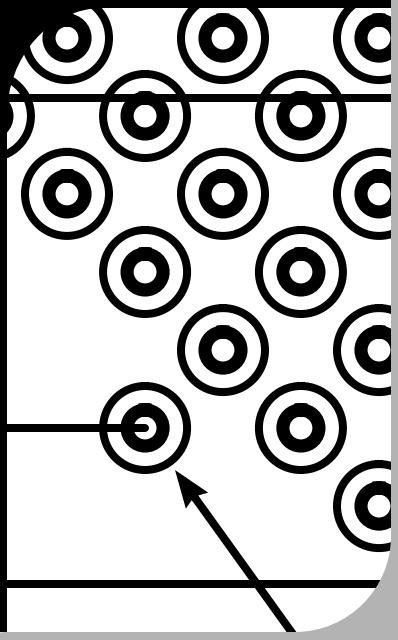
SUN MICROELECTRONICS



DSC

Dual Processor System Controller

User's Manual

February 1997

STP2202ABGA

Sun microsystems

Dual Processor System Controller (DSC) User's Manual

Part No: 802-7511-02

This is a preliminary document. Please send all comments to edgar.you@eng.sun.com.

Revision History

Document Part Number	Release Date	Remarks
802-7511-01	October 1996	Limited Preliminary Release
802-7511-02	February 1997	General Release

Table of Contents

Table of Contents

Overview 1

1.1 Introduction

This document describes the Dual Processor System Controller ASIC, also referred to as “DSC” (STP2202ABGA) used in Ultra 2. Ultra 2 is a low-cost, high-performance two-processor system.

1.2 DSC Summary

DSC’s primary functions are to provide coherence control, memory control, datapaths control, flow control, transaction ordering, and address routing. It regulate the flow of requests and data on the system’s “UPA” interconnect based on dual cache coherent tags for each of the two processors. DSC also controls resets going to all UPA clients.

1.3 Features

1.3.1 UPA Ports

In UPA interconnection architecture cache coherence is maintained with point-to-point packet transactions from a centralized system controller (DSC). The system controller maintains duplicate set of cache tags for each processor and performs duplicate tag lookup and main memory initiation for each coherent transaction.

DSC implements four UPA ports on two UPA address busses, as shown in Figure 1-2. There are 2 CPU ports and the U2S port on Address Bus 0 and the FFB (Fast Frame Buffer) port on Address Bus 1. Ultra 2 can also be configured with only one CPU and the DSC will work properly.

Address bus 1 is a UPA64S subset implementation that supports slave only graphics. The P_REPLY and S_REPLY for the graphics slave are subsets of the full P_REPLY and S_REPLY, the address bus is only 29 bits wide instead of 35 bits, and does not have a parity bit. The bus is unidirectional from the DSC to the port. UPA64S is described in further detail in the “UPA Interconnect Architecture” specification.

1.3.2 Data Path Control

UPA provides data path interconnections between UPA ports and memory. Three data busses, processor data bus, memory data bus and UPA 64-bit data bus, are implemented in the Ultra 2 system. They are interconnected by the Buffered Crossbar chip (XB1).

DSC schedules the XB1 chips, and controls the flow of data on the UPA data busses. Memory data bus, connecting the memory SIMMs and XB1 chip, is 576 bits wide with 512 bits of data and 64 bits of ECC. The CBT switches which provide bank level of memory muxing are also under DSCDSC control. (Figure 1-2)

1.3.3 Memory Interface

DSC contains the memory controller. It can control up to sixteen 60 ns SPARC 20 type DRAM SIMMs. The capacity of the DRAM SIMMs planned to date are: 16MB, 64MB and 128MB. Thus the total memory capacity of an Ultra 2 system will be 2GB.

The memory system is based on a banking mechanism that allows simultaneous access to an entire 64 byte block. This improves the memory performance by reducing average latency for memory reads.

1.3.4 Resets

DSC controls and generates a number of resets for the system. It accepts reset inputs from the RISC ASIC and forwards them to other parts of the system.

1.3.5 EBus Interface

Since DSC has no data bus, an 8 bit asynchronous EBus interface is provided to allow reading and writing of DSC's internal registers. EBus is controlled by Slavio. It is through the EBus that the memory controller is initialized.

1.3.6 JTAG Interface

DSC provides a JTAG interface for full chip scan, which is used only for ATPG and in system interconnect testing. DSC's boundary is shadowed to allow for board level test.

1.4 Package and Die

The DSC uses a 372 ball BGA (ball grid array) package. The die has 320 pads. There are 244 signal pads with 76 power and ground pads. The is the same package and power pin layout as the U2S ASIC.

1.5 Frequency of Operation

DSC will operate at a maximum frequency of 83.3 MHz (12.0 nanosecond cycle time). It is a completely synchronous edge-triggered register-based design which uses only the rising edge of the clock to update the flip-flops during normal operation. Clocking in the JTAG section of the design permits clocking on the negative edge of the JTAG TCK.

The chip also contains a phase-locked loop (PLL) to remove the skew introduced by the internal clock distribution network.

1.6 Gate Count

The current gate count for DSC is approximately 160K gates. DSC contains no custom RAM cells and uses custom 372 BGA package.

Table 1-1

Block name	Gate approximation
CC	**20K**
PIF	**69K**
MC	**45K**
DPS	**16K**
EBUS	**7K**
JTAG	**3K**
Total	**~160K**

1.7 Block Diagrams

A high level block diagram of DSC is shown in the figure below, with abbreviated signal names.

Figure 1-1 DSC Block Diagram

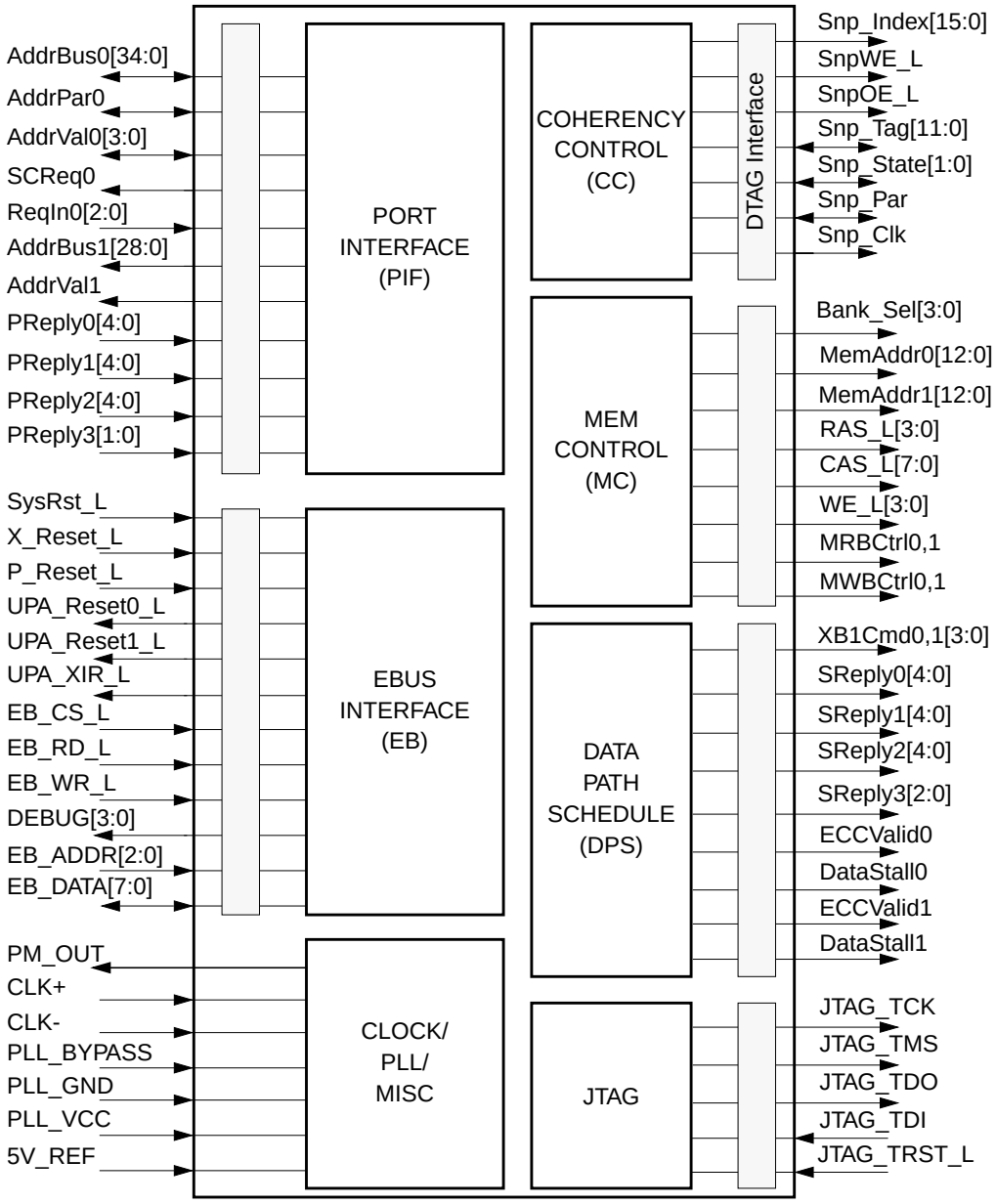
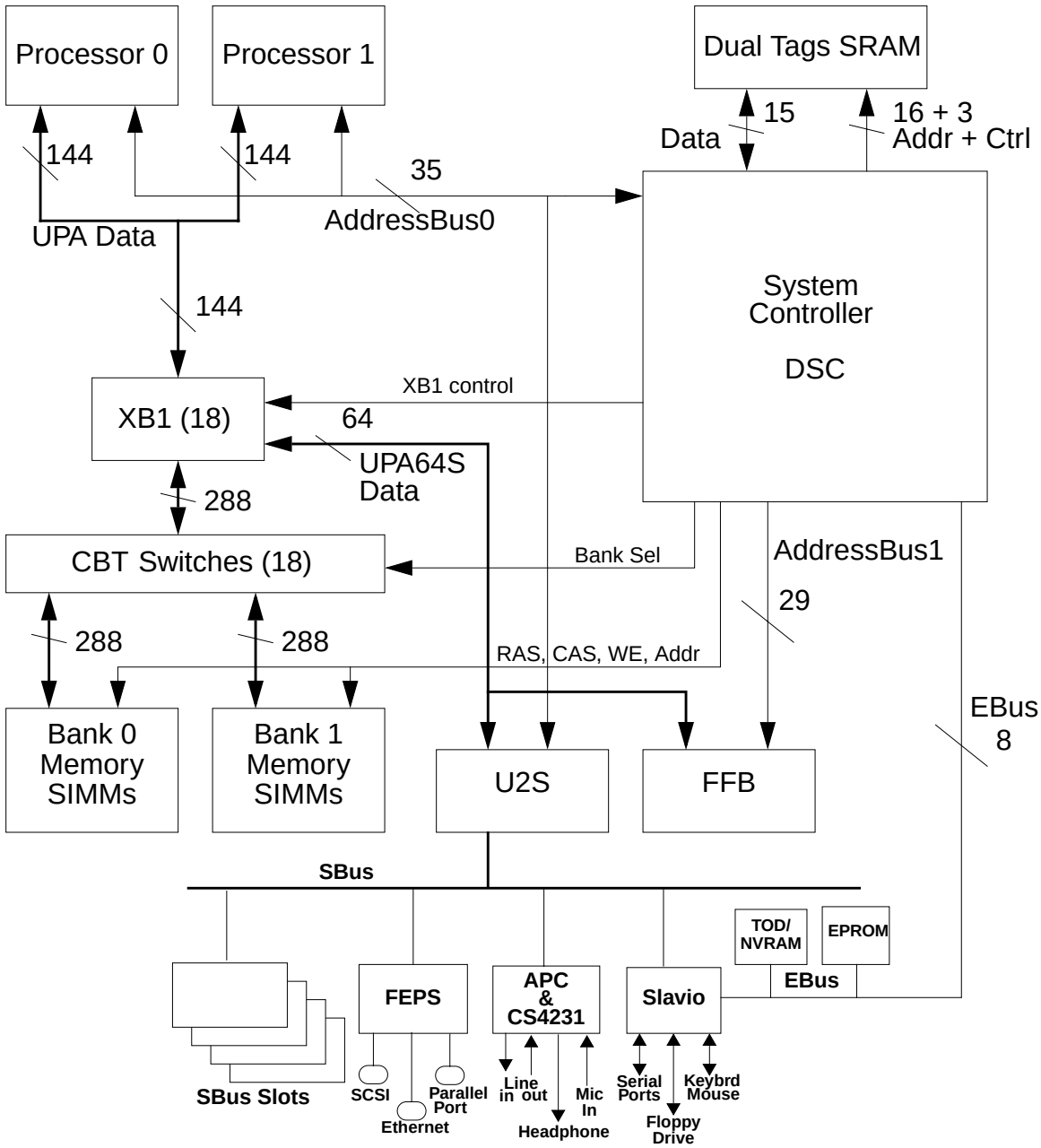


Figure 1-2 System Controller in Ultra 2



Pin List and Description 2

This chapter lists all of DSC's signal pins. Please refer to the ATT HL400C Data Book for a description of the buffer type.

2.1 UPA Interface Signals

Table 2-1 UPA Interface Signals

Signal Name	Pin Count	I/O	Buffer Type	Description
UPA_ADDRBUS0<34:0>	35	I/O	holding amp	address bus 0 (2 processors/ U2S)
UPA_ADDRPAR0	1	I/O	holding amp	parity for address bus 0
UPA_ADDRVAL0[2:0]	3	I/O	holding amp	[0] - processor0, [1] -processor1, [2] - U2S
UPA_SCREQ0	1	O		SC request for address bus 0
UPA_REQIN0<2:0>	3	I		client address bus 0 arbitration requests: [0] - processor0, [1] - processor1, [2] U2S
UPA_ADDRBUS1<28:0>	29	O		address bus 1 - UPA64S (slave only)

Signal Name	Pin Count	I/O	Buffer Type	Description
UPA_ADDRVAL1	1	O		address valid for UPA64S
UPA_SREPLY0<4:0>	5	O		S_Reply for processor 0
UPA_SREPLY1<4:0>	5	O		S_Reply for processor 1
UPA_SREPLY2<4:0>	5	O		S_Reply for U2S
UPA_SREPLY3<2:0>	3	O		S_Reply for UPA64S
UPA_PREPLY0<4:0>	5	I		P_Reply from processor 0
UPA_PREPLY1<4:0>	5	I		P_Reply from processor 1
UPA_PREPLY2<4:0>	5	I		P_Reply from U2S
UPA_PREPLY3<1:0>	2	I		P_Reply from UPA64S
UPA_RESET0_L	1	O		reset for processors, tied to U2S's UPA_ArbReset_L
UPA_RESET1_L	1	O		reset for U2S/FFB
UPA_XIR_L	1	O		XIR reset for processors only
UPA_ECCVALID0	1	O		ECC valid for processors
UPA_ECCVALID1	1	O		ECC valid for U2S
UPA_DATASTALL0	1	O		stall data to processors
UPA_DATASTALL1	1	O		stall data to U2S
Total UPA Port Interface	**115**			

2.2 Dual Tag Interface

Table 2-2 Dual Tag Signals

Signal Name	Pin Count	I/O	Buffer Type	Description
SNP_INDEX[15:0]	16	O		Snoop Ram address
SNPWE_L	1	O		Snoop Ram Write Enable
SNPOE_L	1	O		Snoop Ram Output Enable
SNPTAG[11:0]	12	I/0		Snoop Tags
SNPSTATE[1:0]	2	I/0		Snoop Tag States

Signal Name	Pin Count	I/O	Buffer Type	Description
SNPPAR	1	I/0		Snoop Ram Parity
SNPCLK	1	O		Snoop Clock
Total DTAG Interface	**34**			

2.3 Memory Interface Signals

Table 2-3 Memory Interface Signals

Signal Name	Pin Count	I/O	Buffer Type	Description
MEMADDR0<12:0>	13	O		row/column address
MEMADDR1<12:0>	13	O		row/column address
RAS_L<3:0>	4	O		RAS
CAS_L<7:0>	8	O		CAS
WE_L[3:0]	4	O		WE
BANK_SEL[3:0]	4	O		BANK_SEL[3:2] is duplicate of BANK_SEL[1:0]
Total Memory Interface	**46**			

2.4 XB1 Interface Signals

Table 2-4 XB1 Interface Signals

Signal Name	Pin Count	I/O	Buffer Type	Description
BMXCMD0[3:0]	4	O		command to XB1
MRBCTRL0	1	O		fill the XB1 read buffer
MWBCTRL0	1	O		drain the XB1 write buffer
BMXCMD1[3:0]	4	O		duplicate of BMXCMD0

Signal Name	Pin Count	I/O	Buffer Type	Description
MRBCTRL1	1	O		duplicate of MRBCTRL0
MWBCTRL1	1	O		duplicate of MWBCTRL0
Total XB1 Interface	**12**			

2.5 EBus Interface

Table 2-5 EBus Interface

Signal Name	Pin Count	I/O	Buffer Type	Description
EBUS_DATA<7:0>	8	I/O	5V tolerant	Data in and out pins - TTL levels
EBUS_CS_L	1	I	5V tolerant	chip select for DSC on EBus
EBUS_ADDR	3	I	5V tolerant	EBus address
EBUS_WR_L	1	I	5V tolerant	indicates write on EBus
EBUS_RD_L	1	I	5V tolerant	indicates read on EBus
Total EBus Interface	**14**			

2.6 Miscellaneous Signals

Table 2-6 Miscellaneous Signals

Signal Name	Pin Count	I/O	Note	Description
SYS_RESET_L	1	I	5V tolerant	power on reset
P_RESET_L	1	I	5V tolerant	POR Button Reset
X_RESET_L	1	I	5V tolerant	XIR Button Reset

Signal Name	Pin Count	I/O	Note	Description
CLK	1	I	PECL	System Clock (differential)
CLK_L	1	I	PECL	System Clock (differential)
PLL_BYPASS	1	I		bypass internal PLL
JTAG_TDI	1	I	5V tolerant	test data input
JTAG_TDO	1	O		test data output
JTAG_TCK	1	I	5V tolerant	scan clock
JTAG_TMS	1	I	5V tolerant	test mode select
JTAG_TRST_L	1	I	5V tolerant	reset TAP controller
PLL_VCC	1	I		VCC for PLL
PLL_GND	1	I		ground for PLL
5V_REF	1	I		5V reference for 5V tolerant inputs
DEBUG<3:0>	4	O		External view of internal signals
PM_OUT	1	O		Process Monitor output
Total Misc. Signals	**19**			

2.7 Power and Ground

Table 2-7 Power and Ground

Signal Name	Pin Count	I/O	Note	Description
VDD	38			+3.3V
VSS	38			ground
Total Power and Ground	**76**			

2.8 Total Pin Count

Table 2-8 Pin Count

Signal Group	Total Signals
UPA Port Interfaces	115
Dual Tag Interface	34
Memory Interface	46
BMX Interface	12
EBus Interface	14
Miscellaneous	19
Total SC Signals	**240**
Power & Gnd	**76**
Spares	**4**
SC Total	**320**

Functional Description 3

3.1 Introduction

This chapter gives a brief overview of the internal structure of DSC. The purpose of this chapter is to provide the reader with a brief overview of how functionality is divided up inside DSC

DSC is divided into five major blocks:

1. Port Interface (PIF)
2. Datapath Scheduler (DPS)
3. Memory Controller (MC)
4. Coherency Controller(CC)
5. EBus Interface (EB)

These functions are described in further detail in the following sections.

3.2 Port Interface (PIF)

The PIF supports two UPA busses. It is responsible for receiving packets, decoding their destinations, and forwarding packets to the proper destinations. Cached transactions are typically forwarded to the Memory Controller. Non-cached transactions are typically forwarded to the proper slave.

UPA address bus 0 has three UPA ports. Each Processor and the U2S can act as either a UPA master or a slave.

UPA address bus 1 implements the UPA64S subset only. It supports only a single graphics slave. This bus is output only, not bidirectional, and SC is always the master of this bus. In addition, the graphics slave will only generate and receive truncated PReply and SReply signals.

PIF controls the arbitration on UPA address bus 0. There are three other masters on this bus, the two processors and U2S. PIF uses a distributed round robin algorithm as described in the "UPA Interconnect Architecture" document.

PIF also receives all PReplys from UPA clients.

PIF contains four sets of the following registers (one for each UPA port):

1. SC_Port_Config registers.
2. SC_Port_Status registers.

These registers are described in further detail in the "Fusion Desktop System Specification" document.

3.3 Datapath Scheduler (DPS)

The DPS is responsible for regulating the flow of data throughout the system. It generates the following:

1. XB1 commands.
2. S_REPLYs for all clients.
3. UPA_DATASTALL signals.
4. UPA_ECCVALID signals.

DPS contains no software visible registers.

3.4 Memory Controller (MC)

MC is responsible for controlling the SIMMs. It performs the following functions.

1. Generates timing for read, write, and refresh.
2. Converts the physical address in the UPA packet into row and column addresses.
3. Maintains refresh timer.
4. Controls loading and unloading of data from the XB1 read and write buffers.

5. Controls the CBT memory switch (FET multiplexer) selects.

PIF forwards memory requests to the MC. MC communicates with the DPS to schedule delivery of data.

The operation of MC and the memory subsystem is described in further detail in Chapter 7, “Memory System”.

3.5 Coherency Controller (CC)

The CC is responsible for maintaining cache coherency at the system level. A copy of each processors tags is kept in “Dual Tag” RAM that is connected to the CC. The CC checks the Dual tags or “snoops” on all of the cacheable transactions. From this snoop it can determine the appropriate response. The operation of the Coherency Controller is further described in Chapter 6, “Coherence Controller.”

3.6 EBus Interface (EB)

EB implements an interface to EBus, an asynchronous 8-bit interface controlled by Slavio. Since DSC contains no data path, all reading and writing of internal registers has to take place via EBus. Since all internal registers are 32 bits wide, EB has to perform packing and unpacking.

EB is the only block which can be used in both USC and DSC with minimal change.

The EB block implements reset logic.

The EB block contains a number of global registers.

1. SC_Control register for controlling resets and logging reset status.
2. SC_ID register, which contains DSC’s JEDEC ID number, implementation and version numbers, and the number of UPA ports that this chip supports.
3. Performance counters: SC_Perf_Ctrl, SC_Perf1, and SC_Perf2. These counters can be configured to count various events for performance analysis.

The global registers are described in further detail in the “Fusion Desktop System Specification” document.

The operation of EB is described in further detail in the “Resets” and “Ebus” chapters.

3.7 Performance Monitors

The DSC has a small block that contains logic to monitor performance. Performance registers can be read after being set up by programming the SC_PerfCtrl register. These registers are defined in the Fusion Desktop System Specification. These registers are accessed through the EBus.

3.8 JTAG Interface

The DSC has a JTAG interface. This interface block is common to the Fusion ASICs and is detailed in the Fusion Desktop System Specification.

3.9 UPA Master ID Assignment

The UPA master IDs are hardwired into DSC.

Table 3-1 Master ID Table

Master ID[4:0]	Client
5'b00000	CPU0
5'b00001	CPU1
5'b11111	SYSIO

Since the UPA64S client (FFB) is a slave only device, it does not have a master ID.

3.10 System Controller Queues

The lengths of some important queues are listed in the tables below.

Table 3-2 SC Queue Lengths and Depths

Queue Name	Queue Depth in Packets	Note
MRQ0/CPU	1	master request queue for CPU class 0
MRQ1/CPU	8	master request queue for CPU class 1
MRQ/SYSIO	2	master request queue for SYSIO

3.11 Code Structure and Hierarchy

3.11.1 Signal Naming Conventions

In an effort to improve the readability of the code, all the internal interface signal names in DSC follow a standard convention.

<source><destination>_<signal name>

Source and destination are a single upper case character.

Table 3-3 Source and Destination Designators

Source/Destination Block	Letter Designator
External Signal	E
Port Interface	P
Datapath Scheduler	D
Memory Controller	M
EBus Interface	B
Coherence Controller (DSC only)	C
Multi-drop destination	X

Programming Model 4

This Chapter provides an overview of the address partitioning and software visible registers and their respective functionality. The physical address associated with each of these registers is listed along with a description of the register. Information contained in this document should not duplicate that found in other documents; however, in cases of conflict, the definitions described below takes precedence.

Warning – Registers which are designated as Write Only may be read, but the data returned is UNDEFINED. No error is reported for such an access. Software should never rely on the value returned. Writes to Read Only registers have no affect. No error is reported for such an access.

4.1 Physical Address Space Allocation

The DSC interfaces to 4 UPA ports and Main Memory. When a UPA Master generates a request with an explicit address, the system controller decodes this address to determine its destination. The mapping between UPA ports and the addresses under which they fall is hard wired into the SC in accordance with the Sun-5 Architecture guidelines. Refer to the figure

below for the format of all IO accesses. For these, the top three bits are 1, the next 5 bits comprise the UPA port number (UPN) of the desired UPA port, and the lower 33 bits form the port address.

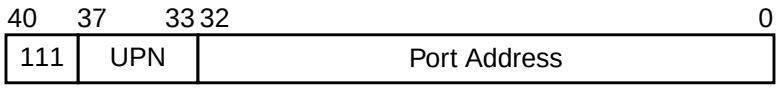


Figure 4-1 System Register Address Formation

Table 4-1 specifies the Physical address to Port ID mapping in the system controller.

Table 4-1 SC Address Map

Address Range in UPA Address<40:0>[1]	Size	UPA Port Addressed	Port ID[2]	Notes
0x0 - 0x0FF.FFFF.FFFF	1 Tera Byte	Main Memory	NA	SC Responds to entire address range.[3]
0x100.0000.0000- 0x1BF.FFFF.FFFF	768 Gbytes	Reserved	NA	
0x1C0.0000.0000- 0x1C1.FFFF.FFFF	8 Gbytes	Processor 0	0	
0x1C2.0000.0000- 0x1C3.FFFF.FFFF	8 Gbytes	Processor 1	1	
0x1C4.0000.0000- 0x1FB.FFFF.FFFF	224 Gbytes	Reserved		
0x1FC.0000.0000- 0x1FD.FFFF.FFFF	8 Gbytes	expansion (slave)	30	UPA Slave (FFB)
0x1FE.0000.0000- 0x1FF.FFFF.FFFF	8 Gbytes	U2S	31	System IO space,

1. Accesses to slave addresses MUST be non-cacheable.
2. PortID is the Master ID (MID) for devices which support mastership.
3. However, on the USC, the max. memory is 1 GB. Addresses beyond 1 GB wrap. The DSC has a max memory of 2 GB and addresses beyond this also wrap.

Accesses to reserved regions will result in a timeout reply to the initiator.

Table 4-2 specifies the Physical address to SBus map in the U2S.

Table 4-2 U2S SBus Address Map

Address Range in UPA Address<40:0>	Size	SBus port addressed	Notes
0x1FF.0000.0000 - 0x1FF.0FF.FFFF	256 MB	slot 0	
0x1FF.1000.0000 - 0x1FF.1FFF.FFFF	256 MB	slot 1	
0x1FF.2000.0000 - 0x1FF.2FFF.FFFF	256 MB	slot 2	neutron & pulsar
0x1FF.3000.0000 - 0x1FF.3FFF.FFFF	256 MB	slot 3	pulsar only
0x1FF.4000.0000 - 0x1FF.CFFF.FFFF	2.256 GB	Reserved	
0x1FF.D000.0000 - 0x1FF.DFFF.FFFF	256 MB	APC (slot 13)	
0x1FF.E000.0000 - 0x1FF.EFFF.FFFF	256 MB	FEPS (slot 14)	electron & pulsar
0x1FF.F000.0000 - 0x1FF.FFFF.FFFF	256 MB	Slavio (slot 15)	

Note – Invalid accesses to valid regions, for example an NC access to memory, will generate a bus error (see the Spitfire PRM discussion of the Asynchronous Fault Status Register). Valid accesses to reserved regions will generate a Time out error. Refer to Chapter 5 for details of packet handling. Transactions marked RTO in that chapter will cause a time out, while transactions marked error will report a bus error.

4.2 Memory Address Assignments

Main memory spans a 2 GB region starting at physical address 0x000.0000.0000. Ultra2 has 16 SIMM sockets which range in size from 16MB per SIMM to 128 MB per SIMM. That makes a total of 2GB memory capacity at maximum. SIMMs must be loaded in 4 pieces, called a group. If the same size memory SIMMs are not installed, software should either disable the group, or configure them to the lower sized SIMM.

Addresses mapped to memory must be cacheable. Transfers between any UPA port and memory has to be done in cache line size of 64 bytes. Non-cacheable accesses to memory are not supported by Sun5 and will be treated as an error. Parameters that affect the address assignments of each SIMM module are SIMM size and which group the SIMM is installed. SIMMs must be installed in matching pairs for proper operation. Any mixture of sizes is permitted among pairs.

Each SIMM in the system has a unique address range based on type and size of the SIMM. Software can identify type and size of the SIMM based on its address range. The address ranges for each SIMM in are listed below.

Table 4-3 Memory Address Map (Pulsar)

SIMM group-Number	SIMM Type	Address Range (PA[30:0])[1]
0	16 MB	0x0000_0000 to 0x03ff_ffff
0	32 MB	0x0000_0000 to 0x07ff_ffff
0	64 MB	0x0000_0000 to 0x0fff_ffff
0	128 MB	0x0000_0000 to 0x1fff_ffff
1	16 MB	0x2000_0000 to 0x23ff_ffff
1	32 MB	0x2000_0000 to 0x27ff_ffff
1	64 MB	0x2000_0000 to 0x2fff_ffff
1	128 MB	0x2000_0000 to 0x3fff_ffff
2	16 MB	0x4000_0000 to 0x43ff_ffff
2	32 MB	0x4000_0000 to 0x47ff_ffff
2	64 MB	0x4000_0000 to 0x4fff_ffff
2	128 MB	0x4000_0000 to 0x5fff_ffff
3	16 MB	0x6000_0000 to 0x63ff_ffff
3	32 MB	0x6000_0000 to 0x67ff_ffff
3	64 MB	0x6000_0000 to 0x6fff_ffff
3	128 MB	0x6000_0000 to 0x7fff_ffff

1. While the SIMM type is 16 MB, SIMMs are always installed in groups of 4, so the address range is four times the SIMM size

4.3 System Controller Registers

The System Controller internal address map is shown below. Note that the addresses specified are only 8 bits wide. This is because the addresses are not system addresses, but SC internal addresses. The SC is not addressed directly by software, rather the internal address map is addressed indirectly.

For all the registers that follow, not all 32 bits are specified. If a field is not specified explicitly, then it is considered RESERVED. Writes to RESERVED fields are ignored, and data read from a reserved field is always 0. Reads from an unimplemented SC internal addresses return UNDEFINED data. Writes to unimplemented SC internal addresses have unknown effects, as the address may wrap to an implemented address.

4.3.1 SC System Addresses

The SC requires 2 system addresses be allocated to it. The first system address is for the SC_Address Register. This need only be a single byte. The second system address is for the SC_Data Register. This must be a 32 bit word address, but it is accessed by software as 4 individual bytes.

The access to SC_Address Register and SC_Data Register is done through SLAVIO generic port. Some address bits in the generic port address space are decoded to select the SC registers. The following table shows the addresses of these two registers.

Table 4-4 Address Assignments of System Controller Address and Data Registers

Physical Address	Register
0x1FF.F130.0000	SC_Address Register
0x1FF.F130.0004	SC_Data Register

SC_Address Register (0x1FF.F130.0000)

The SC Address Register is a byte access register. Table 4-7 gives the values software would write into this register to access other internal registers. The SC_Address Register can be read and written.

Table 4-5 SC_Address Register

Field	Bits	Pupstate[1]	Description	Type
Address	7:0	X	Address Pointer to SC internal register Map	R/W

1. Pupstate = Power-up state

SC Data Register (0x1FF.F130.0004)

Software accesses the SC_Data_Register with 4 consecutive, atomic byte accesses to addresses 0x1FF.F130.0004, 5, 6, and 7. Note that since atomicity of the 4 accesses is not guaranteed by hardware, software must guarantee this by the use of a mutex lock or other such mechanism.

Since the access to this register is really an indirect access to another register in the SC, software must take care to access the bytes in the appropriate order.

Note – Byte ordering within the SC data register is big-endian (e.g., byte 0 corresponds to bits 31:24).

For read and write accesses, the order is byte 0, followed by byte 1, then 2, then 3. Reading byte 0 causes the entire word to transfer to a holding register for subsequent reading and returns byte 0 data; subsequent reads of bytes 1, 2, and 3 return the contents of the holding register. Writes are similar in that writes of bytes 0, 1, and 2 are stored in a temporary write register until byte 3 is written. Then the entire word is transferred to the requested internal register.

Table 4-6 SC_Data Register

Field	Bits	Pupstate	Description	Type
Data	31:0	X	Writing byte 3 initiates the indirect write, reading byte 0 initiates the indirect read.	R/W

SC Indirect Addressing

To access any of the SC Internal registers, software must first write the value of the internal address into the SC_Address register. Subsequent accesses to the SC_Data register actually access the register pointed to by the value in the SC_ADDRESS register. Note that not all addresses are implemented.

Warning – There is no error detection on SC Internal addresses.

Table 4-7 SC Internal Address Space

Internal Address	Register Name	Associated Port
0x00	SC_Control	SC Overall
0x04	SC_ID	SC Overall
0x08	SC_Perf0	SC Overall
0x0C	SC_Perf1	SC Overall
0x10	SC_PerfShadow	SC Overall
0x20	SC_Perf_Ctrl	SC Overall
0x30	SC_Debug_Pin_Ctrl	SC Overall
0x40	P0_Config	Processor 0
0x44	P0_Status	Processor 0
0x48	P1_Config	Processor 1
0x4C	P1_Status	Processor 1
0x50	U2S_Config	U2S
0x54	U2S_Status	U2S
0x58	FFB_Config	FFB
0x5C	FFB_Status	FFB
0x60	MC_Control0 (SCUP)	Memory Controller
0x64	MC_Control1 (SCUP)	Memory Controller
0x70	CC_Diag	Coherence Controller
0x74	CC_SnpVec	Coherence Controller
0x78	CC_Fault	Coherence Controller
0x7C	CC_PROC_Index	Coherence Controller
0x80	MemCtrl00 (DSC)	Memory Controller
0x84	MemCtrl01 (DSC)	Memory Controller
0x88	MemCtrl02 (DSC)	Memory Controller
0x8C	MemCtrl03 (DSC)	Memory Controller
0x90	MemCtrl04 (DSC)	Memory Controller

Table 4-7 SC Internal Address Space

Internal Address	Register Name	Associated Port
0x94	MemCtrl05 (DSC)	Memory Controller
0x98	MemCtrl06 (DSC)	Memory Controller
0x9C	MemCtrl07 (DSC)	Memory Controller
0xA0	MemCtrl08 (DSC)	Memory Controller
0xA4	MemCtrl09 (DSC)	Memory Controller
0xA8	MemCtrl0A (DSC)	Memory Controller
0xAC	MemCtrl0B (DSC)	Memory Controller

4.3.2 SC Internal Register Definitions

The SC Internal control and status registers are divided into the following groups:

- SC Overall Control and Status Registers - These control overall functions such as chip reset, and provide overall error status
- SC Per UPA Port Registers - Contain fields for customizing the port for each device that might plugged into that slot.
- SC Memory Controller Registers - Controls memory addressing functions of the system memory.
- SC Coherence Controller Registers - contains the per-port index register as well as registers to control the diagnostic access of the dtags.

4.3.2.1 SC Register Notation Conventions

The fields of each register in the descriptions that follow are given in tabular form. The left column gives the field name, the next column indicates the bits within the register that field resides. The third column (Pupstate) specifies the power up state that the field initializes to after POR, SOFT_POR, B_POR, or FATAL. The fourth column is a brief description of the field, and the fifth column specifies a code indicating how the field is accessed. The access codes are described below:

Table 4-8 Register Access Type Codes

Code	Description
R	Read
W	Write
W1C	Write 1 to clear
R0	Read As 0
RW	Read / Write

4.3.2.2 SC Register Updating Restrictions

Due to the nature of many of the registers in the SC, special conventions must be followed when updating certain fields within the registers. The conventions below MUST be followed at all times in order to insure proper operation.

- Reads from the SC's internal registers have no side effects and may be done at any time.
- SPRQS, SPDQS, SPIQS, and SQUEN in UPA port configuration registers must all be written simultaneously.
- Queue sizes must never be decreased, they can only be increased. All queue sizes default to minimum at power up.
- One read should never be cleared until after SPRQS, SPDQS, SPIQS, and SQUEN have been initialized to the desired values. Once this bit is cleared, it cannot be set again except by a reset. It is permissible to clear the Oneread bit coincident with manipulating SPRQS, SPDQS, and SPIQS.
- All W1C bits are implemented such that any write to clear which occurs on the same clock cycle as a setting event results in the bit remaining set: the event has precedence over the W1C.

4.3.3 SC Overall Control Registers

These registers have an overall effect on the SC operation. They are the SC_Control Registers, the SC_ID Register, the SC_Perf1, SC_Perf2, and SC_Perf_Ctrl Registers.

4.3.3.1 SC_Control Register (0x00)

The SC_Control Register is used to implement the SC's reset strategy. It provides 3 functions: reset cause detection, software reset capability, and wakeup reset control.

Table 4-9 SC_Control Register

Field	Bits	Pupstate	Description	Type
POR	31	*[1]	Power On Reset - Asserted only after the SC Sees the assertion of Sys_Reset_L	R/W1C
SOFT_POR	30	*	Set if the last reset was due to software setting the Soft_Power On Reset Bit.	R/W
SOFT_XIR	29	*	Set by software to indicate an XIR Reset Condition.	R/W
B_POR	28	*	Set If the last reset was due to the assertion of P_Reset_L on the SC. Can be set due to a scan reset, or through reset generated when writing the frequency margining register.	R/W1C
B_XIR	27	*	Set if the last reset was due to the assertion of an X_Reset_L on the SC. Can alternatively be set by scan.	R/W1C
WAKEUP	26	*	Set if the last reset was due to a wakeup event. Affects only for port 0 on Electron/Neutron and ports 0/1 on Pulsar (processor ports)	R/W1C
FATAL	25	*	Set if a FATAL error was detected by the SC or reported to it. When this bit is set, a system reset will occur (SC CSR bits are not affected).	R/W1C
Reserved	24	0	Reserved	R0
IAP	23	0	Invert Parity on the address busses	R/W
EN_WKUP_POR	22	0	Enable wakeup POR generation. Cleared by POR, SOFT_POR, B_POR, FATAL, and WAKEUP	R/W
Reserved	21:0	0	Reserved	R0

1. The highest priority reset source will have its bit set. Only one of the bits marked with "*" will be set.

Among the reset bits, only one will be set when reset terminates. If multiple resets occur simultaneously, the following order will be chosen for the reporting:

1. POR
2. FATAL
3. B_POR
4. WAKEUP
5. SOFT_POR
6. B_XIR
7. SOFT_XIR

POR - Power On Reset

This bit is set if the last reset of the SC was due to the assertion of SYS_RESET_L Pin by an external source, and will occur whenever the machine power cycles.

SOFT_POR - Soft Power On Reset

Writing a 1 to this bit has the same effect as power-on reset, except different status bit in the SC Control Register is set. Memory refresh is not affected.

Writing a 0 to this bit clears it and has no other affect.

SOFT_XIR - Soft Externally Initiated Reset

Writing a 1 to this bit causes the SC to assert UPA_XIR_L for 1 cycle. This should cause a software trap for any UPA that is a processor.

B_POR - Button Reset

This bit is set as a result of a “button” reset which is caused by an external switch and the assertion of P_RESET_L on the SC. It can also be set by scan and the frequency margining reset. Memory refresh is not affected.

The actions and results of this reset are identical to that of Power-on Reset, except a different status bit is set.

B_XIR - XIR Button Reset

This bit is set as a result of a “button” XIR Reset which is caused by an external switch which asserts the X_RESET_L on the SC, or by scan.

The actions and results of this reset are identical to that of SOFT_XIR, except a different status bit is set.

WAKEUP - Wakeup

This bit is set if a wakeup event occurred from one of the UPA Ports and EN_WKUP_POR is true. A wakeup event is an interrupt to a port that has its “Sleep” Bit set - See SC_Port_Config Register Definitions. Upon a wakeup event, the SC Asserts UPA_RESET_L<0>, to all nodes that can sleep.

Software reads this bit to determine that the last reset it received was due to a wakeup. This allows it to enter the wakeup related code.

FATAL

This bit is set if one of the following fatal errors was detected by the SC:

1. Address Bus Parity Error (Refer to Section 4.3.4.2, “P{0,1}_Status Register.(0x44, 0x4c),” on page 4-38”)
2. Coherence Error (Refer to Section 4.3.6.3, “CC Fault Register (0x78),” on page 4-59”)
3. P_FERR Reported by a UPA (Refer to Section 4.3.4.2, “P{0,1}_Status Register.(0x44, 0x4c),” on page 4-38”)
4. DTAG Parity Error (Refer to Section 4.3.6.3, “CC Fault Register (0x78),” on page 4-59”)
5. A Master Port overflowed its queue (Refer to Section 4.3.4.2, “P{0,1}_Status Register.(0x44, 0x4c),” on page 4-38”)
6. Handshake error between DPS and Memory Controller. (Refer to Section 4.3.5.2, “Mem_Ctrl0 Register,” on page 4-46)

Software should also read the registers listed above to determine the actual source of the error and clear any bits associated with it.

IAP - Invert Address Parity

This bit is used by diagnostic software to invert the address parity generated by the SC. This is useful for checking the hardware and software involved in logging address parity errors. It is cleared after any POR and unchanged after XIR, or wakeup.

EN_WKUP_POR - Enable Wakeup POR

Enables POR generation on receipt of a wakeup event directed to a port with slave sleep set. Software should set this bit when putting the system to sleep. This bit is cleared on power up, including a wakeup power up.

Note – Interrupts to ports which don't have slave sleep set, including interrupts to invalid ports, will not cause a wakeup reset

4.3.3.2 SC_ID Register (0x04)

This register contains Read only ID information for the SC.

Table 4-10 SC_ID Register

Field	Bits	PupState	Description	Type
JEDEC	31:16	0x3340 or 0xACF1	SC's JEDEC ID; 0x3340 is for USC, while 0xACF1 is for DSC.	R
UPANUM	15:12	0x4	The Number of UPA ports supported by this implementation of the SC	R
Reserved	11:8	0	Reserved	R0
IMPLNUM	7:4	0	Implementation Number	R
VERNUM	3:0	00	Version number; starts at 0 and increments for each revision of this ASIC	R

4.3.3.3 SC_Perf0 Register (0x08)

This is a 32-bit read-only register. Value read back contains the counts of the event selected by the SC_PerfCtrl Register. Whenever this register is read, the shadow register is updated with the contents of the SC_Perf1 counter. When the counter reaches its maximum count, it will wrap around to 0x0 and continue to count. Software needs to detect and handle the overflow condition.

Table 4-11 SC_Perf0 Register

Field	Bits	Description	Type
CNT0<31:0>	31:00	Contains value for event counter 0	R

4.3.3.4 SC_Perf1 Register (0x0c)

This is a 32-bit read-only register. Value read back contains the counts of the event selected by the SC_PerfCtrl Register. Whenever this register is read, the shadow register is updated with the contents of the SC_Perf0 counter. When the counter reaches its maximum count, it will wrap around to 0x0 and continues counting. Software needs to detect and handle the overflow condition.

Table 4-12 SC_Perf1 Register

Field	Bits	Description	Type
CNT1<31:0>	31:00	Contains value for event counter 1	R

4.3.3.5 SC_PerfShadow Register (0x10)

This is a 32-bit read-only register. Value read back contains the count of the event shadowed by the read of the last SC_Perf register. The shadow provides a mechanism to “atomically” read the values of both counters.

The shadow register is sampled at the time byte 0 is read for either performance register.

Table 4-13 SC_Perf Shadow Register

Field	Bits	Description	Type
CNT1<31:0>	31:00	Contains value for event counter 0 or 1	R

4.3.3.6 SC_PerfCtrl Register (0x20)

This register controls the events to be monitored by the Performance Counter Registers. The event counter in the Performance Counter Register will be reset when the CLR bit is asserted. Sources for each of the counters is given in the tables below.

Note – There are only two counters with each counter multiplexing one of sixteen possible sources. It is not possible to track more than 2 events simultaneously.

Table 4-14 Performance Monitor Control Register

Field	Bits	Description	Type
Reserved	31:16	Reserved, read as 0	R
CLR1	15	Clears the counter 1	W
Reserved	14:12	Reserved, read as 0	R
SEL1	11:8	Select event source for counter 1. Counter 1 is cleared when CLR1 field is written. Resets to 0xf. This value was chosen to minimize power.	RW
CLR0	7	Clears the counter 0	W
Reserved	6:4	Reserved, read as 0.	R
SEL0	3:0	Select event source for counter 0. Counter 0 is cleared when CLR0 field is written. Resets to 0xf. This value was chosen to minimize power.	RW

The performance registers each measure one out of 16 total events. A list of possible selections for counter 0 is shown in the table below.

Table 4-15 SC Performance Counter 0 Event Sources

SEL	Event Sources
0x0	system clock count
0x1	Number of P-Requests from all sources
0x2	Number of P-Requests from Processor 0
0x3	Number of P-Requests from Processor 1
0x4	Number of P-Requests from U2S

Table 4-15 SC Performance Counter 0 Event Sources (Continued)

SEL	Event Sources
0x5	Number of cycles the UPA 128 bit data is busy
0x6	Number of cycles the UPA 64 bit data bus is busy
0x7	Number of cycles stalled during PIO
0x8	Number of memory requests issued
0x9	Number of cycles Memory Controller is busy[1]
0xA	Number of cycles stalled because of a Pending Transaction Scoreboard hit.
0xB	Number of coherent write miss requests from P0 (RDO)
0xC	Number of coherent write miss requests from P1 (RDO)
0xD	Number of coherent intervention transactions (CPI+INV+CPB+CPD)
0xE	Number of data transactions (RDO+RDD+WRB+WRI) from U2S
0xF	Number of coherent read (CPB+CPD) transactions issued

1. This includes all events which cause the memory controller to be non-idle including memory references and refresh.

Similarly, counter 1 counts one of 16 possible sources. A list of possible selections is shown in the table below. Note that this list contains some duplications with the SC_Perf1 Register, AND some unique items. Common sources retain the same SEL number.

Table 4-16 SC Performance Counter 1 Event Sources

SEL	Event Sources
0x0	system clock count
0x1	Number of P-Requests from all sources
0x2	Number of P-Requests from Processor 0
0x3	Number of P-Requests from Processor 1
0x4	Number of P-Requests from U2S
0x5	Read requests from P0
0x6	Coherent read misses from P0
0x7	Number of PIO accesses (NCxx) from P0
0x8	Number of memory requests issued
0x9	Number of Memory Requests completed

Table 4-16 SC Performance Counter 1 Event Sources (Continued)

SEL	Event Sources
0xA	Read requests from P1
0xB	Coherent read misses from P1
0xC	Number of PIO accesses (NCxx) from P1
0xD	Number of coherent write transactions (CPI+INV) issued
0xE	Number of data transactions (RDO+RDD+WRB+WRI) from U2S
0xF	reserved

4.3.4 SC Port Interface Registers

There are multiple copies of these registers (as specified). They are used to enable/disable certain port features. Each of the registers is listed individually to indicate which bits are hard wired in this implementation. Those bits which are hard wired (read only) are shown with the normal field name rather than being listed as reserved.

Implementations must guarantee that SW setting of queue sizes beyond the implemented maximum are benign. The suggested behavior is to default any value greater than maximum to the maximum value. In other words, if a queue size field is 4 bits, an implementation could choose to limit the maximum value to 9. Programming a value larger than this in the field must not cause failure. Implemented values less than the field maximum should be indicated.

4.3.4.1 P{0,1}_Config Register (0x40, 0x48)

The following table shows both the Port 0 (processor 0) and Port 1 (processor 1) configuration register. They are identical.

Table 4-17 P0_Config and P1_Config Registers

Field	Bits	PupState	Description	Type
MD	31	0	Master Disable. If set, the SC will block all requests in its master request queue for this master port. All processor ports power up enabled. Once enabled, any requests which exist in the master request queues will proceed.	RW
S_Sleep	30	0	Slave Sleep. If set the slave is currently asleep and an interrupt packet directed to that slave results in a wakeup sequence by the SC and the interrupting port receives a NACK reply. If clear, interrupt packets are treated normally.	RW
Reserved	29:28	0	Reserved	R0
SPRQS[1]	27:24	1	Slave P_request queue size. SW initializes this field with the size in 2 Cycle Packets of the corresponding slave request queue.	RW
Reserved	23:18	0	For ports implementing this feature, would be the slave port data queue size. Not programmable in this implementation	R
SPIQS	17:16	0	Slave Port Interrupt Queue Size. SW programs this field with the length in Address packets of the corresponding interrupt queue.	RW

Table 4-17 P0_Config and P1_Config Registers (Continued)

Field	Bits	PupState	Description	Type
SQUEN	15	0 (write-only)	Writes of SPRQS, and SPIQS are qualified with this bit. If set, the values in these fields are updated. If not, they are unchanged.	W
Oneread	14	1	SW Sets to 0 if the number of outstanding reads allowed to this port is greater than 1. When set to 1, the SC will accept P_RASB, P_RAB, or P_RAS replies. When cleared, only P_RAB or P_RAS are valid.	RW
Reserved	13:0	0	Reads as 0. Not writable.	R

1. An implementation may choose to set a maximum value which is less than what can be programmed in this register. In that case, a larger value must still result in correct operation, but not the desired number of requests being forwarded.

4.3.4.2 P{0,1}_Status Register.(0x44, 0x4c)

Table 4-18 P0_Status and P1_Status Registers

Field	Bits	PupState	Description	Type
FATAL[1]	31	0	This bit is set if this port sent a P_FERR error reply OR if a port returns a P_RASB reply when oneread is FALSE. This condition will cause a chip reset.	RW1C
IADDR	30	0	This bit is set if the SC detects that this port attempted to send a request (read OR write) to an illegal address or a nonexistent port. Accesses forwarded to valid ports will not set this bit regardless of the reply. Writes to processor ports also set IADDR. This is an advisory bit.	RW1C
IPORT	29	0	This bit is set if this port attempted to send an interrupt packet to an illegal destination port, or that port's SPIQS field is set to 0. This is an advisory bit and the interrupt packet is dropped (normal ack).	RW1C
IPRTY	28	0	This bit is set if a packet sent from this port resulted in the SC detecting an address parity error. It has priority over any other fatal condition (e.g., if IPRTY, you may also see IADDR; SW must sort out this case). This condition will cause a chip reset.	RW1C
MC0OF	27	0	Master Class 0 over flow. Set if this master tried to send a packet when no space was available in the queue. This condition causes a chip reset.	RW1C
MC1OF	26	0	Master Class 1 Overflow. Set if this master tried to send a packet when no space was available in the queue. This condition causes a chip reset.	RW1C
MC0Q	25:23	See Table 4-23	These bits indicate to system sw the number of requests packets that can be issued to master class 0 before it overflows	R

Table 4-18 P0_Status and P1_Status Registers (Continued)

Field	Bits	PupState	Description	Type
MC1Q	22:20	See Table 4-23	These bits indicate to system sw the number of request packets that can be issued to master class 1 before its queue overflows	R
MC1Q<0> in DSC	19	0	MC1Q<0> in pulsar only. In electron and neutron, read only register which returns 0.	R
Reserved	18:0	0	Read only as 0.	R

1. Refer to the FATAL bit in Section entitled “SC_Control Register (0x00)””

4.3.4.3 U2S_Config Register (0x50)

Table 4-19 U2S_Config Register

Field	Bits	PupState	Description	Type
MD	31	1	Master Disable. If set, the SC will block all requests in its master request queue for this master port. Once enabled, any requests which exist in the master request queues will proceed. Must be set to 0 for DVMA or interrupts.	RW
Reserved	30:28	0	Reserved; would be Slave Sleep for a port implementing this bit.	R
SPRQS[1]	27:24	1	Slave P_request queue size. SW initializes this field with the size in 2 Cycle Packets of the corresponding slave request queue.	RW
SPDQS	23:18	4	Slave Port Data Queue Size. SW initializes this field with the length in Address packets of the corresponding slave data queue. **The UPA specification dictates that this field be present, but Hardware ignores any nonzero value. If SPDQS is nonzero, then Hardware assumes its value to be 4 times SPRQS.**	RW
Reserved	17:16	0	Reserved. Would be Slave Port Interrupt Queue Size for devices implementing this feature.	R
SQUEN	15	0 (write only)	SW Sets this bit when updating SPRQS, SPDQS and SPIQS, which must be done all at once.	W
Oneread	14	1	U2S may be set as a oneread, or multi-read device. Value should be programmed after a probe of the UPA Port ID register for the U2S.	RW
Reserved	13:0	0	Reserved	R

1. An implementation may choose to set a maximum value which is less than what can be programmed in this register. In that case, a larger value must still result in correct operation, but not the desired number of requests being forwarded.

4.3.4.4 U2S_Status Register. (0x54)

Table 4-20 U2S_Status Register

Field	Bits	PupState	Description	Type
FATAL[1]	31	0	This bit is set if this port sent a P_FERR error reply or a P_RASB reply when oneread is false. This condition will cause a chip reset.	RW1C
IADDR	30	0	This bit is set if this port attempted to send a request to an illegal address or a non-existent port. This is an advisory bit.	RW1C
IPORT	29	0	This bit is set if this port attempted to send an interrupt packet to an illegal destination port. This is an advisory bit.	RW1C
IPRTY	28	0	This bit is set if a packet sent from this port resulted in the SC detecting an address parity error. This condition can cause a chip reset.	RW1C
MC0OF	27	0	Master Class 0 over flow. Set if this master tried to send a packet when no space was available in the queue. This condition will cause a chip reset.	RW1C
Reserved	26	0	Reserved. Would be Master Class 1 Overflow for ports implementing this feature.	R0
MC0Q	25:23	See Table 4-23	These bits indicate to system sw the number of requests packets that can be issued to master class 0 before it overflows	R
Reserved	22:20	0	Reserved. Would indicate Master Class 1 request packets for ports implementing this feature.	R
Reserved	19:0	0	Reserved.	R0

1. Refer to the FATAL bit in Section entitled “SC_Control Register (0x00)

4.3.4.5 FFB_Config Register (0x58)

Table 4-21 FFB_Config Register

Field	Bits	PupState	Description	Type
Reserved	31	1	Reserved. Would be Master Disable in ports implementing this feature.	R
Reserved	30:28	0	Reserved.	R
SPRQS[1]	27:24	1	Slave P_request queue size. SW initializes this field with the size in 2 Cycle Packets of the corresponding slave request queue.	RW
SPDQS	23:18	4	Slave Port Data Queue Size. SW initialized this field with the length in Address packets of the corresponding slave data queue. **The UPA specification dictates that this field be present, but Hardware ignores any nonzero value. If SPDQS is nonzero, then Hardware assumes its value to be 4 times SPRQS.**	RW
Reserved	17:16	0	Reserved. Would be Slave Port Interrupt Queue Size in ports implementing this feature.	R
SQUEN	15	0 (write only)	SW Sets this bit when updating SPRQS, SPDQS and SPIQS, which must be done all at once.	W
Oneread	14	1	Always oneread. UPA slave interface will not support multiple outstanding reads.	R
Reserved	13:0	0	Reserved	R

1. An implementation may choose to set a maximum value which is less than what can be programmed in this register. In that case, a larger value must still result in correct operation, but not the desired number of requests being forwarded.

4.3.4.6 FFB_Status Register. (0x5c)

Table 4-22 FFB_Status Register

Field	Bits	PupState	Description	Type
Reserved	31:0	0	Reserved.	R0

This register is present for architectural consistency. No mastership is supported by the FFB, and all fields are reserved and will read back as 0.

Table 4-23 below gives the Port number and the number of 2 cycle requests for that master's input queues, as specified by the MC1Q and MC0Q.

Table 4-23 SC Port_Status Register MC0Q and MC1Q Init Table.

Port/ Class	Type of Master	Queue Size
Port 0, Class 0	Processor 0	1
Port 0, Class 1	Processor 0	8
Port 1, Class 0	Processor 1	1
Port 1, Class 1	Processor 1	8
Port 2, Class 0	U2S[1]	2

1. Only 1 class is allow for the U2S

4.3.5 Memory Controller Registers

The registers shown below control the functionality of the SC Memory Controller. The USC and DSC memory controllers share only one register, Mem_Control0. The USC implements one other register, Mem_Control1 while the DSC implements 14 total registers including Mem_Control0. The difference in number of registers reflects the different design goals: USC is designed to be the most efficient, high performance, single bank memory controller possible, while the DSC implements the highest performance memory controller possible. To provide complete flexibility, the DSC uses programmable wave form generators which result in a large number of CSRs.

Sections which follow contain all of the registers which are specific to the memory controller used in the DSC. The following table shows the summary of all DSC memory controller registers.

Table 4-24 Memory Controller Registers

Address	Name
0x80	Mem_Ctrl 00 Register
0x84	RAS_Control Register
0x88	CAS_RD_Control Register
0x8C	Bank Sel_Control Register
0x90	BMX_Buffer_Control Register
0x94	CAS_WR_Control Register
0x98	Phase_Level_Control Register
0x9C	SIMM_Busy_Rd_Control Register
0xA0	Count_Control Register
0xA4	Refresh_Control Register
0xA8	Row_Control Register
0xAC	Reserved
0xB0	SIMM_Busy_Wr_Control Register
0xB4	SIMM_Busy_Refr_Control Register

4.3.5.1 Generic Waveform Generator

The requirements for the DSC memory controller necessitates an extremely programmable memory controller.

To achieve the above goal, a generic waveform generator has been created which is the basis for generating the external signals RAS, CAS, WE, MRBCtrl, MWBCtrl, and BankSel, and the internal signal RowGen which is used to select the address output to the DRAMs themselves.

Originally, the MC was designed to operate in two different modes. A Single Bank Mode was specified and is no longer supported, however, there are a few remnants from this mode left behind. In single bank mode, the MC had to generate two low pulses for CAS. The Wavegen Generator was therefore broken up into five phases. For each phase, the number of clockcycles for that phase (from 1 through 8 clocks) and the phase level (high or low) are programmable. The initial value is also programmable. These numbers are taken from the CSR's in the MC depending on whether the request is a read, write, or refresh. This allows us to generate almost any waveform from 5 cycles to 40 cycles long.

A Phase Control word is a 16 bit quantity in the following form:

Table 4-25 Phase Control Word

Bits	Description
15:13	Number of cycles in Phase 5. (000 = 8)
12:10	Number of cycles in Phase 4. (000 = 8)
9:7	Number of cycles in Phase 3. (000 = 8)
6:4	Number of cycles in Phase 2. (000 = 8)
3:1	Number of cycles in Phase 1. (000 = 8)
0	Initial value.

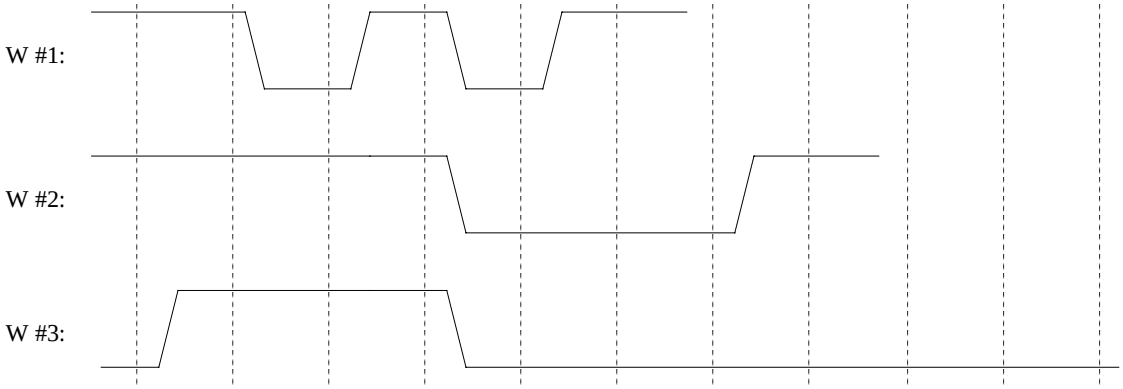
The Phase Level word is a 5 bit quantity in the following form

Table 4-26 Phase Level Word

Bit	Description
4	Level in Phase 5
3	Level in Phase 4
2	Level in Phase 3
1	Level in Phase 2
0	Level in Phase 1

Note – Initial value and level in phase 1 are both needed. If they are the same, then there is no transition in the signal during the first phase. Otherwise, there is. See waveform 3 below for an example of different values.

Now, if we wish to program the following waveforms:



We can use the following values:

Table 4-27 Waveform example

Waveform	Phase Control Word	Phase Level Word
W#1	001 001 001 001 001 1 = 0x2493	10101 = 0x15
W#2	001 010 001 010 001 1 = 0x28A3	10011 = 0x13
W#3	010 010 011 010 001 0 = 0x49A2	00011 = 0x03

4.3.5.2 Mem_Ctrl0 Register

The Mem_Ctrl0 Register contains a number of bits which program the Refresh Controller in the Memory Controller and several other miscellaneous bits.

Table 4-28 Mem_Ctrl00 Register

Field	Bits	PupState	Description	Type
RefEnable	31	0	Refresh enable	RW
Reserved	30:29	0x0	Reserved - Read as 0	R0
FSMError	28	0	MC FSM Error	RW1C
PingPongError	27	0	Ping Pong Buffer Error	RW1C

Field	Bits	PupState	Description	Type
DPSError	26	0	Data path scheduler Error	RW1C
MCError	25	0	Memory Controller Error	RW1C
MissedRefrError	24	0	Missed Refresh Error	RW1C
RefrPhiMC	23	1	Ras Phi 0 Magic Cookie for refresh cycles	RW
RasWrPhiMC	22	1	Ras Phi 0 Magic Cookie for writes	RW
Reserved	21	0	reserved- read as 0	R0
StretchWr0<4:0>	20:16	0x01	Stretch count value for first write	RW
Reserved	15:12	0x0	Reserved - Read as 0	R0
SIMMPresent<3:0>	11:8	0xF	Determines which SIMMs to refresh	RW
RefInterval<7:0>	7:0	0x14	Interval between refreshes. Each encoding is 8 system clocks	RW

RefEnable

Main memory is composed of dynamic RAMs, which require periodic “refreshing” in order to maintain the contents of the memory cells. The RefEnable bit is used to enable refresh of main memory.

```
0 = disable refresh
1 = enable refresh
```

Disabling refresh causes the RefInterval timer to stop counting.

POR is the only condition to clear refresh_enable

SOFT_POR, B_POR, FATAL, WAKEUP, B_XIR, and SOFT_XIR leave refresh_enable unchanged.

Note – This bit may be written to at any time. Any refresh operation in progress at the time of clearing will be aborted with a possible loss of data.

FSMError

This bit signifies that the Main Finite State Machine in the MC has somehow reached an illegal state. Results in a system reset.

PingPongError

This bit signifies that a serious error has occurred in the MC's Ping-Pong buffer management. Results in a system reset.

DPSError

This bit signifies that the DPS has somehow made an error in it's handshake with the MC regarding the Read or Write buffers in the BMX (i.e. - either it has acknowledged before the BMX Read buffer was written or it acknowledged twice for one Read or it has written a second time before the Write buffer has been emptied.) Results in a system reset.

MCError

This bit signifies that the MC has somehow made an error in it's handshake with the DPS regarding the Read or Write buffers in the BMX (i.e. - either it has overwritten the BMX Read buffer or has prematurely read the BMX Write buffer or the BMX buffer FSMs have reached an illegal state.) Results in a system reset.

MissedRefrError

This bit signifies that a refresh request has not been honored and that memory could be corrupted. This could occur for example if a coherent write is issued and for some reason, the handshake that the MC is waiting for signifying that the BMX Write Buffer has been filled, never occurs. The MC will be stuck waiting and thus miss its refresh cycles. A separate watchdog triggers this error. Results in a system reset.

PingPongError, DPSError, MCError, and MissedRefrError are all registers which once set, remain set until either a system reset or they are individually reset by writing a one into them. In any case, they are here mainly for debug and should never be set during normal operation. Any one of these will have the MC request that the E-Bus Interface signal a fatal error and reset the system.

RefrPhiMC

This bit will take the place of the RAS_Phi<0> bit (see Section 4.3.5.8) in the case of a Refresh operation. This was necessary so that on reads we have the ability to drop RAS immediately. Not having this bit would have precluded our ability to do CAS before RAS Refreshes.

RasWrPhiMC

This bit will take the place of the RAS_Phi<0> bit (see Section 4.3.5.8) in the case of a Write operation. This was necessary so that on reads we have the ability to drop RAS immediately, whereas, on writes, we may wish to wait until a more appropriate time.

StretchWR0<4:0>

This register is programmed with the value of the first stretch point in a coherent write cycle. See section for more details.

SIMMPresent<3:0>

This field is used to indicate the presence/absence of SIMMS so that the SC does not spend more time than is necessary refreshing unpopulated SIMMs. A zero in this field indicates the corresponding SIMM is not present, a 1 indicates presence. These bits must be set by software after probing.

The SIMM to bit correspondence is given in Table 4-29.

Table 4-29 SIMMPresent Encoding

SIMMPresent<i>	SIMM Slot
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7

Note – Refresh must be disabled first by clearing the RefEnable bit before changing this field, or the RefInterval. Refresh may be enabled again simultaneously with writing SIMMPresent and RefInterval. Failure to follow this rule may result in unpredictable behavior.

RefInterval

RefInterval specifies the interval time between refreshes, in quanta of 8 system cycles. SW should program RefInterval according to the formula and table in Section 7.6.2, “Mem_Control0 Register,” on page 7-118, which also offers a detailed description of the register.

4.3.5.3 RAS_Control Register

The RAS_Control Register is also referred to as Memory Control Register 01. It contains the Phase Control words for the RAS Waveform generators. SW should program RAS_Control according to the tables in Section 7.6.2.

Table 4-30 RAS_Control Register

Field	Bits	PupState	Description	Type
RAS_WR<15:0>	31:16	0x25c7	Phase control word for RAS on Writes	RW
RAS_RD<15:0>	15:0	0x4ca5	Phase control word for RAS on Reads	RW

4.3.5.4 CAS_RD_Control Register

The CAS_RD_Control Register is also referred to as Memory Control Register 02. It contains the Phase Control words for the CAS Waveform generators for read operations. SW should program CAS_RD_Control according to the tables in Section 7.6.2.

Table 4-31 CAS_RD_Control Register

Field	Bits	PupState	Description	Type
Reserved	31:16	0x0000	reserved- read as 0	R0
CAS_RD<15:0>	15:0	0x2d95	Phase control word for CAS on Reads	RW

4.3.5.5 BankSel_Control Register

The BankSel_Control Register is also referred to as Memory Control Register 03. It contains the Phase Control words for the BankSel Waveform generators for read and write operations. SW should program BankSel_Control according to the tables in Section 7.6.2.

Table 4-32 BankSel_Control Register

Field	Bits	PupState	Description	Type
BS_WR<15:0>	31:16	0x2926	Phase control word for BankSel on Writes	RW
BS_RD<15:0>	15:0	0x249a	Phase control word for BankSel on Reads	RW

4.3.5.6 BMX_Buffer_Control Register

The BMX_Buffer_Control Register is also referred to as Memory Control Register 04. It contains the Phase Control words for the MRBCtrl and MWBCtrl Waveform generators. SW should program BMX_Buffer_Control according to the tables in Section 7.6.3.

Table 4-33 BMX_Buffer_Control Register

Field	Bits	PupState	Description	Type
MWB<15:0>	31:16	0x2896	Phase control word for MWBCtrl on Writes	RW
MRB<15:0>	15:0	0x249a	Phase control word for MRBCtrl on Reads	RW

4.3.5.7 CAS_WR_Control Register

The CAS_WR_Control Register is also referred to as Memory Control Register 05. It contains the Phase Control words for the CAS Waveform generators for write operations for both bank zero and bank one. SW should program CAS_WR_Control according to the tables in Section 7.6.4.

Table 4-34 CAS_WR_Control Register

Field	Bits	PupState	Description	Type
CAS_WR_1<15:0>	31:16	0x24bb	Phase control word for CAS on Writes to bank one.	RW
CAS_WR_0<15:0>	15:0	0x24b7	Phase control word for CAS on Writes to bank zero.	RW

4.3.5.8 Phase_Level_Control Register

The Phase_Level_Control Register is also referred to as Memory Control Register 06. It contains the Phase Level words for all the Waveform generators except the internal signal Waveform generator, Row. Row phi levels are located in Memory Control Register 08 detailed in Section . SW should program Phase_Level_Control according to the tables in Section 7.6.5.

Table 4-35 Phase_Level_Control Register

Field	Bits	PupState	Description	Type
Reserved	31:30	00	reserved -read as zero.	R0
MWB_Phi<4:0>	29:25	01010	Phase level word for MWBCtrl	RW
MRB_Phi<4:0>	24:20	01010	Phase level word for MRBCtrl	RW
BS_Phi<4:0>	19:15	11000	Phase level word for BankSel	RW
Reserved	14:10	00000	reserved -read as zero.	R0
CAS_Phi<4:0>	9:5	10011	Phase level word for CAS	RW
RAS_Phi<4:0>	4:0	11000	Phase level word for RAS	RW

4.3.5.9 SIMM_Busy_Rd_Control Register

The SIMM_Busy_Rd_Control Register is also referred to as Memory Control Register 07. The DSC MC uses countdown timers on a given SIMM to determine how soon after an operation, it can issue a new operation to the same SIMM. Same_Busy_X_Rd contain the values those countdown timers should use for a given operation to the same SIMMs given that an operation, X, is requested next and the current operation is a Read. Diff_Busy_X_Rd are analogous timer values when addressing a different SIMM. SW should program SIMM_Busy_Control according to the tables in Section 7.6.6.Count_Control Register

Table 4-36 SIMM_Busy_Rd_Control Register

Field	Bits	PupState	Description	Type
Reserved	31:30	00	reserved -read as zero.	R0
Diff_Busy_Refr_Rd	29:25	00001	Timer value for a Refresh to a different SIMM with a Read in progress	RW
Same_Busy_Refr_Rd	24:20	01011	Timer value for a Refresh to the same SIMM with a Read in progress	RW

Table 4-36 SIMM_Busy_Rd_Control Register

Field	Bits	PupState	Description	Type
Diff_Busy_Wr_Rd	19:15	01011	Timer value for a Write to a different SIMM with a Read in progress	RW
Diff_Busy_Rd_Rd	14:10	00101	Timer value for a Read to a different SIMM with a Read in progress	RW
Same_Busy_Wr_Rd	9:5	01100	Timer value for a Write to the same SIMM with a Read in progress	RW
Same_Busy_Rd_Rd	4:0	01000	Timer value for a Read to the same SIMM with a Read in progress	RW

The Count_Control Register is also referred to as Memory Control Register 08. The Count_Control contains various values necessary for the MC's programmability.

Row_Phi is the Phi Level value for an internal signal which determines whether the row address or the column address is output on the memory DRAM address lines.

StretchWR1 and StretchRD counts along with StretchWR0 from Memory Control Register 00, are the stretch points in the appropriate cycle when the MC checks the status of the appropriate buffer and determines whether the operation can proceed or if it needs to stall and stretch out the cycle until a time when it can finish the operation. i.e. - If we are doing consecutive reads from memory, and a CPU or U2S, for whatever reason, has not yet read out the XB1 Read buffer from the previous read, then the memory system cannot write into that buffer until the original requestor has emptied it. The MC however would like to go as far into the memory read cycle as it possibly can rather than stall the operation before it even drops RAS and CAS, so that when the buffer is ready, it will have the data ready also, and immediately load up the XB1 buffer with the correct data. (Similarly for writes, the MC would like to go as far into the write cycle as it can so that once the CPU or U2S writes into the buffer, the MC can immediately retrieve the data from the XB1 and store it.) The StretchXXX counts provide the mechanism for stalling at the correct part of the cycle.

The points where one would want to stretch the read or write cycle vary as a function of the system clock. The proper place to stretch reads and writes are as follows: In the case of a read, the mc should stretch the cycle starting from the cycle before the first positive going MRB_Ctrl pulse. In the case of a write, the first stretch point should be the cycle before the first MWB_Ctrl pulse, and the second should be before the second MWB_Ctrl pulse. To obtain the appropriate value for the CSR, we need to count the number of clocks from the start of the transaction to the appropriate pulse, then subtract two from that number. That is the correct

value to place the stretch at the appropriate place in the cycle. For example if the MRB_Ctrl stays low for five cycles, then pulses high, then programming StretchRd to 3 will place the stretch point at the correct part of the cycle.

Lastly, the MC has a set of “Ping-Pong” buffers which allow it to accept a second operation within 5 clock cycles while it is working on the first. PPXXCnt values, are values which control when the ping-pong buffers need to flip as a function of the operation (i.e. - a Read may require the address Ping-Pong buffer to remain for six cycles from the start of the operation, whereas, a Write may require something closer to eleven cycles before it can switch to the other buffer.) SW should program Count_Control according to the tables in Section 7.6.7.

The value of the PPXXCnt fields specify how long the transaction will drive the address bus with the correct address of the current transaction. When the count expires, the address output will flip their values to either the next pending transaction's address (if it has already queued up) or to the garbage address in the other ping-pong buffer which will be overwritten as soon as the next transaction comes in. In any case, the key point to note is to be sure that the address for the current transaction persists long enough for the CAS address hold time to be satisfied. The specific value for the ping pong buffer counts is calculated exactly as described above for the Stretch count values. i.e. - you count the number of cycles you need the address to drive onto the address bus, subtract two and program that value. If we had sixty nanosecond DRAMs with a CAS address hold time of 15 ns in a system with only a 12 ns clock, and CAS drops after three clock cycles, then we need the address to continue driving for an additional two clock cycles to satisfy the 15 ns CAS hold time, then the total number of clock cycles that the address needs to drive is five. The correct value then is 5 - 2 = 3. Then PPRdCnt needs to be programmed to be 3.

Table 4-37 Count_Control Register

Field	Bits	PupState	Description	Type
Reserved	31:30	00	reserved -read as zero.	R0
Row_Phi<4:0>	29:25	00011	Phase level word for internal Row signal	RW
Col1_Phi<4:0>	24:20	00000	Phase level word for internal Col1 signal	RW
StretchWr1<4:0>	19:15	00011	Stretch count value for second write	RW
StretchRd<4:0>	14:10	00011	Stretch count value for read	RW
PPWrCnt<4:0>	9:5	01001	Ping-Pong count value for write	RW
PPRdCnt<4:0>	4:0	00100	Ping-Pong count value for read	RW

4.3.5.10 Refresh_Control Register

The Refresh_Control Register is also referred to as Memory Control Register 09. It contains the Phase Control words for the RAS and CAS Waveform generators for a Refresh operation. Note that the Phase Level value for the first phase of RAS in a refresh cycle is NOT taken from RAS_Phi<0>, but from RefrPhiMC. SW should program Refresh_Control according to the tables in Section 7.6.8.

Table 4-38 Refresh_Control Register

Field	Bits	PupState	Description	Type
CAS_Ref<15:0>	31:16	0x2893	Phase control word for CAS on Refresh	RW
RAS_Ref<15:0>	15:0	0x2537	Phase control word for RAS on Refresh	RW

4.3.5.11 Row_Control Register

The Row_Control Register is also referred to as Memory Control Register 0A. It contains the Phase Control words for the Row Waveform generator. Row is an internal signal which controls the driving of the address lines with the Row part of the address. SW should program Row_Control according to the tables in Section 7.6.8.

Table 4-39 Row_Control Register

Field	Bits	PupState	Description	Type
Row_WR<15:0>	31:16	0x25b5	Phase control word for Row on Writes	RW
Row_RD<15:0>	15:0	0x2493	Phase control word for Row on Reads	RW

4.3.5.12 Reserved Register

The Reserved Register is also referred to as Memory Control Register 0B. It used to contain the Phase Control words for an internal signal for single bank mode which no longer exists.

4.3.5.13 SIMM_Busy_Wr_Control Register

The SIMM_Busy_Wr_Control Register is also referred to as Memory Control Register 0C. The DSC MC uses countdown timers on a given SIMM to determine how soon after an operation, it can issue a new operation to the same SIMM. Same_Busy_X_Wr contain the values those

countdown timers should use for a given operation to the same SIMMs given that an operation, X, is requested next and the current operation is a Write. Diff_Busy_X_Wr are analogous timer values when addressing a different SIMM. SW should program SIMM_Busy_Wr_Control according to the tables in Section 7.6.8.

Table 4-40 SIMM_Busy_Wr_Control Register

Field	Bits	PupState	Description	Type
Reserved	31:30	00	reserved -read as zero.	R0
Diff_Busy_Refr_Wr	29:25	00001	Timer value for a Refresh to a different SIMM with a Write in progress	RW
Same_Busy_Refr_Wr	24:20	01000	Timer value for a Refresh to the same SIMM with a Write in progress	RW
Diff_Busy_Wr_Wr	19:15	01001	Timer value for a Write to a different SIMM with a Write in progress	RW
Diff_Busy_Rd_Wr	14:10	00100	Timer value for a Read to a different SIMM with a Write in progress	RW
Same_Busy_Wr_Wr	9:5	01010	Timer value for a Write to the same SIMM with a Write in progress	RW
Same_Busy_Rd_Wr	4:0	01000	Timer value for a Read to the same SIMM with a Write in progress	RW

4.3.5.14 SIMM_Busy_Refr_Control Register

The SIMM_Busy_Refr_Control Register is also referred to as Memory Control Register 0D. The DSC MC uses countdown timers on a given SIMM to determine how soon after an operation, it can issue a new operation to the same SIMM. Same_Busy_X_Refr contain the values those countdown timers should use for a given operation to the same SIMMs given that an operation, X, is requested next and the current operation is a Refresh. Diff_Busy_X_Refr are analogous timer values when addressing a different SIMM. SW should program SIMM_Busy_Refr_Control according to the tables in Section 7.6.9.

Table 4-41 SIMM_Busy_Refr_Control Register

Field	Bits	PupState	Description	Type
Reserved	31:30	00	reserved -read as zero.	R0
Diff_Busy_Refr_Refr	29:25	00001	Timer value for a Refresh to a different SIMM with a Refresh in progress	RW
Same_Busy_Refr_Refr	24:20	01011	Timer value for a Refreshto the same SIMM with a Refresh in progress	RW
Diff_Busy_Wr_Refr	19:15	00001	Timer value for a Write to a different SIMM with a Refresh in progress	RW
Diff_Busy_Rd_Refr	14:10	00001	Timer value for a Read to a different SIMM with a Refresh in progress	RW
Same_Busy_Wr_Refr	9:5	01010	Timer value for a Write to the same SIMM with a Refresh in progress	RW
Same_Busy_Rd_Refr	4:0	01000	Timer value for a Read to the same SIMM with a Refresh in progress	RW

4.3.6 SC Coherence Control Registers

The SC Coherence control registers exist in the MP system only. They serve two purposes:

- To provide diagnostic access to the DTAG SRAM connected to the SC
- To log DTAG Parity Errors or Coherence Errors

4.3.6.1 CC_Diagnostic Register (0x70)

The CC_Diagnostic Register and the CC_Snoop Vector Register provide the ability to read and write the system DTAG SRAM.

Table 4-42 CC_Diagnostic Register

Field	Bits	PupState	Description	Type
SnpIndex	31:16	x	SRAM Address	RW
DMODE	15	0	Enables writing to the DTAG	RW
Reserved	14:0	0	Reserved - read as 0	R0

SnpIndex

This field represents the SRAM address to be read or written by subsequent accesses to the CC_Data Register. The index covers the entire range of tag rams which could be needed. Only those parts corresponding to the current cache configuration have meaning.

Note – Bit 16 selects between DTAG 0, the tag for P0, and DTAG 1, the tag for P1. 0 = P0, 1 = P1.

DMODE

SW sets this bit to enable the indirect access of the SRAM DTAG through the CC_Snoop_Vector Register. This bit may be set for diagnostic purposes only. As listed in the section describing the fault register, diagnostic mode accesses which detect a parity error are logged but do not cause a system (fatal) reset. This allows diagnostics software to check out the parity generation and checking logic.

4.3.6.2 CC_Snoop Vector Register (0x74)

When DMODE in the CC_Diagnostic register is set, SW can access the DTAG SRAM directly by reading and writing to this register. The address of the SRAM for the access is taken from the SnpIndex field of the CC_Diagnostic Register.

Table 4-43 CC_Data Register

Field	Bits	PupState	Description	Type
Reserved	31	0	Reserved	R0
SnpTag	30:19	NA	The “tag” portion of the SRAM Data	R/W

Table 4-43 CC_Data Register

Field	Bits	PupState	Description	Type
SnpState	18:17	NA	The “state” portion of the SRAM Data	R/W
SParity	16	NA	The parity portion of the SRAM Data; when writing data in diagnostics mode, parity comes from this bit, not the parity generation logic.	R/W
Reserved	15:0	0	Reserved - Read as 0	R0

4.3.6.3 CC Fault Register (0x78)

This register contains error information on a port by port basis that was reported during a coherent transfer. The Coherence Errors are specified in the “UPA Interconnect Architecture” Document.

Note – All of the errors listed in this register are FATAL and will result in a system reset if detected during normal operation. During diagnostic mode access, parity errors detected on a read will be logged in the fault register, but will NOT cause a system reset.

Table 4-44 CC_Fault Register

Field	Bits	PupState	Description	Type
PERR0	31	0	Dual Tag Parity error detected Proc. 0	R/W1C
CERR0	30	0	Coherence Error detected, Proc 0	R/W1C
PERR1	29	0	Dual Tag Parity Error detected, Proc. 1	R/W1C
CERR1	28	0	Coherence Error detected, Proc. 1	R/W1C
FLTIndex	27:13	X	Index of fault (parity or coherence error)	R/W
Reserved	12:0	0	Reserved - Read as zero	R0

4.3.6.4 CC_Proc_Index Register (0x7c)

This register contains a mask of a width equivalent to the width of the cache index for the 2 Processor Ports the SC Supports.

Table 4-45 CC_Proc_Index Register

Field	Bits	PupState	Description	Type
Reserved	31	x	Reserved	R/W
PIndex	30:0	0xXXXX	address index mask for Ports 0 & 1	R/W

Software must set the value of this register before it enables system caches. It contains a mask which is 1 in the bit positions corresponding to the tag portion of the physcial address.

To calculate the value for this register, the following steps are taken:

a. Create a mask of 1's starting at bit 30 and going to the bit number which is the log base 2 of the cache size installed in ports 0 and 1. All other bits in the mask should be 0.

Note – Cache sizes for the CPUs must match. If they don't, boot SW should disable one of the processors.

---

b. Write this value to the CC_Proc_Index register.

Only 3 cache sizes are supported: 512KB, 1 MB, and 2 MB. The values which should be used for these cases are given below.

- 512K = 0x7ff8.0000
- 1 M = 0x7ff0.0000
- 2 M = 0x7fe0.0000

Packet Handling 5

5.1 Introduction U2S

One of the major functions of DSC is the forwarding and handling of UPA packets. This chapter describes in detail how DSC accomplishes this.

The primary address bus of the system is address bus 0. The DSC examines every packet on this bus. It then “forwards” this packet to the appropriate destination based on the physical address of the packet. For address bus 0 the possible sources for packets are:

- Processor 0
- Processor 1
- U2S or other UPA IO device
- System Controller

The possible destinations for packets for address bus 0 are:

- Processor 0
- Processor 1
- U2S
- Memory Controller which resides on the DSC
- FFB through Addressbus 1, which is also connected to the DSC

Addressbus 1 is a unidirectional bus from the DSC to the FFB. The FFB does not generate any packets for this bus.

Not all packets are supported by every port. Below is a table of the possible transactions and whether or not they are supported by a particular port.

Table 5-1 Transactions supported

Transaction Type	Name	Processor	U2S	FFB	System Controller
P_RDS_REQ	ReadToShare	Gen			Rec
P_RDSA_REQ	ReadToShareAlways	Gen			Rec
P_RDO_REQ	ReadToOwn	Gen	Gen		Rec
P_RDD_REQ	ReadToDiscard	Gen	Gen		Rec
S_CPD_MSI_REQ	CopybackGotoState	Not used	Not used	Not used	Not used
P_NCRD_REQ	NoncachedRead	Gen/Rec	Gen/Rec	Rec	For
P_NCBRD	NoncachedBlockRead	Gen	Gen/Rec	Rec	For
P_NCBWR	NoncachedBlockWrite	Gen	Gen/Rec	Rec	For
P_WRB_REQ	Writeback	Gen	Gen		Rec
P_WRI_REQ	WritebackInvalidate	Gen	Gen		Rec
S_INV_REQ	Invalidate	Rec			Gen
S_CPB_REQ	CopyBack	Rec			Gen
S_CPI_REQ	CopyBackInvalidate	Rec			Gen
S_CPD_REQ	CopyBackToDiscard	Rec			Gen
P_NCWR_REQ	NoncachedWrite	Gen	Gen/Rec	Rec	For
P_INT_REQ	Interrupt	Gen/Rec	Gen		For
Gen = Generates, Rec = Receives, For = Forwards					

5.2 Address mapping

Below you will find a table for the entire address range of Ultra 2 System.

Table 5-2 DSC Address Map

Address Range in UPA Address<40:0>	Range name	Size (Gbytes)	UPA Port Addressed	Port ID[1]	Notes
0x0 - 0x000.7FFF.FFFF	MEM_ADDR	2	Main Memory	NA	SC Responds to entire address range.
0x000.8000.0000 - 0x0FF.FFFF.FFFF	ERR_ADDR	1022	Unimpleme nted space		Results in RTO on Reads SC flags IADDR.
0x100.0000.0000 - 0x1BF.FFFF.FFFF	ERR_ADDR	768	Invalid		Results in RTO on Reads. SC flags IADDR.
0x1C0.0000.0000- 0x1C1.FFFF.FFFF	CPU_ADDR	8	Processor 0	0	NC registers.
0x1C2.0000.0000- 0x1C3.FFFF.FFFF	CPU_ADDR	8	Processor 1	1	NC registers.
0x1C4.0000.0000- 0x1FB.FFFF.FFFF	RSVD_ADDR	224	Reserved		SC flags IADDR.
0x1FC.0000.0000- 0x1FD.FFFF.FFFF	FFB_ADDR	8	On Board	1e[2]	UPA Slave (FFB)
0x1FE.0000.0000- 0x1FF.FFFF.FFFF	U2S_ADDR	8	U2S	1f	System IO space.

1. PortID is the Master ID (MID) for devices which support mastership (in hex).

2. FFB is always slave

5.3 DSC Transaction Table

Transactions on Address bus 0 may result in additional transactions on the two address busses due to forwarding and cache coherency maintenance. The following table shows the transaction and it's additional transactions.

5.3.1 Definitions and Abbreviations

Each of the UPA transaction names has been shortened here to minimize the table size. Here are the abbreviations:

Table 5-3 Transaction table Abbreviations

Transaction Type	Name	Abbreviation
P_RDS_REQ	ReadToShare	RDS
P_RDSA_REQ	ReadToShareAlways	RDSA
P_RDO_REQ	ReadToOwn	RDO
P_RDD_REQ	ReadToDiscard	RDD
S_CPD_MSI[1]	CopybackGotoState	MSI
P_NCRD_REQ	NonCachedRead	NCRD
P_NCBRD	NonCachedBlockRead	NCBRD
P_NCBWR	NonCachedBlockWrite	NCBWR
P_WRB_REQ	Writeback	WRB
P_WRI_REQ	WritebackInvalidate	WRI
S_INV_REQ	Invalidate	INV
S_CPB_REQ	CopyBack	CPB
S_CPI_REQ	CopyBackInvalidate	CPI
S_CPD_REQ	CopyBackToDiscard	CPD
P_NCWR_REQ	NonCachedWrite	NCWR
P_INT_REQ	Interrupt	INT

1. This transaction is supported by UltraSPARC-I but not in UPA compliant interconnect.

Table 5-4 Transaction Table address and interrupt name abbreviations

Range name	Definition
MEM_ADDR	Main memory address
CPU_ADDR	CPU address space
U2S_ADDR	U2S address space

Table 5-4 Transaction Table address and interrupt name abbreviations (Continued)

Range name	Definition
FFB_ADDR	FFB address space
SLAVE_ADDR	CPU_ADDR or U2S_ADDR or FFB_ADDR
ERR_ADDR	Not supported address range
RSVD	Reserved.
SAME_ADDR	Addressed to one's self
VAL_ID	Valid interrupt target ID
INV_ID	Invalid target ID
SAME_ID	Target ID is one's self
IADDR	IADDR bit in SC_Port_Status register
IPORT	IPORT bit in SC_Port_Status register

Table 5-5 P_Reply definitions

Name	Definition	Notes
P_IDLE	IDLE	
P_RTO	Read time out	
P_RERR	Read error	
P_FERR	Fatal error	
P_RAS	Read Ack Single	
P_RAB	Read Ack Block	
P_RASB	Read Ack Single or Block	
P_WAS	Write Ack Single	
P_WAB	Write Ack Block	
P_IAK	Interrupt Ack	
P_SACK	Coherent Read Ack Block	
P_SACKD	Coherent Read Ack Block	With dirty victim
P_SNACK	Nonexistent block	No data transferred

Table 5-6 S_Reply definitions

Name	Definition	Destination
S_IDLE	IDLE	
S_RTO	Read time-out	Master
S_ERR	Error, no data transferred	Master
S_WAS	Write Ack Single	Master
S_WAB	Write Ack Block/ Int Ack	Master
S_OAK	Ownership Ack	Master
S_RBU	Read Block Unshared	Master
S_RBS	Read Block Shared	Master
S_RAS	Read Ack Single	Master
S_SRS	Read Single Ack	Slave
S_SRB	Read Block Ack	Slave
S_CRAB	Copyback Read Block Ack	Slave
S_SWB	Write Block Ack	Slave
S_SWS	Write Single Ack	Slave
S_SWIB	Interrupt Write Block	Slave
S_WBCAN	Writeback Cancel	Master
S_INAK	Interrupt no-ack	Master

5.3.2 Transaction Table

Table 5-7 DSC Transaction Table

Trans-action Type	Master ID	Address	P_ or S_ request	P_Reply	S_Reply to Master	S_Reply to Slave	Comments
NCRD	CPU, U2S	CPU_ADDR U2S_ADDR (OneRead = 0)	NCRD	P_RAS P_RERR P_RTO	S_RAS S_ERR S_RTO	S_SRS S_SRS -	1 to 16 byte transfer - -
		CPU_ADDR U2S_ADDR (OneRead = 1)	NCRD	P_RASB[7] P_RERR P_RTO	S_RAS S_ERR S_RTO	S_SRS S_SRS -	1 to 16 byte transfer - -
		FFB_ADDR	NCRD	P_RASB	S_RAS	S_SRS	
		MEM_ADDR	-	-	S_ERR	-	SC sets IADDR
		ERR_ADDR	-	-	S_ERR	-	SC sets IADDR
		SAME_ADDR	-	-	S_RAS	-	Returns ambiguous data
		RSVD_ADDR	-	-	S_RTO	-	Reserved address range
NCBRD	CPU, U2S	CPU_ADDR U2S_ADDR (OneRead = 0)	NCBRD	P_RAB P_RERR P_RTO	S_RBU S_ERR S_RTO	S_SRB S_SRB -	64 bytes - -
		CPU_ADDR U2S_ADDR (OneRead = 1)	NCBRD	P_RAB P_RASB P_RERR P_RTO	S_RBU S_RBU S_ERR S_RTO	S_SRB S_SRB - -	64 bytes - - -
		FFB_ADDR	NCRD	P_RASB	S_RAS	S_SRS	
		MEM_ADDR	-	-	S_ERR	-	SC sets IADDR
		ERR_ADDR	-	-	S_RTO	-	SC sets IADDR
		SAME_ADDR	-	-	S_RBU	-	Returns ambiguous data
		RSVD_ADDR	-	-	S_RTO	-	Reserved address range
NCWR	CPU, U2S	U2S_ADDR FFB_ADDR	NCWR	P_WAS	S_WAS	S_SWS	1 - 16 byte write
		CPU_ADDR	-[2]	-	S_WAS	-	SC sets IADDR, data dropped

Table 5-7 DSC Transaction Table (Continued)

Trans-action Type	Master ID	Address	P_ or S_ request	P_Reply	S_Reply to Master	S_Reply to Slave	Comments
		MEM_ADDR	-	-	S_WAS	-	SC sets IADDR, data dropped
		ERR_ADDR	-	-	S_WAS	-	SC sets IADDR, data dropped
		SAME_ADDR	-	-	S_WAS	-	SC sets IADDR, data dropped
		RSVD_ADDR	-	-	S_WAS	-	SC sets IADDR, data dropped
NCBWR	CPU, U2S	SLV_ADDR	NCBWR	P_WAB	S_WAB	S_SWB	
		CPU_ADDR	-[3]	-	S_WAB	-	SC sets IADDR, data dropped
		MEM_ADDR	-	-	S_WAB	-	SC sets IADDR, data dropped
		ERR_ADDR	-	-	S_WAB	-	SC sets IADDR, data dropped
		SAME_ADDR	-	-	S_WAB	-	SC sets IADDR, data dropped
		RSVD_ADDR	-	-	S_WAB	-	SC sets IADDR, data dropped
INT	CPU, U2S	VAL_ID	INT	P_IAK	S_WAB	S_SWIB	
		INV_ID	-	-	S_WAB	-	SC sets IPORT
		VAL_ID but nacked	-	-	S_INAK	-	-
		SAME_ID	-	-	S_WAB	-	Packet is dropped,
RDS	CPU	MEM_ADDR, other Proc = I, S[1]	-	-	S_RBU	-	from memory

Table 5-7 DSC Transaction Table (Continued)

Trans-action Type	Master ID	Address	P_ or S_ request	P_Reply	S_Reply to Master	S_Reply to Slave	Comments
		MEM_ADDR, other Proc = O,M	CPB	P_SACK P_SACKD P_SNACK	S_RBS S_RBS -	S_CRAB S_CRAB -	from other processor Block is pending writeback Fatal Error
		SLAVE_ADDR	-	-	S_ERR	-	SC sets IADDR
		ERR_ADDR	-	-	S_ERR	-	SC sets IADDR
		SAME_ADDR	-	-	S_RBU	-	Return ambiguous data
		RSVD_ADDR	-	-	S_ERR	-	Coherent read to slave addr
	U2S	any address	UNDE-FINED[4]	-	-	-	UNDEFINED
RDSA	CPU	MEM_ADDR, other Proc = I, S[5]	-	-	S_RBS	-	data from memory
		MEM_ADDR, other Proc = O,M	CPB	P_SACK P_SACKD P_SNACK	S_RBS S_RBS -	S_CRAB S_CRAB -	from other processor Block is pending writeback Fatal Error
		SLAVE_ADDR	-	-	S_ERR	-	
		ERR_ADDR	-	-	S_ERR	-	SC sets IADDR
		SAME_ADDR	-	-	S_RBS	-	Return ambiguous data
		RSVD_ADDR	-	-	S_ERR	-	Coherent read to slave addr
	U2S	any address	UNDE-FINED[1]	-	-	-	UNDEFINED[1]
RDO	CPU	MEM_ADDR	-	-	S_RBU	-	from memory

Table 5-7 DSC Transaction Table (Continued)

Trans-action Type	Master ID	Address	P_ or S_ request	P_Reply	S_Reply to Master	S_Reply to Slave	Comments
	CPU St = I	MEM_ADDR, other Proc = S,O,M	CPI	P_SACK P_SACK D P_SNAC K	S_RBU S_RBU -	S_CRAB S_CRAB -	from other processor Block is pending writeback Fatal Error
	CPU St = S	MEM_ADDR, other Proc = S,O	INV	P_SACK P_SACK D P_SNAC K	S_OAK S_OAK S_OAK	- - -	- Block pend. wb cancel Should never happen in MP
	CPU St = O	MEM_ADDR, other Proc = S	INV	P_SACK P_SNAC K	S_OAK S_OAK	- -	- Should never happen in MP
	U2S	MEM_ADDR	-	-	S_RBU	-	from memory
	U2S	MEM_ADDR, other Proc = S,O,M	CPI for S,O,M, INV for added S	P_SACK P_SACK D P_SACK P_SNAC K P_SNAC K	S_RBU S_RBU - - S_RBU	S_CRAB S_CRAB - -	- Block is pending writeback Invalidate response Fatal Error to CPI Should never happen in MP
	CPU, U2S	SLAVE_ADDR	-	-	S_ERR	-	-SC sets IADDR
		ERR_ADDR	-	-	S_ERR	-	SC sets IADDR
		SAME_ADDR	-	-	S_RBU	-	Return ambiguous data
		RSVD_ADDR	-	-	S_ERR	-	Coherent read to slave addr
RDD	CPU	MEM_ADDR	-	-	S_RBS	-	from memory, no state change

Table 5-7 DSC Transaction Table (Continued)

Trans-action Type	Master ID	Address	P_ or S_ request	P_Reply	S_Reply to Master	S_Reply to Slave	Comments
		MEM_ADDR, other Proc = S,O,M	CPD	P_SACK P_SACKD P_SNACK	S_RBS S_RBS -	S_CRAB S_CRAB -	Send to 1 processor only Block is pending writeback Fatal Error
	U2S	MEM_ADDR	CPD	P_SACK P_SACKD P_SNACK	S_RBS S_RBS S_RBS	S_CRAB S_CRAB	- - Fatal Error
	CPU, U2S	SLAVE_ADDR	-	-	S_ERR	-	SC sets IADDR
		ERR_ADDR	-	-	S_RTO	-	SC sets IADDR
		SAME_ADDR	-	-	S_RBS	-	Return ambiguous data
		RSVD_ADDR	-	-	S_ERR	-	Coherent read to slave addr, still S_ERR
WRB	CPU, U2S	MEM_ADDR	-	-	S_WAB	-	Proc has O or M data. Memory gets updated.
		MEM_ADDR A Proc does WRI	-	-	S_WBCAN	-	Reply when data has been invalidated.
	CPU	MEM_ADDR, Other Proc wants data	-	-	S_CRAB and S_WBCAN	-	The S_CRAB transfers data to the other proc. Then it cancels the Writeback.
	U2S	MEM_ADDR A Proc wants data	-	-	S_WAB	-	Other processor is blocked until data is written to memory.
	CPU, U2S	SLV_ADDR	UNDE-FINED[5]	-	-	-	
		ERR_ADDR	UNDE-FINED[6]	-	-	-	

Table 5-7 DSC Transaction Table (Continued)

Trans-action Type	Master ID	Address	P_ or S_ request	P_Reply	S_Reply to Master	S_Reply to Slave	Comments
		SAME_ADDR	UNDEFINED	-	-	-	Same as ERR_ADDR
		RSVD_ADDR	UNDEFINED	-	-	-	
WRI	CPU, U2S	MEM_ADDR	INV to CPU(s)	P_SACK P_SACKD P_SNACK	S_WAB S_WAB S_WAB	- - -	write to memory write to memory Should never happen in MP
		CPU_ADDR	-	-	S_WAB	-	CPU doesn't accept Writes, data is dropped, SC sets IADDR
		U2S_ADDR	-	-	S_WAB	-	U2S doesn't accept writes, data is dropped, IADDR set
		FFB_ADDR	-	-	S_WAB	-	FFB doesn't accept WRIs, data is dropped, IADDR set
		ERR_ADDR	-	-	S_WAB	-	data is dropped, SC sets IADDR
		SAME_ADDR	-	-	S_WAB	-	data is dropped, IADDR set
		RSVD_ADDR	-	-	S_WAB	-	data is dropped, IADDR set

1. Although this could be considered an intervention update policy, we break the rules when the data is in the shared state and get the data from memory instead. This is done for speed reasons.
2. Any P_NCWR_REQ / P_NCBWR_REQ transactions directed to the processor are not forwarded (because the processor has a slave data queue size of 0), and data is dropped on the floor.
3. Any P_NCWR_REQ / P_NCBWR_REQ transactions directed to the processor are not forwarded (because the processor has a slave data queue size of 0), and data is dropped on the floor.
4. U2S is incapable of generating a P_RDS_REQ or P_RDSA_REQ transaction. If U2S does generate one of these transactions, DSC will not detect it as an error, and what happens is undefined.
5. DSC does not support cacheable address space at the UPA slaves. If a UPA port issues a P_WRB_REQ to a UPA slave, then a hardware error has occurred since it could not have ever successfully completed a coherent read to that address. However, DSC will not detect it as an error, and what happens is undefined.
6. If a UPA port issues a P_WRB_REQ to an illegal or invalid address, then a hardware error has occurred since it could not have ever successfully completed a coherent read to that address. However, DSC will not detect it as an error, and what happens is undefined
7. The processor cannot generate P_RASB. U2S is capable of generating P_RASB.

Note – If the processor is performing a coherent read to an Illegal address space (above 2 GB) and is evicting a modified line because of that Read, DSC will handle the writeback properly and return a RTO for the Read.

5.3.3 Global view of S_Replys

The Table 5-8 shows global view of S_Replys over the DSC address range. Shown are also special cases of S_Replys when P0 or P1 or FFB are not present in system.

Table 5-8 Global view of S_Replys

Address Range (from - to)	coh_rd	coh_wr	nc_rd	nc_wr
0x000.0000.0000 **(first 2 GB of memory)** 0x000.7fff.ffff	see UPA spec	see UPA spec	S_ERR[1]	S_WAS/S_WAB (data dropped)
0x000.8000.0000 **(memory over 2 GB)** 0x0ff.ffff.ffff	S_RTO[1]	S_WAB (hangs for WRB)	S_RTO[1]	S_WAS/S_WAB (data dropped)
0x100.0000.0000 **(Reserved)** 0x1bf.ffff.ffff	S_RTO[1]	S_WAB (hangs for WRB)	S_RTO[1]	S_WAS/S_WAB (data dropped)
0x1c0.0000.0000 **(Processor 0)** 0x1c1.ffff.ffff	S_ERR (if P0 is not present S-Reply = S_RTO)[1]	S_WAB (hangs for WRB)	S_RAS/S_RBU (if P0 is not present S-Reply = S_RTO)[1]	S_WAS/S_WAB[1] (data dropped)
0x1c2.0000.0000 **(Processor 1)** 0x1c3.ffff.ffff	S_ERR (if P1 is not present S-Reply = S_RTO)[1]	S_WAB (hangs for WRB)	S_RAS/S_RBU (if P1 is not present S-Reply = S_RTO)[1]	S_WAS/S_WAB[1] (data dropped)

Table 5-8 Global view of S_Replys (Continued)

0x1c4.0000.0000 **(Reserved)** 0x1fb.ffff.ffff	S_RTO[1]	S_WAB (hangs for WRB)	S_RTO[1]	S_WAS/S_WAB (data dropped)
0x1fc.0000.0000 **(FFB)** 0x1fd.ffff.ffff	S_ERR (if FFB is not present S-Reply = S_RTO)[1]	S_WAB (hangs for WRB)	S_RAS/S_RBU (if FFB is not present S-Reply = S_RTO)[1]	S_WAS/S_WAB
0x1fe.0000.0000 **(U2S)** 0x1ff.ffff.ffff	S_ERR[1]	S_WAB (hangs for WRB)	S_ERR	S_WAS/S_WAB

1. Set IADDR bit in Processor's Status Register.

5.4 Coherent Transactions

5.4.1 Coherence Algorithm

Coherent transactions (P_RDS_REQ, P_RDSA_REQ, P_RDD_REQ, P_RDO_REQ, P_WRB_REQ, P_WRI_REQ) can only be issued to main memory. DSC does not support cacheable accesses to any slave address space other than main memory. Coherent requests always result in the transfer of an entire block (64 bytes) of data.

DSC implements a dual tag system. It maintains state and tag information for every line in each of the two processors. The processors E-tags have five states: M,O,E.S,I. The dual tags contain four states, M, O, S, and I. The E-tags states M and E are both represented by the dual tag M state. The processor is free to change from the E to M state without accessing the system.

As it turns out, DSC would only have to implement a MSI system, where M and O are treated identically.

The cache coherency mechanism is designed to work with a U2S which has only a single line merge buffer. There are very strict rules on how U2S handles this merge buffer. U2S maintains a single 64 byte merge buffer for performing writes smaller than 64 bytes to coherent space. If U2S wants to perform a write of less than 64 bytes to coherent space, then it must issue a P_RDO_REQ, perform the write, and flush it back to main memory immediately using the P_WRB_REQ. If U2S wants to read a block with no intent to write, then it issues a P_RDD_REQ transaction. If U2S wants to do a block write to memory, then it uses the P_WRI_REQ.

When U2S has possession of a memory block, any requests to that block from the processor will be blocked until U2S writes it back to main memory. CC maintains the address and compare all incoming requests from the processor against that address. If there is a hit on that address, then that request and all subsequent requests from that class will stall. After U2S issues the P_WRB_REQ, then the stalled transactions will unblock. Therefore it is important that U2S write back the block as soon as possible.

Address checking is done against the physical address bits in the first half of the request packet: PA[40:14] and PA[8:6]. For main memory accesses, bits 39:31 are ignored.

DSC will never issue an S_Request (S_INV_REQ, S_CPB_REQ, S_CPI_REQ, S_CPD_REQ) to U2S because of a coherent request issued from a processor.

The IVA bit in Ultra 2 is ignored. (**I**n**V**alidate me **A**dvisory bit in **P_WRI_REQ** transaction only is used in implementations without Dtags. The master UPA Port sets this bit if it wants the System Controller to send a **S_INV_REQ** to this port.)

Every time U2S issues a coherent read transaction, DSC checks the Dtags to see if copybacks and/or invalidations to the processor(s) are required.

U2S is not allowed to issue P_RDS_REQ or P_RDSA_REQ.

DSC will not detect the case when U2S does not issue a P_WRB_REQ to flush the block from the merge buffer back to main memory (for example, two back to back P_RDO_REQs). This is a coherence error and what happens is undefined.

Note – The PIF does not maintains a “writeback cancel” bit for each processor as does the USC. In DSC is this functionality in CC block which keeps track of Writeback Buffer index.

5.4.2 Coherence State Transition Tables

The followings tables show what happens to the Dual tags when a coherent transaction is processed. P0 is the requesting processor. The following states are illegal in the cache coherency algorithm and are not specified in the tables.

- P0 = S and P1 = M.
- P0 = O and P1 = M.
- P0 = M and P1 = M.
- P0 = O and P1 = O.

(While one cache has the block in Dtag M state no other cache may have a copy of that block. While one cache has the block in Dtag O state, the other cache having that block must have it in the Dtag S state only.)

5.4.2.1 Exceptions from UPA Specification.

Under the Error Management section of the UPA, there is a statement:
"Coherent read of data when it is already in ones own cache is illegal and returns undefined data." The DSC will detect this and cause a fatal error instead.

Table 5-9 P_RDS_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
I	I	M	I	-	data from memory
I	S	S	S	-	data from memory
I	O	S	O	S_CPB_REQ	data from P1
I	M	S	O	S_CPB_REQ	data from P1
S	I, S,O, M	-	-	-	Fatal Error
O	I, S, O, M	-	-	-	Fatal Error
M	I, S, O, M	-	-	-	Fatal Error

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified

Table 5-10 P_RDSA_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
I	I	S	I	-	data from memory
I	S	S	S	-	data from memory
I	O	S	O	S_CPB_REQ	Data from P1

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified

Table 5-10 P_RDSA_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
I	M	S	O	S_CPB_REQ	Data from P1
S	I, S, O, M	-	-	-	Fatal Error
O	I, S, O, M	-	-	-	Fatal Error
M	I, S, O, M	-	-	-	Fatal Error

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified

Table 5-11 P_RDO_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
I	I	M	I	-	Data from memory
I	S	M	I	S_CPI_REQ	Data from P1
I	O	M	I	S_CPI_REQ	Data from P1
I	M	M	I	S_CPI_REQ	Data from P1
S	I	M	I	-	S_OAK reply
S	S	M	I	S_INV_REQ	S_OAK reply
S	O	M	I	S_INV_REQ	S_OAK reply
S	M	-	-	-	Fatal Error
O	I	M	I	-	S_OAK reply
O	S	M	I	S_INV_REQ	S_OAK reply
O	O, M	-	-	-	Fatal Error
M	I, S, O, M	-	-	-	Fatal Error

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified

Table 5-12 P_RDD_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
X	I	X	I	-	data from memory
X	S	X	S	-	data from memory
X	O	X	O	S_CPD_REQ	No state change
X	M	X	M	S_CPD_REQ	No state change

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified, X = Don't Care (CC does not check P0 state)

Table 5-13 P_WRB_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
O	I	I	I		data put in mem
O	S	I	S		data put in mem
M	I	I	I		data put in mem

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified, The PO Initial State is the state of the block being written back and is kept in the "Nth +1" cache location in the DSC.

Table 5-14 P_WRI_REQ state transitions

P0 Initial State	P1 Initial State	P0 Final State	P1 Final State	Secondary Trans to other Processor	Comments
I	I	I	I		
I	S	I	I	S_INV_REQ	
I	O	I	I	S_INV_REQ	
I	M	I	I	S_INV_REQ	
S	I	I	I	S_INV_REQ[1]	
S	S	I	I	S_INV_REQ[1]	
S	O	I	I	S_INV_REQ[1]	
O	I	I	I	S_INV_REQ[1]	
O	S	I	I	S_INV_REQ[1]	
M	I	I	I	S_INV_REQ[1]	

P0 = requesting processor, I = invalid, S = Shared, O = Owner, M = Modified

1. Both processors, including the originator, must have this item in invalid state before the SC can give the S_WAB reply.

5.4.3 Request Flow

5.4.3.1 Coherent Read from Processor

The PIF block forwards all coherent reads within the Valid address range to both the MC and the CC. This allows the MC to start the DRAM access while the CC is determining the actions that should be taken by looking up the Dual tags. Should the data from memory not be needed, the MC is asked to squash that memory access. Data is taken from the other processor only in the Modified or Owned states (RDO gets data from other processor even in S state). The CC tells the PIF that invalidate and/or copyback requests are required to be sent. The CC does not update the tags until the CPU has taken the data or more information, see the preceding tables in this chapter.

5.4.3.2 Coherent Write from Processor

Once again, the PIF forwards the cacheable writes to the CC. The CC then determines what appropriate coherent operations are needed. All valid, coherent writes go to memory. For more information, see the preceding tables in this chapter.

5.4.3.3 Coherent Read from U2S

U2S is only allowed to issue P_RDO_REQ or P_RDD_REQ. Since it has no cache, it cannot issue P_RDS_REQ or P_RDSA_REQ.

5.4.3.4 Coherent Write from U2S

U2S will issue P_WRB_REQ (as part of a RMW) or P_WRI_REQ to inject new data into the coherence domain. If DSC sees a P_WRB_REQ, it assumes that it is coupled with a previous P_RDO_REQ. While the U2S has possession of the cache line, any processor trying to access that cache line has its request blocked. Once the write is issued to main memory the cache line becomes “unblocked”; that is, processors can now access this memory line.

If SC sees a P_WRI_REQ, it checks the dual tags and issues S_INV_REQ to processors that have valid data (State = S,O, or M). Then the SC waits for the PREPLYs to come back before accepting the data from the U2S.

5.4.3.5 Coherent Requests to FFB

Transactions to the FFB require special handling because FFB does not generate a full 5 bit PREPLY. It is unable to signal an error to DSC. If a coherent request is issued to FFB, it has no way of signalling P_RERR or P_RERRC. To handle these transactions, DSC will generate an S_ERR on behalf of the FFB if it sees a P_RDS_REQ, P_RDSA_REQ, P_RDD_REQ, or P_RDO_REQ directed to FFB. The packet will not be forwarded to FFB, it will be intercepted by DSC. If DSC sees P_WRB_REQ issued to FFB, the results are undefined. A P_WRI_REQ to the FFB results in the SC issuing a S_WAB but dropping data on the floor.

5.5 Non-cached Transactions

Non-cached transactions (P_NCBWR_REQ, P_NCWR_REQ, P_NCBRD_REQ, P_NCRD_REQ) can be generated by the processor or U2S. They are simpler than coherent transactions because handling of these transactions is limited to the PIF and the DPS, and the MC is never involved.

For non-cached reads (P_NCBRD_REQ, P_NCRD_REQ), the PIF will forward the packet to the appropriate slave, and wait for a PREPLY from the slave. Then the DPS will issue the SREPLYs to both master and slave to transfer the data. If the PREPLY indicates an error, then this is forwarded to the master by the DPS.

For non-cached writes (P_NCBWR_REQ, P_NCWR_REQ), the PIF will forward the packet to the appropriate slave. The DPS will then issue the SREPLYs to master and slave to transfer the data. Then the PIF waits for a PREPLY to be issued from the slave, so that it can decrement its count of outstanding transactions to that slave.

5.6 Interrupts

P_INT_REQ is very similar to P_NCBWR_REQ with the following differences.

DSC will check to see if the slave's interrupt queue has enough space for the interrupt. If yes, then it will transfer the interrupt packet from the master to the slave. If not, it issues a S_INAK to master, who will have to retry the interrupt request later.

Also, if the target port is in sleep mode, DSC will begin the wake up sequence for the target, and issue S_INAKs to the master until the SLEEP bit in the DSC for that slave has been cleared.

5.7 Flow Control

The sizes of DSC's master request queues is listed in the table below.

Table 5-15 DSC Master Queue Sizes

Queue Name	Queue Depth is 2 Cycle Packets	Note
MRQ0/CPU	1	master request queue for CPU class 0
MRQ1/CPU	8	master request queue for CPU class 1
MRQ/U2S	2	master request queue for U2S

All transactions issued by U2S collapse into a single master request queue, regardless of the class bit in the transaction. DSC expects all requests from U2S to be issued from class 0.

DSC performs flow control by knowing the length of the PRequest and Interrupt queues at the slave, per the UPA Architecture. These are programmed into the SC_Port_Config registers by the boot processor after power on. DSC will never issue more requests to a slave than it has room for.

The maximum number of outstanding transactions varies substantially and is generally determined by the queue sizes in the SC and at the UPA ports. The goal in sizing DSC's internal resources will be to match the processor, U2S, and FFB's slave request queues so that DSC itself is never the limiting factor in how many outstanding transactions can be issued to the slave ports at one time.

5.8 Blocking Conditions

The DSC keeps transactions from each UPA master strongly ordered within each class. There is no ordering between different classes or requests from different UPA master. The S_REPLY for the transactions in each class are issued by SC in the same order to the requesting master UPA port, as the order in which the transaction requests were originally issued by it.

5.8.1 Blocking for Non-cached Transactions

In this and the next section, "pending" means that an S_REPLY for the transaction has not been generated yet.

A non-cached transaction from any class is blocked if:

1. There is a preceding read transaction and the current transaction is a write from the same class and the same source.
2. There is a preceding transaction going to a different slave from the same class and the same source.
3. There is a preceding cached transaction from the same master class from the same source and the current transaction is ncwr, ncbwr or int.
4. There is a preceding ncwr, ncbwr or int and the current transaction is ncrd (dps blocking rule).
5. The slave queue for the addressed slave is full.
6. The DPS queue is full.

These blocking rules still allow multiple outstanding writes from the same source and the same class as well as multiple outstanding reads from the same source and the same class should the slave allow it.

5.8.2 Blocking Conditions for Cached Transactions

A cached transaction is blocked if:

1. The current transaction is a read and the CC or MC block is busy.
2. The MC is busy and the current transaction is a write.
3. There is an outstanding SRequest and the current transaction needs to send an SRequest to the same processor
4. There is outstanding nc transaction and the current transaction is coherent.
5. There is already one outstanding coherent transaction from the requesting processor.
6. U2S is doing a RMW and the current CPU transaction accesses the same block.
7. There is an outstanding SRequest and the current transaction is a CPU writeback for the same block.
8. If the CC is busy updating the Dtag.
9. If the MC's queue is full.

5.9 Class Symmetry

5.9.1 Processor

DSC does not implement perfect class symmetry; that is, it assumes that certain transactions are always issued in certain classes from the processor. In particular, DSC will only support the transaction to class allocation that UltraSPARC-I actually implements.

1. PROC_CLASS0: P_RDS_REQ, P_RDSA_REQ, P_RDO_REQ, P_RDD_REQ, P_NCBRD_REQ.
2. PROC_CLASS1: P_WRI_REQ, P_NCBWR_REQ, P_INT_REQ, P_NCWR_REQ, P_NCRD_REQ, P_WRB_REQ.

These are the only allocations that DSC is guaranteed to support.

5.9.2 U2S

DSC only implements a single master class for U2S, and this is class 0. The U2S class 0 is different than class 0 for Processor.

1. U2S_CLASS0: P_RDO_REQ, P_RDD_REQ, P_NCBRD_REQ, P_WRI_REQ, P_WRB_REQ, P_NCBWR_REQ, P_INT_REQ, P_NCWR_REQ, P_NCRD_REQ,
2. U2S_CLASS1: 0

5.10 Presence Detection

DSC assumes that at least one processor and the U2S are always present. It does not perform any type of presence detection for these UPA clients, since a system without either one would not function. Open processor slots will give a Read Time-out error on the P_REPLY when they are unoccupied, because the P_REPLY is defined as being all ones when it is unconnected and that is the encoding for P_RTO.

For the UPA64S client, DSC will examine UPA_PREPLY3[1:0] as soon as it comes out of reset. If it sees that these lines are a nonzero value, then it concludes that there is no UPA64S client present. Any read transaction sent to this client, cached or non-cached, will result in a S_ERR being returned to the master, and the IADDR bit being set. A P_WRB_REQ issued to UPA64S will have undefined results regardless of whether UPA64S client is present or not. Therefore, it is required that the UPA64S client drive its PREPLY lines to an idle value during reset and after coming out of reset.

5.11 Data Stall

DSC has two data stall signals, UPA_DATASTALL0 (for the two processors) and UPA_DATASTALL1 (for U2S), which are used to regulate the flow of data when accessing a slower device (such as memory) or a device on a narrower bus. UPA_DATASTALL1 is now obsolete after dropping single bank mode support.

DataStall is only asserted for block transfers; it is never asserted for single cycle transfers.

The table below shows under what conditions DataStall is asserted.

Table 5-16 Data stall assertion.

Master	Operation	Slave	DataStall
CPU	READ	MEMORY	No
CPU	WRITE	MEMORY	No
U2S	READ	MEMORY	No
U2S	WRITE	MEMORY	No
CPU	READ	U2S	UPA_DATASTALL0

Master	Operation	Slave	DataStall
CPU	WRITE	U2S	No
U2S	READ	CPU	No
U2S	WRITE	CPU	Not Supported
CPU	READ	FFB	UPA_DATASTALL0
CPU	WRITE	FFB	No
U2S	READ	FFB	No
U2S	WRITE	FFB	No

5.12 Reserved P_Replys

If DSC receives a P_Reply from a slave that is documented in the “UPA Interconnect Architecture” as being “reserved,” then the results will be undefined.

5.13 Error Handling

Error handling is detailed in a Error Management section of the UPA Specification, an Error Handling chapter in the Sun-5 System Architecture Specification, and an Error Handling Chapter in the Fusion Desktop System Specification. Here are a list of important points:

1. No software programming error can hang the system.
2. All transactions complete end-to-end.
3. Transactions which are addressed to the slave address space of the same UPA port or directed to ones self in any way have the following effect:
    a. Reads to ones own slave address space return ambiguous data.
    b. Writes to ones own address space are dropped silently.
    c. Interrupts sent to ones self are dropped silently.
    d. Coherent read of data when it is already in ones own cache is illegal, and is terminated with S_ERR.

5.13.1 Fatal Errors

Fatal errors in the DSC can result from the following:

- Address parity error
- Master queue overflow
- DTAG parity error
- Coherence error
- P_FERR from a slave

Software can enable these Fatal errors to cause a system POR reset by setting the EN_FATAL bit in the SC_Control Register.

These are the errors that are detected by DSC:

1. Parity error on UPA address bus 0. This is logged in the IPRTY bit in the SC_Port_Status register of the corresponding master.
2. DSC receives a P_FERR from CPU or U2S. P_FERR can be issued at any time by a slave.
3. Parity error in the Dual tags.
4. Master queue overflow. *This is logged in MCxOF bit in the appropriate SC_Port_Status register.*
5. Coherence errors. When a processor No-acks (P_SNACK) a S_CPB_REQ, S_INV_REQ, S_CPI_REQ or S_CPD_REQ then a fatal error has occurred, since the caches are no longer coherent.
6. If an illegal cache state, such as P0=M and P1=S for the same block.

5.13.2 Nonfatal Errors

When nonfatal errors occur, the DSC will not set the FATAL error bit and will respond to the packet requests.

DSC has two bits in the PO_Status and P1_Status registers that are used to notify the system that certain nonfatal errors occurred. These are called IADDR and IPORT. Some nonfatal errors are not logged.

The DSC forwards S_ERR to the master when it receives a P_RERR or P_RERRC reply from a slave.

The DSC forwards S_RTO to the master when it receives a P_RTO from a slave.

The majority of the nonfatal DSC errors are noted in the DSC transaction table earlier in this chapter.

1. Access to a unimplemented address. This is logged in the IADDR bit in the SC_Port_Status register of the corresponding master. This includes accesses to the UPA64S client when no client is detected to be present.
2. Access to a port that is not present by a misdirected interrupt request. This is logged in the IPORT bit in the SC_Port_Status register of the corresponding master. This is not a fatal error. In DSC, only the processor is allowed to receive interrupt packets.
3. Coherent read requests to FFB will result in an S_ERR reply to the master. The UPA64S specification states that the SC could simply ignore them, but this S_ERR reply is more consistent with coherent reads to other slave addresses. The IADDR bit is set.
4. Non-cached accesses to memory. S_ERR reply. This is logged in the IADDR bit.

Coherence Controller 6

The UPA cache coherence protocol is point-to-point write-invalidate. The unit of cache coherence is a block size of 64 bytes. Coherent read / write transactions transfer data in 64-byte blocks only, using 4 quadwords. In order to avoid snoop interference with a processor's cache Dual set of tags (Dtags) is maintained by DSC. The Dtags contain the 4 MOSI cache states (E & M states are merged) see Table 6-2. The Dtags support direct mapped cache.

The primary function of the Coherence Controller (CC) is to keep data consistency between all the cached UPA ports. The CC maintains consistency using dual tags (or Dtags) that maintain a copy of each ports tags. The CC then performs snoops and updates on these Dtags to determine the proper consistency transactions.

The snoop requests will be handed to the CC by the Port Interface (PIF) block. The CC will first make sure that the incoming transaction has no cache index conflict with all active transactions. For coherent read requests, the Memory Controller (MC) block will kick off the memory access until the CC either informs the MC to abort the memory operation or realizes that memory is the proper data source. The PIF will only send the coherent write requests to the CC. When snooping result is known, the PIF will be informed as to whether to continue or cancel the write operation.

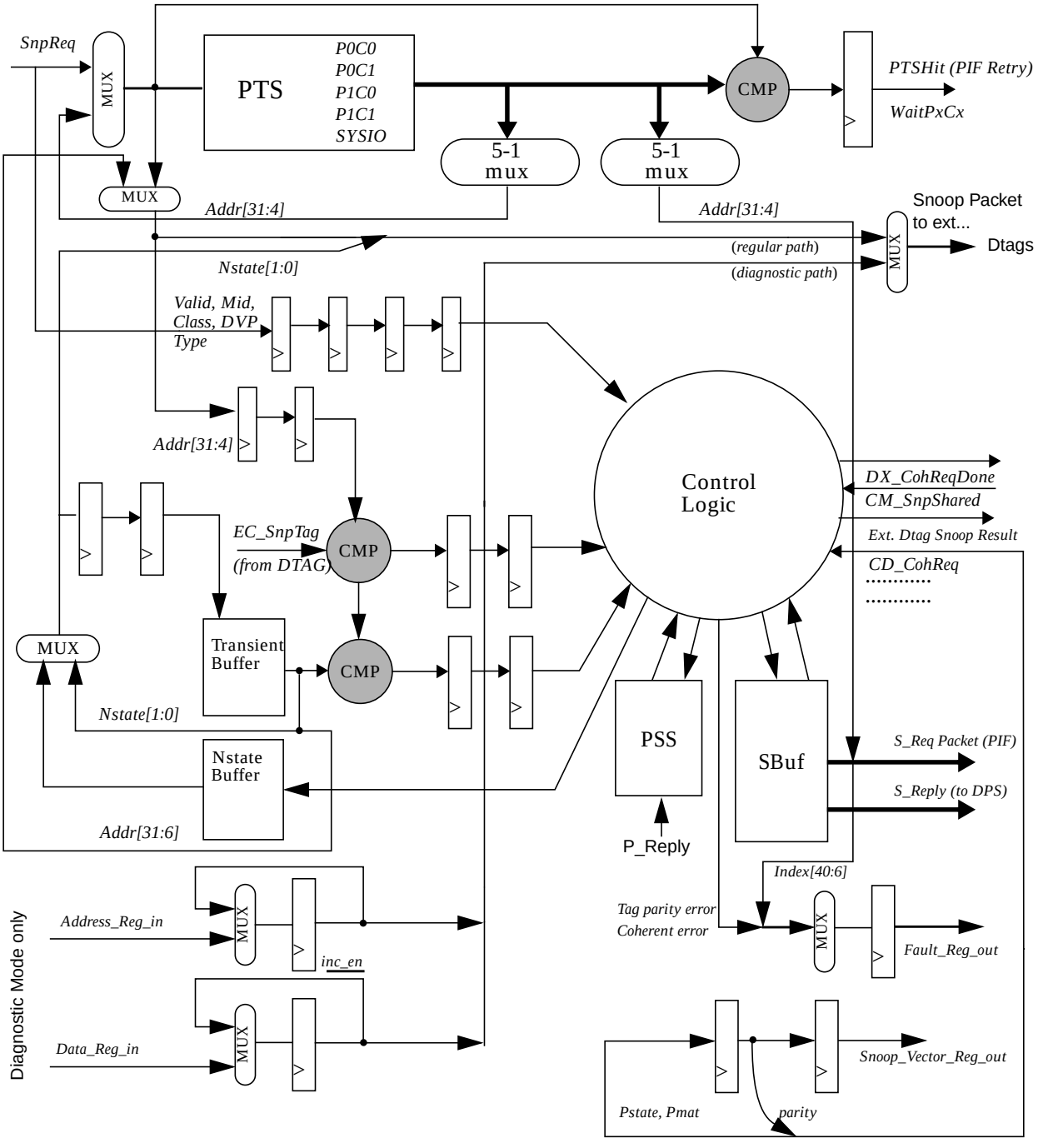
A single 18-bit wide SRAM contains the Dtags for both processors. Since the tag and status bits are 15 bits wide, only one Dtag can be accessed each cycle. It will take two consecutive clock cycles to snoop the Dtags of both processors. For timing of the Dtag accesses, please see the DSC Data Sheet, STP2202ABGA, document part no. 802-7328-02. The S_Requests will be sent to the PIF if any invalidation or copyback is needed. When the appropriate P_Reply has been received from the corresponding UPA port, the CC will send a request to the DPS for issuing the S_Reply and arrange the data transfer if necessary. The Dtags will be updated when the coherent transaction is finished.

6.1 CC Composition

Basically, the CC is composed of seven blocks. A block diagram is shown below.

1. Pending Transaction Scoreboard (PTS)
2. Transient Buffer(TB)
3. Nstate Buffer(NstB)
4. S_Request Buffer (SBuf) + Pending S_Request Scoreboard (PSS)
5. Fault_Register Block
6. Diagnostic Address & CC Registers
7. Control Logic
    a. TS & Snoop Control
    b. Transient Buffer Control
    c. Nstate Buffer Control
    d. S_Request Control
    e. Fault Register Control

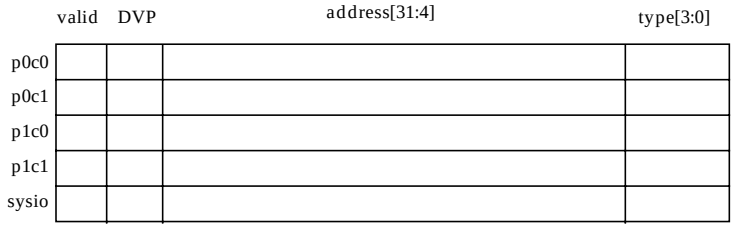
Figure 6-1 CC Block DiagramPending Transaction Buffer (PTS)



6.2 Pending Transaction Buffer

There are five entries in the PTS including one for each class in each processor and one entry for the U2S ASIC.

Figure 6-2 Format of Pending Transaction Buffer



Description of the entries is as follows:

valid: This entry of PTS is in use.

DVP: Dirty Victim Pending, set if the transaction displaces a dirty victim block in the cache.

address[31:4]: Address of the transaction used to updated the Dtags and to generate S_Req. It is also used to compare with address of incoming transactions to generate PTSHit.

type[3:0]: Transaction type[3:0]

6.3 Blocking Rules

Blocking rules are rules that regulate the flow of transactions and actions of the DSC to ensure cache consistency and simplify the design.

1. The incoming transaction will be blocked from activation if there is an index match with any active transaction.
2. The only exception is that the incoming coherent WRB transaction is activated if the blocking active READ transaction with DVP set is from the same UPA port as the WRB transaction.

6.4 Index Comparison

There are two rules regarding the Dtag index and tag.

1. For processors 0 & 1, the index mask is from CC_Proc_Reg.
2. The number of bits used for index compare will be the intersection of the size of incoming transaction cache index and the active transaction cache index. In the Pulsar system, the size of bits for comparison will be the index mask specified in the CC_Proc_Reg.

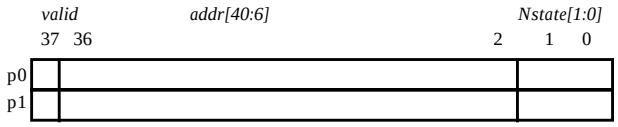
6.5 Transient Buffer (TB)

The cache coherence protocol restricts outstanding dirty victim writeback transactions to one per UPA port.

The Transient Buffer (TB) is used to store the address and state of dirty victim Read (Read with DVP set) if the associated line in the Dtag is still valid. TB will be updated at the same time as updating Dtag. This can prevent inconsistencies between Etag and Dtag. It will also be easier to debug in the bringup stage. See Table 6-1 on page 94 for detail of TB control.

There are two entries transient buffer, one for each processor.

Figure 6-3 Format of Transient Buffer



valid	The entry is in use.
address[31:6]	The physical address of the READ with DVP set.
Nstate[1:0]	The new state of master UPA Dtag. M/O/S/I.

6.6 CC actions

This section gives a table of the actions that the CC takes given various input controls.

6.6.1 Terminology

PTS	Pending Transaction Buffer
TB	Transient Buffer

Master UPA	The UPA which issues the current P_Request slave UPA. All the UPAs have cache except the master UPA
P_Req Type	Transaction type of the P_Request
Self TB Hit	The TB entry of master UPA port is valid, the tag is matched and Nst is not invalid
Y	Yes
N	No
X	Don't Care
I	Invalid state
S	Shared Clean State
O	Shared Modified State
M	Exclusive & potentially modified state
NC	No change

Table 6-1 CC Action table

Incoming Trans.	TB Valid?	Dtag (old)	Dtag (new)	WRB in Processing	RD_DVP in Processing	WRB done	RD_DVP done	Comments
RD with DVP	X	M/O	NC*	0	1	0	1	no WRB. Nstate -> TB Do nothing to Dtag.
WRB	no	M/O	I	1	0	1	0	no RD. write Dtag(new) to Dtag
WRB	yes	M/O	(TB)	1	0	1	0	no RD. (TB) -> Dtag

Table 6-1 CC Action table

Incoming Trans.	TB Valid?	Dtag (old)	Dtag (new)	WRB in Processing	RD_DVP in Processing	WRB done	RD_DVP done	Comments
RD with DVP followed by WRB	X	M/O	I/ (TB) see ->	1	1	0	1	RD finishes first. Nstate -> TB, Do nothing to Dtag. when WRB finishes, (TB) -> Dtag
				1	1	1	0	WRB finishes first, update Dtag with Dtag(new). when RD finishes, Nstate -> Dtag
				1	1	1	1	WRB & RD finish at same time. Nstate -> Dtag.

Where:

NC* No change to Dtag. The Nstate of the READ will be stored into the TB.

Dtag(old) The state of current cache line.

Dtag(new) The new state of the cache line.

“WRB in processing”, “RD_DVP in processing”, “WRB done” and “RD_DVP done” are four control signals to control the timing to update TB/Dtag and destination of new state.The four control signals are updated by the following conditions:

To set:

1. “WRB in processing” and “RD_DVP in processing” will be set at the same clock cycle when the snoop control logic is checking the snoop results.
2. “WRB done” and “RD_DVP done” will be set when respective transaction is finished.

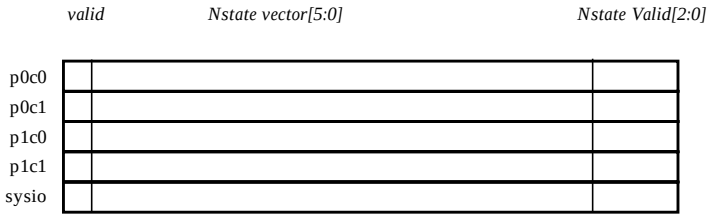
To clear:

1. These four signals will be cleared at the same time as the last finished transaction is done.

6.7 Nstate Buffer (NBuf)

The Nstate Buffer(NBuf) is used to store the new state information of the active transactions. There is one entry for each class. The address of corresponding Nstate is stored in the PTS. There are five entries for the NBuf, one for each processor class and one for the U2S ASIC.

Figure 6-4 Format of the Nstate Buffer



Where:

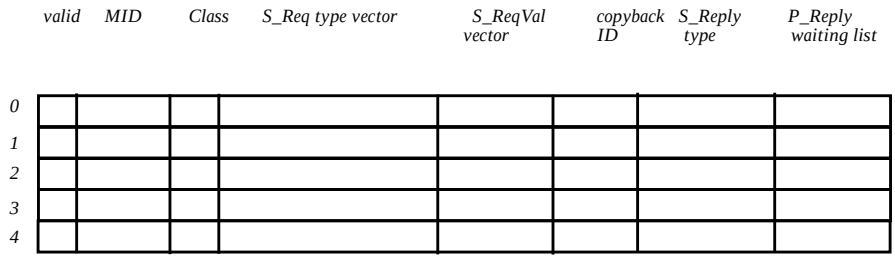
valid	The entry is valid.
Nstate Vector[3:0]	Two bits for each UPA port. Six bits total to store the new state of this cache line. M/O/S/I.
Nstate Valid[2:0]	One bit for each UPA port to indicate if the UPA port needs to be updated with the new state.

Note: The address of the Nstate will be from PTS.

6.8 S_Request Buffer (SBuf)

The S_Request Buffer (SBuf) is used to store all the information needed to generate S_Request and CP_CohReq. There are five entries in the SBuf.

Figure 6-5 S_Request Buffer (SBuf)



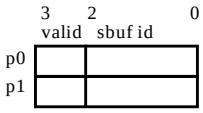
Where:

valid	The entry is in use.
MID[1:0]	ID of master UPA.
Class	The class of master UPA
S_Request vector[3:0]	The S_Requests will be sent to all UPA ports. The entry will be clear after the S_Request has been sent to appropriate UPA port
S_ReqVal vector[1:0]	The qualifiers for S_Request vector. [1] for Processor 1, [0] for Processor 0
Copyback source ID	The ID of the UPA port that SC needs to copy data from. This ID is used to generate CD_CohReq.
S_Reply type[3:0]	The transaction type of S_Reply for this snoop request. This is also used to generate CD_CohReq.
P_Reply waiting list[1:0]	This is a list of P_Replys that current snoop request should receive. Initially, this list is identically to S_Request vector. The P_Replys from UPA ports will clear the entry. When S_Request vector and P_Reply waiting list both are empty, it is time to issue CD_CohReq to assert S_Reply.

6.9 Pending S_Request Scoreboard (PSS)

The Pending S_Request Scoreboard (PSS) keeps track of all UPA slave ports. When a S_Request is sent out, the corresponding PSS entry will be validated and the SBuf entry number in the sbuf id field will be updated. There are two entries in this scoreboard, one for each processor.

Figure 6-6 Pending S_Request Scoreboard format

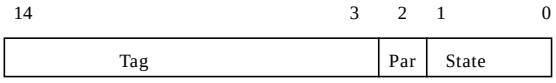


valid	The entry is in use.
SBuf ID[3:0]	The SBUF entry ID will be stored in this field if a S_Request has been sent to that UPA port for copyback/invalidate. This field will be used to identify the SBuf entry to clear when the P_Replys are received.

6.10 Dtags

There are 3 supported external Dtag sizes. CC maintains coherency for 512K, 1M or 2M byte sizes of DTAGs and Processor's E-$. The Dtag RAM is implemented by Synchronous SRAM with a 64K x18 configuration. The following figure shows the how the external Dtag bits are formatted.

Figure 6-7 External Dtag pin format



DTAG states are encoded in Table 6-2

Table 6-2 Encoding of Cache Coherent states.

SNPSTATE(1:0)	Symbol	State Name	Note
00	**I**	Invalid	mirror processor's Etag state
01	**S**	Shared Clean	mirror processor's Etag state
10	**O**	Shared Modified	mirror processor's Etag state
11	**M**	Exclusive & Potentially Modified	**E & M** states are merged

Memory System 7

7.1 Introduction

This chapter describes the structure and operation of the Ultra 2 memory system that is under control of DSC.

7.2 SIMMs

Ultra 2 uses the modified version of the Campus II DSIMM which uses 60 ns DRAMs. The controller is able to be programmed to use the older version of the Campus II DSIMM which used 80 ns parts and the EDO (Enhanced DRAM Operation) 70 ns DRAMs which the industry is currently discussing as a standard. The specific timing numbers for the EDO DRAMs are not included in this specification however a CSR set can be developed for these parts. The table below shows the SIMM sizes which are supported by DSC.

Table 7-1 SIMMs Supported by DSC.

SIMM Size	Base Device	Number of Devices	Note
16 MB	4 Mb (1M x 4)	36	
32 MB	16 Mb (2M x 8)	18	DSC will only support parts with 2K rows and 1K columns. There are currently no plans to build this SIMM.
64 MB	16 Mb (4M x 4)	36	DSC will only support 4k refresh parts (4k rows, 1k columns).
128 MB	64 Mb (8M x 8)	18	DSC will only support 8k refresh parts (8k rows, 1k columns); 5.0 V SIMM.

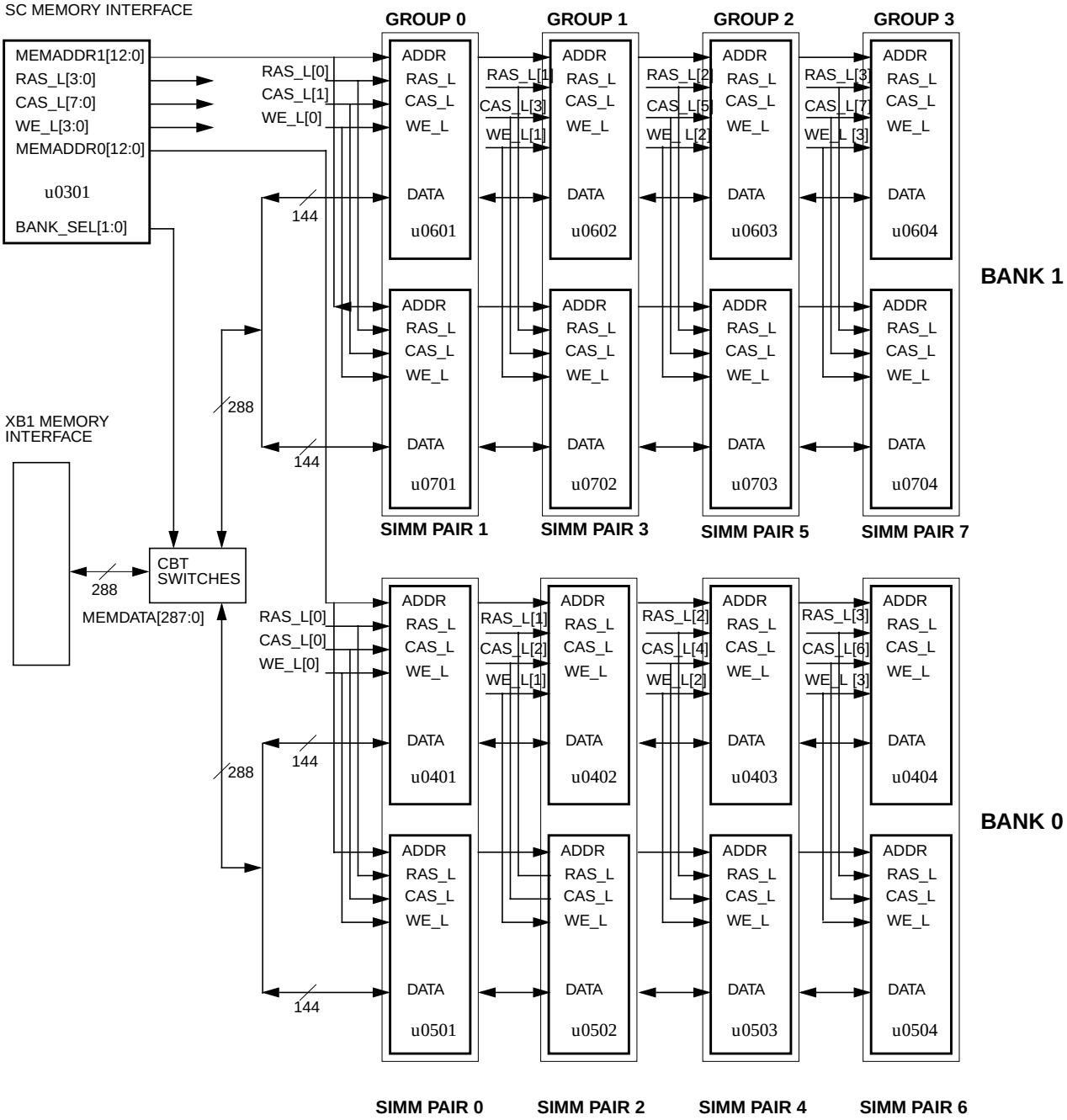
The Ultra 2 memory system is designed as a +5V memory system. The XB1 has been designed as a 3.3V part which has 5V tolerant inputs. The CBT switches have a 5v supply voltage. They will also be built such that the 3.3V drive from the DSC control logic (BankSel) will be amplified to ensure that there is a negligible voltage drop through the switches and we incur no threshold drop. This will allow us to migrate seamlessly from 5V memory SIMMs to 3.3V memory SIMMs. DSC's Memory Controller (MC) can handle up to sixteen SIMMs. If 16 Mb memory technology is used, then a maximum of 1 GB is possible. If 64 Mb memory technology is used, then a maximum of 2 GB is possible.

For 64 Mb parts, DSC will only support 8M x 8 with a 8k x 1k organization. It will not support 16M x 4 parts.

The minimum memory configuration is 64 MB, or four 16 MB SIMMs as a group.

The diagram below shows the interconnection between DSC and the memory system.

Figure 7-1 Memory System Interconnection



Note that the LSB of the SIMM Pair number is the Bank number.

The memory data bus to the XB1 is 288 bits wide: 256 bits of data and 32 bits of ECC. Since each SIMM supplies only 144 bits of data, two SIMMs are required to fill the data bus. Further, the Ultra 2 memory system differs from the Ultra 1memory system in that a second bank of SIMMs has been added. The two banks of memory are connected to the XB1 via CBT switches which are simple transmission gates. The DSC controls which bank is connected to the XB1 using the BANK_SEL signals.

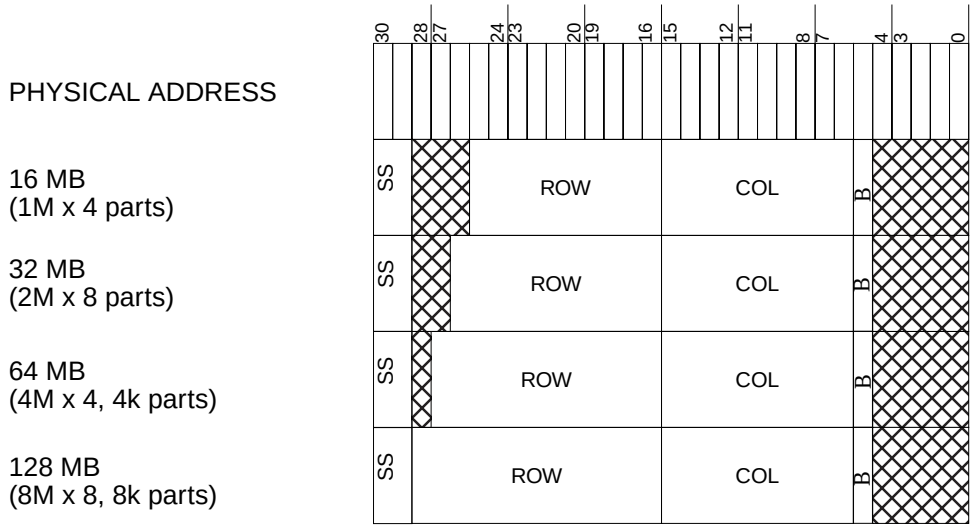
Both SIMM banks are always accessed therefore SIMMs must be populated by groups which consist of two pairs (4 SIMMs). When populating memory, it is important to ensure that both SIMM pairs are filled, and that all 4 SIMMs are of the same type. Failure to follow this rule will result in an inoperative SIMM group. For example, using two pair of 16MB SIMMs obtains the smallest memory configuration of 64 MB.

The MEMADDR1[12:0] and MEMADDR0[12:0] signals, all have 8 loads. Each SIMM pair has a unique CAS_L. Each Group (two pair) has unique RAS_L and WE_L signals. Each SIMM actually has two loads on RAS_L, two loads on CAS_L and one load on WE_L. Therefore, there are eight loads on each RAS_L, four loads on each CAS_L, and four loads on each WE_L All address, RAS_L, CAS_L, and WE_L signals are buffered by the SIMM before being distributed to the DRAM array.

7.3 Addressing

The figure below shows the addressing scheme used in the DSC MC.

Figure 7-2 Addressing in a dual bank system



SS= SIMM SELECT - also = the Group number
B= The Bank number.

In this scheme, PA[30:29] is used as a Group select. The way that PA[30:29] maps into RAS_L, CAS_L, and WE_L is shown in the table 7-2 below.

Table 7-2 PA[30:29] to RAS_L, CAS_L, and WE_L Mapping

PA[30:29]	RAS_L, CAS_L, and WE_L Asserted
00	RAS_L[0], CAS_L[0], CAS_L[1], and WE_L[0]
01	RAS_L[1], CAS_L[2], CAS_L[3], and WE_L[1]
10	RAS_L[2], CAS_L[4], CAS_L[5], and WE_L[2]
11	RAS_L[3], CAS_L[6], CAS_L[7], and WE_L[3]

In a Ultra 2 system, PA[28:16] map into the row address (MEMADDR[12:0]) and PA[15:6] map into the column address (MEMADDR[9:0]).

Bits PA[5:4] always select the critical word first. The Datapath Scheduler selects odd or even alignment based on PA[4], but the memory system reorders the two double words in a requested quad-word based on PA[5]. In an Ultra 2 system, PA[5] will cause Bank 1 to output first when set, otherwise Bank 0 outputs first.

SCMC will only support 16 Mb parts that have a 4k organization (4k rows, 1k columns). 2k parts (2k rows, 2k columns) are not supported by this scheme.

SCMC will only support 64 Mb parts that have a 8k organization (8k rows, 1k columns).

For diagnostics, memory accesses are tagged with the master ID and class bit of the initiator. The two bit master ID is carried on MEMADDR[12:11] and the class bit is carried on MEMADDR[10] during the column address cycles. The only legal tags are:

1. 000: access from CPU #0 master class 0
2. 001: access from CPU #0 master class 1
3. 010: access from CPU #1 master class 0
4. 011: access from CPU #1 master class 1
5. 110: access from U2S master class 0

Any other tag should be considered an error. These tag bits are unused by the SIMM during the presented cycle and have no affect on the memory system.

DSC will not detect accesses which go to an out-of-range address or to a nonexistent SIMM. An access to memory will either wrap back to a SIMM which does exist, or return garbage data.

Table 7-3 below shows the same information as Figure 7-2, but in a different form.

Table 7-3 Memory Address Map

SIMM Number	SIMM Type	Address Range (PA[30:0])
0	16 MB	0x0000_0000 to 0x03ff_ffff (first double word)
0	32 MB	0x0000_0000 to 0x07ff_ffff (first double word)
0	64 MB	0x0000_0000 to 0x0fff_ffff (first double word)
0	128 MB	0x0000_0000 to 0x1fff_ffff (first double word)
1	16 MB	0x0000_0000 to 0x03ff_ffff (second double word)
1	32 MB	0x0000_0000 to 0x07ff_ffff (second double word)
1	64 MB	0x0000_0000 to 0x0fff_ffff (second double word)
1	128 MB	0x0000_0000 to 0x1fff_ffff (second double word)
2	16 MB	0x2000_0000 to 0x23ff_ffff (first double word)
2	32 MB	0x2000_0000 to 0x27ff_ffff (first double word)
2	64 MB	0x2000_0000 to 0x2fff_ffff (first double word)
2	128 MB	0x2000_0000 to 0x3fff_ffff (first double word)
3	16 MB	0x2000_0000 to 0x23ff_ffff (second double word)
3	32 MB	0x2000_0000 to 0x27ff_ffff (second double word)
3	64 MB	0x2000_0000 to 0x2fff_ffff (second double word)
3	128 MB	0x2000_0000 to 0x3fff_ffff (second double word)
4	16 MB	0x4000_0000 to 0x43ff_ffff (first double word)
4	32 MB	0x4000_0000 to 0x47ff_ffff (first double word)
4	64 MB	0x4000_0000 to 0x4fff_ffff (first double word)
4	128 MB	0x4000_0000 to 0x5fff_ffff (first double word)
5	16 MB	0x4000_0000 to 0x43ff_ffff (second double word)
5	32 MB	0x4000_0000 to 0x47ff_ffff (second double word)
5	64 MB	0x4000_0000 to 0x4fff_ffff (second double word)
5	128 MB	0x4000_0000 to 0x5fff_ffff (second double word)
6	16 MB	0x6000_0000 to 0x63ff_ffff (first double word)
6	32 MB	0x6000_0000 to 0x67ff_ffff (first double word)

SIMM Number	SIMM Type	Address Range (PA[30:0])
6	64 MB	0x6000_0000 to 0x6fff_ffff (first double word)
6	128 MB	0x6000_0000 to 0x7fff_ffff (first double word)
7	16 MB	0x6000_0000 to 0x63ff_ffff (second double word)
7	32 MB	0x6000_0000 to 0x67ff_ffff (second double word)
7	64 MB	0x6000_0000 to 0x6fff_ffff (second double word)
7	128 MB	0x6000_0000 to 0x7fff_ffff (second double word)

7.4 Refresh

CAS before RAS refresh is used.

DSC uses a distributed refresh strategy where MC cycles through the SIMMs and issues refresh operations periodically. Two pair of SIMMs are refreshed at a time, so in a full memory system there are four pairs of SIMM pairs to cycle through.

MC uses the contents of the SIMM Present vector in Mem_Cntl0 to determine which SIMMs to refresh. MC will not refresh a SIMM group unless its corresponding bit in the SIMM Present vector is set.

MC uses the contents of the RefInterval field in Mem_Cntl0 to determine the interval between refreshes. Each LSB of RefInterval corresponds to eight system clocks. The timer will periodically signal the need for a refresh. When the timer issues a refresh request, it will automatically reset itself and begin counting again, even before the refresh request has been honored. This ensures refreshes do not accumulate any additional delay due to the fact the refreshes cannot always be serviced immediately.

MC uses a very simple algorithm to arbitrate between refreshes and memory accesses. If a memory read or write is going on, then MC will wait for the read or write to finish before issuing the refresh request. MC will never interrupt a read or write to issue a refresh. If a read or write reaches MC the same clock cycle as a refresh request, then the request will be honored immediately; the MP_QFull signal will remain high throughout the request and subsequent refresh to hold up any further requests; the subsequent refresh will then occur; and finally, the MP_QFull signal will drop allowing other requests from the PIF.

After power-on or any reset, refresh is disabled. Setting the RefEnable bit in Mem_Cntl0 will allow refreshes to begin. RefInterval and SIMMPresent should be set to the proper values before setting the RefEnable bit.

If a “soft” reset occurs (this happens when DSC detects a fatal error and resets the system) refresh will be turned off and the contents of memory will be lost.

7.5 Memory Controller (MC)

7.5.1 Overview and the Generic Waveform Generator

Several factors and requirements have come together in Ultra 2 which complicate the memory system timing considerably. First, we have a range of system clock timings which need to be supported. This is the same requirement that exists in Ultra 1.

Second, we would like the highest performance in terms of memory bandwidth for our system. In order to do this, we would like to start requests to different SIMM groups as early as possible. (i.e. - If we access a different group from the one we are currently accessing or refreshing, then we can start the RAS/CAS cycle early. We do not have to wait for the previous RAS cycle to finish, or for the RAS precharge to occur. We are only limited by the DRAM output buffer turn-off times, since all the Groups share the same data bus. At the present time, the fastest back to back latency to different SIMM Groups which the internal Finite State Machines will allow is 6 clock cycles, and this is *the* limiting factor in the MC. Actually achieving 6 will require an improvement in either t_{off}, the DRAM buffer turn-off time, or t_{cac}, the DRAM access time from CAS. Seven or eight clock cycles is a more realistic estimate at this time.) Starting different SIMM Group accesses early requires at least one set of waveform engines per group. Allowing concurrent refreshes to different SIMMs on a per SIMM pair basis would require one set of waveform engines per SIMM pair for the RAS, CAS, and WE lines therefore this is not supported. Refresh is done on SIMM pairs in each group.

Lastly, Ultra 2 was initially required to support a two SIMM minimum memory configuration. This forced the memory controller to not only have the pulse widths programmable, but to also have the number of pulses themselves programmable, if we were to achieve the highest performance. This Single Bank Mode requirement severely impacted the design itself. Initially it was thought extremely important to have this mode so that we could drop from our current 64 MB minimum memory system to a 32 MB entry level system. Near the very end of the DSC design cycle it was decided to drop this requirement and force users to install simms in groups of four. This was done for several reasons. First, it did indeed impact the cycle time of the design as was initially predicted. Removing it eased meeting our timing goal, but left many vestiges that could not be removed without redesigning the entire block. It eliminated the need to test Single Bank Mode. It eliminated the need to check and maintain two sets of CSR values. Lastly, it eliminated confusion in populating SIMMs. Overall, we are still better off without that mode except that there are now a lot of little details which remain which no longer seem

to make sense (i.e. - why RAS and CAS would need five phases for example... there are many others...) The author has tried to eliminate all of the Single Bank references, but in reading this and other documents, you still might find a reference or two.

The above three requirements resulted in an MC which is extremely programmable, but also rather large.

To achieve the above goals, a generic waveform generator has been created which is the basis for generating RAS, CAS, WE, MRBCtrl, MWBCtrl, BankSel, and RowGen which are internal signals used to select the address output to the DRAMs themselves.

Since in single bank mode, the MC must generate two low pulses for CAS, the Wavegen Generator is broken up into five phases. For each phase, the number of clock cycles for that phase is programmable from 1 through 8, and the phase level (high or low) is programmable. The initial value is also programmable. These numbers are taken from the CSR's in the MC (described in the next section currently, but will be moved into the Fusion Desktop System Specification in the next release) depending on whether the request is a read, write or refresh and involving a Processor or the U2S. This allows us to generate almost any waveform from 5 cycles to 40 cycles long.

A Phase Control word is a 16 bit quantity in the following form:

Table 7-4 Phase Control Word

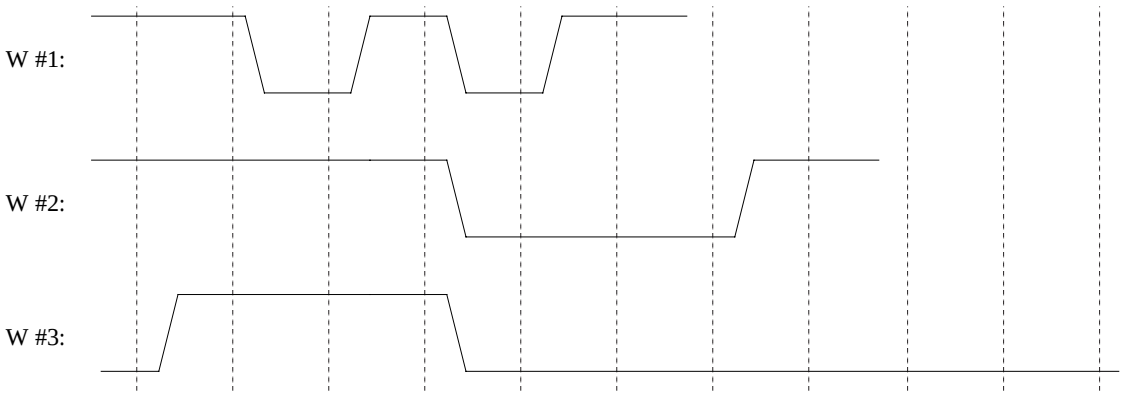
Bits	Description
15:13	Number of cycles in Phase 5. (000 = 8)
12:10	Number of cycles in Phase 4. (000 = 8)
9:7	Number of cycles in Phase 3. (000 = 8)
6:4	Number of cycles in Phase 2. (000 = 8)
3:1	Number of cycles in Phase 1. (000 = 8)
0	Initial value.

The Phase Level word is a 5 bit quantity in the following form:

Table 7-5 Phase Level Word

Bit	Description
4	Level in Phase 5.
3	Level in Phase 4.
2	Level in Phase 3.
1	Level in Phase 2.
0	Level in Phase 1.

Now, if we wish to program the following waveforms:



We can use the following values:

Table 7-6 Waveform example

Wave-form	Phase Control Word	Phase Level Word
W#1	001 001 001 001 001 1 = 0x2493	10101 = 0x15
W#2	001 010 001 010 001 1 = 0x28A3	10011 = 0x13
W#3	010 010 011 010 001 0 = 0x49A2	00011 = 0x03

The above explanation of the waveform generator internals was provided in an attempt to make the CSRs slightly more sensible. This document also contains the released timing for a 60 ns DSIMM/12 ns System clock/Dual Bank configuration. Note that the Power On/Hard Reset default is not functional.

7.5.2 Top Level Block Diagram

Figure 7.3 shows a simplified block diagram of the MC. The major blocks include mc_fsm, mc_simm_eng, mc_addrgen, mc_refreng, mc_csr, mc_stall_rb and mc_stall_wb.

7.5.2.1 mc_fsm

This is the main FSM for the MC. This machine looks at incoming requests and kicks off the appropriate SIMM engines. It also accepts the next request before the current one is finished. The FSM uses the SIMMBusy and SIMMBusyDiff signals to determine SIMM availability. This signals are a product of the SIMM Busy counters which are programmed via the CSRs.

Refresh is also controlled in mc_fsm. When a refresh is requested, it raises MP_QFull and then waits an additional cycle for incoming PM_ReqVal signals before proceeding. The PIF will not issue PM_ReqVal when it receives MP_QFull. MP_QFull is held active during the refresh period to service the refresh without interruption.

7.5.2.2 mc_simm_eng

The mc_simm_eng block generates most of the waveforms for the MC. All the control signal waveform generators are located in this block. Also, all the Start/Kill signals and hold checks occur here as well as the stretch timers. Start signals from the mc_fsm start off all the waveform generators. The Kill signals are used to stop the waveform generation in the event that the CC block has a cache index hit and retries the transaction. The stretch counters pick when during a cycle it is important for the XB1 buffers to be either filled (in the case of a write) or emptied (in the case of read following a read). When the timers expire, the RBBusy and WBBusy signals from mc_stall_rb and mc_stall_wb are checked and the operation is stretched if the data for a write is not present or if the data from the previous read has not yet been consumed. Figures 7.4 and 7.5 show a simplified mc_simm_eng.

7.5.2.3 mc_addrgen

The main function of this block is to buffer up the incoming address (or flow through the first row address if the MC is currently idle) and reformat the addresses appropriately for the row and column addresses to the DRAMs. Two buffers are provided for this which are called the Ping and Pong address buffers. The ping-pong FSM is also instantiated in this block. It controls which set of buffers get loaded up and set is currently driving. The internal signal Row coming from mc_simm_eng determine whether the block should be driving row or column address.

7.5.2.4 mc_refrgen

This block is based off the DSC MC's refresh counter code. It generates refresh requests that are received by the mc_fsm for SIMMs needing refresh.

7.5.2.5 mc_csr

The mc_csr block contains all the CSR registers and the control circuitry to read and write the registers from the EBus. The description of these CSRs can be found later in this chapter.

7.5.2.6 mc_stall_rb and mc_stall_wb

These two blocks generate the RB_Busy and WB_Busy signals respectively. These signals are used to help track the state of the XB1 buffers. RB_Busy inputs include DM_RBAck, MRBCtrl(internal version), and cancel which is used when a transaction is cancelled. In addition, to track the particular transaction, the MID bits are also tracked. These are used to determine what transaction to cancel. The WB_Busy block is similar in nature except that uses MWBCtrl and DM_WBACk and has no cancel or MID bits.

Figure 7-3 Block Diagram of the Memory Controller

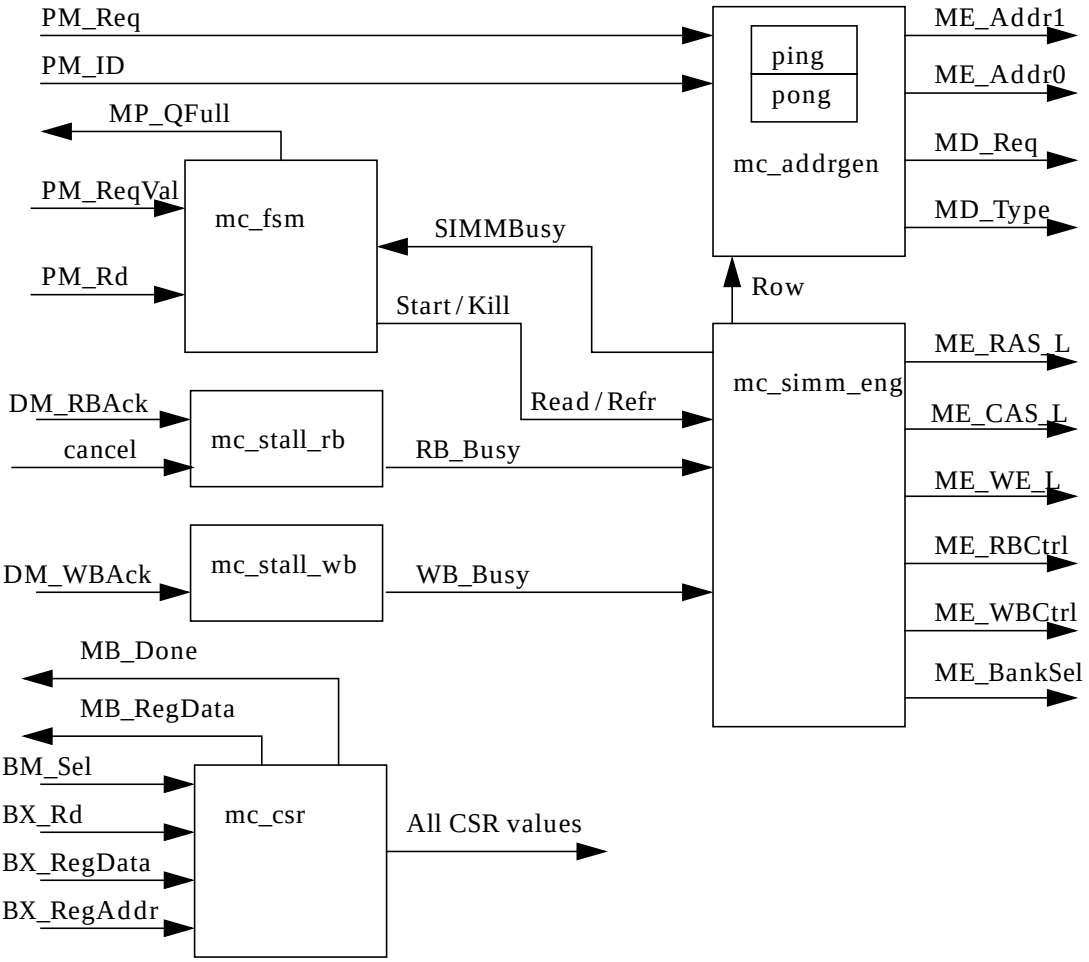


Figure 7-4 mc_simm_eng

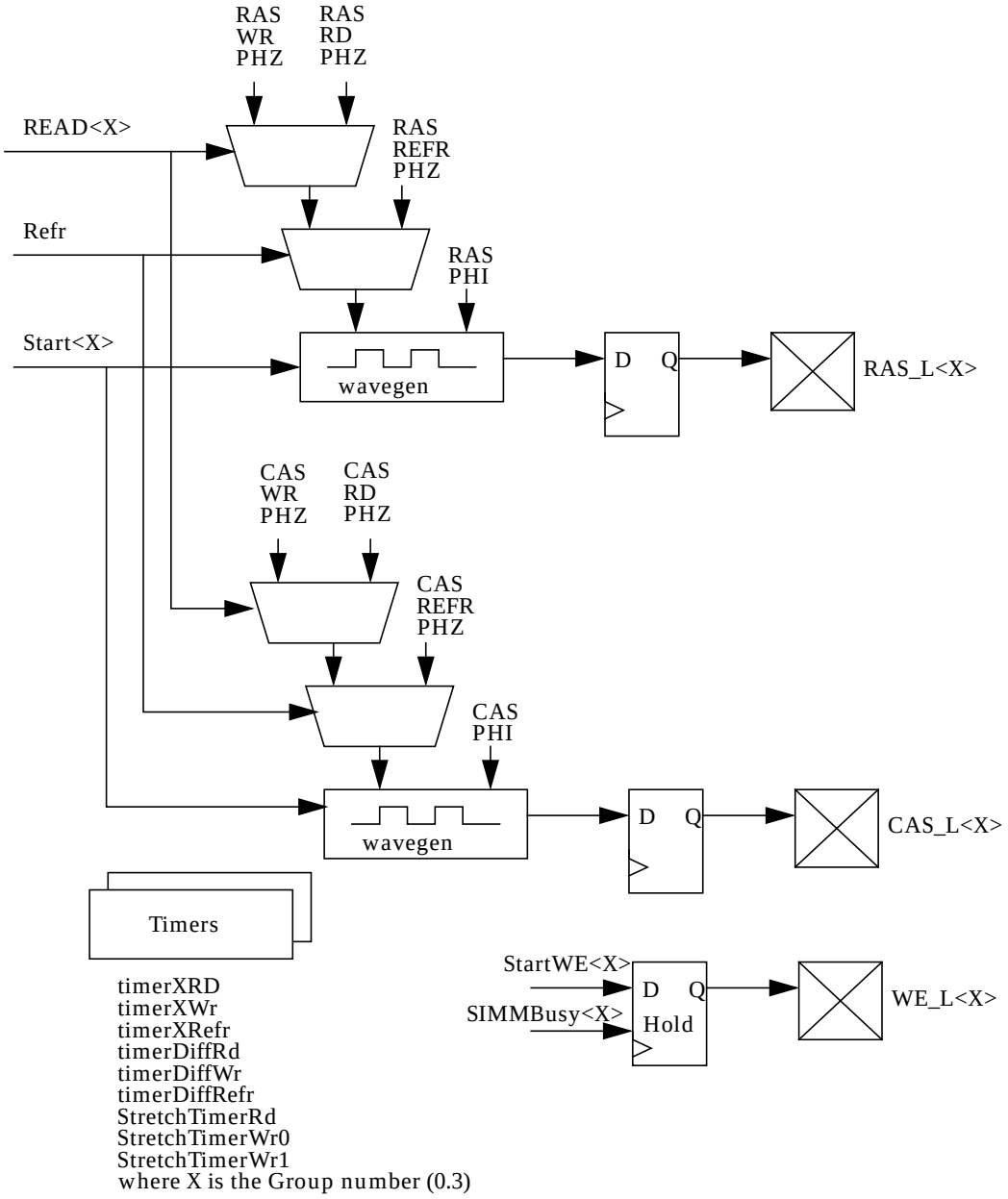
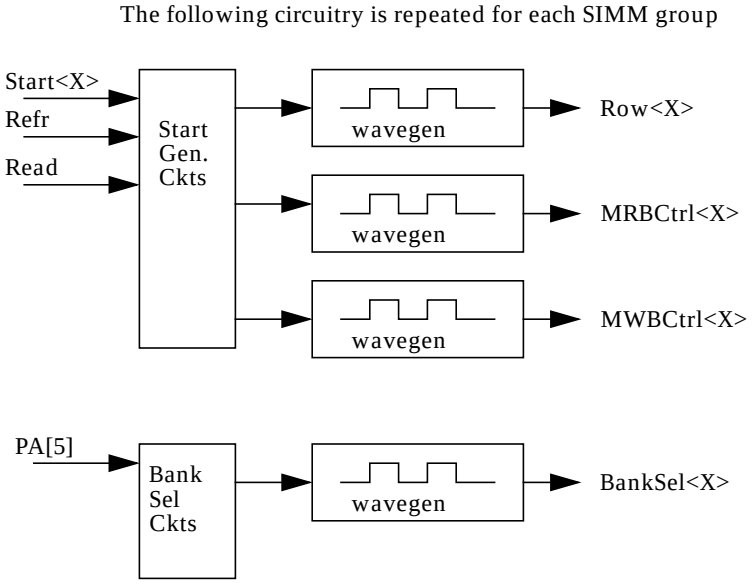


Figure 7-5 mc_simm_eng, continued



7.6 MC CSRs

The Memory Controller's command and status registers(CSR) provide the ability to program the memory controller waveform generators as well as check DSC MC status.

7.6.1 CSR address map

Here is an address map of the MC registers. In the current revision of the ASIC, the Column_Control register was replaced by a reserved register. Thus, the original address mapping was preserved. Column_Control was obsoleted by the elimination of Single Bank mode. Please note that the power-on default values will not work in the system. All registers must be reprogrammed before the DRAM memory system is accessed the first time.

Table 7-7 CSR map

SC Address	Function	Name in Previous Revision	Name in Current Revision	Power Up Value
80	Mem_Control0	MC Register 0	MC Register 0	00c1 0f14
84	RAS_Control	MC Register 1	MC Register 1	25c7 4ca5
88	CAS_RD_Control	MC Register 2	MC Register 2	0000 2d95
8c	BankSel_Control	MC Register 3	MC Register 3	2926 249a
90	XB1_Buffer_Control	MC Register 4	MC Register 4	2896 249a
94	CAS_WR_Control	MC Register 5	MC Register 5	24bb 24b7
98	Phase_Level_Control	MC Register 6	MC Register 6	14ac 0278
9c	SIMM_Busy_Rd_Control	MC Register 7	MC Register 7	02b5 9588
a0	Count_Control	MC Register 8	MC Register 8	0601 8d24
a4	Refresh_Control	MC Register 9	MC Register 9	2893 2537
a8	Row_Control	MC Register A	MC Register A	25b5 2493
ac	Column_Control(Rev. 0)	MC Register B	Reserved	0000 0000
b0	SIMM_Busy_Wr_Control	MC Register C	MC Register B	0284 9148
b4	SIMM_Busy_Refr_Control	MC Register D	MC Register C	02b0 8548

Important Note:
The following register descriptions (from Tables 7-8 to 7-22) contain CSR values for three frequency ranges. These numbers should be considered generic specifications and should not be used for actual MC programming. The three sets of CSRs provided in Section 7.6.10 are the only proven timings the DSC will support in the Ultra 2 system. Other timings are possible but would be very difficult to generate and test.

7.6.2 Mem_Control0 Register

The Mem_Control0 Register is also referred to as Memory Control Register 00 at address 80. It contains a number of bits which control the Refresh Controller and configuration in the Memory Controller and several other miscellaneous bits found only in the DSC MC.

Table 7-8 Mem_Control0 Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5MHz	> 83.5 MHz	Description	Type
RefEnable	31	0	0	0	Refresh enable	RW
Reserved	30:29	0x0	0x0	0x0	Reserved- read as 0	R0
FSMError	28	0	0	0	MC FSM Error	RW1C
PingPongError	27	0	0	0	Ping Pong Buffer Error	RW1C
DPSError	26	0	0	0	Data path scheduler Error	RW1C
MCError	25	0	0	0	Memory Controller Error	RW1C
MissedRefrError	24	0	0	0	Missed Refresh Error	RW1C
RefrPhiMC	23	1	1	1	RAS Phi 0 Magic Cookie for refresh cycles	RW
RasWrPhiMC	22	1	1	1	RAS Phi 0 Magic Cookie for writes	RW
Reserved	21	0	0	0	Reserved- read as 0	R0
StretchWr0<4:0>	20:16	0x01	0x01	0x01	Stretch count value for first write	RW
Reserved	15:11	0x0	0x0	0x0	Reserved- read as 0	R0
SIMMPresent<3:0>	11:8	0xF*	0xF*	0xF*	Determines which SIMMs to refresh	RW
RefInterval<7:0>	7:0	0x14*	0x14*	0x14*	Interval between refreshes. Each encoding is 8 system clocks	RW

** these values are reprogrammed based on the installed SIMM Population.*

RefEnable

Main memory is composed of dynamic RAMS, which require periodic “refreshing” in order to maintain the contents of the memory cells. The RefEnable bit is used to enable refresh of main memory.

```
0 = disable refresh; 1 = enable refresh
```

Disabling refresh causes the RefInterval timer to stop counting, hence the refresh requests to the main FSM stop occurring. Only POR will clear refresh_enable.

FSMError

This bit signifies that the Main Finite State Machine in the MC has somehow reached an illegal state. Results in a system reset.

PingPongError

This bit signifies that a serious error has occurred in the MC's Ping-Pong buffer management. Results in a system reset.

DPSError

This bit signifies that the DPS has somehow made an error in it's handshake with the MC regarding the Read or Write buffers in the XB1 (i.e. - either it has acknowledged before the XB1 Read buffer was written or it acknowledged twice for one Read or it has written a second time before the Write buffer has been emptied.) Results in a system reset.

MCError

This bit signifies that the MC has somehow made an error in it's handshake with the DPS regarding the Read or Write buffers in the XB1 (i.e. - either it has overwritten the XB1 Read buffer or has prematurely read the XB1 Write buffer or the XB1 buffer FSMs have reached an illegal state.) Results in a system reset.

MissedRefrError

This bit signifies that a refresh request has not been honored and that memory could be corrupted. This could occur for example if a coherent write is issued and for some reason, the handshake that the MC is waiting for signifying that the XB1 Write Buffer has been filled, never occurs. The MC will be stuck waiting and thus miss its refresh cycles. A separate watchdog triggers this error. Results in a system reset.

PingPongError, DPSError, MCError, and MissedRefrError are all registers which once set, remain set until either a system reset or they are individually reset by writing a one into them. In any case, they are here mainly for debug and should never be set during normal operation. Any one of these will have the MC request that the E-Bus Interface signal a fatal error and reset the system.

RefrPhiMC - this bit must always be set to 1

This bit will take the place of the RAS_Phi<0> bit (see Section 7.6.5) in the case of a Refresh operation. This was necessary so that on reads we have the ability to drop RAS immediately. Not having this bit would have precluded our ability to do CAS before RAS Refreshes.

RasWrPhiMC - this bit must always be set to 1

This bit will take the place of the RAS_Phi<0> bit (see Section 7.6.5) in the case of a Write operation. This was necessary so that on reads we have the ability to drop RAS immediately, whereas, on writes, we may wish to wait later in the cycle.

StretchWR0<4:0>

This register is programmed with the value of the first stretch point in a coherent write cycle.

SIMMPresent<3:0>

This field is used to indicate the presence/absence of SIMMS so that the SC does not spend more time than is necessary refreshing unpopulated SIMMs. A zero in this field indicates the corresponding group of SIMMs are not present, a 1 indicates presence. These bits must be set by software after probing. The SIMM pair to bit correspondence is given in Table 7-9.

Table 7-9 SIMMPresent Encoding

SIMMPresent<i>	SIMM Pair Slot
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7

RefInterval<7:0>

RefInterval specifies the interval time between refreshes, in quanta of 8 system cycles. Software should program RefInterval according to the formula or tables below based on the UPA speed.

$$\text{Ref Value} = \frac{\text{Refresh Period}}{(\text{No. of Rows X UPA Clock Period X No. of Groups})}$$

DSC formula

$$\text{Ref Value} = \frac{1.953125}{\text{UPA Clock Period X No. of SIMM Groups}}$$

If the formulas are used to calculate the refresh rate programming code then always round up to the next highest hex number.

The following table defines the lower 8 bits (7:0) of the DSC memory control register (0) for the various frequencies the Ultra 2 system is required to run. If the table is used instead of the formula then for any frequency programmed between these steps always round the frequency down. i.e. 56.6 MHz is programmed, use the values for 55.5 MHz (18 NSec).

Table 7-10 Refresh Rate Programming Values: All values in decimal

Refresh Value Table	4 SIMMs	8 SIMMs	12 SIMMs	16 SIMMs
50.0 MHz or 20 NSec	98	49	33	25
52.6 MHz or 19 NSec	103	52	35	26
55.5 MHz or 18 NSec	109	55	37	28
58.8 MHz or 17 NSec	115	58	39	29
62.5 MHz or 16 NSec	123	62	41	31
66.6 MHz or 15 NSec	131	66	44	32
71.4 MHz or 14 NSec	140	70	47	35
76.9 MHz or 13 NSec	151	76	51	38
83.3 MHz or 12 NSec	163	82	55	41
90.9 MHz or 11 NSec	178	89	60	45
100.0 MHz or 10 NSec	196	98	66	49

7.6.2.1 RAS_Control Register

The RAS_Control Register is also referred to as Memory Control Register 01 at address 84. It contains the Phase Control words for the **RAS Waveform generators**.

Table 7-11 RAS_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5MHz	> 83.5 MHz	Description	Type
RAS_WR<15:0>	31:16	29b7	29b7	51b7	Phase control word for RAS on Writes	RW
RAS_RD<15:0>	15:0	2495	24a5	2525	Phase control word for RAS on Reads	RW

7.6.2.2 CAS_RD_Control Register

The CAS_RD_Control Register is also referred to as Memory Control Register 02 at address 88. It contains the Phase Control words for the **CAS Waveform generators for read** operations.

Table 7-12 CAS_RD_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5MHz	> 83.5 MHz	Description	Type
Reserved	31:16	0000	0000	0000	Reserved - read as 0.	R0
CAS_RD<15:0>	15:0	2d23	2d95	2da5	Phase control word for CAS on Reads from a Processor	RW

7.6.2.3 BankSel_Control Register

The BankSel_Control Register is also referred to as Memory Control Register 03 at address 8c. It contains the Phase Control words for the **BankSel Waveform generators**.

Table 7-13 BankSel_RD_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5MHz	> 83.5 MHz	Description	Type
BS_WR<15:0>	31:16	2936	2936	4a36	Phase control word for BankSel on Writes	RW
BS_RD<15:0>	15:0	2612	2692	26a2	Phase control word for BankSel on Reads	RW

7.6.3 BMX_Buffer_Control Register

The XB1_Buffer_Control Register is also referred to as Memory Control Register 04 at address 90. It contains the Phase Control words for the **MRBCtrl and MWBCtrl Waveform generators**.

Table 7-14 MRB_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
MWB<15:0>	31:16	2d16	2d16	3128	Phase control word for MWBCtrl on Writes	RW
MRB<15:0>	15:0	2498	249a	249c	Phase control word for MRBCtrl on Reads	RW

7.6.4 CAS_WR_Control Register

The CAS_WR_Control Register is also referred to as Memory Control Register 05 at address 94. It contains the Phase Control words for the **CAS Waveform generators for write** operations for both bank zero and bank one.

Table 7-15 CAS_WR_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
CAS_WR_1<15:0>	31:16	24cb	24cb	255d	Phase control word for CAS on Writes to bank one.	RW
CAS_WR_0<15:0>	15:0	84b7	6537	a547	Phase control word for CAS on Writes to bank zero.	RW

7.6.5 Phase_Level_Control Register

The Phase_Level_Control Register is also referred to as Memory Control Register 06 at address 98. It contains the Phase Level words for all the Waveform generators except the internal signal Waveform generator Row. The Row phi level is located in Memory Control Register 08 - Count Control Register.

Table 7-16 Phase_Level_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
Reserved	31:30	00	00	00	reserved -read as zero.	R0
MWB_Phi<4:0>	29:25	01010	01010	01010	Phase level word for MWBCtrl	RW
MRB_Phi<4:0>	24:20	01010	01010	01010	Phase level word for MRBCtrl	RW
BS_Phi<4:0>	19:15	11000	11000	11000	Phase level word for BankSel	RW
Reserved	14:10	00000	00000	00000	reserved -read as zero.	R0
CAS_Phi<4:0>	9:5	10011	10011	10011	Phase level word for CAS	RW
RAS_Phi<4:0>	4:0	10000	10000	10000	Phase level word for RAS	RW
Above Bits in Hex Format	31:16 15:0	14ac 0270	14ac 0270	14ac 0270	Phase Level Control Register	

7.6.6 SIMM_Busy_Rd_Control Register

The SIMM_Busy_Rd_Control Register is also referred to as Memory Control Register 07 at address 9c. The DSC MC uses countdown timers on a given SIMM to determine how soon after a DRAM operation it can issue a new operation to the same SIMM. Same_Busy_X_Rd contain the values those countdown timers should use for a given operation to the same SIMMs given that a current operation, X, is followed by a Read. Diff_Busy_X_Rd are analogous timer values when addressing a different SIMM.

Table 7-17 SIMM_Busy_Rd_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
Reserved	31:30	00	00	00	reserved -read as zero.	R0
Diff_Busy_Refr_Rd	29:25	00001	00001	00001	Timer value for a Refresh followed by a Read to a different SIMM	RW
Same_Busy_Refr_Rd	24:20	01001	01100	01101	Timer value for a Refresh followed by a Read to same SIMM	RW
Diff_Busy_Wr_Rd	19:15	01010	01010	01100	Timer value for a Write followed by a Read to a different SIMM	RW
Diff_Busy_Rd_Rd	14:10	00101	00110	01000	Timer value for a Read followed by a Read to a different SIMM	RW
Same_Busy_Wr_Rd	9:5	01100	01101	10000	Timer value for a Write followed by a Read to a same SIMM	RW
Same_Busy_Rd_Rd	4:0	00111	01000	01010	Timer value for a Read followed by a Read to same SIMM	RW
Above Bits in Hex Format	31:16 15:0	0295 1587	02c5 19a8	02d6 220a	Busy Read Timers Control Register	

7.6.7 Count_Control Register

The Count_Control Register is also referred to as Memory Control Register 08 at address a0.

Row_Phi is the Phi Level value for an internal signal which determines whether the row address or the column address is output on the memory DRAM address lines.

StretchWR1 and StretchRD counts along with StretchWR0 from Memory Control Register 00, are the stretch points in the appropriate cycle when the MC checks the status of the appropriate buffer and determines whether the operation can proceed or if it needs to stall and stretch out the cycle until a time when it can finish the operation. i.e. - If we are doing consecutive reads from memory, and a CPU or U2S, for whatever reason, has not yet read out the XB1 Read buffer from the previous read, then the memory system cannot write into that buffer until the original requestor has emptied it. The MC however would like to go as far into the memory read cycle as it possibly can rather than stall the operation before it even drops RAS and CAS, so that when the buffer is ready, it will have the data ready also, and immediately load up the XB1 buffer with the correct data. (Similarly for writes, the MC would like to go as far into the write cycle as it can so that once the CPU or U2S writes into the buffer, the MC can immediately retrieve the data from the XB1 and store it.) The StretchXXX counts provide the mechanism for stalling at the correct part of the cycle.

Lastly, the MC has a set of “Ping-Pong” buffers which allow it to accept a second operation relatively quickly while it is working on the first. PPXXCnt values, are values which control when the ping-pong buffers need to flip as a function of the operation (i.e. - a Read may require the address Ping-Pong buffer to remain for six cycles from the start of the operation, whereas, a Write may require something closer to eleven cycles before it can switch to the other buffer.)

Table 7-18 Count_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
Reserved	31:30	00	00	00	reserved -read as zero.	R0
Row_Phi<4:0>	29:25	00001	00011	00011	Phase level word for internal Row signal	RW
Reserved	24:20	00000	00000	00000	Reserved - read as 0.	R0
StretchWr1<4:0>	19:15	00100	00100	00110	Stretch count value for second write	RW
StretchRd<4:0>	14:10	00010	00011	00100	Stretch count value for read	RW

Table 7-18 Count_Control Register (Continued)

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
PPWrCnt<4:0>	9:5	01010	01010	01100	Ping-Pong count value for write	RW
PPRdCnt<4:0>	4:0	00100	00101	00111	Ping-Pong count value for read	RW
Above Bits in Hex Format	31:16 15:0	0202 0944	0602 0d45	0603 1187	Count Control Register	

7.6.8 Refresh_Control Register

The Refresh_Control Register is also referred to as Memory Control Register 09 at address a4. It contains the Phase Control words for the RAS and CAS Waveform generators for a Refresh operation. Note that the Phase Level value for the first phase of RAS in a refresh cycle is NOT taken from RAS_Phi<0>, but from RefrPhiMC.

Table 7-19 Refresh_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
CAS_Ref<15:0>	31:16	2893	25b9	c913	Phase control word for CAS on Refresh	RW
RAS_Ref<15:0>	15:0	2527	2929	2d29	Phase control word for RAS on Refresh	RW

7.6.8.1 Row_Control Register

The Row_Control Register is also referred to as Memory Control Register 0A at address a8.It contains the Phase Control words for the Row Waveform generator. Row is an internal signal which controls the driving of the address lines with the Row (RAS) or column (CAS) part of the address.

Table 7-20 Row_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
Row_WR<15:0>	31:16	252b	29b5	4db5	Phase control word for Row on Writes	RW
Row_RD<15:0>	15:0	2495	2493	6493	Phase control word for Row on Reads	RW

7.6.8.2 SIMM_Busy_Wr_Control Register

The SIMM_Busy_Wr_Control Register is also referred to as Memory Control Register 0B at address b0. The DSC MC uses countdown timers on a given SIMM to determine how soon after an operation, it can issue a new operation to the same SIMM. Same_Busy_X_Wr contain the values those countdown timers should use for a given operation to the same SIMMs given that a current operation, X, is followed by a Write. Diff_Busy_X_Wr are analogous timer values when addressing a different SIMM.

Table 7-21 SIMM_Busy_Wr_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
Reserved	31:30	00	00	00	reserved -read as zero.	R0
Diff_Busy_Refr_Wr	29:25	00001	00001	00001	Timer value for a Refresh followed by a Write to a different SIMM	RW
Same_Busy_Refr_Wr	24:20	00111	01001	01010	Timer value for a Refresh followed by a Write to the same SIMM	RW
Diff_Busy_Wr_Wr	19:15	01001	01001	01100	Timer value for a Write followed by a Write to a different SIMM	RW
Diff_Busy_Rd_Wr	14:10	01000	01001	01010	Timer value for a Read followed by a Write to a different SIMM	RW
Same_Busy_Wr_Wr	9:5	01010	01010	01101	Timer value for a Write followed by a Write to the same SIMM	RW
Same_Busy_Rd_Wr	4:0	01000	01001	01010	Timer value for a Read followed by a Write to the same SIMM	RW
Above Bits in Hex Format	31:16 15:0	0274 a148	0294 a549	02a6 29aa	Busy Write Control Register	

7.6.9 SIMM_Busy_Refr_Control Register

The SIMM_Busy_Refr_Control Register is also referred to as Memory Control Register 0C at address b4. The DSC MC uses countdown timers on a given SIMM to determine how soon after an operation, it can issue a new operation to the same SIMM. Same_Busy_X_Refr contain the values those countdown timers should use for a given operation to the same SIMMs given that the current operation, X, is followed by a Refresh. Diff_Busy_X_Refr are analogous timer values when addressing a different SIMM.

Table 7-22 SIMM_Busy_Control Register

Field	Bits	<= 66.7 MHz	>66.7, <=83.5 MHz	> 83.5 MHz	Description	Type
Reserved	31:30	00	00	00	reserved -read as zero.	R0
Diff_Busy_Refr_Refr	29:25	00111	01001	01010	Timer value for a Refresh followed by a Refresh to a different SIMM	RW
Same_Busy_Refr_Refr	24:20	00111	01001	01010	Timer value for a Refresh followed by a Refresh to the same SIMM	RW
Diff_Busy_Wr_Refr	19:15	00001	00001	00001	Timer value for a Write followed by a Refresh to a different SIMM	RW
Diff_Busy_Rd_Refr	14:10	00001	00001	00001	Timer value for a Read followed by a Refresh to a different SIMM	RW
Same_Busy_Wr_Refr	9:5	01010	01010	01101	Timer value for a Write followed by a Refresh to the same SIMM	RW
Same_Busy_Rd_Refr	4:0	00111	01000	01001	Timer value for a Read followed by a Refresh to the same SIMM	RW
Above Bits in Hex Format	31:16 15:0	0e70 8547	1290 8548	14a0 85a9	Busy Refresh Control Register	

7.6.10.CSR Programming Sets

Table 7-23 CSR map and values

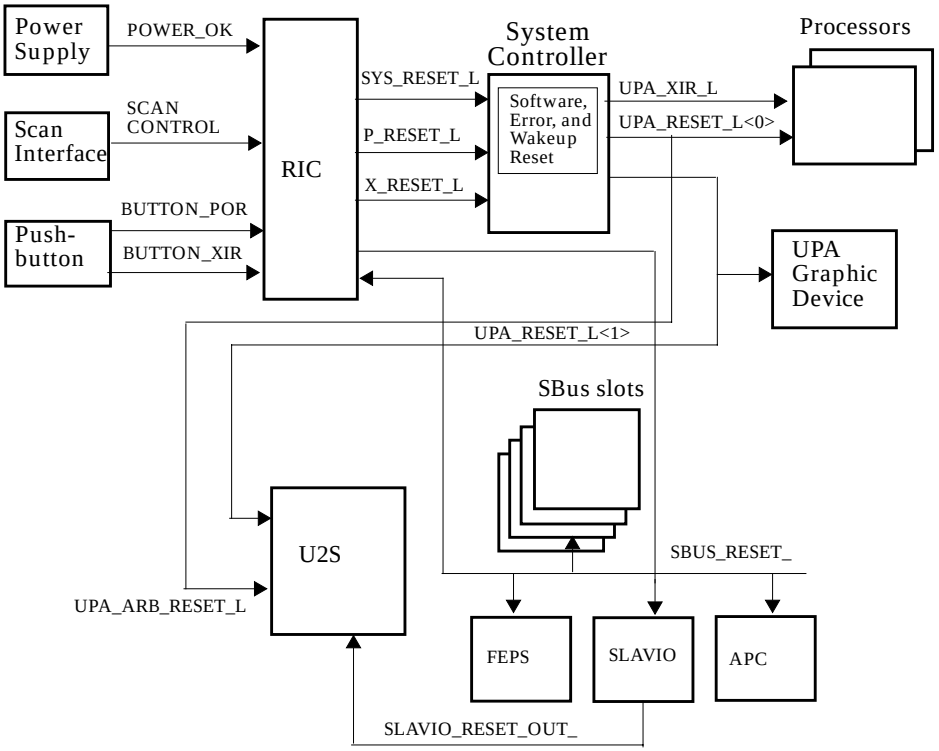
Address	Function	Power Up Value	83.3 MHz Value (Hex)	66.7 MHz Value (Hex)	100 MHz Value (Hex)
80	Mem_Control0	00c1 0f14	a0c1 0f18	a0c1 0f18	a0c1 0f18
84	RAS_Control	25c7 4ca5	29b7 24a5	29b7 2495	51b7 2525
88	CAS_RD_Control	0000 2d95	0000 2d95	0000 2d23	0000 2da5
8c	BankSel_Control	2926 249a	2936 2692	2936 2612	4a36 26a2
90	XB1_Buffer_Control	2896 249a	2d16 249a	2d16 2498	3128 249c
94	CAS_WR_Control	24bb 24b7	24cb 6537	24cb 84b7	255d a547
98	Phase_Level_Control	14ac 0278	14ac 0270	14ac 0270	14ac 0270
9c	SIMM_Busy_Rd_Control	02b5 9588	02c5 19a8	0295 1587	02d6 220a
a0	Count_Control	0601 8d24	0602 0d45	0202 0944	0603 1187
a4	Refresh_Control	2893 2537	a913 2929	2893 2527	c913 2d29
a8	Row_Control	25b5 2493	29b5 2493	252b 2495	4db5 6493
b0	SIMM_Busy_Wr_Control	0284 9148	0294 a549	0274 a148	02a6 29aa
b4	SIMM_Busy_Refr_Control	02b0 8548	1290 8548	0e70 8547	14a0 85a9

Resets 8

8.1 Introduction

All of the reset generation logic in DSC is contained in the EBus block. Resets may be triggered by external signals, internal register bits, or detection of internal errors. This chapter describes in detail what the reset behavior of the SC is. The diagram in Figure 8-1 below shows the reset inputs and reset outputs of the DSC, as well as the interconnection of resets in the system.

Figure 8-1 Resets



Note: In addition to the external signals shown above, the DSC also generates two internal signals: BX_HardReset, and BX_SoftReset.

8.2 Reset Signals

8.2.1 SYS_RESET_L

SYS_RESET_L is the power on reset.

SYS_RESET_L is supplied by the RIC chip, and is only asserted during power on. SYS_RESET_L has highest precedence, and causes everything to be reset. Because SYS_RESET_L is an asynchronous signal, it must be synchronized before being used internally by DSC.

Note – The reset outputs (UPA_RESET{0,1}_L) from the DSC are asserted asynchronously and do not require the clocks to be functioning. This coupled with asynchronous tri-stating of all bussed signals avoids drive fights.

Assertion of SYS_RESET_L causes UPA_RESET0_L, UPA_RESET1_L, BX_HardReset, and BX_SoftReset to be asserted. These output signals will be extended as necessary to meet minimum reset timing for the UPA clients. Currently the minimum requirement for the resets is five milliseconds.

Assertion of SYS_RESET_L will be logged by the POR bit in SC_Control register.

8.2.2 P_RESET_L

P_Reset_L is an output of the RIC chip which is asserted whenever there is a BUTTON_POR, or a SCAN_POR at the RIC chip. The behavior of the system in the presence of this signal is identical with SYS_RESET_L except for the logging information which indicates this as the source of the reset. This signal is synchronized to UPA clock in the same manner as SYS_RESET_L.

8.2.3 X_RESET_L

X_RESET_L is an output of the RIC chip which is asserted whenever there is a BUTTON_XIR, or a SCAN_XIR at the RIC chip. It results in UPA_XIR_L being asserted. This signal is asynchronous to the SC, and is therefore synchronized before being used. The output generated as a result of this signal is synchronously asserted and de-asserted.

8.2.4 UPA_RESET0_L

UPA_RESET0_L is used to reset the CPUs. It is also used to reset U2S's arbiter whenever the CPUs are powered down or reset.

8.2.5 UPA_RESET1_L

UPA_RESET1_L is used to reset U2S and the FFB.

8.2.6 UPA_XIR_L

UPA_XIR_L connects only to the processors. This signal is asserted for only a single UPA clock cycle. XIR is intended as a "light weight" way of regaining control of the processor when something has gone astray. Memory refresh continues and only the processors are disrupted. If an XIR is insufficient for gaining control, one of the PORs should be used.

8.2.7 BX_HardReset, BX_SoftReset

BX_HardReset and BX_SoftReset are signals generated by the EBus block and distributed internally to all logic inside the SC.

A hard reset will only occur as a result of a power on condition. BX_HardReset is triggered by SYS_RESET_L, P_RESET_L, or SOFT_POR and causes most state to be reset.

A soft reset occurs uniquely when the DSC detects a error. The cause of the error is logged in one of DSC's internal registers. BX_SoftReset will only reset state machines and counters, it will not reset control and status registers, so that the cause of the reset is preserved. Blocks reset only during error conditions must also be reset during POR conditions. Therefore, to avoid blocks having to OR together HardReset and SoftReset, the SoftReset signal is ALWAYS asserted whenever HardReset is asserted.

8.2.8 XB1 Reset

The XB1 is reset by asserting a XB1CMD of 4'b1111.

Note – Clocks are required to be functioning in order to reset the XB1 chip. As a result, all devices attached to the XB1 should make sure to have their data busses tri-stated at power up time.

8.3 Reset Operation

DSC's reset strategy is summarized in the table below. This table is taken from the Programming Model chapter in the Fusion System Specification document.

Table 8-1 DSC Reset Strategy

Reset Type	Bit Set	Signal Assertions	SC FSM's	SC Status Regs
Power On	POR	UPA_Reset_L<1:0 for 12.8 ms	All Reset	All Reset
Software	SOFT_POR	UPA_Reset_L<1:0 for a minimum of 5 ms	All Reset	All reset except SOFT_POR
XIR	SOFT_XIR	XIR to all processors	Unaffected	Only XIR source affected.
Button Reset	B_POR	UPA_Reset_L<1:0 for a minimum of 5 ms	All Reset	All Reset except B_POR
XIR Button	B_XIR	UPA_XIR_L for 1 System Cycle	Not affected	Only B_XIRsource affected.
Wakeup Interrupt	Wakeup	UPA_Reset_L<0> for a minimum of 5 ms	SC's UPA Arbiter is Reset	Only WAKEUP source affected.
Fatal error condition	FATAL	UPA_Reset_L for a minimum of 5 ms	All Reset	Only FATAL source affected.

1. FATAL is only set if EN_FATAL is also set. Also, the source of the error will be set in one of the port status registers.
2. For any of the reset types, the correct reset condition is logged into the SC_Status_REG.

There are several ways in which a reset can be triggered:

1. Assertion of SYS_RESET_L input.
2. Assertion of P_RESET_L input.
3. Assertion of X_RESET_L input.
4. Setting the SOFT_POR bit in the SC_Control register.
5. Setting the SOFT_XIR bit in the SC_Control register.
6. Detection of a fatal error.
7. Wakeup interrupt.

The table below shows the same information as the previous table, but in a slightly different form. It also adds BX_SoftReset and BX_HardReset.

Table 8-2 Reset Table

Reset Trigger	UPA_RESET0L	UPA_RESET1_L	UPA_XIR_L	BX_Hard Reset	BX_Soft Reset	XB1 Reset	Bit Set	Note
SYS_RESET_L	Y	Y	N	Y	Y	Y	POR	
P_RESET_L	Y	Y	N	Y	Y	Y	B_POR	1
X_RESET_L	N	N	Y	N	N	N	B_XIR	
SOFT_POR bit	Y	Y	N	Y	Y	Y	SOFT_POR	
SOFT_XIR bit	N	N	Y	N	N	N	SOFT_XIR	
fatal error	Y	Y	N	N	Y	Y	FATAL	2
wakeup	Y	N	N	N	N	N	WAKEUP	3

1. The B_POR bit must be set, all other register bits must be reset.
2. Resets are generated only if the EN_FATAL bit is set. The error is logged regardless of whether EN_FATAL is set or not.
3. UPA address bus 0 arbiter must be reset.

8.4 Wakeup Functionality

Only the processors are allowed to go to sleep in the Ultra 2 system, other system components remain awake. The SLEEP bit for each processor should be set in its own SC_Port_Config register. When an interrupt packet is directed to a processor, the DSC will issue a S_INAK, and the interrupter is forced to resend the interrupt. At the same time, PIF informs the EB block, which asserts UPA_RESET0L to wake up the processors and reset the UPA address bus arbiters in U2S and DSC. DSC will continue issuing S_INAKs until the SLEEP bit is cleared by a processor after it has woken up. In the mean time, any reads directed to any sleeping processor will cause an S_ERR to be issued, and any writes will be dropped on the floor, per the UPA Interconnect Architecture document.

EBus Interface 9

9.1 Introduction

The DSC EBus block(EB) is almost a clone of the USC EB. This chapter is essentially the same as the USC EBus Interface chapter.

Since DSC does not contain a data path, an 8 bit EBus port has been provided to allow reading and writing of DSC's internal registers.

The EBus is controlled by the Slavio chip. Slavio can handle up to three EBus clients: EPROM, TOD/NVRAM, and a generic port. DSC is hooked up to the generic port. Therefore, DSC's register space is actually part of SBus address space.

The EBus interface is implemented in the EBus (EB) block.

9.2 Functional Description

Since EBus runs asynchronously with respect to DSC's clock (EBus signals are clocked using the SBus clock), all EBus inputs to DSC pass through a boundary register and a dual-ranked synchronizer. All outputs from DSC to EBus (EB_RDY_L and EB_DATA) are clocked out through a boundary register using DSC's clock and are synchronized by Slavio using dual-ranked synchronizers.

Registers in DSC are read or written using one level of indirection. EB contains only two registers, an 8 bit EB_Addr register and a 32 bit EB_Data register. Whenever the processor wants to read or write one of DSC's internal registers it must first write the internal address of

the register it wants to access into the EB_Addr register, then perform a read or write to the Data register. The address map for DSC's internal registers is documented in the "System Register Definitions" document.

Figure 9-1 EBus Register Map

EB_ADDR[2:0]	Register
000	EB_Addr[7:0]
001	maps into EB_Addr[7:0]
010	maps into EB_Addr[7:0]
011	maps into EB_Addr[7:0]
100	EB_Data[31:24]
101	EB_Data[23:16]
110	EB_Data[15:8]
111	EB_Data[7:0]

Only three bits are needed for EB_ADDR.

There are some very strict rules that the processor must follow when it is accessing DSC through the EBus, since it only supports byte reads and writes, and we are trying to access 32 bit data.

For reads, the 32 bit datum pointed to by the contents of the EB_Addr register is fetched into a 32 bit staging register when byte 0 (EB_ADDR[2:0] == 100) is read, and byte 0 is returned on the EBus. Successive reads of bytes 1, 2, and 3 of the EB_Data register will return the remaining three bytes in the staging register.

For writes, the processor must begin by writing to byte 0 of the EB_Data register, then bytes 1, 2, and 3. The EBus block will store the bytes into a 32 bit staging register until all four bytes have been written into it. The EBus block will not actually update the register pointed to by the contents of EB_Addr until byte 3 is written.

The read or write should be done as an atomic operation. Atomicity is not enforced in hardware, and has to be enforced in software using some sort of lock. EBus accesses should can be interrupted, so long as they are allowed to complete. Only one processor can be allowed access to the EBus at any given time. Otherwise results will be unpredictable.

Internally, EB will decode the address and generate a select for either PIF, MC, or CC (for DSC

only), and pass on the register address. It will perform all the packing and unpacking and handshaking necessary to complete the transfer.

9.3 Timing Diagrams

The timing diagrams below show both internal timing and external timing for EBus accesses.

Figure 9-2 Internal Read Timing

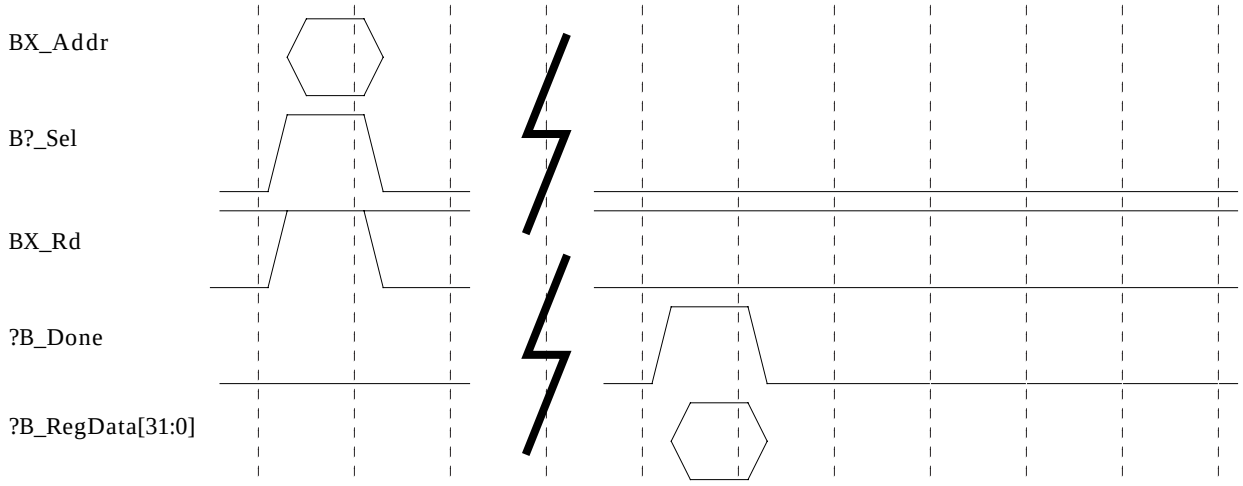


Figure 9-3 Internal Write Timing

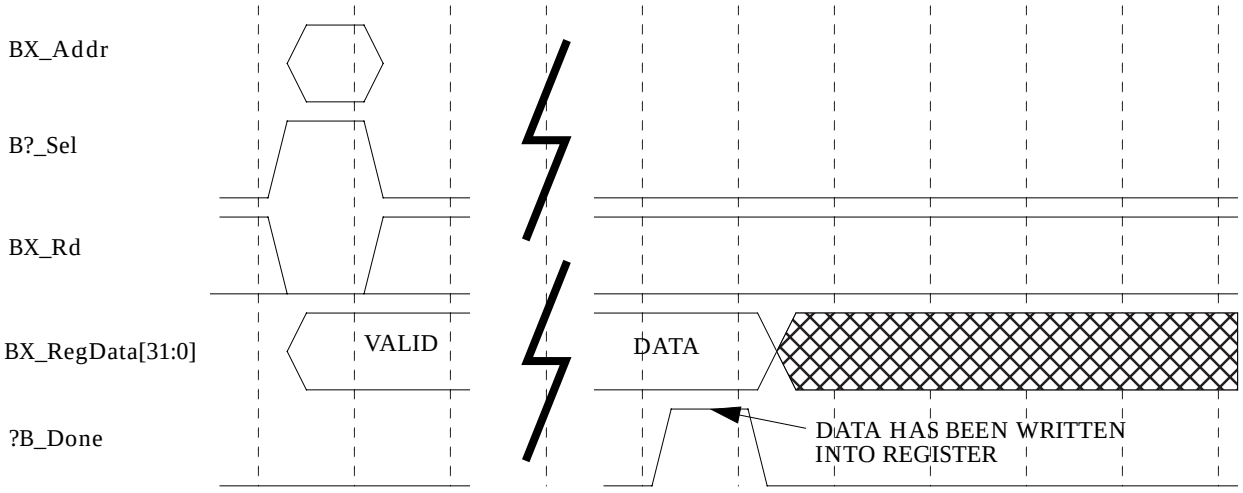


Figure 9-4 External EBus Read Timing

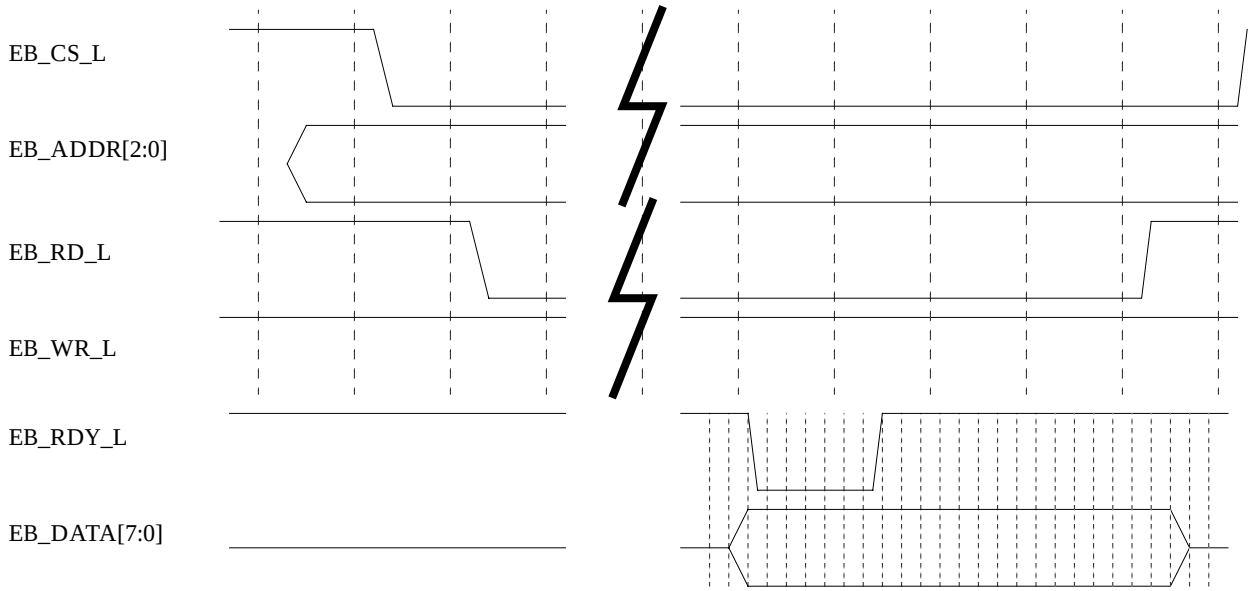
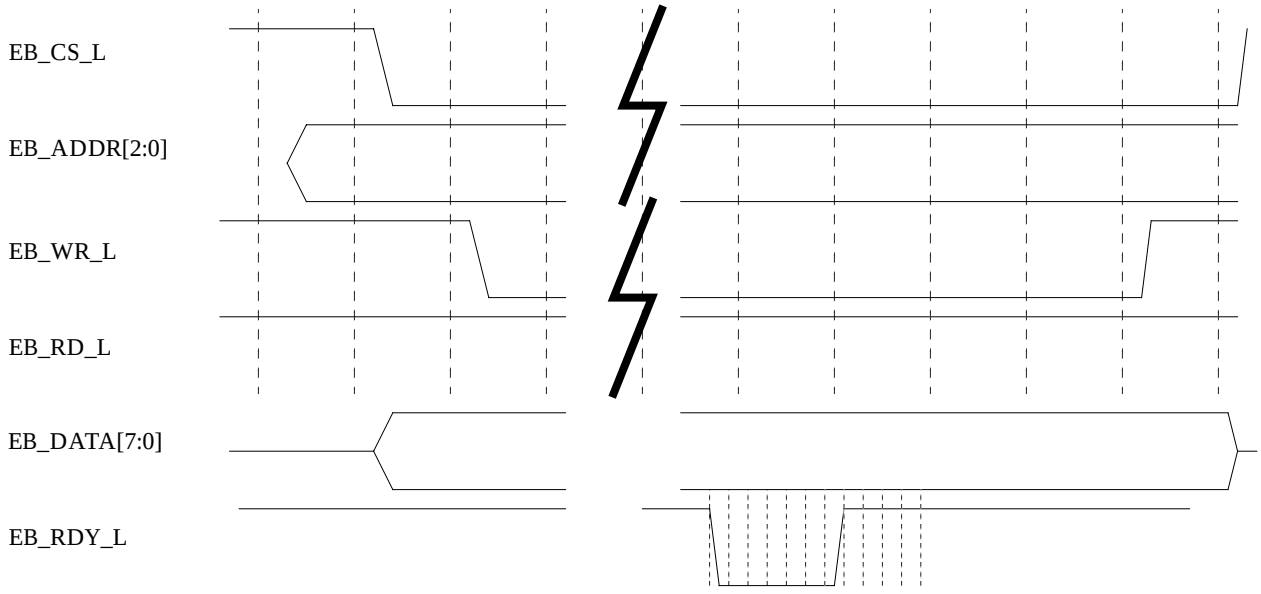


Figure 9-5 External EBus Write Timing



SUN MICROELECTRONICS

2550 Garcia Avenue
Mountain View, CA, USA 94043
1-800-681-8845
www.sun.com/sparc

Part Number: 802-7511-02