Verification and Management of a multimillion-gate embedded core design

Johann Notbauer
Siemens, Austria
Erdberger Laende 26
A-1030 Vienna, Austria
+43 1 1707 36087
johann.notbauer@siemens.at

Thomas Albrecht
Siemens, Austria
Erdberger Laende 26
A-1030 Vienna, Austria
+43 1 1707 35850
thomas.albrecht@siemens.at

Georg Niedrist
Siemens, Austria
Erdberger Laende 26
A-1030 Vienna, Austria
+43 1 1707 37925
georg.niedrist@siemens.at

Stefan Rohringer
Siemens Semiconductors
Babenberger Strasse 10
A-8020 Graz, Austria
+43 316 7271 100
stefan.rohringer@siemens-scg.com

ABSTRACT

Verification is one of the most critical and time-consuming tasks in today's design processes. This paper demonstrates the verification process of a 8.8 million gate design using HW-simulation and cycle simulation-based HW/SW-coverification. The main focuses are overall methodology, testbench management, the verification task itself and defect management. The chosen verification process was a real success: the quality of the designed hard- and software was increased and furthermore the time needed for integration and test of the design in the context of the overall system was greatly reduced.

Keywords

HW/SW-coverification, cycle-based simulation

1. INTRODUCTION

The Siemens Switching System EWSD Innovation program (EWSD Power Node) includes the implementation of a message buffer (MB) which is part of the EWSD coordination processor and acts as a signaling channel controller and message router. Design requirements include maximizing traffic capacity and being able to customize product derivatives through hardware and software configuration.

Since the Siemens EWSD switch is designed for incremental expansion in processing power, connectivity and services, the system architecture is modular, enabling different configurations.

1.1 Hardware structure

The message buffer hardware consists of

- 3 different types of ASICs, each containing a RISC core, memory controllers and ASIC specific interfaces, equivalent gate count in the range from 270k to 420 k gates
- 3 different types of boards, each containing a number of one type of ASIC and off-the-shelf components implementing the following interfaces: 200 MHz ATM, 8xHDLC serial links and a proprietary system interface
- 1 type of backplane which interconnects multiple instantiations of the three types of boards

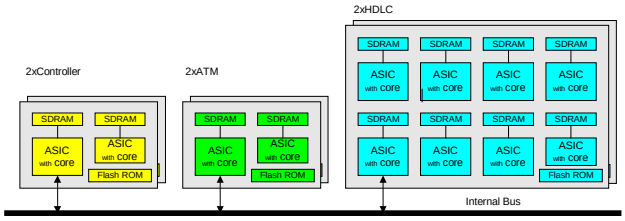

Figure 1: Minimum configuration of the MB

The MB will be realized with 2 redundant halves running independent and sharing their loads. The minimum configuration (see Figure 1) consists of 2 HDLC, 2 ATM and 2 controller boards, the maximum configuration consists of 16 HDLC, 10 ATM and 2 controller boards. This scaleable multiple-processor platform is used to ensure the flexibility with respect to capacity and still keeps the number of different hardware-solutions low.

1.2 Software structure on the message buffer

From the point of view of SW development, the message buffer presents itself as a cluster of RISC processors, each equipped with memory (DRAM, caches, on-chip-memories) along with some communication interface HW. Since the 3 types of message buffer ASICs are almost identical apart from their specific communication interfaces, it is natural to reuse SW components across ASICs.

The basic message buffer SW structure is shown in Figure 2. The shaded areas denote the SW components reused on every ASIC.

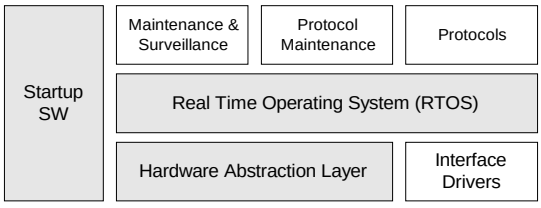

Figure 2: Message buffer SW structure

The message buffer SW consists of a hardware abstraction layer and utility package, a real-time operating system (RTOS), various interface drivers, a startup SW package and an Application SW.

DAC 99, New Orleans, Louisiana

1.3 Simulation strategy

Based on a benchmark [1] done at the beginning of the project it has been decided to use HW/SW-coverification. It enables SW engineers to start HW/SW integration already on a virtual base with no need to wait for physical hardware prototypes.

In a traditional approach HW and SW development start in parallel after a common specification phase with a small delay but with only a few interactions (see Figure 3). With the availability of physical HW prototypes the integration starts from scratch with quite often many ”simple” but time consuming errors to fix on both sides.

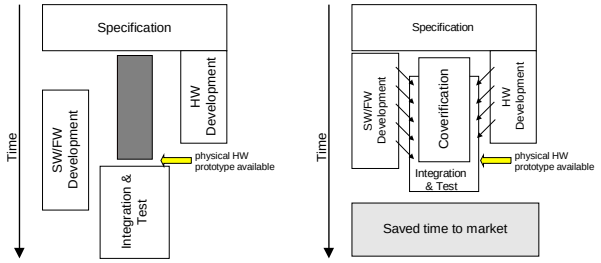

Figure 3: The traditional approach vs HW/SW coverification

Using HW/SW-coverification the "wall" between the HW and SW-community is never built at all (Figure 3). After the specifications have been agreed upon both teams work in parallel. As soon as some initial functions are realized in both HW and SW integration can start right away on a simulation platform with reasonable performance.

The ASIC-team provided their modules with a certain, not necessarily complete functionality at an early stage for HW-system-simulation. Not very much later - still a while before all functionalities have been implemented - the available code was used by the HW/SW-coverification as a virtual base for their cosimulations. This approach led to a higher number of intermediate ASIC-deliveries and regression tests and to enormous feedback to ASIC team and the chance for HW modifications driven by FW engineers.

To enable this closely coupled verification steps a strict and elaborate planning and scheduling for the implementation steps of the different functionalities in hard- and software is essential. On the bottomline this leads to a higher verification coverage earlier in the design process for the cost of some additional planning and supervision.

The functional verification of the hardware has been carried out by three distinct groups of engineers, each with a focus of their own:

- ASIC-designers checked their modules before integration into the ASIC mainly focused on implementation specifics.
- System and board designers stressed the design from their system-function point of view, focussed on interrupts, ASIC interoperability, communication on the boards, and so on.
- Software-designers used the simulation just as virtual hardware to run their software on.

This combination of different verification approaches was a main contributor to the high quality of the hardware as it became available. The integration will completed on the virtual base a lot earlier. With the availability of physical HW prototypes the SW-integration starts already at a high quality level thus gaining a very positive impact on the time schedule.

2. TESTBENCH DESIGN AND CONFIGURATION MANAGEMENT

In order to minimize the effort for the generation and maintenance of the message buffer testbench we agreed on a concept which implements only one testbench which realizes the different hardware configurations by VHDL configurations. The testbench mainly consists of a toplevel entity, which instantiates the design under test, and a set of generator models for stimulating and tracers for observing activities on all relevant internal and external interfaces. The hierarchical structure of the testbench equals the hierarchical structure of the design.

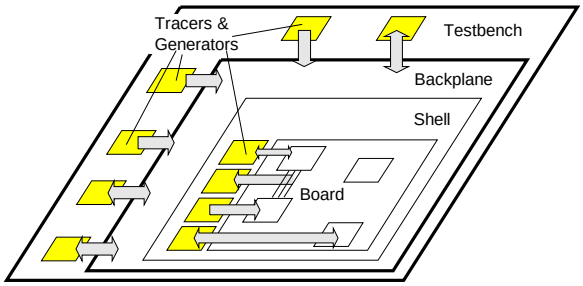

Figure 4: Hierachical architecture of the testbench

The implementation of the testbench for the simulation of the message buffer had to meet the following requirements:

- high simulation performance
- the external interfaces have to be modeled as realistic as possible
- easy, semi-automated, check of simulation results
- ability for regression testing
- the same testbench for HW simulation and HW/SW cosimulation

The following sections show how we tried to meet these requirements.

2.1 Writing performant testbenches

A performance estimation (please refer to [1] for further details) done at the beginning of the project showed that the hardware's simulator performance has to be at least 150 simulated clock cycles per second (for a single core design) to satisfy the performance requirements in order to achieve reasonable test coverage.

In order to reach this ambitious aim we decided to focus on two things

- use of cycle-based simulation and
- writing high performant simulation models.

Using cycle-based simulation means to have especially take care of the coding style of the design and the testbench elements.

Whereas ASICs anyway have to be coded at RT-level, they don't cause many problems. But there is the need for a cycle based version of the ASIC vendor's IO cell library and memory models.

More effort has to invested into the implementation of the testbench models. Usually testbench elements are modeled in high

level, behavioral HDL code, stressing the features of the language. Unfortunately this often doesn't fit into the cycle-based concept. So for example you can't use access types, global signals or `inout` parameters in procedures. Cylce-based testbench design means a significant change in how to implement the necessary testbench functionality. Synopsys' Cyclone requires the design to be synthesizable; in a testbench however, additional constructs are supported, including `textio` and a wider variety of `wait` statements.

Some techniques for writing high performant cycle-based generators and tracers are listed below (see also [3] and [6] for methods to enhance simulation speed):

- Do not use signals for inner-process communications, use variables instead
- Use separate processes for each clock domain to prevent evaluation of statements due unrelated clock signals
- For clock/reset generation use process suspension statements (`wait for`) instead of queuing events (`after` statement)
- Do not use assertion statements due the need of string conversions done by inefficient functions

All of this coding style hints improve the performance of an event-driven simulator, too. Following these guidelines testbench coding for cycle based simulation can be done in a very performant way. The reachable (and reached) performance gain ranges from 5 to 20x over an event-driven simulator.

2.2 Testbench configuration management

The large number of changing components (3 ASICs, 3 boards and 1 backplane) during the implementation phase requires a testbench configuration managment which primarly allows the generation of new configurations, e.g. due a new ASIC release, with less effort. As mentioned before this aim is reached by keeping only one testbench for both HW simulation and HW/SW coverification. To control the design hierarchy several VHDL configurations were used.

During the simulation phase of the project on average of 10 versions a board and 50 versions a ASIC were incorporated into the testbench. This leads to a total count of about 350 configurations for HW simulation.

To fulfill the requirement of one and the same testbench for HW simulation and HW/SW cosimulation the embedding of the RISC core was done in way allowing to select the kind of testbench by a VHDL configuration. Either the VHDL RTL model or the VHDL wrapper for the ISS was taken up by different configurations for one and the same testbench. This way another 100 configurations for HW/SW cosimulation were produced during cosimulation phase with no need to generate a new testbench.

3. DESIGN VERIFICATION

3.1 Verifying the hardware

During the design process appr. 300 testcases have been carried out at the ASIC-module level and appr. 80 at the systemlevel focussing on the verification of the functionality implemented in hardware.

	number of testcases	number of HW-bugs found
ASIC-testcases	300	79
System-testcases	80	248
Total	380	327

Table 1: Overview of hardware specific testcases

The number of bugs found with system testcases and HW/SW cosimulation was higher than one might expect due to the fact, that intermediate ASIC-deliveries took place as early as possible and thus shifting the load of verification to the other teams to gain an overall improvement in quality and schedule.

The testcase development methodology for the system level simulations is greatly described in [4]. System level testcases are mainly implemented as small pieces of C and assembler code running on the system's embedded cores. Additionally up to 100 file controlled testbench models were stimulating external system interfaces to get the desired test scenario.

In order to fulfill the requirements to automated testcase evaluation and regression test ability the testbench consists of tracer models connected to several system interfaces. Those tracers collect the information on their inputs and convert them into a human readable format. The results are stored in a set of output files which were used for simulation analysis and comparison with reference simulations in case of regression simulations.

Applying this methodology a number of 80 system testcases were simulated. Table 2 shows a overview about some testcase characteristics (simulation runtimes measured on a SUN UltraSPARC 2, 200 MHz).

testcase name	Gate count of configuration	LOC C	LOC Assembler	real-time	simulation time
I2C_ARB	720k	450	2,000	2,5 ms	8200 s
I2C_REG	370k	400	2,000	2 ms	1 h
IBUS_R	940k	-	5,500	3 ms	6 h
	700k	-	5,500	3 ms	4 h
	3760k	-	5,500	3 ms	16 h

Table 2: LOC, realtime and simulation runtime of some testcases

3.2 SW verification on the message buffer

When choosing a SW verification strategy for the message buffer, several major influences had to be taken into account:

- The tight project schedule naturally demanded that ASIC and SW be developed in parallel, and both HW and SW implementations had to be verified early in the project.
- The exceptionally high requirements for message throughput led to a closely coupled and sophisticated HW and SW design. Again, the implementations had to be verified as early as possible.
- Observability is greatly reduced for SW running on embedded cores, compared to using a distinct processor.
- The massively parallel structure of the message buffer demanded a debugging concept, where several cores could be debugged simultaneously.
- Main SW components (internal message transfer, routing and configuration) have been designed to support the parallel structure of the message buffer. Clearly, a verification environment supporting a fully equipped message buffer or at least a reasonably large configuration was required.

Considering the above constraints, HW/SW coverification was chosen as a verification strategy fulfilling our requirements. Of course the message buffer SW also includes SW components

without any direct connection to HW, one example being the administration of routing tables; for these, a PC based simulation environment (MTT, module test tool) was setup primarily for module tests and later on enhanced to support basic integration steps. However, complete integration was carried out using HW/SW coverification. Table 3 shows the verification strategies employed for different SW components and project phases, along with the final count of code lines (NLOCs):

Project phase	Component	HW/SW coverification	MTT	Size (NLOCs)
Module tests	Bootstrap, Startup	❍		2,000
	Hardware abstraction layer	❍		2,000
	Drivers	❍		10,000
	RTOS	❍	❍	4,000
	Protocol SW	❍	❍	12,000
Integration tests (1 ASIC)	Maintenance SW + Protocols		❍	30,000
	Complete SW	❍		48,000
Integration tests (systems)	Maintenance SW + Protocols		❍	30,000
	Complete SW	❍		48,000

Table 3: Software components and how they were verified

An essential goal has been to keep the SW running under the coverification environment identical with the final target version. However, performance limitations forced some minor modifications to shorten simulation times:

- Startup SW usually does a lot of DRAM setup. To avoid long startup delays. Instead, DRAM was preloaded directly by the HW simulator.
- Timer periods have been reduced.
- SW for background HW testing (which would not make sense in a simulated environment anyway) has been turned off.

Note that if ASIC emulation had been used as a coverification tool, severe changes to SW would have been necessary: being able to emulate just 1 ASIC per board instead of 8 (the true count, which we did in cosimulation) would have had a severe impact on configuration and maintenance SW. This would have meant a much higher effort and a higher risk for final target tests.

Using the setup outlined above, SW engineers were able to have all SW tested before actual silicon was available. In sum, 154 testcases ranging from 50 to 4,000 NLOCs each were tested in the coverification environment. The final test phase on the real target was reduced from an estimated 9 months without coverification to about 3 months, essentially reducing time-to-market by half a year.

4. DEFECT TRACKING

Different geographical locations of the development departments and the complexity of the design require and efficient way for defect tracking. Furthermore there is the need for some quality metrics of the developed design for management decisions.

In former projects an email describing the found problem was sent to a dedicated incident report (IR) manager, who was responsible for the further activities, e.g. IR numbering, e-mail distribution and so on (please see [2] for more details).

In the message buffer project the Distributed Defect Tracking System ClearDDTS (former PureDDTS) from Rational was used for IR management.

Figure 5 shows an IR management report used for decision when - or if - the developed design is ready for fabrication. During simulation phase many IRs were generated and opened for bug fixing. After fixing the bugs and re-simulation the number of open IRs became smaller and so the curve goes into saturation. This was one of the indicators to allow ASIC sign-off.

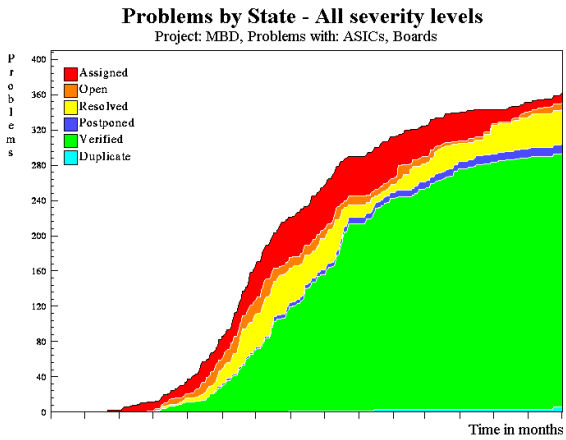

Figure 5: Found bugs during simulation over time

5. CONCLUSIONS

It has been shown that HW/SW-Coverification with cycle based simulation is a very powerful and flexible verification methodology on a virtual platform. Today's commercial available tools can provide reasonable performance and can also handle that immense complexity if they are used in the right manner and are applied in an efficient methodology.

Coverification enables a "formal" checking of design results of both HW and SW teams on a common virtual platform from the initial design step on through the whole design process.

The success of this project encourages us to improve our verification process in terms of efficiency and testcase throughput.

6. REFERENCES

[1] Albrecht, Notbauer, Rohringer: *"HW/SW Coverification Performance Estimation & Benchmark for a 24 Embedded RISC Core Design"*, 35th ACM/IEEE Design Automation Conference, pages 808-811, 1998

[2] Albrecht, *"Concurrent Design Methodology and Configuration Management of the Siemens EWSD-CCS7E Processor System Simulation"*, 32nd ACM/IEEE Design Automation Conference, pages 222-227, 1995

[3] Cohen: *"VHDL, Answers to Frequently Asked Questions"*, Kluwer Academic Publishers, 1997

[4] Jantsch, Notbauer, Albrecht, *"Testcase Development for large Telecom Systems"*, 2nd IEEE International High Level Design Validation and Test Workshop, 1997

[5] Pure Software, *"Distributed Defect Tracking System (PureDDTS), Administrator's Manual"*, 1995

[6] Synopsys Inc. *"Cyclone VHDL Coding Style Guide V1.1b"*, 1997