Firefly Embedded MicroController ASICs

(Incorporating the ARM7TDMI Core)

DS4874 - 1.0 September 1998

INTRODUCTION

Mitel Semiconductor has combined advanced, compact ASIC technology with MicroController design expertise and the ARM7TDMI processor core to produce the uniquely versatile Firefly range of Embedded MicroController ASICs.

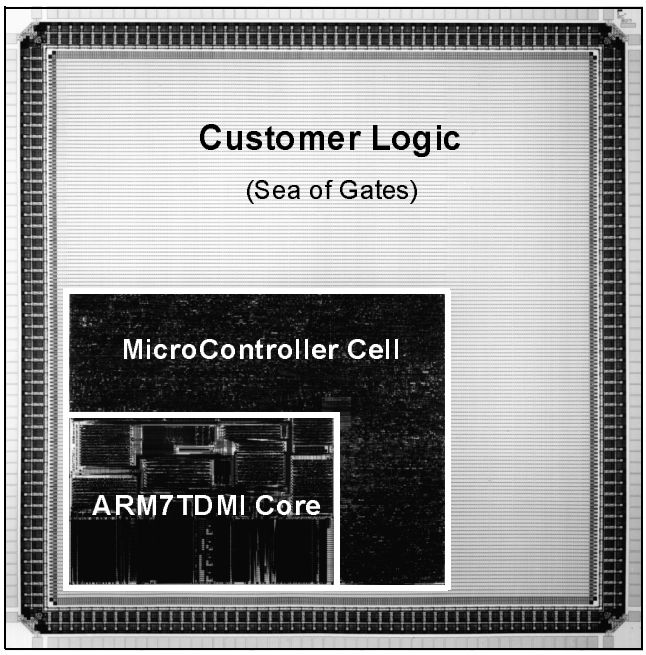
The ARM7TDMI from Advanced RISC Machines is a 32-bit RISC processor, specifically developed for deeply embedded applications, such as wireless communications, networking, media and mass-storage. It is a member of the Mitel Semiconductor SystemBuilder™ library of embeddable macrofunctions.

A wide variety of Firefly Embedded MicroController cores can be assembled by Mitel Semiconductor. In every case, an internal multi-master bus is used to connect the ARM7TDMI core to a suitable set of local peripherals. These peripherals (e.g. timers, interrupt controllers, memory interfaces) are selected from Mitel Semiconductor's own library of fully-verified on-chip macrofunctions.

The ASIC technology used to develop Mitel's MicroController ASICs also allows designers to add their own logic (implemented in a sea-of-gates) around the Firefly MicroController core, together with on-chip RAM and ROM blocks. The amount of RAM, ROM, customer logic and I/O determines the size of array base used for each design.

Standard ASIC Die sizes are used whenever possible, to reduce tooling charges and prototype lead times.

This document presents the MF1 as an example of a Firefly Core, but others are also available; new peripheral functions and new MicroController cores can readily be created in response to specific customer needs.



Example Firefly MicroController ASIC

FEATURES

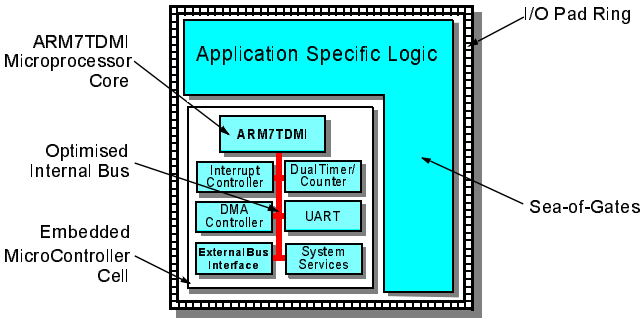
- Uses the industry standard ARM7TDMI processor core:
  - Small size
  - Low power consumption
  - High performance
  - High code density
  - 16- and 32-bit instruction sets
- Available in both Embedded Array and Standard Cell technologies
- Available on two Mitel CMOS processes:
  - 0.6µm (90K series) - for 5V or 3.3V operation
  - 0.35µm (200 series) - 2V and 3.3V operation
- Supported by Mitel's full SystemBuilder™ capability:
  - Extensive library of proven, re-usable macros
  - Simulation Models and Test Vectors supplied for MicroController cores and all other macros
  - Documented Design Flows for Industry Standard CAE tools, including 'How-to' Development Guides
  - World-wide Design Centre support
  - Dedicated Software and Hardware Applications Engineering support
- Software Interface Function (SIF) Library:
  - Supports all MicroController Peripherals and many other SystemBuilder™ macros
  - Facilitates all low-level initialisation and control tasks

BENEFITS

- Mitel Semiconductor's MicroController expertise enables:
  - High levels of integration
  - Fast time-to-market
  - Right-first-time
- SystemBuilder™ methodology and efficient ASIC technologies lead to effective solutions for volume applications, where cost, performance and power consumption are all critical
- Low implementation risk through the re-use of proven macrofunctions
- Embedded Array ASICs enable short prototype cycle times. By fixing the embedded functions and pin-out early in the design, the base wafers can be manufactured in parallel with verification and fine-tuning of the design, so only the metal layers remain to be fabricated after design sign-off. This gives a prototype cycle time equivalent to a Gate Array, and the ability to create upgrades or variants rapidly, by using existing base wafer stocks.
- Standard Cell ASICs offer the opportunity to fine-tune the chip size for the precise application. This minimises cost and power dissipation, and maximises performance, but requires a full Standard Cell prototype cycle time. A Sea-of Gates block can be designed in, to allow rapid metal-layer-only modification, using existing base wafer stocks.

Firefly ASIC Cores

The Firefly range of the MicroController cores combines the widely adopted **ARM7TDMI** RISC processor core with Mitel Semiconductor's MicroController integration experience and Embedded Array ASIC technologies to deliver MicroController engines for customer-specific designs.



Example Firefly MicroController ASIC

Firefly MicroController ASIC cores from Mitel Semiconductor integrate system support functions, such as memory and peripheral interfaces, interrupt controllers, timers, UARTs and DMA controllers within a modular bus-based architecture to allow the rapid development of application-specific MicroControllers, optimised to meet performance, size and power constraints.

Mitel's ASIC technologies allow designers to surround the chosen Firefly core with blocks of **RAM** and **ROM**, complex re-usable functions (e.g proven, synthesisable IPR blocks) and their own application-specific logic. Either Embedded Array or Standard Cell techniques may be used to develop Firefly MicroController ASICs.

In Embedded Arrays, the gate array area consists of a dense array of core cells, which has been designed to allow very efficient metal interconnections, including over-cell routing, resulting in high utilisation. The core architecture also allows highly efficient register file RAM to be implemented.

Both dual- and single-port RAM, and ROM can be implemented in a gate array area. This is suitable for modest-sized memory blocks, and minimises the engineering charge, as it requires no custom embedded blocks.

Single- and dual-port embedded RAM and ROM blocks are available for larger memories in both Embedded Array or Standard Cell ASICs, and for applications which require very fast access times.

Mitel's Array and Standard Cell Libraries also offer Phase Locked Loops, Oscillators and a range of analogue I/O functions, including ADCs, DACs, Power-on Reset cells and Reference cells, any of which can be included in any MicroController ASIC design.

Currently, designs of up to 3 million equivalent gates complexity are being undertaken on Mitel's 0.35µm CMOS technology.

A Hierarchy of Benefits

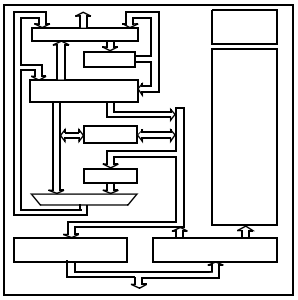
At the heart of the current range of Firefly MicroController ASICs is the ARM7TDMI, the world's most successful 32-bit **Embedded RISC Processor Core**:

ARM7TDMI Microprocessor Core

Small area

Low power

High code density



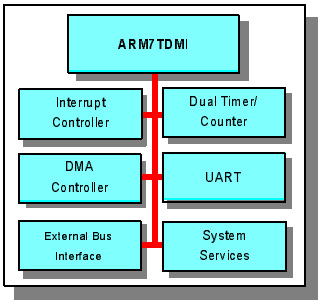
To extend the capabilities, and to simplify the use of the ARM7TDMI Core, Mitel has developed a set of versatile peripherals, connected to the processor core by an internal multi-master bus, which it uses to build **Firefly Embedded MicroController Cores**:

The Firefly MF1 MicroController Core

Proven design

Highly integrated

Well documented



To tailor Firefly ASICs for their precise needs, Customers can finally add their own logic, optionally with other custom blocks from Mitel, such as compact ROM/RAM, to produce their ideal **Embedded MicroController ASICs**.

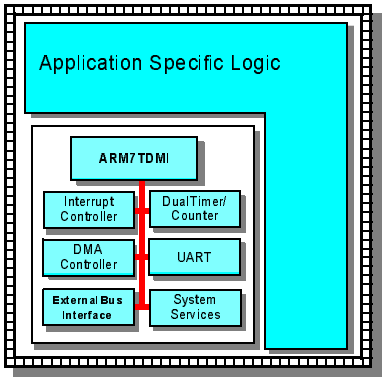
The customer can gain confidence from the extensive re-use of well-proven functions, giving fast time-to-market and a high probability of first time success:

Firefly MicroController ASIC

Customer specific

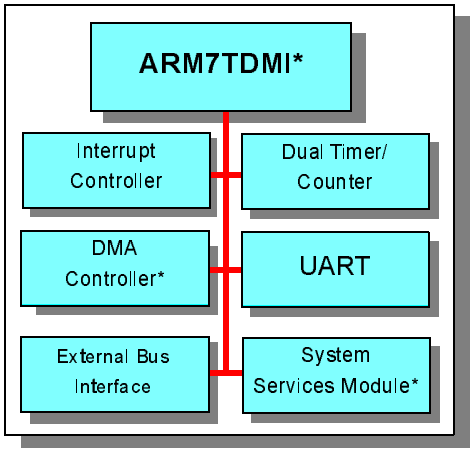
Fast time to market

Low cost



Mitel Semiconductor Firefly ASIC Design Hierarchy

Example: Mitel Firefly MF1 MicroController Core



(Note: Asterisks denote functions with Bus Master capability)

MicroController Core Block Diagram

Shown above is the simplest Firefly MicroController core, the MF1. This example incorporates:

- The ARM7TDMI processor core
- A Serial I/O port (UART)
- A programmable External Bus (memory, etc.) interface
- A programmable interrupt controller
- Two dual 32-bit timer/counters
- A Direct Memory Access (DMA) controller providing 2 fly-by channels or 1 memory-memory channel
- A test/diagnostic interface (System Services Module)

Architectural Overview

Mitel's Firefly MicroController cores are designed around a modular 32-bit bus architecture developed by Mitel in collaboration with Advanced RISC Machines. This internal, multi-master bus architecture has been developed to facilitate fully-testable, reliable and debuggable integrated MicroController products.

A range of embedded peripheral macrocells, a few of which are listed in the example above, has been designed by Mitel. Each macrocell must comply with the bus interface specification and operate independently of all other macrocells. All peripherals are fully testable through the internal bus.

To facilitate ease-of-use, a library of Software Interface Functions (SIFs) is supplied, which greatly improves programmer productivity in setting up and controlling all peripheral macrocells.

This architecture allows Mitel and its customers to produce right-first-time ASIC MicroController systems efficiently, meeting the stringent cost and time-to-market demands of today's embedded control market.

The functional blocks in this example are described in detail below.

The Internal Module Bus

The existence of an internal standard bus is a key element of Mitel's success in quickly developing new embedded MicroController cores to meet specific demands. By adhering to rigorous bus communication standards for both master and slave modules, it is possible to re-use any module in a new design without modifications to resolve timing, or other inter-module conflicts.

It is not only modules that can be re-used: all the test programs, characterisation results and software routines, created the first time a module was used, can also be used again without modification. This gives an enormous advantage in time-to-market terms.

The standard bus architecture also simplifies the task of designing new peripheral functions, when needed, quickly and with a high right-first-time success rate.

System Services Module

The System Services Module is responsible for all the administrative tasks associated with the bus, both in normal operation and for test and diagnostic purposes. It performs the following functions:

- Bus mode control (RESET, RUN, TEST, etc.)
- Bus master arbitration
- System address decoding for peripheral functions
- Diagnostic broadcast of bus status
- External bus master access
- Manufacturing test control
- Top-level System Configuration

Bus Interfaces

(Memory/Peripheral Controller and Up-Integration Module)

The Memory/Peripheral Controller (MPC) acts as the main gateway between the internal and external bus systems of an embedded MicroController core. The external bus has to be capable of interfacing both to the multitude of separate devices which could be used in conjunction with a MicroController ASIC, and with the logic and memory located elsewhere on the same chip. This second requirement is serviced by the Up-Integration Module (UIM).

The MPC provides the following functions:

- Support for 8-, 16- and 32-bit data transfers and memory widths
- Flexible address interface – providing typically 4 (maximum 7) memory areas, each of 4 Mbytes
- Accesses to memories (ROM, SRAM), and peripherals (e.g. ADCs, DACs, other communications interfaces)
- Generation of control signals to access external components (chip-selects, write-enables, output-enables, etc.)
- Dynamic bus sizing, so that only accesses of the correct width (i.e. 8-, 16- or 32-bits) are directed to each type of external component

- Generation of programmable wait-states for accessing external devices of different speeds
- Generation of programmable stop-wait states, allowing for the slow turn-off time of some external devices (e.g. ROM)
- External memory accesses with zero wait-states at up to the maximum system clock speed, assuming suitable memory devices are used
- Fly-by DMA operations. Internal -> external, external -> internal, and external -> external modes are supported
- RAM and ROM swapping after initialisation, so that interrupt routines and other time-critical software can operate from fast memory once the system has booted

The simplistic approach to integrating customer functions on-chip would be simply to connect them to the off-chip address/ data busses, and control signals, from the MPC. However, all these interconnections would then carry the capacitive overhead of the bond pads and external loads, and their performance would be both impaired and variable.

Mitel's UIM consists of multiplexers and tristatable I/O cells which disconnect on-chip customer logic (when required) from the external world, thus minimising power consumption and ensuring that anticipated performance is obtained, regardless of what else is present on the circuit board. If desired, these signals can still be transmitted from the external address and data lines, providing an invaluable diagnostic tool for the system software developer.

The UIM also allows on-chip ROM to be disabled and replaced by external ROM, so changes to ROM-based code can be tried out before new masks are made.

Universal Asynchronous Receiver Transmitter (UART)

The UART provides an asynchronous full-duplex RS232-type channel, which supports both software and hardware flow control mechanisms. The Receive and Transmit channels are double-buffered. Each UART may be polled, or can use an interrupt scheme for module bus transfers. An internal Baud-rate generator provides selectable data rates derived from on- or off-chip sources. Directly-triggered DMA transfers involving the UART are also possible, without the need for CPU intervention. Features include:

- Full duplex operation, independent transmit and receive channels
- 7 or 8-bit serial data length. 1 or 2 stop bits
- Even, odd or no parity generation
- Internally selectable baud rate generation, derived from either the system clock or an external clock source
- XON/XOFF (s/w) or RTS/CTS (h/w) flow control
- Double-buffered transmit and receive channels
- Software polling to determine channel status
- Optional interrupt generation on transmit channel becoming empty
- Optional interrupt generation on receive channel becoming full
- Detection of parity, overrun, and framing errors on receive channel, with optional interrupt generation
- Support for modem signals RTS, CTS and DCD with edge detection and optional interrupt generation
- Additional signals DTR, DSR and RI also supported
- Digital input filter to improve noise immunity

Interrupt Controller

The ARM7TDMI core accepts two types of interrupt: Normal (IRQ) and Fast (FIQ). All generated Interrupts can be switched between types, depending upon the relative priorities required.

The Interrupt Controller is the central control logic that assigns priority levels, and handles all interrupt request signals. External Interrupts can be set for edge or level sensitivity, and have a polarity option. To minimise interrupt latency, there is a hard-wired priority scheme covering all channels and both types; alternatively this can be over-ridden, and the priority assessment handled in software.

Features include:

- 32 independently controlled interrupt channels
- Hardware priority encoding for FIQ and IRQ interrupts to indicate the highest priority active channel for each interrupt type
- Generation of either a FIQ or IRQ request
- Independent masking of the channel interrupt source
- Response to edge-triggered or level-sensitive sources
- Sensitivity to both active-low and active-high interrupts

DMA Controller

Two DMA engines are available; these may be configured as single channels for flyby DMA transfers between off-chip requestors and either on-chip or off-chip locations, or as a channel-pair to provide a memory-to-memory DMA capability between any locations in the memory space. Features include:

- Maximum transfer rates of 100 MBytes per second (single addressing), 50 MBytes per second (dual addressing) at 25 MHz: zero wait-state transactions
- 32 bit (4 GByte) addressing range: address increment, decrement and hold
- Data transfer sizes of 8, 16, and 32 bits (statically sized)
- 16 bit (65536 item) maximum transfer count
- Transfers can be triggered by software
- Maskable level- or edge-sensitive hardware transfer triggers, with selectable polarity
- Maskable hardware transfer acknowledge signals, with selectable polarity
- Block and Packet mode transfers supported
- Wait-state insertion, when indicated by slow memory
- Auto-initialisation of channels on completion

- Chained DMA transfers supported for scatter-gather operations
- Fixed channel priority
- Optional bus locking to prevent interruption of DMA services by other bus masters
- Abort mechanism for illegal access with interrupt generation and transfer halt.

Timer/Counters

Four independent 32-bit timer/counters, each with an 8-bit prescaler capability are provided by two Dual Timer/ Counters modules. These are synchronous to the system clock, and may be polled or set up to generate interrupts on over-run and auto-reload. The main features of each Timer/ Counter module are:

- Two, independently controlled Timer/Counter elements
- Prescalers to generate a Timer Clock signal from the system clock
- Prescale count of 8 bits
- Division ratio selection by software
- Multiple Timer/Counter modes:
  - countdown to zero
  - free running
  - reload and count on trigger
  - pulse width modulation
- Fully software programmable time-out period
- Maskable interrupt on time-out

Power Distribution

Mitel Embedded MicroController ASICs use a grid methodology for power distribution. This grid, which is automatically constructed during layout, uses metal layers one and three for horizontal power rails and metal layer two for vertical connections. Metal layer four can also be used on 0.35µm for vertical connections, in order to increase the effective gate density of larger chips.

Clock Distribution

Mitel supports a number of clock distribution methodologies, selected according to the particular design and the CAD tools being used by the designer. For small designs with a light clock load, a single, large buffer may be sufficient. For large designs, with large clock loads, a clock grid or clock tree is recommended, to avoid metal electromigration in the clock network. Clock trees can either be synthesised or manually specified as a clock hierarchy by the designer. Mitel's clock grid methodology uses up to three stages of buffering, where each stage drives a grid which feeds the next stage. The final stage grid is a starting point for routing to the actual clocked inputs.

Other Embedded MicroController Cores

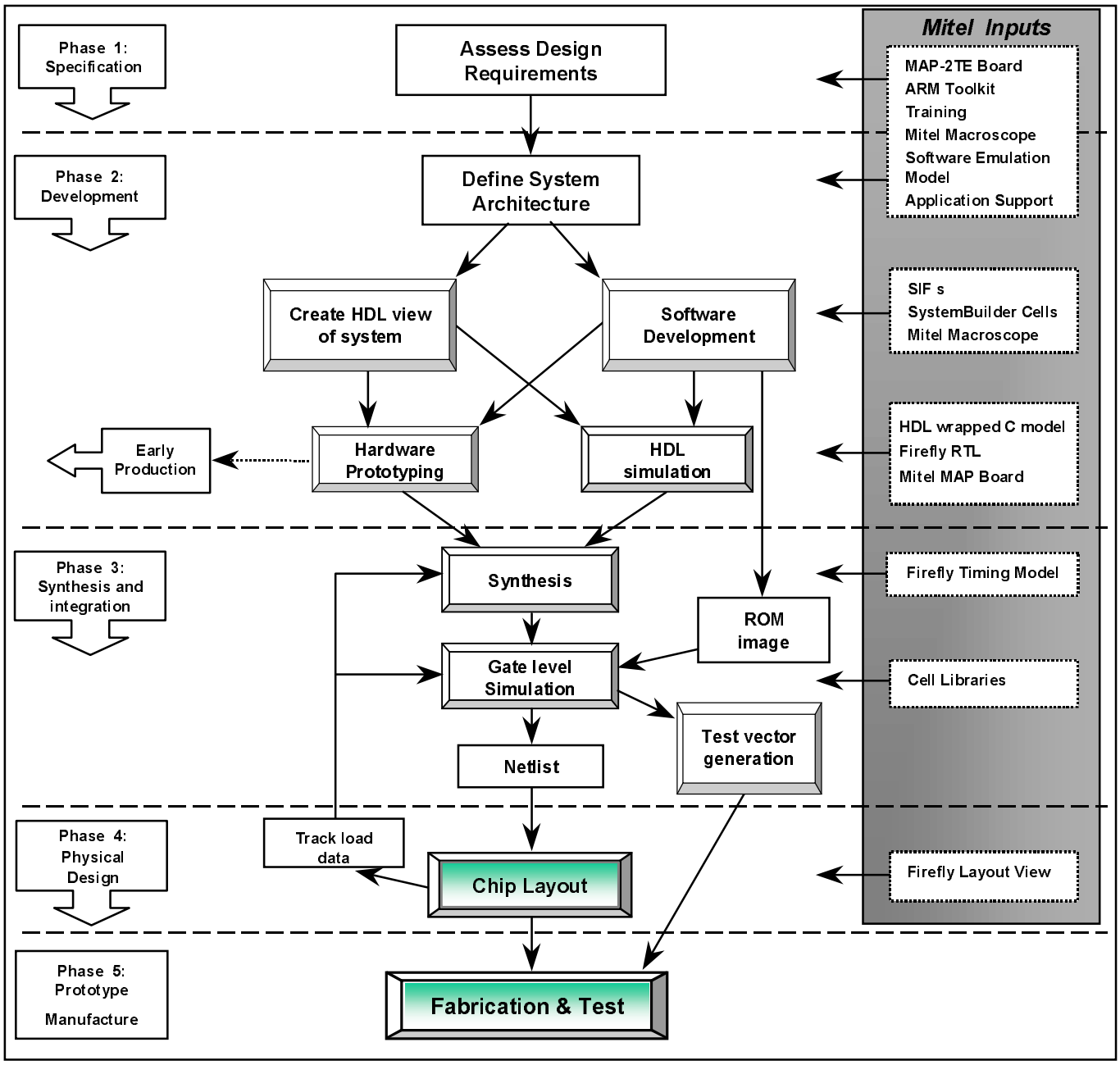
Mitel is able to produce new Embedded MicroController cores rapidly, by re-using peripheral blocks from a library which is continually being extended as new peripherals are identified. The library contains a number of other functions which are not used in the earlier example, including:

- Watchdog Timer
  - Driven from the system clock
  - Programmable primary time-out period, ending in a processor interrupt
  - System RESET is invoked if the watchdog timer is not re-started within the programmable secondary timeout period after the interrupt was issued
  - An external control signal enables/disables the timer
- Programmable Peripheral Interface
  - 8 data lines
  - Each line:
    - can be individually set, reset or read
    - can be defined as input, output or bidirectional
  - All lines can be combined to form a byte-wide parallel port
  - Static, Strobed or Interrupt-driven I/O modes
  - An interrupt can be generated if any line changes state, effectively adding up to 8 more external interrupt sources
- Power Control Module
  - Provides individual clock control to all other modules on the bus, to minimise operational power consumption
  - Initiates a Standby mode, suspending all bus and module activity until a hardware interrupt is received
  - Initiates a Sleep mode for the ARM7TDMI core, which the remains inactive until an interrupt occurs
- SIM Card Interface
  - Serial half duplex port
  - Parity generation and checking in hardware: protocols and timing in software
  - Bit timing compliant with ISO 7816-3
  - remains inactive until an interrupt is received
- Keypad Scanner
  - Supports 6 x 6, 7 x 5 or 8 x 8 matrix keypads, or any subset of these

Further MicroController ASIC cores are continually being evaluated and developed in conjunction with customers, to address particular application needs. Recent examples include PCI, USB, Global Positioning by Satellite, Digital Cellular Communications and Digital Imaging.

Unless subject to commercial restrictions, any extra functions which are developed will be made available to other developers of Mitel Embedded MicroController ASICs.

Firefly ASIC Design Flow



Firefly ASIC Design Flows are available for both Cadence (VHDL and/or Verilog) and Mentor (VHDL) systems, both using Synopsys Logic Synthesis. Typically the designer will be provided with suitable models of the Embedded MicroController Core and any other embedded hard-wired functions, in addition to the appropriate Mitel ASIC libraries on CD-ROM. These libraries cover both Embedded Arrays and Standard Cell on both 0.6µm and 0.35µm processes

With Mitel's library elements, designers will be able to design, synthesise and validate Firefly ASIC designs (see ASIC Simulation Design Flow, below), before passing the netlist data to Mitel for lay-out. The Firefly MicroController Core is supplied with its own Test-bench, so designers only need to develop tests for the additional circuitry they have designed.

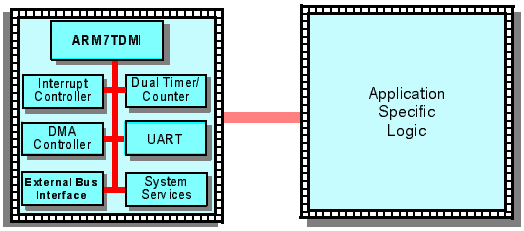
After layout by the nearest Mitel Semiconductor Design Centre, the designer can perform final simulations prior to prototype Manufacture.

As an alternative to the pure CAE approach, Mitel also supports a Hardware prototyping Design Flow (see above). Both flows produce the same end product - a fully-integrated application specific MicroController.

Hardware Prototyping Design Approach

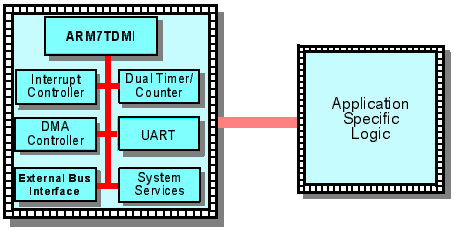
Designers often face the contradictory demands of early proof-of-feasibility of a design, and low ultimate cost for high-volume applications. The Hardware Prototyping Design Approach provides a route which allows early validation of design decisions, but is focused on producing a highly-integrated end product.

Initially the design may be prototyped for validation purposes using an FPGA for the application-specific logic, and a MicroController ASIC development chip from Mitel. Mitel also offers Prototyping Boards, described later in this Datasheet.



Hardware Prototype System (FPGA)

Once design validation is performed, the FPGA may be converted to a Mitel ASIC for pilot production in the shortest possible time frame.

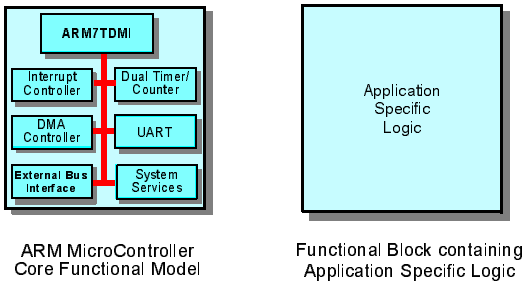


Pilot System (Gate Array)

Ultimately the two devices may be up-integrated into one device so that the required cost targets may be achieved.

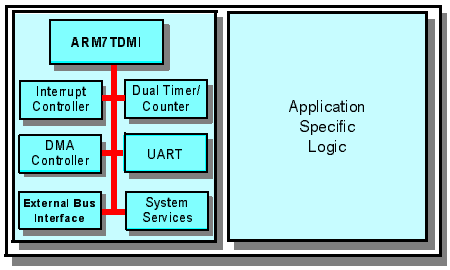
ASIC Simulation Design Approach

In many cases, the performance constraints of the system under design are such that no hardware prototyping strategy is feasible. The ASIC Simulation Design Approach allows designers to combine models of the MicroController and of their own application-specific logic within a software simulation environment.



Prototype System (Simulation)

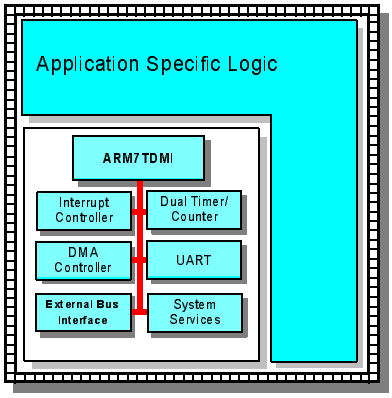
Once the function of the application-specific logic is proven against the available test cases or test data streams, the description of the logic may be modified to integrate it closely with the MicroController ASIC core, taking into account issues such as bus interfacing and manufacturing test.

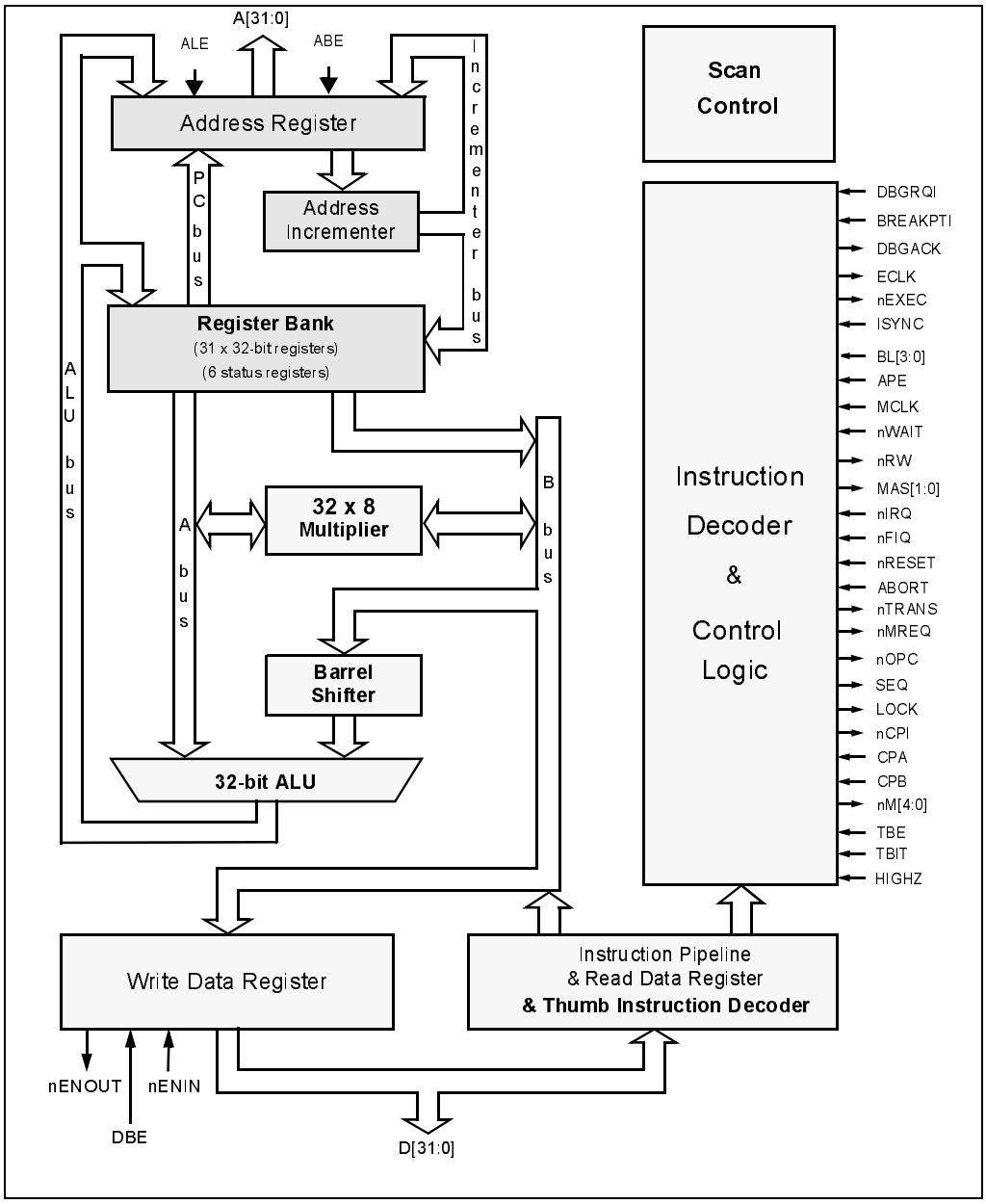


Integrated System (Simulation)

Upon design verification, the application-specific MicroController ASIC may be fabricated as a single device, allowing the required cost targets to be achieved.

Both Design Approaches produce the same final Firefly Embedded MicroController ASIC:





ARM7TDMI Architecture

The ARM7TDMI is a member of the Advanced RISC Machines (ARM) family of general purpose 32-bit microprocessors, which offer high performance for very low power consumption and price.

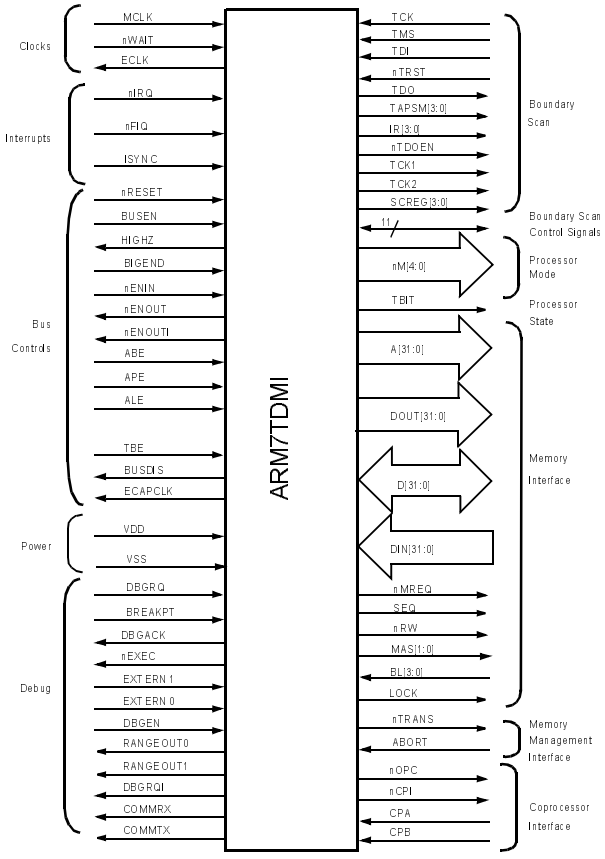
It is a static design, consuming dynamic power only when the clock is active. Additional features include a fast multiplier (32x8bit), debug support, and the 'Thumb' extension.

'Thumb' is an additional 16-bit instruction set which is decompressed into normal 32-bit ARM instructions in real time by the ARM7TDMI core. It allows cheaper 16-bit memory to be used for non-critical code segments.

The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler than those of microprogrammed Complex Instruction Set Computers. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective core.

Pipelining is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The ARM memory interface has been designed to allow the performance potential to be realised without incurring high costs in the memory system. Speed-critical control signals are pipelined to allow system control functions to be implemented in standard low-power logic, and these control signals facilitate the exploitation of the fast local access modes offered by industry standard dynamic RAMs.



ARM7TDMI Interfaces

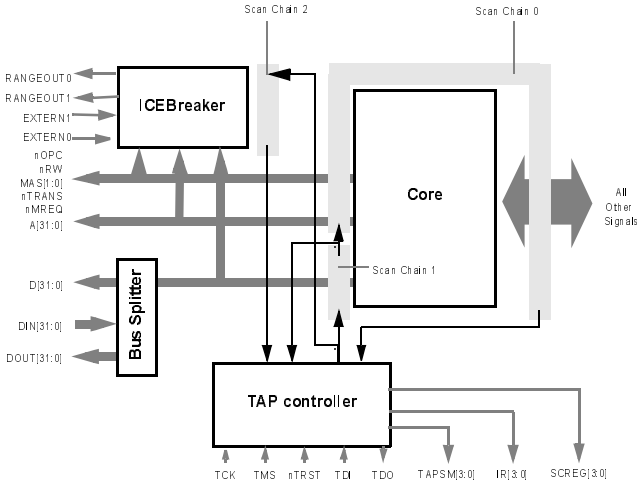
Debugging Embedded ARM7TDMI Systems

The ARM7TDMI processor supports ARM's EmbeddedICE debug methodology. A special 'ICEbreaker' unit is provided to store watchpoint/breakpoint information, and to halt program execution at a specific instruction fetch (breakpoint) or data access (watchpoint). Register or memory contents can then be examined, and optionally modified, then execution can be single-stepped, or allowed to run to the next break/watchpoint.

Communication with EmbeddedICE is through a JTAG-like interface. Four scan chains are provided, to:

- test both the ARM7TDMI core and the surrounding logic
- program the EmbeddedICE breakpoint/watchpoint registers
- debug program execution in the ARM7TDMI
- test another scan chain, typically around the entire pad ring of the Embedded MicroController device.

EmbeddedICE is non-intrusive, so it supports breakpointing of ROM-based code. The JTAG interface is typically controlled by the ARM Software Development Toolkit, operating on a PC, Sun or HP platform, via ARM Ltd's EmbeddedICE interface Unit



ARM7TDMI Debug Interface

ARM7TDMI Instruction Set Architecture

The ARM7TDMI processor employs a unique architectural strategy known as THUMB, which makes it ideally suited to high-volume applications with memory restrictions, or applications where high code density is essential.

The THUMB Concept

The key idea behind THUMB is that of a super-reduced instruction set. Essentially, the ARM7TDMI processor has two instruction sets. The original ARM 32-bit instruction set is available, unchanged, but ARM have taken the 36 most frequently used 32-bit instructions and compressed them to 16 bits, with some restrictions, e.g. fewer accessible registers and smaller immediate values stored in the instruction word.

Thumb 16-bit instructions are decoded in hardware without any loss of processor speed, and are presented to the ALU as conventional 32-bit instructions for execution.

Two additional instructions switch the processor between ARM and Thumb modes, whenever the programmer decides that a change is appropriate.

ARM standard 32-bit instruction set

Mnemonic	Instruction	Action
ADC	Add with carry	Rd := Rn + Op2 + Carry
ADD	Add	Rd := Rn + Op2
AND	AND	Rd := Rn AND Op2
B	Branch	R15 := address
BIC	Bit Clear	Rd := Rn AND NOT Op2
BL	Branch with Link	R14 := R15, R15 := address
BX	Branch and Exchange	R15 := Rn, T bit := Rn[0]
CDP	Coprocessor Data Processing	(Coprocessor-specific)
CMN	Compare Negative	CPSR flags := Rn + Op2
CMP	Compare	CPSR flags := Rn - Op2
EOR	Exclusive OR	Rd := (Rn AND NOT Op2) OR (op2 AND NOT Rn)
LDC	Load coprocessor from memory	Coprocessor load
LDM	Load multiple registers	Stack manipulation (Pop)
LDR	Load register from memory	Rd := (address)
MCR	Move CPU register to coprocessor register	cRn := rRn {<op>cRm}
MLA	Multiply Accumulate	Rd := (Rm * Rs) + Rn
MOV	Move register or constant	Rd : = Op2
MRC	Move from coprocessor register to CPU register	Rn := cRn {<op>cRm}
MRS	Move PSR status/flags to register	Rn := PSR
MSR	Move register to PSR status/flags	PSR := Rm
MUL	Multiply	Rd := Rm * Rs
MVN	Move negative register	Rd := 0xFFFFFFFF EOR Op2
ORR	OR	Rd := Rn OR Op2
RSB	Reverse Subtract	Rd := Op2 - Rn
RSC	Reverse Subtract with Carry	Rd := Op2 - Rn - 1 + Carry
SBC	Subtract with Carry	Rd := Rn - Op2 - 1 + Carry
STC	Store coprocessor register to memory	address := CRn
STM	Store Multiple	Stack manipulation (Push)
STR	Store register to memory	<address> := Rd
SUB	Subtract	Rd := Rn - Op2
SWI	Software Interrupt	OS call
SWP	Swap register with memory	Rd := [Rn], [Rn] := Rm
TEQ	Test bitwise equality	CPSR flags := Rn EOR Op2
TST	Test bits	CPSR flags := Rn AND Op2

THUMB 16-bit instruction set

Mnemonic	Instruction	Action	Lo/Hi register operands	Condition codes set
ADC	Add with Carry	Rd := Rd + Rs + C	Lo	Yes
ADD	Add	Rd := Rn + Rs	Lo/Hi	Yes*
AND	AND	Rd := Rd AND Rs	Lo	Yes
ASR	Arithmetic Shift Right	Rd := Rd ASR Rs	Lo	Yes
B	Unconditional branch	PC := PC +/- Offset11	Lo	
B*xx*	Conditional branch	PC := PC +/- Offset8	Lo	
BIC	Bit Clear	Rd := Rd AND NOT Rs	Lo	Yes
BL	Branch and Link	PC := PC +/- Offset LR := PC + 2		
BX	Branch and Exchange	PC := Rs	Lo / Hi	
CMN	Compare Negative	Rd + Rs	Lo	Yes
CMP	Compare	CPSR flags :=Rd - Rs	Lo / Hi	Yes
EOR	EOR	Rd := Rd EOR Rs	Lo	Yes
LDMIA	Load multiple	Stack manipulation (Pop)	Lo	
LDR	Load word	Rd32 := [Rb + Immediate5]	Lo	
LDRB	Load byte	Rd8 := [Rb + Immediate5]	Lo	
LDRH	Load halfword	Rd16 := [Rb + Immediate5]	Lo	
LSL	Logical Shift Left	Rd := Rd << Rs	Lo	Yes
LDSB	Load sign-extended byte	Rd8 := [Rb + Immediate5]	Lo	
LDSH	Load sign-extended half-word	Rd16 := [Rb + Immediate5]	Lo	
LSR	Logical Shift Right	Rd := Rd >> Rs	Lo	Yes
MOV	Move register	Rd := Immediate8	Lo / Hi	Yes*
MUL	Multiply	Rd := Rs * Rd	Lo	Yes
MVN	Move Negative register	Rd := NOT Rs	Lo	Yes
NEG	Negate	Rd := -Rs	Lo	Yes
ORR	OR	Rd := Rd OR Rs	Lo	Yes
POP	Pop registers	[SP] ++ := Rlist (LR)	Lo	
PUSH	Push registers	Rlist (LR):= [SP] --	Lo	
ROR	Rotate Right	Rd := Rd ROR Rs	Lo	Yes
SBC	Subtract with Carry	Rd := Rd -Rs - NOT C	Lo	Yes
STMIA	Store Multiple	[Rb]++ := Rlist	Lo	
STR	Store word	[Rb + Immediate5] := Rd32	Lo	
STRB	Store byte	[Rb + Immediate5] := Rd8	Lo	
STRH	Store halfword	[Rb + Immediate5] := Rd16	Lo	
SWI	Software Interrupt	OS call		
SUB	Subtract	Rd := Rd - Immediate8	Lo	Yes
TST	Test bits	CPSR flags :=Rd AND Rs	Lo	Yes

The THUMB set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because THUMB code operates on the same 32-bit register set as ARM code.

THUMB code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system.

THUMB's Advantages

THUMB instructions operate with the standard ARM register configuration, allowing excellent interoperability between ARM and THUMB states. Each 16-bit THUMB instruction has a corresponding 32-bit ARM instruction with the same effect on the processor model.

The major advantage of a 32-bit (ARM) architecture over a 16-bit architecture is its ability to manipulate 32-bit integers with single instructions, and to address a large address space efficiently. When processing 32-bit data, a 16-bit architecture will take at least two instructions to perform the same task as a single ARM instruction.

However, not all the code in a program will process 32-bit data (for example, code that performs character string handling), and some instructions, like Branches, do not process any data at all.

If a 16-bit architecture only has 16-bit instructions, and a 32-bit architecture only has 32-bit instructions, then overall the 16-bit architecture will have better code density, and better than one half the performance of the 32-bit architecture. Clearly 32-bit performance comes at the cost of code density.

THUMB breaks this constraint by implementing a 16-bit instruction length on a 32-bit architecture, making the processing of 32-bit data efficient with a compact instruction coding. This provides far better performance than a 16-bit architecture, with better code density than a 32-bit architecture.

THUMB also has a major advantage over other 32-bit architectures with 16-bit instructions. This is the ability to switch back to full ARM code and execute at full speed. Thus critical loops, for applications such as fast interrupts and DSP algorithms, can be coded using the full ARM instruction set, and linked with THUMB code. The overhead of switching from THUMB code to ARM code is folded into sub-routine entry time. Various portions of a system can be optimised for speed or for code density by switching between THUMB and ARM execution as appropriate.

Operating Modes

ARM7TDMI supports seven modes of operation. Mode changes may be made under software control, or may be brought about by external interrupts or exception processing. Most application programs will execute in User mode. The non-user modes - known as *privileged modes* - are entered in order to service interrupts or exceptions, or to access protected resources.

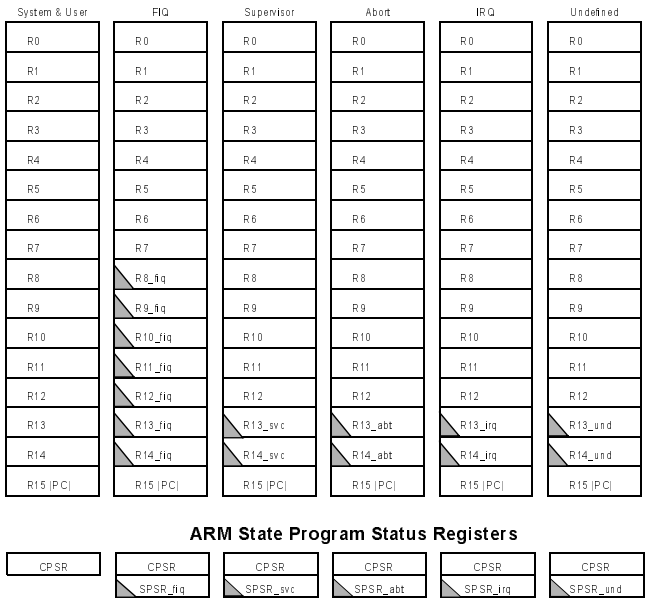
ARM7TDMI Operating Modes

User (usr)	The normal ARM program execution state
FIQ (fiq)	Designed to support a data transfer or channel process
IRQ (irq)	Used for general-purpose interrupt handling
Supervisor (svc)	Protected mode for the operating system
Abort mode (abt)	Entered after a data or instruction prefetch abort
System (sys)	A privileged user mode for the operating system
Undefined (und)	Entered when an undefined instruction is executed

Register Sets

In ARM (32-bit) state, 16 general registers and one or two status registers are visible at any one time. In privileged (non-User) modes, mode-specific banked registers are switched in. The table below shows which registers are available in each mode: the banked registers are indicated by a triangle.

ARM State General Registers and Program Counter



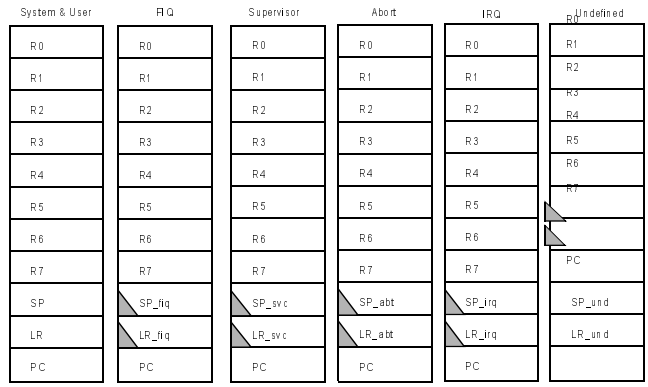
ARM State Program Status Registers

= *banked register*

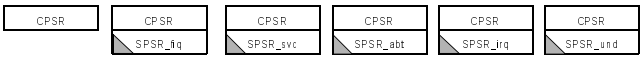
Key:
CPSRCurrent Program Status register

The THUMB (16-bit) state register set is a subset of the ARM state set. The programmer has direct access to eight general registers, R0-R7, as well as the Program Counter (PC), a stack pointer register (SP), a link register (LR), and the CPSR. There are banked Stack Pointers, Link Registers and Saved Program Status Registers (SPSRs) for each privileged mode. This is shown below:

THUMB State General Registers and Program Counter



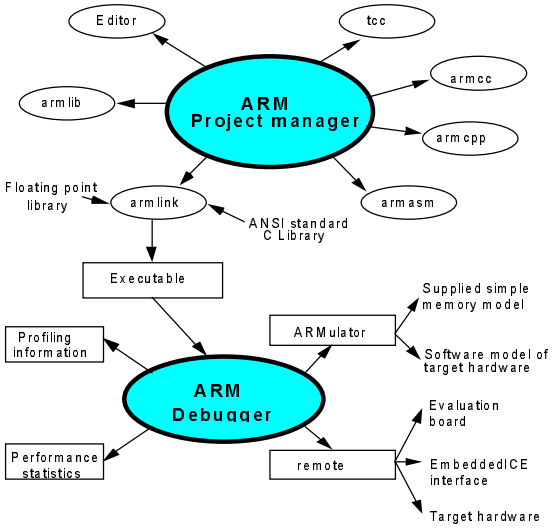
THUMB State Program Status Registers



= *banked register*

Software Development and Debug

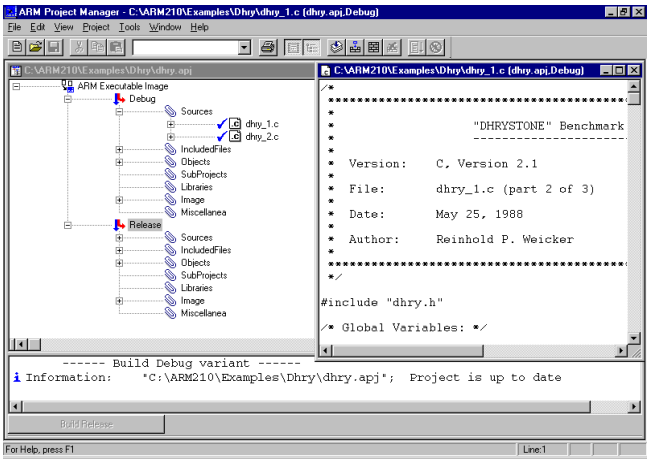
Both software and hardware development tools are available to aid the designer in the development and debugging of the target application. The Software Development Toolkit contains an industry-standard optimising 'C' compiler and Assembler for both ARM and Thumb code, a linker, and a Windows-based debugger.



Software Development Toolkit - Tool Chain

The task of creating the project environment within which the user can write, maintain and build program code is significantly eased by the inclusion of the ARM Project Manager, an integrated development environment.

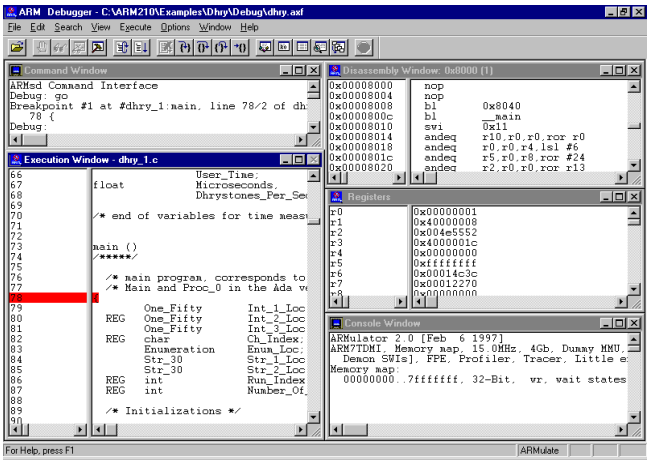
The Project Manager provides the mechanisms required to configure and build complex Embedded Applications. The interdependencies between source files in a project are automatically detected at build time, removing the need to write complex make files.



Software Development Toolkit - Project Manager Interface

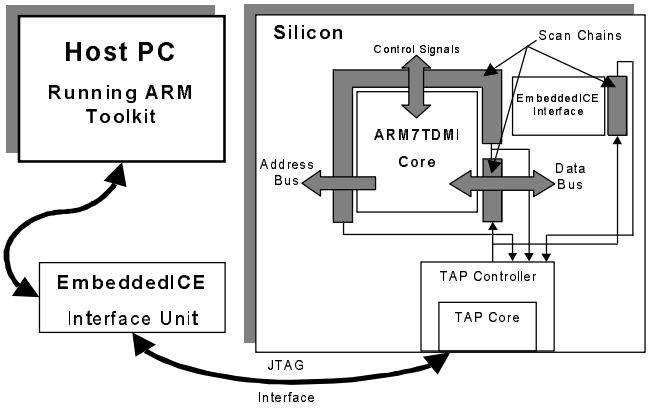
The ARM Windows Debugger is a source-level debugger, with optional Source/Assembler interleaving. 'C' and Assembler may be single-stepped and breakpointed. Breakpoints may be simple or compound, occurring only on a specific set of events. Variables and memory may be watched during execution. Modifying or setting memory and variables to specific values can trigger breakpoints.

The debugger can interface to a software simulation environment, such as the ARMulator software emulator, and also to a target-resident debug monitor via a serial port or other communications channel.



Software Development Toolkit - Debugger Interface

Alternatively, the debugger may connect directly to the processor via the EmbeddedICE JTAG Debugger interface unit in the target hardware environment. For MicroController ASICs where the board-level hardware and interface software are still being developed, this represents a particularly attractive debug option. The correct operation of the ARM7TDMI processor and other on-chip resources may be verified prior to the availability of debugged boards and software.



ARM7TDMI MicroController JTAG Debug Interface

Software Interface Functions

In order to facilitate the writing of driver code for the individual interfaces of the MicroController ASIC cores, a library of low-level Software Interface Functions (SIFs) is provided, which may be called explicitly from 'C' code. The SIFs are a collection of robust library routines to interface to the macrocells of the ARM MicroController ASIC cores. The library is built on a layered approach, as is common in the industry (e.g. ISO OSI Reference model) and it incorporates recognized principles of software engineering, such as abstraction, high cohesion and low coupling. The use of the Software Interface Functions allows the firmware writer to produce interface code more quickly, and with reduced risk of error. Additionally, the use of the functions leads to better documented and hence more easily maintained code. The optimising compiler will eliminate any redundancy in the resultant code.

There are many advantages of using the SIFs.

- They provide the user with an easy and efficient way of using the Firefly MicroController cores.
- They provide automatic bounds checking.
- They are written in ANSI 'C' and are directly callable from assembly language.
- They provide efficient code for benchmarking.
- They provide automatic address checking.
- The function names are self documenting and unambiguous.
- They have a low number of parameters and are not data dependent i.e. they do not have to operate on input data to decide the actions to perform.
- They provide the user with a memory-to-speed trade-off choice at compile time, by electing to use either #define Macros or callable Functions.

In summary the advantages of using the SIFs are that the code can be developed using the 'C' functions to exploit the potential of type, access and bounds checking. The selected files may then be compiled with either in-line macros or functions, as the user requires for the final application.

Documentation

Each Firefly MicroController ASIC core is fully documented. The documentation is provided in the form of a Firefly Core Design Manual. For each macrofunction, the manual includes:

- an Interface Specification
- an Operation Description
- a Programmer's Model
- a Timing Specification, where applicable

In addition, Mitel provides an ARM7TDMI Core Manual, which describes in detail the architecture of the core, the Programmers Model and both the 32- and 16-bit Instruction Sets.

Design Support

Mitel Semiconductor offers fully flexible design support, allowing customers a wide choice of design interfaces. Each customer design is supported fully by an Applications Engineering group, with CAE support from the local Mitel ASIC Design Centre.

In addition, a series of 'How-To' guides is available to ease the process of learning how to integrate a Firefly MicroController core into an ASIC design.

The process incorporates a Design Audit procedure to verify compliance with the customer's specification, and to ensure manufacturability. The procedure includes three design reviews, held at key stages of the design process to control device performance and timescales.

Design Review 1: Held at the beginning of the design cycle, to check and agree on performance, packaging, specification and design time scales

Design Review 2: Held after logic simulation, but prior to layout, to ensure satisfactory functionality, gross timing performance and fault coverage

Design Review 3: Held after layout and post-layout simulation to provide verification of satisfactory performance after the insertion of actual track loads. This is the final check of all device specifications prior to prototype manufacture.

Design Kits

Firefly MicroController ASIC products are offered within the standard ASIC design flows supported by Mitel Semiconductor.

Features of design kits include:

- 'How-To' design guides
- full top-down design flow support
- behavioural and timing models of the MicroController core
- Synopsys Design Compiler timing model
- sign-off simulation by the customer on Cadence Verilog, Mentor Quicksim, and Synopsys VSS
- support for floorplanning
- full test patterns (95% coverage) for both serial scan and parallel access to the ARM7TDMI processor
- direct routes to layout and test

Software Development Tools

Mitel supplies the ARM Software Development Toolkit to support development of application code for Firefly ASICs. This toolkit, which runs on PC, Sun and HP platforms, includes:

- Project Management Tools
- 'C' Compilers for both Instruction Sets
- Assemblers for both Instruction Sets
- Linker

- Debugger
- A 'C' Emulation Model of the ARM7TDMI Core
- ARM's Angel™ Debug Monitor

The Debugger operates equally well with a software model or real hardware as the target. Mitel develops a 'C' model for each of the Firefly MicroController Cores, which extends the power of the Toolkit when using a software target. This power is further extended by Mitel's unique 'Macroscope' Tool, which gives the user full visibility of the status of all internal registers in all the MicroController Core's peripherals.

Hardware Development Support

Mitel provides powerful support for the Hardware Prototyping design approach (described earlier) by offering development chips and development boards. For example, the Firefly MF1 Core is supported by the MF1 Developer Chip, which forms the basis of the Mitel MAP-2TE Hardware Prototyping Board. MAP-2TE is a basic, expandable system, containing:

- the MF1 Developer Chip (MVT905001)
- a 512kByte block of SRAM
- sockets for adding a second 512kByte SRAM block
- the Angel™ Debug Monitor, pre-loaded into a Flash ROM
- a JTAG-type connector to support ARM's EmbeddedICE™ debug methodology
- a RS232-level UART interface
- a 96-way Expansion Socket, for attaching customer-specific prototype circuitry
- a 25MHz system clock oscillator, with the option of using an external clock
- manual Reset and Interrupt buttons

The board is supplied with the full library of MF1 Core SIFs, together with full documentation on the use of SIFs and a set of code examples.

The combination of MAP-2TE, the customer's expansion board, Mitel's SIF library and the ARM Software Toolkit allows application code to be developed and debugged efficiently, using the Angel™ debug monitor.

To exploit the added facilities offered by EmbeddedICE, an EmbeddedICE Interface unit is required. This is also available from Mitel.

Please consult Mitel for information on MAP Kits to support other Firefly Cores.

Experience

Mitel Semiconductor has been an ARM semiconductor partner since 1992. As such, the company has developed significant expertise in the integration and manufacture of ARM-based designs. The company has worked closely with Advanced RISC Machines to develop reuse methodologies and internal bus standards for ARM-based designs. The company also manufactures ARM-based standard product MicroControllers, in addition to ARM MicroController ASICs developed for key customers in markets such as cellular communications, networking and mass-storage.

This experience in both the hardware and the software aspects of ARM-based design and integration is available through the regional design centres of Mitel Semiconductor, with additional support provided from a dedicated applications engineering team.

Manufacturing

MicroController ASICs are manufactured in Mitel Semiconductor's state-of-the-art facility near Plymouth, England. This facility is purpose-built, and is equipped with the latest automated technology for 8-inch wafer processing. This equipment utilises mini-environments, together with the use of SMIF boxes, to achieve ultra-clean processing conditions. Computer-Aided Manufacturing ensures production efficiency. In addition to the world class wafer fabrication facility, the probe and final test areas are equipped with the latest analog and digital testers. Mitel Semiconductor is committed to continuous investment to provide state-of-the-art CMOS ASICs.

A qualified second source for this silicon process is also available.

Availability

The Mitel Firefly ASIC capability is available for design today on both 0.6µm and 0.35µm processes, and in both Embedded Array and Standard Cell technologies.

Notes

ASIC SYSTEMS DESIGN CENTRES

UNITED KINGDOM: Swindon, Tel: (01793) 518000 Fax: (01793) 518411. **UNITED STATES OF AMERICA**: San Jose, CA, Tel: (408) 451-4700 Fax: (408) 451-4710. Irvine, CA, Tel: (714) 852-3900 Fax: (714) 852-3910. **FRANCE**: Les Ulis Cedex, Tel: (1) 69 18 90 00 Fax: (1) 64 46 06 07. **JAPAN**: Tokyo, Tel: (03) 5276-5501 Fax: (03) 5276-5510.

CUSTOMER SERVICE CENTRES

- **FRANCE & BENELUX** Les Ulis Cedex
- Tel: (1) 69 18 90 00 Fax: (1) 64 46 06 07
- **JAPAN** Tokyo Tel: (03) 5276-5501 Fax: (03) 5276-5510
- **KOREA** Seoul Tel: (2) 5668141 Fax: (2) 5697933
- **NORTH AMERICA** Scotts Valley, USA
- Tel: (408) 438 2900 Fax: (408) 438 5576/6231
- **SOUTH EAST ASIA** Singapore
- Tel:(65) 333 6193 Fax: (65) 333 6192
- **SWEDEN** Stockholm Tel: 46 8 702 97 70 Fax: 46 8 640 47 36
- **TAIWAN, ROC** Taipei Tel: 886 2 25461260 Fax: 886 2 27190260
- **UK, EIRE, DENMARK, FINLAND & NORWAY**
- Swindon Tel: (01793) 518000 Fax: (01793) 518582

These are supported by Agents and Distibutors in major countries worldwide.