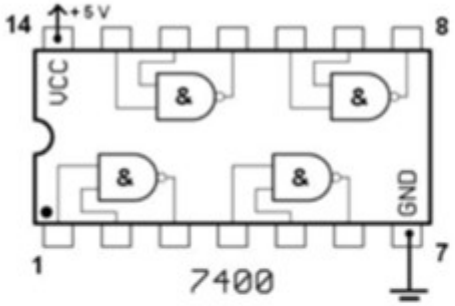
Field Programmable Gate Arrays (FPGA)

Digital CMOS Integrated Circuit Design

Y. Fırat Kula

Good old days…



- Discrete logic – based on gates or small packages containing small digital building blocks (at most a 1-bit adder)
- De Morgan's theorem – theoretically we only need 2-input NAND or NOR gates to build anything
- Tedious, expensive, slow, prone to wiring errors

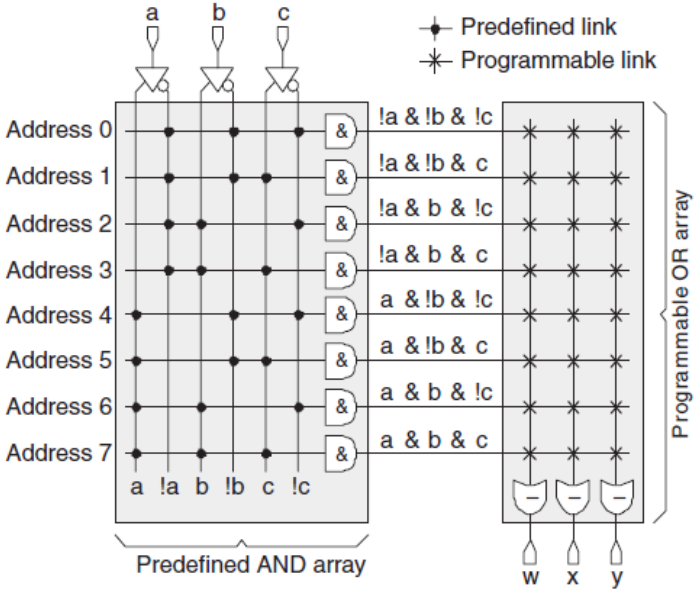
Programmable Logic

- Simple Programmable Logic Devices (SPLD)

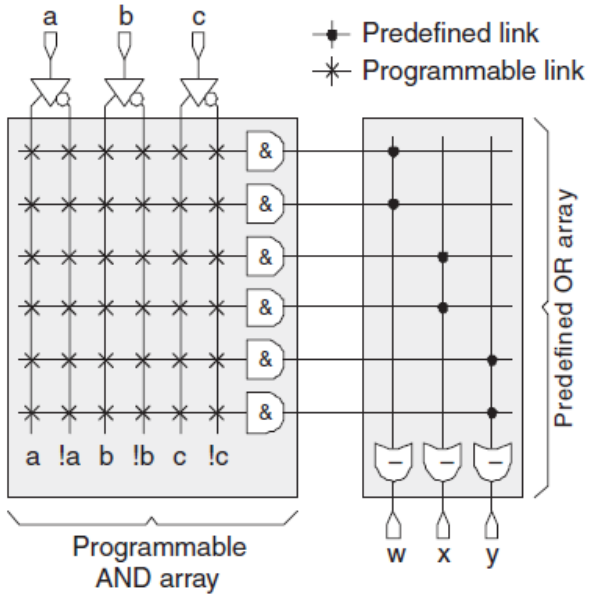
  ** PAL (Programmable Array Logic) , PLA (Programmable Logic Array)
- Complex PLD's (CPLD)

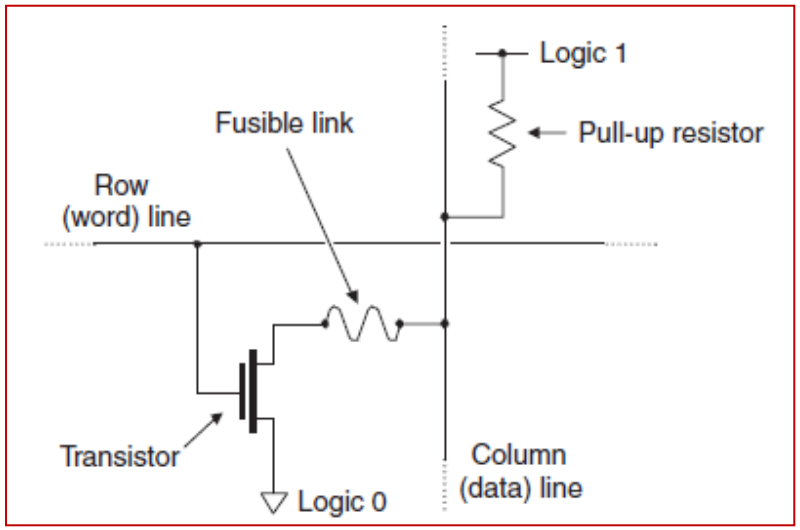
  ** Multiple SPLD type of blocks with interconnect
- Field Programmable Gate Array (FPGA)

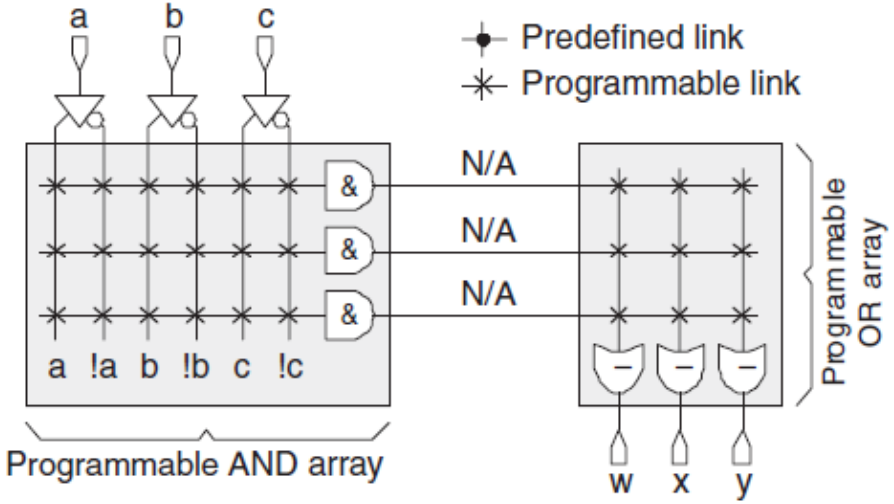
PROM



PLA (1975)

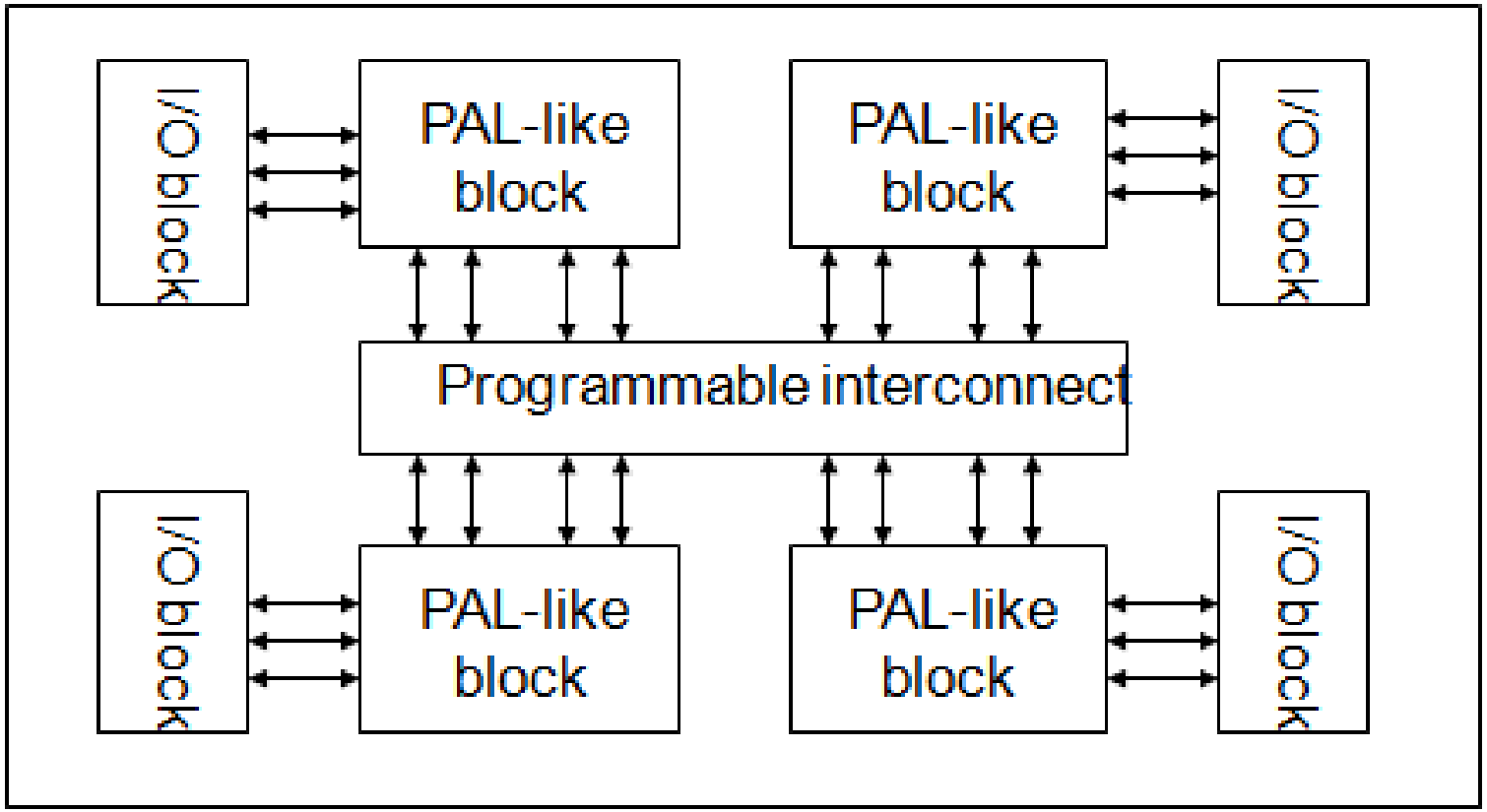




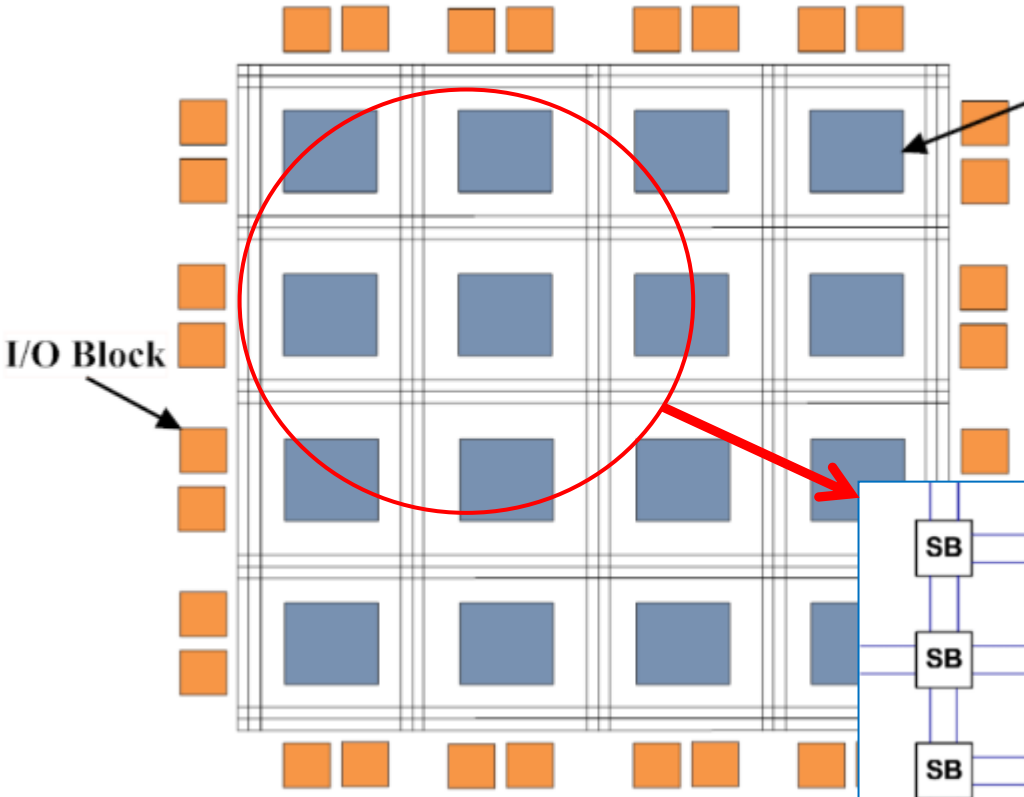


PAL (1978)

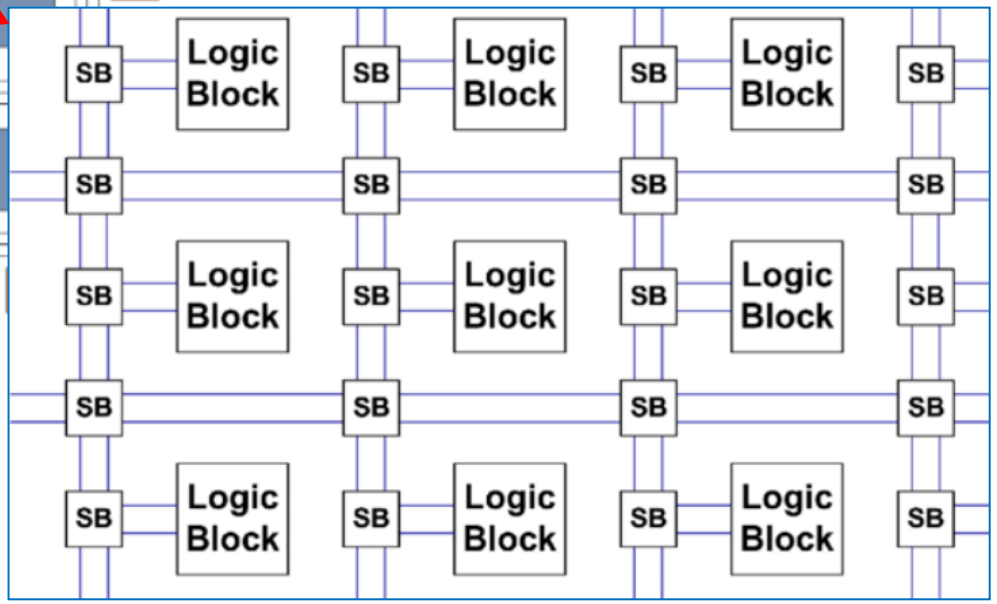
CPLD (1983)


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

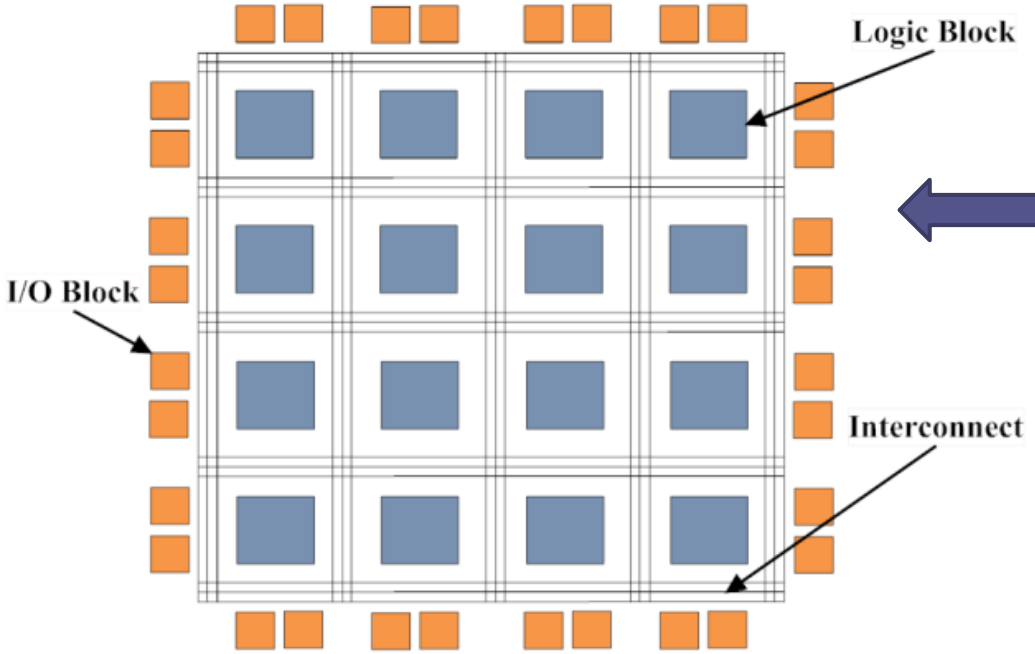
FPGA (1985)



Large Number of Logic Block 'Islands'
1,000 … 100,000+
in a 'Sea' of Interconnects

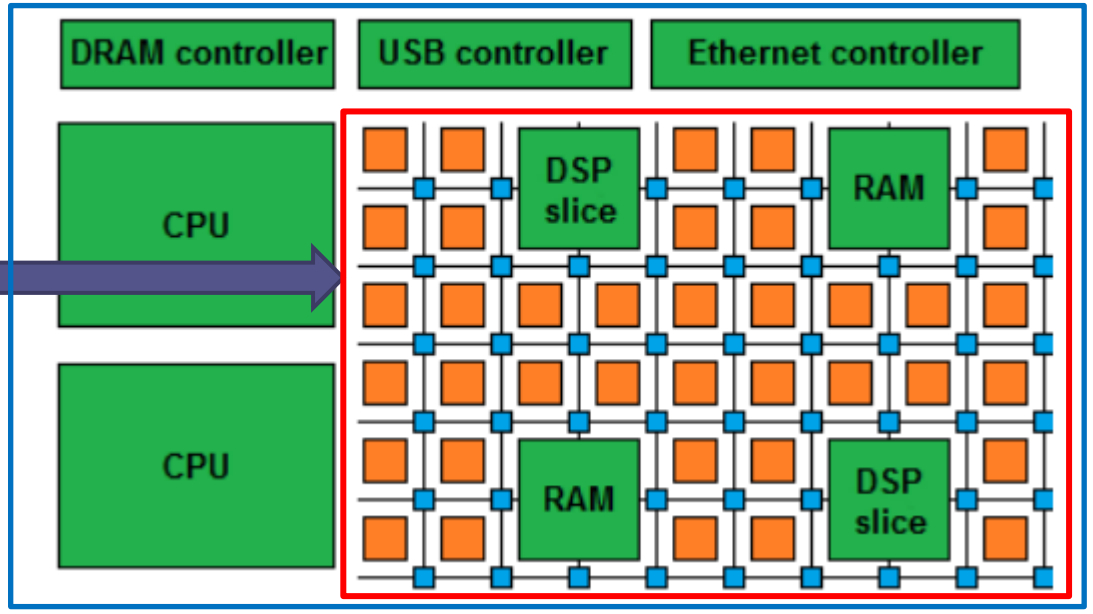


Newer designs may also contain:
DSP blcoks
Specific arithmetic blocks
RAM blocks

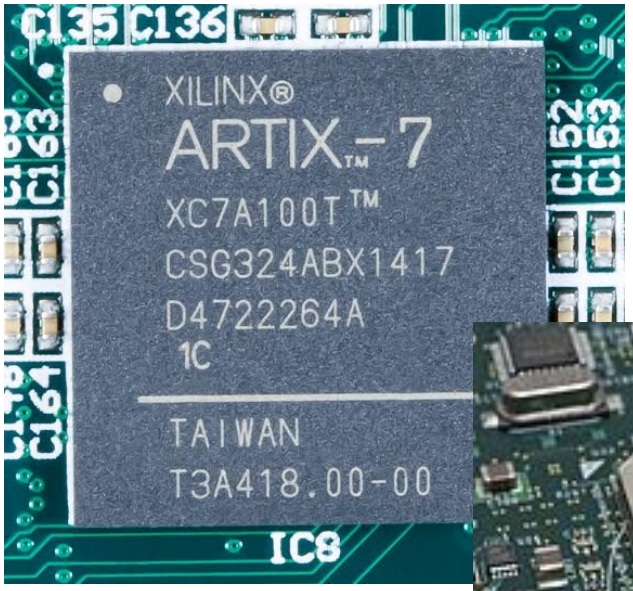


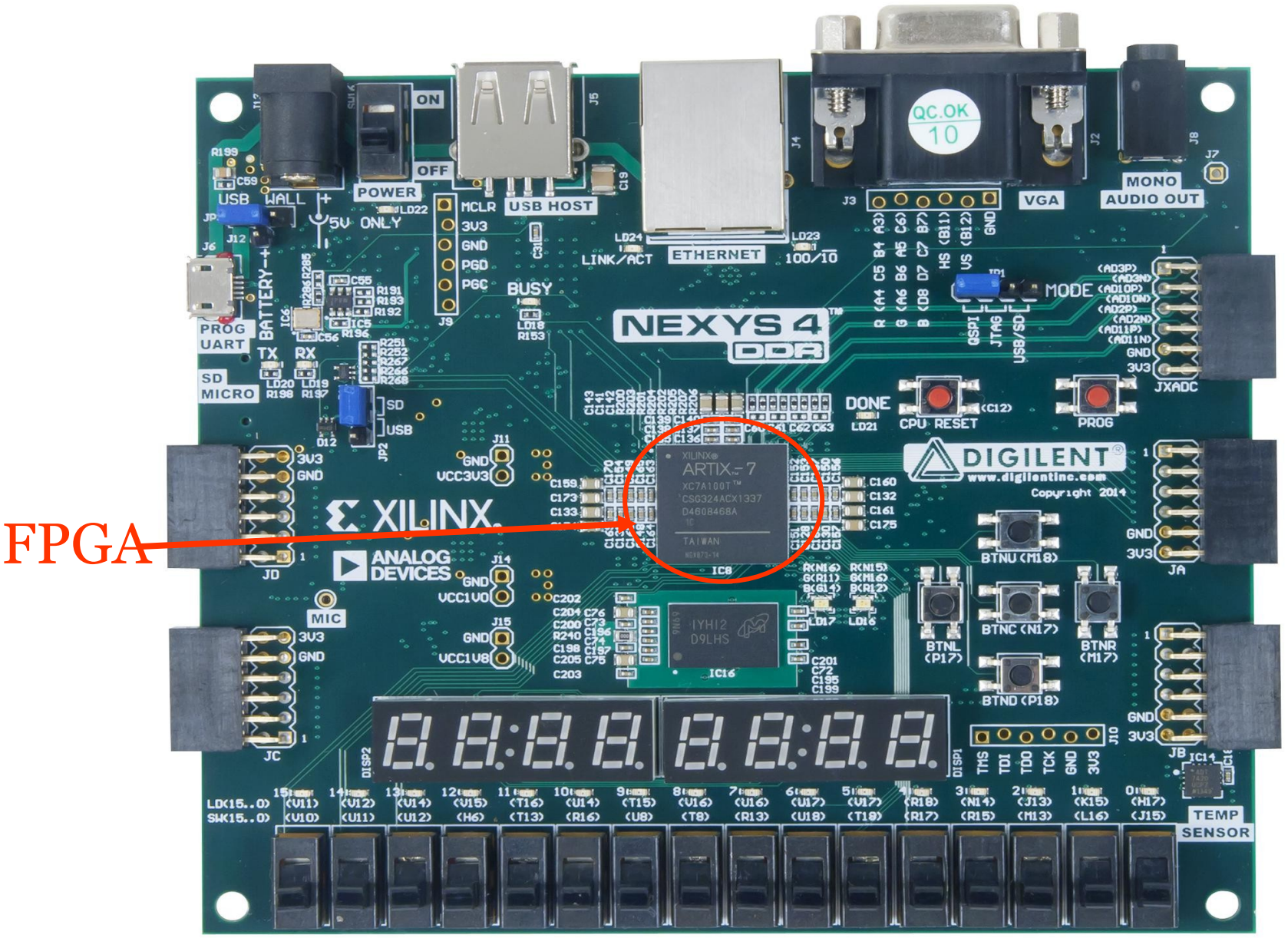
Classical Designs

Modern Designs



- Field programmable?




FPGA
NEXYS 4
DDR
XILINX
ARTIX-7
XC7A100T
DIGILENT
www.digilentinc.com
Copyright 2014
ANALOG DEVICES
POWER
ON
OFF
USB HOST
ETHERNET
VGA
MONO AUDIO OUT
PROG UART
SD MICRO
MODE
CPU RESET
PROG
DONE
BTNU (M18)
BTNC (N17)
BTNL (P17)
BTNR (M17)
BTND (P18)
MIC
TEMP SENSOR
JXADC
JA
JB
JC
JD

Areas of Usage

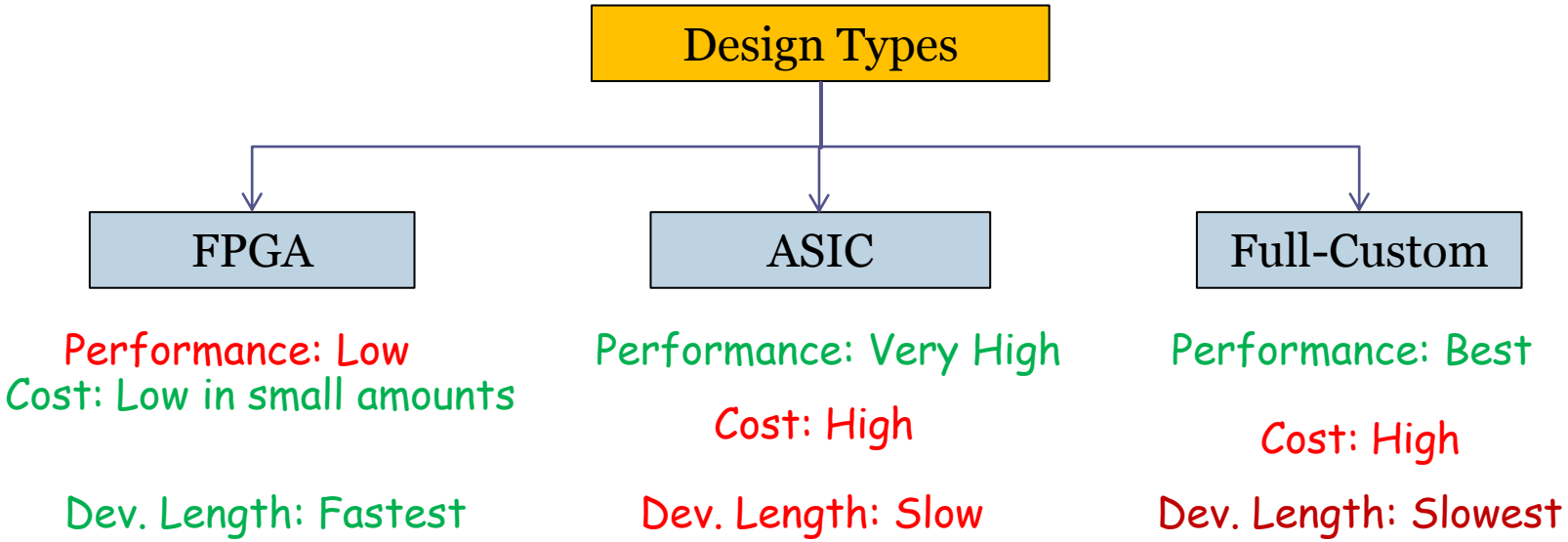
- Digital signal processing, communications, aerospace, cryptography…
- Hardware acceleration (Parallel computing)
- Design prototyping
- Can be substitute to ASIC when a design needs to be quickly produced in small quantities.

Pros & Cons

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

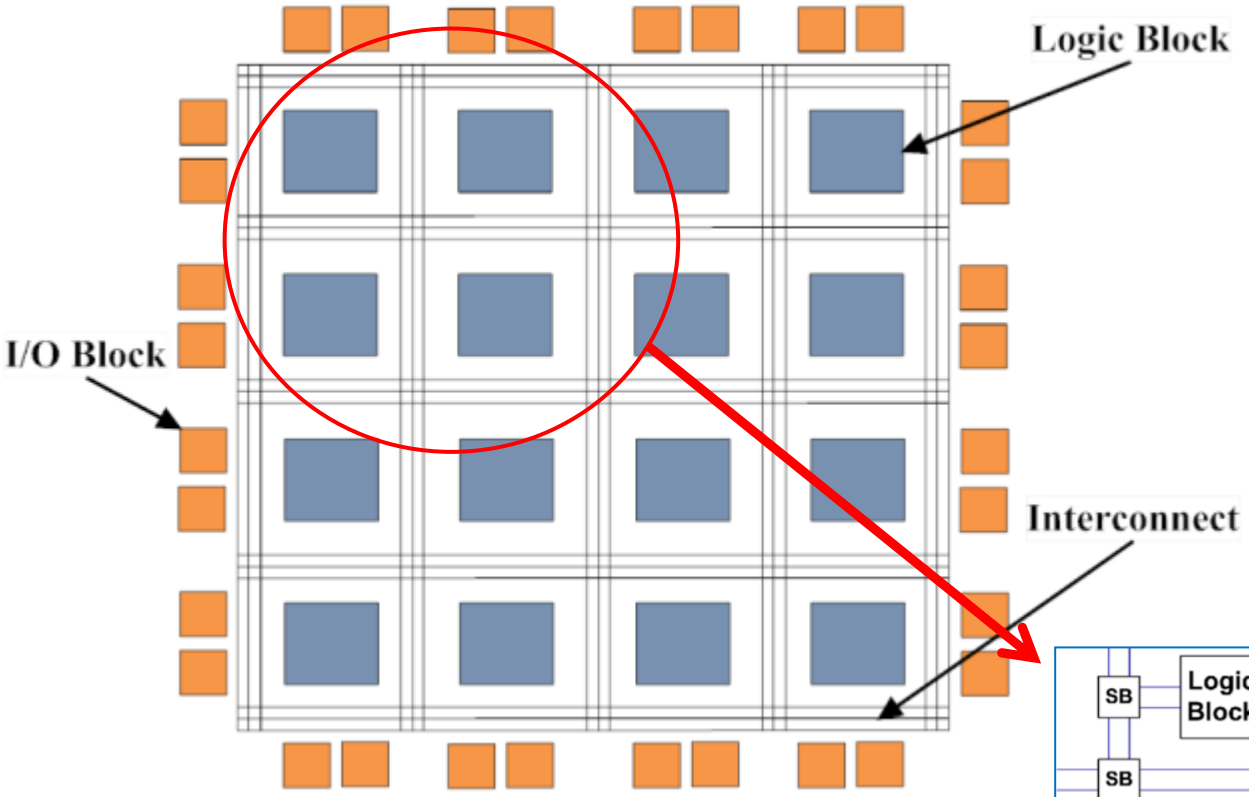
– Performance-wise, it is not as good as ASIC (Application Specific Integrated Circuits).
–Not suitable for mass marketing, since the high cost of the single FPGA makes it impractical compared to the much cheaper mass producing costs of ASIC

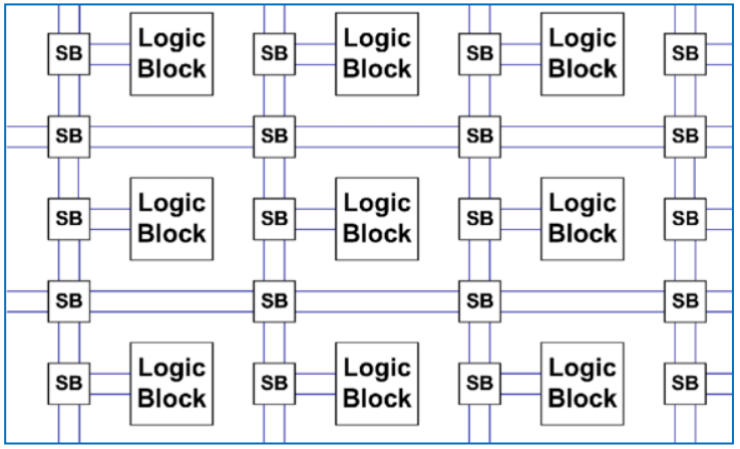
Different approaches…



- 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

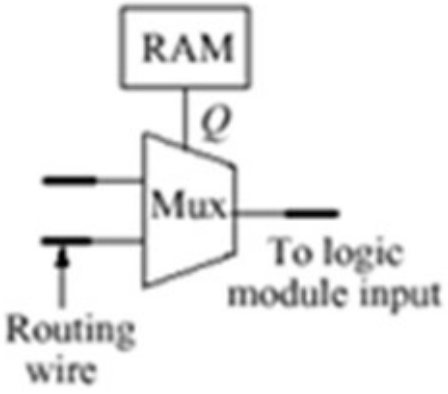
Inside of an FPGA?

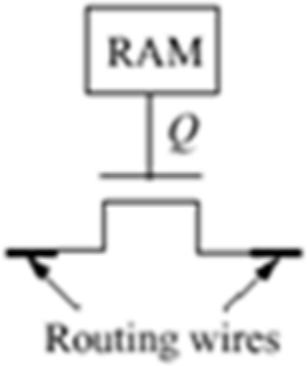


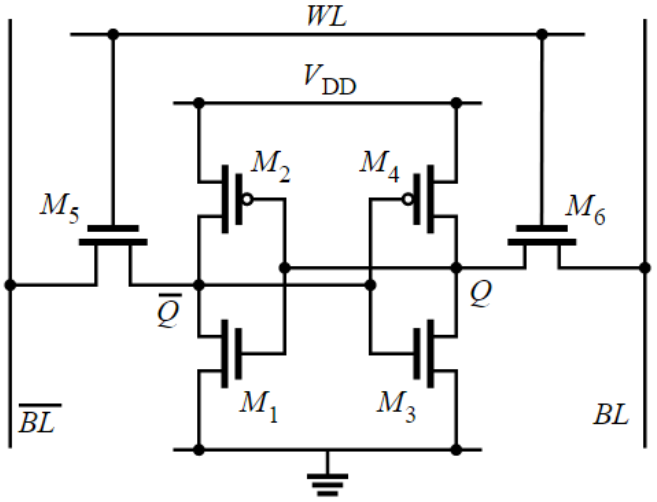


Programming uses small RAM cells…

- 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 …

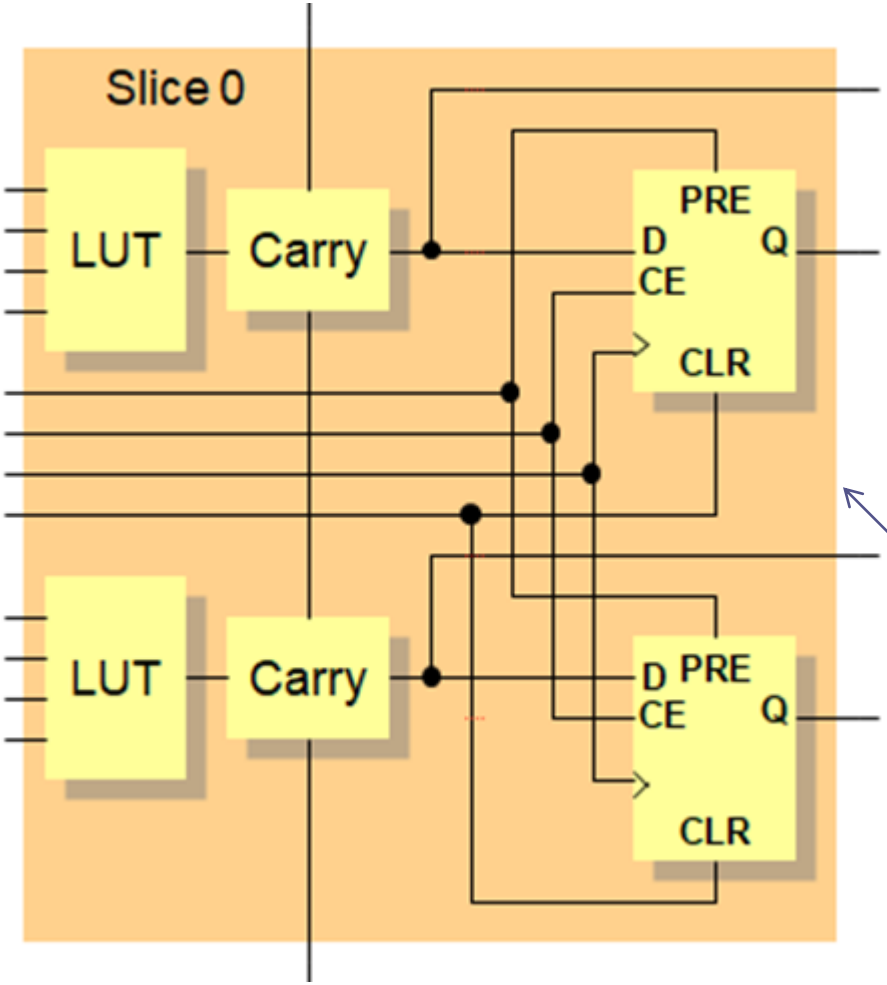


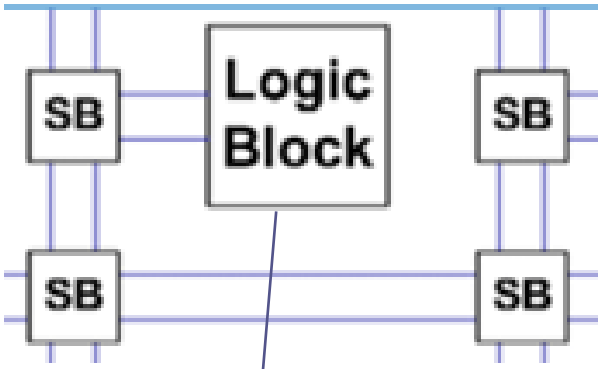




6-transistor SRAM circuit

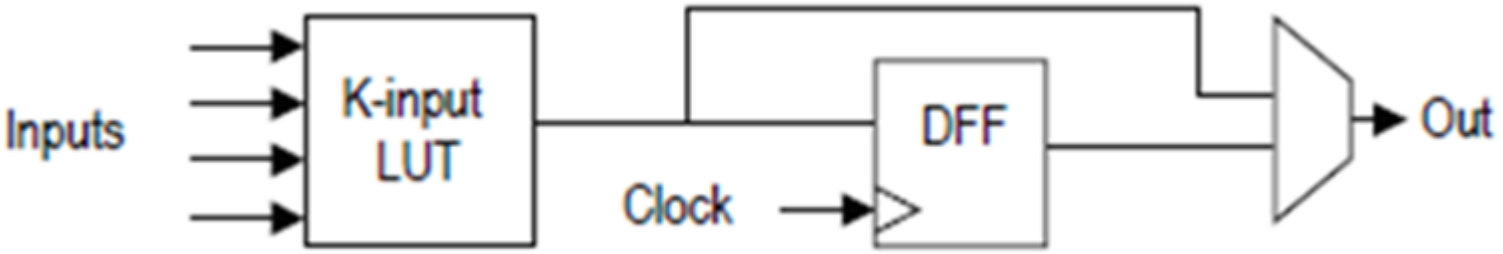
Look-up Table (LUT) based Logic Blocks

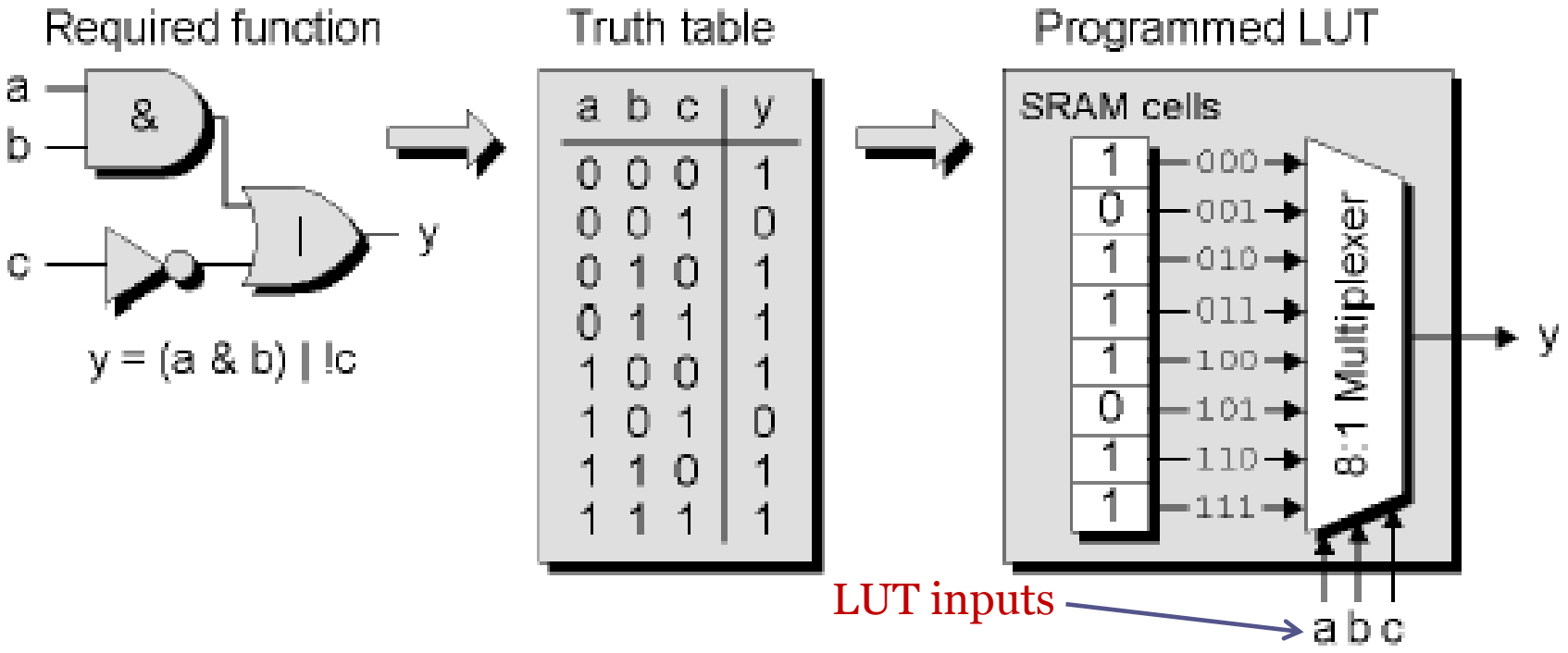




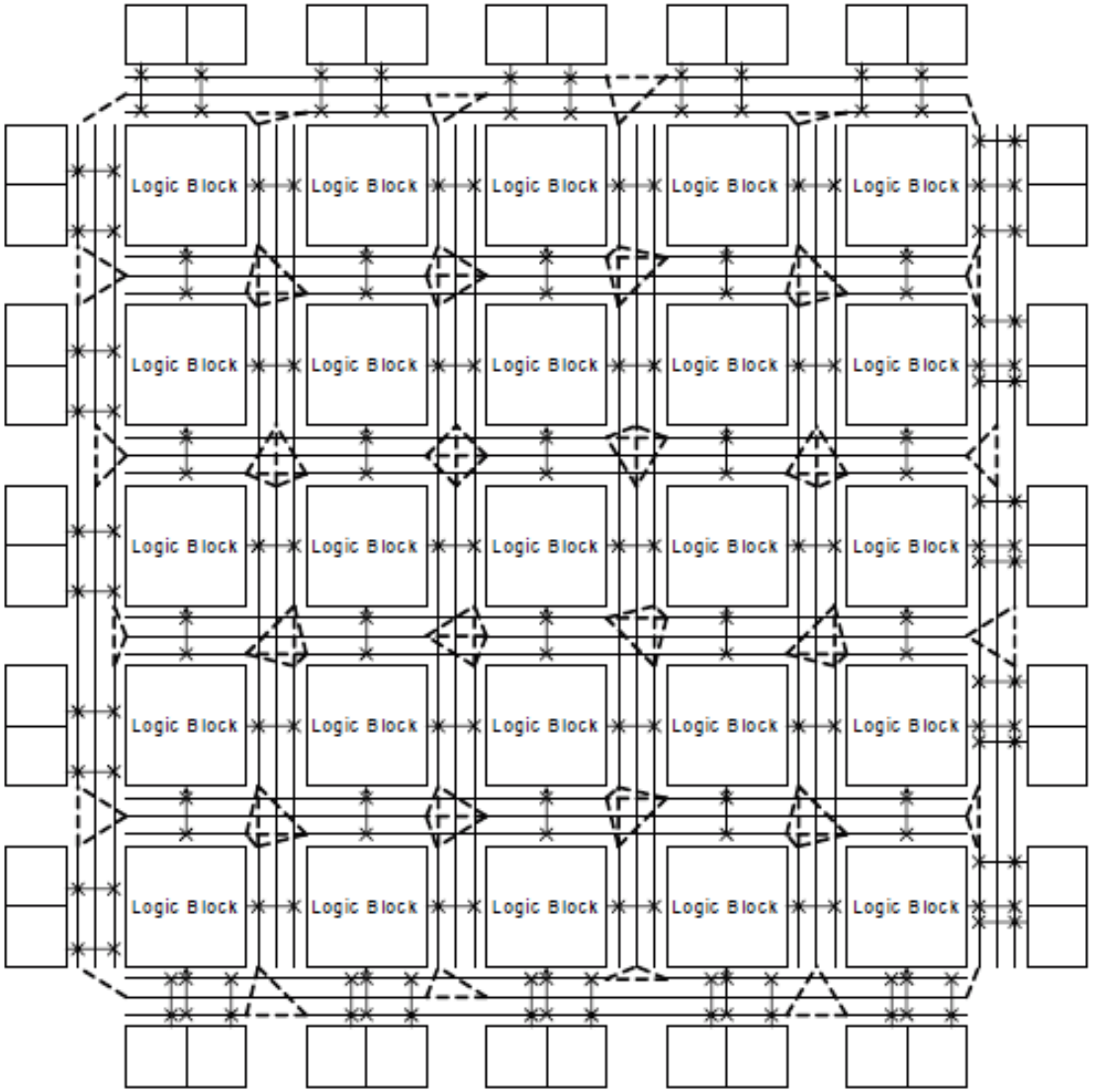
Logic blocks in Xilinx devices consists of multiple **slices**

Look-up Table (LUT) based Logic Blocks

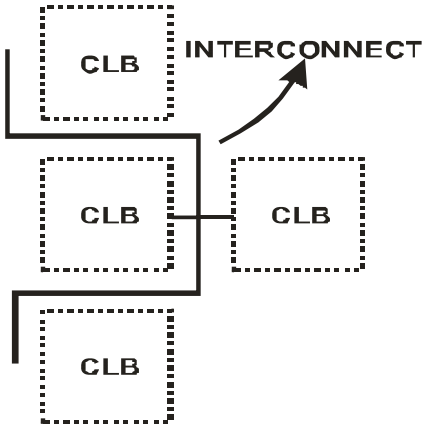


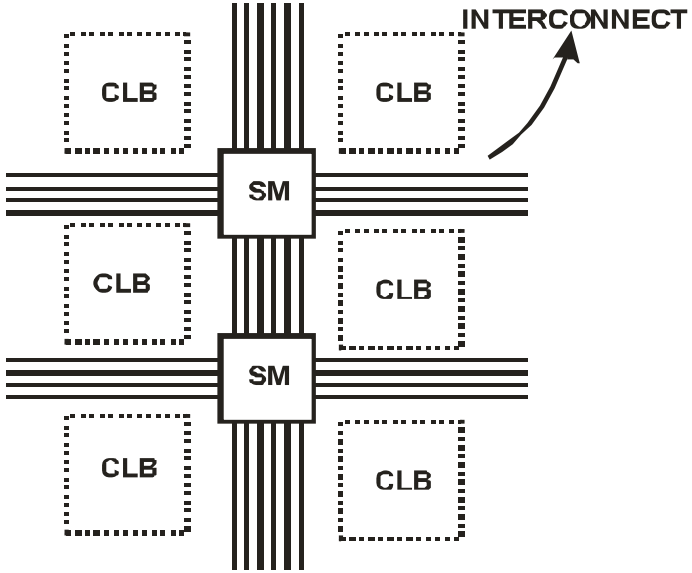


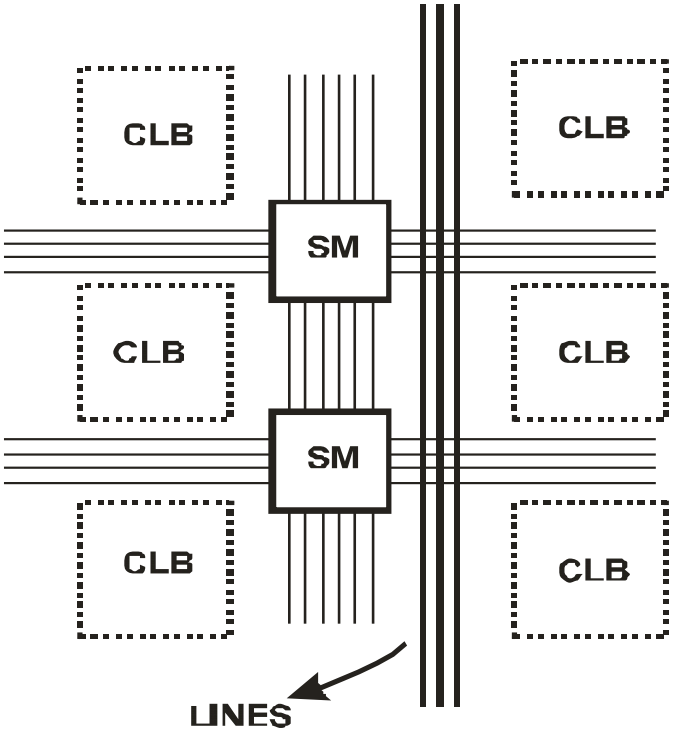
Interconnections



Interconnections

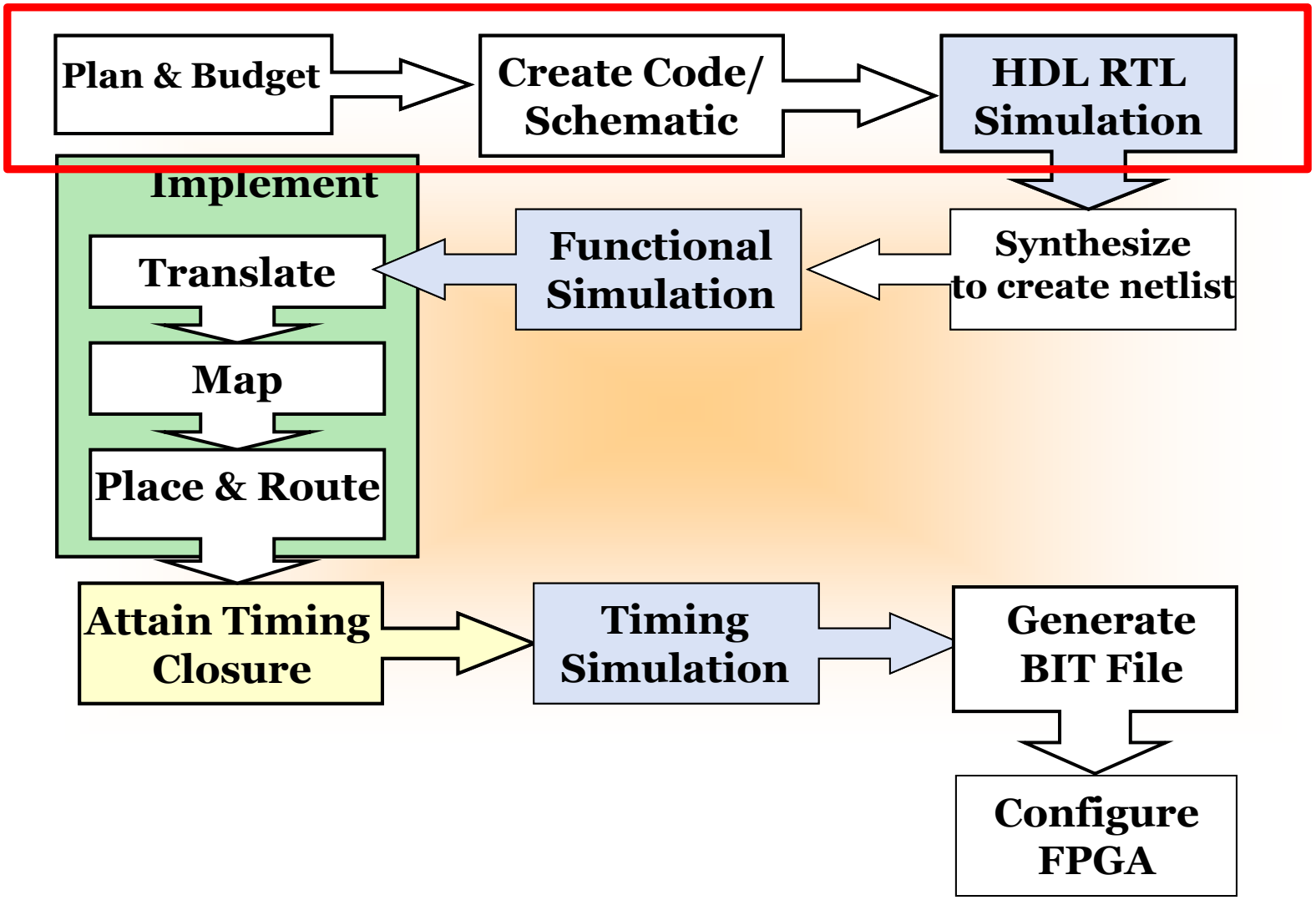






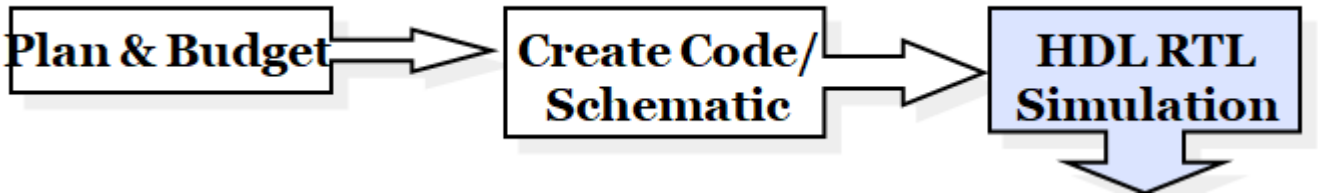
Interconnection switches are also programmed with RAM cells

FPGA Design Flow

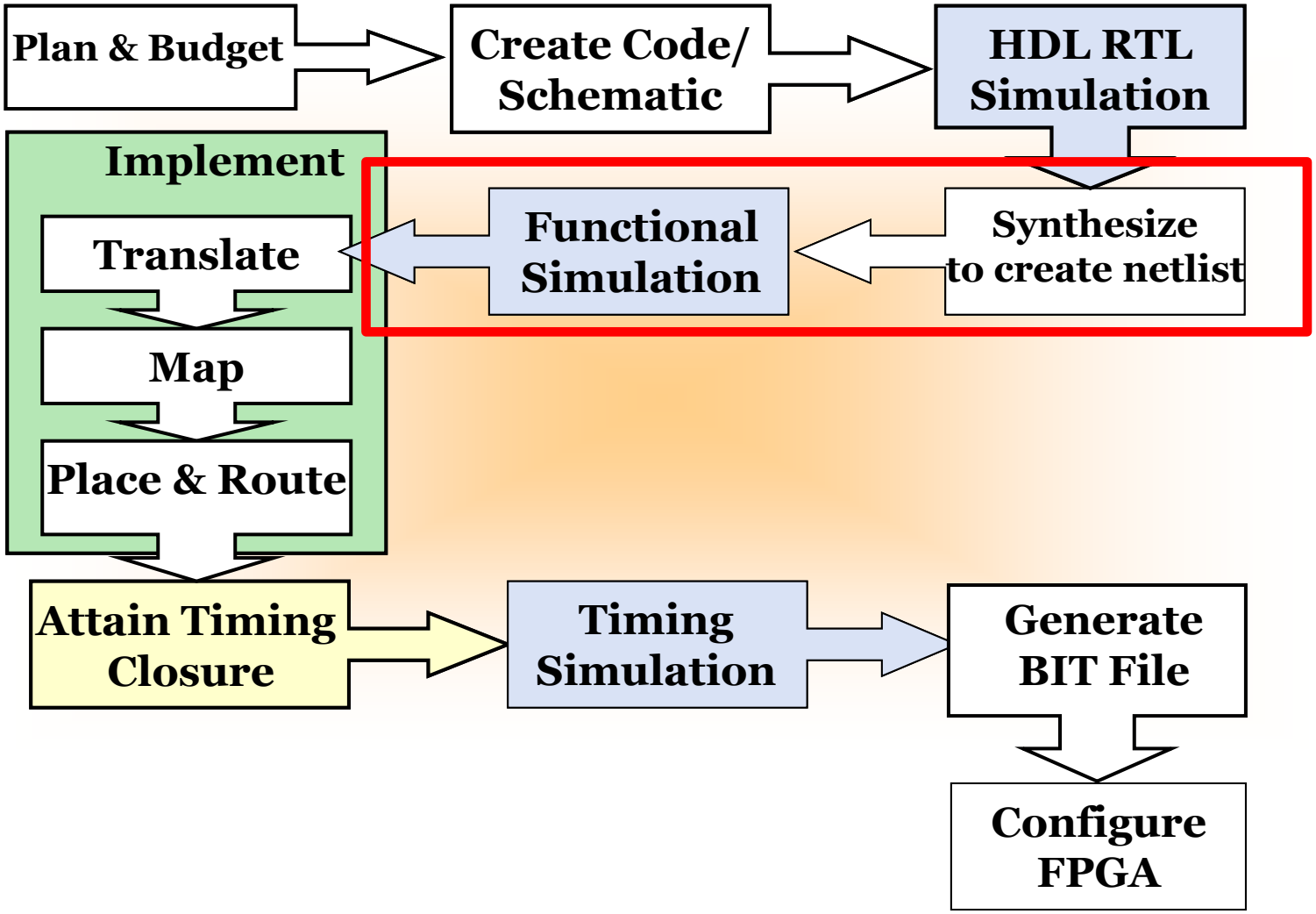


FPGA Design Flow – Design Entry

- Plan and budget: Determine a suitable device for our design
  (Number of logic blocks, arithmetic blocks, RAM blocks, etc..)
- Code your design: Hardware Description Languages (HDL), *Ex: VHDL, Verilog*
- Simulate the design to ensure that it works as expected!



FPGA Design Flow

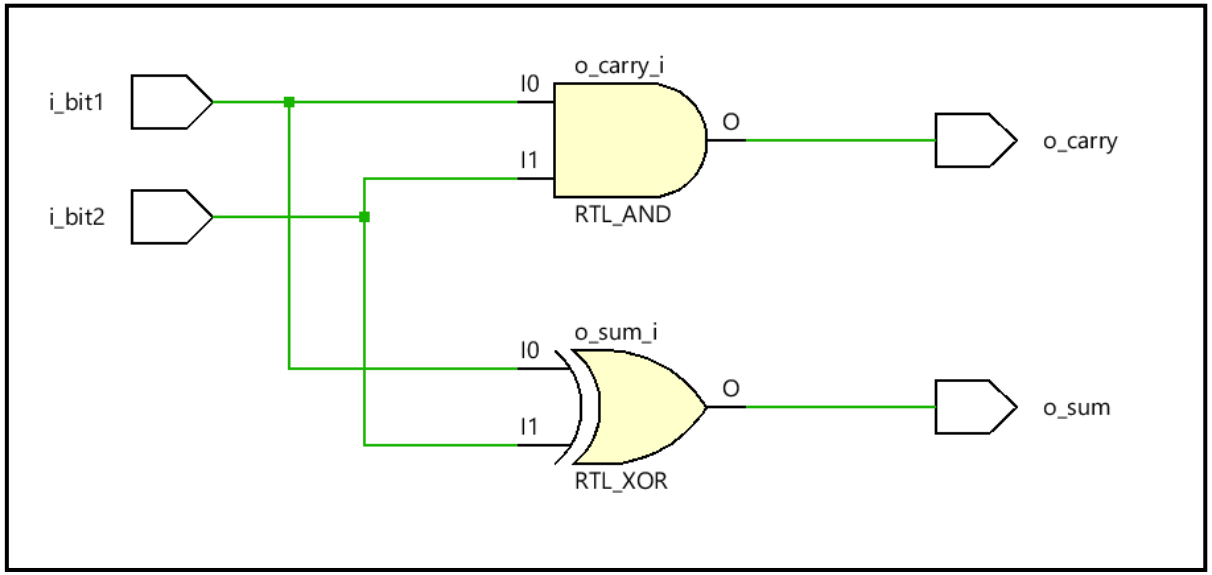


FPGA Design Flow – Synthesis

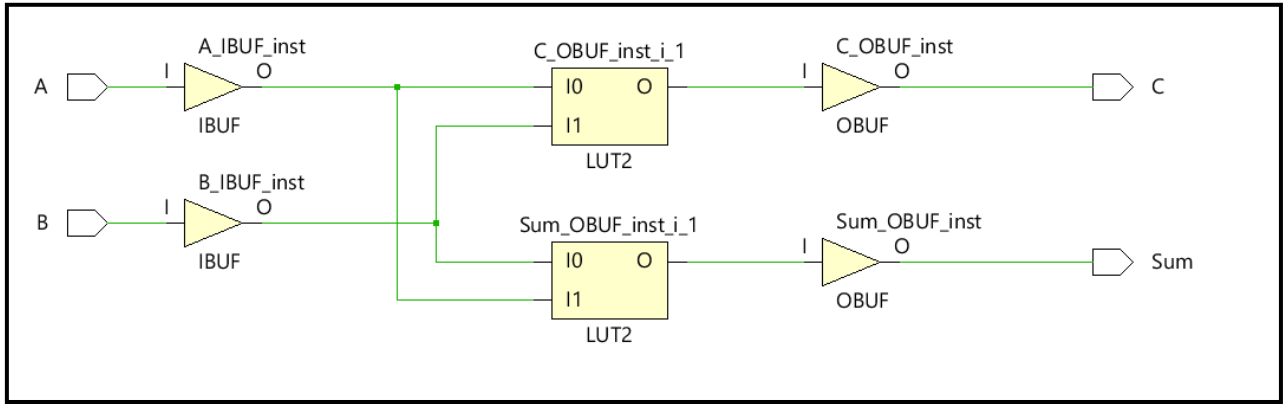
- Synthesis means converting our HDL coded design to logic element level netlist
- You will need a tool to generate an EDIF or NGC netlist to bring into the vendor specific implementation tools
  - Popular synthesis tools include: XST , Synplify, Precision, FPGA Compiler II
- Synthesised netlist can be simulated to see if it shows the desired fucntionality (Post Synthesis Functional 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

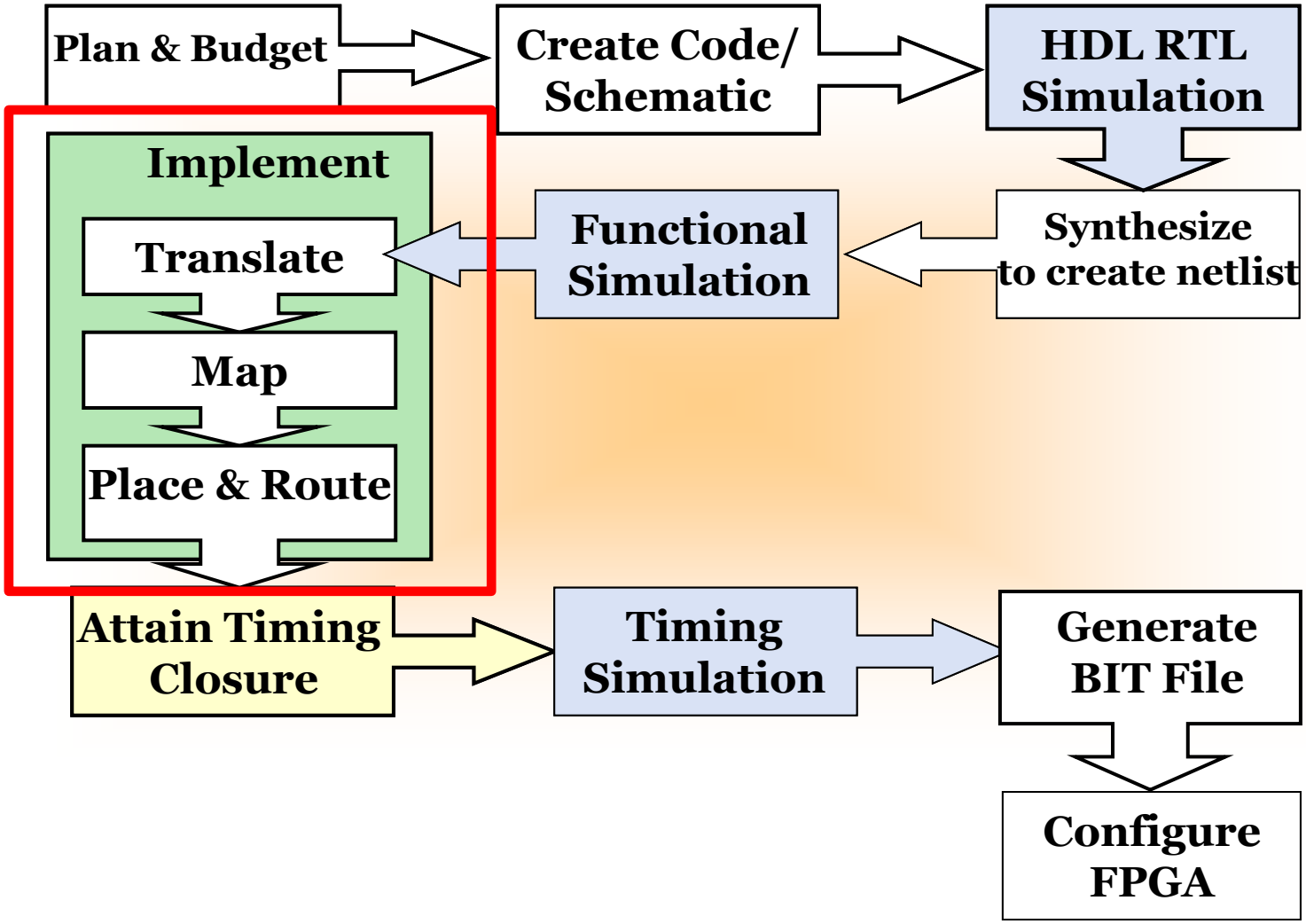


RTL based netlist schematic



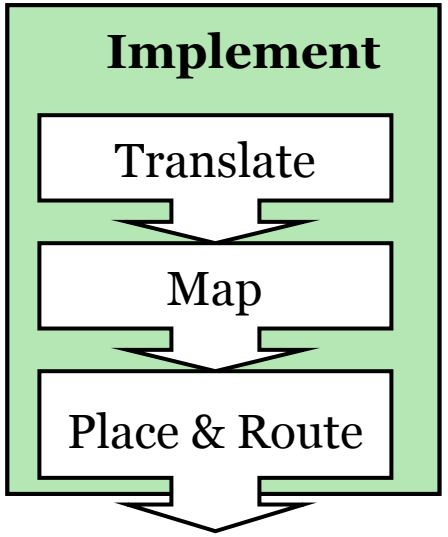
Technology schematic

FPGA Design Flow

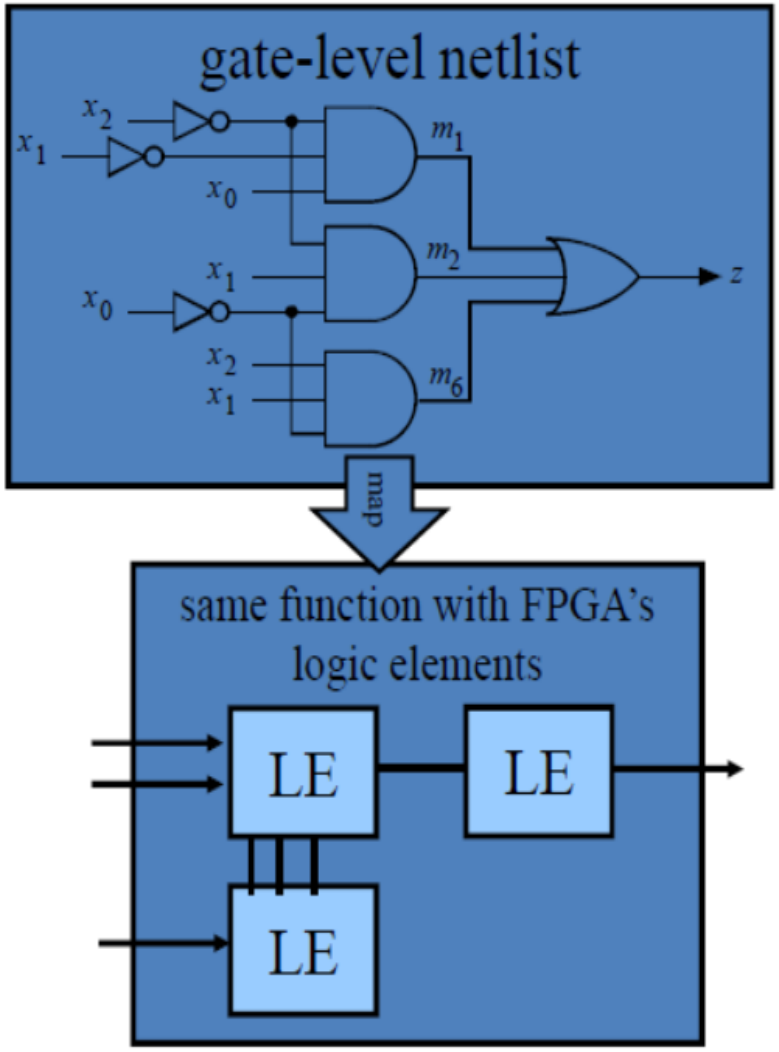


FPGA Design Flow - Implementation

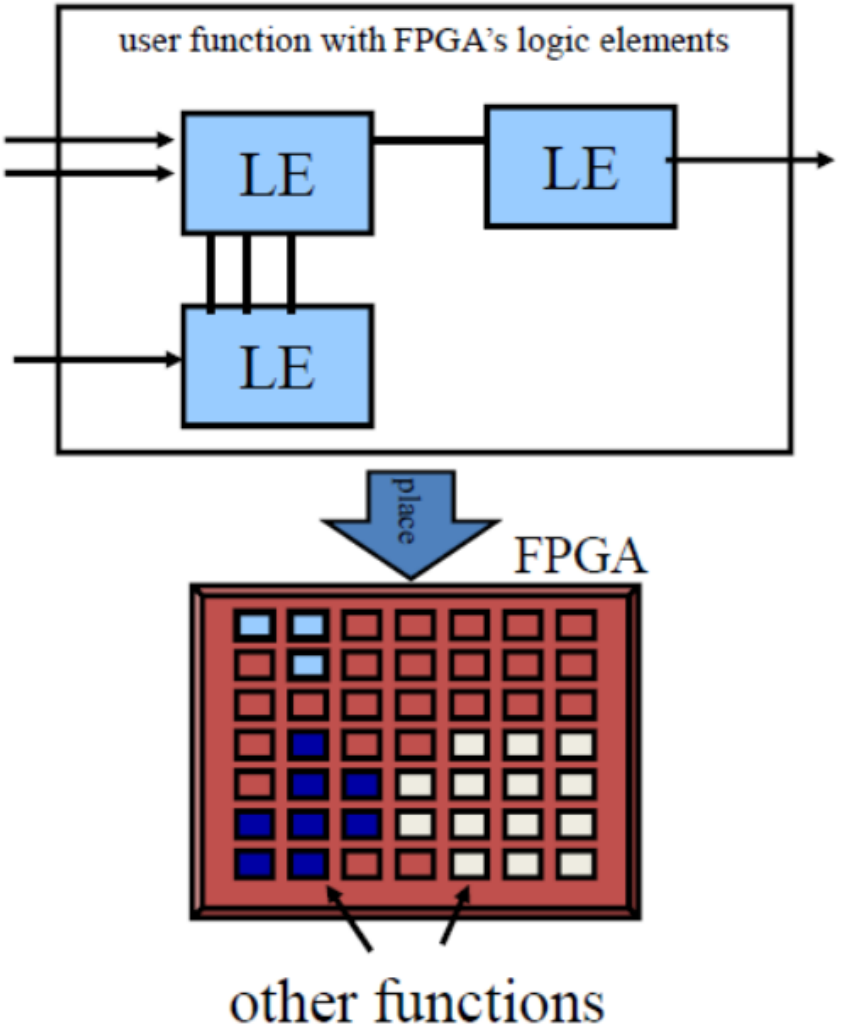
Translate: Merge all netlist files and express in terms of FPGA primitives and set it ready to be mapped
Map: Choose which device specific primitives will be used in place of translated design blocks
Place & Route: Place the design on the target device and make interconnections



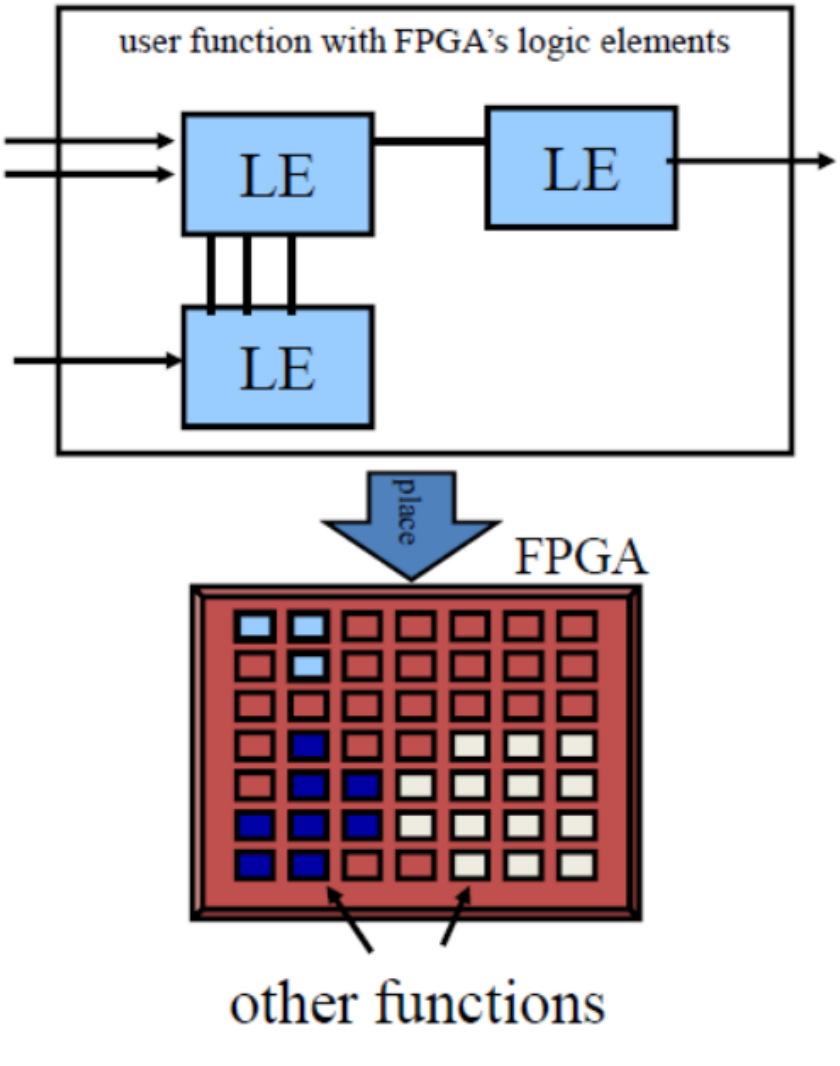
We need to use vendor specific implementation tools and device libraries.



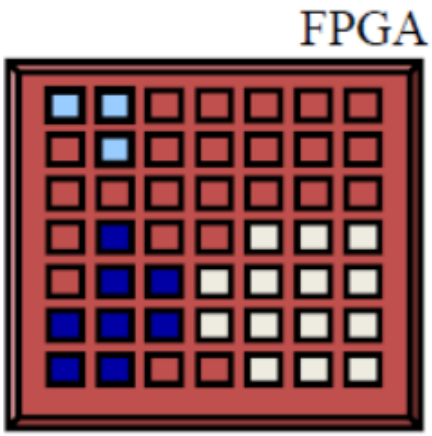
Translate and Map

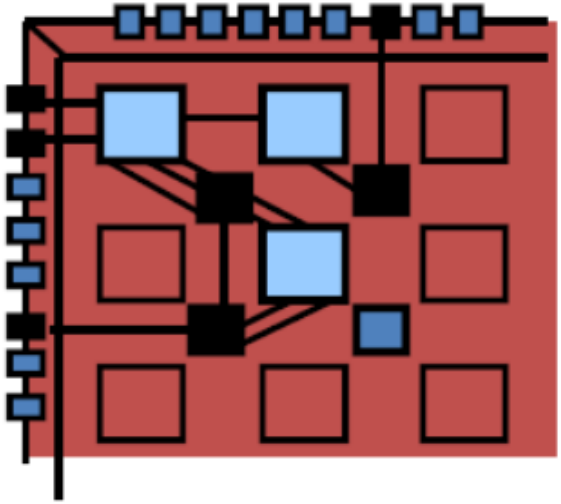


Place

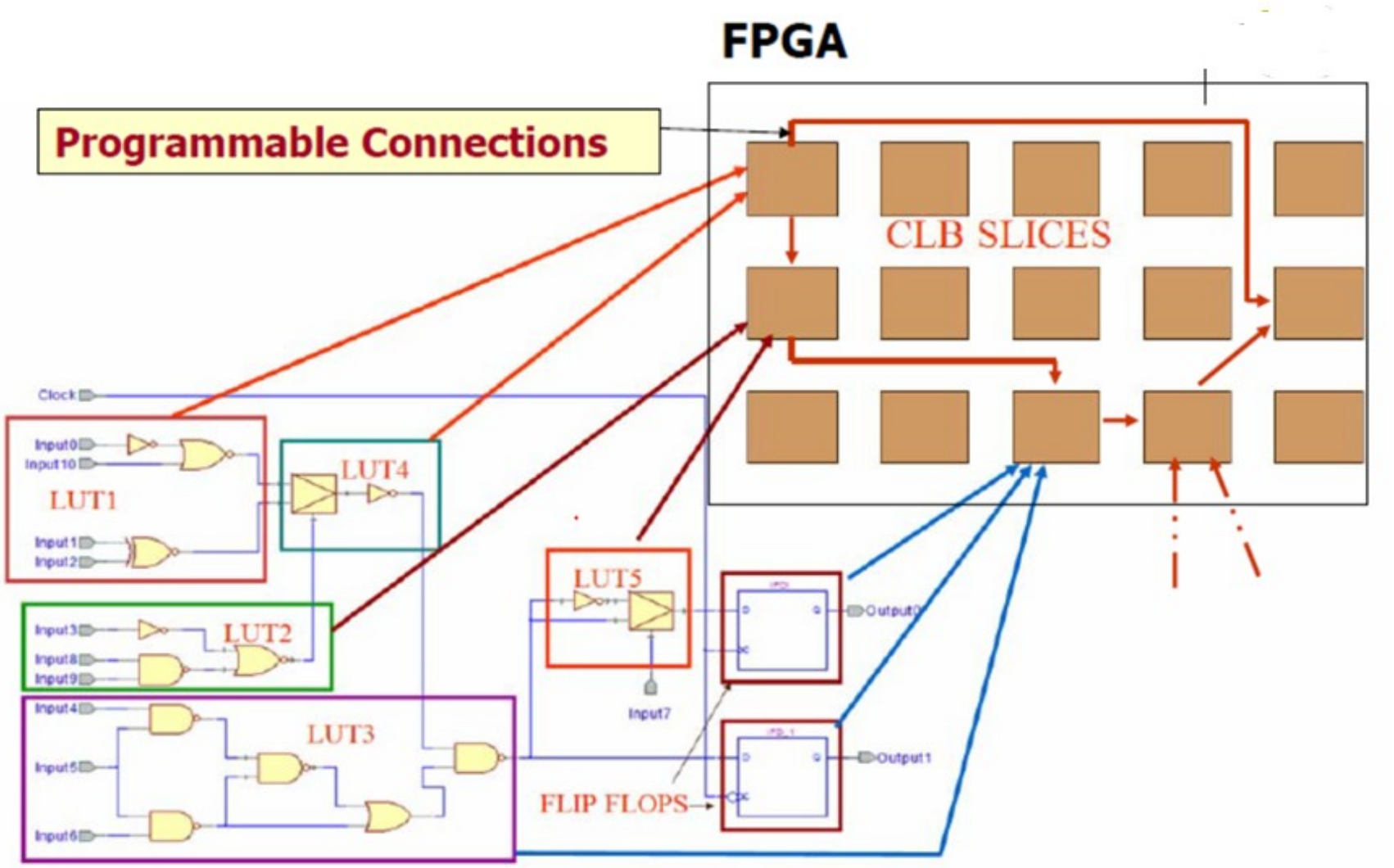

user function with FPGA's logic elements
LE
LE
LE
place
FPGA
other functions

Place

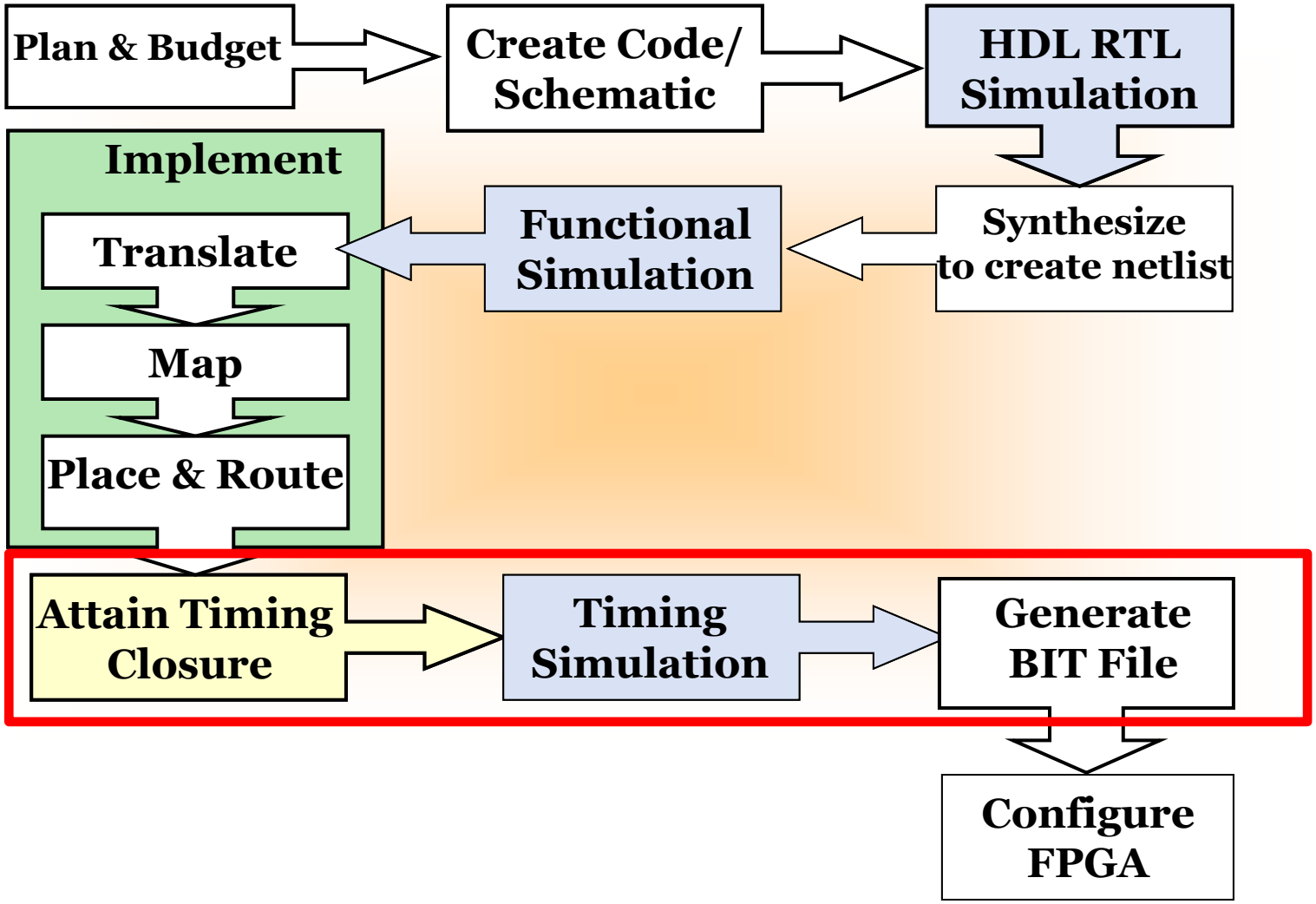

FPGA



Route

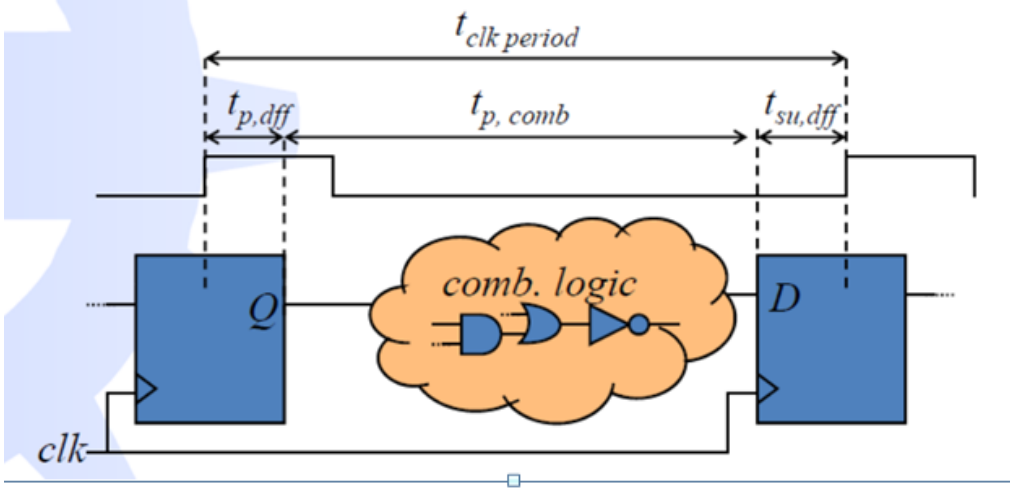

FPGA
Programmable Connections
CLB SLICES
Clock
Input0
Input10
LUT1
Input1
Input2
LUT4
Input3
Input8
Input9
LUT2
Input4
Input5
Input6
LUT3
LUT5
Input7
FLIP FLOPS
Output0
Output1

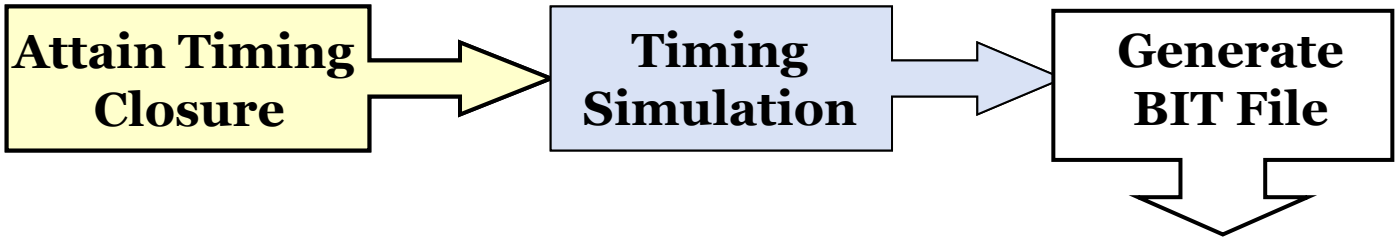
FPGA Design Flow



FPGA Design Flow – Final Steps

- Attain Timing Closure: Obtain critical path, timing reports…
- Post Implementation Simulation: Check if design works correctly on actual hardware, time delays are considered here.
- 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