Verification of Multiprocessor system using Hardware/Software Co-simulation

Hassan M Raza and Rajendra M Patrikar

***Abstract*--Co-simulation for verification has recently been introduced as an alternative to testbenches and in some cases to fast prototyping. In embedded systems especially when the systems are complex enough, the probability of asynchronous software-hardware communication increases. Modeling of such system requires realization of hardware on to a reconfigurable hardware emulator (FPGA). The software can be compiled and linked to this hardware emulator for co-simulation of the entire system. To implement this approach we use IMAGE system as the hardware emulator while Leon multiprocessor as the embedded system under test (SUT). Evaluated result of initial prototype hardware in terms of simulation acceleration achieved during the Co-simulation of multiprocessor system is shown. System-level simulation is used to demonstrate the real-time saving during verification of the proposed hardware. The RTL simulation of the SUT on host estimated a time requirement of 1sec real-time when run for 10 ms. while the same RTL simulation of the SUT after porting it on to the FPGA was 0.67 sec real-time when run for 10 ms. The performance of this architecture has been compared to a software implementation using the same test data set. We achieved an acceleration of 10% in other words time saving while performing Co-simulation of the Leon3 multiprocessor system.**

***Index Terms*--Co-simulation, hardware emulator, multi FPGA board, rapid prototyping.**

I. Introduction

CO-SIMULATION for verification has recently been introduced as an alternative to testbenches and in some cases to fast prototyping. Traditionally followed approach for Hardware-software co-simulation is to model the hardware processor and run the developed software on it. In embedded systems especially when the systems are complex enough, the probability of asynchronous software-hardware communication increases. Modeling of such system requires realization of hardware on to a reconfigurable hardware emulator (FPGA). The software can be compiled and linked to this hardware emulator for co-simulation of the entire system. This approach enables software to be developed and verified simultaneously with the system, speeding verification of the overall system.

To implement this approach we use IMAGE system as the hardware emulator while Leon multiprocessor as the embedded system under test (SUT). The work presented in this paper exploits the possibility of using IMAGE multi FPGA chip system for rapid prototyping of embedded system. In this context we map the Hardware component on FPGA platform while the software component on to the host. Eventually, hardware architecture of the Leon multiprocessor has been implemented in a Xilinx Virtex-II FPGA. While API (Application program interface) is used that leverages the hardware acceleration unit to run the requisite software on the hardware. Result of initial prototype hardware in terms of simulation acceleration achieved during the Co-simulation of complete multiprocessor system is shown to evaluate its performance.

In section 2, work related to use of co-simulation as a verification tool are briefly described. Exploration of IMAGE hardware accelerator system as a mean to performing the hardware/software co-design simulation has been discussed in section 3. Verification of multiprocessor system (Leon multiprocessor SPARC V8 architecture) achieved by co-simulation has been presented in section 4. Finally, results of System-level simulation are shown in section 5, to demonstrate the real-time saving during verification of the proposed hardware.

II. Related Work

In addition to simulation and other forms of traditional design verification, Hardware emulator plays an increasingly important role in the hardware simulation of today's digital systems. Similarly Multiple-processor in FPGA enables a more efficient use of processing power by partitioning tasks based on time and power. Using hardware accelerator for prototyping of a Multiprocessor seems to be a break through step and a new approach for system verification in digital design. Although during the recent decade outstanding contribution in the similar field has been witnessed, but verification of multiprocessor system using Hardware/Software co-simulation approach in reconfiguration environment was missing. Contribution by M. Porrmann et al group [1] describes a dynamically reconfigurable hardware accelerator for the simulation of self-organizing feature maps with scalable FPGA modules. The system achieves a

H. M. Raza is Research assistant and PG student with the Department of Electronics and Computer Science Engineering, Visvesvaraya National Institute of Technology, Nagpur, INDIA (e-mail: hassan_raza@ece.vnit.ac.in).

R. M. Patrikar is Professor with the Department of Electronics and Computer Science Engineering, Visvesvaraya National Institute of Technology, Nagpur, INDIA (e-mail: rajendra@computer.org).

maximum performance of more than 50 GCPS (Giga Connections per Second) a due inspiration for IMAGE. Yuichi Nakamura et al group [2] presented hardware/software co-verification method for System-On–a-Chip, based on the integration of a C/C++ simulator and an FPGA emulator. They describe the application of this environment to the verification of SoC, supporting concurrent hardware and embedded software development. James A. Rowson [4] put forward techniques available for co-simulation with an eye toward the strengths and weaknesses of each, according to him fastest raw performance comes from emulator at the cost of limited access into the emulated hardware for debugging and visibility into the internal states of the add-on standard products. We propose to introduce a union of these three distinct approaches to verify the multiprocessor system using IMAGE [5] which provides facility of internal signal monitoring and at the same time C interface for debugging the emulated hardware. Other works dealing indirectly with simulation accelerator and Hardware/Software co-simulation are presented in [3] by different group.

III. IMAGE System For Co-Simulation

In this section exploration of IMAGE hardware accelerator system as a mean to performing the hardware/software co-design simulation is described.

1) Introduction of IMAGE system
2) Verification can be performed on IMAGE
3) Rapid prototyping of Multiprocessor system

A. *IMAGE system - Introduction*

In this work we had used IMAGE system as our Hardware Software Co-design platform. IMAGE is a product manufactured by *Powai Lab* [5]. It can be used as behavioral or RTL simulation acceleration platform that can accelerate the simulation performance rate 1000x as compared to any software simulator available in the market today. Thus one can meet the verification goals of a design/verification team in a short period of time. IMAGE is a seamless solution that simulates our design in the same manner as traditional software simulator. However, it maps an entire design (or part of a design) onto hardware to achieve this dramatic improvement in performance. IMAGE can currently accelerate up to 2.4 million gate designs with 24 MB (megabytes) RAM. This architecture can scale up to 40 million gate designs for higher range.

Image comes equipped with the IMAGE Communication Software, IMAGE Mapping Tools and the IMAGE hardware box. The Communication Software communicates with the simulator and the IMAGE hardware box, the mapping tools move the design to the IMAGE hardware, and the IMAGE hardware comprises of a highly parallel architecture using very high speed and high density FPGAs on multiple boards, It also provides special feature by which we can monitor internal signal, force value to internal signal and call external memory available on its hardware.

B. *Verification on IMAGE*

IMAGE system uses Xilinx Virtex-II series XC2v4000F115 FPGAs as hardware while C API (Application Program Interface) to run the software on the hardware platform. IMAGE board is designed for use through PCI slot, keeping in mind the data rates and PCI operation frequency. IMAGE configuration used for Co –design comprised of two FPGAs with on board physical SRAM of 8MB distributed as 2MB of 3 Integrated Chips on board. This 2MB SRAM is available as 512K X 16 so if we have to access it SRAM of 2MB than we should have 19 Bit address ready with data, to feed to those many locations. One can access them only through set of component instantiations in your VHDL / Verilog program before using them. They are accessible in the form of 2MB each not as a whole 8MB at once. Meaning to use 8MB of memory on board user will have to declare routine four times with different reference name. IMAGE communication software keeps track of those references.

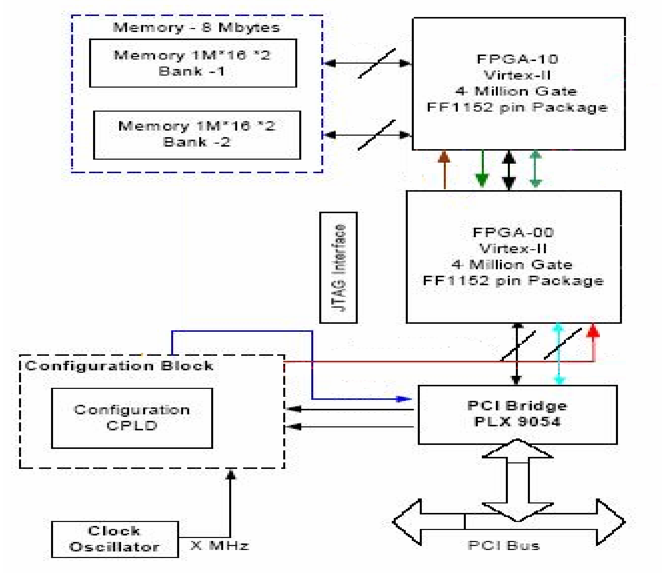


Fig. 1. Block diagram of IMAGE board.

The block diagram with 2 Xilinx FPGA and 8 MB external memory is shown in figure1.

IMAGE flow uses script based command for performing the simulation using MODELSIM tool. The simulation result on the hardware produce the same result as the simulation of model in a standard VHDL simulator with acceleration.

C. *Rapid prototyping for co-simulation*

As discussed before for getting simulation acceleration we map the SUT (System under Test) on the FPGAs and perform hardware simulation. Multiple FPGAs with high speed interconnect increases the usability of the board for prototyping. Apart from synthesis of the SUT, Mapping, Place & Route and Hex file generation is done using Xilinx XST and ISE, while the programming is done using Xilinx

IMPACT. IMAGE uses its pre compiled library to instantiate the external memory. So for a multiprocessor co-design environment this memory serves the need of external main memory. An external debug host can access the Debug Unit of the processor through several different interfaces, such as a serial UART (RS232), JTAG, PCI, Ethernet or IMAGE API (Application Programmable Interface) in our case. The interface of the processor communicates with the host through the IMAGE API which is 11 in number. So when we say prototyping of multiprocessor system on hardware, it comprises of the complete system being configured on to the hardware (here FPGA) along with the software running on it. To get the optimum utilization of the resource the design must be partitioned in such a way so to incorporate the design using the full resource utilized without any spill off. SRAM required for the system is the 8 MB memory (4 module of 2 MB each) present on the IMAGE board.

IV. Verification Of Leon Multi-Processor System

Verification of Leon multiprocessor [6] SPARC V8 architecture achieved by co-simulation is discussed in this section. Initial about Leon3 multiprocessor and its architecture, than how co-simulation of HW and SW is can be done.

A. Leon multiprocessor

Leon3MP is 32 bit SPARC V8 compliant freely distributed core. It is a simple extension of Leon3 processor architecture as shown in Figure 3. The debug support and interrupt handling is implemented separately for each LEON3 instantiation using a generate statement in a multi-processor system. There is only one debug support unit (DSU) in the design, supporting multiple LEON3 processors and only one interrupt controller, supporting multiple LEON3 processors. LEON3 processor implements a debug mode during which the pipeline is idle and the processor is controlled through a special debug interface. Debug Support Unit (DSU) is used to control the processor during debug mode. DSU acts as an AHB slave and can be accessed by any AHB master. An external debug host can therefore access the DSU through several different interfaces. Such an interface can be a serial UART (RS232), JTAG, PCI, USB or Ethernet. DSU supports multi-processor systems and can handle up to 16 processors

So here we targeted DSU for booting linux and loading design configuration, and for application development/ testing. We kept only core of Leon3MP design to save resources on IMAGE co-design platform. We now focus our attention to booting such multiprocessor design. There are three ways to boot kernel for multiprocessor, after all its configurations are read by processor through its various configuration registers.

- Through its external PROM
- Through Memory
- Through Debug Support Unit (by default)

We actually wanted to have full configurability in our hand, so that after porting Linux kernel we should also have provision to add drives/application to multiprocessor design. We used DSU for booting processors configuration. Snapgear [7] distribution was used which has port for kernel version 2.6.11 for Leon3 Linux port with some modification for our co-design platform.

B. Co-simulation of Leon multiprocessor

Here we place two Leon3 processor, Debug support unit, Asynchronous memory controller and High Speed Bus (HSB) controller on to the FPGA. Leon3 MP can be configured to boot from an internal prom or from the debug support unit. By asserting *DSUEN* and *DSUBRE* at reset time, the processor will directly enter debug mode without executing any instructions. The system can then be initialized from the communication link, and applications can be downloaded and debugged. We tested our compiled kernel on TSIM, which is actually the Leon2/3 emulator. Compiled using SPARC cross compiler provided with Snapgear distribution. After testing kernel successful boot on emulator configured with similar configuration as we had for our design. To talk to DSU at some baud rate pre-configured in the configuration file we are provided with Remote monitor RDBMON. This synchronizes with DSU on some standard boards only, and is designed for use at serial port not at PCI slot. So we had to write our own remote monitor unit to communicate with DSU referred as CMON, since source code for RDBMON is not distributed freely.

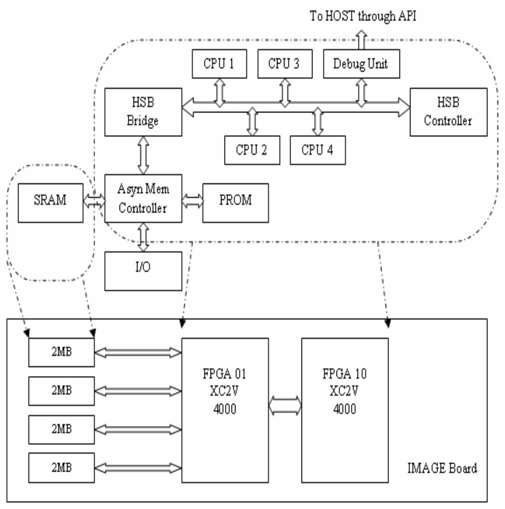


Fig. 2. Co-simulation of Leonmp on IMAGE.

Snapgear distributed port of Linux after successful compilation gives kernel image to be used for test with TSIM, DSU, RAM, etc. in ELF (Executable and linking) format. DSU has in built provision to extract headers and data segments. With simple file handling in C and keeping in mind Endean (Big/Small) of host platform and target platform, and IMAGE API's we communicated to design on IMAGE platform. After resetting Multiprocessor on board and applying appropriate signals to DSU we loaded kernel image on board. With an intermediate routine running in the form of CMON to direct and collect the outputs from DSU on IMAGE platform vice versa, and display it to host computer prompt. We verified whole setup of hardware software co-design execution by attempting various command in busy box tools.

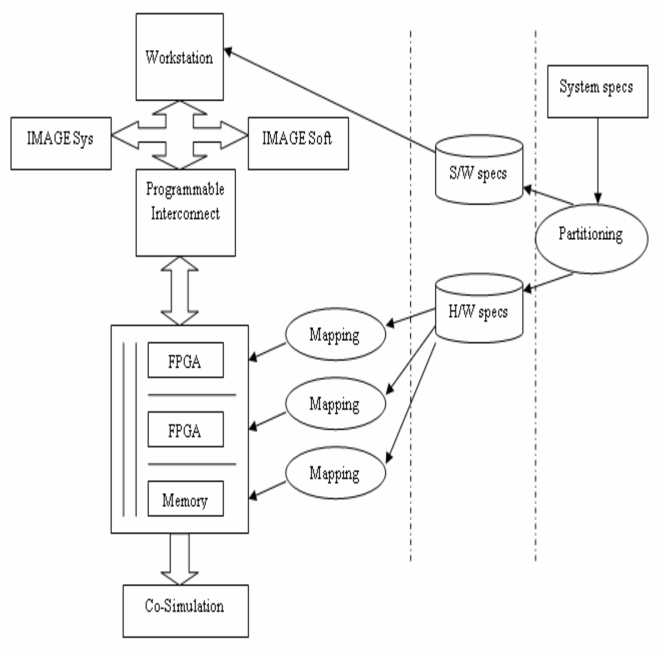


Fig. 3. HW/SW partitioning of Leonmp system.

To achieve the HW/SW co-simulation we partitioned the whole multiprocessor system into hardware component and software component according to the system specification and feasibility of the tools used.

C. *Work done*

Accordingly the hardware part was modeled using VHDL while software using C language. The VHDL code of the complete system was run through the complete IMAGE flow where by it is first analyzed by the analyzer than synthesized using Xilinx ISE than mapped on to the FPGA of the board. The final script command is

"Image.py -k vhdl -te leon3s -d top -s source -L $LPATH -sim_tool modelsim -sim_mode gui -synth_tool xilinx –r 2 –M"

After the command is run the HEX file of complete system is generated which is configured on the FPGA. Image takes care of timing and synchronizing information and is mapped along with the design.

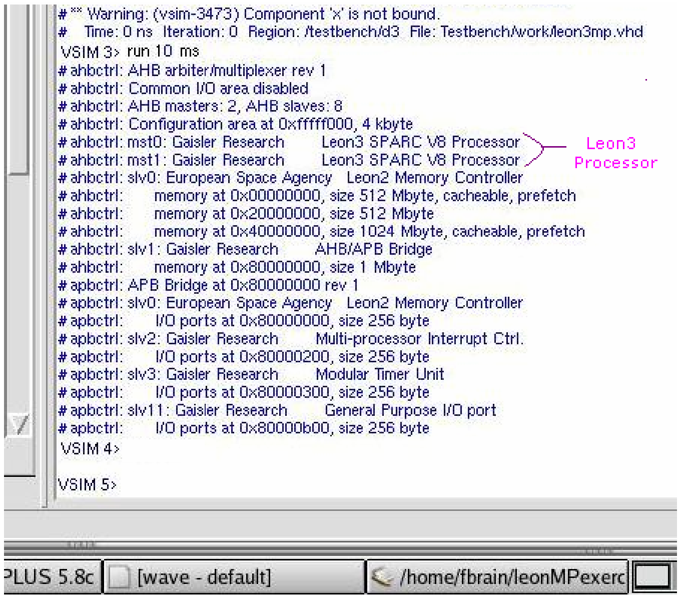
Parallel to this work the booting sequence along with linux kernel of the multiprocessor is compiled in the leccs-cross compiler available at gaisler research. The ELF file generated after this exercise is loaded on to the external memory using the C Interface of IMAGE. Then the C testbench through C Interface is used to ignite the processor. Simulation result of the complete system can than be seen on Modelsim tool.

V. RESULTS

Once the complete flow of IMAGE is run and the hex file of the software is loaded through the C API we are set to see the simulation of the complete system on Modelsim tool evoked by IMAGE.

The RTL simulation of the SUT on host estimated a time requirement of 1sec real-time when run for 10 ms. while the same RTL simulation of the SUT after porting it on to the FPGA was 0.67 sec real-time when run for 10 ms. The performance of this architecture has been compared to a software implementation using the same test data set. We achieved an acceleration of 10% in other words time saving while performing Co-simulation of the Leon3 multiprocessor system.

Figure 4 shows the screen shot of the multiprocessor first boot on the IMAGE system with two Leon3 processors as AHB (Advance High performance Bus) masters, 1 Mb cache memory, 1 Mb external memory and general purpose I/O port.


```
# ** Warning: (vsim-3473) Component 'x' is not bound.
#    Time: 0 ns  Iteration: 0  Region: /testbench/d3  File: Testbench/work/leon3mp.vhd
VSIM 3> run 10 ms
# ahbctrl: AHB arbiter/multiplexer rev 1
# ahbctrl: Common I/O area disabled
# ahbctrl: AHB masters: 2, AHB slaves: 8
# ahbctrl: Configuration area at 0xfffff000, 4 kbyte
# ahbctrl: mst0: Gaisler Research       Leon3 SPARC V8 Processor
# ahbctrl: mst1: Gaisler Research       Leon3 SPARC V8 Processor
# ahbctrl: slv0: European Space Agency   Leon2 Memory Controller
# ahbctrl:      memory at 0x00000000, size 512 Mbyte, cacheable, prefetch
# ahbctrl:      memory at 0x20000000, size 512 Mbyte
# ahbctrl:      memory at 0x40000000, size 1024 Mbyte, cacheable, prefetch
# ahbctrl: slv1: Gaisler Research       AHB/APB Bridge
# ahbctrl:      memory at 0x80000000, size 1 Mbyte
# apbctrl: APB Bridge at 0x80000000 rev 1
# apbctrl: slv0: European Space Agency   Leon2 Memory Controller
# apbctrl:      I/O ports at 0x80000000, size 256 byte
# apbctrl: slv2: Gaisler Research       Multi-processor Interrupt Ctrl.
# apbctrl:      I/O ports at 0x80000200, size 256 byte
# apbctrl: slv3: Gaisler Research       Modular Timer Unit
# apbctrl:      I/O ports at 0x80000300, size 256 byte
# apbctrl: slv11: Gaisler Research      General Purpose I/O port
# apbctrl:      I/O ports at 0x80000b00, size 256 byte
VSIM 4>

VSIM 5>
```

Fig. 4. Modelsim screen shot on IMAGE showing simulation of Leon multiprocessor system.

VI. CONCLUSION

Multiprocessor in FPGAs provides good system integration, partitioning the system between hardware and software adds flexibility to the system. Designers can implement the system's software components in the embedded processors and implement the hardware components in the FPGAs general logic resources with a Host to verify the system Integrity. Using hardware accelerator for rapid prototyping of multiprocessor system seems to be futuristic step for Hardware/Software co-design and an inevitable platform for cycle accurate and fast co-simulation. While synthesizing multiprocessor on hardware accelerator it is seen that modeling its memory component requires large simulation time. If we partion the design in such a way so as to keep the memory component in the software side while rest of the hardware on board. We get faster simulation as compared to placing the complete system on the board. While modeling the clock intensive module in hardware accelerator requires more computational time for synthesis. An approximate time for synthesis of the complete system is as follows

Memory module	-	70%
Clock module	-	10%
Remaining design	-	10% of the overall synthesis time.

VII. References

[1] M. Porrmann, M. Franzmeier, H. Kalte, U. Witkowski and U. Rückert. A Reconfigurable SOM Hardware Accelerator. ESANN'2002 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342.

[2] Yuichi Nakamura, Kouhei Hosokawa, Ichiro Kuroda, Ko Yoshikawa and Takeshi Yoshimura. A Fast Hardware/Software Co-Verification Method for System-On-a-Chip by Using a C/C++ Simulator and FPGA Emulator with Shared Register Communication. ESANN'2002 proceedings - DAC 2004, June 7–11, 2004, San Diego, California, USA. Copyright 2004 ACM 1-58113-828-8/04/0006.

[3] Young-Il Kim, Wooseung Yang, Young-Su Kwon and Chong-Min Kyung. Communication-Efficient Hardware Acceleration for Fast Functional Simulation. Proceedings of the 41st annual conference on Design automation San Diego, CA, USA, 2004, SESSION: Advances in accelerated simulation pp. 293 – 298.

[4] James A. Rowson "*Hardware/Software Co-Simulation*" 31ST ACM/IEEE Design Automation Conference 1994 ACM 0-89791-653-0/94/0006 3.50.

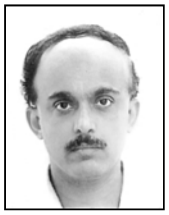
[5] Powai Labs, "The high quality, affordable and robust price-performance Simulation Accelerator and Emulators," Available at http://www.powailabs.com.

[6] Gaisler research "Product LEON processor, a 32-bit synthesizable processor core based on the SPARC V8 architecture," Available at http://www.gaisler.com/cms/index.php

[7] SnapGear Embedded Linux, "SnapGear Linux is a full source package, containing kernel, libraries and application code for rapid development of embedded Linux systems," Available at http://www.gaisler.com/cms/index.php.

VIII. Biographies

Hassan M. Raza is Working as Research assistant at Electronics and Computer Science Department and also doing his Masters from Visvesvaraya National Institute of Technology, Nagpur. He completed his bachelor of Engineering from YCCE, Nagpur. His areas of research are Reconfigurable multiprocessor design and verification.



Rajendra M Patrikar is working as Professor at Electronics and communication Department, Visvesvaraya National Institute of Technology, Nagpur, INDIA. His areas of interest are VLSI design and Technology, VLSI design automation