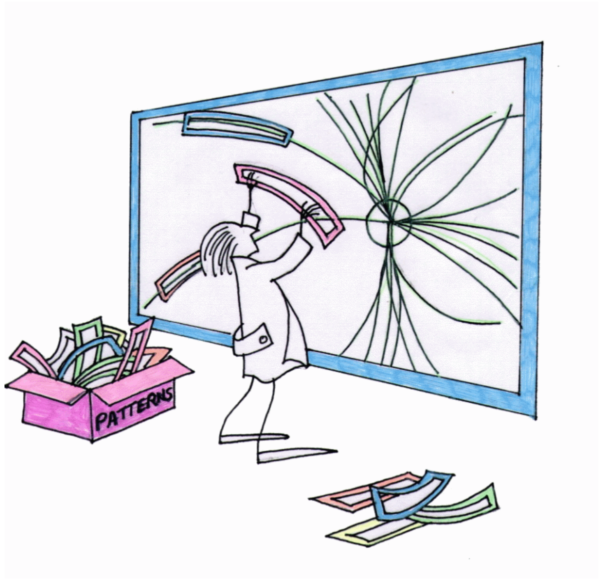
MARIE CURIE IAPP: FAST TRACKER FOR HADRON COLLIDER EXPERIMENTS

1ST SUMMER SCHOOL: VHDL BOOTCAMP

PISA, JULY 2013

Introduction to FPGAs

Calliope-Louisa Sotiropoulou
PhD Candidate/Researcher
Aristotle University of Thessaloniki



AUTH e-LAB
Aristotle University of Thessaloniki-Electronics Laboratory

IN SUPREMÆ DIGNITATIS 1343
UNIVERSITÀ DI PISA

VHDL Bootcamp - DISCLAIMERS

- The school targets VHDL beginners with an experience in computer programming
- The purpose of this course is to show you what a powerful tool VHDL and FPGAs are together with the XILINX tools
- We will be able to use VHDL but it is impossible to learn VHDL in 4 days…

- As with everything in life…

Practice makes perfect!

Introduction to FPGAs

- What is an FPGA
- Why we use FPGAs

Introduction – Embedded Systems

- An **embedded system** is a computer system designed to do one or a few dedicated and/or specific functions often with real-time computing constraints. It is **embedded** as part of a complete device often including hardware and mechanical parts.

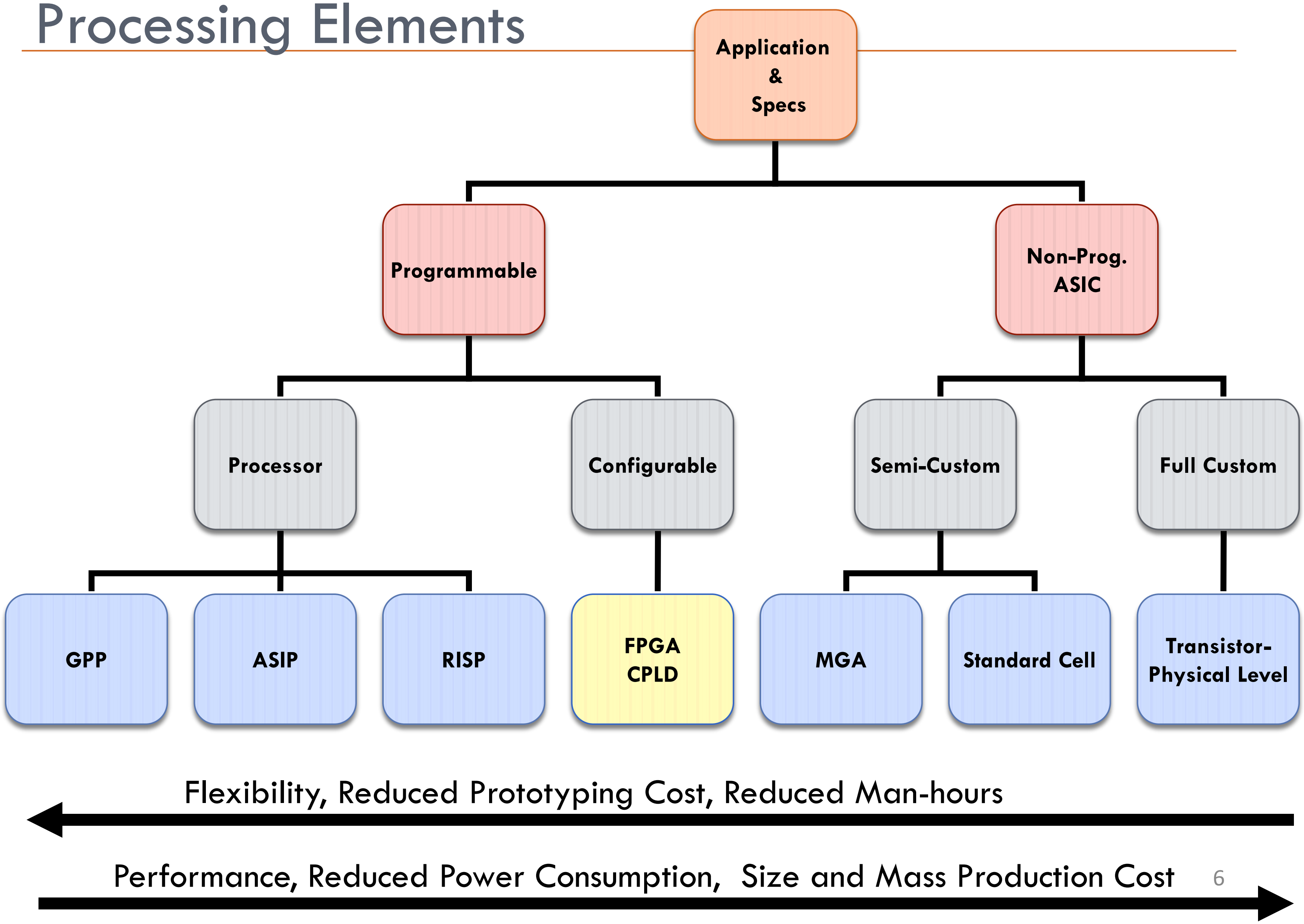
- Embedded systems are application specific
  - Optimize to reduce size
  - Optimize to reduce cost
  - Optimize to increase performance
  - Optimize to increase reliability

Introduction – FPGA Technology

- **Field Programmable Gate Arrays (FPGAs)** are programmable semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects.

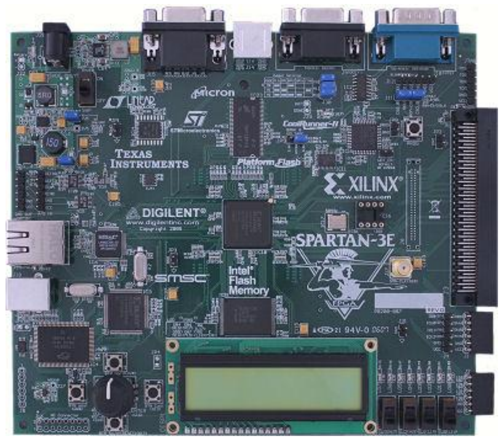
- As opposed to **Application Specific Integrated Circuits (ASICs)** where the device is custom built for the particular design, FPGAs can be programmed to the desired application or functionality requirements.

Introduction – Processing Elements


Application & Specs
Programmable
Non-Prog. ASIC
Processor
Configurable
Semi-Custom
Full Custom
GPP
ASIP
RISP
FPGA CPLD
MGA
Standard Cell
Transistor-Physical Level
Flexibility, Reduced Prototyping Cost, Reduced Man-hours
Performance, Reduced Power Consumption, Size and Mass Production Cost

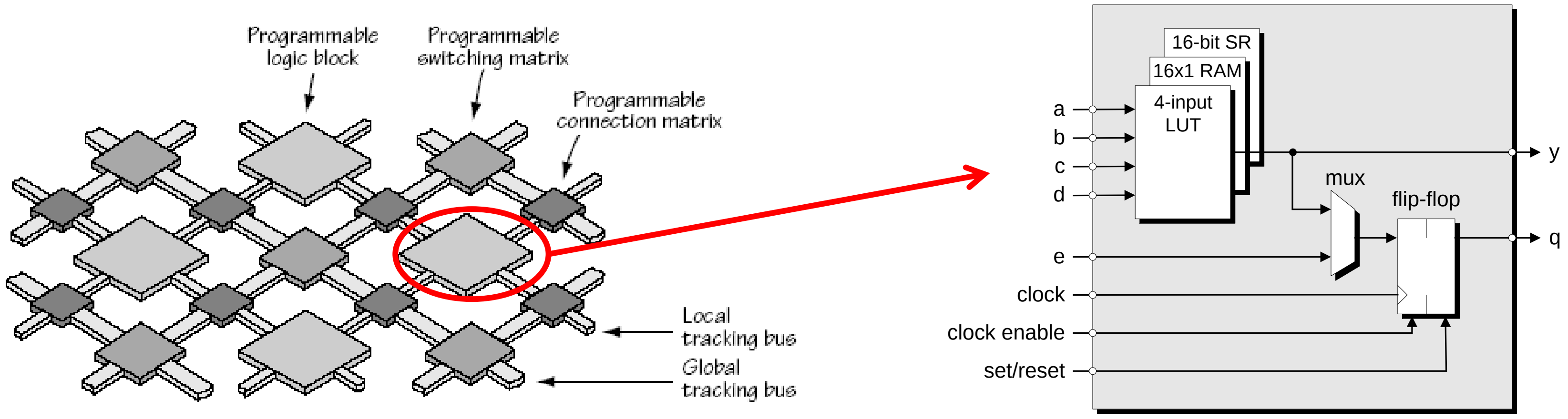
Introduction – FPGA Technology

- Modern FPGA devices
  - Performance
  - Area
  - Application domain specific devices
  - Reduction of development costs
  - Time-to-market



Programmable logic blocks

- The "heart" of every FPGA
  - Logic functions implemented in look-up tables (LUTs)
  - Clocked storage elements (flip-flops)
  - N-to-1 Multiplexers



Programmable logic blocks - 2

- Flexible and versatile
  - Combinational logic (adders, subtractors, shifters,etc.)
  - Sequential logic (registers, data pipelines, ROMs, RAMs, etc.)
  - Combine logic blocks to form larger logic functions
- Architecture varies with vendor
  - Base functionality is essentially the same
  - HDL description of a circuit abstracts implementation
- In most cases, tools map code to hardware efficiently
  - However, knowledge of the underline vendor architecture is essential for **optimal circuit implementation !**

Vendor specific PLBs

- Each FPGA vendor specifies its own PLB architecture

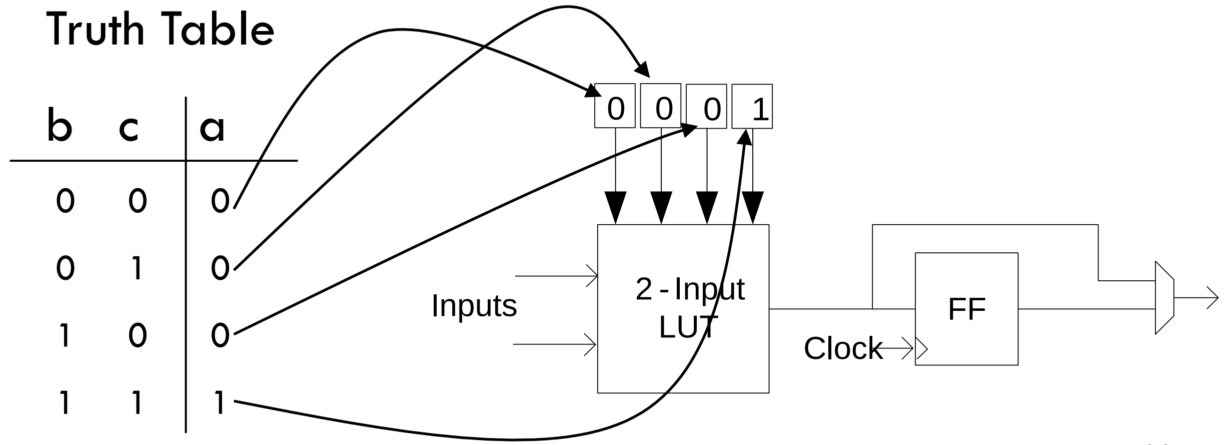
Vendor	Programmable logic block hierarchy	
	1st Level	2nd Level
Xilinx	Configurable Logic Block (CLB)	Slice
Altera	Logic Array Block (LAB)	Adaptive Logic Module (ALM)
Lattice	Programmable Function Unit (PFU)	Slice
MicroSemi (Actel)	VersaTile	-

Logic Implementation on an FPGA

VHDL Code

```
{….
a<=b and c
….}
```

Truth Table

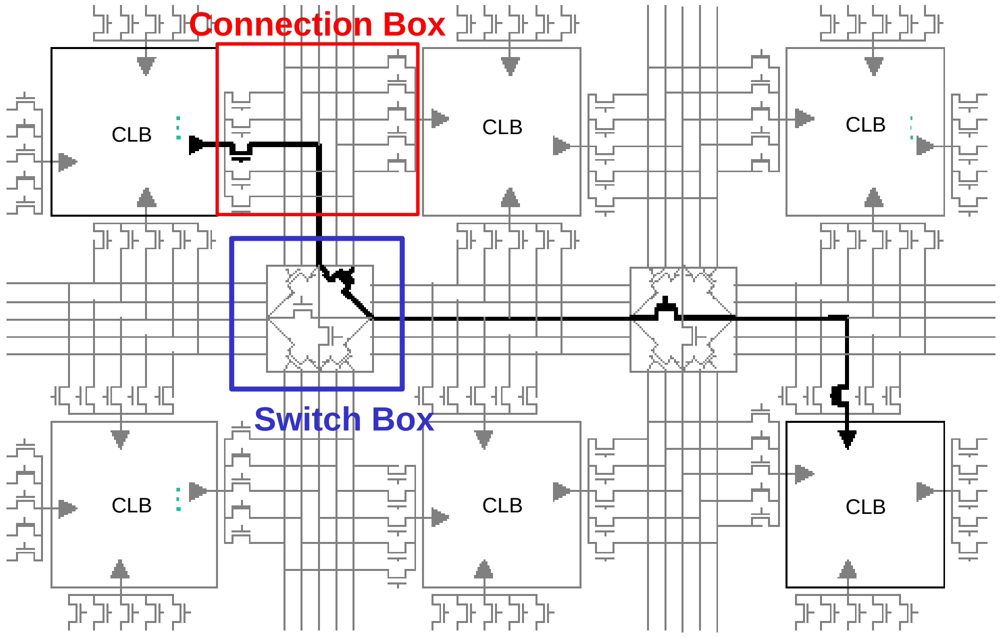


b	c	a
0	0	0
0	1	0
1	0	0
1	1	1

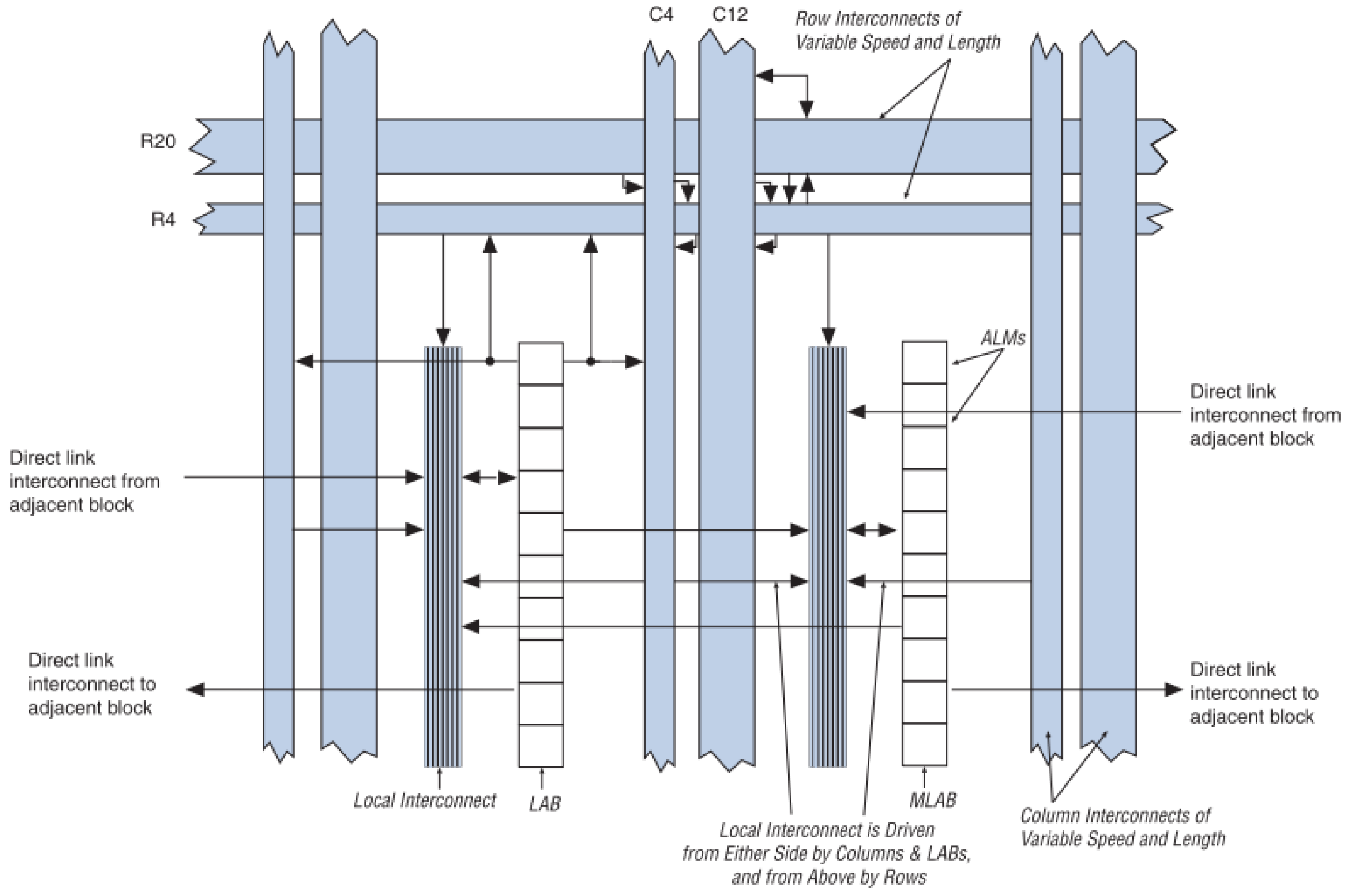
- The truth table of a simple operator is loaded to a LUT
- The two combined inputs are used as address
- The output is stored on a flip-flop for synchronization

Routing on an FPGA

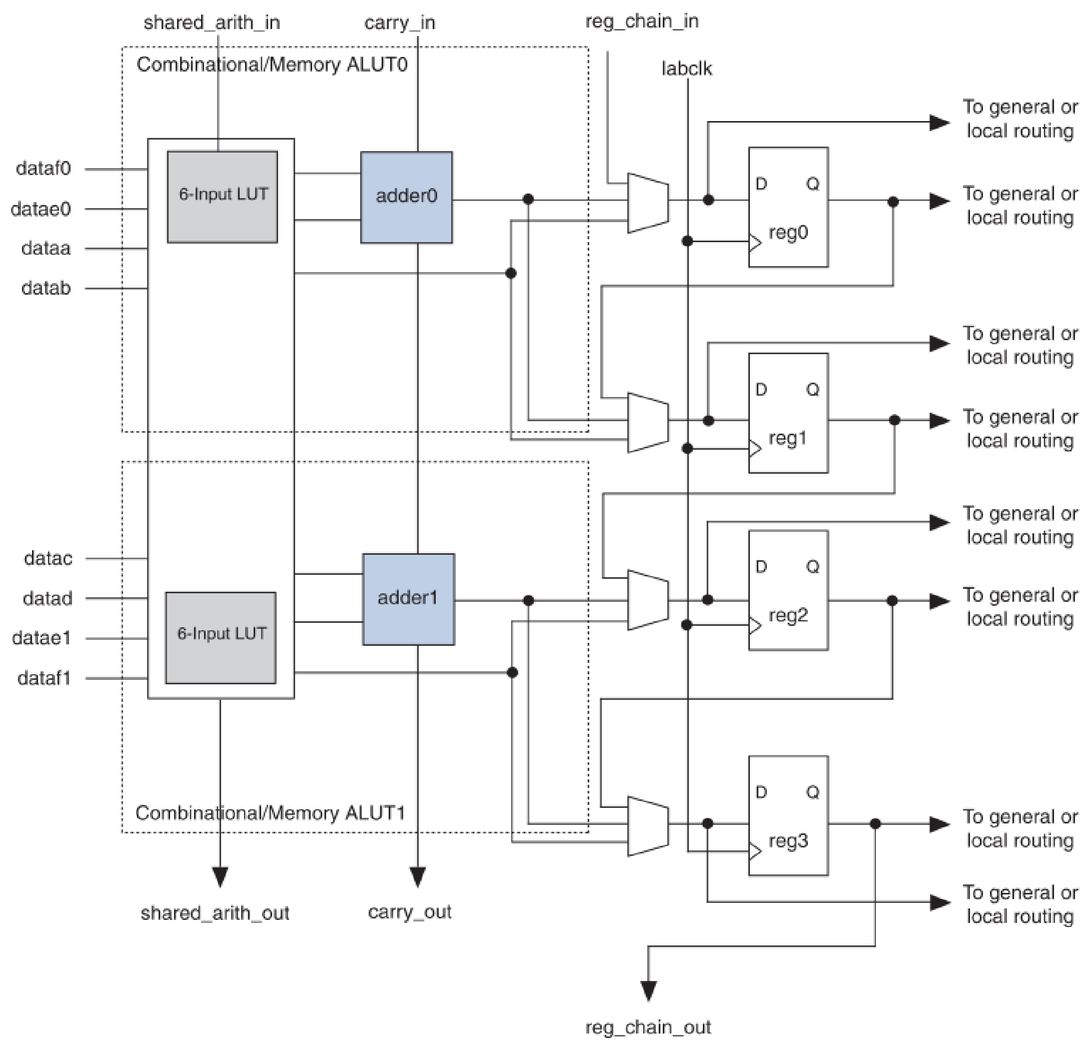
- Programmable connection and switch boxes



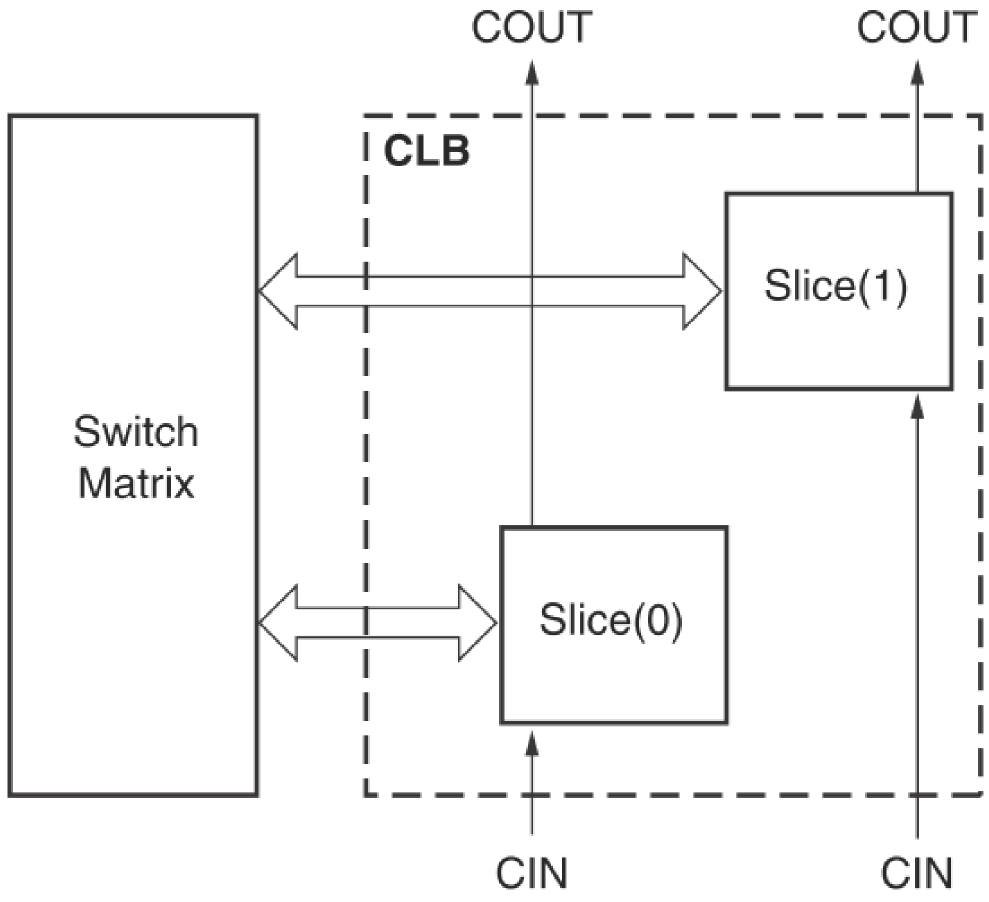
ALTERA Stratix V, Logic Array Block (LAB)


C4
C12
Row Interconnects of Variable Speed and Length
R20
R4
ALMs
Direct link interconnect from adjacent block
Direct link interconnect from adjacent block
Direct link interconnect to adjacent block
Direct link interconnect to adjacent block
Local Interconnect
LAB
MLAB
Local Interconnect is Driven from Either Side by Columns & LABs, and from Above by Rows
Column Interconnects of Variable Speed and Length

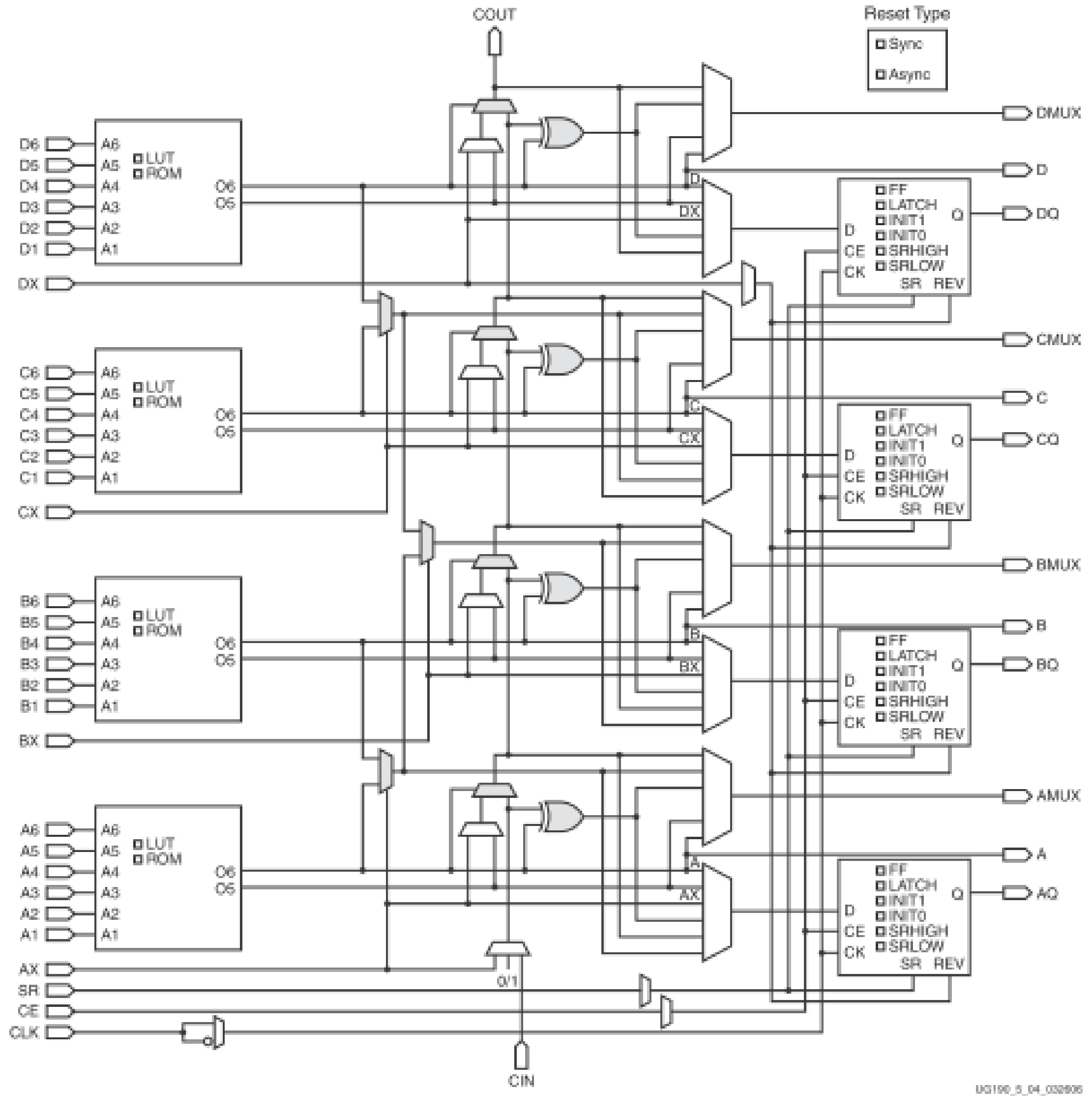
ALTERA Stratix V, Adaptive Logic Module (ALM)



Xilinx Virtex-5, Configurable Logic Block (CLB)



Xilinx Virtex-5, Slice (SLICEM)



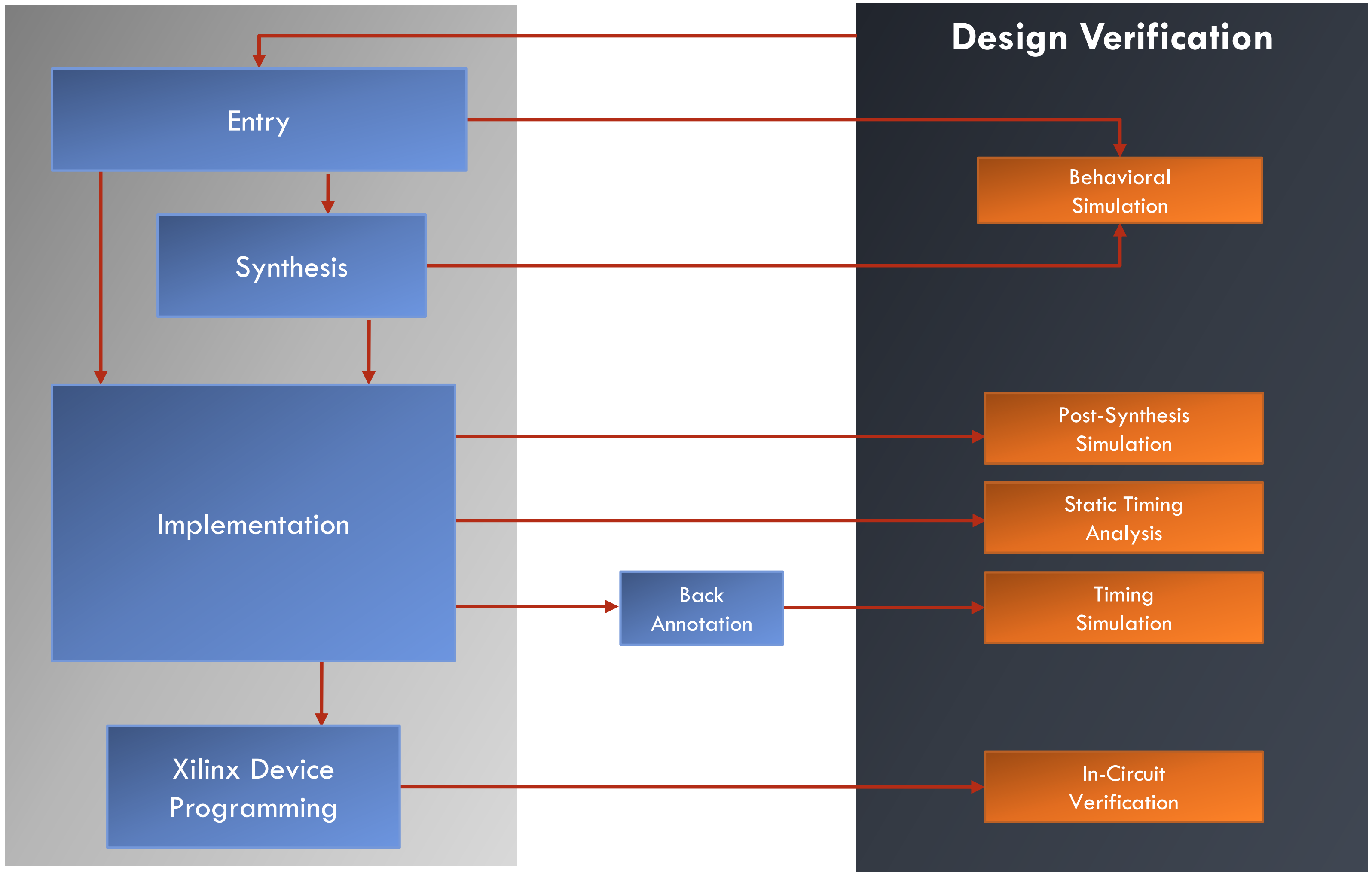
Hard Blocks

- Built-in functionality for a multitude of applications
  - Programmable I/O (LVDS, SSTL, HSTL, RSDS, and more)
  - High speed serializers/deserializers (SerDes) for 10Gbs rates
  - Embedded RAM
  - Embedded DSP blocks
  - Embedded processors (PPC440, ARM Cortex-A9, ARM Cortex-M3)
  - Multi clock, multi phase clock managers
- Significant advantages over general purpose logic
- Tight connection to the rest of the FPGA fabric

Hard blocks - 2

- Design entry varies depending on the hard block
  - General HDL (VHDL, Verilog) coding
  - Vendor specific HDL coding
  - IP cores
  - Vendor tools schematic entry
- Some blocks can be utilized in many different ways
  - Some require specific design entries
- Knowledge of vendor specific architecture is **essential**

Xilinx ISE Design Flow



Design with HDL

- Advantages of using an HDL (VHDL or Verilog)
  - Top-down approach for large projects
  - Functional simulation early in the design flow
  - Decreased design time
  - Reduced number of errors
  - Allows you to apply the automation techniques used by the synthesis tool
  - Early testing of various design implementations
  - Reuse of register transfer level (RTL) code

Design entry for Xilinx FPGAs

- Instantiation
  - Control the exact placement of the individual blocks
- Inference
  - Complete flexibility and portability of the code to multiple architectures
  - Inference gives the tools the ability to optimize for performance, area, or power
- Macro Support
  - Used to instantiate primitives that are too complex to instantiate
- Coregen & Wizards
  - Through Xilinx CORE Generator or other Wizards
  - For large blocks of any FPGA primitive that cannot be inferred
  - We have to re-generate our cores for each architecture that we are targeting