Reusable Embedded Debugger for 32bit RISC Processor Using the JTAG Boundary Scan Architecture

Dae-Young Jung
Dept. of Electrical Engineering
Yonsei University, 134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea
82-2-2123-4731
jdy21@spark.yonsei.ac.kr

Sung-Ho Kwak
Dept. of Electrical Engineering
Yonsei University, 134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea
82-2-2123-4731
iris@spark.yonsei.ac.kr

Moon-Key Lee
Dept. of Electrical Engineering
Yonsei University, 134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea
82-2-2123-2867
mklee@mail.yonsei.ac.kr

Abstract

The traditional debug tools for chip tests and software developments need a huge investment and a plenty of time. These problems can be overcome by Embedded Debugger based the JTAG boundary Scan Architecture. Thus, the IEEE 1149.1 standard is adopted by ASIC designers for the testability problems. We designed the RED(Reusable Embedded Debugger) using the JTAG boundary Scan Architecture. The proposed debugger is applicable for not a chip test but also a software debugging. Our debugger has an additional hardware module (EICEM : Embedded ICE Module) for more critical real-time debugging.

Keywords

Reusable embedded debugger; 32bit RISC processor; JTAG; boundary scan; TAP controller

1. INTRODUCTION

With the increasing complexity of chip design, the software development and system debugging stage of product now contribute to a significant proportion of the time to market. And in order to remain competitive, the product development cycle should be kept to minimum.

The traditional methods of debugging an embedded system design is an ICE (In Circuit Emulator) and a logic analyzer. However, these traditional debug tools are very expensive and take a plenty of time for verification because VLSI circuits are very integrated and the system clock frequency is increased. In many applications, the processor, memory and peripherals are all integrated on the same device. It makes ICE access to processor difficult or impossible. Unfortunately, the proliferation of different microprocessor versions within the same family implies the acquisition of different probes. Thus, it incurs additional cost [1].

Therefore, we overcome such problems by embedding an in-circuit-debugger based on the boundary scan architecture in 32bit RISC processor. We also add special hardware (Embedded ICE Module : EICEM) to the debugger module for more time-critical debugging. This debugger operates as stand alone system by supporting PC-interface ; it does not need an expensive equipment such as IMS, logic analyzer etc. Such feature greatly decreases cost and time for a chip testing and a software debugging.

Section 2 of this paper will briefly describe the debug system and the architecture of the Reusable Embedded Debugger(RED). Section 3 will describe the TAP controller for RED module. Section 4 will describe the debug solution of RED architecture. Section 5 will describe the implementation on the SPK-611A. Section 6 will conclude this paper with final remarks.

2. THE ARCHITECTURE OF RED

Generally the debug system using the IEEE 1149.1 standard consists of three parts : debug host, protocol converter and target system. The debug host is a platform such as a personal computer, where the debugger software runs. As in the figure 1 the debug host connects with a target system through protocol converter.

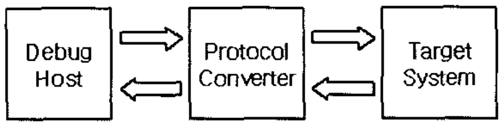

Figure 1. The debug system

The proposed RED module is integrated with a target processor and connected directly with the protocol converter via a JTAG(Joint Test Action Group).

Figure 2 displays the architecture of RED. RED is composed of major three blocks : Boundary Scan Architecture, EICEM and boundary scan register.

The Boundary Scan Architecture controls the action of the scan chain via a JTAG serial interface and the EICEM. The EICEM is an additional module for more critical real-time debugging. This

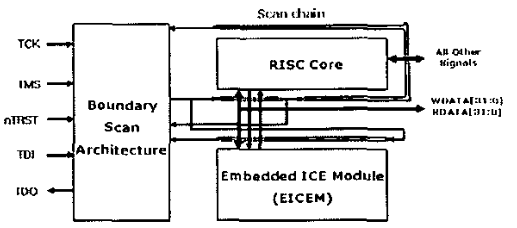

Figure 2. The RED diagram

module comprises a set of registers and comparators, and generates debug exceptions. As in the figure 2 the RED has three type of boundary scan chain. These boundary scan chains surround a core and the EICEM.

The RED forces the core to be stopped on either a given instruction fetch(breakpoint), data access(watchpoint) or a debug-request asynchronously. When one of them occurs, it changes into the debug state and the core is isolated from the memory system. At this state, the internal state and the external state of the system can be examined. After examination is completed, the core and the system state can be restored and the program execution can be resumed.

3. BOUNDARY SCAN ARCHITECTURE AND SCAN CHAINS

Figure 3 shows the Boundary Scan Architecture for the RED architecture. And table 1 presents instructions for Our Boundary Scan Architecture.

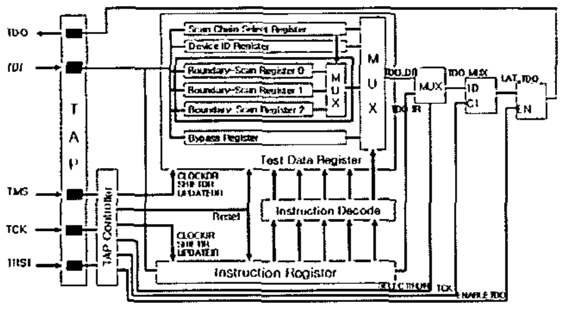

Figure 3. The Boundary Scan Architecture

Table 1. Instructions for Boundary Scan Architecture

Instructions	Binary Code
EXTEST	0000
SCAN_N	0010
INTEST	1100
IDCODE	1110
BYPASS	1111

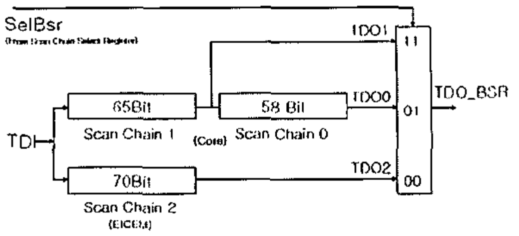

Figure 4. The Boundary Scan Chains

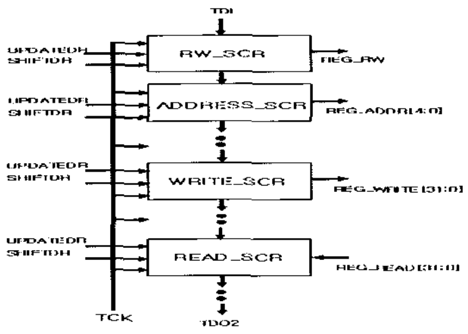

Figure 5. Scan Chain2 diagram

General Boundary Scan Architecture has one scan chain. In case of EXTEST and INTEST, the scan chin is automatically selected[2,3]. However, RED has three scan chains. Therefore, RED has the scan chain select register to select one of the scan chains. After scan chain is selected by SCAN_N instruction, instructions of EXTEST and INTEST are operated.

There are three JTAG style scan chains. Figure 4 displays three scan chains. These are used for testing, debugging and EICEM programming. The scan chains are controlled by the JTAG style Boundary Scan Architecture. But the scan cells are not fully compliant for the JTAG[3].

Scan chain 0

Scan chain 0 allows access to the entire periphery of the RISC core, including the data bus. All the inputs and outputs can be controlled via scan chain0. The scan chain functions allow inter-device testing (EXTEST) and serial testing of the core (INTEST).

Scan chain 1

Scan chain 1 is a subset of the signals that are accessible through scan chain 0. Access to the core's write data bus, WDATA[31:0], the core's read data bus, RDATA[31:0] and the BREAKPT signal is available serially.

Scan chain 2

This scan chain simply allows access to the EICEM registers. This scan chain is used to control the registers in the EICEM. Figure 5 shows scan chain2 diagram. There are read/write register, address data registers, data registers for writing and data registers for reading. Each of registers of Scan chain 2 is different.

4. DEBUG SOLUTION OF RED ARCHITECTURE

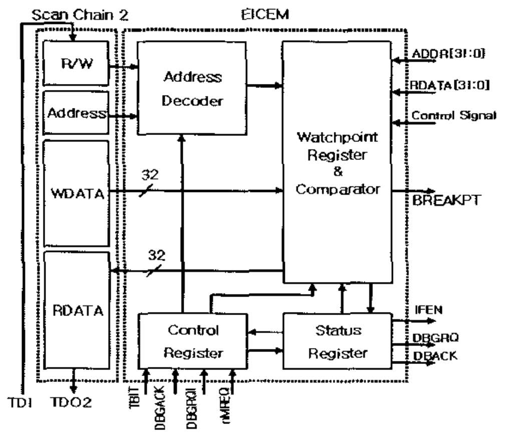

Figure 6. The EICEM block diagram

The RED allows programs to execute at full system speed and the core to be halted by the EICEM when a breakpoint or a watchpoint is seen by the EICEM. The core then enters a debug state which allows the internal system state to be examined and modified if required. Figure 5 shows block diagram of EICEM.

As in figure 6, EICEM is largely divided into four blocks: Address Decoder, Control Register, Status Register, and Watchpoint Register & Comparator.

Through scan chain 2, the value of address and data are delivered to EICEM. When the value of address is put into the address decoder, the watch point register is selected. After then the value of register is read or is written, according to the value of R/W. Through this process, the breakpoint and the watchpoint are programmed to EICEM register. Comparator always checks the address, the data bus, and the control signals. When this programmed value of breakpoint or watchpoint is matched with the one of EICEM register, BREAKPT signal is sent to the core. At this time, the core and acknowledgement signal are sent to EICEM. However the cycle timing differs between breakpoint and watchpoint. In the case of breakpoint, a signal is sent at the moment when instruction arrives at execution stage, while the watchpoint is sent when the instruction is completely executed.

The core may also be forced into debug state on debug request. This can be done either through EICEM programming or by the assertion of the external debug request signal. This signal is an asynchronous input signal and is thus synchronized before it takes effect. Control register controls other parts of EICEM through external debug request signal and core signals.

Once the core enters debug state, it stops fetching instructions from the data bus and isolates itself from the memory system. The EICEM may now scan instructions into the pipeline via scan chain1 and clock the core. Registers and memory contents can also be examined from debug state.

5. IMPLEMENTATION ON THE SPK611A

We designed the RED architecture by the Verilog HDL, simulated behaviorally it by using the Virsim of Synopsys Co., and synthesized it by using Design Analyzer (0.25μm design rule). Figure 7 is the result of simulation and table 1 is the result of synthesis. We got the result that total gate counts are 3174 gates and maximum operating frequency is 385M Hz.

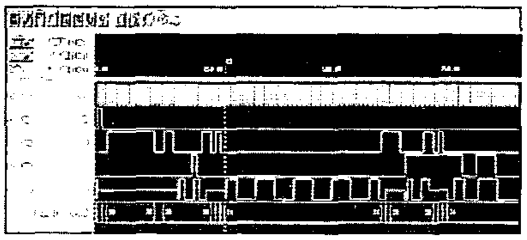

Figure 7. Waveform of simulation

Table 1. Gate Counts of modules

Module	Gate counts
EICEM	412
Boundary Scan Architecture	446
Boundary Scan Chain 0,1 (core)	1662
Boundary Scan Chain 2 (EICEM)	654
Total	3174
Max frequency	385M Hz

The RED architecture can execute a test and debugging of processor in any kind of processor which is embedded in it. However, we embedded the RED architecture in SPK-611A, the 32bit RISC processor developed by Yonsei university[8].

Table 2 shows an area overhead of SPK-611K with RED.

Table 2. Overhead of SPK-611K with RED

Design	SPK-611K	SPK-611K + RED	Overhead(%)
Gate counts	80000	83174	4.0%

6. CONCLUSION

In system designs where the processor core is embedded within an ASIC design, the traditional methods of debugging are often no longer suitable and an embedded debug architecture becomes necessary to ensure that the system can beat the time to market requirement of today's complex systems.

We designed the Reusable Embedded Debugger (RED) reusing the boundary scan architecture for the purpose of overcoming test and debug problems of huge investment and a plenty of time that traditional method needs. And we added to an additional hardware (Embedded ICE Module : EICEM) for more critical real-time debugging.

We synthesized the RED with 0.25μm design rule and got the result that total gate counts are 3174 gates and maximum operating frequency is 385M Hz

We embedded the RED in the SPK-611A, the 32bit RISC processor developed by Yonsei university and got the result of 4.0 percent of area overhead.

7. REFERENCE

[1] Harry Bleeker, Peter van den Eijnden and Frans de Jong, "Boundary-Scan Test : A Practical Approach", Kluwer Academic Publishers, 1993.

[2] Kenneth P.Parker, "The Boundary-Scan handbook", Kluwer Academic Publishers, 1992.

[3] IEEE Standard 1149.1-1990 Test Access Port and Boundary Scan Architecture, (ANSI/IEEE), IEEE, Piscataway, NJ, 1990.

[4] Ing-Jer Huang, Hsin-Ming Chen and Chung-Fu Kao, "Reusable Embedded In-Circuit Emulator", Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific , 30 Jan.-2 Feb. 2001.

[5] Gustavo R. Daniel Aga, Ovidiu Mosuc and J. M. Martins Ferreira, "Debug and Test of Microcontroller Based Applications using the Boundary Scan Test Infrastructure," in Student Forum within the IEEE International Symposium on Industrial Electronics, IEEE Industrial Electronics Society, 1997.

[6] Gustavo R. Alves and J. M. Martins Ferreira, "From design-for-test to design-for-debug-and-test: analysis of requirements and limitations for 1149.1", VLSI Test Symposium, Proceedings. 17th IEEE , 1999.

[7] Jurgen Haufe, Christoph Fritsch, Matthias Gulbins, Volker Luck and Peter Schwarz, "Real-Time Debugging of Digital Integrated Circuits", Proc. Design, Automation and Test in Europe Conference, User Forum, Paris, March 27-30, 2000.

[8] Moon Key Lee, Byeong Yoon Choi, Kwang Yub Lee and Seong Ho Lee, "Data-staionary controller for 32-bit application-specific RISC", Proceedings of ISCAS'93, Vol.3, pp.1933-1936, Chicago, May 3-6, 1993