HW/SW CoVerification Performance Estimation & Benchmark for a 24 Embedded RISC Core Design

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

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

Stefan Rohringer
Siemens, Austria
Erdberger Laende 26
A-1030 Vienna, Austria
+43 1 1707 - 37630
stefan.rohringer@siemens.at

1 ABSTRACT

This paper describes the benchmarking of a HW/SW-coverification design strategy. The benchmark results were the base for making a principal verification decision for an already ongoing project at Siemens AG, Public Communication Network Group. The intention for this benchmark was to verify whether commercial available coverification tools can handle the design complexity of an embedded system containing 24 embedded RISC cores and provides the necessary performance in terms of simulation speed and throughput.

2 INTRODUCTION

2.1 Why Aiming HW/SW-CoVerification

The design to be coverified consists of

- 3 different types of ASICs each containing an embedded RISC core, SDRAM arbiter, protocol specific controller and interface to serve the board external interface, system internal controller and interface to serve board-to-board serial links. The gate count of a single ASIC is about 250kGates each.
- 3 different types of printed-circuit boards each containing a number of one type ASIC and off-the-shelf-components.
- Each type of those boards has one of the following external interfaces:
  200 MHz ATM serial interface, 8x HDLC serial interfaces, proprietary system interface
- 2 boards of each type in the minimum configuration (equals 24 embedded core ASICs) and a sum of 26 boards in the maximum configuration (equals 76 embedded core ASICs) will be delivered to our customer.
- Each board is connected by a serial link with each other of the boards.

At the time when the benchmark was carried out the design described above was in specification phase.

The reasons for doing the benchmark and to ensure high performance coverification capabilities for the design are several:

- No visibility and controllability of the RISC core's signals since they are all inside ASICs
- Limited access via a special type of In-Circuit-Emulator
- Early Integration of HW and SW on a virtual base
- Higher SW quality available for HW prototype tests
- „Breaking" the wall between SW- and HW-engineers from the system specification phase on
- Very tight project schedule

The goal of our coverification is to simulate the whole SW startup routine, operating system, HW-device driver and protocol stack with reasonable performance in the minimum configuration of the system to be delivered to our customer - this means to coverify a 24-core embedded RISC core design to be the largest coverification configuration. This design configuration equals to 6,000 kGates.

2.2 Performance Factors in HW/SW CoVerification

The traditional verification methodology [7] is based on simulation of the design mainly described in a HDL. Processors are either modeled in a HDL or they are integrated into simulation via a hardware model. Both of them have the disadvantages that the simulation performance is mainly given by the processor and that there is no possibility to connect a debugger to the processor.
Figure 1 shows a principal block diagram of a HW/SW-coverification tool [1].

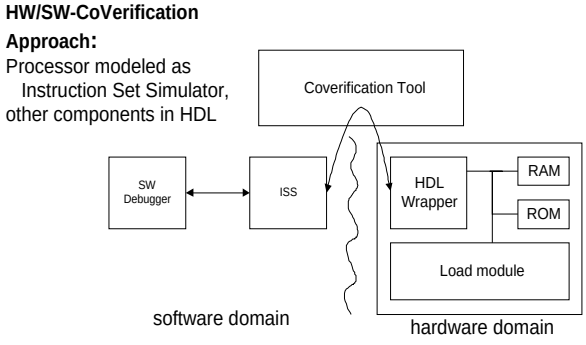

Figure 1 Blockdiagram of a HW/SW-coverification tool

The HDL model of the RISC core is replaced by an InstructionSetSimulator (ISS) of the core. The ISS is connected to the HW simulator by the coverification tool and instanciated

DAC 98, June 15-19, 1998, San Francisco, CA USA
ISBN 1-58113-049-x/98/06…$5.00

by a HDL wrapper. This HDL wrapper has to behave like a bus-functional model controlled by the ISS in order to provide the HW-design with cycles. The remaining HW is still modeled in the HDL simulator. Beside the execution unit the ISS is also able to model local memory areas. The key of simulation speedup is to reduce the number of events in the HDL-simulator. The percentage of memory cycles running across the HW-simulator to the HDL-type of memory model is called „core-I/O-activity". 100% core-I/O-activity means every single memory access is running across the HW-simulator, no portion of memory is modeled inside the ISS. The portion of memory area to be modeled either in the ISS or to be accessed via the HW-simulator to the HDL-type of memory must be defined in a mapping file accordingly to the design's memory layout.

The ISS has also an interface to a commercial SW-debugger. This is beside simulation throughput essential to ensure acceptance by SW engineers.

3 ESTIMATION OF REQUIRED HW/SW COVERIFICATION PERFORMANCE

A set of coverification testcases has been defined for different HW-configurations (see Table 1). For each testcase a core-I/O-activity and the number of assembler commands has been assumed.

Testcases	# embedded cores	assumed core-I/O activity	# Assembler instructions per testcase	# memory accesses	# memory cycles running accross the HW-simulator
Driver	1	10	50.000	5.000	30.000
Startup	1	27	14.000	3.780	22.680
Diagnostics	1	10	200.000	20.000	120.000
RTOS	1	1	17.000	170	1.020
ASIC integration, 1 core	1	10	65.000	6.500	39.000
system integration, 2 cores	2	10	65.000	13.000	78.000
system integration, 8 cores	8	10	65.000	52.000	312.000
system integration, 24 cores	24	10	65.000	156.000	936.000

Table 1: Testcase Scenarios

In order to stress the protocol stack SW a certain amount of data has to be transferred via the serial interfaces (ATM or HDLC) of the boards into the design. For 1-,2-, 8- and 24-core integration testcases it has been considered to play 50 HDLC frames, each with a message length of 70 byte, and to play 50 ATM cells to the design. It has also been considered that the HDLC interface will be accelerated for simulation purposes by 10x. For this acceleration the original VHDL HDLC interface block will be replaced in the VHDL database by a speeded-up VHDL HDLC interface block that allows to run the HDLC-protocol 10 times faster in simulation than in reality.

It is important to consider the impact of the serial interfaces with respect to the real design because protocol payload will be stored in a common memory area. Therefore this memory area must be mapped to the HDL-type of memory modeled in the HW-simulator rather than in the local ISS memory.

Table 2 gives the estimated performance numbers in 1000 Instructions-per-second and simulation run time for a scenario of simulated clock-cycles-per second and for a given ISS performance.

System Clock (MHz)		50							
ISS performance (1.000 instr/se		100							
speed up of seriell HDLC links		10	max speed up with min add. effort						
Scenarios	Clock cycles simulated/sec, 1-core configuration								
	10	20	30	40	50	100	150	200	
Results									
CoVerification Performance (IPS)									
Driver	17	33	50	67	83	166	249	332	
Start up	6	12	19	25	31	62	93	123	
Diagnostics	17	33	50	67	83	166	249	332	
RTOS	166	332	498	662	826	1.639	2.439	3.226	
ASIC integration	6	13	19	25	31	63	94	126	
System integr. 2 cores	3	6	9	13	16	31	47	63	
System integr. 8 cores	1	2	2	3	4	8	12	16	
System integr. 24 cores	0	1	1	1	1	3	4	5	
CoVerification Run Time (min)									
Driver	50	25	17	13	10	5	3	3	
Start up	38	19	13	9	8	4	3	2	
Diagnostics	200	100	67	50	40	20	13	10	
RTOS	2	1	1	0	0	0	0	0	
Asic integration	172	86	57	43	34	17	11	9	
System integr. 2 cores	345	172	115	86	69	34	23	17	
System integr. 8 cores	1.378	689	459	345	276	138	92	69	
System integr. 24 cores	4.135	2.037	1.378	1.034	827	413	276	207	

Table 2 Performance Estimation Sheet

Most of the testcases will be carried out of the ASIC-integration testcase group. A simulation run time of 10 to 20 minutes seems to be reasonable for this 1-core configuration. Using the performance estimation sheet it became obvious that the HW simulator must provide a performance in the range of 150 to 200 simulated clock cycles per second.
It turns out that the HW-simulator is the bottleneck in overall coverification performance. Therefore we have considered a cycle-based HDL simulator in our benchmark as well.

A variation of the ISS performance shows that at a given HW-simulator performance the ISS has not a substantial impact on the overall coverification performance. Only for testcases having a very low core-I/O-activity (around 1%) the ISS performance has some impact on the overall performance when the ISS is low performant (<5k Instructions per second).

4 THE BENCHMARK

4.1 The Benchmark Design

Due to the lack of having the real design available as described above, we had the need to build up a benchmark design.
The benchmark design is modeled in a way allowing easy scaling of the hardware's complexity. We decided to build up a basic module consisting of all the components necessary for a successful simulation run. Essentially these components are the RISC core, a clock generation unit, some memories and a load

module. The scaling of the hardware's complexity is done by simply multiple instanciations of the above mentioned basic module. That means, our benchmark design consists of 24 slots, each of them capable to embed a basic module. The number of basic modules in a specific configuration of the benchmark design is determined by a specific HDL configuration.

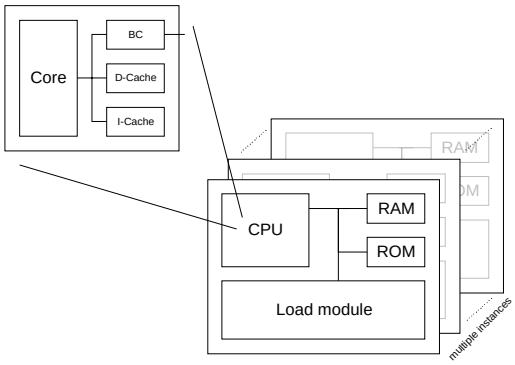

Figure 2 The Benchmark Design Configurations

When replacing the HDL model of the CPU by an ISS the load for the hardware simulator is only given by the memory models. In order to get a more realistic modeling of the target design we added a so-called load module to the main components of the basic module. It's aim is to represent the load of the hardware simulator generated by the specific interfaces (ATM, HDLC, ...) in the target design. The gate complexity of the load module (approx. 250 kGates) was chosen to be approximately equivalent to the estimated gate count numbers of the target design. The benchmark simulations were done with configurations consisting of 1, 2, 8 and 24 cores (basic modules).

4.2 Benchmark Testcases

The Instruction Set Simulator of the RISC core gives the possibility to map specific memory ranges either in the workstations memory (internal memory) or in the memory simulated by the hardware simulator (external memory). Accesses to the internal memory do not require any bus cycles and can therefore be done very fast. Typically code fragments are stored in this internal memory. The mapping of specific memory ranges to the external memory is necessary for data exchange between the software and the hardware.

The Benchmark testcases are designed to allow an easy modification of the access rate to the external memory. There are two loops which are performing read and write cycles to the both memories. By changing the maximum loop counts the density of the access cycles to the external memory can be adapted. The benchmark simulation runs were made for the following core-I/O-activities: 0%, 1%, 5%, 10% and 100 %.

The benchmark testcases have been defined in a way that the verification runs provide answers to the following questions:

- What is the simulation performance that can be achieved in HW/SW-CoVerification? Is it acceptable for SW engineers?
- What's the difference of the simulation performance when using either a HDL model for the processor or an ISS?
- How large is the performance gain when using the cycle based simulator Cyclone instead of the event driven QuickHDL?
- Is there a linear dependence between number of cores and simulated clock cycles per second?
- Is there a linear growth of the simulation runtime with the core-I/O-activity?
- Is there an impact from the number of CPUs in the workstation used for the simulation?

A total of 120 simulation runs have been done in order to present the performance data given in this paper.

4.3 Tools that Have Been Used

The following commercial available tools have been used:
EagleI CoVerification Tool by Viewlogic [1]
Cycle-Based Simulator CYCLONE by Synopsys [2] [3]
Event-Driven Simulator QuickHDL by MentorGraphics
ISS MiniSim by LSI Logic [4]
SW-Debugger CrossView by Tasking [6]

5 BENCHMARK RESULTS

Benchmark simulations were done in various configurations (Table 3) to answer the questions from section 4.2.

The carried out benchmark simulations gave clear performance figures and answers to the above mentioned questions:

- The required HW-simulator performance of 150 to 200 cycles-per-second and therefore the overall coverification performance has been achieved when using Cyclone as the HW-simulator (Figure 3 and Figure 4)
- The simulation performance when using the ISS of the processor instead of the VHDL model is twice as high (Figure 3 and Figure 4).
- There is a linear reciprocal dependency between the number of cores in the design and the number of simulated clock cycles per second, double number of cores half number of simulated clock cycles (Figure 4).
- The verification run-time increases linear to the core-I/O-activity.
- The HW-simulator performance in cycles-per-second is independent of the core-I/O activity. The core-I/O activity has its impact on the overall verification runtime. The higher the core-I/O activity the more memory cycles must run across the HW-simulator.
- There is a very little impact of the number of CPUs in the simulation workstation on the simulation drun-time. The main part of the simulation effort is consumed by the hardware simulation process which can not be splitted to run on different CPUs.(Figure 3 and Figure 4)
- The performance gain when using the Synopsys' Cyclone (cycle-bases simulator) instead of Mentor's QuickHDL (event-driven simulator) is approximately 10x when simulating in Cyclone 4 state debug mode, and approximately 20x when simulating in 4 state performance mode (Figure 3 and Figure 4).

Case 1	QuickHDL, pure VHDL
Case 2	QuickHDL, ISS, 1 CPU WS
Case 3	QuickHDL, ISS, 2 CPU WS
Case 4	Cyclone 4state, debug, ISS, 2 CPU WS
Case 5	Cyclone 4state, perf, ISS, 1 CPU WS

Table 3: Legend for the cases in Figure 3 and Figure 4

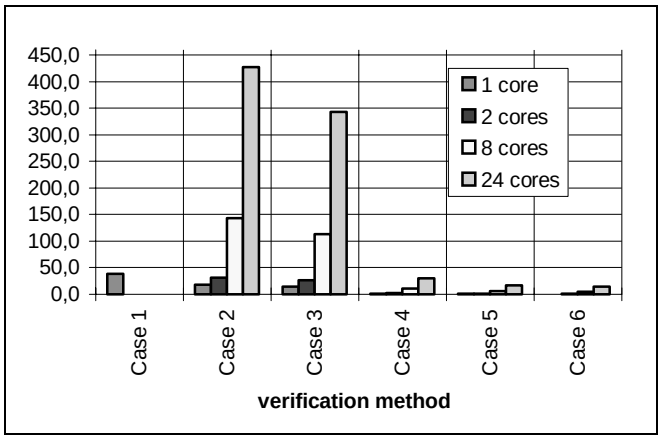

Figure 3 Verification RunTime in minutes depending on # of cores and verfication toolset at a given core I/O-activity of 10%

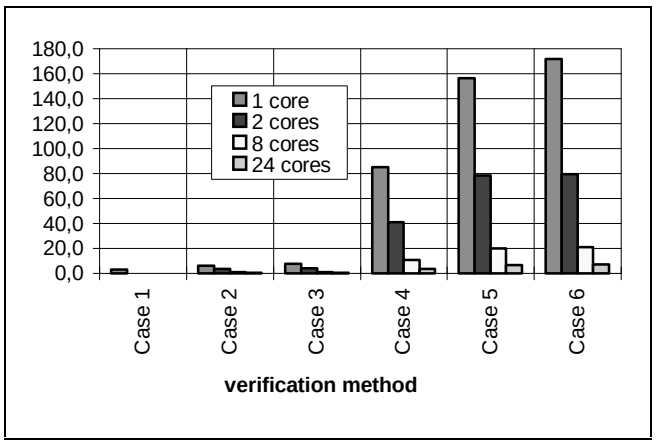

Figure 4 HW-simulator performance in simulated clock-cycles per second depending on # of cores and verification toolset

6 CONCLUSIONS

It turned out that HW/SW-coverification is a very powerful verification strategy. It has been decided to use HW/SW-coverfication in the ongoing project. It enables SW-engineers to start HW/SW integration already on a virtual base with no need to wait for physical HW samples. Also HW engineers make use of the cosimulation environment to debug their testcases written in Assembler or C-code to stimulate the design from the processor core site.

Despite the speed in terms of throughput performance that can be achieved with emulation, it has been decided not to use emulation for this project.

The reasons for this decision were:

- not capable to emulate a 24 core embedded RISC core design
- Emulation database and HW environment cannot be replicated very easily
- Emulation maintenance effort is high compared to a virtual based solution on workstations
- Man power costs for emulation-type of printed-circuit boards
- Emulation not flexible, e.g. in case of late ASIC-pinout changes (respin of boards required, ...)
- Emulation cannot start that early in the design process than HW/SW coverification can do, even for this type of system design.

The conclusion is that for this type of complex system designs the cost parameter for emulation is significantly higher than for HW/SW-CoVerification.

At the time this paper has been submitted are already first results of our real-world HW/SW-coverification application available. The coverification performance achieved using Eaglei/Cyclone is slightly better than the benmark results.

7 REFERENCES

[1] Viewlogic Systems Inc, „Hardware/Software Co-development with Eagle Tools“, http://www.viewlogic.com/products/eagletools.html

[2] Synopsys Inc, „Cyclone VHDL Coding Style Guide V1.1b“, 1997

[3] Synopsys Inc. „Cyclone VHDL Reference V1.1b“, 1997

[4] LSI Logic, „MiniSIM Architectural Simulator Family User's Guide“, November 1995

[5] Gerry Kane, Joe Heinrich, „MIPS RISC Architecture“, Prentice Hall 1992 Mips Technology

[6] Tasking Inc. „R3000 v3.2 CrossView Debugger User's Guide“, 1996

[7] Thomas Albrecht, „Concurrent Design Methodology and Configuration Management of the Siemens EWSD-CCS7E Processor System Simulation“, Proceedings of the 32nd ACM/IEEE Design Automation Conference, pages 222-227

[8] Matthias Bauer, Wolfgang Ecker, „Hardware/Software Co-Simulation in a VHDL-Based Test Bench Approach“, Proceedings of the 34th ACM/IEEE Design Automation Conference, Pages 774-779