Revisiting Black-Hat HLS: A Lightweight Countermeasure to HLS-Aided Trojan Attack

IEEE Embedded Systems Letters

Mahendra Rathor and Anirban Sengupta, "Revisiting Black-Hat HLS: A Lightweight Countermeasure to HLS-Aided Trojan Attack," IEEE Embedded Systems Letters, vol. 16, no. 2, pp. 170-173, June 2024.

INTRODUCTION

- The CAD synthesis tools are popular at all the various design phases, viz., high-level synthesis (HLS), logic synthesis and physical or layout synthesis [1], [2].
- In the modern design paradigm of highly complex systems, designers rely on HLS which automatically translates the algorithmic description into a register transfer level (RTL).
- Therefore, the HLS tools are increasingly becoming useful in design houses [3].

RELATED WORK

Sr. No.	Existing Work	Technique Used	Remarks
1.	C. Pilato et. al. [4]., (2019)	insertion of fake/extra operations during the scheduling phase of HLS	However, [4] did not present the insertion of fake operations in such applications which may undergo operation count related optimizations
2.	M. Abderehman et. al., [7] (2022)	Trojan detection approach based on extraction of finite-state machine with datapath (FSMD).	[7] incurs higher detection time due to involving complex approach of FSMD extraction from RTL and its analysis during equivalence checking.

MOTIVATION

- In this letter, authors presents a light weight and efficient detection approach for detecting fake operations inserted by a malicious HLS framework (causing battery exhaustion attack).
- This detection approach is based on the fact that the commercial HLS tools, such as Vivado HLS or academic HLS tools, such as Bambu provide the scheduling information along with the final RTL description as output.
- The fake operations are detected using the scheduling information provided by the HLS, hence not depending on RTL unlike [7].

KEY CONTRIBUTIONS

1) Authors target Trojan detection in a compromised HLS tool by comparing the operation-count from the output of the HLT tool (adopted from [8]) and the scheduled output from the (compromised) HLS tool.

2) The proposed approach also targets to detect such Trojans in applications in which the operation count related optimizations can be applied before scheduling, whereas the related works [4], [5] do not consider such optimizations while inserting Trojan.

3) Authors achieve very low detection time while maintaining the accuracy of detection to be 100%, irrespective of the size/complexity of the designs.

PROPOSED WORK

A. Threat Model

- Trojan insertion in our approach is motivated from the fact that existing functional units (FUs) (which are unused in particular control steps) are reused for executing the Trojan.
- This letter specifically considers the Trojan induced through the insertion of the fake operations in the scheduling phase of HLS process to perform the battery exhaustion attack.

B. Proposed Approach

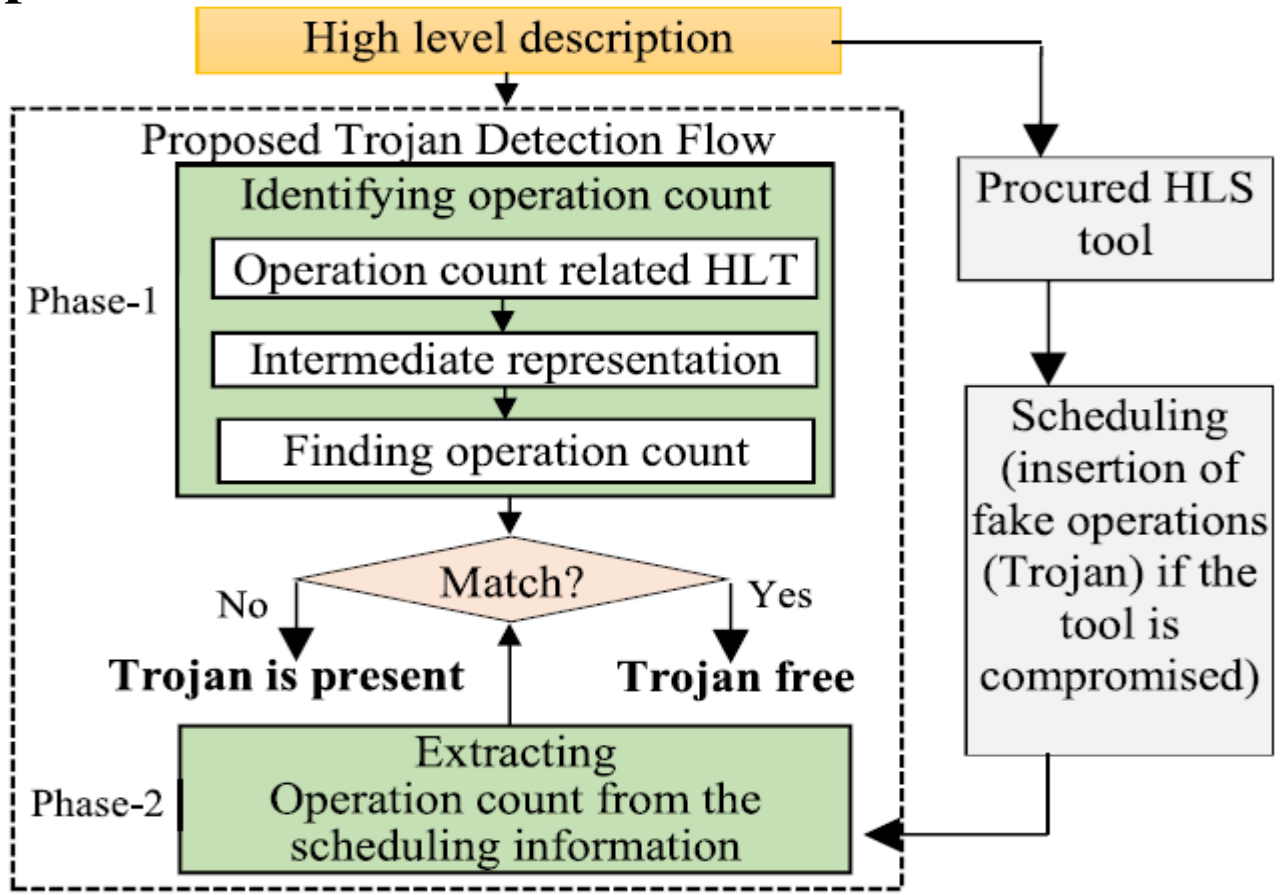


Fig. 1. Proposed high level Trojan detection flow.

C. Demonstration of the Proposed Trojan Detection Approach

- The proposed high level Trojan detection approach is demonstrated for the following two different cases.

1. Detecting trojan in the applications that do not undergo any operation count related transformation
2. Detecting trojan in the applications that may undergo the operation count related transformation

PROPOSED WORK

1. Detecting trojan in the applications that do not undergo any operation count related transformation

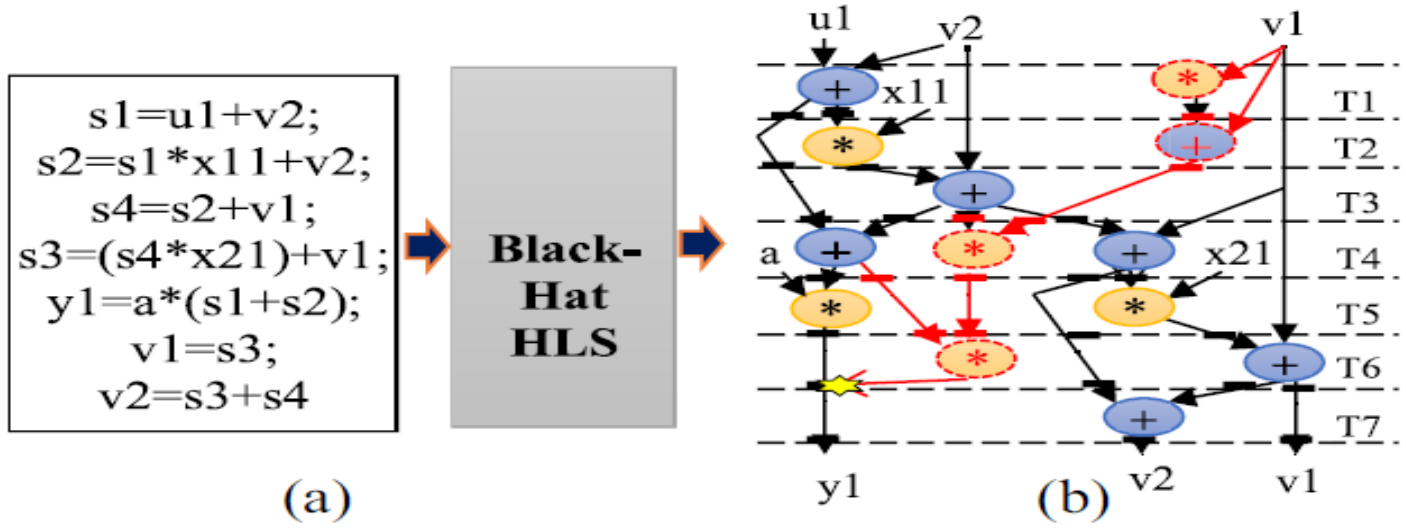


Fig. 2. Demonstration of fake operation insertion by a compromised HLS; where scheduling is performed using two multipliers and two adders and fake operations are shown using red dotted circles.
(a) Algorithmic description of a sample application and
(b) Fake operation insertion by a compromised HLS during scheduling.

PROPOSED WORK

2. Detecting trojan in the applications that may undergo the operation count related transformation

```
for (int i=0; i<2; i+1) {
 Y[i] = h[i]*x[i];
  t=k+d;
  v=3*u;
  Q[i] =k+d+ x[i];
  S[i] = v+Q[i]; }
```

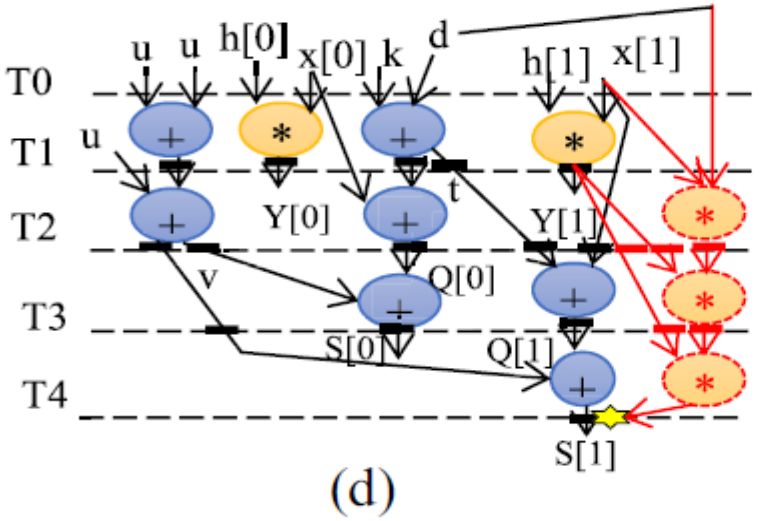
(a)

```
After LT, LICM and ROE:
 t=k+d;
 v=u+u+u;
 for (int i=0; i<2; i+2) {
  Y[i] = h[i]*x[i];
  Q[i] = t + x[i];
  S[i] = v+Q[i]; }
```

(b)

```
After LU:
 t=k+d;
 v=u+u+u;
  Y[0] = h[0]*x[0];
  Q[0] = t + x[0];
  S[0] = v+Q[0];
  Y[1] = h[1]*x[1];
  Q[1] = t + x[1];
  S[1] = v+Q[1];
```

(c)



(d)

Black-Hat HLS

Fig. 3. Demonstration of fake operations insertion in scheduling while considering the HLTs performed by the malicious HLS; where scheduling is performed using two multipliers and two adders.
(a) Algorithmic description of a sample application,
(b) HLT using LT, LICM and ROE,
(c) HLT using LU, and
(d) fake operation insertion (post HLTs) by the compromised HLS during scheduling.

RESULTS AND DISCUSSION

TABLE I
ANALYSIS OF THE EFFECT OF HLTS ON OPERATION COUNT (OPN)

Benchmark	LU	ROE	LT	LICM	# Opn pre-HLTs	# Opn post- HLTs
Test-sample	✓	✓	✓	✓	8	9
JPEG	--	✓	--	--	112	95
ARF	--	✓	--	--	28	23
8-tap FIR	✓	--	--	--	4	15

RESULTS AND DISCUSSION

TABLE II
ANALYSIS OF FAKE OPERATIONS INSERTION BY A COMPROMISED HLS FOR TROJAN DETECTION

Benchmark	Resource constraints	Expected operation count(O_E)	Extracted operation count (O_T) from the scheduling	Trojan detection
Test-sample	2+,2*	9	12	✓
Mesa Horner	1+,2*	14	16	✓
JPEG IDCT	12+,12*	95	103	✓
MPEG	2+,3*	28	33	✓
DWT	1+,2*	17	22	✓
ARF	2+,4*	23	30	✓
8-tap FIR	1+,4*	15	22	✓

RESULTS AND DISCUSSION

Table III
Analysis of the False Positive, False Negative and Detection Time

False Positive	False Negative	Estimated average detection time	
		M. Rathore et. al.	[7]
0	0	~50 ms	~200 ms

RESULTS AND DISCUSSION

TABLE IV
AREA (IN μM 2) AND DELAY (IN PS) OVERHEAD ANALYSIS DUE TO FAKE OPERATION INSERTION BY THE COMPROMISED HLS

Benchmark	Area without Trojan	Area with Trojan	Delay without Trojan	Delay with Trojan
Test-sample	200.25	207.91	667.69	667.69
Mesa Horner	198.62	198.62	2191.54	2191.54
JPEG IDCT	1308.83	1308.83	2334.28	2334.28
MPEG	323.67	323.67	2278.29	2278.29
DWT	213.96	217.79	2849.25	2874.75
ARF	381.91	389.58	1885.83	1936.83
8-tap FIR	341.95	353.45	1488.37	1539.37

RESULTS AND DISCUSSION

Some Important Considerations:

- In order to be sure that the operation-count related transformations introduced by our framework do not modify the number of operations in a different way as the HLS tool, the following cautions should be taken:

1) The transformations are always performed in the same way in the two tools (frontend has to be essentially same).
2) The order of transformations applied can be kept same in the two tools. In this way, we can assure that the number of operations in the HLT and the HLS tool are always the same for the Trojan-free designs.

RESULTS AND DISCUSSION

TABLE V
COMPARISON OF PROPOSED APPROACH AND EXISTING HLS FRAMEWORKS IN TERMS OF OPERATION COUNT POST HLT

HLS tools analysed for transformations	**O_E post LU-HLT for FIR**		**O_E post ROE-HLT for ARF**	**O_E post LT HLT for test-sample**
	I=8 ,UF=8	I=8 ,UF=4		
Proposed approach	15	9	23	9
Vivado HLS	15	9	Did not perform ROE	Did not perform LT
Islam et al. 2021 [9]	15	9		
Sengupta et al. 2018 [10]	Did not perform LU		23	9
Sengupta et al. 2017 [11]	Did not perform LU		23	9
Pilato et al. 2013 [6] (Bambu)	Did not perform LU, ROE, LT			

REFERENCE

[1] I. Polian, G. T. Becker, and F. Regazzoni, “Trojans in early design steps—An emerging threat,” in Proc. TRUDEVICE, 2016,pp. 55:1–55:6.
[2] J. S. Hurtarte, E. A. Wolsheimer, and L. M. Tafoya, Understanding Fabless IC Technology. Amsterdam, The Netherlands: Elsevier, 2007.
[3] C. Pilato, S. Garg, K. Wu, R. Karri, and F. Regazzoni, “Securing hardware accelerators: A new challenge for high-level synthesis,” IEEE Embedded Syst. Lett., vol. 10, no. 3, pp. 77–80, Sep. 2018.
[4] C. Pilato, K. Basu, F. Regazzoni, and R. Karri, “Black-hat high-level synthesis: Myth or reality?” IEEE Trans. Very Large Scale Integr. (VLSI)Syst., vol. 27, no. 4, pp. 913–926, Apr. 2019.
[5] K. Basu et al., “CAD-Base: An attack vector into the electronics supply chain,” ACM Trans. Des. Autom. Electron. Syst., vol. 24, no. 4, pp. 1–30, 2019.
[6] C. Pilato and F. Ferrandi, “Bambu: A modular framework for the highlevel synthesis of memory-intensive applications,” in Proc. FPL, 2013, pp. 1–4.
[7] M. Abderehman, R. Gupta, R. R. Theegala, and C. Karfa, “BLAST: Belling the black-hat high-level synthesis tool,” IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., vol. 41, no. 11, pp. 3661–3672, Nov. 2022.

REFERENCE

[8] A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, “DSP design protection in CE through algorithmic transformation based structural obfuscation,” IEEE Trans. Consum. Electron., vol. 63, no. 4, pp. 467–476, Nov. 2017.
[9] S. A. Islam, L. K. Sah, and S. Katkoori, “High-level synthesis of keyobfuscated RTL IP with design lockout and camouflaging,” ACM Trans. Des. Autom. Electron. Syst., vol. 26, no. 1, pp. 1–35, 2021.
[10] A. Sengupta, S. P. Mohanty, F. Pescador, and P. Corcoran, “Multiphase obfuscation of fault secured DSP designs with enhanced security feature,” IEEE Trans. Consum. Electron., vol. 64, no. 3 pp. 356–364, Aug. 2018.
[11] A. Sengupta and D. Roy, “Protecting an intellectual property core during architectural synthesis using high-level transformation based obfuscation,” IET Electron. Lett., vol. 53, no. 13, pp. 849–851, 2017.
[12] “NanGate 15nm open cell library.” Accessed: Sep. 2022. [Online]. Available: http://www.nangate.com/?pageid=2328

Thank You!