INNOVATIVE VERIFICATION STRATEGY REDUCES DESIGN CYCLE TIME FOR HIGH-END SPARC PROCESSOR

by
Val Popescu
Metaflow Technologies, Inc.
La Jolia, CA
and
Bill McNamara
Zycad Corporation
Irvine, CA

Abstract

Superscalar processor developers are creatively leveraging best-in-class design verification tools to meet narrow market windows. Accelerated simulation is especially useful owing to its flexibility for verifying at many points during the design cycle. A unique "verification backplane" makes continuous verification at any level(s) of abstraction available to each design team member throughout the design cycle.

Introduction

The use of verification methodologies at many points throughout the design cycle is becoming a stock in trade for microprocessor and other IC design teams. The complexity of submicron CMOS chips with millions of transistors makes relying solely on traditional late stage verification a thing of the past. Only by iteratively verifying their IC designs can developers keep them on track with design goals, and on schedule in a competitive market.

A commitment to verification is an especially important strategy for smaller companies that lack their own foundry. Their resources allow little room for error. A design must be fabricated successfully with a minimum number of chip turns. Multiple silicon re-spins can spell disaster by delaying a chip's introduction and incurring cost overruns. A well-implemented verification strategy minimizes the likelihood of these problems, and acts as an equalizer in a small company's drive to win market share against Fortune 500 rivals.

In developing our Thunder SPARC processor -- a high-performance, moderately priced CPU compatible with the SPARC Version 8 specification -- we at Metaflow Technologies relied heavily on design verification. We used several verification vehicles in a regression suite [Table 1]. We recognized at the outset that a comprehensive verify-as-you-go approach was necessary if we were to have any chance of ferreting out previously unencountered and progressively insidious bugs in a processor with nearly six million transistors. To that end, we implemented a verification scheme that would enable us to run as many clocks as possible in as short a time as possible.

Our verification strategy was critical for meeting a very narrow time-to-market window in a highly competitive market. It enabled our small SPARC development team to take the processor from a Verilog model to tape-out in just nine months. Subsequent post-silicon design validation took eight months. By contrast, other recent high-end microprocessor development projects have involved much larger design teams working for up to three years.

Successive Lines of Defense Against "The Logic Bug"

Sim/Emulator (DUT)	Test Suite	No. of Clocks/Sec	Run Platform	Run Time
VCS (System)	Regression	10	50 workstations	16 hrs.
A-Sim	SPEC92	1,000	4 workstations	35 hrs.
Zycad (IU Chip)	Regression		1 Paradigm Box	4 days
Quickturn	Regression	500,000	1 M3000 Box	60 sec

Table 1

Best-in-Class Verification Tools

The products comprising our verification suite were in keeping with our strategy of using best-in-class tools. The regression suite was run on:

- VCS Verilog simulator from Chronologic (since acquired by Viewlogic, Inc.) for system-level verification
- TimeMill from Epic for block-level verification
- HSPICE circuit simulator from Meta-Software, Inc., for verification of selected transistor circuits
- Paradigm XP Model 2016 logic and fault simulation accelerator from Zycad Corp. for running accelerated structural (gate/transistor) level simulations for chip-level verification
- System Realizer M3000 from Quickturn Design Systems for running in-circuit emulations (sometimes also referred to as "rapid prototyping")

It should be noted that once we established these tools as stable and productive, we generally did not avail ourselves of software upgrades from the vendors. In our experience, the benefits of new tool features and enhancements generally do not outweigh the time and effort required to get up to speed with new releases on the fly. Thus, for example, we chose not to implement what would have been a useful modular compilation feature on the Zycad simulation accelerator which became available in mid-project.

Innovative 'Verification Backplane'

In what we believe is an important innovation in the area of design verification, we were able to integrate most of these otherwise distinct modalities into a single "verification backplane" [Figure 1]. This backplane makes continuous verification at any level(s) of abstraction available to each design team member throughout the design cycle. The backplane interface creates the appropriate netlist for use with any of the verification tools, such as the Zycad accelerator, Verilog simulator, TimeMill, HSPICE circuit simulator, or the Quickturn emulator. These testbeds are all drawn from the Golden Verilog by capturing the inputs and outputs of the signals at the boundary of the defined block (core chip).

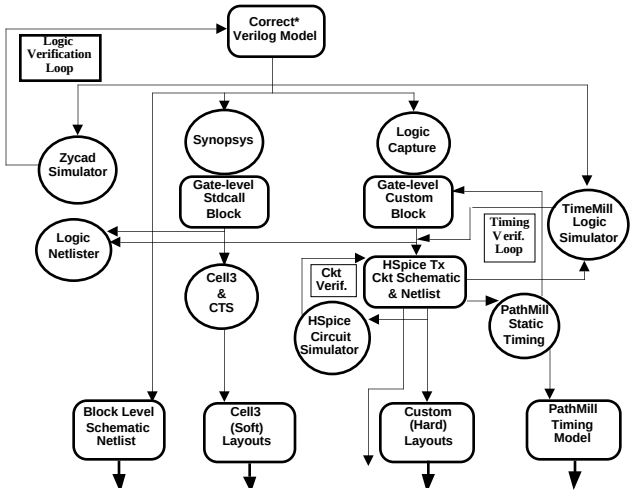

Figure 1. Block Level Design & Verification

For example, a designer who wants to verify the register file block at the transistor (switch) level target the Zycad accelerator. The backplane translates the input and output vectors captured in Verilog at the designated block boundaries into stimulus and output test vectors for the simulation accelerator. The output is compared with the same vectors in the Verilog model. Mis-compares flag the designer that the transistor schematic simulated in the accelerator does not respond in the same way as the Verilog model.

If the designer is satisfied that the transistor netlist is logical – i.e., binarily correct and the same as the Verilog on a clock-by-clock basis -- he or she can verify further, say, by applying TimeMill to verify the transistor schematic with timing information.

This kind of verification can be done continually. Of course, the tools with the highest accuracy, which run the slowest (such as TimeMill), tend to be used more toward the end of the cycle. At that point, because fewer bugs remain, the tool can still be effective running without running a high number of cycles.

We maximize the productivity of our verification tool suite by using the UNIX demon "Basher" to automatically generate and run test programs whenever a workstation is idle or its CPU cycles have fallen below a certain level. Thus we are running tests day and night that compare various models -- e.g., Verilog, Zycad, Quickturn -- and report any mismatches. Verification never stops.

High-density SPARC CPU Architected for High Throughput at Moderate Clock Rates

The Thunder SPARC processor, which will begin shipping in quantity to customers in Q1 1996, is a full custom, 0.5 micron CMOS microprocessor for SPARC Mbus high-end workstations and parallel processing servers [Figure 2]. Although its clock frequency is only 80 MHz, the processor achieves high throughput by doing a lot of work on each clock cycle: three integer, two floating-point and one branch instruction can be issued in a cycle. This is made possible by an architecture that provides a large number of function units working in parallel, out-of-order instruction issue, and heavy use of speculative execution. The latter is an architectural innovation pioneered by Metaflow Technololies ten years ago, and is now being implemented by a number of microprocessor vendors. With these features, the Thunder SPARC chip set provides 200 SPECint92 and 350 SPECfp92 performance.

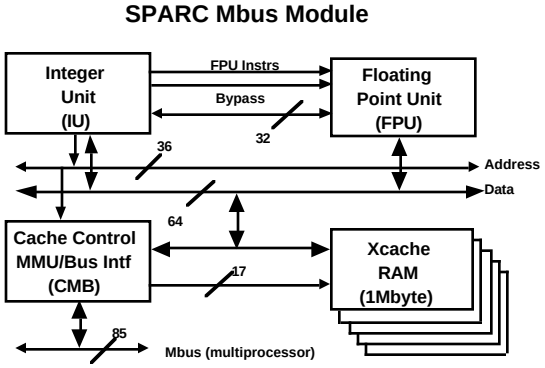

Figure 2. Block Diagram

The processor is packaged as a three-chip set using ball-grid-array (BGA) technology. An integer processor (IU) packs 3 million transistors on an 11.85 mm by 11.85 mm die. A floating-point unit processor (FPU) implements 1.0 million transistors on a 9.05 mm by 9.0 mm silicon die. And a cache controller/MMU/bus interface (CMB) uses 1.8 million transistors, also on a 9.05 mm by 9.00 mm die. In addition to on-chip instruction caches, the chip set employs a 1-Mbyte external cache, fabricated from commodity 9-ns synchronous SRAMs.

The three processor chips and cache fit together on an MBus CPU board, making the Thunder SPARC available as a plug-in upgrade to existing SPARCstations. The processor's very high performance at relatively low clock frequency -- and hence moderate system cost -- make it ideally suited for that market.

Verification Throughout the Design Flow

Verification is a theme that runs through the lifecycle of the Thunder SPARC processor [Figure 3]. Every verification tool was used more or less continuously once the design reached a stage where the tool was applicable. Thus Verilog simulation was a constant throughout the cycle. Later, accelerated simulation using the Zycad Paradigm XP began, and then

continued. Likewise, the Quickturn system launched into action when it was time to begin in-circuit emulation, and it continued as well. Several modes of verification were always running in parallel.

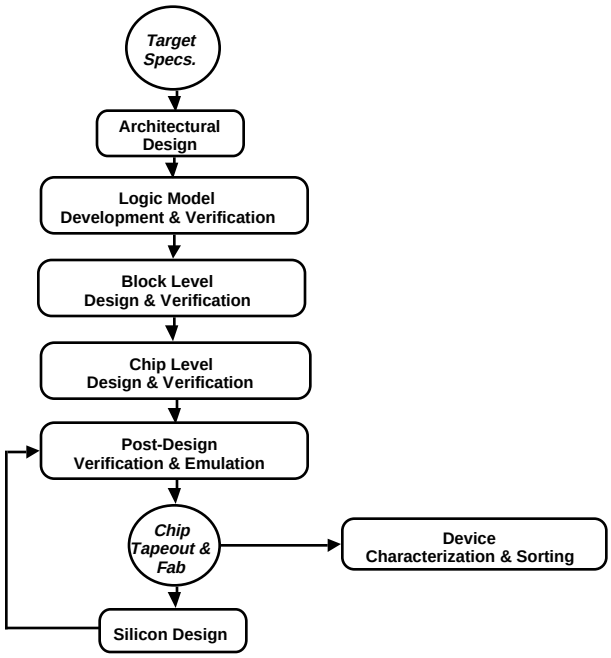

Figure 3. Design Flow

Verification begins early on, in the Architectural Design stage, using a C-language microarchitecture simulator (called "A-Sim") we developed to verify the potential performance of the architecture defined in the specification. The A-Sim tool provides the original reference model for all subsequent verification; i.e., it is used to verify the Verilog model, which is in turn used to verify the Zycad simulation accelerator model, which is then used to verify the Quickturn emulator model, etc.

Once the architectural spec is translated into Verilog high-level design language blocks, each block is simulated using the VCS Verilog simulator with Behavioral Models (BMODs) as a "Golden" simulator [Figure 4]. The Verilog RTL source netlist is downloaded into the simulator and run against Target, Random, and Instruction Set Architecture (ISA) test vectors.

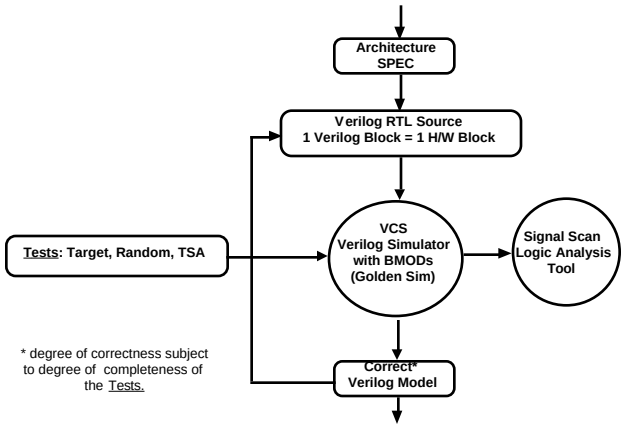

Figure 4. Verilog Model Development and Verification

At the Block Level Design and Verification stage [Figure 1], we created a 'backplane' into which several state-of-the-art point tools from different EDA vendors could be linked to create a coherent design methodology. Here, logic simulation is performed using VCS and the Zycad Paradigm XP simulation accelerator. Transistor-level circuit evaluation (including timing) was performed using TimeMill and the Meta-Software HSPICE circuit simulator.

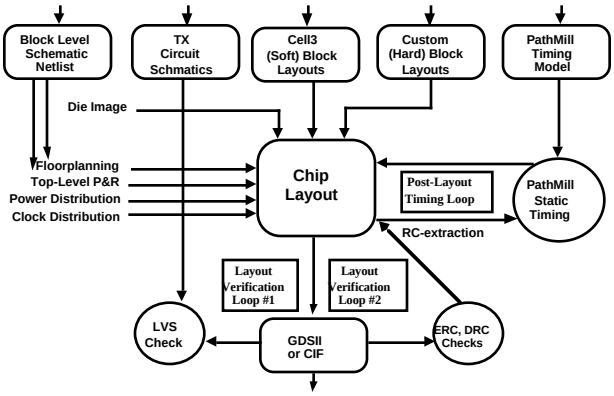

Figure 5. Chip Level Design and Verification Flow

The logic netlist for the Quickturn emulator is verified using the Zycad simulation accelerator before being translated into the FPGA emulation netlist. This ensures that the FPGA model behaves identically to the Verilog logic model since a testbed previously sent to the Zycad accelerator was verified against the Verilog model [Figure 6. Post-Design Verification via Emulation]. As a result, this approach eliminates the possibility that logic bugs that would be created along with the gate-level netlist for the Quickturn emulator. We refer to this use of the simulation accelerator in conjunction with the emulator as "closed loop verification," and we believe it is another valuable innovation that has grown out of the Thunder SPARC program.

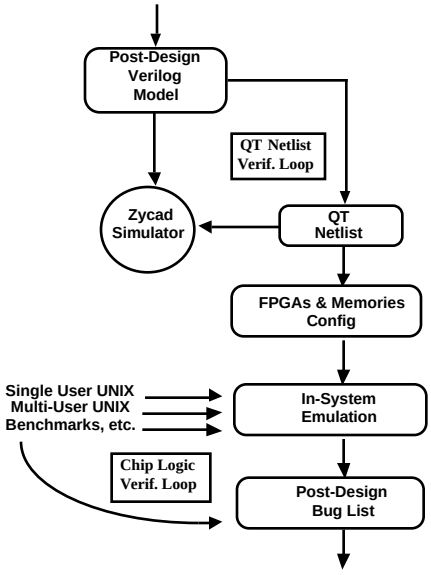

Figure 6. Post-Design Verification via Emulation

Simulation and Emulation: The First and Last Lines of Defense Against Bugs

It is clear that a verification strategy for a superscalar custom processor such as the Thunder SPARC must include both simulation -- particularly accelerated simulation -- and emulation. Simulation provides a first line of defense against a multitude of bugs [Figure 7. First Line of Defense].

The Zycad Paradigm XP simulation accelerator played an important verification role for us in detecting the "first 1,000" bugs. It could do this by delivering 10 to 20 times the performance of the fastest Verilog simulation software on today's most powerful workstations. It also allows the use of switch-level (transistor) circuit models for accurate representation of the design. The Paradigm XP 2016 accelerator we used is a 16-board configuration which offers capacity expansion to 16 million gates for logic simulation and four million gates for fault simulation.

The accelerator also has an interface from Verilog which enables our design engineers to co-simulate behavioral and RTL Verilog with accelerated gate and transistor level modules. This was another important area of innovation for us during the development of the Thunder SPARC: we built this co-simulation backplane to the Open Verilog simulation interface provided by Zycad. We took Zycad's generic accelerated simulator, which was intended for gate-level simulation only, and developed a method for switching individual blocks from one level of hierarchy to the other. In this way, the accelerator works only on blocks that have recently changed. Everything else is frozen at the behavioral level. The interface we developed is fairly universal in that it allows us to create testbeds for our Epic TimeMill simulator, HSPICE tool, and even our HP 83000 tester to test the real silicon. As a result, it represents a tremendous productivity enhancement.

The accelerator also provides a high-performance fault simulation capability. When it is not running logic simulations, the accelerator is in constant use grading vectors for use by our foundry and for incoming receiving inspections.

Emulation Detects the Most Elusive Bugs

As bugs are eliminated through simulation, the number of simulation clocks required to detect remaining bugs increases. At some point, simulation alone ceases to be useful: detection of many timing-dependent bugs, for example, may require a number of clocks that is beyond the practical limit of simulation. Moreover, the simulation model may not be sufficiently accurate to detect some bugs.

Finding the last few dozen bugs requires emulation. Typically, these are bugs that "hide" behind other bugs or are the results of previous bug fixes. As a result, the convergence toward a condition of "zero bugs" resembles a damped oscillator. Debugging using an emulator provides a much faster convergence toward a bug-free state. This is because a well-modeled emulation provides virtually the same degree of accuracy in debugging as fabricated silicon.

As mentioned earlier, we undertook an in-house integration effort that significantly enhances the emulation process by leveraging our simulation accelerator into it. We initiated this development work on the premise that simulating the emulation gate-level netlist before it was translated into the emulator's FPGA netlist offered two benefits: It would tend to identify errors created in the process of translating between the gate-level netlist and the emulation netlist. And it reduced the chance that bugs would be masked and not introduced into the emulation netlist, with the result that they would not be detected in the emulation. We believe this approach to using simulation and emulation together in a closed-loop fashion marks an important advance in verification methodology.

Conclusion

Design verification as we have implemented it in our Thunder SPARC program is now a way of life at our company. Our next design, as well as upgrades to the Thunder SPARC, will use this same methodology, but with one important difference: we will significantly expand our simulation and emulation gate capacity.