ISTANBUL TECHNICAL UNIVERSITY

Digital System Design & Applications

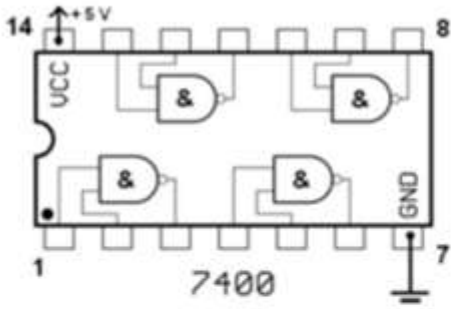
FPGAs and Digital Design Flow

RES. ASST. FIRAT KULA

Contents

- History of Programmable Logic
  - Good old days…
  - PLDs (1956 ->)
  - FPGAs (1985 ->)
- FPGA
  - Areas of Usage
  - Pros & Cons
  - FPGA Inner Structure
    - Programming In FPGA
    - LUT Based Logic Realization
    - Interconnections
- FPGA Design Flow
  - Design Entry
  - Synthesis
  - Implementation
  - Final Steps/Configuration

Good old days…



- ❑Discrete Logic – based on gates or small packages containing small logic building blocks (1-bit adders at most)
- ❑De Morgan's Theorem: Theoretically we can build anything with just 2-input NAND or NOR gates
- ❑Tedious, expensive, slow, prone to wiring errors…

- *Building an 8-bit breadboard computer: https://www.youtube.com/watch?v=HyznrdDSSGM&list=PLowKtXNTBypGqImE405J2565dvjafglHU*

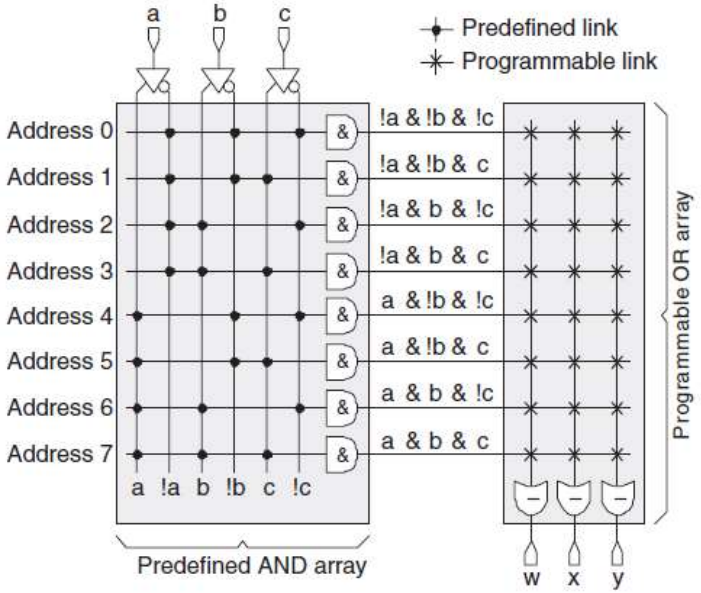
Programmable Logic Devices(PLD)

- Simple Programmable Logic Devices (SPLD)

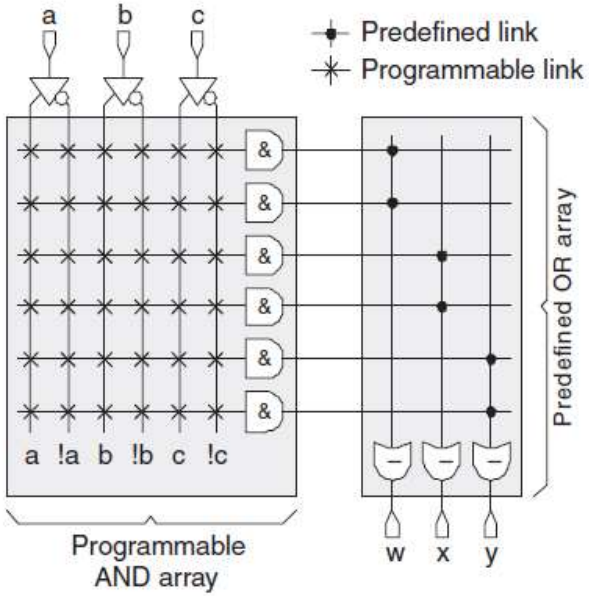
  ** PAL (Programmable Array Logic) , PLA (Programmable Logic Array)
- Complex PLD's (CPLD)

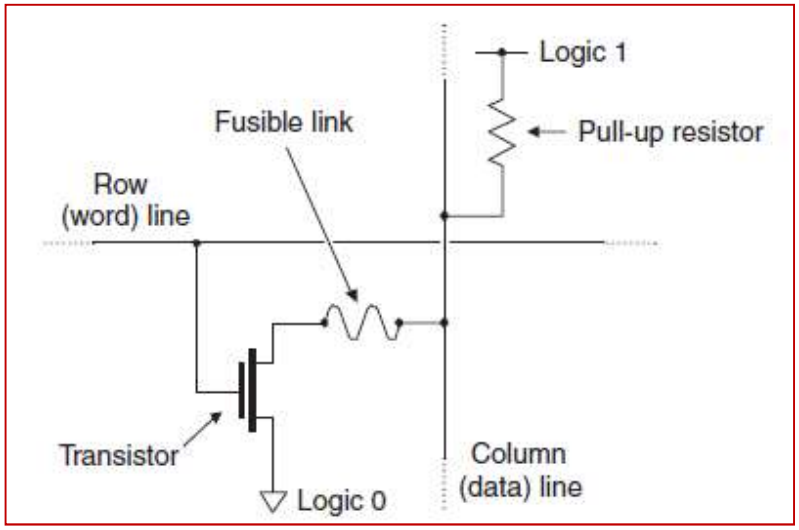
  ** Multiple SPLD type of blocks with interconnect
- Field Programmable Gate Array (FPGA)

PROM

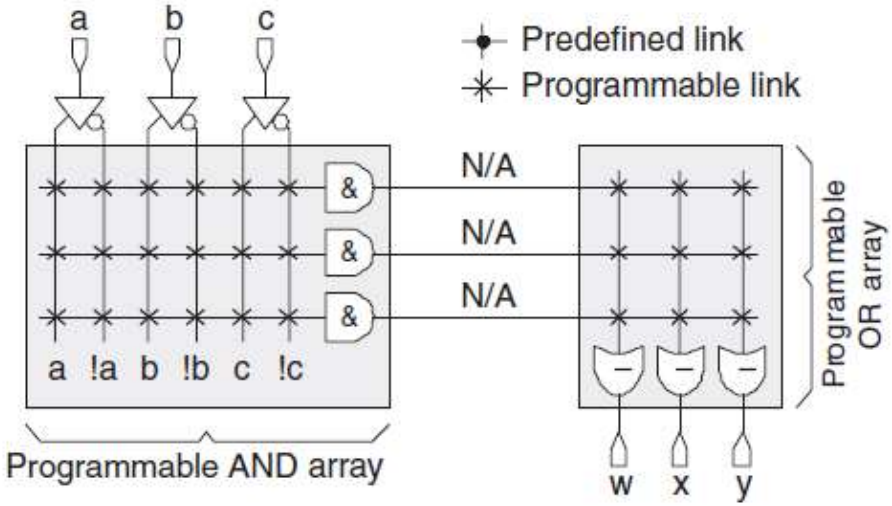


PLA (1975)



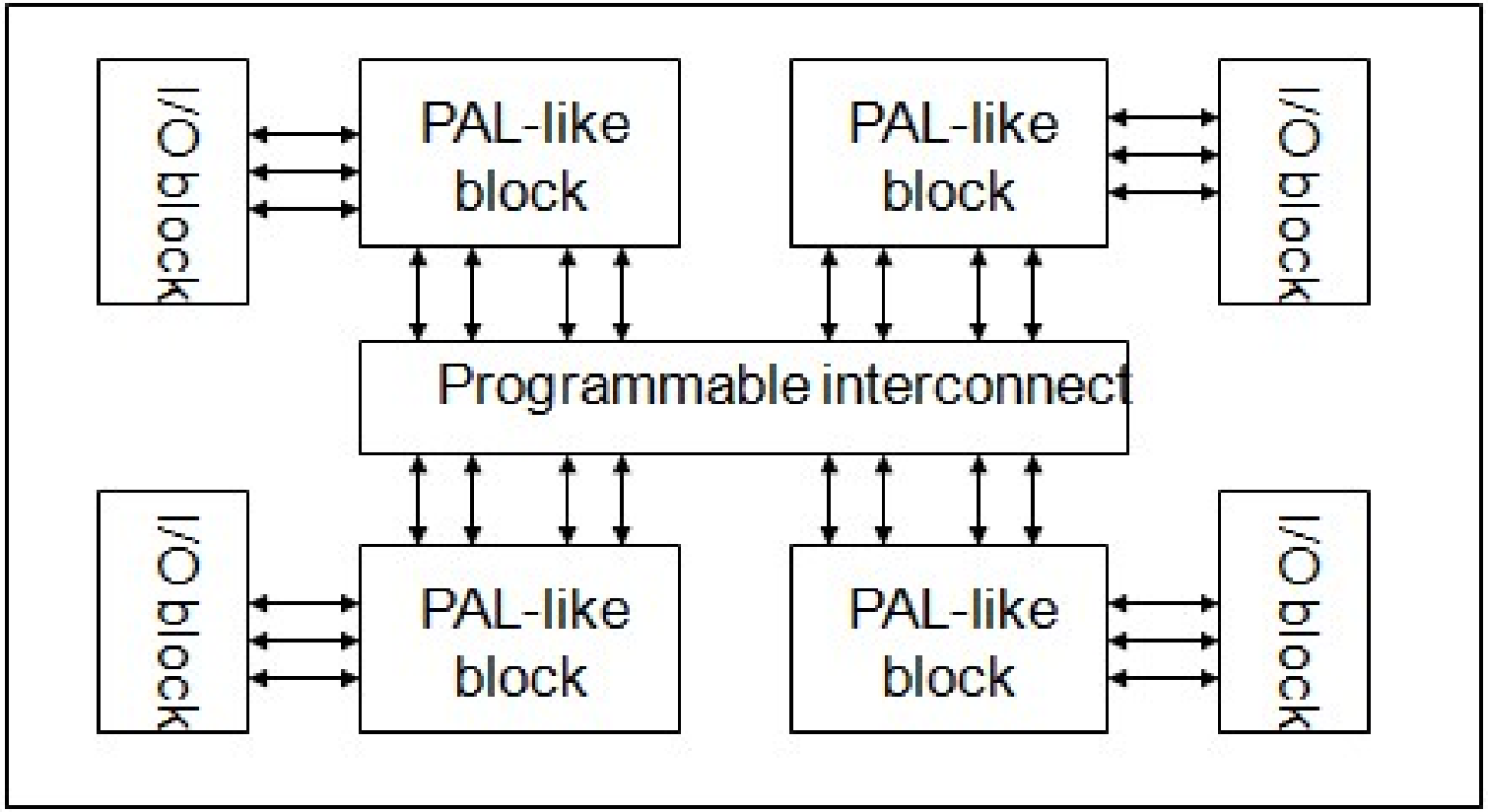


PAL (1978)

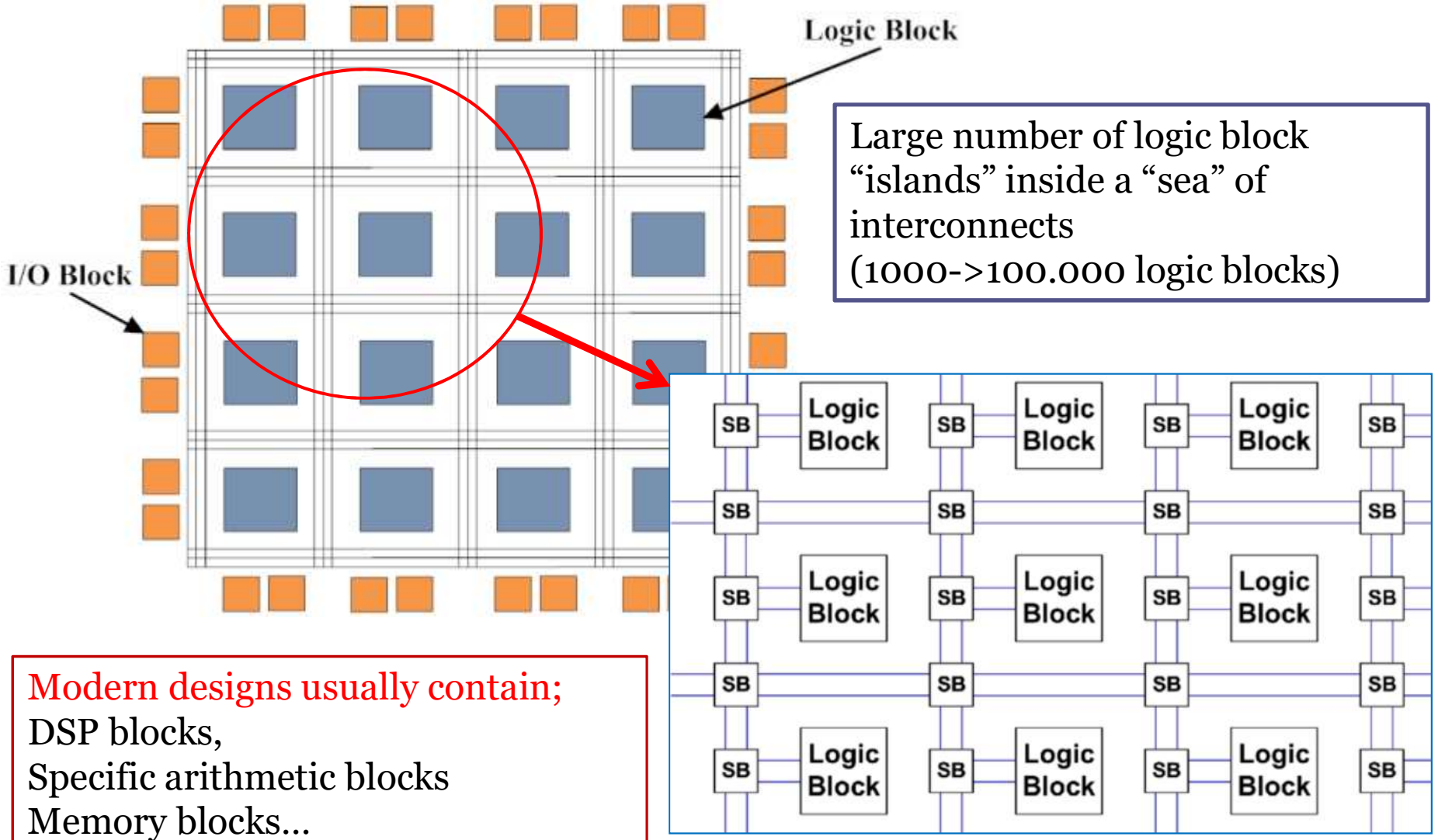


- *Ref: M. Clive, The Design Warrior's Guide to FPGA's, Elsevier, 2004*

CPLD (1983)


I/O block
PAL-like block
PAL-like block
I/O block
Programmable interconnect
I/O block
PAL-like block
PAL-like block
I/O block

FPGA (1985)

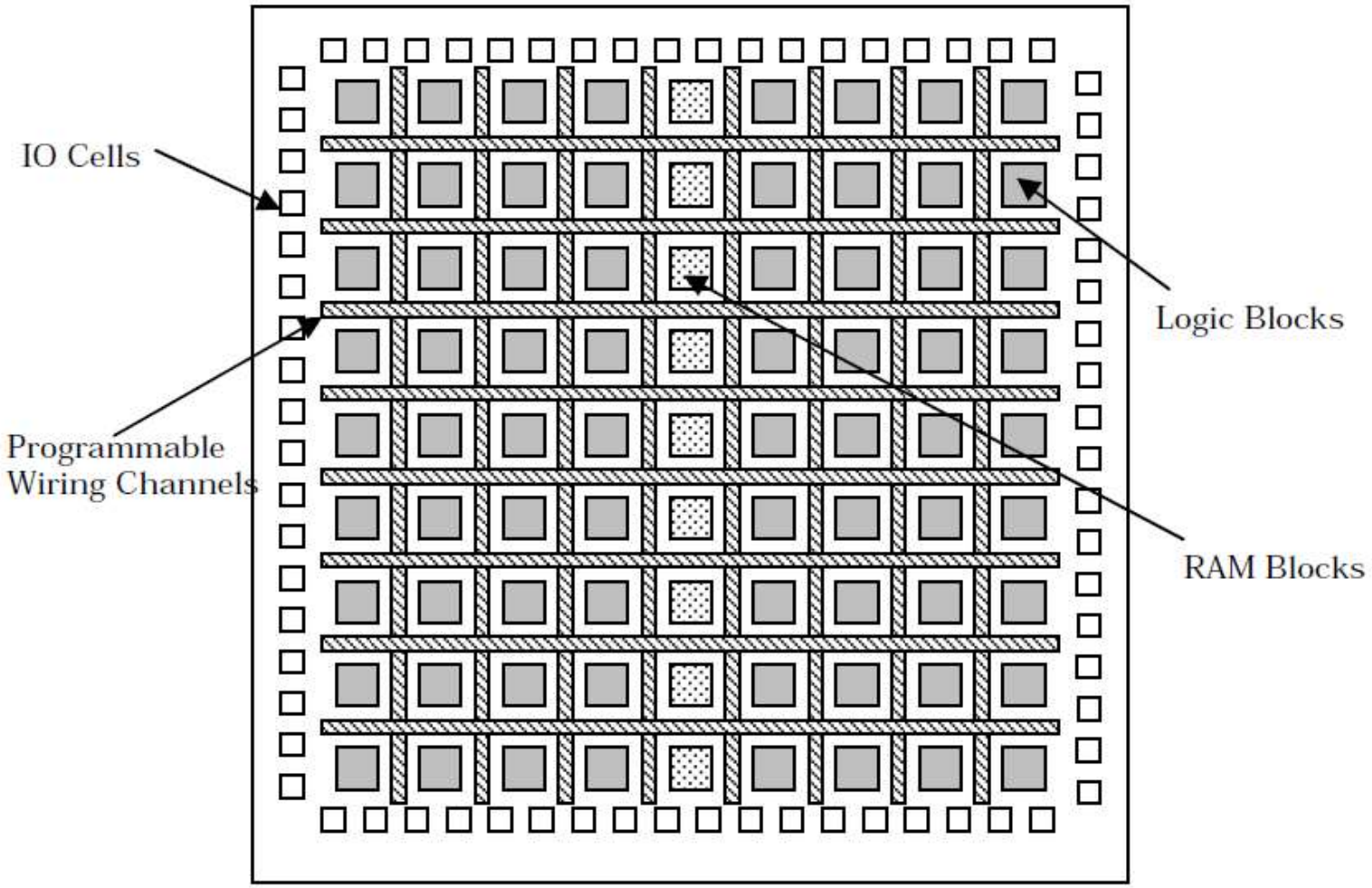


Large number of logic block "islands" inside a "sea" of interconnects (1000->100.000 logic blocks)

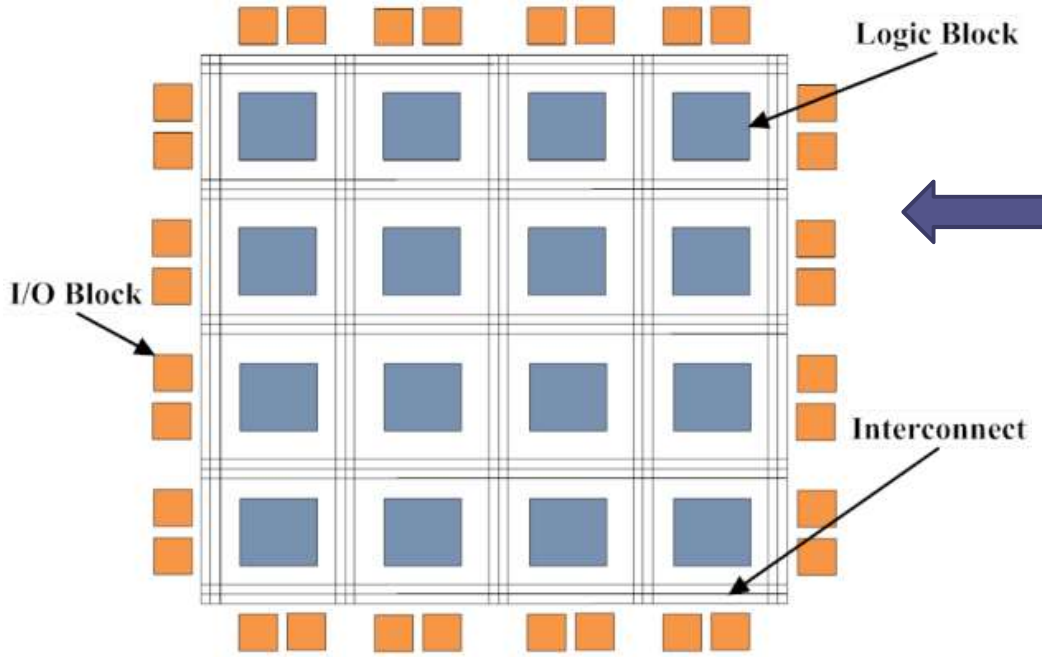
Modern designs usually contain;
DSP blocks,
Specific arithmetic blocks
Memory blocks…

- *Ref: S Brown and J. Rose, Architecture of FPGAs and CPLDs – A Tutorial, Department of Electrical and Computer Engineering, University of Toronto*

FPGA (1985)

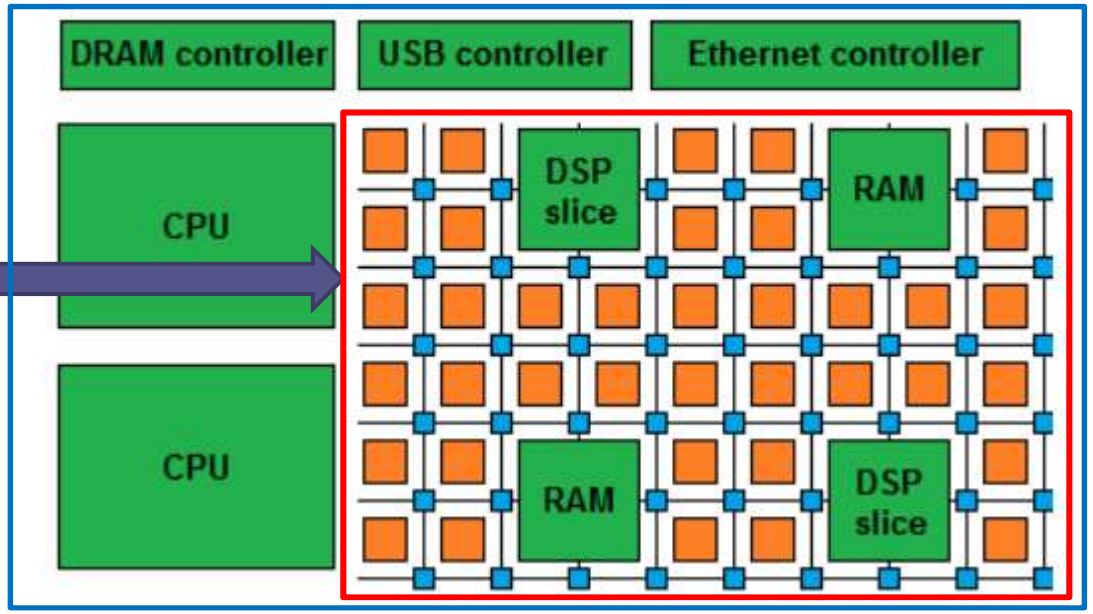


- *Ref: Z. Navabi, Digital Design and Impelentation with Field Programmable Devices, Springer, 2004*

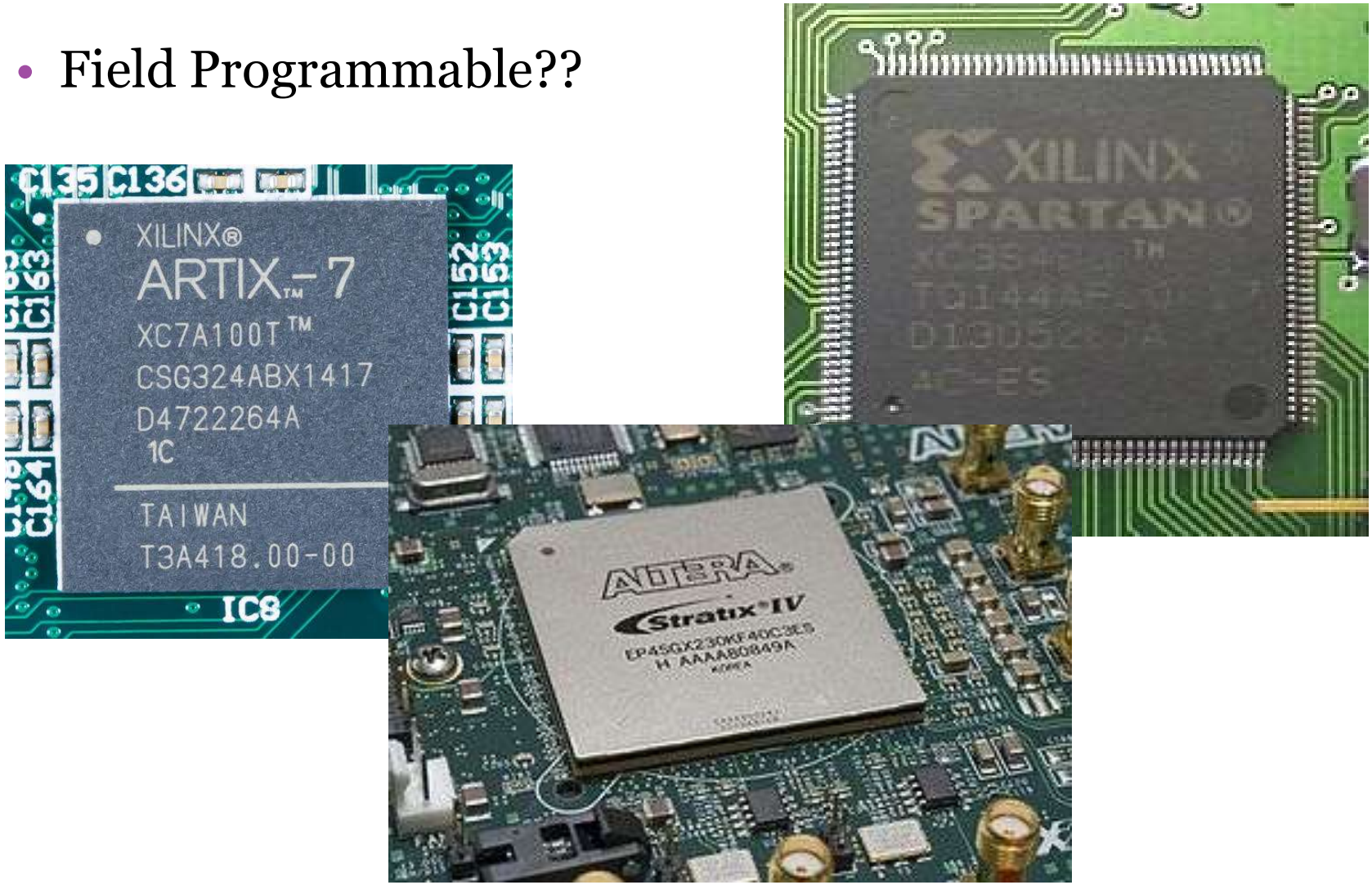


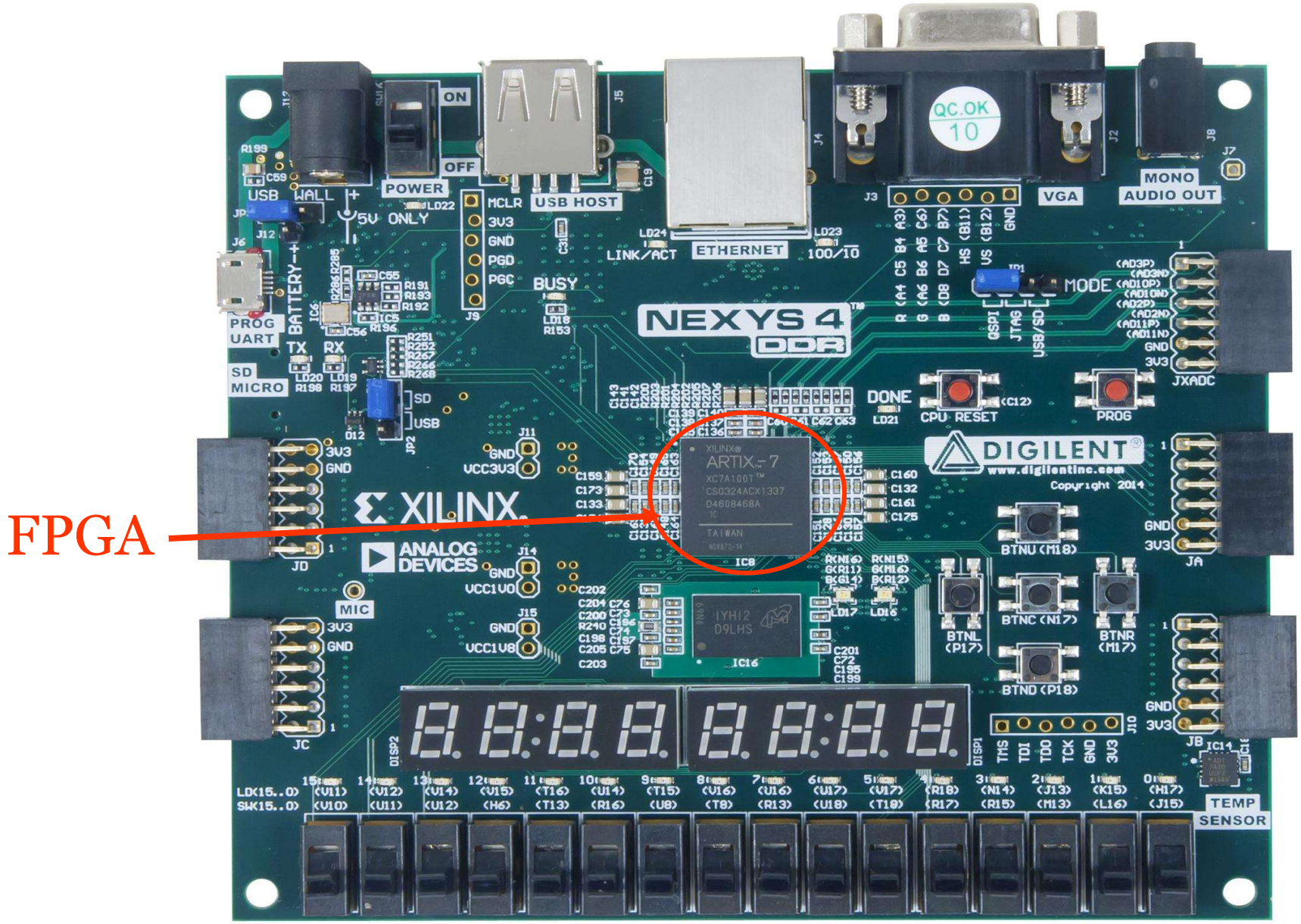
Classical Designs

Modern Designs



- Field Programmable??




FPGA
NEXYS 4
DDR
XILINX
ARTIX-7
DIGILENT
www.digilentinc.com
ANALOG DEVICES
VGA
ETHERNET
USB HOST
POWER
MONO AUDIO OUT
PROG UART
SD MICRO
JXADC
JA
JB
JC
JD
CPU RESET
PROG
BTNU (M18)
BTNL (P17)
BTNC (N17)
BTNR (M17)
BTND (P18)
MIC
TEMP SENSOR

Areas of Usage

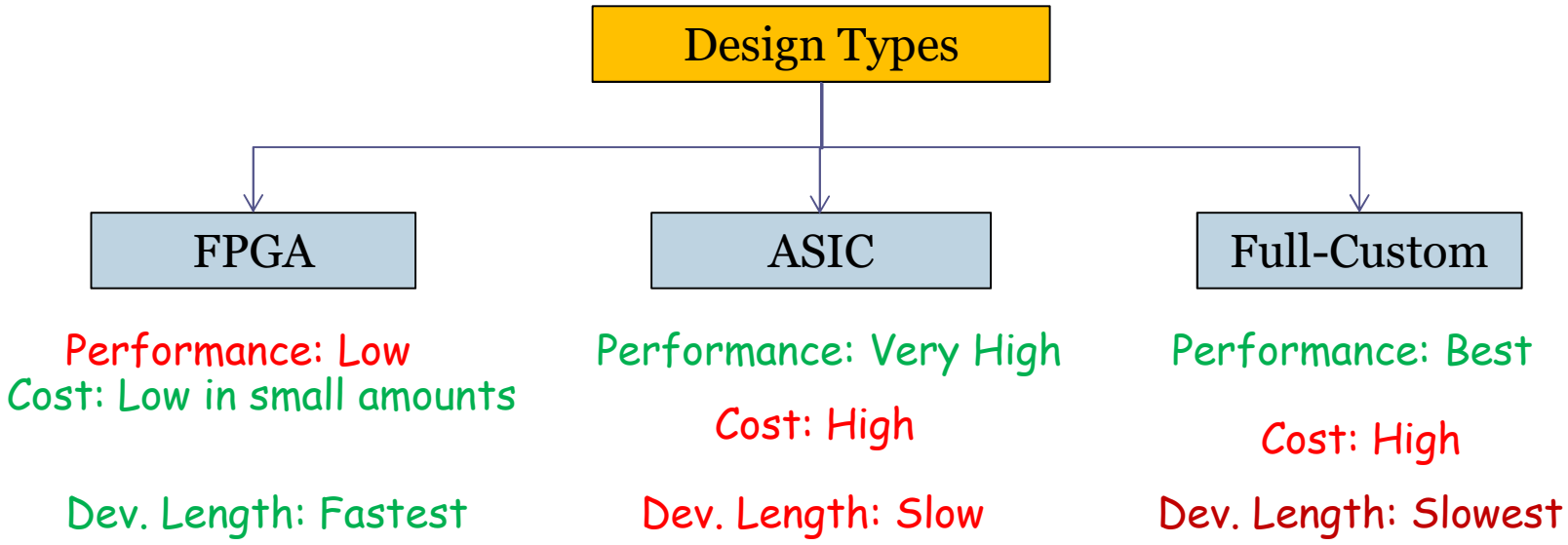
- Digital signal processing, communications, aerospace, cryptography, machine learning…
- Hardware acceleration (Parallel computing)
- Design prototyping
- Can be substitute to ASIC when a design needs to be quickly produced in small quantities.

Pros & Cons

+ Faster designing process
+ Cheaper for small quantity productions
+ Reconfigurable

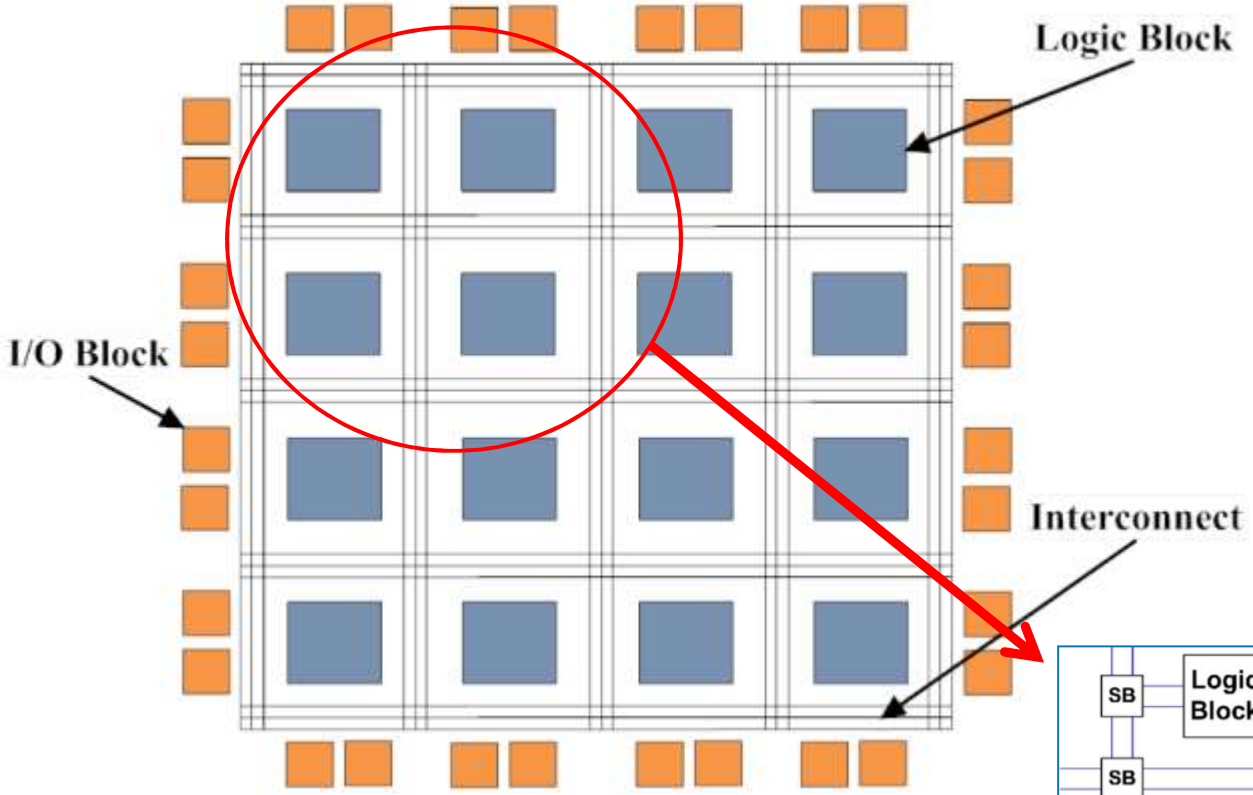
– Performance-wise, it is not as good as ASIC (Application Specific Integrated Circuits).
–Not suitable for mass marketing, since the high cost of the single FPGA makes it impractical compared to the much cheaper mass producing costs of ASIC

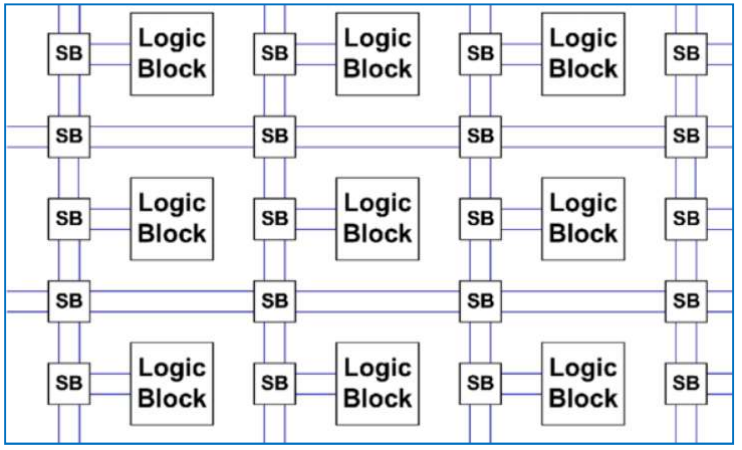
Different Digital Design Approaches



- Manufacturing cycle for ASIC is very costly, lengthy and engages lots of manpower. Mistakes not detected at design time have large impact on development time and cost
- FPGAs are perfect for rapid prototyping of digital circuits, and as hardware accelerators

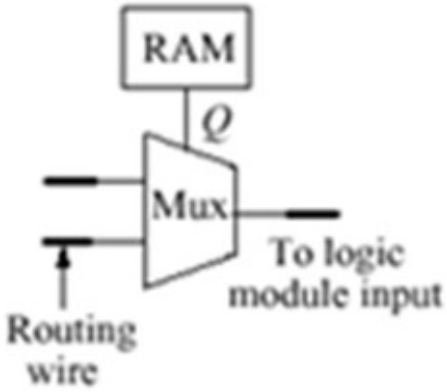
Inside of an FPGA?

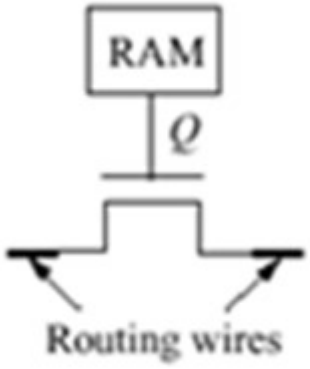


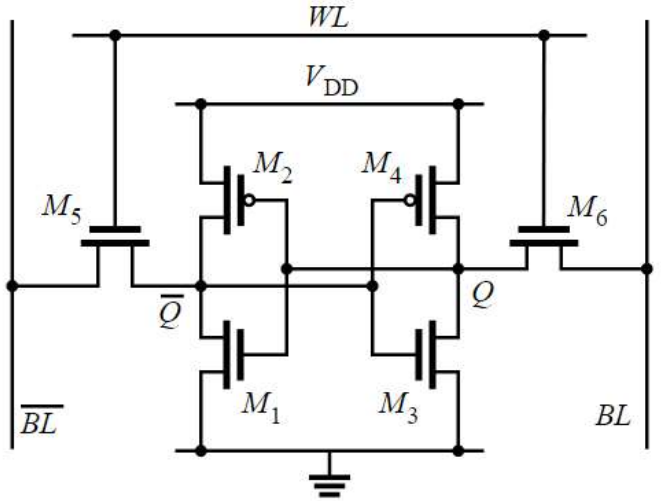


Programming in FPGA

- Mostly SRAM's (Very suitable with the CMOS process the FPGA's are made from).
- Volatile (Needs to be reprogrammed after power-down)
- Used in LUT elements, interconnection switches …



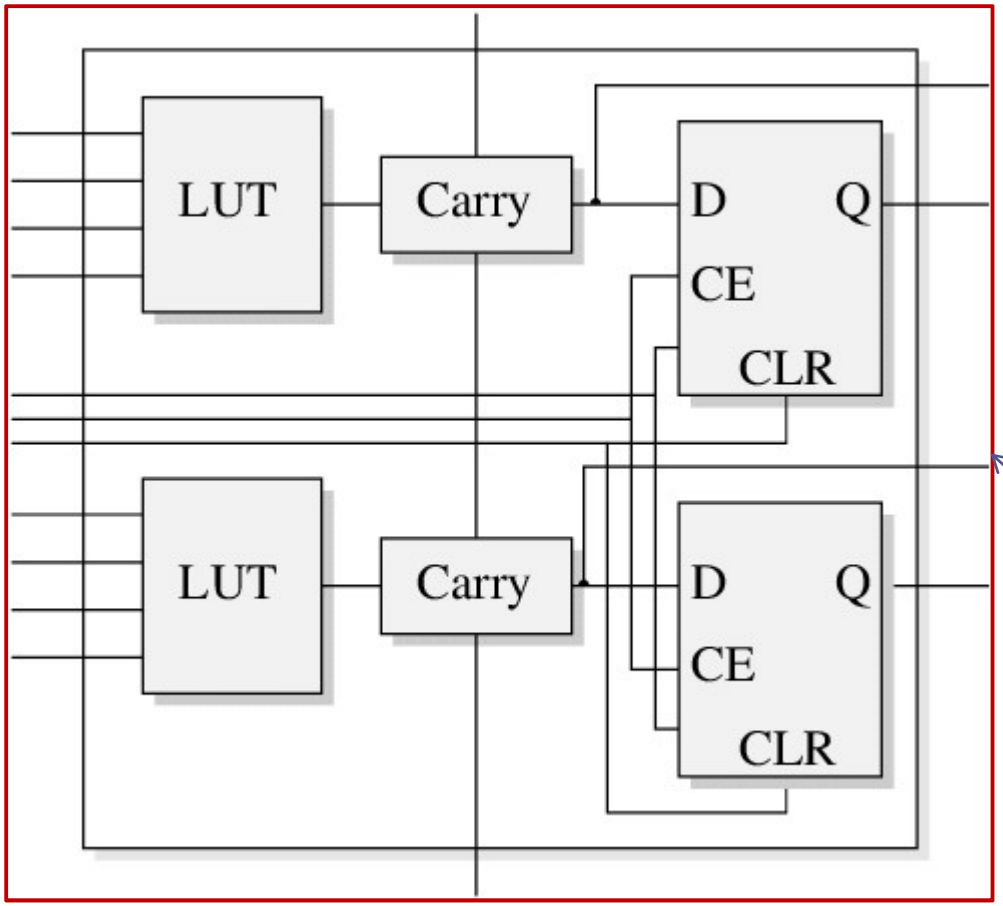


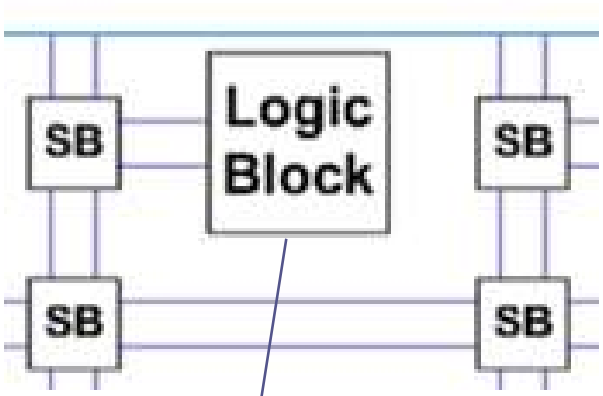


6-transistor SRAM circuit

- *Ref: I. Kuon, A. Tessier, J Rose, FPGA Architecture: Survey and Challenges, Foundations and Trends in Electronic Automation Vol. 2, 2008*

Look-up Table (LUT) Based Logic

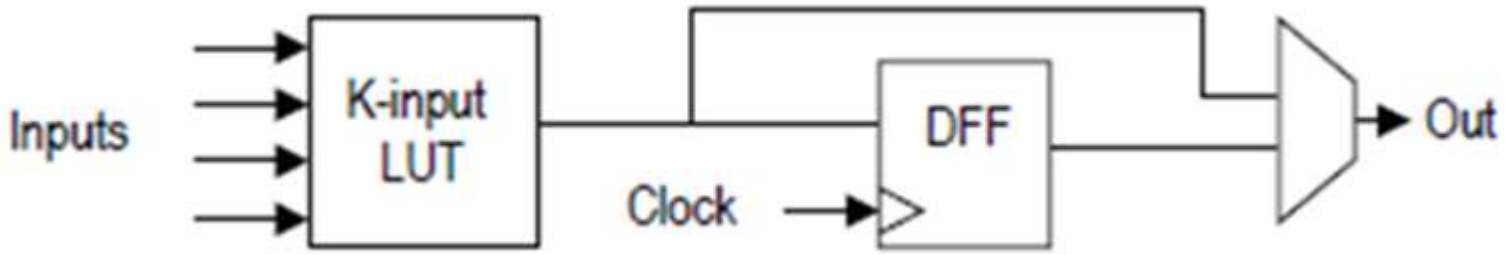




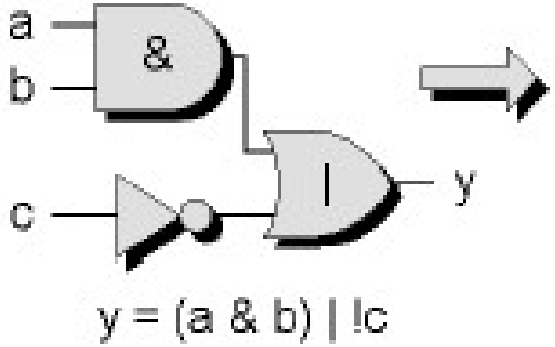
Logic blocks in Xilinx devices consists of multiple **slices**

- *Ref: B. Pasca, Customizing Floating-Point Operators for Linear Algebra Acceleration on FPGAs, Universite de Lyon, 2008*

Look-up Table (LUT) Based Logic



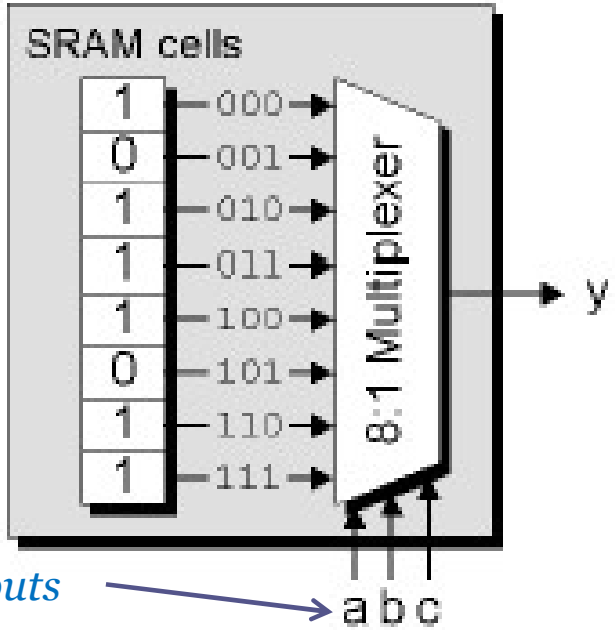
Logic Function



Truth Table

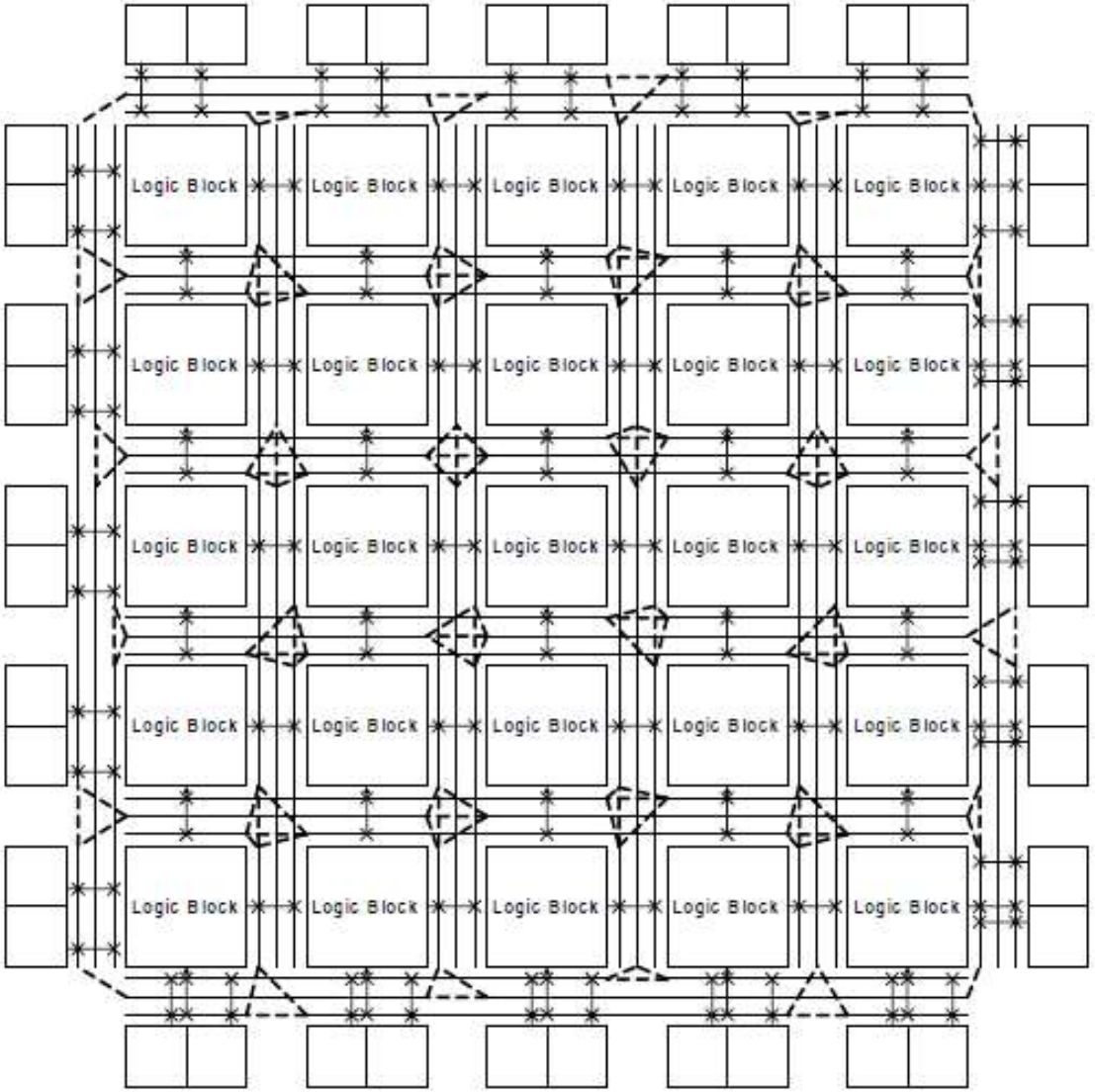
a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Programmed LUT

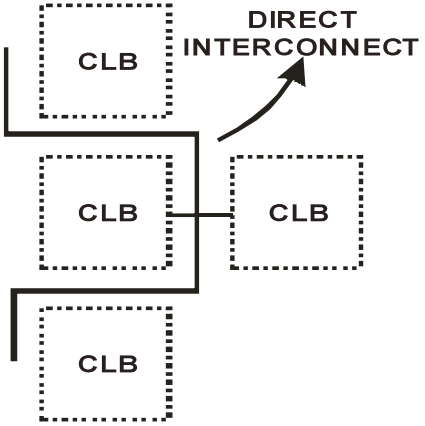


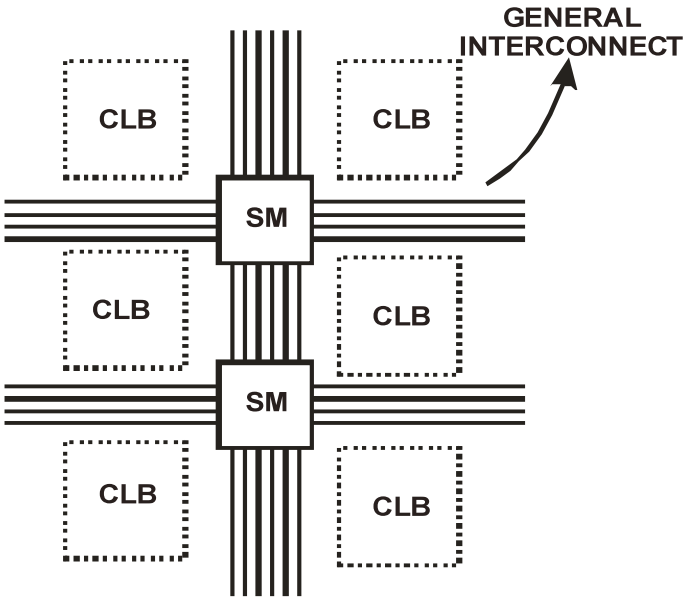
- *Ref: M. Clive, The Design Warrior's Guide to FPGA's, Elsevier, 2004*

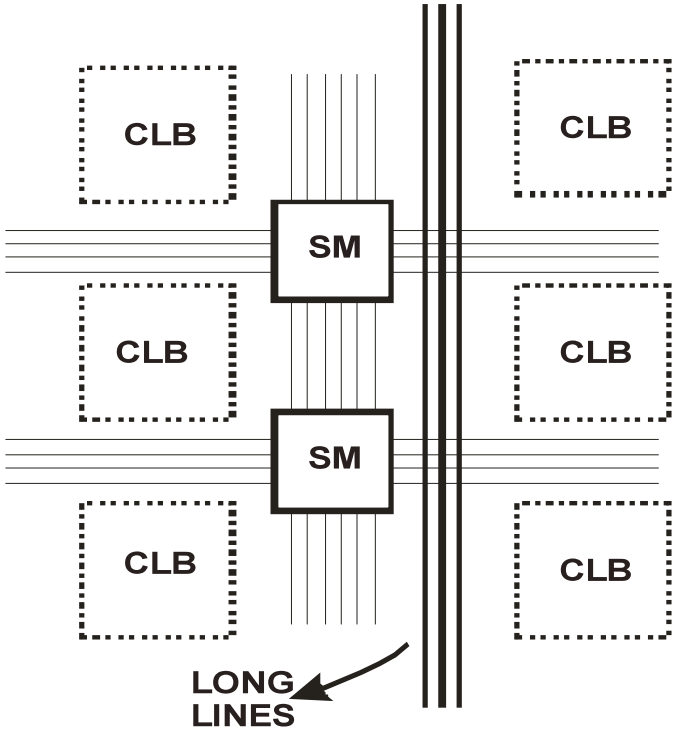
Interconnections



Interconnections

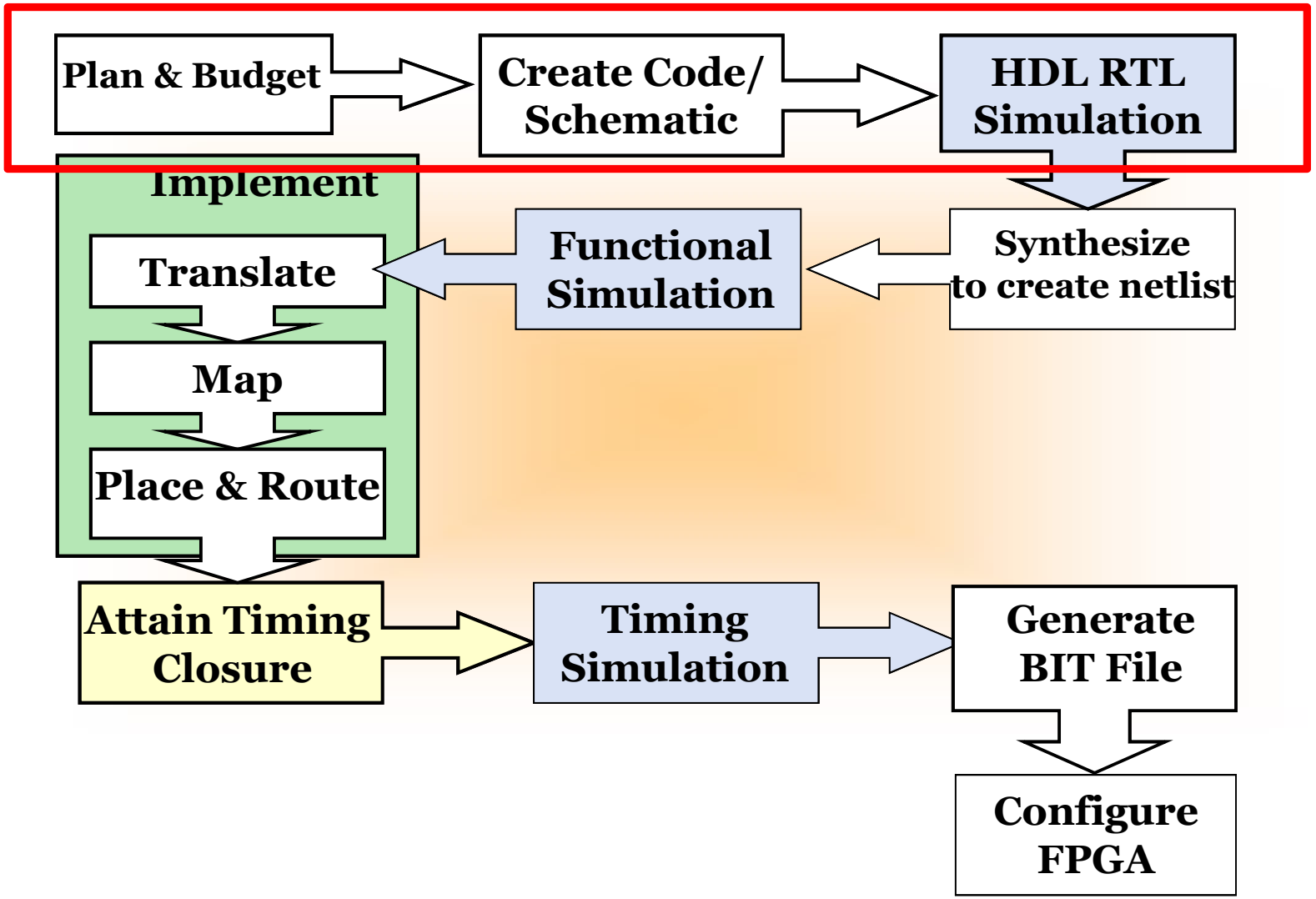






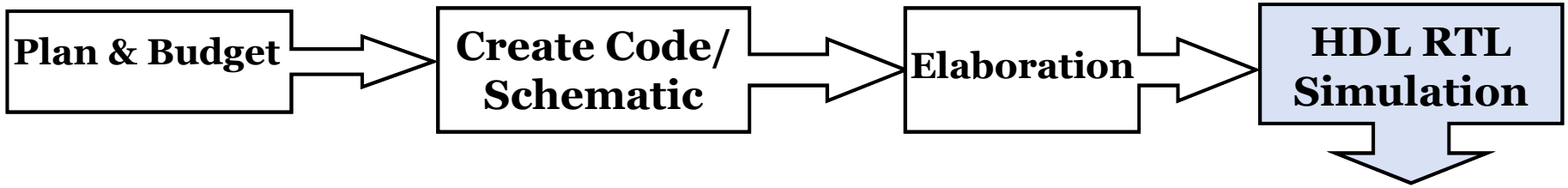
Interconnection switches are also programmed with SRAM cells

FPGA Design Flow

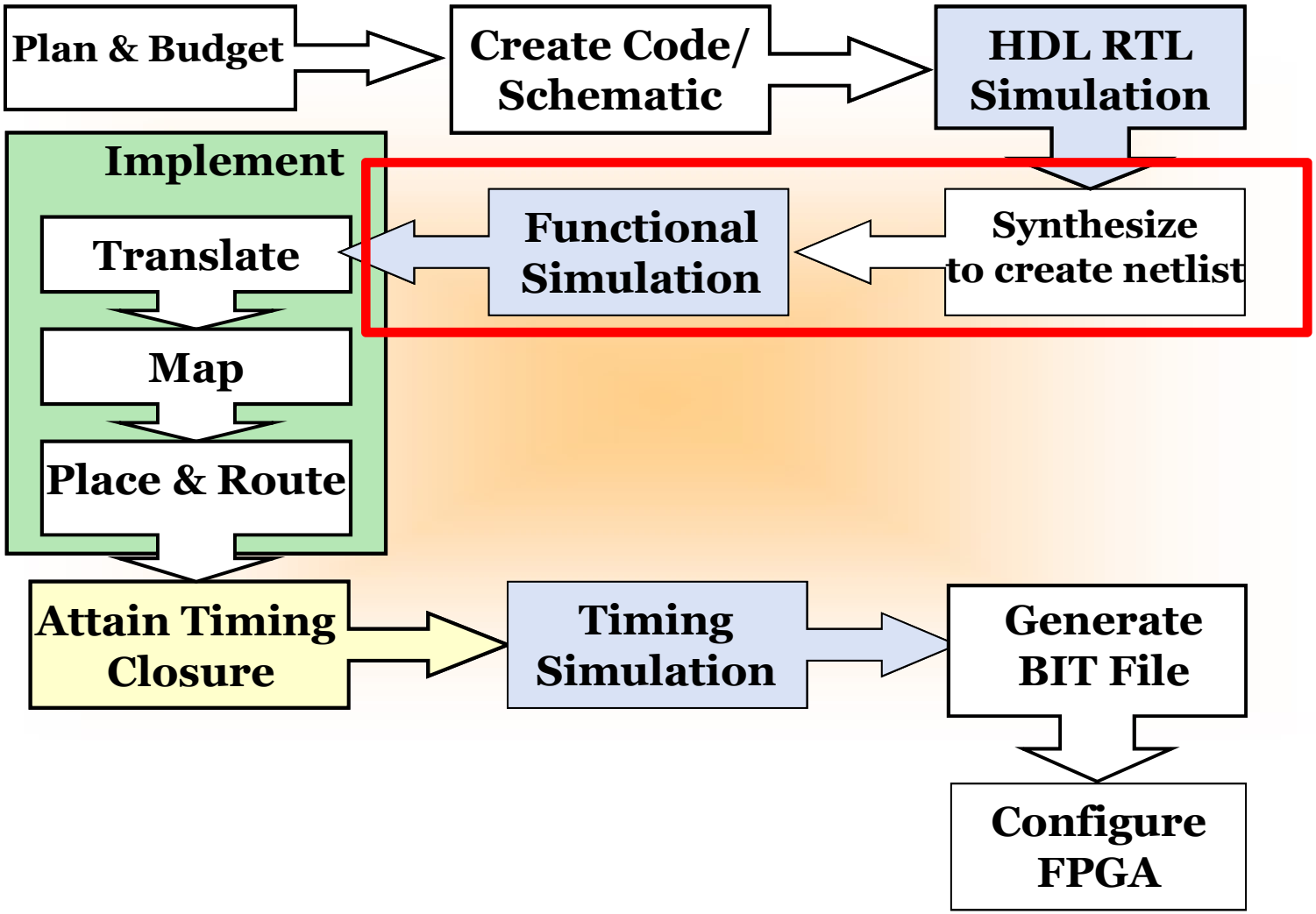


FPGA Design Flow – Design Entry

- **Plan and budget:** Selecting an FPGA chip/board that is suitable to the application on hand
  (Number of logic blocks, special blocks, power consumption, peripherals…)
- **Code your design:** Express the design using Hardware Description Languages (HDL) *Ex: VHDL, Verilog*
- **Elaboration:** The HDL code is converted to an RTL representation
- Simulate this RTL representation to ensure that it works as expected!



FPGA Design Flow

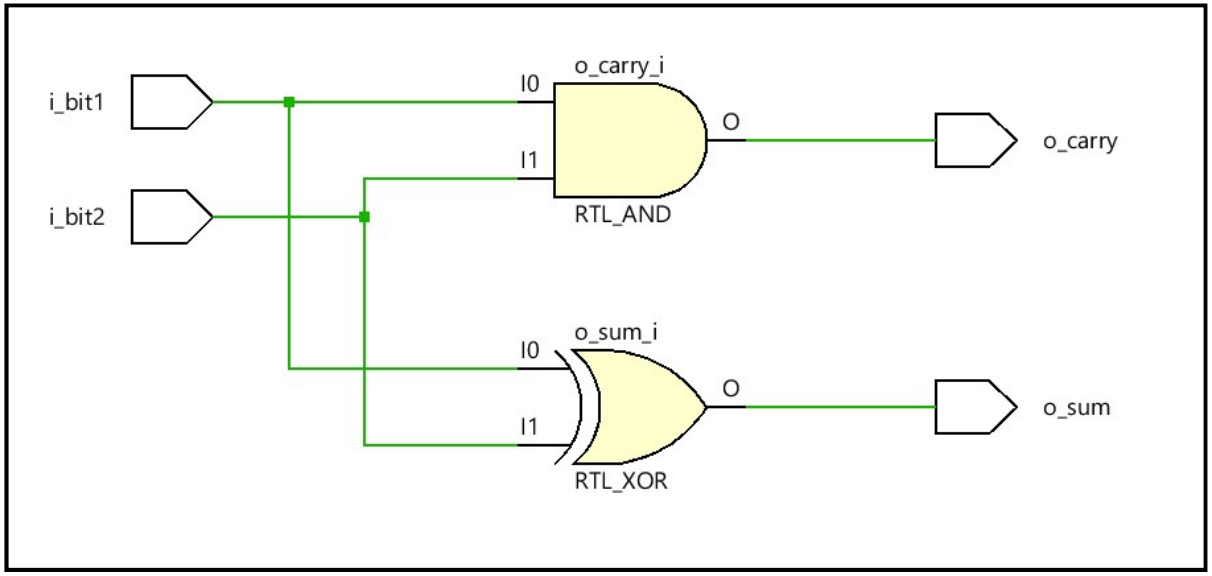


FPGA Design Flow - Synthesis

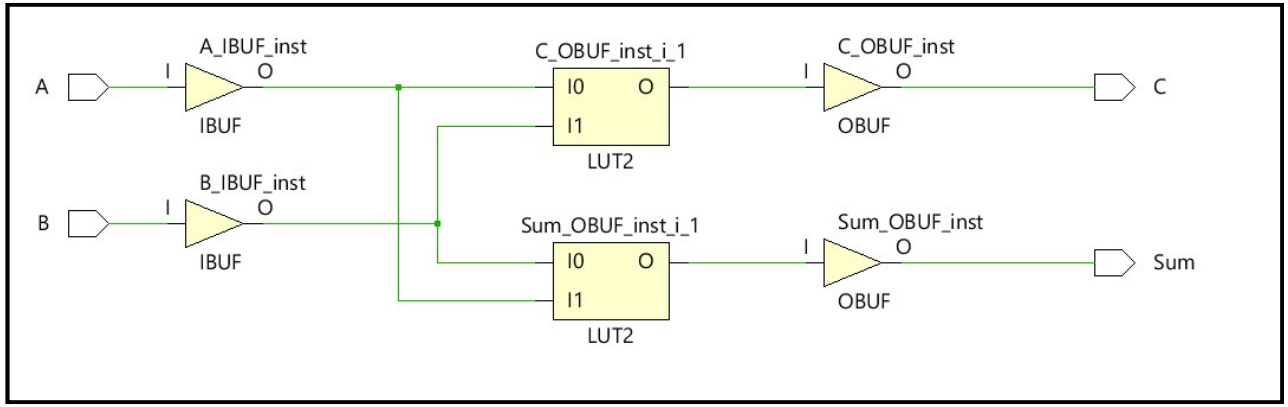
- Synthesizing means converting the HDL coded/RTL formatted design to a netlist consisting generic FPGA cell elements
- Synthesizing requires tools that use logic synthesis algorithms (High Level Synthesis - HLS), and they usually need vendor-specific standard cell libraries.
  - Some frequently used FPGA synthesis tools: XST , Synplify, Precision, FPGA Compiler
- A synthesised design must be simulated as well, in order to be sure that it shows the desired functionality (Post Synthesis Functional/Timing Simulation).

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity half_adder is
  port (
    i_bit1  : in std_logic;
    i_bit2  : in std_logic;
    --
    o_sum   : out std_logic;
    o_carry : out std_logic
    );
end half_adder;

architecture rtl of half_adder is
begin
  o_sum   <= i_bit1 xor i_bit2;
  o_carry <= i_bit1 and i_bit2;
end rtl;
```

Hardware Description Language (HDL) code

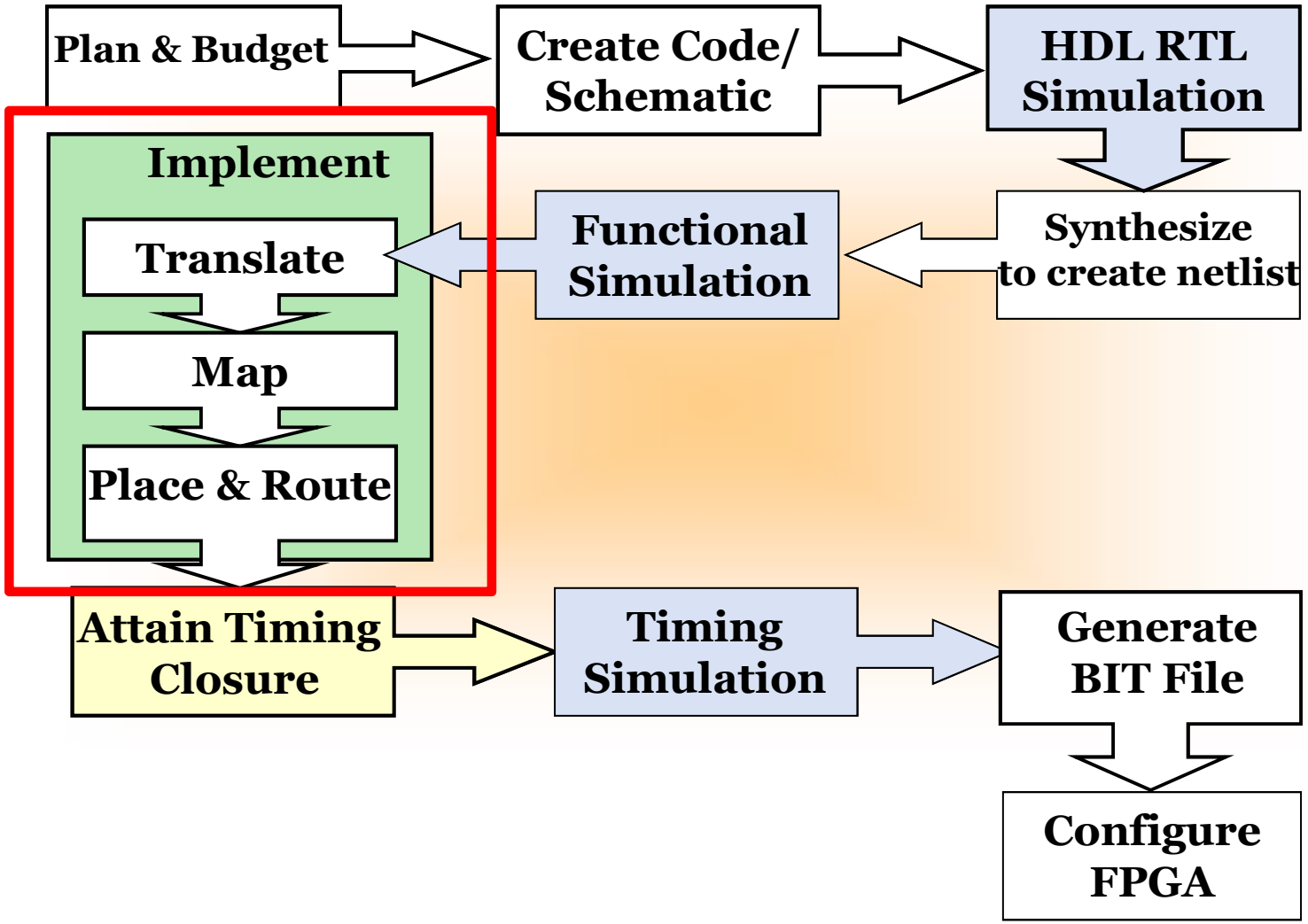


RTL based netlist schematic



Technology schematic

FPGA Design Flow

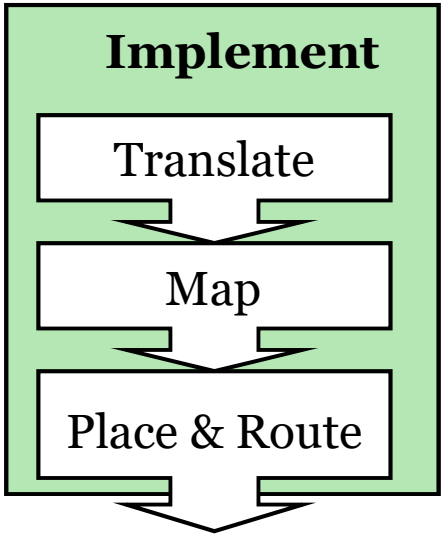


FPGA Design Flow - Implementation

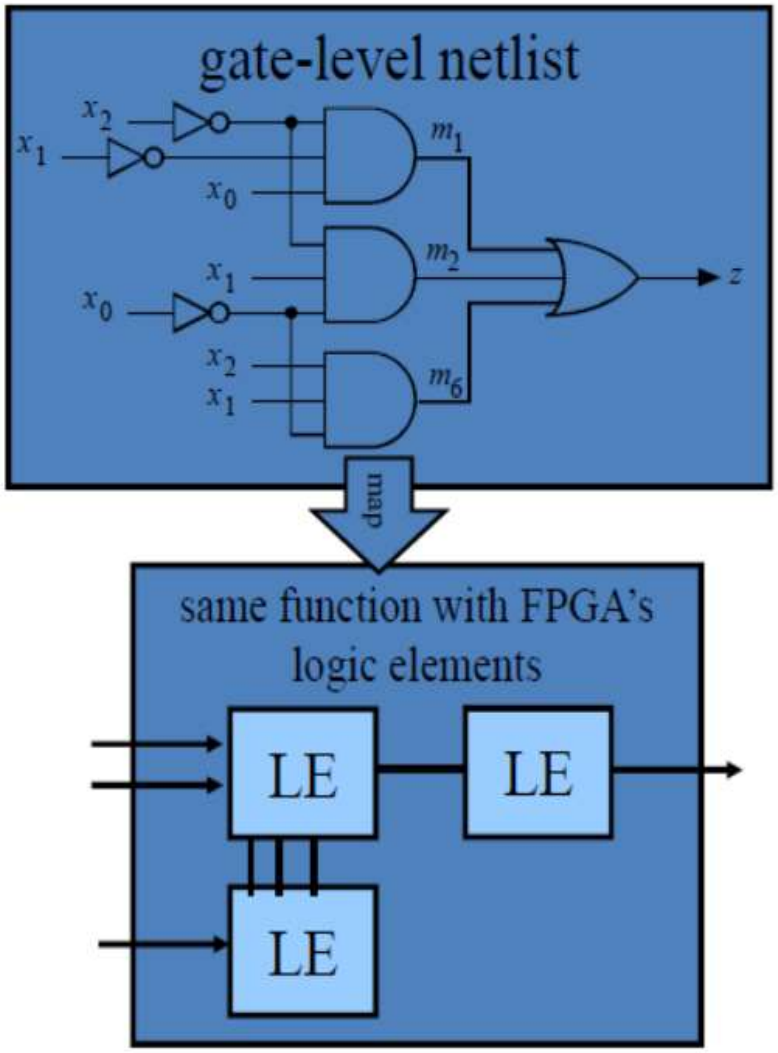
Translate: Merge all netlist files and express in terms of FPGA primitives and set it ready to be mapped

Map: Choose which device specific primitives will be used in place of translated design blocks

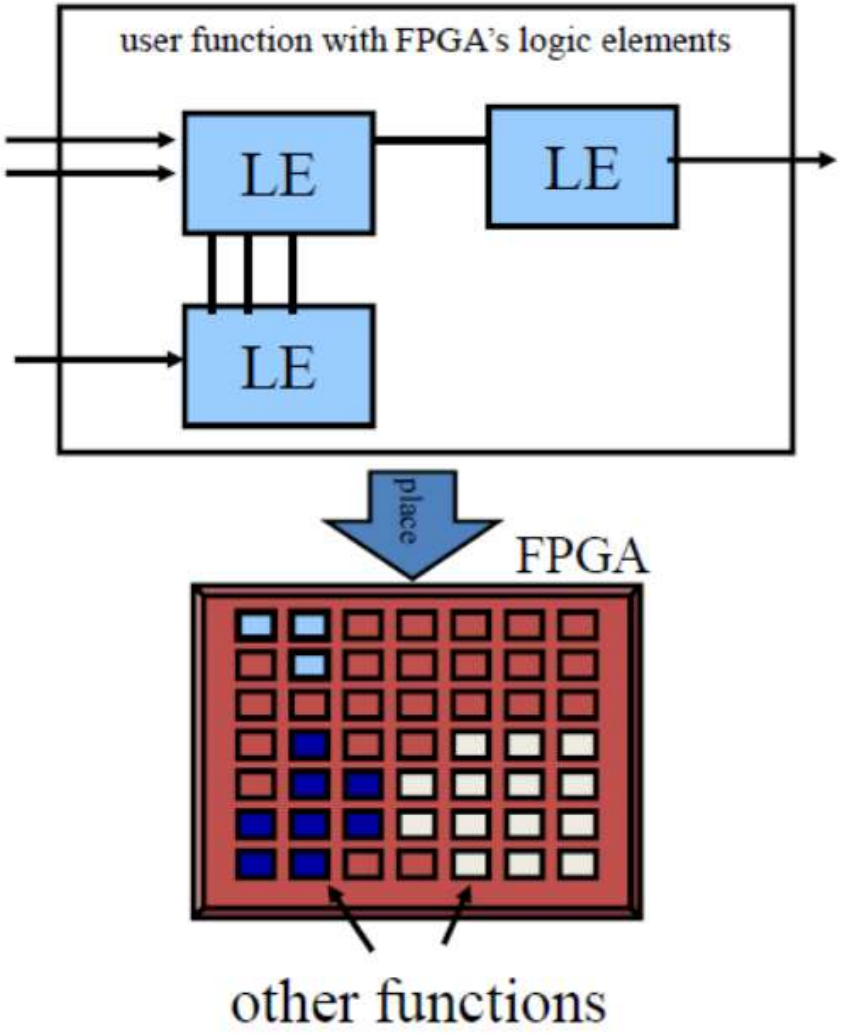
Place & Route: Place the design on the target device and make interconnections



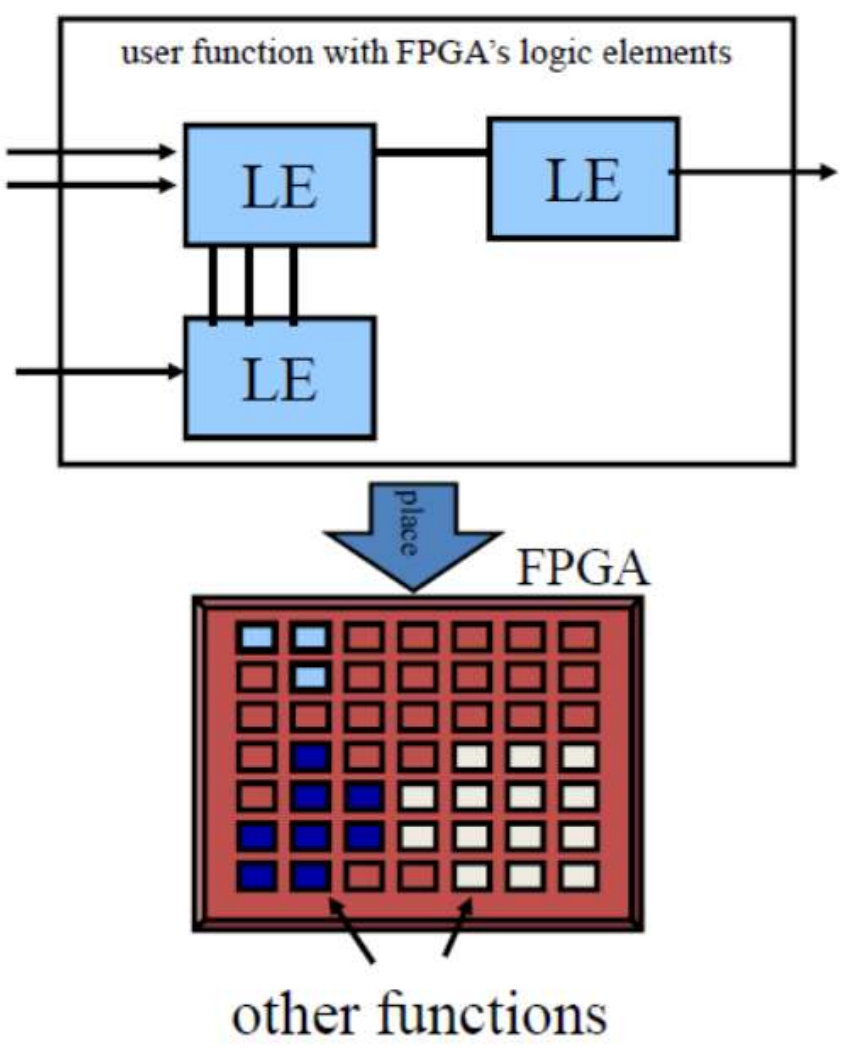
We need to use vendor specific implementation tools and device libraries.



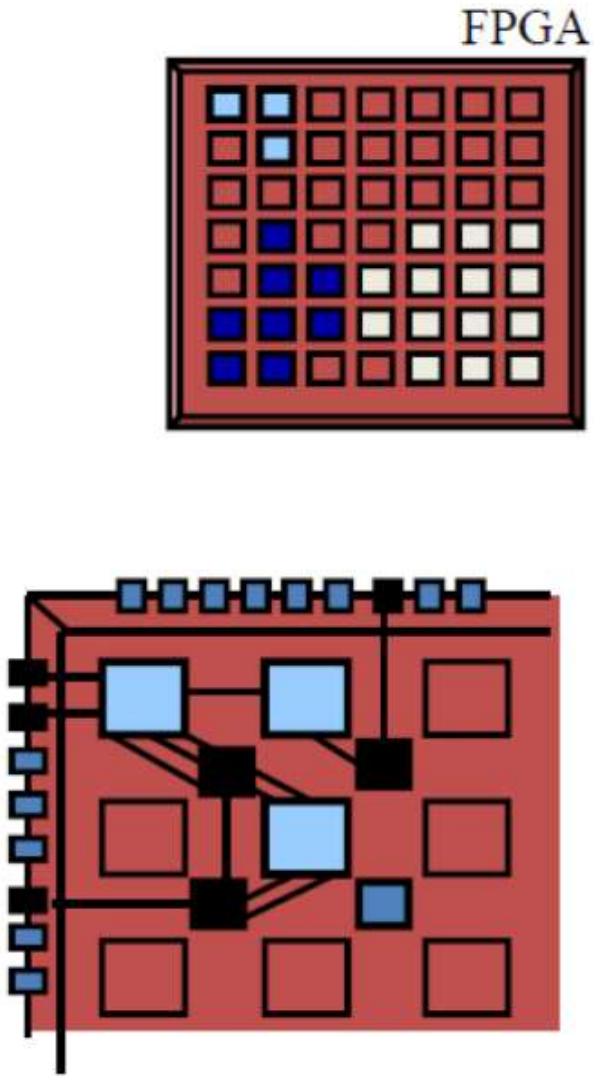
Synthesis+ Translate



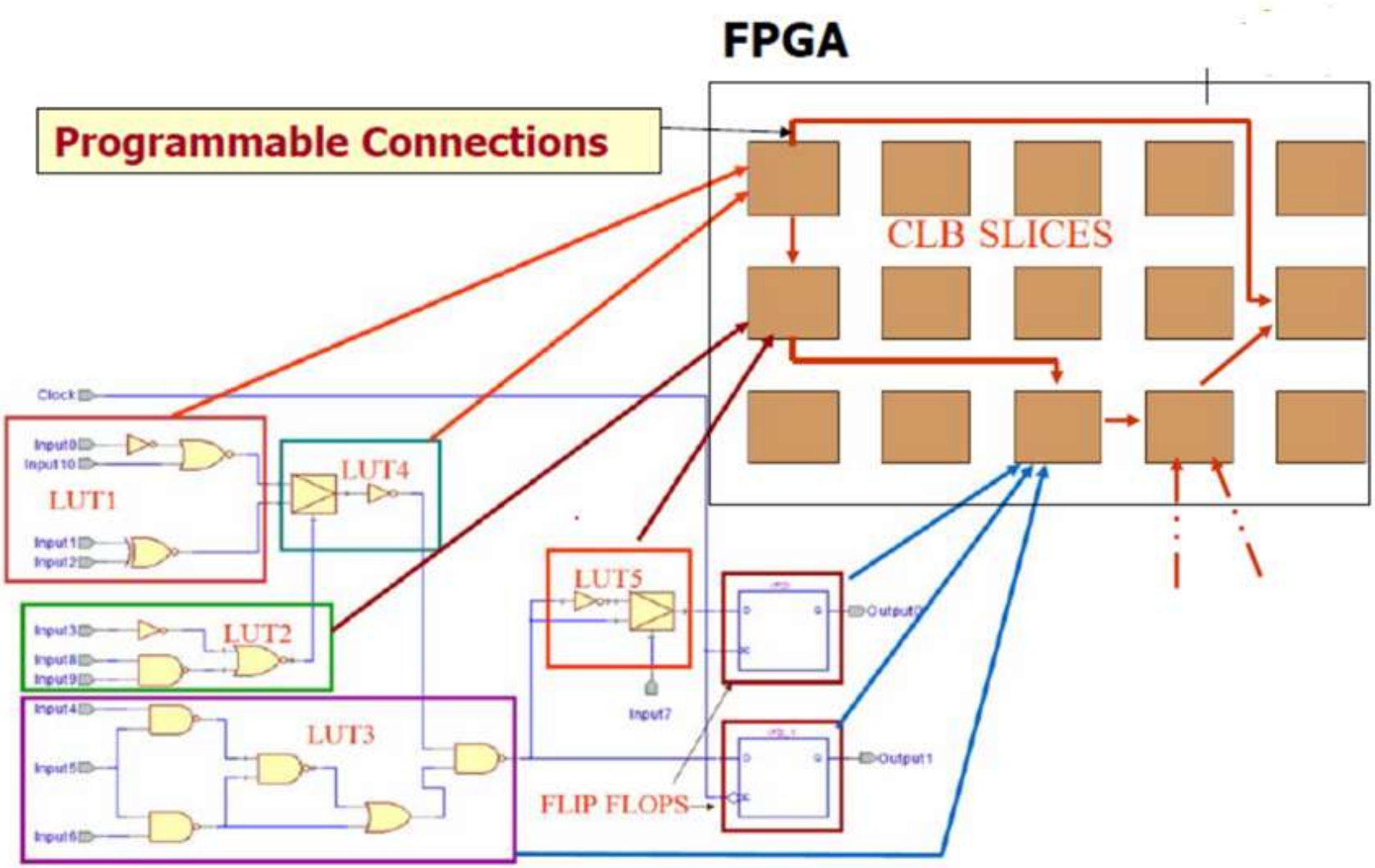
Map +Place



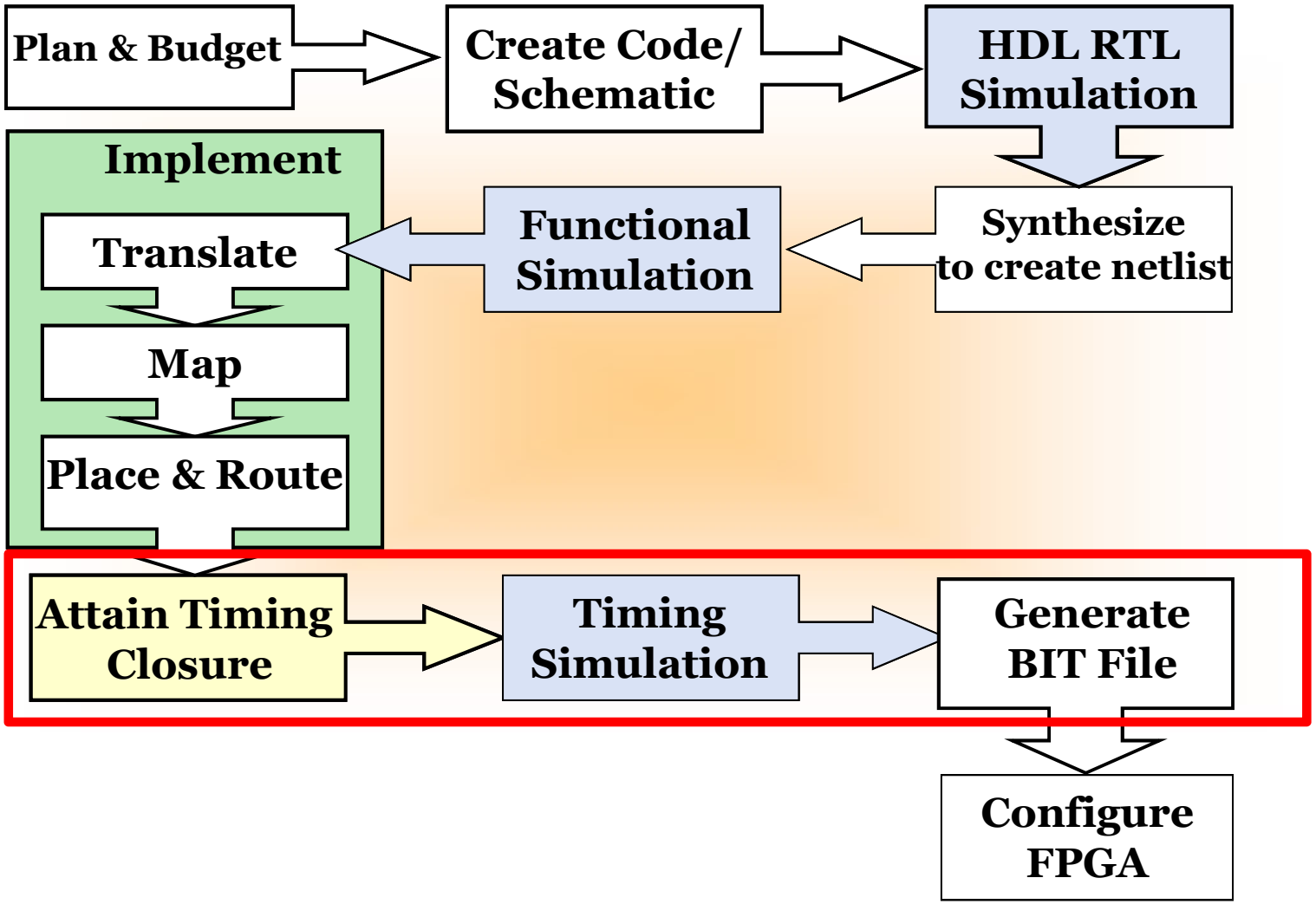
Map + Place



Route


FPGA
Programmable Connections
CLB SLICES
Clock
Input0
Input10
LUT1
Input1
Input2
LUT4
Input3
Input8
Input9
LUT2
Input4
Input5
Input6
LUT3
LUT5
Input7
Output0
Output1
FLIP FLOPS

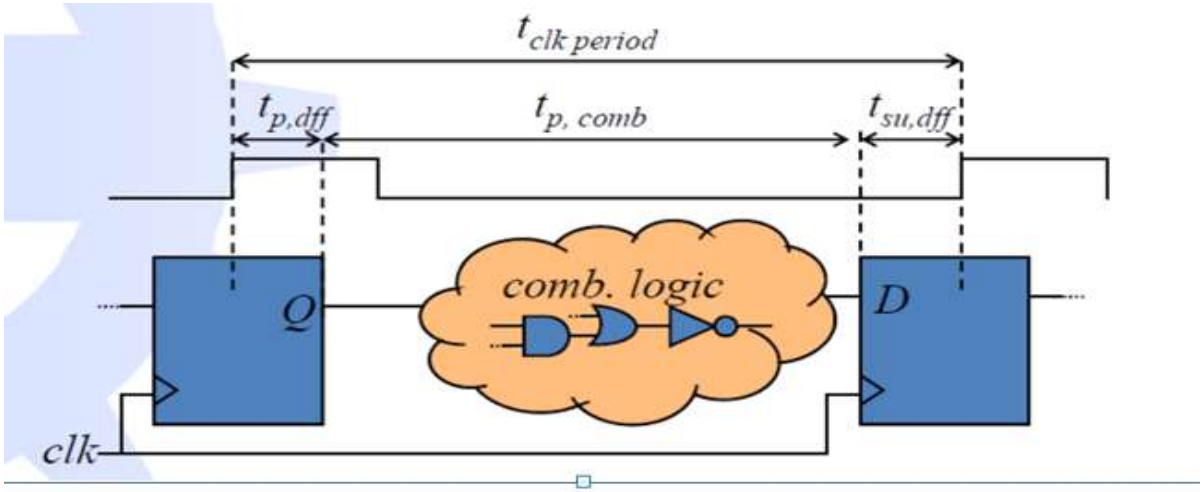
FPGA Design Flow

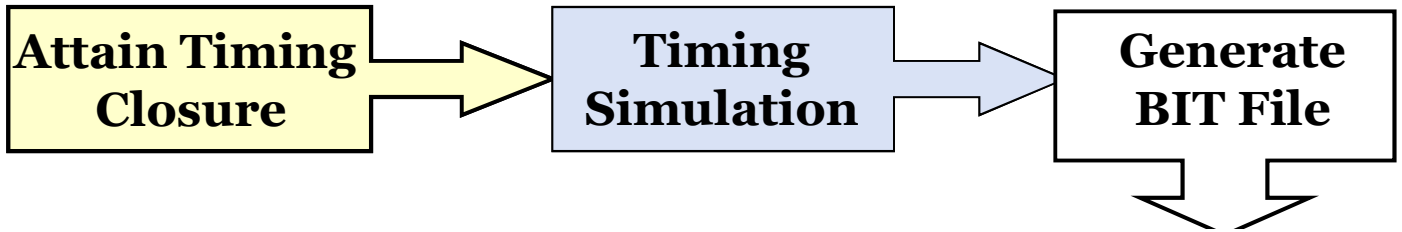


FPGA Design Flow – Final Steps

- Timing Closure: Obtain timing information of the design, timing reports, setup/hold violations, clock frequency adjustments…

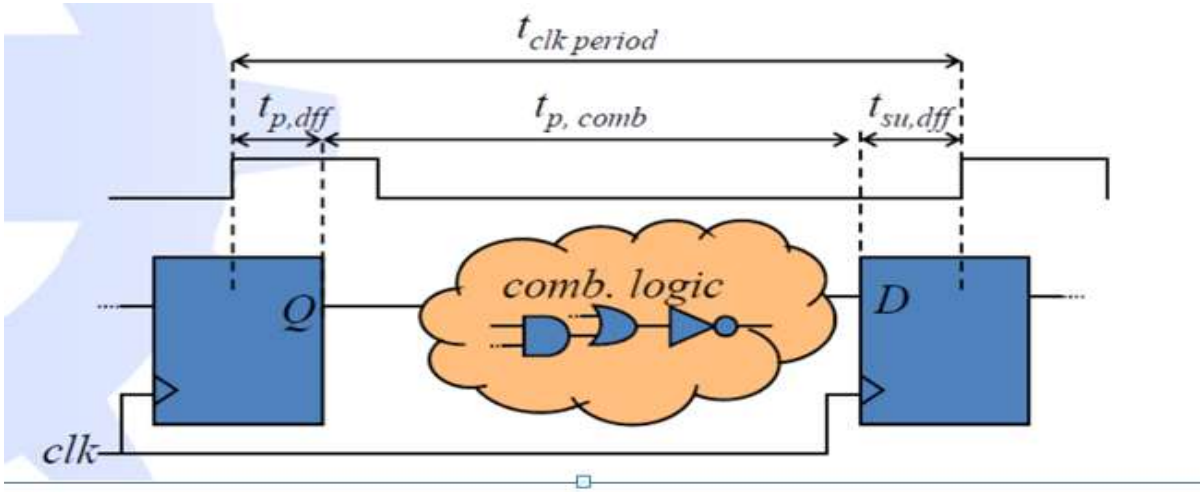
- Post Implementation Simulations: Simulating the placed&routed design; effects such as actual FPGA cell internal delays, interconnection delays, timing violations will be considered there

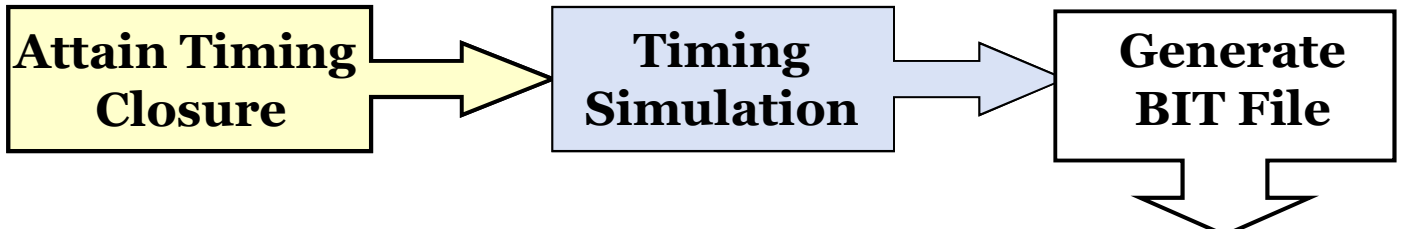




FPGA Design Flow – Final Steps

- Generate Bit File: Creates a large bitstream for each programmable location on FPGA, determines what value they will have. Completed connections will produce the described physical circuitry.





Configuration

- Once a design is implemented, you must create a file that the FPGA can understand
  - This file is called a bitstream: a BIT file (.bit file extension)
- The BIT file can be downloaded…
  - …Directly into the FPGA
    - Use a download cable such as Platform USB
  - …To external memory device such as a Xilinx Platform Flash PROM
    - Must first be converted into a PROM file