Fast Development of Source-level Debugging System Using Hardware Emulation

Sang-Joon Nam, Jun-Hee Lee, Byoung-Woon Kim,
Yeon-Ho Im, Young-Su Kwon and Chong-Min Kyung

VLSI Systems Lab., Dept. of Electrical Engineering
Korea Advanced Institute of Science and Technology,
Teajon, Korea
Tel: +82-42-866-0700
Fax: +82-42-866-0702
e-mail: {sjnam, munjigi, bwkim, mini, yskwon}@duo.kaist.ac.kr,
kyung@eekaist.kaist.ac.kr

Kyong-Gu Kang

MRI Divison,
Medison Corporation,
Teajon, Korea
Tel: +82-42-488-9201
Fax: +82-42-488-9202
e-mail kedison@mri.medison.co.kr

Abstract— We describe the co-development of a processor and its source-level debugging system using an emulation-based validation technology including hardware emulation, not simulation. Since a source-level debugging system is essential to develop an application system and it takes a long time to validate the functionality of the source-level debugging system, we have adopted hardware emulation for a fast validation and system development. Using this methodology, we were able to validate the source-level debugging system successfully before the chip fabrication.

I. Introduction

As the complexity of the application programs is growing, most of them are implemented with high-level programming languages for fast development, easy debugging and easy upgrading. Therefore, in developing an application program, it is essential to support the source-level debugging system, which allows the program developer to follow any execution path within the complex programs.

One of the advantages of the source-level debugging system is that this system enables a programmer to run his/her application programs at full speed within the target hardware. Unlike simulation environment, this system is capable of operating in the actual target-application hardware environment considering interrupts, PCI (Peripheral Component Interconnect) and host interfaces. Therefore, complete testing and verification of real-time program at the source level requires running application on the target hardware[1]. Running real-time software while controlling and observing the state of the running program requires emulator.

There are two implementations for an emulation-based source-level debugging system. One is JTAG emulator which is based on the boundary scan technique as defined by IEEE 1149.2[2], and the other is In-Circuit Emulator (ICE) which uses a special bond-out processor to observe the states of the target processor precisely. While ICE requires an expensive bond-out chip to observe and control the processor, JTAG emulator supports these features using an additional small control logic and five testing pins which are defined by IEEE 1149.1.

However, so far, the source-level debugging system has been implemented and verified after the fabrication of processor, which is a great disadvantage in terms of the fast time-to-market request[3]. Hardware emulation allows truly concurrent engineering in the system development as depicted in Fig.1. At some point in the traditional development flow, software and hardware efforts need to wait for the delivery of the first silicon for further debugging and continuing the development. Hardware emulation is equivalent to receiving "the first silicon" as soon as synthesizable RT-level design is complete. Using hardware emulation, software and hardware can be integrated and verified concurrently at the earlier stage.

In this paper, we propose a methodology for the co-development of a processor and its source-level debugging system using hardware emulator. A co-development requires a system co-validation. Two approaches have been used for co-validation, i.e., simulation-based validation and emulation-based validation [4]. The choice depends on the complexity of the system being designed. If the coverage of the functionality of the design can reach the satisfactory degree in a short time using simulation, the simulation-based approach would be preferred due to its high observability. However, to validate the source-level debugging system, it is necessary to run a lot of testing programs and application programs on a processor model, which can be modeled by either software or hardware. Many processor companies have used emulation-based approach to validate the functionality of processors because it takes too long time to run programs on the software model [3][5][6]. Validation of the source-level debugging system within a processor is as complex as that of the processor because the source-level debugging sys-

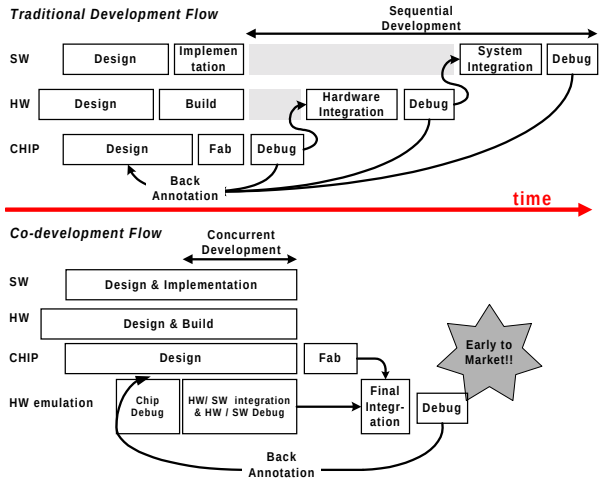

Fig. 1. Co-development using hardware emulation.

tem uses both the processor's normal mode and debug mode, which is invoked by interrupt requests from a on-chip debug unit. Thus, we adopted an emulation-based validation approach for the co-development of a processor called FLOVA (FLOating-point VLIW Architecture) and a corresponding source-level debugging system targeted for image processing and 3D graphics [7].

The organization of this paper is as follows. Section 2 describe FLOVA source-level debugging system. In Section 3, the hardware emulation environment for co-development is explained. Section 4 shows the hardware and software results of the proposed co-development method. Finally, conclusions are presented in Section 5.

II. Source-level Debugging System of FLOVA

Fig. 2 shows FLOVA source-level debugging system consisting of source-level debugging library, source-level debugger, JTAG control driver, JTAG emulator, on-chip debug unit, JTAG unit, and hardware emulator. After the target program is downloaded, the debugger program can control the target program and read or write on-chip registers and internal/external memories. These debugging control operations are done by on-chip debug unit in FLOVA, which is controlled by the debugger program through JTAG control driver.

A. Source-level Debugger

The C source-level debugger is an advanced programmer's interface that helps the program writer to develop, test, and refine C programs. The debugger program is run on the host computer and is need to download a target program into the target system through the JTAG port, followed by the debugging of the target program. Especially, the source-level debugger can debug interrupt service routine. These debugging operations are performed using source-level debugging library.

B. Source-level Debugging Library

Source-level debugging library consists of two parts. One part provides the relation between a line of source program and "Program Counter", the location of a variable and the information about source files of a target program. Another part consists of several C functions, which control JTAG emulator board, access the on-chip debug unit and control source-level debug operations.

C. JTAG control driver and JTAG emulator

The JTAG control driver controls the JTAG controller in JTAG emulator, accesses debug registers in the on-chip debug unit and performs specific debug operations such as stopping, restarting, stepping, setting breakpoints. This driver consists of three hierarchical layers according to its functions.

- Bottom layer: JTAG control functions to read or write the contents of registers in the JTAG controller.
- Middle layer: On-chip debug register access functions to read or write the contents of on-chip debug registers.
- Top layer: Source-level debugging functions to read or write the contents of on-chip registers or internal/external data memory, and do the specific debug operations such as step, run, stop, set breakpoints, clear breakpoints, etc.

The JTAG emulator has the JTAG controller that translates parallel data into the JTAG serial data or vice versa, and ISA interface controller [8]. This board is connected to the source-level debugger via JTAG header to run on the board and debug the programs in the target system.

D. On-chip Debug Unit

To support debugging in hardware, FLOVA has an on-chip debug unit whose functions are to stop/resume/run single-step/set breakpoints/access the registers of FLOVA. The debug unit is controlled via JTAG ports, which is the boundary scan technique and standardized as IEEE 1149.1. Using the JTAG ports, the source-level debugger can set hardware breakpoints on two program addresses and four data addresses, where several conditions ($=$, $\neq$, $>$, and $<$) can be checked for each address using breakpoint logic as shown in Fig. 3. If a condition is met on specified address, an interrupt occurs to stop the normal execution of a program and to enter the debugging mode where FLOVA waits for an instruction from the

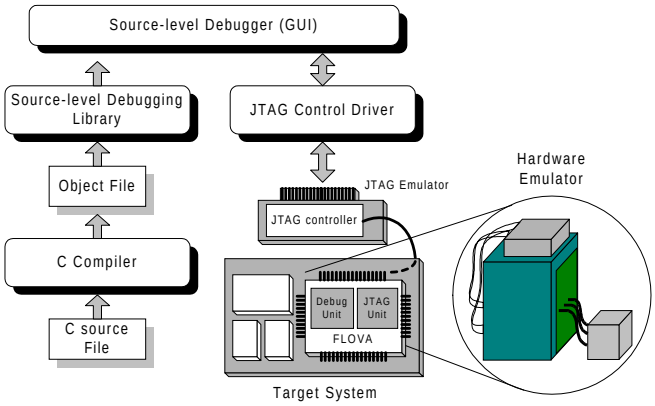

Fig. 2. FLOVA source-level debugging system consisting of a source-level debugging library, a source-level debugger, a JTAG control driver, a JTAG emulator, an on-chip debug unit, a JTAG unit, and a hardware emulator.

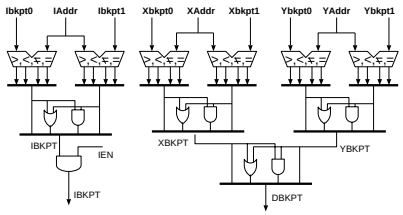

Fig. 3. Breakpoint logic for a program address and two data addresses in FLOVA

debug unit while executing NOP (No-operation) instructions. Then, the instruction to be executed on FLOVA in debug mode, can be transferred via the JTAG ports to a debug instruction register (DIR) whose content would be shifted through FLOVA instruction chain for a single instruction execution. Using this controllability, FLOVA supports debugging functions. Fig. 4 shows instruction chain for debug logic. Detailed explanation for breakpoint and debug logic is omitted in this paper.

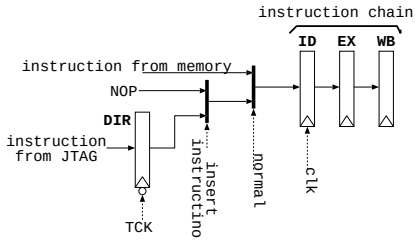

Fig. 4. Debug unit for a single instruction execution in FLOVA

III. Hardware Emulation

The greatest advantage of hardware emulation over simulation is in its higher verification speed. Hardware emulation is only 100 times slower than a real chip. Furthermore, the integration of the environment is much easier and high expenditure for the generation of test benches can be avoided. Therefore, nowadays hardware emulation becomes the standard technique in processor verification. Since the speed advantage of hardware emulation is more apparent compared with simulation, hardware emulation can be expected to become an essential part of co-development methodology.

One of the main disadvantages of hardware emulation over simulation is that it is hard or impossible to detect the timing errors. Therefore, hardware emulation mostly serves the testing of functional correctness. There are further disadvantages: slow compilation once the circuit changes, different design flows for implementation and emulation, and the high expenditure, which may be up to one dollar per emulated gate.

We use M3000 class of Quickturn [9] as hardware model of FLOVA. M3000 allows a maximum capacity of 3 million gates. The support software includes partitioning and mapping on the different FPGAs. There is also an integrated instrumentation available to capture and process real-time data for design-debugging. Quickturn M3000 model especially supports a very flexible memory emulation. Small scattered memories can be compiled into the FPGAs' internal RAM which can emulate single, dual, and triple-port memories. In addition, special target interface modules (TIM) interfaces the emulator with the target system. Furthermore, complex ASIC functions including micro-controllers, processors, peripheral controllers and etc. are integrated into the emulated environment by plugging real chips into the TIM. Like all other emulation systems, it allows debugging in the test vector mode and in the dynamic mode. In the test vector mode, test vectors are applied and results are stored. A special functional test mode supports regression testing where 128K of vectors can be applied at up to 4MHz in an IC tester-like functional validation environment. In

the dynamic mode, a logic analyzer can be connected to the emulation environment. This logic analyzer includes a state machine based trigger and acquire capability with up to 8 states, 8 event trigger support, 128K channel depth and 1152 channels. The trigger-and-acquire conditions are very similar to advanced commercial logic analyzers.

IV. Results

For co-development of the source-level debugging system, we first developed the JTAG emulator and the simple target system.

We have validated the functionality of the debugging system easily with the help of the hardware emulation speed in the environment as shown in Fig. 5. If FLOVA model were simulated using a general RTL simulator, it would have taken a long time to validate the source-level debugger system because less than 10 FLOVA instructions could be executed per second.

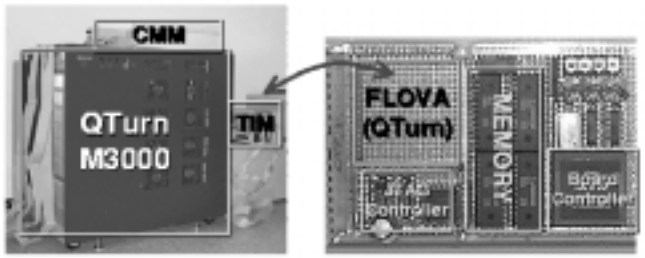

Fig. 5. Co-development environment of FLOVA source-level debugging system.

Fig. 6 shows an window example of source-level debugger GUI (Graphic User Interface) consisting of several information windows such as a source program window, a breakpoint window, a register window, and a memory window.

Hardware emulation facilitates concurrent development and validation by allowing system hardware and software integration even before design tape-out.

V. Conclusions

In this paper, we described the co-development methodology of a processor and its source-level debugging system using an emulation-based validation technology including hardware emulation. Since a source-level debugging system is essential to develop an application system and it takes a long time to validate the functionality of the source-level debugging system, we adopted hardware emulation for a fast validation and development of FLOVA source-level debugging system. Using this methodology,

Fig. 6. An example of source-level debugger consisting of a source program window, a breakpoint window, a register window, and a memory window.

we have validated the source-level debugging system successfully before a chip fabrication. To meet the challenges of shortened product life-cycles and reduced time-to-market, the co-development using hardware emulation is indispensable.

Acknowledgements

This work was performed as a part of ASIC Development Project supported by the Ministry of Trade, Industry & Energy, the Ministry of Information and Communication, and the Ministry of Science and Technology.

References

[1] R.A.Gott, "Debugging embedded software," *Computer Design*, pp.71-78, Feb. 1998.

[2] IEEE, "*IEEE Standard Test Port and Boundary-Scan Architecture*," IEEE, 1990.

[3] J.Kumar, N.Strader, J.Freeman, and M.Miller, "Emulation Verification of the Motorola 68060," *Internal Conference on Computer Design*, pp.150-158, Oct. 1995.

[4] G.J.Bunza, "The Impact of Hardware/Software Co-development on Design Process Methodology: Big Changes and Bigger Success," *Proc. Design, Automation and Test in Europe*, pp.37-41, Feb. 1998.

[5] G.Ganapathy, R.Narayan, G.Jorden, and D.Fernandez, "Hardware Emulation for Functional Verification of K5," *Proc. Design Automation Conference*, pp.315-318, June 1996.

[6] J.Gateley et.al, "UltraSPARC-I Emulation," *Proc. Design Automation Conference*, pp.13-18, June 1995.

[7] S.J.Nam et.al, "VLIW Geometry Processor for 3D Graphics Acceleration," *International Symposium on Low-Power and High-Speed Chips (COOL Chips)*, pp.107-120, Apr. 1999.

[8] Texas Instruments, "*SN74ACT8990, Test-Bus Controllers IEE STD 1149.1 TAP masters with 16-bit Generic Host Interfaces*," http://www.ti.com.

[9] Quickturn Design Systems, Inc., "*System Realizer User's Guide Version 5.0*,"