Intersecting Hardware Cybersecurity with Machine Learning

Prof. Anirban Sengupta, *FIET, FBCS, FIETE*

Distinguished Contributor of the IEEE Computer Society

Distinguished Visitor of the IEEE Computer Society

Chair, Distinguished Visitor Selection Committee of IEEE Computer Society

Department of Computer Science and Engineering

Indian Institute of Technology (IIT) Indore

Online Faculty Development Programme (FDP) on the theme "AI Toolkit for 21st Century Educators"

10-OCT-2025

INTRODUCTION

- A hardware Trojan is a malicious logic inserted into an IP design during its design or manufacturing process.
- Hardware Trojan are designed to remain undetected until triggered by specific conditions.

MOTIVATION AND THREAT MODEL

- The goal of this approach is to highlight how a malicious HLS framework is capable of inserting hardware Trojans during the Mux-based interconnect stage of watermarked IP design.
- The impact of a malicious HLS tool may be performance degradation or denial of service.

TROJAN VULNERABILITY IN HLS-BASED WATERMARKED IPS

- Approaches [1], [2], [3] are examples of HLS-based watermarking for IP designs.

- After embedding secret security constraints in the register allocation phase of HLS, the multiplexer-based interconnect design may get altered and yield a free port (input pin of mux) that can be exploited with malicious intent by an attacker to insert Trojan.

Overview

- The talk provides a broad perspective for readers on how security vulnerabilities can be exploited by hackers during the design of machine learning (ML) accelerators such as convolutional neural networks.

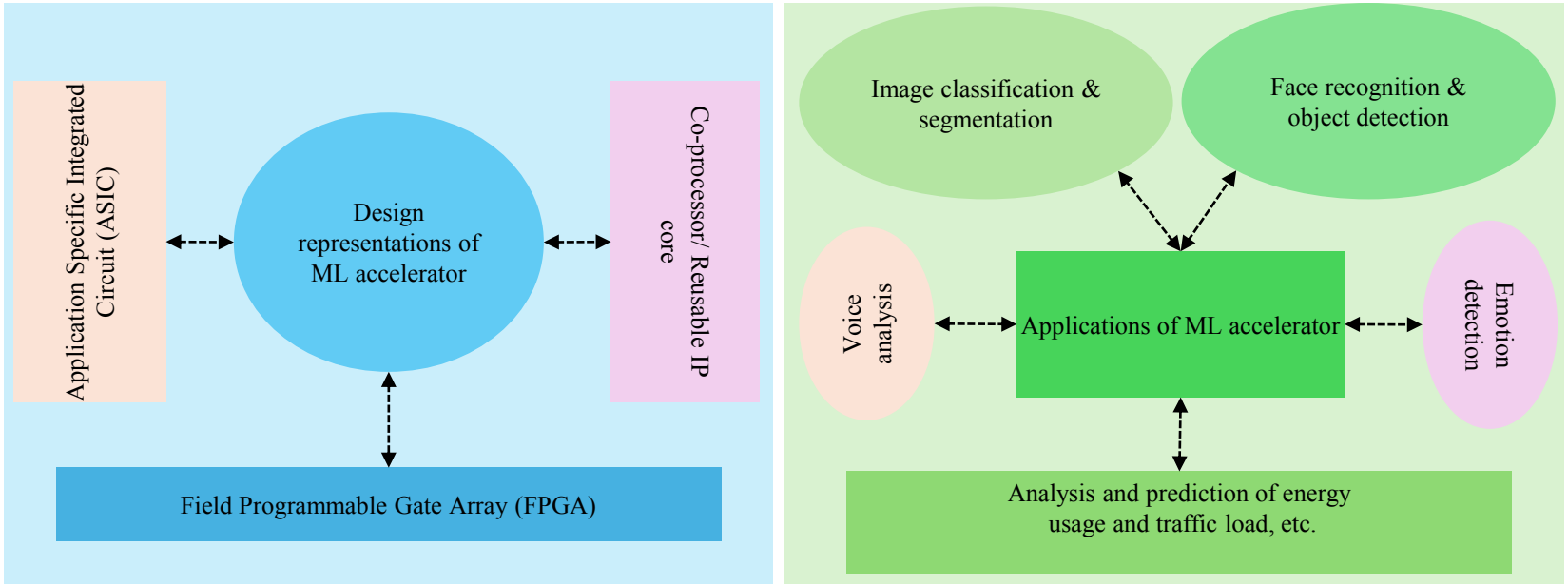
- It also provides a broad-spectrum review of existing literatures on how Trojan attacks can be injected, triggered, and exploited to cause various payloads, such as degraded performance, denial of service, data damage, and power/battery exhaustion, on ML accelerators.

- The article also discusses possible detection/mitigation techniques from the literature for some of the attacks and recommends a possible solution.

A. Sengupta, A. Anshul, V. Chourasia and N. Bhui, "Security Vulnerability (Backdoor Trojan) During Machine Learning Accelerator Design Phases," ***IEEE IT Professional***, vol. 27, no. 1, pp. 65-72, 2025.

ML/Hardware Accelerators

- ML/hardware accelerators enhance data-intensive tasks like image recognition, natural language processing, and autonomous driving, enabling faster data processing and improved efficiency in real-life applications.
  - Some real-life examples of ML accelerators are NVIDIA Deep Learning Accelerator (NVDLA) [4], LR hardware accelerator [6], convolutional layer hardware accelerator [9], etc.



(a) (b)

Fig. 1. (a) Different design representations for machine learning (ML) accelerators, and (b) Different applications of ML accelerators

[4] N. Gupta, A. Jati and A. Chattopadhyay, AI Attacks AI: Recovering Neural Network architecture from NVDLA using AI-assisted Side Channel Attack, *Cryptology {ePrint} Archive*, Paper 2023/368, 2023, url = https://eprint.iacr.org/2023/368.

[6] A. Sengupta, R. Chaurasia, M. Rathor: HLS-based swarm intelligence driven optimized hardware IP core for linear regression-based machine learning, *IET Journal of Engineering*, e12299 (2023).

[9] A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol.68, no. 3, pp. 291-306, 2022.

Security Vulnerability in ML/Hardware Accelerators

➢ **How do hackers (attackers) exploit security vulnerability in ML/Hardware accelerators?**

- ML accelerators or in general hardware accelerators may be designed using HLS framework/RTL designing [3], [6], [9]. In various steps of HLS/RTL design, attackers (within the design house) can compromise and exploit a computer-aided design (CAD) software tool and/or RTL design to covertly inject backdoor Trojans.
- As shown in Fig. 1. (c), a hardware Trojan attack on a crypto-accelerator has capability to bypass the encryption circuit and leak confidential information. On rare even triggering, the encryption is bypassed easily.
- Moreover, security vulnerability of NVIDIA accelerators (NVDLA) has been exposed in [4], as shown in Fig. 1. (d). Power and side channel leakage information from CNN models have been used to train CNN based attack models.

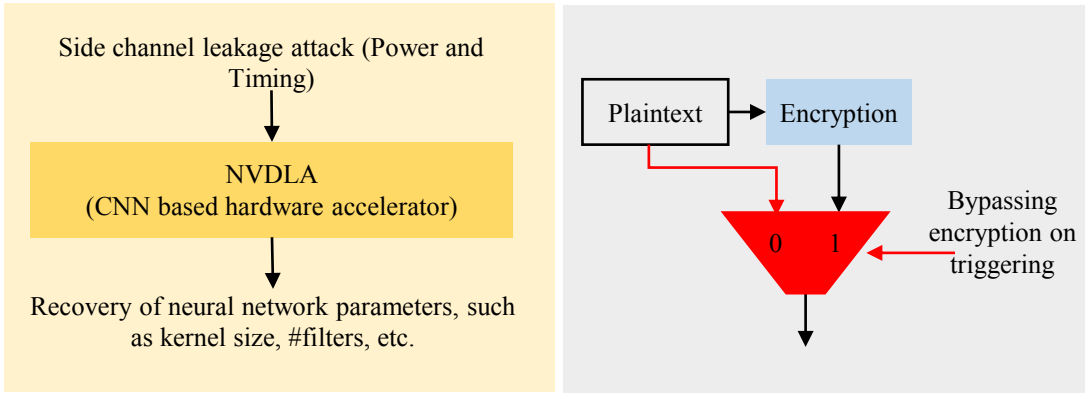


Fig. 1. (c) Example of Trojan attack and security vulnerability in a cryptographic accelerator, and (d) Example of Trojan attack and security vulnerability of a real-world application (NVDLA accelerator)

[3]Why you Need HLS for Machine Learning Accelerators, accessed in 2024, Available: https://resources.sw.siemens.com/en-US/video-why-you-need-hls-for-machine-learning-accelerators.
[4] N. Gupta, A. Jati and A. Chattopadhyay, AI Attacks AI: Recovering Neural Network architecture from NVDLA using AI-assisted Side Channel Attack, *Cryptology {ePrint} Archive*, Paper 2023/368, 2023, url = https://eprint.iacr.org/2023/368.
[6] A. Sengupta, R. Chaurasia, M. Rathor: HLS-based swarm intelligence driven optimized hardware IP core for linear regression-based machine learning, *IET Journal of Engineering*, e12299 (2023).
[9] A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol.68, no. 3, pp. 291-306, 2022.

Security Vulnerability in ML/Hardware Accelerators

- **How do hackers (attackers) exploit security vulnerability in ML/Hardware accelerators?**
  - In another real-life scenario, an attacker can accelerate the aging process of a computing device, such as digital signal processing (DSP) accelerator, by exploiting negative bias temperature instability (NBTI) stress as hardware Trojan.
  - By applying NBTI stress based Trojan attack, an attacker puts stress on PMOS transistors by increasing their threshold voltage. This causes them to degrade in terms of performance delay and can expedite the aging related performance degradation. This has been established in [5].
  - Furthermore, an attacker can inject a trojan during scheduling phase, allocation phase and max interconnect design phase. For example, a hacker can also secretly insert Trojan (pseudo/fake) operations during the scheduling phase of the HLS design process (resulting in a battery exhaustion attack) [17].
  - Further, a hacker can also exploit the Mux-based interconnect design stage during HLS to secretly insert Trojans (such as denial-of-service hardware Trojan (DoS HT), performance degradation hardware Trojan (PD-HT), data damage hardware Trojan (DD-HT)) into the ML accelerators (adopted from [7]).

  The goal of the attacker while launching such Trojan attacks is to maliciously exploit any unused free port or under-utilized resources to inject Trojan logic.

[5] D. Kachave and A. Sengupta, "Digital Processing Core Performance Degradation Due to Hardware Stress Attacks," *IEEE Potentials*, vol. 38, no. 2, pp. 39-45, March-April 2019.
[7] A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.
[17] C. Pilato, K. Basu, F. Regazzoni and R. Karri, "Black-Hat High-Level Synthesis: Myth or Reality?," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 913-926, 2019.

Security Vulnerability in ML/Hardware Accelerators

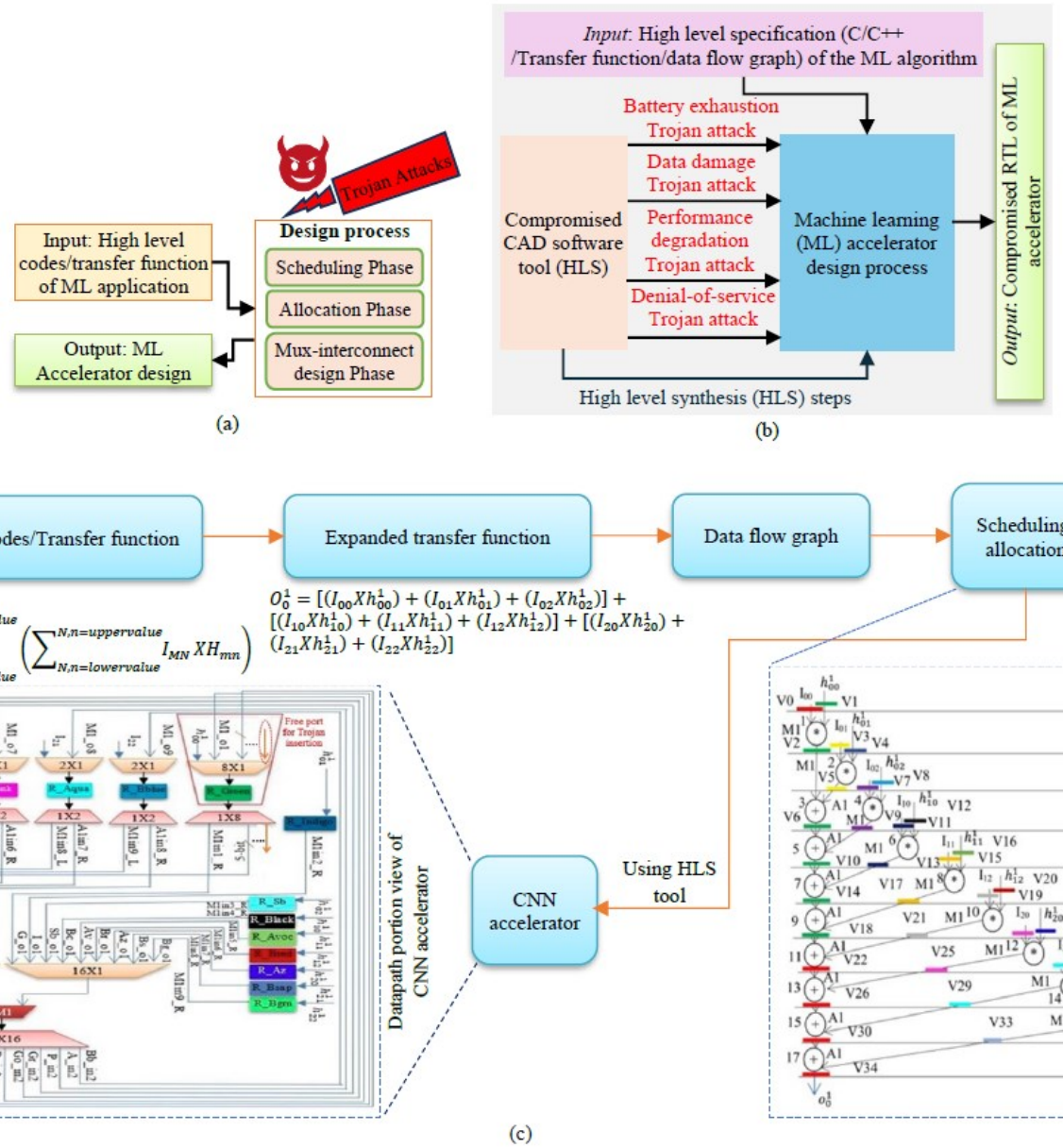


Fig. (a) Overview of Trojan attack during ML accelerator design process, (b) Established Trojan attacks on ML accelerator, and (c) Design flow of ML (CNN) co-processor/accelerator (adopted from [9]).

*Note: '$\boldsymbol{I_{MN}}$' and '$\boldsymbol{H_{mn}}$' in the transfer function represents the input image of size **MxN** and kernal of size **mxn** respectively. $\boldsymbol{O_y}$ denotes the output value of each element/pixel corresponding to output feature map; further in the expanded transfer function, each pixel value of the input image matrix and each kernel value of kernel matrix 't' is represented by $\boldsymbol{I_{ab}}$ and $\boldsymbol{h_{pq}^t}$ respectively*

[9] A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol.68, no. 3, pp. 291-306, 2022.

Security Vulnerability in ML/Hardware Accelerators

- As shown in Fig. (c), initially, the high-level code/transfer function of the ML application is taken as input. For example, the CNN convolution layer's transfer function is shown in Fig. 2.(c).
- Further, the expanded transfer function is generated. Next, the corresponding data flow graph (DFG/CDFG) is generated [9]. Subsequently, the DFG is fed as input to the HLS scheduling and allocation block. Finally, ML accelerator RTL datapath is generated post datapath synthesis.
- Fig. (c) also depicts the datapath portion view of the CNN convolutional layer accelerator [9]. As evident in the datapath portion view, some input ports are free (unutilized) in the Mux-based interconnect design of the shown datapath (ports shown in orange).
- It has been established in the literature [7], that these unused free ports can be exploited by the attacker during compromising a CAD HLS tool (to secretly insert the Trojan), without the knowledge of the ML accelerator designer (who is using the tool), causing different payloads (such as denial-of-service hardware Trojan (DoS HT), performance degradation hardware Trojan (PD-HT), data damage hardware Trojan (DD-HT)).
- Additionally, it has also been established in the literature [17], that an attacker can also exploit the scheduling phase of the HLS framework to insert pseudo/fake operations to launch a battery exhaustion attack.
- The various Trojan payloads [2] can cause different types of adversarial effects in ML accelerators.

[2] Xue, M., Gu, C., Liu, W., Yu, S. and O'Neill, M. (2020), Ten years of hardware Trojans: a survey from the attacker's perspective. *IET Comput. Digit. Tech*., 14: 231-246.
[7] A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.
[9] A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol.68, no. 3, pp. 291-306, 2022.
[17] C. Pilato, K. Basu, F. Regazzoni and R. Karri, "Black-Hat High-Level Synthesis: Myth or Reality?," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 913-926, 2019.

TROJAN VULNERABILITY IN HLS-BASED WATERMARKED IPS

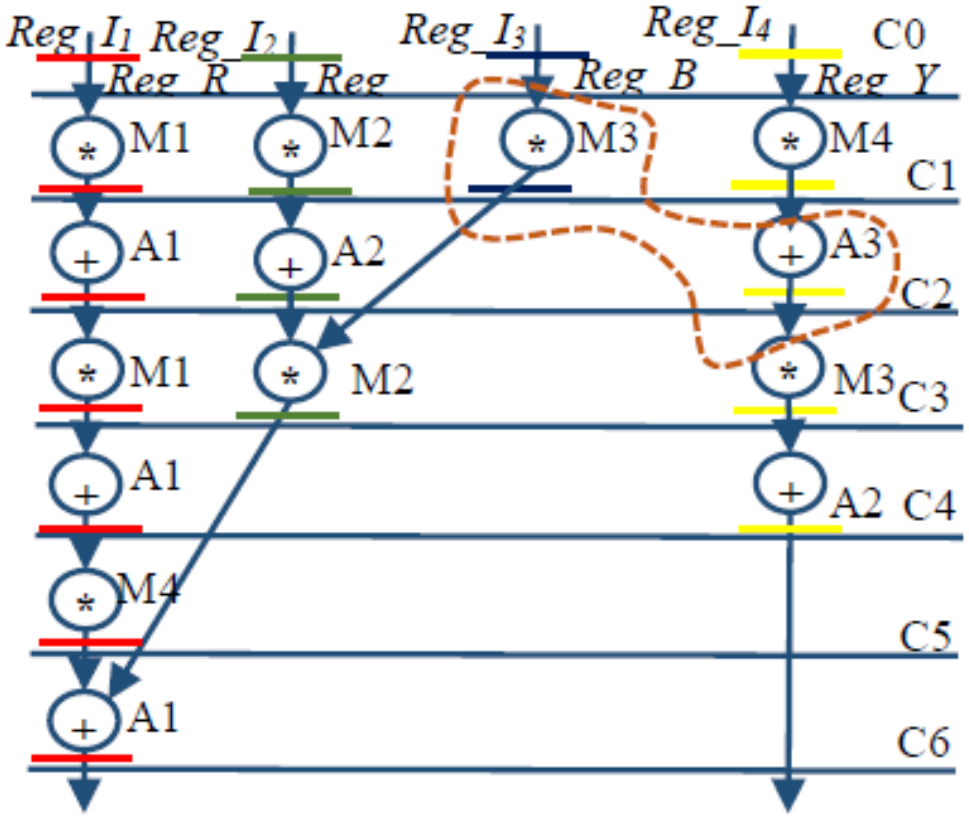


Fig. Scheduled data flow graph of MESA Horner Bezier without watermark

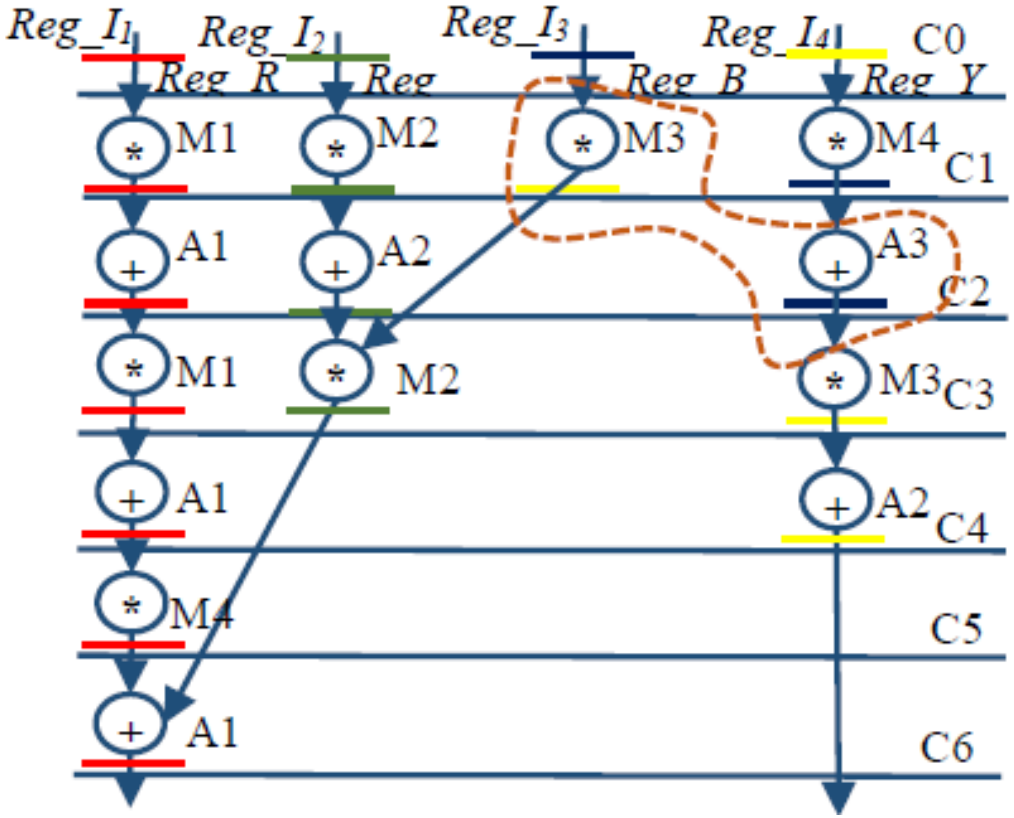


Fig. Scheduled data flow graph of MESA Horner Bezier with embedded watermark

TROJAN VULNERABILITY IN HLS-BASED WATERMARKED IPS

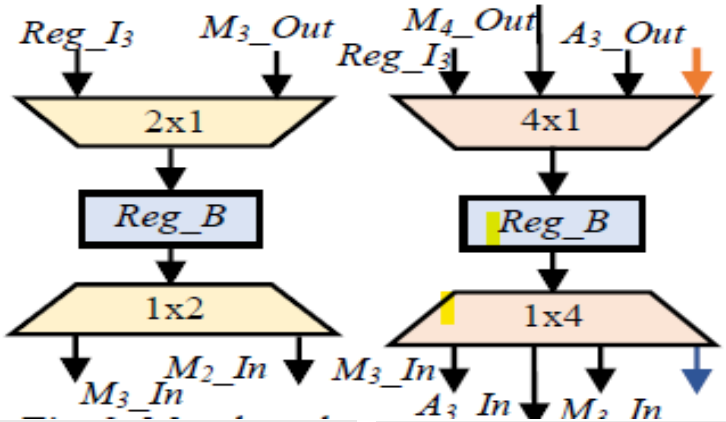


Fig. Mux-based interconnect design of datpath without watermark in register (Reg_B) corresponding to Fig. 1

Fig. Mux-based interconnect design of datpath watermarking in register (Reg_B) corresponding to Fig. 2

PROPOSED MALEVOLENT HLS FRAMEWORK

- Fig.highlights the proposed malevolent high-level synthesis framework.

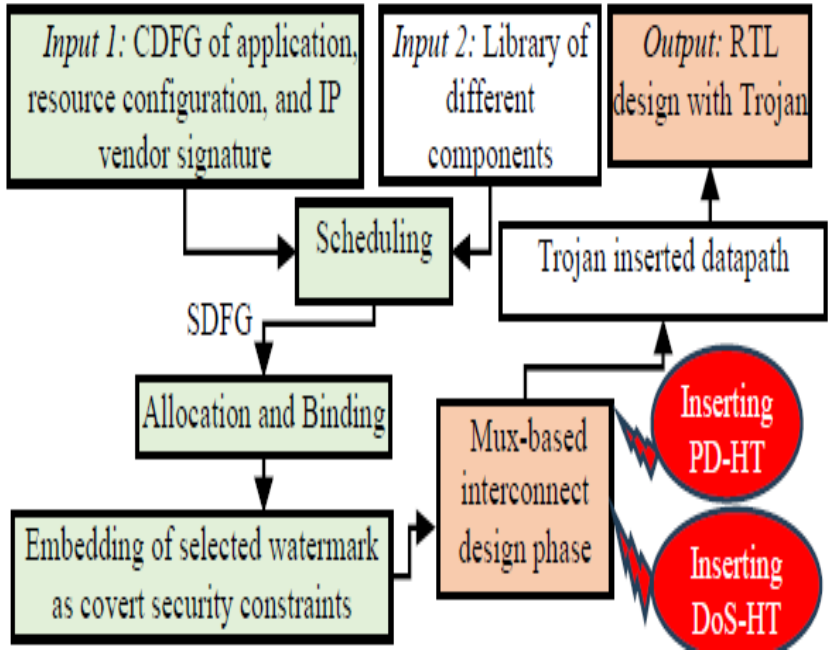


Fig. Malevolent High-Level Synthesis framework

Types of Trojan Attacks

- The four established Trojan attacks on ML accelerators are as follows:
  - Performance degradation hardware Trojan (PD-HT),
  - Data damage hardware Trojan (DD-HT),
  - Denial-of-service hardware Trojan (DoS-HT), and
  - Battery exhaustion Trojan (BE-HT).

Note: *Here, the orange-colored components indicate Trojan logic inserted by the attacker and orange free port input on top of 8x1 multiplexer indicates unutilized port exploited by the hacker for secret Trojan insertion.*

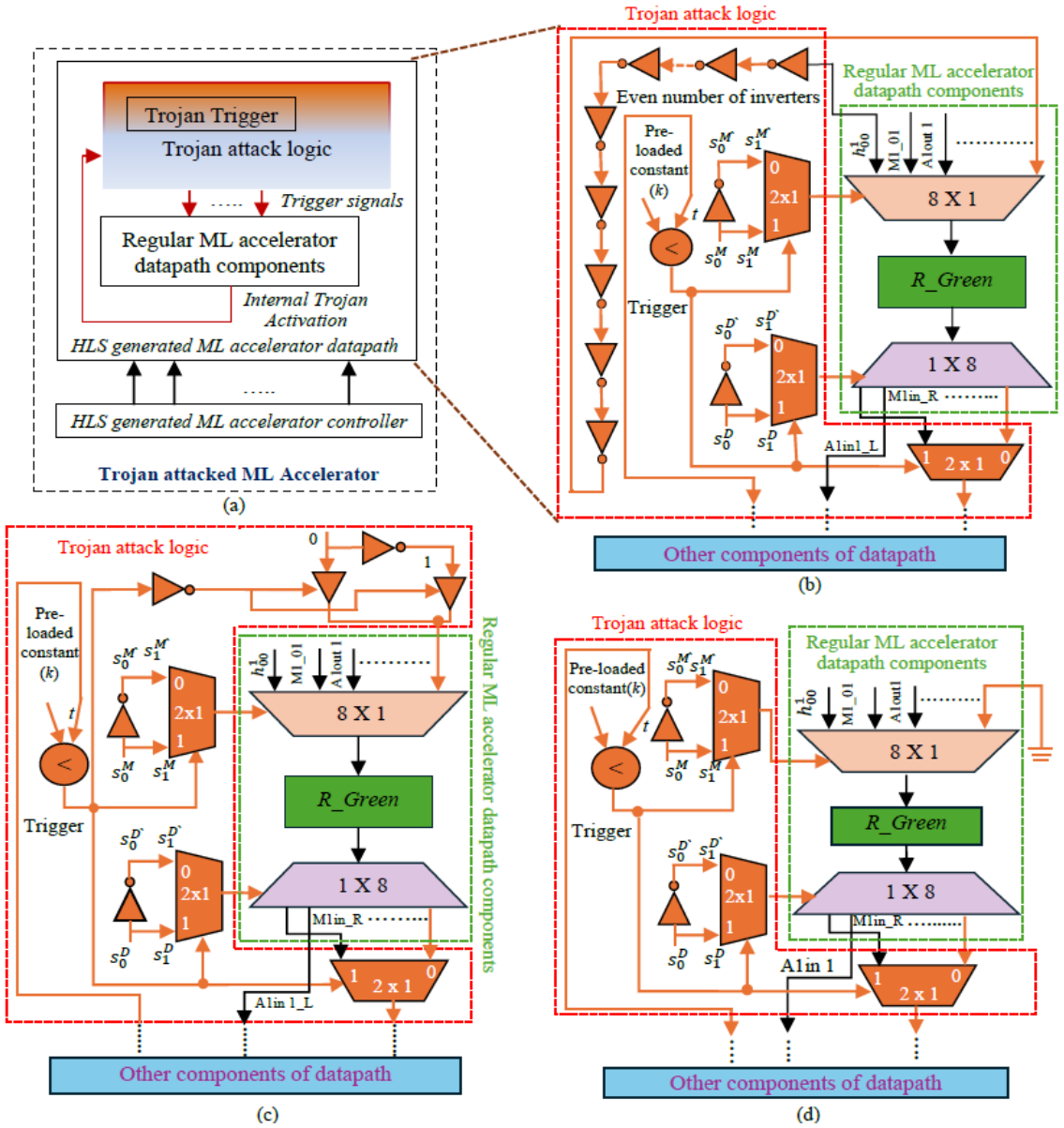
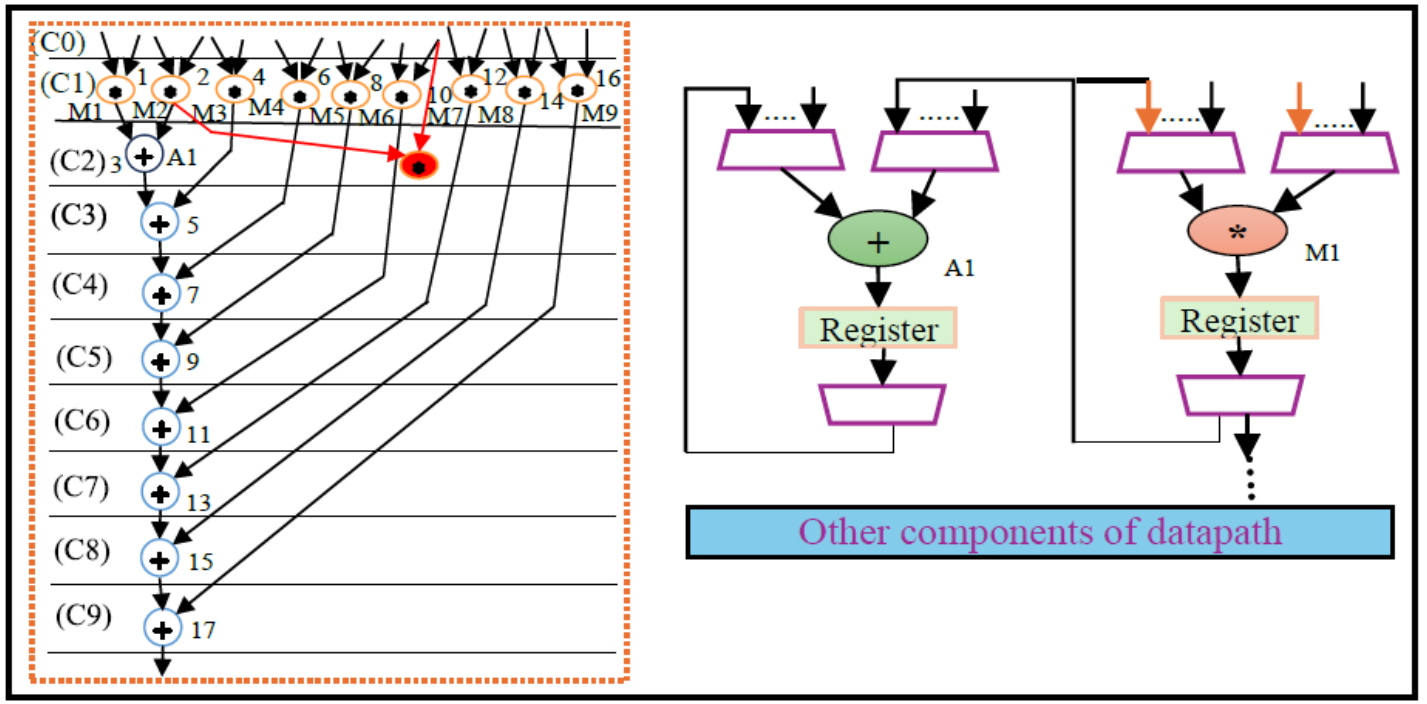


Fig. (a) Trojan attacked ML Accelerator design, (b) Performance degradation Trojan attack, (c) Denial-of-service Trojan attack, and (d) Data Damage trojan attack

Types of Trojan Attacks (Contd.)

- Fig. (e) demonstrates the integration of a BE-HT in the ML accelerator, designed to increase power consumption and speed up battery depletion.
  - The primary goal is to reuse the idle functional units (FUs) in the ML accelerator datapath. Multipliers, which have larger power dissipation, are chosen to increase the overall power consumption of the accelerator. Such modifications in the datapath have nominal area and power overhead.
  - This technique does not affect the final computational output while concurrently not enhancing the power overhead of the design substantially.



(e)

Fig.(e) Battery exhaustion Trojan attack in CNN convolutional layer accelerator

Triggering of Backdoor Trojan

- Hardware Trojans are stealthily inserted into systems that become active only when a specific rare condition (predetermined by the attacker) is met.
- This activation has been managed through comparator logic that switches on the Trojan's payload when the rare condition is satisfied.
- These Trojans are extremely difficult to detect because they stay inactive (dormant and undetected until triggered by specific conditions) during regular system operations. During the insertion of the Trojan, an attacker programs a constant value 'k' into memory (electrically programmable) [7].
- As shown in Figures (b), (c), and (d), the first input (t) of the comparator is connected internally to the functional units (such as adders, multipliers, etc.) of remaining ML accelerator datapath, and the second input is connected to memory holding pre-loaded constant 'k'. Once the system's state (t) matches this constant (k), the Trojan becomes triggered, causing it to execute its intended malicious effects.
- The above-explained trigger condition is the same for PD-HT, DD-HT, and DoS-HT. However, BE-HT has been designed in the literature to become triggered after a certain count value of the counter [17].
- These Trojans not only compromise the security of ML designs but also erode the trust between the ML accelerator vendors and CAD software communities [5].

[5] D. Kachave and A. Sengupta, "Digital Processing Core Performance Degradation Due to Hardware Stress Attacks," *IEEE Potentials*, vol. 38, no. 2, pp. 39-45, March-April 2019.
[7] A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.
[17] C. Pilato, K. Basu, F. Regazzoni and R. Karri, "Black-Hat High-Level Synthesis: Myth or Reality?," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 913-926, 2019.

Detection Techniques Employed for Backdoor Trojans in ML Accelerators

S. No.	Different detection techniques	Performance degradation Trojan attack	Data damage Trojan attack	Denial-of-service Trojan attack	Battery (Power) exhaustion Trojan attack
1	C to RTL Equivalence checking [10]	×	✓	×	✓
2	TL-HLS (DMR based security-aware scheduling) [11]	×	×	×	×
3	Side channel analysis [12]	×	×	×	×
4	Detection using reverse engineering [13]	×	×	×	×
5	Detection using path delay fingerprint [14]	×	×	×	×
6	GNN based detection [15]	×	×	×	×
7	HLT based detection [16]	×	×	×	✓

(a)

Fig. (a) Analysis of different detection techniques on Trojan infected ML accelerator designs (Note: '×' indicates "not detectable")

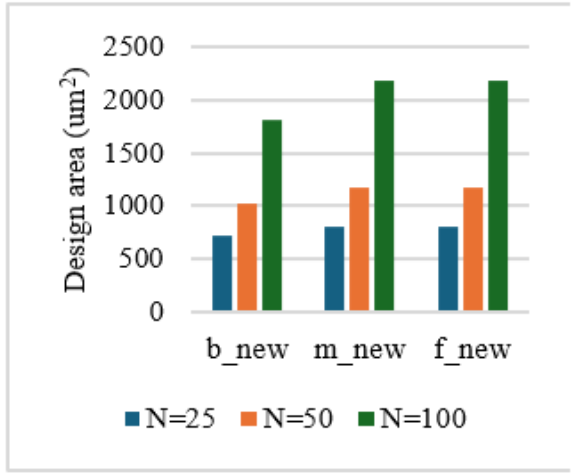
[10] M. Abderehman, R. Gupta, R. R. Theegala and C. Karfa, "BLAST: Belling the Black-Hat High-Level Synthesis Tool," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3661-3672, 2022.
[11] A. Sengupta, S. Bhadauria and S. P. Mohanty, "TL-HLS: Methodology for Low Cost Hardware Trojan Security Aware Scheduling With Optimal Loop Unrolling Factor During High Level Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 655-668, 2017.
[12] Y. Huang, S. Bhunia and P. Mishra, "Scalable Test Generation for Trojan Detection Using Side Channel Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2746-2760, 2018.
[13] M. Ludwig, A. -C. Bette and B. Lippmann, "ViTaL: Verifying Trojan-Free Physical Layouts through Hardware Reverse Engineering," IEEE Physical Assurance and Inspection of Electronics, USA, 2021, pp. 1-8.
[14] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," *IEEE International Workshop on HOST*, 2008, pp. 51–57.
[15] R. Yasaei, L. Chen, S. -Y. Yu and M. A. A. Faruque, "Hardware Trojan Detection using Graph Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.
[16] M. Rathor and A. Sengupta, "Revisiting Black-Hat HLS: A Lightweight Countermeasure to HLS-Aided Trojan Attack," *IEEE Embedded Systems Letters*, Volume: 16, Issue: 2, 2024, pp. 170-173.

Detection Techniques Employed for Backdoor Trojans in ML Accelerators (Contd.)

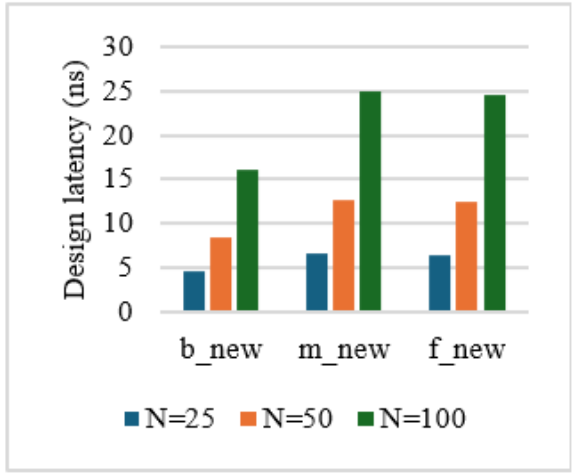
- Different types of Trojans, like PD-HT and DoS-HT, impacting performance and operational state, without altering functionality, makes them difficult to detect using equivalence checking [7], [10].
- On the other hand, DD-HT affects data output under rare condition triggering, making detection somehow possible through equivalence analysis [10].
- Further, BE-HT has been successfully detected using C to RTL equivalence checking based on finite state machine datapath (FSMD) extraction. Further, these Trojans remain undetected through side-channel analysis [12] as they don't leak significant parametric information (such as delay and power).
- Techniques like path delay fingerprinting [14], attempt to differentiate normal designs from those compromised by Trojans, however, they become impractical for complex HLS-generated ML accelerator.
- Detection tools relying on Graph Neural Networks (GNN) [15] face limitations in accurately detecting Trojans within ML accelerators, because its performance/accuracy for complex ML accelerators is lower due to weaker learning behavior.
- Moreover, the detection technique [16] is only capable of handling BE-HT attacks, as PD-HT, DD-HT, and DoS-HT do not induce Trojan payload using fake operation insertion.
- Therefore, based on the published detection techniques for Trojans, C to RTL functional equivalence checking [10] has been the most effective technique as it is capable of detecting both BE-HT and DD-HT.

[10] M. Abderehman, R. Gupta, R. R. Theegala and C. Karfa, "BLAST: Belling the Black-Hat High-Level Synthesis Tool," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3661-3672, 2022.

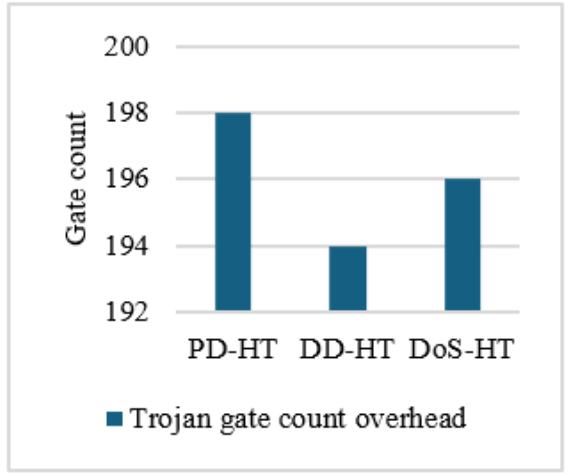
Analysis of ML Accelerators in terms of Design Area, Latency, and Resources



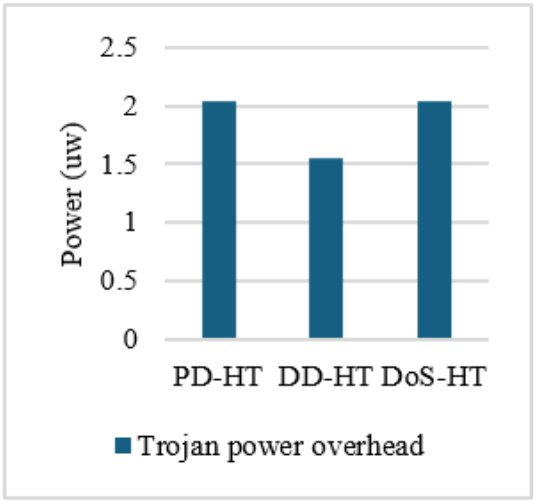
(b)



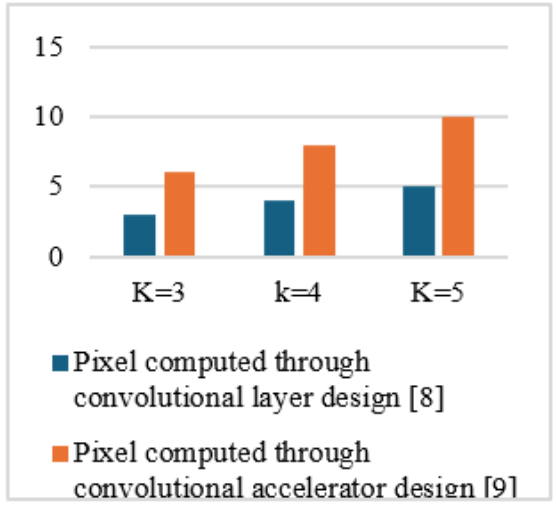
(c)



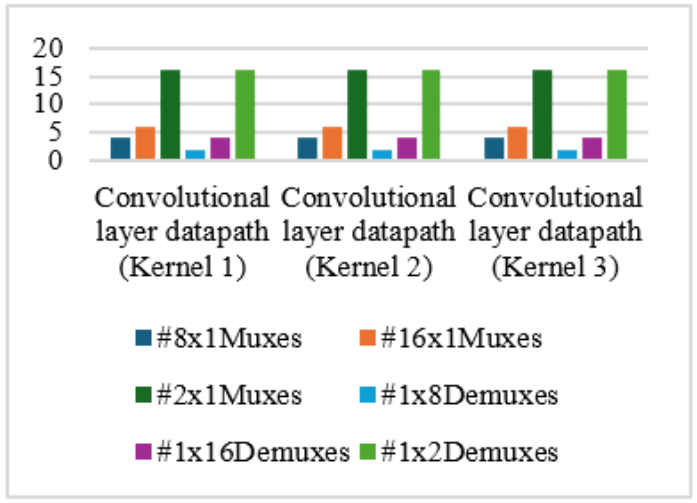
(d)



(e)



(f)



(g)

Fig. (b) Design area for LR-ML accelerator corresponding to different numbers of datasets (N) [6], (c) Design latency for LR-ML accelerator corresponding to different numbers of datasets (N) [6], (d) Trojan design area overhead (in terms of gate count) corresponding to convolutional layer CNN accelerator [7], (e) Trojan power overhead corresponding to convolutional layer CNN accelerator [7], (f) Comparison of number of pixels computed between [8] and [9] for different convolutional kernel filters (K), and (g) Resources required for convolutional layer datapath w.r.t. three different kernels

Analysis of ML Accelerators in terms of Design Area, Latency, and Resources (Contd.)

- This section presents the analysis of different ML accelerator designs from the literature. Figures (b) and (c) depict the design area and latency for the LR-ML accelerator corresponding to different numbers of datasets (N), respectively [6].
- The design area and latency are directly proportional to the number of datasets it handles. Subsequently, figures (d) and (e) show the design area (in terms of gate count) and power overhead corresponding to convolutional layer CNN accelerator after Trojan injection, respectively [7].
- The Trojan-infected design, on average, incurs a minimal increase in the ~196 gate count value and ~1.6 μw power as compared to the baseline ML-accelerator design [7].
- Next, Fig. (f) shows the comparison of pixel computation between [8] and [9] for different convolutional kernel filters (K).
- Approach [9] surpasses [8] in terms of pixel computation value due to parallel pixel computation process owing to loop unrolled architecture. Finally, Fig. (g) highlights required resources for convolutional layer accelerator datapath w.r.t. three different kernels [9].

[6] A. Sengupta, R. Chaurasia, M. Rathor: HLS-based swarm intelligence driven optimized hardware IP core for linear regression-based machine learning, *IET Journal of Engineering*, e12299 (2023).
[7] A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.
[8] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," *International Conference on Engineering and Technology*, Turkey, 2017, pp. 1-6.
[9] A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol.68, no. 3, pp. 291-306, 2022.

HLS Trojan Detection using Machine Learning Technique

- A HLS Trojan detection technique has been proposed that is capable of detecting the following HLS Trojans: performance degradation hardware Trojan (PD-HT), denial-of-service hardware Trojan (DoS-HT), battery exhaustion hardware Trojan (BE-HT), downgrade attack hardware Trojan (DA-HT), and functional hardware Trojan (F-HT).
- A feature extraction block has been proposed, which is responsible for the extraction of several crucial features from hardware intellectual property (IP) register transfer level (RTL) design (such as VHDL).
- A HLS Trojan detection technique has been proposed using lightweight decision tree classifier-based machine learning (ML) model.

- ***Threat model***: Trojans in HLS-based hardware IP designs present significant security vulnerability, exploiting the automated and abstract nature of HLS tools. Compromised HLS flows/tools may embed malicious Trojans, degrading IP integrity during key HLS design phases, like scheduling, resource allocation, interconnect design, and datapath/controller synthesis phase, enabling national-level attacks on system-on-chip (SoC) development and undermining hardware security [1], [2].

[1] A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.
[2] C. Pilato, K. Basu, F. Regazzoni and R. Karri, "Black-Hat High-Level Synthesis: Myth or Reality?," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 913-926, April 2019.

TABLE I

DIFFERENT TYPES OF HLS-TROJANS, THEIR HLS BASED INSERTION STAGES, AND CORRESPONDING TRIGGERING MECHANISM, PAYLOAD

HLS Trojans	HLS Insertion Stage	Triggering mechanism	Payload	Remarks
PD-HT	Mux-interconnect design	*Comparator based trigger*	Performance degradation	Exploiting *chain of inverters* to cause excessive delay
DoS-HT	Mux-interconnect design	*Comparator based trigger*	Denial-of-service	Exploiting *TSB* to cause high impedance state
BE-HT	Scheduling/FSM	*Counter-based trigger*	Battery exhaustion/power drainage	Exploiting *unutilized functional units* to cause excess power drainage based on a certain execution count
DA-HT	Datapath	*Input sequence detector*	Downgrade attack/compromised security	Exploiting a *2:1 Mux* to implement downgrade attack
F-HT	HLS library	*External signal, FSM counter*	Compromising functionality/data damage	Exploiting a *2:1 Mux* and *inverter* to alter the functionality of used functional units

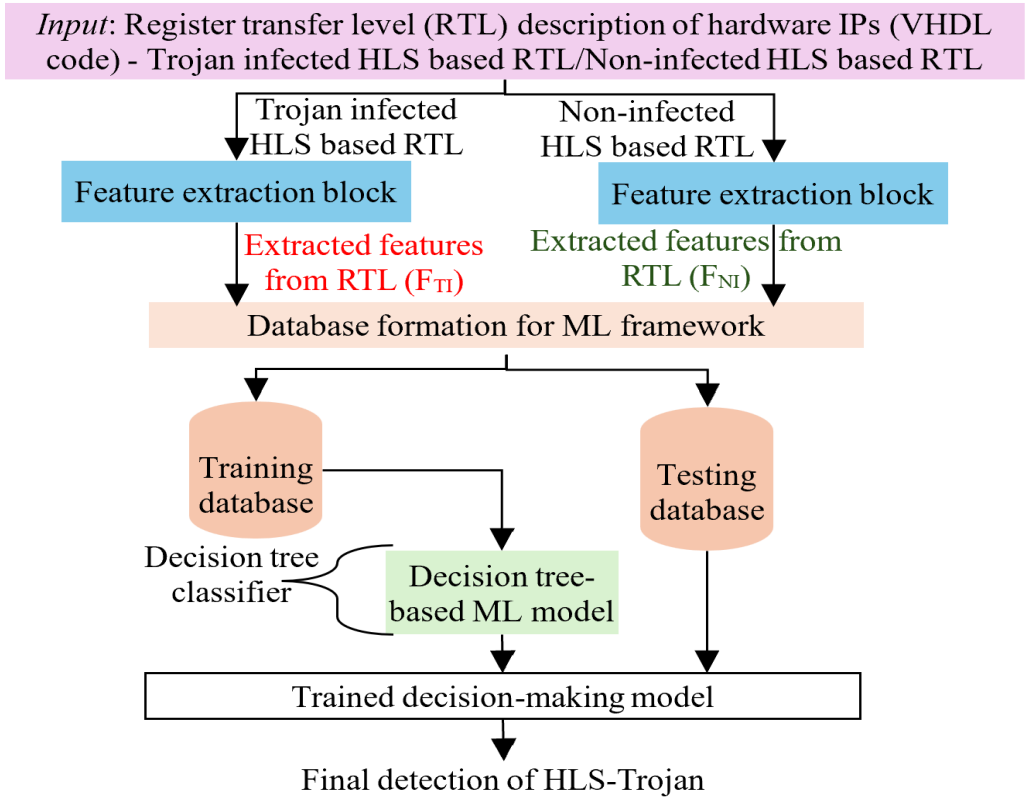


Fig. Details of the proposed ML-based HLS-Trojan detection methodology

***Input*:** RTL description of hardware IPs (*VHDL* code) - Trojan infected HLS based RTL/non-infected HLS based RTL
***Output*:** Feature set corresponding to input RTL

R ← RTL description file
Φ ← Port map information present in R
FOR each line (i) in R
IF (*mux_2_to_1* ∃ i)
*2*1 Mux* = *2*1 Mux* + 1
IF (*mux_4_to_1* ∃ i)
*4*1 Mux* = *4*1 Mux* + 1
IF (*mux_8_to_1* ∃ i)
*8*1 Mux* = *8*1 Mux* + 1
IF (*mux_16_to_1* ∃ i)
*16*1 Mux* = *16*1 Mux* + 1
IF (*mux_32_to_1* ∃ i)
*32*1 Mux* = *32*1 Mux* + 1
IF (demux_2_to_1 ∃ i)
*2*1 DeMux* = *2*1 DeMux* + 1
IF (*demux_4_to_1* ∃ i)
*4*1 DeMux* = *4*1DeMux* + 1
IF (*demux_8_to_1* ∃ i)
*8*1 DeMux* = *8*1 DeMux* + 1
IF (*demux_16_to_1* ∃ i)
*16*1 DeMux* = *16*1 DeMux* + 1
IF (*demux_32_to_1* ∃ i)
*32*1 DeMux* = *32*1 DeMux* + 1
IF (*latch* ∃ i)
Latch = *Latch* + 1
……
……
IF (*Adder* ∃ i)
Adder = *Adder* + 1
IF (*Subtractor* ∃ i)
Subtractor = *Subtractor* + 1
IF (*Multiplier* ∃ i)
Multiplier = *Multiplier* + 1
IF (*Comparator* ∃ i)
comparator = *1*
IF (*up_counter* ∃ i)
Up-counter = *1*
IF (*tri_state_buffer* ∃ i)
Tri-state buffer = 1
END IF
END FOR
IF (Output of register → input of *comparator* || Output of register → input of Mux *in* Φ)
Memory element = 1
IF (Output of Mux → input of another Mux *in* Φ)
Mux output acting as input for another Mux = 1
IF (Output of NOT gate → input of another NOT gate in Φ)
Inverter Chain = 1
IF (Third input port of a component (functional unit) ∃ Φ)
Third_FU_Input = 1

Fig. Algorithm (Pseudocode) to perform feature extraction from the input RTL description (VHDL code) of hardware IP design

TABLE II
THE COMPARISON OF PROPOSED ML-BASED HLS-TROJAN DETECTION APPROACH WITH DIFFERENT HARDWARE TROJAN DETECTION APPROACHES IN STATE-OF-THE-ART (SOTA) (NOTE: '×' INDICATES "NOT DETECTABLE")

S. No.	Different Trojan detection techniques	PD-HT	DA-HT	DoS-HT	BE-HT	F-HT	Detection data
1	C to RTL Equivalence checking [3]	×	✓	×	✓	✓	FN>0% | FP >0%
2	TL-HLS (DMR based security-aware scheduling) [4]	×	×	×	×	✓	FN=0% | FP =0%
3	Detection using path delay fingerprint [5]	×	×	×	×	✓	FN=0% | FP =0%
4	HLT based detection [6]	×	×	×	✓	×	FN=0% | FP =0%
5	TMR based HLS detection [7]	×	×	×	×	✓	FN=0% | FP =0%
6	Proposed ML based detection methodology	✓	✓	✓	✓	✓	FN=0% | FP >0%

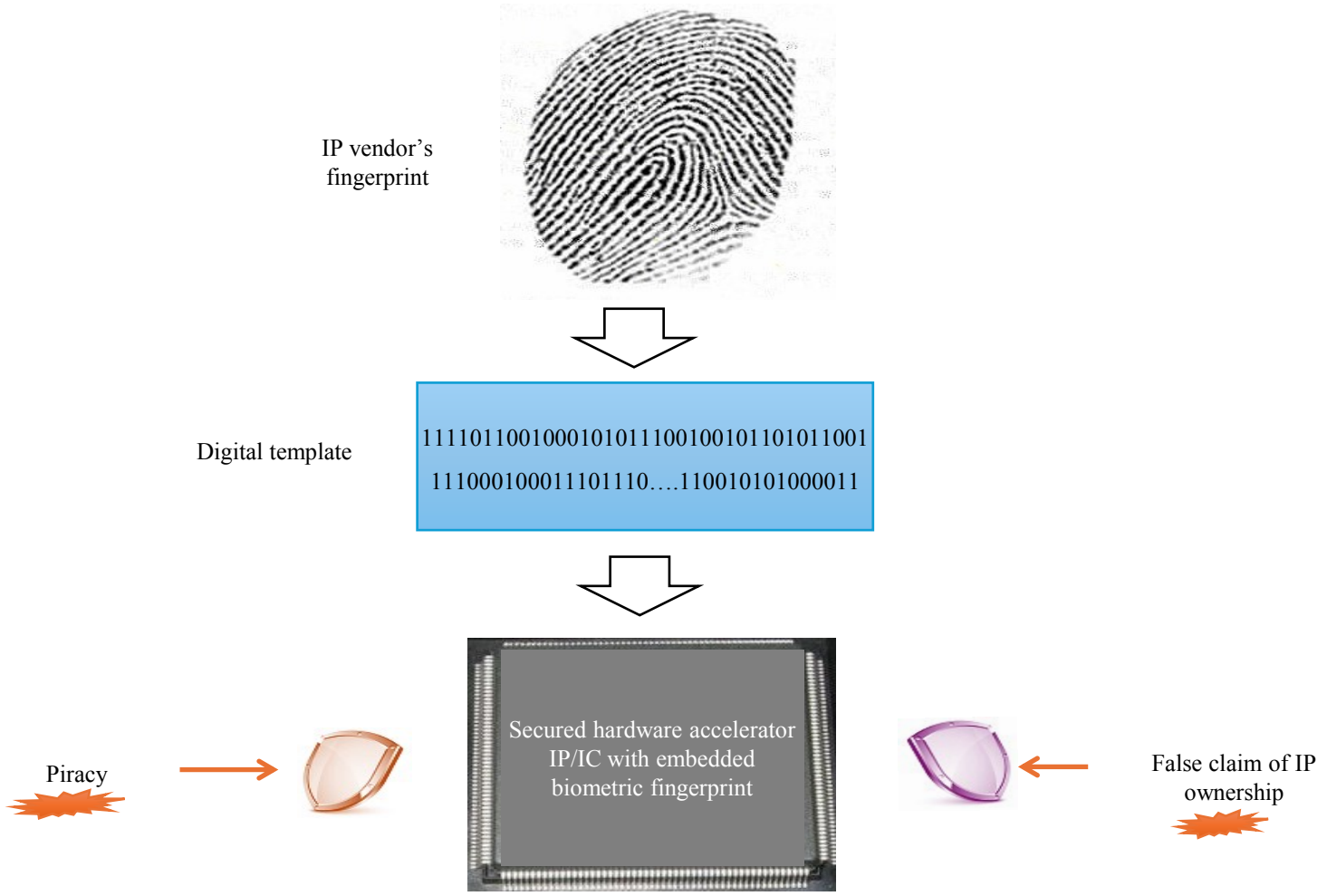
TABLE III
TROJAN DETECTION STATUS ALONG WITH DETECTION TIME

HLS IP designs	HLS Trojan types	Detection Status	Detection time (ms)
CNN convolutional layer IP, FIR Filter IP, DCT IP, IIR Filter IP, and Sharpening Filter IP	PD-HT, DoS-HT, DA-HT,BE-HT, F-HT	Yes	789

- Our feature extraction code is available publicly in [8].
- The HLS-Trojan database consists of several instances corresponding to several benchmarks. The details of the benchmarks including their CDFGs and transfer functions along with created database are publicly available in [8].

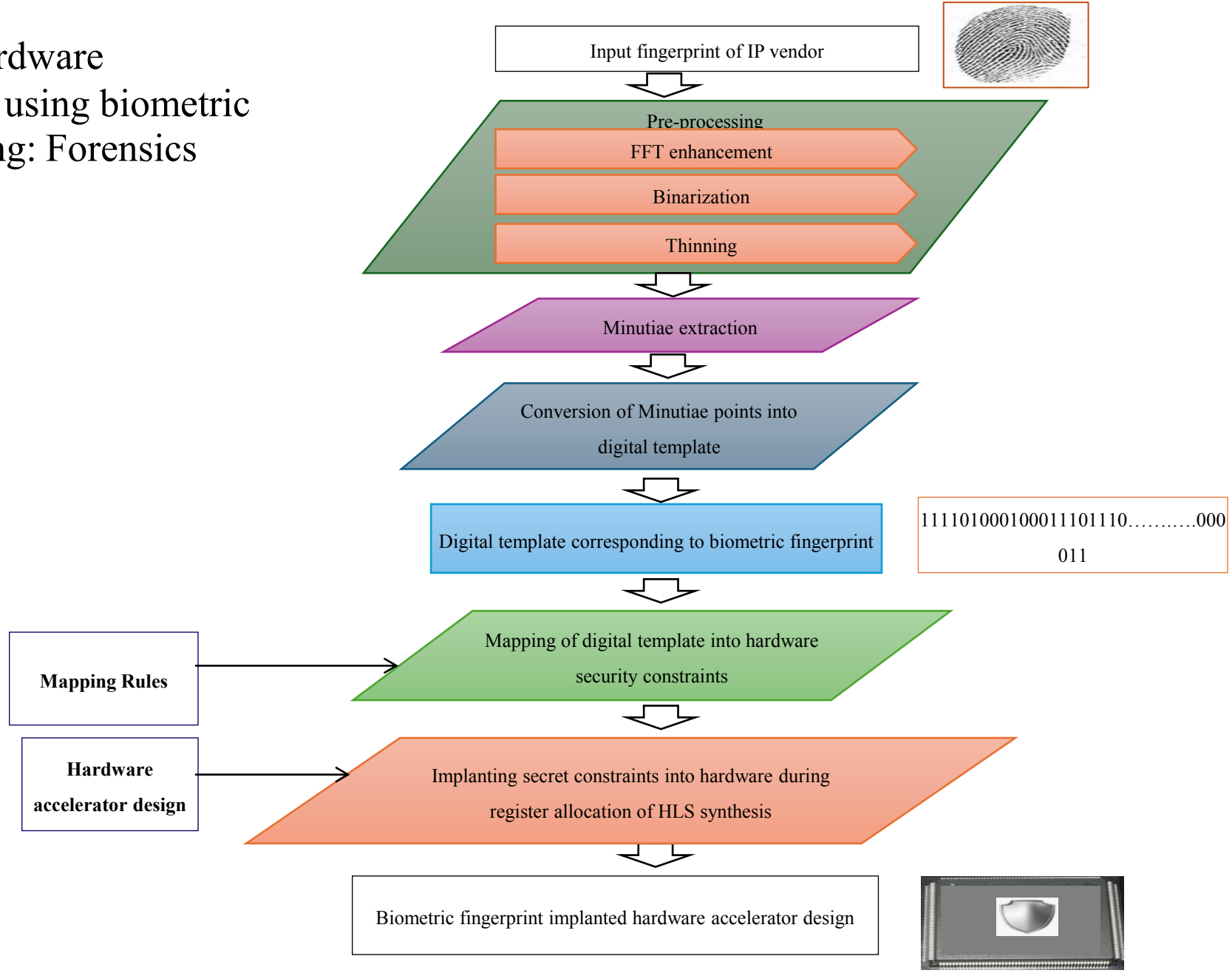
[3] M. Abderehman, R. Gupta, R. R. Theegala and C. Karfa, "BLAST: Belling the Black-Hat High-Level Synthesis Tool," *IEEE Transactions on CAD*, vol. 41, no. 11, pp. 3661-3672, 2022.
[4] A. Sengupta, S. Bhadauria, S. P. Mohanty, "TL-HLS: Methodology for Low Cost Hardware Trojan Security Aware Scheduling With Optimal Loop Unrolling Factor During High Level Synthesis," *IEEE Transactions on CAD*, vol. 36, no. 4, pp. 655-668, April 2017.
[5] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," *IEEE International Workshop on HOST*, 2008, pp. 51–57.
[6] M. Rathor and A. Sengupta, "Revisiting Black-Hat HLS: A Lightweight Countermeasure to HLS-Aided Trojan Attack," *IEEE Embedded Systems Letters*, Volume: 16, Issue: 2, 2024, pp. 170-173.
[7] A. Sengupta, A. Anshul, R. Chaurasia, Exploration of optimal functional Trojan-resistant hardware intellectual property (IP) core designs during high level synthesis, *Elsevier Microprocessors and Microsystems*, Vol. 103, 2023, 104973.
[8] GitHub Repository, Available [Online]: https://github.com/ryderaadi/HLS-Trojan-Database--HLS-TD.

Securing hardware accelerators using biometric fingerprinting: Forensics



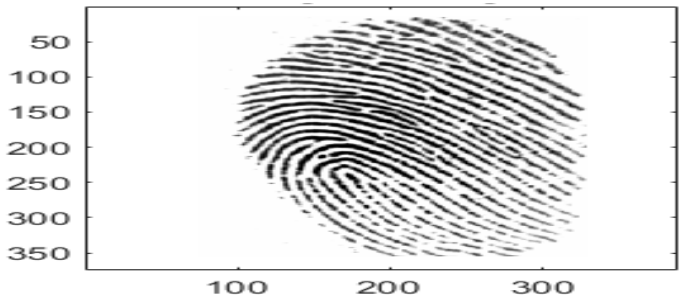
Anirban Sengupta, Mahendra Rathor "Securing Hardware Accelerators for CE Systems using Biometric Fingerprinting", **IEEE Transactions on Very Large Scale Integration Systems (TVLSI)** , Accepted, 2020

Securing hardware accelerators using biometric fingerprinting: Forensics

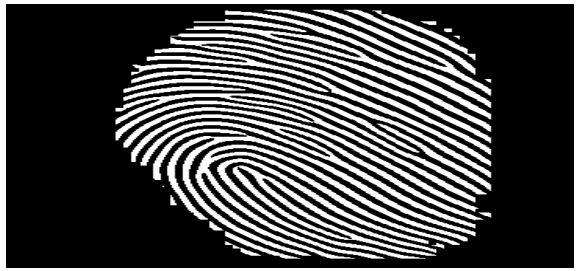


Anirban Sengupta, Mahendra Rathor "Securing Hardware Accelerators for CE Systems using Biometric Fingerprinting", **IEEE Transactions on Very Large Scale Integration Systems (TVLSI)** , Accepted, 2020

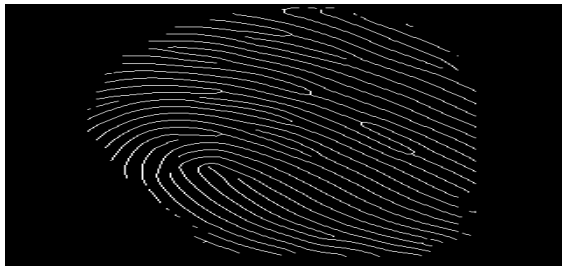
Securing hardware accelerators using biometric fingerprinting: Forensics: An example



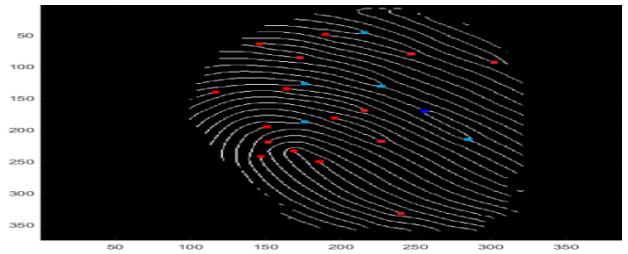
(a) Input fingerprint image (101_1)



(b) Binary image



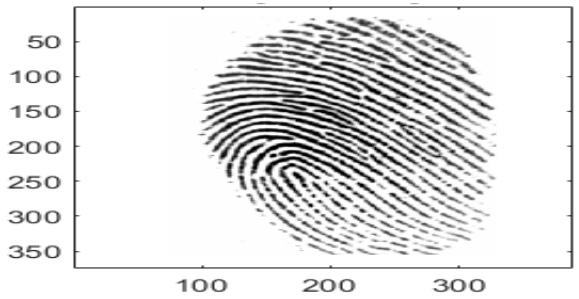
(c) Thinned image



(d) Minutiae points

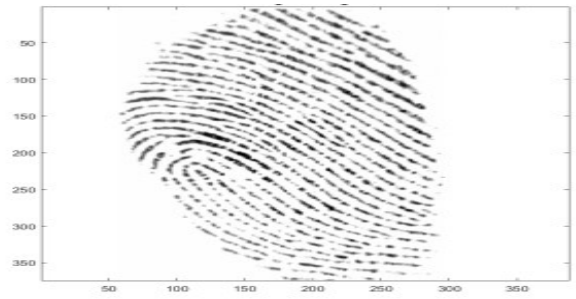
Minutiae points extraction flow (a) Captured fingerprint image (b) Binary fingerprint image post enhancement (c) Fingerprint image post applying thinning (d) Fingerprint image with minutiae points located

Secret biometric constraints for various fingerprint images

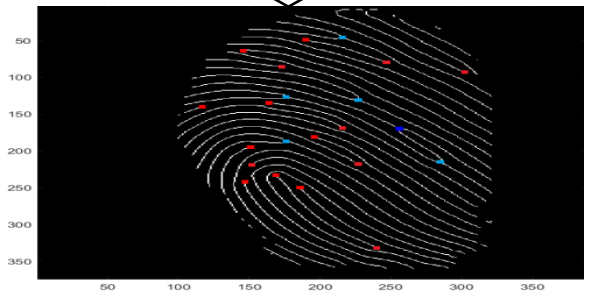


Fingerprint image (101_1)

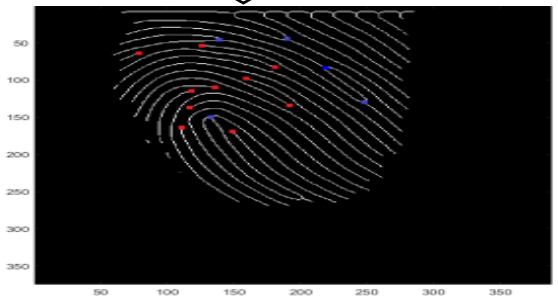
Fingerprint image:101_2



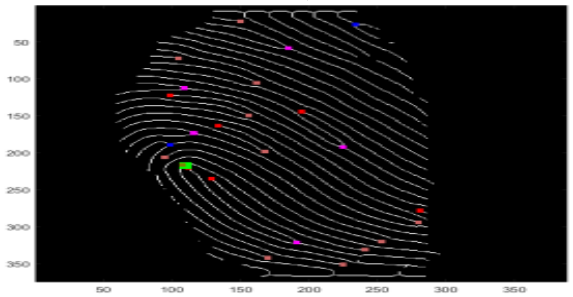
Fingerprint image:101_8



Minutiae points=22



Minutiae points=15



Minutiae points=24

1101100010111011111010111110……..1
01001100111101

Corresponding Digital Template
(Total size =526 bits)

#0s= 224
#1s= 302

Secret biometric constraints for IC/IP

1011111010110111100010100010111…….
.01001111101111

Corresponding Digital Template
(Total size =350 bits)

#0s= 148
#1s= 202

Secret biometric constraints for IC/IP

1001011010110110100111010101101…..
110101111111110

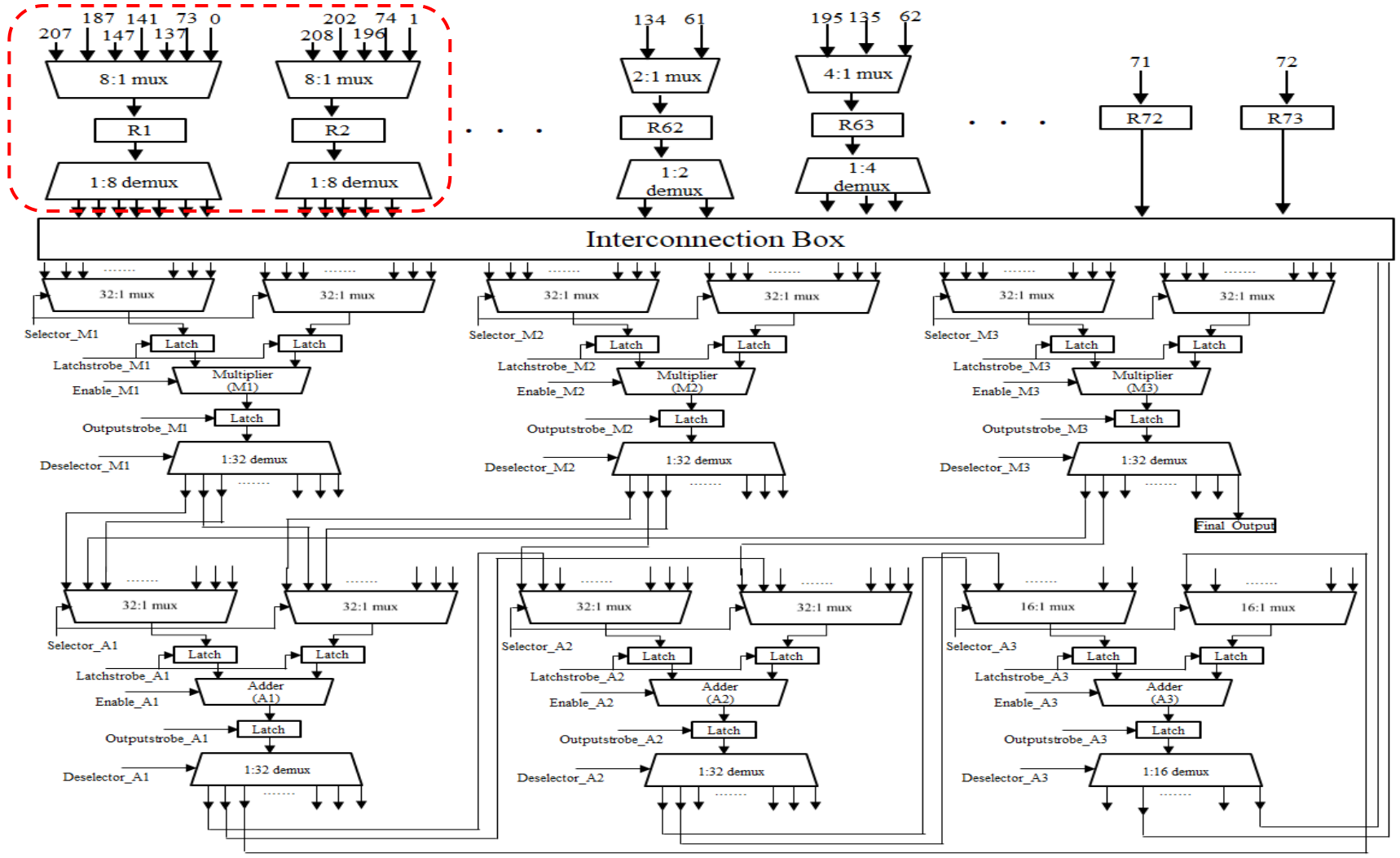
Corresponding Digital Template
(Total size =555 bits)

#0s= 242
#1s= 312

Secret biometric constraints for IC/IP

Anirban Sengupta, Mahendra Rathor "Securing Hardware Accelerators for CE Systems using Biometric Fingerprinting", **IEEE Transactions on Very Large Scale Integration Systems (TVLSI)** , Accepted, 2020

Secured datapath of JPEG compression hardware accelerator implanted with biometric fingerprint



Anirban Sengupta, Mahendra Rathor "Securing Hardware Accelerators for CE Systems using Biometric Fingerprinting", **IEEE Transactions on Very Large Scale Integration Systems (TVLSI)** , Accepted, 2020

Detecting Biometric Fingerprint in a hardware accelerator

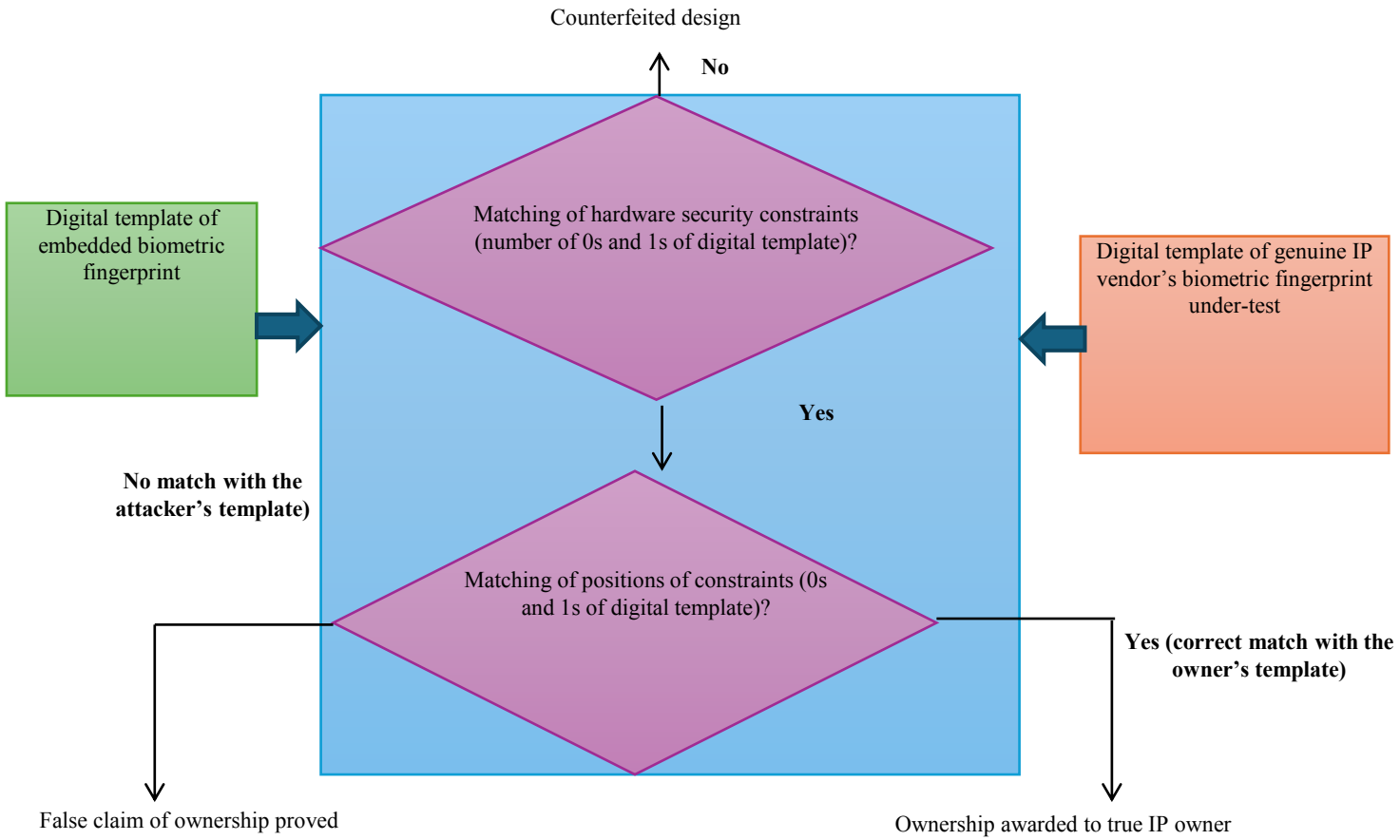
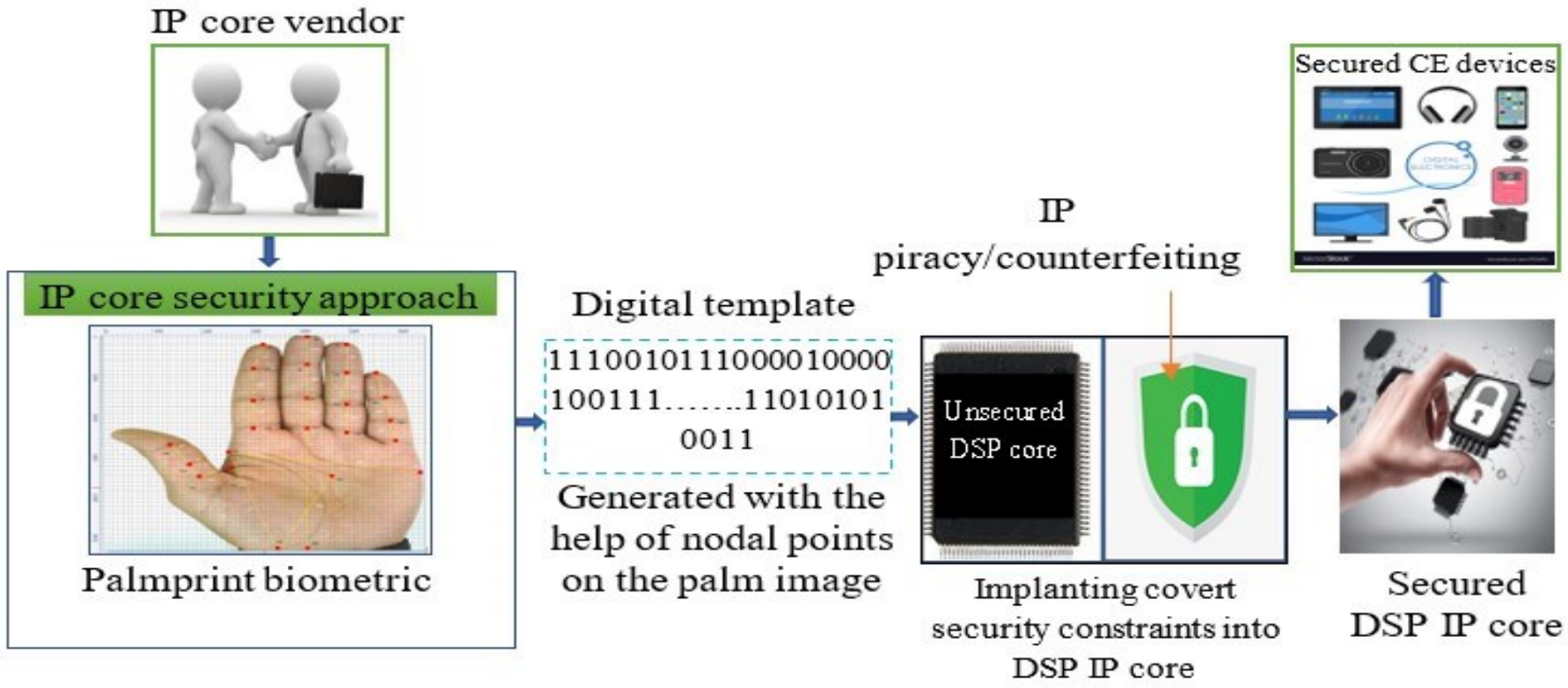


Fig. 9. Proving true IP ownership using proposed detection approach

Anirban Sengupta, Mahendra Rathor "Securing Hardware Accelerators for CE Systems using Biometric Fingerprinting", **IEEE Transactions on Very Large Scale Integration Systems (TVLSI)** , Accepted, 2020

Palmprint based Hardware Security for IPP

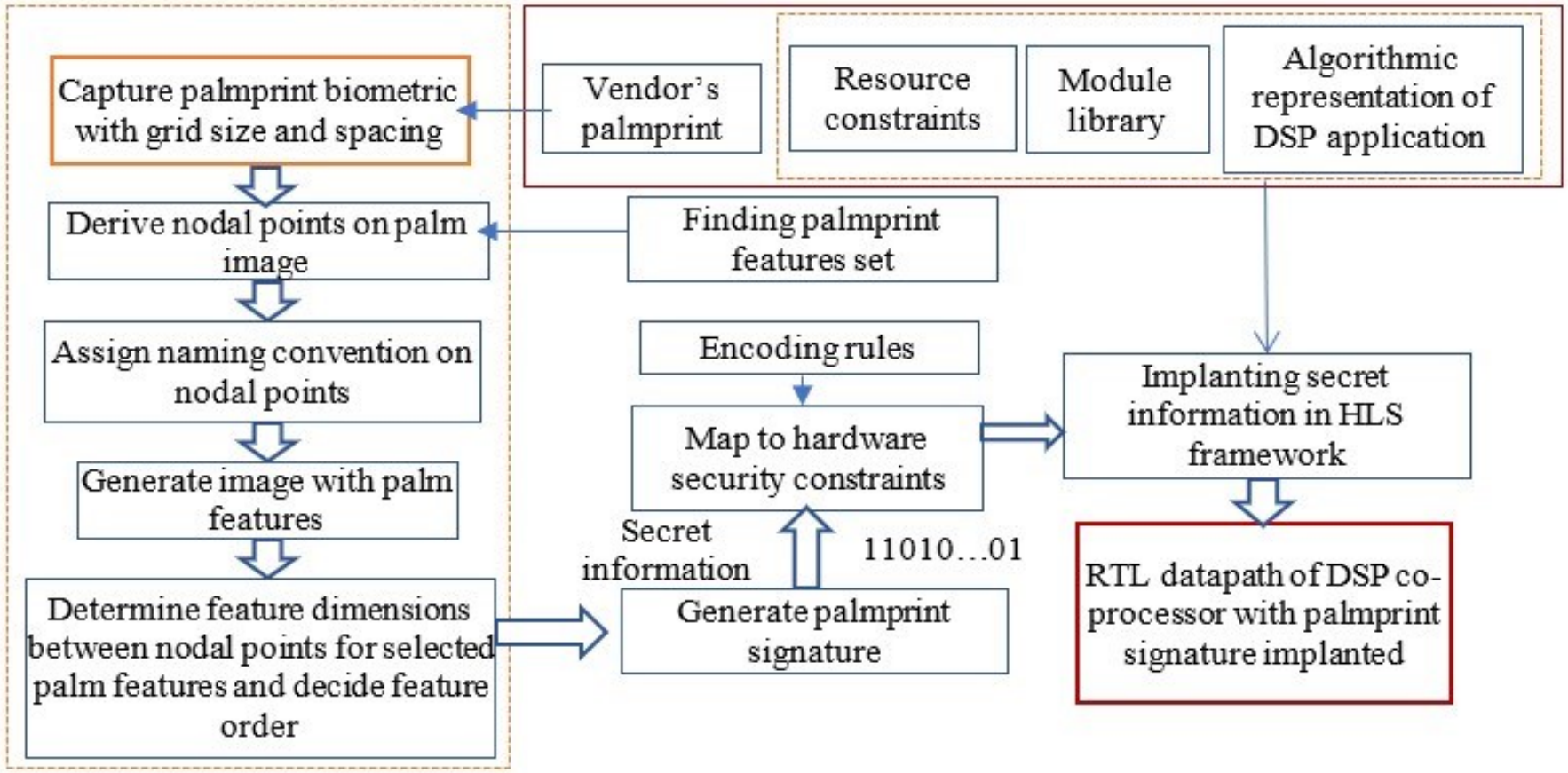


Securing reusable DSP IP core used in CE systems

Anirban Sengupta, Rahul Chaurasia, Tarun Reddy "Contact-less Palmprint Biometric for Securing DSP Coprocessors used in CE systems", **IEEE Transactions on Consumer Electronics (TCE)** , Volume: 67, Issue: 3, August 2021, pp. 202-213

Rahul Chaurasia, Aditya Anshul, Anirban Sengupta "Palmprint Biometric vs Encrypted Hash based Digital Signature for Securing DSP Cores Used in CE systems", **IEEE Consumer Electronics (CEM)** , Volume: 11, Issue: 5, September 2022, pp. 73-80

Palmprint based Hardware Security for IPP



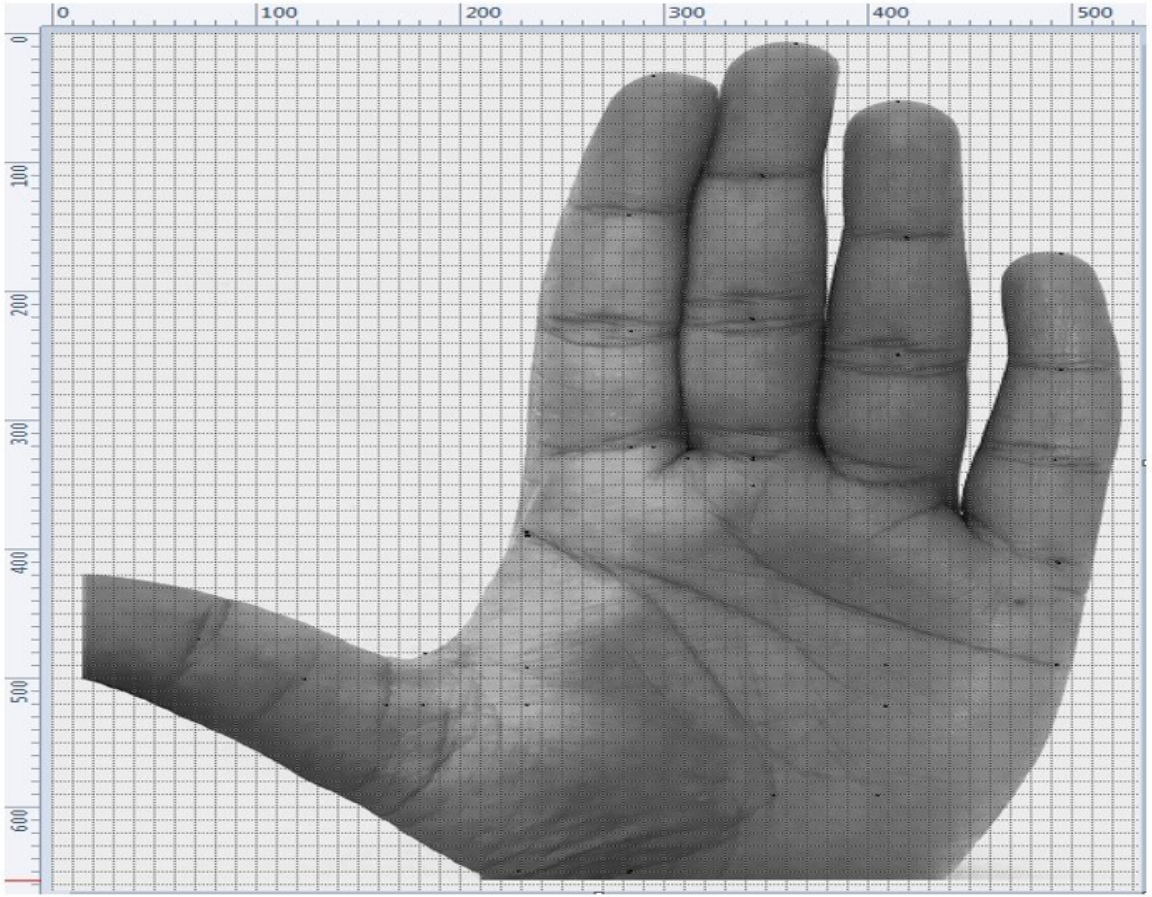
Anirban Sengupta, Rahul Chaurasia, Tarun Reddy "Contact-less Palmprint Biometric for Securing DSP Coprocessors used in CE systems", **IEEE Transactions on Consumer Electronics (TCE)** , Volume: 67, Issue: 3, August 2021, pp. 202-213

Rahul Chaurasia, Aditya Anshul, Anirban Sengupta "Palmprint Biometric vs Encrypted Hash based Digital Signature for Securing DSP Cores Used in CE systems", **IEEE Consumer Electronics (CEM)** , Volume: 11, Issue: 5, September 2022, pp. 73-80

Palmprint based Hardware Security for IPP

- **Capturing palm image**

- At first the palmprint biometric of the authentic vendor or designer is captured and subsequently image of the captured palmprint is subjected to a specific grid size/spacing.
- This helps in generating the nodal points precisely.



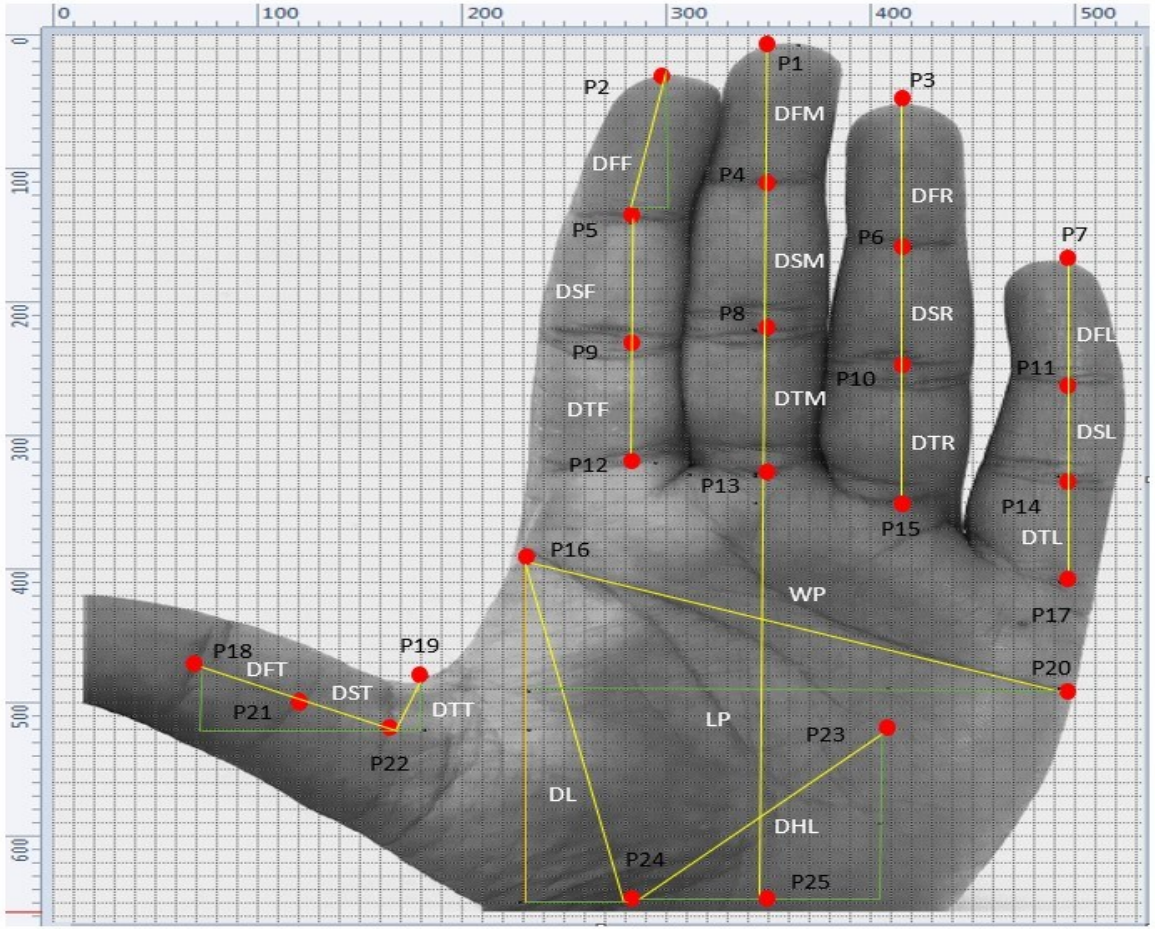
Anirban Sengupta, Rahul Chaurasia, Tarun Reddy "Contact-less Palmprint Biometric for Securing DSP Coprocessors used in CE systems", **IEEE Transactions on Consumer Electronics (TCE)** , Volume: 67, Issue: 3, August 2021, pp. 202-213

Palmprint based Hardware Security for IPP

- **Generating image with chosen palm features and nodal points**

  - Finding Palmprint Feature Set and Deriving Nodal Points for Captured Palmprint Biometric.

  - Assigning Naming Convention and Deriving Palmprint Image with Selected Feature set.



Anirban Sengupta, Rahul Chaurasia, Tarun Reddy "Contact-less Palmprint Biometric for Securing DSP Coprocessors used in CE systems", **IEEE Transactions on Consumer Electronics (TCE)** , Volume: 67, Issue: 3, August 2021, pp. 202-213

Palmprint based Hardware Security for IPP

Feature #	Palmprint feature name	Naming conventions of nodal points	Co-ordinates (x1,y1)- (x2,y2)
F1	Distance between start of life line and end of life line (DL)	(P16) – (P24)	(230, 390)- (285, 650)
F2	Distance between datum points of head line and life line (DHL)	(P23) – (P24)	(405, 520) -(285, 650)
F3	Width of the palm (WP)	(P16) – (P20)	(230, 390)- (495, 490)
F4	Length of palm (LP)	(P13) – (P25)	(350, 325)- (350, 650)
F5	Distance between first consecutive intersection points of forefinger (DFF)	(P2) – (P5)	(300, 30)- (285, 130)
F6	Distance between second consecutive intersection points of forefinger (DSF)	(P5) – (P9)	(285, 130)- (285, 230)
F7	Distance between third consecutive intersection points of forefinger (DTF)	(P9) – (P12)	(285, 230)- (285, 320)
F8	Distance between first consecutive intersection points of middle finger (DFM)	(P1) – (P4)	(350, 5)- (350, 110)
F9	Distance between second consecutive intersection points of middle finger (DSM)	(P4) – (P8)	(350, 110)- (350, 220)
F10	Distance between third consecutive intersection points of middle finger (DTM)	(P8) – (P13)	(350, 220)- (350, 325)
F11	Distance between first consecutive intersection points of ring finger (DFR)	(P3) – (P6)	(415, 50)- (415, 160)
F12	Distance between second consecutive intersection points of ring finger (DSR)	(P6) – (P10)	(415, 160)- (415, 245)
F13	Distance between third consecutive intersection points of ring finger (DTR)	(P10) – (P15)	(415, 245)- (415, 355)
F14	Distance between first consecutive intersection points of little finger (DFL)	(P7) – (P11)	(495, 170)- (495, 265)
F15	Distance between second consecutive intersection points of little finger (DSL)	(P11) – (P14)	(495, 265)- (495, 335)
F16	Distance between third consecutive intersection points of little finger (DTL)	(P14) – (P17)	(495, 335)- (495, 405)
F17	Distance between first consecutive intersection points of thumb finger (DFT)	(P18) – (P21)	(70, 470)- (120, 495)
F18	Distance between second consecutive intersection points of thumb finger (DST)	(P21) – (P22)	(120, 495)- (165, 520)
F19	Distance between starburst point and third intersection point of thumb (DTT)	(P19) – (P22)	(180, 480) -(165, 520)

SELECTED 19 PALMPRINT FEATURES, CORRESPONDING NODAL POINTS AND CO-ORDINATES

Palmprint based Hardware Security for IPP

- **Finding Feature Dimensions and Deriving Palmprint Signature Based on the Selected Feature Order**

- For example, a palmprint signature for the selected order of palmprint features ("DL╪ DHL --- ╪ DTT". Where, '╪' represents the concatenation operator) after concatenation is as follows:

- Palmprint Signature:

  "100001001.1110110000.1110100011110101 11.---.11111"

FEATURE DIMENSION AND CORRESPONDING BINARY REPRESENTATION OF CHOSEN PALMPRINT FEATURES

Feature #	Feature name	Feature dimension	Binary representation
F1	DL	265.75	100001001.11
F2	DHL	176.91	10110000.111010001111010111
F3	WP	283.24	100011011.001111010111000 0101
F4	LP	325	101000101
F5	DFF	101.11	1100101.00011100001010001111
F6	DSF	100	1100100
F7	DTF	90	1011010
F8	DFM	105	1101001
F9	DSM	110	1101110
F10	DTM	105	1101001
F11	DFR	110	1101110
F12	DSR	85	1010101
F13	DTR	110	1101110
F14	DFL	95	1011111
F15	DSL	70	1000110
F16	DTL	70	1000110
F17	DFT	55.90	110111.111001100110011
F18	DST	51.45	110011.011100110011 0011
F19	DTT	42.72	101010.101110000101000 11111

Note: Size of the palmprint signature varies based on the number of chosen palm features by the vendor for signature generation (depending on the required security strength corresponding to target application).

Palmprint based Hardware Security for IPP

➢ **Deriving the Covert Security Constraints and Implanting into Target IP core Design**

- Post obtaining the digital template of palmprint signature, corresponding hardware security constraints are generated based on the encoding rules.
- The encoding rules for the signature bits are as follows:

The bit '1' embeds an edge between node pair (odd-odd), bit '0' embeds an edge between node pair (even-even). Moreover, the binary bit '.' embeds an edge between node pair (0, integer) into the CIG of target DSP design.

- For example, for a sample design having 31 storage variables (T0 to T30) executing through 8 registers (R1 to R8), the generated security constraints corresponding to the zeros are: <T0, T2>, <T0, T4>---<T16, T28>, the security constraints corresponding to ones are: <T1, T3>, -----<T27, T29> and corresponding to the binary points are: <T0, T1>, <T0, T3>, -- -, <T0, T11>.

TABLE I
REGISTER ALLOCATION OF A TARGET HARDWARE IP CORE POST IMPLANTATION

Registers	i0	i1	i2	i3	i4	i5	i6	i7	i8	i9
R1	T0	T8	T17	T24	T25	T26	T27	T28	T29	T30
R2	T1	T9	T16	--	--	--	--	--	--	--
R3	T2	T11	T18	T18	--	--	--	--	--	--
R4	T3	T10	T19	T19	T19	--	--	--	--	--
R5	T4	T4	T13	T20	T20	T20	--	--	--	--
R6	T5	T5	T12	T21	T21	T21	T21	--	--	--
R7	T6	T6	T15	T22	T22	T22	T22	T22	--	--
R8	T7	T7	T14	T23	T23	T23	T23	T23	T23	--
R9	--	T8	T19	T19	T19	--	--	--	--	--
R10	--	T9	--	T24	--	T26	--	T28	--	T30
R11	--	--	T18	T18	T25	--	--	--	--	--
R12	--	--	--	T20	T20	T20	T27	--	--	--
R13	--	--	--	T22	T22	T22	T22	T22	T29	--
R14	--	--	--	T21	T21	T21	T21	--	--	--
R15	--	--	--	T23	T23	T23	T23	T23	T23	--

RESULTS AND DISCUSSION

- The proposed palmprint biometric approach is analyzed in terms of security and design overhead.

Security Analysis:

- The security of the proposed approach is analyzed in terms of probability of coincidence (Pc) and temper tolerance (TT) ability.
- The Pc metric is formulated as follows:

$$\mathrm{Pc} = \left(1 - \frac{1}{\tau}\right)^{S} \qquad (1)$$

- The TT metric is formulated as follows:

$$\mathrm{TT} = P^{Q} \qquad (2)$$

Conclusion

- This talk presents Trojan attacks on ML accelerators and its detection techniques.
- The talk also infers that tackling the problem of security vulnerability in ML accelerators is a wide-open research area for the cybersecurity community.
- The talk also discusses potential countermeasures for HLS Trojan detection.

References

- C. Jiang, D. Ojika, B. Patel and H. Lam, "Optimized FPGA-based Deep Learning Accelerator for Sparse CNN using High Bandwidth Memory," *IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines*, USA, 2021, pp. 157-164.
- Xue, M., Gu, C., Liu, W., Yu, S. and O'Neill, M. (2020), Ten years of hardware Trojans: a survey from the attacker's perspective. *IET Comput. Digit. Tech*., 14: 231-246.
- Why you Need HLS for Machine Learning Accelerators, accessed in 2024, Available: https://resources.sw.siemens.com/en-US/video-why-you-need-hls-for-machine-learning-accelerators.
- N. Gupta, A. Jati and A. Chattopadhyay, AI Attacks AI: Recovering Neural Network architecture from NVDLA using AI-assisted Side Channel Attack, *Cryptology {ePrint} Archive*, Paper 2023/368, 2023, url = https://eprint.iacr.org/2023/368.
- D. Kachave and A. Sengupta, "Digital Processing Core Performance Degradation Due to Hardware Stress Attacks," *IEEE Potentials*, vol. 38, no. 2, pp. 39-45, March-April 2019.
- A. Sengupta, R. Chaurasia, M. Rathor: HLS-based swarm intelligence driven optimized hardware IP core for linear regression-based machine learning, *IET Journal of Engineering*, e12299 (2023).
- A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," *IEEE Embedded Systems Letters*, 2024, doi: 10.1109/LES.2024.3416422.
- S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," International Conference on Engineering and Technology, Turkey, 2017, pp. 1-6.
- A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network Using Facial Biometric," *IEEE Transactions on Consumer Electronics*, vol.68, no. 3, pp. 291-306, 2022.
- M. Abderehman, R. Gupta, R. R. Theegala and C. Karfa, "BLAST: Belling the Black-Hat High-Level Synthesis Tool," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3661-3672, 2022.
- A. Sengupta, S. Bhadauria and S. P. Mohanty, "TL-HLS: Methodology for Low Cost Hardware Trojan Security Aware Scheduling With Optimal Loop Unrolling Factor During High Level Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 655-668, 2017.
- Y. Huang, S. Bhunia and P. Mishra, "Scalable Test Generation for Trojan Detection Using Side Channel Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2746-2760, 2018.
- M. Ludwig, A. -C. Bette and B. Lippmann, "ViTaL: Verifying Trojan-Free Physical Layouts through Hardware Reverse Engineering," *IEEE Physical Assurance and Inspection of Electronics*, USA, 2021, pp. 1-8.
- Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," *IEEE International Workshop on HOST*, 2008, pp. 51–57.
- R. Yasaei, L. Chen, S. -Y. Yu and M. A. A. Faruque, "Hardware Trojan Detection using Graph Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.
- M. Rathor and A. Sengupta, "Revisiting Black-Hat HLS: A Lightweight Countermeasure to HLS-Aided Trojan Attack," *IEEE Embedded Systems Letters,* Volume: 16, Issue: 2, 2024, pp. 170-173.
- C. Pilato, K. Basu, F. Regazzoni and R. Karri, "Black-Hat High-Level Synthesis: Myth or Reality*?," IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 913-926, 2019.
- K. I. Gubbi et al., "Securing AI hardware: Challenges in detecting and mitigating hardware trojans in ML accelerators," *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Tempe, AZ, USA, pp. 821–825, 2023.

Thank You !!!