VietNam National University
University of Engineering and Technology

THIẾT KẾ MẠCH TÍCH HỢP SỐ
(DIGITAL IC DESIGN)

TS. Nguyễn Kiêm Hùng
Email: kiemhung@vnu.edu.vn

Laboratory for Smart Integrated Systems

Lecture 2: Field-Programmable Logic Devices

Objectives

In this lecture you will be introduced to:

- **The programmable logic Technology,**
- **The features of FPGA architecture**

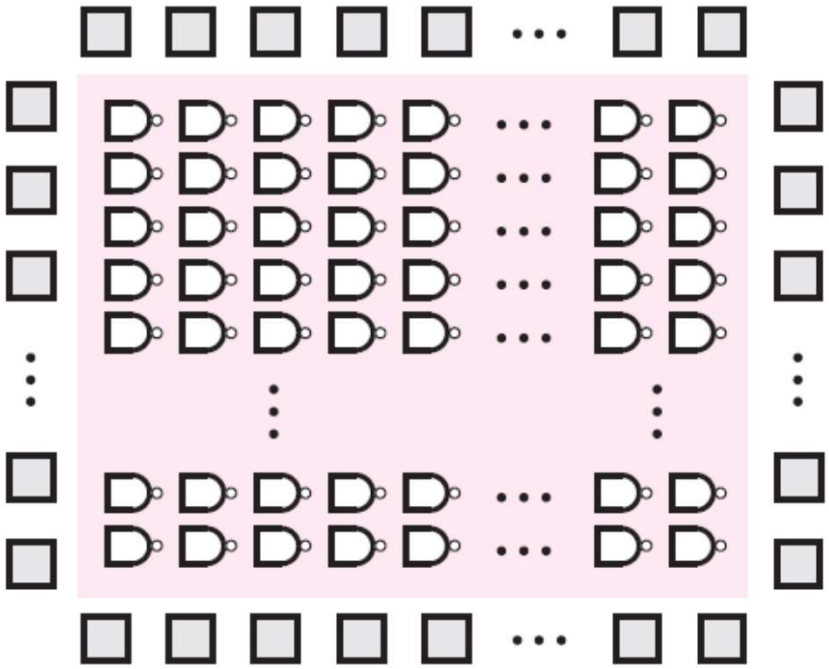
Review

- **Existing Integrated Circuits (ICs) can be classified into (1):**
  - **Standard ICs:**
    - **realize some commonly used logic circuits**
    - conform to **an agreed-upon standard** in terms of **functionality and physical configuration**
    - For example:
      - 7400-series, etc.
      - Memories, microcontroller, microprocessors, etc.

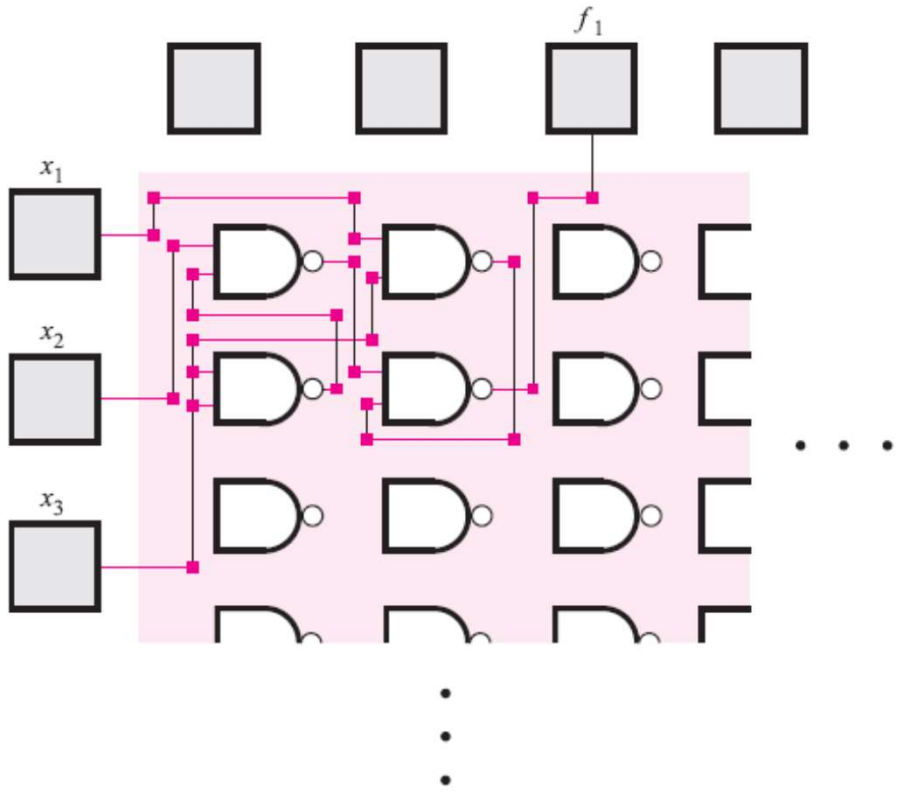
Review

- **Existing Integrated Circuits (ICs) can be classified into (2):**
  - **Programmable Logic Devices (PLD):**
    - Contain a **regular structure** and **a collection of programmable switches** that allow the internal circuitry in the chip to be **configured by the user** to implement a wide range of different logic circuits.
    - **Mask-programmable PLDs** and **Field-programmable PLDs**.
    - **Field-programmable PLDs** classified into:
      - Programmable Logic Array (PLA): both the AND and OR planes are programmable.
      - Programmable Array Logic (PAL): programmable AND plane, the is fixed OR plane.
      - **Field Programmable Gate Array (FPGA)**
    - **Field-programmable PLDs can be programmed multiple times.**

Example of Mask-Programmable PLD

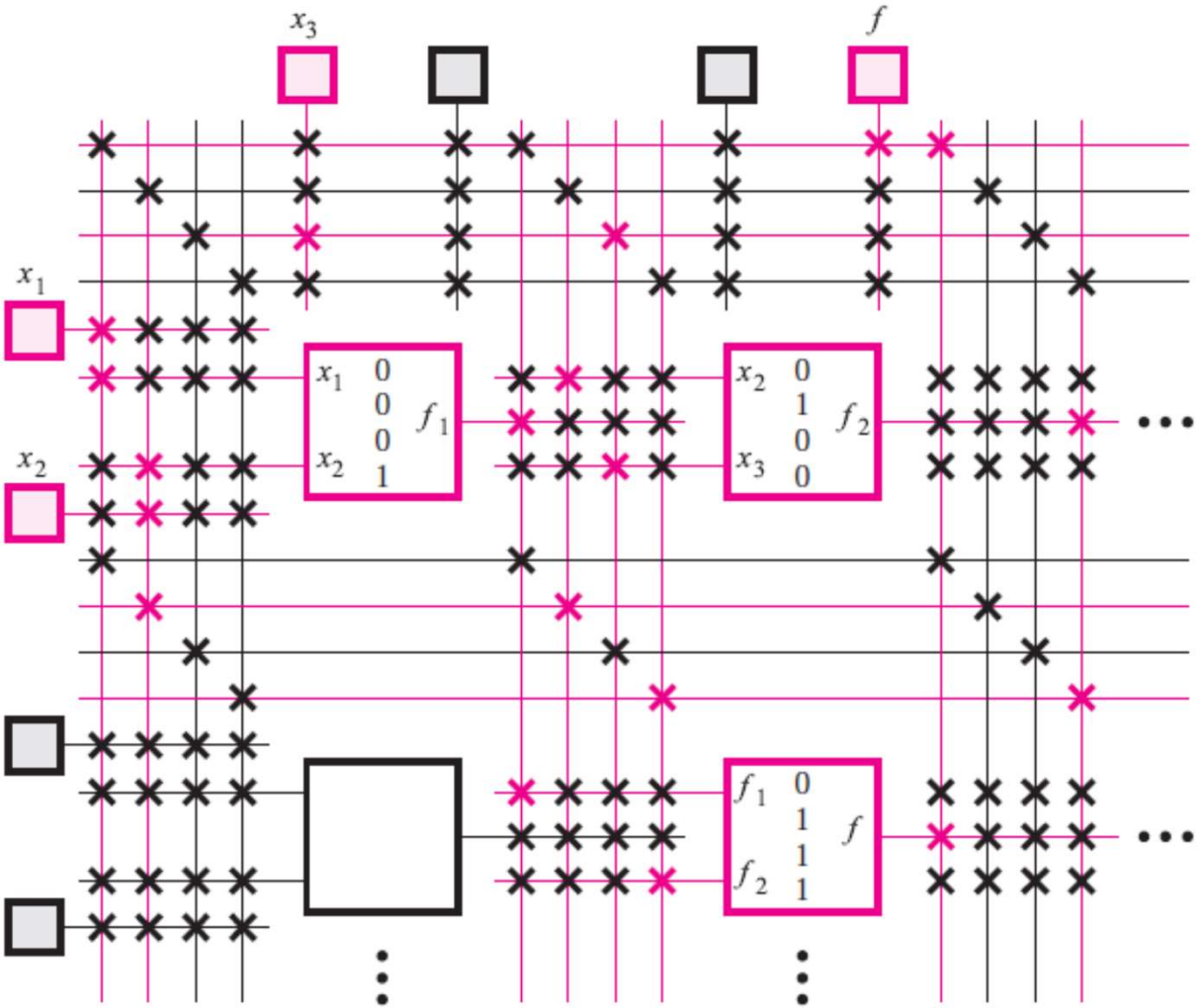


A sea-of-gates gate array



$$f_1 = x_2\overline{x_3} + x_1x_3$$

Example of Field-Programmable PLD

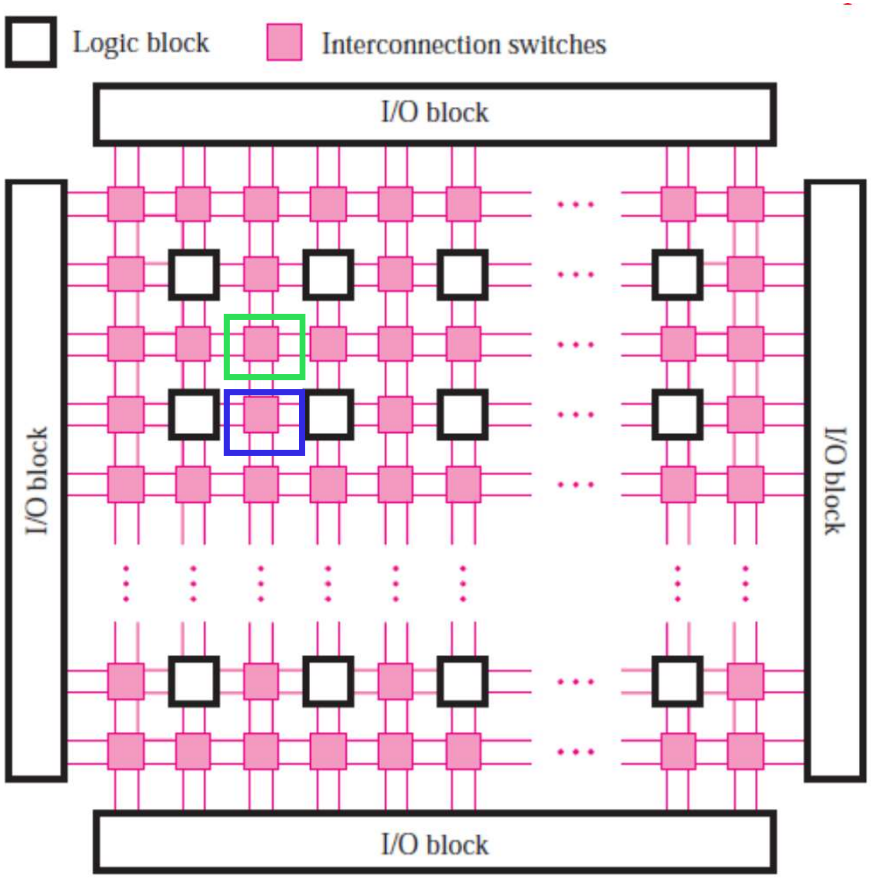


$$f = f_1 + f_2 = x_1 x_2 + \bar{x}_2 x_3$$

Review

- **Existing Integrated Circuits (ICs) can be classified into (3):**
  - **Application Specific IC (ASIC) or Custom-Designed Chips:**
    - **Aim to meet the desired performance or cost objectives.**
    - **chip is designed first and then manufactured by a company that has the fabrication facilities**
    - **Designed for:**
      - Video processing,
      - An interface between memory and CPU,
      - automobile, etc.

What is FPGA?



Overview of the FPGA architecture

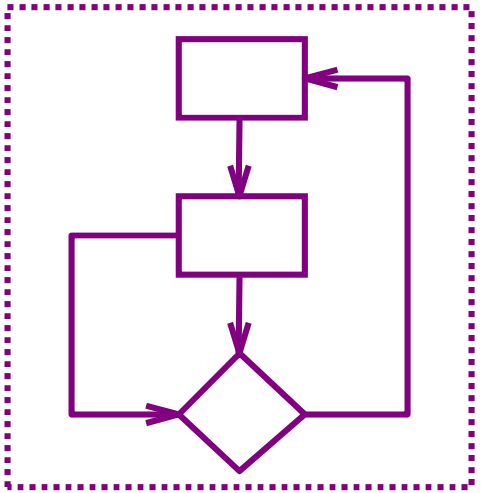
Field Programmable Gate Array:

- **Pre-fabricated digital** (IC) devices
- Electrically **programmed to become almost any kind of digital circuit or system**
- **Programming takes place "in the field".**
- Comprises of
  - Configurable logic blocks (CLB),
  - Programmable routing resources: wires and switches
  - I/O blocks.
- Adopts the configuration technologies:
  - SRAM-based technology
  - Flash/EEPROM technology
  - Anti-fuse technology

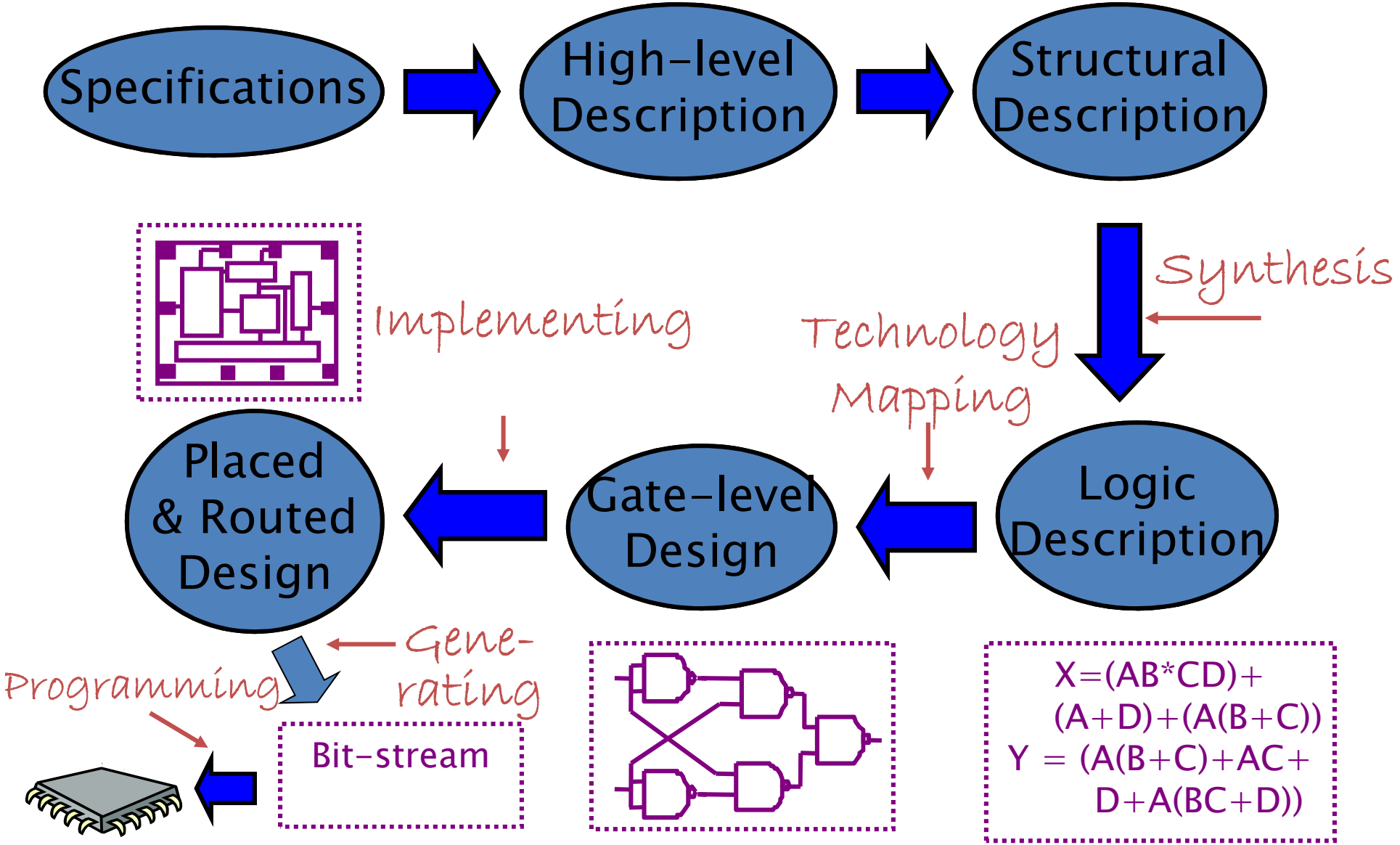
FPGA Design Flow

Behavioral
VHDL, C

Structural
VHDL



FPGA Design Flow



Contrasting Architectures

ASIC
Application Specific Integrated Circuit

- full custom design from behavioral description to physical layout
- designs must be sent for expensive and time consuming fabrication in semiconductor foundry

FPGA
Field Programmable Gate Array

- no physical layout design; design ends with a bitstream used to configure a device
- bought off the shelf and reconfigured by designers themselves

Contrasting Architectures

- **ASIC architecture compared to the Xilinx FPGA architecture**
  - Granularity: Gates vs. LUTs
  - Delays: Low vs. High
  - Performance: High vs. Low
- **Fundamental considerations for selecting ASIC or FPGA**

Parameters	ASIC	FPGA
Cost (for first device)	100K~1000K $	1~xK $
Size	-	20 to 35 times more area
Performance		3 to 4 times slower
Power		10 times more dynamic power
Analog Circuit	OK	Don't support
Time to market (Fabrication)	months	Few seconds
Reprogrammable	No	Very flexible

Contrasting Architectures

FPGAs

Buying FPGA off the shelf

$100

ASICs

$$\frac{\text{Cost of preparing the masks}}{\text{Number of chips}}$$

Cost of fabricating an ASIC = $3

Cost of preparing the masks = $1,000,000

Case 1: Number of chips = 1000

Total cost = $1,000,000/1000 + $3 = **$1003**

Case 2: Number of chips = 100,000

Total cost = $1,000,000/100,000 + $3 = **$13**

FPGA Applications

- **Implementing the prototype for ASIC designs**
- **Providing a hardware platform to verify the physical implementation of new algorithms in:**
  - Digital signal processing (DSP),
  - Baseband processing in communication,
  - Software-defined radios,
  - Radar,
  - Video, image processing,
  - Physical layer communication interfaces, etc
- **On-Chip embedded processing systems**
- **Functioning reconfigurable hardware in Reconfigurable Computing**

FPGA's Key Properties

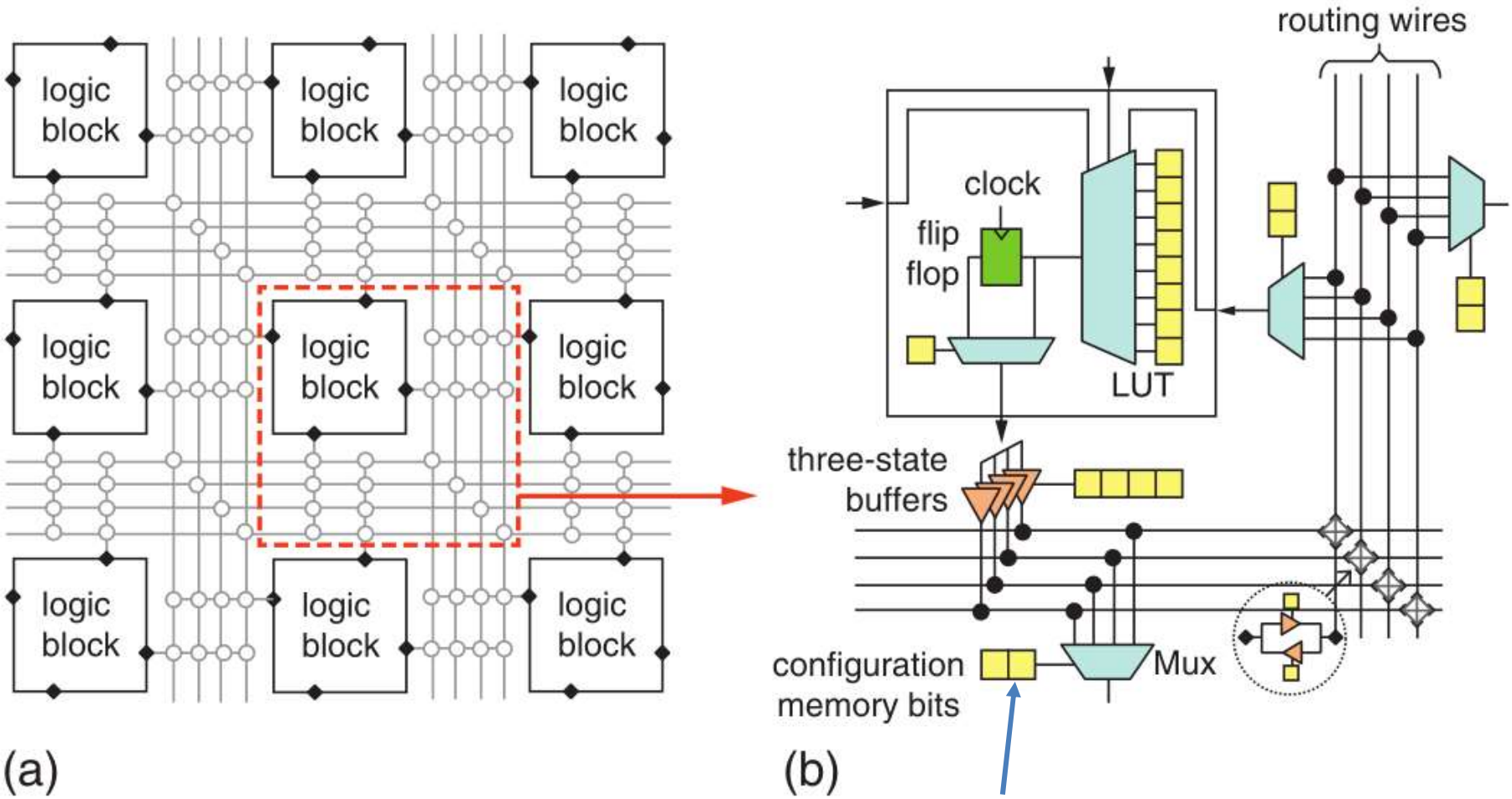
Key properties of an FPGA device depend on two factors

- Configuration technology:
  - What are the programmable links?
  - How is the configuration stored?
  - How many times can it be changed?
  - Can this be done without removing the device from the board?
- Organization of hardware resources:
  - What prefabricated hardware resources are made available to customers?
  - How can they be made to form a larger circuit?

➢All **configuration techniques** in use today have their roots in semiconductor memory technology (SRAM, ash, PROM).

Configuration Technologies

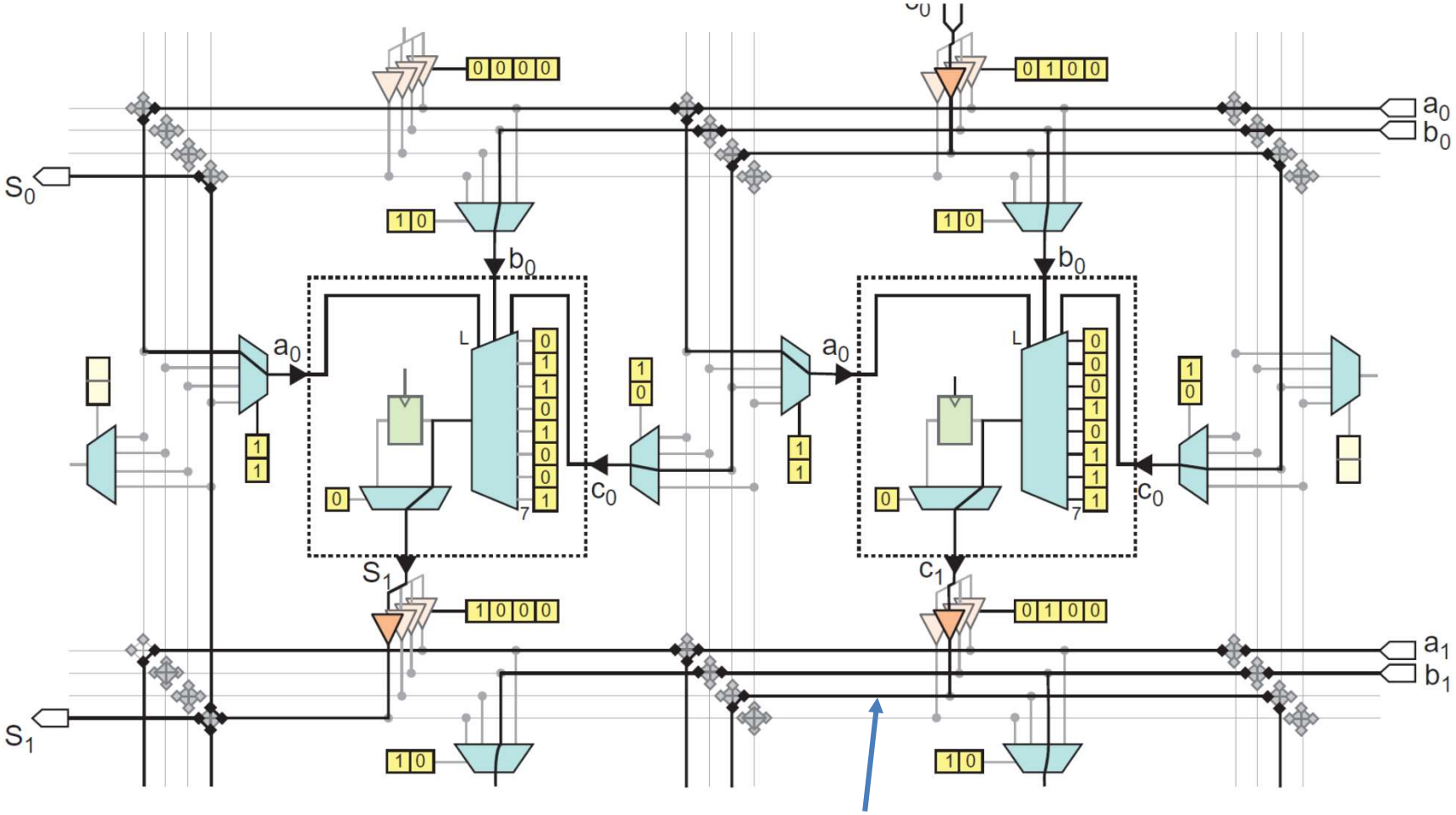
FPGA hardware resources before configuration.



A memory element for storing configuration information

Configuration Technologies

FPGA hardware resources after configuration.



Highlighted lines show the wires activated by configuration bits

Configuration Technologies

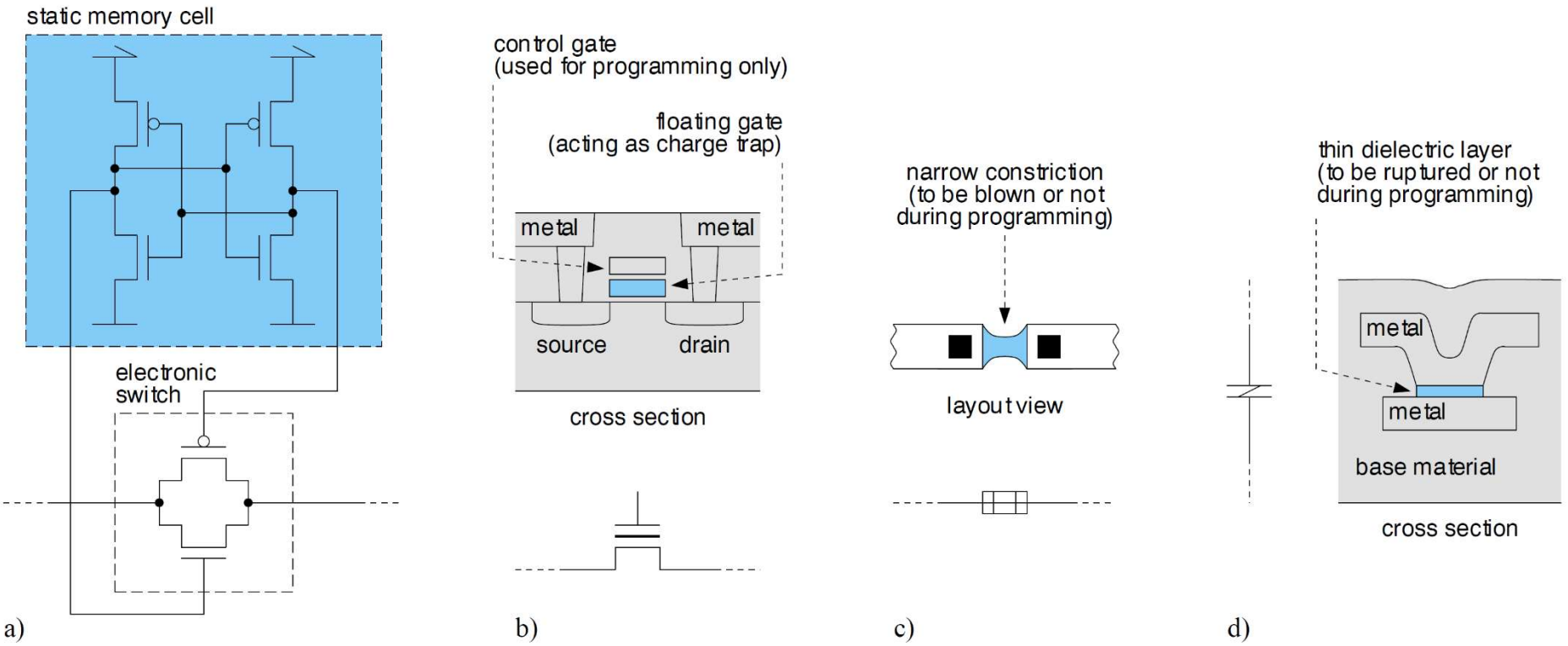


Figure: Electrical connections that can be done by electrical means.

- SRAM: Switch steered by static memory cell (a),
- Flash: MOSFET controlled by trapped charge (b),
- PROM: fuse (c) and antifuse (d).

Configuration Technologies

(1) SRAM-Based Programming Technology

- **Characteristics:**
  - Static memory cells are used as the basic cells,
  - the dominant approach for the existing FPGAs
- **Advantages:**
  - **re-programmability; the use of standard CMOS process technology**
  - **higher speed and lower dynamic power consumption**
- **Disadvantages:**
  - **Larger area compared to other programming technologies**
    - **an SRAM cell requires 6 transistors**
  - **SRAM cells are volatile**

Configuration Technologies

(2) EEPROM/Flash-based Programming Technology

- **Characteristics:**
  - can be electrically programmed
- **Advantages:**
  - nonvolatile
  - Is more efficient in term of area than SRAM-based programming technology
- **Disadvantages:**
  - can not be reconfigured/reprogrammed an infinite number of times
  - flash-based technology uses non-standard CMOS process

Configuration Technologies

(3) Anti-fuse Programming Technology

- **Characteristics:**
  - one-time programmable (OTP)
- **Advantages:**
  - low area;
  - non-volatile
- **Disadvantages:**
  - does not use standard CMOS process
  - can not be reprogrammed

Major FPGA Vendors

SRAM-based FPGAs

- **Xilinx, Inc.** **~ 56% of the market**
- **Altera Corp.** **~ 34% of the market**

} **~ 90%**

- Lattice Semiconductor
- Quick Logic Corp.
- Achronix
- Atmel

Flash & antifuse FPGAs

- Actel Corp. (Microsemi SoC Products Group)

Xilinx FPGA Families

Technology	Low-cost	Mid-range	High-performance
220 nm			Virtex
180 nm	Spartan-II, IIE		
120/150 nm			Virtex-II, Pro
90 nm	Spartan-3		Virtex-4
65 nm			Virtex-5
45 nm	Spartan-6		
40 nm			Virtex-6
28 nm	Artix-7, Spartan-7	Kintex-7	Virtex-7
20 nm		Kintex UltraScale	Virtex UltraSCALE
16 nm	Artix UltraScale+	Kintex UltraScale+	Virtex UltraSCALE+

Altera FPGA Devices

Technology	Low-cost	Mid-range	High-performance
130 nm	Cyclone		Stratix
90 nm	Cyclone II		Stratix II
65 nm	Cyclone III	Arria I	Stratix III
40 nm	Cyclone IV	Arria II	Stratix IV
28 nm	Cyclone V	Arria V	Stratix V
20 / 14 nm		Arria 10	Stratix 10
10 nm			Agilex

Organization of Hardware Resources

Five primary elements

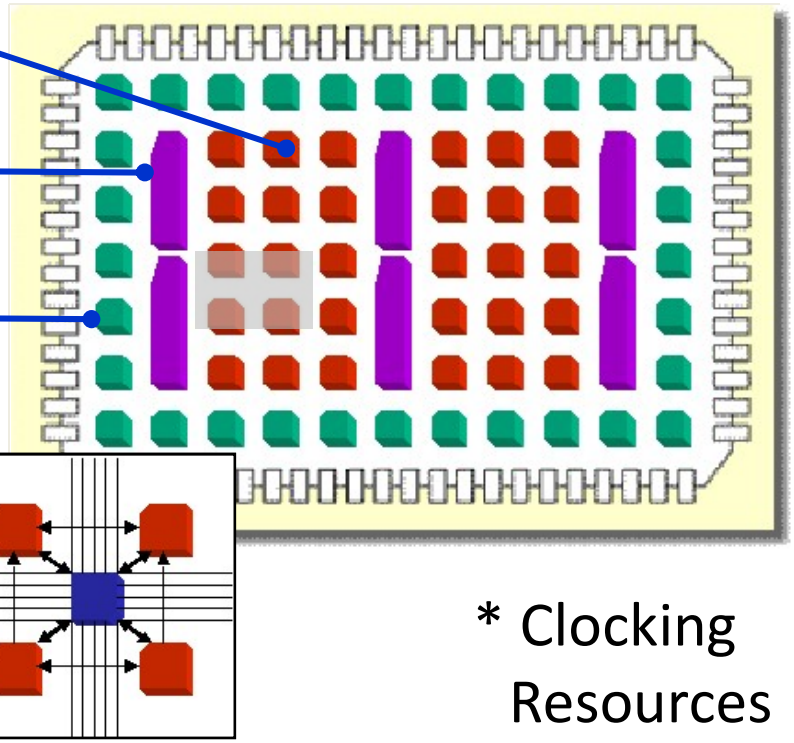
Configurable logic blocks

Dedicated blocks

Input and output blocks

Routing

Xilinx FPGAs



* Clocking Resources

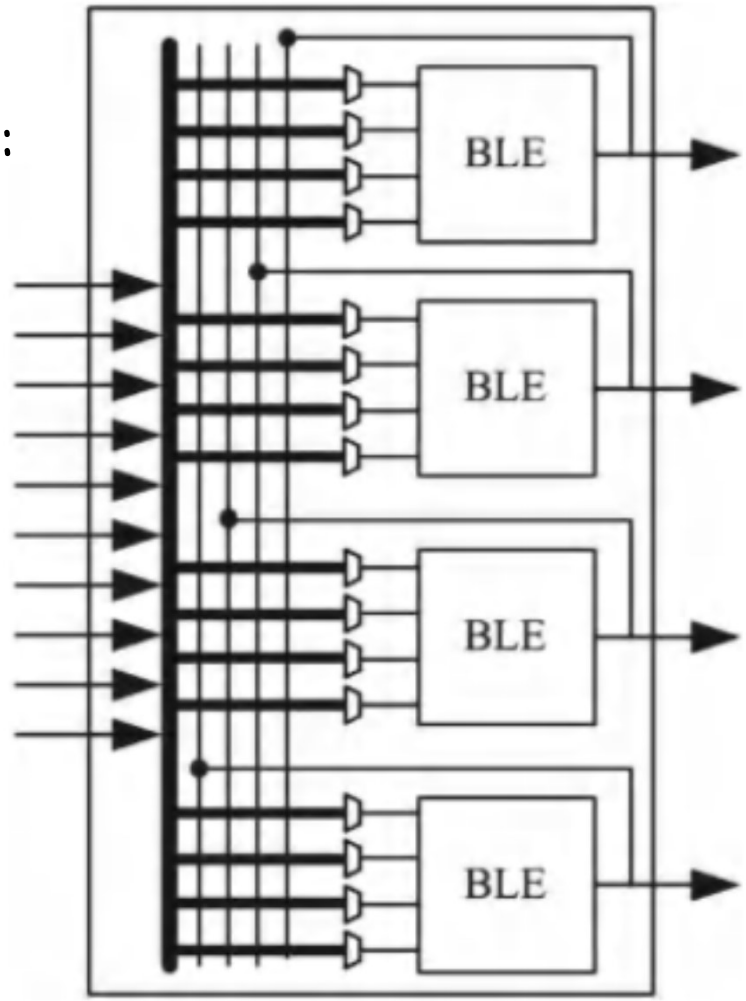
Configurable Logic Blocks

- Basic component, provides the **basic logic** and **storage functionality**
- Granularity:
  - Fine-grained: Logic Gates
  - Medium-grained: Multiplexors, LUTs, Flip-Flop, etc
  - Coarse-grained: Processor cores, ALU, etc
- Specific Purpose Hard Block: Memory, Multipliers, Adders, and DSP blocks, high-speed input/output (I/O) interfaces
  - very efficient at implementing specific functions
  - wasting huge amount of logic and routing resources if unused

Configurable Logic Blocks

– 2 types of architecture:

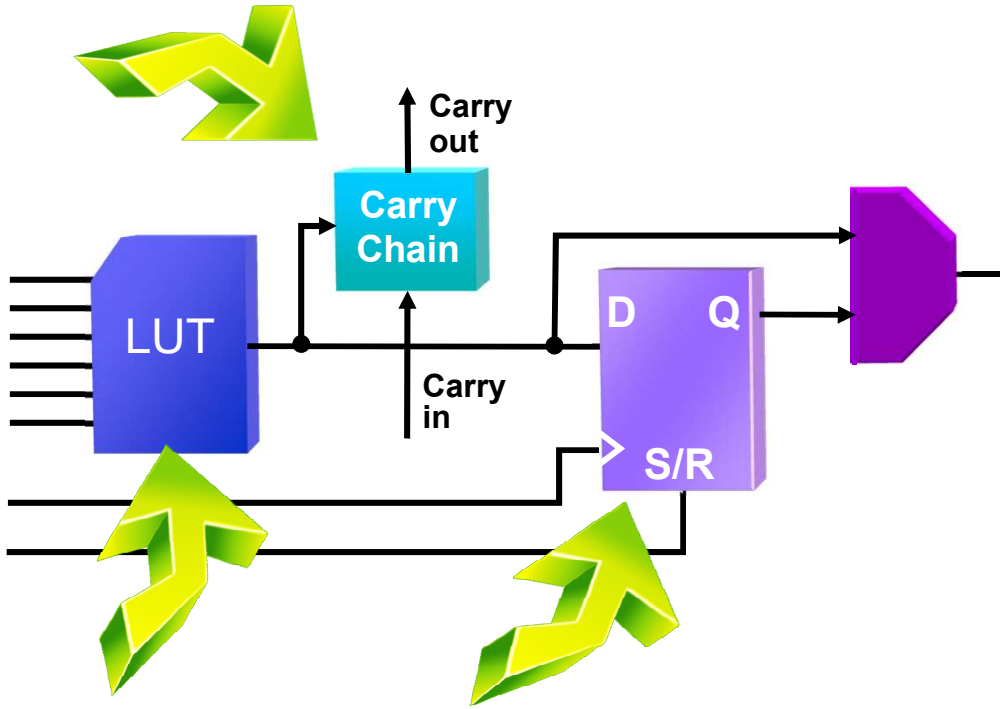
- A single basic logic element (BLE): also called **Logic cells**
- Cluster of locally interconnected BLEs: also called **Slices**



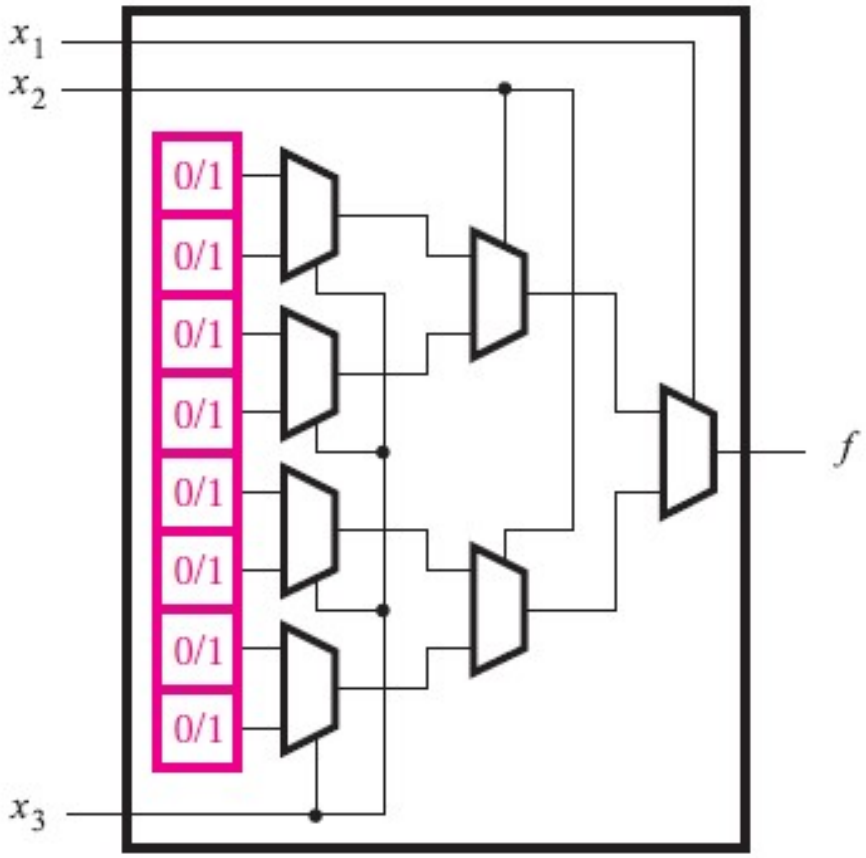
A configurable logic block (CLB) having four BLEs

BLE or Logic Cell

- **Logic cells include**
  - Combinatorial logic, arithmetic logic, and a register
- **Combinatorial logic** is implemented using Look-Up Tables (LUTs)
- **Register** can function as latches, JK, SR, D, and T-type flip-flops
- **Arithmetic logic** is a dedicated carry chain for implementing fast arithmetic operations



LUT: Lookup Table

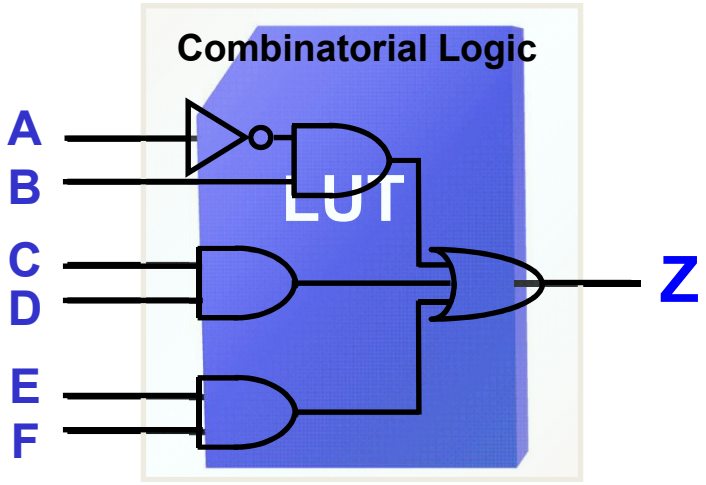


- **Used to implement a small logic function**
- **Composed of:**
  - *storage cells* store values that produce the output of the logic function *f*
  - Multiplexers select the content of one of the storage cells as the output of the LUT
- **LUT's *size*, *k*, is defined by the number of inputs,**
  - Each LUT can implement any function of k variabls

Combinatorial Logic

A	B	C	D	E	F	Z
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	1
0	0	0	1	0	0	1
0	0	0	1	0	1	
	1			.	.	.
0	0	1	1	0	0	0
0	0	1	1	0	1	0
0	0	1	1	1	0	0
0	0	1	1	1	1	1

- LUTs function as a Memory or can **perform any combinatorial function**

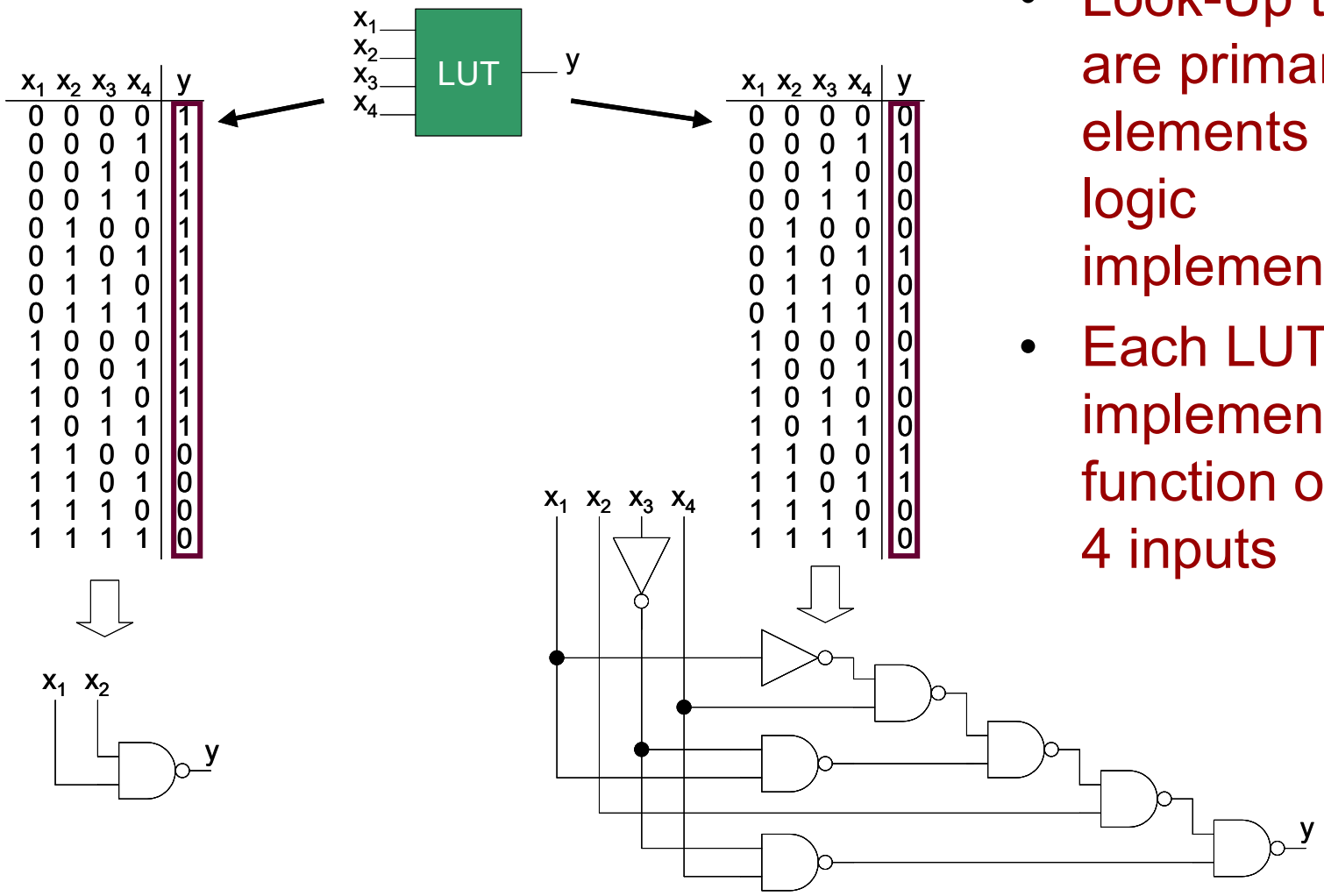


They generate the output value…

for a given set of inputs

- Constant delay through a LUT
- Limited by the number of inputs and outputs, not by complexity

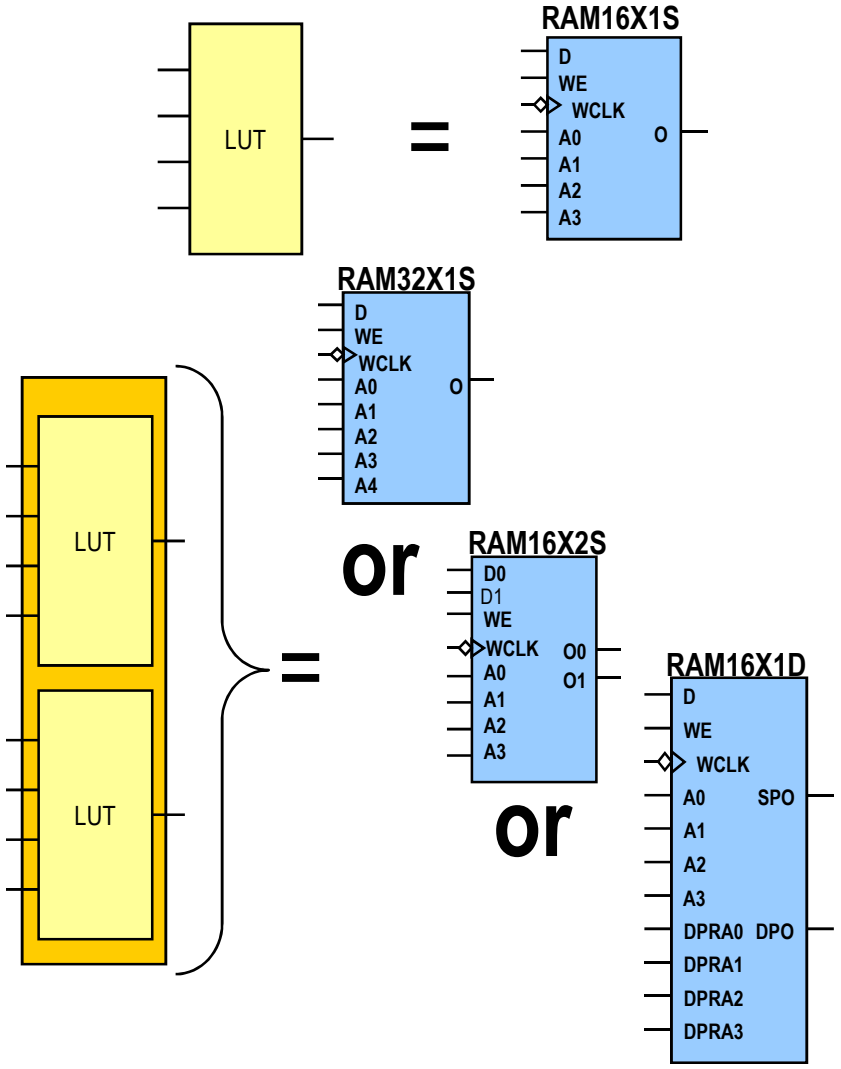
4-input LUT (Look-Up Table)



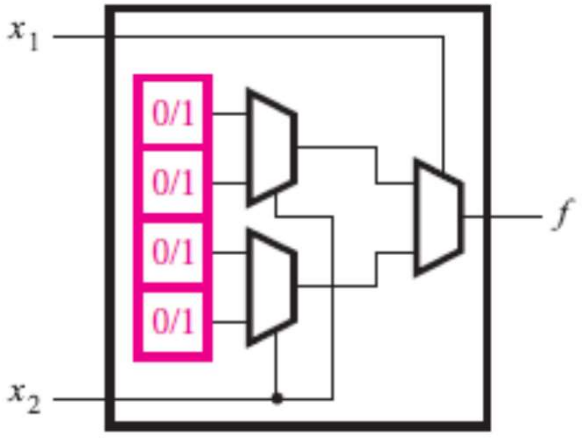
- Look-Up tables are primary elements for logic implementation
- Each LUT can implement any function of 4 inputs

Distributed RAM

- CLB LUT configurable as Distributed RAM
  - A single 4-LUT equals 16x1 RAM
  - Two LUTs implement Single and Dual-Port RAMs
  - Cascade LUTs to increase RAM size
- Synchronous write
- Synchronous/Asynchronous read
  - Accompanying flip-flops used for synchronous read



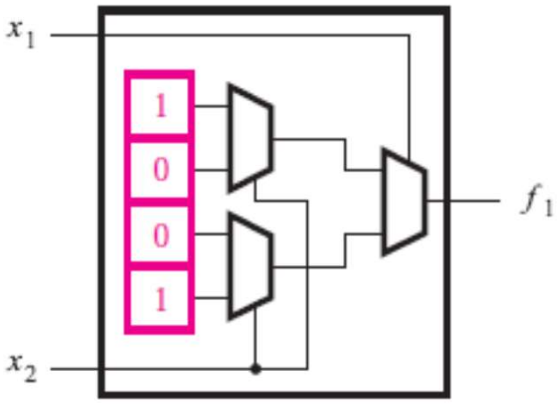
LUT: A Simple Example



(a) Circuit for a two-input LUT

x_1	x_2	f_1
0	0	1
0	1	0
1	0	0
1	1	1

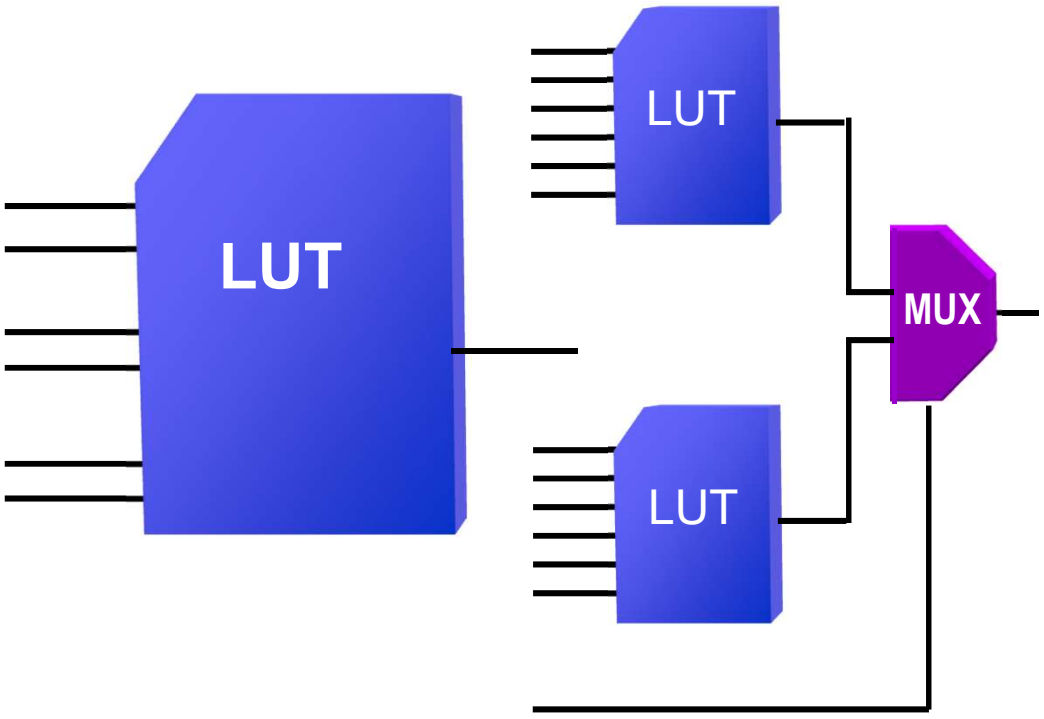
(b) $f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$



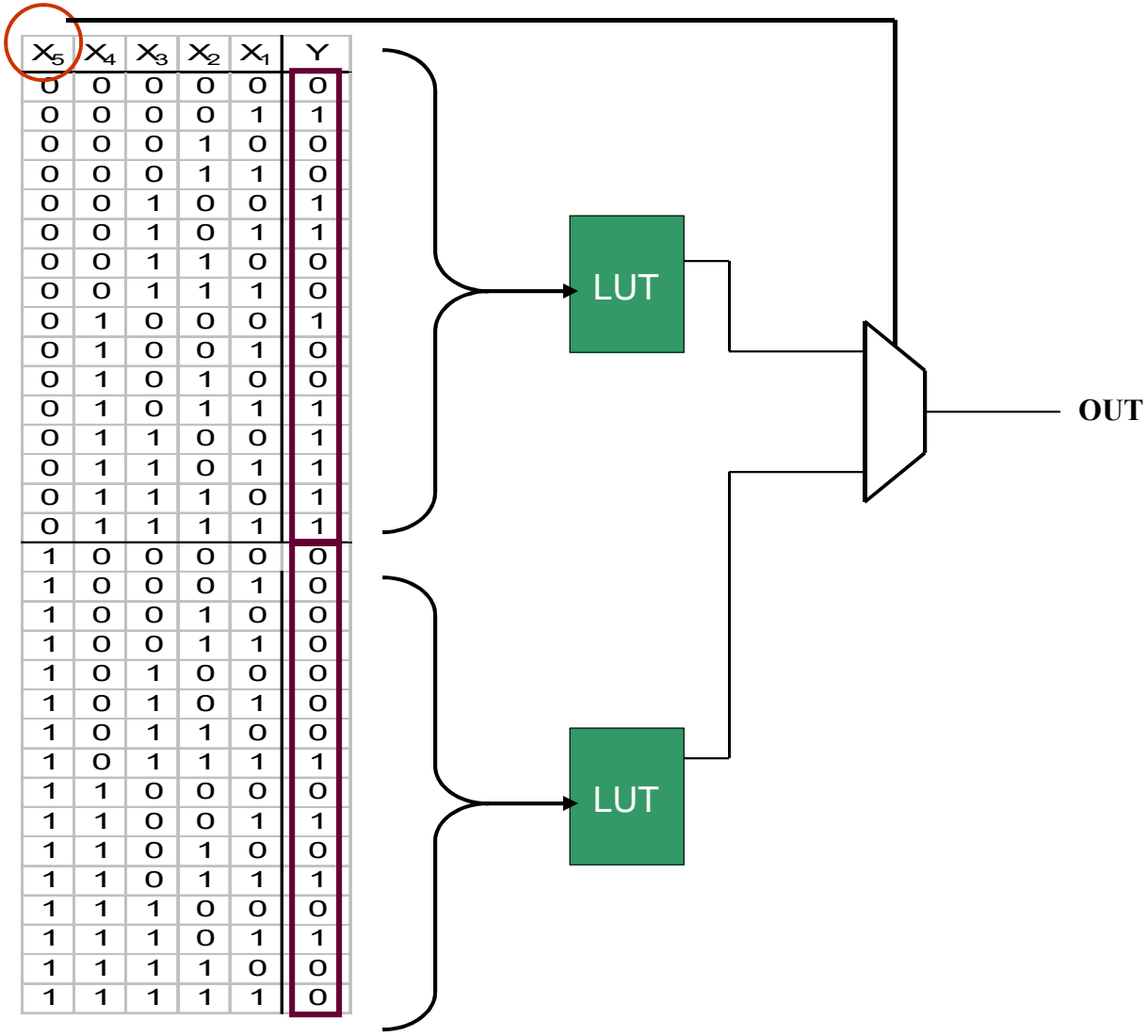
(c) Storage cell contents in the LUT

Wide Input Functions

- For wider input functions, LUTs can be combined using a multiplexer
- These muxes are dedicated, so they are fast



5-Input Functions implemented using two LUTs

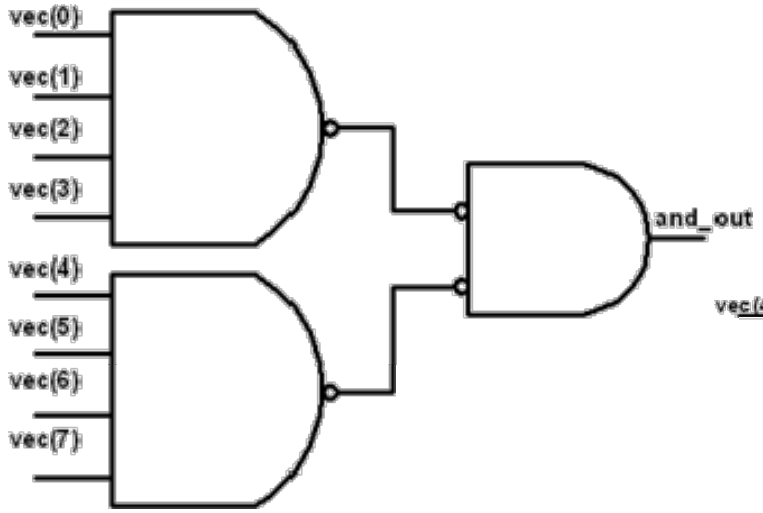


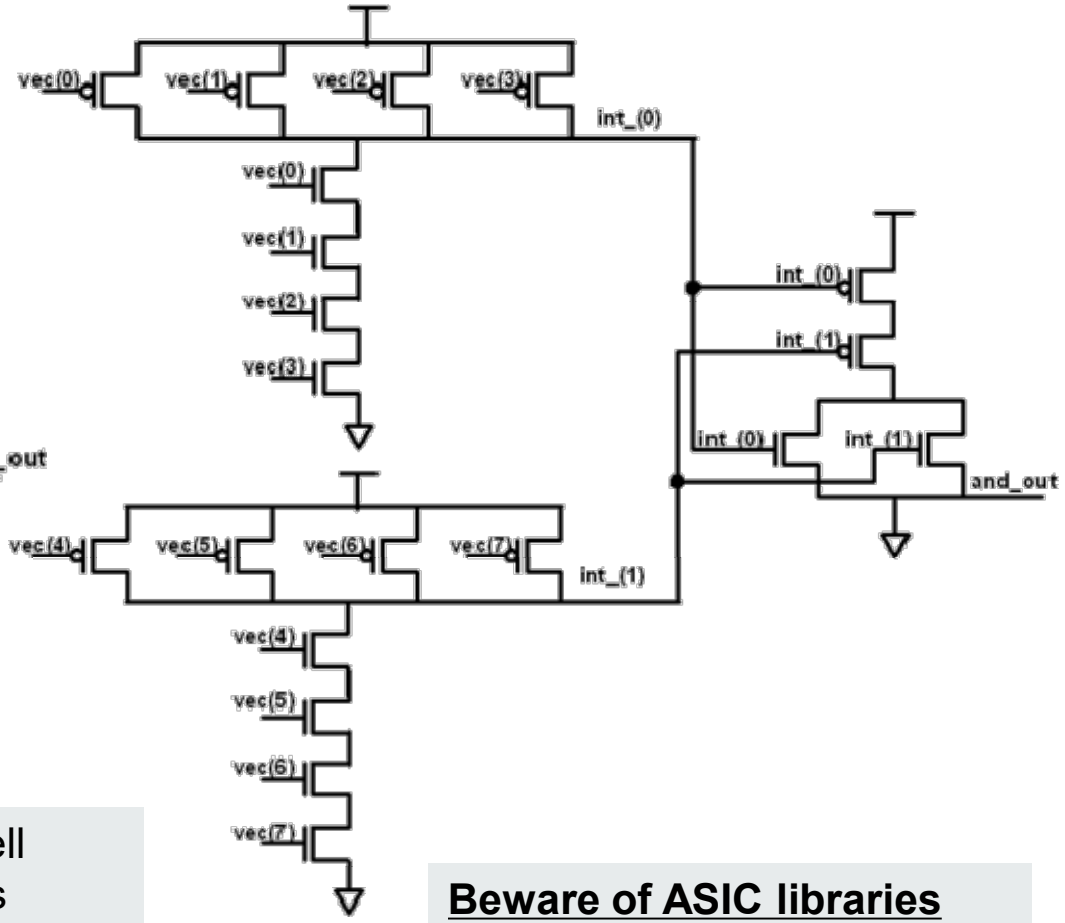
X_5	X_4	X_3	X_2	X_1	Y
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	0

ASIC Implementation

8-input AND gate

Two four-input NAND gates feeding a two-input NOR gate



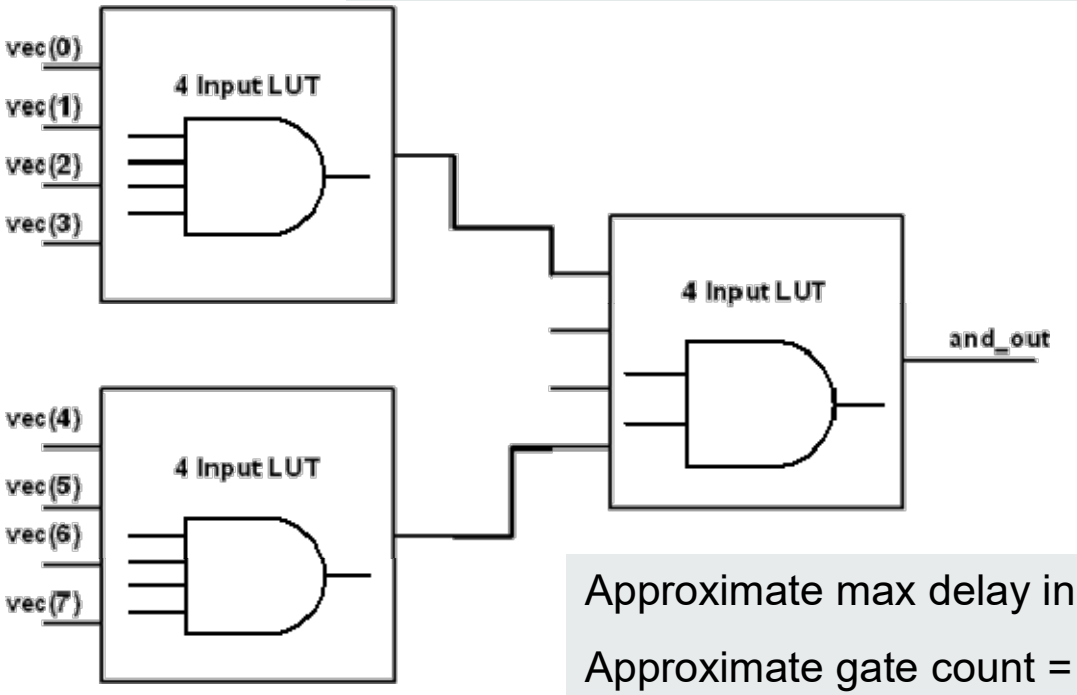


Approximate delay in a standard-cell ASIC with 0.13-μ process = 0.47 ns

Beware of ASIC libraries with very wide gate types!

Xilinx Implementation

8-input AND gate implemented in three 4-input LUTs and two logic levels



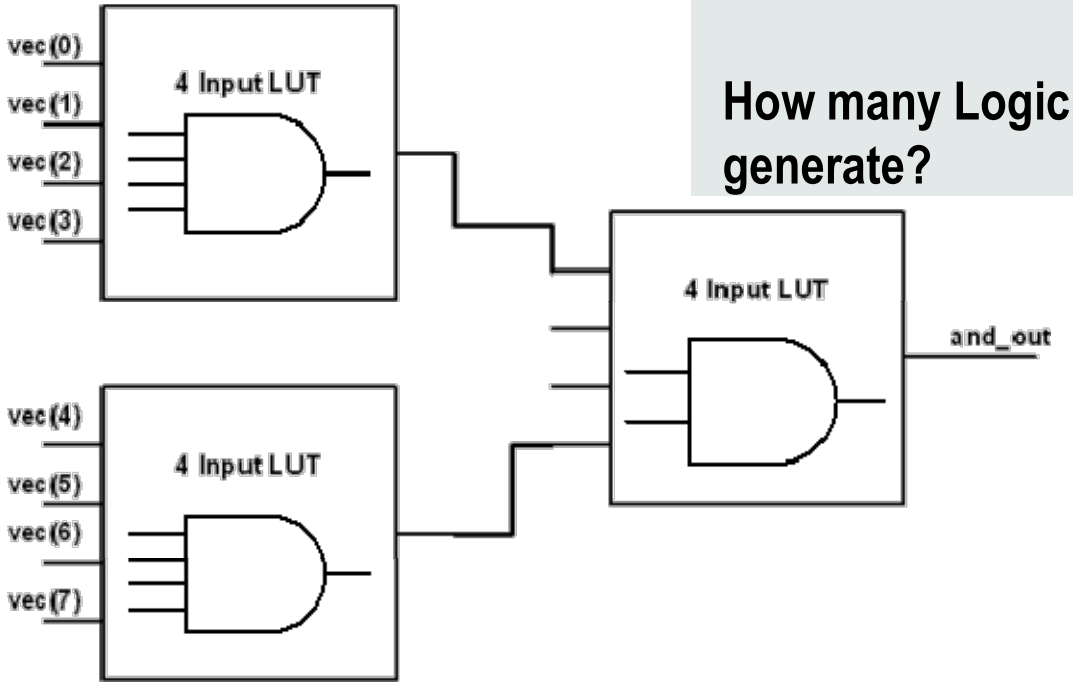
Approximate max delay in a Spartan®-3 FPGA = 0.678 ns

Approximate gate count = 18 gates

Approximate max delay in a Virtex-5 FPGA = 0.435 ns

Approximate gate count = 18 gates

Quiz

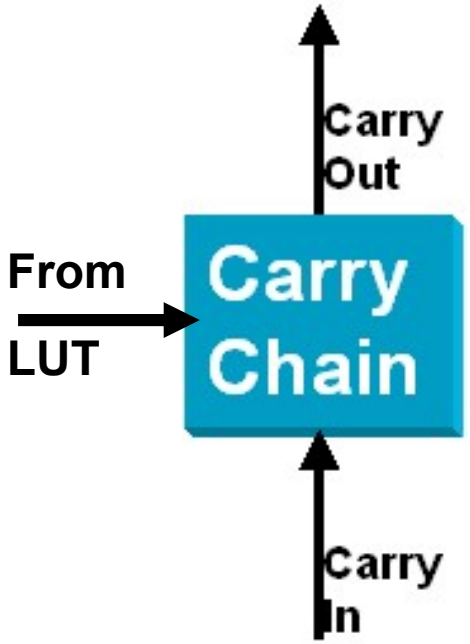


How many 4-input LUTs would be required to implement a 32-input OR gate?

How many Logic Levels would they generate?

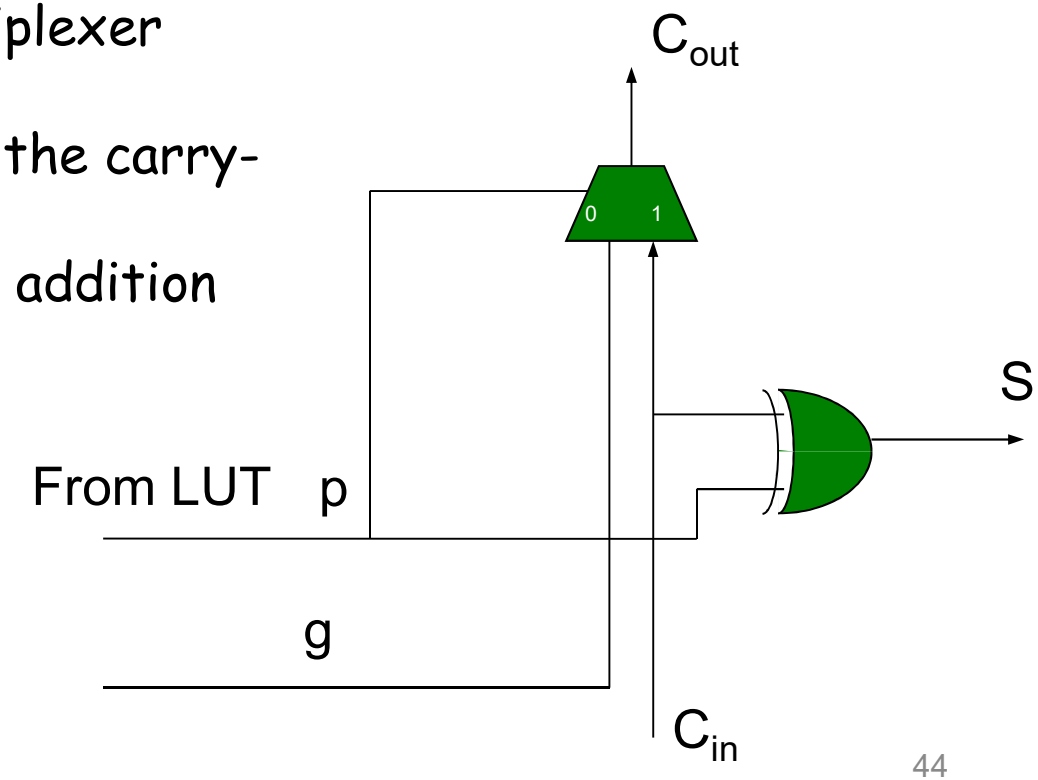
Carry Logic

- Each CLB contains separate logic and routing resource for the fast generation of sum & carry signals
  - Increases efficiency and performance of adders, subtractors, accumulators, comparators, and counters
- Carry logic is independent of normal logic and routing resources

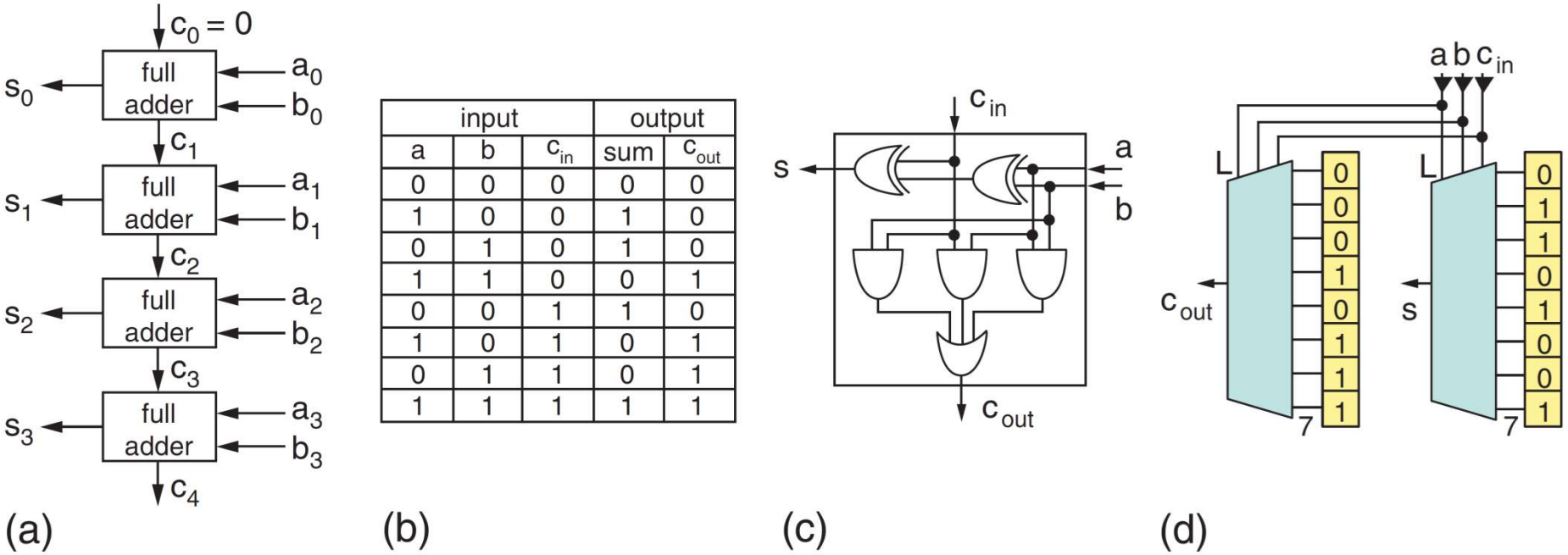


Carry Logic

- The carry logic chain is dedicated logic that computes high-speed arithmetic logic functions
- The carry chain generally consists of a multiplexer and an XOR gate
  - The LUT computes the multiplexer selector
  - The multiplexer determines the carry-out
  - The XOR gate computes the addition



Carry Logic



input			output	
a	b	c_{in}	sum	c_{out}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

An n-bit ripple-carry adder: General diagram (a), truth table of one full adder slice (b), gate-level circuit (c), and implementation by 3-input LUTs (d).

The problem with this implementation is that it requires two LUTs for every input bit, and the Carry propagates through the full LUT delay for each bit.

Carry Logic

Alternative Implementations of Full Adder

x	y	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

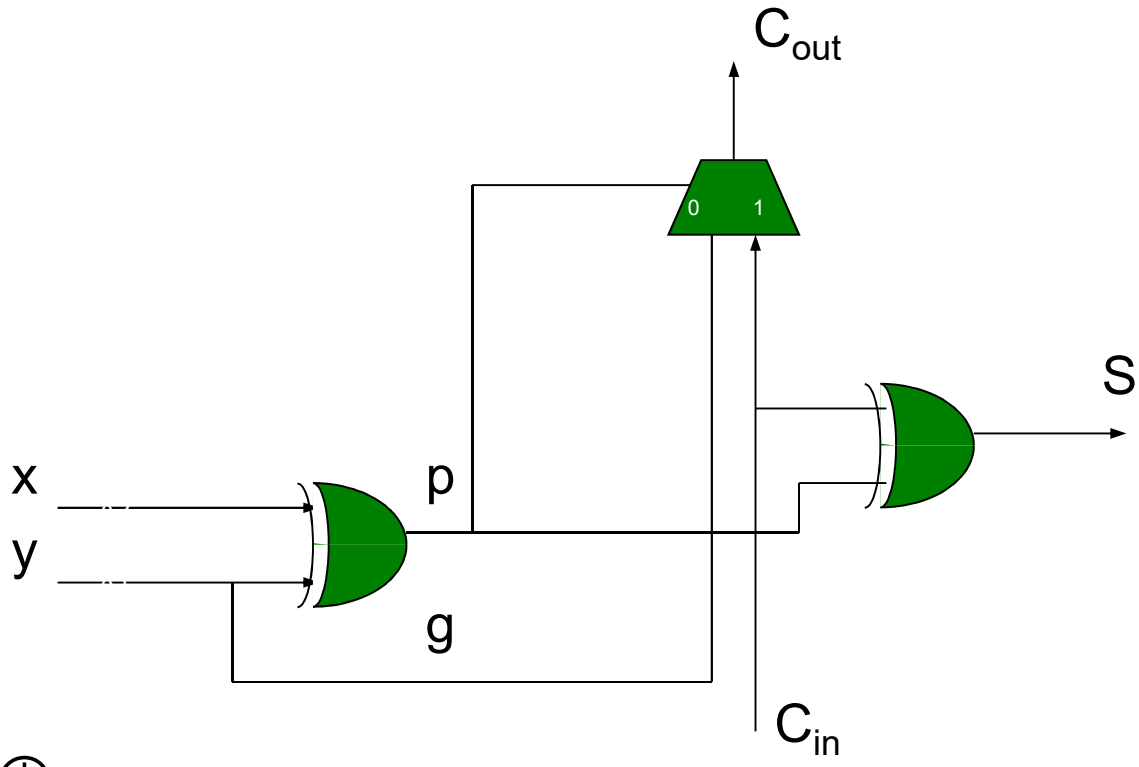
x	y	c_{out}	s
0	0	y	c_{in}
0	1	c_{in}	$\overline{c_{in}}$
1	0	c_{in}	$\overline{c_{in}}$
1	1	y	c_{in}

An n-bit ripple-carry adder **vs.** **Look ahead carry adder**

Carry Logic

Look-Ahead Carry Implementations of Full Adder

x	y	c_{out}
0	0	y
0	1	c_{in}
1	0	c_{in}
1	1	y



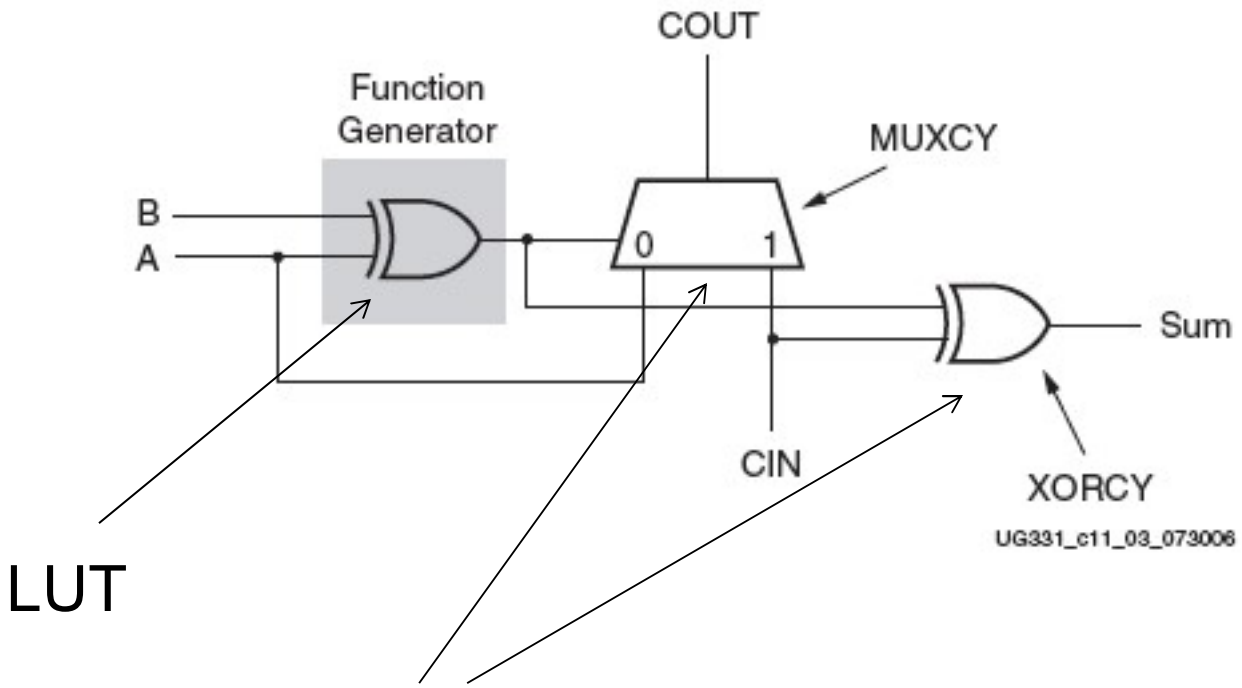
$p = x \oplus y$

$g = y$

$s = p \oplus c_{in} = x \oplus y \oplus c_{in}$

Carry Logic

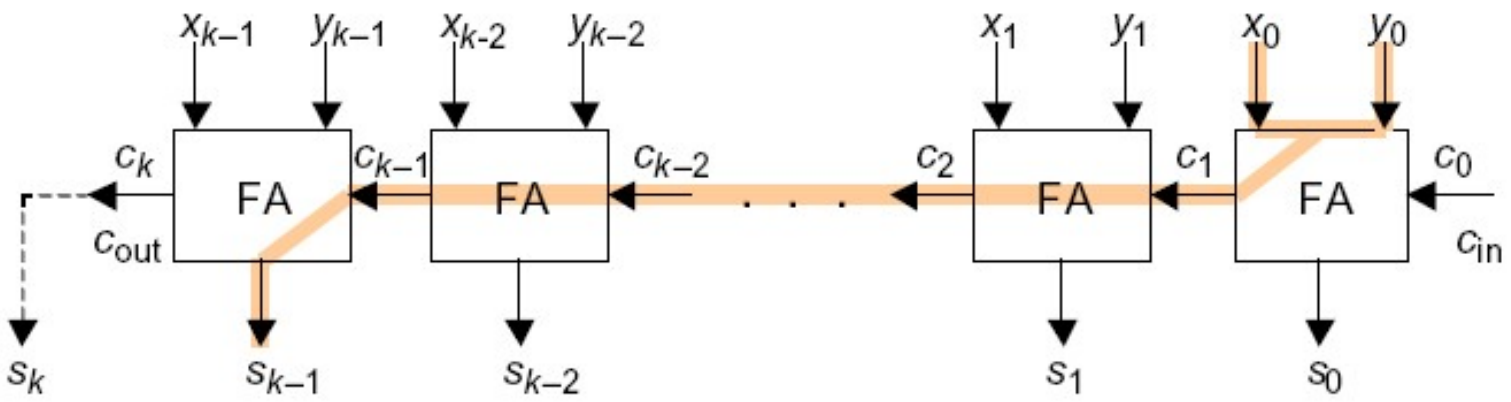
Look-ahead carry implementation by the fast carry logic in Xilinx FPGAs



Hardwired (fast) logic

Carry Logic

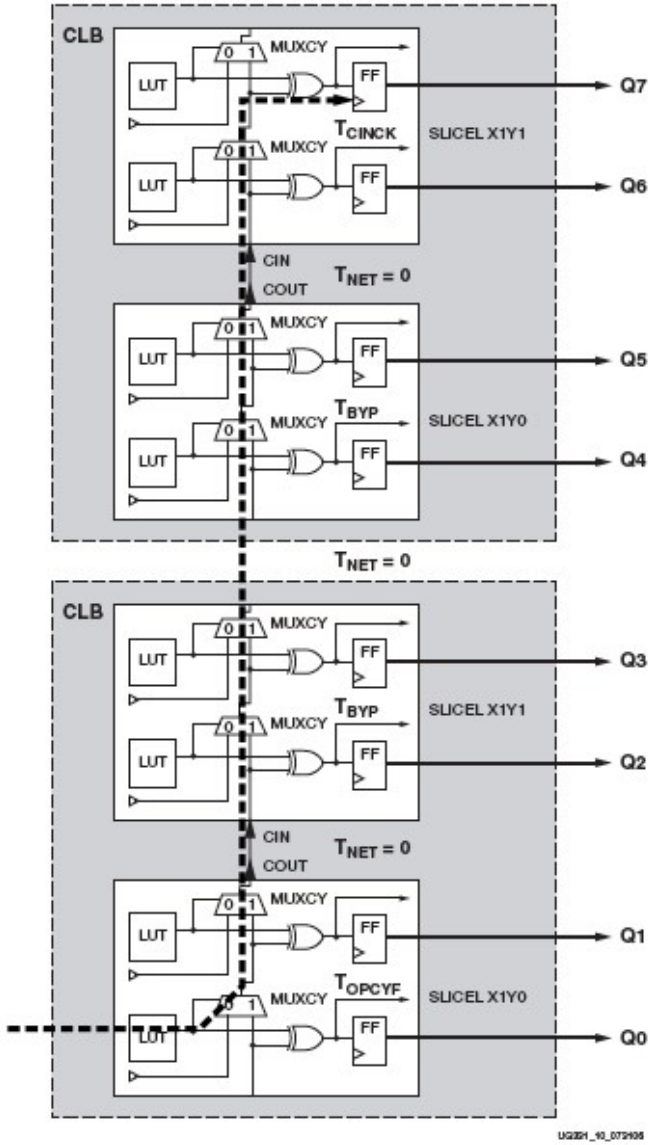
$$T_{\text{ripple-add}} = T_{FA}(x,y \rightarrow c_{out}) + (k-2) \times T_{FA}(c_{in} \rightarrow c_{out}) + T_{FA}(c_{in} \rightarrow s)$$



Critical path in a k-bit ripple-carry adder

Carry Logic

Critical Path for an Adder Implemented Using Xilinx Spartan 3/Spartan 3E FPGAs



Accessing Carry Logic

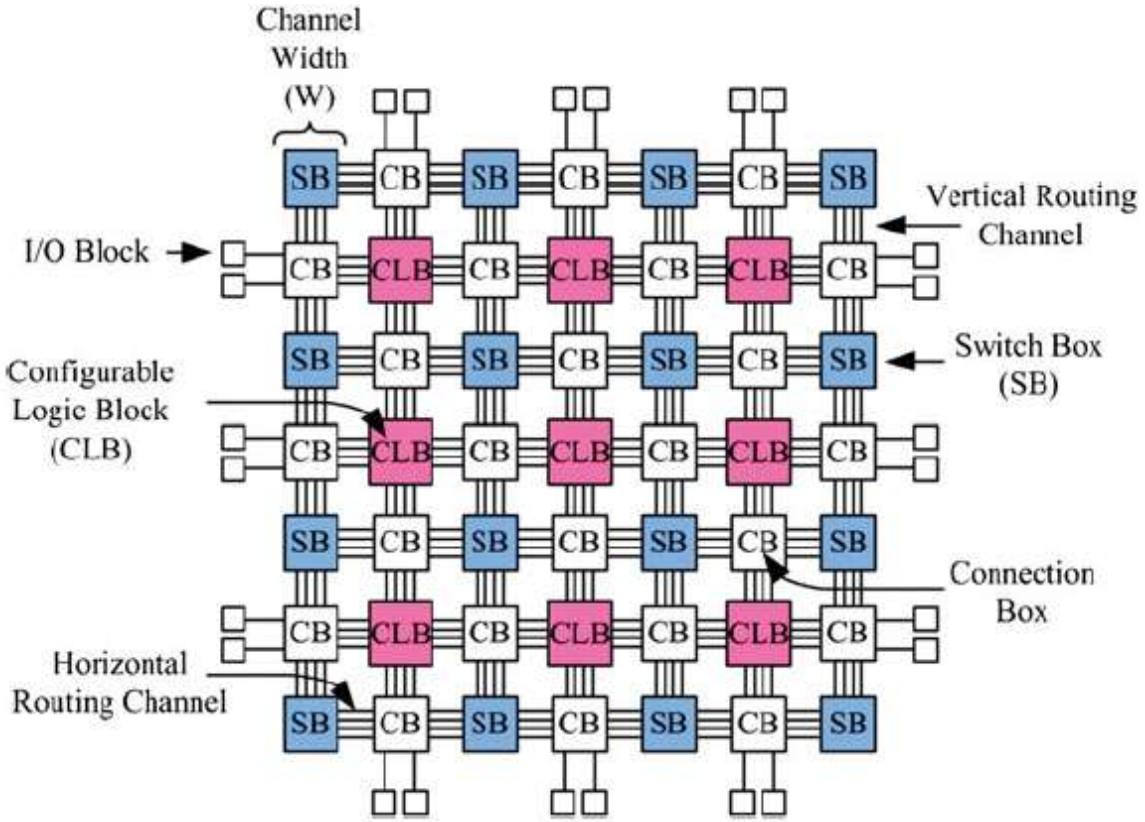
- All major synthesis tools can infer carry logic for arithmetic functions
  - Addition (SUM <= A + B)
  - Subtraction (DIFF <= A - B)
  - Comparators (if A < B then…)
  - Counters (count <= count +1)

Routing Network Architecture

- **Provides connections among logic blocks and I/O blocks to implement any user-defined circuit**
- **Comprises of wires and programmable switches**
- **Design Requirements:**
  - **Must be very flexible to accommodate a wide variety of circuits**
  - **Must be very efficiency to offer high performance**
- **Be optimized by taking into account the common characteristics of these circuits:**
  - Locality: requiring abundant short wires
  - some distant connections: leads to the need for sparse long wires.
- **Can be categorized as:**
  - Island-style
  - Hierarchical

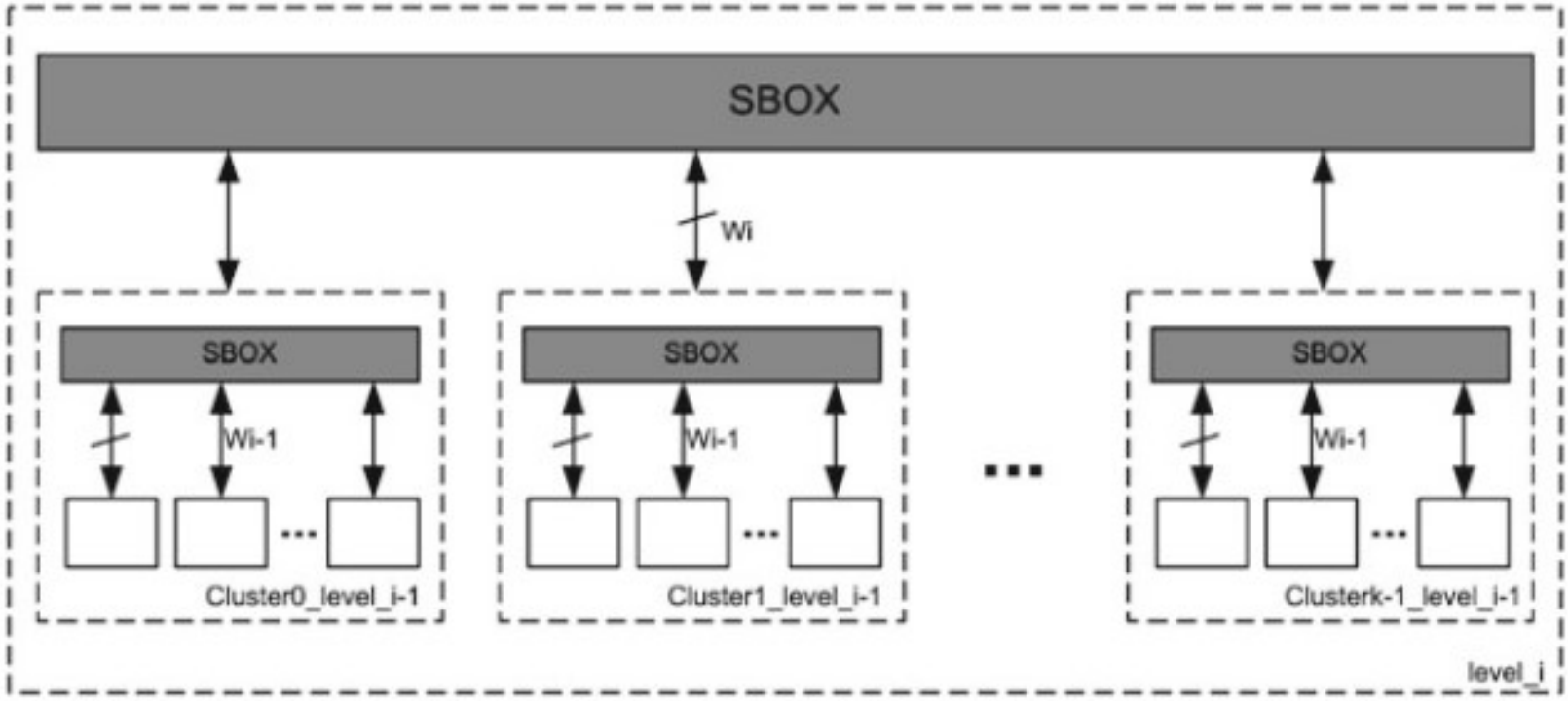
Routing Network Architecture

- **Island-style Architecture** (or mesh-based FPGA architecture):
  - The most commonly used architecture among commercial FPGAs
  - Configurable logic blocks look like islands in a sea of routing interconnect (the routing network occupies 80–90% of total area)



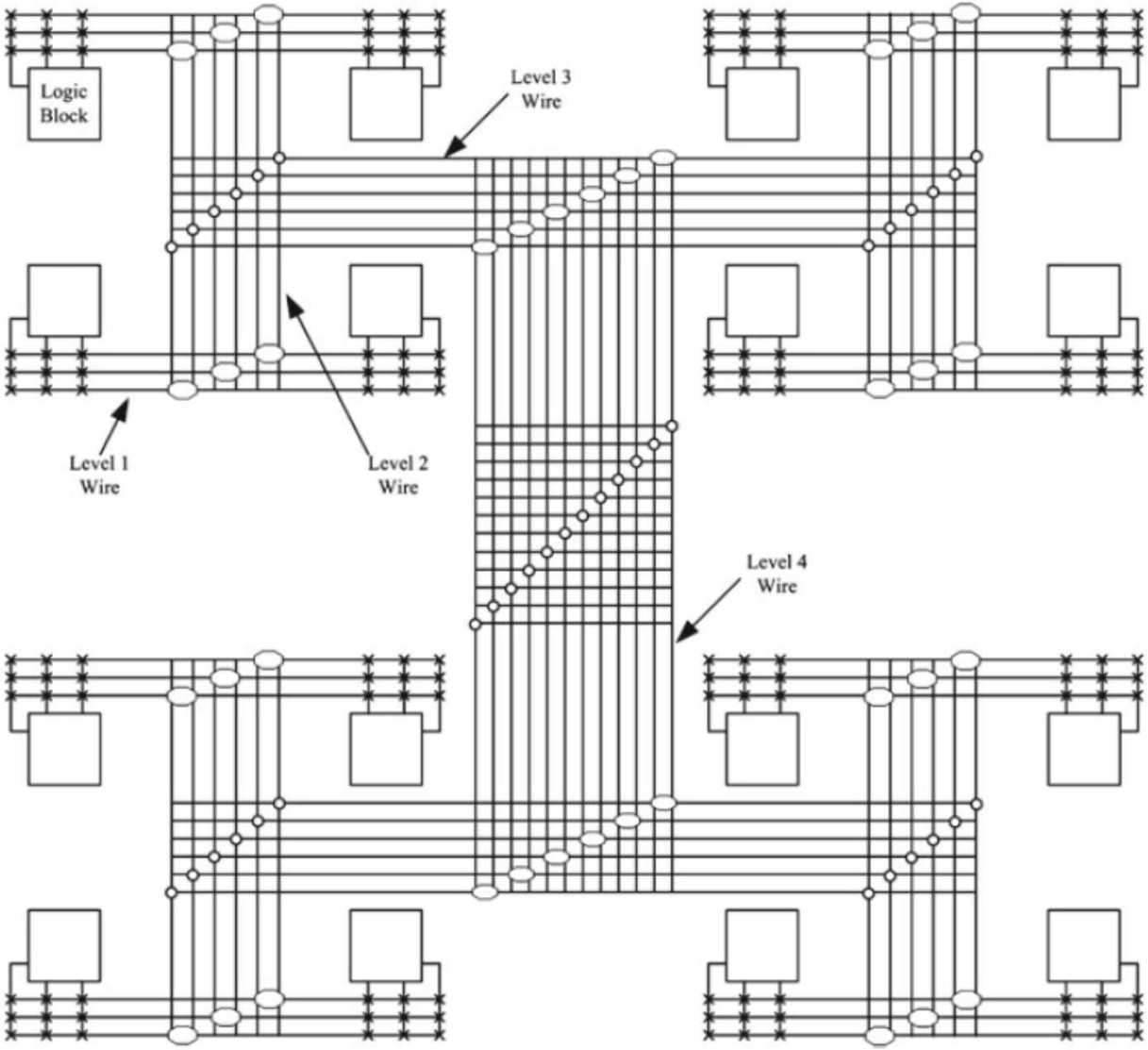
Routing Network Architecture

- **Hierarchical Architecture:**
  - Exploit this locality by dividing logic blocks into separate clusters
    - The connections between logic blocks within same cluster are made by wire segments at the lowest level of hierarchy
    - the connection between blocks residing in different groups require the traversal of one or more levels of hierarchy.



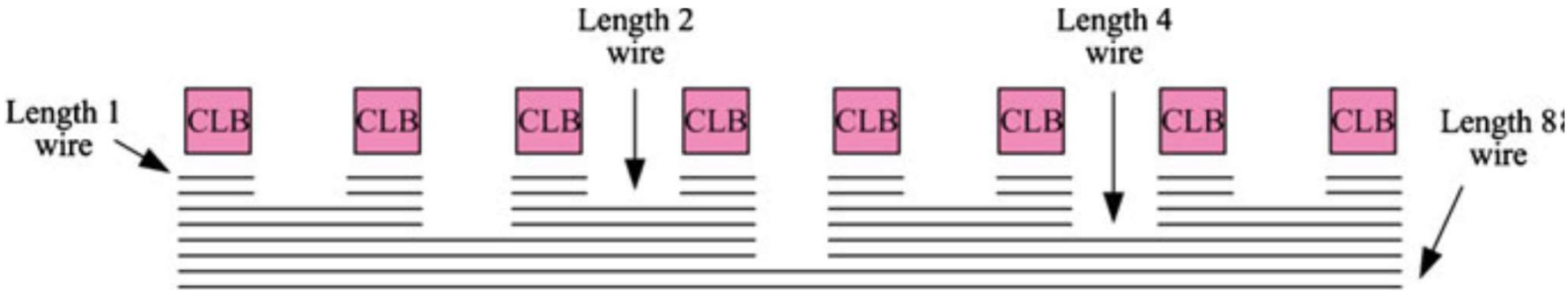
Routing Network Architecture

- **Hierarchical Architecture: Example**



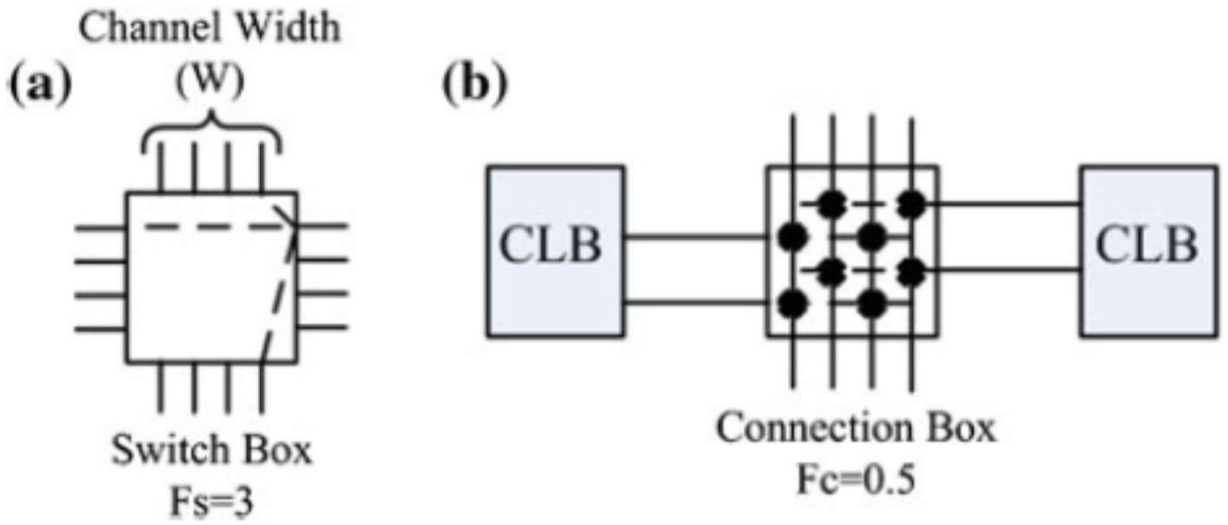
Routing Network Architecture

- **Multi-length wires are created to balance flexibility, area and delay of the routing networ**k
  - Longer wire segments:
    - Span multiple blocks and require fewer switches, thereby reducing routing area and delay
    - But also decrease routing flexibility, which reduces the probability to route a hardware circuit successfully



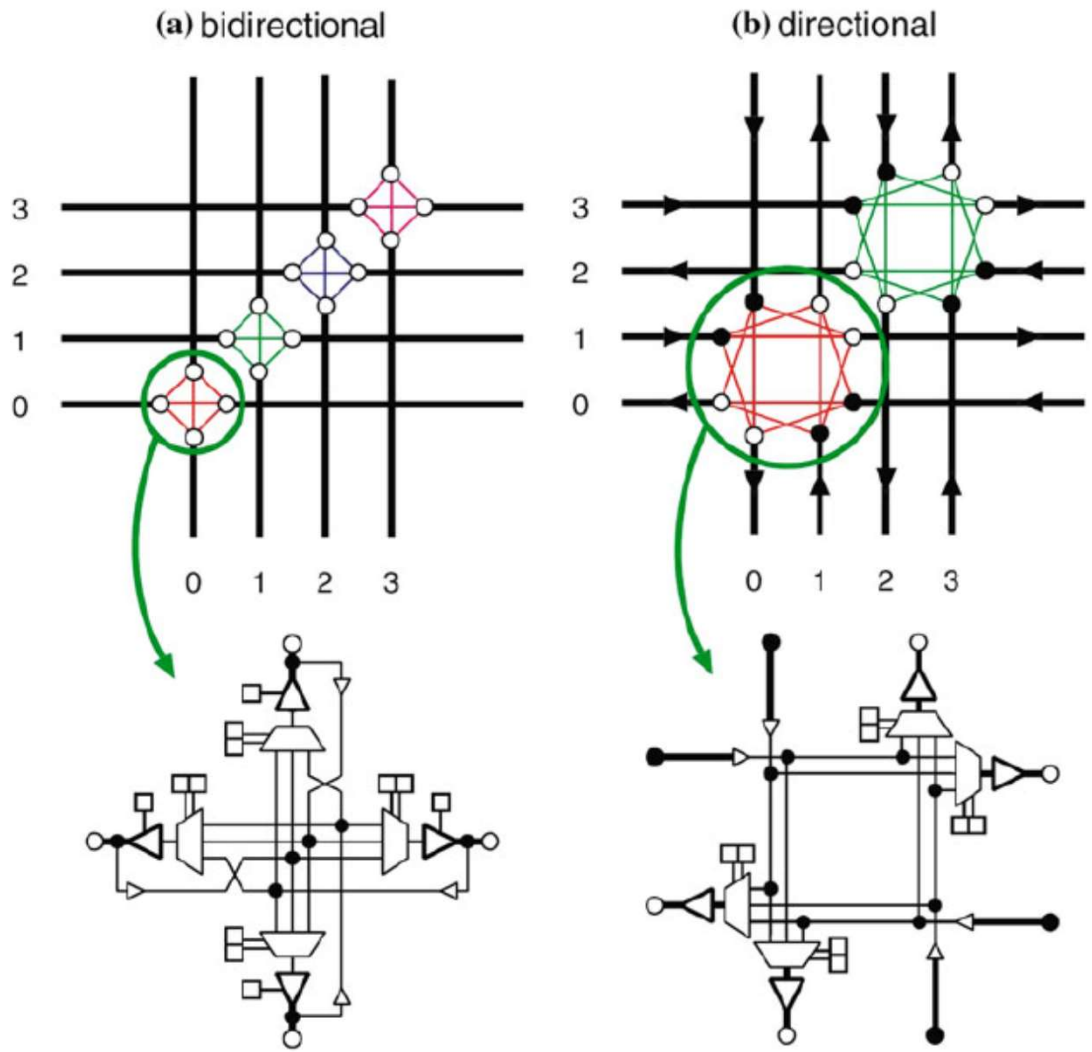
Routing Network Architecture

- **Channel width: is the number of wires in routing channel**
- **Connection boxes (CB): connects Logic blocks** and **routing network**
  - Flexibility of a CB (Fc) is the number of routing wires of adjacent channel which are connected to the pin of a block
    - Fc(in): the connectivity of input pins of logic blocks
    - Fc(out): the connectivity of output pins of logic blocks
- **Switch boxes (SB): connects horizontal and vertical routing wires**
  - Flexibility of a SB (Fs) is the total number of wires which every wire entering in the switch box connects to



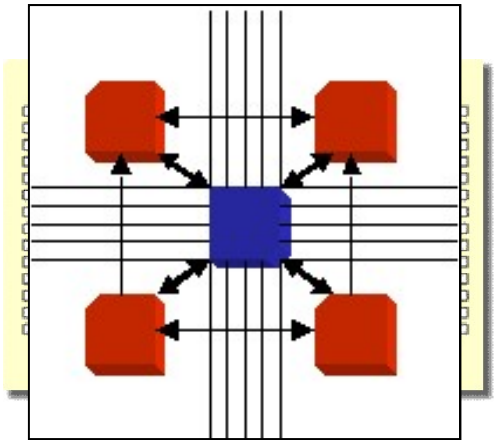
Routing Network Architecture

- **Routing tracks can be bidirectional or unidirectional**
  - Channel width of unidirectional wiring must be in multiples of 2



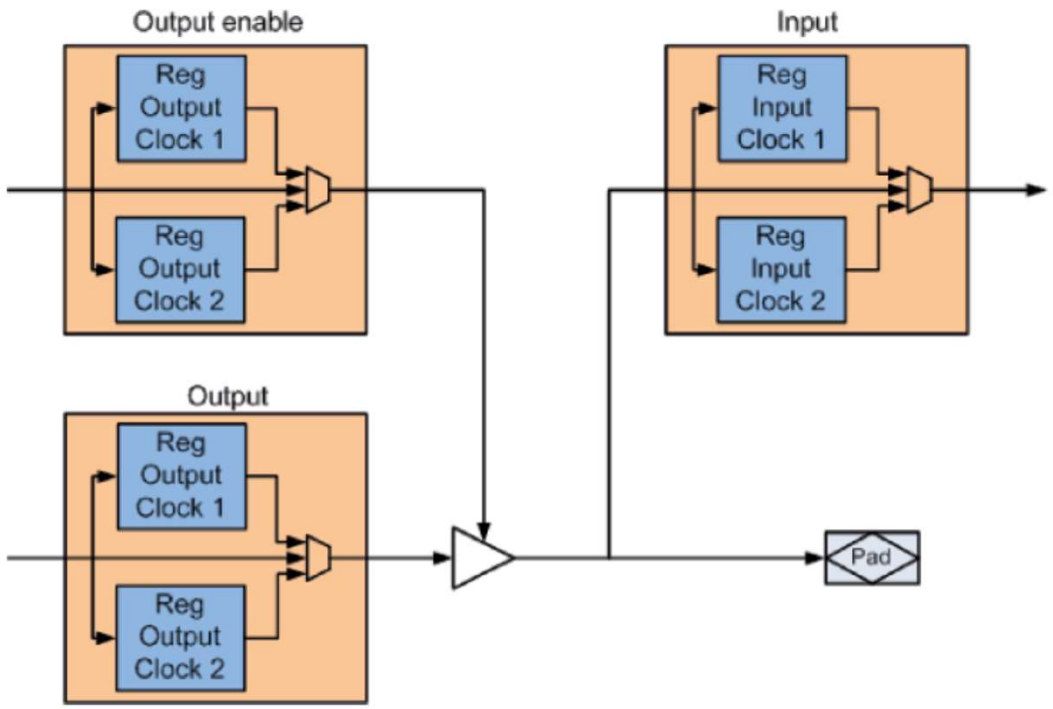
Dedicated Routing

- A combination of programmable and dedicated routing lines
- Dedicated routing
  - Global clocks with predefined clock tree
  - Regional clocks and IO clocks
  - Global low-skew routing resources for other high fan-out signals
  - Carry chain routing
  - Dedicated routing among other dedicated resources
- General interconnect
  - Routing of local signals between CLBs and IOBs



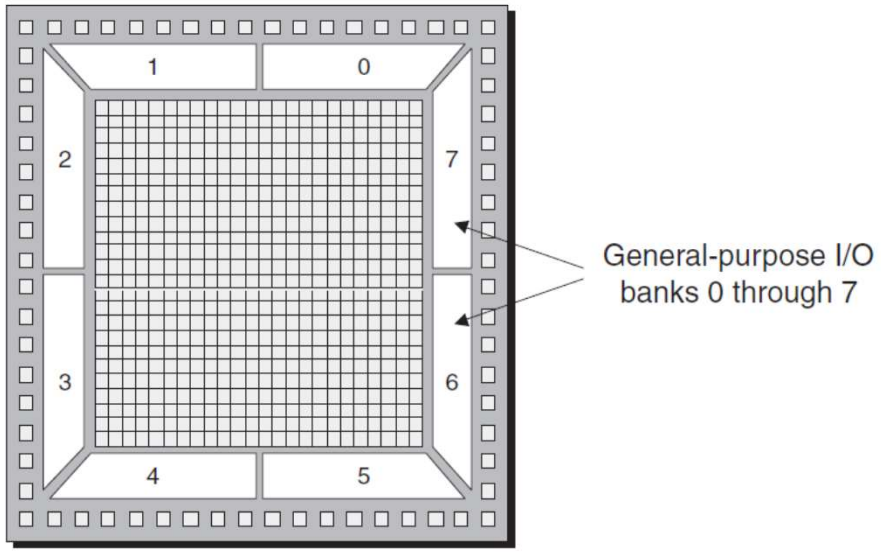
IOB Element

- **Control the flow of data between the I/O pins and the internal logic of the device**
- **Can configure a single interface pin as input, output or bidirectional**
- **Include an input block, an output block and an output enable block**
  - *A pair of Dual-Data Rate (DDR) registers*
- Two operation modes of DDR registers:
  - **Single data rate (SDR): data are copied into the I/O registers on the rising clock edge only**
  - **Double data rate (DDR): data are copied into the I/O registers on both the rising clock edge and falling clock edge**



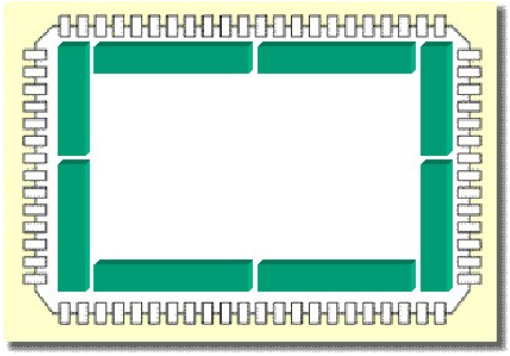
Configurable I/O standards

- **"Standard" refers to electrical aspects of the signals, such as their logic 0 and logic 1 voltage levels**
- **I/O can be configured to accept and generate signals conforming to whichever standard is required**
- **I/O signals will be split into a number of banks, each bank can be configured individually to support a particular I/O standard**
  - allows the FPGA to work with devices using multiple I/O standards
  - allows the FPGA to actually be used to interface (translate) between different I/O standards



I/O Translators

- Programmable input and output thresholds
- Supported standards include
  - LVCMOS (several classes), LVPECL, HSTL (several classes), SSTL (several classes), PCI, PCI-X, LVDS (several classes), GTL, GTL+, and HyperTransport™ (LDT) technology
  - Supported standards vary, check your data sheet



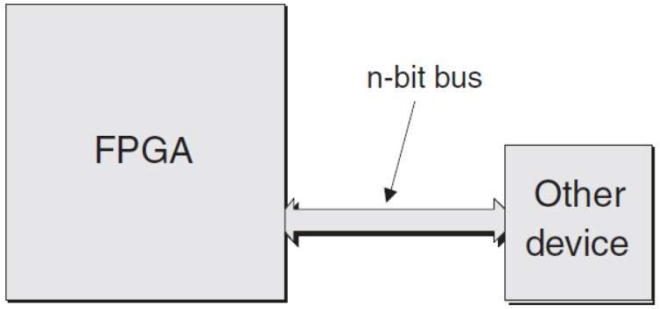
- Different I/O standards require a separate input and output reference voltage for each bank supporting a separate I/O standard
- Generally, each bank can support several standards, as long as they share the same vref (input) or vcco (output)

Dedicated Blocks

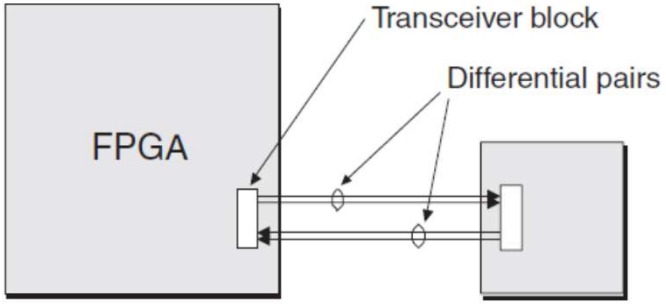
- **Hard IP**
  - **Pre-implemented hardware blocks** such as microprocessor cores, gigabit interfaces, multipliers, adders, MAC functions etc.
  - Designed to be as **efficient as possible in terms of power consumption, silicon area, and performance**
- **Soft IP:**
  - **source-level library of high-level functions** in a hardware description language, or HDL, such as Verilog or VHDL at the *register transfer level (RTL)* of abstraction
- **Firm IP:**
  - **a library of high-level functions in *netlist* (**i.e. these functions have **already been optimally mapped, placed, and routed** into a group of programmable logic blocks)

Gigabit transceivers

- **Motivated by the problems of the bus-based communication:**
  - requires a lot of pins on the device
  - Difficult to ensure the same length and impedance
  - difficult to manage signal integrity issues
- **Special hard-wired transceiver blocks**
- **Use one pair of differential signals to transmit (TX) data and another pair to receive (RX) data**
- **Can transmit and receive billions of bits of data per second**



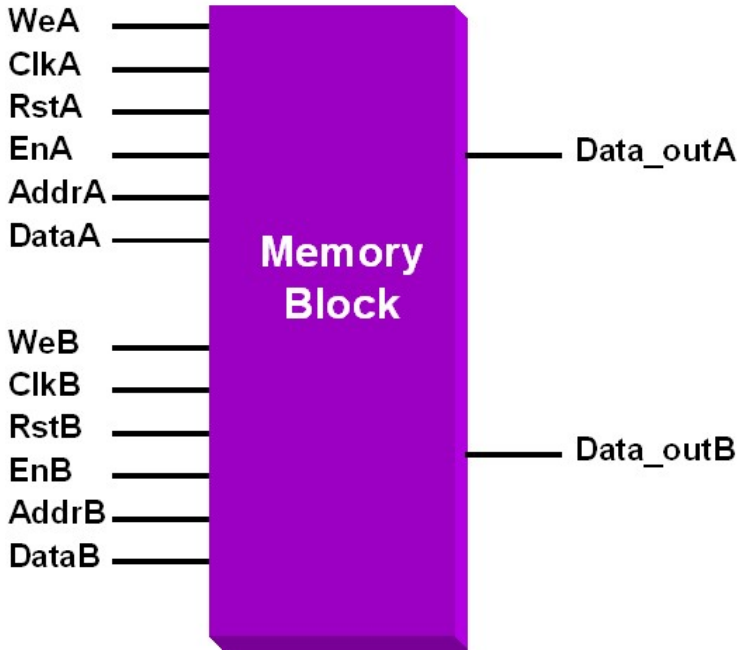
Using a bus to communicate between devices



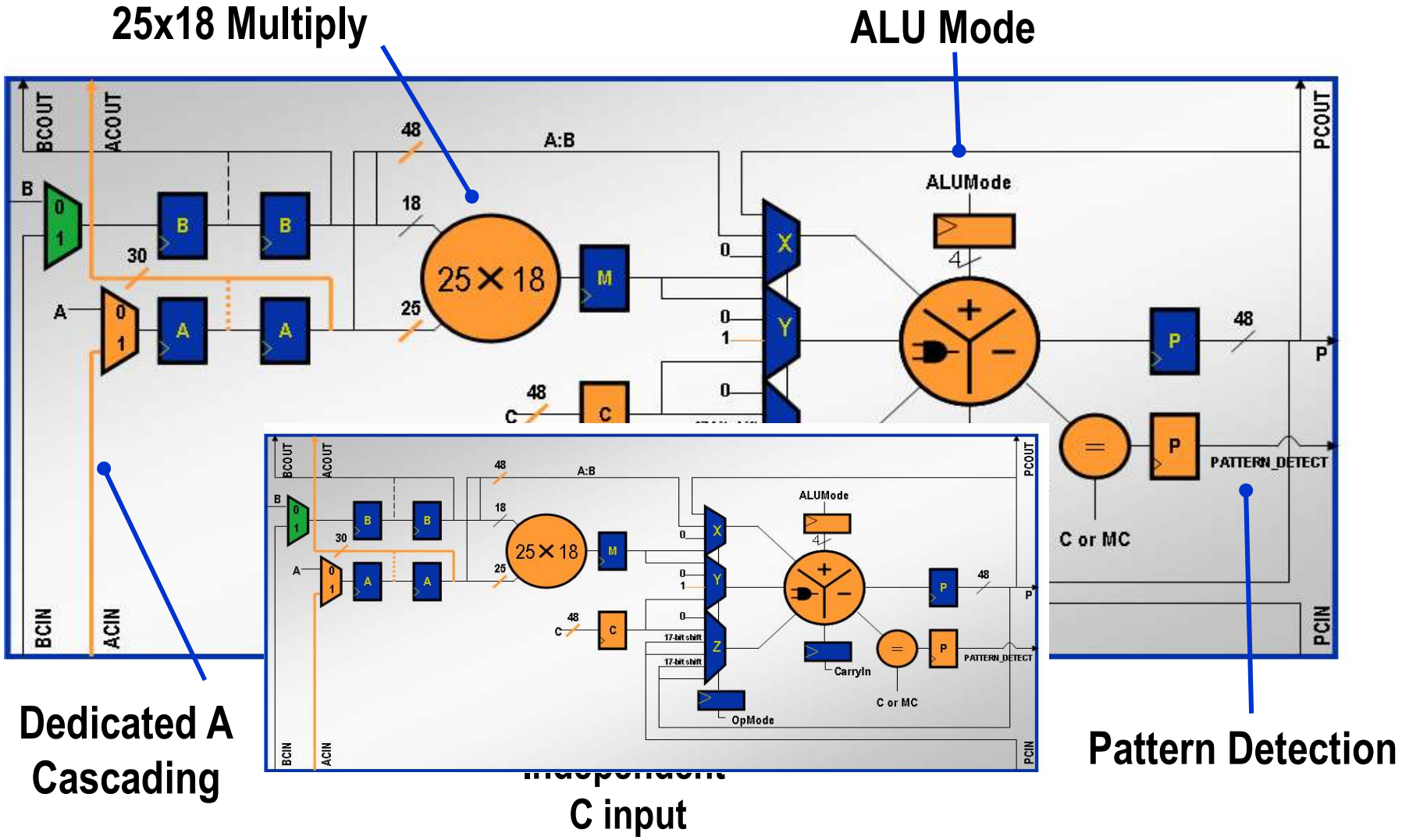
Using high-speed transceivers to communicate between devices

Memory Blocks

- Support **single- and dual-port synchronous operations**
- In dual-port mode, these RAM blocks support fully independent ports for both reading and writing
- Each block of RAM can be used independently, or multiple blocks can be combined together to implement larger blocks by dedicated cascade logic
- Blocks of memory are generally spread out across the die
- Dedicated FIFO logic enables each RAM to be configured as a FIFO
- Contain from **tens to hundreds of these RAM blocks**
  - Total storage capacity of a few hundred thousand bits up to several million bits



Specific Purpose Hard Blocks: XILINX DSP SLICE



Specific Purpose Hard Blocks: Microprocessor cores

- Easy to build on-chip computing systems:
  - saves the cost **of having two devices (FPGA and microprocessor)**
  - eliminates large numbers of tracks, pads, and pins on the circuit board
  - makes the board smaller and lighter
- Existing in two formats:
  - Hard core: e.g. Xilinx PowerPC, ARM Cortex-A,…
  - Soft core: e.g. Xilinx MicroBlaze

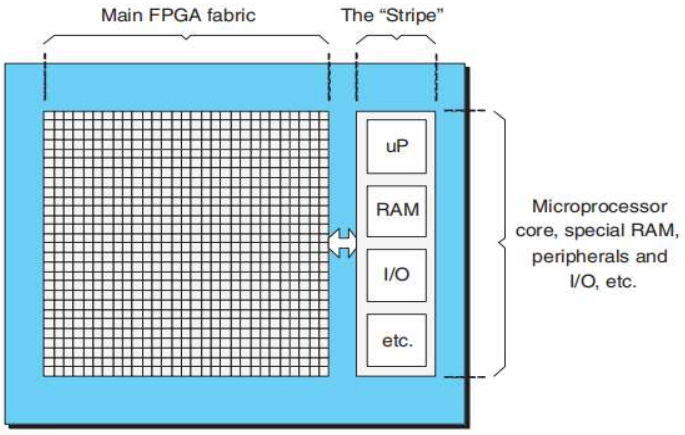
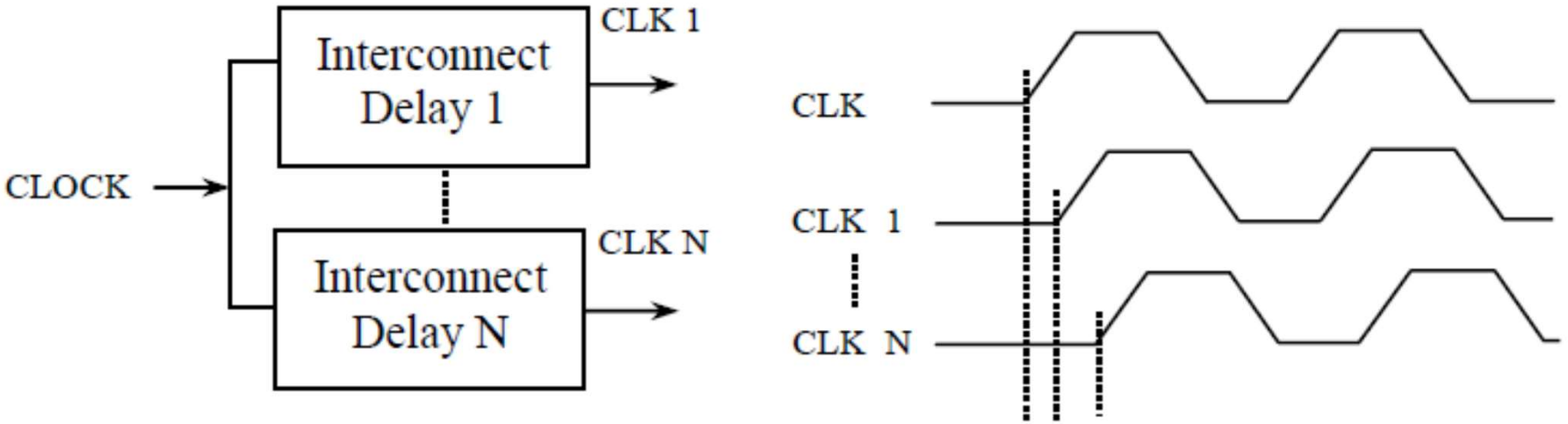


FIGURE 2-13 Bird's-eye view of chip with embedded core outside of the main fabric.

Clock Management

Skew:

- **refers to the rising edge** (or the falling edge) of **a clock arriving at different times at register clock inputs in a synchronous sequential circuit**
- **results in missing the data** at **high frequency**

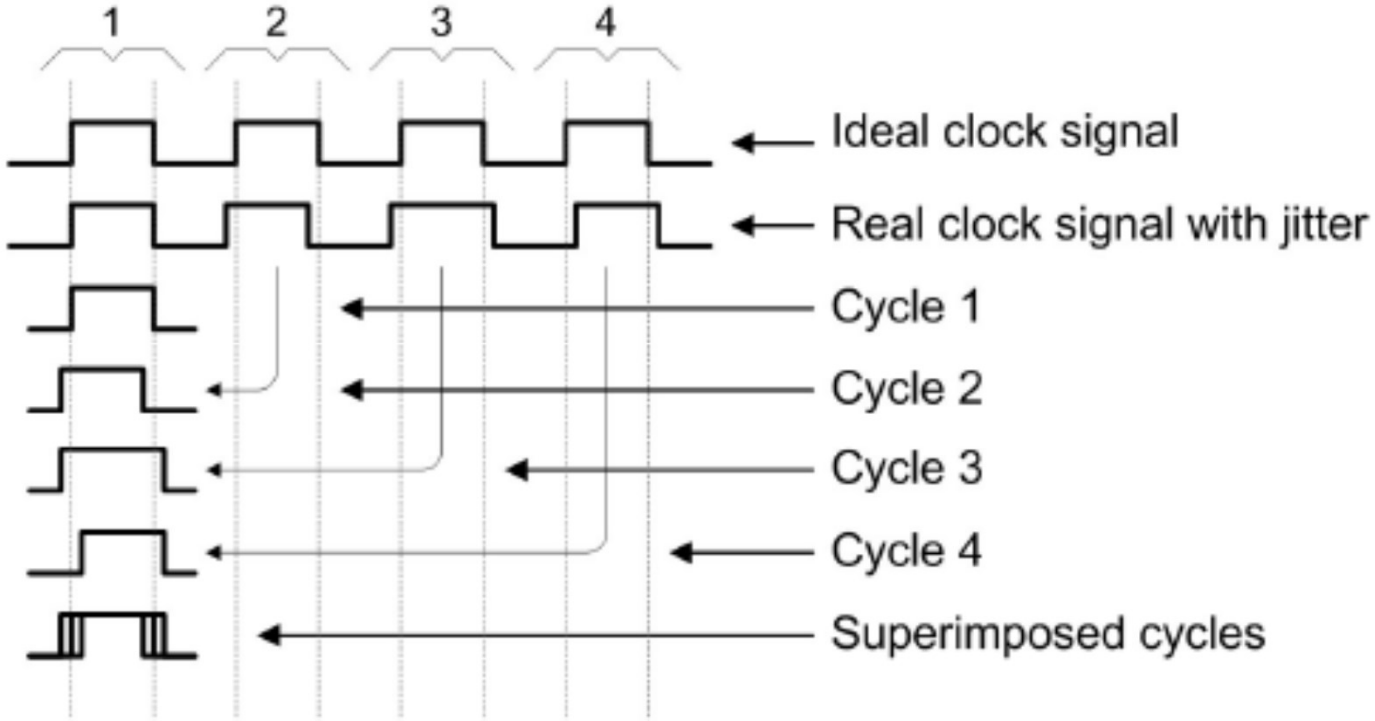


Clock Management

Jitter:

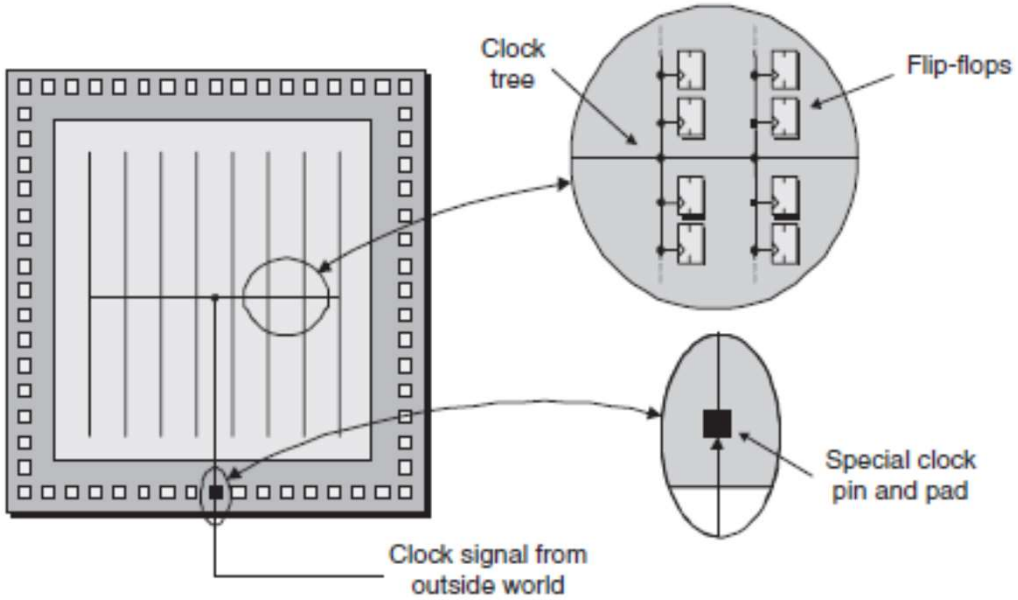
- clock edges may arrive a little early or a little late

- if superimpose multiple edges on top of each other; the result would be a "fuzzy" clock



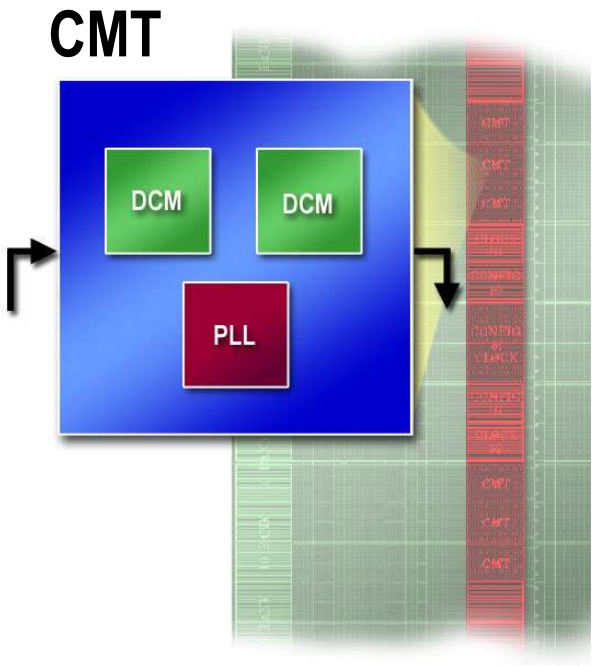
Clock Management

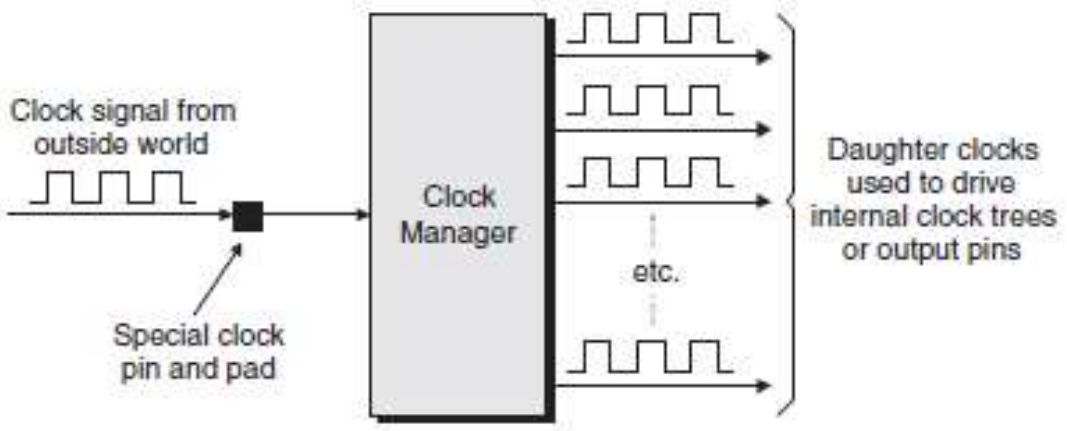
- **Dedicated clock trees are pre-optimized clock networks that balance the skew, and minimize delay**
- **Using special tracks and is separate from the general-purpose programmable interconnect**
  - Virtex-5 FPGA has 32 separate clock networks
  - Spartan™-3 FPGA has 8 separate clock networks
    - Each can be configured for a built-in clock enable (BUFGCE) or switching clock sources (BUFGMUX)



Clock Managers

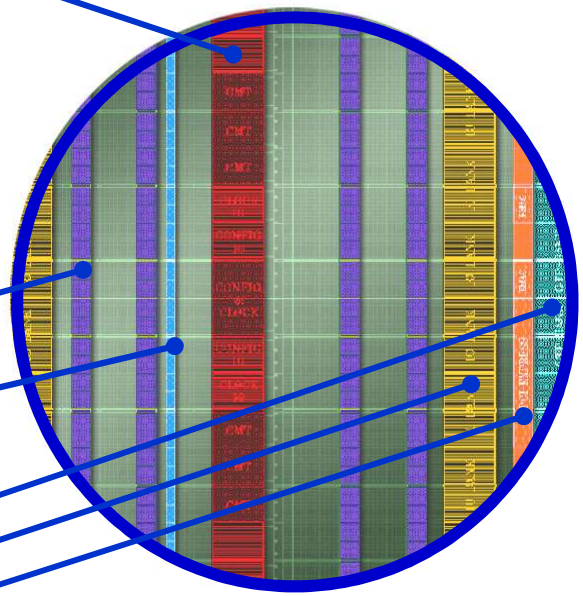
- **Generates a number of daughter clocks**
  - ❖ **PLL (Phase Lock Loop)**
  - – synthesizing clock frequencies
  - – reducing clock jitter
  - ❖ **Digital Clock Manager (DCM):**
  - – generating clock frequencies,
  - – correcting clock duty cycles, and phase shifting clocks
  - ❖ **DCM consists of**
  - – Digital Delay Locked Loop (DLL)
  - – Digital Frequency Synthesis (DFS
  - – Digital Phase Shifter (DPS)





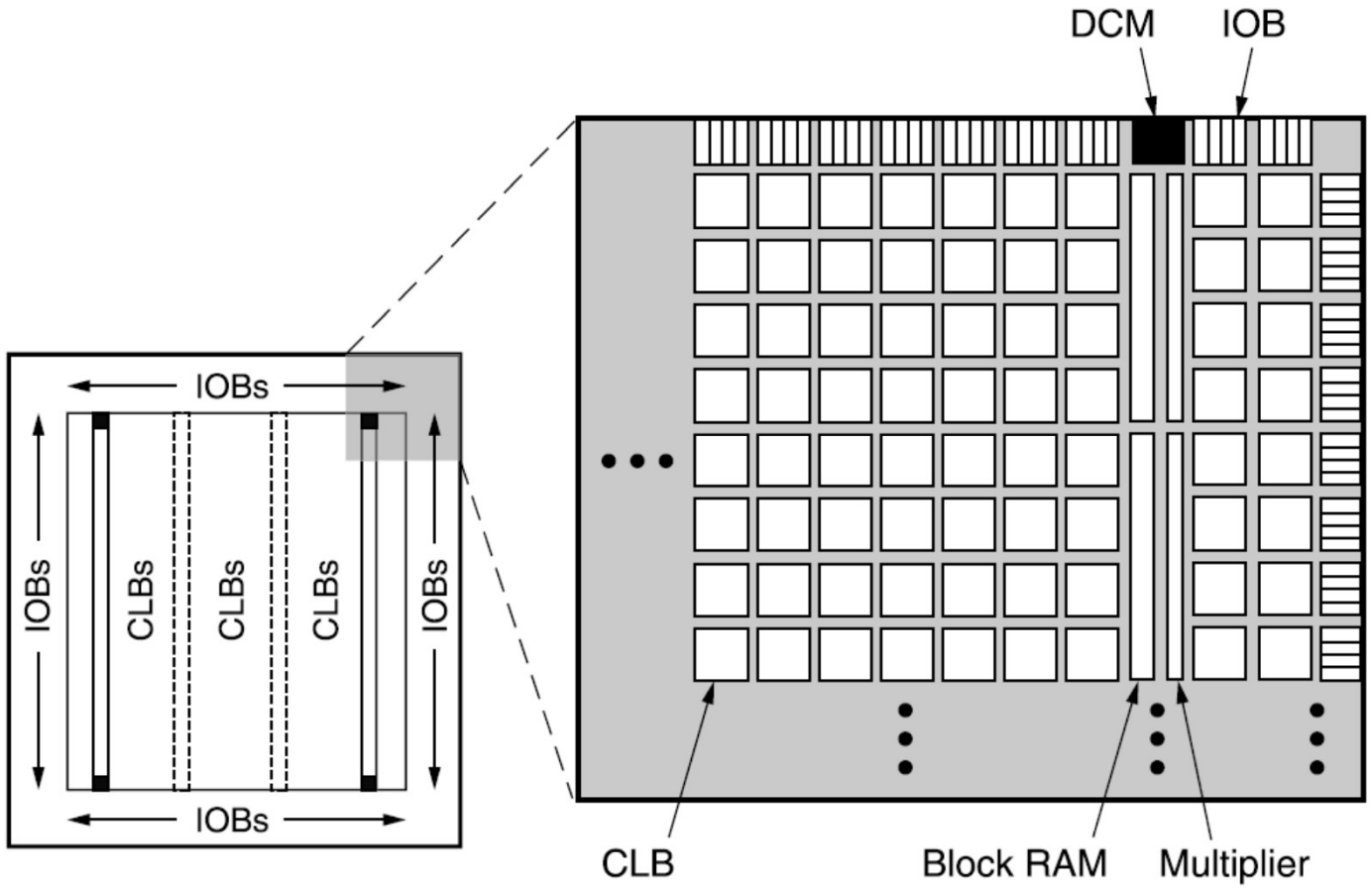
Dedicated and Special Resources

- **Clock management (CMT)**
  - DCM and PLL
  - Dedicated clock trees (not shown)
- **Test logic**
  - Built-in JTAG
- **I/O translators**
  - Supporting many different thresholds
- **Other resources**
  - Dual-Data Rate (DDR) registers in IOB
  - SERDES resources
- **Dedicated Cores**
  - Block RAM
  - DSP Slices
  - Gigabit transceivers, MGTs (all devices)
  - Tri-mode Ethernet MAC (all devices)
  - PCI Express® core (all devices)
- **Additional FXT Cores**
  - PowerPC® 440 processors (not shown)
  - Faster GTX transceiver (not shown)



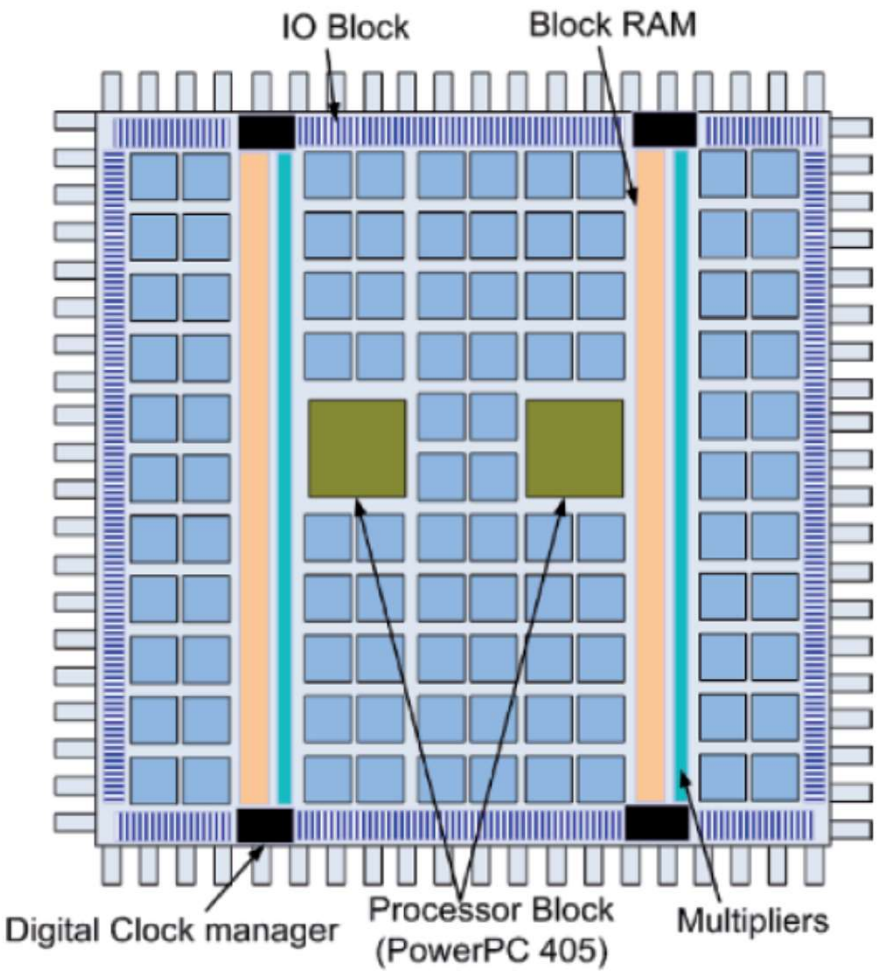
The dedicated resources for Virtex-5

EXAMPLES



Spartan-3 Family Architecture

EXAMPLES



Structure of a Xilinx Virtex II Pro FPGA with two PowerPC 405 Processor blocks

EXAMPLES (here)

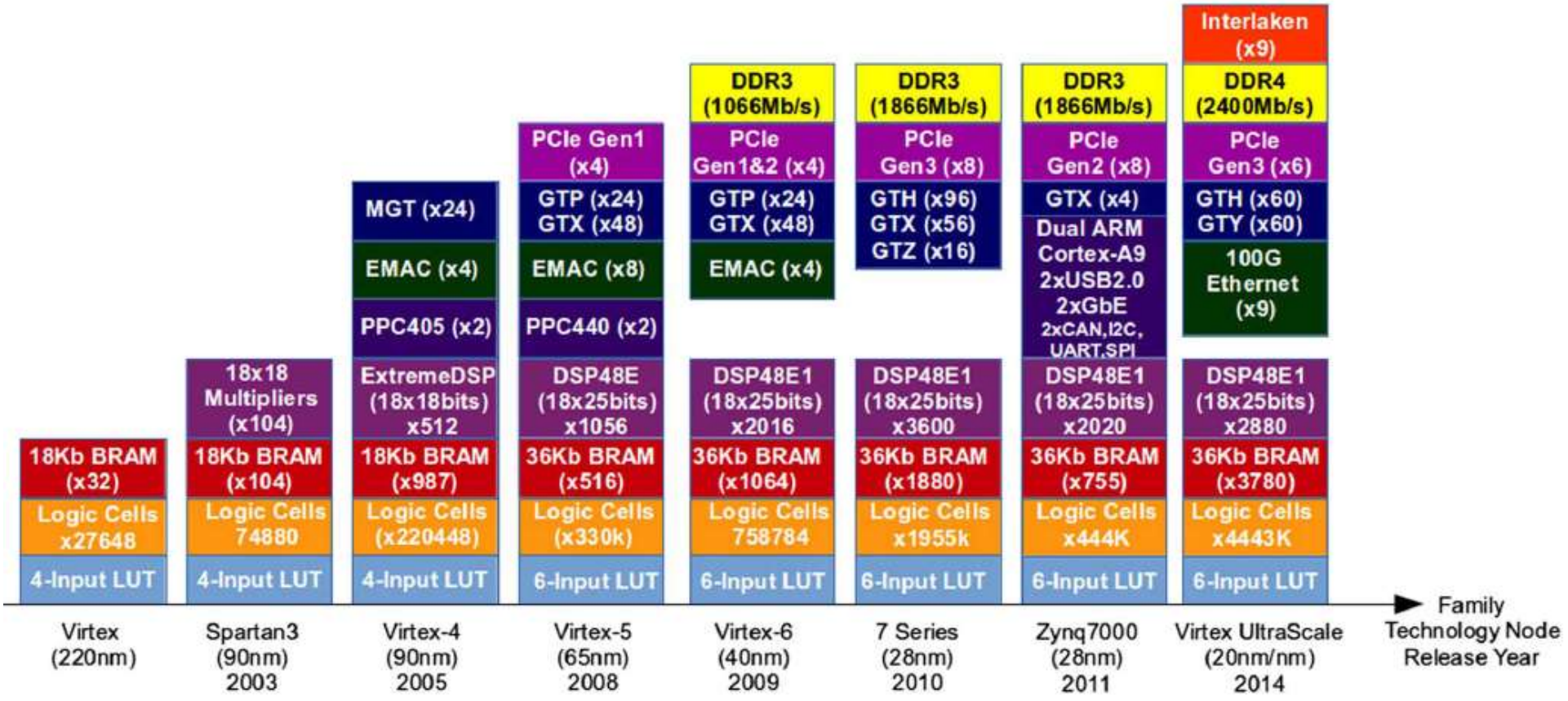
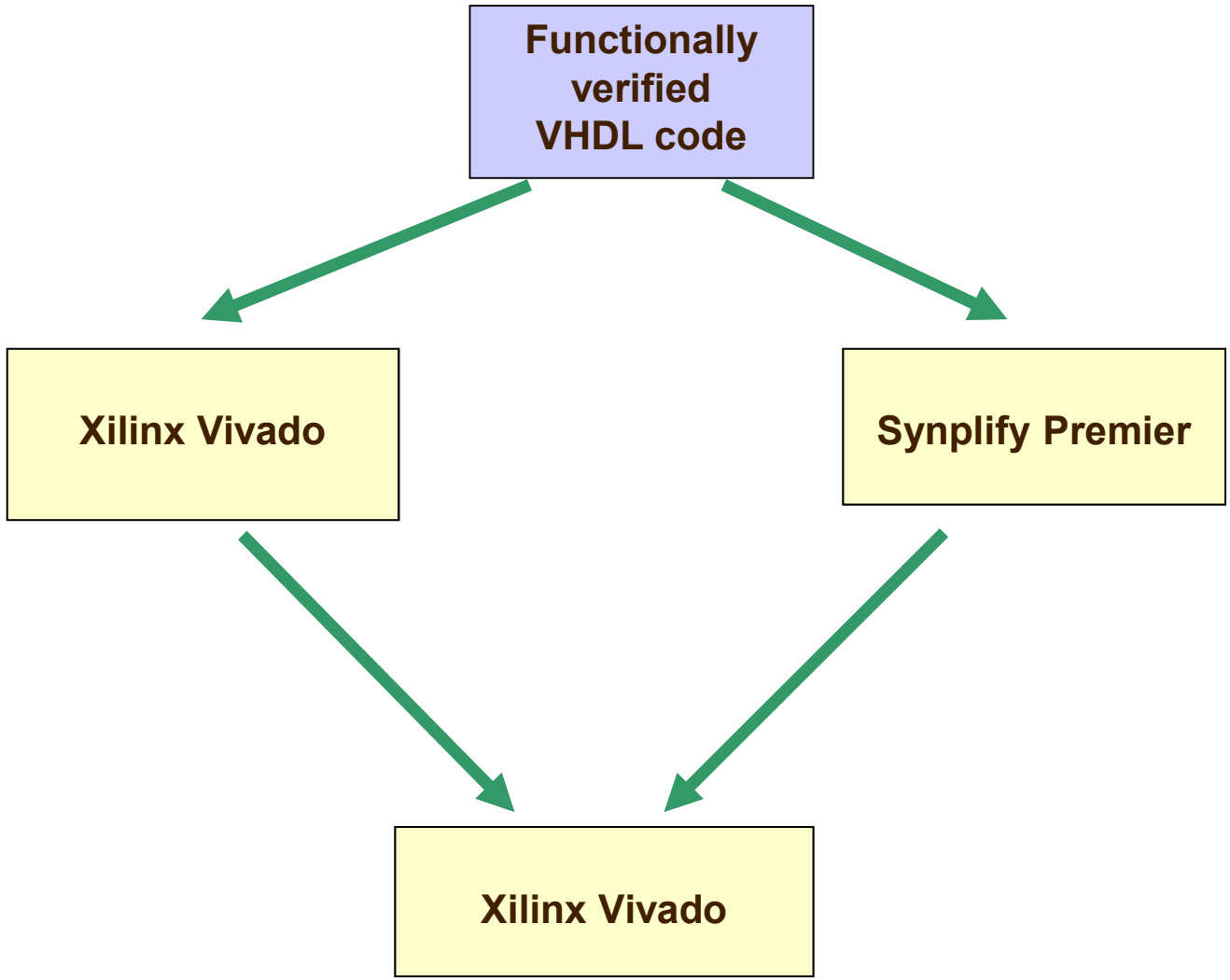
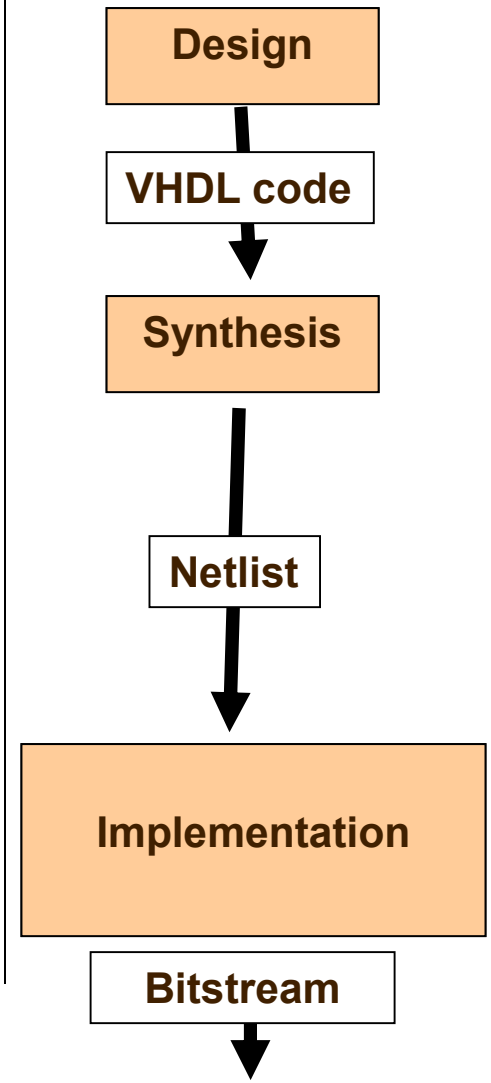


Fig. **Evolution of number and complexity of IP blocks included in FPGAs.**

Tools in Xilinx FPGA Design Flow





FPGA Design process (1)

Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

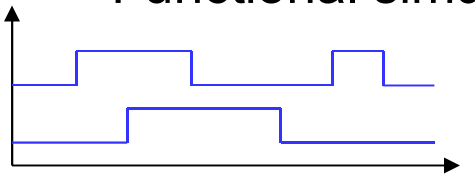
Specification / Pseudocode

On-paper hardware design
(Block diagram & ASM chart)

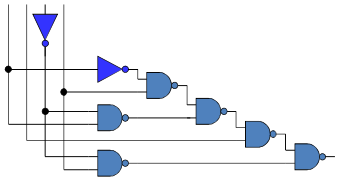
VHDL description (Your Source Files)

```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
            clock, reset, encr_decr: in std_logic;
            data_input: in std_logic_vector(31 downto 0);
            data_output: out std_logic_vector(31 downto 0);
            out_full: in std_logic;
            key_input: in std_logic_vector(31 downto 0);
            key_read: out std_logic;
        );
end AES_core;
```

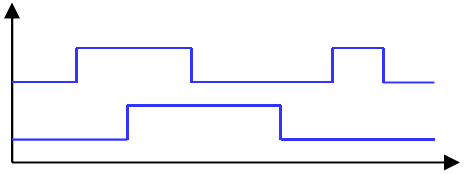
Functional simulation



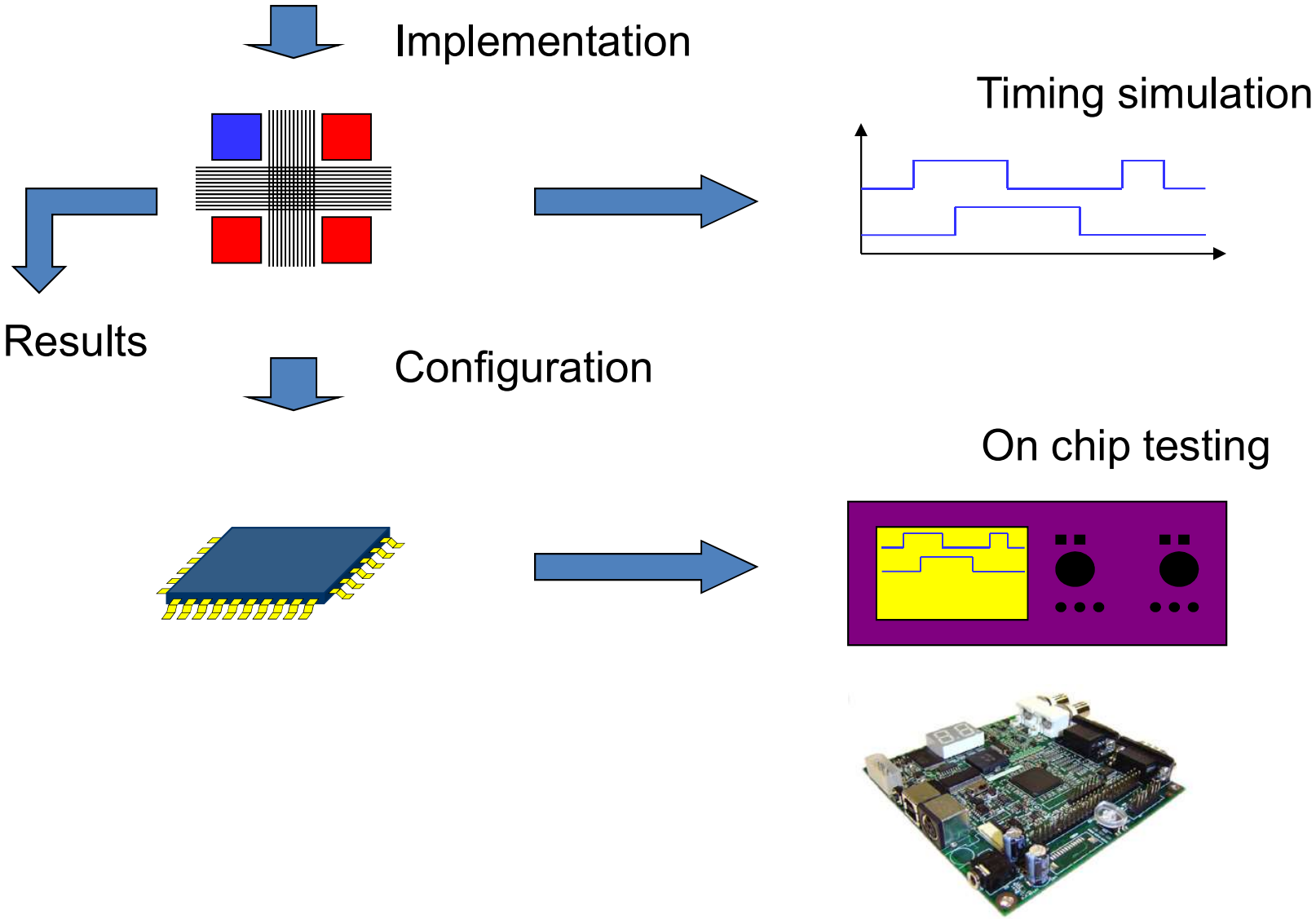
Synthesis



Post-synthesis simulation



FPGA Design process (2)



VietNam National University
University of Engineering and Technology

Synthesis

Laboratory for Smart Integrated Systems

Synthesis Tools

Vivado Design Suite

Synplify Premier

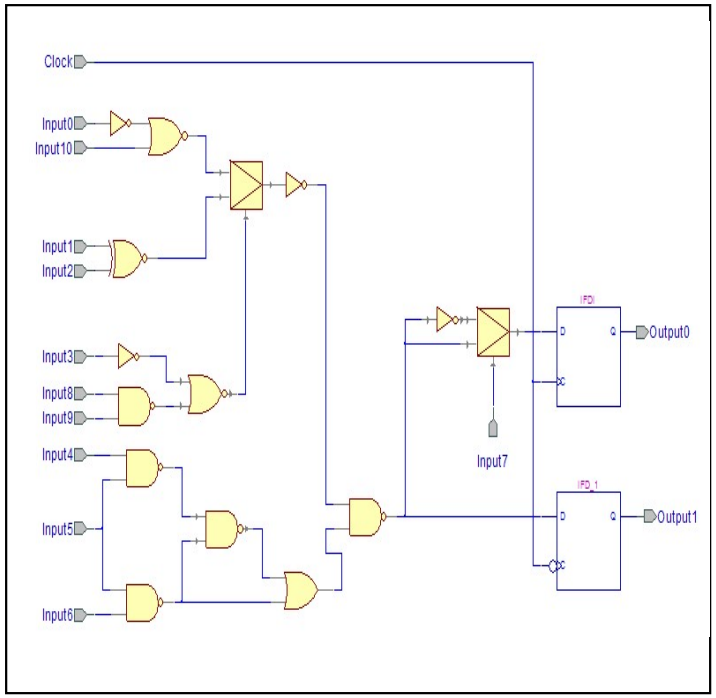
… and others

Logic Synthesis

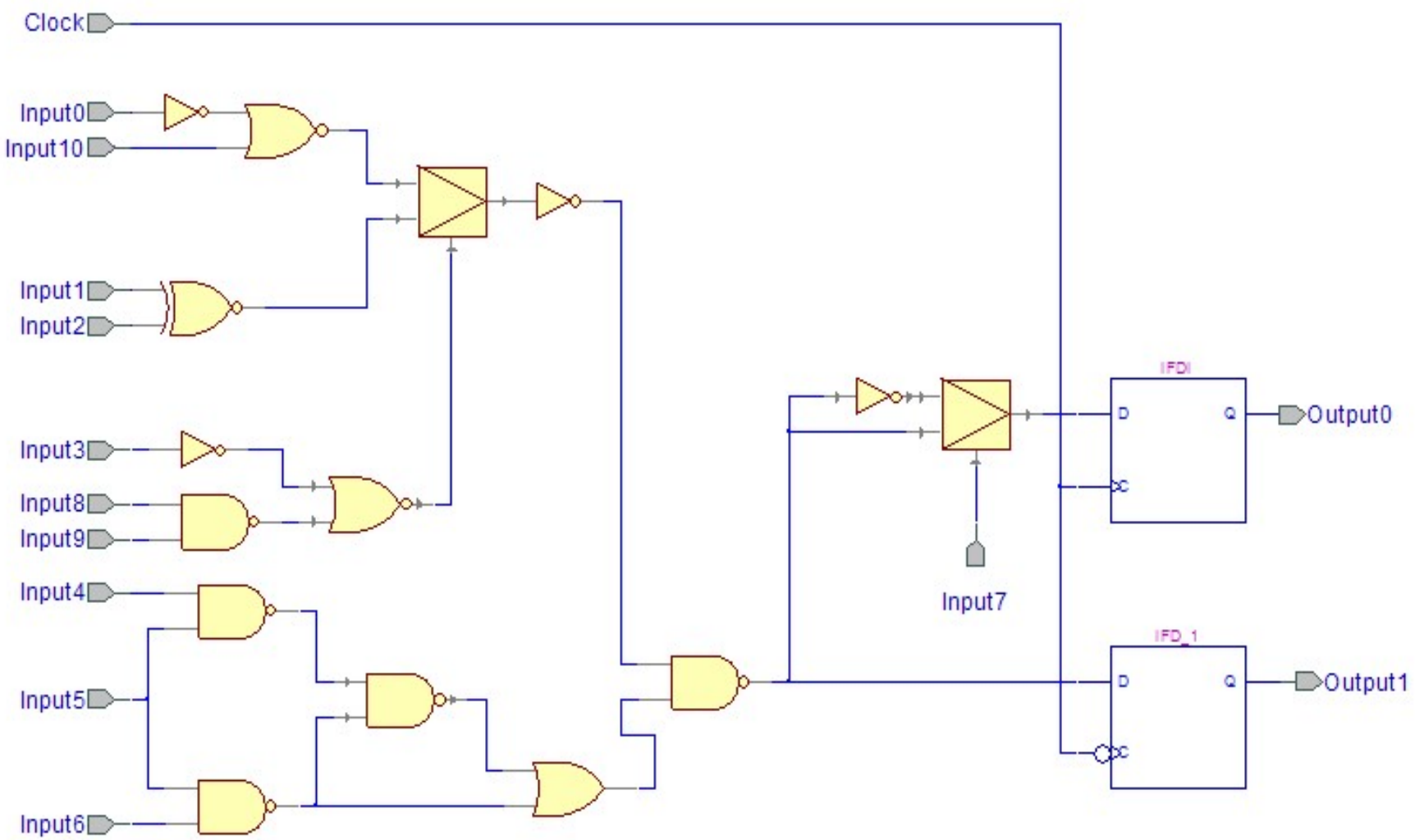
VHDL description

```
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
                A1<=A when (NEG_A='0') else
                                not A;
                B1<=B when (NEG_B='0') else
                                not B;
                Y<=Y1 when (NEG_Y='0') else
                                not Y1;

                MUX_0<=A1 and B1;
                MUX_1<=A1 or B1;
                MUX_2<=A1 xor B1;
                MUX_3<=A1 xnor B1;

                with (L1 & L0) select
                                Y1<=MUX_0 when "00",
                                                MUX_1 when "01",
                                                MUX_2 when "10",
                                                MUX_3 when others;

end MLU_DATAFLOW;
```

Circuit netlist



Circuit netlist (RTL view)

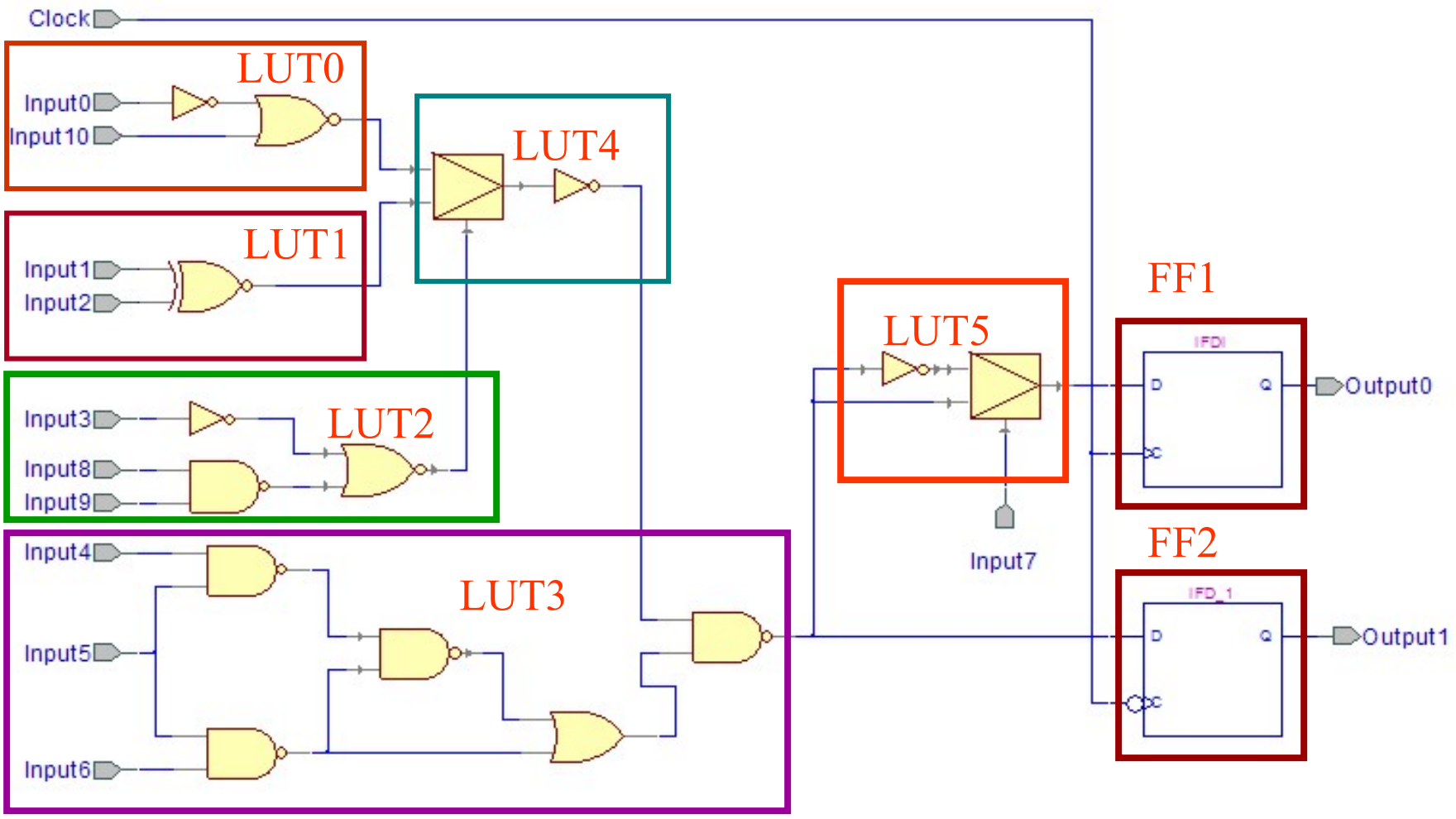


Implementation

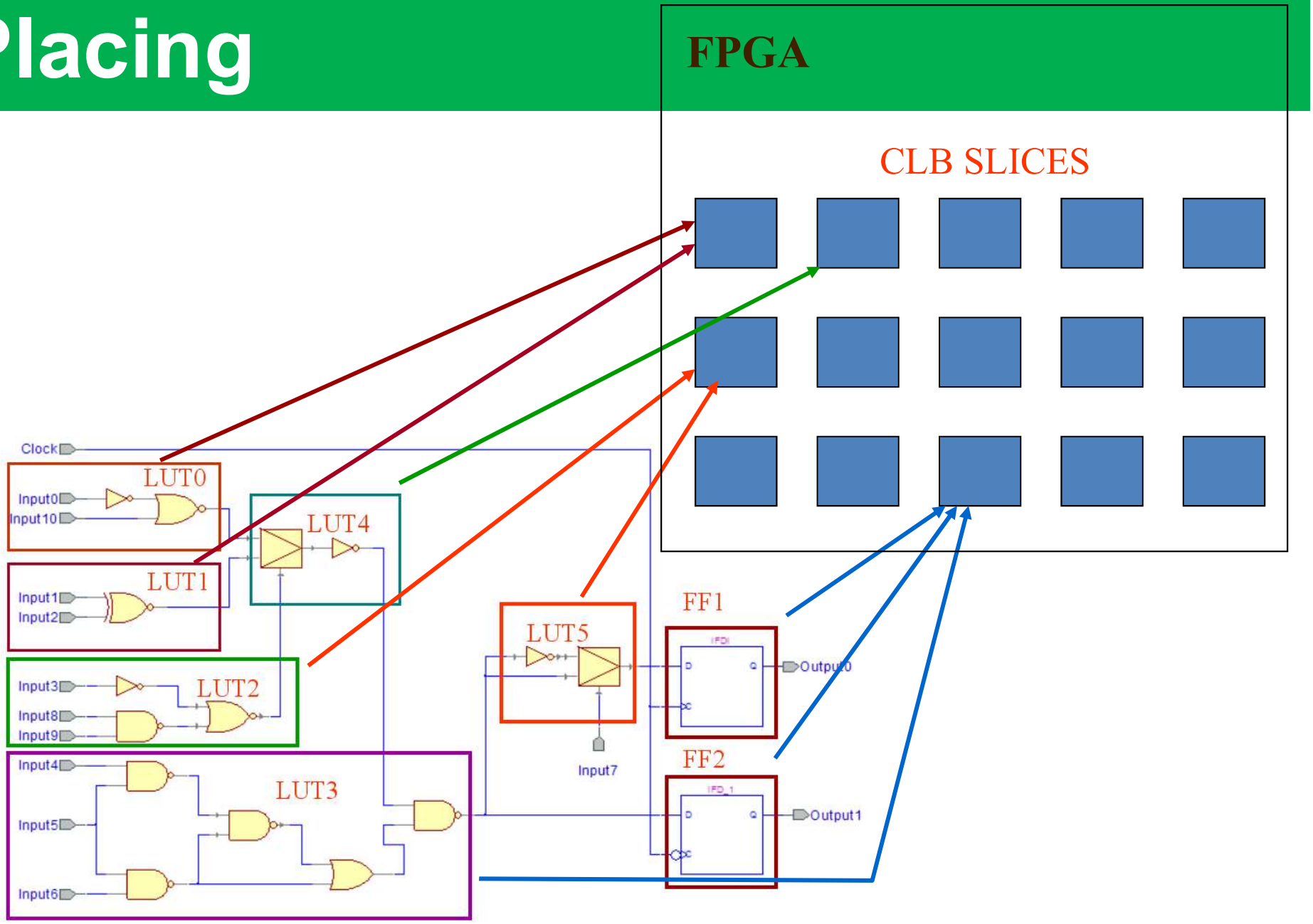
Implementation

- After synthesis the entire implementation process is performed by FPGA vendor tools

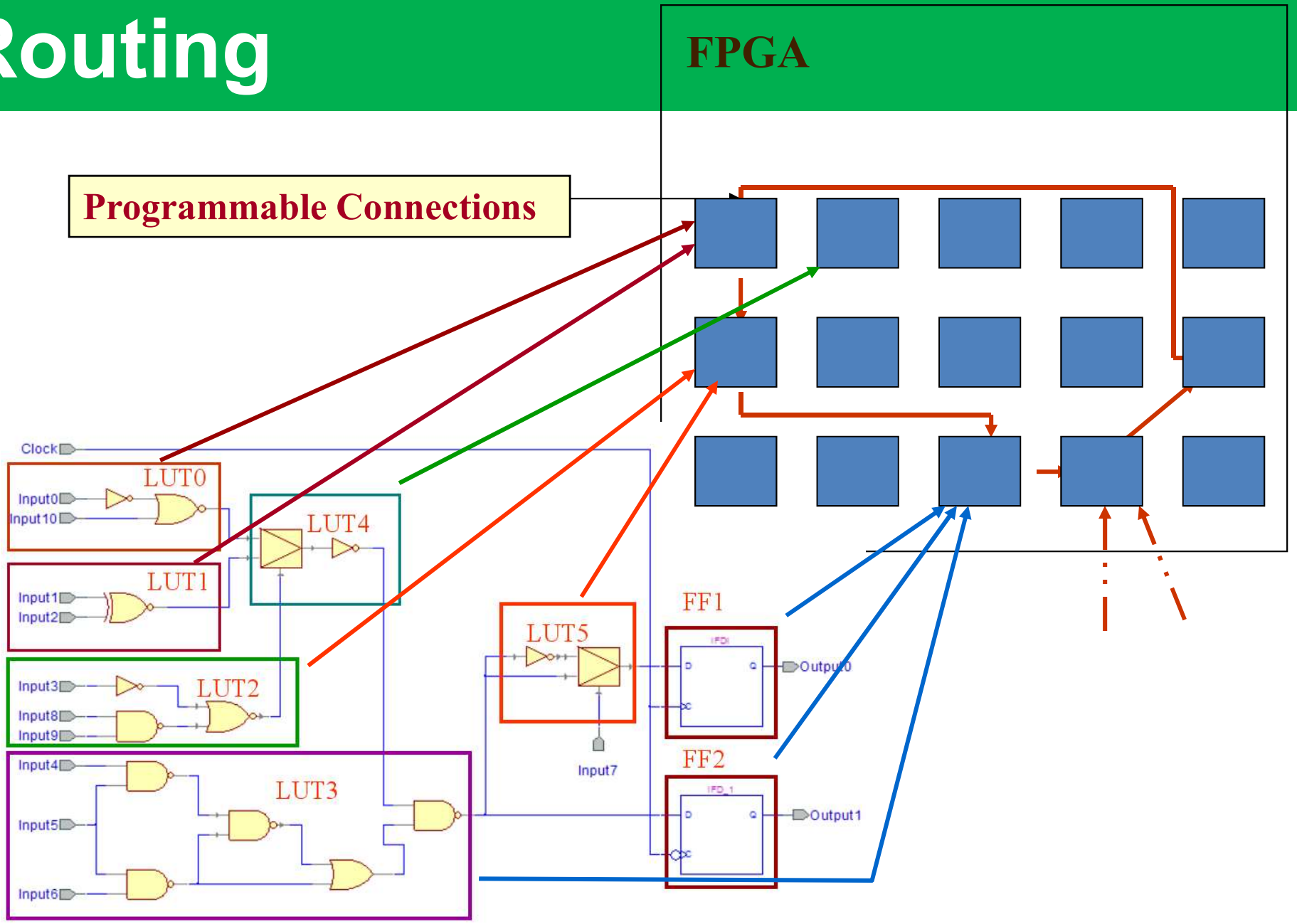
Mapping


Clock
LUT0
Input0
Input10
LUT1
Input1
Input2
LUT4
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT5
Input7
FF1
IFDI
D
Q
Output0
FF2
IFD_1
D
Q
Output1

Placing


FPGA
CLB SLICES
Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT5
Input7
FF1
FF2
Output0
Output1

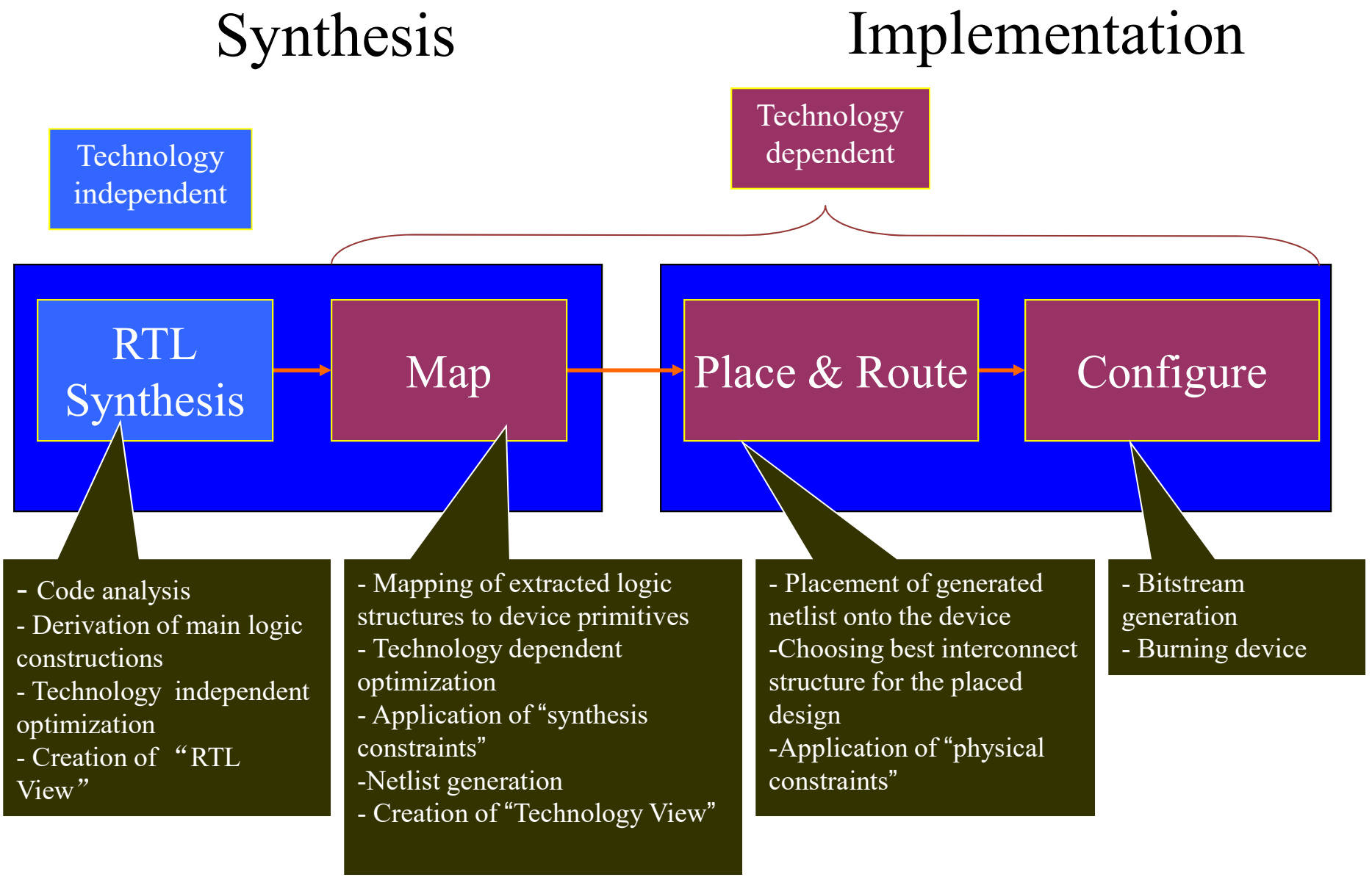
Routing


FPGA
Programmable Connections
Clock
LUT0
Input0
Input10
LUT1
Input1
Input2
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT4
LUT5
Input7
FF1
FF2
Output0
Output1

Configuration

- Once a design is implemented, you must create a file that the FPGA can understand
  - This file is called a bit stream: a BIT file (.bit extension)
- The BIT file can be downloaded directly to the FPGA, or can be converted into a PROM file which stores the programming ir

Two main stages of the FPGA Design Flow



Vivado Design Suite

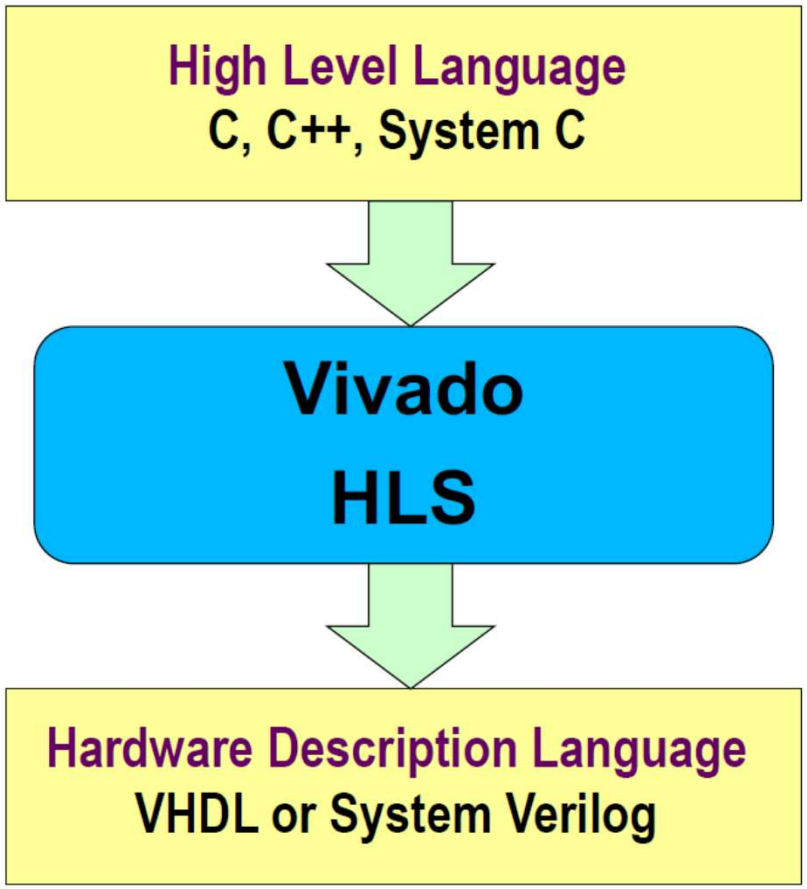
- 4 years of development and 1 year of beta testing
- first version released in Summer 2012
- scalable data model, supporting designs with up to 100 million ASIC gate equivalents (GEs)
- based on industry standards, such as
  - AMBA AXI4 interconnect
  - IP-XACT IP packaging metadata
  - Tool Command Language (Tcl)
  - Synopsys Design Constraints (SDC)

Vivado Design Suite

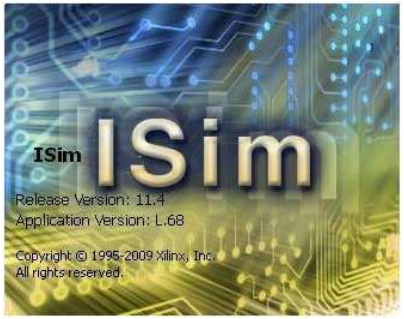
- Design Entry Methods
  - VHDL
  - Verilog
  - System Verilog
  - C, C++, System C
  - Matlab
  - Simulink

Vivado Design Suite

- **High-Level Synthesis**



Additional Simulation Tools



Active-HDL™
Complete FPGA Verification Environment

ModelSim®

Additional Simulation Tools

ModelSim:

- Industry standard for simulation
- Significantly faster than Vivado Simulator
- Windows, Linux OS
- Mixed-language support: VHDL, Verilog, System Verilog
- Recommended for advanced users and more complex designs
- To be used primarily as a standalone tool for functional simulation

- Features of the Starter Edition:
  - Free, no license required
  - 10,000 executable line limit

Summary

- Concepts and applications of FPGA
- FPGA architecture
  - Configurable Logic Block
  - Routing Network Architecture
  - Clock Management
- FPGA Design flow