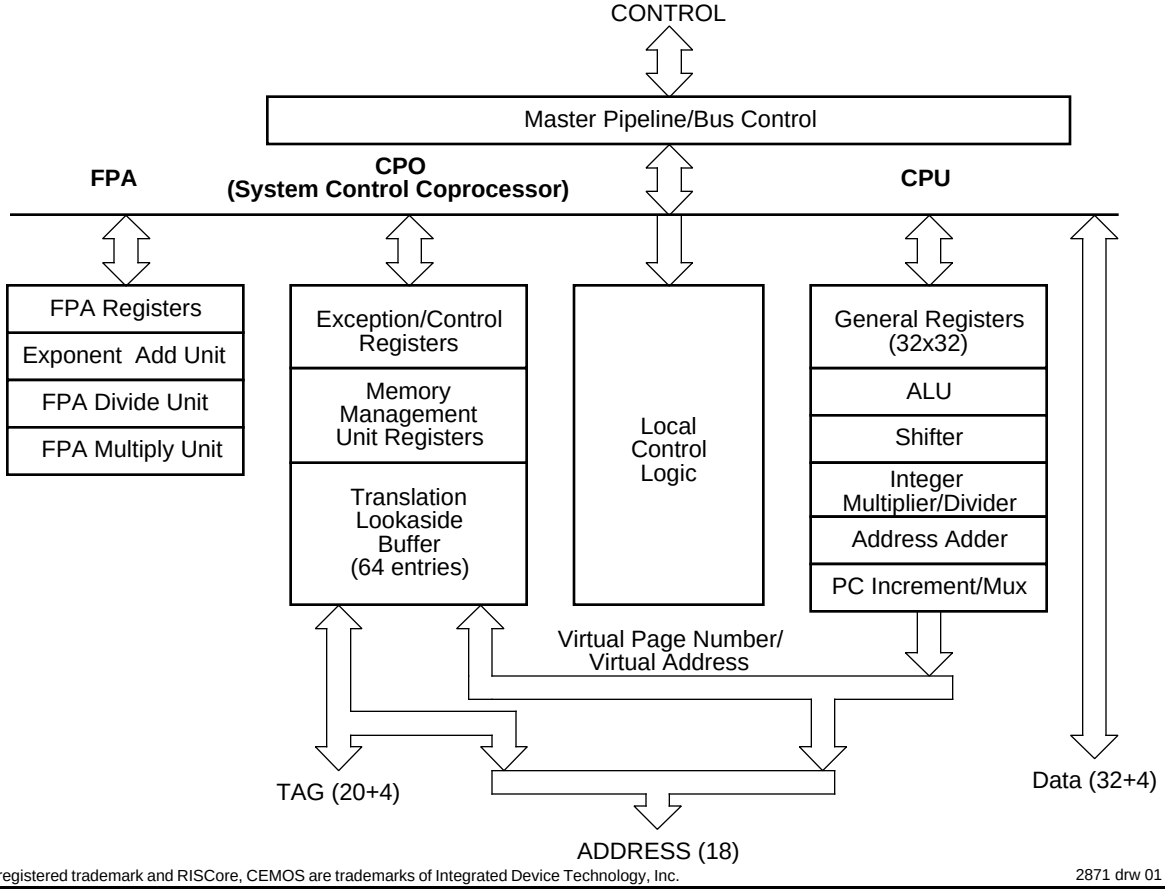
## FEATURES:

- Efficient Pipelining—The CPU's 5-stage pipeline design assists in obtaining an execution rate approaching one instruction per cycle. Pipeline stalls and exceptions are handled precisely and efficiently.
- On-Chip Cache Control—The IDT79R3500 provides a high-bandwidth memory interface that handles separate external Instruction and Data Caches ranging in size from 4 to 256kBs each. Both caches are accessed during a single CPU cycle. All cache control is on-chip.
- On-Chip Memory Management Unit—A fully-associative, 64-entry Translation Lookaside Buffer (TLB) provides fast address translation for virtual-to-physical memory mapping of the 4GB virtual address space.
- Dynamically able to switch between Big- and Little- Endian byte ordering conventions.
- Optimizing Compilers are available for C, FORTRAN, Pascal, COBOL, Ada, PL/1 and C++.
- 20 through 40MHz clock rates yield up to 32VUPS sustained throughput.
- Supports independent multi-word block refill of both the instruction and data caches with variable block sizes.
- Supports concurrent refill and execution of instructions.
- Partial word stores executed as read-modify-write.
- 6 external interrupt inputs, 2 software interrupts, with single cycle latency to exception handler routine.
- Flexible multiprocessing support on chip with no impact on uniprocessor designs.
- A single chip integrating the R3000 CPU and R3010 FPA execution units, using the R3000A pinout.
- Software compatible with R3000, R2000 CPUs and R3010, R2010 FPAs.
- TLB disable feature allowing a simple memory model for Embedded Applications.
- Programmable Tag bus width allowing reduced cost cache.
- Hardware Support of Single- and Double-Precision Floating Point Operations that include Add, Subtract, Multiply, Divide, Comparisons, and Conversions.
- Sustained Floating Point Performance of 11 MFlops single precision LINPACK and 7.3MFLOPS double precision
- Supports Full Conformance With IEEE 754-1985 Floating Point Specification
- 64-bit FP operation using sixteen 64-bit data registers
- Military product compliant to MIL-STD 833, class B

## IDT79R3500 PROCESSOR






2871 drw 01

## DESCRIPTION:

The IDT79R3500 RISC Microprocessor consists of three tightly-coupled processors integrated on a single chip. The first processor is a full 32-bit CPU based on RISC (Reduced Instruction Set Computer) principles to achieve a new standard of microprocessor performance. The second processor is a system control coprocessor, called CP0, containing a fully-associative 64-entry TLB (Translation Lookaside Buffer), MMU (Memory Management Unit) and control registers, supporting a 4GB virtual memory subsystem, and a Harvard Architecture Cache Controller achieving a bandwidth of 320MBs/second using industry standard static RAMs. The third processor is the Floating Point Accelerator which performs arithmetic operations on values in floating-point representations. This processor fully conforms to the requirements of ANSI/IEEE Standard 754-1985, "IEEE Standard for Binary Floating-Point Arithmetic." In addition, the architecture fully supports the standard's recommendations.

The programmer model of this device will be the same as the programmer model of a system which uses a discrete IDT79R3000 with the IDT79R3010: 32 integer registers, 16 floating point registers; co-processor 0 registers; floating point status and control register; RISC integer ALU; Integer Multiply and Divide ALU; Floating Point Add/Subtract, Multiply, and Divide ALUs. The device pipeline will be the same as for the IDT79R3000, as will the co-processor 0 functionality. No new instructions have been introduced. Pin compatibility extends to AC and DC characteristics, software execution and initialization mode vector selection.

This data sheet provides an overview of the features and architecture of the IDT79R3500 CPU, Revision 3.0. A more detailed description of the operation of the device is incorporated in the *R3500 Family Hardware User Manual*, and a more detailed architectural overview is provided in the *MIPS RISC Architecture* book, both available from IDT. Documentation providing details of the software and development environments supporting this processor are also available from IDT.

### IDT79R3500 CPU Registers

The IDT79R3500 CPU provides 32 general purpose 32-bit registers, a 32-bit Program Counter, and two 32-bit registers that hold the results of integer multiply and divide operations. Only two of the 32 general registers have a special purpose: register r0 is hardwired to the value "0", which is a useful constant, and register r31 is used as the link register in jump-and-link instructions (return address for subroutine calls).

The CPU registers are shown in Figure 2. Note that there is no Program Status Word (PSW) register shown in this figure: the functions traditionally provided by a PSW register are instead provided in the Status and Cause registers incorporated within the System Control Coprocessor (CP0).

## FPA REGISTERS

The IDT79R3010A FPA provides 32 general purpose 32-bit registers, a Control/Status register, and a Revision Identification register.

Floating-point coprocessor operations reference three types of registers:

- Floating-Point Control Registers (FCR)
- Floating-Point General Registers (FGR)
- Floating-Point Registers (FPR)

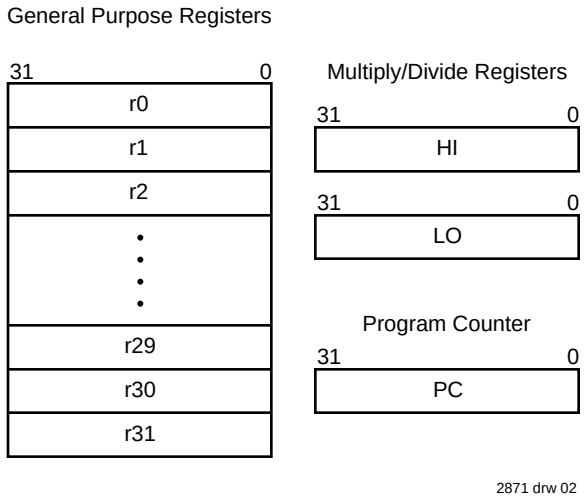



**Figure 2. IDT79R3500 CPU Registers**

### Floating-Point General Registers (FGR)

There are 32 Floating-Point General Registers (FGR) on the FPA. They represent directly-addressable 32-bit registers, and can be accessed by Load, Store, or Move Operations.

### Floating-Point Registers (FPR)

The 32 FGRs described in the preceding paragraph are also used to form sixteen 64-bit Floating-Point Registers (FPR). Pairs of general registers (FGRs), for example FGR0 and FGR1 (Figure 3) are physically combined to form a single 64-bit FPR. The FPRs hold a value in either single- or double-precision floating-point format. Double-precision format FPRs are formed from two adjacent FGRs.

### Floating-Point Control Registers (FCR)

There are 2 Floating-Point Control Registers (FCR) on the FPA. They can be accessed only by Move operations and include the following:

- Control/Status register, used to control and monitor exceptions, operating modes, and rounding modes;
- Revision register, containing revision information about the FPA.

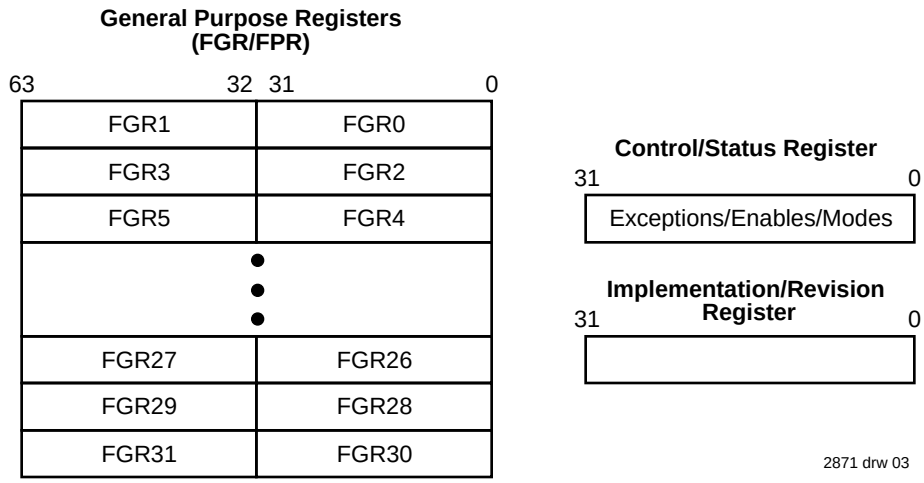



**Figure 3. FPA Registers**

## Instruction Set Overview

All IDT79R3500 instructions are 32 bits long, and there are only three instruction formats. This approach simplifies instruction decoding, thus minimizing instruction execution time. The IDT79R3500 processor initiates a new instruction on every run cycle, and is able to complete an instruction on almost every clock cycle. The only exceptions are the Load instructions and Branch instructions, which each have a single cycle of latency associated with their execution. Note, however, that in the majority of cases the compilers are able to fill these latency cycles with useful instructions which do not require the result of the previous instruction. This effectively eliminates these latency effects.

The actual instruction set of the CPU was determined after extensive simulations to determine which instructions should be implemented in hardware, and which operations are best synthesized in software from other basic instructions. This methodology resulted in the IDT79R3500 having the highest performance of any available microprocessor.

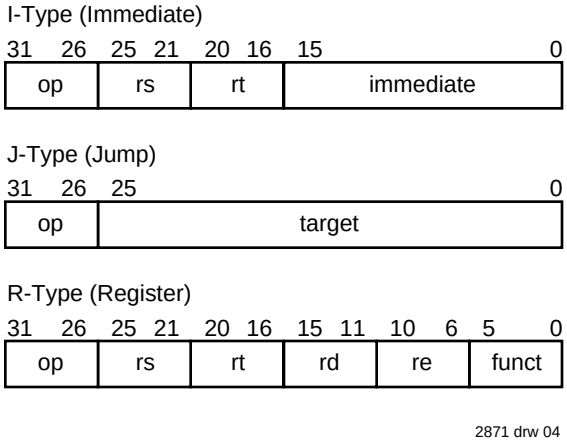



**Figure 4. IDT79R3500 Instruction Formats**

The IDT79R3500 instruction set can be divided into the following groups:

- **Load/Store** instructions move data between memory and general registers. They are all I-type instructions, since the only addressing mode supported is base register plus 16-bit, signed immediate offset.
  The Load instruction has a single cycle of latency, which means that the data being loaded is not available to the instruction immediately after the load instruction. The compiler will fill this delay slot with either an instruction which is not dependent on the loaded data, or with a NOP instruction. There is no latency associated with the store instruction.
  Loads and Stores can be performed on byte, half-word, word, or unaligned word data (32-bit data not aligned on a modulo-4 address). The CPU cache is constructed as a write-through cache.
- **Computational** instructions perform arithmetic, logical and shift operations on values in registers. They occur in both R-type (both operands and the result are registers) and I-type (one operand is a 16-bit immediate) formats. FP computational instructions perform arithmetic operations on floating point values in the FPA registers. Note that computational instructions are three operand instructions; that is, the result of the operation can be stored into a different register than either of the two operands. This means that operands need not be overwritten by arithmetic operations. This results in a more efficient use of the large register set.
- **Conversion** instructions perform conversion operations on the floating point values in the FPA registers.
- **Compare** intructions perform comparisons of the contents of FPA registers and set a condition bit based on the results. The result of the compare operations is tied directly to Cp Cond (1) for software testing.
- **Jump and Branch** instructions change the control flow of a program. Jumps are always to a paged absolute address formed by combining a 26-bit target with four bits of the Program counter (J-type format, for subroutine calls), or 32-bit register byte addresses (R-type, for returns and

| OP | Description |
|---|---|
| | **Load/Store Instructions** |
| LB | Load Byte |
| LBU | Load Byte Unsigned |
| LH | Load Halfword |
| LHU | Load Halfword Unsigned |
| LW | Load Word |
| LWL | Load Word Left |
| LWR | Load Word Right |
| SB | Store Byte |
| SH | Store Halfword |
| SW | Store Word |
| SWL | Store Word Left |
| SWR | Store Word Right |
| | **FPA Load/Store/Move Instructions** |
| LWC1 | Load Word to FPA |
| SWC1 | Store Word from FPA |
| MTC1 | Move Word to FPA |
| MFC1 | Move Word from FPA |
| CTC1 | Move Control word to FPA |
| CFC1 | Move Control word from FPA |
| | **Arithmetlc Instructions (ALU Immediate)** |
| ADDI | Add Immediate |
| ADDIU | Add Immediate Unsigned |
| SLTI | Set on Less Than Immediate |
| SLTIU | Set on Less Than Immediate Unsigned |
| ANDI | AND Immediate |
| ORI | OR Immediate |
| XORI | Exclusive OR Immediate |
| LUI | Load Upper Immediate |
| | **Arithmetic Instructions (3-operand, register-type)** |
| ADD | Add |
| ADDU | Add Unsigned |
| SUB | Subtract |
| SUBU | Subtract Unsigned |
| SLT | Set on Less Than |
| SLTU | Set on Less Than Unsigned |
| AND | AND |
| OR | OR |
| XOR | Exclusive OR |
| NOR | NOR |
| | **FPA Computational Instructions** |
| ADD.fmt | Floating point Add |
| SUB.fmt | Floating point Subtract |
| MUL.fmt | Floating point Multiply |
| DlV.fmt | Floating point Divide |
| ABS.fmt | Floating-point Absolute value |
| MOV.fmt | Floating point Move |
| NEG.fmt | Floating point Negate |
| | **FPA Compare Instructions** |
| C.cond.fmt | Floating-point Compare |
| | **Shift Instructions** |
| SLL | Shift Left Logical |
| SRL | Shift Right Logical |

| OP | Description |
|---|---|
| | **Shift Instructions (Cont.)** |
| SRA | Shift Right Arithmetic |
| SLLV | Shift Left Logical Variable |
| SRLV | Shift Right Logical Variable |
| SRAV | Shift Right Arithmetic Variable |
| | **FPA Conversion Instructions** |
| CVT.S.fmt | Floating point Convert to Single FP |
| CVT.D.fmt | Floating point Convert to Double FP |
| CVT.W.fmt | Floating point Convert to fixed point |
| | **MultIply/Divide Instructions** |
| MULT | Multiply |
| MULTU | Multiply Unsigned |
| DIV | Divide |
| DIVU | Divide Unsigned |
| MFHI | Move From HI |
| MTHI | Move To HI |
| MFLO | Move From LO |
| MTLO | Move To LO |
| | **Jump and Branch Instructions** |
| J | Jump |
| JAL | Jump and Link |
| JR | Jump to Register |
| JALR | Jump and Link Register |
| BEQ | Branch on Equal |
| BNE | Branch on Not Equal |
| BLEZ | Branch on Less than or Equal to Zero |
| BGTZ | Branch on Greater Than Zero |
| BLTZ | Branch on Less Than Zero |
| BGEZ | Branch on Greater than or Equal to Zero |
| BLTZAL | Branch on Less Than Zero and Link |
| BGEZAL | Branch on Greater than or Equal to Zero and Link |
| | **Special Instructions** |
| SYSCALL | System Call |
| BREAK | Break |
| | **Coprocessor Instructions** |
| LWCZ | Load Word from Coprocessor |
| SWCZ | Store Word to Coprocessor |
| MTCZ | Move To Coprocessor |
| MFCZ | Move From Coprocessor |
| CTCZ | Move Control to Coprocessor |
| CFCZ | Move Control From Coprocessor |
| COPZ | Coprocessor Operation |
| BCZT | Branch on Coprocessor z True |
| BCZF | Branch on Coprocessor z False |
| | **System Control Coprocessor (CPO) Instructions** |
| MTC0 | Move To CP0 |
| MFC0 | Move From CP0 |
| TLBR | Read indexed TLB entry |
| TLBWI | Write Indexed TLB entry |
| TLBWR | Write Random TLB entry |
| TLBP | Probe TLB for matching entry |
| RFE | Restore From Exception |

**IDT79R3500 Instruction Summary**

2871 tbl 01

dispatches). Branches have 16-bit offsets relative to the program counter (I-type). Jump and Link instructions save a return address in Register 31. The R3500 instruction set features a number of branch conditions. Included is the ability to compare a register to zero and branch, and also the ability to branch based on a comparison between two registers. Thus, net performance is increased since software does not have to perform arithmetic instructions prior to the branch to set up the branch conditions.

- **Coprocessor** instructions perform operations in the coprocessors. Coprocessor Loads and Stores are I-type.
- **Coprocessor 0** instructions perform operations on the System Control Coprocessor (CP0) registers to manipulate the memory management and exception handling facilities of the processor.
- **Special** instructions perform a variety of tasks, including movement of data between special and general registers, system calls, and breakpoint. They are always R-type.

Table 1 lists the instruction set of the IDT79R3500 processor.

### IDT79R3500 System Control Coprocessor (CP0)

The IDT79R3500 can operate with up to four tightly-coupled coprocessors (designated CP0 through CP3). The System Control Coprocessor (or CP0), is incorporated on the IDT79R3500 chip and supports the virtual memory system and exception handling functions of the IDT79R3500. The virtual memory system is implemented using a Translation Lookaside Buffer and a group of programmable registers as shown in Figure 5.

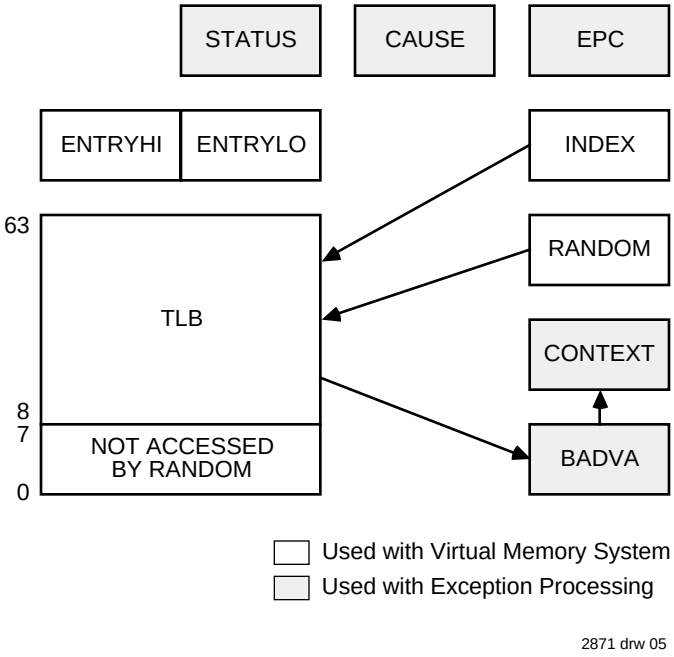
### System Control Coprocessor (CP0) Registers

The CP0 registers shown in Figure 5 are used to control the memory management and exception handling capabilities of the IDT79R3500. Table 2 provides a brief description of each register.

## SYSTEM CONTROL COPROCESSOR (CP0) INSTRUCTIONS

| Register | Description |
|---|---|
| EntryHi | High half of a TLB entry |
| EntryLo | Low half of a TLB entry |
| Index | Programmable pointer into TLB array |
| Random | Pseudo-random pointer into TLB array |
| Status | Mode, interrupt enables, and diagnostic status info |
| Cause | Indicates nature of last exception |
| EPC | Exception Program Counter |
| Context | Pointer into kernel's virtual Page Table Entry array |
| BadVA | Most recent bad virtual address |
| PRId | Processor revision identification (Read only) |

2871 tbl 02




2871 drw 05

**Figure 5. The System Coprocessor Registers**

### Memory Management System

The IDT79R3500 has an addressing range of 4GB. However, since most IDT79R3500 systems implement a physical memory smaller than 4GBs, the IDT79R3500 provides for the logical expansion of memory space by translating addresses composed in a large virtual address space into available physical memory address. Two TLB modes are supported. When the TLB is used, the 4GB address space is divided into 2GBs which can be accessed by both the users and the kernel, and 2GBs for the kernel only. Virtual addresses within the kernel/user segment are translated to physical addresses on a 4kB page basis. This mode is typical of UNIX and other sophisticated operating systems. When the TLB is disabled, mapping is locked as 2GBs as kernel/user, and 1.5GBs as kernel only. This mode requires no TLB manipulation, provides large linear address space, and is typical for embedded applications.

### TLB (Translation Lookaside Buffer)

Virtual memory mapping is assisted by the Translation Lookaside Buffer (TLB). The on-chip TLB provides very fast virtual memory access and is well-matched to the requirements of multi-tasking operating systems. The fully-associative TLB contains 64 entries, each of which maps a 4kB page, with controls for read/write access, cacheability, and process identification. The TLB allows each user to access up to 2GBs of virtual address space.

Figure 6 illustrates the format of each TLB entry. The Translation operation involves matching the current Process ID (PID) and upper 20 bits of the address against PID and VPN (Virtual Page Number) fields in the TLB. When both match (or the TLB entry is Global), the VPN is replaced with the PFN (Physical Frame Number) to form the physical address.

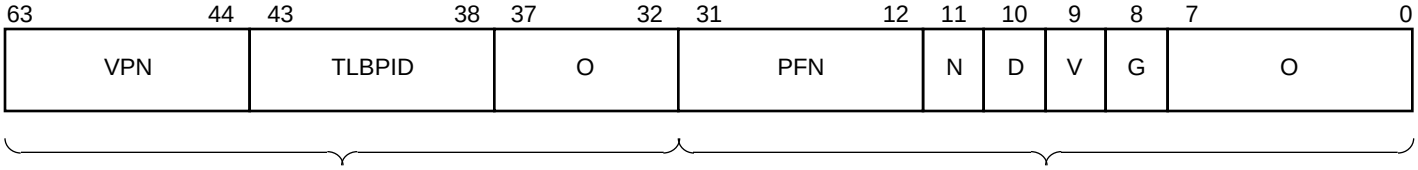
TLB misses are handled in software, with the entry to be replaced determined by as imple RANDOM function. The routine to process a TLB miss in the UNIX environment requires only 10-12 cycles, which compares favorably with many CPUs which perform the operation in hardware.

### TLB Disabled Operation

Many embedded systems do not like the complexity or uncertainty associated with the on-chip TLB. However, many systems still desire the ability to implement a kernel/user mode. Therefore, to implement a hierachical task model, the TLB must be used. The R3500 gives the system designer one more option, allowing the TLB to be disabled and performing a fixed mapping of virtual to physical addresses, while maintaining separation of kernel and user resources.

The user may elect to disable the TLB through the reset sectors. In this case, the mapping shown in Figure 8. is used, and device power consumption is reduced. Note tha “cached“ segments means that there is no mechanism to exclude addresses in these regions from the cache.

This mapping means that applications designed to run in kseg0 and kseg1 (to avoid the TLB) can use the R3500, disable the TLB to reduce power, and not have to change software to take advantage of this new feature.




VPN – Virtual Page Number
TLBPID – Process ID
PFN – Physical Frame Number
N – Non-cacheable flag
D – Dirty flag (Write protect)
V – Valid entry flag
G – Global flag (ignore PID)
O – Reserved

2871 drw 06

**Figure 6. TLB Entry Format**

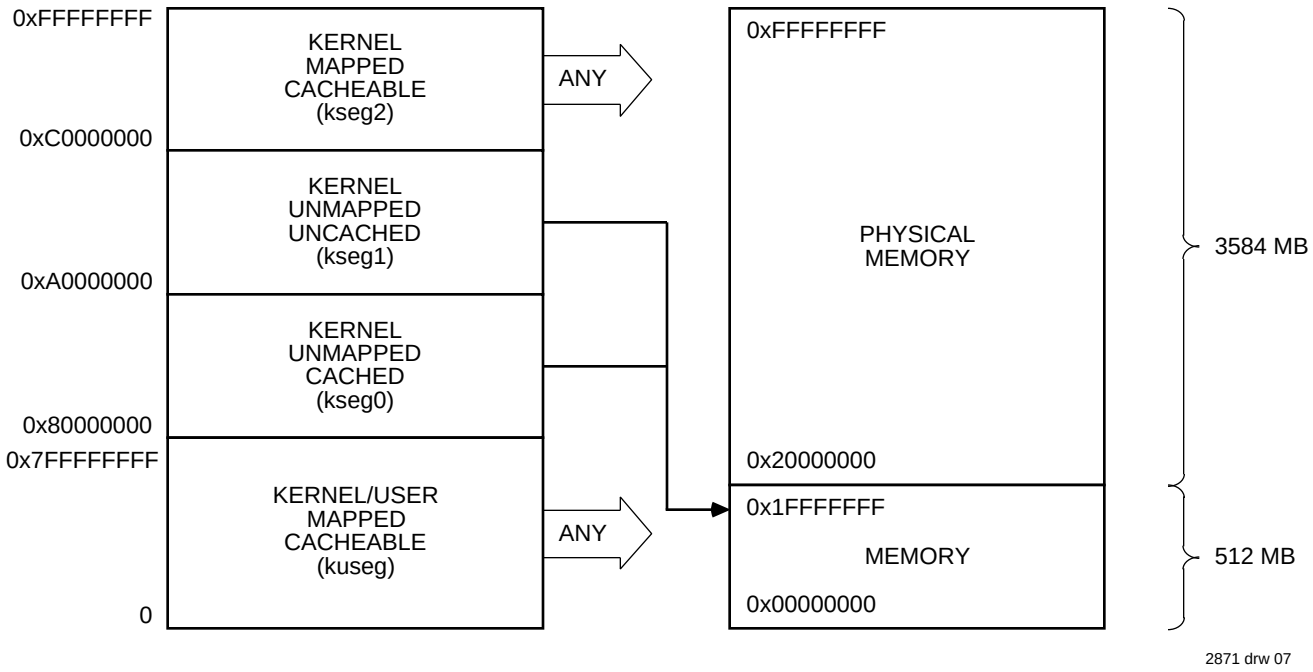
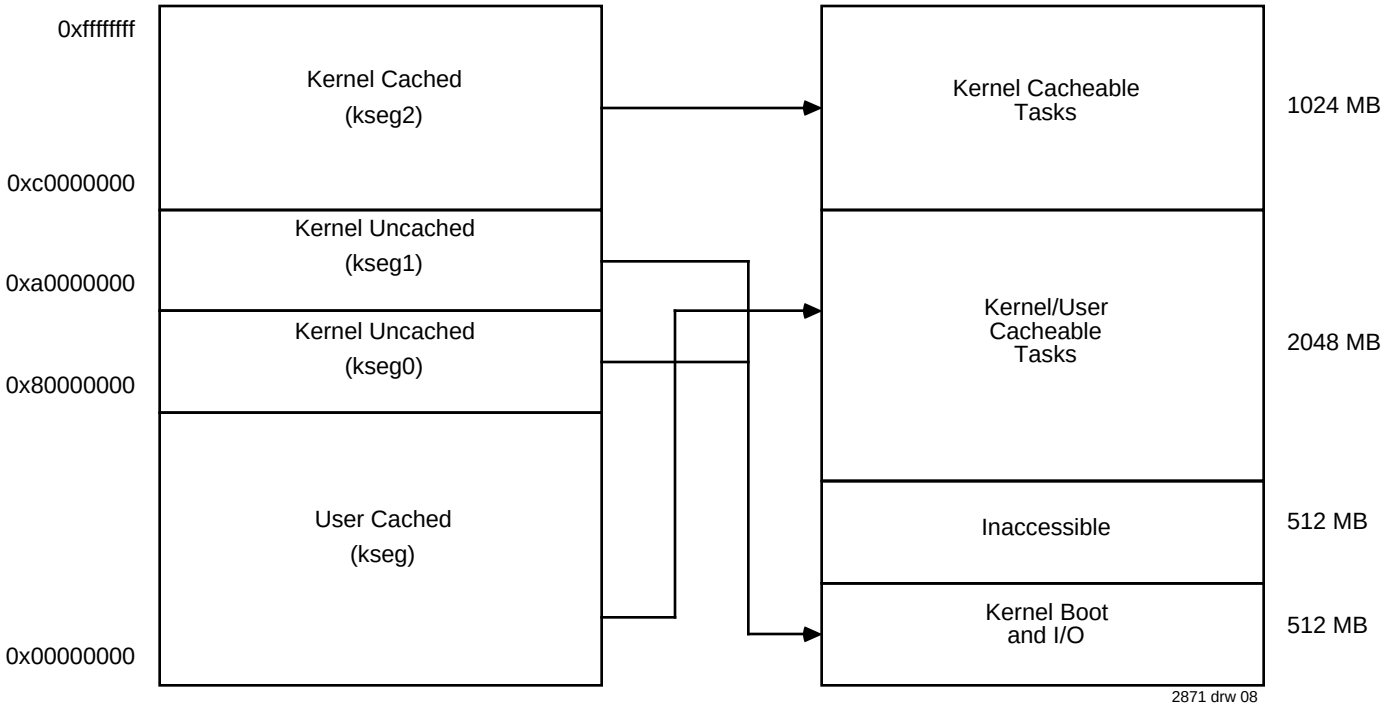



**Figure 7. IDT79R3500 Virtual Address Mapping**

MNU Address Translation
Virtual –> Physical
(TBL Disabled)




**NOTE:** This model is consistent with the mapping available in the IDT79R3051 family. The identical mapping provides software compatibility to the lower cost CPUs.

**Figure 8. TLB Disabled Mapping**

### Operating Modes

The IDT79R3500 has two operating modes: User mode and Kernel mode. The IDT79R3500 normally operates in the User mode until an exception is detected forcing it into the Kernel mode. It remains in the Kernel mode until a Restore From Exception (RFE) instruction is executed. The manner in which memory addresses are translated or mapped depends on the operating mode of the IDT79R3500. Figure 7 shows the MMU translation performed for each of the operating modes.

**User Mode**—in this mode, a single, uniform virtual address space (kuseg) of 2GB is available. When the TLB is used, each virtual address is extended with a 6-bit process identifier field to form unique virtual addresses. All references to this segment are mapped through the TLB. Use of the cache for up to 64 processes is determined by bit settings for each page within the TLB entries. If the TLB is not used, these addresses are translated to begin at 1GB of the physical address space.

**Kernel Mode**—four separate segments are defined in this mode:

- *kuseg*—when in the kernel mode, references to this segment are treated just like user mode references, thus streamlining kernel access to user data.
- *kseg0*—references to this 512MB segment use cache memory but are not mapped through the TLB. Instead, they always map to the first 0.5GB of physical address space.
- *kseg1*—references to this 512MB segment are not mapped through the TLB and do not use the cache. Instead, they are hard-mapped into the same 0.5GB segment of physical address space as kseg0.
- *kseg2*—when the TLB is not used, references to this 1GB segment directly addresses the upper 1GB of physical address space. These addresses are defined to be kernel mode which are cacheable. When the TLB is used, references to this 1GB segment are always mapped through the TLB and use of the cache is determined by bit settings within the TLB entry.

## FPA COPROCESSOR OPERATION (CP1)

The FPA continually monitors the processor instruction stream. If an instruction does not apply to the coprocessor, it is ignored; if an instruction does apply to the coprocessor, the FPA executes that instruction and transfers necessary result and exception data synchronously to the main processor.

The FPA performs three types of operations:

- Loads and Stores;
- Moves;
- Two- and three-register floating-point operations.

### Load, Store, and Move Operation

Load, Store, and Move operations data between memory or the integer registers and the FPA registers. These operations perform no format conversions and cause no floating-point exceptions. Load, Store, and Move operations reference a single 32-bit word of either the Floating-Point General Registers (FGR) or the Floating-Point Control Registers (FCR).

### Floating-Point Operations

The FPA supports the following single- and double-precision format floating-point operations:

- Add
- Subtract
- Multiply
- Divide
- Absolute Value
- Move
- Negate
- Compare

In addition, the FPA supports conversions between single- and double-precision floating-point formats and fixed-point formats.

The FPA incorporates separate Add/Subtract, Multiply, and Divide units, each capable of independent and concurrent operation. Thus, to achieve very high performance, floating point divides can be overlapped with floating point multiplies and floating point additions. These floating point operations occur independently of the actions of the CPU, allowing further overlap of integer and floating point operations. Figure 9 illustrates an example of the types of overlap permissible.

### Exceptions

The FPA supports all five IEEE standard exceptions:

- Invalid Operation
- Inexact Operation
- Division by Zero
- Overflow
- Underflow

The FPA also suppoerts the optional, Unimplemented Operation exception that allows unimplemented instructions to trap to software emulation routines.

The FPA provides precise exception capability to the CPU; that is, the execution of a floating point operation which generates an exception causes that exception to occur at the CPU instruction which caused the operation. This precise exception capability is a requirement in applications and languages which provide a mechanism for local software exception handlers within software modules.

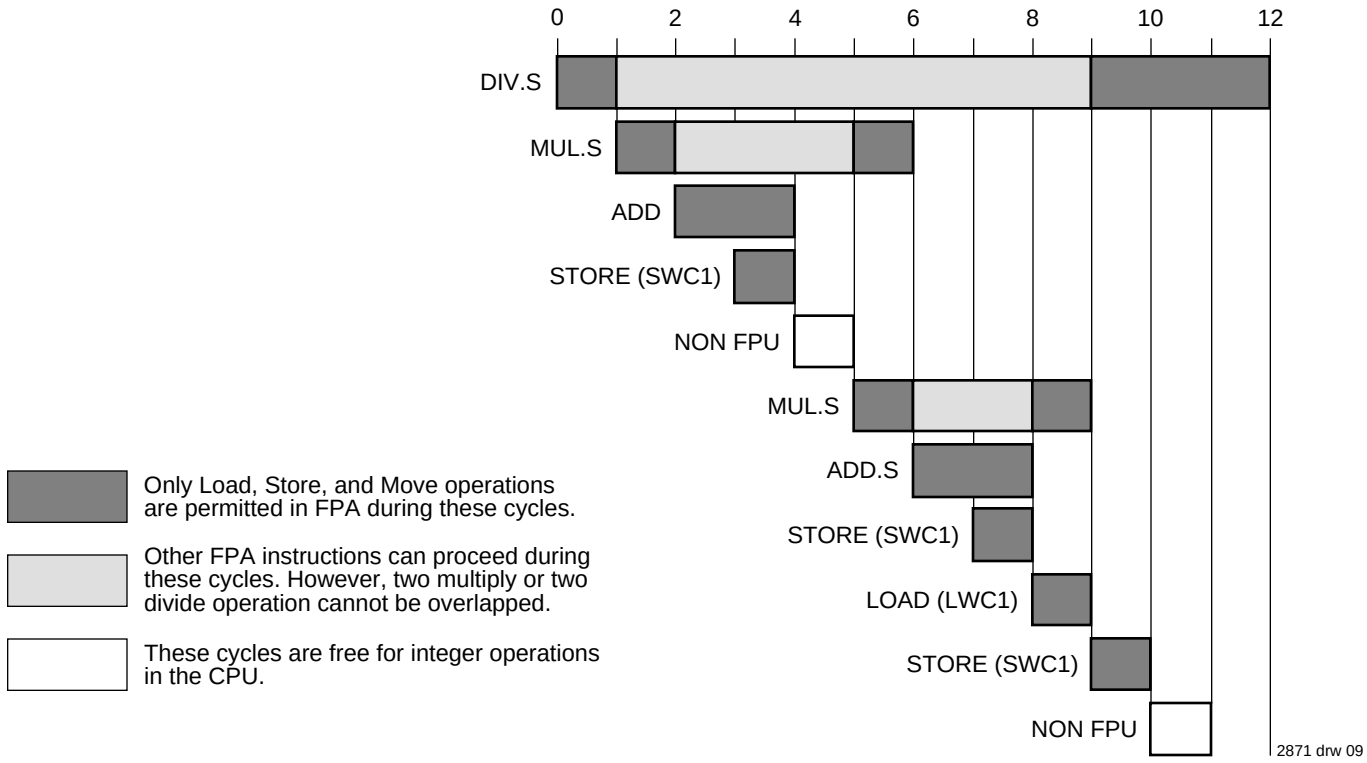



**Figure 9. Examples of Overlapping Floating Point Operation**

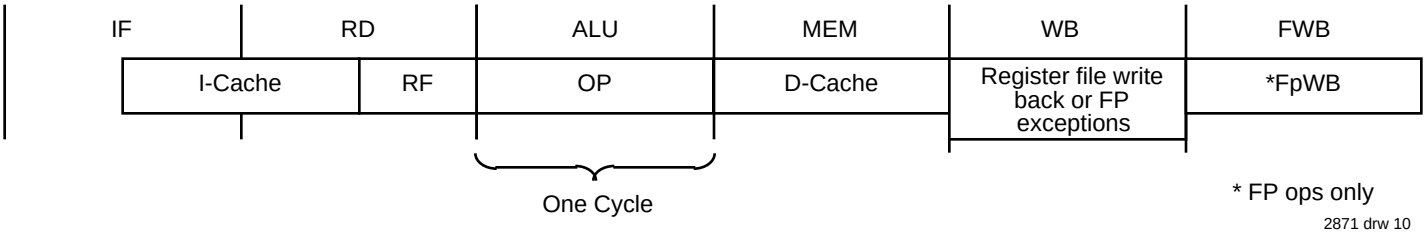



**Figure 10. Instruction Execution**

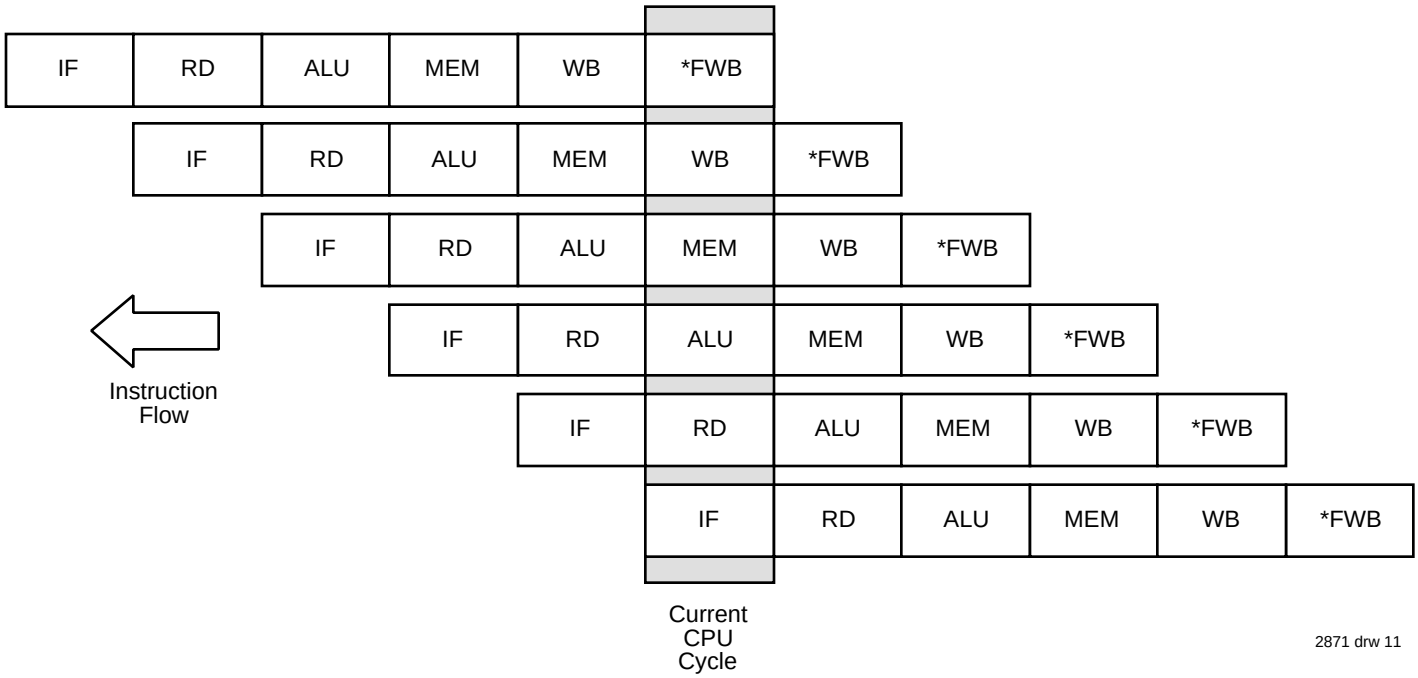



**Figure 11. IDT79R3500 Execution Sequence**

## IDT79R3500 PIPELINE ARCHITECTURE

The execution of a single IDT79R3500 integer instruction consists of five pipe stages while floating point instruction takes six pipe stages. They are:

1) IF—Instruction fetch. The processor calculates the instruction address required to read from the I cache.
2) RD—The instruction is present on the data bus during phase one of this pipe stage. Instruction decode occurs during phase two. Operands are read from the registers if required.
3) ALU—Perform the required operation on instruction operands. If this is a FPA instruction, instruction execution commences.
4) MEM—Access memory. If the instruction is a load or store, the data is presented or captured during phase 2 of this pipe stage.
5) WB—Write integer results back into register file. In FPA cycles this pipe stage is used for exceptions.
6) FWB—The FPA uses this stage to write back ALU results to its register file.

Each of these steps requires approximately one FPA cycle as shown in Figure 10. (parts of some operations spill over into another cycle while other operations require only 1/2 cycle.)

The CPU uses a five stage pipeline while while the FPA uses a 6 stage to achieve an instruction execution rate approaching one instruction per cycle. Thus, execution of six instructions at a time are overlapped as shown in Figure 11.

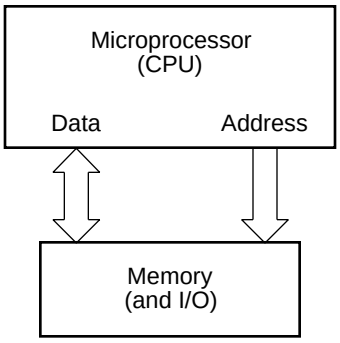
This pipeline operates efficiently because different CPU resources (address and data bus accesses, ALU operations, register accesses, and so on) are utilized on a non-interfering basis.

## MEMORY SYSTEM HIERARCHY

The high performance capabilities of the IDT79R3500 processor demand system configurations incorporating techniques frequently employed in large, mainframe computers but seldom encountered in systems based on more traditional microprocessors.

A primary goal of systems employing RISC techniques is to minimize the average number of cycles each instruction requires for execution. Techniques to reduce cycles-per-instruction include a compact and uniform instruction set, a deep instruction pipeline (as described above), and utilization of optimizing compilers. Many of the advantages obtained from these techniques can, however, be negated by an inefficient memory system.

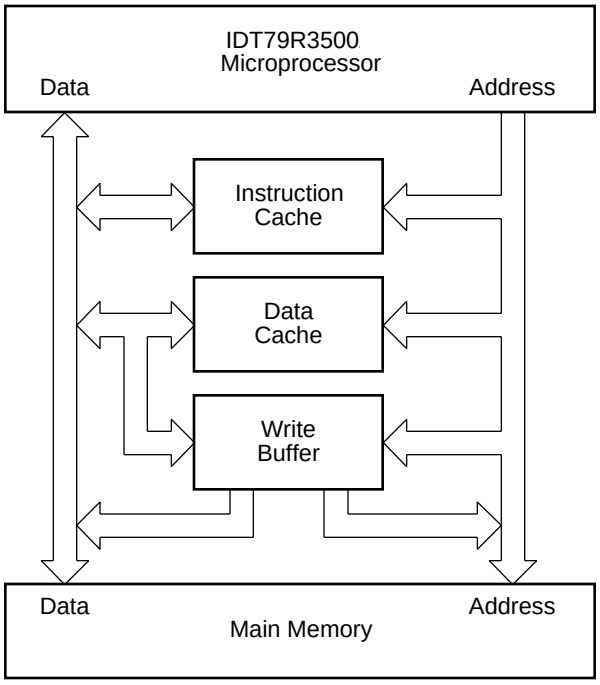
Figure 12 illustrates memory in a simple microprocessor system. In this system, the CPU outputs addresses to memory and reads instructions and data from memory or writes data to memory. The address space is completely undifferentiated: instructions, data, and I/O devices are all treated the same. In such a system, a primary limiting performance factor is memory bandwidth.




2871 drw 12

**Figure 12. A Simple Microprocessor Memory System**

Figure 13 illustrates a memory system that supports the significantly greater memory bandwidth required to take full advantage of the IDT79R3500's performance capabilities. The key features of this system are:




2871 drw 13

**Figure 13. An IDT79R3500 System with a High-Performance Memory System**

- **External Cache Memory**—Local, high-speed memory (called cache memory) is used to hold instructions and data that is repetitively accessed by the CPU (for example, within a program loop) and thus reduces the number of references that must be made to the slower-speed main memory. Some microprocessors provide a limited amount of cache memory on the CPU chip itself. The external caches supported by the IDT79R3500 can be much larger; while a small cache can improve performance of some programs, significant improvements for a wide range of programs require large caches.
- **Separate Caches for data and Instructions**—Even with high-speed caches, memory speed can still be a limiting factor because of the fast cycle time of a high-performance microprocessor. The IDT79R3500 supports separate caches for instructions and data and alternates accesses of the two caches during each CPU cycle. Thus, the processor can obtain data and instructions at the cycle rate of the CPU using caches constructed with commercially available IDT static RAM devices.

  In order to maximize bandwidth in the cache while minimizing the requirement for SRAM access speed, the IDT79RR3500 divides a single-processor clock cycle into two phases. During one phase, the address for the data cache access is presented while data previously addressed in the instruction cache is read; during the next phase, the data operation is completed while the instruction cache is being addressed. Thus, both caches are read in a single processor cycle using only one set of address and data pins.
- **Write Buffer**—in order to ensure data consistency, all data that is written to the data cache must also be written out to main memory. The cache write model used by the IDT79R3500 is that of a write-through cache; that is, all data written by the CPU is immediately written into the main memory. To relieve the CPU of this responsibility (and the inherent performance burden) the IDT79R3500 supports an interface to a write buffer. The IDT79R3020 Write Buffer captures data (and associated addresses) output by the CPU and ensures that the data is passed on to main memory.

**IDT79R3500 Processor Subsystem Interfaces**

Figure 14 illustrates the three subsystem interfaces provided by the IDT79R3500 processor:

- **Cache control** interface (on-chip) for separate data and instruction caches permits implementation of off-chip caches using standard IDT SRAM devices. The IDT79R3500 directly controls the cache memory with a minimum of external components. Both the instruction and data cache can vary from 0 to 256kB (64K entries). The IDT79R3500 also includes the TAG control logic which determines whether or not the entry read from the cache is the desired data. The IDT79RR3500 implements an advanced feature that allows certain tag comparisons to be eliminated, which in turn reduces the number of cache SRAMs required. The Int(5) reset mode vector contains two bits which sets the tag comparison options. Table 3 illustrates the tag disable encoding. The first row in the table implements the standard IDT79R3000A operating mode where all the tag and tag parity are used. The second row eliminates the upper 4 tag bits, eliminating normally required SRAMs and limiting main memory addressing to 128mB. The third row elimnates the lower 4 tag bits, which requires the cache to be at least 64kB each. The fourth row eliminates the upper 4 and lower 4 tag bits, requiring at least 16K cache entries, and limits main memory addressing to 128mB. In all cases, the IDT79R3500 continues to check tag parity which are selected as driven from the cache. The IDT79R3500 cache controller implements a direct mapped cache for high net performance (bandwidth). It has the ability to refill multiple words when a cache miss occurs, thus reducing the effective miss rate to less than 2% for large caches. When a cache miss occurs, the IDT79R3500 can support refilling the cache in 1, 4, 8, 16, or 32 word blocks to minimize the effective penalty of having to access main memory. The IDT79R3500 also incorporates the ability to perform instruction streaming; while the cache is refilling, the processor can resume execution once the missed word is obtained from main memory. In this way, the processor can continue to execute concurrently with the cache block refill.
- **Memory controller** interface for system (main) memory. This interface also includes the logic and signals to allow operation with a write buffer to further improve memory bandwidth. In addition to the standard full word access, the memory controller supports the ability to write bytes and half-words by using partial word operations. The memory controller also supports the ability to retry memory accesses if, for example, the data returned from memory is invalid and a bus error needs to be signalled.
- **Coprocessor Interface**—The IDT79R3500 features a set of on board tightly coupled coprocessors. Coprocessor 0 is defined to be the system control coprocessor and Coprocessor 1 is the Floating Point Accelerator. They have direct access to the internal data bus which allows them direct load and store of data in the same fashion as accessing the CPU registers. This relieves the typical bottleneck of having to load data into the CPU register set and then passing that data off to the co-processors.

  In applications where the FPA was off chip, as in using the IDT79R3010A, several control pins were used for communications with the CPU and a Phase Lock Loop was located on the IDT79R3010A to synchronize the two together. As they are now integrated into a single chip, these are no longer needed. The FpCond output, which is used in coprocessor branch instructions, is now internally tied to the CpCond(1) input of the CPU leaving the external CpCond(1) pin available for another function. This signal is selectable to either output the FpBusy or the Fplnt. Cp

Cond(1) output selection is determined at reset time according to the value read on Int(4). Table 4 illustrates the options that allow the FpInt to be routed to either the CpCond(1) output, or one of the internal Int pins. If it is internally routed, that interrupt is dedicated and that input will no longer affect the IDT79R3500. The selection of using CpCond(1) allows some external Logic to be added to the path, which might be required in some applications. Another method for Fpint handling is also accommodated. A mode pin, previously $V_{CC}$ can be programmed to route the FPU interrupt to a dedicated Fpint output that was previously a GND. If the mode pin is sampled at reset as a 0, the dedicated Fpint indicates the FPU interrupt - if a 1, then the routing of Table 4 applies.

The internal CPBusy input, which is used to stall the CPU if the coprocessor needs to hold off subsequent operations, has two sources-FPBusy and the external CpBusy pin which are logically ORed together. Further, Run and Exception of both the FPA and CPU are internally tied and brought out with the external CPBusy input to accommodate off chip coprocessor 2 and 3. This external interface is available to support application specific functions.

| Tag Mode 1 | Tag Mode 0 | Check Which TAGs | Ignore Which Tags |
|---|---|---|---|
| 0 | 0 | Tag (31:12) | None |
| 0 | 1 | Tag (27:12) | Tag (31:28) |
| 1 | 0 | Tag (31:16) | Tag (15:12) |
| 1 | 1 | Tag (27:16) | Tag (31:28;15:12) |

2871 tbl 03

**Table 3. Tag Disable Encoding**

| W Cycle | X Cycle | Y Cycle | Z Cycle | Action |
|---|---|---|---|---|
| X | X | X | "HIGH" | FPint driven onto CpCond(1) |
| "LOW" | "LOW" | "LOW" | "LOW" | Use Int(3) for Fpint |
| "LOW" | "LOW" | "HIGH" | "LOW" | Use Int(1) for Fpint |
| "LOW" | "HIGH" | "LOW" | "LOW" | Use Int(2) for Fpint |
| "LOW" | "HIGH" | "HIGH" | "LOW" | Use Int(0) for Fpint |
| 'HIGH" | "LOW" | "LOW" | "LOW" | Use Int(4) for Fpint |
| 'HIGH" | "LOW" | "HIGH" | "LOW" | Use Int(5) for Fpint |
| 'HIGH" | "HIGH" | "LOW" | "LOW" | Reserved, Undefined |
| 'HIGH" | "HIGH" | "HIGH" | "LOW" | Reserved, Undefined |

2871 tbl 04

**Table 4. Int(4) Encoding for Fpint**

## MULTIPROCESSING SUPPORT

The IDT79R3500 supports multiprocessing applications in a simple but effective way. Multiprocessing applications require cache coherency across the multiple processors. The IDT79R3500 offers two signals to support cache coherency: the first, MPStall, stalls the processor within two cycles of being received and keeps it from accessing the cache. This allows an external agent to snoop into the processor data cache. The second signal, MPInvalidate, causes the processor to write data on the data cache bus which indicates the externally addressed cache entry is invalid. Thus, a subsequent access to that location would result in a cache miss, and the data would be obtained from main memory.

The two MP signals would be generated by a external logic which utilizes a secondary cache to perform bus snooping functions. The IDT79R3500 does not impose an architecture for this secondary cache, but rather is flexible enough to support a variety of application specific architectures and still maintain cache coherency. Further, there is no impact on designs which do not require this feature. The IDT79R3500 further allows the use of cache RAMs with internal address latches in multiprocessor systems.

## ADVANCED FEATURES

The IDT79R3500 offers a number of additional features such as the ability to swap the instruction and data caches, facilitating diagnostics and cache flushing. Another feature isolates the, caches, which forces cache hits to occur regardless of the contents of the tag fields. The IDT79R3500 allows the processor to execute user tasks of the opposite byte ordering (endianness) of the operating system, has a programmable Tag width bus, and further allows certain parity checking to be disabled. More details on these features can be found in the *IDT79R3000A Family Hardware User's Manual*.

Further features of the IDT79R3500 are configured during the last four cycles prior to the negation of the RESET input. These functions include the ability to select cache sizes and cache refill block sizes; the ability to utilize the multiprocessor interface; whether or not instruction streaming is enabled; whether byte ordering follows "Big-Endian" or "Little-Endian" protocols, etc. Additionally, the IDT79R3500 mode must be

true to enable any of the new features that the X,Y, and Z cycles define. Table 6 shows the configuration options selected at Reset. These are further discussed in the *IDT79R3000A Family Hardware User's Manual*.

## BACKWARD COMPATIBILITY

The primary goal of the IDT79R3500 is the ability to replace the IDT79R3000A and IDT79R3010A with a single chip solution. The pinout of the IDT79R3500 has been selected to ensure this compatibility, with new functions mapped onto previously used pins. The instruction set is compatible with that of the R2000 at the binary level. As a result, code written for the older processor can be executed.

In most IDT79R3000A applications, the IDT79R3500 can be placed in the socket with no modification to initialization settings. Additionally, the IDT79R3500 can be used in systems that did not include the IDT79R3010 in the original design. Further application assistance on these topics are available from IDT.

## PACKAGE THERMAL SPECIFICATIONS

The IDT79R3500 utilizes special packaging techniques to improve both the thermal and electrical characteristics of the microprocessor.

In order to improve the electrical characteristics of the device, the package is constructed using multiple signal planes, including individual power planes and ground planes to reduce noise associated with high-frequency TTL parts. In addition, the 161-pin PGA package utilizes extra power and ground pins to reduce the inductance from the internal power planes to the power planes of the PC Board.

In order to improve the thermal characteristics of the microprocessor, the device is housed using cavity down packaging. In addition, these packages incorporate a copper-tungsten thermal slug designed to efficiently transfer heat from the die to the case of the package, and thus effectively lower the thermal resistance of the package. The use of an additional external heat sink affixed to the package thermal slug further decreases the effective thermal resistance of the package.

The case temperature may be measured in any environment to determine whether the device is within the specified operating range. The case temperature should be measured at the center of the top surface opposite the package cavity (the package cavity is the side where the package lid is mounted).

The equivalent allowable ambient temperature, $T_A$, can be calculated using the thermal resistance from case to ambient ($\varnothing ca$) for the given package. The following equation relates ambient and case temperature:

$$T_A = Tc - P*\varnothing ca$$

where P is the maximum power consumption, calculated by using the maximum Icc from the DC Electrical Characteristics section.

Typical values for $\varnothing ca$ at various airflows are shown in Table 5 for the various CPU packages.

| | Airflow - (ft/min) | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 200 | 400 | 600 | 800 | 1000 |
| $\varnothing ca$ (161-PGA) | 21 | 7 | 3 | 2 | 1 | 0.5 |
| $\varnothing ca$ (160 MQUAD) | 17 | 11 | 9 | 8 | 7 | 6.5 |

2871 tbl 05

**Table 5. R3500 Package Characteristics**

| Input | W Cycle | X Cycle | Y Cycle | Z Cycle |
|---|---|---|---|---|
| Int0 | DBIkSize0 | DBIkSize1 | Extend Cache | Big Endian |
| Int1 | IBIkSize0 | IBIkSize1 | MPAdrDisable | TriState |
| Int2 | DispPar/RevEnd | IStream | IgnoreParity | NoCache |
| Int3 | Reserved(1) | StorePartial | MultiProcessor | BusDriveOn |
| Int4 | FPINT decode | FPINT decode | FPINT decode | FPINT onto CpCond |
| Int5 | 7R3500 mode | TLB disable | Tag Mode 1 | Tag Mode 0 |

2871 tbl 06

**NOTES:**
1. Reserved entries must be driven high.
2. These values must be driven stable throughout the enfire RESET period.

**Table 6. R3500 Mode Selectable Features**

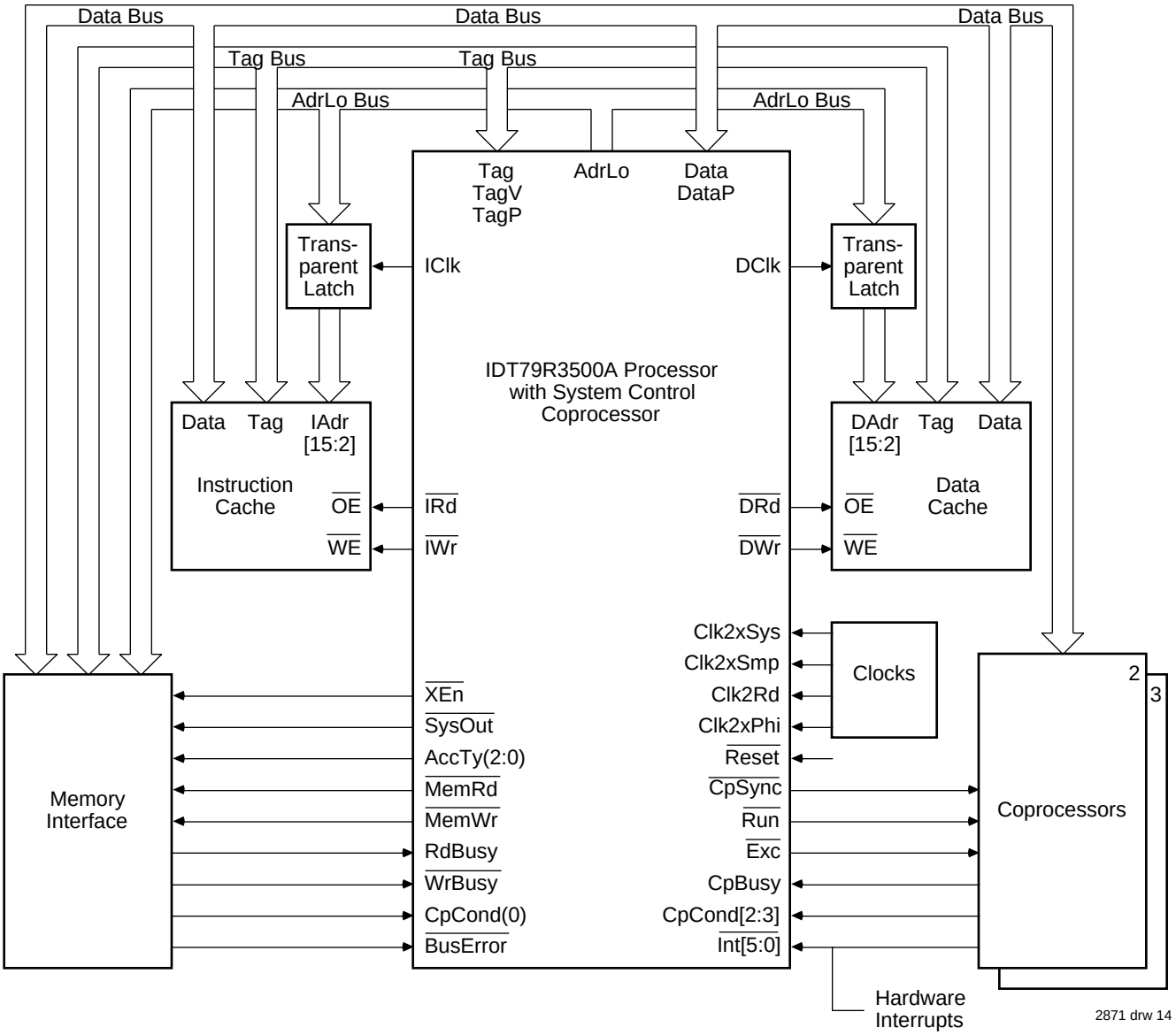



**Figure 14. IDT79R3500 Subsystem Interfaces Example; 64 KB Caches**

## PIN CONFIGURATION

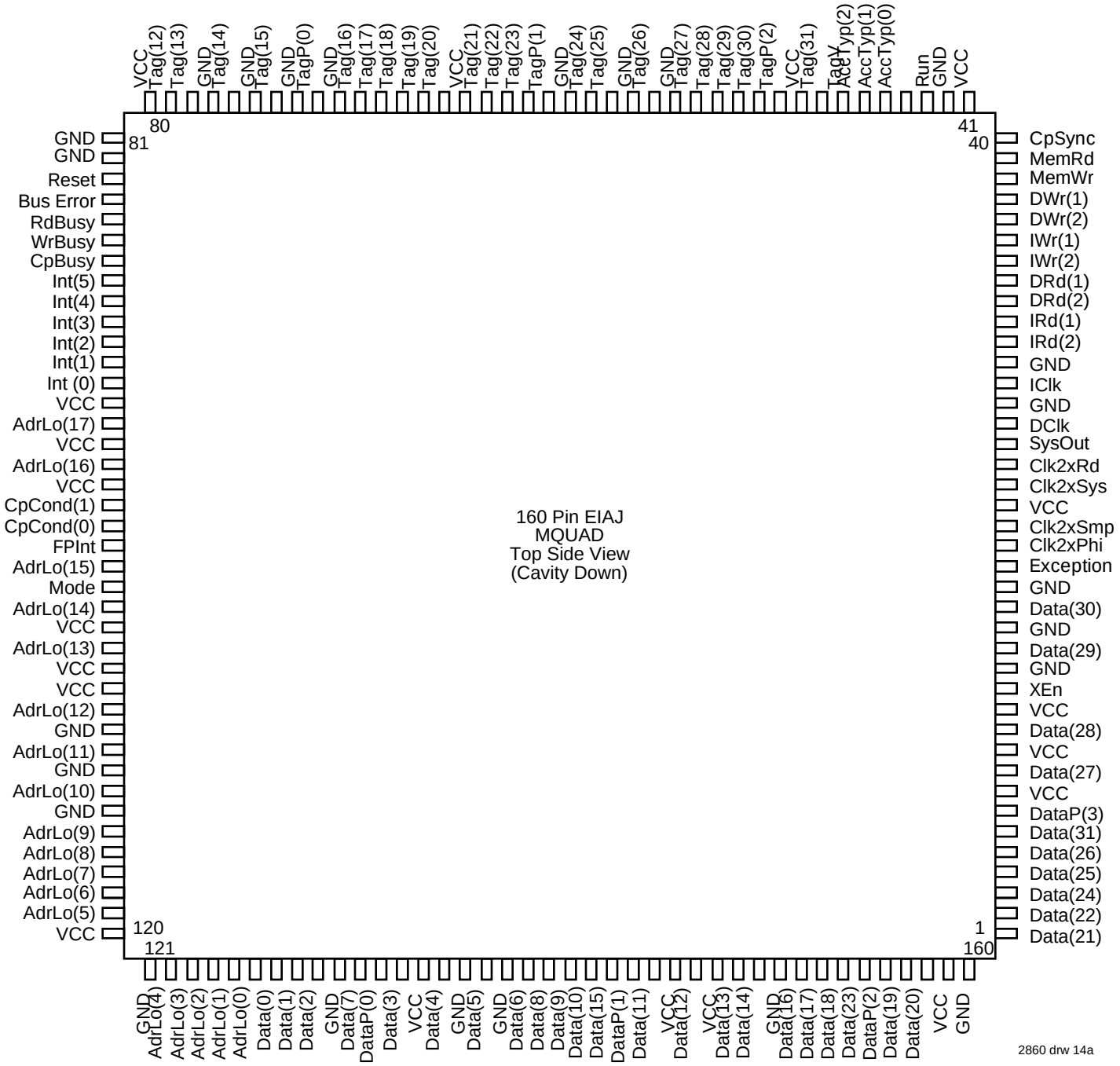
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | (No Pin) | AdrLo 6 | AdrLo 10 | AdrLo 11 | VCC | AdrLo 14 | AdrLo 15 | CpCond 0 | AdrLo 16 | AdrLo 17 | Int(2) | Int(5) | Wr Busy | Reset | VCC |
| B | AdrLo 3 | DRd2 | AdrLo 7 | AdrLo 9 | AdrLo 12 | IRd2 | AdrLo 13 | CpCond 1 | Int(1) | Int(3) | Cp Busy | Bus Error | DWr2 | Tag12 | Tag15 |
| C | AdrLo 0 | AdrLo 4 | Mode | AdrLo 5 | AdrLo 8 | GND | GND | VCC | Int(0) | Int(4) | Rd Busy | GND | Tag13 | TagP0 | Tag18 |
| D | Data 1 | AdrLo 2 | FpInt | GND | VCC | GND | VCC | GND | VCC | GND | VCC | GND | Tag14 | Tag17 | Tag19 |
| E | DataP 0 | Data 0 | AdrLo 1 | | | | | | | | | | Tag16 | Tag20 | VCC |
| F | VCC | Data 7 | Data 2 | | | | | | | | | | GND | Tag21 | Tag23 |
| G | Data 4 | Data 3 | GND | | | | | | | | | | GND | Tag22 | TagP1 |
| H | Data 6 | Data 5 | Data 8 | | | | | | | | | | VCC | Tag25 | Tag24 |
| J | Data 10 | DataP 1 | Data 9 | | | | | | | | | | Tag28 | Tag29 | Tag26 |
| K | Data 15 | Data 11 | GND | | | | | | | | | | GND | TagP2 | Tag27 |
| L | VCC | Data 12 | Data 17 | | | | | | | | | | Acc Typ2 | Tag31 | Tag30 |
| M | Data 13 | Data 16 | DataP 2 | GND | VCC | GND | VCC | GND | VCC | GND | VCC | GND | GND | Acc Typ1 | VCC |
| N | Data 14 | Data 18 | Data 19 | GND | Data 24 | DataP 3 | VCC | VCC | GND | GND | DRd1 | Mem Wr | Mem Rd | Run | TagV |
| P | Data 23 | Data 20 | IWr2 | Data 22 | Data 26 | Data 27 | XEn | Data 30 | Clk2x Sys | Clk2x Rd | DClk | IRd1 | IWr1 | Cp Sync | Acc Typ0 |
| Q | VCC | Data 21 | Data 25 | Data 31 | Data 28 | GND | Data 29 | Excep-tion | Clk2x Phi | Clk2x Smp | SysOut | VCC | IClk | DWr1 | VCC |

2871 drw 16

**NOTE:**

1. AdrLo 16 and 17 are multifunction pins which are controlled by mode select programming on interrupt pins at reset time
   AdrLo 16: MP Invalidate, CpCond (2).
   AdrLo 17: MP Stall, CpCond (3).
2. This package is pin-compatible with the 175-pin PGA for the R3000A.

## PIN CONFIGURATION




2860 drw 14a

**NOTE:**

1. AdrLo 16 and 17 are multifunction pins which are controlled by mode select programming on interrupt pins at reset time
   AdrLo 16: MP Invalidate, CpCond (2).
   AdrLo 17: MP Stall, CpCond (3).
2. This package is pin-compatible with the 175-pin PGA for the R3000A.