Presentation Outline

1. FPGA specificity and vulnerability
2. Overview of countermeasures in FPGAs
3. Protection by DPL in FPGAs
4. Protection by Masking in FPGAs
5. Conclusions
1. FPGA specificity and vulnerability

2. Overview of countermeasures in FPGAs

3. Protection by DPL in FPGAs

4. Protection by Masking in FPGAs

5. Conclusions
FPGA specificity

- Price to pay for reconfigurability:
  - Size 35X ⇒ 18X, Consumption 14X ASIC size (Kuon and all 2007)
  - Many high-gain DFFs
  - Many memories:
    - distributed: LUTs
    - embedded
  - Many DSPs
  - Many long lines and switches: Interconnect = 80% of the total area, and unknown
Vulnerability against side-channel attacks

Comparison between ASIC and FPGA in terms of power leakage:

1. **SecMat v3[ASIC]:**
   - Shared power supply between all modules

2. **SecMat v3[FPGA]:**
   - SecMat v3[ASIC] VHDL code synthesized in an Altera Stratix EPS1S25
   - Global power supply
   - 10,157 logic elements and 286,720 RAM bits for the whole SoC
   - DES alone is 1,125 logic elements (LuT4)

The power traces acquired from those three circuits are available for download from [http://www.dpacontest.org/](http://www.dpacontest.org/).
SecMat v3[ASIC]:
- Typical trace: 38 mV
- Typical DPA: 0.6 mV
- Side-channel leakage: 1.5 %
SecMat v3[FPGA]:

- Typical trace: 19 mV
- Typical DPA: 0.19 mV
- \( \Rightarrow \) Side-channel leakage: 1.0 %
Targeted strategies

- Protocol-level:
  - Most wanted since provable

- Register-Transfer Level:
  - **Masking**, boolean or algorithmic.
  - Encrypted leakage
  - Glitch-full circuits

- Netlist or implementation level:
  - **Hiding** = DPL, Dual-rail with Precharge Logic

- Degenerated counter-measures
  - Noise generator, Dummy instructions, Varying clock, etc.
if \( \approx 1 \) bit is leaked per 100 encryptions...

The FPGAs designs can take advantage of Reconfigurability to change regularly the implementation.
## Masking

### Principle
- Every variable $s$, potentially sensible, is represented as a share $\{s_0, s_1, \ldots, s_{n-1}\}$
- To reconstruct $s$, all the $s_i$ are required.
- Example: $n = 2$, $s = s_0 \oplus s_1$.

### Constraints and Drawbacks
- Leakage resistant since variables are never used plain.
- Attractive but works only fine for registers.
- Efforts done to protect also the combinational logic.
- Sensitive to Hi-orders attacks.
- Ineffective against Fault attacks.
Encrypted Leakage

\[ y = \text{DES}(x, k_c) \]

\[ k_b \rightarrow k_c \rightarrow \text{Masked DES} \rightarrow \text{Masked DFF} \rightarrow y = \text{DES}(x, k_c) \]

Side-channel: EMA, power

ASIC (tamper-proof)

\[ k_c \rightarrow k_i \rightarrow \text{Masked DFF} \rightarrow y = \text{DES}(x, k_c) \]

Side-channel: EMA, power

Personalization

NVM
Hiding by using DPL: Dual Rail with Precharge Logic

**a ↔ (a_f, a_t) DPL representation:**

- **a is VALID if** \( a_f \oplus a_t = 1 \). \( \text{VALID} = \{\text{VALID0, VALID1}\} \) or \( \text{VALID} = \{(1, 0), (0, 1)\} \).
- **a is NULL if** \( a_f \oplus a_t = 0 \). \( \text{NULL} = \{\text{NULL0, NULL1}\} \) or \( \text{NULL} = \{(0, 0), (1, 1)\} \).

**Precharge:**

**Evaluation:** (output disclosed)
A common DPL: WDDLL\(=\)Waveform Dynamic Differential Logic

A digital circuit and its WDDL equivalent

Only positive gates could be used for netlist synthesis.
Important constraints in DPL: No glitches +

No Early Evaluation

PRE/EVAL

Precharge Evaluation

$\Delta t_1$, $\Delta t_2$

Cause of Early Evaluation

No Technological Biaias

- OR consumption = AND consumption
- routing T = routing F
Presentation Outline

1. FPGA specificity and vulnerability
2. Overview of countermeasures in FPGAs
3. Protection by DPL in FPGAs
4. Protection by Masking in FPGAs
5. Conclusions
Security constraints 1/2

Logic without glitches and early propagation

⇒ Synchronization

The rules to be “synchronized”:

- **Rule 1**: Evaluation starts after all the input signals are valid.
- **Rule 2**: Precharge starts:
  1. Either after all the inputs becomes NULL\(^1\) but the outputs need to be memorized or
  2. Or before the first input becomes NULL (which does not need any memorization).

\(^1\)NULL is the value in precharge phase
Security constraint 2/2

Logic with a minimum of technological biais

- Special care at placing and routing (but the FPGA vendors give few informations)
- Use of the same logic structure for True and False (e.g. MDPL with majority gates)
- Statistical balancing

Logic resistant to fault attacks

- Detection capability or
- Resilience
Cost and Speed constraints

Logic with a minimum cost

- A few more than X2
- Use of RAMs and DSP in FPGAs

Fast speed

- speed divided by 2. Possible to be better?
The BCDL gate: Synchronization with Global Precharge

- No need of memorization as a **global precharge** PRE is faster than any inputs.
- $U/PRE$ falls to 0 $\Rightarrow$ precharge is forced immediately.
- $U/PRE$ rises to 1 $\Rightarrow$ evaluation begins after “unanimity to 1”.
- Tables T and F can be fully separated $\Rightarrow$ huge complexity gain.
Exemple of a 2-input OR gate
Robustness against FA

In-Built Robustness against Fault Attacks

- Automatically detects symmetric faults: \{VALID0, VALID1\}
- \{NULL0, NULL1\}(1 \rightarrow 0 \text{ or } 0 \rightarrow 1).
- “Error state” is propagated throughout the design \Rightarrow \textbf{Fault resilience.}

<table>
<thead>
<tr>
<th>PRECHARGE</th>
<th>Fault detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>state \neq {NULL0, NULL1}</td>
</tr>
<tr>
<td>0</td>
<td>state \neq {VALID0, VALID1}</td>
</tr>
</tbody>
</table>

"PRECHARGE" state \neq \{NULL0, NULL1\} \implies \text{Fault resilience.}
Fault Detection with DSP blocks

- based on $A \times B = (-A) \times (-B) \Rightarrow (2A + 1) \times (2B + 1) = (2\overline{A} + 1) \times (2\overline{B} + 1)$
- Allows to detect and locate either during precharge or evaluation
Area

T and F easy to implement
- Not limited to positive functions
- separable
  - 1 additional input \((U/PRE)\) + duplication\((T\) and \(F\))
  - Area of tables \(= 2.2^{n+1} < 2^{2n}\) if \(n > 2\)
  - \( \Rightarrow \) S-Box area = only 4 times the size of an unprotected one.

Total Area
\[ = DFF(\times 4) + [SYNC(a\ few\ gates) + T + F] \times n. \]

Special case: MUX driven by single rail signal
- No needs of synchronization.
Speed optimization

WDDL or Basic BCDL

Speed-optimized BCDL

Faster than other DPLs

- Evaluation time > precharge time ⇒ performances ↑
- Speed / ∼ 1.25 ↔ 1.75
results in FPGA Stratix for an AES implementation

<table>
<thead>
<tr>
<th></th>
<th>ALM</th>
<th>Reg</th>
<th>RAM</th>
<th>Max. freq.</th>
<th>Max. throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>no protection</td>
<td>1078</td>
<td>256</td>
<td>40 Kb</td>
<td>71.88 MHz</td>
<td>287.52 Mbps</td>
</tr>
<tr>
<td>WDDL</td>
<td>4885</td>
<td>1024</td>
<td>—</td>
<td>37.07 MHz</td>
<td>74.14 Mbps</td>
</tr>
<tr>
<td>BCDL</td>
<td>1841</td>
<td>1024</td>
<td>160 Kb</td>
<td>50.64 MHz</td>
<td>151.92 Mbps</td>
</tr>
</tbody>
</table>

CPA results
- Attack processed on 150000 power consumption traces.
- No subkey found for BCDL.
MIA results for different subbytes implementations

- stdcell_gf (1)
- stdcell_lut (2)
- stdcell_gb (3)
- wddl_0 (5)
- wddl_1 (6)
- wddl_2 (7)
- wddl_4 (8)
- ewddl_4 (9)
- seclib_1 (10)
- seclib_2 (11)
- seclib_4 (12)
- seclib_4ema (13)

More secure

Less secure

Noise standard deviation, denoted \( \sigma \) [V]
Comparison with other DPLs in FPGAs

- **WDDL**: Propagation of the NULL state with positive functions
- **RCDDL**: WDDL with factored logic, which amplifies the early evaluation
- **MDPL**: $T$ gate = $F$ gate = Majority, random Mask to balance the True and False networks
- **STTL**: A third wire is added to synchronize with the last stable signal.
- **DRSL**: As MDPL with a synchronization before evaluation
- **IWDDL**: Isolated WDDL with separated $T$ and $F$ networks by means of superpipelining
- **BCDL**: The logic presented here
- **MBCDL**: BCDL with mask
## Comparison with other DPLs

<table>
<thead>
<tr>
<th>Logic</th>
<th>Compl.</th>
<th>Speed</th>
<th>Robust. SCA</th>
<th>Robust. FA</th>
<th>Design Constr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>EE</td>
<td>T. B.</td>
<td>Fault</td>
</tr>
<tr>
<td>WDDL</td>
<td>*</td>
<td>&lt; 1/2</td>
<td></td>
<td>asym</td>
<td>comb</td>
</tr>
<tr>
<td>MDPL</td>
<td>*</td>
<td>&lt; 1/2</td>
<td>✓</td>
<td>asym</td>
<td>comb</td>
</tr>
<tr>
<td>STTL</td>
<td>*</td>
<td>&lt; 1/4</td>
<td>✓</td>
<td>sym</td>
<td>seq</td>
</tr>
<tr>
<td>DRSL</td>
<td>*</td>
<td>&lt; 1/2</td>
<td>partly</td>
<td>✓</td>
<td>sym</td>
</tr>
<tr>
<td>IWDDL</td>
<td></td>
<td>&lt; 1/2·n</td>
<td>✓</td>
<td>asym</td>
<td>comb</td>
</tr>
<tr>
<td>BCDL</td>
<td>**</td>
<td>&gt; 1/2</td>
<td>✓</td>
<td>sym</td>
<td>comb</td>
</tr>
<tr>
<td>MBCDL</td>
<td>*</td>
<td>&gt; 1/2</td>
<td>✓</td>
<td>✓</td>
<td>sym</td>
</tr>
</tbody>
</table>
FPGA specificity and vulnerability
Overview of countermeasures in FPGAs
Protection by DPL in FPGAs
Protection by Masking in FPGAs
Conclusions

Presentation Outline

1. FPGA specificity and vulnerability
2. Overview of countermeasures in FPGAs
3. Protection by DPL in FPGAs
4. Protection by Masking in FPGAs
5. Conclusions
FPGA specificity and vulnerability
Overview of countermeasures in FPGAs
Protection by DPL in FPGAs
Protection by Masking in FPGAs
Conclusions

Zero-offset implementation
Squeezed Leakage

ROM Hardware masking

“Zero Offset” From Waddle et al., Peeters et al..

- Activity:
  \[ A = HW[(x \oplus m) \oplus (S(x \oplus k) \oplus m')] + HW[m \oplus m'] \]

- The register data Hamming distance is:
  \[ \Delta(x) = x \oplus S(x \oplus k) \]

- The register mask Hamming distance is:
  \[ \Delta(m) = m \oplus m' \]

- Then:
  \[ A = HW[\Delta(x) \oplus \Delta(m)] + HW[\Delta(m)] \]

Masked DES implemented with ROMs.
Problem # 1: HO-attacks

Power distributions of the five possible values of $\text{HW}(\Delta(x, k))$.

Theoretic MIA attack evaluation

Table: Theoretical conditional entropy of the ROM masked DES.

<table>
<thead>
<tr>
<th>Theoretical entropies</th>
<th>The correct key</th>
<th>Any wrong key</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H(O</td>
<td>\text{HW}(\Delta(x, k)))$</td>
<td>1.3992 bit</td>
</tr>
</tbody>
</table>

Jean-Luc Danger
Problem # 2: ROM too complex for FPGAs

- Need of $2^{2n}$ memory
- Use of external Mask recomposition with USM: Universal S-Box Masking

\[ S(x \oplus k) \oplus m' > m' \]

But attackable on the combinatorial logic!
Solution #1: Squeezed leakage by encoding tables

\[ (x \oplus k) \oplus m \]

\[ B(m) \]

\[ B^{-1}(m) \]

Gates or ROM

ROM

\[ S(x \oplus k) \oplus m' \]

\[ B(m') \]
Solution #2: Squeezed leakage by encoding tables with USM

\[ S(x \oplus k) \oplus m' \]

\[ B_i(m) \]

\[ B_i^{-1} \]

\[ L \]

\[ R \]

\[ M \]

\[ E \]

\[ k \]

\[ S \]

\[ P \]

\[ L \]

\[ R \]

\[ M \]

\[ E \]

\[ k \]

\[ S \]

\[ P \]
Table 1: Complexity and speed results. “l. s.” denotes the “leakage squeezing” countermeasure.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>ALMs</th>
<th>Block memory [bit]</th>
<th>M4Ks</th>
<th>Throughput [Mbit/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unprotected DES (reference)</td>
<td>276</td>
<td>0</td>
<td>0</td>
<td>929.4</td>
</tr>
<tr>
<td>DES masked USM</td>
<td>447</td>
<td>0</td>
<td>0</td>
<td>689.1</td>
</tr>
<tr>
<td>DES masked ROM</td>
<td>366</td>
<td>131072</td>
<td>32</td>
<td>398.4</td>
</tr>
<tr>
<td>DES masked ROM with l. s.</td>
<td>408</td>
<td>131072</td>
<td>32</td>
<td>320.8</td>
</tr>
<tr>
<td>DES masked USM with l. s.</td>
<td>488</td>
<td>0</td>
<td>0</td>
<td>582.8</td>
</tr>
</tbody>
</table>
MIA results with leakage squeezing

![Graph showing MIA results with leakage squeezing]

Figure 1: Mutual information metric computed on several DES implementations.
Squeezed leakage by mask decomposition

\[ S(x \oplus k) \oplus m' \]
Distributions obtained for different $\Theta$

### addition

<table>
<thead>
<tr>
<th>Activity A</th>
<th>$HW(\Delta(x)) = 0$</th>
<th>$HW(\Delta(x)) = 1$</th>
<th>$HW(\Delta(x)) = 2$</th>
<th>$HW(\Delta(x)) = 3$</th>
<th>$HW(\Delta(x)) = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 3 4 5 6 7 8</td>
<td>3 4 5 6 7 8 9 10</td>
<td>2 3 4 5 6 7 8 9 10</td>
<td>3 4 5 6 7 8 9 10</td>
<td>3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>

### alpha

<table>
<thead>
<tr>
<th>Activity A</th>
<th>$HW(\Delta(x)) = 0$</th>
<th>$HW(\Delta(x)) = 1$</th>
<th>$HW(\Delta(x)) = 2$</th>
<th>$HW(\Delta(x)) = 3$</th>
<th>$HW(\Delta(x)) = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 3 4 5 6 7 8 9 10</td>
<td>2 3 4 5 6 7 8 9 10</td>
<td>2 3 4 5 6 7 8 9 10</td>
<td>2 3 4 5 6 7 8 9 10</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>
MIA by Squeezed leakage by mask decomposition

- Zero offset implementation
- Mask decomposition, XOR
- Mask decomposition, addition
- Mask decomposition, multiplication
- Mask decomposition, alpha

FPGA Acquisitions

SNR

Mutual Information [bit]

SNR

-13.43
Presentation Outline

1. FPGA specificity and vulnerability
2. Overview of countermeasures in FPGAs
3. Protection by DPL in FPGAs
4. Protection by Masking in FPGAs
5. Conclusions
The FPGAs need efficient countermeasures to be protected against physical attacks.

Three levels:

- **Protocol:**
  - Reconfiguration can be done in FPGAs
  - **RTL:** Masking by taking advantages of RAMs but care has to be taken against HO-DPA. Examples:
    - Leakage squeezing
    - Mask decomposition

- **Netlist:** By using DPL. Examples:
  - **STTL:** no EE, need of 3rd wire, care of P/R
  - **BCDL:** no EE, low complexity, care of P/R
  - **MBCDL:** BCDL + easy P/R
Thanks for your attention. Any question?