FUJITSU

TurboSPARC Microprocessor User's Guide

October 1996

Revision 1.0

TurboSPARC™

Table of Contents

List of Tables

Chapter 1

The TurboSPARC Microprocessor

The TurboSPARC microprocessor is a high frequency, highly integrated single-chip CPU. Implementing the SPARC architecture v8 specification, the TurboSPARC is ideally suited for low-cost uniprocessor applications.

The TurboSPARC microprocessor provides balanced integer and floating point performance in a single VLSI component, implementing a Harvard-style architecture with separate instruction and data busses. Large 16 KByte instruction and data caches help to minimize cache misses, while the on-chip cache controller is capable of supporting up to 1 MByte of secondary cache.

The TurboSPARC microprocessor contains an Integer Unit (IU), Floating Point Controller (FPC), Floating Point Unit (FPU), Instruction and data caches, Memory Management Unit (MMU), and Secondary cache, DRAM, and SBus controllers. Figure 1-1. shows a block diagram of the TurboSPARC microprocessor. Each of the blocks is then briefly discussed.

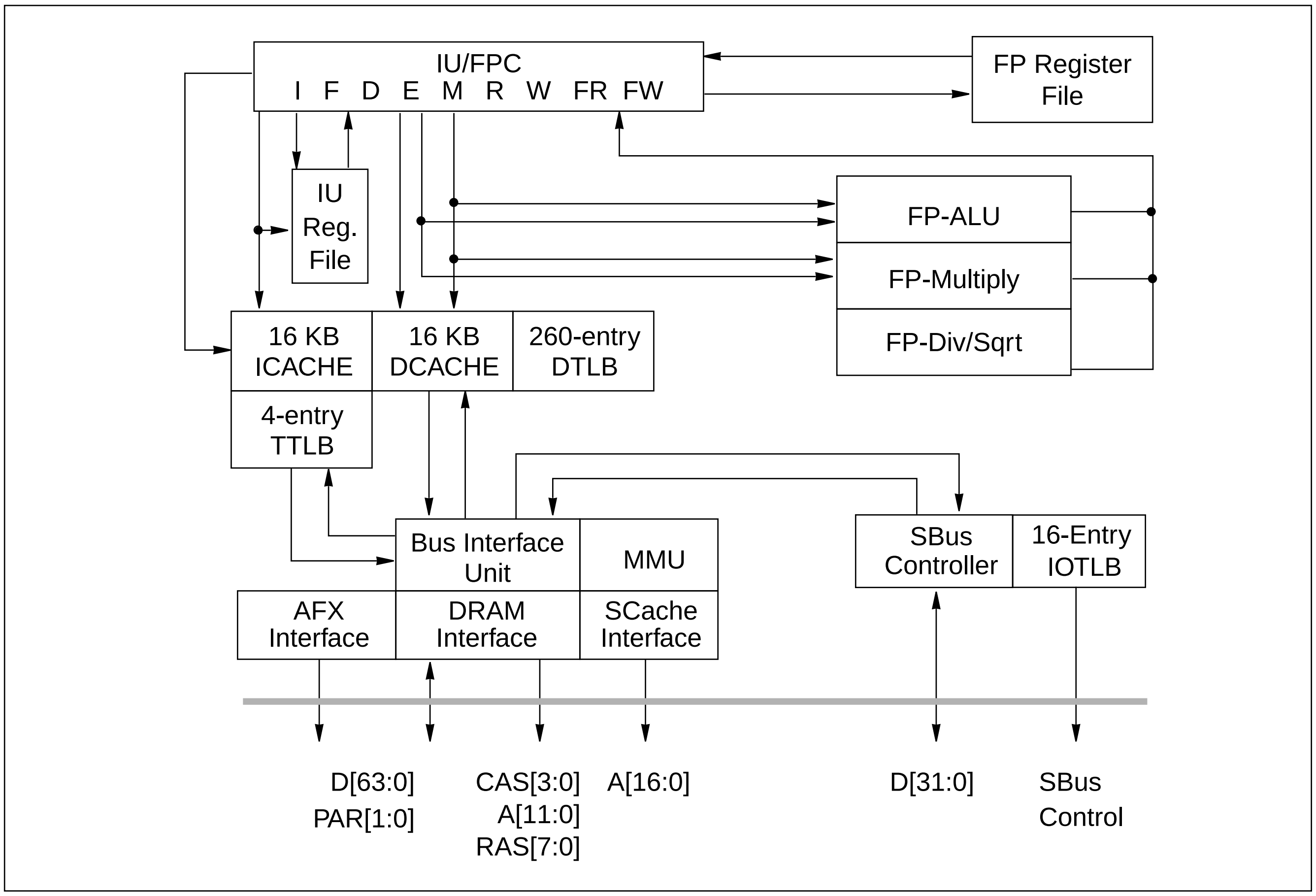


Figure 1-1. TurboSPARC Microprocessor Block Diagram

1.1 INTEGER UNIT AND FLOATING POINT CONTROLLER

As shown in Figure 1-1., the integer unit (IU) and floating point control (FPC) are merged into a 9-stage pipeline. Below are some of the features of the IUFPC.

- 9 stage instruction pipeline.
- No branch folding. Branch instructions (taken/non-taken) execute in one cycle.
- Harvard architecture.
- IMUL and IDIV implemented using Floating Point Multiplier and Divider.
- Three read and one write port 8-Window register file.
- Single cycle store operation.
- Fully integrated pipelined FPC within IU pipeline.

1.2 FLOATING POINT UNIT

The FMUL and FDIV units handle both floating point and integer multiply and divide operations. The FALU unit handles floating point add and subtract operations, while integer ALU operations are performed within the IU. A fourth block in the floating point unit is FPUIO, which serves as the interface between the three blocks floating point arithmetic blocks and the IUFPC.

The floating point is 100% IEEE 754 Standard compliant and follows all recommendations in the SPARC Architecture Manual v.8, Appendix N. The TurboSPARC microprocessor is always in "standard mode" as defined in the floating point status register. The floating point unit implements all operations on all single and double precision operand types. Quad operations cause an illegal instruction trap in the IUFPC; the floating point unit will never generate an fp_execption of an unfinished type.

1.3 INSTRUCTION CACHE

The instruction cache is 16 KBytes in size and direct mapped. The cache is virtually indexed and virtually tagged. The instruction cache line size is 32 bytes and implements a *'streaming'* cache architecture. The first doubleword of an instruction cache line fill is the one required to resolve the cache miss. However, if while the other three remaining doublewords are being written to the instruction cache, an instruction cache hit occurs to any line in the cache, including these three doublewords, that doubleword can be fetched while the line fill is in progress. The result is that the cache is not blocked until the line fill is completed.

The instruction cache also includes a fully-associative 4-entry instruction TLB which contains only page table entries (PTE) and is accessed only on an instruction cache miss.

In addition to the ITLB, an 8-bit context identifier is attached to each instruction cache entry. This identifier helps to differentiate between multiple processes within the same cache and helps to reduce cache flushing by allowing the operating system to invalidate only those lines whose process is no longer valid. The instruction cache can also be either selectively flushed, or globally flushed by clearing all of the valid bits in one operation. Refer to Section 3.2 for more information on the instruction cache.

1.4 DATA CACHE

The data cache is also 16 KBytes in size and direct mapped. The data cache is virtually indexed, allowing data cache to be indexed at the same time as the translation lookaside buffer (TLB) is performing the virtual-to-physical address translation. If there is a hit to the primary data cache, data can be written or read in the same clock as the TLB lookup. In the case of a primary data cache miss, the virtual-to-physical address translation has already occurred and the secondary cache access can begin immediately. The data cache is physically tagged in order to maintain coherency with the secondary cache.

The data cache implements a write-back protocol and has a 32-byte fixed line size. The data cache also implements the *'streaming'* architecture, as defined in the instruction cache section above, and has a 32 byte write-back buffer. Each incoming data cache line is compared to a dirty bit RAM. Any dirty lines are written to the write-back buffer before the actual cache line can be filled. The subsequent write-back does not occur until after the line fill is completed.

The data cache has a flush mechanism by which all data cache lines can be invalidated with one operation. Note that modified data is not written out before being invalidated.

To enhance data cache performance, a dedicated Data cache TLB is used to translate data cache addresses. This cache is divided into two sections, a 4-entry, fully associated content addressable memory (CAM), and a 256-entry direct mapped RAM. In the multi-level address mapping scheme of the TurboSPARC processor, the CAM is dedicated to translating level 1 and level 2 page table entries, while the RAM translates level 3 entries. The virtual address mapping scheme is discussed in detail in Section 6, Programming Interface. Refer to Section 3.3 for more information on the data cache.

1.5 MEMORY MANAGEMENT UNIT

The memory management unit (MMU) of the TurboSPARC microprocessor performs virtual-to-physical address translation as well as memory protection. Virtual-to-physical translation is done using a 4 KByte fixed page size. Any virtual page can be mapped to any available physical page. Memory protection is provided so that one process cannot read or write to the address space of another process. This allows multiple processes to reside in physical memory at the same time.

The MMU implements a three level virtual address mapping scheme, called a '*table walker'*. On a TLB miss, table-walking hardware generates the physical address for the requesting TLB virtual address by 'walking' through the three levels of page tables. The first and second levels contain *'Page Table Pointers'* (PTP), which are pointers to the next level down. The first level contains 256 entries, while the second and third levels each contain 64 entries. The third level is always a *'Page Table Entry'* (PTE), which points to a physical page. Once the physical address pertaining to the virtual address of the TLB miss is determined, the address is driven out onto the bus and a secondary cache access is initiated.

1.6 BUS INTERFACE UNIT

The bus interface unit of the TurboSPARC microprocessor includes a DRAM interface, a secondary cache interface, an I/O (SBus) interface, and a graphics interface (AFX). The DRAM interface generates all standard DRAM control signals and can support up to 256 MBytes of memory divided into eight 32 MByte banks. A 64-bit external data bus allows a cache line fill to be accomplished in four sequential memory transfers.

The Sbus interface supports up to five slots and contains a dedicated 16-entry IOTLB. The SBus supports I/O transactions between the processor and other SBus devices. The SBus has a direct virtual address memory interface and works

in conjunction with the MMU to arbitrate for system and memory resources.

TurboSPARC Architecture

2.1 INTEGER UNIT AND FLOATING POINT CONTROLLER

The integer unit (IU) and floating point control (FPC) are merged into a 9-stage pipeline. Below are some of the features of the IUFPC.

- 9 stage instruction pipeline.
- No branch folding. Branch instructions (taken/non-taken) execute in one cycle.
- Harvard architecture.
- IMUL and IDIV implemented using Floating Point Multiplier and Divider.
- Three read and one write port 8-Window register file.
- Single cycle store operation.
- Fully integrated pipelined FPC within IU pipeline.

2.1.1 Integer Unit

Figure 2-1. shows the stages and sequence of the TurboSPARC pipeline.

I	F	D	E	M	R	W	FR	FW

Figure 2-1. 9-Stage Instruction Pipeline

I - Issue:	In the I-stage, the Instruction Virtual Address(IVA) is presented to the Instruction Cache. The IVA is actually generated in either the D or E stages. However, the issue stage is where the IVA/PC pipeline originates. The IVA generation logic is designed to allow for an early arrival of the IVA.
F - Fetch:	In the F-stage, the instruction cache (I-cache) is accessed and an instruction pair is returned. The instruction pair, and an F-stage Prefetch Buffer, is used to perform speculative fetching, in order to avoid any cycle penalty on branches taken or not-taken. Regardless of the availability of instructions, at most one instruction is issued to the Decode stage.
D - Decode:	In the D-stage, the single instruction received from the F-stage is decoded, and two integer operands(*rs1* and *rs2*) are generated. The two operands can be read from either the IU Register File (IU_RF), or forwarded from the E, M, R, or W stages. The Decode stage is also where any FP structural & data hazards are detected. The decode stage also contains an adder to compute the Branch and CALL target address.
E - Execute:	The E-stage contains a 32-bit left & right shifter and a SPARC ALU. The shifter and ALU are used to compute integer arithmetic and logical operations. The adder in the ALU yields the data reference address for LD/ST, and RETT/JMPL instructions.
M - Memory:	In the M-stage, all IU and FP Load (LD) and Store (ST) instructions check the Data cache tag. In addition to the tag lookup, data cache reads are also performed in the case of an LD instruction. For ST instructions, the store align is completed, with the store data being posted to a “store-data-holding register”. Posted store data is written out to the data cache whenever a vacancy appears.

	The store aligner is also used to perform the FPcpy (*fmov/fabs/fneg*) instructions.
R - Defer:	The purpose of the R-stage is to be able to cancel any write data going into the IU-Register File
W - Write:	All integer results are written to the IU-RF in this stage. In the case of a floating point divide (DIV) or square root (SQRT) instruction, that instruction is posted to a special holding slot. Also in the W-stage, traps are issued according to any pending trap. If a trap was pending, it is posted to the I-stage.
FR - FP Defer:	All FP ALU and multiply (MUL) related instructions complete in the FR-stage. In addition, the result of floating point DIV and SQRT instructions are posted. ALU operations and MUL instructions complete in 4 cycles and are synchronized with the IU/FPC pipeline. Floating Point DIV and SQRT instructions complete in approximately 25 cycles. Due to the FP instruction scheduling in the D-stage, at most one FPop instruction will complete in the FR-stage. Posting FP results to the FP Register-File is only done if the present (and prior) instruction was completed without any FP exception. If an FP instruction (or prior) completed with an execution, the corresponding program counter value and corresponding opcode is posted to the Floating point Queue (FQ).
FW - FP Write:	In the FW-stage, any posted write is written to the floating point register file, and any faulting instructions are inserted in the floating point queue. The floating point status register (FSR) is also updated in this stage.

2.1.2 Floating Point Controller

The Floating Point Controller resides in stages D through FW. All floating point instructions are tested for structural and operand hazards in the D-stage. FPop operands are gathered by reading the floating point register file and are dispatched to the FPU in the E-stage. In general, all FPop instructions complete in the FR stage, and the result is written to the floating point register file in the FW stage. The FP-RF shares the FW stage with a 3-entry Floating Point Queue, and the Floating Point Status register.

The floating point controller (FPC) can at most issue one floating point related instruction. FP-CPY instructions (*fmovs/fnegs.fabs*) and floating point load and store instructions complete in M stage and the result is pipelined through to FR stage.

All floating point ALU and MUL instructions complete in the FR stage, regardless of subnormal or normal operands or results. Floating point DIV instructions are pipelined in the M, R, and W stages, after which they enter a DIV/SQRT holding slot. These instructions complete in the FR stage once the floating point divide unit has completed execution and there is no floating point instruction in W stage. Hence, at most one floating point instruction can complete in FR stage, leaving the floating point register file with 1-write and 2-read ports.

The floating point queue is 3-entries deep, with each instruction having a corresponding address (PC) entry. FPop instructions only enter the floating point queue upon detecting an FP exception.

The integer multiply and divide instructions are also dispatched to FMUL and FDIV units respectively in the E-stage. The pipeline is held until the results are available and then written to integer unit register file in W stage.

There are two types of hazards that must be handled by the Floating Point Controller:

- Structural: Where the physical structure of the design doesn't allow overlapping of certain combinations of instructions- i.e. a resource conflict.
- Data: Where, due to pipelining, the results of an instruction are needed before that instruction is completed.

2.1.2.1 FPU Structural Hazards

In the TurboSPARC microprocessor, there are two kinds of structural hazards:

- The Falu (for Floating Point Adds, Subtracts, Compares, and Converts), Fmul (for Floating Point Multiplies), and the Fdiv (for Floating Point Divides and Square Roots) operations are not pipelined. Hence, once an operation has begun, another operation requiring the same resources cannot begin until the previous operation has completed. The obvious solution is to not launch a FPop (Floating Point Operate) instruction when the target arithmetic unit is busy. This is done by comparing the type of instruction in the D stage with the type of instruction in E, M, and R stages. If the FP arithmetic units match, assert ifde_hold and send a nop to E. For an FP Divide or Sqrt in D stage, the instruction type in E, M, R, W, and DS stages must be compared to see if any are FP Divides or Square roots.

- Since the FDIV unit takes 15-25 cycles to reach the FR stage, while all other FP instructions take 3 cycles, there could be an FDIV and another floating point instruction (FPop or Load-Float) that both reach the FR stage at the same time, causing a bus-contention hazard. TurboSPARC gives the 3-cycle FP instructions priority over a finished FP Divide instruction. The FP Divide instruction can move from DS (Divide Slot) stage to FR stage as soon as the there is an opening. An integer instruction, or a NOP-ed floating point instruction in W stage allows the DS divide instruction to proceed to the FR stage in the next clock cycle. It would seem like TurboSPARC should give priority to the oldest instruction in the pipeline (which would be the divide), and thus by giving it lower priority performance would give reduced. But real code will probably have an instruction that is dependent on the divide waiting in E stage for the result, when the divide finishes. Thus the instruction in W stage would be a bubble (NOP-ed).

2.1.2.2 FPU Data Hazards

There are two classifications of data hazards, both of which can occur in the FPU:

- Read After Write: An instruction tries to read a source before a previous instruction writes it, and consequently fetches the old value.
- Write After Read: An instruction tries to write a destination before a previous instruction has read it, causing the earlier instruction fetch the wrong value. In this single-issue, in-order-issue pipeline, WAR Hazards should not occur.

2.1.3 Pipeline Interlocks

- The pipeline is interlocked under the following conditions:
- Single cycle integer load usage interlock
- Single cycle integer multiply/divide usage interlock
- Single cycle interlock on a CTI pair (any CTI/(bicc, bfcc, call))
- Single cycle interlock on a store followed by load to the same address
- 6 cycle interlock on Fcmp followed by Fbfcc
- 4 cycle interlock on back-to-back FP-ALU or FP-MUL instructions
- Variable cycle interlock for back-to-back FP-DIV single/double instructions

2.1.4 Instruction Timings

Table 2-1. shows a listing of the various instructions and the number of cycles required to complete them.

Table 2-1. TurboSPARC Instruction Timings

Instruction	Cycles
Unsigned Integer Load	1
Signed Integer Load	2
Integer Single (Single)	1
Integer Load/Store (Double)	2
Atomics	3
FP Load/Store	1
CALL, Bicc, Bfcc	1
JMPL, RETT	2
Ticc (taken)	9
WRSPR	2
WRWIM, WRTBR, WRY	1
Arithmetical, Logical	1
Integer Multiply	7
Integer Divide	8-33
FP-ALU (faddx, fsubs, fxtox)	4
FP-MUL (fmulx, fsmuld)	4
FP-DIV (fdivd)	8-50 (35 Typical)
FP-DIV(fdivds)	8-27 (21 Typical)
FP-SQRT (fsqrtd)	8-50 (35 Typical)
FP-SQRT (fsqrts)	8-27 (21 Typical)
FP-CPY (fmovs, fnegs, fabs)	1

2.1.5 IU Registers

The IU register set contains four basic registers. Each register is described below.

2.1.5.1 Processor Status Register

The processor status register (PSR) is a 32-bit register which contains mode, control and status information. Configuration information is contained in the CPU configuration register (CCR). This register is part of the MMU register set and resides in a unique virtual address space. Figure 2-2. shows the bit definitions of the PSR.

31 28	27 24	23 20	19 14	13	12	11 8	7	6	5	4 0
impl	ver	icc	rsv	EC	EF	PIL	S	PS	ET	CWP
4	4	4	6	1	1	4	1	1	1	5

Figure 2-2.Processor Status Register

IMPL The 4-bit Implementation field is hard-wired and together with the VER field define the TurboSPARC processor as a unique implementation of the SPARC v.8 architecture. The IMPL field is read-only and returns 0x0.

VER The 4-bit Version field works in conjunction with the IMPL field to define the TurboSPARC processor as a unique implementation of the SPARC v.8 architecture. This field is read-only and returns 0x5 normally, 0x4 when in uSPARC2 mode.

ICC The 4-bit Integer-Condition-Code field contains the IU condition codes. The ICC field is modified by specific arithmetic and logical instructions, and by the WRPSR instruction. Other specific instructions cause a transfer of control based on the value of these four bits. Each bit is defined as shown in Table 2-2.

Table 2-2. Integer condition Code Fields

Icc Bit #	Description
23	Negative: Indicates whether a 32-bit 2’s complement ALU result was negative for the last instruction that modified the ICC field. 1 = Negative. 0 = Not Negative
22	Zero: Indicates whether the ALU result was zero for the last instruction which modified the ICC field. 1 = Zero. 0 = Non-zero
21	Overflow: Indicates whether the ALU result was within the appropriate range for the last instruction which modified the ICC field. 1 = Overflow. 0 = No overflow.
20	Carry: Indicates whether a 2’s complement carry out or borrow occurred for the last instruction which modified the ICC field. 1 = Carry. 0 = No carry.

rsv The Reserved field returns zero’s when read by the RDPSR instruction. Conversely, execution of the WRPSR instruction should assure that these bits are zero.

EC The Enable Coprocessor bit indicates whether the coprocessor is enabled. In the TurboSPARC processor the coprocessor is disabled and this bit is hard-wired to zero. Execution of a coprocessor instruction causes a trap.

EF The Enable Floating-point bit indicates whether the FPU is enabled. If the bit is set (1), the FPU is enabled. If the bit is clear (0), execution of a floating-point instruction causes a trap.

PIL The 4-bit Processor Interrupt Level Field indicates the interrupt level above which the processor will accept an interrupt. The ET bit must be set in order for the PIL field to have meaning. The incoming interrupt is compared to the value in the PIL field. If the value on the IRL[3:0] pins is greater than that of the PIL field, or if the interrupt is level-15, the processor takes an interrupt request trap. Refer to the SPARC Architecture Manual v.8, chapter 7, for more information.

S The Supervisor bit indicates whether the processor is in supervisor or user mode. 1 = supervisor, 0 = user.

PS	The Previous Supervisor bit contains the value of the S bit at the time of the most recent trap.
ET	The Enable Traps bit indicates whether traps are enabled. When traps are disabled (ET=0), interrupt requests are ignored and an exception trap causes the IU to halt execution. A trap automatically clears the ET bit. If traps are to be enabled, software must set the bit (ET=1).
CWP	The 5-bit Current Window Pointer field contains a pointer which identifies the current window in the IU registers. The pointer field is actually an 8-bit counter which is incremented during RESTORE and RETT instructions, and decremented on a SAVE instruction. CWP[4:3] are hard-wired to zero.

2.1.5.2 Window Invalid Mask Register

The Window Invalid Mask (WIM) register is accessible in supervisor mode via the RDWIM and WRWIM instructions. During execution of the SAVE, RESTORE, or RETT instructions, the WIM register is used by hardware to determine whether an overflow or underflow trap should be generated. The lower eight bits in the WIM each represent one window. Hence up to 8 windows can be supported. Bits [31:8] are hard-wired to zero. During execution of the SAVE, RESTORE, or RETT instructions, the current window pointer (CWP) value is compared to the WIM. If one of these instructions causes the CWP to point to a window whose corresponding WIM bit is set (equals 1), a window overflow or underflow trap is generated. Bits corresponding to unimplemented windows are read as zero. Figure 2-3. shows the format of the WIM register.

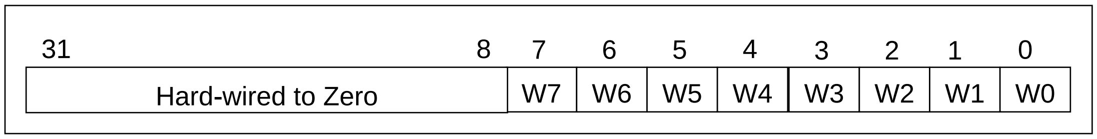


Figure 2-3. Window Invalid Mask Register

2.1.5.3 Trap Base Register

The Trap Base Register (TBR) contains the virtual address where control is transferred on a trap, as well as an 8-bit trap type field which is written by hardware and indicates the type of trap. Figure 2-4. shows the format of the TBR register.

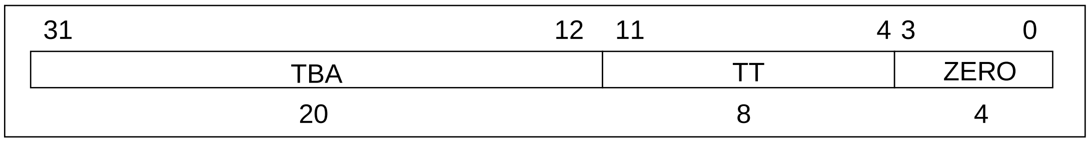


Figure 2-4. Trap Base Register

TBA	The trap base address is written during execution of the WRTBR instruction in supervisor mode. The TBA is the only field of the TBR register which is written by software.
TT	The 8-bit trap type field is written by hardware when a trap occurs and provides an offset into the trap table. The TT field remains static until the next trap, where it is again written by hardware. Execution of the WRTBR instruction does not affect the TT field.
ZERO	The 4-bit ZERO field always returns a value of zero. This field is not affected by execution of the WRTBR instruction.

2.1.5.4 Multiply/Divide Register

The Multiply/Divide(Y) register can be written or read using the WRY and RDY instructions and contains one of the following formats based on the type of mathematical calculation being executed.

1. Integer multiplication: The Y register contains the most significant word of a double precision product of either an integer multiply instruction, or a routing which uses the integer multiply step instruction.
2. Integer Divide: The Y register contains the most significant word of a double precision dividend of an integer divide instruction.

2.1.5.5 Floating Point Status Register

The FPU register set contains thirty-two 32-bit general purpose registers. An instruction can access any of the 32 registers at any time. These registers are accessed by FPop instructions as well as single and double precision floating point load and store instructions. A single register can hold one single precision operand. Double precision operations require two aligned registers.

In addition to the general purpose registers, The FPU status register contains mode, status, and configuration information about the FPU. Figure 2-5. shows the format of the Floating Point Status (FSR) register.

31 30	29 28	27 23	22	21 20	19 17	16 14	13	12	11 10	9 5	4 0
RD	U	TEM	NS	RES	VER	FTT	QNE	U	FCC	AEXC	CEXC
2	2	5	1	2	3	3	1	1	2	5	5

Figure 2-5. Floating Point Status Register

RD The 2-bit rounding direction field selects the rounding direction for floating point values based on ANSI/IEEE standard 754-1985.

00 = Round toward nearest even number

01 = Round toward zero

10 = Round toward positive infinity

11 = Round toward negative infinity

U The unused field pertains to bits 29:28, and bit 12. Software must assure that LDFSR instructions contain zero values for these bits.

TEM The 5-bit trap enable mask field contains the five enable bits for the five floating point exceptions which can be indicated in the CEXC field. An FP exception trap is generated whenever an FP operate instruction generates an exception and the corresponding TEM bit is set (1). If the corresponding TEM bit is clear (0), a trap is not generated by the exception.

NS When set, the non-standard bit can cause the FPU to generate results which may not be compatible with ANSI standards. This bit is hard-wired to zero.

RES The 2-bit reserved field returns a zero value when read.

FTT The 3-bit floating point trap type field encodes the type of floating point exception. The value is retained until either an STSFR or FPop instruction is executed. The LDFSR instruction does not affect the FTT field. Table 2-3. shows the encoding of the FTT field. Note that FTT values 5, 6, and 7 are not supported.

Table 2-3. FTT Field Encoding

FTT value	Trap Type
0	None
1	IEEE 754 exception
2	Unfinished FPop
3	Unimplemented FPop
4	Sequence error
5	Hardware error (not supported)
6	Invalid FP register (not supported)
7	Reserved (not supported)

QNE Floating point queue not empty: set when there is a valid entry in the queue.

FCC The 2-bit floating point condition code field corresponds to the single or double values in the floating point registers specified in the *rs1* and *rs2* fields of the instruction. The FTT field is updated by floating point compare instructions. The field is read by the LDFSR instruction, and written by the STFSR instruction. The floating point condition codes are shown below. *frs1* and *frs2* correspond to single, double, or quad precision values in the floating point registers specified by the rs1 and rs2 fields of a given instruction. The '?' represents an unordered relation. The FCC field remains unchanged if execution of either the FCMP or FCMPE instructions generates an IEEE_754_exception trap.

00	frs1 = frs2
01	frs1 < frs2
10	frs1 > frs2
11	frs1 ? frs2

AEXC The 5-bit accrued exception field represents five different types of accrued floating point exceptions and accumulates them while the exception traps are masked using the TEM field.

CEXC The 5-bit current exception field represents five different types of current floating point exceptions and indicates that one or more were generated by the most recently executed FPop instruction.

For more information on the floating point exception fields, refer the SPARC v.8 architecture manual.

2.1.6 Traps

Several instructions in the SPARC architecture induce exceptions or traps. All exceptions related to the IU are precise traps meaning these traps are recognized before any program-visible state has been changed by the trap-inducing instruction. Exceptions generated by the floating point arithmetic units are deferred traps which are recognized by a floating point instruction which may be immediately or several instructions following the trap inducing instruction. Since floating point loads and stores are handled by the IU/FPC pipeline, exceptions induced by these instructions are precise traps.

Associated with every stage in the pipeline, there are some common signals which can be used to implement traps effectively.

x_nop:	When this signal is asserted, the instruction corresponding to the pipeline stage is ignored. Normal activities such as updating register file, special registers etc. are disabled. The instruction needs to be ignored if it happens to be the delay slot of an annulled branch, load usage interlock, instruction induced a trap or a trap was recognized ahead in the pipeline.
x_buddy:	This bit is asserted if the instruction corresponding to the pipeline stage happens to be the second cycle of a multicycle instruction. No integer instruction takes more than two cycles to execute with the exception of integer multiply and atomic load/stores. This bit is not asserted for integer multiply as the pipeline is held until the multiply completes. Buddy cycles of instructions cannot induce or recognize traps.
x_trap_pending:	This bit is asserted if a trap is recognized in a particular pipeline stage.
x_void_op:	This bit is set if x_trap_pending is asserted or if a trap is recognized ahead in the pipeline. The purpose of this signal is to void all instructions down the pipeline once a trap is recognized in a particular stage.
x_trap_type[7:0]:	This bus indicates the trap type seen in a particular pipeline stage. Several traps may be induced by a single instruction in a particular stage. These traps are priority encoded as shown in Table 2.4 and the appropriate trap type is generated to be written later into the Trap Base Register(TBR).

All of the above signals except x_void_op are pipelined. x_void_op is an asynchronous signal starting from the W stage all the way down the pipeline. When a trap is recognized in a particular stage, x_nop, x_trap_pending, x_void_op and x_trap_type are asserted and pipelined through the various stages.

If traps are enabled, the IU/FPC pipeline takes a trap on an instruction when the instruction reaches W stage. Standard trap actions such as updating PSR and TBR take place. The PC is written into the IU register file and in the following cycle nPC is written to the IU register file and the trap address is provided to the instruction cache. To make sure nPC is written if a multicycle instruction traps, the buddy cycle of a multicycle instruction carries the next PC rather than the current PC.

However, if traps are disabled, error_mode is detected in R stage and standard trap actions such as updating PSR and saving PC and nPC do not take place. The TBR is updated only when a RETT instruction traps when traps are disabled.

If traps are enabled, the IU/FPC pipeline takes a trap on an instruction when the instruction reaches W stage. Standard trap actions such as updating PSR and TBR take place. The PC is written into the IU register file and in the following cycle nPC is written to the IU register file and the trap address is provided to the instruction cache. To make sure nPC is written if a multicycle instruction traps, the buddy cycle of a multicycle instruction carries the next PC instead of the current PC.

2.1.6.1 Floating Point Queue

Since floating point instructions finish execution after they leave the W stage (and for FP Divides, long after leaving the W stage), floating point exceptions are implemented as deferred, and not precise. If a Floating Point operate instruction (FPop-- FAdd, FMul, FDiv...) cannot complete normally (example FAdd: overflow; FDiv: Divide by zero), the instruction opcode and its PC value are written into the Floating Point Queue (FQ), and the FPC enters exception-pending state. The next valid FP instruction (FPop, FP Load/Store, or FBfcc) that enters M stage will recognize that the FPC is in exception pending state and will take an fp_exception trap. The floating point instructions (FAdd, FSub, Fxtoy (conversions), FCmp, FMul, FDiv, FSqrt) between the instruction that caused the FPC to enter exception-pending state, and the instruction that recognized that the FPC was in exception pending state, are written into the FQ also. Thus fp_exceptions are precise, except for the instructions that are in the FQ. Note that FP copy operations (FMov, FNeg, FAbs) and FP Load/Stores do complete while the FPC is in exception-pending state.

Table 2-4. Exception and Interrupt Request Information

Exception/Interrupt Request	Priority	Pipeline stage recognized	x_trap_type[7:0]
reset	1	w	undefined
data_store_error	2	not implemented	
instruction_access_MMU_miss	2	not implemented	
instruction_access_error	3	d	0x21
r_register_access_error	4	not implemented	
instruction_access_exception	5	d	0x01
priveleged_instruction	6	d	0x03
illegal_instruction	7	e	0x02
fp_disabled	8	d	0x04
cp_disabled	8	d	0x24
unimplemented_flush	8	not implemented	
watchpoint_detected	8	not implemented	
window_overflow	9	d	0x05
window_underflow	9	d	0x06
mem_address_not_aligned	10	e/m	0x07
fp_exception	11	m	0x08
cp_exception	11	not implemented	
data_access_error	12	r	0x29
data_access_MMU_miss	12	not implemented	
data_access_exception	13	r	0x09
tag_overflow	14	m	0x0a
division_by_zero	15	not implemented	
trap_instruction	16	m	0x80 - 0xff
interrupt_level_15	17	e	0x1f
interrupt_level_14	18	e	0x1e
interrupt_level_13	19	e	0x1d
interrupt_level_12	20	e	0x1c
interrupt_level_11	21	e	0x1b
interrupt_level_10	22	e	0x1a
interrupt_level_9	23	e	0x19
interrupt_level_8	24	e	0x18
interrupt_level_7	25	e	0x17
interrupt_level_6	26	e	0x16
interrupt_level_5	27	e	0x15
interrupt_level_4	28	e	0x14
interrupt_level_3	29	e	0x13
interrupt_level_2	30	e	0x12
interrupt_level_1	31	e	0x11

2.2 FLOATING POINT UNIT

The floating point unit (FPU) is made up of three arithmetic blocks (FALU, FMUL, FDIV) along with a fourth block (FPUIO) which serves as the interface between the three blocks and the Integer Unit/Floating Point Controller (IUFPC).

The floating point unit implements all FPops except for those involving quad format operands or results. For quad operations, the IUFPC takes and unimplemented instruction trap. All other operation/operand combinations are handled by the FPU; no unfinished floating point exceptions are created (FMOVs, FNEGs and FABSs are actually implemented in the IUFPC).

All results created by the FPU are in compliance with the IEEE 754 standard. They are also in compliance with the recommendations in the version 8 SPARC Architecture Manual, Appendix N. The FPU always operates in “standard” mode; that is, the NS bit of the FSR is always zero.

2.2.1 FPUIO

The FPUIO serves as an interface between the three FPU arithmetic blocks, the IUFPC, and the data cache. Operands for a given FPU instruction can be sourced by the IUFPC/FP registers, the data cache, or the output of one the FPU blocks themselves. Operand selection is done by the FPUIO based on control signals from the IUFPC.

The FPUIO also sends the correct FPU result back to the IUFPC by selecting the result from the correct arithmetic unit. The FP exception information is combined and sent to the IUFPC.

2.2.2 FALU

The FALU block executes the following instructions:

FADD.d, FADD.s, FSUB.d, FSUB.s, FCMP.d, FCMP.s, FCMPE.d, FCMPE.s, FdTO.s, FdTO.d, FiTO.d, FiTO.s, FdTOi, FsTOi.

The FALU is not pipelined and is given four cycles to complete each operation. The FALU can not handle any instructions involving quad format. However, it can handle any combination of FP data types in the single and double formats. No unfinished exceptions are generated.

2.2.3 FMUL

The FMUL block executes the following multiply instructions:

FMUL.d, FMUL.s, FsMUL.d, sMULcc, sMUL, uMULcc, uMUL.

The FMUL is not pipelined and is given four cycles to complete each operation. The FMUL can not handle any instruction involving quad format. However, it can handle any combination of integer of floating point data types in single and double formats. No unfinished exceptions are generated.

2.2.4 FDIV

The FDIV executes the following SPARC divide and square root instructions:

FDIV.d, FDIV.s, FSQRT.d, FSQRT.s, sDIVcc, sDIV, uDIVcc, uDIV.

The FDIV can take a variable number of cycles based on the data format and type. The actual numbers are listed in Table 2.1. The FDIV sends a signal to the IUFPC when the results are ready. Another requirement of the FDIV block is that the divisor be positive for integer divides; the IUFPC does any necessary conversion. The FDIV can not handle any instruction involving quad format. However, it can handle any combination of integer or floating point data types in the single and double formats. No unfinished exceptions are generated.

2.3 INTERRUPTS AND WATCHDOG DETECTION

The TurboSPARC microprocessor provides the CP_STAT[1:0] outputs which indicate that either a watchdog reset is ongoing (CP_STAT[1] high), or an interrupt-level 15 should be asserted (CP_STAT[0] high).

The watchdog reset sequence occurs when the Integer Unit (IU) attempts to take a trap when the ET bit of the Processor Status Register (PSR) is zero. The processor internally forces a reset to the IU only and sets the BM bit in the Memory Management Unit (MMU) Control Register (CR), located at VA=0/ASI=4. Once the reset sequence is completed, the processor starts fetching instructions from boot-PROM space (VA=oxo/PA=0x70000000. The assertion of CP_STAT[1] should be latched in an external device in order for the boot code to detect the shutdown case. The CP_STAT[1] signal remains asserted for more than 1 µS during a watchdog reset sequence.

The request for interrupt level-15 is generated in any of the folowing conditions:

1. An SBus Late error occurs during a PIO write transaction.
2. An SBus Timeout/Error acknowledgment occurs during a PIO read or write transaction.
3. The physical address points to an address space other than memory during a write-back.
4. An AFX Timeout occurs during a write operation.
5. Memory parity errors.

Once CP_STAT[0] is asserted, an external device such as an interrupt controller responds by asserting IRL[3:0] to 0xF (non-maskable interrupt)

On interrupts generated due to SBus transactions, the fault status and fault address are updated in the AFSR and AFAR registers. On interrupts due to AFX timeouts, status and address are updated in the MFSR and MFAR registers. Interrupts generated externally are driven to the processor through the IRL[3:0] input pins and are detected by the processor in the Execute stage. Interrupts are ignored if the Enable Trap bits in the Processor Status Register (PSR) are disabled. Refer to Section 2.1.5.1 for more information on the Processor Status Register.

2.4 RESET

The TurboSPARC microprocessor can be forced into a reset state at any time by asserting INPUT_RESET. The processor reset state is defined as follows:

1. VCO & PLL remain active
2. All clocks remain operational, including SBus and AFX
3. All output-only and bi-directional signals are brought to a high-impedance state through an asynchronous path from INPUT_RESET. Note that output-only signals have an internal pull-up and are floated during the reser sequence.
4. The following register fields are forced to known states as shown in Table 2-5.

Table 2-5. Initializing Register Fields on Reset

Register Name	Bit or Field Name	State	Default Condition
Program Counter	---	0x0	Initialize program counter to 0 zero.
Processor Status (PSR)	ET	0	Traps are disabled
	S	1	Puts procesor in supervisor mode.
MMU Control (MMUCR)	PMC	000	DRAM Page mode Control. RAS_ is negated at the end of each memory cycle.
	PE	0	Parity is disabled.
	PC	0	Even parity.
	BM	1	Causes instructions to fetch from PA[30:27] = 0x7. Only relevant if ME = 0.
	RFR	0000	DRAM refresh. Default is IOCLK / 780.
	IE	0	Instruction cache is disabled.
	DE	0	Data cache is disabled.
	ICS	0	Instruction cache snooping is disabled.
	NF	0	No fault bit. Data faults are acknowledged.
	ME	0	MMU Enable bit. Address translation is disabled.
CPU Configuration (CPUCR)	IOCLK	IOCLK_DIV2	Sets IOCLK speed as determined by the state of the IOCLK_DIV2 input.
	AXCLK	IOC_RANGE [1:0]	sets AFX clock speed as determined by the state of the IOC_RANGE[1:0] inputs.
	SNP	0	DVMA snooping is disabled.
	RAH	00	DRAM row address hold. Defaults to one hold cycle.
	WS	1, IOC_RANGE [1:0]	DRAM wait states. Sets the uppermost bit of the three bit field to 1. Lower two bits are determined by the state of the IOC_RANGE[1:0] inputs at reset.
	SBC	IOCLK_DIV6 [1:0]	SBus clock divide ratio. This two-bit field is set by the state of the IOCLK_DIV6[1:0] inputs at reset.
	WT	0	Write-through mode is disabled.
	uS2	0	microSPARCII compatibility mode is disabled.
	SE	0	Secondary cache is disabled.
	SCC	CPU_MODE [2:0]	Sets the secondary cache configuration as determined by the state of the CPU_MODE[2:0] inputs at reset.

2.5 INITIALIZATION

After completion of a reset sequence, some of the bits listed in Table 2.5 must be modified in order to properly enable those features listed below.

- Select the proper DRAM speed. Although two of the three bits in the WS field are initialized based on the state of the IOC_RANGE[1:0] pins, a given application may decide to overwrite this setting depending on the type of memory devices used. For example, a wait-state setting of 0x0-0x3 (WS[2]=0) permits the use of low-speed DRAM’s, whereas settings 0x4 - 0x7 permits the use of high-speed DRAM’s. This would require the boot code to set WS[2] high.
- The DRAM addres hold (RAH) parameter typically remains constant for a given application and should be initialized to the proper setting.
- AFX clock speed should be defined.
- The page mode DRAM (PMC), parity control (PC) and parity enable (PE) bits should be initialized per the system requirements.

After the above parameters have been defined, the following initialization procedure is recommended to turn-on the MMU and caches.

1. Flush MMU (refer to ASI=0x3)
2. Flush instruction cache (refer to ASI=0x36)
3. Flush data dache (refer to ASI=0x37)
4. Purge secondary cache (refer to ASI=0x30)
5. Initialize MMU page tables in memory
6. Initialize context ID and context table pointer
7. Turn-on MMU by setting the SE bit in the MMU control register
8. Turn-on instruction cache by setting the IE bit in the MMU control register
9. Turn-on data cache by setting the DE bit in the MMU control register
10. Turn-on secondary cache by setting the SE bit in the CPU configuration register

Once the above initialization procedure has been completed, optional features such as write policy, microSPARCII compatibility, and snooping can be added as necessary.

Chapter 3

Memory Management Unit / Caches

3.1 MMU OVERVIEW

The MMU, which conforms to the SPARC Reference MMU Architecture, provides three primary functions:

- Performs address translations from virtual addresses of each running process to physical addresses in physical memory. This mapping is done in units of 4KB pages so that, for example, an 8MB process does not need to located in a contiguous section of main memory. Any virtual page can be mapped into any available physical page frame.
- Provides memory protection, so a process cannot read or write the address space of another process. This is necessary for most operating systems to allow multiple processes to safely reside in physical memory at the same time.
- Implements virtual memory. The page tables track which pages are in main memory; the MMU signals a page fault is a memory reference occurs to a page not currently resident.

The main features of the TurboSPARC MMU are:

- 32-bit virtual address, & 31-bit physical address
- Fixed 4KB page size
- Support for sparse address spaces with 3-level map
- Support for linear mappings (4KB, 256KB & 16MB)
- Support for 256 contexts
- Page-level protections
- 4-entry Instruction Translation Lookaside Buffer (TLB)
- 260-entry Data TLB
- 6-entry I/O TLB
- 4-entry Page Table Pointer CAM

3.1.1 MMU Address Translation

The memory management units are responsible for translating a 32-   bit virtual address into a 31-bit physical address. The physical address consists of a physical page number (PPN), and an offset, which is a portion of the virtual address.The size of the offset changes based on the page size. For every valid virtual page resident in memory, there is a corresponding Page Table Entry (PTE) that contains the physical page number for that virtual page. Translating the virtual address to a physical address replaces the virtual page number (VPN) with a physical page number (PPN). Four GigaByte pages, which are derived from the context table, are not supported in the TurboSPARC microprocessor. Figure 3-1. shows the physical address formats for the three type of pages.

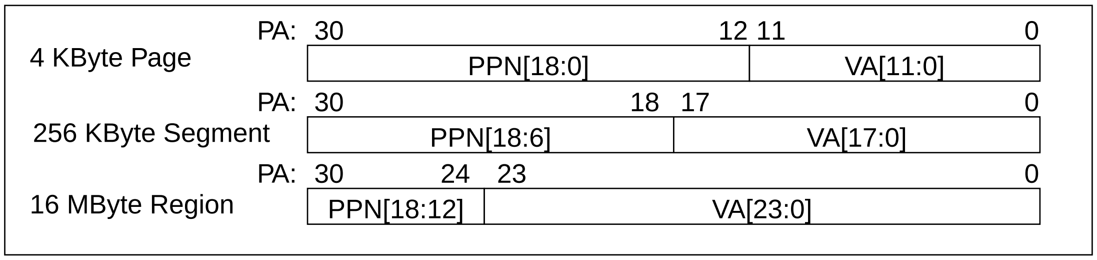


Figure 3-1. Physical Address Formats

The offset for each page size shown in Figure 3-1. is derived from a portion of the virtual address. The offset is used to address any location within a given page size. The larger the page size, the larger the offset needed to address each location within that page. Figure 3-2. shows the format of the 32-bit virtual address. The numbered indices indicate each level of the table walker and the number of virtual address bits required to address any location within that table. The table walker is discussed in Section 3.1.4.

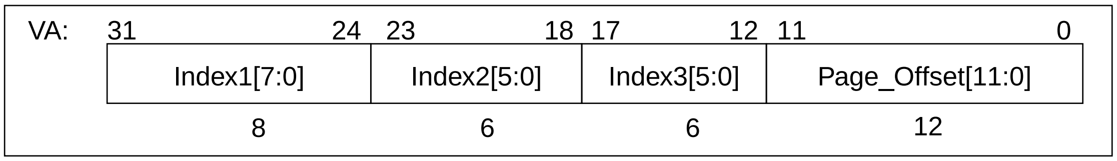


Figure 3-2. Virtual Address Format During a Table Walk

3.1.1.1 Address Translation Modes

Translation of a virtual address to a physical address depends on the instruction access type, as well as the CPU boot mode and MMU enable bits. In some cases the translation occurs, and in other cases the virtual address is passed directly to the physical address without translation. Table 3.1 shows each address translation mode along with the corresponding address space identifier (ASI) and resulting physical address.

Table 3-1. Address Translation Modes

Access Type	ASI	Boot Mode	MMU Enable	PA[30:0]
Boot Instr. Fetch	0x9:8	On	Don't Care	0x7, VA[27:0]
Pass Through	0x9:8	Off	Off	VA[30:0]
Translate	0x9:8	Off	On	(VA to PA Translation)
Pass Through	0xB:A	Don't Care	Off	VA[30:0]
Translate	oxB:A	Don't Care	On	(VA to PA Translation)
Bypass	0x20	Don't Care	Don't Care	VA[30:0]

The first three entries in Table 3.1 correspond to ASI values 0x9 and 0x8 and are used for User/Supervisor instruction accesses. The last three entries correspond to ASI values 0xB and 0xA and are used for User/Supervisor data accesses. Table 6.1 lists all of the ASI values. A description of the Boot Mode and MMU Enable bits can be found in the MMU Control Register in Section 6.2.1.

3.1.1.2 Physical Memory Map

The TurboSPARC microprocessor supports a 2 GByte physical address space. Figure 3-3. shows the physical address map.

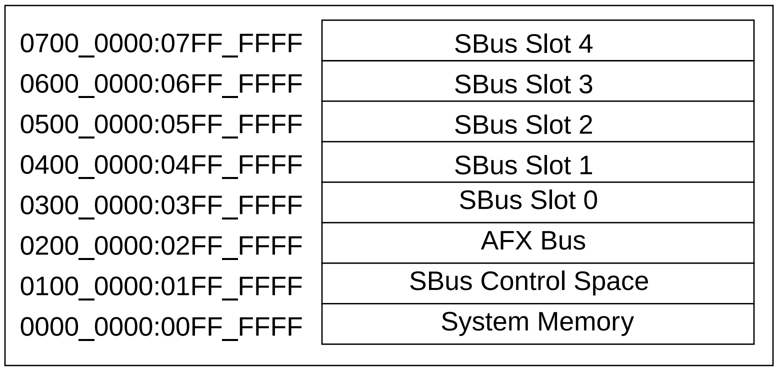


Figure 3-3. Physical Address Map

All accesses to system memory space are routed to the on-chip DRAM or secondary cache controllers. The SBus control registers are located in the SBus control space. The AFX space is dedicated for use by the Sun Microsystems compatible AFX graphics bus.

3.1.2 Page Table Descriptor

Mapping a virtual address space is accomplished using three levels of page tables. Each level of the table can contain either a Page Table Entry (PTE) for a given page size, or a Page Table Descriptor (PTD), which is used as a pointer to the next-level table. A PTD contains the physical address for a given page table. The format of a PTD is shown in Figure 3-4..

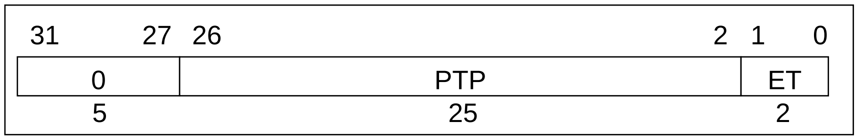


Figure 3-4. Format of a Page Table Descriptor

PTP The Page Table Pointer contains the physical base address of the next-level page table. During a TLB miss, the PTP appears on PA[30:6]. The page table which is pointed to by the PTP must be aligned on a boundary equal to the size of the page table. Table 3.2 shows the sizes of the three page tables and the context table.

Table 3-2. Page Table Sizes

Table	Page Table Size
Context	1024 Bytes
Level-1	1024 Bytes
Level-2	256 Bytes
Level-3	256 Bytes

ET The Entry Type field differentiates a page table entry from a page table descriptor.The ET field is contained in both page table descriptor and page table entries. The 2-bit ET field is encoded as shown in Table 3-3.

Table 3-3. Entry Type Field

ET	Entry Type
0	Invalid
1	PTD
2	PTE
3	Reserved

3.1.3 Page Table Entry

The first two levels of the address translation table can contain either a Page Table Descriptor (PTD) or a Page Table Entry (PTE). Level 3 contains only page table entries. For example, 16 Mbyte page table entries are contained in level-1, 256 KByte page table entries are contained in level-2, and 4 KByte page table entries are contained in level-3. Therefore, in a system which implements 4 KByte pages, level-1 would contain a PTD to level-2, and level-2 would contain a PTD to level-3. Figure 3-5. shows the format of a page table entry. This format is used throughout all table levels regardless of the page size.

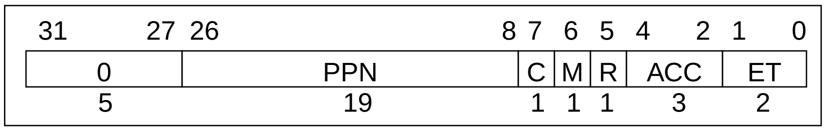


Figure 3-5.Format of a Page Table Entry

PPN	The Physical Page Number is comprised of the high-order 19 bits of the 31-bit physical address for that page. The PPN appears on bits PA[30:12] when the address translation is completed.
C	If the Cacheable bit is set, the corresponding page is cacheable by both the instruction and data caches. The C bit should be cleared for all I/O locations mapped by the MMU. If the C bit is set while the MMU page tables are being mapped, the data cache must be flushed prior to the MMU accessing the modified page tables.
R	The Reference bit is set by the MMU during page read/write accesses.
M	The Modified bit is set by the MMU during page write accesses.
ACC	The 3-bit Access Permission field indicates whether access to a given page is allowed relative to the transaction being attempted. The ASI value corresponding to the access determines the type of access as well as the permissions. Table 3.4 shows the encoding of the access permission field.

Table 3-4. Access Permission Field

ACC	User (ASI=0x9:8)	Supervisor (ASI=0xB:A)
0	Read Only	Read only
1	Read/Write	Read/Write
2	Read/Execute	Read/Execute
3	Read/Write/Execute	Read/Write/Execute
4	Execute Only	Execute Only
5	Read Only	Read/Write
6	No Access	Read/Execute
7	No Access	Read/Write/Execute

ET	The 2-bit Entry Type field must have a value of 0x2 for a page table entry. The encoding of the ET field is shown in Table 3.3.

3.1.4 Table Walker

All address translation information required by the MMU resides in physical address data structures in main memory. The MMU Table Walker fetches address translations from these data structures as needed. On an address translation miss, table-walker hardware generates the physical address corresponding to the virtual address by 'walking' through the 3-level table data located in either the Page Table Pointer (PTP) cache or in main memory. The PTP cache consists of three registers which contain the most-recently-used pointers for each of the three table levels. PTP caching is discussed in Section 3.1.6. A table-walk is also initiated on the first write to a given page, even if the page is a hit in the TLB. This is so the Modified (M) bit in the page table can be updated.

Each level can contain both Page-Table-Entries (PTE) and Page-Table-Descriptors (PTD). Figure 3-6. shows a diagram of the 3-level table walker address mapping scheme.

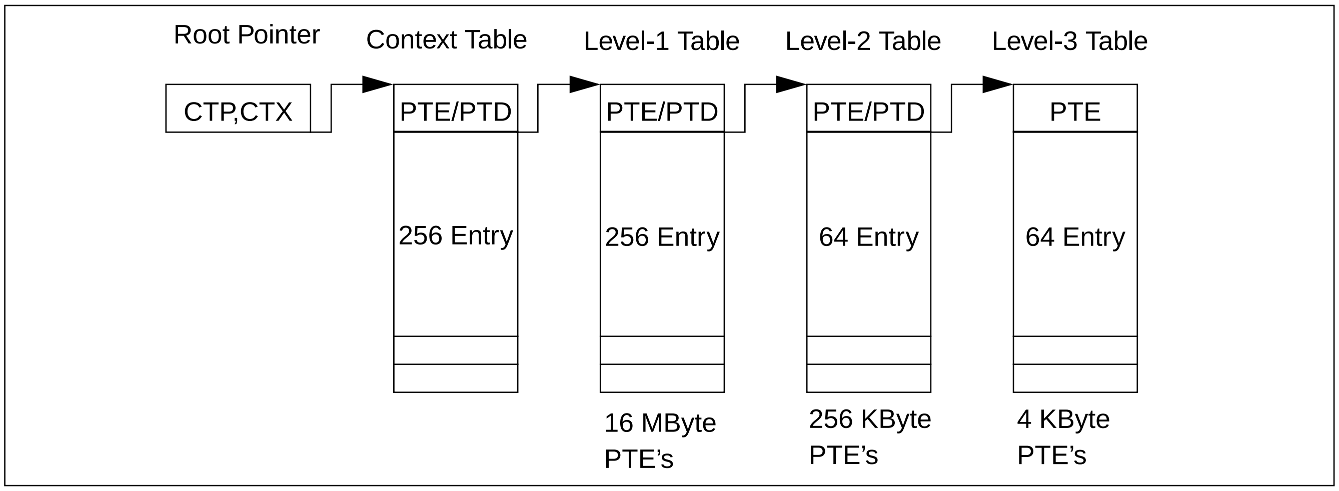


Figure 3-6. MMU Table Walker

Each table level in Figure 3-6. requires one clock cycle to access. In addition, a final update cycle may be required if the reference bit or modified bit of the page table entry needs to be set (see Figure 3-5.). Therefore, if 4 KByte paging is used and either the reference or modified bits need to be set, the complete page-level table walk can involve up to 5 memory accesses. The following sections discuss each of the table levels in Figure 3-6..

3.1.4.1 Root Pointer and Context Table

On a TLB miss, the root pointer shown in Figure 3-6. must be fetched by doing a memory access to the address formed by concatenating the value in the Context Table Pointer Register (CTPR) with an 8-bit context number. This operation fetches the base address and forms the pointer into the context table. The physical address accessed in the context table functions as a pointer to the level-1 table. 4 GByte page table entries are not supported in the TurboSPARC microprocessor, hence the context table contains only page table descriptors. Figure 3-7. shows the format of the root pointer.

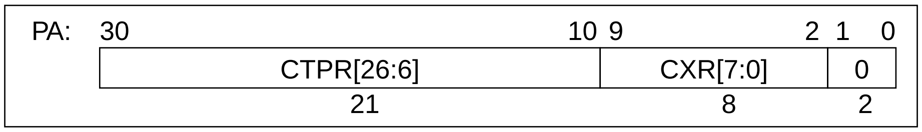


Figure 3-7. Root Pointer Format

3.1.4.2 Level-1 Table (Region)

Level-1 of the table walker contains page table entries for the 16 MByte page size. The address for level-1 table entries is formed by concatenating the root pointer with VA[31:24]. This virtual address field forms an index to the level-1 table as shown in Figure 3.2. The 8 virtual address bits are used to decode one of 256 entries in the level-1 table. The lower 2-bits of the physical address are always zero. The physical address is formed as shown in Figure 3-8..

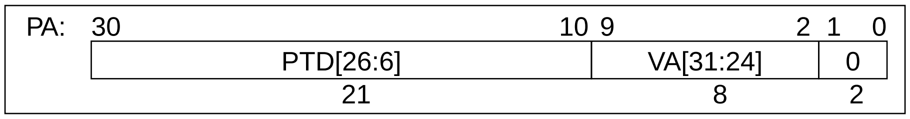


Figure 3-8. Level-1 Physical Address Format

If the 2-bit Entry Type (ET) field returned on PA[1:0] indicates an invalid page or an undefined PTE format, an exception is generated to the IU and the corresponding virtual address is loaded into the Synchronous Fault Address Register (SFAR). The SFAR is defined in Section 6.2.5.

3.1.4.3 Level-2 Table (Segment)

Level-2 of the table walker contains page table entries for the 256 KByte page size. The address for level-2 table entries is formed by concatenating bits [26:4] of the returned level-1 table entry with VA[23:18]. This virtual address field forms an index to the level-2 table as shown in Figure 3-2. The 6 virtual address bits are used to decode one of 64 entries in the level-2 table. The lower 2-bits of the physical address are always zero. The physical address is formed as shown in Figure 3-9.

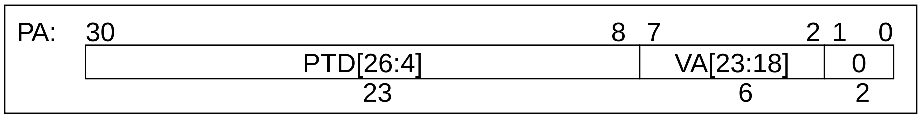


Figure 3-9. Level-2 Physical Address Format

3.1.4.4 Level-3 Table (Page)

Level-3 of the table walker contains page table entries for the 4 KByte page size. The address for level-3 table entries is formed by concatenating bits [26:4] of the returned level-2 table entry with VA[17:12]. This virtual address field forms an index to the level-3 table as shown in Figure 3-2.. The 6 virtual address bits are used to decode one of 64 entries in the level-3 table. The lower 2-bits of the physical address are always zero. The physical address is formed as shown in Figure 3-10..

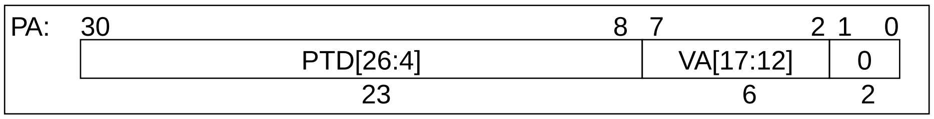


Figure 3-10. Level-3 Physical Address Format

For all level-3 accesses the table should contain only page table entries (ET=0x2). If the **ET** field indicates a PTE, that entry is loaded into the TLB. However, if the **ET** field reflects an invalid page, an undefined PTE format, or a PTD, an exception is generated.

Figure 3-11. shows the general flow of a table walk.

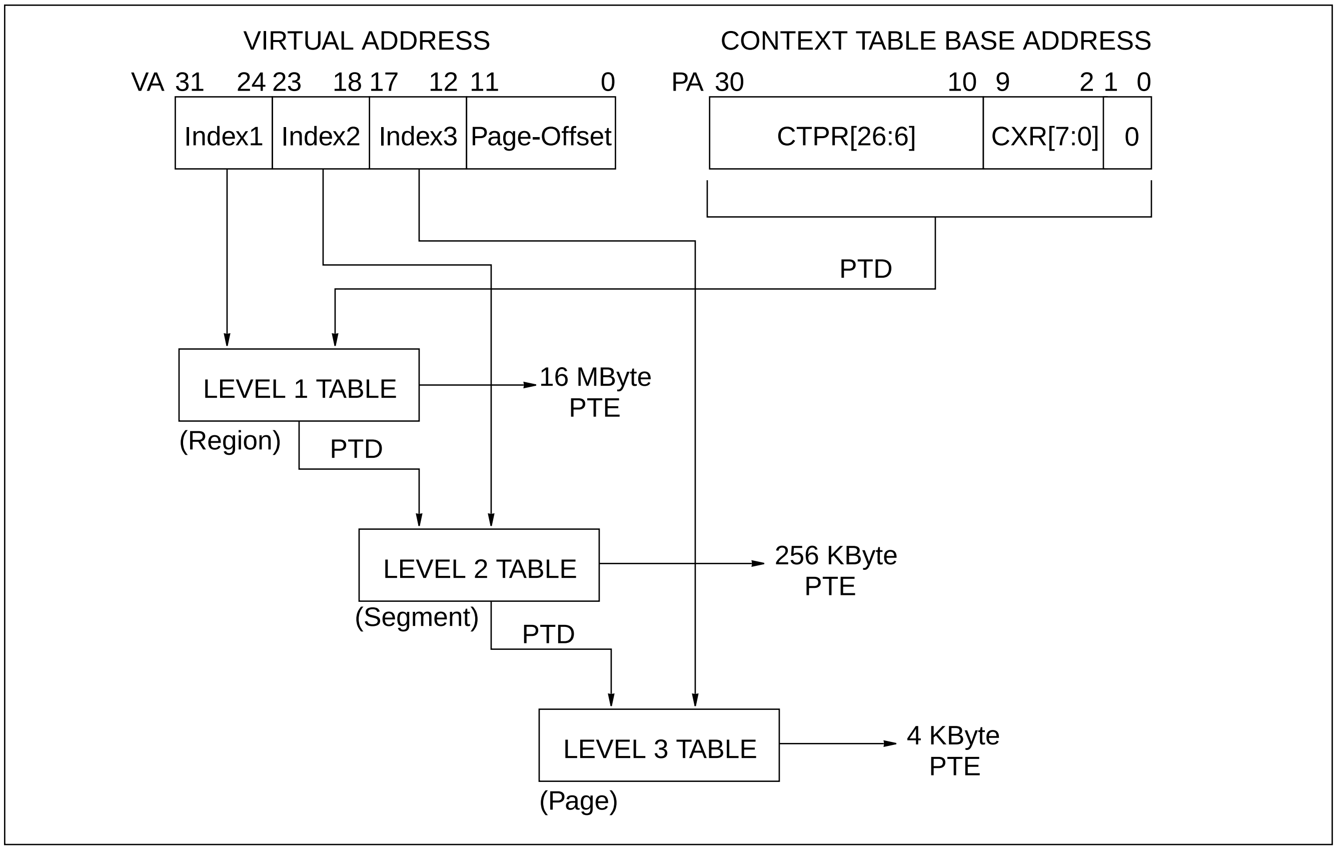


Figure 3-11. Table Walker Flow Diagram

3.1.5 Page Table Pointer Caching

As stated in Section 3.1.4, each table-walk can involve up to five memory accesses. To reduce the amount of table walks required, the table-walker hardware contains three dedicated registers for caching the most-recently-used page table pointers. The three registers are defined as follows.

1. **PTP Root-Level Pointer**: Contains data and address for the last root-level access. If the current address is found in the root-level pointer register, the PTP for the next level is available immediately. If there is no match, a memory access is generated using this address in order to fetch the root map entry. The returned data (PTP) and address will overwrite the previous data and address information stored in the root-level pointer register.
2. **PTP Region-Level Pointer**: Contains data and address for the last region-level access. If the current address is found in the region-level pointer register, the PTP for the next level is available immediately. If there is no match, a memory access is generated using this address in order to fetch the region map entry. The returned data (PTP) and address will overwrite the previous data and address information stored in the region-level pointer register.
3. **PTP Segment-Level Pointer**: Contains data and address for the last segment-level access. If the current address is found in the segment-level pointer register, the PTP for the next level is available immediately. If there is no match, a memory access is generated using this address in order to fetch the segment map entry. The returned data (PTP) and address will overwrite the previous data and address information stored in the segment-level pointer register.

In addition to these registers, a special 4-entry Content Addressable Memory (CAM) containing PTP entries is searched prior to the table walk. All three PTP registers and the CAM are invalidated during an MMU flush, whenever the context pointer is changed, and when the context ID is changed. Figure 3-12. shows a diagram of the 4-entry segment PTP cache.

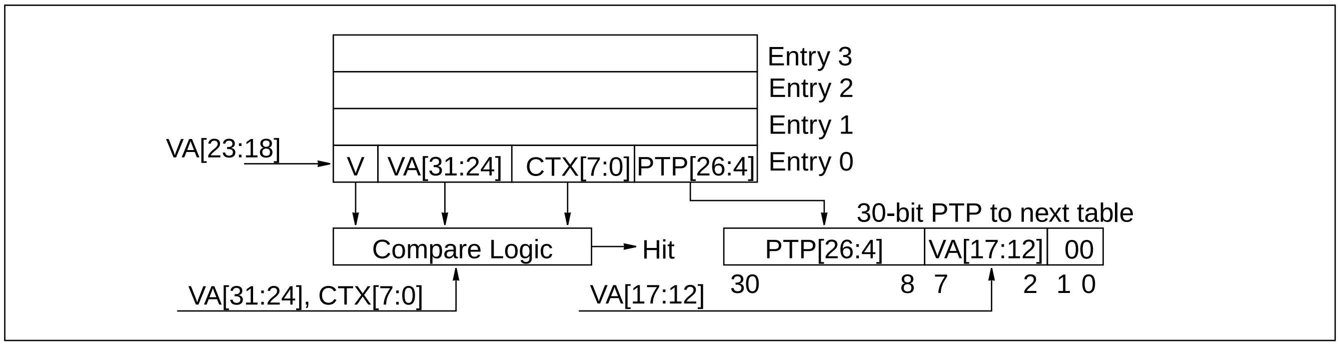


Figure 3-12. 4-Entry Page Table Pointer Cache

3.1.6 Page Table Entry Caching

The page table entries are cached in one of the following address translation caches:

1. 4-entry fully associative Instruction TLB.
2. 4-entry fully associative Segment/Region Data TLB.
3. 256-entry direct mapped Page Data TLB.

3.1.6.1 Instruction Translation Lookaside Buffer (ITLB)

The 4-entry fully associative ITLB contains the most-recently-used page table entries for the current instruction stream. The ITLB is accessed only on instruction cache misses. On an instruction cache miss, the ITLB compares the virtual address of the fetch to the tags associated with each TLB entry. The ITLB caches all three levels of page table entries, including the context table. Level-0 (Context) and level-1 (Region) entries match as if they were level-2 entries. An ITLB ‘hit’ occurs based on the match criteria shown in Table 3.5.

Table 3-5. ITLB Match Criteria

PTE Level	ACC Bits	VA[31:18]	VA[17:12]	Context	S bit in PSR
3	2 or 3 or 4	Match	Match	Match	Ignored
3	6 or 7	Match	Match	Ignored	Supervisor
2	2 or 3 or 4	Match	Ignored	Match	Ignored
2	6 or 7	Match	Ignored	Ignored	Supervisor
1	2 or 3 or 4	Match	Ignored	Match	Ignored
1	6 or 7	Match	Ignored	Ignored	Supervisor
0	2 or 3 or 4	Match	Ignored	Match	Ignored
0	6 or 7	Match	Ignored	Ignored	Supervisor

An MMU flush can be performed by execution of a Store Alternate (STA) instruction to ASI 0x3. All ITLB entries are flushed regardless of the virtual address, level, or context. Since the instruction cache is virtually tagged, it must also be flushed in order to ensure full consistency after a page table entry change. The MMU flush operation is described in Section 6.1.2.

3.1.6.2 Data Translation Lookaside Buffer (DTLB)

The DTLB translates the access’s virtual address to a physical address during the M stage of an access. If the DTLB cannot translate the virtual address, the IUFPC pipeline is held, and a translation request is sent to the Table Walker. In addition, any stores to a page that has not been stored to before will cause a translation request for the purpose of setting the Modified bit in the page tables. Any access violations will cause a translation requests so that the Table Walker can update the Fault Status Register and reply with the proper message to return the IUFPC pipeline.

For translation, the TLB-CAM receives virtual address VA[31:18] and the Context ID. For entries marked as Regions: Context ID and VA[31:24] are compared with the VA for that entry; for entries marked as Segments: Context ID and VA[31:18] are compared. If any of the four match (hit), that PTE (Page Table Entry) is selected as the DTLB entry used to translate the access’s virtual address, and to check access permissions. Note that if more than one entry hits (matches), then the result of the translation is unpredictable.

In parallel with the TLB-CAM lookup, the access’s VA[19:12] are used as an index into the direct-mapped 256-entry TLB-RAM. The Context ID and VA[31:20] stored in that entry of the TLB-RAM will be used to compare with the access’s Context ID and VA[31:20] to determine if there was a TLB-RAM hit.

Note that for both the TLB-CAM and TLB-RAM, the Context ID comparison will not be done for entries marked for *supervisor access only*; only the virtual address is compared to determine DTLB hits. If the access happens to be a user access (ASI = 0x08 or ASI = 0x0A) then the translation will fail due to access permissions, described below.

The TLB-CAM hit condition is used to select between TLB-CAM PTE and TLB-RAM PTE, which will be used for further comparisons. For the translation to succeed all of the following must be true:

- If the TLB-RAM was selected, it must be a hit-- Context ID and VA[31:20] match.
- No access permission violations. (Examples of access permission violations are 1. writing a Read-only page, 2. user access to a Supervisor-only page, 3. executing a non-executable page. Note that “page” refers to segments and regions too.)
- The access is not writing to a page/segment/region that is not yet modified (M bit is negated).

If the translation does not succeed, the DTLB will request a translation from the Table Walker. The result of the translation could be a new PTE, or an error to return to the IUFPC. (In the case of a write to an unmodified page, the Table Walk is done to set the M bit, and the result returned to the DTLB would be a new PTE with the M bit set.)

3.2 INSTRUCTION CACHE

The instruction cache is a 16-KByte, direct mapped, virtually indexed and virtually tagged cache organized into 512 lines. Line size of 32 bytes. The cache is 64 bits wide, allowing two sequential instructions to be fed to the pipeline every cycle. On an instruction cache miss, 32 bytes of data are requested from the Bus Interface Unit (BIU), resulting in four 64-bit data transfers. The doubleword that caused the miss is transferred first so that the miss can be resolved, followed by the remaining three doublewords.

3.2.1 Instruction Cache Tag RAM

The instruction cache tag RAM stores virtual address bits VA[31:14] along with a Context_id[7:0] and an access protection bit. The access protection bit indicates that the line was fetched from a VA[31:14] match. The context_id in the tag must match the current context number in the MMU Context Register. In addition, either the Supervisor (S) bit in the PSR is set, or the access protection bit in the cache tags is not set. Since the instruction cache is virtually tagged, the ITLB is only accessed on an instruction cache miss. Therefore, supervisor software must maintain consistency between the instruction cache and the page tables.

On an instruction cache miss, the ITLB sends a memory request along with the appropriate physical address to the BIU. Once the first double word of data is available in the read FIFO of the BIU, it is written into the instruction cache. Once the miss is resolved, the instruction pipeline resumes execution while the rest of the line is being copied from memory into the read FIFO of the BIU. The remainder of the line is then filled into the instruction cache. In the event that a subsequent instruction cache access requests a double word that has not yet been written into the instruction cache, the pipeline is again stalled until the requested double word becomes available.

3.2.2 Instruction Cache Flush

The instruction cache can be flushed using one of two methods.

1. A store alternate instruction to ASI 0x36 causes all valid bits in the instruction cache tag RAM to be invalidated, regardless of whether the instruction cache is enabled. This operation is typically used during instruction cache initialization.
2. The FLUSH instruction, and a store to ASI 0x10:14 or 0x18:1C clears the valid bit of the line that is accessed by VA[13:5], regardless of the tag entry or context. Accessing ASI values 0x10:14 causes both the instruction and data caches to be flushed. Accessing ASI values 0x18:1C causes only the instruction cache to be flushed. For more information refer to Sections 6.1.12 and 6.1.13.

3.2.3 Instruction Caching Restrictions

Instructions are not cached under the following conditions:

- The page from which the instruction was fetched has been declared non-cacheable by clearing the C bit of the Page Table Entry.
- The MMU control register MMU enable (EN) bit has been cleared (written as 0).
- The MMU control register instruction cache enable (IE) bit has been cleared (written as 0).

3.2.4 Instruction Cache Snooping

When instruction cache snooping is enabled, all instruction cache line fill requests are first checked to see if the requested line is present in the data cache. If found, the data cache line is flushed prior to the data being sent to the instruction cache. Instruction cache snooping is enabled by setting bit 2 in the MMU control register (MMUCR).

3.3 DATA CACHE

The data cache is a 16-Kbyte, direct mapped, virtually indexed & physically tagged cache organized as 512 lines. Each line is 64-bits wide. Line size is 32 bytes. The data cache is physically tagged to help maintain coherency between the primary and secondary caches and avoid any memory aliasing between processes. The data cache requires that the virtual memory reference address, available in the E-stage of the pipeline, be translated in order to detect a cache-tag hit or miss. The data cache is accessed in the M-stage of the pipeline. Both the Page-TLB and Segment/Region-TLB are read in parallel during this stage. The page-table-entry is selected from the TLB which contains the valid address. If neither TLB contains the translated address, the pipeline is stalled and the data cache issues a translation request to the table-walker, which is discussed in Section 3.1.4. The cache can be configured in one of two ways.

1. Write-through with Write-allocate.
2. Writeback.

3.3.1 Write Back Mode

In write back mode, only the data cache is written and the corresponding bit in the dirty-bit RAM is set. No external bus cycle is generated. When a dirty line is to be replaced, the data cache requests a writeback operation to the bus interface and the dirty line is written into the writeback buffer.

3.3.1.1 Read Cycles in Write-Back Mode

On a data cache read in write-back mode, the data cache tag RAM is accessed in parallel with the address translation. Virtual address bits VA[13:5] are used to index one of 512 lines in the cache. The tag information is comprised on PA[31:12] and a Valid bit. If the physical address of the tag does not match the translated address, or if there is a match but the valid bit is not set, the pipeline is stalled.

In parallel with the address translation and tag match operations, the doubleword of data indexed by VA[13:3] is read out from the RAM and transferred to the pipeline as well as the FPU, in case there is a dependent FPU operation. If the translated address and the data cache tag match, indicating a data cache hit, the following sequence of events occurs:

1. A linefill is requested, using the physical address of the doubleword that contains the missed data.
2. Any linefill currently in progress is completed.
3. Any posted writes are written to the data cache RAM.
4. If the indexed cache line is dirty, meaning that the line indexed by VA[13:5] has been written to, but not yet written out to the secondary cache, the entire line (four doublewords) are written to the FIFO.
5. When the requested data arrives from either the secondary cache or main memory, it is written into the data cache, the data cache tag RAM is updated, and the dirty-bit is cleared. At the same time the first doubleword of data is written to the cache, it is routed to the pipeline and the pipeline is restarted.
6. A request is generated for to transfer the dirty data in the FIFO out to memory.
7. The linefill continues in the background until it is completed.

3.3.1.2 Write Cycles in Write Back Mode

In write-back mode, write data cannot be written into the data cache until it is determined that the address data indeed resides in the cache. Otherwise data could be lost. Instead, writes are *posted* to a buffer. The actual write to the cache occurs after the address translation has completed and there is an address match.

During the M-stage of the pipeline, the following sequence of events occurs:

1. Address translation is performed to determine if the data cache tag address matches the translated virtual address. Virtual address VA[13:5] is used to index the data cache tag RAM. The physical address is retrieved from the tag RAM and compared to the virtual address generated on VA[13:5]. In the case of a tag miss, an external bus cycle is generated and the processor follows the sequence listed in Section 3.3.1.1.
2. Any previously posted write cycles are written to the data cache. If no posted write exists, but a previous linefill is in progress, the linefill data is written into the data cache.
3. The write data is posted.

3.3.2 Write Through Mode

In write through mode, any write operation updates both the data cache and secondary cache. If a secondary cache exists, data from a primary cache write which is written to the secondary cache may or may not also be written to main memory depending on the data transfer protocol of the secondary cache. If the secondary cache is write-through, data will also be written to main memory. If the secondary cache is configured as write-back, the data is not written to main memory. In write-through mode the primary cache is always a line-for-line subset of the secondary cache. If no secondary cache exists, primary cache writes are always written out to main memory. The write through mode is only intended for aiding in booting generic operating systems and has not been optimized for high performance.

Read cycles are identical to those used in write-back mode, where dirty data is written out to main memory during the line fill operation. Write cycles which hit in the cache cause the pipeline to stall until the data is written to the primary cache and a main memory write transaction is requested. In the case of a cache miss, a write transaction to main memory is requested and the data in the primary cache is not affected. In write-through mode, write cycles to the primary cache always generate a memory write, hence the dirty bit in the primary cache is never set.

3.3.3 Non-Cacheable Operations

During the M-stage of a read or write operation, the Page- and Segment/Region-TLB's, collectively known as the DTLB, are accessed, and the pipeline is stalled. In case of a DTLB miss, a translate request is made to the tablewalker. On a read, the pipeline is restarted as soon as the data becomes available. On a write, the pipeline is restarted when the memory as begun its write operation, or the on-chip SBus controller has accepted a write transaction, which is just before the write cycle begins on the SBus.

3.3.4 Data Cache Tags

The data cache tag RAM stores Physical Address bits PA[35:12] along with a valid bit (V). The DTLB (defined in Section 3.3.3) stores the translated physical address PA[35:12], the context_id[7:0], access protection bits (ACC), a cacheable bit (C), a modified bit (M), and virtual address bits VA[31:20]. On any read or write the DTLB checks to see whether a particular virtual address has already been translated, whether the access protection is valid, and whether the context ID and VA[31:20] fields match. A tag hit occurs on the data cache tags only when physical address PA[35:12] in the data cache tag RAM is matched with the corresponding DTLB entry, the context ID has matched, and the appropriate access protection has been validated.

On a data cache miss and DTLB miss, the data cache requests that the translation be fetched from memory through the Bus Interface unit (BIU). In the case of a data cache miss and DTLB hit, the data cache uses the physical address from the data cache tag RAM to request a line fill through the BIU. Once the first double word of data is available in the read FIFO of the BIU, it is written into the data cache. The instruction pipeline resumes execution while the rest of the line is being copied from memory into the read FIFO. The remainder of the line is filled into the data cache in the background. In the event that a subsequent read or write access requests a double word that has not yet been written into the data cache, the pipeline is stalled until the requested double word becomes available.

3.3.5 Data Cache Flush

The data cache can be flushed using one of two methods.

1. A store alternate instruction to ASI 0x37 causes all valid bits in the data cache to be invalidated, regardless of whether the data cache is enabled. Dirty lines are not written out to memory before being invalidated. This operation is typically used during data cache initialization.
2. The FLUSH instruction, and a store to ASI 0x10:14 clears the valid bit of the line that is accessed by VA[13:5], regardless of the tag entry or context. Execution of this ASI value flushes both the instruction and data caches. Refer to Section 6.1.12 for more information.

3.3.6 Data Caching Restrictions

Data cannot be cached under the following conditions:

- The page from which the data was fetched has been declared non-cacheable by clearing the C bit of the Page Table Entry.
- The MMU control register MMU enable (EN) bit has been cleared (written as zero).
- The MMU control register data cache enable (DE) bit has been cleared (written as zero).

3.3.7 Data Cache Snooping

Data cache snooping is used on Direct-Virtual-Memory-Accesses (DVMA). The Bus Interface Unit (BIU) of the TurboSPARC processor contains a mechanism for detecting if DVMA packet is present in the primary data cache. If the requested DVMA packet is found to be in the data cache, the pending DVMA transaction is suspended until the data cache flush is completed. In primary cache write-back mode, a flush operation causes those lines whose dirty bit is set to be written out to memory before the line is invalidated. This is true for both read and write cycles. The write-back operation is completed before the DVMA transaction can occur.

3.3.7.1 Snoop Hit

The Bus Interface Unit (BIU) of the TurboSPARC microprocessor contains a 512 x 19-bit Snoop SRAM, which stores a copy of the data cache tags. During a cache linefill request, the snoop controller of the BIU updates the snoop RAM with the following 18-bit address and valid bit; {PA[29:12], V-flag}.

The 9-bit index to the Snoop RAM is composed of 7-bits of physical address and a 2-bit virtual address index; {VA[13:12], PA[11:5]}. The 2-bit virtual address index originates from the data memory reference issued by the IUFPC discussed in Section 1.1. The index is provided solely for the purpose of updating the SNoop RAM during a cache linefill. Since the DVMA transaction posted to the BIU contains only a physical address (PA[29:0]), the snoop control logic must perform four Snoop RAM lookups, thus going through all permutations of VA[13:12] in order to find a possible snoop hit.

3.3.7.2 Snoop Flush

Once a snoop hit has been detected, the VA[13:12] corresponding to the location which hit in the Snoop RAM, is concatenated with PA[11:5] to form the virtual address VA[13:5]. The snoop control logic then generates a data cache flush request. If the accessed data cache line is dirty, a write-back is issued prior to the data cache control logic acknowledging the flush operation to the BIU. Hence a write-back takes precedence over a DVMA transaction. In the case of a snoop miss, the DVMA transaction is passed onto the main BIU controller.

3.3.7.3 Snoop Enable

During normal operation such as data cache linefills, the Snoop RAM is always updated regardless of the state of the Snoop-Enable bit (SNP) in the CPU configuration register. However, the Snoop RAM is not accessed on a DVMA transaction unless the SNP bit is set. Refer to Section 6.2.6 for more information on programming the SNP bit.

Chapter 4

Hardware Interface

The TurboSPARC microprocessor contains a sophisticated hardware interface to enhance I/O performance, simplify system design, and reduce system component count. All of the main hardware interface control functions are performed on-chip. The TurboSPARC contains secondary cache, DRAM, SBus, and AFX graphics bus controllers. Each controller is programmed using various control and configuration registers. The following sections discuss each of these hardware interfaces.

4.1 SECONDARY CACHE INTERFACE

The TurboSPARC microprocessor supports 256 KByte, 512 KByte, and 1 MByte secondary cache configurations. The timing and loading characteristics of the secondary cache controller are optimized for use with the Fujitsu 32K x 36 SRAM, part number: MB82VP036. The secondary cache is implemented as write-through and has a 32 byte line size. In order to enhance performance, secondary cache accesses are done in parallel with main memory accesses. If the requested data is found to be present in the secondary cache, the main memory access is aborted. If the requested data is not found in the secondary cache, the main memory access is already in progress. Data is retrieved from main memory and placed in the secondary cache. On a secondary cache ‘miss’, four 64-bit doublewords are transferred from main memory to fill a 32-byte secondary cache line. Data is transferred to the primary cache in parallel with the secondary cache line fill.

4.1.1 Secondary Cache Control

The CPU configuration register (CCR), located at virtual address VA[0000_06xx], contains general secondary cache control and configuration information. Specific fields within the CCR indicate whether a secondary cache is present, and if so, what size it is. The secondary cache size is encoded onto the EC_CONFIG[2:0] pins which are driven by external logic during power-up. The state of these pins is written into the 3-bit SCC field of the CCR and can be modified by software. Refer to Section 6.2.6 for more information on the CPU configuration register.

4.1.2 Secondary Cache Coherency Protocol

The secondary cache coherency protocol implemented in the TurboSPARC processor requires that all external memory cycles update the secondary cache tag RAM. A non-block write purges the indexed tag-entry, while a 32-byte block-write updates the tag entry. All non-block reads bypass the secondary cache, while block reads result in a secondary cache lookup. If the requested data is not found in the secondary cache, the corresponding data is retrieved from main memory and written into the cache as soon as it becomes available. The secondary cache tag is then updated.

If a parity error is encountered during either of the first two double-words of a secondary cache block read, the access is aborted and treated as a cache miss. The secondary cache line which incurred the parity error is then updated by performing an access to main memory and retrieving the data. However, any parity error encountered during the access to the remaining two double-words immediately results in a parity error being issued.

A parity error on a memory block read, assuming there was a secondary cache miss, purges the indexed secondary cache line.

4.1.3 Secondary Cache Tags

The cache tag for the secondary cache is embedded within the secondary cache SRAMs. Each cache entry is 72-bits wide. 64-bits are used for data, 2-bits for data parity, and the remaining 6-bits are used to form one-half of a 12-bit tag. A secondary cache line is 32-bytes in size and is comprised of four 64-bit doublewords. The upper 6-bits of each 72-bit entry contain one-half of the 12-bit secondary cache tag, hence two successive reads are required to form one secondary cache tag. Figure 4-1. shows the format of a secondary cache line. Physical address bits

PA[4:3] are used to decode each entry of a given cache line.

	71 66	65	64 33	32	31 0
Entry 0	TAG0	PAR0[0]	DATA0[63:32]	PAR0[1]	DATA0[31:0]
Entry 1	TAG1	PAR1[0]	DATA1[63:32]	PAR1[1]	DATA1[31:0]
Entry 2	TAG0	PAR2[0]	DATA2[63:32]	PAR2[1]	DATA2[31:0]
Entry 3	TAG1	PAR3[0]	DATA3[63:32]	PAR3[1]	DATA3[31:0]

Figure 4-1. Format of a Secondary Cache LIne

As shown in Figure 4-1., ‘Tag0’ comprises the lower 6-bits of a secondary cache tag, while ‘Tag1’ comprises the upper 6-bits. The format of a 12-bit secondary cache tag is shown in Figure 4-2.

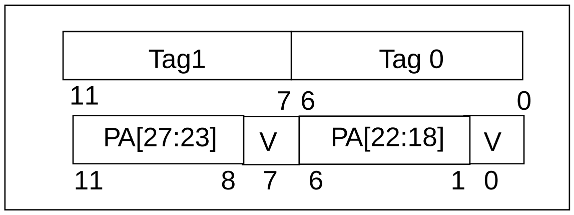


Figure 4-2. Format of a Secondary Cache Tag

V	Cache Tag Valid Bits - Bit [7] indicates if the Cache Tag corresponding to PA[27:23] contains valid data. Bit [0] indicates if the Cache Tag corresponding to PA[22:18] contains valid data.
PA[27:18]	Cache Tag Address field - The upper tag address PA[27:23] and the lower tag address PA[22:18] are concatenated to form the physical tag address. This field remains the same regardless of the cache size.

4.1.4 Secondary Cache Configurations

The TurboSPARC CPU supports secondary cache sizes of 256KB, 512KB, and 1MB. The cache can be implemented using either 32Kx36, 64Kx18, or 128x9 SRAMs. All SRAMs share a common set of address, data & control signals, with the exception of CS[3:0], which are used to access each one of four 64-bit banks depending on the secondary cache size. Figure 4-3. shows a diagram of a 1 MByte secondary cache.

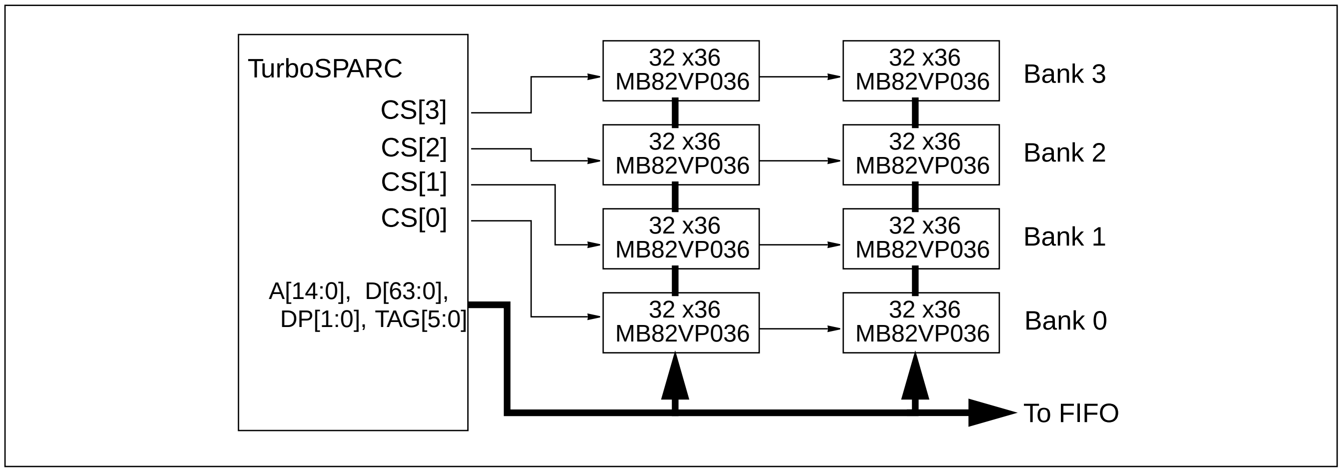


Figure 4-3. One MByte Secondary Cache Configuration

Implementing a 1 MByte secondary cache requires eight 32K x 36 SRAM devices and uses of all four chip selects CS[3:0]. Table 4.1 shows how Figure 4-3. could be modified to accommodate other secondary cache configurations.

Table 4-1. Secondary Cache configuration

SCache Size	Chip Selects Used	Number of Devices	Banks Required
256 KByte	CS[0]	2	0
512 KByte	CS[1:0]	4	0, 1
1 MByte	CS[3:0]	8	0, 1, 2, 3

4.2 DRAM INTERFACE

The TurboSPARC processor provides an on-chip DRAM controller for use with standard page-mode DRAM’s. The controller supports up to 256 MBytes of main memory. A main memory access is initiated on every primary cache miss, regardless of whether or not a secondary cache is present. If no secondary cache is present in the system, all primary cache misses are resolved through main memory. If a secondary cache is present, both a secondary cache and a main memory access are initiated on a primary cache miss. Initiating a main memory access before it is known whether the data exists in the secondary cache helps to reduce memory latency in the case of a secondary cache ‘miss’. Much of the access time to main memory can be hidden during the time it takes to determine whether the data is in the secondary cache. If the data is found in the secondary cache, the main memory access is simply aborted.

4.2.1 DRAM Programming

The majority of DRAM control and configuration information can be found in one of two registers. The MMU Control Register (CR), located at virtual address VA[0000_00xx], controls such parameters as the type of page-mode used in the DRAM interface, enable and control of the parity bits, and the DRAM refresh rate. Refer to Section 6.2.1 for more information on the MMU control register.

The CPU Configuration Register (CCR), located at virtual address VA[0000_06xx] contains fields which control the number of DRAM row address hold cycles and the number of wait states implemented in the memory array. Refer to Section 6.2.6 for more information on the CPU configuration register.

4.2.2 DRAM Configuration with Secondary Cache

In a memory system which implements a secondary cache, the DRAM data lines are connected to the CPU through an IDT162701FIFO read/write buffer. The FIFO is used to buffer all four doublewords of a block write, thus reducing bus usage by freeing-up the bus for secondary cache read operations. Up to eight banks of DRAM may be connected. Banks may be any combination of 8 MByte(10x10 matrix) and 32 MByte (12x10 or 11x11 matrix) DRAM modules. Care should be taken when using 8 MByte DRAM modules, as the top two address bits are unused. This results in the 8 MBytes of DRAM appearing four times in the 32 MByte address space of a given bank.

Figure 4-4. shows how the main memory can be configured with a secondary cache. This example shows 256 KBytes of secondary cache. The FIFO helps reduce CPU data bus usage during write cycles. All four doublewords of the write are transferred into the FIFO, as opposed to waiting for each doubleword to be written into memory before another can be transferred. This allows a secondary cache access to occur immediately following the write of the fourth doubleword into the FIFO, even though the data has not actually been written to main memory. Contention of the address bus during these back-to-back cycles is not an issue as the TurboSPARC microprocessor contains dedicated secondary cache and main memory address buses. An expanded view of DRAM array connections is shown in Figure 4-4..

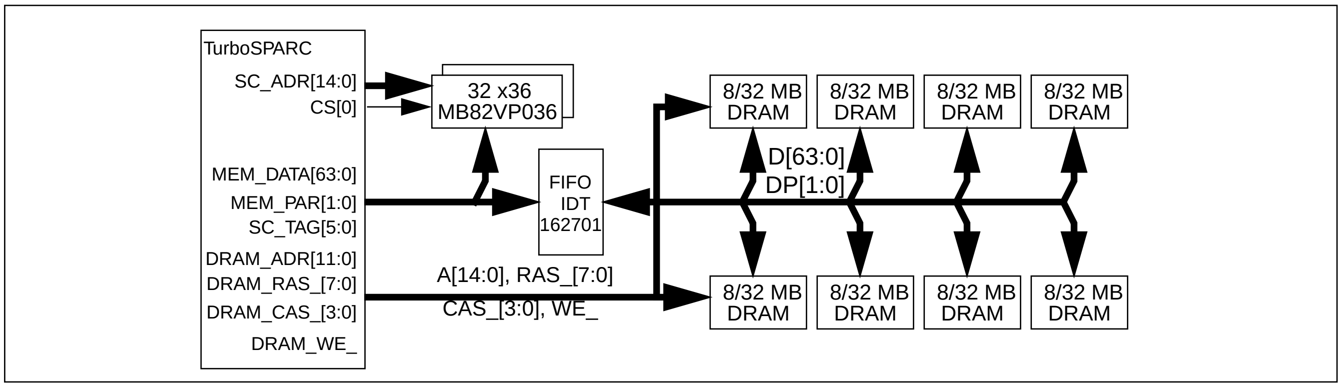


Figure 4-4. DRAM Configuration With Secondary Cache

4.2.3 DRAM Configuration without Secondary Cache

In a memory system where a secondary cache is not present, the DRAM data lines are connected directly to the CPU. Up to eight banks of DRAM may be connected. Banks may be any combination of 8 MByte(10x10 matrix) and 32 MByte (12x10 or 11x11 matrix) DRAM modules. Care should be taken when using 8 MByte DRAM modules, as the top two address bits are unused. This results in the 8 MBytes of DRAM appearing four times in the 32 MByte address space of a given bank. Figure 4-5. shows how the main memory can be configured without an external cache present. The FIFO is not necessary in this example since all primary cache misses require main memory accesses in order to resolve the miss. An expanded view of DRAM array connections is shown in Figure 4-5.

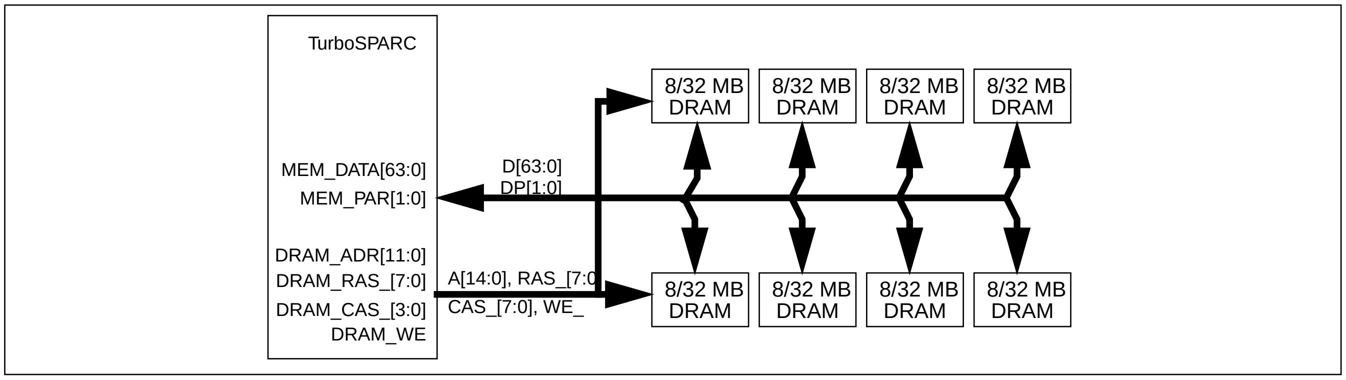


Figure 4-5.DRAM Configuration Without Secondary Cache

4.2.4 DRAM Refresh

The refresh rate of the DRAM is selected by writing to the RFR field in the MMU Control Register. The refresh rate is set to a default value according to the value on the **IOC_RANGE[1:0]** pins during power-up and reset. The refresh rate can be re-programmed by writing a new value to the RFR field. Refer to Section 6.2.1 for more information on the MMU control register.

4.2.5 Partitioning of DRAM Banks

The CPU provides eight row address selects (RAS_), four column address selects (CAS_), and one write enable (WE_), supporting up to eight DRAM banks. Each DRAM bank is connected to one of the RAS_ selects. The DRAM banks can be broken into four even and four odd banks. The odd and even banks refer are accessed using the RAS_[7:0] select lines. Only one even and one odd bank may be active (RAS_ asserted) at any time. Each DRAM bank is divided into 2 x 32-bit data plus 1-bit parity columns. The selection of these columns are done using CAS_[3:0]. The WE_ signal is used to indicate a write operation to the currently selected DRAM bank. Figure 4-6. illustrates the wiring of the eight DRAM banks.

4.2.6 Addressing of DRAM's

RAS_[7:0] and CAS_[3:0] are decoded from the PA[27:25] as follows;

PA[27:25]	RAS Select	CAS Select
3'b000	RAS0_	CAS0_, CAS1_
3'b001	RAS1_	CAS2_, CAS3_
...		
3'b110	RAS6_	CAS0_, CAS1_
3'b111	RAS7_	CAS2_, CAS3_

The DRAM row and column address's are sourced from the PA[24:3] as follows:

DRAM ADDR[11:0]	**PA**
Row_addr	PA[24:13]
Column_addr	{BM[0], PA[24], PA[12:3]}

The row and column multiplexing of the PA will support DRAM's built from 10x10, 11x11 and 12x10 matrices. **BM[0]** is a byte mask indicator used by an AFX unit. Note that TurboSPARC's dram addressing differs from microSPARC-II, which also supports a 9x9 DRAM matrix.

The TurboSPARC DRAM controller makes extensive use of the DRAM Fast Page Mode. During a cache linefill or copyback, the RAS_ signal for the accessed bank remains asserted throughout the given transaction (4 64-bit reads or writes). By enabling CR:PMC (MMUControlRegister:PageModeControl. VA=32’h0, ASI=8’h04), the RAS_ signal for the just accessed bank remains asserted until a subsequent access to another memory bank or row, or due to a DRAM refresh operation. Fast Page Mode between DRAM operations (i.e. linefills or copybacks) is only used if the subsequent access, to a DRAM bank which already has RAS asserted, has a match of the current PA[24:13] equal the prior access’s PA[24:13].

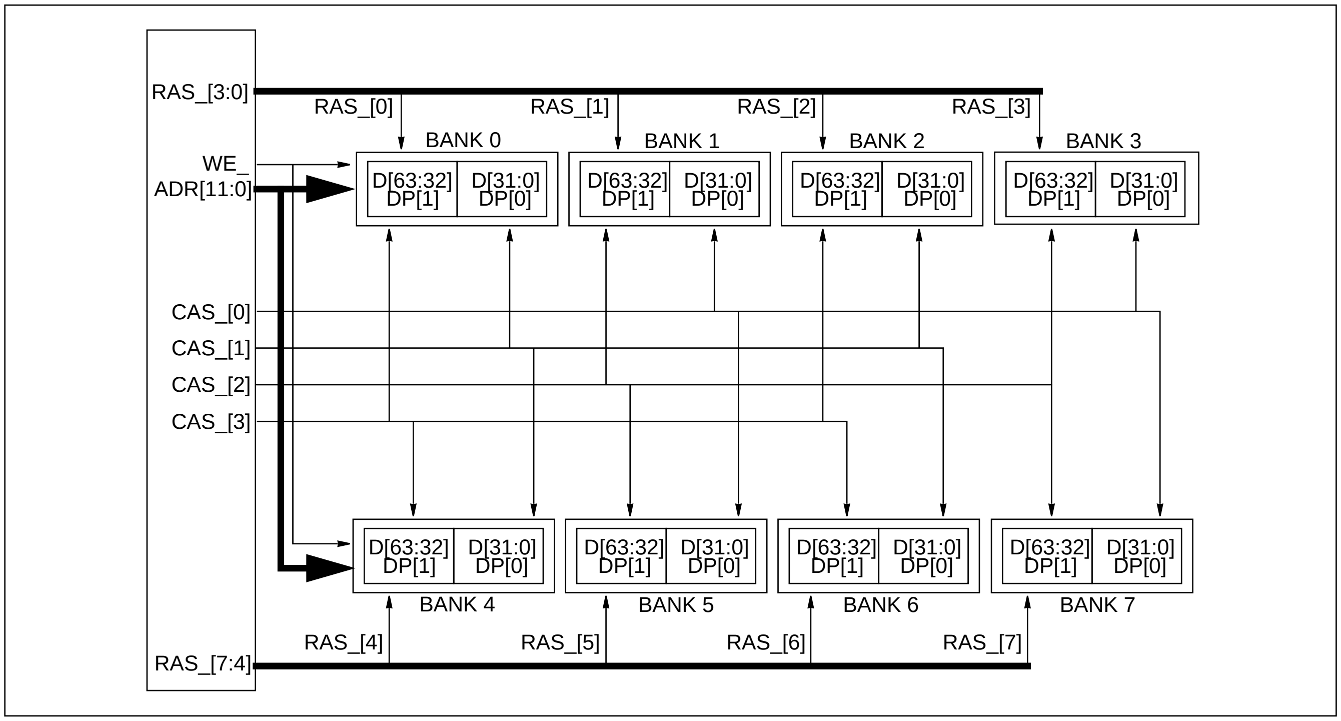


Figure 4-6. Memory Array Signal Connections

4.2.7 Programming DRAM Wait States

DRAM wait state timing is controlled by two fields in the CPU configuration register. The 2-bit Row Address Hold (RAH) field controls the row address hold delay, while the 3-bit Wait State (WS) field controls the number of DRAM wait states. The RAH field is preset to 00 on power-up. The lower 2-bits of the WS field defaults to the values on the IOC_RANGE[1:0] pins at power-up. The upper most bit is always zero. Both values can be changed at any time.

4.3 SBUS INTERFACE

The TurboSPARC CPU provides access to I/O devices through the SBus. The SBus controller is integrated within the TurboSPARC microprocessor and is responsible for initiating each SBus cycle. The SBus controller provides the following features.

- SBus master clock between 16 and 25MHz.
- 32 bit SBus protocol
- A 32-bit virtual address translation capability for other SBus masters.
- A 28-bit physical address per SBus slave.
- Supports up to 6 SBus masters.
- Supports 5 separate slave select signals.
- Supports data transfers of 1, 2, 4, 8, 16, 32 and 64 bytes.
- Does not support extended data transfers, data parity, and SBus Interrupts.
- Completely synchronous operation.

4.3.1 SBus Transactions

The TurboSPARC supports the following SBus transactions.

- Programmed Input/Output (PIO) transactions between the CPU and SBus devices.
- Direct Virtual Memory Access (DVMA) transactions between SBus masters and local memory (referred to Local DVMA).
- Direct Virtual Memory Access (DVMA) transactions between SBus masters and other SBus slave devices (referred to Bypass DVMA).

In addition to the on-chip SBus controller, the TurboSPARC processor also provides a dedicated SBus interface (SBC) to handle PIO as well as Local DVMA transactions. The SBC supports dynamic bus-sizing during PIO transactions as well as address wrapping during burst transfers and retry protocol. The SBus Controller performs arbitration on the SBus in addition to general SBus control.

4.3.2 SBus Arbitration

The SBus Controller arbitrates requests from the processor as well as SBus masters. The controller uses the Module Identification Register (MID) located at PA[1000_2000] to obtain SBus configuration information and determine whether SBus devices can arbitrate for the bus. The SBus controller implements a round-robin arbitration scheme starting from BR_[0] and progressing through BR_[5]. Processor requests have the lowest priority.

4.3.3 SBus Clocking

The SBus clock is a derivative of the I/O clock. The ratio between the two clocks depends on the SBC field of the CPU configuration register (CCR). The SBus clock operates at a frequency equivalent to either one-fourth, one-fifth, or one-sixth that of the I/O clock. Refer to the CPU configuration register in Section 6.2.6 for more information.

4.3.4 SBus Error Reporting

Errors generated during SBus transactions are reported via the Asynchronous Fault Status Register (AFSR) located at PA[1000_1000]. The ASFR contains bits which report the various types of errors such as a late error, timeout error, or bus error. Late errors occur exactly two cycles following a non-idle acknowledgment during an SBus slave cycle. The error is ignored if it follows either a retry or error acknowledgment. A valid SBus late error during PIO or DVMA transactions allows the transaction to complete. The AFSR and the Asynchronous Fault Address Register (AFAR) are updated to reflect the error. Late errors occurring during PIO transactions are reported asynchronously to the processor through external interrupt lines. In addition to reporting the type of error, the AFSR also reports the type of transaction which caused the error (read/write), as well as the size of the transfer. Refer to the AFSR in Section 6.3.4 for more information.

4.3.5 Programmable I/O transactions

Programmable I/O (PIO) transactions occur when the processor executes either a load or a store to SBus address space. PIO transactions consist of an SBus slave cycle only. Address translation is done prior to requesting the SBus. The maximum data transfer size for PIO transactions is 64-bits.

On a PIO read, the processor is stalled until data from the slave becomes available. If the transaction results in an error acknowledgment from the SBus slave, or if a timeout error occurs, it is reported synchronously to the processor. A PIO write is posted to the on-chip SBus controller, allowing the processor to continue processing while the actual SBus write transaction is in progress. If the transaction results in an error acknowledgment from the SBus slave, or if a timeout error occurs, it is reported asynchronously to the processor through the external interrupt lines IRL[3:0]. The fault status and address are updated in the Asynchronous Fault Status Register (ASFR) and Asynchronous Fault Address Register (AFAR) respectively.

A retry acknowledgment during a PIO transaction causes the SBus controller to retry the transaction until it is completed. There is no limit to the number of times a transaction can be retried. Bus-sizing is done on PIO transactions when the slave device responds with an acknowledgment which is smaller than the transaction size. Bus-sizing only occurs on non-burst transfers. A retry acknowledgment during bus-sizing causes the SBus controller to restart the entire transaction.

Doubleword reads or writes are done as a burst transfer providing the SBus slave can support burst cycles as defined in the SBus slot configuration register. If the SBus slave does not support burst transfers, then the doubleword transaction is split into two word transactions. During boot-up, doubleword instruction fetches over the SBus are split into two word transactions.

Atomic instructions such as LDSTUB and SWAP to SBus address space are divided into two independent SBus cycles. Although the arbiter considers an atomic transaction as a single cycle, atomicity is maintained by granting the bus to the processor until the entire transaction is completed. A retry acknowledgment during the write cycle of an atomic transaction causes the SBus controller to retry only the write.

4.3.6 DVMA Transactions

Direct Virtual Memory Access (DVMA) transactions consist of a translation cycle followed by a slave cycle. During the translation cycle, the master drives a virtual address onto the data bus. The I/O memory management unit (IOMMU) of the TurboSPARC processor translates this virtual address into a physical address. The target of the slave cycle is determined once the translation cycle completes. If the translation results in an error due to an invalid I/O page table entry, a write-protected page, or an unsupported transfer size, the SBus controller issues an error acknowledgment to the master and terminates the transaction.

If a DVMA transaction results in an error acknowledgment during either a translation or a slave cycle, the fault status and address are updated in the Asynchronous Fault Status Register (ASFR) and Asynchronous Fault Address Register (AFAR) respectively. An SBus timeout can occur only during the slave cycle of a DVMA transaction.

The maximum data transfer size is 32-bytes for Local DVMA transactions, and 64-bytes for Bypass DVMA transactions. DVMA atomic transactions are not supported.

4.3.7 SBus Slot Configurations

The IOMMU register set of the TurboSPARC microprocessor contains five SBus slot configuration registers (SSCR), each housing configuration information for each of the five SBus slots.

The format of these registers is identical. The registers provide information such as the type of transfer, and when the IOMMU can be bypassed. The IOMMU is bypassed when virtual address bits VA[31:30] are both zero. Refer to Section 6.3.6 for more information on the SBus configuration registers.

4.3.8 I/O Memory Management

The I/O memory management unit (IOMMU) of the TurboSPARC processor is part of the SBus controller and contains a 16-entry, fully associative I/O address translation cache (IOTLB) which is used to cache recently used I/O page table entries using a random replacement algorithm. The main function of the IOMMU is to translate those virtual addresses generated by DVMA masters into physical addresses. I/O page table entries (IOPTE) are stored in the I/O page table in main memory. The page table address is generated based on the **RANGE** field in the IOMMU control register. The base address of the I/O page table is defined by the IOMMU Base Address Register (IOBAR). Refer to Section 6.3.1 and 6.3.2 for more information on these registers. The virtual address generated by the DVMA master must be in a valid range, otherwise an SBus error acknowledgment is sent to the DVMA master.

The entire IOTLB can be flushed by writing to physical address 0x1000_0014, whereas single entries can be flushed by writing to physical address 0x1000_0018. A listing of all IOMMU register address spaces is listed in Table 6.22.

The TurboSPARC processor includes an IOMMU bypass mode which allows intelligent SBus masters to do their own address translations. This mode is enabled by setting the Bypass Enable (BE) bit in the appropriate SBus slot configuration register. In order to bypass the IOMMU, the DVMA master must set VA[31:20] to zero. During IOMMU bypass, the virtual address is the same as the physical address. Refer to Section 6.3.6 for more information on the SBus slot configuration registers.

4.4 AFX BUS INTERFACE

The TurboSPARC microprocessor provides support for the Sun Microsystems compatible AFX graphics bus. The AFX bus clock is a derivative of the I/O clock. The **AXClk** field of the CPU configuration register determines the ratio between the two clocks. The AFX clock frequency can be one-half, one-third, one-fourth, or one-fifth that of the I/O clock frequency.

The IOMMU contains three registers dedicated to AFX bus usage. Two AFX queue level registers determine when processor AFX bus reads can be issued. Only bit [0] of this register is used. Processor reads can be issued to the AFX bus whenever this bit is set. The AFX queue status register reflects the AFX bus queue status.

4.5 CLOCK INTERFACES

The clock interface of the TurboSPARC microprocessor is responsible for supplying all internal and external clocks. A single incoming clock is multiplied by the PLL clock multiplication logic and then used to generate the rest of the clocks.

4.5.1 CPU Clock Generation

The input clock is multiplied to form the VCO clock, which is then divided to become the CPU clock. The CPU Clock (CPU_CLK) is always a divide by 2 of the HF_CLK. Table 4-2. summarizes generation of CPU_CLK. EXT_CLK1 is the incoming clock signal which is multiplied to become the HF_CLK. The multiplication factor depends on the state of the DIV2 input pin. The HF_CLK is then divided by two to become the CPU clock.

Table 4-2. CPU Clock Generation

EXT_CLK1	Multiple By	HF_CLK	Divide By	CPU_CLK	IOCLK_DIV2
25-75 MHz	8	200-600 MHz	2	100-300 MHz	1
25-75 MHz	12	300-900 MHz	2	150-450 MHz	0

4.5.2 I/O Clock generation

The IO_CLK is used by all synchronous IO cells and is either a divide-by-4 or divide-by-6 of the HF_CLK. This clock is also sent to the secondary cache. Table 4-3. summarizes generation of IO_CLK.

Table 4-3. I/O Clock Generation

EXT_CLK1	Multiple By	HF_CLK	Divide By	IO_CLK	IOCLK_DIV2
25-75 MHz	8	200-600 MHz	4	50-150 MHz	1
25-75 MHz	12	300-900 MHz	6	50-150 MHz	0

4.5.3 SBus Clock generation

The SBus Clock is generated either as a divide by 4, 5 or 6 of the IO_CLK. Input pins DIV6[1:0] determine the divide ratio. The SBus clock must be within the range of 16-67 to 25 MHz. Table 4.4 summarizes generation of SB_CLK.

Table 4-4. SBus Clock Generation

IO_CLK	DIVISION	SB_CLK	IOCLK_DIV6[1:0]
[66.7-100]	4	[16.6-25.0]	0 x
[100.0-125.0]	5	[20.0-25.0]	1 0
[125.0-150.0]	6	[20.8-25.0]	1 1

4.5.4 AFX Clock generation

The AFX Clock is generated either as a divide by 2, 3, 4 or 5 of the IO_CLK. Input pins IOC_RANGE[1:0] determine the divide ratio. Table 4.5 summarizes generation of AFX_CLK.

Table 4-5. AFX Clock Generation

IO_CLK	AFX_CLK	DIVISION	IOC_RANGE[1:0]
66.7-72.0	33.3-36.0	2	00
66.7-100	22.2-33.3	3	01
100.0-125.0	31.3-41.6	3	01
100.0-125.0	25.0-31.3	4	10
125.0-150.0	25.0-30.3	5	11

4.6 SIGNAL DESCRIPTIONS

Table 4.6 lists the external signals interface of the TurboSPARC microprocessor. Throughout this table, an underscore (_) at the end of a signal name indicates that the signal is active low. Inputs are 5.0V but are 3.3V tolerant.

Table 4-6. Signal Descriptions

Signal	Type	Voltage	Description
MEM_DATA[63:0]	I/O	3.3	64-bit memory data bus for transferring data to and from main memory.
MEM_PAR[1:0]	I/O	3.3	Bidirectional memory data parity pins. Parity is provided on a word basis (for DRAM only).
DRAM_ADDR[11:0]	Out	5.0	DRAM address pins. These memory address pins may require external buffering to provide the necessary drive for the memory array.
DRAM_RAS_[7:0]	Out	5.0	DRAM Row Address Strobe. Eight separate active low RAS signals, buffered externally to provide sufficient drive to connect directly to the DRAMs. Up to eight DRAM banks are supported. Refer to Figure 4.6.
DRAM_CAS_[3:0]	Out	5.0	DRAM Column Address Strobes. Four separate active low CAS signals are provided for word access. These signals may require external to provide sufficient drive to connect directly to the memory array. Two separate banks are supported by the four CAS lines. Refer to Figure 4.6.
DRAM_WE_	Out	5.0	DRAM Write Enable output pin. This active low signal may require external buffering to provide sufficient drive to connect directly to the memory array.
CPU_MODE[2:0]	In	5.0	These bits encode the secondary cache size as well as the number of banks. The information on these pins is loaded into the SCC[2:0] field of the CPU configuration register.
SC_ADDR[16:0]	Out	3.3	Secondary cache address bus. How many bits are used depends on the size of the SRAM device being used. 32K x36 -bit SRAM devices use SC_ADDR[14:0]. 64 K x 18-bit SRAM devices use SC_ADDR[15:0]. 128K x 9-bit SRAM devices use SC_ADDDR[16:0].
SC_CE[3:0]	Out	3.3	Secondary cache bank select signals select one of four possible SRAM bank s. Which bank is selected depends on the state of the SCC field in the CPU configuration register.
SC_ADSC_	Out	3.3	Active low Secondary cache address strobe. Asserted along with SC_ADDR to indicate the start of a secondary cache access.
SC_ADV_	Out	3.3	Active low Secondary cache address Advance. This signal is typically asserted for 3 cycles following the assertion of SC_ADSC_, in order to read or write an entire cache line.
SC_GW_	Out	3.3	Active low Secondary cache global write. This signal is asserted for the duration of a cache line write cycle. All writes to the secondary cache are 64-bits wide.
SC_OE_	Out	3.3	Secondary Cache Output Enable.
SC_TAG[5:0]	I/O	3.3	Secondary cache tag bits. These bits contain 5-bits (one-half) of the address tag, along with a valid bit. When paired with an adjacent cache entry, they compose a complete address tag.
SC_CLK[3:0]	Out	3.3	Secondary cache clock signals. These signals contain four duplicate copies of the same clock in order to minimize the loading on any given signal.

Signal	Type	Voltage	Description
MF_RST_	Out	3.3	Active low memory FIFO reset.
MF_WCE_	Out	3.3	Active low memory FIFO write enable.
MF_RCE_	Out	3.3	Active low memory FIFO read enable.
MF_OEBA_	Out	3.3	Memory FIFO output enable for enabling external cache data onto the CPU data bus.
MF_OEBA_	Out	3.3	Memory FIFO output enable for DRAM data bus.
MF_LE	Out	3.3	Memory FIFO latch enable.
MF_CLK[1:0]	Out	3.3	Memory FIFO clock signals.
AFX_AEN	Out	5.0	AFX bus address enable output. Indicates when valid address information is on the bus.
AFX_SREPLY	Out	5.0	Graphics system reply. Driven by the processor and indicates that the graphics bus has been selected along with the type of access. On write cycles, the data follows in the following cycle. On read cycles, data is driven onto the bus in the same cycle as AFX_PREPLY is asserted.
AFX_PREPLY[1:0]	In	5.0	Graphics bus port reply driven by the graphics bus slave. On writes cycles, assertion of this signal indicates that data has been removed from the write buffer. On read cycles, assertion indicates that read data is available in the read latch.
AFX_ADDR[2:0]	Out	5.0	Graphics bus address bits.
AFX_CLK	Out	5.0	Graphics clock output for the AFX bus. The AFX clock frequency is a derivative of the I/O clock as defined in the CPU configuration register. The frequency can range between 25 and 42 MHz. The clocks may be used in differential form to improve the common mode noise immunity at high clock rates.
SB_ADDR[27:0]	Out	5.0	SBus Address output pins provide the Physical Address to SBus slave devices.
SB_DATA[31:0]	I/O	5.0	Bidirectional SBus data pins. The 32-bit SBus data pins provide SBus support for the CPU, and support DBMA cycle access via the CPU SBus controller.
SB_SEL_[4:0]	I/O	5.0	Output Slave Select pins. A separate active low slave select is driven to each SBus slot. The slave select pins are used in conjunction with the physical address for accessing each SBus device.
SB_SIZE[2:0]	I/O	5.0	Bidirectional SBus transfer Size description pins. These three pins encode the size of the data transfer of the current SBus operation.
SB_RD	I/O	5.0	Bidirectional SBus Read/Write pin. This pin indicates whether the current transfer is a read or a write operation.
SB_CLK[2:0]	Out	5.0	SBus clock frequency ranging from 16.6 MHz to 25 MHz. The actual frequency depends on the input frequency and its ratio to the I/O clock. Refer to Table 4.4.
SB_AS_	Out	5.0	SBus Address Strobe Output pin. This active low signal indicates that a valid address is on the SBus.
SB_ACK_[2:0]	I/O	5.0	SBus transfer acknowledge pins. The bidirectional ACK[2:0] pins encode the status of the current SBus transfer, from the slave.

Signal	Type	Voltage	Description
SB_LERR_	In	5.0	SBus Late data Error input pin. This active low signal is driven by the current SBus slave, and aborts the current SBus transfer.
SB_BR_[5:0]	In	5.0	Active low Bus Request input pins. There is one pin per bus master.
SB_BG_[5:0]	Out	5.0	Active low Bus Grant output pins. There is one pin per bus master.
SB_CR_	Out	5.0	Active low SBus CPU request output. For Debug purposes only.
SB_CG_	Out	5.0	Active low SBus CPU grant output. For Debug purposes only.
IRL[3:0]	In	5.0	The four interrupt request line input pins encode the highest priority interrupt pending. These pins are driven by the external interrupt logic chip directly to the CPU.
CP_STAT_[1:0]	Out	5.0	Active low CP Status output pins. Used to indicate CP interrupt/trap status as follows: 11 - Normal 10 - Level 15 interrupt 01 - Trap occurred, when trap disabled 00 - Reserved
EXT_CLK1	In	5.0	CPU input clock pin. EXT_CLK1 is used to synchronize the phase lock loop (PLL).
EXT_CLK2	In	5.0	CPU input clock pin. EXT_CLK2 is used to eXclusiveOR with EXT_CLK1 to produce the clock in bypass PLL mode.
PLL_BYP_	In	5.0	Selects either Phase Lock Loop (when high) or the clock inputs directly (when low).
IOCLK_DIV2	In	5.0	Determines the divide ratio between the CPU clock and the I/O clock.
IOCLK_DIV6[1:0]	In	5.0	Determines the divide ratio between the I/O clock and the SBus clock.
IOC_RANGE[1:0]	In	5.0	Determines the divide ratio between the I/O clock and the AFX clock.
INPUT_RESET_	In	5.0	Power-up reset input pin. This signal is active low.
STANDBY	In	5.0	Input pin to put the CPU in Standby mode. Apply a low logic level in normal operation.
JTAG_CLK	In	5.0	Test (JTAG) input clock for boundary scan registers.
JTAG_TMS_	In	5.0	Test Mode Select input pin.
JTAG_TDI	In	5.0	Test Data Input pin (JTAG standard).
JTAG_TRST	In	5.0	JTAG Reset pin (JTAG standard).
JTAG_TDO	Out	3.3	Test Data Output pin (JTAG standard).
VDD1/VSS1	In	3.3	Core power/ground.
VDD2/VSS2	In	5.0	Power/ground for 5.0V I/O
VDD3/VSS3	In	3.3	Power/ground for 3.3V I/O.
VDD4/VSS4	In	3.3	Power/ground for PLL.

Chapter 5

Bus Operations

The TurboSPARC microprocessor contains four bus controllers along with four dedicated bus interfaces.

1. Secondary cache bus interface and controller
2. Main memory (DRAM) bus interface and controller
3. SBus interface and controller
4. AFX graphics bus interface and controller

The secondary cache interface is specific to the TurboSPARC microprocessor. The main memory, SBus, and AFX bus interfaces conform to the MicroSPARC II standard. This section discusses some of the common bus transactions between the processor and each of the four interfaces. Throughout this section, an underscore (_) at the end of a signal name indicates that the signal is active low.

5.1 SECONDARY CACHE BUS CYCLES

Timing Diagrams not available at the time of this printing.

5.2 MAIN MEMORY BUS CYCLES

This section discusses five commonly used main memory bus transactions. Each transaction involves a timing diagram showing the interaction between each of the signals. The goal of this section is to show the relationships between the various signals generated by the DRAM controller. Depending on the speed of the memory bus and the access times of the devices used, the number of clocks between the various parameters may change. For example, CAS_ is shown as being asserted three clocks before data is available on a read cycle. If either the speed of the bus or the access times of the device are altered, the number of clocks between CAS_ assertion and data availability can either increase of decrease.

5.2.1 Page Mode Read

Figure 5-1. shows a page mode read cycle. Conforming to the standard DRAM signal assertion protocol, RAS_ is asserted along with the row address, followed by CAS_ along with the column address. WE_ is not asserted at the same time as CAS_, indicating a read cycle. Data is available three clocks after CAS_ is asserted. CAS_ must then be negated for one clock, then re-asserted along with a new column address. This sequence repeats until all four doublewords are retrieved. The four doublewords comprise a 32 byte cache line.

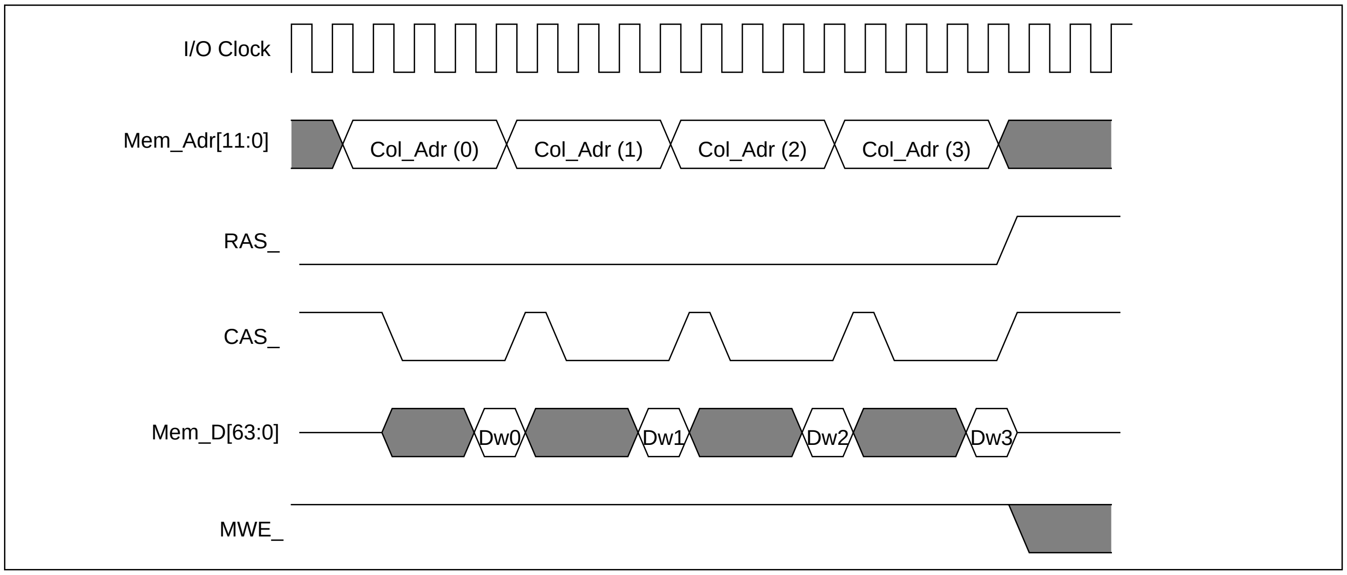


Figure 5-1. Page Mode Read

5.2.2 Page Mode Write Followed by Read

Figure 5-2. shows a write-back followed by a read cycle. Only the first transfer of the read cycle is shown. Memory write data is driven by the processor in the clock following the address. However, before the data can be written to the memory, CAS_ must be asserted and the proper amount of set-up time must elapse. Assertion of the MWE_ signal indicates a write cycle. Since the read cycle is to the same page as the write cycle, RAS_ can remain asserted. Only a new column address must be driven. The read data is driven by the DRAM three clocks after the corresponding CAS_ is asserted.

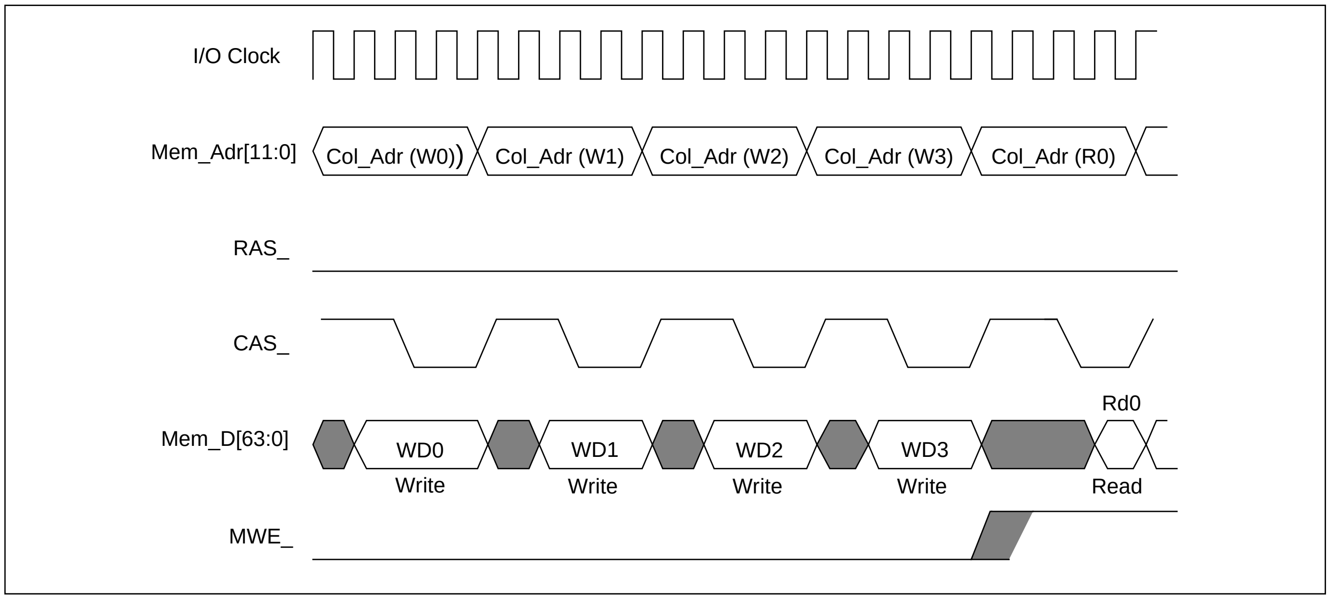


Figure 5-2. Page Mode Write Followed by Read

5.2.3 Page Mode Read-Modify-Write

Figure 5-3. shows the data being retrieved from a given memory location. The data is read into the processor where it is modified by an operation which takes one clock to complete. The modified data is then driven out onto the memory bus. Since the read and corresponding write are to the same location, and hence the same DRAM page, RAS_ can remain asserted throughout the entire operation. The assertion of MWE_ indicates that the data will be written to the DRAM after the appropriate access times have been met. The negation of RAS_ indicates the end of the cycle.

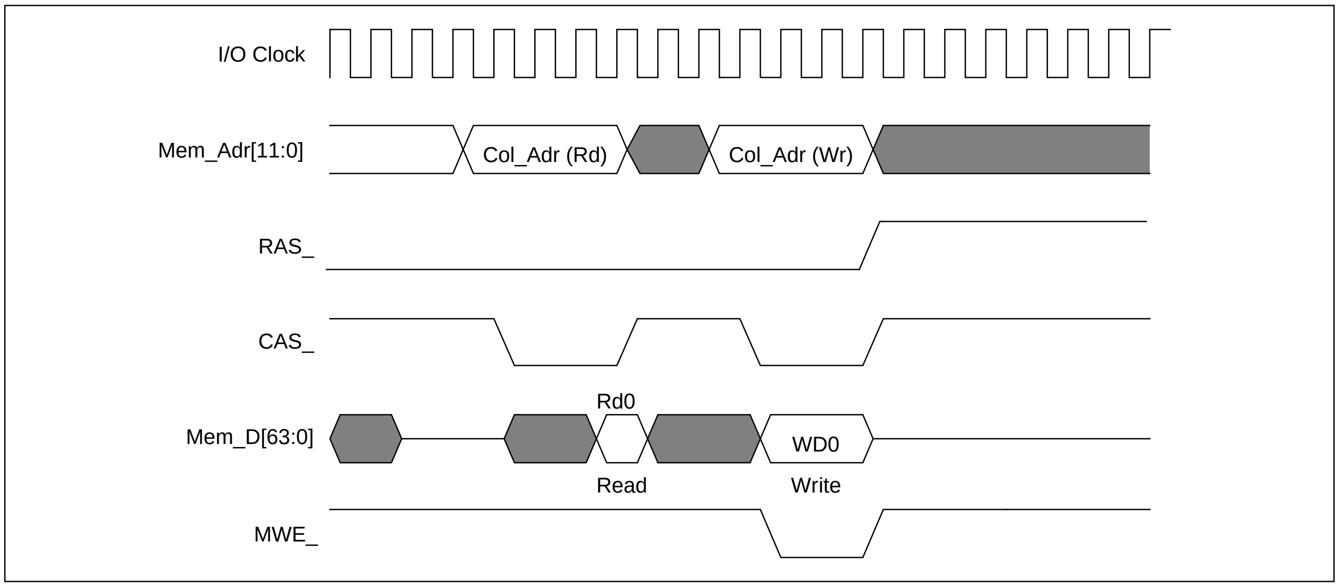


Figure 5-3. Page Mode Read Followed by Write

5.2.4 Non-Page Mode Read Cycles

A memory access is defined as a page-mode access when those memory locations being accessed all reside in the same DRAM row. In page mode accesses the row address select (RAS_) can remain asserted throughout the operation since all access are to the same row. When back-to-back accesses are to different rows, RAS_ must be negated and then re-asserted to indicate the row address of the next access. This sequence is shown in Figure 5-4. The assertion of CAS_ yields read data three clocks later. RAS_ is then negated to indicate the end of that memory access. RAS_ must remain negated for some number of clocks as defined in the DRAM specification, after which it can again be asserted to start a new memory access. The assertion of RAS_ causes the DRAM to latch the row address internally. After the appropriate RAS_ access time has elapsed, the column address can be driven onto the DRAM address bus along with CAS_, which causes the DRAM to latch the column address internally. After CAS_ is asserted, some number of clocks must elapse before data can be driven by the DRAM. The negation of RAS_ again indicates the this particular memory operation has completed. Once RAS_ has been negated, another row address and corresponding RAS_ signal must be asserted in order for the DRAM device to initiate an internal memory cycle.

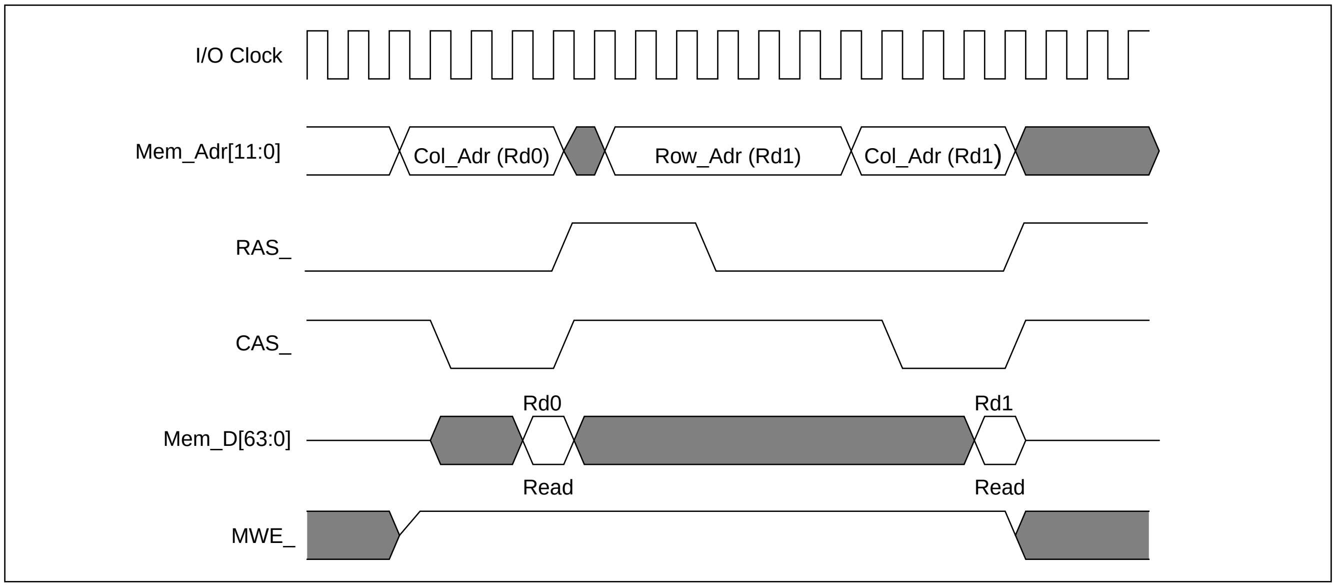


Figure 5-4. Non-Page Mode Back-to-Back Reads

5.2.5 Non-Page Mode Read Followed by Write

The first portion of Figure 5-5. shows a non-page mode read cycle. The row address is asserted along with RAS_, followed by a column address along with CAS_. Data is available three clocks after CAS_ is asserted. Because these accesses are non-page mode, RAS_ must be negated after the read data is driven onto the bus to indicate the end of the cycle. A new row address is driven by the processor along with a corresponding RAS_. Write data is driven by the processor one clock after the address. However, at the time the write data is driven onto the bus the exact location in memory where the data is to be written is not known. Data cannot be written until the column address has been driven, the corresponding CAS_ asserted, and the proper access time parameters have been met. The write data remains valid on the bus a predetermined number of cycles depending on the speed and access times of the DRAM device being used. When this time has elapsed, RAS_ is negated by the processor to indicate that the cycle is complete.

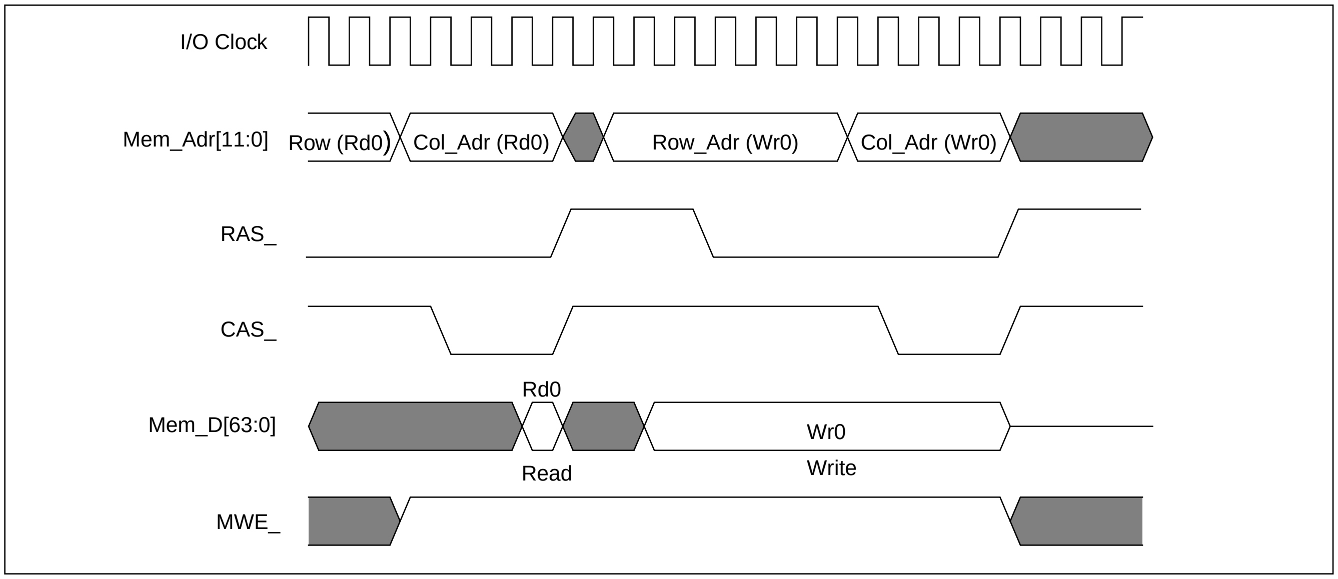


Figure 5-5. Non-Page Mode Read Followed by Write

5.3 SBUS INTERFACE BUS CYCLES

The TurboSPARC provides access to peripheral I/O devices through the SBus. The SBus can be clocked between 16 MHz and 25 MHz. The TurboSPARC processor supports up to six SBus masters. An on-chip Module Identification Register (MID) determines which SBus slots and their corresponding devices can arbitrate for ownership of the SBus. For a listing of the common bus transactions and protocols used on the SBus, refer to the SBus User’s Manual available from Sun Microsystems, part #800-5922-10.

5.4 AFX GRAPHICS INTERFACE BUS CYCLES

The AFX graphics bus provides a dedicated interface between the processor and the graphics subsystem. The AFX graphics bus incorporates a 64-bit data path and 28-bit physical address path per slave. The bus can operate at up to 75 MHz. For a listing of the common bus transactions and protocols used on the SBus, refer to the microSPARC II User’s Manual. Appendix A.

Chapter 6

Programming Interface

6.1 ADDRESS SPACE IDENTIFIERS

Load and Store instructions executed by the TurboSPARC microprocessor includes an 8-bit address space identifier (ASI) field which determines the type of operation to be executed. The 8-bit field allows up to 256 different ASI values. The format of the virtual address changes based on the ASI number attached to the instruction. For example, flush operations, register accesses, accesses to user or supervisor space, and accesses to the caches, all have different ASI values which denote that type of operation. Any ASI access can be either a load or a store instruction and contains a unique virtual address format. Read and write data are transferred on dedicated read and write buses.

Table 6-1. lists the ASI values and the corresponding operation

Table 6-1. ASI Assignments

ASI (hex)	Operation	Type of Access	Access Size
02:00	Unassigned	---	---
03	MMU Probe	Read	Word
03	MMU Flush	Write	Word
04	MMU Registers	Read/Write	Word
05	Instruction TLB Diagnostic	Read/Write	Word
06	Data TLB Diagnostic	Read/Write	Word
07	IOTLB Diagnostic	Read/Write	Word
08	User Instruction Space	Read/Write	All
09	Supervisor Instruction Space	Read/Write	All
0A	User Data Space	Read/Write	All
0B	Supervisor Data Space	Read/Write	All
0C	Instruction Cache Tag Access	Read/Write	Word
0D	Instruction Cache Data Access	Read/Write	Word
0E	Data Cache Tag Access	Read/Write	Word
0F	Data Cache Data Access	Read/Write	Word
14:10	Data Cache Flush	Write Only	Word
17:15	Unassigned	---	---
1C:18	Instruction Cache Flush	Write Only	Word
1F:1D	Unassigned	---	---
20	MMU Pass-Through	Read/Write	All
2F:21	Unassigned	---	---
30	Secondary Cache Diagnostic	Read/Write	Doubleword
31	Snoop RAM	Read/Write	Word
32	Page-Table-Descriptor Diag.	Read/Write	Word
35:33	Unassigned	---	---
36	Instruction Cache Flash Clear	Write Only	Word
37	Data Cache Flash Clear	Write Only	Word
3F:38	Unassigned	---	---

The following subsections define the virtual address and data formats for each ASI value.

6.1.1 MMU Probe (ASI = 0x03, Read)

Execution of an LDA instruction with this ASI value causes a read cycle which returns a page table from main memory. The address where the page table is located is indicated by the **VFPA** field VA[31:12]. The 4-bit **Type** field indicates the type of probe operation as defined in Table 2.4. The remainder of the virtual address field must be zero. All probe operations bypass the Data-TLB and go directly to main memory to fetch the required page-table-entry. The MMU table walker is cleared prior to performing the probe operation. If the probe operation is successful, the PTE is not brought into the Data-TLB regardless of the state of the **Type** field.

There are two types of errors which can occur during a probe operation.

1. An entry with the PTE(ET) not equal to 1(PTD) is encountered before the level being probed is reached.
2. A memory error occurs. On a memory error, the **fault** registers are updated, but no access exception is generated.

Figure 6-1. shows the format of the virtual address during a probe or flush operation.

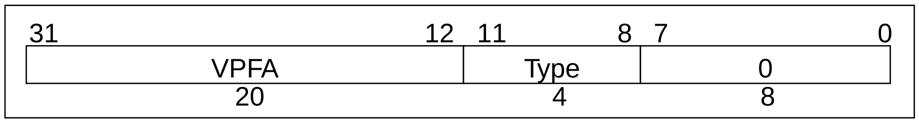


Figure 6-1. Virtual Address Format During an MMU Probe/Flush

- **VPFA** - Virtual flush or probe address.
- **Type** - Type of flush operation (see table below)
- **0** - These bits must be zero

If either of the above errors occurs, the probe operation returns a zero value. If the operation is successful, the page table at the level indicated by the **Type** field is returned. Table 6-2. shows the various transfers which can occur based on the **Type** field. A ‘0’ indicates that a zero value is returned. An ‘x’ indicates that the requested page table entry is returned. A ‘=>’ indicates that the next level page entry is to be examined.

Table 6-2. Values Returned During an MMU Probe Operation

Probe Type	Probe Operation Successful															
	Level 0 Entry				Level 1 Entry				Level 2 Entry				Level 3 Entry			
	pte	res	inv	ptd	pte	res	inv	ptd	pte	res	inv	ptd	pte	res	inv	ptd
0 (Page)	0	0	0	=>	0	0	0	=>	0	0	0	=>	x	0	x	0
1 (segment)	0	0	0	=>	0	0	0	=>	x	0	x	x				
2 (region)	0	0	0	=>	x	0	x	x								
3 (context)	x	0	x	x												
4 (entire)	x	0	0	=>	x	0	0	=>	x	0	0	=>	x	0	0	0
F:5	Undefined															

Probe types [4:0] do not update the Reference (R) bit in the page table entry. Probe types [F:5] are unassigned and when accessed return an undefined value. All probe requests bypass the Data-TLB and go directly to the MMU table-walker. The table-walker is cleared prior to performing the actual probe operation. The accessed PTE is not transferred into the Data-TLB regardless of the probe type.

6.1.2 MMU Flush (ASI = 0x03, Write)

The MMU flush operation, indicated by a write to ASI 03, removes one or more entries from the data and instruction TLB’s and clears the MMU table walker. The table walker is discussed in section 3.1.4. A flush is accomplished by executing a store alternate (STA) instruction to the address designated in the **VFPA** field. The data supplied by the store alternate instruction is ignored. Table 6-3. lists the types of flush operations.

Table 6-3. DTLB Page Entry Flush Field

VA[11:8]	Flush Operation	PTE Flush Match Criteria
0000	Page	PTE pointed to by VA[19:12]
0001	Segment	No match criteria
0010	Region	No match criteria
0011	Context	No match criteria
0100	Entire	No match criteria
1111:0101	Unassigned	------

In a **page** flush operation, indicated as the first entry in Table 6-3., the DTLB indiscriminately flushes the entry in the DTLB RAM indexed by VA[19:12].

For the **segment** or **region**, and **context** flushes, the DTLB indiscriminately flushes both the entire Page-TLB and Segment/Region TLB.

The ITLB flushes all entries in the 4-entry ITLB regardless of the flush operation.

The table walker flushes all intermediate page-table descriptors regardless of the flush operation. The MMU flush operation does not effect the IOTLB, and can be performed regardless of the state of the MMU Enable bit (ME) located in the MMU Control register.

6.1.3 MMU Registers (ASI = 0x04, Read/Write)

The MMU registers contain general status, control, and configuration information. A given register can be accessed by executing a privileged load or store alternate instruction with the appropriate virtual address. The following registers can be accessed through ASI 04.

Table 6-4. MMU Registers

VA[31:0]	Register Accessed
0000_00xx	Control Register (CR)
0000_01xx	Context Table Pointer Register (CTPR)
0000_02xx	Context Register (CXR)
0000_03xx	Synchronous Fault Status Register (SFSR)
0000_04xx	Synchronous Fault Address Register (SFAR)
0000_05xx	Reserved
0000_06xx	CPU Configuration Register (CCR)
0000_07xx: 0000_12xx	Reserved
0000_13xx	Synchronous Fault Status Register (SFSR) read only
0000_14xx	Synchronous Fault Address Register (SFAR) read only

See section 6.2 for descriptions of these registers.

6.1.4 ITLB Accesses (ASI = 0x05, Read/Write)

This address space is used to read or write entries in the instruction TLB (ITLB). The ITLB is a 4-entry, fully associative address translation cache. A read operation returns the TLB entry which matches bits [31:12] of the virtual address. The read returns either the physical page number, or the match information, depending on the state of bit 2 (ld_ppn) of the virtual address.

A write operation replaces the least-recently-used (LRU) TLB entry with the data provided in the format shown in Figure 6-5.. The TLB supports only level-2 and level-3 page table entries. Level-0 and level-1 TLB entries are stored in the TLB, but are matched as if they were level-2 entries.

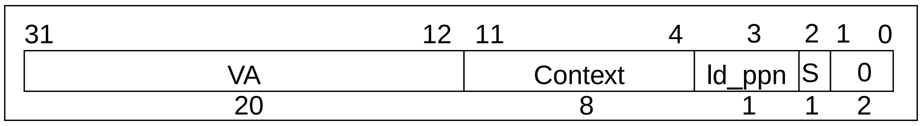


Figure 6-2. Virtual Address Field During ITLB Accesses

VA	Virtual Address [31:12]
Context	Context Number.
ld_ppn	load physical page number. 1 = PPN data, 0 = match data.
S	Used instead of PSR S-bit (Supervisor access) for TLB matching. If this bit is set, the PTE pertains to a supervisor access page only and no context comparison is performed.
0	These bits must be zero.

6.1.4.1 ITLB Read Operation

As shown in Figure 6-2., the **ld_ppn** bit can assume one of two values. The format of the returned data differs depending on the state of this bit. Figure 6-3. and Figure 6-4. show the data format for each value.

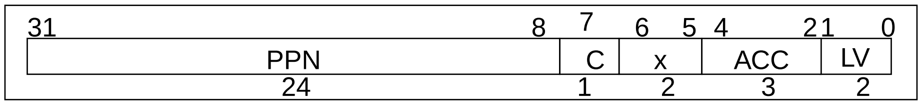


Figure 6-3. Data Format During an ITLB Read (ld_ppn = 1)

PPN	Contains bits [35:12] of the physical page number.
C	Cacheable bit. 1 = Cacheable, 0 = Not Cacheable
x	Reserved. Should be zero.
ACC	Access permission bits as stored in the page table entry. The three-bit ACC field is normally 010, but changes to 110 if the permission is supervisor only.
LV	Level of page table entry.

00, 01 = Region

10 = Segment

11 = Page.

The LV field is normally 10, but changes to 11 during level 3 accesses.

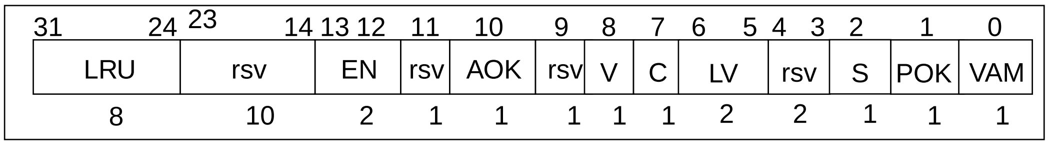


Figure 6-4. Data Format During an ITLB Read (ld_ppn = 0)

LRU	Contains the LRU state machine data used by the ITLB to replace the LRU entry. LRU data is updated only on an ITLB write operation.
EN	Returns ITLB entry number which matched the virtual address.
x	Reserved. Should be zero.
AOK	TLB access OK. 1 = Virtual address match. 0 = no match.
V	TLB entry valid.
C	Cacheable bit. 1 = Cacheable, 0 = Not Cacheable.
LV	Level of matched page table entry. 00 = level 0, 11 = level 3.
POK	Access permission OK. 1 = access to supervisor page.
VAM	Virtual address match in ITLB. All other fields in this format except AOK are invalid if the VAM bit is zero.

6.1.4.2 ITLB Write Operation

Figure 6-5. shows the data format during an ITLB write operation.

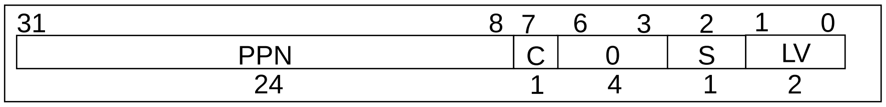


Figure 6-5. Data Format During an ITLB Write

PPN	Bits [35:12] of the physical page number
C	Cacheable bit. 1 = Cacheable, 0 = Not Cacheable.
0	These bits must be zero.
S	Supervisor access bit. Setting this bit indicates that the page accessible only to the supervisor.
LV	Level of page table entry. 00 = level 0, 11 = level 3.

6.1.5 DTLB Accesses (ASI = 0x06, Read/Write)

This address space is used for accesses to the DTLB, which consists of a 256-entry, direct mapped page-TLB, and a 4-entry, fully associative segment/region TLB. The page-TLB maps 4 KByte pages only, while the segment/region TLB maps 256 KByte segments and 16 MByte regions. 4 GByte pages, which normally come directly from the root pointer, are not supported in the TurboSPARC microprocessor. Context maps are treated as region maps.

Reads cycles return the TLB entry which matches the VA portion of the effective address. If no entry matches, the indexed page-TLB is returned. DTLB reads are similar to ITLB reads in that the data format returned depends on the state of the **ld_ppn** bit, represented by virtual address bit VA[5]. A logical 1 on this bit returns the physical page number. A logical 0 returns match information.

Write cycles to level 3 (LVL = 0x3) write into the 256-entry page-TLB. Writes to any other level replace the LRU entry in the 4-entry segment/region-TLB.

Figure 6-6. shows the format of the virtual address during DTLB diagnostic accesses.

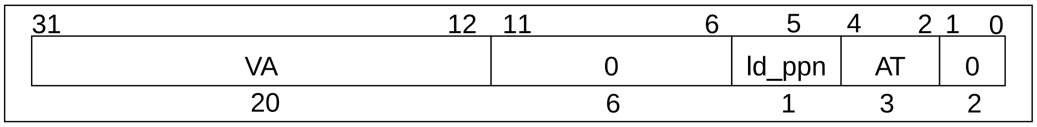


Figure 6-6. Virtual Address During DTLB Read Accesses

VA	Virtual Address [31:12]
ld_ppn	Load physical page number. 1 = PPN data, 0 = match data.
0	These bits must be zero
AT	Access Type. Three bit field denotes one of eight operations.

000	Load from User Data Space
001	Load from Supervisor Data Space
010	Load from User Instruction Space
011	Load from Supervisor Instruction Space

100 Store to User Data Space
101 Store to Supervisor Data Space
110 Store to User Instruction Space
111 Store to Supervisor Instruction Space

6.1.5.1 DTLB Read Operation

As shown in Figure 2.7, the **ld_ppn** bit can assume one of two values. The format of the returned data differs depending on the state of this bit. Figure 6-7. and Figure 6-8. show the data formats for each value.

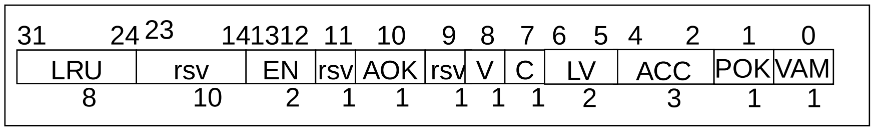
31 8	7	6	5	4 2	1 0
PPN	C	M	rsv	ACC	LV
24	1	1	1	3	2

Figure 6-7. Data Format During a DTLB Read (ld_ppn = 1)

PPN	Physical Page Number. The two highest-order bits are equal to zero. If the virtual address and context ID match any of the entries on the 4-entry segment/region-DTLB, then the PPN returned will be from that entry, with the high-order two bits and the low-order six bits of the PPN equal to zero. Otherwise the PPN is returned for the indexed 256-entry page-DTLB, regardless of whether either the virtual address or current context ID match, or that page entry is valid. The high-order two bits of the PPN are returned as zero.
C	Cacheable bit. 1 = Cacheable, 0 = Not Cacheable. The cacheable bit is set under one of two conditions: the virtual address and context ID match a valid entry in the 4-entry segment/region-DTLB, provided that particular segment or region is marked cachable, or the entry indexed by VA[19:12] in the 256-entry page-TLB has the cacheable bit set. It does not matter whether or not this entry is valid.
M	The modified bit indicates whether the addressed page, segment, or region, has been modified. This bit is set under one of two conditions; the virtual address and current context ID match a valid entry in the 4-entry segment/region-DTLB, and that segment has been modified. If there is no match in the segment/region-DTLB, the entry indexed by VA[19:12] in the 256-entry page-DTLB is returned. It does not matter if the entry is valid or not, or if the virtual address and context ID match.
rsv	Reserved. Software should write these bits as zero.
ACC	The 3-bit ACC field indicates types of permissions (read, write, execute) for user and supervisor accesses. These permissions are defined in the page table entry, which is described in Section 3.1.3. If the virtual address and current context ID are not found in the 4-entry segment/region-DTLB, the ACC field for the indexed entry in the 256-entry page-DTLB is returned, regardless of whether that entry is valid or the context ID matches.
LV	The two-bit LV field indicates the level of page table entry in the segment/region DTLB (00 or 01 = region, 10 = segment), providing the virtual address and context ID match a valid entry. If there is no match, the level of the entry indexed by VA[19:12] in the 256-entry page-DTLB (11 = page) is returned. It does not matter whether or not this entry is valid.

Figure 6-8. shows the data format for a DTLB read with ld_ppn = 0.

.



31 24	23 14	13 12	11	10	9	8	7	6 5	4 2	1	0
LRU	rsv	EN	rsv	AOK	rsv	V	C	LV	ACC	POK	VAM
8	10	2	1	1	1	1	1	2	3	1	1

Figure 6-8. Data Format During a DTLB Read (ld_ppn = 0)

LRU	Contains the LRU state machine data used by the segment/region-DTLB to replace the LRU entry. LRU data is updated only during an ITLB write operation.
EN	Returns segment/region-DTLB entry number accessed when the virtual address matches a segment/region-DTLB entry. The LV and V fields are used to distinguish entry 0 from a no-entry match.
rsv	Reserved. Should be zero.
AOK	TLB access OK. This bit is set under the following conditions: 1. There is a valid entry in the DTLB for the virtual address. 2. The context ID of the entry either matches the current context ID number, or the access is a supervisor access. 3. The read/write and supervisor/user access permissions are allowed as defined in the AT field above. 4. Either the modified bit is set, or the bit is clear, the access specified is a read, and the MMU is enabled.
V	The valid bit is set if either virtual address and context ID match a valid entry in the segment/region DTLB, or if the entry indexed by VA[19:12] in the page-DTLB is valid.
C	The cacheable bit is set if the virtual address and context ID match a valid entry in the segment/region DTLB, and either that segment or region is marked valid, or if the entry indexed by VA[19:12] in the page-DTLB, valid or invalid, has the cacheable bit set.
LV	Indicates the page table entry level of the entry in the DTLB. The LV field assumes a value of 01 for a region map, or 10 for a segment map, if the virtual address and context ID matches a valid entry in the segment/region-DTLB. The LV field assumes a value of 11 otherwise, regardless of whether a valid translation exists in the page-DTLB.
ACC	The 3-bit ACC field indicates types of permissions (read, write, execute) for user and supervisor accesses. These permissions are defined in the page table entry. If the virtual address and current context ID are not found in the 4-entry segment/region-DTLB, the ACC field for the indexed entry in the 256-entry page-DTLB is returned, regardless of whether that entry is valid or the context ID matches.
POK	The POK bit indicates whether the access type specified in the AT field is permitted. If there is a virtual address and context ID match in the segment/region-DTLB, then the POK bit is set if the access type is allowed. If the access type is allowed for the indexed entry in the page-DTLB, the POK bit is set, regardless of whether the virtual address and context ID matches or that entry is valid.
VAM	The virtual address match bit is set if the virtual address and context ID specified match any of the entries in the DTLB providing the entry is valid. The VAM bit is also set if the virtual address matches and the access specified is a supervisor access.

6.1.5.2 DTLB Write Operation

Figure 6-9. shows the virtual address format during a DTLB write operation.

31 12	11 4	3	2 0
VA	CxtID	V	0
20	8	1	3

Figure 6-9. Virtual Address Format During a DTLB Write

VA	The 20-bit VA field contains the virtual address bits field to be translated. For region and segment (levels 1 and 2) entries, VA[31:18] are written into the segment/region-DTLB. Data is stored into the LRU entry and the LRU state is updated. For page (level 3) entries, VA[19:12] gives the index into the page-DTLB. VA[31:20] are written into the page-DTLB.
CtxID	The 8-bit CxtID field is used to match the context ID during address translation. For region or segment entries, the context ID is written into the segment/region-DTLB. For page entries, the context ID is written into the page-DTLB.
V	Valid bit. If this bit is set (1), the entry is written as valid. If clear (0), the entry is invalid.
0	These bits must be zero.

Figure 6-10. shows the data format for a DTLB write operation.

31 8	7	6	5	4 2	1 0
PPN	C	M	0	ACC	LV
24	1	1	1	3	2

Figure 6-10. Data Format During a DTLB Write

PPN	The PPN field contains physical address bits PA[35:12]. For segment (level 1) and region (level 2) entries write data bits [29:4] which are transferred on PA[33:18], are written into the segment/ region-DTLB. On page (level 3) accesses write data bits [29:8] are transferred on PA[33:12] and are stored in the page-DTLB. For all write operations, data bits [31:30] should be zero.
C	Cacheable bit. If set, the segment, region, or page to be written is cacheable.
ACC	The 3-bit access permission field indicates the access permissions of the corresponding Region, Segment, or Page. Different ASI values are used during User and Supervisor accesses. The ACC field is part of the page table entry and is encoded as shown in Table 3.4.
M	Indicates the state of the modified bit for the appropriate segment, region, or page. Indicates the access permissions for the appropriate segment, region, or page, as defined in the page table entry.
LV	The 2-bit LV field indicates the level of the entry. 00 or 01 = Region, 10 = segment, 11 = page. Each entry is written into the appropriate DTLB.

6.1.6 IOTLB Accesses (ASI = 0x07, Read/Write)

The IOTLB is a 16-entry, fully associative TLB which implements a random replacement algorithm. The IOTLB is used to translate addresses during accesses to the SBus. The IOTLB is used to cache the most-recently-used IO page table entries. Any entry in the IOTLB can be accessed either explicitly or by matching the virtual address. Explicit accesses use the 4-bit **LOC** field to determine the exact entry. Figure 6-11. shows the format of the virtual address during IOTLB accesses.

31 12	11 8	3	6 0
VA	LOC	Type	0
20	4	1	7

Figure 6-11. Virtual Address Format During an IOTLB Access

VA	This 20-bit field contains the virtual address to be translated.
LOC	This 4-bit field allows access to any of the 16 IOTLB entries.The LOC field is used when accessing an explicit entry as opposed to a matching entry.
Type	The type bit is set if the matching entry is accessed. It is cleared if the entry based on the LOC field is accessed.
0	These bits must be zero.

6.1.6.1 IOTLB Read Operation

As described above, the state of the **Type** bit indicates whether the access is explicit based on the **LOC** field, or whether the data is returned from the location which matched the virtual address field. The data format changes slightly depending on the type of access. If the **Type** bit in Figure 2.12 is clear, the access is explicit using the **LOC** field. In addition, the **ENT** field can be ignored since the exact location in the TLB is already known from the **LOC** field. If the **Type** bit is set, the access is to the location based on the virtual address field. If there is a match, data bit [31] is set and the physical page number returned. If bit [31] is clear, the rest of the data is ignored. Figure 6-12. shows the data format during an IOTLB read operation.

31	30 12	11	10	9 8	7 4	3 0
0/VAM	PPN[18:0]	W	V	0	ENT	0
1	19	1	1	2	4	4

Figure 6-12. Data Format During an IOTLB Read Operation

0/VAM	This bit defaults to zero during explicit accesses. During matched accesses, the VAM bit is set if the virtual address matched in the IOTLB. VAM is clear if there is no match, indicating that the returned data should be ignored.
W	This bit indicates a write access.
V	Indicates whether the given page table entry is valid.
ENT	This 4-bit field indicates the accessed location in the IOTLB.
0	These bits must be zero.

6.1.6.2 IOTLB Write Operation

During an IOTLB write operation, virtual address bits VA[31:12] are written into the CAM portion of the IOTLB, and the store data is written into the RAM portion of the IOTLB. The CAM portion of the TLB houses the virtual address, and the RAM portion of the TLB houses the physical address. Which location is written depends on the state of the **Type** bit, as described in Figure 6-11.. If the bit is clear, the write occurs to the location indicated by the **LOC** field. If the bit is set, the write occurs at the location indicated by the random replacement counter.Figure 6-13. shows the data format during an IOTLB write operation.

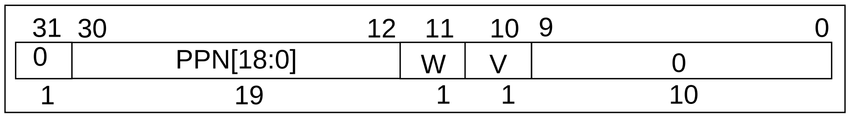


Figure 6-13. Data Format During an IOTLB Write Operation

PPN	Contains the physical page number matching the virtual address.
W	This bit indicates a write access.
V	Indicates whether the given page table entry is valid.
0	These bits are ignored and must be zero.

6.1.7 User/Supervisor Access (ASI = 0x0B:0x08, Read/Write)

The SPARC architecture defines four virtual address spaces dedicated to User and Supervisor accesses. Supervisor instructions and data for a given process, as well as user instructions and data, are accessed using these ASI values. Table 6-5. shows how each ASI value is used.

Table 6-5. User/Supervisor Accesses

ASI	Operation
0x08	User Instruction
0x09	Supervisor Instruction
0x0A	User Data
0x0B	Supervisor Data

6.1.8 Instruction Cache Tag Access (ASI = 0x0C, Read/Write)

The Instruction cache tag RAM is a direct-mapped, virtually indexed, 512-entry RAM. Figure 6-14. shows the virtual address format during an instruction cache tag RAM access.

31 … 5	4 3	2	1 0
VA[31:5]	0	S	0
27	2	1	2

Figure 6-14. Virtual Address Format During Icache Tag Accesses

VA	The virtual address is divided into two portions. VA[13:5] are used to index the tag RAM. VA[31:14] are then used to compare the tag data. On a write operation, VA[31:14] are stored into the tag RAM.
S	Indicates supervisor access. During a write operation the S bit is written into the tag RAM, indicating that the line is available only in supervisor mode. If the S bit is set during read operation, the access is being done in supervisor mode.
0	These bits are ignored and must be zero.

6.1.8.1 Instruction Cache Tag Read

The data format returned during an instruction cache tag RAM read operation is shown in Figure 6-15. Since the instruction cache in the TurboSPARC microprocessor is virtually indexed and virtually tagged, a virtual address is returned as opposed to a physical address.

31 … 14	13 … 11	10	9	8	7 … 0
VA[31:14]	0	TOK	S	V	Context
18	3	1	1	1	8

Figure 6-15. Data Format During an Icache Tag Read

VA	The virtual address field.
TOK	Indicates that the line in the tag RAM is valid, the access permissions are valid, and either the context ID matches, or the S bit is set (1).
S	Indicates the status of the supervisor bit as stored in the instruction cache tag RAM.
V	Indicates the status of the valid bit as stored in the instruction cache tag RAM.

6.1.8.2 Instruction Cache Tag Write

Figure 6-16. shows the data format during an instruction cache tag RAM write operation. During a write, VA[31:14] and the 8-bit context ID number are stored into the tag RAM.

31 … 9	8	7 … 0
0	V	Context
23	1	8

Figure 6-16. Data Format During an Icache Tag Write

V	Indicates the status of the valid bit to be written into the instruction cache tag RAM.
Context	Indicates the context ID number to be written into the instruction cache tag RAM.

6.1.9 Instruction Cache Data Access (ASI = 0x0D, Read/Write)

This ASI value is used during instruction cache data accesses. The instruction cache is a direct-mapped, 16 KByte cache which contains 2048 indexed locations. Instruction cache data can be written or read directly using the load-alternate (LDA) and store-alternate (STA) instructions. The correct ASI value is embedded within these instructions. Read and write accesses are word-only. The TurboSPARC microprocessor does not support 64-bit cache accesses, hence the load-double-alternate (LDDA) and store-double-alternate (STDA) instructions are not supported and the results are undefined. Figure 6-17. shows the virtual address format during an instruction cache data access.

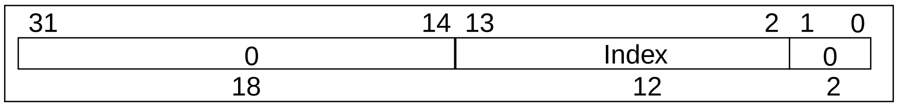


Figure 6-17. Virtual Address Format During an Icache Data Access

Index	Indicates one of 2048 entries in the instruction cache.

6.1.10 Data Cache Tag Access (ASI = 0x0E, Read/Write)

The data cache tag RAM in the TurboSPARC microprocessor contains 512 entries. Each entry consists of a physical address and a valid bit. Physical address bits PA[35:32] are not implemented. The data cache is virtually indexed and physically tagged. Virtual address bits VA[13:5] are used to index the tag RAM. Physical address bits PA[31:12] are compared to PA[31:12] from the DTLB and combined with the valid bit in order to determine whether data cache tag hit or miss occurred. The virtual address formats differ between the read and write operations.

6.1.10.1 Data Cache Tag RAM Read Operation

Figure 6-18. shows the virtual address format for a data cache tag read

.

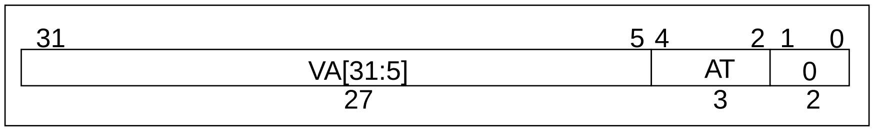


Figure 6-18. Virtual Address Format During a Data Cache Tag Read

VA	VA[13:5] are used to index the data cache tag RAM. VA[31:14] are used to for compare the physical address in the DTLB.
AT	This 3-bit access type field indicates the type of access to be performed. The eight access types are listed in Section 6.1.5, DTLB Accesses.

Figure 6-19. shows the data format during a data cache tag read.

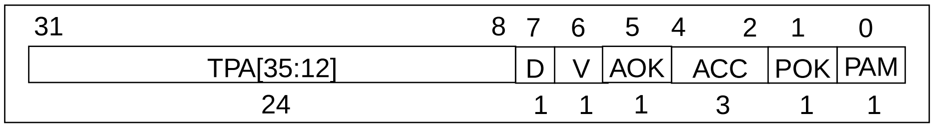


Figure 6-19. Data Format During a Data Cache Tag Read

TPA	Tag Physical address bits PA[35:12]. The two highest order bits [31:30], which represent PA [35:34], are always zero.
D	When set, the dirty bit indicates that the corresponding data cache line has been modified. This line must be written out to memory the next time it is updated.
V	The valid bit is set to indicate that the cache tag is valid.
AOK	TLB access OK. This bit is set under the following conditions: 1. There is a valid entry in the DTLB for the virtual address. 2. The context ID of the entry either matches the current context ID number, or the access is a supervisor access. 3. The read/write and supervisor/user access permissions are allowed as defined in the AT field in Section 6.1.5. 4. Either the modified bit is set, or the bit is clear, the access specified is a read, and the MMU is enabled.
ACC	This three bit field indicates types of permissions (read, write, execute) for user and supervisor accesses. These permissions are defined in the page table entry. If the virtual address and current context ID are not found in the 4-entry segment/region-DTLB, the ACC field for the indexed entry in the 256-entry page-DTLB is returned, regardless of whether that entry is valid or the context ID matches.
POK	This bit indicates whether the access type specified in the AT field is permitted. If there is a virtual address and context ID match in the segment/region-DTLB, then the POK bit is set if the access type is allowed. If the access type is allowed for the indexed entry in the page-DTLB, the POK bit is set, regardless of whether the virtual address and context ID matches or that entry is valid.
PAM	The physical address match bit is set if the physical address from the cache tag (TPA) matches the virtual-to-physical translation in the DTLB for that virtual address. In addition to an address match, the cache tag valid bit must be set (V), and the data cache must be enabled.

6.1.10.2 Data Cache Tag RAM Write

As mentioned in the previous section, the virtual address format is different between the data cache tag read and the data cache tag write. Figure 6-20. shows the virtual address format for a data cache tag write operation.

.

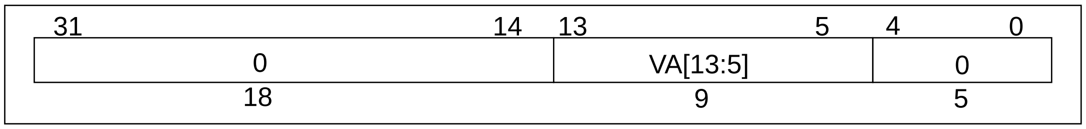


Figure 6-20. Virtual Address Format During a Data Cache Tag Write

VA	Virtual address bits VA[13:5]. These bits represent the index into the cache tag array.
0	These bits must be zero.

Figure 6-21. shows the data format during a data cache tag write.

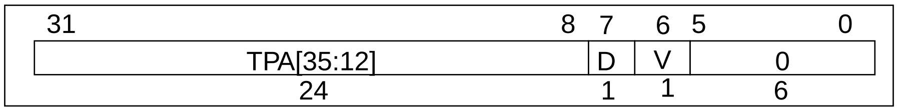


Figure 6-21. Data Format During a Data Cache Tag Write

TPA	Physical address bits PA[35:12]. The two highest order bits [31:30], which represent PA [35:34], are always zero.
D	When set, the dirty bit indicates that the corresponding data cache line has been modified. This line must be written out to memory the next time it is updated.
V	The valid bit is set when the TPA and D fields are valid.

6.1.11 Data Cache Data Access (ASI = 0x0F, Read/Write)

The data cache of the TurboSPARC microprocessor is 16 KBytes in size and consists of 2048 64-bit entries. The data cache is written or read using the LDA and STA instructions. The virtual address format is the same for both read and write operations. Data is written and read in 64-bit quantities. Figure 6-22. shows the virtual address format during a data cache data access.

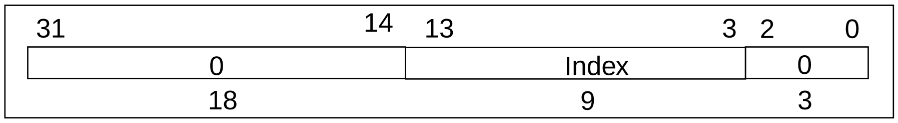


Figure 6-22. Virtual Address Format During a Data Cache Data Access

Index	Indexes one of 2048 lines in the data cache.

6.1.12 Instruction/Data Cache Flush Operation (ASI = 0x14:0x10, Write Only)

This block of virtual address spaces are used to flush single cache lines in both the instruction and data caches. A cache line flush is initiated by a single STA instruction and results in a single line being removed from both the instruction and data caches. Instruction cache lines are invalidated regardless of the status of the line. Data cache lines are written out before being invalidated based on the state of the dirty (D) bit. In order to flush all 512 lines in each cache, 512 STA instructions must be executed using all bit combinations of the 9-bit CLVA field. Figure 6-23. shows the virtual address format for an instruction/data cache flush operation.

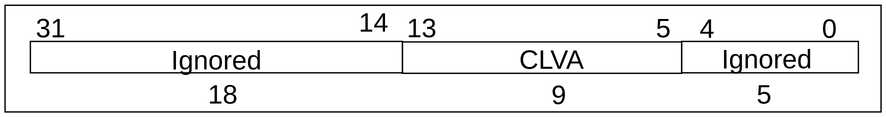


Figure 6-23. Virtual Address Format During an I/D Cache Flush

CLVA	Cache line virtual address. Used to index to one of 512 cache lines during a flush operation.

6.1.13 Instruction Cache Line Flush Operation (ASI = 0x1C:0x18, Write Only)

This block of virtual address spaces are used to flush single cache lines in the instruction cache. The data cache is not affected by this operation. A cache line flush is initiated by a single STA instruction and results in a single line being removed from the instruction cache. Instruction cache lines are invalidated regardless of the status of the line. The virtual address format for an instruction cache line flush operation is identical to the format shown in Figure 6-23..

6.1.14 MMU Physical Address Pass-Through (ASI = 0x20, Read/Write)

Execution of a privileged STA or LDA instruction causes the virtual address [VA31:0] to be passed directly onto the physical address pins [PA30:0]. The MMU pass-through operation allows the processor to gain access to non-MMU mapped addresses.

6.1.15 Secondary Cache Accesses (ASI = 0x30, Read/Write)

Execution of a STA or LDA instruction to ASI 0x30 allows access to both the secondary cache tag and data RAM's. All accesses to the secondary cache are doubleword. A write to this address causes both the data and tags to be written. Reading from this address fetches either the secondary cache tags or data, depending on the state of the virtual address bit [VA31]. If VA[31]=0, a doubleword read fetches 64-bits of secondary cache data. If VA[31]=1, an 8-bit read is read is performed on the secondary cache tags. A double-word write causes 64-bits of data to be written to the data cache. VA[29:24] are used to write the a 5-bit cache tag along with a valid bit. Table 6.6 shows the secondary cache read and write operations.

Table 6-6. Accessing the Secondary Cache

Operation	VA[31]	Type of transfer
Read	0	64-bit SC data read => D[63:0]
Read	1	8-bit SC Tag Read: (PH,PL,V,tag4:0) => D[7:0]
Write	----	D[63:0] => 64-bit SC Data Write
		VA[29:24] => (V,tag[4:0])
Legend:		V = Valid Bit, PH = High word parity bit, PL = Low word parity bit, VA = Virtual address, SC = Secondary Cache

6.1.16 Snoop Ram Access (ASI = 0x31 Read/Write)

The Snoop RAM is organized as a 512-entry by 19-bit RAM. The upper 18-bits [18:1] contain the physical tag, and the lowest-order bit [0] indicates a valid entry when set. When debugging the TurboSPARC processor, the Snoop RAM is updated on any data cache linefill, regardless of the state of the snoop enable (**SNP)** bit in the CPU Configuration register. Refer to Section 6.2.6 for more information on this register.

During a Snoop RAM access the virtual address is formatted as shown in Table 6-7.

Table 6-7. Virtual Address Format During a Snoop RAM Access

Virtual Address	Usage
VA[31:30]	Not Used
VA[29:12]	Snoop TAG[18:1]
VA[11:5]	Snoop ADDR[6:0]
VA[4]	Snoop TAG[0]
VA[3:2]	Snoop ADDR[8:7]
VA[1:0]	Must be 00 to avoid word alignment trap

During writes to the Snoop RAM, the TAG[18:1] and ADDR[8:0] fields are derived from the virtual address, while the actual write data is ignored. During reads of the Snoop RAM, the virtual address supplies the ADDR[8:0] field as well as the tag reference address VA[29:12] which is used to compare against the content of the Snoop RAM. The data format of a Snoop RAM read is shown in Table 6-7.

Table 6-8. Data Format During a Snoop RAM Read

Virtual Address	Usage
DA[31:14]	Snoop TAG[18:1]
DA[13:5]	Snoop INDEX[8:0] = (VA[3:2],[11:5])
DA[4]	Snoop TAG[0] = (Valid Bit)
DA[3]	Tag Hit (VA[29:12] = TAG[18:1]
DA[2:0]	000

6.1.17 Page Table Descriptor Accesses (ASI = 0x32, Read/Write)

This ASI value allows access to cached Page Table Descriptors (PTD). The MMU table-walker contains a cache of four segment-level PTD’s. A PTD entry consists of both a virtual address and a physical address field. The virtual address is used by the compare logic to determine if the address is contained in the table. If there is a match, the physical address portion of the PTP entry is used to point to an entry in the next table. A PTD entry consists of the following two fields:

VA[31:18]	Virtual address field used for address ‘hit’ comparison.
PA[27:4]	Physical base address of the level 3 page table for the given virtual address match.

A write to this ASI value writes a new PTD entry into the least-recently-used (LRU) PTD cache. The virtual address field is applied to VA[31:18] of the PTD entry, while the store data is written into the physical address field. The LRU state is updated at the completion of the operation. Read cycles result in a PTD cache lookup compare. The virtual address field **VA[31:18]** is compared to the four entries in the PTD cache, and the physical address **PA[27:4]** corresponding to the hit is returned on **D[27:4]**.

6.1.18 Instruction Cache Flash Clear (ASI = 0x36, Write Only)

Execution of an STA instruction to this ASI value clears all valid bits in the instruction cache tag RAM regardless of whether the instruction cache is enabled. This operation can be used during instruction cache initialization.

6.1.19 Data Cache Flash Clear (ASI = 0x37, Write Only)

Execution of a STA instruction to this ASI value clears all valid bits in the data cache regardless of whether the data cache is enabled. Dirty lines are not written out to memory prior to being invalidated. This operation can be used during data cache initialization.

6.2 MMU REGISTERS

The MMU register set consists of eight registers. Each register resides at a dedicated virtual address space as defined in Table 6-9.

Table 6-9. MMU Registers Locations

VA[31:0]	Register
0000_00xx	Control Register (CR)
0000_01xx	Context Table Pointer Register (CTPR)
0000_02xx	Context Register (CXR)
0000_03xx	Synchronous Fault Status Register with Clear (SFSR)
0000_04xx	Synchronous Fault Address Register (SFAR)
0000_06xx	CPU Configuration Register (CCR)
0000_13xx	Synchronous Fault Status Register with Read and Clear (SFSR)
0000_14xx	Synchronous Fault Address Register (SFAR)

6.2.1 Control Register (VA = 0x0000_00xx)

The control register contains general mode, status, and configuration information. As with each MMU register, the control register resides at a specific virtual address. The control register is accessed by LDA and STA instructions to the appropriate virtual address. Figure 6-24. shows the format of the control register.

31 28	27 24	23 21	20 19	18	17	16	15	14	13 10	9	8	7	6 3	2	1	0
IMPL	VER	rsv	PMC	PE	PC	rsv	rsv	BM	RFR	IE	DE	PSO	rsv	ICS	NF	ME
4	4	3	2	1	1	1	1	1	4	1	1	1	5		1	1

Figure 6-24. MMU Control Register

IMPL	The 4-bit Implementation field identifies the Fujitsu implementation of the MMU. This bit is hard-wired and returns 0x00 when read. This field is read only.
VER	The 4-bit Version field identifies the Fujitsu MMU version. This bit is hard-wired and returns 0x05 when read. This field is read only.
RSV	Bits [23:21], [16], [15], and [6:2] are reserved. These bits return a zero value when read. Software must assure that these bits are zero when the register is written.
PMC	The 3-bit page mode control field allows for speculative assertion of the RAS_ signal in order to enhance memory performance. Setting this field to 0x3 causes the on-chip DRAM controller to keep RAS_ asserted speculatively at the end of any main memory cycle, in case the follow-on memory access is to the same DRAM page. Setting this field to 0x0, which is the default condition entered on power-up, causes RAS_ to be negated at the end of each main memory cycle. This bit is read/write.
PE	When the parity enable bit is set (1), the two parity bits are checked on all memory read cycles. There is one parity bit per word of data. This bit is read/write.
PC	The parity control bit controls whether even or odd parity is used. When the bit is set, odd parity is used. When the bit is clear, even parity is used. This bit is read/write and has meaning only when the PE bit is set. This bit is read/write.
BM	The boot mode bit is set on both power-on-reset and during IU error mode reset. When this bit is set, all instruction fetches are done from PA[30:27] = 0x7. Setting the BM bit has meaning only if the ME bit is clear. This bit is read/write.
RFR	The 4-bit refresh rate field indicates the DRAM refresh rate. The RFR divisor is clocked by the IO-CLK and always defaults to a binary value of 0000 during reset, regardless of the state of the IOC_RANGE[1:0] input pins. The typical minimum refresh rate is 15.6 uS per period. Table 6-10. shows the refresh rates. This field is read/write.

Table 6-10. DRAM Refresh Rates

RFR	Divisor
0000	780
0001	no-refresh
0010	1170
0011	1560
0100	1950

IE	The instruction cache enable bit controls the activation of the primary instruction cache. When set, the instruction cache is enabled. All lines in the cache must be invalidated before this IE bit is set. The IE bit is cleared on power-up and the instruction cache should not be enabled until after the MMU has been initialized. This bit is read/write.
DE	The data cache enable bit controls the activation of the primary data cache. When set, the data cache is enabled. All lines in the cache must be invalidated before this DE bit is set. The DE bit is cleared on power-up and the data cache should not be enabled until after the MMU has been initialized. This bit is read/write
PSO	The Partial Store Order bit is hard-wired and returns a zero value when read, indicating that total store ordering is being used. Partial store ordering is not supported. This bit is read only.

ICS	Instruction Cache Snoop Enable. When set, enables snooping of the instruction cache.
NF	When the No-Fault bit is set, all data faults signalled to the pipeline are suppressed. However, the synchronous fault status and address registers (SFSR, SFAR) are updated to indicate the fault which occurred. The reporting of instruction faults is not affected by the state of the NF bit. This bit is read/write.
ME	The state of the MMU Enable bit determines whether instruction and data accesses will be translated. The MMU Enable bit is cleared during reset and must be set after the MMU is initialized in order for the IE and DE bits to have any affect. This bit is read/write.

6.2.2 Context Table Pointer Register (VA = 0000_001x)

The Context Table Pointer Register (CTPR) is a read/write register which points to the context table in main memory. The address of the context table is determined by the 26-bit CTP field. The table is then indexed by the 8-bit context field of the context register. The 26-bit CTP field is driven onto PA[30:10] when this level of table-walking is required. Figure 6-25. shows the format of the Context Table Pointer Register.

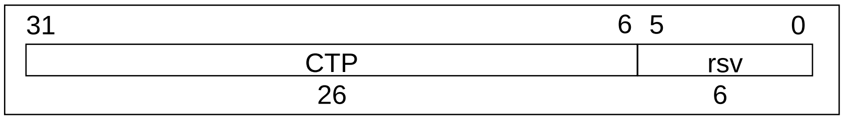


Figure 6-25. Context Table Pointer Register

6.2.3 Contest Register (VA = 0x0000_002x)

The 8-bit context field is used in conjunction with the context table pointer field in Figure 2.30 to access the context table in main memory. Figure 6-26. shows the format of the context register.

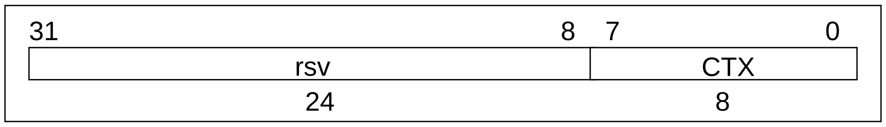


Figure 6-26. Context Register

6.2.4 Synchronous Fault Status Register (VA = 0x0000_03xx/13xx)

The Synchronous Fault Status Register provides information on exceptions (faults) issued by the MMU. Since the CPU is pipelined, several faults may occur before a trap is taken. The faults are grouped into three classes:

- Instruction access faults
- Data access faults
- Translation table access faults.

If a subsequent instruction access fault is detected by the CPU, the MMU writes the status of the latest fault into the Synchronous Fault Address Register (SFAR), and sets the **OW** bit to indicate that the previous fault status has been lost. Table 6-11. shows the addresses at which the SFAR can be accessed, and the corresponding operation.

Table 6-11. Accessing the Synchronous Fault Status Register

VA[15:8]	Read/Write	Operation
0x03	Read	Read and clear SFSR
0x03	Write	No effect
0x13	Read	Read SFSR
0x13	Write	Write SFSR

MMU page table access which cause an external system error generate a translation table access fault. If a translation table access fault overwrites a previous instruction or data access fault, the **OW** bit is cleared. An instruction or data access fault does not overwrite a translation table access fault. Figure 6-27. shows the format of the SFSR.

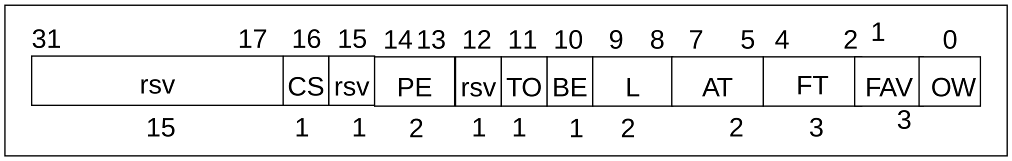


Figure 6-27. Synchronous Fault Status Register

rsv	The rsv fields, comprised of bits [31:17], [15], and [12], are hard-wired and return a zero value when read. Writes to the SFSR do not affect these bits.
CS	The Control Space error bit is set when an access is made to an unassigned ASI space. The FAV bit is set, and the OW bit is cleared regardless of any prior error status. Table 6-12. lists those ASI values which cause the CS bit to be set. The bit is set on either a read or a write access to any of these values.

Table 6-12. Unassigned ASI Values

ASI value range	
From	**To**
0x00	0x02
0x15	0x17
0x1d	0X1F
0X21	0X2F
0X33	0X35
0X37	0XFF

PE The 2-bit Parity Error field indicates the data word on which a memory parity error occurred during a memory read. Non-cacheable byte/halfword/word memory writes can also result in a parity error since a memory read is performed prior to performing the write. PE[14] indicates a parity error for D[63:32], while PE[13] is for D[31:0]. The PE field is always set in conjunction with the AT, FT & OW fields. The SFAR does not contain the fault address for a parity error, thus the FAV is not set for parity errors. However, the address of a parity error can be found in the Memory Fault Status Register (MFSR) and Memory Fault Address Register (MFAR).

TO The TimeOut error bit is set when an SBus timeout occurs on a processor initiated SBus transaction. A timeout will occur if a given (or non existing) slave does not respond within 256 SBus cycles.

BE The SBus Error bit is set if an SBus error acknowledgment or Late error occurred on a processor initiated SBus transaction.

L The 2-bit MMU page current Level field is set to the page table level of the entry which caused the fault. If an external bus error is encountered while fetching a PTE or PTP, the L field records the page level of the page table containing the entry. Table 6-13. shows the encoding of the L field.

Table 6-13. Page Table Entry Levels

L[9:8]	PTE Level
0	Entry in Context Table
1	1
2	2
3	3

AT The 3-bit Access Type field defines the type of access which caused the fault. Loads and stores to user/supervisor instruction space can be caused by load/store alternative instructions with ASI values of 08 or 09. The encoding of the AT field is shown in Section 6.1.5.

FT The 3-bit Fault Type field defines the current fault type. Table 6-14. shows the encoding of the FT field.

Table 6-14. Fault Type Field Encoding

FT	Fault Type
0	None
1	Invalid Address Error
2	Protection Error
3	Privilege Violation Error
4	Translation Error
5	Access Bus Error
6	Internal Error
7	Reserved

An **Invalid Address** error occurs when a PTE is found where ET = 0 (invalid).

A **Protection** or **Privilege Violation** error may occur depending on the Access Type and the access permission (ACC) field of the corresponding PTE. Table 6-15. shows access type and corresponding page table entry access permissions.

Table 6-15. Access Types and PTE Permissions

	PTE(ACC)							
AT	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	--	--	--	--	2	--	3	3
1	--	--	--	--	2	--	--	--
2	2	2	--	--	--	2	3	3
3	2	2	--	--	--	2	--	--
4	2	--	2	--	2	2	3	3
5	2	--	2	--	2	--	2	--
6	2	2	2	--	2	2	3	3
7	2	2	2	--	2	2	2	--

A **Translation** error is indicated if an external bus error is generated while the MMU is fetching an entry from a page table, a PTD is found in a level-3 page table, or a PTE has ET=3. An **Access Bus** error is generated when an external bus error occurs during memory access that is not a page table walk access. An **Internal Error** is generated if:

1. The PTE[C] is set, and the PPN is pointing to non-DRAM/AFX space, or
2. A PTD[PPN] is pointing to non-DRAM space.

FAV	The Fault Address Valid bit is set when the contents of the Synchronous Fault Address Register (SFAR) is valid. The SFAR is valid for all types of faults.
OW	The Overwrite bit is set when the Fault Status Register has been written more than once by faults of the same class since the last time is was read. If an instruction access fault occurs, and the OW bit is set, system software must determine the cause by probing the MMU and/or memory.

6.2.5 Synchronous Fault Address Register (VA = 0x0000_04xx/14xx)

The synchronous fault address register (SFAR) contains the 32-bit virtual address of the fault which was recorded in the synchronous fault status register (SFSR). Fault addresses in the SFAR are overwritten according to the same priority used in the SFSR. Table 6-16. shows the addresses at which the SFAR can be accessed, and the corresponding operation.

Table 6-16. Accessing the Synchronous Fault Address Register

VA[15:8]	Read/Write	Operation
0x04	Read	Read SFAR
0x04	Write	No effect
0x14	Read	Read SFAR
0x14	Write	Write SFAR

6.2.6 CPU Configuration Register (VA = 0x0000_06xx)

The CPU Configuration Register (CCR) is unique to the TurboSPARC microprocessor and provides clock, secondary cache, and internal diagnostic information. Figure 6-28. shows the format of the CPU configuration register.

31	30	29 28	27 26	25 23	22 20	19 16	15 8	7 6	5	4	3	2 0
IOClk	SNP	AXClk	RAH	WS	rsvd	rsvd	rsvd	SBC	WT	uS2	SE	SCC
1	1	2	2	3	3	4	8	2	1	1	1	3

Figure 6-28. CPU Configuration Register

IOClk	The I/O clock bit indicates the divide ratio between the IU/FPC/FPU clock and the I/O clock. Setting the bit indicates a divide ratio of 2, while clearing the bit indicates a divide ratio of 3.
SNP	Direct Virtual Memory Access snoop enable. When set, the DVMA snoop is enabled. All DVMA transactions are snooped in the data cache.
AXClk	This 2-bit field indicates the ratio of the AFX clock to I/O clock. At power-up, the AXClk field is initialized to the ratio determined by the IOC_Range[1:0] inputs on power-up. However, the field can be altered by execution of a store alternate instruction. Table 6-17. shows the encoding of the AXClk field.

Table 6-17. AFX Clock to I/O Clocks Divisors

AXClk[1:0]	Divide By
00	2
01	3
10	4
11	5

RAH	The 2-bit Row Address Hold field controls the number of DRAM row-address hold cycles. The field is cleared (00) in power-up, and can be modified by executing an STA instruction. Table 6-18. shows the encoding of the RAH field and the corresponding number of wait states.

Table 6-18. Encoding the Row Address Hold Field

RAH[1:0]	Number of Hold Cycles
00	1
01	2
10	3
11	4

WS The 3-bit Wait State field controls the number of DRAM wait states and other DRAM timing factors. The field is cleared (000) on power-up and can be modified by writing to the CCR. Table 6-19. shows the encoding of the WS field.

Table 6-19. DRAM Access Times

WS[2:0]	t_ASR	t_ASC	t_CAH	t_CAS	t_CP	t_RAS	t_RP	t_CSR	Unit
000	3	3	6	3	6	11	8	3	nS
001	4	3	7	3	7	11	8	4	nS
010	5	4	9	4	9	13	9	5	nS
011	6	5	11	5	11	13	9	6	nS
100	1	1	2	1	2	7	5	1	nS
101	2	1	3	1	3	7	6	2	nS
110	2	2	4	2	4	9	7	2	nS
111	3	2	5	2	5	9	7	3	nS

t_ASR - row address setup time

t_ASC - column address setup time

t_CAH - column address hold time

t_CAS - CAS pulse width time

t_CP - CAS precharge time (min)

t_RAS - RAS pulse with time

t_RP - RAS precharge time (min)

t_CSR - CAS setup time before refresh

Table 6-20. DRAM Setup and Hold Times

WS[2:0]	t_RDS SCC=0	t_RDS SCC≠0	t_RDH SCC = 0	t_RDH SCC≠0	t_RDLS SCC = 0	t_RDLS SCC≠0	t_DS SCC = 0	t_DS SCC≠0	t_DH SCC = 0	t_DH SCC≠0	Unit
000	5	8	0	0	-	6	3	3	6	4	nS
001	6	9	0	0	-	7	3	4	7	4	nS
010	8	12	0	0	-	9	4	5	9	5	nS
011	10	15	0	0	-	11	5	6	11	6	nS
100	1	2	0	0	-	2	1	1	2	2	nS
101	2	3	0	0	-	3	1	2	3	2	nS
110	3	5	0	0	-	4	2	2	4	3	nS
111	4	6	0	0	-	5	1	3	5	3	nS

SCC - SCC field of CPU configuration register: 0 - no secondary cache,

not 0 - secondary cache

t_RDS - CPU read data setup - CAS falling edge to IO_CLK rising edge

t_RDH - CPU read data hold - CAS falling edge to IO_CLK rising edge

t_RDLS - FIFO read data setup - CAS falling edge to LE falling edge

t_DS - write data setup time

t_DH - write data hold time

SBC The 2-bit SBus Clock field indicates the external SBus clock control setting and may be modified by writing to the CCR. The SBC bit controls the ratio between the I/O clock and the SBus clock.

00	SBus clock = IOClk/4
01	SBus clock = IOClk/4
10	SBus clock = IOClk/5
11	SBus clock = IOClk/6

WT The write-through enable bit determines when the data cache operates in write-through mode. The WT bit is cleared at power-up and can be modified by writing to the CCR. The data cache must be turned OFF when changing the state of the WT bit. This bit is read/write.

SE The Secondary Cache enable bit indicates the presence of a secondary cache in the system. The bit is cleared on power-up and can be set by writing to the CCR. This bit is read/write.

uS2 The MicroSPARC-II compatibility mode bit allows the TurboSPARC processor to maintain compatibility with the MicroSPARC-II. The bit is cleared on power-up, indicating that the processor is in TurboSPARC mode. Setting the bit causes the version (ver) fields in the MMU Configuration Register, the Processor Status Register, and the Floating Point Status Register to change from 0x5, which indicates the TurboSPARC version, to 0x4, which indicates the MicroSPARC-II version. When the uS2 bit is set, write accesses to ASI 0xE (data cache tag accesses) are interpreted as a data cache flush (ASI 0x10:14). This bit is read/write.

SCC The 3-bit SCC field indicates the type of secondary cache configuration. The SCC field is set based on the value of the EC_CONFIG[2:0] pins during power-up and can be modified by writing to the CCR. Table 6-21. shows the various secondary cache configurations which are supported. This bit is read/write.

Table 6-21. Secondary Cache Configurations

SCC[2:0]	Number of Banks	SCache Size
000	--	No Secondary Cache
001	1	256 KBytes
010	2	512 KBytes
011	4	1 MByte
100	1	512 KBytes
101	2	1 MByte
110	1	1 MByte
111	--	Factory Test Mode

6.3 IOMMU REGISTERS

The IOMMU register set resides in physical address space and is accessed through load and store alternate instructions. The IOMMU register set contains general purpose mode and status registers, SBus specific registers, and specific registers for use with the Sun Microsystems compatible AFX graphics bus. All IOMMU registers are directly accessible by software and one or more entries can be flushed using a control space access. Each register is mapped to a dedicated physical address space. Table 6-22. lists the IOMMU registers and their corresponding physical address.

Table 6-22. IOMMU Registers Set Address Map

PA[30:0]	Register	Access
1000_0000	IOMMU Control	R/W
1000_0004	IOMMU Base Address	R/W
1000_0014	Flush All IOTLB Entries	W
1000_0018	Address Flush	W
1000_1000	Asynchronous Fault Status	R/W
1000_1004	Asynchronous Fault Address	R/W
1000_1010	SBus Slot Configuration 0	R/W
1000_1014	SBus Slot Configuration 1	R/W
1000_1018	SBus Slot Configuration 2	R/W
1000_101C	SBus Slot Configuration 3	R/W
1000_1020	SBus Slot Configuration 4	R/W
1000_1050	Memory Fault Status	R/W
1000_1054	Memory Fault Address	R/W
1000_2000	Module Identification	R/W
1000_3018	Mask Identification	R
1000_4000	AFX Queue Level	W
1000_6000	AFX Queue Level	R
1000_7000	AFX Queue Status	R

6.3.1 IOMMU Control Register (PA = 1000_0000)

The IOMMU Control Register (IOCR) contains control and status bits for the IOMMU. Figure 6-29. shows the format of the IOMMU control register.

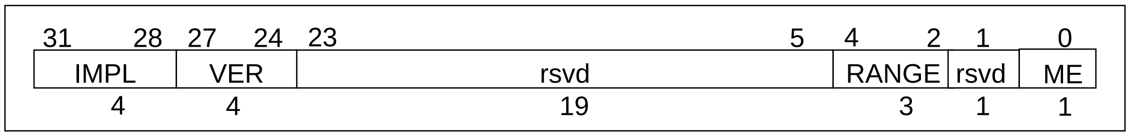


Figure 6-29. IOMMU Control Register

TurboSPARC DaughterBoard Addendum

Table of Contents

1.0 Introduction:

The TurboSPARC daughterboard or module is designed to replace the microSPARCII microprocessor operating in a SPARC Station 5 workstation. Performance enhancement of two times or more over existing system equipped with the microSPARCII is feasible with the use of the daughterboard.

The daughterboard is 3.162" by 4.170" in dimension and has an L-shaped outline that fits neatly onto the motherboard without interfering with the space of Sbus slots, or other system connectors. Using an on-board frequency generator, the CPU clock operates at 160 Mhz.

2.0 Features of the TurboSPARC DaughterBoard

- Pin to pin compatible with the microSPARCII
- Plug into the Swift socket using a surface-mounted 321-pin connector.
- 512KB of direct mapped, write through L2 Cache
- L2 Cache implemented using SRAM of 12ns cycle time or better
- Runs existing software applications without modifications
- High Efficiency Switching Regulator for 3.5 V supply
- On-board Programmable Fequency Generator
- 4-level deep Fifos for buffering to the DRAM
- Fixed Factory setting using pull-up/pull-down resistors
- 12-layer, 50-ohm controlled impedance, double-sided PCB

3.0 Functional Description of the TurboSPARC DaughterBoard

The TurboSPARC daughterboard is a highly densed board that has four 100-pin TQFP package SRAM that are used as secondary cache, 4 FiFos for buffering to the DRAM, an AT&T Power SIP (switching regulator) to supply 3.5V supply to the CPU core, I/O and SRAM, a clock generation circuitry, a filtering circuit to supply PLL voltage to the CPU, and other passive components.

The secondary cache is configured in two banks as 256KB per bank. The CPU supplies a spearate clock to each bank for power-down purposes. The cache is implemented using fast synchronous SRAM that are in linear pipeline burst mode. For CPU_CLK of 160 MHz, the IO_CLK is 80 MHZ or 12.5ns. This means that 12 ns cycle time or better of SRAM should be used. The SRAM should also have 5V tolerant I/O. (Motorola MCM69P536-5 32KX36 Synchronous SRAM has been used for this purpose).

A functional block diagram of the TuboSPARC daughterboard is shown below:

TurboSPARC DaughterBoard Block Diagram

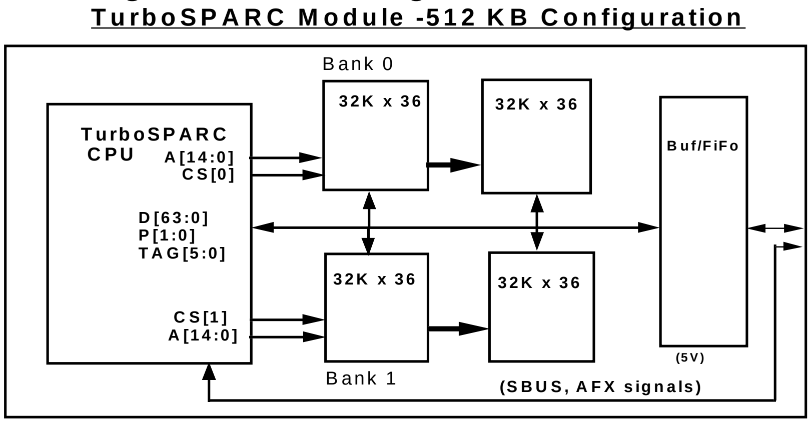


Figure 1.0

4.0 Power Dissipation

The TurboSPARC daughterboard operates on the 5V supply through the Swift connector (see Figure 1.0) on the motherboard. The 3.5V supply from the motherboard is not connected to the Swift connector. The daughterboard has its own 3.5V regulator that supplies 3.5 V needed on microprocessor and the SRAM. Power dissipation on the TurboSPARC microprocessor is estimated to be approximately 7W. The CPU core voltage runs at 3.5V and consumes about 2A of current. Power dissipation on the SRAM, FiFos, passive components, regulator, is estimated to be 8W. So, total power comsumption is estimated to be about 15W.

The following compares the power dissipation between TurboSPARC DaughterBoard and MicroSPARCII microprocessor:

TurboSPARC Daughterboard vs microSPARCII Power Requirements

TurboSPARC DaughterBoard	microSPARCII CPU
CPU supply voltages	CPU supply voltages:
Vdd1 = Vdd core = 3.5V	Vdd1 = Vdd core = 3.5V
Vdd2 = Vdd 5IO = 5.0V	Vdd2 = Per IO = 5.0V
Vdd3 = Vdd 3IO = 3.5V	Vdd3 = Mem IO = 5.0V
Vdd4 = Vdd PLL = 3.5V	Vdd4 = Mem IO = 3.5V
	Vdd5 = Vdd PLL = 5.1V
3.5V supply stepped down from 5V on-board switching regulator	3.5V from 321-pin Swift socket
CPU 7 W (typ)	CPU 9W (max)
SRAM, Fifo etc 8W (est)	
Total: 15W (est)	

5.0 Frequency Generation

The daughterboard has on-board circuitry that generates external clocking frequency required to drive the TurboSPARC microprocessor. This frequency is called External Clock 1 (EX_CLK1). If EXT_CLK1 =40.0 MHz, TurboSPARC would be operating at 160 MHz   CPU_CLK, depending on the divide ratio as will be discussed in section 5.1.

EXT_ CLK1 depends on the setting of the input to the programmable frequency generator (AV9107). The input switches (FS3, FS2, FS1, FS0) are either tied high or low by the use of pull-up or pull-down resistors. The programmable frequency generator takes an input reference frequency of 16 Mhz and scale it by a factor depending on the input switches.

Frequency Decoding Table. 16 MHz Input

FS3	FS2	FS1	FS0	EXCLK1(MHz)	CPU_CLK(MHz)	
				IOCLK_DIV2 =	1	= 0
0	0	0	0	24	96	144
0	0	0	1	26.4	105.6	158.4
0	0	1	0	28	112	168
0	0	1	1	30	120	180
0	1	0	0	34	136	204
0	1	0	1	36	144	216
0	1	1	0	36.8	147.2	220.8
0	1	1	1	38	152	228
1	0	0	0	38.4	153.2	230.4
1	0	0	1	40	160	240
1	0	1	0	42	168	252
1	0	1	1	44	176	264
1	1	0	0	46	184	276
1	1	0	1	48	192	288
1	1	1	0	50	200	300
1	1	1	1	80	320	480

Figure 3.0

So, to obtain a CPU_CLK of 160 MHz, (FS3, FS2, FS1, FS0) = 1001 with IOCLK_DIV2=1.

5.1 Clocking Scheme

The following is an example depicting the generation of various clocks (SBus clock, AFX clock, CPU_CLK, IO_CLK) using the on-chip PLL and different division factors available on the TurboSPARC microprocessor. Please refer to Figure 4.0 for the details of the TurboSPARC microprocessor clock interface. The example below is for 160 Mhz module.

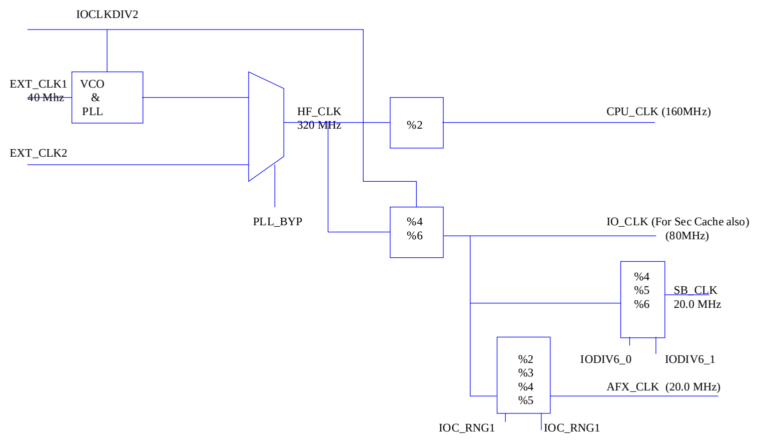


Figure 4.0

A 40 MHz clock input is assumed at the EXT_CLK1 input pin of the TurboSPARC CPU in this example. The EXT_CLK2 pin is tied to the ground.

If the PLL_BYP pin is tied to high, then in non-bypass mode the PLL (Phase Locked Loop) will generate the HF_CLK.

The multiplication factor for the PLL is dependent on the state of the pin IOCLK_DIV2.

If IOCLK_DIV2 is tied to high then
HF_CLK = EXT_CLK1 * 8 = 40 MHz * 8 = 320 MHz

If IOCLK_DIV2 is tied to low then
HF_CLK = EXT_CLK1 * 12 = 40 MHz * 12 = 480 MHz

5.2 CPU Clock

In this example we have selected the option of IOCLK_DIV2 tied to high. Because when IOCLK_DIV2 is tied to low, HF_CLK = 480 MHz.

A 480 MHz HF_CLK will yield 240 MHz CPU_CLK, which is not in the specified range of the TurboSPARC microprocessor clock. The maximum TurboSPARC microprocessor clock allowed is 170MHz. (later version may allow up to 200 MHz.)

Also CPU_CLK will always be
= HF_CLK / 2 (independent of the state of pin IOCLK_DIV2)
= 320MHz / 2 = 160MHz

5.3 IO Clock

Generation of IO_CLK is based on the division factor derived from the state of pin IOCLK_DIV2.

When IOCLK_DIV2 is tied to High the HF_CLK = EXT_CLK1 * 8, and the IO_CLK = HF_CLK / 4

Therefore IO_CLK = 320 / 4 = 80 MHz

When IOCLK_DIV2 is tied to Low the HF_CLK = EXT_CLK1 * 12, and the IO_CLK = HF_CLK / 6

Therefore IO_CLK = 480 / 6 = 80 MHz

5.4 SBus Clock

Generation of the SBus clock (SB_CLK) depends on the state of the pins IOCLK_DIV6[0] and IOCLK_DIV6[1]. The source of the SBus clock is the IO_CLK derived from the HF_CLK. SBus Clock should be in the range 16.67-25.0MHz.

The following table 5.0 depicts various division factors of the SB_CLK:

IO_CLK MHz	Pin IOCLK_DIV6[0]	Pin IOCLK_DIV6[1]	Division Factor	SB_CLK MHz
66.7 - 100.0	0	X	4	16.6 - 25.0
100.0 - 125.0	1	0	5	20.0 - 25.0
125.0 - 150.0	1	1	6	20.8 - 25.0

Table 5.0

5.5 AFX Clock

Generation of the AFX clock depends on the state of the pins IOC_RANGE[0] and IOC_RANGE[1]. The source of the AFX clock is the IO_CLK derived from the HF_CLK. The following table 6.0 depicts AFX clock generation.
AFX Clock should be in the range 20.0-42.0MHz.

IO_CLK MHz	Pin IOC_RANGE[0]	Pin IOC_RANGE[1]	Division Factor	AFX_CLK MHz
66.7 - 72.0	0	0	2	33.3 - 36.0
66.7 - 100.0	0	1	3	22.2 - 33.3
100.0 - 125.0	1	0	4	25.0 - 31.3
100.0 - 125.0	0	1	3	33.3 - 41.66
125.0 - 150.0	1	1	5	25.0 - 30.3

Table 6.0

6.0 Secondary Cache Configurations

The CPUMODE bits are responsible for setting up the Secondary Cache Configuration field (SCC[2:0]) in the TurboSPARC CPU Configuration Register(CCR) (VA = 0x0000_06xx).

This field controls the Secondary Cache Configuration as well as the number of banks. It is normally preset during the Power-On-Reset with a value asserted on the external configuration pins SC_CONFIG[2:0] i.e. The information on these pins is loaded into the SCC[2:0] field of the CPU configuration Register. The SCC[2:0] field, after Power-On-Reset may be altered by writing into the CCR.

The following Table 7.0 describes the relationship between the CPUMODES and the SCC[2:0] field. CPUMODE0 = SCC bit 0, CPUMODE1 = SCC bit 1 and CPUMODE2 = SCC bit2.

CPUMODE2	CPUMODE1	CPUMODE0	# of banks	Configuration
0	0	0	-	No Sec Cache
0	0	1	1	256KB
0	1	0	2	512KB
0	1	1	4	1MB
1	0	0	1	512KB
1	0	1	2	1MB
1	1	0	1	1MB
1	1	1	-	-

Table 7.0

7.0 Resistor Pull-up / Pull-down

The input signals described above IO_DIV2, IOC_RNG[1:0], IOCLK_DIV6[1], FS3, FS2, FS1, FS0 are all pulled-up or pulled-down based on the loading of its corresponding resistors. (Note that IOCLK_DIV6[0] and CPUMODE2 are hardwired to GND independent of the resistors.)

The Following Table 8.0 summarises which resistor to load/no-load based which logic level, based on its reference designators.

	For Logic Level ‘0’	For Logic Level ‘1’
IO_DIV2	Don’t load R68 Load R69	Don’t load R69 Load R68
IOC_RNG[1]	Don’t load R66 Load R67	Don’t load R67 Load R66
IOC_RNG[0]	Don’t load R70 Load R71	Don’t load R71 Load R70
IOCLK_DIV6[1]	Don’t load R64 Load R65	Don’t load R65 Load R64
FS3	Don’t load R43 Load R77	Don’t load R77 Load R43
FS2	Don’t load R42 Load R76	Don’t load R76 Load R42
FS1	Don’t load R41 Load R75	Don’t load R75 Load R41
FS0	Don’t load R40 Load R74	Don’t load R74 Load R40
CPUMODE1	Don’t load R72 Load R73	Don’t load R73 Load R72
CPUMODE0	Don’t load R62 Load R63	Don’t load R63 Load R62

For the daughterboard to operate the microprocessor at 160 MHz CPU_CLK, the following resistors are NOT loaded onto the board:

R73	R62	R64	R71	R66	R69
R42	R41	R74	R77	RP1	RP3

Table 8.0

8.0 OBP (Open Boot Prom)

The Open Boot PROM on the SPARC Station 5 motherboard needs to be upgraded to reflect the changes in the TurboSPARC microprocessor. This PROM is supplied together with the daughterboard.

OpenBoot PROM Addendum

OBP for the TurboSPARC

The OBP version 2.15 has been modified for the TurboSPARC microprocessor in a SS5-like system environment. This new code makes an assumption that Swift CPU has been removed from SparcStation 5 system and has been replaced with TurboSPARC CPU.

There is a **diff** file that contains the differences between the 2.15 source code and TurboSPARC source code. The new code contains a new directory structure below the arch/sun4m directory. The following is a list of all the new directories created with source code.

1. **arch/sun4m/microsparc/sr71**
2. **arch/sun4m/skylark**
3. **arch/sun4m/skylark/release**
4. **arch/sun4m/post/skylark**

The **diff** file has been created from the root directory of the source code. The **diff** file was created with -c option. This option enables an easy recreation of the new code from 2.15 code with the **patch** utility. Steps to generate the binary file for the TurboSPARC OBP is listed at the end of this document.

Overview of the Code Changes

Basically the code has been modified to account for the changes between the two processors. The following is a comparison between Swift(microSPARCII) and TurboSPARC:

	SWIFT	**TurboSPARC**

CPU CLOCK

	70/85/110 MHz	**200 MHz**

ARCHITECTURE

ISA	**SPARC V8**	**SPARC V8**
MMU	**SRMMU**	**SRMMU**
Kernel	**SUN4M**	**SUN4M**

ADDRESS SPACE

Virtual	**32-bit**	**32-bit**
Physical	**31-bit**	**31-bit**
Pagesize	**4K**	**4K**

INTEGER UNIT

Pipe Stages	5	9
#Iwindows	8	8
IMUL/IDIV	Yes	Yes, using FPU

FPU

FSMULD	Yes	Yes
FQUAD	Unimpl Trap	Unimpl Trap

ICACHE

Type	Direct Map VAC	Direct Map VAC
Style	VIVT	VIVT
Size	16K	16K
Linesize	32B	32B
Numlines	512	512
IO Snoop	No	No
Flush	SPARC V8 (ASI 0x10-0x14)	SPARC V8 (ASI 0x10-0x14, 0x18-0x1C), FLASH 0x36
DiagAccess	ASI 0xC-0xD	0xC-0xD
DiagFormat	SWIFT	TurboSPARC

DCAHE

Type	Direct Map VAC	Direct Map VAC
Mode	Write-thru, Non-allocate	Write-back, Non-allocate
Style	VIVT	VIPT
Size	8K	16K
Linesize	16B	32B
Numlines	512	512
IO Snoop	No	Yes
Flush	SPARC V8 (ASI 0x10-0x14)	SPARC V8 (ASI 0x10-0x14), FLASH 0x37
DiagAccess	ASI 0xE-0xF	0xE-0xF
DiagFormat	SWIFT	TurboSPARC

MMU

Style	**SRMMU**	**SRMMU**
Contexts	**256**	**256**
TLB Arch	**Unified I/D/IO**	**Split ITLB, DTLB, IOTLB**
TLB Entries	**64**	**ITLB = 4, DTLB = 260, IOTLB = 16**
PTP Cache	**PPTP, L2 VPTP**	**1 Entry for each level + 4 entry CAM**
Bypass	**ASI 0x20**	**ASI 0x20**
DiagFormat	**SWIFT**	**TurboSPARC**

IOMMU

Style	**SUN4M**	**SUN4M**
TLB Entries	**64 (shared TLB)**	**16 (dedicated IOTLB)**

SBUS

Slots	**5**	**5**

AFX

Supported	**Yes**	**Yes**

MEMORY INTERFACE

Memory Config	**16-256MB**	**16-256MB**
Error Detect	**Parity 1bit/word**	**Parity 1bit/word**

EXTERNAL CACHE

Type	**Not supported**	**Direct Map PAC**
Mode		**Write-thru, Non-allocate**
Style		**PIPT**
Size		**256K-1024K**
Linesize		**32B**
NumLines		**8K-32K**
DiagAccess		**ASI 0x30**

DiagFormat	**TurboSPARC**

SCAN

Boundary	**Yes**	**Yes**
Internal	**Yes**	**No**

Steps to add the TurboSPARC code changes to the existing OBP code version 2.15

- **Suppose this is your original root directory:**

  ..xxxx/obp/obp-2.15/obp

- **Create a parallel test root directory at the same level as the original root directory. This is recommended, so that original obp files are not over-written.**

  ...xxxx/obp/obp_fujitsu/obp

- **Use the following command to copy the files into the test root directory**

```
..obp/obp_fujitsu/obp%   cp -rR  ../obp/obp-2.15/obp  .   <CR>
```

- **Also copy the following files into the test directory ..obp/obp_fujitsu/obp**

  obp.diff
  retrieve
  post2.0.bin.Z

- **Retrieve a program called "patch" from the following ftp site "prep.ai.mit.edu" or from any GNU ftp site.**

  Use the following commands to download the following patch/related files

```
cd pub <CR>                          //change directory to public
cd gnu <CR>                          //change directory to gnu
dir patch* <CR>                                  //Look for the patch file
The "dir" command displays the file "patch-2.1.1.tar.gz", now
set binary <CR>                   //Otherwise patch file may be downloaded as an
ASCII file
get patch-2.1.1.tar.gz <CR>          //download patch utility
get gzip-1.2.4.tar.gz  <CR>          //this is the zip/unzip utility program
```

  Use the following commands to Unzip and create the "patch.exe"

```
configure <CR>
make <CR>                            //Creates gzip.exe
gzip -d patch-2.1.1.tar.gz <CR>      //Unzips patch utility
tar -xvf patch-2.1.1.tar  <CR>       //Creates patch-2.1 directory
cd patch-2.1 <CR>                    //Enter the patch utility directory
configure <CR>
make <CR>                            //Creates patch.exe
```

- **Copy the program "patch.exe" into ...obp/obp_fujitsu/obp directory.**

- **Check the test root directory ..obp/obp_fujitsu/obp for the presence of the following files (the patch program file could be sitting in the /bin also)**

  obp.diff
  retrieve
  patch.exe

post2.0.bin.Z

- **At the root directory prompt type**

  ..obp/obp_fujitsu/obp% *retrieve obp.diff post2.0.bin.Z* **<CR>**

- **After the prompt % is returned, type**

  ..obp/obp_fujitsu/obp% *make sr71* **<CR>**

- **The binary file called "forth.prom" for the boot PROM will be available under the directory**

..obp/obp_fujitsu/obp/arch/sun4m/skylark/release

After downloading the "forth.prom" file into the programmer, make sure that the first four words starting at address 000000 read the following hex bytes:

```
10802F68
A1480000
10000000
10000000
```

Display on the Workstation screen after booting from the modified OBP

```
Power-ON Reset


Power-On Self Test

IU Register File Test   MMU Context Table Reg Test    MMU Context Register Test     MMU
Sync Fault Stat Reg Test  MMU Sync Fault Addr Reg Test  D-Cache Tag Ram Test
D-Cache Data Ram Test
D-TLB CAM Test
D-TLB RAM Test
I-Cache Tag Ram Test
I-Cache Data Ram Test
E-Cache Addr Bus Test
E-Cache Tag Bus Test
E-Cache Data Ram Test
E-Cache Tag Ram Test
Snoop Ram Test
IOMMU-TLB Test
IOMMU Control Register Test
IOMMU Base Addr Register Test
IOMMU SBus Config Registers Test
Set RAH to 3
Set WS to 3
DRAM Data Bus Test  Probing Bank 0 -- 32 Megs Found
Probing Bank 1 -- 8 Megs Found
Probing Bank 2 -- 0 Megs Found
Probing Bank 3 -- 0 Megs Found
Probing Bank 4 -- 0 Megs Found
Probing Bank 5 -- 0 Megs Found
Probing Bank 6 -- 0 Megs Found
Probing Bank 7 -- 0 Megs Found
DRAM Address Bus Test
Memory Address Pattern Test
D-Cache Hit Miss Test
D-TLB Hit Miss Test
I-TLB Hit Miss Test
I-Cache Hit Miss Test
E-Cache Hit Miss Test, cache_size=0x00080000 bytes
I-TLB Ram Test
FPU Register File Test        FPU Misaligned Reg Pair Test  FPU Single-precision Tests    FPU
Double-precision Tests    FPU SP Invalid CEXC Test      FPU SP Overflow CEXC Test     FPU
SP Divide-by-0 CEXC Test  FPU SP Inexact CEXC Test      FPU SP Trap Priority >  Test  FPU
SP Trap Priority <  Test  FPU DP Invalid CEXC Test      FPU DP Overflow CEXC Test     FPU
DP Divide-by-0 CEXC Test  FPU DP Inexact CEXC Test      FPU DP Trap Priority >  Test  FPU
DP Trap Priority <  Test  PROC0 Interrupt Regs Tests    Soft Interrupts OFF Test      Soft
Interrupts ON Test       PROC0 User Timer Test         PROC0 Counter/Timer Test      DMA2
E_CSR Register Test      LANCE Address Port Tests      LANCE Data Port Tests         DMA2
D_CSR Register Test      DMA2 D_ADDR Register Test     DMA2 D_BCNT Register Test
DMA2 D_NADDR Register Test    ESP Registers Tests           DMA2 P_CSR Register Test
DMA2 P_ADDR Register Test     DMA2 P_BCNT Register Test     PPORT Registers Tests
NVRAM Access Test             TOD Registers Test            initializing TLB initializing cache
```

```
Setting WS to fast mode

Allocating SRMMU Context Table  Setting SRMMU Context Register
Setting SRMMU Context Table Pointer Register
Allocating SRMMU Level 1 Table
Mapping RAM
Mapping ROM

ttya initialized
Probing Memory Bank #0 32 Megabytes
Probing Memory Bank #1 8 Megabytes
Probing Memory Bank #2 Nothing there
Probing Memory Bank #3 Nothing there
Probing Memory Bank #4 Nothing there
Probing Memory Bank #5 Nothing there
Probing Memory Bank #6 Nothing there
Probing Memory Bank #7 Nothing there
Probing CPU FMI,MB86907  512 KB Secondary Cache Present  IOCLK Setting = CPUCLK/2
SBUSCLK Setting = CPUCLK/8  Running in TurboSPARC Native Mode  Probing
/iommu@0,10000000/sbus@0,10001000 at 5,0  espdma esp sd st SUNW,bpp ledma le  Probing
/iommu@0,10000000/sbus@0,10001000 at 4,0  SUNW,CS4231 power-management  Probing
/iommu@0,10000000/sbus@0,10001000 at 1,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 2,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 3,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 0,0  Nothing there screen not found.
Probing Memory Bank #0 32 Megabytes
Probing Memory Bank #1 8 Megabytes
Probing Memory Bank #2 Nothing there
Probing Memory Bank #3 Nothing there
Probing Memory Bank #4 Nothing there
Probing Memory Bank #5 Nothing there
Probing Memory Bank #6 Nothing there
Probing Memory Bank #7 Nothing there
Probing CPU FMI,MB86907  512 KB Secondary Cache Present  IOCLK Setting = CPUCLK/2
SBUSCLK Setting = CPUCLK/8  Running in TurboSPARC Native Mode  Probing
/iommu@0,10000000/sbus@0,10001000 at 5,0  espdma esp sd st SUNW,bpp ledma le  Probing
/iommu@0,10000000/sbus@0,10001000 at 4,0  SUNW,CS4231 power-management  Probing
/iommu@0,10000000/sbus@0,10001000 at 1,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 2,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 3,0  Nothing there
Probing /iommu@0,10000000/sbus@0,10001000 at 0,0  Nothing there

SPARCstation 5, Keyboard Present
ROM Rev. 2.15, 40 MB memory installed, Serial #0.
Ethernet address 8:0:20:1:2:3, Host ID: 80000000.



putting patch to put cpu in  uSII mode  Snoop and IC-Snoop turned On  Testing  40 megs of
memory at addr        0  39 ok
```

********************** end Display screen *************