SPARC Technology Business

Preliminary

STP1030

May 1995

UltraSPARC-I

DATA SHEET — High-Performance 64-Bit RISC Processor

INTRODUCTION

The STP1030, UltraSPARC-I, is a high-performance, highly-integrated superscalar processor implementing the SPARC V9 64-bit RISC architecture. The STP1030 is capable of sustaining the execution of up to four instructions per cycle even in the presence of conditional branches and cache misses. This sustained performance is supported by a decoupled Prefetch and Dispatch Unit with Instruction Buffer to feed the Execution Unit. On the output side of the Execution Unit, Load and Store buffers completely decouple pipeline execution from data cache misses. Instructions predicted to be executed are issued in program order to multiple functional units, execute in parallel and can complete out of order. In order to further increase the number of instructions executed per cycle, instructions from different blocks (e.g. instructions before and after a conditional branch) can be issued in the same group.

The STP1030 supports 2D, 3D graphics, image processing, video compression and decompression and video effects through the sophisticated VISual Instruction Set. This instruction set supports high levels of multimedia performance including real-time H.261 video compression/ decompression and 2 streams of MPEG-2 decompression at full broadcast quality with no additional hardware support.

Features:

- SPARC V9 Architecture Compliant
- Binary Compatible with all SPARC Application code
- VISual (Multimedia Capable) Instruction Set
- Multi-Processing Support
  - Glueless 4-processor connection with minimum latency
  - Snooping or Directory Based Protocol Support
- 4-way SuperScalar Design with 9 execution units
  - 4 Integer Execution Units
  - 3 Floating-point Execution Units
  - 2 Graphics Execution Units
- Selectable Little or Big Endian Byte Ordering
- 64-Bit Address Pointers
- 16KByte Non-blocking Data Cache
- 16KByte Instruction Cache
  - In-Cache 2-bit Branch Prediction
  - Single Cycle Branch Following
- Integrated 2nd Level Cache Controller
  - Supports .5-4MBytes Cache Sizes
  - Sustained throughput of 1 load/cycle
  - 2.6Gbyte/sec Processor-Cache bandwidth
- Block Load/Store Instructions
  - 1.3GByte/sec processor-memory bandwidth
  - 600 MByte/sec Sustained Processor-Memory Transfers
- Ease of Use
  - JTAG Boundary scan
  - Performance Instrumentation
- Technology/packaging
  - 0.5um 4-layer metal CMOS process
  - Operates at 3.3V
  - 521 pin plastic Ball Grid Array (BGA)
- Power management

BLOCK DIAGRAM

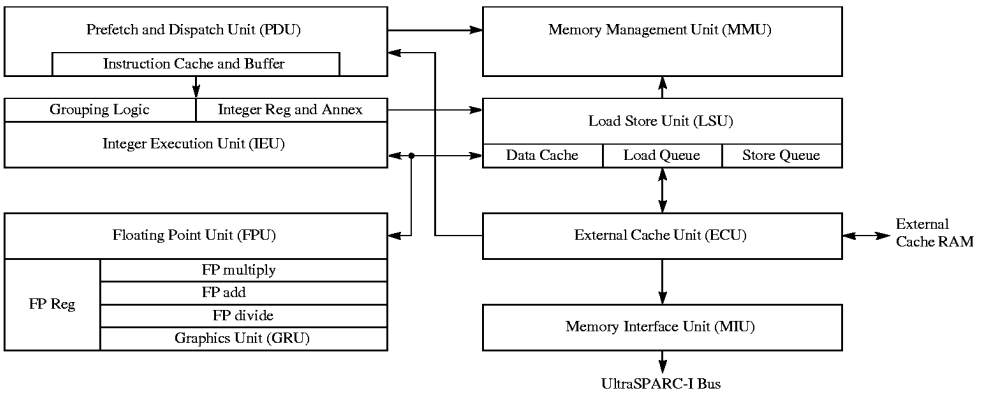

Figure 1. Functional Block Diagram

ULTRASPARC-I COMPONENT OVERVIEW

In a single chip implementation, the UltraSPARC-I processor integrates the following components (see *Figure 1*):

- A prefetch, branch prediction and dispatch unit
- A 16 Kbytes instruction cache
- An MMU composed of a 64-entry iTLB and a 64-entry dTLB
- An integer execution unit with two ALUs
- One load/ store unit with a separate address generation adder
- A load buffer and a store buffer decoupling data accesses from the pipeline
- A 16 Kbyte data cache
- A floating-point unit with independent add, multiply and divide/ square root sub-units
- a graphics unit composed of two independent execution pipelines
- a unit controlling accesses to the external cache
- a unit responsible for main memory and I/ O accesses

Prefetch and Dispatch Unit

The prefetch and dispatch unit fetches instructions ahead of time (before they are actually needed in the pipeline) so that the execution units do not starve for instructions. Instructions can be prefetched from all levels of the memory hierarchy, i.e. the instruction cache, the external cache and main memory. In order to prefetch across conditional branches, a dynamic branch prediction scheme is implemented in

hardware. The outcome of a branch is based on a two-bit history of the branch. A "next field" associated with every four instructions in the instruction cache (I-cache) points to the next I-cache line to be fetched. The use of the next field makes it possible to follow taken branches and basically provided the same instruction bandwidth achieved while running sequential code. Prefetched instructions are stored in the instruction buffer until they are sent to the rest of the pipeline. Up to 12 instructions can be buffered.

Instruction Cache

The instruction cache is a 16 Kbyte two-way set associative cache with 32 byte blocks. The cache is physically indexed and contains physical tags. The set is predicted as part of the "next field" so that only the index bits of an address are necessary to address the cache (13 bits which matches the minimum page size). The instruction cache returns up to 4 instructions from an 8 instruction wide line.

Memory Management Unit (MMU)

The MMU provides mapping between a 44-bit virtual address and a 41-bit physical address. That is accomplished through a 64-entry iTLB for instructions and a 64-entry dTLB for data, both fully associative. UltraSPARC-I provides hardware support for a software-based TLB miss strategy. A separate set of global registers is available whenever an MMU trap is encountered. Page sizes of 8K, 64K, 512K and 4 Mbytes are supported.

Integer Execution Unit (IEU)

Two ALUs form the main computational part of the IEU. An early-out multi-cycle integer multiplier and a multi-cycle integer divider are also part of the IEU. Eight register windows and four sets of global registers are provided (normal, alternate, MMU and interrupt globals). The trap registers (UltraSPARC-I supports five levels of traps) are part of the IEU.

Load/Store Unit (LSU)

The LSU is responsible for generating the virtual address of all loads and stores (including atomics and ASI loads), for accessing the data cache, for decoupling load misses from the pipe through the load buffer, for decoupling the stores through a store buffer. One load or one store can be issued per cycle.

Data Cache

The data cache is a write-through non-allocating 16 Kbyte direct mapped cache with two 16-byte sublocks per line. It is virtually indexed and physically tagged. The tag array is dual ported so that tag updates due to line fills don't collide with tag reads for incoming loads. Snoops to the D-cache use the second tag port so that incoming load can proceed without being held up by a snoop.

Floating-Point Unit (FPU)

The separation of the execution units in the FPU allows UltraSPARC-I to issue and execute two floating-point instructions per cycle. Source data and results data are stored in the 32-entry register file, where

each entry can contain a 32-bit value or a 64-bit value. Most instructions are fully pipelined (throughput of one per cycle) have a latency of three and are not affected by the precision of the operands (same latency for single or double precision). The divide and square root instructions are not pipelined and take 12/ 22 cycles (single/ double) to execute but they do not stall the processor. Other instructions, following the divide/ sqrt can be issue, executed, and retired to the register file before the divide/ sqrt finishes. A precise exception model is maintained by synchronizing the floating-point pipe with the integer pipe and by predicting traps for long latency operations.

Graphics Unit (GRU)

UltraSPARC-I introduces a comprehensive set of graphics instructions that provide fast hardware support for two-dimensional and three-dimensional image and video processing, image compression, audio processing, etc. 16-bit and 32-bit partitioned add, boolean and compare are provided. 8-bit and 16-bit partitioned multiplies are supported. Single cycle pixel distance, data alignment, packing and merge operations are all supported in the GRU.

External Cache Unit (ECU)

The main role of the ECU is to handle I-cache and D-cache misses efficiently. The ECU can handle one access per cycle to the external cache. Accesses to the external cache are pipelined, take three cycles (pin-to-pin) and return 16 bytes of instructions or data per cycle. This can effectively make the external cache a part of the pipeline which means that for programs with large data sets, data can be maintained in the external cache and instructions scheduled with load latencies based on the E-cache latency. Floating-point applications can use this feature to effectively "hide" D-cache misses. The size of the external cache can be 512K, 1M, 2M or 4Mbyte, where the line size is always 64 bytes. A MOESI (modified, own, exclusive, shared, invalid) protocol is used to maintain coherency across the system.

The ECU provides overlap processing during load and store misses. For instance stores that hit the E-cache can proceed while a load miss is being process. The ECU is also capable of processing reads and writes indiscriminately without a costly turn around penalty (only 2 cycles). Snoops are also handle by the ECU.

Block loads and block stores, which load/ store a 64-byte line of data from memory to the floating-point register file, are also processed efficiently by the ECU, providing high transfer bandwidth without polluting the external cache.

Memory Interface Unit (MIU)

All transactions to the system such as external cache misses, interrupts, snoops, writebacks, etc. are handled by the MIU. The MIU communicates with the system at a frequency lower than UltraSPARC-I frequency (either 1/ 2 or 1/ 3).

TABLE 1: Quick Pin Reference - System Interface

Symbol	Type	Name and Function
SYSADR[35:0]	I/O	Bidirectional UltraSPARC-I Bus transaction request bus. Maximum of 3 other masters and 1 system controller also connected to this bus.
ADR_VLD	I/O	Bidirectional radial UltraSPARC-I Bus signal between UltraSPARC-I and the System. Driven by UltraSPARC-I to initiate SYSADR transactions to the System. Driven by System to initiate Coherency, Interrupt or Slave transactions to UltraSPARC-I. Synchronous to system clock.
NODE_RQ[2:0]	I	UltraSPARC-I system address bus arbitration request from up to 3 other UltraSPARC-I Bus ports that might be sharing the SYSADR. Used by UltraSPARC-I for the distributed SYSADR arbitration protocol. Connection to other UltraSPARC-I Bus ports is strictly dependent on the Master ID allocation. Synchronous to system clock.
SC_RQ	I	UltraSPARC-I system address bus arbitration request from the system. Used by UltraSPARC-I for the distributed SYSADR arbitration protocol. Synchronous to system clock.
S_REPLY[[3:0]	I	UltraSPARC-I system Reply packet, driven to UltraSPARC-I. Bit 4 of the UltraSPARC-I Bus S_REPLY is not used by UltraSPARC-I. Synchronous to system clock.
DATA_STALL	I	This is asserted with or after an S_REPLY to hold output system data or signal the delay in arrival of input data from the system.
P_REPLY[4:0]	O	UltraSPARC-I system reply packet, driven by UltraSPARC-I to the system. Synchronous to system clock.
NODEX_RQ	O	UltraSPARC-I system address bus arbitration request. Asserted when UltraSPARC-I needs to drive SYSADR. Connected to all other UltraSPARC-I Bus ports which share this address bus, and the system. Synchronous to system clock.

TABLE 2: Quick Pin Reference - External Cache Interface

Symbol	Type	Name and Function
EDATA[127:0]	I/O	Ecache Data bus. Connects UltraSPARC-I to the Ecache data rams and the UDB. Synchronous to processor clock.
EDPAR[15:0]	I/O	Data bus parity. Odd parity is driven for all EDATA transfers, and checked if UltraSPARC-I or the UDB is the receiver. The most significant bit serves as the parity for the most significant byte of EDATA. Synchronous to processor clock.
TDATA[24:0]	I/O	Bidirectional data bus for Ecache tag rams. Bits 24:22 carry the MOESI state: Dirty, Exclusive, Valid. Bits[21:0] carry the physical address bits [40:19]. This allows a minimum cache size of 512kbytes. All of the TDATA bits are used, even when the ecache is > 512kbytes. This is because there is no sizing in the tag compare for ecache hit generation. Synchronous to processor clock.
TPAR[3:0]	I/O	Bidirectional data bus for Ecache tag rams. Odd Parity for TDATA[24:0]. TPAR[3] covers TDATA[24:22]. TPAR[2] covers TDATA[21:16]. TPAR[1] covers TDATA{15:8], TPAR[0] covers TDATA[7:0]. Synchronous to processor clock.

TABLE 2: Quick Pin Reference - External Cache Interface

Symbol	Type	Name and Function
$\overline{\text{BYTEWE}}$[15:0]	O	Byte write enables for synchronous pipelined Ecache srams. Bit [0] controls EDATA[127:120]. Bit 15 control EDATA[7:0]. Byte write control is necessary because the first-level data cache is write-through. Synchronous to processor clock.
ECAD[17:0]	O	Address for Ecache data srams. Corresponds to physical address [21:4]. Allows a maximum 4mbyte Ecache. Synchronous to processor clock.
ECAT[15:0]	O	Address for Ecache tag srams. Corresponds to physical address [21:6]. Allows a maximum 4mbyte Ecache. Synchronous to processor clock.
$\overline{\text{DSYN\_WR}}$	O	Write enable for Ecache data srams. Active low. Synchronous to processor clock.
$\overline{\text{DOE}}$	O	Active low for all sram data reads and writes. Synchronous to processor clock.
$\overline{\text{TSYN\_WR}}$	O	Write enable for Ecache tag srams. Active low. Synchronous to processor clock.
$\overline{\text{TOE}}$	O	Active low for all tag data sram reads and writes. Synchronous to processor clock.

TABLE 3: Quick Pin Reference - Clock Interface

Symbol	Type	Name and Function
CLKA	I	This pin provides STP1030 with its primary clock source and is the positive differential clock input.
CLKB	I	This pin provides STP1030 with its primary clock source and is the negative differential clock input.
LOOPCAP	I	The external PLL loop filter connects to this pin to filter the analog voltage which controls the PLL VCO.
SCLK_MODE	I	Indicates clock divider mode - system frequency is /2 or /3 of the processor frequency
SDBCLKA, SDBCLKB	I	These are buffered complementary versions of the system clock that drives the UDB. They are used to generate the phase signal which allows UltraSPARC-I to synchronize communication to the system with respect to the system clock.
PLLBYPASS	I	When asserted this pin caused the phase-lock loop to be bypassed. The clock from the differential receiver is directly passed to the clock trunk.
STOP_CLOCK	O	Indicates clock has stopped.
L5CLK	O	A buffered version of UltraSPARC-I's internal level 5 clock to be used for determining PLL lock or clock tree delay when UltraSPARC-I is in PLL bypass mode.

TABLE 4: Quick Pin Reference - JTAG/Debug Interface

Symbol	Type	Name and Function
TDO	O	IEEE 1149 test data output. A three-state signal driven only when that TAP controller is in the shift-DR state.
TDI	I	IEEE 1149 test data input. This pin is internally pulled to logic one when not driven.
TCK	I	IEEE 1149 test clock input. This pin if not hooked to a clock source must always be driven to a logic 1 or a logic 0.

TABLE 4: Quick Pin Reference - JTAG/Debug Interface (Continued)

Symbol	Type	Name and Function
TMS	I	IEEE 1149 test mode select input. This pin is internally pulled to logic one when not driven.
$\overline{\text{TRST}}$	I	IEEE 1149 test reset input (active low). This pin is internally pulled to logic one when not driven.
RAM_TEST	I	When asserted this pin forces the processor into SRAM test mode allowing direct access to the cache SRAMs for memory testing.
MISC_BIDIR[14:0]	I/O	These are miscellaneous bidirectional signals used for test, debug and instrumentation. Some of them are used to improve internal operation observability, such as pipeline monitoring signals. Their exact functions are TBD.
EXT_EVENT	I/O	This is an open drain bidirectional signal used to indicate the clock should be stopped. This signal is wired-or with the other ext_event signals from other devices so that once one is activated all are activated. It is a debug signal which is set inactive on production systems.
PM_OUT	O	Used for on-chip process monitors (reserved for IC manufacturing only).
TEMP_SEN[1:0]	O	Defines the end points of the temperature sense element on the module used to measure the processor temperature (reserved for IC manufacturing only).

TABLE 5: Quick Pin Reference - Initialization Interface

Symbol	Type	Name and Function
$\overline{\text{RESET}}$	I	Driven for POR (power-on) resets. Asserted asynchronously. Deasserted synchronous to system clock. Active low.
$\overline{\text{XIR}}$	I	Driven to signal XIR resets. Actually acts like a non-maskable interrupt. Synchronous to system clock. Active low.
EPD	O	Asserted when UltraSPARC-I is in power-down mode.

TABLE 6: Quick Pin Reference - UDB Chip Interface

Symbol	Type	Name and Function
SDB_UEH	I	Asserted when the High UDB drives Edata[127:64], if there is an uncorrectable ECC error associated with that data. Synchronous to system clock.
SDB_UEL	I	Asserted when the Low UDB drives Edata[63:0], if there is an uncorrectable ECC error associated with that data. Synchronous to system clock.
SDB_CEH	I	Asserted when the High UDB drives Edata[127:64], if the data has a corrected single-bit error. Synchronous to system clock.
SDB_CEL	I	Asserted when the Low UDB drives Edata[63:0], if the data has a corrected single-bit error. Synchronous to system clock.
SDB_CNTL[4:0]	O	UltraSPARC-I controls the UDB's drive and receive of EDATA. Asserted with valid EDATA when driving data to UDB. Asserted the cycle before the UDB should drive data. Synchronous to system clock.

ULTRASPARC-I SUBSYSTEM

A complete UltraSPARC-I subsystem consists of the UltraSPARC-I processor, synchronous SRAM components for the external cache tags and data, and two system data buffer(UDB) chips. The UDBs isolate the external cache from the system, and provide data buffers for incoming and outgoing system transactions as well as provide ECC generation and checking.

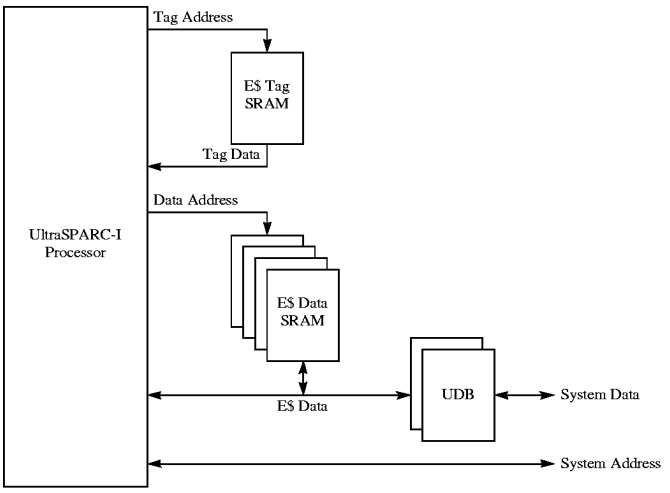

Figure 2. UltraSPARC-I Subsystem System Interface

Introduction

In this chapter the interaction of the UltraSPARC-I CPU with the external cache (E-cache) and the data buffer (UDB) is described. We first give an overview of the main buses used by UltraSPARC-I when interacting with the E-cache, and the UDB.

Transactions occurring at the pins (as opposed to transactions such as an I-cache miss vs. a D-cache miss) are discussed. Logical timing diagram (based on cycles) accompany the discussion.

The physical characteristics of transactions (setup time, propagation delay, hold time, etc.) occurring within the module and at the boundary of the module are provided. Some other physical characteristics of UltraSPARC-I, such as clocking requirements, reset operation, and test support are also included.

Overview of UltraSPARC-I Interface

The main interfaces from/ to UltraSPARC-I are shown in Figure 1-16. A typical module includes an external cache composed of the tag part and the data part. Both of them can be implemented using commodity RAMs. Separate address and data buses are provided from/ to the tag and data RAMs for

increased performance.The main role of the Data Buffer Chip is to isolate UltraSPARC-I and its external cache from the main system data bus so that the interface can operate at processor speed (reduced capacitance loading). The data buffer also provides overlapping between system transactions and local E-cache transactions even when the latter needs to use part of the data buffer. The logic to control the buffer chip is included on UltraSPARC-I to provide fast data transfers from/ to UltraSPARC-I or from/ to the external cache and the system. A separate address bus and separate control signals are provided for supporting system transactions. Clock signals, reset pins, observability pins and JTAG support are also part of UltraSPARC-I interfaces and will be described thereafter.

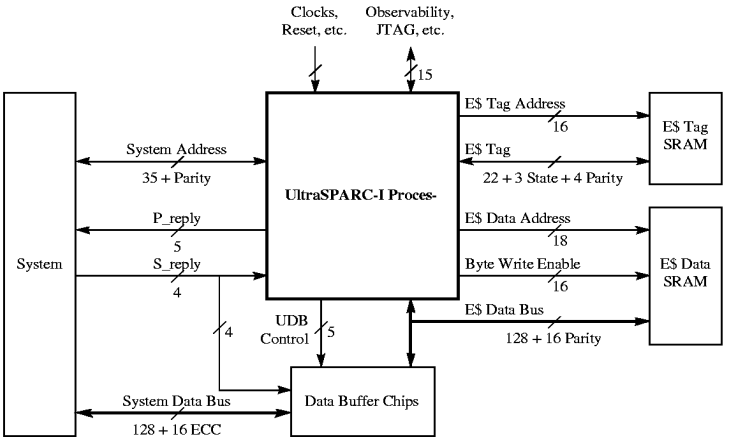

Figure 3. Main UltraSPARC-I Interfaces

Cache Coherence Protocol

This section describes the protocol used to maintain coherency between UltraSPARC-I's internal caches, the external cache and the system. "System" refers to any other location within the same coherency domain as UltraSPARC-I, for instance it includes caches of other processors connected to the interconnect.

Inclusion in the E-cache (all lines containing data currently held in the internal caches are in the external cache, even when the caches are turned off) is maintained for both the I-cache and the D-cache. The state of these lines forms a part of the tag kept in the external tag RAM.

The cache coherence protocol is point-to-point write-invalidate. It is based on the 5 MOESI states maintained in the E-cache tags of each master port (e.g. UltraSPARC-I). The E-cache tags have one of the following five states (MOESI):

- Exclusively Modified (M)
- Shared Modified (O)
- Exclusive Clean (E)
- Shared Clean (S)
- Invalid (I)

Three bits in the tag RAM defined the state of each line as follow:

TABLE 7: External Cache Coherency State Definition

STATE	State Bit		
	Valid	Modified	Exclusive
Invalid (I)	0	0	0
Shared Clean (S)	1	0	0
Exclusive Clean (E)	1	0	1
Shared Modified (O)	1	1	0
Exclusively Modified (M)	1	1	1

The cache coherence protocol operates only on Physically Indexed Physically Tagged (PIPT) writeback caches. The unit of cache coherence is a block size of 64 bytes which corresponds to one E-cache line. Coherent read/ write transactions transfer data in a 64-byte blocks only, using 4 quadwords.

The state diagram representing the allowed transactions is shown in *Figure 4*.

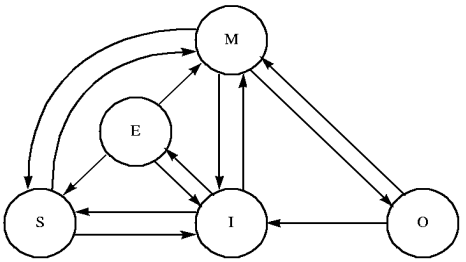

Figure 4. Cache Coherency Protocol State Diagram

Table 8 describes all transitions shown in *Figure 4*. It also shows the transactions that are initiated by either UltraSPARC-I or the system and the acknowledgment that is expected following that transaction.

TABLE 8: Transitions Allowed for Cache Coherency Protocol

Transition	Description	Transition request to/by UltraSPARC-I	Acknowledgment
I → E	Load miss; data coming from memory to an invalid line (no other cache has the data).	P_RDS_REQ	S_RBU
I → S	Load miss; data provided by another cache to an invalid line. Instruction cache miss, instructions provided either from another cache or from memory.	P_RDS_REQ P_RDSA_REQ	S_RBS S_RBS
I → M	Store miss on an invalid line.	P_RDO_REQ	S_RBU
E→ M	Store hit to an Exclusive Clean line in the cache.	No Transaction	No Transaction
E → S	Request from system to share this line (load miss from other processor).	S_CPB_REQ	P_SACK|P_SACKD followed by S_CRAB
E → I	i) A clean line is victimized by the processor. ii) Request from system to invalidate this line (store miss from other processor to an Exclusive Clean line). iii) Block Store with invalidate with also produce this transaction.	P_RDS_REQ or P_RDSA_REQ or P_RDO_REQ S_CPI_REQ P_WRI_REQ S_INV_REQ	S_RBU or S_RBS S_RBS S_RBU P_SACK|P_SACKD followed by S_CRAB S_WAB P_SACK|P_SACKD
S → M	Store hit to a Shared Clean line.	P_RDO_REQ	S_OAK
S → I	i) A Shared Clean line is victimized by UltraSPARC-I. ii) Another processor wants to write this shared datum. iii) Block Store with invalidate to a Shared Clean line, either from this STP1030 or from another processor connected to the system.	P_RDS_REQ or P_RDSA_REQ or P_RDO_REQ S_INV_REQ P_WRI_REQ S_INV_REQ	S_RBU or S_RBS S_RBS S_RBU P_SACK|P_SACKD S_WAB P_SACK|P_SACKD
M → 0	Request from other processor to read a modified line, line stays modified (as opposed to M-->S).	S_CPB_REQ	P_SACK|P_SACKD followed by S_CRAB

TABLE 8: Transitions Allowed for Cache Coherency Protocol (Continued)

Transition	Description	Transition request to/by UltraSPARC-I	Acknowledgment
M → I	i) A Modified line is victimized by the processor (writeback).	P_WRB_REQ	S_WAB
	ii) Request from the system to invalidate this line (store miss from other processor to Modified line). iii) Block Store with invalidate from this STP1030 or from another processor connected to the system, to a Modified line.	S_CPI_REQ	P_SACK|P_SACKD followed by S_CRAB (and later followed by S_WBCAN instead of S_WAB if the datum was victimized)
		P_WRI_REQ S_INV_REQ	S_WAB P_SACK|P_SACKD (and later followed by S_WBCAN instead of S_WAB if the datum was victimized)
M → S	Request from other processor to read a modified line, line becomes clean (as opposed to M-->O)	P_CPB_MSI	P_SACK|P_SACKD followed by S_CRAB
O → I	Request from other processor to invalidate this line (store hit or miss from other processor to a Share Modified line).	S_INV_REQ or S_CPI_REQ	P_SACK|P_SACKD P_SACK|P_SACKD followed by S_CRAB
O → M	Store hit to a Share Modified line.	P_RDO_REQ	S_OAK

UltraSPARC-I as a UltraSPARC-I Bus Port

The UltraSPARC-I Bus Interconnect Architecture (refer to document) defines the architecture for a family of tightly coupled, cache consistent, shared memory multiprocessor systems. UltraSPARC-I Bus provides low latency to memory, high bandwidth and fast MP data sharing. UltraSPARC-I Bus transactions are carried over a packet-switched bus with independent scheduling of separate (and possibly multiple) address and data buses.

UltraSPARC-I and its cache subsystem (including UDB) forms a UltraSPARC-I Bus module, and interfaces to the interconnect using the UltraSPARC-I Bus interface definition called the UltraSPARC-I Bus port. Similarly, the I/ O subsystem, and the graphics subsystem may reside on a UltraSPARC-I Bus module.

The physical connection between UltraSPARC-I and the UltraSPARC-I Bus mainly consist of a bidirectional address bus for transaction request from UltraSPARC-I to the UltraSPARC-I Bus interface and from UltraSPARC-I Bus interface to UltraSPARC-I, two unidirectional (one incoming, one outgoing) reply buses for flow control, and a bidirectional requests for a distributed address bus arbitration scheme. The snoop bus, although not part of the UltraSPARC-I module, is present and is used to manage duplicate tags, for efficient data sharing.

Table 9 shows the UltraSPARC-I Bus Port interface as specified in the UltraSPARC-I Bus spec and the corresponding pins for UltraSPARC-I.

TABLE 9: UPA Port Interface for UltraSPARC-I

UPA Port Interface		UltraSPARC-I Interface
UPA_DataBus[144]	↔	EDATA[127:0], EDPAR[15:0]
UPA_ECC_Valid[2]	↔	
UPA_AddressBus[37]	↔	SYSADR[38]
UPA_Addr_Valid	↔	ADR_VLD
UPA_Addr_Arb[5]	↔	NODE_RQ[2:0], REQUEST_OUT, SC_RQ
UPA_P_REPLY[5]	←	P_REPLY[5]
UPA_S_REPLY[5]	→	S_REPLY[3:0]
UPA_SnoopBus[36]	→	
UPA_SnoopCntl[13]	↔	
UPA_Port_ID[5]	→	
UPA_Reset	→	RESET
UPA_Sys_Clk[2]	→	CLKA, CLKB
UPA_ClkCntl[4]	↔	
UPA_JTAG[4]	→	
UPA_Slave_INT	←	
UPA_Mode	→	
UPA_Wakeup_Reset	→	

UltraSPARC-I is both a UltraSPARC-I Bus master and a UltraSPARC-I Bus slave.

As a UltraSPARC-I Bus master it issues read/ write transactions to the interconnect using part of the UltraSPARC-I Bus transaction set (Section 5.7.). UltraSPARC-I splits transactions into two independent classes:

- Class 0 contains read transactions due to cache misses and block loads
- Class 1 contains writeback requests, Write Invalidate requests, block stores, interrupt requests, and non-cached read/ write request.

Transactions in each class are strongly ordered by the interconnect. As a UltraSPARC-I Bus master, UltraSPARC-I also has a physically addressed coherent cache (E-cache), which participates in the MOESI cache coherence protocol, and responds to the interconnect for copyback/ invalidation requests.

As a UltraSPARC-I Bus slave, UltraSPARC-I responds to a non-cached read of its UltraSPARC-I Bus port ID. Notice that this could be a request generated by UltraSPARC-I itself as a master.

UltraSPARC-I is both an interrupter and an interrupt receiver. It has the capability to generate interrupt packets to other UltraSPARC-I Bus interrupt receivers and it can receive interrupts coming from other interrupters.

UPA Transactions Supported by UltraSPARC-I

Transactions initiated by UltraSPA RC-I

The UltraSPARC-I Bus transactions initiated by UltraSPARC-I are sent off through the system address bus. Four bits in the packet identifying the transaction type are encoded according to the UltraSPARC-I Bus definition of the corresponding transaction.

1. Read To Share (P_RDS_REQ)

   This coherent read with intent of sharing is issue by UltraSPARC-I due to a load miss.

2. Read to Share Always (P_RDSA_REQ)

   This coherent read with intent to share "always" is issued by UltraSPARC-I due to an external cache miss generated by an instruction fetch.

3. Read to Own (P_RDO_REQ)

   This coherent read with invalidate is generated by UltraSPARC-I due to a store miss, a store hit on a share line, or a read with intent to write for merging partial writes such as for read-modify-writes.

4. Read to Discard (P_RDD_REQ)

   This coherent read with no intent to cache the data is issued by UltraSPARC-I during block loads.

5. Writeback (P_WRB_REQ)

   This writeback request is generated when a dirty victimized block from the external cache must be written back to its home location. A writeback is associated with a prior coherent read transaction to the same E-cache location.

6. Write Invalidate ((P_WRI_REQ)

   This coherent write and invalidate request is generated by UltraSPARC-I during a block store (the version with invalidate).

7. Interrupt (P_INT_REQ)

   Interrupt transaction request packet. Generated by UltraSPARC-I for delivering a packetized interrupt consisting of a 64 byte block of data to the destination (see ASI Registers definition in Programmer's Reference Manual).

8. Non cached Read (P_NCRD_REQ)

   This read is issued when a load or a block load is issued to a non-cacheable location. 1, 2, 4, 8, or 16 bytes can be read with this transaction.

9. Non cached Block Read (P_NCBRD_REQ)

   This transaction is used when a block read (64 bytes) is made to a non-cacheable location.

10. Non cached Block Write (P_NCBWR_REQ)

    Generated by UltraSPARC-I when a block write (64 bytes) is made to a non-cacheable location.

System transactions accepted by UltraSPARC-I

1. Invalidate (S_INV_REQ)

   Invalidate request from the UltraSPARC-I Bus interface to UltraSPARC-I following a Read To Own (P_RDO_REQ) or Write Invalidate (P_WRI_REQ) request for a block from another UltraSPARC-I Bus port.

2. Copyback (S_CPB_REQ)

   Copyback request from the UltraSPARC-I Bus interface to UltraSPARC-I following a Read To Share (P_RDS_REQ) or Read To Share Always (P_RDSA_REQ) request for a block from another UltraSPARC-I Bus port.

3. Copyback Invalidate (S_CPI_REQ)

   Copyback and Invalidate request from the UltraSPARC-I Bus interface to UltraSPARC-I in response to a Read To Own (P_RDO_REQ) request for a block from another UltraSPARC-I Bus port.

4. Copyback To Discard (S_CPD_REQ)

   This is sent at the UltraSPARC-I Bus interface to UltraSPARC-I in order to service a Read To Discard (P_RDD_REQ) issued by another UltraSPARC-I Bus port. Notice that the "other" UltraSPARC-I Bus port could be UltraSPARC-I itself which would generate a data loopback. This transaction does not generate a state change for the E-cache and does not require a flush of the store buffer tag check.

5. Non-Cached Read (P_NCRD_REQ)

   This is the only slave read transaction that should be sent to UltraSPARC-I. UltraSPARC-I responses to this request by sending the value of its UltraSPARC-I Bus port ID to the UltraSPARC-I Bus data bus. The transaction starts as a P_NCRD_REQ from a UltraSPARC-I Bus master (which could be UltraSPARC-I itself), is forwarded by the UltraSPARC-I Bus interface to UltraSPARC-I, UltraSPARC-I replies through a P_RAS, a S_SRS is issued at the UltraSPARC-I Bus interface to drive the data on the UltraSPARC-I Bus bus, and finally the requesting master gets the data when a S_RAS is issued at the UltraSPARC-I Bus interface.

   Note: Do we ACK other forwarded transactions so that the system does not hang?

Responses to Transactions Initiated by the System (P_Reply)

P_reply is an acknowledgment from UltraSPARC-I to the System, in response to a request that system sent to UltraSPARC-I previously. There are five unidirectional (output only) pins on UltraSPARC-I connected directly to system.

1. Idle (P_IDLE)

   This is the default state of the wires. It indicates no reply.

2. Non Existent Block (P_NXB)

   Reply by UltraSPARC-I indicating that the requested block from a snoop does not exist in the external cache (only set when DTAGs are not present).

3. Read Acknowledge Single (P_RAS)

   16 bytes of read data is ready in the output data queue on the UDB. Sent following a single non-cacheable read request from a UltraSPARC-I Bus port (reply to P_NCBRD_REQ).

4. Coherent Read Acknowledge for a Block (P_CRAB)

   64 bytes of read data is ready in the UDB output data queue. Sent following a coherent read from another UltraSPARC-I Bus port (reply to P_RDS_REQ, P_RDSA_REQ, P_RDD_REQ, P_RDO_REQ).

5. Interrupt Acknowledge (P_IAK)

   UltraSPARC-I sends a P_IAK to acknowledge that the interrupt transaction delivered by system has been serviced. This implies that there is room on the UDB for another interrupt request and its 64 bytes of data.

6. Invalidate Acknowledge (P_IVAK)

   UltraSPARC-I acknowledges that the invalidate request from system(S_INV_REQ) has been serviced and that there is room on the UDB for another S_REQ transaction from system.

7. Reserved (P_RSVD)

   Reserved for future use.

TABLE 10: P_REPLY Encodings

P_REPLY	Name	Reply to Which Transaction	Class Type<3:0>
P_IDLE (single)	Idle	Default State	x 0000
P_WAB (single)	Write Ack Block	Any Block write request	C 0001
P_WAS (single)	Write Ack Single	P_NCWR_REQ	C 0010
P_RASB (single)	Write Ack Single/Block	MID, Size maintained by SC	C 0011
P_FERR (single)	Fatal Error	Address Parity Error / other Fatal	X 0100
P_RERR (two)	Read Data Error	Non-Cached slave read request	C 0101
Reserved			C 0110
P_NXB	Non-Existent Block	Copyback-invalidate S_REQ	C 0111
P_RAS (two)	Read Ack Single	P_NCRD_REQ	C 1000
P_RAB (two)	Read Ack Block	P_NCBRD_REQ	C 1001
P_CRAB (two)	Coherent Read Ack Block	Copyback S_REQ	C 1010
P_RABC (two)	Read Ack Block Coherent	Coherent P_REQ to UPA slave	C 1011

TABLE 10: P_REPLY Encodings (Continued)

P_REPLY	Name	Reply to Which Transaction	Class Type<3:0>
P_IAK (two)	Interrupt Acknowledge	P_INT_REQ	C 1100
P_IVAK (two)	Invalidation Acknowledge	S_INV_REQ	C1101
P_RERRC (two)	Read Data Error Coherent	Coherent P_REQ to UPA slave	C 1110
P_RTO (two)	Read Time Out	Non-chaced slave read request	C 1111

System Responses (S_REPLY) Due to a Transaction Request (P_REQ) or an Acknowledgment (P_REPLY) from UltraSPARC-I.

This is also a unidirectional point-to-point connection between system and UltraSPARC-I.

1. Idle (S_IDLE)

   This is the default state of the wires. It indicates no reply.

2. Read Time-out (S_RTO)

   This system reply is a forwarding of the Read Time Out (P_RTO) reply from the slave UltraSPARC-I Bus port that UltraSPARC-I tried to access. Notice that Time-out on writes are reported asynchronously via interrupt by the detecting slave UltraSPARC-I Bus port.

3. Error (S_ERR)

   This is asserted by system if the situations described in the UltraSPARC-I Bus spec Sec 3-9 occur.

4. Write Acknowledge Single (S_WAS)

   This is generated by system following a non-cacheable write request from UltraSPARC-I (P_NCWR_REQ). It causes 16 bytes of data from UDB to be put on the UltraSPARC-I Bus data bus.

5. Write Acknowledge Block (S_WAB)

   This is generated by system following a non-cacheable block store (P_NCBWR_REQ), a writeback request (P_WRB_REQ), or a write invalidate request during a block store with invalidate (P_WRI_REQ). It causes 64 bytes of data to be put on the UltraSPARC-I Bus data bus.

6. Ownership Acknowledged Block (S_OAK)

   Generated by system when UltraSPARC-I wants permission to write to a block that is already in the E-cache. No data transfer occurs.

7. Read Block Unshared Acknowledge (S_RBU)

   System commands the input data queue of UDB to accept 64 bytes of unshared or non-cached data from the UltraSPARC-I Bus data bus. This is a response to a P_RDS_REQ, a P_RDO_REQ, or a P_NCBRD_REQ.

8. Read Block Shared Acknowledge (S_RBS)

   System commands the input data queue of UDB to accept 64 bytes of shared data from the UltraSPARC-I Bus data bus. This is a response to a P_RDS_REQ or a P_RDSA_REQ.

9. Read Acknowledge Single (S_RAS)

   In response to a non cacheable read request from UltraSPARC-I (P_NCRD_REQ), System commands UDB to accept 16 bytes of data from the UltraSPARC-I Bus data bus.

10. Read Single Acknowledge (S_SRS)

    System commands UDB to drive 16 bytes of data onto the UltraSPARC-I Bus data bus. This follows a P_RAS from UltraSPARC-I which indicated that the data was ready in UDB.

11. Read Block Acknowledge (S_SRB)

    System commands UDB to drive 64 bytes of data onto the UltraSPARC-I Bus data bus. This follows a P_RAB from UltraSPARC-I which indicated that the data was ready in UDB. This is used for the normal slave read sequence: P_REQ -> P_RAB -> S_SRB.

12. Copyback Read Block Acknowledge (S_CRAB)

    System commands UDB to drive 64 bytes of copyback data onto the UltraSPARC-I Bus data bus. This follows a P_CRAB from UltraSPARC-I which indicated that the copyback data was ready in UDB.

13. Interrupt Write Block Acknowledge (S_SWIB)

    System commands UDB to accept 64 bytes of interrupt data from the UltraSPARC-I Bus data bus. In parallel the packet P_INT_REQ that was originally sent by the interrupting UltraSPARC-I Bus port is sent to UltraSPARC-I on the UltraSPARC-I Bus address bus.

14. Writeback Cancel Acknowledge (S_WBCAN)

    System will generate this if a previous writeback by UltraSPARC-I (P_WRB_REQ) needs to be cancelled.

15. Interrupt NACK (S_INAK)

    This is generated by system if the receiver of an interrupt that UltraSPARC-I sends out (through P_INT_REQ) cannot accept another interrupt packet at the moment. This reply effectively removes the interrupt packet from the UDB queue (software should retry later). This is the only transaction that is NACK'ed by system. A S_INAK sets a bit in an ASI register on UltraSPARC-I (see programmer's reference manual)

16. Ignored S_REPLYs (S_SWB and S_SWS)

    These two S_REPLYs are ignored by UltraSPARC-I (they should not occur).

TABLE 11: S_REPLY Encodings

S_REPLY	Name	Reply to Which Transaction	Class Type<3:0>
S_IDLE	Idle	Default State	x 0000
S_ERR	Error	Report Error to master	C 0001
S_CRAB	Coherent Read Ack block	To slave for P_CRAB reply	C 0010
S_WBCAN	Writeback Cancel	To master for P_WRB_REQ	C 0011

TABLE 11: S_REPLY Encodings

S_REPLY	Name	Reply to Which Transaction	Class Type<3:0>
S_WAS	Write Ack Single	To master for P_NCWR_REQ	C 0100
S_WAB	Write Ack Block	To master for any block write	C 0101
S_OAK	Ownership Ack	To master for P_RDO_REQ	C 0110
S_INAK	Interrupt Nack	To master for P_INT_REQ	C 0111
S_RBU	Read Block Ack Unshared	To master for any block read	C 1000
S_RBS	Read block Ack Shared	To master for coherent shared read	C 1001
S_RAS	Read Ack Single	To master for P_NCRD_REQ	C 1010
S_RTO	Read Time Out	To master, forwarding of P_RTO	C 1011
S_SWS	Slave Write Single	Write 16 bytes data to slave	0 1100
S_SWB	Slave Write Block	Write 64 bytes data to slave	0 1101

Interaction with the E-cache and Data Buffer

Overview

External cache accesses, although asynchronous with respect to other instructions (e.g. ALU operations), are closely coupled to the pipeline. Full throughput to the external cache is supported and can make the E-cache look like a very large D-cache. The micro architecture used to support this consists of the load buffer, dual ported tags, separate address busses for tag and data, etc.

The Data Buffer chip isolates the system data bus from UltraSPARC-I (Figure 1-16). It allows data transfers between UltraSPARC-I and the memory system (e.g. non-cacheable stores) or I/O and between the E-cache and the memory system (e.g. writebacks) to occur much more rapidly since system arbitration and system throughput are hidden by the internal buffering of the UDB. Overlapping of transactions is also possible which increases overall bandwidth. Interrupt "packets" are also handled by the UDB. ECC bits are generated and checked by the UDB.

More details regarding the E-cache and the Data Buffer are given in the following paragraphs.

The external cache consists of two parts:

- the *E$TAG RAM* which contains the physical tags of the cached lines and three bits of state information and,
- the *E$DATA RAM* which contains the actual data for each cache line.

Both parts can be built out of commodity RAMs. The parts operate synchronously with UltraSPARC-I (Synchronous Static RAMs). The external cache sizes supported by UltraSPARC-I are: 512K, 1M, 2M and 4 Mbytes. The size of the cache is established at boot time by software.

Each byte in the RAMs is accompanied by a parity bit (three bits for the tags and 16 bits for data).

The clients for the external cache are UltraSPARC-I and the data buffer chip. More specifically for UltraSPARC-I they are the load buffer, the store buffer, the prefetch unit, and the data buffer. Loads that miss the data cache are sent to the E-cache based on the fact that the working set which was too

large for the D-cache may fit in the E-cache. All cacheable stores go to the E-cache (the D-cache is write-through) not necessarily in order with respect to load accesses. All I-cache misses generate a request for the E-cache. The data buffer chip returns data from main memory during an E-cache miss, or a load to non-cacheable locations. Writebacks (the process of writing a dirty line back to memory before a fill), generate data transfers from the E-cache to the data buffer, controlled entirely by the cpu. Copybacks (responses to snoop hits) also generate transfers from the E-cache to the UDB.

Among the clients of the external cache, the request for the second 16-byte of data from the I-cache has the highest priority, followed by the data buffer, the load buffer, the store buffer and the I-cache/ prefetch unit. This priority is dynamically modified based on the number of entries in the store buffer and based on I-cache request (i.e. I-cache request get higher priority the second time around). Details of the arbitration to the E-cache are described more thoroughly in the MIU specs.

The UDB has a four entry by 16 bytes read buffer that can hold a 64 byte line coming from main memory due to an E-cache read miss or a non-cacheable read. The outgoing buffer, i.e. the buffers receiving data from the UltraSPARC-I side and sending it to the rest of the system, is divided into three parts. There is a8 X 16 byte writeback buffer, an 8 X 16 byte non-cacheable store buffer, and a 4 X 16 byte snoop buffer. Notice that the writeback buffer can be snooped. Consequently, internal bypass is provided to send the writeback data to the port requesting the snoop on the interconnect. Three 64-bit registers are provided to hold an incoming Mondo Vector, while three more are provided for Mondo Vector send.

LOGICAL TIMING DIAGRAMS

This Section describes the logical timing for the transactions occurring between UltraSPARC-I, the external cache, and the data buffer. The diagrams are based on a clock with a 50% duty cycle. The transitions represented in the diagrams show what is seen at the pins of UltraSPARC-I. The position of the transitions relative to the clock transitions is correct but not drawn to scale (i.e. a set up time of 1ns is represented by showing the transition of the incoming signal changing slightly before the rising edge of the clock).

Coherent Read Hit

Coherent reads that hit the E-cache are represented in *Figure 5*. With UltraSPARC-I there is no difference between burst reads and two consecutive reads, the signals used for a single read are simply duplicated for each subsequent read.

The timing diagram shows three consecutive reads that hit the E-cache. The control signals ($\overline{\text{TWE}}$, $\overline{\text{TOE}}$) and the address for the tag read (ECAT) as well as the control signals ($\overline{\text{DWE}}$, $\overline{\text{DOE}}$) and the address for the data (ECAD) are shown to transition shortly after the rising edge of the clock. Two cycles later, the data for both the tag read and data read is back at the pins of the CPU shortly before the next rising edge (meets set up time and clock skew). Notice that the reads are fully pipelined and thus full throughput is achieved (there are three requests made before the data of the first request comes back and the latency of each request is three cycles).

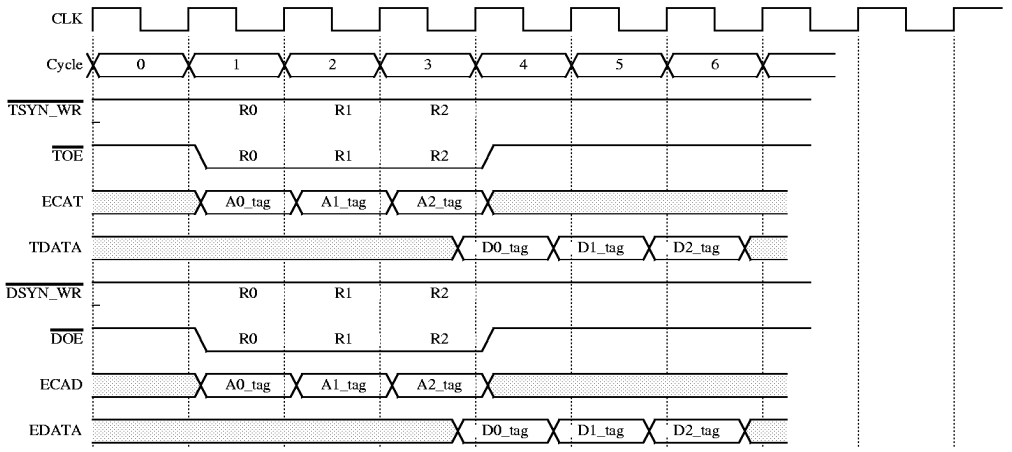

Figure 5. Coherent Read Hit Timing

Coherent Write Hits

Writes to the external cache are processed through independent tag and data transactions. First the tag and the state bits of the E-cache line corresponding to the write are read. If the access is a hit and the state is exclusive or modified, the data is written to the data RAM.

In the timing diagram shown in *Figure 6*, we show three consecutive write hits to M state lines. Access to the first tag (D0_tag) is started by asserting $\overline{\text{TWE}}$ and $\overline{\text{TOE}}$ and by sending the tag address (A0_tag). In the cycle after the tag data (D0_tag) comes back, it is determined by UltraSPARC that the access is a hit and that the line is in M state (Modified). In the next clock, a request is made to write the data. The data address is presented on the ECAD pins in the cycle after the request (cycle 7 for W0) and the data is sent in the following cycle (cycle 8), as shown in *Figure 6*. Separating the address and the data by one cycle reduces the turn around penalty when reads are immediately followed by writes (discussed in *"Coherent Read Followed by a Coherent Write" on page 25*).

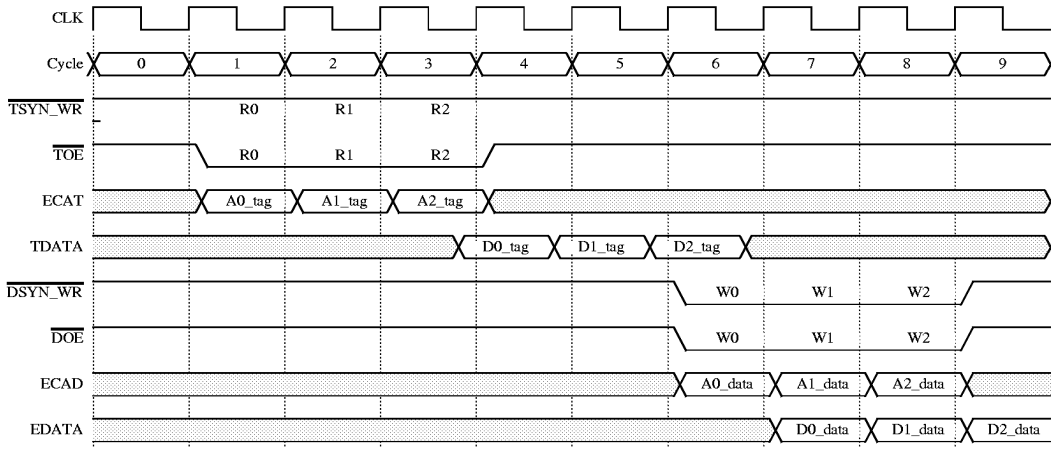

Figure 6. Coherent Write Hit to M State Line

If the line is in exclusive state then the tag is updated to Modified at the same time as the data is written as shown below.

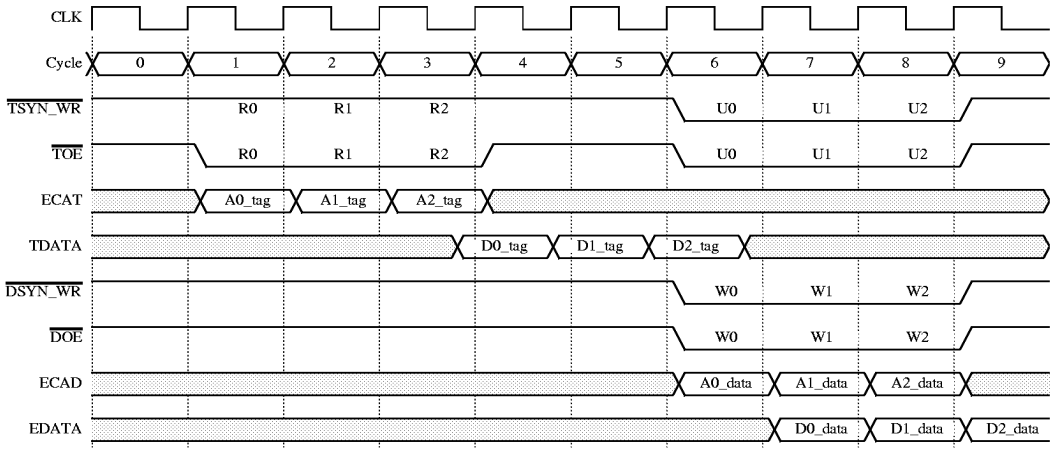

Figure 7. Coherent Writes with E to M Updates

Otherwise, the tag port is available for a tag check of a younger store during the data write. In the timing diagram shown in *Figure 6*, the store buffer is empty when the first write request is made. That is why there is no overlap between the tag accesses and the write accesses. In normal operation the tag access for one write can be done in parallel with the data write of previous write. This independence of the tag and data buses make the peak store bandwidth as high as the load bandwidth (one per cycle). The over-

lap of tag and data accesses is shown in *Figure 7*. The data for three previous writes (W0, W1 and W2) is written while three tag accesses (reads) are made for three younger stores (R3, R4 and R5).

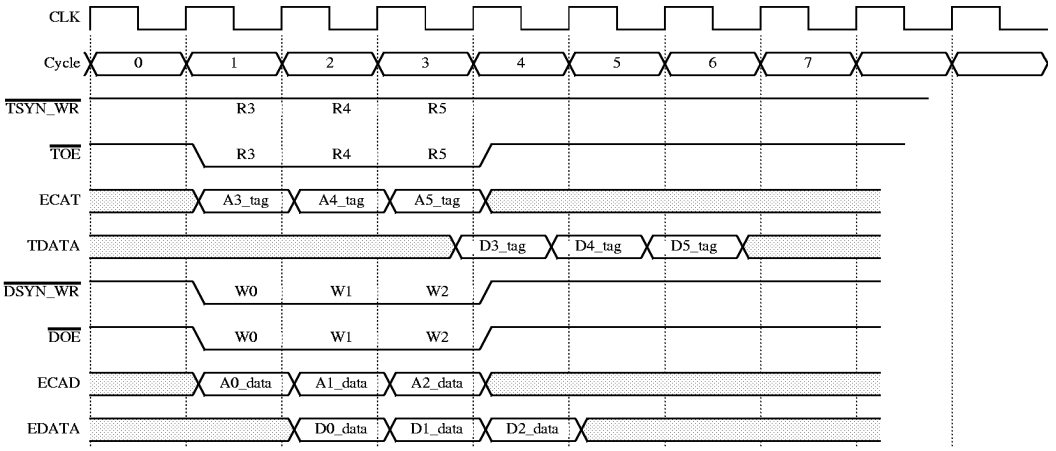

Figure 8. Overlap Between Tag Access and Data Write for Coherent Writes

If the line is in Shared or Owned state, then a read for ownership is performed before writing the data. If the access is a miss then a line is victimized and the data is written after the new line is brought in (discussed in a later section).

Coherent Read Followed by a Coherent Write

When a read is made to the E-cache, the three cycle latency causes the data bus to be busy two cycles after the address appears at the pins. For a processor without *delayed writes*, writes have to be held for two cycles in order to avoid collisions between the data of the write and the data coming back from the read. Additionally, an extra dead cycle is necessary to switch the driver of the E-cache data bus from the SRAMs to UltraSPARC-I (electrical considerations). UltraSPARC-I uses a one-deep write buffer in the data SRAMs to reduce the *turn around* penalty to two cycles (going from reads to writes). The data of a write is sent one cycle after the address (*Figure 9*). Notice that there is no penalty for going from writes to reads.

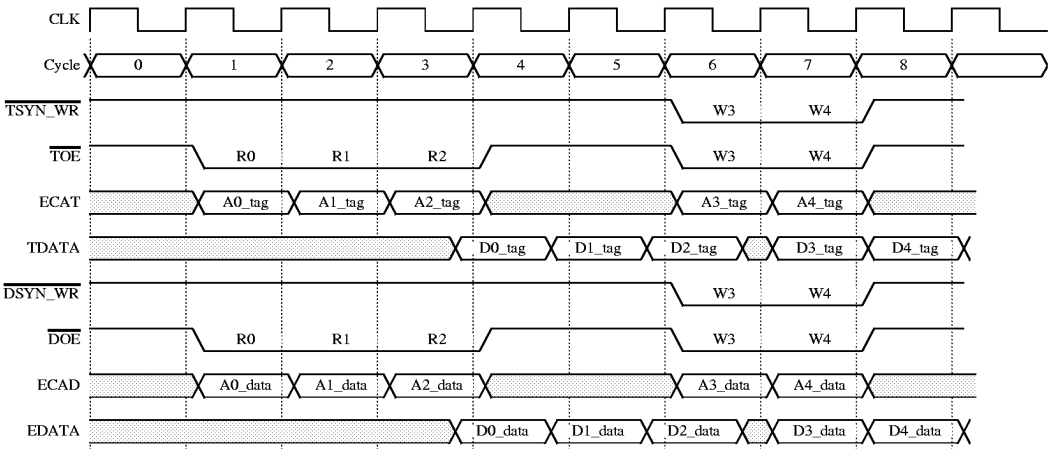

Figure 9. Reads Followed by Writes; Turn Around Penalty

In *Figure 9* we show the two cycle penalty between reads and writes. The figure represents three reads followed by two writes and two tag updates.The two cycle penalty applies to both tag accesses and data accesses (two dead cycles between A2_tag and A3_tag as well as between A2_data and A3_data).

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings [1]

Symbol	Parameter	Rating	Units
V_{CC}	Supply voltage range	0 to 4.0	V
V_I	Input voltage range [2]	-0.5 to V_{CC} + 0.5	V
V_O	Output voltage range	-0.5 to V_{CC} + 0.5	V
I_{IK}	Input clamp current (V_I < 0 or V_I > V_{CC})	±20	mA
I_{OK}	Output clamp current (V_O < 0 or V_O > V_{CC})	±50	mA
	Current into any output in the low state	50	mA
T_{STG}	Storage temperature	-40 to 150	°C

1. Operation of the device at values in excess of those listed above will result in degradation or destruction of the device. All voltages are defined with respect to ground. Functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions "is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
2. Unless otherwise noted, all voltages are with respect at V_{SS}.

Recommended Operating Conditions

Symbol	Parameter		Min	Typ	Max	Units
V_{CC}	Supply voltage		3.2	3.3	3.465	V
V_{SS}	Ground		–	0	–	V
V_{IH}	High-level input voltage	All except CLK	2.0	–	V_{CC} + 0.3	V
		CLK	2.4	–	V_{CC} + 0.3	V
V_{IL}	Low-level input voltage	All except CLK	-0.3	–	0.8	V
V_{IL}		CLK	-0.3	–	1.6	V
I_{OH}	High-level output current		–	–	-4.0	mA
I_{OL}	Low-level output current		–	–	8.0	mA
T_J	Operating junction temperature		–	–	105	°C
T_A	Operating ambient temperature		0	–	[1]	°C

1. Maximum ambient temperature is limited by air flow such that the maximum junction temperature does not exceed T_J.

DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{OH}	High-level output voltage	V_{CC} = Min, I_{OH} = Max	2.4	–	–	V
V_{OL}	Low-level output voltage[1]	V_{CC} = Min, I_{OL} = Max	–	–	0.4	V
V_{IH}	High-level input voltage (except CLKA, CLKB, SDBCLKA, SDB-CLKB, LOOPCAP)	V_{CC} = Max	2.0	–	–	V
	High -level input voltage (CLKA, CLKB, SDBCLKA, SDBCLKB)	V_{CC} = Max	2.4	–	–	V
V_{IL}	Low-level input voltage (except CLKA, CLKB, SDBCLKA, SDBCLKB, LOOPCAP)	V_{CC} = Min	–	–	0.8	V
	Low-level input voltage (CLKA, CLKB, SDBCLKA, SDBCLKB, LOOPCAP)	V_{CC} = Min	–	–	0.8	V
V_{IH}, V_{IL}	High-level, low-level input voltage for LOOPCAP	Pin should be grounded	–	0	–	V
I_{DDD}	Supply current	V_{CC} = Max, freq = 143 MHz	–	9	–	A
		V_{CC} = Max, freq = 166 MHz	–	10	–	A
I_{DDQ}	Quiescent supply current	V_{CC} = Max, Freq = 0 MHz, $V_I = V_{SS}$ or V_{CC}	–	–	TBD	mA
I_{OZ}	High-impedance output current[2]	V_{CC} = Max, $V_O = V_{CC}$	-10	–	10	µA
		V_{CC} = Max, $V_O = V_{SS}$	-10	–	10	µA
I_I	Input current	V_{CC} = Max, $V_O = V_{SS}$ to V_{CC}	-10	–	10	µA
C_I	Input capacitance[3]		–	5	–	pF
C_O	Output capacitance		–	10	–	pF

1. STOP_CLK has no V_{OL} specification.
2. Only bidirectional lines can be three-state, output-only cannot three-state. All bidirectional lines will be three-stated when $\overline{\text{RESET}}$ is held LOW and SRAM_TEST is held high.
3. This specification is provided as an aid to board design. This specification is not assured during manufacturing testing.

AC Characteristics - Signal Timing (Except Clock and JTAG)[1]

Symbol	Parameter	Signals	Conditions	143 MHz		166 MHz		Units
				Min	Max	Min	Max	
t_{SU} (1)	Input setup time to CLK	SYSADR[35:0], ADR_VLD, SCLK_MODE, NODE_RQ[2:0], SC_RQ, S_REPLY[3:0], DATA_STALL, EDATA[127:0], EDPAR[15:0], TDATA[24:0], TPAR[3:0], $\overline{\text{RESET}}$ [2], $\overline{\text{XIR}}$		2.7	–	2.4	–	ns
t_H(1)	Input hold time to CLK			0.9	–	0.9	–	ns

AC Characteristics - Signal Timing (Except Clock and JTAG) [1]

Symbol	Parameter	Signals	Conditions	143 MHz Min	143 MHz Max	166 MHz Min	166 MHz Max	Units
t_{PD}(1)	Output delay from CLK	SYSADR[35:0], ADR_VLD, EDATA[127:0], EDPAR[15:0], TDATA[24:0], TPAR[3:0], BYTEWE, ECAD[17:0], ECAT[15:0], DSYN_WR, DOE, TSYN_WR, TOE, P_REPLY[4:0], NODEX_RQ	I_{OL} = 8 mA I_{OH} = -4 mA C_L = 35 pF V_{LOAD} = 1.5V	–	5.5	–	4.7	ns
t_{OH}(1)	Output hold time from CLK			0.2	–	0.2	–	ns
t_{PD}(2)	Output delay from CLK	STOP_CLK, EPD		–	TBD	–	TBD	ns
t_{OH}(2)	Output hold time from CLK	EPD		TBD	–	TBD	–	ns
t_{LOCK}	PLL acquisition time			7.0		6.0		µs
				1000		1000		cycle
t_{SKEW}	SDBCLK skew to CLK [3]			0	1.3	0	1.3	ns

1. All timing requirements are specified with PLL enabled.
2. $\overline{\text{RESET}}$ is asserted asynchronously but deasserted synchronously to the system clock.
3. SDBCLK is before CLK as shown in Figure xxx.

AC Characteristics - Clock Timing

Symbol	Parameter	143 MHz Min	143 MHz Typ	143 MHz Max	166 MHz Min	166 MHz Typ	166 MHz Max	Units
t_{CYC} (CLK)	Processor clock cycle time [1]	7	–	–	6	–	–	ns
t_W(CLK)	Processor clock duty cycle [1]	40	50	60	40	50	60	%
t_{SLEW}(CLK)	Clock input slew rate [1]	TBD	–	–	TBD	–	–	V/ns
t_{CYC} (TCK)	TCK clock cycle time	TBD	–	TBD	TBD	–	TBD	ns
t_W(TCK)	TCK clock duty cycle	TBD	TBD	TBD	TBD	TBD	TBD	%
t_{CYC} (SDBCLK)	SDBCLK clock cycle time (DIViDE BY 2 MODE)	t_{CYC} (CLK) x2	–	–	t_{CYC} (CLK) x2	–	–	ns
t_{CYC} (SDBCLK)	SDBCLK clock cycle time (DIViDE BY 3 MODE)	t_{CYC} (CLK) x3	–	–	t_{CYC} (CLK) x3	–	–	ns
t_W (SDBCLK)	SDBCLK duty cycle	40	50	60	40	50	60	%
t_W ($\overline{\text{RESET}}$)	RESET pulse width LOCK MODE (See figure x)	10	–	–	10	–	–	ns
t_W ($\overline{\text{RESET}}$)	RESET pulse width BYPASS MODE (See figure x)	t_{CYC} (CLK) x3	–	–	t_{CYC} (CLK) x3	–	–	ns

1. This is for the PLL enabled.

AC Characteristics - JTAG Timing

Symbol	Parameter	Signals	Conditions	143 MHz			166 MHz			Units
				Min	Typ	Max	Min	Typ	Max	
t_{SU} ($\overline{TRST}$)	Input setup time to TCK	$\overline{TRST}$		10.0	–	–	10.0	–	–	ns
t_{SU} (TDI)	Input setup time to TCK	TDI		–	1.5	–	–	1.5	–	ns
t_{SU} (TMS)	Input setup time to TCK	TMS		–	3.0	–	–	3.0	–	ns
t_H($\overline{TRST}$)	Input hold time to TCK	$\overline{TRST}$		–	0	–	–	0	–	ns
t_H(TDI)	Input hold time to TCK	TDI		–	1.0	–	–	1.0	–	ns
t_H(TMS)	Input hold time to TCK	TMS		–	1.0	–	–	1.0	–	ns
t_{PD}(TDO)	Output delay from TCK [1]	TDO	I_{OL} = 8 mA I_{OH} = -4 mA	–	8.0	–	–	8.0	–	ns
t_{OH}(TDO)	Output hold time from TCK [1]	TDO	C_L = 35 pF V_{LOAD} = 1.5V	–	–	TBD	–	–	TBD	ns

1. TDO is referenced from falling edge of TCK.

AC Characteristics - T_{PD} (Output) Capacitive Derating Factor [1]

Symbol	Parameter	143 MHz			166 MHz			Units
		Min	Typ	Max	Min	Typ	Max	
t_{PD}	Capacitive derating factor	0.2	0.4	0.55	0.2	0.3	0.4	ns/pf

1. Derating factors are shown to aid in board design. This specification is not verified during manufacturing testing.

Thermal Resistance vs. Air Flow [1]

Symbol	Air Flow (ft/min)				Units
	100	200	300	500	
$Theta_{JA}$	TBD	TBD	TBD	TBD	(°C/W)

1. TJ can be calculated by: $T_J = T_A + P_D \times Theta_{JA}$.
Thermal resistance measured using UltraSPARC-I heatsink. P_D = Power Dissipation.

PARAMETER MEASUREMENT INFORMATION

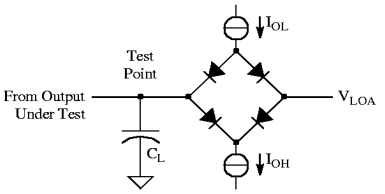

Figure 10. Load Circuit

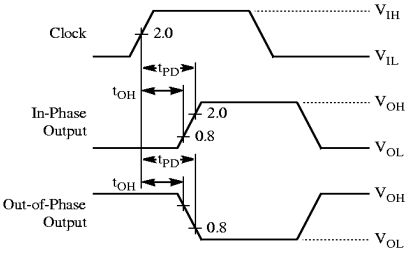

Figure 11. Voltage Waveforms - Propagation Delay Times

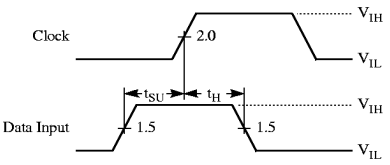

Figure 12. Voltage Waveforms - Setup and Hold Times

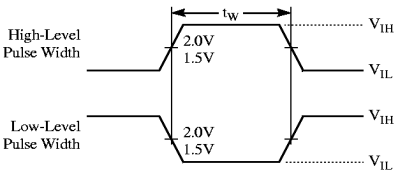

Figure 13. Voltage Waveforms - Clock Pulse Duration

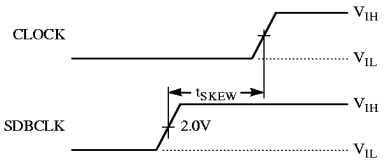

Figure 14. Voltage Waveforms - Clock Skew

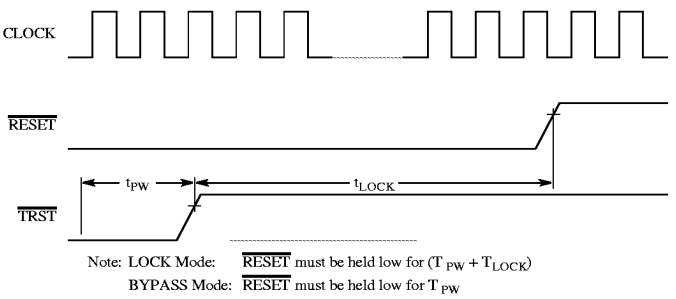

Figure 15. Reset Timing

PACKAGING INFORMATION

PBGA 521 Pin Assignment [1] [2] [3]

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name	Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
A5	VSSO	B30	VDDO	D23	EDATA[58]	F32	EDATA[111]	L33	EDATA[74]	S1	SYSADR[14]
A6	MISC_BIDIR[9]	C3	VSSC	D24	VDDO	F33	EDATA[108]	M1	NODEX_RQ	S2	SYSADR[15]
A7	MISC_BIDIR[5]	C4	TDO	D25	EDATA[51]	G1	SC_RQ	M2	ADR_VLD	S3	VSSC
A8	MISC_BIDIR[1]	C5	MISC_BIDIR[[14]	D26	EDATA[48]	G2	XIR_L	M3	VSSO	S4	VSSO
A9	EDATA[125]	C6	MISC_BIDIR[12]	D27	VSSO	G3	SDB_CEL	M4	VSSC	S5	SYSADR[17]
A10	EDATA[121]	C7	VSSO	D28	EDATA[22]	G4	SDB_UEL	M5	VDDC	S29	VSSC
A11	EDATA[120]	C8	MISC_BIDIR[2]	D29	spare	G5	SDB_UEH	M29	VSSC	S30	EDATA[12]
A12	EDATA[116]	C9	EDATA[127]	D30	VDDC	G29	EDATA[110]	M30	VDDO	S31	EDATA[13]
A13	EDATA[113]	C10	VDDO	D31	VSSC	G30	EDATA[109]	M31	EDATA[73]	S32	VSSO
A14	EDATA[93]	C11	EDATA[122]	D32	VDDO	G31	VSSO	M32	EDATA[47]	S33	EDPAR[1]
A15	EDATA[91]	C12	EDATA[118]	E1	VSSO	G32	EDPAR[13]	M33	EDATA[72]	T1	SYSADR[16]
A16	VDDC	C13	VSSO	E2	L5CLK	G33	EDATA[107]	N1	SYSADR[1]	T2	SYSADR[19]
A17	EDATA[87]	C14	EDATA[94]	E3	EPD	H1	S_REPLY[0]	N2	SYSADR[0]	T3	SYSADR[18]
A18	EDATA[86]	C15	VSSC	E4	VSS_QUIET	H2	DATA_STALL	N3	SYSADR[2]	T4	RAM_TEST
A19	EDATA[84]	C16	VDDO	E5	VDD_QUIET	H3	NODE_RQ[2]	N4	VDDO	T5	LOOPCAP
A20	EDATA[81]	C17	VDDC	E6	MISC_BIDIR[13]	H4	NODE_RQ[1]	N5	VDDC	T29	EDATA[11]
A21	EDATA[61]	C18	VSSC	E7	MISC_BIDIR[8]	H5	NODE_RQ[0]	N29	VDDC	T30	VDDO
A22	EDPAR7	C19	VSSO	E8	MISC_BIDIR[4]	H29	VSSC	N30	VDDC	T31	EDATA[9]
A23	EDATA[56]	C20	EDATA[82]	E9	MISC_BIDIR[0]	H30	EDATA[106]	N31	VSSO	T32	EDATA[10]
A24	EDATA[52]	C21	EDATA[63]	E10	VDDC	H31	VDDC	N32	EDATA[45]	T33	EDATA[8]
A25	EDATA[49]	C22	VDDO	E11	EDPAR[15]	H32	VDDO	N33	EDATA[46]	U1	SDBCLKA
A26	EDATA[29]	C23	EDATA[57]	E12	EDATA[119]	H33	EDATA[105]	P1	SYSADR[3]	U2	SDBCLKB
A27	EDATA[28]	C24	EDATA[54]	E13	EDPAR[14]	J1	S_REPLY[2]	P2	SYSADR[4]	U3	RESET_L
A28	EDATA[26]	C25	VSSO	E14	EDATA[112]	J2	S_REPLY[1]	P3	SYSADR[5]	U4	CLKA
A29	VSSO	C26	EDATA[30]	E15	VDDC	J3	VDDC	P4	SYSADR[6]	U5	VSS_PLL
B4	VDDO	C27	EDATA[27]	E16	EDPAR[11]	J4	VSSC	P5	VSSC	U29	VDDC
B5	MISC_BIDIR[11]	C28	EDATA[23]	E17	EDATA[90]	J5	S_REPLY[3]	P29	VSSC	U30	VSSC
B6	MISC_BIDIR[10]	C29	EDPAR[2]	E18	EDATA[89]	J29	VSSC	P30	EDPAR[5]	U31	VSSO
B7	MISC_BIDIR[6]	C30	EDATA[19]	E19	VDDC	J30	VSSO	P31	EDATA[44]	U32	EDATA[103]
B8	VDDO	C31	VSSO	E20	EDPAR[10]	J31	EDATA[104]	P32	VDDO	U33	EDATA[102]
B9	EDATA[126]	D2	VDDO	E21	EDATA[80]	J32	EDATA[79]	P33	EDATA[41]	V1	PLLBYPASS
B10	EDATA[124]	D3	VDDC	E22	VDDC	J33	EDATA[78]	Q1	SYSADR[9]	V2	TRST_L
B11	VSSO	D4	VSSC	E23	EDATA[59]	K1	P_REPLY[0]	Q2	VSSO	V3	CLKB
B12	EDATA[117]	D5	VDDC	E24	EDATA[55]	K2	P_REPLY[1]	Q3	SYSADR[7]	V4	SCLK_MODE
B13	EDATA[114]	D6	VDDO	E25	EDPAR[6]	K3	P_REPLY[2]	Q4	SYSADR[10]	V5	VDD_PLL
B14	VDDO	D7	MISC_BIDIR[7]	E26	EDATA[31]	K4	VSSC	Q5	SYSADR[8]	V29	EDATA[101]
B15	EDATA[92]	D8	MISC_BIDIR[3]	E27	EDATA[25]	K5	VSSC	Q29	EDATA[15]	V30	EDPAR[12]
B16	VSSC	D9	VSSO	E28	EDATA[20]	K29	VDDC	Q30	VSSO	V31	EDATA[100]
B17	VSSO	D10	VSSC	E29	spare	K30	VSSC	Q31	EDATA[42]	V32	VDDO
B18	EDATA[88]	D11	EDATA[123]	E30	EDATA[18]	K31	VDDO	Q32	EDATA[43]	V33	EDATA[99]
B19	EDATA87[85]	D12	VDDO	E31	EDATA[16]	K32	EDATA[76]	Q33	EDATA[40]	W1	SYSADR[20]
B20	VDDO	D13	EDATA[115]	E32	EDATA[17]	K33	EDATA[77]	R1	SYSADR[13]	W2	VDDO
B21	EDATA[62]	D14	EDATA[95]	E33	VSSO	L1	P_REPLY[3]	R2	SYSADR[11]	W3	SYSADR[21]
B22	EDATA[60]	D15	VSSO	F1	SDB_CEH	L2	VDDO	R3	VDDO	W4	SYSADR[23]
B23	VSSO	D16	VDDC	F2	TMS	L3	P_REPLY[4]	R4	SYSADR[12]	W5	SYSADR[22]
B24	EDATA[53]	D17	VSSC	F3	TCK	L4	VSSC	R5	VDDC	W29	EDATA[98]
B25	EDATA[50]	D18	VDDO	F4	TDI	L5	VDDC	R29	VDDC	W30	VSSO
B26	VDDO	D19	VSSC	F5	EXT_EVENT	L29	VDDC	R30	VDDC	W31	EDATA[97]
B27	EDPAR[3]	D20	EDATA[83]	F29	VSSC	L30	EDATA[75]	R31	VDDO	W32	EDATA[96]

PBGA 521 Pin Assignment (Continued) [1] [2] [3]

Pin	Pin Name	Pin	Pin Name	Pin	Pin Name	Pin	Pin Name	Pin	Pin Name	Pin	Pin Name
B28	EDATA[24]	D21	VSSO	F30	VDDO	L31	EDPAR[9]	R32	VSSC	W33	EDATA[71]
B29	EDATA[21]	D22	VSSC	F31	VDDC	L32	VSSO	R33	EDATA[14]	X1	SYSADR[24]
X2	SYSADR[25]	AA31	VSSO	AE4	TDATA[12]	AF8	VDDC	AG14	ECAD[9]	AH22	ECAT[6]
X3	VSSO	AA32	EDATA[37]	AE5	VDDC	AF9	VSSC	AG15	VDDO	AH23	VDDO
X4	SYSADR[26]	AA33	EDATA[36]	AE6	TDATA[18]	AF10	VSSO	AG16	VDDC	AH24	ECAT[13]
X5	SYSADR[27]	AB1	TDATA[3]	AE7	TDATA[21]	AF11	VDDC	AG17	VDDC	AH25	BYTEWE_L[0]
X29	VDDC	AB2	TDATA[4]	AE8	TDATA[24]	AF12	ECAD[3]	AG18	VSSO	AH26	VSSO
X30	VSSC	AB3	VDDO	AE9	VDDC	AF13	VDDO	AG19	ECAD[17]	AH27	BYTEWE_L[5]
X31	VDDO	AB4	TDATA[5]	AE10	DOE_L	AF14	VSSC	AG20	SDB_CNTL[4]	AH28	BYTEWE_L[10]
X32	EDATA[70]	AB5	TDATA[9]	AE11	ECAD[1]	AF15	ECAD[11]	AG21	VDDO	AH29	BYTEWE_L[14]
X33	EDATA[69]!	AB29	EDATA[33]	AE12	ECAD[2]	AF16	VSSO	AG22	VDDC	AH30	VSSO
Y1	SYSADR[28]	AB30	EDATA[34]	AE13	VSSC	AF17	VSSC	AG23	ECAT[9]	AJ5	VDDO
Y2	SYSADR[30]	AB31	EDATA[35]	AE14	VDDC	AF18	VDDC	AG24	VSSO	AJ6	TDATA[19]
Y3	SYSADR[31]	AB32	VDDO	AE15	VDDC	AF19	VDDO	AG25	ECAT[15]	AJ7	TDATA[22]
Y4	VDDO	AB33	EDPAR[4]	AE16	VSSC	AF20	SDB_CNTL[3]	AG26	VSSC	AJ8	TOE_L
Y5	SYSADR[32]	AC1	TPAR[0]	AE17	VDDC	AF21	ECAT[1]	AG27	VDDO	AJ9	TSYN_WR_L
Y29	EDATA[68]	AC2	TDATA[6]	AE18	ECAD[15]	AF22	VSSO	AG28	BYTEWE_L[11]	AJ10	ECAD[0]
Y30	EDPAR[8]	AC3	TDATA[7]	AE19	VSSC	AF23	ECAT[7]	AG29	BYTEWE_L[12]	AJ11	VDDC
Y31	EDATA[67]	AC4	VSSO	AE20	SDB_CNTL[2]	AF24	ECAT[12]	AG30	VDDC	AJ12	ECAD[5]
Y32	VSSO	AC5	TDATA[11]	AE21	ECAT[2]	AF25	VDDO	AG31	VDDO	AJ13	ECAD[8]
Y33	EDATA[66]	AC29	EDATA[5]	AE22	ECAT[5]	AF26	BYTEWE_L[3]	AH4	VSSO	AJ14	ECAD[10]
Z1	SYSADR[29]	AC30	VSSO	AE23	VSSC	AF27	BYTEWE_L[7]	AH5	TDATA[17]	AJ15	ECAD[13]
Z2	SYSADR[33]	AC31	EDATA[7]	AE24	ECAT[11]	AF28	VSSO	AH6	TPAR[2]	AJ16	ECAD[12]
Z3	SYSADR[34]	AC32	EDATA[6]	AE25	VDDC	AF29	BYTEWE_L[15]	AH7	TDATA[23]	AJ17	VSSC
Z4	VDD_PECL	AC33	EDATA[32]	AE26	BYTEWE_L[4]	AF30	VSSC	AH8	VSSO	AJ18	ECAD[14]
Z5	TDATA[0]	AD1	TDATA[8]	AE27	BYTEWE_L[9]	AF31	TEMP_SEN[0]	AH9	TPAR[3]	AJ19	SDB_CNTL[1]
Z29	EDATA[65]	AD2	TDATA[10]	AE28	BYTEWE_L[13]	AF32	VDDO	AH10	DSYN_WR_L	AJ20	ECAT[0]
Z30	VDDO	AD3	TPAR[1]	AE29	TEMP_SEN[1]	AG3	VDDC	AH11	VDDO	AJ21	ECAT[3]
Z31	EDATA[64]	AD4	TDATA[13]	AE30	VSSC	AG4	PM_OUT	AH12	ECAD[4]	AJ22	ECAT[8]
Z32	EDATA[39]	AD5	TDATA[15]	AE31	EDATA[0]	AG5	SPARE	AH13	ECAD[7]	AJ23	ECAT[10]
Z33	EDATA[38]	AD29	EDPAR[0]	AE32	EDATA[1]	AG6	VDDC	AH14	VSSO	AJ24	ECAT[14]
AA1	SYSADR[35]	AD30	EDATA[2]	AE33	VSSO	AG7	TDATA[20]	AH15	VSSC	AJ25	BYTEWE_L[1]
AA2	VSSO	AD31	VDDC	AF2	VSSO	AG8	VSSC	AH16	VSSC	AJ26	BYTEWE_L[2]
AA3	STOP_CLOCK	AD32	EDATA[3]	AF3	VDDC	AG9	VDDO	AH17	VDDO	AJ27	BYTEWE_L[6]
AA4	TDATA[1]	AD33	EDATA[4]	AF4	VSSC	AG10	VSSC	AH18	ECAD[16]	AJ28	BYTEWE_L[8]
AA5	TDATA[2]	AE1	VDDO	AF5	VSSC	AG11	VSSC	AH19	SDB_CNTL[0]	AJ29	VDDO
AA29	VDDC	AE2	TDATA[14]	AF6	VSSC	AG12	VSSO	AH20	VSSO		
AA30	VSSC	AE3	TDATA[16]	AF7	VDDO	AG13	ECAD[6]	AH21	ECAT[4]		

1. V_{DD}/V_{SS} Core: V_{DD}/V_{SS} for the Input, Core and Memory are tied together on the die to the same power plane. The V_{DD}/V_{SS} core pins are attached to these planes.
2. V_{DD}/V_{SS} Out: V_{DD}/V_{SS} outputs on the die have their own separate planes which are accessed through the V_{DD}/V_{SS}–OUT pins.
3. V_{DD}/V_{SS} Quite, PLL and PECL are bonded directly to the package pins.

PBGA 521 Package Dimensions