Verifying the Correctness of the PA 7300LC Processor

Functional verification was divided into presilicon and postsilicon phases. Software models were used in the presilicon phase, and fabricated chips and real systems were used in the postsilicon phase. In both phases the goals were the same—to find design bugs and ensure that customers get the highest quality part possible.

by Duncan Weir and Paul G. Tobin

Ensuring the correctness of the complex PA 7300LC design required an extensive verification effort. We wanted to ensure that no customer would ever encounter a design bug. To reach this goal, we set out to exercise the design more extensively than is done with user software. Previous HP processors have maintained a well-earned reputation for quality, and we wanted the PA 7300LC to meet or exceed the quality of its predecessors.

This paper discusses the methodology used to verify the correctness of the PA 7300LC and the diagnostic hardware incorporated into the design to support debugging.

Functional Verification

The functional verification effort was divided into presilicon and postsilicon phases. The presilicon phase involved creating a software model of the chip and an environment in which the model could be thoroughly tested and debugged. The modeling environment provided many features to aid verification including the ability to initialize the machine state, inject stimuli, and see into all portions of the design for debugging. One major drawback of the modeling environment was the slow simulation speed.

Complementing the presilicon effort, an extensive postsilicon verification program was completed that took advantage of the test throughput available when running on an actual computer.

Extensive testing of the physical circuit design of the PA 7300LC was done in presilicon and postsilicon environments to ensure that the circuits would meet frequency, voltage, and temperature targets. This topic is covered in ***Article 8***.

Presilicon Verification

For better efficiency, we chose to divide the design of the PA 7300LC into two components: the CPU core and the memory and I/O controller (MIOC). These two portions of the design were logically separated by a well-documented interface that enabled us to verify each component independently. Verifying the two components independently provided several benefits:

- Smaller and faster models
- Precise control over the stimuli at the CPU-MIOC interface
- Simpler model management (because less coordination was needed)
- Reduced debugging time (since it was known which portion of the design contained the bug).

As the design neared completion and both the CPU and MIOC had been extensively verified, we created a single merged model that included both components. This provided a thorough check of the interface between the components and was a double check of the independent verification work. In addition, the MIOC was incorporated into a model with external I/O devices to ensure that the PA 7300LC design would work with the components needed for a complete computer system.

The presilicon verification environment consists of three parts: modeling environment (model), test case environment (stimuli), and checking environment (checks).

Modeling Environment

We modeled the PA 7300LC design using the Verilog hardware description language. The design was primarily modeled at the logic gate level with connectivity extracted from the physical design. Some key portions of the design like the caches, TLBs, and floating-point execution units were modeled at a higher level to improve the size and speed of the model.

Fig. 1 shows the CPU and MIOC modeling environments. Software emulators were connected to the model interfaces to provide input and respond to output from the model. The programmable nature of the emulators allowed test cases to exercise the interfaces fully.

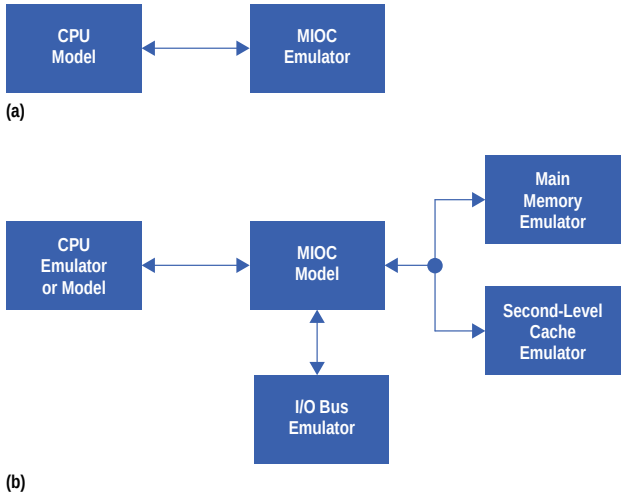

Fig. 1. *Presilicon verification modeling environments. (a) CPU modeling environment. (b) Memory and I/O controller (MIOC) modeling environment.*

New Modeling Process. Managing the modeling environment of a large design is a time-consuming task requiring coordination among all team members. Problems with a model build could lead to downtime that would stall the verification effort. To minimize these problems, a new model building process was implemented for the PA 7300LC design. All blocks of the modeling environment were placed under revision control. Any changes had to be included in a process change order that documented the purpose of the change, the blocks affected, the dependencies existing between this and other process change orders, and the testing needed to verify the change. In addition, an automated model build procedure was put in place to allow designers to integrate their changes into a private copy of the model and verify them in isolation before submitting a process change order. Finally, before a model was released to the verification team, it would undergo regression testing to eliminate blatant errors. Using the new system resulted in a consistently stable model that accelerated the verification effort.

Test Case Environment

Test cases control the stimuli applied to a model, thereby providing the event interactions that stress the design. Having an efficient way for test cases to stress the entire design is an important factor for improving quality. The strategy used for the PA 7300LC was largely leveraged from the successful PA 7100LC effort.[1] It provided a simple way to initialize machine-state resources like registers, caches, TLBs, and memory. It also allowed high-level coordination of instructions executed by the CPU along with transactions occurring at the model interfaces.

Test cases for the PA 7300LC came from three sources: cases leveraged from the PA 7100LC, new cases focused on the PA 7300LC, and randomly generated cases.

Thousands of cases that were written to cover the PA 7100LC design were leveraged to run on the PA 7300LC. Most cases needed no modifications to be effective because of similarities in the designs of the two chips. For the portions of the PA 7300LC design that were different, new cases were produced. Some of these cases were written to focus on particular aspects of the design such as instruction-cache misses, the CPU-MIOC interface, and the second-level cache. Other cases were produced using random code generators that were designed to stress the PA 7300LC.

Random code generators are mainly employed for postsilicon verification, but the PA 7300LC team also emphasized their use for presilicon testing. Although challenges were encountered, the results were positive. Many subtle bugs that might not have been found until postsilicon testing were discovered early in the design process. Random code generators also provided an efficient way of achieving broad coverage with fewer engineers than other testing methods. See ***Subarticle 9a*** "Random Code Generation" for more on this topic.

Checking Environment

A modeling environment and interesting stimuli are only two pieces of the verification puzzle. The other critical piece is verifying the model's response to stimulation. On a complex design like the PA 7300LC, with many designers and tens of thousands of test cases, it would have been impossible to verify correct model behavior without aids to automate the process. As a result, a significant part of the PA 7300LC verification effort was spent creating software modules that automatically verified the model's response to events created by test cases.

Modules were compiled into the model to check that the MIOC followed the proper I/O bus protocol and to ensure that both the CPU and the MIOC followed the protocol at the CPU-MIOC interface. Checkers were also written to ensure that the memory controller obeyed proper timing protocol on the main memory and second-level cache buses.

CPU Testing. For the CPU core, we linked a PA-RISC architectural simulator to run synchronously with the model to ensure that instructions were executed as the architecture requires. When an instruction finished executing, the results were compared between the model and the simulator. A special module called a *depiper* was written to translate internal CPU signals into architectural events that could be checked by the simulator. After a test case finished, the model's final machine state was compared against the simulator's final machine state.

New Transaction Checker. Logically, the MIOC converts inbound transactions on one interface to outbound transactions on a different interface. For example, the CPU core might initiate a cache line copyin that the MIOC converts to a read on the memory port. When the memory supplies the data, the MIOC returns the cache line to the CPU. A special transaction checker, called the *metachecker*, was written to verify that proper transaction conversions occurred. The metachecker matched inbound transactions with their associated outbound transactions. Mismatched transactions were reported as failures.

New Cache Checker. The cache controllers for the PA 7300LC are among the most complex portions of the design. As a result, a checker was written to verify their operation. It monitored the instruction pipeline, the cache read and write ports, and the CPU-MIOC interface. Any incorrect behavior was detected and reported.

Ad Hoc Checks. Finally, a collection of small, ad hoc checks were included in our presilicon testing to cover things that might otherwise be missed. Some were signal-level checks (for example, checking that a set of signals were mutually exclusive), others were special checks required by test cases. Some checked that performance features such as the superscalar pipeline were operating correctly.

Together, the checkers formed a seamless net to ensure that incorrect model behavior would be detected. There was some overlap between the checkers. Many times a design flaw would get flagged by several of the checkers, but dividing the work between multiple checkers was an effective way to reduce the risk of a design flaw escaping detection, while allowing verification engineers to work in parallel.

Model Matches Physical Design. Once the physical design and the verification effort stabilized, we verified that the Verilog model matched the physical design. This was done by deriving a switch-level model from the actual chip artwork and running thousands of tests on both it and the Verilog model, comparing key signals on every clock phase of simulation.

Postsilicon Verification

Once the design is fabricated, the nature of the verification effort changes completely. The goals are still the same—to find the design bugs and ensure that customers get the highest-quality part possible, but the tools and the approach are different.

Test Systems. The environment for testing the design shifted from software models to real computer systems that included PA 7300LC chips. We set up a number of test systems, each of which could be controlled remotely from a host workstation using remote debugger software. The remote debugger provided us with the ability to load and run programs on the test system and to examine portions of the machine state. It also gave us complete control over the machine without any operating system layers obstructing our access to system resources.

Because the PA 7300LC is designed to work in a number of different system configurations, we set up systems that had different clock frequencies, cache configurations, and memory timing. To ensure that the design would work with a variety of different I/O cards, exercisers for the GSC I/O bus were created that could change their behavior to mimic any type of I/O card.

Random Code Generation. Random code generators are an efficient way to take advantage of the speed of postsilicon testing. With a small amount of human control, these programs can create millions of unique tests to exercise every nuance of a complex design. We used random code generation extensively on the PA 7300LC by employing six different generators. One targeted the floating-point design, one was directed at the MIOC, and four covered the entire chip operation.

Extensive Suite of Tests. We supplemented the random testing with an extensive suite of tests using I/O exercisers to stress the MIOC design. Many tests were leveraged from postsilicon testing of the PA 7100LC and were modified for the PA 7300LC. Additional tests were written to provide better coverage, especially for areas where the PA 7300LC design differed from the PA 7100LC.

Self-Checking Tests. The elaborate checking methodology from presilicon verification was of no use in postsilicon testing because it was not possible for the checking software to observe the design now embedded on a VLSI chip running in a system. To compensate, all of the postsilicon tests were self-checking. The generators that created the random tests also ensured that the chip responded properly to them.

System Test. A final element of the postsilicon testing was verifying that operating systems and application programs ran properly on computer systems built around the PA 7300LC. A large amount of testing was done by several different organizations within HP and included operating system reliability tests, benchmark programs, and key user applications.

Verification Results

The PA 7300LC verification work was a success. Presilicon testing eliminated over 800 design bugs, and more than 1200 process change orders were added to the model in one year. The quality of the first revision of the chip was very high. Only eight functional bugs were found in postsilicon testing. Of these, only one affected our design partners, and it had a simple workaround. The HP-UX* operating system was booted shortly after first revision parts arrived. Our postsilicon testing was far more extensive than what we had previously done with the PA 7100LC or its predecessors. The verification effort ensured that the PA 7300LC will maintain HP's reputation for quality processors.

Debug Support

The high level of integration on the PA 7300LC reduces the visibility into chip operation that aids in debugging prototype silicon. In particular, moving the primary caches onto the chip removed a valuable source of debug data while also introducing a new source of potential functional and electrical problems.

Since the MIOC, floating-point coprocessor, and TLB are also contained on the same die, the only external pads visible to debuggers were for the I/O bus and the memory interface. At the same time, the PA 7300LC had new challenges such as a large primary on-chip cache, a new IC process, higher operating frequencies, and a second-level cache. Debug support was important to improve the signal visibility and to reduce the risks associated with the new technology.

Debug Mechanisms

Signal visibility is of primary importance when debugging a failure, so several techniques were used to make internal signals accessible.

- Idle cycles on the GSC I/O bus were used to drive debug information.
- Seventeen special chip pads are dedicated to driving real-time debug information. To reduce cost, these pads are not bonded in production parts.
- Thorough implementation of IEEE 1149.1 and sample-on-the-fly (a scan technique invented for the PA 7100LC)[1,2] allowed a very broad, but only one-cycle-deep, snapshot of the chip state to be reported. Custom data capture hardware was designed to gather the debug traces and present them to a logic analyzer.

New Pattern Mapping Failure Isolation Technique

Traces captured from the debug ports can be overwhelming in size, making it difficult to isolate the failure. The PA 7300LC addressed this problem by implementing circuits to recognize internal chip state patterns. The patterns are programmed from software using special instructions implemented on the PA 7300LC, and the capture of debug traces can be predicated on a state pattern match. Debug traces are thus shortened to an interesting region. It is also possible to alter the program flow upon a pattern match, allowing a branch to diagnostic software to probe for a failure. By providing a flexible scheme for programming repeatable patterns, the task of isolating a failure and performing experiments to determine its root cause was greatly simplified.

Target Applications For Debug

Functional and electrical verification were the primary applications for which the debug circuitry was designed, but the debug features were general enough that they could be used to diagnose processor problems encountered during bringing up the operating system, firmware development, and benchmarking.

Electrical verification relies more extensively on debug hardware because failures cannot be reproduced in our software model of the CPU. Engineers working to verify a chip's thermal and electrical margins use debug features to investigate and understand failures occurring at extreme operating points.

Debug Features

The PA 7300LC debug features are intended to work in any environment used to test the CPU—wafer test, package test, and system test. The debug features are operable and portable across these environments. In addition, debug circuits were designed to tighter specifications than the rest of the PA 7300LC. This ensured that they functioned properly well into the operating regions where the CPU core is expected to fail. We achieved this through the use of simple logic and conservative timing budgets.

Although no major problems were found during qualification of the PA 7300LC, debug features were relied upon to help fix the problems that arose, helping us to achieve quick time to market for PA 7300LC-based systems.

Conclusion

The extensive verification of the PA 7300LC design was based on the successful strategy used for the PA 7100LC. Improvements were made in the model building process and in the extensive use of random code generation in the presilicon and postsilicon phases. Many features were added to the PA 7300LC design to allow efficient debugging of postsilicon failures. Together, these efforts ensure that customers get the highest quality part possible.

Acknowledgments

Many people contributed to the verification effort of the PA 7300LC including members of engineering systems laboratory in Fort Collins and the Integrated Circuits Business Division at the Fort Collins design center. Key contributions were also made by individuals from the Fort Collins systems laboratory, the computer technology laboratory in Cupertino, and the UNIX® development laboratory in Fort Collins.

References

1. M. Bass, T. Blanchard, D. Josephson, D. Weir, and D. Halperin, "Design Methodologies for the PA 7100LC Microprocessor," *Hewlett-Packard Journal*, Vol. 46, no. 2, April 1995, pp. 23-35.
2. *IEEE Standard 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Standards Board, May 1990.

▶ Go to Subarticle 9a

▶ Go to Next Article

▶ Go to Journal Home Page