Design of Halt-Mode and Monitoring-Mode On-Chip Debugger 2G for Core-A

Xuelong Xu, Jingzhe Xu, Donghoon Lee, Daekeon Park, and Jusung Park

***Abstract*—Nowadays, the SoC is concentrated by all over the world with interest. The design trend of the SoC is hardware and software co-design which includes the design of hardware structure in RTL level and the development of embedded software. As the complexity of SoC design increases with technology development, the observability of the SoC's internal state is no longer easy to achieve. Because of the above reasons, debugging the SoC system becomes very difficult and time-consuming. So we need a reliable debugger to find the bugs in the SoC and embedded software. In this paper, we deal with implementation of a hardware debugger named OCD2G which is based on IEEE 1149.1 JTAG standard and supports halt-mode and monitoring-mode debugging. In order to verify the operation of OCD2G, the designed debugger is integrated into the 32bit RISC processor - Core-A (Core-A is an embedded processor designed in South Korea) and is tested by interconnecting with software debugger.**

***Index Terms*—Debugger, JTAG, On-Chip Debugger, Processor.**

I. Introduction

Some bugs about the SoC (System on Chip) or its application programs will appear only when executing the applications in real. OCD2G (On-Chip Debugger Second Generation) is considered as the act for adding debug support to the SoC under the realization that not every application will work correctly at the first time. Thus OCD2G is very necessary for SoC design and can reduce time-to-market and design cost.

Most of the SoCs include the RISC processor which is integrated with its unique debug unit. In this paper, we proposed an OCD2G which can support the halt-mode and monitoring-mode debugging. OCD2G can observe the internal operation of the SoC due to its several debugging functions such as breakpoint assignment and detection, internal state observation and modification, single step, run and stop.

Manuscript received October 20, 2012; revised November 30, 2012. This work was supported by the Pioneer R&D Program for converging technology through the Korea Science and Engineering Foundation, funded by the Ministry of Education, Science and Technology, Rep. of Korea (M10711270001-08M1127-00110, Development of Displacement Sensing CMOS Circuit and Pattern Recognition System) and the Industrial Strategic Technology Development Program funded by the Ministry of Knowledge Economy, Rep. of Korea (No. 10039173, The development of system semiconductor technology for IT fusion revolution).

The authors are with Electronics Engineering Department at Pusan National University, Rep. of Korea. Busan, 609-735 Korea (e-mail: xuxuelong@pusan.ac.kr; kchuh@pusan.ac.kr; kdhyi17@nate.com; nsodium@pusan.ac.kr; juspark@pusan.ac.kr).

II. Related Work

The following Fig. 1 shows the target system which embeds the OCD2G to prove the functionality of debugger. The system embeds the main processor – Core-A [1] and many peripherals with bus connection. The bus type will be an AMBA or a Wishbone bus. When doing the verification, the OCD2G will be embedded in Core-A.

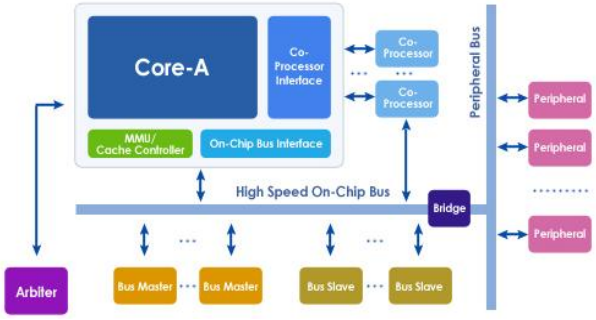


Fig. 1. Core-A SoC platform

Core-A is a general purpose 32-bit microprocessor. Its architecture is based on RISC (Reduced Instruction Set Computer) and has the Harvard architecture. It has 5 stage pipelines – fetch, decode, execute, memory and write-back stage. Core-A is implemented with simple hardware structure and has small gate count. It can process DSP programs and has efficient code density. The processor also has the interface for coprocessor such as FPU. Core-A can be sponsored to anyone in Korea in type of fully synthesizable soft IP.

III. On-Chip Debugger 2G

The proposed OCD2G offers system control and internal state control function to observe the operation of the SoC. System control function can halt and resume the program flow executed in the SoC and internal state control function can read and write both the register and memory values. Due to the above 2 functions, software debugger [2] can execute several debugging operations by OCD2G. OCD2G can debug the target in method of inserting debug clock called halt-mode debugging. Also it can do debugging functions in exception method called monitoring-mode debugging.

A. Architecture

In order to achieve the above functions, as shown in Fig. 2, OCD2G is composed of 4 main modules. First, JTAG is used as intermediate for transferring data between software debugger and OCD2G through emulator board. Second,

DOI: 10.7763/IJIEE.2013.V3.327

On-Chip Emulator (OCE) offers breakpoint setting and detection. Third, MM_COP supports the coprocessor interface for monitoring-mode debugging. Fourth, Debug Control Unit (DCU) is the most important module in OCD2G. DCU controls the functions of the OCD2G to execute the debugging operations and supports halt-mode and monitoring-mode debugging.

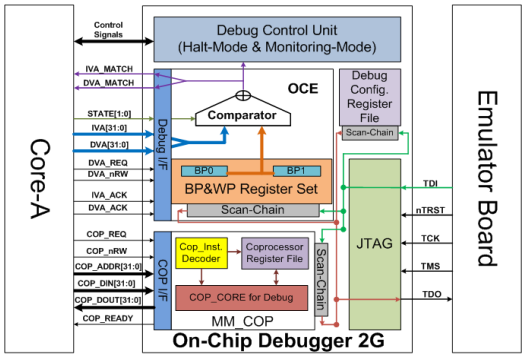


Fig. 2. OCD2G Architecture

1) JTAG

JTAG [3] module uses only 5 I/O pins to interface the software debugger and OCD2G. SoC includes so many IPs that there are not enough pins available. Therefore, we chose the JTAG protocol for transferring data between hardware debugger and software debugger.

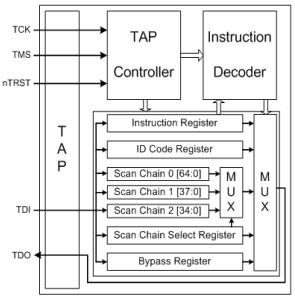


Fig. 3. JTAG Block Diagram

As shown in Fig. 3, the designed JTAG module includes TAP, TAP controller, 3 scan chains and several registers. It delivers data by using scan chains.

- Scan chain 0 includes 65 bits which is connected with 32bit instruction bus, 32bit data bus and 1bit control signal. It executes the data transferring function between OCD2G and processor.
- Scan chain 1 is 38bit long and can access the OCE's registers. The roles of scan chain 1 are breakpoint assignment and debug state control.
- Scan chain 2 is 35bit long and can access the MM_COP's registers. It transfers data between MM_COP and software debugger.

2) On-chip emulator

OCE does the function of breakpoint assignment and detecting. It is composed of 2 breakpoint register sets and a comparator as shown in Fig. 2. We can assign the breakpoint through the scan chain 0 and detect the breakpoint by comparing the value of breakpoint register set and address bus.

It is quite necessary to have the debugging functions such as single step, range breakpoint, coupled hardware and software breakpoints for system debugging. In order to use the debugging functions, there must be at least 2 breakpoint register sets. So we implement 2 breakpoint register sets in OCE block.

3) Monitoring-mode coprocessor

MM_COP is a coprocessor for monitoring-mode debugging. It contains the coprocessor interface part and 4 registers for transferring data between software debugger and Core-A. Monitoring-mode debugging forces the Core-A into exception for debugging the processor and does not halt the target. The following Fig. 4 shows the detail information about MMCR. We can access the MMCR through the scan chain 2 and do the monitoring-mode debugging by setting the MMCR.

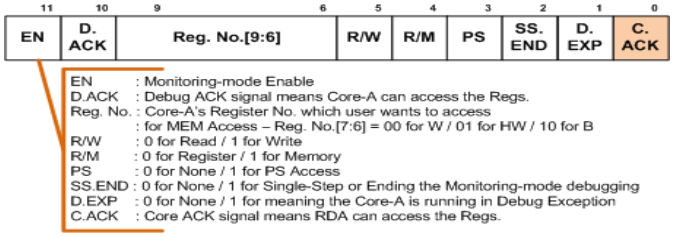


Fig. 4. Monitoring-Mode Control Register (MMCR) format

4) Debug Control Unit

DCU is the kernel block in OCD2G. It controls the debug mode entry and exit of processor. The timings of debug mode entry and exit are different for breakpoint types, breakpointed instructions and debugging mode. For deciding the timings, DCU refers to the 5 signals – IE (Instruction End), IV (Instruction Valid), MI (Multi-cycle Instruction), BI (Branch Instruction) and DM (Debugging Mode).

There are 3 major behaviors needed for debugging functions. First is the debug mode entry by breakpoint or watchpoint. Second is the automatic transferring of (debug mode)-(system mode)-(debug mode) for memory access. The third is the debug mode exit for resume. As shown in Fig. 5, DCU controls the behaviors and generates only one signal – pcon (Processor Control) to manage the processor's operation for debugging. The pcon signal controls the internal pipeline of target in halt-mode debugging and forces the Core-A into exception for monitoring-mode debugging.

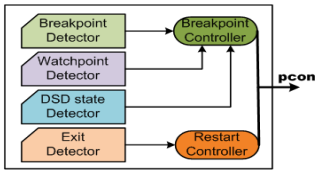


Fig. 5. Debug Control Unit

B. Mechanism

There are 2 control mechanisms of controlling OCD2G to debug the processor. One is halt-mode mechanism and the other one is monitoring-mode.

1) Halt-mode debugging

Halt-mode mechanism [3] is shown in Fig. 6. The designed OCD2G takes the debug mode which gives the control authority of processor to OCD2G for debugging. In

debugging procedure, OCD2G repeats the debug mode entry and exit. Processor entries into debug mode for debugging and exits from debug mode and enters into system mode after ending debugging. While checking the processor's state, the processor will entry into debug mode if OCD2G detects the breakpoint or watchpoint. During the debugging, all debugging functions will be realized by register read/write, memory read/write and debug mode entry/exit.

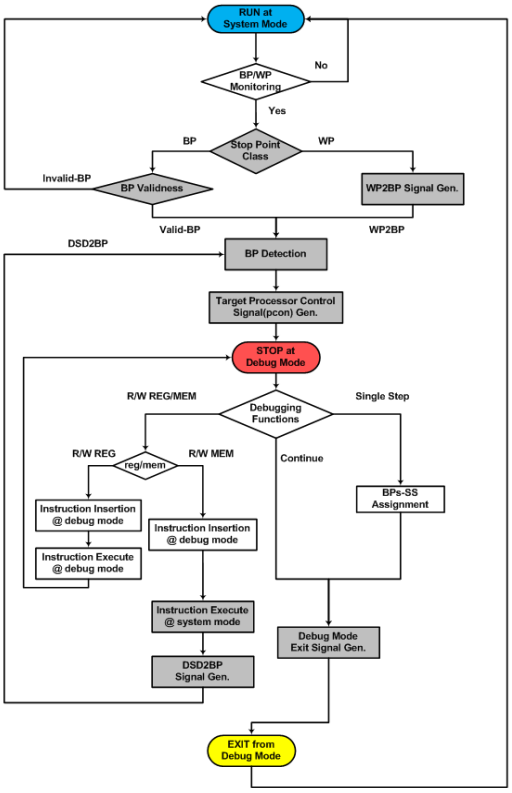


Fig. 6. Halt-Mode Control Mechanism

In halt-mode debugging, OCD2G uses the instruction insertion method to debug the processor. Processor enters into debug mode by the request of OCD2G and be controlled by debugging instructions inserted through scan chain 0 connected with 32bit bus. The procedure of halt-mode debugging is shown in Fig. 7. We show the case of appointing a breakpoint in the particular program address "N".

Firstly, we assign the breakpoint register set of OCE as "N" through scan chain 1.

Secondly, OCE will active the BREAKPT signal as 1 when comparator detects the value "N" carried on address bus.

Thirdly, after receiving the BREAKPT signal, DCU checks the processor's state to determine the validation of breakpoint. Then, it changes the main clock as debug clock and controls processor to enter into debug mode when "N+16" shows address bus.

Forthly, during debug mode, we can debug the target through software debugger by executing some debugging functions.

Fifthly, after finishing the debugging, software debugger restores the values and the processor will resume its previous state.

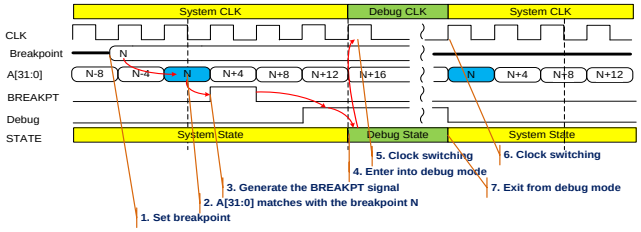


Fig. 7. Halt-Mode Debugging Procedure

2) *Monitoring-mode debugging*

Monitoring-mode mechanism is shown in Fig. 8. The procedure of monitoring-mode is almost the same with halt-mode. The difference is that the monitoring-mode does not halt the Core-A for debugging.

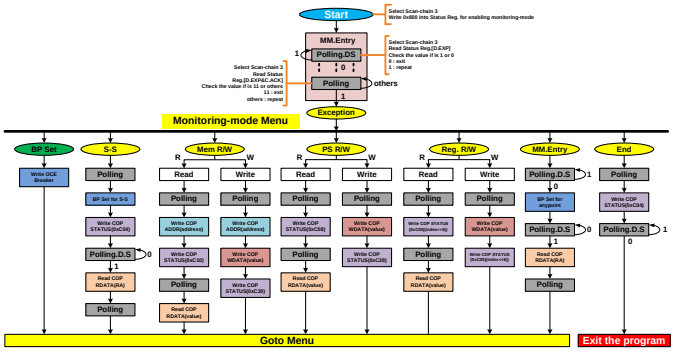


Fig. 8. Monitoring-Mode Control Mechanism

The procedure of monitoring-mode debugging is shown in Fig. 9.

Firstly, we assign the breakpoint register set of OCE through scan chain 1. Then, OCE will active the BREAKPT signal as 1 when comparator detects breakpoint or watchpoint.

Secondly, after receiving the BREAKPT signal, DCU checks the processor's state to determine the validation of breakpoint. Then, it forces the Core-A into exception for monitoring-mode.

Thirdly, during debug mode, we can debug the target through software debugger by executing service routine of exception for monitoring-mode.

Fourthly, after finishing the debugging, the Core-A will return from exception into system mode and the core will resume its previous state.

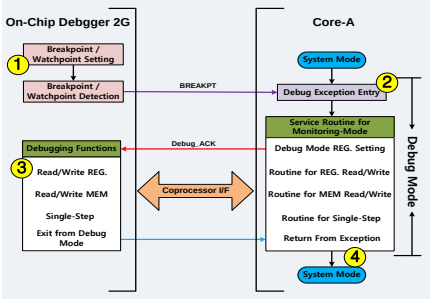


Fig. 9. Monitoring-Mode Debugging Procedure

In debug mode both for halt-mode and monitoring-mode debugging, OCD2G offers several abilities - register read/write, memory read/write and breakpoint assignment.

We can implement all software debugging functions by using the above abilities of OCD2G.

IV. Verification and Result

We have done 2 stage verification procedures – functional level simulation by simulator and FPGA level verification. Functional level simulation costs much time, but has the benefit that can apply desired various methods and algorithms in verification. FPGA verification tests the applications in real time, so it can do the verification about timing-sensitive and exception cases.

In functional simulation verification, we have tested the debugging functions of OCD2G in all considerable cases - single instruction test, combination instruction test, various application algorithms test and so on. The tested functions are breakpoint assignment, single step, register read/write, memory read/write and debug mode entry and exit for halt-mode and monitoring-mode debugging. With the help of the functional level simulation, we not only prove the OCD2G's functionality, but also the reliability of OCD2G.

Functional level simulation does not consider the gate delay, exceptions and real-time, so we do the FPGA level verification as well.

Before testing the OCD2G, we inspect if the target do the wrong operation. As things turned out, the processor with a built-in OCD2G works normally.

We have done the OCD2G verification by implementing the whole debug system as shown in Fig. 10. The Emulator Board which does the interface role between the target system and software debugger is made by out laboratory. Also the software debugger is. We modify the target dependent part of GDB [4]-[10] and link it with eclipse GUI(Graphic User Interface) to implement the software debugger. We embedded the OCD2G into Core-A SoC platform and downloaded it in FPGA and connected with software debugger through Emulator Board. In the debug system, we have tested all the debugging functions at desired timing for halt-mode and monitoring-mode debugging. The tested functions [11] are breakpoint assignment and detection, run and stop, single step, register read/write, memory read/write, variable read/write and so on. During the verification, Core-A executes several audio application algorithms such as ADPCM, SOLA and MP3. We also have done the test with changing the bus to Wishbone bus.

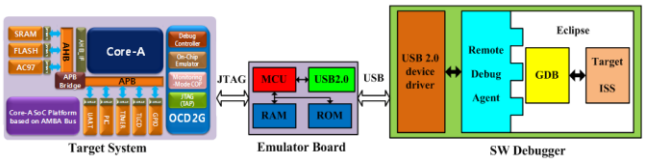


Fig. 10. Debug System

The synthesis result of the designed OCD2G by using 0.18μm CMOS cell library is shown in table 1. Gate count 1 is referenced by 2-input NAND gate. There is 17.20% gate count overhead.

TABLE I: Gate Count

Module name	Gate count
JTAG	2233
On-Chip Emulator (OCE)	2530
Monitoring-Mode Coprocessor (MM_COP)	1652
Debug Control Unit (DCU)	288
On-Chip Debugger 2G (OCD2G)	6627
Target processor with OCD2G	38532

V. Conclusion and Future Work

A. Figures and Tables

In this paper, we designed the debugger – OCD2G which can debug the RISC processor in halt-mode and monitoring-mode. We embed the OCD2G into Core-A SoC platform and test it by linking with emulator board and software debugger based on GDB and Eclipse GUI. Through the test, we prove the OCD2G's functionality and reliability as a debugger.

There are some more tasks to do in the future which improve the OCD2G's performance. The first one is the verification and application of the OCD2G with silicon chip. The second one is adding the real-time debugging function in the OCD2G.

References

[1] Core-A IP Manual. (2009). [Online]. Available: http://www.core-a.net/, 2009.

[2] Open On-Chip Debugger. [Online]. Available: http://openocd.berlios.de/web/

[3] *IEEE Standard Test Access Port and Boundary-Scan Architecture*, Test Technology Standards Committee, 2001.

[4] J. Xu, H. Park, S. Jung, J. Park, "Design and Verification of Efficient On-Chip Debugger for Core-A," *IEEK Semiconductor & Devices*, vol. 47, no. 4, pp. 50-61, Apr. 2010.

[5] R. Stallman, R. Pesch, and S. Shebs, "GDB User Manual: Debugging With GDB(The GNU Source-Level Debugger)," GDB version 6.4. Technical report, Free Software Foundation, Cambridge, MA.

[6] B. Gatliff, *Embedding with GNU: The gdb Remote Serial Protocol*. Red Hat Developer Network (RHDN), 1999.

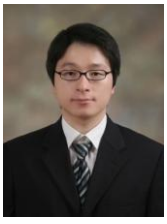
[7] M. Tan. (2002). A minimal GDB stub for embedded remote debugging. [Online]. Available: http://www1.cs.columbia.edu/~sedwards/classes/2002/w4995-02/tan-final.pdf

[8] S. Shebs, *GDB: An Open Source Debugger for embedded Development*. Red Hat, 2000.

[9] R. Pizzi , *GNU gdb Internal Architecture*, 1993.

[10] J. Gilmore and S. Shebs. (2004). GDB Internals, *Cygnus Solutions*, [Online]. Available: www.gnuarm.com/pdf/gdbint.pdf.

[11] J. Bennet. Howto: Porting the GNU Debugger. Practical Experience with the OpenRISC 1000 Architecture. (2008). [Online]. Available: http://www.embecosm.com/download/ean3.html



Xuelong Xu received the Bachelor's Degree majoring in Measuring&Control Technology and Instrumentations from Harbin Engineering University, Harbin, Heilongjiang, China, in 2010. He is currently working toward the Master of Science in electronics engineering at Pusan National University, Rep. of Korea. His research interests include microprocessor design, multi-media SoC platform implementation and low-power PCB design.

Jing-Zhe Xu received the BS in electronic communication engineering from Yanbian University of Science and Technology, Yanji, Jilin, China, and the MS in electronic engineering from Pusan National University, Busan, Rep. of Korea, in 2005 and 2008, respectively. He is currently working toward the PhD in electronics engineering at Pusan National University, Rep. of Korea. His research interests include microprocessor design, multicore platform implementation and on-chip debug architecture.

Donghoon Lee received B.S. and M.S. degrees in electronic engineering from Pusan National University, Busan, Korea, in 2005 and 2007, respectively. He is currently working toward a Ph.D. degree at the VLSI Design Laboratory, Department of Electronic Engineering, Pusan National University. His research interests include design of digital signal processor, design of multiprocessor SoC platform, and development of audio algorithm.

Daekeon Park received B.S. degrees in electronic engineering from Pusan National University, Busan, Korea. He is currently working toward the M.S. degree in electronics engineering at Pusan National University. His research interests include microprocessor design, M-PHY transceiver IP design and implementation.

Jusung Park was born in Junju, Korea, on 1953 December 19. He received a B.S degree in electronic engineering from Pusan National University, Busan,

Korea, in 1976, an M.S. degree in electrical engineering from KAIST, Seoul, Korea, in 1978, and a Ph.D. degree in electrical engineering from the University of Florida, Gainsville, US, in 1989. From 1978 to 1991 he was with ETRI, Daejun, Korea, as a principal research engineer, manger, and director of the IC design group. While at ETRI he designed several bipolar analog ICs and was in charge of developing VCR ICs, CMOS 8-bit microprocessors, and telecommunication chips. In 1991 he joined the Electronics Department of Pusan National University, Busan, Korea, where he is now a professor of electrical engineering. His current research interests are microprocessor and DSP core design, and digital audio algorithm implementation by hardware and software co-design.