

Coherency Hub Design for Multi-Node Victoria Falls Server Systems

John Feehrer, Paul Rotker, Milton Shih, John Heath Sebastian Turullols
 Paul Gingras, Peter Yakutis, Stephen Phillips *Univ. of Maine Pano Logic Inc.*
Sun Microsystems Inc.
firstname.lastname@sun.com heath@usm.maine.edu sturullols@yahoo.com

Abstract

This paper describes the design of a coherency hub ASIC for a 4-socket highly-threaded multiprocessor using Sun's Victoria Falls processor. Victoria Falls is an 8-core CMT processor in the Niagara family, with 8 threads per core and a shared L2 cache. The coherency hub, named Zambezi, enables cost-effective scaling to 4 sockets with a total thread count of 256 and near-linear performance scaling on transaction processing workloads. Extending a 2-socket "glueless" system to a 4-socket system with no change to the processor was a key requirement. Zambezi broadcasts snoop requests to all nodes (i.e. sockets), serializes requests to the same address, and consolidates snoop responses. The hub communicates with each node via point-to-point serial links, using a proprietary data link layer implemented over an FBDIMM PHY. In this paper, we summarize the ASIC micro-architecture and coherency scheme, highlight how we addressed the engineering challenges we faced, and report performance scalability results we achieved on key commercial server benchmarks. Conflicting constraints (800 MHz operation and a 6-stage pipeline budget) presented the primary challenge to architecture, design and layout.

1. Introduction

This paper describes the design and implementation of a coherency hub ASIC used in Sun's latest highly-threaded servers based on the Victoria Falls (VF, known officially as UltraSPARC T2+) processor [1], a derivative of the Niagara 2 (UltraSPARC T2) processor [2]. Initial Niagara chip multi-threading (CMT) systems [3] were single-socket and could only scale to the extent possible in a single silicon die. In order to bring the benefits of CMT to larger workloads, there was a need to scale beyond a single socket. Since CMT requires massive memory bandwidth to achieve its throughput performance, the challenge was to develop a coherency link and fabric to enable near-linear scaling of performance with thread count in a multi-node (i.e.

multi-socket¹) system. This challenge required some creative engineering solutions which we highlight in this paper.

The Victoria Falls processor is an 8-core / 64-thread UltraSPARC processor with integrated FBDIMM memory controllers, an FBDIMM-based coherency link and a x8 PCI-Express port. These integrated functions provide balanced memory and I/O performance for the on-chip threads and allow for scaling memory and I/O bandwidth in multi-socket configurations. VF server platforms are focused on throughput computing applications commonly employed in large datacenters, including database management, web-base services, transaction processing, application serving, ERP, and CRM. A 2-node system can be built by connecting two VF's directly together in a "glueless" configuration; the T5140/T5240 servers based on this concept were launched in April 2008 [4]. With the coherency hub (Zambezi, known officially as UltraSPARC T2 Plus XBR) systems with up to 4 nodes can be built. The primary design goal for Zambezi was to extend the existing "glueless" 2-node configuration to 4-node to enable near linear scaling of performance with thread count – without requiring a re-spin of the processor (VF). While a 4-node "glueless" configuration would minimize inter-node latency, time-to-market and development cost considerations dictated an ASIC hub solution. Furthermore, performance modeling confirmed that near linear scaling could still be achieved with this approach. While directory coherence (e.g. [5]) is a more scalable solution, its additional overhead and design/verification complexity was not deemed necessary for the 4-node space for which this product was focused.

The following sections describe respectively the system architecture, the Zambezi micro-architecture which supports the scalability goal, the design and layout rules followed by designers in order to meet the 800 MHz clock rate and low-latency requirement, the chip and system verification strategy employed, and the performance and scalability results achieved on the 4-node systems.

¹In this paper the terms "socket" and "node" both refer to a single VF.

2. Multinode CMT systems

The CMT style of computer architecture is becoming more and more prevalent as semiconductor feature size continues to shrink. CMT places a symmetric multiprocessor architecture on a single processor die, where each processor core can execute multiple software threads simultaneously. Sun's Niagara processors are unique in the industry in the total number of threads they can run. Essentially CMT trades off thread latency for aggregate thread throughput, recognizing that memory latency is not improving at the rate of CPU clock speed, and capitalizing on the high degree of thread parallelism inherent in commercial server applications. The CMT philosophy allows for a relatively simple core design, in contrast to more power-hungry cores that employ speculation and out-of-order execution to improve the average clocks-per-instruction.

The Victoria Falls processor architecture was described at the Hot Chips conference in 2007 [1]. VF is a SPARC v9 compliant architecture that supports 64 threads (8 cores x 8 threads/core). In a 4-node VF server, the total number of threads is 256. VF and its platforms are able to scale to large data sets and workloads by way of the memory hierarchy.

Victoria Falls platforms comply with the SPARC memory model [6], [7]. In particular, Total Store Order is maintained by the core combined with the load/store unit and L2\$ crossbar. The crossbar connecting cores to the L2\$ is nonblocking; it establishes memory order between transactions from the same and different L2\$ banks, and guarantees delivery of transactions to L2\$ banks in the same order. A high degree of concurrency is provided by an 8-deep store buffer per thread. A processor can have up to 384 transactions in flight.

The 4MB L2\$ is shared by all cores. It is subdivided into 8 banks with independent L2\$ pipelines. It uses a writeback protocol and a modified-LRU replacement algorithm. To minimize L2\$ conflicts, the design features 16-way associativity and set index hashing. The "micro-parallelization" challenge for software is dividing serial code segments among threads and coming up with lock-free synchronization primitives. Synchronization overhead is minimized due to core-to-core communication optimization through the L2\$. This prevent threads from stalling during synchronization events.

VF contains a pair of on-chip memory controllers, each providing 2 channels of FBDIMM. Using DDR2-667 DRAM, this structure yields a theoretical peak of 21 GB/sec read bandwidth and 10.5 GB/s write bandwidth per VF. Up to 128 GB capacity is supported. Each VF also contains a Root Complex providing a x8

PCIe 1.1 root port with 2 GB/s of dual simplex peak theoretical bandwidth.

A 4-node SMP system is constructed by tying together four VF processors through a snoopy coherency interconnect with Zambezi serving as the central hub. See Illustration 1. There are four independent "coherence planes." The plane is identified by bits 13:12 of the physical address. Each plane touches two L2 banks on each VF. Each coherence plane in turn has a corresponding Zambezi chip to handle its addresses. Each Zambezi has a connection to each of the four VF nodes. Because planes are independent, there are no connections between Zambezi chips. The connection from Zambezi to VF is a pair of unidirectional serials links; the pair is referred to as a "coherency link."² Physical address conflicts (i.e. multiple requests issued by different nodes to the same physical address) are serialized by Zambezi: one of the conflicting requests is broadcast for snoop and the others are stored in a pending queue.

Cache coherency is maintained through a MOESI protocol; cache-to-cache transfer occurs if a request hits a line held in the M/O/E/S state by another node. There are 3 virtual channels carrying request, reply, and data packets, respectively. The data channel distinguishes the critical (first 32B) and non-critical (second 32B) halves of a cacheline, so that critical-word-first transfer optimizations can be done. For example, the data channel arbiter in each Zambezi output port gives priority to critical data over non-critical data when examining multiple cachelines of data ready to send from each input port. Critical-word first transfer is especially useful for code fetches. The protocol supports out-of-order snoop responses. The interconnect is also used for inter-node I/O traffic (both PIO and DMA) and interrupts (both I/O interrupts and cross-calls between nodes).

3. Zambezi micro-architecture

Illustration 2 shows a top level block diagram of the Zambezi ASIC. There are four instances of the Link Port Unit (LPU), which includes the FBDIMM SerDes, Link Framing Unit (LFU), an input block, output block, and transaction scoreboard (TSB). The block at the top is the Address Serialization Unit (ASU) which is responsible for resolving address conflicts among coherent request from the four ports and creating a serial order for their broadcast to each port. The block at the bottom is the General Purpose Design (GPD) module, which handles all programmed I/O requests to internal configuration/status registers (CSRs),

²Coherency links carry all inter-node traffic, including programmed I/O reads and writes to noncacheable addresses mapped to locations on nodes other than the requesting node.

implements a Low-Pin Count (LPC) [8] slave interface for communication with a Service Processor, manages reset sequencing, PLL configuration, and clock distribution, and does other general housekeeping functions.

The coherency link uses a SerDes and PHY layer that is based on FBDIMM [9]. FBDIMM was chosen because it is an industry standard and provides the high bandwidth needed to support the heavy coherency and I/O traffic load of a 4-node system. Since it is a standard, the SerDes/PHY is readily available in most ASIC vendors' IP libraries. There are 14 transmit and 14 receive lanes per channel. The link supports 4.8 GT/s (8.4 GB/s) raw bandwidth per direction. The data link layer is proprietary, based on a 144-bit frame and 24-bit CRC. This yields 6.4 GB/s peak data bandwidth per link. Packet transmission is reliable; a frame is replayed if there is a CRC error. When an excessive number of CRC errors is detected within a short period of time, the link is retrained. An additional RAS (Reliability/Accessibility/Serviceability) feature is the ability to retrain the link with a faulty lane removed. The transaction layer defines 3 sizes of packets: 3B, 7B, and 18B packets. Packets may cross frame boundaries. Each output port of Zambezi implements a weighted round robin arbiter to select a packet from among the egress virtual channel queues. A 64B cacheline data transfer is done in four frames (each carrying a 16B "chunk") which can be time-interleaved with frames carrying other request, response, or data packets.

The main coherency functions performed by Zambezi are snoop broadcast and snoop response consolidation. A cacheable memory request arriving on one port must be broadcast to each of the other ports; in addition a "forwarded request ack" must also be sent back to the requester in conjunction with broadcasting the request to other ports. Each port returns a snoop response packet (indicating miss, hit in shared state, hit in owned state, or hit in modified state). Zambezi returns the consolidated snoop response to the requesting node.

The ASU (Address Serialization Unit) is responsible for serializing coherent requests to the same memory address and for routing both coherent and non-coherent transactions to their proper destination port(s). Illustration 3 shows the internal structures for the ASU. The ASU has four FIFOs to buffer incoming coherent requests from all four ports, and four more FIFOs to buffer incoming non-coherent requests from all four ports. The ASU insures that for any given address in the system, only one coherent transaction is outstanding (i.e. is actively being processed) on that address. All other transactions to that address are pended in the ASU. The ASU will wake up and reissue pended transactions as earlier transactions to the same address are completed. Non-coherent transactions do not

require address serialization; a non-coherent request is simply routed to its destination port and enqueued there with other (coherent and non-coherent) requests.

When the ASU receives a coherent read or write transaction from a port, the ASU looks up the address of the transaction in a 96-entry CAM containing addresses of outstanding transactions from all four ports. If the address of the transaction being looked up misses in the CAM, then the transaction is forwarded to the appropriate port(s). If the address hits in the CAM, then the ASU will not forward the transaction to the other ports but instead, will insert the transaction at the end of a linked list containing all other pended transactions to the same address in the order they were originally received by the ASU. There are 96 linked lists, one corresponding to each CAM entry. As transactions are completed, the ASU checks to see if there are any other transactions to the same address which are waiting to be reissued. If so, then the transaction at the head of the linked list corresponding to the address of the just-completed transaction is woken up and forwarded to the appropriate port(s). Patents covering these ASU concepts have been filed.

There is no need for Zambezi to maintain a global order for requests, but it does need to follow a simple ordering rule regarding forwarded acks. A forwarded ack associated with a request R_a for a given address X must be sent back to the issuing node N before a forwarded request R_b to address X that entered the ASU serialization stream after R_a entered it is sent to N . This requirement is needed to satisfy TSO. VF systems maintain PCI-Express ordering rules as well, but Zambezi itself plays no role in enforcing these rules; they are enforced by the Root Complex and modules handling inter-node communication in VF.

Zambezi contains enough internal storage to buffer all pending request, response, and data packets (cacheable and non-cacheable). However, VF input buffers are not maximally sized to the total number of packets that can be in flight to the given VF. For this reason, flow control credits are maintained by each port to prevent overflow of the buffers in the VF which that port services. Flow control credits are tracked for each virtual channel independently. Flow control is needed for PIO write data and coherent writeback data (and is linked to the corresponding requests), but not for read return data, as the VF requester guarantees space for all data associated with its pending read requests.

Error handling is straightforward and architected in a way that minimizes timing impact. All RAM structures beyond a certain threshold size are protected with parity. The threshold was determined from the overall system-level FIT rate budgets for the various error types (silent uncorrectable, reported uncorrectable, reported correctable). Because Zambezi does not

contain any large RAMs and each RAM occupancy time is relatively short, there was no need to add ECC protection on any of its RAMs to meet system-level FIT requirements. A parity error on any internal control structure or packet except data (i.e. packet payload) is treated as fatal to the system and is reported to the Service Processor through the LPC port. The Service Processor responds to a fatal error by asserting a system-wide reset which clears all state but preserves error logs. Data parity errors are non-fatal and are signaled using a bit of the data packet. Fault management software is informed of these errors and can take corrective action. Certain protocol errors, including illegal snoop responses (e.g. Hit-modified from more than one VF) are also detected and reported as fatal errors. The Service Processor has access to all CSRs including error logs. Fault management software can perform diagnosis and fault isolation with the goal of isolating the fault to a specific field-replaceable unit (FRU). VF nodes can also access CSRs, but the “backdoor” path via the Service Processor is required for fault diagnosis when the fault prevents access through the coherency links.

4. Design and layout strategy and rules

The 800 MHz clock rate was an aggressive target for a 65 nm ASIC process. The ASIC was fabricated by Texas Instruments using a 7-layer metal stack. This section describes the design, integration, and layout rules that were followed to ensure successful timing closure at this frequency.

The most important rules put in place for designers at the beginning of the project were those governing fanout. Minimal fanout was critical for making timing, because reducing fanout lowers congestion and logic levels. The team relied on a set of tools including Spyglass [10], and Magma BlastLogic [11] to help identify high fanout areas and to guide restructuring of the RTL to reduce the number of endpoints on nets. These tools were actually run by a separate synthesis team, which provided restructuring recommendations to the RTL designers. Furthermore, a “fanout-of-1” discipline was maintained for all nets going between floorplanned objects (referred to as “clusters”): each such net could have only a single destination.

Illustration 4 shows the floorplan. Note the “AGS” refers to the floorplanned unit consisting of the ASU and GPD. Those portions of the design that were not timing-critical, such as CSR access paths and error signaling, were pipelined liberally to prevent synthesis tools from working too hard on them at the expense of the critical paths passing packets across the chip. As logic was restructured/replicated to reduce fanout, gate counts did not grow since the synthesis tools could focus

on proper synthesis that would result in more efficient gate selection.

As a result of following these rules, timing was closed in less than 3 months from the RTL completion date. Synthesis was done across all corners at 800 MHz. The Primitime Signal Integrity tool [12] with Composite Current Source was used to accurately model crosstalk and parasitic effects that can be significant in 65 nm designs. The target clock rate for synthesis was actually 1 GHz, which derates to 800 MHz with 20% margin for clock skew and process/voltage/temperature variation in DC synthesis. The graphs in Illustration 5 shows the Magma FixCell timing and endpoint fanout reduction as a function of each successive integration. Specific details of the ASIC are given in the table below.

| | |
|------------------------|---|
| Gate count | 3.6M |
| Number of storage bits | 300K bits (impl as flip-flops) |
| Process | 65 nm, TI C021.A |
| Die size | 8.4 mm x 9.4 mm |
| Signal I/Os | 341 |
| Package | Organic, flip-chip BGA, 900 balls, 1mm ball pitch |
| Power dissipation | 25 W |

Table 1: Zambezi physical characteristics

Because the chip was pad-limited, it was possible to implement all memory structures using flip-flops rather than SRAM cells or latch arrays. In addition to eliminating the front-end design effort to produce SRAM macros and improving FIT rates, using flip-flops rather than memory cells also simplified DFT, since MBIST was not needed.

The “fanout of 1” rule at the top level was a key strategy to allow predictable top level timing closure. This in effect made the design at the top level correct by construction. At 65 nm, there is potentially a long cycle time of analysis at the top level accounting for on-chip variation (OCV) effects. The amount of hand tuning that would be required to tune top level routes and buffering based on this analysis could be prohibitive. The top level fanout constraint effectively eliminated the need for any such tuning to fix inter-cluster routes, and allowed final timing closure less than 3 months from RTL freeze.

5. Verification strategy

Design and verification teams worked closely together to ensure a high-quality design. As a result, only a single functional bug, in non-critical error

handling functionality, was discovered during lab testing. The Zambezi verification strategy was based on a common verification methodology called CVF (common verification framework), which allowed transactors, checkers and coverage objects to be written and reused across different testbench environments (from module to ASIC to System verification). The re-usability of these components allowed the team to develop high level testbenches (ASIC/System) significantly faster than on past projects, given the components being used were already written/debugged in stand alone testbench environments. The Zambezi verification team developed a total of over 30 common transactors, checkers and coverage components, which were used in 7 different testbench environments (4 module stand-alone environments, 1 interoperability environment, an ASIC testbench, and a 4-way System verification environment).

A key component of the verification strategy was a phased design release. That is, the design and verification teams developed a detailed list of design features to be delivered into verification based on the verification requirements (for example, for verification, it was beneficial to get egress functionality first followed by ingress functionality). Each of the 4 major design modules had specific features outlined along with a scheduled design release date. An important aspect of our testing philosophy was that RTL for a new design release was verified in the stand-alone testbench environment first; only when the functionality was deemed to be working was the design released to ASIC verification. A similar process was followed from ASIC verification to System. The objective was to find bugs in the most efficient environment. Illustration 6 shows the per-block functional coverage over time of the project.

An important part of the verification strategy was in verifying all pieces of the design (VF and Zambezi) together in a single model - this testing was done in system verification and was composed of two separate environments: RTL simulation using VCS, and hardware emulation using Palladium [13]. The RTL simulation had a variety of different model configurations including 2, 3 and 4-node models with varying number of CPU cores. The RTL simulation models were effective in that they allowed the team to flush out issues between VF and Zambezi early in the process. The downside was that the simulation performance was very slow (1 cycle/sec), so did not lend itself well to “bug hunting” tasks. For these class of problems, the team relied on Palladium, an FPGA-based hardware acceleration and hardware/software co-verification platform which improved simulation speed significantly. As in the VCS environment, the team built a variety of different model configurations (2, 3 and 4 node) along with a varying number of threads (the

largest being 192 threads). Perhaps the biggest value/benefit of the Palladium work was that the team was able to run the entire firmware initialization sequence (including the Hypervisor, Service Processor code, and boot code) and was able to boot Solaris prior to tape-out which greatly increased the confidence in the system. In the process of doing this, the team found/resolved several software issues (in addition to finding one hardware problem).

The verification team used industry-standard tools including VCS [14], Vera [15], and Oin [16] (the latter for assertions, coverage, and clock domain crossings). Verification progress was measured using code and protocol coverage metrics which were defined from the Zambezi macro/micro-architectural specifications and from specifications detailing the coherency link protocol. Protocol coverage (which included over 1600 coverage points representing all parts of the coherency protocol) was in excess of 99% covered, prior to tape-out.

6. Performance and scalability

As emphasized earlier, the primary goal for the project was to minimize latency through Zambezi. This led to a latency budget for the core pipeline (excluding the LFU) of only 6 cycles. This budget had to be met while meeting the design rules cited above. Latency is critical because it directly impacts performance of those portions of an application that are single-threaded; for example log manager functions in database applications sometimes are inherently single-threaded. The latency through the SerDes was fixed by the vendor's design. Latency through the LFU was largely fixed and determined by link layer requirements, though in some places a bypass FIFO design was used to optimize LFU latency. Similar bypass optimizations were made in the Zambezi core pipeline encompassing the input port, ASU, and output port. Because most errors are treated as fatal to the system, much of the error checking is done on the side, so as not to interfere with timing on the critical path. The pin-to-pin latency for a request packet (entering one port and forwarded to other ports for snoop), as shown in Illustration 7, was 33.1 nsec, broken down as follows:

- SerDes in: 5.3 nsec
- LFU in: 10.0 nsec
- Input port ->ASU -> output port: 6 x 800 MHz cycles = 7.5 nsec
- LFU out: 5.0 nsec
- SerDes out: 5.3 nsec

For response and data packets flowing through the chip, the latency is nearly the same as shown above. Note that 30% of the pin-to-pin delay is taken up by serialization and deserialization. This delay could be

reduced by using parallel links, but at the expense of extra pins (increasing the package size and thus cost), additional board traces and more difficult clock distribution. The benefits of serial communication outweighed these disadvantages.

Actual benchmark results cannot be shared at this time because the product has not been publicly announced yet, but early testing shows applications that capitalize on many threads scale well. A standard Java application benchmark representing order processing for a wholesale supplier shows good scaling of 1/1.95/3.37 in going from the 1-node to the 2-node to the 4-node system. The scaling for a standard floating-point component exhibits scaling of 1/1.97/3.51. Detailed cycle-accurate simulations of representative workloads yield the results shown in the tables below, normalized to a single-node VF system.

| Metric | 1-way VF | 2-way VF | 4-way VF |
|---------------------------|----------|----------|----------|
| instrs/cycle | 1 | 0.951 | 0.823 |
| avg L2 rd miss latency | 1 | 1.177 | 1.632 |
| avg L2 write miss latency | 1 | 1.208 | 1.663 |

Table 2: Simulation results for 1st workload

| Metric | 1-way VF | 2-way VF | 4-way VF |
|---------------------------|----------|----------|----------|
| instrs/cycle | 1 | 0.932 | 0.791 |
| avg L2 rd miss latency | 1 | 1.196 | 1.668 |
| avg L2 write miss latency | 1 | 1.226 | 1.715 |

Table 3: Simulation results for 2nd workload

Note that average latencies shown here cover three cases: local DRAM memory, remote DRAM memory (through Zambezi), and cache-to-cache transfers (through Zambezi). Effort is underway to parallelize code sections that are currently serial to improve execution time on strategic applications.

7. Conclusion and acknowledgements

Many commercial server applications can benefit from highly-threaded multi-core processors such as Sun's Victoria Falls. Due to high levels of inherent parallelism these applications can capitalize on a higher thread count than a single processor node can provide. The Zambezi ASIC described in this paper enables a 4-node 256-thread VF-based architecture demonstrating near-linear scaling of performance with thread count. Zambezi functions as a coherency bridge to broadcast

snoop requests, resolve address conflicts, and consolidate snoop response. High bandwidth and low latency are both important for this coherent interconnect. FBDIMM technology is used for the coherency links in order to achieve the high bandwidth needed for scaling. Following a well-defined set of design and signal fanout rules critical for quick timing closure in the 65 nm process, the design team was able to meet aggressive latency requirements, as well, at an 800 MHz clock rate. A common verification framework allowing re-use and leverage of key components ensured an extremely high RTL quality level. A combination of RTL simulation and hardware accelerator for system verification enabled a smooth hardware/software validation and bringup process in the lab.

The authors acknowledge contributions of the members of the Zambezi architecture, design, verification, and integration teams as well as engineers and managers at Texas Instruments who worked on the physical realization of this ASIC.

8. References

- [1] Stephen Phillips, "Victoria Falls: Scaling Highly-Threaded Processor Cores", Hot Chips, 19th Annual IEEE Symposium on High Performance Chips, 8/26/07.
- [2] Sun Microsystems Inc., "UltraSPARC T2 Processor". <http://www.sun.com/processors/UltraSPARC-T2/>
- [3] Sun Microsystems Inc., "New Sun Servers Capitalize on Design Innovation, Pack the Power of 64 Servers in a Single System", October 9th 2007. www.sun.com/aboutsun/pr/2007-10/sunflash.20071009.1.xml.
- [4] [Sun SPARC Enterprise T5240 Server: www.sun.com/servers/coolthreads/t5240.](http://www.sun.com/servers/coolthreads/t5240) (also [coolthreads/t5140](http://www.sun.com/servers/coolthreads/t5140))
- [5] [John L. Hennessy and David A. Patterson, Computer Architecture: A Quantitative Approach, 4th Edition. Morgan Kaufmann, 2007.](http://www.morgankaufmann.com)
- [6] Sun Microsystems, "UltraSPARC IV Processor Architecture Overview", February 2004. <http://www.sun.com/processors/whitepapers/>
- [7] D. Weaver and T. Germond (Editors), *The SPARC Architecture Manual Version 9*. Prentice-Hall, Englewood Cliffs, NJ. 1994-2000. <http://developers.sun.com/solaris/articles/sparcv9.pdf>.
- [8] "Low Pin Count Interface Specification," Intel. (www.intel.com/design/chipsets/industry/lpc.htm).
- [9] "FB-DIMM Draft Specification: Architecture and Protocol" (JEDEC Standard Proposal), March 2005, JEDEC Solid State Technology Association.
- [10] Atrenta Inc. (www.atrenta.com).
- [11] www.magma-da.com
- [12] www.synopsys.com/products/analysis/primetime_ds.htm
- [13] www.cadence.com/products/functional_ver/accel_emul/
- [14] www.synopsys.com/products/simulation/
- [15] www.synopsys.com/products/vera/vera.html
- [16] www.mentor.com/products/fv/

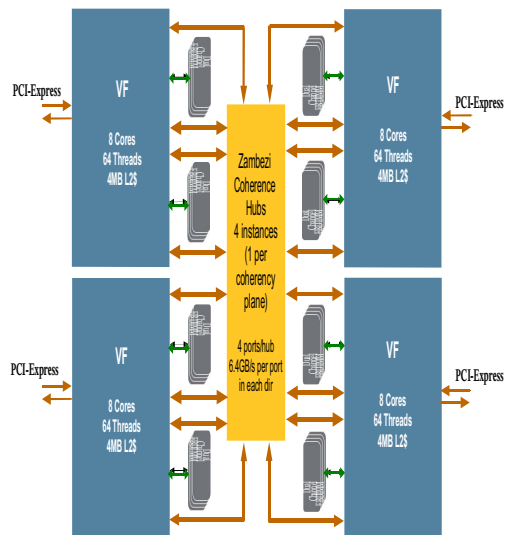


Illustration 1: Block diagram of 4-way VF system. Has 4 Zambezi chips (1 per plane).

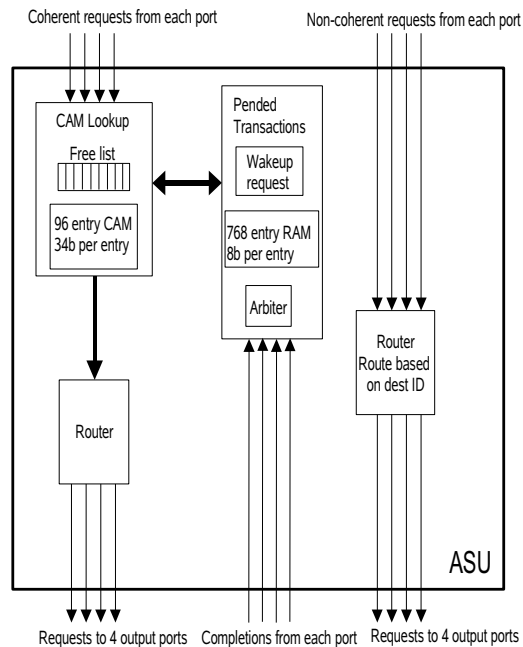


Illustration 3: Block diagram of Address Serialization Unit.

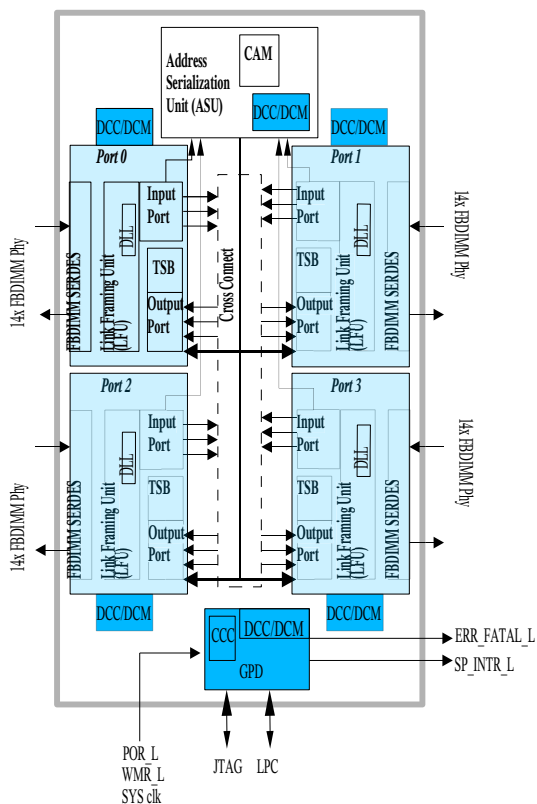


Illustration 2: Block diagram of Zambezi.

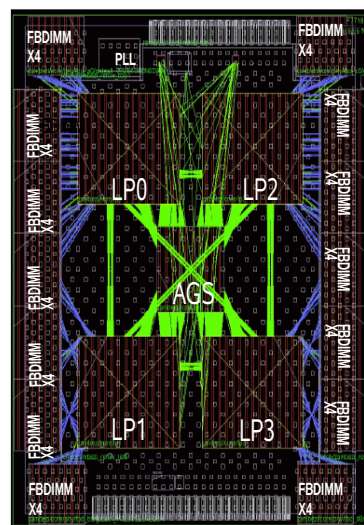


Illustration 4: Zambezi floorplan. "AGS" consists of ASU and GPD.

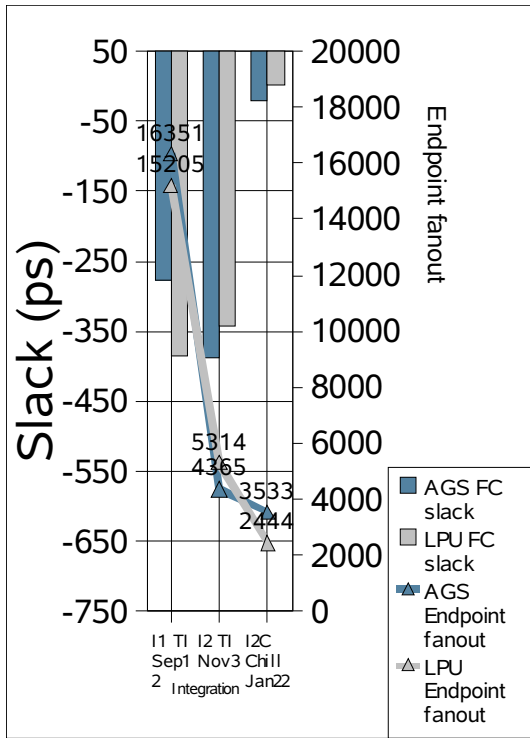


Illustration 5: Per-unit timing slack and endpoint fanout for each integration.

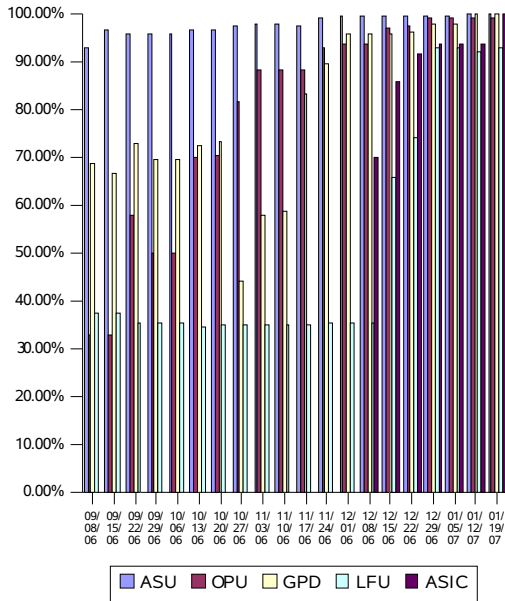


Illustration 6: Functional coverage over course of project.

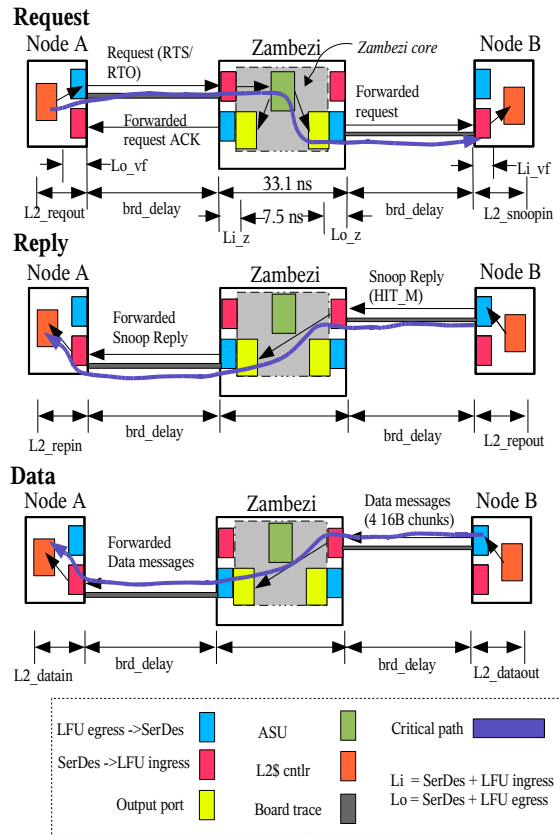


Illustration 7: Latency components in VF-to-VF paths through Zambezi.