A novel method for on-chip debugging based on RISC-V processor

Shan Gao[1,2], *Dehua* Wu[1,2], *Wan'ang* Xiao[3,4], *Zetao* Wang[1,2], *Zhenghong* Yang[5], and *Wanlin* Gao[1,2,*]

[1]Key Laboratory of Agricultural Information Standardization, Ministry of Agriculture and Rural Affairs, China Agricultural University, Beijing 100083, China
[2]College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China
[3]Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China
[4]Center of Materials Science and Optoelectronics Engineering, School of Microelectronics, University of Chinese Academy of Sciences, Beijing, China
[5]Dept Math, China Agricultural University, Beijing 100083, China

Abstract. An on-chip debugging method based on the RISC-V processor is introduced, which simplifies the complicated debugging operation into instructions and improves the debugging efficiency effectively. The method adopts a JTAG interface to realize the debugging functions of the processor, such as running control, software breakpoint, hardware breakpoint and single-step execution. The method was verified by simulation at the RTL level, and the logic synthesis was carried out in SMIC 180nm process library.

Keywords: On-chip debugging, JTAG, RISC-V processor.

1 Introduction

RISC-V, incubated at the University of California, Berkeley, is a fifth-generation instruction set architecture based on the principle of the reduced instruction set. Its applications involve IOT (Internet of Things), high-performance computing, and so on. Most of the current instruction set architectures are protected by patents, which discourages small companies and limits the development and innovation of the processor industry. The RISC-V's open source and free feature have injected new vitality into the development of processors. With the increase of processor area and frequency, the development and debugging of software become more complicated, and the requirement of debugging means is higher and higher[1]. Good debugging features are designed to help software developers quickly locate errors. Therefore, debugging design is very important to promote the use of RISC-V processors, which can effectively promote the development of RISC-V processor ecology.

* Corresponding author: gaowlin@cau.edu.cn

JTAG debugging technology is based on the serial interface, and the complexity of debugging instruction has a great impact on the debugging speed. In this paper, the common debugging operations are simplified into instructions, avoiding the input of a large amount of data through the shift register chain.

2 Debugging techniques

Initially, the probe was connected directly to the PIN of the CPU to debug using online emulation, but the rapid increase in CPU speed made ICE difficult and costly to develop[2]. In the highly integrated SOC, it is difficult to get the on-chip CPU signal out of the off-chip, so it is increasingly necessary to support debugging on the hardware, and on-chip debugging comes into being[3]. BDM on-chip debugging technology is only for Motorola processors. It is built directly into the processor to get all the information of the kernel, and the debugging command is obtained through the serial interface to complete the corresponding operation. JTAG(Joint Test Action Group) is an open international standard developed by IEEE in the mid-1980s[4]. Most microcontrollers in the embedded field support JTAG. It can be used for on-chip debugging by extending JTAG instructions and adding logic inside the processor.

3 On-chip debugging implementation

3.1 Debug interface

To improve the debugging efficiency, as shown in Figure 1, the instructions corresponding to the debugging operation are directly input through the TDI port. After processing by the DM module, signals are sent to the CPU to control the operation of the processor.

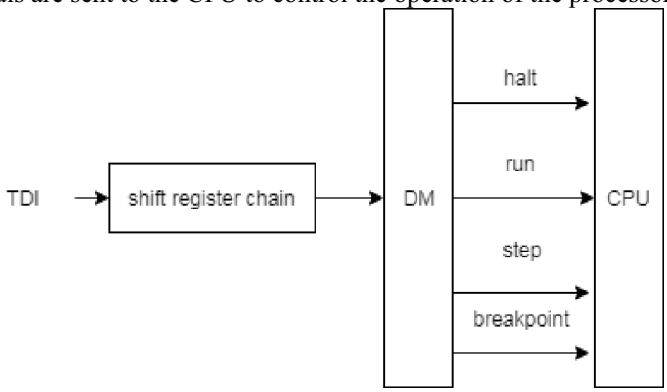

Fig. 1. Debug interface.

3.2 Overall structure

This debugging design is based on the two-stage pipeline RISC-V processor. As shown in Figure 2, the first level of the pipeline is fetching instructions, which is completed by IFU. The second level is implemented by the EXU unit and includes the capabilities of decoding, execution, delivery, and write-back. The hardware mechanism of this RISC-V architecture does not support hardware interrupt nesting. Once responding to interrupt requests and entering the abnormal mode, the interrupt is turned off globally and cannot respond to new interrupt requests. The debugging function is implemented in the processor as an interrupt.

The debugging module generates a debugging interrupt, which is passed to the delivery module to generate a pipeline flushing request to the finger fetching module, and then the whole pipeline is scoured.

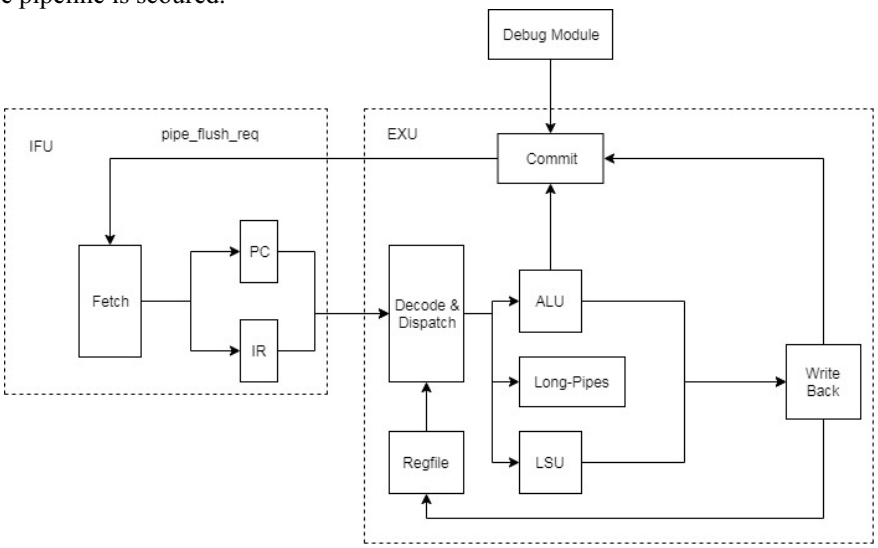

Fig. 2. The overall structure.

3.3. The kernel modifications

To support on-chip debugging, the processor needs to make the following changes: set the debug mode; add hardware breakpoint module; control pipeline; handle single-step execution.

3.3.1 Set the debug mode

Debug mode has higher permissions than machine mode. You can modify the relevant debug registers only in debug mode. After entering the debug mode, the processor saves the PC to the DPC and records the interrupt reason in the DCSR register. Debug mode prevents untrusted user code from interfering with the normal execution of debug operations.

3.3.2 Add hardware breakpoint module

Multiple hardware breakpoints can be included in the processor. When a breakpoint match occurs, debug mode is triggered. The registers in the module are tselect, tdata, and mcontrol. The tselect register is used to select hardware breakpoint modules. The tdata register is used to store the address at which the breakpoint will be generated. The mcontrol register is the control center of the hardware breakpoint which handles the behavior of the match and determines whether the instruction that triggered the breakpoint will be executed.

3.3.3 Control pipeline

Add pipeline flushing logic. when debugging interrupt occurs, start to get new instructions, then turn to debug exception processing.

3.3.4 Handle single-step execution

Add a register to hold the single-step state of the current instruction, which will mask other debug interrupt triggering conditions. The next instruction triggers an interrupt, thus single-step execution is completed.

Experiment

The debugging function of the design is verified by writing testbench in the Vivado integrated design environment of Xilinx. Adopting 180nm, the total area of the chip obtained by logical synthesis with Synopsys' Design Compiler is about 11mm^2.

Figure 3 shows the detailed process for the processor to step through the function. Inputting STEP and INTERRUPT commands in turn to the TDI port, trigger the debug interrupt to enter the interrupt service routine, then jump to the Debug RAM to execute the single-step program, and at last exit the Debug mode. After executing a program, the delivery signal cmt_ena arrives, then triggers a single-step exception. The simulation result is shown in Figure 4.

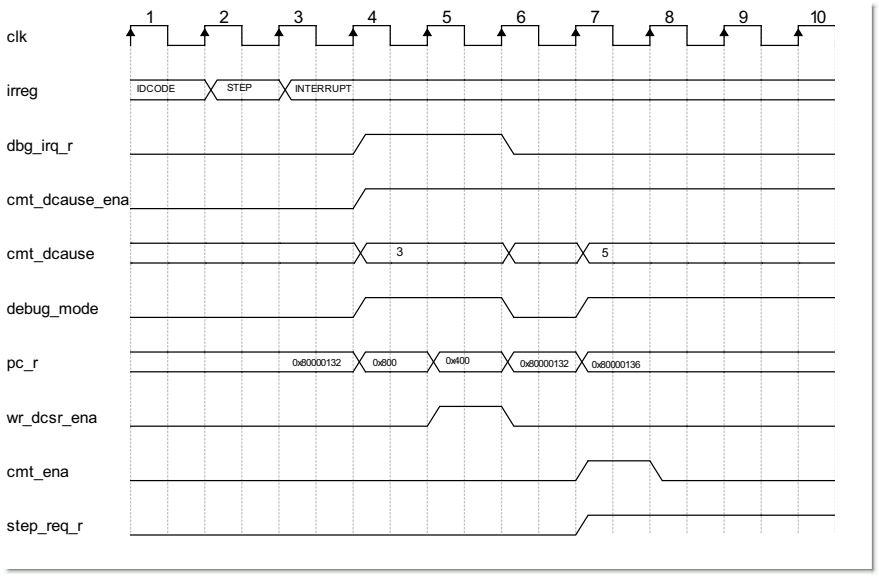

Fig. 3. Single-step execution waveform.

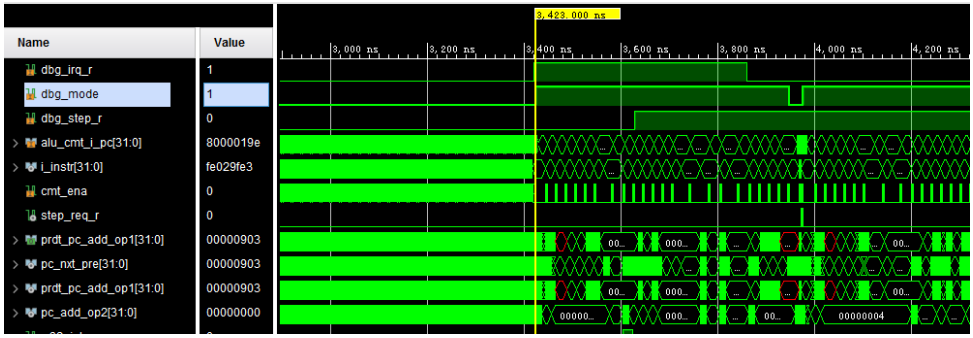

Fig. 4. The simulation results.

Conclusion

This paper implements a debugging design based on the exception and interrupt mechanism of the processor, which greatly facilitates the software development and application program optimization based on the RISC-V processor. The modified part of the kernel for debugging is introduced in detail, and the function is verified by simulation. Finally, the logic synthesis of the whole chip is carried out.

References

1. HOPKINS A B T, MCDONALD-MAIER K D., "Debug Support for Complex Systems On-Chip: A Review," IEE Proceedings - Computers and Digital Techniques, vol. 153, pp. 197, 2006.
2. L. Peng, Y. Li-xin, Q. Hui, Z. Hai-yang, "Summary on Debug Technique of Common Embedded Processors", MICROPROCESSORS, vol. 04, pp. 16–20, 2011.
3. B. Du, M.S. Reorda, L. Sterpone, et al., "Online Test of Control Flow Errors: A New Debug Interface-Based Approach", IEEE Trans. Comput. vol. 65, pp.1846–1855, 2016.
4. IEEE Std P1149.1-IEEE Standard Test Access Port and Boundary-Scan Architecture, 2001.