POLITECNICO
MILANO 1863

Protecting data and intellectual property in accelerator-rich architectures with high-level methods

Christian Pilato

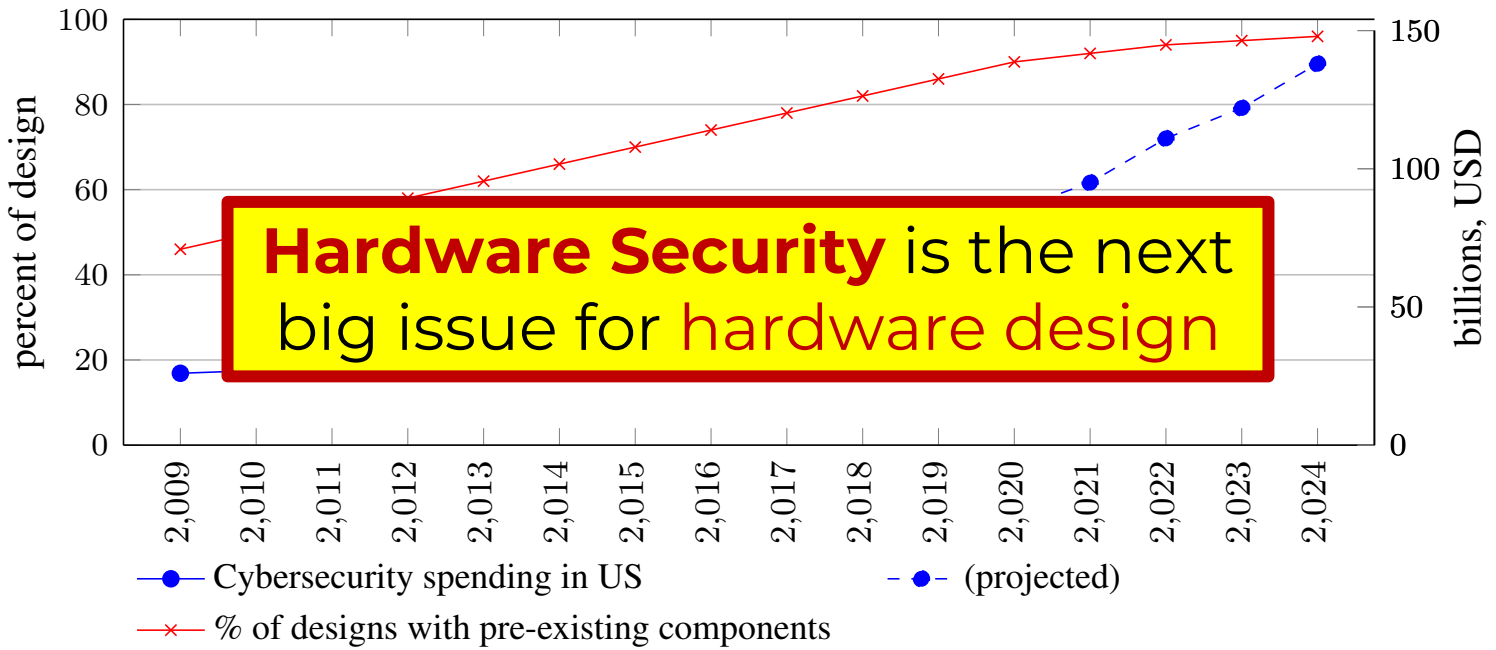
Dipartimento di Elettronica, Informazione e Bioingegneria

christian.pilato@polimi.it

System Complexity and Hardware Security

Increasing system complexity demands **design & reuse approaches**

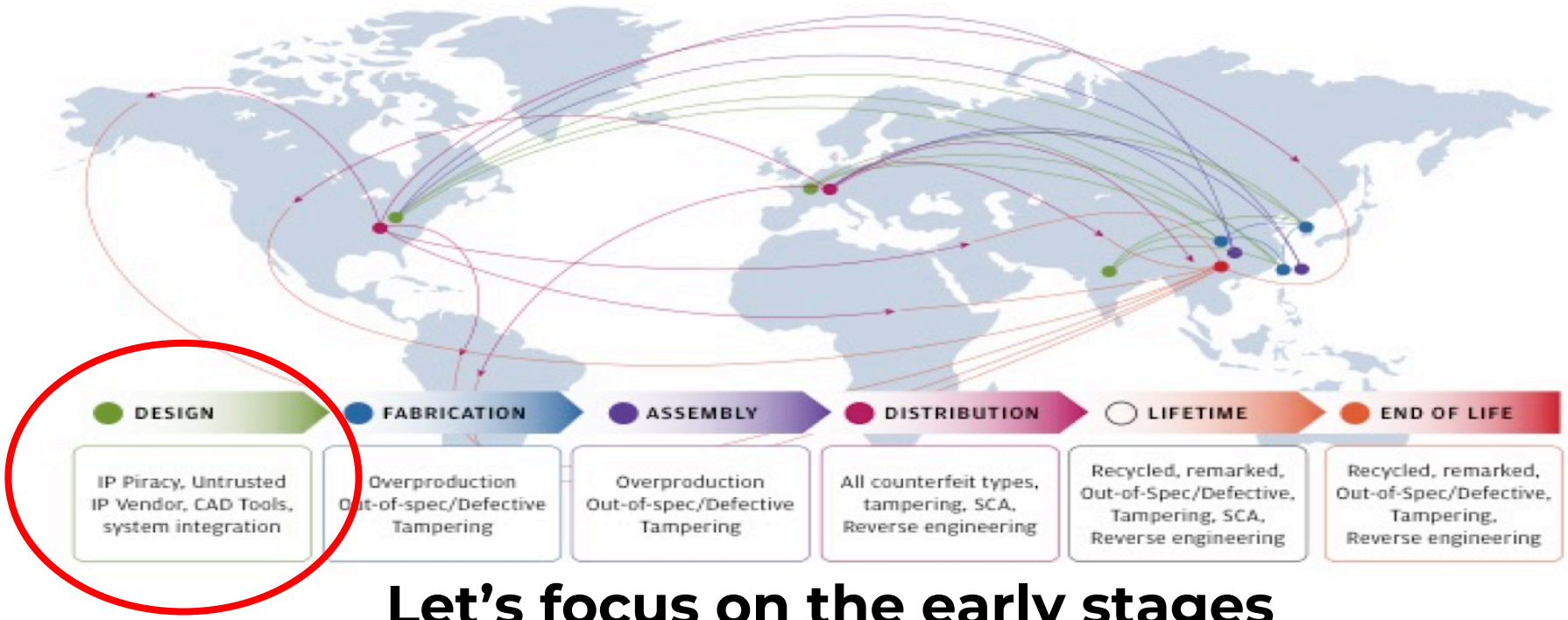
- IP components may be coming from many vendors
- Designers need to assemble to create the SoC
- Most of the design houses are becoming fab-less



Globalization of the Supply Chain

Supply chain is more and more distributed to **reduce costs**

- **Many security threats**
- Cost of addressing them is exponentially increasing from level to level



Let's focus on the early stages of the design process...

What (and How) to Protect?

How Sensitive Data is Elaborated by the System-on-Chip Architectures

Analysis of the data elaboration to identify the hardware modifications to improve the overall security (prevent also software-based attacks)

Intellectual Property in the Design of Components and Architectures

Analysis of the digital design (component or architecture) to apply security protection methods against IP theft and counterfeiting

Let's Raise the Abstraction Level for Hardware Security

Hardware Threats

Data Injection

Injection of **spurious data** to exploit software or hardware/software vulnerabilities

- Buffer overflow attacks
- Memory corruption
- …

Side Channel

Methods to create additional communication channels to **steal sensitive data**

- Differential power analysis for key extraction
- Timing channels for reverse enginnering
- …

Reverse Engineering and IP Theft

Methods to **extract chip functionality** from circuit designs in order to create illegal copies

- Steal the technology
- Cut design costs
- Enter into a market
- …

Hardware Trojans

Malicious modifications of an existing chip design to introduce an additional functionality

- Steal data (e.g., through side channels)
- Harm the normal operations of the chips (e.g., DoS attacks)
- Altern the chip functionality (e.g., errors)
- …

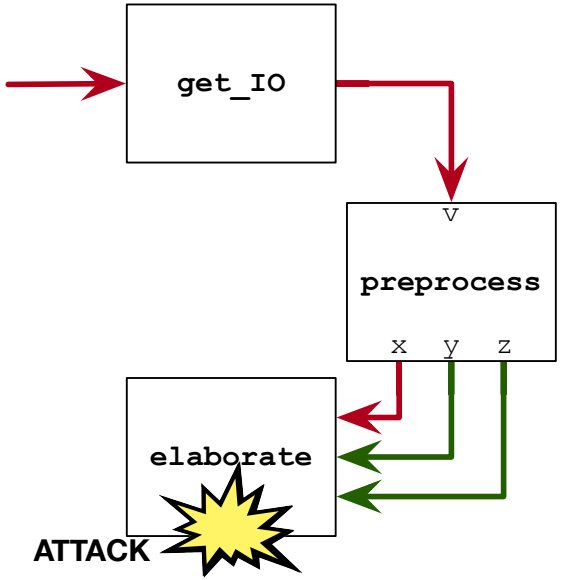
Data Protection & Information Flow Tracking

Marking Data coming for untrusted sources with tags (**taints**)

- Trap to OS if tainted data are used in critical operations
  - Pointer dereference, jump address, modified code or data, …

```
void preprocess (int v){
  struct results ret;
  if (v > 0)
    ret.x = 1;
  else
    ret.x = 4;
  ret.y = 10;
  ret.z = 5;
  return ret;
}
```

```
struct results
{
  int x;
  int y;
  int z;
};
...
get_IO(&v);
...
ret = preprocess(v);
...
elaborate(ret);
```

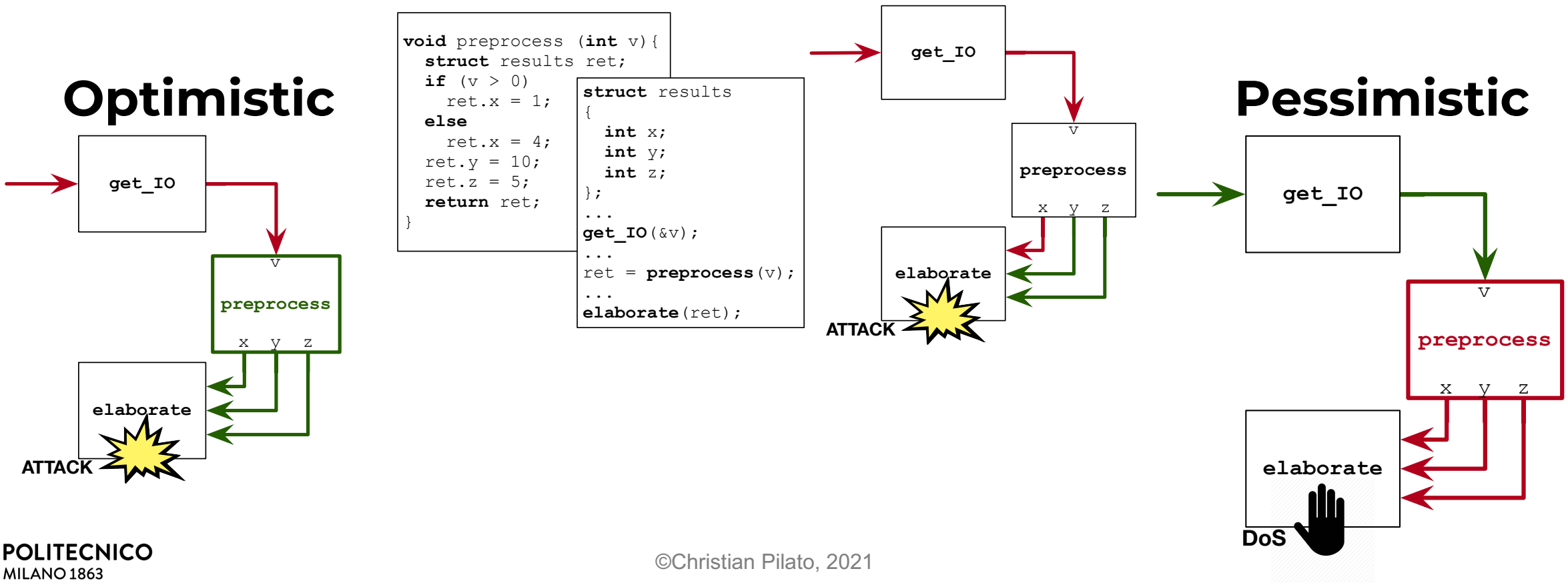


DIFT can protect from several **software-based attacks**

DIFT in Heterogeneous Architectures

Applications interleave tasks between hardware and software

- What happens when accelerators are executed *before* the potential attack point?



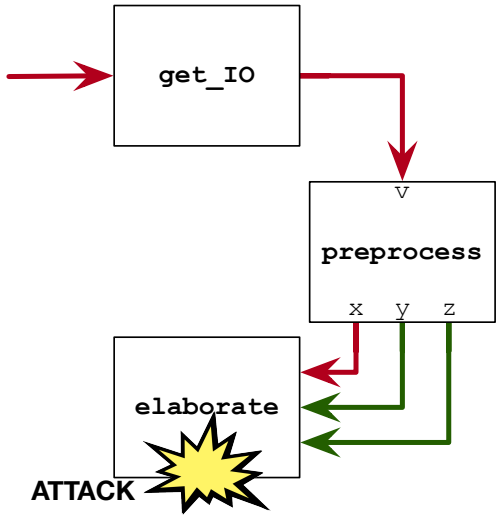
DIFT in Heterogeneous Architectures

Applications interleaves tasks in both hardware and software

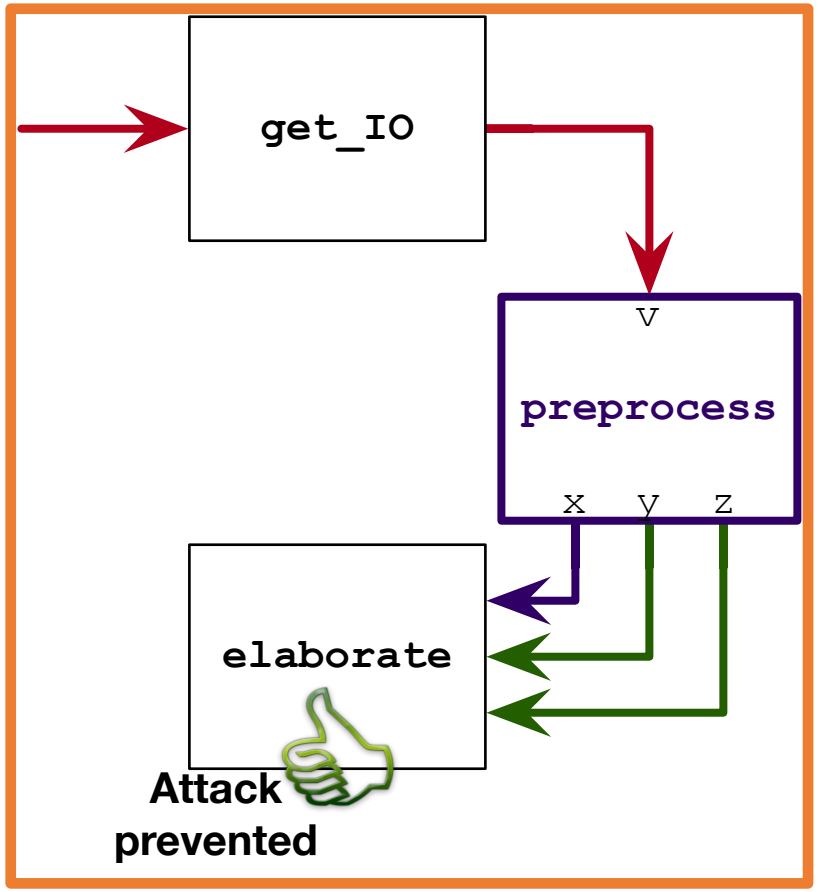
- What happens when accelerators are executed *before* the potential attack point?

```
void preprocess (int v){
  struct results ret;
  if (v > 0)
    ret.x = 1;
  else
    ret.x = 4;
  ret.y = 10;
  ret.z = 5;
  return ret;
}
```

```
struct results
{
  int x;
  int y;
  int z;
};
...
get_IO(&v);
...
ret = preprocess(v);
...
elaborate(ret);
```



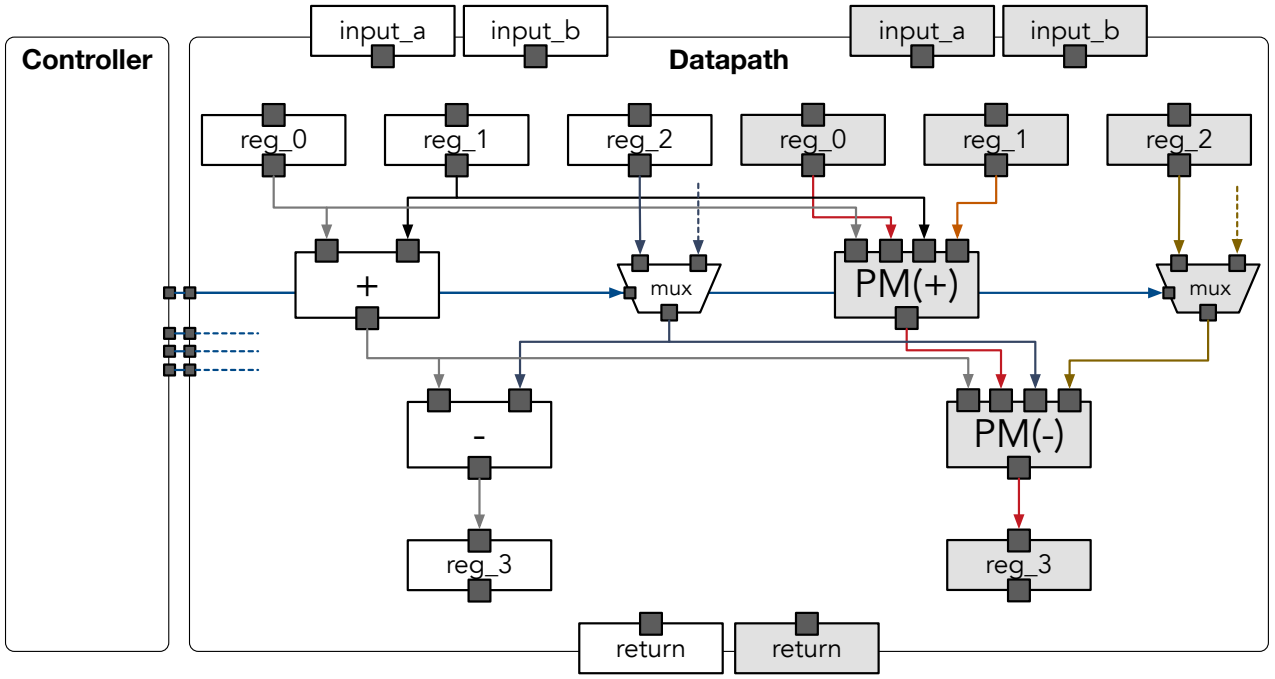
Accelerators require **fine-grained support for DIFT**

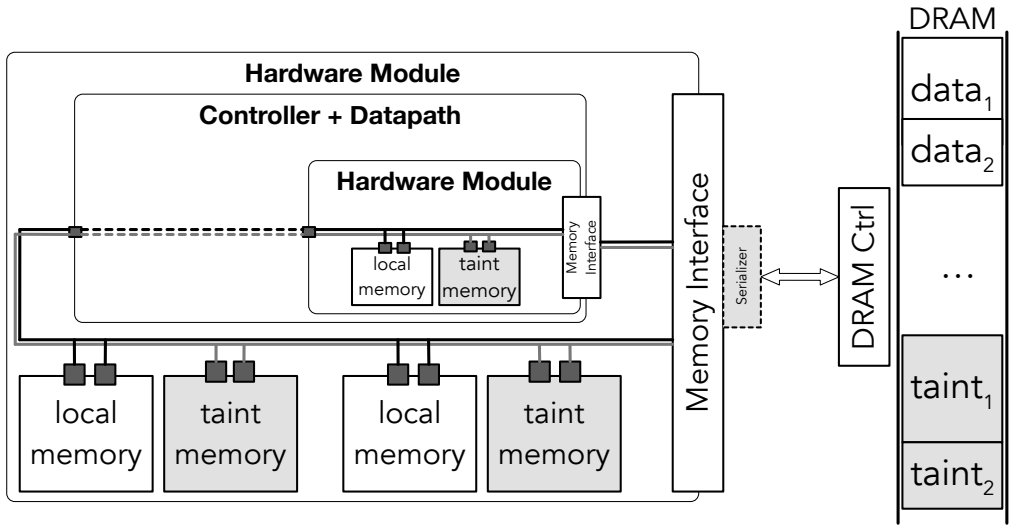


TaintHLS: DIFT Support within HLS

Data path extended with **shadow logic** and memory architecture with **taint memories**

- **HLS-based methodology** for automatic generation based on HLS results

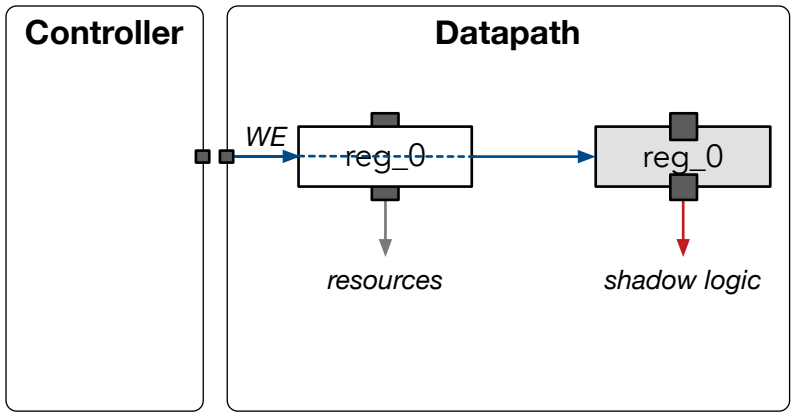


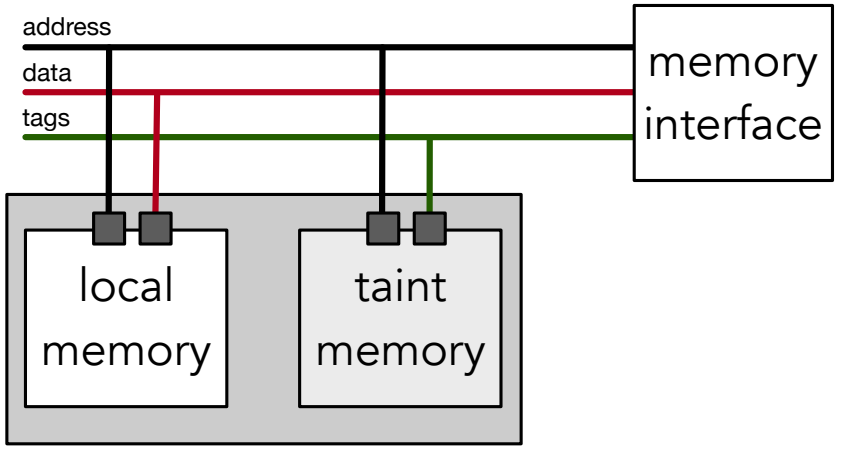


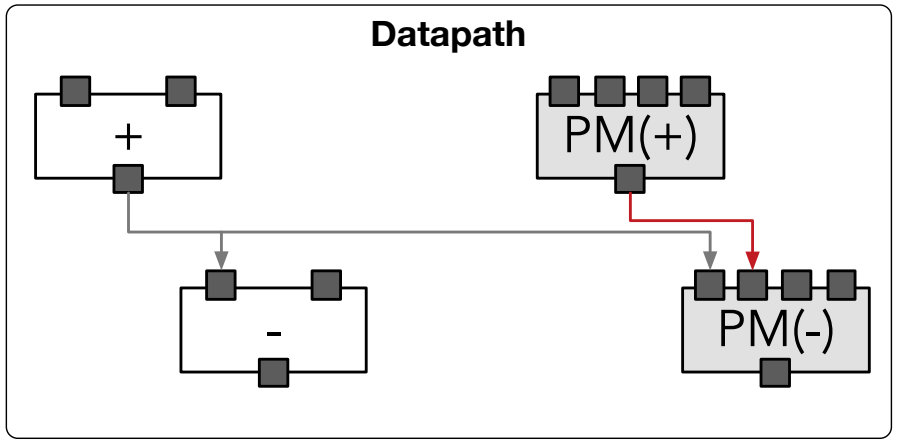
Almost **no performance overhead** with optimizations to limit area overhead

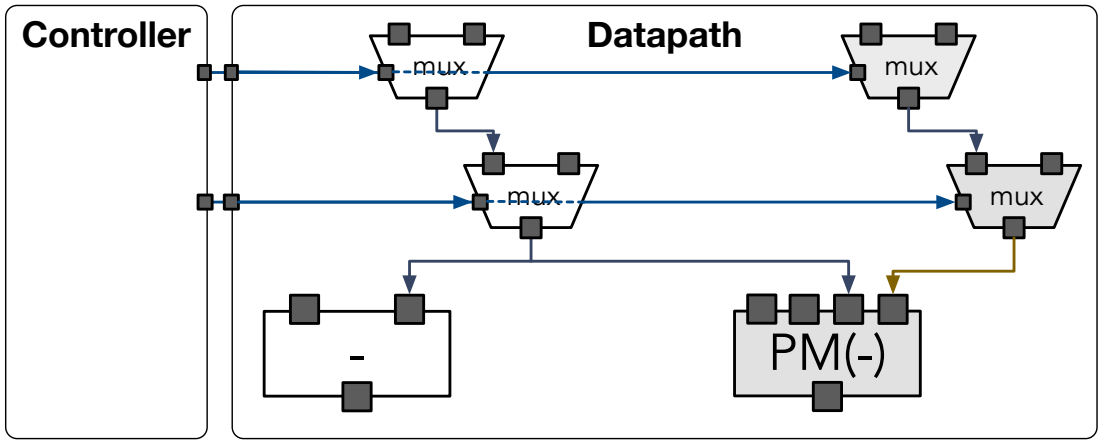
Data Flow Consistency

Microarchitectural solutions to **propagate data and tags in parallel**





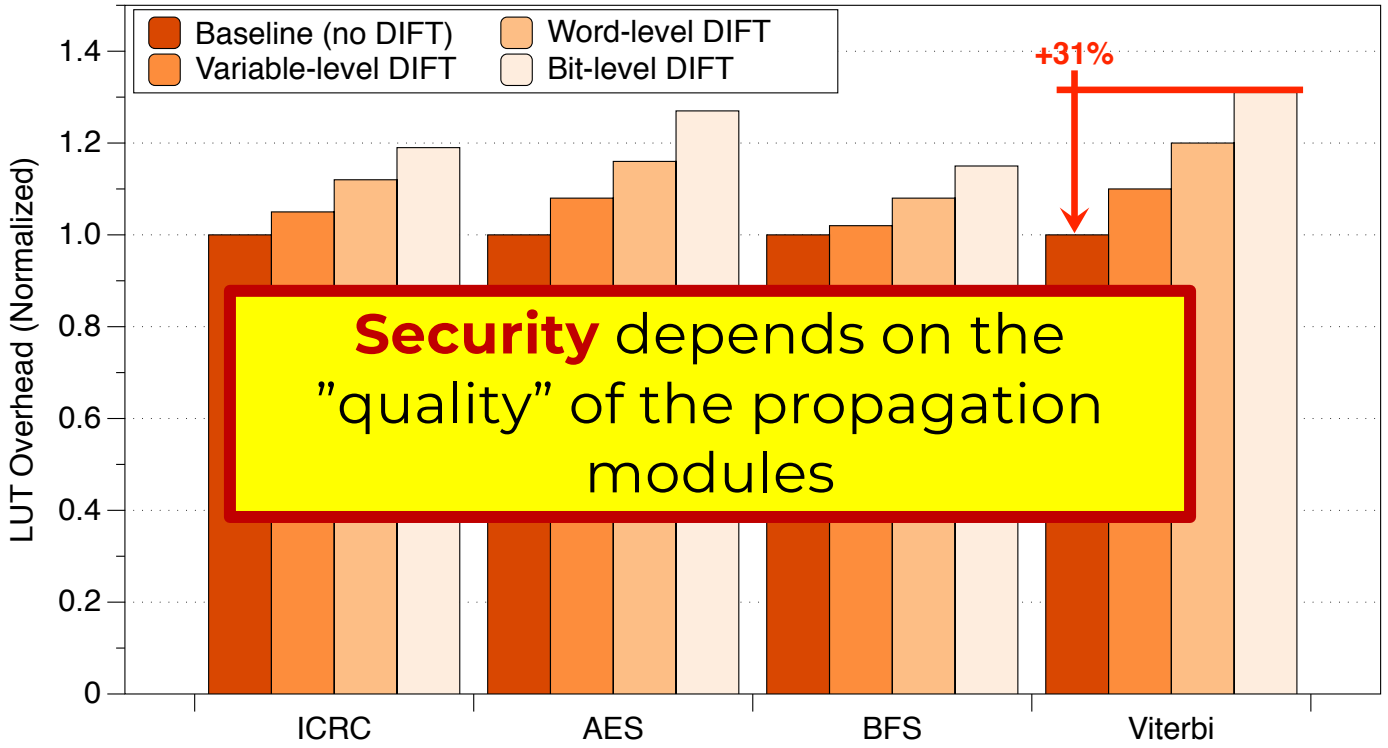




Area overhead

Area overhead of each granularity wrt the **baseline** version

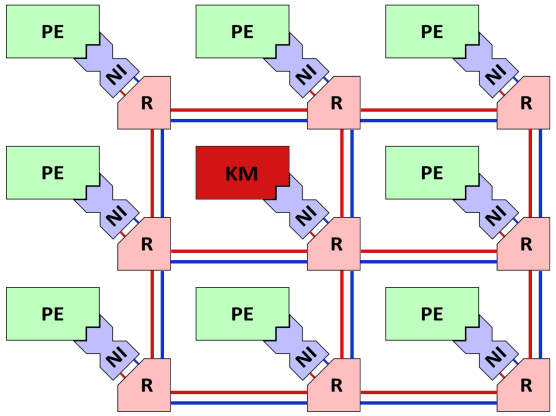
- Xilinx Virtex-7 FPGA @ 100 MHz

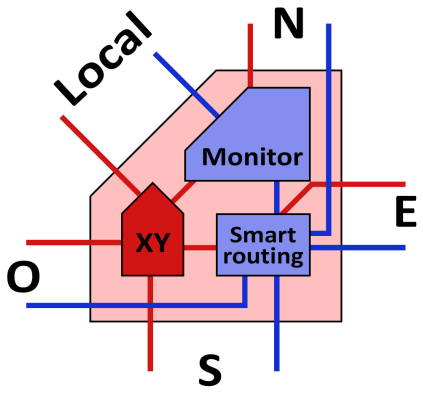


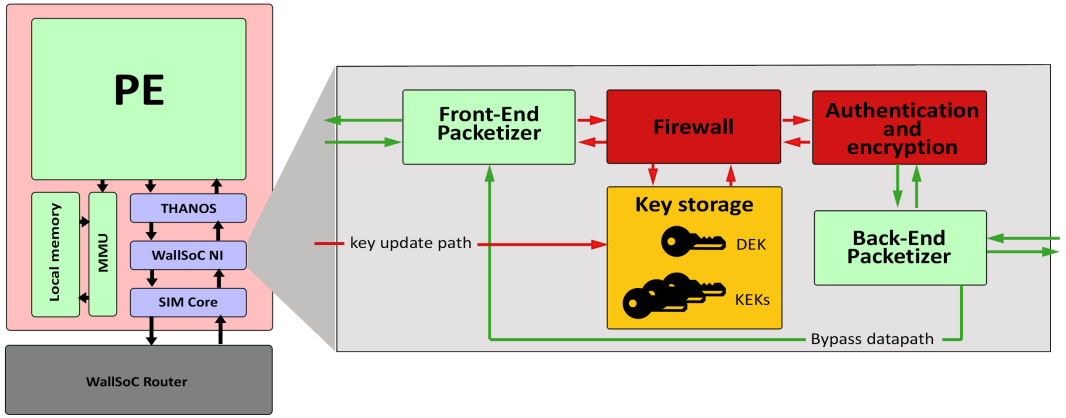
Protection of On-Chip Communications

Attackers can exploit on-chip communications to make **DoS attacks** or **NoC-channel attacks**

- **Security regions** can isolate tiles and **packet encryption** can prevent "sniffing"





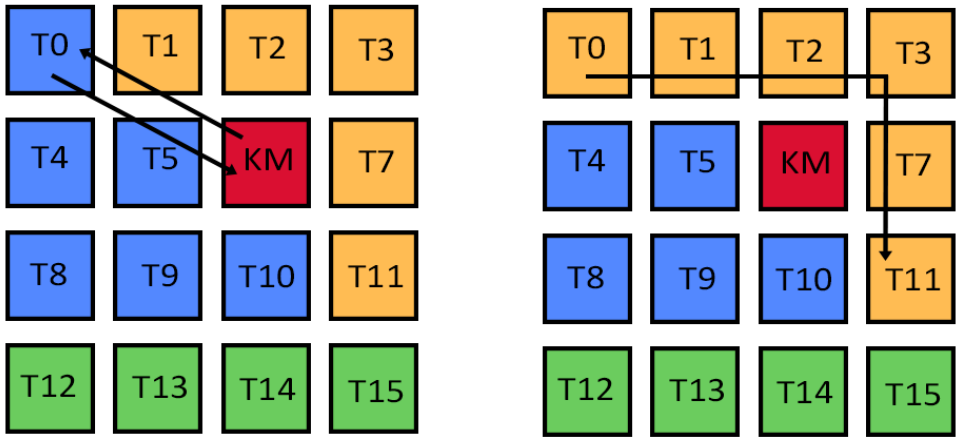


Dynamic security regions can improve **system performance**

Dynamic Security Regions

A tile can join/leave a security region upon request

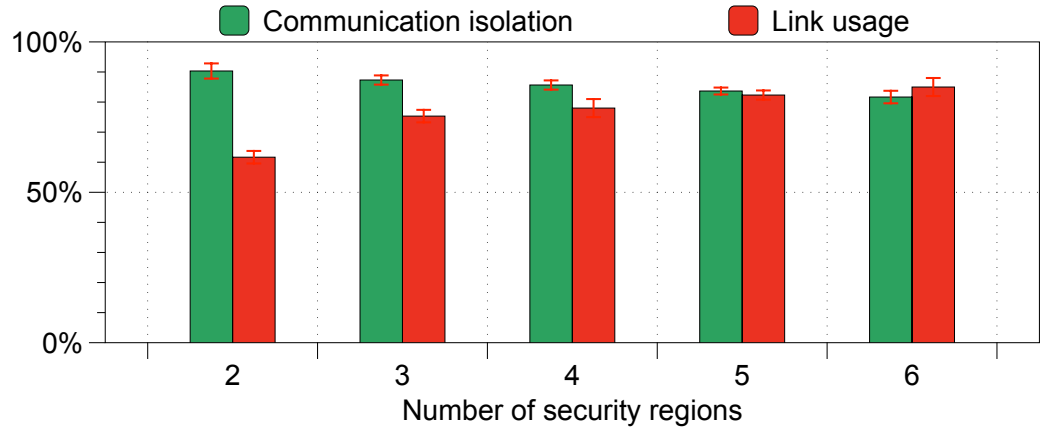
- The **smart routing** search for an **isolated path** to protect the communication



Packets are encrypted with a **group key** to ensure only the tiles in the region can read the data

System Effects

Prototype in **gem5+Noxim**, and estimations on **RTL descriptions**

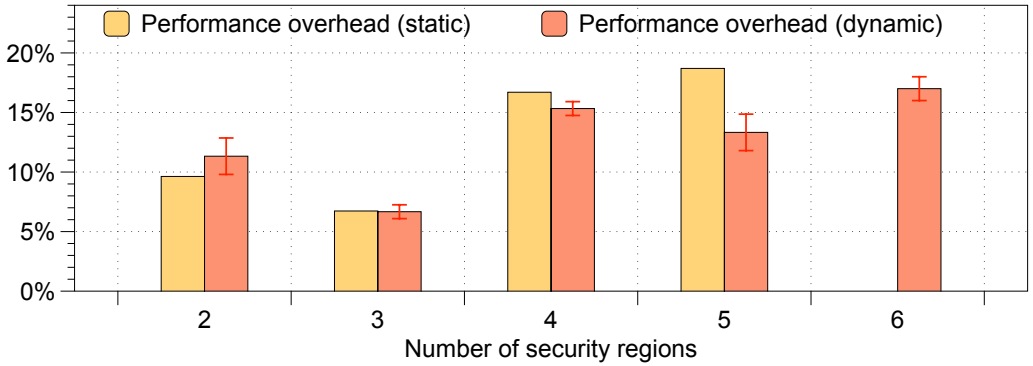


Smart routing can achieve **high communication isolation**

- Increasing the link usage leads to more power consumption

Dynamic regions remove limitations on task mapping

- This can mitigate the performance overhead (now around 17%)
- We can create architectures with more security regions



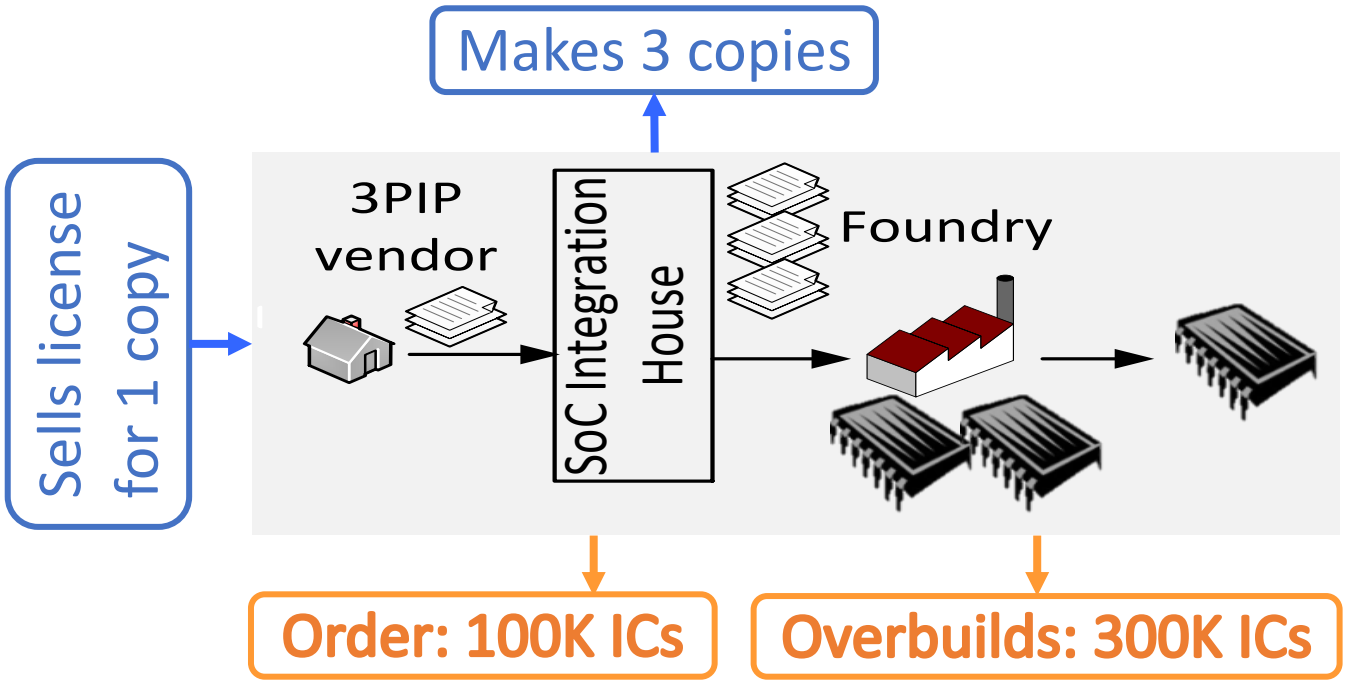
IC/IP piracy and overbuilding

Steal and claim ownership of IC and/or illegal use

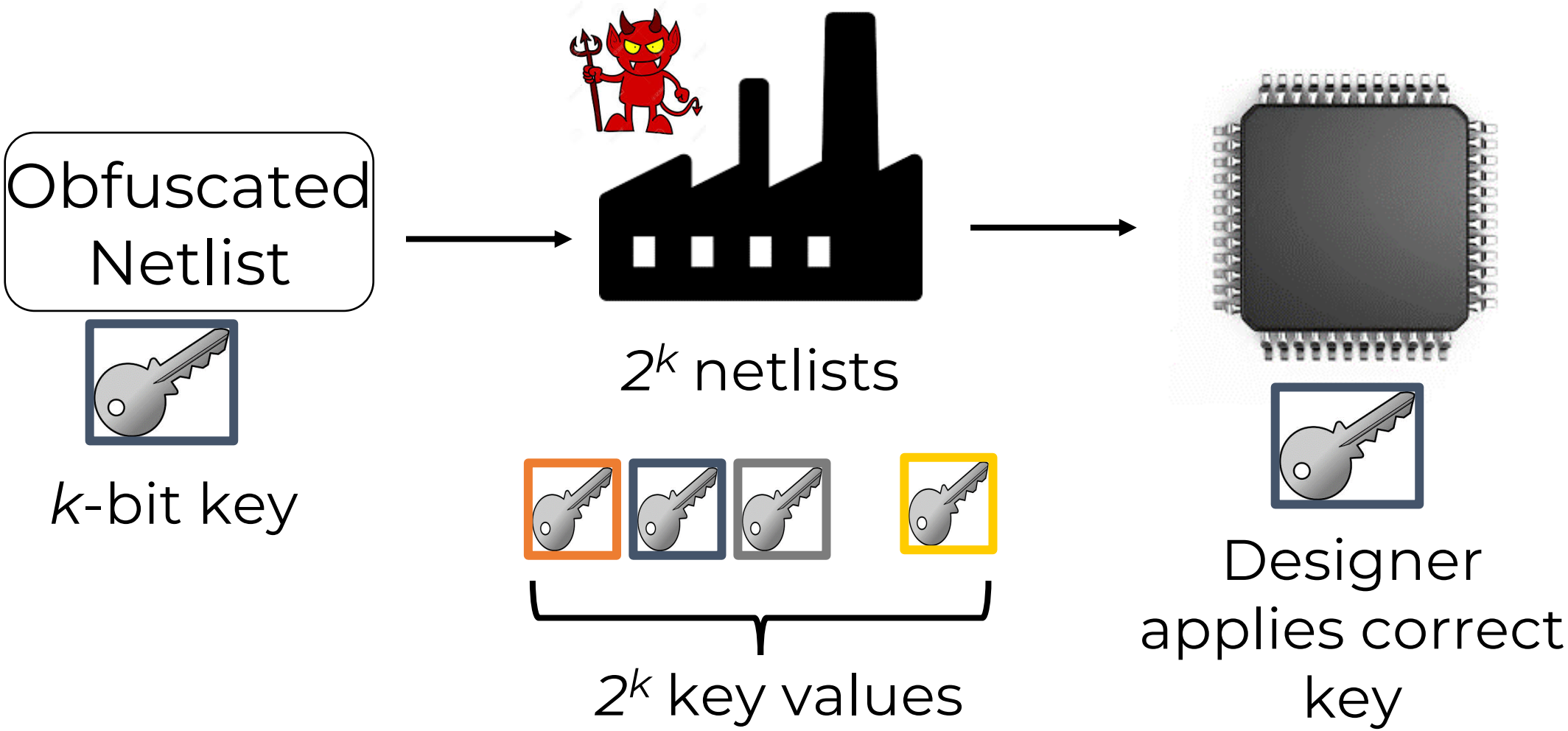
- Malicious SoC integration house
- Malicious foundry

Real-life impact

- $4,000,000,000 loss per year to IC industry
- ARM detected IP piracy in 2000

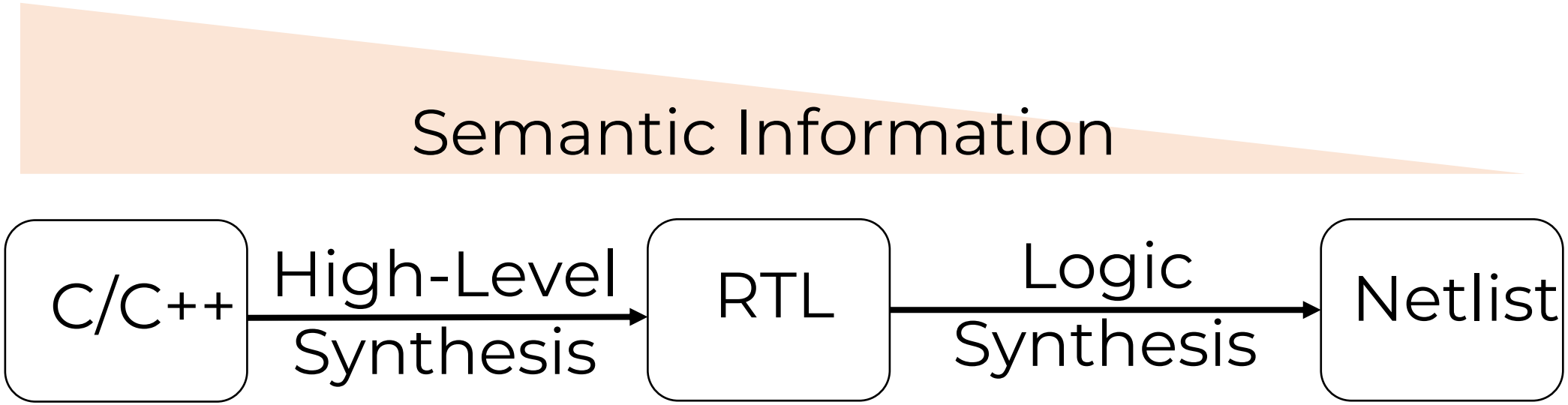


Logic obfuscation



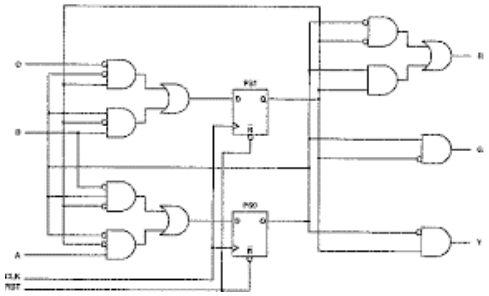
Raising the abstraction level

- Key Idea: obfuscate a design at the algorithm-level so that the obfuscation is semantically meaningful



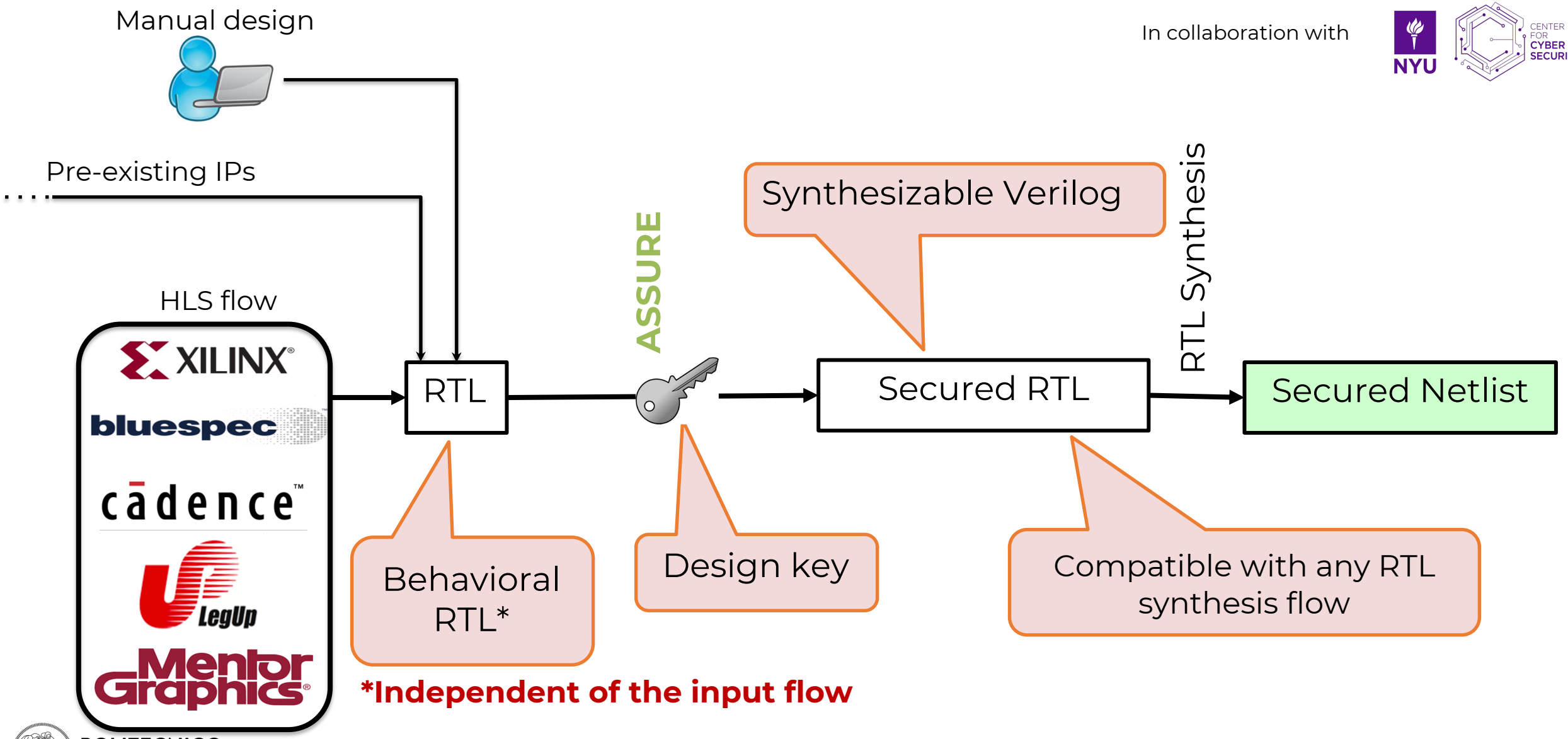

```
for (i=0; i<N; i++)
    c[i] = a[i]+b[i];
    .....
```

```
always @ posedge clk
    a[i] <= b[i] + c[i];
    .....
```



RTL Transformations for Security

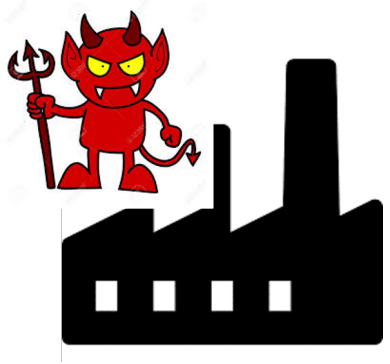
In collaboration with NYU CENTER FOR CYBER SECURITY



Threat Model: Untrusted Foundry

Attacker has **access to layout files** and can **reverse engineer** the **functionality of the netlist**

- Simulation and re-synthesis of obfuscated design
- No prior information on the design



The attacker has **no activated chip**

- Unknown input/output relationship (obviates SAT attacks)

Security is guaranteed when **all input keys are equally plausible**

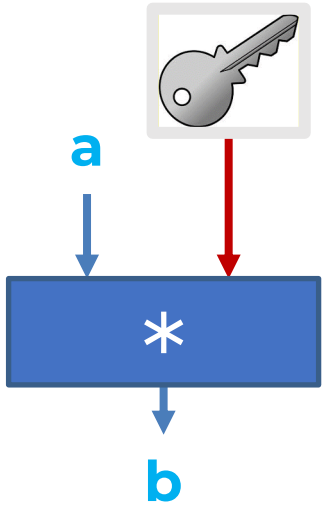
- Make random guesses without knowing if it is correct
- No insights on whether one key is correct or not

ASSURE Features

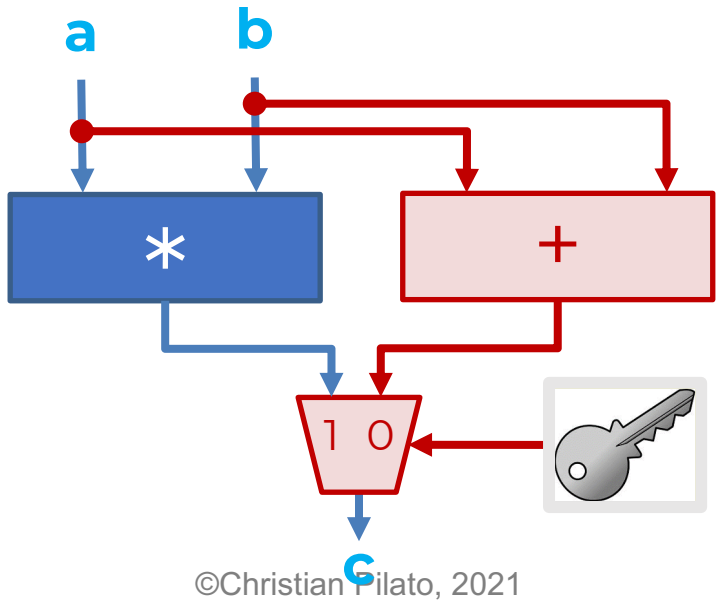
Easy-to-use **command-line tool** for **Verilog-to-Verilog RTL elaboration**

- Minimal requirements: runs with no modifications on DARPA Cloud
- Supports three high-level obfuscation techniques

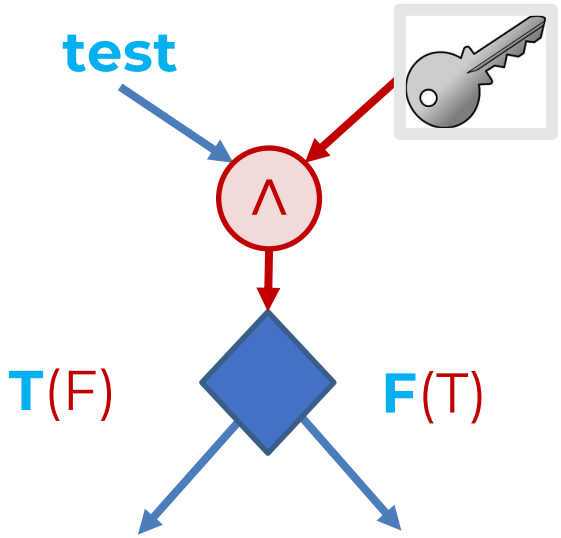
Constant obfuscation of sensitive data of the design (e.g., coefficients)



Operation obfuscation of arithmetic operations by inserting additional ones



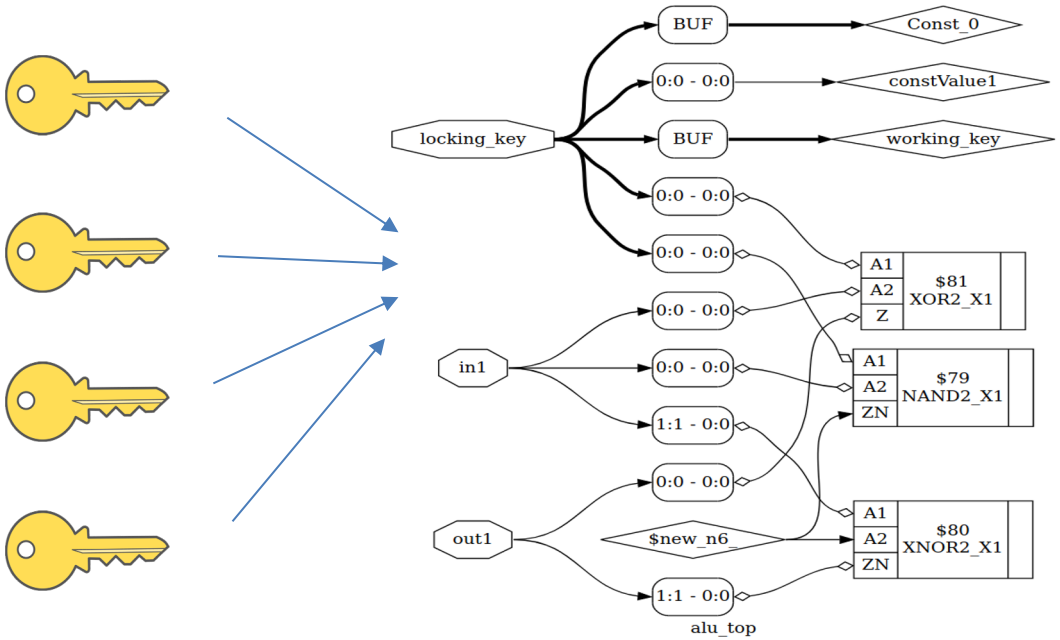
Control/branch obfuscation masking of control branches

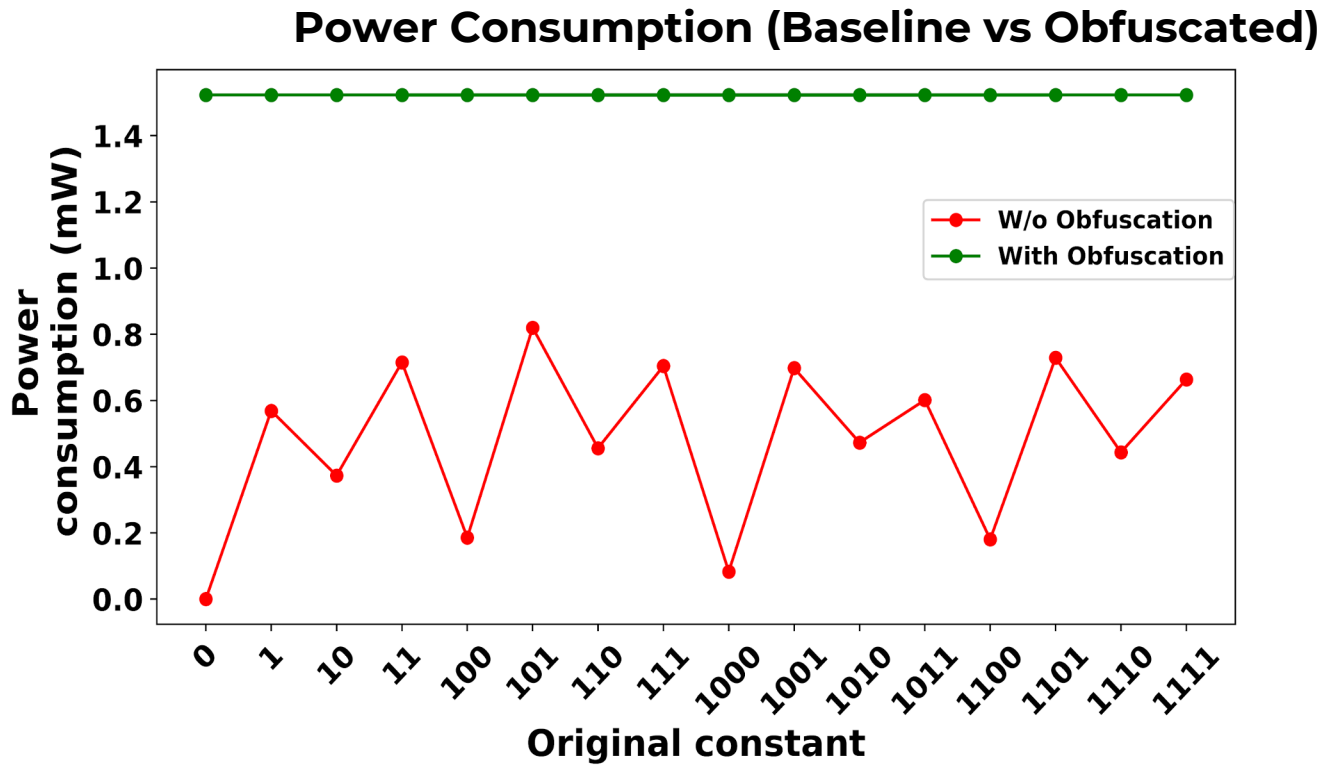


ASSURE Security Analysis

Obfuscated netlists are *isomorphic* (i.e., exactly the same) regardless of the key choice

- Attacker cannot infer key from the design
- Thus ASSURE achieves 2^K security for K key bits





ASSURE Evaluation – PASS Metrics

Correctness of obfuscated RTL designs verified using *Synopsys Formality*, i.e., with correct key, obfuscated design matches baseline

Power, Area, Speed: Overhead compared to baseline design using *Synopsys Design Compiler*: Logic synthesis for area minimization

- **Power**: Total power consumption
- **Area**: Total chip area
- **Speed**: Delay of critical path(s)

Security: *Formal proofs* of obfuscation techniques

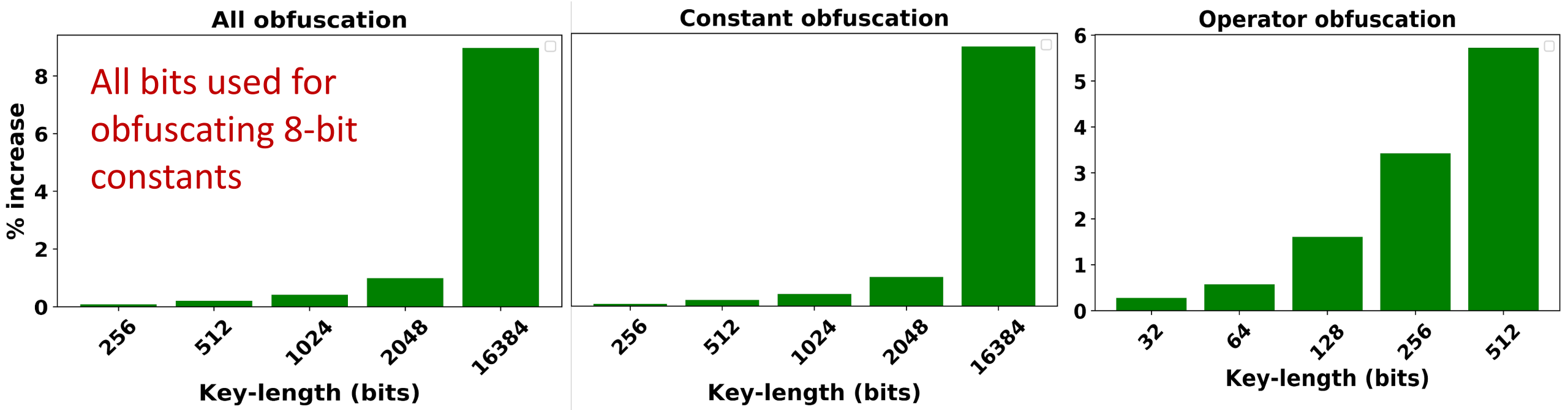
- $2^{\text{number of input key bits}}$

Benchmarks - Bits used for Locking

	Constants	Operations	Branches	Max Security
AES-192 (Datapath)	819,296 (102,403 constants)	429	1	$2^{(820K+429+1)}$
IIR Filter (Datapath)	608 (19 constants)	43	0	$2^{(608+43+0)}$
I2C-Slave (Controller)	244 (104 constants)	14	11	$2^{(244+14+11)}$
Ethernet MAC (Controller)	2414 (487 constants)	1217	218	$2^{(2414+1217+218)}$

Security vs Area Trade-offs (AES)

Synthesized with Synopsys Design Compiler J2018.SP5 targeting Nangate 15nm library (**area opt**)

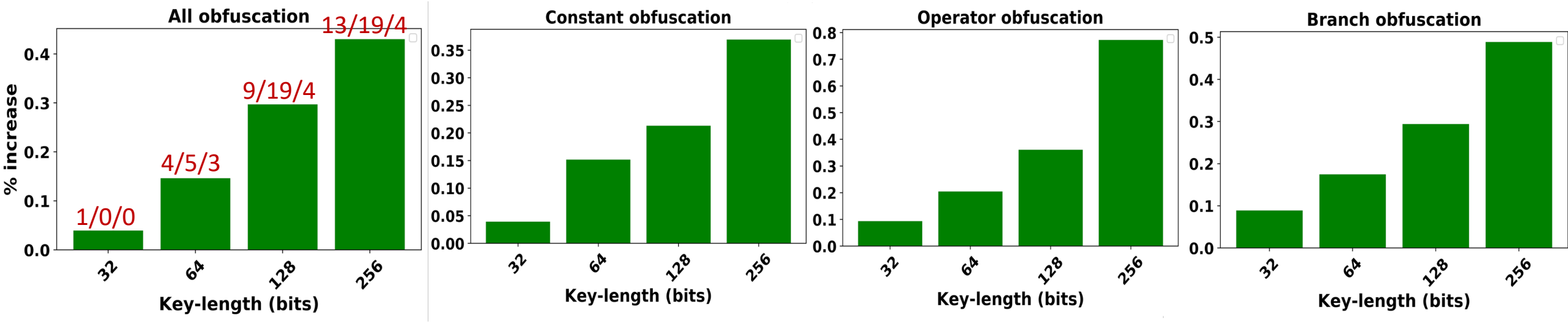


Take-Aways

1. Constant obfuscation dominates the area overhead
2. Operator obfuscation has *greater* overhead per key-bit compared to const obfuscation
3. Branch obfuscation: Limited impact because there is only 1 branch
4. Full obfuscation => 3x area overhead (~820K key bits for constants; impractical?)

Security vs Area Trade-offs (E-Mac)

Synthesized with Synopsys Design Compiler J2018.SP5 targeting Nangate 15nm library (**area opt**)



Take-Aways

1. Control-dominated design: 487 constants, 1217 operations and 218 branches
2. Branch obfuscation becomes as expensive as constant obfuscation
3. Operator obfuscation is always more expensive than constant and branch obfuscation
4. However, compared to AES, operator obfuscation is less expensive for EthernetMac

CAD Tools as Potential Attack Vectors

CAD Tools are Designed by Humans…

- Can you always trust a programmer?

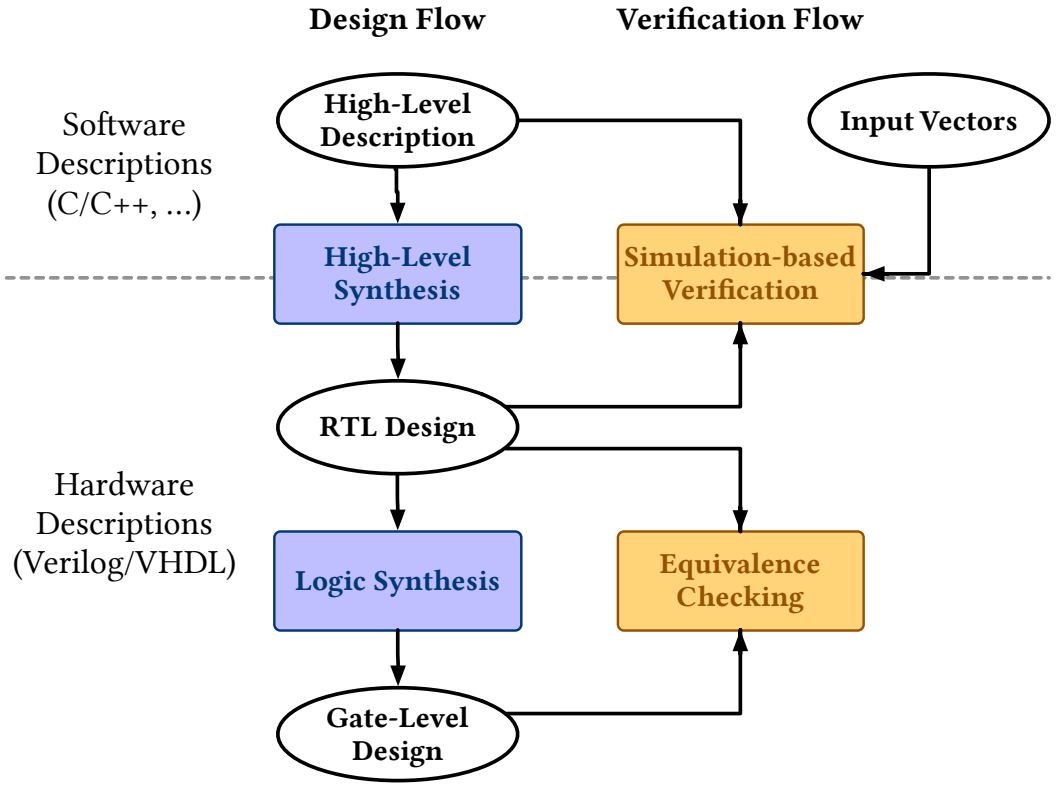
Design houses (or competitors) may have interest to **degradate IPs** after a certain amount of time

- Pushing customers to change device

Italy Fines Apple, Samsung A Few Mil For 'Planned Obsolescence' In Phones

(Forbes, Oct 28, 2018)

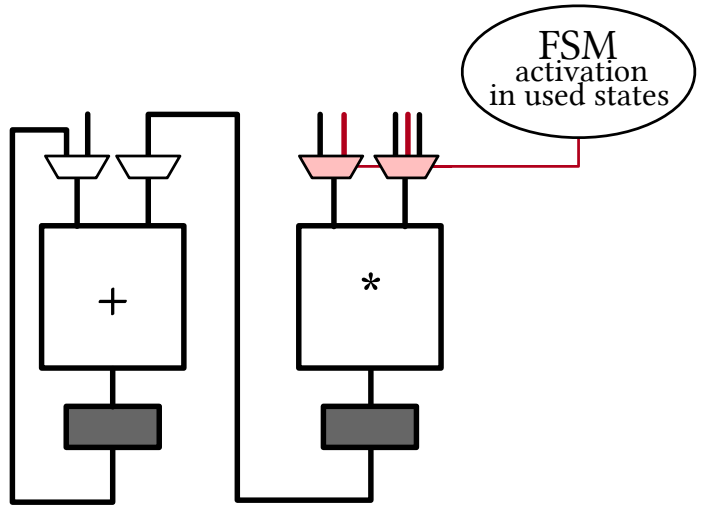
Very difficult to check
non-functional properties



Battery Exhaustion Attack

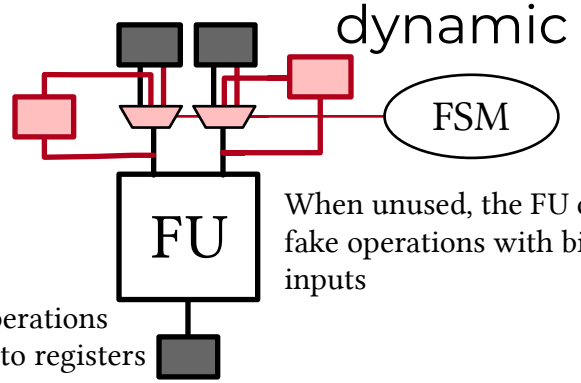
Accelerated battery discarging can motivate people to change device

- HLS knows which functional units are used in each clock cycle
- Unused units can be used to drain extra current



Selected functional units are extended with extra logic active only in specific states

Extra logic to increase switching activity (more dynamic power)



When unused, the FU computes fake operations with bit-flipped inputs

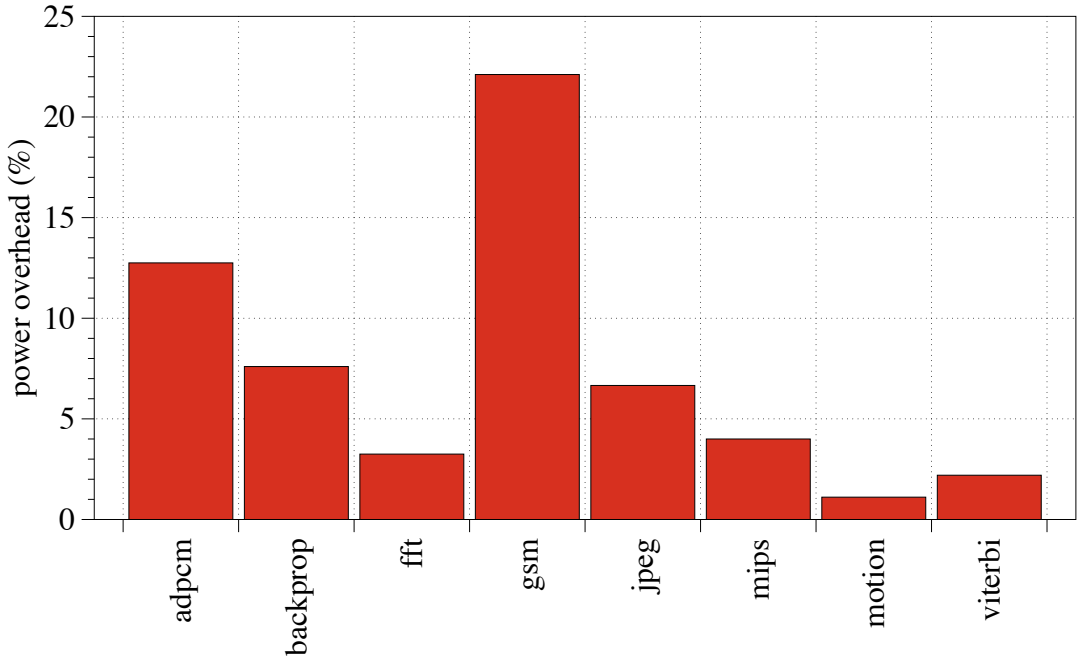
Results of faked operations are never stored into registers

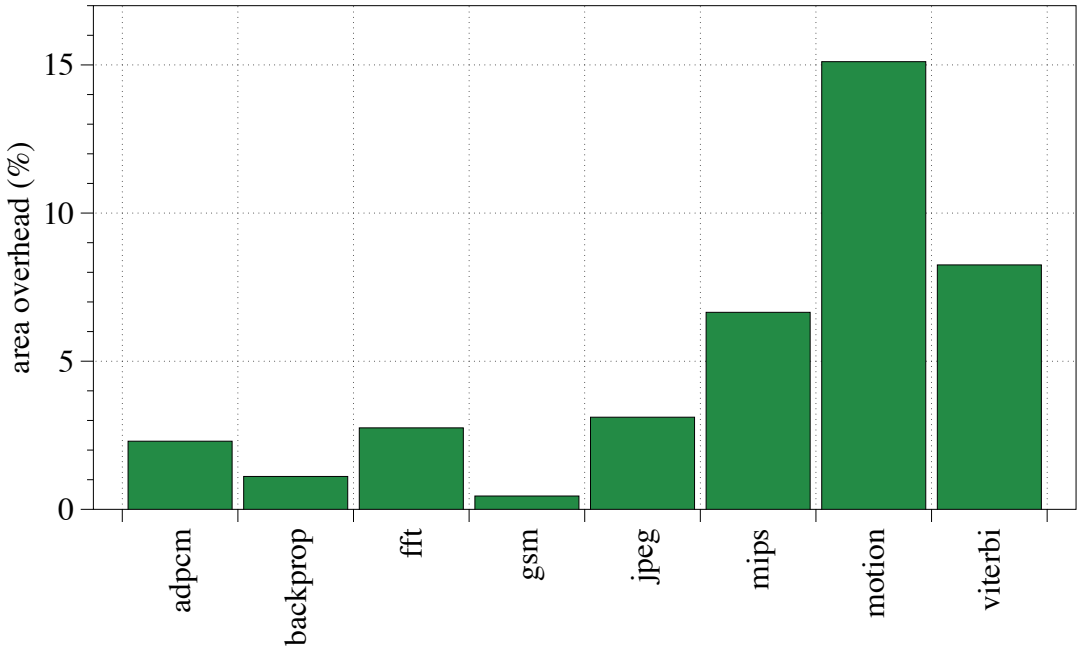
This is no golden model before HLS for power analysis

Battery Exhaustion Attack in Bambu

We added a **malicious pass** after binding to add extra logic

- Tech library provides information about power consumption





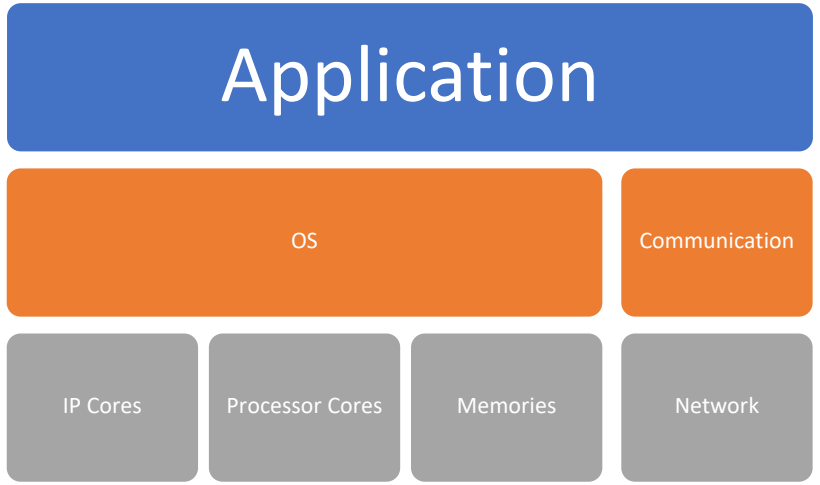
Select only the 5 most unused functional units to minimize area overhead

Minimize area overhead with a 30% power overhead budget

What is Still Missing?

Security must be address at ALL levels

- Provably-secure algorithms
- Robust OS and protected communications
- Secure components, secure architectures, secure component integration, etc…



Complete and integrated solutions are missing at all levels!

- **Separation of (security) concerns** are required for scalable solutions

Creating **awareness of the problems** is as much important as proposing countermeasures

Thank you!

Christian Pilato, christian.pilato@polimi.it

Home Reading

- C. Pilato, S. Garg, K. Wu, R. Karri, F. Regazzoni, “Securing Hardware Accelerators: A New Challenge for High-Level Synthesis,” Embedded Systems Letters 10(3): 77-80, 2018
- C. Pilato, K. Wu, S. Garg, R. Karri, F. Regazzoni, "TaintHLS: High-Level Synthesis for Dynamic Information Flow Tracking," IEEE Trans. on CAD of Integrated Circuits and Systems 38(5): 798-808 (2019)
- M. Tibaldi, C. Pilato, "WallSoC: Protecting On-Chip Communications with Dynamic Security Regions," submitted to Computer Architecture Letters (2021)
- C. Pilato, F. Regazzoni, R. Karri, S. Garg, "TAO: techniques for algorithm-level obfuscation during high-level synthesis," in Proceedings of the Design Automation Conference (DAC) 2018: 155:1-155:6
- C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg, R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2021)
- K. Basu, S. M. Saeed, C. Pilato, M. Ashraf, M. Thari Nabeel, K. Chakrabarty, R. Karri, "CAD-Base: An Attack Vector into the Electronics Supply Chain," in ACM Trans. Design Autom. Electr. Syst. 24(4): 38:1-38:30 (2019)
- C. Pilato, K. Basu, F. Regazzoni, R. Karri, "Black-Hat High-Level Synthesis: Myth or Reality?" in IEEE Trans. VLSI Syst. 27(4): 913-926 (2019)