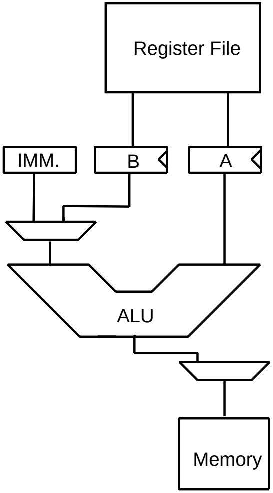
A Design Flow and Evaluation Framework for DPA-resistant Instruction Set Extensions

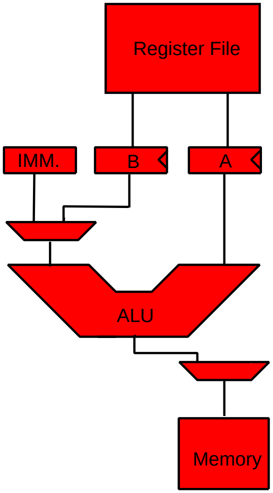
Francesco Regazzoni[1,4], Alessandro Cevrero[2,3], François-Xavier Standaert[1], Stephane Badel[3], Theo Kluter[2], Philip Brisk[2], Yusuf Leblebici[3], and Paolo Ienne[2]

[1]UCL Crypto Group, Université catholique de Louvain, Louvain-la-Neuve, Belgium.
[2]School of Computer and Communication Sciences - EPFL, Lausanne, Switzerland.
[3]School of Engineering - EPFL, Lausanne, Switzerland.
[4]ALaRI - University of Lugano, Lugano, Switzerland.

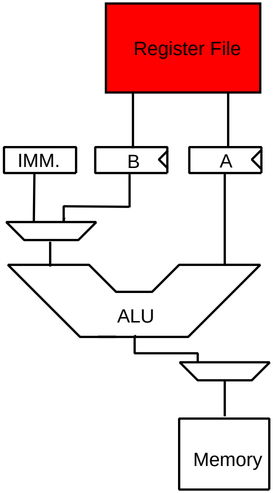
A Motivating Example



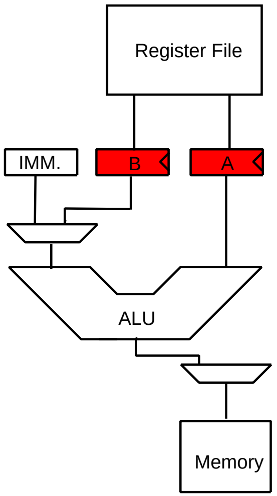
A Motivating Example



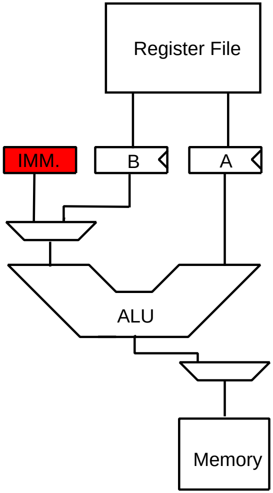
A Motivating Example



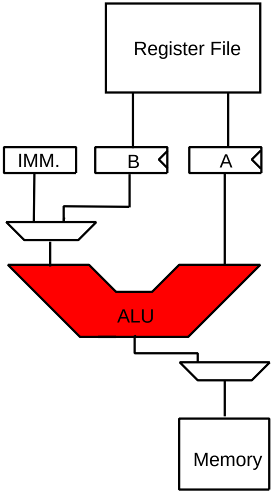
A Motivating Example



A Motivating Example



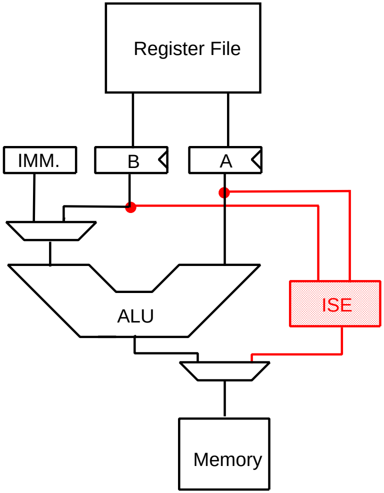
A Motivating Example



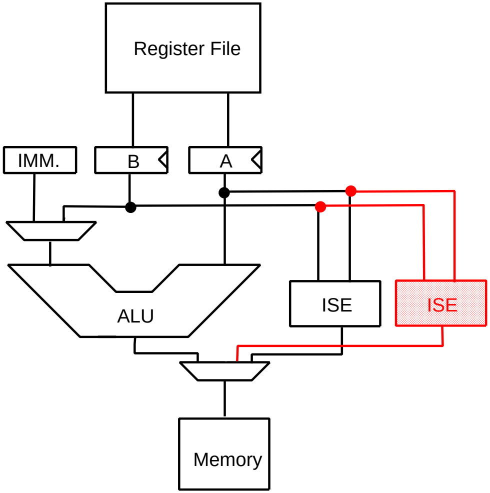
A Motivating Example

- Something easier?

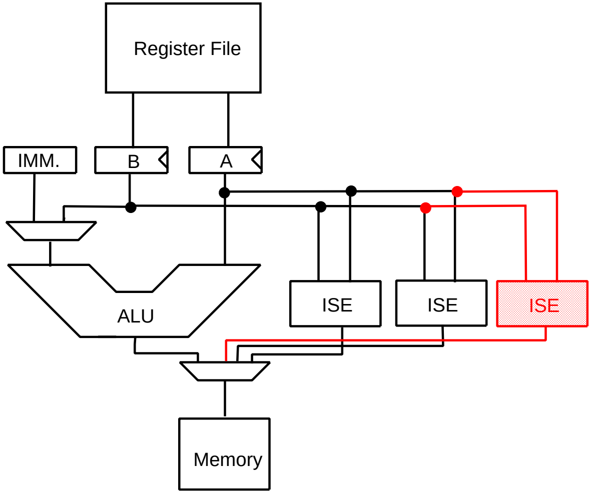
A Motivating Example



A Motivating Example



A Motivating Example



Contributions and Goals

- Bring **a security metric** to the forefront of **design variables** to be optimized
- Create an automated design flow for **combining protected and non protected logic** styles
- Explore type and amount of protected circuit vs level of security **trade offs**

Needed “Bricks”

- Tool for extracting ISE...

Needed “Bricks”

- Tool for extracting ISE... ✓

Needed “Bricks”

- Tool for extracting ISE... ✓
- Protected logic and its design flow...

Needed "Bricks"

- Tool for extracting ISE... ✓
- Protected logic and its design flow... ✓

Needed "Bricks"

- Tool for extracting ISE... ✓
- Protected logic and its design flow... ✓
- Simulation environment...

Needed “Bricks”

- Tool for extracting ISE... ✓
- Protected logic and its design flow... ✓
- Simulation environment... ✓

Needed "Bricks"

- Tool for extracting ISE... ✓
- Protected logic and its design flow... ✓
- Simulation environment... ✓
- Metric...

Needed "Bricks"

- Tool for extracting ISE... ✓
- Protected logic and its design flow... ✓
- Simulation environment... ✓
- Metric... ✓

Needed “Bricks”

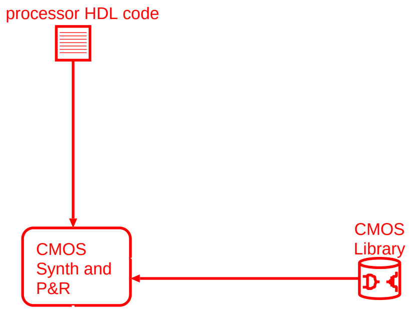
- Tool for extracting ISE... ✓
- Protected logic and its design flow... ✓
- Simulation environment... ✓
- Metric... ✓

Main question

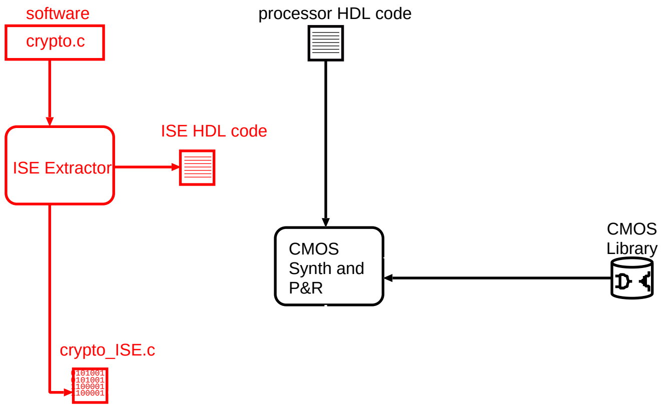
- **Can I really plug all the bricks together and obtain something meaningful?**

Outline

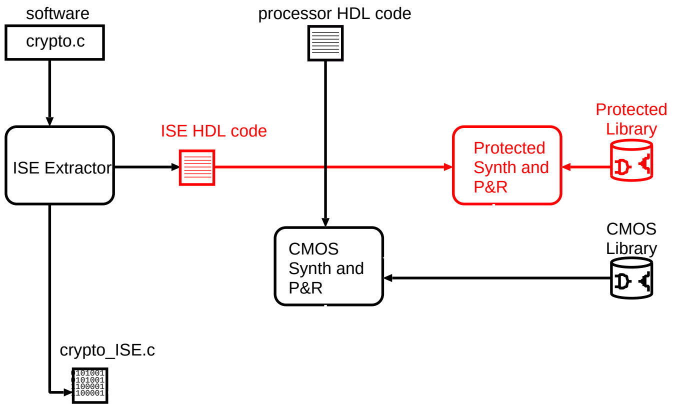
The CMOS Design Flow



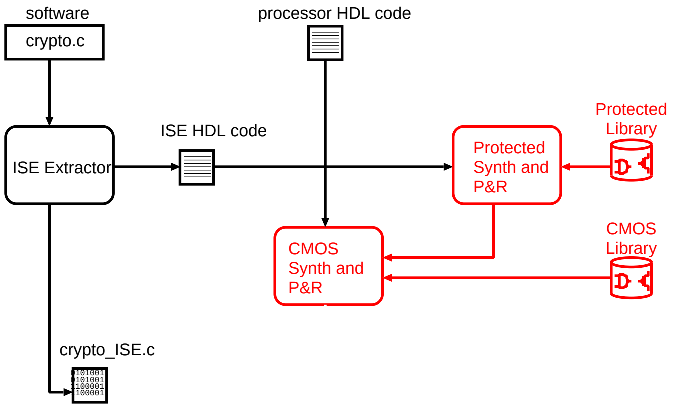
The Processor Customization



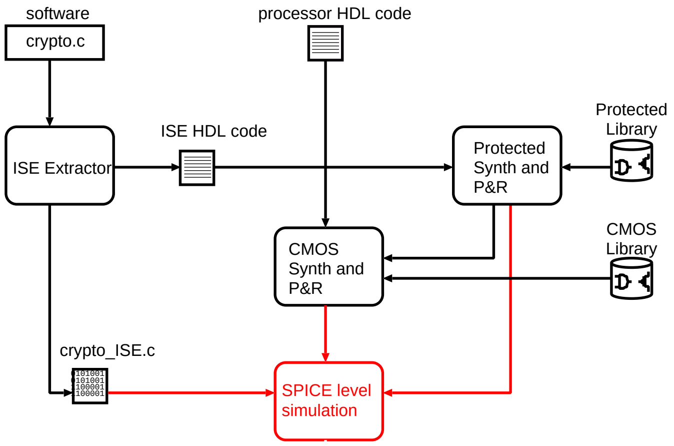
The Protected Design Flow



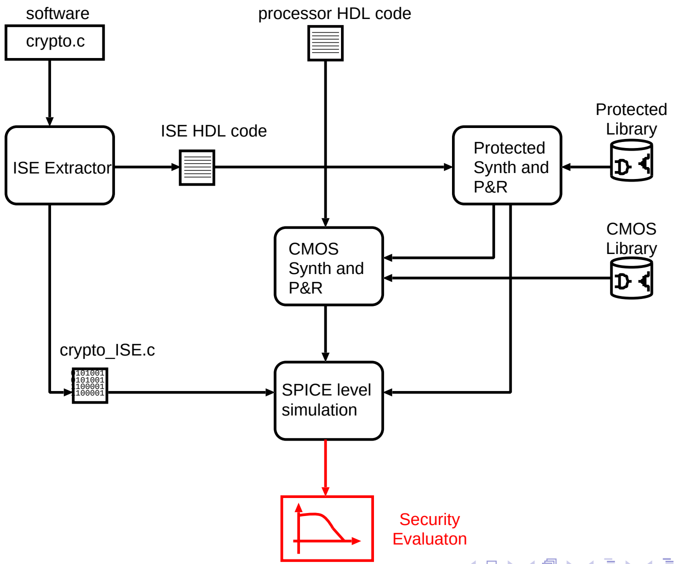
The Hybrid Design Flow



The Simulation Environment



The Design Evaluation



Outline

1. What we put together
2. Validation of our design flow
3. Results and comments

Features of OpenRISC

- OpenRISC 1000 ISA
- 32 bit
- 5 stages pipeline
- ISE support
- 100 MHz
- Compiler: cross-compiler gcc 3.4.4

Features of CMOS and MCML

- CMOS target library: commercial 0.18μm

- Protected logic: **MOS Current Mode Logic (MCML)**
  - Standard cell **Library** (roughly 150 gates)
  - High speed, fully differential, almost constant power consumption
  - Differential routing (wire pairs along the same path)
  - Fully automated design flow

Features of Information Theory Metric

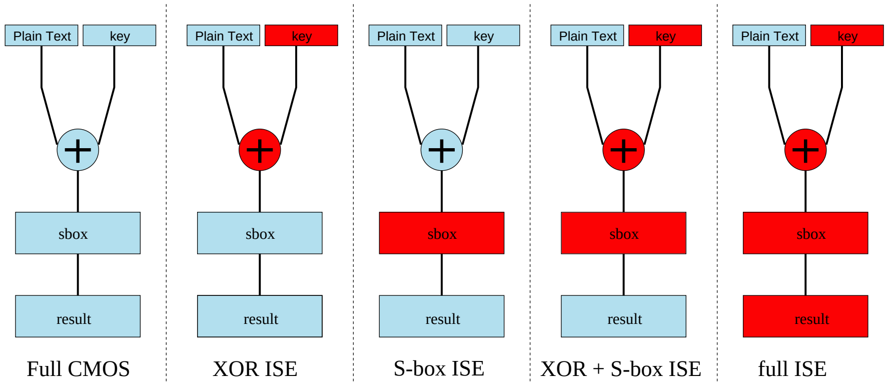
- Measures **asymptotic resistance** against the strongest attacker
- **Independent** from the DPA scenario
- Overcomes limitations of specific leakage models
- Main Steps:
  - Inputs: power consumption trace, secret key
  - Add white noise
  - Reduce the dimension using PCA
  - Compute the mutual information

Features of PRESENT

- Lightweight block cipher
- **4 bit** S-box
- *addRoundKey*, *sBoxLayer*

```
// Calculate S-box (plaintext XOR key)
int PRESENT(int plaintext, int key) {
int result = 0; // initialize the result
plaintext = plaintext ^key; // perform the xor with the key
result = S[plaintext]; // perform the S-box
return result; }; // return the result
```

Partitioning of the PRESENT algorithm S-box



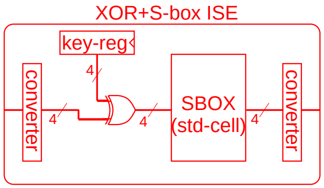
Legend	
🟦 non protected logic	🟥 protected logic

Example of ISE and its Source Code

```
// Calculate S-box (plaintext XOR key)
int PRESENT(int plaintext, int key) {
1 int result = 0; // initialize the result
2 plaintext = plaintext ^key; // perform the xor with the key
3 result = S[plaintext]; // perform the S-box
4 return result; }; // return the result
```

Example of ISE and its Source Code

```
// Calculate S-box (plaintext XOR key)
int PRESENT(int plaintext, int key) {
int result = 0; // initialize the result
plaintext = plaintext ^key; // perform the xor with the key
result = S[plaintext]; // perform the S-box
return result; }; // return the result
```



```
// Calculate S-box (plaintext XOR key)
int PRESENT_XOR+S-box-ISE(int plaintex) {
int result = 0; // initialize the result
 // instantiate the new instruction s-box(pt ^key)
Instr_1(plaintex, result);
return result; }; // return the result
```

Outline

It is possible!

PC Features:

- CPU: Intel(R) Core(TM)2 Quad CPU Q6700
- GHz 2.6
- Memory: 4 GB

It is possible!

PC Features:

- CPU: Intel(R) Core(TM)2 Quad CPU Q6700
- GHz 2.6
- Memory: 4 GB

Example program 470 clock cycles (boot+cipher)

SPICE Level Simulation (Synopsys Nanosim resolution: 1ps):

- Total simulated time 4700ns
- Total simulation time more or less 20 minutes
- 2.8s per clock cycle (full processor simulation core+ISE)

Security Evaluation

- 4 hours per partitioning

It is possible!

PC Features:

- CPU: Intel(R) Core(TM)2 Quad CPU Q6700
- GHz 2.6
- Memory: 4 GB

Example program 470 clock cycles (boot+cipher)

SPICE Level Simulation (Synopsys Nanosim resolution: 1ps):

- Total simulated time 4700ns
- Total simulation time more or less 20 minutes
- 2.8s per clock cycle (full processor simulation core+ISE)

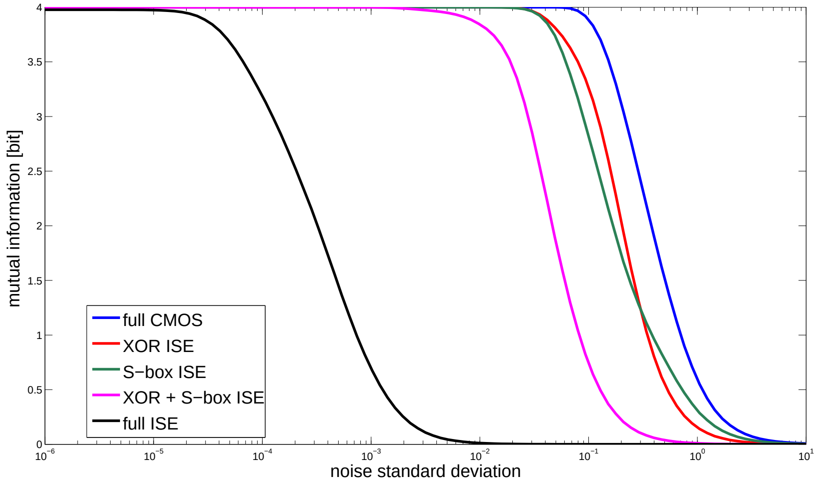
Security Evaluation

- 4 hours per partitioning

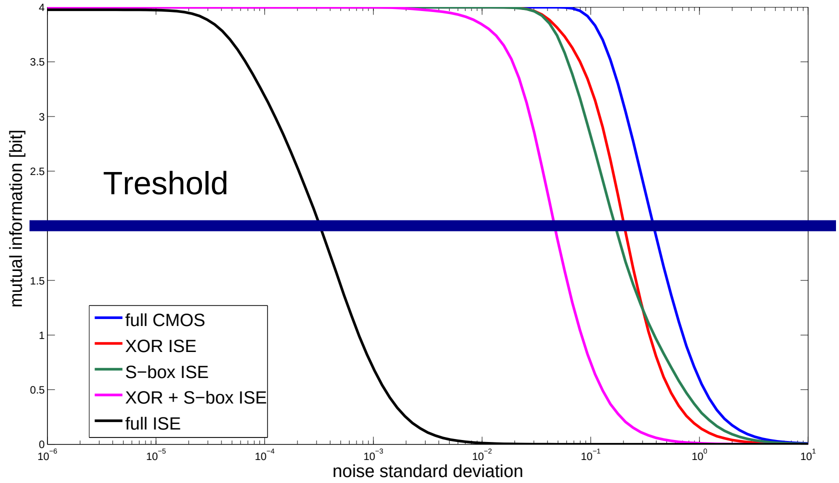
Full case study

- Worst case: 15 days on a single PC
- Parallelizable! Actual experiment: 2 days on 8 PCs

Security Evaluation



Security Evaluation



Comments on Area occupation and Power Consumption

- Area difference between largest and smallest ISE: **0.2%**

- Most protected core:
  - power consumption increased by **47.9%**
  - area increased by **6.7%**

Comments on Area occupation and Power Consumption

- Area difference between largest and smallest ISE: **0.2%**
- Most protected core:
  - power consumption increased by **47.9%**
  - area increased by **6.7%**

- **Size** of the PRESENT S-box
- MCML library optimized for high-speed, not low-power

Conclusions

- **It is possible** to put everything together
- Our results **confirm** the previous ones
- **ISEs** in **protected** logic styles could be a reasonable countermeasure
- Our design flow enables a deeper **design space exploration**

Future Works and Open problems

One step in a direction, but...

Future Works and Open problems

One step in a direction, but...

- Nice design flow... **use it!**

Future Works and Open problems

One step in a direction, but...

- Nice design flow... **use it!**
- Great! Obtained promising chips... **make them!**

Questions?

Acknowledgments