Towards the Automatic Applications of Side Channel Countermeasures

Francesco Regazzoni
Why Electronic Design Automation for security?

- Security is very often considered at later stages of design
- Cost and Time to Market
- Possible Security pitfalls
Why Electronic Design Automation for security?

- Security is very often considered at later stages of design
- Cost and Time to Market
- Possible Security pitfalls

EXTRA CONSTRAINT

Use as much as possible “standard” EDA commodities!
Motivating Example

```c
void maskedARK() {
    unsigned char i;
    for (i=0;i<16;i++){
        st[i] = pt[i] ^
            (key[i] ^ mask[i]);
    }
}
```

```
.text
.global ARK
.type ARK, @function

ARK:
/* prologue: function */
/* frame size = 0 */
/* stack size = 0 */
.L__stack_usage = 0

lds r24,key
lds r25,pt
eor r24,r25

lds r25,mask
eor r24,r25
sts st,r24
lds r24,key+1
lds r25,pt+1
eor r24,r25
...
```

avr-gcc-4.5.3-O3
1 Motivations

2 DPA Resistant Synthesis

3 DPA Resistant Place and Route

4 DPA Resistant Instruction Set Extension

5 Quick Note on Software
Simplified Hardware Design Flow (ASIC)

Algorithm Design
C, Matlab, VHDL

RTL (Architecture) Design
Synthesizable HDL

Gate

Layout
Let’s focus on Synthesis

RTL (Architecture) Design
Synthesizable HDL

Logic Synthesis

Gate Level
Logic Synthesis Input and Output

**INPUT:**
- HDL Description
- Technological Library (area, timing, power)
- Synthetic Library (multipliers...)
- Constraints

**OUTPUT:**
- Gate Level Netlist
- Estimation of area, timing, power (!)
- Timing constraints
Countermeasures

Power consumption independent from processed key dependent data

Intermediate values of the cryptographic algorithm

Intermediate values processed by the device

Power consumption of the cryptographic device
Power consumption independent from processed key dependent data

Intermediate values of the cryptographic algorithm

Intermediate values processed by the device

Power consumption of the cryptographic device

Masking Countermeasures
Power consumption **independent** from processed key dependent data

Diagram:

1. Intermediate values of the cryptographic algorithm
2. Intermediate values processed by the device
3. Power consumption of the cryptographic device

- **Masking Countermeasures**
- **Hiding Countermeasures**
Power consumption independent from processed key dependent data

Intermediate values of the cryptographic algorithm

- Masking Countermeasures

Intermediate values processed by the device

- Hiding Countermeasures

Power consumption of the cryptographic device

They can be implemented in **Software** or in **Hardware**
### Approach One

**INPUT:**
- HDL Description
- Technological Library (area, timing, power)
- Synthetic Library (multipliers...)
- Constraints

**OUTPUT:**
- DPA resistant Gate Level Netlist
- Estimation of area, timing, power (!)
- Timing constraints
Approach Two

INPUT:
- HDL Description
- Technological Library (area, timing, power)
- Synthetic Library (multipliers...)
- Constraints (limit the gates)

OUTPUT:
- Gate Level Netlist

“Cell Substitution”:
- Replace cells with
- Reload the design for correct area and timing

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motivations</td>
</tr>
<tr>
<td>2</td>
<td>DPA Resistant Synthesis</td>
</tr>
<tr>
<td>3</td>
<td>DPA Resistant Place and Route</td>
</tr>
<tr>
<td>4</td>
<td>DPA Resistant Instruction Set Extension</td>
</tr>
<tr>
<td>5</td>
<td>Quick Note on Software</td>
</tr>
</tbody>
</table>
**INPUT:**
- Gate level description of the circuit
- Physical view of the library (pin placement, ...)
- Constraints from synthesis

**OUTPUT:**
- Gate Level Netlist
- Position and interconnection of the gates
- Estimation of area, timing, power (!)
This is not yet the end!

- Security Evaluation
- Chip Finalization
- Tape Out
- Security Evaluation
- Toggle Count
- ... 
- SPICE simulation 
- Real measures on fabricated chip
- Measure directly
- Less Freedom
- Tools more “closed”
Contents

1 Motivations
2 DPA Resistant Synthesis
3 DPA Resistant Place and Route
4 DPA Resistant Instruction Set Extension
5 Quick Note on Software
Protect PRESENT with secure hardware

- Lightweight block cipher
- 4 bit S-box
- \texttt{addRoundKey, sBoxLayer}

```c
// 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
```
What can I do?

ALU

Register File

IMM.

B

A

ALU

Memory

Francesco Regazzoni

23 October 2015, Chia, Italy

P. 23
What can I do?
What can I do?
What can I do?

ALU

Register File

IMM.

B <

A <

ALU

Memory

Francesco Regazzoni
23 October 2015, Chia, Italy
What can I do?
What can I do?
What can I do?

Something easier?
Protected / Non Protected CO-Design!
Protected / Non Protected CO-Design!
Automatic design of DPA resistant ISE

Identify sensitive parts

Partition Sensitive / Non Sensitive

Protect Sensitive

Security Evaluation
Needed “Basic Blocks”

- Generate useful power traces?
Needed “Basic Blocks”

- Identify sensitive parts
- Partition sensitive / non-sensitive
- Protect sensitive
- Security evaluation

- Generate useful power traces?
- Measure the DPA resistance?
Needed “Basic Blocks”

- Identify sensitive parts
- Partition Sensitive / Non Sensitive
- Protect Sensitive
- Security Evaluation

- Generate useful power traces?
- Measure the DPA resistance?
- Countermeasure and its design flow?
Needed “Basic Blocks”

- Identify sensitive parts
- Partition Sensitive / Non Sensitive
- Protect Sensitive
- Security Evaluation

Generate useful power traces?
- Measure the DPA resistance?
- Countermeasure and its design flow?
- Partition the algorithm?
Needed “Basic Blocks”

- Generate useful power traces?
- Measure the DPA resistance?
- Countermeasure and its design flow?
- Partition the algorithm?
The CMOS Design Flow

processor HDL code

CMOS Synth and P&R

CMOS Library

crypto.c

Francesco Regazzoni 23 October 2015, Chia, Italy
The Processor Customization

- **crypto.c**
- **ISE Extractor**
- **ISE HDL code**
- **crypto_ISE.c**
- **processor HDL code**
- **CMOS Synth and P&R**
- **CMOS Library**
The Protected Design Flow

- **software**
  - `crypto.c`
- **processor HDL code**
- **ISE HDL code**
- **Protected Library**
- **CMOS Synth and P&R**
- **crypto_ISE.c**

Diagram:

- ISE Extractor
- Protected Synth and P&R
- CMOS Synth and P&R

Francesco Regazzoni
23 October 2015, Chia, Italy
The Hybrid Design Flow

- **software**
  - crypto.c

- **processor HDL code**

- ISE HDL code

- **ISE Extractor**

- **Protected Library**

- **Protected Synth and P&R**

- **CMOS Synth and P&R**

- **CMOS Library**

Francesco Regazzoni
23 October 2015, Chia, Italy
The Simulation Environment

- software
  - crypto.c

- processor HDL code

- ISE HDL code

- ISE Extractor

- Protected Library

- Protected Synth and P&R

- CMOS Library

- CMOS Synth and P&R

- crypto_ISE.c

- SPICE level simulation
Partitioning of the PRESENT algorithm S-box

Legend

<table>
<thead>
<tr>
<th></th>
<th>non protected logic</th>
<th>protected logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbox</td>
<td>blue</td>
<td>red</td>
</tr>
<tr>
<td>result</td>
<td>blue</td>
<td>red</td>
</tr>
</tbody>
</table>

- Full CMOS
- XOR ISE
- S-box ISE
- XOR + S-box ISE
- full ISE

Francesco Regazzoni
23 October 2015, Chia, Italy
Overall Software Flow

Transformation

Target Identification

Code Transformation

Information Leakage Analysis

Input Software Implementation

Sensitive Parts

Targets for Protection Example (A)

Protected Implementation Example (A)

Targets for Protection Example (B)

Protected Implementation Example (B)

sbci r21,0xfd
ld r25,Y
movw r18,r26
subi r18,0x4f

Example (A)

Protected Implementation

Example (B)
Example on Software

Instruction - time mapping of unprotected implementation

Sensitivity (Mutual information)

Clock cycle

Francesco Regazzoni
23 October 2015, Chia, Italy
Example on Software

Sensitivity (Mutual information)

Instruction - time mapping of unprotected implementation

Instruction - time mapping of protected implementation

Francesco Regazzoni
23 October 2015, Chia, Italy
Conclusions and Tips

- Initial steps for power analysis are promising
- This is just the beginning...
- Don’t forget verification!
Acknowledgments

Thank you for your attention!

mail: regazzoni@alari.ch