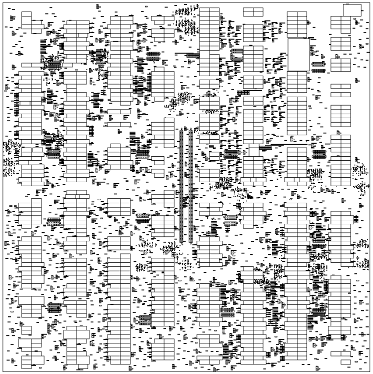
A Yosys plugin for logic locking

Gabriel Gouvine – Coloquinte

LIP6
gabriel.gouvine_moosic@m4x.org

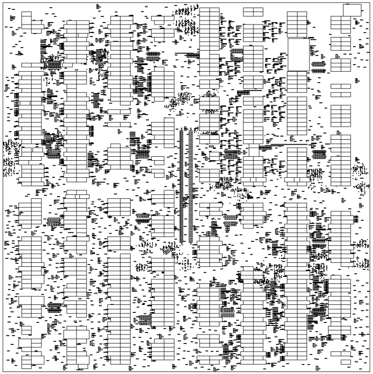
July 11, 2023

EDA Projects



Coloquinte: Coriolis placement tool

EDA Projects



Coloquinte: Coriolis placement tool

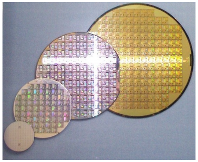
Moosic: This presentation

- Presented last year by Roselyne Chotin and Lilia Zaourar
- This year: integration in Yosys

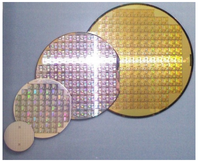
FSiC2022
Free Silicon Conference

Supply-chain security for integrated circuits

Most circuit conception is fabless



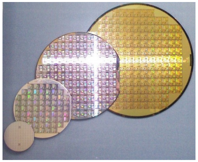
Supply-chain security for integrated circuits



Most circuit conception is fabless

You trust the fab and tools not to:

Supply-chain security for integrated circuits

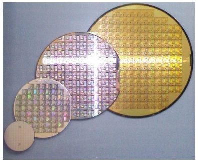


Most circuit conception is fabless

You trust the fab and tools not to:

- introduce trojans

Supply-chain security for integrated circuits

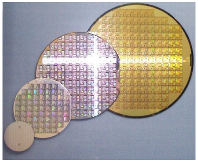


Most circuit conception is fabless

You trust the fab and tools not to:

- introduce trojans
- reuse your design

Supply-chain security for integrated circuits



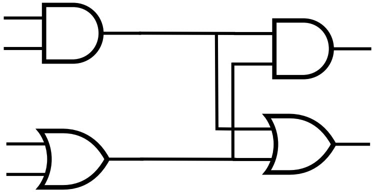
Most circuit conception is fabless

You trust the fab and tools not to:

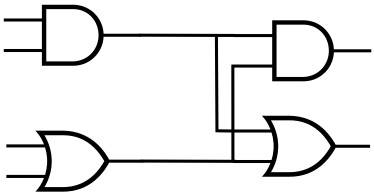
- introduce trojans
- reuse your design

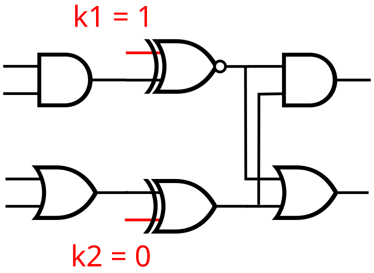
How?

Logic locking example



Logic locking example





Logic locking

Add logic that doesn't work without the right key

Simple method: Xor/Xnor gate insertion

Effect:

Logic locking

Add logic that doesn't work without the right key

Simple method: Xor/Xnor gate insertion

Effect:

- Mangling $\Rightarrow$ mitigates trojans

Logic locking

Add logic that doesn't work without the right key

Simple method: Xor/Xnor gate insertion

Effect:

- Mangling ⇒ mitigates trojans
- Locking ⇒ prevents reuse

Logic locking

Add logic that doesn't work without the right key

Simple method: Xor/Xnor gate insertion

Effect:

- Mangling ⇒ mitigates trojans
- Locking ⇒ prevents reuse

Can we attack it?

Guessing the key: structural approaches

By default, keys are easy to find:

- Xor $\rightarrow$ 0 key
- Xnor $\rightarrow$ 1 key

Defense:

Guessing the key: structural approaches

By default, keys are easy to find:

- Xor $\rightarrow$ 0 key
- Xnor $\rightarrow$ 1 key

Defense:

- Resynthesis (merge inverters)

Guessing the key: structural approaches

By default, keys are easy to find:

- Xor $\rightarrow$ 0 key
- Xnor $\rightarrow$ 1 key

Defense:

- Resynthesis (merge inverters)
- Complex locking (Mux, LUT, ...)

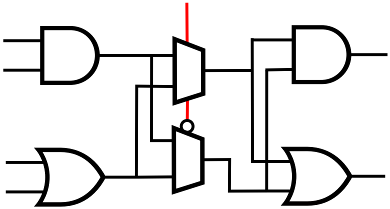
Guessing the key: structural approaches

By default, keys are easy to find:

- Xor → 0 key
- Xnor → 1 key

Defense:

- Resynthesis (merge inverters)
- Complex locking (Mux, LUT, ...)



Guessing the key: structural approaches

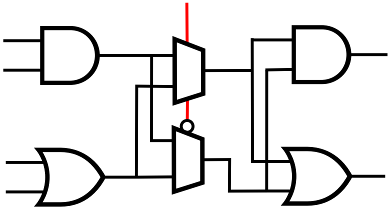
By default, keys are easy to find:

- Xor → 0 key
- Xnor → 1 key

Defense:

- Resynthesis (merge inverters)
- Complex locking (Mux, LUT, ...)

ML attacks may still break those



Finding the key: SAT attack

Expected behaviour + logic circuit

Finding the key: SAT attack

Expected behaviour + logic circuit
⇒ SAT problem

Finding the key: SAT attack

Expected behaviour + logic circuit
$\Rightarrow$ SAT problem

Defense:

Finding the key: SAT attack

Expected behaviour + logic circuit
⇒ SAT problem

Defense:

- More complex locking

Finding the key: SAT attack

Expected behaviour + logic circuit
$\Rightarrow$ SAT problem

Defense:

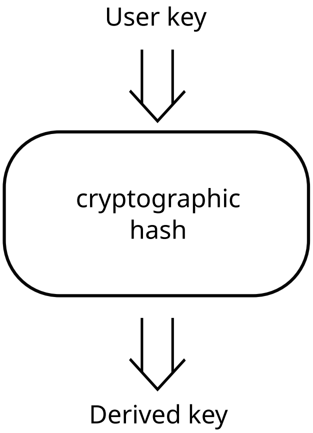
- More complex locking
- Better choice of locked signals

Finding the key: SAT attack

Expected behaviour + logic circuit
⇒ SAT problem

Defense:

- More complex locking
- Better choice of locked signals
- Derived keys (crypto...)



What makes a good logic locking?

Hard to guess the key

Disrupts circuit functionality

Mixed with the logic

Metrics

Metrics

Output corruption: the wrong key changes many output values

Metrics

Output corruption: the wrong key changes many output values

Pairwise security: key bits cannot be silenced individually

Why a Yosys plugin

Why a Yosys plugin

Open-source existing research

Why a Yosys plugin

Open-source existing research

Large ecosystem

Why a Yosys plugin

Open-source existing research

Large ecosystem

Easy to install and integrate

Plugin functionalities

Xor- and Mux- based logic locking

Automation of Xor-based logic locking (metrics)

Design space exploration (area vs security)

Not included

Not included

Key handling left to the user

- Too HW-dependent (memory, boot, scan-chain...)
- Linked to crypto primitives

Not included

Key handling left to the user

- Too HW-dependent (memory, boot, scan-chain...)
- Linked to crypto primitives

No attack methods (ML or SAT)

Not included

Key handling left to the user

- Too HW-dependent (memory, boot, scan-chain...)
- Linked to crypto primitives

No attack methods (ML or SAT)

Only flat modules

In practice

Number of signals to lock

Number of test vectors

Metrics

Design space exploration?

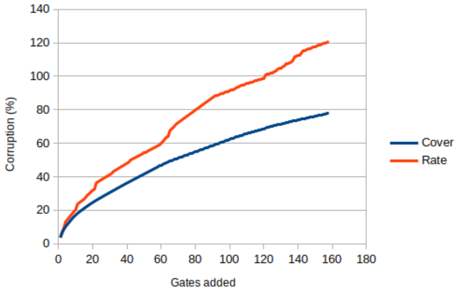
In practice

Number of signals to lock

Number of test vectors

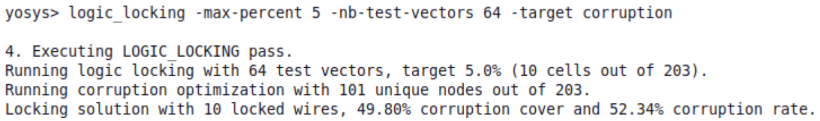
Metrics

Design space exploration?



In practice

In practice



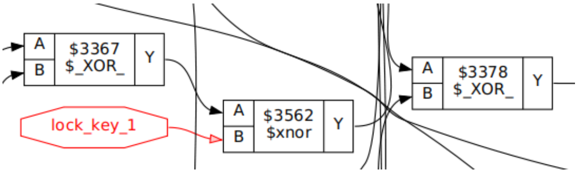
```
yosys> logic_locking -max-percent 5 -nb-test-vectors 64 -target corruption

4. Executing LOGIC_LOCKING pass.
Running logic locking with 64 test vectors, target 5.0% (10 cells out of 203).
Running corruption optimization with 101 unique nodes out of 203.
Locking solution with 10 locked wires, 49.80% corruption cover and 52.34% corruption rate.
```

In practice

```
yosys> logic_locking -max-percent 5 -nb-test-vectors 64 -target corruption

4. Executing LOGIC_LOCKING pass.
Running logic locking with 64 test vectors, target 5.0% (10 cells out of 203).
Running corruption optimization with 101 unique nodes out of 203.
Locking solution with 10 locked wires, 49.80% corruption cover and 52.34% corruption rate.
```



Next steps

Connect with users:

Research: metrics implementation and evaluation

Thank you

https://github.com/Coloquinte/moosic-yosys-plugin