



**POLITECNICO  
DI MILANO**

# FPGA: back to the future in the reconfigurable computing domain



16 Maggio, 2014  
Politecnico di Milano  
POuL, C.I.1

**Marco D. Santambrogio**  
[marco.santambrogio@polimi.it](mailto:marco.santambrogio@polimi.it)



The process of physically altering the location or functionality of network or system elements. Automatic configuration describes the way sophisticated networks can readjust themselves in the event of a link or device failing, enabling the network to continue operation.



*Gerald Estrin, 1960*



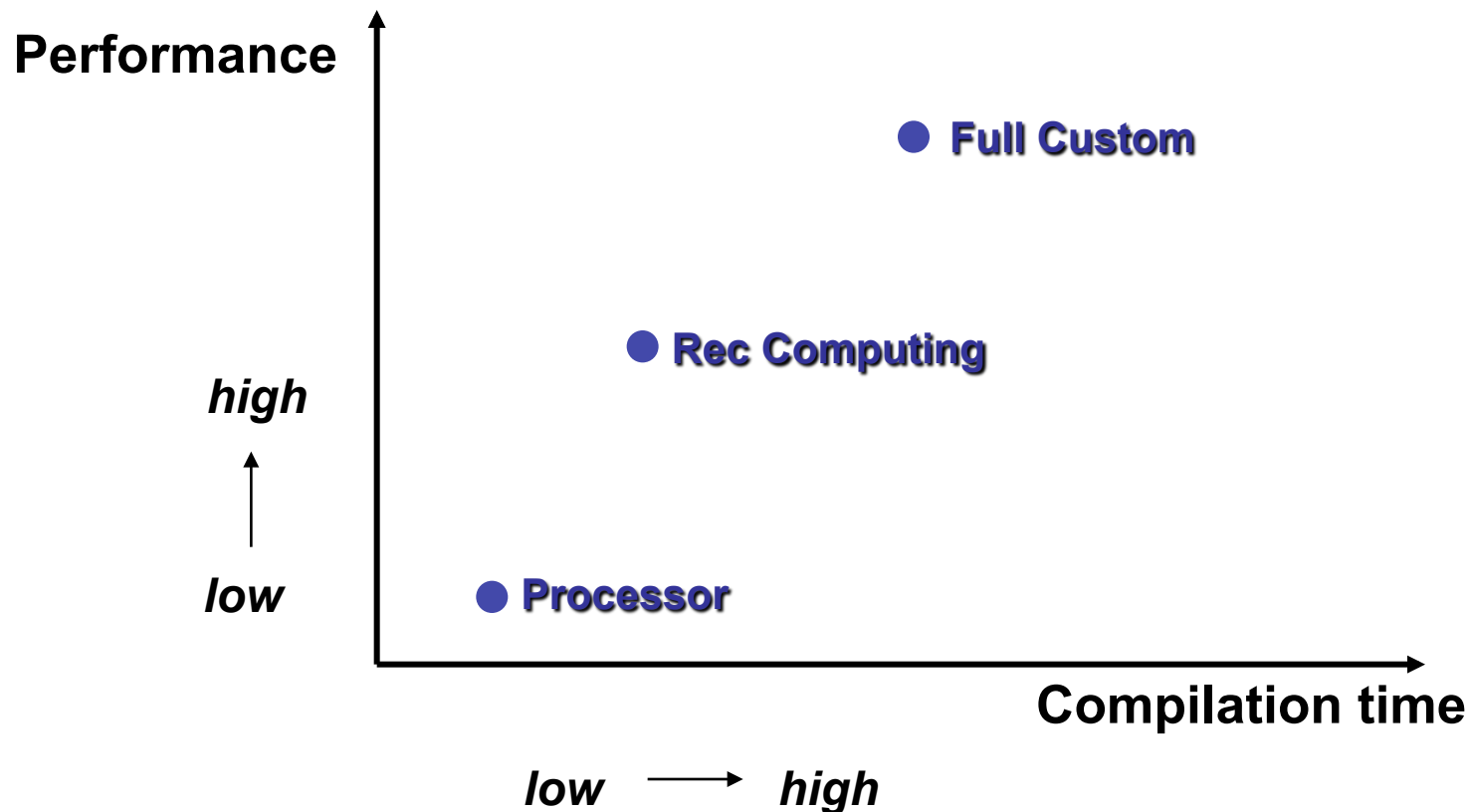


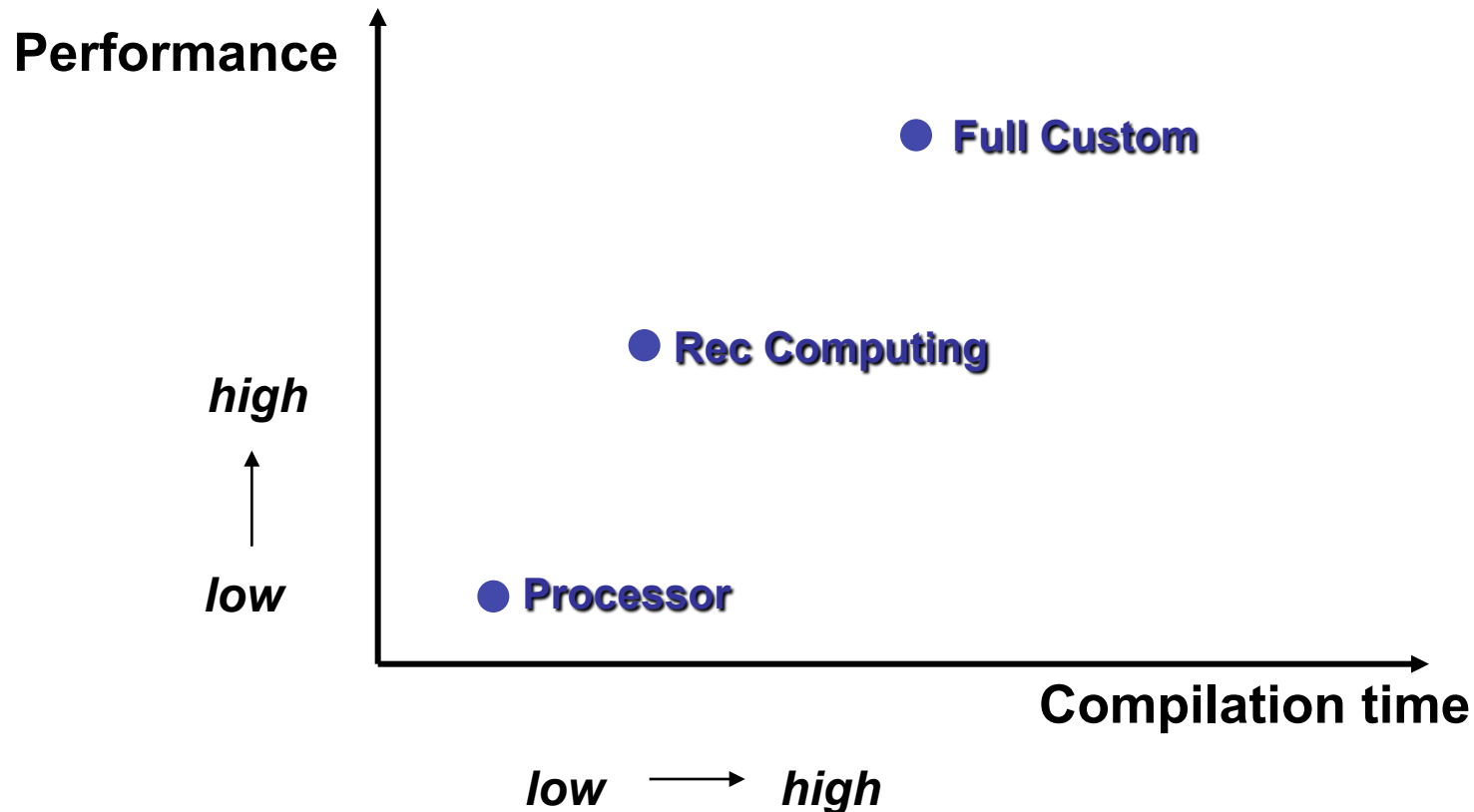
- A bird's eye view on the Reconfigurable Computing
- FPGA
- The roadmap
- What's missing



- A bird's eye view on the Reconfigurable Computing
  - The RC dawn and the FPGA revolution
  - Some !FPGA architecture
  - The accademic efforts
  - Choose the optimal hardware platform for a given application
- FPGA
- The roadmap
- What's missing





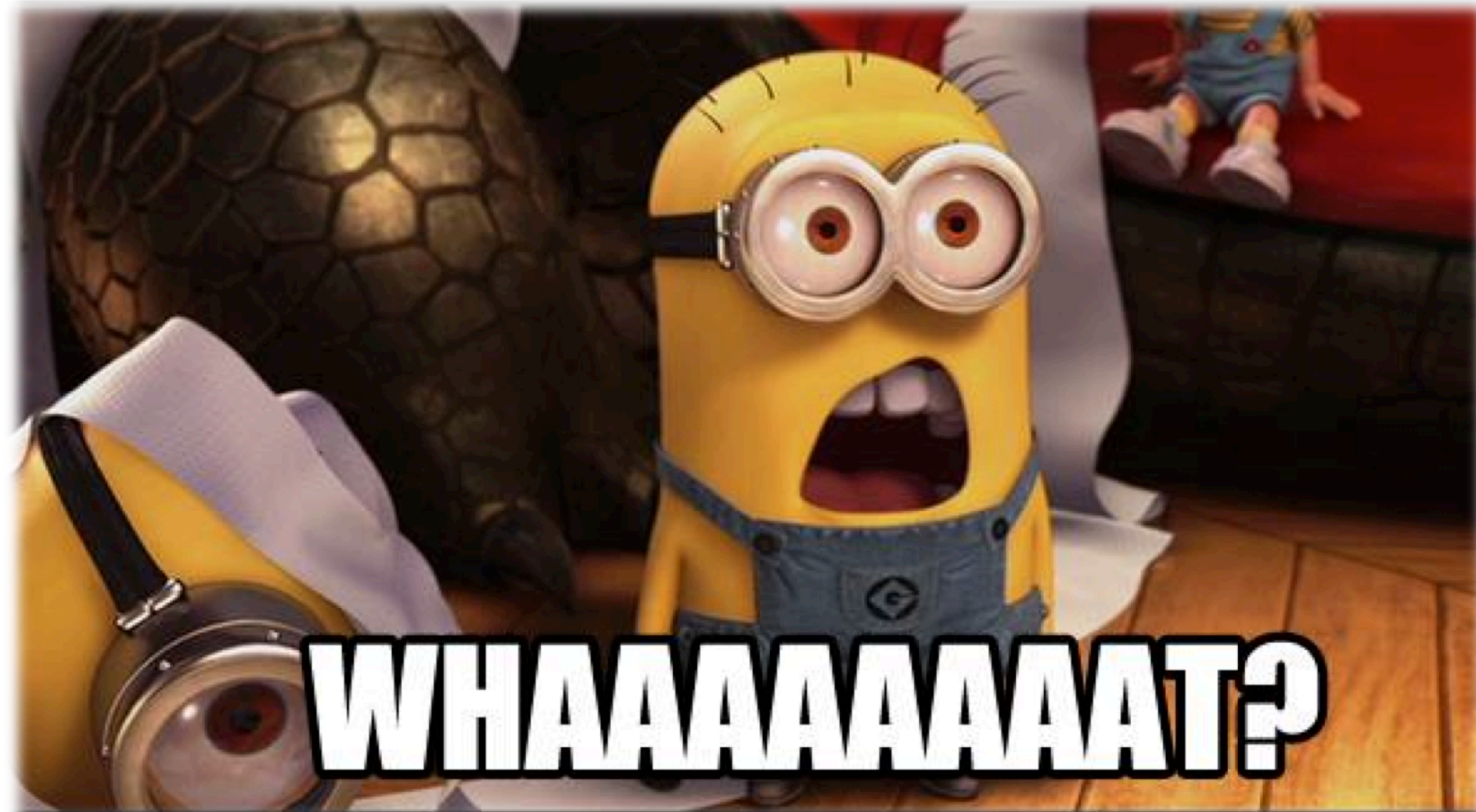


Reconfigurable computing is defined as the study of computation using reconfigurable devices

*Christophe Bobda, 2007*



# Such a definition!

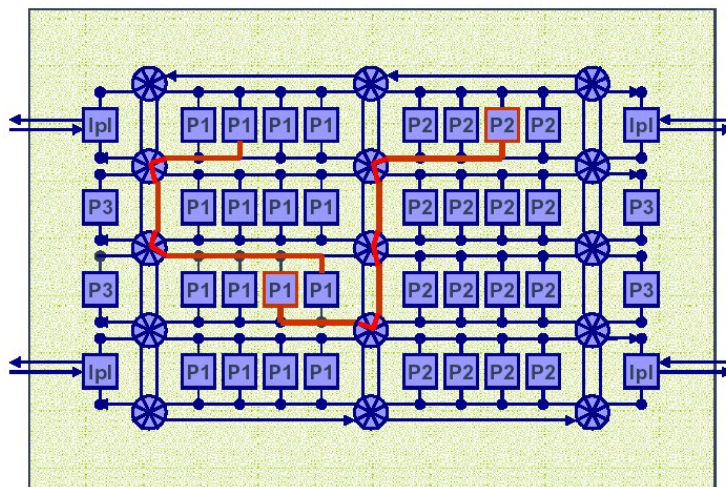




- The Estrin Fix-Plus Machine, 1959
- The Ramming Machine, 1977
- Hartenstein's *XPuter*, 1980
- mid-1980s: the FPGA revolution/era
  - The PAM Machine, SPLASH II, PRISM, Garp, DISC, DPGA



- The Pact XPP Device
- The NEC-DRP Architecture
- The picoChip Reconfigurable Device
- PicoChip solution:
  - Array of heterogeneous processors
  - Communication flexibility between processors achieved through reconfigurable technology



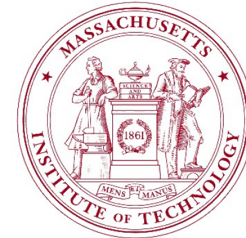
**picoChip**  
flexible wireless

 Array Processing Element

 Switch Matrix

 Inter-picoArray Interface

- The Reconfigurable Architecture Workstation (RAW) – MIT
- The Matrix Architecture – MIT
- The Reconfigurable Multimedia Array Coprocessor (REMARC) – Stanford
- MorphoSys – University of California, Irvine
- Chimaera – Northwestern
- PipeRench – CMU
- RaPiD – University of Washington
- Garp – UC Berkeley
- Bee2 – UC Berkeley





## What are the drivers for this choice?

- Time: How long does it take to compute the answer?
- Area: How much silicon space is required to determined the answer?
- Costs: How much does it costs (performance, \$)?
- Power: How much does it consume?
- Processor generally fixes computing area. Problem evaluated over time through instructions.
- FPGA can create flexible amount of computing area. Effectively, the configuration memory is the computing instruction.
  - Flexibility means variety, which means different kinds of reconfiguration





# Reconfiguration in everyday life



**Soccer**  
(Partial - Static)



**Football**  
(Complete - Static)

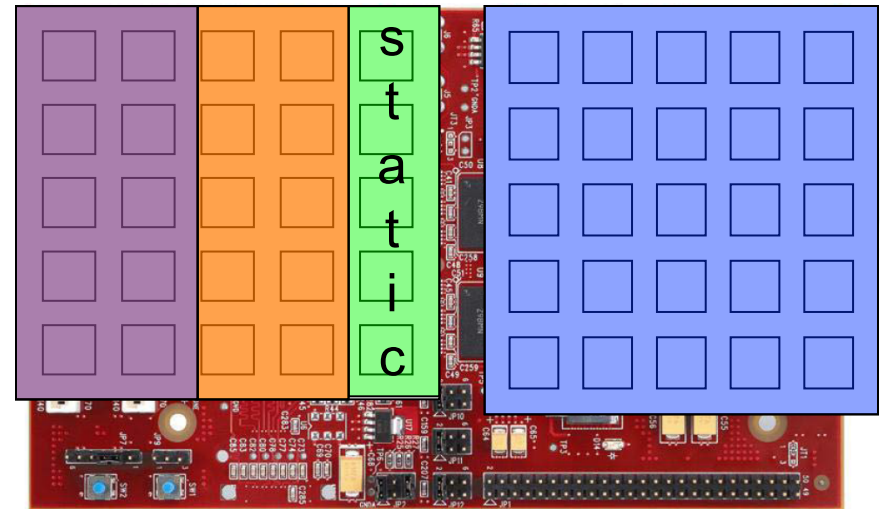


**Hockey**  
(Partial - Dynamic)





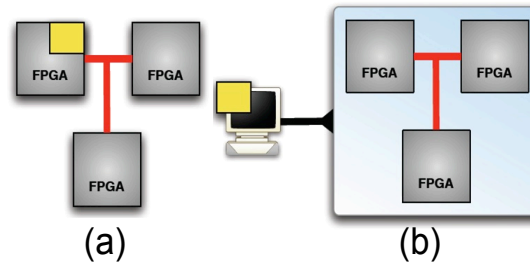
## Embedded Partial VS Complete




- SoC (System on Chip)
  - Embedded Vs External
  - Complete Vs Partial
  - Dynamic VS Static

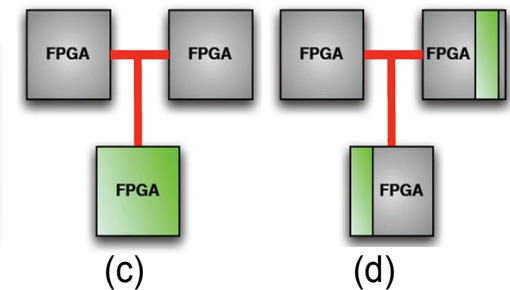
- SoMC (System on Multiple-Chip)
  - Embedded Vs External
  - Complete Vs Partial
  - Dynamic VS Static

## Who



 = Reconfiguration controller / Reconfigurator

## Complete/Partial



 = Reconfigurable module



# Reconfiguration: some applicative scenarios

**Structural  
modification**



**Recovery  
from a  
damage**



**Behavioral evolution**



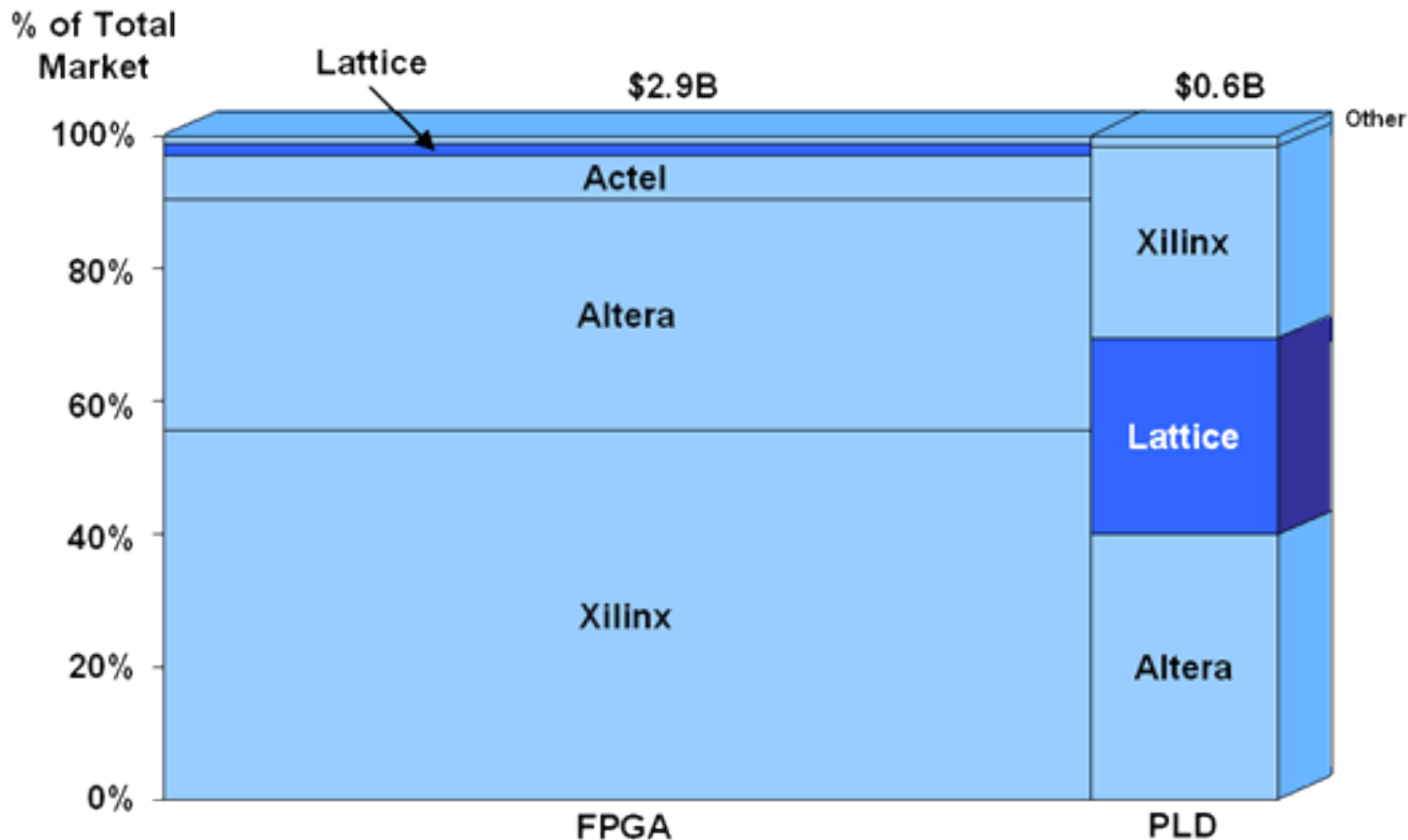


- A bird's eye view on the Reconfigurable Computing
- **FPGA**
  - Technology
  - CLB, Slice, LUT
  - Frame
  - Configuration bitstream
- The roadmap
- What's missing

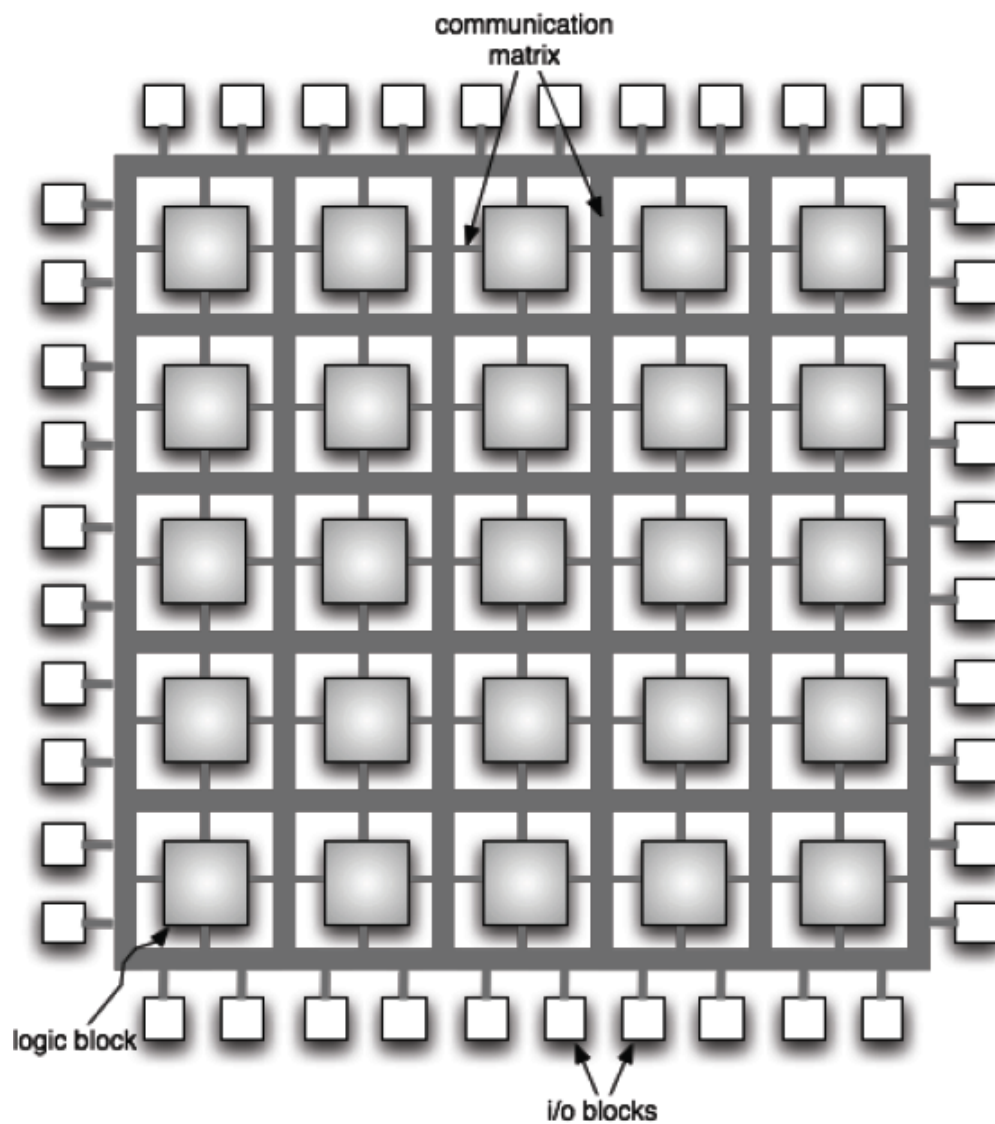


# Commercial FPGA Companies

2007

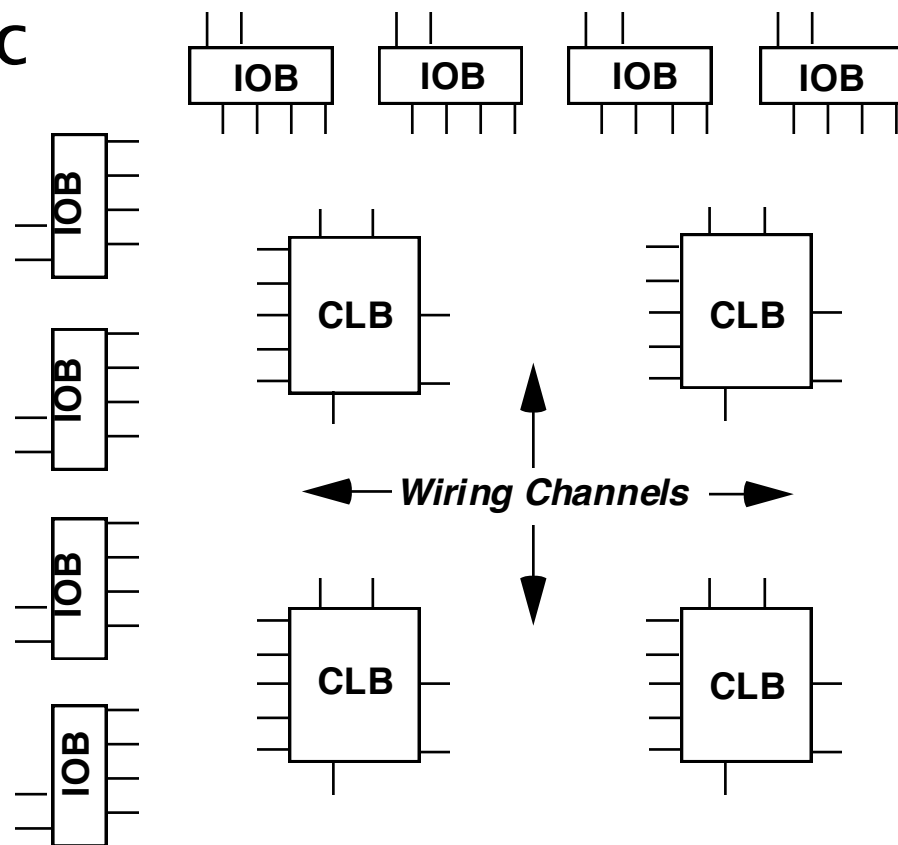


Lattice official webiste





- CLB – Configurable Logic Block
- Built-in fast carry logic
- Can be used as memory
- Three types of routing
  - direct
  - general-purpose
  - long lines of various lengths
- RAM-programmable
  - can be reconfigured



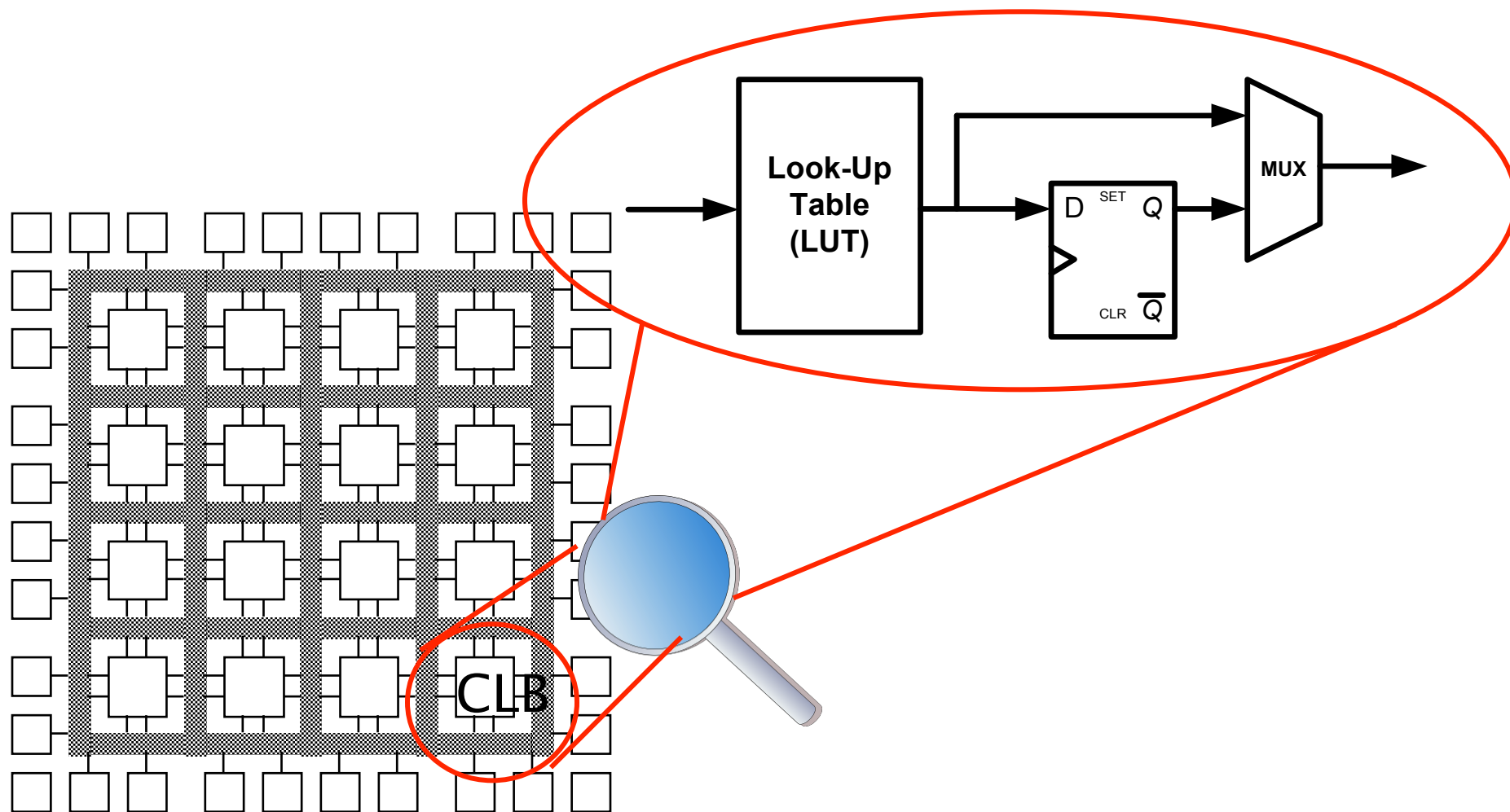


- CLB made of Slices
  - sVirtex-E 2-slice
  - VIIP 4-slice
- Slices made of LookUp Tables (LUTs)
- LookUp Tables
  - 4-input, 1 output functions
  - **Newest FPGA 2 6-input 2 output**





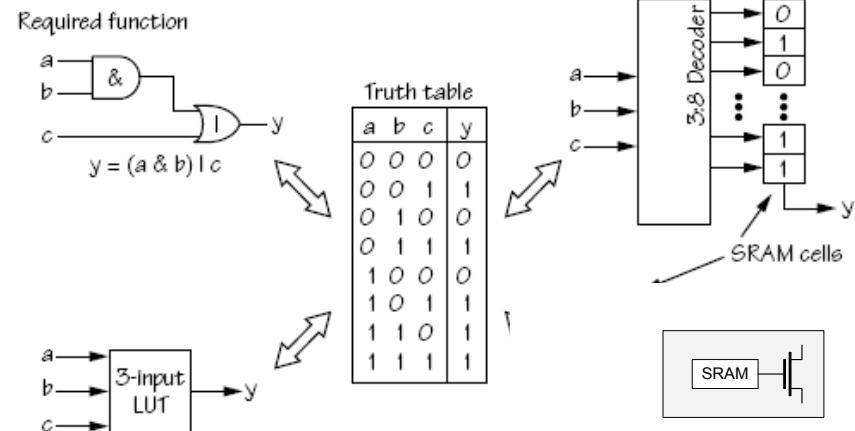
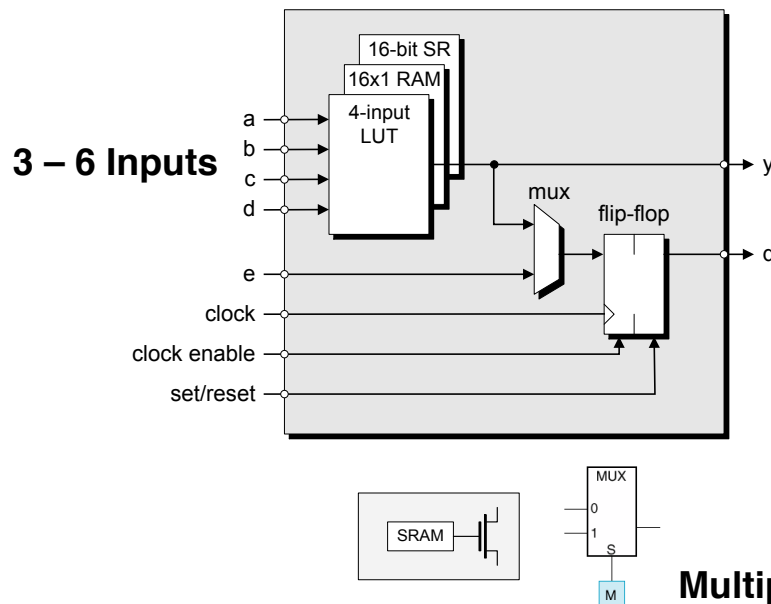
# Simplified CLB Structure





- LUT contains Memory Cells to implement small logic functions
- Each cell holds '0' or '1' .
- Programmed with outputs of Truth Table
- Inputs select content of one of the cells as output

## 3 Inputs LUT -> 8 Memory Cells

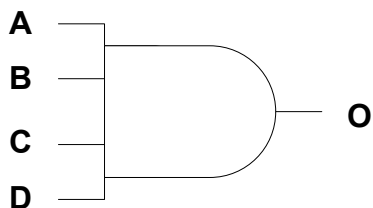


**Static Random Access Memory  
SRAM cells**

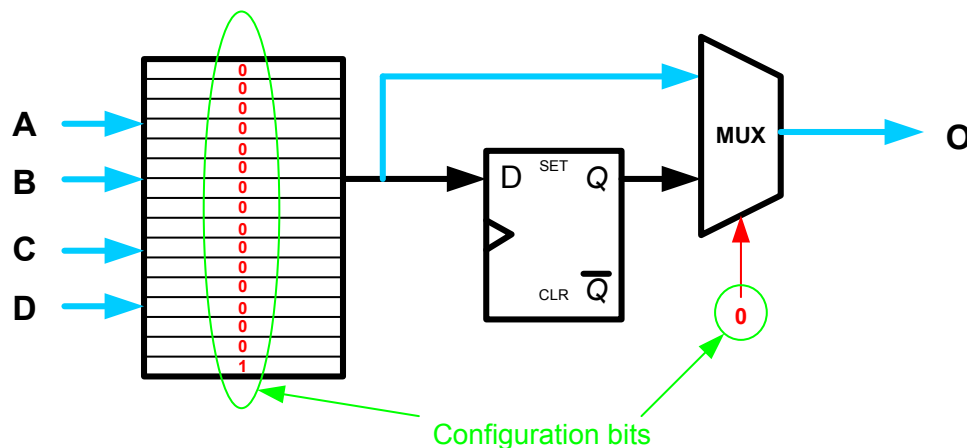
**Multiplexer MUX**

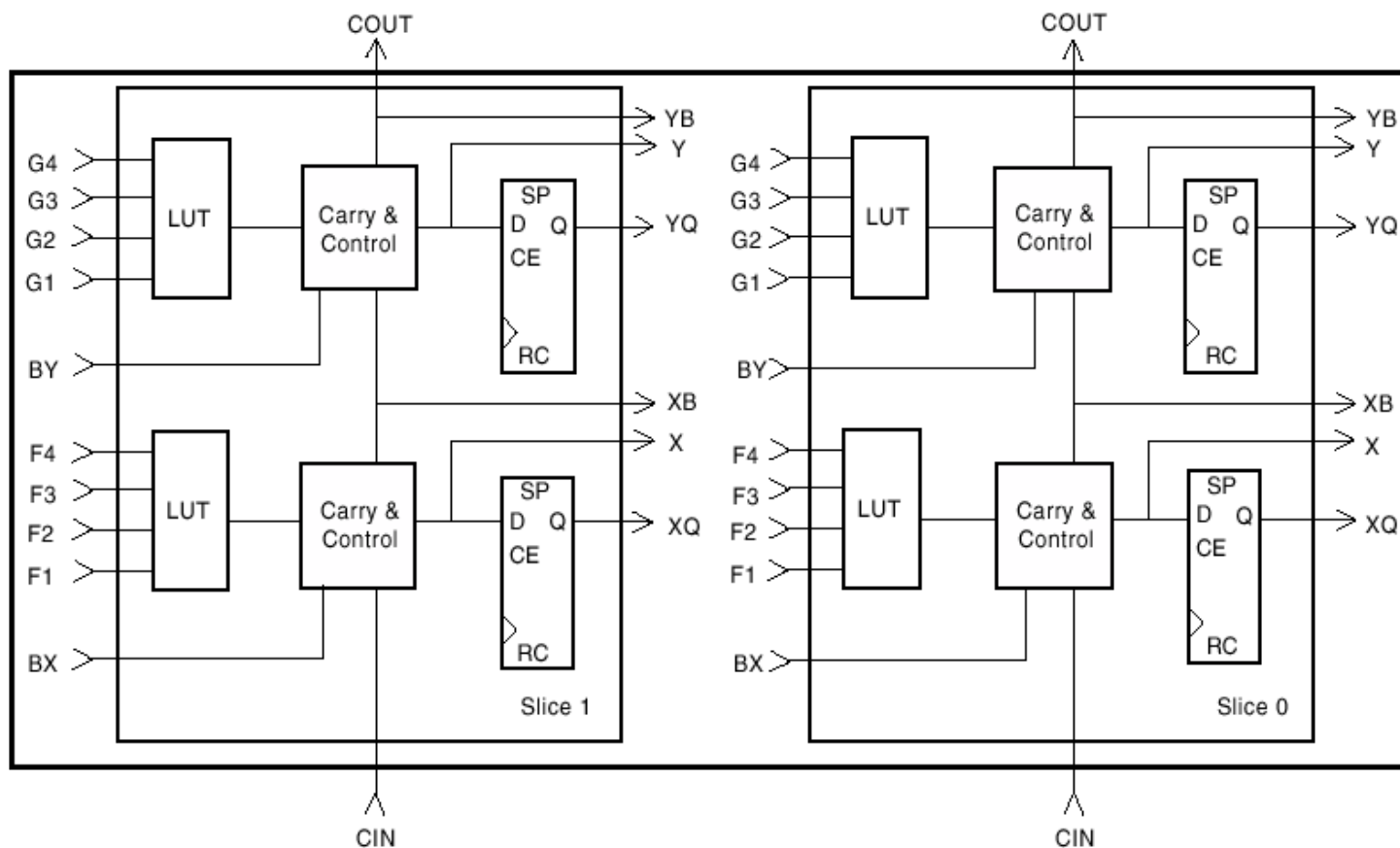


# Example: 4-input AND gate



A	B	C	D	O
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



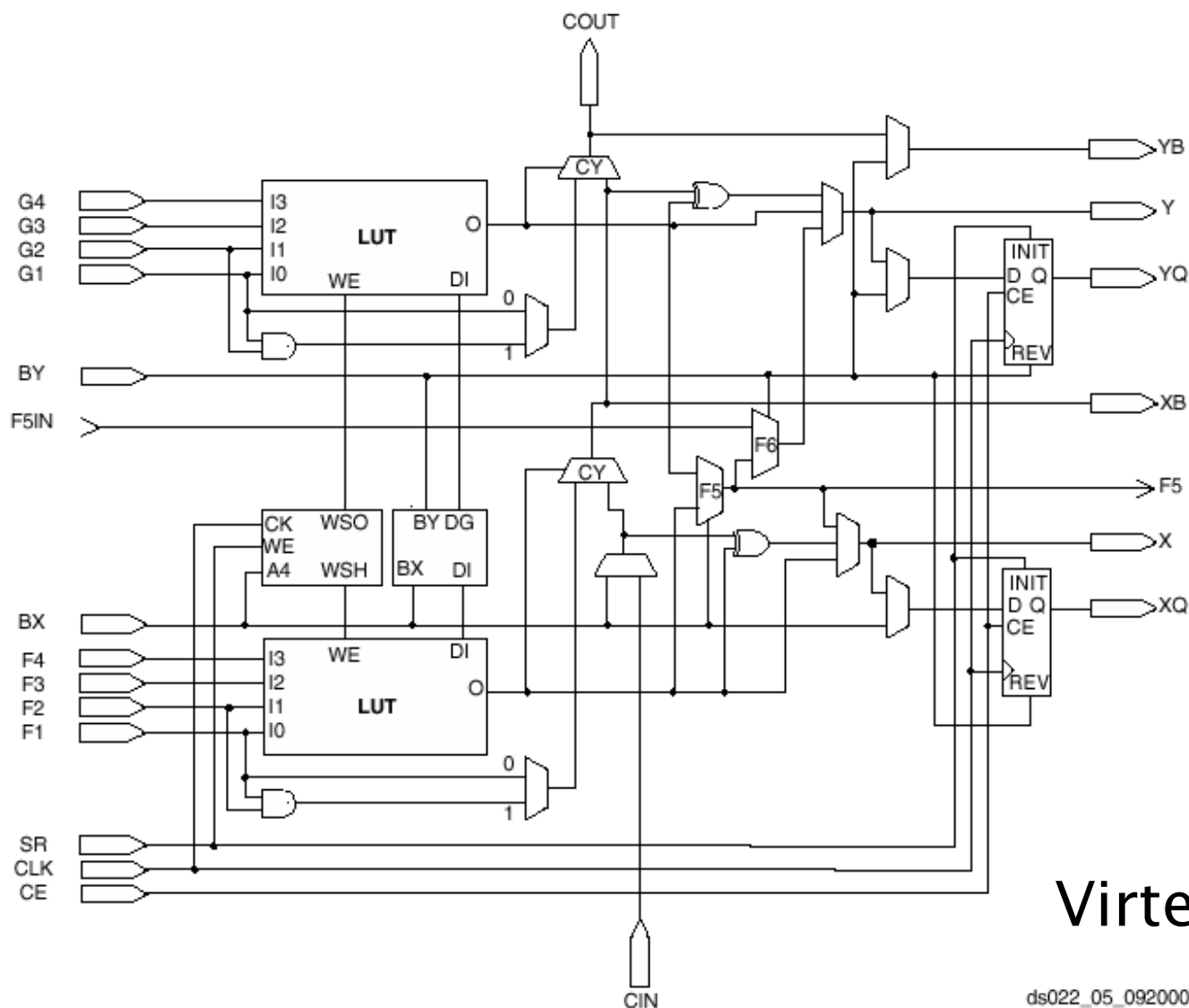


## 2-Slice Virtex-E CLB

ds022\_04\_121799



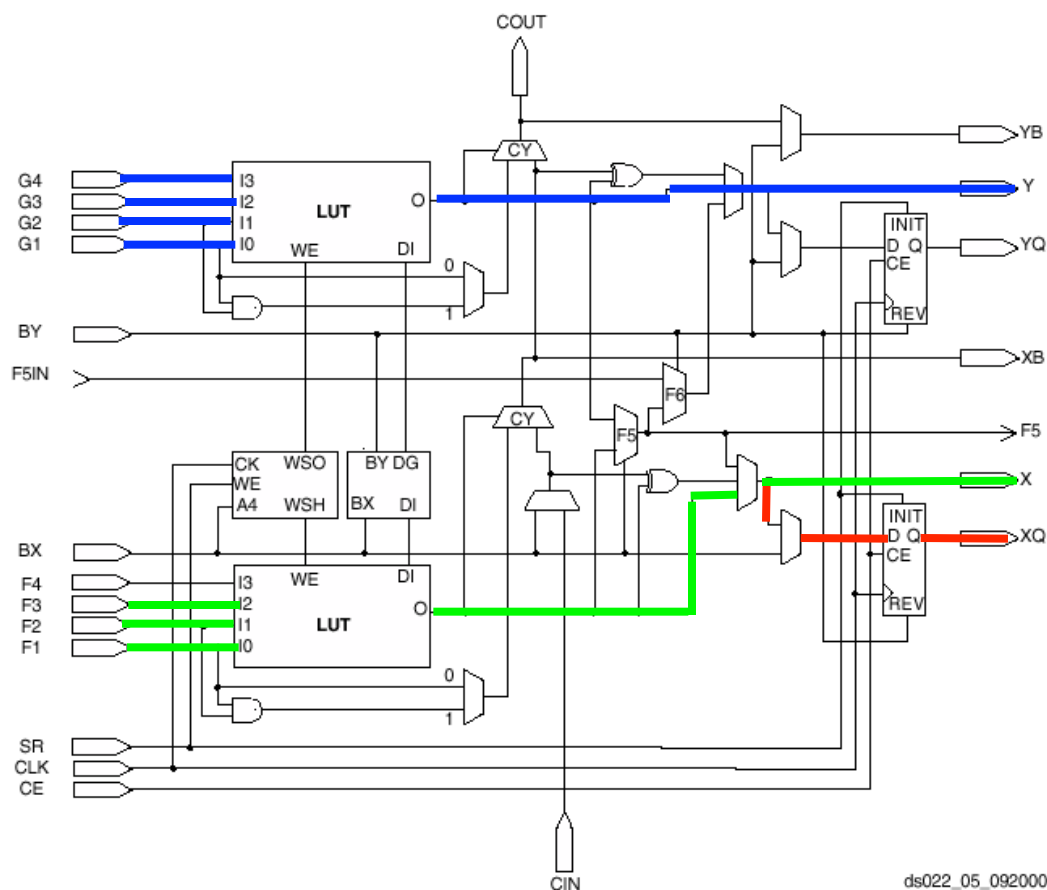
# Details of One Virtex Slice



ds022\_05\_092000



# Implements any Two 4-input Functions

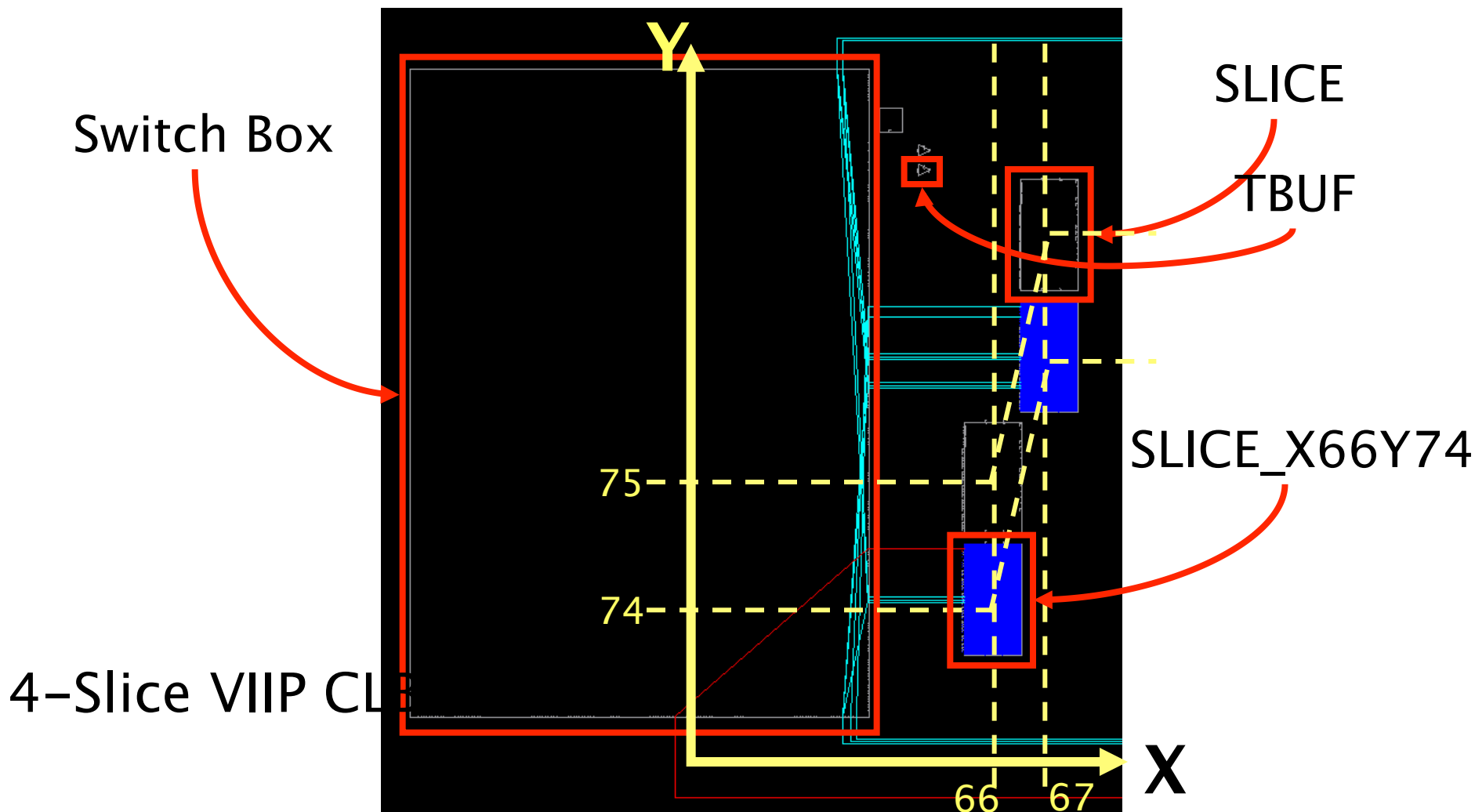


4-input  
function

3-input  
function;  
registered

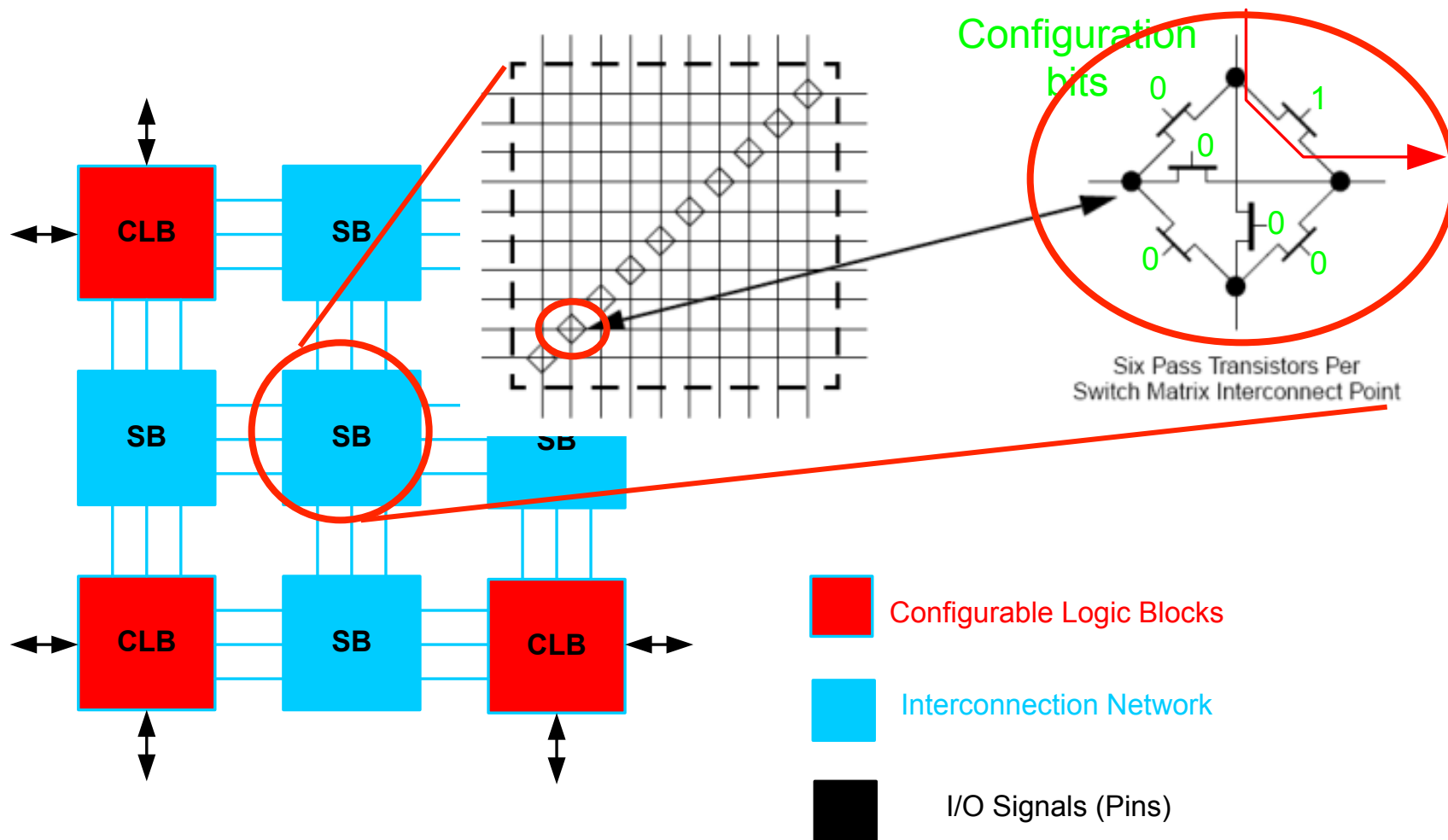
ds022\_05\_092000

Virtex-E Slice





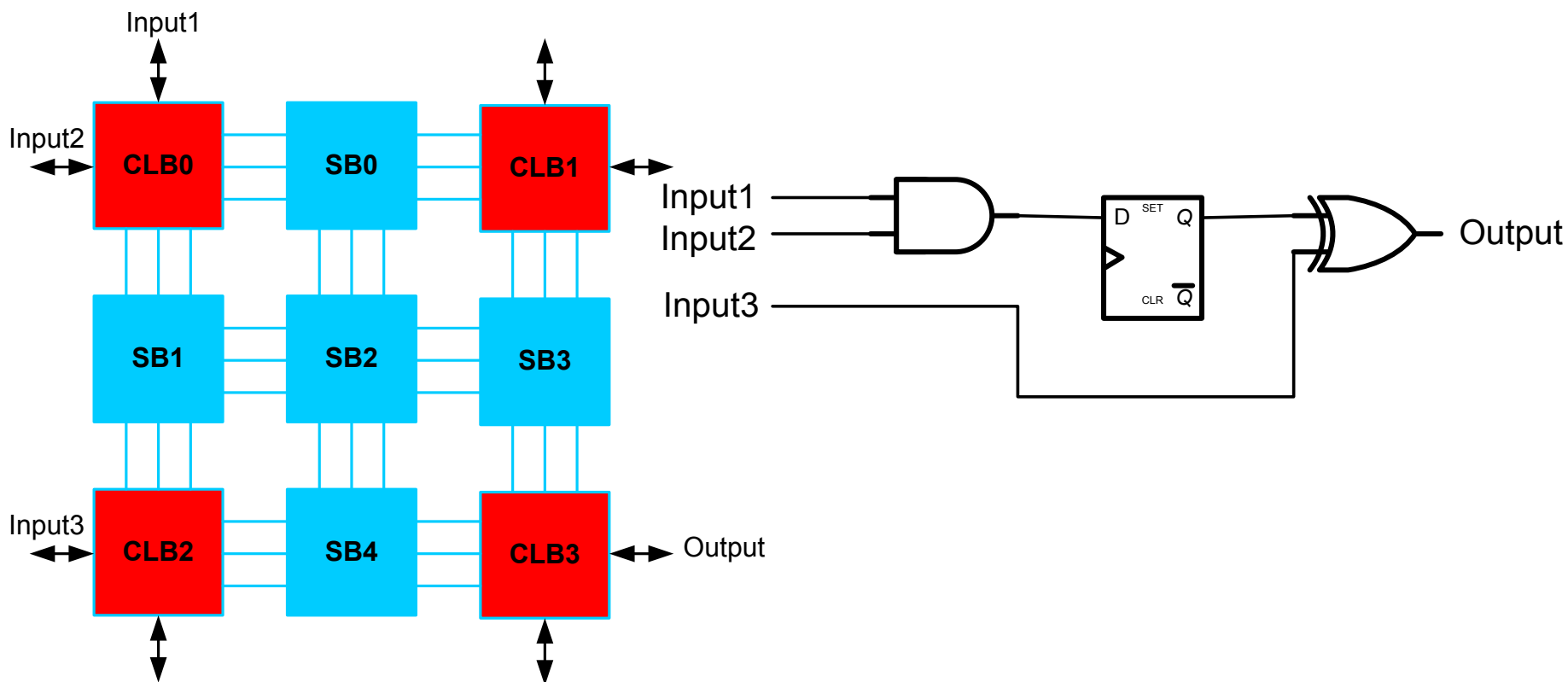
# Interconnection Network





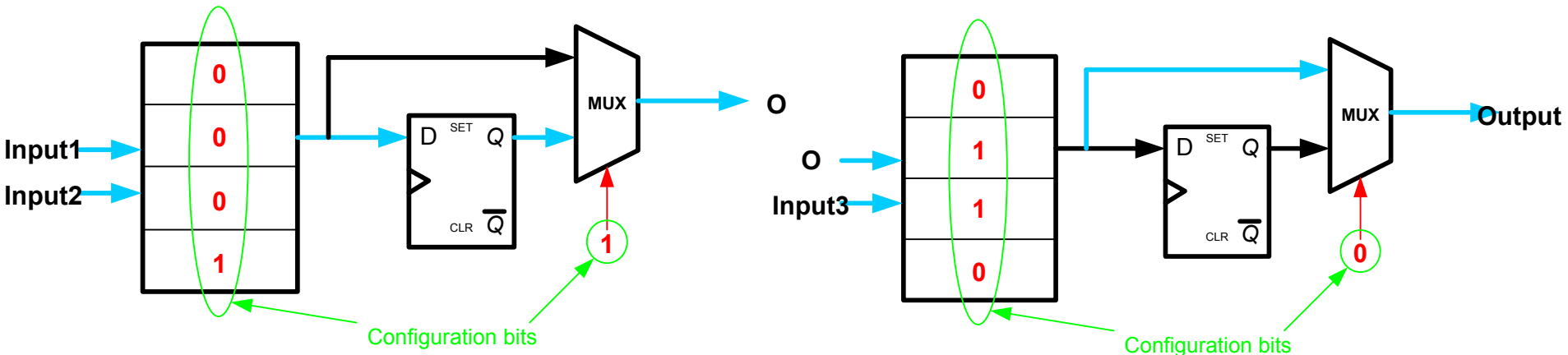
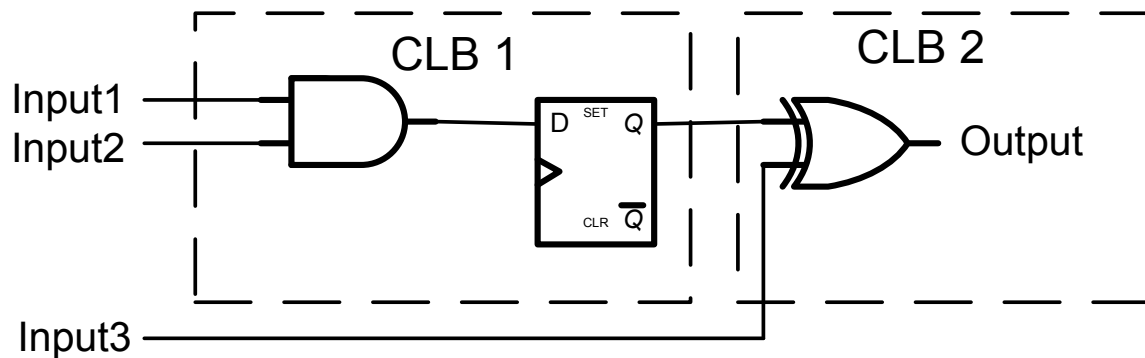
# Example

- Determine the configuration bits for the following circuit implementation in a 2x2 FPGA, with I/O constraints as shown in the following figure. Assume 2-input LUTs in each CLB.

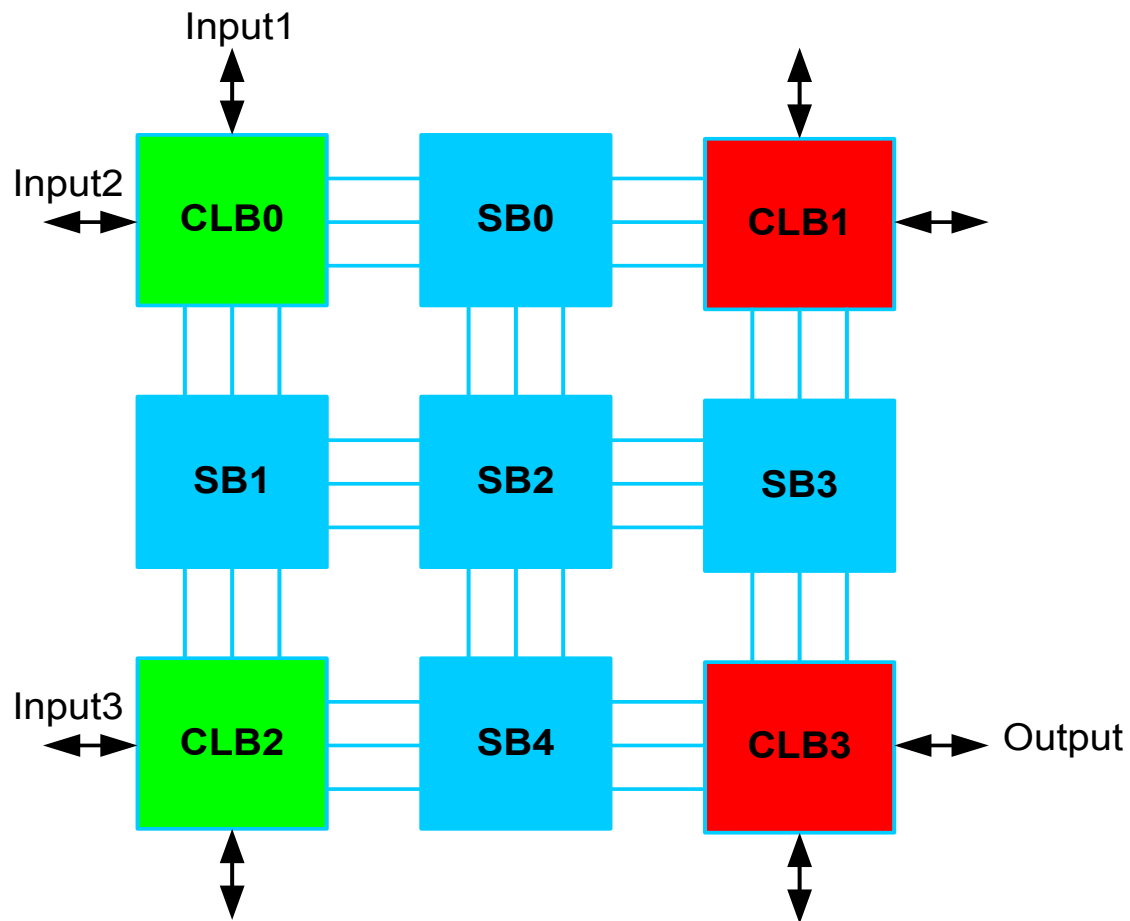




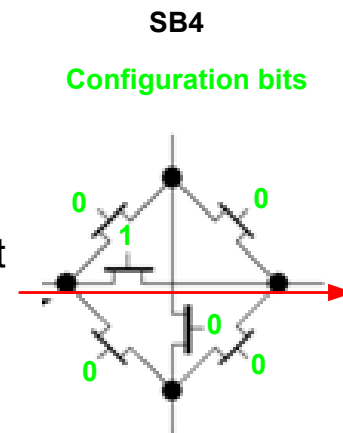
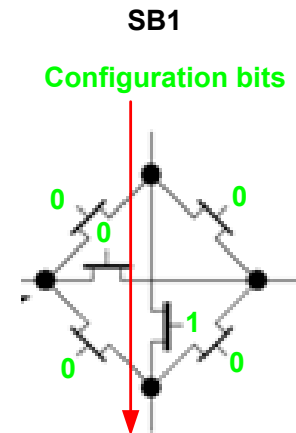
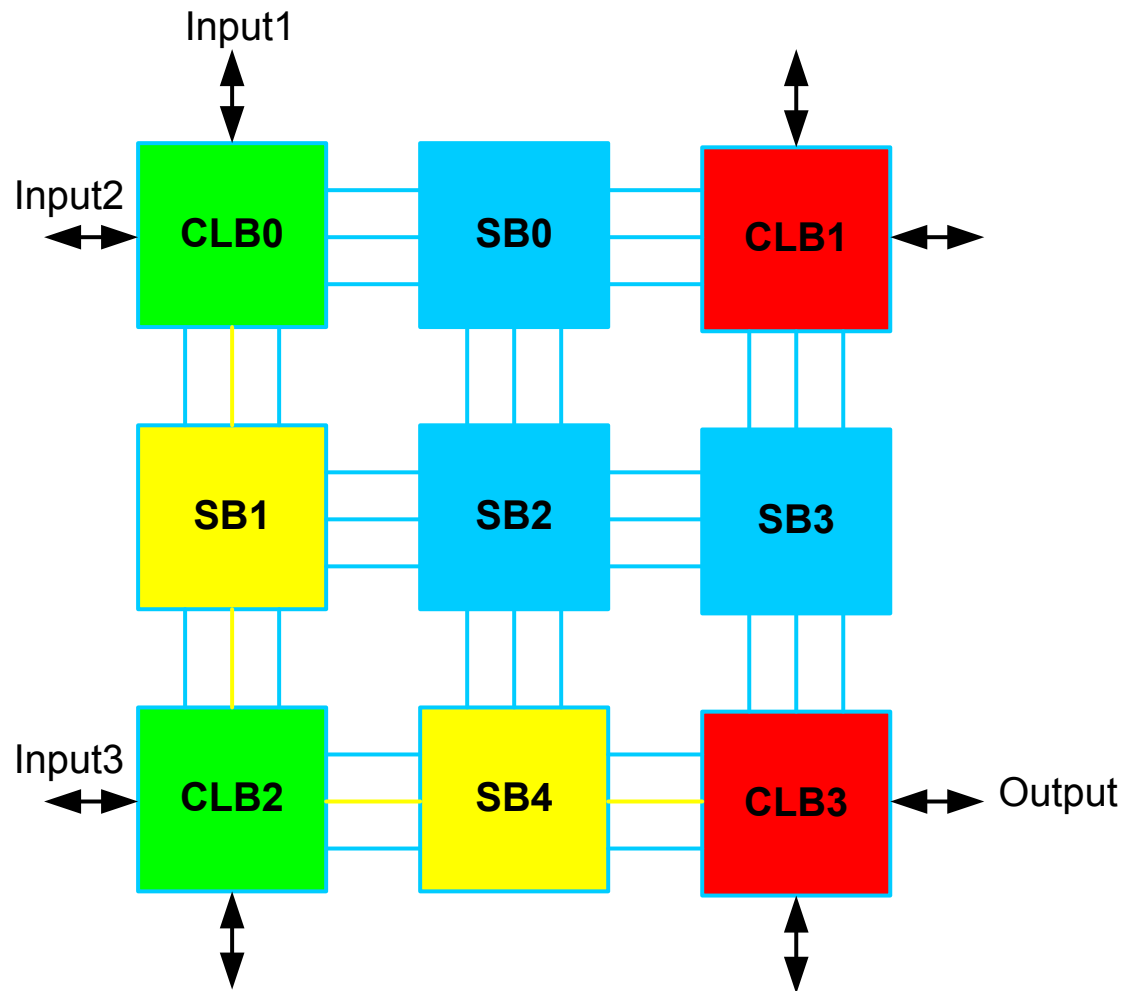
# CLBs configuration



# Placement: Select CLBs



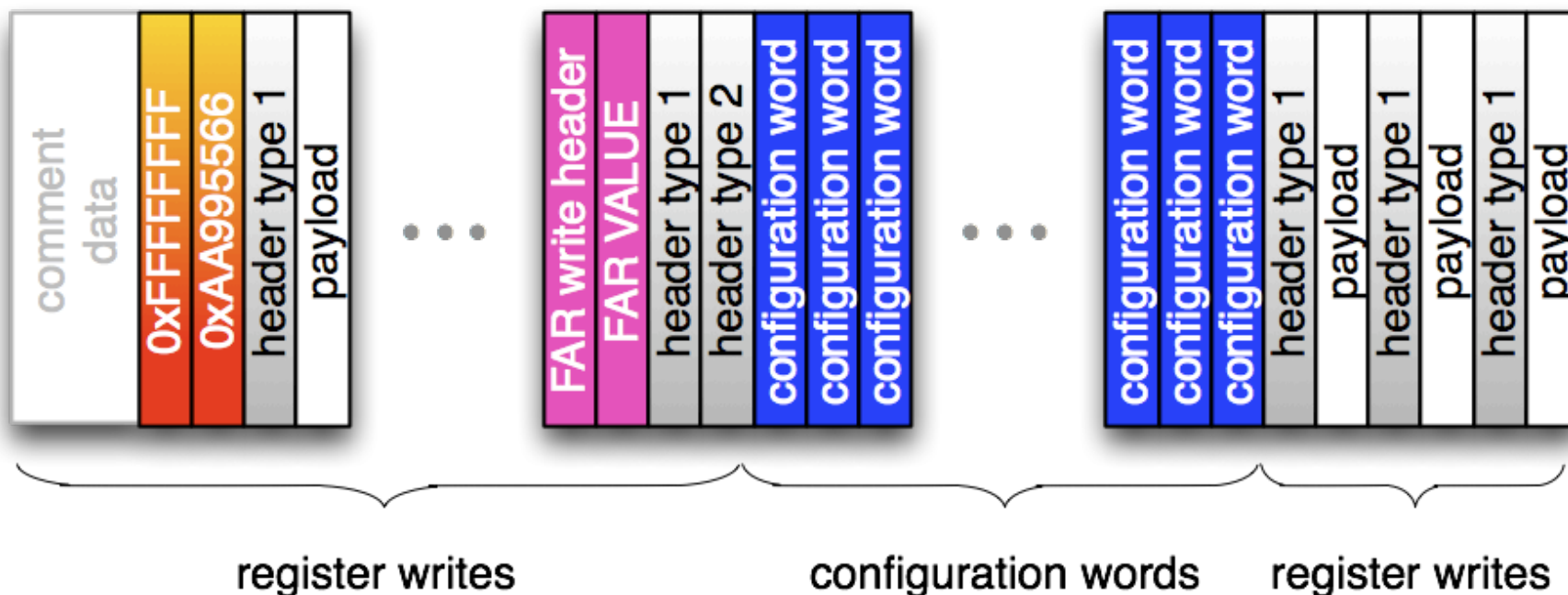
# Routing: Select path



- The configuration bitstream must include ALL CLBs and SBs, even unused ones
- CLB0: 00011
- CLB1: ??????
- CLB2: 01100
- CLB3: XXXXX
- SB0: 000000
- SB1: 000010
- SB2: 000000
- SB3: 000000
- SB4: 000001



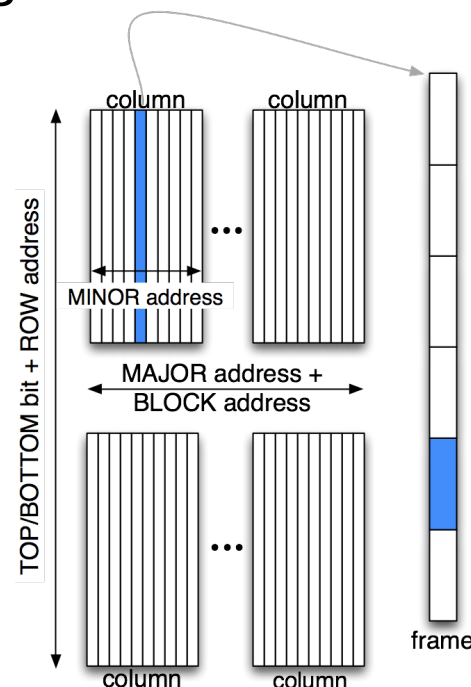
# The configuration bitstream



- Occupation must be determined only on the basis of
  - Number of configuration words
  - Initial Frame Address Register (FAR) value



- Virtex-II Pro
  - Configuration memory is arranged in vertical frames that are one bit wide and stretch from the top edge of the device to the bottom
  - Frames are the smallest addressable segments of the VIIP configuration memory space
    - all operations must act on whole configuration frames.
- Virtex-4
  - ▶ Configuration memory is arranged in frames that are tiled about the device
  - ▶ Frames are the smallest addressable segments of the V4 configuration memory space
    - all operations must therefore act upon whole configuration frames





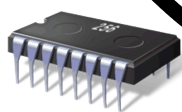
- **Object Code**: the executable active physical (either HW or SW) implementation of a given functionality
- **Core**: a specific representation of a functionality. It is possible, for example, to have a core described in VHDL, in C or in an intermediate representation (e.g. a DFG)
- **IP-Core**: a core described using a HD Language combined with its communication infrastructure (i.e. the bus interface)
- **Reconfigurable Functional Unit**: an IP-Core that can be plugged and/or unplugged at runtime in an already working architecture
- **Reconfigurable Region**: a portion of the device area used to implement a reconfigurable core



## bitstream files



ICAP



SelectMAP

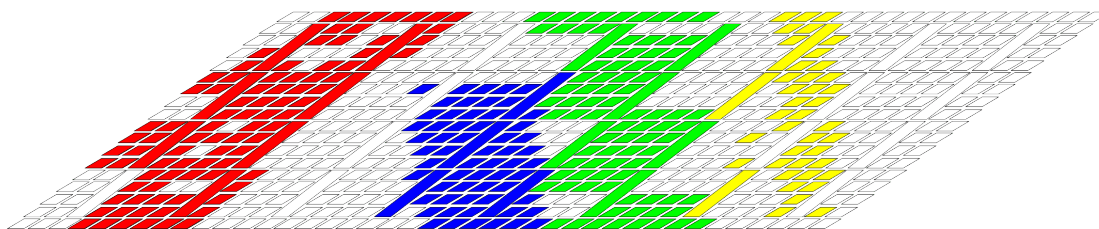


Reconfiguration  
Controllers

ObjectCode<sub>i</sub>



FPGA



column type	GCLK	IOB	IOI	CLB	CLB	CLB	CLB	...	CLB	CLB	CLB	...	BRAM	BRAM	BRAMINT	BRAMINT
Block Address	0	0	0	0	0	0	0	...	0	0	0	...	1	1	2	2
Major Address	0	1	2	3	4	5	6	...	n + 2	n + 3	n + 4	...	0	m	0	m

Configuration Memory





- A bird's eye view on the Reconfigurable Computing
- FPGA
- The roadmap
  - The 90% - 10% Rule
  - Programmable System on a Chip
  - Multi-FPGA
  - Complex heterogeneous adaptive systems
- What's missing

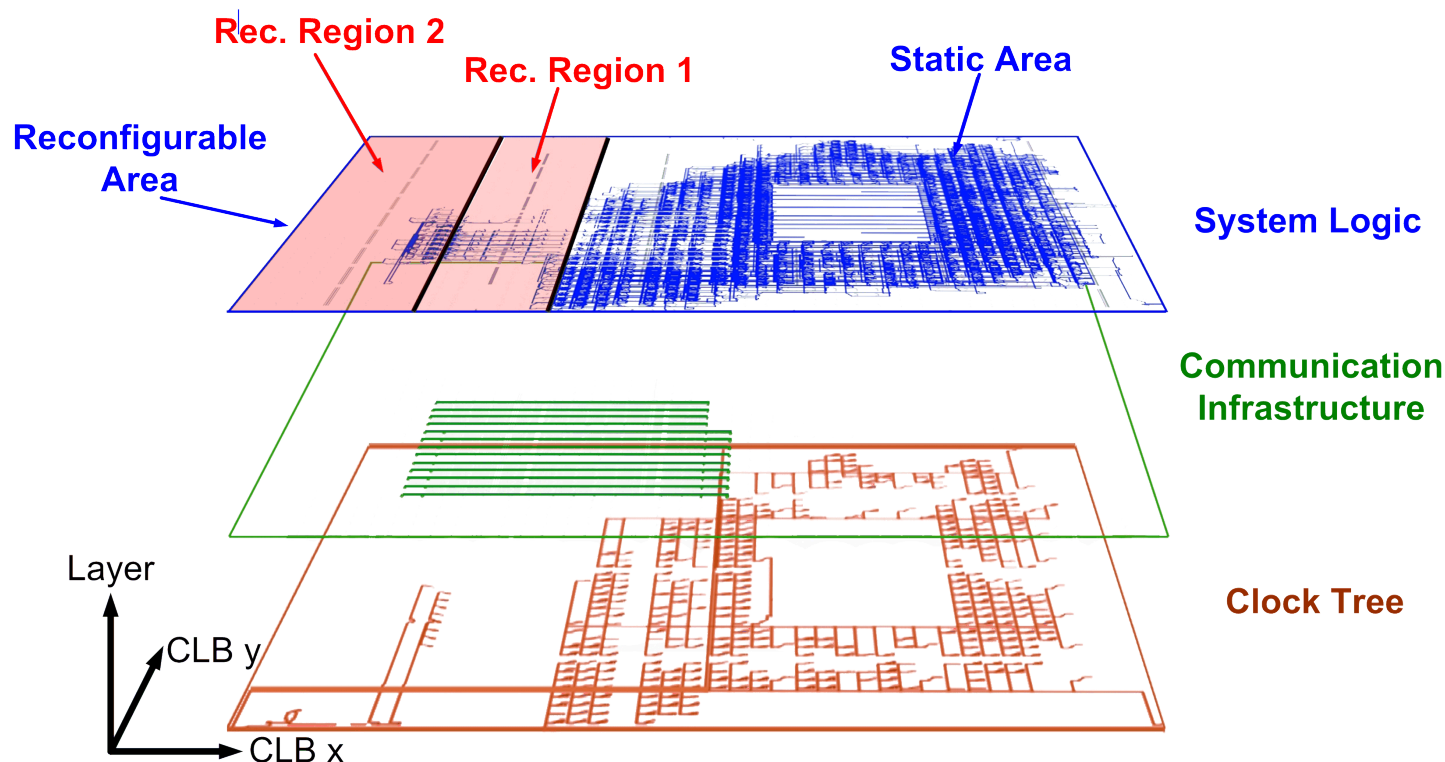


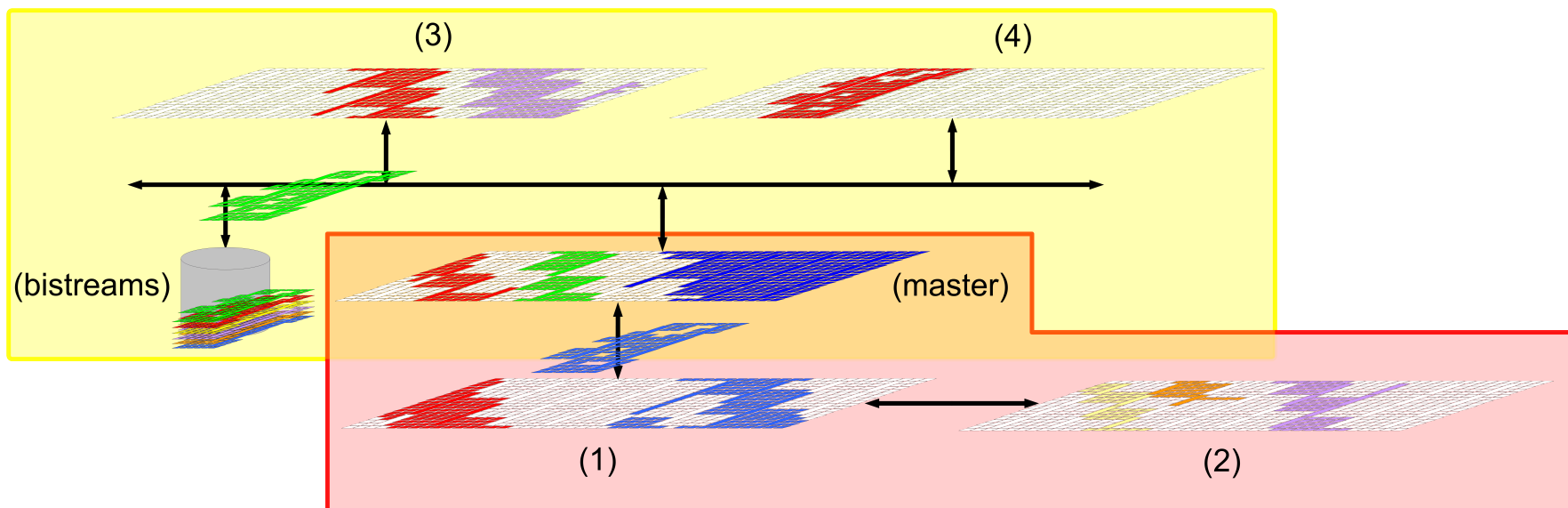
# The 90% - 10% Rule

- 90% of the execution is spent in 10% of the code
  - Inner loops in algorithms
  - Computational intense code
- 10% of the execution is spent in 90% of the code
  - Exceptions
  - User interaction
- The 10% computational intense code has to be executed as hardware on reconfigurable devices
- The 90% exception code is run as executable files on processors



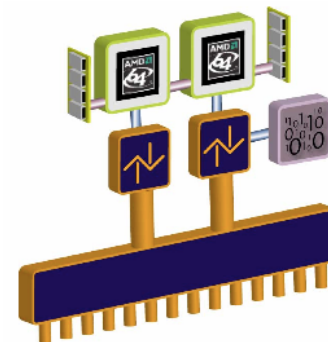
- No longer just a bunch of reconfigurable elements
- DSPs, GPP, reconfigurable elements, etc. etc...







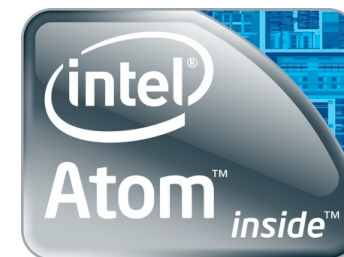
- Due to the complexity in the demand, the system has to be heterogeneous and able to autonomously adapt and evolve
  - FPGAs
  - DSPs
  - GPP (Multi-cores)
- Adaptive systems learn how they can be used to address a particular problem
  - Respond to user goals
  - Build self-performance models
  - Identify what they need to learn
  - Adapt to changing goals, resources, models, operating conditions
  - Gracefully adapt to failures
  - Optimize their own behavior



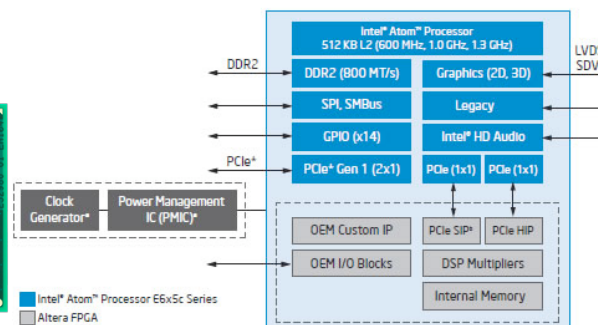


- Heterogeneous *Multicore*
  - An Intel Atom E6XX processor

- # Cores: 1
- # Threads: 2
- L2 Cache: 512 KB



- An Altera Field Programmable Gate Array



\*PMIC and Clock Generator products are available from third parties. An integrated PMIC and Clock Generator (on a single chip) is also available from a third party.  
\*PCIe Soft IP is licensed from third-party vendors.



- Basic Idea
- FPGA
- The roadmap
- What's missing
  - Design methodologies
  - Runtime reconfiguration management
  - Rationale behind DRES







- Dynamic reconfigurable embedded systems are gathering, an increasing interest from both the scientific and the industrial world
  - The need of a comprehensive framework which can guide designers through the whole implementation process is becoming stronger
- There are several techniques to exploit partial reconfiguration, but..
  - Few approaches for frameworks and tools to design dynamically reconfigurable systems
  - They don't take into consideration both the HW and the SW side of the final architecture
  - They are not able to support different devices
  - They cannot be used to design systems with different architectural solutions





- To identify from an high-level or RT-Level specification how to define reconfigurable functional units
- To define which IP-Core has to become a reconfigurable functional unit
- To define interconnections among reconfigurable functional units
- To manage at runtime the swap of reconfigurable functional units
- To perform the actual swap of reconfigurable functional units and the FPGA reconfiguration

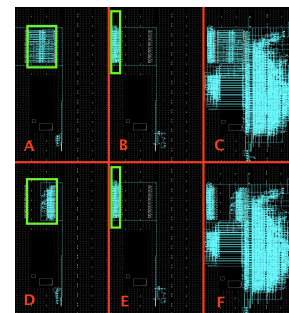
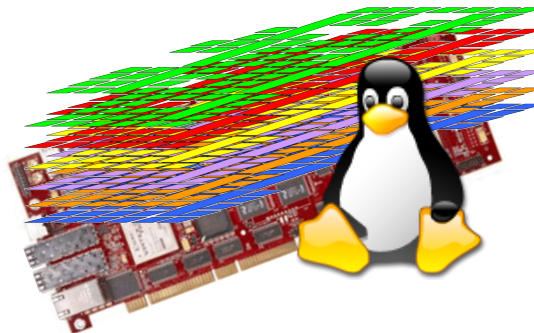
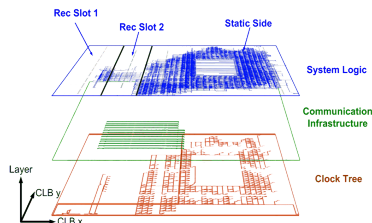
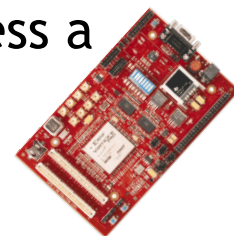


- The flexibility of a reconfigurable system comes from the possibility of downloading different hardware configurations onto the chip at different times
  - Reconfiguration time overhead 
  - Memory usage to store the bitstreams 
- There are different models of reconfiguration classified according to the following scheme (5W):
  - *who* manages the reconfiguration 
  - *where* the reconfiguration controller is located
  - *when* the configurations are generated
  - *which* is the granularity of the reconfiguration
  - *what* dimension (1D vs 2D) the reconfiguration operates 





- Exploit dynamic reconfigurability for different target reconfigurable architectures.
- Definition of a complete design flow (*Brain to Bit*) to process a specification to make it suitable for reconfigurable implementation
- Definition and implementation of a new generation of self reconfigurable architectures based on Linux
- Increase the reconfiguration performances via novel techniques, i.e. runtime reconfigurable cores relocation, reconfigurable cores identification, reconfigurable cores reuse





- Definition of a methodology for the design of dynamically reconfigurable FPGA applications:
  - to define a design flow that exploits, wherever possible, the commercial tools available
  - to provide guidelines on the identification and definition of reconfigurable cores
  - to define the modules schedule trying to reduce the reconfiguration time
  - to increase the system performances and to reduce the number of bitstreams to store in memory for reconfiguration
    - using the runtime bistreams relocation technique
  - to manage the runtime swap of reconfigurable cores



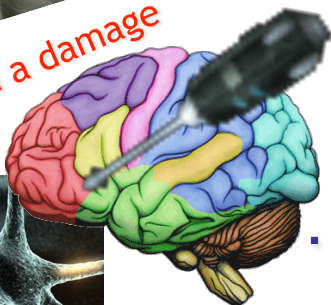


# From Classical Systems...

Structural  
modification



Recovery from a damage



Behavioral evolution



## ... to Self-Aware and Adaptive Ones

Self aware systems learn how they can be used to address a particular problem and adapt their structure towards it

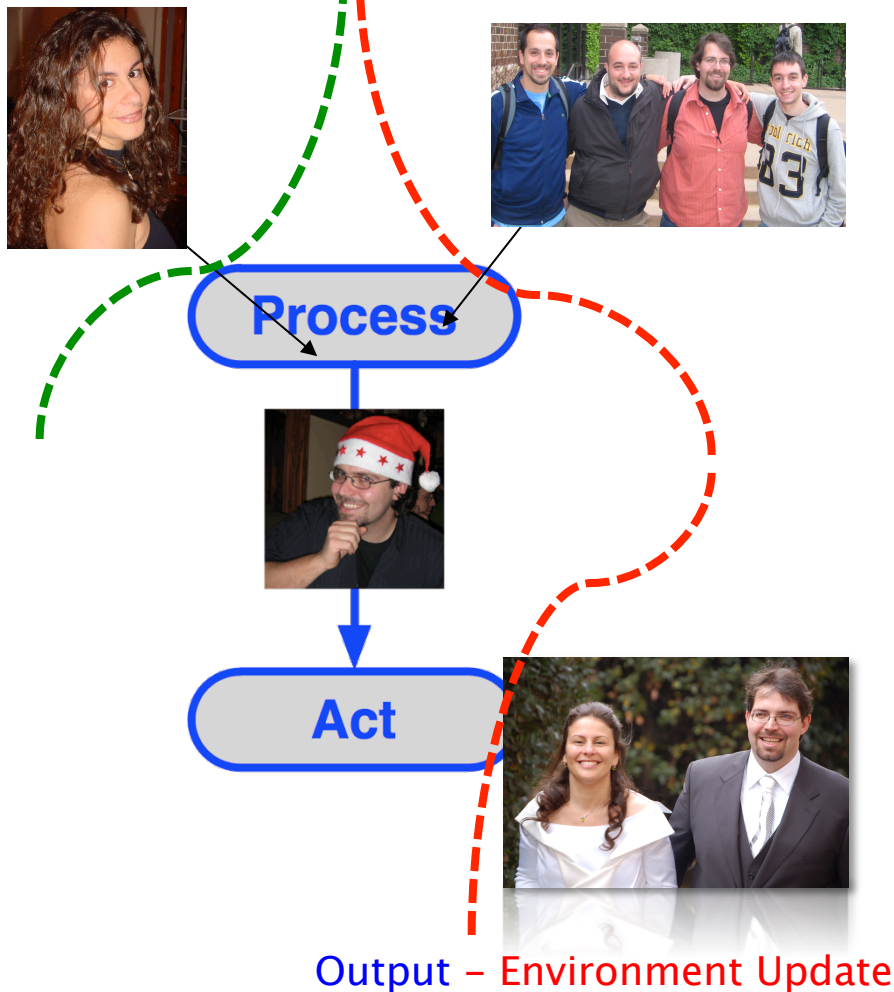
- ❑ Respond to user goals
- ❑ Build self-performance models
- ❑ Adapt to changing goals, resources, models, operating conditions
- ❑ Identify what they need to learn
- ❑ Gracefully adapt to failures
- ❑ Optimize their own behavior





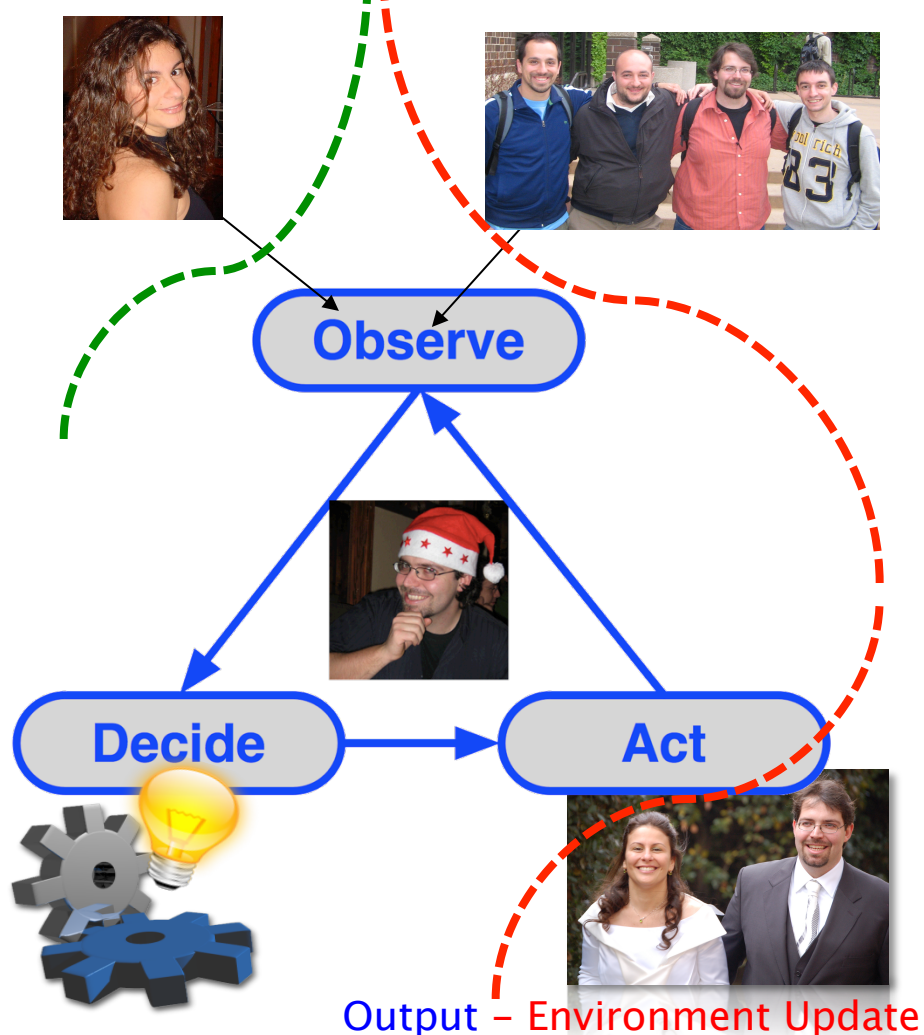
## Online Static Solution

Data Environment



## Adaptive Solution

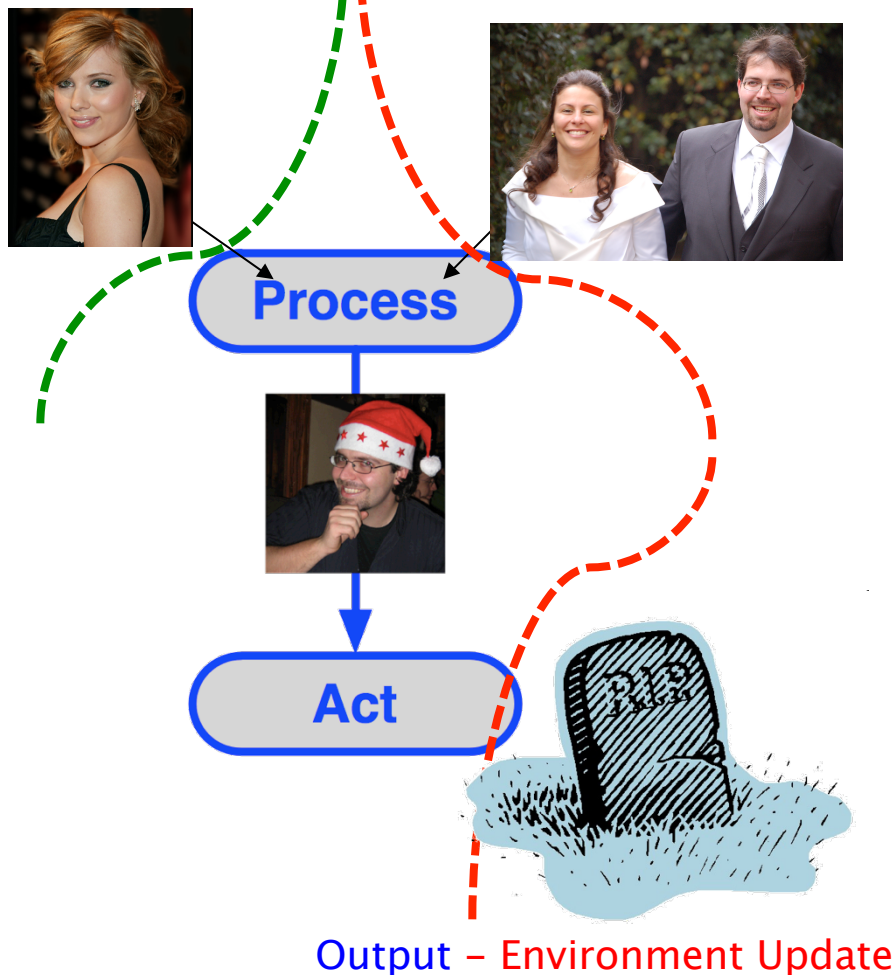
Data Environment





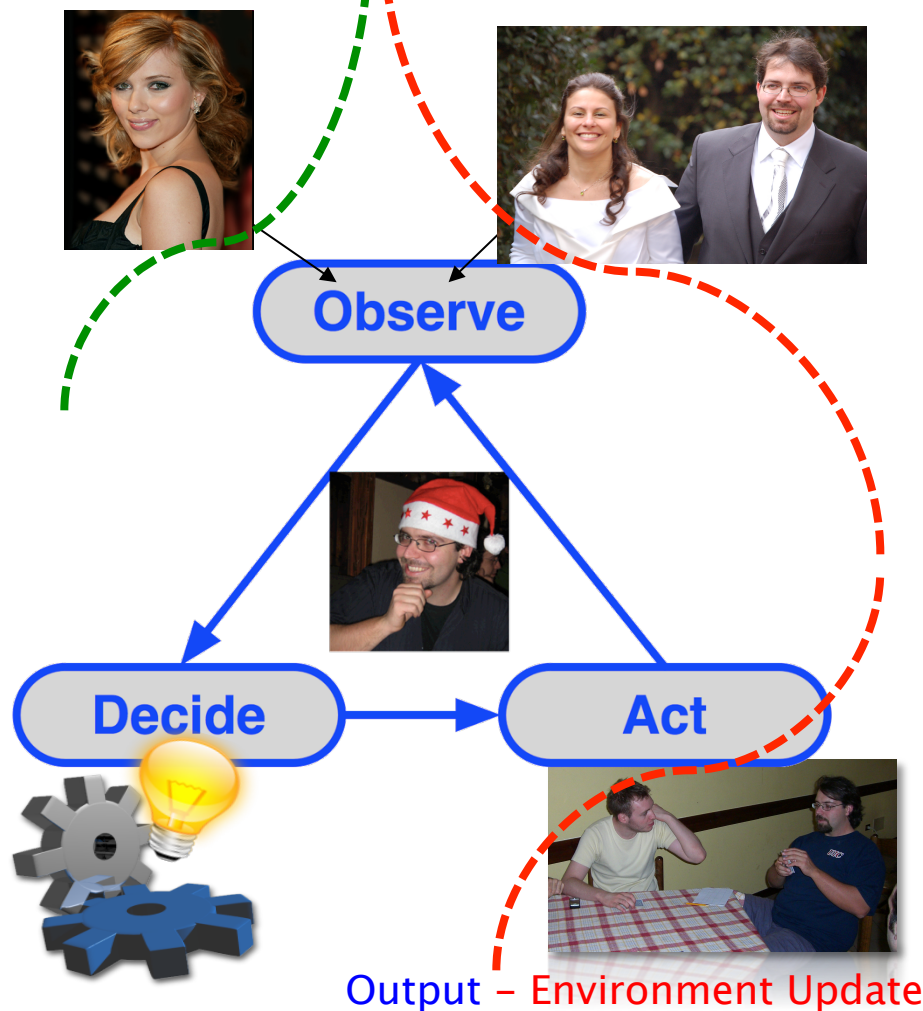
## Online Static Solution

Data Environment



## Adaptive Solution

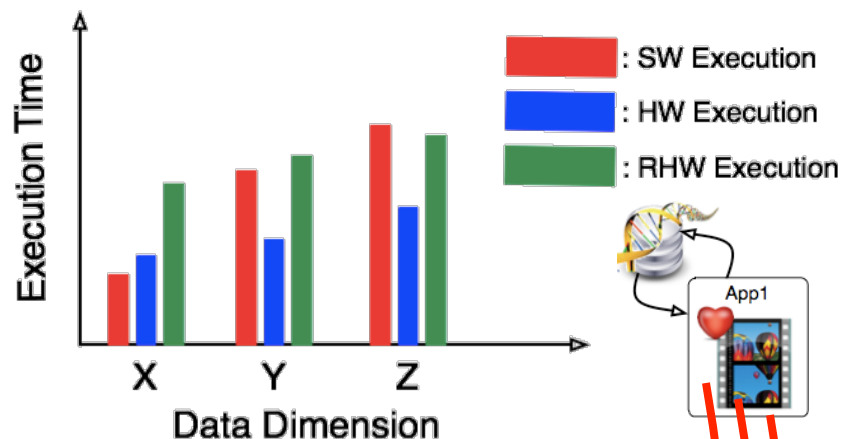
Data Environment



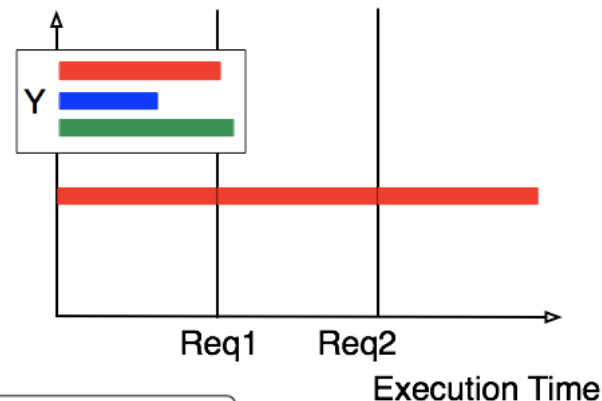




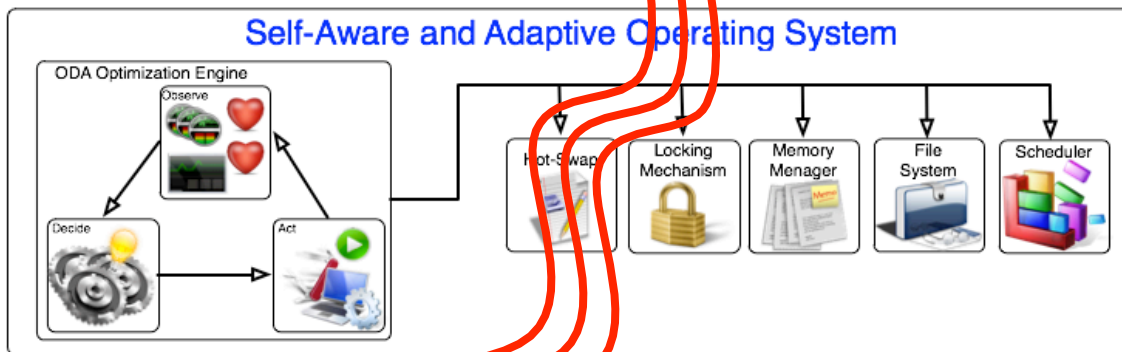
# Reconfiguration: Online Static



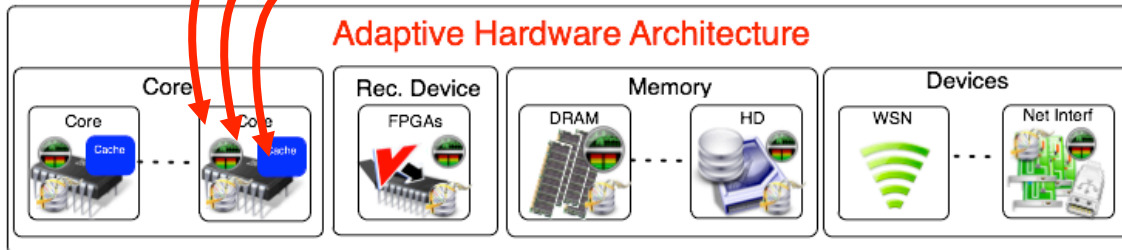
3 requests on Y data (NO HW configured)



## Self-Aware and Adaptive Operating System



## Adaptive Hardware Architecture



Process



Act



: Adaptive Libraries



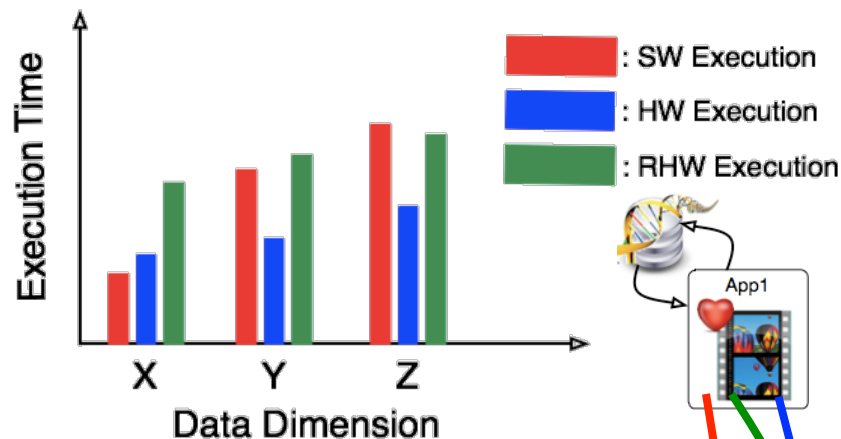
: Performance



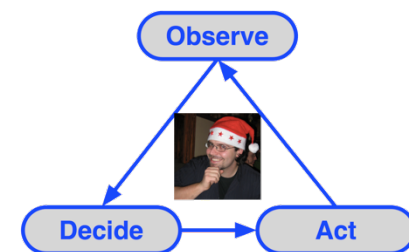
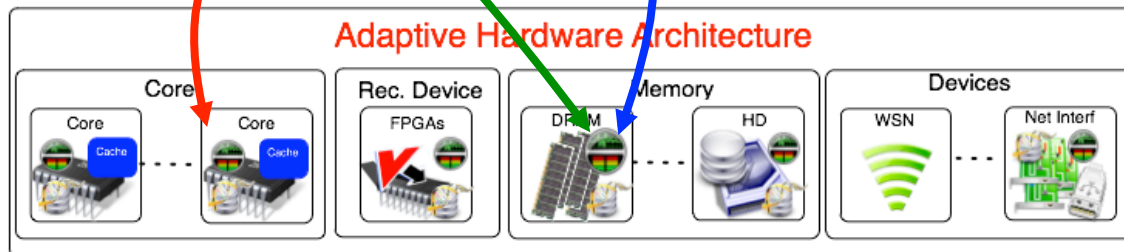
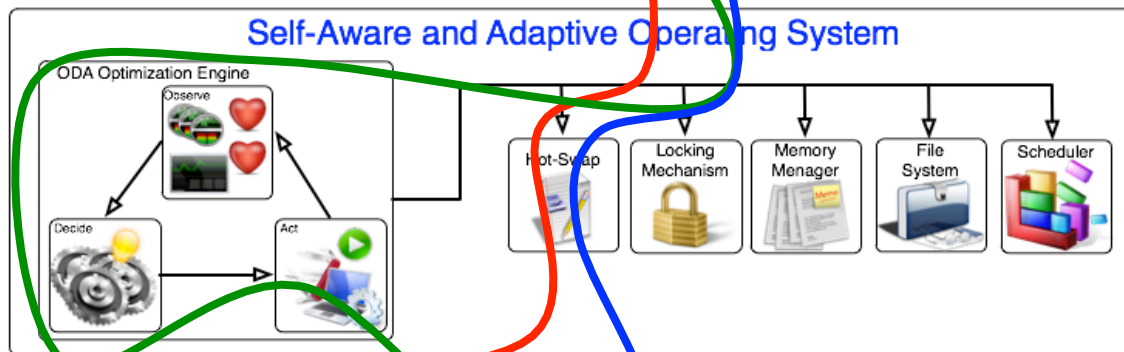
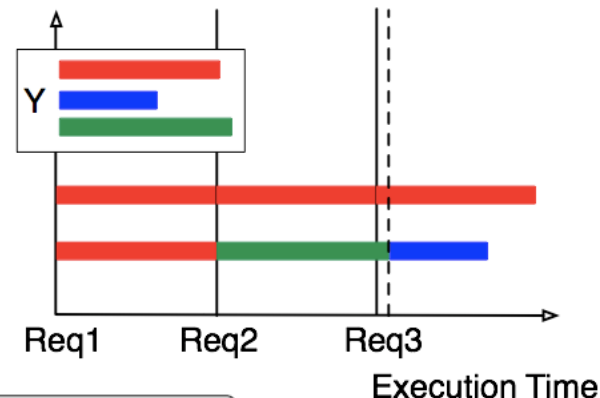
: Monitors



# Reconfiguration: Self-Aware



3 requests on Y data (NO HW configured)



: Adaptive Libraries



: Performance



: Monitors



# Questions

**I have no idea what you're talking about...**



**...so here's a bunny with a pancake on its head.**