microSPARC-IIep Megacell Reference

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300

Part Number: 806-1955-01
July 1999

Please
Recycle

Contents

Figures

Tables

Preface

This book, *microSPARC-IIep Megacell Reference*, is one of a four-manual documentation set for the microSPARC-II technology. The other three manuals are:

- *Multiprocessor SPARC™ Architecture Simulator (MPSAS) Programmer's Guide*, which explains the facilities for creating or modifying MPSAS modules, or otherwise extending MPSAS.
- *Multiprocessor SPARC Architecture Simulator (MPSAS) User's Guide*, which describes how to use MPSAS and its associated programs.
- *microSPARC-IIep Validation Catalog*, which describes a suite of validation tests for the microSPARC-IIep technology

Organization of This Book

This book is intended for designers who need to build the microSPARC-IIep megacells. It contains the following chapters and appendixes:

Chapter 1, *Requirements* describes the microSPARC-IIep megacells' requirements.

Chapter 2, *SRAM* discusses the microSPARC-IIep SRAM megacells.

Chapter 3, *TLB CAM/SRAM* discusses the Translation Lookaside Buffer (TLB) Content Addressable Memory(CAM) /Static Random Access Memory (SRAM) megacells.

Chapter 4, *IOTLB CAM/SRAM* describes the Input/Output Translation Lookaside Buffer (IOTLB) Content Addressable Memory(CAM) /Static Random Access Memory (SRAM) megacells.

Chapter 5, *256 x 64-bit Read-Only Memory (ROM)* describes the read-only memory circuit.

Chapter 6, *136 x 32-bit Register File*describes the larger of the two microSPARC-IIep register files.

Chapter 7, *16 x 64 Register File* describes the smaller of the two microSPARC-IIep register files.

Prerequisite Knowledge

It is assumed that you are familiar with programming in the C language in the UNIX® environment and that you have a basic familiarity with computer architecture, in particular the SPARC architecture. For further information, see the list of documents in the following section.

Related Books and References

The following documents contain material that further explains or clarifies information presented in this guide.

The SPARC Architecture Manual/Version 8 by David Weaver, Prentice Hall; ISBN: 0138250014

The C Programming Language by Brian W. Kernighan and Dennis M. Ritchie, Prentice Hall; ISBN: 0131103628

The Annotated C++ Reference Manual by Margaret Ellis and Bjarne Stroustrup

Typographic Conventions

TABLE P-1 describes the typographic conventions used in this book.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, instructions, files, and directories; on-screen computer output; email addresses; URLs	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have m`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, section titles in cross-references, new words or terms, or emphasized words	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.
<>	A bit number or colon-separated range of bit numbers within a field; bit 0 is the least significant bit.	`WB_VECTOR`<15:0>

Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals by using this program.

For a list of documents and how to order them, see the catalog section of the SunExpress™ Internet site at `http://www.sun.com/sunexpress`.

Sun Documentation Online

The `docs.sun.com` Web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`.

Disclaimer

The information in this manual is subject to change and will be revised from time to time. For up-to-date information, contact your Sun representative.

CHAPTER 1

Requirements

This chapter lists requirements that apply to all megacells:

- Terminals
- Conditions and procedures for simulation
- Miscellaneous requirements, including power terminals and capacitance

1.1 Terminals

All megacell terminals must adhere to the following requirements:

- Signal terminals on Metal 1, Metal 2 (preferred)
- Power terminals on Metal 3 (preferred)
- Clock terminal(s) *must* be on Metal 3 (tbd)
- No terminals on any layers other than Metal 1, Metal 2, or Metal 3
- All terminals must be on the edges of the megacell bounding box
- All terminals must be named exactly as specified

1.2 Simulation Conditions and Procedures

All megacells must be simulated under the following conditions and with the following procedures.

1. All megacell circuit simulations must be performed with Hspice, using the vendor-supplied Spice models. Worst-case performance simulations must use the slow process Spice model, a junction temperature of 109˚C, and a supply voltage determined by megacell designers but within the guidelines described below.

2. Guardbanding of 15% must be applied to all worst-case simulated delays, setup times, and hold times. This requirement can be met in one of two ways: all worst-case simulated delays, setup times, and hold times must be adjusted in a positive direction by 15%; or specified megacell timing parameters must be adjusted in a negative direction by 15% before simulation and the simulation results themselves must satisfy these derived parameters.

3. For simulations before megacell Final Acceptance, automated parasitic extraction with Timemill and an agreed-upon extraction rules file must be used. Intermediate simulations may use hand-estimated parasitics or other high-confidence estimations, but their accuracy must be considered preliminary.

4. A supply rail differential of 3.0V (vdd 3V, vss 0.0V) is guaranteed at the slow process, high temperature (109˚C) corner at the power terminals of the megacells. Internal IR drops must be calculated by megacell designers to find rail differentials at the transistors. No other rail separations are guaranteed at other conditions.

5. All megacell outputs should drive their specified loads no slower than a 1 ns rise and fall time, measured between the 10% and 90% points of the transition.

6. All timing parameters are measured between the 50% points of the signal transitions in question.

7. Fujitsu extract and qualify Motive pin level timing for each Megacell from layout.

8. Sun will provide rtl model to describe the functionality of the megacell. Functional and corner case test vector will also be provided.

9. For each megacell, timing numbers are included. Timing numbers are specified for the block before layout, with zero clock skew. Global clock skew is analyzed and matched upon completion of layout.

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

1.3 Other Requirements

1. Power terminals must allow the megacell to be placed and rotated any way on the chip, with global power distribution being in only one dimension.

2. Very large numbers of power terminals are not desirable; it is preferable that large numbers of terminals be grouped into no more than about 10 per side of the megacell.

3. Default maximum input capacitances for all input signal terminals is 100 fF. Exceptions without negotiation include the clock input(s), scan enable input, and reset input if any.

CHAPTER 2

SRAM

This chapter describes SRAM modules with respect to the following areas:

- Functional description, including data SRAM and tag SRAM descriptions, switching characteristics, and timing and clock input
- I/O signal description
- Area and power dissipation
- Layout
- Initialization and abnormal conditions
- Testability, including detailed functional description and timing notes
- Implementation notes

Please refer to Chapter 1 (Introduction) for information, guidelines, and requirements that apply to all megacells.

2.1 Functional Description

The microSPARC-IIep processor requires four single port, synchronous (clocked) SRAM modules: the `mc_icache`, `mc_itag`, `mc_dcache`, and `mc_dtag`. Two major types of SRAM are used: `mc_icache` and `mc_dcache` use Data SRAMs, while the `mc_itag` and `mc_dtag` use Tag SRAMs. The major difference between the two types of SRAMs are that the Tag SRAMs have built-in compare and protection check functions.

This chapter presents a “generic” SRAM description, then describes each of the four target modules separately. You should be aware that only one basic SRAM design can be used to generate all of the four modules, so that in the future, alternate organizations can easily be created (within reasonable characterization limits).

In this section, all the SRAM modules are described individually. TABLE 2-1 shows the required SRAM organizations. The valid bit is implemented only as part of the Tag SRAMs.The Tag SRAMs also contain Protection and Context bits.

TABLE 2-1 SRAM Organizations

Module	Organization	Ain (bits)	Di/Do (bits)
`mc_icache`	2048x64	11	64
`mc_dcache`	1024x(8x8)	10	8x8
`mc_itag` (including VB)	512x(19+8+5+1)	9	33
`mc_dtag` (including VB)	512x(19+8+5+1)	9	33

The minimum cycle time for all SRAMs is 14.2 ns. All SRAMs must allow back-to-back read operations or write operations or any combination of both, to any sequence of addresses.

2.1.1 Data SRAM

There are two target modules for this type of SRAM: the `mc_icache` (16 Kbytes) and the `mc_dcache` (8 Kbytes). We describe the organization and operation for each of these modules. Testability of the Data SRAMs is discussed in *Testability* on page 27.

The Data SRAMs have three cycle types under system operation:

- **Read cycle**. The addressed contents of the required location are read out on to the output ports (*mod*_`do`) of the megacell.
- **Write cycle.** Input data is written to the addressed location and is also bypassed directly to the output data bus (the signals *mod*_wle and *mod*_be are both asserted).
- **Bypass cycle**. In this cycle (only the signal *mod*_`be` is asserted), the input bus is bypassed onto the output bus.

The control signal encoding for each of these cycle types is shown in the following tables.

TABLE 2-2 is a logical view of the SRAM cycle, input encoding. It shows that for a Read cycle, the write enables and bypass enables must be turned off. It further shows that for a Write cycle both the write enables and the bypass enables are asserted. The Bypass only, cycle is recognized, when only the bypass enable is asserted. One observation to be made here is that the output port of the SRAMs are always used.

TABLE 2-2 Data SRAM Cycle Type Logical Encoding

Cycle Type	mod_we	mod_be	mod_do
Read	0	0	Data
Write(+Bypass)	1	1	Data
Bypass only	0	1	Data

Now, to speed up the access time of the SRAM and to reduce the amount of decoding to be done within the SRAM, this input encoding is changed slightly. This is what will be implemented in the actual megacells. This encoding is shown in TABLE 2-3.

TABLE 2-3 Data SRAM Cycle Type Actual Encoding

Cycle Type	mod_wle	mod_be	mod_power
Read	0	0	0
Write(+Bypass)	1	1	0
Bypass only	0	1	0
Output= 64'b 1	0	0	0
F-Powerdown	0	0	1
Not Allowed	0	1	1
Not Allowed	1	0	1
Not Allowed	1	1	1

Here, the input pins of the megacell are *mod*_wle and *mod*_be, where *mod*_wl stands for Wordline enable and *mod*_be stands for Bypass enable. The bypass enable differentiates between full and partial writes of the SRAM (in the Data Cache SRAM). The P-Powerdown (Partial Powerdown) mode should reduce the power dissipated to less than 25% of the normal operating power of the SRAMs, whereas the F-Powerdown mode (Full Powerdown) should reduce operating power to at least 10% of the normal operating power. However, no combination of inputs and state should result in excessive current or damage to the SRAMs, even if the SRAM powers up without clocks running.

Note – *mod*_wle, *mod*_power and *mod*_be are registered inputs to the SRAM; their states shown in TABLE 2-3 are the states of the SRAM flip-flops holding their values. For all other input signal combinations not shown in the table, the Data SRAM behavior is a don't care.

The mc_icache Cell

FIGURE 2-1 illustrates the mc_icache cell described in this section.

Cell Name: mc_icache

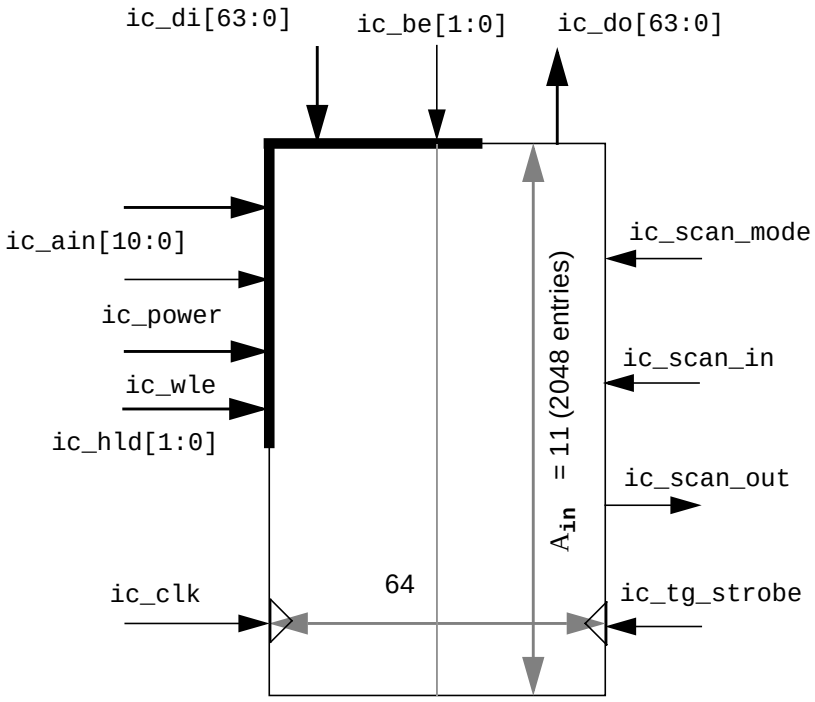

FIGURE 2-1 mc_icache Block Diagram

As shown in the figure, the ic_ain, ic_di,ic_be, and ic_wle inputs are registered, so they are available at the inputs of the SRAM one cycle prior to operation (read or write).

A Read operation requires ic_ain, ic_be, ic_power, and ic_wle (driven active) setup to the rising edge of ic_clk (start of the Read cycle). Following the rising edge of the ic_clk, ic_do will contain the contents of the location addressed by ic_ain, after a time defined to be the access time of the target SRAM.

A Bypass cycle requires only the ic_be input be asserted with enough setup to the rising edge of ic_clk. After the rising edge of ic_clk, ic_do will contain the value latched from ic_di at the beginning of the present Bypass cycle.

A Write operation requires ic_ain, ic_di, ic_wle, (driven active), and ic_be (driven active) set up to the rising edge of ic_clk (start of the Write cycle). During the write cycle, the ic_do output must equal ic_di (since ic_be is asserted during the Write cycle). The mc_icache megacell has two bypass enables and thus has the capability to write 32-bit words into the SRAM, depending on which of the two is asserted at the beginning of the cycle. ic_be[1] writes ic_di[31:0], and ic_be[0] writes ic_di[63:32] into the addressed location.

The input ic_hld [1: 0], when asserted, causes the ic_di latch to retain its previously latched value until this signal is deasserted. Again, ic_hld[1] controls ic_di[31:0], and ic_be[0] controls ic_di[63:32].

The testing of this module is explained in *Testability* on page 27. Note that internally the registers holding ic_ain, ic_we, and ic_be are connected in a scan chain (shift register) when ic_scan_mode is asserted. The chain's input is ic_scan_in; output is ic_scan_out. No input combination to the mc_dcache should cause multiple drivers on the output bus. The SRAMs should also not power up in a state such that the output bus is being driven by multiple drivers, even if no clocks are issued for an arbitrary period of time.

The mc_dcache Cell

FIGURE 2-2 illustrates the mc_dcache cell described in this section.

Cell Name: mc_dcache

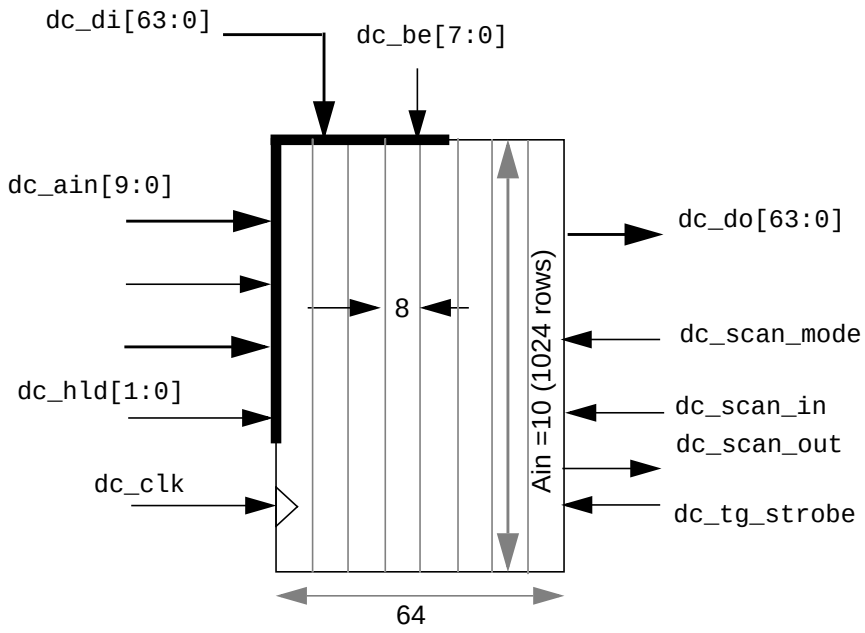

FIGURE 2-2 mc_dcache Block Diagram

Similar to the mc_icache, the mc_dcache's dc_ain, dc_di, dc_be, and dc_wle inputs are registered inputs set up to the rising edge of dc_clk. One distinguishing feature of the mc_dcache vs. the mc_icache is *byte-writes*. This feature is implemented by having a separate dc_be input for each byte. dc_di is aligned externally in the correct manner prior to a write operation.

Note – Writing different bytes of the mc_dcache is controlled by different bypass enables.

---

dc_di[63:56] is controlled by dc_be[0],
dc_di[55:48] is controlled by dc_be[1],
dc_di[47:40] is controlled by dc_be[2],
dc_di[39:32] is controlled by dc_be[3],
dc_di[31:24] is controlled by dc_be[4],
dc_di[23:16] is controlled by dc_be[5],
dc_di[15:8] is controlled by dc_be[6], and
dc_di[7:0] is controlled by dc_be[7].

A Read operation requires dc_ain, dc_be (deasserted), and dc_wle (asserted) be set up to the rising edge of dc_clk (start of the Read cycle). Following the rising edge of dc_clk, dc_do contains the contents of the location addressed by dc_ain after a time defined to be the access time of the target SRAM.

Note – During a Read operation, all 64 bits are read out simultaneously.

A Write operation requires dc_ain, dc_di, dc_wle (asserted), and dc_be (asserted) set up to the rising edge of dc_clk (start of the Write cycle). During the write cycle, the dc_do output must equal the latched value of dc_di (since dc_be is asserted as the Write cycle is defined).

A Bypass cycle requires dc_be to be active and dc_wle to be deasserted, during which the contents of dc_di are available *(after it is registered in the SRAM)* at dc_do, after a time equal to the access time of that particular SRAM. Note that external control logic can assert any one, two, four, or eight bits of dc_be during a write operation.

The input dc_hld [1: 0], when asserted, causes the dc_di latch to retain its previously latched value until this signal is deasserted. Again, dc_hld[1] controls dc_di[31:0], and dc_hld[0] controls dc_di[63:32].

As in the mc_icache megacell, the mc_dcache megacell has a separate *mod*_power signal to support powerdown features. This signal *(mod*_power) is enabled synchronously with the clock to disable the Data SRAMs until the next clock edge, when this control signal is reenabled. No data should be lost when the SRAMs come back from the powerdown mode.

2.1.2 Tag SRAMs

The Tag SRAMs are more complicated than the Data SRAMs, although the Core RAM design can be the same. The differences are described below.

- The Tag SRAMs have a Valid bit associated with each word of the SRAM. The Valid bit has a separate write enable input. (Please note that since bypass enable doesn't exist for the Tag SRAM, we sometimes use the term *write enable* to clarify our intent, even though the signal is named with an abbreviation of *mod*_be, to be consistent with the terminology of the previous Data SRAMs).
- The Tag SRAMs have an 8-bit Context field associated with each word of the SRAM. The Context bits have a separate write enable input. This write enable is shared with the Access Protection field.
- The Tag SRAMs also have a (5) bit field associated with every word of the SRAM. These are Access Protection bits and are used to check different privileges of the cache lines. These bits are written into the cache at the same time the Context bits are written in. The *mod*_protection_match signal is deasserted if an access

violation is detected. This *mod*_protection_match signal is merged with the final *mod*_hit/*mod*_miss output of the megacell to indicate a miss on that particular cache line.

- The Tag SRAMs also have a comparator and some extra logic to perform the protection check, the tag compare, and the various flush match operations. These checks are performed at the end of every Read cycle.
- The mc_itag and the mc_dtag megacells are identical copies of each other and thus can be replicated in layout. They are shown separately here only to distinguish their signal names.

The Tag SRAMs have two cycle types under system operation:

- **Read cycle**. A read access is performed, then the protection plus the tag check are done after the access. The flush checks are also performed in parallel. As far as the Tag SRAM is concerned, there is no separate flush cycle.
- **Write cycle**. Input data are written to the addressed location, depending on which write enable bit is asserted. Output data are ignored.

The input control signal encoding for each of these cycle types is shown in the TABLE 2-4.

TABLE 2-4 Tag SRAM Cycle Type Encoding

Cycle Type	Read	Write
mod_be mod_be_vb mod_be_cntx	All of these are 0	Any of these is 1
mod_wle	1	1
mod_do	Data at mod_ain	x

The mc_itag Cell

FIGURE 2-3 illustrate the mc_itag cell described in this section.

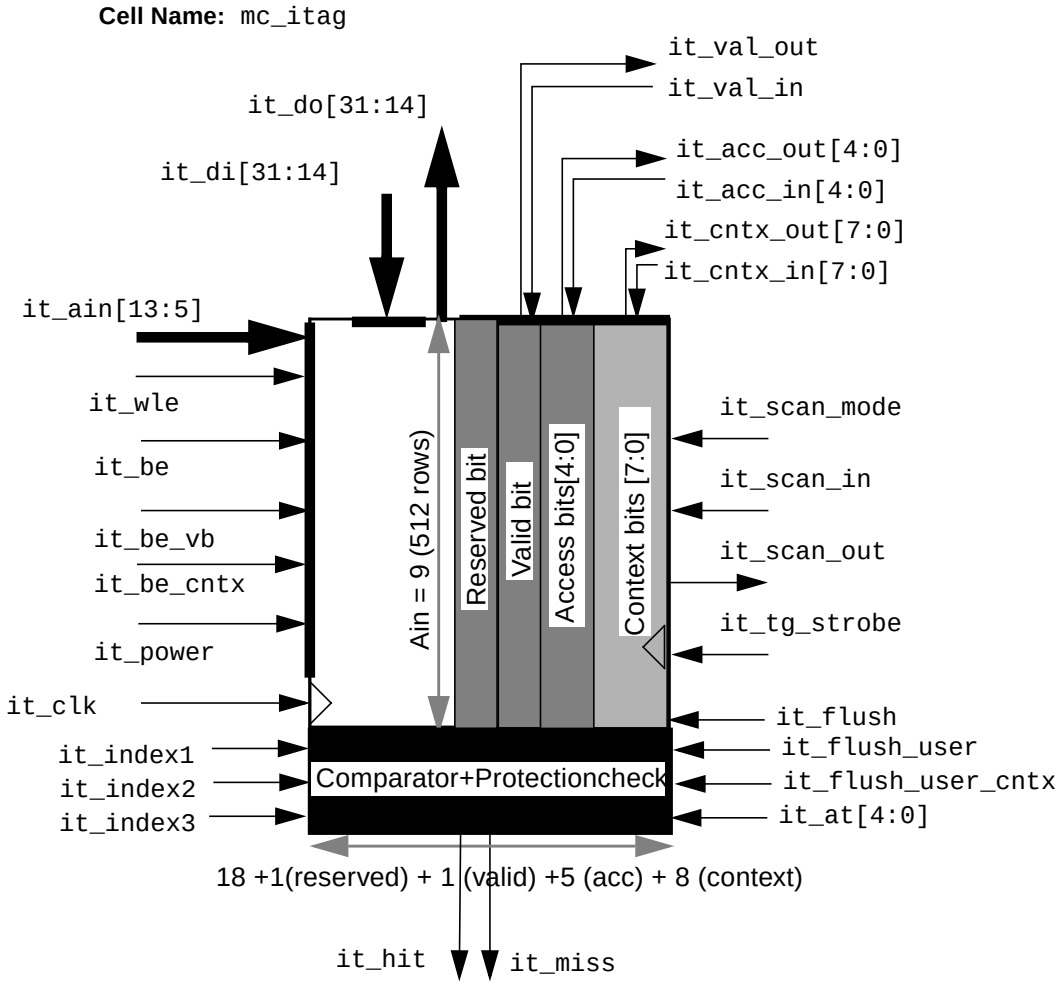

FIGURE 2-3 mc_itag Block Diagram

The operation of the mc_itag SRAM is similar to the operation described in the mc_icache read and write operations. The width of the mc_itag SRAM is 33 bits, the same as for the mc_dtag megacell. The fields are shown in FIGURE 2-3. The height of the mc_itag SRAM is 512 entries.

The new operations are the tag / flush match and a separately controlled write to the Valid bit, the Context bits, and the Access bits of each entry of the SRAM.

During a read cycle, the it_val_out, it_cntx_out, and it_acc_out bits are read out with it_do[31:14]. The Tag checks are done with the latched value of it_di[31:14] bus, and the it_cntx_in[7:0] bus.

A tag write operation that is enabled by assertion of any of the three be's writes all bits of the word addressed by it_ain *into* the Valid bit, the Context bits, and the Access bit field of the entry, depending on which of the corresponding be's are asserted. The tag is written with the data present on it_di. A Valid bit write operation enabled by assertion of it_be_vb writes only the Valid bit belonging to the location addressed by it_ain. The Valid bit is written with the value of it_val_in. Similarly, by asserting the it_be_cb write enable signal, all the other locations (namely, the Access bit and the Context bit) should be written at the location addressed by it_ain, with the data on it_acc_in, and it_cntx_in, respectively.

Note – The normal system write operation can sometimes assert any of the three write enables in any combination it chooses (it_be, it_be_vb, and it_be_cb). In this case, all the asserted fields addresses by it_ain should be written into with the data on the corresponding inputs.

The mc_dtag Cell

FIGURE 2-4 illustrates the mc_dtag cell described in this section.

Cell Name: mc_dtag

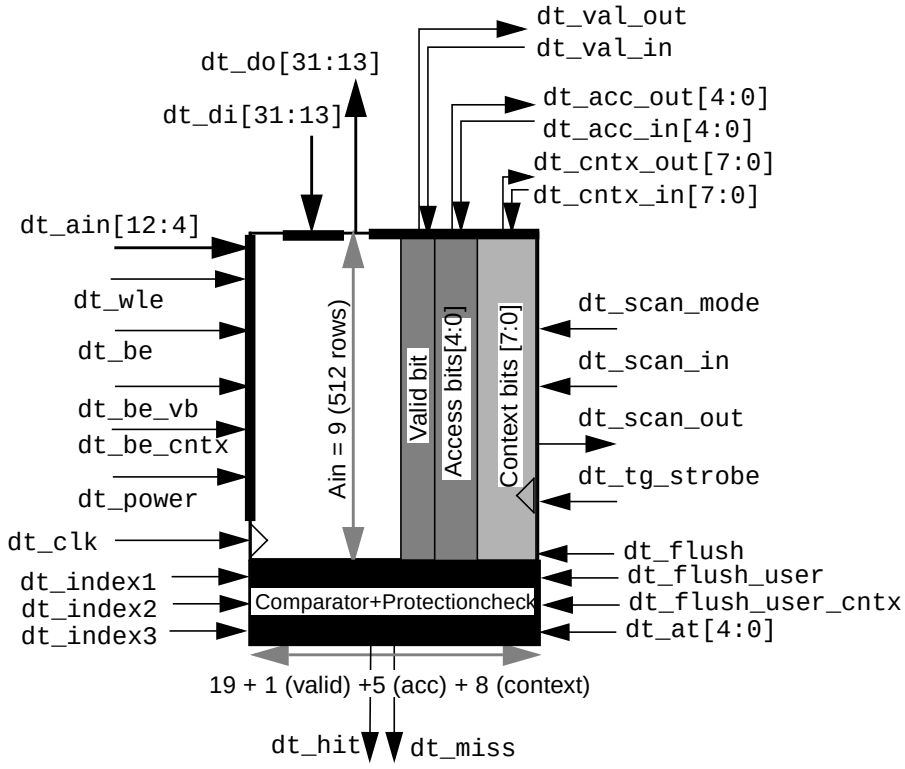

FIGURE 2-4 mc_dtag Block Diagram

The layout of mc_dtag SRAM is an identical copy of the mc_itag SRAM.

All the operations—read, write, and Tag / flush / protection checks—are the same as those of the mc_itag.

Logically, all the bits in the mc_dtag megacell are used, unlike the mc_itag megacell, in which the tag field is 1 bit less than for the mc_dtag megacell. This basically means that there is no Reserved bit in the mc_dtag megacell.

Tag SRAM Flush, Tag-Hit, and Protection Checks

The *mod*_page_hit, *mod*_segment_hit, *mod*_region_hit, *mod*_context_hit, and the *mod*_user_hit (all internal signals) signals are asserted if the various conditions shown in FIGURE 2-5 are true. The bits are shown indexed from the top. Since only 19 bits of the tag are stored, the page match flush is only partial. The lower bits for a page match check are implicit in the index of the two direct mapped Tag SRAMs.

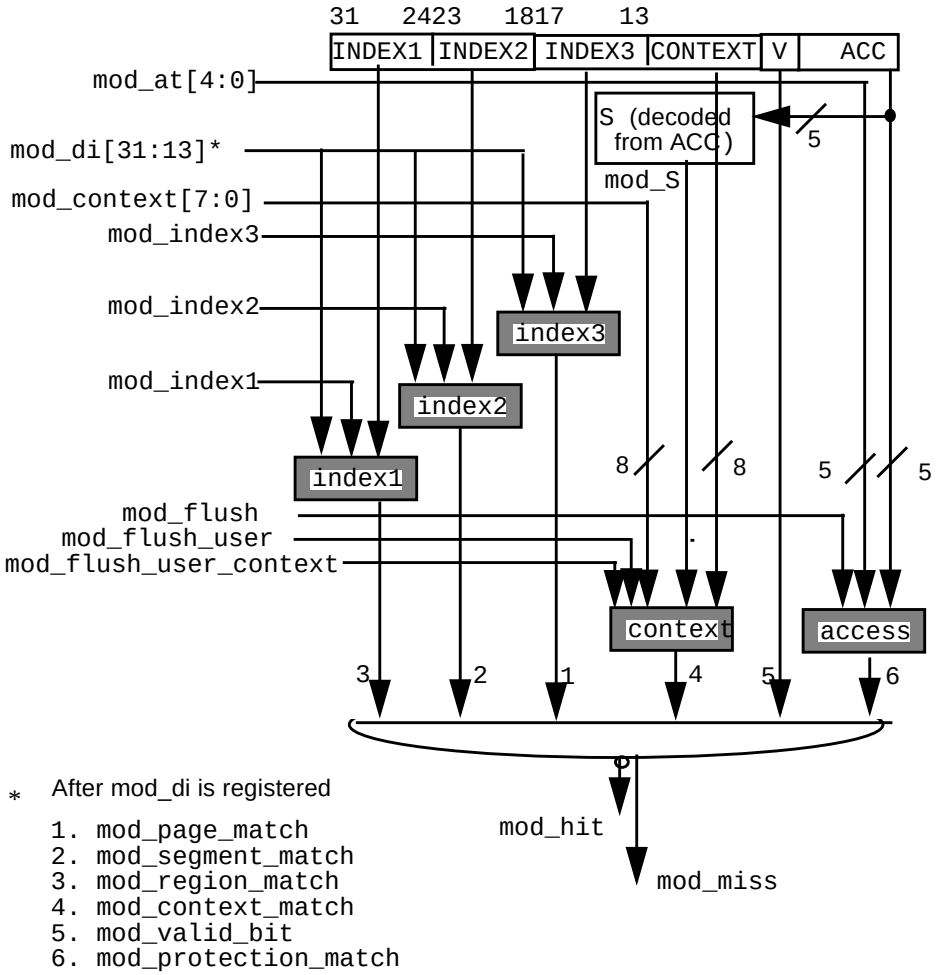

FIGURE 2-5 Tag SRAM Flash Clear Timing

The (*mod*_hit) is the most timing-critical signal, and it should be valid at most 8.5 ns into the read cycle. The flush/tag-hit checks are done on *every* read and write cycle. Notice that the Valid bits are used to disable all the flushes to save unnecessary flushes of cache lines whose tags do match but whose Valid bits are turned off. Also notice that the tag match logic and the flush match logic are combined into one hit/miss signal (*mod*_hit/*mod*_miss), where the intermediate signals are as follows.

- *mod*_S (Supervisor page indicator) =

  [(*mod*_acc[2:0] == 6) | (*mod*_acc[2:3] == 7)];
  (*mod*_S is thus decoded from the protection bits stored inside the TAGRAM.)

- *mod*_page_match (Page match o.k.indicator) =

  [(*mod*_S | (*mod*_cntx[7:0] == CONTEXT[7:0]))
  & (*mod*_din [31:13] == VA [31:13])];

- *mod*_segment_match (Segment match o.k.indicator) =

  [(*mod*_S | (*mod*_cntx[7:0] == CONTEXT[7:0]))
  & (*mod*_din [31:18] == VA [31:18])];

- *mod*_region_match (Region match o.k. indicator) =

  [(*mod*_S | (*mod*_cntx[7:0] == CONTEXT[7:0]))
  & (*mod*_din [31:24] == VA [31:24])];

- *mod*_context_user_match (Context match o.k.indicator) =

  [(~*mod*_S) & (*mod*_cntx[7:0] == CONTEXT[7:0])];

- *mod*_user_match (User match o.k.indicator) =

  [(~*mod*_S)];

- *mod*_protection_match (Protection match o.k.indicator) =

  | (*mod*_acc[0] & *mod*_at[0])
  | (*mod*_acc[1] & *mod*_at[1])
  | (*mod*_acc[2] & *mod*_at[2])
  | (*mod*_acc[3] & *mod*_at[3])
  | (*mod*_acc[4] & *mod*_at[4])] & ~*mod*_flush;
  (*mod*_acc[4:0] bits are the protection bits stored inside the TAGRAM.)

The main outputs of the megacells (*mod*_miss and *mod*_hit) are therefore (in terms of the above and previously mentioned intermediate signals):

- mod_miss = *mod*_page_match & *mod*_segment_match

  & *mod*_region_match & *mod*_context_user_match
  & *mod*_protection_match & *mod*_V];

  (*mod*_V is the Valid bit stored inside the TAGRAM.)

- mod_hit = ~ *mod*_miss;

2.1.3 AC Switching Characteristics

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

In this section, we define or derive critical timing parameters based on the application context of these SRAMs. This section will be updated as more details about the timing libraries and floor plan of the chip become available.

SRAM Read Timing

The read cycle timing diagram for all SRAMs and values for a few timing parameters common to all SRAMs are illustrated in FIGURE 2-6. Derivation of module-specific timing parameters is described in following sections.

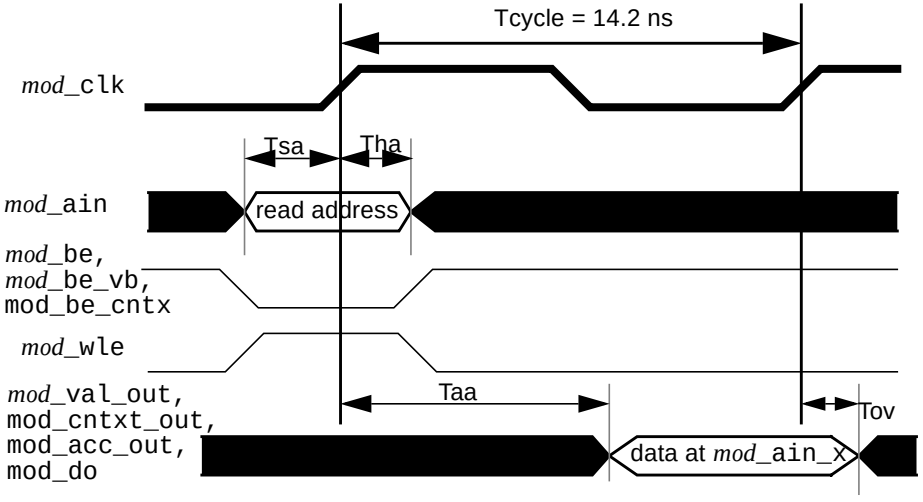

FIGURE 2-6 SRAM Read Timing

Tcycle = 14.2 ns
Tsa = Address setup time = 2.5 ns
Tha = Address hold time = 1.0 ns
Tov = Output valid time = 1.0 ns min (w.r.t next rising clock edge)

Data SRAM Critical Access Path

Of the two caches, the `mc_dcache` is more critical than the `mc_icache`. Hence, the critical path of the data RAMs is determined by the `mc_dcache`. The `mc_dcache` access time is derived from the access path logic as illustrated in FIGURE 2-7. The `mc_icache` required access time is ~6 ns.

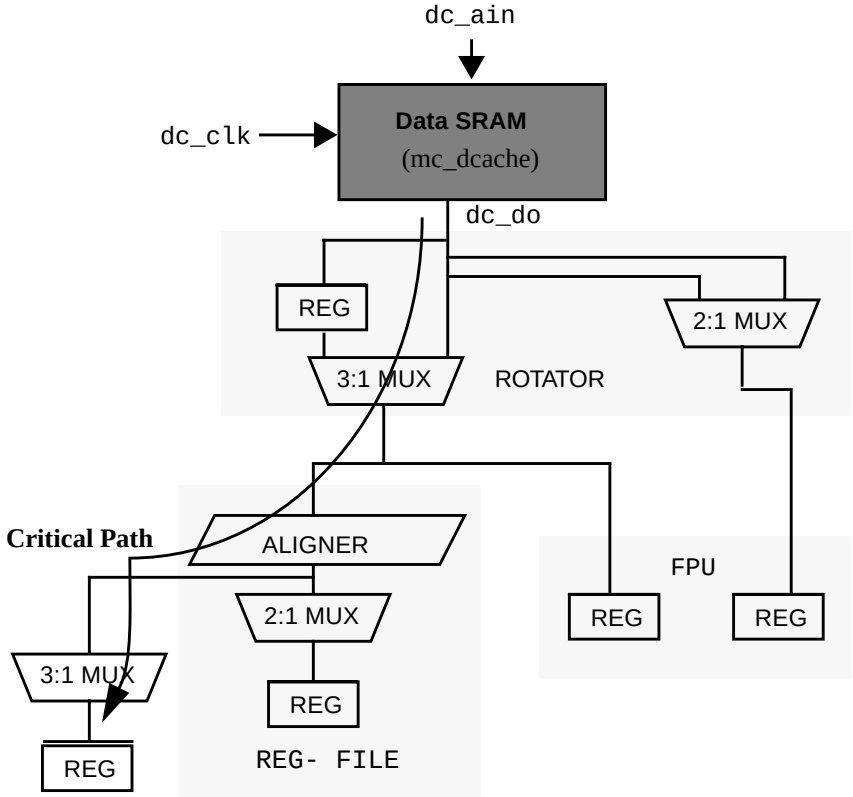

FIGURE 2-7 `mc_dcache` Critical Access Path

$T_{acc} = T_{cycle} - [T_{skew} + T_{wire} + T_{gate} + T_{setup}]$
T_{skew} = 1.0 ns (chip-level skew is 1.0ns)
T_{setup} = 1.6 ns
T_{wire} = 2.1 ns (4 nodes with moderate fan-out)
T_{gate} = 1.0 + 1.5 (load aligner) + 1.0 = 3.5 ns
T_{cycle} = 14.2 ns
T_{acc} = 14.2 - [1.0 + 2.1 + 3.5 + 1.6]
T_{acc} = 14.2 - [8.2]
T_{acc} = 6.0 ns (D-cache SRAM)

TAGRAM Critical Access Path

In the case of the TAGRAMs, access time is measured from the *mod*_hit output of the mc_itag and mc_dtag cache tag blocks. So, in that sense, the access time calculated in not the true access time of the TAGRAM, but rather it is a constraint under which to calculate the access time. The critical access time of both the TAGRAMs should be assumed to be equal (~7.0 ns). See FIGURE 2-8.

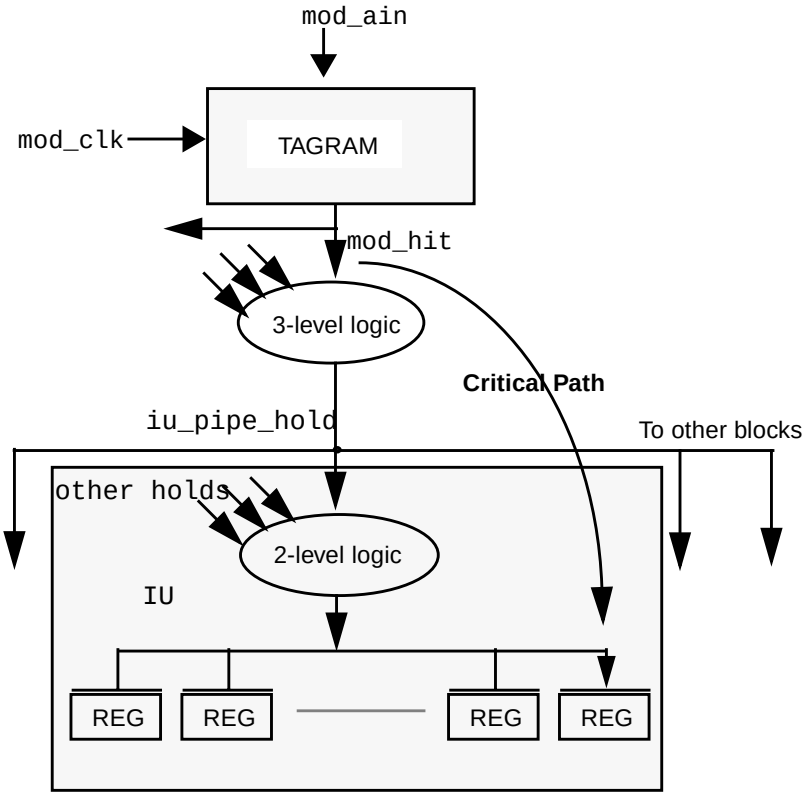

FIGURE 2-8 TAGRAM Critical Access Path

$T_{acc} = T_{cycle} - [T_{skew} + T_{wire} + T_{gate} + T_{setup}]$
T_{skew} = 1.0 ns (chip-level skew is 1.0 ns)
T_{setup} = 2.0 ns
T_{wire} = 2.0 ns (iu_pipe_hold drives a large fan-out > ~5pf)
T_{gate} = 1.5 + 1.0 = 2.2 ns
T_{cycle} = 14.2 ns
T_{acc} = 14.2 - [1.0 + 2.0 + 2.2 + 2.0]
T_{acc} = 14.2 - [7.2]
T_{acc} = 7.0 ns

Cache Data SRAM Bypass Mode

When the cache Data SRAMs are in bypass mode, the data on *mod*_di appears on *mod*_do (see FIGURE 2-9). The timing requirements on the bypass path are derived from the requirement that for external logic receiving the SRAM data output, the worst case timing is the same as for a regular cache SRAM read access.

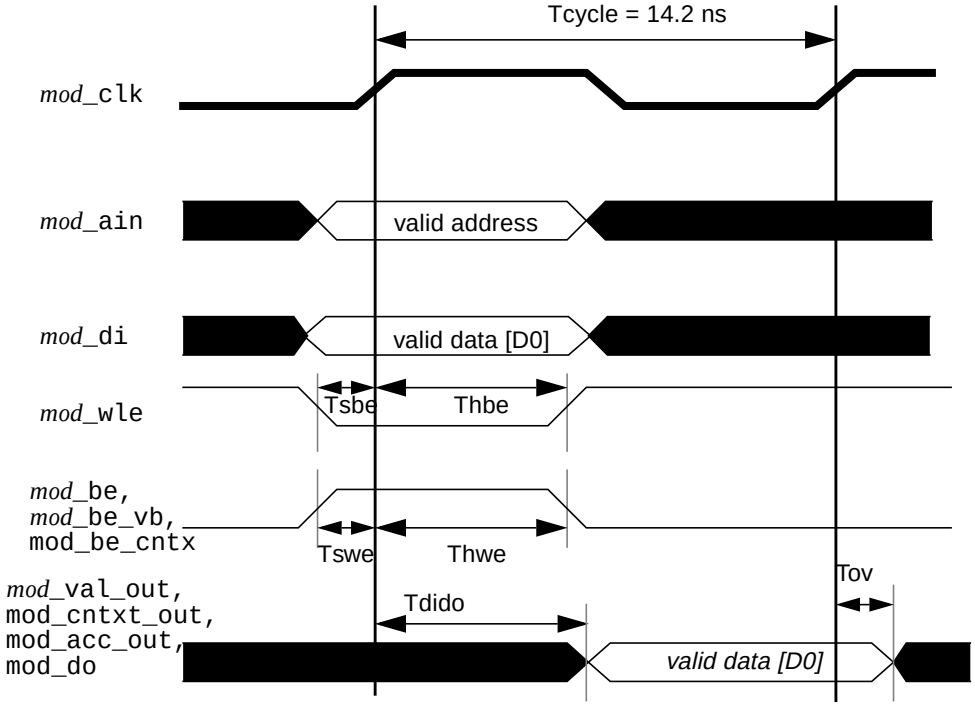

FIGURE 2-9 Cache SRAM Bypass Timing

Tsbe = Bypass Enable setup time = 1.0 ns
Thbe = Bypass Enable hold time = 1.0 ns
Tswle = Write Enable setup time = 3.0 ns
Thwle = Write Enable hold time = 3.0 ns
Tdido = Data-in to Data-out Delay = Taa (Same as access time)
Tov = Output valid time (w.r.t next rising clock edge) = 1.0 ns

Assume that *mod*_di arrives just in time to satisfy the setup requirement of the SRAM (Tsdi), just before the rising clock edge of the write cycle. The output data on *mod*_do is required to be stable no more than 6.0 ns after the rising clock edge (Taa).

SRAM Write Timing

FIGURE 2-10 shows the write cycle timing diagram for all SRAMs. The *mod*_be signal is vectored for the mc_dcache SRAM. All elements of the vector must satisfy the timing requirement shown for the unvectored signal.

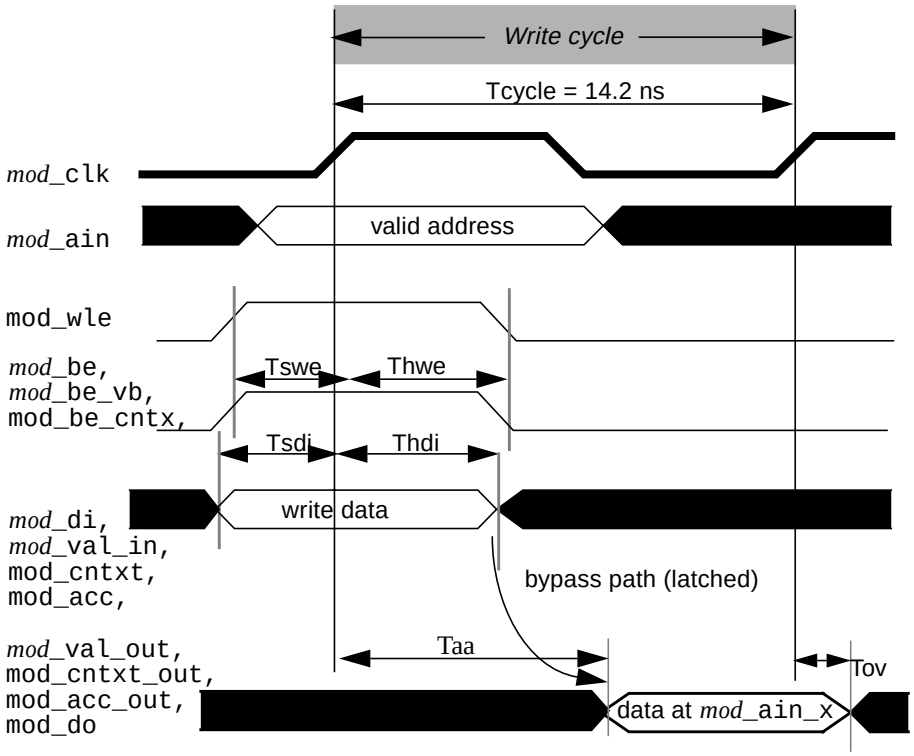

FIGURE 2-10 SRAM Write Timing

Tcycle = 14.2 ns min, no specified max
Tswle = Write Enable setup time = 1.0 ns
Thwle = Write Enable hold time = 1.0 ns
Tsdi = Data-in setup time = 0.0 ns
Thdi = Data-in hold = 2.0 ns

Note – Any of the various bypass enables can be active in a particular write cycle. So, the write cycle mechanism should be triggered by the assertion of any (logical OR) one of the bypass enables.

2.1.4 Timing Parameter Summary

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

TABLE 2-5 summarizes the timing values. All units are in nanoseconds (ns).

TABLE 2-5 Timing Numbers

Parameter	mc_dcache	mc_dtag	mc_icache	mc_itag
Taa	6.0	7.0 *	6.0	7.0 *
Tsa	2.5	2.5	2.5	2.5
Tha / Thsi	1.0	1.0	1.0	1.0
Tsdi	0.0	0.0	0.0	0.0
Thdi	2.0	2.0	2.0	2.0
Tsbe	1.0	1.0	1.0	1.0
Thbe	1.0	1.0	1.0	1.0
Tswle / Tspd	3.0	3.0	3.0	3.0
Thwle / Thpd	3.0	3.0	3.0	.3.0
Tssm	2.0	2.0	2.0	2.0
Thsm	2.0	2.0	2.0	2.0
Tssi	1.0	1.0	1.0	1.0
Tcycle	14.2	14.2	14.2	14.2

*. The TAGRAM access time is defined with respect to the mod_hit port and not mod_do.

The symbols used in the previous table are as follows:

*T*aa = Delay from clock to `mod_do.`
*T*sa = Setup time for `mod_ain` w.r.t clock.
*T*ha = Hold time for `mod_ain` w.r.t clock.
*T*sdi = Setup time for `mod_di` w.r.t clock.(This includes all data input signals that are stored in the SRAM).
*T*hdi = Hold time for `mod_di` w.r.t clock. (This includes all data input signals that are stored in the SRAM).
*T*hsi = Hold time for `mod_scan_in` w.r.t clock.

The next four symbols collectively control the read / write cycles of the SRAM)

*T*sbe = Setup time for `mod_be`[] w.r.t clock.
*T*hbe = Hold time for `mod_be`[] w.r.t clock.
*T*swle = Setup time for `mod_wle` w.r.t clock.
*T*hwle = Hold time for `mod_wle` w.r.t clock.

*T*spd = Setup time for `mod_power_down` w.r.t clock.
*T*hpd = Hold time for `mod_power_down` w.r.t clock.
*T*ssm = Setup time for `mod_scan_mode` w.r.t clock.
*T*hsm = Hold time for `mod`_scan_mode w.r.t clock.
*T*ssi = Setup time for `mod_scan_in` w.r.t clock.

Output Loading

The timing parameters in this specification must be met under the output loading conditions noted in this section.

The output loading on the *mod*_`do` (and *mod*_`val_out` for Tag SRAMs) outputs are as follows:

`mc_dcache`, output — 2.0 pF
`mc_icache` — 3.0 pF
`mc_dtag` — 1.5 pF (`dt_hit` also)
`mc_itag` — 1.5 pF. (`it_hit` also)

All *mod*_`scan_out` outputs are loaded with 1.0 pF. More loading information about the critical signals (*mod*_hit) will be added as the design/library is finalized.

2.1.5 Clock Input

The clock input to all SRAMs should be unbuffered. The input capacitance and the series resistance of interconnects must be accurately characterized and documented upon megacell delivery.

2.2 I/O Description

This section describes the I/Os for a generic SRAM and uses generic I/O names. The I/Os and functionality for each specific SRAM module are described in detail in following sections.

The naming convention for signal names when more than one specific SRAM module is being discussed is *mod*_signal, where *mod* is the specific module name used as a prefix to each signal of the module, and _signal is the signal name common to all modules. In the following sections, *mod* is replaced by the actual module name where appropriate.

Note – All registered inputs are registered on the rising edge of *mod*_clk.

- *mod*_ain**.** Address input, registered. The word at the location addressed by *mod*_ain is accessed for either reading or writing its contents. The width of *mod*_ain is log2(no. of words in SRAM).
- *mod*_wle**.** Word line enable, registered. In the cycle following *mod*_wle's assertion, the SRAM performs a read / write from / to the location addressed by *mod*_ain.
- *mod*_di**.** Data input, registered. The value of *mod*_di is written to the location addressed by the contents of the *mod*_ain address register if *mod*_wle *and* *mod*_be were asserted at the previous rising edge of *mod*_clk. The value of *mod*_di is registered and bypassed to *mod*_do if *only* *mod*_be is asserted.
- *mod*_do**.** Data output. Contains data addressed by *mod*_ain during a read access if *mod*_be is not asserted (but *mod*_wle is asserted), or data on *mod*_di if *mod*_wle *and* *mod*_be are asserted.
- *mod*_be**.** Bypass / write enable, registered. In the cycle after *mod*_be's assertion, *mod*_do is driven with the value of *mod*_di. *mod*_be is guaranteed to be asserted for a Write cycle.
- *mod*_clk**.** (Clock) This input is a single-phase clock. Any additional clocks from either *mod*_clk or *mod*_tg_strobe (phases, polarities, delayed versions, etc.) must be generated by the SRAM itself.
- *mod*_clk_**hld.** This input is a clock hold. It is present only on the SRAM megacells and not the TAGRAM megacells. When asserted, the input data register must retain its previous value, on the next rising clock edge.
- *mod*_scan_mode**,** *mod*_scan_in**,** *mod*_scan_out**,** and *mod*_tg_strobe. Signals related to SRAM testability, described in detail in the Testability section of the SRAM specification.

2.3 Area and Power Dissipation

For the mc_icache, mc_dcache, mc_itag, and mc_dtag SRAM modules, a total power budget of 2.0 watts has been allocated, for worst case environment and process, and under normal operation where either a read or a write access to every SRAM is performed each 14.2 ns cycle. During partial powerdown, it is expected

that the power consumed will reduce by at least 75%; during full powerdown, it is assumed that the power consumed will reduce by more than 90% of the normal operating power consumed.

2.4 Layout

All SRAMs described in this chapter should conform to the approximate layout shown in FIGURE 2-11. The *mod*_do, *mod*_di I/O buses should enter and leave as shown.

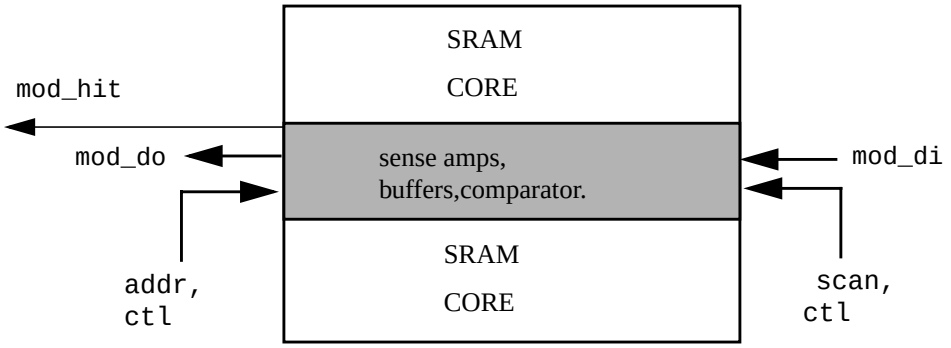

FIGURE 2-11 SRAM Layout Bounding Box

2.5 Initialization and Abnormal Conditions

No specific initialization sequences or states are required for correct system operation. At least 10 clock cycles will be issued to the SRAMs before the chip expects the SRAMs to operate.

The SRAMs must not draw excessive current or be otherwise damaged during a typical system powerup sequence, during which time the clocks are not guaranteed to be running.

The SRAMs must operate within all parameters described in this specification under the following conditions:

- Clock frequency may be any frequency at or under 70 MHz, *including zero frequency.*

- *mod*_clk and *mod*_tg_strobe can be held indefinitely in their high states for an undefined long period of time.
- All internal clocks are derived from a single double-frequency source, and because its frequency is reduced from 140 MHz (as during test or use in other applications), most timing relationships scale uniformly.

2.6 Testability

Two methods can be used to test the SRAMs. The primary method for production testing is for the Integer Unit to execute a small program that has been loaded into the Icache, which exhaustively tests all SRAMs. However, for debug purposes, a scan-based testing method is required. The terminals of the SRAMs are not available directly at the pins of the chip.

2.6.1 Functional Description

For scan-based testing and debug, the SRAM's internal address register and control bits are loaded by JTAG-controlled scan. Whether the SRAM is in scan mode or in normal operation mode is determined by the state of *mod*_scan_mode. When *mod*_scan_mode is asserted, the address register and all control bits in the SRAM must be connected into a single shift register.

The scan operation is not allowed to be destructive to any data in the SRAM core. Therefore, the outputs of the flip-flops containing the Write Enable must be disabled during scan mode and enabled only when scan mode is exited, at which time all control bits contain their desired states for the test.

After the scan sequence is completed and the address and control bits of SRAM contain the states required for the test, the SRAMs are cycled once *without disturbing the state of any flip-flops in the scan chain*. This requires a "strobe" input to the SRAMs which starts the internal self-timed circuits for the SRAM access (either read, write, or clear) but does not advance the address register or control bit flip-flops. This strobe is generated external to the SRAM and is called *mod*_tg_strobe.

Each test (read, write, or clear) requires one scan-in sequence followed by a strobe cycle.

Scan operations take place between all write or read operations that are controlled through the JTAG interface. A latency of the number of address and control bits in the SRAM plus about 6 cycles may be incurred between any two consecutive SRAM tests or vectors during JTAG-based SRAM tests.

Four I/Os are required per SRAM module to support testability. They are:

- *mod*_scan_in — Single bit input that is the scan chain (shift register) input
- *mod*_scan_out — Single bit output that is the scan chain (shift register) output
- *mod*_scan_mode — Scan mode enable signal
- *mod*_tg_strobe — Special “strobe” input, similar to the *mod*_clk input but does not disturb the address register’s contents

Scan chain connection order is not specified. The implementor may choose any scan chain connection order desired.

2.6.2 Scan Mode Timing

FIGURE 2-12 is the timing diagram showing the basic timing relationships between signals during scan mode. The timing parameters during scan mode operation are substantially relaxed from those during regular operation. If the SRAM implementors require any additional timing flexibility, they can normally be accommodated.

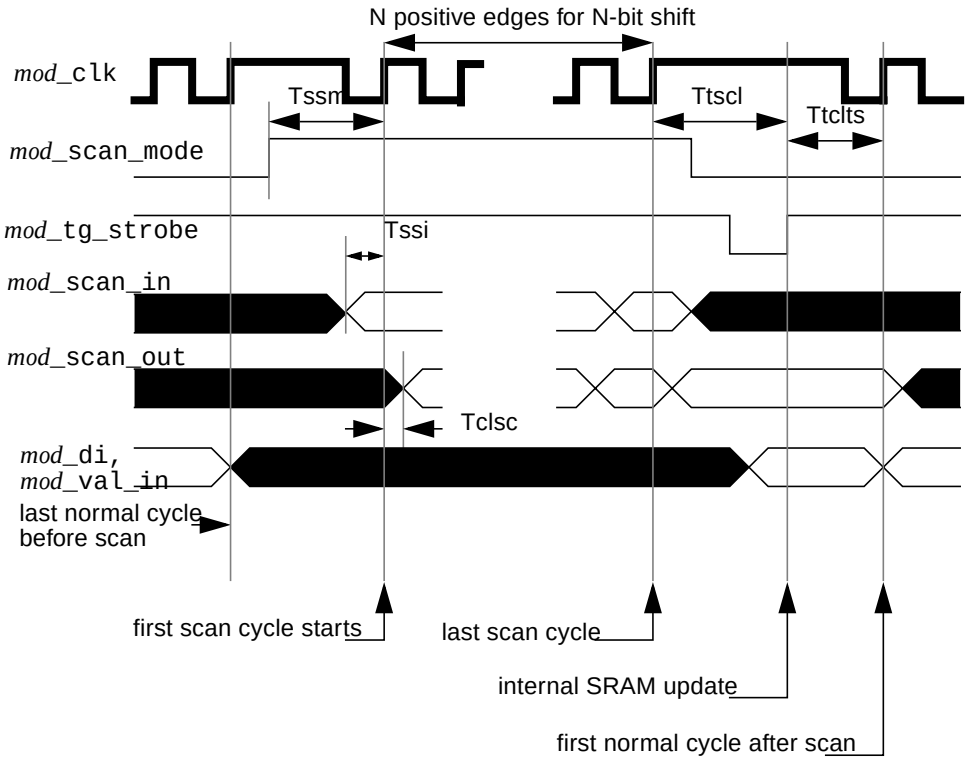

FIGURE 2-12 Scan Mode Timing

Note – *mod*_clk is held in its high state for as long as necessary so that all events that need to occur while it is held are allowed sufficient timing margin. The length of time *mod*_clk is held is determined by a state machine that can easily be adjusted to allow for longer or shorter hold times.

Clock frequency during scan mode is not the same as before and after scan mode.

2.7 Implementation Notes

The SRAM implementor has some latitude to make design trade-offs to maximize design simplicity and generality (reusability). An SRAM that is implemented in accordance with this specification can make design trade-offs in the following ways.

- The `mc_dcache` SRAM has eight separate pairs of Byte Bypass Enable inputs, whereas the `mc_icache` SRAM has only two. The `mc_icache` may also have eight pairs of Bypass Enable inputs which control logic will tie together to generate only two Bypass Enables.
- All input signals unique to any SRAM module may appear on any other SRAM modules and will simply be driven to an inactive state for modules that do not require that input.

CHAPTER 3

TLB CAM/SRAM

This chapter describes the TLB CAM/SRAM with respect to the following areas:

- TLB organization overview, describing the TLB CAM, select block, and SRAM
- I/O signal descriptions for TLB CAM, SRAM, and decode select
- Functional description, including content-addressed read compare operation, error detection for that operation, direct-addressed read and write operations, and flush operation
- Timing diagrams for CAM read operation, SRAM direct-addressed and content addressed read operations, CAM/SRAM write timing, and the CAM flush operation
- TLB I/O pin description
- AC Switching Characteristics
- Area and power estimate
- Initialization and abnormal conditions
- Testability

Please refer to Chapter 1, *Requirements*, for information, guidelines, and requirements that apply to all megacells.

Note – The TLB (Chapter 3) and PCI Controller IOTLB (Chapter 4) are each implemented as a closely coupled CAM and SRAM combination. The IOTLB CAM/SRAM combination is identical to the processor's CAM/SRAM macrocell, with the exception of the number of entries.

3.1 TLB Organization Overview

The TLB is implemented as a closely coupled CAM and SRAM combination. The CAM contains the tag and the SRAM contains the data portion of a PTE (Page Table Entry) or PTP (Page Table Pointer). The CAM is a 32-entry, 42-bit-wide fully

associative array; the SRAM is a 32-entry by 28-bits array. The CAM/RAM combination is named `mc_tlb`. All models, netlists, layouts, and bounding boxes must reflect this name.

The TLB performs virtual-to-physical address translations by using a fully associative CAM lookup to index into a RAM that holds the translated address. In addition, the TLB holds the Page Table Pointers (PTPs) required to read the correct translated page address (Page Table Entry or PTE) from the page tables in memory. Both the CAM and RAM sections are accessible independently for reading and writing in diagnostic or update modes. For a complete description of the functional requirements of the TLB, please refer to the *SPARC Architecture Specification* (Appendix H, Reference MMU Architecture).

FIGURE 3-1 illustrates a simplification of the TLB. The blocks in the diagram are described below and in the subsequent subsections.

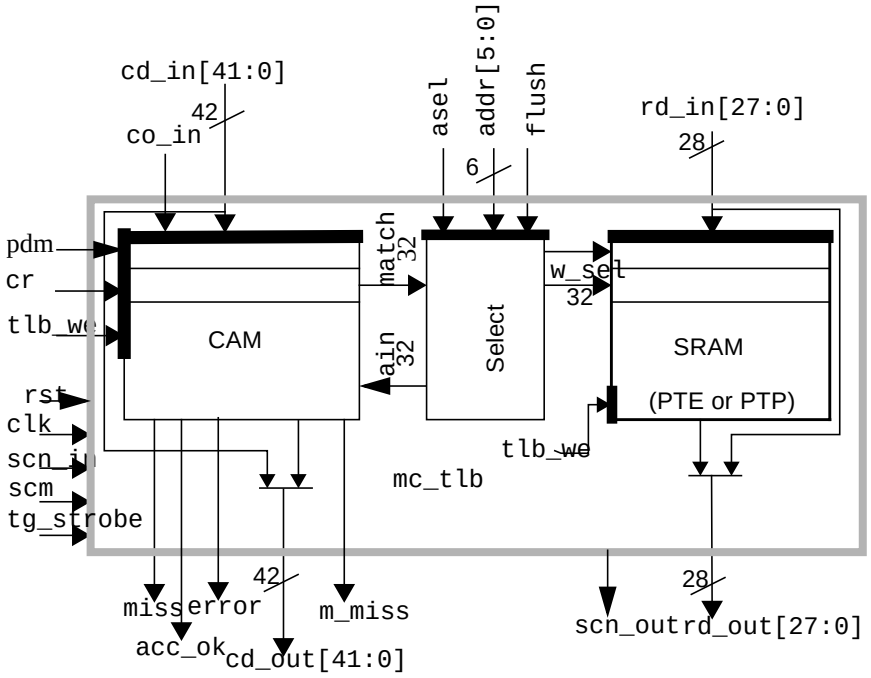

FIGURE 3-1 TLB (Simplified)

The TLB comprises tightly coupled CAM and SRAM arrays. Each of the 32 entries in the TLB consists of two parts: a 42-bit tag word in the CAM array and a 28-bit data word in the SRAM array.

Under the normal operation of content-addressed address translations, `cd_in` is compared simultaneously with all tag words in the CAM array. A tag can match on the basis of the values in `cd_in` and the states of mask and flag bits in each tag. On

a match, the SRAM's data word for the matching entry is driven to `rd_out`. Multiple entry matches can occur, in which case the `rd_out` and `cd_out` outputs are undefined; system software attempts to prevent this condition, but prevention cannot be guaranteed in all cases.

3.1.1 TLB CAM

The CAM is an array of sixty-four 42-bit words. It has registered `tlb_we`, `cr`, and `c_en` inputs and a 42-bit wide `cd_in` input. FIGURE 3-2 illustrates a fully associative CAM.

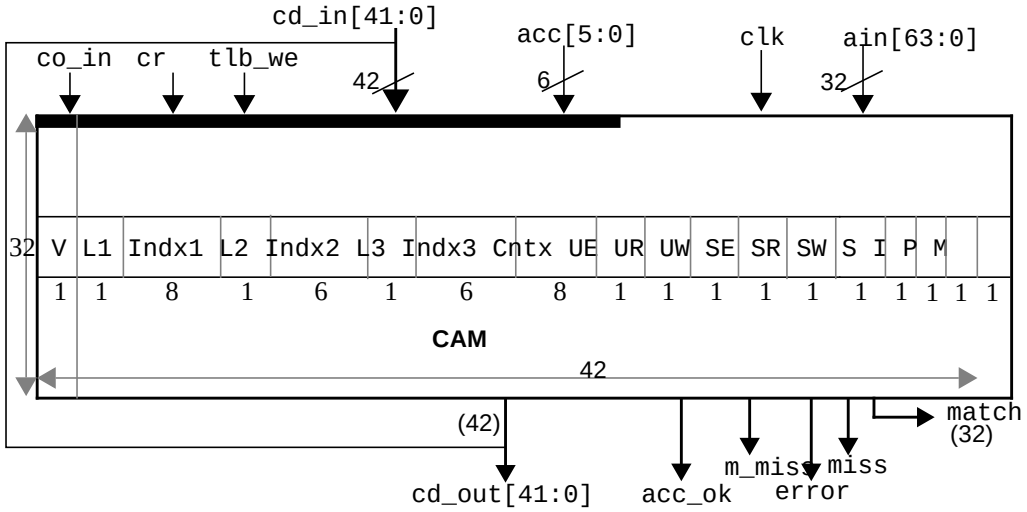

FIGURE 3-2 32-Entry Fully Associative CAM

The `L1`, `L2`, `L3`, and `S` bits of the CAM word are mask bits that control whether the subfield matches of `cd_in` against the corresponding index and context fields are used in the full entry match. The `V`, `I`, and `P` bits are flag bits that enable or disable the full entry match. A Verilog-like logical expression that defines the match is shown below.

```
Entry_match = (V == V_in) &&
((L1 || L1_in) || (Indx1 == Indx1_in)) &&
((L2 || L2_in) || (Indx2 == Indx2_in)) &&
((L3 || L3_in) || (Indx3 == Indx3_in)) &&
((S || S_in) || (Cntx == Cntx_in)) &&
(P == P_in) || (co_in) &&
(I == I_in) || (co_in));
```

The cd_in bits corresponding to the mask and flag bits in the CAM word require specific input values during compare and write operations. These values are guaranteed to be provided by external logic. A brief explanation on how the word subfields are used follows.

Each CAM word contains four Virtual Address fields. In order of decreasing significance, these are Context (CID[7:0]), Index1 (VA[31:24]), Index2 (VA[23:18]), and Index3 (VA[17:12]). Three of these fields have corresponding match-enable bits stored in the CAM.

- When the Supervisor bit of a CAM entry is 1, then the match on its Context field is forced to be true regardless of the value on the Context input lines.
- When the Level-1 bit of a CAM entry is 1, then the match on its Index1 field is forced to be true regardless of the value on the VA[31:24] input lines of the CAM.
- When the Level-2 bit of a CAM entry is 1, then the match on its Index2 field is forced to be true regardless of the value on the VA[23:18] input lines of the CAM.
- When the Level-3 bit of a CAM entry is 1, then the match on its Index3 field is forced to be true regardless of the value on the VA[19:12] input lines of the CAM.

Since the Supervisor, Level-2, and Level-3 bits are implemented with XOR CAM cells that match against an input line, the corresponding input lines of the CAM are driven by external logic to the following constant values during CAM match accesses in order for the above behavior to be realized.

```
Supervisor_in = 0 for normal access, 1 for supervisor mode access
Level-1_in = 0;
Level-2_in = 0;
Level-3_in = 0;
```

In addition to the above four Virtual address match-enable bits, each CAM word also has Valid, PTP, and IO flag bits, which are matched against corresponding input lines. These input lines are driven by external logic to the following values during CAM match accesses.

```
Valid_in = 1;
PTP_in = 0 for translations, 1 for PTP (table walk) accesses;
IO_in = 0 for CPU accesses, 1 for I/O accesses.
```

The results of all seven of these matches (Context, Index1, Index2, Index3, Valid, PTP, and IO) are ANDed to produce the match output for each CAM entry, as described in the logic equation provided earlier.

When a CAM entry is created, the fields are written by external logic as follows.

```
 Level-3 PTE:
    Valid=1; PTP=0;
    Supervisor=0;
    Level1=0; Level2=0; Level3=0;
    Context=CID; Index1=VA[31:24]; Index2=VA[23:18]; Index3=VA[17:12].

 Level-2 PTE:
    Valid=1; PTP=0;
    Supervisor=0;
    Level1=0; Level2=0; Level3=1;
    Context=CID; Index1=VA[31:24]; Index2=VA[23:18]; Index3=dont-care.

 Level-1 PTE:
    Valid=1; PTP=0;
Supervisor=0;
Level1=0; Level2=1; Level3=1;
Context=CID; Index1=VA[31:24]; Index2=dont-care; Index3=dont-care.
PTP:
Valid=1; PTP=1;
Supervisor=1;
Level1=0; Level2=0; Level3=0
Index1=PA[27:20]; Index2=PA[19:14]; Index3=PA[13:8]; Context=PA[7:2].
```

The data output from the CAM (`cd_out`) is also required for a parallel, single-entry translation register and for diagnostic support. Additionally, the three Level bits (L1, L2, and L3) are required to control the multiplexing of the outputs from the TLB SRAM. This makes the data output from the CAM, specifically the L1, L2, and L3 bits, part of a critical path through the TLB during address translation.

The Valid bit (V) is a unique bit in the CAM, in that it supports the ability to be “flush” cleared: the Valid bit in all CAM words is cleared when the flush clear is performed. The flush clear is performed by asserting the flush operation with those mask bits set to 1 (lvl in, as 111).

A logic 1 on the `miss` output from the CAM indicates that none of the entries “matched” the input. If one of the entries matches the input, the `miss` output will be low.

3.1.2 TLB Select Block

The TLB CAM and SRAM require addresses provided from two separate sources, to support two different modes of operation. The compare mode, used during address translation, has the CAM and RAM word select lines driven by the CAM match lines. The update mode, used when updating the TLB (or diagnostic access), has the CAM and RAM word select lines driven by a decode of the `addr` inputs (6 bits). The control of the address selects is performed by the address select decode block, which conceptually resides between the CAM and SRAM arrays. FIGURE 3-3 illustrates the logic.

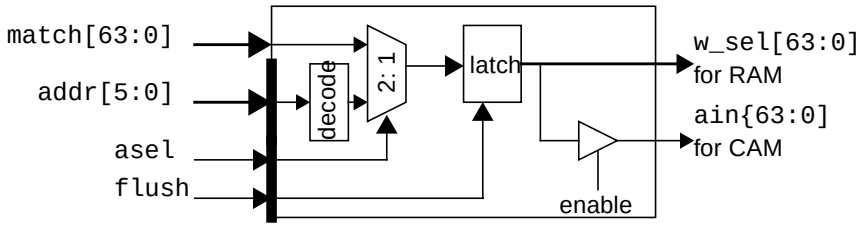

FIGURE 3-3 Address (Word) Select Decode Logic

The address select input (`asel`), if low, selects the `match` word line to be used, or, if high, the decoded address input (`addr[5:0]`). The registered `hold` input, when high, indicates that the latch should be held (that is, not transparent) in the following cycle. It is important to note that the output of the word select decode block drives the word enables for both the CAM and the SRAM arrays.

The word select for the CAM can be driven by the select decode block when a normal content-addressed operation is not in progress. The enable is defined by the following equation:

```
CAM_wsel_enable = (asel || (flush && tlb_we));
```

3.1.3 TLB SRAM

The TLB SRAM is a 32-entry by 28-bit array, illustrated in FIGURE 3-4. The `w_sel[63:0]` input directly addresses the SRAM words.

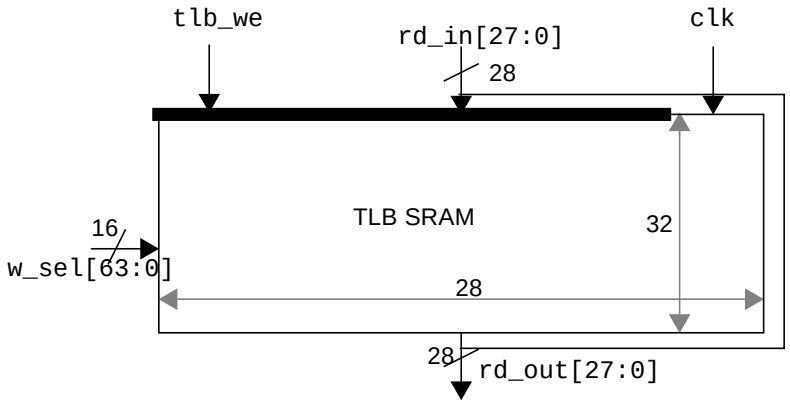

FIGURE 3-4 TLB SRAM Array

3.2 I/O Description

The CAM, TLB_SRAM, and TLB decode select logic I/O is described separately below, although the TLB megacell (`mc_tlb`) is a single layout block.

3.2.1 TLB CAM

`cd_in[41:0].` This input is used during both Read and Write operations of the CAM. During a content-addressed Read operation, this input contains the data vector upon which a fully associative search will be performed in the CAM array to check if a match occurs. For a Write operation, this input contains the data vector to be stored at the location pointed to by `ain` when the registered `tlb_we` is active.

`ain[63:0].` This is a 32-bit input that supplies the word select lines for the CAM. It instructs the CAM array as to the location to which a data vector is to be loaded when `tlb_we` is active (direct-addressed Write), the CAM location to read during a direct-addressed Read, or the CAM entry to be invalidated during a Flush operation. The `ain` lines are driven from either the decoded `addr` input or the CAM `match` lines, based on registered `asel`. These signals are used and generated only internal to the TLB.

`acc[5:0].` This input indicates the kind of operation being tried. During a content addressed read operation, this input contains the decoded operation type that will be compared to the matched CAM entries stored access privilege.

tlb_we. This signal enables a Write operation to the CAM and RAM array in the following cycle. This input is registered.

cr. This signal indicates that a Read operation to the CAM array will be executed in the next two cycles. This input is registered. Note that c_rd may not be driven active unless asel is active. That means only direct-addressed reads of the CAM entries are supported by the TLB. Content-addressed reads of CAM entries are not supported.

co_in. This input overrides the compare of the P bit and the I bit in the CAM. This input is registered.

m_miss. This output indicates the state of the M bit for the CAM entry that matched the cd_in criteria during a c_en compare cycle.

rst. rst is not a real reset signal to reset all register within tlb. It just tries to force TLB into TLB SRAM direct read operation state while reset is asserted. This action is to prevent a multiple hit on tlb entry while a reset happen.

clk. This is a single-phase clock input. If a special clocking, such as two-phase nonoverlapping clocks are required, they must be generated locally to the megacell.

match[63:0]. During a Read operation, the results of an associative match are reported on this output bus, with each match bit reflecting whether the corresponding tag in the CAM array matched c_data_in (qualified by the associated flags). Under normal operation, zero or one match lines should be active at a time, although this cannot be guaranteed under all conditions. The TLB must ensure that if multiple match lines are active at one time, the TLB will not be damaged and will not output illegal CMOS voltage levels at any output, although the logical states of the outputs are irrelevant. The match signals are internal to the megacell.

cd_out[41:0]. In a content-addressed Read operation that results in a single matched entry, the contents of the matching CAM tag are output here. In a direct-addressed Read operation, the tag of the entry being addressed is output here. In all other cases, this output is ignored; however, the CAM must guarantee that cd_out settles to legal CMOS signal levels within 10 ns of the specified access time for a valid access.

miss. The miss output from the CAM indicates that none of the entries matched c_data_in. If any of the entries matched, the miss output will be low.

error. The error output from the CAM indicates that the CAM that matched the c_data_in criterion has an access privilege level that will not allow the current access indicated by the asi bits to be done. If there is no access privilege error for the matched entry, then the error output will be low.

acc_ok. This output indicates that the current CAM lookup is a hit with no access error and the m bit is set appropriately for the desired access.

pdm. (powerdown mode). This is a pin to switch off the sense amplifier when entering powerdown mode. When entering powerdown mode, the `clk` stops and stays in level high. The `clk` high causes the tlb's sense amplifier to turn on. A separated signal `pdm` is used to turn off sense amplified.

3.2.2 TLB SRAM

rd_in[27:0]. This input is used only during a Write operation. It contains the data to be written in the location selected by `w_sel` in the cycle `tlb_we` was high.

w_sel[63:0]. This 32-bit input is used during a Read or Write operation to select individual SRAM words for reading or writing. The `w_sel` lines are driven from either the decoded `addr` input or the CAM `match` lines. These signals are used and generated only internally to the TLB.

clk. As with the CAM array, a single-phase clock is the input. Based on circuit requirements, any special clocking scheme must be generated locally.

rd_out[27:0]. During a Read operation, `rd_out` will contain the contents of the SRAM word selected by `w_sel[63:0]`. In the case where no, or more than one, `w_sel` line is active, the logical values on `rd_out` are ignored; however, the SRAM must guarantee that `rd_out` settles to legal CMOS signal levels within 10 ns of the specified access time for a valid access.

3.2.3 TLB Decode Select

asel. This input selects either `addr` or `match` as the source of the address that generates `w_sel[63:0]` and `ain[31:0]` in the following cycle (registered input). When the registered `asel` is low, the `match` input is selected; when `asel` is high, the decoded `addr` input is selected.

addr[5:0]. This 6-bit input supplies an encoded address used for direct-address operations. This input is registered, so the address is used in the subsequent cycle from which it was supplied.

flush. This input indicates the following cycle is a PTE flush cycle. This input is registered.

ain[63:0]. This is a 32-bit output that supplies the word select lines for the CAM. The `ain` lines can be driven from either the decoded `addr` input or the CAM `match` lines, based on registered `asel`, or they can be held from the previous cycle when `hold` is active. These signals are used and generated only internally to the TLB.

w_sel[63:0]. This is a 32-bit output that supplies the word select lines for the SRAM. The w_sel lines can be driven from either the decoded addr input or the CAM match lines, or they can be held from the previous cycle when hold is active. These signals are used and generated only internal to the TLB.

3.3 Functional Description

The TLB CAM and SRAM arrays operate in a tightly coupled manner to perform address translations. Since the caches are virtually tagged, an address translation is not required for each instruction fetch and memory reference. Also, the goal of this processor is to approach a 1 cycle-per-instruction execution rate for instructions and data that are resident in the caches. This goal requires that address translation, which includes a TLB lookup, execute in a single stage of the processor pipeline. The target frequency of 70 MHz, with significant logical operations to be performed on the translated address, dictates a need for an extremely fast TLB lookup. Therefore, we took care to keep the functionality simple and to direct the logic to speed the translation as much as possible. Since both the TLB CAM and SRAM are so tightly coupled in their operation, the following functional description considers them as one unit.

3.3.1 Content-Addressed Read Operation (Compare)

The TLB CAM and SRAM are always addressed as a single unit (the same address is decoded for both blocks). The address is supplied in one of two ways: either through the compare match operation (content addressable) or through an encoded address supplied on the addr address lines. The match operation allows both the TLB CAM and SRAM to be addressed by the contents of the CAM. This is the mode in which address translations occur.

During a content-addressed read operation, the data for cd_in is supplied from an external register, which passes through a 5:1 multiplexer. A compare cycle is indicated by two signals, the asel and c_en inputs. The asel registered input, when not asserted, indicates that the subsequent cycle will address the CAM and RAM from the match lines. The c_en input indicates that a compare cycle is enabled in the following cycle. The CAM then performs the compare, outputting an asserted match line(s) corresponding to the entry(ies) that matched. The CAM also detects whether no entry matched and asserts the miss output to indicate no match. The asel input is held low to select the match lines to drive the word selects to both the CAM and SRAM arrays (ain for CAM, w_sel for RAM). Each block then drives the data outputs with the contents of the entry corresponding to the word select line.

3.3.2 Error Detection

The CAM will contain the decoded access privilege bits that correspond to the access privilege assigned to that entry. When a content-addressed read (compare) operation is done, the access privilege bits of the current operation are compared to the access privilege bits stored in the CAM location that matched the hit criterion present when the `c_en` signal was asserted. The outcome of the protection compare has no effect on the hit-or-miss outcome of the CAM. It merely conveys the legitimacy of the current operation that matches the CAM hit criterion. The `error` signal indicates errors as defined by the following equation.

```
error = (Valid == 1) &&
((!UE && UE_in) || (!UR && UR_in) || (!UW && UW_in) ||
(!SE && SE_in) || (!SR && SR_in) || (!SW && SW_in));
```

Additionally, an acc_ok output is used to indicate that the current access is OK. This output is governed by the following equation.

```
acc_ok = !miss && !error && m_ok;
```

Also, an indicator for the “modified” bit of the CAM entry corresponds to a hit. This bit is m_ok and is described by the following equation. The `m_miss` output pin is logically ORing the m_ok to tell that “modified” is `miss` or not. The following equation shows the relationship.

```
m_ok = mbit || (!UW_in && !SW_in);
m_miss = miss | ~m_ok;
```

3.3.3 Direct-Addressed Read Operation

The TLB CAM and SRAM can alternatively be read when an external address is supplied to the `addr` address lines of the select decode block. This is indicated by asserting the `asel` input. The address provided results in the corresponding CAM and SRAM entry data being driven onto the outputs. The data output access timing requirement of this operation is dictated by the levels of logic through which the output data must pass, as determined in a normal read operation.

3.3.4 Direct-Addressed Write Operation

The TLB CAM and SRAM are, as stated previously, always addressed as a single unit. However, both the CAM and SRAM have separate registered write enables, which allow either or both parts (tag and data) of an entry to be updated in the current operation. The CAM and SRAM can also be written in the same cycle, in which case both the CAM and SRAM words of the addressed entry are updated.

The standard mode in which the TLB is addressed during a write operation is via the `addr` input. The timing for both write enables (`c_we`) is the same because they are both registered inputs.

The write data should have a “fall through” path such that write data is available at output pins during the same cycle as the write operation.

3.3.5 Content-Addressed Flush Operation (Single-Entry Invalidation)

The content-addressed flush operation invalidates a single or multiple TLB entries based on the result of the tag compare. This mode is required for implementing the flush functionality, where a specific entry, identified by its tag (CAM word), needs to be invalidated (have its V bit set to 0).

Asserting the `flush` and `c_en` inputs, both registered inputs, indicates that the following cycle will be a content-addressed flush operation.

The `cd_in` inputs are to be set to cause a specific tag or tags to match, in the flush cycle. During the content-addressed flush operation, the valid bit of all entries that matched is cleared. All other bits of the corresponding entry are ignored and so can be written to any value during the content-addressed flush operation. The `miss` output is ignored by the MMU on a flush operation. The “flush clear” and “flush segment, region, …, etc.“can be done by controlling the mask bit (`cd_in` level bit). For example, flush clear can be performed by asserting mask bit to 111 and asserting flush operation. This operation causes the V bit of every entry to be flushed.

3.4 Timing Diagrams

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

This section presents timing for the following:

- TLB CAM direct-addressed read
- TLB SRAM direct-addressed read
- TLB SRAM content-addressed read
- TLB CAM / SRAM direct-addressed write
- TLB CAM flush

3.4.1 TLB CAM Direct-Addressed Read Timing

The address is supplied by `ain` (not from the match of the CAM lines).

Tcycle = **14.28 ns**
Tis = **1.5 ns** (Input setup time)
Tih = **1.5 ns** (Input hold time)
Tacd = **7.0 ns** (CAM Read Data Access time with respect to `clk`)
Tov= **0.5 ns** (Output Valid time)

TABLE 3-1 presents the truth table for TLB CAM direct-addressed read timing; FIGURE 3-5 illustrates the timing.

TABLE 3-1 TLB CAM Direct-Addressed Read Timing Truth Table

	Logical Value at Set-up to clk
`cd_in`	X
`cr`	1
`asel`	1
`addr`	00-1F
`flush`	0
`tlb_we`	0

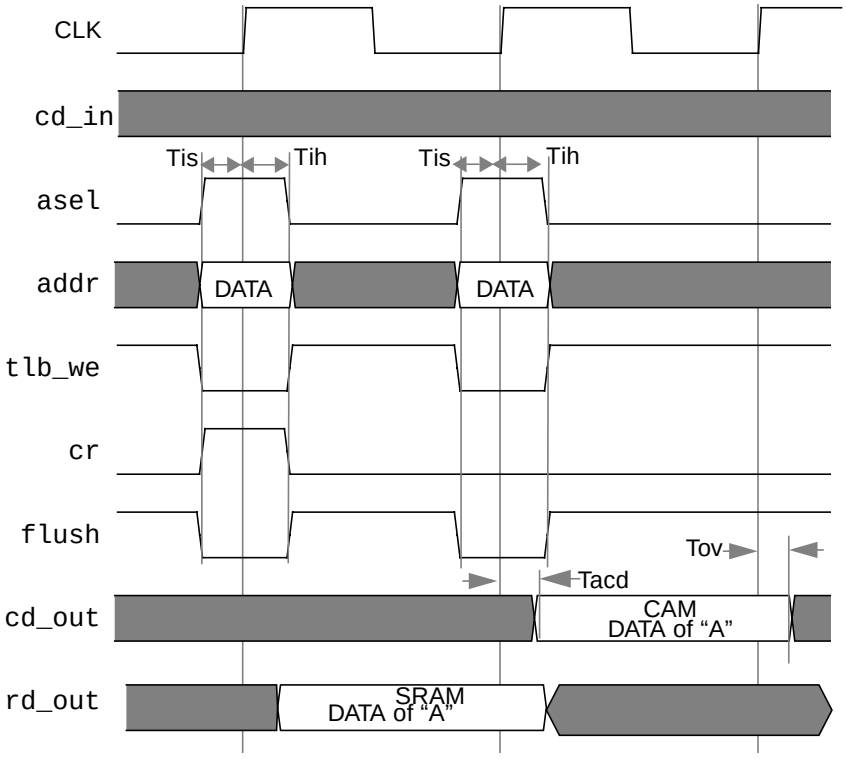

FIGURE 3-5 TLB CAM Direct-Addressed Read Timing

3.4.2 TLB SRAM Direct-Addressed Read Timing

The address is supplied by `ain` (not from the match of the CAM lines).

Tcycle = **14.28 ns**
Tis = **1.5 ns** (Input setup time)
Tih = **1.5 ns** (Input hold time)
Tacd = **10.0 ns** (CAM Read Data Access time with respect to `clk`)
Tard = **10.0 ns** (SRAM Read Data Access time with respect to `clk`)
Tov = **0.5 ns** (Output Valid time)

TABLE 3-2 presents the truth table for TLB SRAM direct-addressed read timing; FIGURE 3-6 illustrates the timing.

TABLE 3-2 TLB SRAM Direct-Addressed Read Timing Truth Table

	Logical Value at Set-up to clk
cr	0
cd_in	X
rd_in	X
asel	1
addr	00-1F
flush	0
tlb_we	0

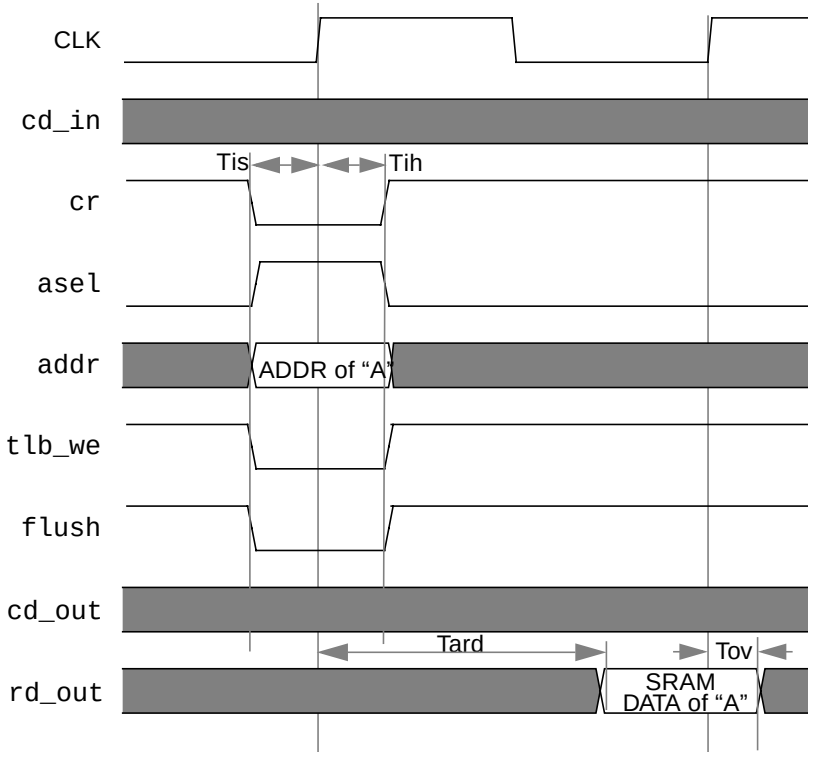

FIGURE 3-6 TLB SRAM Direct-Addressed Read Timing

3.4.3 TLB SRAM Content-Addressed Read Timing

The address is supplied by the match of the CAM tags against `c_data_in`.

Tcycle = **14.28 ns**
Tis = **1.5 ns** (Input setup time)
Tih= **1.5 ns** (Input hold time)
Tacc = **10.0 ns** (CAM Data Access time)
Tarc = **10.0 ns** (RAM Data Access time)
Tmiss = **10.0 ns** (`miss` Access time)
Terror = **10.0 ns** (error Access time)
Tov = **0.5 ns** (Output Valid time)

Notes:

- Critical path of system
- Delay of `c_data_in` needs to be SPICEd

TABLE 3-3 presents the truth table for TLB SRAM content-addressed read timing; FIGURE 3-7 illustrates the timing.

TABLE 3-3 TLB SRAM Content-Addressed Read Timing Truth Table

	Logical Value at Set-up to clk
`cr`	0
`cd_in`	Match Criteria
`rd_in`	X
`asel`	0
`addr`	X
`flush`	0
`tlb_we`	0

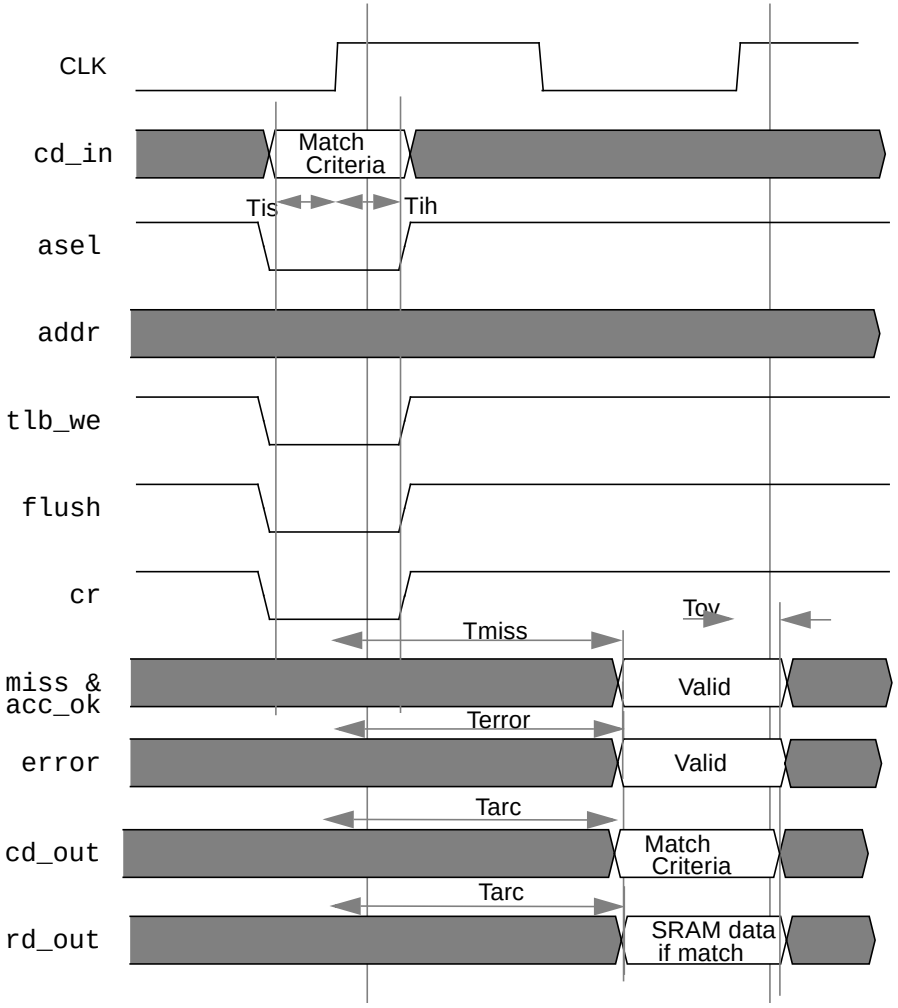

FIGURE 3-7 TLB SRAM Content-Addressed Read Timing

3.4.4 TLB CAM/SRAM Direct-Addressed Write Timing

The write enables are registered.

Tcycle = **14.28 ns**
Tis = **1.5 ns** (Input setup time)
Tih = **1.5 ns** (Input hold time)
Tas = **1.5 ns** + 1Cycle (Addr setup)

Notes:

- There are separate write enables for each the CAM and the RAM.
- The write enables have the same timing.

TABLE 3-4 presents the truth table for TLB CAM/SRAM direct-addressed write timing; FIGURE 3-8 illustrates the timing.

TABLE 3-4 TLB CAM/SRAM Direct-Addressed Write Timing Truth Table

	Logical Value as Setup to clk
cr	0
cd_in	DATA
rd_in	DATA
asel	1
addr	00-1F
tlb_we	1
flush	0

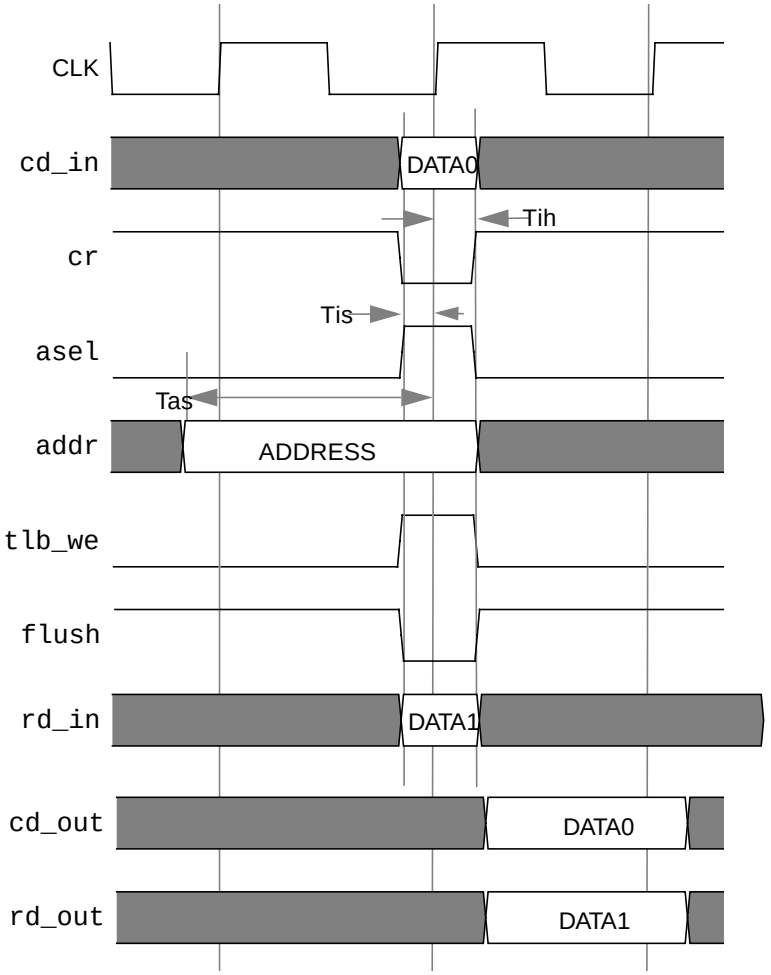

FIGURE 3-8 TLB CAM/SRAM Direct-Addressed Write Timing

3.4.5 TLB CAM Flush Timing

Tcycle = **14.28 ns** (total for flush cycle)
Tis = **1.5 ns** (Input setup time)
Tih = **1.5 ns** (Input hold time)
Tov = **0.5 ns** (Output Valid time)

Notes:

- `c_data_in` timing is same as content-addressed read.
- The only data changing is the valid bit in the CAM (being set inactive).

TABLE 3-5 presents the truth table for TLB CAM flush timing; FIGURE 3-9 illustrates the timing.

TABLE 3-5 TLB CAM_Flush Timing Truth Table

	Logical Value as setup to clk
`cr`	0
`cd_in`	Match Criteria
`rd_in`	X
`asel`	0
`addr`	X
`flush`	1
`tlb_we`	0

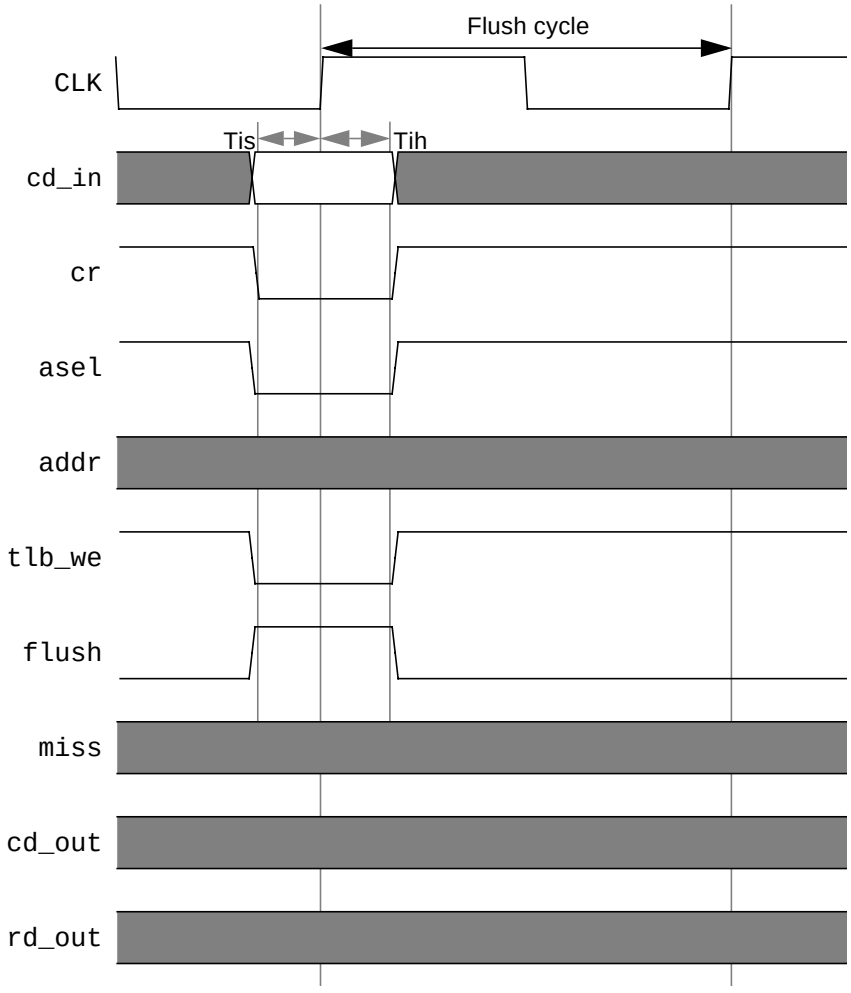

FIGURE 3-9 TLB CAM Flush Timing

3.5 TLB I/O Pin Description

This section describes I/O loading and clock input.

3.5.1 Input and Output Loading

TABLE 3-6 describes loading on the I/O pins.

TABLE 3-6 TLB I/O Pin Description

Pin Name	In/Out	# of Pins	Fanout	Cap. Load	Registered?
cd_in	In	41	-	-	Yes
tlb_we	In	1	-	-	Yes
cr	In	1	-	-	Yes
co_in	In	1	-	-	Yes
asel	In	1	-	-	Yes
addr	In	5	-	-	Yes
flush	In	1	-	-	Yes
acc_ok	Out	1	2	2.0 pF	No
miss	Out	1	2	2.0 pF	No
m_miss	Out	1	2	2.0 pF	No
error	Out	1	2	2.0 pF	No
rd_in	In	28	-	-	Yes
cd_out	Out	41	2	2.0 pF	-
rd_out	Out	28	2	2.0 pF	-
clk	In	1	-	?	No
scm	In	1	-	-	No
scn_in	In	1	-	-	No
scn_out	Out	1	1	1.0 pF	No
tg_strobe	In	1	-	-	No
pdm	In	1	-	-	No

All inputs are assumed to be buffered; otherwise, present input capacitances are not to exceed 100 fF, with the exception of the `clk` input, which has an unspecified input capacitance. Exceptions must be noted and documented; their impact on the system design will be analyzed on a case-by-case basis.

3.5.2 Clock Input

The clock input to all TLBs should be unbuffered. The input capacitance must be accurately characterized and documented upon megacell delivery.

3.6 AC Switching Characteristics

TABLE 3-7 lists timing values.

TABLE 3-7 Timing Values

Pin names	In/out	# of pins	Setup	Hold	Delay
cd_in	In	41	1.5 ns	1.5 ns	-
tlb_we	In	1	1.5 ns	1.5 ns	-
cr	In	1	1.5 ns	1.5 ns	-
co_in	In	1	1.5 ns	1.5 ns	-
asel	In	1	1.5 ns	1.5 ns	-
addr	In	5	1.5 ns	1.5ns	-
flush	In	1	1.5 ns	1.5 ns	-
rd_in	In	28	1.5 ns	1.5 ns	-
acc_ok	Out	1	-	-	10 ns
miss	Out	1	-	-	10 ns
m_miss	Out	1	-	-	10 ns
error	Out	1	-	-	10 ns
cd_out	Out	41	-	-	10 ns
rd_out	Out	28	-	-	10 ns

** Assume clock buffering will take 1.5 ns delay from clk input of tlb **

3.7 Initialization and Abnormal Conditions

The TLB must power up in a stable state and not draw excessive power during powerup or reset. The reset input is asynchronous; it must clear all of the registered inputs but not alter the contents of either the CAM or SRAM.

Abnormal conditions are as follows:

- The TLB must be able to operate in the range of frequencies from 0 MHz up to and including 70 MHz.
- The TLB may not draw excessive current or be damaged during powerup or if the clock (`clk`) is held at a high level indefinitely.
- The TLB may not draw excessive current or be damaged if more than one match line is asserted during any TLB operation.
- The TLB shall maintain all state (contents) when the clock is held high and resume normal operation when the clock resumes.

3.8 Testability

Special testability support of the TLB allows system debug using nondestructive scan and clock single-stepping. Functional testing is performed under program control and requires no special support from the TLB. Supporting scan-based debug requires some additional functionality, described in this section.

3.8.1 Functional Description of Single-Step Support

To allow scan-based system debug, all registered inputs to the TLB are loaded by JTAG-controlled scan. These inputs fall into the categories of address and control bits.

Whether the TLB is in scan mode or in normal operation mode, is determined by the state of `scan_mode`. When it is asserted, the address register and all control bits in the TLB must be connected into a single shift register.

The scan operation is not allowed to be destructive to any data in the TLB core. Therefore, the outputs of the flip-flops containing the Clear or Write Enable controls (`flush`, `tlb_we`) must be disabled during scan mode and enabled only when scan mode is exited, at which time all control bits contain their desired states for the test.

After the scan sequence is completed and the address and control bits of the TLB contain the states required for the test, the TLB is cycled once *without disturbing the state of any flip-flop in the scan chain*. This requires a separate "strobe" input to the TLB (`tg_strobe`) which starts the internal, self-timed circuits for the TLB access (either read, write, or flush clear) but does not advance the address register or control bit flip-flops. This strobe is generated externally to the TLB.

3.8.2 Scan Mode I/O

Four I/Os are required by the TLB to support testability:

- `scn_in` — Single bit input that is the scan chain (shift register) input
- `scn_out` — Single bit output that is the scan chain (shift register) output
- `scm` — Scan mode enable signal
- `tg_strobe` — Special "strobe" input, which has all the functions of `clk` except that it does not disturb the state of any flip-flops in the scan chain

3.8.3 Scan Mode Timing

Tssm = **10.0 ns** (Setup time for scn_mode with respect to `clk`)
Tssi = **1.5 ns** (Setup time for scn_in with respect to `clk`)
Tclsc = **10.0 ns** (Delay from `clk` to `scn_out`)
Ttssc = **0.0 ns** (Setup time for data_in with respect to `tg_strobe`)
Ttscl = **120.0 ns** (Setup time for `clk` with respect to `tg_strobe`)
Tclts = **40.0 ns** (Setup time for `tg_strobe` with respect to `clk`)
Tslt = **40.0 ns** (minimum `scan_mode` low to `tg_strobe` low time)
Ttw = **40.0 ns** (minimum `tg_strobe` pulse width)

3.8.4 Scan Chain Order

Scan Order -> addr[0:5] -> cd_in[23:18]->cd_in[24]

->cd_in[30:25]->cd_in[31]->cd_in[39:32]->cd_in[40]->cd_in[17:10]

->cd_in[3]->cd_in[2:1]->co_in->asel->flush->tlb_we->rst->cr->cd_in[41]->

->cd_in[9,8,6,5,7,4]->cd_in[0]->rd_in[0:27]->scn_out

A timing diagram showing the basic timing relationships between signals during scan mode is shown in FIGURE 3-10. The timing parameters during scan mode operation are substantially relaxed from those during regular operation.

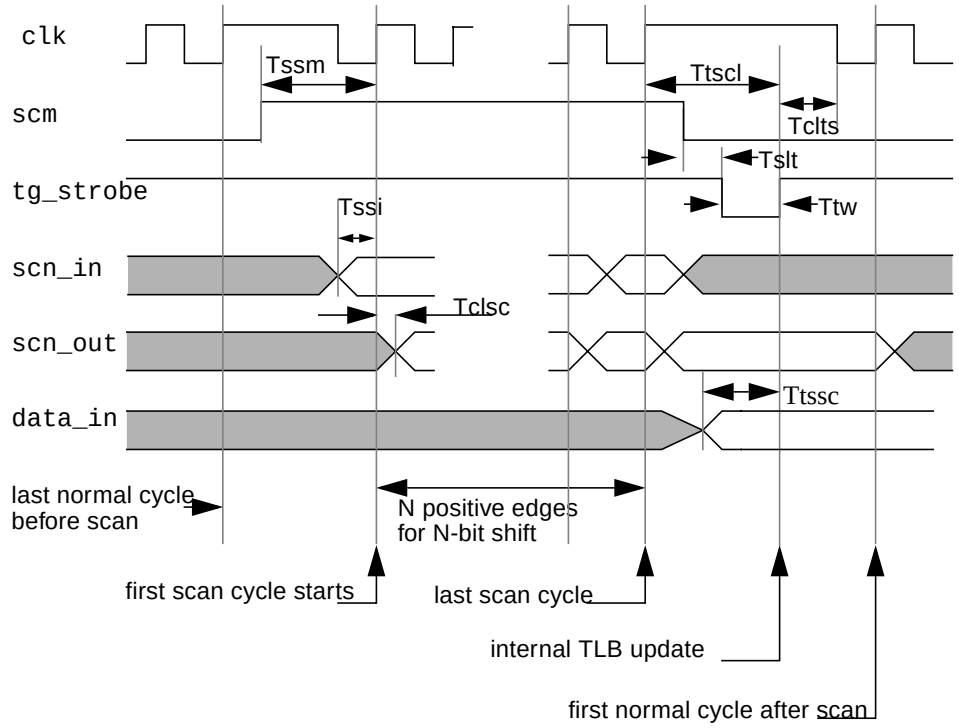

FIGURE 3-10 Scan Mode Timing

Note – `clk` is held in its high state for as long as necessary so that all events that need to occur while it is held are allowed sufficient timing margin.

Clock frequency during scan mode may not be the same frequency as before and after scan mode.

CHAPTER 4

IOTLB CAM/SRAM

This chapter describes the IOTLB CAM/SRAM with respect to:

- TLB organization overview, describing the IOTLB CAM, select block, and SRAM
- I/O signal descriptions for IOTLB CAM, IOTLB SRAM, and IOTLB decode select

Please refer to Chapter 3, *TLB CAM/SRAM* for information regarding:

- Functional description, including content-addressed read compare operation, error detection for that operation, direct-addressed read and write operations, and flush operation
- Timing diagrams for CAM read operation, SRAM direct-addressed and content addressed read operations, CAM/SRAM write timing, and the CAM flush operation
- TLB I/O pin description
- Timing Values
- Area and power estimate
- Initialization and abnormal conditions
- Testability

Please refer to Chapter 1, *Requirements*, for information, guidelines, and requirements that apply to all megacells.

Note – The TLB (Chapter 3) and PCI Controller IOTLB (Chapter 4) are each implemented as a closely coupled CAM and SRAM combination. The IOTLB CAM/SRAM combination is identical to the processor's CAM/SRAM macrocell, with the exception of the number of entries.

4.1 IOTLB Organization Overview

The PCI Controller IOTLB is implemented as a closely coupled CAM and SRAM combination. The CAM/SRAM combination is identical to the processor's CAM/SRAM macrocell, with the exception of the number of entries. The IOTLB has 16 entries devoted exclusively to I/O address translation.

No PTPs are stored in the IOTLB. All table walking and IOTLB management is done by the software operating system. The CAM contains the tag, and the SRAM contains the data portion of an IOPTE (I/O Page Table Entry). The IOCAM is a 16-entry, 42-bit-wide, fully associative array; the SRAM is a 16-entry by 28-bits array. The CAM/RAM combination is named "`io_tlb`." All models, netlists, layouts, and bounding boxes must reflect this name.

The TLB performs virtual-to-physical address translations by using a fully associative CAM lookup to index into a RAM that holds the translated address. Both the CAM and RAM sections are accessible independently for reading and writing in diagnostic or update modes. For a complete description of the functional requirements of the TLB, please refer to the *microSPARC-IIep Specification* and the *SPARC Architecture Specification* (Appendix H, Reference MMU Architecture).

FIGURE 4-1 illustrates the IOTLB CAM/SRAM. The blocks in the diagram are described below and in the subsequent subsections.

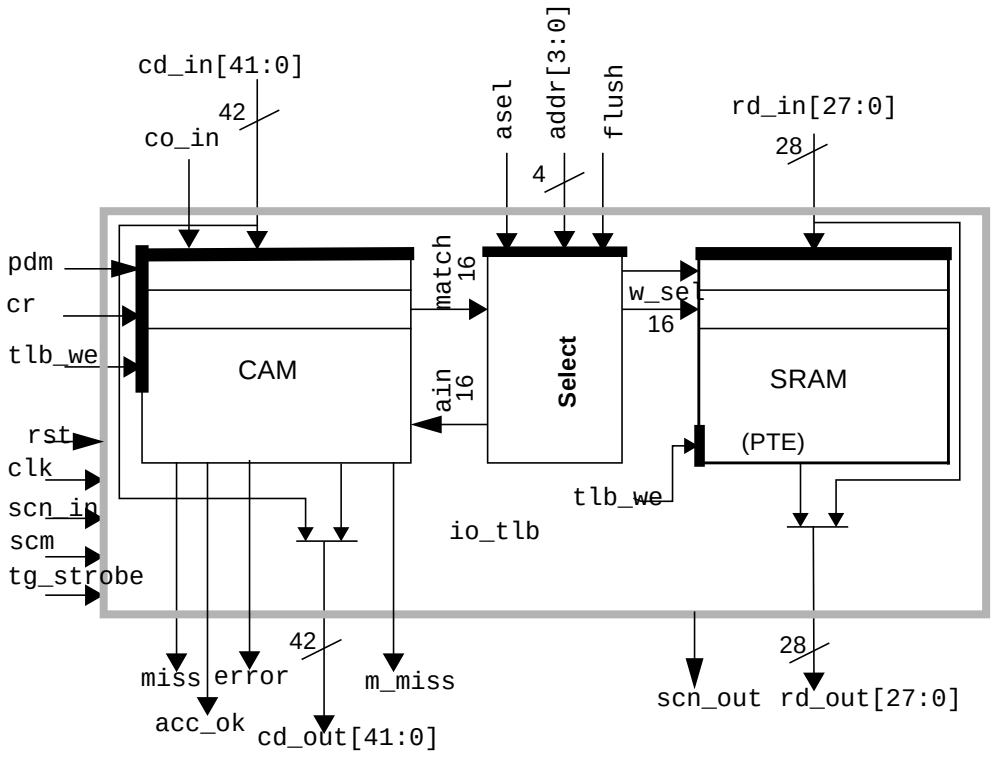

FIGURE 4-1 IOTLB Block Diagram

The IOTLB comprises tightly coupled CAM and SRAM arrays. Each of the 16 entries in the IOTLB consists of two parts: a 42-bit tag word in the CAM array and a 28-bit data word in the SRAM array.

Under the normal operation of content-addressed address translations, `cd_in` is compared simultaneously with all tag words in the CAM array. A tag can match on the basis of the values in `cd_in` and the states of mask and flag bits in each tag. On a match, the SRAM's data word for the matching entry is driven to `rd_out`. Multiple entry matches can occur, in which case the `rd_out` and `cd_out` outputs are undefined; system software attempts to prevent this condition, but it cannot be guaranteed in all cases.

4.1.1 IOTLB CAM

The IOCAM is an array of sixteen 42-bit words. It has registered `tlb_we`, `cr`, and `c_en` inputs and a 42-bit wide `cd_in` input. FIGURE 4-2 illustrates the IOTLB CAM.

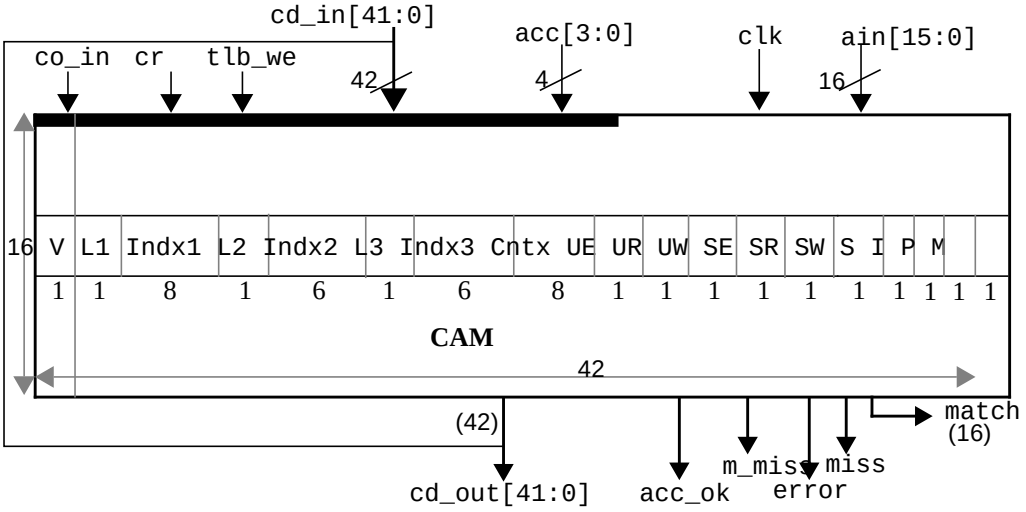

FIGURE 4-2 16-Entry Fully Associative CAM

The `L1`, `L2`, `L3` and `S` bits of the CAM word are mask bits that control whether the subfield matches of `cd_in` against the corresponding index and context fields are used in the full entry match. The `V`, `I`, and `P` bits are flag bits that enable or disable the full entry match. A Verilog-like logical expression that defines the match is shown below.

```
Entry_match = (V == V_in) &&
((L1 || L1_in) || (Indx1 == Indx1_in)) &&
((L2 || L2_in) || (Indx2 == Indx2_in)) &&
((L3 || L3_in) || (Indx3 == Indx3_in)) &&
((S || S_in) || (Cntx == Cntx_in)) &&
(P == P_in) || (co_in) &&
(I == I_in) || (co_in));
```

The `cd_in` bits corresponding to the mask and flag bits in the CAM word require specific input values during compare and write operations. These values are guaranteed to be provided by external logic. A brief explanation on how the word subfields are used follows.

Each CAM word contains four Virtual Address fields. In order of decreasing significance these are: `Context` (`CID[7:0]`), `Index1` (`VA[31:24]`), `Index2` (`VA[23:18]`), and `Index3` (`VA[17:12]`). Three of these fields have corresponding match-enable bits stored in the CAM:

- When the Supervisor bit of a CAM entry is 1, then the match on its `Context` field is forced to be true regardless of the value on the `Context` input lines.
- When the Level-1 bit of a CAM entry is 1, then the match on its `Index1` field is forced to be true regardless of the value on the `VA[31:24]` input lines of the CAM.
- When the Level-2 bit of a CAM entry is 1, then the match on its `Index2` field is forced to be true regardless of the value on the `VA[23:18]` input lines of the CAM.
- When the Level-3 bit of a CAM entry is 1, then the match on its `Index3` field is forced to be true regardless of the value on the `VA[19:12]` input lines of the CAM.

Since the Supervisor, Level-2, and Level-3 bits are implemented with XOR CAM cells that match against an input line, the corresponding input lines of the CAM are driven by external logic to the following constant values during CAM match accesses in order for the above behavior to be realized.

```
Supervisor_in = 0 for normal access, 1 for supervisor mode access
Level-1_in = 0;
Level-2_in = 0;
Level-3_in = 0;
```

In addition to the preceding four Virtual address match-enable bits, each CAM word also has Valid, PTP, and IO flag bits, which are matched against corresponding input lines. These input lines are driven by external logic to the following values during CAM match accesses.

```
Valid_in = 1;
PTP_in = 0 for translations, 1 for PTP (table walk) accesses;
IO_in = 0 for CPU accesses, 1 for I/O accesses.
```

The results of all seven of these matches (Context, Index1, Index2, Index3, Valid, PTP, and IO) are ANDed to produce the match output for each CAM entry, as described in the logic equation provided earlier.

When a CAM entry is created, the fields are written by external logic as follows:

```
 Level-3 PTE:
    Valid=1; PTP=0;
    Supervisor=0;
    Level1=0; Level2=0; Level3=0;
    Context=CID; Index1=VA[31:24]; Index2=VA[23:18]; Index3=VA[17:12].

 Level-2 PTE:
    Valid=1; PTP=0;
    Supervisor=0;
    Level1=0; Level2=0; Level3=1;
    Context=CID; Index1=VA[31:24]; Index2=VA[23:18]; Index3=dont-care.

 Level-1 PTE:
    Valid=1; PTP=0;
Supervisor=0;
Level1=0; Level2=1; Level3=1;
Context=CID; Index1=VA[31:24]; Index2=dont-care; Index3=dont-care.
PTP:
Valid=1; PTP=1;
Supervisor=1;
Level1=0; Level2=0; Level3=0
Index1=PA[27:20]; Index2=PA[19:14]; Index3=PA[13:8]; Context=PA[7:2].
```

The data output from the CAM (`cd_out`) is also required for a parallel, single-entry translation register and for diagnostic support. Additionally, the three Level bits (L1, L2, and L3) are required to control the multiplexing of the outputs from the IOTLB SRAM. This makes the data output from the CAM, specifically the L1, L2, and L3 bits, part of a critical path through the IOTLB during address translation.

The Valid bit (V) is a unique bit in the CAM, in that it supports the ability to be “flush” cleared: the Valid bit in all CAM words are cleared when the flush clear is performed. The flush clear is performed by asserting the flush operation with those mask bits set to 1 (lvl in as 111).

A logic `1` on the `miss` output from the CAM indicates that none of the entries “matched” the input. If one of the entries match the input, the `miss` output will be low.

4.1.2 IOTLB Select Block

The IOTLB CAM and SRAM require addresses provided from two (2) separate sources, to support two (2) different modes of operation. The compare mode, used during address translation, has the CAM and RAM word select lines driven by the CAM match lines. The update mode, used when updating the IOTLB (or for diagnostic access), has the CAM and RAM word select lines driven by a decode of the `addr` inputs (6 bits). The control of the address selects is performed by the address select decode block, which conceptually resides between the CAM and SRAM arrays. FIGURE 4-3 illustrates the logic for this behavior.

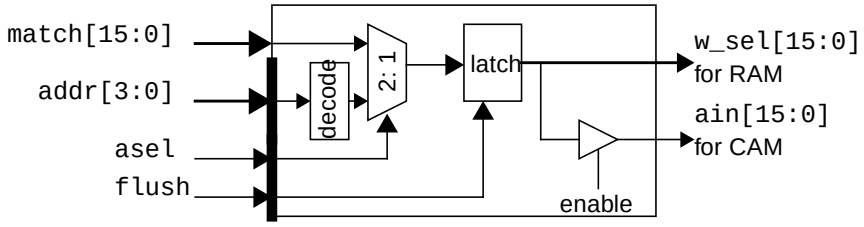

FIGURE 4-3 Address (Word) Select Decode Logic

The address select input (`asel`), if low, selects the `match` word line to be used, or, if high, the decoded address input (`addr[3:0]`). The registered `hold` input, when high, indicates that the latch should be held (that is, not transparent) in the following cycle. It is important to note that the output of the word select decode block drives the word enables for both the CAM and the SRAM arrays.

The word select for the CAM can be driven by the select Decode block when a normal content-addressed operation is not in progress. The enable is defined by the following equation:

```
CAM_wsel_enable = (asel || (flush && tlb_we));
```

4.1.3 IOTLB SRAM

IOTLB SRAM is a 16-entry by 28-bit array, illustrated in FIGURE 4-4. The `w_sel[15:0]` input directly addresses the SRAM words.

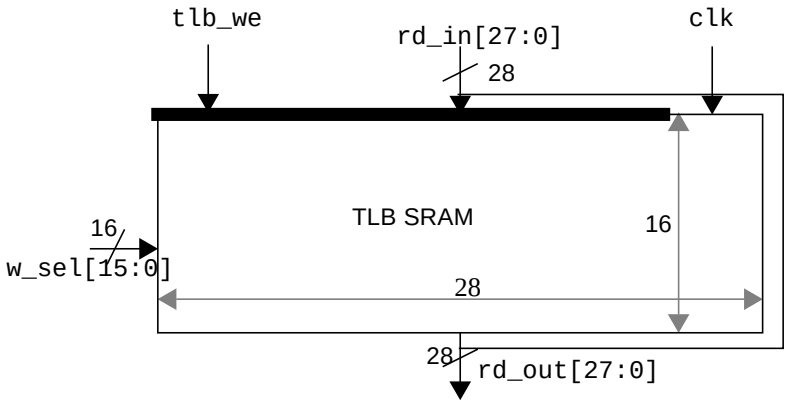

FIGURE 4-4 IOTLB SRAM Array

4.2 I/O Description

The IOCAM, IOTLB_SRAM, and IOTLB Decode Select Logic I/O are described separately below, although the IOTLB megacell (`io_tlb`) is a single layout block.

4.2.1 IOTLB CAM

`cd_in[41:0].` This input is used during both Read and Write operations of the CAM. During a content-addressed read operation, this input contains the data vector upon which a fully associative search will be performed in the CAM array to check if a match occurs. For a Write operation, this input contains the data vector to be stored at the location pointed to by `ain` when the registered `tlb_we` is active.

`ain[15:0].` This is a 16-bit input that supplies the word select lines for the CAM. It instructs the CAM array as to the location to which a data vector is to be loaded when `tlb_we` is active (direct-addressed Write), the CAM location to read during a direct-addressed Read, or the CAM entry to be invalidated during a flush operation. The `ain` lines are driven from either the decoded `addr` input or the CAM `match` lines, based on registered `asel`. These signals are used and generated only internally to the TLB.

`acc[5:0].` This input indicates the kind of operation being tried. During a content addressed read operation, this input contains the decoded operation type that will be compared to the matched CAM entries stored access privilege.

tlb_we. This signal enables a Write operation to the CAM and RAM array in the following cycle. This input is registered.

cr. This signal indicates a Read operation to the CAM array will be executed in the next two cycles. This input is registered. Note that c_rd may not be driven active unless asel is active. That means only direct addressed reads of the CAM entries are supported by the TLB. Content-addressed reads of CAM entries are not supported.

co_in. This input overrides the compare of the P bit and the I bit in the CAM. This input is registered.

m_miss. This output indicates the state of the M bit for the CAM entry that matched the cd_in criteria during a c_en compare cycle.

rst. rst is not a real reset signal to reset all register within tlb. It just tries to force TLB into TLB SRAM direct read operation state while reset is asserted. This state prevents a multiple hit on tlb entry while a reset happens.

clk. This is a single-phase clock input. If a special clocking, such as two-phase non-overlapping clocks are required, they must be generated locally to the megacell.

match[15:0]. During a Read operation, the results of an associative match are reported on this output bus, with each match bit reflecting whether the corresponding tag in the CAM array matched c_data_in (qualified by the associated flags). Under normal operation, zero or one match lines should be active at a time, although this cannot be guaranteed under all conditions. The IOTLB must ensure that if multiple match lines are active at one time, the IOTLB will not be damaged and will not output illegal CMOS voltage levels at any output, although the logical states of the outputs are irrelevant. The match signals are internal to the megacell.

cd_out[41:0]. In a content-addressed Read operation that results in a single matched entry, the contents of the matching CAM tag is output here. In a direct-addressed Read operation, the tag of the entry being addressed is output here. In all other cases, this output is ignored; however the CAM must guarantee that cd_out settles to legal CMOS signal levels within 10 ns of the specified access time for a valid access.

miss. The miss output from the CAM indicates that none of the entries matched c_data_in. If any of the entries matched, the miss output will be low.

error. The error output from the CAM indicates that the CAM that matched the c_data_in criterion has an access privilege level that will not allow the current access indicated by the asi bits to be done. If there is no access privilege error for the matched entry, the error output will be low.

acc_ok. This output indicates that the current CAM lookup is a hit with no access error and that the m bit is set appropriately for the desired access.

pdm. (Powerdown mode) This is a pin to switch off the sense amplifier when in enter powerdown mode. When powerdown mode is entered, clk stops and stays in level high. The clk high causes the tlb's sense amplifier to turn on. A separated signal pdm turns off the sense amplifier when the chip enters powerdown mode.

4.2.2 IOTLB SRAM

rd_in[27:0]. This input is used only during a Write operation. It contains the data to be written in the location selected by w_sel in the cycle tlb_we was high.

w_sel[15:0]. This 16-bit input is used during a Read or Write operation to select individual SRAM words for reading or writing. The w_sel lines are driven from either the decoded addr input or the CAM match lines. These signals are used and generated only internally to the TLB.

clk. As with the CAM array, a single-phase clock is the input. Based on circuit requirements, any special clocking scheme must be generated locally.

rd_out[27:0]. During a Read operation, rd_out will contain the contents of the SRAM word selected by w_sel[15:0]. In the case where no, or more than one, w_sel line is active, the logical values on rd_out are ignored; however, the SRAM must guarantee that rd_out settles to legal CMOS signal levels within 10 ns of the specified access time for a valid access.

4.2.3 IOTLB Decode Select

asel. This input selects either addr or match as the source of the address that generates w_sel[15:0] and ain[15:0] in the following cycle (registered input). When the registered asel is low, the match input is selected; when asel is high, the decoded addr input is selected.

addr[3:0]. This 4-bit input supplies an encoded address used for direct-address operations. This input is registered, so the address is used in the subsequent cycle from which it was supplied.

flush. This input indicates the following cycle is a PTE flush cycle. This input is registered

ain[15:0]. This is a 16-bit output that supplies the word select lines for the CAM. The ain lines can be driven from either the decoded addr input or the CAM match lines based on registered asel, or they can be held from the previous cycle when hold is active. These signals are used and generated only internal to the TLB.

w_sel[15:0]. This is a 16-bit output, that supplies the word select lines for the SRAM. The w_sel lines can be driven from either the decoded addr input or the CAM match lines, or can be held from the previous cycle when hold is active. These signals are used and generated only internal to the TLB.

CHAPTER 5

256 x 64-bit Read-Only Memory (ROM)

This chapter describes the 256 x 64-bit ROM with respect to the following areas:

- Functional Description
- I/O Description
- AC Switching Characteristics
- Area and Power Dissipation
- Layout
- Initialization and Abnormal Conditions
- Testability
- ROM Program Code

Please refer to Chapter 1, *Requirements*, for information, guidelines, and requirements that apply to all megacells.

5.1 Functional Description

The circuit is organized as 256 words of 64-bits each. The output is always enabled, and should be purely combinational.

The ROM includes an internal register for the address inputs. This register does not need a load enable (can be free-running), however it must be scannable. The order of the scan chain does not matter. `ss_clock` is a single-phase clock input.

FIGURE 5-1 illustrates the functional units of the ROM.

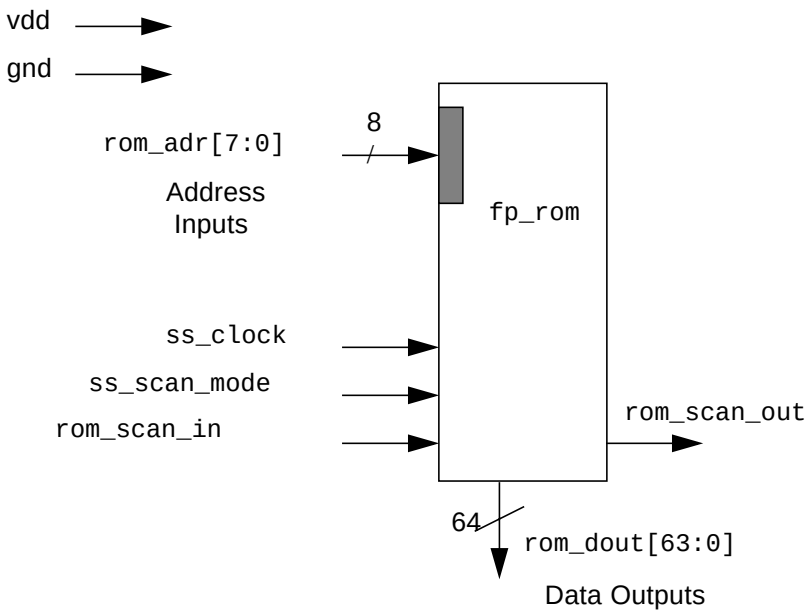

FIGURE 5-1 ROM Block Diagram

5.2 I/O Description

Following is a description of inputs and outputs shown in FIGURE 5-1 for the cell named `fp_rom`.

- **`rom_adr[7:0].`** Address inputs (bit 7 is the most-significant bit)
- **`rom_dout[63:0].`** Data outputs (bit 63 is the most-significant bit)
- **`ss_clock.`** Clock signal for address register and ROM
- **`ss_scan_mode.`** When asserted, allows `rom_scan_in` data to be scanned into the address register
- **`rom_scan_in.`** Scan data input to the address register
- **`rom_scan_out.`** Scan data output of the address register

5.3 AC Switching Characteristics

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

TABLE 5-1 summarizes timing values for the ROM. All units are in nanoseconds (ns).

TABLE 5-1 Timing Values for the ROM (with no Clock Buffer Delay)

Symbol	Parameter	Min	Typ	Max
t_{PW}	Clock pulse width	4.0	–	–
t_{ACC}	Clock to output delay	–	–	11.0
t_{C}	Read cycle time	–	–	14.0
t_{SA}	Address setup time	1.0	–	–
t_{HA}	Address hold after rising clock	0.5	–	–
t_{SD}	Clock to `rom_scan_out`	–	–	10.0
t_{SS}	`ss_scan_mode` setup	10.0	–	–
t_{SI}	`rom_scan_in` setup	1.5	–	–

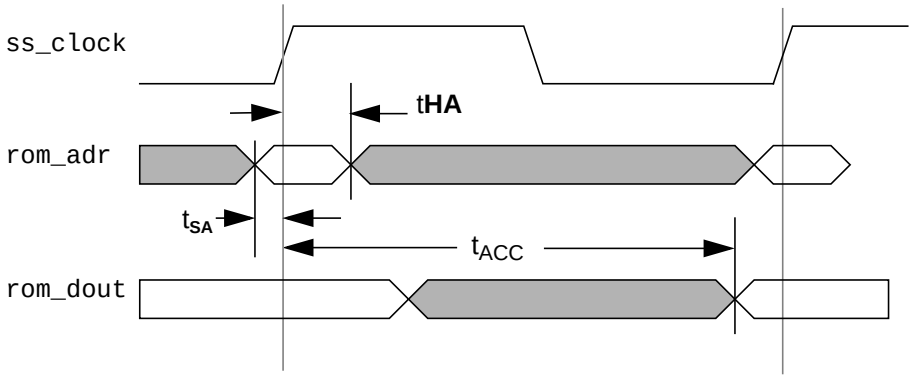

FIGURE 5-2 ROM Timing Diagram

5.3.1 Input/Output Loading

All ROM inputs should be under 100 fF in input capacitance, with the exception of the `ss_clock` input, whose input capacitance is unspecified.

All data outputs of the ROM must meet their timing specifications with a load of 0.7 pF. `rom_scan_out` must meet its timing specification with a load of 1.0 pF.

5.3.2 Clock Input

The ROM clock input should be unbuffered. The input capacitance must be accurately characterized and documented upon megacell delivery.

5.3.3 Registers

To track the rest of the chip over process and environmental variations, designers should design input and output registers of the ROM by using flip-flops similar to those used in the standard cell library. The flip-flops have a single clock input; buffer it internally before using it for either master or slave sections.

5.4 Area and Power Dissipation

The area usage of this block is expected to be less than 1 sq. mm.

Power dissipation under normal operation (one access every 14 ns) is expected to be under 0.1 watts.

5.5 Layout

The bounding box and approximate terminal locations are not yet determined for this array.

5.6 Initialization and Abnormal Conditions

The ROM is mask programmed at the foundry and does not require any special initialization.

The ROM must not draw excessive current during powerup.

After powerup, the output of the ROM may be unknown until the first cycle after a valid address has been clocked in.

5.7 Testability

On assertion of `ss_scan_mode`, the address input register must convert itself into a simple shift register, whose input is `rom_scan_in` and whose output is `rom_scan_out`.

The application includes an external register for the data outputs. The ROM is tested by scanning an address into the ROM's address input register and scanning out the data (from the external register) for each word in the ROM.

5.8 ROM Program Code

You must program these data into the ROM.

```
mem[0] = 64'h0000000000000000;
mem[1] = 64'h00000011100085c4;
mem[2] = 64'h00000041100085a4;
mem[3] = 64'h0000000000000000;
mem[4] = 64'h03103010c804f865;
mem[5] = 64'h0020800113800000;
mem[6] = 64'h00000000000085ac;
mem[7] = 64'h121200000000806c;
mem[8] = 64'h041440404800f865;
mem[9] = 64'h0020800113800000;
mem[10] = 64'h05145051c800f865;
mem[11] = 64'h0020800113800000;
mem[12] = 64'h000000100000860c;
mem[13] = 64'h121200512000807c;
```

```
mem[14] = 64'h000000100000860c;
mem[15] = 64'h12120000000081dc;
mem[16] = 64'h00000000000080a4;
mem[17] = 64'h00000000000087dc;
mem[18] = 64'h0000004110000000;
mem[19] = 64'h00000001e8001dac;
mem[20] = 64'h002080111000d57f;
mem[21] = 64'h0020800123800000;
mem[22] = 64'h001430000000c9cd;
mem[23] = 64'h0010300188000000;
mem[24] = 64'h0a14507968037040;
mem[25] = 64'h0300001968200000;
mem[26] = 64'h0a145079680371f0;
mem[27] = 64'h0300000168200000;
mem[28] = 64'h0a14507968037040;
mem[29] = 64'h0300001968200000;
mem[30] = 64'h0a145079680371f0;
mem[31] = 64'h0300000168200000;
mem[32] = 64'h0014000048000000;
mem[33] = 64'h0a1430000000c1cc;
mem[34] = 64'h0a000000000069cc;
mem[35] = 64'h0200000000008254;
mem[36] = 64'h0014010048000000;
mem[37] = 64'h0c1850518000c2f4;
mem[38] = 64'h0c1850000000c83c;
mem[39] = 64'h02000018000082fe;
mem[40] = 64'h0a14507968037040;
mem[41] = 64'h0300001968200000;
mem[42] = 64'h0a145079680371f0;
mem[43] = 64'h0300000168200000;
mem[44] = 64'h0a14507968037040;
mem[45] = 64'h0300001968200000;
mem[46] = 64'h0a145079680371f0;
mem[47] = 64'h0300000168200000;
mem[48] = 64'h8a1030000000842c;
mem[49] = 64'h80103380000080b4;
mem[50] = 64'h801030780000842c;
mem[51] = 64'h82103390680080b4;
mem[52] = 64'h801033800000866c;
mem[53] = 64'h8a103380000086c4;
mem[54] = 64'h80103380000086c4;
mem[55] = 64'h0a1433800000c90c;
mem[56] = 64'h001430000000c77e;
mem[57] = 64'h801030000000ea55;
mem[58] = 64'h0014008188000000;
mem[59] = 64'h1212005120008624;
mem[60] = 64'h0000000000000001;
```

```
mem[61] = 64'h0000000001000000;
mem[62] = 64'h001450000000c737;
mem[63] = 64'h0000000001000000;
mem[64] = 64'h8a1030780000861c;
mem[65] = 64'h80103000000080b4;
mem[66] = 64'h801030000000861c;
mem[67] = 64'h82103018000080b4;
mem[68] = 64'h801030000000812c;
mem[69] = 64'h00000000000080b4;
mem[70] = 64'h82103018000080b4;
mem[71] = 64'h0a1430000000c90c;
mem[72] = 64'h001430000000c9cc;
mem[73] = 64'h000000000000977e;
mem[74] = 64'h04100017a0000000;
mem[75] = 64'h0300029188180001;
mem[76] = 64'h001400004b400000;
mem[77] = 64'h5e10001140000000;
mem[78] = 64'h5810501140000000;
mem[79] = 64'h5e95501120008470;
mem[80] = 64'h5e95500000000000;
mem[81] = 64'h80103027c8003cfc;
mem[82] = 64'h10100047e00022a4;
mem[83] = 64'h0514000000000000;
mem[84] = 64'h0a000011a0008268;
mem[85] = 64'h2018501140000000;
mem[86] = 64'h5e10001140008470;
mem[87] = 64'h5810500000000000;
mem[88] = 64'h8300039160128350;
mem[89] = 64'h03000017c8100000;
mem[90] = 64'h0000028000000000;
mem[91] = 64'h0000007800008244;
mem[92] = 64'haa120391680283f0;
mem[93] = 64'hca18401720000000;
mem[94] = 64'h0c1850518000c605;
mem[95] = 64'h0020800002800000;
mem[96] = 64'h0014008048000000;
mem[97] = 64'h80103027c8003cfc;
mem[98] = 64'h10100047e0002324;
mem[99] = 64'h0514000000000000;
mem[100] = 64'h0a000011a00082b0;
mem[101] = 64'h2018501140000000;
mem[102] = 64'h0300000000108448;
mem[103] = 64'h0300000000100000;
mem[104] = 64'h8300039160120c48;
mem[105] = 64'h03000017c8100000;
mem[106] = 64'h0300000000108330;
mem[107] = 64'h0300000000100000;
```

```
mem[108] = 64'haa120391680283f8;
mem[109] = 64'hca18401720000000;
mem[110] = 64'h001430000000c9cd;
mem[111] = 64'h0010300003800000;
mem[112] = 64'h8a103380000083d4;
mem[113] = 64'h8010338000008374;
mem[114] = 64'h801033800000870c;
mem[115] = 64'h82103380000080b4;
mem[116] = 64'h8a103078000086dc;
mem[117] = 64'h8010338000008374;
mem[118] = 64'h82103380000080b4;
mem[119] = 64'h0a1433800000c90c;
mem[120] = 64'h001430000000c254;
mem[121] = 64'h00000000000081cc;
mem[122] = 64'h0c1850118000c6ee;
mem[123] = 64'h03000010c8040000;
mem[124] = 64'haa12001168000bf8;
mem[125] = 64'hca18401720000000;
mem[126] = 64'h78c0400000000000;
mem[127] = 64'h78c0404380000000;
mem[128] = 64'h78c04041e800e400;
mem[129] = 64'h78c0400000000000;
mem[130] = 64'h01105011c800c454;
mem[131] = 64'h041500004800f865;
mem[132] = 64'h0020800113800000;
mem[133] = 64'h0c1853918000c37e;
mem[134] = 64'h03000010c8040000;
mem[135] = 64'h8300001160108b4c;
mem[136] = 64'h03000017c8100000;
mem[137] = 64'h0700000000000000;
mem[138] = 64'h041440104800f865;
mem[139] = 64'h0020800113800000;
mem[140] = 64'h0b0000158018ae4c;
mem[141] = 64'h0414101720008654;
mem[142] = 64'h5e95500000002a54;
mem[143] = 64'h5e955011c800e470;
mem[144] = 64'hde95500000000000;
mem[145] = 64'h101800106800dc9c;
mem[146] = 64'h1013400000000000;
mem[147] = 64'h03105000c804c454;
mem[148] = 64'h041500004800f865;
mem[149] = 64'h0020800113800000;
mem[150] = 64'h001850000000ccc4;
mem[151] = 64'h001430000000c736;
mem[152] = 64'h0a145079680171f0;
mem[153] = 64'h0300000168200000;
mem[154] = 64'h88103877c8009d5c;
```

```
mem[155] = 64'h00000000000087b4;
mem[156] = 64'h8810387728003f6c;
mem[157] = 64'h041400172800f865;
mem[158] = 64'h0020800113800000;
mem[159] = 64'h080000604800e2d4;
mem[160] = 64'h041850118000c37e;
mem[161] = 64'h03000010c8040000;
mem[162] = 64'h00000027c8008a94;
mem[163] = 64'h0000000000008314;
mem[164] = 64'h001850000000c724;
mem[165] = 64'h001430000000c54c;
mem[166] = 64'h0000000000003546;
mem[167] = 64'h0000000000006d4d;
mem[168] = 64'h0000000001c00000;
mem[169] = 64'h020000000000858c;
mem[170] = 64'h88103077c8001fb4;
mem[171] = 64'h00000015c8009f9c;
mem[172] = 64'h0000004000000000;
mem[173] = 64'h0300000168202e34;
mem[174] = 64'h03000408c81c0001;
mem[175] = 64'h002080004b800000;
mem[176] = 64'h001430600000c1fe;
mem[177] = 64'h0000000000003546;
mem[178] = 64'h000000000000ed47;
mem[179] = 64'h0000000001800000;
mem[180] = 64'h00000001e800985e;
mem[181] = 64'h0000004180000000;
mem[182] = 64'h0300000168200000;
mem[183] = 64'h04142000000087cc;
mem[184] = 64'h000000000000bd7e;
mem[185] = 64'h0411101188000000;
mem[186] = 64'h00208011c800d73f;
mem[187] = 64'h002080004a000000;
mem[188] = 64'h0c1850158000c5fe;
mem[189] = 64'h03000010c8040000;
mem[190] = 64'h0200000048009fad;
mem[191] = 64'h0014000003800000;
mem[192] = 64'h041030100000977e;
mem[193] = 64'h000002800000b77e;
mem[194] = 64'h000003000000877e;
mem[195] = 64'h101850418000c77e;
mem[196] = 64'h0a14507968017040;
mem[197] = 64'h0300001968200000;
mem[198] = 64'h04141400000080a0;
mem[199] = 64'h03000008c81c0000;
mem[200] = 64'h0b00081580182c6c;
mem[201] = 64'h001410172000c37e;
```

```
mem[202] = 64'h03000010c8040000;
mem[203] = 64'h001400004800f865;
mem[204] = 64'h0020800113800000;
mem[205] = 64'h021430000000c37e;
mem[206] = 64'h001430100000c37e;
mem[207] = 64'h04110000000085bc;
mem[208] = 64'h00000000000081fe;
mem[209] = 64'h8010300000008584;
mem[210] = 64'h00000060000081fe;
mem[211] = 64'h8210301800008584;
mem[212] = 64'h80103000000084b4;
mem[213] = 64'h8010300000008584;
mem[214] = 64'h8210301800008584;
mem[215] = 64'h8010300000008524;
mem[216] = 64'h0a1850000000c9cc;
mem[217] = 64'h001850000000c8be;
mem[218] = 64'h0000000000008254;
mem[219] = 64'h0c1853800000cef4;
mem[220] = 64'h0c1850000000c255;
mem[221] = 64'h001030018b000000;
mem[222] = 64'h0c1850518000c37e;
mem[223] = 64'h03000010c8040000;
mem[224] = 64'h0a000070680083dc;
mem[225] = 64'h0c1850718000c37e;
mem[226] = 64'h03000010c8040000;
mem[227] = 64'h0a000070680083e4;
mem[228] = 64'h001430000000cd8c;
mem[229] = 64'h00000000000091ff;
mem[230] = 64'h0000000000000000;
mem[231] = 64'h0020800122000000;
mem[232] = 64'h00000877c8003f55;
mem[233] = 64'h0014000723800000;
mem[234] = 64'h881030786800e0b4;
mem[235] = 64'h04185011e000c37e;
mem[236] = 64'h0000001720008654;
mem[237] = 64'h801030106800e784;
mem[238] = 64'h001850100000ce7d;
mem[239] = 64'h001400004b800000;
mem[240] = 64'h001430000000c0be;
mem[241] = 64'h04100017a000ea54;
mem[242] = 64'h030002918818877e;
mem[243] = 64'h03000468c81cafec;
mem[244] = 64'h80103000680065e4;
mem[245] = 64'h0410301000008266;
mem[246] = 64'h000002806800e7c4;
mem[247] = 64'h001850100000c37e;
mem[248] = 64'h041100100000877e;
```

```
mem[249] = 64'h002080000000d2ff;
mem[250] = 64'h0020800182800000;
mem[251] = 64'h002080111000d5c4;
mem[252] = 64'h00000011200085c4;
mem[253] = 64'h0410001188000001;
mem[254] = 64'h001110004bc00000;
mem[255] = 64'h0000000000000000;
```

CHAPTER 6

136 x 32-bit Register File

This chapter describes the 136 x 32-bit register file with respect to the following areas:

- Functional description
- I/O signal description
- AC switching characteristics, including loading and read/write timing
- Area and power dissipation
- Layout
- Testability
- Translation, describing the logical-to-physical translation of the register file

Please refer to Chapter 1, *Requirements*, for information, guidelines, and requirements that apply to all megacells.

6.1 Functional Description

This register file used by the Integer Unit from which operands are read and results are written. It is organized as an array of 136 words of 32-bits each and includes address translation logic to implement a windowed SPARC register file of 8 windows. There are three independent read ports and one independent write port that allow three 32-bit values to be read and one 32-bit value to be written simultaneously. The cycle time of the register file is 14 ns.

A store aligner is to be provided on one of the read ports for the SPARC store byte, halfword, and word instructions. A load aligner to support SPARC load byte and load halfword operations is not required to be part of the register file as this function will be handled externally.

FIGURE 6-1 illustrates the 136 by 32-bit register file.

Cell Name: Mregfile

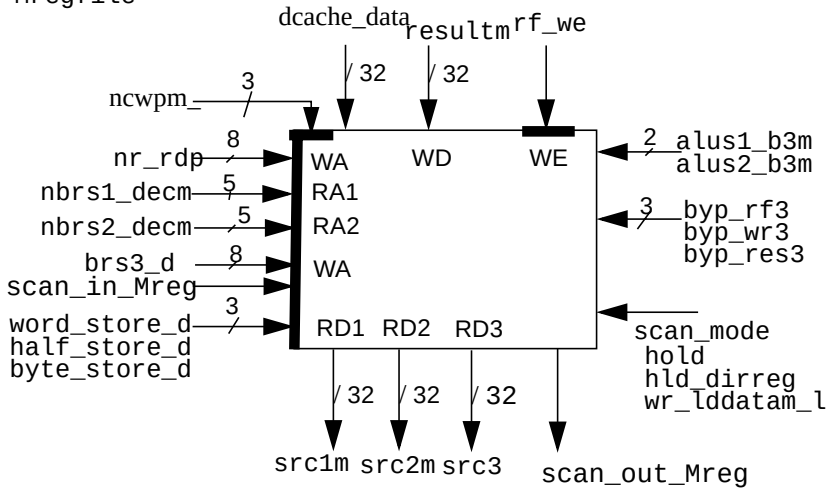

FIGURE 6-1 136 x 32 Register File Block Diagram

The three read addresses and one write address are provided to the register file synchronously. Two of read addresses (for read ports 1 and 2) are logical, 5-bit addresses. The logical read addresses and the Current Window Pointer pass through the logical-to-physical address translations portion of the register file where the physical addresses are generated. The third read port's address (read port 3) is provided as an 8-bit physical address, so no translation is required.

The read operation starts as soon as the physical addresses are available. In the case of read ports 1 and 2, this is after the logical addresses have been translated to physical addresses. In the case of read port 3, the read can begin immediately.

The write must take place during the second half of the master clock cycle (when the master clock is low). The write enable signal and write data are registered by the register file. In the case of load data from the data cache, the write data is passed through the load aligner and then registered by the register file.

This register file does not incorporate a load data aligner. This function is handled externally to the register file.

Because the integer unit is a pipelined design that requires bypassing of operands from earlier instructions in the pipe, we require that certain bypass paths be built into the register file (see FIGURE 6-2). The first of these (called bypass 3) is to bypass data from the write data register of the register file to each of the three independent read ports (for port 3, via the store aligner). If any of the read ports' bypass 3 is asserted, the write must still occur to the correct register. If there is an attempt to read a register that is being written at the same time, we guarantee that bypass 3 or another external bypass will be enabled for that read port.

The second of these bypasses (called bypass 2) is to bypass data from the ALU output register to read port 3 (via the store aligner) only. (Bypass 2 for ports 1 and 2 is handled externally to the register file.) This data must still be correctly registered by the register file on the next rising edge of clock (if hold is inactive).

Writes to register 0 (in SPARC, register %g0) produce no effect on data read from register 0. Data read from register 0 should always be 0.

All address and data registers and the write enable register in the register file must be scannable through the `scan_mode` signal and holdable (retain its value) through a hold signal. In addition, `scan_mode` must disable the write enable.

FIGURE 6-2 illustrates a simplified register block diagram.

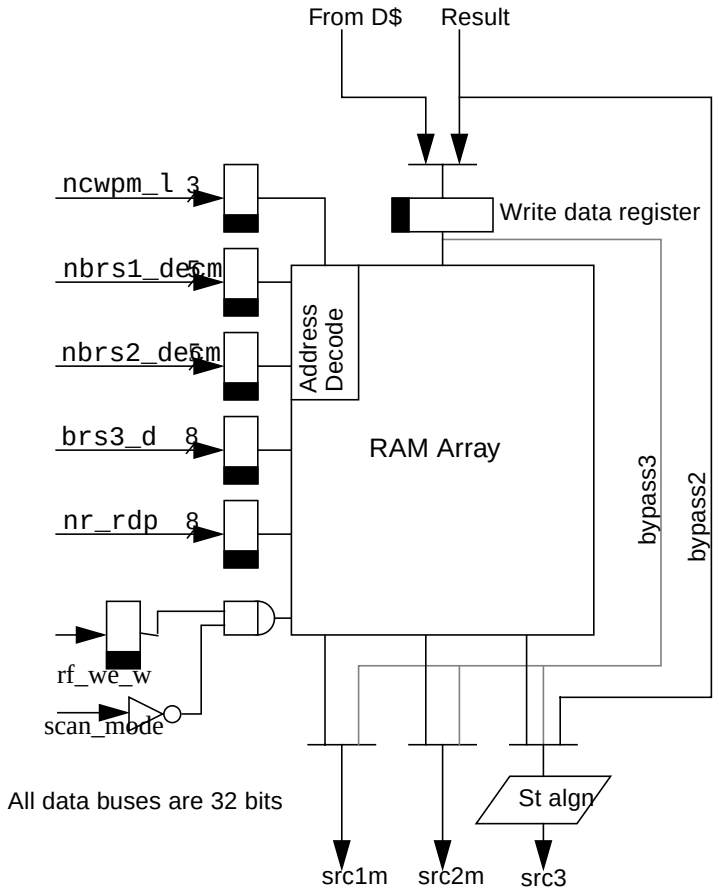

FIGURE 6-2 136 x 32 Register File Functional Block Diagram

In FIGURE 6-2, controls for bypass muxes and aligners are not shown. All registers are scannable via scan_mode and holdable via hold.

6.2 I/O Description

Below are described the I/O signals illustrated in FIGURE 6-1 and FIGURE 6-2.

src1m[31:0]. Register file read port 1 data

src2m[31:0]. Register file read port 2 data

src3[31:0]. Register file read port 3 data

resultm[31:0]. Register file result data port

dcache_data[31:0]. Register file data-cache data port (after load alignment)

nbrs1_decm[4:0]. Register file logical address for read port 1

nbrs2_decm[4:0]. Register file logical address for read port 2

brs3_d[7:0]. Register file physical address for read port 3

nr_rdp[7:0]. Register file physical address for write port

word_store_d. Store aligner control to indicate a full word store

half_store_d. Store aligner control to indicate a halfword store

byte_store_d. Store aligner control to indicate a byte store

byp_rf3. Read port 3 select from register file array

byp_res3. Read port 3 select from W cycle register file input

byp_wr3. Read port 3 select from R cycle register

byp_2. Bypass 2 enable for read port 3

ncwpm_[2:0]. SPARC Current Window Pointer (active low)

rf_we_w. Active high write enable. Do not write the register file if this signal is low.

scan_mode. Enables scan for all registers and disables write enable

hold. Prevents registers from propagating values from D to Q. Registers that this signal holds include rfwrdata, r_rdpm, cwpm_, rs3_phys_e, word_store_e, half_store_e, byte_store_d, and rf_we_r_almost

hld_dirreg. Special hold for holding brs1_decm and brs2_decm registers

wr_lddatam_l. When ON, selects resultm as RF write data. When OFF, selects yc

scan_in_Mreg. Scan input port

scan_out_Mreg. Scan output port

6.3 AC Switching Characteristics

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

TABLE 6-1 summarizes timing parameters. All units are in nanoseconds (ns).

TABLE 6-1 Timing Parameter Summary

Symbol	Parameter	Min	Maz
Taa	Clock to Output (src1/src2) delay	-	6.0
Tas	Address setup time to clock	1.3	-
Tah	Address hold time to clock	-	0.8
Tbs	Bypass setup time to clock	1.3	
Tbh	Bypass hold time to clock	-	0.8
Tba	Bypass enable to output valid	-	2.2
Trc	Read cycle time	14.2	-
Twc	Write cycle time	14.2	-
Tas	Write address setup time to clock	1.3	-
Tah	Write address hold time to clock	-	0.8
Tws	Write enable setup time to clock	1.3	-
Twh	Write enable hold time to clock		0.8
Twa	Write data to Src3 Output valid		2
Tdcs	D$ data setup time to clock	1.3	-
Tdch	D$ data hold time to clock	-	0.8
Trs	Result select setup time to clock	1.3	-

TABLE 6-1 Timing Parameter Summary

Symbol	Parameter	Min	Maz
Trh	Result select hold time to clock		0.8
Trds	Result data setup time to clock	1.3	-
Trdh	Result data hold time to clock		0.8
Taa2	Clock to output (src3) delay	-	6.9

6.3.1 Loading Conditions

The outputs of the register file drive a load of tbd pF. All timing parameters in this specification must be met under these conditions.

6.3.2 Read Case

The IU control logic supplies the read addresses for register file read ports 1 and 2 in logical form as a window pointer (`cwpm_`) and two register numbers (`brs1_decm` and `brs2decm`). A delay occurs after the read addresses are valid, defined as Taa; the read port data outputs should be valid.

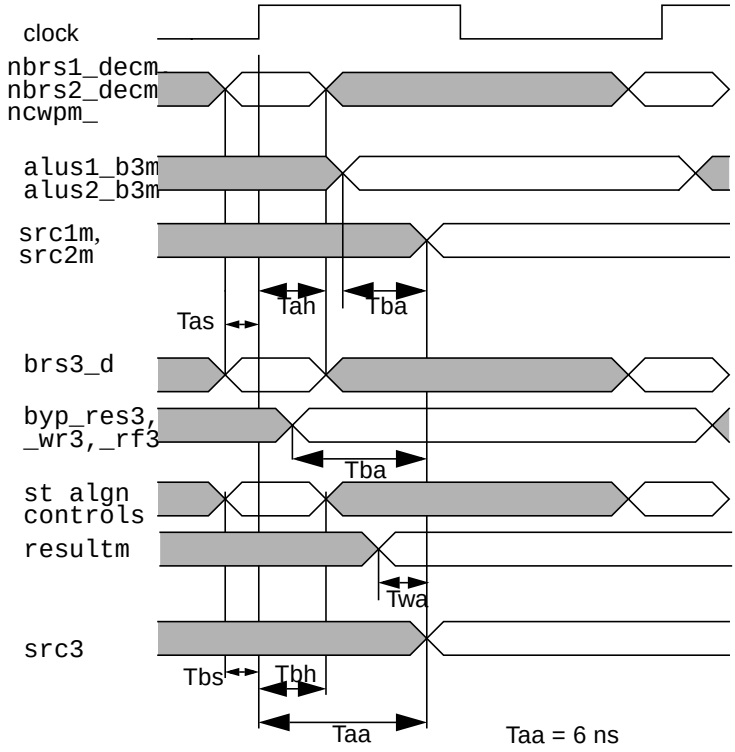

FIGURE 6-3 136 x 32 Register File Read Timing

6.3.3 Write Case

Assuming the minimum pulse width of clock is 6 ns, a write to the register file has to be done in 6 ns because pulse shrinkage through clock gating to form the write pulse takes up to 2 ns.

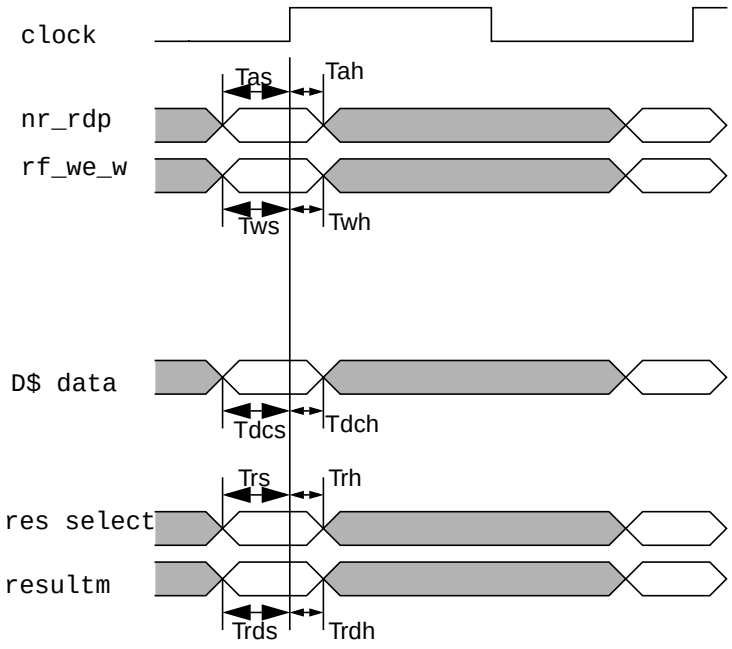

FIGURE 6-4 136 x 32 Register File Write Timing

Writes can occur each cycle. On each non-held positive edge of clock, `rf_we_w`, write address, and mux/aligner controls are registered. If the registered value of `rf_we_w` is 1, then the register location given by the write address is written in the second half of the clock cycle (clock low). The result select mux control (selects between ALU data or Data Cache data) is used to properly select incoming data.

6.4 Area and Power Dissipation

The area usage of this block is expected to be 2.94 sq. mm or less.

6.5 Layout

FIGURE 6-5 illustrates register file layout.

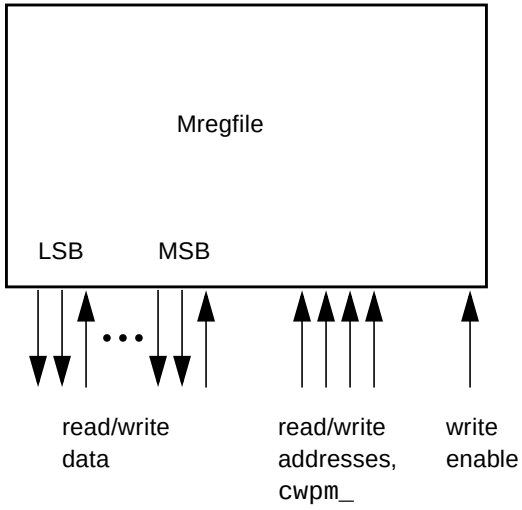

FIGURE 6-5 136 x 32 Register File Layout

6.6 Initialization and Abnormal Conditions

The register file requires no special initialization.

The register file must not draw excessive current or be exposed to potential damage during system powerup and must be able to operate to these specifications at any access frequency at or under 70.2 MHz, including zero frequency.

6.7 Testability

The register file is on the IU datapath. It is our experience that testing datapath elements is not difficult with functional code. We believe that this strategy will not cause loss of test coverage.

Designers who want to functionally test the register logical-to-physical address translation should refer to TABLE 6-2, below.

The scan chain order is shown below, running from `scan_in_Mreg` to `scan_out_Mreg`.

```
r_rdpm[7} downto r_rdpm[1]
rs3_phys_e[7] downto rs3_phys_e[1]
brs2_decm[4] downto brs2_decm[1]
cwpm_[2] downto cwpm_[0]
brs1_decm[4] downto brs1_decm[0]
brs2_decm[0]
rs3_phys_e[0]
byte_store_e
word_store_e
half_store_e
rf_we_r_almost
r_rdpm[0]
rfwrdata[31] downto rfwrdata[0]
```

6.8 Translation

The logical-to-physical translation of the register file is presented in TABLE 6-2.

TABLE 6-2 136 x 32 Register File Logical-to-Physical Translation

CWP (active low)	Logical address[4:3]	Physical address[7:3]
XXX	00	00000
000	01	00010
000	10	00001
000	11	10000
001	01	00100
001	10	00011
001	11	00010
010	01	00110
010	10	00101
010	11	00100

TABLE 6-2 136 x 32 Register File Logical-to-Physical Translation (Continued)

CWP (active low)	Logical address[4:3]	Physical address[7:3]
011	01	01000
011	10	00111
011	11	00110
100	01	01010
100	10	01001
100	11	01000
101	01	01100
101	10	01011
101	11	01010
110	01	01110
110	10	01101
110	11	01100
111	01	10000
111	10	01111
111	11	01110

The final physical address is the four physical address bits from the preceding table concatenated with the three low order bits of the logical address. Thus, if the CWP (active low) = 101 and the logical address = 10101, then the physical address would be 01011_101.

Note that when two high order bits of the logical address = 00, the four bits from the table will be 00000 regardless of the CWP. Example: CWP = 010, logical address = 00111, the physical address will be 00000_111.

Multiple encodings of CWP and high order bits of the logical address yield the same 4-bit result. This is the result of SPARC overlapping register windows.

CHAPTER 7

16 x 64 Register File

This chapter describes the 16 x 64 register file, with respect to the following areas:

- Functional description, including block diagrams of the register file
- I/O signal description
- AC switching characteristics, including loading conditions and bypass
- Area and power dissipation
- Initialization and abnormal conditions
- Testability

Please refer to Chapter 1, *Requirements*, for information, guidelines, and requirements that apply to all megacells.

7.1 Functional Description

The circuit is organized as a synchronous 4-port register file (3 read, 1 write) with 16 words of 64-bits each. The ports are fully independent, allowing three 64-bit values to be read and one 64-bit value to be written each cycle. The write operation is controlled by 2 write enable signals to allow independent control of the write of the most significant 32 bits and least significant 32 bits. Internal bypass circuitry allows the most significant 32 bits and least significant 32 bits of the write data to be bypassed independently to any or all three read ports. This allows a 64-bit read to occur while a 32-bit write to the same register is occurring (32 bits will be bypassed and 32 bits will be read from the register that is not being written).

FIGURE 7-1 and FIGURE 7-2 illustrate the 16 x 64-bit register and the internals of the FPU register file, respectively.

Cell Name: `fp_rf`

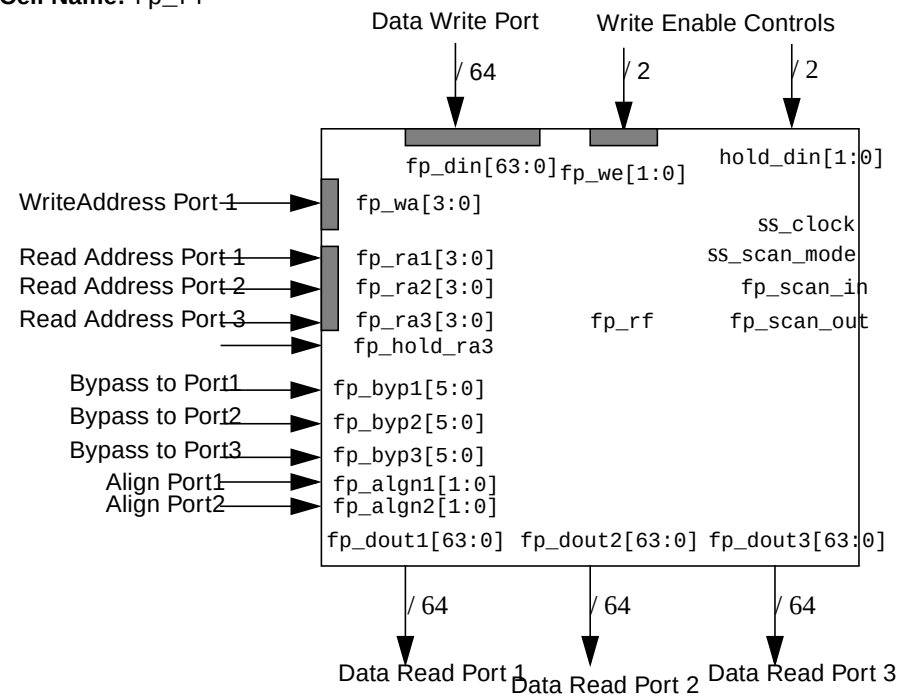

FIGURE 7-1 16 x 64-bit Register File Block Diagram

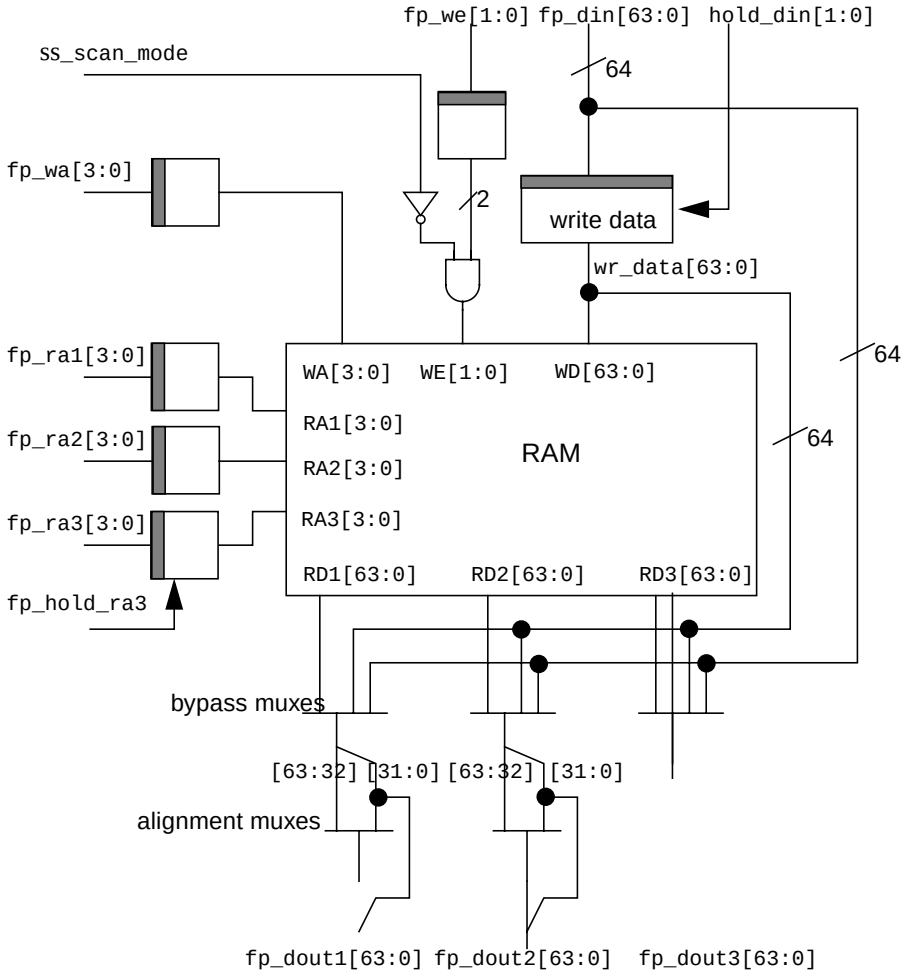

FIGURE 7-2 FPU Register File Internal Block Diagram

Note – For simplicity, FIGURE 7-2 does not show the mux controls, clock, and scan connections between registers.

7.2 I/O Description

The I/O signal description for the cell named `fp_rf` is presented below.

`fp_we``[1]`. Write enable control for most significant 32 bits – `fp_din[63:32]`

`fp_we``[0]`. Write enable control for least significant 32 bits – `fp_din[31:0]`

`fp_wa``[3:0]`. Write port address (bit 3 is most significant bit)

`fp_ra1``[3:0]`. Port 1 read address (bit 3 is most significant bit)

`fp_algn1``[1:0]`. Port 1 alignment control

`fp_byp1``[5:0]`. Bypass write data `fp_din` to port1 `fp_dout1`

`fp_ra2``[3:0]`. Port 2 read address (bit 3 is most significant bit)

`fp_algn2``[1:0]`. Port 2 alignment control

`fp_byp2``[5:0]`. Bypass write data `fp_din` to port2 `fp_dout2`

`fp_ra3``[3:0]`. Port 3 read address (bit 3 is most significant bit)

`fp_hold_ra3`. Port 3 read address hold control (1==hold previous value)

`fp_byp3``[5:0]`. Bypass write data `fp_din` to port3 `fp_dout3`

`fp_din``[63:0]`. Write port data input (bit 63 is most significant)

`fp_dout1``[63:0]`. Read port 1 data output (bit 63 is most significant)

`fp_dout2``[63:0]`. Read port 2 data output (bit 63 is most significant)

`fp_dout3``[63:0]`. Read port 3 data output (bit 63 is most significant)

`hold_din``[1:0]`. Write data register hold control input (1==hold previous value)

`ss_clock`. Clock input

`ss_scan_mode`. Scan enable input

`fp_scan_in`. Scan data input

`fp_scan_out`. Scan data output

TABLE 7-1, TABLE 7-2, and TABLE 7-3 describe write enable, file alignment, and file bypass control functions, respectively.

TABLE 7-1 16 x 64 Register File Write Enable Control Function

fp_we[1:0]		Action
0	0	No change
0	1	Write bits [31:0] of the RAM cell word addressed by `fp_wa`[3:0]
1	0	Write bits [63:32] of the RAM cell word address by `fp_wa`[3:0]
1	1	Write bits [63:0] of the RAM cell word address by `fp_wa`[3:0]

TABLE 7-2 16 x 64 Register File Alignment Control Function

fp_align1:0]		Action
0	0	Undefined
0	1	`fp_dout`[63:0] = `bypass_mux_out`[63:0]
1	0	`fp_dout`[63:0] = {`bypass_mux_out`[31:0], `bypass_mux_out`[31:0]}
1	1	Undefined

TABLE 7-3 16 x 64 Register File Bypass Control Function

byp[5]	byp[4]	byp[3]	byp[2]	byp[1]	byp[0]	fp_dout[63:32]	fp_dout[31:0]
0	0	0	0	0	0	Undefined	Undefined
0	0	1	0	0	1	Register	Register
0	0	1	0	1	0	Register	`fp_din[31:0]`
0	0	1	1	0	0	Register	`wr_data[31:0]`
0	1	0	0	0	1	`fp_din`[63:32]	Register
0	1	0	0	1	0	`fp_din`[63:32]	`fp_din[31:0]`
0	1	0	1	0	0	`fp_din`[63:32]	`wr_data[31:0]`
1	0	0	0	0	1	`wr_data`[63:32]	Register
1	0	0	0	1	0	`wr_data`[63:32]	`fp_din[31:0]`
1	0	0	1	0	0	`wr_data`[63:32]	`wr_data[31:0]`

If more than 1 select is active within `byp[5:3]` or `byp[2:0]`, then the output is undefined.

7.3 AC Switching Characteristics

Note – This design has been implemented by Sun in a 0.35um 3 metal layer process, and is presently shipping at 100Mhz. The megacell timing diagrams have been drawn as a guideline to achieving similar results.

This section presents 16 x 64 register timing. See TABLE 7-4 for timing values; all units are in nanoseconds (ns).

TABLE 7-4 16 x 64 Register File Timing

Symbol	Parameter	Min	Max
t_C	Read / write cycle time	-	14.0
t_{CPW}	ss_clock pulse width	4.0	-
t_{ACC}	Clock to Output (`fp_dout1, 2, 3`)	1.5	8.5
t_{SRA}	Address setup to rising `ss_clock`	-0.5	
t_{SD}	Data setup to rising `ss_clock`	-0.5	
t_{SWE}	Write enable setup to rising `ss_clock`	-0.5	
t_{HA}	Address hold after rising `ss_clock`	2.0	-
t_{HD}	Data hold after rising `ss_clock`	2.0	-
t_{HWE}	Write enable hold after rising `ss_clock`		2.0
t_{WDB}	Write Data to Bypass Output valid	-	2.0
t_{BSD}	Bypass Select to Data Output valid	-	3.0
t_{ASD}	Alignment Select to Data Out valid	-	2.0
t_{SCS}	`ss_can_mode`, `fp_san_in` setup	-0-.5	

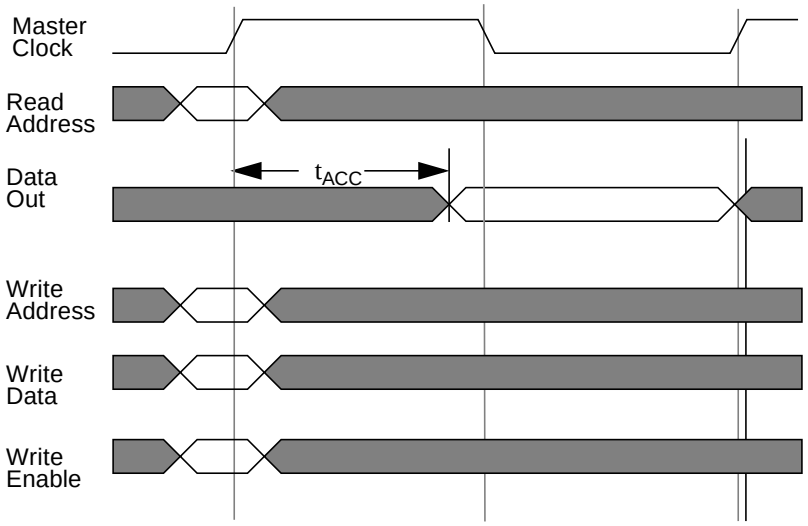

FIGURE 7-3 16 x 64 Register File Timing Diagram

7.3.1 Loading Conditions

The capacitance of each input pin is expected to be no more than 100 fF.

The output of the register file must meet timing with the following loading: `fp_dout1,2,3` have 400 fF per output.

7.3.2 Bypass Case

Bypass can be used during normal operation of the register file, when reading a register location currently being written to. FP control logic guarantees that the bypass enable signals are asserted when needed; no address collision detection is needed within the register file.

7.4 Area and Power Dissipation

The area usage of this block is expected to be less than 2.0 sq. mm.

Power dissipation under normal operation (three reads and one write every 14 ns) is expected to be under 0.4 watts.

7.5 Initialization and Abnormal Conditions

The register file requires no data initialization and its contents may be unknown on powerup and after reset. This register file must not draw excessive current during powerup.

7.6 Testability

This register file will be tested by functional vectors (under control of the IU). All internal registers must be scannable; the order of the scan chain is unimportant.

Index

U

V

W