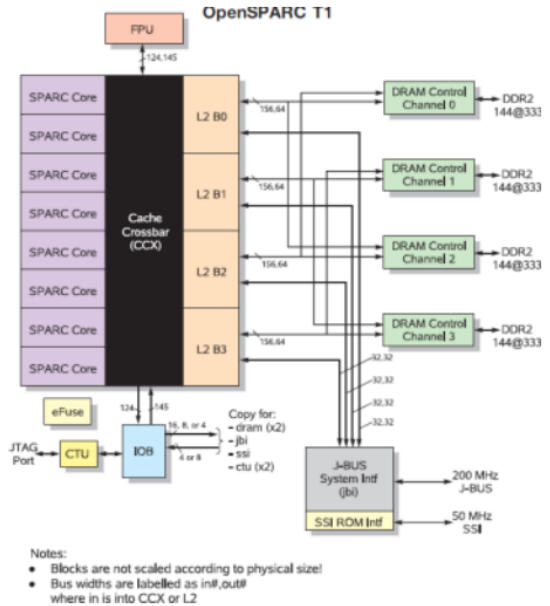
Open SPARC T1 Processor

System level block diagram



SPARC Core

Each SPARC core has hardware support for four threads. This support consists of a full register file (with eight register windows) per thread, with most of the address space identifiers (ASI), ancillary state registers (ASR), and privileged registers replicated per thread. The four threads share the instruction, the data caches, and the TLBs. Each instruction cache is 16 Kbytes with a 32-byte line size. The data caches are write through, 8 Kbytes, and have a 16-byte line size. The TLBs include an autodemap feature which enables the multiple threads to update the TLB without locking.

Each SPARC core has single issue, six stage pipeline. These six stages are:

1.Fetch

2.Thread Selection

3.Decode

4.Execute

5.Memory

6.Write Back

Each SPARC core has the following units:

1.Instruction fetch unit (IFU) includes the following pipeline stages – fetch, thread selection, and decode. The IFU also includes an instruction cache complex.

2. Execution unit (EXU) includes the execute stage of the pipeline.

3. Load/store unit (LSU) includes memory and writeback stages, and a data cache complex.

4. Trap logic unit (TLU) includes trap logic and trap program counters. 5. Stream processing unit (SPU) is used for modular arithmetic functions for crypto. 6. Memory management unit (MMU).

7. Floating-point frontend unit (FFU) interfaces to the FPU.

Instruction Fetch Unit

The thread selection policy is as follows – a switch between the available threads every cycle giving priority to the least recently executed thread. The threads become unavailable due to the long latency operations like loads, branch, MUL, and DIV, as well as to the pipeline stalls like cache misses, traps, and resource conflicts. The loads are speculated as cache hits, and the thread is switched-in with lower priority.

Instruction cache complex has a 16-Kbyte data, 4-way, 32-byte line size with a single ported instruction tag. It also has dual ported (1R/1W) valid bit array to hold cache line state of valid/invalid. Invalidates access the V-bit array, not the instruction tag. A pseudo-random replacement algorithm is used to replace the cache line.

There is a fully associative instruction TLB with 64 entries. The buffer supports the following page sizes: 8 Kbytes, 64 Kbytes, 4 Mbytes, and 256 Mbytes. The TLB uses a pseudo least recently used (LRU) algorithm for replacement. Multiple hits in the TLB are prevented by doing an autodemap on a fill.

Two instructions are fetched each cycle, though only one instruction is issued per clock, which reduces the instruction cache activity and allows for an opportunistic line fill. There is only one outstanding miss per thread, and only four per core. Duplicate misses do not issue requests to the L2-cache.

The integer register file (IRF) of the SPARC core has 5 Kbytes with 3 read/2 write/1 transport ports. There are 640 64-bit registers with error correction code (ECC). Only 32 registers from the current window are visible to the thread. Window changing in background occurs under the thread switch. Other threads continue to access the IRF (the IRF provides a single-cycle read/write access).

Execution Unit

The execution unit (EXU) has a single arithmetic logic unit (ALU) and shifter. The ALU is reused for branch address and virtual address calculation. The integer multiplier has a 5 clock latency, and a throughput of half-per-cycle for area saving. One integer multiplication is allowed outstanding per core. The integer multiplier is shared between the core pipe (EXU) and the modular arithmetic (SPU) unit on a round-robin basis. There is a simple non-restoring divider, which allows for one divide outstanding per SPARC core. Thread issuing a MUL/DIV will be rolled back and switched out if another thread is occupying the MUL/DIV units.

Load/Store Unit

The data cache complex has an 8-Kbyte data, 4-way, 16-byte line size. It also has single ported data tag. There is a dual-ported (1R/1W) valid bit array to hold cache line state of valid or invalid. Invalidates access the V-bit array but not the data tag. A pseudo-random replacement algorithm is used to replace the data cache line. The loads are allocating, and the stores are non-allocating. The data TLB operates similarly to the instruction TLB.

The load/store unit (LSU) has an 8 entry store buffer per thread, which is unified into a single 32 entry array, with RAW bypassing. Only a single load per thread outstanding is allowed. Duplicate requests for the same line are not sent to the L2-cache. The LSU has interface logic to interface to the CPU-cache crossbar (CCX). This interface performs the following operations:

■ Prioritizes the requests to the crossbar for floating-point operation (Fpops), streaming operations, I$ and D$ misses, stores and interrupts, and so on.

■ Request priority: imiss>ldmiss>stores,{fpu,stream,interrupt}.

■ Assembles packets for the processor-cache crossbar (PCX).
The LSU handles returns from the CPX crossbar and maintains the order for cache updates and invalidates.

Floating-Point Frontend Unit

The floating-point frontend unit (FFU) decodes floating-point instructions and it also includes the floating-point register file (FRF). Some of the floating-point instructions like move, absolute value, and negate are implemented in the FFU, while the others are implemented in the FPU. The following steps are taken when the FFU detects a floating-point operation (Fpop):

■The thread switches out.

■The Fpop is further decoded and the FRF is read.

■Fpops with operands are packetized and shipped over the crossbar to the FPU. ■The

computation is done in the FPU and the results are returned by way of the crossbar.

■Writeback completed to the FRF and the thread restarts.

Trap Logic Unit

The trap logic unit (TLU) has support for six trap levels. Traps cause pipeline flush and thread switch until trap program counter (PC) becomes available. The TLU also has support for up to 64 pending interrupts per thread.

Stream Processing Unit

The stream processing unit (SPU) includes a modular arithmetic unit (MAU) for crypto (one per core), and it supports asymmetric crypto (public key RSA) for up to a 2048-byte size key. It shares an integer multiplier for modular arithmetic operations. MAU can be used by one thread at a time. The MAU operation is set up by the store to control register, and the thread returns to normal processing. The MAU unit initiates streaming load/store operations to the L2-cache through the crossbar, and compute operations to the multiplier. Completion of the MAU can be checked by polling or issuing an interrupt.

CPU-Cache Crossbar

The eight SPARC cores, the four L2-cache banks, the I/O Bridge, and the FPU all interface with the crossbar. FIGURE displays the crossbar block diagram. The CPU-cache crossbar (CCX) features include:

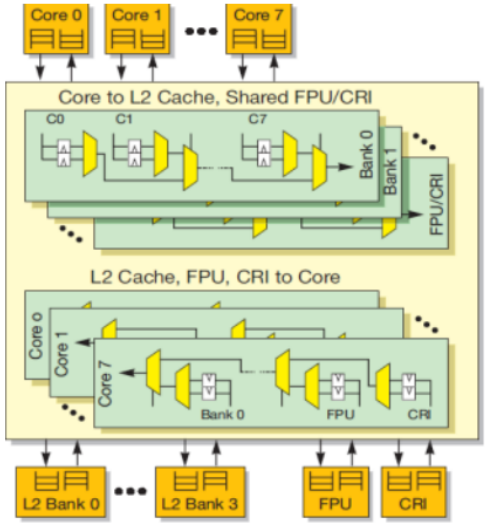
■Each requester queues up to two packets per destination.

■Three stage pipeline – request, arbitrate, and transmit.

■Centralized arbitration with oldest requester getting priority.

■Core-to-cache bus optimized for address plus doubleword store.

■Cache-to-core bus optimized for 16-byte line fill. 32-byte I$ line fill delivered in two back to-back clocks.



CCX Block Diagram

Floating-Point Unit

A single floating-point unit (FPU) is shared by all eight SPARC cores. The shared floating point unit is sufficient for most commercial applications in which typically less than one percent of the instructions are floating-point operations.

L2-Cache

The L2-cache is banked four ways, with the bank selection based on the physical address bits 7:6. The cache is 3-Mbyte, 12-way set-associative with pseudo-least recently used (LRU) replacement (the replacement is based on a used bit scheme). The line size is 64 bytes. Unloaded access time is 23 cycles for an L1 data cache miss and 22 cycles for an L1 instruction cache miss.

L2-cache has a 64-byte line size, with 64 bytes interleaved between banks. Pipeline latency in the L2-cache is 8 clocks for a load, 9 clocks for an I-miss, with the critical chunk returned first. 16 outstanding misses per bank are supported for a 64 total misses. Coherence is maintained by shadowing the L1 tags in an L2-cache directory structure (the L2-cache is a point of global visibility). DMA from the I/O is serialized with respect to the traffic from the cores in the L2-cache.

The L2-cache directory shadows the L1 tags. The L1 set index and the L2-cache bank interleaving is such that one forth of the L1 entries come from an L2-cache bank. On an L1 miss, the L1 replacement way and set index identifies the physical location of the tag which will be updated by the miss address. On a store, the directory will be cammed. The directory entries are collated by set, so only 64 entries need to be cammed. This scheme is quite power efficient. Invalidates are a pointer to the physical location in the L1-cache, eliminating the need for a tag lookup in the L1-cache.

Coherency and ordering in the L2-cache are described as:

- Loads update directory and fill the L1-cache on return
- Stores are non-allocating in the L1-cache
- There are two flavors of stores: total store order (TSO) and read memory order (RMO).

Only one outstanding TSO store to the L2-cache per thread is permitted in order to preserve the store ordering. There is no such limitation on RMO stores.

- No tag check is done at a store buffer insert
- Stores check directory and determines an L1-cache hit
- Directory sends store acknowledgements or invalidates to the SPARC core
- Store updates happens to D$ on a store acknowledge
- Crossbar orders the responses across cache banks.

DRAM Controller

The OpenSPARC T1 processor DRAM controller is banked four ways, with each L2 bank interacting with exactly one DRAM controller bank (a two-bank option is available for cost constrained minimal memory configurations). The DRAM controller is interleaved based on physical address bits 7:6, so each DRAM controller bank must have identical dual in-line memory modules (DIMM) installed and enabled.

The OpenSPARC T1 processor uses DDR2 DIMMs and can support one or two ranks of stacked or unstacked DIMMs. Each DRAM bank/port is two-DIMMs wide (128-bit + 16-bit ECC). All installed DIMMs must be identical, and the same number of DIMMs (that is, ranks) must be installed on each DRAM controller port. The DRAM controller frequency is an exact ratio of the core frequency, where the core frequency must be at least three times the DRAM controller frequency. The double data rate (DDR) data buses transfer data at twice the frequency of the DRAM controller frequency.

The OpenSPARC T1 processor can support memory sizes of up to 128 Gbytes with a 25 Gbytes/sec peak bandwidth limit. Memory access is scheduled across 8 reads plus 8 writes, and the processor can be programmed into a two-channel mode for a reduced configuration. Each DRAM channel has 128 bits of data and 16 bytes of ECC interface, with chipkill support, nibble error correction, and byte error detection.

I/O Bridge

The I/O bridge (IOB) performs an address decode on I/O-addressable transactions and directs them to the appropriate internal block or to the appropriate external interface (J-Bus or the

serial system interface). Additionally, the IOB maintains the register status for external interrupts.

J-Bus Interface (JBI)

The J-Bus interface (JBI) is the interconnect between the OpenSPARC T1 processor and the I/O subsystem. The J-Bus is a 200 MHz, 128-bit wide, multiplexed address or data bus, used predominantly for direct memory access (DMA) traffic, plus the programmable input/output (PIO) traffic used to control it.
The J-Bus interface is the functional block that interfaces to the J-Bus, receiving and responding to DMA requests, routing them to the appropriate L2 banks, and also issuing PIO transactions on behalf of the processor threads and forwarding responses back.

Functional Description

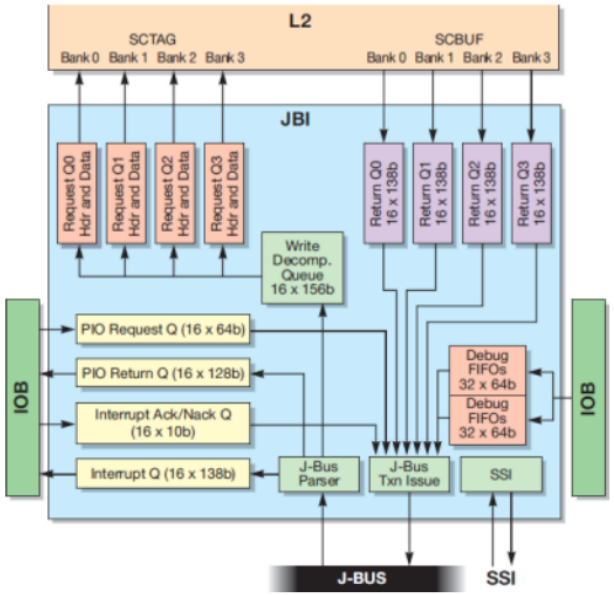
For a detailed description on the external J-Bus interface, refer to *OpenSPARC T1 Processor External Interface Specification*. The OpenSPARC T1 J-Bus interface (JBI) block generates J-Bus transactions and responds to external J-Bus transactions.

The JBI block:

- Interfaces with following blocks in an OpenSPARC T1 processor:
  - L2-cache (scbuf and sctag) to read and write data to L2-cache
  - I/O Bridge (IOB) - for programmed input/output (PIO), interrupts, and debug port • J-Bus I/O pads

- Most of the JBI sub-blocks use the J-Bus clock, and remaining part runs at the CPU core clock or *cmp clk*. The data transfer between the two clock domains is by way of queues within the two clock domains, these are the Request header queues and the Return data queues. The interface to the L2-cache is through the direct memory access (DMA) reads and DMA writes.
- The IOB debug port data is stored in the debug FIFOs and then it is sent out to the external J-Bus.
- IOB PIO requests are stored in the PIO queue and the return data is stored in the PIO return queue. Similarly, there is an interrupt queue and an interrupt ACK/NACK queues in the JBI in order to interface to the IOB.
- There are only two sub-blocks in the JBI (J-Bus parser and J-Bus transaction issue) specific to J-Bus. All of the other blocks are J-Bus independent. J-Bus independent blocks can be used for any other external bus interface implementation.

JBI Functional Block Diagram



I/O Signal list

tion

Signal Name I/O Source/ Destina		Description

cmp_gclk In CTU CMP clock.
cmp_arst_l In CTU CMP clock domain async reset. cmp_grst_l In CTU CMP clock domain reset. jbus_gclk In CTU J-Bus clock.
jbus_arst_l In CTU J-Bus clock domain async reset. jbus_grst_l In CTU J-Bus clock domain reset. ctu_jbi_ssiclk In CTU J-Bus clk divided by 4
ctu_jbi_tx_en In CTU CMP to JBI clock domain crossing synchronization pulse.
ctu_jbi_rx_en In CTU JBI to CMP clock domain crossing synchronization pulse.
ctu_jbi_fst_rst_l In CTU Fast reset for capturing port present bits (J_RST_L +

1).

clk_jbi_jbus_cken In CTU Jbi clock enable.
clk_jbi_cmp_cken In CTU Cmp clock enable.
global_shift_enable In CTU Scan shift enable signal.
ctu_tst_scanmode In CTU Scan mode.
ctu_tst_pre_grst_l In CTU
ctu_tst_scan_disable In CTU
ctu_tst_macrotest In CTU
ctu_tst_short_chain In CTU
ddr3_jbi_scanin18 In DFT
jbusr_jbi_si In DFT

sctag0_jbi_iq_deque_ue 0 (size=4) is being unloaded.
In SCBUF 0
sctag0_jbi_wib_dequ_eue SCTag is unloading a request from its 2 request queue. Return data
scbuf0_jbi_data[31:0]
In SCT AG Write invalidate buffer
0
In SCT AG
scbuf0_jbi_ctag_vld In SCBU F 0 Current data cycle has a uncorrectable
scbuf0_jbi_ue_err In SCBU F 0
Header cycle of a new response packet. error.

sctag0_jbi_por_req_ buf In SCT A G1 request from its 2 request queue.
sctag1_jbi_iq_deque_ue In SCT A G1
sctag1_jbi_wib_deq_ueue In SCBU F 1 Write invalidate buffer (size=4) is being unloaded.
scbuf1_jbi_data[31:0] Request for DOK_FATAL.
In SCT A G0 Return data
SCTag is unloading a
Current data cycle has a uncorrectable
scbuf1_jbi_ctag_vld In SCBUF 1
scbuf1_jbi_ue_err In SCBUF 1 error.
Header cycle of a new response packet.

sctag1_jbi_por_req_ buf In SCTA G2 request from its 2 request queue.
sctag2_jbi_iq_deque_ue In SCTA G2
sctag2_jbi_wib_deq_ueue In SCBUF 2 Write invalidate buffer (size=4) is being unloaded.
scbuf2_jbi_data[31:0] Request for DOK_FATAL.
In SCTA G1 Return data
SCTag is unloading a
Header cycle of a new response packet.
scbuf2_jbi_ctag_vld In SCBUF 2
scbuf2_jbi_ue_err In SCBUF 2 Current data cycle has a uncorrectable

error.

sctag2_jbi_por_req_ buf In SCTA G3 request from its 2 request queue.
sctag3_jbi_iq_deque ue In SCTA G3
sctag3_jbi_wib_deq ueue In SCBUF 3 Write invalidate buffer (size=4) is being unloaded.
scbuf3_jbi_data[31:0] Request for DOK_FATAL. Return data
In SCTA G2
SCTag is unloading a
Current data cycle has a uncorrectable
scbuf3_jbi_ctag_vld In SCBUF 3
scbuf3_jbi_ue_err In SCBUF 3 error.
Header cycle of a new response packet.

sctag3_jbi_por_req_ buf Request for
In SCTA G3 DOK_FATAL.

iob_jbi_pio_stall In IOB PIO stall
iob_jbi_pio_vld In IOB PIO valid
iob_jbi_pio_data[63: In IOB PIO data
0]
iob_jbi_mondo_ack In IOB Mondo acknowledgement
iob_jbi_mondo_nack In IOB Mondo negative acknowledgement
io_jbi_ssi_miso In PADS SSI Master in slave out from pad.
io_jbi_ext_int_l In PADS External interrupt iob_jbi_spi_vld In IOB Valid packet from IOB.
iob_jbi_spi_data[3:0] In IOB Packet data from IOB.

iob_jbi_spi_stall In IOB Flow control to stop data.
io_jbi_j_req4_in_l In PADS J-Bus request. 4 input

io_jbi_j_req5_in_l In PADS J-Bus request. 5 input
io_jbi_j_adtype[7:0] In PADS J-Bus packet type io_jbi_j_ad[127:0] In PADS J-Bus address/data bus io_jbi_j_pack4[2:0] In PADS J-Bus ACK 4 io_jbi_j_pack5[2:0] In PADS J-Bus ACK 5 io_jbi_j_adp[3:0] In PADS J-Bus parity for AD bus io_jbi_j_par In PADS J-Bus parity for request/PACK
iob_jbi_dbg_hi_data In IOB Debug data high [47:0]

iob_jbi_dbg_hi_vld In IOB Debug data high valid
iob_jbi_dbg_lo_data In IOB Debug data low [47:0]

iob_jbi_dbg_lo_vld In IOB Debug data low valid
jbi_ddr3_scanout18 O ut

DFT Scan out

jbi_clk_tr Out CTU Debug_trigger. jbi_jbusr_so Out

DFT Scan out jbi_jbusr_se Out DFT Scan enable

jbi_sctag0_req[31:0] O ut L2-cache request

jbi_scbuf0_ecc[6:0] O ut

jbi_sctag0_req_vld O ut

jbi_sctag1_req[31:0] O ut Next cycle will be header of a new request packet.

jbi_scbuf1_ecc[6:0] O ut

jbi_sctag1_req_vld O ut L2-cache request

jbi_sctag2_req[31:0] O ut

SCT
A G0

SCBUF 0 Next cycle will be header of a new request packet.

SCTA G0

SCTA G1 L2-cache request

SCBUF 1 jbi_scbuf2_ecc[6:0] O ut

SCTA G1 jbi_sctag2_req_vld O ut

SCTA G2 jbi_sctag3_req[31:0] O ut

jbi_iob_pio_vld O ut IOB PIO valid

a[15: 0] IOB PIO data

jbi_iob_pio_dat O ut

MONDO valid

jbi_iob_pio_stall O ut

jbi_iob_mondo_vld O ut

IOB PIO stall IOB

jbi_iob_mondo_data [7:0] O ut IOB MONDO data

pad. PADS Serial clock to pad.

jbi_io_ssi_mosi O ut

jbi_io_ssi_sck O ut IOB Valid packet from UCB.

jbi_iob_spi_vld O ut

PADS Master out slave in to

jbi_iob_spi_data[3:0] O ut IOB Packet data from UCB.

PADS J-Bus request 0

jbi_iob_spi_stall O ut

jbi_io_j_req0_out_l O ut

IOB Flow control to stop data.

jbi_scbuf3_ecc[6:0] O ut

jbi_sctag3_req_vld O ut

SCBUF 2

SCTA G2

SCTA G3

SCBUF 3

SCTA G3

Next cycle will be header of a new request packet.

L2-cache request

Next cycle will be Header of a new request packet.

jbi_io_j_req0_out_e n PADS J-Bus request
O ut 0 enable
J-Bus address/data PADS J-Bus
jbi_io_j_adtype[7:0] O ut
jbi_io_j_adtype_en O ut address/data enable PADS J-Bus
jbi_io_j_ad[127:0] O ut
jbi_io_j_ad_en[3:0] O ut ACK. 0
jbi_io_j_pack0[2:0] O ut
jbi_io_j_pack0_en O ut PADS J-Bus ACK. 0 enable
PADS J-Bus type

PADS J-Bus type enable PADS

jbi_io_j_pack1[2:0] O PADS J-Bus ACK. 1
ut
jbi_io_j_pack1_en O ut
jbi_io_j_adp[3:0] O ut
jbi_io_j_adp_en O ut
PADS J-Bus ACK. 1 enable

PADS J-Bus address/data Parity

PADS J-Bus address/data parity

enable

jbi_io_config_dtl[1: 0]
PADS J-Bus I/O DTL
configuration
O
ut

jbi_io_j_req0_out_e n PADS J-Bus request
O ut 0 enable
J-Bus address/data PADS J-Bus
jbi_io_j_adtype[7:0] O ut
jbi_io_j_adtype_en O ut address/data enable PADS J-Bus
jbi_io_j_ad[127:0] O ut
jbi_io_j_ad_en[3:0] O ut ACK. 0
jbi_io_j_pack0[2:0] O ut
jbi_io_j_pack0_en O ut PADS J-Bus ACK. 0 enable
PADS J-Bus type

PADS J-Bus type enable PADS

jbi_io_j_pack1[2:0] O PADS J-Bus ACK. 1
ut
jbi_io_j_pack1_en O ut
jbi_io_j_adp[3:0] O ut
jbi_io_j_adp_en O ut
PADS J-Bus ACK. 1 enable

PADS J-Bus address/data Parity

PADS J-Bus address/data parity

enable

jbi_io_config_dtl[1: 0]
PADS J-Bus I/O DTL
configuration
O
ut