A fast on-chip debugging design for RISC-V processor

Shan Gao1,2, Wan'ang Xiao3,4, Zhenghong Yang5, Dehua Wu1,2 and Wanlin Gao1,2*

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

*Corresponding author. Email: gaowlin@cau.edu.cn

Abstract. In order to improve the efficiency of on-chip debugging, a fast on-chip debugging design is proposed, which adopts JTAG interface and is applied in RISC-V processor. In this paper, we extend some debugging instructions, effectively reducing data entry by operating the debugging bus directly, and realize the breakpoint, pause, single step, et al., providing conveniences for the development and debugging of the software system.

1. Introduction
Riding on the rapid development of semiconductor technology, the integrated level of processors has been progressively increasing, making it more difficult to detect errors in the process of using[1,2]. Good debugging functions can help software developers locate errors quickly, especially in complex projects. Debugging design is very important to promote the popularization and application of processors.

On-chip debugging is the idea of extending additional designs on a processor to control how it runs and have a look inside[3,4]. This method of debugging is cost-effective, efficient, and non-intrusive to the application, so it is widely used. JTAG boundary-scan technology [5](IEEE1149.1) was originally used in the test of integrated circuits, which is now widely used in on-chip debugging because of the flexible and efficient characteristics. Debugging based on JTAG interface needs to input information repeatedly through TDI port, so the amount of input data has a great impact on the efficiency of debugging[6]. This paper presents a debugging circuit, which reduces the operation of boundary-scan and the amount of input data.

2. The implementation of on-chip debugging
This debugging design is integrated into the RISC-V processor. The processor is oriented to IoT field, adopts a two-stage pipeline structure, supports RV32 instruction set, and has high energy efficiency ratio. The overall design of on-chip debugging for RISC-V microprocessor includes debugging interface module and debugging processing module. 1) Debugging interface module uses TAP state machine to realize the serial-parallel conversion between JTAG signal and bus signal, to make the
Debugger send instructions and data to the chip system, and in turn make the key information generated in the chip fed back to the debugger in time. 2) Connected to the RISC-V core, the debugging processing module analyzes the received debugging instructions, and then sends debugging interrupt, breakpoint, single step, et al. to the core. As shown in Table 1 are realized functions for the debugging design.

| function          | description                                         |
|-------------------|-----------------------------------------------------|
| halt              | stop the core when it is running                    |
| run               | activate the core after it’s halted                 |
| Single step       | stop the core after executing an instruction         |
| Reading           | read data from memory or registers                   |
| Writing           | Write to memory or registers                        |
| breakpoint        | Stop the core in the appointed pc                   |

2.1. Debugging interface module
As shown in Figure 1, the debugging interface module includes TAP controller, instruction register, data register group, instruction shift register chain, data shift register chain, data selector, and serial-parallel conversion mechanism. The ports which connect debugging interface module with the debugger are TDI, TDO, TCK, TMS, and TRST. The TAP state machine is used to generate control signals. The data register group includes device identification register, control register, debugging bus access register, debugging interrupt register, halt register, single step register, and bypass register. All data registers are connected to the TDO through the data shift register chain to realize the output of information in the registers. Only the debugging bus access register can be connected to the TDI through the data shift register chain to achieve data input. The debugging interrupt register and the single-step register are directly connected to the debugging bus access register, which avoids the input of data through TDI and improves efficiency. The debugging bus access register has a total of 41 bits, including a 2-bit operation field, a 34-bit data field, and a 5-bit address field. The serial-parallel conversion mechanism is controlled by the TAP state machine, which converts serial data from the TDI port into parallel data to the debug bus, and can also output feedback data from the debug bus through the TDO port.

**Figure 1.** Debugging interface module.
2.2. Debugging processing module

As shown in Figure 2, the debugging processing module includes reset control, debugging control, debugging bus comparator, debugging bus selector, debugging bus register group, debugging interrupt control, pause flag, system bus comparator, system bus selector, and system bus register group. The reset control uses a core reset signal to control the reset operation of debugging design to make both reset signals consistent. Debugging control is used to send breakpoint, single step, pause, running, and other debugging signals to the core in debugging mode. The debugging bus comparator is connected to the debugging bus selector to set up feedback, which is from debugging bus register group, Debug RAM or Debug ROM. Debugging interrupt control is used to control the interrupt signal. The pause flag is used to mark whether the core is halted due to debugging interrupt. The system bus comparator and the system bus selector are connected to set the content of the feedback to the core, including the system bus register group, etc.

![Figure 2. Debugging processing module.](image)

3. Experiment

The debugging circuit is implemented by Verilog HDL language, and the debugging module is mounted on RISC-V IP. The RTL simulation verification is finished by Vivado, which is the integrated design environment of Xilinx Corporation. The specific process is as follows: write an assembly program, compile the program into machine code with RISC-V GCC tool, and read machine code with designed testbench at last.

The detailed waveform of the halt function is shown in figure 3. Enter the command HALT by controlling the TAP state machine, then write the assembler program into the Debug RAM, and at last turn into the Debug mode with the INTERRUPT command. At this point, the PC jumps to 0x800 which is the entry of interrupt service routine, and then switches to 0x400, the location of Debug RAM, in which first set HALT and then firing the halt exception after performing a Debug interrupt cleanup operation. As shown in figure 4 is the experiment result.
4. Conclusion

In this paper, a fast on-chip debugging circuit based on RISC-V processor is proposed to improve debugging efficiency. Then, the principle of debugging design is analyzed in detail from the whole to the part, and the halt function is verified from RTL level.

References

[1] M.-C. Hsieh, C.-T. Huang, An Embedded Infrastructure of Debug and Trace Interface for the DSP Platform, 2008 45th ACM/IEEE Design Automation Conference 866-871 (2008).
[2] B. Du, M.S. Reorda, L. Sterpone, et al. Online Test of Control Flow Errors: A New Debug Interface-Based Approach, IEEE Trans. Comput. 65, 1846–1855 (2016).
[3] A.B.T. Hopkins, K.D. McDonald-Maier, Debug support for complex systems on-chip: a review, IEE Proc. Comput. Digit. Tech. 153, 197 (2006).
[4] L. Peng, Y. Li-xin, Q. Hui, Z. Hai-yang, Summary on Debug Technique of Common Embedded Processors, MICROPROCESSORS 04, 16–20 (2011).
[5] IEEE Std P1149.1-IEEE Standard Test Access Port and Boundary-Scan Architecture (2001).
[6] Y. Jun, D. Yun-fei, Design of High Speed JTAG On-line Simulator, Computer Engineering 37, 228 (2011).