and minicomputers used in number crunching applications. The block diagram of Figure 1 illustrates the SLU environment, consisting of a multi-port register file, such as Signetics' 82S112, and a control sequencer which could also be designed with additional FPLA's.

Any two registers, not necessarily different, can be read on the A and B ports and supplied as left and right operands to the SLU. The result, Z, can be returned to any register in the register file.

The SLU is composed of 4 FPLA pairs programmed as SLU1 and SLU2, for a total of 8 FPLA's. These are connected as shown in Figure 2 to implement the overall SLU instruction set tabulated in Figure 3.

The SLU and register file are 16 bits wide, and are both controlled by an 8-bit word from the control sequencer. The control word, mapped in Figure 4, commands the SLU to perform three basic operations which are selected by mode bits M0 and M1. These are:

SYSTEM BLOCK DIAGRAM

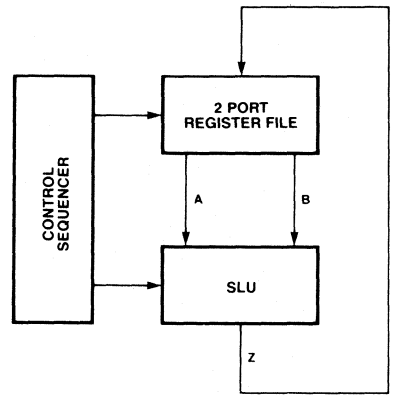


Figure 1.

SLU DIAGRAM SHOWING I/O CONNECTIONS FOR EACH FPLA PAIR

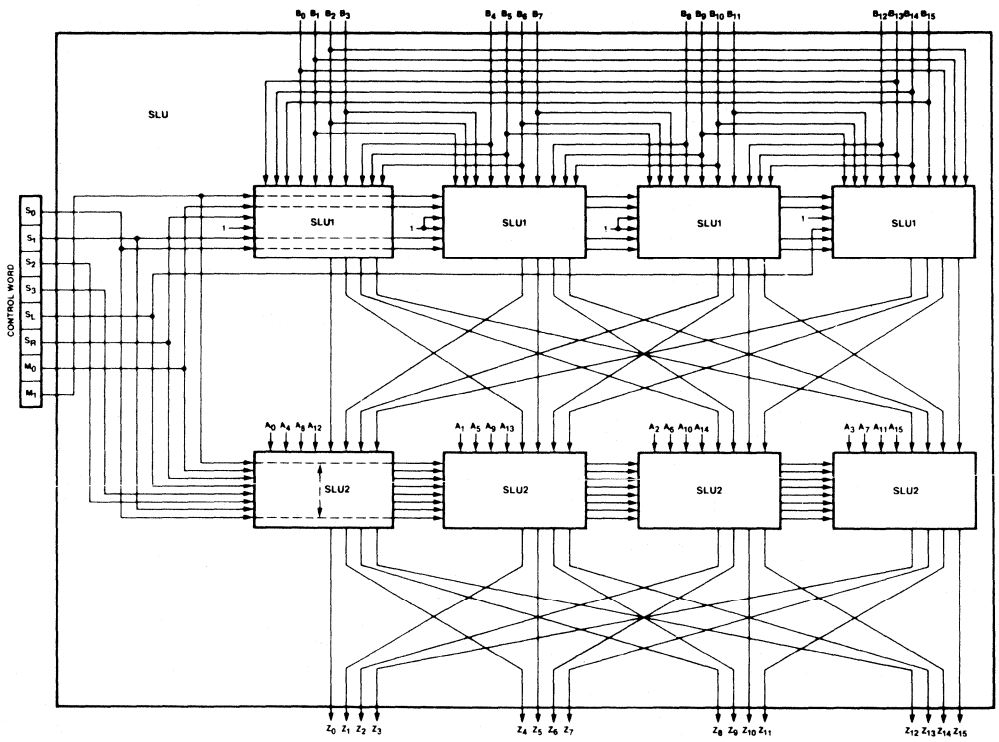


Figure 2.

SLU INSTRUCTION SET

Mnemonic	M_1	M_0	S_R	S_L	S_3	S_2	S_1	S_0	Boolean Operation	Comment
ZERO	0	1	X	X	0	0	0	0	0	Logical zero
NOR	0	1	X	X	0	0	0	1	$\bar{A}\bar{B}$	NOR
NOR NOT	0	1	X	X	0	0	1	0	$\bar{A}B$	NOR NOT
NLID	0	1	X	X	0	0	1	1	$\bar{A}$	Left identity complement
AND NOT	0	1	X	X	0	1	0	0	$A\bar{B}$	AND NOT
NRID	0	1	X	X	0	1	0	1	$\bar{B}$	Right identity complement
EXOR	0	1	X	X	0	1	1	0	$A \oplus B$	Exclusive or
NAND	0	1	X	X	0	1	1	1	$\bar{A} + \bar{B}$	NAND
AND	0	1	X	X	1	0	0	0	AB	AND
EQV	0	1	X	X	1	0	0	1	$A \odot B$	Equivalence
RID	0	1	X	X	1	0	1	0	B	right identity
NAND NOT	0	1	X	X	1	0	1	1	$\bar{A} + B$	NAND NOT
LID	0	1	X	X	1	1	0	0	A	left identity
OR NOT	0	1	X	X	1	1	0	1	$A + \bar{B}$	OR NOT
OR	0	1	X	X	1	1	1	0	$A + B$	OR
ONE	0	1	X	X	1	1	1	1	1	Logical one
SOFF RN	1	0	0	1	(	N_2^*		)		Shift B right end-off N places
SCIR RN	1	0	1	1	(	N_2		)		Shift B right circular N places
SOFF LN	1	1	1	0	(	N_2		)		Shift B left end-off N places
SCIR LN	1	1	1	1	(	N_2		)		Shift B left circular N places
TEST	0	0	X	X	(	N_2		)		Logical zero if B = 0 Logical one if B ≠ 0

*N_2 = binary number specified by $S_{0\sim3}$.

Figure 3.

MAP OF SLU CONTROL WORD

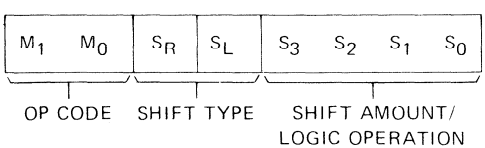


Figure 4.

A. Shift left or right, circular or end-off any number of places.
B. Execute any of 16 boolean operations for 2 variables.
C. Test for zero.

Shift-type bits, S_R and S_L, control execution of circular or end-off shift. These bits are ignored in other than shift operation. Select bits S_{0-3} supply the shift amount (0 thru 15) in shift modes, or determine one of 16 boolean functions in logic mode. The SLU instruction set has been partitioned in two distinct logic equation sets incorporated in two FPLA types designated respectively SLU1 and SLU2. The

I/O assignment for each FPLA type is defined in Figure 5. The program table for each FPLA can be derived by implementing the logic equation sets tabulated Figures 6 and 7 respectively. Both figures express in a short-hand notation (similar to a multiplication table) the logic equations for SLU1 and SLU2, involving their inputs, outputs, and number of Product Terms necessary to implement each function group. Unlisted input combinations are not allowed, and can be treated as Don't Care during minimization. For example, the logic equation for SLU1 output X_0 is obtained by minimizing with a Karnaugh map all logic products tabulated in Figure 6. These are formed by plotting all entries in column X_0 of the FPLA outputs in appropriate squares corresponding to the logic value of the FPLA inputs. This procedure is illustrated in the map of Figure 8, in which all unlisted input combinations have been assigned a Don't Care because of input constraints.

After minimizing all adjacent squares, output X_0 is given by the sum of 12 product terms as follows:

$$X_0 = B_0(\overline{M}_1 M_0) + B_0(M_1\overline{S}_1\overline{S}_0) + B_{-1}(M_1\overline{M}_0 S_R\overline{S}_1 S_0) + B_{-2}(M_1\overline{M}_0 S_R S_1\overline{S}_0) + B_{-3}(M_1\overline{M}_0 S_R S_1 S_0) + B_1(M_1 M_0\overline{S}_1 S_0) + B_2(M_1 M_0 S_1\overline{S}_0) + B_3(M_1 M_0 S_1 S_0) + (B_0+B_1+B_2+B_3)\overline{M}_1\overline{M}_0$$

Outputs $X_{1\text{-}3}$ are treated the same way, and we obtain a total of 36 product terms which are tabulated in the FPLA Program Table for SLU1 in Figure 9. The Program Table for SLU2 is derived by following a similar procedure.

I/O ASSIGNMENT OF FPLA PAIR SLU1 and SLU2

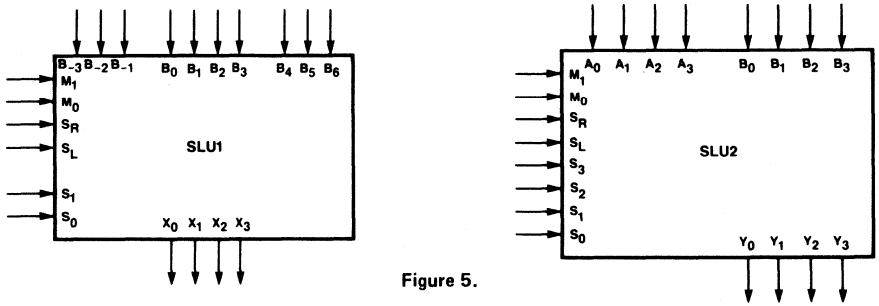


Figure 5.

SLU1 FUNCTION TABLE. Note that B_0–B_6 and B_{-1}→B_{-3} are also FPLA inputs

FPLA INPUTS						FPLA OUTPUTS				P-TERMS REQUIRED
M_1	M_0	S_R	S_L	S_1	S_0	X_0	X_1	X_2	X_3	
0	1	X	X	X	X	B_0	B_1	B_2	B_3	4
1	0	0	1	0	0	B_0	B_1	B_2	B_3	16
1	0	0	1	0	1	0	B_0	B_1	B_2	
1	0	0	1	1	0	0	0	B_0	B_1	
1	0	0	1	1	1	0	0	0	B_0	
1	0	1	1	0	0	B_0	B_1	B_2	B_3	
1	0	1	1	0	1	B_{-1}	B_0	B_1	B_2	
1	0	1	1	1	0	B_{-2}	B_{-1}	B_0	B_1	
1	0	1	1	1	1	B_{-3}	B_{-2}	B_{-1}	B_0	
1	1	1	0	0	0	B_0	B_1	B_2	B_3	12
1	1	1	0	0	1	B_1	B_2	B_3	0	
1	1	1	0	1	0	B_2	B_3	0	0	
1	1	1	0	1	1	B_3	0	0	0	
1	1	1	1	0	0	B_0	B_1	B_2	B_3	
1	1	1	1	0	1	B_1	B_2	B_3	B_4	
1	1	1	1	1	0	B_2	B_3	B_4	B_5	
1	1	1	1	1	1	B_3	B_4	B_5	B_6	
0	0	X	X	X	X	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	4

Figure 6.

SLU2 TRUTH TABLE. Note that A_{0-3} and B_{0-3} are also FPLA inputs

FPLA INPUTS								FPLA OUTPUTS				P-TERMS
M_1	M_0	S_R	S_L	S_3	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	REQUIRED
0	1	X	X	0	0	0	0	0	0	0	0	
0	1	X	X	0	0	0	1	$\bar{A}_0\bar{B}_0$	$\bar{A}_1\bar{B}_1$	$\bar{A}_2\bar{B}_2$	$\bar{A}_3\bar{B}_3$	
0	1	X	X	0	0	1	0	$\bar{A}_0B_0$	$\bar{A}_1B_1$	$\bar{A}_2B_2$	$\bar{A}_3B_3$	
0	1	X	X	0	0	1	1	$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	
0	1	X	X	0	1	0	0	$A_0\bar{B}_0$	$A_1\bar{B}_1$	$A_2\bar{B}_2$	$A_3\bar{B}_3$	
0	1	X	X	0	1	0	1	$\bar{B}_0$	$\bar{B}_1$	$\bar{B}_2$	$\bar{B}_3$	
0	1	X	X	0	1	1	0	$A_0\bar{B}_0+\bar{A}_0B_0$	$A_1\bar{B}_1+\bar{A}_1B_1$	$A_2\bar{B}_2+\bar{A}_2B_2$	$A_3\bar{B}_3+\bar{A}_3B_3$	
0	1	X	X	0	1	1	1	$\bar{A}_0+\bar{B}_0$	$\bar{A}_1+\bar{B}_1$	$\bar{A}_2+\bar{B}_2$	$\bar{A}_3+\bar{B}_3$	16
0	1	X	X	1	0	0	0	A_0B_0	A_1B_1	A_2B_2	A_3B_3	
0	1	X	X	1	0	0	1	$A_0B_0+\bar{A}_0\bar{B}_0$	$AA_1B_1+\bar{A}_1\bar{B}_1$	$A_2B_2+\bar{A}_2\bar{B}_2$	$AA_3B_3+\bar{A}_3\bar{B}_3$	
0	1	X	X	1	0	1	0	B_0	B_1	B_2	B_3	
0	1	X	X	1	0	1	1	$\bar{A}_0+B_0$	$\bar{A}_1+B_1$	$\bar{A}_2+B_2$	$\bar{A}_3+B_3$	
0	1	X	X	1	1	0	0	A_0	A_1	A_2	A_3	
0	1	X	X	1	1	0	1	$A_0+\bar{B}_0$	$A_1+\bar{B}_1$	$A_2+\bar{B}_2$	$A_3+\bar{B}_3$	
0	1	X	X	1	1	1	0	A_0+B_0	A_1+B_1	A_2+B_2	A_3+B_3	
0	1	X	X	1	1	1	1	1	1	1	1	
1	0	0	1	0	0	X	X	B_0	B_1	B_2	B_3	
1	0	0	1	0	1	X	X	0	B_0	B_1	B_2	
1	0	0	1	1	0	X	X	0	0	B_0	B_1	
1	0	0	1	1	1	X	X	0	0	0	B_0	
1	0	1	1	0	0	X	X	B_0	B_1	B_2	B_3	
1	0	1	1	0	1	X	X	B_1	B_0	B_1	B_2	28
1	0	1	1	1	0	X	X	B_2	B_3	B_0	B_1	
1	0	1	1	1	1	X	X	B_1	B_2	B_3	B_0	
1	1	1	0	0	0	X	X	B_0	B_1	B_2	B_3	
1	1	1	0	0	1	X	X	B_1	B_2	B_3	0	
1	1	1	0	1	0	X	X	B_2	B_3	0	0	
1	1	1	0	1	1	X	X	B_3	0	0	0	
1	1	1	1	0	0	X	X	B_0	B_1	B_2	B_3	
1	1	1	1	0	1	X	X	B_1	B_2	B_3	B_0	
1	1	1	1	1	0	X	X	B_2	B_3	B_0	B_1	
1	1	1	1	1	1	X	X	B_3	B_0	B_1	B_2	
0	0	X	X	X	X	X	X	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	$B_0+B_1+B_2+B_3$	4

Figure 7.

KARNAUGH MAP FOR OUTPUT X_0 OF SLU1, WHERE $C = B_0 + B_1 + B_2 + B_3$, and (X) = DON't CARE

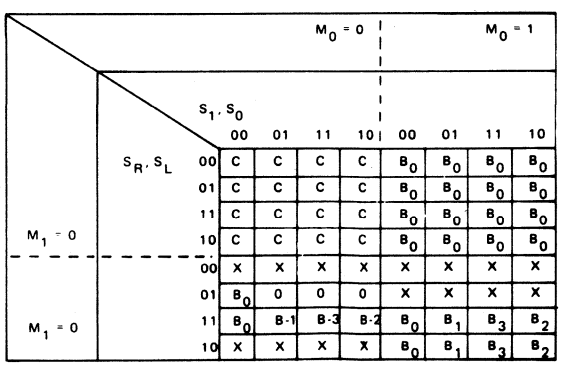


Figure 8.

FPLA PROGRAM TABLE FOR SLU1

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H H H H H H H H — OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	L	H	–	–	–	–	–	–	–	H	–	–	–	–	–	–	A	•	•	•	A	A	A	A
1	L	H	–	–	–	–	–	–	–	–	H	–	–	–	–	–	•	A	•	•	A	A	A	A
2	L	H	–	–	–	–	–	–	–	–	–	H	–	–	–	–	•	•	A	•	A	A	A	A
3	L	H	–	–	–	–	–	–	–	–	–	–	H	–	–	–	•	•	•	A	A	A	A	A
4	H	–	–	–	L	L	–	–	–	H	–	–	–	–	–	–	A	•	•	•	A	A	A	A
5	H	–	–	–	L	L	–	–	–	–	H	–	–	–	–	–	•	A	•	•	A	A	A	A
6	H	–	–	–	L	L	–	–	–	–	–	H	–	–	–	–	•	•	A	•	A	A	A	A
7	H	–	–	–	L	L	–	–	–	–	–	–	H	–	–	–	•	•	•	A	A	A	A	A
8	H	L	H	–	L	H	–	–	H	–	–	–	–	–	–	–	A	•	•	•	A	A	A	A
9	H	L	–	–	L	H	–	–	–	H	–	–	–	–	–	–	•	A	•	•	A	A	A	A
10	H	L	–	–	L	H	–	–	–	–	H	–	–	–	–	–	•	•	A	•	A	A	A	A
11	H	L	–	–	L	H	–	–	–	–	–	H	–	–	–	–	•	•	•	A	A	A	A	A
12	H	L	H	–	H	L	–	H	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	A
13	H	L	H	–	H	L	–	–	H	–	–	–	–	–	–	–	•	A	•	•	A	A	A	A
14	H	L	–	–	H	L	–	–	–	H	–	–	–	–	–	–	•	•	A	•	A	A	A	A
15	H	L	–	–	H	L	–	–	–	–	H	–	–	–	–	–	•	•	•	A	A	A	A	A
16	H	L	H	–	H	H	H	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	A
17	H	L	H	–	H	H	–	H	–	–	–	–	–	–	–	–	•	A	•	•	A	A	A	A
18	H	L	H	–	H	H	–	–	H	–	–	–	–	–	–	–	•	•	A	•	A	A	A	A
19	H	L	–	–	H	H	–	–	–	H	–	–	–	–	–	–	•	•	•	A	A	A	A	A
20	H	H	–	–	L	H	–	–	–	–	H	–	–	–	–	–	A	A	•	•	A	A	A	A
21	H	H	–	–	L	H	–	–	–	–	–	H	–	–	–	–	•	A	•	•	A	A	A	A
22	H	H	–	–	L	H	–	–	–	–	–	–	H	–	–	–	•	•	A	•	A	A	A	A
23	H	H	–	H	L	H	–	–	–	–	–	–	–	H	–	–	•	•	•	A	A	A	A	A
24	H	H	–	–	H	L	–	–	–	–	–	H	–	–	–	–	A	•	•	•	A	A	A	A
25	H	H	–	–	H	L	–	–	–	–	–	–	H	–	–	–	•	A	•	•	A	A	A	A
26	H	H	–	H	H	L	–	–	–	–	–	–	–	H	–	–	•	•	A	•	A	A	A	A
27	H	H	–	H	H	L	–	–	–	–	–	–	–	–	H	–	•	•	•	A	A	A	A	A
28	H	H	–	–	H	H	–	–	–	–	–	–	H	–	–	–	A	•	•	•	A	A	A	A
29	H	H	–	H	H	H	–	–	–	–	–	–	–	H	–	–	•	A	•	•	A	A	A	A
30	H	H	–	H	H	H	–	–	–	–	–	–	–	–	H	–	•	•	A	•	A	A	A	A
31	H	H	–	H	H	H	–	–	–	–	–	–	–	–	–	H	•	•	•	A	A	A	A	A
32	L	L	–	–	–	–	–	–	–	H	–	–	–	–	–	–	A	A	A	A	A	A	A	A
33	L	L	–	–	–	–	–	–	–	–	H	–	–	–	–	–	A	A	A	A	A	A	A	A
34	L	L	–	–	–	–	–	–	–	–	–	H	–	–	–	–	A	A	A	A	A	A	A	A
35	L	L	–	–	–	–	–	–	–	–	–	–	H	–	–	–	A	A	A	A	A	A	A	A
I/O ASSIGNMENT	M_1	M_0	S_R	S_L	S_1	S_0	B_{-3}	B_{-2}	B_{-1}	B_0	B_1	B_2	B_3	B_4	B_5	B_6	X_0	X_1	X_2	X_3	unused			

Figure 9.

An existing sequential circuit can be easily augmented or modified by using an FPLA in conjunction with a 2:1 multiplexer, and thus drastically cut costs associated with a new design.

Given a sequential circuit with the function illustrated in the map of Figure 1a, it becomes a trivial task to alter and increase its state table as in Figure 1b, by adding an FPLA and a 2:1 multiplexer to the initial circuit, as shown in Figure 2. With reference to Figure 1, the (X,Y) combinations corresponding to the circled next states in augmented state table (B) are not programmed in the FPLA; they are provided rather by the existing circuit (A). During these combinations, FPLA output k = "0" by default. The additional states of B, as well as the unmatched portion of (A) are provided by proper programming of the FPLA, at which time k = "1". The function of output k is easily implemented with virtually no dedicated P-terms, except in the case where a next state supplied by the FPLA has an all "0" coding.

D-type flip-flops will be needed to implement the augmented portion of the state table (B). However, as FPLA technology improves they can be included within the FPLA; in addition the FPLA can have more input variable (X) lines than sequential circuit (A).

A MODEL FOR MODIFYING AND/OR AUGMENTING AN EXISTING SEQUENTIAL CIRCUIT

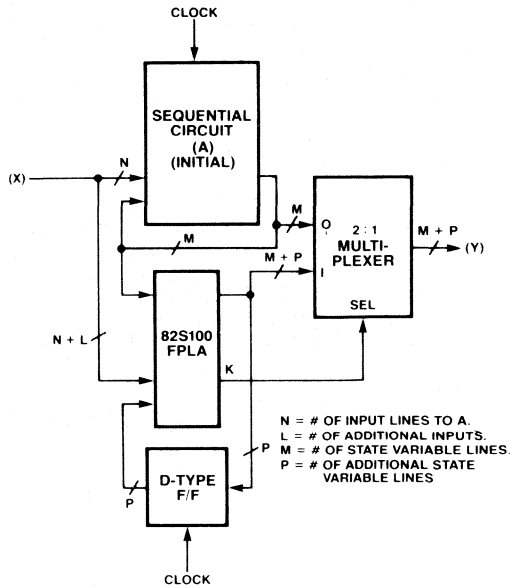


Figure 2.

SEQUENTIAL CIRCUIT MODIFICATION. TRANSITION MAPS WITH STABLE STATE ENTRIES AS A FUNCTION OF INPUT AND PRESENT STATE. CIRCLED ENTRIES DENOTE MATCHED AREAS.

Y \ X	00	01	10	11
A	(A)	D	C	(B)
B	(B)	(C)	(D)	(A)
C	(C)	(A)	B	D
D	(A)	C	(D)	B

a. Sequential circuit (A)

Change to

Y \ X	000	001	010	011	100	101
A	(A)	C	D	(B)	E	F
B	(B)	(C)	(D)	(A)	D	A
C	(C)	(A)	D	B	C	A
D	(A)	B	(D)	D	F	E
E	B	E	D	B	D	E
F	D	C	F	D	F	D

b. Modified sequential circuit

Figure 1.

Two or more dissimilar pieces of digitally tuned communications equipment can be operated with a single bandswitch using an FPLA code converter. For example, two transceivers may be simultaneously bandswitched using the code from one transceiver. In another case, a power amplifier requiring a cyclic bandswitching code may be operated from a transceiver supplying a BCD code.

In an actual case a single FPLA is capable of replacing 10 IC's and 36 pull-up resistors.

The circuit in Figure 1 illustrates how to obtain a 5-wire code, such as used for positioning a power amplifier tuning turret, from a transceiver.

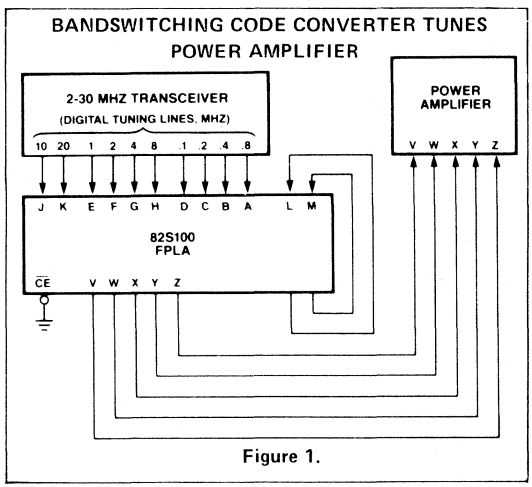


Figure 1.

The power amplifier's bandswitching code is tabulated in Figure 2. The 100KHz BCD tuning lines can be encoded in the FPLA for the X.0 to X.4 MHz and X.5 to X.9 MHz ranges according to the table in Figure 3. Two additional functions, L and M, are fed back into the FPLA to be encoded with the other BCD lines, and are defined as follows:

X.0 to X.4 MHz : $L = \overline{AB} + \overline{BCD}$

X.5 to X.9 MHz : $M = A + BC + BD$

If the other relevant BCD tuning lines are labelled E, F, G, H, J, K, for 1, 2, 3, 4, 8, 10, and 20 MHz lines respectively, a Boolean expression including the two 100KHz functions L and M can be written and encoded into the FPLA for each of the 5 desired output functions. For example, line Y of the 5-wire code is represented:

$$Y = \overline{EFGHJK}M + EF\overline{GHJK}L + EFGH\overline{JK}M + \overline{EFG}HJ\overline{K} + \overline{EFGHJK} + EFG\overline{HJK} + \overline{EFG}HJK + EFGHJ\overline{K} + \overline{EFGHJK} + EFG\overline{HJK} + \overline{EFG}HJ\overline{K} + EFGHJ\overline{K} + \overline{EFGHJK} + EFG\overline{HJK} + \overline{FHGHJK}$$

The entire equation set can be programmed in a single FPLA using 12 inputs, 7 outputs, and 35 Product Terms.

AMPLIFIER INPUT CODE ASSIGNMENT

BAND	FREQ. (MHZ)	V W X Y Z	BAND	FREQ. (MHZ)	V W X Y Z
1	2.0 – 2.4	0 0 0 0 1	16	15 – 15.9	0 0 0 1 0
2	2.5 – 2.9	0 0 0 1 1	17	16 – 16.9	0 0 1 0 1
3	3.0 – 3.4	0 0 1 1 1	18	17 – 17.9	0 1 0 1 1
4	3.5 – 3.9	0 1 1 1 1	19	18 – 18.9	1 0 1 1 0
5	4 – 4.9	1 1 1 1 0	20	19 – 19.9	0 1 1 0 1
6	5 – 5.9	1 1 1 0 1	21	20 – 20.9	1 1 0 1 0
7	6 – 6.9	1 1 0 1 1	22	21 – 21.9	1 0 1 0 1
8	7 – 7.9	1 0 1 1 1	23	22 – 22.9	0 1 0 1 0
9	8 – 8.9	0 1 1 1 0	24	23 – 23.9	1 0 1 0 0
10	9 – 9.9	1 1 1 0 0	25	24 – 24.9	0 1 0 0 1
11	10 – 10.9	1 1 0 0 1	26	25 – 25.9	1 0 0 1 1
12	11 – 11.9	1 0 0 1 0	27	26 – 26.9	0 0 1 1 0
13	12 – 12.9	0 0 1 0 0	28	27 – 27.9	0 1 1 0 0
14	13 – 13.9	0 1 0 0 0	29	28 – 28.9	1 1 0 0 0
15	14 – 14.9	1 0 0 0 1	30	29 – 29.9	1 0 0 0 0

Figure 2.

FPLA ENCODING OF kHz TUNING LINES

800 KHz (A)	400 KHz (B)	200 KHz (C)	100 KHz (D)	RANGE (MHZ)
0	0	0	0	X.0
0	0	0	1	X.1
0	0	1	0	X.2
0	0	1	1	X.3
0	1	0	0	X.4
0	1	0	1	X.5
0	1	1	0	X.6
0	1	1	1	X.7
1	0	0	0	X.8
1	0	0	1	X.9

Figure 3.

RF-CHANNEL CONTROL SELECTOR 14

The circuit shown in Figure 1 is used in the output scheduling portion of a digital controller subsystem that generates RF signals for testing receiver antennas in a multipurpose electromagnetic environment simulator.

In this system a channel match signal is used to determine available channels that can produce a signal. One channel match, or all eight channel matches can occur simultaneously. Therefore, the channel match signals and a 5-bit assignment code are used to address an FPLA programmed to resolve a preassigned priority.

The assignment code is used to control the selection of an RF channel in which a signal can be generated. The assignment parameter or code is intended to ensure that a particular signal will always appear on a specific channel when desired. Three levels of assignment priorities are used.

LEVEL 1 – No assignment; selects first available channel, beginning with lowest number channels.

LEVEL 2 – Preferential assignment; selects first available channel starting with the channel given in assignment code.

LEVEL 3 – Mandatory assignment – Selects only the channel given in assignment parameter.

The assignment parameter is a 5-bit code which is used as tabulated in Figure 2. The circuit in Figure 1 implements a matrix function for priority Levels 1 and 2. These are resident in 2 Tri-state FPLA's programmed respectively with sub-tables A and B, shown respectively in Figures 3 and 4. Both FPLAs are operated in parallel and controlled by I_{10}, the ASGN 2 signal, or by the M ASGN bit via their $\overline{CE}$ input.

Priority Level 3 is a mandatory assignment and the FPLA's are disabled. The assignment code (AS0-AS3) is decoded and, provided the channel match for the decoded channel is present, the EN CH signal is then generated.

CIRCUIT DIAGRAM OF RF CHANNEL SELECTOR

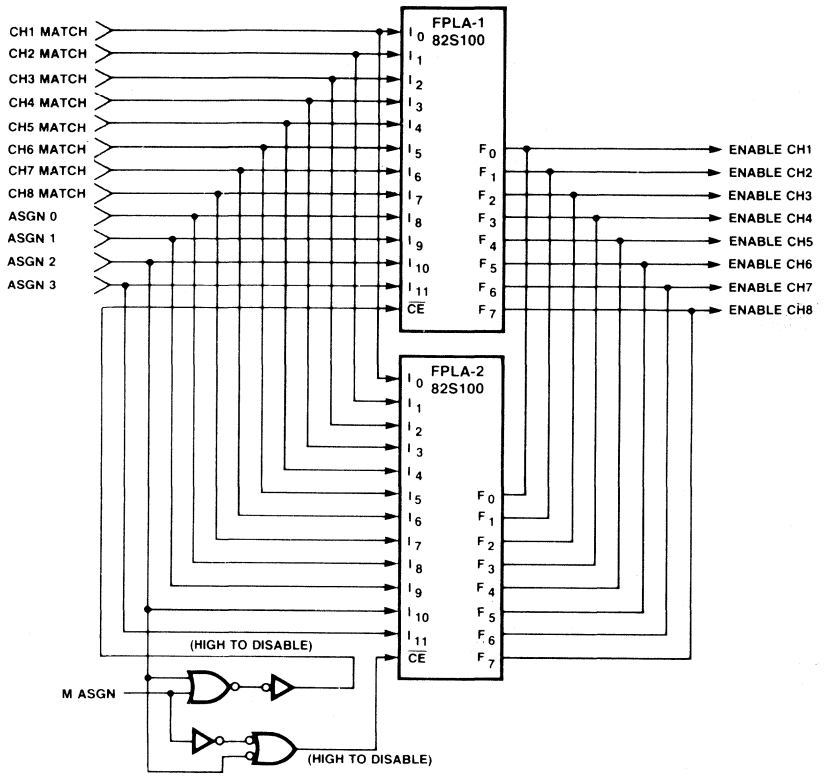


Figure 1.

CHANNEL CODE ASSIGNMENT

CASE	(msb) M ASGN*	AS3	AS2	AS1	(lsb) AS0	CHANNEL
1	O	0	0	0	0	Any
2	M	0	0	0	1	1
3	M	0	0	1	0	2
4	M	0	0	1	1	3
5	M	0	1	0	0	4
6	M	0	1	0	1	5
7	M	0	1	1	0	6
8	M	0	1	1	1	7
9	M	1	0	0	0	8

*M ASGN when a Logic "0" is a preferential assignment, and when a Logic "1" is a mandatory assignment.

Figure 2.

SUBTABLE A PROGRAMMED IN FPLA-1

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	L	L	L	–	–	–	–	–	–	–	–	H	•	•	•	•	•	•	•	A
1	–	–	–	–	L	L	L	–	–	–	–	–	–	–	H	L	•	•	•	•	•	•	A	•
2	–	–	–	–	L	L	L	–	–	–	–	–	–	H	L	L	•	•	•	•	•	A	•	•
3	–	–	–	–	L	L	L	–	–	–	–	–	H	L	L	L	•	•	•	•	A	•	•	•
4	–	–	–	–	L	L	L	–	–	–	–	H	L	L	L	L	•	•	•	A	•	•	•	•
5	–	–	–	–	L	L	L	–	–	–	H	L	L	L	L	L	•	•	A	•	•	•	•	•
6	–	–	–	–	L	L	L	–	–	H	L	L	L	L	L	L	•	A	•	•	•	•	•	•
7	–	–	–	–	L	L	L	–	H	L	L	L	L	L	L	L	A	•	•	•	•	•	•	•
8	–	–	–	–	L	L	H	L	–	–	–	–	–	–	H	–	•	•	•	•	•	•	A	•
9	–	–	–	–	L	L	H	L	–	–	–	–	–	H	L	–	•	•	•	•	•	A	•	•
10	–	–	–	–	L	L	H	L	–	–	–	–	H	L	L	–	•	•	•	•	A	•	•	•
11	–	–	–	–	L	L	H	L	–	–	–	H	L	L	L	–	•	•	•	A	•	•	•	•
12	–	–	–	–	L	L	H	L	–	–	H	L	L	L	L	–	•	•	A	•	•	•	•	•
13	–	–	–	–	L	L	H	L	–	H	L	L	L	L	L	–	•	A	•	•	•	•	•	•
14	–	–	–	–	L	L	H	L	H	L	L	L	L	L	L	–	A	•	•	•	•	•	•	•
15	–	–	–	–	L	L	H	L	L	L	L	L	L	L	L	H	•	•	•	•	•	•	•	A
16	–	–	–	–	L	L	H	H	–	–	–	–	–	H	–	–	•	•	•	•	•	A	•	•
17	–	–	–	–	L	L	H	H	–	–	–	–	H	L	–	–	•	•	•	•	A	•	•	•
18	–	–	–	–	L	L	H	H	–	–	–	H	L	L	–	–	•	•	•	A	•	•	•	•
19	–	–	–	–	L	L	H	H	–	–	H	L	L	L	–	–	•	•	A	•	•	•	•	•
20	–	–	–	–	L	L	H	H	–	H	L	L	L	L	–	–	•	A	•	•	•	•	•	•
21	–	–	–	–	L	L	H	H	H	L	L	L	L	L	–	–	A	•	•	•	•	•	•	•
22	–	–	–	–	L	L	H	H	L	L	L	L	L	L	L	H	•	•	•	•	•	•	•	A
23	–	–	–	–	L	L	H	H	L	L	L	L	L	L	H	L	•	•	•	•	•	•	A	•
24	–	–	–	–	H	L	L	L	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
25	–	–	–	–	H	L	L	L	L	–	–	–	–	–	–	H	•	•	•	•	•	•	•	A
26	–	–	–	–	H	L	L	L	L	–	–	–	–	–	H	L	•	•	•	•	•	•	A	•
27	–	–	–	–	H	L	L	L	L	–	–	–	–	H	L	L	•	•	•	•	•	A	•	•
28	–	–	–	–	H	L	L	L	L	–	–	–	H	L	L	L	•	•	•	•	A	•	•	•
29	–	–	–	–	H	L	L	L	L	–	–	H	L	L	L	L	•	•	•	A	•	•	•	•
30	–	–	–	–	H	L	L	L	L	–	H	L	L	L	L	L	•	•	A	•	•	•	•	•
31	–	–	–	–	H	L	L	L	L	H	L	L	L	L	L	L	•	A	•	•	•	•	•	•
I/O ASSIGNMENT	UNUSED				ASGN 3	ASGN 2	ASGN 1	ASGN 0	CH 8 MATCH	CH 7 MATCH	CH 6 MATCH	CH 5 MATCH	CH 4 MATCH	CH 3 MATCH	CH 2 MATCH	CH 1 MATCH	CH 8 ENB	CH 7 ENB	CH 6 ENB	CH 5 ENB	CH 4 ENB	CH 3 ENB	CH 2 ENB	CH 1 ENB

Figure 3.

SUBTABLE B STORED IN FPLA-2

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	L	H	L	L	–	–	–	–	H	–	–	–	•	•	•	•	A	•	•	•
1	–	–	–	–	L	H	L	L	–	–	–	H	L	–	–	–	•	•	•	A	•	•	•	•
2	–	–	–	–	L	H	L	L	–	–	H	L	L	–	–	–	•	•	A	•	•	•	•	•
3	–	–	–	–	L	H	L	L	–	H	L	L	L	–	–	–	•	A	•	•	•	•	•	•
4	–	–	–	–	L	H	L	L	H	L	L	L	L	–	–	–	A	•	•	•	•	•	•	•
5	–	–	–	–	L	H	L	L	L	L	L	L	L	–	–	H	•	•	•	•	•	•	•	A
6	–	–	–	–	L	H	L	L	L	L	L	L	L	–	H	L	•	•	•	•	•	•	A	•
7	–	–	–	–	L	H	L	L	L	L	L	L	L	H	L	L	•	•	•	•	•	A	•	•
8	–	–	–	–	L	H	L	H	–	–	–	H	–	–	–	–	•	•	•	A	•	•	•	•
9	–	–	–	–	L	H	L	H	–	–	H	L	–	–	–	–	•	•	A	•	•	•	•	•
10	–	–	–	–	L	H	L	H	–	H	L	L	–	–	–	–	•	A	•	•	•	•	•	•
11	–	–	–	–	L	H	L	H	H	L	L	L	–	–	–	–	A	•	•	•	•	•	•	•
12	–	–	–	–	L	H	L	H	L	L	L	L	–	–	–	H	•	•	•	•	•	•	•	A
13	–	–	–	–	L	H	L	H	L	L	L	L	–	–	H	L	•	•	•	•	•	•	A	•
14	–	–	–	–	L	H	L	H	L	L	L	L	–	H	L	L	•	•	•	•	•	A	•	•
15	–	–	–	–	L	H	L	H	L	L	L	L	H	L	L	L	•	•	•	•	A	•	•	•
16	–	–	–	–	L	H	H	L	–	–	H	–	–	–	–	–	•	•	A	•	•	•	•	•
17	–	–	–	–	L	H	H	L	–	H	L	–	–	–	–	–	•	A	•	•	•	•	•	•
18	–	–	–	–	L	H	H	L	H	L	L	–	–	–	–	–	A	•	•	•	•	•	•	•
19	–	–	–	–	L	H	H	L	L	L	L	–	–	–	–	H	•	•	•	•	•	•	•	A
20	–	–	–	–	L	H	H	L	L	L	L	–	–	–	H	L	•	•	•	•	•	•	A	•
21	–	–	–	–	L	H	H	L	L	L	L	–	–	H	L	L	•	•	•	•	•	A	•	•
22	–	–	–	–	L	H	H	L	L	L	L	–	H	L	L	L	•	•	•	•	A	•	•	•
23	–	–	–	–	L	H	H	L	L	L	L	H	L	L	L	L	•	•	•	A	•	•	•	•
24	–	–	–	–	L	H	H	H	–	H	–	–	–	–	–	–	•	A	•	•	•	•	•	•
25	–	–	–	–	L	H	H	H	H	L	–	–	–	–	–	–	A	•	•	•	•	•	•	•
26	–	–	–	–	L	H	H	H	L	L	–	–	–	–	–	H	•	•	•	•	•	•	•	A
27	–	–	–	–	L	H	H	H	L	L	–	–	–	–	H	L	•	•	•	•	•	•	A	•
28	–	–	–	–	L	H	H	H	L	L	–	–	–	H	L	L	•	•	•	•	•	A	•	•
29	–	–	–	–	L	H	H	H	L	L	–	–	H	L	L	L	•	•	•	•	A	•	•	•
30	–	–	–	–	L	H	H	H	L	L	–	H	L	L	L	L	•	•	•	A	•	•	•	•
31	–	–	–	–	L	H	H	H	L	L	H	L	L	L	L	L	•	•	A	•	•	•	•	•
I/O ASSIGNMENT	UNUSED				ASGN 3	ASGN 2	ASGN 1	ASGN 0	CH 8 MATCH	CH 7 MATCH	CH 6 MATCH	CH 5 MATCH	CH 4 MATCH	CH 3 MATCH	CH 2 MATCH	CH 1 MATCH	CH 8 ENB	CH 7 ENB	CH 6 ENB	CH 5 ENB	CH 4 ENB	CH 3 ENB	CH 2 ENB	CH 1 ENB

Figure 4.

The circuit shown in Figure 1 filters asynchronous serial data as it enter an asynchronous receiver. These devices, often used to deserialize Teletype signals, typically sample each bit of a character only once, in the middle of the bit time.

CIRCUIT DIAGRAM OF SERIAL DATA FILTER

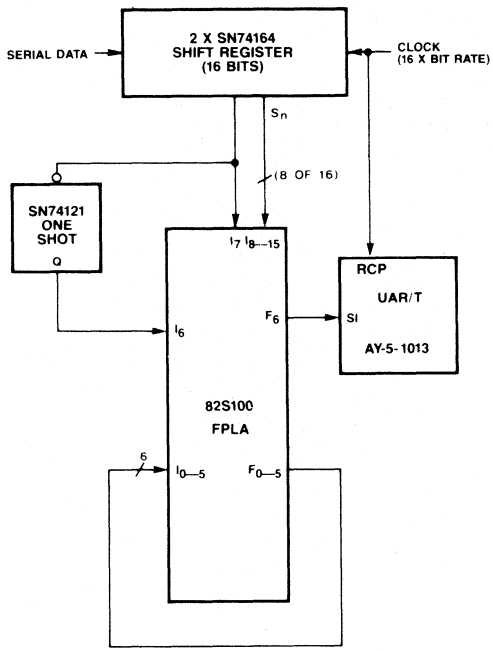


Figure 1.

If a noise pulse, perhaps caused by dirty Teletype distributor, occurs at a sample time, an error results. Low-pass filtering may remove the noise, but may also cause the receiver to start late and erratically since the leading edge of the start bit, used for a time reference is blurred.

A better solution to this problem can be obtained by using an FPLA to implement a 5-out-of 9 majority function from 9 samples of each bit stored in a 16-bit shift register. Since asynchronous receivers typically require a clock at 16 times the bit rate, this is used for the shift register also. For a clean start, an additional input is used to select between a single sample and the majority function. It is controlled by a one-shot which, when fired by the start bit, substitutes the majority function for the middle sample during the rest of the character.

Since a straightforward implementation of the majority function would require 6435 product terms, a 2 level approach is used. The inputs are grouped into triplets (I_{7-9}, I_{10-12}, and I_{13-15},). The 1-out-of-3 functions (F_0, F_1, and F_2, respectively) and 2-out-of-3 functions (F_3, F_4, and F_5, respectively) are generated for each triplet. These are fed back to unused inputs and the complete function obtained from output F_6. The complete equation set is tabulated in Fig. 2, which indicates that there are 16 inputs, 29 product terms, and 7 outputs. These are programmed in the FPLA Program Table of Fig. 3.

Other shift registers and receivers will work just as well as those shown. And, if a smaller number of samples is used, say for 4-out-of-8, set $I_2 = I_{15}$ = "1" in the FPLA equations.

The samples should be distributed nearly evenly across the shift register, but may be adjusted to accommodate known signal characteristics such as uncertain bit-boundary locations. The concept can be extended to other systems, such as communication equipment for telephone links etc., even to include nonuniform input weighting to solve a particular noise problem by modifying the FPLA program.

LOGIC EQUATION SET RESIDENT IN THE FPLA

$$I_0 = F_0 = \overline{\bar{I}_7 \bar{I}_8 \bar{I}_9} = I_7 + I_8 + I_9$$

$$I_1 = F_1 = \overline{\bar{I}_{10} \bar{I}_{11} \bar{I}_{12}} = I_{10} + I_{11} + I_{12}$$

$$I_2 = F_2 = \overline{\bar{I}_{13} \bar{I}_{14} \bar{I}_{15}} = I_{13} + I_{14} + I_{15}$$

$$I_3 = F_3 = I_7 I_8 + I_7 I_9 + I_8 I_9$$

$$I_4 = F_4 = I_{10} I_{11} + I_{10} I_{12} + I_{11} I_{12}$$

$$I_5 = F_5 = I_{13} I_{14} + I_{13} I_{15} + I_{14} I_{15}$$

$$F_6 = I_6 I_0 I_1 I_{13} I_{14} I_{15} + I_6 I_0 I_4 I_5 + I_6 I_0 I_{10} I_{11} I_{12} I_2 + I_6 I_3 I_{13} I_{14} I_{15} + I_6 I_3 I_1 I_5 + I_6 I_3 I_4 I_2 + I_6 I_3 I_{10} I_{11} I_{12} + I_6 I_7 I_8 I_9 I_5 + I_6 I_7 I_8 I_9 I_1 I_2 + I_6 I_7 I_8 I_9 I_4 + \bar{I}_6 I_7$$

Figure 2.

FPLA PROGRAM TABLE FOR 5-OUT-OF-9 MAJORITY FUNCTION
FOR 9-BIT WORD SAMPLE OF INPUT DATA

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
1	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
2	–	–	–	–	–	–	H	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
3	–	–	–	–	–	H	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•
4	–	–	–	–	H	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•
5	–	–	–	H	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•
6	–	–	H	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•
7	–	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•
8	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•
9	–	–	–	–	–	–	–	H	H	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
10	–	–	–	–	–	–	H	–	H	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
11	–	–	–	–	–	–	H	H	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
12	–	–	–	–	H	H	–	–	–	–	–	–	–	–	–	–	A	•	•	A	•	•	•	•
13	–	–	–	H	–	H	–	–	–	–	–	–	–	–	–	–	A	•	•	A	•	•	•	•
14	–	–	–	H	H	–	–	–	–	–	–	–	–	–	–	–	A	•	•	A	•	•	•	•
15	–	H	H	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	•	•	•	•	•
16	H	–	H	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	•	•	•	•	•
17	H	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	•	•	•	•	•
18	H	H	H	–	–	–	–	–	–	H	–	–	–	–	H	H	A	A	•	•	•	•	•	•
19	–	–	–	–	–	–	–	–	–	H	H	H	–	–	–	H	A	A	•	•	•	•	•	•
20	–	–	–	H	H	H	–	–	–	H	–	–	–	H	–	H	A	A	•	•	•	•	•	•
21	H	H	H	–	–	–	–	–	–	H	–	–	H	–	–	–	A	A	•	•	•	•	•	•
22	–	–	–	–	–	–	–	–	–	H	H	–	H	–	H	–	A	A	•	•	•	•	•	•
23	–	–	–	–	–	–	–	–	–	H	–	H	H	H	–	–	A	A	•	•	•	•	•	•
24	–	–	–	H	H	H	–	–	–	H	–	–	H	–	–	–	A	A	•	•	•	•	•	•
25	–	–	–	–	–	–	H	H	H	H	H	–	–	–	–	–	A	A	•	•	•	•	•	•
26	–	–	–	–	–	–	H	H	H	H	–	–	–	H	H	–	A	A	•	•	•	•	•	•
27	–	–	–	–	–	–	H	H	H	H	–	H	–	–	–	–	A	A	•	•	•	•	•	•
28	–	–	–	–	–	–	–	–	H	L	–	–	–	–	–	–	A	A	•	•	•	•	•	•
I/O ASSIGNMENT	S_{15}	S_{13}	S_{11}	S_9	S_7	S_5	S_3	S_0	S_8	Q	F_5	F_4	F_3	F_2	F_1	F_0	unused	SI	I_5	I_4	I_3	I_2	I_1	I_0

Figure 3.

The circuit shown in Figure 1 generates an output pulse on one of eight channels whenever the duration of an input pulse falls within the preselected window corresponding to that channel.

The input pulse is applied to input I_0, of the 82S100. It also drives network R_1 C_1, whose exponentially rising and falling output voltage is squared up by G1 to look like a 1/4 μS delayed, inverted version of the input pulse, and is applied to input I_1 of the FPLA. The FPLA forms the logical product $\overline{I_0} \cdot I_1$ (see timing diagram, Figure 2) as part of each of its 48 product terms, to mark the trailing edge of the input pulse. Simultaneously, the input pulse turns on crystal controlled multivibrator G2-G3, whose pulse output is accumulated by counters IC2 and IC3. Thus as long as the input pulse is high, FPLA inputs I_2 through I_9 will indicate elapsed microseconds. The FPLA is programmed with successive counts and selector switch settings as shown in the Program Table of Figure 3, so that when the trailing edge signal $\overline{I_0} \cdot \overline{I_1}$ is coincident with a specific count and switch setting, the coincidence pulse will appear on the appropriate output line. Network R_2 C_2 differentiates the trailing edge of the delayed input pulse to provide a reset pulse for the counter. The MSB on counter IC3 is inverted by G4 to cage multivibrator G2-G3 whenever the counter overranges (i.e., goes to its upper half-range). This avoids the ambiguity created when long extraneous noise pulses drive the multivibrator for several full counting cycles (256n counts) past any specific desired count. If the pulse transmission line is known to be free of noise pulses, this circuit can be deleted, doubling the count capacity of the counter. With the given program, the FPLA will detect input pulses in the pulse width windows 1 to 2μS, 2 to 3μS, 3 to 4μS, ... , 48 to 49μS. By programming the FPLA so that count bits I_2 through I_9 range from 49 (LLHHLLLH) through 96 (LHHLLLLL) the windows 49 to 50 μS, ... , 96 to 97 μS can be detected instead. Thus with the overrange lock-out by G_4, up to 127 separate pulse widths can be detected by some change in the programming; without the overrange lock-out implemented this number increases to 255.

This circuit can be used to control multi-function machines in several locations throughout a small plant by using the power mains as a pulse transmission line. The system is relatively noise-immune because of the narrow tolerances on the control pulse-widths. The output pulses F_0 through F_7 can be used to control latching solid state A.C. relays, SCRs for motor phase-control, stepper motors, and can even relay audio information by toggling a flip-flop whose filtered, buffered output is applied to a speaker.

PULSE WIDTH DISCRIMINATOR.

G_{1-4} ARE 7400 GATES

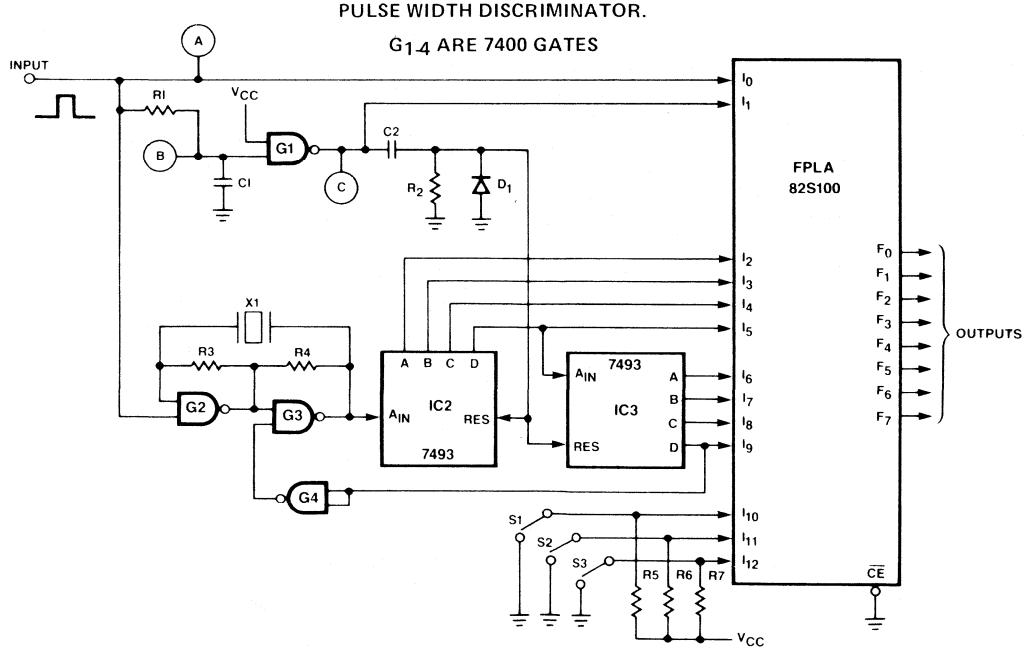


Figure 1.

PARTIAL TIMING DIAGRAM

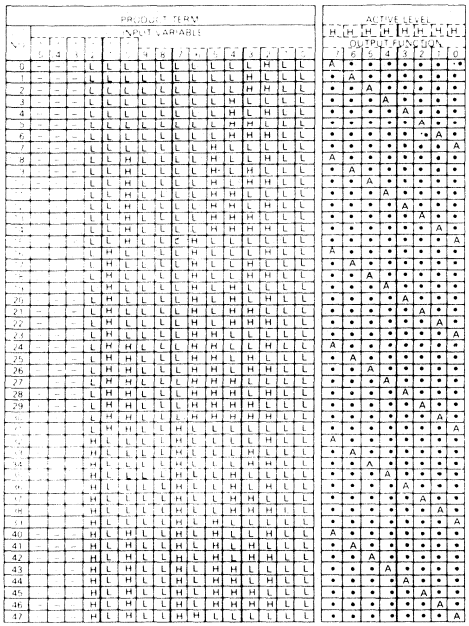


Figure 2.

FPLA PROGRAM FOR 1μS PULSE WINDOWS UP TO 50μS.

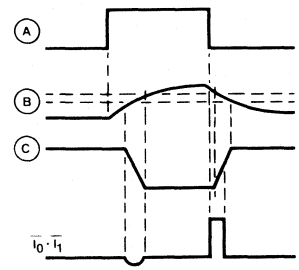


Figure 3.

TIMING DECODER & SEQUENCE CONTROLLER FOR DATA ACQUISITION SYSTEM

The functional diagram of Figure 1 shows a data acquisition peripheral which samples four analog signals and sequentially supplies digitized data to a minicomputer. Computer commands vary sample timing. Four digitized samples are sent to the computer via interrupts every one millisecond.

The system operates with a basic repetition interval of 1Ms. Sample acquisition time begins at the start of each 1Msec interval, and samples are held at various times after the beginning of the interval. Sample No. 1 is always held at 200 μsec into the interval. Sample No. 2 follows No. 1 by 2.5 μsec. The timing of Samples 3 and 4 are varied before and after sample 1 in 50 μsec steps by a 3-bit command from the computer. Sample 4 follows Sample 3 by 2.5 μsec. The timing commands and the resulting Sample 3 hold times (from start of 1Ms) are:

Command	0	1	2	3	4	5	6	
Sample 3 Time	50	100	150	200	250	300	350	. . .(μsec)

As shown in the circuit diagram of Figure 2, the counter outputs and the command bits are decoded in FPLA No. 1 to provide turn-off signals to the Sample & Hold gate flip-flops. A 16-state sequential controller operates the multiplexers, A/D converter and interfaces with the computer interrupt system. The controller is implemented in FPLA No. 2 and is expanded into the remaining parts of FPLA No. 1 because more than 8 outputs are required. Both FPLA Program Tables and I/O assignment are tabulated in Figures 4 and 5, respectively.

The Sequence Controller decodes counter outputs at the end of the 1 Msec interval and resets the counter synchronously. The Sample gate flip-flops are also turned on. The controller then waits until Sample 1 is held and starts the A/D Converter pulsing an output for one cycle. When the converter is no longer busy, the controller sets the interrupt output active until the computer recognizes it, and responds by reading the A/D output data. The multiplexer is advanced to Sample 2 and the process is repeated. The A/D conversion takes longer than 2.5 μsec, so Sample 2 can be converted immediately. Conversion of Sample 2 may finish before Sample 3 is held in the latest cases, so the sequencer may have to wait before converting. After the conversion and interrupt sequence is completed for all four samples, the sequencer waits for the end of the 1 Msec interval. One of the multiplexer switches is active at all times. A flowchart

of the control sequence is shown in Figure 3. The system interface is through the following signals:

Clock: 2 MHz TTL square wave.

Analog Inputs: F_C = 100 KHz, 30 KHz Bandwidth Amplitude: ± 1 volt.

Data from Computer: 3 bits of command data and a strobe. The strobe is pulsed to load the command register after the computer has responded to all four interrupts in an interval.

Data to Computer: The A/D Converter provides 10 bits of parallel data in TTL compatible levels.

Interrupt: Low TTL level active while the controller has valid data for the computer to read.

Response to Interrupt: A 2 μsec active-low pulse after the computer has read the A/D data.

FUNCTIONAL DIAGRAM OF DATA ACQUISITION SYSTEM

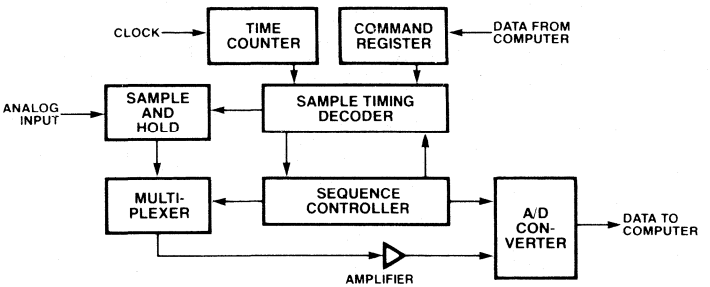


Figure 1.

CIRCUIT DIAGRAM OF DAS CONTROLLER

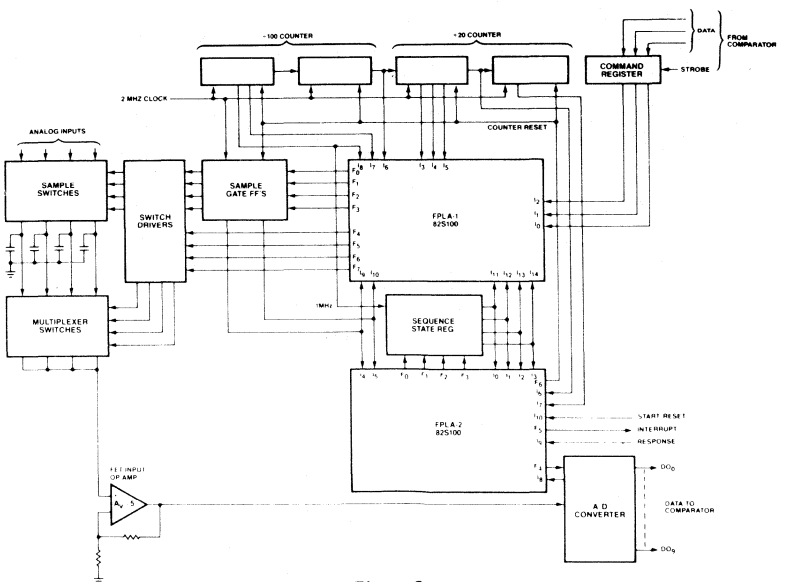


Figure 2.

FLOWCHART OF A/D CONVERSION AND DATA INTERFACE SEQUENCE

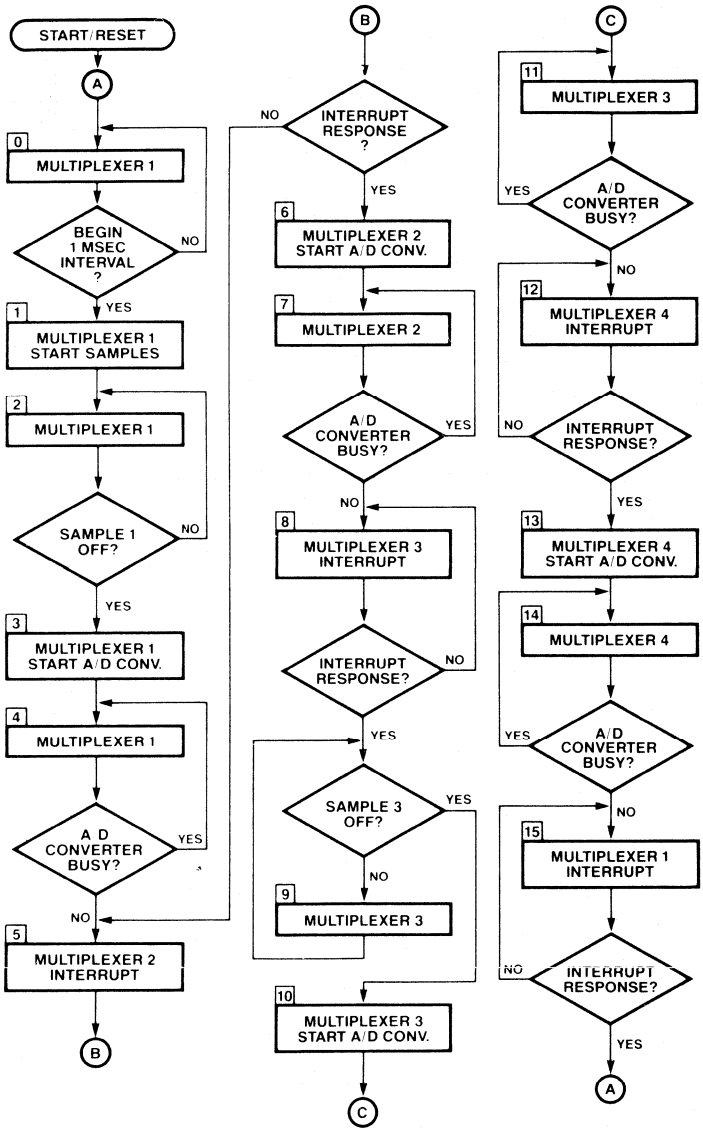


Figure 3.

PROGRAM TABLE FOR FPLA 1

	PRODUCT TERM															
	INPUT VARIABLE															
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	–	L	L	–	–	–	–	–	–	–	–	–	–	–	–	–
1	–	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–
2	–	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–
3	–	L	H	L	H	–	–	–	–	–	–	–	–	–	–	–
4	–	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–
5	–	H	L	–	–	–	–	–	–	–	–	–	–	–	–	–
6	–	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–
7	–	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–
8	–	–	–	–	–	–	L	H	H	–	–	–	–	–	–	–
9	–	–	–	–	–	L	–	H	H	–	–	–	–	–	–	–
10	–	–	–	–	–	–	–	–	–	H	L	H	H	–	–	–
11	–	–	–	–	–	–	–	–	–	H	L	L	L	L	L	L
12	–	–	–	–	–	–	–	–	–	H	L	L	H	L	L	H
13	–	–	–	–	–	–	–	–	–	H	L	H	L	L	H	L
14	–	–	–	–	–	–	–	–	–	H	L	H	H	L	H	H
15	–	–	–	–	–	–	–	–	–	H	H	L	L	H	L	L
16	–	–	–	–	–	–	–	–	–	H	H	L	H	H	L	H
17	–	–	–	–	–	–	–	–	–	H	H	H	L	H	H	L

	ACTIVE LEVEL							
	L	L	L	L	H	H	H	H
	OUTPUT FUNCTION							
	7	6	5	4	3	2	1	0
	•	•	•	A	•	•	•	•
	•	•	•	A	•	•	•	•
	•	•	•	A	•	•	•	•
	•	•	A	•	•	•	•	•
	•	•	A	•	•	•	•	•
	•	A	•	•	•	•	•	•
	A	•	•	•	•	•	•	•
	A	•	•	•	•	•	•	•
	•	•	•	•	•	•	A	•
	•	•	•	•	A	•	•	•
	•	•	•	•	•	•	•	A
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•
	•	•	•	•	•	A	•	•

I/O ASSIGNMENT

0-2: Command Register bits.
3-5: Count of 50 μsec intervals from ÷ 20 counter.
6: .5 μsec pulse every 50 μsec.
7: 2 μsec output from ÷ 100 counter.
8: .5 μsec output from ÷ 100 counter. This is also the signal used to clock the sequence state register.
9-10: Sample 1 and 3 gates.
11-14: Four sequence state bits.

0-3: Four sample Gate turn-off signals.
4-7: Four multiplexer control signals.

Figure 4.

PROGRAM TABLE FOR FPLA-2

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H	L	L	H	H	H	H	H	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	H	–	–	–	–	–	–	–	–	–	–									A	A	•	•	•	•	•	A
1	–	–	–	–	–	L	–	–	H	H	–	–	L	L	L	L									A	A	•	•	•	•	•	A
2	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H									A	A	•	•	•	•	A	•
3	–	–	–	–	–	L	–	–	–	–	–	H	L	L	H	L									A	•	•	•	•	•	A	•
4	–	–	–	–	–	L	–	–	–	–	–	L	L	L	H	L									A	•	•	A	•	•	A	A
5	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	H									A	•	•	A	•	A	•	•
6	–	–	–	–	–	L	–	H	–	–	–	–	L	H	L	L									A	•	•	•	•	A	•	•
7	–	–	–	–	–	L	–	L	–	–	–	–	L	H	L	L									A	•	A	•	•	A	•	A
8	–	–	–	–	–	L	H	–	–	–	–	–	L	H	L	H									A	•	A	•	•	A	•	A
9	–	–	–	–	–	L	L	L	–	–	–	–	L	H	L	H									A	•	•	A	•	A	A	•
10	–	–	–	–	–	L	–	–	–	–	–	–	L	H	H	L									A	•	•	A	•	A	A	A
11	–	–	–	–	–	L	–	H	–	–	–	–	L	H	H	H									A	•	•	•	•	A	A	A
12	–	–	–	–	–	L	–	L	–	–	–	–	L	H	H	H									A	•	A	•	A	•	•	•
13	–	–	–	–	–	L	H	–	–	–	–	–	H	L	L	L									A	•	A	•	A	•	•	•
14	–	–	–	–	–	L	L	–	–	–	H	–	H	L	L	L									A	•	•	•	A	•	•	A
15	–	–	–	–	–	L	L	–	–	–	L	–	H	L	L	L									A	•	•	A	A	•	A	•
16	–	–	–	–	–	L	–	–	–	–	H	–	H	L	L	H									A	•	•	•	A	•	•	A
17	–	–	–	–	–	L	–	–	–	–	L	–	H	L	L	H									A	•	•	A	A	•	A	•
18	–	–	–	–	–	L	–	–	–	–	–	–	H	L	H	L									A	•	•	A	A	•	A	A
19	–	–	–	–	–	L	–	H	–	–	–	–	H	L	H	H									A	•	•	•	A	•	A	A
20	–	–	–	–	–	L	–	L	–	–	–	–	H	L	H	H									A	•	A	•	A	A	•	•
21	–	–	–	–	–	L	H	–	–	–	–	–	H	H	L	L									A	•	A	•	A	A	•	•
22	–	–	–	–	–	L	L	–	–	–	–	–	H	H	L	L									A	•	•	A	A	A	•	A
23	–	–	–	–	–	L	–	–	–	–	–	–	H	H	L	H									A	•	•	A	A	A	A	•
24	–	–	–	–	–	L	–	H	–	–	–	–	H	H	H	L									A	•	•	•	A	A	A	•
25	–	–	–	–	–	L	–	L	–	–	–	–	H	H	H	L									A	•	A	•	A	A	A	A
26	–	–	–	–	–	L	H	–	–	–	–	–	H	H	H	H									A	•	A	•	A	A	A	A

I/O ASSIGNMENT	
	0-3: Four sequence state bits. 4-5: Sample 1 and 3 gates. 6-7: ÷ 20 counter outputs to decode 1 Msec. 8: A/D Converter busy. 9: Computer interrupt response. 10: Start/Reset signal from computer.

0-3: Four sequence state bits for next step. 4: A/D Converter start pulse-high logic level for 1 μsec. 5: Interrupt to computer. 6: Reset signal to counter and Sample Gates

Figure 5.

One application where the FPLA proves ideal is in the control for time-division multiplexing a data channel among a number of data sources according to a fixed schedule, or a small number of selectable fixed schedules.

In the system shown in Figure 1 there are two identical instruments providing data words upon demand. The proportion of the channel capacity allocated to each data source depends on the frequency with which that information is needed for the eventual data evaluation process. In this case there are three different formats which may be selected, depending on the currently available channel capacity, which requires skewing the ideal data word mix to neatly fit the aggregate bit rate. A further variation allows the choice of ignoring one of the two instruments, and devote the entire data output to the other instrument to allow higher resolution measurements to be performed. These functions are summarized in the Major Frame logic truth-table of Figure 2, and incorporated in two FPLAs programmed as in Figures 3 and 4.

Each half-second increment in the format represents one complete measurement cycle for the instrument. Additional FPLAs can be used to implement a Minor Frame logic function for decoding time increments within that half-second interval to flag the start of new operating modes (zero-check, calibrate, gain-range, sweep, etc.), initiate old converter sequences, etc.

The outputs of the FPLAs for both Major and Minor logic decoding functions are logically ANDed together to produce the actual enable signals which connect a given data source to the output bus. This output bus then feeds the input of a first-in/first-out memory to smooth the data flow for application to the uniform-rate synchronous data channel. Applied to an actual working design, the two FPLAs for the Major Frame logic replaced 11 MSI and SSI logic packages.

BLOCK DIAGRAM OF SINGLE TDM DATA CHANNEL

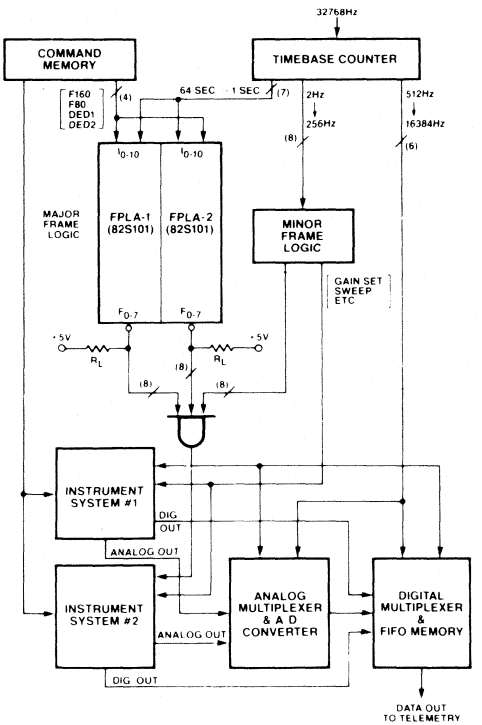


Figure 1.

FUNCTIONAL TRUTH TABLE FOR MAJOR FRAME LOGIC

F160	F80	Function
0	0	128 BPS Format
0	1	80 BPS Format
1	0	160 BPS Format
1	1	not allowed
DED 1	**DED 2**	**Function**
0	0	Timeshared Format
0	1	TM dedicated to System 2
1	0	TM dedicated to System 1
1	1	not allowed

Figure 2.

PROGRAM TABLE FOR FPLA 1 IN MAJOR FRAME LOGIC

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																L	L	L	H	L	L	L	L
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	L	L	L	L	–	–	–	–	H	L	L	•	•	•	A	•	•	•	A
1	–	–	–	–	–	L	L	L	L	L	–	–	L	L	H	L	•	•	•	A	•	•	•	A
2	–	–	–	–	–	L	L	L	L	H	–	–	H	L	H	L	•	•	•	A	•	•	•	A
3	–	–	–	–	–	L	L	–	–	–	–	–	–	H	H	L	•	•	•	A	•	•	A	•
4	–	–	–	–	–	L	L	L	L	–	–	–	L	L	H	L	•	•	•	A	•	•	A	•
5	–	–	–	–	–	L	L	L	H	–	–	–	H	L	H	L	•	•	•	A	•	•	A	•
6	–	–	–	–	–	L	–	–	–	–	–	–	–	H	H	L	•	•	•	A	•	A	•	•
7	–	–	–	–	–	L	L	–	–	–	–	–	L	L	H	L	•	•	•	A	•	A	•	•
8	–	–	–	–	–	L	H	–	–	–	–	–	H	L	H	L	•	•	•	A	•	A	•	•
9	–	–	–	–	–	L	L	L	L	L	L	L	–	H	–	L	•	•	•	A	A	•	•	•
10	–	–	–	–	–	L	L	L	L	L	L	L	L	L	–	L	•	•	•	A	A	•	•	•
11	–	–	–	–	–	–	L	L	L	L	L	H	–	H	–	L	•	•	•	A	•	•	•	•
12	–	–	–	–	–	–	L	L	L	L	L	H	L	L	–	L	•	•	•	A	•	•	•	•
13	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	L	•	•	A	A	•	•	•	•
14	–	–	–	–	–	–	L	–	–	–	–	–	H	–	–	L	•	A	•	A	•	•	•	•
15	–	–	–	–	–	–	H	–	–	–	–	–	L	–	–	L	•	A	•	A	•	•	•	•
16	–	–	–	–	–	–	L	L	L	L	L	L	–	–	–	L	A	•	•	A	•	•	•	•
17	–	–	–	–	–	L	L	L	–	–	–	–	–	H	L	–	•	•	•	A	•	•	•	A
18	–	–	–	–	–	L	L	L	L	–	–	–	L	L	L	H	•	•	•	A	•	•	•	A
19	–	–	–	–	–	L	L	L	H	–	–	–	H	L	L	H	•	•	•	A	•	•	•	A
20	–	–	–	–	–	L	L	–	–	–	–	–	–	H	L	–	•	•	•	A	•	•	A	•
21	–	–	–	–	–	L	L	L	–	–	–	L	–	L	L	H	•	•	•	A	•	•	A	•
22	–	–	–	–	–	L	L	H	–	–	–	H	–	L	L	H	•	•	•	A	•	•	A	•
23	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	•	•	•	A	•	A	•	•
24	–	–	–	–	–	L	L	L	L	L	L	–	–	H	L	H	•	•	•	A	A	•	•	•
25	–	–	–	–	–	L	L	L	L	L	L	L	–	L	L	H	•	•	•	A	A	•	•	•
26	–	–	–	–	–	L	L	L	L	L	L	H	–	L	L	H	•	•	•	A	•	•	•	•
27	–	–	–	–	–	L	–	–	–	–	–	–	–	H	–	–	•	•	A	A	•	•	•	•
28	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H	•	•	A	A	•	•	•	•
29	–	–	–	–	–	–	L	–	–	–	–	H	–	–	L	H	•	A	•	A	•	•	•	•
30	–	–	–	–	–	–	H	–	–	–	–	L	–	–	L	H	•	A	•	A	•	•	•	•
31	–	–	–	–	–	–	L	L	L	L	L	L	–	–	L	H	A	•	•	A	•	•	•	•
32	–	–	–	–	–	L	L	L	L	–	–	L	–	L	L	L	•	•	•	A	•	•	•	A
33	–	–	–	–	–	L	L	L	H	–	–	H	–	L	L	L	•	•	•	A	•	•	•	A
34	–	–	–	–	–	L	L	L	–	–	–	–	L	L	L	L	•	•	•	A	•	•	A	•
35	–	–	–	–	–	L	L	H	–	–	–	–	H	L	L	L	•	•	•	A	•	•	A	•
36	–	–	–	–	–	L	–	–	–	–	–	L	–	L	L	L	•	•	>	A	•	•	•	•
I/O ASSIGNMENT	UNUSED					DED 2	1 Sec	2 Sec	4 Sec	8 Sec	16 Sec	32 Sec	64 Sec	DED 1	F80	F160	HEN	CNARR	SC1SAM	unused	SC1EN	DV1EN	NI1EN	VO1EN

Figure 3.

PROGRAM TABLE FOR FPLA 2 IN MAJOR FRAME LOGIC

NO	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: H	H	L	L	H	L	L	L
																	OUTPUT FUNCTION: 7	6	5	4	3	2	1	0
0	–	–	–	–	–	L	L	L	L	–	–	–	–	H	L	L	A	A	•	•	A	•	•	A
1	–	–	–	–	–	L	L	L	L	L	–	–	L	L	H	L	A	A	•	•	A	•	•	A
2	–	–	–	–	–	L	L	L	L	H	–	–	H	L	H	L	A	A	•	•	A	•	•	A
3	–	–	–	–	–	L	L	L	–	–	–	–	–	H	H	L	A	A	•	•	A	•	A	•
4	–	–	–	–	–	L	L	L	L	–	–	–	L	L	H	L	A	A	•	•	A	•	A	•
5	–	–	–	–	–	L	L	L	H	–	–	–	H	L	H	L	A	A	•	•	A	•	A	•
6	–	–	–	–	–	L	–	–	–	–	–	–	–	H	H	L	A	A	•	•	A	A	•	•
7	–	–	–	–	–	L	L	–	–	–	–	–	L	L	H	L	A	A	•	•	A	A	•	•
8	–	–	–	–	–	L	H	–	–	–	–	–	H	L	H	L	A	A	•	•	A	A	•	•
9	–	–	–	–	–	L	L	L	L	L	L	L	–	H	–	L	A	A	•	•	A	•	•	•
10	–	–	–	–	–	L	L	L	L	L	L	L	L	L	–	L	A	A	•	•	A	•	•	•
11	–	–	–	–	–	–	L	L	L	L	L	H	–	H	–	L	A	A	•	A	A	•	•	•
12	–	–	–	–	–	–	L	L	L	L	L	H	L	L	–	L	A	A	•	A	A	•	•	•
13	–	–	–	–	–	L	–	–	–	–	–	–	L	L	H	L	A	A	A	•	A	•	•	•
14	–	–	–	–	–	–	–	–	–	–	–	–	H	–	–	L	A	A	•	•	A	•	•	•
15	–	–	–	–	–	H	–	–	–	–	–	–	L	–	–	L	A	A	•	•	A	•	•	•
16	–	–	–	–	–	L	L	L	L	L	L	L	–	–	–	L	A	A	•	•	A	•	•	•
17	–	–	–	–	–	L	L	L	–	–	–	–	–	H	L	–	A	A	•	•	A	•	•	A
18	–	–	–	–	–	L	L	L	L	–	–	–	L	L	L	H	A	A	•	•	A	•	•	A
19	–	–	–	–	–	L	L	L	H	–	–	–	H	L	L	H	A	A	•	•	A	•	•	A
20	–	–	–	–	–	L	L	–	–	–	–	–	–	H	L	–	A	A	•	•	A	•	A	•
21	–	–	–	–	–	L	L	L	–	–	–	L	–	L	L	H	A	A	•	•	A	•	A	•
22	–	–	–	–	–	L	L	H	–	–	–	H	–	L	L	H	A	A	•	•	A	•	A	•
23	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–	A	A	•	•	A	A	•	•
24	–	–	–	–	–	L	L	L	L	L	L	–	–	H	L	H	A	A	•	A	A	•	•	•
25	–	–	–	–	–	L	L	L	L	L	L	L	–	L	L	H	A	A	•	•	A	•	•	•
26	–	–	–	–	–	L	L	L	L	L	L	H	–	L	L	H	A	A	•	A	A	•	•	•
27	–	–	–	–	–	L	–	–	–	–	–	–	–	H	–	–	A	A	A	•	A	•	•	•
28	–	–	–	–	–	L	–	–	–	–	–	–	L	L	L	H	A	A	A	•	A	•	•	•
29	–	–	–	–	–	–	L	–	–	–	–	H	–	–	L	H	A	A	•	•	A	•	•	•
30	–	–	–	–	–	–	H	–	–	–	–	L	–	–	L	H	A	A	•	•	A	•	•	•
31	–	–	–	–	–	–	L	L	L	L	L	–	–	–	L	H	A	A	•	•	A	•	•	•
32	–	–	–	–	–	L	L	L	L	–	–	L	–	L	L	L	A	A	•	•	A	•	•	A
33	–	–	–	–	–	L	L	L	H	–	–	H	–	L	L	L	A	A	•	•	A	•	•	A
34	–	–	–	–	–	L	L	L	–	–	–	–	L	L	L	L	A	A	•	•	A	•	A	•
35	–	–	–	–	–	L	L	H	–	–	–	–	H	L	L	L	A	A	•	•	A	•	A	•
36	–	–	–	–	–	L	–	–	–	–	–	L	–	L	L	L	A	A	A	•	A	•	•	•
I/O ASSIGNMENT	UNUSED					DED 1	1 Sec	2 Sec	4 Sec	8 Sec	16 Sec	32 Sec	64 Sec	DED 2	F 80	F 160	unused		SC2 SAM	SC2 EN	unused	DV2 EN	NI2 EN	VO2 EN

Figure 4.

Until now designers of ATC (Air Traffic Control) transponder encoders have had little choice other than stacking up numerous stages of shift registers and banks of gates, or go the route of custom LSI. The former choice involves close to two dozen chips, while the second entails a large initial expense for masks. The design of Figure 1 uses an FPLA to solve the problem of converting large amounts (25 lines total) of parallel data into unique combinations of serial bit streams using only six commercially available chips. While the full capability of the FPLA's is not utilized, the design represents considerable savings over previous options, and with only slight modifications can be used in other parallel to serial data conversion applications.

With reference to Figure 1, the start command resets A4. The outputs of A4 start the external clock and allow counters A2 and A5 to toggle. A2 and A5 count until maximum count (19) is decoded by A1, which in turn sets A4, thereby stopping the cycle and resetting the counters. During the first 16 clock pulses, either the Mode-A or Mode-C programming is outputted from A3 or A6 respectively, depending on the state of the MODE CONTROL as gated through A7, A8 and A9. The special Mode-A IDENT pulse is decoded on the eighteenth (18th) clock pulse and enabled all through gates A10 and A11. A12 gates the output with the clock to eliminate ripple-through spikes and race problems.

ATC TRANSPONDER ENCODER CIRCUIT DIAGRAM

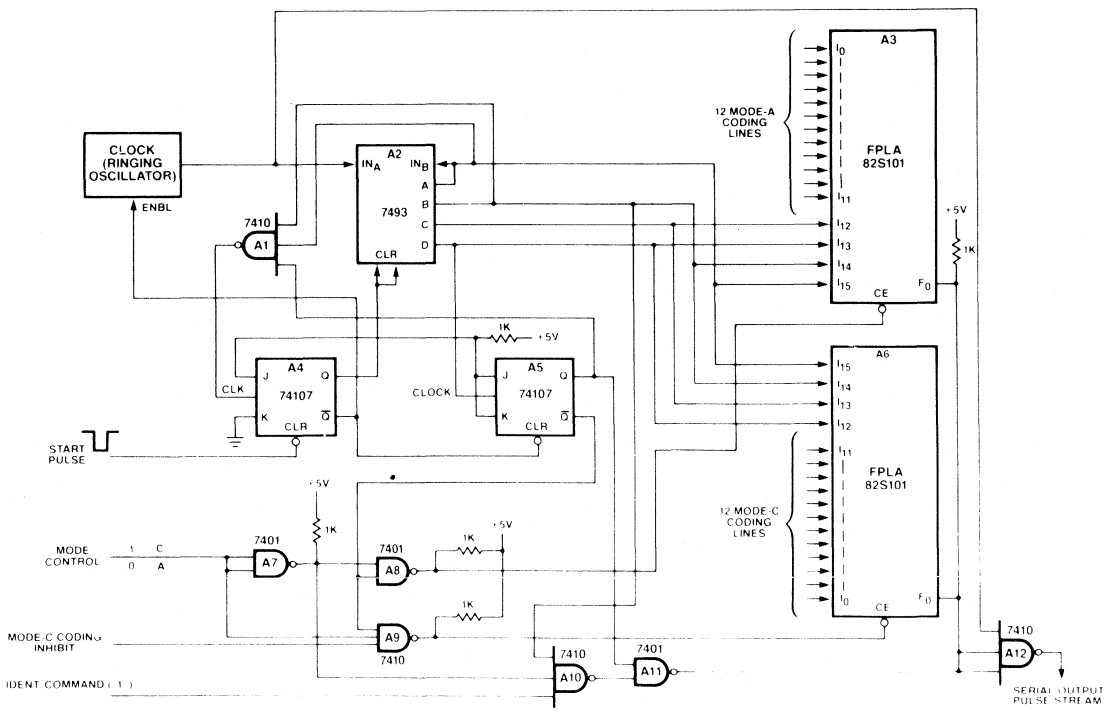


Figure 1.

In many applications it is necessary to have multi channel two way radiotelephone for mobile communications. However, in emergency situations it is important to be able to change channels rather quickly, especially when the transceiver is mounted in a vehicle where the driver cannot look at the radio channel selector switch and the driveway at the same time. This situation is analogous to the car radio stations. Channel selection can be made easier by incorporating in the transceiver a Frequency Synthesizer with a Field Programmable Logic Array as a Channel Memory, in which the channels are programmed digitally and are selected by a keyboard that has the channels designations JJ, JL, YL, . . . etc.

The block diagram of Figure 1 shows a Phase-Locked Channel Selector Synthesizer using a Hughes HCTR0320 Frequency Synthesizer integrated circuit operating at low frequencies (1-3 MHz), together with a Voltage Controlled Oscillator (VCO) and a low pass filter for closing the loop.

The HCTR0320 chip has a programmable counter/divider, with BCD and binary inputs, which sets the output frequency to an exact multiple of the reference frequency. The reference frequency is generated by a stable crystal controlled oscillator and divided down by a prescaler (counter-divider) for supplying the 30 KHz channel spacing necessary for the VHF band.

An 82S101 FPLA programmed with different settings of the programmable counter fixes the channel frequencies, and is addressed by a keyboard that selects a channel by pressing one of its keys. Part of the binary output of this FPLA is used to set the programmable counter in the HCTR0320 chip. A second 82S101 FPLA is used as an optional binary to BCD converter for addressing the BCD inputs of the programmable counter in the Frequency Synthesizer chip.

An interface is placed between the keyboard and the FPLA Channel Memory, to provide contact debounce and for changing transmit and receive mode. This permits the use of one crystal for generating both transmitting frequencies and

FREQUENCY SYNTHESIZED MOBILE FM-VHF TRANCEIVER
USING DIGITAL MEMORY FOR QUICK CHANNEL SELECTION

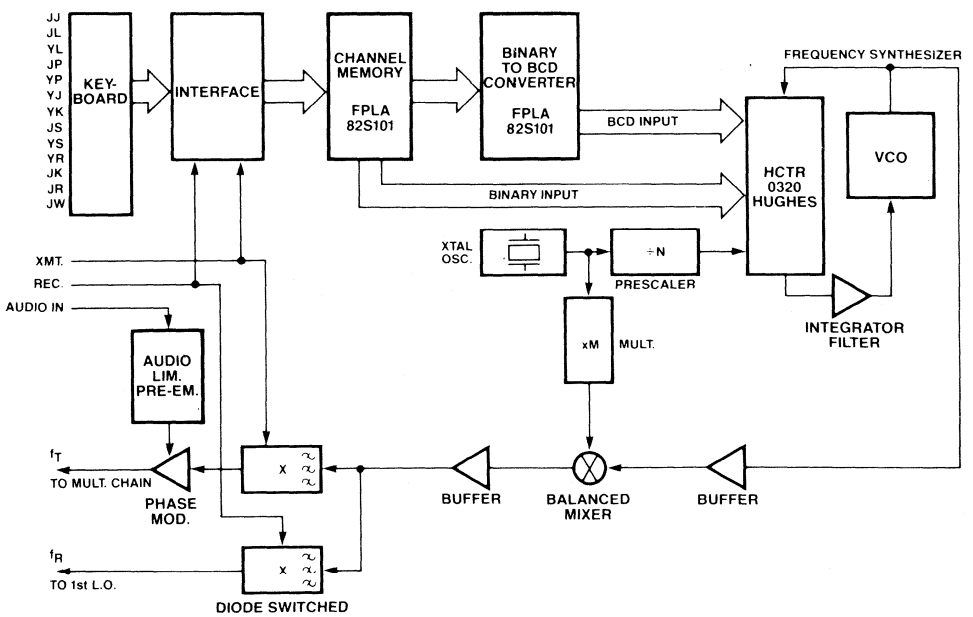


Figure 1.

the local oscillator frequencies for the receiver mixer. This can be done by mixing the signal from the crystal oscillator reference, previously multiplied, with the selected channel frequency from the synthesizer. A balanced mixer with diode switched bandpass filters, one for transmit and the other for receive is used. These are selected by a PTT switch or other suitable means.

In the transmit mode, the synthesized channel frequency goes thru a phase modulator that receives the audio signal from a limiter/pre-emphasis network. The output of the phase modulator is fed to a standard multiplier chain.

In the receive mode, the frequency generated f_R is used as a local oscillator for the receiver mixer ($f_R = F_T + f_{IF}$).

As the HCTR0320 can generate 1021 different frequencies/channels (in the programmable counter $3 \leqslant N \leqslant 1023$), it is possible to choose the Frequency Synthesizer output with its 13 channels centered in a convenient range as:

$$(3) \times 30\ KHz \leqslant f_{VCO} \leqslant 30\ KHz \times (1023)$$
$$90\ KHz \leqslant f_{VCO} \leqslant 30{,}690\ KHz$$

The choice of this center frequency is determined by the VCO available, quality of the mixer and the crystal frequency.

As noted, the second FPLA is optional, depending if the BCD inputs to the programmable counter are used or not, as this determines the high frequency operation of the Frequency Synthesizer. Alternately, these BCD inputs could be strapped such that the operating frequency of the VCO is around some desirable value.

The proposed Frequency Synthesizer with Channel Memory could form the basic building block for any two way radio telephone, as it has the definitive advantages of single crystal operation, and quick selection of the desired communication channel. The number of channels can be extended up to the limit set by the frequency synthesizer and the FPLA capacity.

The circuit of Figure 2 uses FPLA's in a programmable divider for a frequency synthesizer, as part of a 23 channel CB transceiver using a 10.7 MHz IF frequency. There are spare resources in the two FPLA's to go to the new 40 channels, once the frequencies of the additional channels are established by the FCC.

The block diagram of the frequency synthesizer is shown in Figure 1.

The scheme utilizes a divide by 20/21 prescaler such that the total divide ratio is N = (Px20) +Q. Table 1 gives the values of P and Q for both transmit and receive. FPLA #1 decodes when the counters count down to Q+1, and F_1 gives a HIGH output which resets the R-S NOR flip-flop causing the prescaler to divide by 21. When the counter reaches State 1, The F_0 output goes HIGH, providing a HIGH input for the D shift register which drives LOAD. On the next rising edge of the clock, LOAD goes LOW and presets the counter to P_1. Also at this time the ÷21 control is switched back to a divide by 20 mode.

Both Channel Select and Divide Control functions are programmed in the FPLA's as shown in the Program Tables of Figure 3 and 4.

SYNTHESIZER BLOCK DIAGRAM

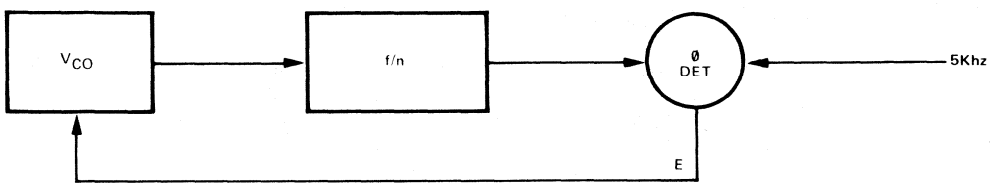


Figure 1.

PROGRAMMABLE FREQUENCY DIVIDER

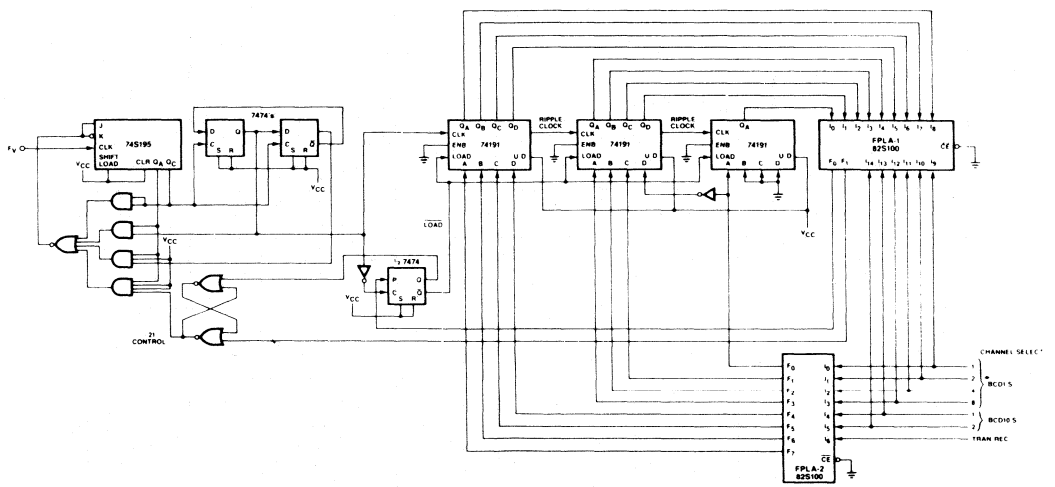


Figure 2.

P AND Q VALUES FOR TRANSMIT AND RECEIVE

Channel #	Frequency	Divide Ratio	P ÷ 20	Q ÷ 21	Frequency	Divide Ratio	P ÷ 20	Q ÷ 21
1	26.965	5393	269	13	16.265	3253	162	13
2	26.975	5395	269	15	16.275	3255	162	15
3	26.985	5397	269	17	16.285	3257	162	17
4	27.005	5401	270	1	16.305	3261	163	1
5	27.015	5403	270	3	16.315	3263	163	3
6	27.025	5405	270	5	16.325	3265	163	5
7	27.035	5407	270	7	16.335	3267	163	7
8	27.055	5411	270	11	16.355	3271	163	11
9	27.065	5413	270	13	16.365	3273	163	13
10	27.075	5415	270	15	16.375	3275	163	15
11	27.085	5417	270	17	16.385	3277	163	17
12	27.105	5421	271	1	16.405	3281	164	1
13	27.115	5423	271	3	16.415	3283	164	3
14	27.125	5425	271	5	16.425	3285	164	5
15	27.135	5427	271	7	16.435	3287	164	7
16	27.155	5431	271	11	16.455	3291	164	11
17	27.165	5433	271	13	16.465	3293	164	13
18	27.175	5435	271	15	16.475	3295	164	15
19	27.185	5432	271	17	16.485	3297	164	17
20	27.205	5441	272	1	16.505	3301	165	1
21	27.215	5443	272	3	16.515	3303	165	3
22	27.225	5445	272	5	16.525	3305	165	5
23	27.255	5451	272	11	16.555	3311	165	11

Table 1.

FPLA-2 PROGRAM TABLE FOR CHANNEL SELECT FUNCTIONS

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	L	L	L	L	L	L	H	•	•	A	•	•	•	A	•
1	–	–	–	–	–	–	–	–	–	L	L	L	L	H	H	–	•	•	A	•	•	•	A	•
2	–	–	–	–	–	–	–	–	–	L	L	L	L	H	–	–	•	•	A	•	•	•	A	A
3	–	–	–	–	–	–	–	–	–	L	L	L	H	L	L	–	•	•	A	•	•	•	A	A
4	–	–	–	–	–	–	–	–	–	L	L	H	L	L	L	–	•	•	A	•	•	•	A	A
5	–	–	–	–	–	–	–	–	–	L	L	H	L	L	H	–	•	•	A	•	•	A	•	•
6	–	–	–	–	–	–	–	–	–	L	L	H	L	H	–	–	•	•	A	•	•	A	•	•
7	–	–	–	–	–	–	–	–	–	L	L	H	H	L	L	–	•	•	A	•	•	A	•	•
8	–	–	–	–	–	–	–	–	–	L	H	L	L	L	–	–	•	•	A	•	•	A	•	A
9	–	–	–	–	–	–	–	–	–	H	L	L	L	L	L	H	A	•	•	•	A	A	•	A
10	–	–	–	–	–	–	–	–	–	H	L	L	L	L	H	–	A	•	•	•	A	A	•	A
11	–	–	–	–	–	–	–	–	–	H	L	L	L	H	–	–	A	•	•	•	A	A	A	•
12	–	–	–	–	–	–	–	–	–	H	L	L	H	L	L	–	A	•	•	•	A	A	A	•
13	–	–	–	–	–	–	–	–	–	H	L	H	L	L	L	–	A	•	•	•	A	A	A	•
14	–	–	–	–	–	–	–	–	–	H	L	H	L	L	H	–	A	•	•	•	A	A	A	A
15	–	–	–	–	–	–	–	–	–	H	L	H	L	H	–	–	A	•	•	•	A	A	A	A
16	–	–	–	–	–	–	–	–	–	H	L	H	H	L	L	–	A	•	•	•	A	A	A	A
17	–	–	–	–	–	–	–	–	–	H	H	L	L	L	–	–	A	•	•	A	•	•	•	•

Figure 3.

FPLA-1 PROGRAM TABLE FOR DIVIDE CONTROL FUNCTION.

SINCE F_{2-7} ARE UNUSED, ALL LINKS IN THAT AREA OF THE DEVICE HAVE BEEN LEFT INTACT

PRODUCT TERM																	ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	–	L	L	L	L	L	H	L	H	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
1	–	L	L	L	L	H	L	L	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
2	–	L	L	L	L	H	H	L	H	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
3	–	L	L	L	H	L	L	L	H	L	L	L	L	L	L	L	A	A	A	A	A	A	A	•
4	–	L	L	L	H	L	H	L	L	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
5	–	L	L	L	H	H	L	L	H	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
6	–	L	L	L	H	H	H	L	L	L	H	L	L	L	L	L	A	A	A	A	A	A	A	•
7	–	L	L	H	L	L	L	L	L	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
8	–	L	L	H	L	L	H	L	H	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
9	–	L	H	L	L	L	L	L	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
10	–	L	H	L	L	L	H	L	H	L	L	L	L	L	L	L	A	A	A	A	A	A	A	•
11	–	L	H	L	L	H	L	L	L	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
12	–	L	H	L	L	H	H	L	H	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
13	–	L	H	L	H	L	L	L	L	L	H	L	L	L	L	L	A	A	A	A	A	A	A	•
14	–	L	H	L	H	L	H	L	L	H	L	H	L	L	L	L	A	A	A	A	A	A	A	•
15	–	L	H	L	H	H	L	L	H	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
16	–	L	H	H	L	L	L	L	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
17	–	L	H	H	L	L	H	H	L	L	L	H	L	L	L	L	A	A	A	A	A	A	A	•
18	–	H	L	L	L	L	L	L	H	L	L	L	L	L	L	L	A	A	A	A	A	A	A	•
19	–	H	L	L	L	L	H	L	L	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
20	–	H	L	L	L	H	L	L	H	H	L	L	L	L	L	L	A	A	A	A	A	A	A	•
21	–	H	L	L	L	H	H	L	L	H	H	L	L	L	L	L	A	A	A	A	A	A	A	•
22	–	–	–	–	–	–	–	–	L	L	L	L	L	L	L	–	A	A	A	A	A	A	•	A

UNUSED

Figure 4.

PROGRAMMABLE FREQUENCY SYNTHESIZER 22

The circuit shown in Figure 1 is a keyer and/or V.C.A., as part of a synthesizer. The FPLA is used as a memory to store the shape and amplitude of the envelope for a given selection of the switches. The programmable timer operates in the astable mode with "C_X" and "R_X" determining the frequency of the multivibrator. The FPLA compares the state of the timer/counter with the selection of the switches. Its output goes to the appropriate transmission gate(s) which, in conjunction with the precision resistors, determine the gain of the Op-Amp and the final output frequency envelope. It should be noted that the resistors used must be *precision* or the envelope will lack uniformity. This circuit is very practical in forming waveforms that would be virtually impossible by analog means.

PROGRAMMABLE FREQUENCY SYNTHESIZER

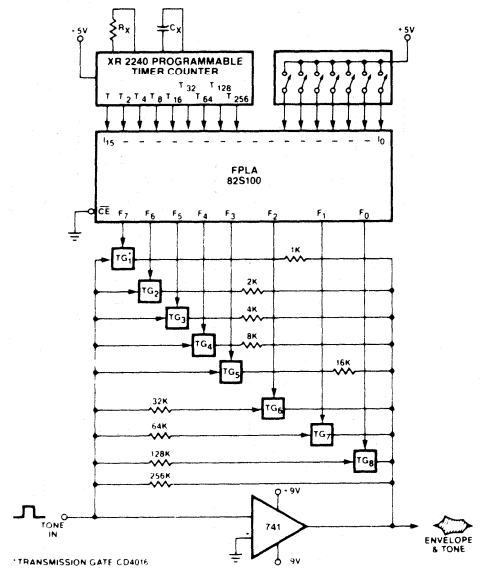


Figure 1.

AN IEEE-488 TO SPEECH SYNTHESIZER INTERFACE 23

The circuit shown in Figure 1 uses an FPLA for interfacing a Telesensory Systems Speech Synthesizer to an IEEE-488 standard instrument bus. This interface will allow blind electronic technicians to "hear" the digital output from digital test equipment equipped with an IEEE-488 interface.

The S15001-A Speech Synthesizer, described in Telesensory Systems Inc., Application Note 003, requires a Data Strobe to clock data on the bus, and it provides a Busy signal to complete the handshake. The inputs to the FPLA are the bus data lines D101 through D107, ATN, DAV and IFC. A power-on Reset input is provided for circuit initialization. Other inputs are the address comparator output, the Busy line from the speech board, the Listen feedback output of the FPLA and the Listen Only switch. The outputs of the FPLA are the bus handshake lines RFD and DAC, Listen, and Data Strobe. Circuit operation is as follows. The power on circuitry initializes the FPLA to a known state, unlisten or listen, depending on the Listen Always switch. Based on the interface bus signals, if the proper listen address is given and ATN is true, the handshake is completed and the Listen line goes true. When ATN is false and Listen is true, the interface accepts each successive byte of data with a Data Strobe signal to the speech board and then waits for the Busy signal from the speech board to go false to complete the bus handshake. The Unlisten command is implemented in FPLA programming; if the character "?" is sent on the data bus with ATN true, the feedback signal Listen goes false.

IEEE-488 STD SPEECH SYNTHESIZER INTERFACE

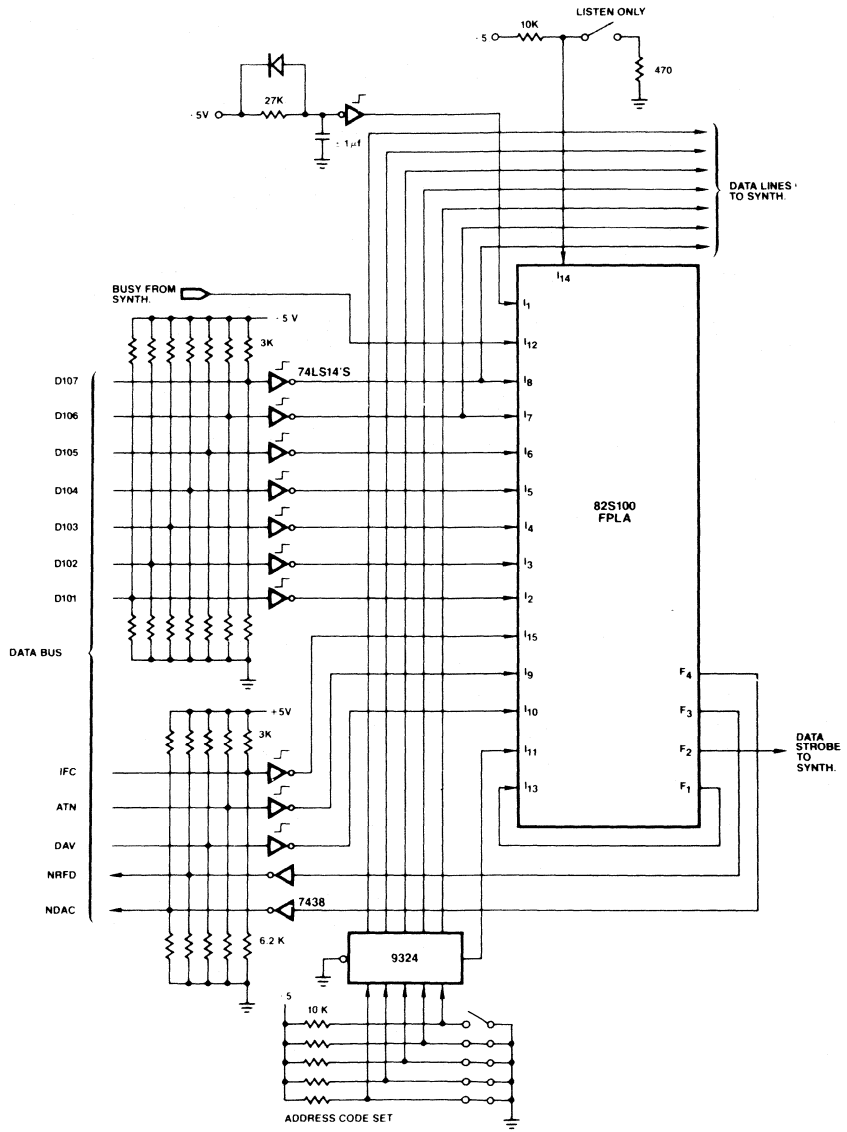


Figure 1.

The circuit shown in Figure 1 is a CAMAC Module for a dual channel, 12-bit A/D converter. CAMAC is an international standard for interfacing modular instrumentation to computers through a bused backplane called the Dataway. Within the context of this application, the Dataway contains the following signals:

SIGNAL	LINES	SIGNAL	LINES
Read Data	24	Station Number	24
Write Data	24	(control station address	
Function	5	to each station)	
Sub Address	4	Look-At-Me (LAM)	24
Status – Q, X	2	(control station interrupt)	
Control and Timing	3	to each station)	
(Z, S1, S2)		Power	±6V, ±24V

There are 32 possible functions and 15 sub addresses in each module. The CAMAC rules state that modules must completely decode all Function and Sub Address lines, and that the X status line must be asserted for every combination of Function and Sub Address implemented by the module. Q is a general purpose status line used primarily for testing conditions in modules. S1 and S2 are 200ns strobe signals which occur in sequence during a Dataway cycle. Z is a general reset function.

Decoding F and A, and generating Q, X, and LAM generally requires from 10 to 20 SSI and MSI packages. An FPLA can replace about 90% of this decoding logic, saving considerable design time and leaving more space for "functional" logic on the fixed size CAMAC printed circuit board.

The module in Figure 1 implements the functions tabulated in Figure 2.

Differing interpretations of the standards have resulted in some CAMAC users preferring A(15) for LAM functions while others prefer A(0). This example illustrates the power of the FPLA In that both options can be provided with no additional hardware.

CAMAC MODULE AD-212 FOR DUAL CHANNEL A/D CONVERTER

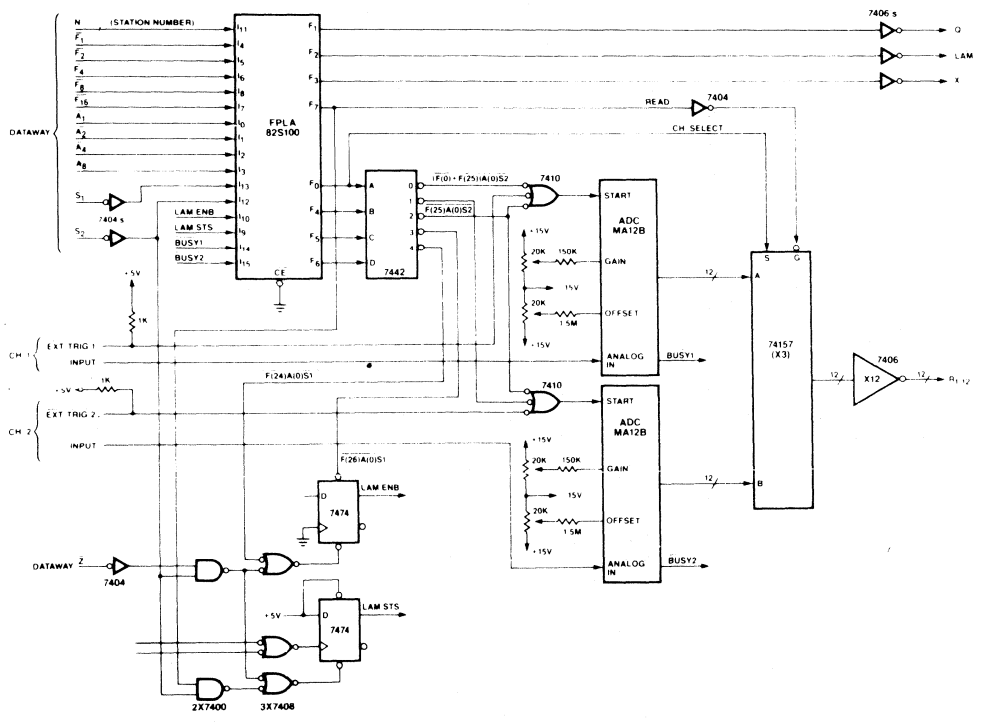


Figure 1.

FUNCTIONS EXECUTED BY CAMAC MODULE

CONDITION	FUNCTION	STATUS
F(0) A(0)	Read Channel 1	Q=$\overline{\text{BUSY 1}}$
F(0) A(1)	Read Channel 2	Q=$\overline{\text{BUSY 2}}$
*F(2) A(0)	Read Channel 1, start new conversion	Q=$\overline{\text{BUSY 1}}$
*F(2) A(1)	Read Channel 2, start new conversion	Q=$\overline{\text{BUSY 2}}$
F(8) A(0)	Test LAM	Q= LAM
F(8) A(15)	Test LAM	Q= LAM
F(24) A(0)	Rest LAM Enable	Q= 1
F(24) A(15)	Reset LAM Enable	Q= 1
*F(25) A(0)	Start Conversion in Channel 1	Q=$\overline{\text{BUSY 1}}$
*F(25) A(1)	Start Conversion in Channel 2	Q=$\overline{\text{BUSY 2}}$
*F(25) A(2)	Start Conversion in both channels	Q=$\overline{\text{BUSY 1 + BUSY 2}}$
F(26) A(0)	Set LAM Enable	Q=1
F(26) A(15)	Set LAM Enable	Q=1
F(27) A(0)	Test Channel 1 BUSY	Q=BUSY 1
F(27) A(1)	Test Channel 2 BUSY	Q=BUSY 2

*For F(2) and F(25) functions, conversion is not started if Q=0.

Figure 2.

PROGRAMMABLE WAVEFORM GENERATOR 25

An FPLA can provide an easy way of generating complex waveforms, whose shape can be quickly tailored to fit each application. In the circuit of Figure 1, a 16-stage binary counter drives the FPLA inputs which are programmed to detect specific counts. At each programmed count the FPLA supplies a corresponding binary weighted output which is summed thru the ladder network to set the desired amplifier gain.

The FPLA program table is derived from a staircase approximation of the desired output waveform, and duty cycle. The frequency is set by the clock. Each cycle is repeated by programming F_7 to produce a "1" output at end of cycle count. At the next positive transition of the clock all counter stages are reset, and the cycle repeats until stopped by the start switch.

PROGRAMMABLE WAVEFORM GENERATOR

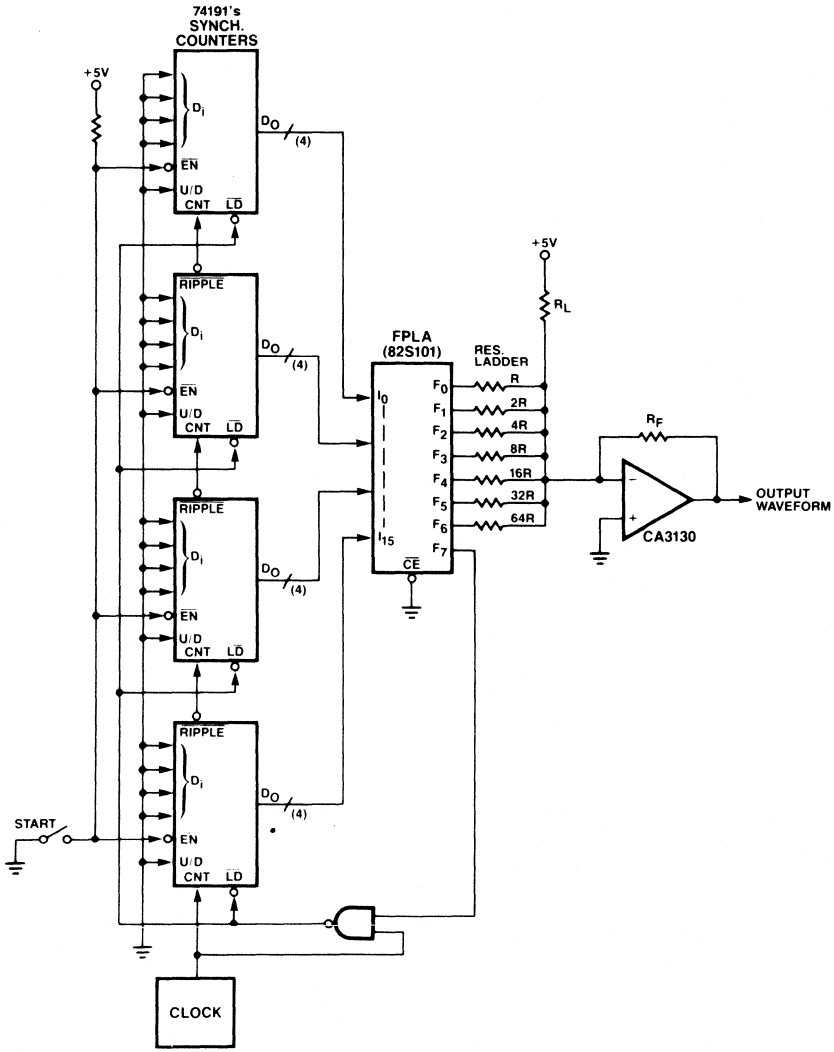


Figure 1.

The circuit shown in Figure 1 uses two FPLA's as the heart of a programmable function generator which can simulate any analog waveform by means of a piecewise linear approximation technique. A 555 timer serves as the clock generator for a 12-bit binary counter consisting of three 74LS163's, which establishes the time base of the generator. The FPLA's continually monitor the state of this counter to determine when the slope of the waveform should be changed and what its new value should be, as well as when to stop the clock, set the counter to zero, and discharge the output integrator. The value of the slope is retained in a pair of 74LS374 octal latches and fed into a 12-bit digital-to-analog converter (DA1201IC), whose output is then integrated by an LF356 op-amp to yield the analog output waveform as a continuous series of linear slope segments. Other support circuitry includes a means to encode up to 4 patterns in the same pair of FPLA's, and a provision to generate either one output cycle and stop, or to repeat continuously. It is interesting to note that the same features which make the FPLA's ideally suited for use in generating highly irregular analog waveforms also make them useful for generating some highly-repetitive waveforms, such as high-precision sine waves. Therefore, this circuit can also be very useful in applications requiring either a test generator or a waveform synthesizer.

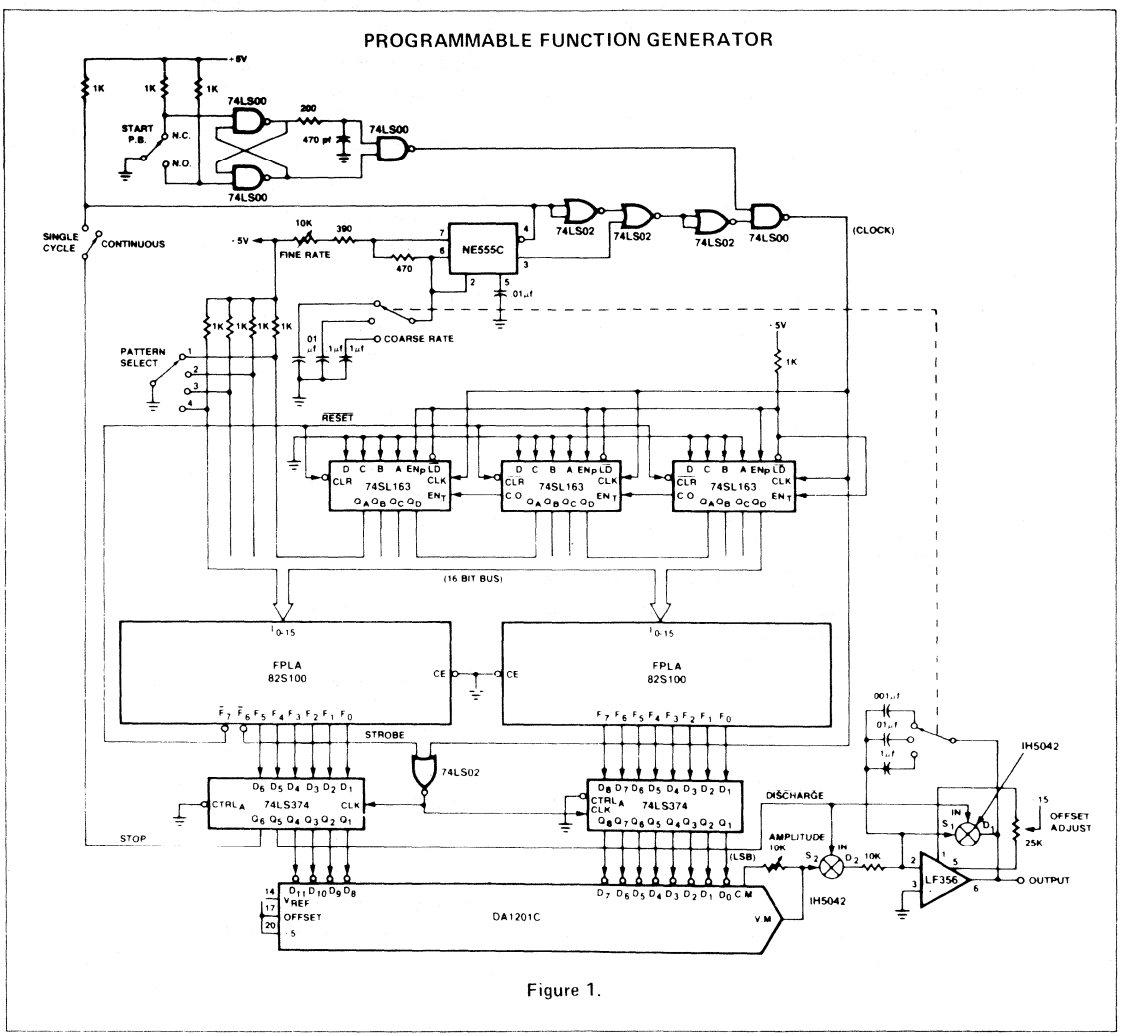


Figure 1.

The block diagram of Figure 1 shows an FPLA used to control the register select ports of 4-bit slice micrprocessors in a microprogrammed processor. Four bits for each port come from the microcode when a fifth bit is 0, or from various bits of the machine language instruction when the fifth bit is 1. Independent control of the A and B ports for 4 instruction formats is obtained with the following assignments.

FPLA INPUTS

$I_0 \rightarrow I_9$ = Pipeline Register (Control ROM) bits

Where:

$I_0 \rightarrow I_3$ = A Port select bits 0 thru 3
I_4 = A Port key bit
$I_5 \rightarrow I_8$ = B Port select bits 0 thru 3
I_9 = B Port Key bit

$I_{10} \rightarrow I_{15}$ = Instruction Register bits

Where:

I_{10} = IR Bit 3 I_{13} = IR Bit 6
I_{11} = IR Bit 4 I_{14} = IR Bit 13
I_{12} = IR Bit 5 I_{15} = IR Bit 14

FPLA OUTPUTS

$I_0 \rightarrow I_3$ = Drive A Port Bits 0 thru 3
$I_4 \rightarrow I_7$ = Drive B Port Bits 0 thru 3

Registers in the microprocessor slice register files are assigned to programmable functions in the machine language being implemented by microprogramming as follows:

MACHINE REGISTER	SLICE REGISTER
A Accumulator	0
B Accumulator	4
Index Reg. 1	1
Index Reg. 2	2
Index Reg. 3	3
Stack Pointer	5
Program Counter	6

Registers are selected by the FPLA when executing machine language instructions in the following formats:

1. **Single Word Memory Reference**
   Bits 13 and 14 call for indexed addresses as:
   00 = No Indexing 10 = Index Reg. 2
   01 = Index Reg. 1 11 = Index Reg. 3
2. **Double Word Memory Reference**
   Bits 4 and 5 call for indexed addressing, and are coded as above.
3. **General Register Operations**
   Bits 4, 5 and 6 indicate the register to be operated upon as:
   000 = Illegal 100 = A Accumulator
   001 = Index Reg. 1 101 = B Accumulator
   010 = Index Reg. 2 110 = Stack Pointer
   011 = Index Reg. 3 111 = Program Counter
4. **Double Register Operations**
   Bits 3 and 4 indicate one operand as:
   00 = A Accumulator 10 = Index Reg. 2
   10 = Index Reg. 1 11 = Index Reg. 3
   Bits 5 and 6 indicate the operand coded as above except:
   00 = B Accumulator.

The FPLA is used to decode and multiplex bits for each register port according to Table 1. This gives rise to the final Program Table for the FPLA shown in Figure 2.

BLOCK DIAGRAM OF REGISTER PORT SELECTOR

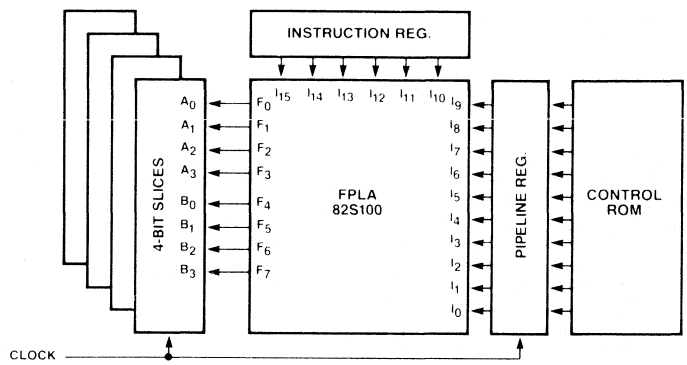


Figure 1.

CROM CODE SELECT ASSIGNMENT

Control ROM Bits 4	3	2	1	0	Register Port Bits 3	2	1	0	Function
0	x	x	x	x	CR3	CR2	CR1	CR0	Direct microprogram control
1	0	0	0	1	0	0	IR14	IR13	Single Word Memory Ref.
1	0	0	1	0	0	0	IR5	IR4	Double Word Memory Ref.
1	0	0	1	1	If IR6 = (0) 0	0	IR5	IR4	
					If code = (101) 0	1	0	0	General Register Operations
					If code = (110) 0	1	0	1	
					If code = (111) 0	1	1	0	
1	0	1	0	0	0	0	IR4	IR3	Two-reg. Operations First Operand
1	0	1	0	1	0	$\overline{(IR6 \cdot IR5)}$	IR6	IR5	Two-reg. Operations Second Operand

Table 1.

FPLA PROGRAM TABLE FOR REGISTER PORT SELECTOR

	PRODUCT TERM INPUT VARIABLE																ACTIVE LEVEL							
																	H	H	H	H	H	H	H	H
NO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	–	–	–	–	–	–	–	L	H	–	–	–	•	•	•	•	A	•	•	•
1	–	–	–	–	–	–	–	–	–	–	–	L	–	H	–	–	•	•	•	•	•	A	•	•
2	–	–	–	–	–	–	–	–	–	–	–	L	–	–	H	–	•	•	•	•	•	•	A	•
3	–	–	–	–	–	–	–	–	–	–	–	L	–	–	–	H	•	•	•	•	•	•	•	A
4	H	–	–	–	–	–	–	–	–	–	–	H	L	L	L	H	•	•	•	•	•	•	A	•
5	–	H	–	–	–	–	–	–	–	–	–	H	L	L	L	H	•	•	•	•	•	•	•	A
6	–	–	–	H	–	–	–	–	–	–	–	H	L	L	H	L	•	•	•	•	•	•	A	•
7	–	–	–	–	H	–	–	–	–	–	–	H	L	L	H	L	•	•	•	•	•	•	•	A
8	–	–	L	H	–	–	–	–	–	–	–	H	L	L	H	H	•	•	•	•	•	•	A	•
9	–	–	L	–	H	–	–	–	–	–	–	H	L	L	H	H	•	•	•	•	•	•	•	A
10	–	–	H	L	H	–	–	–	–	–	–	H	L	L	H	H	•	•	•	•	•	A	•	•
11	–	–	H	H	L	–	–	–	–	–	–	H	L	L	H	H	•	•	•	•	•	A	•	A
12	–	–	H	H	H	–	–	–	–	–	–	H	L	L	H	H	•	•	•	•	•	A	A	•
13	–	–	–	–	H	–	–	–	–	–	–	H	L	H	L	L	•	•	•	•	•	•	A	•
14	–	–	–	–	–	H	–	–	–	–	–	H	L	H	L	L	•	•	•	•	•	•	•	A
15	–	–	L	L	–	–	–	–	–	–	–	H	L	H	L	H	•	•	•	•	•	A	•	•
16	–	–	H	–	–	–	–	–	–	–	–	H	L	H	L	H	•	•	•	•	•	•	A	•
17	–	–	–	H	–	–	–	–	–	–	–	H	L	H	L	H	•	•	•	•	•	•	•	A
18	–	–	–	–	–	–	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	•
19	–	–	–	–	–	–	L	–	H	–	–	–	–	–	–	–	•	A	•	•	•	•	•	•
20	–	–	–	–	–	–	L	–	–	H	–	–	–	–	–	–	•	•	A	•	•	•	•	•
21	–	–	–	–	–	–	L	–	–	–	H	–	–	–	–	–	•	•	•	A	•	•	•	•
22	H	–	–	–	–	–	H	L	L	L	H	–	–	–	–	–	•	•	A	•	•	•	•	•
23	–	H	–	–	–	–	H	L	L	L	H	–	–	–	–	–	•	•	•	A	•	•	•	•
24	–	–	–	H	–	–	H	L	L	H	L	–	–	–	–	–	•	•	A	•	•	•	•	•
25	–	–	–	–	H	–	H	L	L	H	L	–	–	–	–	–	•	•	•	A	•	•	•	•
26	–	–	L	H	–	–	H	L	L	H	H	–	–	–	–	–	•	•	A	•	•	•	•	•
27	–	–	L	–	H	–	H	L	L	H	H	–	–	–	–	–	•	•	•	A	•	•	•	•
28	–	–	H	L	H	–	H	L	L	H	H	–	–	–	–	–	•	A	•	•	•	•	•	•
29	–	–	H	H	L	–	H	L	L	H	H	–	–	–	–	–	•	A	•	A	•	•	•	•
30	–	–	H	H	H	–	H	L	L	H	H	–	–	–	–	–	•	A	A	•	•	•	•	•
31	–	–	–	–	H	–	H	L	H	L	L	–	–	–	–	–	•	•	A	•	•	•	•	•
32	–	–	–	–	–	H	H	L	H	L	L	–	–	–	–	–	•	•	•	A	•	•	•	•
33	–	–	L	L	–	–	H	L	H	L	H	–	–	–	–	–	•	A	•	•	•	•	•	•
34	–	–	H	–	–	–	H	L	H	L	H	–	–	–	–	–	•	•	A	•	•	•	•	•
35	–	–	–	H	–	–	H	L	H	L	H	–	–	–	–	–	•	•	•	A	•	•	•	•

Figure 2.

FPLAs bring economy and flexibility to the design of the control unit in microprogrammed computers, by allowing complete freedom in allocating subroutines in microstore, and enhancing the use of variable formats in the op-code field from the Instruction Register.

The block diagram in Figure 1 shows a general MPCU organization. A key element in this structure is the op-code decoder for cracking each op-code from the IR into sub-routine-start addresses in control memory, and address mode definition.

To implement these functions for the MPCU in the PDP-11 minicomputer, three FPLAs are all that is required, as shown in the circuit in Figure 2. The first FPLA is used as Address Mode Interpreter, while FPLA-2 and FPLA-3 function as Instruction Interpreter. Translator function (op-code, source address mode, or destination address mode) is selected by control inputs A and B from memory, also defined in Figure

TYPICAL STRUCTURE OF MICROPROGRAMMED CONTROL UNIT

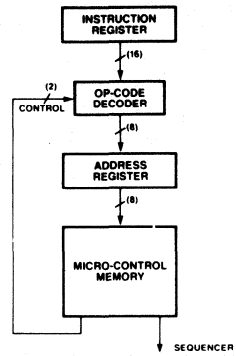
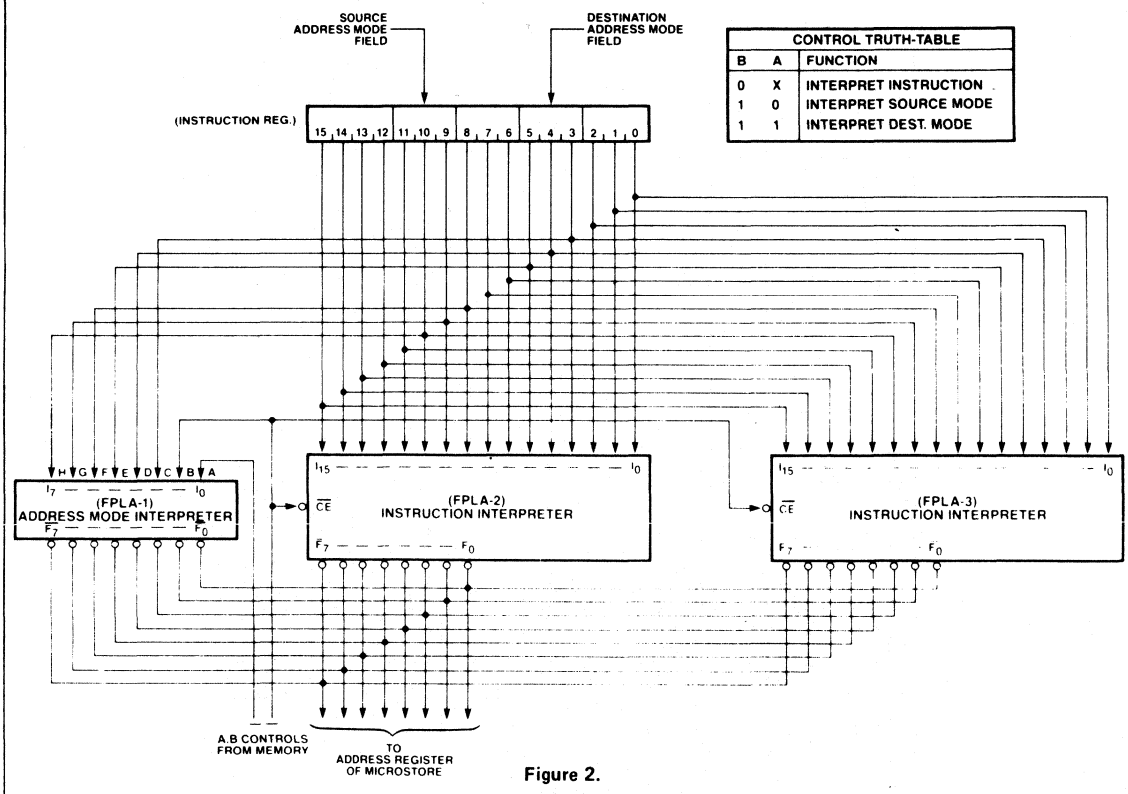


Figure 1.

OP-CODE DECODER NETWORK
ALL FPLAs ARE OPEN COLLECTOR (82S101). PULL-UP RESISTORS NOT SHOWN



CONTROL TRUTH-TABLE		
B	A	FUNCTION
0	X	INTERPRET INSTRUCTION
1	0	INTERPRET SOURCE MODE
1	1	INTERPRET DEST. MODE

Figure 2.

2. PDP-11 instructions have variable formats, from 4 to 16 bits. Their binary codes are tabulated in the input field of the Program Tables in Figures 3 and 4 for the Instruction Interpreter FPLAs. The output field of both FPLAs is programmed with a binary number corresponding to an arbitrary starting address in control memory. Bits 3 to 5 and 9 to 11 of the IR contain address mode information which is decoded by FPLA-1 as shown in the input field of the Program Table in Figure 5. The outputs of this FPLA are also assigned an arbitrary code pointing to a starting address in control memory. Note that all undefined instruction codes are not programmed, and will appear as an all I's address from the interpreter. This leaves 13 spare product terms in the Instruction Interpreter FPLAs to be used for editing, or future expansion.

A ROM version of this circuit would require 65K x 8 for the Instruction Interpreter and 128 x 8 for the Addressing Mode Interpreter. With 4K x 8 ROMs, this would require a total of 17 packages.

PROGRAM TABLE FOR INSTRUCTION INTERPRETER FPLA-2

INSTRUCTION	PRODUCT TERM NO.	INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HALT	0	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
WAIT	1	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H
RETURN FROM INTERRUPT	2	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L
BREAKPOINT TRAP	3	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	H
INPUT/OUTPUT TRAP	4	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L	L
RESET	5	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L	H
RETURN, INHIBIT TRACE TRAP	6	L	L	L	L	L	L	L	L	L	L	L	L	L	H	H	L
JUMP	7	L	L	L	L	L	L	L	L	L	H	–	–	–	–	–	–
RETURN FROM SUBROUTINE	8	L	L	L	L	L	L	L	L	H	L	L	L	L	–	–	–
RESERVED INSTRUCTION (8 CODES)	9	L	L	L	L	L	L	L	L	H	L	L	L	H	–	–	–
RESERVED INSTRUCTION (8 CODES)	10	L	L	L	L	L	L	L	L	H	L	L	H	L	–	–	–
NO OPERATION	11	L	L	L	L	L	L	L	L	H	L	H	L	L	L	L	L
CHANGE CODITION CODE (16 CODES)	12	L	L	L	L	L	L	L	L	H	L	H	–	H	–	–	–
CHANGE CONDITION CODE (16 CODES)	13	L	L	L	L	L	L	L	L	H	L	H	H	–	–	–	–
SWAP BYTES	14	L	L	L	L	L	L	L	L	H	H	–	–	–	–	–	–
UNCONDITIONAL BRANCH	15	L	L	L	L	L	L	L	H	–	–	–	–	–	–	–	–
BRANCH NOT EQUAL	16	L	L	L	L	L	L	H	L	–	–	–	–	–	–	–	–
BRANCH EQUAL	17	L	L	L	L	L	L	H	H	–	–	–	–	–	–	–	–
BRANCH GREATER OR EQUAL	18	L	L	L	L	L	H	L	L	–	–	–	–	–	–	–	–
BRANCH LESS THAN	19	L	L	L	L	L	H	L	H	–	–	–	–	–	–	–	–
BRANCH GREATER THAN	20	L	L	L	L	L	H	H	L	–	–	–	–	–	–	–	–
BRANCH LESS OR EQUAL	21	L	L	L	L	L	H	H	H	–	–	–	–	–	–	–	–
JUMP TO SUBROUTINE	22	L	L	L	L	H	L	L	–	–	–	–	–	–	–	–	–
CLEAR	23	L	L	L	L	H	L	H	L	L	L	–	–	–	–	–	–
COMPLEMENT	24	L	L	L	L	H	L	H	L	L	H	–	–	–	–	–	–
INCREMENT	25	L	L	L	L	H	L	H	L	H	L	–	–	–	–	–	–
DECREMENT	26	L	L	L	L	H	L	H	L	H	H	–	–	–	–	–	–
NEGATE	27	L	L	L	L	H	L	H	H	L	L	–	–	–	–	–	–
ADD CARRY	28	L	L	L	L	H	L	H	H	L	H	–	–	–	–	–	–
SUBTRACT CARRY	29	L	L	L	L	H	L	H	H	H	L	–	–	–	–	–	–
TEST	30	L	L	L	L	H	L	H	H	H	H	–	–	–	–	–	–
ROTATE RIGHT	31	L	L	L	L	H	H	L	L	L	L	–	–	–	–	–	–
ROTATE LEFT	32	L	L	L	L	H	H	L	L	L	H	–	–	–	–	–	–
ARITHMETIC SHIFT RIGHT	33	L	L	L	L	H	H	L	L	H	L	–	–	–	–	–	–
ARITHMETIC SHIFT LEFT	34	L	L	L	L	H	H	L	L	H	H	–	–	–	–	–	–
MARK	35	L	L	L	L	H	H	L	H	L	L	–	–	–	–	–	–
SIGN EXTEND	36	L	L	L	L	H	H	L	H	H	H	–	–	–	–	–	–
MOVE	37	L	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–
COMPARE	38	L	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT TEST	39	L	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–
BIT CLEAR	40	L	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT SET	41	L	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–
ADD	42	L	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–
MULTIPLY	43	L	H	H	H	L	L	L	–	–	–	–	–	–	–	–	–
DIVIDE	44	L	H	H	H	L	L	H	–	–	–	–	–	–	–	–	–
SHIFT ARITHMETIC	45	L	H	H	H	L	H	L	–	–	–	–	–	–	–	–	–
SHIFT ARITHMETIC COMBINED	46	L	H	H	H	L	H	H	–	–	–	–	–	–	–	–	–
EXCLUSIVE OR	47	L	H	H	H	H	L	L	–	–	–	–	–	–	–	–	–

ACTIVE LEVEL: L L L L L L L L

OUTPUT FUNCTION: 7 6 5 4 3 2 1 0

BINARY CODE OF START ADDRESS ASSIGNED IN MICROSTORE

Figure 3.

PROGRAM TABLE FOR INSTRUCTION INTERPRETER FPLA-3

INSTRUCTION	NO.	PRODUCT TERM INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLOATING ADD	0	L	H	H	H	H	L	H	L	L	L	L	L	–	–	–	–
FLOATING SUBTRACT	1	L	H	H	H	H	L	H	L	L	L	L	L	H	–	–	–
FLOATING MULTIPLY	2	L	H	H	H	H	L	H	L	L	L	L	H	L	–	–	–
FLOATING DIVIDE	3	L	H	H	H	H	L	H	L	L	L	L	H	H	–	–	–
SUBTRACT 1 & BRANCH	4	L	H	H	H	H	H	H	–	–	–	–	–	–	–	–	–
BRANCH PLUS	5	H	L	L	L	L	L	L	L	–	–	–	–	–	–	–	–
BRANCH MINUS	6	H	L	L	L	L	L	L	H	–	–	–	–	–	–	–	–
BRANCH HIGHER	7	H	L	L	L	L	L	H	L	–	–	–	–	–	–	–	–
BRANCH LOWER OR SAME	8	H	L	L	L	L	L	H	H	–	–	–	–	–	–	–	–
BRANCH OVERFLOW CLEAR	9	H	L	L	L	L	H	L	L	–	–	–	–	–	–	–	–
BRANCH OVERFLOW SET	10	H	L	L	L	L	H	L	H	–	–	–	–	–	–	–	–
BRANCH CARRY CLEAR	11	H	L	L	L	L	H	H	L	–	–	–	–	–	–	–	–
BRANCH CARRY SET	12	H	L	L	L	L	H	H	H	–	–	–	–	–	–	–	–
EMULATOR TRAP	13	H	L	L	L	H	L	L	L	–	–	–	–	–	–	–	–
TRAP	14	H	L	L	L	H	L	L	H	–	–	–	–	–	–	–	–
CLEAR BYTE	15	H	L	L	L	H	L	H	L	L	L	–	–	–	–	–	–
COMPLEMENT BYTE	16	H	L	L	L	H	L	H	L	L	H	–	–	–	–	–	–
INCREMENT BYTE	17	H	L	L	L	H	L	H	L	H	L	–	–	–	–	–	–
DECREMENT BYTE	18	H	L	L	L	H	L	H	L	H	H	–	–	–	–	–	–
NEGATE BYTE	19	H	L	L	L	H	L	H	H	L	L	–	–	–	–	–	–
ADD CARRY TO BYTE	20	H	L	L	L	H	L	H	H	L	H	–	–	–	–	–	–
SUBTRACT CARRY FROM BYTE	21	H	L	L	L	H	L	H	H	H	L	–	–	–	–	–	–
TEST BYTE	22	H	L	L	L	H	L	H	H	H	H	–	–	–	–	–	–
ROTATE RIGHT BYTE	23	H	L	L	L	H	H	L	L	L	L	–	–	–	–	–	–
ROTATE LEFT BYTE	24	H	L	L	L	H	H	L	L	L	H	–	–	–	–	–	–
ARITH SHIFT RIGHT BYTE	25	H	L	L	L	H	H	L	L	H	L	–	–	–	–	–	–
ARITH SHIFT LEFT BYTE	26	H	L	L	L	H	H	L	L	H	H	–	–	–	–	–	–
MOVE TO PGM STATUS	27	H	L	L	L	H	H	L	H	L	L	–	–	–	–	–	–
MOVE FROM PGM STATUS	28	H	L	L	L	H	H	L	H	H	H	–	–	–	–	–	–
MOVE BYTE	29	H	L	L	H	–	–	–	–	–	–	–	–	–	–	–	–
COMPARE BYTE	30	H	L	H	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT TEST BYTE	31	H	L	H	H	–	–	–	–	–	–	–	–	–	–	–	–
BIT CLEAR BYTE	32	H	H	L	L	–	–	–	–	–	–	–	–	–	–	–	–
BIT SET BYTE	33	H	H	L	H	–	–	–	–	–	–	–	–	–	–	–	–
SUBTRACT	34	H	H	H	L	–	–	–	–	–	–	–	–	–	–	–	–

ACTIVE LEVEL L L L L L L L L L L

OUTPUT FUNCTION 7 6 5 4 3 2 1 0

BINARY CODE OF START ADDRESS ASSIGNED IN MICROSTORE

Figure 4.

PROGRAM TABLE FOR ADDRESS MODE INTERPRETER FPLA-1
WHEN B = "0" ALL DEVICE OUTPUTS ARE FORCED HIGH, DELEGATING BUS CONTROL TO THE INSTRUCTION INTERPRETER FPLAs

INSTRUCTION	NO.	PRODUCT TERM INPUT VARIABLE 15	14	13	12	11	10	9	8	H 7	G 6	F 5	E 4	D 3	C 2	B 1	A 0
DESELECT ADDR MODE INTERP	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	L	–
SOURCE, REGISTER MODE	1	–	–	–	–	–	–	–	–	L	L	L	–	–	–	H	–
SOURCE, REGISTER DEFERRED	2	–	–	–	–	–	–	–	–	L	L	H	–	–	–	H	L
SOURCE, AUTO INCREMENT	3	–	–	–	–	–	–	–	–	L	H	L	–	–	–	H	L
SOURCE, AUTO INCR DEFERRED	4	–	–	–	–	–	–	–	–	L	H	H	–	–	–	H	L
SOURCE, AUTO DECREMENT	5	–	–	–	–	–	–	–	–	H	L	L	–	–	–	H	L
SOURCE, AUTO DECR DEFERRED	6	–	–	–	–	–	–	–	–	H	L	H	–	–	–	H	L
SOURCE, INDEX	7	–	–	–	–	–	–	–	–	H	H	L	–	–	–	H	L
SOURCE, INDEX DEFERRED	8	–	–	–	–	–	–	–	–	H	H	H	–	–	–	H	L
DESTINATION, REGISTER MODE	9	–	–	–	–	–	–	–	–	–	–	–	L	L	L	H	H
DESTINATION, REGISTER DEFERRED	10	–	–	–	–	–	–	–	–	–	–	–	L	L	H	H	H
DESTINATION, AUTO INCREMENT	11	–	–	–	–	–	–	–	–	–	–	–	L	H	L	H	H
DESTINATION, AUTO INCR DEFERRED	12	–	–	–	–	–	–	–	–	–	–	–	L	H	H	H	H
DESTINATION, AUTO DECREMENT	13	–	–	–	–	–	–	–	–	–	–	–	H	L	L	H	H
DESTINATION, AUTO DECR DEFERRED	14	–	–	–	–	–	–	–	–	–	–	–	H	L	H	H	H
DESTINATION, INDEX	15	–	–	–	–	–	–	–	–	–	–	–	H	H	L	H	H
DESTINATION, INDEX DEFERRED	16	–	–	–	–	–	–	–	–	–	–	–	H	H	H	H	H

UNUSED

ACTIVE LEVEL L L L L L L L L L L

OUTPUT FUNCTION 7 6 5 4 3 2 1 0

BINARY CODE OF START ADDRESS ASSIGNED IN MICROSTORE

Figure 5.

One of the most difficult tasks of a general purpose microprogrammed emulator shown in Figure 1 is the decoding of the target instruction. The circuit shown in Figure 2 provides a fairly general instruction decoder whose decoding can be changed to decode a variety of instruction sets by programming an FPLA. The equivalent circuit using

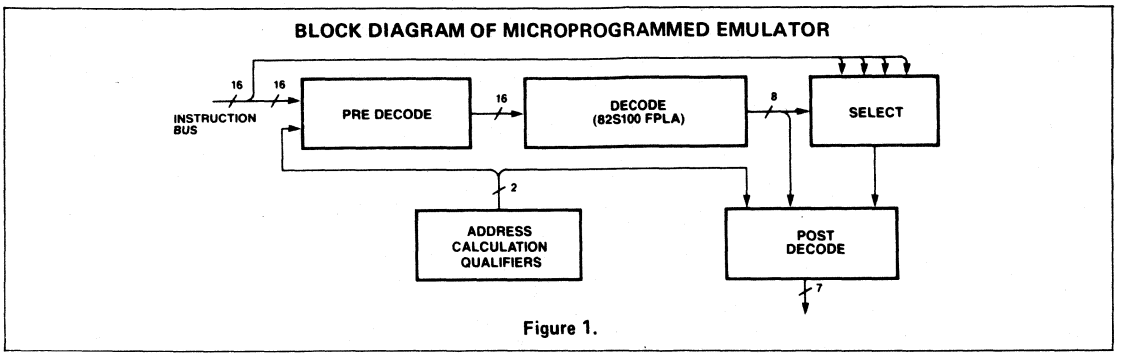


Figure 1.

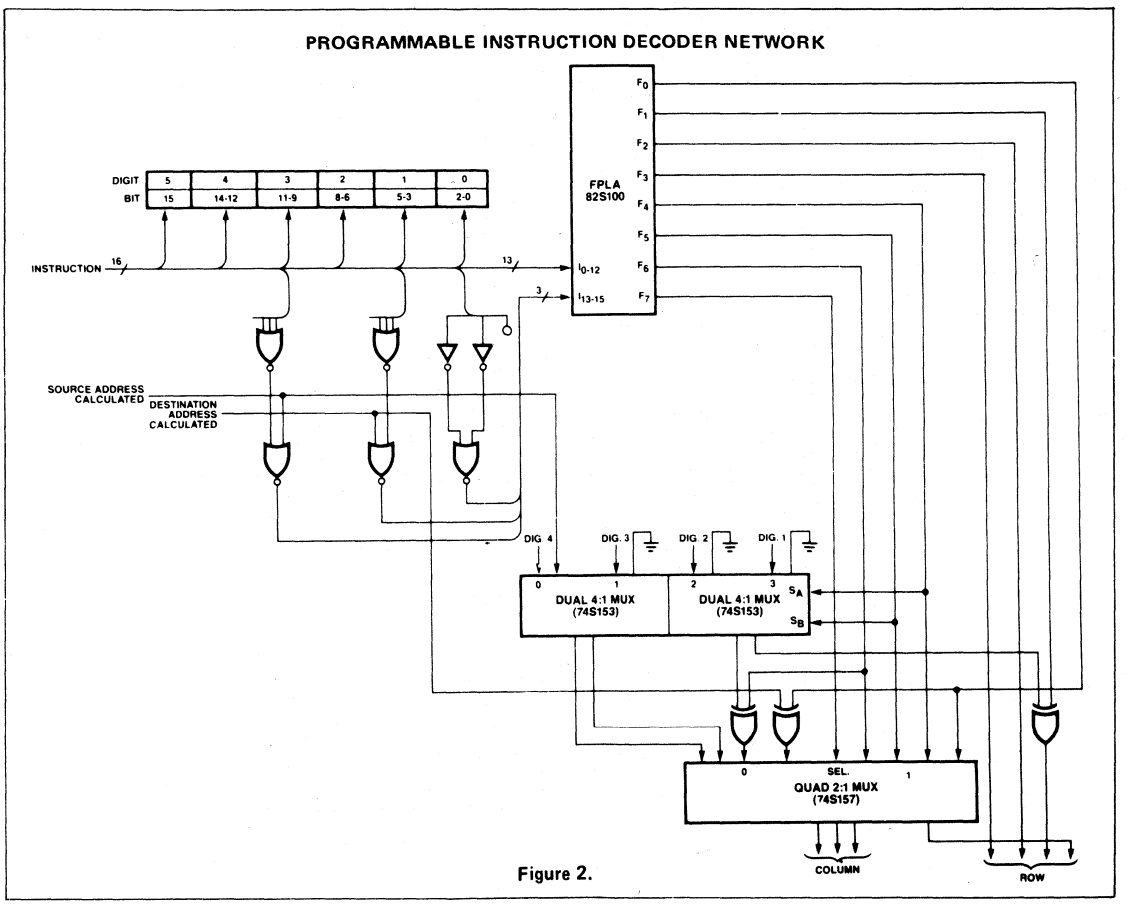


Figure 2.

PROMs would take over one hundred IC's.

The circuit has as input a sixteen bit instruction and two bits that tell whether address calculation has taken place. This is so that when the instruction is decoded and no memory reference is required, the decoder will generate the microinstruction address of the appropriate macroinstruction routine. If the instruction requires a memory reference, the address of the address calculation routine is generated, in which the address calculation bit is set. The microinstruction causes the decoder to be used again, but the second time through, the presence of the address calculation bit causes the decoder to generate the address of the appropriate microroutine, instead of the address calculation routine.

The FPLA in the circuit has been programmed to decode the DEC PDP-11 series of instructions. These are tabulated in the FPLA Program Table of Figure 3. Given a 16 bit opcode from a PDP-11, the circuit will return a microcode starting address. In addition, if a memory reference is required, the circuit will generate the address of the memory reference routine. The microcode can then set a status bit, so that during the second pass thru the network the memory reference routine is bypassed.

FPLA PROGRAM TABLE FOR PDP-11 DECODE

PRODUCT TERM																ACTIVE LEVEL								
	INPUT VARIABLE															H	H	H	H	H	H	H	H	
NO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION 7	6	5	4	3	2	1	0	COMMENTS
0	–	L	H	–	–	–	–	–	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	•	CALCULATE SOURCE ADDRESS, WORD
1	–	–	L	H	–	–	–	–	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	•	
2	–	H	–	L	–	–	–	–	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	•	
3	H	L	H	–	–	–	–	L	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	CALCULATE SOURCE ADDRESS, BYTE
4	H	L	H	–	–	–	–	–	L	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
5	H	H	L	–	–	–	–	L	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
6	H	H	L	–	–	–	–	–	L	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
7	H	–	L	H	–	–	–	L	–	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
8	H	–	L	H	–	–	–	–	L	–	–	–	–	H	–	–	•	•	•	A	•	A	A	A	
9	–	L	H	–	–	–	–	–	–	–	–	–	L	H	–	–	•	•	A	•	•	A	•	A	CALCULATE DESTINATION ADDRESS, WORD
10	–	–	L	H	–	–	–	–	–	–	–	–	L	H	–	–	•	•	A	•	•	A	•	•	
11	–	H	–	L	–	–	–	–	–	–	–	–	L	H	–	–	•	•	A	•	•	A	•	•	
12	L	L	L	L	L	L	L	L	–	H	–	–	–	H	–	–	•	•	A	•	•	A	•	•	
13	–	L	L	L	H	–	–	–	–	–	–	–	–	H	–	–	•	•	A	•	•	A	•	•	
14	L	H	H	H	L	–	–	–	–	–	–	–	–	H	–	–	•	•	A	•	•	A	•	•	
15	L	H	H	H	H	L	L	–	–	–	–	–	–	H	–	–	•	•	A	•	•	A	•	•	
16	H	L	H	–	–	–	–	–	–	–	–	–	L	H	H	–	•	•	A	•	•	A	•	•	CALCULATE DESTINATION ADDRESS, BYTE
17	H	H	L	–	–	–	–	–	–	–	–	–	L	H	H	–	•	•	A	•	•	A	•	A	
18	H	–	L	H	–	–	–	–	–	–	–	–	L	H	H	–	•	•	A	•	•	A	•	A	
19	H	L	L	L	H	–	–	–	–	–	–	–	–	H	H	–	•	•	A	•	•	A	•	A	
20	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	–	A	•	•	•	A	A	A	A	INTERRUPT INSTR.
21	L	L	L	L	L	L	L	L	–	–	–	–	–	–	L	–	•	A	A	A	A	A	A	•	SPEC/JMP/RTS/SWAP
22	L	L	L	L	L	L	L	L	H	L	L	L	H	–	–	–	A	A	A	A	A	A	A	A	RESERVED
23	L	L	L	L	L	L	L	L	H	L	L	H	L	–	–	–	A	A	A	A	A	A	A	A	
24	L	L	L	L	L	L	L	L	H	L	L	H	H	–	–	–	A	A	•	•	A	A	A	A	SPL
25	L	L	L	L	L	L	L	L	H	L	H	L	–	–	–	–	A	•	•	•	•	A	•	•	COND. CODE CLR
26	L	L	L	L	L	L	L	L	–	L	H	H	–	–	–	–	A	•	•	•	•	A	•	A	COND. CODE SET
27	L	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	A	A	•	BRANCH, TRUE COND
28	L	L	L	L	H	L	L	–	–	–	–	–	–	L	–	–	•	•	•	•	•	•	A	•	JSR
29	–	L	L	L	H	L	H	–	–	–	–	–	–	L	L	–	•	•	A	A	A	•	A	•	SINGLE OP., #1
30	–	L	L	L	H	H	L	–	–	–	–	–	–	L	L	–	•	•	A	A	A	•	•	•	SINGLE OP., #2
31	–	L	L	L	H	H	H	–	–	–	–	–	–	–	–	–	A	A	A	A	A	A	A	A	RESERVED
32	L	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	•	•	A	A	A	A	•	•	EIS
33	L	H	H	H	H	L	H	–	–	–	–	–	–	–	–	–	A	A	•	•	A	A	A	•	RESERVED
34	L	H	H	H	H	L	H	L	L	L	L	–	–	–	–	–	A	A	•	•	A	A	A	A	FLOATING POINT
35	L	H	H	H	H	H	L	–	–	–	–	–	–	–	–	–	A	A	•	A	A	A	A	A	RESERVED
36	L	H	H	H	H	H	H	–	–	–	–	–	–	–	–	–	A	A	A	A	A	A	A	A	SOB
37	H	L	L	L	L	–	–	–	–	–	–	–	–	–	–	–	A	A	A	A	A	A	A	•	BRANCH, FALSE COND.
38	H	L	L	L	H	L	L	L	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	•	EMT
39	H	L	L	L	H	L	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	A	TRP
40	H	L	L	L	H	H	L	H	L	L	–	–	–	L	L	–	•	A	A	A	A	•	•	•	MTPS
41	–	L	L	L	H	H	L	H	L	H	–	–	–	L	L	–	A	A	•	•	A	•	•	•	MT/F • I/D
42	–	L	L	L	H	H	L	H	H	L	–	–	–	L	L	–	A	A	A	•	A	•	•	•	
43	H	L	L	L	H	H	L	H	H	H	–	–	–	L	L	–	•	A	A	A	A	•	•	•	MKRS
44	H	H	H	L	–	–	–	–	–	–	–	–	–	L	L	–	•	A	•	•	•	•	A	•	SUBTRACT
45	H	H	H	H	–	–	–	–	–	–	–	–	–	–	–	–	A	A	A	•	A	A	A	A	EXT. FLOAT. PNT
46	L	L	H	L	H	H	L	H	L	L	–	–	–	–	–	–	A	A	•	A	A	A	A	•	MARK

Figure 3.

SINGLE OR BLOCK ADDRESS DECODING FOR M6800 μP 30

The 82S100 is ideally suited for decoding a 16 bit Address Bus into as many as 48 locations or blocks of locations. As shown in Figure 1, only 4¼ packages are required to generate 48 active-LOW chip enable or device select outputs.

FPLA outputs F4, F5, and F6 are programmed active-LOW. When an assigned address is on the bus, these outputs select one of the three 74154 Decoders, while outputs F0 through F3 functions as a 0-to-15 binary counter to index the decoder according to the programmed address. assignments. The "don't care" input programming feature allows block sizes of integral powers of 2 (as in memory chips) to be as easily assigned as singly decoded addresses.

The proper memory clock timing for an M6800 system is provided to the 74154 decoders by the $\phi 2 \cdot \overline{VMA}$ output of the 7400 Nand Gate.

This circuit would be most useful in a small microprocessor system having a lot of miscellaneous devices to be addressed.

ADDRESS DECODING IN MC6800 μP BASED SYSTEM

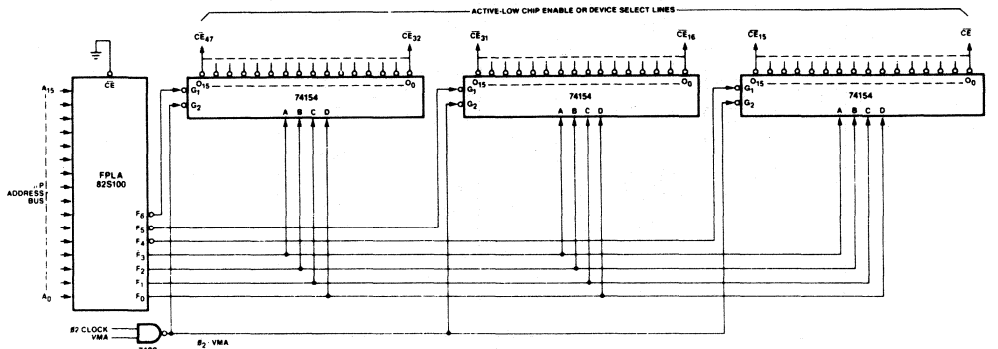


Figure 1.

INTERRUPT PRIORITY CONTROL FOR M6800 μP 31

The circuit in Figure 1 provides interrupt prioritization for a maximum of eleven devices, thereby eliminating the need for software polling techniques. The FPLA combines the functions of an eleven input "OR" gate, a quad Multiplexer, an eleven input Priority Encoder, and a 13x4 ROM. The circuitry adds a maximum of 125 ns to the settling time of the address bus.

Without prioritizing, all interrupt request inputs would be OR'ed together and applied to the $\overline{IRQ}$ input (pin 4) of the MPU. Upon receiving an interrupt request the microprocessor successively addresses FFF8 and FFF9. The 16 bits of data at these two addresses comprise the starting address for the interrupt routine. The MPU must then poll each device to determine who initiated the request and, in the case of multiple requests, must further determine which device will be serviced first. With the priority circuitry shown, all of the inputs are brought to the FPLA. Here they are OR'ed together at F_4 to produce the $\overline{IRQ}$ signal and where they are available for controlling F_0 through F_3 during the interrupt sequence. The other inputs are A_1 through A_4 from the MPU, and the output of the interrupt vector decode logic, for recognizing when the microprocessor outputs addresses FFF8 and FFF9. The function of the circuit is summarized in the truth-table of Figure 2. When there is no interrupt request the address bus is the same as the address outputted by the MPU. When any of the inputs goes LOW, F_4 goes LOW and the FPLA is ready to modify bits A_1 thru A_4 of the address bus with a specific starting address. These have been assigned to the device which initiated the interrupt in accordance with the vector table of Figure 3. This substitution occurs when I_4 goes HIGH. With proper programming the FPLA will automatically indicate the correct starting address when multiple requests are present. For example, if I_5 is to have the highest priority, the product matrix of I_6 through I_{15} contain I_5 as inhibit function. If I_6 is the next highest priority, all the product matrixes of I_7 through I_{15} would also contain I_6 as an inhibit function. The sum matrix is programmed according to the starting address assigned to the interrupt to be serviced. For example, if the device connected to I_7 is to have a starting address at FFE4 and FFF5, the product term for I_7 would be connected in the sum matrix to produce a (0010) at F0, F1, F2, and F3 respectively. The logic equation set to be programmed in the FPLA is tabulated in

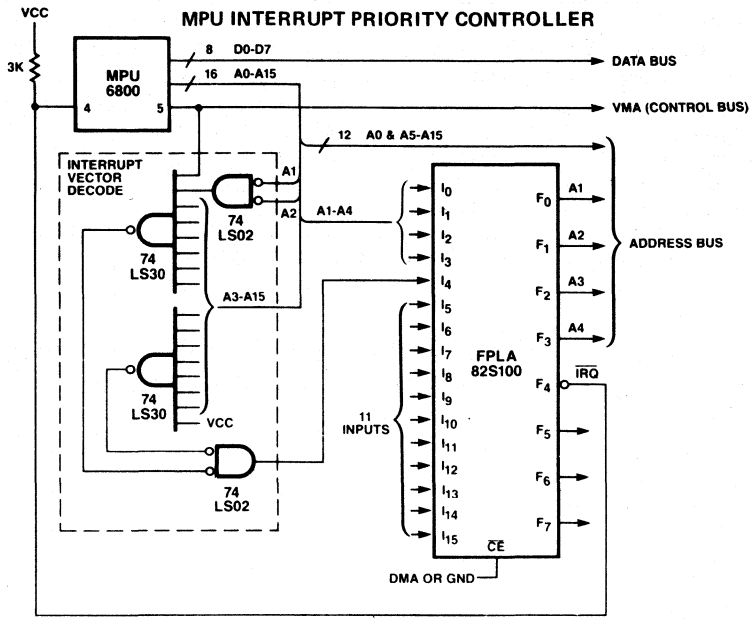


Figure 1.

Figure 4. The range of interrupt addresses is FFE0 through FFF9. The data at these locations can be either hardwired in, bootstrapped in during initialization, or firmware resident.

There are two extra address pairs and three spare outputs. These give the circuit a few additional capabilities not normally found with conventional priority circuits. The extra addresses can be used to designate special routines for certain combinations of multiple inputs. The extra outputs can be programmed to provide supplementary information. These can signal the operator, or MPU, of input combinations that are illegal or require immediate servicing. Outputs F5, F6, and F7 could be used to illuminate LEDs and/or be applied to microprocessor inputs such as $\overline{NMI}$, $\overline{HALT}$, or $\overline{RESET}$. If there is a need to place the address bus in the Hi-Z state, $\overline{CE}$ of the FPLA would be a function of, for example, a DMA controller.

The circuit is designed for use with the M6800 MPU family of components, but can be easily applied to other microprocessors.

CIRCUIT TRUTH-TABLE

I_5-I_{15}	I_4	$\overline{CE}$	$\overline{F4}$	ADD. BUS
X	X	H	HI-Z	HI-Z
H	L	L	H	A0 - A15
ANY INPUT(S) LOW	L	L	L	A0 - A15
	H	L	L	ADDRESS VECTOR FFE0 → FFF9

Figure 2.

VECTOR ASSIGNMENT

	ADDRESS BUS							
HEX	A15 - - - - - - -			A4	A3	A2	A1	A0
FFE0	1111	1111	111	0	0	0	0	0
FFE1	1111	1111	111	0	0	0	0	1
.	.	.	.			FPLA		.
.	.	.	.			OUTPUTS		.
.	.	.	.					.
FFF9	1111	1111	111	1	1	0	0	1

Figure 3.

LOGIC EQUATION SET OF FPLA

$F_0 = \overline{I_4} \cdot I_0 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_5)
$F_1 = \overline{I_4} \cdot I_1 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_{15})
$F_2 = \overline{I_4} \cdot I_2 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_{15})
$F_3 = \overline{I_4} \cdot I_3 + I_4 \cdot$ (FUNCTION OF I_5 THRU I_{15})
$F_4 = \overline{I_5} + \overline{I_6} + \ldots + \overline{I_{15}}$
$\overline{CE}$ = INPUT FROM DMA CONTROLLER OR GROUND

Figure 4.

UNIBUS ADDRESS MONITOR 32

The circuit in Figure 1 monitors how often a specific address or a specific group of addresses is referenced on the PDP-11 unibus. Up to eight groups can be monitored at the same time. The FPLA outputs are programmed to recognize a specific input address or group of addresses. When a particular address is referenced, the corresponding output fires a pulse to the corresponding counter and increments it. A switch selectable LED display and a frequency counter measurement test point is also provided. The circuit is very useful in measuring the effective data rate of peripherals.

UNIBUS ADDRESS MONITOR

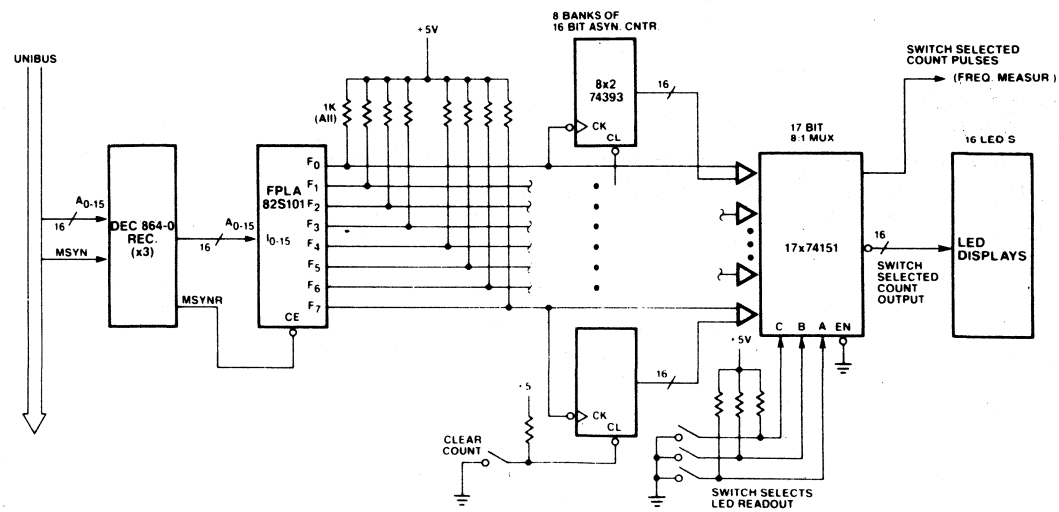


Figure 1.

The circuit shown in Figure 1 can be used for implementing hardware MACROS, or for calling conditional Diagnostics in a general processing system.

If $\overline{CE}$, EN1 and EN2 are set, and a 'keyword' instruction is fetched from memory, the FPLA is actuated to sequentially place new instructions in the input instruction stream. If possible the program counter is inhibited for subsequent instruction fetches, otherwise the last MACRO code resets the program counter. The next FPLA instruction is determined by decoding the last FPLA code (via 74174) until a 1's condition is set, which returns control to the memory bus. EN1 and EN2 allow conditional selection of different 'keyword' sets.

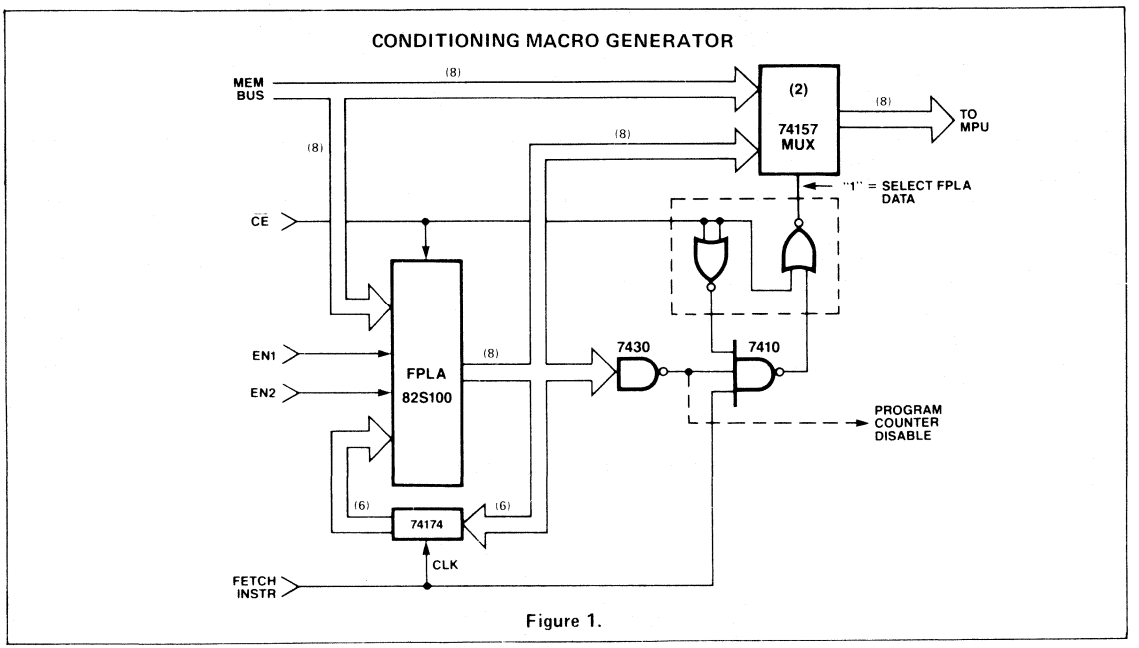


Figure 1.

FPLA ENHANCES PIPELINED PROCESS ARCHITECTURE 34

In a microcoded 16 bit processor the instruction fetch/decode/execute cycles can be overlapped in a pipelined fashion to increase execution speed. As shown in the block diagram of Figure 1, while instruction A is executing in firmware, instruction B is decoded by the decode FPLA to generate the next starting firmware address, and instruction C is fetched from memory into the pipeline register.

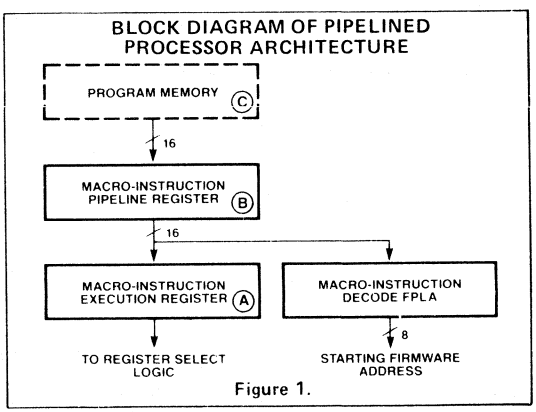


Figure 1.

The circuit in Figure 1 shows an FPLA as an 8-bit I/O port decoder for generating chip select signals to 32 PIA modules, type 6820, used in microprocessor designs. These could also be ACIA 6850, or other similar interface adapters.

Since the I/O address for 2 ports is fully decoded by one FPLA product term, the circuit can select 64 ports and a maximum of 96 when using all 48 product terms in one FPLA. For 16 or less ports, a single 1 of 16 decoder is sufficient.

Besides vastly reducing the number of decoding circuits, the FPLA avoids hardware limitations of software I/O addresses because it permits selection of more than one I/O port at a time, or one I/O port at more than one software address, or both. Also, since it allows to program some inputs as true "Don't Cares", a port can be assigned to be at an entire page or more of addresses.

Note that there remain two spare inputs to the FPLA. These can be used for hardware I/O control, such as automatic control for out of paper condition on printer.

I/O PORT DECODER

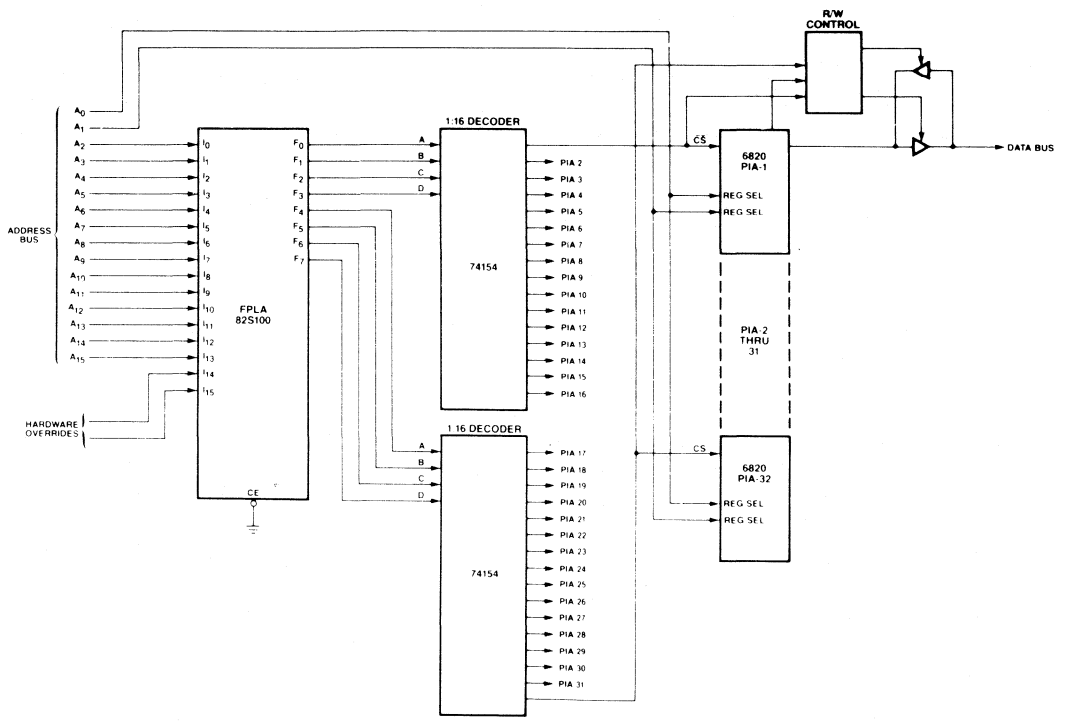


Figure 1.

PERIPHERAL ADDRESS DECODER 36

The circuit shown in Figure 1 is a simple but effective network in designs using microprocessors. The FPLA is used to fully decode the addresses of all peripheral circuits supporting the microprocessor system. If there are more than 8 peripherals, they can be easily accommodated by another FPLA. The outputs of the FPLA are connected directly to the chip select input of each device.

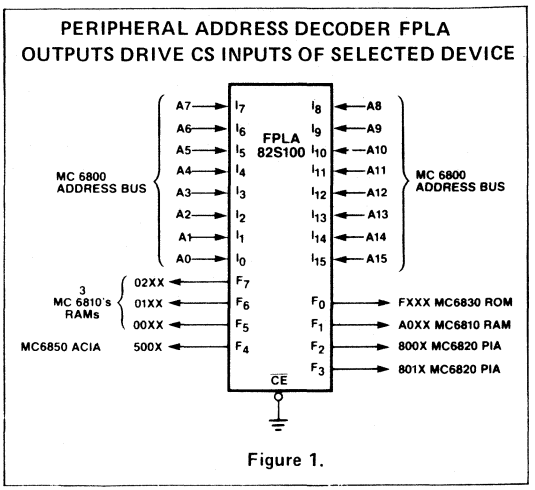


Figure 1.

MEMORY ADDRESS TRAP 37

The block diagram of Figure 1 shows an FPLA in conjunction with a small RAM for implementing a logically compact address trap, which can be readily modified. The FPLA will trap up to 48 locations in ROM and substitute a RAM location so that software subroutines may be easily called.

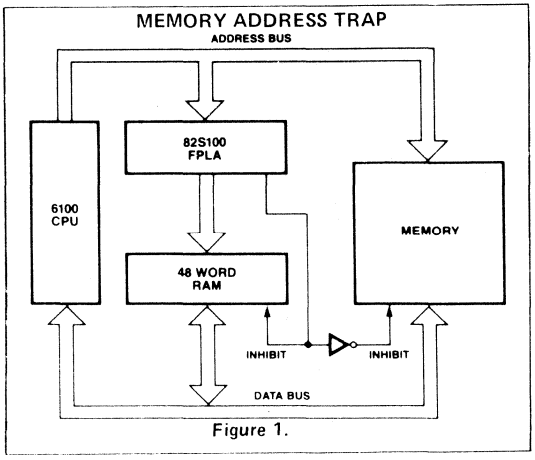


Figure 1.

MASKABLE INTERRUPT VECTOR GENERATOR 38

The circuit shown in Figure 1 generates maskable interrupt vectors for up to 8 interrupt sources. It also generates restart addresses and non-maskable interrupt addresses. Any of 8 possible interrupts sets the standard interrupt to the CPU by setting one of the interrupt flip-flops. When the CPU sends out the addresses which normally contains the vector address, the FPLA prioritizes the flip-flops so as to return the proper address. Each interrupt handler should clear its associated flip-flop via the shown output port. The device hold-off signal can be used to hold off the source of interrupts until each is serviced. The output addresses shown in the FPLA Program Table of Figure 2 represent a typical assignment.

INTERRUPT ADDRESS VECTOR GENERATOR FOR 6800 OR 650X MICROPROCESSORS

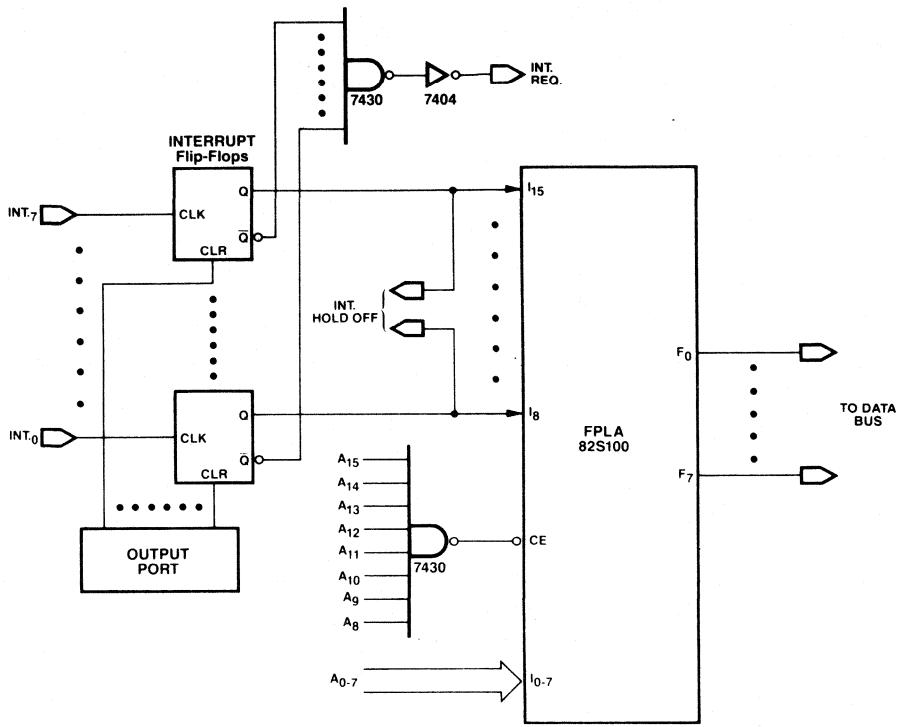


Figure 1.

TYPICAL FPLA PROGRAM TABLE

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: OUTPUT FUNCTION 7	6	5	4	3	2	1	0	COMMENTS
0	–	–	–	–	–	–	–	–	H	H	H	H	H	L	H	L	•	•	•	•	•	•	•	•	
1	–	–	–	–	–	–	–	–	H	H	H	H	H	L	H	H	•	•	•	•	•	•	A	•	address of non-maskable interrupt routine
2	–	–	–	–	–	–	–	–	H	H	H	H	H	H	L	L	•	•	•	•	•	•	•	•	
3	–	–	–	–	–	–	–	–	H	H	H	H	H	H	L	H	•	•	•	•	•	A	•	•	address of restart routine
4	–	–	–	–	–	–	–	H	H	H	H	H	H	H	H	L	•	•	•	•	•	•	•	•	
5	–	–	–	–	–	–	–	H	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	address of highest priority (INT_0) interrupt handler
6	–	–	–	–	–	–	H	L	H	H	H	H	H	H	H	L	•	•	A	•	•	•	•	•	
7	–	–	–	–	–	–	H	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	address of second highest priority (INT_1) interrupt handler
8	–	–	–	–	–	H	L	L	H	H	H	H	H	H	H	L	•	A	•	•	•	•	•	•	
9	–	–	–	–	–	H	L	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	address of third highest priority (INT_2) interrupt handler
16	–	H	L	L	L	L	L	L	H	H	H	H	H	H	H	L	A	A	•	•	•	•	•	•	
17	–	H	L	L	L	L	L	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	address of second lowest priority (INT_6) interrupt handler
18	H	L	L	L	L	L	L	L	H	H	H	H	H	H	H	L	A	A	A	•	•	•	•	•	
19	H	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	•	•	•	A	•	•	•	•	address of lowest priority (INT_7) interrupt handler

Figure 2.

15-INPUT PRIORITY ENCODER AND I.D. GENERATOR 39

In interfacing multiple devices to a mini or micro-computer bus, some sort of interrupt priority structure is generally needed. A conventional scheme uses a daisy chain circuit which allows a priority signal to pass if the device is not requesting an interrupt, and blocks it otherwise. In most computers the interrupting device is allowed to place its identifying code on the bus. If several devices share one P.C. board or chassis, one F.P.L.A. can be used to adjudicate priorities between the devices and generate the identifying codes, and thus save a large array of gates for each device serviced. The I/O assignment for the FPLA is shown in Figure 1. The FPLA program table is easily derived from the typical logic path in Figure 2, in which any arbitrary priority and device identification codes can be assigned.

FPLA IMPLEMENTS Nth ELEMENT OF DAISY-CHAIN PRIORITY INTERRUPT ENCODER

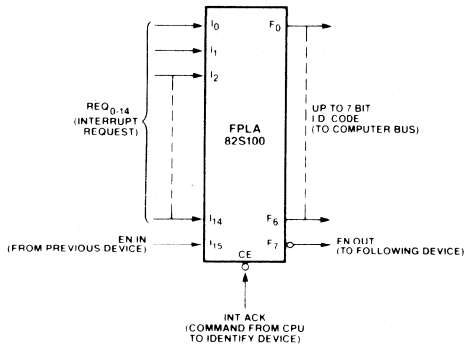


Figure 1.

TYPICAL LOGIC PATH WITHIN THE FPLA

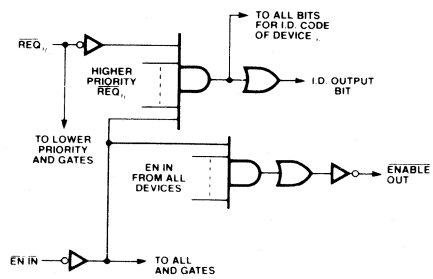


Figure 2.

FPLA SIMPLIFIES ALPHA-NUMERIC DISPLAY OPERATION 40

Controlling a cluster of 5 x 7 dot-matrix alpha-numeric displays is much more involved than numeric only, and generally requires a lot of circuitry which can be greatly reduced with FPLAs. The difficulty is that instead of displaying a whole character at once, as with 7-segment displays, multiplexing must occur on a column by column basis. Therefore the display must be fed information for part of each character, then illuminated, then the next part of each character and illuminated, etc.

The circuit shown in Figure 1 uses an FPLA to implement the necessary display control for any desired character-cluster lengths, obtained by simply reprogramming the FPLA. The circuit uses an H-P 5082-7150 display which requires all bits for column 1 of all characters to be serially clocked in. The display has a holding register, in this case 140 bits long (7 bits/column x 20 characters). When all 140 bits are clocked in, the serial clocking must stop and the column driver is turned on for the duration desired (1.5 ms). Then the column driver is turned off and the clocking begins for the next 140 bits (all of the column 2 bits) and column 2 is turned on.

Obviously all of the data for each column for all 20 characters must be readily available to this circuit, so a RAM was selected allowing external circuitry to load column data into the RAM. The circuit then takes this data and forms the 20 characters from it. In this case the RAM is loaded off of a transmission-line bus driven by a microprocessor. Placement of the data into the RAM is shown in Figure 2, and the microprocessor sends dot data to the RAM, not an ASCII character.

Close inspection of Figure 1 shows pin numbers and output bits mixed, but only to simplify PC layout. For instance, the RAM outputs feeding the 93L12 are mixed but not an error, so long as the RAM is being loaded with data in such a way as to form the character desired.

Since central control is afforded by the FPLA, the programming of the FPLA must be understood in order to know how the circuit works. With reference to Figure 3, note that all addresses are in octal. It takes 100 decimal memory locations to hold all of the date (5 cols/char x 20 chars). Assuming the RAM is loaded with the data in Figure 2, let's begin at address 0 set by the 93L66 counter, whose outputs also drive inputs to the 82S100 FPLA.

When the address is 0, column 5 is turned on by output F_7, but nothing will be displayed yet. After the 96L02 expires,

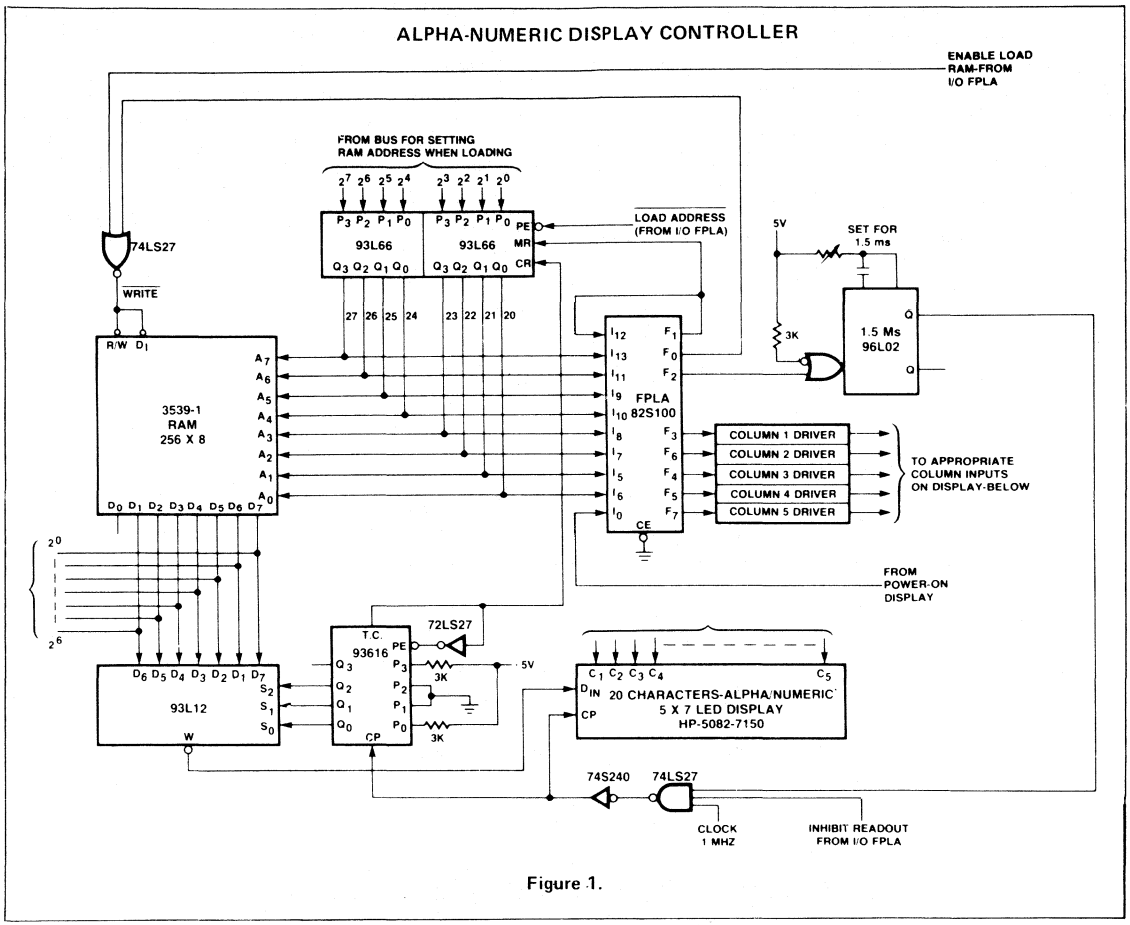


Figure 1.

data will begin to load into the display. Here's how it happens.

Since the address is 0, the RAM places the 7 column-1 bits for the character in address 0 on its output. The 93L16 is sitting at 001_2, thus the 93L12 is selecting a bit which appears at its W output, and is clocked into the display. At the next clock pulse, the 93L16 increments and the 93L12 selects the 2nd bit, which is subsequently clocked into the display. After 7 cycles the 93L16 hits top count, and the 93L66's are incremented causing address 1 to be referenced, which makes new data available to the 93L12. Top count also resets the 93L16 to octal 11, which is effectively 001_2. Note that the display and 93L16 use opposite trigger edges, so no race results. As the counter is incrementing nothing is happening at the 82S100, but as soon as address 24 is referenced the FPLA is activated. F_2 is enabled, which blocks the clock from further shifting-counting-selecting for the duration of 96L02 delay.

Simultaneously, F_3 enables the column-1 driver which enables all column-1 bits of each character. (Remember that the display has a holding register).

When the delay expires the clock is allowed to continue, data is shifted-counted-selected as before until the 93L66 reaches address 50, at which point the clock is again stopped and column 2 is enabled. This cycle is repeated until finally address 144 is reached, and the FPLA is again activated. F_1 is now enabled, resetting the address counter to zero so that the circuit recycles back to do all columns over again.

To avoid meaningless data from being displayed upon power-up, the FPLA can save additional circuitry by

supplying a CLR RAM signal. When power-on occurs, a timer enables FPLA input I_0 which is programmed to unconditionally reset the 93L66 counters to address zero, and enable the RAM to write. Now the clock is not inhibited so the counter begins counting (after the power-on I_0 goes away). But since F_1 is tied to I_{12} the counter addresses do not enable column drivers. Also, while counting, the RAM is writing zeros into itself, thus the RAM is being cleared at every address. Nothing else happens until the counter reaches 200, where the 82S100 is programmed to drop the F_0 output, and activate F_1. These will respectively disable RAM write and reset the 93L66 counter. This time at address 0, normal operation resumes because I_0 and I_{12} are not present.

The FPLA can also supply the additional function of clearing the RAM intentionally of its data. All that is required is to cause the 93L66 to be loaded with address 177, which the FPLA uses to force the circuit into the same clear-memory loop as the power-on did by enabling F_0 and F_1. This saves the trouble of loading each RAM address individually with zeros just to clear the display.

RAM MEMORY MAP

RAM ADR	Data Used For
00	Column 1 of character 1
↓	↓ ↓
23	Column 1 of character 20
24	Column 2 of character 1
↓	↓ ↓
47	Column 2 of character 20
50	Column 3 of character 1
↓	↓ ↓
73	Column 3 of character 20
74	Column 4 of character 1
↓	↓ ↓
117	Column 4 of character 20
120	Column 5 of character 1
↓	↓ ↓
143	Column 5 of character 20

CHARACTER NUMBERING OF ALPHA-NUMERIC DISPLAY

Figure 2.

PROGRAM TABLE OF DISPLAY CONTROL FPLA

RAM ADDRESS ASSIGNMENT IS IN OCTAL, WHERE:
177 = MASTER CLEAR ADDRESS
XXX = POWER-ON OR CLEAR-MEMORY LOOP
200 = TERMINATE CLEAR-MEMORY LOOP

RAM ADDR$_8$	NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: L	L	L	L	L	H	H	H
																		OUTPUT FUNCTION: 7	6	5	4	3	2	1	0
000	0	–	–	L	L	L	L	L	L	L	L	L	–	–	–	–	L	A	•	•	•	•	A	•	•
024	1	–	–	L	L	L	H	L	L	H	L	L	–	–	–	–	L	•	•	•	•	A	A	•	•
050	2	–	–	L	L	L	L	H	H	L	L	L	–	–	–	–	L	•	A	•	•	•	A	•	•
074	3	–	–	L	L	L	H	H	H	H	L	L	–	–	–	–	L	•	•	•	A	•	A	•	•
120	4	–	–	L	L	H	H	L	L	L	L	L	–	–	–	–	L	•	•	A	•	•	A	•	•
144	5	–	–	L	L	H	L	H	L	H	L	L	–	–	–	–	L	•	•	•	•	•	•	A	•
177	6	–	–	L	L	H	H	H	H	H	H	H	–	–	–	–	L	•	•	•	•	•	•	A	A
XXX	7	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	H	•	•	•	•	•	•	A	A
XXX	8	–	–	L	H	–	–	–	–	–	–	–	–	–	–	–	L	•	•	•	•	•	•	•	A
200	9	–	–	H	H	L	L	L	L	L	L	L	–	–	–	–	L	•	•	•	•	•	•	A	•
I/O ASSIGNMENT		UNUSED		2^7	F_1	2^6	2^4	2^5	2^3	2^2	2^0	2^1	UNUSED				Power-on	COL. 5	COL. 2	COL. 4	COL. 3	COL. 1	REFRESH	RESET CNTR	ENABLE RAM WE

Figure 3.

Any digital instrument can be difficult to read when its measurements are displayed rapidly to high arithmetic precision, and are moderately noisy to begin with. The programmable discrete filter provides an inexpensive solution to this problem by performing a numerical smoothing operation on the data, as it arrives from the measurement device in BCD form. It is especially well suited in design situations where the raw signal being measured is inherently digital, and cannot be conditioned using analog techniques, such as a digital phase meter for navigational applications.

The design uses a single FPLA as a bus controller, data decoder, and fixed program sequencer to control an inexpensive 4-function calculator with display. This approach was selected primarily for its simplicity and low cost. But the "post-production" editing capability of the FPLA provides another important advantage: three program constants K, K-1, and B defined below are to be determined and updated periodically during instrument calibration in the field. Since efficient microcoding of the fixed program has left over fifty percent of the FPLA memory available for these variable parameters, each unit can be recalibrated several times before replacing the FPLA. In addition, storing the instrument adjustments digitally within the FPLA should tend to discourage unauthorized modification.

The circuit mechanizes the following filter algorithm:

$$F_n = \frac{(K-1)F_{n-1} + (X_n - B)}{K}$$

where, in the present application,

F_n = the next filtered value to be displayed after processing the current measurement.

F_{n-1} = the last filtered value computed and displayed.

X_n = a five-digit BCD measurement.

B = a constant offset correction determined during instrument calibration (up to five digits in length).

K = a single-digit filter weight, which can be custom programmed to match the filter to its operation environment.

To avoid unnecessary transients at startup, the first filtered value (F_1) is initialized to the first available corrected data value (X_1–B) before beginning the smoothing process.

The following table demonstrates the dramatic improvement in display readability that is possible with the discrete filter operating on a noisy data stream in real time:

Sample Number (n)	Raw Data (X_n)	Unfiltered Data (X_n-B)	Filtered Data (F_n)
1	124.87	123.45	123.45
2	126.67	125.25	123.65
3	126.87	125.45	123.85
4	120.77	119.35	123.35
5	126.57	125.15	123.55
6	126.77	125.35	123.75
7	121.57	120.15	123.35
8	127.47	126.05	123.65
9	122.37	120.95	123.35
10	125.67	124.25	123.45

The filter weight K and offset constant B used in this example are 9 and 1.42, respectively. Larger values of K produce heavier filtering (and greater noise rejection), but as K increases, the filter responds less rapidly to legitimate changes in the signal being measured. In fact, the discrete filter behavior is analogous to that of an RC low-pass filter whose time constant is

$$T = \frac{\Delta t}{\ln\left[\frac{K}{K-1}\right]}$$

where Δt is the time between measurement samples.

A flowchart of the 16-step algorithm, block diagram of the filter/display, FPLA program table, and detailed circuit diagram appear in Figures 1 thru 5.

In the block diagram of Figure 2, when the measurement cycle is complete and a valid five-digit BCD word is present on DIGIT 1 – DIGIT 5, the measurement device (not shown) pulls $\overline{\text{START}}$ momentarily LOW. This begins the computation cycle by clearing the Program Counter.

$\overline{\text{DONE}}$ is a handshake return from the filter to the measurement device which remains HIGH until the computation is complete. DIGIT 1 – DIGIT 5 should not be permitted to change until $\overline{\text{DONE}}$ goes LOW.

As shown in Figure 4 the first cycle in each program step is subdivided into 2 half-cycles: a 'Fixed Program' half-cycle and a 'Decode' half-cycle. During the first half-cycle, a digit address is always set up and latched for the Digit Selector Network. If the current program step calls for a fixed operation, key closures are also simulated continuously during this half-cycle. (The on-chip key debounce feature of the calculator dictates that each key closure be simulated without interruption for at least 11.4 msec, followed by a pause of the same duration).

During the second half-cycle, the selected data is decoded, and key closures are simulated for the appropriate digit key.

If the current program step does not call for a BCD digit to be decoded, the FPLA addresses a unique sixth digit during the 'Fixed Program' half-cycle. This sixth digit is hardwired to a binary 15, which is ignored during the 'Decode' half-cycle since it is not a valid BCD code.

FILTER IN/OUT must be LOW for at least one computation cycle after the unit is turned on, in order to 'unlock' the calculator chip and initialize the filter. A panel switch can be used to control this signal; this would permit the operator to select either filtered or unfiltered data for display. (Note that the bias offset correction is applied in either mode).

An external system clock controls the program sequence. CLOCK should be approximately 40 Hz, 50% duty cycle, and can be free-running.

The FPLA steps the calculator through the program, by simulating multiplexed key closures in the sequence dictated by the Program Counter.

When the last operation is completed, the FPLA is disabled, $\overline{\text{DONE}}$ goes LOW, and the Program Counter remains locked in its final state. The calculator displays the results of the computation until the next $\overline{\text{START}}$ pulse arrives from the external measurement device. FILTER IN/OUT may be permitted to change state at any time during this pause for display.

Each program step, corresponding to a single state of the Program Counter, consists of two separate cycles of equal time duration. The FPLA is always enabled for the first cycle and disabled for the second.

16-STEP ALGORITHM ADAPTED TO POLISH FORM

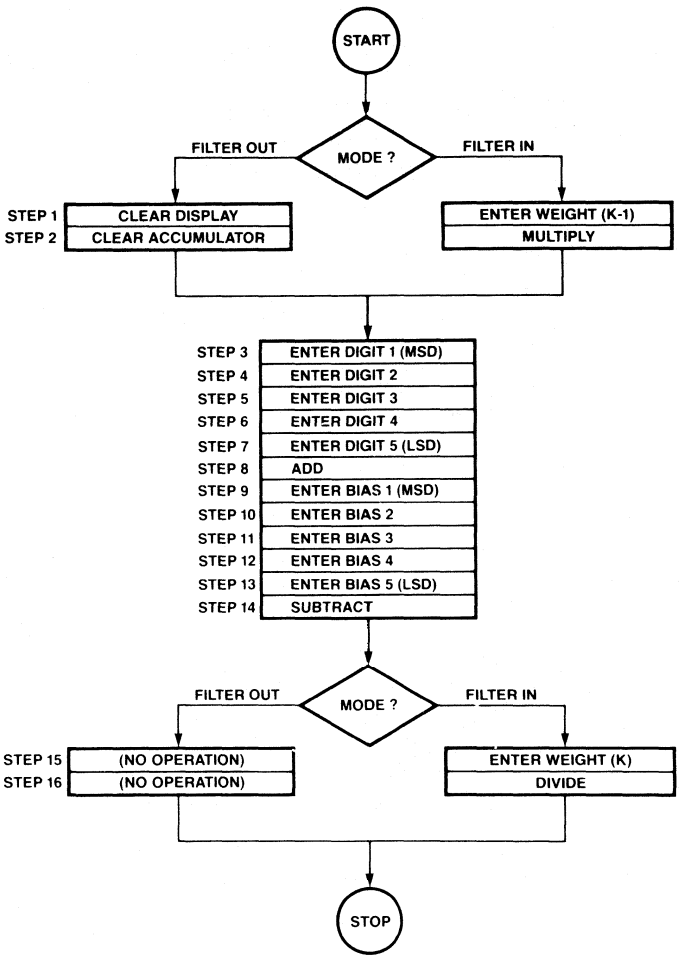


Figure 1.

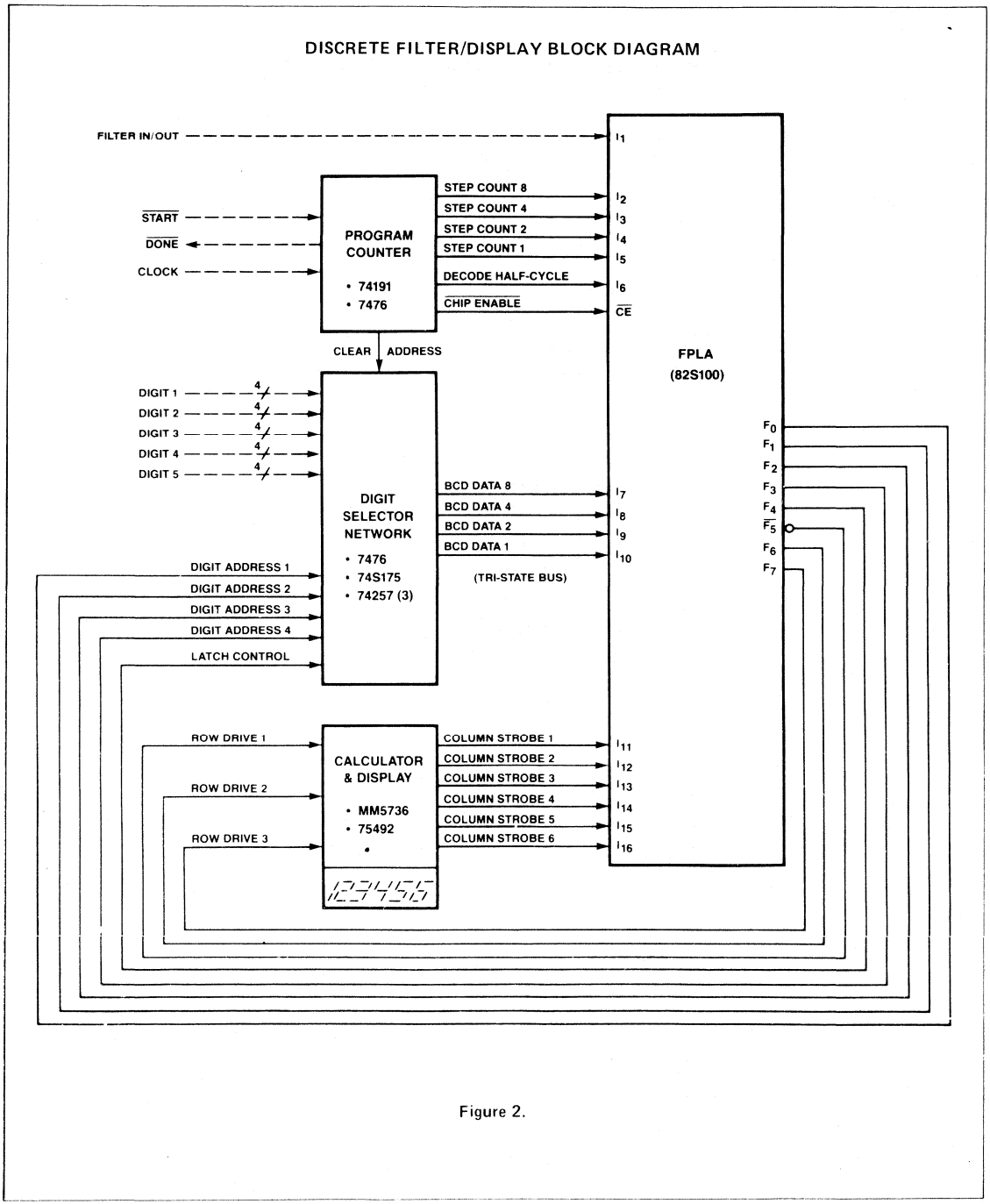

DISCRETE FILTER/DISPLAY BLOCK DIAGRAM
FILTER IN/OUT
I1
START
DONE
CLOCK
PROGRAM COUNTER
• 74191
• 7476
STEP COUNT 8
STEP COUNT 4
STEP COUNT 2
STEP COUNT 1
DECODE HALF-CYCLE
CHIP ENABLE
I2
I3
I4
I5
I6
CE
CLEAR
ADDRESS
FPLA
(82S100)
DIGIT 1
DIGIT 2
DIGIT 3
DIGIT 4
DIGIT 5
4
DIGIT SELECTOR NETWORK
• 7476
• 74S175
• 74257 (3)
BCD DATA 8
BCD DATA 4
BCD DATA 2
BCD DATA 1
(TRI-STATE BUS)
I7
I8
I9
I10
DIGIT ADDRESS 1
DIGIT ADDRESS 2
DIGIT ADDRESS 3
DIGIT ADDRESS 4
LATCH CONTROL
F0
F1
F2
F3
F4
F5
F6
F7
ROW DRIVE 1
ROW DRIVE 2
ROW DRIVE 3
CALCULATOR & DISPLAY
• MM5736
• 75492
COLUMN STROBE 1
COLUMN STROBE 2
COLUMN STROBE 3
COLUMN STROBE 4
COLUMN STROBE 5
COLUMN STROBE 6
I11
I12
I13
I14
I15
I16

Figure 2.

FPLA PROGRAM TABLE
X VALUES IN THE TABLE ARE DETERMINED AT CALIBRATION.
P-TERMS 30 THRU 48 CAN BE USED FOR SUBSEQUENT FIELD CALIBRATION, IF NECESSARY, TO REPLACE P_3 AND P_{11} THRU P_{17}.

FUNCTION	NO.	PRODUCT TERM / INPUT VARIABLE																ACTIVE LEVEL							
		I 1	I 2	I 3	I 4	I 5	I 6	I 7	I 8	I 9	I 10	I 11	I 12	I 13	I 14	I 15	I 16	H	H	H	H	L	H	H	H
																		OUTPUT FUNCTION							
																		1	2	3	4	5	6	7	8
Clear Display	1	L	L	L	L	L	L	–	–	–	–	L	H	H	H	H	H	•	•	A	•	A	•	•	A
Clear Accumulator	2	L	L	L	L	H	L	–	–	–	–	L	H	H	H	H	H	•	•	A	•	A	•	•	A
Enter Weight (K-1)	3	H	L	L	L	L	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Multiply	4	H	L	L	L	H	L	–	–	–	–	H	H	H	H	L	H	•	•	A	•	A	•	•	A
Address Digit 1 (MSD)	5	–	L	L	H	L	L	–	–	–	–	–	–	–	–	–	–	A	•	•	A	A	•	•	•
Address Digit 2	6	–	L	L	H	H	L	–	–	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•
Address Digit 3	7	–	L	H	L	L	L	–	–	–	–	–	–	–	–	–	–	•	A	•	A	A	•	•	•
Address Digit 4	8	–	L	H	L	H	L	–	–	–	–	–	–	–	–	–	–	•	A	•	•	A	•	•	•
Address Digit 5 (LSD)	9	–	L	H	H	L	L	–	–	–	–	–	–	–	–	–	–	•	•	A	A	A	•	•	•
Add	10	–	L	H	H	H	L	–	–	–	–	H	H	H	L	H	H	•	•	A	•	A	•	•	A
Enter Bias Digit 1 (MSD)	11	–	H	L	L	L	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Enter Bias Digit 2	12	–	H	L	L	H	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Enter Bias Digit 3	13	–	H	L	H	L	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Enter Bias Digit 4	14	–	H	L	H	H	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Enter Bias Digit 5 (LSD)	15	–	H	H	L	L	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Subtract (or add)	16	–	H	H	L	H	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	•	•	A
Enter Weight (K)	17	H	H	H	H	L	L	–	–	–	–	X	X	X	X	X	X	•	•	A	•	A	X	X	•
Divide	18	H	H	H	H	H	L	–	–	–	–	H	H	H	H	H	L	•	•	A	•	A	•	•	A
No Operation	18	L	H	H	H	H	L	–	–	–	–	–	–	–	–	–	–	•	•	A	•	A	•	•	•
Decode BCD '0'	20	–	–	–	–	–	H	L	L	L	L	L	H	H	H	H	H	•	•	•	•	•	A	•	•
Decode BCD '1'	21	–	–	–	–	–	H	L	L	L	H	H	L	H	H	H	H	•	•	•	•	•	A	•	•
Decode BCD '2'	22	–	–	–	–	–	H	L	L	H	L	H	H	L	H	H	H	•	•	•	•	•	A	•	•
Decode BCD '3'	23	–	–	–	–	–	H	L	L	H	H	H	H	H	L	H	H	•	•	•	•	•	A	•	•
Decode BCD '4'	24	–	–	–	–	–	H	L	H	L	L	H	H	H	H	L	H	•	•	•	•	•	A	•	•
Decode BCD '5'	25	–	–	–	–	–	H	L	H	L	H	H	H	H	H	H	L	•	•	•	•	•	A	•	•
Decode BCD '6'	26	–	–	–	–	–	H	L	H	H	L	H	L	H	H	H	H	•	•	•	•	•	•	A	•
Decode BCD '7'	27	–	–	–	–	–	H	L	H	H	H	H	H	L	H	H	H	•	•	•	•	•	•	A	•
Decode BCD '8'	28	–	–	–	–	–	H	H	L	L	L	H	H	H	L	H	H	•	•	•	•	•	•	A	•
Decode BCD '9'	29	–	–	–	–	–	H	H	L	L	H	H	H	H	H	L	H	•	•	•	•	•	•	A	•
I/O ASSIGNMENT		FILTER IN/OUT (1=IN)	STEP COUNT 8	STEP COUNT 4	STEP COUNT 2	STEP COUNT 1	DECODE HALF-CYCLE	BCD8	BCD4	BCD2	BCD 1	COLUMN STROBE 1	COLUMN STROBE 2	COLUMN STROBE 3	COLUMN STROBE 4	COLUMN STROBE 5	COLUMN STROBE 6	DIGIT ADDRESS 1	DIGIT ADDRESS 2	DIGIT ADDRESS 3	DIGIT ADDRESS 4	LATCH CONTROL	ROW DRIVE 1	ROW DRIVE 2	ROW DRIVE 3

Figure 3.

SYSTEM TIMING DIAGRAM.
CLOCK IS 40 HZ, 50% DUTY CYCLE

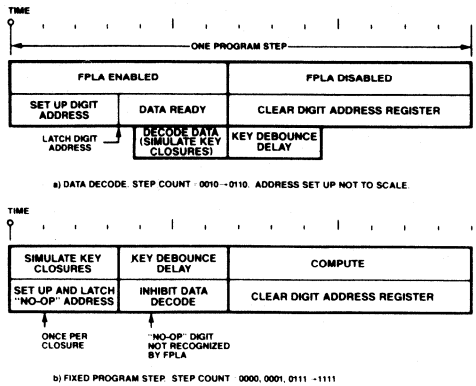


Figure 4.

CIRCUIT DIAGRAM OF DISPLAY CONDITIONER

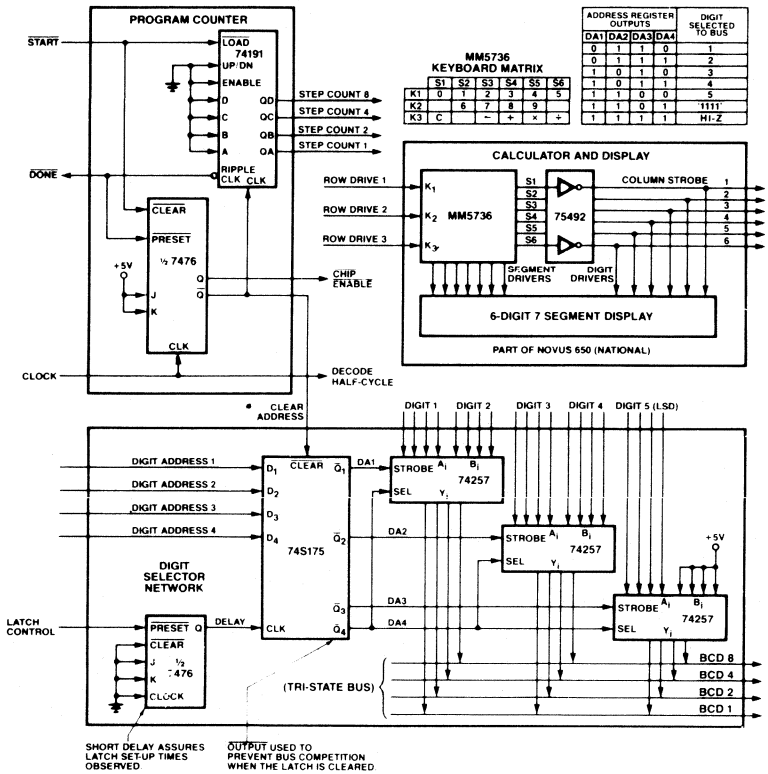


Figure 5.

There are numerous applications where a quick and accurate measurement of rate is desired. But in many of these cases, the events do not occur very often, and so they require a long time before a reading is obtained. Usually an analog low-pass filter is used to filter and derive the DC component of a monostable, triggered by the event. The voltage level is then directly proportional to the rate. However, in order to minimize errors due to the charging and discharging of the filter, long time constants are used, which adversely affect settling times. As illustrates in Figure 1, an accurate reading can only be made if the filter output is sampled when V(avg) equals V(charge) or V(discharge), as opposed to sampling at constant time intervals. Furthermore, if the proper sampling time as a function of input rate were known, one could compensate for filters with a faster rise-time, and thus obtain readings much sooner. This technique can be incorporated in a rate monitor system as shown in the circuit of Figure 2. The FPLA is used as a look-up table addressed by the counters after the monostable (MONO) goes LOW. The output counters are loaded every time a product term is activated, causing output F_7 to go HIGH. When an event occurs, the output counter has stored in it the proper sampling time (Ts), and will count down to zero during the monostable time. When it reaches zero, the A/D will be triggered.

SAMPLING FOR MAXIMUM RATE MEASUREMENT ACCURACY.
Tm, $\overline{Tm}$ ARE RESPECTIVELY THE ON AND OFF TIMES OF THE MONOSTABLE

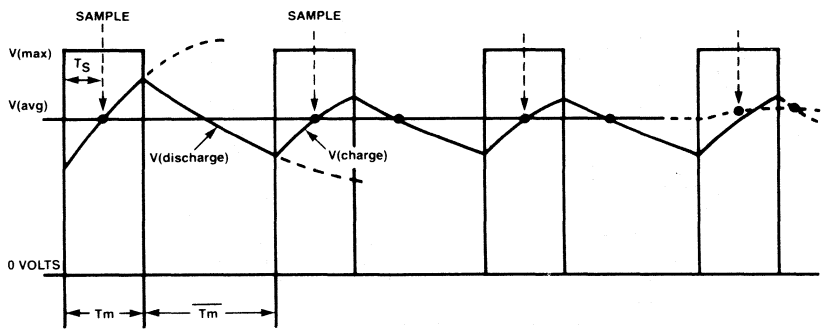


Figure 1.

The value of Ts varies with V(max), Tm and the RC time constant of the averager. It can be computed for various values of $\overline{Tm}$ as follows:

$$V_1 = V_2 \exp(-\overline{Tm}/RC) \qquad (1)$$

$$V_2 = Vmax + (V_1 - Vmax) \exp(-Tm/RC) \qquad (2)$$

V_2 can be found by substituting (2) into (1). Each corresponding value of Ts to be digitized in the FPLA is then obtained by solving:

$$Vavg = Vmax \frac{Tm}{Tm + \overline{Tm}} = V_2 \exp(-Ts/RC)$$

This circuit can be used in many industrial, medical and control applications. Using Heart Rate as a example, Tm = 150 msec, implying a maximum rate of 400BPM. Minimum heart rate is a function of the number of inputs to the FPLA, and the counter clock rate. Using the Signetics 82S100, and 1 kHz, the minimum H.R. = 1 BPM. Since one FPLA output must be used as clock enable only 7 outputs remain. These should be sufficient, for in this case Ts (max) = Tm/2 = 75. As a result the 48 product terms have to be used sparingly, but in most cases this provides more than adequate precision. We can see how the FPLA lends itself to this kind of application for, if we were to use a PROM, it would require a 16K x 8 configuration.

LOW FREQUENCY EVENT MONITOR

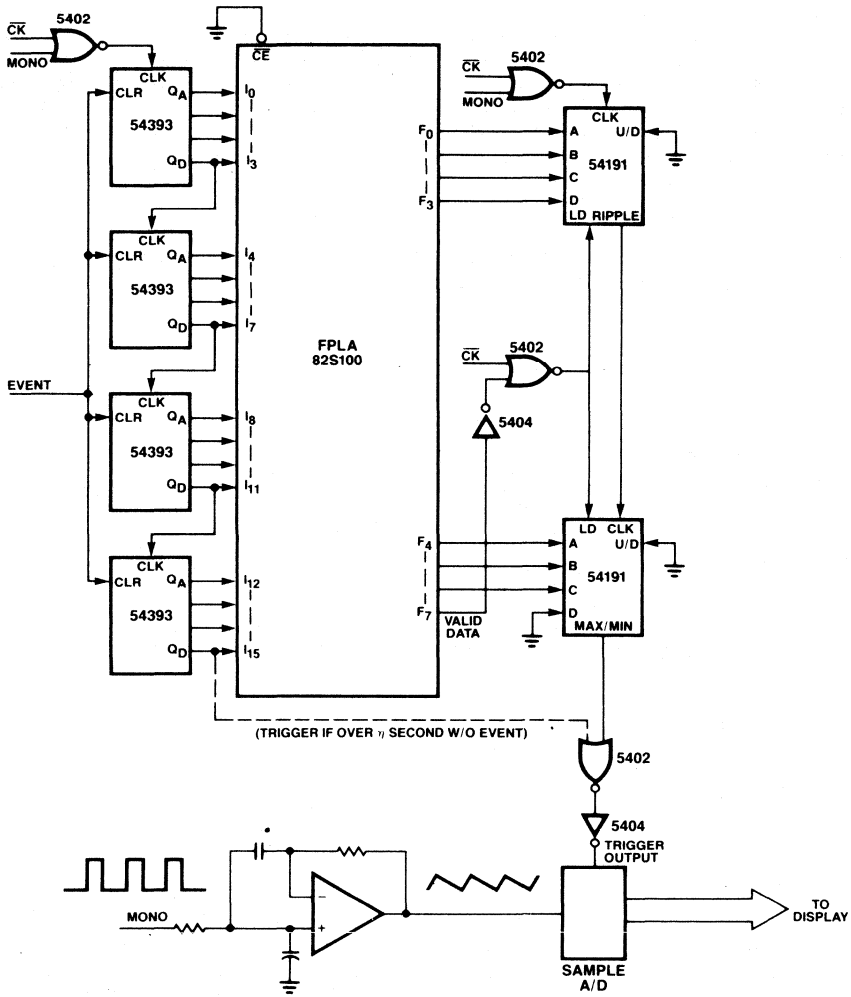


Figure 2.

The circuit shown in Figure 1 is a logic analyzer which can store 8 channels of data at rates up to 4 MHz and allow viewing of the data on any oscilloscope. A flip-flop type input circuit enables spikes and glitches far shorter than the clock to be detected. A trigger word detector allows the recording sequence to begin on any combination of input data.

The FPLA functions as the control element for the device. Switch closures, three clock phases (2 for recording and 1 fixed for playback), counter stages, and latches are fed into the FPLA which, in turn, generates Write enable for the memory, S-R information for the latches, and increment pulses for the counters.

There are two modes of operation: record and real time. In record, pressing the record switch arms the devices, so it can record when the trigger word appears. Recording stops depending on the position of the "time frame select" switch and a counter indicating the number of cycles since the start of recording. This way either 1/8, 1/2, or 7/8 of the total period may be displayed before the trigger word. After recording, the device goes into playback mode, except that the device is automatically re-armed after each playback cycle.

The input data first goes to a comparator which serves as an impedance buffer and a variable threshold detector for different logic families. The two latches gather data during alternate write cycles. With the D flip-flop and the EX-OR gates, they function as data change detectors, so only truly useful data is written in memory.

The playback cycle is carried out at 200 kHz; fast enough to avoid flicker, but slow enough for most scopes. The horizontal position is taken from the memory address, while the vertical position is generated from the data and the output multiplex address. The use of an FPLA allows easy modification of operating modes and timing. Since the FPLA is a Schottky device, the maximum clock frequency can be increased simply by using faster memory.

LOGIC ANALYZER

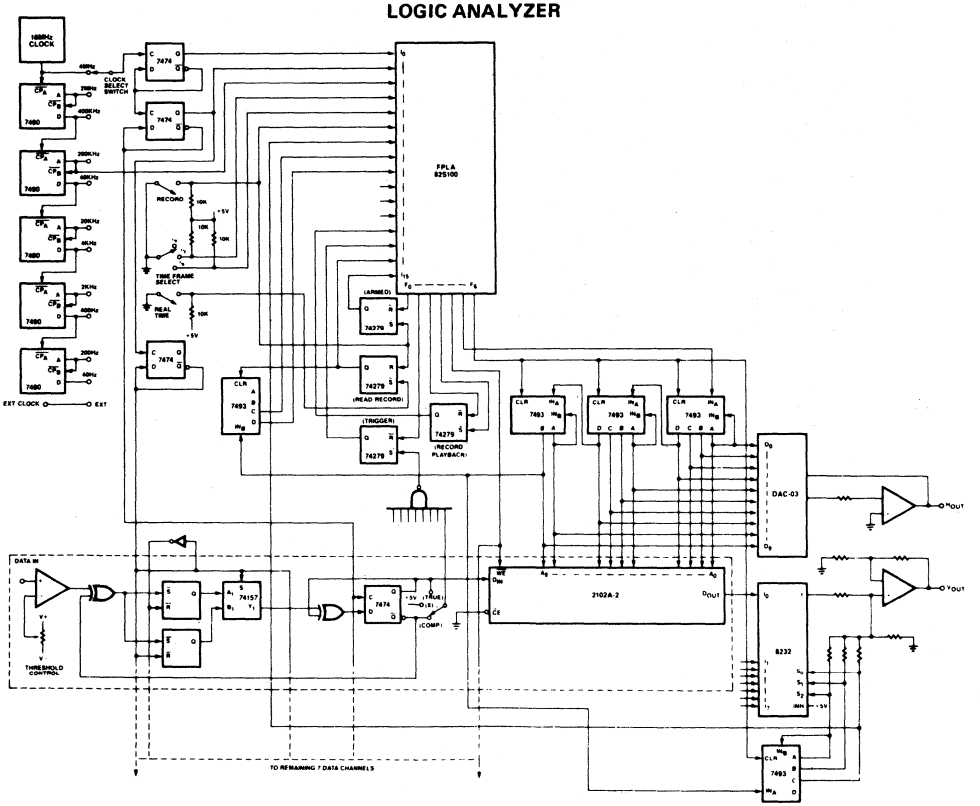


Figure 1.

In the design of any TV display it is necessary to develop the standard SYNC and BLANKING signals to drive the monitor. Although there are ICs which develop these signals, their beam position counter signals are not brought out to external pins. Therefore, to develop video one must duplicate the counter, and nothing will be saved. The circuit shown in Figure 1 uses an FPLA to develop SYNC and BLANKING, plus any video that remains fixed on the TV. Since the outputs of the vertical and horizontal beam position counter are available at the input to the FPLA, any combination of these signals can be used to develop fixed video on the monitor. This fixed video can be anything, such as ruling lines for a TV typewriter, or the background for a video game. For example, in designing a Pong-like video game the FPLA could be used to make the border, the net across the center of the screen, a foul line around the periphery of the court, plus all the necessary SYNC and BLANKING. The remaining circuit would generate the two movable paddles and the ball.

The circuit in Figure 1 generates all necessary signals to operate a standard T.V. monitor in non-interlaced mode with 262 lines per frame, and 60 frames per second. It follows that 262 lines/frame x 60 frames/sec = 15,720 lines/sec → 63.6 μsec/line.

SYNC VIDEO GENERATOR CIRCUIT

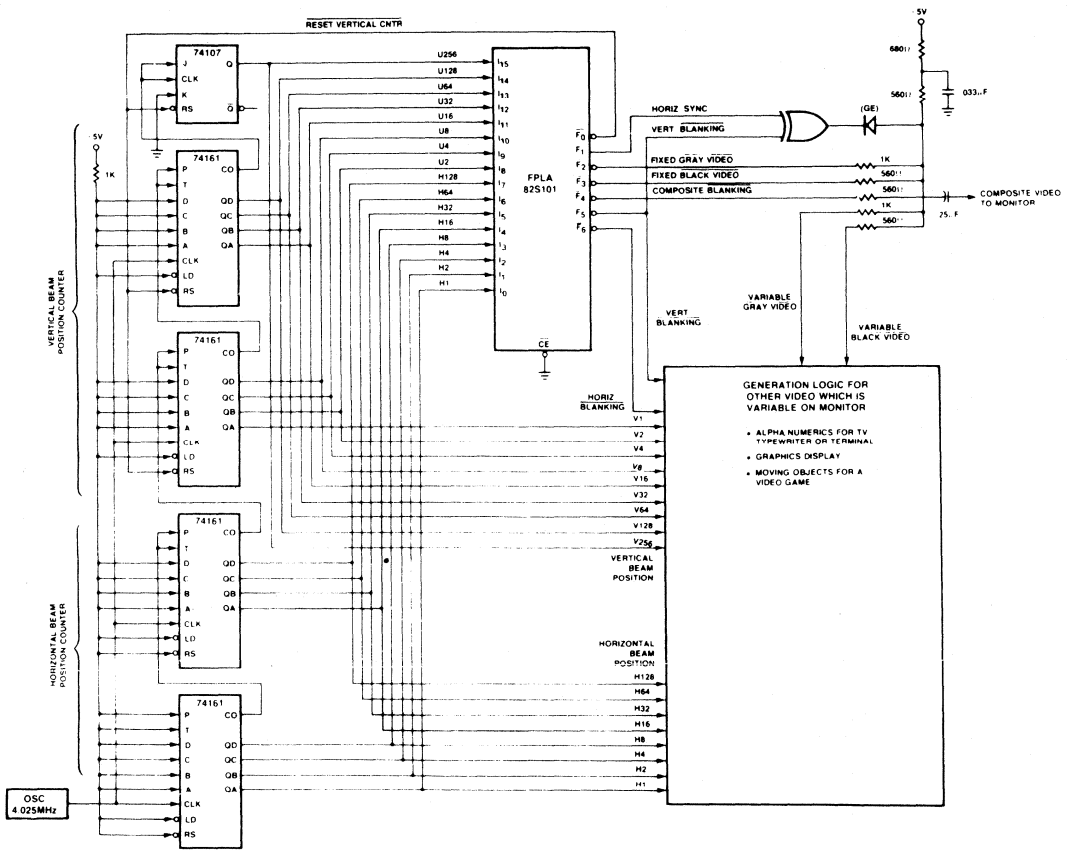
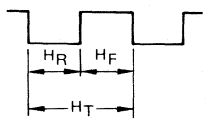


Figure 1.

Assuming each horizontal time of 63.6 μsec to be divided into 256 time units called picture elements (pels), we obtain 63.6 μsec/256 = 248.44 ns/(time unit) → 4.025 MHz. The TV monitor visible area is partitioned timewise as shown in Figure 2. The total horizontal time (H_T) is composed of two parts:

1) The forward visible time (H_F)
2) The retrace time (H_R)



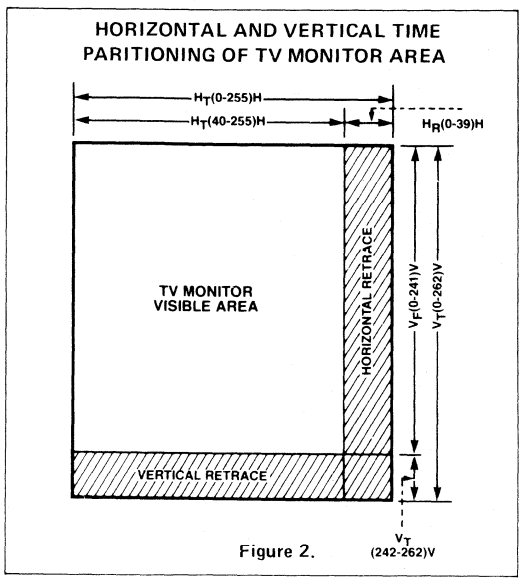
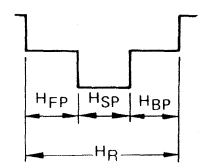


Figure 2.

Where $H_T = H_F + H_R = 63.6\mu sec$

The horizontal retrace time (H_R) is composed of three parts:

1) Front Porch (H_{FP})
2) Horizontal Sync Pulse (H_{SP})
3) Back Porch (H_{BP})



From the Radio Engineers Handbook we get the following information:

$H_R = .155\ H_T = 9.9\ \mu sec = 40$ pels
$H_{FP} = .02H_T = 1.3\ \mu sec = 5$ pels
$H_{SP} = .08H_T = 5.1\ \mu sec = 19$ pels
$H_{BP} = .06H_T = 3.816\mu sec = 16$ pels
$V_R = 0.75\ V_T = 20$ lines

The horizontal sync signals can be developed by using an eight bit counter, operating modulo 256 at the picture element rate of 4.025 MHz, with its outputs connected to an FPLA. There the appropriate counts will be decoded to generate the sync signals as follows:

Count	Signal	Count	Signal
0 → 39	H_R	25 → 39	H_{BP}
0 → 4	H_{FP}	40 → 255	H_F
5 → 23	H_{SP}		

Every time the horizontal counter wraps back to all zeros, the vertical counter will be incremented by one. There will be 20 lines of vertical retrace (V_R), and 242 lines of vertical visible time (V_F) for a total vertical time (V_T) of 262 lines. Starting at the top of the screen the 9 bit vertical counter is all zeros, and is incremented by one at the end of each horizontal forward time (right hand edge of screen). When the counter reaches 241, the vertical blanking signal will be enabled and the polarity of horizontal sync reversed. The counting of horizontal lines continues until count 262 is reached, at which time the vertical counter is asynchronously reset to zero and the vertical blanking signal turned off, giving:

Count	Signal
0-241	V_F
242-263	V_R (reverse polarity of horizontal sync)

The logic equation set for decoding the appropriate counts with the FPLA is tabulated in Figure 3, while the corresponding FPLA Program Table is shown in Figure 4.

FPLA LOGIC EQUATION SET IN TERMS OF WEIGHTED BINARY INPUTS FROM THE COUNTER

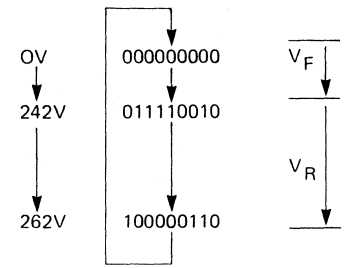
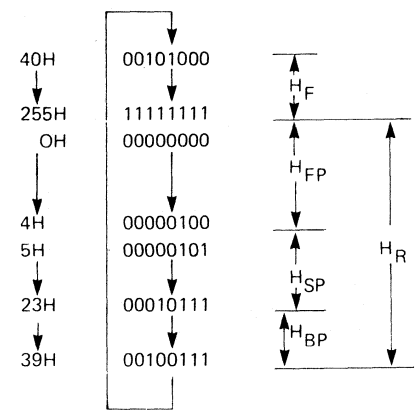
LOGIC EQUATION	ADDRESS SEQUENCE
Reset Vertical CNTR = 256V • 4V • 2V (262V) Vertical Blanking = (256V•128V•64V•32V•16V•8V•4V•2V)+ (242V - 262V) + (256V•128V•64V•32V•16V•8V•4V)+ + (256V•128V•64V•32V•16V•8V•)+ + (256V)	
Horizontal Blanking = (128H•64H•32H)+ (OH-39H) + (128H •64H•32H•16H•8H) Horizontal SYNC = (128H•64H•32H•16H•8H•4H•2H•1H)+ (5H-23H) + (128H•64H•32H•16H•8H•4H•2H)+ + (128H•64H•32H•16H•8H)+ + (128H•64H•32H•16H•8H)	

Figure 3.

FPLA PROGRAM TABLE

COMMENTS	NO.	PRODUCT TERM / INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL H	L	L	L	L	L	H	L / OUTPUT FUNCTION 7	6	5	4	3	2	1	0
RESET	0	H	–	–	–	–	–	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A
HORIZONTAL BLANKING	1	L	H	H	H	H	L	L	H	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
	2	L	H	H	H	H	L	H	–	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
	3	L	H	H	H	H	H	–	–	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
	4	H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	A	•	A	A	•	•	•	•
VERTICAL BLANKING	5	–	–	–	–	–	–	–	–	L	L	L	–	–	–	–	–	A	A	•	A	•	•	•	•
	6	–	–	–	–	–	–	–	–	L	L	H	L	–	–	–	–	A	A	•	A	•	•	•	•
HORIZONTAL SYNC	7	–	–	–	–	–	–	–	–	L	L	L	L	L	H	L	H	A	•	•	•	•	•	A	•
	8	–	–	–	–	–	–	–	–	L	L	L	L	L	H	H	–	A	•	•	•	•	•	A	•
	9	–	–	–	–	–	–	–	–	L	L	L	L	H	–	–	–	A	•	•	•	•	•	A	•
	10	–	–	–	–	–	–	–	–	L	L	L	H	L	–	–	–	A	•	•	•	•	•	A	•
	11																								
SPARE TERMS AVAILABLE FOR MAKING FIXED BLACK OR GRAY VIDEO	12																								
	13																								
	14																								
	15																								
	16																								
	17																								
	18																								
	19																								
	20																								
	21																								
	22																								
	23																								
	24																								
	25																								
	26																								
	27																								
	28																								
	29																								
	30																								
	31																								
	32																								
	33																								
	34																								
	35																								
	36																								
	37																								
	38																								
	39																								
	40																								
	41																								
	42																								
	43																								
	44																								
	45																								
	46																								
	47																								
I/O ASSIGNMENT		256V	128V	64V	32V	16V	8V	4V	2V	128H	64H	32H	16H	8H	4H	2H	1H	SPARE	H_R	V_R	BLANKING	BLACK VIDEO	GRAY VIDEO	H_{SP}	RESET

Figure 4.

The circuit shown in Figure 1 provides tennis court boundaries and scorekeeping capabilities for electronic games, such as the one introduced in EDN August 5, 1976. The TV screen is divided into a matrix of 256 by 256 squares. The FPLA serves to connect the squares so that they take on the shape of a tennis court. Sync signals are generated along with other signals that are necessary for determining which boundary the ball has hit. The score is displayed 0 to 15 using an 8 segment format. The score is disabled while the ball is rallied. The FPLA is particularly suited to this application since the scoreboard requires 16-input AND gates.

PING-PONG COURT AND SCOREBOARD GENERATOR
FOR DETAILS OF PLAYER, BALL, AND CONTROL LOGIC SEE EDN, AUGUST 5, 1975, P. 50

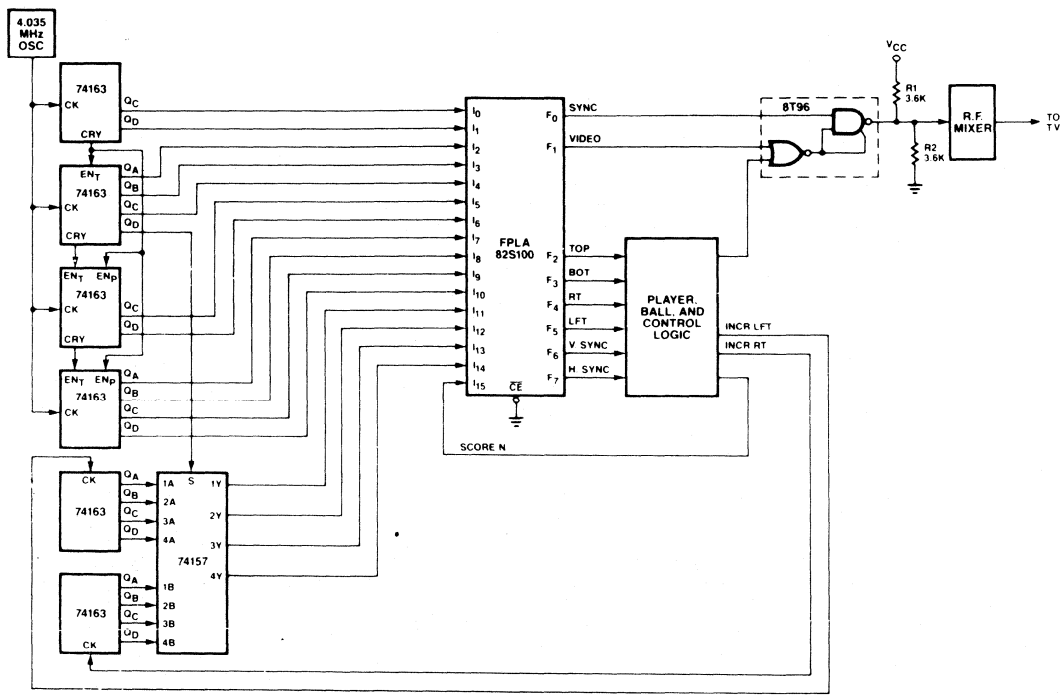


Figure 1.

A TV video display can be controlled via incoming data by using an FPLA to detect machine commands as data from a tape reader. Although a ROM could supply the necessary functions, it cannot use its extra inputs and outputs to form flip-flops that can be set internally by selected character or groups of characters. This technique is used in the proposed design to detect the order of characters passing through the tape reader for creating a "tape backward" alarm, used when reading tape by hand. The block diagram of Figure 1 shows an FPLA used to control the interface between a TV and a tape reader.

The system most critical control function is that of detecting invalid characters, including parity errors, which can be avoided by strobing the outputs, giving them time to settle. Carriage return (CR) is used to reset the character counter in the TV display, while line feed (LF) is used to advance the line counter by one. The "A" character designates a "time" message and is used to select one of two message lengths to detect dropouts or run-ons.

These functions are implemented in the FPLA connected as shown in Figure 2, and suitably programmed. Nothing that CR is always followed by LF, if the first F/F is set by CR

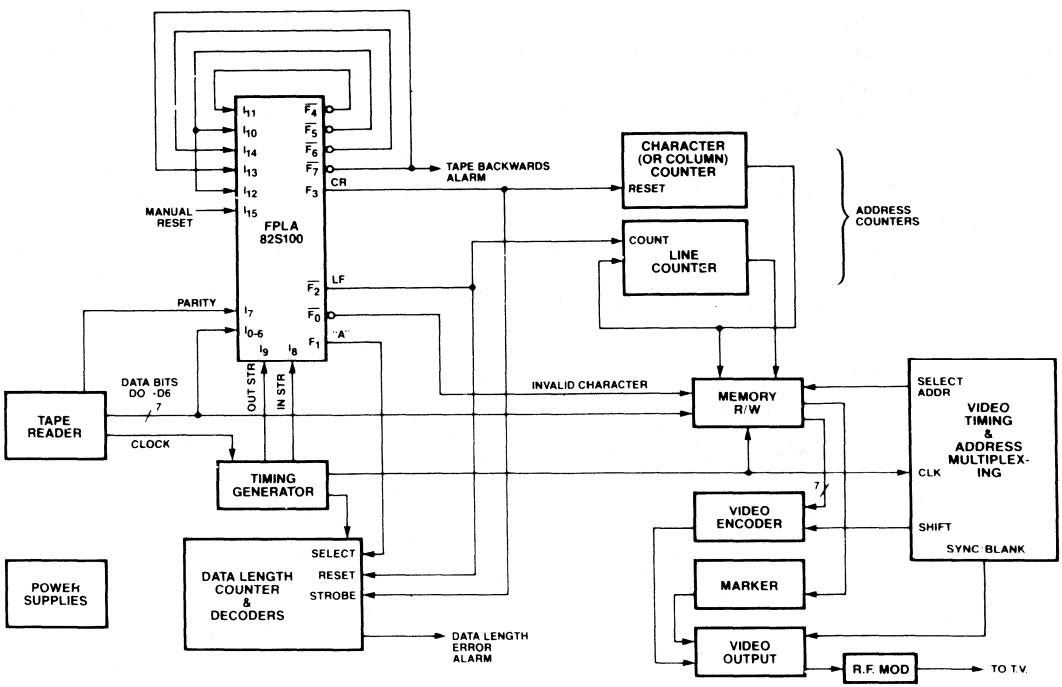


Figure 1.

and reset by LF, its output state is determined by the last of the two characters read. The output that is HIGH after CR is received is "AND" gated with a character that is always present in a message, and then used to set a second F/F. If the tape is being read in the right direction, the 1st F/F is reset before its output can be used to set the 2nd F/F (which goes to the alarm). This approach permits any series of data to be checked for a certain order of characters to flag special conditions, where data can be a set of sensors outputs, as well as digital words.

The proposed TV tape reader processes incoming data, checking for invalid characters (including parity errors), length of messages (number of characters), which length out of two instructions for positioning characters on the TV screen, and which direction the data is moving through the reader. The same circuit using TTL gates would require over 25 chips. An FPLA instead allows up to 42 characters to be detected for validation and controls, with only six product terms dedicated to implement the fold-back flip-flops.

FPLA I/O ASSIGNMENT SHOWING FOLD-BACK CONNECTIONS FOR REALIZING F/F_1 AND F/F_2 CONTROL FLIP-FLOPS

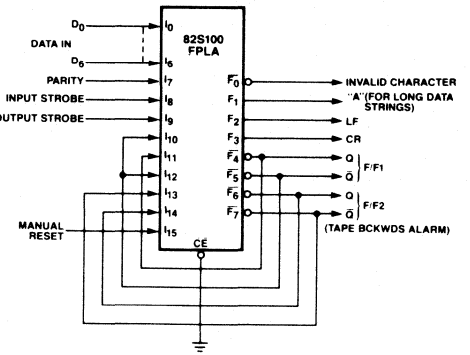


Figure 2.

The input selection property of FPLAs can be combined with simple Set/Reset flip-flops to implement basic logic modules which can be cascaded as shown in Figure 1 to detect group of words in a string of data, such as used by data terminals and other computer controlled devices.

BLOCK DIAGRAM OF MULTISTAGE DIGITAL MESSAGE DECODER

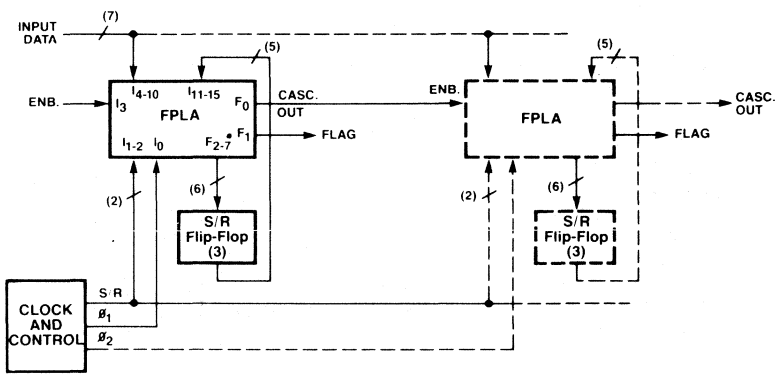


Figure 1.

A typical module, shown in Figure 2, contains 3 stages comprised of an FPLA controlling the setting and resetting of inputs of 3 flip-flops, after proper decoding of the incoming data. Additional circuitry common to all stages/modules generates all necessary clocking signals.

The FPLA is programmed such that product term P_0 contains inputs Ø1, CLK SET, ENABLE (or CASC OUT) from the previous stage, and the True or Complement inputs of the data bus corresponding to the word being decoded. When all these conditions are met, Q1 is set. But if the received word is incorrect, P_1 instead receives as inputs P_0, CLK RESET, and Ø1. This will cause Q1 to be reset anytime an incorrect word is clocked into the module.

To keep the data True signal from the previous stage from being lost while reading the current word, two clock phases are used. While the current state is in use, the previous one is disabled. And since clock Ø2 is a simple inversion of Ø1, only one FPLA input is needed for both.

TYPICAL MODULE OF DIGITAL MESSAGE DECODER
DASH CONNECTIONS RUN *OUTSIDE* FPLA

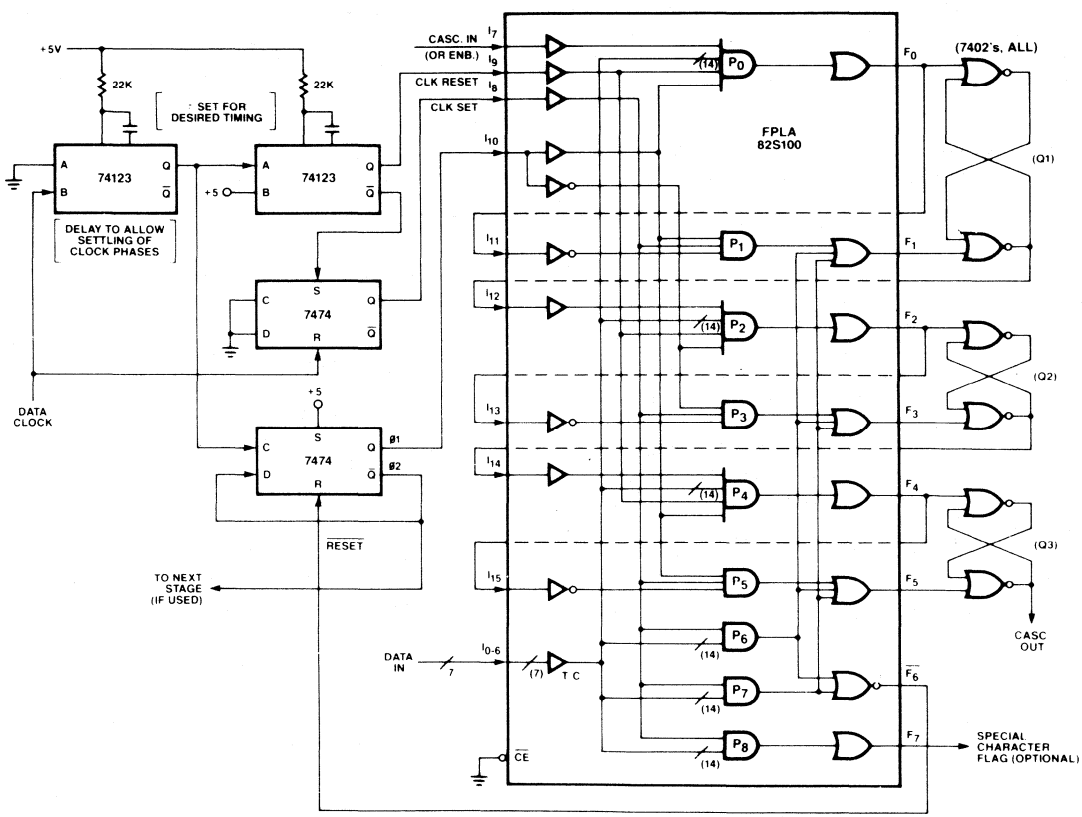


Figure 2.

Note that both phases are generated because if an additional module is used, its first clock phase will be number 2. This way if the circuit is used as an electronic combination lock, it can be changed partly by interchanging FPLAs. That is, each FPLA contains a three word code that can be moved to a different location. As additional security precaution, one person could hold the key, while only another is entrusted with the code. There could even by two keys, both simultaneously needed.

To appreciate the difficulty of breaking such a code to gain access of a system so equipped, note that with two PLA chips it would take over a century to check all the combinations, a rate of one thousand times a second. With four, the time is about 5×10^{11} years.

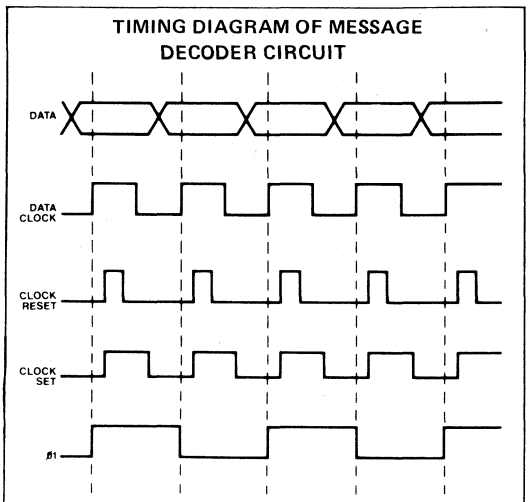


Figure 3.

CALCULATOR CHIP INTERFACE 48

An FPLA can provide an efficient means to interface a calculator chip to other circuts. The calculator can then be used as a terminal, a part of an instrument, or to drive a printer or CRT. The circuit also allows the calculator to be remotely operated. The interface is complete as shown in Figure 1. Timing is derived from the BCD Digit Output. The control and flow of data in and out is described below and by the calculator specification (National MM3738). The actual implementations of the equipment connected to the interface will depend on the desired application. The timing is derived from the Digit select outputs. The first 10 terms in the FPLA may be used alone as a 7 segment to BCD decoder.

The interface contains signals allocated to the FPLA inputs and outputs as follows:

A) Control – $I_{15\text{-}16}$, where:

I_{15}	I_{16}	Function
0	0	Read data from the calculator
0	1	Input numbers 0 - 9, ., %
1	0	Input functions +, –, etc.
1	1	Request D.P. location

B) Data out – $F_{0\text{-}3}$, F_7, where:

1. With interface control at "00", read multiplexed data in BCD form F_0 (A) to F_3 (D). Digit location corresponds to the BCD digit output. Also, F_7 is a "0" for output 1-9, and becomes a "1" when output 0 is displayed. Two other codes exist when F_7 is a "1":

DCB A	Code
111 0	Negative Sign
111 1	Error Condition

2. With interface control at "11", and the D.P. line gated to I_8 (Data input D), the DCP location will be output in BCD corresponding to the digit location of the decimal point. Two word cycles of the calculator output are required to get both data and decimal point location into a buffer or memory.

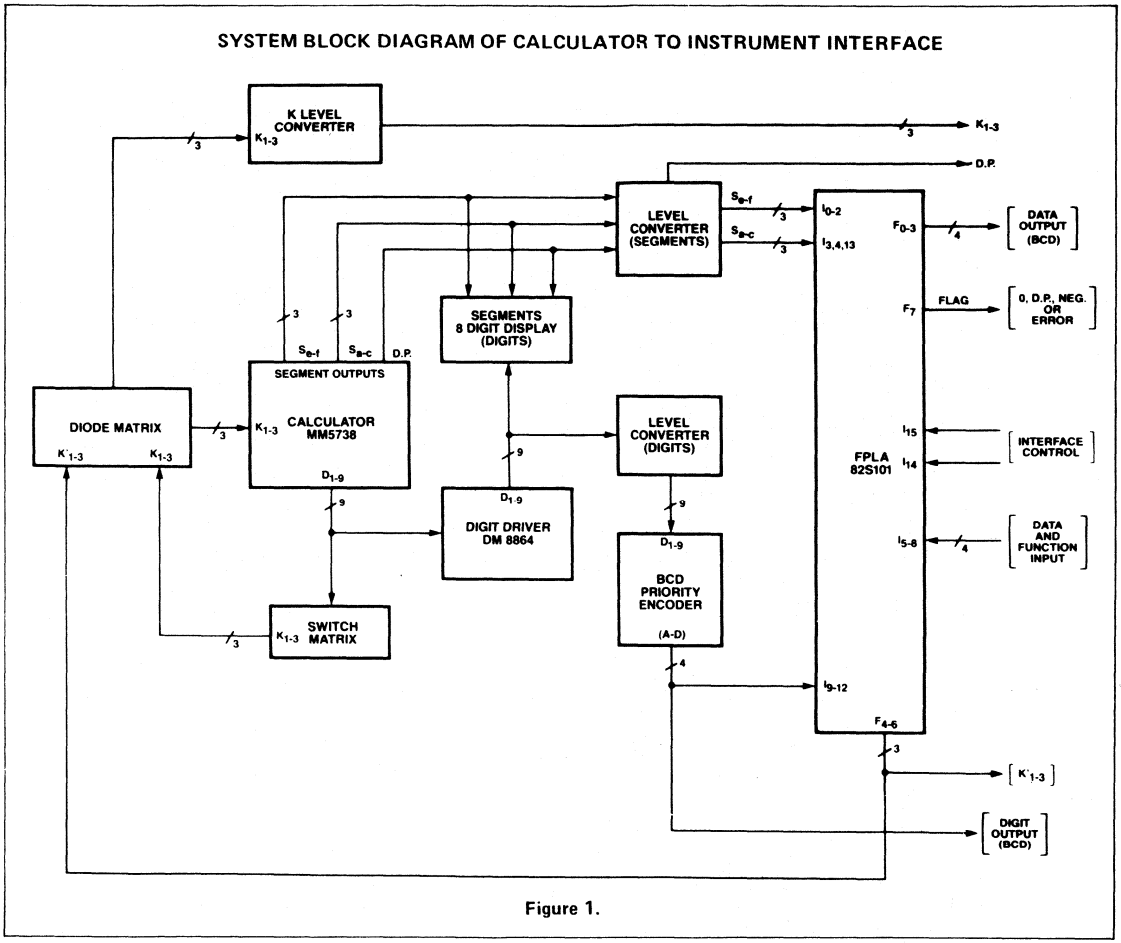


Figure 1.

C) Data Input – With the interface control at "01", data is clocked in at I_5, I_6, I_7, and I_8 as follows:

I_8	I_7	I_6	I_5	Data
0	0	0	0	"0"
0	0	0	1	"1"
0	0	1	0	"2"
0	0	1	1	"3"
0	1	0	0	"4"
0	1	0	1	"5"
0	1	1	0	"6"
0	1	1	1	"7"
1	0	0	0	"8"
1	0	0	1	"9"
1	0	1	0	"."
1	0	1	1	% function

D) Function Input – With the interface control at "10", functions are input as follows:

I_8	I_7	I_6	I_5	Function
0	0	0	1	C
0	0	1	0	K
0	0	1	1	–
0	1	0	0	+
0	1	0	1	MR
0	1	1	0	MS
0	1	1	1	X
1	0	0	0	÷
1	0	0	1	=

FPLA PROGRAM TABLE
Di (I_8) IS ALSO USED FOR D.P. LOCATION

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: OUTPUT FUNCTION 7	6	5	4	3	2	1	0	COMMENT	
0	L	L	L	–	–	–	–	–	–	–	–	H	L	L	L	L	A	•	•	•	•	•	•	•	0	OUTPUT DATA
1	L	L	L	–	–	–	–	–	–	–	–	H	H	H	L	H	•	•	•	•	•	•	•	A	1	
2	L	L	H	–	–	–	–	–	–	–	–	L	H	L	L	L	•	•	•	•	•	•	A	•	2	
3	L	L	L	–	–	–	–	–	–	–	–	L	H	H	L	L	•	•	•	•	•	•	A	A	3	
4	L	L	L	–	–	–	–	–	–	–	–	L	L	H	L	H	•	•	•	•	•	A	•	•	4	
5	L	L	L	–	–	–	–	–	–	–	–	L	L	H	H	L	•	•	•	•	•	A	•	A	5	
6	L	L	L	–	–	–	–	–	–	–	–	L	L	L	H	–	•	•	•	•	•	A	A	•	6	
7	L	L	L	–	–	–	–	–	–	–	–	H	–	H	L	L	•	•	•	•	•	A	A	A	7	
8	L	L	L	–	–	–	–	–	–	–	–	L	L	L	L	L	•	•	•	•	A	•	•	•	8	
9	L	L	L	–	–	–	–	–	–	–	–	L	L	H	L	L	•	•	•	•	A	•	•	A	9	
10	L	H	–	L	L	L	H	L	L	L	L	–	–	–	–	–	•	•	•	A	•	•	•	•	0	INPUT DATA
11	L	H	–	L	L	H	L	L	L	L	H	–	–	–	–	–	•	•	•	A	•	•	•	A	1	
12	L	H	–	L	L	H	H	L	L	H	L	–	–	–	–	–	•	•	•	A	•	•	A	•	2	
13	L	H	–	L	H	L	L	L	L	H	H	–	–	–	–	–	•	•	•	A	•	•	A	A	3	
14	L	H	–	L	H	L	H	L	H	L	L	–	–	–	–	–	•	•	•	A	•	A	•	•	4	
15	L	H	–	L	H	H	L	L	H	L	H	–	–	–	–	–	•	•	•	A	•	A	•	A	5	
16	L	H	–	L	H	H	H	L	H	H	L	–	–	–	–	–	•	•	•	A	•	A	A	•	6	
17	L	H	–	H	L	L	L	L	H	H	H	–	–	–	–	–	•	•	•	A	•	A	A	A	7	
18	L	H	–	H	L	L	H	H	L	L	L	–	–	–	–	–	•	•	•	A	A	•	•	•	8	
19	L	H	–	L	L	H	L	H	L	L	H	–	–	–	–	–	•	•	A	•	A	•	•	A	9	
20	L	H	–	L	L	H	H	H	L	H	L	–	–	–	–	–	•	•	A	•	A	•	A	•	•	
21	L	H	–	L	H	L	L	H	L	H	H	–	–	–	–	–	•	•	A	•	A	•	A	A	%	
22	H	L	–	L	L	L	H	L	L	L	H	–	–	–	–	–	•	A	•	•	•	•	•	A	C	
23	H	L	–	L	L	H	L	L	L	H	L	–	–	–	–	–	•	A	•	•	•	•	A	•	K	
24	H	L	–	L	L	H	H	L	L	H	H	–	–	–	–	–	•	A	•	•	•	•	A	A	–	
25	H	L	–	L	H	L	L	L	H	L	L	–	–	–	–	–	•	A	•	•	•	A	•	•	+	
26	H	L	–	L	H	L	H	L	H	L	H	–	–	–	–	–	•	A	•	•	•	A	•	A	MR	
27	H	L	–	L	H	H	L	L	H	H	L	–	–	–	–	–	•	A	•	•	•	A	A	•	MS	
28	H	L	–	L	H	H	H	L	H	H	H	–	–	–	–	–	•	A	•	•	•	A	A	A	X	
29	H	L	–	H	L	L	L	H	L	L	L	–	–	–	–	–	•	A	•	•	A	•	•	•	÷	
30	H	L	–	H	L	L	H	H	L	L	H	–	–	–	–	–	•	A	•	•	A	•	•	A	=	
31	H	H	–	L	L	L	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	•	A	1	D.P. LOCATION
32	H	H	–	L	L	H	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	•	2	
33	H	H	–	L	L	H	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	•	A	A	3	
34	H	H	–	L	H	L	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	•	4	
35	H	H	–	L	H	L	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	•	A	5	
36	H	H	–	L	H	H	L	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	•	6	
37	H	H	–	L	H	H	H	H	–	–	–	–	–	–	–	–	A	•	•	•	•	A	A	A	7	
38	H	H	–	H	L	L	L	H	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	•	8	
39	H	H	–	H	L	L	H	H	–	–	–	–	–	–	–	–	A	•	•	•	A	•	•	A	9	
40	L	L	H	–	–	–	–	–	–	–	–	L	H	H	H	H	A	•	•	•	A	A	A	•	OUTPUT	
41	L	L	H	–	–	–	–	–	–	–	–	L	L	L	H	L	A	•	•	•	A	A	A	A	ERROR	
I/O ASSIGNMENT	CONTROL	SC	D	C	B	A	Di	Ci	Bi	Ai	Sg	Sf	Se	Sb	Sa	FLAG	K_3	K_2	K_1	D	C	B	A		

Figure 2.

Note that when the Functions or Data are input to the calculator, they are read out on the data lines for comparison (if desired). This feature is different from the Data Output by the presence of a "1" in either K_1, K_2, or K_3.

E) Digit Outputs (not an output of the FPLA) – These lines are the digit select outputs of the calculator encoded in BCD. They are used primarily when storing Output Data and Decimal Point location in memory.

F) K_1, K_2, K_3 – These outputs at F_4, F_5 and F_6 are used to input data to the calculator. The calculator recognizes data on the basis of a K input and the Digit Select cycle. The FPLA will activate the proper K line in synchronism with the digit select lines. The K_1, K_2, and K_3 lines as shown on the diagram will indicate switch (keyboard) operation as well as remote data input. The K lines may be used to discriminate between data input and data output functions at the interface.

G) Digit Select inputs to FPLA, I_9, I_{10}, I_{11}, and I_{12} (A' to D' respectively) – These lines are used to select the proper K output when inputting data or functions.

H) Segment inputs to FPLA (I_0 to I_4) These are S_a, S_b, S_e, S_f, S_g respectively, and uniquely identify the BCD outputs of 0 to 9 and the (–) sign. The error condition requires segment S_c input to I_{14} for a unique identification.

The FPLA Program Table containing all codes necessary to service the above interface lines is shown in Figure 2.

MICROPROCESSOR TO CALCULATOR-CHIP INTERFACE 49

The circuit shown in Figure 1 will interface directly between a microprocessor and a calculator chip in systems where low cost, high computational power, but no need for a high speed computation ability (i.e., FFTs and R.T. systems) exist. It could be used in low cost, low speed navigational systems and business applications. FPLA U1 acts as a keyboard simulator receiving output port information from the microprocessor via the data bus, and converting a binary code into the correct keyboard entry to the calculator. The MP will know when to output by interrogating Input Port 1 through FPLA U2 to ascertain if the matrix it wishes to talk to is available at that time. Since the calculator will run in the kHz range and the MP in the MHz range, no problem should be encountered in interfacing.

The U1, U2 combination provide the calculator with the serial inputs normally provided by push buttons. U3 and U4 provide the microprocessor with the answer when the computation is completed.

FPLA-U3 would decode from 8 strobe lines and 7 segment signals an appropriate binary code on to the data bus, thereby providing the microprocessor with answers and evidence of completion of a specific task. It also could provide a method of acknowledgement of receipt of numbers or tasks. FPLA-U4 decodes the upper eight bits of numbers. Since FPLAs are field programmable the system can be adapted to different calculator configurations and efficient software configurations for the microprocessor chosen.

CALCULATOR CHIP TO μP INTERFACE

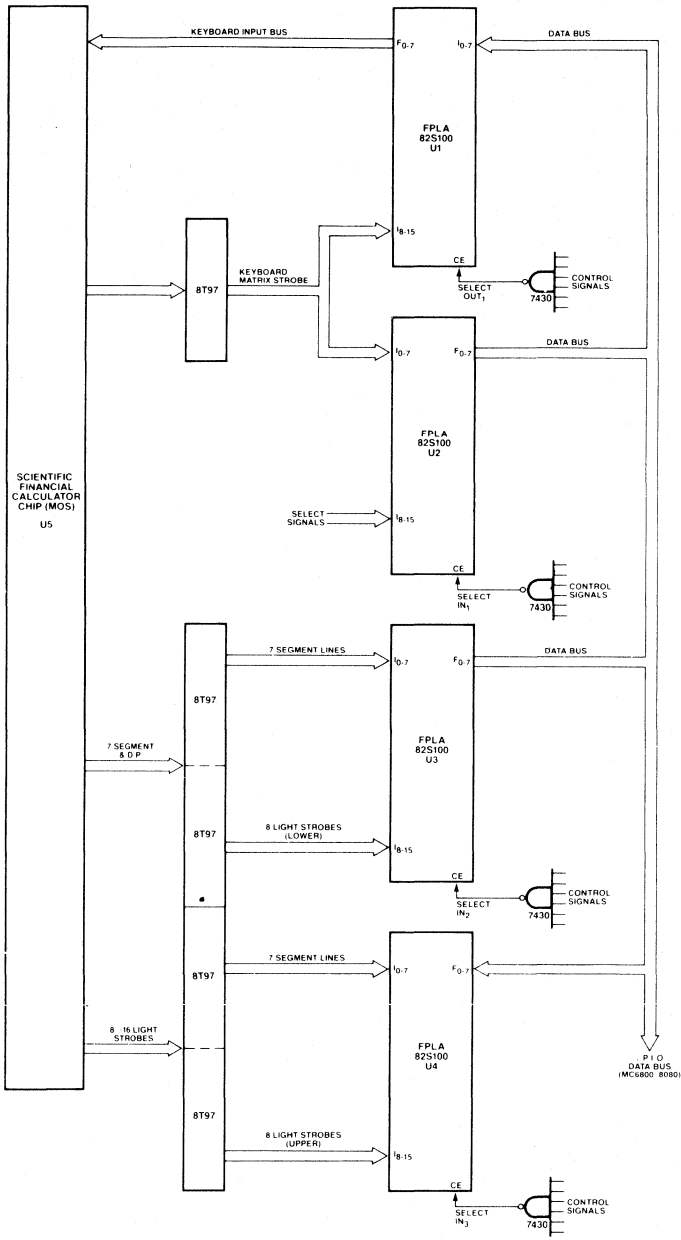


Figure 1.

The circuit shown in Figure 1 uses two Signetics' 82S100 FPLAs, and a few discrete components, to implement a full 64 character ASCII keyboard compact enough to be hand-held or built into the front panel of a computer. Its compactness is due to the use of only 23 keys: 16 character keys, 4 mode keys, and 3 special purpose keys. The keyboard switches need only be SPST-NO; the transistor is not critical, and the LEDs should be efficient enough to be visible with 6 mA current.

The circuit output is a positive logic 7-bit ASCII code, with data valid on the rising edge of the positive-going strobe pulse. Approximately 2 ms of strobe delay is provided for key contact debouncing.

Each of the 16 character keys generates 1 of 4 different characters, depending upon which of four modes the keyboard is in, as selected by the mode keys. When a mode is selected, the keyboard stays in that mode for as many characters as needed until another mode is selected. Included is protection against false character codes due to two keys being depressed at once, key contact debouncing, and 2-key rollover to speed data entry. Also provided are three single purpose keys for carriage return, line feed, and rubout. All characters and functions are supplied by the two FPLAs, programmed as shown in the program tables in Figures 2 and 3, with just 17 Product Terms used in each FPLA. Partially used or partially functional FPLAs may also be used, as long as there are 17 useable Product Terms, and all the 8 outputs operate. The decoding scheme is straightforward. The 16 character keys are divided up into two 8-member groups, and within each group the individual keys are encoded down to the least significant 3 bits of the output. Bit 3 (the fourth bit) is "0" for the group (1) keys and "1" for the group (2) keys. Bits 4 and 5 are set by which of the four modes the keyboard is in. Bit 6 is simply the inverse of bit 5 except when one of the special keys (CR, LF, or RO) is depressed. The remaining of the 17 Product Terms are used to generate the strobe pulse and implement the 2-key rollover feature with the false code protection.

ASCII KEYBOARD ENCODE LOGIC

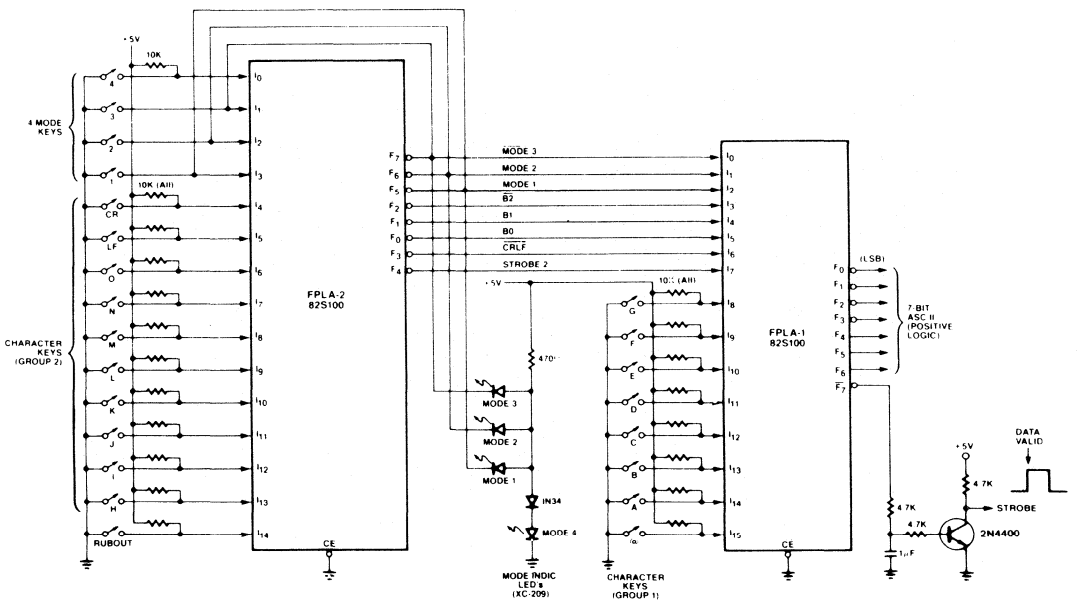


Figure 1.

PROGRAM TABLE FOR FPLA-1

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: L	H	H	H	L	L	L	L	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	–	–	–	H	H	H	H	–	–	–	–	H	–	–	–									•	•	•	•	•	A	•	•
1	–	–	H	H	–	–	H	H	–	–	–	H	–	–	–	–									•	•	•	•	•	•	A	•
2	–	H	–	H	–	H	–	H	–	–	H	–	–	–	–	–									•	•	•	•	•	•	•	A
3	–	–	–	–	–	–	–	–	–	H	–	–	–	–	L	–									•	A	•	•	•	•	•	•
4	–	–	–	–	–	–	–	–	H	–	–	–	–	–	–	–									•	•	•	•	A	•	•	•
5	–	–	–	–	–	–	–	–	–	H	–	–	–	H	H	–									•	•	A	•	•	•	•	•
6	–	–	–	–	–	–	–	–	–	H	–	–	–	H	–	H									•	•	•	A	•	•	•	•
7	–	–	–	–	–	–	–	–	–	H	–	–	–	L	–	–									•	A	•	•	•	•	•	•
8	L	H	H	H	H	H	H	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
9	H	L	H	H	H	H	H	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
10	H	H	L	H	H	H	H	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
11	H	H	L	H	H	H	H	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
12	H	H	H	H	L	H	H	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
13	H	H	H	H	H	L	H	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
14	H	H	H	H	H	H	L	H	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
15	H	H	H	H	H	H	H	L	H	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•
16	H	H	H	H	H	H	H	H	L	–	–	–	–	–	–	–									A	•	•	•	•	•	•	•

Figure 2.

PROGRAM TABLE FOR FPLA-2

NO.	PRODUCT TERM: INPUT VARIABLE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ACTIVE LEVEL: L	L	L	L	L	L	L	L	OUTPUT FUNCTION 7	6	5	4	3	2	1	0
0	–	H	–	–	–	–	H	H	H	H	–	H	–	–	–	–									•	•	•	•	•	A	•	•
1	–	H	–	–	H	H	–	–	H	H	H	–	–	–	–	–									•	•	•	•	•	•	A	•
2	–	H	–	H	–	H	–	H	–	H	–	H	–	–	–	–									•	•	•	•	•	•	•	A
3	–	L	H	H	H	H	H	H	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
4	–	H	L	H	H	H	H	H	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
5	–	H	H	L	H	H	H	H	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
6	–	H	H	H	L	H	H	H	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
7	–	H	H	H	H	L	H	H	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
8	–	H	H	H	H	H	L	H	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
9	–	H	H	H	H	H	H	L	H	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
10	–	H	H	H	H	H	H	H	L	H	H	H	–	–	–	–									•	•	•	A	•	•	•	•
11	–	H	H	H	H	H	H	H	H	L	H	H	–	–	–	–									•	•	•	A	•	•	•	•
12	–	H	H	H	H	H	H	H	H	H	L	H	–	–	–	–									•	•	•	A	A	•	•	•
13	–	H	H	H	H	H	H	H	H	H	H	L	–	–	–	–									•	•	•	A	A	•	•	•
14	–	H	–	–	–	–	–	–	–	–	–	–	L	H	H	H									•	•	A	•	•	•	•	•
15	–	H	–	–	–	–	–	–	–	–	–	–	H	L	H	H									•	A	•	•	•	•	•	•
16	–	H	–	–	–	–	–	–	–	–	–	–	H	H	L	H									A	•	•	•	•	•	•	•

Figure 3.

In many applications it is desirable to expand the capabilities of inexpensive calculators so that they may be easily interfaced to printers, BCD displays, or computer data input. The output of most calculator chips is automatically multiplexed and presented in standard 7-segment form. For each digit there is a digit select output signal which appears sequentially while segment outputs provide the decoded figure.

CONVERSION TRUTH-TABLE TO BE PROGRAMMED IN THE FPLA

NUMBER	INPUTS: DIGIT SELECT LINE 10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0	INPUTS: SEGMENT LINE a	b	c	d	e	f	g	OUTPUTS: BCD 2^3	2^2	2^1	2^0	OUTPUTS: DIGIT 2^2	2^1	2^0	OUTPUTS: STROBE
0	DON'T CARE (X)								1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1									0	1	1	0	0	0	0	0	0	0	1	0	0	0	0
2									1	1	0	1	1	0	1	0	0	1	0	0	0	0	0
3									1	1	1	1	0	0	1	0	0	1	1	0	0	0	0
4									0	1	1	0	0	1	1	0	1	0	0	0	0	0	0
5									1	0	1	1	0	1	1	0	1	0	1	0	0	0	0
6									1	0	1	1	1	1	1	0	1	1	0	0	0	0	0
7									1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
8									1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
9									1	1	1	1	0	1	1	1	0	0	1	0	0	0	0
BLANK									0	0	0	0	0	0	0	(1	1	1	1)*	0	0	0	0
	1	0	0	0	0	0	0	0	DON'T CARE (X)							0	0	0	0	0	0	1	1
	0	1	0	0	0	0	0	0								0	0	0	0	0	1	0	1
	0	0	1	0	0	0	0	0								0	0	0	0	0	1	1	1
	0	0	0	1	0	0	0	0								0	0	0	0	1	0	0	1
	0	0	0	0	1	0	0	0								0	0	0	0	1	0	1	1
	0	0	0	0	0	1	0	0								0	0	0	0	1	1	0	1
	0	0	0	0	0	0	1	0								0	0	0	0	1	1	1	1
	0	0	0	0	0	0	0	1								0	0	0	0	1	1	1	1

* or (0000)

Figure 1.

The circuit in Figure 1 shows how an FPLA can provide a simple means for converting the 7-segment calculator output to Binary Coded Decimal. Further decoding of up to eight calculator digit select lines provides a binary number indicating which digit is being displayed, along with a strobe pulse to indicate when a valid digit occurs. The full conversion truth-table to be programmed in the FPLA is tabulated in Figure 2.

CIRCUIT DIAGRAM

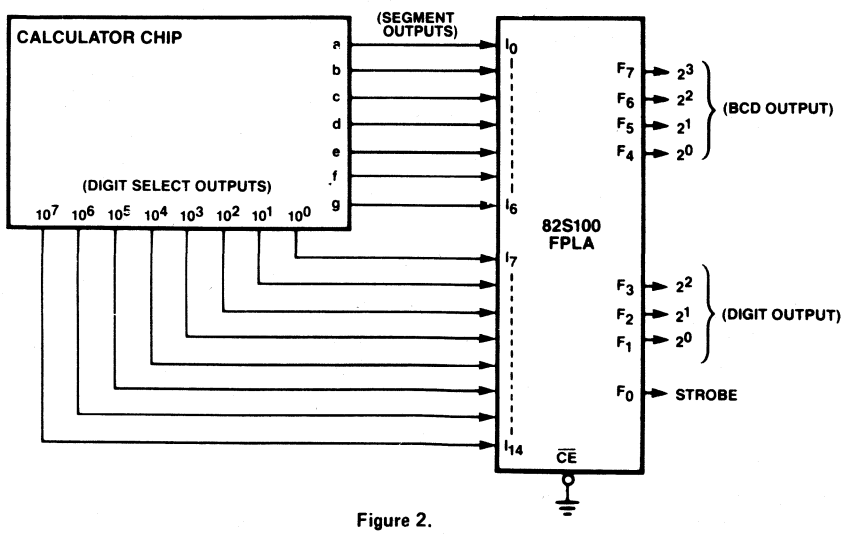


Figure 2.

In wide range and multifunction instruments, the FPLA can be effectively used to provide proper decimal point display and to illuminate function legends. The circuit in Figure 1 shows the application of an FPLA in an autoranging digital multimeter. The FPLA receives inputs from a shift register in the autoranging section. It also accepts inputs from the function select switch. The open collector outputs of the FPLA could directly drive the decimal point within its limit. Other outputs are used to turn on appropriate LEDs displaying legends such as V, MV, R, KΩ, A, MA etc. Buffering may be necessary. The FPLA can accommodate large flexible ranges as well as many functions, thus reducing design time and IC gate packages in a family of instruments.

D.P. AND LEGEND DISPLAY DRIVER

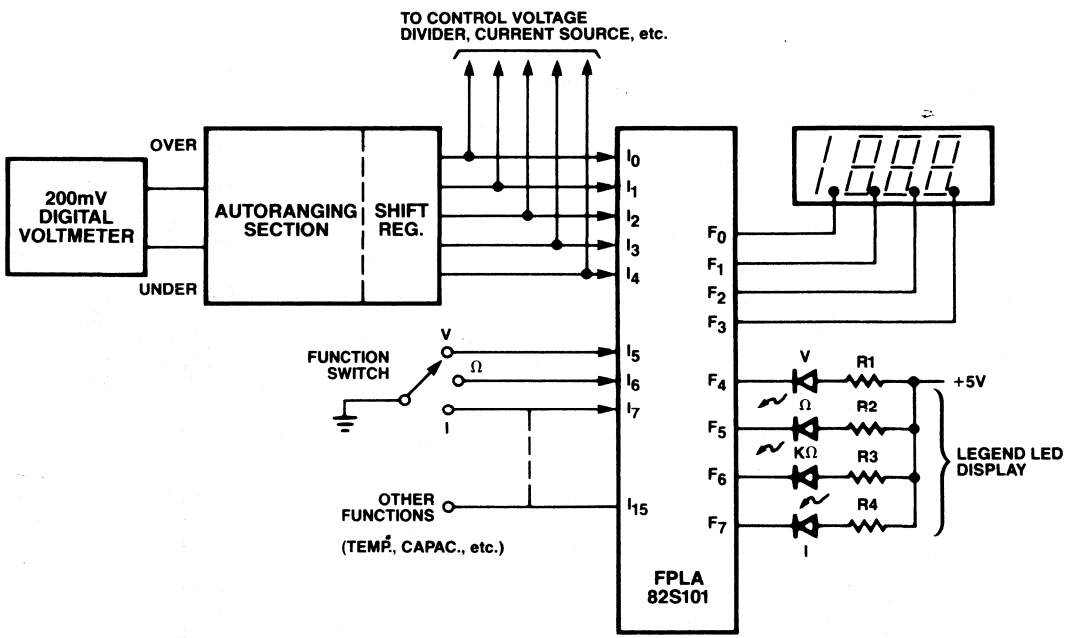


Figure 1.

The circuit diagrammed in Figure 1 is designed to accept a string of hexadecimal characters from a calculator type keyboard and provide a 10 second signal to a user defined actuator such as a solenoid driven door latch, when the character string matches the programmed combination.

Four 16-character combinations may be programmed in the FPLA. The combination to be used is specified by the position of the two combination select switches S_1 and S_2. Combinations of shorter length may be programmed to allow easier entry to the device being locked, but maximum security will be realized using a 16-character combination. Before using, the lock must first be reset in a known state. Depressing the RESET button on the keyboard forces the reset line LOW. The RESET signal is buffered by the FPLA and appears as an active-LOW signal called BRESET, which causes the TRUE flip-flop to be forced true, the OPEN timer to be forced off, and the character counter to be loaded with zero. The trial combination is then entered by depressing the numerical keys in sequence. Upon depressing the last key in a correct sequence a 10 second actuation will occur.

Depressing a numerical key causes a hexadecimal data pattern to appear on the four data lines D_0-D_3. An active-LOW strobe from the keyboard informs the FPLA that a valid data pattern is being presented. The STROBE signal is buffered by the FPLA to become the active-LOW signal BSTROBE. While STROBE is LOW, the data field to the FPLA is compared to the programmed data character asso-

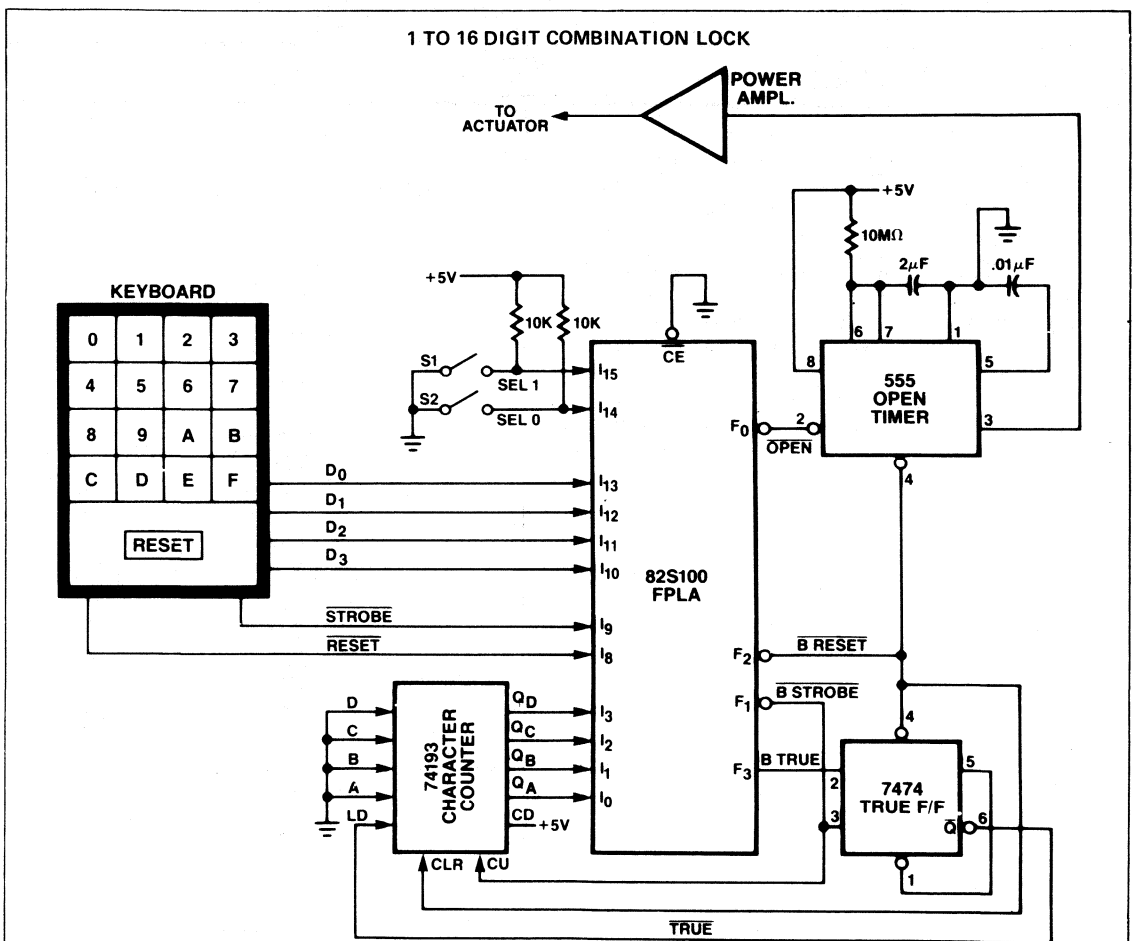


Figure 1.

ciated with the current character count (C_0-C_3), and combination select (SEL_0-SEL_1). If the data matches the programmed character, signal BTRUE will be forced HIGH; if not, it will appear as a LOW. At the rising edge of BSTROBE, the state of BTRUE is latched into the TRUE flip-flop, and the character counter is incremented. If a mismatch had existed between the data and the programmed character, a LOW would be clocked into the TRUE flip-flop, causing it to latch itself in the false state, resetting the character counter and ignoring all future data comparisons until the RESET key is depressed.

The character counter increments for each sequential successful match until a preprogrammed terminal count is reached. The OPEN signal is activated when the terminal count has been reached, STROBE is LOW, and the data field matches the programmed character associated with terminal count and the selected combination. When the OPEN signal goes LOW, it fires a 10 second interval timer. The HIGH output from the running timer is applied to a power amplifier to drive the user defined actuator. For a safe, or door lock application, the actuator could be a solenoid driven latch pin. For automotive applications, the actuator could be an ignition enable relay. For computer applications, this device could provide restricted use of certain peripherals or of the entire computing mechanism.

Additional outputs are available in the FPLA to provide an alarm function. For example, an alarm bit could be programmed to be activated when an excessive number of characters were detected in a string. In safe door lock applications, a battery back up to the normal supply is recommended to prevent permanently sealing the safe upon loss of power to the unit.

DATA SECURITY ENCODER 54

The security of a data train on a common carrier is enhanced when data is encoded to a particular hard-wired encoded version. This results in the correct detection only in a similarly programmed receiving terminal.

The system shown in Figure 1 seems to have well over 8!=(403020) different combinations without resorting to anything but pulse permutations. In applications for computer systems, the computer could use a random sequence to program devices, as only devices which communicate need to know how they are programmed. Data security is assured, since only the programmer knows the permutation in use.

DATA SYSTEM SECURITY ENCODER

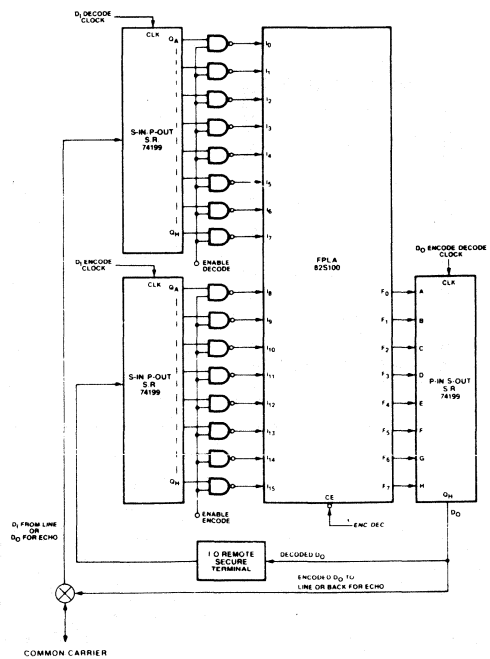


Figure 1.

The calendar clock date reminder decodes 48 unique dates in a calendar year. This would be used to give the owner a reminder of birthdays, anniversaries or any other special day he feels he should remember.

The FPLA is programmed to recognize the dates selected by the owner. Practically, it would be programmed to activate a monitor several days before the actual event, to give the owner time to respond to the reminder.

The circuit in Figure 1 gates the normally multiplexed output of a calendar clock circuit into 11 input lines to the FPLA. To do this, it must demultiplex the BCD output and latch it into registers for a stable input to the FPLA. This is accomplished by the calendar clock's internal digit driver used to gate the appropriate 74175 register. For example, the clock digit driver that is used to display the least significant days digit, D0, is used to latch this data into IC_4 for input into the FPLA as I_0 thru I_3.

When the month and day BCD input match the programmed date, the FPLA outputs F0 - F7 are used to visually indicate the reminder. This can be implemented simply with LEDs driven by the FPLA outputs to indicate various events.

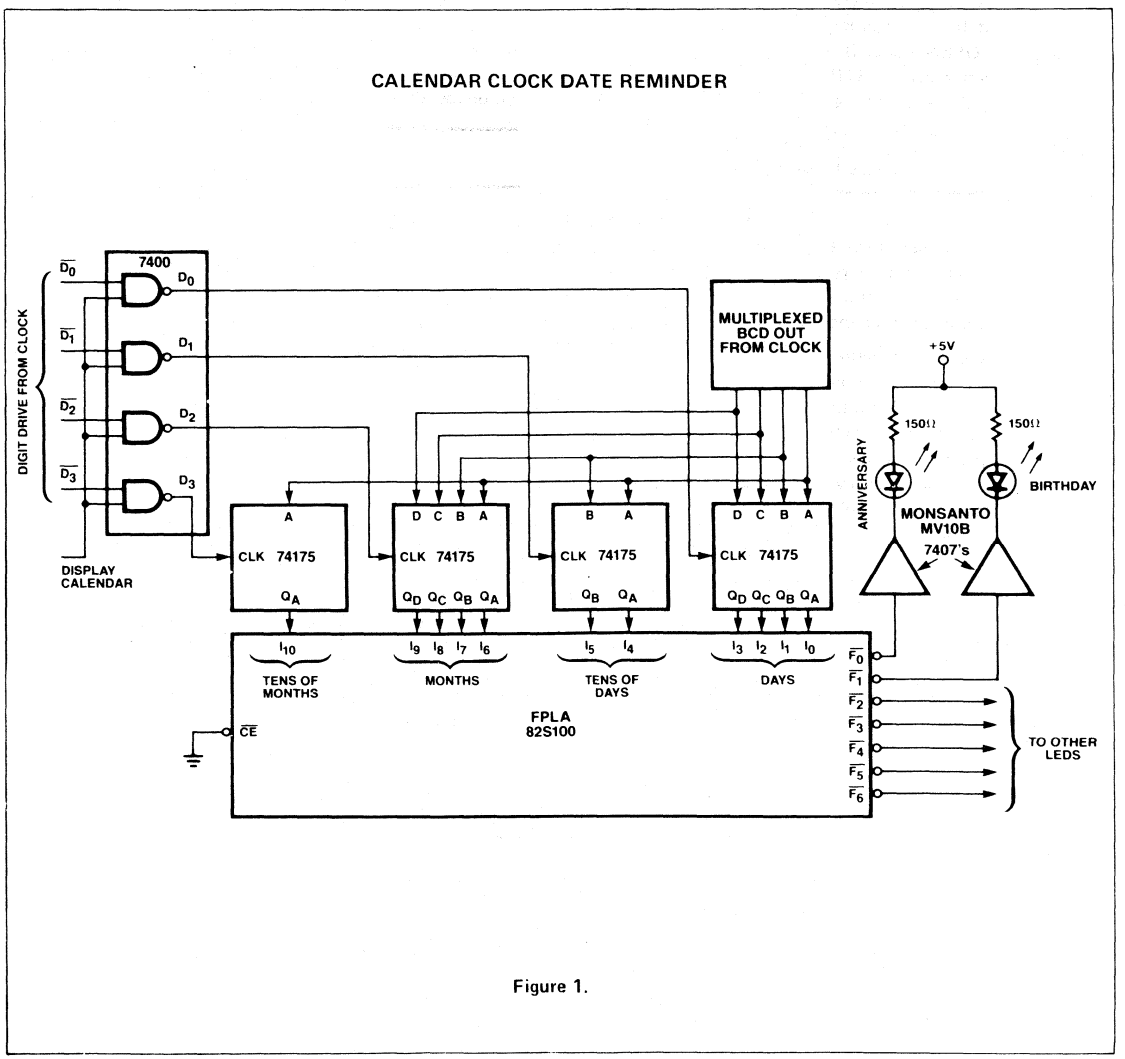


Figure 1.

A preventive Maintenance Monitor can be used to give an accurate visual or audio indication of maintenance required for any type of hardware or equipment at specified time intervals. It could be used with costly machinery that required various maintenance actions at diverse times, without which degradation of equipment would result.

Equipment where such a positive maintenance indication can be utilized range from simple computer hardware such as mag-tape transports and disks, to more complex electro-mechanical machinery and systems such as airplanes, etc.

The circuit shown in Figure 1 utilizes the elapsed time meter usually present on any critical piece of hardware/system. The ETM can be modified, like a thumbwheel switch, to provide BCD outputs to the Preventive Maintenance Monitor.

Although general in principle, the circuit gives an example of a PMM utilized to implement a typical maintenance schedule for an automobile, as recommended by the manufacturer's specifications. The input to the system in this case is the odometer reading. (The odometer has been modified to provide BCD outputs). The outputs of the system are real time indications of which items need maintenance, and when. There are two types of LED indicators: one indicates that some function should be checked, while the other indicates something needs to be replaced or changed.

The inputs to the FPLA are four BCD digits, representing miles in tens, hundreds, thousands, and tens of thousands. Using this scheme, when the odometer value is decoded by the FPLA to indicate a maintenance action, the LED indicators for that action will remain lit for ten miles until the tens BCD changes. By using less inputs the indicators would stay lit longer. For example, 12 inputs, (eliminating tens), would cause the indicators to stay on for a hundred miles, which may be more desirable. Any thing less than 12 inputs (3 BCD's) would make the degree of resolution time when action is required too low.

PREVENTIVE MAINTENANCE MONITOR
(N) FOR A GIVEN F_X DEPENDS ON THE NUMBER OF MAINTENANCE ITEMS DRIVEN BY EACH FPLA OUTPUT. FOR EXAMPLE, F_0 WOULD NEED 6 LED BRANCHES TO DRIVE 1) ENGINE COOLANT CHANGE, 2) PLUG REPLACEMENT 3) CHECK BRAKE LINING AND DRUM, 4) CHANGE BRAKE FLUID, 5) CHECK AND REPLENISH STEERING GEAR BOX, AND 6) REPLACE PCV VALVE

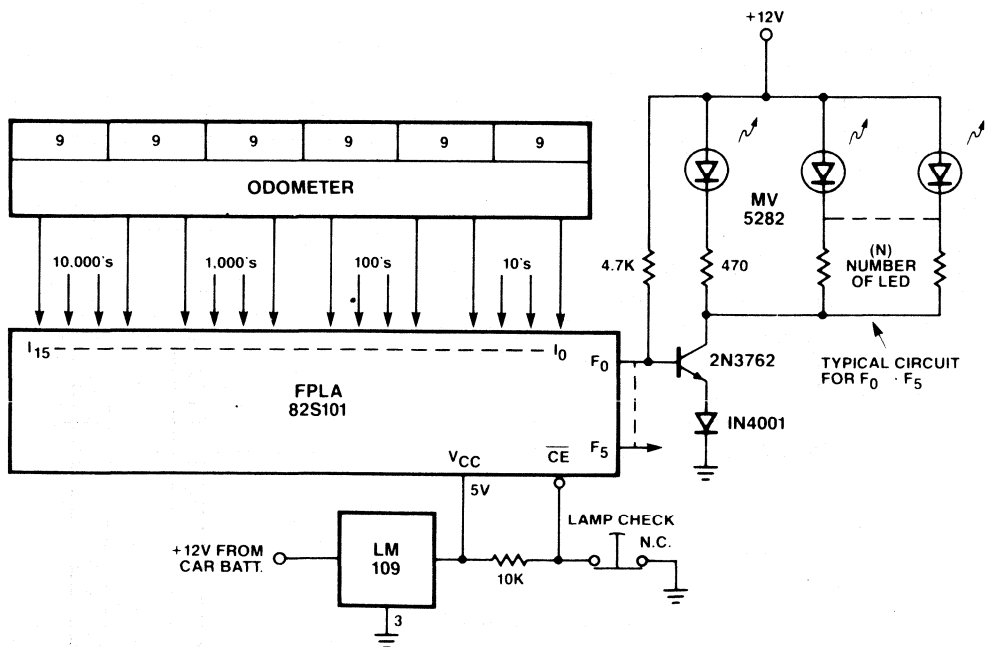


Figure 1.

Just one FPLA easily implements the manufacturer's maintenance schedule outlined in Figure 2. As shown in the Program Table in Figure 3, only 34 product terms are necessary to carry maintenance out to 99,000 miles. After 100,000 miles the PMM system would wraparound and be functional for however long the car lasted. Five outputs (0-4) are used for the four frequency classes of maintenance as shown on the left column of the maintenance schedule.

In addition, F5 is used for indicating replacements or changes needed. F6 and F7 are not used but could be used, along with some of the unused product terms, for some other indication such as a reminder as to when the car should be brought in for its warranty check-ups. The momentary push button switch is used as a lamp-check by forcing all outputs to the default HIGH state via the $\overline{CE}$ input.

TYPICAL AUTOMOBILE MAINTENANCE SCHEDULE
(R) MEANS REPLACE OR CHANGE. F_X ARE FPLA OUTPUTS ACTIVATED AT THE GIVEN MILEAGE INTERVALS

FX ASSIGNMENT	UNIT: 1,000 miles	1	3	6	9	12	15	18	21	24	27	30	33	36	Remarks
0	Change engine coolant					X				X				X	or within 12 months
1	Replace oil filter element	X		X		X		X		X		X		X	
2	Check battery acid level & specific gravity	X	X	X	X	X	X	X	X	X	X	X	X	X	
2	Check & adjust fan and belt tension	X	X	X	X	X	X	X	X	X	X	X	X	X	
2	Check & adjust distributor dwell angle, points & gap	X	X	X	X	X	X	X	X	X	X	X	X	X	
0,1	Check & clean or replace spark plugs	X		X		R		X		R		X		R	
5,2	Clean or replace air cleaner element		X	X	X	X	X	R	X	X	X	X	X	R	
3	Replace fuel filter element									X					
	Tighten nuts & bolts on engine	X													
1	Check & adjust valve clearance	X		X		X		X		X		X		X	
2	Check engine idling speed & carburator condition	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	Check damage of fuel hoses	X		X		X		X		X		X		X	Replace every 48 months
2	Check & adjust ignition timing	X	X	X	X	X	X	X	X	X	X	X	X	X	
4	Check resistive cord resistance (spark plug leads)	X				X				X				X	
	CHASSIS & BODY														
0	Check brake lining & drum					X				X				X	
2	Check brake pad & disc		X	X	X	X	X	X	X	X	X	X	X	X	
1	Check brake booster operation			X		X		X		X		X		X	
2	Check & adjust brake pedal free play & parking brake	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	Check & adjust clutch pedal free play	X		X		X		X		X		X		X	
1	Check steering free play, linkage, front suspension & ball joints	X		X		X		X		X		X		X	
1	Rotate tires			X		X		X		X		X		X	
4	Check front end alignment (side slip)	X				X				X				X	
1	Check leakage of oil, fuel, fluid & water, damage of brake pipes & hoses	X		X		X		X		X		X		X	
4	Tighten nuts & bolts on chassis & body	X				X				X				X	
	LUBRICATION														
2	Change engine oil	X	X	X	X	X	X	X	X	X	X	X	X	X	or within 3 months
0	Change brake fluid (w/disc brake)					X				X				X	
0	Check & replenish steering gear box					X				X				X	
5,1	Check & replenish or change transmission & differential gear oil			X		X		R*		X		X		R*	*or within 24 months
5,2	Check & replenish or change automatic transmission fluid	X	X	X	X	X	X	R*	X	X	X	X	X	R*	*or within 24 months
3	Lubricate ball joints & front wheel bearings									X					or within 24 months
	CRANKCASE & EXHAUST EMISSION CONTROL SYSTEM														
4	Check operation of throttle positioner, speed marker & vacuum switching valve, check hose connections	X				X				X				X	or every 12 months
0	Replace positive crankcase ventilation valve, clean & check connections					X				X				X	or every 12 months

Figure 2.

FPLA PROGRAM TABLE FOR IMPLEMENTING ASSUMED MAINTENANCE SCHEDULE

	PRODUCT TERM																ACTIVE LEVEL							
	INPUT VARIABLE																H	H	H	H	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION							
																	7	6	5	4	3	2	1	0
0	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	•
1	H	H	H	H	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
2	H	H	H	H	H	L	L	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
3	H	H	H	H	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
4	H	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	A
5	H	H	H	L	H	L	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
6	H	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	A	A	A	•	•	A	A	•
7	H	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
8	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	A	A	•	A	A	A	A	A
9	H	H	L	H	H	L	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
10	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
11	H	H	L	L	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
12	H	H	L	L	H	L	L	H	H	H	H	H	H	H	H	H	A	A	A	A	•	A	A	A
13	H	H	L	L	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
14	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
15	H	L	H	H	H	L	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
16	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	A	A	•	A	A	A	A	A
17	H	L	H	L	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
18	H	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	A	A	A	•	•	A	A	•
19	H	L	H	L	H	L	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
20	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	A
21	H	L	L	H	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
22	H	L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
23	H	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
24	H	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	A	A	A	A	A	A	A	A
25	H	L	L	L	H	L	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
26	H	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	A	A	•	•	•	A	A	•
27	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
28	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	A	A	•	A	•	A	A	A
29	L	H	H	H	H	L	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
30	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	A	A	A	•	•	A	A	•
31	L	H	H	L	H	H	L	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•
32	L	H	H	L	H	L	L	H	H	H	H	H	H	H	H	H	A	A	•	A	A	A	A	A
33	L	H	H	L	L	H	H	L	H	H	H	H	H	H	H	H	A	A	•	•	•	A	•	•

Figure 1.

Programming Services

Bipolar Memory Products

Bipolar LSI Products

Now the Signetics Service center Nördlingen Can Program and Test All Your PROM and Fusible Logic Devices and Cut Your Costs in the Process

Introducing Signetics Programming Services

Signetics' Programming Center is our in-house service for fast, complete programming and testing of fusible products - PROMs (Programmable Read-Only Memories) and IFL devices (Integrated Fuse Logic) and LSI support chips such as I/O Ports.

It's a service that has been growing quietly into one of the most sophisticated, high-quality programming and testing operations running anywhere today.

In short, we can give you total service across the board in fusible products, whether you need only programming or both programming and parametric testing.

A wealth of Experience

Signetics has had years of practical experience in fusible link technology, from development and manufacturing, through programming and testing. In fact, in NiChrome fuse technology alone, we're the leaders, with well over a decade of experience[1]. Since 1978, our IFL (FPLA, FPLS, FPGA) procucts have been coming on strong, and we're reaching the same high levels of consistency in this product line as we have with the PROMs. Today, we're utilizing our experience to help us move efficiently into new frontiers in fusible link technology. Our goal is to lead the way with new and more complex devices.

A strong Commitment

Our commitment to fusible products has grown along with our experience. Over the years we've recognized the potential power and versatility of PROMs and IFLs, and we've paced their growing importance in the marketplace.

With our customers' needs in mind, we're dedicated to maintaining the leadership position in fusible products by emphasizing new process development, highest quality, and ease of use.

Quality is the key

Today's economy is emphasizing the hidden savings to be realized by increasing system reliability. Field failures are expensive due to repair labor, parts cost, and possible loss of future business.

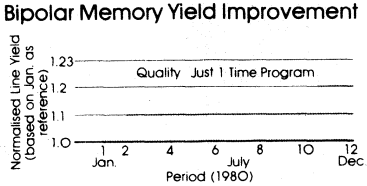
Building in quality is a matter we take very seriously at Signetics. In fact, our history can be told in terms of process improvements and manufacturing refinements. Quality products result from the careful coming together of such process improvements, each making a contribution to the inherent reliability of the IC.

As a part of Signetics' ongoing thrust for ever-higher quality we have put a major corporate drive into place for the 80's. This drive consists of two programs, each of which will significantly benefit our customers.

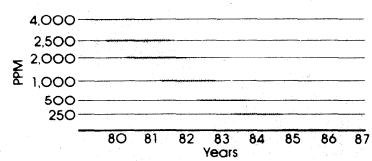
"Doing it Right the First Time" is our motto for a major program for quality and yield improvement. The program was initially implemented in wafer fabrication and began to show results in the third quarter of 1980, as indicated by our Yield Improvement curve.

Our second major program for achieving higher quality is based on our "Quality Circle" teams. The team concept promotes the individual employee's commitment to improved quality, and is dedicated to help eliminate Acceptable Quality Level (AQL) as a quality measure. We have an ambitious goal - going from a quality standard of 2500 PPM (parts per million) today to 250 PPM by 1985 - but we have the programs in place to assure its success. The quality assurance PPM Target Level graph shows the results we expect to achieve.

Bipolar Memory Yield Improvement



Quality PPM Target Level



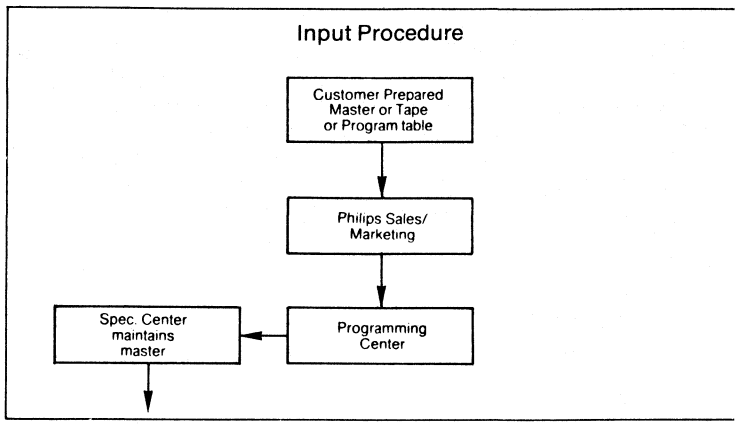
At every step in the evolution of a fusible product, we're ensuring that highest quality standards are met. Whether it's in fabrication (where we use the best in proven processing techniques), at the programming stage (where our experienced teams are using the most modern equipment), or in parametric electrical testing after programming, **Signetics has made the investment for top quality.** That's why the Programming Center provides such a valuable opportunity for you. Signetics' quality saves you money.

[1]US Patent No. 3778886. Mike Shields et al.

Put the Programming Center to Work for You

Our Programming Center can do a lot for you. First, and most obviously, it can reduce your programming costs. No more headaches over the capital equipment and personnel needed for programming and testing.

Another real plus: since you're taking delivery on a fully-programmed and tested product, your administrative and logistics costs are greatly reduced.



Code Input Procedures

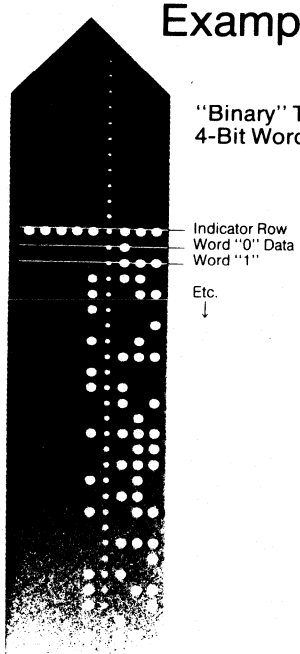
For Efficient, Accurate Programming

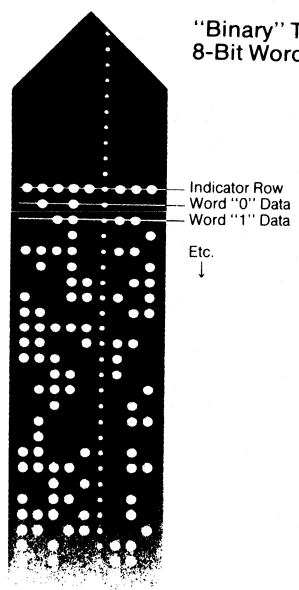
Signetics has the capability to accept the majority of current code input methods. However, the two methods we most prefer are paper tape and/or Master PROMs or IFLs.

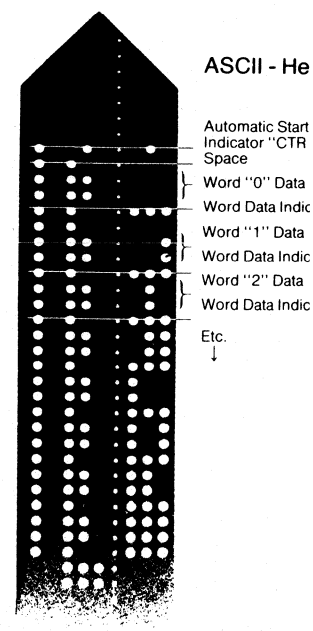
A total of 20 different input formats for PROMs are acceptable. Please contact us for obtaining the detailed specification.

The IFL program table and tape format for each IFL device type is explained in the Integrated Fuse Logic data manual. For I/O Ports please refer to the data sheet.

Examples of PROM tape formats







Signetics Fusible Products

A Growing Selection

Pick a PROM. Or an IFL. We've made selecting the right product easy, thanks to our wide range of parts. The Selection Guide shows you just how many choices you have.

IFL

20-Pin

82S150/151 FPGA (gate array)
82S152/153 FPLA (logic array)
82S154/155 FPLS (logic sequencer)
82S156/157 FPLS (logic sequencer)
82S158/159 FPLS (logic sequencer)

28-Pin

82S102/103 FPGA (gate array)
82S100/101 FPLA (logic array)
82S104/105 FPLS (logic sequencer)

PROM

16-Pin

82S23	256-bit	32 x 8
82S123	256-bit	32 x 8
10139	256-bit	32 x 8
10149	1k	256 x 4
82S126	1k	256 x 4
82S129	1k	256 x 4
82S130	2k	512 x 4
82S131	2k	512 x 4

18-Pin

82S137	4k	1k x 4
82S137A	4k	1k x 4
82S185	8k	2k x 4
82S185A	8k	2k x 4

20-Pin

82S147	4k	512 x 8
82HS195	16k	4k x 4*

24-Pin

82S115	4k	512 x 8 (latched)
82S141	4k	512 x 8
82S2708	8k	1k x 8
82S180	8k	1k x 8
82S181	8k	1k x 8
82S181A	8k	1k x 8
82LS181	8k	1k x 8
82S181B	8k	1k x 8
82S183	8k	1k x 8 (latched)
82S191	16k	2k x 8
82S191A	16k	2k x 8
82S321	32k	4k x 8
82HS321	32k	4k x 8*

8-BIT LATCHED ADDRESSABLE BIDIRECTIONAL I/O PORTS

24-Pin	(addresses 0-255)
8T32	Synchronous TS
8T33	Synchronous OC
8T35	Asynchronous OC
8T36	Asynchronous TC
8X37 . .	series

24-Pin	slim line (addresses 0-15)
8X32	Synchronous TS
8X36	Asynchronous TS
8X42	Synchronous TS 4 input/4 output

* Available soon.

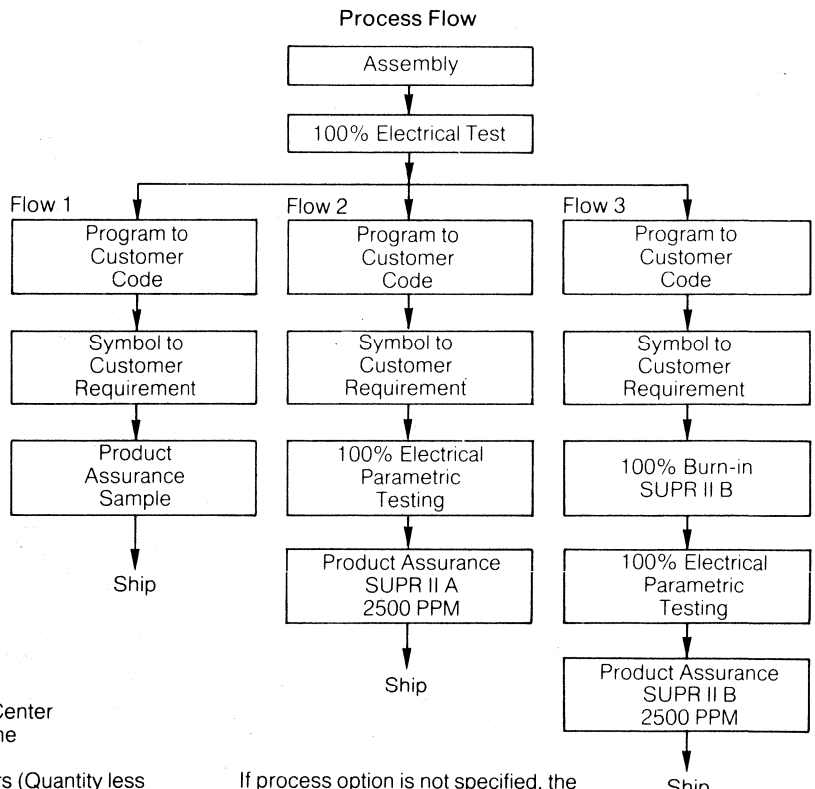
Three Process Options to Choose From

At the Signetics Programming Center all programming is done on precision equipment, monitored and calibrated by our experienced staff. Further, we perform only ONE fusing attempt per link; this promotes the best reliability expectation, since the fusing has not been forced, nor the device reworked.

When you order PROMs or IFLs from Signetics, you'll choose from one of three options, depending on your needs. With **Flow 1**, all your programming is done at the Signetics Programming Center before shipment. With this option, as with the other two, the highest quality standards are enforced.

Flow 2 includes an additional step - 100% parametric electrical testing after programming. This testing procedure is called SUPR II, Level A, and it features an improvement over the usual industry standard parametric tests: **outgoing PPM is guaranteed at 2500, compared to the typical industry guarantee of 6500 PPM.** With this testing alone, at 2500 PPM, you could realize as much as a 20% reduction in board rework!

Flow 3 includes an extra step before parametric electrical testing: an operating burn-in that stresses the device at maximum temperature. This burn-in procedure, in combination with the parametric electrical testing, is called SUPR II, Level B. Its value lies in removing the majority of infant mortality failures.



Programming Center Throughput Time

Flow 1 24 hours (Quantity less than 100)
Flow 2 3 days
Flow 3 depending on quantities, please contact the Philips sales organisation.

If process option is not specified, the programming center will choose flow 1 for quantities of less than 100 pcs. and flow 2 for quantities above 100 pcs.

SIGNETICS INTEGRATED FUSE LOGIC (IFL)

GENERAL
CONTENTS

SELECTION GUIDE

INTRODUCTION

IFL SERIES 20

IFL SERIES 28

IFL PROGRAMMING

MILITARY PRODUCTS

PACKAGE OUTLINES

APPLICATIONS

Electronic components and materials for professional, industrial and consumer uses from the world-wide Philips Group of Companies

Argentina: PHILIPS ARGENTINA S.A., Div. Elcoma, Vedia 3892, 1430 BUENOS AIRES, Tel. 541-7141/7242/7343/7444/7545.
Australia: PHILIPS INDUSTRIES HOLDINGS LTD., Elcoma Division, 67 Mars Road, LANE COVE, 2066, N.S.W., Tel. 427 08 88.
Austria: ÖSTERREICHISCHE PHILIPS BAUELEMENTE Industrie G.m.b.H., Triester Str. 64, A-1101 WIEN, Tel. 62 91 11.
Belgium: N.V. PHILIPS & MBLE ASSOCIATED, 9, rue du Pavillon, B-1030 BRUXELLES, Tel. (02) 242 74 00.
Brazil: IBRAPE, Caixa Postal 7383, Av. Brigadeiro Faria Lima, 1735 SAO PAULO, SP, Tel. (011) 211-2600.
Canada: PHILIPS ELECTRONICS LTD., Electron Devices Div., 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8, Tel. 292-5161.
Chile: PHILIPS CHILENA S.A., Av. Santa Maria 0760, SANTIAGO, Tel. 39-4001.
Colombia: SADAPE S.A., P.O. Box 9805, Calle 13, No. 51 + 39, BOGOTA D.E. 1., Tel. 600 600.
Denmark: MINIWATT A/S, Emdrupvej 115A, DK-2400 KOBENHAVN NV., Tel. (01) 69 16 22.
Finland: OY PHILIPS AB, Elcoma Division, Kaivokatu 8, SF-00100 HELSINKI 10, Tel. 1 72 71.
France: R.T.C. LA RADIOTECHNIQUE-COMPELEC, 130 Avenue Ledru Rollin, F-75540 PARIS 11, Tel. 355-44-99.
Germany: VALVO, UB Bauelemente der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-0.
Greece: PHILIPS S.A. HELLENIQUE, Elcoma Division, 52, Av. Syngrou, ATHENS, Tel. 9215111.
Hong Kong: PHILIPS HONG KONG LTD., Elcoma Div., 15/F Philips Ind. Bldg., 24-28 Kung Yip St., KWAI CHUNG, Tel. (0)-24 51 21.
India: PEICO ELECTRONICS & ELECTRICALS LTD., Elcoma Div., Ramon House, 169 Backbay Reclamation, BOMBAY 400020, Tel. 295144.
Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Elcoma Div., Panim Bank Building, 2nd Fl., Jl. Jend. Sudirman, P.O. Box 223, JAKARTA, Tel. 716 131.
Ireland: PHILIPS ELECTRICAL (IRELAND) LTD., Newstead, Clonskeagh, DUBLIN 14, Tel. 69 33 55.
Italy: PHILIPS S.p.A., Sezione Elcoma, Piazza IV Novembre 3, I-20124 MILANO, Tel. 2-6752.1.
Japan: NIHON PHILIPS CORP., Shuwa Shinagawa Bldg., 26-33 Takanawa 3-chome, Minato-ku, TOKYO (108), Tel. 448-5611.
(IC Products) SIGNETICS JAPAN LTD., 8-7 Sanbancho Chiyoda-ku, TOKYO 102, Tel. (03)230-1521.
Korea: PHILIPS ELECTRONICS (KOREA) LTD., Elcoma Div., Philips House, 260-199 Itaewon-dong, Yongsan-ku, C.P.O. Box 3680, SEOUL, Tel. 794-420
Malaysia: PHILIPS MALAYSIA SDN. BERHAD, No. 4 Persiaran Barat, Petaling Jaya, P.O.B. 2163, KUALA LUMPUR, Selangor, Tel. 77 44 11.
Mexico: ELECTRONICA, S.A. de C.V., Carr. Mexico-Toluca km. 62.5, TOLUCA, Edo. de Mexico 50140, Tel. Toluca 91(721)613-00.
Netherlands: PHILIPS NEDERLAND, Marktgroep Elonco, Postbus 90050, 5600 PB EINDHOVEN, Tel. (040) 79 33 33.
New Zealand: PHILIPS ELECTRICAL IND. LTD., Elcoma Division, 110 Mt. Eden Road, C.P.O. Box 1041, AUCKLAND, Tel. 605-914.
Norway: NORSK A/S PHILIPS, Electronica Dept., Sandstuveien 70, OSLO 6, Tel. 68 02 00.
Peru: CADESA, Av. Alfonso Ugarte 1268 , LIMA 5, Tel. 326070.
Philippines: PHILIPS INDUSTRIAL DEV. INC., 2246 Pasong Tamo, P.O. Box 911, Makati Comm. Centre, MAKATI-RIZAL 3116, Tel. 86-89-51 to 59.
Portugal: PHILIPS PORTUEGESA S.A.R.L., Av. Eng. Duarte Pacheco 6, LISBOA 1, Tel. 68 31 21.
Singapore: PHILIPS PROJECT DEV. (Singapore) PTE LTD., Elcoma Div., Lorong 1, Toa Payoh, SINGAPORE 1231, Tel. 25 38 811.
South Africa: EDAC (Pty.) Ltd., 3rd Floor Rainer House, Upper Railway Rd. & Ove St., New Doornfontein, JOHANNESBURG 2001, Tel. 614-2362/9.
Spain: MINIWATT S.A., Balmes 22, BARCELONA 7, Tel. 301 63 12.
Sweden: PHILIPS KOMPONENTER A.B., Lidingövägen 50, S-11584 STOCKHOLM 27, Tel. 08/67 97 80.
Switzerland: PHILIPS A.G., Elcoma Dept., Allmendstrasse 140-142, CH-8027 ZÜRICH, Tel. 01-488 22 11.
Taiwan: PHILIPS TAIWAN LTD., 3rd Fl., San Min Building, 57-1, Chung Shan N. Rd, Section 2, P.O. Box 22978, TAIPEI, Tel. (02)-5631717.
Thailand: PHILIPS ELECTRICAL CO. OF THAILAND LTD., 283 Silom Road, P.O. Box 961, BANGKOK, Tel. 233-6330-9.
Turkey: TÜRK PHILIPS TICARET A.S., EMET Department, Inonu Cad. No. 78-80. ISTANBUL, Tel. 43 59 10.
United Kingdom: MULLARD LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. 01-580 6633.
United States: (Active Devices & Materials) AMPEREX SALES CORP., Providence Pike, SLATERSVILLE, R.I. 02876, Tel. (401) 762-9000.
(Passive Devices) MEPCO/ELECTRA INC., Columbia Rd., MORRISTOWN, N.J. 07960, Tel. (201)539-2000.
(Passive Devices & Electromechanical Devices) CENTRALAB INC., 5855 N. Glen Park Rd., MILWAUKEE, WI 53201, Tel. (414)228-7380.
(IC Products) SIGNETICS CORPORATION, 811 East Arques Avenue, SUNNYVALE, California 94086, Tel. (408) 739-7700.
Uruguay: LUZILECTRON S.A., Avda Uruguay 1287, P.O. Box 907, MONTEVIDEO, Tel. 91 43 21.
Venezuela: IND. VENEZOLANAS PHILIPS S.A., Elcoma Dept., A. Ppal de los Ruices, Edif. Centro Colgate, CARACAS, Tel. 36 05 11.

For all other countries apply to: Philips Electronic Components and Materials Division, Corporate Relations & Projects, Building BAE-3, P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands, Tel. +31 40 72 33 04, Telex 35000 phtc nl/nl be vec.

A32

Printed in The Netherlands

9398 117 20011