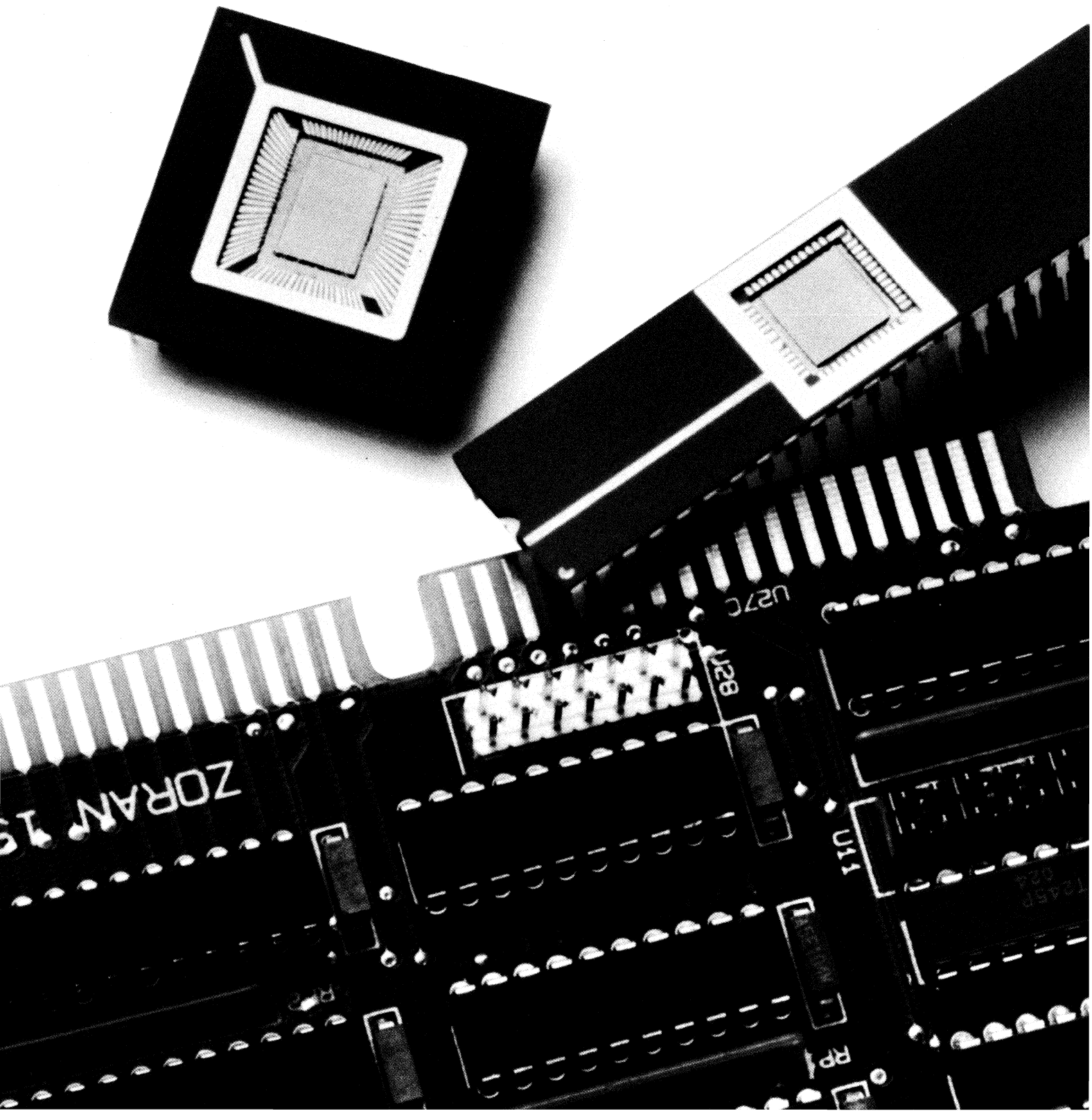


THE LAST WORD IN DSP. ZORAN.

Digital Signal Processors
Data Book



PRESIDENT'S MESSAGE

Dear Customer:

ZORAN Corporation is dedicated to providing system processor solutions for high-performance digital signal processing (DSP) applications. The company designs, manufactures and markets proprietary, monolithic DSP systems processors and associated support tools. By applying its digital signal processing systems and silicon design expertise, ZORAN designs its products top-down with systems requirements foremost in mind. ZORAN's optimized architectures result in powerful, efficient and economical solutions for its customers. A single ZORAN system processor replaces multi-component, board-level products at a fraction of the space, power and cost. ZORAN's highly-integrated, high performance processors result from its distinctive competence in digital signal processing systems, processor architecture, device physics and process technology.

ZORAN's product line consists of two families—Vector Signal Processors (VSP) and Digital Filter Processors (DFP). The VSP family excels at vector operations, including transform computations such as the FFT and the DCT. These products are highly integrated, microprocessor-like systems processors, with instruction sets and I/O buses. The DFP Family excels at sums-of-products operations such as Finite Impulse Response (FIR) filters. These products contain multiple arithmetic processors and support sample rates up to 20 MHz.

Support tools for the VSP family include a software simulator and system development environment, a VSP assembler, and two kinds of full-speed VSP hardware development boards. The DFP family support includes a digital filter design software package and a full-speed DFP hardware development board. These constitute complete software and hardware development environments for each product family.

Our worldwide sales, marketing and applications organizations are dedicated to the support of you, the customer. We appreciate your interest and look forward to supplying your present and future requirements.

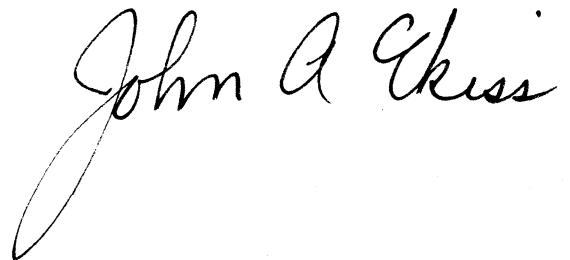
A handwritten signature in black ink that reads "John A. Ekiss". The signature is written in a cursive style with a large, looping initial "J".

TABLE OF CONTENTS

1987 VECTOR SIGNAL PROCESSOR COURSES

VECTOR SIGNAL PROCESSORS

| | |
|--|----|
| VSP Family Overview | 7 |
| Market Applications | 7 |
| Functional Applications | 7 |
| Performance Benchmarks | 7 |
| Description | 7 |
| General Features | 7 |
| Why the VSP is Unique | 7 |
| Introduction | 8 |
| Architectural Features | 8 |
| Instruction Set Summary | 10 |
| Development Support Tools | 10 |
| System Integration | 10 |
| Multiple-VSP Applications | 11 |
| Programming the VSP | 11 |
| Program Examples | 12 |
| Program Size | 13 |
| Block Floating-Point | 13 |
| VSP Application Examples | 14 |
| VSP Simulator | 16 |
| Features | 16 |
| Assembler | 17 |
| VSP Evaluation Package | 17 |
| VSP Development Package | 17 |
| VSP161 Engineering Data Sheet | 19 |

DIGITAL FILTER PROCESSORS

| | |
|---|----|
| DFP Family Overview | 69 |
| Preface | 69 |
| Vector Signal Processors (VSP) | 69 |
| Digital Filter Processors (DFP) | 69 |
| Market Applications | 70 |
| Performance Benchmarks | 70 |
| Functional Applications | 70 |
| DFP Family Members | 70 |
| General Features | 70 |
| Description | 71 |
| Why the DFP is Unique | 71 |
| DFP Configurations | 73 |
| Basic 1-D Finite Impulse Response (FIR) Filtering | 73 |
| Higher Order 1-D Filters | 74 |
| Decimation | 74 |
| Interpolation | 75 |
| Higher Sample Rate Filters | 75 |
| Higher Precision 1-D Filters | 75 |
| Two-Dimensional FIR Filters | 77 |
| DFP Configuration Selection Guide | 78 |
| DFP Development Tools | 80 |
| ZR33481 Digital Filter Processor | 81 |
| Distinctive Features | 81 |
| Applications | 81 |
| Description | 81 |
| Interface Signal Description | 82 |
| Functional Description | 84 |
| DFP Filter Cell | 84 |
| The DFP Output Stage | 86 |
| DFP Arithmetic | 87 |
| Basic FIR Operation | 88 |
| Extended FIR Filter Length | 89 |
| Cascade Configuration | 89 |
| Single DFP Configuration | 90 |
| Extended Coefficient and Data Sample Word Size | 90 |
| Decimation/Resampling | 92 |
| Other DFP Applications | 94 |
| DFP Timing Parameters | 94 |

| | |
|---|-----|
| Absolute Maximum Ratings | 95 |
| Operating Range | 95 |
| AC Characteristics Over Operating Range | 95 |
| Ordering Information | 97 |
| ZR33881 Digital Filter Processor | 99 |
| Distinctive Features | 99 |
| Applications | 99 |
| Description | 99 |
| Interface Signal Description | 100 |
| Functional Description | 101 |
| DFP Filter Cell | 103 |
| DFP Output Stage | 104 |
| DFP Arithmetic | 105 |
| Basic FIR Operation | 105 |
| Extended FIR Filter Length | 107 |
| Cascade Configuration | 107 |
| Single-Processor Configuration | 108 |
| Extended Coefficient and Data Sample Word Size | 109 |
| Decimation/Resampling | 109 |
| Other DFP Applications | 109 |
| DFP Timing Parameters | 111 |
| Absolute Maximum Ratings | 112 |
| Operating Range | 112 |
| DC Characteristics Over Operating Range | 112 |
| AC Characteristics Over Operating Range | 113 |
| Ordering Information | 114 |
| ZR33891 Digital Filter Processor (Preliminary) | 115 |
| Distinctive Features | 115 |
| Applications | 115 |
| Description | 115 |
| Interface Signal Description | 116 |
| Functional Description | 118 |
| DFP Filter Cell | 118 |
| The DFP Output Stage | 120 |
| DFP Arithmetic | 121 |
| Basic FIR Operation | 121 |
| Extended FIR Filter Length | 122 |
| Cascade Configuration | 124 |
| Single DFP Configuration | 125 |
| Extended Coefficient and Data Sample Word Size | 126 |
| Decimation/Resampling | 126 |
| Other DFP Applications | 126 |
| AC Characteristics Over Operating Range | 129 |
| Absolute Maximum Ratings | 130 |
| Operating Range | 130 |
| DC Characteristics Over Operating Range | 130 |
| Ordering Information | 132 |
| ZR33892 Digital Filter Processor | 133 |

DEVELOPMENT TOOLS

| | |
|---|-----|
| Vector Signal Processor Support Tool Environment | 137 |
| Development Environment | 137 |
| Development Board Options | 138 |
| Vector Signal Processor Assembler (VSPA) | 139 |
| Description | 139 |
| Example VSP Program | 139 |
| Explanation of VSP Program | 139 |
| Host Computer Requirements | 139 |
| Vector Signal Processor Simulator (VSPS) | 140 |
| Distinctive Features | 140 |
| Description | 140 |
| Simulator Organization | 141 |
| Using the VSP Simulator | 142 |
| Host Computer Requirements | 143 |
| Feature Variations on Different Computers | 143 |
| Ordering Information | 143 |

| | |
|---|-----|
| VSP Board Support | 143 |
| Updates | 143 |
| VSP Board Packages | 144 |
| VSP Board Software | 144 |
| VSP-161 Programmers Tool Kit | 144 |
| VSP Board Applications | 145 |
| VSP Board Hardware | 145 |
| VSPE Hardware | 145 |
| VSPX Hardware | 147 |
| VSPD Hardware | 148 |
| Host Computer Requirements | 150 |
| Feature Variations on Different VSP Boards | 150 |
| VSP Logic Analyzer Interface Module | 151 |
| Order Information | 151 |
| Digital Filter Processor Board | 152 |
| Distinctive Features | 152 |
| Description | 152 |
| Functional Description | 152 |
| Board Setup Parameters and Options | 154 |
| Sample Rate | 154 |
| Precision and Decimation Options | 154 |
| Number of Taps | 154 |
| Clock Modes and Synchronizing | 154 |
| Operating Environment | 155 |
| Stand Alone Operation | 155 |
| IBM PC/AT or PC/XT with Software Control | 155 |
| PC/AT or PC/XT Operation Without Software Control | 155 |
| Hardware Interface and Requirements | 156 |
| IBM PC Bus (P1) | 156 |
| The DFPB Edge Connector (P3) | 156 |
| Board Characteristics | 157 |
| Reference Documentation | 158 |
| Software Requirements | 158 |
| Product Description/Order Information | 158 |
| Digital Filter Software (DFPS) | 159 |
| Distinctive Features | 159 |
| Description | 159 |
| Filter Design | 159 |
| Signal Generation | 159 |
| Signal Filtering | 160 |
| Transforms | 160 |
| Outputs and Displays | 160 |
| Quantization | 160 |
| PROM Programming | 160 |
| Menu Interface | 160 |
| DFPS Function Menu | 161 |
| Command Mode Interface | 162 |
| Hardware Requirements | 162 |
| Ordering Information | 162 |

| | |
|---|-----|
| Hardware Accessories | 162 |
| Operating System Requirements | 162 |
| Updates | 162 |
| APPLICATIONS | |
| VLSI Brochure | 165 |
| Markets | 165 |
| Functions | 165 |
| Applications | 165 |
| User Benefits | 165 |
| Key Features | 165 |
| Introduction | 165 |
| 2-D Spatial Domain Processing | 166 |
| 2-D Spatial Domain Convolution or Correlation | 166 |
| Edge Detection | 167 |
| 1-D Finite Impulse Response (FIR) Filters for | |
| Digital Video | 169 |
| Decimation and Interpolation | 169 |
| Pyramid Processing | 170 |
| Frequency Domain Processing | 171 |
| Processing Unique to the Frequency Domain | 171 |
| The Fast Fourier Transform | 171 |
| 2-D Fast Convolution or Correlation in the Frequency Domain Using the FFT | 172 |
| Homomorphic or Generalized Linear Filtering | 174 |
| Image Compression Using the 2-D Fast Cosine Transform | 174 |
| Typical Applications | 177 |
| Machine Vision | 177 |
| Telecommunications | 177 |
| Medical Imaging | 177 |
| Digital Video | 177 |
| Electronic Cameras | 178 |
| Military Reconnaissance | 178 |
| Application Listings | 179 |
| Digital VLSI for Broadcast Video | 179 |
| VLSI for Industrial Process Monitoring | 180 |
| VSP Remote Data Collection and Reduction | 181 |
| VLSI Digital Signal Processing Solutions for Medical Equipment | 182 |
| Digital VLSI for Machine Vision | 183 |
| GENERAL INFORMATION | |
| Facilities | 187 |
| Reliability and Quality Assurance | 188 |
| General | 188 |
| Quality Planning | 188 |
| Organization | 188 |
| Implementing Reliability and Quality Assurance at Zoran | 189 |
| Product Reliability and Quality | 189 |
| Lifetest Data at 150°C | 189 |
| Zoran Product Summary Flow | 190 |

Copyright © 1987 ZORAN Corporation, Santa Clara, California. All rights reserved. Contents of the publication contain proprietary information and may not be reproduced in any form without written permission of ZORAN Corporation.

The material in this manual is for information only. Zoran Corporation assumes no responsibility for errors or omissions and reserves the rights to change, without notice, product specifications, operating characteristics, packaging, etc. Zoran Corporation assumes no liability for damages resulting from the use of information contained in this document.

Vector Signal Processor, VSP, Vector Signal Processor Simulator, and VSPS are trademarks of Zoran Corporation.

DEC, VMS, ULTRIX, and VT100 are trademarks of Digital Equipment Corporation.

IBM, PC, PC/XT, and PC/AT are trademarks of International Business Machines.

MS-DOS is a trademark of Microsoft Corporation.

1987 VECTOR SIGNAL PROCESSOR COURSES

VECTOR SIGNAL PROCESSOR (VSP) AND DEVELOPMENT TOOLS

THREE DAY INTENSIVE LEARNING SESSION*

OPTIONAL FOURTH DAY FOR APPLICATION ASSISTANCE.

MAY 12-14, 1987

JULY 28-30, 1987

SEPT 29-OCT 1, 1987

DEC 8-10, 1987

DESCRIPTION

ZORAN Corporation is offering its customers a three-day technical training course on the Vector Signal Processor (VSP) and VSP development tools. The course will provide the customer with experience using the VSP Systems Processor, Simulator, and development boards through a series of lectures and hands-on tutorial sessions. After attending the course, participants will be able to apply the VSP to their unique design problems and use the powerful development tools as an aid in the design process.

The primary objectives of this course include: understanding the architecture and operations of the VSP; familiarization with a range of hardware and software interfacing techniques; and providing examples which illustrate how to program the VSP using its high-level instruction set. Sessions on FFT theory and transform domain processing using the VSP are also covered.

In addition, the course offers laboratory sessions on the use of the development tools that will enhance the participants' proficiency in the use of ZORAN's design tools and aids. DSP architecture and systems engineers will also have the opportunity to explore and evaluate new applications using the ZR34161 VSP.

CONTENTS

FFT Theory

VSP Hardware

VSP Memory and Registers
VSP Interface
Instruction Set
Scaling
Overlapped Operations
System Architecture and Timing
VSP Arithmetic

VSP Simulator

VSPS Menus and Signal Generator Lab
VSPS Instruction Tutorial Lab
Simple VSP Programs and the VSPS
High Level Languages
VSPS Queueing and Timing

VSP Development Boards

VSP Assembler

VSP/VSPS COURSE OBJECTIVES

- Provide a detailed description of the VSP architecture and thoroughly understand its operation.
- Describe a range of hardware and software interfacing techniques.
- Instruct how to program the VSP using its high level instruction set.
- Demonstrate and understand the use of the VSP Simulator in the designing of VSP-based products.
- Enhance proficiency in the use of ZORAN'S design tools and aids.
- Enable DSP architecture and systems engineers to explore and evaluate new applications for the ZR34161 VSP.

FEE

\$1,000 per attendee.

LOCATION

ZORAN Corporation Headquarters
3450 Central Expressway
Santa Clara, California 95051

CONTACT

Course Registrar
Tel: (408) 720-0444
ELN: 62942343

*PREREQUISITES

Basic DSP experience, prior exposure to FFT techniques and familiarity with contents of VSP engineering data sheets.

VECTOR SIGNAL PROCESSORS



MARKET APPLICATIONS

- Radar/Sonar
- Image Processing
- Communications
- Image/Data Compression
- Spectral Analysis
- Speech Processing

FUNCTIONAL APPLICATIONS

- 1-D and 2-D FFT
- 1-D and 2-D DCT
- Auto/Cross Correlation
- Convolution/Filtering
- Modulation/Demodulation
- Vector Multiply/Add

PERFORMANCE BENCHMARKS

| Function | time (μsec) |
|---|--------------------------|
| 1024-point block-floating complex FFT | |
| —Single VSP | 3300 |
| —Two parallel VSPs | 1900 |
| —Four parallel VSPs | 1200 |
| 1024-point integer complex FFT | 2600 |
| 128-point block-floating complex FFT | 237 |
| 16 x 16 FCT | 1100 |
| 8 x 8-point 2-D complex FFT | 164 |
| 64 x 64-point 2-D block-floating complex FFT | 18000 |
| 128-point x 128-point complex vector multiply | 53 |
| 128-point magnitude-square/accumulate | 26 |
| 128-point complex modulation or demodulation | 52 |
| 4 x 4 matrix multiplication | 33 |

DESCRIPTION

The ZR34161 Vector Signal Processor (VSP™) is a member of Zoran's family of high-performance Systems Processors™. The VSP is a 16-bit processor which introduces algorithm-level and vector-oriented instructions to digital signal processing (DSP) system design. It functions as the key element in high-performance DSP applications. In coordination with a host controller, the VSP performs computation-intensive tasks while making minimum demands on host resources and system I/O capacity.

The block diagram shown in Figure 1 illustrates a system which is useful in a number of different applications, such as doppler frequency estimation or "zoom" FFT spectral analysis. It also serves to illustrate the processing power of the VSP. This type of high-performance system is implemented very efficiently by the VSP; most of the blocks in the diagram can be implemented using a single instruction.

WHY THE VSP IS UNIQUE

The VSP achieves its high performance through a number of unique architectural features. Firstly, the VSP uses a "high-level", vector-oriented instruction set; for instance, "FFT" is a single instruction within the VSP. These types of instructions can greatly simplify the amount of programming effort required to implement signal processing algorithms. Secondly, the VSP provides a block floating-point arithmetic capability which will help retain the original dynamic range of an input signal when performing FFT operations. This typically provides dynamic-range performance significantly greater than that of 16-bit fixed-point integer machines. Finally, the nature of its architecture and instruction set allows additional VSPs to be paralleled on the same bus to increase the signal processing throughput well beyond that of a single VSP.

GENERAL FEATURES

- High-performance 16-bit digital signal processor
- Architecture optimized for DSP operations
- High-level Vector-oriented instructions
- Block floating-point arithmetic for FFT
- Concurrent I/O and arithmetic processing
- Easily paralleled for greater throughput
- Fabricated in two micron DLM CMOS
- < 300mW power dissipation
- Powerful hardware and software development tools

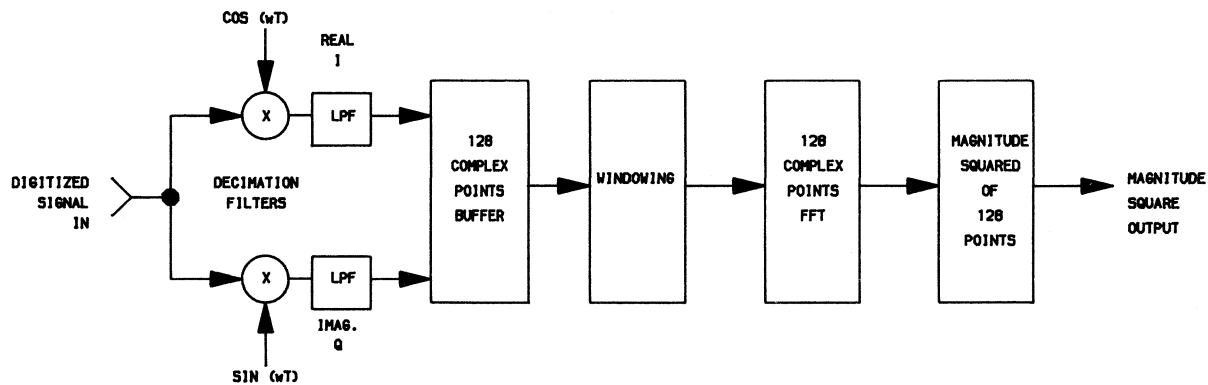


FIGURE 1. BLOCK DIAGRAM OF GENERAL COMPLEX DEMODULATION PROCESS IMPLEMENTED EFFICIENTLY BY THE VSP.

INTRODUCTION

While the signal processing performance of the VSP is quite high, the concept of its operation is straightforward. Data typically is loaded to the VSP as a vector of up to 128 complex samples, operated on by one or more of the VSP's 15 arithmetic instructions, and stored back to external memory. Operating on vectors in external and/or internal RAM, the VSP instruction repertoire includes FFT, vector addition/multiplication, complex conjugate, magnitude square, and modulate/demodulate. These algorithm-level instructions result in compact and highly efficient programs. Block floating-point arithmetic for the FFT gives the VSP superior performance on large transforms as compared to ordinary 16-bit fixed-point processors.

Zoran's focus on providing a complete system solution is reflected in the available hardware and software development tools. A number of software tools are available, including a software simulator and assembler. The VSP Simulator software provides an environment for exploring and learning the capabilities of the VSP, and for testing the functional aspects of an application by generating, processing, and plotting test signal results. Plug-in boards and debugger software work with both the simulator and assembler to allow programs written for the VSP to be run in real-time.

ARCHITECTURAL FEATURES

The VSP architecture contains three primary functional units as shown in Figure 2: the bus-interface unit (BIU), execution unit (EU), and memory and registers. The BIU handles all communication between the VSP internal resources and the external environment. The EU handles all arithmetic processing. The memory and registers are shared between the BIU and EU, and contain the instruction FIFO, operating registers, and data RAM.

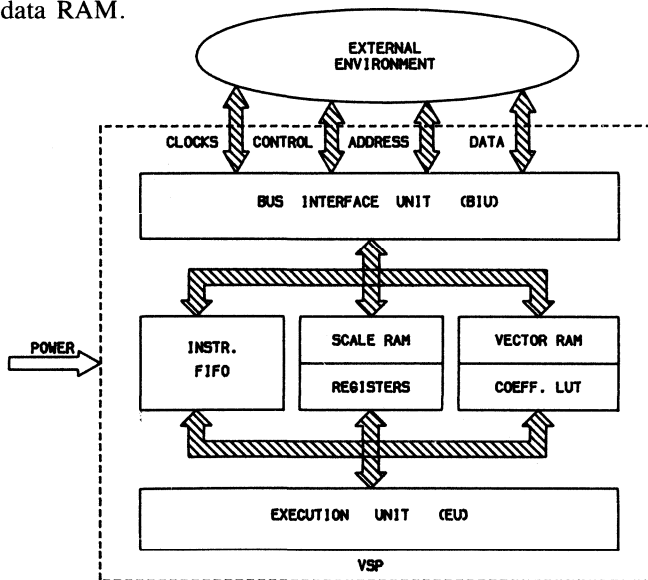


FIGURE 2. VSP FUNCTIONAL ARCHITECTURE.

Bus-Interface Unit:

- DMA (block transfer) capability
- Separate 64K-word data and program address spaces
- Bidirectional 16-bit address, data, and control buses

Execution Unit:

- Architecture optimized for general vector and FFT butterfly operations
- Internal 17-bit arithmetic precision
- $17 \times 17 = 17$ -bit multiplier
- Two adder/subtractors
- Two 25-bit accumulators

Memory and Registers:

- Internal 128×38 -bit complex-word RAM
- Four-instruction internal FIFO
- Internal sin/cos look-up table supports up to 1024-point FFT

Bus-Interface Unit: The VSP is a two-cycle processor, using a maximum 20MHz input clock (CLK). The CLK signal is internally divided by two to create a 10MHz clock which is provided on an output pin (ICLK) and also clocks all internal activity.

The BIU has a familiar microprocessor-style interface as seen in the logical pinout of Figure 3. It is designed to interface easily to a wide variety of potential host microprocessors and controllers. All instructions to be executed and data to be processed by the VSP first pass through the BIU into the appropriate memory or register location.

The normal mode of program execution is for the VSP to be given control of the bus, at which time it will fetch its own instructions and data. It requests the bus by asserting the $\overline{\text{BRQ}}$ (bus request) pin, and takes control only after receiving an active $\overline{\text{BACK}}$ (bus acknowledge) signal in response. Once granted control of the bus, the VSP uses its DMA capability to read/write

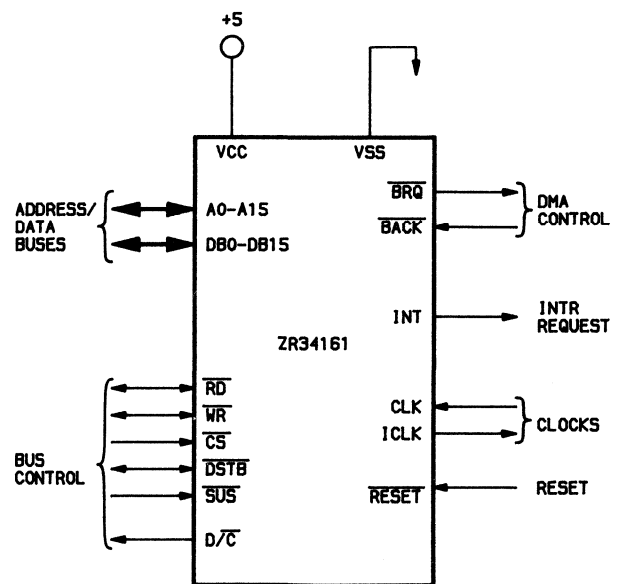


FIGURE 3. VSP LOGICAL PINOUT.

blocks of external memory. Use of the D/\overline{C} (Data/Code) pin allows the VSP to access separate 64K word memory spaces: one for data, the other for instructions. When fetching instructions, the D/\overline{C} pin is LOW; when fetching data, D/\overline{C} is HIGH.

Three control signals (\overline{RD} , \overline{WR} , and \overline{DSTB}) are bidirectional in nature, as controlled by the state of the \overline{CS} pin. As outputs, the VSP uses these signals for controlling external memory when fetching instructions and data. As inputs, they allow a host processor to access the internal memory and memory-mapped registers of the VSP. To address its internal memory, the host uses the \overline{CS} (chip select) pin to establish the address pins as inputs.

The INT (interrupt) output pin from the VSP provides an indication to the host processor that one or more of the five internal interrupt bits has been set. The host should read the VSP status register to interpret the source of the interrupt, which in turn clears the INT pin.

Execution Unit: The execution unit (EU), shown in Figure 4, contains the hardware multiplier, adders, and accumulators for performance of all arithmetic operations. In particular, the EU is organized for efficient vector and FFT-butterfly computations. This is illustrated by the VSP processing rate of 2.5 million FFT-butterfly operations per second. For the FFT, MODLT, and DEMO instructions (Figure 5) the required sine/cosine values

are obtained from the internal look-up table (LUT). The LUT contains 256 17-bit values corresponding to the first quarter cycle of a cosine wave. Together with appropriate logic, this LUT provides all complex weighting factors required for FFTs of up to 1024 points in length.

Memory and Registers: The VSP contains a single 16-bit mode register which programs its operating characteristics. The mode register bits are programmed to provide options such as: selecting the speed of external memory, the number of RAM sections in internal memory, and individual enabling of the VSP interrupt bits.

Internal VSP RAM is structured as 128 38-bit complex words, and optionally as two separate sections of 64 complex-words each. Each complex word consists of 19-bit real and imaginary halves. Many of the VSP's instructions affect not only the full complex word, but also the real or imaginary halves separately. Data values are transferred between the VSP and external RAM as 16-bit quantities, but internally the VSP maintains 17 bits of precision (the remaining 2 out of the 19 bits are used for block floating-point arithmetic).

When the internal RAM is logically divided into two 64 complex-word sections, the activities of the BIU and the EU may be overlapped. Specifically, while the EU is performing arithmetic operations on the data in one RAM section, the BIU can transfer data between external RAM and the internal RAM

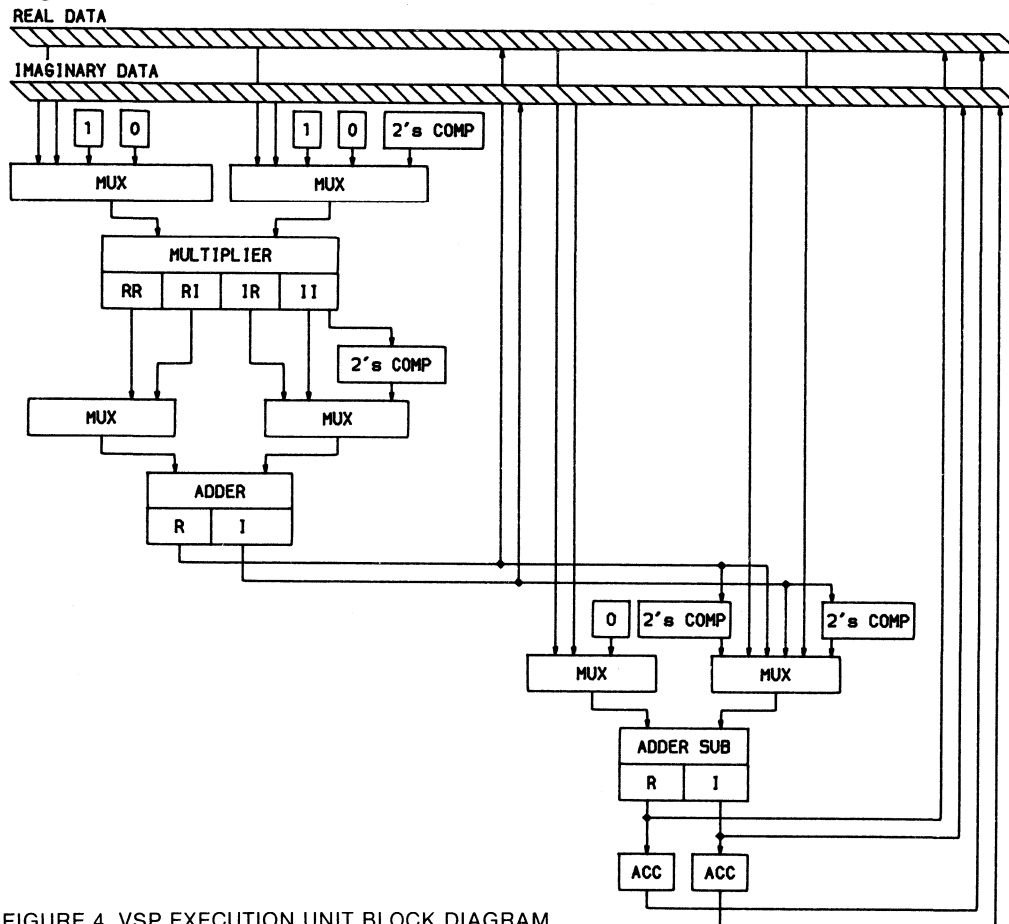


FIGURE 4. VSP EXECUTION UNIT BLOCK DIAGRAM.

section not being used by the EU. This overlapping technique provides increased throughput, particularly when slow external memory is being used. (External memory can have an access time that requires one or two ICLKs for data transfer, as programmed in the mode register.)

The VSP executes instructions out of its instruction FIFO, in which up to four instructions may be buffered. The FIFO can be automatically replenished by the BIU fetching its own instructions, or can be controlled entirely by the host.

INSTRUCTION SET SUMMARY

The 23 instructions of the VSP are listed in Figure 5. They are shown grouped into their four logical categories. By selecting values for the parameters associated with these instructions, it is possible to achieve a wide variety of results. For example, when loading a block of data, the parameters associated with the LD instruction allow demultiplexing of multichannel data, storage into the real or imaginary column of the VSP RAM, interpolation of zero values, and bit-reversal reordering for subsequent FFT processing. Similar flexibility is associated with the ST and STB instructions. The STI instruction writes the contents of the VSP's internal registers, including the two accumulators, to external memory. Of the eleven instructions that only operate on internal RAM contents, three—FFT, DEMO, and MODLT—also use the internal LUT. Two special instructions provide support for the vector scaling operations

used in performing large FFTs in block floating-point arithmetic. The four vector add/multiply instructions that use both internal and external memory do not affect external memory contents; only the store instructions do that. The instructions in the control category affect program flow; the NOP and HLT functions are obvious, and the JMPI instruction provides a program looping capability.

DEVELOPMENT SUPPORT TOOLS

Program development for the VSP is supported through a variety of software and hardware tools. The VSP simulator (VSPS) software is a comprehensive tool for assessing the performance of an entire signal processing system, including the relationship between the VSP and the external environment. The VSPS contains facilities for generating test signal data, executing VSP instructions through a tutorial process, and outputting tabular or graphical data. All internal RAM and register contents may be monitored during VSP program execution, and clock cycles counted for timing purposes. The VSPS is menu-driven, with a command mode for expert use. VSP code can be combined (intermixed) with C or FORTRAN code that simulates external system operations, and linked to the VSPS for subsequent execution. The evaluation board for the IBM PC/AT can be used as an accelerator for the VSPS or in a stand-alone fashion supported by an assembler, debugger, and utility software. A development board, also for the PC/AT, has hardware breakpoint control and I/O buffers to allow real-time debugging of VSP-based system operation.

SYSTEM INTEGRATION

The VSP is designed as a "loosely-coupled" peripheral processor under the control of either a host microprocessor or simple state machine controller. "Loosely-coupled" means that the host can direct the VSP to perform signal processing functions without tight control over VSP activity. For instance, a host will usually provide a starting address to the VSP which points to the beginning of a signal processing algorithm in external memory. The VSP will fetch its own instructions and data for executing the algorithm and inform the host when it has completed the task.

Integration of the VSP into a system requires a minimum of communication between the system host and the VSP. Figure 6 shows how the VSP fits into a bus-oriented system, with the host controlling VSP access to the bus by means of the \overline{BRQ} and \overline{BACK} pins. When the host requires the VSP to execute a program, the starting address of the program can be written to the VSP, which then begins fetching instructions and associated data from system RAM and ROM as appropriate. All VSP access to the bus is thereafter managed by the bus arbitration mechanism. VSP programs terminate with a HLT instruction, and the host can be notified of this condition through the INT pin. In this mode of operation, the VSP is a peripheral processor which can perform significantly long computational tasks without close supervision by the host.

| | | | |
|--|---|-------|--|
| DATA MOVEMENT | [| LD | — Load |
| | | LDSM | — Load Scale/Mode Registers |
| | | ST | — Store |
| | | STB | — Store Backwards |
| | | STI | — Store Information Registers |
| INTERNAL RAM OPERATIONS | [| ABS | — Absolute Value |
| | | ACCI | — Accumulate Imaginary |
| | | ACCR | — Accumulate Real |
| | | CMCN | — Complex Conjugate |
| | | CMLT | — Cross Multiply/Accumulate |
| | | DEMO | — Demodulate |
| | | FFT | — Fast Fourier Transform |
| | | MGSQ | — Magnitude Square/Accumulate |
| | | MODLT | — Modulate |
| | | SCL | — Scale |
| | | SCLT | — Scale Literal |
| EXTERNAL/ INTERNAL RAM OPERATIONS | [| ADDC | — Vector Add Complex |
| | | ADDR | — Vector Add Real |
| | | MLTC | — Vector Multiply Complex/ Accumulate |
| | | MLTR | — Vector Multiply Real/Accumulate |
| CONTROL | [| HLT | — Halt |
| | | JMPI | — Jump Indirect |
| | | NOP | — No Operation |

FIGURE 5. VSP INSTRUCTION SET.

MULTIPLE-VSP APPLICATIONS

The unique nature of the VSP architecture and instruction set support applications which require signal processing performance beyond that which can be achieved with a single VSP. For instance, using two VSPs operating in parallel on a single bus, a 1024-point complex block floating-point FFT can be computed in 1900 μsec . Four VSPs can compute the same operation in 1200 μsec .

Because each VSP instruction operates on vectors, or arrays of data, the execution-time of each instruction may be relatively long. For instance, it takes the VSP 1850 clocks to execute a 128-point FFT from a single instruction. Because this operation occurs entirely within the VSP, the data bus is free for additional VSPs (or other peripherals) to use.

The hardware architecture to implement this type of system is very similar to the interface shown in Figure 6. However, the single VSP in the figure would be replaced by multiple VSPs. The bus arbiter would then be responsible for arbiting the bus between all of the VSPs connected in the system.

PROGRAMMING THE VSP

The flexibility of VSP instructions can be illustrated by discussing the use of the LD, ST, FFT and MGSQ instructions to build a simple program for generating a magnitude-squared spectrum. The instruction parameters are listed in Figure 7, and a simplified listing of the program is in Figure 8(a). Each of these instructions can process a vector of up to 128 complex values. Before discussing the use of these instructions, note that the EI parameter is common to all of them. Setting the EI parameter will allow an interrupt signal to be generated when the instruction has finished execution.

For the LD instruction, the NMPT parameter sets the vector size, and the RS parameter selects internal RAM section 0 or 1 if the RAM is logically partitioned into two sections (done by setting a bit in the mode register). The vector is always loaded

into the lowest NMPT locations of the selected internal RAM section. Individual locations of the internal RAM are not addressable using the VSP instructions; they are, however, accessible by the host.

The data is located in external RAM starting at physical word address MBA. The number of words read from external RAM is conditioned by the parameter MDF which specifies whether only the real, imaginary, or full complex word of the internal RAM is to be affected. If only the real or imaginary part of the RAM receives data, then NMPT words are read from external RAM. For a complex-valued vector, $2 \cdot \text{NMPT}$ words will be read, where the VSP assumes the order in external RAM is real-imaginary.

The parameters MBS and MSS allow demultiplexing of multichannel data by selecting blocks of MBS data values spaced MSS values apart. INTRP allows up to three zero-valued elements to be inserted between each sample; ZR can put zeros in either the real or imaginary part of the internal RAM; and ZP can make the last half of the length-NMPT vector zero-valued. The ST instruction has similar features, as indicated by the common parameters in Figure 7.

For use with the FFT instruction, both LD and ST have the RV parameter which allows bit-reversal reordering of data values. In combination with the MBS and MSS parameters, this also allows bit-reversal reordering of blocks, or data within blocks—features that are required for performing large or multidimensional FFTs. For the FFT instruction, NMBT is the total number of data values to be processed. If FSIZ, the size of the FFT, is smaller than NMBT, then multiple small FFTs can be performed in a single instruction. With NMBT limited to 128, it is possible to do two 64-point, four 32-point, etc. transforms at once. Parameter I selects forward or inverse transform, R indicates if the input data is bit-reversed in order, and AS specifies fixed-point or block floating-point arithmetic.

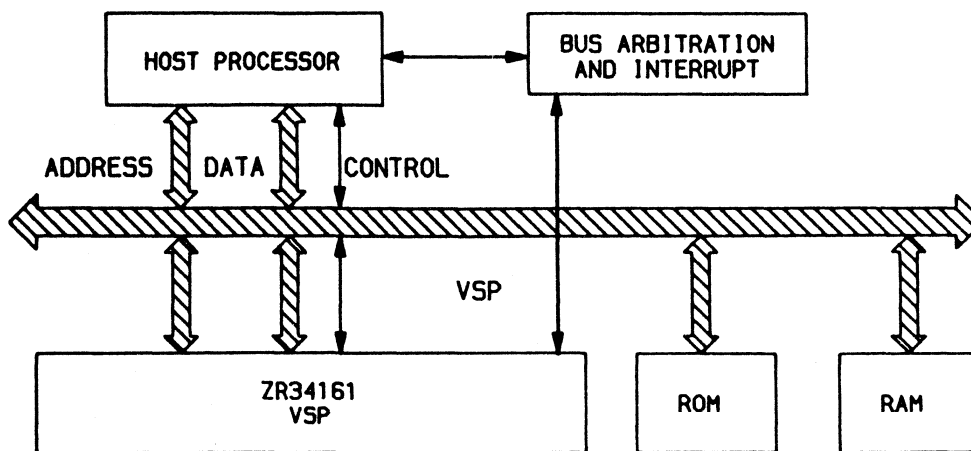


FIGURE 6. VSP SYSTEM INTERFACE.

The FFT instruction also has three parameters which play a role in performing large FFTs: the FPS and LPS parameters define the separation between the two data points which are combined in a butterfly operation at the first and last passes through the data, respectively; and RBA selects the starting location in the sine/cosine LUT for obtaining complex weights. Proper use of these parameters does require analysis of the FFT algorithm; however, Zoran supplies a software library for transforms up to length 16384.

By comparison with the LD, ST, and FFT instructions, the MGSQ instruction has few parameters, and only one is different from those discussed already. Normally the squares of the real and imaginary parts are added and stored in the real part, scaled down by 1/2 to avoid overflow, with the total accumulated in the real accumulator. If ADF is zero then only the accumulator is affected—the values in the internal RAM are unchanged.

| <u>LD (LOAD)</u> | <u>ST (STORE)</u> | <u>FFT (FAST FOURIER TRANSFORM)</u> | <u>MGSQ (MAGNITUDE SQUARE/ ACCUMULATE)</u> |
|----------------------|-----------------------|---|--|
| NMPT | NMPT | NMBT | NMPT |
| RS | RS | RS | RS |
| INTRP | | EI | ADF |
| EI | EI | FPS | EI |
| MBS | MBS | LPS | |
| MSS | MSS | RBA | |
| RV | RV | FSIZ | |
| MDF | MDF | AS | |
| ZR | | ! | |
| ZP | | R | |
| MBA | MBA | | |

FIGURE 7. EXAMPLE OF VSP INSTRUCTION PARAMETERS.

| | | | | | |
|------|----------|------|--------|--------|----------|
| LD | NMPT:128 | RS:0 | MDF:2 | ZR:1 | MBA:0; |
| FFT | NMBT:128 | RS:0 | FPS:64 | LPS:1; | |
| MGSQ | NMPT:128 | RS:0 | ADF:2; | | |
| ST | NMPT:128 | RS:0 | RV:1 | MDF:2 | MBA:256; |

(A) MAGNITUDE-SQUARE SPECTRUM.

DEFAULT NMPT:128 RS:0 INTRP:0 EI:0 MDF:3 ZR:0 ZP:0;

| | | | | | |
|----------|----------|----------|---------|---------|---------------|
| DEFAULT | NMBT:128 | FSIZ:128 | AS:1 | ! :0 | R:0; |
| (0) LDSM | NMPT:1 | UP:1 | MD:1 | | MBA:16383; |
| (1) LD | NMPT:128 | MBS:1 | MSS:2 | RV:0 | MBA:IN; |
| (2) FFT | NMBT:128 | FPS:64 | LPS:1 | RBA:0; | |
| (3) ST | NMPT:128 | MBS:1 | MSS:2 | RV:0 | MBA:IN; |
| (4) LD | NMPT:128 | MBS:1 | MSS:2 | RV:0 | MBA:IN + 2; |
| (5) FFT | NMBT:128 | FPS:64 | LPS:1 | RBA:0; | |
| (6) ST | NMPT:128 | MBS:1 | MSS:2 | RV:0 | MBA:IN + 2; |
| (7) LD | NMPT:128 | MBS:128 | MSS:128 | RV:0 | MBA:IN; |
| (8) FFT | NMBT:128 | FPS:1 | LPS:1 | RBA:0; | |
| (9) ST | NMPT:128 | MBS:1 | MSS:2 | RV:1 | MBA:OUT; |
| (10) LD | NMPT:128 | MBS:128 | MSS:128 | RV:0 | MBA:IN + 256; |
| (11) FFT | NMBT:128 | FPS:1 | LPS:1 | RBA:14; | |
| (12) ST | NMPT:128 | MBS:1 | MSS:2 | RV:1 | MBA:OUT + 2; |

(B) 256-POINT COMPLEX FFT.

FIGURE 8. VSP PROGRAMS FOR FFT.

PROGRAM EXAMPLES

Figure 8 lists two programs—one simple, the other more involved—both concerned with using the instructions discussed above. The instruction format includes only those parameter values essential to reading the listing, and also those parameters which change value in succeeding instructions. The remaining parameters are set with the “DEFAULT” directive.

In Figure 8(a) the magnitude-squared spectrum for 128 real-valued time samples is computed. That the input samples are real is indicated by MDF:2 in the LD instruction. The 128 real samples are located in external memory starting at physical address zero, and are loaded in normal order (RV:0). A full 128-point FFT is performed, as indicated by the FPS value of 64 and the LPS value of 1. In the FFT instruction, AS:1 selects fixed-point arithmetic with a scaling by 0.5 (right-shift by one bit) at each stage. The magnitude-squared values are stored in the real part of the internal RAM (ADF:2) by the MGSQ instruction. The ST instruction puts the spectral values into external RAM starting at address 256, and does a bit-reversal reordering in the process (RV=1).

To facilitate discussion of the program example in Figure 8(b), the key instructions have been numbered; this numbering is not used for actual VSP programs. The program for the 256 complex-point FFT performs two separate 128-point FFTs and then combines the results to obtain the spectral values for the full 256-point transform. The procedure is based upon converting an N-point transform into two N/2-point transforms by grouping the input samples into even- and odd-indexed sets—a procedure known as decimation-in-time, or DIT. For simplicity, block floating-point scaling is not performed; instead, a fixed shift by one bit is performed in each pass (AS:1).

Instruction (0) simply programs the VSP mode register to the desired contents. Instruction (1) loads every other sample starting at the beginning of the vector—external memory location IN—to give the even-indexed samples. Instruction (4) loads the odd-indexed samples starting at IN+2 (complex samples occupy two successive memory locations in external RAM). Each 128-point vector is transformed and stored back to external RAM, overwriting the time samples. This process is done with instructions (1) to (6). When the transform values are stored back to external RAM they are not bit-reversed reordered; this allows adjacent elements in each 128-point vector of transform values to be combined using a single butterfly operation.

Instructions (8) and (11) have FPS=LPS:1 to indicate the performance of a single stage of butterflies operating on adjacent samples in the vector—a sample separation of 1. The final 128-point vector requires a different set of complex weights from the previous vector; hence, in instruction (11) an offset into the LUT—RBA:14—is specified in order to get odd powers

of the 256th root of unity. The transform values are stored in bit-reversed order in a 256-sample block starting at address OUT—instructions (9) and (12). Because each 128-point vector is separately reordered, the overall FFT operation is not an in-place procedure. The complete 256-point complex FFT computation requires less than 650 microseconds.

The use of IN and OUT for memory base reference is an assembler feature—the assembled program will have direct memory references. To use the 256-point FFT program to operate on different segments of external memory would require either modification of addresses by the host or duplication of the program code for each segment. The second option is actually very reasonable because of the small size of VSP programs. In a situation where the host maintains double buffers for both input and output, it would only be necessary to have two copies of the program—one for each pairing of an input and output buffer. The first option, host modification of data addresses in VSP programs stored in external memory, is aided by the fact that address references in VSP instructions always occupy the last word of the three-word instruction.

PROGRAM SIZE

Conventional DSP microprocessors have scalar-oriented instruction sets which need loop-control code in order to operate on the blocks of data processed in the majority of DSP algorithms. Because DSP processors are optimized to perform multiply-accumulate operations, the time required for loop-control arithmetic is frequently a very significant percentage of the total execution time. For this reason, time-optimized programs are written as straight-line code. Naturally, this results in programs of considerable size.

Table 1 compares statistics for FFT programs on the VSP and the TMS32020. The first two entries in the table correspond

to the programs just discussed. The advantages of vector-oriented instructions over scalar-oriented instructions with loop control are clearly evident. However, even when straight-line code is employed, the VSP still enjoys a significant speed advantage. Of course, the size of the straight-line coded 256-point FFT on the TMS32020 is orders of magnitude greater than the VSP program size. The next generation scalar-oriented processors may have a factor of two improvement in execution time, but program size will remain substantially the same.

BLOCK FLOATING-POINT

The highly repetitive and sequential operations of the FFT algorithm present a well-known source of overflow and roundoff noise accumulation problems. In ordinary fixed-point arithmetic the strategy for countering overflow is a scaling of each sample by 0.5 at each stage. This procedure is not sufficient to guarantee no overflows, and in many cases results in unnecessary loss of accuracy. The VSP allows fixed-point computation with fixed scaling by 0.5 in the FFT, but more importantly the VSP supports block floating-point (BFP) arithmetic for FFT computation. In BFP arithmetic an exponent, or scale factor, is associated with a block or vector of data, rather than with each element as in full floating-point arithmetic.

The worst-case growth in value after a butterfly operation is about 1.5 bits. Both real and imaginary halves of the VSP RAM contain 2 bits to hold this overflow. After the full complex vector has been processed in a stage of the FFT there will be a maximum overflow count for the vector. When the vector is accessed again, either for the next FFT stage or for storage to external RAM, each element of the vector is right-shifted by 0, 1, or 2 bits in accordance with the overflow count. At the conclusion of the FFT the total number of shifts performed is placed in one of the 4-bit nibbles of the scale register, which can be written to external RAM. The maximum expected shift

| | Program Memory Requirement (Words) | | VSP Advantage | Execution Time (millisec) | | VSP Advantage |
|-------------------------------------|---------------------------------------|---------|---------------|---------------------------|---------|---------------|
| | TMS32020 | ZR34161 | | TMS32020 | ZR34161 | |
| 128-Point Magnitude Spectrum | 160 (looped) | 12 | 13x | 4.375 | 0.2636 | 16x |
| 256-Point FFT | 124 (looped) | 39 | 3x | 8.483 | 0.6328 | 13x |
| 256-Point FFT (straight-line) | 20,000 | 39 | 513x | 4.519 | 0.6328 | 7x |
| 1024-Point FFT Fixed-point | 20,000 (uses 256-Point as subroutine) | 300 | 66x | 31.82 | 2.4 | 13x |
| 1024-Point FFT Block Floating-Point | — | 600 | Not Computed | — | 3.3 | Not Computed |

TABLE 1. Program Size and Execution Comparison.¹

¹ PORTIONS OF THE DATA IN THIS TABLE OBTAINED FROM: "IMPLEMENTATION OF FAST FOURIER TRANSFORM ALGORITHMS WITH THE TMS32020", DIGITAL SIGNAL PROCESSING APPLICATIONS WITH THE TMS32020 FAMILY.

count for a 128-point FFT is 8, giving an effective 48 dB of “head-room”. With 17 bits precision for each VSP RAM word there is a total dynamic range of 150 dB in computing a 128-point FFT.

For FFTs larger than 128 points the VSP contains a scale RAM which can hold 64 scale nibbles—sufficient for processing all longer-length FFTs. When vectors of 128, or fewer, elements are to be combined in the process of computing a large FFT, the scale RAM and the maximum scale register are used by the SCL instruction to shift the elements of the vectors so that a single scale factor, or exponent, applies to the total collection of sample values.

The benefits to be obtained from BFP arithmetic operation are summarized in Table 2. The values are for the signal-to-computational-noise ratio at the output of an FFT of the size indicated. The input signal is assumed to be white noise. The results for BFP arithmetic are based on an expected number of scalings. Because scaling by 0.5 is not required at each stage, the discretionary scaling policy of BFP arithmetic results in the minimum reduction in signal amplitude. As Table 2 illustrates, the expected performance of BFP arithmetic is substantially superior to that of a fixed scaling policy. In many applications the actual signal-to-computational-noise ratio for BFP arithmetic will be significantly better than is indicated by the values in Table 2.

| FFT Size | Output Signal/Noise (dB) | |
|-------------|--------------------------|-------------------------------|
| | Block Floating Point | Scale by 1/2 at each Stage |
| 32 | 78 | 71 |
| 64 | 76 | 68 |
| 128 | 75 | 65 |
| 256 | 74 | 62 |
| 512 | 73 | 59 |
| 1024 | 72 | 56 |
| 2048 | 71 | 53 |
| 4096 | 70 | 50 |
| 8192 | 69 | 47 |
| 16384 | 68 | 44 |

TABLE 2. FFT SNR Performance Comparison.²

VSP APPLICATION EXAMPLES

In addition to its powerful FFT instruction, the other vector-oriented instructions of the VSP are ideally suited to the naturally occurring operations of most digital signal processing proce-

dures. The two examples presented here are illustrative of the wide applicability of the ZR34161.

Zoom FFT: “Zoom” FFT is the label used for the process of shifting a signal spectrum down in frequency, lowpass filtering and decimating, and then applying an FFT to view in detail a small region of the original spectrum. This process is described spectrally in Figure 9(a). Although much the same objective could be realized by performing a large FFT and selecting only the spectral components of interest, the zoom FFT approach requires much less data memory and can better isolate low-level spectral lines from the leakage effects of strong spectral peaks. A flow diagram of the signal processing involved is shown in Figure 9(b). The block diagram of Figure 1 shows more descriptively the process taking place in the complex demodulation and the decimation/lowpass filter blocks.

To be specific, consider the requirements and performance of a 10x zoom with a 128-point FFT—equivalent to a 1280-point discrete Fourier transform. The FIR decimating filter is specified as having 60 dB stopband rejection and 0.5 dB peak-to-peak ripple over a two-sided passband that is 90% of the sampling frequency. A single FIR filter would be of length 395; by cascading two FIR filters which respectively decimate by 2 and 5, the desired frequency response can be obtained with FIR filters of lengths 37 and 79. The operating sequence is to frequency shift every 10 input samples, then cycle the decimate-by-2 FIR filter every 2 samples to produce 5 values which are processed by the decimate-by-5 FIR filter which ultimately produce a single sample. This set of computations is done 128 times to produce the input data for the FFT computation.

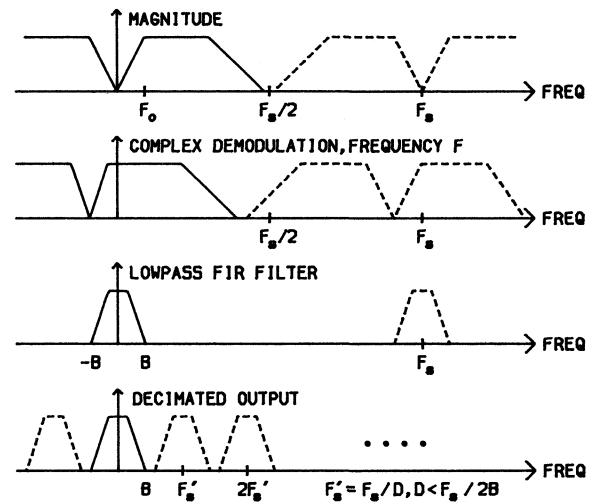
The complex modulation—frequency shifting—and FIR filtering can be done by the VSP in 10.36 msec assuming that the host does all circular buffer management and address modification for the VSP program. In another 0.237 msec the VSP can compute the FFT and store the complex spectral values in external memory. With the host managing all data buffers and system I/O, the VSP can perform the 10x 128-point zoom FFT at more than a 120 kHz input sample rate. The 128 spectral values are centered about the shift frequency which, using the VSP’s DEMO instruction to do the shifting, can be any integer multiple of $f_s/1024$, where f_s is the sampling frequency.

Frequency-domain LMS: Although simple to implement, the ordinary time-domain LMS (least-mean-squared) adaptive filter has unfortunate convergence properties. To speed up convergence, while reducing noise induced by fluctuations of weight values, several modifications to the elementary LMS process have been introduced. The frequency domain procedure—FLMS—improves convergence the most, and has the lowest computational burden of the modified LMS procedures.

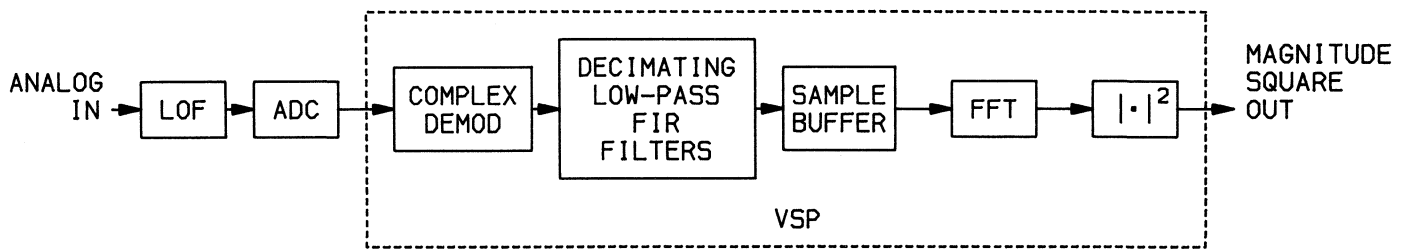
² PORTIONS OF THE DATA IN THIS TABLE OBTAINED FROM: OPPENHEIM, A., AND SCHAFFER, R., DIGITAL SIGNAL PROCESSING.

The computational steps for the FLMS procedure, as outlined in Figure 10, are perfectly matched by the VSP instruction set. In addition to the FFT instruction, the operations of complex conjugation, vector multiplication and addition, and magnitude-squared are key elements in the FLMS adaptive filter algorithm. The key to the improved convergence rate is the updating of the estimated covariance matrix; by assuming this matrix diagonal it is the variances of the spectral components which are estimated at each step. The spectral components influence the updated values of the filter weighting coefficients in inverse proportion to their variance; this effectively whitens the spectral values and forces approximately equal time constants for all filter weighting coefficients.

Assuming that the host manages the sample buffers, the VSP can implement a 32-tap FLMS adaptive filter with 4700 updates/second, and a 64-tap filter with 2500 updates/second. Because the FLMS approach achieves convergence with only one-half to one-third the number of adaptations of the ordinary LMS approach, the FLMS implementation on the VSP indicates a convergence time equivalent to a time-domain LMS filter operating at 14100 updates/second for 32 taps, or 7500 updates/second for 64 taps.



(A) SPECTRAL DESCRIPTION OF THE ZOOM FFT.



(B) SIGNAL PROCESSING BLOCK DIAGRAM.

FIGURE 9. ZOOM FFT APPLICATION.

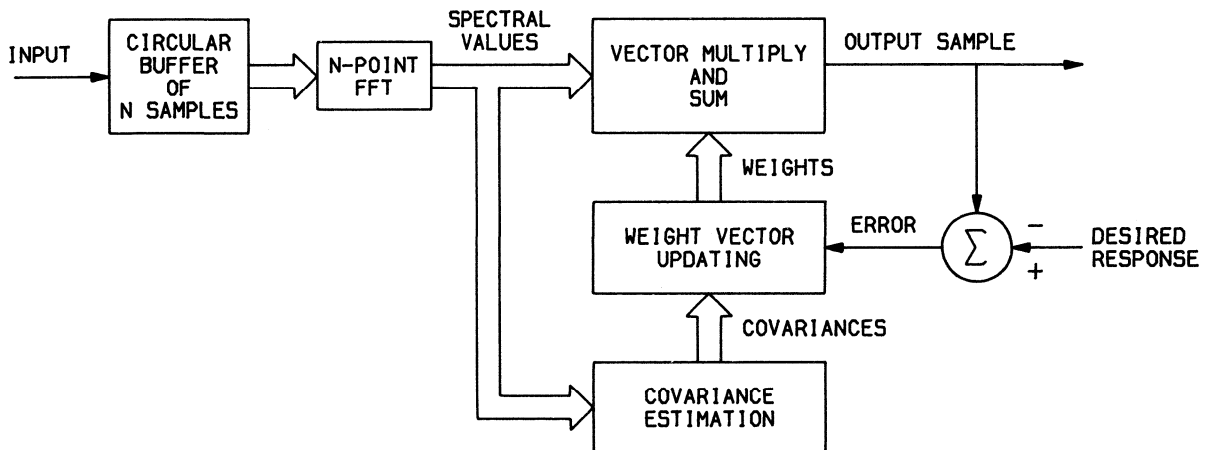


FIGURE 10. FREQUENCY-DOMAIN LMS ADAPTIVE FILTERING.

VSP SIMULATOR

The Vector Signal Processor Simulator (VSPS) software provides a comprehensive set of tools to support the user during all phases of VSP program development and system performance analysis.

FEATURES

- Full simulation of VSP instructions and operation
- Menu-driven, with command mode for expert use
- Generation of test signal samples
- Interactive specification/execution of VSP instructions
- Graphics plotting for time and frequency domain samples
- Simulation of host operations with embedded user program
- Libraries of VSP programs and floating-point routines
- Full display of VSP internal RAM/registers and timing
- Resident on PC/AT under MS-DOS and VAX under ULTRIX and VMS

The VSPS provides an environment that encourages the user to explore system performance characteristics by executing VSP programs on test signals or actual data values. Test signal samples can be generated using a flexible process which automatically maintains normalization of 16-bit values; the samples then can be processed by VSP instructions that are specified and executed interactively. Both the input signal samples and the VSP-processed results can be printed and plotted. Initial use of the VSPS is facilitated by its menu-driven operation. Expert use is supported by a command capability with macro facility. Customization is possible by embedding a user-coded program within a copy of the VSPS, and performing both host and VSP functions with the full support of VSPS tools.

The scope of the VSPS's capabilities can be illustrated by discussing the steps for creating and running a user-coded program as outlined in the flow diagram in Figure 11. The source code can be either FORTRAN or C, and the program can be composed of alternating sections of VSP and non-VSP operations. One possible sequence would be generation of special signal data (or reading of actual data from a disk file), simulation of host operations, execution of a VSP program, simulation of additional host operations, and then display of the results in a user-determined format. The full features of the VSPS, such as signal generation and graphics plotting, are available both before and after running the user program.

In Figure 11, the mixed VSP/host code source program is seen to be processed by a parser which converts the VSP code into calls to the instruction simulation routines. The host code is left unchanged in the high-level language in which it was written. The parsed file is then compiled and linked with the VSPS object modules to create an executable module. The executable module contains the entire VSPS plus the user's program. Upon executing this customized version of the VSPS, the user program can be selected from the main menu. User-programmed input and output will appear on the screen as during any normal FORTRAN or C program execution. VSPS options for selecting the display of internal RAM and registers, and for single-stepping the program execution are still available. When the user's VSP program is running correctly, the code can be converted into one of a number of different output formats for use with a PROM programmer.

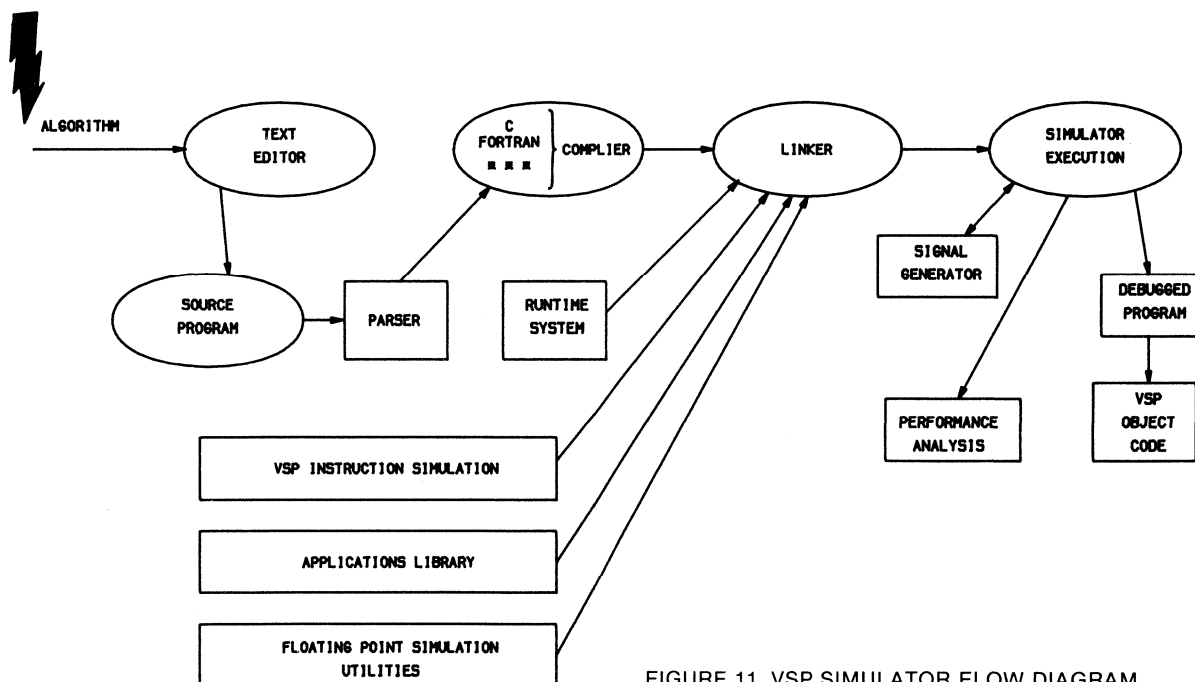


FIGURE 11. VSP SIMULATOR FLOW DIAGRAM.

In certain applications it is important to measure roundoff effects in fixed-point arithmetic, or even in the block floating-point arithmetic used in FFT calculations. This is facilitated by the inclusion of selected routines in the VSPS from the floating-point IEEE signal-processing library. Thus, for example, power spectrum calculations may be done using both a VSP program and the IEEE program, and the results compared. Comparisons of vectors are supported in the VSPS through a facility to print or plot differences between vectors.

Clock-cycle counts and bus-usage statistics can be provided by the VSPS so that real-time operations and host-VSP interactions can be analyzed. Debugging of a VSP program is aided by a range of display options which allow instructions to be executed in single-step fashion, and which allow viewing of all affected RAM and register locations. Other system features, such as simulation of multiple-VSP configurations and an external memory queue for VSP instructions, are also supported.

ASSEMBLER

The ASM-161 assembler produces an Intel hex format output file which can be loaded into the RAM on the VSPE-161 board or the VSP simulator; a listing file is also provided. Assembler directives support setting the program location counter, as well as allocating memory locations for constants and other data.

The DEFAULT directive is specific to the VSP instruction format in which several parameter values must be specified. Often these parameter values are fixed for much or all of a program. With the DEFAULT directive, parameter values can be set for all instructions which require the parameter, or only for specific instructions. This technique is illustrated in the previous program example in Figure 8(b). Alphanumeric labels for program lines are supported; labeled instruction lines are the target of a JMPI (jump direct) instruction. Labels are also used as symbolic memory location references.

VSP EVALUATION PACKAGE

The Vector Signal Processor Evaluation Package (VSPE-161) consists of a plug-in board and support software compatible with the IBM PC/XT or PC/AT. The board contains a ZR34161 processor, 16K words of 120 nsec static RAM, local bus arbitration logic, and I/O status ports. The software provides a complete development environment consisting of a debugger, utilities for controlling board functions, and a library of example VSP programs. The VSPE-161 board can also function as a hardware accelerator for the VSP Simulator software.

Debugger: Testing of programs on the VSPE-161 board is greatly facilitated by the Debugger software. The Debugger provides a command-driven environment for:

- loading and saving program and data files
- creating test signal samples in the board's RAM
- displaying, modifying, and moving memory contents
- controlling the execution of a VSP program through single-step and breakpoint-setting features
- plotting of real/complex data in the board's RAM
- composing macros of Debugger commands and other macros.

A typical sequence of operations with the Debugger would begin with the loading of a hex file produced by the Assembler. Data samples to be processed by the VSP program can be loaded from a file, typed in from the keyboard, or produced with the signal-generation command. Any segment of the board's memory can be plotted on the screen as real or complex data. Program execution is controlled either with the 'RUN' command and breakpoint settings, or with the single-step command. Program performance can be monitored by displaying the contents of board memory, VSP memory, or VSP registers. Minor program editing can be done using the disassemble command to view instructions, and the interactive-fill command to type in replacement hex programs.

Utilities: A library of utility procedures is provided for controlling and communicating with the board from a user program running under PC-DOS. The utility routines are written in C, and can be invoked from a source program coded in Microsoft C or FORTRAN. There are eleven routines for file reading and writing. An example application is included with the utilities to illustrate their use. The user program can simulate the host operations in a VSP-based system, and have all VSP code executed by the VSPE-161 board.

Example VSP Programs: The source code for several VSP programs is included in the VSPE-161 package. The detailed comments which annotate the listings provide full details on the background of the algorithm, and on the procedures for adapting the programs to the user's needs. These programs serve as examples of assembler format and VSP programming style. They also provide immediately available programs for use with the assembler and debugger in learning the application of the VSPE-161 package. Included in the library are complete programs for the FFT of 128, 256, and 1024 complex points.

VSP DEVELOPMENT PACKAGE

The Vector Signal Processor Development Package (VSPD-161) consists of a plug-in board and support software compatible with the IBM PC/AT. The board contains a ZR34161 Vector Signal Processor running at 20MHz, 64K words of 45 nsec static RAM, local bus arbitration logic, hardware breakpoint logic, two 512 word by 16-bit parallel I/O FIFO buffers, and I/O status ports. A 50-pin connector provides data I/O and control lines to the external system. The software provides a complete

development environment consisting of a debugger, utilities for controlling board functions, and a library of example VSP programs. The VSPD-161 board can function as an element of an external real-time system, thus providing the capability for system-level program development and debugging.

FIFO Usage: Real-time I/O is accomplished with the dual 512-word FIFOs. These FIFOs are mapped into the data RAM address space of the VSP so that vector transfers may be made directly to and from the external system. The FIFOs are also mapped to the PC/AT address space so that their contents may be set and read during program debugging, for example.

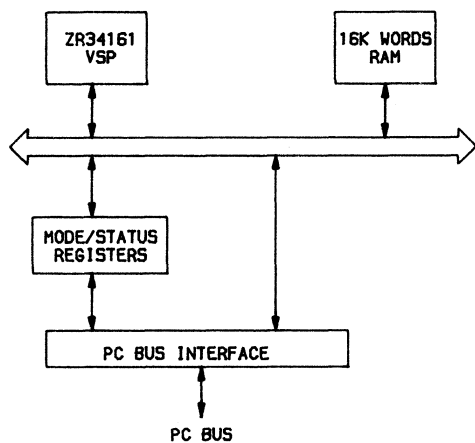


FIGURE 12. VSPE BOARD DIAGRAM.

PC/AT Interface: Full access to board functions and memory is available through the PC/AT bus. Four I/O ports are used for board configuration and operation control. Setting bits in the configuration port allows internal VSP RAM/registers and on-board RAM to be addressed in a single 64K byte segment of the PC/AT address space. The operation control port can specify breakpoint action on VSP address generation—as specified by the value in the breakpoint register—or on an external event such as reading a tagged word in the input FIFO. Strobe and board status registers control and monitor the FIFOs and bus on the VSPD-161 board.

Breakpoint Operation: The on-board breakpoint register is set from the PC/AT. When the VSP reads or writes to the specified address, the VSP is stopped by denying it access to the bus, and interrupt signals go to the PC/AT and the external connector. VSP activity is resumed after a PC/AT write to the strobe register port. Externally-initiated breakpoints come from tagged data in the input FIFO, or from the IRQBRK pin on the external connector.

Support Software: Support software consists of VSPD-specific utilities such as a debugger, board control and communication drivers, and a VSP program library. The function of this software is very similar in nature to the software described in detail in the VSPE section. Programs assembled using the VSP assembler may be loaded onto the VSPD board and executed using the debugger software. Alternatively, high-level language programs may be written (in C or Fortran for instance) which call the VSPD board and control its operation.

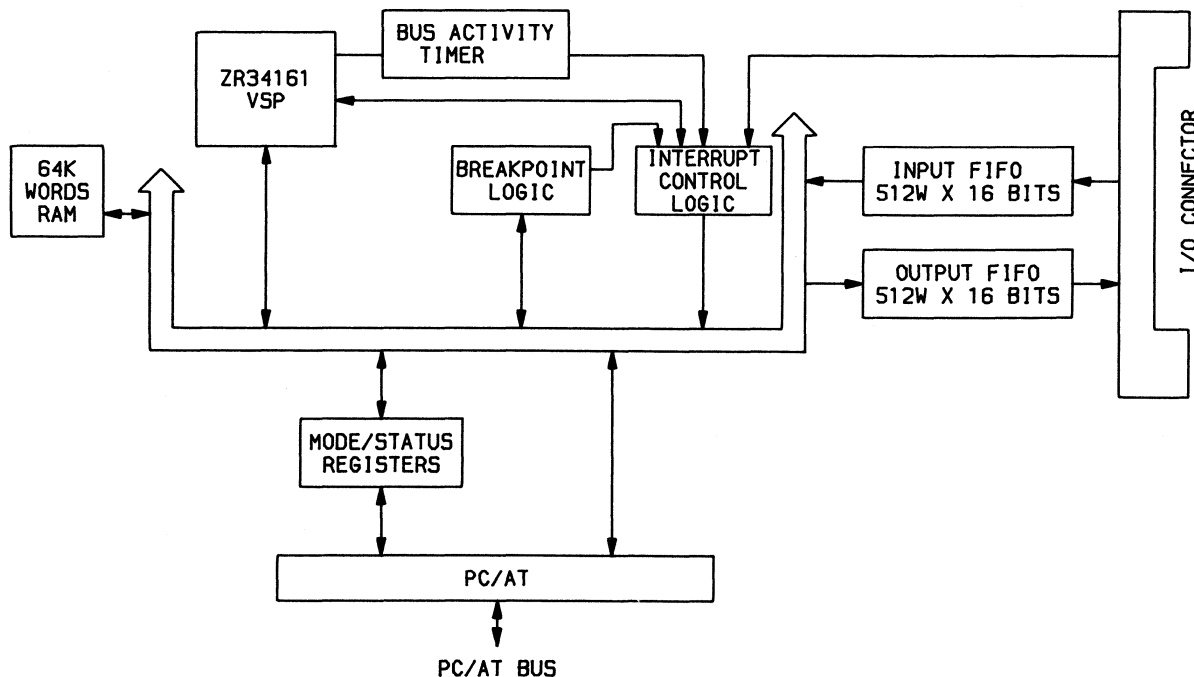


FIGURE 13. VSPD-161-BLOCK DIAGRAM.

UPDATES FROM 30 JULY 1986 ENGINEERING DATA

This section highlights changes or updates to ZR34161 Engineering Data relative to the previous (30 July 1986) publication. It is provided to allow engineers to scan these updates without reading the document in fine detail. Minor updates incorporated simply for clarity purposes are not highlighted.

CHAPTER 3:

- Add initial condition to Table 2 (Section 3.1).
- Modify CLK description (Section 3.2).
- Add "status bits notes" section at end of Section 3.3.1.

CHAPTER 4:

- Add to description of RS parameter (Section 4.2).
- Add NOTE to LDSM instruction description (section 4.3).

- Add NOTE to ADDR, MLTC, and MLTR instruction description (Section 4.4).
- Modify CMCN instruction description (Section 4.5).
- Modify bit in DEMO instruction field (Section 4.5).
- Modify bit in MODLT instruction field (Section 4.5).
- Add footnotes to Table 6 (Section 4.7).

CHAPTER 5:

- Modify host-access recovery-time specification (Section 5.1).
- Modify specifications for timing signal numbers 13, 25, 26, 39
- Redraw (for clarity) VSP master-mode read/write timing diagrams (Figures 17-20) (Section 5.2).
- Redraw (for clarity) host read/write access of VSP resources timing diagrams (Figures 21-22) (Section 5.2).
- Redraw (for clarity) preemptive bus arbitration timing diagram (Figure 25) (Section 5.2).

TABLE OF CONTENTS

| | |
|---|--|
| <p>CHAPTER 1 Processor Overview22</p> <p> 1.1 Functional22</p> <p> 1.2 Architecture23</p> <p> 1.2.1 Bus-Interface Unit23</p> <p> 1.2.2 Execution Unit24</p> <p> 1.2.3 Memory and Registers25</p> <p> 1.3 Instruction Set25</p> <p>CHAPTER 2 Functional Operation and Operating Modes 27</p> <p> 2.1 Modes of Instruction Fetch27</p> <p> 2.1.1 Slave Mode27</p> <p> 2.1.2 Master Mode28</p> <p> 2.1.3 Instruction Length28</p> <p> 2.1.4 Continuous Bus Request28</p> <p> 2.2 Modes of Instruction Execution29</p> <p> 2.2.1 Sequential Instruction Execution 29</p> <p> 2.2.2 Concurrent Instruction Execution 30</p> <p> 2.3 Data Fetch30</p> <p> 2.3.1 Master Mode31</p> <p> 2.3.2 Slave Mode31</p> <p> 2.3.3 Interrupts31</p> <p> 2.4 Interfacing Issues31</p> <p> 2.4.1 Synchronous/Asynchronous Operation31</p> <p> 2.4.2 External Memory Speed and the SUS Pin32</p> <p> 2.4.3 Bus Arbitration and Pre-emption33</p> <p>CHAPTER 3 Detailed Description34</p> | <p> 3.1 VSP Initial Conditions34</p> <p> 3.2 Interface Signals34</p> <p> 3.3 Internal Memory and Registers35</p> <p> 3.3.1 Status Register36</p> <p> 3.3.2 Mode Register37</p> <p> 3.3.3 Data RAM38</p> <p> 3.3.4 Instruction FIFO39</p> <p> 3.3.5 Next Fetch Address Register39</p> <p> 3.3.6 Instruction Base/Start Register39</p> <p> 3.3.7 Scale RAM40</p> <p> 3.3.8 Scale Register40</p> <p> 3.3.9 Maximum Scale Register40</p> <p> 3.3.10 Old Maximum Scale Register40</p> <p> 3.3.11 Accumulators41</p> <p> 3.3.12 Sinusoidal Look-Up Table (LUT)41</p> <p>CHAPTER 4 Instruction Set42</p> <p> 4.1 Literal and Logical Parameter Values ..42</p> <p> 4.2 Common Instruction Parameters42</p> <p> 4.3 Memory Instructions44</p> <p> 4.4 ALU/External Memory Instructions...46</p> <p> 4.5 Internal ALU Instructions47</p> <p> 4.6 Control Instructions52</p> <p> 4.7 Instruction Clock-Cycle Performance..52</p> |
|---|--|

LIST OF FIGURES

| | | |
|----------|---|--|
| Figure 1 | VSP logical pinout21 | |
| Figure 2 | VSP pinout21 | |
| Figure 3 | VSP architectural block diagram22 | |
| Figure 4 | Detailed VSP architecture diagram23 | |

| | | |
|----------------|---|----|
| Figure 5 | Block diagram of the VSP execution unit..... | 24 |
| Figure 6 | A host processor fetches instructions from external memory and writes them into the VSP instruction FIFO..... | 27 |
| Figure 7(a-d) | Master-mode instruction fetch and execution process..... | 29 |
| Figure 8 | Instruction concurrency is implemented by the BIU stealing one ICLK cycle from the EU for each complex sample.... | 30 |
| Figure 9 | Bus arbitration and pre-emption mechanism..... | 33 |
| Figure 10 | VSP status register format..... | 36 |
| Figure 11 | VSP mode register format..... | 37 |
| Figure 12 | Data memory organization..... | 38 |
| Figure 13(a-c) | Instruction fetch registers..... | 39 |
| Figure 14(a-d) | Scale RAM and registers..... | 40 |
| Figure 15 | VSP accumulators..... | 41 |
| Figure 16 | Look-up table organization..... | 41 |
| Figure 17 | One-cycle asynchronous read/write with/without wait state..... | 58 |
| Figure 18 | Two-cycle asynchronous read/write with/without wait state..... | 58 |
| Figure 19 | One-cycle synchronous read/write with/without wait state..... | 59 |
| Figure 20 | Two-cycle synchronous read/write with/without wait state..... | 59 |
| Figure 21 | Control timing..... | 60 |

| | | |
|-----------|---|----|
| Figure 22 | VSP basic bus arbitration timing..... | 60 |
| Figure 23 | External post RD access to VSP resources..... | 61 |
| Figure 24 | External host WR access to VSP resources..... | 61 |
| Figure 25 | Preemptive bus arbitration..... | 62 |
| Figure 26 | A.C. test load..... | 63 |
| Figure 27 | Normal A.C. test load..... | 63 |
| Figure 28 | ZR34161 typical Icc requirements..... | 65 |

LIST OF TABLES

| | | |
|---------|--|----|
| Table 1 | VSP performance benchmarks..... | 22 |
| Table 2 | Initial VSP conditions—effect of reset..... | 34 |
| Table 3 | Tabular listing of the internal VSP registers and their addresses..... | 35 |
| Table 4 | Logical vs. literal interpretation for the NMPT parameter..... | 42 |
| Table 5 | Example illustrating address bit-reversal..... | 44 |
| Table 6 | VSP instruction clock-cycle performance..... | 53 |
| Table 7 | VSP event timing display..... | 54 |

This data sheet is organized into five primary sections, each of which provides a different type and depth of description. The Table of Contents outlines the topics presented in each section.

DISTINCTIVE FEATURES

- High-performance 16-bit digital signal processor
- Both integer and block floating-point arithmetic
- 23 high-level DSP-oriented instructions
- Concurrent I/O & ALU operations
- On-device:
 - RAM: 128 complex words by 38 bits
 - sine/cosine look-up table
 - DMA interface
 - Instruction FIFO
- 16-bit address and data buses
- Easily paralleled for greater throughput
- Powerful hardware and software development environment
- Low power CMOS (<300mW)
- 48-pin DIP package

DESCRIPTION

The ZR34161 Vector Signal Processor (VSP) is a high-performance, programmable digital signal processor with an internal architecture optimized for efficient and rapid execution of digital signal processing (DSP) algorithms such as: Fast Fourier Transforms, convolutions, correlations, sine/cosine transforms, general vector and matrix operations, and power spectrum calculations.

The VSP is designed as a peripheral signal processor to work in conjunction with a host controller as a functional replacement for a board-level signal/array processor. The device is programmable through a high-functionality, vector-oriented instruction set, thus greatly reducing the time required for DSP algorithm development as compared with traditional instruction sets. The block floating-point arithmetic used for the FFT provides a significantly better signal-to-computation-noise ratio than is achievable with traditional fixed-point calculations. ROM and

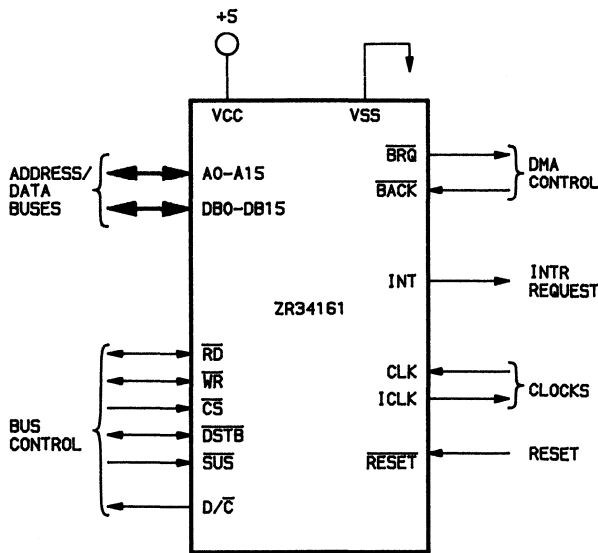


FIGURE 1. VSP LOGICAL PINOUT.

APPLICATIONS

- 1-D and 2-D FFTs
- Sine/cosine transforms
- Convolutions and correlations
- Spectral analysis
- Spectral moments
- Vector/Matrix arithmetic
- Image processing
- Radar/sonar processing
- General transform-domain processing
- Beam forming
- Frequency shifting (complex demodulation)

RAM contained on the processor are used for moderate-sized vector operations, and are combined with external memory for longer length vector operations. Extremely high throughput is supported by allowing simultaneous execution of both input/output (I/O) and ALU operations. An instruction FIFO and simple DMA structure contained on the processor reduces system bus/host overhead.

A powerful software development environment is available for creating and debugging complete applications using the VSP. This environment not only simulates the processor—its architecture, instruction set, bus timing, and arithmetic—but also allows high-level language simulation of the environment external to the VSP. An optional hardware board is also available which allows algorithm development with full-speed execution of VSP programs.

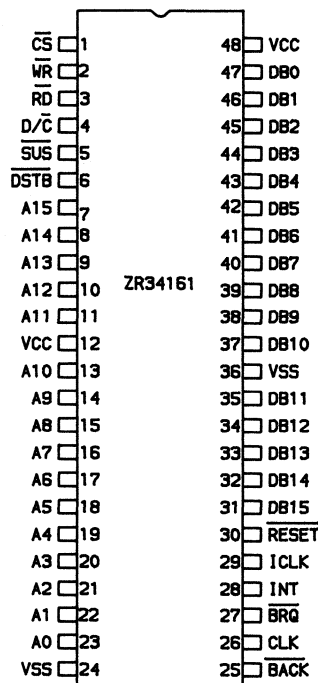


FIGURE 2. VSP PINOUT.

1.0 PROCESSOR OVERVIEW

1.1 FUNCTIONAL

The ZR34161 Vector Signal Processor (VSP) is a high-performance digital signal processing component. It functions as the system processor in designs that require high-performance execution of digital signal processing algorithms. The VSP interfaces easily to a wide variety of host microprocessors or controllers which direct its activities. Figure 3 shows a block diagram of the major internal VSP blocks as well as its general interface to the external world.

A dual-processor architecture supports simultaneous execution of I/O operations and arithmetic/DSP operations. One of the processor blocks is the *bus-interface unit* which controls all data and instruction movement to and from the VSP. A second processor block is the *execution unit* which performs the arithmetic operations. The *memory and registers* support both processor blocks by providing data RAM, a coefficient Look-Up Table (LUT) and operational registers. The registers program the VSP operational characteristics, provide processor status, and support other internal or external activities. To implement execution of the high-level instructions, an instruction sequencer is contained in hardware on the processor. The burden on the external system is further reduced by the ability of the VSP to fetch its own instructions and data using its powerful yet simple DMA structure. Its architecture permits fast and efficient execution of DSP operations with minimal overhead required for I/O.

The VSP has the unique property that its 23 high-level instructions operate on complex *vectors* or *arrays* of data. The instructions are grouped into one of four categories: data movement, internal arithmetic, internal-external arithmetic, or control. An entire complex array of data may be loaded into the VSP in a single instruction, then operated upon by one or more DSP-oriented instructions, and finally written back to external memory with a single instruction. More traditional signal processing components operate on only a single element of a complex array at a time using standard scalar-oriented instructions.

Because of the high-level nature of the VSP instruction set, a relatively short amount of time is spent fetching VSP instructions relative to their execution times. Because of this, it is possible to parallel multiple VSPs on a single bus for even higher signal processing throughput.

The VSP is memory-mapped into the host address space so both can share the same external data and instruction memory. It can operate as a stand-alone processor with minimal external support logic or as a peripheral processor on the system bus under the control of a more sophisticated host.

Table 1 provides benchmarks of the times required for the VSP to execute certain signal processing algorithms.

Table 1. VSP Performance Benchmarks

| <u>Operation</u> | <u>time (μs)</u> |
|--|------------------|
| 1024-point block-floating complex FFT | 3300 |
| 1024-point integer complex FFT | 2600 |
| 1024-point block-floating complex FFT ¹ | 1200 |
| 128-point block-floating complex FFT | 237 |
| 8 x 8-point 2-D complex FFT | 164 |
| 16 x 16-point 2-D FCT | 1100 |
| 128-point x 128-point complex vector multiply | 53 |
| 128-point magnitude square/accumulate | 26 |
| 128-point complex demodulation | 52 |
| 4 x 4 matrix multiplication | 33 |
| load/store 128 complex data samples | 26 |

¹ 4 parallel VSPs on a single bus

NOTE: The FFT execution times *include* data loading and storing times as well as execution times of the algorithms.

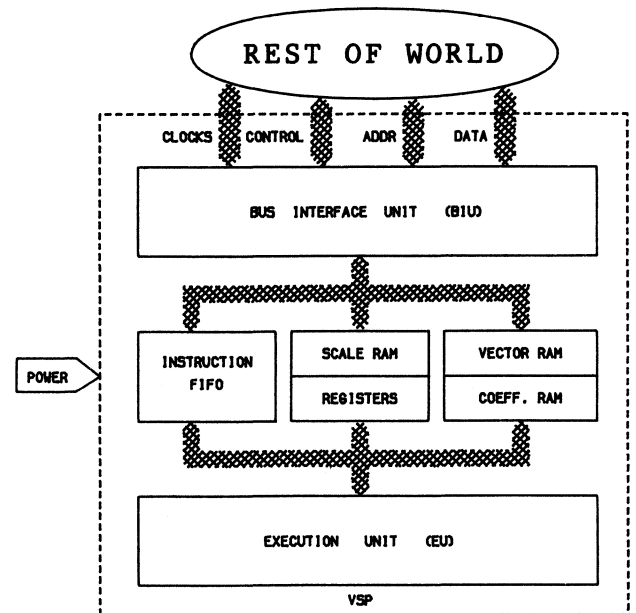


FIGURE 3. VSP ARCHITECTURAL BLOCK DIAGRAM.

1.2 ARCHITECTURE

Figure 4 shows a block diagram of the architecture of the VSP. It consists of three main blocks: the Bus-Interface Unit (BIU), the Execution Unit (EU), and the memory and registers. The BIU and EU are independent processor blocks which share the memory and registers.

1.2.1 BUS-INTERFACE UNIT (BIU)

The Bus-Interface Unit comprises everything on the left side of the figure. Included in the BIU are:

- data bus buffers
- address generator
- instruction fetch unit
- bus-interface control

The BIU is responsible for executing all bus-related operations including instruction fetching and decoding, data I/O, and communication among the internal and external memory devices. DMA activities between the VSP and external memory are also

controlled within the BIU. An instruction FIFO (first-in, first-out buffer) is present in the instruction fetch block which is able to store up to four VSP instructions.

The VSP transfers data and fetches instructions in a manner similar to traditional microprocessor peripherals with a DMA interface. Because of the "high-level" vector nature of the instruction set and the on-chip FIFO, both data I/O and instruction fetching may be done in blocks. Hence, the VSP has a simple DMA structure using the \overline{BRQ} and \overline{BACK} pins for effecting the block move operations. The DMA structure may operate in conjunction with a bus arbiter in the host system. Each data I/O instruction may transfer as many as 256 16-bit words over the bus.

The \overline{RD} , \overline{WR} , and \overline{DSTB} pins are bi-directional depending on whether the VSP is using them for input or output. When the \overline{CS} input pin is enabled, the pins are used as inputs. When the VSP is reading or writing to or from external memory in the master mode they are output pins. When neither operation is taking place, the pins assume a high-impedance condition. It may be necessary to provide external pull-ups on these pins depending on the system configuration.

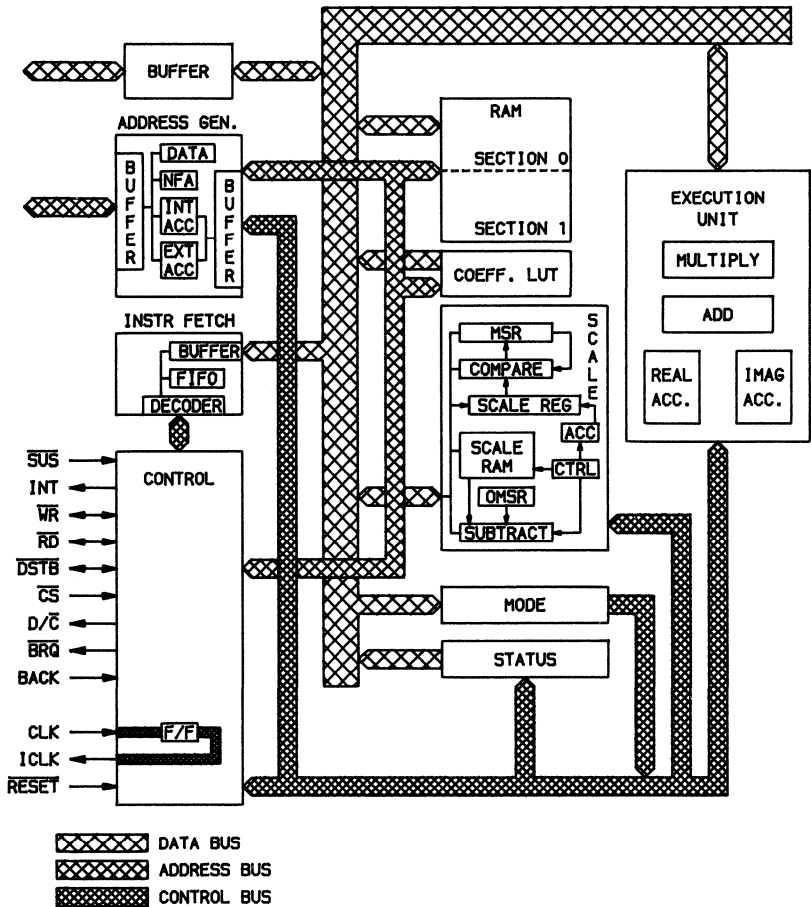


FIGURE 4. DETAILED VSP ARCHITECTURE DIAGRAM.

1.2.2 EXECUTION UNIT

The execution unit (on the right side of the figure) is responsible for all ALU-intensive operations. Included in the EU are:

- 17 x 17 bit multiplier
- adder
- adder/subtractor
- two 25-bit accumulators
- logic for efficient implementation of FFT butterflies

The EU performs DSP operations with an architecture designed specifically to efficiently implement FFT butterflies. Because of the inherent structure of these types of operations, this same architecture can also be used to perform all of the other signal processing tasks provided by the VSP. The calculations are performed with an internal accuracy of 17 bits; both fixed and block floating-point FFT arithmetic are supported. The EU performs the following types of DSP vector operations using a single "high-level" instruction:

- Real or complex dot product
- Real or complex addition
- Real or imaginary vector accumulation
- Scalar multiplication
- Absolute value
- Magnitude square/accumulate
- Complex conjugate
- Fast Fourier Transform

The architecture of the execution unit is shown in Figure 5. The addition of various data paths from the outputs of internal points in this architecture provide the ease for which the VSP implements all of the other signal processing instructions besides the FFT.

The 25-bit accumulators allow accumulation with no overflow of the 256 17-bit words which are required for the 128 complex-point dot products. One of the accumulators is provided for the real part of the complex results and the second for the imaginary part.

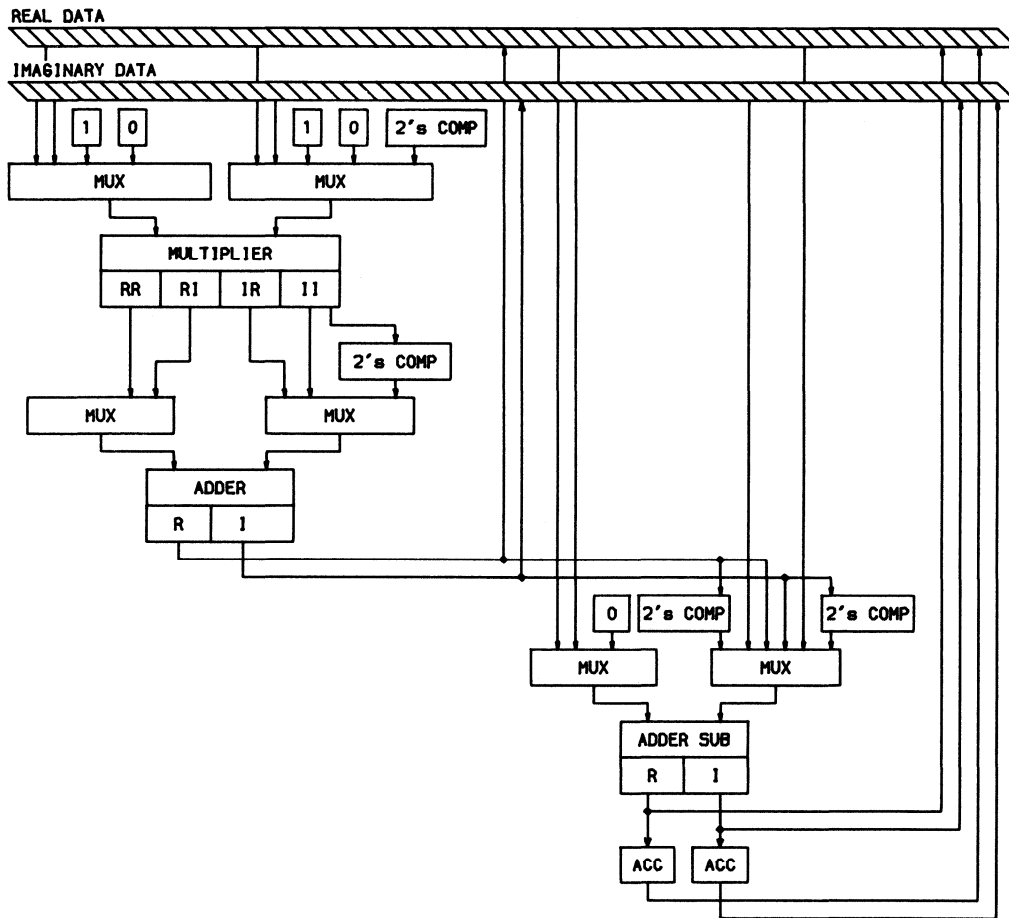


FIGURE 5. BLOCK DIAGRAM OF THE VSP EXECUTION UNIT.

The EU hardware supports both fixed-point as well as block floating-point operations for FFT computations. Using fixed-point arithmetic, the results of the ALU are scaled (divided by 2) at the end of each FFT pass. This eliminates the possibility of overflow during the addition required for the FFT butterfly.²

With block floating-point arithmetic, the results of ALU operations at the end of each FFT pass are scaled *only if an overflow has occurred*. The vector scale factor is determined by the size of the maximum overflow in that pass (up to 2 bits²). The block floating-point exponent then keeps track of the number of shifts performed during the FFT instruction. Thus, the block floating-point capability retains the precision of the input signal during FFT computations by not performing right shifts if no overflow has occurred. The VSP uses a four-bit scale factor which, when using block floating-point arithmetic, allows scaling of up to 2¹⁵.

The length of the data vectors on which the EU can operate in a single instruction is limited to the VSP RAM size of 128 complex samples. When performing signal processing operations which use data arrays larger than this, the operations must be factored into multiple smaller-size operations of 128 samples or less. The appropriate instruction is then used as a “kernel” for completing the larger operation, where the appropriate instruction is called multiple times to complete the large operation. For instance, applications for FFTs of larger than 128 points must call the FFT instruction multiple times in order to complete the transform.

1.2.3 MEMORY AND REGISTERS

Coupled very tightly to both the execution unit and the bus-interface unit are memory and registers, consisting of:

- 128 complex-word x 38-bit data RAM
- 64-nibble x 4-bit scale RAM
- scale registers and logic
- 256-word x 17-bit sine/cosine look-up table
- mode and status registers

The data RAM contains 128 complex 38-bit words. Each complex word consists of a 19-bit real part and a 19-bit imaginary part. The upper two bits are “guard bits” to prevent overflow when using block floating-point FFT computations, and are not available externally. In addition, the LSB of the RAM in each word is not available externally. It is carried internally to provide extra precision for the unbiased rounding scheme used by the VSP.

² NOTE: UNDER CERTAIN, RARE CIRCUMSTANCES, IT IS POSSIBLE FOR A PASS OF THE FFT USING FIXED-POINT SCALING TO GENERATE A 2-BIT OVERFLOW. USING BLOCK FLOATING-POINT ARITHMETIC, THE VSP WILL SCALE CORRECTLY.

The RAM can be configured as two independent sections, each containing 64 complex words. One of the independent RAM sections can be accessed by the BIU while the other section is being accessed simultaneously by the EU. This powerful feature allows I/O instructions to be performed concurrently with ALU operations in many cases. As an example, assume that an application requires continuous real-time FFT calculations. While the EU is doing FFT butterfly calculations in one RAM section, the BIU simultaneously can first store the results of the previous FFT instruction to external memory and then read in the data to be transformed next to the other RAM section.

The VSP also contains a number of registers which are shown in the architecture diagram in Figure 4. The registers serve both to control the operation of the VSP and to provide operational status to the host controller.

Both the BIU and the EU have access to the internal data RAM, scale RAM, and LUT under control of the instruction parameters and the various VSP registers. All external data to be operated on by the execution unit first passes through the bus-interface unit. Figures 10-16 present detailed diagrams of the organization of the VSP internal memory and registers accompanied by their respective memory-mapped addresses.

1.3 INSTRUCTION SET

The VSP provides 23 DSP-oriented instructions, each of which exist in one of four categories: data I/O, ALU, external memory/ALU, or control. The instructions vary in length from one to three words. The instruction set is designed to be highly functional; each instruction can be thought of as analogous to a subroutine kernel in a signal processing library. The VSP is programmed at the *functional* level, not the assembly-language level required of other signal processing components.

Most of the instructions have several parameters, the values of which control the way the instruction is executed.

The 23 instructions within the VSP are categorized into four functional types as follows:

- 1) data movement on blocks of data in either direction between the VSP internal memory or registers and external memory:

| | |
|-------------|-------------------------------|
| LD | (Load) |
| LDSM | (Load Scale/Mode Register) |
| ST | (Store) |
| STB | (Store Backward) |
| STI | (Store Information Registers) |

- 2) ALU/memory instructions with two data vectors as operands, one residing in the internal VSP RAM and the other residing in external memory:

ADDC (Vector Add Complex)
ADDR (Vector Add Real)
MLTC (Vector Multiply Complex/Accumulate)
MLTR (Vector Multiply Real/Accumulate)

- 3) ALU instructions which operate on a single data vector stored internally in the VSP RAM:

ABS (Absolute Value)
ACCI (Accumulate Imaginary)
ACCR (Accumulate Real)
CMCN (Complex Conjugate)
CMLT (Cross Multiply)
DEMO (Demodulate)
FFT (Fast Fourier Transform)
MGSQ (Magnitude Square/Accumulate)
MODLT (Modulate)
SCL (Scale)
SCLT (Scale Literal)

- 4) control instructions:

JMPI (Jump Indirect)
HLT (Halt)
NOP (No Operation)

2.0 FUNCTIONAL OPERATION AND OPERATING MODES

The VSP can be programmed to operate in different modes when performing tasks such as instruction fetching, instruction execution, and data fetching. In addition, it provides significant flexibility in its interface with the host microprocessor and the external memory. Those topics are all covered in this section.

2.1 MODES OF INSTRUCTION FETCH

The internal instruction FIFO, out of which the VSP executes all instructions, is written to in one of two modes:

Slave Mode: a host processor writes instructions directly into the VSP instruction FIFO using its memory mapped address.

Master Mode: the VSP fetches its own instructions from memory and stores them in the instruction FIFO.

In case of both master and slave modes, the VSP can interface with external memory and the host processor in either a synchronous or asynchronous manner, as described in section 2.4.1.

2.1.1 SLAVE MODE

In the slave mode, the host writes program instructions into one of two addresses in the VSP instruction FIFO—either *buffering address* 302H or *immediate-execution address* 303H. Instructions may be written into the FIFO concurrently with instructions that the VSP is executing. Up to twelve instruction words can be queued into the VSP.

Writing instructions into memory-mapped address 303H (immediate execution address) of the FIFO initiates their execution in the slave mode. If the host requires a delayed start of instruction execution, memory-mapped address 302H (buffering address) should be used for writing the instructions into the FIFO. Up to three instructions can be queued without execution by using the buffering address. The last word of the fourth instruction must be written to the immediate execution address in order to start instruction execution. Writing more than four instructions into the FIFO will result in the loss of the excess instructions.

Once started, instruction execution will continue as long as the FIFO is not empty. Assuming the respective bits in the mode register are set, the VSP will interrupt the host with the INT pin on two conditions in the slave mode: 1) the final instruction in the FIFO has completed execution, at which point the ILI bit in the status register will be set; 2) the host attempts to write a word to the FIFO when there are no free slots available, at which point the IFO bit in the status register will be set.

This technique is illustrated in Figure 6. The host processor fetches instructions from an instruction memory space and feeds these instructions into the VSP instruction FIFO from where they are executed. If the host is writing instructions into the buffering address (302H), the instructions are queued for execution. As soon as the host writes the final word of an instruction into the immediate-execution address (303H), the VSP will begin execution with the first full instruction written to the FIFO. This mode of operation is particularly useful during system development for single-stepping through VSP programs.

NOTE: It is not required that the host write instructions to the buffering address; all instructions may be written to the immediate execution address.

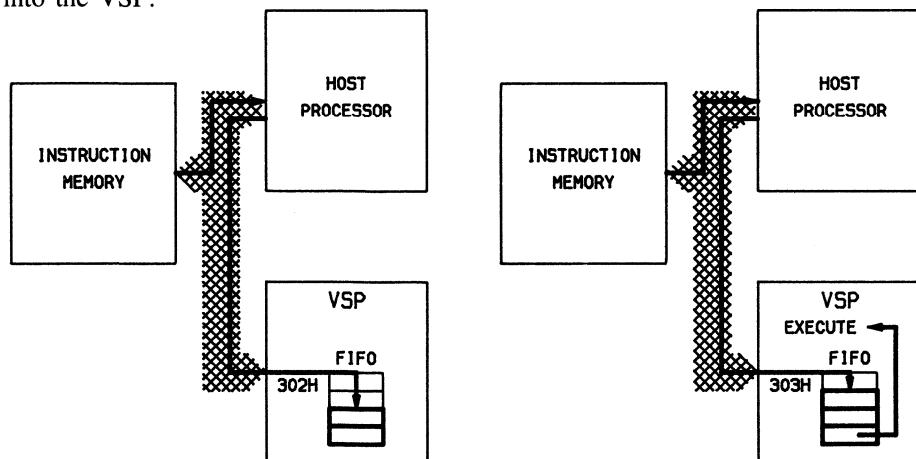


FIGURE 6. A HOST PROCESSOR FETCHES INSTRUCTIONS FROM EXTERNAL MEMORY AND WRITES THEM INTO THE VSP INSTRUCTION FIFO.

2.1.2 MASTER MODE

The VSP fetches its own instructions from external memory when it is used in the master mode. Two ICLK cycles are used for fetching each word of an instruction. This allows the use of “slower” (less expensive) memory than that used for data.

NOTE: Mode-register bit NCS in no way affects the number of ICLKS used by the VSP for fetching instructions.

The VSP is programmed into the master mode in one of two ways: 1) by the host writing the program base address into the instruction base/start register, or 2) by the host writing a JMPI instruction to the FIFO in the slave mode (which is subsequently executed by the VSP). In the first case, the VSP will begin fetching its own instructions at the address contained in the base/start register. In the second case, the VSP will begin fetching instructions at the address contained at the location pointed to by the MBA field in the JMPI instruction. Please refer to the Event Timing Table (Table 7) which describes when in time these events will occur.

Figure 7 illustrates the first technique described above. The host processor writes the starting address to the VSP instruction base/start register (306H) (Figure 7(a)). The VSP then begins instruction fetching and execution beginning at that address (Figure 7(b)). Instruction fetching will continue until a HLT instruction is fetched (Figures 7(c and d)). The occurrence of the HLT instruction will stop VSP instruction fetch; however, instructions already in the FIFO will continue to be executed until the FIFO empties.

The VSP will fetch instructions until three out of the four FIFO slots plus the first word of the fourth slot are full. Instruction fetch and decoding are done in parallel. By the time the fetch mechanism has decoded the last word in the third instruction, the VSP has already fetched the first word of the fourth instruction and stored it in the FIFO.

The VSP suspends instruction fetch at the first word of the fourth instruction and removes \overline{BRQ} . Since the fourth instruction could be only one word in length, performing further fetches might be an attempt to read a fifth instruction. The FIFO can hold at most four instructions, so the fifth instruction would be lost. In the master mode, this means that there can be at most 10 words in the FIFO, and this occurs only when the first three instructions are each three words in length. Instruction fetch will resume when there are at least two empty instruction slots in the FIFO. The fetch mechanism tries to keep the FIFO full, and it will use every opportunity when the bus is available to do so.

A JMPI (Jump Indirect) instruction temporarily stops the fetch mechanism at the time the instruction is loaded into the FIFO. When it subsequently executes, the content of the next fetch address register is loaded with the value pointed to by the MBA field; fetch operations are then resumed at the new fetch address after the NFA register is loaded.

A HLT (Halt) instruction stops the fetch mechanism at the time the instruction is loaded into the FIFO.

The host can follow the progress of program execution in the master mode by reading the next fetch address register using its memory-mapped address.

2.1.3 INSTRUCTION LENGTH

INL controls whether one-word VSP instructions are fetched with their defined length (INL=0), or with a fixed length of three words per instruction (INL=1). When fetched as three-word instructions, the external code must contain the dummy instruction words which extend the instruction length to three words; these two words should both contain zeros. Two-word instructions may be extended to a fixed three-word length using the LN parameter in the instructions. HLT, the one exception, is always a two-word instruction.

The fixed-length form may be desired in cases where the VSP is used in an instruction-fetch mode with simple external logic controlling the instruction feeding process to the VSP.

2.1.4 CONTINUOUS BUS REQUEST

The priority of instruction fetch in the VSP is controlled by mode-register bit CRQ (Continuous Bus-request). When CRQ=1, the VSP *will not* relinquish the bus (deactivate \overline{BRQ}) between consecutive memory instructions, but will instead move from the first memory instruction directly to the second. This eliminates the need for a new bus request/acknowledge handshake sequence, the protocol of which may take additional clock cycles to complete. In this case, the VSP instruction fetch unit is not able to fetch additional instructions for the FIFO between memory instructions. Note that if the code contains a continuous sequence of memory instructions, the instruction FIFO will be inhibited from fetching additional instructions and the FIFO may empty. As soon as the data bus becomes available, however, the instruction fetch unit will fetch additional instructions.

Programming mode-register bit CRQ=0 forces the VSP to provide one free ICLK cycle between consecutive memory instructions. If the FIFO needs additional instructions, the instruction fetch unit will take control of the bus during this time and keep control until the FIFO is full. If the FIFO does not need additional instructions, the VSP will toggle \overline{BRQ} inactive for one ICLK cycle and bring it active again.

It is important to note that when the VSP deactivates \overline{BRQ} , the host should immediately deactivate \overline{BACK} in response. *If the VSP reactivates \overline{BRQ} prior to the deactivation of \overline{BACK} , the VSP will not recognize that \overline{BACK} is active.*

2.2 MODES OF INSTRUCTION EXECUTION

There are two modes of instruction execution within the VSP:

Sequential Mode: instructions are executed sequentially, one-at-a-time.

Concurrent Mode: The RAM must be partitioned into two functionally independent sections of 64 complex samples each; the EU and BIU may work concurrently on the different sections. This enables the VSP to execute two instructions simultaneously.

The instruction fetch mechanism works the same way for both sequential and concurrent instruction execution. The following rule for instruction fetch applies to both modes: instruction fetching requires the BIU, but has lower priority than data I/O; therefore, instruction fetch waits for bus-idle cycles in order to gain control of the bus for fetching new instructions. Once instruction fetching controls the bus, it will release the bus only after it has finished filling the FIFO.

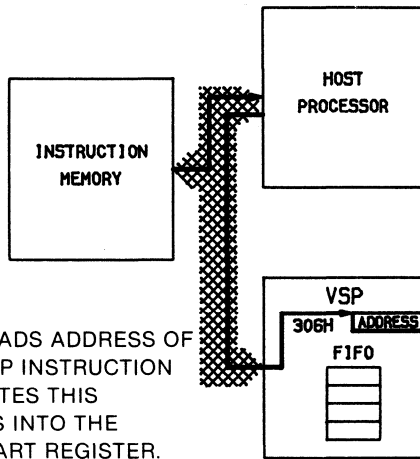


FIGURE 7(A). HOST READS ADDRESS OF FIRST VSP INSTRUCTION AND WRITES THIS ADDRESS INTO THE BASE/START REGISTER.

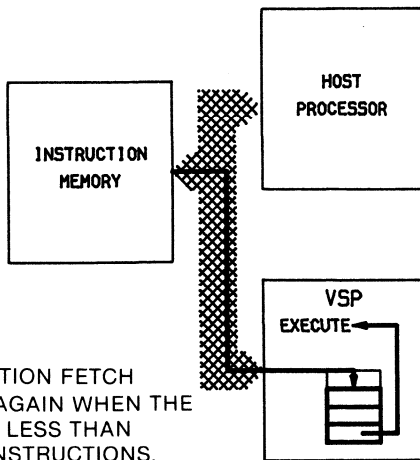


FIGURE 7(C). INSTRUCTION FETCH BEGINS AGAIN WHEN THE VSP HAS LESS THAN THREE INSTRUCTIONS.

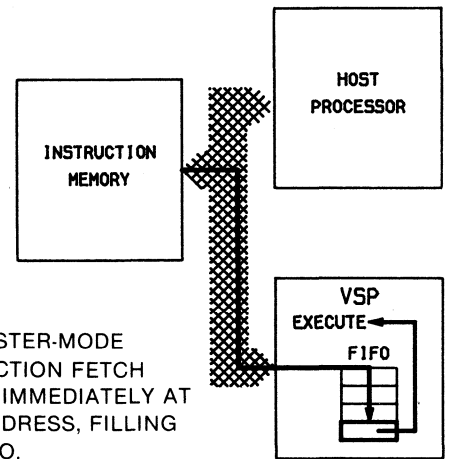


FIGURE 7(B). VSP MASTER-MODE INSTRUCTION FETCH BEGINS IMMEDIATELY AT THIS ADDRESS, FILLING THE FIFO.

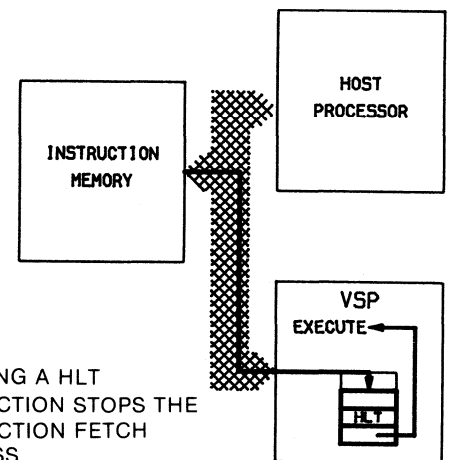


FIGURE 7(D). FETCHING A HLT INSTRUCTION STOPS THE INSTRUCTION FETCH PROCESS.

FIGURE 7(A-D). MASTER-MODE INSTRUCTION FETCH AND EXECUTION PROCESS.

2.2.1 SEQUENTIAL INSTRUCTION EXECUTION

When mode-register bit NMS=1 (one RAM section), the data RAM is treated as one consecutive array of 128 complex words, and instructions in the FIFO are executed sequentially. The vector length on which instructions can operate is up to 128 complex words.

In the sequential mode, instructions are executed one-after-the-other, and instruction fetch still overlaps with instruction execution. The instruction fetch mechanism can take control of the bus while the VSP is performing internal ALU instructions, or between consecutive data I/O instructions if mode-register bit CRQ=0 (See section 2.1.4 for discussion of CRQ).

2.2.2 CONCURRENT INSTRUCTION EXECUTION

Setting mode-register bit NMS=0 will define the data RAM as consisting of two 64 complex-word sections. The VSP will automatically execute instructions concurrently whenever NMS=0 and the instructions involved obey the following concurrency rule:

Two VSP instructions can be executed simultaneously when each requires execution by a different VSP processor (BIU or EU) and also uses a different RAM section.

Note that when NMS=0, the maximum data vector length on which instructions can operate is 64 complex samples.

The next instruction in the FIFO is always checked for the required resources for its execution.(processor and RAM section); if the resources are available, execution will commence immediately. Two conditions exist where correct instruction ordering will cause instructions to be executed concurrently:

- 1) An internal ALU instruction follows a memory instruction (or vice-versa) where each requires execution by a different processor and each addresses a different RAM section.
- 2) Move instruction (Memory or Memory/ALU) following a Memory/ALU instruction—the last few (5-9) ICLK cycles of a Memory/ALU instruction execution are performed entirely by the EU, therefore releasing the BIU for the next data move instruction. In the case where the data operated upon in external memory is a constant, concurrency will begin as soon as the constant is loaded into the VSP.

An instruction requiring the use of both the BIU and EU cannot be executed concurrently with another instruction. The exception to this is case 2 above which will perform partial concurrency at the completion of the BIU portion of the instruction if a valid I/O instruction follows. The user maintains control over which RAM section is to be used for each instruction by using the RS field contained in each instruction.

Note that by correct choice of RAM section, the user can force sequential execution of instructions when such is necessary.

The VSP implements instruction concurrency by sharing the internal address and data buses. This is effected by the BIU taking internal bus priority and freezing execution of the EU for one cycle at a time when it needs to move data to or from internal data memory.

Note that the stolen cycles for concurrency must be added to the execution time for EU instructions.

This technique of sharing the internal address and data buses is shown in Figure 8. In this particular example, a LD or ST instruction using the BIU is being executed concurrently with an ALU instruction using the EU. Because the internal data bus is 38 bits in width, this results in the loss of *one ICLK* for each BIU access of a complex word. When performing load and store operations concurrently with ALU operations, the BIU will move both the real and imaginary parts of a complex word (32 bits) simultaneously to or from internal memory on a single clock cycle. Hence, only one clock cycle per complex sample will be stolen from EU operations. Note that the example assumes that one-cycle external memory is being used. For cases where two-cycle external memory is being used, only one cycle out of four ICLKs is stolen from internal ALU operations.

2.3 DATA FETCH

The VSP memory and certain registers may be read from or written to in one of two ways:

Master Mode: The VSP is the master of the bus, and will transfer data between its internal memory and registers and external memory without host assistance.

Slave Mode: The host processor may read from and write to internal VSP memory and registers using their respective memory-mapped addresses.

In the case of both master and slave modes, the VSP can interface with external memory and the host processor in either a synchronous or asynchronous manner, as described in section 2.4.1.

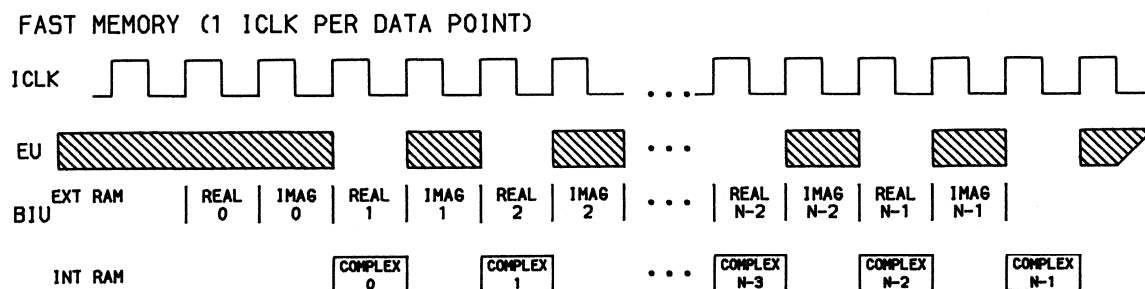


FIGURE 8. INSTRUCTION CONCURRENCY IS IMPLEMENTED BY THE BIU STEALING ONE ICLK CYCLE FROM THE EU FOR EACH COMPLEX SAMPLE.

2.3.1 MASTER MODE

When fetching data in the master mode, the VSP may transfer up to 256 16-bit words in a single instruction, as defined by the *number of points* field (among others) in the instruction setup. The \overline{RD} , \overline{WR} , and \overline{DSTB} pins are all outputs in this mode. Prior to taking control of the bus, the VSP will engage in a bus request/bus acknowledge protocol with the host as discussed in section 2.4.3. Until the VSP has been granted the bus, it will never generate outgoing \overline{RD} , \overline{WR} , or \overline{DSTB} signals.

If the VSP has guaranteed access to external memory as defined by the two conditions below, then the VSP should be operated in the synchronous mode with 1 ICLK per word or 2 ICLK per word access (NCS=0 or 1) as appropriate for the external memory speed:

- 1) the memory will always be ready for access;
- 2) the setup-and-hold times will always be met.

If the first is not true, then the \overline{SUS} (suspend) signal must be used. If the latter is not true, then the asynchronous mode must be used.

There are three motivations for using the \overline{SUS} input pin in a synchronous system:

- 1) memory is shared between the VSP and some other user;
- 2) the system contains multiple memories of mixed access times, or with at least one of them being shared with another user; or
- 3) the memory is too slow for synchronous operation at 2 ICLKs per word.

2.3.2 SLAVE MODE

In the slave mode of data fetching, the host processor can perform both memory-mapped read and write operations to the VSP memory and registers. The reading and writing times of memory and registers may occur at the discretion of the host. This is true even if the host wants to access VSP data RAM when the processor is in the middle of an arithmetic operation. If the host attempts to read or write RAM during an arithmetic instruction, the BIU will steal a cycle from the EU just as the case when I/O operations are performed simultaneously with arithmetic operations.

During slave-mode data access of the VSP, the \overline{RD} , \overline{WR} and \overline{DSTB} pins are conditioned as inputs when the \overline{CS} pin is enabled by the host.

2.3.3 INTERRUPTS

The VSP status register contains bits which provide information to the external system about events happening within the processor. Five of these bits (IMI, IAI, IDO, ILI and IFO) are capable of setting the external interrupt pin (INT) when they are set in the status register. Each of the five may be individually enabled so that the setting of the respective bit in the status register may cause the INT pin to be set HIGH. Three of the bits (IDO, ILI, and IFO) are enabled in the mode register. IMI and IAI are enabled by setting the EI bit in an individual instruction word.

Disabling the bit will not allow the INT pin to be set when the respective bit is set in the status register. A read of the status register clears the five status bits in the status register. Table 7, in section 5.0, lists the number of clock cycles after an event is sensed until both the status bit and INT pin are set.

2.4 INTERFACING ISSUES

Three additional issues exist regarding the interface of the VSP to the host system for both data and instruction fetching.

2.4.1 SYNCHRONOUS/ ASYNCHRONOUS OPERATION

The VSP may be programmed via the SYN bit in the mode register to operate either synchronously or asynchronously in its manner of sensing the input signals for read/write control (\overline{RD} , \overline{WR} , \overline{SUS}) and bus arbitration (\overline{BACK}). This issue is pertinent for data and instruction fetch in two cases:

- 1) during the slave mode when the host is master of the bus and is performing memory-mapped reads or writes with the VSP;
- 2) during the master mode when the VSP must sense the states of the \overline{SUS} and \overline{BACK} input signals.

In the synchronous mode, the VSP requires that the external system meet rigid setup-and-hold times, and the interface operates on a continuous (consecutive) cycle basis. It is imperative that the external signals be synchronized with the VSP in this mode. The host may not remove \overline{SUS} or \overline{BACK} (for example) at random times. (Setup and hold times for the synchronous mode are given in Section 5.2)

The asynchronous mode is provided for situations in which such specifications cannot be guaranteed, and the VSP will operate with relaxed setup-and-hold times. This will not necessarily result in continuous-cycle fetching, and in general the memory and registers access will be slower if the specified setup-and-hold times are not met.

Asynchronous timing is illustrated in Section 5.2, Figure 26. Simply, the setup time for the VSP to recognize external input signals is relative to the event of CLK rising with ICLK high—i.e., one clock cycle in advance of the event of CLK rising with ICLK low from which the internal signals are specified. **NOTE:** This would be the pertinent event for both internal and external signals in the synchronous mode. If control-signal setup times are not met on a particular cycle, the VSP will recognize the signal on the next cycle.

2.4.2 EXTERNAL MEMORY SPEED AND THE $\overline{\text{SUS}}$ PIN

If the VSP has guaranteed access to external memory as defined by the conditions below, then it should be operated in the synchronous mode, and can perform at maximum efficiency for data fetching, acquiring a data sample on every cycle:

- (i) the memory will *always* be ready for access;
- (ii) the setup-and-hold times will be met.

The number of clocks per sample (NCS) should be set to match the speed of the external memory:

NCS=0 for 2 ICLK per sample data fetch
NCS=1 for 1 ICLK per sample data fetch

(Refer to Section 5 to determine specifications of memory access times for each case).

However, if the first condition is not true, then the suspend ($\overline{\text{SUS}}$) signal must be used. If the latter condition is not true, then the asynchronous mode must be used. There are three motivations for using the $\overline{\text{SUS}}$ signal in a synchronous system:

- (i) the memory is shared between the VSP and some other user;
- (ii) the system contains multiple memories of mixed speed, or with at least one of them being shared;
- (iii) the memory access is too slow even for 2 ICLKs per sample.

An important issue is whether the memory must be checked for readiness on every word accessed, or only a block-accessed basis. The design of the system will determine this, and the design of the bus arbitration logic will reflect it.

For word-ready status, the VSP must be operated in a 2 ICLK per sample mode, and also the synchronous mode. The (external) arbitration logic may then inspect the address issued by the VSP, and conditionally set $\overline{\text{SUS}}$ on a sample-by-sample basis. In this case, $\overline{\text{SUS}}$ works as a traditional RDY signal.

For block-ready status, the VSP may be operated in either a 1 ICLK per sample or a 2 ICLK per sample mode, and either synchronously or asynchronously. At the beginning of a transfer, the VSP will check the $\overline{\text{SUS}}$ signal, and add wait states until it is HIGH. Once the transfer begins, the arbitration logic must dedicate bus access to the VSP for as long as the VSP holds $\overline{\text{RD}}$ or $\overline{\text{WR}}$ low, indicating this by holding $\overline{\text{SUS}}$ HIGH.

2.4.3 BUS ARBITRATION AND PRE-EMPTION

Figure 9 shows at a system level how the VSP DMA structure operates. The VSP requests the bus by asserting the bus request (\overline{BRQ}) pin, and takes control over the bus only when the host processor (or arbiter) asserts the bus acknowledge (\overline{BACK}) pin in response. As long as \overline{BACK} is active, the VSP may control and use the bus. The VSP generates data addresses and controls data transfers using the \overline{RD} , \overline{WR} , and \overline{DSTB} pins. Please refer to the diagrams in Section 5.2 for timing details of data I/O and master-mode instruction fetching.

The host always maintains control of bus arbitration and has the right to force the VSP off of the bus at any time by removing the \overline{BACK} pin. If the host deactivates the \overline{BACK} pin while the VSP is using the bus (thus signaling termination of bus grant), the VSP will release the bus in one of two ways:

- 1) if the VSP is fetching data, the bus will be relinquished after emptying the fetch pipeline;
- 2) if the VSP is fetching instructions, the VSP will release the bus after the FIFO is full.

The host is notified about the bus release by deactivation of the \overline{BRQ} pin. In a similar manner, if the VSP deactivates \overline{BRQ} while \overline{BACK} is active, it is important for the host to remove \overline{BACK} prior to the VSP reassertion of \overline{BRQ} . This is especially important to notice when the VSP is being used with $CRQ=0$ where the VSP may generate one free-bus ICLK cycle between data I/O instructions. *If the VSP finds \overline{BACK} active when \overline{BRQ} is asserted, the VSP will await the deactivation and reactivation of \overline{BACK} again before assuming control over the bus.*

If the VSP still requires the bus (after a pre-emption by the host) for completion of an interrupted data fetch, it will reactivate \overline{BRQ} after one ICLK has completed. The host should not reactivate \overline{BACK} (after it is deactivated) before \overline{BRQ} is deactivated in response.

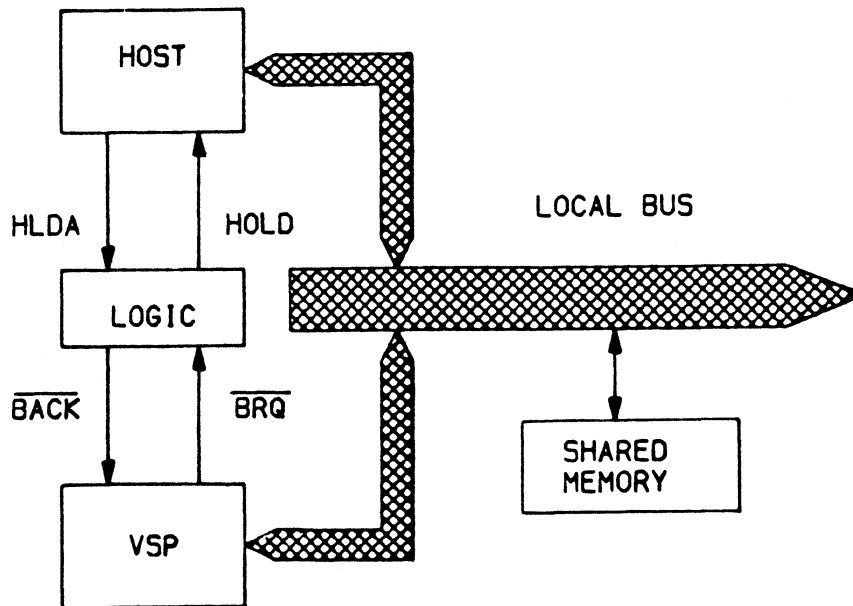


FIGURE 9. BUS ARBITRATION AND PRE-EMPTION MECHANISM.

3.0 DETAILED DESCRIPTION

This section contains detailed descriptions of the VSP initial conditions, external pins, and internal memory and registers.

3.1 INITIAL CONDITIONS

Upon either a hardware or software reset of the VSP, it assumes the initial conditions shown in Table 2. The VSP may be “awakened” from the idle condition to begin instruction fetching in a manner described in section 2.1.

Table 2. Initial VSP conditions—effect of reset.

- All activity terminates—VSP enters idle condition.
- INT pin set LOW
- Phase of ICLK set to commence HIGH
- Interrupt status bits in status register reset
- Maximum scale register cleared
- Scale register pointer set to least-significant nibble
- Mode register bits set as follows:

| | | |
|--------|-------|------------|
| SYN =0 | RSS=1 | NCS=0 |
| NMS=1 | INL=0 | CYCMEM=111 |
| CRQ=1 | | |

3.2 INTERFACE SIGNALS

Figures 1 and 2 show the logical and actual VSP pinouts respectively. Please refer to the timing diagrams and event timing-table (Table 7) at the end of the data sheet for details of all timing information. Some of the VSP control signals can be programmed to operate either synchronously or asynchronously as described in section 2.4.1.

Vcc +5V (± 5%) power supply input.

Vss Power supply ground input.

CLK (Clock) Input
 This input pin provides the system clock to the VSP. The maximum clock frequency is 20 MHz; the minimum is 5 MHz. The CLK input frequency is internally divided by two to generate the ICLK output.

ICLK (Internal Clock) Output
 The signal on this output pin is derived by dividing the CLK input signal by two. All internal activity is synchronized with ICLK transitions; ICLK can therefore be used to synchronize external signals with internal VSP activity. ICLK has a 50% duty cycle and is set HIGH following a reset to the VSP.

$\overline{\text{RESET}}$ (Reset) Input
 This active low input pin terminates all VSP activity and forces it into an idle state. $\overline{\text{RESET}}$ is internally synchronized with the LOW-to-HIGH transition of ICLK following its activation. The hardware reset pin provides the same function as the software RST bit in the mode register. Initial VSP conditions assumed upon reset are shown in Table 2.

DB0-15 (Data Bus 0 - 15) Input/Output
 These 16 bidirectional pins are used for transferring data and instructions between the VSP and the external system (host controller, memories, etc.). When the VSP does not have control of the bus, the data bus pins float in a high-impedance state.

A0-15 (Address Bus 0 - 15) Input/Output
 These 16 bidirectional address bus pins are used by: a) the VSP as outputs for addressing external memory, and b) by an external controller (host) as VSP inputs for addressing the internal memory (RAM, LUT, and registers). Only A0-A9 are needed for addressing internal VSP memory; A10 - A15 are “don’t care”. When the VSP does not have control of the bus, the address bus pins float in a high-impedance state.

D/\overline{C} (Data/Code) Output
 This Data/Code output signal distinguishes between the external memory access cycles of data fetch and instruction (program code) fetch. It can be used to reference two separate 64K word memories, one used for data and the other for program instructions, thereby extending the actual available memory directly accessible to 128K words. D/\overline{C} is HIGH during data fetch and LOW during instruction fetch. D/\overline{C} floats in a high-impedance state whenever the VSP does not have control of the bus.

$\overline{\text{RD}}$ (Read) Input/Output
 This bidirectional, active LOW signal is used by: a) the VSP when reading instructions and data from external memory (output), and b) the external host when reading data from the VSP internal memory (input). $\overline{\text{RD}}$ is an output whenever the VSP has control of the bus and an input when enabled by the $\overline{\text{CS}}$ input pin. At other times $\overline{\text{RD}}$ remains in a high-impedance condition. $\overline{\text{RD}}$ will remain continuously LOW during multiple-data-sample (vector) VSP read operations. As an input, $\overline{\text{RD}}$ is programmable via the VSP Mode Register to be either synchronous or asynchronous with ICLK.

$\overline{\text{WR}}$ (Write) Input/Output
 This bidirectional, active LOW signal is used by: a) the VSP when writing data into external memory (output), and b) by the external host when writing instructions and data into the VSP (input). $\overline{\text{WR}}$ is an output whenever the VSP has control of the bus and an input when enabled by the $\overline{\text{CS}}$ input pin. At other times $\overline{\text{WR}}$ remains in a high-impedance condition. $\overline{\text{WR}}$ will remain continuously LOW during vector VSP write operations. As an input, $\overline{\text{WR}}$ is programmable via the VSP Mode Register to be either synchronous or asynchronous with ICLK.

$\overline{\text{DSTB}}$ (Data Strobe)

Input/Output

This bidirectional, active LOW pin is used by: a) the VSP to strobe data into external memory (output), and b) the host to write data into the VSP (input). $\overline{\text{DSTB}}$ remains in a continuous LOW state during a VSP read from external memory. $\overline{\text{DSTB}}$ is an output whenever the VSP has control of the bus and an input when enabled by the $\overline{\text{CS}}$ input pin. At other times $\overline{\text{DSTB}}$ remains in a high-impedance condition.

$\overline{\text{CS}}$ (Chip Select)

Input

This active LOW input pin allows the host controller to use the VSP control signals (A0-A9, DB0-DB15, $\overline{\text{WR}}$, $\overline{\text{RD}}$ and $\overline{\text{DSTB}}$) for reading from or writing to internal VSP memory or registers. The $\overline{\text{WR}}$, $\overline{\text{RD}}$, and $\overline{\text{DSTB}}$ control signals as well as A0-A9 are all inputs during host read/write operations. $\overline{\text{CS}}$ is ignored when the $\overline{\text{BACK}}$ input signal is active (the VSP is master of the bus).

$\overline{\text{SUS}}$ (Suspend)

Input

This active HIGH input pin is used for coordination of operations during VSP external memory access. If the VSP, upon sampling $\overline{\text{SUS}}$, finds it in a LOW condition, it will extend the memory cycle by 1 ICLK, and sample $\overline{\text{SUS}}$ again 1 ICLK later. $\overline{\text{SUS}}$ is programmable through a bit in the VSP mode register to be either synchronous or asynchronous with ICLK.

$\overline{\text{BRQ}}$ (Bus Request)

Output

This active LOW output pin notifies the host of a VSP request for bus control.

$\overline{\text{BACK}}$ (Bus Acknowledge)

Input

This active LOW input pin notifies the VSP that the host has granted control of the bus to the VSP. If $\overline{\text{BACK}}$ returns HIGH while the VSP has control of the bus, the VSP will immediately begin the process of relinquishing control of the bus and will notify the host when it has done so by setting $\overline{\text{BRQ}}$ high.

INT (Interrupt)

Output

A HIGH on this output pin notifies the host controller of a VSP interrupt. INT is enabled by the VSP after the internal setting of any of the bits in the status register.

3.3 INTERNAL MEMORY AND REGISTERS

The organization of the VSP memories and registers is designed to support vector DSP operations. The memories and registers are shared by both the bus-interface unit and the execution unit. The registers program the operation of the VSP as well as provide status information to the host about internal VSP activity. Table 3 summarizes the registers, accumulators, data RAM, and sinusoidal look-up table in the VSP.

| Register Name | Hex Address | Length (Bits) | Access |
|----------------------------|-------------|---------------|--------------|
| Mode | 300 | 16 | (write) |
| Instruction Base/Start | 306 | 16 | (write) |
| FIFO (buffering) [1] | 302 | 192 | (write) |
| FIFO (execute immediately) | 303 | NA | (write) |
| Old Maximum Scale | 304 | 4 | (write) |
| Scale Vector RAM | 310-31F | 16 | (read/write) |
| Maximum Scale | 302 | 4 | (read) |
| Scale | 303 | 16 | (read) |
| Next Fetch Address | 300 | 16 | (read) |
| Status | 301 | 16 | (read) |
| Imaginary Accum (MSB) | 304 | 8 | (read) |
| Imaginary Accum (LSB) | 305 | 16 | (read) |
| Real Accum (MSB) | 306 | 8 | (read) |
| Real Accum (LSB) | 307 | 16 | (read) |
| RAM [2] | 000-0FF | 4864 | (read/write) |
| Sine/cosine LUT | 100-1FF | | (read) |

[1] 12 16-bit words stores up to four three-word instructions.

[2] 256 19-bit words organized as 128 complex words of 38 bits.

Table 3. Tabular listing of the internal VSP registers and their addresses.

3.3.1 STATUS REGISTER

This 16-bit register contains all status information about the VSP except for scaling. Scaling information is contained in the scaling registers. The format of the status register is shown in Figure 10. Status register information may be obtained from the VSP in one of two ways:

- 1) reading directly from its memory-mapped address, or
- 2) The VSP can write status information to external memory when the STI (Store Information Registers) instruction is executed.

The names and functions of each of the bits in the status register are as follows:



FIGURE 10. VSP STATUS REGISTER FORMAT.

ARINSCO (Arithmetic Instruction Code). These five bits hold the operation code of the most recently executed arithmetic instruction. The operation code of each instruction corresponds to the five most significant bits of the first word of the instruction.

MIC (Move Instruction Code). MIC is coded to show the last move instruction executed as shown below:

- 0 - LD, LDSM, or JMPI
- 1 - ST, STI, or STB

FIFOSTAT (FIFO Status). FIFOSTAT provides a binary number from 0 to 4 indicating the number of empty instruction slots in the instruction FIFO.

- 000 FIFO is full (no available slots).
- 001 one available slot.
- .
- .
- .
- 100 FIFO is empty (all four slots are available).

NOTE: When the status register is written to external memory with the STI instruction, the FIFOSTAT field indicates the number of instruction slots empty *after* completion of the instruction. The exception to this is when STR=1 in the STI instruction (status register written first), where FIFOSTAT equals the number of instruction slots empty *before* the execution of the STI instruction.

IMI (Interrupt on Move Instruction). IMI is set at the completion of an instruction which moves data to or from the VSP.¹

IAI (Interrupt on Arithmetic Instruction). IAI is set at the completion of an arithmetic instruction.¹

IDO (Interrupt on Data Overflow). IDO is set when an overflow is detected in the ALU.¹

ILI (Interrupt on Last Instruction). ILI indicates that both the EU and BIU are idle. ILI is set on the following conditions:¹

- 1) the last instruction existing in the FIFO has completed execution, and
- 2) The bus-interface unit is not attempting to fetch additional instructions. In the VSP master mode, this indicates that the VSP has executed a HLT instruction.

IFO (Interrupt on FIFO Overflow). IFO is set if a new instruction is written into the FIFO by the host while the FIFO is full. The contents of the FIFO are not affected, and the VSP disregards the instruction.

Status bit notes:

- Interrupt status bits IAI, IMI, IDO, ILI, and IFO are cleared when the status register is read. All other status bits are updated as each new instruction is executed; they are not cleared by a read operation.
- IDO, ILI, and IFO are enabled by the corresponding bit in the mode register.
- IAI and IMI are enabled by the EI bit in each instruction.

3.3.2 MODE REGISTER

This 16-bit register programs the operating mode of the VSP. Its format is shown in Figure 6. The mode register may be loaded either:

- 1) writing directly to its memory-mapped address, or
- 2) through executing the LDSM instruction with MD=1.

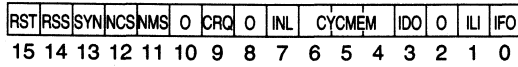


FIGURE 11. VSP MODE REGISTER FORMAT.

The names and functions of each of the bits in the mode register are as follows:

RST (Reset). RST=1 triggers the generation of an internal pulse which resets the VSP. The duration of the reset procedure is three ICLKS, and the procedure begins two ICLKS after the RST bit is set. The RST bit provides the same function as the hardware reset controlled by the RESET pin. VSP initial conditions are shown in Table 2.

RSS (Restart Scale). RSS=1 resets (to 0) both the maximum scale register and the pointer to the scale register. This clears the maximum scale register and causes the next scale factor (generated from an FFT instruction) to be written to the least significant nibble of the scale register.

SYN (Synchronization). SYN programs whether input control signals RD, WR, SUS, and BACK operate synchronously (SYN=1) or asynchronously (SYN=0) with ICLK. Additional discussion on synchronous versus asynchronous operation is contained in section 2.4.1.

NCS (Number of Clocks Per Sample). NCS programs the number of ICLKs required for external data memory access by the VSP. *NCS is not used for instruction fetch; instruction fetch always assumes two ICLKS per memory fetch.* After a reset NCS=0 .

- NCS=0 - 2 ICLKs per sample for data fetch
- NCS=1 - 1 ICLK per sample for data fetch

NMS (Number of RAM sections). NMS programs the number of sections into which internal VSP RAM is partitioned. Two RAM sections are used for concurrent operation where the EU can execute ALU instructions in one section while the BIU simultaneously executes I/O instructions in the other section.

- NMS=0—2 internal RAM sections.
- NMS=1—1 internal RAM section.

CRQ (Continuous Bus Request). Programming CRQ to 1 allows the VSP to retain the data bus (BRQ remains active) between consecutive memory or memory/ALU instructions.

Programming CRQ to 0 will force the VSP to provide one free ICLK between consecutive memory instructions. If the instruction fetch unit is waiting for the bus, the VSP will transfer control of the bus *internally* from data fetch to instruction fetch. If the instruction fetch unit does not take control of the bus during this cycle, the VSP will disable BRQ for a single ICLK. Additional discussions on the use of the CRQ bit is contained in section 2.1.4.

- CRQ=0 one free ICLK between consecutive memory instructions.
- CRQ=1 no free ICLK between consecutive memory instructions.

INL (Instruction Length). INL controls whether one-word VSP instructions are fetched with their defined length, or with a fixed length of three words per instruction.

Additional discussion of INL occurs in section 2.1.3.

- INL=0 Normal (one to three word) instruction length
- INL=1 Fixed three-word instruction length

CYCMEM (Cyclic Memory). CYCMEM defines the size of data memory blocks used for instructions which access external memory. When the VSP reaches the end of this memory block during a load or store instruction, it loops back to the beginning of the block. The value of the three bits in binary is added to nine to give the power of two representing the number of words in the memory block.

- CYCMEM=000-> 2⁽⁹⁺⁰⁾-> 512-word blocks
- .
- .
- .
- CYCMEM=111-> 2⁽⁹⁺⁷⁾-> 64K-word blocks

For example, the following instruction:

```
LD NMPT=8, MBA=1020
```

executed while CYCMEM=0 in the mode register defines a cyclic memory block size of 512 words. When the instruction is executed, eight samples will be loaded from memory with the following addresses: 1020, 1021, 1022, 1023, 512, 513, 514, and 515.

If the VSP uses the STB (store backwards) instruction, then on reaching the *beginning* of the memory block, the address generator will loop back to its end.

After a reset CYCMEM=111.

IDO (Interrupt on Data Overflow). IDO=1 allows a hardware interrupt to be generated when the IDO bit in the status register is set by the VSP. IDO=0 prohibits a hardware interrupt from being generated based on the IDO bit in the status register.

ILI (Interrupt on Last Instruction). ILI=1 allows a hardware interrupt to be generated when the ILI bit in the status register is set by the VSP. ILI=0 prohibits a hardware interrupt from being generated based on the ILI bit in the status register. After a reset ILI=0.

IFO (Interrupt on FIFO Overflow). IFO=1 allows a hardware interrupt to be generated when the IFO bit in the status register is set by the VSP. IFO=0 prohibits a hardware interrupt from being generated based on the IFO bit in the status register. After a reset IFO=0.

3.3.3 DATA RAM

The VSP data RAM is structured as 128, 38-bit complex words. A diagram of the organization of the RAM is shown in Figure 12. The real portion of the complex word occupies the first 19 bits, and the imaginary portion occupies the second 19 bits. Only the lower 17 bits out of the 19 dedicated to the storage of each part of the complex words are used for arithmetic computations. The remaining upper two bits are “guard bits” which allow temporary storage of overflows generated during block floating-point FFT computations. *The user does not have access to the guard bits.* Although internal arithmetic accuracy is 17 bits, only

the 16 most significant of these bits are accessible by devices external to the VSP. The LSB of the 17 bits is used internally for extra precision in rounding.

All arithmetic instructions operate on data vectors stored in the RAM. Since the data RAM is 128 complex-words in length, the maximum vector length available for operations is also 128 samples, regardless of whether the data is real, imaginary, or complex. When the RAM is partitioned for concurrent I/O and ALU operations (NMS=0), each RAM section is 64 complex words in length. The maximum vector length on which instructions can then operate in the concurrent mode is 64 samples.

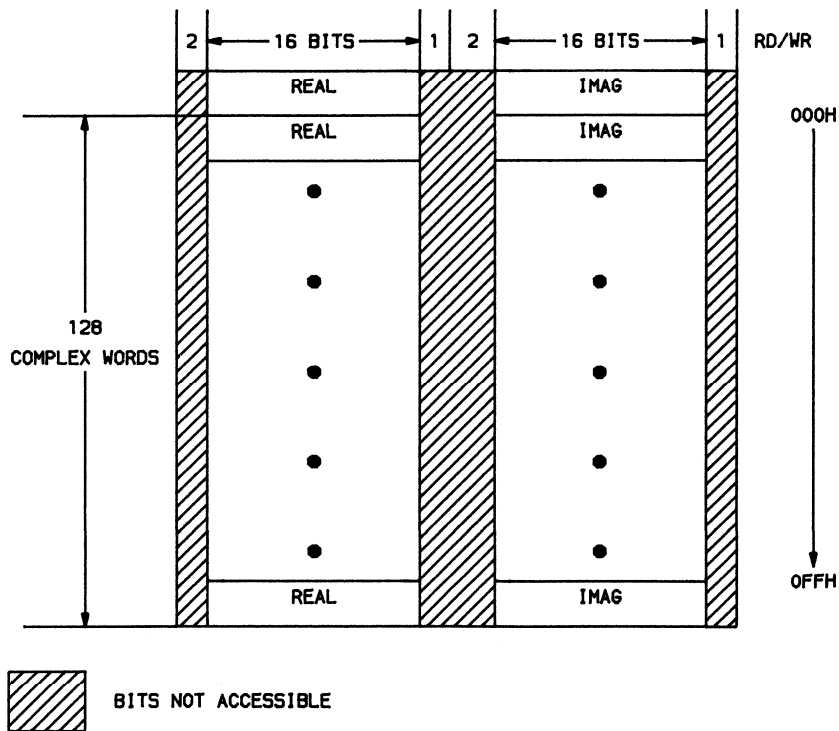


FIGURE 12. DATA MEMORY ORGANIZATION.

3.3.4 INSTRUCTION FIFO

The instruction FIFO consists of 12 16-bit registers organized as four 3-word instruction “slots”. A diagram of the organization of the instruction FIFO is shown in Figure 13(a). Each FIFO slot can store only one VSP instruction, even though some instructions may be less than three words in length. The VSP fetches its own instructions in the *master mode*, whereas the host processor must write instructions to the FIFO in the *slave mode*. In either mode, new instructions are placed at the end of the FIFO by the BIU while instructions at the beginning of the FIFO will be executed first.

When the VSP is operated in the slave mode, the host controller will write instructions directly into the VSP FIFO. The FIFO is mapped into two external memory-space addresses as shown in Table 3: a *buffering* address, and an *immediate-execution* address.

The buffering address is used for placing VSP instructions into the instruction FIFO without their execution; the instructions will be queued for later execution. The immediate-execution address is used for starting instruction execution beginning with the first instruction in the FIFO.

When the final word of an instruction is written to the immediate-execution address, VSP instruction execution will begin. All subsequent instructions written to the FIFO to either the buffering or immediate-execution address will be properly queued for their eventual execution. Instruction execution will continue until the instruction FIFO becomes empty. When the final instruction in the FIFO has completed execution, the VSP will enable the ILI bit in the status register.

If the host attempts to fill the FIFO beyond its four-word capacity, the VSP will set the IFO bit in the status register. All extra words written to the FIFO will be lost, and must be written to the FIFO a second time when an instruction slot becomes available.

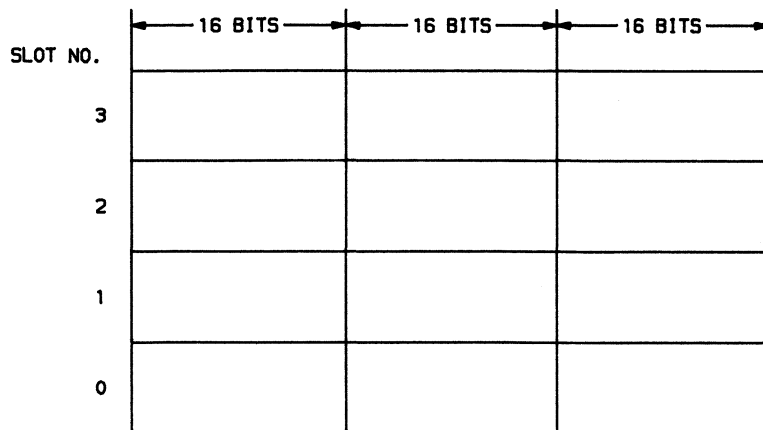


FIGURE 13(A). INSTRUCTION FIFO.

Because of the maximum 4-instruction length of the FIFO and internal pipelining considerations, three instructions plus up to *two words* of the fourth instruction may be buffered *without initiating execution*. The final word written to the FIFO must be written to the “immediate-execution” address.

NOTE: *If the host writes the twelfth word to the FIFO’s buffering address (302H) without initiating execution, a software or hardware reset must be performed in order to clear the FIFO. The host must reload the instruction FIFO before further execution can begin.*

In the master mode, the VSP fetches its own instructions and data after it has been started by a host processor. Additional discussion of VSP instruction fetching in the master and slave modes takes place in section 2.1.

3.3.5 NEXT FETCH ADDRESS REGISTER

This 16-bit readable register holds the address of the next instruction to be fetched by the VSP in the master mode. It may be read by the host by using its memory-mapped address; its contents may be written to external memory by using the STI (Store Information Registers) instruction. Figure 13(b) shows the organization of the next fetch address register.

3.3.6 INSTRUCTION BASE/START REGISTER

This 16-bit register is used by the host to write a program base address that points to external memory. Writing an address to this register commands the VSP to begin instruction fetch and execution in the master mode at the base address pointed to by the register. Instruction fetch will continue until a HLT instruction is fetched by the VSP. Figure 13(c) shows the organization of the instruction base/start register.

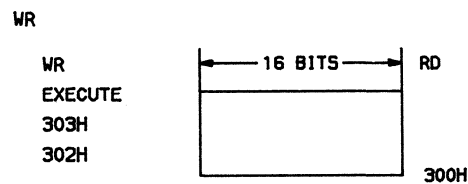


FIGURE 13(B). NEXT FETCH ADDRESS REGISTER

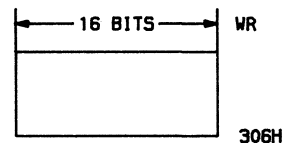


FIGURE 13(C). INSTRUCTION BASE/START REGISTER.

FIGURE 13(A-C). INSTRUCTION FETCH REGISTERS.

3.3.7 SCALE RAM

The scale RAM consists of 16 words by 16 bits organized as 64 4-bit nibbles. The scale nibbles are used with the SCL (Scale) instruction for scaling of data vectors of varying lengths as defined in the instruction parameter fields. Figure 14(a) shows a diagram of organization of the scale RAM. Each scale nibble stores the number of right-shifts to be performed on the data vectors. Its primary use is for preserving the dynamic range of input signals during FFT computations; it can also be used in cases where data normalization is required.

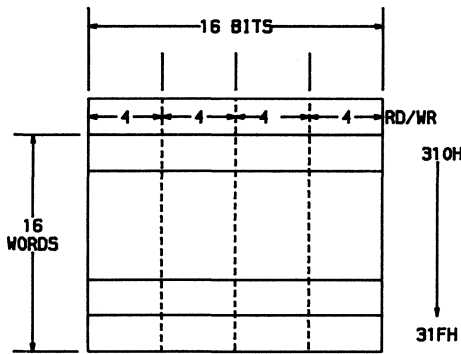


FIGURE 14(A). SCALE RAM.

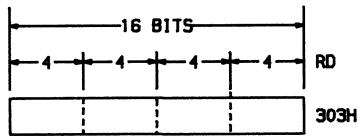


FIGURE 14(B). SCALE REGISTER.

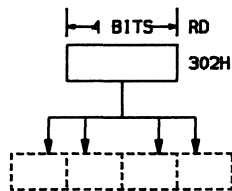


FIGURE 14(C). MAXIMUM SCALE REGISTER.

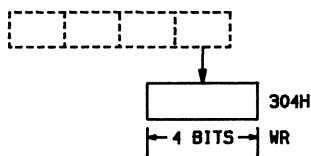


FIGURE 14(D). OLD MAXIMUM SCALE REGISTER.

FIGURE 14(A-D). SCALE RAM AND REGISTERS.

The Scale RAM can be loaded by either:

- 1) writing directly to its memory-mapped address, or
- 2) through executing the LDSM instruction with MD=0.

3.3.8 SCALE REGISTER

This 16-bit register is used as four 4-bit nibbles for storage of up to four scale factors for FFT computations. The organization of the scale register is shown in Figure 14(b). A 4-bit scale factor is generated and stored in the register each time an FFT instruction is executed with block floating-point arithmetic. A scale register pointer indicates which of the four nibbles will store the next FFT scale factor; the pointer is incremented after each scale factor is stored. Because the scale register will hold a maximum of four scale nibbles, it must be stored to external memory after every four FFT instructions in order not to overwrite the scale nibbles by subsequent FFT instructions. The scale register pointer reverts to the least significant nibble in the following cases:

- 1) after the most significant nibble is used,
- 2) following a chip Reset, or
- 3) through executing the LDSM instruction with the parameters MD=0 and UP=1.

Scale register contents may be obtained from the VSP in one of two ways:

- 1) reading directly from its memory-mapped address, or
- 2) through executing the STI (Store Information Registers) instruction with STR:5.

3.3.9 MAXIMUM SCALE REGISTER

This 4-bit register is used for keeping track of the largest scale factor generated during a sequence of FFT instructions. It contains the largest scale factor generated since the last scale reset. It is reset to zero by either a system reset or by execution of the LDSM instruction with the parameters MD=0 and UP=1. The organization of the maximum scale register is shown in Figure 14(c).

The maximum scale register contents may be obtained from the VSP in one of two ways:

- 1) reading directly from its memory-mapped address, or
- 2) through executing the STI (Store Information Registers) instruction with STR:6.

When the maximum scale register contents is read using its memory-mapped address, the 4-bit scale factor will be repeated identically in each of the four 4-bit nibbles read by the host.

3.3.10 OLD MAXIMUM SCALE REGISTER

This 4-bit register is updated with the current value in the maximum scale register by the execution of the LDSM instruction with the parameters MD=0 and UP=1. The organization of the old maximum scale register is shown in Figure 14(d).

3.3.11 ACCUMULATORS

The real and imaginary accumulators are each 25 bits in length, but only the 24 most significant bits are externally accessible through the bus interface. The LSB of each accumulator is used internally for extra precision in rounding. The organization of the accumulators with their memory-mapped addresses shown in Figure 15. Each accumulator can be read directly as two successive memory addresses. The first address (memory-mapped address with the LSB=0) holds the eight most-significant bits of the accumulator in its least-significant byte. The most-significant byte of this word contains an extension of the accumulator sign bit. The second accumulator address contains the 16 least-significant bits of the accumulator which are addressable externally. Accumulators may also be written to memory by the VSP through execution of an STI (Store Information Registers) instruction with STR:1.

Instructions which use accumulators will first clear the appropriate accumulator before using it. Instructions using only one of the two accumulators will only clear the accumulator which will be used; the contents of the other accumulator will not be affected.

3.3.12 SINUSOIDAL LOOK-UP TABLE (LUT)

The sinusoidal look-up table is an on-chip ROM that contains 256 16-bit values representing the sines of the angles from 0 to $\pi/2$ in increments of $\pi/512$. The organization of the look-up table is shown in Figure 16. Because of the symmetry of sinusoids, the LUT values can be extended to represent 1024 sinusoidal values from 0 to 2π . A seventeenth bit is generated in hardware allowing representation of the sign bit of the angle selected. The 17-bit result has an implied binary point to the right of the sign bit. The representation of 1.0 is accurately generated internally using special logic.

The LUT coefficients are used to compute FFTs of up to 1024 points in length without supplying external coefficients. Transforms larger than 1024 points require the host to supplement the coefficients from external memory. The LUT coefficients are also used by the MODLT and DEMO instructions for modulation and demodulation of signal vectors respectively.

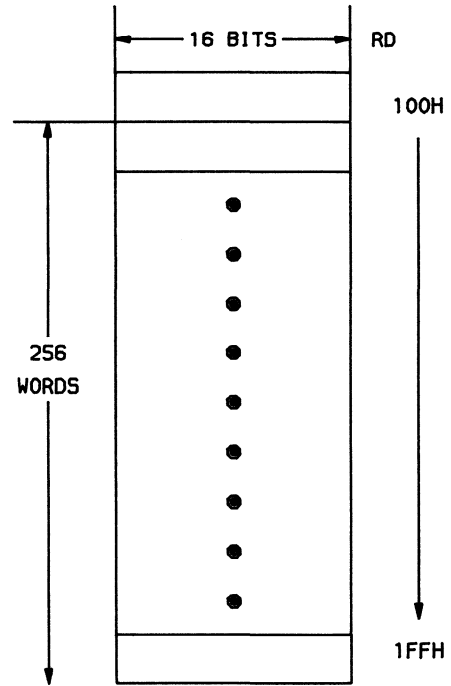


FIGURE 16. LOOK-UP TABLE ORGANIZATION.

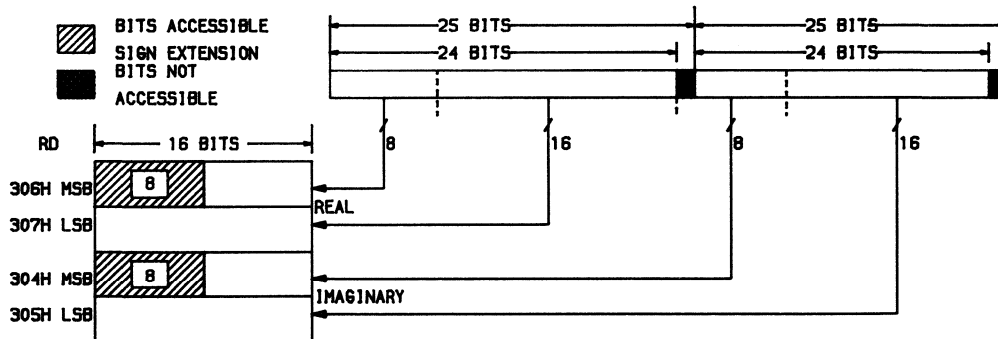


FIGURE 15. VSP ACCUMULATORS.

4.0 INSTRUCTION SET

This section describes each of the 23 instructions provided by the VSP. The description of the instructions is broken into four sections: data I/O, ALU/external memory, internal memory, and control; each of the instructions is covered in one of the categories. As some of the parameters are common to a large number of instructions, these are described in section 4.2.

The description of the instruction set is partitioned into the same four categories. Each of the VSP instructions and its respective parameters are covered in detail in one of the four categories.

In the format fields for each instruction, fixed bits are shown as 0 or 1; variable fields are shown by label (field name); “don’t care” bits are left blank. The bits in each word are numbered from 0 on the right end (LSB) to 15 on the left end (MSB).

Parameters which are common to a majority of instructions are listed alphabetically in the following *common instruction parameters* section (4.2). This eliminates the duplication of discussing these parameters repetitively for each instruction.

4.1 LITERAL AND LOGICAL PARAMETER VALUES

When discussing the instruction parameters, reference is made to *literal* and *logical* values. Literal values are the binary executable parameter values used by the VSP for execution of the instructions. While these parameter values are unambiguous for the VSP, they can lead to confusion for users. For this reason, the concept of logical parameter values was developed.

For example, the NMPT field of VSP instructions has a logical meaning of defining the number of data samples on which an instruction is to operate. This varies in length from 1 to 128 data samples. The NMPT field would require eight bits in order to represent the logical value 128. However, the VSP is able to encode these bits in a manner which is more efficient in silicon (saving a bit) as shown in Table 4 below. Notice that only seven bits are required to represent the 128 values using this implementation.

Table 4. Logical vs literal interpretation for the NMPT parameter.

| <u>Literal</u> | <u>Logical</u> |
|----------------|----------------|
| 000 0001 | 1 sample |
| 000 0010 | 2 samples |
| . | . |
| . | . |
| . | . |
| 111 1111 | 127 samples |
| 000 0000 | 128 samples |

Other VSP parameters—but not all—also have different logical and literal values. These parameters and their translation formulas from a logical to a literal value are always presented for each parameter where this occurs.

NOTE: VSP development tools always allow user programming of parameters as either logical or literal values. This removes the burden of the user having to remember the parameter translations. In general, it is a good idea to always use logical parameter values when writing VSP programs and allow the development tools to translate the logical values into the literal values required by the processor at execution time.

To distinguish between the two, the convention has been adopted of expressing literal assignments using an “=” sign, e.g. NMPT=0 and logical assignments using a “:”, e.g., NMPT:128.

4.2 COMMON INSTRUCTION PARAMETERS

This section lists instruction parameters which are common to a large number of VSP instructions. The parameters are listed alphabetically for quick reference. Reference to the common parameters section is made throughout the instruction set discussion when these common parameters are included in an instruction field.

ADF Arithmetic Data Format: Selects which part (real, imaginary, or complex) of the results from arithmetic instructions are to be stored in internal RAM.

- 00 result is not stored in RAM
- 01 store imaginary part only
- 10 store real part only
- 11 store complex result

If ADF=00 is selected, neither the real nor the imaginary part of memory is stored. However, instructions providing accumulation results will store these summations in the accumulators without storing the intermediate sample results.

CN Constant: Used to specify that a constant in external memory, instead of a vector, is used to operate with the vector in internal RAM.

- 0 use an external vector
- 1 use an external constant

EI Enable Interrupt: Programs whether an interrupt will be generated on the INT pin at the completion of instruction execution or not.

- 0 no interrupt is generated at the completion of instruction execution. The appropriate status bit is set in the status register.
- 1 interrupt generated and status bit set at the completion of instruction execution.

NOTE: ALU/external memory instructions will set the IAI bit (and not the IMI bit).

MBA Memory Base Address: The starting physical address of the data array in external memory. The real portion of the complex word is always accessed first.

MBS Memory Block Size: The number of consecutive real, imaginary, or complex data samples to be transferred in each memory step is defined in size by MSS. When $MSS > MBS$, the remaining number of points in the memory step ($MSS - MBS$) will not be transferred.

| <u>Literal</u> | <u>Logical</u> |
|----------------|----------------|
| 000 | 1 point |
| . | . |
| . | . |
| . | . |
| 111 | 128 points |

Logical 2^N translates to N literal.

For example, the following instruction:

LD NMPT:32, MBS:4, MSS:16;

will divide contiguous external memory into step sizes of 16 samples (MSS:16). When loading the 32 samples (NMPT:32) into VSP memory, the first four (MBS:4) samples from each memory step will be loaded, while the next twelve samples will be ignored. (Please refer forward to the MSS parameter for a description of its operation.)

MDF Memory Data Format: Used to program the interpretation by the VSP of data (real, imaginary, or complex) loaded (LD) into internal memory or stored (ST) out to external memory.

- 00 do not load or store internal memory
- 01 (LD) - sequential data in external memory is loaded into the *imaginary* part of internal memory.
(ST) - data in the *imaginary* part of internal memory is written to sequential locations in external memory.
- 10 (LD) - sequential data in external memory is loaded into the *real* part of internal memory.
(ST) - data in the *real* part of internal memory is written to sequential locations in external memory.
- 11 (LD) - sequential data in external memory is loaded alternately into the *real and imaginary* parts of internal memory.
(ST) - data in the alternating real and imaginary parts of internal memory is written sequentially out to external memory. Data words alternate as: *real, imaginary, real, imaginary*, etc.

MSS Memory Step Size: Defines a *step size* in words of external memory which either all of, or a portion of, may be transferred. MSS equals the number of points specified in MBS plus the number of points not transferred. An example of the usage of the MSS parameter exists in the previous discussion of the MBS parameter.

| <u>Literal</u> | <u>Logical</u> |
|----------------|----------------|
| 000 | 2 points |
| 111 | 256 points |

Logical $2^{(N+1)}$ is translated to N literal.

NMPT Number of Points: Defines the number of *samples* of data on which to operate. A sample may consist of either real, imaginary, or complex data, as interpreted by the appropriate MDF or ADF parameter.

| <u>Literal</u> | <u>Logical</u> |
|----------------|----------------|
| 000 0001 | 1 sample |
| . | . |
| . | . |
| . | . |
| 111 1111 | 127 samples |
| 000 0000 | 128 samples |

Logical 128 is translated to 0 literal.

The following parameters have definitions which are the same as those given in the common parameters section: RS, EI, and MBA.

Parameters which are unique to the STI instruction:

NMPT **Number of Information Registers:** To store to external memory. Note that this definition of NMPT is different from the one used in earlier instructions. Values can be 1 through 8 decimal, expressed as 000 0001 through 000 1000 binary.

STR **Starting Register:** The beginning number in the following list where the count of NMPT registers starts.

The logical numbers and names of the VSP registers are shown as follows for use with the STR parameter:

- 1 - Real Accumulator, LSB
- 2 - Real Accumulator, MSB
- 3 - Imaginary Accumulator, LSB
- 4 - Imaginary Accumulator, MSB
- 5 - Scale Register
- 6 - Maximum Scale Register
- 7 - Status Register
- 8 - Next Fetch Address

| Literal | 1st register stored(logical) | max(NMPT) |
|---------|------------------------------|-----------|
| 011 | begin storage with reg #1 | 8 |
| 010 | begin storage with reg #5 | 4 |
| 001 | begin storage with reg #7 | 2 |
| 000 | begin storage with reg #8 | 1 |

Logical to literal translations:

1->3, 5->2, 7->1, 8->0.

There is an implied relationship between NMPT which defines the number of registers to store and STR which defines the register with which to begin the storage. This relationship is shown in the table above. For instance, if storage begins with logical register number 5 (scale register), the maximum number of registers which can be stored is 4. This is because the register address counter will not roll over back to register number 1.

OR **Order:** Order of arrangement of two of the accumulator registers above

- 0 numbers 2 and 3 are interchanged
- 1 the order above stands unchanged

NOTE: The STI instruction must be executed with the NCS parameter in the mode register programmed to 0. If external memory normally operates with NCS:1, it is necessary to execute an LDSM instruction with NCS:0 prior to the STI instruction, and LDSM with NCS:1 after the STI instruction. An example of this is shown below:

```

.
.
.
user code with NCS:1
.
.
.
LDSM — programs NCS:0
STI
LDSM — programs NCS:1
.
.
.
user code with NCS:1
.
.
.
    
```

4.4 ALU/EXTERNAL MEMORY INSTRUCTIONS

This section covers the four ALU instructions which operate on two separate data vectors. One of the vectors must already exist in the internal VSP memory, and the other must exist in external memory. All instructions in this section are three words in length.

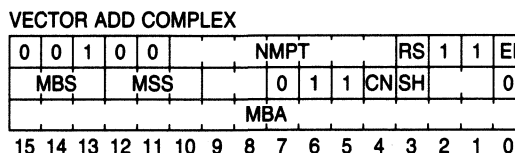
The instructions covered in this chapter are:

- ADDC** (Vector Add Complex)
- ADDR** (Vector Add Real)
- MLTC** (Vector Multiply Complex Accumulate)
- MLTR** (Vector Multiply Real Accumulate)

ADDC Vector Add Complex

ADDC adds the real and imaginary parts of a complex vector existing in external memory to the respective real and imaginary parts of a complex vector in internal RAM. The vector summation is stored in internal memory. External memory remains unchanged.

ADDC is a three-word instruction.

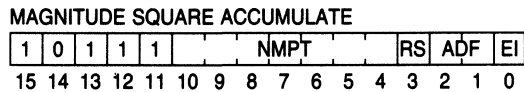


All of the parameters contained in the ADDC instruction are defined in the common parameters section.

MGSQ Magnitude Square/Accumulate

MGSQ calculates the square of the magnitude for each complex sample in internal memory. Each result is scaled down by 2 (right-shifted) to prevent overflow, and is written into the real part of the internal memory. The summation of the magnitude-squared elements is stored in the real accumulator. ADF must be 10 or 00. If ADF is 00, only the accumulators are updated.

MGSQ is a one-word instruction.



All of the parameters contained in the MGSQ instruction are defined in the common parameters section.

MODLT Modulate

MODLT multiplies the samples of a complex vector in internal memory by a series of complex coefficients generated from the sine-cosine look-up table. The coefficients are specified in the instruction by a ROM base-address (initial phase) and increment factor (frequency). The resulting complex samples are stored in internal memory.

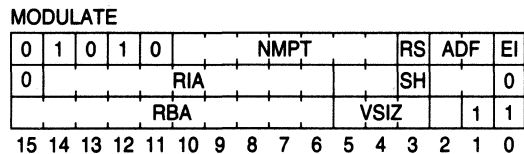
The multiply operation follows the formula:

$$VSPRAM [i] = VSPRAM [i] * (\cos \Theta + j \sin \Theta)$$

where $\Theta = RBA + RIA * i$

The VSP LUT contains 256 sine values from 0 to $\pi/2$. The multiplication performed in the MODLT instruction modulates or frequency translates an input signal by performing an element-by-element complex multiplication with a complex sinusoid. RBA corresponds to the *initial phase* and RIA determines the *frequency* of the sinusoid. MODLT is precisely the same as the DEMO instruction except for the direction of rotation of the complex vector. The imaginary portion of the sinusoid is incremented instead of decremented as in the DEMO instruction.

MODLT is a three-word instruction.



The following parameters have definitions which are the same as those given in the common parameters section: NMPT, RS, ADF, EI and SH.

Parameters unique to the MODLT instruction:

RIA ROM Increment Address: address used to define the incremental angles for successive coefficients

DEGREES*10 is the logical value for RIA.
DEGREES*1024/360 is the literal value for RIA.

RBA ROM Base Address: address of the first value (initial phase) to be addressed from the sinusoidal LUT for the 1024 angles from 0 to 2π .

DEGREES*10 is the logical value for RBA.
DEGREES*1024/360 is the literal value for RBA.

VSIZ Specifies the number of samples beginning with RBA to be addressed from the internal sine/cosine LUT, after which the LUT address rolls back to the RBA value.

| <u>Literal</u> | <u>Logical</u> |
|----------------|----------------|
| 000 | 4 points |
| 001 | 8 points |
| 010 | 16 points |
| 011 | 32 points |
| 100 | 128 points |

SCL Scale

SCL scales an internal complex vector down in magnitude by performing an integer number of right shifts on the samples of the data vector. The number of bits of right-shifting performed is determined by elements of a scale vector which must have been loaded previously into the VSP scale RAM.

The scale vector must be loaded into the VSP scale RAM through the use of the LDSM instruction, or, alternatively written by the host using memory-mapped writes. The scale vector can be 1 to 64 nibbles in length as specified in the SCLVL instruction parameter. Each nibble is a scaling factor from 0 to 15 representing the number of right-shifts—divide-by-twos—to apply to the elements of the internal vector. Also specified is the number of successive points in the data vector to be scaled by the same scaling factor (SCLBL).

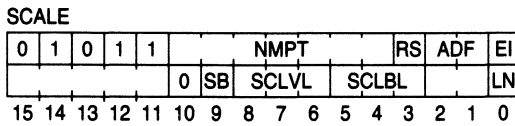
The simplest example is a scale vector the same length as the internal data vector, where each sample in the data vector is scaled by the respective factor in the scale vector. If the number of scale nibbles in the scale vector is exhausted before NMPT points are scaled, the scale vector will be reused as many times as necessary to complete NMPT scale operations. If the scale vector is only one nibble in length, the instruction allows specification of which nibble out of the first four in the Scale RAM is used in the instruction execution.

SCL also sums the scaled results into both the real and imaginary accumulators.

The content of each nibble of the scale vector is the number of right shifts (divide-by-twos) performed on the operand vector as shown below:

| Binary scale nibble | Scaling Result |
|---------------------|-----------------------------|
| 0000 | no effect |
| 0001 | divide by 2 |
| 1111 | divide by 2^{15} (32,768) |

SCL is a two-word instruction. It can be extended to three words in length by using the LN parameter.



The following parameters have definitions which are the same as those given in the common parameters section: NMPT, RS, ADF, and EI.

Parameters unique to the SCL instruction:

SB Subtract: source of the content of the scale factor. SB is used to uniformly scale the output from large FFTs.

- 0 the scale factor is contained in the scale RAM
- 1 the scale factor is the value in the old maximum scale register *minus* the scale RAM value

SCLVL Scale Vector Length: The length in nibbles of the scale vector in the scale RAM

| Literal | Logical |
|---------|------------|
| 000 | 1 nibble |
| 110 | 64 nibbles |

Logical 2^N translates to N literal.

SCLBL Scale Block Length: The number of successive data samples in the VSP RAM to be scaled by the same scale factor.

| Literal | Logical |
|---------|-----------|
| 000 | 1 point |
| 101 | 32 points |

Logical 2^N translates to N literal.

If the scale vector length is one, SCLBL defines the nibble in the first four Scale RAM nibbles to use.

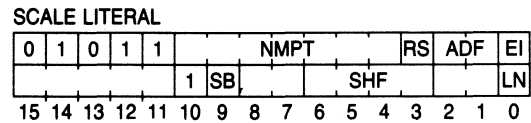
LN word length of instruction:

- 0- used as a three-word instruction
- 1- used as a two-word instruction

When mode-register bit INL=1, LN should be programmed to 0 to ensure that the SCL and SCLT instructions are also three words in length. When INL=0, LN may be programmed as either 0 or 1.

SCLT Scale Literal

SCLT scales an internal complex vector down in magnitude by performing an integer number of right shifts on the samples of the data vector. The number of bits of right-shifting performed is programmed by the constant SHF parameter in the instruction. SCLT is a two-word instruction.



The following parameters have definitions which are the same as those given in the common parameters section: NMPT, RS, ADF, and EI.

Parameters unique to the SCLT instruction:

SHF Shift: the number of right shifts to apply to each vector element

| Literal | Logical |
|---------|-----------------------------|
| 0000 | no effect |
| 0001 | divide by two |
| 1111 | divide by 2^{15} (32,768) |

SB Subtract: Source of the content of the scale factor.

- 0 the scale factor is contained in the scale SHF parameter of the instruction
- 1 the scale factor is the value in the old maximum scale register *minus* the SHF value

LN word length of instruction:

- 0 used as a three-word instruction
- 1 used as a two-word instruction

When mode-register bit INL=1, LN should be programmed to 0 to ensure that the SCL and SCLT instructions are also three words in length. When INL=0, LN may be programmed as either 0 or 1.

4.6 CONTROL INSTRUCTIONS

This section covers the three instructions which control program flow in the VSP. They vary in length from one to three words. The three instructions are:

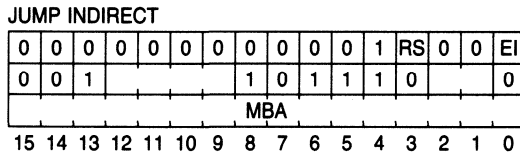
- JMPI** (Jump Indirect)
- HLT** (Halt)
- NOP** (No Operation)

JMPI Jump Indirect

JMPI is the only program-flow control instruction within the VSP. JMPI causes the VSP to load a new instruction fetch address into the next fetch address register. The NFA register is loaded with the contents of the word existing at the memory base address defined by MBA in the instruction word. JMPI is executed by the BIU.

Because JMPI contains the RS parameter, it will be executed concurrently with other instructions if the concurrency rules presented earlier are met.

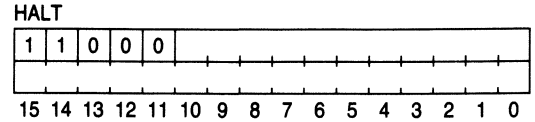
JMPI is a three-word instruction.



HLT Halt

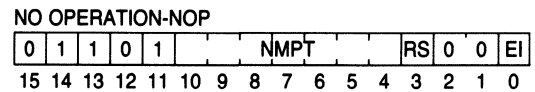
HLT stops the VSP bus-interface unit from fetching additional instructions. It is always used as the last instruction in a program when the VSP is operating in the master mode or when it is otherwise desired to halt instruction fetch. Any instructions executing or queued in the instruction FIFO when a HLT instruction is encountered are not affected, and will complete execution. A HLT instruction written to the VSP in the slave mode has no meaning.

HLT is a two-word instruction, where all but the first five bits are DON'T CARE.



NOP No Operation

NOP is a one-word null instruction which has no effect on the execution of other instructions, nor on registers (except the status registers which are updated) or memory. NOP is implemented as a CMCN instruction with ADF:00. Note that the execution time of the NOP instruction is a function of the NMPT parameter defined in the instruction. This allows variable-length NOP instructions for applications requiring predictable delays. It is sometimes useful to insert NOPs in a program to reserve space for later use or to time operations for real-time applications.



Because the NOP instruction contains the RS parameter and is really a non-functional arithmetic instruction, it will be executed concurrently with other instructions if the concurrency rules presented earlier are met.

All programmable parameters in the NOP instruction are defined in the common parameters section.

4.7 INSTRUCTION CLOCK-CYCLE PERFORMANCE

VSP instructions take varying lengths of time to execute as a function of a number of parameters, such as: external memory speed, number of points on which to operate, and whether data is real or complex. Table 6 presents a summary of the VSP instruction set performance in clock cycles for a number of different parameters.

Table 6. VSP Instruction Clock-Cycle Performance.

| Number of Points | Number of Clock Cycles (100 ns) | | | | | | | |
|-------------------------------|---------------------------------|----|----|----|-----|-----|-----|------|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| Instructions | | | | | | | | |
| LD/STC(1 CK/CY) | 8 | 10 | 14 | 22 | 38 | 70 | 134 | 262 |
| LD/STR ¹ (1 CK/CY) | 7 | 8 | 10 | 14 | 22 | 38 | 70 | 134 |
| LD/STC(2 CK/CY) | 10 | 14 | 22 | 33 | 70 | 134 | 262 | 518 |
| LD/STR ¹ (2 CK/CY) | 8 | 10 | 14 | 22 | 38 | 70 | 134 | 262 |
| JMPI | 8 | - | - | - | - | - | - | - |
| MLTC ² | 19 | 23 | 31 | 47 | 79 | 143 | 271 | 527 |
| MLTR ² | 15 | 17 | 21 | 29 | 45 | 77 | 141 | 269 |
| ADDC(1 CK/CY) | 13 | 15 | 19 | 27 | 43 | 75 | 139 | 267 |
| ADDC(2 CK/CY) | 15 | 19 | 27 | 43 | 75 | 139 | 267 | 523 |
| ADDR ² | 13 | 15 | 19 | 27 | 43 | 75 | 139 | 267 |
| FFT ³ | - | 14 | 34 | 74 | 162 | 362 | 818 | 1850 |
| DEMO | 10 | 14 | 22 | 38 | 70 | 134 | 262 | 518 |
| SCL | 8 | 10 | 14 | 22 | 38 | 70 | 134 | 262 |
| MGSQ | 8 | 10 | 14 | 22 | 38 | 70 | 134 | 262 |
| ABS | 8 | 10 | 14 | 22 | 38 | 70 | 134 | 262 |
| CMCN | 8 | 10 | 14 | 22 | 38 | 70 | 134 | 262 |
| CMLT | 6 | 8 | 12 | 20 | 38 | 68 | 132 | 260 |
| ACCI | 4 | 5 | 7 | 11 | 19 | 35 | 67 | 131 |
| ACCR | 4 | 5 | 7 | 11 | 19 | 35 | 67 | 131 |
| HLT | 0 | - | - | - | - | - | - | - |

1 LDSM and STI correspond to loading and storing real data.

2 The execution time of this instruction is independent of the NCS bit in the mode register.

3 Because the parameters associated with the FFT instruction significantly effect its execution time, execution times can be calculated using the following equation:

$$T = 4*(literal\ NMBT+2)*(number\ of\ passes)+2$$

5.0 TIMING AND SPECIFICATIONS

5.1 EVENT TIMING-DELAY TABLE

Certain events within the VSP often occur after delays that can be measured in multiple input CLK cycles (called T1, see timing diagrams). These delays are summarized in Table 7.

Table 7. VSP Event Timing Delays.

Interrupts

The delay between the event and the INT pin becoming ACTIVE

| <u>Interrupt After:</u> | <u>T1 cycles</u> |
|-------------------------|------------------|
| 1. Execution (IMI, IAD) | 2 |
| 2. Data Overflow (IDO) | 4 |
| 3. FIFO full (IFO) | 8 ¹ |
| 4. VSP Idle (ILI) | 4 ² |

Host Access (Slave Mode or Memory-Mapped Access)

Read—the delay between the host asserting \overline{RD} and data valid.

| <u>When accessing:</u> | <u>T1 cycles</u> |
|------------------------|------------------|
| 1. Scale RAM | 3 |
| 2. Data RAM | 6 |
| 3. ROM, Registers | 2 |

Write—the delay between the host asserting \overline{WR} and data accepted.

| <u>When accessing:</u> | <u>T1 cycles</u> |
|--------------------------|------------------|
| All memory and registers | 3 |

Recovery Time—the delay required between successive host accesses.

After

| | |
|----------|---|
| 1. Write | 6 |
| 2. Read | 4 |

Bus Preemption

The delay between host removal of $\overline{\text{BACK}}$ and VSP removal of $\overline{\text{BRQ}}$.

| <u>Preemption During:</u> | <u>T1 cycles</u> |
|----------------------------------|------------------------------------|
| 1. Instruction Fetch | 40 (max) ³ |
| 2. Data I/O | 9 (max) ³ (2 ICLK/word) |
| | 7 (max) ³ (1 ICLK/word) |

Initiating Execution

The delay before the VSP begins fetching instructions.

| <u>After:</u> | <u>T1 cycles</u> |
|---|---------------------------------|
| 1. Host Write to Instruction Base/Start Reg | 8 (sync mode) 9 (async mode) |
| 2. Execution of JMPI | 6 (after execution) |

Software Reset

| | <u>T1 cycles</u> |
|------------------------------|-------------------------|
| Delay after load of register | 0 |
| Length of reset impulse | 3 |

¹ After trailing $\overline{\text{WR}}$ edge for first word of fifth instruction.

² After completion of last instruction, a HLT has been fetched.

³ Assumes $\overline{\text{SUS}}=1$; otherwise indefinite.

5.2 TIMING DIAGRAMS

NOTE: All AC timing characteristics shown refer to the synchronous mode of operation except where explicitly stated in the “Test Conditions” column.

| AC CHARACTERISTICS OVER OPERATING RANGE (Clock Timing) | | | | | | |
|--|------------------|----------------|-----|-----|-------|-----------------|
| Signal Number | Symbol | Parameter | Min | Max | Units | Test Conditions |
| 1 | T _{CP} | Clock Period | 50 | 200 | ns | |
| 2 | T _{CH} | Clock High | 20 | | ns | |
| 3 | T _{CL} | Clock Low | 20 | | ns | |
| 4 | T _{CR} | Clock Rise | | 5 | ns | 1.0V to 3.5V |
| 5 | T _{CF} | Clock Fall | | 5 | ns | 3.5V to 1.0V |
| 6 | T _{COD} | ICLK Out Delay | | 35 | ns | |

Note: T1 = T_{CP}

| AC CHARACTERISTICS OVER OPERATING RANGE (Data Fetch—VSP Master Mode) | | | | | | |
|--|-------------------|------------------------------------|-----------------|----------|-------|----------------------------------|
| Signal Number | Symbol | Parameter | Min | Max | Units | Test Conditions |
| 7 | T _{AVD} | Address Valid Delay | | 30 40 | ns | CLK = 20 MHZ CLK = 10, 15 MHZ |
| 9 | T _{CTVD} | Control Valid Delay (WR, RD, DSTB) | | 40 | ns | |
| 10 | T _{DSW} | DSTB Width | T1-15 2T1-10 | | ns | 1 CLK/cycle 2 CLK/cycle |
| 11 | T _{AOS} | Address Out Setup | 10 T1-20 | | ns | 1 CLK/cycle 2 CLK/cycle |
| 12 | T _{AOH} | Address Out Hold | T1-40 T1-20 | | ns | 1 CLK/cycle 2 CLK/cycle |
| 13 | T _{DOS} | Data Out Setup | T1-30 T1/2 | | ns | 1 CLK/cycle 2 CLK/cycle |
| 14 | T _{DOH} | Data Out Hold | (T1/2)-15 | | ns | |
| 15 | T _{DIS} | Data In Setup | 5 | | ns | |
| 16 | T _{DIH} | Data In Hold | 10 | | ns | |

| AC CHARACTERISTICS OVER OPERATING RANGE (Data Fetch—VSP Slave Mode) | | | | | | |
|---|-------------------|--|-----|-----|-------|-----------------|
| Signal Number | Symbol | Parameter | Min | Max | Units | Test Conditions |
| 17 | T _{CSS} | \overline{CS} Setup | 10 | | ns | |
| 18 | T _{CSH} | \overline{CS} Hold | 10 | | ns | |
| 19 | T _{AISW} | Address In Setup to \overline{WR} | 5 | | ns | |
| 20 | T _{AIH} | Address in Hold to \overline{WR} , \overline{RD} | 10 | | ns | |
| 21 | T _{DISW} | Data In Setup to \overline{WR} | 5 | | ns | |
| 22 | T _{DIHW} | Data In Hold to \overline{WR} | 10 | | ns | |
| 23 | T _{WRW} | \overline{WR} Width | 3T1 | | ns | |
| 24 | T _{AISR} | Address In Setup to \overline{RD} | 5 | | ns | |
| 25 | T _{RDW} | \overline{RD} Width | 8T1 | | ns | |
| 26 | T _{DOV} | Data Out Valid | | 8T1 | ns | |
| 27 | T _{DOHR} | Data Out Hold to \overline{RD} | 10 | | ns | |

AC CHARACTERISTICS OVER OPERATING RANGE (Control Setup and Hold Timing)

| Signal Number | Symbol | Parameter | Min | Max | Units | Test Conditions |
|---------------|-------------------|---|-----|-----|-------|-----------------|
| 28 | T _{CTS} | Control Setup (\overline{WR} , \overline{RD} , \overline{BACK}) | 5 | | ns | |
| 29 | T _{CTSS} | Control Setup (\overline{SUS}) | 30 | | ns | |
| 30 | T _{CTH} | Control Hold | 10 | | ns | |
| 31 | T _{BIOV} | \overline{BRQ} , INT Out Valid | | 60 | ns | |
| 32 | T _{BFD} | Bus Float Delay | | 50 | ns | |
| 33 | T _{ASS} | Async Setup | 5 | | ns | Async mode |
| 34 | T _{ASH} | Async Hold | 25 | | ns | Async mode |

AC CHARACTERISTICS OVER OPERATING RANGE (Arbitration Timing)

| Signal Number | Symbol | Parameter | Min | Max | Units | Test Conditions |
|---------------|-------------------|--|------------|--------------------|----------------|---|
| 35 | T _{PF} | \overline{BACK} valid to start Instruction Fetch | 1T1 | | ns | |
| 36 | T _{PD} | \overline{BACK} valid to start Data Fetch | 1T1 | | ns | |
| 37 | T _{PSF} | End Instruction Fetch to \overline{BRQ} Off | 1T1 | | ns | |
| 38 | T _{PSD} | End Data Fetch to \overline{BRQ} Off | 3T1 | | ns | |
| 39 | T _{BADF} | \overline{BACK} preemption to Fetch Off | | 40T1 7T1 9T1 | ns ns ns | Instruction Fetch 1 ICLK Data I/O 2 ICLK Data I/O |
| 40 | T _{BRO} | Fetch Off to \overline{BRQ} Off | 3T1 1T1 | | ns ns | Instruction Fetch Data I/O |
| 41 | T _{BADN} | \overline{BACK} valid to restart Fetch | 9T1 5T1 | | ns ns | Instruction Fetch Data I/O |
| 42 | T _{WR} | Write Recovery | 6T1 | | ns | |
| 43 | T _{RR} | Read Recovery | 4T1 | | ns | |

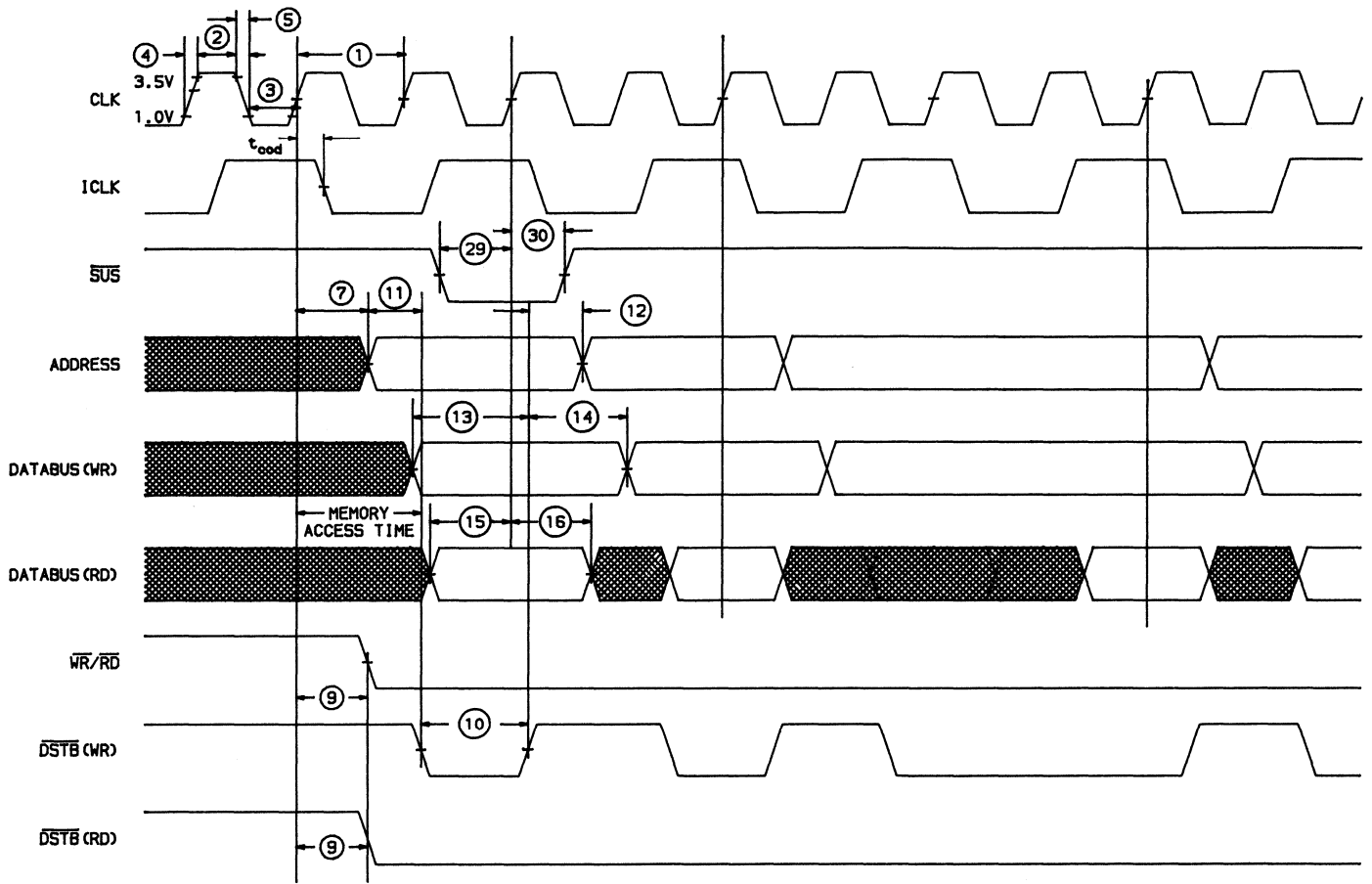


FIGURE 17. ONE-CYCLE ASYNCHRONOUS READ/WRITE WITH/WITHOUT WAIT STATE.

NOTE: \overline{SUS} is sampled at the start of each ICLK cycle. If \overline{SUS} is LOW when sampled, the next cycle will be stretched.

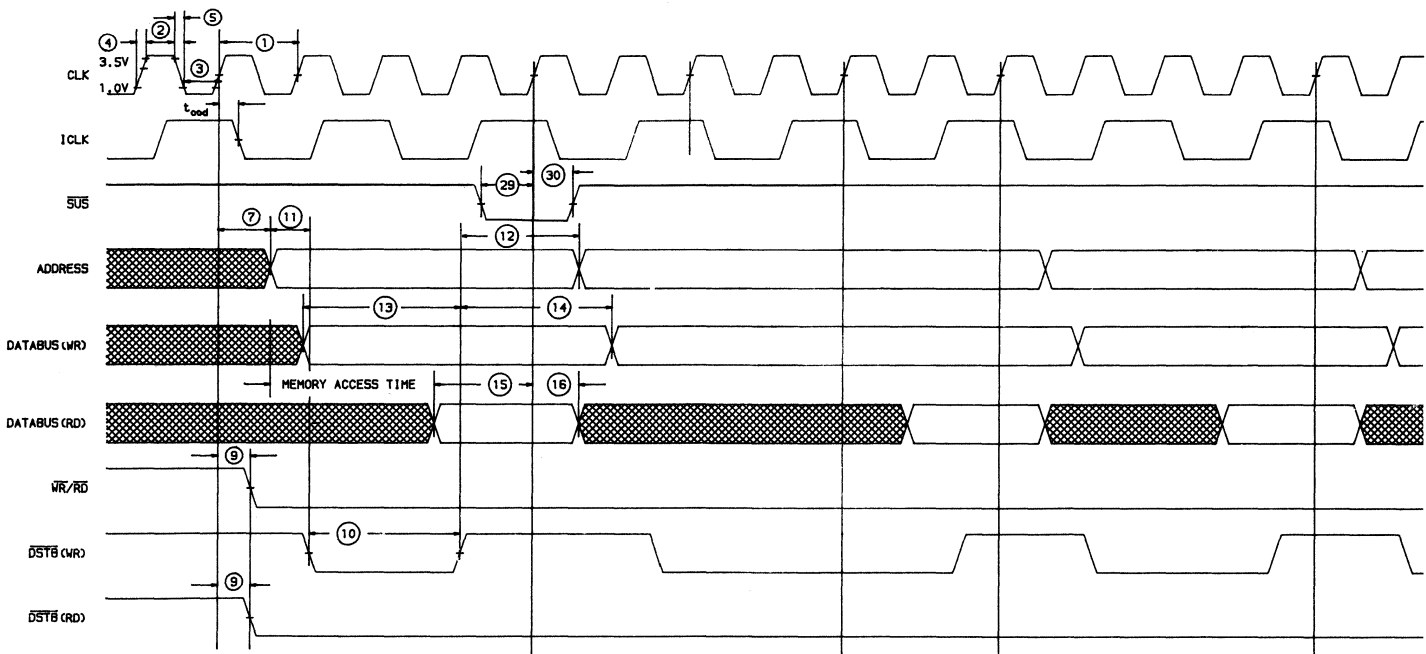


FIGURE 18. TWO-CYCLE ASYNCHRONOUS READ/WRITE WITH/WITHOUT WAIT STATE.

NOTE: \overline{SUS} is sampled at the start of alternate ICLK cycle. If \overline{SUS} is LOW when sampled, the next RD/WR cycle will be stretched.

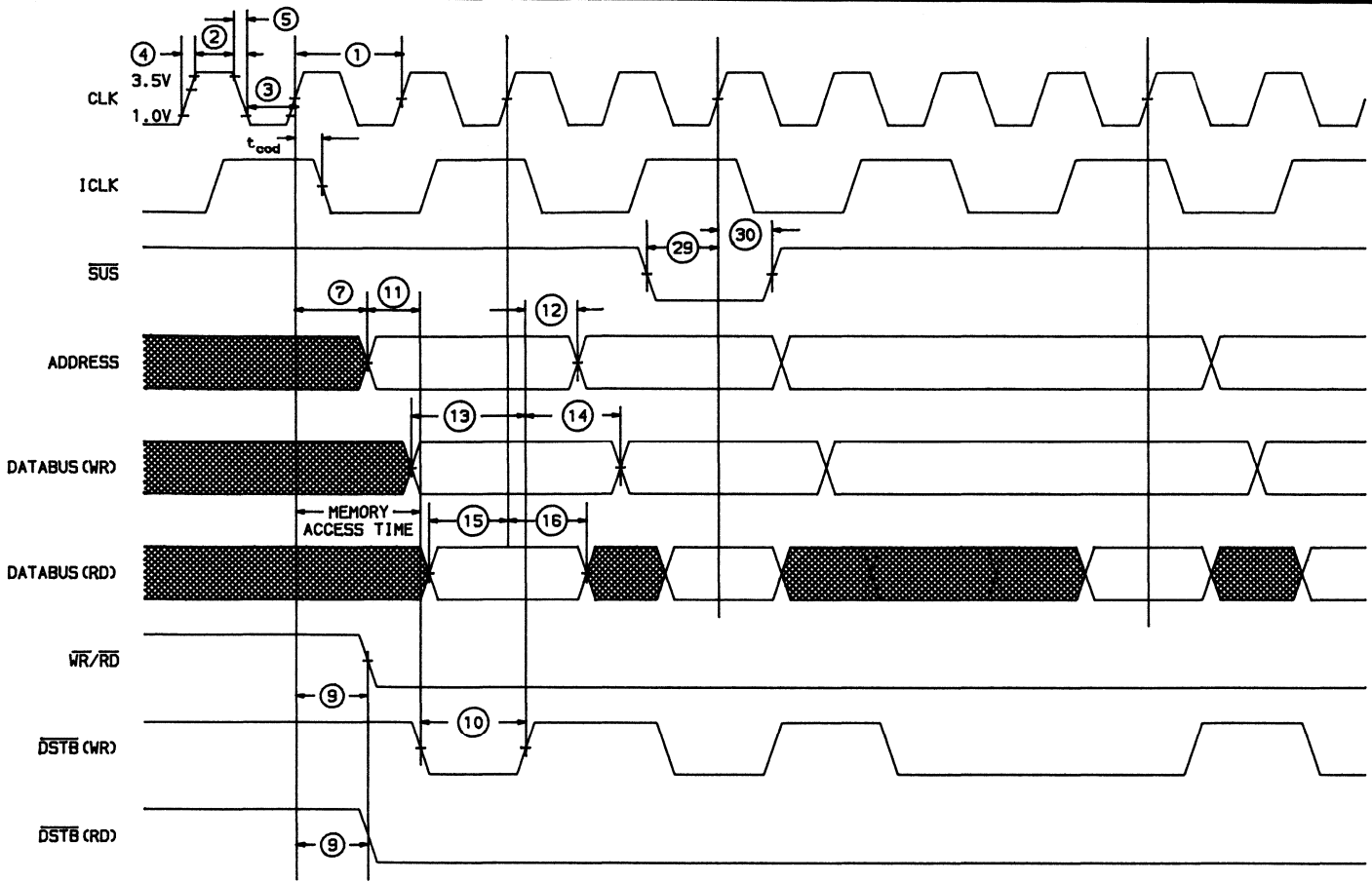


FIGURE 19. ONE-CYCLE SYNCHRONOUS READ/WRITE WITH/WITHOUT WAIT STATE.

NOTE: \overline{SUS} is sampled at the start of each ICLK cycle. If \overline{SUS} is LOW when sampled, the current cycle will be stretched.

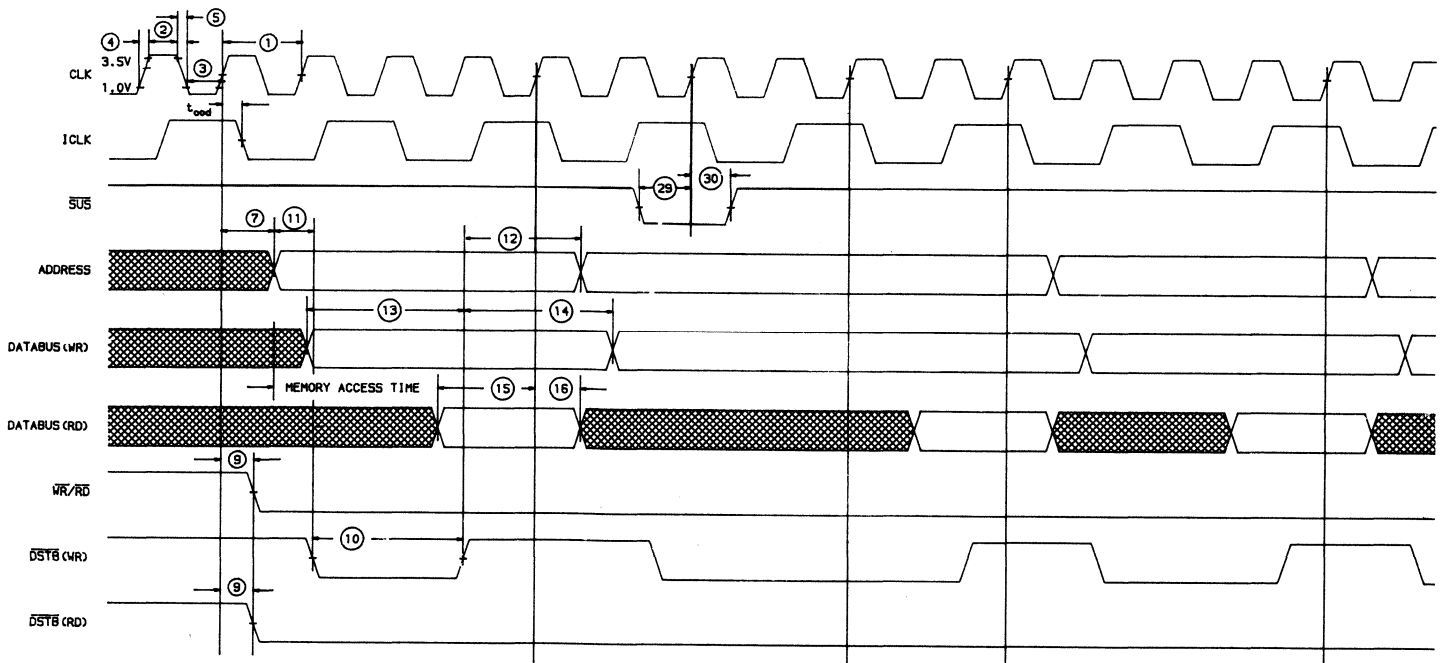


FIGURE 20. TWO-CYCLE SYNCHRONOUS READ/WRITE WITH/WITHOUT WAIT STATE.

NOTE: \overline{SUS} is sampled at the start of alternate ICLK cycle. If \overline{SUS} is LOW when sampled, the current $\overline{RD}/\overline{WR}$ cycle will be stretched.

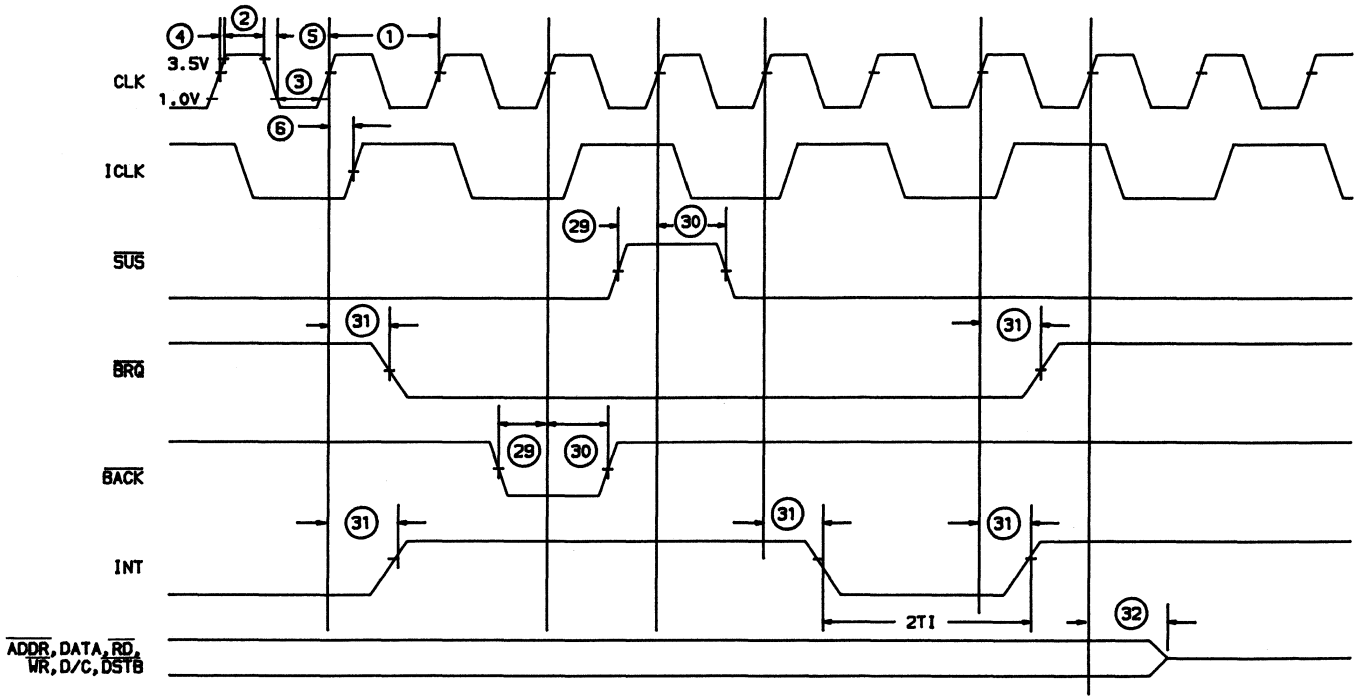


FIGURE 21. CONTROL TIMING.

NOTE: 1) \overline{CS} may be held LOW during successive \overline{RD} operations. 2) \overline{DSTB} is ignored during host \overline{RD} operations

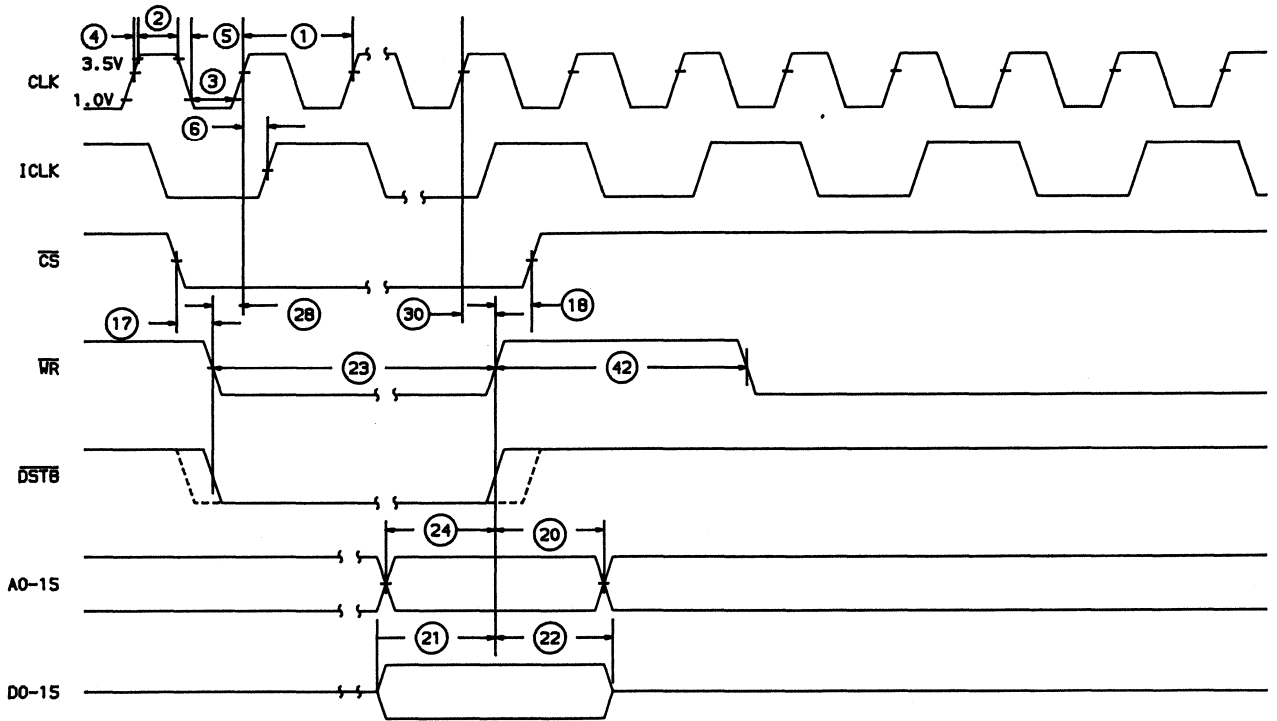


FIGURE 22. VSP BASIC BUS ARBITRATION TIMING.

NOTE: 1) \overline{CS} and \overline{DSTB} may be held LOW during successive \overline{WR} operations. 2) \overline{DSTB} may follow either \overline{WR} or \overline{CS} .

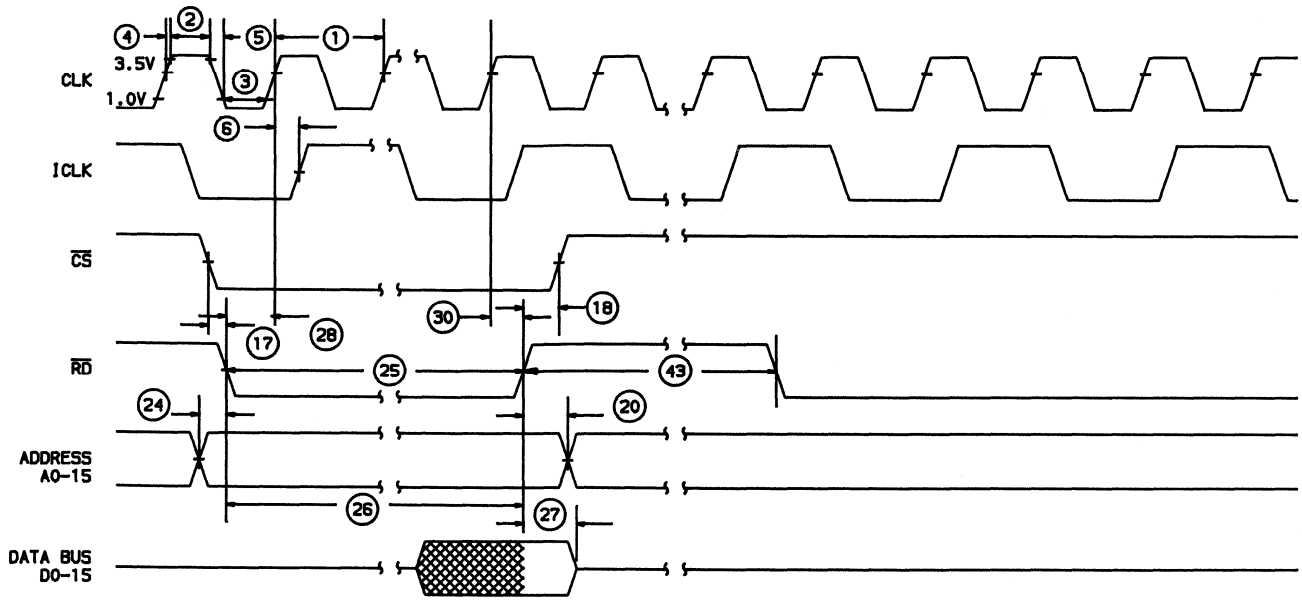


FIGURE 23. EXTERNAL HOST \overline{RD} ACCESS TO VSP RESOURCES.

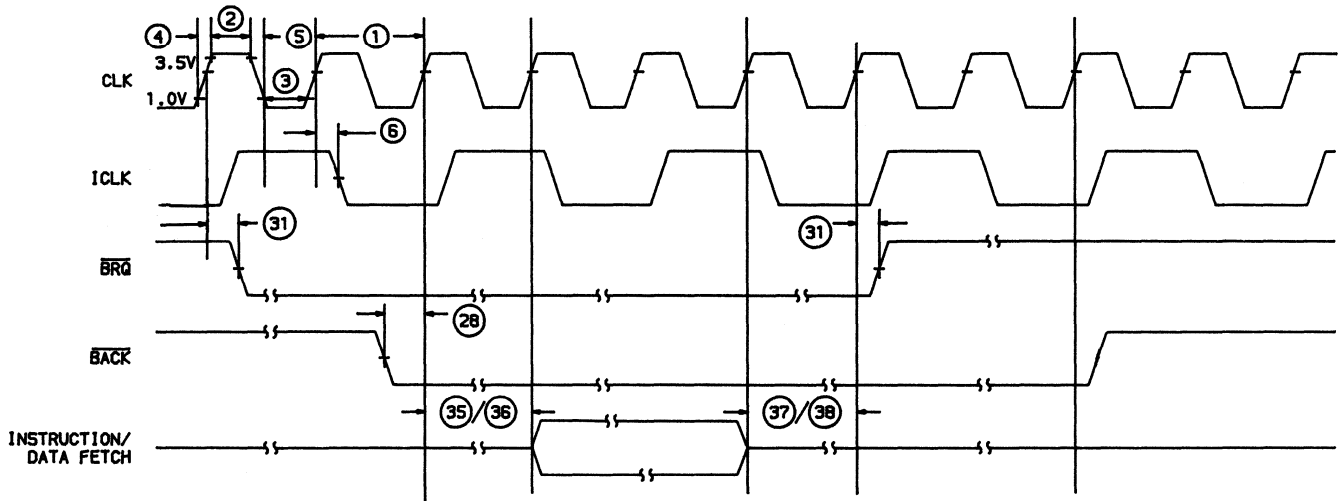


FIGURE 24. EXTERNAL HOST \overline{WR} ACCESS TO VSP RESOURCES.

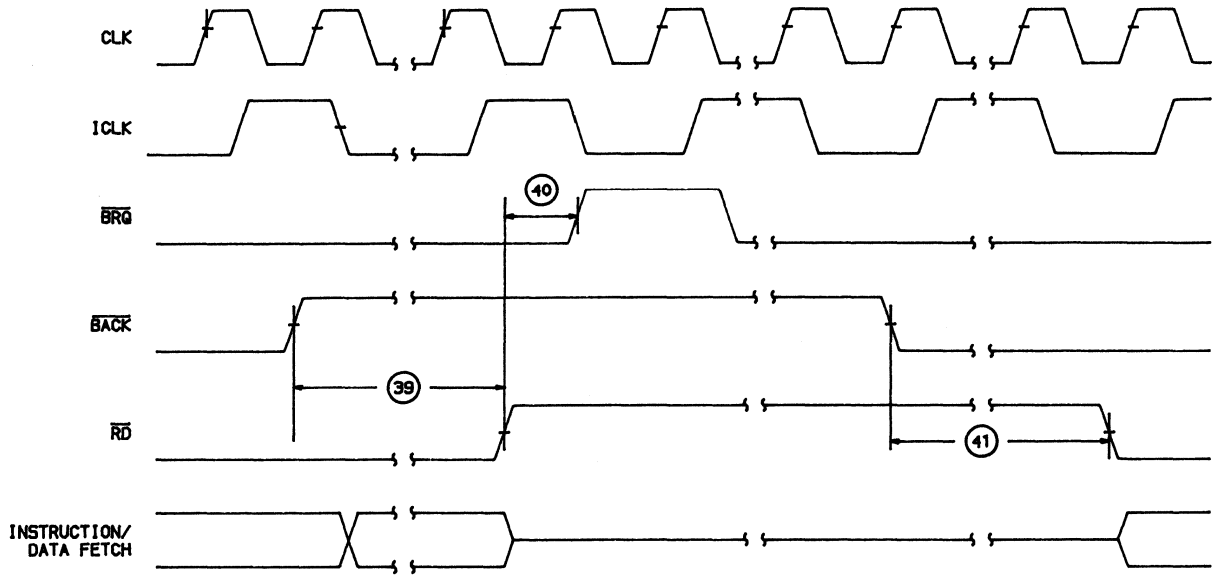


FIGURE 25. PREEMPTIVE BUS ARBITRATION.

A.C. TESTING INPUT, OUTPUT WAVEFORM

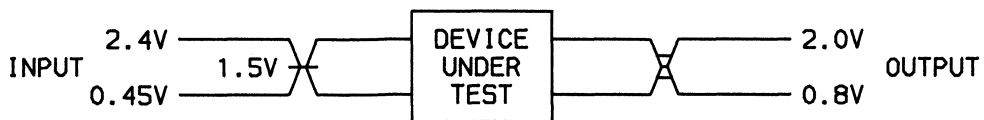


FIGURE 26. A.C. TESTING. INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". FOR BOTH INPUTS AND OUTPUTS, TIMING MEASUREMENTS ARE MADE AT 1.5V FOR BOTH A LOGIC "1" AND "0".

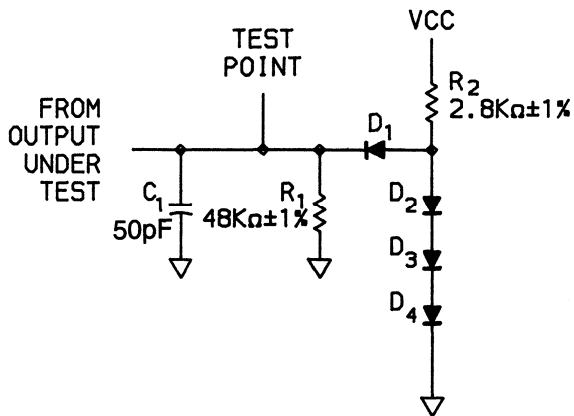


FIGURE 27. NORMAL A.C. TEST LOAD.

| DC CHARACTERISTICS OVER OPERATING RANGE unless otherwise specified | | | | | |
|--|------------------------------------|------|----------------|---------------|-------------------------------------|
| Symbol | Parameter | Min | Max | Units | Test Conditions |
| V_{IL} | Input Low Voltage | -0.5 | 0.8 | V | |
| V_{IH} | Input High Voltage | 2.4 | $V_{CC} + 0.5$ | V | |
| V_{OL} | Output Low Voltage | | 0.45 | V | $I_{OL} = 2 \text{ mA}$ |
| V_{OH} | Output High Voltage | 2.6 | | V | $I_{OH} = -400 \text{ }\mu\text{A}$ |
| I_{CC} | Power Supply Current | | 120 | mA | |
| I_{LI} | Input Leakage Current | | ± 10 | μA | $0 < V_{IN} < V_{CC}$ |
| I_{LO} | Output Leakage Current | | ± 10 | μA | $0.45 < V_{OUT} < V_{CC}$ |
| V_{CL} | Clock in Low Voltage | -0.5 | 0.6 | V | |
| V_{CH} | Clock in High Voltage | 4.2 | $V_{CC} + 0.5$ | V | |
| C_{IN} | Input Capacitance | | 10 | pF | $f_C = 1 \text{ MHz}$ |
| C_{IO} | I/O, Clock, and Output Capacitance | | 10 | pF | $f_C = 1 \text{ MHz}$ |

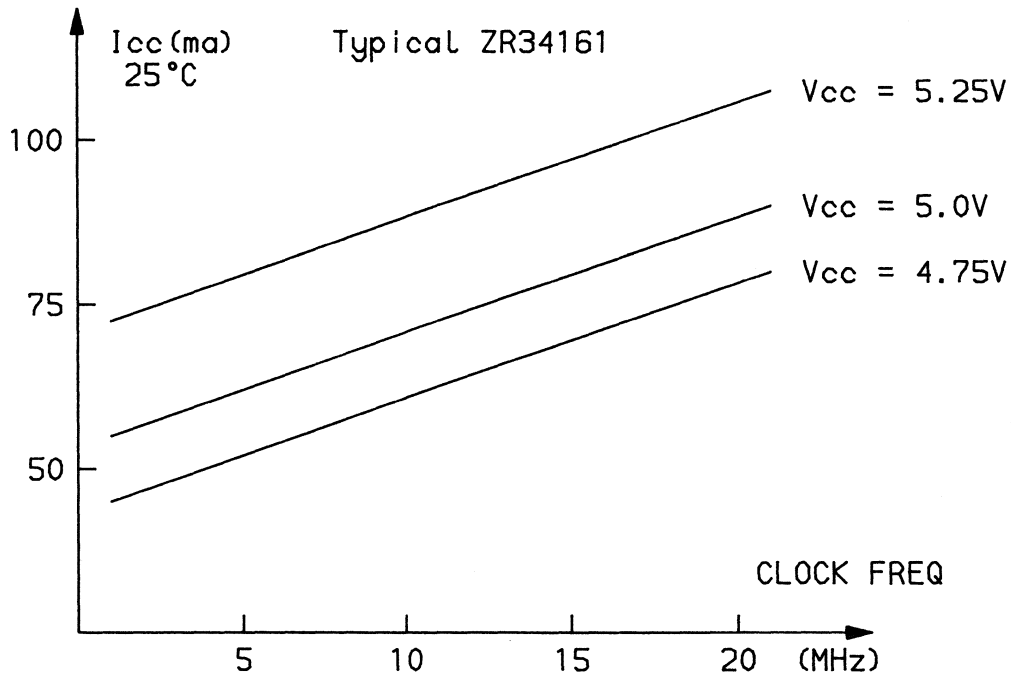


FIGURE 28. ZR34161 TYPICAL I_{CC} REQUIREMENTS.

ABSOLUTE MAXIMUM RATINGS
above which useful life may be impaired

Storage Temperature -65 to +150C
 Supply Voltage to Ground
 Potential (Continuous) -0.5 to +7.0V
 DC Voltage Applied to outputs
 for High Output State -0.5V to +Vcc max
 DC Input Voltage -0.5 to +5.5V
 DC Output Current, into Outputs 20mA
 DC Input Current -30 to +5.0mA

OPERATING RANGE

Commercial (C) Devices (10, 15, 20MHz)
 Temperature $0 \leq T_a \leq +70C$
 Supply Volt $4.75 \leq V_{cc} \leq 5.25V$

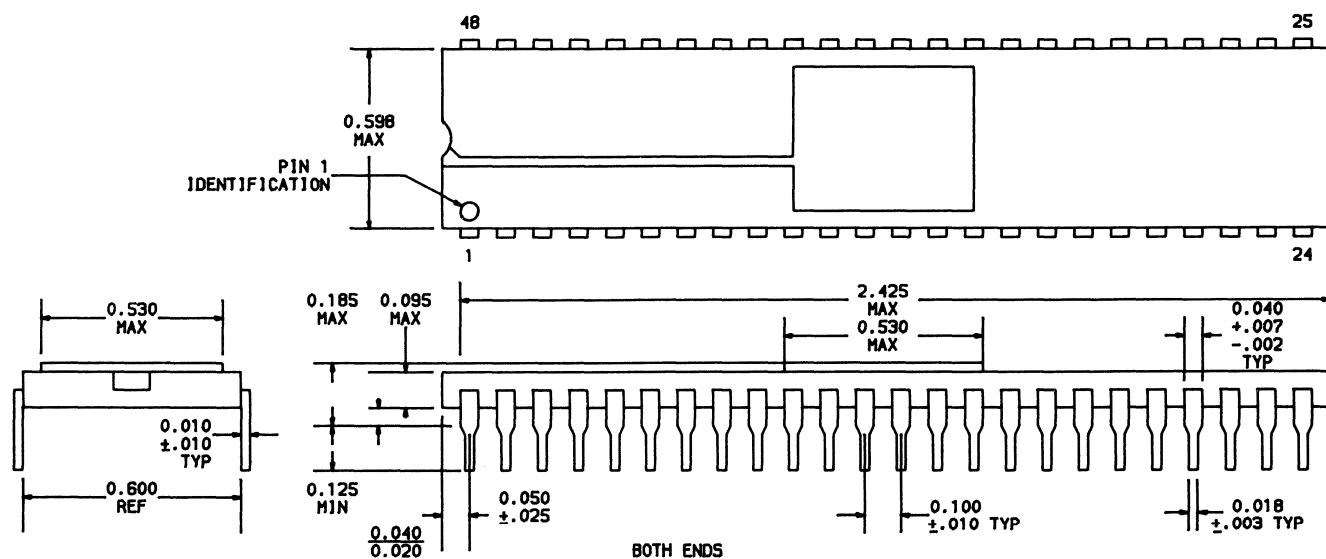
Military (M) Devices (18MHz)
 Temperature $-55 \leq T_a \leq +125C$
 Supply Volt $4.50 \leq V_{cc} \leq 5.50V$

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

ZORAN Corporation cannot assume responsibility for use of any circuitry described other than circuitry embodied in a Zoran product.

PHYSICAL DIMENSIONS

48-pin DIP



ORDERING INFORMATION

| Package Type | Temperature Range | Order Number |
|--------------------|----------------------------|--------------|
| Ceramic 48-pin DIP | $0 \leq T_a \leq +70 C$ | ZR34161DC |
| Ceramic 48-pin DIP | $-55 \leq T_a \leq +125 C$ | ZR34161DM |

DIGITAL FILTER PROCESSORS



PREFACE

This document is intended to provide a thorough overview of the Digital Filter Processor product line. Included in this description are the ZR33481 (8-bit, 4-cell), ZR33881 (8-bit, 8-cell) and ZR33891 (9-bit, 8-cell) DFPs as well as their associated hardware and software tools. For more detailed engineering information on any of the products, please refer to the following, more technical ZORAN literature:

VECTOR SIGNAL PROCESSORS (VSP)

- ZR34161 Data Sheet
- Vector Signal Processor Reference Card
- Vector Signal Processor (VSP) Family Overview
- "VLSI for Digital Image Processing" brochure (also applies to the DFP family)

DIGITAL FILTER PROCESSORS (DFP)

- ZR33481 Data Sheet
- ZR33881 Data Sheet
- ZR33891 Data Sheet
- DFPS Software Technical Description
- DFPB Development Board Technical Description
- "Some Applications of Digital Signal Processing Techniques to Digital Video" Application Note
- "Real-time Two-dimensional Spatial Filtering with the ZORAN Digital Filter Processor Family" Application Note

MARKET APPLICATIONS

- Digital Video and Audio
- Image Compression
- Pattern Recognition
- Radar/Sonar Processing
- Image Enhancement
- Satellite Communications
- Transmultiplexors

FUNCTIONAL APPLICATIONS

- 1-D and 2-D FIR Filtering
- Correlation/Convolution
- Adaptive FIR Filtering
- Matrix Multiplication
- Windowing
- Decimation/Interpolation

PERFORMANCE BENCHMARKS

| <u>Function</u> | <u>Sample Rate</u> | <u>#DFPs</u> | <u>Comments</u> |
|-----------------|--------------------|--------------|---------------------------|
| 8-tap | 20 MHz | 1 | Basic Configuration |
| 16-tap | 20 MHz | 2 | Cascaded |
| 32-tap | 20 MHz | 4 | Cascaded |
| 16-tap | 6.96 MHz | 1 | “Reuse” 8-tap device |
| 16-tap | 20/10 MHz | 1 | Decimate 2:1 |
| 24-tap | 20/6.67 MHz | 1 | Decimate 3:1 |
| 32-tap | 20/5 MHz | 1 | Decimate 4:1 |
| 32-tap | 20/10 MHz | 2 | Decimate 2:1 |
| 16-tap | 20/40 MHz | 2 | Interpolate 1:2 |
| 16-tap | 40 MHz | 4 | True 40 MHz |
| 3 x 3 | 6.67 MHz | 1 | Output every three cycles |
| 3 x 3 | 10 MHz | 2 | Output every two cycles |
| 3 x 3 | 20 MHz | 3 | Output every cycle |
| 3 x 3 x 2 | 5 MHz | 1 | Two kernels, same image |
| 5 x 5 | 6.67 MHz | 2 | Output every three cycles |
| 7 x 7 | 5 MHz | 2 | Output every four cycles |
| 7 x 7 | 10 MHz | 4 | Output every two cycles |

DFP FAMILY MEMBERS

| <u>Processor</u> | <u># Filter Cells</u> | <u>Word Length</u> | <u>Sample Rates</u> |
|------------------|-----------------------|--------------------|---------------------|
| ZR33481 | 4 | 8 bits | 10, 15, 20 MHz |
| ZR33881 | 8 | 8 bits | 10, 15 MHz |
| ZR33891 | 8 | 9 bits | 10, 15, 20 MHz |

GENERAL FEATURES

- Four or eight filter cells
- 20 MHz sample rate
- 8-bit or 9-bit coefficient and data
- Unsigned and two's complement arithmetic
- 8 x 8 or 9 x 9-bit parallel multiplier per cell
- 25 or 26-bit accumulator per cell
- Filter lengths in excess of 1,000 taps without overflow
- Shift and add output stage for combining filter outputs
- Expandable coefficient/data size and filter length
- Decimation by 2, 3 or 4
- Low power CMOS

DESCRIPTION

The ZORAN Digital Filter Processors (DFP) are video speed devices optimized to compute sum-of-products operations for real-time applications. The DFPs achieve high performance for implementing 1-D and 2-D finite impulse response (FIR) digital filters, multibit correlators, adaptive filters and other high speed signal processing operations. The DFP family obtains building block performance by incorporating multiple filter cells in a true parallel processing configuration on a single device. Both four and eight cell processors are available. Each filter cell contains a parallel multiplier, accumulator, and three decimation registers. Additional resources are included to implement a complete FIR filter function. The DFPs support sampling rates up to 20 MHz with a single IC. Higher sampling rates are attained with the use of multiple devices. A single eight-cell processor has a computation bandwidth of 340 million operations/second.

The DFPs support both unsigned and two's complement arithmetic which are independently selectable for coefficients and data.

A typical circuit consists of one or more DFPs controlled by an external sequencer. The sequencer controls the inputting of data and coefficients to the DFP and selects the proper cell output and clearing functions.

ZORAN's focus on providing the designer with a complete system solution is reflected in the available hardware and software tools. The Digital Filter Processor Software (DFPS) combines a comprehensive filter design package with signal generation and frequency response analysis. In addition, filter coefficients can be quantized and formatted for a PROM programmer. The Digital Filter Processor Board (DFPB) is a complete PC-based hardware system which includes four DFPs, signal and coefficient memory and 20 MHz I/O ports.

WHY THE DFP IS UNIQUE

The Digital Filter Processor family achieves high performance through its parallel processor architecture. The DFP integrates several high speed multiplier/accumulator cells on a single device which is optimized to perform sum-of-products operations. This integration provides the designer with a powerful alternative to building block architectures. Building block systems typically consist of many discrete multipliers, accumulators, and shift registers which are organized in a direct form FIR filter configuration (Figure 1). The DFP implements these same operations on a single device (Figure 2).

Especially unique to the Digital Filter Processor family are the decimation registers. Each cell contains three decimation registers which support downsampling (reducing the sampling rate by 2, 3 or 4). These decimation registers also provide the necessary delay for the implementation of two dimensional filters.

In addition to its high performance, the DFP family exhibits flexibility and modularity by allowing the system designer to trade off speed, precision, and filter order. For example, multiple DFPs can be cascaded to increase the number of filter taps while maintaining maximum throughput. Similarly, wordlength can be expanded with additional DFPs. For applications with lower sampling rates, a single DFP can be used to add taps or increase precision.

Each cell of the DFP is responsible for an independent sum-of-products. As new input data samples enter the DFP, they are simultaneously distributed to all cells (Figure 2). Each new input is then multiplied by the cell's current coefficient, and added to its respective accumulator. By maintaining independent sums within cells, a new output value can be available on each cycle. Coefficients are then transferred to the input of the adjacent cell.

The DFP's high level of integration enhances the system's overall speed and reliability while decreasing board space, power and external interconnects. The DFP family combines raw processing speed with a high level of flexibility to address many digital signal processing needs.

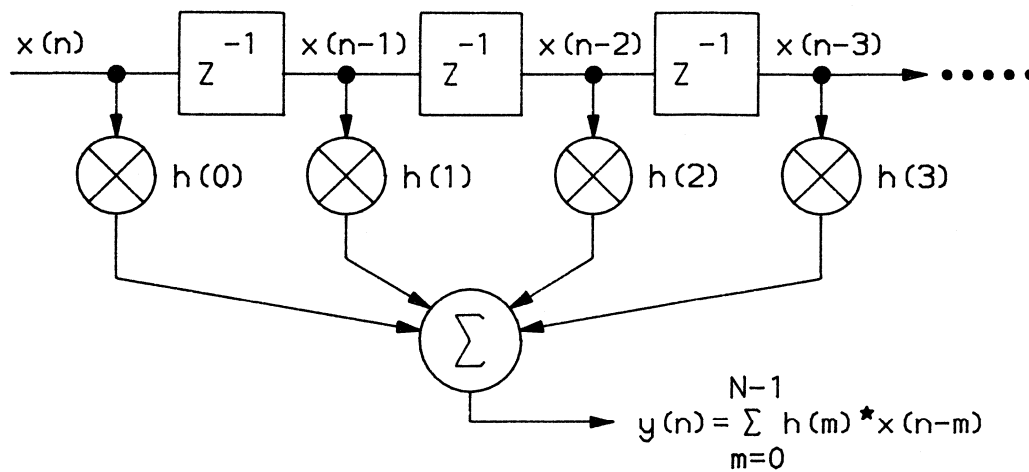


FIGURE 1. DIRECT FORM REPRESENTATION OF FIR FILTER.

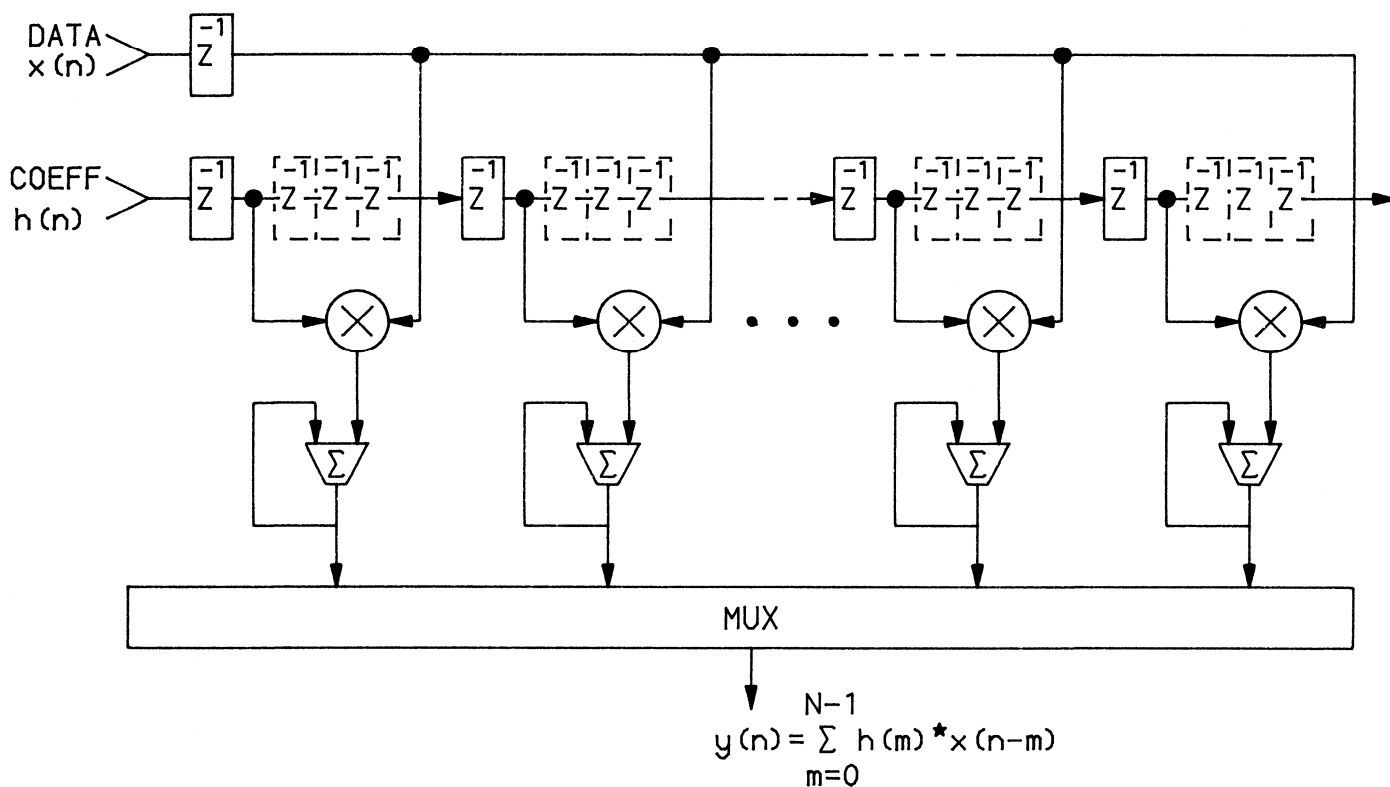


FIGURE 2. ZORAN DFP FAMILY BLOCK DIAGRAM.

DFP CONFIGURATIONS

The following section describes various DFP configurations which build upon the basic concepts of the DFP architecture.

BASIC 1-D FINITE IMPULSE RESPONSE (FIR) FILTERING

The finite impulse response filter (FIR) is simply a finite-length sum-of-products digital filter (Figure 1). Each output sample is a weighted sum of the new input value and the N-1 previous inputs, where N is the order of the filter. The weighted coefficient values are selected to produce the frequency response of a particular digital filter. These coefficients can be directly calculated with the aid of the ZORAN Digital Filter Processor Software (DFPS).

In its basic form, the ZR33891 DFP implements a 9 x 9-bit, 8-tap FIR at 20 MHz. New data samples are simultaneously input to all eight filter cells. Each cell accumulates the sum-of-products for one output point which is staggered in time so that a new output is available every cycle. Coefficients enter the first cell and are transferred to adjacent cells on subsequent cycles. An external sequencer controls the DFP and addresses filter coefficients (Figure 3).

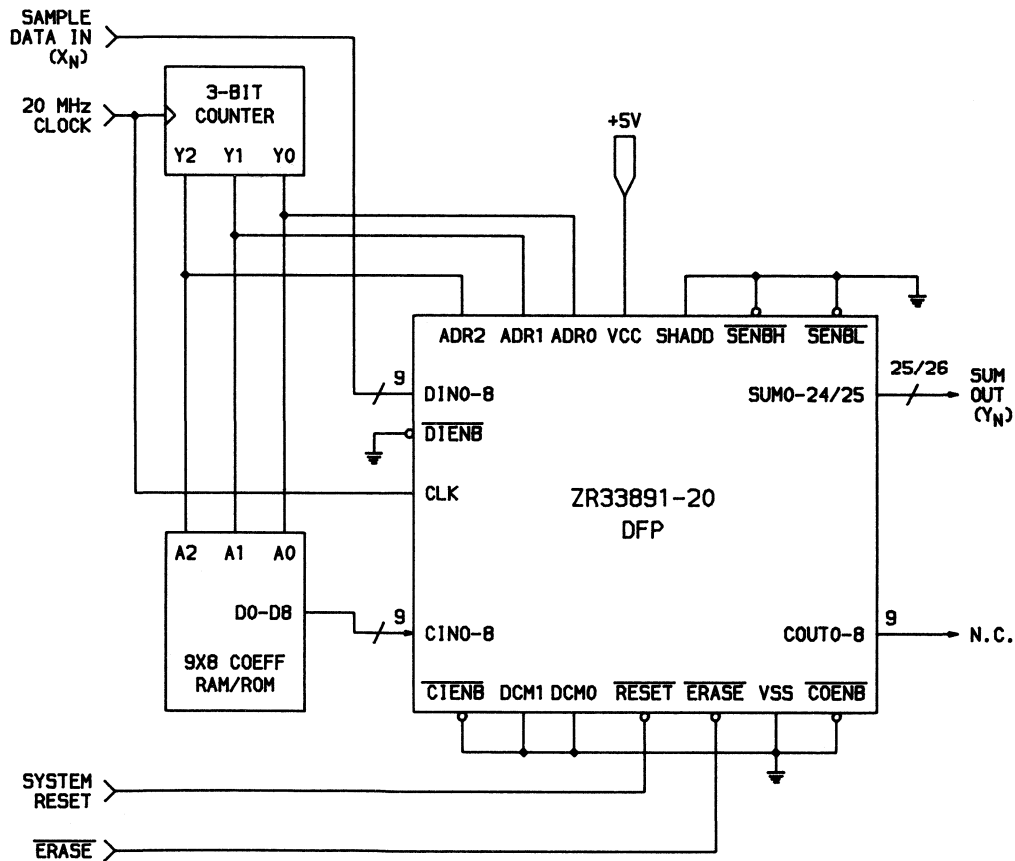


FIGURE 3. SCHEMATIC OF A BASIC 8-TAP DFP CONFIGURATION.

HIGHER ORDER 1-D FILTERS

Several applications require digital filters with sharper frequency response characteristics such as faster roll-off or higher rejection in the stop band. This can be achieved by increasing the number of filter taps. Filter lengths greater than eight taps can be implemented by cascading multiple DFPs (Figure 4) or by recirculating input data into the same DFP for less time critical applications (Figure 5). For full speed 20 MHz FIR filters the multiple DFP configuration is utilized with coefficients being passed between DFPs. In practice, filters in excess of 1,000 taps can be implemented by cascading DFPs without the risk of overflow. High speed communications receivers are an example of a multiple DFP application. These receivers require sampling rates between 10 and 20 MHz with filter orders in excess of 32 taps.

DECIMATION

Many digital filtering applications require a decrease in the effective sampling rate of the system. Decimation allows an integer decrease in the sampling rate by low pass filtering the signal and keeping every Nth sample. Often only the low frequency portion of the spectrum contains the desired signal. Therefore, low pass filtering does not result in loss of information. Furthermore, since decimation discards many samples, the effective number of taps in the filter is increased. For example, an 8-cell DFP which normally implements an 8-tap FIR at 20 MHz can perform a 16-tap FIR at 10 MHz with a decimation factor of 2 to 1.

By using the decimation registers in each cell, the DFP can reduce sampling rates by a factor of 2, 3 or 4 and still maintain full efficiency of the multiplier/accumulators. This is accomplished by processing only those samples which are to be saved.

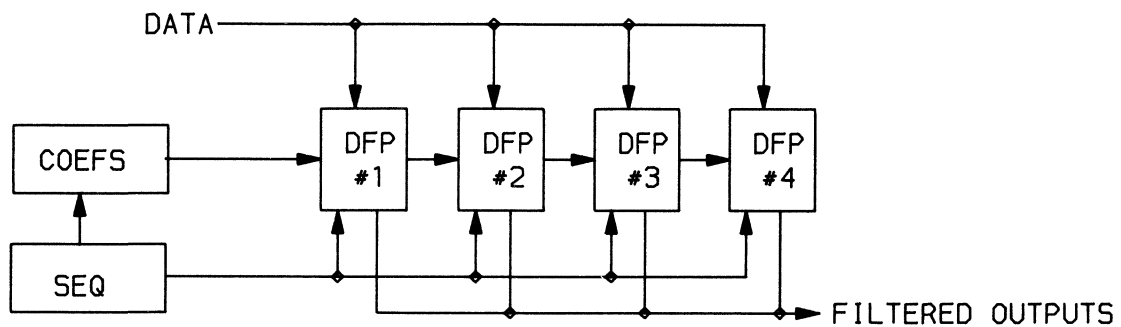


FIGURE 4. HIGHER ORDER 1-D FILTER WITH MULTIPLE DFPs.

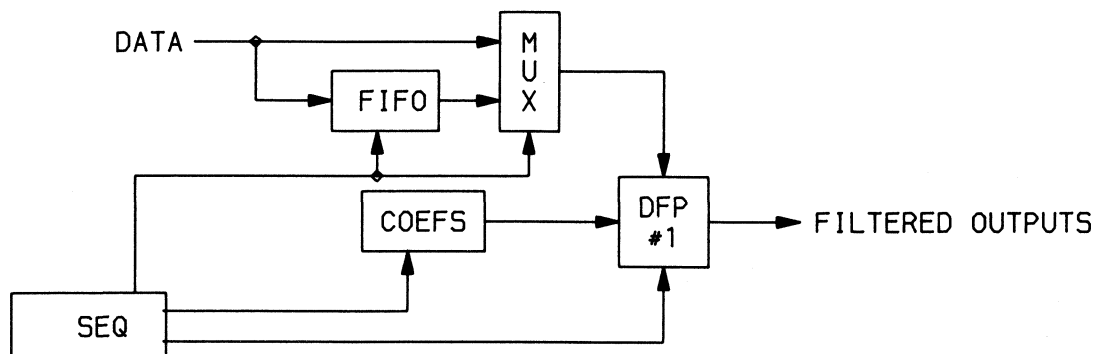


FIGURE 5. HIGHER ORDER 1-D FILTER WITH SAME DFP.

For example, a single 15 MHz, 8-cell DFP device can decode a 14.3 MHz composite signal into one of its chrominance components by employing a 32-tap FIR filter at a decimation of 4 to 1 (Figure 6). The signal must first be band pass filtered, then demodulated and low pass filtered. The mathematical properties of digital filtering allow these three operations to be combined into one 32-tap FIR filter.

INTERPOLATION

Similar to the rationale for decimation, interpolation increases the effective sampling rate of the system. The sample rate can be increased by converting back to the analog domain and resampling. However, this approach introduces noise and distortion and requires the use of expensive high speed A/D converters. Instead, interpolation can be employed. Interpolation inserts N-1 zeros between each sample and low pass filters the output, thus increasing the sample rate.

By separating the even and odd coefficients of the filter and taking advantage of the fact that the inserted zeros do not contribute to the sums, an 8-tap, 20 to 40 MHz interpolator can be implemented using two 4-cell DFPs (Figure 7). A 40 MHz multiplexor is utilized to recombine the outputs of the two DFPs.

A common application utilizing interpolation is the conversion of video standards. For example, NTSC to PAL (phase alteration by line) requires a change in sampling rate from 14.3 to 17.8 MHz. The systems designer can convert between these sampling rates by using a combination of interpolation and decimation techniques. Another example of interpolation is in time division multiplexing (TDM) to frequency division multiplexing (FDM) translation. The TDM signals are interpolated to the higher FDM sampling rate, where they are then modulated and summed.

HIGHER SAMPLE RATE FILTERS

Applications with sampling rates in excess of 20 MHz require multiple DFPs in a configuration which processes intermediate results in parallel. These results are then summed by external adders and output multiplexed at the higher rate. Note that this procedure is significantly different than interpolation. Interpolation is used to increase the effective sampling rate of a signal. Higher sample rate filters actually perform filtering operations on higher frequency signal.

Some typical applications of high sample rate filtering are found in radar demodulators and satellite communications receivers which must operate in real-time. Incoming data and coefficients are separated into even and odd paths and processed by different DFPs. For example, a 16-tap, 40 MHz filter can be designed with four DFPs (Figure 8).

HIGHER PRECISION 1-D FILTERS

Several digital signal processing applications require longer wordlength calculations to maintain precision. Data samples and coefficients can be extended by utilizing several DFPs in parallel to achieve maximum speed, or a single DFP at a reduced sample rate. Each DFP operation computes a partial product which is shifted and added with other partial products to produce the desired result. The shift and add operation can be performed within the DFP, or with external adders for higher sampling rates.

Wordlength can be increased by computing the partial products of the larger multiplication. For example, a 16 x 8-bit multiply can be calculated by computing the most significant 8 x 8 product, shifting and then adding to the least significant 8 x 8 product.

A 16 x 24-bit sum-of-products can be implemented using a single ZR33481 4-cell DFP at 5 MHz. Six 8 x 8-bit partial products are internally shifted and added to produce a 40-bit sum every four cycles (200 ns).

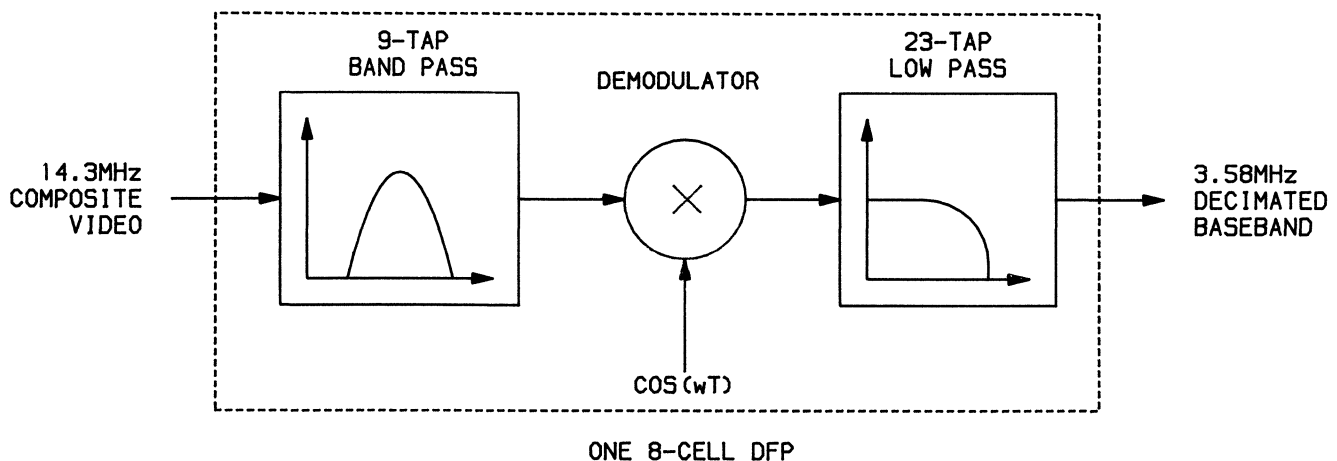


FIGURE 6. DIGITAL VIDEO COMPOSITE TO COMPONENT DECODER.

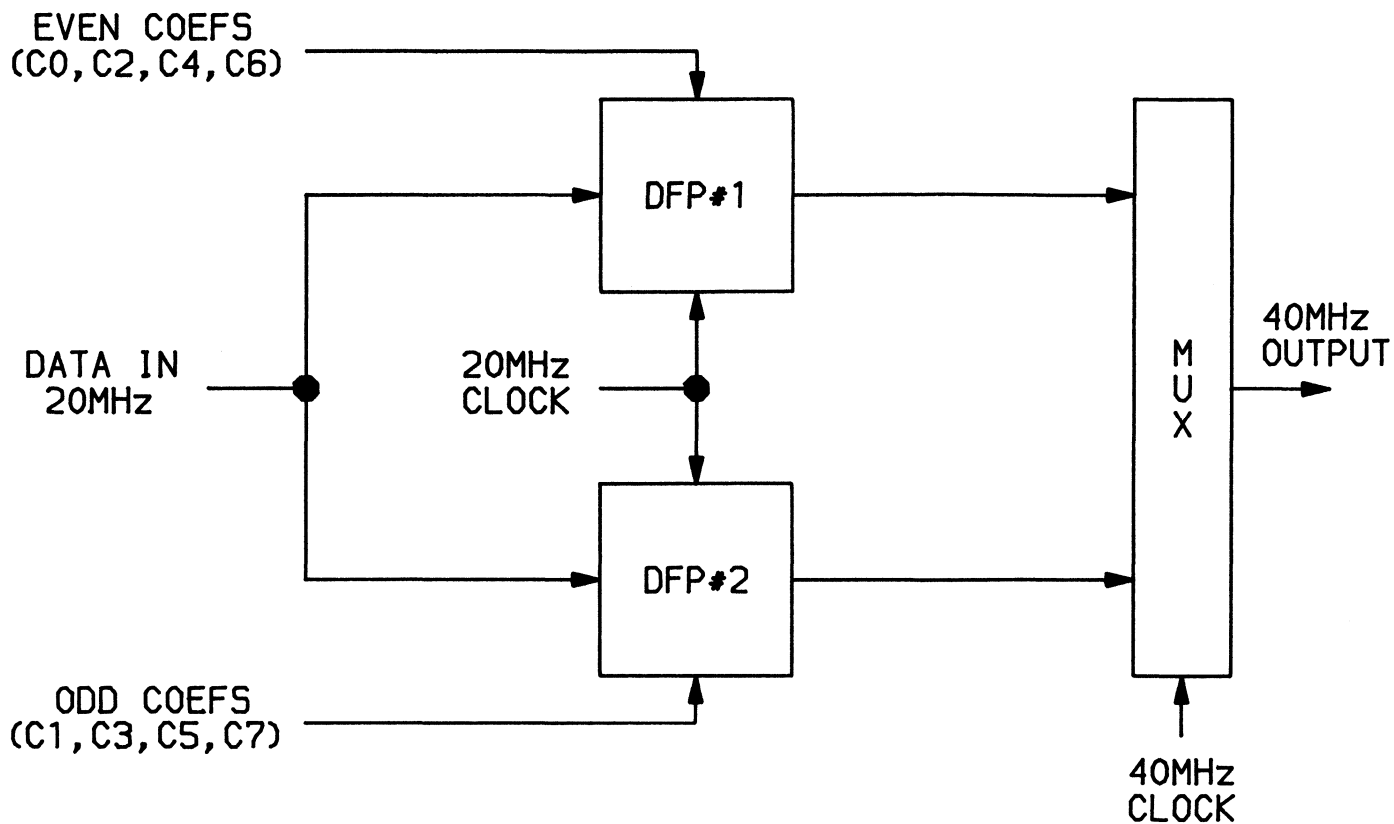


FIGURE 7. 20 MHz TO 40 MHz INTERPOLATOR USING TWO DFPs.

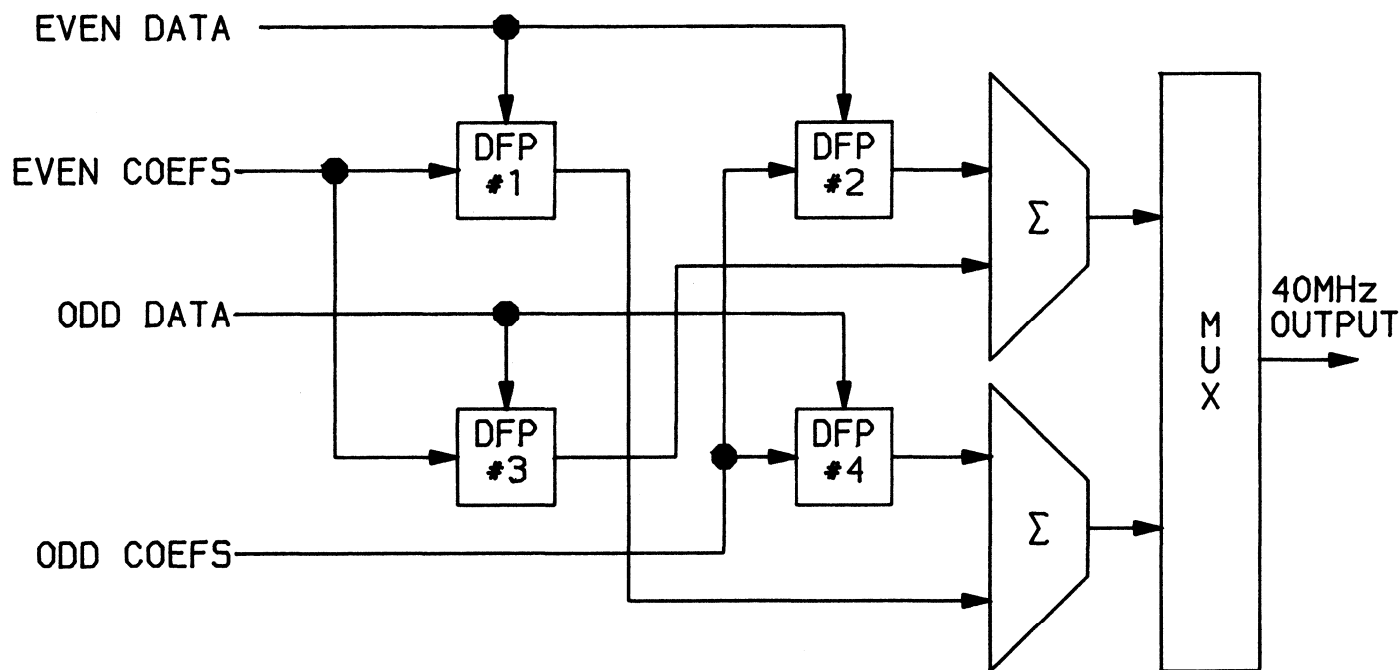


FIGURE 8. 40 MHz, 16-TAP FILTER WITH FOUR DFPs.

TWO-DIMENSIONAL FIR FILTERS

Two-dimensional signals, such as images, use many of the same concepts found in one-dimensional signal processing algorithms. Instead of performing a sum-of-products calculation on a linear stream of input data, image processing operates on a small 2-D window or kernel of the image. The coefficients of this kernel are determined for a particular filtering operation such as edge detection or smoothing. For example, a 3 x 3 kernel calculates one output pixel by performing a nine-term sum-of-products in a single cell. By moving this kernel over the entire image, the desired operation can be performed on the 2-D image. 2-D filtering makes extensive use of the DFP's internal decimation registers. Specifically, the decimation registers provide a delay mechanism for organizing the kernel's coefficients into rows and columns. Each cell receives coefficients and data in column scanned order. After a cell has completed a column, the coefficients enter the next adjacent cell.

In the example of a 3 x 3 kernel with one DFP, three decimation registers are employed. This creates a three cycle delay so that the correct coefficients and image data are multiplied in subsequent cells. A 3 x 3 high pass filter can be executed on a 256 x 256 x 8-bit image in 9.8 ms with a single 4-cell DFP (Figure 9). A three-line row buffer is used to enter data in column scanned order. Each 3 x 3 sum-of-products is accumulated in a single cell with a new output value every three cycles. By combining three DFPs, one new sum can be available every cycle.

Note that in this example the fourth cell is unused. A 4 x 4 kernel can process this image in 13.1 ms by using the same 4-cell DFP. In this case a new value would be output every four cycles. The DFP architecture easily adapts to a variety of larger kernel sizes such as 7 x 7, for use in real-time systems. This topic is covered in greater detail in the application note "Real-time 2-D Spatial Filtering with the ZORAN Digital Filter Processor Family."

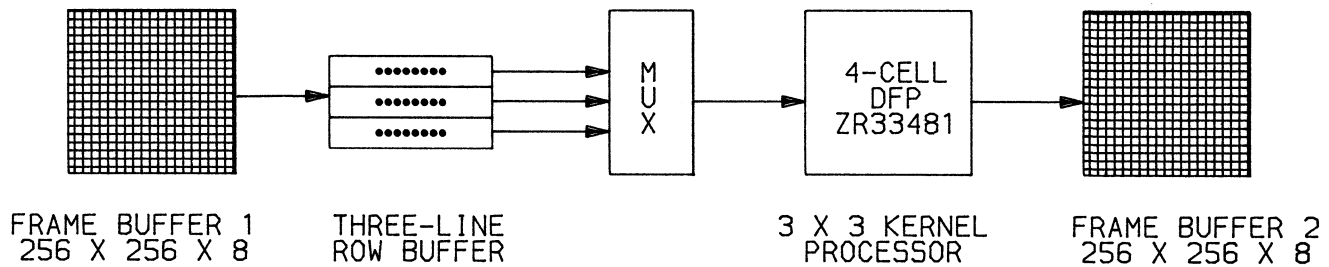


FIGURE 9. 3 x 3 KERNEL IMAGE PROCESSOR FOR REAL TIME EDGE DETECTION ON A 256 x 256 x 8 IMAGE.

DFP CONFIGURATION SELECTION GUIDE

The following DFP Configuration Selection Guide is meant to serve as a reference for the designer who is evaluating DFP benchmarks. The concepts of cascade processors, decimation, increased wordlength, etc. are shown individually for clarity. By combining these techniques, the designer can often achieve higher performance suitable for a particular application. The DFP Configuration Selection Guide is divided into 1-D and 2-D sections. Note: The ZR33891 9 x 9-bit, 8-cell DFP can be substituted for the ZR33881 8 x 8-bit, 8-cell DFP for increased precision.

| <u>Application</u> | <u>Solution</u> | <u>Comment</u> |
|----------------------------------|-----------------|----------------------------------|
| 1-D Filtering | | |
| 8-tap, 8 x 8-bit, 20 MHz | 1 x ZR33881-20 | Basic 8 x 8 mult, 20 MHz, 8-cell |
| 8-tap, 9 x 9-bit, 20 MHz | 1 x ZR33891-20 | Basic 9 x 9 mult, 20 MHz, 8-cell |
| 8-tap, 9 x 9-bit, 10 MHz | 1 x ZR33891-10 | Basic 9 x 9 mult, 10 MHz, 8-cell |
| 8-tap, 8 x 8-bit, 5.45 MHz | 1 x ZR33481-20 | 4-cell DFP, 3 outputs/11 cycles |
| 8-tap, 8 x 8-bit, 4.09 MHz | 1 x ZR33481-15 | 4-cell DFP, 3 outputs/11 cycles |
| 16-tap, 8 x 8-bit, 20 MHz | 2 x ZR33881-20 | Two cascaded 8-cell DFPs |
| 16-tap, 8 x 8-bit, 6.96 MHz | 1 x ZR33881-20 | 8 outputs/23 cycles |
| 24-tap, 8 x 8-bit, 5.16 MHz | 1 x ZR33881-20 | 8 outputs/31 cycles |
| 32-tap, 8 x 8-bit, 4.10 MHz | 1 x ZR33881-20 | 8 outputs/39 cycles |
| 32-tap, 8 x 8-bit, 20 MHz | 4 x ZR33881-20 | Four cascaded 8-cell DFPs |
| 32-tap, 8 x 8-bit, 15 MHz | 4 x ZR33881-15 | Use 15 MHz DFPs |
| 32-tap, 8 x 8-bit, 10 MHz | 4 x ZR33881-10 | Use 10 MHz DFPs |
| 32-tap, 8 x 8-bit, 6.81 MHz | 2 x ZR33881-20 | 16 outputs/47 cycles |
| 32-tap, 8 x 8-bit, 5.11 MHz | 2 x ZR33881-15 | 16 outputs/47 cycles |
| 64-tap, 8 x 8-bit, 20 MHz | 8 x ZR33881-20 | Eight cascaded 8-cell DFPs |
| 64-tap, 8 x 8-bit, 6.74 MHz | 4 x ZR33881-20 | 32 outputs/95 cycles |
| 64-tap, 8 x 8-bit, 4.81 MHz | 2 x ZR33881-20 | 16 outputs/79 cycles |
| 16-tap, 8 x 8-bit, 20/10 MHz | 1 x ZR33881-20 | Decimate 2:1 from 20 to 10 MHz |
| 24-tap, 8 x 8-bit, 20/6.67 MHz | 1 x ZR33881-20 | Decimate 3:1 |
| 32-tap, 8 x 8-bit, 20/5 MHz | 1 x ZR33881-20 | Decimate 4:1 |
| 32-tap, 8 x 8-bit, 6.80/3.40 MHz | 1 x ZR33881-20 | Decimate 2:1, 8 output/47 cycles |
| 32-tap, 8 x 8-bit, 20/10 MHz | 2 x ZR33881-20 | Decimate 2:1 |
| 48-tap, 8 x 8-bit, 20/6.67 MHz | 2 x ZR33881-20 | Decimate 3:1 |
| 64-tap, 8 x 8-bit, 20/5 MHz | 2 x ZR33881-20 | Decimate 4:1 |
| 8-tap, 8 x 8-bit, 20/40 MHz | 2 x ZR33481-20 | Interpolate 1:2, 4-cell DFP |
| 16-tap, 8 x 8-bit, 20/40 MHz | 2 x ZR33881-20 | Interpolate 1:2 |
| 32-tap, 8 x 8-bit, 20/40 MHz | 4 x ZR33881-20 | Interpolate 1:4 |
| 8-tap, 8 x 8-bit, 40 MHz | 4 x ZR33481-20 | 4-cell DFP, 4 terms, 3 adders |
| 16-tap, 8 x 8-bit, 40 MHz | 4 x ZR33881-20 | 4 terms, uses 3 adders |
| 24-tap, 8 x 8-bit, 40 MHz | 9 x ZR33881-20 | 9 terms, uses 8 adders |

DFP CONFIGURATION SELECTION GUIDE (cont.)

| <u>Application</u> | <u>Solution</u> | <u>Comment</u> |
|----------------------------------|-----------------|--------------------------------|
| 2-D Spatial Filtering | | |
| 3 x 3 on 256 x 256 x 8 image | 1 x ZR33481-20 | 9.8 ms, real-time |
| 3 x 3 on 256 x 256 x 8 image | 1 x ZR33481-15 | 13.1 ms, real-time |
| 3 x 3 on 256 x 256 x 8 image | 1 x ZR33481-10 | 19.6 ms, real-time |
| 3 x 3 on 512 x 512 x 8 image | 1 x ZR33481-20 | 39.3 ms |
| 3 x 3 on 512 x 512 x 8 image | 2 x ZR33481-20 | 26.2 ms, 1 adder, real time |
| 3 x 3 on 512 x 512 x 8 image | 3 x ZR33481-20 | 13.1 ms, 2 adders, real-time |
| 3 x 3 on 512 x 512 x 8 image | 2 x ZR33481-15 | 35.0 ms, 1 adder |
| 3 x 3 on 512 x 512 x 8 image | 3 x ZR33481-10 | 26.2 ms, 2 adders, real-time |
| 3 x 3 x 2 on 256 x 256 x 8 image | 1 x ZR33881-20 | 13.1 ms, 2 kernels, same image |
| 3 x 3 x 2 on 512 x 512 x 8 image | 1 x ZR33881-20 | 52.4 ms, 2 kernels, same image |
| 4 x 4 on 256 x 256 x 8 image | 1 x ZR33481-20 | 13.1 ms, real-time |
| 4 x 4 on 256 x 256 x 8 image | 1 x ZR33481-15 | 17.5 ms, real-time |
| 4 x 4 on 256 x 256 x 8 image | 1 x ZR33481-10 | 26.2 ms, real-time |
| 4 x 4 on 512 x 512 x 8 image | 1 x ZR33481-20 | 52.4 ms |
| 4 x 4 on 512 x 512 x 8 image | 2 x ZR33481-20 | 26.2 ms, 1 adder, real-time |
| 4 x 4 on 512 x 512 x 8 image | 4 x ZR33481-20 | 13.1 ms, 3 adders, real-time |
| 5 x 5 on 256 x 256 x 8 image | 2 x ZR33481-20 | 19.7 ms, 1 adder, real-time |
| 5 x 5 on 256 x 256 x 8 image | 2 x ZR33481-15 | 26.2 ms, 1 adder, real-time |
| 5 x 5 on 256 x 256 x 8 image | 2 x ZR33481-10 | 39.3 ms, 1 adder, real-time |
| 5 x 5 on 256 x 256 x 8 image | 2 x ZR33881-20 | 9.8 ms, 1 adder, real-time |
| 5 x 5 on 256 x 256 x 8 image | 2 x ZR33881-15 | 13.1 ms, 1 adder, real-time |
| 5 x 5 on 256 x 256 x 8 image | 2 x ZR33881-10 | 19.7 ms, 1 adder, real-time |
| 5 x 5 on 512 x 512 x 8 image | 2 x ZR33881-20 | 39.2 ms, 1 adder |
| 5 x 5 on 512 x 512 x 8 image | 2 x ZR33881-15 | 52.3 ms, 1 adder |
| 5 x 5 on 512 x 512 x 8 image | 2 x ZR33881-10 | 78.6 ms, 1 adder |
| 7 x 7 on 256 x 256 x 8 image | 2 x ZR33481-20 | 30.3 ms, 1 adder, real-time |
| 7 x 7 on 256 x 256 x 8 image | 2 x ZR33481-15 | 40.4 ms, 1 adder |
| 7 x 7 on 256 x 256 x 8 image | 2 x ZR33881-20 | 13.1 ms, 1 adder, real-time |
| 7 x 7 on 256 x 256 x 8 image | 2 x ZR33881-15 | 17.5 ms, 1 adder, real-time |
| 7 x 7 on 256 x 256 x 8 image | 2 x ZR33881-10 | 26.2 ms, 1 adder, real-time |
| 7 x 7 on 512 x 512 x 8 image | 2 x ZR33881-20 | 52.4 ms, 1 adder |
| 7 x 7 on 512 x 512 x 8 image | 4 x ZR33881-20 | 26.2 ms, 3 adders, real-time |
| 7 x 7 on 512 x 512 x 8 image | 8 x ZR33881-20 | 13.1 ms, 7 adders, real-time |

DFP DEVELOPMENT TOOLS

ZORAN provides a complete set of software and hardware tools to accelerate the design cycles for its customers.

■ The Digital Filter Processor Software (DFPS) is a complete menu-driven 1-D FIR filter design package which supports both Parks-McClellan and windowing techniques (Figure 10). DFPS includes signal generator, signal and filter analysis, coefficient

quantizer and graphics display capability and operates in PC XT/AT™ (MS-DOS™), or VAX™ (ULTRIX™ or VMS™) environments.

■ The Digital Filter Processor Board (DFPB) is a complete 20 MHz digital filter system which gives the designer a high-performance prototyping tool (Figure 11). The DFPB incorporates four DFP devices with 2K each of signal, coefficient and output memory. The system operates in stand alone mode or can be connected directly to an IBM-PC bus.

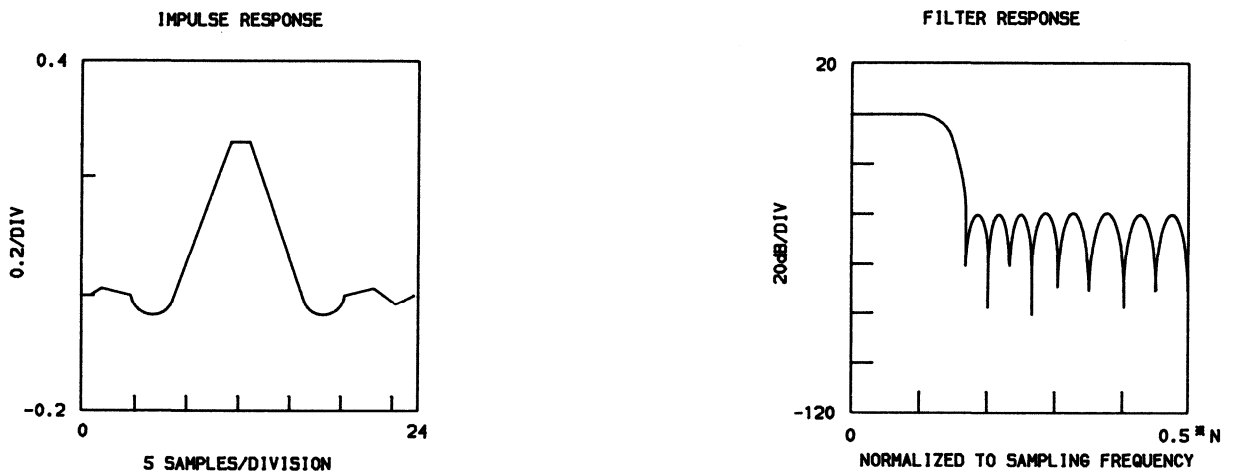


FIGURE 10. DFPS - DISPLAY OF IMPULSE AND FREQUENCY RESPONSES.

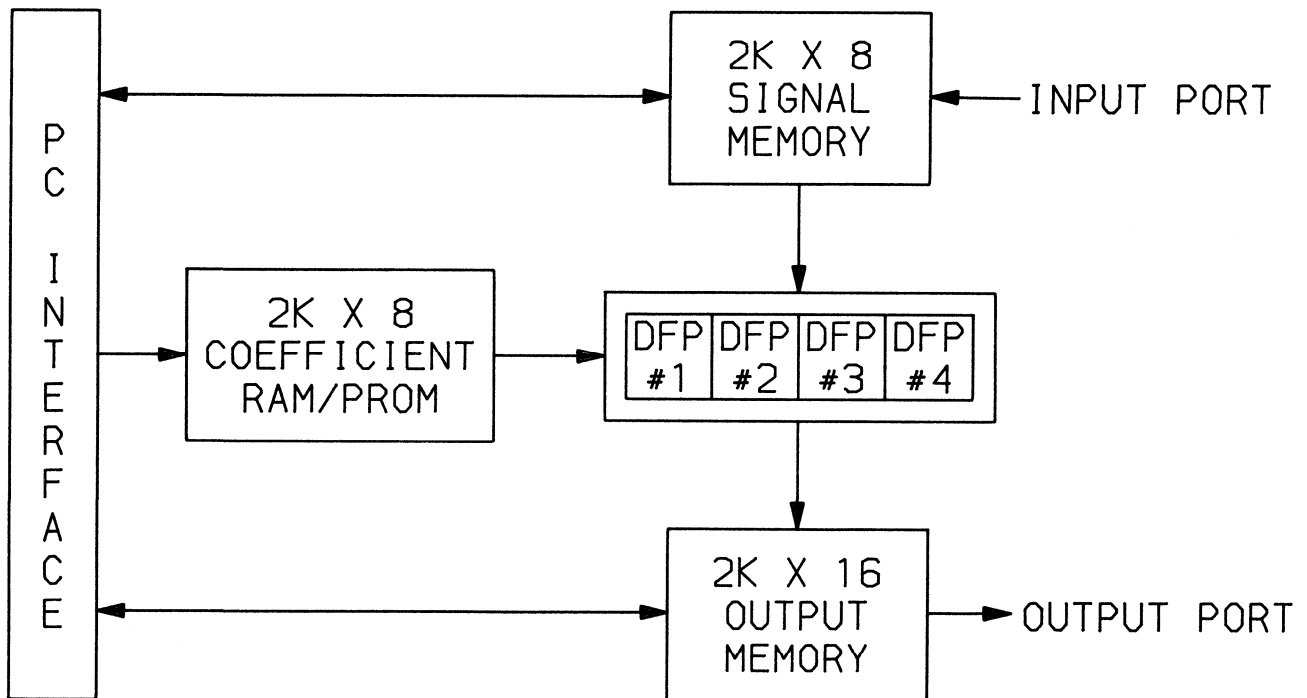


FIGURE 11. DFPB - BLOCK DIAGRAM OF FILTER DEVELOPMENT BOARD.

DISTINCTIVE FEATURES

- Four filter cells
- Up to 20 MHz sample rate
- 8-bit coefficients and signal data
- 26-bit accumulator per stage
- Filter lengths up to 1032 tap
- Shift-and-add output stage for combining filter outputs
- Expandable coefficient size, data size and filter length
- Decimation by 2, 3 or 4
- CMOS power dissipation characteristics

APPLICATIONS

- 1-D and 2-D FIR filters
- Radar/Sonar
- Digital video and audio
- Adaptive filters
- Echo cancellation
- Correlation/convolution
- Complex multiply-add
- Butterfly computation
- Matrix multiplication
- Sample rate converters

DESCRIPTION

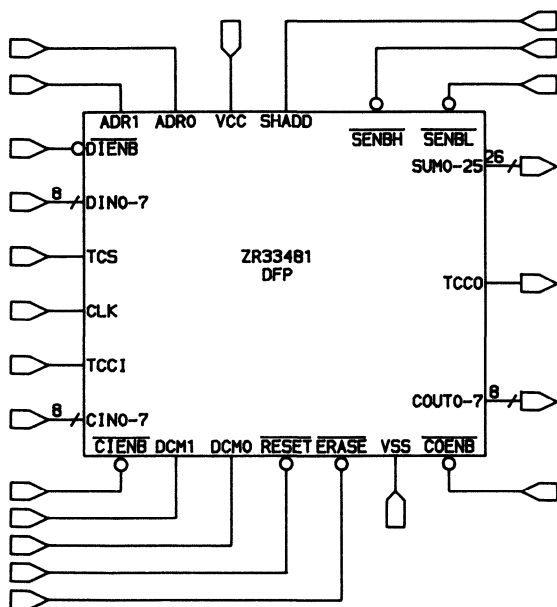
The ZR33481 is a video-speed Digital Filter Processor (DFP) designed to efficiently implement vector operations such as FIR digital filters. It is comprised of four filter cells cascaded internally and a shift-and-add output stage, all in a single integrated circuit. Each filter cell contains an 8×8 multiplier, three decimation registers and a 26-bit accumulator. The output stage contains an additional 26-bit accumulator which can add the contents of any filter cell accumulator to the output stage accumulator shifted right by eight bits. The ZR33481 has a maximum sample rate of 20 MHz. The effective multiply-accumulate (mac) rate is 80 MHz.

The ZR33881 DFP can be configured to process expanded coefficient and word sizes. Multiple DFPs can be cascaded for

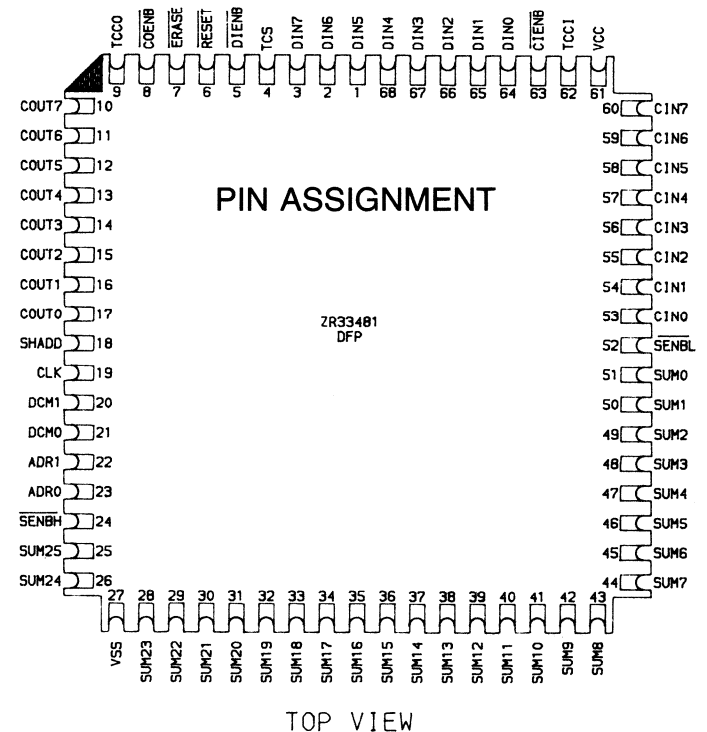
larger filter lengths without degrading the sample rate or a single DFP can process larger filter lengths at less than 20 MHz with multiple passes. The architecture permits processing filter lengths of over 1000 taps with the guarantee of no overflows. In practice, most filter coefficients are less than 1.0, making even larger filter lengths possible. The DFP provides for unsigned or two's complement arithmetic, independently selectable for coefficients and signal data.

Each DFP filter cell contains three resampling or decimation registers which permit output sample rate reduction at rates of 1/2, 1/3 or 1/4 the input sample rate. These registers also provide the capability to perform 2-D operations such as $N \times N$ spatial correlations/convolutions for image processing applications.

ZR33481 LOGIC SYMBOL



PIN ASSIGNMENT



INTERFACE SIGNAL DESCRIPTION

V_{cc} +5V power supply input

V_{ss} Power supply ground input.

CLK The CLK input provides the DFP system sample clock. The maximum clock frequency is 20 MHz. The minimum frequency is 200 KHz.

DIN0-7 These eight inputs are the data sample input bus. Eight-bit data samples are synchronously loaded through these pins to the X register of each filter cell of the DFP simultaneously. The $\overline{\text{DIENB}}$ signal enables loading, which is synchronous on the rising edge of the clock signal.

TCS The TCS input determines the number system interpretation of the data input samples on pins DIN0-7 as follows:

TCS = LOW → unsigned arithmetic

TCS = HIGH → two's complement arithmetic

The TCS signal is synchronously loaded into the X register in the same way as the DIN0-7 inputs.

$\overline{\text{DIENB}}$ A low on this input enables the data sample input bus (DIN0-7) to all the filter cells. A rising edge of the CLK signal occurring while $\overline{\text{DIENB}}$ is low will load the X register of every filter cell with the 8-bit value present on DIN0-7. A high on this input forces all the bits of the data sample input bus to zero; a rising CLK edge when $\overline{\text{DIENB}}$ is high will load the X register of every filter cell with all zeros. This signal is latched inside the DFP, delaying its effect by one clock internal to the DFP. Therefore it must be low during the clock cycle immediately preceding presentation of the desired data on the DIN0-7 inputs. Detailed operation is shown in later timing diagrams.

CIN0-7 These eight inputs are used to input the 8-bit coefficients, The coefficients are synchronously loaded into the C register of filter CELL0 if a rising edge of CLK occurs while $\overline{\text{CIENB}}$ is low. The $\overline{\text{CIENB}}$ signal is delayed by one clock as discussed below.

TCCI The TCCI input determines the number system interpretation of the coefficient inputs on pins CIN0-7 as follows:

TCCI = LOW → unsigned arithmetic

TCCI = HIGH → two's complement arithmetic

The TCCI signal is synchronously loaded into the C register in the same way as the CIN0-7 inputs.

$\overline{\text{CIENB}}$ A low on this input enables the C register of every filter cell and the D registers (decimation) of every filter cell according to the state of the DCM0-1 inputs. A rising edge of the CLK signal occurring while $\overline{\text{CIENB}}$ is low will load the C register and appropriate D registers with the coefficient data present at their inputs. This provides the mechanism for shifting the coefficients from cell to cell through the device. A high on this input freezes the contents of the C register and the D registers, ignoring the CLK signal. This signal is latched and delayed by one clock internal to the DFP. Therefore it must be low during the clock cycle immediately preceding presentation of the desired coefficient on the CIN0-7 inputs. Detailed operation is shown in later timing diagrams.

COUT0-7 These eight three-state outputs are used to output the 8-bit coefficients from filter CELL7. These outputs are enabled by the $\overline{\text{COENB}}$ signal low. These outputs may be tied to the CIN0-7 inputs of the same DFP to recirculate the coefficients, or they may be tied to the CIN0-7 inputs of another DFP to cascade DFPs for longer filter lengths.

TCCO The TCCO three-state output determines the number system representation of the coefficients output on COUT0-7. It tracks the TCCI signal to this same DFP. It should be tied to the TCCI input of the next DFP in a cascade of DFPs for increased filter lengths. This signal is enabled by $\overline{\text{COENB}}$ low.

$\overline{\text{COENB}}$ A low on the $\overline{\text{COENB}}$ input enables the COUT0-7 and the TCCO output. A high on this input places all these outputs in their high impedance state.

DCM0-1 These two inputs determine the use of the internal decimation registers as follows:

| <u>DCM1</u> | <u>DCM0</u> | <u>Decimation Function</u> |
|-------------|-------------|-------------------------------------|
| 0 | 0 | Decimation registers not used |
| 0 | 1 | One decimation register is used |
| 1 | 0 | Two decimation registers are used |
| 1 | 1 | Three decimation registers are used |

The coefficients pass from cell to cell at a rate determined by the number of decimation registers used. When no decimation registers are used, coefficients move from cell to cell on each clock. When one decimation register is used, coefficients move from cell to cell on every other clock, etc. These signals are latched and delayed by one clock internal to the DFP.

SUM0-25 These 26 three-state outputs are used to output the results of the internal filter cell computations. Individual filter cell results or the result of the shift-and-add output stage can be output. If an individual filter cell result is to be output, the ADR0-1 signals select the filter cell result. The SHADD signal determines whether the selected filter cell result or the output stage adder result is output. The signals $\overline{\text{SENBH}}$ and $\overline{\text{SENBL}}$ enable the most significant and least significant bits of the SUM0-25 result respectively. Both $\overline{\text{SENBH}}$ and $\overline{\text{SENBL}}$ may be enabled simultaneously if the system has a 26-bit or larger bus. However, individual enables are provided to facilitate use with a 16-bit bus.

$\overline{\text{SENBH}}$ A low on this input enables result bits SUM16-25. A high on this input places these bits in their high impedance state.

$\overline{\text{SENBL}}$ A low on this input enables result bits SUM0-15. A high on this input places these bits on their high impedance state.

ADR0-1 These two inputs select the one cell whose accumulator will be read through the output bus (SUM0-25) or added to the output stage accumulator. They also determine which accumulator will be cleared when $\overline{\text{ERASE}}$ is low. For selection of which accumulator to read through the output bus (SUM0-25) or which to add to the output stage accumulator, these inputs are latched in the DFP and delayed by one clock internal to the device. These inputs are latched in the DFP and delayed by one clock internal to the DFP. If the ADR0-1 lines remain at the same address for more than one clock, the output at SUM0-25 will not change to reflect any subsequent accumulator updates in the addressed cell. Only the result available during the first clock, when ADR0-1 selects the cell, will be output. This does not hinder normal operation since the ADR0-1 lines are changed sequentially. This feature facilitates the interface with slow memories where the output is required to be fixed for more than one clock.

SHADD The SHADD input controls the activation of the shift-and-add operation in the output stage. This signal is latched in the DFP and delayed by one clock internal to the device. Detailed explanation is given in the DFP Output Stage section.

$\overline{\text{RESET}}$ A low on this input synchronously clears all the internal registers, except the cell accumulators. It can be used with $\overline{\text{ERASE}}$ to also clear all the accumulators simultaneously. This signal is latched in the DFP and delayed by one clock internal to the DFP.

$\overline{\text{ERASE}}$ A low on this input synchronously clears the cell accumulator selected by the ADR0-1 signals. If $\overline{\text{RESET}}$ is also low simultaneously, all cell accumulators are cleared.

FUNCTIONAL DESCRIPTION

The Digital Filter Processor (DFP) is composed of four filter cells cascaded together and an output stage for combining or selecting filter cell outputs (Figure 1). Each filter cell contains a multiplier-accumulator and several registers (Figure 2). Each 8-bit coefficient is multiplied by a 8-bit data sample, with the result added to the 26-bit accumulator contents. The coefficient output of each cell is cascaded to the coefficient input of the next cell to its right.

DFP FILTER CELL

A 8-bit coefficient (CIN0-7, TCCI) enters each cell through the C register on the left and exits the cell on the right as signals COUT0-7 and TCCO. The coefficients may move directly from the C register to the output, exiting the cell on the clock following its entrance. When decimation is selected the coefficient exit is delayed by 1, 2 or 3 clocks by passing through one or more decimation registers (D1, D2 or D3).

The combination of D registers through which the coefficient passes is determined by the state of DCM0 and DCM1. The output signals (COUT0-7, TCCO) are connected to the CIN0-7 and TCCI of the next cell to its right. The $\overline{\text{COENB}}$ input signal enables the COUT0-7 and TCCO outputs of the rightmost cell to the COUT0-7 and TCCO pins of the DFP.

The C and D registers are enabled for loading by $\overline{\text{CIENB}}$. Loading is synchronous with CLK when $\overline{\text{CIENB}}$ is low. Note that $\overline{\text{CIENB}}$ is latched internally. It enables the register for loading after the next CLK following the onset of $\overline{\text{CIENB}}$ low. Actual loading occurs on the second CLK following the onset of $\overline{\text{CIENB}}$ low. Therefore $\overline{\text{CIENB}}$ must be low during the clock cycle immediately preceding presentation of the coefficient on the CIN0-7 inputs. In most basic FIR operations, $\overline{\text{CIENB}}$ will be low throughout the process, so this latching and delay sequence is only important during the initialization phase. When $\overline{\text{CIENB}}$ is high, the coefficients are frozen.

These registers are cleared synchronously under control of $\overline{\text{RESET}}$, which is latched and delayed exactly like $\overline{\text{CIENB}}$.

The output of the C register is one input of the multiplier.

The other input of the multiplier comes from the output of the X register. This register is loaded with a data sample from the DFP input signals DIN0-7 and TCS discussed above. The X register is enabled for loading by $\overline{\text{DIENB}}$. Loading is synchronous with CLK when $\overline{\text{DIENB}}$ is low. Note that $\overline{\text{DIENB}}$ is latched internally. It enables the register for loading after the next CLK following the onset of $\overline{\text{DIENB}}$ low. Actual loading occurs on the second CLK following the onset of $\overline{\text{DIENB}}$ low. Therefore, $\overline{\text{DIENB}}$ must be low during the clock cycle immediately preceding presentation of the data sample on the DIN0-7

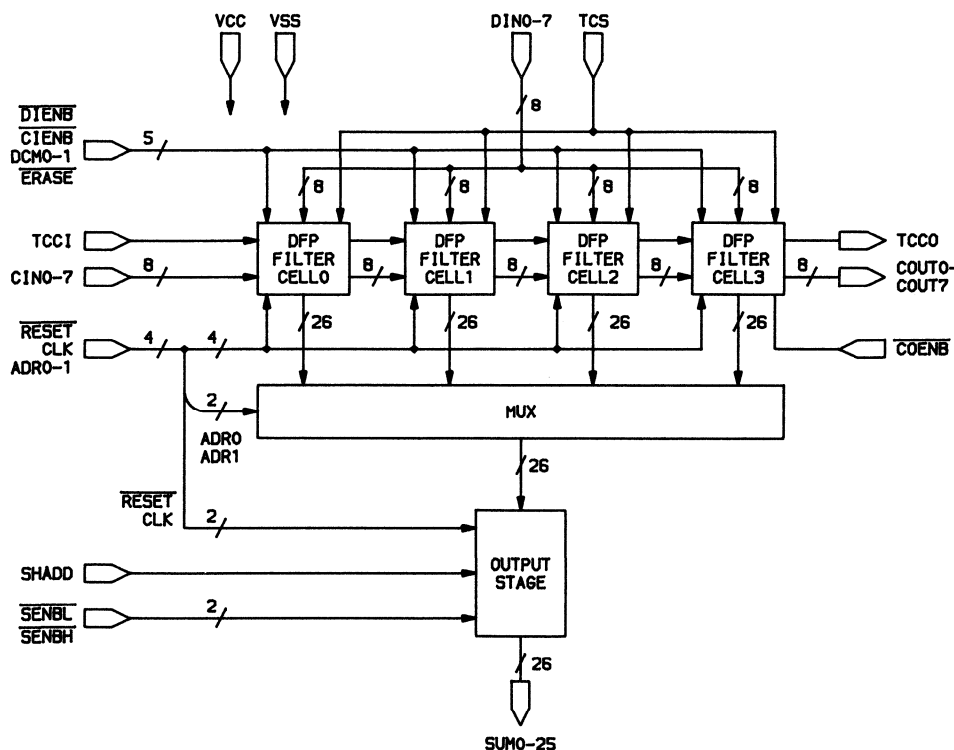


FIGURE 1. ZR33481 DFP BLOCK DIAGRAM.

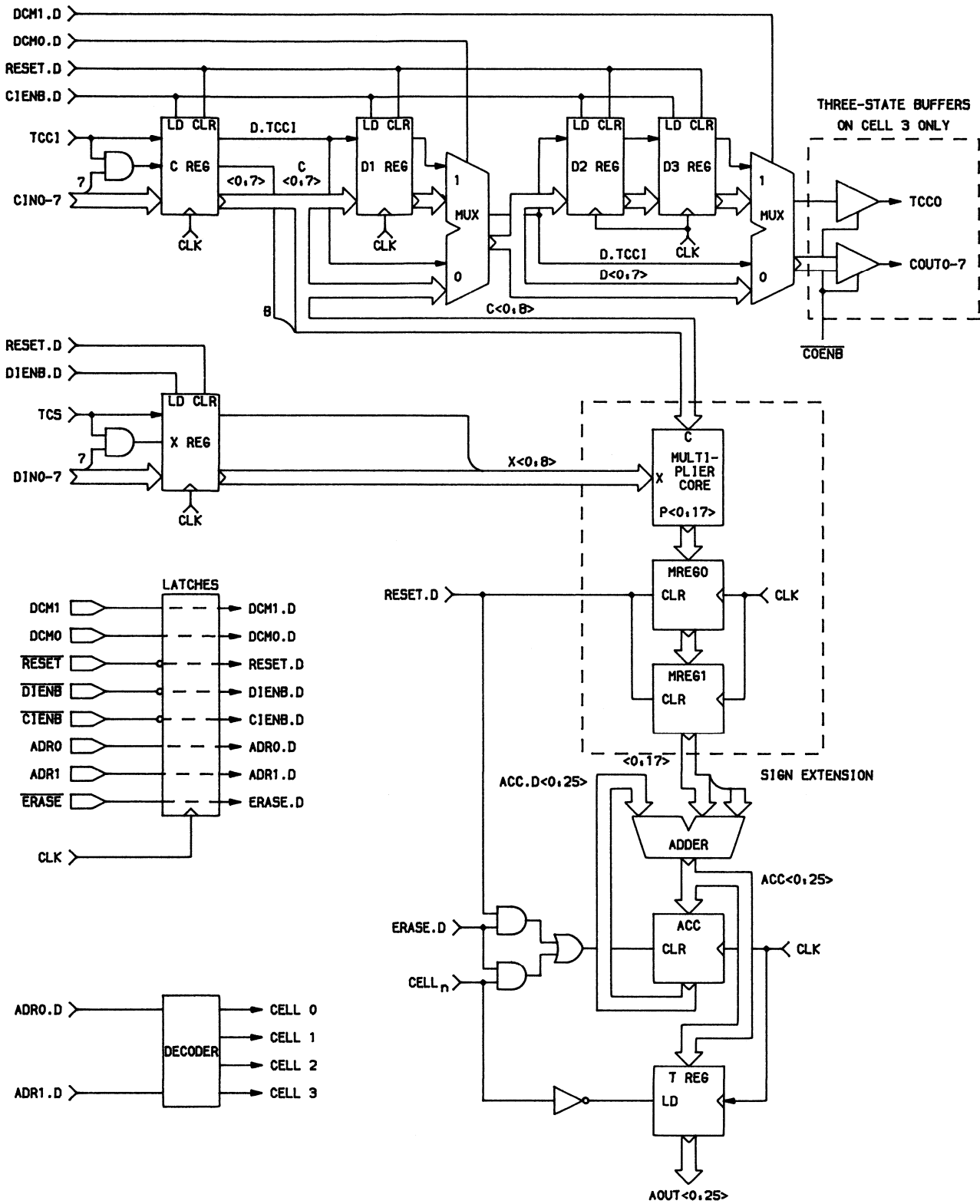


FIGURE 2. ZR33481 DFP FILTER CELL.

inputs. In most basic FIR operations, \overline{DIENB} will be low throughout the process, so this latching and delay sequence is only important during the initialization phase. When \overline{DIENB} is high, the X register is loaded with all zeros.

The multiplier is pipelined and is modeled as a multiplier core followed by two pipeline registers, MREG0 and MREG1 (Figure 2). The multiplier output is sign extended and input as one operand of the 26-bit adder. The other adder operand is the output of the 26-bit accumulator. The adder output is loaded synchronously into both the accumulator and the TREG.

The TREG loading is disabled by the cell select signal, CELLn, where n is the cell number. The cell select is decoded from the ADR0-1 signals to generate the TREG load enable. The cell select is inverted and applied as the load enable to the TREG. Operation is such that the TREG is loaded whenever the cell is not selected. Therefore, TREG is loaded every other clock except the clock following cell selection. The purpose of the TREG is to hold the result of a sum-of-products calculation during the clock when the accumulator is cleared to prepare for the next sum-of-products calculation. This allows continuous accumulation without wasting clocks.

The accumulator is loaded with the adder output every clock unless it is cleared. It is cleared synchronously in two ways. When \overline{RESET} and \overline{ERASE} are both low, the accumulator is cleared along with all other registers in the DFP. Since both \overline{ERASE} and \overline{RESET} are latched and delayed one clock internally, clearing occurs on the second CLK following the onset of both \overline{ERASE} and \overline{RESET} low.

The second accumulator clearing mechanism clears a single accumulator in a selected cell. The cell select signal, CELLn, decoded from ADR0-1 and the \overline{ERASE} signal, CELLn enable clearing of the accumulator on the next CLK.

The \overline{ERASE} and \overline{RESET} signals clear the DFP internal registers and states as follows:

| \overline{ERASE} | \overline{RESET} | CLEARING EFFECT |
|--------------------|--------------------|---|
| 1 | 1 | No clearing occurs, internal state remains same |
| 1 | 0 | Reset only active, all registers except accumulators are cleared, including the internal pipeline registers. |
| 0 | 1 | Erase only active, the accumulator whose address is given by the ADR0-1 inputs is cleared. |
| 0 | 0 | Both \overline{RESET} and \overline{ERASE} active, all accumulators as well as all other registers are cleared. |

THE DFP OUTPUT STAGE

The output stage consists of a 26-bit adder, 26-bit register, feed-back multiplexer from the register to the adder, an output multiplexer and a 26-bit three-state driver stage (Figure 3).

The 26-bit output adder can add any filter cell accumulator result to the 18 most significant bits of the output buffer. This result is stored back in the output buffer. This operation takes place in one clock period. The eight LSBs are lost. The filter cell accumulator is selected by the ADR0-1 inputs.

The 18 MSBs of the output buffer actually pass through the zero mux on their way to the output adder input. The zero mux is controlled by the SHADD input signal and selects either the output buffer 18 MSBs or all zeros for the adder input. A low on the SHADD input selects zero. A high on the SHADD input selects the output buffer MSBs, thus activating the shift-and-add operation. The SHADD signal is latched and delayed by one clock internally.

The 26 least significant bits (LSBs) from either a cell accumulator or the output buffer are output on the SUM0-25 bus. The output mux determines whether the cell accumulator selected by ADR0-1 or the output buffer is output to the bus. This mux is controlled by the SHADD input signal. Control is based on the state of the SHADD during two successive clocks; in other words, the output mux selection contains memory. If SHADD is low during a clock cycle and was low during the previous clock, the output mux selects the contents of the filter cell accumulator addressed by ADR0-1. Otherwise the output mux selects the contents of the output buffer.

If the ADR0-1 lines remain at the same address for more than one clock, the output at SUM0-25 will not change to reflect any subsequent accumulator updates in the addressed cell. Only the result available during the first clock when ADR0-1 selects the cell will be output. This does not hinder normal FIR operation since the ADR0-1 lines are changed sequentially. This feature facilitates the interface with slow memories where the output is required to be fixed for more than one clock.

The SUM0-25 output bus is controlled by the \overline{SENBH} and $\overline{SENB L}$ signals. A low on $\overline{SENB L}$ enables bits SUM0-15. A low on $\overline{SENB H}$ enables bits SUM16-25. Thus all 26 bits can be output simultaneously if the external system has a 26-bit or larger bus. If the external system bus is only 16 bits, the bits can be enabled in two groups of 16 and 9 bits (sign extended).

DFP ARITHMETIC

Both data samples and coefficients can be represented as either unsigned or two's complement numbers. The TCS and TCCI input signals determine the type of arithmetic representation. Internally all values are represented by a 9-bit two's complement number. The value of the additional ninth bit depends on arithmetic representation selected. For two's complement arithmetic, the sign is extended into the ninth bit. For unsigned arithmetic, bit 9 is 0.

The multiplier output is 18 bits and the accumulator is 26 bits. The accumulator width determines the maximum possible number of terms in the sum-of-products without overflow. The maximum number of terms depends also on the number system and the distribution of the coefficient and data values. As a worst case assume the coefficients and data samples are always at their maximum absolute values. Then the maximum numbers of terms in the sum products are:

| Number System | Maximum Number of Terms |
|--|-------------------------|
| Two unsigned vectors | 1032 |
| Two two's complement vectors | |
| • Two positive vectors | 2080 |
| • Two negative vectors | 2047 |
| • One positive and one negative vector | 2064 |
| One unsigned and one two's complement vector | |
| • positive two's complement vector | 1036 |
| • negative two's complement vector | 1028 |

For practical FIR filters, the coefficients are never all near maximum value, so even larger vectors are possible in practice.

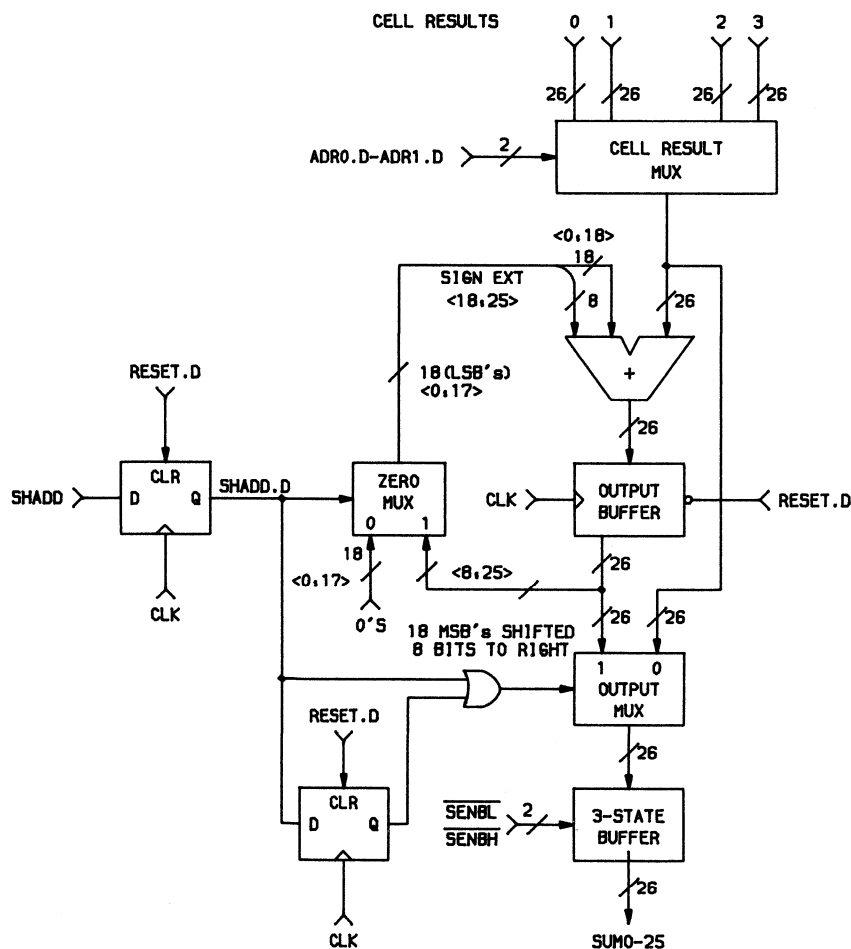
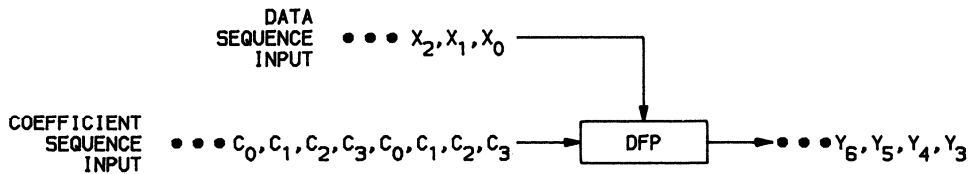


FIGURE 3. ZR33481 DFP OUTPUT STAGE.

BASIC FIR OPERATION

A simple, 20 MHz 4-tap filter example serves to illustrate more clearly the operation of the DFP. The sequence table (Table 1) shows the results of the multiply accumulate in each cell after each clock. The coefficient sequence, C_n , enters the DFP on

the left and moves from left to right through the cells. The data sample sequence, X_n , enters the DFP from the top, with each cell receiving the same sample simultaneously. Each cell accumulates the sum-of-products for one output point. Four sums-of-products are calculated simultaneously, but staggered in time so that a new output is available every system clock.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | SUM/CLR |
|-----|-------------------|-------------------|-------------------|-------------------|-----------|
| 0 | $C_3 \cdot X_0$ | 0 | 0 | 0 | — |
| 1 | $+ C_2 \cdot X_1$ | $C_3 \cdot X_1$ | 0 | 0 | — |
| 2 | $+ C_1 \cdot X_2$ | $+ C_2 \cdot X_2$ | $C_3 \cdot X_2$ | 0 | — |
| 3 | $+ C_0 \cdot X_3$ | $+ C_1 \cdot X_3$ | $+ C_2 \cdot X_3$ | $C_3 \cdot X_3$ | CELL0(Y3) |
| 4 | $C_3 \cdot X_4$ | $+ C_0 \cdot X_4$ | $+ C_1 \cdot X_4$ | $+ C_2 \cdot X_4$ | CELL1(Y4) |
| 5 | $+ C_2 \cdot X_5$ | $C_3 \cdot X_5$ | $+ C_0 \cdot X_5$ | $+ C_1 \cdot X_5$ | CELL2(Y5) |
| 6 | $+ C_1 \cdot X_6$ | $+ C_2 \cdot X_6$ | $C_3 \cdot X_6$ | $+ C_0 \cdot X_6$ | CELL3(Y6) |
| 7 | $+ C_0 \cdot X_7$ | $+ C_1 \cdot X_7$ | $+ C_2 \cdot X_7$ | $C_3 \cdot X_7$ | CELL0(Y7) |

TABLE 1. ZR33481 20 MHz, 4-TAP FIR FILTER SEQUENCE.

Detailed operation of the DFP to perform a basic 4-tap, 8-bit coefficient, 8-bit data, 20 MHz FIR filter is best understood by observing the schematic (Figure 4) and timing diagram (Figure 5). The internal pipeline length of the DFP is four (4) clock cycles, corresponding to the register levels CREG (or XREG), MREG0, MREG1, and TREG (Figures 2 and 3). Therefore the delay from presentation of data and coefficients at the DIN0-7 and CIN0-7 inputs to a sum appearing at the SUM0-25 output is:

$$k + T_d$$

where

$$k = \text{filter length}$$

$$T_d = 4, \text{ the internal pipeline delay of DFP}$$

After the pipeline has filled, a new output sample is available every clock. The delay to last sample output from last sample input is T_d .

The output sums, Y_n , shown in the timing diagram are derived from the sum-of-products equation:

$$Y(n) = C(0) \times X(n) + C(1) \times X(n-1) + C(2) \times X(n-2) + C(3) \times X(n-3) + C(4) \times X(n-4) + C(5) \times X(n-5) + C(6) \times X(n-6) + C(7) \times X(n-7)$$

EXTENDED FIR FILTER LENGTH

Filter lengths greater than four taps can be created by either cascading together multiple DFPs or "reusing" a single DFP. Using multiple devices, an FIR filter of over 1024 taps can be constructed to operate at a 20 MHz sample rate. Using a single DFP clocked at 20 MHz, an FIR filter of over 1024 taps can be constructed to operate at less than a 20 MHz sample rate. Combinations of these two techniques are also possible.

CASCADE CONFIGURATION

To design a filter length $L > 4$, $L/4$ DFPs are cascaded by connecting the COUT0-7 outputs of the (i)th DFP to the CIN0-7 inputs of the (i+1)th DFP. The DIN0-7 inputs and SUM0-25 outputs of all the DFPs are also tied together. A specific example of two cascaded DFPs illustrates the technique (Figure 6). Timing (Figure 7) is similar to the simple 4-tap FIR, except the $\overline{\text{ERASE}}$ and $\overline{\text{SENBL}}/\overline{\text{SENBH}}$ signals must be enabled independently of the two DFPs in order to clear the correct accumulators and enable the SUM0-25 output signals at the proper times.

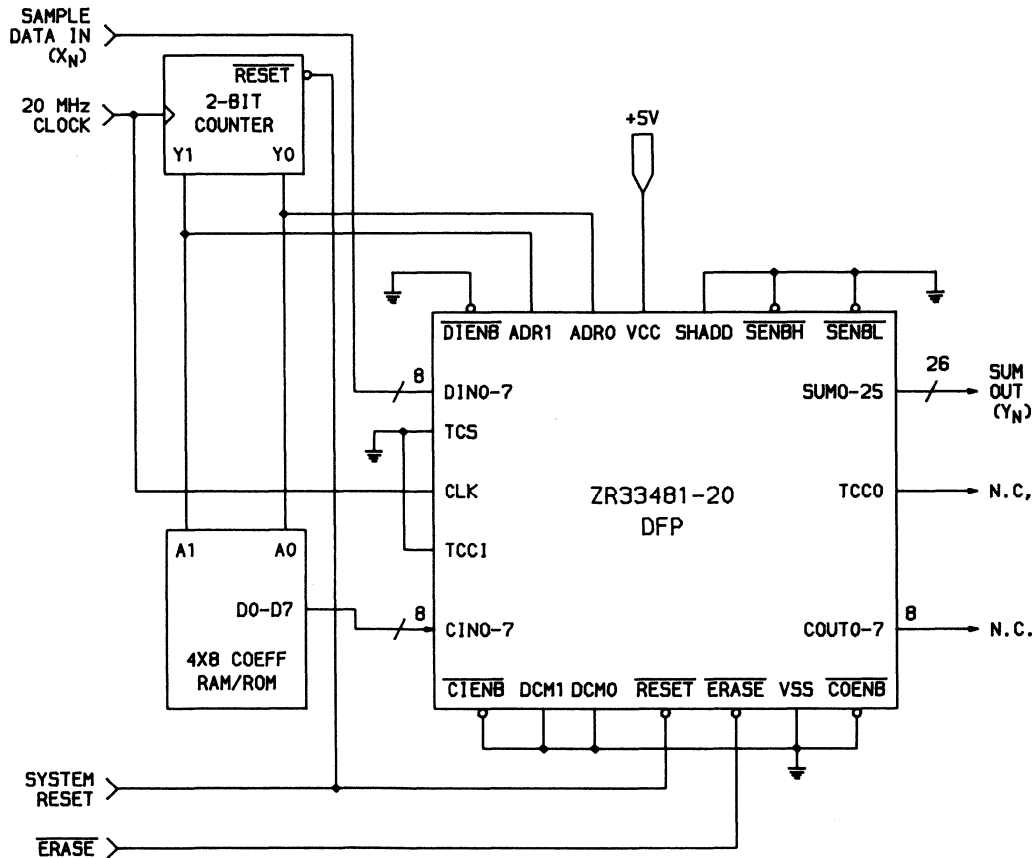


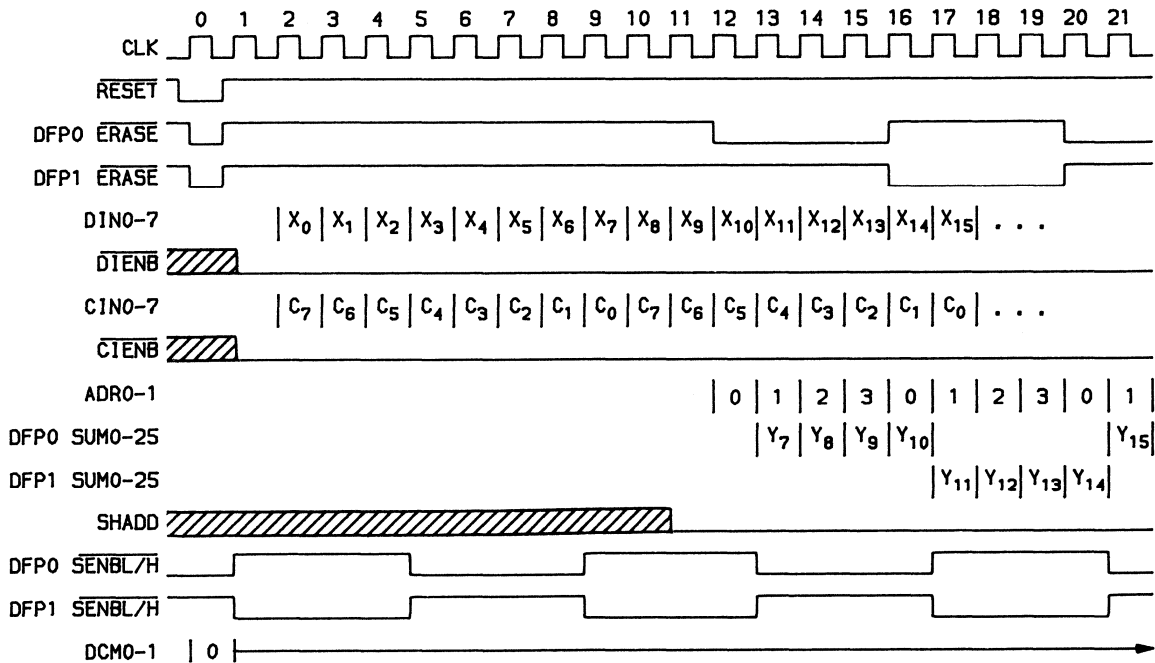
FIGURE 4. ZR33481 20 MHz, 4-TAP FIR FILTER APPLICATION SCHEMATIC.

SINGLE DFP CONFIGURATION

Using a single DFP, a filter of length $L > 4$ can be constructed by processing in $L/4$ passes as illustrated in the following table (Table 2) for a 8-tap FIR. Each pass is composed of $T_p = 7 + L$ cycles and computes four output samples. In pass i , the sample with indices $i74$ to $i74 + (L-1)$ enter the $DIN0-7$ inputs. The coefficients C_0-C_{L-1} enter the $CIN0-7$ inputs, followed by three zeros. As these zeros are entered, the result samples are output and the accumulators reset. Initial filling of the pipeline is not shown in this sequence table. Filter outputs can be put through a FIFO to even out the sample rate.

EXTENDED COEFFICIENT AND DATA SAMPLE WORD SIZE

The sample and coefficient word size can be extended by utilizing several DFPs in parallel to get the maximum sample rate or a single DFP with resulting lower sample rates. The technique is to compute partial products of 8×8 and combine these partial products by shifting and adding to obtain the final result. The shifting and adding can be accomplished with external adders (for full speed) or with the DFP's shift-and-add mechanism contained in its output stage (at reduced speed).



$$Y_N = \sum_{K=0}^7 C_K \cdot X_{N-K}$$

FIGURE 5. ZR33481 20 MHz, 4-TAP FIR FILTER TIMING.

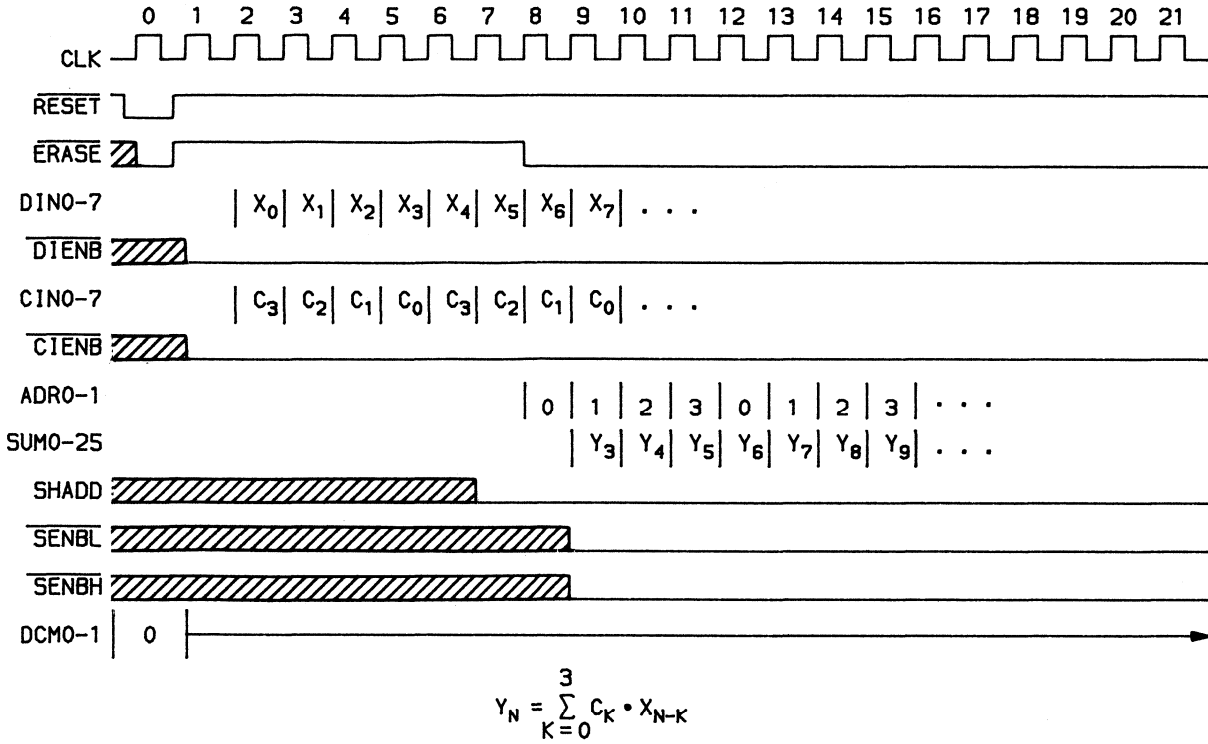


FIGURE 6. ZR33481 20 MHz, 8-TAP FIR FILTER CASCADE APPLICATION SCHEMATIC.

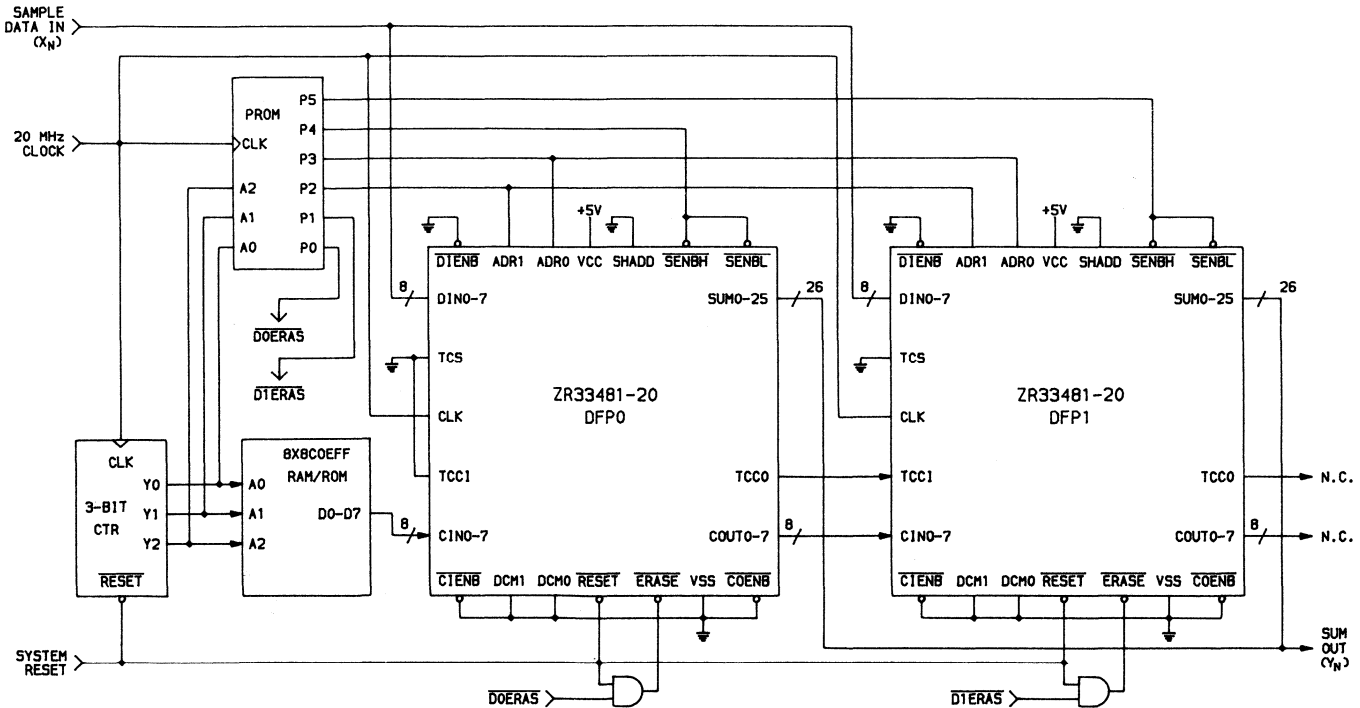


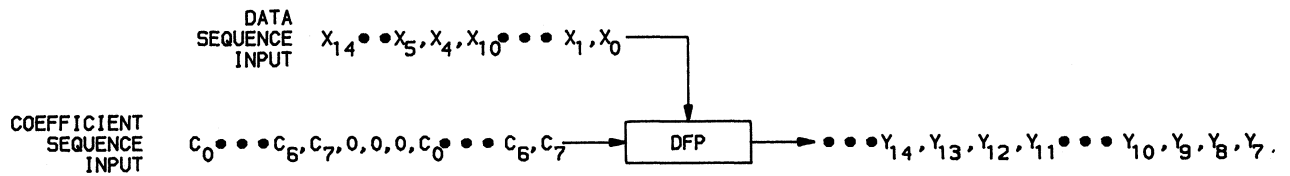
FIGURE 7. ZR33481 20 MHz 8-TAP FILTER TIMING USING TWO CASCADED ZR33481s.

DECIMATION/RESAMPLING

The ZR33481 DFP provides a mechanism for decimating by factors of 2, 3 or 4. From the DFP filter cell block diagram (Figure 2), note the three D registers and two multiplexers in the coefficient pass through the cell. These allow the coeffi-

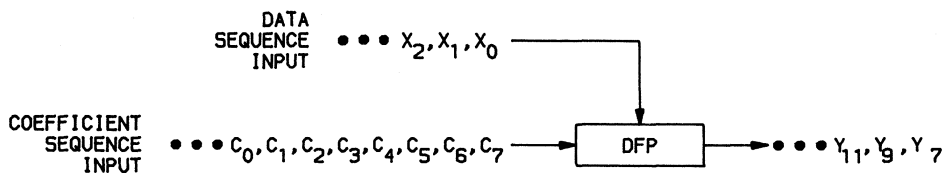
cients to be delayed by 1, 2 or 3 clocks through the cell. The sequence table (Table 3) for a decimate-by-two filter illustrates the technique.

Detailed timing for a 20 MHz input sample rate, 10 MHz output sample rate (i.e., decimate-by-two), 8-tap FIR filter, including pipelining, is shown in Figure 8.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | SUM/CLR |
|-----|----------------------|----------------------|----------------------|----------------------|------------|
| 0 | $C_7 \cdot X_0$ | 0 | 0 | 0 | — |
| 1 | $+ C_6 \cdot X_1$ | $C_7 \cdot X_1$ | 0 | 0 | — |
| 2 | $+ C_5 \cdot X_2$ | $+ C_6 \cdot X_2$ | $C_7 \cdot X_2$ | 0 | — |
| 3 | $+ C_4 \cdot X_3$ | $+ C_5 \cdot X_3$ | $+ C_6 \cdot X_3$ | $C_7 \cdot X_3$ | — |
| 4 | $+ C_3 \cdot X_4$ | $+ C_4 \cdot X_4$ | $+ C_5 \cdot X_4$ | $+ C_6 \cdot X_4$ | — |
| 5 | $+ C_2 \cdot X_5$ | $+ C_3 \cdot X_5$ | $+ C_4 \cdot X_5$ | $+ C_5 \cdot X_5$ | — |
| 6 | $+ C_1 \cdot X_6$ | $+ C_2 \cdot X_6$ | $+ C_3 \cdot X_6$ | $+ C_4 \cdot X_6$ | — |
| 7 | $+ C_0 \cdot X_7$ | $+ C_1 \cdot X_7$ | $+ C_2 \cdot X_7$ | $+ C_3 \cdot X_7$ | CELL0(Y7) |
| 8 | 0 | $+ C_0 \cdot X_8$ | $+ C_1 \cdot X_8$ | $+ C_2 \cdot X_8$ | CELL1(Y8) |
| 9 | 0 | 0 | $+ C_0 \cdot X_9$ | $+ C_1 \cdot X_9$ | CELL2(Y9) |
| 10 | 0 | 0 | 0 | $+ C_0 \cdot X_{10}$ | CELL3(Y10) |
| 11 | $C_7 \cdot X_4$ | 0 | 0 | 0 | — |
| 12 | $+ C_6 \cdot X_5$ | $C_7 \cdot X_5$ | 0 | 0 | — |
| 13 | $+ C_5 \cdot X_6$ | $+ C_6 \cdot X_6$ | $C_7 \cdot X_6$ | 0 | — |
| 14 | $+ C_4 \cdot X_7$ | $+ C_5 \cdot X_7$ | $+ C_6 \cdot X_7$ | $C_7 \cdot X_7$ | — |
| 15 | $+ C_3 \cdot X_8$ | $+ C_4 \cdot X_8$ | $+ C_5 \cdot X_8$ | $+ C_6 \cdot X_8$ | — |
| 16 | $+ C_2 \cdot X_9$ | $+ C_3 \cdot X_9$ | $+ C_4 \cdot X_9$ | $+ C_5 \cdot X_9$ | — |
| 17 | $+ C_1 \cdot X_{10}$ | $+ C_2 \cdot X_{10}$ | $+ C_3 \cdot X_{10}$ | $+ C_4 \cdot X_{10}$ | — |
| 18 | $+ C_0 \cdot X_{11}$ | $+ C_1 \cdot X_{11}$ | $+ C_2 \cdot X_{11}$ | $+ C_3 \cdot X_{11}$ | CELL0(Y11) |
| 19 | 0 | $+ C_0 \cdot X_{12}$ | $+ C_1 \cdot X_{12}$ | $+ C_2 \cdot X_{12}$ | CELL1(Y12) |
| 20 | 0 | 0 | $+ C_0 \cdot X_{13}$ | $+ C_1 \cdot X_{13}$ | CELL2(Y13) |
| 21 | 0 | 0 | 0 | $+ C_0 \cdot X_{14}$ | CELL3(Y14) |

TABLE 2. ZR33481 8-TAP FIR FILTER SEQUENCE USING SINGLE DFP.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | SUM/CLR |
|-----|----------------------|----------------------|----------------------|----------------------|------------|
| 0 | $C_7 \cdot X_0$ | 0 | | | — |
| 1 | $+ C_6 \cdot X_1$ | 0 | | | — |
| 2 | $+ C_5 \cdot X_2$ | $C_7 \cdot X_2$ | | | — |
| 3 | $+ C_4 \cdot X_3$ | $+ C_6 \cdot X_3$ | | | — |
| 4 | $+ C_3 \cdot X_4$ | $+ C_5 \cdot X_4$ | $C_7 \cdot X_4$ | | — |
| 5 | $+ C_2 \cdot X_5$ | $+ C_4 \cdot X_5$ | $+ C_6 \cdot X_5$ | | — |
| 6 | $+ C_1 \cdot X_6$ | $+ C_3 \cdot X_6$ | $+ C_5 \cdot X_6$ | $C_7 \cdot X_6$ | — |
| 7 | $+ C_0 \cdot X_7$ | $+ C_2 \cdot X_7$ | $+ C_4 \cdot X_7$ | $+ C_6 \cdot X_7$ | CELL0(Y7) |
| 8 | $C_7 \cdot X_8$ | $+ C_1 \cdot X_8$ | $+ C_3 \cdot X_8$ | $+ C_5 \cdot X_8$ | CELL0(Y7) |
| 9 | $+ C_6 \cdot X_9$ | $+ C_0 \cdot X_9$ | $+ C_2 \cdot X_9$ | $+ C_4 \cdot X_9$ | CELL1(Y9) |
| 10 | $+ C_5 \cdot X_{10}$ | $C_7 \cdot X_{10}$ | $+ C_1 \cdot X_{10}$ | $+ C_3 \cdot X_{10}$ | CELL1(Y9) |
| 11 | $+ C_4 \cdot X_{11}$ | $+ C_6 \cdot X_{11}$ | $+ C_0 \cdot X_{11}$ | $+ C_2 \cdot X_{11}$ | CELL2(Y11) |
| 12 | $+ C_3 \cdot X_{12}$ | $+ C_5 \cdot X_{12}$ | $C_7 \cdot X_{12}$ | $+ C_1 \cdot X_{12}$ | CELL2(Y11) |
| 13 | $+ C_2 \cdot X_{13}$ | $+ C_4 \cdot X_{13}$ | $+ C_6 \cdot X_{13}$ | $+ C_0 \cdot X_{13}$ | CELL3(Y13) |
| 14 | $+ C_1 \cdot X_{14}$ | $+ C_3 \cdot X_{14}$ | $+ C_5 \cdot X_{14}$ | $C_7 \cdot X_{14}$ | CELL3(Y13) |
| 15 | $+ C_0 \cdot X_{15}$ | $+ C_2 \cdot X_{15}$ | $+ C_4 \cdot X_{15}$ | $+ C_6 \cdot X_{15}$ | CELL0(Y15) |

TABLE 3. ZR33481 8-TAP DECIMATE-BY-TWO FIR FILTER SEQUENCE, 20 MHz IN, 10 MHz OUT.

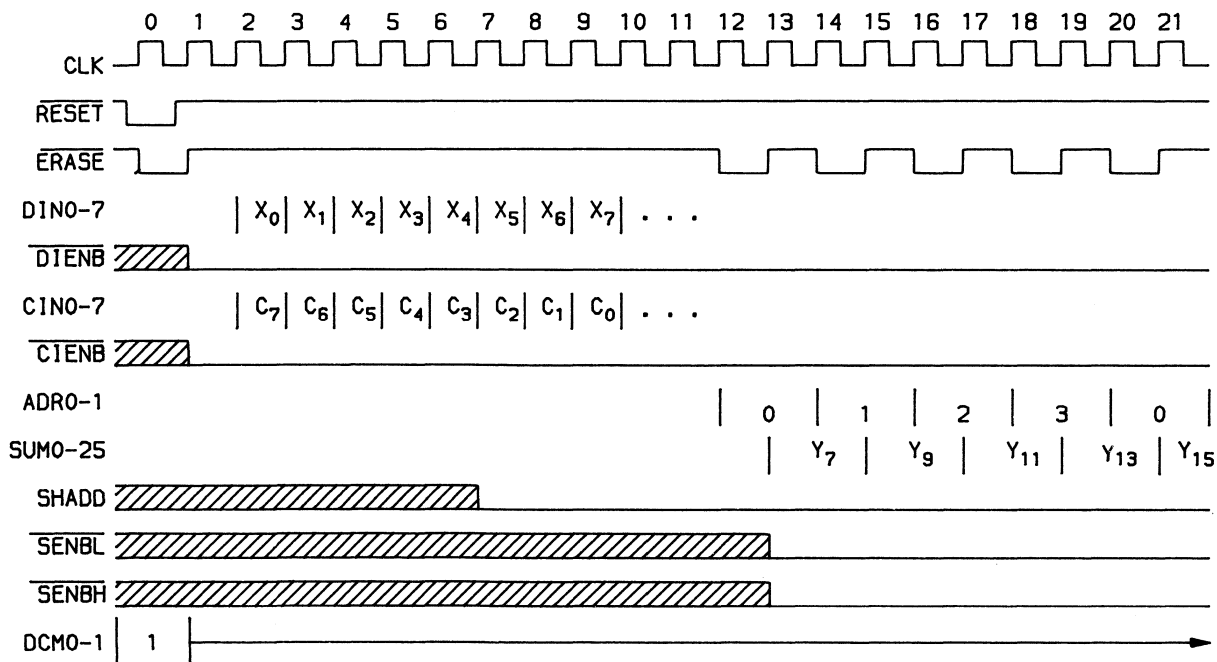


FIGURE 8. ZR33481 8-TAP DECIMATE-BY-TWO FIR FILTER TIMING, 20 MHz IN 10 MHz OUT.

OTHER DFP APPLICATIONS

The ZR33481 DFP is a versatile device with many applications beyond simple FIR filtering. The following list is a small sample of the many applications possible with the DFP. Implementations of these applications are discussed in greater detail in the various ZR33481 Zoran Application Notes.

- Higher precision arithmetic (e.g. 16×16)
 - Single DFP implementation
 - Multiple DFP implementation
- Higher bandwidth (40 MHz and up)
- Decimation/interpolation
- 2-D FIR spatial filtering/convolution/correlation (e.g. 3×3 kernel convolution at 20 MHz rate)
- Complex multiply/accumulate
- Adaptive filters
 - Echo cancellation
 - Adaptive equalization
- Butterfly computation
- Reverberation generators
- DFT
- Beam former
- Decoders
- Video Composite/Component

DFP TIMING PARAMETERS

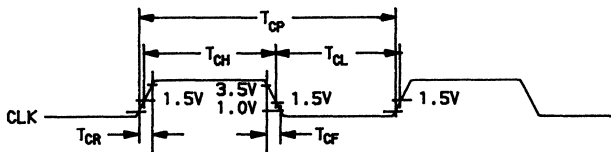
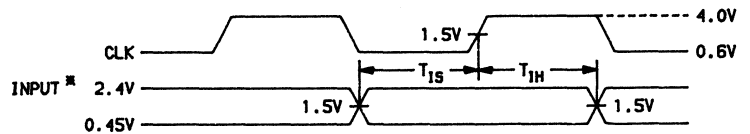
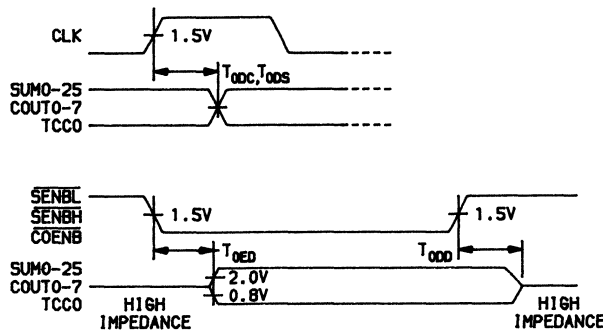


FIGURE 9. CLOCK AC PARAMETERS.



INPUT INCLUDES $\overline{DINO-7}$, $\overline{CINO-7}$, \overline{DTENB} , \overline{CTENB} , \overline{ERASE} , \overline{RESET} , $\overline{DCMO-1}$, $\overline{ADRO-1}$, TCS, TCC1, SHADD

FIGURE 10. INPUT SETUP AND HOLD.



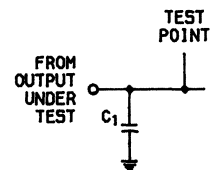
SUM0-25, COUT0-7, TCC0 ARE ASSUMED NOT TO BE IN HIGH-IMPEDANCE STATE

FIGURE 11. SUM0-25 COUT0-7, TCC0 OUTPUT DELAYS.



A.C. TESTING, INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". INPUT AND OUTPUT TIMING MEASUREMENTS ARE MADE AT 1.5V FOR BOTH A LOGIC "1" AND "0"

FIGURE 12. A.C. TESTING INPUT, OUTPUT WAVEFORM.



NOTE: $C_1 = 50\text{pF}$ INCLUDING STRAY AND WIRING CAPACITANCE

FIGURE 13. NORMAL A.C. TEST LOAD.

ABSOLUTE MAXIMUM RATINGS

(Above which useful life may be impaired)

- Temperature Under Bias -55°C to $+125^{\circ}\text{C}$
- Storage Temperature -65°C to $+150^{\circ}\text{C}$
- Supply Voltage to Ground Potential
 - Continuous -0.5V to $+7.0\text{V}$
- DC Voltage Applied to Outputs for High
 - Output State -0.5V to $+7.0\text{V}$
- DC Input Voltage -0.5V to $+5.5\text{V}$
- DC Output Current, into Outputs
 - (not to exceed 200 mA total) 20mA/output
 - Latch up Trigger Current $>200\text{mA}$

OPERATING RANGE

Commercial (C) Devices

- Temperature $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$
- Supply Voltage $4.75\text{V} \leq V_{CC} \leq 5.25\text{V}$

Physical Dimensions Note: All dimensions are in inches.

Zoran Corporation cannot assume responsibility for use of any circuitry described other than circuitry embodied in a Zoran product.

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC CHARACTERISTICS OVER OPERATING RANGE unless otherwise specified

| Symbol | Parameter | Min | Max | Units | Test Conditions |
|----------|------------------------------------|------|----------------|---------------|---|
| V_{IL} | Input Low Voltage | -0.5 | 0.8 | V | |
| V_{IH} | Input High Voltage | 2.2 | $V_{CC} + 0.5$ | V | |
| V_{OL} | Output Low Voltage | | 0.45 | V | $I_{OL} = 2\text{ mA}$ |
| V_{OH} | Output High Voltage | 2.4 | | V | $I_{OH} = -400\ \mu\text{A}$ |
| I_{CC} | Power Supply Current | | 100 | mA | $T_A = 0^{\circ}\text{C}$, $V_{CC} = \text{Max}^1$ |
| I_{LI} | Input Leakage Current | | ± 10 | μA | $0 < V_{IN} < V_{CC}$ |
| I_{LO} | Output Leakage Current | | ± 10 | μA | $0.45 < V_{OUT} < V_{CC}$ |
| V_{CL} | Clock in Low Voltage | -0.5 | 0.6 | V | |
| V_{CH} | Clock in High Voltage | 4.0 | $V_{CC} + 0.5$ | V | |
| C_{IN} | Input Capacitance | | 10 | pF | $f_C = 1\text{ MHz}$ |
| C_{IO} | I/O, Clock, and Output Capacitance | | 10 | pF | $f_C = 1\text{ MHz}$ |

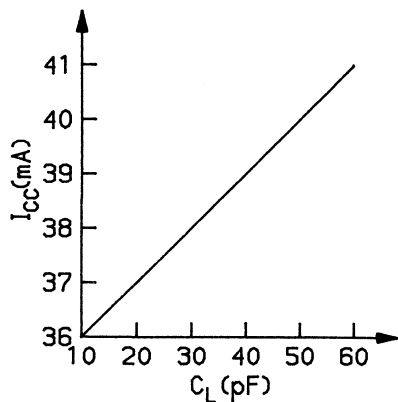


FIGURE 14.

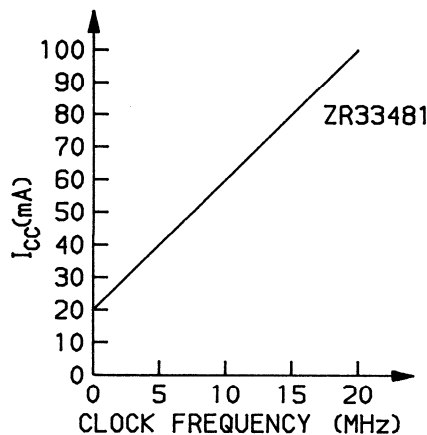


FIGURE 15.

AC CHARACTERISTICS OVER OPERATING RANGE ZR33481-20

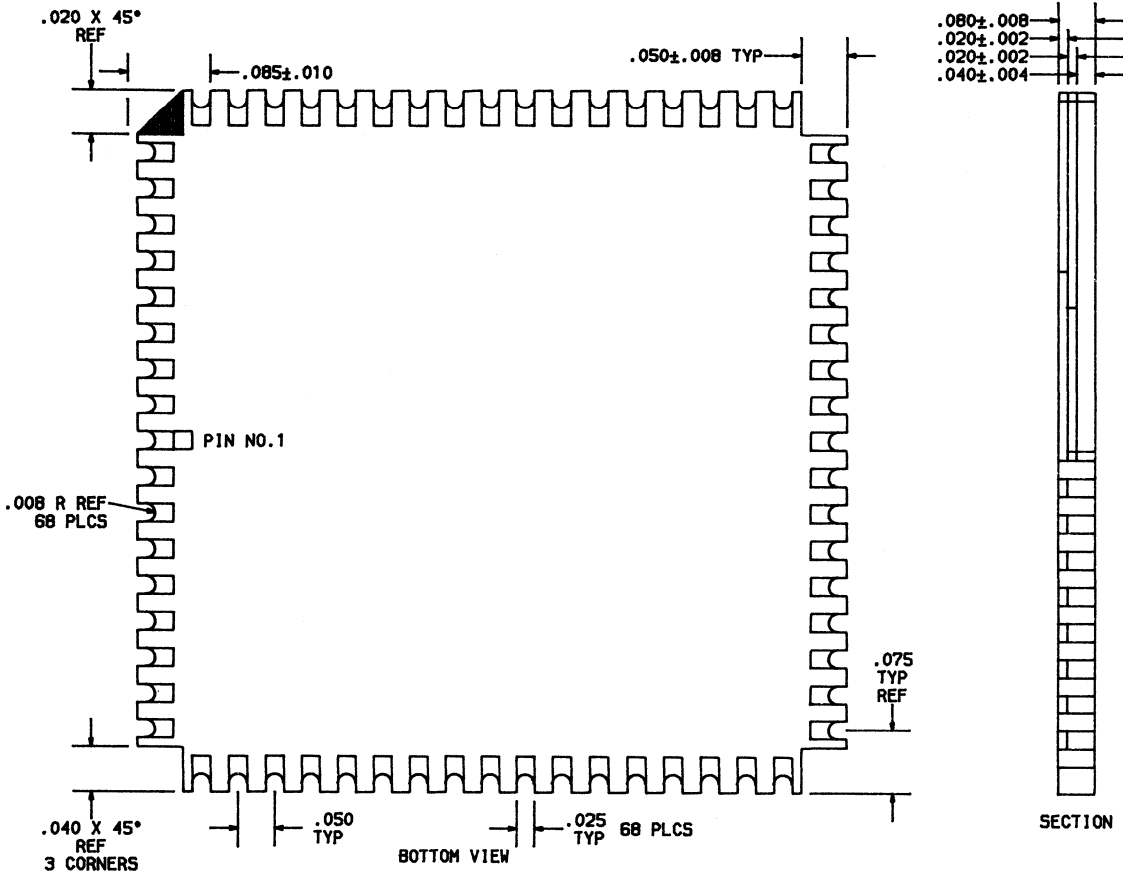
| Symbol | Parameter | COM ZR33481JC-20 | | Units | Test Conditions |
|------------------|---------------------------------|---------------------|------|-------|-----------------|
| | | Min | Max | | |
| T _{CP} | Clock Period | 50 | 5000 | ns | |
| T _{CL} | Clock Low | 22 | | ns | |
| T _{CH} | Clock High | 22 | | ns | |
| T _{IR} | Input Rise | | 5 | ns | 1.0V to 3.5V |
| T _{IF} | Input Fall | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 10 | | ns | |
| T _{IH} | Input Hold | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 40 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 20 | ns | |
| T _{ODD} | Output Disable Delay | | 20 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 40 | ns | |

AC CHARACTERISTICS OVER OPERATING RANGE ZR33481-15

| Symbol | Parameter | COM ZR33481JC-15 | | Units | Test Conditions |
|------------------|---------------------------------|---------------------|------|-------|-----------------|
| | | Min | Max | | |
| T _{CP} | Clock Period | 67 | 5000 | ns | |
| T _{CL} | Clock Low | 30 | | ns | |
| T _{CH} | Clock High | 30 | | ns | |
| T _{IR} | Input Rise | | 5 | ns | 1.0V to 3.5V |
| T _{IF} | Input Fall | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 14 | | ns | |
| T _{IH} | Input Hold | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 50 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 25 | ns | |
| T _{ODD} | Output Disable Delay | | 25 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 50 | ns | |

AC CHARACTERISTICS OVER OPERATING RANGE ZR33481-10

| Symbol | Parameter | COM ZR33481JC-10 | | Units | Test Conditions |
|------------------|---------------------------------|---------------------|------|-------|-----------------|
| | | Min | Max | | |
| T _{CP} | Clock Period | 100 | 5000 | ns | |
| T _{CL} | Clock Low | 40 | | ns | |
| T _{CH} | Clock High | 40 | | ns | |
| T _{IR} | Input Rise | | 5 | ns | 1.0V to 3.5V |
| T _{IF} | Input Fall | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 30 | | ns | |
| T _{IH} | Input Hold | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 65 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 40 | ns | |
| T _{ODD} | Output Disable Delay | | 40 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 65 | ns | |



ORDERING INFORMATION

| Speed | Package Type | Temperature Range | Order Number |
|--------|--------------|---|--------------|
| 10 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33481JC-10 |
| 15 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33481JC-15 |
| 20 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33481JC-20 |

INTERFACE SIGNAL DESCRIPTION

V_{cc} +5V power supply input
V_{ss} Power supply ground input.
CLK The CLK input provides the DFP system sample clock. The maximum clock frequency is 15 MHz for the ZR33881-15. The minimum frequency is 200 KHz.
DIN0-7 These eight inputs are the data sample input bus. Eight-bit data samples are synchronously loaded through these pins to the X register of each filter cell of the DFP simultaneously. The $\overline{\text{DIENB}}$ signal enables loading, which is synchronous on the rising edge of the clock signal.
TCS The TCS input determines the number system interpretation of the data input samples on pins DIN0-7 as follows:
 TCS = LOW → unsigned arithmetic
 TCS = HIGH → two's complement arithmetic
 The TCS signal is synchronously loaded into the X register in the same way as the DIN0-7 inputs.
 $\overline{\text{DIENB}}$ A low on this input enables the data sample input bus (DIN0-7) to all the filter cells. A rising edge of the CLK signal occurring while $\overline{\text{DIENB}}$ is low will load the X register of every filter cell with the 8-bit value present on DIN0-7. A high on this input forces all the bits of the data sample input bus to zero; a rising CLK edge when $\overline{\text{DIENB}}$ is high will load the X register of every filter cell with all zeros. This signal is latched inside the DFP, delaying its effect by one clock internal to the DFP. Therefore it must be low during the clock cycle immediately preceding presentation of the desired data on the DIN0-7 inputs. Detailed operation is shown in later timing diagrams.
CIN0-7 These eight inputs are used to input the 8-bit coefficients, The coefficients are synchronously loaded into the C register of filter CELL0 if a rising edge of CLK occurs while $\overline{\text{CIENB}}$ is low. The $\overline{\text{CIENB}}$ signal is delayed by one clock as discussed below.
TCCI The TCCI input determines the number system interpretation of the coefficient inputs on pins CIN0-7 as follows:
 TCCI = LOW → unsigned arithmetic
 TCCI = HIGH → two's complement arithmetic
 The TCCI signal is synchronously loaded into the C register in the same way as the CIN0-7 inputs.

$\overline{\text{CIENB}}$ A low on this input enables the C register of every filter cell and the D registers (decimation) of every filter cell according to the state of the DCM0-1 inputs. A rising edge of the CLK signal occurring while $\overline{\text{CIENB}}$ is low will load the C register and appropriate D registers with the coefficient data present at their inputs. This provides the mechanism for shifting the coefficients from cell to cell through the DFP. A high on this input freezes the contents of the C register and the D registers, ignoring the CLK signal. This signal is latched and delayed by one clock internal to the DFP. Therefore it must be low during the clock cycle immediately preceding presentation of the desired coefficient on the CIN0-7 inputs. Detailed operation is shown in later timing diagrams.
COUT0-7 These eight three-state outputs are used to output the 8-bit coefficients from filter CELL7. These outputs are enabled by the $\overline{\text{COENB}}$ signal low. These outputs may be tied to the CIN0-7 inputs of the same DFP to recirculate the coefficients, or they may be tied to the CIN0-7 inputs of another DFP to cascade DFPs for longer filter lengths.
TCCO The TCCO three-state output determines the number system representation of the coefficients output on COUT0-7. It tracks the TCCI signal to this same DFP. It should be tied to the TCCI input of the next DFP in a cascade of DFPs for increased filter lengths. This signal is enabled by $\overline{\text{COENB}}$ low.
 $\overline{\text{COENB}}$ A low on the $\overline{\text{COENB}}$ input enables the COUT0-7 outputs and the TCCO output. A high on this input places all these outputs in their high impedance state.
DCM0-1 These two inputs determine the use of the internal decimation registers as follows:

| DCM1 | DCM0 | Decimation Function |
|------|------|-------------------------------------|
| 0 | 0 | Decimation registers not used |
| 0 | 1 | One decimation register is used |
| 1 | 0 | Two decimation registers are used |
| 1 | 1 | Three decimation registers are used |

 The coefficients pass from cell to cell at a rate determined by the number of decimation registers used. When no decimation registers are used, coefficients move from cell to cell on each clock. When one decimation register is used, coefficients move from cell to cell on every other clock, etc. These signals are latched and delayed by one clock internal to the DFP.

SUM0-24 These 25 three-state outputs are used to output the results of the internal filter cell computations. Individual filter cell results or the result of the shift-and-add output stage can be output. If an individual filter cell result is to be output, the ADR0-2 signals select the filter cell result. The SHADD signal determines whether the selected filter cell result or the output stage adder result is output. The signals $\overline{\text{SENBH}}$ and $\overline{\text{SENBL}}$ enable the most significant and least significant bits of the SUM0-24 result respectively. Both $\overline{\text{SENBH}}$ and $\overline{\text{SENBL}}$ may be enabled simultaneously if the system has a 25-bit or larger bus. However individual enables are provided to facilitate use with a 16-bit bus.

$\overline{\text{SENBH}}$ A low on this input enables result bits SUM16-24. A high on this input places these bits in their high impedance state.

$\overline{\text{SENBL}}$ A low on this input enables result bits SUM0-15. A high on this input places these bits in their high impedance state.

ADR0-2 These three inputs select the one cell whose accumulator will be read through the output bus (SUM0-24) or added to the output stage accumulator. They also determine which accumulator will be cleared when $\overline{\text{ERASE}}$ is low. These

inputs are latched in the DFP and delayed by one clock internal to the DFP. If the ADR0-2 lines remain at the same address for more than one clock, the output at SUM0-24 will not change to reflect any subsequent accumulator updates in the addressed cell. Only the result available during the first clock, when ADR0-2 selects the cell will be output. This does not hinder normal operation since the ADR0-2 lines are changed sequentially. This feature facilitates the interface with slow memories where the output is required to be fixed for more than one clock.

SHADD The SHADD input controls the activation of the shift and add operation in the output stage. This signal is latched in the DFP and delayed by one clock internal to the DFP. Detailed explanation is given in the DFP Output Stage section.

$\overline{\text{RESET}}$ A low on this input synchronously clears all the internal registers, except the cell accumulators. It can be used with $\overline{\text{ERASE}}$ to also clear all the accumulators simultaneously. This signal is latched in the DFP and delayed by one clock internal to the DFP.

$\overline{\text{ERASE}}$ A low on this input synchronously clears the cell accumulator selected by the ADR0-2 signals. If $\overline{\text{RESET}}$ is also low simultaneously, all cell accumulators are cleared.

FUNCTIONAL DESCRIPTION

The Digital Filter Processor (DFP) is composed of eight filter cells cascaded together and an output stage for combining or selecting filter cell outputs (Figure 1). Each filter cell contains

a multiplier-accumulator and several registers (Figure 2). Each 8-bit coefficient is multiplied by a 8-bit data sample, with the result added to the 26-bit accumulator contents. The coefficient output of each cell is cascaded to the coefficient input of the next cell to its right.

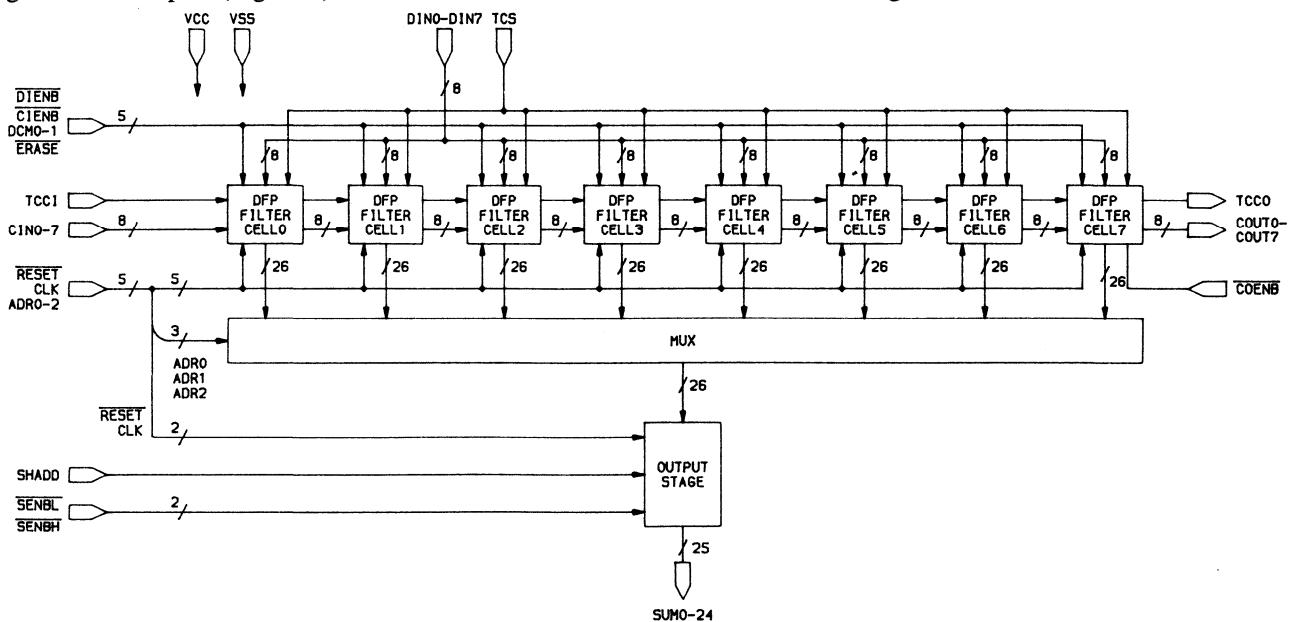


FIGURE 1. ZR33881 DFP BLOCK DIAGRAM.

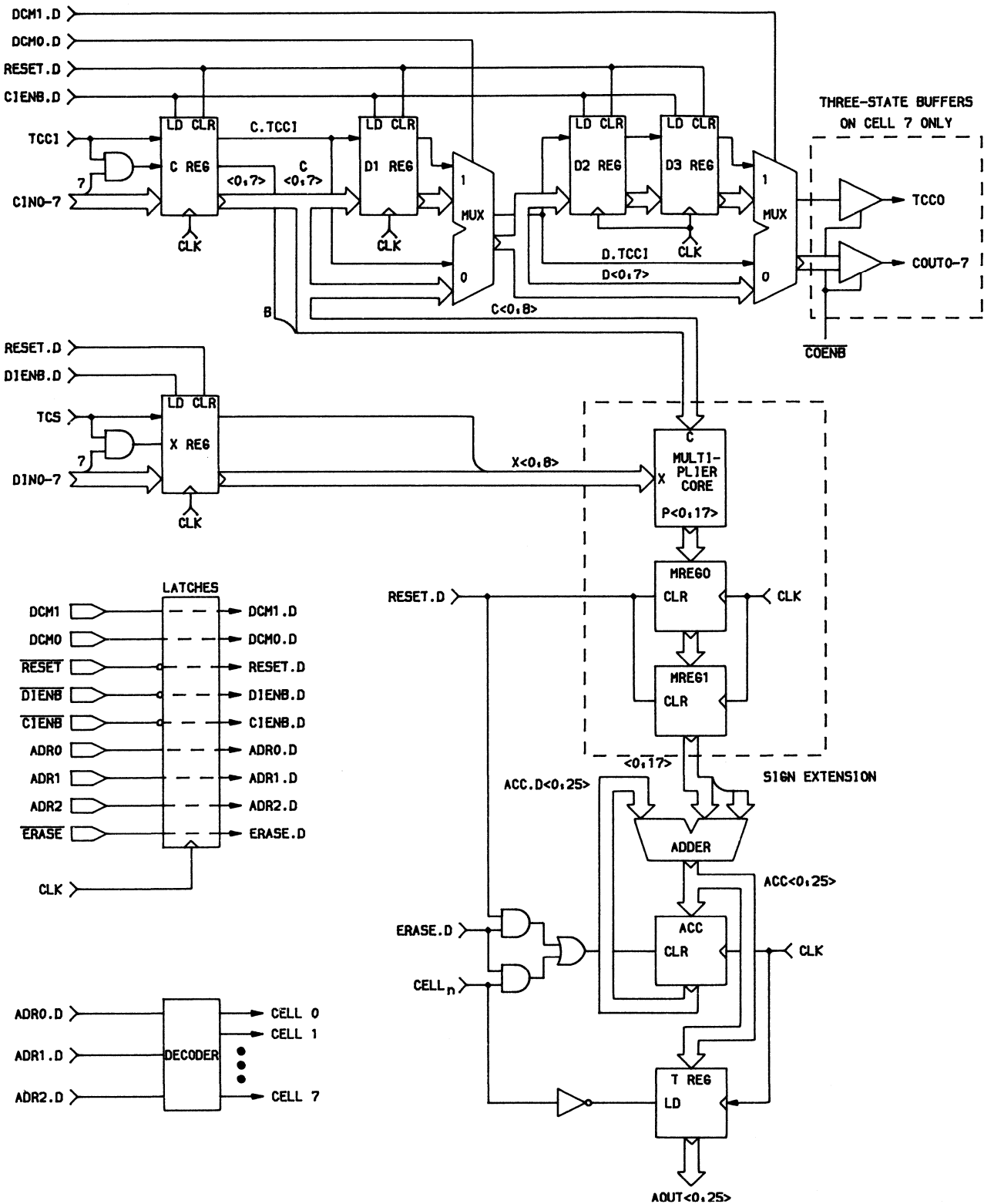


FIGURE 2. ZR33881 DFP FILTER CELL.

DFP FILTER CELL

A 8-bit coefficient with sign control(CIN0–7, TCCI) enters each cell through the C register on the left and exits the cell on the right as signals COUT0–7 and TCCO. The coefficients may move directly from the C register to the output, exiting the cell on the clock following its entrance. When decimation is selected the coefficient exit is delayed by 1, 2 or 3 clocks by passing through one or more decimation registers (D1, D2 or D3).

The combination of D registers through which the coefficient passes is determined by the state of DCM0 and DCM1. The output signals (COUT0–7, TCCO) are connected to the CIN0–7 and TCCI of the next cell to its right. The $\overline{\text{COENB}}$ input signal enables the COUT0–7 and TCCO outputs of the right-most cell to the COUT0–7 and TCCO pins of the DFP.

The C and D registers are enabled for loading by $\overline{\text{CIENB}}$. Loading is synchronous with CLK when $\overline{\text{CIENB}}$ is low. Note that $\overline{\text{CIENB}}$ is latched internally. It enables the register for loading after the next CLK following the onset of $\overline{\text{CIENB}}$ low. Actual loading occurs on the second CLK following the onset of $\overline{\text{CIENB}}$ low. Therefore $\overline{\text{CIENB}}$ must be low during the clock cycle immediately preceding presentation of the coefficient on the CIN0–7 inputs. In most basic FIR operations, $\overline{\text{CIENB}}$ will be low throughout the process, so this latching and delay sequence is only important during the initialization phase. When $\overline{\text{CIENB}}$ is high, the coefficients are frozen.

These registers are cleared synchronously under control of $\overline{\text{RESET}}$, which is latched and delayed exactly like $\overline{\text{CIENB}}$.

The output of the C register (C<0:8>) is one input of the 8×8 multiplier.

The other input to the 8×8 multiplier comes from the output of the X register. This register is loaded with a data sample from the DFP input signals DIN0–7 discussed above. The X register is enabled for loading by $\overline{\text{DIENB}}$. Loading is synchronous with CLK when $\overline{\text{DIENB}}$ is low. Note that $\overline{\text{DIENB}}$ is latched internally. It enables the register for loading after the next CLK following the onset of $\overline{\text{DIENB}}$ low. Actual loading occurs on the second CLK following the onset of $\overline{\text{DIENB}}$ low. Therefore $\overline{\text{DIENB}}$ must be low during the clock cycle immediately preceding presentation of the data sample on the DIN0-7 inputs. In most basic FIR operations, $\overline{\text{DIENB}}$ will be low throughout the process, so this latching and delay sequence is only important during the initialization phase. When $\overline{\text{DIENB}}$ is high, the X register is loaded with all zeros.

The multiplier is pipelined and is modeled as a multiplier core followed by two pipeline registers, MREG0 and MREG1 (Figure 2). The multiplier output is sign extended and input as one operand of the 26-bit adder. The other adder operand is the output of the 26-bit accumulator. The adder output is loaded synchronously into both the accumulator and the TREG.

The TREG loading is disabled by the cell select signal, CELL_n , where n is the cell number. The cell select is decoded from the ADR0D-ADR2D signals to generate the TREG load enable. The cell select is inverted and applied as the load enable to the TREG. Operation is such that the TREG is loaded whenever the cell is not selected. Therefore, TREG is loaded every clock, except the clock following cell selection. The purpose of the TREG is to hold the result of a sum-of-products calculation during the clock when the accumulator is cleared to prepare for the next sum-of-products calculation. This allows continuous accumulation without wasting clocks.

The accumulator is loaded with the adder output every clock unless it is cleared. It is cleared synchronously in two ways. When $\overline{\text{RESET}}$ and $\overline{\text{ERASE}}$ are both low, the accumulator is cleared along with all other registers on the DFP. Since $\overline{\text{ERASE}}$ and $\overline{\text{RESET}}$ are latched and delayed one clock internally, clearing occurs on the second CLK following the onset of both $\overline{\text{ERASE}}$ and $\overline{\text{RESET}}$ low.

The second accumulator clearing mechanism clears a single accumulator in a selected cell. The cell select signal, CELL_n , decoded from ADR0-2 and the $\overline{\text{ERASE-D}}$ signal enable clearing of the accumulator on the next CLK.

The $\overline{\text{ERASE}}$ and $\overline{\text{RESET}}$ signals clear the DFP internal registers and states as follows:

| $\overline{\text{ERASE}}$ | $\overline{\text{RESET}}$ | CLEARING EFFECT |
|---------------------------|---------------------------|---|
| 1 | 1 | No clearing occurs, internal state remains same |
| 1 | 0 | Reset only active, all registers except accumulators are cleared, including the internal pipeline registers. |
| 0 | 1 | Erase only active, the accumulator whose address is given by the ADR0–2 inputs is cleared. |
| 0 | 0 | Both $\overline{\text{RESET}}$ and $\overline{\text{ERASE}}$ active, all accumulators as well as all other registers are cleared. |

DFP OUTPUT STAGE

The output stage consists of a 26-bit adder, 26-bit register, feed-back multiplexer from the register to the adder, an output multiplexer and a 25-bit three-state driver stage (Figure 3).

The 26-bit output adder can add any filter cell accumulator result to the 18 most significant bits of the output buffer. This result is stored back in the output buffer. This operation takes place in one clock period. The eight LSBs are lost. The filter cell accumulator is selected by the ADR0-2 inputs.

The 18 MSBs of the output buffer actually pass through the zero mux on their way to the output adder input. The zero mux is controlled by the SHADD input signal and selects either the output buffer 18 MSBs or all zeros for the adder input. A low on the SHADD input selects zero. A high on the SHADD input selects the output buffer MSBs, thus activating the shift-and-add operation. The SHADD signal is latched and delayed by one clock internally.

The 25 least significant bits (LSBs) from either a cell accumulator or the output buffer are output on the SUM0-24 bus. The output mux determines whether the cell accumulator selected by ADR0-2 or the output buffer is output to the bus.

This mux is controlled by the SHADD input signal. Control is based on the state of the SHADD during two successive clocks; in other words, the output mux selection contains memory. If SHADD is low during a clock cycle and was low during the previous clock, the output mux selects the contents of the filter cell accumulator addressed by ADR0-2. Otherwise the output mux selects the contents of the output buffer.

If the ADR0-2 lines remain at the same address for more than one clock, the output at SUM0-24 will not change to reflect any subsequent accumulator updates in the addressed cell. Only the result available during the first clock when ADR0-2 selects the cell will be output. This does not hinder normal FIR operation since the ADR0-2 lines are changed sequentially. This feature facilitates the interface with slow memories where the output is required to be fixed for more than one clock.

The SUM0-24 output bus is controlled by the $\overline{\text{SENBH}}$ and $\overline{\text{SENBL}}$ signals. A low on $\overline{\text{SENBL}}$ enables bits SUM0-15. A low on $\overline{\text{SENBH}}$ enables bits SUM16-24. Thus all 25 bits can be output simultaneously if the external system has a 25-bit or larger bus. If the external system bus is only 16 bits, the bits can be enabled in two groups of 16 and 9 bits (sign extended).

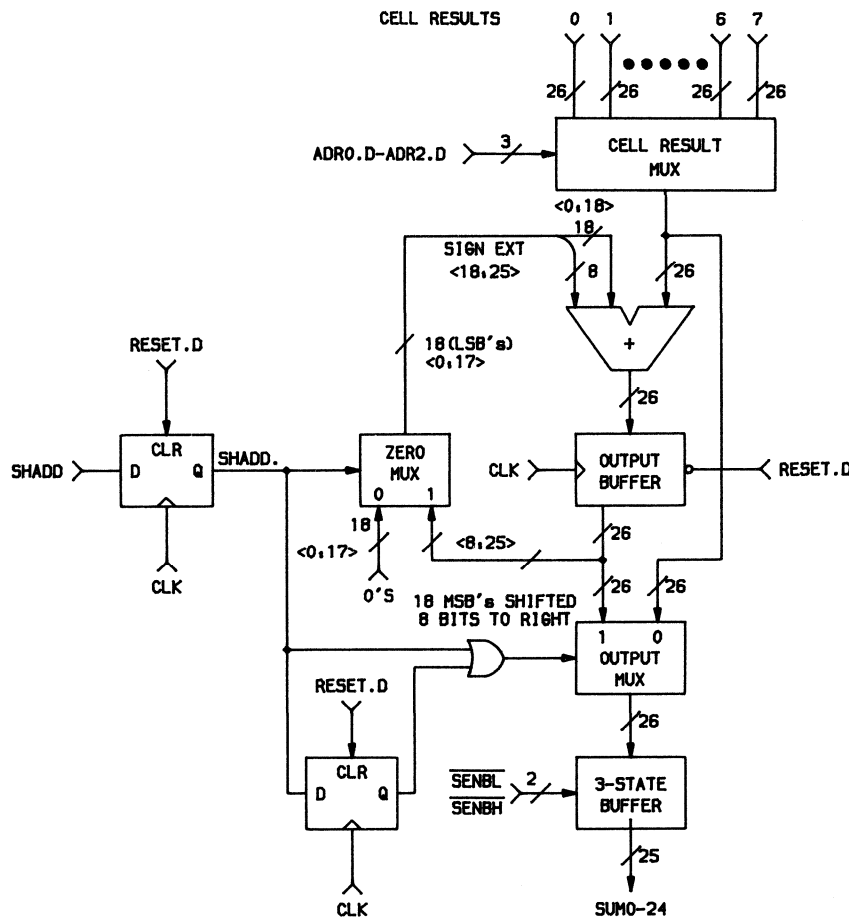


FIGURE 3. ZR33881 DFP OUTPUT STAGE.

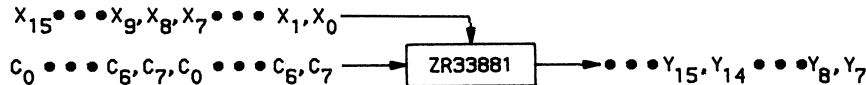
DFP ARITHMETIC

Both data samples and coefficients can be represented as either unsigned or two's complement numbers. The TCS and TCCI input signals determine the type of arithmetic representation. Internally all values are represented by a 9-bit two's complement number. The value of the additional ninth bit depends on arithmetic representation selected. For two's complement arithmetic, the sign is extended into the ninth bit. For unsigned arithmetic, bit 9 is 0.

The multiplier output is 18 bits and the accumulator is 26 bits. The accumulator width determines the maximum possible number of terms in the sum-of-products without overflow. The maximum number of terms depends also on the number system and the distribution of the coefficient and data values. As a worst case assume the coefficients and data samples are always at their maximum absolute values. Then the maximum numbers of terms in the sum products are:

| Number System | Maximum Number of Terms |
|--|-------------------------|
| Two unsigned vectors | 516 |
| Two two's complement vectors | |
| • two positive vectors | 1040 |
| • two negative vectors | 1024 |
| • one positive & one negative vector | 1032 |
| One unsigned and one two's complement vector | |
| • positive two's complement vector | 518 |
| • negative two's complement vector | 514 |

For practical FIR filters, the coefficients are never all near maximum value, so even larger vectors are possible in practice.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | CELL4 | CELL5 | CELL6 | CELL7 | SUM/CLR |
|-----|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------|
| 0 | $C_7 \cdot X_0$ | 0 | 0 | 0 | | | | | — |
| 1 | $+ C_6 \cdot X_1$ | $C_7 \cdot X_1$ | 0 | 0 | | | | | — |
| 2 | $+ C_5 \cdot X_2$ | $+ C_6 \cdot X_2$ | $C_7 \cdot X_2$ | 0 | | | | | — |
| 3 | $+ C_4 \cdot X_3$ | $+ C_5 \cdot X_3$ | $+ C_6 \cdot X_3$ | $C_7 \cdot X_3$ | | | | | — |
| 4 | $+ C_3 \cdot X_4$ | $+ C_4 \cdot X_4$ | $+ C_5 \cdot X_4$ | $+ C_6 \cdot X_4$ | $C_7 \cdot X_4$ | | | | — |
| 5 | $+ C_2 \cdot X_5$ | $+ C_3 \cdot X_5$ | $+ C_4 \cdot X_5$ | $+ C_5 \cdot X_5$ | $+ C_6 \cdot X_5$ | $C_7 \cdot X_5$ | | | — |
| 6 | $+ C_1 \cdot X_6$ | $+ C_2 \cdot X_6$ | $+ C_3 \cdot X_6$ | $+ C_4 \cdot X_6$ | $+ C_5 \cdot X_6$ | $+ C_6 \cdot X_6$ | $C_7 \cdot X_6$ | | — |
| 7 | $+ C_0 \cdot X_7$ | $+ C_1 \cdot X_7$ | $+ C_2 \cdot X_7$ | $+ C_3 \cdot X_7$ | $+ C_4 \cdot X_7$ | $+ C_5 \cdot X_7$ | $+ C_6 \cdot X_7$ | $C_7 \cdot X_7$ | CELL0(Y7) |
| 8 | $C_7 \cdot X_8$ | $+ C_0 \cdot X_8$ | $+ C_1 \cdot X_8$ | $+ C_2 \cdot X_8$ | $+ C_3 \cdot X_8$ | $+ C_4 \cdot X_8$ | $+ C_5 \cdot X_8$ | $+ C_6 \cdot X_8$ | CELL1(Y8) |
| 9 | $+ C_6 \cdot X_9$ | $C_7 \cdot X_9$ | $+ C_0 \cdot X_9$ | $+ C_1 \cdot X_9$ | $+ C_2 \cdot X_9$ | $+ C_3 \cdot X_9$ | $+ C_4 \cdot X_9$ | $+ C_5 \cdot X_9$ | CELL2(Y9) |
| 10 | $+ C_5 \cdot X_{10}$ | $+ C_6 \cdot X_{10}$ | $C_7 \cdot X_{10}$ | $+ C_0 \cdot X_{10}$ | $+ C_1 \cdot X_{10}$ | $+ C_2 \cdot X_{10}$ | $+ C_3 \cdot X_{10}$ | $+ C_4 \cdot X_{10}$ | CELL3(Y10) |
| 11 | $+ C_4 \cdot X_{11}$ | $+ C_5 \cdot X_{11}$ | $+ C_6 \cdot X_{11}$ | $C_7 \cdot X_{11}$ | $+ C_0 \cdot X_{11}$ | $+ C_1 \cdot X_{11}$ | $+ C_2 \cdot X_{11}$ | $+ C_3 \cdot X_{11}$ | CELL4(Y11) |
| 12 | $+ C_3 \cdot X_{12}$ | $+ C_4 \cdot X_{12}$ | $+ C_5 \cdot X_{12}$ | $+ C_6 \cdot X_{12}$ | $C_7 \cdot X_{12}$ | $+ C_0 \cdot X_{12}$ | $+ C_1 \cdot X_{12}$ | $+ C_2 \cdot X_{12}$ | CELL5(Y12) |
| 13 | $+ C_2 \cdot X_{13}$ | $+ C_3 \cdot X_{13}$ | $+ C_4 \cdot X_{13}$ | $+ C_5 \cdot X_{13}$ | $+ C_6 \cdot X_{13}$ | $C_7 \cdot X_{13}$ | $+ C_0 \cdot X_{13}$ | $+ C_1 \cdot X_{13}$ | CELL6(Y13) |
| 14 | $+ C_1 \cdot X_{14}$ | $+ C_2 \cdot X_{14}$ | $+ C_3 \cdot X_{14}$ | $+ C_4 \cdot X_{14}$ | $+ C_5 \cdot X_{14}$ | $+ C_6 \cdot X_{14}$ | $+ C_7 \cdot X_{14}$ | $+ C_0 \cdot X_{14}$ | CELL7(Y14) |
| 15 | $+ C_0 \cdot X_{15}$ | $+ C_1 \cdot X_{15}$ | $+ C_2 \cdot X_{15}$ | $+ C_3 \cdot X_{15}$ | $+ C_4 \cdot X_{15}$ | $+ C_5 \cdot X_{15}$ | $+ C_6 \cdot X_{15}$ | $C_7 \cdot X_{15}$ | CELL0(Y15) |

TABLE 1. ZR33881 20 MHz, 8-TAP FIR FILTER SEQUENCE.

BASIC FIR OPERATION

A simple, 15 MHz 8-tap FIR filter example serves to illustrate more clearly the operation of the DFP. The sequence table (Table 1) shows the results of the multiply accumulate in each cell after each clock. The coefficient sequence, C_n , enters the DFP on the left and moves from left to right through the cells. The data sample sequence, X_n , enters the DFP from the top, with each cell receiving the same sample simultaneously. Each cell accumulates the sum-of-products for one output point. Eight sums-of-products are calculated simultaneously, but staggered in time so that a new output is available every system clock. Detailed operation of the DFP to perform a basic 8-tap, 8-bit coefficient, 8-bit data, 15 MHz FIR filter is best understood by observing the schematic (Figure 4) and timing diagram (Figure 5). The internal pipeline length of the DFP is four (4) clock cycles, corresponding to the register levels CREG (or XREG), MREG0, MREG1, and TREG (Figures 2 and 3). Therefore the delay from presentation of data and coefficients at the DIN0-7 and CIN0-7 inputs to a sum appearing at the SUM0-24 output is:

$$k + T_d$$

where

$$k = \text{filter length}$$

$$T_d = 4, \text{ the internal pipeline delay of DFP}$$

After the pipeline has filled, a new output sample is available every clock. The delay to last sample output from last sample input is T_d .

The output sums, Y_n , shown in the timing diagram are derived from the sum-of-products equation:

$$Y(n) = C(0) \times X(n) + C(1) \times X(n-1) + C(2) \times X(n-2) + C(3) \times X(n-3) + C(4) \times X(n-4) + C(5) \times X(n-5) + C(6) \times X(n-6) + C(7) \times X(n-7)$$

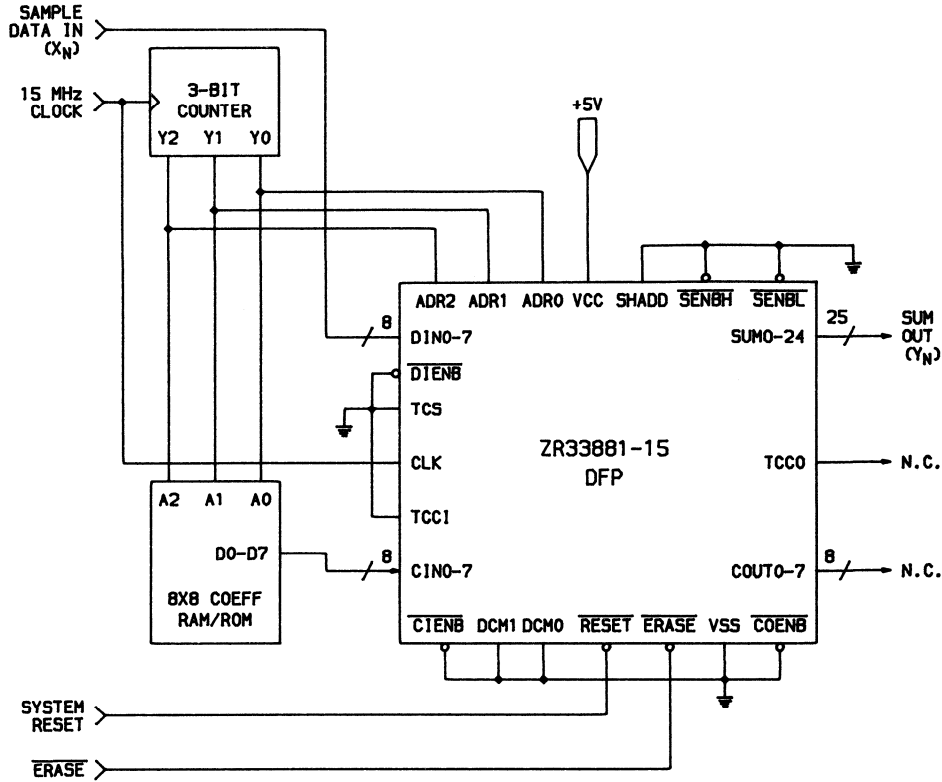
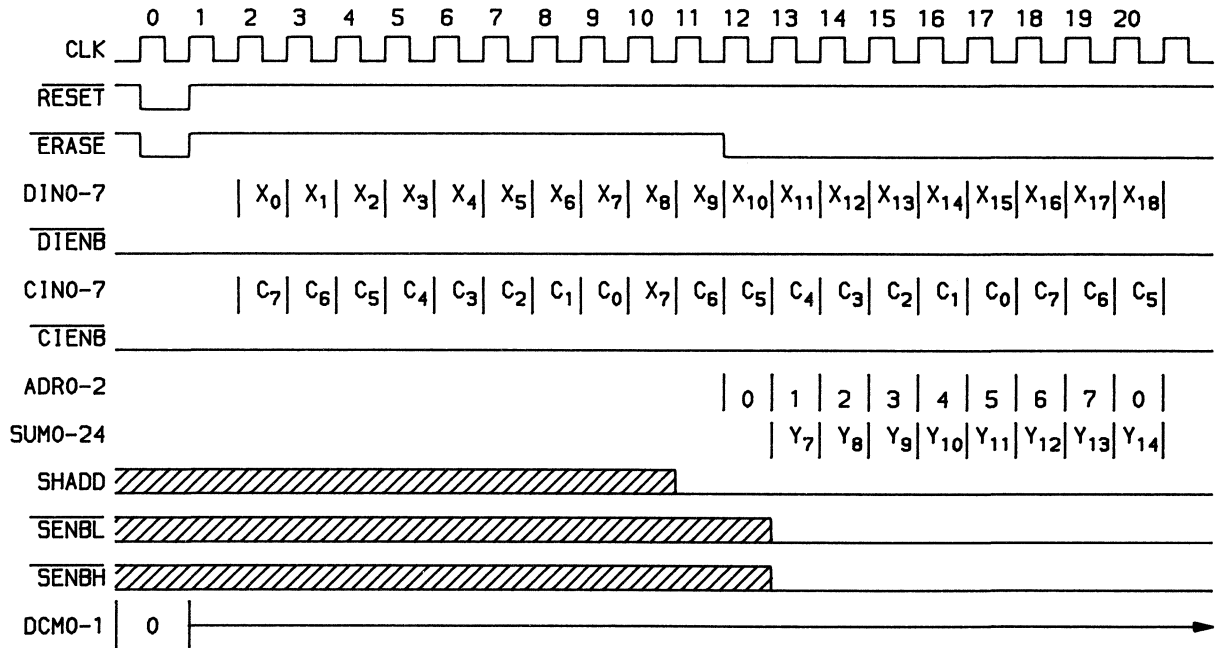


FIGURE 4. ZR33881 20 MHz, 8-TAP FIR FILTER APPLICATION SCHEMATIC.



$$Y_N = \sum_{K=0}^7 C_K \cdot X_{N-K}$$

FIGURE 5. ZR33881 20 MHz, 8-TAP FIR FILTER TIMING.

Note: SCHEMATIC AND TIMING SHOWN IN FIGURES 4, 5 MAY PRODUCE TRANSIENTS UPON STARTUP. TRANSIENTS MAY BE CONTROLLED BY SLIGHTLY MORE COMPLEX SEQUENCING LOGIC.

EXTENDED FIR FILTER LENGTH

Filter lengths greater than eight taps can be created by either cascading together multiple DFPs or "reusing" a single DFP. Using multiple DFPs, an FIR filter of up to 514 taps can be constructed to operate at a 15 MHz sample rate. Using a single DFP clocked at 15 MHz, an FIR filter of up to 514 taps can be constructed to operate at less than a 15 MHz sample rate. Combinations of these two techniques are also possible.

CASCADE CONFIGURATION

To design a filter length $L > 8$, $L/8$ DFPs are cascaded by connecting the COUT0-7 outputs of the (i)th DFP to the CIN0-7 inputs of the (i+1)th DFP. The DIN0-7 inputs and SUM0-24 outputs of all the DFPs are also tied together. A specific example of two cascaded DFPs illustrates the technique (Figure 6). Timing (Figure 7) is similar to the simple 8-tap FIR, except the ERASE and SENBL/SENBH signals must be enabled independently for the two DFPs in order to clear the correct accumulators and enable the SUM0-24 output signals at the proper times.

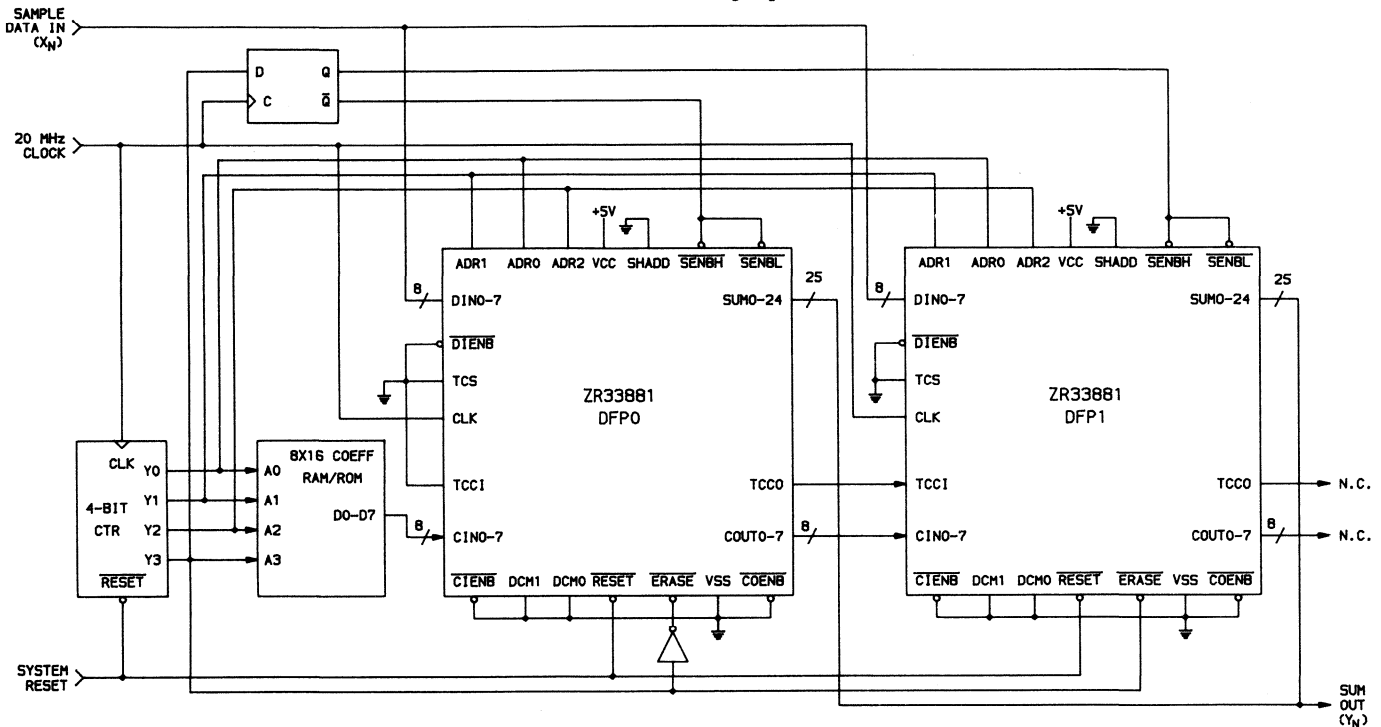
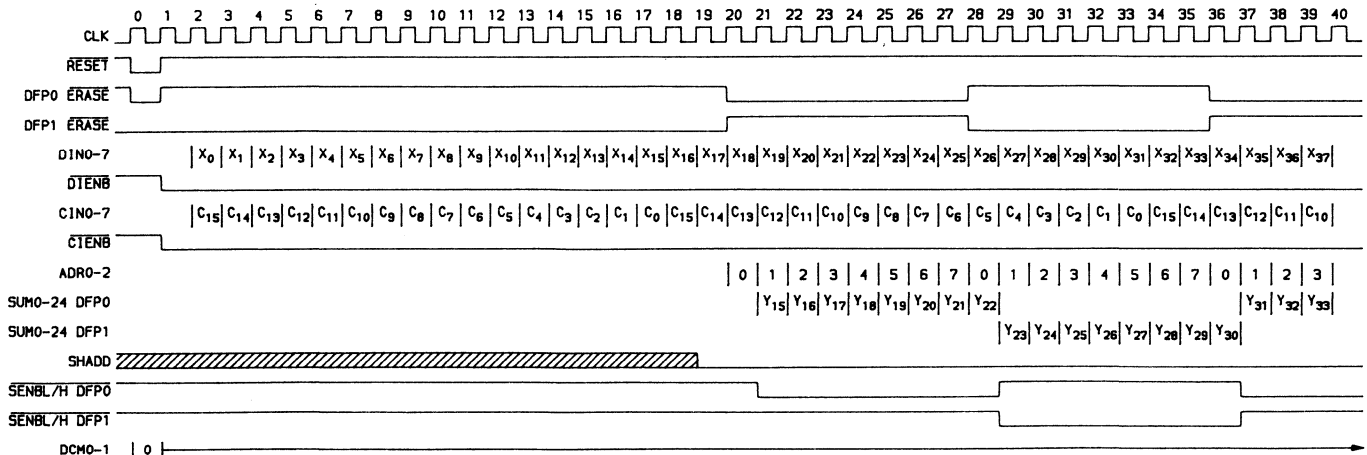


FIGURE 6. ZR 33881 20 MHz, 16-TAP FIR FILTER CASCADE APPLICATION SCHEMATIC.



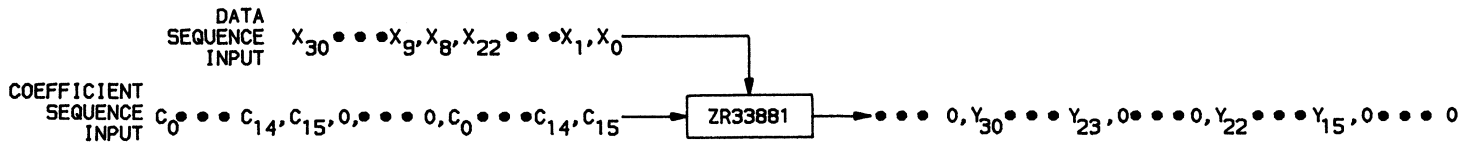
$$Y_N = \sum_{K=0}^{15} C_K \cdot X_{N-K}$$

FIGURE 7. ZR33881 16-TAP 20 MHz FIR FILTER TIMING USING TWO CASCADED ZR33881s.

SINGLE-PROCESSOR CONFIGURATION

Using a single DFP, a filter of length $L > 8$ can be constructed by processing in $L/8$ passes as illustrated in the table below (Table 2) for a 16-tap FIR. Each pass is composed of $T_p =$

$7 + L$ cycles and computes eight output samples. In pass i , the sample with indices $i*8$ to $i*8 + (L-1)$ enter the DIN0-7 inputs. The coefficients $C_0 - C_{L-1}$ enter the CIN0-7 inputs, followed by seven zeros. As these zeros are entered, the result samples are output and the accumulators reset. Initial filling of the pipeline is not shown in this sequence table. Filter outputs can be put through a FIFO to even out the sample rate.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | CELL4 | CELL5 | CELL6 | CELL7 | SUM/CLR |
|-----|-------------------------|--------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-------------------------|------------|
| 6 | $C_{15} \cdot X_0$ | 0 | 0 | 0 | | | | | — |
| 7 | $+ C_{14} \cdot X_1$ | $C_{15} \cdot X_1$ | 0 | 0 | | | | | — |
| 8 | $+ C_{13} \cdot X_2$ | | $C_{15} \cdot X_2$ | 0 | | | | | — |
| 9 | $+ C_{12} \cdot X_3$ | | | $C_{15} \cdot X_3$ | | | | | — |
| 10 | $+ C_{11} \cdot X_4$ | | | $+ C_{14} \cdot X_4$ | $C_{15} \cdot X_4$ | | | | — |
| 11 | $+ C_{10} \cdot X_5$ | | | $+ C_{13} \cdot X_5$ | | $C_{15} \cdot X_5$ | | | — |
| 12 | $+ C_9 \cdot X_6$ | | | $+ C_{12} \cdot X_6$ | | | $C_{15} \cdot X_6$ | | — |
| 13 | $+ C_8 \cdot X_7$ | | | $+ C_{11} \cdot X_7$ | | | | $C_{15} \cdot X_7$ | — |
| 14 | $+ C_7 \cdot X_8$ | | | $+ C_{10} \cdot X_8$ | | | | $+ C_{14} \cdot X_8$ | — |
| 15 | $+ C_6 \cdot X_9$ | | | $+ C_9 \cdot X_9$ | | | | $+ C_{13} \cdot X_9$ | — |
| 16 | $+ C_5 \cdot X_{10}$ | | | $+ C_8 \cdot X_{10}$ | | | | $+ C_{12} \cdot X_{10}$ | — |
| 17 | $+ C_4 \cdot X_{11}$ | | | $+ C_7 \cdot X_{11}$ | | | | $+ C_{11} \cdot X_{11}$ | — |
| 18 | $+ C_3 \cdot X_{12}$ | | | $+ C_6 \cdot X_{12}$ | | | | $+ C_{10} \cdot X_{12}$ | — |
| 19 | $+ C_2 \cdot X_{13}$ | | | $+ C_5 \cdot X_{13}$ | | | | $+ C_9 \cdot X_{13}$ | — |
| 20 | $+ C_1 \cdot X_{14}$ | | | $+ C_4 \cdot X_{14}$ | | | | $+ C_8 \cdot X_{14}$ | — |
| 21 | $+ C_0 \cdot X_{15}$ | | | $+ C_3 \cdot X_{15}$ | | | | $+ C_7 \cdot X_{15}$ | CELL0(Y15) |
| 22 | 0 | $C_0 \cdot X_{16}$ | | $+ C_2 \cdot X_{16}$ | | | | $+ C_6 \cdot X_{16}$ | CELL1(Y16) |
| 23 | 0 | 0 | $C_0 \cdot X_{17}$ | $+ C_1 \cdot X_{17}$ | | | | $+ C_5 \cdot X_{17}$ | CELL2(Y17) |
| 24 | 0 | 0 | 0 | $+ C_0 \cdot X_{18}$ | | | | $+ C_4 \cdot X_{18}$ | CELL3(Y18) |
| 25 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{19}$ | | | $+ C_3 \cdot X_{19}$ | CELL4(Y19) |
| 26 | 0 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{20}$ | | $+ C_2 \cdot X_{20}$ | CELL5(Y20) |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{21}$ | $+ C_1 \cdot X_{21}$ | CELL6(Y21) |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $+ C_0 \cdot X_{22}$ | CELL7(Y22) |
| 29 | $C_{15} \cdot X_8$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — |
| 30 | $+ C_{14} \cdot X_9$ | $C_{15} \cdot X_9$ | 0 | 0 | 0 | 0 | 0 | 0 | — |
| 31 | $+ C_{13} \cdot X_{10}$ | | $C_{15} \cdot X_{10}$ | 0 | 0 | 0 | 0 | 0 | — |
| 32 | $+ C_{12} \cdot X_{11}$ | | | $C_{15} \cdot X_{11}$ | 0 | 0 | 0 | 0 | — |
| 33 | $+ C_{11} \cdot X_{12}$ | | | | $C_{15} \cdot X_{12}$ | 0 | 0 | 0 | — |
| 34 | $+ C_{10} \cdot X_{13}$ | | | | | $C_{15} \cdot X_{13}$ | 0 | 0 | — |
| 35 | $+ C_9 \cdot X_{14}$ | | | | | | $C_{15} \cdot X_{14}$ | 0 | — |
| 36 | $+ C_8 \cdot X_{15}$ | | | | | | | $C_{15} \cdot X_{15}$ | — |
| 37 | $+ C_7 \cdot X_{16}$ | | | | | | | $+ C_{14} \cdot X_{16}$ | — |
| 38 | $+ C_6 \cdot X_{17}$ | | | | | | | $+ C_{13} \cdot X_{17}$ | — |
| 39 | $+ C_5 \cdot X_{18}$ | | | | | | | $+ C_{12} \cdot X_{18}$ | — |
| 40 | $+ C_4 \cdot X_{19}$ | | | | | | | $+ C_{11} \cdot X_{19}$ | — |
| 41 | $+ C_3 \cdot X_{20}$ | | | | | | | $+ C_{10} \cdot X_{20}$ | — |
| 42 | $+ C_2 \cdot X_{21}$ | | | | | | | $+ C_9 \cdot X_{21}$ | — |
| 43 | $+ C_1 \cdot X_{22}$ | | | | | | | $+ C_8 \cdot X_{22}$ | — |
| 44 | $+ C_0 \cdot X_{23}$ | | | | | | | $+ C_7 \cdot X_{23}$ | CELL0(Y23) |
| 45 | 0 | $C_0 \cdot X_{24}$ | | | | | | $+ C_6 \cdot X_{24}$ | CELL1(Y24) |
| 46 | 0 | 0 | $C_0 \cdot X_{25}$ | | | | | $+ C_5 \cdot X_{25}$ | CELL2(Y25) |
| 47 | 0 | 0 | 0 | $C_0 \cdot X_{26}$ | | | | $+ C_4 \cdot X_{26}$ | CELL3(Y26) |
| 48 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{27}$ | | | $+ C_3 \cdot X_{27}$ | CELL4(Y27) |

TABLE 2. ZR33881 5.2 MHz OUTPUT 16-TAP FIR FILTER SEQUENCE USING A SINGLE, 15 MHz DFP.

EXTENDED COEFFICIENT AND DATA SAMPLE WORD SIZE

The sample and coefficient word size can be extended by utilizing several DFPs in parallel to get the maximum sample rate or a single DFP with resulting lower sample rates. The technique is to compute partial products of 8×8 and combine these partial products by shifting and adding to obtain the final result. The shifting and adding can be accomplished with external adders (at full speed) or with the DFP's shift-and-add mechanism contained in its output stage (at reduced speed).

DECIMATION/RESAMPLING

The ZR33881 DFP provides a mechanism for decimating by factors of 2, 3 or 4. From the DFP filter cell block diagram (Figure 2), note the three D registers and two multiplexers in the coefficient path through the cell. These allow the coefficients to be delayed by 1, 2 or 3 clocks through the cell. The sequence table (Table 3) for a decimate-by-two filter illustrates the technique (internal cell pipelining ignored for simplicity).

Detailed timing for a 15 MHz input sample rate, 7.5 MHz output sample rate (i.e., decimate-by-two), 16-tap FIR filter, including pipelining, is shown in Figure 8.

OTHER DFP APPLICATIONS

The ZR33881 DFP is a versatile device with many applications beyond simple FIR filtering. The following list is a small sample of the many applications possible with the DFP. Implementation of these applications are discussed in greater detail in various ZORAN Application Notes.

- Higher precision arithmetic (e.g. 16×16)
 - Single DFP implementation
 - Multiple DFP implementation
- Higher bandwidth (30 MHz and up)
- Decimation/Interpolation
- 2-D FIR spatial Filtering/ Convolution/Correlation (e.g. 7×7 kernel convolution at 10 MHz rate)
- Matrix multiplication (8 elements \times 8 elements in $6 \mu\text{s}$) DFP. Implementations of these applications are discussed in greater detail in various ZORAN Application Notes.
- Complex multiply
- Adaptive filters
 - Echo cancellation
 - Adaptive equalization
- Butterfly computation
- Reverberation generators
- DFT
- Beam former
- Video Decoders

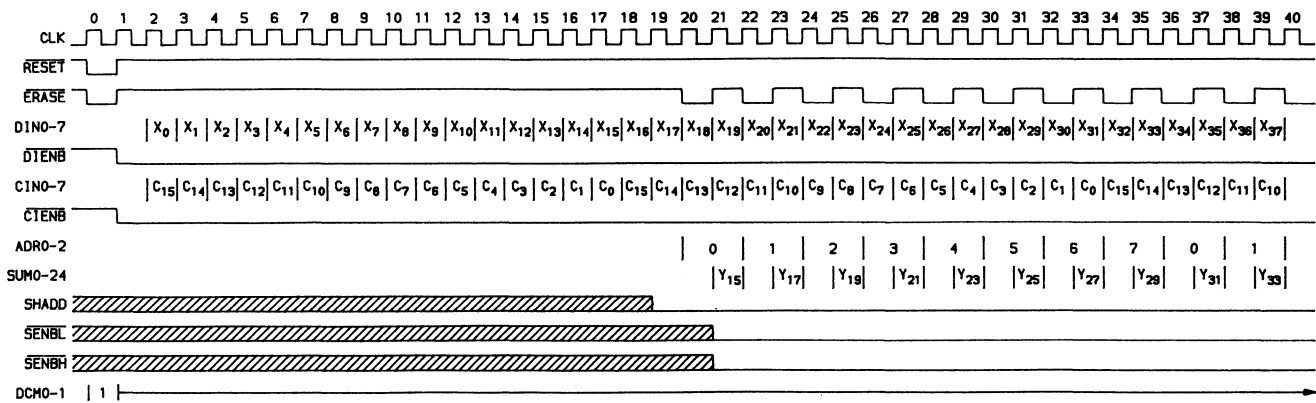
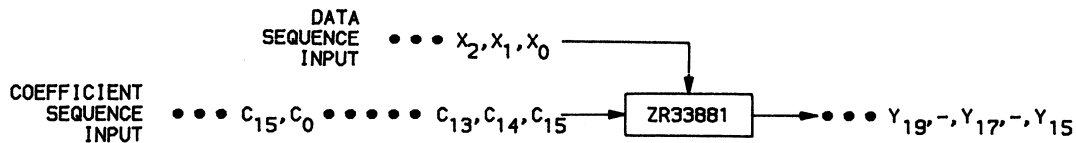


FIGURE 8. ZR33881 16-TAP DECIMATE-BY-TWO FIR FILTER TIMING, 20 MHz in 10 MHz OUT.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | CELL4 | CELL5 | CELL6 | CELL7 | SUM/CLR |
|-----|-------------------------|----------------------|----------------------|----------------------|----------------------|-------------------------|-------------------------|-------------------------|------------|
| 6 | $C_{15} \cdot X_0$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 7 | $+ C_{14} \cdot X_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 8 | $+ C_{13} \cdot X_2$ | $C_{15} \cdot X_2$ | 0 | 0 | 0 | 0 | 0 | 0 | |
| 9 | $+ C_{12} \cdot X_3$ | ↓ | 0 | 0 | 0 | 0 | 0 | 0 | |
| 10 | $+ C_{11} \cdot X_4$ | ↓ | $C_{15} \cdot X_4$ | 0 | 0 | 0 | 0 | 0 | |
| 11 | $+ C_{10} \cdot X_5$ | ↓ | ↓ | 0 | 0 | 0 | 0 | 0 | |
| 12 | $+ C_9 \cdot X_6$ | ↓ | ↓ | $C_{15} \cdot X_6$ | 0 | 0 | 0 | 0 | |
| 13 | $+ C_8 \cdot X_7$ | ↓ | ↓ | ↓ | 0 | 0 | 0 | 0 | |
| 14 | $+ C_7 \cdot X_8$ | ↓ | ↓ | ↓ | $C_{15} \cdot X_8$ | 0 | 0 | 0 | |
| 15 | $+ C_6 \cdot X_9$ | ↓ | ↓ | ↓ | ↓ | 0 | 0 | 0 | |
| 16 | $+ C_5 \cdot X_{10}$ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{10}$ | 0 | 0 | |
| 17 | $+ C_4 \cdot X_{11}$ | ↓ | ↓ | ↓ | ↓ | ↓ | 0 | 0 | |
| 18 | $+ C_3 \cdot X_{12}$ | ↓ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{12}$ | 0 | |
| 19 | $+ C_2 \cdot X_{13}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | 0 | |
| 20 | $+ C_1 \cdot X_{14}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{14}$ | CELL0(Y15) |
| 21 | $+ C_0 \cdot X_{15}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{14} \cdot X_{15}$ | — |
| 22 | $C_{15} \cdot X_{16}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{13} \cdot X_{16}$ | CELL1(Y17) |
| 23 | $+ C_{14} \cdot X_{17}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{12} \cdot X_{17}$ | — |
| 24 | $+ C_{13} \cdot X_{18}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{11} \cdot X_{18}$ | CELL2(Y19) |
| 25 | $+ C_{12} \cdot X_{19}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{10} \cdot X_{19}$ | — |
| 26 | $+ C_{11} \cdot X_{20}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_9 \cdot X_{20}$ | CELL3(Y21) |
| 27 | $+ C_{10} \cdot X_{21}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_8 \cdot X_{21}$ | — |
| 28 | $+ C_9 \cdot X_{22}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_7 \cdot X_{22}$ | CELL4(Y23) |
| 29 | $+ C_8 \cdot X_{23}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_6 \cdot X_{23}$ | — |
| 30 | $+ C_7 \cdot X_{24}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_5 \cdot X_{24}$ | CELL5(Y25) |
| 31 | $+ C_6 \cdot X_{25}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_4 \cdot X_{25}$ | — |
| 32 | $+ C_5 \cdot X_{26}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_3 \cdot X_{26}$ | CELL6(Y27) |
| 33 | $+ C_4 \cdot X_{27}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_2 \cdot X_{27}$ | — |
| 34 | $+ C_3 \cdot X_{28}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_1 \cdot X_{28}$ | CELL7(Y29) |
| 35 | $+ C_2 \cdot X_{29}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_0 \cdot X_{29}$ | — |
| 36 | $+ C_1 \cdot X_{30}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{30}$ | CELL8(Y31) |
| 37 | $+ C_0 \cdot X_{31}$ | $+ C_2 \cdot X_{31}$ | $+ C_4 \cdot X_{31}$ | $+ C_6 \cdot X_{31}$ | $+ C_8 \cdot X_{31}$ | $+ C_{10} \cdot X_{31}$ | $+ C_{12} \cdot X_{31}$ | $+ C_{14} \cdot X_{31}$ | |

TABLE 3. ZR33881 16-TAP DECIMATE-BY-TWO FIR FILTER SEQUENCE, 15 MHz IN 7.5 MHz OUT.

DFP TIMING PARAMETERS

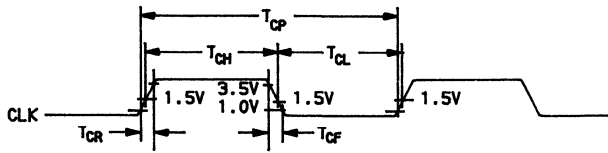
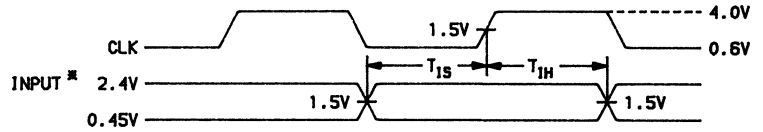
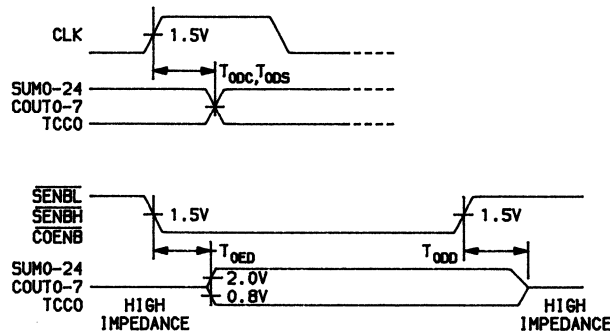


FIGURE 9. CLOCK AC PARAMETERS.



* INPUT INCLUDES D1NO-7, C1NO-7, DTENB, CTENB, ERASE, RESET, DCM0-1, ADRO-2, TCS, TCC1, SHADD

FIGURE 10. INPUT SETUP AND HOLD.



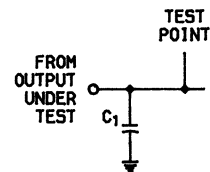
* SUMO-24, COUT0-7, TCCO ARE ASSUMED NOT TO BE IN HIGH-IMPEDANCE STATE

FIGURE 11. SUMO-24, COUT0-7, TCCO OUTPUT DELAYS.



A.C. TESTING, INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". INPUT AND OUTPUT TIMING MEASUREMENTS ARE MADE AT 1.5V FOR BOTH A LOGIC "1" AND "0"

FIGURE 12. A.C. TESTING INPUT, OUTPUT WAVEFORM.



NOTE: C₁ = 50pF INCLUDING STRAY AND WIRING CAPACITANCE

FIGURE 13. NORMAL A.C. TESTLOAD.

ABSOLUTE MAXIMUM RATINGS

(Above which useful life may be impaired)

Temperature under bias -55°C to +125°C
 Storage Temperature -65°C to +150°C
 Supply Voltage to Ground Potential
 Continuous -0.5V to +7.0V
 DC Voltage Applied to Outputs for High
 Output State -0.5V to +7.0V
 DC Input Voltage -0.5V to +7.0V
 DC Output Current, into Outputs 20mA/output
 (not to exceed 200 mA total)
 Latch up Trigger Current >200 mA

OPERATING RANGE

Commercial (C) Devices

Temperature 0°C ≤ T_A ≤ +70°C
 Supply Voltage 4.75V ≤ V_{CC} ≤ 5.25V

| DC CHARACTERISTICS OVER OPERATING RANGE unless otherwise specified | | | | | |
|--|------------------------------------|------|----------------------|-------|--|
| Symbol | Parameter | Min | Max | Units | Test Conditions |
| V _{IL} | Input Low Voltage | -0.5 | 0.8 | V | |
| V _{IH} | Input High Voltage | 2.2 | V _{CC} +0.5 | V | |
| V _{OL} | Output Low Voltage | | 0.45 | V | I _{OL} = 2 mA |
| V _{OH} | Output High Voltage | 2.4 | | V | I _{OH} = -400µA |
| I _{CC} | Power Supply Current | | 100 | mA | T _A = 0°C, V _{CC} = Max ¹ |
| I _{LI} | Input Leakage Current | | ± 10 | µA | 0 < V _{IN} < V _{CC} |
| I _{LO} | Output Leakage Current | | ± 10 | µA | 0.45 < V _{OUT} < V _{CC} |
| V _{CL} | Clock in Low Voltage | -0.5 | 0.6 | V | |
| V _{CH} | Clock in High Voltage | 4.0 | V _{CC} +0.5 | V | |
| C _{IN} | Input Capacitance | | 10 | pF | f _C = 1 MHz |
| C _{IO} | I/O, Clock, and Output Capacitance | | 10 | pF | f _C = 1 MHz |

Note 1: Using standard ZORAN test pattern at 20MHz.

Note 2: T_{ODC} refers to COUT0-7, TCCO. T_{ODC} is tested with a 50 pF load. Normal use is to cascade COUT0-7 of one device to CIN0-7 of another device. Actual load of CIN0-7 on COUT0-7 is 10-20 pF. (See Figure 14). Therefore, actual T_{ODC} in this case will be approximately 3 ns less.

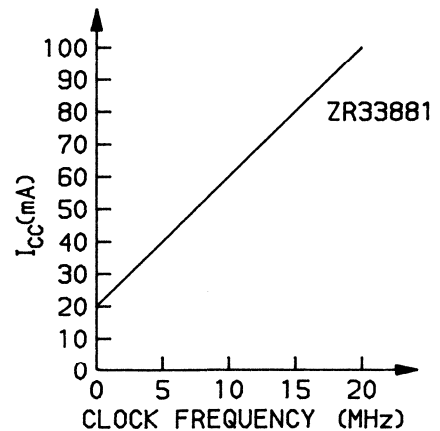
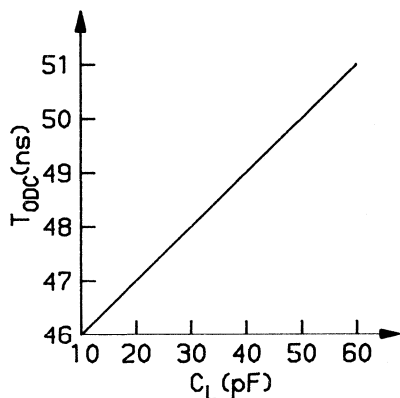
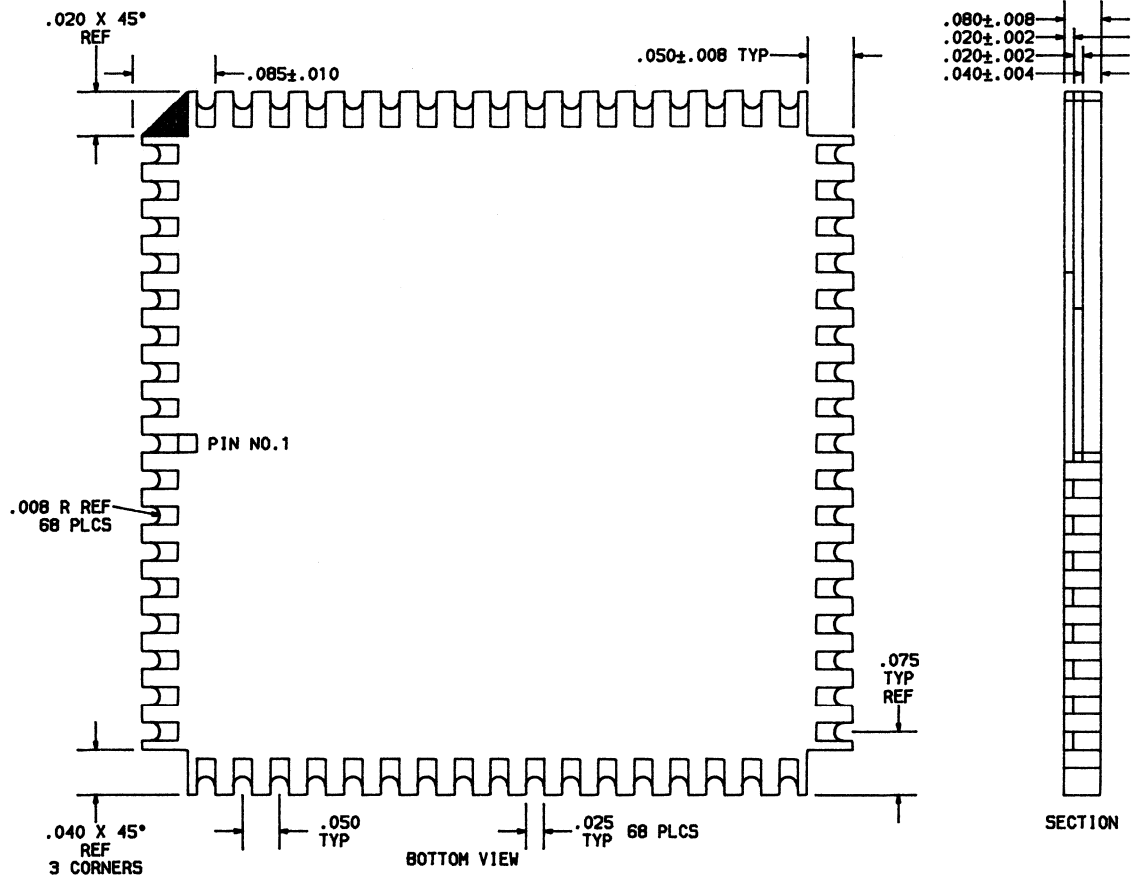


FIGURE 14. WORST CASE T_{ODC} AS A FUNCTION OF LOAD CAPACITANCE. FIGURE 15. TYPICAL I_{CC} VS FREQUENCY AT 5.0V V_{CC}, 25°C.

| AC CHARACTERISTICS OVER OPERATING RANGE ZR33881-15 | | | | | |
|--|---------------------------------|---------------------|------|-------|-----------------|
| Symbol | Parameter | COM ZR33881JC-15 | | Units | Test Conditions |
| | | Min | Max | | |
| T _{CP} | Clock Period | 67 | 5000 | ns | |
| T _{CL} | Clock Low | 30 | | ns | |
| T _{CH} | Clock High | 30 | | ns | |
| T _{CR} | Input Rise | | 5 | ns | 1.0V to 3.5V |
| T _{CF} | Input Fall | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 14 | | ns | |
| T _{IH} | Input Hold | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 50 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 25 | ns | |
| T _{ODD} | Output Disable Delay | | 25 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 50 | ns | |

| AC CHARACTERISTICS OVER OPERATING RANGE ZR33881-10 | | | | | |
|--|---------------------------------|---------------------|------|-------|-----------------|
| Symbol | Parameter | COM ZR33881JC-10 | | Units | Test Conditions |
| | | Min | Max | | |
| T _{CP} | Clock Period | 100 | 5000 | ns | |
| T _{CL} | Clock Low | 40 | | ns | |
| T _{CH} | Clock High | 40 | | ns | |
| T _{CR} | Input Rise | | 5 | ns | 1.0V to 3.5V |
| T _{CF} | Input Fall | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 30 | | ns | |
| T _{IH} | Input Hold | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 65 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 40 | ns | |
| T _{ODD} | Output Disable Delay | | 40 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 65 | ns | |



| ORDERING INFORMATION | | | |
|----------------------|--------------|---|--------------|
| Speed | Package Type | Temperature Range | Order Number |
| 10 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33881JC-10 |
| 15 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33881JC-15 |

DISTINCTIVE FEATURES

- 8 filter cells
- Up to 20 MHz sample rate
- 9-bit coefficients and signal data
- 26-bit accumulator per stage
- Filter lengths over 1000 taps
- Shift-and-add output stage for combining filter outputs
- Expandable coefficient size, data size and filter length
- Decimation by 2, 3 or 4
- CMOS power dissipation characteristics

APPLICATIONS

- 1-D and 2-D FIR filters
- Radar/Sonar
- Digital video and audio
- Adaptive filters
- Echo cancellation
- Correlation/convolution
- Complex multiply-add
- Butterfly computation
- Matrix multiplication
- Sample rate converters

DESCRIPTION

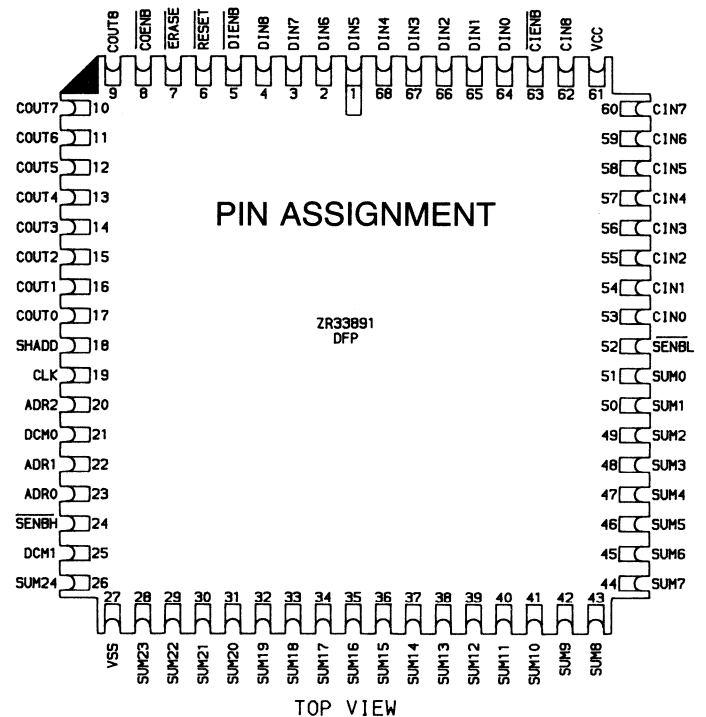
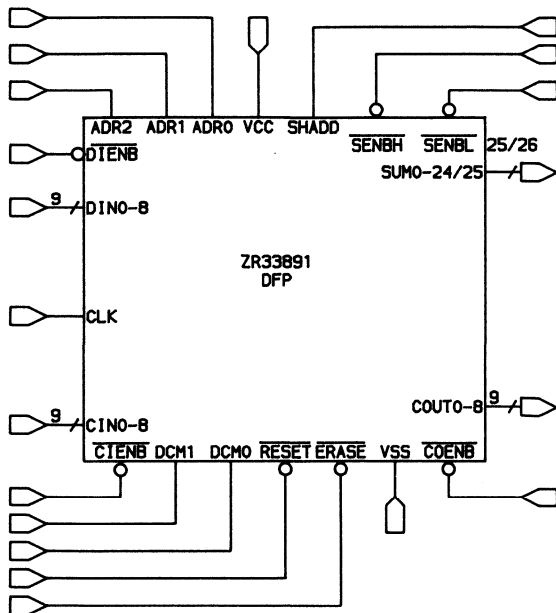
The ZR33891 is a video-speed Digital Filter Processor (DFP) designed to efficiently implement vector operations such as FIR digital filters. It is comprised of eight filter cells cascaded internally and a shift and add output stage, all in a single integrated circuit. Each filter cell contains a 9×9 two's complement bit multiplier, three decimation registers and a 26-bit accumulator. The output stage contains an additional 26-bit accumulator which can add the contents of any filter cell accumulator to the output stage accumulator shifted right by 8 bits. The ZR33891 has a maximum sample rate of 20 MHz. The effective multiply-accumulate (mac) rate is 160 MHz.

The ZR33891 DFP can be configured to process expanded coefficient and word sizes. Multiple DFPs can be cascaded for larger filter lengths without degrading the sample rate or a single

DFP can process larger filter lengths at less than 20 MHz with multiple passes. The architecture permits processing filter lengths of over 1000 taps with the guarantee of no overflows. In practice, most filter coefficients are less than 1.0, making even larger filter lengths possible. The DFP provides for 8-bit unsigned or 9-bit two's complement arithmetic, independently selectable for coefficients and signal data.

Each DFP filter cell contains three resampling or decimation registers which permit output sample rate reduction at rates of 1/2, 1/3 or 1/4 the input sample rate. These registers also provide the capability to perform 2-D operations such as matrix multiplication and $N \times N$ spatial correlations /convolutions for image processing applications.

ZR33891 LOGIC SYMBOL



INTERFACE SIGNAL DESCRIPTION

V_{cc} +5 power supply input

V_{ss} Power supply ground input.

CLK The CLK input provides the DFP system sample clock. The maximum clock frequency is 20 MHz. The minimum frequency is 200 KHz.

DIN0-8 These nine inputs are the data sample input bus. Nine-bit data samples are synchronously loaded through these pins to the X register of each filter cell of the DFP simultaneously. The \overline{DIENB} signal enables loading, which is synchronous on the rising edge of the clock signal.

The data samples can be either 9-bit two's complement or 8-bit unsigned values. For 9-bit two's complement values, DIN8 is the sign bit. For 8-bit unsigned values, DIN8 must be held at logical zero.

\overline{DIENB} A low on this input enables the data sample input bus (DIN0-8) to all the filter cells. A rising edge of the CLK signal occurring while \overline{DIENB} is low will load the X register of every filter cell with the 9-bit value present on DIN0-8. A high on this input forces all the bits of the data sample input bus to zero; a rising CLK edge when \overline{DIENB} is high will load the X register of every filter cell with all zeros. This signal is latched inside the device, delaying its effect by one clock internal to the device. Therefore it must be low during the clock cycle immediately preceding presentation of the desired data on the DIN0-8 inputs. Detailed operation is shown in later timing diagrams.

CIN0-8 These nine inputs are used to input the 9-bit coefficients. The coefficients are synchronously loaded into the C register of filter CELL0 if a rising edge of CLK occurs while \overline{CIENB} is low. The \overline{CIENB} signal is delayed by one clock as discussed below.

The coefficients can be either 9-bit two's complement or 8-bit unsigned values. For 9-bit two's complement values, DIN8 is the sign bit. For 8-bit unsigned values, DIN8 must be held at logical zero.

\overline{CIENB} A low on this input enables the C register of every filter cell and the D registers (decimation) of every filter cell according to the state of the DCM0-1 inputs. A rising edge of the CLK signal occurring while \overline{CIENB} is low will load the C register and appropriate D registers with the coefficient data present at their inputs. This provides the mechanism for shifting the coefficients from cell to cell through the device. A high on this input freezes the contents of the C register and the D registers, ignoring the CLK signal. This signal is latched and delayed by one clock internal to the DFP. Therefore it must be low during the clock cycle immediately preceding presentation of the desired coefficient on the CIN0-8 inputs. Detailed operation is shown in later timing diagrams.

COUT0-8 These nine three-state outputs are used to output the 9-bit coefficients from filter CELL7. These outputs are enabled by the \overline{COENB} signal low. These outputs may be tied to the CIN0-8 inputs of the same DFP to recirculate the coefficients, or they may be tied to the CIN0-8 inputs of another DFP to cascade DFPs for longer filter lengths.

\overline{COENB} A low on the \overline{COENB} input enables the COUT0-8 outputs. A high on this input places all these outputs in their high impedance state.

DCM0-1 These two inputs determine the use of the internal decimation registers as follows:

| DCM1 | DCM0 | Decimation Function |
|------|------|-------------------------------------|
| 0 | 0 | Decimation registers not used |
| 0 | 1 | One decimation register is used |
| 1 | 0 | Two decimation registers are used |
| 1 | 1 | Three decimation registers are used |

The coefficients pass from cell to cell at a rate determined by the number of decimation registers used. When no decimation registers are used, coefficients move from cell to cell on each clock. When one decimation register is used, coefficients move from cell to cell on every other clock, etc. These signals are latched and delayed by one clock internal to the device.

SUM0-25 These 26 three-state outputs are used to output the results of the internal filter cell computations. Individual filter cell results or the result of the shift-and-add output stage can be output. If an individual filter cell result is to be output, the ADR0-2 signals select the filter cell result. The SHADD signal determines whether the selected filter cell result or the output stage adder result is output. The signals $\overline{\text{SENBH}}$ and $\overline{\text{SENB L}}$ enable the most significant and least significant bits of the SUM0-25 result respectively. Both $\overline{\text{SENBH}}$ and $\overline{\text{SENB L}}$ may be enabled simultaneously if the system has a 26-bit or larger bus. However individual enables are provided to facilitate use with a 16-bit bus. The ZR34891 in an LCC package has only 25 SUM outputs, SUM0-24.

$\overline{\text{SENBH}}$ A low on this input enables result bits SUM16-25. A high on this input places these bits in their high impedance state.

$\overline{\text{SENB L}}$ A low on this input enables result bits SUM0-15. A high on this input places these bits on their high impedance state.

ADR0-2 These three inputs select the one cell whose accumulator will be read through the output bus will be cleared when $\overline{\text{ERASE}}$ is low. These inputs are latched in the DFP and delayed by one clock internal to the device. If the ADR0-2 lines

(SUM0-25) or added to the output stage accumulator. They also determine which accumulator remain at the same address for more than one clock, the output at SUM0-25 will not change to reflect any subsequent accumulator updates in the addressed cell. Only the result available during the first clock, when ADR0-2 selects the cell, will be output. This does not hinder normal operation since the ADR0-2 lines are changed sequentially. This feature facilitates the interface with slow memories where the output is required to be fixed for more than one clock.

SHADD The SHADD input controls the activation of the shift and add operation in the output stage. This signal is latched on device and delayed by one clock internal to the device. Detailed explanation is given in the DFP Output Stage section.

$\overline{\text{RESET}}$ A low on this input synchronously clears all the internal registers, except the cell accumulators. It can be used with $\overline{\text{ERASE}}$ to also clear all the accumulators simultaneously. This signal is latched in the DFP and delayed by one clock internal to the device.

$\overline{\text{ERASE}}$ A low on this input synchronously clears the cell accumulator selected by the ADR0-2 signals. If $\overline{\text{RESET}}$ is also low simultaneously, all cell accumulators are cleared.

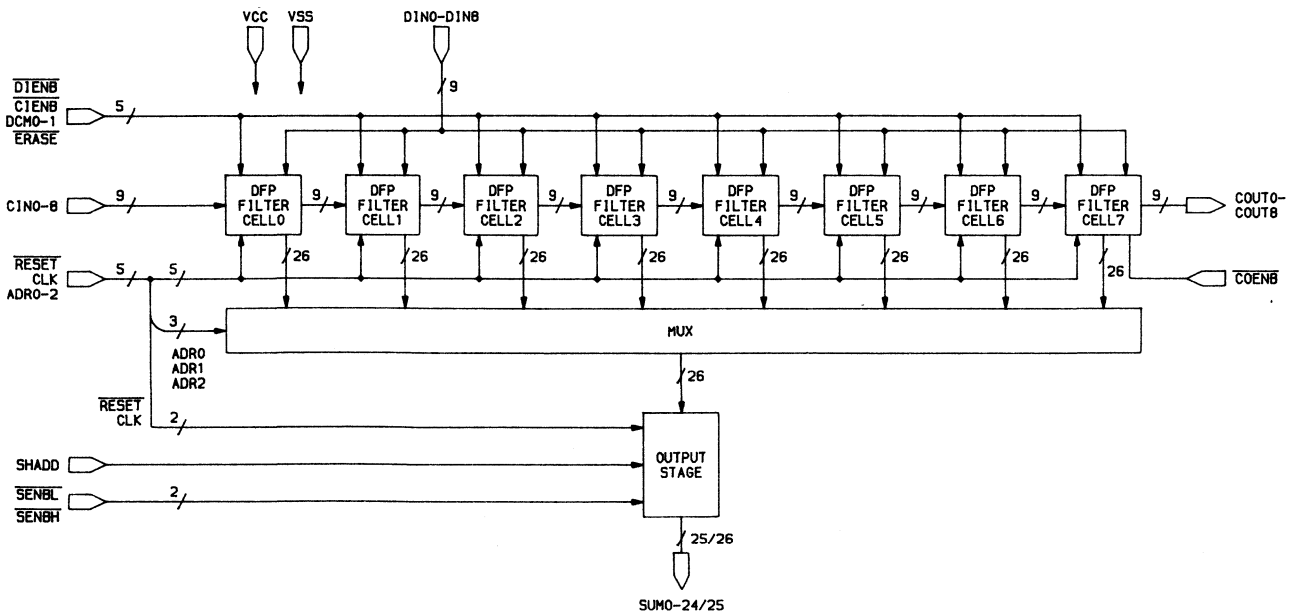


FIGURE 1. ZR33891 DFP BLOCK DIAGRAM.

FUNCTIONAL DESCRIPTION

The Digital Filter Processor (DFP) is composed of eight filter cells cascaded together and an output stage for combining or selecting filter cell outputs (Figure 1). Each filter cell contains a multiplier-accumulator and several registers (Figure 2). Each 9-bit coefficient is multiplied by a 9-bit data sample, with the result added to the 26-bit accumulator contents. The coefficient output of each cell is cascaded to the coefficient input of the next cell to its right.

DFP FILTER CELL

A 9-bit coefficient (CIN0–8) enters each cell through the C register on the left and exits the cell on the right as signals COUT0–8. The coefficients may move directly from the C register to the output, exiting the cell on the clock following its entrance. When decimation is selected the coefficient exit is delayed by 1, 2 or 3 clocks by passing through one or more decimation registers (D1, D2 or D3).

The combination of D registers through which the coefficient passes is determined by the state of DCM0 and DCM1. The output signals (COUT0–8) are connected to the CIN0–8 inputs of the next cell to its right. The $\overline{\text{COENB}}$ input signal enables the COUT0–8 outputs of the right most cell to the COUT0–8 pins of the device.

The C and D registers are enabled for loading by $\overline{\text{CIENB}}$. Loading is synchronous with CLK when $\overline{\text{CIENB}}$ is low. Note that $\overline{\text{CIENB}}$ is latched internally. It enables the register for loading after the next CLK following the onset of $\overline{\text{CIENB}}$ low. Actual loading occurs on the second CLK following the onset of $\overline{\text{CIENB}}$ low. Therefore $\overline{\text{CIENB}}$ must be low during the clock cycle immediately preceding presentation of the coefficient on the CIN0–8 inputs. In most basic FIR operations, $\overline{\text{CIENB}}$ will be low throughout the process, so this latching and delay sequence is only important during the initialization phase. When $\overline{\text{CIENB}}$ is high, the coefficients are frozen.

These registers are cleared synchronously under control of $\overline{\text{RESET}}$, which is latched and delayed exactly like $\overline{\text{CIENB}}$.

The output of the C register (C<0:8>) is one input to 9×9 multiplier.

The other input to the 9×9 multiplier comes from the output of the X register. This register is loaded with a data sample from the device input signals DIN0–8 discussed above. The X register is enabled for loading by $\overline{\text{DIENB}}$. Loading is synchronous with CLK when $\overline{\text{DIENB}}$ is low. Note that $\overline{\text{DIENB}}$ is latched internally. It enables the register for loading after the next CLK following the onset of $\overline{\text{DIENB}}$ low. Actual loading occurs on the second CLK following the onset of $\overline{\text{DIENB}}$ low.

Therefore $\overline{\text{DIENB}}$ must be low during the clock cycle immediately preceding presentation of the data sample on the DIN0–8 inputs. In most basic FIR operations, $\overline{\text{DIENB}}$ will be low throughout the process, so this latching and delay sequence is only important during the initialization phase. When $\overline{\text{DIENB}}$ is high, the X register is loaded with all zeros.

The multiplier is pipelined and is modeled as a multiplier core followed by two pipeline registers, MREG0 and MREG1 (Figure 2). The multiplier output is sign extended and input as one operand of the 26-bit adder. The other adder operand is the output of the 26-bit accumulator. The adder output is loaded synchronously into both the accumulator and the TREG.

The TREG loading is disabled by the cell select signal, CELL_n, where n is the cell number. The cell select is decoded from the ADR0-2 signals to generate the TREG load enable. The cell select is inverted and applied as the load enable to the TREG. Operation is such that the TREG is loaded whenever the cell is not selected. Therefore, TREG is loaded every clock except the clock following cell selection. The purpose of the TREG is to hold the result of a sum-of-products calculation during the clock when the accumulator is cleared to prepare for the next sum-of-products calculation. This allows continuous accumulation without wasting clocks.

The accumulator is loaded with the adder output every clock unless it is cleared. It is cleared synchronously in two ways. When $\overline{\text{RESET}}$ and $\overline{\text{ERASE}}$ are both low, the accumulator is cleared along with all other registers on the device. Since, $\overline{\text{ERASE}}$ and $\overline{\text{RESET}}$ are latched and delayed one clock internally, clearing occurs on the second CLK following the onset of both $\overline{\text{ERASE}}$ and $\overline{\text{RESET}}$ low.

The second accumulator clearing mechanism clears a single accumulator in a selected cell. The cell select signal, CELL_n, decoded from ADR0-2 and the $\overline{\text{ERASE}}$ signal enable clearing of the accumulator on the next CLK.

The $\overline{\text{ERASE}}$ and $\overline{\text{RESET}}$ signals clear the DFP internal registers and states as follows:

| $\overline{\text{ERASE}}$ | $\overline{\text{RESET}}$ | CLEARING EFFECT |
|---------------------------|---------------------------|--|
| 1 | 1 | No clearing occurs, internal state remains same |
| 1 | 0 | Reset only active, all registers except accumulators are cleared, including the internal pipeline registers. |
| 0 | 1 | Erase only active, the accumulator whose address is given by the ADR0–2 inputs is cleared. |
| 0 | 0 | Both Reset and Erase active, all accumulators as well as all other registers are cleared. |

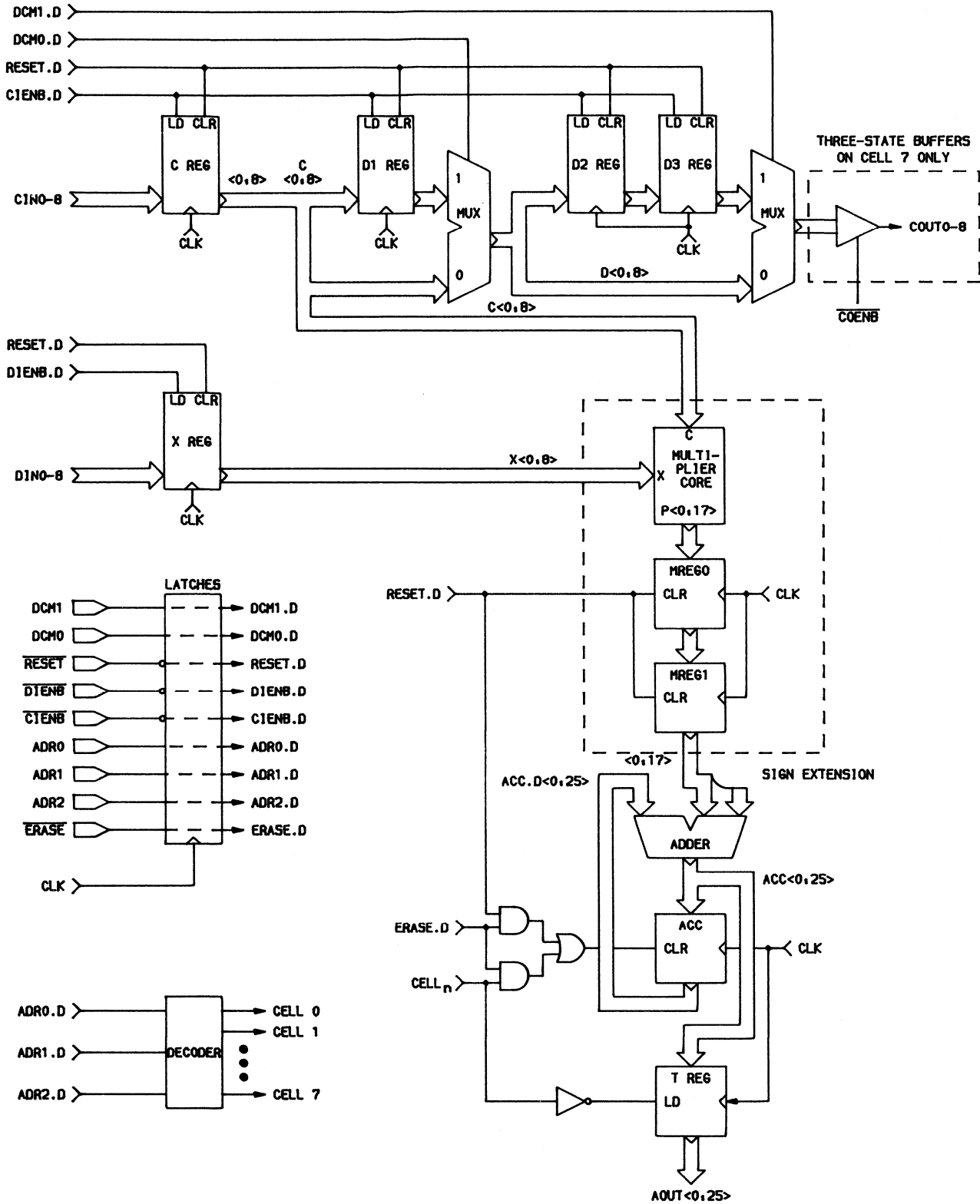


FIGURE 2. ZR33891 DFP FILTER CELL.

THE DFP OUTPUT STAGE

The output stage consists of a 26-bit adder, 26-bit register, feedback multiplexer from the register to the adder, an output multiplexer and a 26-bit three-state driver stage (Figure 3).

The 26-bit output adder can add any filter cell accumulator result to the 18 most significant bits of the output buffer. This result is stored back in the output buffer. This operation takes place in one clock period. The eight LSBs are lost. The filter cell accumulator is selected by the ADR0-2 inputs.

The 18 MSBs of the output buffer actually pass through the zero mux on their way to the output adder input. The zero mux is controlled by the SHADD input signal and selects either the output buffer 18 MSBs or all zeros for the adder input. A low on the SHADD input selects zero. A high on the SHADD input selects the output buffer MSBs, thus activating the shift-and-add operation. The SHADD signal is latched and delayed by one clock internally.

The 26 least significant bits (LSBs) from either a cell accumulator or the output buffer are output on the SUM0-25 bus. The output mux determines whether the cell accumulator selected by ADR0-2 or the output buffer is output to the bus. This mux is controlled by the SHADD input signal. Control is based on the state of the SHADD during two successive clocks; in other words, the output mux selection contains memory. If SHADD is low during a clock cycle and was low during the previous clock, the output mux selects the contents of the filter cell accumulator addressed by ADR0-2. Otherwise the output mux selects the contents of the output buffer.

If the ADR0-2 lines remain at the same address for more than one clock, the output at SUM0-25 will not change to reflect any subsequent accumulator updates in the addressed cell. Only the result available during the first clock when ADR0-2 selects the cell will be output. This does not hinder normal FIR operation since the ADR0-2 lines are changed sequentially. This feature facilitates the interface with slow memories where the output is required to be fixed for more than one clock.

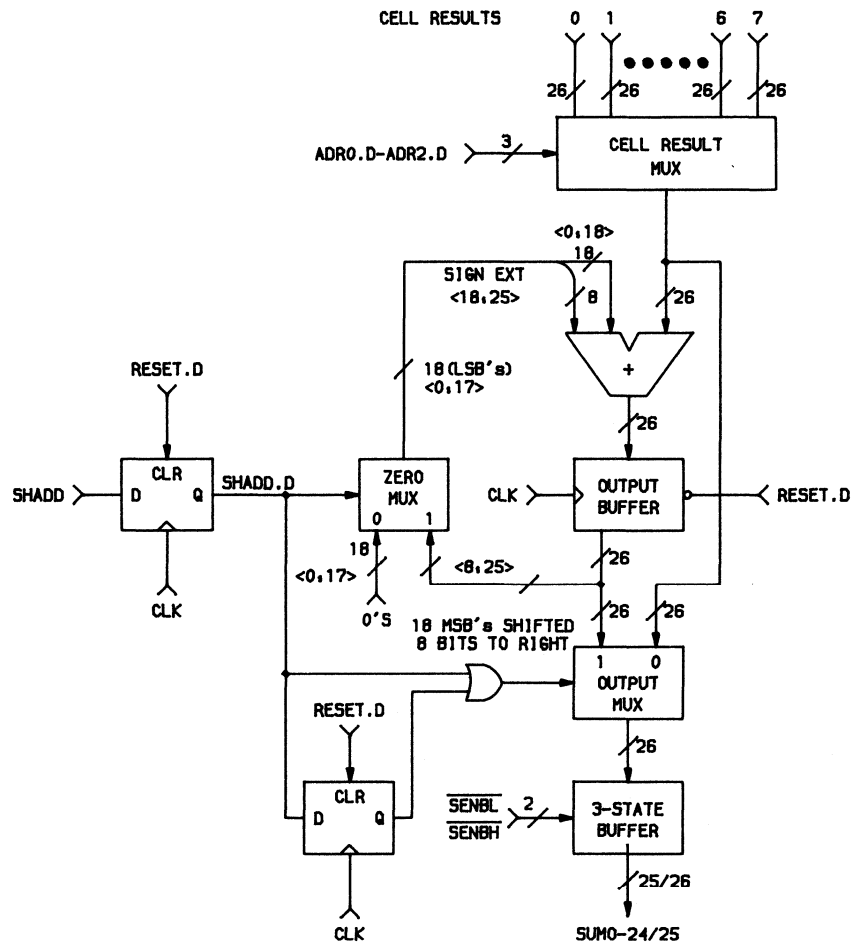


FIGURE 3. ZR33891 DFP OUTPUT STAGE.

The SUM0–25 output bus is controlled by the $\overline{\text{SENBH}}$ and $\overline{\text{SENB L}}$ signals. A low on $\overline{\text{SENB L}}$ enables bits SUM0–15. A low on $\overline{\text{SENBH}}$ enables bits SUM16–25. Thus all 26 bits can be output simultaneously if the external system has a 26-bit or larger bus. If the external system bus is only 16 bits, the bits can be enabled in two groups of 16 and 9 bits (sign extended).

DFP ARITHMETIC

Both data samples and coefficients can be represented as either 8-bit unsigned or 9-bit two’s complement numbers. The 9x9 bit multiplier in each cell expects 9-bit two’s complement operands. The binary format of 9-bit two’s complement is shown below. Note that if the most significant or sign bit is held at logical zero, the 9-bit two’s complement multiplier can multiply 8-bit unsigned operands. Only the upper (positive) half of the two’s complement binary range is used.

The multiplier output is 18 bits and the accumulator is 26 bits. The accumulator width determines the maximum possible number of terms in the sum of products without overflow. The maximum number of terms depends also on the number system and the distribution of the coefficient and data values. As a worst case assume the coefficients and data samples are always at their maximum absolute values. Then the maximum numbers of terms in the sum products are:

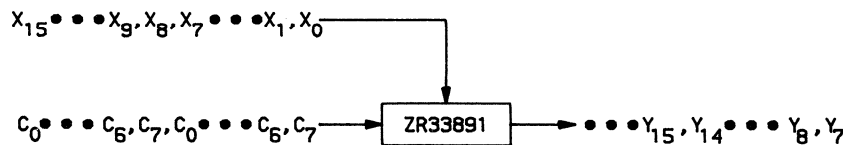
Maximum Number of Terms

| Number System | Maximum Number of Terms | |
|--|-------------------------|-------|
| | 8-bit | 9-bit |
| Two unsigned vectors | 516 | N/A |
| Two two’s complement | | |
| • two positive vectors | 1040 | 258 |
| • negative vectors | 1024 | 256 |
| • one positive and one negative vector | 1032 | 257 |
| One unsigned and one two’s complement vector | | |
| • positive two’s complement vector | 518 | 258 |
| • negative two’s complement vector | 514 | 257 |

For practical FIR filters, the coefficients are never all near maximum value, so even larger vectors are possible in practice.

BASIC FIR OPERATION

A simple, 20 MHz 8-tap filter example serves to illustrate more clearly the operation of the DFP. The sequence table (Table 1) shows the results of the multiply accumulate in each cell after each clock. The coefficient sequence, C_n , enters the DFP on the left and moves from left to right through the cells. The data sample sequence, X_n , enters the DFP from the top, with each cell receiving the same sample simultaneously. Each cell accumulates the sum of products for one output point. Eight sums of products are calculated simultaneously, but staggered in time so that a new output is available every system clock.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | CELL4 | CELL5 | CELL6 | CELL7 | SUM/CLR |
|-----|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|------------|
| 0 | $C_7 \cdot X_0$ | 0 | 0 | 0 | | | | | — |
| 1 | $+ C_6 \cdot X_1$ | $C_7 \cdot X_1$ | 0 | 0 | | | | | — |
| 2 | $+ C_5 \cdot X_2$ | $+ C_6 \cdot X_2$ | $C_7 \cdot X_2$ | 0 | | | | | — |
| 3 | $+ C_4 \cdot X_3$ | $+ C_5 \cdot X_3$ | $+ C_6 \cdot X_3$ | $C_7 \cdot X_3$ | | | | | — |
| 4 | $+ C_3 \cdot X_4$ | $+ C_4 \cdot X_4$ | $+ C_5 \cdot X_4$ | $+ C_6 \cdot X_4$ | $C_7 \cdot X_4$ | | | | — |
| 5 | $+ C_2 \cdot X_5$ | $+ C_3 \cdot X_5$ | $+ C_4 \cdot X_5$ | $+ C_5 \cdot X_5$ | $+ C_6 \cdot X_5$ | $C_7 \cdot X_5$ | | | — |
| 6 | $+ C_1 \cdot X_6$ | $+ C_2 \cdot X_6$ | $+ C_3 \cdot X_6$ | $+ C_4 \cdot X_6$ | $+ C_5 \cdot X_6$ | $+ C_6 \cdot X_6$ | $C_7 \cdot X_6$ | | — |
| 7 | $+ C_0 \cdot X_7$ | $+ C_1 \cdot X_7$ | $+ C_2 \cdot X_7$ | $+ C_3 \cdot X_7$ | $+ C_4 \cdot X_7$ | $+ C_5 \cdot X_7$ | $+ C_6 \cdot X_7$ | $C_7 \cdot X_7$ | CELL0(Y7) |
| 8 | $C_7 \cdot X_8$ | $+ C_0 \cdot X_8$ | $+ C_1 \cdot X_8$ | $+ C_2 \cdot X_8$ | $+ C_3 \cdot X_8$ | $+ C_4 \cdot X_8$ | $+ C_5 \cdot X_8$ | $+ C_6 \cdot X_8$ | CELL1(Y8) |
| 9 | $+ C_6 \cdot X_9$ | $C_7 \cdot X_9$ | $+ C_0 \cdot X_9$ | $+ C_1 \cdot X_9$ | $+ C_2 \cdot X_9$ | $+ C_3 \cdot X_9$ | $+ C_4 \cdot X_9$ | $+ C_5 \cdot X_9$ | CELL2(Y9) |
| 10 | $+ C_5 \cdot X_{10}$ | $+ C_6 \cdot X_{10}$ | $C_7 \cdot X_{10}$ | $+ C_0 \cdot X_{10}$ | $+ C_1 \cdot X_{10}$ | $+ C_2 \cdot X_{10}$ | $+ C_3 \cdot X_{10}$ | $+ C_4 \cdot X_{10}$ | CELL3(Y10) |
| 11 | $+ C_4 \cdot X_{11}$ | $+ C_5 \cdot X_{11}$ | $+ C_6 \cdot X_{11}$ | $C_7 \cdot X_{11}$ | $+ C_0 \cdot X_{11}$ | $+ C_1 \cdot X_{11}$ | $+ C_2 \cdot X_{11}$ | $+ C_3 \cdot X_{11}$ | CELL4(Y11) |
| 12 | $+ C_3 \cdot X_{12}$ | $+ C_4 \cdot X_{12}$ | $+ C_5 \cdot X_{12}$ | $+ C_6 \cdot X_{12}$ | $C_7 \cdot X_{12}$ | $+ C_0 \cdot X_{12}$ | $+ C_1 \cdot X_{12}$ | $+ C_2 \cdot X_{12}$ | CELL5(Y12) |
| 13 | $+ C_2 \cdot X_{13}$ | $+ C_3 \cdot X_{13}$ | $+ C_4 \cdot X_{13}$ | $+ C_5 \cdot X_{13}$ | $+ C_6 \cdot X_{13}$ | $C_7 \cdot X_{13}$ | $+ C_0 \cdot X_{13}$ | $+ C_1 \cdot X_{13}$ | CELL6(Y13) |
| 14 | $+ C_1 \cdot X_{14}$ | $+ C_2 \cdot X_{14}$ | $+ C_3 \cdot X_{14}$ | $+ C_4 \cdot X_{14}$ | $+ C_5 \cdot X_{14}$ | $+ C_6 \cdot X_{14}$ | $+ C_7 \cdot X_{14}$ | $+ C_0 \cdot X_{14}$ | CELL7(Y14) |
| 15 | $+ C_0 \cdot X_{15}$ | $+ C_1 \cdot X_{15}$ | $+ C_2 \cdot X_{15}$ | $+ C_3 \cdot X_{15}$ | $+ C_4 \cdot X_{15}$ | $+ C_5 \cdot X_{15}$ | $+ C_6 \cdot X_{15}$ | $C_7 \cdot X_{15}$ | CELL0(Y15) |

TABLE 1. ZR33891 20 MHz, 8-TAP FIR FILTER SEQUENCE.

Detailed operation of the DFP to perform a basic 8-tap, 9-bit coefficient, 9-bit data, 20 MHz FIR filter is best understood by observing the schematic (Figure 4) and timing diagram (Figure 5). The internal pipeline length of the DFP is four (4) clock cycles, corresponding to the register levels CREG (or XREG), MREG0, MREG1, and TREG (Figures 2 and 3). Therefore the delay from presentation of data and coefficients at the DIN0-8 and CIN0-8 inputs to a sum appearing at the SUM0-25 output is:

$$k + T_d$$

where

k = filter length

T_d = 4, the internal pipeline delay of DFP

After the pipeline has filled, a new output sample is available every clock. The delay to last sample output from last sample input is T_d.

The output sums, Y_n, shown in the timing diagram are derived from the sum-of-products equation:

$$Y(n) = C(0) \times X(n) + C(1) \times X(n-1) + C(2) \times X(n-2) + C(3) \times X(n-3) + C(4) \times X(n-4) + C(5) \times X(n-5) + C(6) \times X(n-6) + C(7) \times X(n-7)$$

EXTENDED FIR FILTER LENGTH

Filter lengths greater than eight taps can be created by either cascading together multiple DFP devices or "reusing" a single device. Using multiple devices, an FIR filter of over 500 taps can be constructed to operate at a 20 MHz sample rate. Using a single device clocked at 20 MHz, an FIR filter of over 500 taps can be constructed to operate at less than a 20 MHz sample rate. Combinations of these two techniques are also possible.

| | Two's Complement, 9-bit | | | | | | | | | Unsigned, 8-bit (with sign bit at logical 0) | | | | | | | | |
|-------|-------------------------|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Max + | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | | | | • | | | | | | | | | • | | | | |
| | | | | | • | | | | | | | | | • | | | | |
| | | | | | • | | | | | | | | | • | | | | |
| +1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | |
| | | | | | • | | | | | | | | | | | | | |
| | | | | | • | | | | | | | | | | | | | |
| | | | | | • | | | | | | | | | | | | | |
| Max - | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | |

Unused

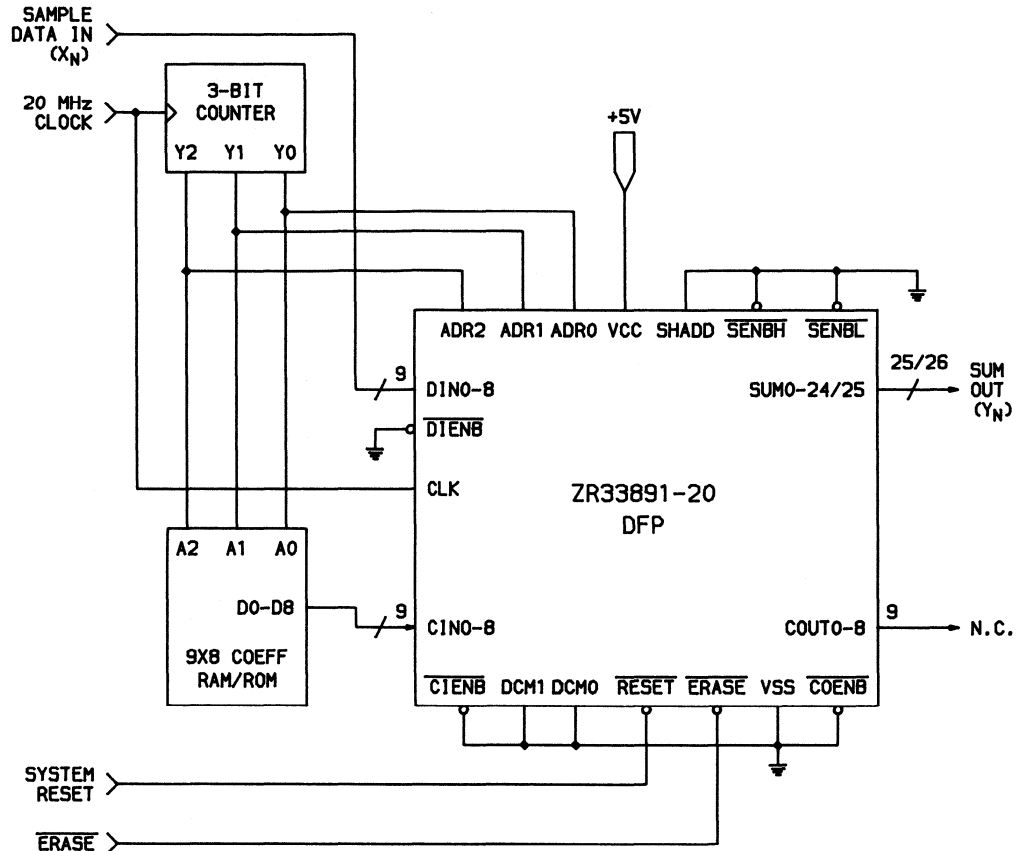
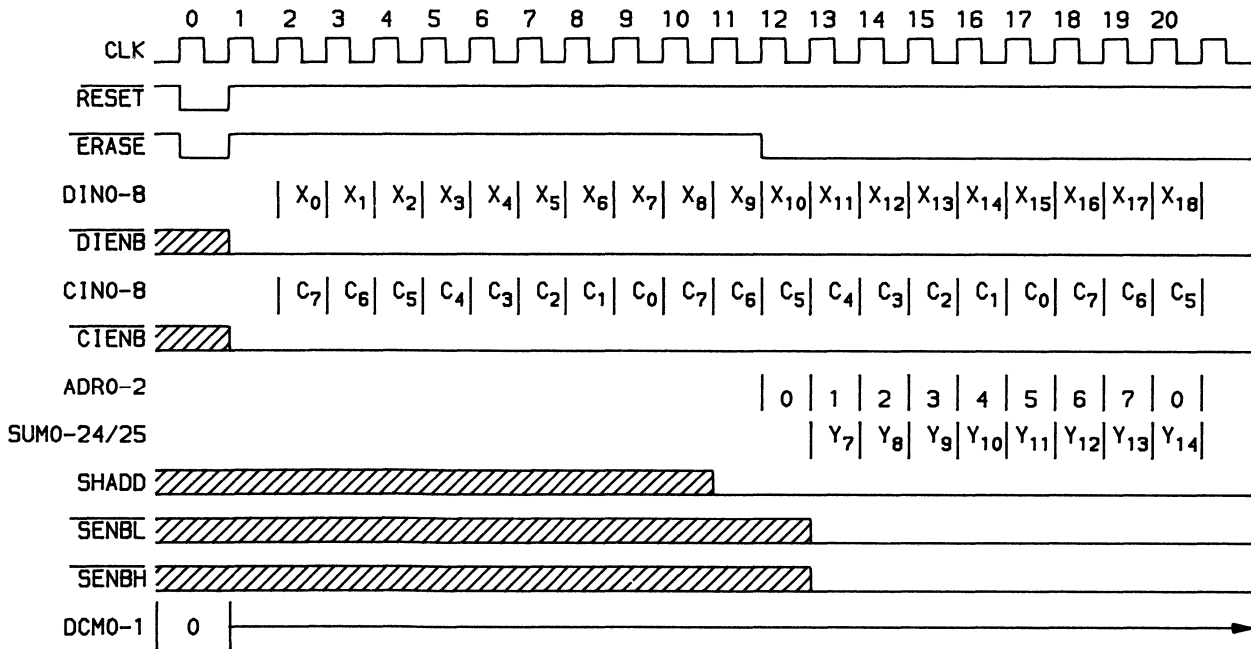


FIGURE 4. ZR33891 20 MHz, 8-TAP FIR FILTER APPLICATION SCHEMATIC.



$$Y_N = \sum_{K=0}^7 C_K \cdot X_{N-K}$$

FIGURE 5. ZR33891 20 MHz, 8-TAP FIR FILTER TIMING.

CASCADE CONFIGURATION

To design a filter length $L > 8$, $L/8$ DFPs are cascaded by connecting the COUT0-8 outputs of the (i)th DFP to the CIN0-8 inputs of the (i+1)th DFP. The DIN0-8 inputs and SUM0-25 outputs of all the DFPs are also tied together. A specific example

of two cascaded DFPs illustrates the technique (Figure 6). Timing (Figure 7) is similar to the simple 8-tap FIR, except the $\overline{\text{ERASE}}$ and $\overline{\text{SENBL}}/\overline{\text{SENBH}}$ signals must be enabled independently for the two DFPs in order to clear the correct accumulators and enable the SUM0-25 output signals at the proper times.

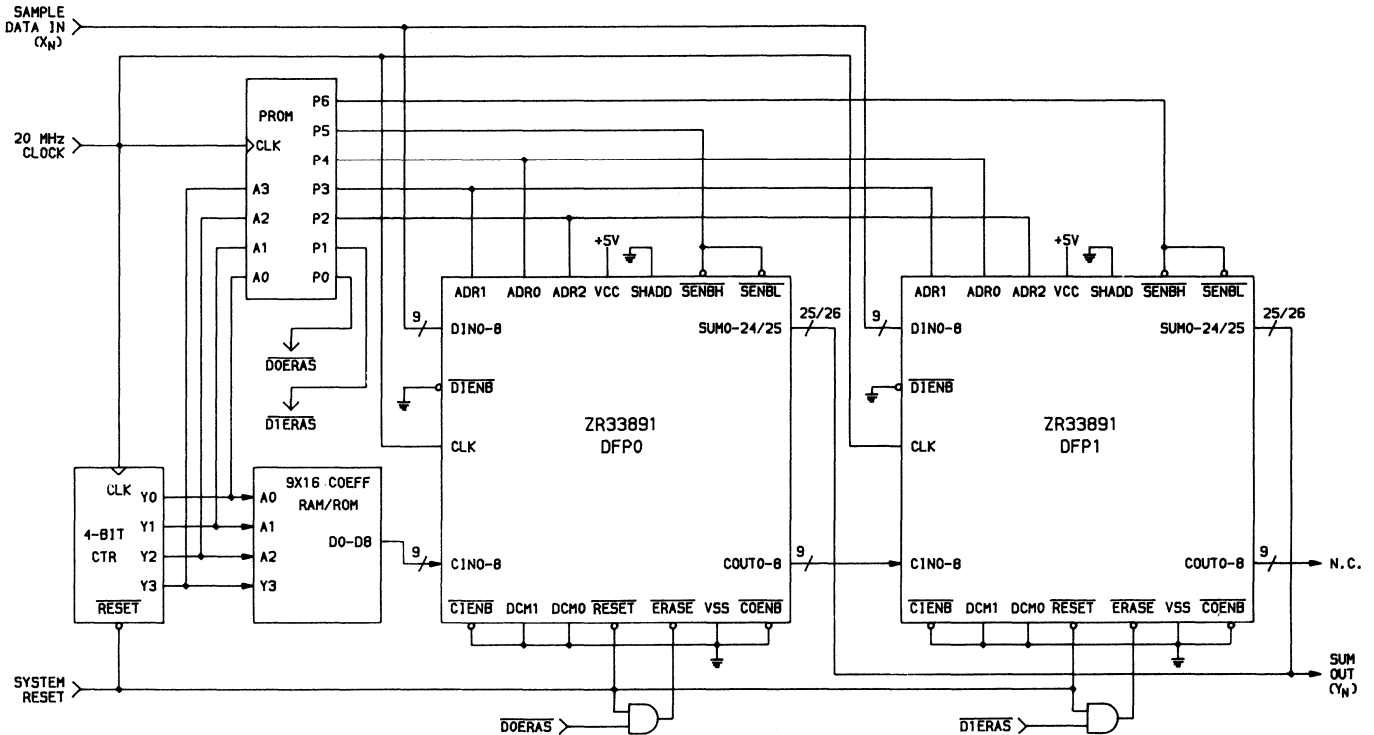
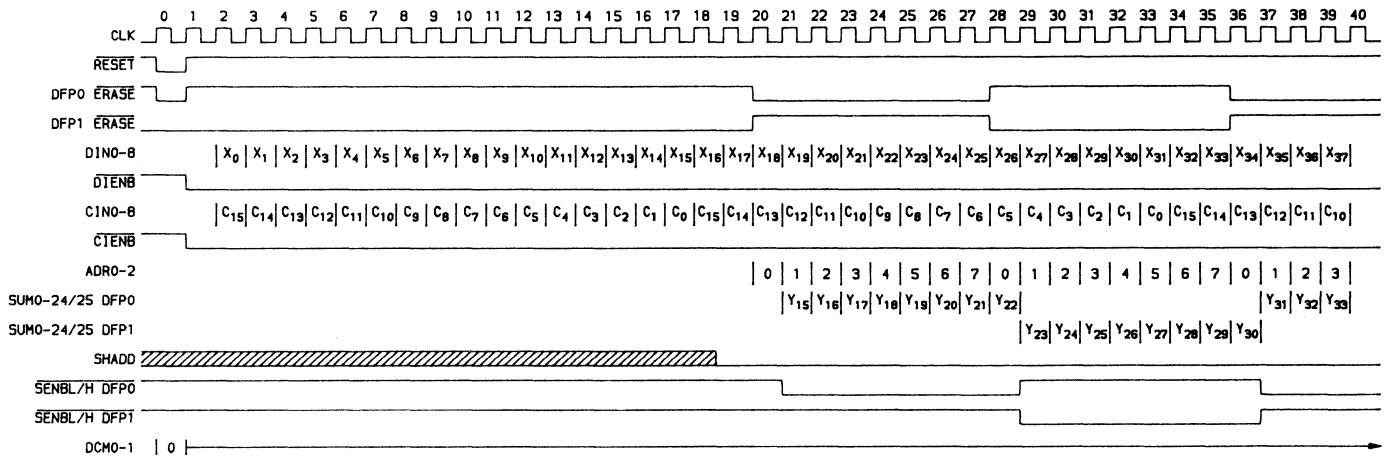


FIGURE 6. ZR33891 20 MHz, 16-TAP FIR FILTER CASCADE APPLICATION SCHEMATIC.



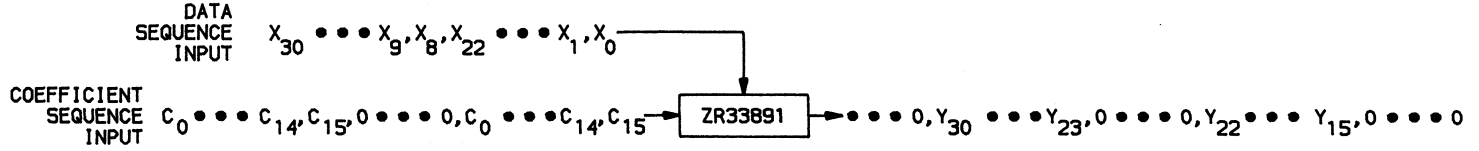
$$Y_N = \sum_{k=0}^{15} C_k \cdot X_{N-k}$$

FIGURE 7. ZR33891 16-TAP 20 MHz FILTER TIMING USING TWO CASCADED ZR33891s.

SINGLE DFP CONFIGURATION

Using a single DFP, a filter of length $L > 8$ can be constructed by processing in $L/8$ passes as illustrated in the following table (Table 2) for a 16-tap FIR. Each pass is composed of $T_p = 7 + L$ cycles and computes eight output samples. In pass i , the

sample with indices $i*8$ to $i*8 + (L-1)$ enter the $DIN0-8$ inputs. The coefficients $C_0 - C_{L-1}$ enter the $CIN0-8$ inputs, followed by seven zeros. As these zeros are entered, the result samples are output and the accumulators reset. Initial filling of the pipeline is not shown in this sequence table. Filter outputs can be put through a FIFO to even out the sample rate.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | CELL4 | CELL5 | CELL6 | CELL7 | SUM/CLR |
|-----|-------------------------|--------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-------------------------|------------|
| 6 | $C_{15} \cdot X_0$ | 0 | 0 | 0 | | | | | — |
| 7 | $+ C_{14} \cdot X_1$ | $C_{15} \cdot X_1$ | 0 | 0 | | | | | — |
| 8 | $+ C_{13} \cdot X_2$ | | $C_{15} \cdot X_2$ | 0 | | | | | — |
| 9 | $+ C_{12} \cdot X_3$ | | | $C_{15} \cdot X_3$ | | | | | — |
| 10 | $+ C_{11} \cdot X_4$ | | | $+ C_{14} \cdot X_4$ | $C_{15} \cdot X_4$ | | | | — |
| 11 | $+ C_{10} \cdot X_5$ | | | $+ C_{13} \cdot X_5$ | | $C_{15} \cdot X_5$ | | | — |
| 12 | $+ C_9 \cdot X_6$ | | | $+ C_{12} \cdot X_6$ | | | $C_{15} \cdot X_6$ | | — |
| 13 | $+ C_8 \cdot X_7$ | | | $+ C_{11} \cdot X_7$ | | | | $C_{15} \cdot X_7$ | — |
| 14 | $+ C_7 \cdot X_8$ | | | $+ C_{10} \cdot X_8$ | | | | $+ C_{14} \cdot X_8$ | — |
| 15 | $+ C_6 \cdot X_9$ | | | $+ C_9 \cdot X_9$ | | | | $+ C_{13} \cdot X_9$ | — |
| 16 | $+ C_5 \cdot X_{10}$ | | | $+ C_8 \cdot X_{10}$ | | | | $+ C_{12} \cdot X_{10}$ | — |
| 17 | $+ C_4 \cdot X_{11}$ | | | $+ C_7 \cdot X_{11}$ | | | | $+ C_{11} \cdot X_{11}$ | — |
| 18 | $+ C_3 \cdot X_{12}$ | | | $+ C_6 \cdot X_{12}$ | | | | $+ C_{10} \cdot X_{12}$ | — |
| 19 | $+ C_2 \cdot X_{13}$ | | | $+ C_5 \cdot X_{13}$ | | | | $+ C_9 \cdot X_{13}$ | — |
| 20 | $+ C_1 \cdot X_{14}$ | | | $+ C_4 \cdot X_{14}$ | | | | $+ C_8 \cdot X_{14}$ | — |
| 21 | $+ C_0 \cdot X_{15}$ | | | $+ C_3 \cdot X_{15}$ | | | | $+ C_7 \cdot X_{15}$ | CELL0(Y15) |
| 22 | 0 | $C_0 \cdot X_{16}$ | | $+ C_2 \cdot X_{16}$ | | | | $+ C_6 \cdot X_{16}$ | CELL1(Y16) |
| 23 | 0 | 0 | $C_0 \cdot X_{17}$ | $+ C_1 \cdot X_{17}$ | | | | $+ C_5 \cdot X_{17}$ | CELL2(Y17) |
| 24 | 0 | 0 | 0 | $+ C_0 \cdot X_{18}$ | | | | $+ C_4 \cdot X_{18}$ | CELL3(Y18) |
| 25 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{19}$ | | | $+ C_3 \cdot X_{19}$ | CELL4(Y19) |
| 26 | 0 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{20}$ | | $+ C_2 \cdot X_{20}$ | CELL5(Y20) |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{21}$ | $+ C_1 \cdot X_{21}$ | CELL6(Y21) |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $+ C_0 \cdot X_{22}$ | CELL7(Y22) |
| 29 | $C_{15} \cdot X_8$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — |
| 30 | $+ C_{14} \cdot X_9$ | $C_{15} \cdot X_9$ | 0 | 0 | 0 | 0 | 0 | 0 | — |
| 31 | $+ C_{13} \cdot X_{10}$ | | $C_{15} \cdot X_{10}$ | 0 | 0 | 0 | 0 | 0 | — |
| 32 | $+ C_{12} \cdot X_{11}$ | | | $C_{15} \cdot X_{11}$ | 0 | 0 | 0 | 0 | — |
| 33 | $+ C_{11} \cdot X_{12}$ | | | | $C_{15} \cdot X_{12}$ | 0 | 0 | 0 | — |
| 34 | $+ C_{10} \cdot X_{13}$ | | | | | $C_{15} \cdot X_{13}$ | 0 | 0 | — |
| 35 | $+ C_9 \cdot X_{14}$ | | | | | | $C_{15} \cdot X_{14}$ | 0 | — |
| 36 | $+ C_8 \cdot X_{15}$ | | | | | | | $C_{15} \cdot X_{15}$ | — |
| 37 | $+ C_7 \cdot X_{16}$ | | | | | | | $+ C_{14} \cdot X_{16}$ | — |
| 38 | $+ C_6 \cdot X_{17}$ | | | | | | | $+ C_{13} \cdot X_{17}$ | — |
| 39 | $+ C_5 \cdot X_{18}$ | | | | | | | $+ C_{12} \cdot X_{18}$ | — |
| 40 | $+ C_4 \cdot X_{19}$ | | | | | | | $+ C_{11} \cdot X_{19}$ | — |
| 41 | $+ C_3 \cdot X_{20}$ | | | | | | | $+ C_{10} \cdot X_{20}$ | — |
| 42 | $+ C_2 \cdot X_{21}$ | | | | | | | $+ C_9 \cdot X_{21}$ | — |
| 43 | $+ C_1 \cdot X_{22}$ | | | | | | | $+ C_8 \cdot X_{22}$ | — |
| 44 | $+ C_0 \cdot X_{23}$ | | | | | | | $+ C_7 \cdot X_{23}$ | CELL0(Y23) |
| 45 | 0 | $C_0 \cdot X_{24}$ | | | | | | $+ C_6 \cdot X_{24}$ | CELL1(Y24) |
| 46 | 0 | 0 | $C_0 \cdot X_{25}$ | | | | | $+ C_5 \cdot X_{25}$ | CELL2(Y25) |
| 47 | 0 | 0 | 0 | $C_0 \cdot X_{26}$ | | | | $+ C_4 \cdot X_{26}$ | CELL3(Y26) |
| 48 | 0 | 0 | 0 | 0 | $C_0 \cdot X_{27}$ | | | $+ C_3 \cdot X_{27}$ | CELL4(Y27) |

EXTENDED COEFFICIENT AND DATA SAMPLE WORD SIZE

The sample and coefficient word size can be extended by utilizing several DFPs in parallel to get the maximum sample rate or a single DFP with resulting lower sample rates. The technique is to compute partial products of 9×9 and combine these partial products by shifting and adding to obtain the final result. The shifting and adding can be accomplished with external adders (at full speed) or with the DFP's shift-and-add mechanism contained in its output stage (at reduced speed).

DECIMATION/RESAMPLING

The ZR33891 DFP provides a mechanism for decimating by factors of 2, 3 or 4. From the DFP filter cell block diagram (Figure 2), note the three D registers and two multiplexers in the coefficient path through the cell. These allow the coefficients to be delayed by 1, 2 or 3 clocks through the cell. The sequence table (Table 3) for a decimate-by-two filter illustrates the technique (internal cell pipelining ignored for simplicity).

Detailed timing for a 20 MHz input sample rate, 10 MHz output sample rate (i.e., decimate-by-two), 16-tap FIR filter, including pipelining, is shown in Figure 8. This filter requires only a single ZR33891 DFP.

OTHER DFP APPLICATIONS

The ZR33891 DFP is a versatile device with many applications beyond simple FIR filtering. The following list is a small sample of the many applications possible with the DFP. Implementations of these applications are discussed in greater detail in the various ZR33891 Zoran Application Notes.

- Higher precision arithmetic (e.g. 16×16)
 - Single DFP implementation
 - Multiple DFP implementation
- Higher bandwidth (40 MHz and up)
- Decimation/Interpolation
- 2-D FIR Spatial Filtering/Convolution/Correlation (e.g. 7×7 kernel convolution at 10 MHz rate)
- Matrix multiplication (8 elements \times 8 elements in $6 \mu\text{S}$)
- Complex multiply
- Adaptive filters
 - Echo cancellation
 - Adaptive equalization
- Butterfly computation
- Reverberation generators
- DFT
- Beam former
- Video Decoders

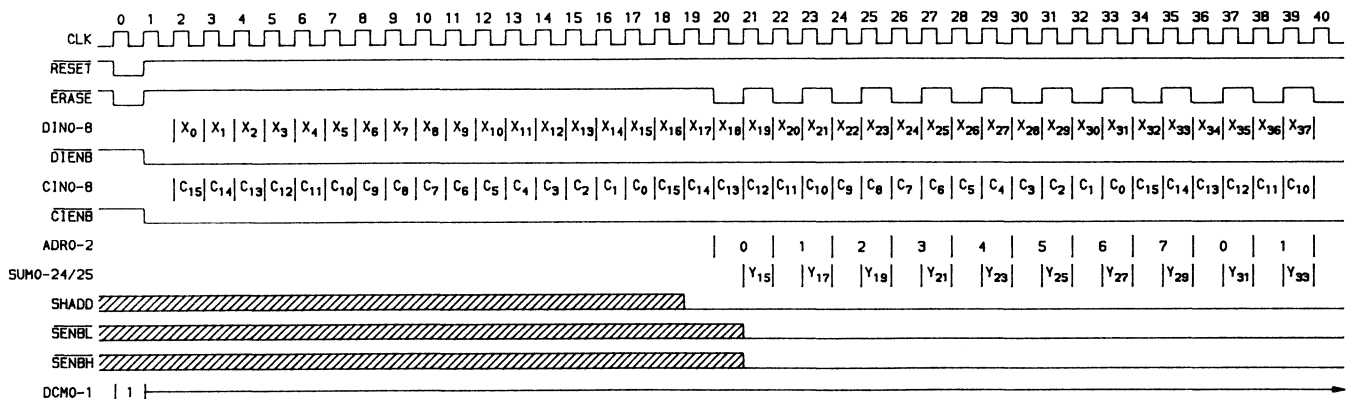
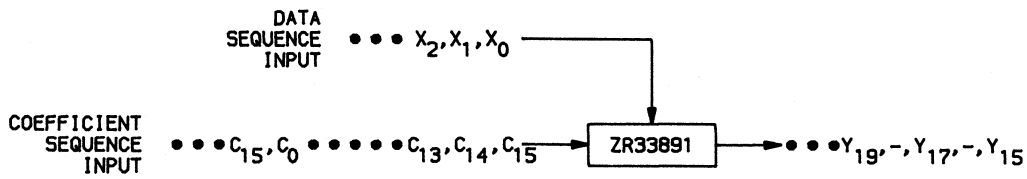


FIGURE 8. ZR33891 16-TAP DECIMATE-BY-TWO FIR FILTER TIMING, 20 MHz IN 10 MHz OUT.



| CLK | CELL0 | CELL1 | CELL2 | CELL3 | CELL4 | CELL5 | CELL6 | CELL7 | SUM/CLR |
|-----|-------------------------|----------------------|----------------------|----------------------|----------------------|-------------------------|-------------------------|-------------------------|------------|
| 6 | $C_{15} \cdot X_0$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 7 | $+ C_{14} \cdot X_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 8 | $+ C_{13} \cdot X_2$ | $C_{15} \cdot X_2$ | 0 | 0 | 0 | 0 | 0 | 0 | |
| 9 | $+ C_{12} \cdot X_3$ | ↓ | 0 | 0 | 0 | 0 | 0 | 0 | |
| 10 | $+ C_{11} \cdot X_4$ | ↓ | $C_{15} \cdot X_4$ | 0 | 0 | 0 | 0 | 0 | |
| 11 | $+ C_{10} \cdot X_5$ | ↓ | ↓ | 0 | 0 | 0 | 0 | 0 | |
| 12 | $+ C_9 \cdot X_6$ | ↓ | ↓ | $C_{15} \cdot X_6$ | 0 | 0 | 0 | 0 | |
| 13 | $+ C_8 \cdot X_7$ | ↓ | ↓ | ↓ | 0 | 0 | 0 | 0 | |
| 14 | $+ C_7 \cdot X_8$ | ↓ | ↓ | ↓ | $C_{15} \cdot X_8$ | 0 | 0 | 0 | |
| 15 | $+ C_6 \cdot X_9$ | ↓ | ↓ | ↓ | ↓ | 0 | 0 | 0 | |
| 16 | $+ C_5 \cdot X_{10}$ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{10}$ | 0 | 0 | |
| 17 | $+ C_4 \cdot X_{11}$ | ↓ | ↓ | ↓ | ↓ | ↓ | 0 | 0 | |
| 18 | $+ C_3 \cdot X_{12}$ | ↓ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{12}$ | 0 | |
| 19 | $+ C_2 \cdot X_{13}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | 0 | |
| 20 | $+ C_1 \cdot X_{14}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{14}$ | CELL0(Y15) |
| 21 | $+ C_0 \cdot X_{15}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{14} \cdot X_{15}$ | — |
| 22 | $C_{15} \cdot X_{16}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{13} \cdot X_{16}$ | CELL1(Y17) |
| 23 | $+ C_{14} \cdot X_{17}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{12} \cdot X_{17}$ | — |
| 24 | $+ C_{13} \cdot X_{18}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{11} \cdot X_{18}$ | CELL2(Y19) |
| 25 | $+ C_{12} \cdot X_{19}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_{10} \cdot X_{19}$ | — |
| 26 | $+ C_{11} \cdot X_{20}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_9 \cdot X_{20}$ | CELL3(Y21) |
| 27 | $+ C_{10} \cdot X_{21}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_8 \cdot X_{21}$ | — |
| 28 | $+ C_9 \cdot X_{22}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_7 \cdot X_{22}$ | CELL4(Y23) |
| 29 | $+ C_8 \cdot X_{23}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_6 \cdot X_{23}$ | — |
| 30 | $+ C_7 \cdot X_{24}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_5 \cdot X_{24}$ | CELL5(Y25) |
| 31 | $+ C_6 \cdot X_{25}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_4 \cdot X_{25}$ | — |
| 32 | $+ C_5 \cdot X_{26}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_3 \cdot X_{26}$ | CELL6(Y27) |
| 33 | $+ C_4 \cdot X_{27}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_2 \cdot X_{27}$ | — |
| 34 | $+ C_3 \cdot X_{28}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_1 \cdot X_{28}$ | CELL7(Y29) |
| 35 | $+ C_2 \cdot X_{29}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $+ C_0 \cdot X_{29}$ | — |
| 36 | $+ C_1 \cdot X_{30}$ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | $C_{15} \cdot X_{30}$ | CELL8(Y31) |
| 37 | $+ C_0 \cdot X_{31}$ | $+ C_2 \cdot X_{31}$ | $+ C_4 \cdot X_{31}$ | $+ C_6 \cdot X_{31}$ | $+ C_8 \cdot X_{31}$ | $+ C_{10} \cdot X_{31}$ | $+ C_{12} \cdot X_{31}$ | $+ C_{14} \cdot X_{31}$ | |

TABLE 3. ZR33891 16-TAP DECIMATE-BY-TWO FIR FILTER SEQUENCE, 20MHz IN 10 MHz OUT.

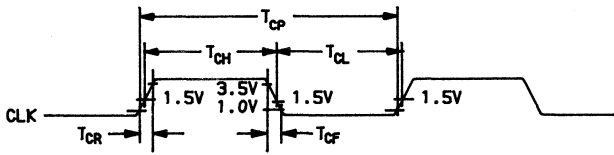
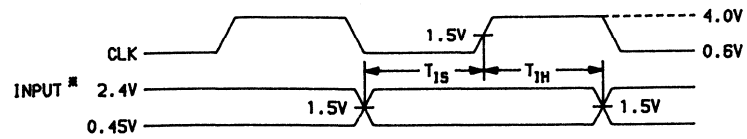
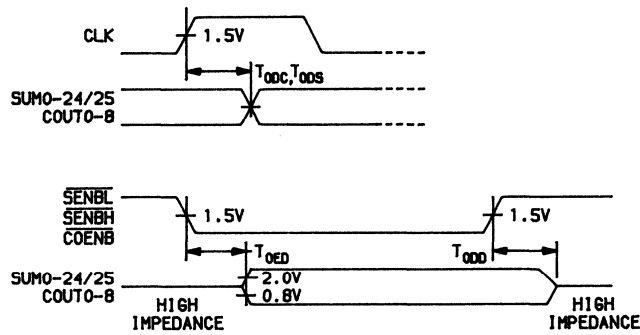


FIGURE 9. CLOCK AC PARAMETERS.



* INPUT INCLUDES D1N0-8, C1N0-8, DTENB, CTENB, ERASE, RESET, DCM0-1, ADRO-2, SHADD

FIGURE 10. INPUT SETUP AND HOLD.



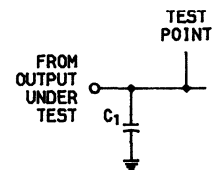
* SUM0-24/25, COUT0-8 ARE ASSUMED NOT TO BE IN HIGH-IMPEDANCE STATE

FIGURE 11. SUM⁰-24/25 COUT0-8, OUTPUT DELAYS.



A.C. TESTING, INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". INPUT AND OUTPUT TIMING MEASUREMENTS ARE MADE AT 1.5V FOR BOTH A LOGIC "1" AND "0"

FIGURE 12. A.C. TESTING INPUT, OUTPUT WAVEFORM.



NOTE: $C_1=50pF$ INCLUDING STRAY AND WIRING CAPACITANCE

FIGURE 13. NORMAL A.C. TEST LOAD.

| AC CHARACTERISTICS OVER OPERATING RANGE ZR33891-20 | | | | | | | |
|--|---------------------------------|---------------------|------|---------------------|------|-------|-----------------|
| Symbol | Parameter | COM ZR33891JC-20 | | MIL ZR33891JM-18 | | Units | Test Conditions |
| | | Min | Max | Min | Max | | |
| T _{CP} | Clock Period | 50 | 5000 | 55 | 5000 | ns | |
| T _{CL} | Clock Low | 22 | | 25 | | ns | |
| T _{CH} | Clock High | 22 | | 25 | | ns | |
| T _{IR} | Input Rise | | 5 | | 5 | ns | 1.0V to 3.5V |
| T _{IF} | Input Fall | | 5 | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 10 | | 10 | | ns | |
| T _{IH} | Input Hold | 0 | | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 40 | | 43 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 20 | | 23 | ns | |
| T _{ODD} | Output Disable Delay | | 20 | | 23 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 40 | | 43 | ns | |

| AC CHARACTERISTICS OVER OPERATING RANGE ZR33891-15 | | | | | | | |
|--|---------------------------------|---------------------|------|---------------------|------|-------|-----------------|
| Symbol | Parameter | COM ZR33891JC-15 | | MIL ZR33891JM-15 | | Units | Test Conditions |
| | | Min | Max | Min | Max | | |
| T _{CP} | Clock Period | 67 | 5000 | 67 | 5000 | ns | |
| T _{CL} | Clock Low | 30 | | 30 | | ns | |
| T _{CH} | Clock High | 30 | | 30 | | ns | |
| T _{IR} | Input Rise | | 5 | | 5 | ns | 1.0V to 3.5V |
| T _{IF} | Input Fall | | 5 | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 14 | | 14 | | ns | |
| T _{IH} | Input Hold | 0 | | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 50 | | 50 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 25 | | 25 | ns | |
| T _{ODD} | Output Disable Delay | | 25 | | 25 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 50 | | 50 | ns | |

| AC CHARACTERISTICS OVER OPERATING RANGE ZR33891-10 | | | | | | | |
|--|---------------------------------|---------------------|------|---------------------|------|-------|-----------------|
| Symbol | Parameter | COM ZR33891JC-10 | | MIL ZR33891JM-10 | | Units | Test Conditions |
| | | Min | Max | Min | Max | | |
| T _{CP} | Clock Period | 100 | 5000 | 100 | 5000 | ns | |
| T _{CL} | Clock Low | 40 | | 40 | | ns | |
| T _{CH} | Clock High | 40 | | 40 | | ns | |
| T _{IR} | Input Rise | | 5 | | 5 | ns | 1.0V to 3.5V |
| T _{IF} | Input Fall | | 5 | | 5 | ns | 1.0V to 3.5V |
| T _{IS} | Input Setup | 30 | | 30 | | ns | |
| T _{IH} | Input Hold | 0 | | 0 | | ns | |
| T _{ODC} | CLK to Coefficient Output Delay | | 65 | | 65 | ns | See Note 2 |
| T _{OED} | Output Enable Delay | | 40 | | 40 | ns | |
| T _{ODD} | Output Disable Delay | | 40 | | 40 | ns | |
| T _{ODS} | CLK to SUM Output Delay | | 65 | | 65 | ns | |

ABSOLUTE MAXIMUM RATINGS

(Above which useful life may be impaired)

Storage Temperature

MIL -65°C to + 150°C

COM -40°C to +85°C

Supply Voltage to Ground Potential

Continuous -0.5V to +7.0V

DC Voltage Applied to Outputs for High

Output State -0.5V to +V_{CC}max

DC Input Voltage -0.5V to +5.5V

DC Output Current, into Outputs

(not to exceed 200 mA total) 20mA/output

DC Input Current -30 to +5.0 mA

OPERATING RANGE

Commercial (C) Devices

Temperature 0°C ≤ T_A ≤ +70°C

Supply Voltage 4.75V ≤ V_{CC} ≤ 5.25V

Military (M) Devices

Temperature -55°C ≤ T_A ≤ +125°C

Supply Voltage 4.50V ≤ V_{CC} ≤ 5.50V

ZORAN Corporation cannot assume responsibility for use of any circuitry described other than circuitry embodied in a Zoran product.

| DC CHARACTERISTICS OVER OPERATING RANGE <small>unless otherwise specified</small> | | | | | |
|---|------------------------------------|------|-----------------------|-------|--|
| Symbol | Parameter | Min | Max | Units | Test Conditions |
| V _{IL} | Input Low Voltage | -0.5 | 0.8 | V | |
| V _{IH} | Input High Voltage | 2.0 | V _{CC} + 0.5 | V | |
| V _{OL} | Output Low Voltage | | 0.45 | V | I _{OL} = 2 mA |
| V _{OH} | Output High Voltage | 2.4 | | V | I _{OH} = -400 μA |
| I _{CC} | Power Supply Current | | 100 | mA | T _A = 0°C, V _{CC} = Max ¹ |
| I _{LI} | Input Leakage Current | | ± 10 | μA | 0 < V _{IN} < V _{CC} |
| I _{LO} | Output Leakage Current | | ± 10 | μA | 0.45 < V _{OUT} < V _{CC} |
| V _{CL} | Clock in Low Voltage | -0.5 | 0.6 | V | |
| V _{CH} | Clock in High Voltage | 4.0 | V _{CC} + 0.5 | V | |
| C _{IN} | Input Capacitance | | 10 | pF | f _C = 1 MHz |
| C _{IO} | I/O, Clock, and Output Capacitance | | 10 | pF | f _C = 1 MHz |

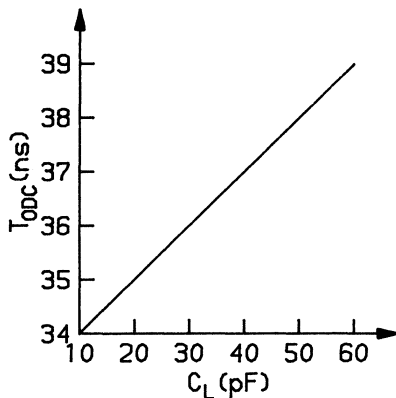


FIGURE 14. WORST CASE T_{0DC} AS A FUNCTION OF LOAD CAPACITANCE.

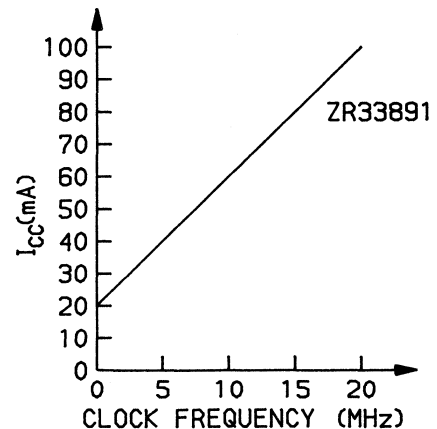


FIGURE 15. TYPICAL I_{CC} VS FREQUENCY AT 5.0V V_{CC}, 25°C.

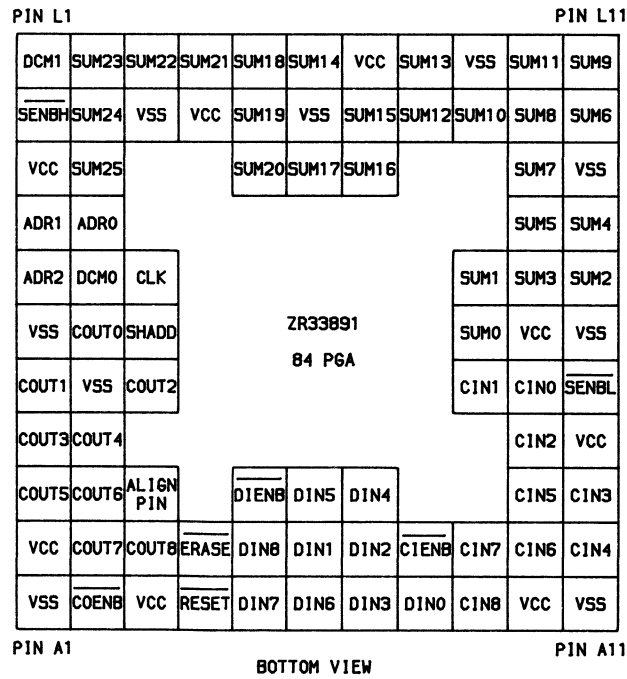
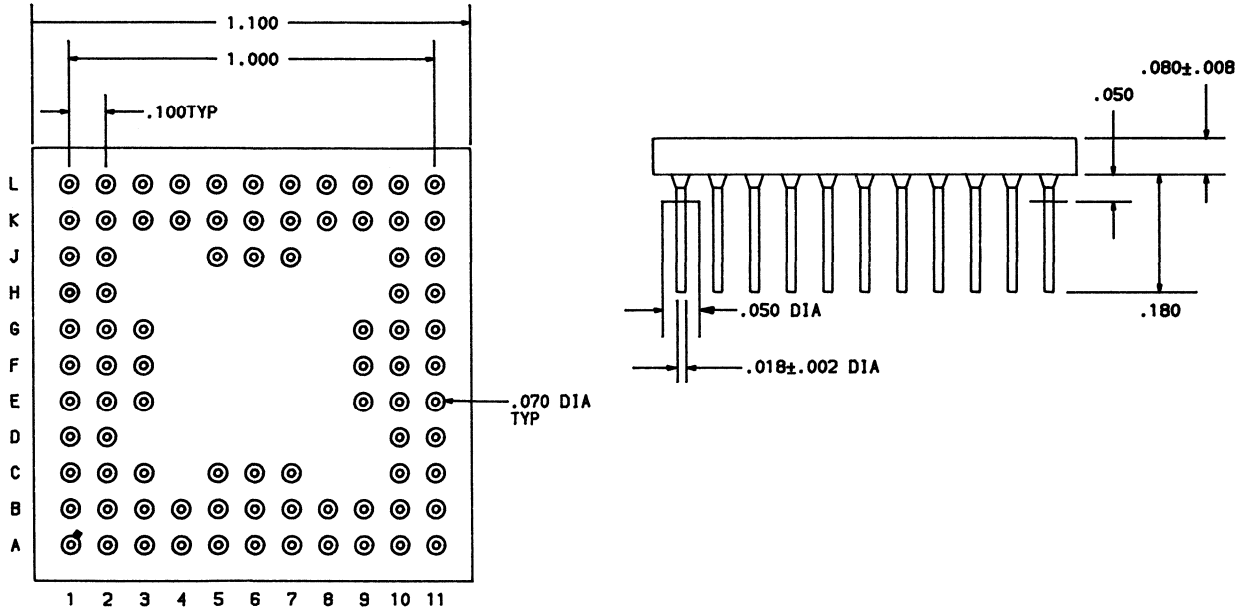
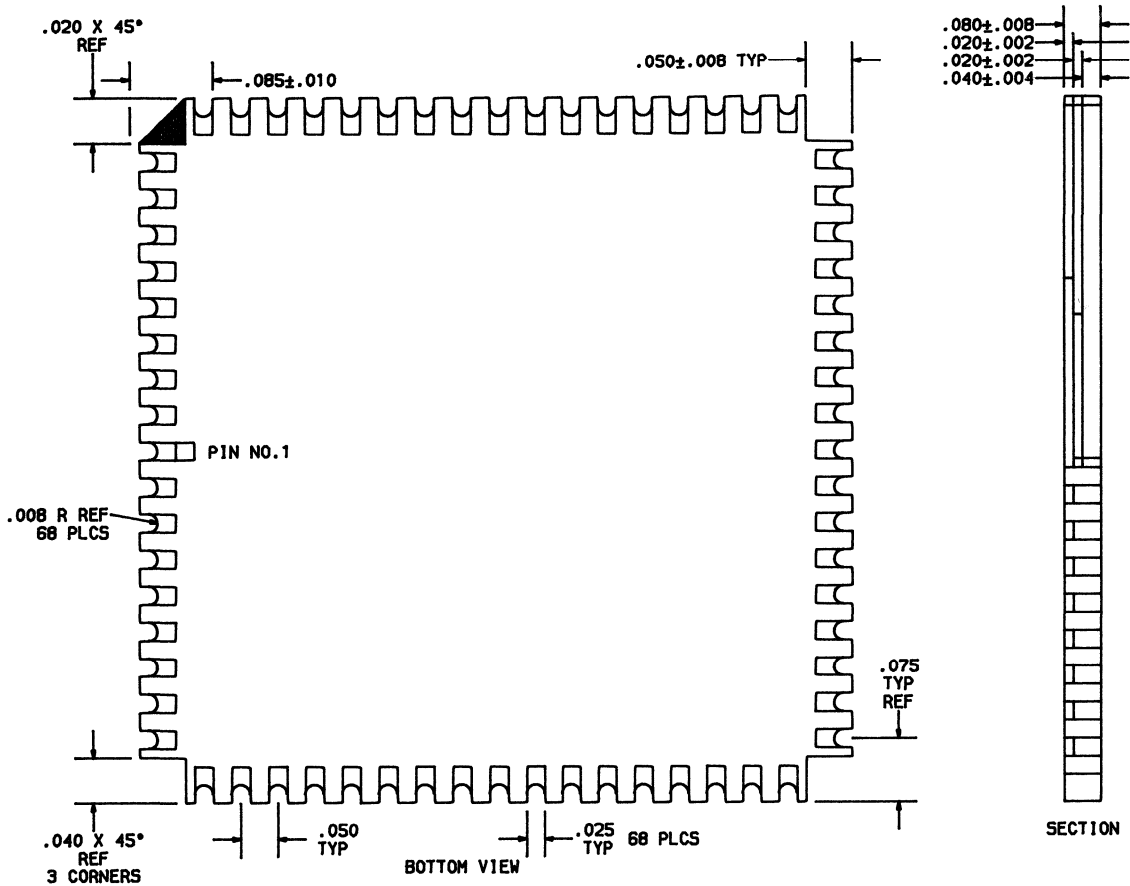


FIGURE 16. 84-PIN GRID ARRAY PACKAGE PINOUT.



| ORDERING INFORMATION | | | |
|----------------------|--------------|---|--------------|
| Speed | Package Type | Temperature Range | Order Number |
| 10 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33891JC-10 |
| 15 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33891JC-15 |
| 20 MHz | LCC | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33891JC-20 |

| ORDERING INFORMATION | | | |
|----------------------|--------------|--|--------------|
| Speed | Package Type | Temperature Range | Order Number |
| 10 MHz | PGA | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33891GC-10 |
| 10 MHz | PGA | $-55^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$ | ZR33891GM-10 |
| 15 MHz | PGA | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33891GC-15 |
| 15 MHz | PGA | $-55^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$ | ZR33891GM-15 |
| 20 MHz | PGA | $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ | ZR33891GC-20 |
| 18 MHz | PGA | $-55^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$ | ZR33891GM-18 |

DISTINCTIVE FEATURES

- 8 filter cells
- Up to 15 MHz sample rate
- 9-bit coefficients and signal data
- 21-bit accumulator per stage
- Expandable coefficient size, data size and filter length
- Decimation by 2, 3 or 4
- CMOS power dissipation characteristics

APPLICATIONS

- Low cost 1-D and 2-D FIR filters
- Analog filter replacement
- Radar/Sonar
- Digital video and audio
- Adaptive filters
- Echo cancellation
- Correlation/convolution
- Matrix multiplication
- Sample rate converters

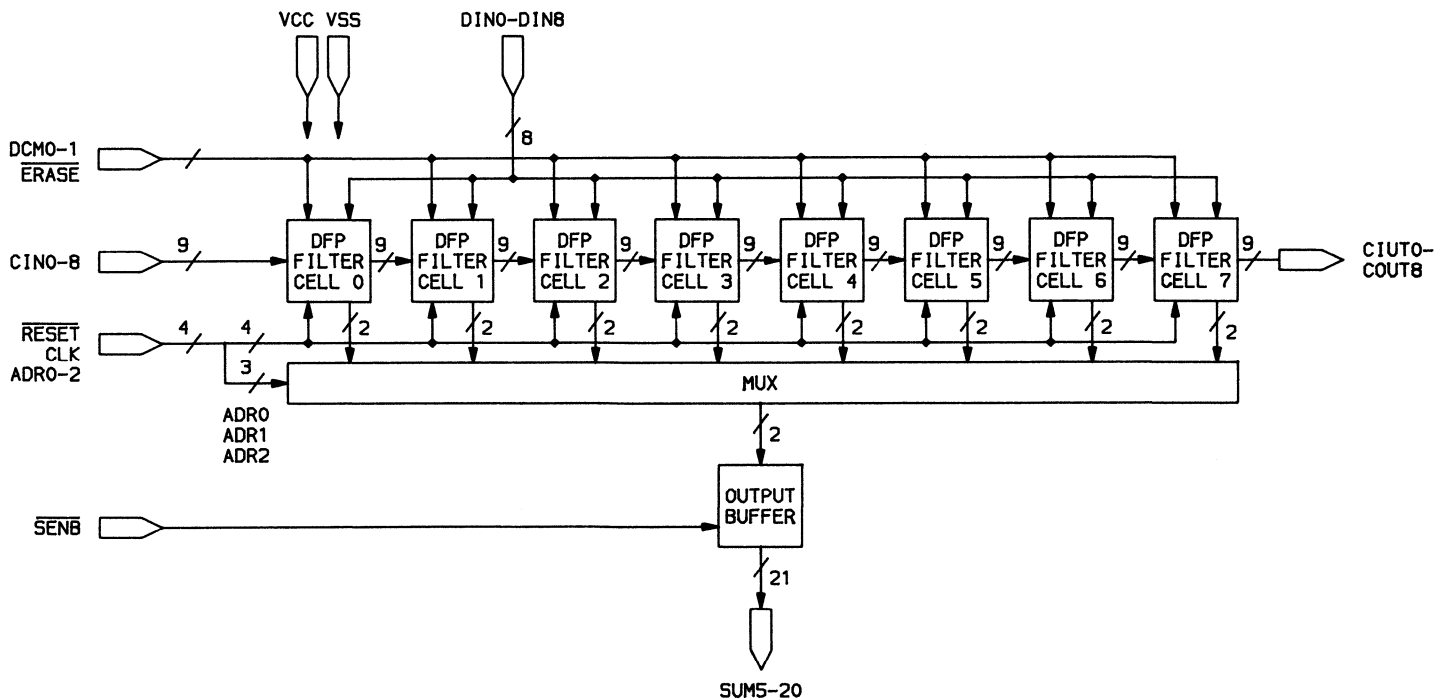
DESCRIPTION

The ZR33892 is a video-speed Digital Filter Processor (DFP) designed to efficiently implement vector operations such as FIR digital filters. It is comprised of eight filter cells cascaded internally. Each filter cell contains a 9×9 two's complement bit multiplier, three decimation registers and a 21-bit accumulator. The ZR33892 has a maximum sample rate of 15 MHz. The effective multiply-accumulate (MAC) rate is 120 MHz.

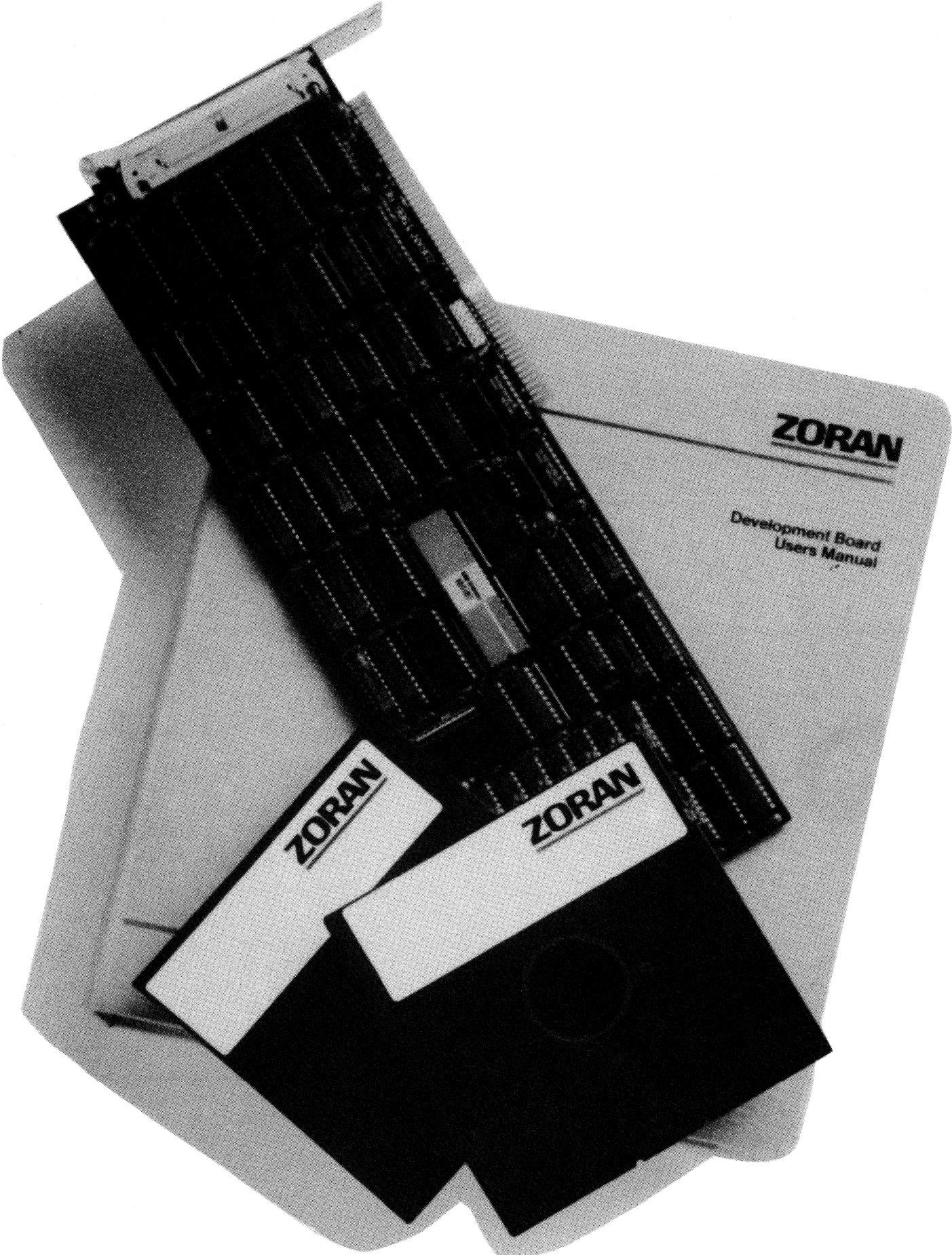
The ZR33892 DFP can be configured to process expanded coefficient and word sizes. Multiple DFPs can be cascaded for

larger filter lengths without degrading the sample rate or a single DFP can process larger filter lengths at less than 20 MHz with multiple passes. The DFP provides for 8-bit unsigned or 9-bit two's complement arithmetic, independently selectable for coefficients and signal data.

Each DFP filter cell contains three resampling or decimation registers which permit output sample rate reduction at rates of 1/2, 1/3 or 1/4 the input sample rate. These registers also provide the capability to perform 2-D operations such as matrix multiplication and $N \times N$ spatial correlations/convolutions for image processing applications.



DEVELOPMENT TOOLS



VECTOR SIGNAL PROCESSOR SUPPORT TOOL ENVIRONMENT

DEVELOPMENT ENVIRONMENT

The Vector Signal Processor Support Tool Environment, as shown in Figure 1, contains a combination of hardware and software packages that support everything from algorithm and program development to simulation, debugging and execution. Application libraries, interfaces to high-level languages, PROM formatting, signal generation and plotting are just some of the features provided. Support is provided for the VAX™ on both VMS™ and Ultrix™ operating systems, as well as the IBM PC™ on MS-DOS. Board level products provide accelerated simulation, execution and real-time data acquisition and processing. Interfacing to Hewlett Packard logic analyzer products is also available. These VSP Support Tools were developed to allow the system designer to easily transition from algorithm development to VSP programming, VSP simulation, and target hardware design.

The VSP Simulator is an excellent place to start for those new to the VSP. Through the instruction tutorial, the instruction set can be executed one instruction at a time in an interactive mode with extensive status and display options. The Simulator also provides a multiple VSP simulation capability, for users requiring more than one VSP in their system.

The initial problem to be tackled in a digital signal processing application is the development of a specific algorithm. The algorithm is normally developed in a high-level language and tested with real or simulated data. This is normally done using floating-point computations to eliminate the effects of rounding and overflow. After the algorithm's performance is considered satisfactory, it must be retested using the arithmetic precision of the target hardware (many times a full floating-point solution is neither fast enough or even necessary).

The Simulator is flexible enough to allow algorithm development independently of the VSP, with signal generation and plotting capabilities as well as a floating-point signal processing library. When the algorithm development is complete, selected portions of the high-level language code can be replaced with VSP assembly code, and tested against the floating-point results. The Simulator accomplishes this by "parsing" the VSP code from the program, assembling it, and relinking it to the initial program. From these results, a decision can be made as to the need for the VSP's block floating-point capability.

Some applications do not require algorithm development, just implementation of a tested algorithm on the VSP. VSP code development can be supported with the Simulator or a VSP Board Package and Assembler. The Board Packages are also well suited to accelerate lengthy simulations, because of the on-board VSP. The IBM/PC Board Packages are all provided with software that allows diagnostics, debugging, and VSP program execution from PC control. These Board Packages can be used in PC-based OEM applications as well.

To integrate the VSP all the way to a target system, the Board Packages can be utilized to compare results with the target system. Additionally, a Logic Analyzer Interface Module (VSPL) compatible with the HP1631D series of logic analyzers is available. Once target hardware has been designed and fabricated, the VSPL can provide extensive trace capability for debugging. Complete status of data and control lines coupled with a reverse assembly capability allow easy debug of target hardware problems. The Interface Module can monitor execution of the target system performance, with automated setup and trace features.

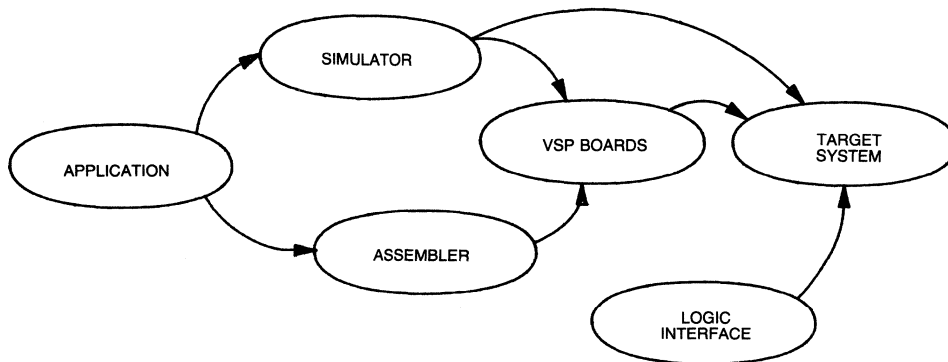


FIGURE 1. VSP SUPPORT TOOL DEVELOPMENT ENVIRONMENT.

DEVELOPMENT BOARD OPTIONS

As users proceed closer to their target system or their simulations become extensive, the VSP Boards, in combination with the Assembler, provide an alternative development path.

There are three separate IBM PC/XT/AT™ add-on board product packages that contain the ZR34161 Vector Signal Processor™, and software that supports communication and monitoring of the board's operation. These three packages are the XT Compatible Evaluation Board (VSPX), the Evaluation Board (VSPE) and the Development Board (VSPD). The VSP Boards are delivered with an Application Program Library and a Programmer's Toolkit. The Programmer's Toolkit consists of three separate software packages; the Board Interface Library, the Debugger, and the Diagnostics program.

The VSP boards can be utilized in three unique ways to develop and run VSP programs, as can be seen in Figure 2. In combination with the VSP Simulator (ZR63401), the VSP Evaluation Boards can host all the VSP operations performed within the Simulator, serving as a hardware accelerator. In combination with the VSP Assembler (ZR63411), the VSP Boards can run assembled programs in two separate modes.

The first utilizes the Debugger, where a program can be stepped through and examined in detail, utilizing the Debugger's signal generator and plotting routines. Verified programs can then be run utilizing The Board Interface Library, included with the boards, which allows application programs to be loaded and executed from PC program control, sharing data with user programs.

The VSPX is the lowest cost, entry-level board for evaluating the VSP ZR34161 for use in a potential design. The VSPE contains the same features contained in the VSPX but with the ability to communicate over the entire 16 bit AT data bus and the inclusion of a full speed, 20 MHz ZR34161. Although developed for evaluation of the ZR34161, both boards can be utilized as array processors for the PC in scientific, research and educational applications. With the addition of customizing software, the boards can also be utilized as OEM products.

The Development Board (VSPD), has all the features of the VSPE/X boards with additional features for real-time system implementations. The VSPD includes 64K words of fast RAM, high-speed parallel input and output ports that are buffered with 512 word FIFOs, hardware breakpoint capability and breakpoint tagging for FIFO delayed data.

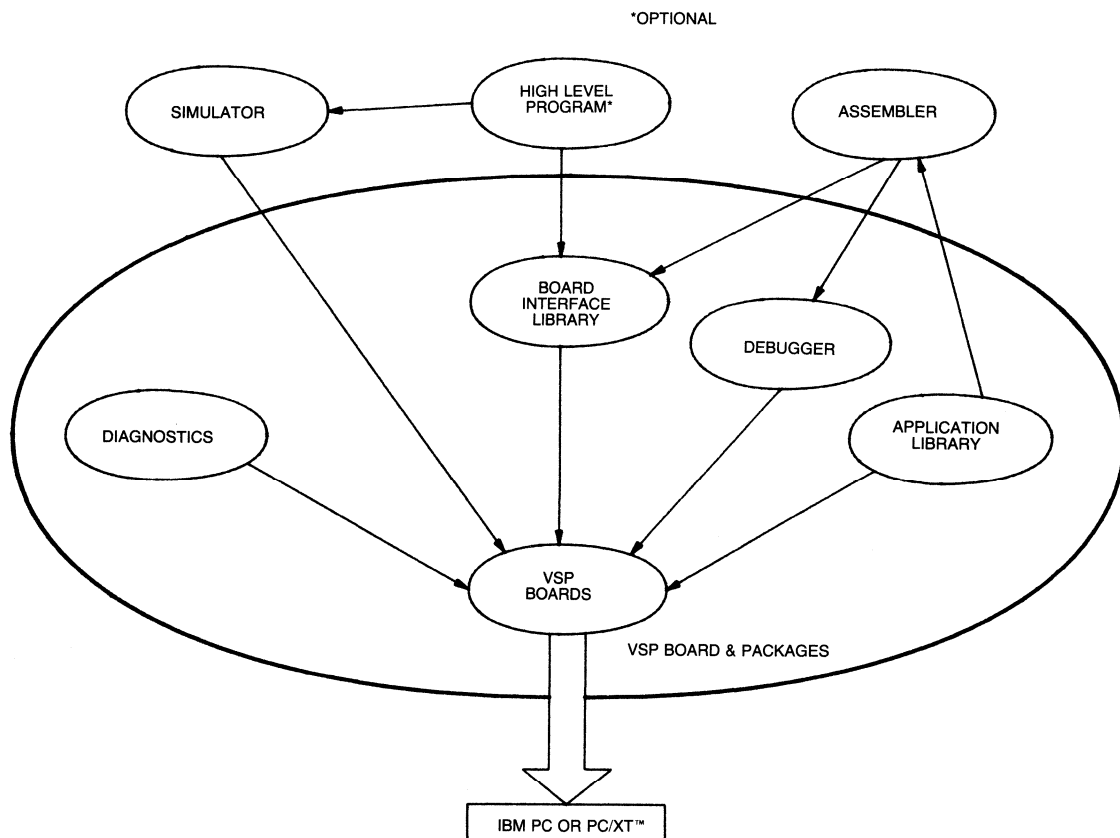


FIGURE 2. USING THE VSP BOARD PACKAGES.

VECTOR SIGNAL PROCESSOR ASSEMBLER (VSPA)

DESCRIPTION

Zoran's Vector Signal Processor Assembler (VSPA) is an engineering tool for developing complete Vector Signal Processor (VSP) code for applications. The function of the VSPA is to provide VSP language assembly, symbol table listings and Intel Hex code listings to support program debug and execution. The VSP Assembler converts VSP language programs created on the user's word processor into an Intel Hex code that can be loaded into VSP board products or used for PROM programming. The Assembler produces conventional listings of VSP instructions and their corresponding addresses in memory. It also accepts default declarations, permits symbolic address references and includes conventional assembler directives such as: ORG, EJECT, etc. An example program is shown below.

EXAMPLE VSP PROGRAM

Demonstration 128 Point FFT Routine for Zoran VSP161 Assembler.

```
org 0;

DEFAULT RS:0, EI:0, INTRO:0, ZP:0, AD:0, AS:0,
        FSIZ:128, RV:0, R:0, MDF:3, ZR:0;

DEFAULT FFT.I:0, FFT.RBA:0;

DEFAULT MBS:128, MSS:128, FPS:64, LPS:1;

LDASM NMPT:1, RS:0, MD:1, UP:0, MBA:MODEADD;

LD NMPT:128, MBA:IN;

FFT NMBT:128, R:0, I:0;

ST NMPT:128, RV:1, MBA:OUT, EI:1;

HLT;

MODEADD:dw 0x4870;
org 1000;
IN: dw0;
org $+255;
OUT:dw 0;
end;
```

EXPLANATION OF VSP PROGRAM

The "org 0" statement sets the program base address to 0.

The DEFAULT directive is specific to the VSP instruction format in which several parameter values may be specified. Often these parameter values are fixed for much or all of a program. With the DEFAULT directive, parameter values can be set for all instructions which require the parameter, or only for specific instructions.

The LDASM instruction loads the mode register with the desired operating configuration pointed to by "MODEADD".

All 128 complex data points are loaded into the VSP using the LD instruction.

A 128-point complex FFT is performed in the next instruction. The addresses of the transformed data are in bit-reversed order after the FFT.

The 128-point complex result is stored to external memory in normal order using the ST instruction.

The following pseudo-ops define the data spaces for the program:

MODEADD is the address at which the mode register value to be loaded is stored.

IN is the address at which the input data will be stored.

OUT is the address at which the results are stored.

HOST COMPUTER REQUIREMENTS

The VSP assembler is written to operate on an IBM PC, PC/XT or PC/AT under DOS version 3.1 or higher. Both the VSP Simulator (VSPS) and the Development Boards (VSPX, VSPE or VSPD) execute programs using the Assembler.

VECTOR SIGNAL PROCESSOR SIMULATOR (VSPS)

DISTINCTIVE FEATURES

• Complete VSP software engineering environment, including simulation of:

- VSP architecture and instruction set
- interaction of devices and events external to the VSP
- instruction timing and bus usage
- multiple VSPs on a bus
- complete VSP applications

- Tutorial on VSP instructions
- Sophisticated waveform generator
- Waveform plotting capability
- VSP and IEEE signal processing libraries
- Library of existing VSP applications
- Output files for burning PROMs
- Directly drives VSPX, VSPE, and VSPD boards
- Either menu or command driven
- Runs on VAX™ and IBM PC™ computers

DESCRIPTION

Zoran's Vector Signal Processor Simulator™ (VSPS™) is a powerful engineering tool for developing Vector Signal Processor™ (VSP™) applications. The function of the VSPS is to provide a software engineering environment which allows development and simulation of complete applications which utilize the VSP. The VSPS accurately simulates the processor while also allowing simulation of the environment external to the VSP. The system design support functions provided in the simulator are shown in Figure 3.

The VSP Simulator models the architecture, instruction set, execution timing, and many other internal features of the processor. The external memory space, memory speed, instruction and data fetching, bus usage, and other external features are also modeled to support simulation of complete applications.

The external environment is designed and programmed by the user in a high-level language, and may include a simulated host controller, microprocessor, or other devices or events. VSP instructions are embedded directly within this high-level language simulation of the complete system.

A number of unique tools are provided within the VSP simulator to simplify the development engineering task. On the front end, these tools allow the VSPS to be integrated into the design path beginning with the initial high-level language floating-point simulation of the application. On the back end, the simulator supports generation of VSP instructions in a number of formats compatible with PROM programming tools.

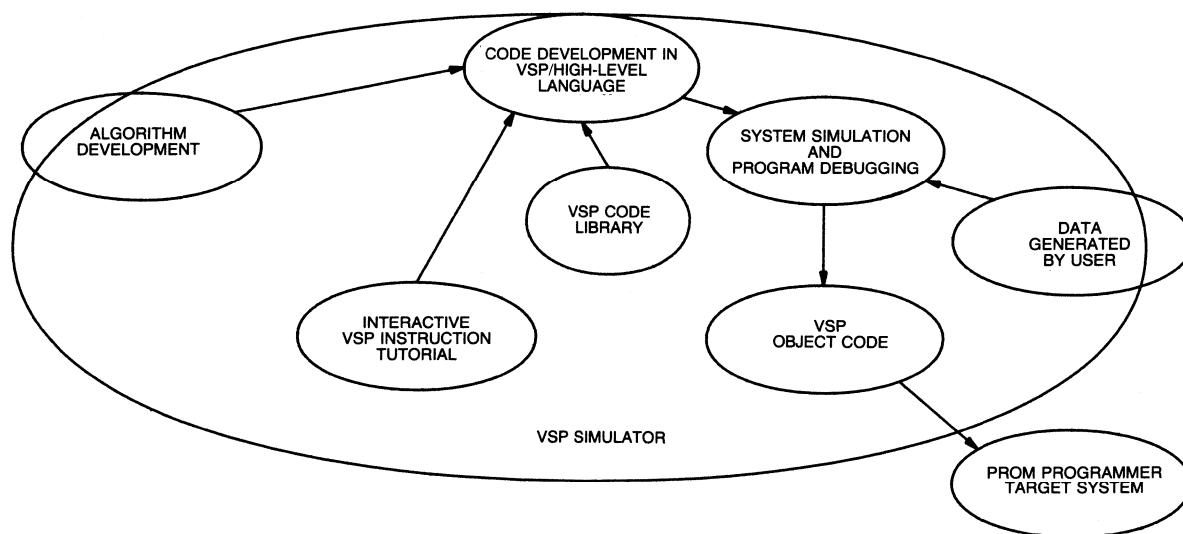


FIGURE 3. SYSTEM DEVELOPMENT FUNCTIONS SUPPORTED BY THE VSP SIMULATOR.

SIMULATOR ORGANIZATION

The VSP Simulator contains a number of useful tools for developing VSP applications ranging from the simplest to the most sophisticated. Control of the simulator is through either its tree-structured menus or through a macro-command interface which bypasses the menus. Figure 4 shows the structure of the menus.

Help (M-1): The Help menu provides a tutorial description of the simulator and the menu options. Within each menu additional Help options are also available.

VSP Instruction Tutorial (M-2): The instruction tutorial is an interactive mode whereby VSP instructions can be executed one-at-a-time. It is quite valuable for learning the instruction set and associated parameters. For instance, a signal may first be created in simulated system memory using the signal generator. The signal could then be loaded into VSP memory, arithmetic operations performed, and the results stored to external memory using instructions from the tutorial.

Data Generation and Display (M-3): Signal generation and plotting capabilities are accessed through this menu option. The VSPS signal generator can create sophisticated waveforms by adding, subtracting, multiplying, or dividing different types of signals with other signals. Individual component signals are sinusoids, square waves, ramps, flat or DC signals, and gaussian noise. For instance, multiplying two signals yields a modulated signal. Results of the signal generation can be plotted and modified if desired.

Display Options (M-4): Menu M-4 allows the user to choose the manner in which various simulator data is displayed. Examples of options controlled by this menu include data format (integer, floating-point, hexadecimal or binary) and bit numbering (whether bit 0 of a word is the LSB or MSB).

VSP Signal Processing Library (M-5): The VSP signal processing library is a collection of useful signal processing programs written using VSP instructions. These programs may be used in a number of different ways. First, they are available interactively within the simulator for performing canned signal processing operations on user-created data. Secondly, they can be dumped to a disk file where they can be edited directly into user VSP programs. Examples of the types of routines provided are: general FFTs up to 4096 points in length, real FFTs, magnitude functions, power spectra calculations, fast convolutions and correlations, and different types of complex vector operations. Users are encouraged to use these library routines in their own applications instead of writing their own programs.

IEEE Signal Processing Library (M-6): The IEEE signal processing library is a collection of floating-point routines which mirror those written using VSP instructions and described above. They are provided for comparing results obtained using integer or block floating-point arithmetic with that obtained using full floating-point. In fact, the differences between floating-point and integer arithmetic may be plotted using the waveform plotting utilities.

Applications Library (M-7): An applications library is included which contains complete VSP applications written by Zoran engineers. To date, this library consists of a 16K FFT and a quadrature-receiver doppler-shift application. Additional applications will be added over time (Not available on PC version).

VSP Program Control (M-8): This menu gives the user control over a wide range of options related to the execution of VSP programs. In addition to executing user-written VSP programs, this menu allows the user to set breakpoints and single-step through a VSP program, model various types of host/VSP interface protocols, perform VSP and BUS timing analyses, and control the operation of a VSPD or VSPE/X board. All the functions necessary to effectively debug VSP programs are accessible through this menu.

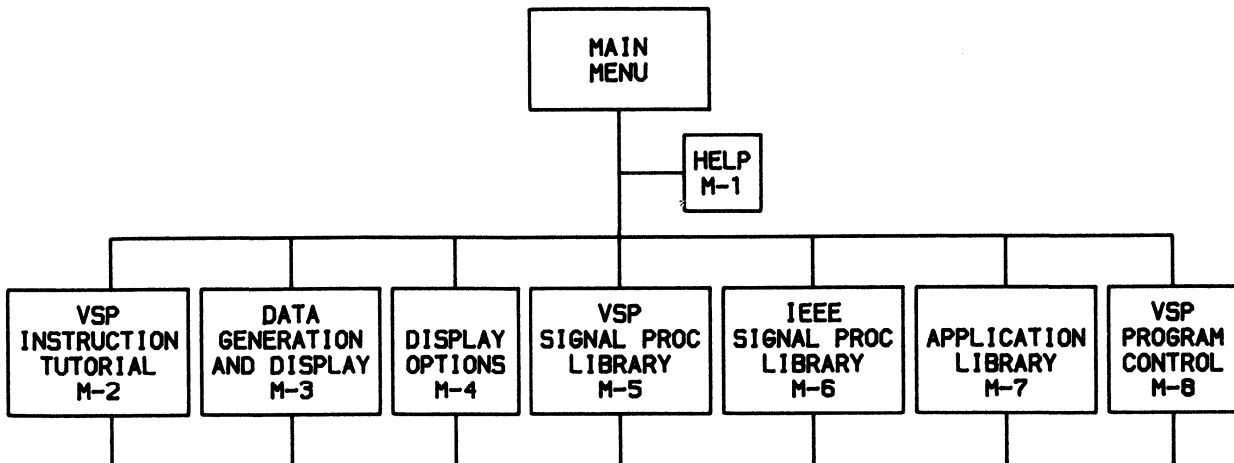


FIGURE 4. MENU AND FUNCTIONAL ORGANIZATION OF THE VSP SIMULATOR.

USING THE VSP SIMULATOR

The VSPS can be used in a number of different ways by users of varying experience for developing VSP applications. In its most simple form, the simulator allows users to interactively create signals and operate on them. Sophisticated users can create detailed VSP programs and simulate interaction with devices and events external to the VSP.

Getting Started

The VSPS supports new users of both the processor and simulator by providing a tutorial mode for executing instructions. In this mode, instructions can be interactively called, parameters customized as desired, then executed. As an example, the four instructions:

- LD (Load)
- FFT (Fast Fourier Transform)
- MGSQ (Magnitude-Square)
- ST (Store)

can easily perform a 128-point complex FFT and magnitude-square function on a 128-point real or complex data array using the tutorial. Prior to the execution of the simple program, the user can create a signal using the signal generator. The signal before, during, or after the program execution can be plotted as desired.

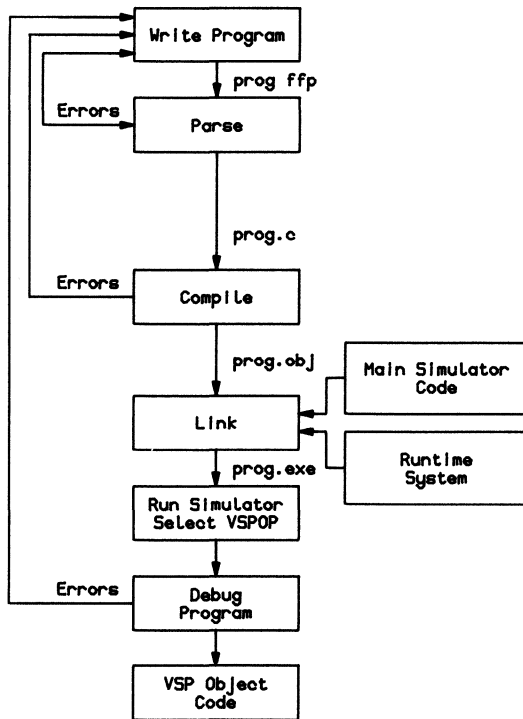


FIGURE 5. VSP CODE DEVELOPMENT CYCLE.

In a similar manner, programs from either of the two signal processing libraries can be executed on a data signal in memory.

Application Development

The power in using the simulator for application development is the ability to simulate not only the VSP, but also the external environment and host interaction. The external environment simulation is written by the user in C or a C-callable language such as Fortran or Pascal. Simulation of the VSP is done by directly embedding VSP instructions in the same source file. The process of creating an application is illustrated in Figure 4, while an example of a mixed-mode C/VSP program is shown in Figure 6.

Note that in Figure 6, VSP instructions are separated from the C instructions by the `'/#'` and `'#/'` delimiters. Code outside of the delimiters is standard C code which utilizes all the power of the particular C compiler used.

In Figure 5, the second step is to "parse" the mixed C/VSP program. The mixed-language file is input to the parser which generates an output file in a form compatible with the C com-

```

/* Example Program of mixed C/VSP code */

#include /* Additional VSPS variables */
"ffp.h"
#define EXTMEM zrmexter->exmem
#define pi 3.141592654
/*D#/* /* Parameter used by parser */
vspop() /* Start of 'vspop' function */
{
    int i, f;
    /* Define VSP instruction default values */
    DEFAULT RS:0, INTRP:0, EI:0, MBS:0, MSS:0;
    DEFAULT RV:0, MDF:2, ADF:2, ZR:0, ZP:0, I:0, R:0;
    DEFAULT FPS:64, LPS:1, RBA:0, FSIZ:128, AS:0;

    /* Start of the executable code */
    /* 11-cycle sinewave generation in C */
    f=11;
    for (i=0;i<128;i++)
        EXTMEM=16383 * sin(2*pi*f*i/128);

    /* VSP code for LD, FFT, MGSQ, ST */
    LD    NMPT:128, MBA:0;
    FFT   NMPT:128;
    MGSQ  NMPT:128;
    Sb    NMPT:128, RV:1, MBA:256;

    /* End of VSP code */
} /* End of 'vspop' function */
## /* End of program */
    
```

FIGURE 6. EXAMPLE OF A MIXED C/VSP PROGRAM.

HOST COMPUTER REQUIREMENTS

| | | | |
|--------------------------|---------------|----------------|--------------------------------------|
| Computer: | VAX | VAX | PC/XT, PC/AT |
| Operating System: | Ultrix (V1.2) | VMS (V4.4) | DOS (V3.1) |
| Compilers: | C | C (V2.2) | Microsoft C (V4.0) |
| | Fortran | Fortran (V4.5) | Microsoft Fortran (V3.31 - optional) |
| Memory: | 1M | 1M | 640K |
| Display: | VT240 | VT240 | CGA, EGA |

FEATURE VARIATIONS ON DIFFERENT COMPUTERS

| | | |
|--|------------|------------------------|
| Computer: | VAX | PC/XT, PC/AT |
| VSP Program/data memory: | 64K words | 16K words |
| Maximum IEEE FFT size: | 8K points | 1K points |
| Number of simulated VSPs: | 8 | 2 |
| Application Library: | Yes | Not in current release |
| VSPX, VSPE, VSPD board support: | No | Yes |

piler. This file is compiled to create a module which in turn is linked to the main body of the VSP simulator. The resulting executable program is a complete version of the simulator with all of the tools previously discussed linked with the user program. Signals can be created as discussed previously, and the user program can operate on these signals.

A number of utilities are provided in the simulator to help debug VSP programs such as: illegal bounds checking of VSP instruction parameters, single-stepping instructions, displaying internal VSP operations on a clock-by-clock basis, and plotting internal or external memory. When a user is satisfied with a program, it may be dumped to a disk file in one of two different formats: Intel Hex or Jedec. These files are compatible with many industry-standard programming tools.

ORDERING INFORMATION

| <u>Version</u> | <u>Order Number</u> |
|----------------|---------------------|
| PC (DOS) | ZR63401 |
| VAX (VMS) | ZR63402 |
| VAX (Ultrix) | ZR63403 |

VSP BOARD SUPPORT

The PC version of the VSPS is accompanied by a set of software drivers which allow it to directly control PC compatible hardware development boards (VSPX, VSPE, VSPD) provided by Zoran. When the boards are used, they act as accelerators for the simulator while maintaining the full capabilities of the VSPS already discussed. The development boards discussed in greater detail in the VSP board sections.

UPDATES

Zoran notifies all VSPS customers when a new simulator release is available. Customers who have purchased software within a period of one year are eligible for a free update. Customers who have purchased software over a year ago may update software at a nominal charge.

VSP BOARD PACKAGES

The VSP board packages are a combination of hardware and software that allows the user to run high-speed digital signal processing simulations on the PC/AT. With a minimum of time and effort, the user can be running VSP programs, examining intermediate results with breakpoints and linking these VSP programs to user data contained on the IBM PC.

The VSPD board package expands this capability even further for users who need real-time program execution with data not available on the PC. High-speed external parallel ports coupled

with the VSP DMA capability allow the VSPD to function as a data acquisition and DSP processor board in one. Routines allow this data to then be stored on IBM PC peripherals.

The packages consist of a PC add-on board, Application Program Library, Programmer's Toolkit, and User's Manual. These packages will allow VSP programs in the Application Program Library to be downloaded and executed on the boards by utilizing the Programmer's Toolkit programs. The Programmer's Toolkit contains Diagnostics, a Debugger and a "C" callable Board Interface Library.

VSP BOARD SOFTWARE

The aspect of the board packages that makes them so powerful and easy to use is the software that is shipped with each board. This software gives you a multitude of ways to use the boards. Total control over board functions are provided with the software. These software tools are useful for taking a current DSP algorithm and converting the computational portions to high-speed VSP code. Let's take a closer look at what is provided.

VSP-161 Application Program Library

A library of example programs for the VSP is included to let you quickly explore the power of the VSP device. This library includes, for example, FFT's of various sizes and a program to compute the power spectrum of a signal. These programs are provided in hex object code form, and can be loaded and run directly through the debugger or incorporated directly into custom application programs. Source code for the routines are also provided to demonstrate coding techniques for the VSP, and to allow modification or combination of these basic programs for your particular application. The optional ASM-161 Assembler can be used to assemble modified source code into loadable and executable object code.

VSP-161 Programmers Tool Kit

Board-Interface Library

This library of board interface utilities provides a set of low-level interface routines that simplify programming the VSP boards. This allows users to write C programs with the board interface utilities embedded in the program. Communication to,

and execution of, VSP programs is then transparent to the PC user. Included are functions which reset the board, load VSP programs or data onto the board, initiate VSP program execution, and wait for an interrupt from the VSP signaling that the VSP program is complete. The routines are written in C and object modules compatible with the Microsoft™ Linker are provided. Source code is also provided to serve as a guide to writing interface routines in other languages.

Debugger

The debugger provides the conventional debugging capabilities of examining and setting memory or VSP registers, and executing programs by single stepping, running with breakpoints, or free running. The debugger, in addition to these features, also includes a powerful macro command feature which lets you construct complex commands from sequences of basic commands, reducing keystrokes for repetitive tracing of results. The Debugger provides a signal generation capability for easily creating test data. This test data can be displayed before and after processing using the resident plotting feature.

Diagnostics

The diagnostic routines provided are simple to use and ensure that the supplied board is functioning correctly. The diagnostics are menu driven and test the board RAM, STATUS, and MODE registers. Additionally, the routines test the internal VSP RAM, ROM and SCALE registers. The routines can perform a functional test from a PC file.

VSP BOARD APPLICATIONS

Optional Software Selection

Assembling VSP instructions into an executable format requires use of the VSP Simulator or the VSP Assembler. These tools, in combination with the VSP Boards provide a variety of development options for the user. Below is a brief description of the use of the VSP Boards using the features of the software support tools.

VSP Simulator (ZR63401)

Prior to the execution of the simple program shown in the Assembler description, the user could have generated a signal using the Simulator signal generator and placed that signal in external Memory Base Address 1000. The original signal, or the spectrum resulting after the program execution, can then be plotted as desired. Integrating the VSP Board to the Simulator environment will speed these simulations by utilizing the actual VSP on the VSP board. This integrated mode provides the powerful combination of the Simulator signal generation and plotting features with the high-speed simulation capability of the VSP Board. For example, a simulation running a 1024-point FFT on the simulator alone would require about thirty seconds to execute, but linked to the board, would be executed in the time it takes to make a keystroke. The power in using the Simulator for application development is the ability to simulate not only the VSP, but also the external environment and host interaction. The external environment simulation is written by the user in C or a C-callable language such as Fortran or Pascal. Simulation of the VSP is done by directly embedding VSP instructions in the same source file. The VSP instructions are then executed on the VSP Board and the host commands executed on the PC.

VSP Assembler (ZR63411)

The VSP Assembler converts VSP language programs, such as the one shown earlier, into an Intel Hex code format that can be loaded to the VSP Board in one of two ways (or used for PROM programming). The first of these is with the Debugger, a VSP Board supplied program that allows interactive testing of the assembled program for accuracy. The Assembler symbol table listing and the Debugger signal generation, plotting, and breakpoint routines provide complete and simple debugging. After the VSP program has been debugged, this program can be utilized a second way. The Board Interface Library provides a powerful method for executing the VSP program from a high-level language program running on the PC. The Interface routines allow a host program to load, execute, and pass data to the VSP program running on the VSP Board under the control of the PC.

High-Level Language Programs

Both the Simulator and the Boards allow linking to a high level program. This is a notable feature in that a development environment is provided to code a complete solution in one program. The interfaces provided utilize the Microsoft™ “C” compiler, Version 4.0. The Board Interface Libraries are written in “C”, and the source code for these are provided as a reference for interfacing to other high-level languages. A high-level language is not required for use of the VSP Boards, Simulator, or Assembler.

VSP BOARD HARDWARE

Contained here is a brief description of the hardware elements of each of the IBM PC/XT/AT packages.

VSPE HARDWARE

The VSPE board, shown in Figure 7, utilizes the IBM PC bus to obtain data, process it, and then return the results to the PC for storage, display or further calculations. Data acquisition boards for the IBM PC can be used in combination with the VSPE to provide a complete signal processing system. The VSPE is finding use in scientific applications where large sets of data must be efficiently processed, such as in radar, sonar, voice, seismology, and many other fields.

The VSPE board is also well equipped to function as a development testbed. It can be used for designers who eventually want to integrate the VSP processor with a target system. The Debugger and the Board Interface Library allows the designer to test the VSP program as well as it's interaction with a host's code before prototyping. To see what makes up the VSPE, let's look at the hardware features that it provides.

The VSPE hardware consists of a 20MHz ZR34161 Vector Signal Processor, 16K words of local memory, a mode control register, a status register, and PC bus-interface and arbitration circuitry between the PC bus and the VSPE board local bus as shown in Figure 8.

The board uses a 16-bit data bus and is driven by either the PC system clock or by an on-board 20MHz oscillator that is jumper selectable. The performance of the VSPE will depend on the clock rate selected as well as the transfer rate of the PC bus. The VSPE will function in an IBM PC/AT, but data transfers will be restricted to the 16 bit bus. A “wait” state generator is provided to assure proper timing for VSP addressing.

The mode and status registers are connected to the least significant byte of the data bus. The purpose of these two registers is to either permit host control of the board or read what activity is occurring on the VSPE. These registers are I/O-mapped to a user configured I/O base address, (factory set to 300H/301H).

The board is memory-mapped into two 32K byte segments of the address bus interface. The board base address and I/O address are jumper selectable, thus permitting multiple boards to be installed in the same machine.

The IBM PC/AT may send data and/or programs to the board, read data from the board memory, read the board status register or read the internally addressable VSP registers and memory. The VSP can read or write the local memory and send an interrupt request to the host.

There are three resets: a manual reset button, a PC-controllable reset line and a programmable reset within the VSP. Under software control, the VSP can interrupt the host. If the host wants control of the VSP local bus, it sends a host request to the mode register, which will force the VSP off the bus.

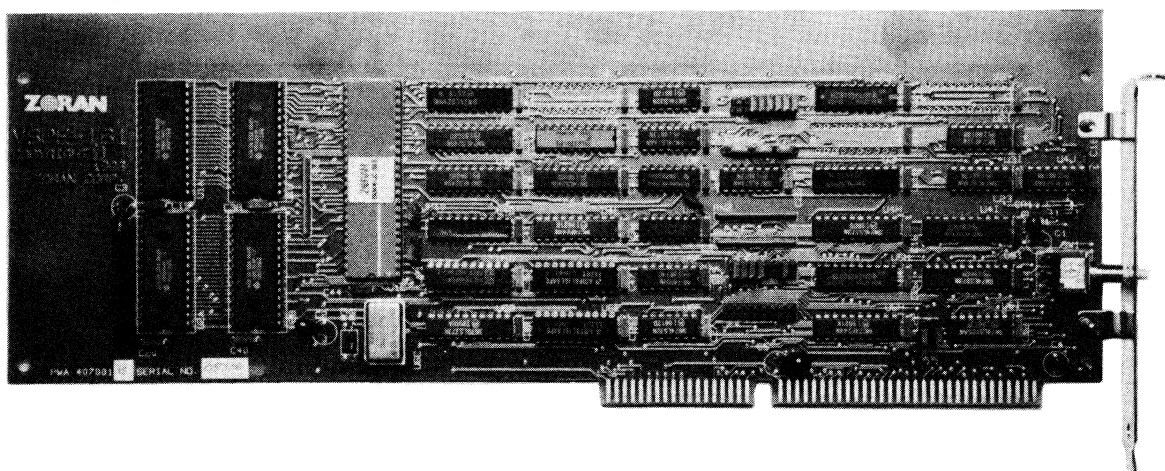


FIGURE 7. VSPE BOARD.

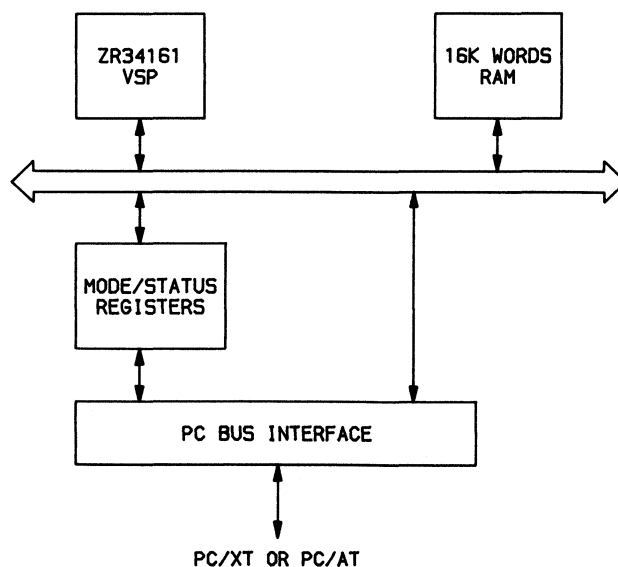


FIGURE 8. VSPX/VSPE BOARD BLOCK DIAGRAM.

VSPX HARDWARE

The VSPX, shown in Figure 9, is the same as the VSPE with the exception of the ZR34161 processor speed and the PC/XT bus interface. The block diagram is the same as is shown in Figure 8. The VSPX hardware consists of a 10 MHz ZR34161 Vector Signal Processor, 16K words of local memory, a mode control register, a status register, and PC bus-interface and

arbitration circuitry between the PC/XT bus and the VSPX board local bus.

The board uses the 8-bit data bus and is driven by either the 6/8 MHz system clock or by an on-board 10MHz oscillator that is jumper selectable. The performance of the VSPX will depend on the clock rate selected as well as the transfer rate of the PC bus. A "wait" state generator is provided to assure proper timing for VSP addressing.

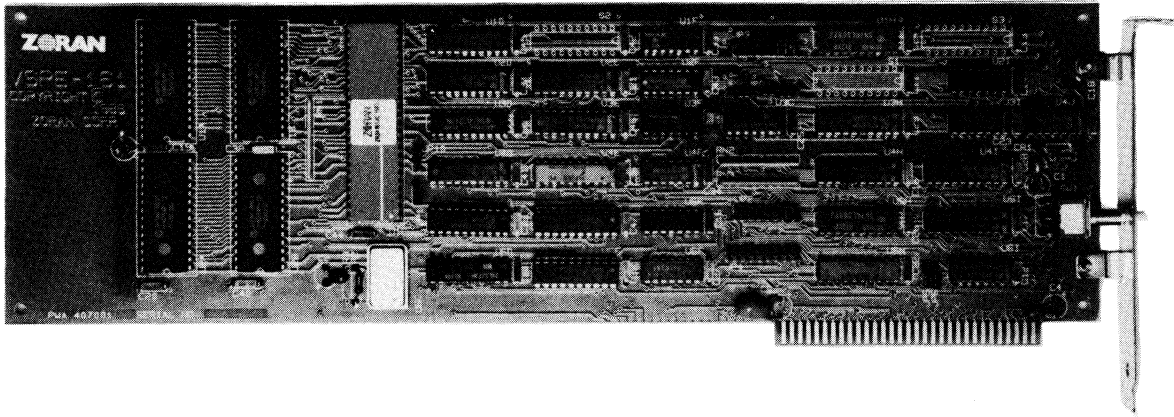


FIGURE 9. VSPX BOARD.

VSPD HARDWARE

The VSPD, shown in Figure 10, was developed for users who need to process real-time data that would exceed the bandwidth available on the IBM PC bus. The VSPD eliminates that bottleneck with external input and output parallel ports. These ports function completely independent of the PC, not requiring the PC's CPU or bus to collect or transfer data. Data acquisition using the VSPD board can be accomplished by the design of external analog-to-digital and digital-to-analog conversion hardware which interfaces to the external parallel ports. Alternatively, other acquisition systems which have provisions for a high-speed parallel bus may be interfaced to the VSPD ports.

The VSPD provides another important feature as well for real-time users, a hardware breakpoint capability. The hardware breakpoints are a reliable tool for program debug. In addition to conventional address breakpoints, the board allows external data sources or sinks the ability to initiate breakpoints. The external breakpoint can execute immediately or be tagged in the FIFO buffer with other data so that it is executed when the tagged data word is read from the FIFO. The VSPD board also provides an extensive interrupt capability for interrupt-driven programs. To see the capabilities of the VSPD, let's discuss the hardware.

The VSPD hardware is provided on a printed circuit card which occupies one slot in the PC/AT. It includes a 20MHz Vector Signal Processor device as well as a 64K x 16 bit array of high-speed (45ns) RAM for VSP program and data storage. Also included are two 16-bit parallel ports, one for input and one for output, to interface to high-speed external data sources and destinations. These ports are each buffered by FIFOs which are 512 words deep. Both the RAM and the parallel ports provide dual-port access from the VSP and the PC/AT. The VSPD hardware also contains a mode control register, a status register, PC bus-interface and arbitration circuitry between the PC bus and the VSPD board. A block diagram of the VSPD board is shown in Figure 11.

The board uses the full 16-bit data bus and is driven by either the 6/8 MHz system clock or by an on-board 20MHz oscillator that is jumper selectable. The performance of the VSPD will depend on the clock rate selected. Parallel port transfers can burst at up to 20 Mbytes/sec. A "wait" state generator is provided to assure proper timing for VSP addressing.

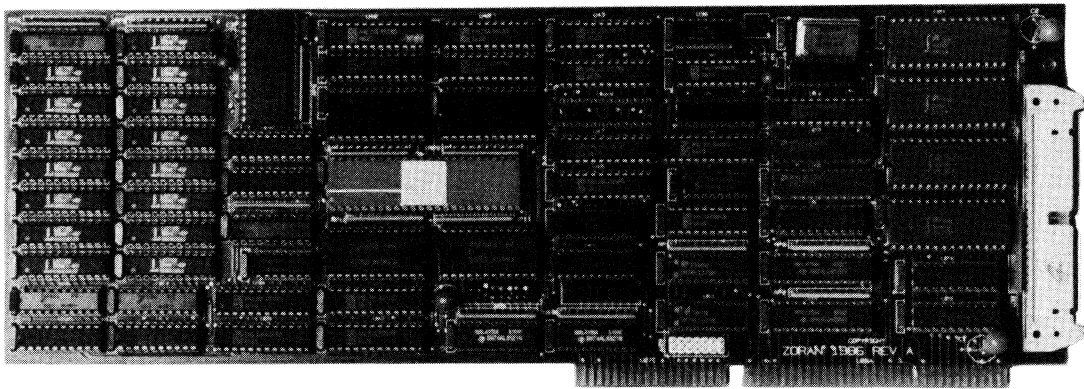


FIGURE 10. VSPD BOARD.

The VSPD includes four control ports that are mapped into the AT's I/O space: a configuration port, an operation control port, a strobe and status port, and a VSP bus activity port. These I/O ports are accessed only by the AT. The AT does not require access to the internal board bus to access these registers, so reading or writing them will not affect the VSPD's performance. Unlike the board's memory-mapped resources, these ports cannot be disabled.

When the board is in operation, the IBM PC/AT may send data and/or programs to the board, read data from the board memory, read any of the four control ports or read the internally addressable VSP registers and memory. The VSP, on the other hand, can only read or write the local memory.

The four I/O ports are used for board configuration and operation control. Setting bits in the configuration port allows internal VSP RAM/registers and on-board RAM to be addressed in a single 64K byte segment of the PC/AT address space. The operation control port can specify breakpoint generation—as specified in the breakpoint register—or on an external event such as reading a tagged word in the input FIFO.

Real-time I/O is accomplished with the dual 512-word FIFOs. These FIFOs are mapped into the data RAM address space of the VSP so that vector transfers may be made directly to and from the external system. The FIFOs are also mapped to the PC/AT address space so that their contents may be set and read during program debugging, for example.

The on-board breakpoint register is set from the PC/AT. When the VSP reads or writes to the specified address, the VSP is stopped by denying it access to the bus, and interrupt signals go to the PC/AT and the external connector.

VSP activity is resumed after a PC/AT write to the strobe register port. Externally initiated breakpoints come from tagged data in the input FIFO, or from the IRQBRK pin on the external connector.

The board is memory-mapped into a 64K byte segment of the 16Mbyte address space of the 24-bit address bus interface. The board base address and I/O address are jumper selectable, thus permitting multiple boards to be installed in the same machine. The memory arbitration scheme for the VSPD is completely implemented in hardware. This arbitration differs from the scheme used on the VSPE, which requires a specific software protocol. The VSPD board differs from the VSPE/X by allowing the simultaneous access by both the VSP and the PC/AT.

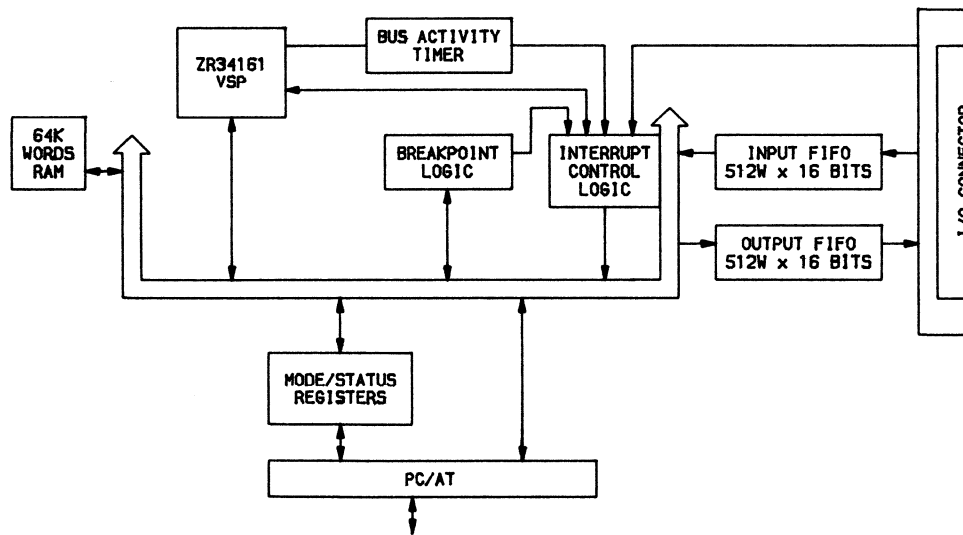


FIGURE 11. BLOCK DIAGRAM OF THE VSPD BOARD.

HOST COMPUTER REQUIREMENTS

Computer: PC/XT or PC/AT 6/8MHz (VSPX), PC/AT(VSPE, VSPD)
Operating System: DOS (V3.1)
 (Compilers are required to use the Interface Libraries)
Compilers: Microsoft C (V4.0)
 Microsoft Fortran (V3.31-optional)
Memory: 256K (640K when linked to Simulator)
Display: CGA or EGA
 (plotting will not operate with Hercules graphics)

FEATURE VARIATIONS ON DIFFERENT VSP BOARDS

| VSP Board: | VSPX | VSPE | VSPD |
|----------------------|-------------|-------------|------------------------------|
| Part Number: | ZR73412 | ZR73411 | ZR73401 |
| Computer: | PC/XT/AT | PC/AT | PC/AT |
| Board memory: | 16K words | 16K words | 64K words |
| Memory speed: | 120nsec | 120nsec | 45nsec |
| VSP Speed: | 10MHz | 20MHz | 20MHz |
| Data I/O: | PC/XT bus | PC/AT bus | PC/AT bus 10MHz I/O ports |
| Breakpoints: | Software | Software | Software/ Hardware |

Programmers Toolkit: Supplied with all three boards

VSP LOGIC ANALYZER INTERFACE MODULE

The ZR73420 VSP Logic Analyzer Interface Module (VSPL), seen in Figure 12, for the HP163XX logic analyzers provides a convenient interface between the target system and the analyzer. The VSPL, in combination with the appropriate HP interface modules, provides full access to all control and I/O lines on the VSP. The connection to the target system is simplified through the use of the low profile 48-pin DIP probe/socket.

The VSPL is installed into a Hewlett Packard 10269B General Purpose Preprocessor that contains interfacing circuits to the analyzer. The HP10269B preprocessor is then attached to the logic analyzer using the logic probes. The VSPL will not function with the HP163xA versions of the logic analyzer because of insufficient channels.

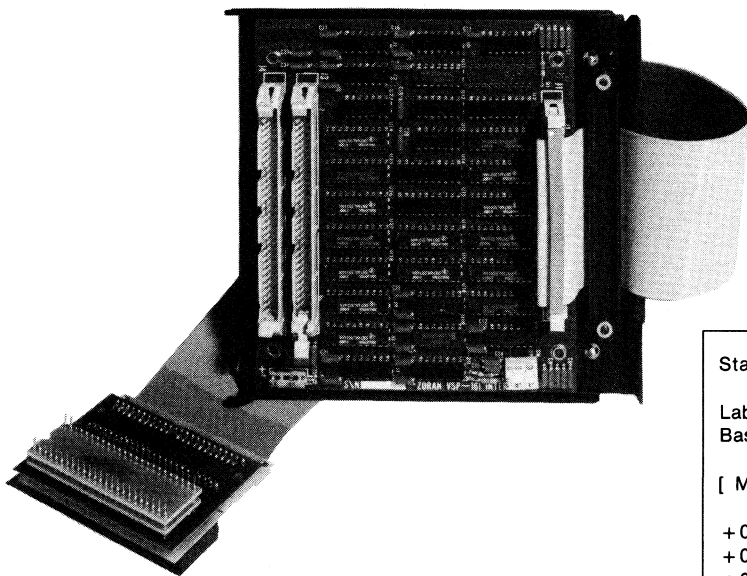


FIGURE 12. INTERFACE MODULE.

The inverse assembler software supports the entire ZR34161 instruction set and automates configuration of the analyzer, as shown in Figure 13. Address, data, and status are available during disassembly. Also a raw data capture mode is available where the state of each VSP address/data and control line is captured and displayed every clock.

The Interface Module allows tracing the VSP operation up to 20Mhz. All VSP signals are loaded with one ALS TTL input with an additional 30pF of probe cable capacitance.

The VSPL includes the hardware interface board, interface cable with low profile 48-pin ZIF probe/socket, disassembler/configuration software and sample displays on a 3.5 in floppy diskette and Users Guide.

| State Listing _____ INSERT to inverse-assemble _____ | | | | | |
|--|-------|-------|-------------------------|-----------------------------------|-----|
| Label > | ADDR | DAT | ZORAN VSP-161 Mnemonics | | <: |
| Base > | [HEX] | [HEX] | [| ASM |] |
| [Mark] | XXXX | XXXX | [| Instructions |] |
| +0000 | 0306 | 0000 | | Instruction base/start register | HWR |
| +0001* | 0000 | 0010 | LDSM | NMPT = 001 RS = 0 EI = 0 | VOP |
| +0002* | 0001 | A170 | | UP = 0 MD = 1 | VOP |
| +0003* | 0002 | 0035 | | MBA = 0035H | VOP |
| +0004* | 0003 | 6810 | NOP | NMPT = 001 RS = 0 EI = 0 | VOP |
| +0005* | 0004 | 0010 | LD | NMPT = 001 RS = 0 INT = 00 EI = 0 | VOP |
| +0006* | 0005 | E060 | MBS111 | MSS000 RV00 AD0 MDF11 ZR0 ZP0 | VOP |
| +0007* | 0006 | 0800 | | MBA = 0800H | VOP |
| +0008* | 0007 | 0810 | ST | NMPT = 001 RS = 0 EI = 0 | VOP |
| +0009 | 0035 | 4870 | | data | VRD |
| +0010* | 0008 | E060 | MBS111 | MSS000 RV00 AD0 MDF11 | VOP |
| +0011* | 0009 | 03E8 | | MBA = 03E8H | VOP |
| +0012* | 000A | 0820 | STB | NMPT = 002 RS = 0 EI = 0 | VOP |
| +0013* | 000B | E068 | MBS111 | MSS000 RV00 AD0 MDF11 | VOP |
| +0014* | 000C | F7F1 | | MBAB = F7F1H | VOP |
| +0015* | 000D | 0810 | STI | NMPT = 001 RS = 0 EI = 0 | VOP |

FIGURE 13. DISASSEMBLY LISTING.

ORDER INFORMATION

Development Tool: HP1630D,G,&HP1631D

Part Number: ZR73420

Required Options: HP10269B

DISTINCTIVE FEATURES

- Complete 20 MHz, 4 to 128 tap digital filter system
- 8 or 16 bit input data and/or coefficients
- 16 or 22 bit output data
- Separate 2K memories for input signal, filtered output signal and filter coefficients
- Up to 20 MHz operation with ZR33481 or ZR33881 Filter Processors
- Operates stand alone or interfaces directly with PC/AT or XT bus
- Selectable parallel data I/O ports
 - 8 bit PC bus
 - 8/16 bit external connector
- Eight selectable filter modes with decimation by 2, 3 or 4
- Jumper selectable on-board oscillator or external clocking source
- Filter coefficients can be defined with ZORAN's "DFPS" filter development software
- Provides real-time performance verification

DESCRIPTION

The ZR73301 Digital Filter Board (DFPB) is a complete 20 MHz digital filter system including sequence control and 2K signal buffer memories for continuous flow through operation. The system is designed to operate as a stand alone filter or operate under software control of a PC/AT or XT with ZORAN's Board Utility Software "(DFPBU)".

The DFPB is useful as a high performance prototyping tool for filters and correlators using ZORAN's Digital Filter Processors. The DFPB will verify filter operation in real time of coefficients defined with ZORAN's Digital Filter Development Software (DFPS) or coefficients obtained from any other source.

FUNCTIONAL DESCRIPTION

The board contains six functional subsections: PC interface, signal memory, coefficient memory, sequence control, filter processor(s) and output control. The PC interface contains clock circuitry and port control. Signal memory consists of a $2K \times 8$ input memory and $2K \times 16$ output memory. All coefficients are stored in a $2K \times 8$ memory (RAM or PROM) segmented into 16 sections of 128 bytes each. The sequencer contains the memory address and DFP address control required for the eight filter modes (see board setup). The output section selects the 16 bits to be saved from the 22 LSB DFP outputs, sends them to the P3 output port and stores them in local board memory. The local output memory may also be accessed by the P1 PC port to obtain the filtered results.

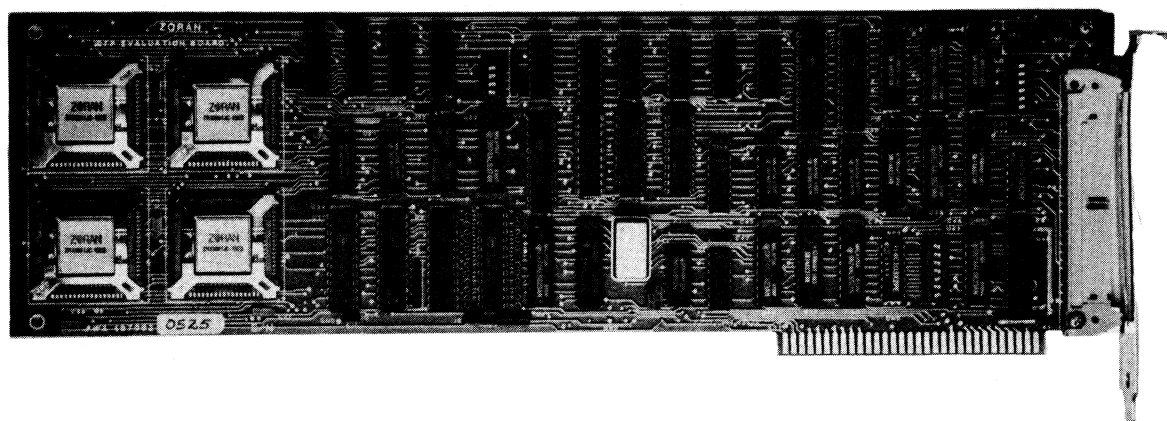
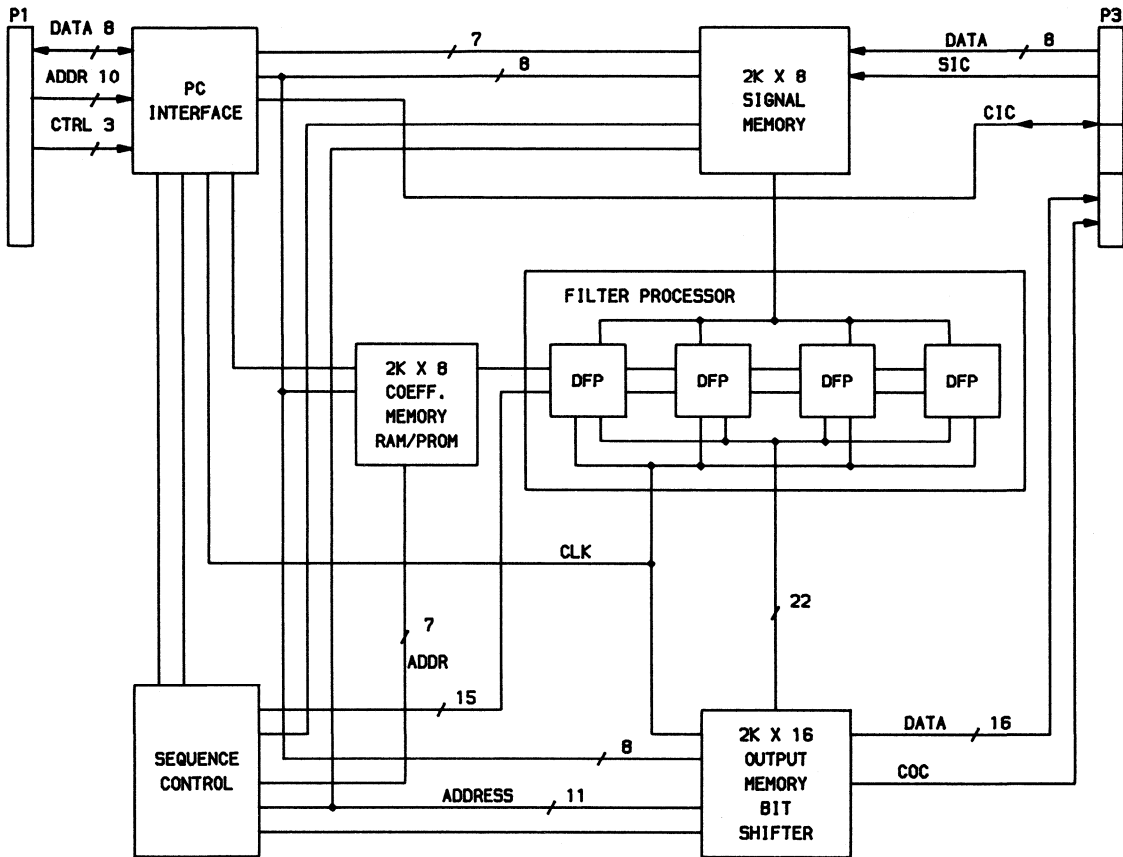


FIGURE 1. ZR3301 DIGITAL FILTER BOARD.

BLOCK DIAGRAM



BOARD SETUP PARAMETERS AND OPTIONS

SAMPLE RATE:

External Clock—.2 to 20 MHz

On-board Clock—14.3182 supplied by ZORAN—other frequencies selected by oscillator module replacement.

PRECISION AND DECIMATION OPTIONS:

| <u>DATA</u> | <u>COEFF.</u> | <u>DECIMATION</u> | <u>ALGORITHM NUMBER</u> |
|-------------|---------------|-------------------|-------------------------|
| 8-BIT | 8-BIT | 1 | 0 |
| 8-BIT | 8-BIT | 2 | 1 |
| 8-BIT | 8-BIT | 3 | 2 |
| 8-BIT | 8-BIT | 4 | 3 |
| 8-BIT | 16-BIT | 1 | 4 |
| 16-BIT | 8-BIT | 1 | 5 |
| 8-BIT | 16-BIT | 2 | 6 |
| 8-BIT | 16-BIT | 4 | 7 |

All data and coefficients must be two's complement format. Maximum filter length is determined by quantity and cell type (4 or 8) of DFPs and decimation rate. A single jumper selects the type of DFPs on-board. All data/coefficient/decimation combinations in the above table are implemented with a single PROM set for a specific number and type of DFPs in the system. A set of three proms controls the sequencing for the memories and DFPs. ZORAN supplies preprogrammed PROMS for two board options: four 4-cell devices (ZR33481) or four 8-cell devices (ZR33881). The user can create his own sequence PROMs for other configurations.

NUMBER OF TAPS:

With ZORAN Supplied Proms and:

| <u>FOUR ZR33481</u> | <u>FOUR ZR33881</u> |
|---------------------|----------------------|
| 1 - 64 | 1 - 128 (dec. by 4) |
| 1 - 48 | 1 - 96 (dec. by 3) |
| 1 - 32 | 1 - 64 (dec. by 2) |
| 1 - 16 | 1 - 32 (no decimate) |

CLOCK MODES AND SYNCHRONIZING:

- External clock: positive or negative edge synchronized (jumper selectable)
- On-board oscillator: 14.3182 crystal

The on-board/off-board clock option is jumper selectable. The on-board clock source is derived from the 14.3182 or 20 MHz oscillator module. This module is replaceable, should other clocking frequencies be required.

For 16-bit input data, the transfer is made with two 8-bit bytes. The SIC (sync) signal determines the high/low byte. The CIC (Clock) signal is the data acquisition reference. The output data is 16-bit and the COC signal is the output data reference.

OPERATING ENVIRONMENT

STAND ALONE OPERATION:

The DFPB, by using a separate power supply, may be used in a stand-alone configuration. In this configuration, I/O occurs through the edge connector (P3). Coefficients cannot be loaded via the P3 edge connector, consequently they must be installed in the coefficient memory locations with some form of non-

volatile memory. Clocking may be controlled by either the external or on-board clock. (Data, Address and Control signals are defined in the section titled Hardware Interface and Requirements.)

IBM PC/AT OR PC/XT WITH SOFTWARE CONTROL:

The PC may be used as a host to perform filtering through the P3 interface. Coefficients may be developed with commercially available software or ZORAN's DFPS Digital Filter Development Software part # ZR63301. After coefficients have been

determined, they may be stored in non-volatile memory and inserted into the board. The board may then be used as if it were in a stand alone mode, using the P3 connector interface only.

PC/AT OR PC/XT OPERATION WITHOUT SOFTWARE CONTROL:

The PC may also use the board as a data acquisition port, an accelerator for filter calculation and as a signal generator. In this mode, the PC transfers a sampled data stream to and from the DFPB over the PC bus. The PC, in turn, will also accept filtered data from the DFPB. Input and filtered data are accessed through the edge connector I/O ports. Input data is stored in the DFPB signal memory. The PC can access this memory, although not on a continuous basis. The filter coefficient memory is also accessible.

Likewise, the filtered signal may be stored in a PC file or supplied to other equipment through the P3 connector.

The main menu identifies the principle features available.

- 1 Help
- 2 Board Status
- 3 Board Setup
- 4 Download Filter to Board
- 5 Upload Filter from Board
- 6 Download Signal to Board
- 7 Upload Signal from Board
- 8 16-Bit Board Filter
- 9 22-Bit Board Filter
- 10 Upload Output from Board
- 11 Manual Filter Load
- 12 Display Memories
- 13 Check Board
- 14 Return to DOS

The DFPB is designed to work with a software utility (DFPBU) supplied with the Board. DFPBU is a high-level menu-driven program which allows the user to set up the board for different filtering modes, load signals and coefficients, and read the results. The software contains a diagnostic tool to check for hardware and operator errors. DFPBU is designed to accept DFPS files or it can use manually entered files. DFPBU accepts signals originating from a PC file or supplied by a outside source, through the P3 connector.

HARDWARE INTERFACE AND REQUIREMENTS

IBM PC BUS (P1):

The DFPB plugs directly into a standard PC/XT or PC/AT bus. The PC mother board connector signals used by the DFPB are:

| | |
|---------|------------------------------|
| SA<0:9> | 10 address input lines |
| AEN | address enable input |
| SD<0:7> | 8 bit-directional data lines |
| IOR | Read strobe |
| IOW | Write strobe |

The board is I/O memory addressed by a user selectable address switch for addresses 3 - 9. Therefore, multiple boards can be used in a single system. The ZORAN supplied address setting is HEX-310.

THE DFPB EDGE CONNECTOR (P3):

The edge connector allows DFPB interface to other sources such as other DFP or VSP boards, digital oscilloscopes, etc. The interface uses a 50-pin male 3M Header connector. It mates with two ZORAN supplied custom AP Products flat cables 18 inches long, using 20-pin female connectors. The 10 pins in the center of the 50-pin connector are not used.

SIGNAL DESCRIPTION AND DESIGNATION FOR EDGE CONNECTOR P3:

| <u>ROW 1</u> | | <u>ROW 2</u> | | <u>COMMENTS</u> | |
|---------------|------------|--------------|---------------|------------------------------|--------------------|
| <u>SIGNAL</u> | <u>PIN</u> | <u>PIN</u> | <u>SIGNAL</u> | | |
| | 1. | | 2. | | |
| | 3. | | 4. | | |
| | 5. | | 6. | | |
| | 7. | | 8. | | |
| ID08 | 9. | 10. | ID09 | IDXX = Input data (8 bits) | |
| ID10 | 11. | 12. | ID11 | | |
| ID12 | 13. | 14. | ID13 | | |
| ID14 | 15. | 16. | ID15 | | |
| SIC | 17. | 18. | | | |
| GND | 19. | 20. | CIC | | |
| | 21. | 22. | | | |
| | 23. | 24. | | | |
| | 25. | 26. | | | |
| | 27. | 28. | | | |
| | 29. | 30. | | | |
| OD00 | 31. | 32. | OD01 | ODXX = Output data (16 bits) | |
| OD02 | 33. | 34. | OD03 | | |
| OD04 | 35. | 36. | OD05 | | |
| OD06 | 37. | 38. | OD07 | | |
| OD08 | 39. | 40. | OD09 | | |
| OD10 | 41. | 42. | OD11 | | |
| OD12 | 43. | 44. | OD13 | | |
| OD14 | 45. | 46. | OD15 | | |
| | 47. | 48. | | | |
| GND | 49. | 50. | COC | | |
| | | | | | COC = Output clock |

Note: All unlabeled pins are not used

BOARD CHARACTERISTICS

Physical Characteristics

IBM PC/XT system expansion board (full size, also fits AT)
 Dimensions: 13.2" wide, 4.2" high

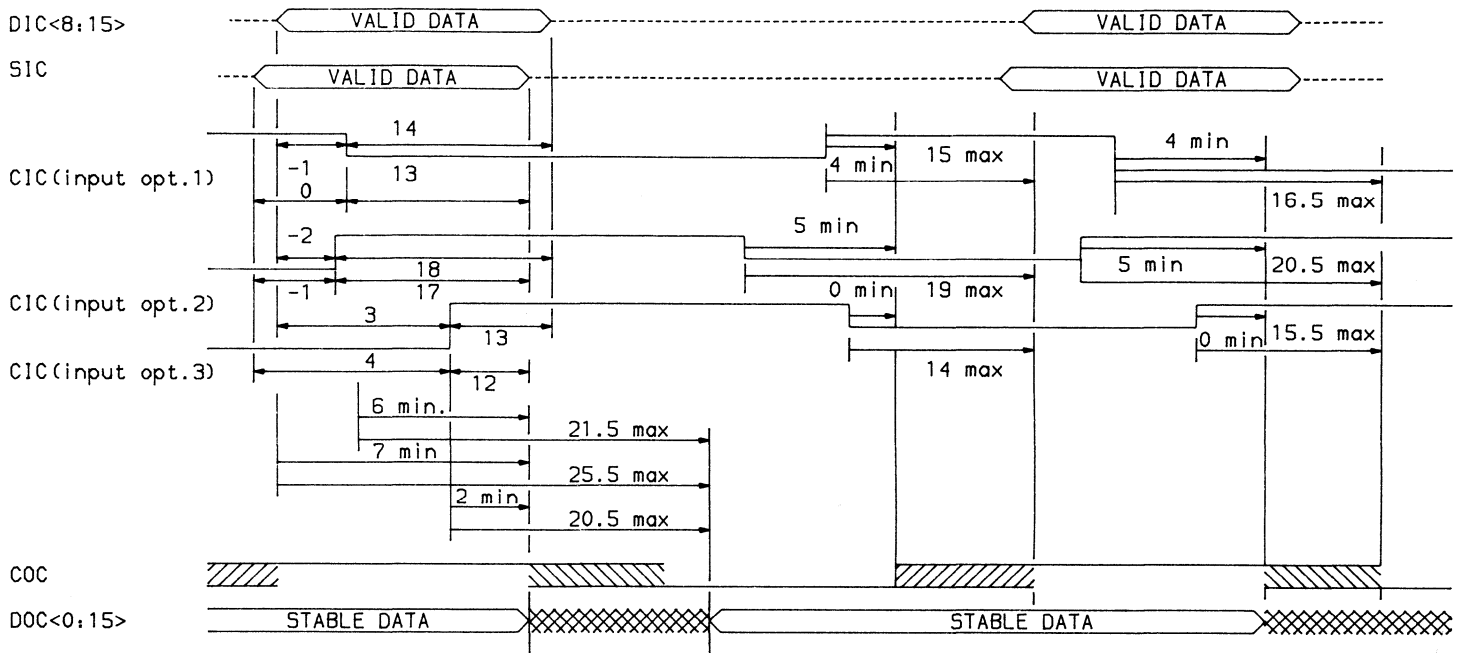
Environmental Characteristics

Temperature: 0-50 degrees centigrade

Electrical Characteristics

DC Power: 3.5 amp, +5V
 Parallel I/O timing: (see Figure 3)

P3 INPUT-OUTPUT CONNECTOR TIMING
 (ALL TIMES IN NANoseconds)



- Option 1 CIC clock sync to rising edge external clock
- Option 2 CIC clock sync to falling edge external clock
- Option 3 CIC output clock for board supplied clock

REFERENCE DOCUMENTATION

DFPB User Manual
 DFP data sheets
 4 cell ZR33481
 8 cell ZR33881
 DFPS data sheet
 DFPS User Manual

SOFTWARE REQUIREMENTS

MSDOS version 2.1 or greater

ZORAN's DFP Board Utility Software (DFPBU) is required for computer control of the board. The software takes 88K of system memory in operation, and is included in the DFPB package.

PRODUCT DESCRIPTION/ORDER INFORMATION

DFPBs are available with four or eight cell DFPs and configured for 15 MHz operation. Customers ordering the completed assembly will receive a completely operational board with coefficient RAM, oscillator, two connector cables, user manual

and software diskettes, sequence PROMs and DFPs. The individual parts are available as replacements, or spares. Oscillator frequency, DFP type, and PROM type must be specified.

| <u>QUANTITY</u> | <u>DESCRIPTION</u> | <u>PART NUMBER</u> |
|-----------------|-----------------------------------|----------------------------|
| 1 | PC Board, w/o OSC, DFPs and PROMs | ZR73301-B |
| 1 | DFPB User's Manual | ZR73301-M |
| 2 | 20 Pin Edge Connector Cables | ZR73301-C |
| 2 | 5 1/4" Utility Software Diskette | ZR73301-U |
| 3 | 1 Set of Sequence PROMs | ZR73301-P-4 (4 cell DFP) |
| | or | ZR73301-P-8 (8 cell DFP) |
| 1 | Crystal Oscillator | ZR73301-O-15 (14.3182 MHz) |
| | or | ZR73301-O-20 (20 MHz) |
| 1 | 2K × 8 Coefficient RAM | ZR73301-R |
| 4 | ZR33481s | ZR33481JC-y |
| 4 | ZR33881s | ZR33881JC-y |
| 1 | Completed assembly | ZR73301-xy |
| | x = 4 or 8 (filter type) | |
| | y = 15 | |

DISTINCTIVE FEATURES

- Determines impulse response coefficients for FIR filters up to 1024 taps in length
- Provides both McClellan-Parks and windowing filter design techniques
- Supports multistage filter design for decimation or interpolation filters
- Allows variable quantization of coefficients

- Supports JEDEC format for PROM programming
- Supports ZORAN DFP hardware board
- Synthesizes filter input signals
- Filters an input signal with the created filter
- Computes and plots filter frequency response, input and output signal spectrum
- Easy-to-use menu-driven and command user interfaces
- Operates in PC XT™/AT™ (MS-DOS™), or VAX™ (ULTRIX™ or VMS™) environments

DESCRIPTION

ZORAN's ZR63301 Digital Filter Processor Software (DFPS) aids the user in the design of Finite Impulse Response (FIR) filters. It is also an analysis, data manipulation and display tool. DFPS permits the design of FIR filters using either the McClellan-Parks or windowing technique. Given user-specified performance criteria, the software determines impulse response coefficients that best satisfy the criteria. The resulting coefficients may be quantized to the actual number of bits used for practical hardware implementation. Signals may be created, quantized and filtered. The filter frequency response, input signal spectrum and filtered output signal spectrum may be computed and graphed. Multistage decimation or interpolation

filters may also be designed. The software is either menu-driven or command-driven and was specifically designed for ease of use.

Hardware implementation of a filter design is facilitated by the JEDEC format PROM programmer output of the DFPS.

Board interface utility software permits operation of ZORAN's Digital Filter Processor Board (DFPB) under DFPS control.

Versions of the DFPS are available for the PC XT/AT under MS-DOS and the VAX under both ULTRIX and VMS. DFPS is a design tool for users of the ZORAN Digital Filter Processor (DFP) family and the Digital Filter Processor Board (DFPB). It is also a generic FIR filter design package that can be used for non-ZORAN applications.

FILTER DESIGN

Given a user-specified and desired filter frequency response, DFPS will estimate the number of taps required and compute the filter impulse response coefficients necessary for the appropriate FIR implementation.

Coefficients may be determined by the McClellan-Parks optimal Chebyshev technique (4-128 taps) or the windowing technique (8-1024 taps). The McClellan-Parks technique can design lowpass, highpass, bandpass, differentiators and Hilbert transform filters.

The windowing technique provides the following window types:

- Rectangular
- Triangular
- Hamming
- General Hamming
- Hanning
- Kaiser
- Chebyshev

Multistage decimation or interpolation filters may also be designed. The software determines the number of stages, amount of decimation or interpolation between each stage and the number of taps per stage from the desired overall characteristics of the filter. The software provides for filters of up to three stages with interpolation or decimation by 2, 3 or 4 between each stage.

SIGNAL GENERATION

DFPS can interactively create a signal of up to 4096 samples. Alternatively, DFPS may use external signal files of the proper format. The file may contain up to 32-bit fixed point real or complex numbers, or floating-point numbers with precision up to the level of the host processor. Signals may be scaled by a constant or point-wise added. Generated signals may be chosen from:

- Sinusoids
- Uniform random noise
- Square waves
- Step or flat functions
- Impulses
- Any combination of the above

Signals may be quantized to simulate the effects of an actual hardware implementation. User-created data initially is in floating-point format. Fixed precision signals may be generated from floating-point signals using the quantized function.

SIGNAL FILTERING

The operation of a filter may be simulated by convolving an input signal file with a filter kernel (coefficients) to create a filtered output signal.

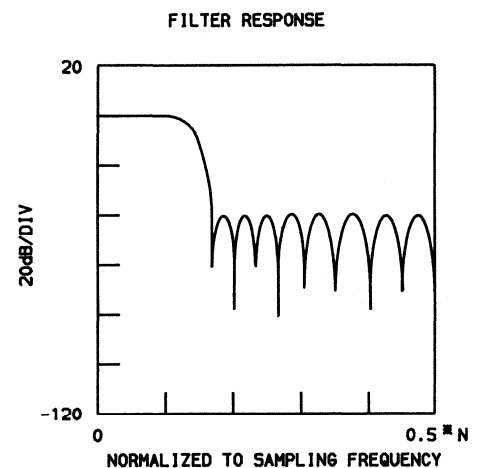
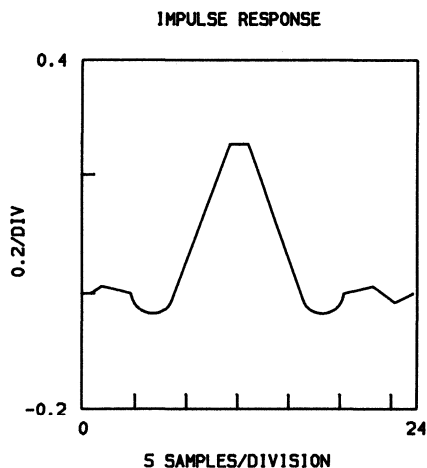
TRANSFORMS

DFPS computes the filter frequency response from the impulse response coefficients. It also computes frequency spectrums for input signals and filtered output signals.

OUTPUTS AND DISPLAYS

Input signals, output signals and the filter impulse response may be listed in decimal format or plotted in linear scale. In addition, the filter frequency response may be plotted in linear or log scale or listed in decimal format. For detailed examination, the impulse response or frequency response functions may be expanded along the horizontal axis.

Coefficients for a single stage may be stored in a file for a PROM programmer in JEDEC standard format. Up to 2048 integer coefficients and their checksums may be stored, if desired, in standard $2K \times 8$ PROM configuration. The coefficients may then be loaded into ZORAN's DFP Development Board (ZR73301) to allow real time evaluation of the synthesized filter.



QUANTIZATION

Coefficients and signals may be quantized to the actual number of integer bits used in a hardware implementation of the filter. These quantized impulse responses and signals, as well as their transforms, may be plotted to view the effects of quantization.

PROM PROGRAMMING

Filter coefficients can be formatted and written to JEDEC standard fuse map files that, in turn, can be used to program PROMs via a PROM programmer. DFPS supports PROMs of 8-bit words, up to 2048 (2K) words long.

MENU INTERFACE

The menu-user interface is exemplified by the main DFPS menu.

DFPS Main Menu

Select:

1. Help
2. Generate an FIR filter
3. Generate a signal
4. Generate a filtered signal
5. Generate a signal spectrum or filter frequency response
6. Operate on an existing data object
7. Format filter coefficient to JEDEC PROM programmer format
8. Enter command interface option
9. Exit to option system

DFPS FUNCTION MENU

M main menu

-1 help

| 2 h(t) | 3 x(t) | 4 y(t) | 5 H(w)&F(w) | 6 data | 7 JEDEC | 8 command mode |
|--|--|-------------------|-----------------|--------------------------------|---|-------------------|
| -1 help | -1 help | -1 help | -1 help | -1 help | -1 help | -1 help |
| -2 FIR Pk = McCL -1 low pass -2 high pass -3 band pass -4 band stop -5 multi band -6 differentiate -7 Hilbert | -2 x(t) -0 quit -1 step -2 impulse -3 cosine -4 random -5 squarewave | -2 x(t) | -2 x(t)*h(t) | -2 H(w) | -2 load data -1 signal -2 filter coef. -3 filtered signal -4 freq. transf. -5 scratch array -6 multistage filt. | -2 coef. to JEDEC |
| -3 FIR window -1 low pass -2 high pass -3 band pass -4 band stop | -3 list x(t) | -3 x(t)*h(t)n | -3 F[x(t)] | -3 list data -same as M-6-2 | -3 list coef. | |
| -4 FIR manual | -4 plot x(t) | -4 list x(t)*h(t) | -4 F[x(t)*h(t)] | -4 plot data -same as M-6-2 | -4 scale coef. | |
| -5 FIR multist. -1 help -2 lowpass -3 multistage -4 list -5 save -6 quantize -7 plot | -5 quantize | -5 plot x(t)*h(t) | -5 H(w)n | -5 save data -same as M-6-2 | -5 quantize coef. | |
| -6 list coef. | -6 save x(t) | -6 list x(t)*h(t) | -6 list H(w) | -6 quantize -same as M-6-2 | -6 to main menu | |
| -7 plot coef. | -7 to main menu | -7 to main menu | -7 plot H(w) | -7 scale -same as M-6-2 | | |
| -8 save coef. | | | -8 save H(w) | -8 decimate -same as M-6-2 | | |
| -9 quantize coef. | | | -9 to main menu | -9 to main menu | | |
| -10 plot H(w) | | | | | | |
| -11 to main menu | | | | | | |

COMMAND MODE INTERFACE

A command mode user interface is available for experienced users. The advantage of this mode is its ability to immediately execute all DFPS functions without a time-consuming search through the menu hierarchy.

| | | |
|---------------------------|-------|--|
| g(en)f(ir)p(m) | | generate an FIR filter with the Parks-McClellan algorithm. |
| g(en)f(ir)w(indow) | | generate an FIR filter with the "window" algorithm. |
| g(en)s(ig) | | generate a test signal interactively. |
| c(o)e(ntry) | | manual entry of filter coefficients. |
| f(ilter) | | convolve signal data with filter data and place results in output data object. |
| t(ransform) | <obj> | compute FFT of data object, place magnitude in transform. |
| l(oad) | <obj> | load the data object from disk. |
| s(ave) | <obj> | save the data object to disk. |
| d(isplay) | <obj> | print a table of the data object values. |
| p(lot) | <obj> | display graph of requested data object on monitor. |
| h(elp) | | print this page. |
| m(enu) | | enter menu interface option. |
| q(uit) | | exit DFPS. |
| p(lot) | <obj> | display graph of requested data. |
| h(elp)m(ore) | | print this page. |
| sc(ale) | <obj> | multiply the data object by a constant. |

HARDWARE REQUIREMENTS

- VAX with VT240 or Tektronix 4014 terminal, or PC/AT or /XT with 640K memory, high resolution monitor and EGA color card
- PROM programmer (optional)

ORDERING INFORMATION

ZORAN requires that a software license agreement be signed before shipment. ZORAN's standard license agreement will be shipped with an acknowledgement of your order.

- DFPS For PC/AT or PC/XT (MS-DOS) ZR63301-100
For VAX (VMS) ZR63302-100
For VAX (ULTRIX) ZR63303-100
(single machine license)
- User Manual—included with DFPS order
—additional copies available:
part #ZR63301-M

All command lines begin with the command keyword. The portion of the keyword enclosed in parentheses need not be typed. A list of commands and a brief description of each is given below.

| | | |
|------------------------|-------|---|
| q(uan)t(ize) | <obj> | quantize by rounding the data object to a fixed precision. |
| in(teger) | <obj> | round the data object to integer values. |
| tr(uncate) | <obj> | make the data object integer values. |
| d(e)c(imate) | <obj> | decimate (throw out alternate samples from) the data object. |
| i(n)t(erpolate) | <obj> | interpolate with zeros. |
| pr(om) | <obj> | generate a PROM fuse map (JEDEC) from filter coefficients. |
| g(en)m(ulti) | | generate a multistage filter. |
| f(il)m(ulti) | | filter signal with multistage filter, place output in output. |
| t(rans)m(ulti) | | transform a multistage filter, place output in transform. |
| s(um)s(ig) | | add the scratch data object to the signal. |
| m(ult)s(ig) | | multiply the signal data object by the scratch data object. |

Data objects, <obj>, are **signal**, **filter**, **output** (filtered signal), **transform**, **scratch** or **multistage**; these may be abbreviated **s,f,o,t,x** and **m** respectively.

HARDWARE ACCESSORIES

- ZR73301—The Digital Filter Processor Board (DFPB) is a flexible 20 MHz 4 to 128 tap digital filter system containing up to four DFPs, signal and coefficient memory, sequence control and a parallel 20 MHz I/O port.

OPERATING SYSTEM REQUIREMENTS

- ULTRIX or VMS (V4.2) for VAX
- MS-DOS version 2.1 or higher for PC/AT or PC/XT

UPDATES

ZORAN supplies software updates for a period of one(1) year from date of purchase to all original purchasers who have submitted their software registration certificate. Updates are available after the initial one(1) year period by ordering the original version (PC or VAX) part number and adding a suffix "-D" (example, ZR63301-100-D).

APPLICATIONS

MARKETS

- Electronic Publishing
- CD ROM
- Machine Vision
- Telecommunication
- Medical Imaging
- Military Reconnaissance
- Digital Video
- Electronic Cameras

FUNCTIONS

- 2-D FCT for Compression (16×16 Block in < 1 ms)
- 2-D Real-time Spatial Domain Convolution/Correlation (7×7 Kernel at 20 MHz)
- 2-D FFT (64×64 in < 15 ms)
- Frequency Domain Convolution/Correlation (32×32 in < 20 ms)
- 1-D Video Rate Convolution/Correlation (> 20 MHz sample rate)
- 1-D or 2-D Decimation or Interpolation

INTRODUCTION

Image processing functions require substantial computational capability. For example, spatial convolution of a 512×512 image with a 3×3 kernel requires nearly 2.4 million multiply-accumulate operations. Performing this rather simple function in real-time on a continuous sequence of images, at say 30 frames per second, requires nearly 71 million operations per second.

Zoran Corporation has developed two families of VLSI Digital Signal Processors (DSPs) for use in all major categories of image processing:

- Compression
- Enhancement
- Reconstruction
- Analysis

APPLICATIONS

- Image Compression
- Image Enhancement
- Image Reconstruction
- Image Analysis

USER BENEFITS

- Expand design opportunities
- Overcome price/performance barriers
- Easy design-in — $>$ quick to market

KEY FEATURES

- Multiple processors on a single IC
- Real-time linear spatial domain processing
- Fast frequency domain processing

These processors significantly improve price/performance ratios for conventional digital image processing. In addition, they offer entirely new possibilities for image processing by overcoming traditional problems of inadequate performance, high cost or both.

The following pages will show how to use these powerful processors for:

- 2-D spatial domain convolution or correlation
- 2-D transform domain processing using the FFT for spectral analysis or very large kernel convolution and correlation
- 2-D Fast Cosine Transform for image compression
- 1-D FIR filters for digital video, decimation, interpolation and pyramid transforms

2-D SPATIAL DOMAIN PROCESSING

Spatial domain processing includes any processing which operates directly on the pixels of the image, including both 1-D and 2-D operations.

2-D SPATIAL DOMAIN CONVOLUTION OR CORRELATION

Two-dimensional spatial convolution or correlation is a fundamental image processing primitive. Linear filters for enhancement and restoration of degraded images and edge detection for feature extraction and measurement use this primitive.

Due to the enormous amount of computation required to implement 2-D spatial convolution, systems designers have restricted designs to small kernel sizes (e.g. 3×3). For real-time performance, even these limited configurations have resulted in large, complex processors using many components. They have been expensive, and consumed space and power.

Larger kernels (e.g. 7×7 , 9×9) provide greater flexibility in filter and edge detector design. They permit better approximations to ideal filtering characteristics. However the computation requirement for these larger kernels increases exponentially with size. Spatially convolving a 512×512 image sequence at 30 frames per second with a 7×7 kernel requires 385 Million multiply-accumulates per second.

The Zoran Digital Filter Processor (DFP) family is designed to excel at computing sums-of-products for 1-D and 2-D convolutions and correlations at video rates. These processors are ideally suited to real-time processing of larger kernels. One member of this family contains eight multiply-accumulators in a single integrated circuit. When operating at its maximum sample rate of 20 MHz, this systolic-like multiprocessor system executes a whopping 340 million arithmetic operations per second.

The DFP family provides the designer:

- Economical real-time small kernels (3×3)
- Real-time larger kernels (7×7)
- Full-precision coefficients (9 bits) for accuracy and flexibility
- Ease of design
- Reduced board space and power

A few basic system configurations illustrate the power of the DFP family for 2-D spatial processing. A 3×3 convolution can be computed at a rate of 6.67 MHz per pixel with a single 4-cell DFP (Figure 1). Hence a 256×256 image can be convolved in 9 ms, well within real-time constraints. Convoluting a 512×512 image with a 3×3 kernel in real-time requires only two 4-cell DFPs.

A 7×7 kernel convolution can be computed at a rate of 5 MHz per pixel with two 8-cell DFPs (Figure 2). Extending this performance to a 10 MHz pixel rate requires only four 8-cell DFPs.

2-D Spatial Convolution Performance

| <u>KERNEL</u> | <u>SAMPLE RATE</u> | <u>DFPs</u> |
|-----------------------|--------------------|-------------|
| 3×3 | 6.67 MHz | 1 |
| 3×3 | 10 MHz | 2 |
| 3×3 | 20 MHz | 3 |
| $2 \times 3 \times 3$ | 5 MHz | 1 |
| 5×5 | 6.67 MHz | 2 |
| 7×7 | 5 MHz | 2 |
| 7×7 | 10 MHz | 4 |
| 9×9 | 6.67 MHz | 6 |

For imaging applications, the DFP provides a compact, practical implementation of real-time 2-D spatial convolution or correlation. The board space and power saved using DFPs brings powerful image processing capability within reach of desktop systems, airborne imaging systems and the like.

EDGE DETECTION

Edges are primitive image features important in many applications. Accurate edge detection is essential to measurement and gauging applications in machine vision. Edges are the basic primitives from which objects are built in pattern recognition systems.

Most edge detection methods require a sum-of-products operation for computing 1-D or 2-D convolution or correlation. The

DFP family excels at sums-of-products and provides basic edge detection computations. The significance of the DFP family is that it permits economical solutions to larger kernels which generate more precise edge detectors. For larger kernels, full precision 8-bit \times 8-bit or 9-bit \times 9-bit multiplication is essential for accuracy and flexibility. The DFP family affords the image processing community greater flexibility in implementing higher order edge detection algorithms economically.

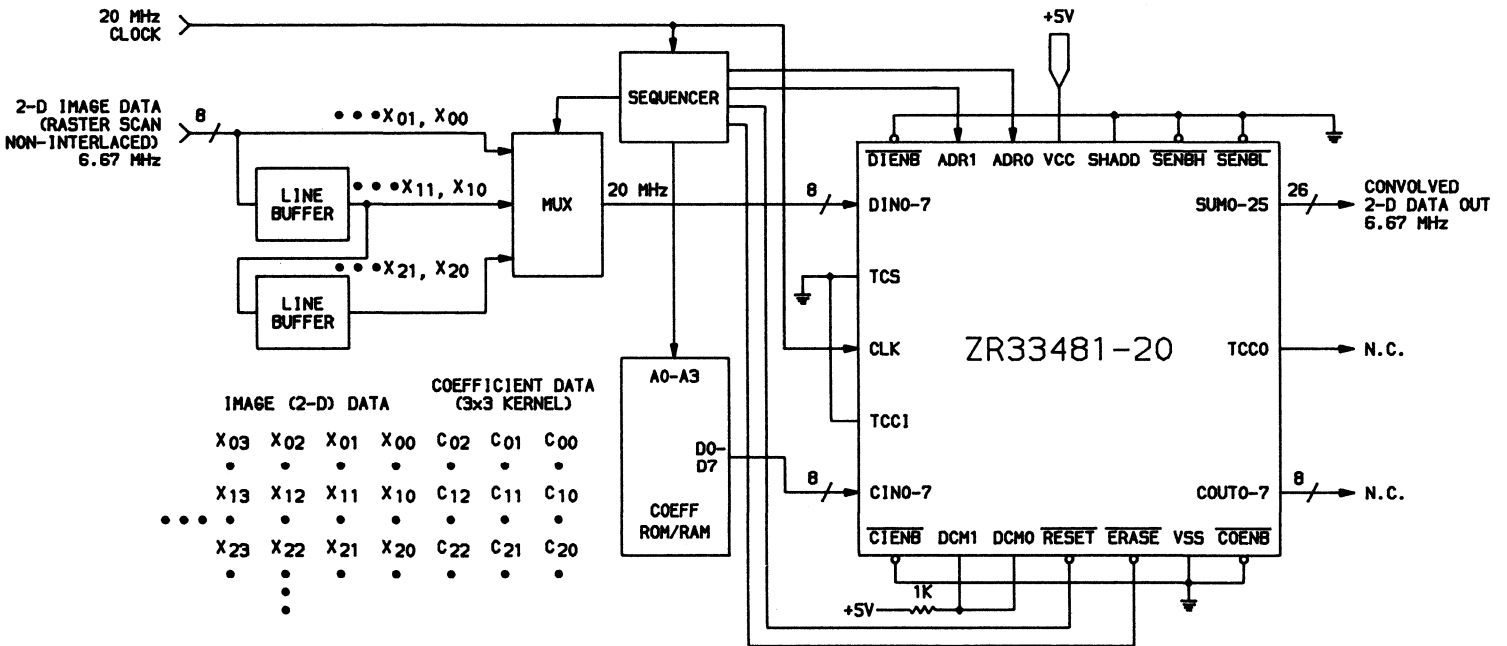


FIGURE 1. 6.67 MHz 3x3 CONVOLUTION.

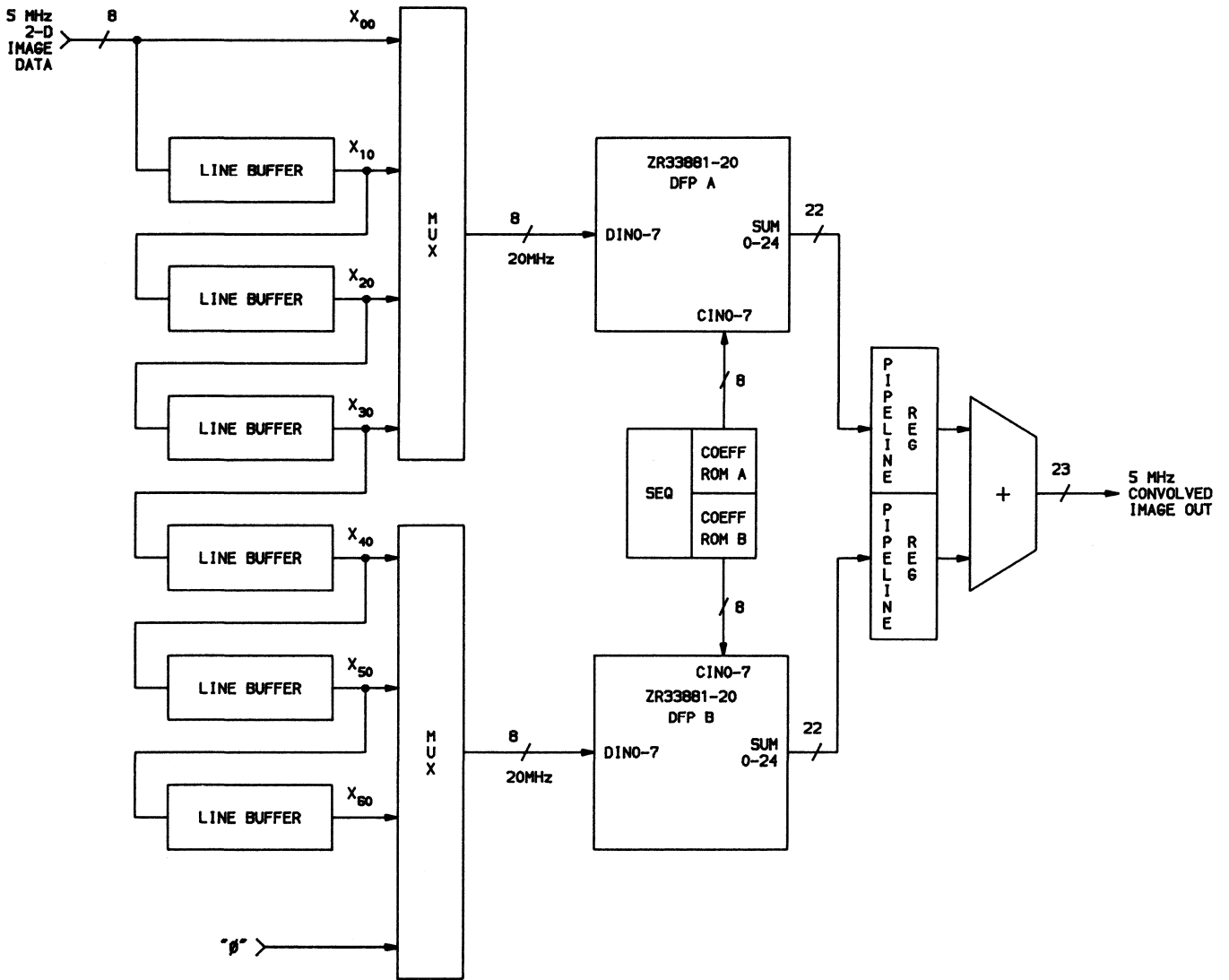


FIGURE 2. 5 MHz 7x7 CONVOLUTION.

1-D FINITE IMPULSE RESPONSE (FIR) FILTERS FOR DIGITAL VIDEO

Digital video systems contain prodigious numbers of FIR filters. Applications include composite-to-component decoders, comb filters, low pass filters for video compression, image enhancement filters, standards conversion, and special effects.

Various types of FIR filters are required. Fixed coefficient filters of 8 to 32 taps and sample rates of 13.5 MHz, 14.3 MHz or 17.7 MHz are common requirements. Nine-bit coefficients and data are necessary for acceptable quality in video equipment. Decimation and interpolation FIR filters are needed for sample rate conversion between different video signal representations

The Zoran Digital Filter Processor (DFP) family is ideally suited to these kinds of FIR filters. The DFP provides the video system designer:

- High integration, multiple processor system
- High sample rates (over 20 MHz)
- Efficient decimation and interpolation
- Ease of design
- Reduced board space and power

An 8-tap programmable, fixed coefficient FIR filter supporting up to a 20 MHz sample rate requires a single DFP (Figure 3). Longer filters are created by simply cascading several DFPs.

1-D FIR Filter

| SIZE | SAMPLE RATE | DFPs |
|--------|-------------|------|
| 8-tap | 20 MHz | 1 |
| 16-tap | 20 MHz | 2 |
| 32-tap | 20 MHz | 4 |

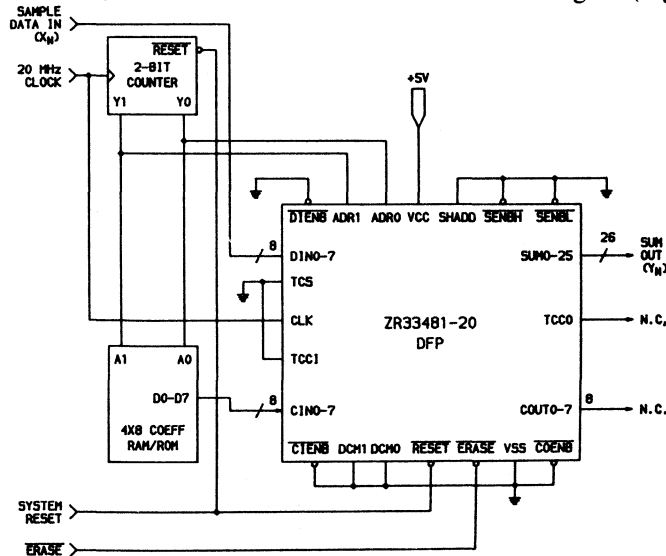


FIGURE 3. 8-TAP, 20 MHz FIR FILTER.

DECIMATION AND INTERPOLATION

The DFP architecture efficiently supports decimation. For example a 32-tap decimate-by-four FIR with 20 MHz input and 10 MHz output sample rates requires a single DFP, in a configuration similar to that shown in Figure 3.

1-D Decimation FIR Filter

| SIZE | DECIMATION | DFPs |
|--------|------------|------|
| 16-tap | by 2 | 1 |
| 24-tap | by 3 | 1 |
| 32-tap | by 4 | 1 |

The DFP can also be configured to implement interpolation filters using a polyphase structure. A 16-tap, 20 MHz to 40 MHz interpolation FIR filter is constructed with only two DFPs.

1-D Interpolation FIR Filter

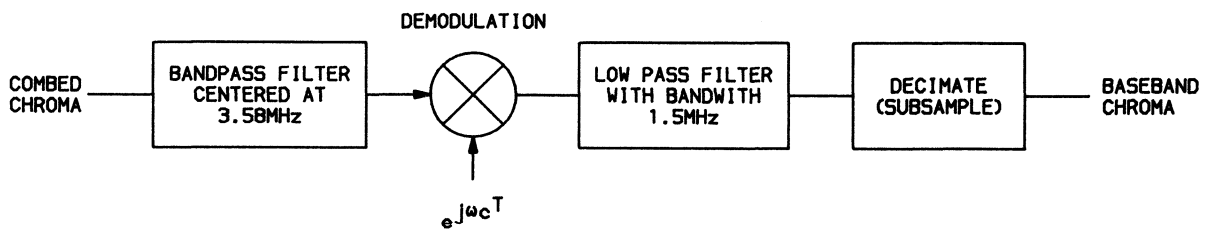
| SIZE | INTERPOLATION | DFPs |
|--------|---------------|------|
| 16-tap | by 2 | 2 |
| 32-tap | by 2 | 4 |

A typical application for the DFP is a composite-to-component decoder. The DFP excels at demodulating and filtering the comb-filtered chroma signal (Figure 4).

PYRAMID PROCESSING

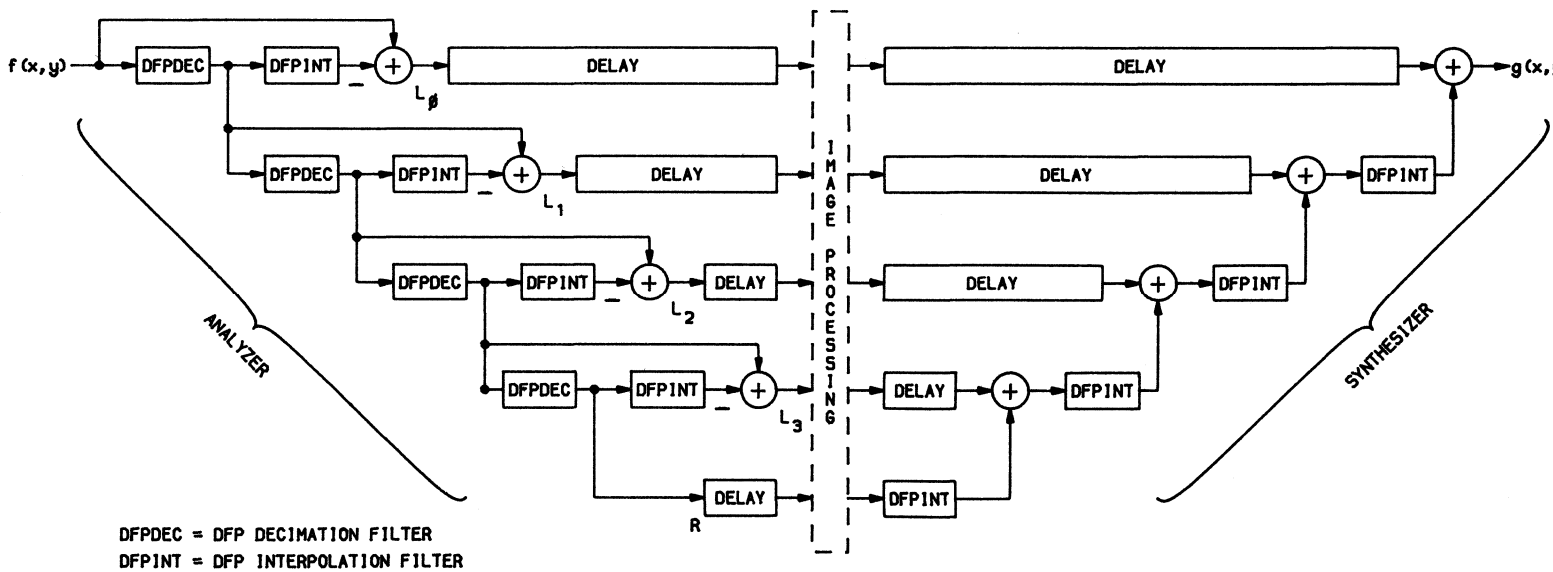
Pyramid processing models the human visual system by decomposing an image into a set of contiguous overlapping spatial frequency bands. This permits the image processing to be optimized in each band to take advantage of the eye's nonlinear visual response. Since each frequency band has less bandwidth than the entire image, the processing required for each band can be substantially less than for the entire image.

Decomposing the image into the bands requires a large number of decimation and interpolation FIR filters operating at video sample rates (Figure 5). The bandpassed image is processed as desired, (e.g. enhancement or coding). Both the Zoran DFP and VSP support this processing. The processed image is then reconstructed by a bank of interpolation FIR filters. The DFP is ideally suited to building the video-rate decimation and interpolation FIR filters.



ω_c = Subcarrier Frequency
 T = Sampling Period

FIGURE 4. FILTERING AND DEMODULATION OF COMB-FILTERED CHROMA.



DFPDEC = DFP DECIMATION FILTER
 DFPINT = DFP INTERPOLATION FILTER

FIGURE 5. PYRAMID PROCESSING.

FREQUENCY DOMAIN PROCESSING

An image is a spatially varying function. In many applications one would like to know or process the frequencies of these variations. To do this the image must be transformed from the spatial to the frequency domain. Low spatial frequencies correspond to slowly varying gray levels such as intensity variation over a continuous surface. High spatial frequencies correspond to rapidly varying information such as edges.

Advantages of processing in the frequency domain include:

- Faster, more efficient convolution or correlation for large kernels
- Expanded design possibilities unique to the representation of images in the frequency domain

PROCESSING UNIQUE TO THE FREQUENCY DOMAIN

Frequency domain processing provides unique processing opportunities, such as:

- Efficient image compression or coding algorithms
- Feature extraction such as Fourier descriptors for boundaries or pattern recognition based on Fourier coefficients

- Restoration of degraded images using deconvolution to remove, for example, blurring or noise
- Spectral analysis of images
- System identification and modeling
- Image detection and registration
- Image rotation in the frequency domain
- Texture feature extraction and analysis
- Image reconstruction from projections

THE FAST FOURIER TRANSFORM

The Fourier Transform or its computationally more efficient cousin, the Fast Fourier Transform (FFT), is the basic transform used in frequency domain processing. The 2-D FFT is computed by performing a series of 1-D FFT operations on all the image rows, then another series of 1-D FFTs on all the columns (Figure 6).

The Zoran Vector Signal Processor (VSP) excels at vector and transform operations such as the FFT. It provides the user with:

- Very high performance in a single IC
- Multiple processor configurations for even higher speed
- Block floating arithmetic for high dynamic range
- Embedded algorithms to reduce design time
- Reduced board space and power

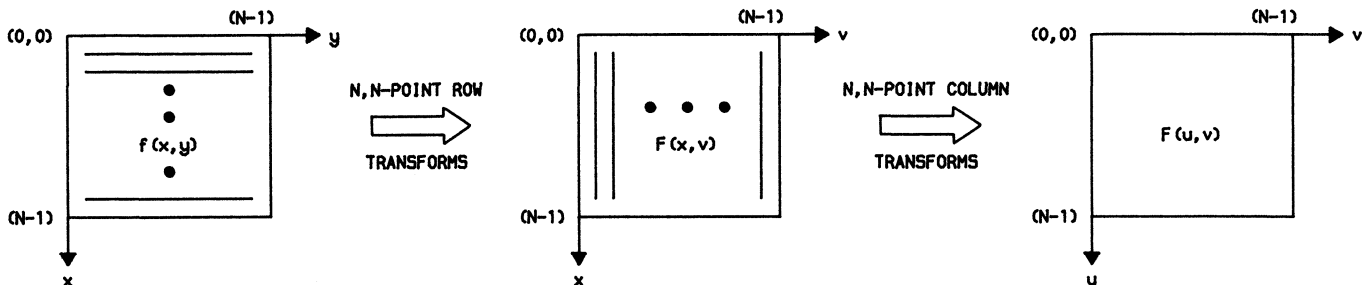


FIGURE 6. 2-D $N \times N$ POINT FFT BLOCK DIAGRAM.

For example the VSP can compute a 2-D 64×64 FFT in less than 15 milliseconds. This is real-time operation for a 30 frames per second image sequence.

A VSP system to perform frequency domain image processing functions loosely couples the VSP as a coprocessor to a host microprocessor (Figure 7).

For higher performance systems multiple processor VSP configurations compute larger FFTs in real-time. This capability has heretofore required complex systems designed with many components, including high-speed MACs, ALUs, address sequencers, etc. Because of the cost, space and power savings, and the ease-of-design, the VSP creates new market opportunities for high-performance imaging processing systems. Frequency domain processing is finally a cost-effective alternative to spatial domain processing.

2-D Fast Fourier Transform Performance

| <u>SIZE</u> | <u>TIME(ms)</u> | <u>VSPs</u> |
|-------------|-----------------|-------------|
| 8×8 | .1 | 1 |
| 16×16 | .4 | 1 |
| 64×64 | 8.0 | 1 |
| 128×128 | 36.0 | 1 |
| 128×128 | 21.0 | 2 |
| 256×256 | 270.0 | 1 |
| 256×256 | 150.0 | 2 |

2-D FAST CONVOLUTION OR CORRELATION IN THE FREQUENCY DOMAIN USING THE FFT

Various image processing applications require very large kernel convolutions or correlations. For example, highly accurate filters for enhancement or edge detection require kernels of size 64×64 and larger. The larger kernel more closely approximates an ideal filter with sharp cutoff. Correlating with a kernel of size 64×64 or larger for pattern matching over large image areas is not an uncommon desire in applications such as machine vision.

Using spatial convolution or correlation, the computational burden increases exponentially with kernel size. At some point it becomes more economical to perform convolution via the frequency domain. A comparison of the number of operations required for 2-D spatial and frequency domain convolution shows the crossover point to be at about a 9×9 kernel size for a 256×256 image (Figure 8).

Therefore, the image processing system designer would like to have the ability to perform convolution and correlation via the frequency domain. The technique is to Fourier transform the image and the kernel, multiply their transforms in the frequency domain, and finally inverse transform this product back to the spatial domain (Figure 9).

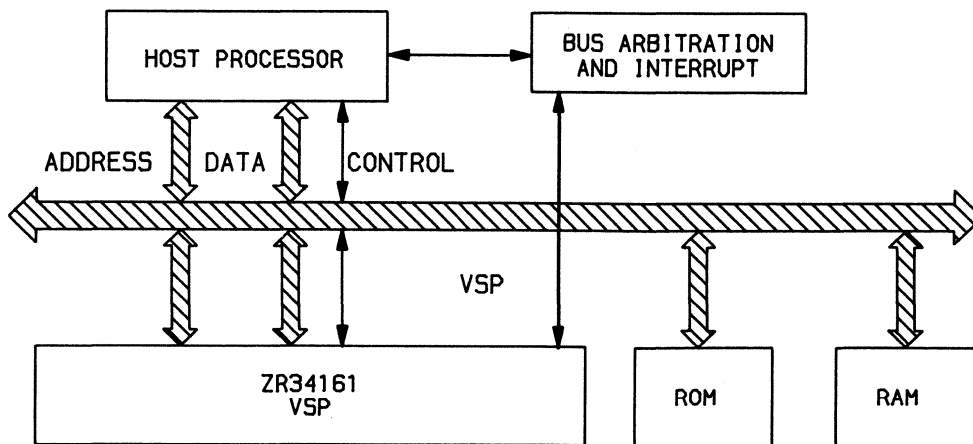


FIGURE 7. TYPICAL VSP-BASED SYSTEM.

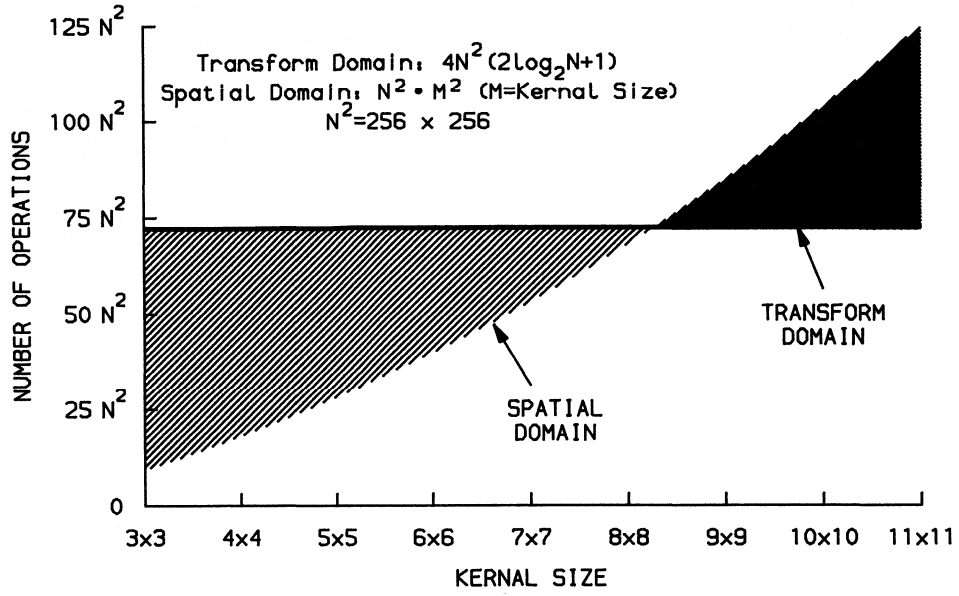


FIGURE 8. SPATIAL VS. FREQUENCY DOMAIN 2-D CONVOLUTION.

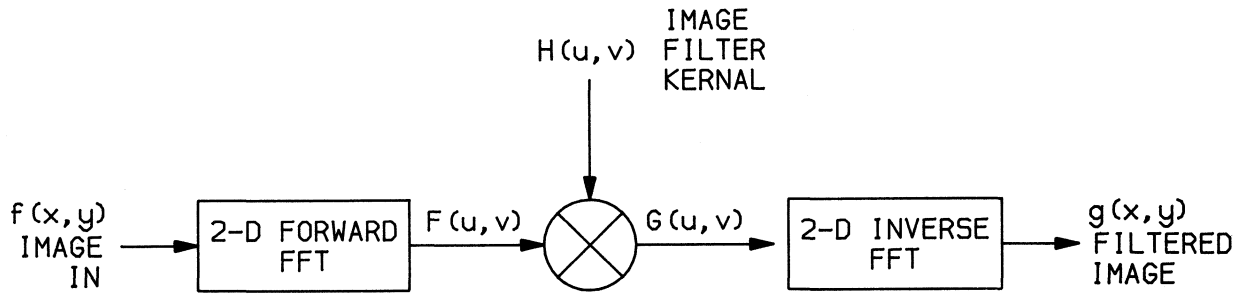


FIGURE 9. 2-D CONVOLUTION WITH FFT.

Zoran's VSP excels at computing the FFT and the vector multiply operations required to multiply the kernel by the image in the frequency domain. For example, the VSP can compute the entire 2-D frequency domain convolution for a 32×32 kernel in less than 20 milliseconds. This is real-time operation for a 30 frames per second image sequence. A VSP system to perform this 2-D convolution is the basic system shown in Figure 7 of the previous section. Multiple VSP configurations can convolve larger images in real-time.

2-D Fast Convolution Performance

| <u>SIZE</u> | <u>TIME(ms)</u> | <u>VSPs</u> |
|-------------|-----------------|-------------|
| 8 × 8 | 1 | 1 |
| 16 × 16 | 4 | 1 |
| 32 × 32 | 18 | 1 |
| 64 × 64 | 80 | 1 |
| 128 × 128 | 570 | 1 |
| 128 × 128 | 300 | 2 |
| 256 × 256 | 2400 | 1 |
| 256 × 256 | 1500 | 2 |

HOMOMORPHIC OR GENERALIZED LINEAR FILTERING

An image can be modeled as a product of its illumination and reflectance components.

$$f(x,y) = i(x,y) \cdot r(x,y)$$

By separating these components and filtering them separately, one can improve the appearance of an image by simultaneous brightness range compression and contrast enhancement. However, because of the product relationship, linear filtering cannot be done directly. Taking the logarithm of the product separates the image into the sum of the logarithms of $i(x,y)$ and $r(x,y)$.

$$\ln f(x,y) = \ln i(x,y) + \ln r(x,y)$$

This function is filtered with a 2-D fast linear convolution via the frequency domain as above. The exponential of the filter output yields the final result.

IMAGE COMPRESSION USING THE 2-D FAST COSINE TRANSFORM

Images require a tremendous number of bits to represent them in their raw form. Fortunately, images contain considerable redundant information, providing an opportunity to reduce the amount of data needed to accurately represent an image. This compression is desirable in a wide variety of applications for reducing storage and transmission bandwidth requirements including:

- Electronic publishing and office automation
- Multimedia CD ROM databases
- Telecommunications, e.g. videoconferencing
- Electronic cameras
- Medical imaging
- Military reconnaissance

There are several techniques for compressing images, including DPCM or predictive coding and transform coding. All depend on the local correlation properties of images. DPCM codes the differences between adjacent pixels. Transform coding concentrates the image energy in a few transform coefficients.

Transform coding (Figure 10) has several desirable characteristics:

- High quality image reconstruction for low bit rate coding
- Robustness to transmission channel errors
- Very few transform coefficients provide an image outline, enabling fast image "browsing" through databases

The main disadvantage of transform coding has been the large computational requirement, particularly the transform itself.

The optimal transform for image compression is the Karhunen Loeve Transform (KLT). However, the KLT requires substantial computation to determine the transform basis vectors, which are unique to each image. Fortunately another unitary transform, the Discrete Cosine Transform (DCT), compresses natural images nearly as well as the KLT and with considerably fewer computations.

Typically, images are coded in 16×16 subblocks by taking the DCT of the subblock. Larger subblocks provide a small incremental image quality because correlation among pixels decreases rapidly beyond 16×16 blocks (Figure 11).

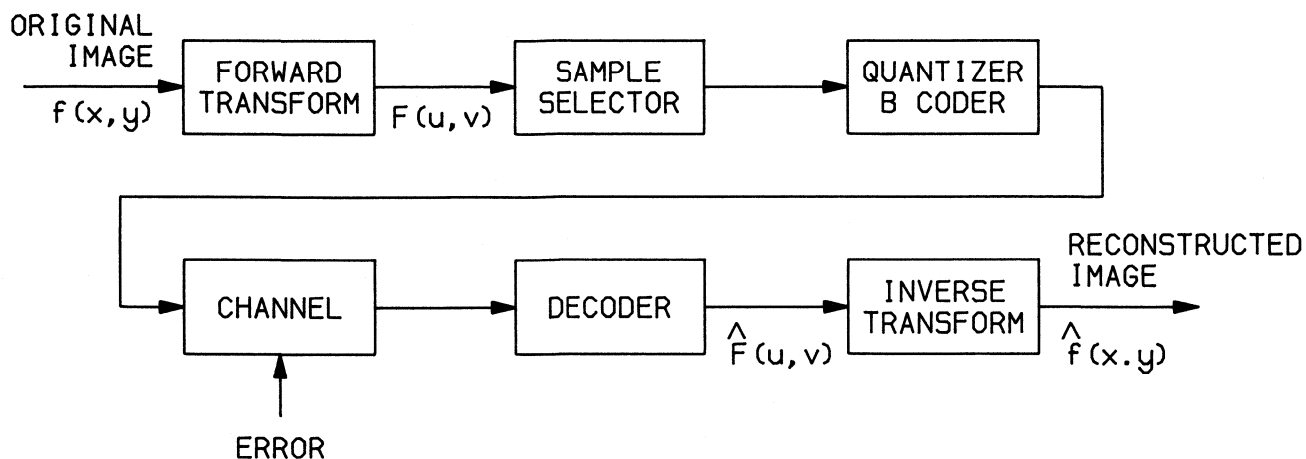


FIGURE 10. TRANSFORM IMAGE CODING SYSTEM.

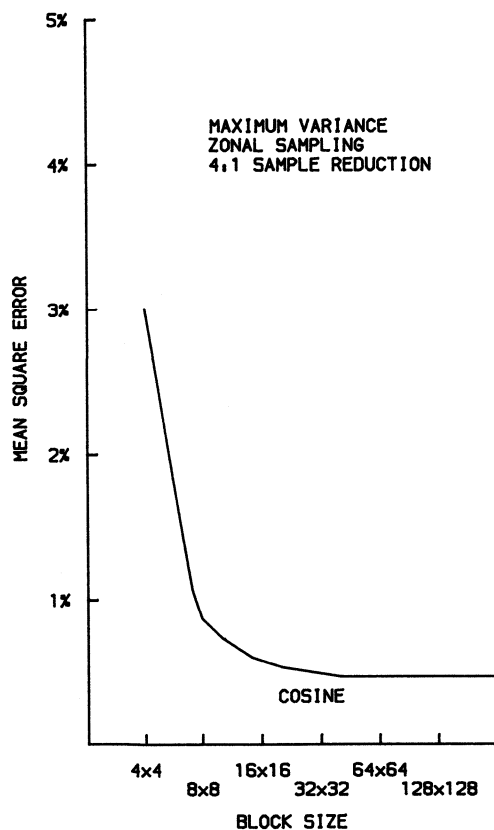


FIGURE 11. MEAN SQUARE ERROR AS A FUNCTION OF BLOCK SIZE.

The DCT can be computed with one of several "fast" algorithms, called Fast Cosine Transform (FCT) algorithms. One of the most efficient was developed by Makhoul and is based on the FFT.

With its high-performance FFT capability, the VSP can compute the FCT quickly using Makhoul's algorithm. The VSP can compute a 16x16 FCT in less than 1 millisecond. It can transform a 256x256 monochrome image in less than 1/4 second or a 512x512 monochrome image in less than 1 second. Multiple processor VSP configurations operate even faster. A dual-VSP system transforms a 512x512 image in less than 1/3 second.

In a typical full-motion video compression system, both the VSP and DFP excel at important functions. (Figure 12). The DFP performs video-rate filtering for front-end functions of decoding and filtering. The VSP performs the FCT on 16x16 blocks and also power computations as part of the conditional replenishment and coefficient selection functions.

2-D Fast Cosine Transform: Subblock Speed

| BLOCK SIZE | TIME(us) | VSPs |
|------------|----------|------|
| 8x8 | 243 | 1 |
| 16x16 | 955 | 1 |
| 16x16 | 500 | 2 |
| 16x16 | 300 | 3 |

2-D Fast Cosine Transform: Image Speed (16x16 Subblocks)

| IMAGE SIZE | TIME(sec) | VSPs |
|------------|-----------|------|
| 256x256 | .25 | 1 |
| 256x256 | .13 | 2 |
| 512x512 | .98 | 1 |
| 512x512 | .75 | 2 |

As in other imaging applications, the VSP brings economical and practical image compression to new markets.

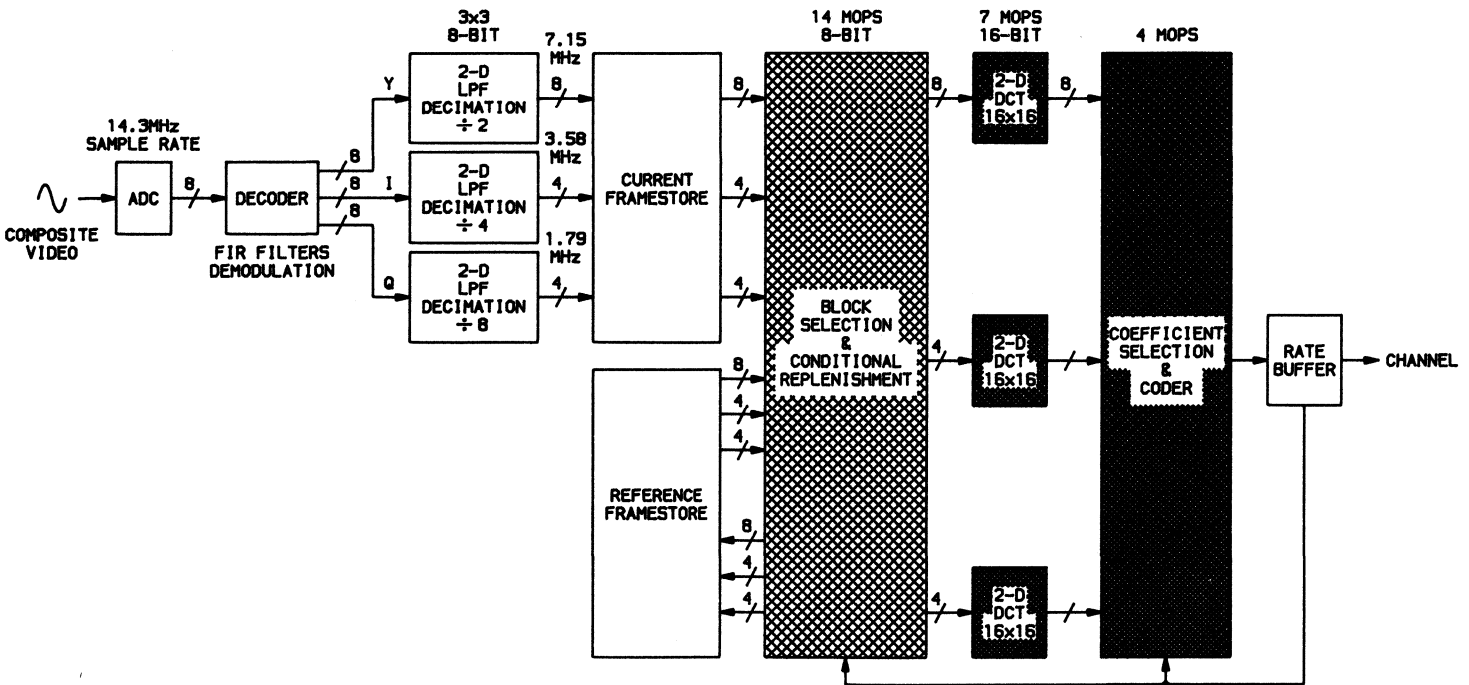


FIGURE 12. IMAGE COMPRESSION USING THE DFP AND VSP.

TYPICAL APPLICATIONS

Electronic Publishing/CD-ROM Databases

As electronic publishing systems move towards integrating text, graphics and natural images, image enhancement and image compression are required. Compression is required to reduce the amount of storage required and the transmission bandwidth required over LANs and PBXs.

For example, a high-quality color image may consist of $4000 \times 4000 \times 3$ 8-bit pixels, requiring an incredible 48 Mbytes of storage. To transfer this image across a 10 Mbps Ethernet LAN requires over 40 seconds. Compressing this image by a factor of 10 reduces the storage requirement to 4.8 Mbytes and the transmission time to four seconds.

The DFP and VSP efficiently filter either monochrome or color images to enhance them. They also provide the high-performance computational capability required for compression techniques such as the Fast Cosine Transform. The FCT has the advantage for electronic publishing and CD-ROM database systems of allowing browsing through the compressed images very quickly at low quality, then progressively updating the selected images to full quality.

MACHINE VISION

Functions such as 2-D convolution and correlation are essential elements of machine vision systems. The large convolution and correlation kernels supported by the DFP and VSP provide more accurate and economical filters, correlators and edge detectors than other implementations.

- **Automatic Inspection** Preprocessing requirements include filtering to enhance images and edge detection to extract features. Template matching of various shapes for feature extraction is a common requirement. These requirements are satisfied in both the spatial and transform domains by Zoran products.
- **Measurement and Gauging** Accurate edge detection is required to determine boundaries for measurement and gauging. Zoran products support accurate, subpixel edge measurement in 1-D and 2-D.
- **3-D Vision Systems** Using structured light and triangulation requires accurate subpixel measurements of light stripe positions. Matched filtering to a Gaussian profile line template is quickly and economically performed with Zoran products.

TELECOMMUNICATIONS

Broadcast video and teleconferencing images are transmitted via satellite, microwave radio and terrestrial communication links. Compression reduces the bandwidth required to transmit these images. The advent of the Integrated Services Digital Network (ISDN) will provide more economical channel bandwidth amenable to transmitting compressed images.

The DFP and VSP provide economical solutions to the filtering and Fast Cosine Transform requirements of image compression schemes.

MEDICAL IMAGING

Nearly all basic image processing techniques are applied in various medical imaging disciplines.

Two-dimensional images are often reconstructed in CT, NMR and ultrasound systems from more primitive 1-D sensor data using FFT techniques. The VSP excels at FFT computations and provides economical solutions for this sector of medical imaging.

These reconstructed images are often enhanced by smoothing to eliminate noise or by sharpening to highlight edges. Both the DFP and VSP are excellent choices for this filtering function. The ability to economically implement larger kernels provides more accurate filters and edge detectors to the medical imager.

Feature extraction and pattern recognition are often used to analyze medical images. The edge detection and correlation functions provided by the VSP and DFP are fundamental to these operations. The efficient, economical 2-D FFT capability of the VSP opens new dimensions for Fourier analysis of medical images.

DIGITAL VIDEO

Digital video systems require FIR filters for applications such as:

- **Composite-to-Component Decoder** Video is generally transmitted in composite form. In a composite signal the luminance (brightness) and color information are combined into a single channel signal. The color information is quadrature modulated onto the color subcarrier. When processing video digitally, the color information must be separated from the luminance and demodulated from the carrier. Operations required for this decoding include band-pass filtering, demodulation, lowpass filtering and decimation. These operations must be performed on data with sample rates between 10 MHz and 20 MHz. The DFP excels at just these kinds of operations.

- **Standards or Sample Rate Conversion** Various video format standards exist such as NTSC composite, PAL composite, RGB component, YIQ component, etc. These standards often have different sample rates and bandwidths. Converting video among these standards requires video rate 1-D filtering with both decimation and interpolation. The DFP family excels at these functions.
- **Enhancement** Video noise reducers require real-time FIR filters. Creating higher resolution video such as for Extended Definition TV requires various FIR filter functions.

ELECTRONIC CAMERAS

Digital electronic cameras require considerable memory to store captured images in raw form. Compression techniques reduce this memory requirement with little or no loss of quality.

The VSP's powerful Fast Cosine Transform capability provides a key image compression function for this application. In addition to the FCT, the VSP performs other necessary compression functions such as filtering and correlation. The high level integration of the VSP provides compression solutions with low electrical power and small physical space. These features are essential to widespread application of electronic cameras.

MILITARY RECONNAISSANCE

Image processing requirements for military reconnaissance span the gamut of imaging techniques. Enhancement in the form of filtering is required to reduce noise. Edge and other feature detection techniques are used to feed pattern recognition algorithms. Compression is used to reduce the bandwidth of communication channels for transmitting the images.

Both the DFP and VSP products are well-suited to performing all these aspects of military reconnaissance imaging. Not only do these products provide new and enhanced processing techniques, but they do this economically and with less power and physical space. Low power and small physical space are essential to building viable airborne reconnaissance systems.

DIGITAL VLSI FOR BROADCAST VIDEO

APPLICATIONS

- COMPOSITE TO COMPONENT DECODERS
 - MATRIX SIGNAL DECODERS
- NOTCH, BANDPASS, LP & HP, COMB FILTERS
- SPECIAL EFFECTS—WIPES, FADES, PAINT
- RESTORATION & ENHANCEMENT OF IMAGES
 - STANDARDS CONVERSION
 - VIDEO COMPRESSION
 - MULTIPATH CANCELLATION
 - TIMEBASE CORRECTORS
 - EDITING EQUIPMENT
- FILM TO TAPE CONVERTERS
- INTERPOLATION/DECIMATION FILTERS

VLSI FOR INDUSTRIAL PROCESS MONITORING

APPLICATIONS

- NON-CONTACT MEASUREMENT
- IMAGE ENHANCEMENT (1-D AND 2-D)
 - MOTION DETECTION/TRACKING
 - SPECTRAL ANALYSIS
- OBJECT RECOGNITION/IDENTIFICATION
 - INSPECTION-EDGE DETECTION
 - TEMPLATE MATCHING
- ULTRASONIC RANGE AND POSITION MEASUREMENT
 - DOPPLER MEASUREMENT
 - CENTROID CALCULATIONS
 - DATA COMPRESSION
 - TEXTURE MAPPING
- PARTICLE ANALYSIS (SIZE AND DISTRIBUTION)
 - DEPTH MEASUREMENT AND 3-D IMAGING
- VIBRATION AND NOISE ANALYSIS/CONTROL
 - ACOUSTIC EMISSION ANALYSIS
 - AIR POLLUTION MEASUREMENT
- ULTRASONIC DEFECT DETECTION
 - EDDY CURRENT ANALYSIS

VSP REMOTE DATA COLLECTION AND REDUCTION**APPLICATIONS**

- GEOPHYSICAL EXPLORATION AND RESEARCH
- SATELLITE RECONNAISSANCE—WEATHER/MILITARY
 - SONAR REQUIREMENTS—SONOBUOYS
 - NAVIGATION AND GUIDANCE SYSTEMS

VLSI DIGITAL SIGNAL PROCESSING SOLUTIONS FOR MEDICAL EQUIPMENT

APPLICATIONS

- COMPUTERIZED TOMOGRAPHY (CT) IMAGING
 - RECONSTRUCTION
 - IMAGE ENHANCEMENT
 - PATTERN RECOGNITION
- NUCLEAR MAGNETIC RESONANCE (NMR) IMAGING
 - RECONSTRUCTION
 - IMAGE ENHANCEMENT
 - PATTERN RECOGNITION
- B-SCAN ULTRASOUND IMAGING
- SPATIAL DECONVOLUTION TO INCREASE RESOLUTION
 - PHASED ARRAY ULTRASOUND IMAGING
 - DOPPLER ULTRASOUND
- DIGITAL TOMOGRAPHIC FILTERING OF RADIOGRAPHS
- IMAGE COMPRESSION FOR STORAGE AND TRANSMISSION (PACS)

DIGITAL VLSI FOR MACHINE VISION

APPLICATIONS

- 1-D, 2-D OR 3-D VISION SYSTEMS
 - AUTOMATIC INSPECTION
 - ROBOTIC VISUAL GUIDANCE
 - MEASUREMENT AND GAUGING
 - EDGE DETECTION
- TEMPLATE MATCHING OR MATCHED FILTERS
 - FEATURE EXTRACTION
- PYRAMID TRANSFORM PROCESSING

GENERAL INFORMATION

Facilities
Reliability and Quality Assurance

FACILITIES

Zoran's wafers are manufactured in its Santa Clara wafer fabrication facility. The fabrication method utilized can be generically described as a two micron, CMOS, double layer metal process. Wafers are produced using the latest process technology and equipment, including direct-step-on-wafer and dry etch lithography.

Each wafer lot withstands a rigid set of in-process inspection and tests during the typical two hundred step procedure. Every wafer is then parametrically tested to assure conformance to Zoran's high acceptance standards.

GENERAL

The successful fabrication of high quality, reliable VLSI products comes about when several important ingredients are in place: an assured design aimed at high reliability, controlled procurement of materials, documented and controlled fabrication steps and a monitor program that assures the quality and reliability of outgoing product.

The task of the Reliability and Quality Assurance program at ZORAN is to assure that all these areas are controlled and that the delivered products conform to the requirements of each order placed with the company. Reliability and Quality Assurance also drives the quality improvement process to optimize product performance and market acceptance.

QUALITY PLANNING

Product quality and reliability depends greatly on assured design rule conformance. ZORAN's quality procedures will assure that the designed quality and reliability of products are sustained by manufacturing operations. Basic responsibility for product quality and defect prevention rests with the individual designers, engineering and production personnel. It is ZORAN's belief that product quality is important for every individual in the

company, as it is only through the integrated effort of all employees that our product will meet the quality objectives of the company.

Reliability and Quality Assurance is responsible for overseeing that the quality programs for monitoring, auditing and final acceptance of material to be shipped are accomplished.

ORGANIZATION

ZORAN has established the Reliability and Quality Assurance department under the Director of Reliability and Quality Assurance, who reports directly to the President. The Director is responsible for developing and implementing the programs, systems and procedures to assure product quality and reliability

performance. He, with the concurrence of the President, has the final authority in decisions regarding design release, shipment of products, and disposition of nonconforming material.

IMPLEMENTING RELIABILITY AND QUALITY ASSURANCE AT ZORAN

ZORAN has established a document control function under Quality Assurance, which is responsible for controlling the creation, revision and distribution of all internal specifications. This assures that all procedures, fabrication operations, designs and the like are carried out according to the latest revisions of ZORAN's specifications.

A program of monitors and audits was instituted for internal and subcontractor operations that allows for early detection, prevention, and correction of all nonconforming materials and products.

A program was created that will allow an effective and timely

feedback for resolution of customer quality or reliability related issues.

ZORAN's standard product screening and quality conformance test program for hermetic IC packages is patterned after the criteria of MIL-STD-883, Class B (see Table).

New processes also have additional qualification tests involving basic circuit components such as transistors, dielectrics and metallization. The stability of these components are monitored using accelerated tests and data will be extrapolated to real device operation conditions. These tests are in addition to the normal lifetests performed on ZORAN's devices.

PRODUCT RELIABILITY AND QUALITY

Failure rate goal:

extrapolated to 55°C less than 200 FITS

Infant mortality goal:

after burn-in at 125°C, 48 hours less than .5%

Outgoing product quality goal less than 500 dpm

LIFETEST DATA AT 150°C

| Device Type | Package Type | Total # Lots | Total # Samples | Device Hours | Total Failures | Failure Rate (FITS)* @ 55°C | | |
|-------------|----------------|--------------|-----------------|--------------|----------------|-----------------------------|--------|-------|
| | | | | | | 0.5eV | 0.65eV | 1.0eV |
| ZR33881 | 68 LCC Ceramic | 2 | 165 | 33,853 | 6 | 1430 | 385 | 27 |

*Fits = Failures in 109 device hours: (90% confidence level)

ZORAN PRODUCT SUMMARY FLOW

| Screen | MIL-STD-883 Test Method/Condition | Product Grade | | | | | |
|--|--|---------------|------------------------------------|--------|------|-------------|-----|
| | | Military | Commercial | | | | |
| Internal Visual | 2010 Condition B | 100% | 100% | | | | |
| High Temperature Storage | 1008 Condition C (24 Hours) | 100% | --- | | | | |
| Temperature Cycle | 1010 Condition C (10 Cycles) | 100% | --- | | | | |
| Constant Acceleration | 2001 Condition D or E (Y ₁ Only) | 100% | --- | | | | |
| Hermeticity Fine Gross | 1014 Condition A or B | 100% | 100% | | | | |
| | Condition C | 100% | 100% | | | | |
| External Visual | 2009 | 100% | 100% | | | | |
| Pre Burn-In Electrical | Per Applicable Zoran Device Specifications at T = +70°C | 100% | 100% | | | | |
| Burn-In | 1015 (T = +150°C Min. or Equivalent) | 100% | 100% | | | | |
| | | 168 Hours | 48 Hours | | | | |
| Post Burn-In PRODUCTION ELECTRICAL Static (DC), Functional and Switching (AC) | <table border="1"> <tr><td>TA</td></tr> <tr><td>Low</td></tr> <tr><td>+ 25°C</td></tr> <tr><td>High</td></tr> </table> | TA | Low | + 25°C | High | - 55°C/100% | --- |
| | | TA | | | | | |
| | | Low | | | | | |
| + 25°C | | | | | | | |
| High | | | | | | | |
| 100% | --- | | | | | | |
| + 125°C/100% | + 70°C/100% | | | | | | |
| Percent Defective Allowable (PDA for DC @ 25°C) | <table border="1"> <tr><td>TA</td></tr> <tr><td>Low</td></tr> <tr><td>+ 25°C</td></tr> <tr><td>High</td></tr> </table> | TA | Low | + 25°C | High | 5% | 10% |
| | | TA | | | | | |
| Low | | | | | | | |
| + 25°C | | | | | | | |
| High | | | | | | | |
| + 55°C/Sample | --- | | | | | | |
| Group A Electrical Static (DC), Functional and Switching (AC) | <table border="1"> <tr><td>TA</td></tr> <tr><td>Low</td></tr> <tr><td>+ 25°C</td></tr> <tr><td>High</td></tr> </table> | TA | Low | + 25°C | High | --- | --- |
| | | TA | | | | | |
| Low | | | | | | | |
| + 25°C | | | | | | | |
| High | | | | | | | |
| + 125°C/Sample | + 70°C/Sample | | | | | | |
| Mark | Zoran Spec | 100% | 100% | | | | |
| + 25°C Electrical Production Monitor | Zoran Spec | Sample | Sample | | | | |
| Final Visual | Zoran Spec | 100% | 100% | | | | |
| Quality Conformance Sample Selection (Group B, C & D) | 5005 | Yes | Group B, Sub 2-5 Group C, Sub 2 | | | | |
| Quality Shipping Inspection (Visual/Outgoing Acceptance) | Zoran Spec | 100% | 100% | | | | |

*Not required if product meets company infant mortality goal

SALES OFFICES

Zoran Corporation

3450 Central Expressway
Santa Clara, CA 95051
(408) 720-0444

Zoran Corporation

200 Reservoir Street, Suite 300
Needham Heights, MA 02194
(614) 449-5990

Zoran Corporation

1821 Walden Office Square, Suite 426
Schaumburg, IL 60173
(312) 397-0710

Zoran Microelectronics Ltd.

Advanced Technology Center
P.O. Box 2495
Haifa, 31024 Israel
011-972-4-533-175

ZORAN

Zoran Corporation

3450 Central Expressway
Santa Clara, CA 95051
(408) 720-0444
FAX: (408) 720-9875

