A Novel Fast Recovery Method for HT Tamper in Embedded Processor

Wanting Zhou[1(✉)], Shiwei Yuan[1], Lei Li[1], and Kuo-Hui Yeh[2]

[1] Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, Chengdu, SC 611731, China
zhouwt@uestc.edu.cn

[2] Department of Information Management, National Dong Hwa University, Hualien 97001, Taiwan

Abstract. Nowadays, embedded processors face various hardware security issues such as hardware trojans (HT) and code tamper attacks. In this paper, a novel cycle-level recovery method for HT tamper in embedded processor is proposed, which consists two units, a General-Purpose Register (GPRs) backup unit and a PC rollback unit. The former one is designed to replace original register files with backup function extra. And the latter one is composed for rollback operations based on the exact PC address corresponding to the wrong instruction. If a HT tamper is detected, the backup unit works in conjunction with PC rollback unit allowing the processor to resume the instruction execution. The proposed method has been implanted into a RISC-V core of PULpino, and the experimental results show that the processor can restore from fault state caused by inserted HT in real time with the latency of 7 clock cycles, including 2 clock cycles for detection.

Keywords: hardware security · GPRs · fault recovery · embedded processor

1 Introduction

In recent years, processors are facing potential security risks due to hardware Trojan (HT) malicious attack [1–13] and code tamper attacks [14–16]. Facing the security issues abovementioned, a fast and effective recovery mechanism is important to protect processor working normal after a fault is detected.

Existing methods for processor fault recovery mechanism mainly include: 1) checkpoint backup and rolling back [17–26], 2) method on combing attack detection and fault repairing [27–30]. The former one is the most common method for recovery, which is based on building a checkpoint file of recording the state of executing program. If a fault occurs, the checkpoint file would be loaded to cover

Supported by University of Electronic Science and Technology of China.

W. Meng and W. Li (Eds.): BlockTEA 2022, LNICST 498, pp. 131–139, 2023.
https://doi.org/10.1007/978-3-031-31420-9_8

the state. There are some shortcomings of this method, including low real-time performance and higher requirements of memory resources. The latter one needs to build a basic block (BB) depended on monitoring and recovery architecture. However, this method leads to a large number of signatures for extraction of all executing program in advance. Besides, the BB dividing according special instruction may ignore the attack on other instructions.

To overcome these inadequacies mentioned above, a real-time recovery approach is proposed herein, which is an extension of our previous work [31] with performance improvement and hardware implantation. If an attack is detected, recovery mechanism is triggered, a GPRs backup unit with dynamic selection is responsible for restoring the right value which should write to GPRs. Meanwhile, the pipeline of processor is suspended, and the PC of the processor is replaced by the corresponding PC value at the exact moment of an attack occurs. Then, the proposed method has been verified into RI5CY, which is a 4-stage in-order RISC-V core of PULpino [32,33]. And the experimental results show that the processor can restore from fault state caused by HT attack only with a latency of 7 clock cycles, including 2 clocks for detection.

2 The Proposed Recovery Method

The fundamental idea of the proposed method is that the instructions are dispatched sequentially and executed sequentially for in-order processors. Instruction sequences in the program often have data dependencies. For example, consider the following two register instructions to a pipeline processor as shown in Fig. 1. The instruction 1 deposits the data into R1 as its execute stage at time t3. Instruction 2 would be decoded at time t3 by using R1 data, and its execution would be complete at t4. In this scenario, the result of current executed instruction is relied on the instruction itself and the GPRs data, which is updated from execution or write-back stage result of the previous instruction.

When a HT attack is detected, fast recovery scheme can be realized through re-executing the wrong instruction. If PC of wrong instruction, PC pointed at the exact time when the attack occurs, and backup GPRs data are provided simultaneously, cycle-level recovery mechanism can be built by PC rollback. Based on this idea, recovery scheme has been proposed which consists two units, a GPRs backup unit and a PC rollback unit. Besides, a GPR-State Real-Time Detection Module (GSRTM) unit, presented in our previous work [34], are adopted to monitor the state of GPRs, the indication signal will be generated to start the recovery operation once an attack detected. Meanwhile, the corresponding wrong instruction and its PC value are provided for rollback unit to perform recovery scheme. The hardware implementation has been implanted into RI5CY, which is shown in Fig. 2.

2.1 The Design and Implement of GPRs Backup Unit

According to analysis above, the GPRs backup unit should have two functions. It needs to bypass the EX or WB stage information which writes to GPRs

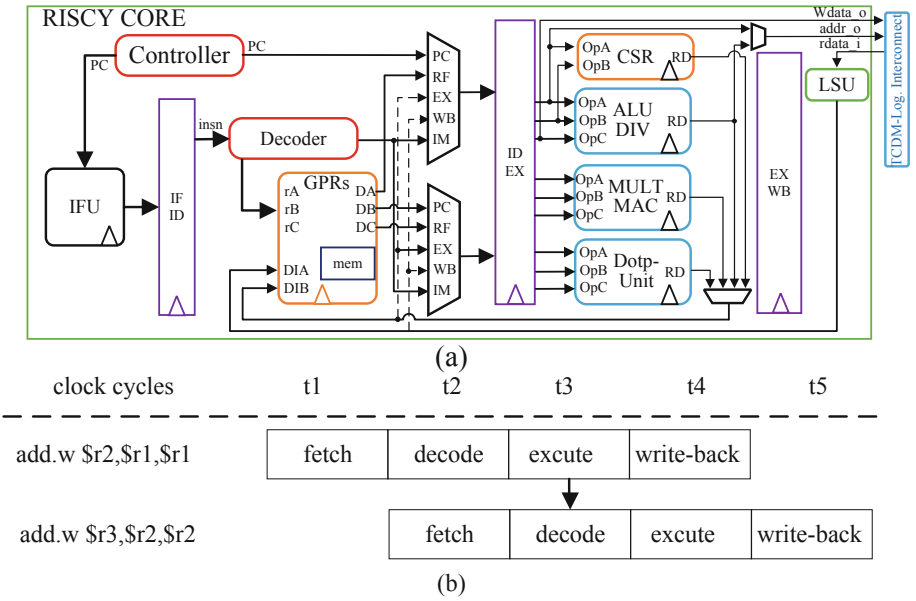

Fig. 1. The function of GPRs information in processor pipeline.

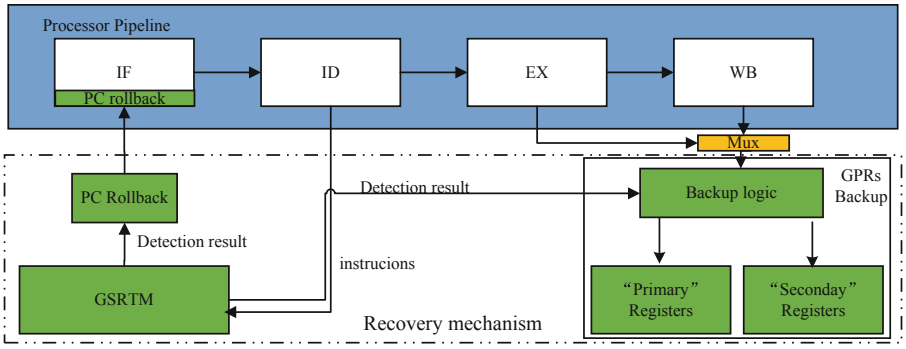

Fig. 2. System diagram based on backup and PC rollback techniques.

registers in normal execution of the program. In addition, it also stores the GPRs information corresponding to the previous executed instruction. Herein, two register groups are needed, "Primary" registers are used in normal work while "Secondary" registers are worked as backup information corresponding the previous executed instruction, which used for restoring. The registers information relationship of instruction life time (an instruction between ID stage to EX or WB stage) between "Primary" and "Secondary" is shown in Fig. 3.

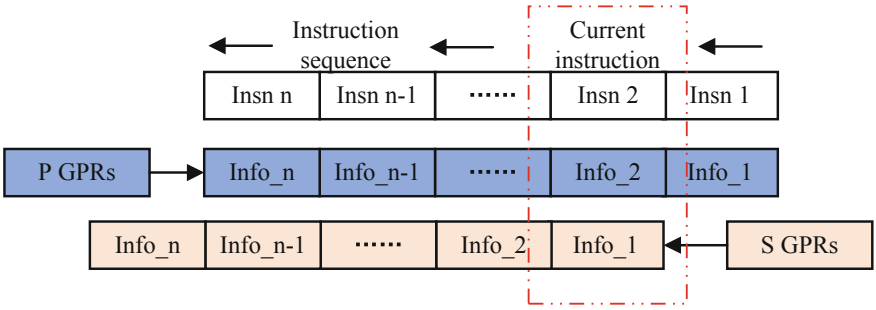

Fig. 3. The information relationship of instruction stream between Primary and Secondary GPRs.

In order to minimize the recovery time, a backup unit with dynamic selection through register label is built, which used to indicate work states of these two groups. If tag = 1, it works as "Primary" registers, else if tag = 0, it works as "Secondary" registers. When an attack is detected, label value is exchanged, and original "Secondary" registers are used as "Primary" registers, the backup information are output for restoring. In the meantime, the original "Primary" registers are used as "Secondary" registers working for storing. If next attack is detected, the process is carried out again. It should be noted that, only one group registers work as normal output, while another is used for storing instruction information corresponding to previous executed instruction.

Consequently, the implement of the proposed unit is given in Fig. 4. GPRs' value is updated from EX or WB stage result, we named this information as write channel, which is input of backup unit. The "Primary" just works as by a bypass channel without any extra process. While "Secondary's input signals are copying of the write channel signals with three clocks delay. Then latches and releases the write channel signal with the delay of three clock cycles according to the current instruction is over or not, signal insn_life_over generated by the GSRTM unit. As shown in Fig. 4, the detection result (signal_strong_warning) decides whether "Primary" registers or "Secondary" registers value should be chosen for output. It should be noted that, the "Primary" or "Secondary" group is decided by a tag, which can guarantee the backup unit would either affect the normal execution of the program or reduce the performance of the embedded processor.

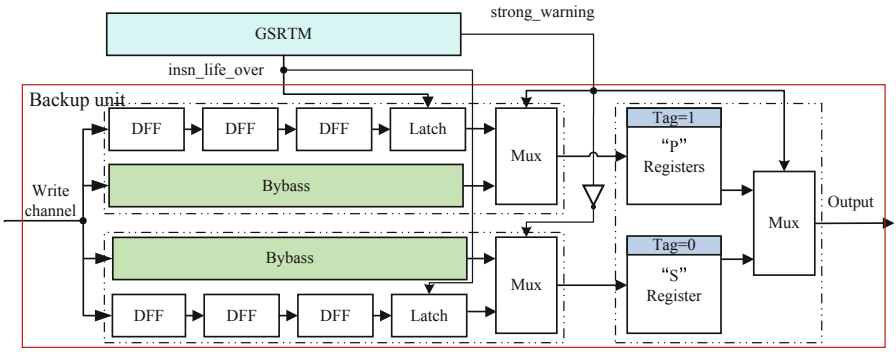

Fig. 4. The implementation of GPRs backup unit.

2.2 The Design and Implement of PC Rollback Unit

The PC rollback unit is similar to program rollback, which is responsible for rolling-back of the exact PC address corresponding to the wrong instruction rather than program checkpoint. Then, utilize the PC address and GPRs information storage in "Secondary" registers to realize fast recovery.

There is an instruction should be processed specially, the STORE instruction stores data from GPRs into a specified location in memory. Thus, if the wrong instruction is STORE, it is necessary to block this instruction from storing memory immediately to ensure that the memory information is corresponding to the previous instruction.

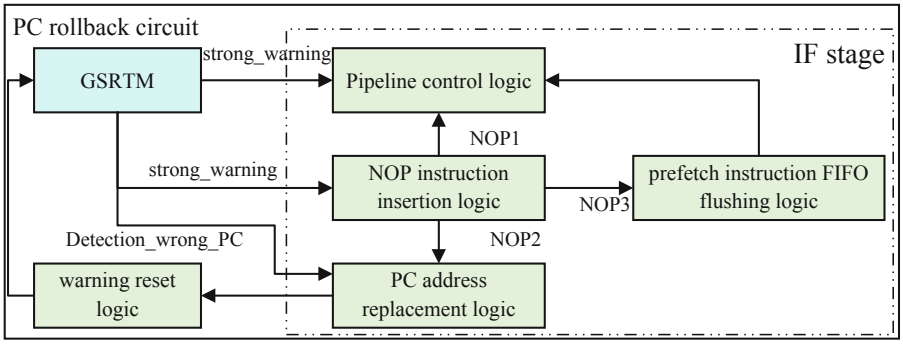

Fig. 5. The implementation of PC rollback unit.

The strong_warning signal is a key signal, which is used to indicate an attack occurs. In order to perform recovery, there are five steps: suspend pipeline, insert NOPs, PC replacement, flushing prefetch FIFO and resetting warning signal. The implementation of PC rollback unit is shown in Fig. 5 which has five logic parts corresponding to the steps mentioned above.

Firstly, pipeline control logic would generate stall signal to suspend the pipeline stages, including IF, ID, EX and WB stages for preventing the propagation of error. Secondly, three NOPs are inserted into instruction transmission path between IF stage and ID stage. Simultaneously, restore the pipelines of ID stage, EX stage and WB stage. This operation can clear the invalid state and useless control of the processor caused by suspending the processor pipeline, and can provide time for the roll back operations subsequently. Thirdly, the current PC address of processor in IF stage is replaced with the corresponding PC value at the time the error occurred when inserting the second NOP. Fourthly, the prefetch FIFO in the IF stage of the processor should be cleared, because this FIFO still stores the instructions that have not been executed before, and execution of these instructions are related to the information of GPRs. At the same time, restoring the pipeline of the IF stage. Lastly, when the instruction prefetched after the PC rollback is detected in the ID stage, it means that the PC rollback recovery has been completed, at the same time reset the warning signal generated by the GSRTM unit.

3 Experimental Results

In order to demonstrate the fast and effectiveness of the proposed method, six programs with different functions were implanted for test, and the test results were shown in Table 1, including number of HT implanted, detection number and recovery time. The experimental results showed that all six programs with 207 HTs had been detected and recovered with the latency of 7 clocks.

Table 1. Detection and recovery results.

Codes	HT number	Detected number	Recovery Time
add.c	5	6	7 cycles
testALU.c	50	50	7 cycles
testClip.c	38	38	7 cycles
testCnt.c	18	18	7 cycles
testMUL.c	82	82	7 cycles
testDivRem.c	17	17	7 cycles

Further, an example, a HT maliciously tampered with x15 of the GPRs, is used to explain the tamper flow which is shown in in Fig. 6. Moreover, the the recovery process which is shown in Fig. 7.

When GPRs are subject to malicious tampering attacks (x15 is changed from 4 to 5, the PC address corresponding to the abnormal moment is 0x0000_041C, and the instruction is 0x0010_0537), the inserted HT attacks can be recovered in real time with the latency of 7 clock cycles, including 2 clock cycles for detection.

The proposed method can realize fast recovery with cycle-level recovery conjunction with detection technology, which has performance improvement compared with the part work in [6],which has at least 100 us sample length for HT detection and more time for analysis and recovery.

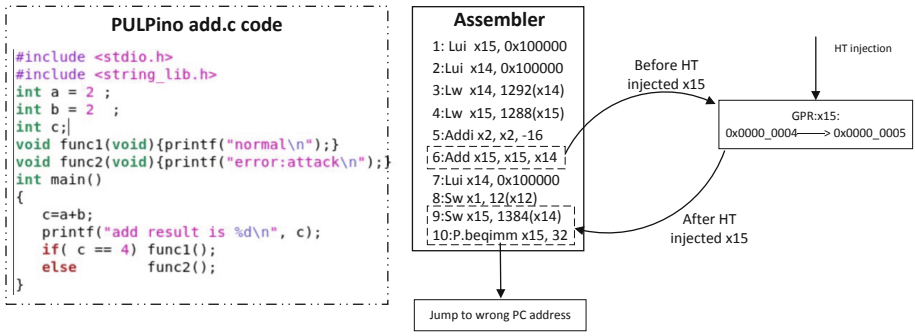

Fig. 6. An example of tamper attack on x15 of GPRs.

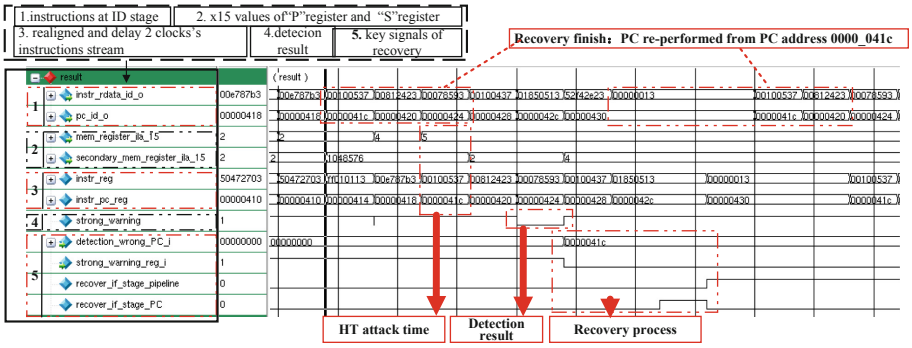

Fig. 7. Simulation wave of HT activated, detection and recovery mechanism.

4 Conclusion

This paper proposes a real-time recovery method after real-time detection, and its implemented way is also given, which has been verified in the RISC-V core of PULpion. The experimental results show that the proposed method can effectively guarantee that processor restored from abnormal state with the latency of 7 clock cycles.

Acknowledgments. This work is partly supported by Sichuan Science and Technology Program under Grant 2021YJ0082. And the authors would like to thank IC Team for providing advice and discussion.

References

1. Bhunia, S., Hsiao, M.-S., Banga, M., Narasimhan, S.: Hardware trojan attacks: threat analysis and countermeasures. In: Proceedings of the IEEE, pp. 1229–1247. IEEE (2014)
2. Kuo, M.-H., Hu, C.-M., Lee, K.-J.: Time-related hardware trojan attacks on processor cores. In: IEEE International Test Conference in Asia (ITC-Asia), pp. 43–48. IEEE, Tokyo (2019)
3. Okane, P., Sezer, S., McLaughlin, K., Im, E.: Malware detection: program run length against detection rate. IET Softw. **8**(1), 42–51 (2014)
4. Duflot, L.: CPU bugs, CPU backdoors and consequences on security. J. Comput. Virol. **5**(2), 91–104 (2008)
5. Zhou, L., Makris, Y.: Hardware-based on-line intrusion detection via system call routine fingerprinting. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1546–1551. IEEE, Lausanne (2017)
6. Liu, L., et al.: Jintide®: a hardware security enhanced server CPU with xeon® cores under runtime surveillance by an in-package dynamicall reconfigurable processor. In: 2019 IEEE Hot Chips 31 Symposium (HCS), pp. 1–25. IEEE, Cupertino (2019)
7. Hoque, T., Wang, X., Basak, A., Karam, R., Bhunia, S.: Hardware Trojan attacks in embedded memory. In: 2018 IEEE 36th VLSI Test Symposium (VTS), pp. 1–6. IEEE, San Francisco (2018)
8. Wang, X., Mal-Sarkar, T., Krishna, A., Narasimhan, S., Bhunia, S.: Software exploitable hardware Trojans in embedded processor. In: 2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 55–58. IEEE, Austin (2012)
9. Zhao, Y., Wang, X., Jiang, Y., Mei, Y., Singh, A.-K., Mak, T.: On a new hardware Trojan attack on power budgeting of many core systems. In: 31st IEEE International System-on-Chip Conference (SOCC), pp. 1–6. IEEE, Arlington, VA, USA (2018)
10. Zhou, J., Li, M., Guo, P., Liu, W.: Mitigation of tampering attacks for MR-based thermal sensing in optical NoCs. In: 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 554–559. IEEE, Limassol, Cyprus (2020)
11. Zaraee, N., Zhou, B., Vigil, K., Shahjamali, M., Joshi, A., Selim, Ü.M.: Gate-level validation of integrated circuits with structured-illumination read-out of embedded optical signatures. IEEE Access **8**, 70900–70912 (2020)
12. Chhabra, S., Lata, K.: Key-based Obfuscation using HT-like Trigger Circuit for 128-bit AES Hardware IP Core. In: 34th International System-on-Chip Conference (SOCC), pp. 164–169. IEEE, Las Vegas, NV, USA (2021)
13. Ma, H., et al.: On-chip trust evaluation utilizing TDC-based parameter-adjustable security primitive. IEEE Trans. TCAD **40**(10), 1985–1994 (2021)
14. Lin D, Wu C.: Real-time active tampering detection of surveillance camera and implementation on digital signal processor. In: 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 383–386. IEEE, Piraeus-Athens, Greece (2012)
15. Baba Y, Homma N, Miyamoto A, Aoki T.: Design of tamper-resistant registers for multiple-valued cryptographic processors. In: 40th IEEE International Symposium on Multiple-Valued Logic, pp. 67–72. IEEE, Barcelona, Spain (2010)
16. Yang, J., Zhang, Y., Gao, L.: Fast secure processor for inhibiting software piracy and tampering. In: 36th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 351–360. IEEE, San Diego, CA, USA (2003)

17. Bashiri, M., Miremadi, S.-G., Fazeli, M.: A Checkpointing technique for rollback error recovery in embedded systems. In: 2006 International Conference on Microelectronics, pp. 174–177. IEEE, Dhahran, Saudi Arabia (2006)
18. Xu, M., Zhao, H., Li, J., Zhang, H.: Steady rollback and recovery policy based on integrity measurement. In: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, pp. 834–836. IEEE, Xiamen (2010)
19. Chen, C.-H., Ting, Y, Heh, J.-S.: Low overhead incremental checkpointing and rollback recovery scheme on windows operating system. In: Third International Conference on Knowledge Discovery and Data Mining, pp. 268–271. IEEE, Phuket, Thailand (2010)
20. Tamir, Y., Tremblay, M.: High-performance fault-tolerant VLSI systems using micro rollback. IEEE Trans. Comput. **39**(4), 548–554 (1990)
21. Slegel, T.-J., et al.: IBM's S/390 G5 microprocessor design. IEEE Micro **19**(2), 12–23 (1999)
22. Sorin, D., Martin, M., Hill, M., Wood, D.: SafetyNet: improving the availability of shared memory multiprocessors with global checkpoint/recovery. In: 29th Annual International Symposium on Computer Architecture, pp. 123–134. IEEE, Anchorage, AK, USA (2002)
23. Salehi, M., Khavari, T.-M., Rehman, S., Shafique, M., Ejlali, A., Henkel, J.: Two-state checkpointing for energy-efficient fault tolerance in hard real-time systems. IEEE Trans. VLSI **24**(7), 2426–2437 (2016)
24. Li, T., Ambrose, J., Parameswaran, S..: RECORD: reducing register traffic for checkpointing in embedded processors. In: 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 582–587. IEEE, Dresden, Germany (2016)
25. Do, X., Ha, V., Tran, V., Renault, É.: The technique of locking memory on Linux operating system - application in checkpointing. In: 6th NAFOSTED Conference on Information and Computer Science (NICS), pp. 178–183. IEEE, Hanoi, Vietnam (2019)
26. Wang, X., Zhao, Z., Xu, D., Zhang, Z., Hao, Q., Liu, M.: An M-cache-based security monitoring and fault recovery architecture for embedded processor. IEEE Trans. VLSI **28**(11), 2314–2327 (2020)
27. Chaudhari, A., Park, J., Abraham, J.: A framework for low overhead hardware based runtime control flow error detection and recovery. In: 31st VLSI Test Symposium (VTS), pp. 1–6. IEEE, Berkeley, CA, USA (2013)
28. Huu, N., Robisson, B., Agoyan, M., Drach, N.: Low-cost recovery for the code integrity protection in secure embedded processors. In: 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 99–104. IEEE, San Diego, CA, USA (2011)
29. Gizopoulos, D., et al.: Architectures for online error detection and recovery in multicore processors. In: 2011 Design, Automation & Test in Europe, pp. 1–6. IEEE, Grenoble, France (2011)
30. Kundu, K., Khan, O.: Efficient error-detection and recovery mechanisms for reliability and resiliency of multicores. In: 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), pp. 12–13. IEEE, Kolkata, India (2016)
31. Zhou, W.-T., Li, L., Yuan, S.-W.: China Patent, vol. 202210262087, pp. 4 (2022)
32. PULpino Datasheet. https://pulp-platform.org/docs/pulpino_datasheet.pdf
33. PULpino Project. https://github.com/pulpplatform/pulpino
34. Yuan, S.-W., Li, L., He, Y.-H., Zhou, W.-T., Li, J.: Real-time detection of hardware trojan attacks on general-Purpose Registers in a RISC-V processor. IEICE Electron. Express **18**(10), 1–3 (2021)