CZECH
TECHNICAL
UNIVERSITY
IN PRAGUE

Faculty of Information Technology

Fully asynchronous QDI implementation of DES in FPGA

Jan Bělohoubek, Jan Schmidt

belohja4@fit.cvut.cz, jan.schmidt@fit.cvut.cz
Czech Technical University in Prague
Faculty of Information Technology

12th CryptArchi Workshop – Annecy 2014
This work was supported by Hlávka Foundation

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Aim of the work

Asynchronous design methodology on FPGA

- Explore the FPGA architecture in the consideration of the asynchronous circuit implementation
- Proper SW and HW selection (preferably available in our department)
- Asynchronous DES cipher implementation
- Review DPA resistivity of the proposed design

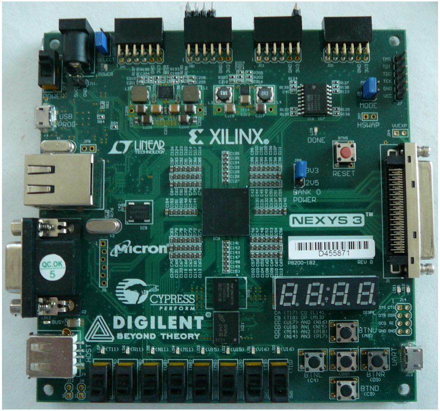
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Used technologies

FPGA and development tools selection

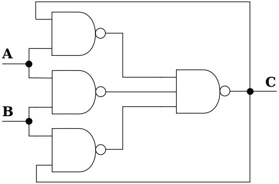
- Aim – use the standard and the most available tools
- VHDL – *ghdl* + *gtkWave*, ModelSim
- Synthesis – *ISE WebPack*
- *Nexys3* – *Spartan-6*
  - *6-LUT*
  - *size* – *# slices*
  - *available in DDD FIT CTU*



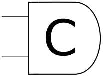
QDI and C-element

QDI is *quasi delay insensitive*

- *QDI* is the asynchronous circuits design methodology
  - → usage of C-elements allows tolerate any delay variations
  - → same delays are required only in *isochronic forks*
- *C-element*
  - → the asynchronous *hazard-free* sequential circuit
  - → holds last common value of its inputs
- Traditional C-element implementation and symbol:



C \ AB				
C	0	0	1	0
	0	1	1	1

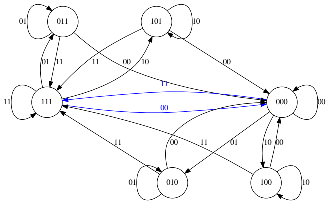


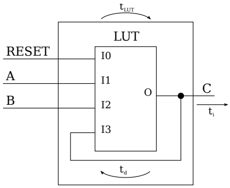
Note:

Proposed design is not fully QDI by definition.

C-element on FPGA

- C-element and its correct behavior – *model*
- C-element FPGA implementation
- Possible sizes of C-element:
  - *6-LUT*
    - → *5-inputs or 4-inputs and reset*
  - *4-LUT*
    - → *3-inputs or 2-inputs and reset*





I3	I2	I1	I0	O
			1	
		1		
		1	1	
	1			
	1		1	
	1	1		1
	1	1	1	
1				
1			1	
1		1		1
1		1	1	
1	1			1
1	1		1	
1	1	1		1
1	1	1	1	

Note:

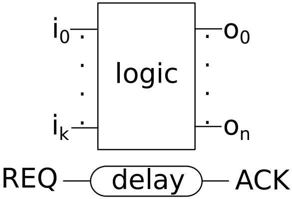
Described C-elements are correctly implemented using the automated synthesis tools.

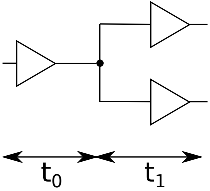
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Implementation

Timing

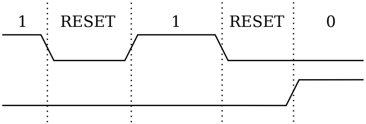


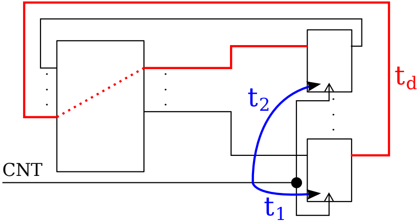


- How to implement the *delay line*?
  - difficult on FPGA – don't use delay lines, if possible
  - if necessary – bring the signals out and create the delay line outside the FPGA
- *QDI* – how to reach same delay in isochronic forks?
  - use clock-buffers

Implementation

Signalling, essential hazard





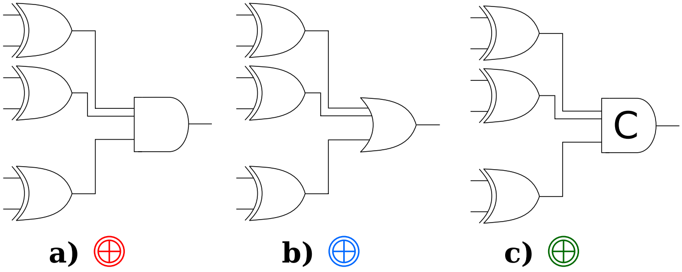
- It's difficult to set up delays on the FPGAs
  - → *QDI* is the solution
- Data signalling (*dual-rail logic*) and completion detection circuits
- *QDI* – which forks are isochronic?
  - → forks driving one group of flip-flops

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Implementation

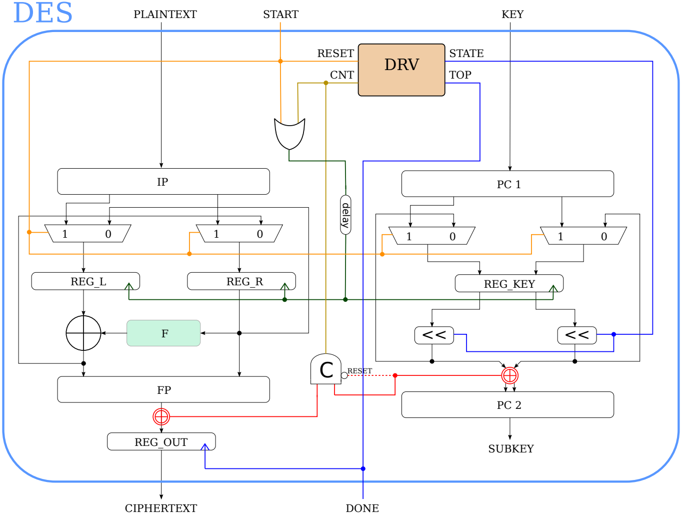
Completion detection



a) completion detection sensitive on the slowest (↑) and the fastest (↓) signal
b) completion detection sensitive on the fastest (↑) and the slowest (↓) signal
c) *DI* completion detection (sensitive on the slowest signal → *delay insensitive*)

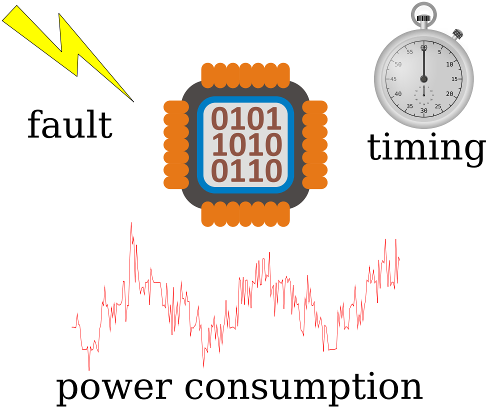
Implementation

DES



Vulnerability

Side channel attacks



Vulnerability

Measurement methodology

a: 0 1 0 1 1 0

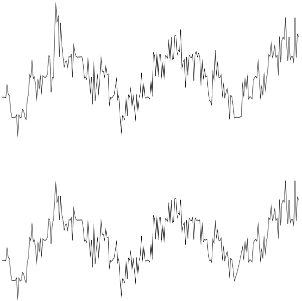
. . . . .

$d_H(a,b)$

b: 0 1 1 1 0 1

???

a:

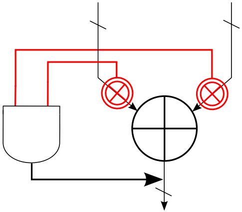


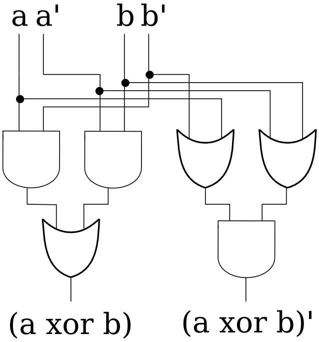
corr(x,y)

b:

Vulnerability

Global and local synchronization in the datapath



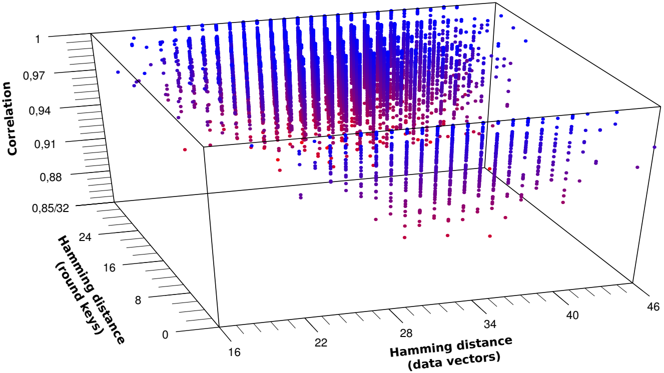


Note:

Global synchronization means hold all outputs in state 00 *until all inputs leave* 00.

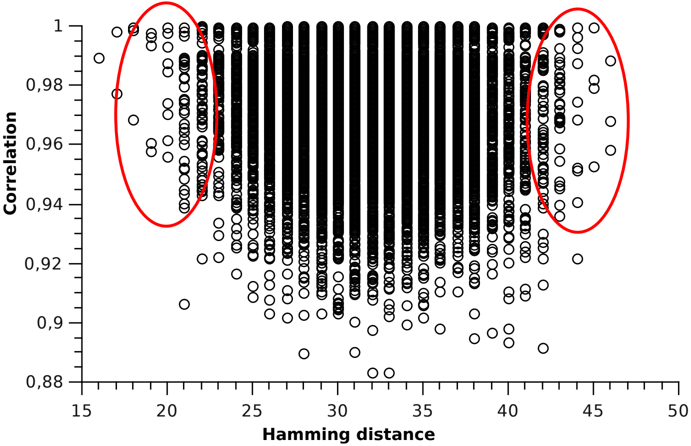
Vulnerability

Measurement results – local synchronization of datapaths



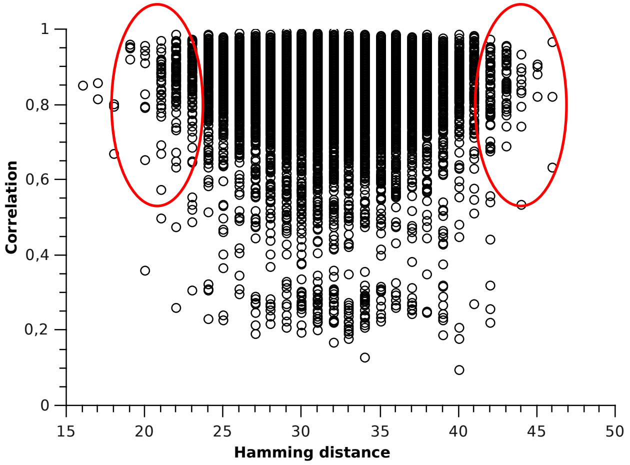
Vulnerability

Measurement results – local synchronization of datapaths



Vulnerability

Measurement results – global synchronization of datapaths



CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Vulnerability

Measurement interpretation

- Inverse and similar vectors has similar power traces $\rightarrow$ dual-rail logic is automatically mapped relatively symmetrical
- Global synchronization leads to worse correlation (period is the Achilles heel)
  - $\rightarrow$ synchronization is derived from the fastest or the slowest signals

Bad news (?)

Vectors are very similar or inverse $\implies$ power traces are similar

Good news

Power traces are similar $\not\Longrightarrow$ vectors are very similar or inverse

Future work

- *Completion detection* reimplementation using C-elements
  - → robust, but sensitive on the slowest signal
  - → larger overhead
- *Completion detection* reimplementation using $\oplus$ and $\oplus$ linked together using C-element
  - → attempt to move closer to the average period in all cases (period will be influenced by the slowest signal from one group and the fastest from the another)
  - → delay line can mask delay variations (same like in proposed implementation)
- Review the *fault-attack* vulnerability
  - → is the natural *QDI* resistivity sufficient?

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Results

- Asynchronous *DES* successfully implemented on FPGA
- Automated synthesis tools can be used when the appropriate methods are followed
- If two power traces are similar, it can't be estimated that the processed data are similar too $\rightarrow$ we assume failure of trivial attacks, but we cannot exclude the possibility of success when using the advanced *DPA* techniques
- Global synchronization of datapaths will probably increase the vulnerability
  - $\rightarrow$ vulnerability is caused by the derivation of synchronization pulses from datapaths

$\Longrightarrow$ Main vulnerability is in the timing area (of course, timing influences the power traces)