UltraSPARC™-I Emulation

James Gateley, Miriam Blatt, Dennis Chen, Scott Cooke, Piyush Desai, Manjunath Doreswamy, Mark Elgood, Gary Feierbach, Tim Goldsbury, Dale Greenley, Raju Joshi, Mike Khosraviani, Robert Kwong, Manish Motwani, Chitresh Narasimhaiah, Sam J. Nicolino Jr., Tooru Ozeki, Gary Peterson, Chris Salzmann, Nasser Shayesteh, Jeffrey Whitman and Pak Wong

SPARC Technology, Sun Microsystems, Inc.
2550 Garcia Avenue, Mountain View, CA 94043

Abstract - The next generation UltraSPARC-I CPU represents a significant step forward in processor performance at the cost of increased design complexity. Added complexity increases the risks in achieving functionally correct first silicon. Existing design verification techniques were supplemented by applying emulation to obtain an early look at functionality. Discussed are the goals, methods and results of the UltraSPARC-I emulation.

I. Introduction

Emulation is a method of functionally implementing a logic design with performance several orders of magnitude faster than simulation for the purpose of design verification. Emulation requires the use of dynamically configured hardware in which to implement the design. Emulation differs from traditional simulation in that the emulation hardware implements the functionality of the design while simulation is a generalized program that models the functionality while executing on a general purpose host computer. Where simulation processes events sequentially, emulation operates in parallel, as will the final silicon, resulting in fast emulation speeds of approximately 500 KHz.

The UltraSPARC-I CPU from SPARC Technology, a division of Sun Microsystems, Inc., is a high-performance 64-bit V9 SPARC implementation 100% compatible with existing binaries [1][2][3][4][5][6]. It is a 4 way superscalar design using 9 functional units. Support for either tightly-coupled or loosely-coupled multiprocessing is provided. New VISual instructions provide outstanding performance for multi-media applications[7][8].

II. Emulation Goals

On-time delivery of new processor designs to market is critical for SPARC Technology to maintain its advantage. Emulating a processor design provides a competitive advantage over those that do not emulate designs. When a product arrives within its market window, it has greater potential for revenue generation.

32nd ACM/IEEE Design Automation Conference ®

The primary goal for processor emulation is to shorten the time-to-market for the design [9][10]. Emulation provides verification capabilities not available through any other technology. Emulation provides five advantages that affect time-to-market:

- Improves confidence in the design prior to tapeout.
- May reduce the number of silicon iterations to arrive at the final design.
- Shortens the time to bring-up the silicon testbed by having software debugged in emulation.
- Provides advanced debug facilities to rapidly isolate any functional problems in the silicon.
- Provides design re-verification prior to additional silicon iterations.

Tapeout is the event in time when the design is released for silicon fabrication. Emulation improves pre-tapeout confidence in the functionality of the design by providing an early look at tests too complex to simulate, including:

- Power-On-Self-Test (POST) and other diagnostics
- Single-user and Multi-user Solaris operating system boot
- OpenWindows and applications
- Benchmarks, stress tests and random exerciser tools

When emulation finds functional bugs in the processor design before tapeout, the quality of the resulting silicon is improved as the bug is corrected before silicon commitment. This reduces the number of silicon iterations to arrive at the final design. Emulation finds bugs that slipped through simulation testing that would otherwise be found during silicon debug.

In a typical CPU project without emulation, system software testing is scheduled to complete by first silicon return. Though completely simulated, subtle problems may escape detection until silicon bring-up. With emulation, the system software development is accelerated to meet an earlier schedule for emulation bring-up. Software bugs found and fixed during emulation bring-up will not cost time during the silicon bring-up, thus, emulation creates a pretested environment for silicon.

The emulation system provides the capability to probe any net within the design and obtain deep history traces of those

nets. As many as 1000 nets per emulation system can be probed at one time. These advanced debugging facilities aid in the rapid isolation of any functional problem discovered in the silicon.

Design changes can be easily incorporated into the emulation database. If additional silicon iterations to include functionality changes are required, emulation provides significant design re-verification prior to tapeout.

III. Emulation Methodology

The Emulation Methodology consists of four major phases: Pre-configuration, Configuration, Testbed, and In-Circuit Emulation (ICE). Fig. 1 shows the relationship between phases.

Pre-Configuration Preparation must be completed prior to an attempt at an emulation database configuration. This phase brings together all of the necessary database control files, netlists, libraries, and other inputs.

The Full Chip Configuration step uses the vendor supplied configuration software to create the downloadable emulation database. This step takes the results of the Pre-Configuration Preparation phase and generates the bitstream necessary to program the emulation hardware.

Testbed Preparation includes the design, planning and implementation of the physical environment that contains the emulation. This testbed environment must provide all necessary real world hardware interfaces and resources.

The In-Circuit Emulation Phase integrates the testbed and the design emulation to deliver a system in which verification and debug can occur.

A. Pre-Configuration Preparation

During Pre-Configuration Preparation, all inputs needed to successfully complete the full design configuration are assembled and prepared for inclusion in the configuration step. Some key components and procedures are discussed in the following subsections.

Leaf-Cell Library Translation. The UltraSPARC-I project made liberal use of custom leaf-cell libraries to represent design elements. To increase the flexibility of the libraries for the design team, several individual cell libraries were created. The custom library cells represent elements in terms of the CPU design primitive elements. For emulation, each element must be expressed in terms of primitives defined by the emulation vendor. Fortunately, a rather large set of emulation primitive elements was available; well in excess of 2000 elements. Using an automated flow to perform this mapping greatly simplified the task and facilitated rapid updating of the emulation library to reflect the latest design library changes.

The emulation libraries must be verified to ensure that each element fully and accurately reflects the original functional intent of the library designers. This verification is accomplished by comparing the results of a simulation of the original libraries to the emulation of the new libraries under the same stimuli. A second verification step for the emulation library elements is implicitly performed during the block level testing described below.

Depending on design style, there can be problem areas for emulation within a leaf-cell library. These must be identified and modified. Possible problems include “Wire” logic (Wired-or, Wired-and, etc.) that must be replaced by appropriate combinational gates, transistor specified elements that must be redesigned into gates, element simplification, and multiple identical elements with different power levels that can be reduced to a single element (power levels are a circuit requirement not an emulation requirement).

Although UltraSPARC-I exclusively used custom leaf-cell libraries, other projects may benefit from emulation vendor supplied pretested libraries for many standard ASIC technologies.

The result of this translation and verification is a set of libraries in terms of emulation primitives available to support netlist configurations.

Design Netlists. Full Chip Configuration requires all necessary gate level netlists to be collected and presented to the configuration software. This can be easy on a small project, but a design such as UltraSPARC-I involves a large number of individual netlists that can be in a variety of standard netlist formats. The netlists originate from different team members (standard cell designers, datapath designers, megacell designers, etc.) and often use specialized libraries. All netlists must be in a format acceptable to the emulation vendor’s configuration software. Translations could be required and tools were developed as needed by the emulation team.

Once netlists are gathered and translated into an appropriate standard syntax, the semantics of the netlists can be reviewed.

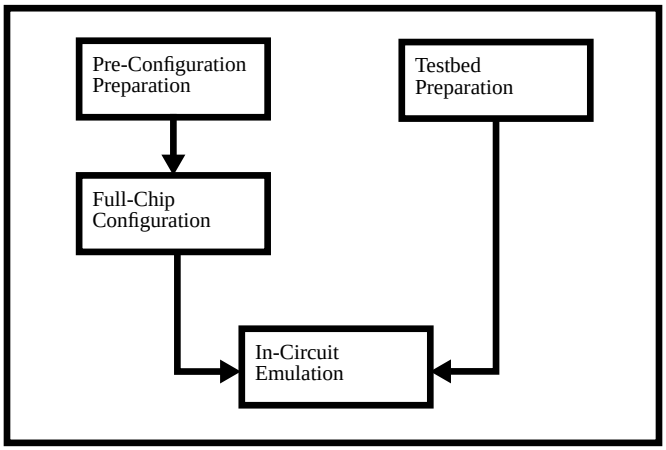

Fig. 1. The Emulation Methodology consists of four major phases. The configuration tasks usually overlap in time with the Testbed Preparation. To successfully go In-Circuit, all tasks must be completed.

There are several possible problem areas where the original design may not be optimally compatible with emulation technology. These incompatibilities are resolved by redesign or emulation-local modifications to the netlists. Areas of concern include gated clocks, feed-thrus, and pre-charge logic.

The emulation hardware implements a clock distribution scheme where highly tuned "low-skew nets" are designed into the emulation system. These low-skew nets ensure that all physical areas of the emulation hardware see the clocks with minimal skew with respect to all other locations within the system. When a design clock is gated it can no longer be routed over these special low-skew nets but must be routed through the system as any other data portion of the design. This has two very negative effects: potential for Hold Time Violations in the design, and costs gates plus pins to route the clock. Gated clocks are redesigned to either remove the gating, or pull the gating to the root of the clock tree to allow use of low-skew nets "down-stream" from the point of gating. The vendor supplied clocktree analysis procedure greatly aids in identification of this situation. Changes to the clock tree will be verified during Block Configuration described below.

The design team may provide gate level netlists containing ports and wires to implement feed-thru paths to reflect the physical layout of the chip. In a multi-system emulation, the partitioning of the design is based on minimizing the inter-system connections while maximizing the capacity of each system. It is very possible that the floorplanning of the emulation design is radically different from that of the real chip; unrelated design blocks may co-exist within a single system while tightly related blocks may be split between systems for capacity reasons. Since the feed-thrus are designed to reflect the silicon floorplanning rather than the emulation floorplan, surprising results can occur. For example, consider a bus that passes from one block through its neighboring block to a destination block. What if the neighboring block is resident in a different system than the source or destination block? The bus could be routed via external cables from the source system to the system containing the neighboring block, then routed via different external cables to the system that contains the destination block. In this example, a feed-thru forces a bus to route via cables to an external emulation system just to satisfy the designed feed-thru. The solution is to flatten design feed-thrus so the emulation configuration sees only direct connections from source block to actual destination block.

Repeaters can lead to the same problems as feed-thrus. Since emulation does not make use of or need repeater devices, they should be flattened from the design for emulation.

Emulation does not support pre-charge logic, pass-thru latches, or domino logic. They must be redesigned into a static representation. Such redesign requires careful verification to ensure that the emulation static model produces a cycle accurate functional representation of the original logic.

Other possible netlist problem areas need to be identified and resolved before full chip configuration begins.

Megacells and Large Memory Array Megacells. Megacells (custom circuits) come in two basic flavors: those containing large memories and those that do not contain large memories. The general discussion on megacells applies to both types while there are additional complications to large memory array megacells.

Gate level descriptions of megacells may not be available early enough in the project to start the emulation effort. However, there must be an RTL level description of each megacell used for simulation. It may be necessary to create an emulation local gate level description for each megacell. These local gate models may ultimately replaced by the actual gate level description from the design team.

Megacells often have creative internal clocking structures that cause problems for emulation (see discussion on low-skew nets above). Frequently, such creative clocking is related to scan control and can be removed or redesigned. Other creative clocking issues must be reviewed and reimplemented for emulation.

Memory arrays are implemented in emulation using external Memory Emulation Modules (MEM cards). Megacells containing MEM arrays require the additional step of partitioning the design into a control portion and a memory portion that maps to one or more MEM cards. Some amount of emulation specific logic may be needed to glue together the control portion of the megacell to the MEM card array.

All the above reimplementation of the megacells must be carefully verified for cycle accurate functionality with respect to the original megacell.

Block Configuration & Quickwave. In previous sections, there have been several statements about verification of certain modifications to ensure cycle accurate functionality with respect to the original design. The method used to perform this verification is to apply simulation generated test vectors to an emulation of the equivalent block of logic.

Before beginning to configure the entire chip, the emulation team configures, in isolation, each individual design block and selected parts of the design hierarchy. Once configured for emulation, these individual chunks of logic are test vector verified to ensure that the top level outputs from the logic are cycle accurate with respect to the simulation model.

Mismatches indicate a problem with the emulation model. This could be a bad library element, an incorrect netlist modification, or an incorrectly redesigned megacell.

Verification at the block level provides early detection of model problems. This provides faster turn-around while debugging any such problems and much greater confidence in the overall emulation model of the design. Once all blocks pass test vector verification, the full chip configuration begins.

B. Full Chip Configuration Phase

Full Chip Configuration combines the design netlists and libraries with control and specification files for delivery to the vendor supplied configuration software. The purpose of the configuration step is to produce a database that can be directly downloaded to program the emulation hardware.

Pre-Configuration Preparation created the set of netlists and libraries for use during configuration. The emulation team also provides several important control files and specification files to guide the configuration process. Many of the control files identify the external interfaces to the emulated design, including Pod Pin maps, External Logic Analyzer (LA) probes, MEM card locations, MEM card pin maps, etc. Other control files define netlist names/paths, logic analyzer probes, clock tree definitions, Timing Analysis (TA) exclusion information, etc.

One of the most complex tasks of this phase is to determine the optimum partitioning of the overall design to fit within the capacity of the emulation systems available while minimizing the number of cables required to route signals between systems. This can be a delicate balance with potentially great cost if a non-optimum solution is derived.

Vendor supplied tools partition the design into the individual systems. A new top level netlist produced by these tools binds together the individual partitions and defines the system-to-system cabling. Once the system level partitioning is complete, individual configurations are performed on each partition to create the downloadable database. In the case of UltraSPARC-I, five emulation systems from Quickturn Design Systems are used (see Fig. 2) and 50 system to system cables required (see Fig. 3); less optimum partitionings could easily require 60 or more cables to implement.

Each individual partition must be fully configured. These can be performed in parallel for all partitions as they are independent configurations. A configuration operation contains several steps:

- Parse netlists
- Semantic analysis and logic optimization
- Clock tree extraction
- Design partitioning within the system
- Logic mapping for each individual Field Programmable Gate Array (FPGA) within the emulation system
- Timing analysis on the design as implemented within the emulation hardware
- Delay insertion fix-up for hold violations
- Final timing analysis

If this sounds like a long compute intensive process, it is. The wall clock flow time for an entire UltraSPARC-I chip configuration is 36 hours using 5 large fast computers to do the main work and using 70 computers during the logic mapping for all the FPGAs. Automating that entire flow was required to ensure reliable execution with a minimum of human error.

C. Testbed Preparation

The testbed is a collection of hardware and lab equipment providing an environment in which the design under emulation can operate. This collection covers three major areas: target In-Circuit Emulation board, External Logic Analyzer, and supporting lab equipment (see Fig. 4).

The purpose target ICE board provides real world hardware support to the emulation. Typical components on a target board include: SIMMs, PROM socket, I/O Ports, etc. In addition to the real world hardware, the target board provides

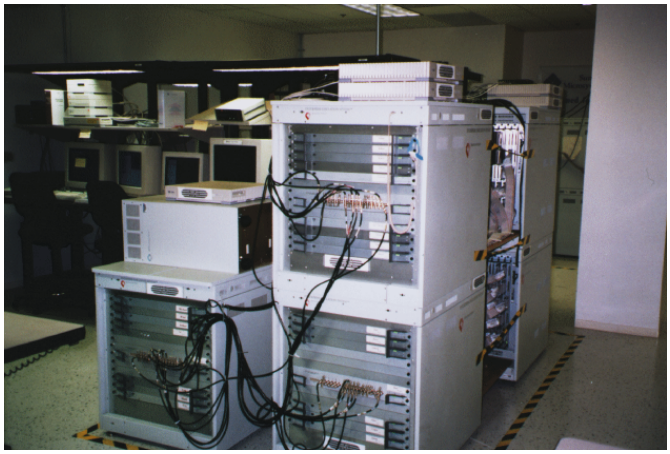

Fig. 2. Five Quickturn Design Systems Enterprise model 330 systems: Two stacked Enterprises in the background on the right, two stacked Enterprises in the foreground in the center, and a single Enterprise on the left side. The black cables between the systems provide clocking. Each Enterprise system is hosted by a Sun SPARCserver 10 (visible atop each Enterprise). This design is approximately 1.1 million emulation gates.

Fig. 3. The UltraSPARC-I target ICE system is shown on the left. The cables extending from the interface pods (left center) attach to the Enterprise systems. Notice the cables that connect between the Enterprises. In the foreground on the right is an InterModule chassis with several cables, this unit provides additional system-to-system signal interconnections. Combining the logic analyzers, over 5000 nets are typically probed.

sockets for emulation Pods that carry the top level I/O pins from the emulated design. Care must be taken to plan the physical and mechanical design requirements for the Pods including pod header spacing.

The target ICE board also provides headers to connect the external logic analyzer probes. These external probes supplement the visibility into the design provided by the internal logic analyzer inside of each emulation system. All totaled, over 5000 nets were routinely probed (128K depth) on the UltraSPARC-I project. A complete strategy for integrating the internal emulation logic analyzer samples with external logic analyzer samples was developed by the emulation team.

In addition to the target ICE board and emulation equipment, other basic lab equipment is necessary: pulse generators, oscillascope, etc. Other important lab issues must be considered including physical setup, electrical requirements and air conditioning.

D. In-Circuit Emulation Phase

The In-Circuit Emulation Phase combines the design emulation database with the testbed (target board and emulation hardware) to perform design verification. In addition, critical software must be available at ICE power-on. The emulation project must coordinate with and ensure that PROMs and Software deliveries are adjusted to fit emulation schedules which are earlier than silicon schedules.

The lab debug and bring-up effort requires a broad range of knowledge and skills. A team of people must be identified to provide this debug expertise during the bringup.

With the integration of the testbed hardware, emulation hardware, design emulation database and software, the design is downloaded into the emulation hardware, reset is released and it just boots Solaris--easy, right!? Unfortunately, like any significant hardware bring-up, every part has to be systematically debugged and fixed where needed. After a long (and often painful) bring-up experience, the Solaris “login” prompt appears on the monitor.

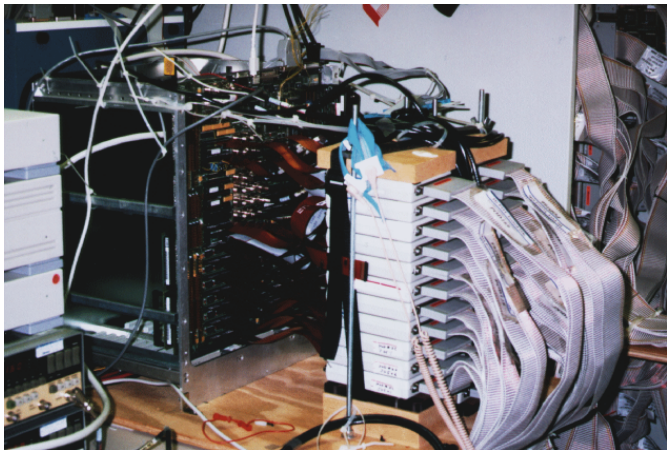

Fig. 4. Closeup of the target ICE board and pods. The single PCB standing vertically in the chassis was custom designed for exclusive use by emulation. It provides access to memory SIMMs, I/O ports and supporting logic. On the right side are two columns of interface pods which pass signals between the emulation and the target board. External disk drives and pulse generators are on the extreme left.

IV. Emulation Results

During the life of an emulation project there are three distinct phases during which value is returned: pre-tapeout, post-tapeout to pre-silicon, and post-silicon. These phases align with the silicon events of first tapeout and first silicon returns from the foundry. During each phase, emulation plays a slightly different role.

Based on past experience with the MicroSPARC II and SuperSPARC II processor emulation projects, a relative value for each of the phases was derived. Experience shows only 25% of the total return on emulation investment is derived during the pre-tapeout phase with the remaining 75% value returned later in the project. All three phases have significant impact on the ultimate delivery of the product to market.

A. Pre-Tapeout Phase

The pre-tapeout phase provides about 25% of the overall return on investment for an emulation project. The main source of value is in the additional functional design verification that can be completed by the emulation testbed prior to tapeout.

The testbed for UltraSPARC-I successfully emulated the chip at execution speeds of several hundred kilohertz--many orders of magnitude faster than simulation in terms of instruction cycles per wall clock second. This makes it practical to boot single-user Solaris in about 1 hour and 30 minutes.

Prior to tapeout, UltraSPARC-I successfully booted single-user Solaris with all CPU features enabled.

During the bring-up of Solaris on the emulated UltraSPARC-I, one serious design bug was encountered that prevented successful booting. Through the enhanced visibility into the design provided by emulation, this problem was isolated to a simple code sequence that under the right circumstances caused a major problem. This problem was recreated in simulation. A design fix was made and verified in both simulation and emulation.

In addition to the UltraSPARC-I bug, a significant number of bugs were found and fixed in the POST (Power-On Self-Test), boot PROM and kernel. These bug fixes were rolled forward into the silicon bring-up saving significant debug time.

B. Post-Tapeout to Pre-Silicon Phase

The window of time between tapeout and silicon return provides an opportunity to continue the emulation effort and increase the overall return on investment. This phase contributes another 25% to the overall return.

During this short window of time the emulation team focuses on additional design verification via stress testing. Efforts during this phase on UltraSPARC-I focused on Multi-User Solaris and OpenWindows in addition to a large assortment of applications and stress tests.

If additional problems are found during this phase, small fixes in the metal masks will improve the overall quality of the first silicon. No problems were discovered during this phase for UltraSPARC-I.

C. Post-Silicon Phase

Following silicon return, emulation takes on a supporting role. When a functional problem is encountered in silicon, emulation may prove to be a more friendly environment in which to debug with the available internal visibility into the design via the logic analyzer. Once a bug has been isolated and a fix developed by the design team, emulation includes the fix in the database and verifies that the problem has been resolved. A suite of regression tests helps to insure that new problems have not been introduced. Approximately 50% of the total return on investment is derived in this phase.

During this phase emulation can provide two additional services: what-if experiments and support for follow-on tapeout verification.

Designers often want to try some design change or enhancements to see what effect it will have on the real system. Prototyping such what-if changes is relatively easy for emulation.

Just as with the first tapeout, the design team desires confidence in the design for any follow-on tapeout events (metal-mask changes, all layers changes, major design revision, etc.). The emulation provides additional confidence in each updated design by booting Solaris and running the suite of regression tests.

V. BENEFITS

Emulation supplements existing design verification techniques providing additional capabilities not available through any other technology. Confidence in the design prior to silicon commitment is improved due to the additional verification accomplished by emulation. Applications can be executed in emulation that are impractical to simulate due to execution duration. Problems discovered and fixed before tapeout improve the quality of the first silicon with a potential to save silicon iterations.

Once silicon is available, bring-up benefits from the pre-tested software environment available as a result of emulation testing. The advanced debugging capabilities in emulation reduce the time necessary to isolate functional design bugs. Design fixes and changes are rapidly prototyped in the emulation system before silicon commitment.

Emulation helps reduce the time-to-market of the complete product increasing life-time revenue and profit potential.

REFERENCES

[1] D. Greenley, *et. al.*, "UltraSPARC: The Next Generation Superscalar 64 bit SPARC", 40th annual Compcon, 1995.

[2] Larry Yang, "System design methodology of UltraSPARC", 32nd Design Automation Conference Proceedings (in press).

[3] Ariel Cao, *et. al.*, "CAD Methodology for the Design of UltraSPARC Microprocessor at Sun", 32nd Design Automation Conference Proceedings (in press).

[4] Marc Tremblay, "A fast and flexible performance simulator for microarchitecture trade-off analysis on UltraSPARC", 32nd Design Automation Conference Proceedings (in press).

[5] S. Mehta, *et. al.*, "Verification of the UltraSPARC Microprocessor", 40th annual Compcon, 1995.

[6] Palnitkar, Saggurati “Finite State Machine Coverage program” Open Verilog International Conf. 1994.

[7] L. Kohn, *et. al.*,"The Visual Instruction Set (VIS) in UltraSPARC", 40th annual Compcon, 1995.

[8] C. Zhou, *et. al.*, "MPEG Video Decoding with UltraSPARC Visual instruction Set", 40th annual Compcon, 1995.

[9] James Gateley, "Logic emulation aids design process", ASIC & EDA, July 1994.

[10] James Gateley, Miriam Blatt, "Reducing time-to-emulation through flow automation", Nikkei Electronics (in press).