Heliyon 9 (2023) e17085

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

A cycle-level recovery method for embedded processor against HT tamper

Wanting Zhou [a,*], Kuo-Hui Ye [b], Shiwei Yuan [a], Lei Li [a]

[a] Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, No. 2006, Xi Yuan Ave., West High-Tech Zone, Chengdu, 611731, Sichuan, China

[b] Department of Information Management, National Dong Hwa University, No. 1, Sec. 2, Da Hsueh Rd., Shoufeng, Hualien, 97001, Taiwan (Province of China)

ARTICLE INFO

Keywords:
Hardware security
GPRs
Fault recovery
Embedded processor
Internet of Things (IoT)

ABSTRACT

As the core of Internet of Things (IoT), embedded processors are being used more and more extensive. However, embedded processors face various hardware security issues such as hardware trojans (HT) and code tamper attacks. In this paper, a cycle-level recovery method for embedded processor against HT tamper is proposed, which builds two hardware-implementation units, a General-Purpose Register (GPRs) backup unit and a PC rollback unit. Once a HT tamper is detected, the two units will carry out fast recovery through rolling back to the exact PC address corresponding to the wrong instruction and resuming the instruction execution. An open RISC-V core of PULPino is adopted for recovery mechanism verification, the experimental results and hardware costs show that the proposed method could guarantee the processor restore from abnormal state in real time with a reasonable hardware overhead.

1. Introduction

In recent years, the Internet of Things (IoT) has been widely used in various electronic product market ranging from industrial products to national strategic military products. At the meantime, the security of the IoT has attracted the attention of research. As the embedded core of IoT, it is vulnerable to hardware Trojan [1–13] and code tamper attacks [14–16], which could have serious consequences, including functional changes, denial of service (DoS), information leakage, and even system crashes. Thus, the security issue of embedded processor becomes extremely prominent, the recovery mechanism after attack detection is an important scheme to guarantee the security of embedded processor.

At present, research on processor-oriented security technology has achieved certain results, which can effectively solve the security problems caused by processor hardware vulnerabilities. Existing approaches for processor fault recovery mechanism could be generally categorized into two kinds, 1) checkpoint backup and rolling back [17–26], 2) combined approach to attack detection followed by fault repairing [27–30]. The first kind approaches take advantage of checkpoint file for recording the state of executing program and performing restoring once a failure occurs. The basic idea of second kind approaches is to build basic blocks (BBs) as minimum monitoring and recovery architecture. Although both approaches could guaranty the security of the embedded processors, they both suffer from lower real-time performance and higher resource requirements.

* Corresponding author.
E-mail address: zhouwt@uestc.edu.cn (W. Zhou).

https://doi.org/10.1016/j.heliyon.2023.e17085

Available online 13 June 2023

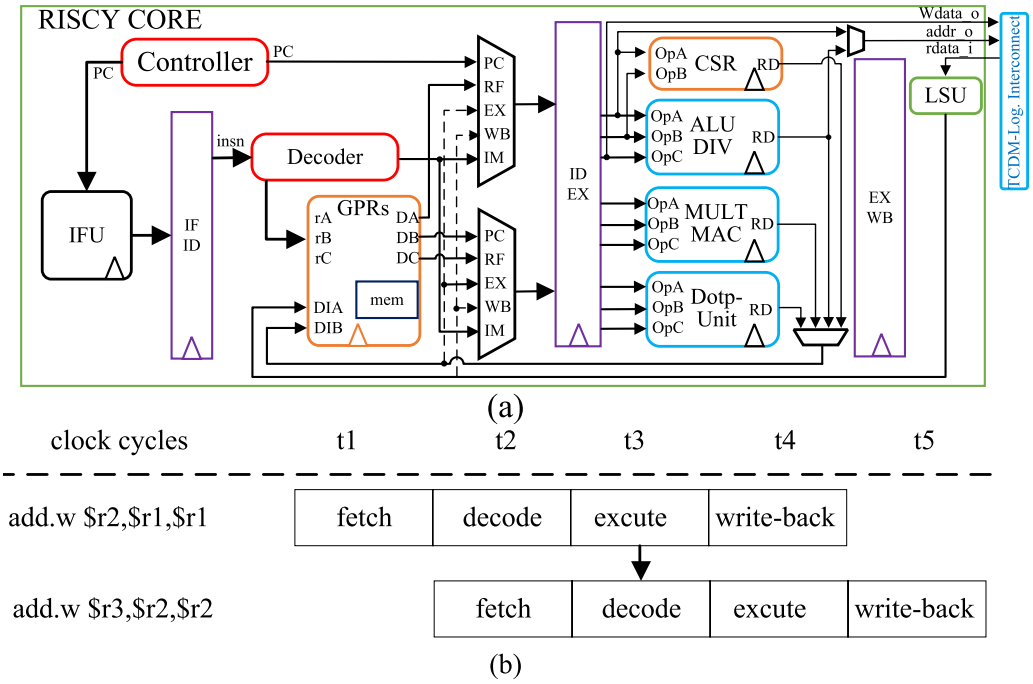

Fig. 1. The function of GPRs information in processor pipeline.

This paper aims to solve the security issues of embedded processor attacked by malicious HT tamper. Research on security technology for processor based on the idea of active defense have been carried on, including cycle-level detection and recovery mechanism. Moreover, hardware implementation has been also investigated. This entails the following.

1) An embedded reference model is built for monitoring whether abnormal state occurs. This part has been presented in our previous work [31]. The instructions worked in processor have certain life time. During the life time of the instruction, the status of the processor GPRs is unique and only changes at a fixed moment in the life cycle of the instruction. Then, any exception status induced by HT tamper on GPRs can be real-time detected in comparison the value of the designed reference model to the value of the execution instruction written to the GPRs.
2) A cycle-level recovery mechanism is proposed to accomplish the restoring the right value which should write to GPRs, as well as suspend all pipelines of the processor and complete pipeline flushing, empty the prefetch instructions in the prefetch FIFO, then use the PC rollback method to replace the value with PC address corresponding to the exact moment of an attack occurs, and finally restore the processor pipeline to complete the proposed fast recovery mechanism.

The rest of the paper is organized as follows. The proposed cycle-level recovery method and hardware implementation is given in section 2. Attack model and some experiments are designed to verify the proposed method. Further, experimental results, hardware overhead, and limitations are also given in section 3. Finally, we give some conclusions in section 4.

2. The proposed cycle-level recovery method and hardware implementation

For in-order embedded processors, instructions are statically scheduled. In other words, instructions are fetched, executed and completed in compiler generated order. Consequently, instructions executed in embedded processor may have data dependencies. An example is given for analyzing the data dependencies scenario in Fig. 1.

The first instruction performs the add operation and stores the result in the R1 register at time t3. The subsequent instruction also performs the add operation, uses the value of the R1 register in the decode stage and completes the output at time t4. In this case, it is obviously that the correct instruction execution depends on the GPRs value written by the previous instruction in addition to the instruction itself.

For in-order embedded processors, if a rollback operation could be restored at the exact time point when the attack occurred, while a GPRs backup consists of storing the correct value of the previous instruction, then the cycle-level recovery mechanism could be achieved by re-executing the instruction which was tampered. Consequently, the proposed recovery scheme consists three function units, and the specific functions of which are described as follows.

GSRTM: In our previous work [31], GPR-State Real-Time Detection Module (GSRTM) was proposed for monitoring the state of GPRs in real time. An embedded self-reference model based on processor instruction set and micro-architecture to realize continuous and real-time detection of abnormal state of GPRs. Once an attack detected, the detection result would be trigged to start the

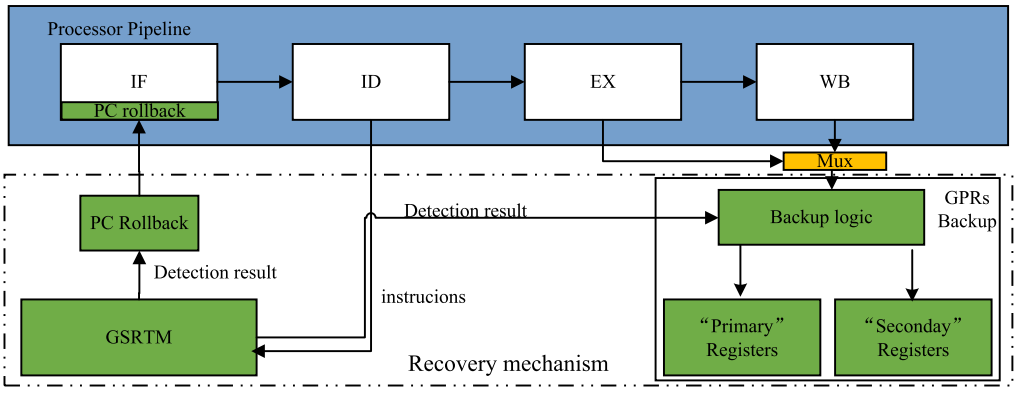

Fig. 2. The proposed approach based on GPRs backup and PC rollback techniques.

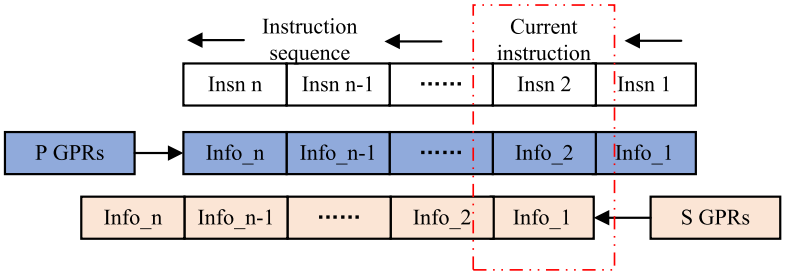

Fig. 3. The information relationship of instruction stream between Primary and Secondary GPRs.

subsequent recovery operations. Meanwhile, the attacked instruction and its PC address are also provided to PC rollback unit to perform the recovery operations.

GPRs Backup Unit: Use latch to back up the GPRs information. Make the backup information always corresponds to the life time of the previous executed instruction.

PC Rollback Unit: First suspend and flush the processor pipeline, then replace the current PC in the instruction fetch stage with the exact PC value when the attack occurred, then switch the GPRs used by the processor to the backup GPRs, finally restore the processor pipeline so that the instruction rolls back to the time when the fault occurred and re-execute.

These three functional units act on different stages of the pipeline and work together to complete the fast recovery. RI5CY, an open four-stage RISC-V core of PULPino [32,33], is adopted as embedded processor core. And the hardware implementation implanted into RI5CY is shown in Fig. 2.

2.1. The function and hardware implementation of GPRs backup unit

It can be known from the above analysis that the designed GPRs backup unit should have two functions. It should not only bypass the result of the current instruction in EX or WB stage to GPRs without extra delay in normal work situation, but also store the GPRs information corresponding to the previous intrusion. Therefore, two groups of register file are needed. The so called "Primary" registers are used in normal work without extra delay. In the meantime, the so called "Secondary" registers are working for store information corresponding to the previous instruction. The information relationship between "Primary" registers and "Secondary" registers during instruction life time (the time of an instruction between ID stage to EX or WB stage) is given in Fig. 3.

Besides, a label is designed within the register group to indicate the current usage of the register group. If tag = 1, it works as "Primary" register group, else if tag = 0, it works as "Secondary" register for backup. When the "Primary" registers are subject to malicious tampering attack, the labels of the two register groups are switched. The register group used by the processor is changed from the "Primary" registers group to the "Secondary" registers group (tag changes from 1 to 0). At the same time, the backup register group is changed from the "Secondary" register group to the "Primary" register group (label changes from 0 to 1), and when another tampering attack is detected, the labels of the two register groups are switched again. Consequently, the hardware-implementation of the proposed unit is given in Fig. 4.

The input of backup unit comes from the execute or write-back stage, named as write channel, which is used for updating corresponding GPRs. As mentioned above, the "Primary" registers work just as a bypass channel without any processing. The inputs of "Secondary" registers are same signals but with three clock cycles delay, then perform a latch operation according to whether the current instruction life time is over or not, indicated by signal insn_life_over. When the current instruction life time ends, the latched write channel signal is released and passed to the "Secondary" registers for backup. Further, the detection result (signal strong_warning) to select whether "Primary" registers output or "Secondary" registers output.

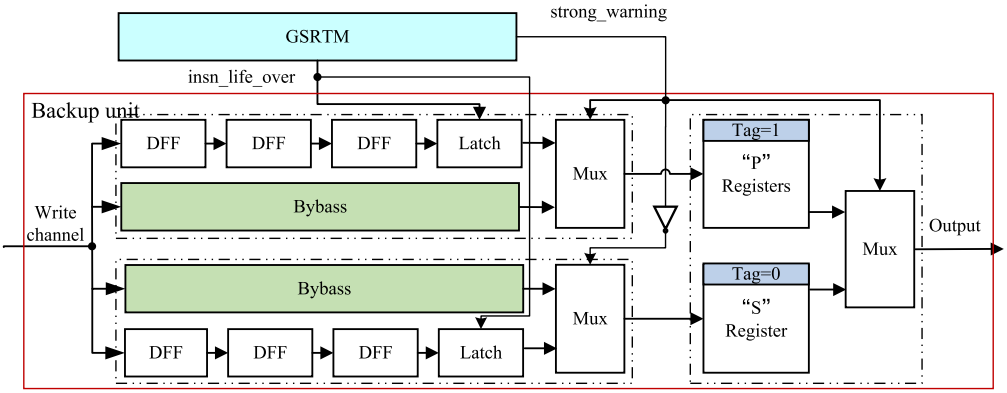

Fig. 4. The implementation of GPRs backup unit.

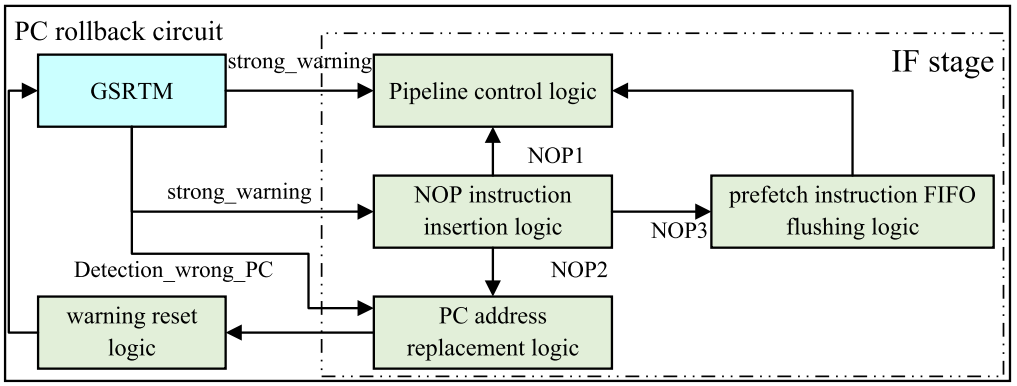

Fig. 5. The implementation of PC rollback unit.

2.2. The function and hardware implementation of PC rollback unit

The PC rollback unit works similarly to program rollback. The difference is that the PC rollback unit rolls back to the exact PC address corresponding to the wrong instruction, while the program rollback unit rolls back to the latest checkpoint. Once a fault is detected, the fast recovery scheme can be realized through re-executing the wrong instruction utilizing the PC address and GPRs information storage in "Secondary" registers.

The reason why PC rollback can be implemented is that the information of GPRs corresponding to the preceding instruction is not related to the executed result of the following instruction, but the execution of the following instruction depends on the information of GPRs corresponding to the preceding instruction. When an abnormal state of the processor GPRs is detected, the information of the "secondary" register backup is related to the previous instruction corresponding to the instruction at the time of the fault detected, that is, to the previous instruction.

The memory access instruction *STORE*, used to store data from GPRs into memory, need to be processed specially. It is only necessary to immediately block the memory access permission of the *STORE* instruction when a fault occurs, and the state before the fault can be maintained. Therefore, it is only need to execute the instruction again, which corresponds to the fault time based on the backup information in "Secondary" registers, and then quickly restore to the normal state.

Once an attack is detected, the indicated signal strong_warning would trigger recovery scheme. The PC rollback unit suspends all pipelines of the processor and completes pipeline flushing, empties the prefetch instructions in the prefetch FIFO, then uses the PC rollback method to replace the current PC with PC address corresponding to the exact moment of an attack occurs, and finally resets warning signal and restores the processor pipeline. The hardware implementation of PC rollback unit is shown in Fig. 5. And there are five steps for perform recovery scheme.

Step 1: When the indicated signal strong_warning is valid, the pipeline control logic would suspend all of the pipeline stages for prevention the fault propagation spread and efficient execution of subsequent step.

Step 2: Insert three No Operations (NOPs) into the instruction transfer path between IF and ID stages, and resume the pipeline of the ID, EX and WB stages simultaneously. This step clears the invalid state by flushing pipeline and provides process time for subsequent operations.

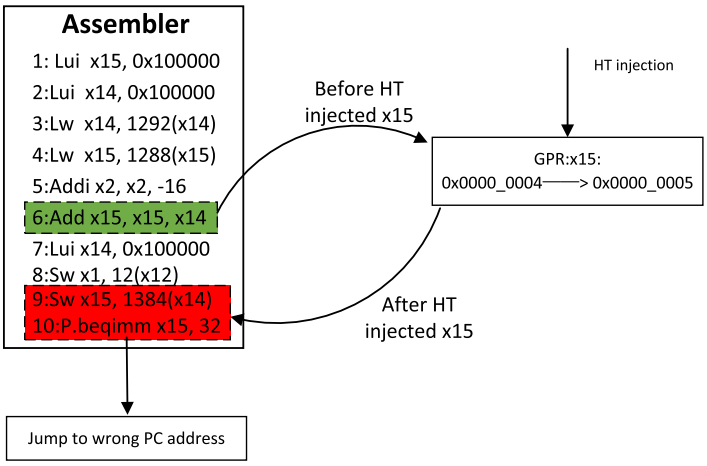

Fig. 6. An example of tamper attack on x15 of GPRs.

Step 3: When the second NOP is inserted, the current PC address output to the ITCM in the IF stage is replaced with the PC address corresponding to the exact moment of an attack occurs.

Step 4: When the third NOP is inserted, the prefetch FIFO located in IF stage is needs to be emptied. This is because the FIFO stores the previously unexecuted instructions which are the subsequence instructions of the corresponding instruction to the exact moment of an attack occurs. And then the pipeline of the processor IF stage is restored.

Step 5: When the instruction to retrieve the PC after rollback is detected in the ID stage, it indicates that the PC rollback recovery has been completed, and reset the signal strong_warning generated by the GSRTM unit.

3. Experimental results and discussion

In order to analyze and verify the security and effectiveness of the processor security technology designed in this paper, PULPino, an open-source single-core microcontroller system, is used as the experimental verification platform. Further, a HT maliciously tampered with x15 of the GPRs, is used to explain the tamper flow.

1. Attack model and tamper flow

An attacker can launch a hardware Trojan attack on the processor GPRs to tamper with the GPRs value, thereby achieving the purpose of modifying the processor's operating function or causing the processor to crash. The tamper flow is shown in Fig. 6.

The function of the assembler is as follows. Firstly, write the data address 0x00100000 into the registers x14 and x15 through the *lui* (Load Upper Immediate) instruction, which used to construct global address or large constant. Secondly, read the data from the data addresses of x14+1292 and x15+1288 through the *lw* (Load Word) instruction, which loads data from data memory through a specified address. Then use the *add* instruction to complete the addition operation of x14 and x15, and write the value 4 into x15. Finally, judge whether the value of x15 is equal to 4, if it is equal to 4, jump to function 1, otherwise jump to function 2. After all, the attacker has completed the change of the intended function of the program by tampering with the register x15.

2. Detection results and discussion

Six programs with different functions are implanted for test. In order to obtain more experimental test data, the activation conditions of the hardware Trojans is set very simply. Pseudo-randomly generates several random numbers, when the number of time cycles during the execution of the processor matches the random number, the hardware Trojan is triggered. And the test results are shown in Table 1, including number of HT implanted, detection number and recovery time.

To explain more clear, take add.c as an example. Fig. 7 shows the relevant information of activated hardware Trojans and detection results. Position_HT is the moment when the hardware Trojan is activated while Register_number_HT denotes the target register for malicious tampering by the hardware Trojan, wrong_Cycles represents the exact moment of an attack occurs, Wrong_PC is the PC address corresponding to the exact moment of an attack occurs, and Wrong_Instr denotes the instruction attacked by HT tamper. As can be seen from Table 1, all six programs with 207 HTs had been detected with the latency of 2 clocks.

In order to evaluate the effectiveness and performance of the proposed cycle-level recovery method, same hardware Trojan are implemented both in the original RI5CY (baseline processor) and security processor with recovery mechanism. Then run the C program add.c on the verification experiment platform of the baseline processor and the security processor respectively, and its code and its running results on the two platforms are shown in Fig. 8. When the hardware Trojan is activated, the baseline processor's jump value is tampered with 5 (the expected value should be 4), and the function func2 is executed against the established program

Table 1
Detection and recovery results.

Codes	HT number	Detected number	Recovery Time
add.c	5	5	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

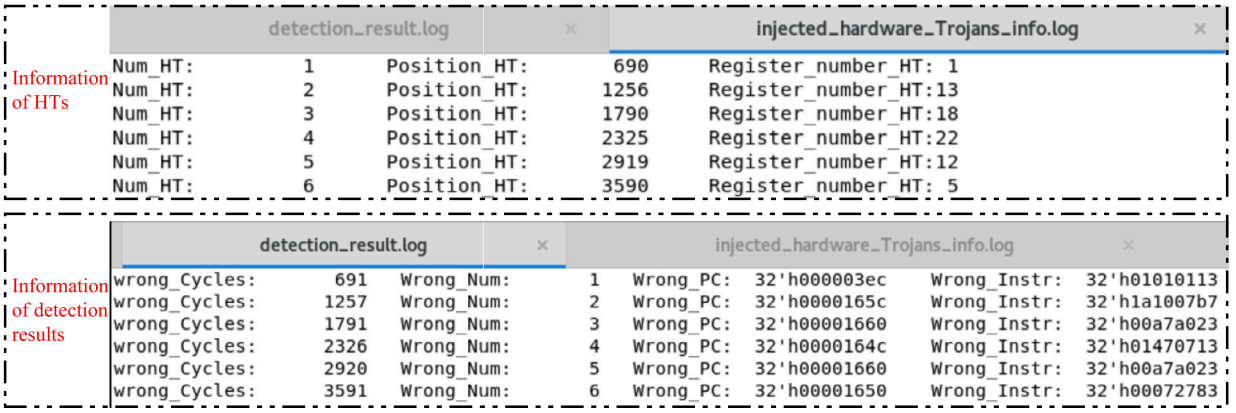

Fig. 7. The relevant information of active HTs and detection results.

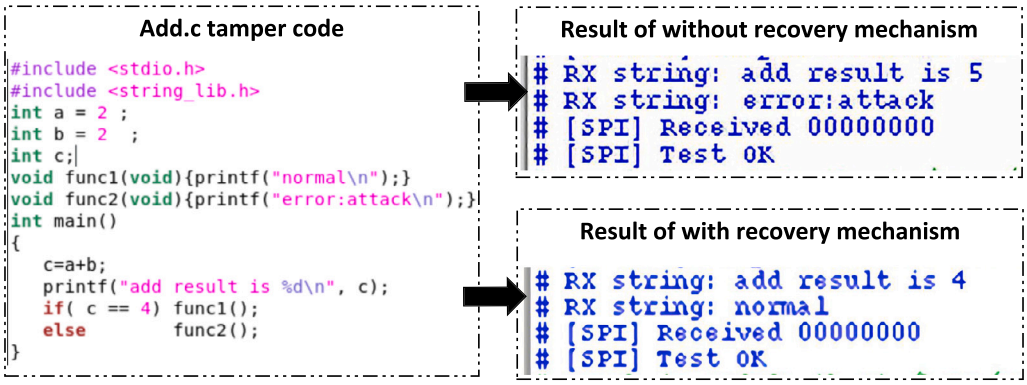

Fig. 8. add.c program implement results on baseline and security processors.

control flow. Because the security processor has a detection and fast recovery mechanism for GPRs, even if the hardware Trojan is activated, its operation result is still normal.

3. Recovery simulation results and analysis

A hardware Trojan is implanted in the processor GPRs, and its function is consistent with the threat tamper example as shown in Fig. 6. More detailed and further analysis to obtain the attack time of the hardware Trojan as shown in Fig. 9. The hardware Trojan has maliciously tampered with the x15 of GPRs, making the operation result of add.c change from 4 to 5, resulting in the program function is running incorrectly. According to the operating waveform of the security processor detection mechanism and fast recovery mechanism, as shown in Fig. 9, it can be known that the proposed detection and fast recovery technology for GPRs designed and implemented in Section 2 can be used when GPRs are subjected 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 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.

Table 2
Hardware overhead of the proposed method.

	LUT	Flip-Flops
Baseline processor	6265	1317
Processor with recovery	6377 (+1.7%)	2361(+79.2%)

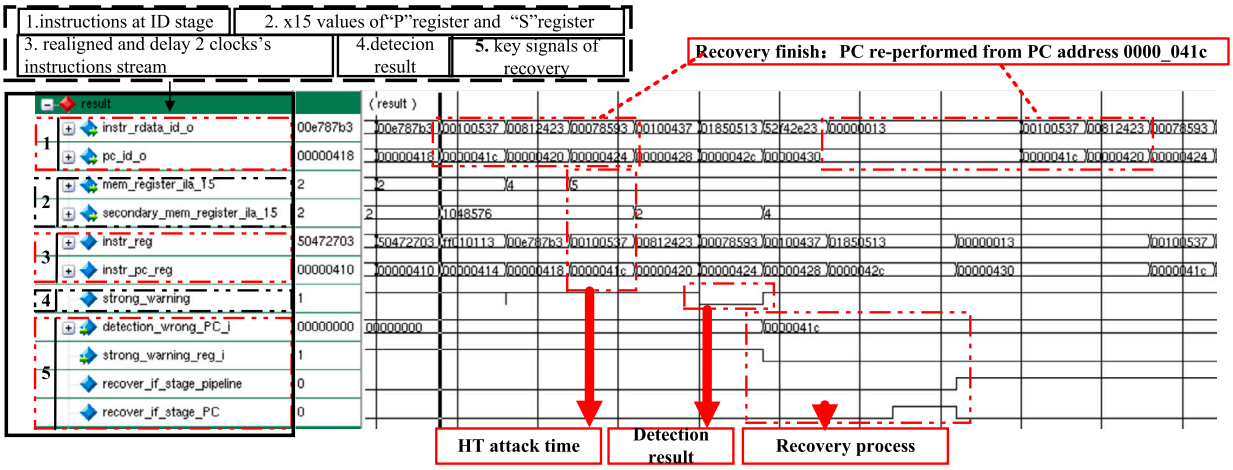

Fig. 9. Simulation wave of HT activated, detection and recovery mechanism.

4. Hardware overhead

The baseline processor and the proposed security processor are implemented on same Xilinx FPGA platform, respectively. And the synthesis results of hardware costs are given in Table 2 above. It shows that the increase amounts of combination logic and sequential logic, presented using number of LUT and number of Flip-Flops (FFs) is 1.7% and 79.2%, respectively. The former one is reasonable while the latter one is rather high, which resulted the register file is duplicated in our proposed mechanism. Compared with hardware implementation results in [28], which aimed at developed a low-cost recovery for embedded processors, the overhead of combination logic and sequential logic, are 18.8% and 68.86%. And the proposed method could deduce a similar result.

5. Limitations

The proposed recovery mechanism is now only effective in in-order microarchitectures. The microarchitectures are different between in-order processor and out-of-order processor. Although, the architecture of out-of-order processor may still decode them and retire them in the actual order of appearance in the program, ILP (Inductive Logic Programming) or Instruction Level Parallelism should be studied carefully for out-of-order execution. Future work will further to consider an effective solution for out-of-order embedded processor.

4. Conclusion

In this paper, a real-time recovery method after real-time detection is presented to protect GPRs against the HT attacks. We introduce a GPRs backup unit and a PC rollback unit to perform fast recovery. Furthermore, the hardware implementation is also investigated and incorporated into RI5CY, a four-stage pipeline core of RISC-V. Experiments are designed to verify the proposed method. And the experimental results demonstrate that the proposed method can effectively guarantee that processor restored from abnormal state with the latency of 7 clock cycles, which introduces a reasonable hardware overhead. Future work will further to consider an effective solution for out-of-order micro architectures.

CRediT authorship contribution statement

Wanting Zhou: Conceived and designed the experiments; Wrote the paper. **Kuo-Hui Ye:** Contributed reagents, materials, analysis tools or data. **Shiwei Yuan:** Performed the experiments; Analyzed and interpreted the data. **Lei Li:** Conceived and designed the experiments.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

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] S. Bhunia, M.S. Hsiao, M. Banga, S. Narasimhan, Hardware trojan attacks: threat analysis and countermeasures, Proc. IEEE 102 (8) (2014) 1229–1247, https://doi.org/10.1109/JPROC.2014.2334493.
[2] M.-H. Kuo, C.-M. Hu, K.-J. Lee, Time-related hardware trojan attacks on processor cores, in: 2019 IEEE International Test Conference in Asia (ITC-Asia), 2019, pp. 43–48.
[3] P. Okane, S. Sezer, K. McLaughlin, E.G. Im, Malware detection: program run length against detection rate, IET Softw. 8 (1) (2014) 42–51, https://doi.org/10.1049/iet-sen.2013.0020.
[4] L. Duflot, Cpu bugs, cpu backdoors and consequences on security, J. Comput. Virol. 5 (2) (2009) 91–104, https://doi.org/10.1007/s11416-008-0109-x.
[5] L. Zhou, Y. Makris, Hardware-based on-line intrusion detection via system call routine fingerprinting, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, 2017, pp. 1546–1551.
[6] L. Liu, A. Luo, G. Li, J. Zhu, Y. Wang, G. Shan, J. Pan, S. Yin, S. Wei, Jintide®: a hardware security enhanced server cpu with xeon® cores under runtime surveillance by an in-package dynamically reconfigurable processor, in: 2019 IEEE Hot Chips 31 Symposium (HCS), 2019, pp. 1–25.
[7] T. Hoque, X. Wang, A. Basak, R. Karam, S. Bhunia, Hardware trojan attacks in embedded memory, in: 2018 IEEE 36th VLSI Test Symposium (VTS), 2018, pp. 1–6.
[8] X. Wang, T. Mal-Sarkar, A. Krishna, S. Narasimhan, S. Bhunia, Software exploitable hardware trojans in embedded processor, in: 2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012, pp. 55–58.
[9] Y. Zhao, X. Wang, Y. Jiang, Y. Mei, A.K. Singh, T. Mak, On a new hardware trojan attack on power budgeting of many core systems, in: 2018 31st IEEE International System-on-Chip Conference (SOCC), 2018, pp. 1–6.
[10] J. Zhou, M. Li, P. Guo, W. Liu, Mitigation of tampering attacks for mr-based thermal sensing in optical nocs, in: 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2020, pp. 554–559.
[11] N. Zaraee, B. Zhou, K. Vigil, M.M. Shahjamali, A. Joshi, M. Selim Ünlü, Gate-level validation of integrated circuits with structured-illumination read-out of embedded optical signatures, IEEE Access 8 (2020) 70900–70912, https://doi.org/10.1109/ACCESS.2020.2987088.
[12] S. Chhabra, K. Lata, Key-based obfuscation using ht-like trigger circuit for 128-bit aes hardware ip core, in: 2021 IEEE 34th International System-on-Chip Conference (SOCC), 2021, pp. 164–169.
[13] H. Ma, J. He, Y. Liu, J. Kuai, H. Li, L. Liu, Y. Zhao, On-chip trust evaluation utilizing tdc-based parameter-adjustable security primitive, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (10) (2021) 1985–1994, https://doi.org/10.1109/TCAD.2020.3035346.
[14] D.-T. Lin, C.-H. Wu, 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, 2012, pp. 383–386.
[15] Y. Baba, N. Homma, A. Miyamoto, T. Aoki, Design of tamper-resistant registers for multiple-valued cryptographic processors, in: 2010 40th IEEE International Symposium on Multiple-Valued Logic, 2010, pp. 67–72.
[16] J. Yang, Y. Zhang, L. Gao, Fast secure processor for inhibiting software piracy and tampering, in: Proceedings, 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003, MICRO-36, 2003, pp. 351–360.
[17] M. Bashiri, S.G. Miremadi, M. Fazeli, A checkpointing technique for rollback error recovery in embedded systems, in: 2006 International Conference on Microelectronics, 2006, pp. 174–177.
[18] M. Xu, H. Zhao, J. Li, H. Zhang, Steady rollback and recovery policy based on integrity measurement, in: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 2, 2010, pp. 834–836.
[19] C.-H. Chen, Y. Ting, J.-S. Heh, Low overhead incremental checkpointing and rollback recovery scheme on windows operating system, in: 2010 Third International Conference on Knowledge Discovery and Data Mining, 2010, pp. 268–271.
[20] Y. Tamir, M. Tremblay, High-performance fault-tolerant vlsi systems using micro rollback, IEEE Trans. Comput. 39 (4) (1990) 548–554, https://doi.org/10.1109/12.54848.
[21] T. Slegel, R. Averill, M. Check, B. Giamei, B. Krumm, C. Krygowski, W. Li, J. Liptay, J. MacDougall, T. McPherson, J. Navarro, E. Schwarz, K. Shum, C. Webb, Ibm's s/390 g5 microprocessor design, IEEE MICRO 19 (2) (1999) 12–23, https://doi.org/10.1109/40.755464.
[22] D. Sorin, M. Martin, M. Hill, D. Wood, Safetynet: improving the availability of shared memory multiprocessors with global checkpoint/recovery, in: Proceedings 29th Annual International Symposium on Computer Architecture, 2002, pp. 123–134.
[23] M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali, J. Henkel, Two-state checkpointing for energy-efficient fault tolerance in hard real-time systems, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 24 (7) (2016) 2426–2437, https://doi.org/10.1109/TVLSI.2015.2512839.
[24] T. Li, J.A. Ambrose, S. Parameswaran, Record: reducing register traffic for checkpointing in embedded processors, in: 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016, pp. 582–587.
[25] X.H. Do, V.H. Ha, V.L. Tran, E. Renault, The technique of locking memory on Linux operating system - application in checkpointing, in: 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), 2019, pp. 178–183.
[26] X. Wang, Z. Zhao, D. Xu, Z. Zhang, Q. Hao, M. Liu, An m-cache-based security monitoring and fault recovery architecture for embedded processor, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 28 (11) (2020) 2314–2327, https://doi.org/10.1109/TVLSI.2020.3021533.
[27] A. Chaudhari, J. Park, J. Abraham, A framework for low overhead hardware based runtime control flow error detection and recovery, in: 2013 IEEE 31st VLSI Test Symposium (VTS), 2013, pp. 1–6.
[28] N.M. Huu, B. Robisson, M. Agoyan, N. Drach, Low-cost recovery for the code integrity protection in secure embedded processors, in: 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, 2011, pp. 99–104.

[29] D. Gizopoulos, M. Psarakis, S.V. Adve, P. Ramachandran, S.K.S. Hari, D. Sorin, A. Meixner, A. Biswas, X. Vera, Architectures for online error detection and recovery in multicore processors, in: 2011 Design, Automation & Test in Europe, 2011, pp. 1–6.
[30] S. Kundu, O. Khan, Efficient error-detection and recovery mechanisms for reliability and resiliency of multicores, in: 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016, pp. 12–13.
[31] S. Yuan, L. Li, J. Yang, Y. He, W. Zhou, J. Li, Real-time detection of hardware trojan attacks on general-purpose registers in a risc-v processor, IEICE Electron. Express 18 (10) (2021) 20210098, https://doi.org/10.1587/elex.18.20210098.
[32] PULPino Datasheet, https://pulp-platform.org/docs/pulpino_datasheet.pdf.
[33] PULPino Project, https://github.com/pulp-platform/pulpino.