SuperSPARC II Addendum A

This document details SuperSPARC II implementation changes from its predecessor, SuperSPARC. It is meant to be a guide to implementors and users and should be used together with the *SuperSPARC and MultiCache Controller User's Guide* (also published by SPARC Technology Business). For changes made to the MultiCache Controller (MXCC), please refer to *MultiCache Controller Addendum*, also available from STB.

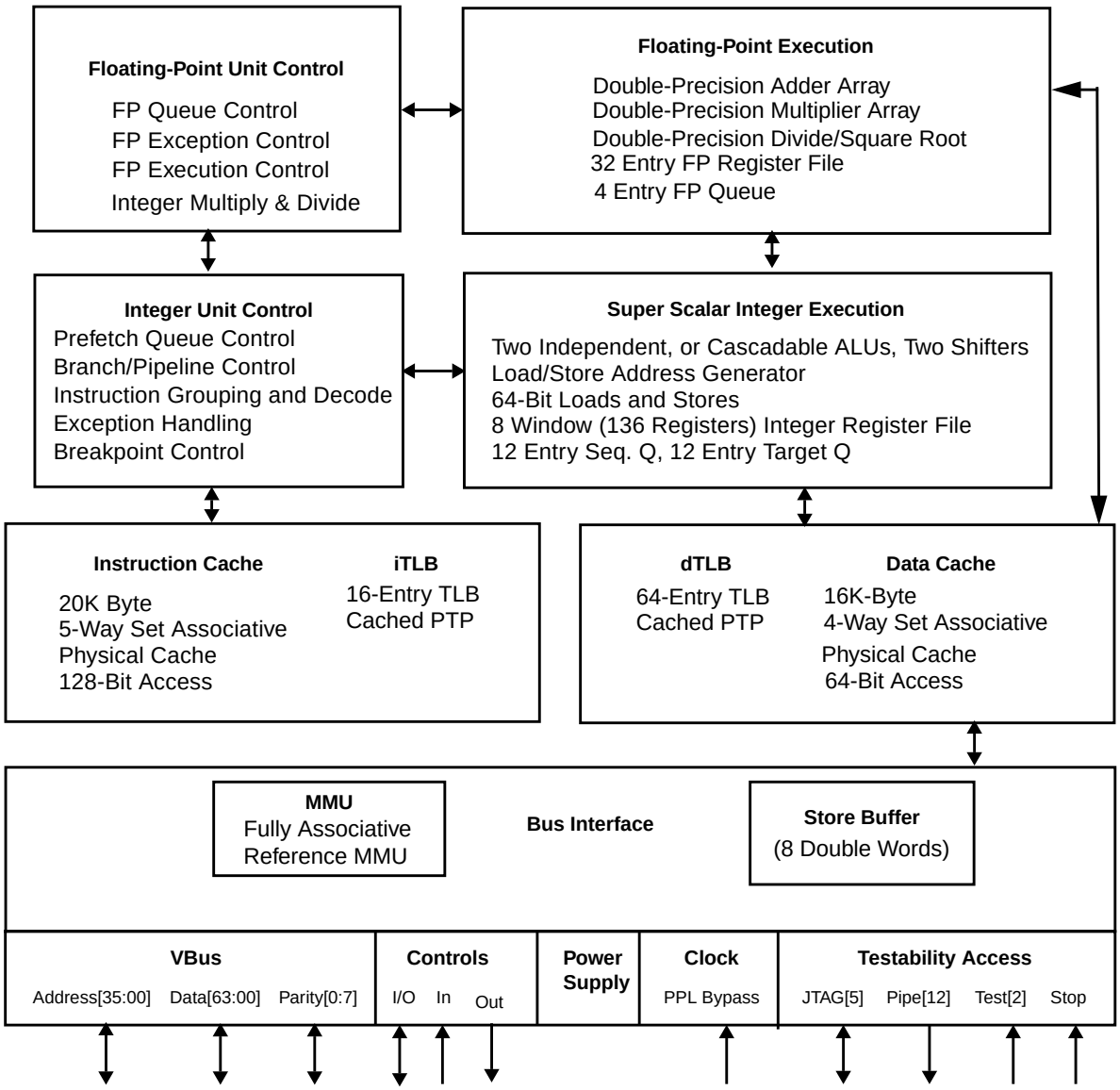
The SuperSPARC II micro processor is a highly integrated, high-performance follow-on to the SuperSPARC Micro Processor, a SPARC RISC implementation. The SuperSPARC II processor offers a low risk system interface upgrade for existing SuperSPARC processor based platforms. Changes from SuperSPARC to SuperSPARC II have been highlighted in this chapter.

A.1 SuperSPARC II Processor

The SuperSPARC II microprocessor integrates most of the support functions normally required to build a SPARC-based system. The SuperSPARC II processor is a highly-integrated, super-scalar SPARC processor for use in single- and multi- processor systems. It features physical instruction and data caches, a Memory Management Unit (MMU) with separate TLBs for the instruction and data cache, a Store Buffer, and a Floating Point Unit, all integrated on-chip.

Shaded areas in this document indicate SuperSPARC II processor implementation changes from the SuperSPARC processor.

SuperSPARC II Functional Block Diagram



Salient features of the SuperSPARC II processor redesign:

- IU pipeline redesign
- Read sources in parallel with decode
- Full cycle branch prediction
- PC Selection and Prefetch Logic
- Split MMU
- Better FPU
- Full Cycle Interface
- FPU-IU D-Cache interface redesign
- D-Cache-BUS Interface redesign
- Rising Edge design
- Better interunit exception handling protocols
- Better DFT features
- Enhanced BSCAN
- Dual Byte Ordering Support

A.2 Integer Unit

The SuperSPARC II integer unit has a redesigned pipeline, register file, instruction prefetcher, next PC generator, and epath/allocation logic. The trap handling logic has also been modified.

A.2.1 SuperSPARC II Integer Pipeline

The SuperSPARC II processor uses a four cycle pipeline. The major change in the integer pipeline between SuperSPARC and SuperSPARC II is the elimination of the decode D2 stage and increasing the write-back stage to a full cycle. All internal units of the SuperSPARC II design interact with each other only on the rising edge of SuperSPARC II clock. In the SuperSPARC II design all falling edge flip-flops are eliminated to ease timing thereby making the design a full cycle design.

Table A-1 SuperSPARC Pipeline Stages

Clock Edge	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
SuperSPARC Stage	F0	F1	D0	D1	D2	E0	E1	WB

Table A-2 SuperSPARC II Pipeline Stages

Clock Edge	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
SuperSPARC II Pipe Stage	F		D		E		W	

The four cycles of the SuperSPARC II pipeline are called F, D, E, and WB. Additional cycles are used to process floating point operations within the FPU. The function of each basic pipeline stage is:

F *(Fetch)*

Fetches instructions from memory and deposits them into the instruction prefetch queue. Most instruction fetches are assumed to hit in the single cycle access instruction cache and return four instruction words.

D *(Decode, Grouping, Read Rfile)*

Selects an in-order instruction group from the candidate instructions at the head of the instruction queue prefetch buffer. Selects and latches register indexes corresponding to the first memory reference. Forms extension words for memory reference and branch displacements. Checks scoreboard dependencies and inserts pipeline bubbles when dependencies can not be met by forwarding through existing bypass paths. Checks cascade (forwarding) requirements.

Assigns E-unit resource ports (resource allocation) to specific instructions in the selected instruction group. Uses forwarding requirements between instruction groups to steer forwarding paths during the lifetime of the instruction group. Reads memory reference address registers. Generates branch target address. Manages next PC and PPC (prefetch PC). Steers forwarding bypass paths for memory reference address computation.

Adds address register(s) to form virtual memory reference address. Reads operand registers. Steers forwarding paths into the next pipeline stage for operand execution.

E ***(Execute)***

Performs IU execution. Accesses data cache, probe store requests. Dispatches floating point commands. Resolves accumulated pipeline exceptions. Based on exception history, commits to all or a portion of the existing memory and register write operations demanded by the instruction group. For exceptions, identifies the faulty instruction and selects an appropriate next value for the trap type register field.

WB ***(Write Back)***

Writes register file and store buffer or data cache. During exceptions, stores the PC and nPC of the faulty instruction into %r17 and %r18. Updates the trap type register field in TBR and jams the selected trap PC in the instruction fetcher. This allows the appropriate trap handler to resolve the source of the exception (PC, nPC and TT).

A.2.2 Integer Unit Register File

The SuperSPARC II register file (rfile) differs from the SuperSPARC register file in three important aspects: no double-pumping, increased number of ports, and operand muxing after the rfile access (SuperSPARC muxes operands before rfile access).

The SuperSPARC rfile is double-pumped, i.e., the rfile is accessed twice in one cycle. In the first access two reads and two writes can be performed and in the second access four reads can be performed. The SuperSPARC II rfile performs one access each cycle.

The SuperSPARC II rfile has ten ports- eight reads and two writes. The SuperSPARC rfile has only four ports. The increase in number of ports eliminates the need to double-pump the rfile.

The increased number of ports allows SuperSPARC II to perform operand muxing after the rfile access. Instruction decoding is done in parallel to the rfile access and the results of decode are used to select which operands are needed. In SuperSPARC, operand muxing is done before rfile access.

Table A-3 SuperSPARC Four Port Rfile (4R,2R2W)

Clock	1	0	1	0	1	0	1	0
add -> rd1 add -> rd2	F0	F1	D0	D1	D2	E0	E1	WB
add rs1, rs2 add rs3, rs4			F0	F1	D0	D1	D2	E0
ldd/st [%rs1,%rs2]					F0	F1	D0	D1
							F0	F1

D0: Decode and identify Load/Store Instruction

D1: Read Two Load/Store Sources

D2: Read Two Load/Store or Four Source Instruction Operands

WB: Write Two Results

Table A-4 SuperSPARC II Ten Port Rfile (8R/2W)

clock	1	1	1	1
add -> %rd1 add -> %rd1	F	D	E	W
std %rs1,%rs2		F	D	E
add rs1,rs2 add rs3,rs4			F	D
add rs5,rs6				
				F

D: Read Six Source Instruction Operands

E: Read Two Source Store Data

W: Write Two Destination Results

Since decode is done in parallel to register file access, all six source operands for the next group of three alu operations are read. However, due to limited resources, grouping rules will not allow three alu operations in the same group.

A.2.3 Instruction Prefetching

Instruction prefetching is done to optimize SuperSPARC II's performance by ensuring that there is a steady flow of instruction candidates to the pipeline. There are three queues that allow this to happen.

The dispatch queue is a three-entry queue that holds the three instruction candidates that may be decoded on the current cycle. Grouping logic tells the dispatch control logic how many of these three instructions can be executed, and dispatch control then determines the new instructions to be dispatched. The addition of the dispatch queue allows the prefetcher to decouple the discard and append functions, which enables the design to meet performance goals while maintaining instructions per cycle (IPC).

The dispatch queue is filled from the remaining pair of queues, which can be referred to as the sequential queue and the target queue. Both queues are twelve entries deep. As long as there are no branches in the instruction stream, the prefetcher will try to fetch ahead until the sequential queue fills, at which point it will stop fetching new instructions. The target queue is used to hold branch target fetches which will be inserted into the dispatch queue if the branch is taken, at which point the queues will be "swapped." The queue that was the former sequential queue becomes the new target queue and vice versa. After the queues are swapped, new instructions are again placed into the sequential queue and the process repeats.

A.2.4 Next PC generation

Generation of the prefetch_pc's is done by the prefetch_control (pcntl) block. Pcntl uses a simple algorithm to determine the next_prefetch_pc. This selection is based upon the current state of the pipeline. The pipeline has four stages; fetch, decode, execute, and writeback. The oldest instructions are always further along in the pipeline. The algorithm works by looking for the oldest control transfer instruction (cti) in the pipe and then determining whether the cti has been completely processed. No state machines are required to implement this algorithm. Rather, a small amount of information is maintained about the pipeline, and all decisions are made based upon the locations of cti's in the pipe via a decode mechanism.

If there is a cti in writeback, then the cti may have a delay instruction. If the delay instruction (di) is not somewhere in the pipe (including fetch), then the di is the next instruction that will be executed, so it is the next fetch candidate. If the delay instruction is in the pipe, and the cti is taken, then the next instruction to be executed is the target of the cti. If the target is somewhere in the pipe, then processing for this branch instruction has been completed.

If there is a cti in execute, the delay and target instruction checking process is repeated.

If there is not a cti in execute, or the cti has been completely processed, then the decode stage of the pipe is examined, and so on. If there are no cti instructions in the pipeline to be processed, then a sequential fetch is issued, where the sequential_pc is simply the previous pc value plus the number of valid instructions that were returned by the instruction cache.

A.2.5 Epath/Alloc

To meet SuperSPARC II's target performance goals required the re-architecting of the epath along with its control mechanism. Due to the elimination of the D2 stage of SuperSPARC's pipeline, it was necessary to decouple the allocation of pipeline resources from the determination of group size. SuperSPARC II assumes that all three instructions will be grouped, and allocates accordingly, based upon the position of the first fpop or floating point load or store.

To ease this task, the Epath was modified to have symmetrical resources. In SuperSPARC, only ALU_1 had a barrel shifter. Both ALU_0 and ALU_1 have barrel shifters in SuperSPARC II.

A similar re-arrangement of logic was done to expedite the processing of call and jump instructions. Lastly, all forwarding muxes were grouped tightly with the resource receiving the forwarded data. This required very wide muxes, but allowed for the addition of the sign-extend function for load instructions without a significant penalty in performance.

A.2.6 Trap Handling

A.2.6.1 Breakpoint on Divide-By-Zero Faulting Instruction

In SuperSPARC II, the case where there is a breakpoint on a divide-by-zero instruction will cause a divide-by-zero exception instead of an Interrupt_12 exception as in SuperSPARC. This modification results in the higher priority interrupt being taken. The divide-by-zero fault has a higher trap priority than Interrupt_12.

A.2.7 IU Processor State Register

The 32-bit processor state register (PSR) holds key status and control information. The instructions that modify the PSR's fields include SAVE, RESTORE, Ticc, RETT, and any instructions that modify the condition codes. The privileged instructions RD PSR and WR PSR read and write the PSR directly.

The SuperSPARC II Processor uses one reserved bit to support dual byte ordering. The DE bit can be read and written using the RDPSR and WRPSR instructions only.

Table A-5 SuperSPARC II Processor State Register

ver	Icc	rsv'd	DE	rsv	EC	EF	PIL	S	PS	ET	CWP
31 24	23 20	19 16	15	14	13	12	11 8	7	6	5	4 0

PSR.DE bit 15 indicates whether the processor is in the big-endian or the little-endian mode of operation. 0=big-endian mode, 1=little-endian mode.

A.3 Floating Point Unit

The SuperSPARC II processor implements a completely redesigned SPARC compliant IEEE754 Floating Point Unit. The SuperSPARC II processor handles all special numeric cases. Programmer visible differences are the SuperSPARC II FPU pipeline interlock latencies, FSR trap type and underflow semantics, and NaN result formats.

A.3.1 Longer Floating Point Pipeline

The SuperSPARC II Floating point pipeline is tightly coupled to the integer pipeline. A floating point operation may be started every cycle. The SuperSPARC II floating point has a five stage pipeline compared to the four stage pipe of SuperSPARC. Forwarding paths exist between the Fw and Fd stages and between the Fw and Fp stages. The latter forwarding path can reduce the latency of dependent fpops by one cycle.

Table A-6 SuperSPARC-FPU Pipeline

Clock	1	0	1	0	1	0	1	0	1	0
SuperSPARC IU Pipe	D2	E0	E1	WB						
SuperSPARC FPU pipe			Fd		Fa		Fr		Fw	

In SuperSPARC II, floating point instructions are issued by the E stage of the integer pipe and are enqueued in a four entry floating point queue, where they remain until an exception-free result is computed. When all prior fpops have advanced to the Fp stage, an fpop in the IU E stage co-resides in the FPU Fd stage.

Table A-7 SuperSPARC II FPU Pipeline

clock	1	0	1	0	1	0	1	0	1	0
SuperSPARC II IU Pipe	E		WB							
SuperSPARC II FPU pipe	Fd		Fp		Fx		Fr		Fw	

Fd Dispatch Stage: Decodes the new floating point instruction from the integer pipe or the floating point queue. Reads the register file for new source operands. Checks dependency and determines candidates for execution. Forwards load data into source operands.

Fp Pre-Process Stage: Pre-processes the instruction operands to detect for sub-normal numbers. Prepares the operands for the Fx stage. The Fp stage also forwards results into source operands. A small portion of the Fadder and Fmultiply execution is done in the Fp stage. The Fp stage stalls until all subnormal source operands are pseudo-normalized.

Fx Execute Stage: Floating point instructions are in either the Add, Multiply or Divide/Square Root Execute stage. The Fx stage stalls for fp divide or square root until an unrounded result is computed.

Fr Rounding Stage: Results from the Fx stage are rounded in this stage depending on the FSR_rounding mode selection. The Fr stage stalls an additional cycle for subnormal results.

Fw Write Back: In this stage results are written back to the floating point register file, data is forwarded to the next stage of the pipe, and IEEE traps are reported when exceptions are detected.

A.3.2 Floating-point Latencies

Table A-8 Floating Point Latencies

FPOP	SuperSPARC II Cycles	SuperSPARC Cycles	SuperSPARC II dep. fpop to fpop	SuperSPARC dep. fpop to fpop
Fmovs,Fnegs, Fabss	4	3	3	3
Fcmps,d	4	3	3	3
Fadds,d	4	3	3	3
Fmuls,d	4	3	3	3
Fdivs	11	6	10	6
Fdivd	18	9	17	9
Fsqrts	10	8	9	8
Fsqrtd	17	12	16	12
Imul	4	4	-	-
Idiv	25	18	-	-

We define fpop latency as the nominal number of cycles for an fpop to advance from the Fp stage to the Fw stage. SuperSPARC II floating point add and multiply latencies have increased by one due to the longer floating point pipe. New divide and square root data path algorithms were adopted that increased cycle count but reduced cycle time. SuperSPARC II square root cycle count is 40% longer than SuperSPARC. SuperSPARC II divide cycle count is twice as long as SuperSPARC's.

Note that SuperSPARC II dependent FPOP to FPOP latencies are one cycle less than for the non-dependent case. This cycle reduction results in floating point add and multiply latencies equal to that of SuperSPARC. The latency reduction is achieved through the use of the Fw to Fp forwarding path.

All the SuperSPARC II numbers in the above table are if the 2 FPOPs are of the same precision. If the producer FPOP is of double precision and consumer FPOP is of single precision add 1 cycle. If the producer FPOP is of single precision and consumer FPOP is of double percussion add 2 cycles.

In SuperSPARC the latencies were same for all mixed precision operations.

A.3.2.1 Floating-Point latencies dependent on input operands

Table A-9 Normal sources, Sub-normal results

FPOP	SuperSPARC II Cycles	SuperSPARC Cycles
FMULs,FMULd,FsMULd	5	4
FDIVs	12	7
FDIVd	19	10

Table A-10 Sub-normal sources, Sub-normal results

	SuperSPARC II max cycles	SuperSPARC max cycles	SuperSPARC II max cycles	SuperSPARC max cycles
FPOP	both	subnormal	one	subnormal
FMULs,FMULd,FsMULd	20	8	13	7
FDIVs	26	10	19	10
FDIVd	33	13	26	13
FSQRTs	-	-	17	11
FSQRTd	-	-	24	15

A.3.3 Differences in IU and FPU Interlock Latencies

The SuperSPARC II floating point unit provides forwarding paths between the Write back stage of the floating point pipe to the pre-processor stage and the decode stage. Since the FPU pipeline length has increased, FP memory references may require more interlock cycles to resolve.

A.3.3.1 Fpop to Fpop Freg Interlock Latency

The following example shows how a floating point register dependency between two fpops activates data forwarding and interacts with the IU and FPU pipelines. The fmuld destination register is a source operand for a subsequent faddd. Both fpops are enqueued in the FPQ during the IU E stage without stalling the IU. A forwarding path

enables the faddd Fd stage to advance to Fp when the fmuld reaches the Fr stage. The second fpop nominally requires three cycles from Fp to Fw which is identical to SuperSPARC.

Table A-11 FPU Pipeline Forwarding Paths

Instruction Group	Clock Cycle/ FPU Instructions	1	2	3	4	5	6	7	8	9
Group 1:	fmuld %f0,%f2,%f4	E Fd	WB Fp	Fx	Fr	Fw				
Group 2:	faddd %f4, %f6, %f8		E Fd	WB Fd	Fd	Fp	Fx	Fr	Fw	
Group 3:	std %f8, [%l0+i1]			E’	E’	E’	E’	E’	E	WB

An FP memory reference in group 3 stalls the IU pipe until faddd reaches the Fw stage. The destination register for the FP memory reference is also the destination register of the faddd. A forwarding path enables the FP memory reference to receive its source data from the faddd Fw stage.

A.3.3.2 Fpop to Store Freg Interlock Latency

The following example shows forwarding paths from the fw stage to the fp and fd stages of the fpu pipeline. In SuperSPARC II, fpop to dependent store freg forwarding takes one cycle more than the SuperSPARC processor.

If the store is a double precision store waiting for a result from a single precision FPOP SuperSPARC II takes 2 more cycles than SuperSPARC.

Table A-12 FPOP to Store Freg Forwarding

Instruction Group	Clock Cycle/ FPU Instructions	1	2	3	4	5	6	7	8	9	10
Group 1:	fmuld %f0,%f2,%f4	E Fd	WB Fp	Fx	Fr	Fw					
Group 2:	faddd %f4, %f6, %f8		E Fd	WB Fd	Fd	Fp	Fx	Fr	Fw		
Group 3:	fdtos %f4, %f10			E Fd	WB Fd	Fd	Fp	Fx	Fr	Fw	
Group 4:	st %f10, [%l0+i1]						E’	E’	E’	E	WB

A.3.3.3 *Fpop to Load Freg Interlock Latency*

An fpop to load freg interlock occurs when a load is writing to the source register of a prior fpop. The subsequent load has to wait until the prior fpop is exception free. If the fpop can be determined to be exception free in the Fr stage, the load will wait three cycles as in SuperSPARC. Otherwise, the load will have to wait four cycles (one cycle more than SuperSPARC).

Table A-13 Case1: Three Cycle Wait for Load

Instruction Group	**Clock Cycle/ FPU Instructions**	1	2	3	4	5
Group 1:	Faddd %f0,%f0,%f0	Fd	Fp	Fx	Fr	Fw
Group 2:	ldd %f0		E’	E’	E	W

Table A-14 Case2: Four Cycle Wait for Load

Instruction Group	**Clock Cycle/ FPU Instructions**	1	2	3	4	5	6
Group 1:	Faddd %f0,%f0,%f0	Fd	Fp	Fx	Fr	Fw	
Group 2:	ldd %f0		E’	E’	E’	E	W

A four cycle dependent LDF latency is visible when any of the following is true:

a. FSR.TEM enables inexact traps,

b. the prior fpop is a convert (e.g. fdtoi, fstoi, etc.)

c. the prior fpop is an fadd/fsub and operand exponents differ by less than 1

d. the prior fpop is a multiply, divide, or square root and the operand exponents are such that the result may overflow/underflow

A.3.3.4 SuperSPARC II divide in execution holds up the whole FP pipeline

Any FPOP in the pipe preceding or following an FDIV will have to wait until the FDIV moves out of Fx stage.

Table A-15 FDIV holds up whole FP pipeline

Instruction Group	**Clock Cycle/ FPU Instructions**	1	2	3	4	5	6	7
Group 1:	Faddd %f0,%f0,%f0	Fd	Fp	Fx	Fr	Fw	Fw	
Group 2:	fmuld %f6,%f8,%f10		Fd	Fp	Fx	Fr	Fr	
Group 3:	fdivd %f20,%f22,%f24			Fd	Fp	Fx1	Fx2	

From cycle 6 until the FDIV moves out of Fx the FP pipe stalls. If a std %f10 is issued, it has to wait until the FMULD moves to Fw stage. But a std %f4 issued in cycle 5 or cycle 6 will complete in the same cycle by using the forwarding path.

A.3.3.5 Store Fsr Freg Interlock Latency

FPU - Exception Mode

In the SuperSPARC II FPU, a st %fsr instruction takes three cycles to complete, whereas in the SuperSPARC processor it completes in one cycle. SuperSPARC II STFSR takes two cycles more than SuperSPARC.

Table A-16 Store FSR - FPU_Exception mode

Instruction Group	**Clock Cycle/ FPU Instructions**	1	2	3	4	5	6	7
Group 1:	FPOP(generates exception)	Fd	Fp	Fx	Fr	Fw (X)		
Group 2:	st %fsr, [%l0+%l1]		E’	E’	E’	E’	E	W

FPU -Normal Mode

When the floating point queue is empty, SuperSPARC II STFSR completes in four cycles. In SuperSPARC, STFSR takes one cycle. Thus, a STFSR takes three cycles longer in SuperSPARC II than in SuperSPARC.

Table A-17 Store FSR - FPU_Normal mode

Instruction Group	**Clock Cycle/ FPU Instructions**	1	2	3	4
Group 1:	st %fsr, [%l0,%l1]	E’	E’	E	W

A.3.3.6 FBfcc Freg (FSR) Interlock Latency

In the SuperSPARC II processor a FBfcc instruction is not issued until all fcompare instructions are completed. The IU pipe stalls in the D stage of the pipe.

In the SuperSPARC processor a FBfcc instruction stalls in the D2 stage of the pipe until all fcompare instructions are complete.

The SuperSPARC II FBfcc instruction takes one cycle longer to complete than SuperSPARC.

A.3.3.7 Unimplemented & Illegal Fpop Interlock Latency

In SuperSPARC II, unimplemented fpops (e.g. quad fpops) hold the IU pipe until the FPU completes all prior fpops in the FP queue. It then releases the IU pipe, enqueues the unimplemented or illegal fpop instruction into the FPQ, enters fpu pending exception mode, and then waits for the next fpop or FP memory instruction (e.g. LDF, STFSR) to enter the IU E pipeline stage. *Thus, in SuperSPARC II, when an unimplemented trap is taken, the unimplemented instruction is the only FPQ instruction. In SuperSPARC, unimplemented fpops can co-reside with other FP instructions in the FPQ.*

A.3.3.8 Illegal Integer Divide Interlock Latency

The SuperSPARC II FPU implements a 52 bit by 32-bit integer divide identical to the SuperSPARC processor (compared to 64-bit by 32-bit specification). SuperSPARC II signals an illegal_instruction trap (trap type 0x2) when the dividend has numerically significant bits beyond bit 51.

SuperSPARC II detects illegal_instruction_trap and divide_by_zero exceptions in the integer pipe without dispatching the idiv to the FPU pipe. SuperSPARC dispatches the illegal idiv to the FPU and does not report the exception until the divide completes execution. Thus, SuperSPARC II requires much less latency to detect these illegal integer divide operations.

A.3.4 Differences in FPU Functionality

A.3.4.1 Floating Point Status Register

FSR_VER bits[19:17]

The SuperSPARC II and SuperSPARC processors set FSR.VER field to 0x0. The PSR.VER, MCNTL.IMPL, and MCNTL.VER numbers identify SuperSPARC II.

Clearing FSR_TRAP_Type

The SuperSPARC II FPU clears the FSR_FTT_bits on a st %fsr instruction, *after* writing the FSR register to memory.

The SuperSPARC processor clears the FSR_FTT bits only when a new FPOP is executed.

A.3.4.2 Quad Precision

Neither SuperSPARC II nor SuperSPARC support quad precision arithmetic.

A.3.4.3 Underflow

The SuperSPARC II Processor detects underflow condition ***before*** rounding.

The SuperSPARC processor detects underflow condition ***after*** rounding.

SuperSPARC II conforms to the SPARC Architecture Manual Appendix recommendation N5. The IEEE specification allows either method.

A.3.4.4 Convert to Integer NaN Result Formats

SuperSPARC II integer convert of a source operand whose value is a NaN (Not A Number) produces one of two result formats: 0x8000000 (-NaN) or 0x7fffffff (+NaN) depending upon the sign of the source operand NaN.

SuperSPARC always produces a fixed representation of 0 in the destination register for either of the above two cases.

A.3.4.5 All Other NaN Result Formats

For all fpops other than convert to integer, SuperSPARC II follows the SPARC Architecture Appendix recommendation N4 for fpop results that are NaN's.

The format of NaN results depends upon the source operand NaN's. If both source operands of a diadic fpop are NaN's, the result will be the NaN value of the second operand.

Table A-18 SuperSPARC II FPU NaN format

Sign	Exponent	Fraction
Sign of Operand Src2 or Src1	All 1's	Fraction of Src2 or Src1 operand

The SuperSPARC FPU NaN results have a fixed format:

Table A-19 SuperSPARC FPU NaN format

Sign	Exponent	Fraction
0x0	All 1's	1000..... 0

A.3.4.6 Monadic Treatment

In the SuperSPARC II f_unit, the load to FPOP dependency checking logic does not take into account that certain instructions are monadic. The result is that since the rs1 field is zero for monadics, a false dependency is detected when the subsequent load

instruction uses %f0. In the case where the instruction following the monadic is “ld %f0,” extra latency will be incurred due to the pipe being stalled until the monadic is done. So any monadic followed by “ld %f0” will cause f_busy.

But the FPOP to FPOP dependency checking logic does take this into account (as in SuperSPARC). That is, in the following sequence the fsqrts will be dispatched without waiting for the fadds to complete, even though the rs1 field of fsqrts is zero.

```
fadds  %f2,%f4,%f0
fsqrts %f6,%f8
```

A.3.4.7 *Floating Point Latencies*

SuperSPARC II Floating Point instruction whose latency depends on input operands

Table A-20 Normal sources, Normal results

Fpop	SuperSPARC II	SuperSPARC
FMULs,FMULd,FsMULd	4	3
FDIVs	11	6
FDIVd	18	9
FSQRTs	10	8
FSQRTd	17	12

Table A-21 Normal sources, Sub-normal results

Fpop	SuperSPARC II	SuperSPARC
FMULs,FMULd,FsMULd	5	4
FDIVs	12	7
FDIVd	19	10

Table A-22 Sub-normal sources, Sub-normal results

Fpop	SuperSPARC II both	SuperSPARC subnormal	SuperSPARC II one	SuperSPARC subnormal
FMULs,FMULd,FsMULd	20 cycles max	8 cycles max	13 cycles max	7 cycles max
FDIVs	26 cycles max	10 cycles max	19 cycles max	10 cycles max
FDIVd	33 cycles max	13 cycles max	26 cycles max	13 cycles max
FSQRTs	-	-	17 cycles max	11 cycles max
FSQRTd	-	-	24 cycles max	15 cycles max

A.4 Memory Management Unit

The SuperSPARC II processor implements split TLBs compatible to SPARC reference memory management unit (MMU). The SuperSPARC II MMU has a 64-entry translation lookaside buffer (TLB) which translates virtual to physical addresses for the Data Cache and a 16-entry translation lookaside buffer (TLB) which translates virtual to physical addresses for the Instruction Cache. The second-level page table pointer and the root pointer are cached to reduce TLB miss penalties. The SuperSPARC II processor implements one unified Root Pointer (PTP0), one Level-2 Page Table Pointer (PTP2) for the instruction cache TLB, and four PTP2's for the data cache TLB.

SuperSPARC supports a 64-entry unified translation lookaside buffer for both Instruction Cache and Data cache. In addition, it caches one Root Pointer (PTP0) and one Level-2 Page Table Pointer (PTP2).

A.4.1 Page Table Pointer

An Entry Type (ET) of 01 in the least significant two bits indicates that the entry type is a page table pointer. A valid page table entry (PTE) has an ET equal to 2 or 3.

Table A-23 The Page Table Pointer Format

PTP		ET	
31	2	1	0

A.4.2 Page Table Entry

Table A-24 The Page Table Entry Format

PPN		C	M	R	ACC		ET	
31	8	7	6	5	4	2	1	0

M: Modified bit ET: Entry Type

The page table entry in SuperSPARC II differs from that of SuperSPARC in two aspects.

The Modified bit of the page table entry is masked in the I-TLB during normal updates and ASI updates. In the D-TLB, the Modified bit is updated when a page is accessed for writing and the bit is clear. The bit may also be set or cleared through ASI access.

Support for dual-byte ordering in SuperSPARC II has led to the redefinition of PTE.ET= 11. In SuperSPARC, the ET=11 decoding was reserved. For SuperSPARC II, ET=11 indicates that the byte-ordering for this page should be reversed, or toggled, from the present PSR.DE bit value. This only applies when the page is accessed as data - instructions are always accessed with “big-endian” ordering.

When a PTE is fetched during a table walk and placed in the TLB, the ET=11 decoding will set the RBO bit for the D-TLB entry (it will be ignored by the I-TLB). This RBO bit can be accessed via ASI=0x6, SEL=0x3. See Dcache for more details.

Table A-25 Page Table Entry - Entry Type Format- D-TLB ONLY

ET	Entry Type
0	Invalid
1	Page Table Pointer
2	Page Table Entry
3	Page Table Entry, w/ RBO

A.4.3 MMU ASI

A.4.3.1 ASI 0x3 MMU Flush and Probe

The privileged load and store ASI instructions are used to flush entries from the MMU/TLB and to probe for entries in the MMU.

Table A-26 MMU Probe and Demap/Flush Format

VFPA	Rsvd	Type	Rsvd
31	11	10	7 0

The TYPE field specifies the extent of a demap size or the level of the entry probed.

Probe Operation:

A probe returns either a page table pointer or a page table entry or generates an error. A probe is accomplished through a Load ASI 0x3 instruction. ***SuperSPARC II can only probe D-TLB entries***.

Table A-27 MMU Probe Format

Type	Probe Object	Returned Data	Updates
0 (Page 4KB)	Level 3 Entry	Level 3 PTE or 0	
1 (Segment 256 KB)	Level 2 Entry	Level 2 PTE/PTP or 0	
2 (Region 16 MB)	Level 1 Entry	Level 1 PTE/PTP or 0	
3 (context 4 GB)	Level 0 Entry	Level 0 PTE/PTP or 0	
4 (Entire)	Level n Entry	Level n PTE or 0	
5-7 (Reserved)			

Demap/Flush Operation

A flush operation removes a PTE from the MMU/TLB. A flush is accomplished through a Store ASI 0x3 instruction. SuperSPARC II can flush both D-TLB and I-TLB entries.

Type	Flush Object	Invalidates
0 (Page 4KB)	Level 3 PTE	TLBE only
1 (Segment 256 KB)	Level 2 and 3 PTE’s	
2 (Region 16 MB)	Level 1, 2 and 3 PTE’s	PTP2, all TLBE’s
3 (context 4 GB)	Level 0,1,2 and 3 PTE’s	
4 (Entire)	All PTE’s	PTP0,PTP2, All TLBE’s
5-7 (Reserved)	-	

All four entries of SuperSPARC II D-TLB Level-2 PTPs are invalidated during a Type 3 (Context) Demap/Flush operation.

Illegal demaps (types 5-7) will be ignored internally. However, the demap will be broadcast to the system through DEMAP_. This behavior is the same as in SuperSPARC.

Note – A data_access_exception is generated on probe types 0x5-0x7

A.4.3.2 ASI 0x4 MMU Register Access

Ref MMU Register (MCNTL) VA=0000

Table A-28 Ref MMU Register:

IMPL	VER	RSVD	PF	RSVD	TC	AC	SE	BT
31	27	23	18		16	15	14	13
PE	**mb**	**SB**	**IE**	**DE**	**PSO**	**RSVD**	**NF**	**EN**
12	11	10	9	8	7	6	1	0

MCNTL_IMPL: Implementation number of the SuperSPARC II Processor. This field is read-only.

MCNTL_VER: Version number of the SuperSPARC II 1.0 processor. This field is read-only. This field will reflect any software changes specific to the SuperSPARC II processor.

Note – MCNTL.IMPL & MCNTL.VER: See SuperSPARC II ID's

MCNTL_mb: MBus Mode. This bit is masked. This bit is read-only in the SuperSPARC processor and reflected the CCRDY_ pin. The SuperSPARC II processor supports only VBus operations and therefore this bit is hard-wired to 0x0.

MCNTL_PF: Data Prefetcher: NO SUPPORT FOR SuperSPARC OR SuperSPARC II PROCESSORS.

Memory Context Table Pointer (MCTP) VA=0x0100

The context table pointer (MCTP) can be changed to point to a different context table. If the MCTP is changed, the cached page table pointers will be invalidated. However, SuperSPARC II, like SuperSPARC, does not automatically demap the tlb's on a context table pointer change.

Note – NO CHANGE FOR SuperSPARC II.

Memory Fault Status Register (MFSR) VA=0x0300

MMU Fault Status Register (MFSR) provides information for SuperSPARC II faults which are associated with the memory system and other internal error sources. The MFSR, along with the reported trap type, is used to distinguish between the various types of errors and faults that can occur.

Note – NO CHANGE FOR SuperSPARC II.

Memory Fault Address Register (MFAR) VA=0x0400

The Memory Fault Address Register (MFAR) records the virtual address of the fault reported in the Memory Fault Status Register (MFSR). This register is only valid when MFSR.FAV is set. Like SuperSPARC, instruction fault addresses are not recorded in the MFAR.

Note – NO CHANGE FOR SuperSPARC II.

Memory Shadow Fault Status Register (MSFSR) VA=0x1500

The MMU Shadow FSR Register (MSFSR) is identical to the MFSR but is used to record memory system errors while in Emulation mode.

The SuperSPARC II processor does not support Emulation and has ***no*** MSFSR Register. A DATA_ACCESS_EXCEPTION trap is generated if this register is accessed through ASI 0x4 operation.

A.4.3.3 ASI 0x5 and 0x6 MMU TLB (Page Descriptor Cache) access

The SuperSPARC II MMU entries are directly accessible with the ASI values of 0x05 and 0x06. SuperSPARC II supports ASI 0x5 for I-TLB access and ASI 0x6 for D-TLB access. The SuperSPARC processor uses ASI 0x6 for the unified I & D TLB.

Table A-29 MMU TLB (PDC) Address Format:

Reserved		TLB Entry		Rsv	Sel		Reserved	
31	18	17	12	11	10	8	7	0

The SEL field of the address format allows access to the TLB entries, cached root pointer, and cached level 2 page table pointers (PTP2's).

In ASI 0x5, the TLB Entry field bits [15:12] are used in TLB entry access to choose one of the sixteen I-TLB entries. The upper two bits are ignored.

In ASI 0x6, the entire TLB Entry field is used in TLB entry access to choose one of the sixty-four D-TLB entries. For PTP2 access, TLB Entry bits 13:12 are used to select one of the four D-PTP2's.

SEL=2 TLB Entry Physical Page Number

The cached PTE entries are identical to the PTE bit fields except for the V and LVL fields. In SuperSPARC II, the I-Cache M bit is masked. There are no changes for D-Cache page table entries.

Table A-30 Page Table Entry - (Sel=2) Format

PPN		**C**	**M**	**V**	**ACC**		**LVL**	
31	8	7	6	5	4	2	1	0

Note – The Cached PTE's are slightly different than the in-memory PTE format. The ET bits are replaced by the LVL bits. The referenced bit is replaced by the Valid bit.

SEL = 3 TLB Entry RBO (D-TLB only) and Lock Bits

Table A-31 MMU TLB Address - Sel 3 Format

Rsvd		RBO	Lock
31	2	1	0

The RBO bit is new to the D-TLB Sel=3 format. This bit indicates whether the page is of the same or opposite endianness to the default endianness specified by the PSR.DE bit. The RBO bit is not present in the I-TLB.

RBO: Reverse Byte Ordering

0: Default endianness - PSR.DE bit.

1: Reverse of default endianness - ~PSR.DE bit.

Note – Setting all lock and valid bits in ***both I TLB and D TLB*** can deadlock and is not recommended. Lock bits are cleared by hardware reset.

SEL=4 Cached Root Pointer - PTP0

To reduce the time required for each table-walk to access the PTE's, the SuperSPARC II MMU, similar to the SuperSPARC processor, caches one Level-0 PTP (shared between the I-TLB and D-TLB.)

PTP0 is the root-pointer for the process in execution. On every context switch this cached entry is invalidated, and the first table-walk for the new process will be used to cache in the new root-pointer.

Note – PTP0: Only bits [31:6] are written. Bits [5:2] are ignored.

The same root-pointer references both I-TLB and D-TLB table-walks.

SEL=5 and 6 Cached Level 2 Page Table Pointer and Virtual Address Tag

The SuperSPARC MMU caches one level-2 PTP for the unified TLB. The SuperSPARC II processor caches ***one*** PTP2 for the Instruction cache TLB and ***four*** PTP2's for the Data cache TLB. A TLB miss and PTP2 virtual tag match would allow the use of the associated PTP2, requiring only the new level-3 PTE to be fetched rather than having to perform the entire three level table-walk fetch operation.

Table A-32 PTP2 - SEL 5 Cached Level-2 Page Table Pointer Format

PTP2		V	
31	2	1	0

The PTP2 is stored in bits 31:2. Bits 1:0 are valid bits where 00 is invalid, 01 is valid, and 10 and 11 are not used.

Table A-33 PTP2 - SEL 6 Cached Level-2 PTP Virtual Address Tag Format

Level 2 Virtual Address		reserved	
31	18	17	0

The four D-PTP2 entries are selected through ASI 0x6 using TLB entry bits 13:12 of the MMU address format (see Table A-29).

The four cached D-TLB PTP2 entries use a pseudo LRU replacement logic to select a PTP2 entry to replace when a new PTP2 is brought in to the MMU for a TLB miss. If one or more of the four entries is invalid, then the replacement logic selects for replacement the first invalid entry encountered starting at entry '0' and progressing to entry '3.' If all the entries are valid the replacement logic selects the entry to be replaced using the information available in the used bits. The algorithm used is a limited history LRU policy, similar to the TLB replacement policy.

Note – SEL=5 PTP2: ONLY bits [31:4] are written. Bits [3:2] are ignored.

A.4.3.4 ASI 0x20 - 0x2f Reference MMU bypass mode

Same support as SuperSPARC. A bypass ASI is treated as a normal memory reference operation except during translation. The virtual address is equal to the physical address with the upper 4 bits of the 36 bit physical address equal to the ASI type.

PA[35:0] <- {ASI[3:0],VA[31:0]}

A.5 Instruction Cache

The SuperSPARC II processor internal instruction cache is a 20K-byte physical address cache. It is organized as a five-way set-associative cache of 64 sets. The line size is 64 bytes, divided in two half-lines of 32 bytes.

The cache is enabled by the MCNTL.IE bit of the MMU control register. The cache is disabled at power-on reset and remains disabled until MCNTL.IE bit is set.

At power-on reset, the contents of the instruction cache are undefined. It is the responsibility of the software to initialize the instruction cache by resetting the valid bits (Instruction Cache Flash Clear).

A.5.1 Instruction Prefetching

Instruction prefetching supplies instructions to the Instruction Queue (IQ). The IQ provides instructions for the pipeline to execute. The IQ comprises one twelve-word sequential instruction queue and one twelve-word branch target queue. The pipeline can consume up to three instructions per cycle from the IQ.

A.6 Data Cache

The SuperSPARC II Processor internal data cache is a 16K-byte, physically addressed cache. It is organized as a four-way-set-associative cache of 128 sets. The line size is 32 bytes.

At reset the contents of the data cache are undefined. It is the responsibility of the software to initialize the data cache by resetting the valid bits (using flash clear). After a watchdog reset, the contents of the data cache are unmodified.

A.6.1 Write-Through Mode (Vbus)

The SuperSPARC II processor cannot operate directly on the MBus(Copy-Back). The SuperSPARC II processor will always be used with the MXCC cache controller chip. When used with the MXCC, all stores are immediately written to the store buffer, and the store buffer will send them out to memory (write-through) as soon as resources are available.

A.6.2 Data Cache Tags ASI=0xe

A.6.2.1 Data cache Ptag Format

Since the data cache is write-through, the shared and dirty bits are not used in SuperSPARC II.

Table A-34 Data cache Ptag Format

Rsvd	V	Rsvd	D	Rsvd	S	Rsvd	Paddr
63	56	55	48	47	40	39	23 0

V: Valid. This bit indicates the line and its associated tag are valid. When this bit is cleared, the tag and the data contained in the line have no meaning. At reset valid bits are undefined.

D: Dirty. *Reserved for SuperSPARC II. (MBus -CopyBack Cache)*

S: Shared. *Reserved for SuperSPARC II. (MBus - CopyBack Cache)*

Note – 24 Ptags (paddr) bits and 1 Valid bit are read and written during normal and ASI operations. All other bits are padded with 0's.

A.6.3 Data Cache Access

Read cycle will be on the rising edge and Write/Snoop Read on the falling edge of the clock.

A.6.4 Data Cache ASI Control

A.6.4.1 ASI & Store Buffer Flush Operations

The d_unit includes an independent ASI controller. Whenever a LDA or STA instruction is issued by the instruction pipe, this ASI controller handles the execution of the instruction.

Before processing the actual load or store ASI, the ASI controller (dacpath) will normally flush the store buffer except in the following cases:

1) The store buffer is already empty (b_sbempty =1);

2) It is an i-unit internal ASI;

3) It is a Control Space Access ASI (0x02) to the MXCC;

4) It is an illegal ASI (in which case a fault will be reported to the i-unit, and it will flush the store buffer before it enters the trap handler).

The reasons behind flushing the store buffer are mainly to avoid complexity and because ASI operations usually come in bunches, so that the overhead of doing the flush is negligible.

A.7 Dual Byte Ordering

SPARC is a “big-endian” architecture, where the address of a double word, word or half-word is the address of its most significant byte. If we divide a 32-bit word of data into bytes, then the byte[0] translates to bits<31:24> and byte[3] to bits<7:0>.

The SuperSPARC II processor, in addition to supporting SPARC “big-endian” ordering, implements “little endian” byte ordering as well. For “little-endian,” byte[0] appears on bits <7:0>, and byte[3] to bits<31:24>. This alternate byte ordering only affects data accesses - instruction fetches and MMU page table accesses and demaps always are performed as “big-endian.”

Table A-35 Byte Order Data Formats - doubleword

Byte Position	big-endian	Float (LDDF/STDF) little-endian[1]	Integer (LDD/STD) little-endian
byte[0]	bits<63:56>	bits<7:0>	bits<39:32>
byte[1]	bits<55:48>	bits<15:8>	bits<47:40>
byte[2]	bits<47:40>	bits<23:16>	bits<55:48>
byte[3]	bits<39:32>	bits<31:24>	bits<63:56>
byte[4]	bits<31:24>	bits<39:32>	bits<7:0>
byte[5]	bits<23:16>	bits<47:40>	bits<15:8>
byte[6]	bits<15:8>	bits<55:48>	bits<23:16>
byte[7]	bits<7:0>	bits<63:56>	bits<31:24>

1. STDFQ is handled as an integer STD, since it stores a 32-bit virtual address and a 32-bit opcode.

These byte orderings are summarized in the above table, for a doubleword of data. Note that little-endian conversions are handled differently for floating-point and integer memory accesses. Floating-point LDDF/STDF operate on a single 64-bit number, and so little-endian conversions are done across the full 64-bit quantity. However, integer LDD/STDs actually operate on two 32-bit quantities. So, integer conversions between “big-endian” and “little-endian” formats do not cross a word boundary. Two separate conversions are made, one for each 32-bit word. This is done to be compatible with SPARC Version 9, which also supports dual-byte ordering.

The SuperSPARC II processor internally differentiates between big-endian and little-endian byte order formats in two ways. The first is the PSR.DE (bit 15), which indicates the overall “default endianness.” For SPARC “big-endian” ordering, DE=0; otherwise, for “little-endian” operation, DE=1. This establishes the data byte-ordering for a given process; since the PSR is saved during traps, it may change upon entry into a trap handler.

The byte-order of a specific access can also be toggled on a page-specific basis. This is accomplished by setting PTE.ET=11 for the page, in the PTE table in memory. When the MMU table-walks to bring in this page, it recognizes this case and sets the RBO bit in the corresponding TLB entry. In subsequent translations, the RBO bit toggles, or inverts the “endianness” of the present data access.

Table A-36 Dual Order Byte RBO Selection

PSR.DE	D-TLB.RBO	Byte Format
0	0	big-endian
0	1	little-endian
1	0	little-endian
1	1	big-endian

The D-TLB.RBO bit also forces the data access to be non-cacheable for SuperSPARC II’s on-chip Data Cache. The cache processes the access as a miss; a read produces a read-single request to the bus, and a write is directed towards the store buffer and bus without modifying the cache. However, external cacheability of this access is not affected by the D-TLB.RBO bit, but continues to be determined by the TLB’s normal cacheability bits. Thus, a cacheable read with D-TLB.RBO=1 would produce a cacheable-read-single bus transaction on the SuperSPARC II pins.

A.7.1 Data Conversion - Data Cache Aligners

All byte-ordering conversions are performed at the boundary between the SPARC IU/FPU registers and memory, as data is transferred between them. On a “little-endian” Read transaction, data is fetched from cache or memory, and its byte-order is converted from big-endian to little-endian only as the data is passed to the IU/FPU; if the access missed the Data cache, the fetched data is written into the Data cache without any byte conversion. Similarly, on a “little-endian” write, the data is converted from little-endian to big-endian format, and then written into the Data cache (on a cache hit) and to the store buffer and memory, all in “big-endian” format. All transactions are based on the PSR.DE bit and D-TLB.RBO bits as shown in the table below.

Table A-37 Little-endian operation - Aligners

Transaction	Data Path	PSR.DE ^ D-TLB.RBO = 1
Read Miss	Mem. to D-Cache	no change
Read Miss	Mem to iReg (Data Forwarding)	big-endian -> little-endian
Read Hit	D-Cache to iReg	big-endian -> little-endian
Write Hit	iReg to Store Buffer (Mem.), D-Cache	little-endian -> big-endian
Write Miss	iReg to Store Buffer (Mem.)	little-endian -> big-endian

^ Exclusive Or function of PSR_bit_15 and D-TLB-RBO_bit

A.8 Store Buffer

The SuperSPARC II processor store buffer is a fully-associative cache of eight double-word entries. The buffer functions to eliminate most of the performance penalties associated with write-through cache and copy-back (flush) operations. This depth is sufficient to hold sixteen registers, the number required to flush a register window.

The store buffer components (tags, data and control) are accessible via ASI spaces 0x30-0x32 for diagnostic purposes.

The store buffer is enabled by the MCNTL.SB bit. The store buffer is disabled at power-on reset and remains disabled until MCNTL.SB bit is set.

A.8.1 Store Buffer Flush

One condition that causes store buffer flushes has changed from SuperSPARC to SuperSPARC II.

In the SuperSPARC processor, a table walk causes store buffer flush only if the PTE of that tablewalk resides in the store buffer. If so, the store buffer is flushed until that PTE is no longer in the Store Buffer.

In the SuperSPARC II processor, a tablewalk causes all entries of the store buffer to be flushed.

Other causes of store buffer flushes remain the same between the SuperSPARC and the SuperSPARC II processors.

A.8.2 Store Buffer & Snoops

Similar to SuperSPARC, SuperSPARC II has no mechanism to snoop the store buffer.The reason is that any data residing in the store buffer *must* have the M-bit of the PTE set. So, in order for some other processor to write to that page, that processor must gain ownership of that page. And since DEMAPs also force store buffer to flush, store buffer consistency of data would be maintained. So, there is no need to snoop into the store buffer.

A.9 SuperSPARC II Testability Features

In order to ease the debugging and testing problems associated with very large and complex microprocessors, a number of testability features have been designed into SuperSPARC II. Some of these features are improvements over the SuperSPARC testability features and others are completely new. The goal is to achieve significantly enhanced diagnosability and testability, and, at the same time minimize the impact on SuperSPARC II performance.

The important changes from SuperSPARC include enhancing JTAG to support two-cycle scan-based timing (delay) fault testing, reducing the four scan domains on SuperSPARC into one single scan domain on SuperSPARC II, improving ATPG test and debug capabilities, adjusting the number and improving the randomness of BIST vectors, introducing stop/resume mode for chip and module debugging, replacing SuperSPARC emulation mode with enhanced boundary scan, and increasing the size of the data bus from eight bits to sixty-four in Sramtest mode.

The following sections briefly describe the functions of each testability feature on SuperSPARC II.

A.9.1 JTAG (with Enhancement)

JTAG is implemented on SuperSPARC II in full compliance with the IEEE1149.1 standard. All mandatory public instructions for the TAP controller are supported. In addition, a number of private instructions are also implemented for various functions.

One enhancement made to the JTAG TAP controller is to support the scan-based two-cycle delay fault tests. A two-cycle delay test requires that the TAP controller be able to stay in the CAPTURE state for two consecutive cycles, whereas the current IEEE 1149.1 allows only one cycle for this state. The SuperSPARC II TAP controller is designed such that when the JTAG DELAYTEST2 instruction is encountered in the JTAG instruction register, an additional CAPTURE state is added. This implementation is a superset of the IEEE 1149.1 specification.

The table below summarizes all JTAG instructions supported on SuperSPARC II.

Table A-38 SuperSPARC II JTAG Instructions

Instruction	**Function Description**
BSCAN_EXTEST	used for testing printed circuit board inter-connects
BSCAN_SAMPLE	used for sampling system operation
BSCAN_INTEST	used for testing SuperSPARC II mounted on printed circuit board
BSCAN_ISSUETCK	issues functional clocks in normal and enhanced boundary scan
BSCAN_STOP	stops functional clock to enter stop mode or enhanced boundary scan
SCAN_TEST	accesses internal scan chain for normal scan test
DELAY_TEST	accesses internal scan chain for one-cycle delay test
DELAY_TEST2	accesses internal scan chain for two-cycle delay test
SIGNATURE	accesses BIST signature register
RUN_BIST0	activates BIST
RUN_BIST1	activates debug BIST
SHOW_PLL	passes PLL clock to TDO pin

Table A-38 SuperSPARC II JTAG Instructions

Instruction	**Function Description**
CID	accesses component ID register
DIE_ID	accesses laser-programmable die tracking ID register
BYPASS	accesses bypass register

A.9.2 Full Internal Scan

Full-scan design is implemented on SuperSPARC II. All flipflops outside of the memory arrays are connected into a single scan chain. The scan chain plays a central role in many SuperSPARC II testability features. It is used for several purposes including conventional and two-cycle delay scan tests, stop mode, and built-in self test (BIST).

One important difference in SuperSPARC II scan design is that there is only one scan chain in the whole chip, instead of the four in the SuperSPARC design. The use of both falling-edge and rising-edge flip-flops on SuperSPARC prohibited a single scan chain since all flip-flops on a scan chain must operate on the same clock edge. SuperSPARC II is a rising-edge design. Also, currently available ATPG tools can handle the larger scan chain.

With a single scan chain, the special interdomain latches used on SuperSPARC to separate different scan domains can be eliminated, reducing the propagation delay overhead for paths that pass through these latches.

A.9.3 Scan-Based Two-Cycle Delay Test

A single scan chain also simplifies the implementation of the two-cycle delay test which allows scan-based timing and delay fault detection.

In addition to conventional scan based ATPG tests, another type of scan test is applied on SuperSPARC II. It is called the two-cycle delay test. A vector is shifted into the internal scan chain, and, instead of one functional clock, as in a conventional scan test, two functional clocks are issued. The second functional clock must have the desired chip cycle time. At the end of the second functional clock cycle, if an expected signal transition along a path is captured correctly at the ending flip-flop, then the path under

test meets desired cycle time. Otherwise, a timing or delay fault is detected. An ATPG tool specifically intended for scan-based two-cycle delay test is used to generate vectors for all critical paths between flip-flops.

A.9.4 Built-in Self Test (BIST)

SuperSPARC II BIST is scan-based. Pseudo-random vectors are generated by a random vector generator on chip and applied through the internal scan chain. All the output vectors are scanned into a signature register to generate a unique signature at the end of BIST. This signature is then read to determine if the chip passes BIST.

As on SuperSPARC, SuperSPARC II BIST covers all circuitry outside of the memory arrays. BIST can be activated in two ways, either through JTAG by entering the RUNBIST0 and RUNBIST1 instructions or through a SPARC instruction STA to 0x39. Both SuperSPARC II and SuperSPARC support two modes of BIST. However, in SuperSPARC II, the randomness of the BIST vectors is improved and the number of vectors applied is adjusted to make BIST more effective and practical.

The 17-bit linear feedback shift register (LFSR) that is used as a random vector generator on SuperSPARC is replaced with a 32-bit LFSR to avoid repetitive vector patterns, making the BIST vectors more random. The 31-bit LFSR used as BIST signature register is the same as on SuperSPARC.

Taking into account both BIST test time and test effectiveness, the long BIST with 64K vectors and short BIST with 1K vectors on SuperSPARC have been changed to a regular BIST with 16K vectors and a debug BIST with two vectors on SuperSPARC II. The debug BIST is intended to be used only for debugging purposes, and the regular BIST is to be used in production and field tests.

The correct signatures for SuperSPARC II BIST will be derived from gate level simulation and silicon debug.

A.9.5 Stop Mode

An important feature introduced on SuperSPARC II is stop mode. It is intended for chip level and SuperSPARC II-MXCC module level debugging on the tester. It provides full observability and controllability for all scannable flipflops on SuperSPARC II, and thus can be very helpful in debugging SuperSPARC II.

In stop mode, the internal SuperSPARC II clock, clk, can be stopped at any desired SuperSPARC II cycle. After clk is stopped, the chip enters internal scan mode through JTAG. The scan chain, which contains the state of all scannable flipflops in the cycle when stop mode is activated, is shifted out to the tester. Then, depending on the situation, either the same signals or a set of modified signals are shifted back into the scan chain. After the scan chain shifting is completed, stop mode is terminated and normal operation can be resumed. Since SuperSPARC II operates on VCK in normal operation and on TCK in scan operation, clk switches between VCK and TCK in stop mode operation.

Stop mode can be activated in two ways. The direct way is to assert the SuperSPARC II pin, STOP_. SuperSPARC II clock is stopped one VCK cycle after STOP_ is asserted. Another way to activate stop mode is through JTAG. When the JTAG instruction, BSCAN_STOP, is entered through the JTAG ports, SuperSPARC II clock is stopped one cycle after the JTAG instruction is updated in the JTAG instruction register. However, because of the discrepancy between VCK and TCK frequencies, it may be difficult to determine the exact cycle in which the SuperSPARC II clock is stopped. Depending on how stop mode is activated, stop mode can be exited by de-asserting the STOP_ pin or by returning the JTAG TAP controller to the reset state.

Stop mode can be enabled in every VCK cycle so that a test can be stepped through with all the internal flip-flop states known to the outside. For any VCK cycle, the cycle time can be adjusted on the tester to test for timing problems between cycles. This is also called clock-stretching, and can be very useful in debugging speed problems.

A.9.6 Boundary Scan

Boundary scan is used in testing printed circuit boards and the chips mounted on those boards. It is crucial in testing the SuperSPARC II-MXCC module. For SuperSPARC II, boundary scan is also used in the enhanced boundary scan mode for testing the SuperSPARC II chip at-speed on low-cost test stations.

Boundary scan cells in full compliance with the IEEE 1149.1 standard are used in all functional I/O pads. All boundary scan cells are connected into a shift register chain. All boundary scan operations are controlled through JTAG.

The basic SuperSPARC II boundary scan options are the same as on SuperSPARC.

A.9.7 Enhanced Boundary Scan

Another new SuperSPARC II testability feature is enhanced boundary scan. This feature is designed to replace part of the SuperSPARC emulation features.

Enhanced boundary scan is to be used together with SRAM test mode to run a functional test at full speed on a low-cost scan station. The goal is to be able to load a functional test program into the SuperSPARC II I-cache via boundary scan, and then start running the loaded program from the I-cache.

To load a test into the I-cache, the chip is put into SRAM test mode. Then data and address are provided at the I/O pins through boundary scan and are loaded into the I-cache. Once the entire program is loaded into the I-cache, a specially designed boot routine is run, and the program starts execution from a particular address.

Enhanced boundary scan involves switching SuperSPARC II's internal clock, clk, between TCK and VCK, and requires glitchless transitions between the two clocks. TCK is used in loading the test program and booting the chip after the program is loaded. VCK is used when the test program starts executing.

A.9.8 SRAM Test Mode

SRAM test mode allows the SuperSPARC II I-Cache and D-Cache to be tested with an SRAM tester. SuperSPARC II provides a full 64-bit data path in SRAM test mode in addition to the 8-bit interface used in SuperSPARC.

As before the low 17 address pins [16:0] are used to select the blocks in the cache. Address pin 17 selects between the 64-bit and the eight-bit data interface.

Addressing of SRAM blocks, column/row redundancy, and the read/write cycle time remain the same as before.

A.10 Self Monitoring Features

SuperSPARC II has similar debugging and performance monitoring capabilities as SuperSPARC. Main differences include the elimination of emulation mode, elimination of physical address breakpoints, addition of a scratch register, ability to count user or

supervisor instructions/cycles, and the expansion of the performance counters from 16 bits to 32 bits each. The following is a table highlighting the ASI changes from SuperSPARC to SuperSPARC II.

Table A-39 Changes in Performance and Breakpoint ASI's

ASI	SuperSPARC	SuperSPARC II
0x38	Breakpoint Registers	Breakpoint Registers
0x3a-0x39	BIST Diagnostics	BIST Diagnostics
0x3a-0x3f	reserved	reserved
0x40	**MTMP1**	**Kernel**
0x41	**MTMP2**	**reserved**
0x42-0x43	**reserved**	**reserved**
0x44	**Emulation Data In**	**reserved**
0x45	**reserved**	**reserved**
0x46	**Emulation Data Out**	**reserved**
0x47	**Emulation Exit PC**	**reserved**
0x48	**Emulation Exit nPC**	**CTRV_A**
0x49	**Emulation Counter Value**	**CTRV_B**
0x4a	**Emulation Counter Control**	**CTRC**
0x4b	**Emulation Counter Status**	**CTRS**
0x4c	ACTION	ACTION

A.10.1 Software Debugging Facilities (Breakpoints)

The SuperSPARC II processor provides the same breakpoint capabilities as the SuperSPARC processor with one exception: breakpoints may only be set on virtual addresses. This change also implies that the Breakpoint Value Register (BKV) and Breakpoint Mask Register (BKM) are thirty-two bits wide since virtual addresses are

thirty-two bits wide. SuperSPARC allowed physical address matching and thus had thirty-six bits for those registers. Though these registers are still double-word accesses in SuperSPARC II, the upper word is ignored.

Breakpoint Control Register (BKC)

Table A-40 Breakpoint Control Register

Rsvd	CSPACE	PAMD	CBFEN	CBKEN	DBFEN	DBREN	DBWEN
63 7	6	5	4	3	2	1	0

PAMD: Physical Address Match is reserved in SuperSPARC II.

Breakpoint Value Register

Table A-41 Breakpoint Value Register

reserved	BKV
63 0	31 0

Breakpoint Mask Register

Table A-42 Breakpoint Mask Register

reserved	BKM
63 0	31 0

A.10.2 SuperSPARC Remote Emulation Features (VICE)

The SuperSPARC processor provides facilities to observe and control processor execution from a remote device using the IEEE 1149.1 JTAG serial scan interface. SuperSPARC II does NOT support emulation. Thus the following SuperSPARC emulation registers do not exist in SuperSPARC II: emulation exit PC/nPC registers (ASI 0x47 & 0x48), MTMP1 and MTMP2 (ASI 0x40 and 0x41), MDIN (ASI 0x44), and MDOUT (ASI 0x46).

Note – NO SUPPORT IN SuperSPARC II PROCESSOR.

A.10.3 Kernel

SuperSPARC II provides a supervisor-only scratchpad register, Kernel. This 32-bit register is accessible through ASI 0x40. In SuperSPARC, this ASI is used for accessing the emulation register MTMP1.

Table A-43 ASI 0x40: Kernel

Kernel	
31	0

A.10.4 SuperSPARC II Performance Counters

The SuperSPARC II processor provides two 32-bit performance counters, CounterA and CounterB. Depending on the select in the Counter Breakpoint Control Register, the counters can be set to monitor user/kernel instruction/cycle count. In SuperSPARC, the instruction and cycle counts are 16-bit fields in the Counter Breakpoint Value Register accessed through ASI 0x49. SuperSPARC also did not differentiate between user and kernel activity.

ASI 0x48: CounterA

Table A-44 ASI 0x48: CounterA

CounterA	
31	0

CounterA is a 32-bit counter accessible through ASI 0x48.

ASI 0x49: CounterB

Table A-45 ASI 0x49: CounterB

CounterB	
31	0

CounterB is a 32-bit counter accessible through ASI 0x49.

ASI 0x4a: Counter Breakpoint Control Register (CTRC)

Table A-46 ASI 0x4a: Counter Breakpoint Control Register

Counter A					Counter B				
SelectA	**UA**	**KA**	**AEN**	**rsvd**	**SelectB**	**UB**	**KB**	**BEN**	**rsvd**
31 24	23	22	21	20 16	15 8	7	6	5	4 0

SelectA/SelectB: Selects what CounterA/CounterB will count.

0: Cycle Count

1: Instruction Count

2- 255: Reserved

UA/UB: User count enable. If this bit is set, the counter will decrement its value if/when the processor is in user mode.

KA/KB: Kernel count enable. If this bit is set, the counter will decrement its value if/when the processor is in supervisor mode.

AEN/BEN: Enable CounterA/CounterB.

Note – If both User and Kernel are enabled for a particular counter, the counter will count both. If neither User nor Kernel are enabled, then nothing will be counted.

ASI 0x4b: Counter Breakpoint Status Register (CTRS)

Table A-47 ASI 0x4b: Counter Breakpoint Status Register (CTRS)

Rsvd	ZACIS	ZBCIS
31 2	1	0

ZACIS: CounterA Zero Count Interrupt Status. This bit is set when CounterA reaches zero and an interrupt is *generated and taken*. In SuperSPARC, this bit indicates the cycle count interrupt status.

ZBCIS: CounterB Zero Count Interrupt Status. This bit is set when CounterB reaches zero and an interrupt is *generated and taken*. In SuperSPARC, this bit indicates the cycle count interrupt status.

The CTRS register is cleared on read, power-on reset, and watchdog reset.

ASI 0x4c: ACTION Register

Table A-48 ASI 0x4c: ACTION Register

rsv		MIX	BCIPL	
63	13	12	11	8

E_CBK	E_ZIC	E_DBK	E_ZCC	I_CBK	I_ZIC	I_DBK	I_ZCC
7	6	5	4	3	2	1	0

The ACTION register is similar to the SuperSPARC ACTION register.

E_ZAC: Enable ESB pin to strobe on a zero instruction count.

E_ZBC: Enable ESB pin to strobe on a zero cycle count.

I_ZAC: Enable interrupt on zero instruction count..

I_ZBC: Enable interrupt on zero cycle count.

In SuperSPARC and SuperSPARC II, the strobe does not occur unless a fault or interrupt is also enabled.

A.11 External Monitors (Pipepins)

Observability will be increased to 48 PIPE pins for the SuperSPARC II processor. The SuperSPARC II processor has twelve dedicated PIPE pins. Four sets of monitoring information are available through the PIPE pins selectable through two PIPESEL pins. The PIPE pins are detailed in Appendix C.

A.12 SuperSPARC II Pins

Table A-49 New SuperSPARC II Pins

Pin	Direction	Description
ADDR20B	O	Memory bank selection (2Meg E-Cache support). Active low of ADDR[20]
STOP	I	Stop mode for scan chain dump H = Stop internal clock L = Normal operation
PIPESEL[1:0]	I	Select 1 of 4 Pipe combinations
PIPE[11:0]	O	Monitor 12 internal states of the SuperSPARC II Processor

Table A-50 SuperSPARC II - SuperSPARC Deleted Pins

Pin	Direction	Description
ARDY	I	Indicates that the system logic is prepared to accept another address or bus cycle.
BUSREQ	I	Bus request for use by SuperSPARC, in both modes. SuperSPARC II is always in bus request mode.
CCRDY	I	Bus mode selection. VBus or MBus.
OWNER	I	Owner of a cache line currently being requested on the bus with a CR or CRI.
SHARED	I	MBus mode only. Copy back of any cache line being requested by Mbus.
ESB	0	Execution strobe output. In SuperSPARC II ESB is a pipe signal.

Note: ARDY, BUSREQ, CCRDY, OWNER and SHARED pins were MBus support pins that are now part of the SPARE Pins.

A.13 STP1021 - STP1021A Changes

A.13.1 Demap Timing

SuperSPARC II STP1021A demap cycle time is one cycle longer than that of Voyager 1.x. This is true for Internal and Externally initiated Demaps.

A.13.2 Aligned I-Cache Addressing

SuperSPARC II STP1021A will send out double-word aligned instruction cache addresses to the Vbus. That is, ADDR[2:0]=0.

Note – In SuperSPARC II STP1021 the address increments as 0,4,8,c.
In SuperSPARC 1021A the address increments as 0,8,0,8

A.13.3 Vbus Timing

SuperSPARC II STP1021 relinquishes the Vbus two cycles after a bus error on a swap; SuperSPARC II STP1021A will relinquish the Vbus the cycle after the bus error.

A.13.4 Thermal Diode

In SuperSPARC II STP1021A, SPARE1 and SPARE6 are connected by a thermal diode which can be used for junction temperature measurements. In SuperSPARC II STP1021 the two pins are unconnected

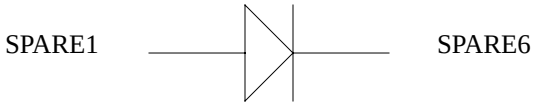


Figure A-1 Thermal Diode between SPARE1 and SPARE6

A.14 SuperSPARC ID's

Table A-51 SuperSPARC Id's

Control Register	bit size	SuperSPARC 3.X	SuperSPARC 4.X	SuperSPARC 5.X
PSR VER	8 bits	0x40	0x40	0x40
MCNTL IMPL	4 bits	0x0	0x0	0x0
MCNTL VER	4 bits	0x1	0x2	0x3
FSR VER	3 bits	0x0	0x0	0x0
JTAG CID	32 bits	0x1000402F	0x2000402F	0x3000402F

A.15 SuperSPARC II ID's

Table A-52 SuperSPARC II Id's

Control Register	bit size	SuperSPARC II 1021 (1.X)	SuperSPARC II 1021A (2.X)
PSR VER	8 bits	0x40	0x40
MCNTL IMPL	4 bits	0x0	0x0
MCNTL VER	4 bits	0x8	0xA
FSR VER	3 bits	0x0	0x0
JTAG CID	32 bits	0x0001602F	0x1001602F

Notes: