The SuperSPARC™ Microprocessor

Technical White Paper

Sun Microsystems
Computer Corporation
A Sun Microsystems, Inc. Business

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Contents

Introduction to the SuperSPARC Microprocessor 1

The SuperSPARC™ microprocessor is a highly integrated superscalar SPARC® microprocessor, fully compatible with the SPARC version 8 architecture [SI 1992]. This paper provides an overview of its architecture, internal operation, and capabilities. The processor contains on a single chip an integer unit, a double-precision floating-point unit, fully consistent instruction and data caches, and a SPARC Reference Memory Management Unit. In addition, a dual-mode bus interface is included that supports either the SPARC standard (MBus) [SI 1990] or an interface optimized for connection to a companion second-level cache controller chip. The chip, with an optional second-level cache controller, is targeted at a wide range of systems from uniprocessor desktop machines to multiprocessing file and compute servers.

Both the microprocessor (3.1 million transistors) and the second-level cache controller (2.2 million transistors) are fabricated with Texas Instruments' Epic2b process: a 0.8 micron, three-layer metal, BiCMOS process. These integrated circuits have die sizes of 16x16mm and 14x14mm respectively.

This paper discusses the design strategy to produce a microprocessor suitable for a range of systems and the required trade-offs before focusing on the main functional blocks. Specifically, the operation of the integer pipeline, the floating point unit, and memory hierarchy are discussed. A chapter is devoted to the design and operation of the second-level cache controller.

Design Philosophy 2

Compute-bound execution rate is a product of three factors: the number of compiler-generated instructions in an execution trace (NI), the sustained rate of instructions completed per cycle (IPC), and the clock rate made possible by the underlying integrated circuit technology [Patterson 1989].

High performance requires advances on all three fronts. All designs will benefit from better compilers and high clock rate. However, cost-effective systems cannot rely exclusively on high clock rate designs. To benefit from economies of scale, mainstream (rather than exotic) technology must be used. SuperSPARC-based systems provide high performance at moderate clock rates by optimizing the IPC. This presents design challenges for managing the processor pipeline and memory hierarchy. Average memory access time must be reduced and the number of instructions issued per cycle must rise; a superscalar architecture is required.

Design Criteria

Superscalar

Known bottlenecks to issuing multiple instructions include control dependencies, data dependencies, the number of ports to processor registers, the number of ports to the memory hierarchy, and the number of floating point pipelines. Our own studies suggest that many applications can benefit from a processor that issues three instructions per cycle.

Table 2-1 Instruction Mix for Integer and Floating Point Applications

Instruction Class	Integer	Floating Point
Integer Arithmetic	50%	25%
FP Arithmetic	0%	30%
Loads	17%	25%
Stores	8%	15%
Branches	25%	5%

Design Choices

The above instruction mix presents many challenges for a three-scalar design executing code at the peak rate. Integer applications will access data memory every cycle, process branches every two cycles, and update memory every three cycles. Floating-point applications will access data memory, and execute floating point and integer arithmetic instructions every cycle. To mitigate these expected performance challenges, the following architectural decisions were made:

- On-chip caches
- A wide instruction fetch (128 bits)
- Extensive forwarding paths
- Overlapped memory access with floating point dispatch
- Write buffer to decouple the processor from store completion

Figure 2-1 shows the functional blocks of the microprocessor.

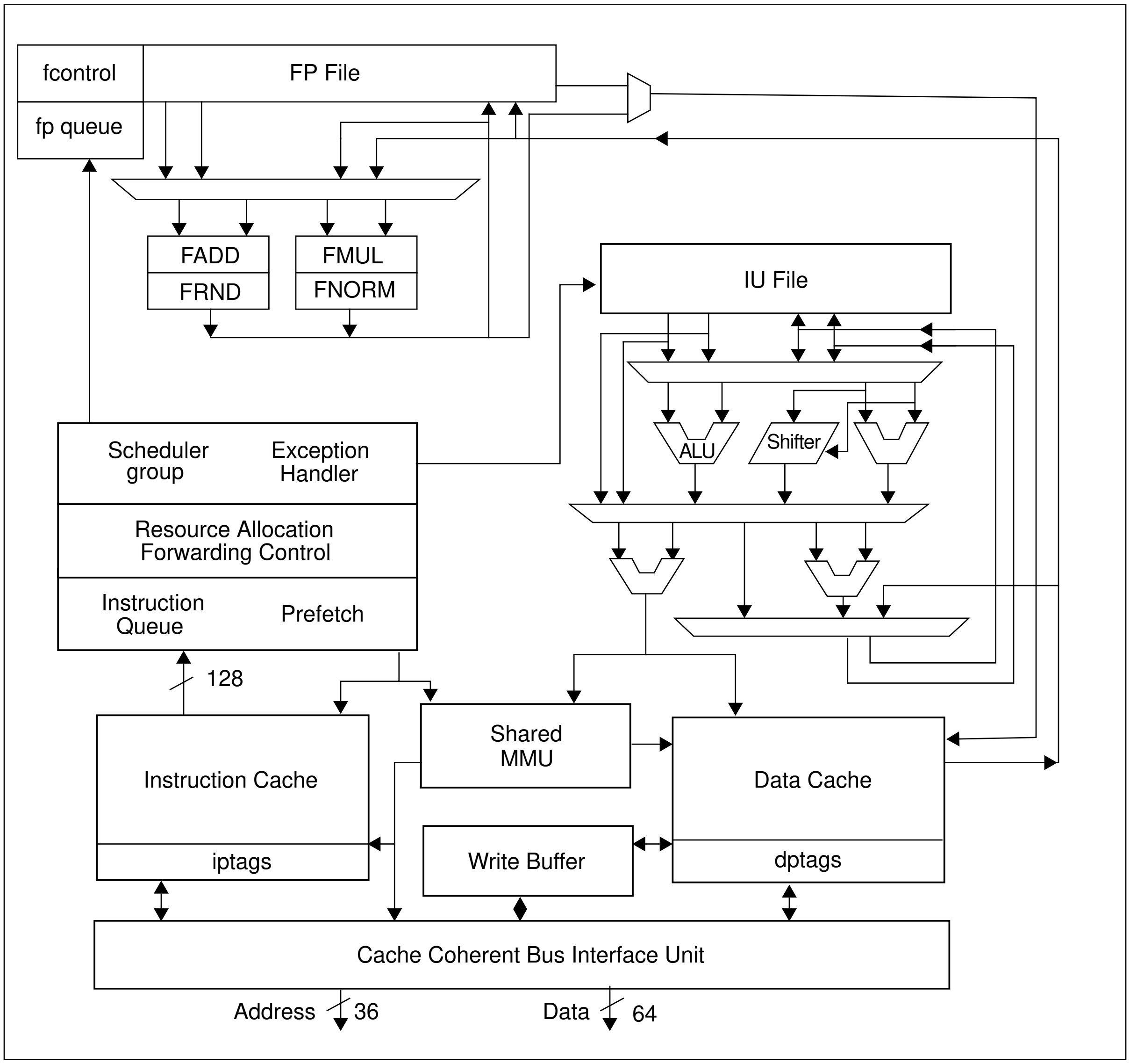

Figure 2-1 Functional Block Diagram

Microarchitecture 3

Integer Unit

This section describe the operation of the integer pipeline arithmetic logic units (ALUs), and strategies to deal with branches and exceptions.

Pipeline Organization

The SuperSPARC processor [ISSCC 1992], [Compcon 1992] uses a four-cycle integer pipeline. Each cycle has two-phases. The first cycle (F0, F1) fetches up to four instructions from the first-level instruction cache into an instruction queue. There are three phases of decode (D0, D1, D2), two stages of execution (EO, EI) and a writeback stage (WB).

Table 3-1 Pipeline stages

Stage	**Activity**
F0	I-cache ram access I-cache TLB lookup
F1	I-cache match detect 4 instructions sent to the iqueue
D0	Issue 1,2 or 3 instructions Select register file indices for LD/ST address registers
D1	Read Rfile for LD/ST address registers Resource allocation for ALU instructions Evaluate branch target address

Table 3-1 Pipeline stages (Continued)

Stage	Activity
D2	Read Rfile for ALU operands Add LD/ST address registers to form effective address
E0	1st stage of ALU D cache RAM access D cache TLB lookup FP instruction dispatch
E1	2nd stage of ALU D cache match detect Load data available Resolve exceptions
WB	Write back into the Rfile Retire store into the store buffer

The operation of the pipeline is illustrated by the execution of a load instruction with a forwarding path in Figure 3-1. The pipeline stages are detailed in Table 3-1.

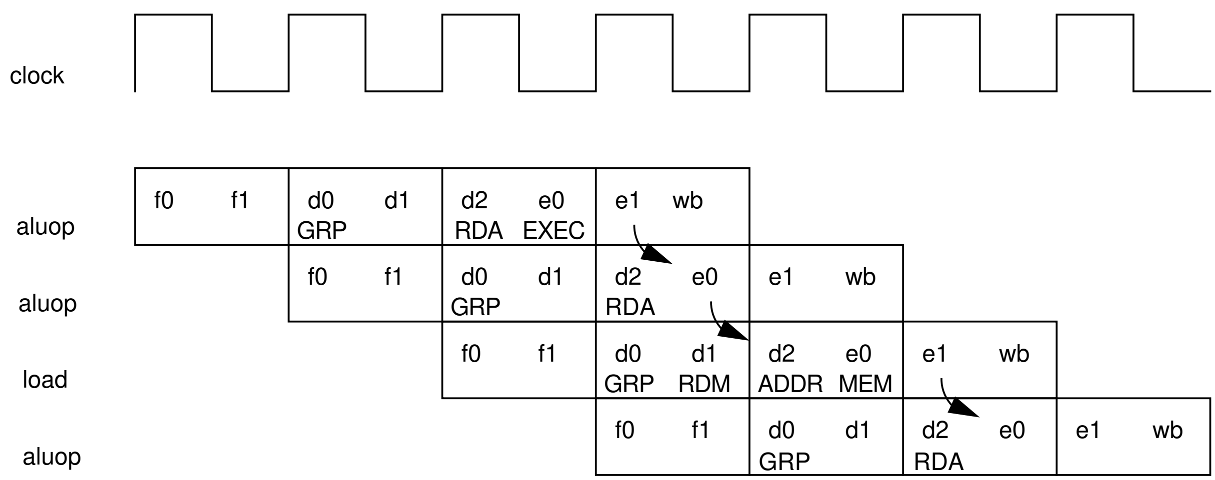

Figure 3-1 Pipelined Execution of Load with Forwarding

The floating-point (FP) pipeline is tightly coupled to the integer pipeline. An operation may be started every cycle; latency of most FP operations is three cycles. In the E0 phase, one FP arithmetic instruction is selected for execution and its operands are read during E1. Two stages of execution latency are required for the double-precision FP adder and FP multiplier. The first cycle of the adder examines exponents, aligns mantissas, and produces a result. The first cycle of the multiplier computes and adds partial products. Independent second stages round and normalize the result of the respective units. Forwarding paths are provided to chain results of one FP operation into the source of a subsequent operation.

Instruction Issue Strategy

Three candidate instructions from the selected instruction pre-fetch queue are presented to instruction grouping logic (D0). Opcodes are decoded and intra/intergroup dependencies are evaluated. Up to four different instruction groups can be actively processed in each machine cycle. Certain instructions (for example, save, restore, integer multiply/divide, control registers, and address space identifier (ASI) memory references) always are executed as a single instruction group to simplify the implementation. SuperSPARC can issue and execute three instructions every cycle subject to the following high-level constraints (per group):

- Maximum of two integer results
- Maximum of one data memory reference
- Maximum of one floating point arithmetic instruction
- Terminate group after each control transfer

The SuperSPARC microprocessor maintains its performance in the presence of data dependencies, which are prevalent in real applications by supporting:

- Cascades — dependent instructions in the same group
- Forwarding — dependent instructions in consecutive groups

ALU results can be forwarded or cascaded to a following ALU, store, branch, or address instruction. Load results can be forwarded to an ALU instruction.

Branch Strategy

Two instruction queues exist: one for untaken branches (sequential instructions) and one for taken branches (target instructions). Both queues can be filled because four instructions can be fetched per cycle while only a maximum of three can be executed. In figures 3-2 and 3-3, target instructions are referred to by T1, T2, and so on, sequential instructions are denoted as S1, S2. The delay instruction is DI; C1 refers to the certainty instruction stream.

To speed execution, the SuperSPARC processor can group an integer-compare instruction with a dependent conditional branch. Figure 3-2 shows the pipelined execution of a simple, taken, conditional branch.

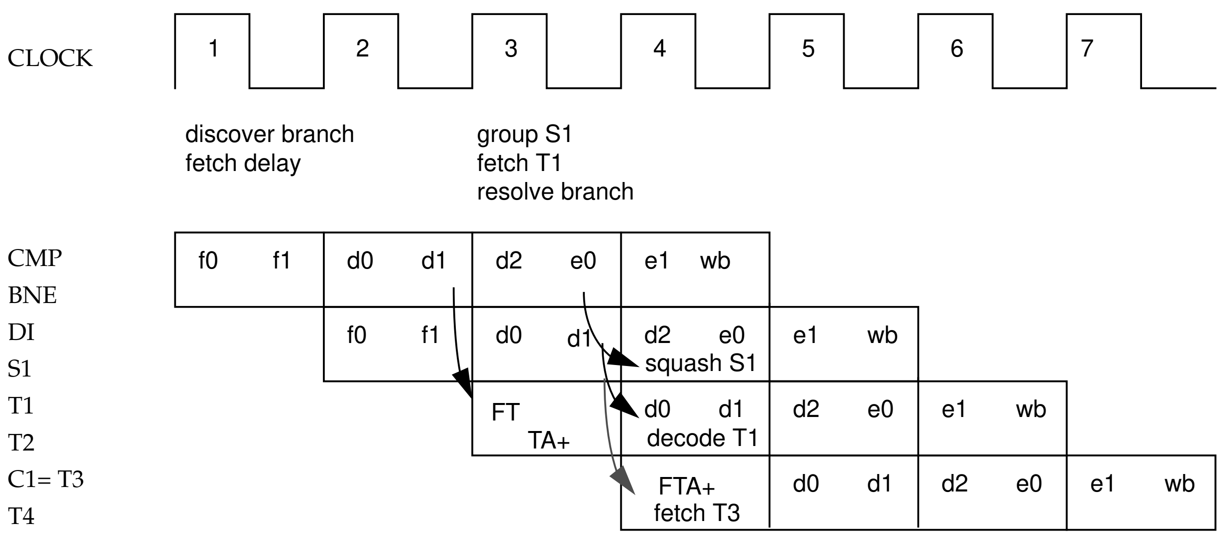

Figure 3-2 Taken Conditional Branch

The processor statically predicts that all branches are taken. When a control transfer instruction (CTI) relative to the program counter (PC) is issued, its delay instruction is fetched concurrently. During D1, the target address is computed. As the branch enters D2, the target instruction stream is fetched. The fetch completes as the delay instruction advances to D1 and the branch (and compare) enter E0. The compare computes new integer condition codes in E0 and the branch direction is resolved. Figure 3-3 shows the pipelined execution of a simple, untaken conditional branch.

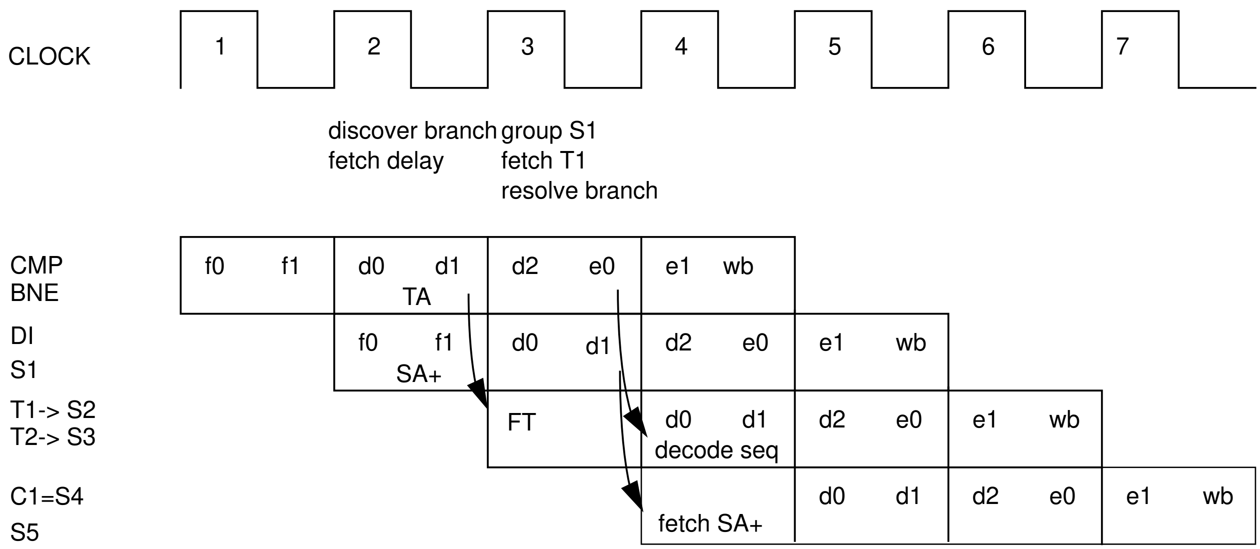

Figure 3-3 Untaken Conditional Branch

When a CTI is taken, all sequential path instructions (grouped together with the delay instruction) are invalidated. When a CTI is untaken, sequential path instructions remain valid and the target instructions fetched are discarded. This scheme never introduces a pipeline bubble for either branch path. The PCs and pre-fetch PCs for *both* directions of branch paths are precomputed. One pair is selected. The SuperSPARC branch implementation can execute untaken branches somewhat more efficiently than taken branches. Due to normal code scheduling restrictions, this difference in peak performance between taken and untaken branches will be less pronounced.

Exception Strategy

The SuperSPARC processor implements a precise exception model [Smith 1982]. At any given time there can be up to 9 instructions in the integer unit (IU) pipeline and 4 more in the floating point queue. Exceptions and the instruction that caused them propagate through the IU pipeline. They are resolved in the execute stage before their results can modify visible state in the register file, control registers, or memory model.

Floating-Point Unit

The floating-point unit (FPU) is a pipelined floating-point processor that conforms to ANSI/IEEE 754-1985 and SPARC V8 floating point architecture specifications. It consists of a floating-point controller (FPC) and two independent pipelines: the FADDER and the FMULTIPLIER. The FPU can operate both on single- and double-precision floating point numbers, normal or subnormal. It performs integer multiply and integer divide instructions as well.

Floating-Point Controller

The floating point controller (FPC) is tightly coupled to the integer pipeline and is capable of executing a floating-point memory event and a floating-point operation in the same cycle. The FPC implements the interface between the integer pipeline and the floating-point core. It contains a register file and a floating-point queue, and handles all floating-point exceptions.

There are two types of floating-point instructions: floating point operations (FPOPs) and floating-point events (FPEVENTs). The FPOPs consist of floating-point operations like add, multiply, convert and so on. FPEVENT's are executed by the FPU but do not enter the FPU queue. They include load/store to floating-point registers, load/store to floating-point status register, store floating-point queue, integer multiply, and integer divide.

The FPU pipeline is shown in Table 3-2

Table 3-2 Floating-Point Pipeline

FRD	FM/FA	FN/FR	FWB
Decode and Read	Execute multiply or add	Normalization and rounding	Write back to FP register file

All floating point instructions are issued by the integer unit in the E0 stage of the IU pipeline. Once issued, the floating-point instructions proceed through the FPU pipeline. The floating-point pipeline stalls the integer pipeline in a few situations: the floating-point queue will become full after several long latency floating-point arithmetic instructions are encountered in close proximity, or when an FPEVENT occurs that is dependent on an FPOP result.

Forwarding paths are provided to chain the result of one FPOP to source operands of a subsequent FPOP without stalling for a floating-point register write. Similarly, an FPEVENT LOAD can forward data to FPOP source operands and an FPOP result can forward data to an FPEVENT STORE.

All floating-point instructions start in order and complete in order. They are executed in one of the two independent units: the FADDER or the FMULTIPLIER. These instructions are held in a first-in-first-out (FIFO) queue that holds up to four entries. Each entry can hold a 32-bit FPOP instruction and a 32-bit FPOP address.

The floating point register file is made of 32 registers of 32 bits each. It is organized as 16 double words (64-bit wide) to optimize double-precision performance. Each word can be accessed separately, however. The register files can be accessed through three read and two write ports, used as follows: two read ports for the FPOP operands, one read port for storing FPEVENTs, one write port for the FPOP result, and one write port for loading FPEVENTs. Each port can sink or source a double-precision operand.

A floating-point exception remains pending until another floating-point operation or FPEVENT is requested by the integer unit. It will be reported to the subsequent floating-point instruction at that time.

Floating-Point Adder

The FADDER performs the following operations: addition, subtraction, format conversions, comparison, absolute value, and negation. It fully supports subnormal numbers, either as operands or as results, without any speed penalty. The ALU consists of two pipeline stages, mantissa addition and normalization/rounding. The Adder is fully pipelined. It can begin a single or double-precision add every cycle.

Floating-Point Multiplier

The FMULTIPLIER performs multiplication, division, square root, integer multiplication, and integer division. Subnormal numbers are fully supported in the hardware; there are no unfinished operations. The multiplier is fully pipelined. It can begin a double- or single-precision multiply every cycle. Division and square root are non-pipelined. Like the FADDER, the FMULTIPLIER is composed of two stages. Square root and division are performed in this stage by using the multiplier over multiple cycles. The second stage performs normalization and IEEE rounding. Subnormal outputs are correctly handled in this stage without software intervention.

Performance

Table 3-3 assumes normal floating point operands and results. The additional penalty for subnormal numbers is nil for FADDER operations, and no more than five cycles for FMULTIPLIER operations.

Table 3-3 Execution Times

Operation	**Throughput (Cycles)**	**Latency (Cycles)**
FADDER operations	1	1
Multiply	1	3
Divide (single precision)	4	6
Divide (double precision)	7	9
Square root (single precision)	6	8
Square root (double precision)	10	12
Integer multiply	4	4
Integer divide	18	18

Memory Hierarchy

Superscalar performance can only be maintained if the integer unit can be supplied with the appropriate instructions and data; this is done by the memory hierarchy. The (20 Kbytes) on-chip instruction cache, fetches up to four instructions per cycle without memory wait states. It plus the 16 Kbyte on-chip data cache (which can handle one 64-bit load or store in each cycle) minimize pipeline bubbles. Memory management and protection is provided by an on-chip SPARC Reference MMU. It provides physical addresses for both caches, enabling them to support hardware cache consistency. In low-cost systems, these caches interact directly with the SPARCstation™ 10 system memory bus (Mbus). However, for high-end uniprocessor and multiprocessor systems, the SuperSPARC processor also supports a large (1 Megabyte) second-level, external cache.

Cache and MMU Operation

The SuperSPARC processor has multi-way associative caches to provide high cache hit rates. Multiple code blocks and data structures can easily reside together in the on-chip caches. Both caches use a pseudo least recently used (LRU) replacement algorithm. The 20-Kbyte instruction cache is five-way set-associative; the 16-Kbyte data cache is four-way set-associative. As a result, each set for both caches contains 4 Kbytes, which matches the minimum MMU page size. Table 3-4 shows some of the parameters of the instruction and data caches.

Table 3-4 Instruction and Data Cache Parameters

Parameter	Instruction Cache	Data Cache
Organization	20 Kbyte five-way set associative	16 Kbyte four-way set-associative
Page size	4 Kbytes	4 Kbytes
Line size	64-bit (32-bit subblock size)	32-bit
Access	128-bit wide fetch path	64-bit r/w data path
Hit rate (SPEC 92)	98%	90%
Replacement	Pseudo LRU	Pseudo LRU

This organization also enables the caches to be fully physically addressed, without compromising performance. Since the caches can be addressed entirely with page-index address bits, which are unaffected by virtual-physical address translations, they can be accessed in parallel with the MMU. The remaining physical address bits arrive later from the MMU, and are used to select the proper set and determine a hit or miss for each cache.

Physical caches greatly simplify cache coherency for multiprocessing (MP) systems. Since both the processor and memory consistency snoop operations use physical addresses, both can be accomplished with common tags. Anti-aliasing software restrictions and excessive cache flushing, common problems with virtual cache machines, are eliminated. Moreover, SuperSPARC tags may be accessed twice per cycle for the processor and memory snoop operations, avoiding the traditional performance impacts of hardware cache coherency.

This organization enables the caches to be accessed in *parallel* with the MMU, since no additional physical address bits are required to index into each cache array. The remaining physical address bits arrive later from the MMU, and are used to select the proper set and determine a hit or miss. This organization is shown in Figure 3-4.

The cache tags are single ported, but are shared in time to allow for non-interfering memory consistency *snoop* operations. The cache arrays are also time shared, allowing for simultaneous read and fill operations.

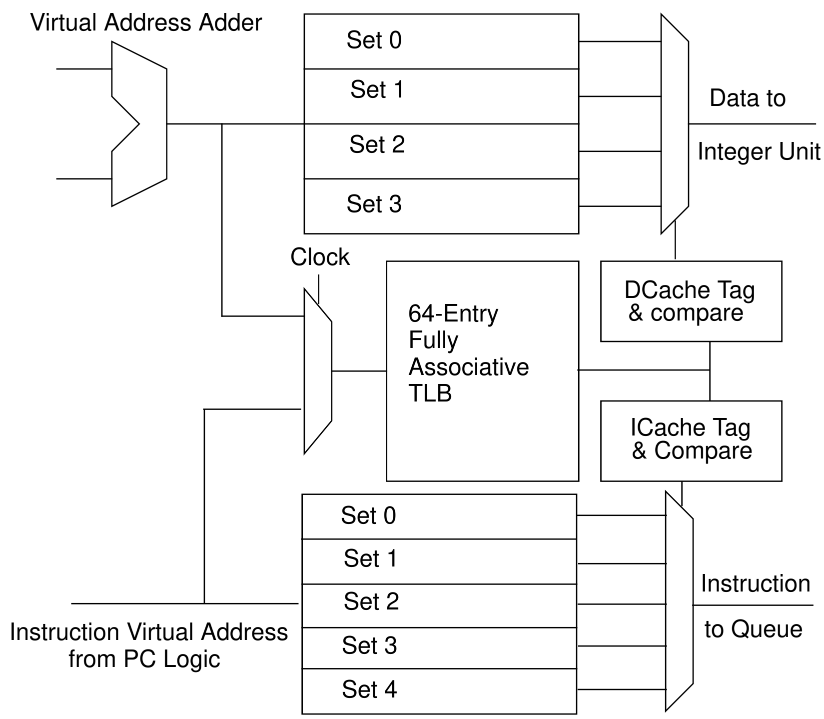

Figure 3-4 Cache/MMU Organization

Bus Operation

The interface to an optional, second-level cache has been optimized for high bandwidth burst transfers of data between the CPU and cache. The processor may send a new address to the cache RAM and cache controller every processor cycle with no idle cycles, except when the bus direction must be changed. The transfer rate is controlled by the external cache controller, allowing for a variety of different RAM configurations and cache speeds.

The SuperCache Controller implements a one Mbyte, direct-mapped SRAM-based cache. It supports a 128 byte line size, with 32 byte subblocks, and obtains a 99% hit rate in SPEC'92 benchmarks. Block-zero and Block-copy operations are supported in hardware. The cache tag array is implemented with 33 Kbytes of RAM, on the SuperCache controller chip. The cache's data array is made from eight, 128 Kbytes x 9 fully registered SRAMs, which support pipelined accesses and parity. A copyback, MOESI interface is maintained to the memory system.

The CPU is isolated from the system bus in two very important ways by the SuperCache controller, beyond simply providing fast data transfers. First, it allows the processor to run asynchronously to the system bus, freeing the CPU's accesses, it greatly reduces the SuperSPARC microprocessor's memory bandwidth needs. This allows a substantial number of processors to be connected in a single machine.

The SuperSPARC microprocessor implements write-through coherency when an external cache is present, writing all store data directly to the external cache. In addition to simplifying the cache controller's coherency hardware, this also avoids write-miss penalties normally incurred in copyback caches. In this configuration, the store buffer holds up to 8 buffered writes, both cachable and noncachable.

Direct MBus Interface Operation

The SPARC memory bus (MBus) is a 64-bit multiplexed bus protocol designed for interconnecting a number of processors in a small system (typically a single board uniprocessor or multiprocessor system). The consistency protocol provides for modified, owned, exclusive, shared, and invalid states (the MOESI protocol [Sweazey 1986]). The SuperSPARC microprocessor direct MBus interface fully supports this protocol. A basic MBus cachable memory reference transfers 32 bytes of data.

The MBus interface uses a copyback, write allocate protocol. The on-chip store buffer is used as a copyback buffer, enabling new data to be brought on the chip before modified old data is transferred to memory. This copyback buffer is kept coherent as well. The SuperSPARC processor responds to *snoop* transactions at cycle "A+3", or three cycles after the address is received.

MBus supports a range of error and retry responses to handle different system requirements. The *relinquish and retry* protocol enables bus deadlock cases to be resolved, as well as enabling disconnect/reconnect transactions to enable efficient transfers to slow peripheral devices.

External Cache Interface

The interface to an optional, second-level cache has been optimized for high bandwidth burst transfer of data between the CPU and the second-level cache. The processor may send a new address to the cache RAM and cache controller every cycle with no idle cycles, unless the bus direction must be changed. The interface provides 64 bidirectional data lines, and a 36-bit physical address bus. All transfers are fully pipelined.

The transfer rate is controlled by the cache controller, allowing for a variety of different RAM configurations. Normal operation is for a fully registered SRAM, as described in the next section. The processor drives the SRAM address inputs and output-enable directly (as well as SRAM write-enable signals) when directed to do so by the cache controller.

Overlapped read and write cycles are supported by this packet-switched interface. In particular, any number of reads may occur while a write-miss is in progress. Conversely, any number of writes may occur while a read miss is in progress. Write-burst transactions are also supported, requiring only a single "tag check" transaction for any number of queued stores within a single cache line.

In this bus mode, the consistency mechanism is *invalidation.* The external cache must include a copy of all information in the internal cache *(inclusion).* Full byte-parity support is provided by the interface. The processor both checks and generates parity when enabled.

Implementation 4

Process

The SuperSPARC microprocessor and SuperCache controller are manufactured using the Texas Instruments Epic2b process. This is a BiCMOS process with a 0.8 μ minimum feature size. It has three layers of metal which, with a salicided silicon layer, enables four-layer interconnect. A local interconnect system is used within the RAMs to yield a small RAM cell, enabling the large cache sizes. Precision implanted resistors are available, and used for such functions as sense-amp biasing, phased locked loop (PLL) control, and output switching time control. Laser programmable fuses enable higher yield through circuit redundancy.

The epitaxial architecture of the process has buried high-conductivity wells (both P and N) on a low-conductivity substrate. This contrasts with the high-conductivity substrate found in most CMOS processes. There are two advantages to this construction. The obvious one is a greater level of latchup protection. Another important benefit is the ability to electrically isolate power supplies on the chip (a high-conductivity substrate would short them). This feature is especially important to prevent noise-induced threshold shifts in the input circuits, and to allow stable PLL operation. Electrostatic discharge (ESD) protection is provided using a combined NPN-MOS structure to achieve greater than 4 KV protection. All I/O pins have ESD protection circuits, and there are also protection structures across and between power supply domains.

Clocking

The SuperSPARC microprocessor and its SuperCache controller utilize PLLs to act as zero-delay clock buffers, and to regenerate the clock duty-cycle. The PLL has a common architecture but takes advantage of the BiCMOS process to produce excellent performance. The phase and frequency detector (PFD) uses BiCMOS circuitry to improve logical functionality and to reduce the reset pulses, thus achieving an increased operating range. An external capacitor is used to control the voltage control oscillator (VCO). The VCO control is moderated by precision resistors, reducing process variability. A five-stage CMOS oscillator with both rising and falling edge control is used to minimize chaotic oscillations. The oscillator is run at double the clock speed and then divided by two to guarantee a good clock duty cycle. The oscillator (each oscillator on the SuperCache controller) is fed with its own isolated power supply, preventing noise from other parts of the chip from affecting PLL operation. Filtering is placed on the SuperSPARC modules to prevent system noise from affecting the oscillators.

Clock distribution with low skew is achieved by using balanced clock distribution paths up to key points, then using delay-matched local buffers to regenerate sharp edges. BiCMOS buffers give low gate-delays into high loads and thus help minimize clock skew across the chip.

Second-Level Cache Controller 5

Introduction

To achieve high performance, the processor needs a fast clock and a low number of cycles per instruction (CPI). However, even with a low CPI, memory latency can degrade the performance significantly, especially with superscalar microprocessors. Caches are used to reduce the impact of long memory latency. With current VLSI technology, a smaller on-chip first-level cache, say 32 Kbytes can be put inside the processor chip. However, a simple computation reveals that even with a small first-level cache, the CPI can be degraded significantly. For example, assume a CPU with 0.75 CPI and a 3% cache miss rate, the memory latency 10 cycles away, and the memory reference rate at 1.33 references per instruction. In this example, the cache miss penalty is 0.4 CPI, and the degradation to the performance is about 35%. By implementing a large second-level cache, the effective memory latency can be reduced due to both a lower miss rate on the second-level cache and the fact that it is much closer to the processor than the main memory.

Multiple Processors

Increasing the number of instructions per cycle executed by the system can be achieved by having multiple processors as well as using superscalar pipelines. The degree of multiprocessing depends on the available multiprocessing bus bandwidth, and the bus bandwidth requirement per processor. With a large second-level cache per processor, the bus bandwidth requirement per processor is greatly reduced due to a much lower miss rate. A given bus can thus support more processors.

The SuperCache controller [Compcon 1992a] is designed to isolate the CPU and external cache from the rest of the system. The processor can then operate at a faster clock speed than the system.

Bus Support

The SuperCache supports the MBus multiprocessor bus, a circuit-switch bus. It implements a shared-write invalidation cache consistency protocol. The SuperCache controller can handle multiple outstanding operations from the SuperSPARC microprocessor with its internal FIFOs. It also includes special hardware to support Block Copy/Block Zero, and performance monitoring.

Processor Subsystem

The SuperSPARC microprocessor is connected to SuperCache controller and external cache through the processor bus, which consists of a 72-bit data bus (64-bit data and 8-bit parity), a 36-bit address bus, and the other 25 control/command signals. The processor bus can support pipelined access to the external cache in burst mode for both read and write. In a burst-read operation, 32 bytes of data can be streamed to the processor in four cycles after the initial two cycles of latency. In a burst-write operation, CPU can write into external cache with a speed of 8 bytes per cycle after the initial three cycles of latency.

A typical processor subsystem configurations for an MBus based system is shown in Figure 5-1.

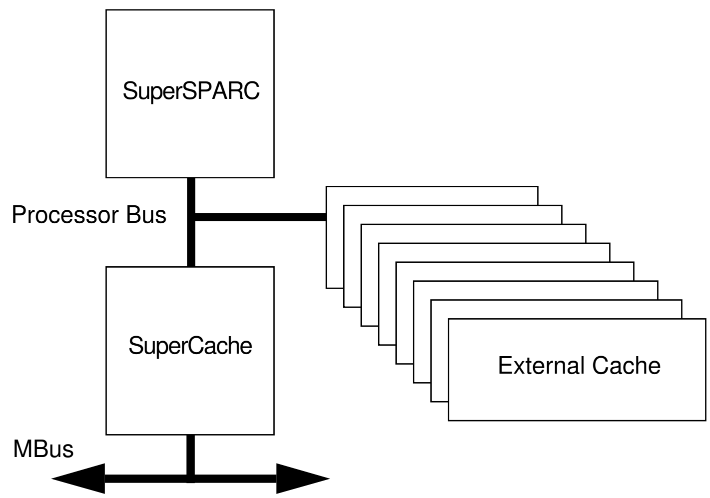

Figure 5-1 Processor-subsystem in MBus Mode

External Second-Level Cache

The SuperSPARC microprocessor has on-chip a 20 Kbyte instruction cache and a 16 Kbyte data cache. When connected to the SuperCache controller, the processor first-level caches work in write-through mode.

The external cache is a direct-mapped, copyback cache. The SuperCache can support up to 1 megabyte of external cache in the MBus configuration. Sub-blocking is used to reduce the size of the external cache directory inside the SuperCache controller. The sub-block size is 32 bytes for the MBus configuration. Four sub-blocks are implemented per block. The cache consistency protocol is maintained, and data is transferred on sub-block basis.

1 Mbyte x 8 or 1 Mbyte x 9 SRAM chips that support pipelined access are used for implementing the external cache. During a read, there are two cycles of latency for latching the address and then latching the data inside the SRAM chips before the data is driven to the processor bus.

Features

The SuperCache controller provides an asynchronous interface between the SuperSPARC microprocessor and system buses so that the processor subsystem can run with a higher clock frequency.

One read miss and one write miss can be handled at any given time. That is, when a processor read (or write) access incurs a miss, the controller can still allow the processor to access the external cache for write (or read) accesses until and including the occurrence of a write (or read) miss. After the second miss, no more write or read accesses are allowed until one miss is resolved.

Multiple shared write requests from the processor are possible, provided they are addressed to different sub-blocks in external cache. In a multiprocessor system, a write to a shared sub-block requires more cycles due to the cache consistency protocol. The ability to handle multiple pending write requests improves the write throughput from processor.

Prefetching is supported in order to reduce effective memory latency during sequential accesses. The prefetch is triggered on a processor burst-read access when its next sequential sub-block, bounded by the block boundary, is not in external cache. Only one prefetch can be outstanding at any given time. The SuperCache controller can handle prefetch and other operations at the same time. The prefetch mechanism can be disabled.

Block Copy and Block Zero are supported with stream hardware. A sub-block can be transferred from one memory location to another through the stream data register in the controller. A stream operation is triggered when the source (or the destination) address is loaded into the stream source (or destination) address register. The SuperCache controller can handle stream operations and other operations at the same time. Stream operations can only be invoked in the supervisor mode.

The SuperCache controller implements an external cache performance monitor with two counters. One keeps track of reference count and the other keeps track of cache miss count.

References

A

[SI 1992] SPARC International, Inc. *The SPARC Architecture Manual - Version 8*, Prentice-Hall, Englewood Cliffs, New Jersey. 1992.

[SI 1990] SPARC International, Inc. *The SPARC MBus Interface Specification*, SPARC International, Menlo Park, California, 1990.

[Pattserson 1989] D. A. Patterson, J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Palo Alto, California, 1989.

[ISSCC 1992] Abu-Nofal, Nanda, et. al, *Three Million Transistor Microprocessor*, 1992 International Solid State Circuit Conference proceedings, IEEE.

[Smith 1982] J. E. Smith and A. R. Pleszkun, *Implementing Precise Interrupts in Pipelined Processors*, IEEE Transactions on Computers, May 1982, pages 562-573.

[Compcon 1992] G. Blanck, S. Kreuger, *The SuperSPARC Microprocessor*, Compcon92, IEEE, Los Alamitos, California, 1992.

[Sweazey 1986] P. Sweazey, A. J. Smith, *A Class of Compatible Cache-Consistency Protocols and Their Support by the IEEE Future Bus*, Proc. 13th International Symposium on Computer Architecture, Tokyo, Japan, 1986.

[Compcon 1992a] J. H. Chang et. al, *A Second Level Cache Controller for a Superscalar SPARC Processor*, Compcon92, IEEE, Los Almitas, California, 1992.

Sun Microsystems
Computer Corporation

A Sun Microsystems, Inc. Business

For U.S. Sales Office locations, call: 800 821-4643
In California: 800 821-4642

Australia: (02) 413 2666
Belgium: +32 2 759 38 11
Canada: 416 477-6745
Finland: +358-0-502 27 00
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0

Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 3221-7021
Korea: 822-563-8700
Latin America: 415 688-9464
The Netherlands: 033 501234

New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (01) 825 71 11

Taiwan: 2-514-0567
UK: 0276 20444
Elsewhere in the world, call
Corporate Headquarters:
415 960-1300
Intercontinental Sales:
415 688-9000

Printed in USA

4/92 FE344-0/30K