國立清華大學電機研究所

Trojan Horse Detection for RISC-V Cores Using Cross-Auditing

Speaker : 黃祥成 Siang-Cheng Huang
Advisor : 黃錫瑜 Shi-Yu Huang

IC-Design Exploration Lab
Department of Electrical Engineering
National Tsinghua University, Hsinchu, Taiwan

Outline

- Introduction
- Related Work
- Preliminaries
- Proposed Methodology
- Experiment Result
- Conclusion

Outline

- Introduction
- Related Work
- Preliminaries
- Proposed Methodology
- Experiment Result
- Conclusion

Motivation

- **Globalization of IC design and manufacturing**
  - To lower the R&D Costs and Time to Market
  - Integration of 3^{rd}-party IP is essential in SOC.

- **3^{rd}-party IP is provided by 3^{rd}-party vendor**
  - Not authorized to access the internal architecture
  - Inadvertently or deliberately implanting malicious circuit (Hardware Trojan)
  - A maliciously hidden Trojan when activated could cause the system's malfunction or the leaking of confidential information.

Motivation

- Verifying that every IP is Trojan-free during the design stage is essential.

- Our goal is to enable designers to identify potential Trojans within 3PIP in a non-invasive manner during the pre-silicon phase, under black box conditions, to ensure hardware root-of-trust/trustworthiness.

Outline

- Introduction
- Related Work
  - Software-Based
  - Hardware-Based
- Preliminaries
- Proposed Methodology
- Experiment Result
- Conclusion

Summarize Related Works

Criterion	Software-Based	Hardware-Based
Test Methods	Unique Program Execution Checking (UPEC) [8], Information Flow Tracking [9], BMC + ATPG [10], ABV[11]-[13]	Microprocessor Protection [14], Memory Protection [15]
Limitations	1. Need to pre-defined the security properties. Once the additional vulnerabilities are beyond those properties the method can't find it. 2. Low scalability. Need to target at specific design/HT to establish properties.	1. Disrupt the original framework which leads to additional overhead, such as impacting factors like area, timing, and more. 2. Low scalability. Need to target at specific design/HT to establish properties.

Merits of our method

- More automatic and user-friendly considering that we do not have to manually specify the Trojan models explicitly.
- It is non-invasive and thus does not need to modify the source code of the target IP.
- It covers all three major types of Trojans as reported in Trust-hub [16] – functionality-changing, information-stealing, and denial-of-service.
- It can detect a Trojan even if the Trojan does not attempt to breach the protected memory area as described in [15].

[15]H. Chi, K. Lee, and T. Jao, "Lightweight Hardware-Based Memory Protection Mechanism on IoT Processors", *Proc. of IEEE Asian Test Symp.*, pp. 13-18, 2021

[16]M. Tehranipoor, R. Karri, F. Koushanfar, and M. Potkonjak, "Trusthub," http:// trust-hub.org.

Outline

- Introduction
- Related Work
- **Preliminaries**
- Proposed Methodology
- Experiment Result
- Conclusion

Preliminaries

- **A Hardware Trojan Horses (HTH) [6][7] is often characterized by the following 2 features:**

(1) What are the activating mechanisms for the Trojans?

(2) After a Trojan is activated, how does it affect the functionality?

[6]B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, “Benchmarking of Hardware Trojans and Maliciously Affected Circuits”, Journal of Hardware and Systems Security (HaSS), pp. 85-102, April 2017.

[7]S. Bhunia, M. S. Hsiao, M. Banga and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," in Proceedings of the IEEE, vol. 102, no. 8, pp. 1229-1247, Aug. 2014.

Preliminaries

- **Threat Model**
  - **Activating Mechanisms**
    - **Externally Direct User Input**
    - **Internally Conditionally**
    - **Internally Time-based**
  - **Trojan Effect**
    - **Denial Signal Transmission**
    - **Change of Functionality**
    - **Leakage of Information**

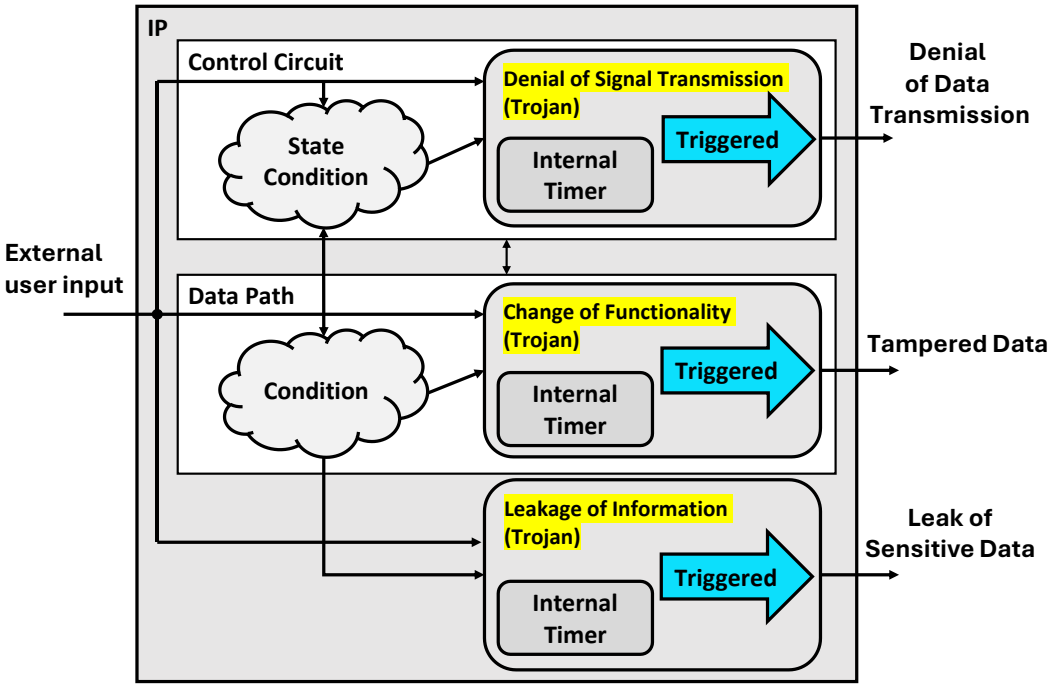


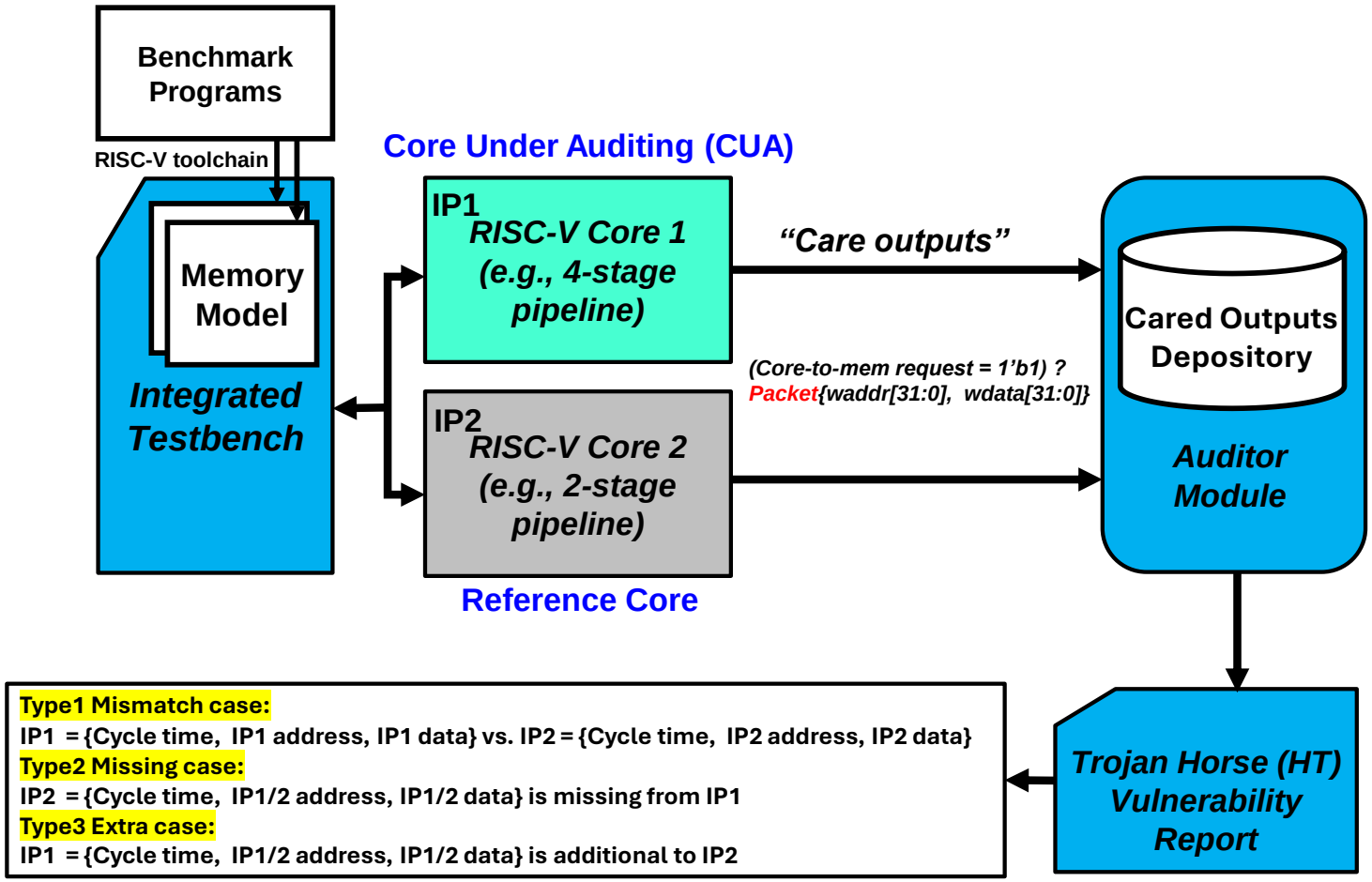
Fig. 3-1. Illustration of a unified threat model of various Trojans.

[16]M. Tehranipoor, R. Karri, F. Koushanfar, and M. Potkonjak, “Trusthub,” http:// trust-hub.org.

Outline

- Introduction
- Related Work
- Preliminaries
- **Proposed Methodology**
- Experiment Result
- Conclusion

Framework Overview

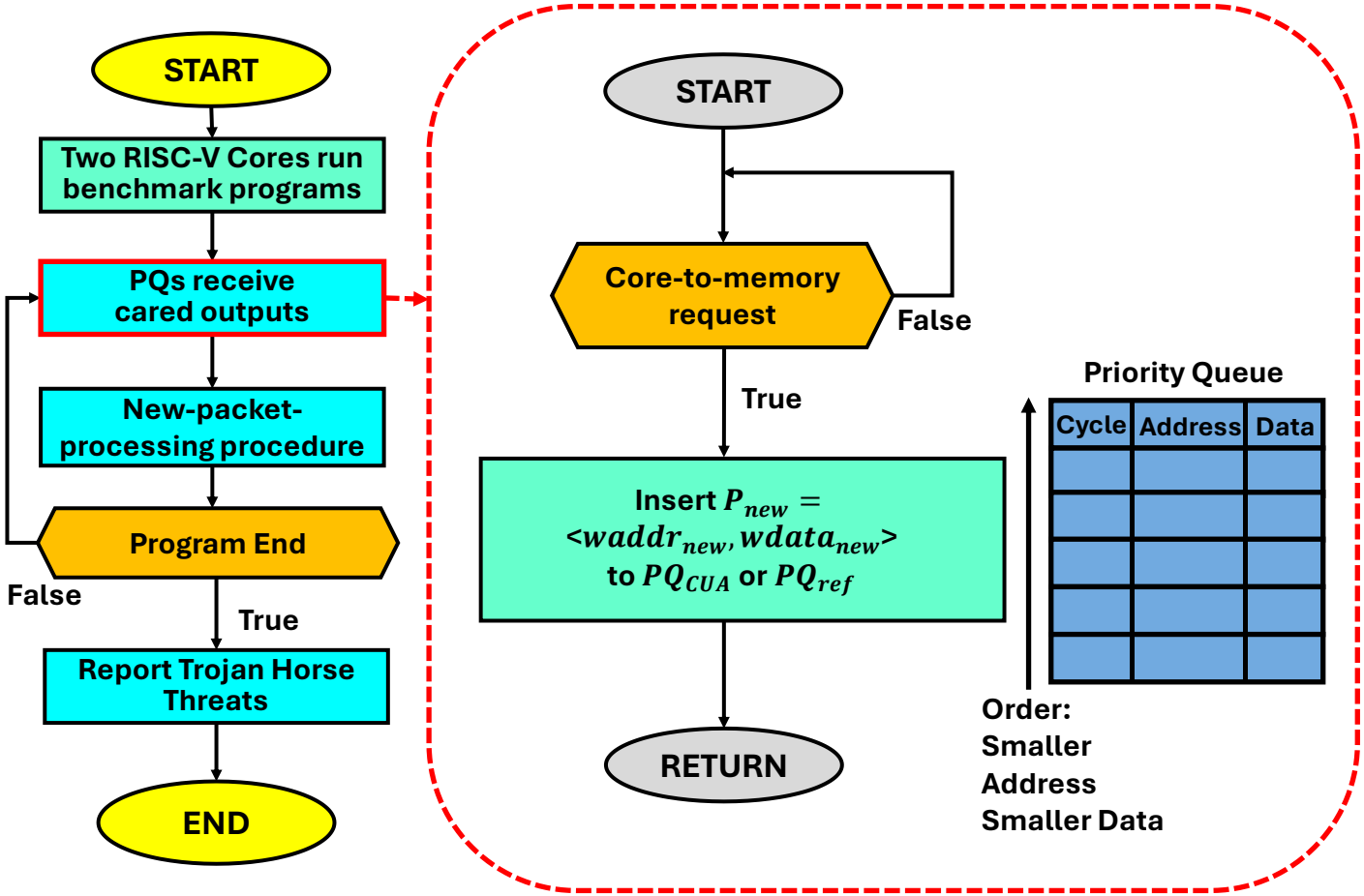


Criterion of a Trojan

- **A CUA is considered Trojan-free if it satisfies the following two conditions:**
  - **(1) The CUA produces all expected outputs correctly.**
  - **(2) The CUA does not produce any additional output.**
- **Two conditions underpin the 3 types of our HT vulnerability report:**
  1) **Mismatch case: Such a case occurs when two cores produce two packets with the same write-address, but different write-data.**
  2) **Missing case: Such a case occurs when we cannot find a "corresponding packet" produced by the CUA for a packet produced by the reference core.**
  3) **Extra case: Such a case occurs when CUA produces a packet that does not have a corresponding one among the packets produced by the reference core.**

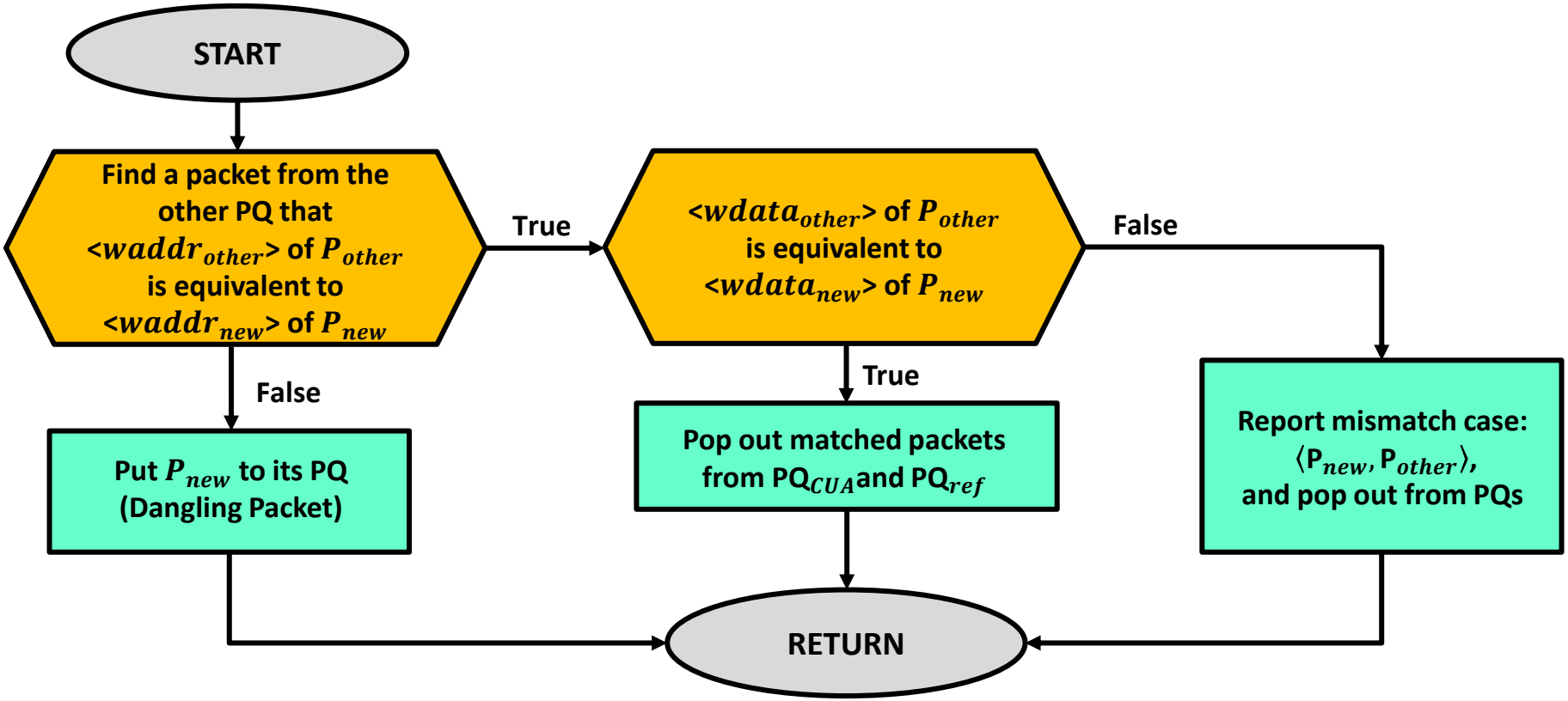
Depository Processing for Efficient Cross-Auditing

- PQs receive cared outputs



Depository Processing for Efficient Cross-Auditing

- New-packet-processing procedure



Depository Processing for Efficient Cross-Auditing

- **Time complexity**
  - **$O(M \cdot \log_2 n)$ where n represents the number of maximum packets stored in any of the two priority queues.**

    M represents packets throughout the entire functional simulation process.
- **Benefit of Dynamic PQ management compared to post-processing PQs**

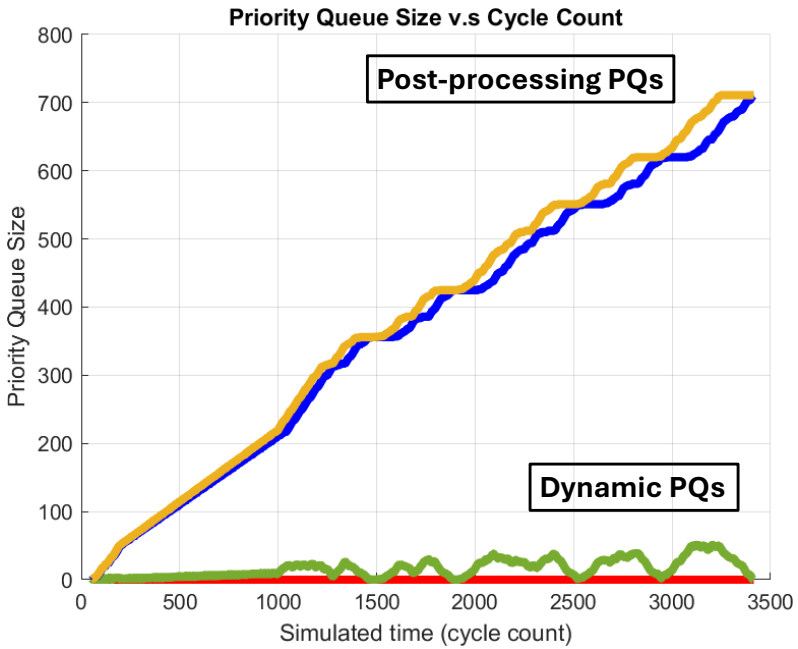


Fig. 3. The priority queue size using posts-processing PQs and dynamic PQs.

Outline

- Introduction
- Related Work
- Preliminaries
- Proposed Methodology
- **Experiment Result**
- Conclusion

Basic Component

- RISC-V Cores
  - A CUA and a reference core (4-stage and 2-stage pipeline respectively)
- Benchmark Program
  - Functional
    - Matrix Multiplication
    - Dhrystone 2.1
    - ISR sample
  - Structural
    - RISC-V Arch
    - RISC-V Compliance
    - RISC-V ISA

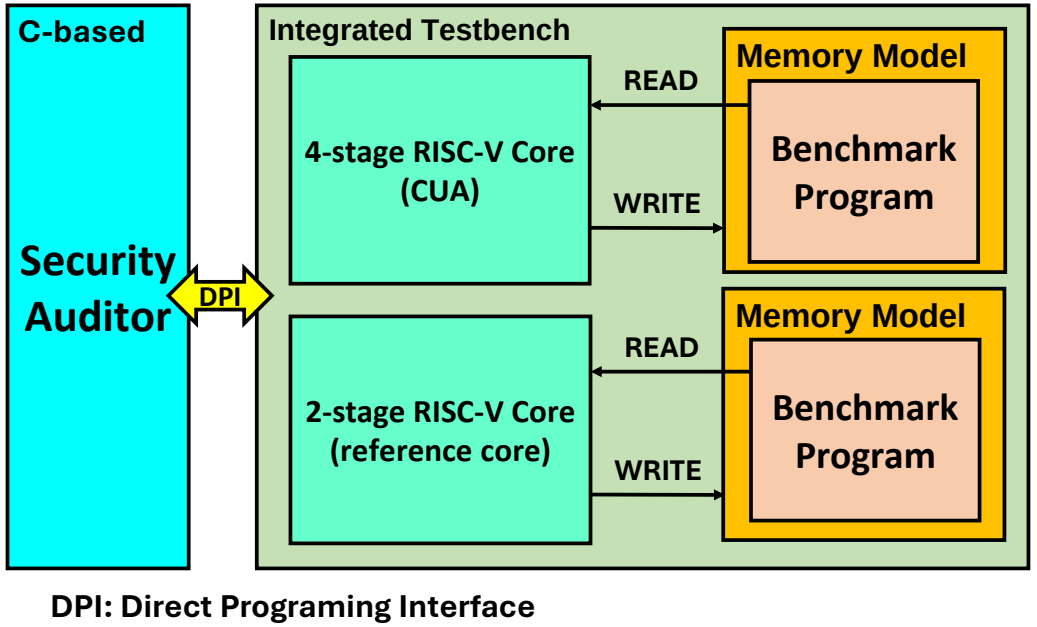


Fig. 5-1. Overview of our framework

Simulation Time Overhead

Program	Simulation Time For CUA alone	Simulation Time Double-Core Cross-Auditing	Simulation Time overhead (%)
1. Matrix Multiplication	0.84 (s)	0.89 (s)	5.95 %
2. Dhrystone2.1	19.44 (s)	52.00 (s)	167.49 %
3. ISR Sample	0.70 (s)	0.73 (s)	4.29 %
4. RISC-V Arch	1.11 (s)	1.47 (s)	32.43 %
5. RISC-V Compliance Test	0.90 (s)	0.96 (s)	6.67 %
6. RISC-V ISA Test	0.77 (s)	0.78 (s)	1.30 %

Table 1. Functional simulation times of single RISC-V versus double-core cross-auditing.

1.30 ~ 167.49 %

Benefit of Dynamic PQ management

Program Name	Program type	Program Description	Processing Time (Basic) (ms)	Processing Time (Dynamic PQ) (ms)	Speedup
Matrix Multiplication	Functional	Compute matrix multiplication program	2.88	0.10	28.56 X
Dhrystone2.1		Core benchmark program	308.82	1.17	264.39 X
ISR Sample		Test Interrupt Service Routine	0.21	0. 13	1.57 X
RISC-V Arch	Structural	Fundamental architecture check, do not check all the combination of instruction sets.	2.50	0.14	17.74 X
RISC-V Compliance		Verify RISC-V processor compatibility, functionality, and adherence to minimal instruction usage.	2.19	0.12	18.09 X
RISC-V ISA		Confirm that all RV32I instructions are operational.	0.11	0.10	1.11 X

Table 5-2. Processing Time benefits of using the dynamic PQ technique versus the basic post-processing technique.

Trojan Experiment

Trojan Name	Trojan Effect	Activation Mechanism	Affected Care Outputs (ACO) Detection Rate
MC8051-T800	Denial of Signal Transmission	Externally direct user input	100
PIC16F84-T700		Internally conditionally triggered	100
PIC16F84-T100			100
PIC16F84-T200			100
PIC16F84-T400			100
MC8051-T600	Change of Functionality	Externally direct user input	100
MC8051-T400		Internally conditionally triggered	100
MC8051-T500			100
B19-T300		Internally time-based triggered	100
B19-T400			100
B19-T500			100
PIC16F84-T300	Leakage of Information	Internally conditionally triggered	100

Table 5-3. Experimental results on detecting various Trojans implanted into the RISC-V core under auditing.

PIC16F84-T100

■ **Denial of Signal Transmission/Internally conditionally activated**

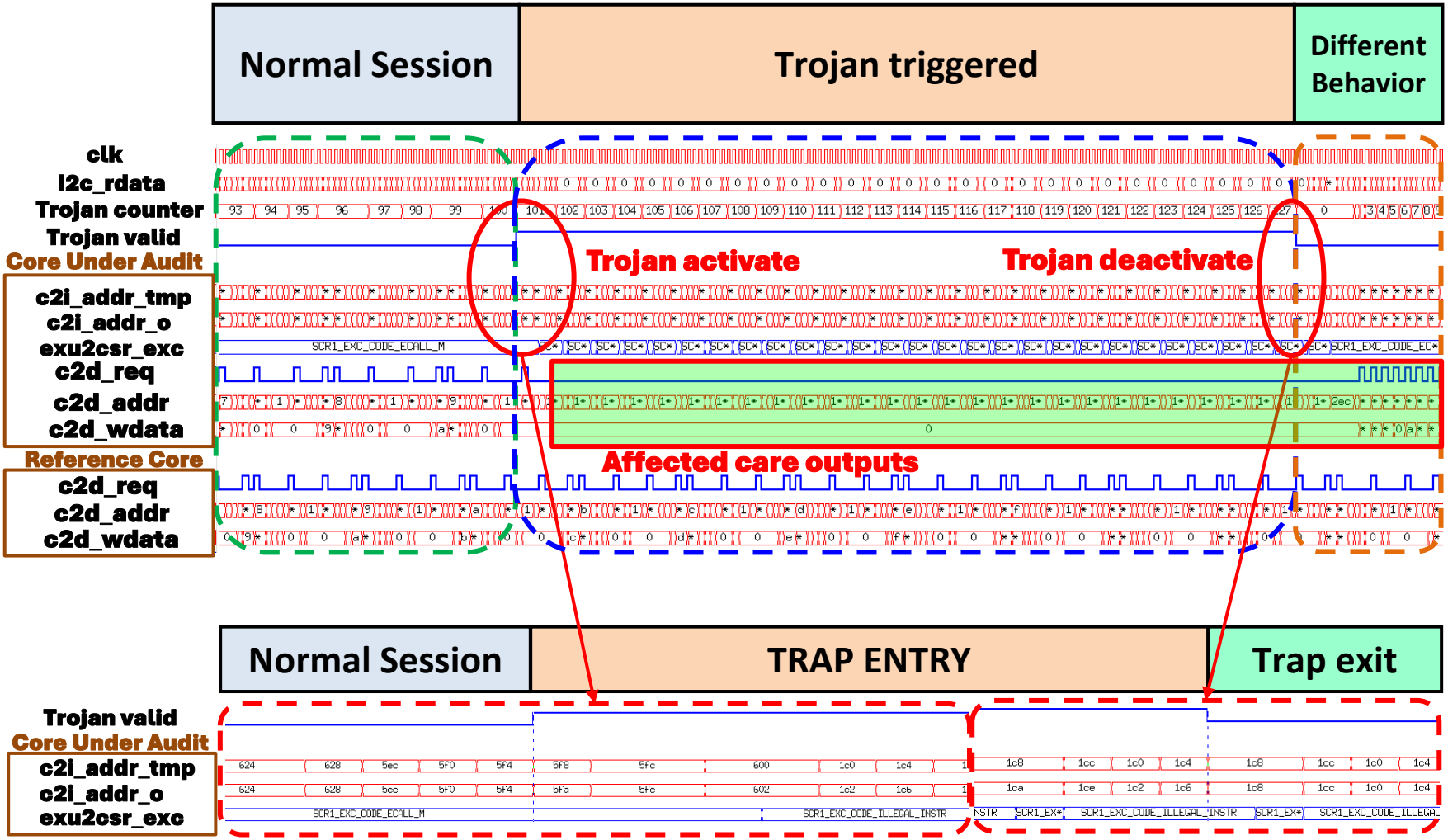


Fig. 5-2. The simulation waveforms of our framework for a RISC-V core implanted with a PIC16F84-T100 Trojan in [16]. The other reference core is Trojan-free.

Outline

- Introduction
- Related Work
- Preliminaries
- Proposed Methodology
- Experiment Result
- Further Discussion
- **Conclusion**

Conclusion

- Cross-auditing framework is very user-friendly as it does not need to specify the Trojan models explicitly.
- It is non-invasive as it does not require the source codes of the core under auditing.
- Low processing-time overhead complexity of only O(log2 n) for each care-output packet produced during the cross-auditing process, with n denoting the unmatched packets between the two cores.
- 9 Trojan types reported in a well-recognized “Trust-hub” platform show its nearly 100% coverage.

The End

Reference

- [1] S. Bhunia, M. Hsiao, M. Banga, and S. Narasimhan, “Hardware Trojan Attacks: Threat Analysis and Countermeasures,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229-1247, 2014.
- [2] M. Tehranipoor and F. Koushanfar, “A Survey of Hardware Trojan Taxonomy and Detection,” *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10-25, 2010.
- [3] J. Rajesh et al., “Hardware trojan attacks in soc and noc,” *in The Hardware Trojan War. Springer*, 2018, pp. 55–74.
- [4] Xiaolong Guo, R. G. Dutta, Yier Jin, F. Farahmandi, and P. Mishra, "Pre-silicon security verification and validation: A formal perspective," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2015.
- [5] H. Salmani, M. Tehranipoor, and R. Karri, "On Design vulnerability analysis and trust benchmark development", *Proc. of IEEE Int’l Conf. on Computer Design (ICCD)*, pp. 471-474, 2013.
- [6] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, “Benchmarking of Hardware Trojans and Maliciously Affected Circuits”, *Journal of Hardware and Systems Security (HaSS)*, pp. 85-102, April 2017.
- [7] S. Bhunia, M. S. Hsiao, M. Banga and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," *in Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229-1247, Aug. 2014.

Reference

- [8] M. R. Fadiheh, D. Stoffel, C. Barrett, S. Mitra, and W. Kunz, “Processor Hardware Security Vulnerabilities and their Detection by Unique Program Execution Checking”, *Proc. of IEEE Design Automation and Test in Europe*, pp. 994-999, 2019.
- [9] Z. Liu, O. Arias, W. Fu, Y. Jin, and X. Guo, “Inter-IP Malicious Modification Detection through Static Information Flow Tracking”, *Proc. of IEEE Design Automation and Test in Europe*, pp. 600-603, 2022.
- [10]J. Rajendran, V. Vedula, and R. Karri, “Detecting Malicious Modifications of Data in Third-Party Intellectual Property Cores”, *Proc. of IEEE Design Automation Conf.*, pp. 1-6, 2015.
- [11]IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language, *IEEE 1800-2012 Std.*, 2013.
- [12]M. Orenes-Vera, A. Manocha, D. Wentzlaff and M. Martonosi, "AutoSVA: Democratizing Formal Verification of RTL Module Interactions," *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 535-540, 2021.
- [13]P. Bhamidipati, S. M. Achyutha, and R. Vemuri, "Security Analysis of a System-on-Chip Using Assertion-Based Verification," *Proc. of IEEE Int’l Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 826-831, 2021.

Reference

- [14]A. Palumbo, L. Cassano, P. Reviriego, G. Bianchi, and M. Ottavi, “A Lightweight Security Checking Module to Protect Microprocessors against Hardware Trojan Horses”, *Proc. of IEEE Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pp. 1-6, 2021.
- [15]H. Chi, K. Lee, and T. Jao, “Lightweight HardwareBased Memory Protection Mechanism on IoT Processors”, *Proc. of IEEE Asian Test Symp.*, pp. 13-18, 2021.
- [16]M. Tehranipoor, R. Karri, F. Koushanfar, and M. Potkonjak, “Trusthub,” http:// trust-hub.org.
- [17]M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, “Meltdown,” arXiv preprint arXiv:1801.01207, 2018.
- [18]P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, “Spectre attacks: Exploiting speculative execution,” arXiv preprint arXiv:1801.01203, 2018.
- [19]SCR1 RISC-V Core – https://github.com/syntacore/scr1.
- [20] “EDA cloud Cell-based Flow” Taiwan Semiconductor Research Institute, TSRI