SeFPGA: a custom FPGA robust against side-channel

Jean-Luc Danger, Emna Amouri, Shivam Bhasin, Yves Mathieu

Cryptarchi 2014

Table of contents

Table of contents

Why to secure FPGAs ?

FPGA is a must-have technology for secure applications

- Low-to medium size markets :
  - Secure storage, Smart car, Military appliances, Connected objects, Home automation,...

But many potential threats

- Physical attacks :
  - Side-channel Analysis
  - Fault Injection
- Attacks on the configuration port
  - Bitstream illegitimate copy
  - Bitstream modification
  - Exploit Backdoors
- Unknown architecture $\Rightarrow$ How can we trust it ?

Physical attacks on the application

FPGA can be easier to attack

- Higher power consumption vs ASIC
- Metallic packages are easy to uncover
- The information can be spread over the big interconnexion area
- Bias due to unknown routing
  - Imbalance in Differential logic (DPL) countermeasures
  - Glitches in masking implementations

Attack on the configuration port

Bitstream security

- Confidentiality : it can be read and re-used by an adversary
- Integrity : it can be modified by an adversary
- Authentication : it has to be downloaded by a trusted user

⇒ Cryptography

- But physical attacks are still possible

Unknown architecture

Many undocumented resources

- Interconnexion
- Spare resources (cells, RAMs,...) to increase the production yield
- Many I/Os could be used to read internal information

⇒ Can we fully trust a commercial FPGA ?

- Your opinion ?

How to secure FPGAs ?

Against Physical attacks of the implementation

- Knowledge of the internal architecture
- Topology intrinsically adapted to protections
- Dynamic reconfiguration for resilient functions

$\Rightarrow$ Against attacks on the bistream

- Cryptography with
- Protections against physical attacks

Table of contents

Why a custom FPGA ?

Open structure

- Everything is under control of the user $\Rightarrow$ *greater trust*
- Full White Box analysis to understand the leakage

Optimized resources to implement protections

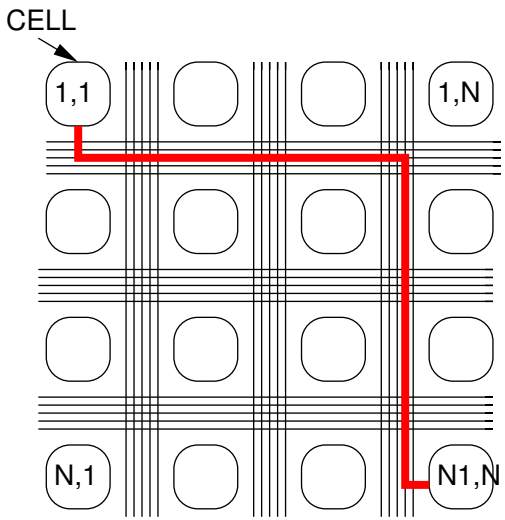
- Symmetry axes for balancing
- Dynamic configuration
- Dedicated cells

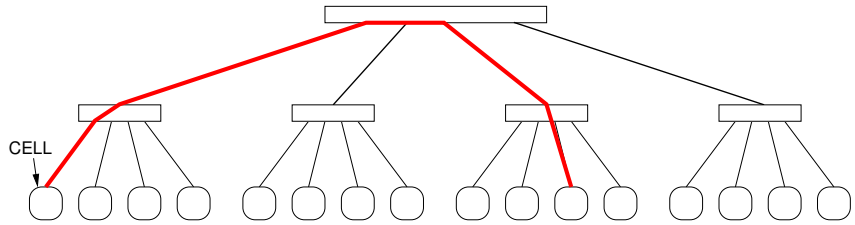
ad-hoc CAD tools

- Balance of DPL
- Glitches removal

Choice : Tree-based FPGA

”SeFPGA” ANR Project with LIP6





MESH of $N \times N$ cells
Routing complexity in $O(N)$

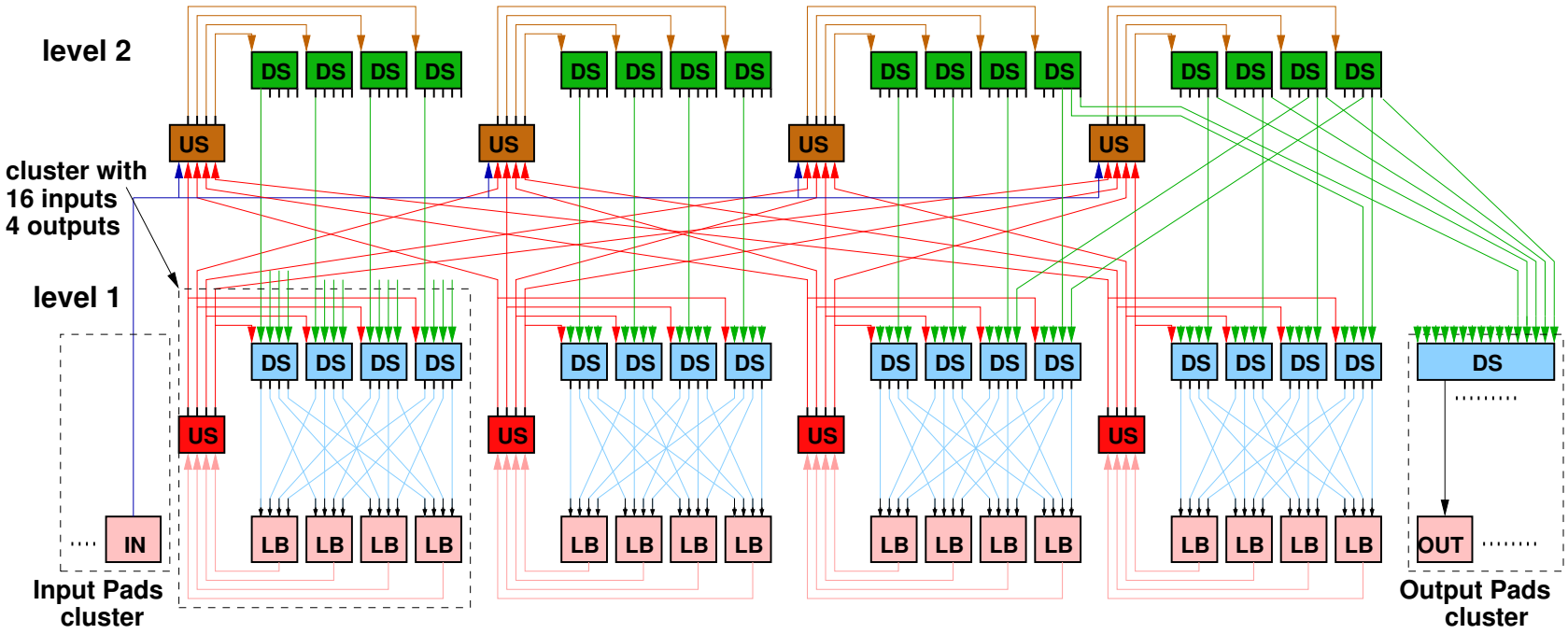
TREE of N^2 cells
Routing complexity in $O(Log_B\ N)$

Tree-based FPGA

Properties

- More timing determinism
- Two independant networks without any possible conflict
  - Up an Down
  - Each composed of Upstream MUXes and Downstream MUXes
- Facilitates Dynamic configuration
  - No care to isolate the current reconfigured cell

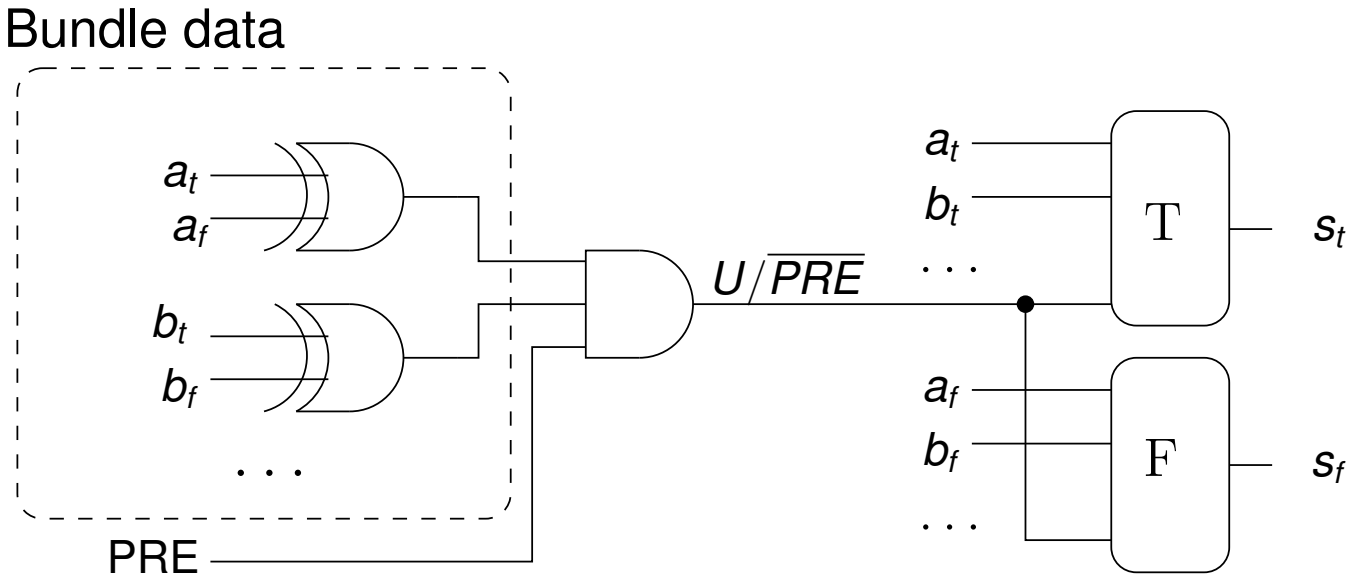
Tree-based FPGA



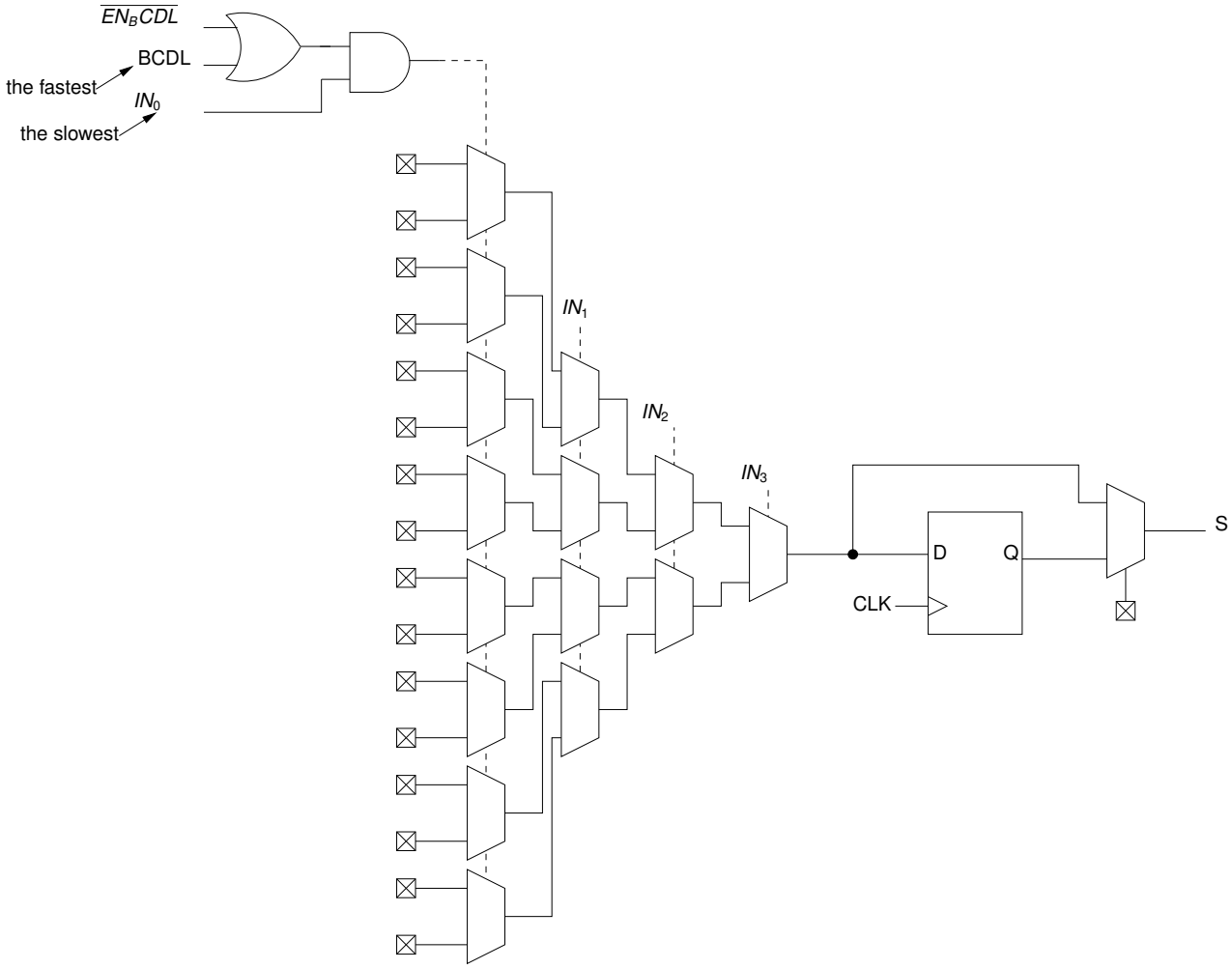
Cell Choice

LuT 4

- Allows the designer to use existing Synthesis tool for LUT4 (e.g. Yosys)
- Facilitates BCDL porting : Special syncronization gate



Tree-FPGA cell



Tree-FPGA Routing problem

Problem

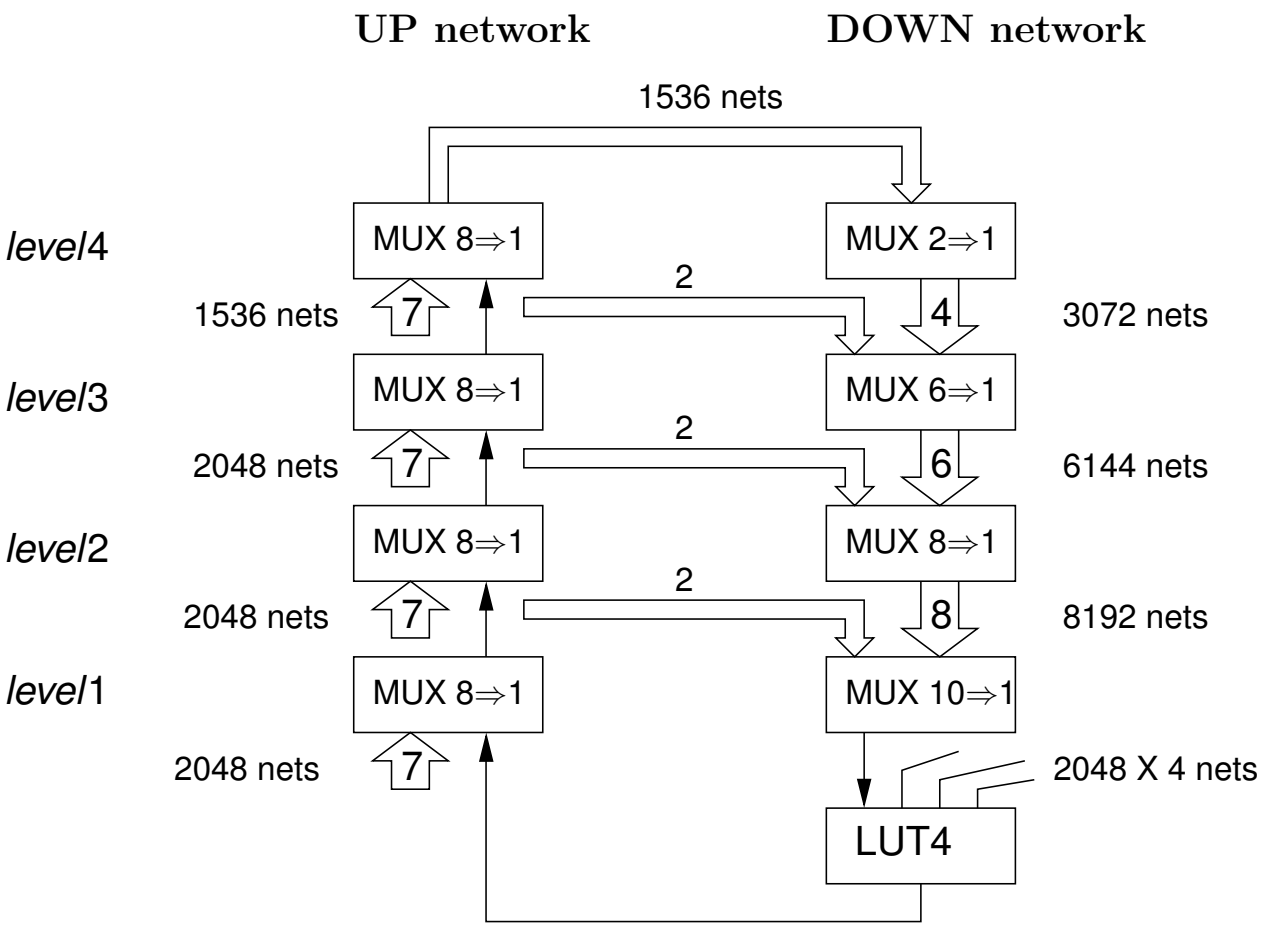
Very difficult to route in a 2-D network

- 2048 cells with Base=8
- 4 hierarchical levels with $8 \times 8 \times 8 \times 8 \times 4$ nets at each level
- These nets are crossing increasingly when going to the upper levels

Solutions

- Nets are depopulated at upper levels : $8 \times 8 \times 8 \times 6 \times 3$ nets without compromising the routing potential.
- MESH of TREE topologies could be used for bigger sizes.

Depopulated networks



Routing Method

Still many nets to route, even with depopulation

For 2048 cells : 34.812 nets in total

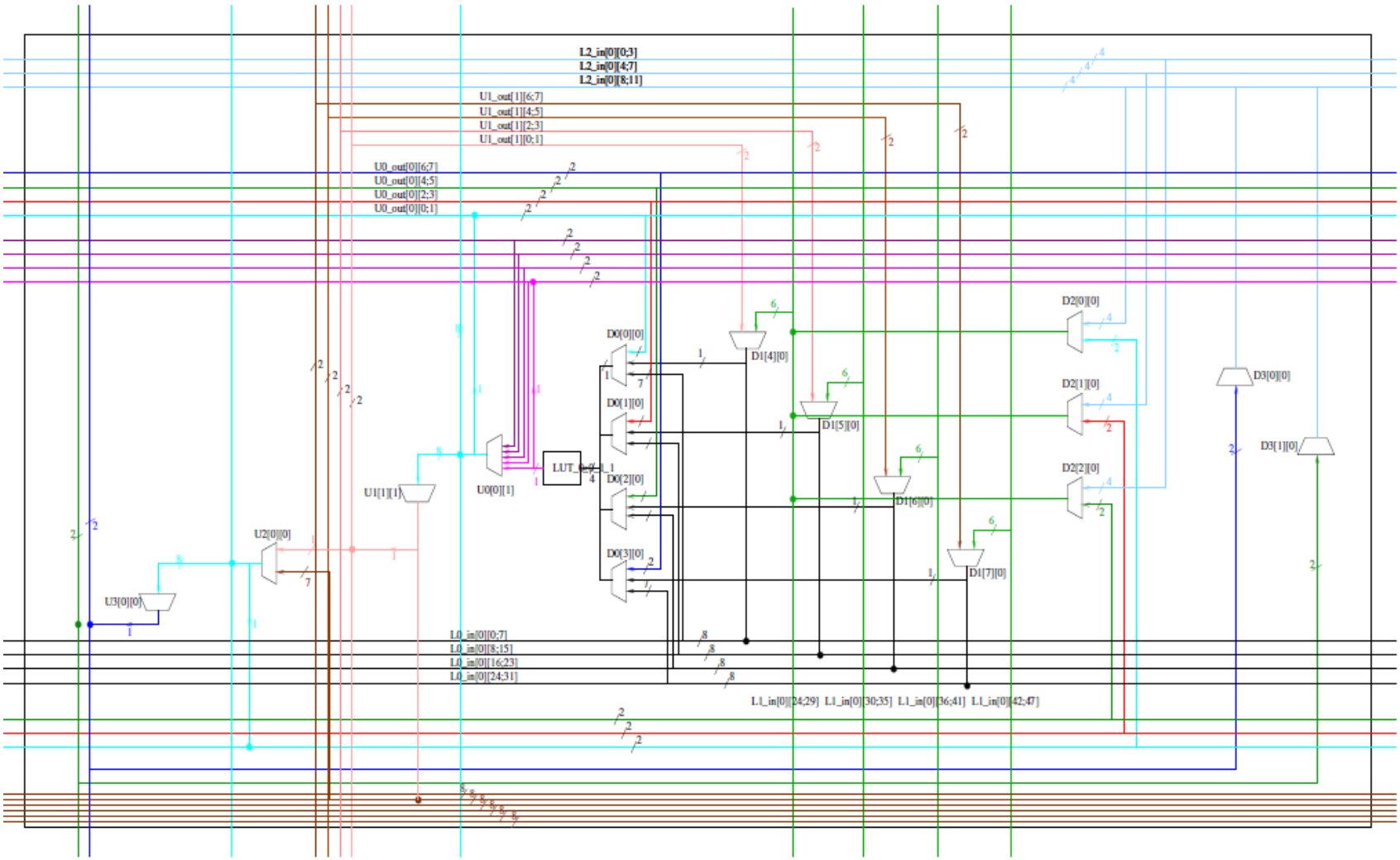
- 8.192 metal lines of length ’1’ in metal 2-3
- 22.528 metal lines of length ’8’ in metal 3-4-5
- 11.276 metal lines of length ’64’ in metal 3-4-5

Method : use a Basic Tile

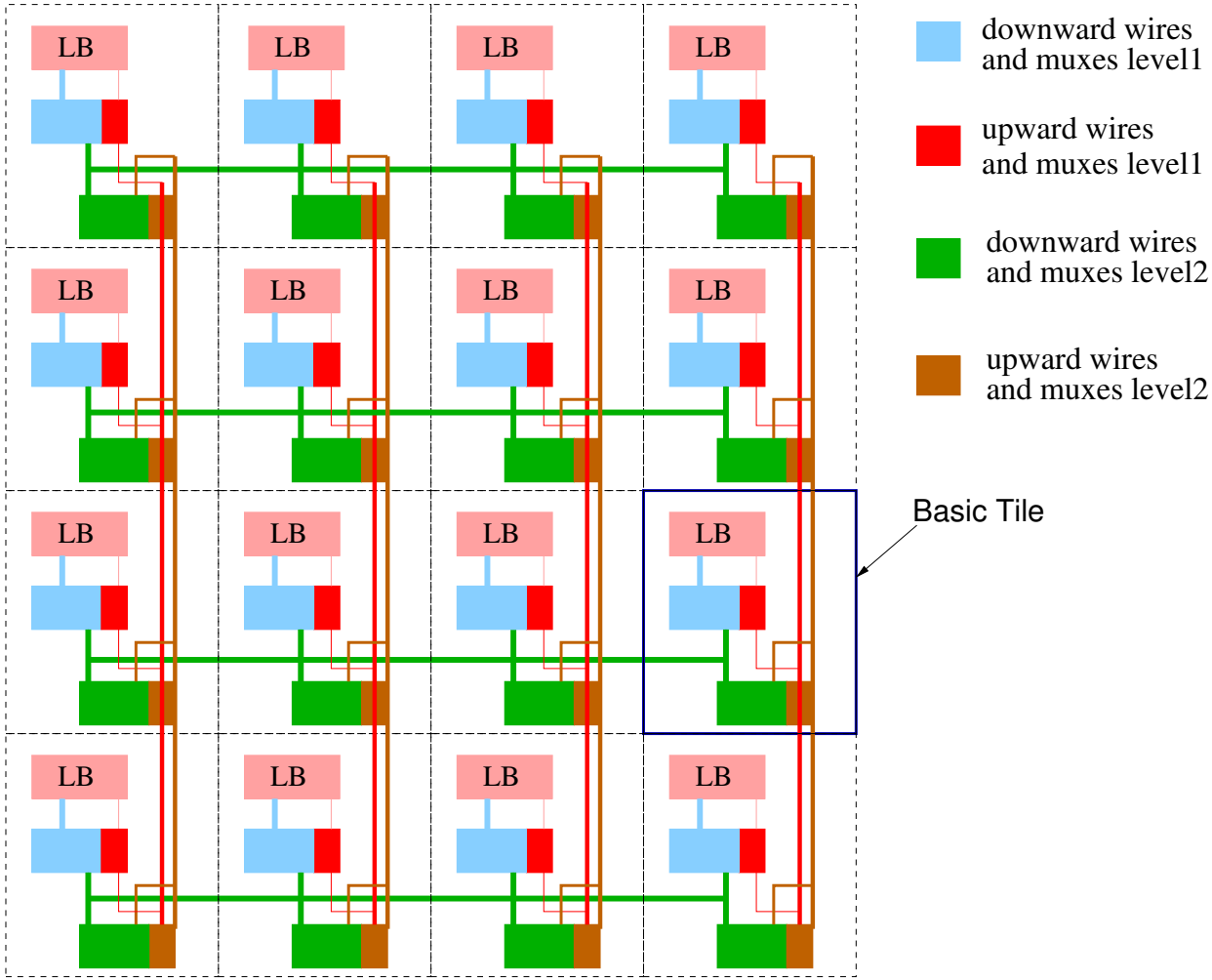
Which includes :

- A LUT4 cell
- Its associated 4 levels of DS MUXes
- Its associated 4 levels of US MUXes

Basic Tiles Schematics

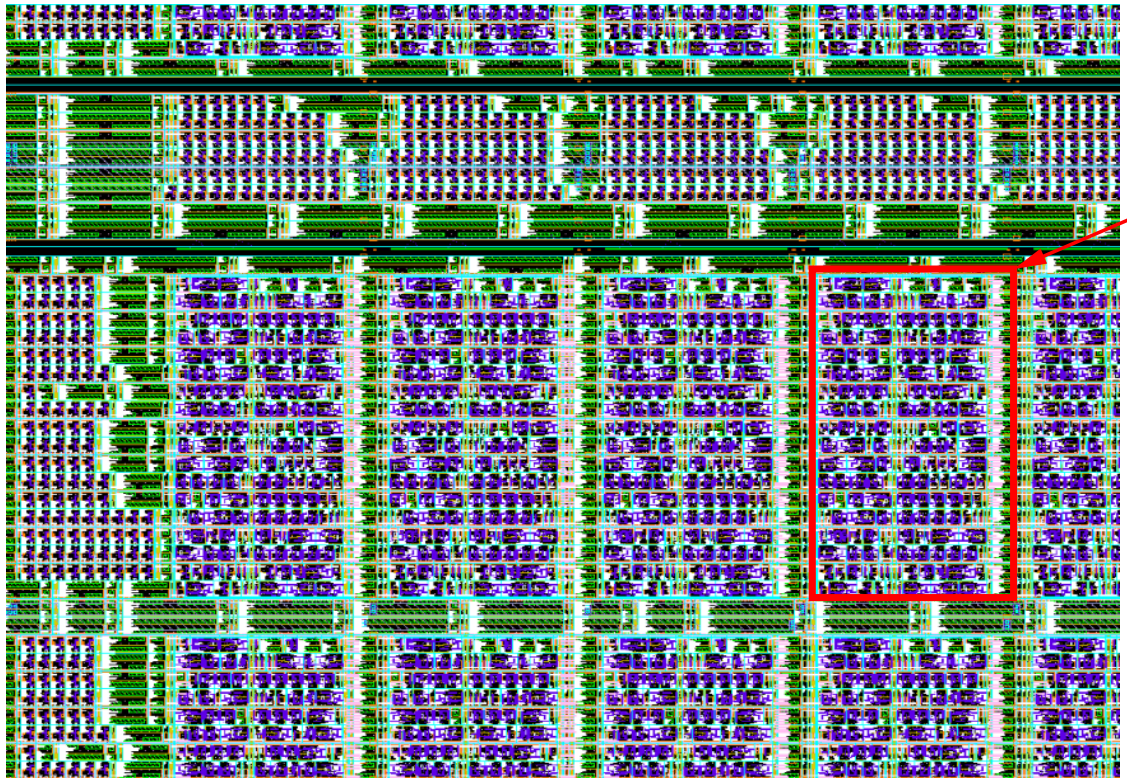


Layout principle



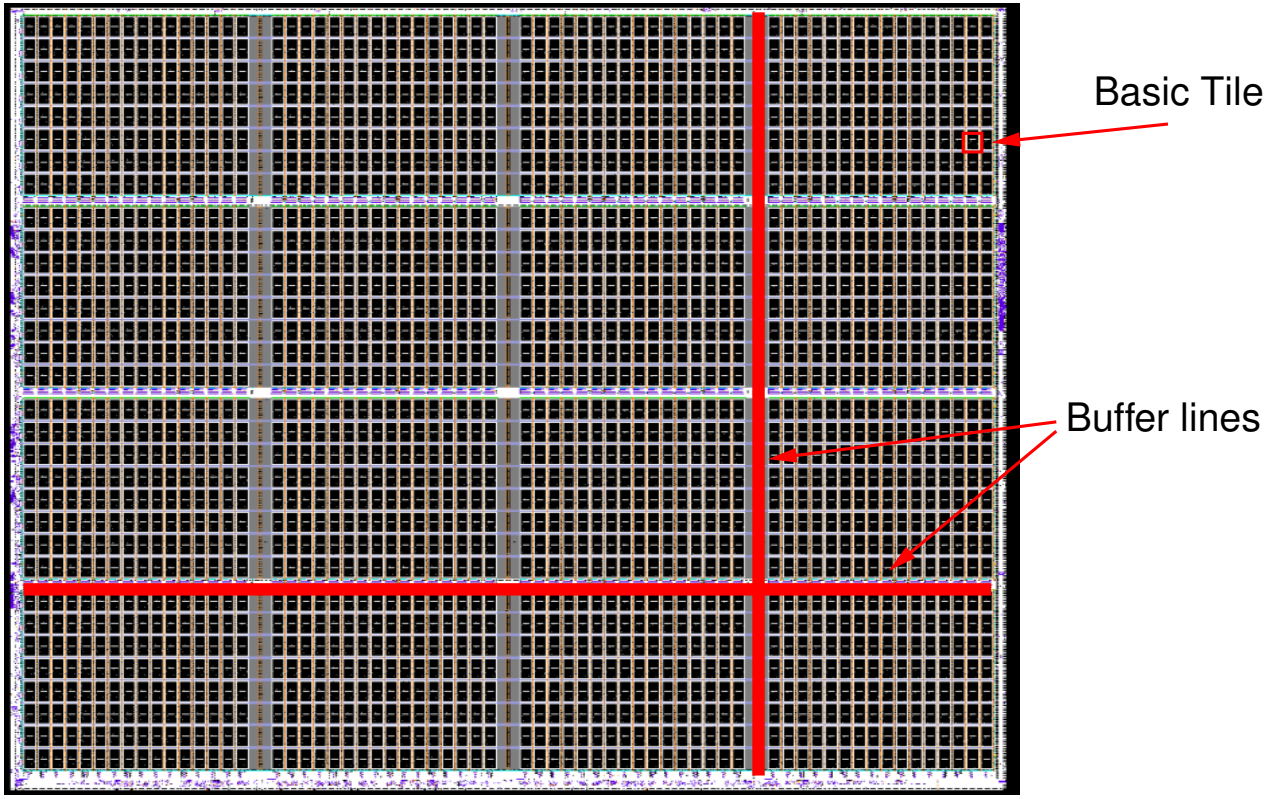
Layout on STM 65nm technology

- Use Standard cell library (no custom cells)
- Basic Tile : use Encounter Auto P/R
- All other levels : use Encounter P/R scripts



FPGA Core Layout

- Use of buffers rows and columns for long lines



SeFPGA Prototype circuit

Resources

- 2048 LUT4 cells
  - 4 hierachical levels $8 \times 8 \times 8 \times 4$ with depopulation
  - enough to implement PRESENT in WDDL or BCDL
- 60 I/Os
  - Interface to get 64 INs at level 3 and 64 OUTs at level 2
  - 3 global signals : CLK, RESET, BCDL

Configuration

- 71 latches/Tile, 1 Tile = LUT4+DS/US $\Rightarrow$ 142Kbits in total
- Dynamic reconfiguration
  - Random access at Byte level, 16 Bytes/Tile

SeFPGA Layout

- Fabrication by CMP with STM 65nm technology
- $4\mu m^2$

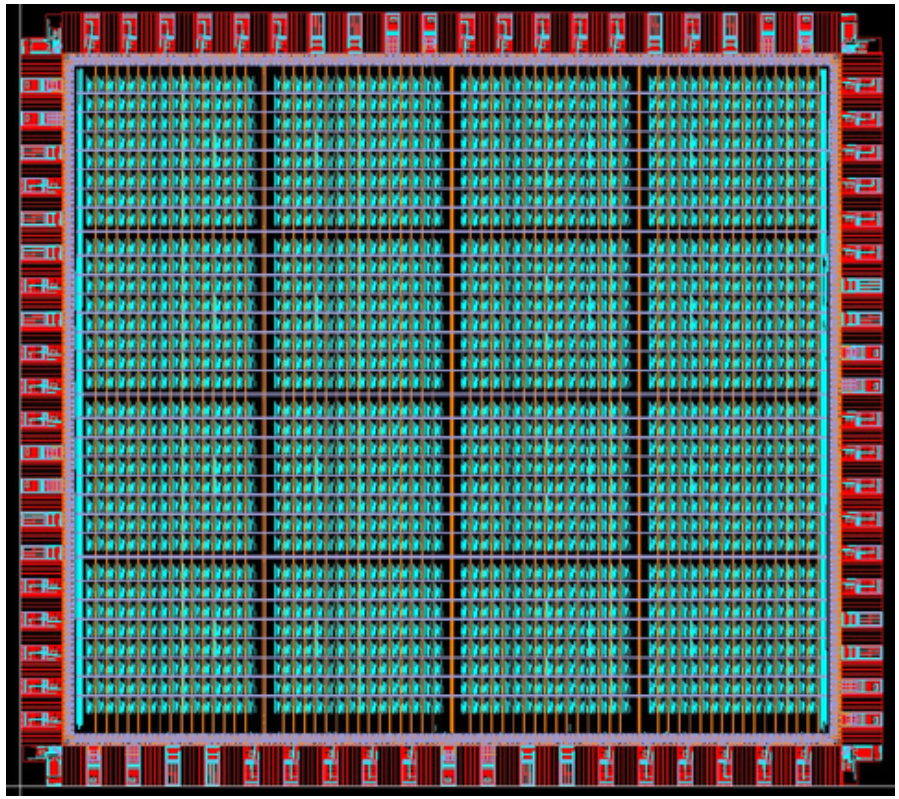
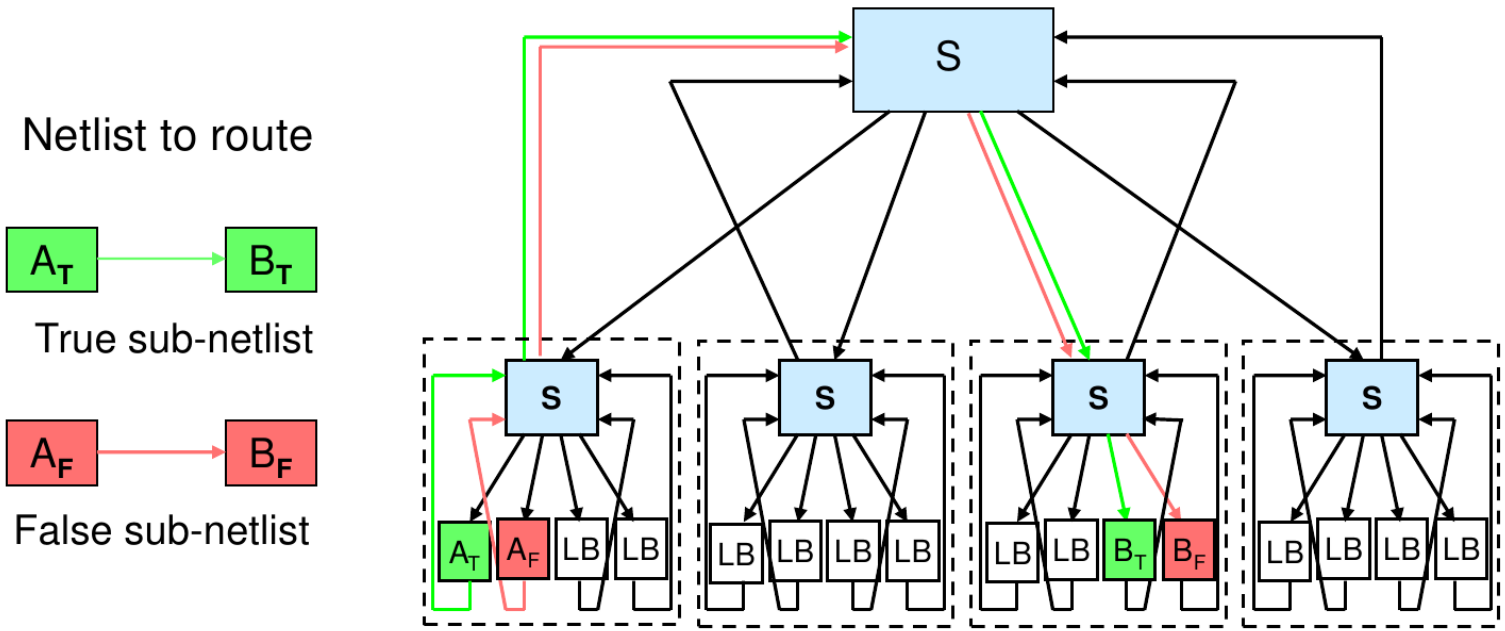


Table of contents

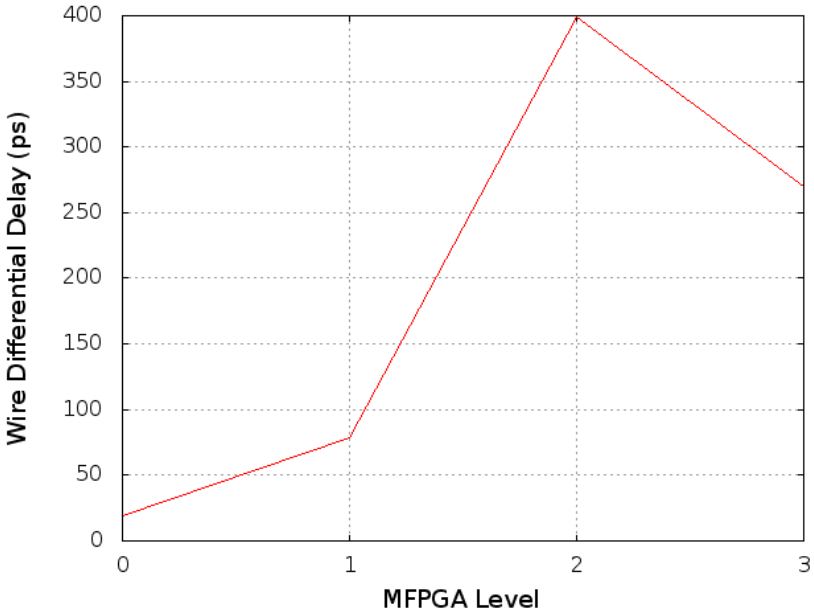
DPL Placement

Adjacent placement

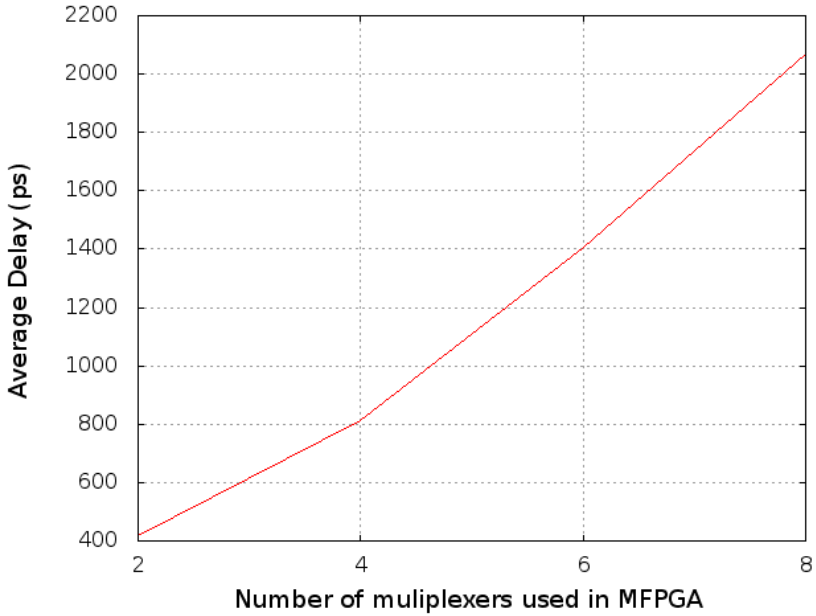
True and False gate are placed in the same cluster



Imbalance factors



Average differential Routing Delay vs hierarchical levels



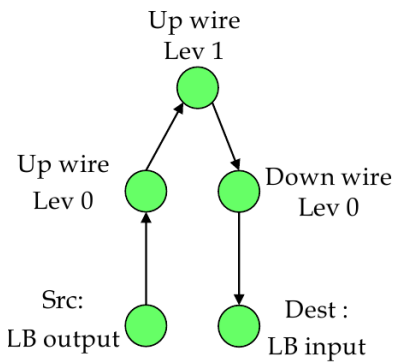
Average delay vs number of MUXes

Routing strategies

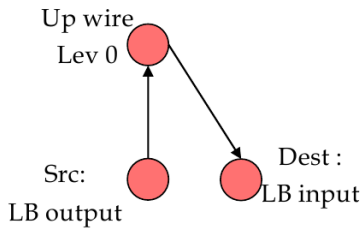
Extension to Pathfinder algorithm

2 strategies : Resources Balance and Timing Balance

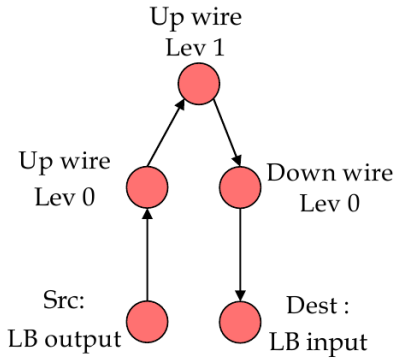
Resources Balance :



True connection routed



False connection routed with Pathfinder router



False connection routed with Resources-Balance-Driven

Timing Balance : Balance the timing and mitigate the resources

Routing results

TABLE: Routing Results

P. & R. Strategy	($\Delta delay$) (ps)			Res_imbalanced connections nb.
	Max	Mean	Std Dev	
Unconstrained P. & R.	1940	188	240	368
Adjacent P. & Res_Balance_Driven R.	694	90	105	0
Adjacent P. & Timing_Balance_Driven R.	114	26	23	0

$$\Delta delay = |delay(TRUE) - delay(FALSE)|$$

Analysis results of SeFPGA

TABLE: Side Channel Attack Results on Unprotected PRESENT

	Minimum Traces to Disclose (MTD) the Secret Key															
SBox	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Unprotected	1206	1108	694	694	500	1310	1498	694	500	1000	596	213	896	1305	800	1213

TABLE: Side Channel Attack Results on PRESENT WDDL

	Security Gain															
SBox	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Unconstrained	8	-	-	-	-	-	66	-	120	17	-	-	24	5	2	70
Res. Bal.	-	16	46	-	182		122	-	-	-	27	-	-	34	-	-
Timing Bal.	-	-	-	-	-	16	36	-	-	9	2	-	-	-	-	-

Problem : unused MUXes are not disabled $\Rightarrow$ ghost activity

Conclusions

Concept of Secure FPGA against SCA :

- Dedicated architecture
  - TREE topology
  - Open and scalable structure
  - Dedicated CAD tools to balance DPL
- Dynamic reconfiguration