Security Closure of IC Layouts Against Hardware Trojans

Fangzhou Wang[1], Qijing Wang[1], Bangqi Fu[1], Shui Jiang[1], Xiaopeng Zhang[1], Lilas Alrahis[2], Ozgur Sinanoglu[2], Johann Knechtel[2], Tsung-Yi Ho[1], and Evangeline F. Y. Young[1]

[1]The Chinese University of Hong Kong, [2]New York University Abu Dhabi

March 29, 2023

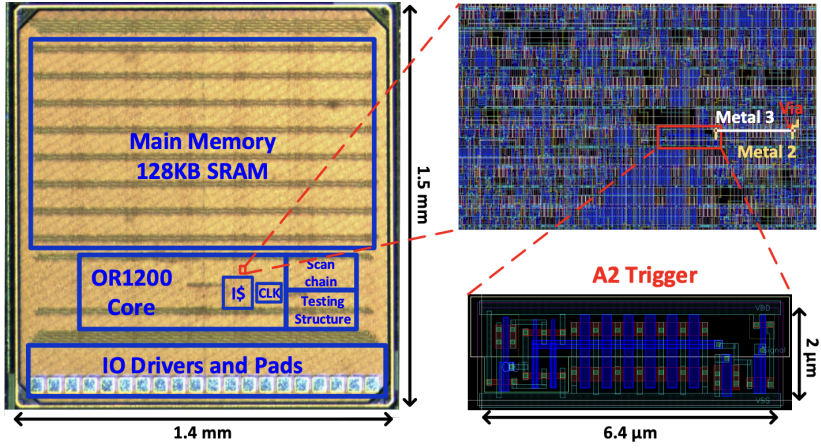
香港中文大學
The Chinese University of Hong Kong

جامعة نيويورك أبوظبي
NYU ABU DHABI

Outline

Outline

Example of Hardware Trojan



An advanced Trojan (A2) Yang et al. 2016.

Hardware Trojans

- Hardware Trojans are malicious hardware modifications that
  - leak information from an IC
  - reduce its performance
  - disrupt its working
- Most Trojans comprise
  - a trigger that activates the payload on some attack condition
    - often based on low-controllability nets (LCNs)
  - a payload that performs an actual attack
    - often targets at sensitive assets like key registers
- Most Trojans require layout-level resources like open placement sites, free routing tracks, and/or available timing slacks.

Proactive Prevention of Trojan Insertion

In this work, we will focus on proactive prevention of hardware Trojans.

- Dummy logics or filler cells to fill the open placement sites? No: can be easily removed
- Buffer insertion or cell shifting? No: can still be easily reversed

Our contributions:

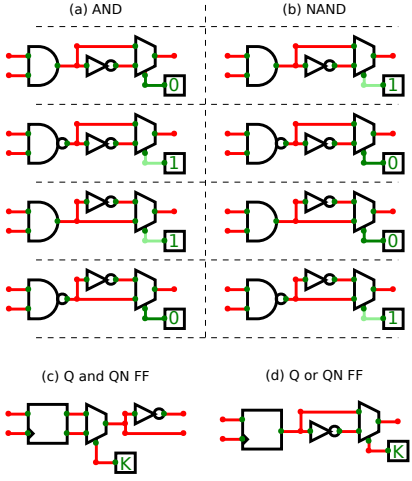
- We propose a logic locking scheme called *TroMUX* to
  - introduce additional logic to reduce the amount of available layout resource
  - obfuscate the design netlist while being resilient to the state-of-the-art machine learning-based attacks.
- To balance security and design performance, we propose a carefully designed physical synthesis flow.

Experimental results show that ours

- can effectively reduce layout-level resources avaialble for Trojan insertion
- is resilient to machine learning-based attacks.

Outline

TroMUX



Design of different TroMUX instances. The key-bit is connected to the MUX select line.

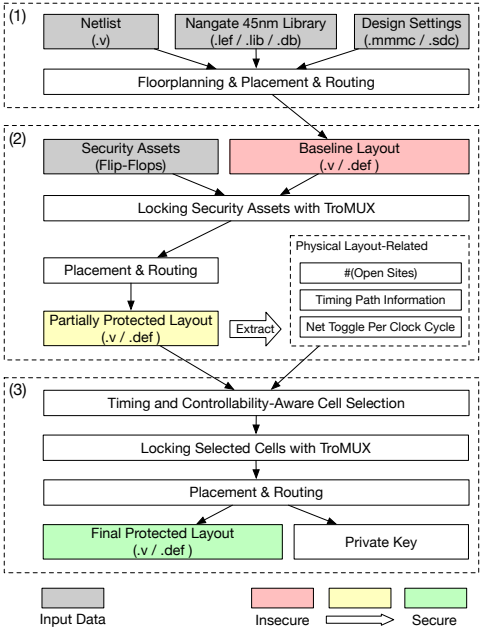
Illustration with example:

- Fig (a, b): Locking of simple AND, NAND gates.
- The four rows show *TroMUX* instances which are pairwise structurally equivalent and thus functionally indistinguishable without knowing the key-bit.
- Fig (c/d): Locking of flip-flops (FFs) with two/one output.

Benefits:

- fills the layout with additional logics
- obfuscates the logic
- is resilient to machine learning-based attacks

Overall Flow



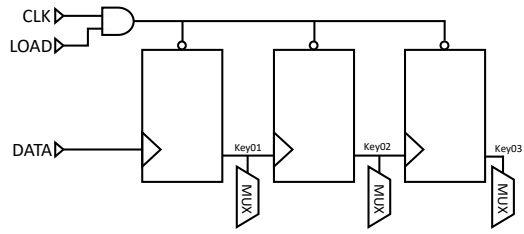
Our physical-synthesis flow consists of three parts:

1. initial synthesis
2. locking of security assets
3. locking considering timing and controllability

Key-bits Storage

We employ a shift register for key-bits storage, with the following advantages:

- only requires two additional primary inputs (DATA, LOAD) for any number of key-bits
- negligible impact on performance and power (after loading keys, it only consumes some static flip-flops power)
- helps to further fill open placement sites



Shift register.

On-Demand Key Length

We determine the additional key length required to fill the physical layout at hand as follows:

$$k = floor\left(\frac{num\_open\_sites}{size(INV) + size(MUX) + size(FF) + \alpha}\right) \quad (1)$$

- $size(INV)$: the size of the smallest INV cell in the library; similar for MUX and FF.
- α: a parameter for timing budget. We use this to reserve some additional space for possible timing optimization.

Cell Selection Considering Timing and Controllability

Scoring function that jointly considers timing and controllability:

$$cellScore(c, N, P) = \sum_{n \in N(c)} netScore(n, P), \tag{2}$$

$$netScore(n, P) = \frac{1}{1 + exp(-2 \cdot MS(n, P))} \cdot \frac{1}{TPC(n) + 10^{-3}}, \tag{3}$$

$$MS(n, P) = \begin{cases} \min\limits_{p \in P(n)} p.slack & \text{, if } |P(n)| > 0, \\ -0.5 & \text{, otherwise,} \end{cases} \tag{4}$$

- $N(c)$: The set of nets driven by c
- $P(n)$: The set of timing paths covering n
- $MS(n, P)$: The minimum slack of paths covering n
- $TPC(n)$: The number of toggles per clock cycle for n

Cell Selection Considering Timing and Controllability

Algorithm Cell Selection Considering Timing and Controllability

Require: Standard Cells C, Nets N, Timing Paths P, Key Length K, Locking Delay σ.
Ensure: The Set of Cells to Lock C'.

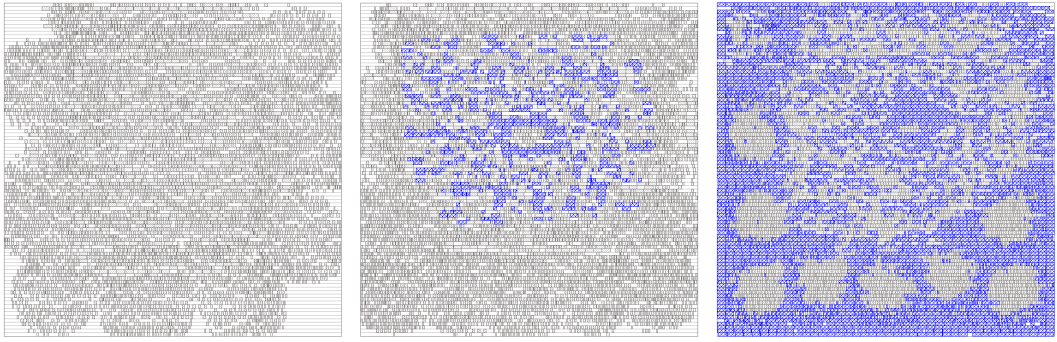
$C' \leftarrow \{\}$;
while $|C'| < K$ **do**
    for all cell $c \in C$ **do**
        $c.score \leftarrow cellScore(c, N, P)$; ▷ Score calculation for each cell
    Let c_h be the cell with the highest score in C;
    $C'.append(c_h)$;
    $C.remove(c_h)$;
    for all net $n \in N(c_h)$ **do** ▷ Pessimistic slack estimation
        for all timing path $p \in P(n)$ **do**
            $p.slack \leftarrow p.slack - \sigma$; ▷ σ equals to the sum of worst-case delays for INV_X1 and MUX2_X1
return C';

Outline

Experiment Settings

- We tested our framework on the benchmark suite from the ISPD'22 contest on security closure.
- Cadence Innovus 20.14 is used for physical synthesis
- Our methodology is implemented in custom TCL scripts and Python code

Visualization



Camellia layout, after initial synthesis (left), locking of security assets (middle), and locking considering timing and controllability (right). The utilization increases from 49.5% to 59.2%, and eventually to 98.9%. Gates introduced by *TroMUX* instances are marked in blue while the other standard cells are filled with grey dots.

Results on the ISPD'22 Contest Benchmarks

Design	Baseline Layout (Resynthesized)						Protected Layout (Final)								
	Utils	#(Open)	TU	WNS	TNS	Power	Utils	#(Open)	Δ(Open)	TU	WNS	TNS	Power	#(LSA)/#(SA)	#(LLCC)/#(LCC)	KL
AES_1	75.4%	43,980	8.7%	0.000	0.000	59.957	96.2%	6,838	-84.5%	11.1%	-0.013	-0.043	60.552	291/291	884/1,389	1,199
AES_2	75.1%	44,420	8.7%	-0.001	-0.003	59.441	96.3%	6,649	-85.0%	10.9%	-0.008	-0.017	61.040	291/291	908/1,431	1,202
AES_3	85.7%	22,129	9.7%	-0.001	-0.002	59.869	96.6%	5,225	-76.4%	11.2%	-0.031	-4.657	62.787	291/291	150/1,310	441
Camellia	49.5%	33,919	9.5%	1.194	0.000	1.233	98.9%	753	-97.8%	13.3%	0.015	0.000	1.488	256/256	724/724	1,271
CAST	49.1%	54,444	9.1%	0.047	0.000	3.136	93.6%	6,879	-87.4%	12.7%	-0.134	-1.455	3.912	192/192	983/994	1,572
MISTY	48.5%	43,359	8.6%	-0.021	-0.037	2.238	94.4%	4,686	-89.2%	12.4%	-0.140	-1.515	2.966	204/204	307/312	1,215
openMSP430_1	49.7%	32,799	6.2%	0.000	0.000	0.375	97.9%	1,389	-95.8%	12.6%	0.000	0.000	0.544	340/340	441/456	1,218
openMSP430_2	70.5%	15,125	7.7%	0.000	0.000	1.239	97.1%	1,497	-90.1%	10.6%	-0.006	-0.015	1.369	334/334	209/696	543
PRESENT	49.8%	6,284	4.4%	6.694	0.000	0.198	98.0%	245	-96.1%	6.8%	4.935	0.000	0.235	80/80	3/3	241
SEED	49.1%	54,444	9.1%	0.047	0.000	3.136	94.3%	6,056	-88.9%	12.7%	-0.182	-1.072	3.908	195/195	979/989	1,612
SPARX	49.9%	69,979	6.3%	2.452	0.000	2.164	98.8%	1,658	-97.6%	14.0%	0.027	0.000	2.822	2,176/2,176	361/375	2,582
TDEA	70.4%	5,559	6.8%	0.049	0.000	1.084	98.4%	304	-94.5%	8.0%	0.027	0.000	1.153	168/168	6/47	214

* Utils: util. after phys. synthesis; #(Open): nr. of open sites; TU: track utilization; WNS (ns): worst negative slack; TNS (ns): total negative slack; Power (mW): total power; Δ(Open): reduction of nr. of open sites; SA: security assets; LSA: locked security assets; LCC: low-controllable cells; LLCC: locked low-controllable cells; KL: key length (bits).

- We are able to reduce open placement sites by 90.3% on average.
- Higher track utilization is achieved.
- All security assets are being locked and LCNs are well protected in most layouts.

ML-Based Attack Results on Different Locking Schemes

Design	KL	*SCOPE* Alaql et al. 2021						*MuxLink* Alrahis et al. 2022							
		COPE (%)	*TroMUX* Key 1		*TroMUX* Key 2		#(X)	*TroMUX*				*D-MUX* Alrahis et al. 2022; Šišejković et al. 2022			
			AC (%)	KPA (%)	AC (%)	KPA (%)		AC (%)	PC (%)	KPA (%)	#(X)	AC (%)	PC (%)	KPA (%)	#(X)
AES_1	1,199	0.1919	28.86	48.39	30.78	51.61	484	28.77	71.48	50.22	512	78.15	79.90	79.54	21
AES_2	1,202	0.1864	31.53	52.71	28.29	47.29	483	28.12	75.04	52.98	564	80.45	81.70	81.47	15
AES_3	441	0.1075	23.36	48.13	25.17	51.87	227	8.16	90.02	45.00	361	88.44	89.12	89.04	3
Camellia	1,271	0.0240	14.16	51.72	13.22	48.28	923	26.12	73.80	49.92	606	90.64	92.60	92.46	25
CAST	1,572	0.0359	18.58	50.52	18.19	49.48	994	37.47	66.03	52.45	449	93.74	94.32	94.29	7
MISTY	1,215	0.1036	23.70	48.90	24.77	51.10	626	33.33	65.84	49.39	395	97.84	98.03	98.02	3
openMSP430_1	1,218	0.0432	19.87	49.90	19.95	50.10	733	23.56	76.60	50.17	646	NA	NA	NA	NA
openMSP430_2	543	0.0306	29.28	52.13	26.89	47.87	238	9.39	90.42	49.51	440	66.85	69.98	69.01	17
PRESENT	241	0.0434	2.49	42.86	3.32	57.14	227	11.62	95.02	70.00	201	NA	NA	NA	NA
SEED	1,612	0.0343	18.55	49.10	19.23	50.90	1,003	37.03	64.02	50.72	435	97.33	97.70	97.70	6
SPARX	2,582	0.0176	21.42	50.23	21.22	49.77	1,481	5.38	94.93	51.48	2,312	NA	NA	NA	NA
TDEA	214	0.0001	0.47	100.00	0.00	0.00	213	1.87	99.07	66.67	208	NA	NA	NA	NA
Avg.	-	0.0682	19.35	53.72	19.25	46.28	-	20.90	80.19	53.21	-	86.68	87.92	87.69	-

* AC: accuracy; PC: precision; KPA: key prediction accuracy; COPE: COnstant Prop. Effect; #(X): nr. of undeciphered key-bits; NA: Locking using the script in Alrahis et al. 2022 fails.

- On average, *SCOPE* and *MuxLink* can only correctly predict 19.35%/19.25% and 20.90% of the key-bits, respectively, with around 50% key-prediction accuracy for both.

Outline

Conclusion

In this work, we present a MUX-based locking scheme along with a carefully tuned physical-synthesis flow, for preventing targeted Trojan insertion.

- We systematically employ locking and layout-level means in unison for Trojan prevention.
- Experimental results show that ours can secure the design with controllable overhead.
- Attempt to remove TroMUX instances will fail as we demonstrate the superior resilience of ours against ML-based attacks.

Thank You!

References

- Alaql, A. et al. (2021). “SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking”. In: *Trans. VLSI* 29.8, pp. 1529–1542.
- Alrahis, L. et al. (2022). “MuxLink: Circumventing Learning-Resilient MUX-Locking Using Graph Neural Network-based Link Prediction”. In: *Proc. DATE*, pp. 694–699.
- Šišejković, D., F. Merchant, L. M. Reimann, and R. Leupers (2022). “Deceptive Logic Locking for Hardware Integrity Protection Against Machine Learning Attacks”. In: *TCAD* 41.6, pp. 1716–1729.
- Yang, K., M. Hicks, Q. Dong, T. Austin, and D. Sylvester (2016). “A2: Analog Malicious Hardware”. In: *Proc. SP*, pp. 18–37.