Reusable Embedded In-Circuit Emulator for a Microprocessor Core in an SOC

Chung-Fu Kao, Ing-Jer Huang and Hsing-Ming Chen

Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan

cfkao@CADAL.cse.nsysu.edu.tw

Abstract

In this paper, we introduce a testing and debugging mechanism for embedded microprocessor core in an SOC. This testing mechanism is "embedded in-circuit emulator" (ICE) that designed by soft-IP style. Our ICE IP provides some features like CPU transparency, emulation the microprocessor, and real-time hardware breakpoint, etc. We also implement ICE IP to ASIC chip and FPGA.

1. Introduction

During the chip development cycle, there are two important parts: testing and debugging. Many SOCs are microprocessor-based design, that mean these SOCs have their own microcontroller or microprocessor core. Sometimes these microprocessor cores are IP cores, system integrators don't understand these microprocessor cores design. When system integrator integrates micro-processor IP core and other logic blocks into a single chip, they can not make sure that whether the microprocessor IP core can work correctly.

We introduce an ICE IP design that can support testing and debugging for embedded microcontroller/ microprocessor core in an SOC. Also, ICE IP can easily embedded into the SOC. The schematic view of our ICE IP and target microprocessor IP core is shown in **Figure 1**.

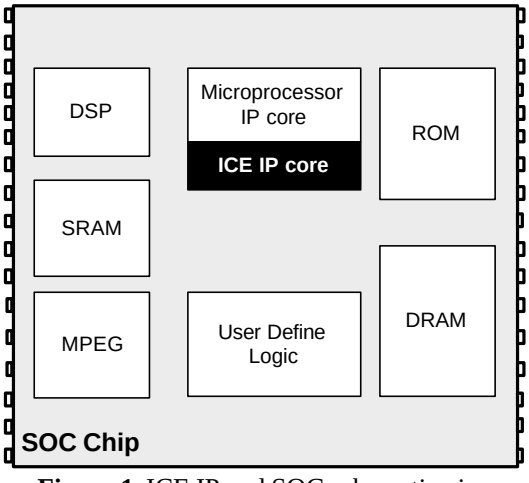

Figure 1. ICE IP and SOC schematic view

Figure 2 shown the block diagram of ICE IP and microprocessor IP core interconnecting.

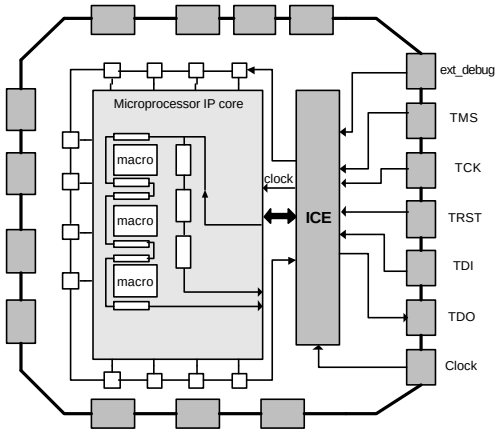

Figure 2. The interconnecting between ICE IP and microprocessor IP

2. ICE IP Architecture

2.1 Overview

Our ICE IP design is based on IEEE Std. 1149.1 JTAG and boundary scan architecture [1]. We use the TAP finite state machine to control signals sequence and enhance the IEEE Std. 1149.1 architecture in our design. In ICE IP design, we add some components to ICE IP, e.g. breakpoint register and breakpoint detection unit (BDU) [2][3]. After ICE IP embedded into SOC with micro-processor core, system integrator can test and debug the microprocessor IP core.

Test function allows system integrator to diagnose the fault, e.g. stuck-at, bridge fault, which cause in implement stage. The test function is achieved by using boundary-scan cells that follow the IEEE Std. 1149.1.

Debug function allows system integrator to trace and observe the register's value in the microprocessor. Breakpoint detection unit can achieve this function. The system integrator can input breakpoint to ICE breakpoint register, ICE will stop the microprocessor clock when the breakpoint is matched. System integrator can observe the

register's value by macro-boundary scan cells which within the microprocessor IP core. **Figure 3** illustrates the ICE IP architecture, which we present in this paper.

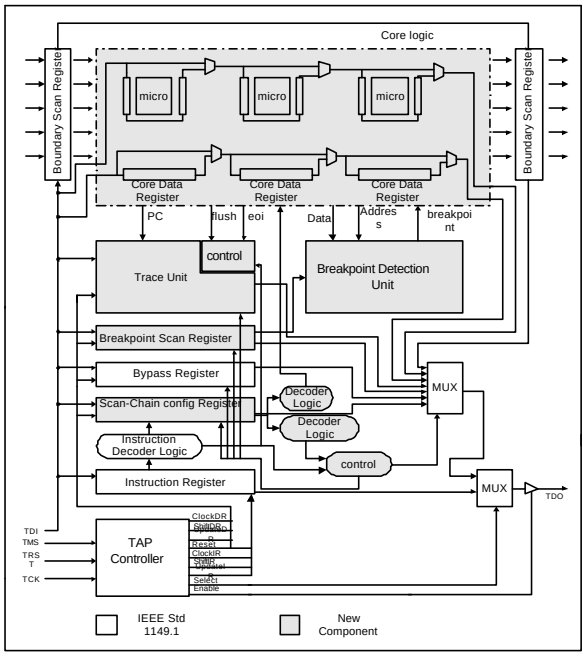

Figure 3. The overall architecture of our ICE IP

In figure 3, the white rectangles are IEEE Std. 1149.1 components. The gray rectangles are new components we add. The micro means Micro Boundary Scan Register, which is the same with Boundary Scan Register except in different level. Boundary Scan Register is in chip level, Micro Boundary Scan Register is in the micro level, e.g. ALU, PLA, special cores.

2.2 ICE Components

The ICE IP architecture has some components, which are IEEE Std. 1149.1-compliant, e.g. TAP controller, instruction register, instruction decode logic, bypass register, and boundary scan register [4][5]. Detail explanations of these components are available in [1], and are omitted here due to space constrains. Basically, the operation of ICE IP is depended on TAP finite state machine (FSM) to control signals properly. The TAP FSM diagram is shown in **Figure 4**.

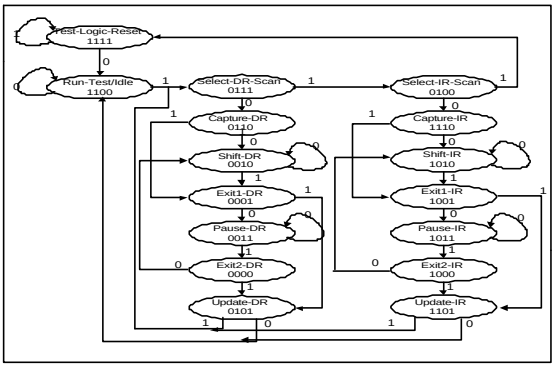

Figure 4. TAP finite state machine diagram

Scan-Chain Config Register is one of the test data registers. The TAP instruction (ConfigN) makes the Scan-Chain Config Register connected between TDI and TDO during the Capture-DR state. In our design, the Scan-Chain Select Register has two functions. First, during the Shift-RD state, the ID number of the desired scan path is shifted into the Scan-Chain Config Register. At the Update-DR state, the ID number will be decoded to select the test data register. Then the test data register will be connected between TDI and TDO. Second, it will be decoded to select the specific Core Data Register that in the microprocessor core. We use 5 bits to do this, so we have $32(2^5)$ different choice. Scan-Register Config Register has 9 bits in our design, 4 bits to select which test data register connected between TDI and TDO, 5 bits to select the specific Core Data Register. The 5 bits should be don't care (useless) in some case. **Figure 5** illustrates the architecture.

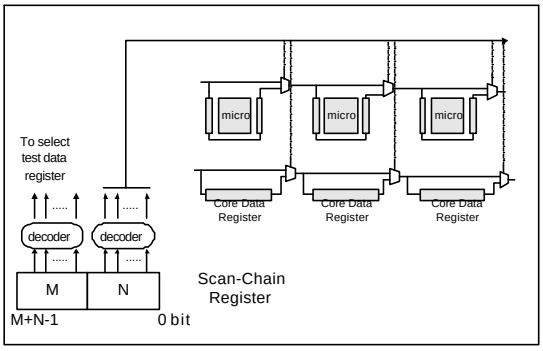

Figure 5. The architecture of Scan-Register, Config-Register

Breakpoint Scan Register is one of the test data registers. Its architecture is the same with Scan-Chain Select Register. The main function of Breakpoint Scan Register is serially accepted data from TDI and parallel translates data to the breakpoint detection unit. The width of Breakpoint Scan Register is depended on the width of target microprocessor's address/data buses, and also need additional 4 bits. The additional bits are used for select the register of the breakpoint detection unit which accept data from Breakpoint Scan Register.

In our ICE IP design, we provide a debug function that system integrator can interrupt the microprocessor clock and shift the register value in microprocessor to observe these values whether is correct. We use a module called Breakpoint Detection Unit (BDU) to detect whether the execution of microprocessor matches the breakpoint condition. If match, breakpoint detection unit will enable a *breakpoint* signal at a specified time to disable the execution of microprocessor. Then debug and test mechanism will execute the command that come from the debug host, e.g. PC or workstation. **Figure 6** illustrates the architecture of BDU.

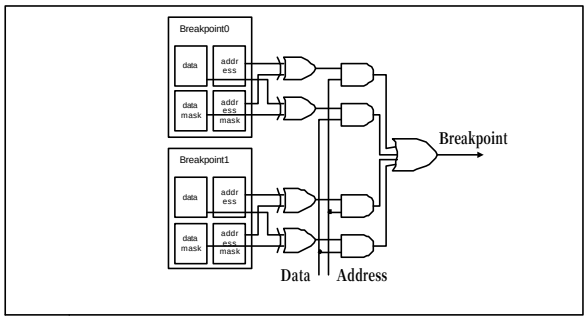

Figure 6. BDU architecture

2.3 ICE IP Configurable Parameters

Traditionally, ICE design is depended on target micro-controller/microprocessor, i.e. different microcontrollers / microprocessors will have different ICE architectures. We propose a ICE IP design that can fit all microcontrollers or microprocessors and also embedded into system chip with target microprocessor. Our ICE IP is belong to soft-IP, and design by Verilog HDL. We provide some configurable parameters for system integrators, so that they can tune these parameters and let the ICE IP fit their SOC design.

The configurable parameters of our ICE IP describe as following

- Instruction Number

  The system integrator can expand TAP instruction number. We provide 7 TAP instructions like Table 1 shown.

Instruction types	Numeric code
INTEST	0001
EXTEST	0000
BYPASS	1111
Config	1110
Restart	1100
RestartS	1101
RestartF	0010

Table 1. TAP instructions

  The system integrator can modify the instruction decode logic in order to add new instruction.

- Instruction Register width

  If system integrator add some new instructions, the width of instruction register (IR) may not enough, for example, the width of IR is 4, that means the maximum of instruction number is 2^4 = 16. So, if system integrator need 17 instructions, he must modify the width of IR.

- Breakpoint Scan Register width

  The width of breakpoint scan register (BSR) is depended on width of target microprocessor' s address or data bus. For example, if target microprocessor is 8-bit, the width of BSR only needs 8+4 = 12-bit; if target microprocessor is 32-bit, the width of BSR needs 32+4 = 36-bit.

- Register Number of BDU

  In the Breakpoint Detection Unit, there are four registers to achieve the breakpoint detection function. They are breakpoint_address, address_mask, breakpoint_data, and data_mask registers. Breakpoint_address register and breakpoint_data register are used for store the breakpoint value that system integrator input. The target microprocessor execution address and data will send to BDU to detect whether match the breakpoint value. If BDU detected the microprocessor execution address same as breakpoint value, the breakpoint signal will rise, the microprocessor clock will be stopped.

  If system integrator hope that he can input 2 or more breakpoint values to BDU, he can add 4 new registers or more (4 multiple). On the other hand, that will increase chip area and gate count.

3. ICE IP Verification Strategy

3.1 ASIC Chip Implementation

When we design ICE IP, we also design an 8-bit microcontroller HT48100 using Verilog HDL[1]. The HT48100 microcontroller is used for ICE IP target microcontroller. We integrate ICE IP and HT48100 into a single chip to verify and test ICE IP.

There are two stages to implement ICE IP chip, RTL and gate level simulation, run HOLTEK® Verilog Design Kit[2] respectively. The flow chart of RTL and gate level simulation is shown in **Figure 7**.

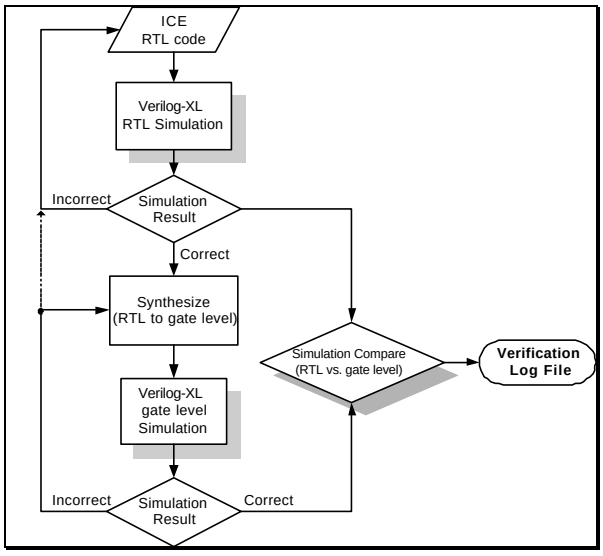

Figure 7. RTL and gate level simulation flow chart

After we compare the simulation result of RTL vs. gate level and make sure that simulation is correct, we run the Verilog Design Kit that HOLTEK® provide. After we finish the process, we send this chip gate level code to HOLTEK to tap-out this chip.

---

[1] HT48100 8-bit microprocessor has been tap-out by HOLTEK® Semiconductor Inc.

[2] Verilog Design Kit is HOLTEK® product for ASIC pre-simulation CAD Tool.

The chip that contains ICE IP and HT48100 layout view and gate count are shown in **Figure 8** and **Table 2**.

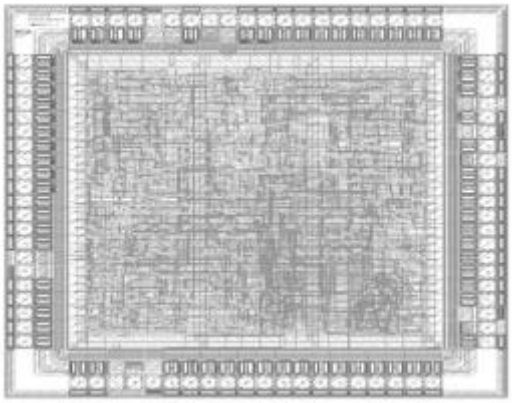

Figure 8. HT48100+ICE layout view (0.5μm)

HT48100+ICE	Chip Area	Gate Count
	2721 x 2822 um	7786

Table 2. Chip Area

3.2 FPGA Emulation

We use Xilinx® FPGA Environment Tool and XC4010XL™ chip to implement and emulate the ICE IP function and circuit. In order to prove that our ICE IP is really a soft-IP design, we use bus model method and different target microcontroller/microprocessor to implement our FPGA design. **Figure 9** is our FPGA emulation environment schematic view.

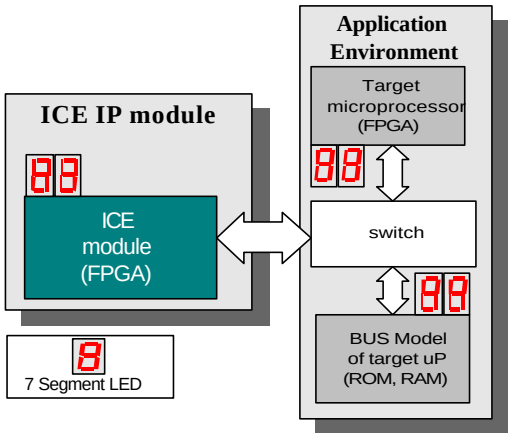

Figure 9. The ICE IP FPGA emulation environment

In figure 9, the target microprocessor can be any microcontroller/microprocessor, like HT48100, ARM7-like, or WDC6502, etc. The bus model of target micro-processor means that if the microprocessor does not implement yet, microprocessor provider can provide the behavior I/O which ICE needs, then we can emulate and test at the first time.

4. ICE IP Application Example

Now, we have two ICE IP application examples, they are HT48100+ICE IP and ARM7-like+ICE IP.

- HT48100 + ICE IP

  We have described in section 3.1. The chip information is shown in **Table 3** (0.5μm).

Design	HT48100	HT48100 + ICE IP
Area	5729	7786

Table 3. The chip information about HT48100

- ARM7-like + ICE IP

  We also design a 32-bit microprocessor based on ARM7TDMI, using Verilog HDL. We have passed the RTL and gate level simulation.

Design	ARM7-like	ARM7-like + ICE
Area	33324	41667

Table 4. The analysis of design area of ARM7-lile μP

5. Conclusion

In this paper, we have presented the concept of a structured and scalable testing and debugging mechanism: ICE IP, for embedded microcontroller/microprocessor core. The main objective of core-based IC design is improvement of design efficiency and time-to-market. In order to prevent test development from becoming the bottleneck in the entire development trajectory, choose a easy and fast testing and debugging mechanism is mandatory.

Our ICE IP is based on IEEE Std. 1149.1, that means our ICE IP not only used for microprocessor testing and debugging, but used for another IP core or user define logic (UDL) testing and debugging.

We also implement ICE IP in ASIC chip and FPGA field to prove our ICE IP design is correct and useful. We try to apply our ICE IP to an SOC with multiple microprocessor IP cores [6][7].

Reference

[1] "IEEE Standard Test Access Port and Boundary Scan Architecture," IEEE Std 1149.1-1990.

[2] Tai-An Lu and Ing-Jer Huang, "IEEE International Symposium on Consumer Electronics," 1998.

[3] Gernot Koch, Udo Kebschull, Wolfgang Rosensitel, "Breakpoints and Breakpoint Detection in Source Level Emulation," ISSS 1996.

[4] AGRAWAL, V.D. and SETH, S.C., "Test generation for VLSI chip," IEEE Computer Society Press, 1988.

[5] BENNETTS, R.G., "Design of testable logic circuits," Addison-Wesley, 1984.

[6] Bhattacharya, D., "Hierarchical test access architecture for embedded cores in an integrated circuit," VLSI Test Symposium, 1998. Proceedings. 16th IEEE, 1998, Page(s): 8 –14

[7] C. A. Papachristou, F. Martin and M. Nourani, "Microprocessor Based Testing for Core-Based System on Chip," Proc. DAC 99, Page(s): 586-591.