FPGA Xpert

Introduction To Digital Design

OUTLINE

- **FPGA Expert Course Overview.**
- **Projects Overview .**
- **Basic’s Of Digital Design’s.**
- **Design Flow Overview .**
- **Summary**

FPGA Expert Course Overview

FPGA Expert Course Overview

Session # 1

Digital Design Fundamentals .

- Basics of Digital Design , with FPGA Oriented .
- Design Flow .
- Using HDL .

Design Methodology

- How to Partition The design.
- How to Give Names To signals.
- Technology Specific Code .
- How To choose Devices

FPGA Expert Course Overview

Session # 2

VHDL .

- VHDL Language Concepts
- Test Benches
- Signal & Data Type
- Operation & expressions
- Finites State Machine
- Concurrent & sequential Statements
- Advanced issue : Working with files , generic &generate
- And lot more

FPGA Expert Course Overview

Session # 3

EDA Tools

- The Most advanced Tools for Digital Design Developments .
- How To use those tools to get more performance to our design .
- Tools for Design verifications .

Main Tools that We will cover in the course :

- Vivado – Xilinx (Simulator , Synthesize , P&R , ChipScop).

FPGA Expert Course Overview

Session # 4

FPGA For Designers .

- Basic FPGA architecture.
- Reading Reports .
- Building Clocks Tree .
- Global Timing Constrain.

FPGA Expert Course Overview

Session # 4

Designing for Performance

- Understanding the FPGA resources
- Understanding Clock Resources
- Reading Timing Reports
- Path Specific Timing Constrains
- Power Estimation for FPGA designs

Advanced FPGA Implementation

- Implement Design Via TCL Command Line
- Edit a User Constraint File (XCF)
- Design Preservation Techniques
- Design Placement
- Timing Constrains (Specific + global)

FPGA Expert Course Overview

Session # 5

Board Design & Signal Integrity

- Board development Process
- Signal Integrity For High Speed Board Designs
- Verification Of Board Design
- Tools and Development Board Design

FPGA Expert Course Overview

Session # 6

Project

- Class Sessions with the students .
- Design Presentation .

Projects Overview

Projects Overview -Tasks

- The Project will include all the flowing :
  - The student will receive specification of the project.
  - The student will build the Micro-Architecture for the project.
  - The student will write the HDL code .
  - The student will build Test Bench To simulate the design.
  - The student will implement the design trough the synthesis ->P&R-> on board verification.
  - The student will verify the code on the chip.
  - The student will present the Design to the class.

Projects Overview-schedule

- Working Plan for the project :
  - The students will received the projects after the first session of the language (about 1 month from the beginning) .
  - Till the end of the course the student will work on the project , on his free time .
  - For each student we will give guidance , for any question or help if necessary .
  - The student will received a board and the necessary software for the implementation.
  - The student will present the project to the class at the end of the course .

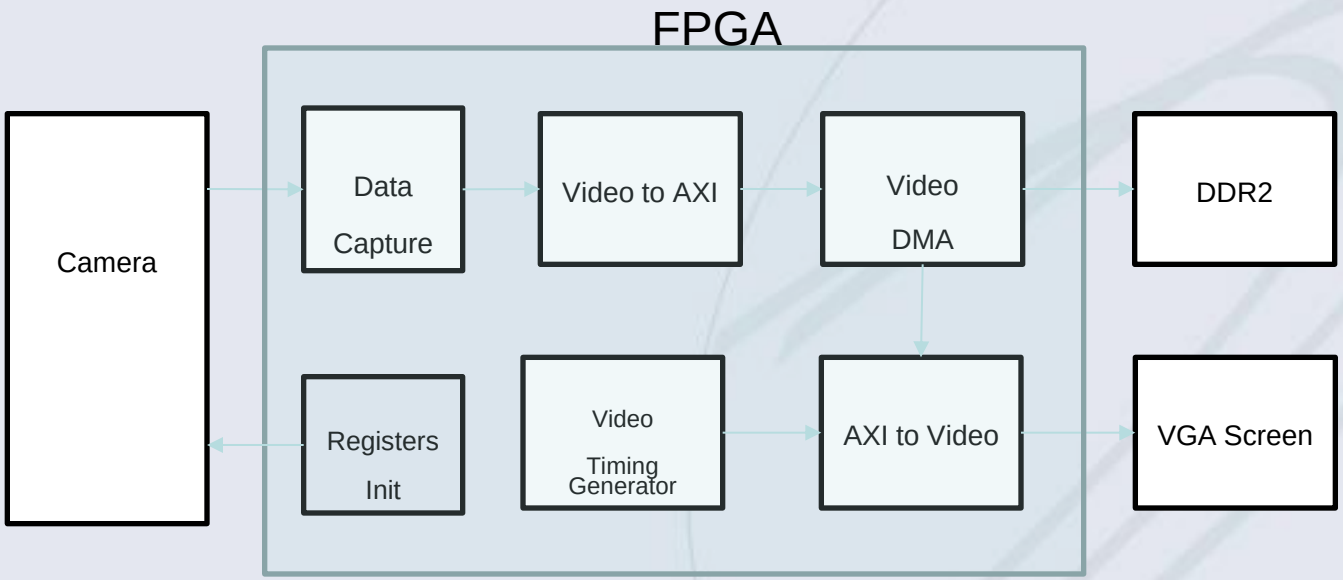
Project example

Description

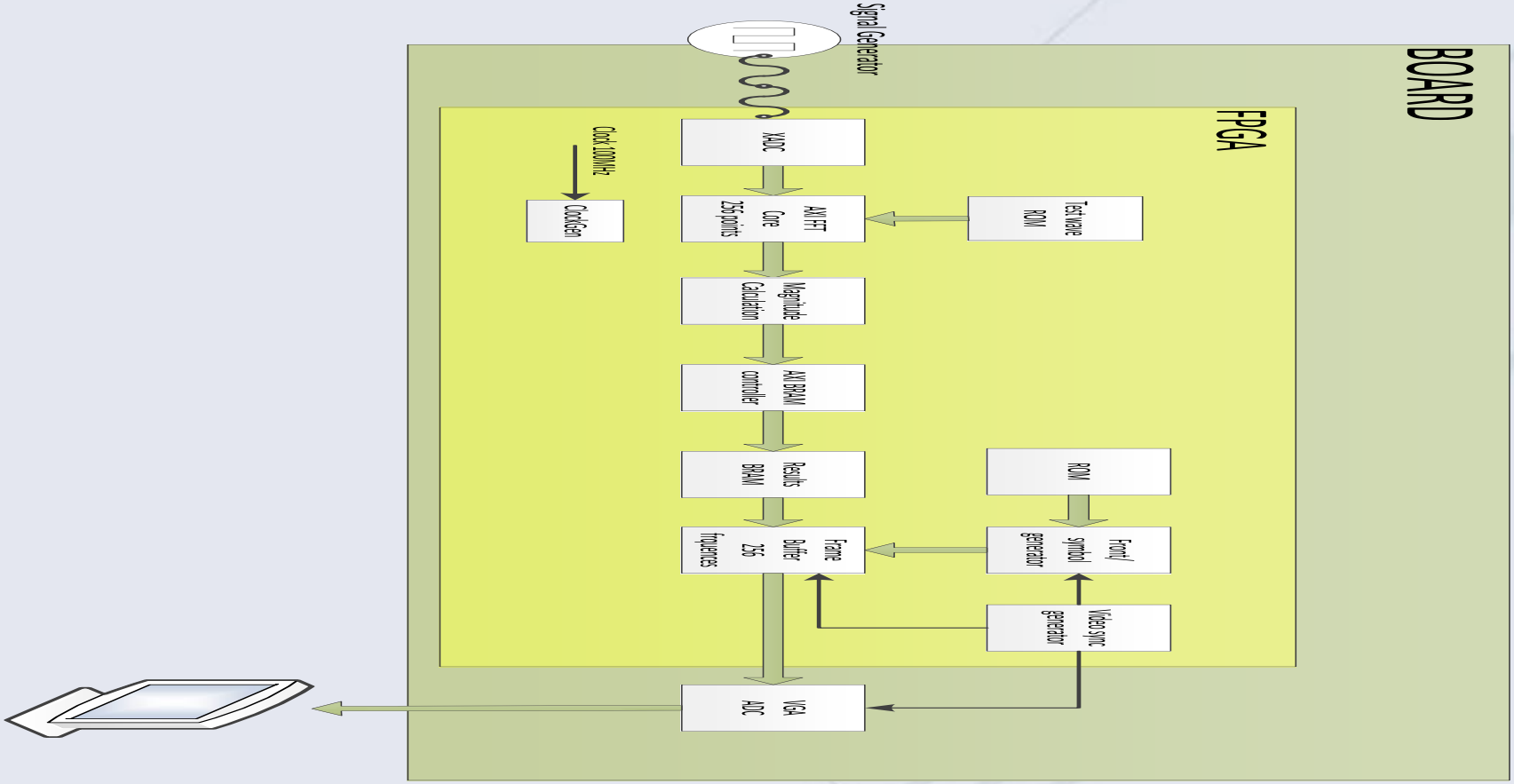
- Camera module is based on OV2640 5MP image sensor
- Video resolution: 640x480
- Frame rate: 24FPS
- Data output: RGB565 format
- Configuration: via I2C bus

Project example

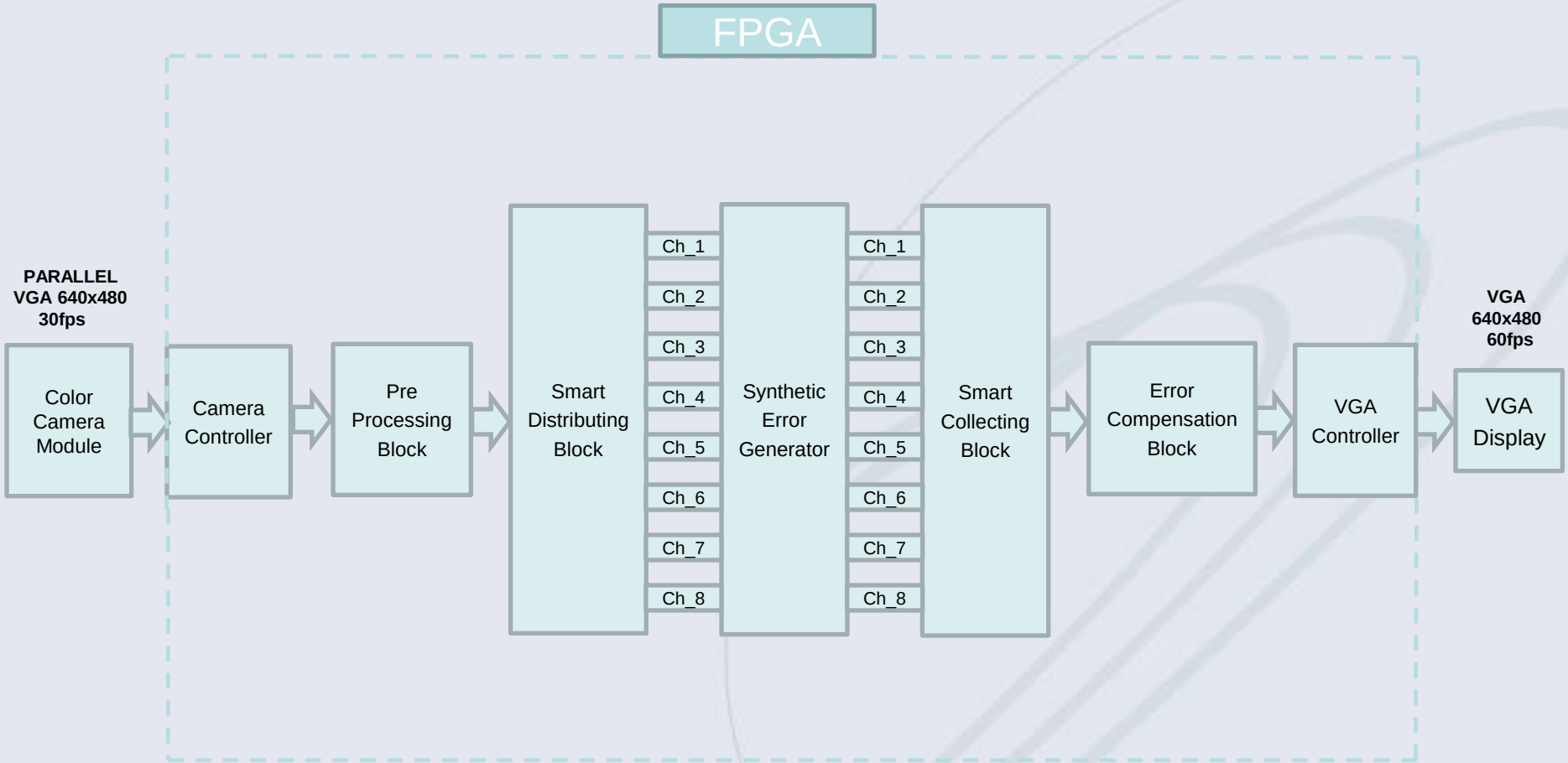
General block diagram



Project example



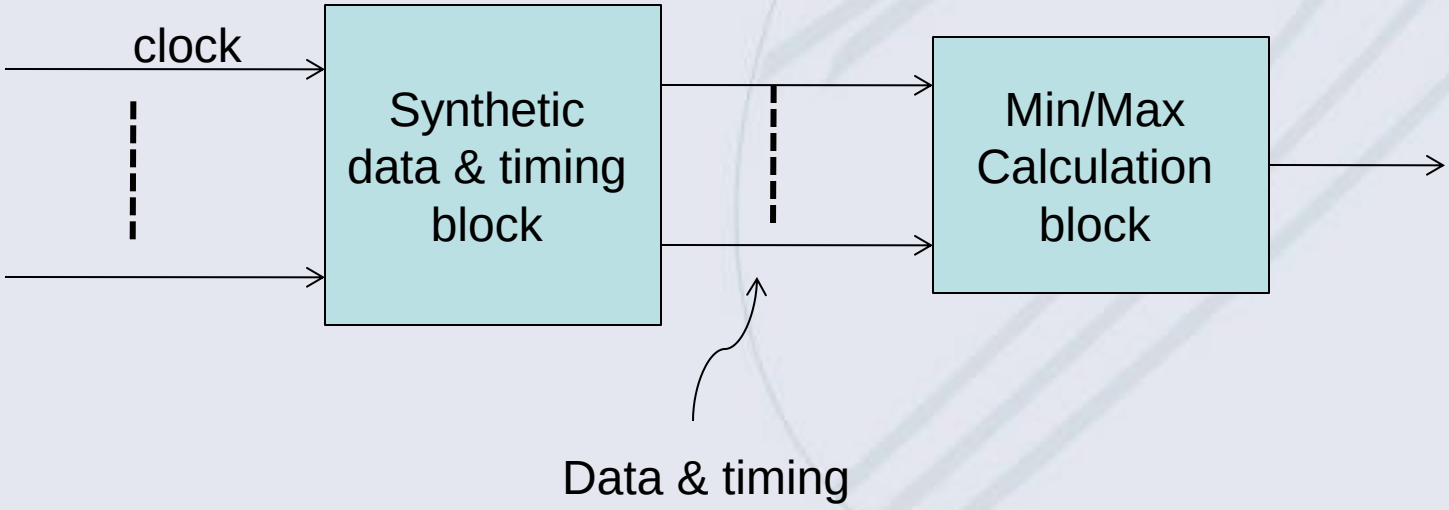
Concept Block Diagram



Projects Overview :examples-3

Min/Max Value Calculation

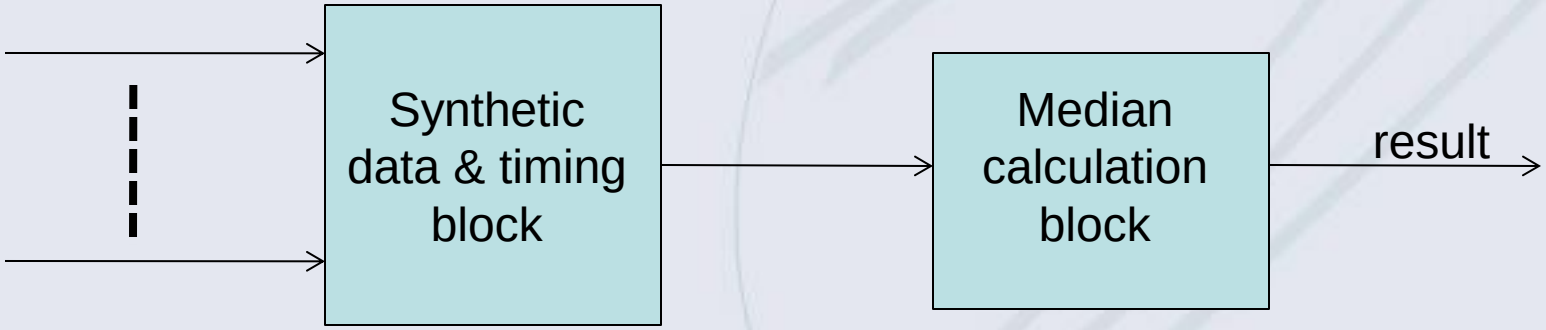
❖ In this Project the students will fined the maximum & the minimum value of the income data .



Projects Overview : Example-4

Median calculation implementation

- In this Project the students will calculate the median value of the income data stream .

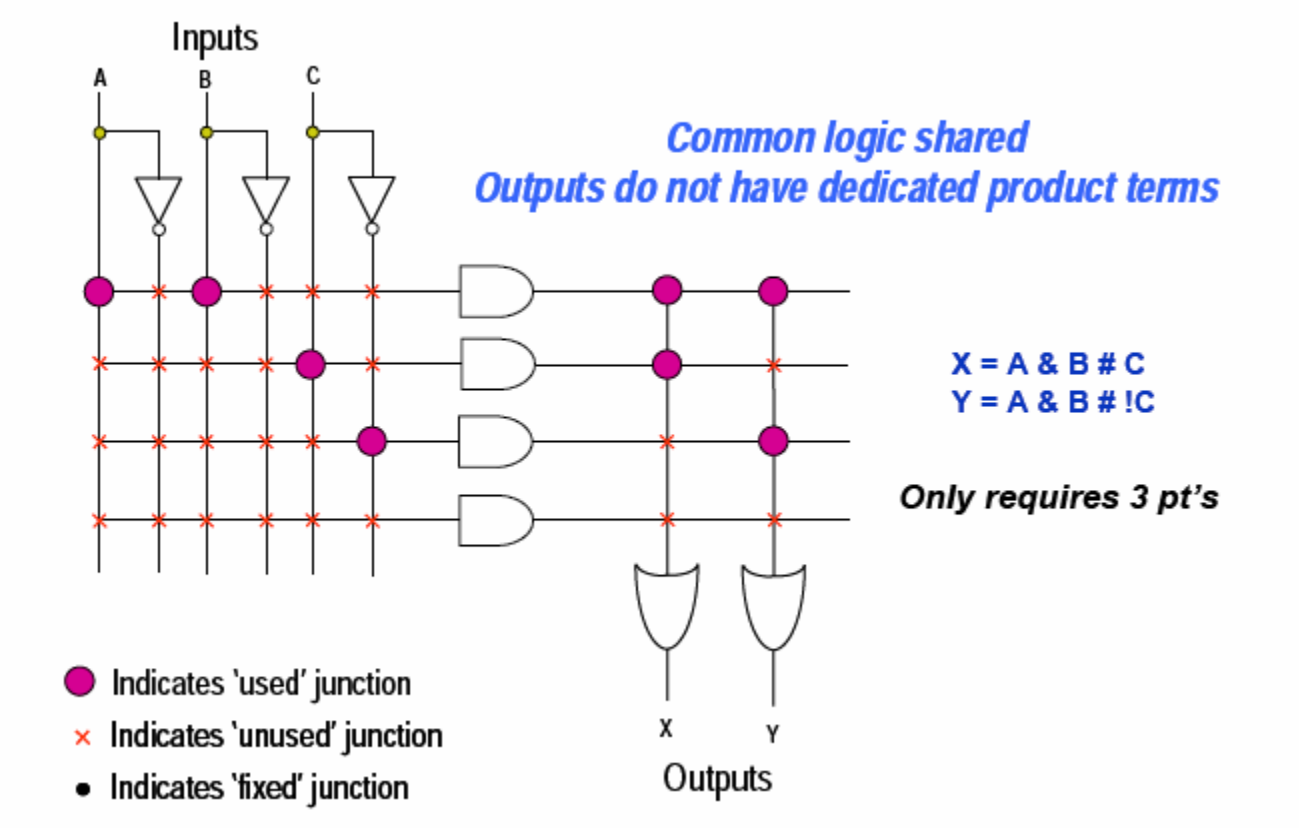


Basic’s Of Digital Design’s

The History of Programmable Logic

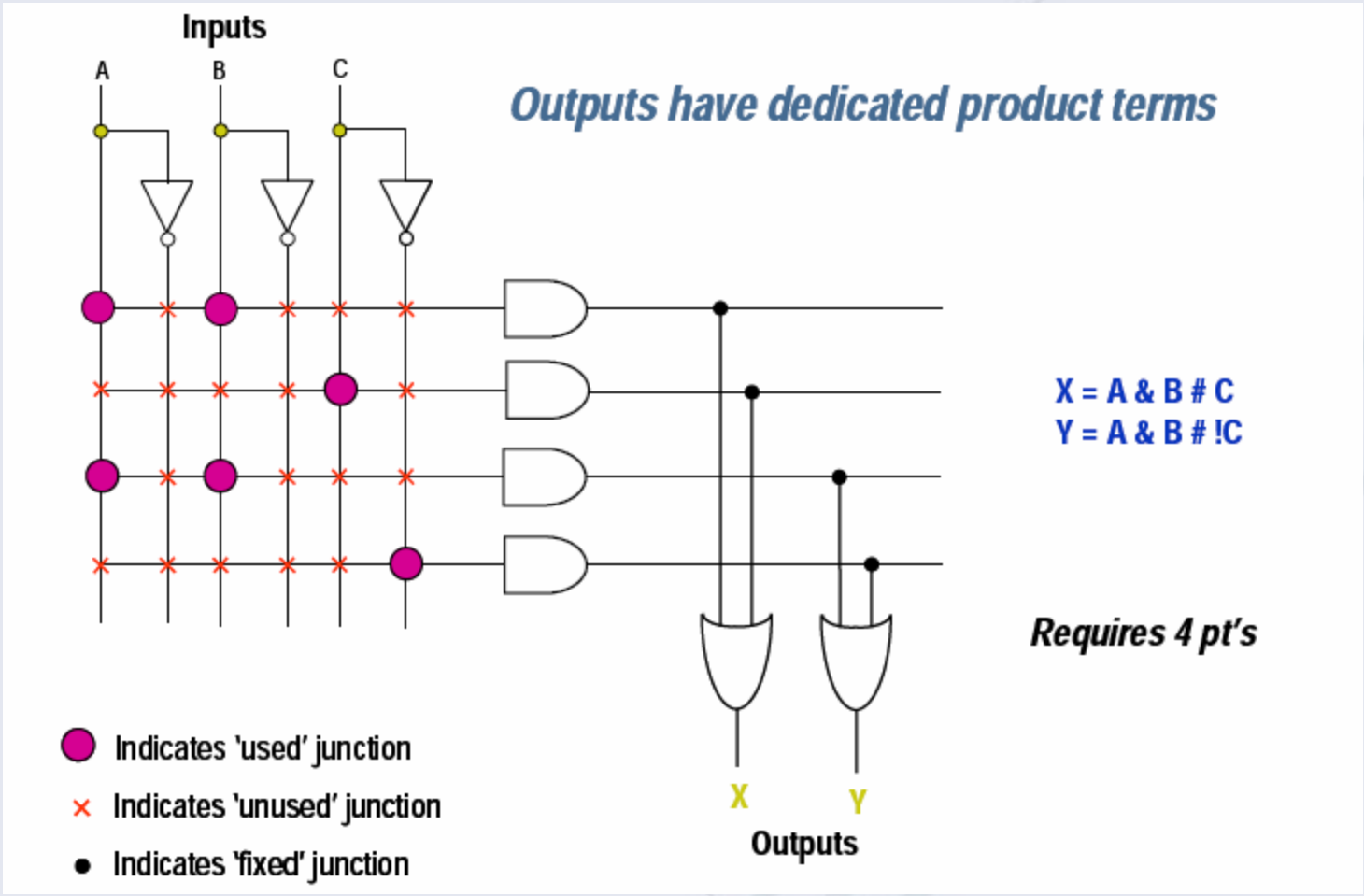
By the late 1970s ,there was the PLA Devices , that include combination of “AND” and “OR” gates , This architecture was very flexible, but at the time wafer geometries of 10 μm made the input-to-output delay (or propagation delay) high, which made the devices relatively slow .

The History of Programmable Logic



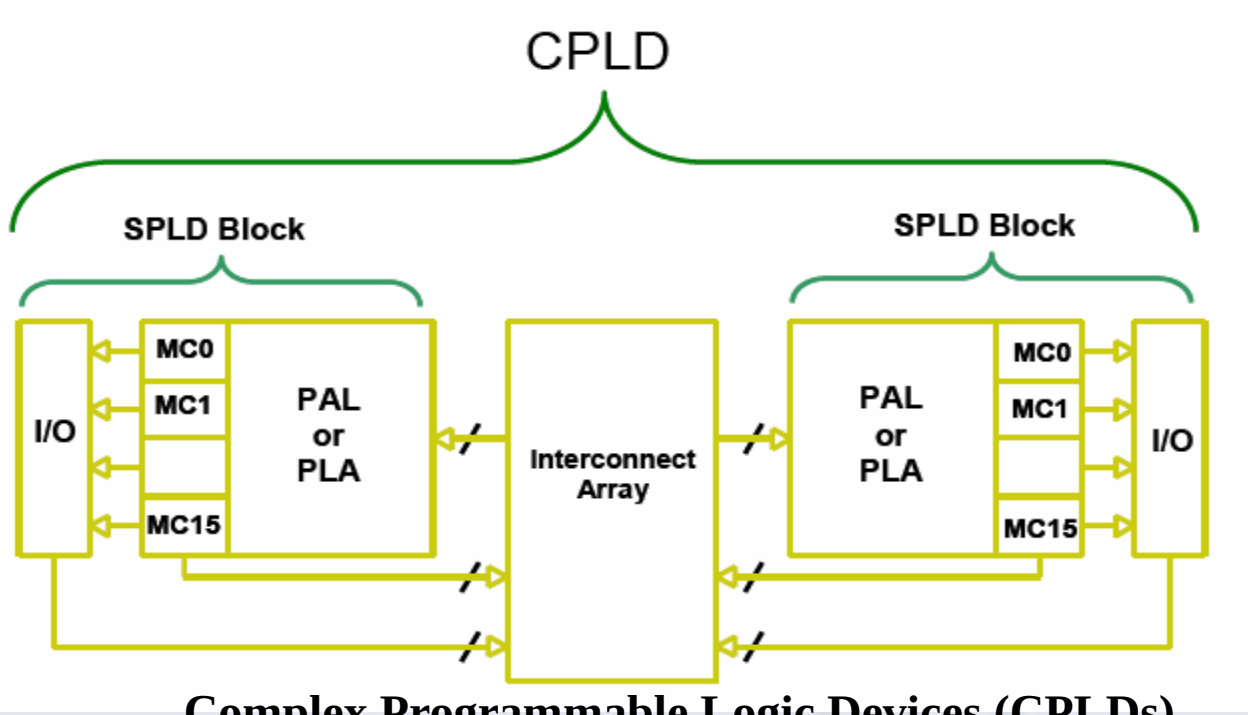
Simple PLA

The History of Programmable Logic



SPLD Architectures (PAL)

The History of Programmable Logic

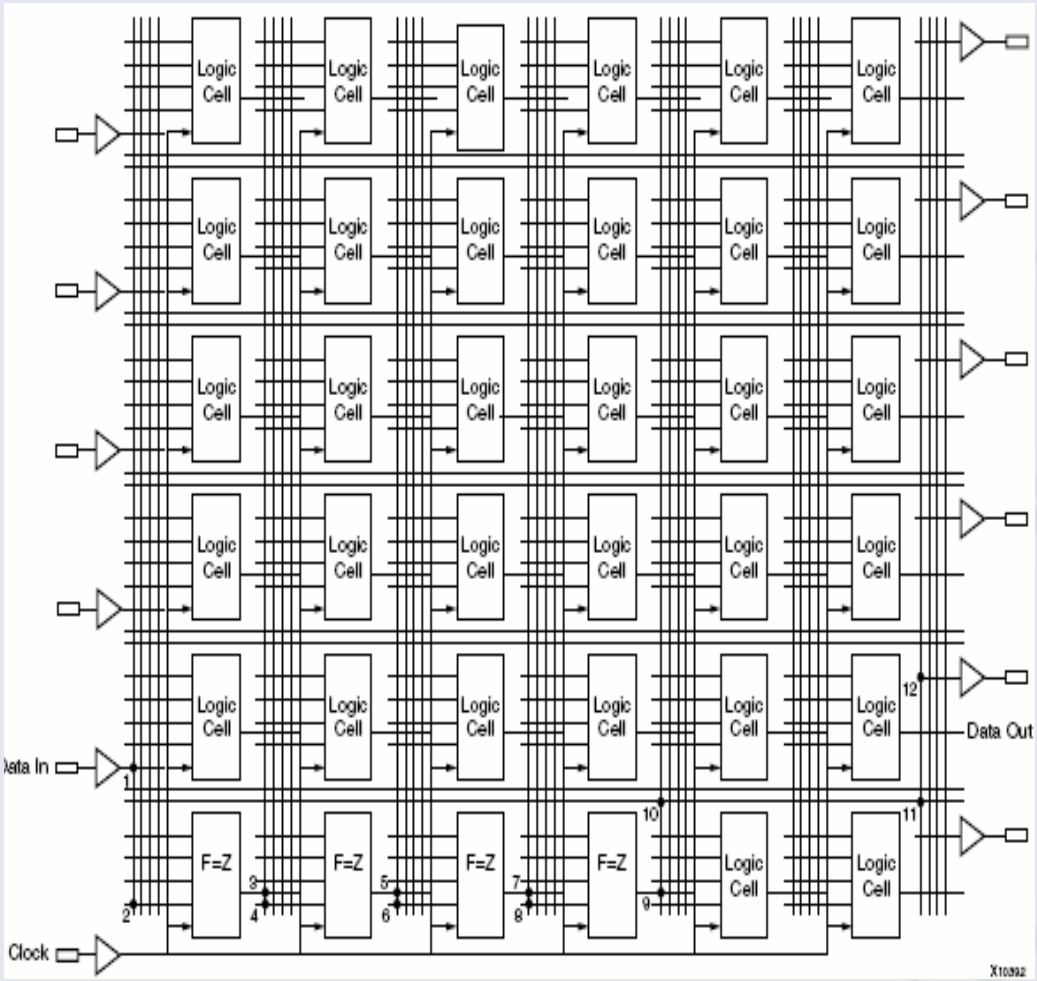


Complex Programmable Logic Devices (CPLDs)

CPLDs feature:

- Central global interconnect
- Simple, deterministic timing
- Easily routed
- PLD tools add only interconnect
- Wide, fast complex gating

The History of Programmable Logic



FPGA Architecture

With FPGAs now exceeding the 10 million gate limit, you can really dream big.

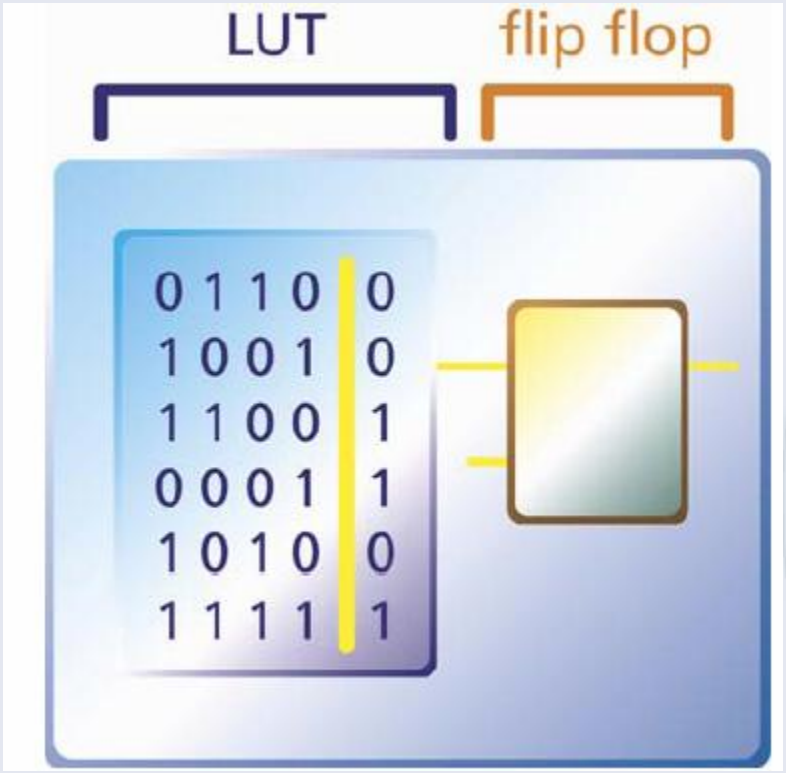
FPGAs feature:

- Channel based routing
- Post layout timing
- Tools more complex than CPLDs
- Fine grained
- Fast register pipelining

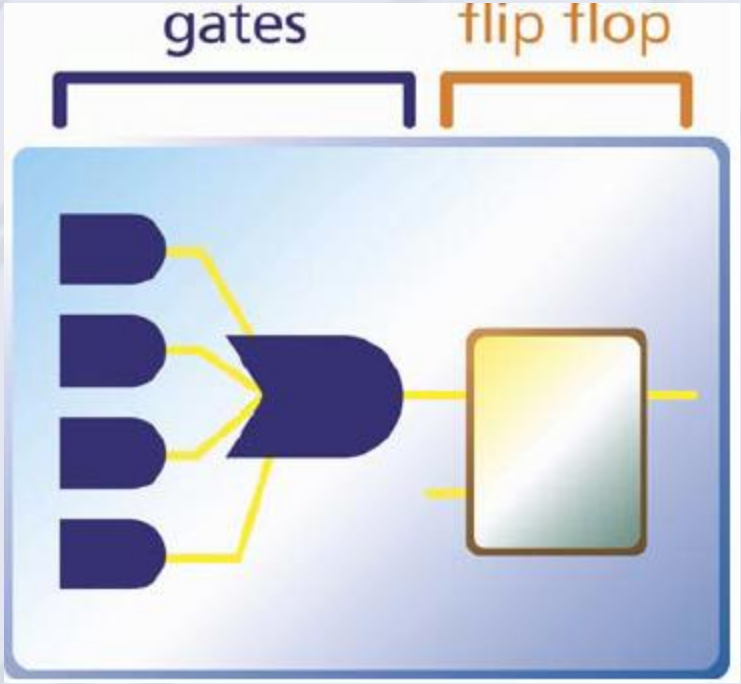
The History of Programmable Logic

There are two basic types of FPGAs:

- SRAM-based reprogrammable .
- OTP (One Time Programmable).



SRAM Logic Cell



OTP Logic Cell

Logic Gate

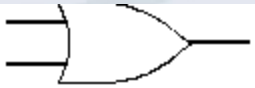
- A logic gate is an elementary building block of a digital circuit .
- Most logic gates have two inputs and one output.
- At any given moment, every terminal is in one of the two binary conditions *low* (0) or *high* (1), represented by different voltage levels.
- In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).
- There are 7 basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

Logic Gate

- The *AND gate* is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator.

AND gate

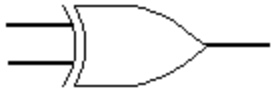
Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1



OR gate

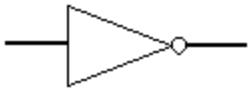
Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gate-Rest of logic gate



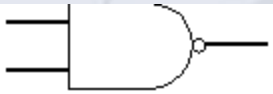
XOR gate

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0



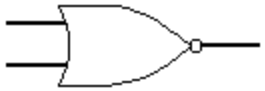
Inverter or NOT gate

Input	Output
1	0
0	1



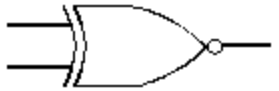
NAND gate

Input 1	Input 2	Output
0	0	1
0	1	1
1	0	1
1	1	0



NOR gate

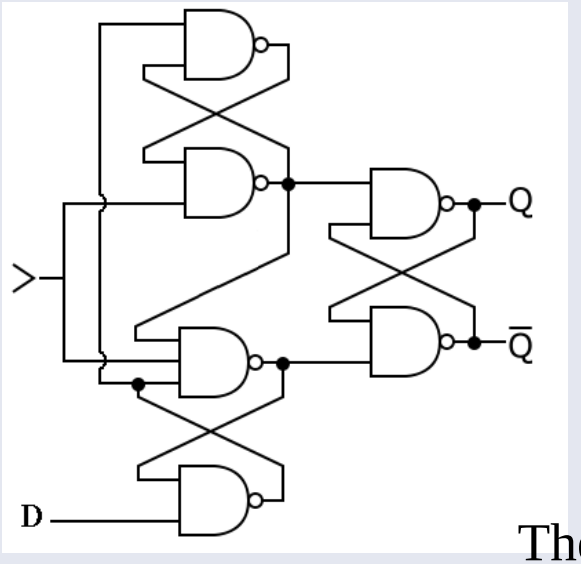
Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

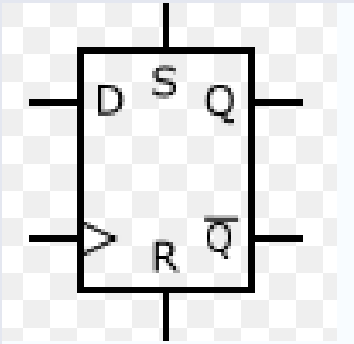


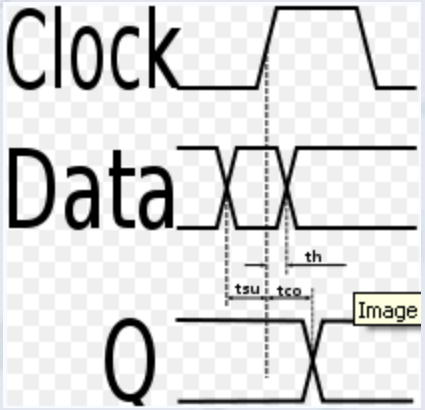
XNOR gate

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	1

D-FF schematic view



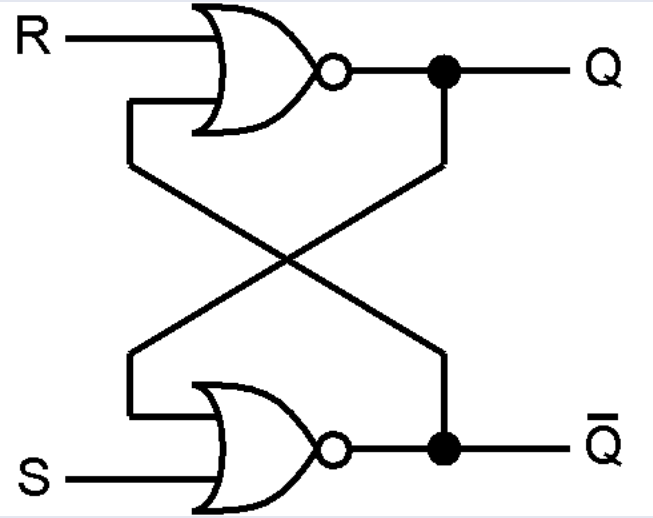


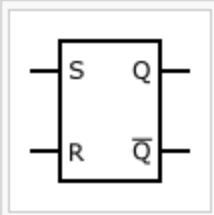


Clock	D	Q	Q_{prev}
Rising edge	0	0	X
Rising edge	1	1	X
Non-Rising	X	constant	

The Q output always takes on the state of the D input at the moment of a rising clock edge. (or falling edge if the clock input is active low) It is called the **D** flip-flop for this reason, since the output takes the value of the **D** input or *Data* input, and *Delays* it by one clock count. The D flip-flop can be interpreted as a primitive memory cell, or delay line.

Latch schematic view





The symbol for an SR latch.

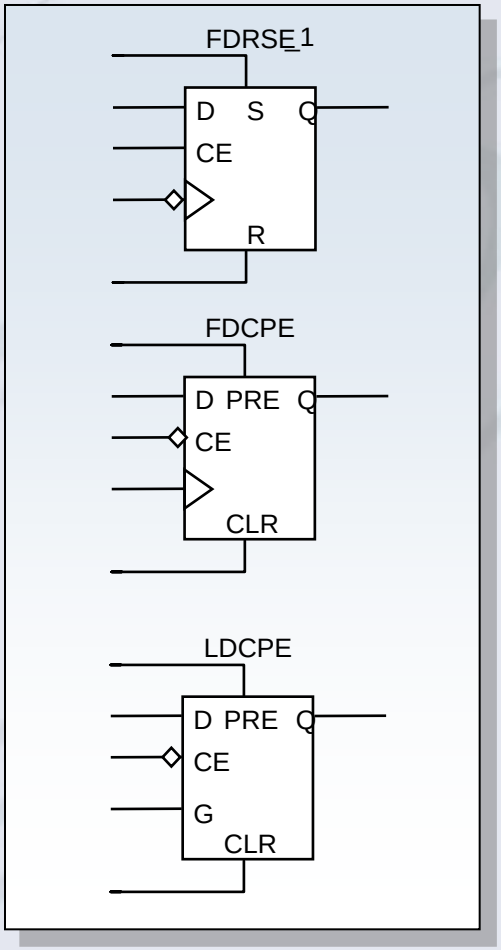
SR latch operation		
S	R	Action
0	0	Keep state
0	1	Q = 0
1	0	Q = 1
1	1	Restricted combination

When using static gates as building blocks, the most fundamental latch is the simple *SR latch*, where S and R stand for *set* and *reset*.

Latch Creation in FPGA

- flexible sequential elements that can be configured as either flip-flops or latches.

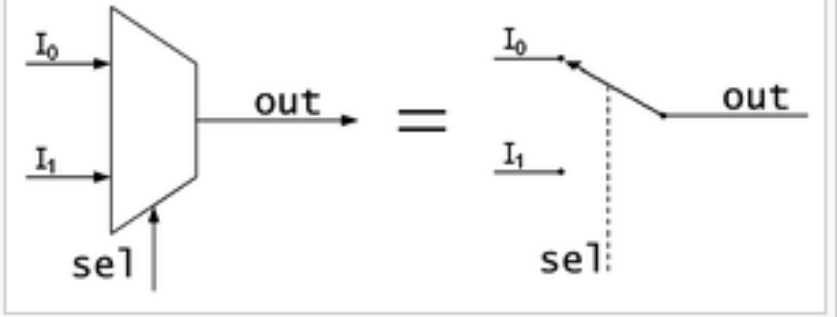
- Inputs to the FFs come from LUTs or from an independent input. There are separate set and reset controls on each FF, which can be programmed to be synchronous or asynchronous.



Latch in FPGA designs - discussion

- How we create Latch in Our Design ?
- It Is good or bad ?
- How We can avoid latch’s in our design ?

Mux schematic view



Schematic of a 2-to-1 Multiplexer. It can be equated to a controlled switch.

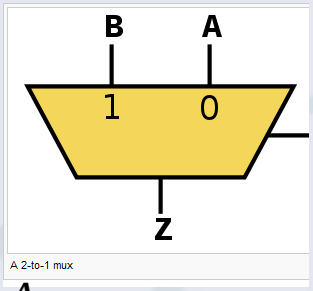
mux is a device that performs multiplexing, it selects one of many analog or digital input signals and outputs that into a single line.

An electronic multiplexer makes it possible for several signals to share one expensive device or other resource, for example one A/D converter or one communication line, instead of having one device per input signal.

Mux schematic view

A 2-to-1 multiplexer has a boolean equation where A and B are the two inputs, S is the selector input, and Z is the output:

$$Z = (A \cdot \overline{S}) + (B \cdot S)$$

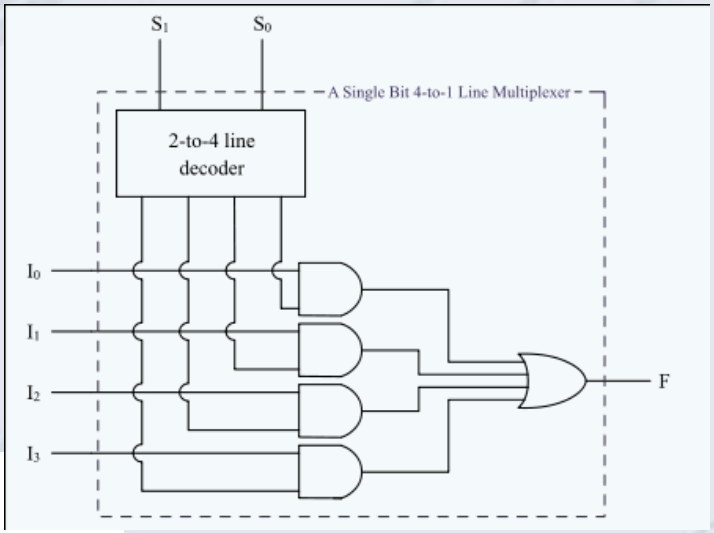


Which can be expressed as a truth table:

S	A	B	Z
0	1	1	1
	1	0	1
	0	1	0
	0	0	0
1	1	1	1
	1	0	0
	0	1	1
	0	0	0

This truth table should make it quite clear that when $S = 0$ then $Z = A$ but when $S = 1$ then $Z = B$. A straightforward realization of this 2-to-1 multiplexer would need 2 AND gates, an OR gate, and a NOT gate.

$\lceil \log_2(n) \rceil$

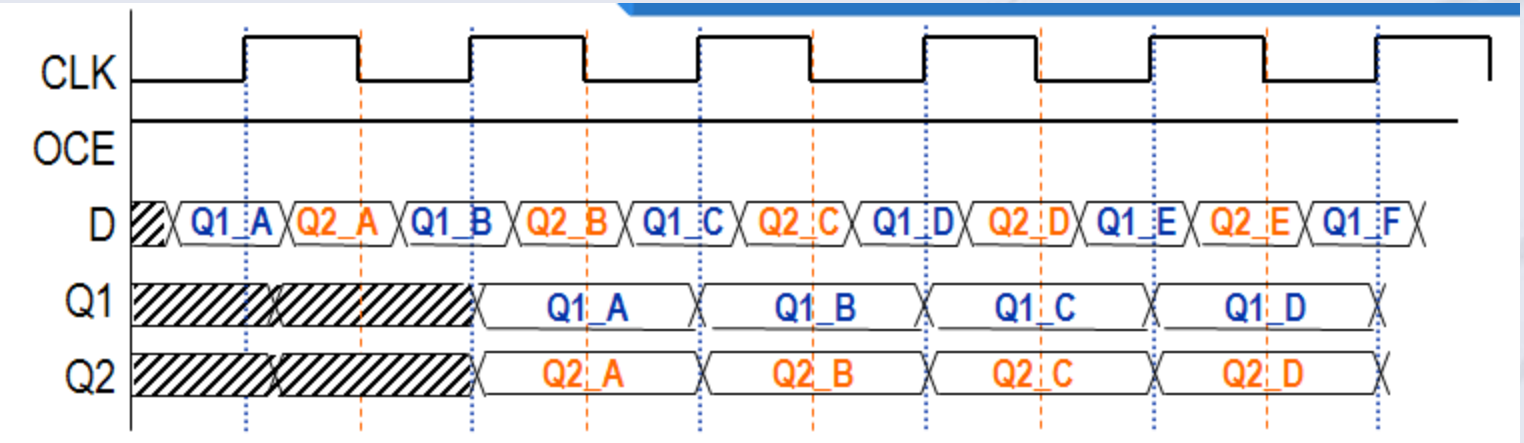


The boolean equation for a 4-to-1 multiplexer is:

$$F = (A \cdot \overline{S_0} \cdot \overline{S_1}) + (B \cdot S_0 \cdot \overline{S_1}) + (C \cdot \overline{S_0} \cdot S_1) + (D \cdot S_0 \cdot S_1)$$

Serialization & dserialization

In many I/O Protocols the data is Transmitted /Recived in Serial Mode and in Very High Frequency that is because the data is Serialize on 1 discreet line and each clock edge Point to different data .

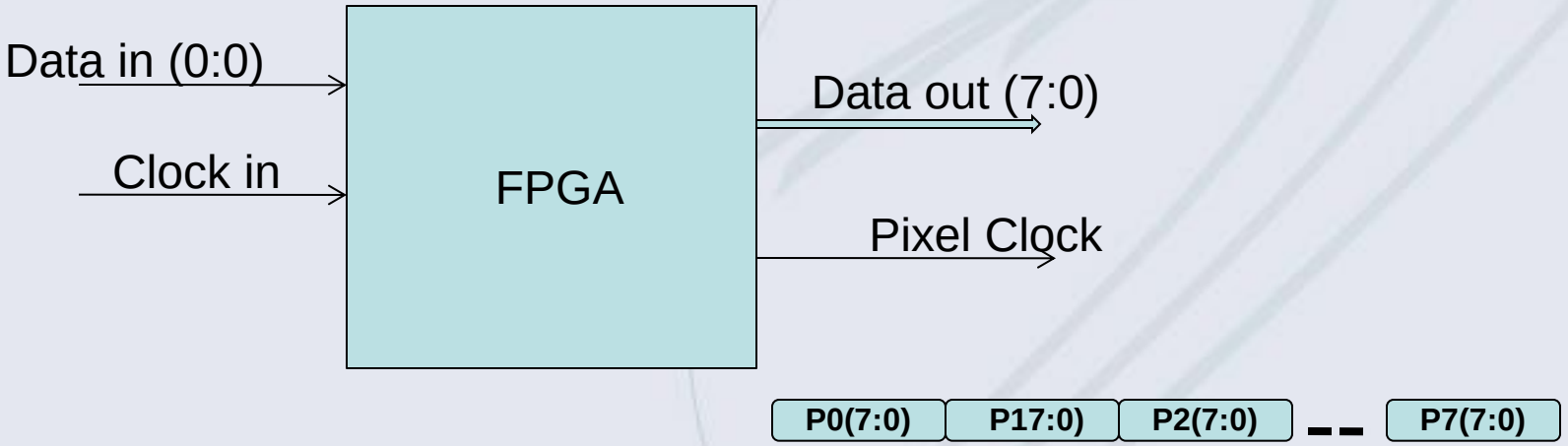


For those application The FPGA Vendors , build very Smart I/O functionality that can decode the income Data.

Why In The I/O ? And not inside the FPGA ?

Exercise

- There is 1 discreet line ,that serialize 8 bit ,each output Pixel is 8 bit length , the Pixel rate is 25 Mhz , what is the income clock rate?



What Is the Benefit to serialize the income data in 1 line ?

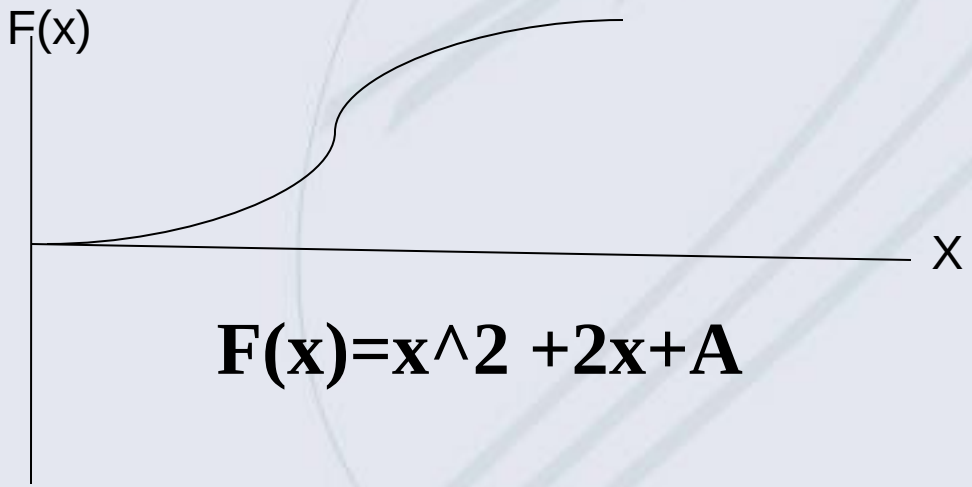
How we will implement this In FPGA?

Solution

- Benefit :To save I/O for different applications .
- Implementation :
  - ✓ To use the I/O capabilities to serialize/Deserialize the income data with the clock .
  - ✓ To build counter that will run in the High frequency clock and in each count different bit will insert to the pixel ,(Of course that we will need to synchronize the clock and data first).

Question

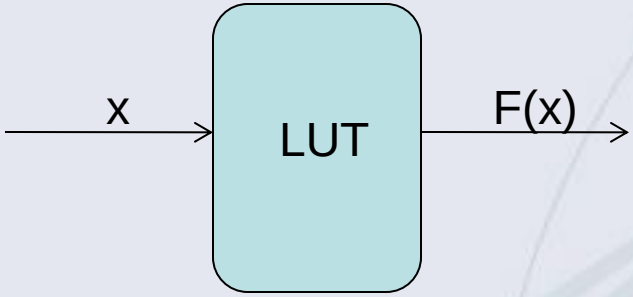
- If I want to Implement function such as F(x) , when X is parameter that enter to my digital logic circuit , in which basic element can I use to implement this function ?



F(x)=x^2 +2x+A

Solution

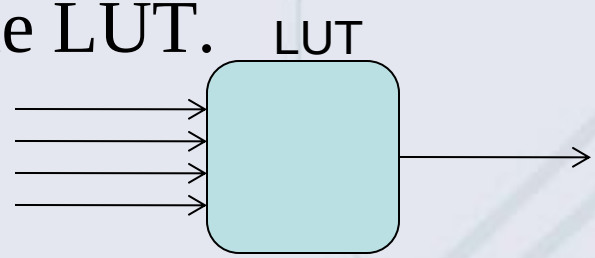
We will do it with LUT , that is lookup table , every income X parameter will point to the F(x) value .



A Lookup table is a data structure ,often use to replace a runtime computation with a simpler array indexing operation.
The savings in terms of Processing time can be significant , since reading value from memory is often faster than calculating it .

LUT

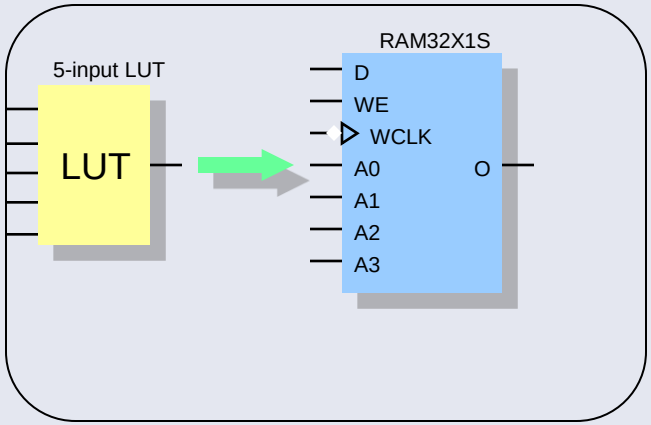
- LUT is the basic elements in FPGA , where the combinatorial logic is implemented , for instant , all the Boolean equation of combinatorial logic gate are implemented .
- Example : if I have the following Boolean Equation : Z=(A * B)+ (C*D) , when A,B,C,D are the input and Z is the Output , the implementation will be in one LUT.

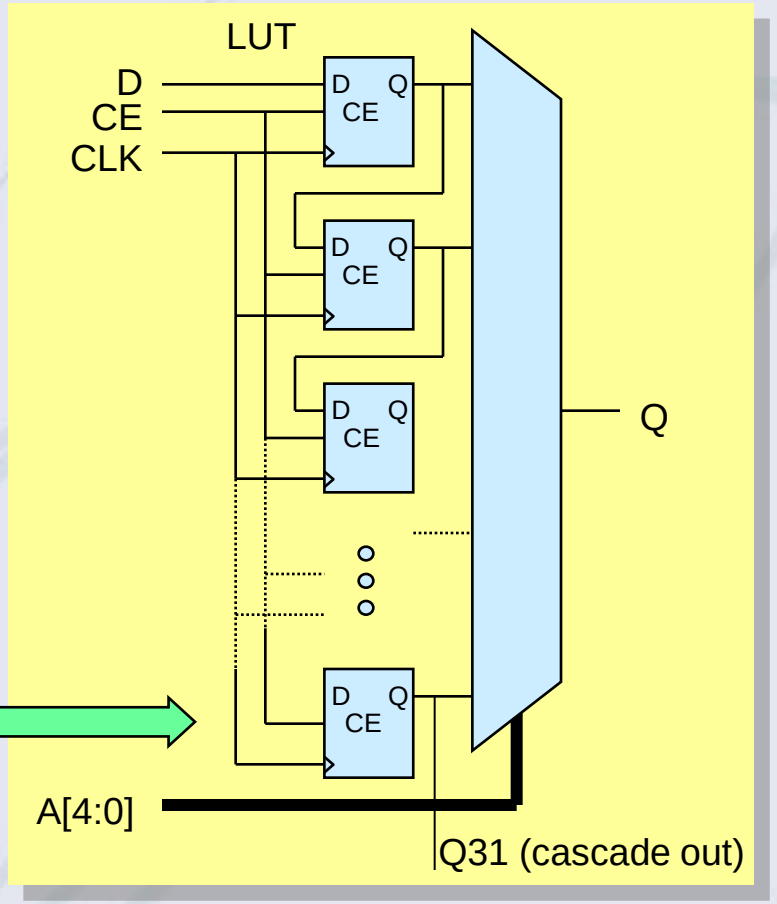


LUTs as a Shift Register (SRL) or Memory

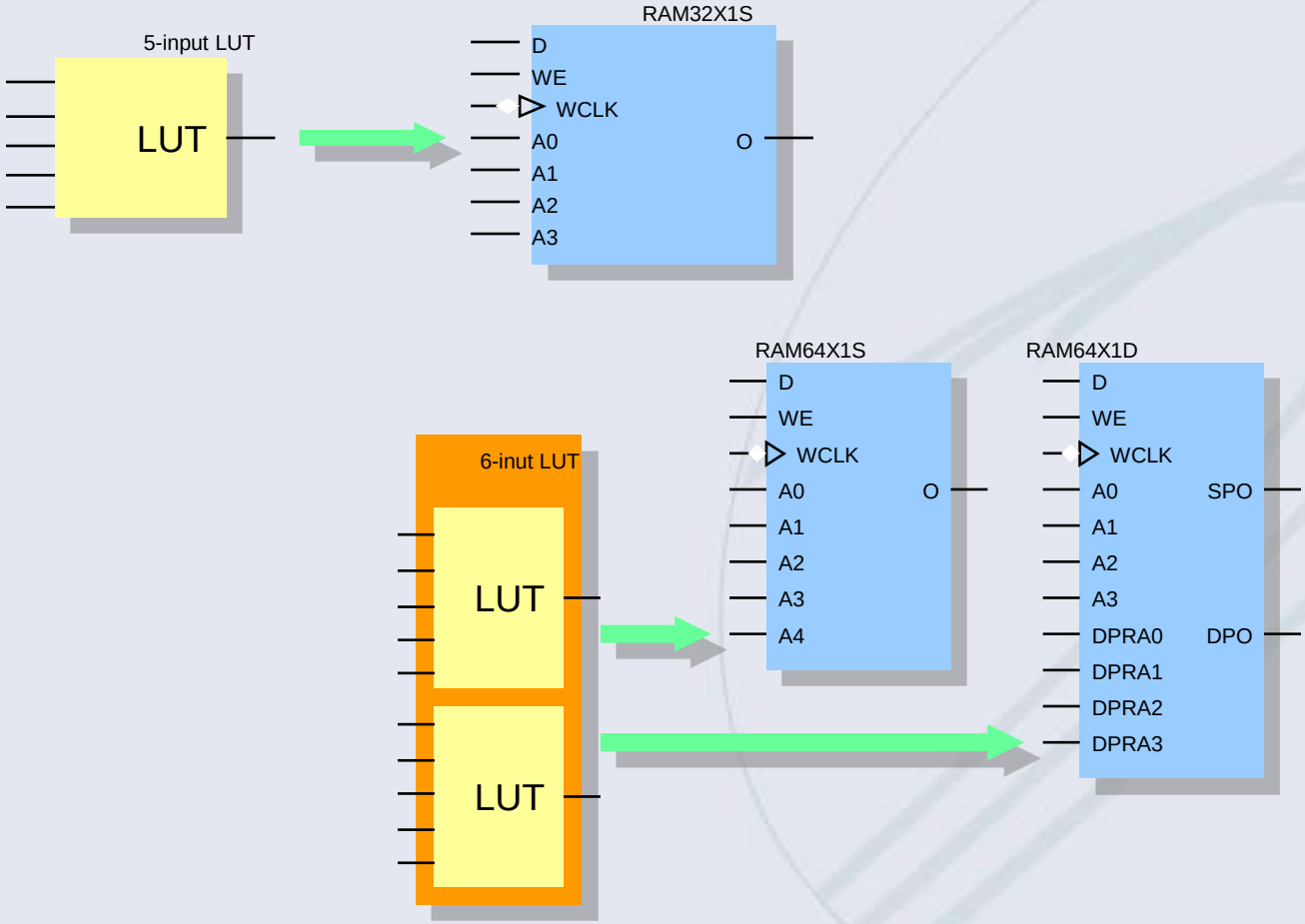
❖ We can use the Lut's as a dynamically Addressable Shift Register.

❖ We Can also Use the LUT as a basic memory cell that can be implemented as a distributed Ram .





Distributed Select RAM Resources Use a LUT



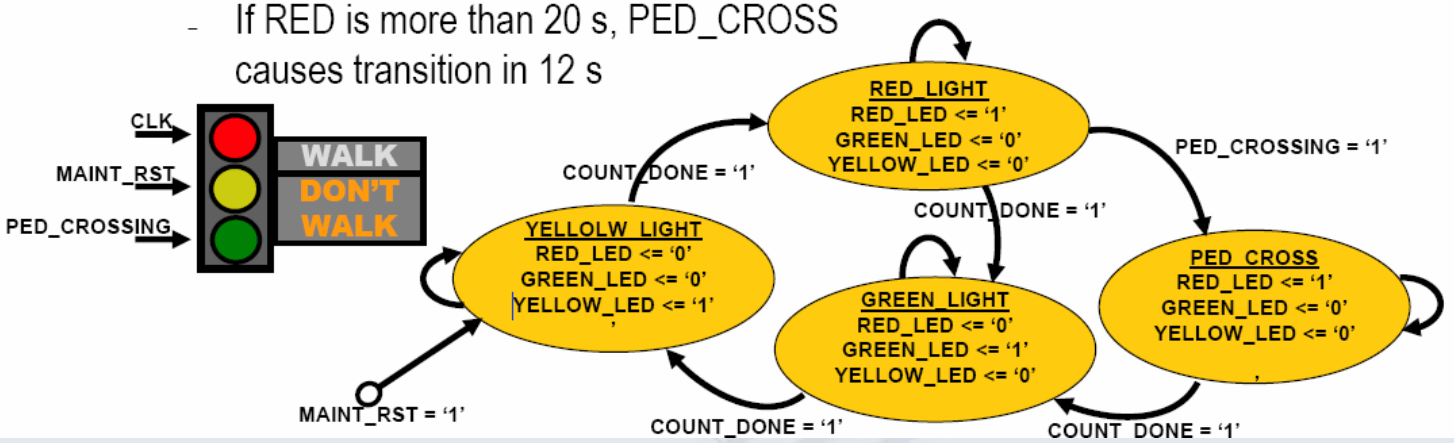
State machine

- FSM – Finite State Machine Is One of the most usable language capability in HDL Designs .
- FSM – is model of behavior composed of a finite number of state , transition between those states , and actions .
- An FSM is an abstract model of machine with a primitive internal memory.

State machine

FSM Coding Example

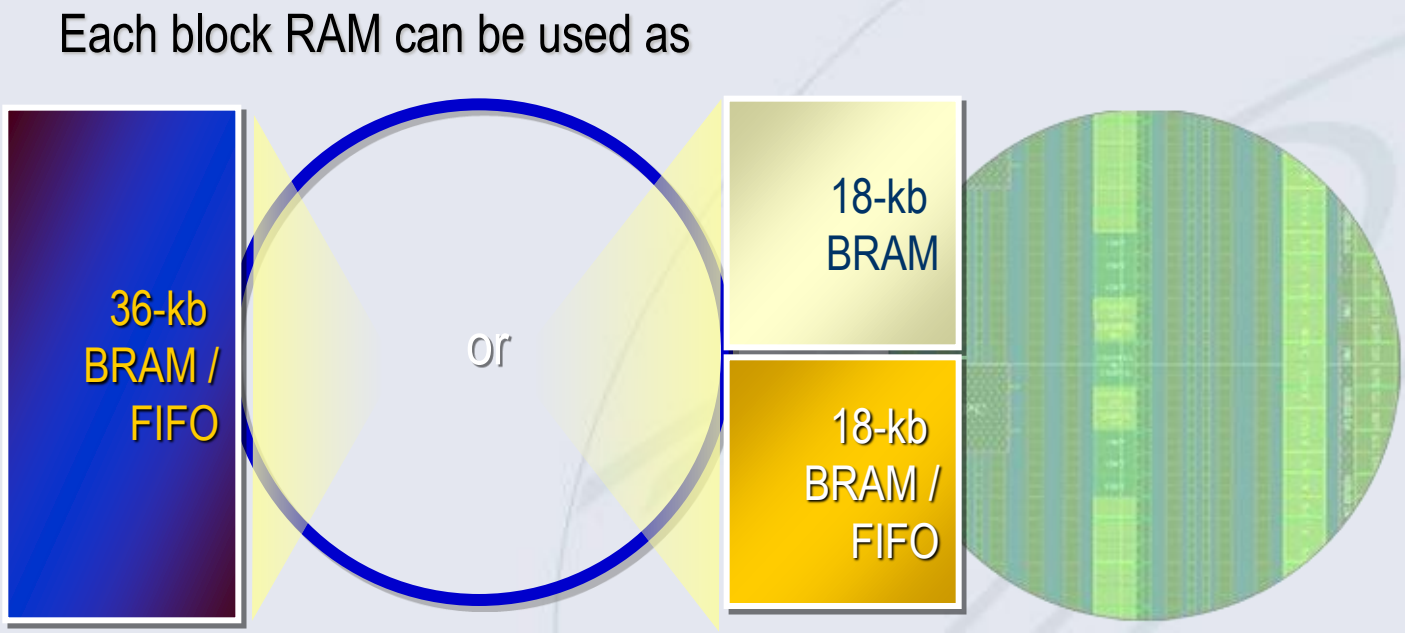
- This FSM models a standard traffic signal with crosswalk
- Functional specifics
  - RED and GREEN lights cycle normally for 80 seconds each
  - YELLOW light timer is 10 seconds, also MAINT_RST state
  - If RED is less than 20 s, PED_CROSS causes transition in 20 s
  - If RED is more than 20 s, PED_CROSS causes transition in 12 s



Memory

- Memory Or Block Ram is one of the mostly basic blocks that in use in many Digital Design applications .
- Most devices today Have build in block Rams (or FIFO)inside the chip .
- There is also the option to write in HDL language memory and the EDA tools have to translate it to block Ram (More Generic RTL) .

Block RAM and FIFO Block



The Size of the block ram is depending on the device family , it also can be DPR or SPR or FIFO, depending on the configuration .

Combinatorial Logic

- What is Combinatorial logic ?

- Combinatorial logic is a concept in which two or more input states define one or more output states, where the resulting state or states are related by defined rules that are independent of previous states.

- And In Our terms , ‘Comb’ Logic is Logic that runs without clock synchronous .

Combinatorial Logic :example

X <= A and (B or C) ;

X<= D and B;

If We write It In VHDL Which value X will get ? Why ?

How The Implementation will look like ? How many Resources ?

Sequential logic

- What is Sequential logic ?
- In Digital circuit theory, **sequential logic** is a type of logic circuit whose output depends not only on the present input but also on the history of the input.

- This is in contrast to Combinational Logic, whose output is a function of, and only of, the present input. In other words, sequential logic has storage (*memory*) while combinational logic does not.

- Most practical computer circuits are a mixture of combinational and sequential logic.

Sequential Logic :example

```
Process(Reset ,clk)
Begin
If rising_edge(clk)then
If (Reset = ‘1’)then
X <= ‘0’;
Else
X <= A and (B or C) ;
X<= D and B;
End if;
End if;
End process;
```

Which value X will get ? Why ?

Basics Terms In Digital Designs

- RTL - Register transfer level .
- Synchronous – logic with clock .
- Asynchronous – logic without clock.
- Capacity – Mostly meaning for device capacity , or in other words How full is my device (logic, memory, clocks resources , ect.)
- Slack – the different between the requested frequency and the real(or estimated) frequency.
- LUT- lookup table .
- CLB- configurable logic Block.

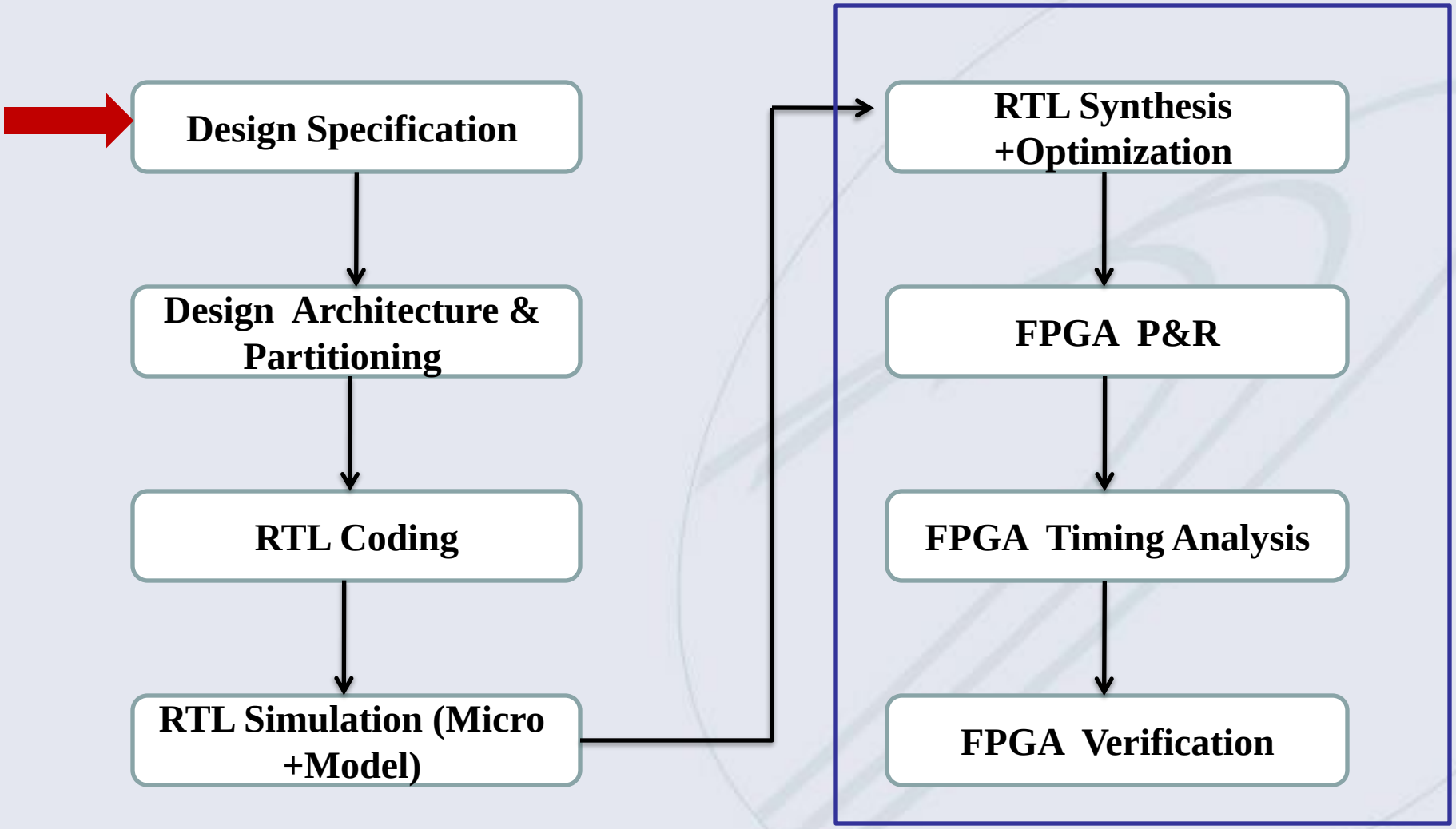
DESIGN FLOW OVERVIEW

DESIGN FLOW

- It is very important to know and understand the Design Flow because it gives to the designers the ability to plan the design cycle and also the effort each stage will tack from Him .

- In the DFO , the design Performance will be impact not just from the RTL code efficient , it will be influenced also from the EDA Tools , and from the Timing/Area Constrain .

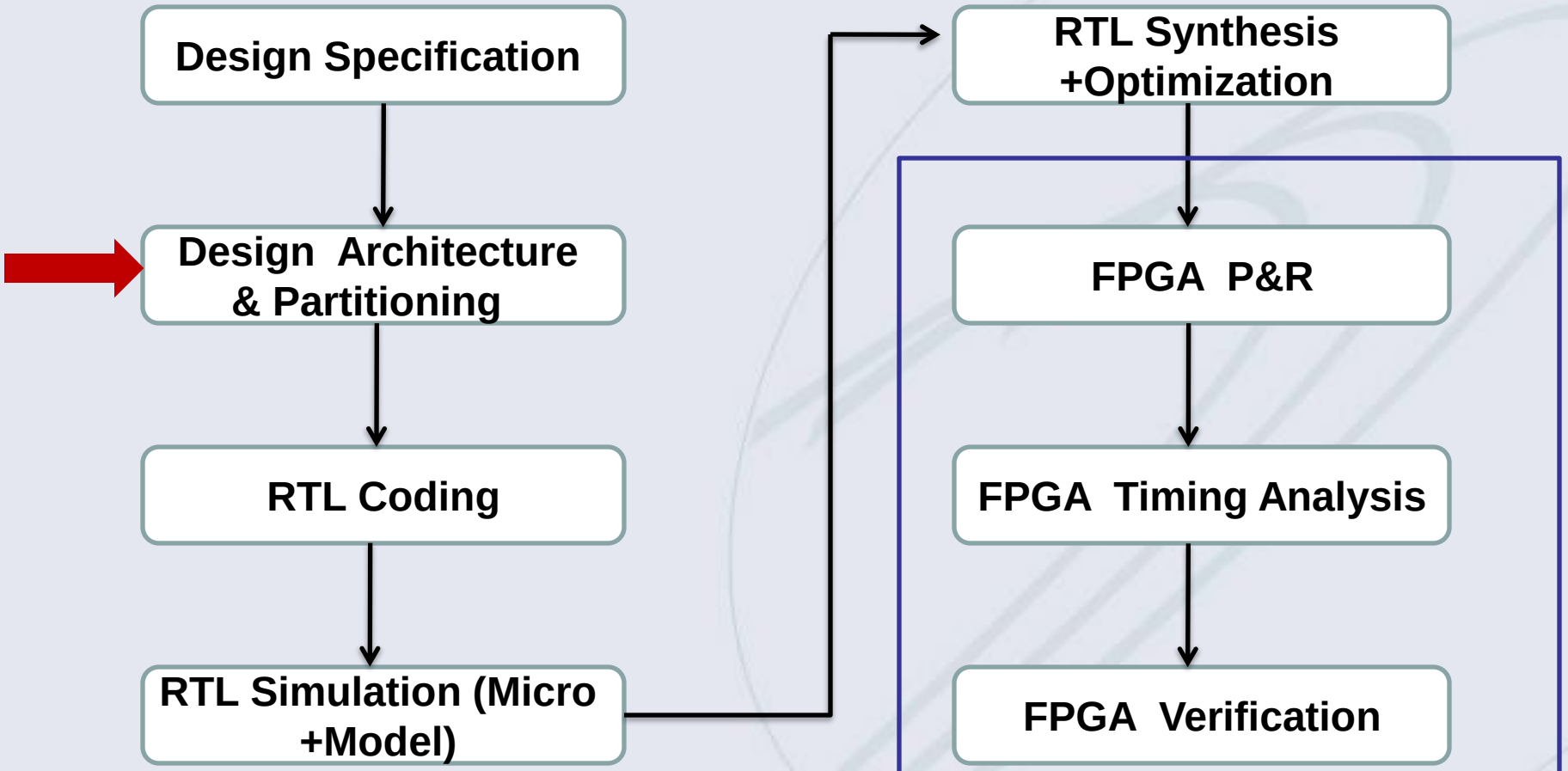
FPGA Design Flow Overview



Design Specification

- This is the first stage of the design cycle , when the engineer get the Spec and learn from it all the details and the connections between the different elements that he will need to build .
- This Is very important stage , it will impact all the stages that will come after that , Like device size, design time ,how Meany designers and so on …
- Mostly done by experienced designer or system engineers .

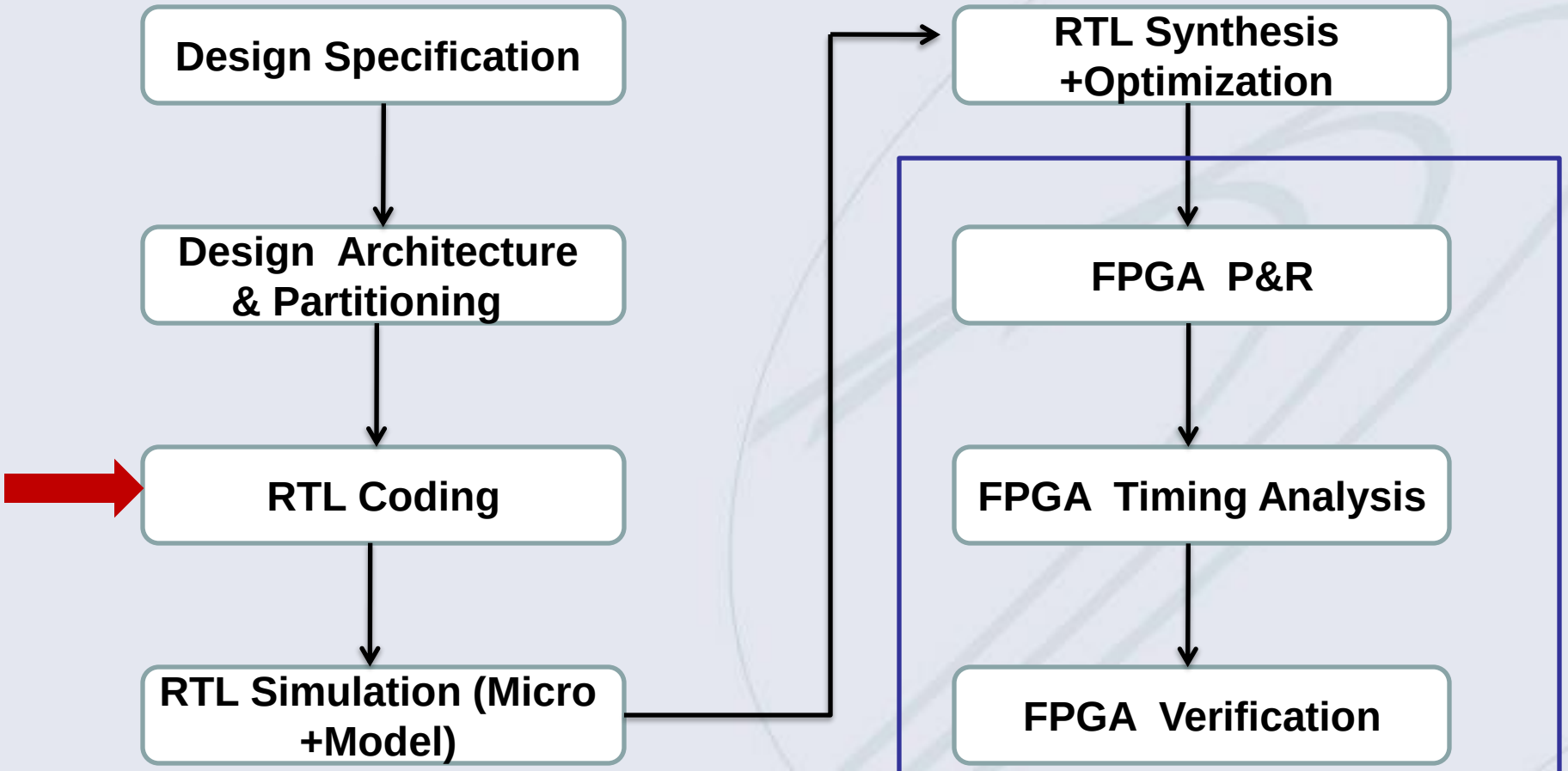
FPGA Design Flow Overview



Design Architecture & Partitioning

- This is the most important stage in the digital design cycle , it means how the designer partition the design block and what are the connections between the different blocks .
- The output of this stage is the Architecture of the design , in this architecture the designer Show the design block unit and there content .
- It is very important to thing on the debug and verification of the design in this stage , it will save lot of time during the verification stage .
- Good Micro Architecture, To show as many details as possible , ant to analyze the Complex section of the design.

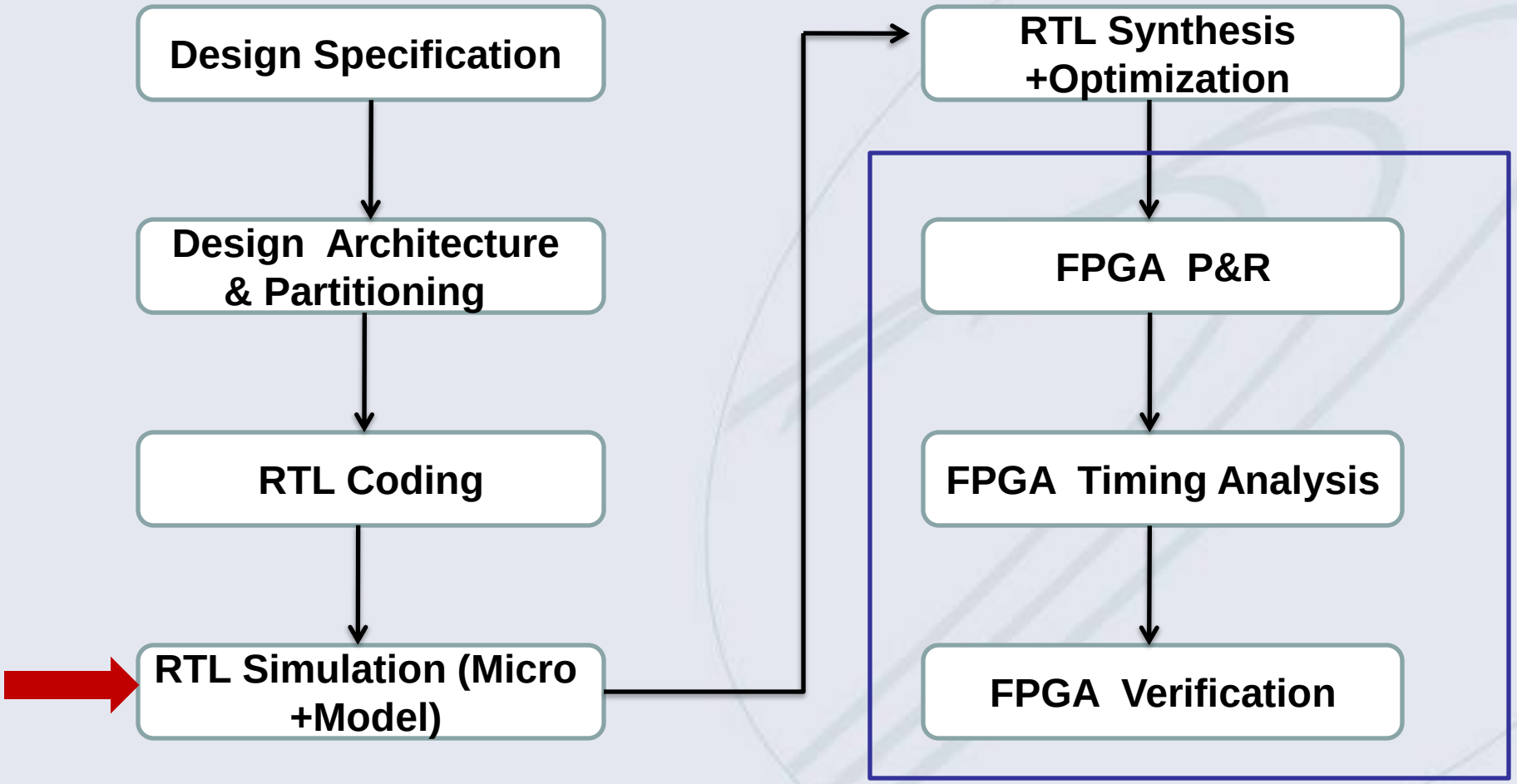
FPGA Design Flow Overview



RTL Coding

- This is the stage where the designer start to implement the design accordingly to the design plan and architecture that he made in the previous stages .
- It is very important for debug issues and understanding the flow , to write efficient number lines of code (about 100 lines per block) for each block unit .
- Efficient RTL coding save lot of effort and Time in the next stages.
- It is Important to describe each unit in the code and give effective signal naming , for easy analyze in the future .
- Code Reading is very effective in the end of this stage .

FPGA Design Flow Overview



RTL Simulation (Micro +Model)

- In The design Cycle the RTL simulation Is the first and fast stage for design verification , in this stage the designer can understand if the design is doing the work .

- In this stage there is no device depending Issue, and also EDA tools optimization are not took into account .

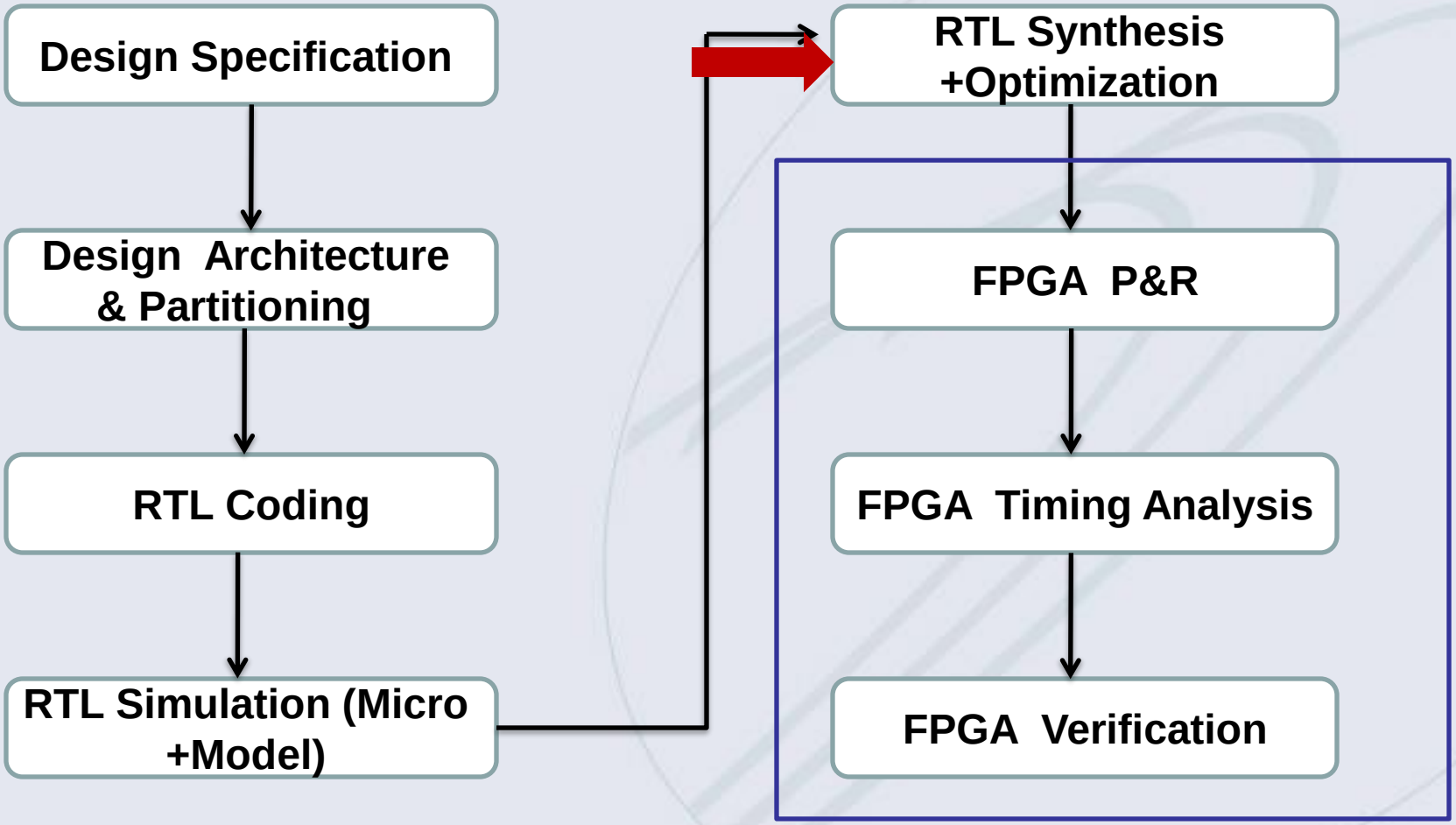
- It is Important to Know ,the different between RTL code that can run trough the synthesis stage and code that can’t .

RTL Simulation (Micro +Model)

- It is very useful to built good Test bench that simulate the real input signals to the design .

- The Expert designers run the simulation for each block unit of the design (in the Micro architecture) , so if there will be a problem in the Design verification stage they will have the ability to analyze it very fast .

FPGA Design Flow Overview

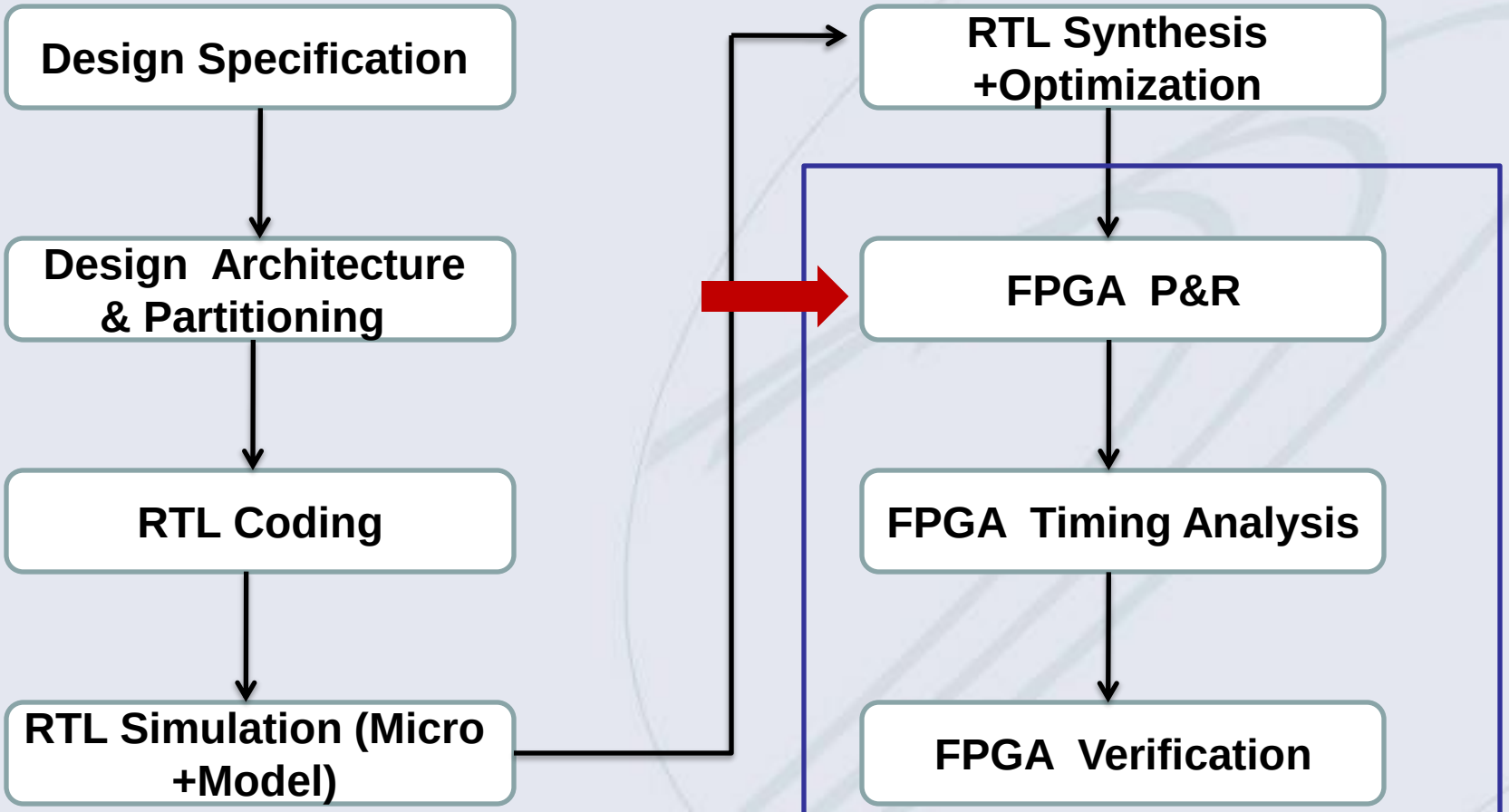


Design Flow Overview

RTL Synthesis +Optimization

- Synthesis is the stage where the RTL code transfer to logic gate net list , this is the first stage that machine interpreted high level code to low level code , by taking into account the device parameters and delay of the basic elements .
- The synthesizer do some optimization to build efficient net list for the next stage .
- Today There are also Physical synthesis that can forward placement constrain to the P&R .
- It is very important to insert Timing Constrain .

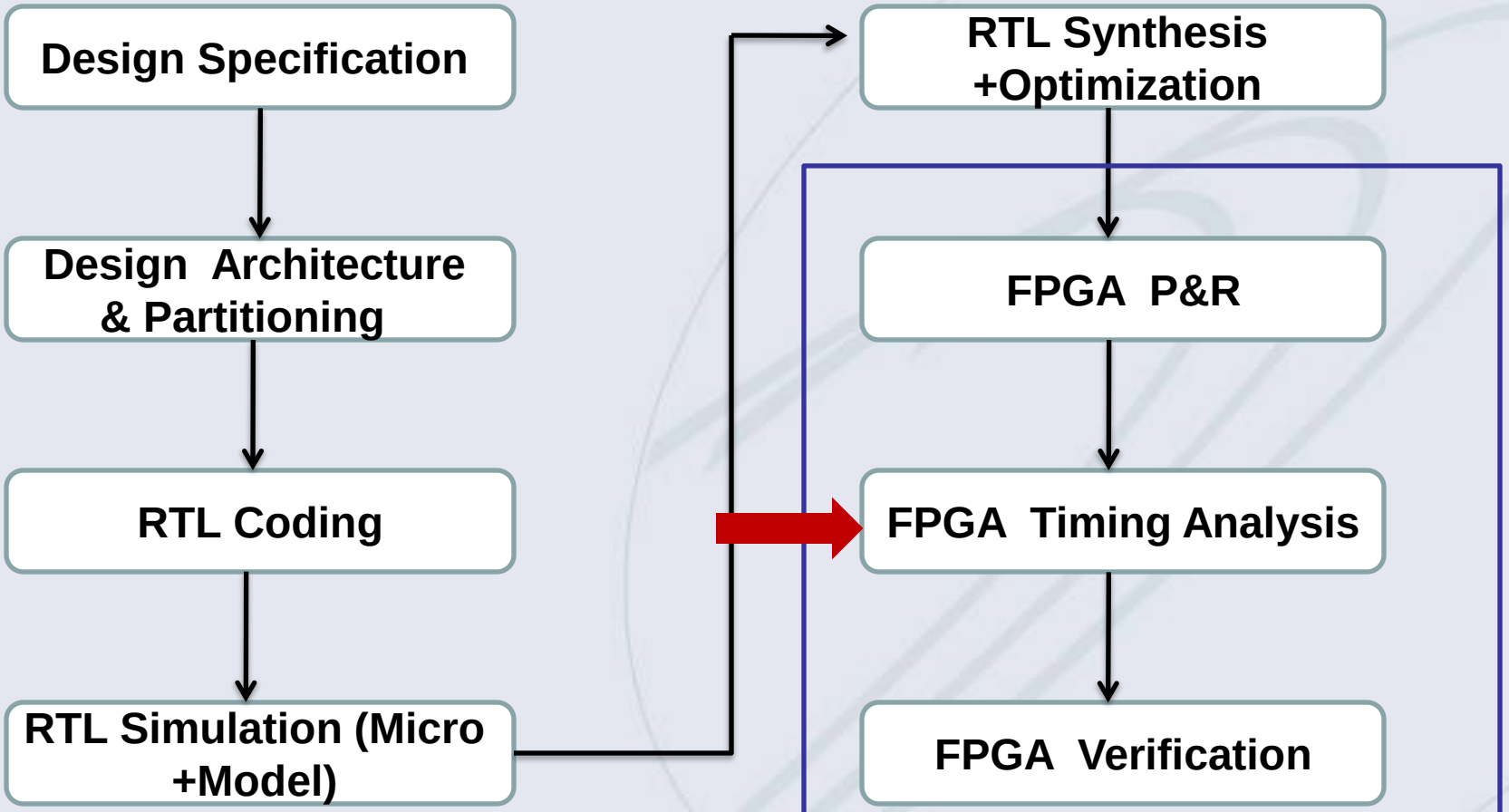
FPGA Design Flow Overview



FPGA P&R

- This Is the stage where the vendor tools do the placement of the Net list and the routing between the different elements in the design .
- The output of this stage will be the programmable bit stream to the chosen device .
- The result of the timing in this stage will be the real timing result as in the chip (usually).
- There is option to guide the P&R tool accordingly to placement constrain , and not to count on the internal algorithms of the tool.

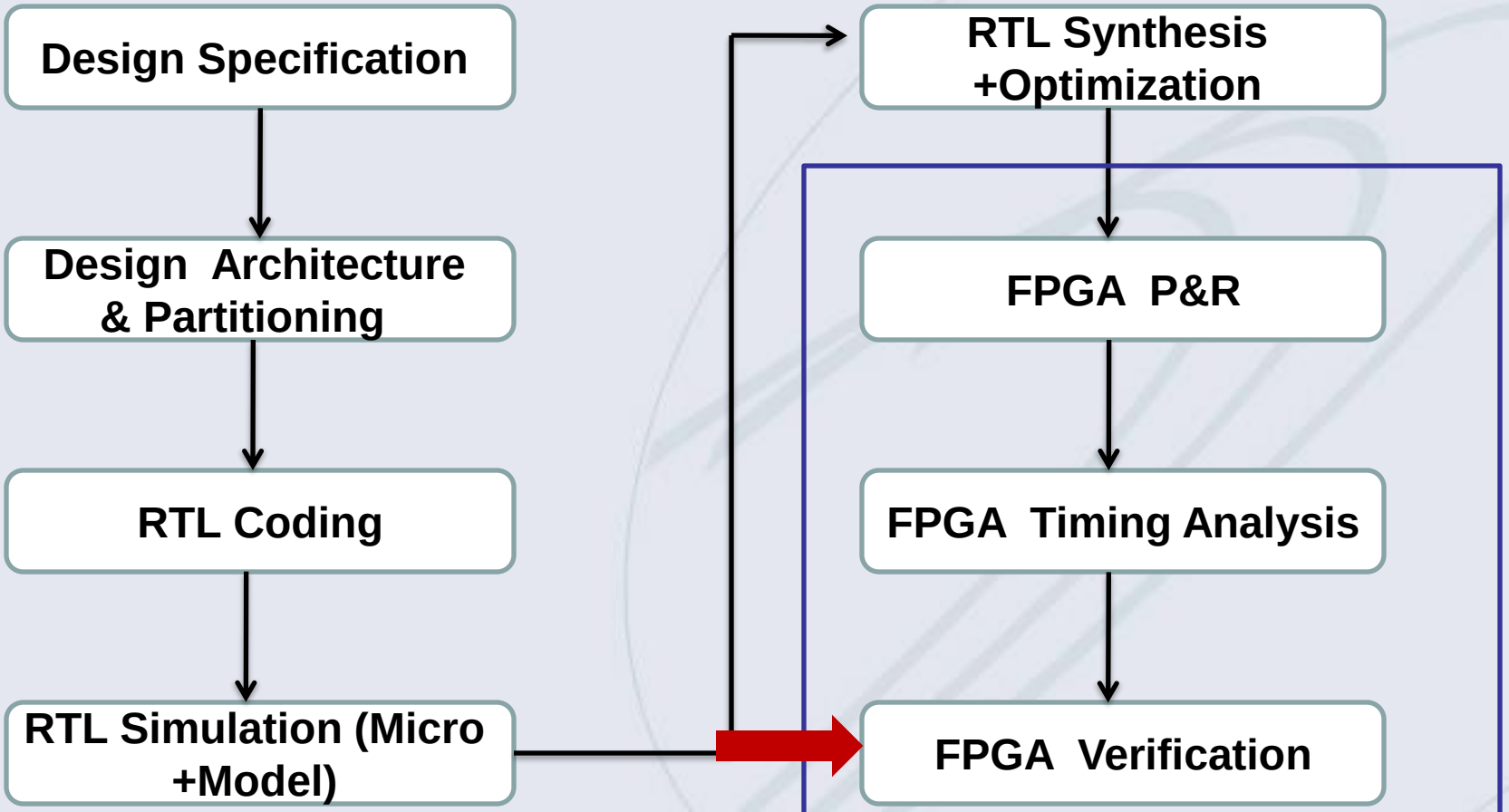
FPGA Design Flow Overview



FPGA Timing Analysis

- In This stage The designer can get information about the timing performance of the design after P&R .
- Accordingly to the timing report the designer decide how to react , he can change the RTL Code , or insert timing constrain or placement constrain ect.
- It is very important to now how to read and understand the timing report .

FPGA Design Flow Overview



FPGA Verification

- In this stage we check the design that run on the chip .
- There several method and methodology's how to check the design .
- There are several tools to do so : Chipscop for xilinx devices , signal-tap for altera devices , Identify For all devices , logic analyzer , scope .

SUMMERY

- There are several topic that we need to learn to become an FPGA expert .
  - ✓ The HDL coding .
  - ✓ The EDA tools .
  - ✓ The Targeting devices features .
  - ✓ The Verification Process and methodology.
  - ✓ The understanding the timing issue in our design .
- In this course we will constraint in the practical side of the FPGA designs .

Thank You !!!