

High Level Synthesis

Ranim Tom

September 2, 2025

Contents

1	Introduction	7
1.1	HLS Overview	7
1.2	Development Environment and Hardware	7
1.3	FGPA Concepts	8
1.3.1	Design Level	8
1.3.2	FGPA and CPU	9
1.3.3	FPGA Structure	11
1.3.4	Look Up Table	13
1.3.5	LUT in FPGA	13
1.3.6	Flip Flop	14
1.4	FPGA Example	15
1.5	Basys 3 Board Anatomy	17
1.5.1	Power Source	18
1.5.2	FPGA Configuration	19

Todo list

introduction	7
Design Level	9
FGPA vs CPU Parallelism and Sequential synthesis tool and bitstream	10
LUT in FGPA	11
FPGA configuration	13

Chapter 1

Introduction

1.1 HLS Overview

This report will talk about high level synthesis (HLS) design using FPGA.

Unlike traditional hardware description language (VHDL, Verilog), HLS uses high level languages such as C and C++ to design products using FPGA. The advantage is the speed of learning compared to traditional languages.

Introduction:

- To adapt more the introduction
- Insert references and books

1.1.1 FPGA Benefits and introducing HLS

FPGA offers some extra benefits, compared to some other computing platforms such as multicore CPU and GPU. These are:

- Reconfigurability: the architecture of an FPGA can be adapted depending on the demand of the application
- Parallelism
- Less power compared to other platforms for the same task

However, FPGA was suffering from serious drawbacks such as:

- The complexity of hardware languages: learning them, and debugging their applications
- Lack of standard libraries to accelerate development

This is why HLS has come and introduced the idea of using *high level languages to design hardware modules*.

1.2 Development Environment and Hardware

The development environment used is from Vivado company, and the board will be the Basys3 from Digilent.

1.3 FGPA Concepts

As stated in [section 1.1](#), HLS is about bringing programming concept to the hardware using FPGA.

In this section, we present a high level overview about the FPGA concept that we should now in order to do some HSL programming. The overview map is presented in [Figure 1.1](#)

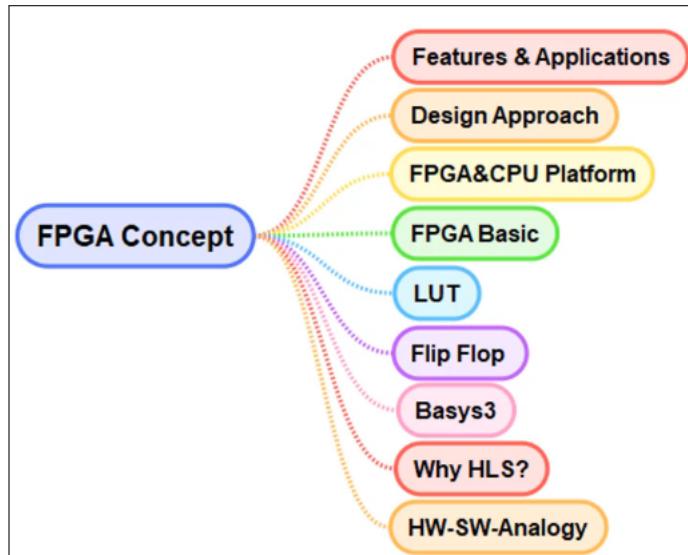


Figure 1.1: FPGA Concepts Overview

1.3.1 Design Level

FPGA desing is categorized in 3 different levels as shown in [Figure 1.2](#).

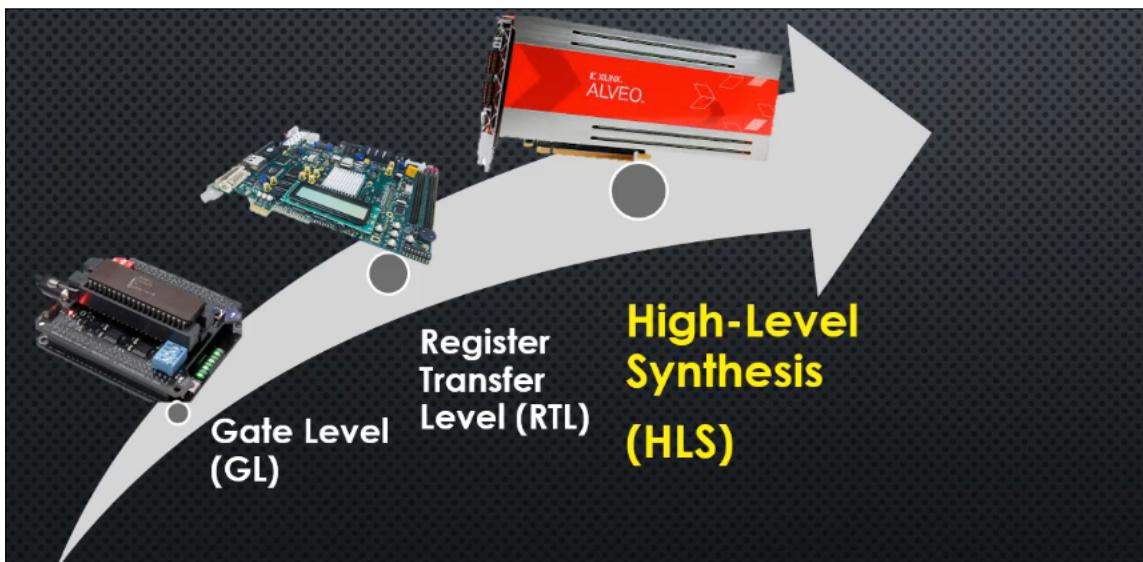


Figure 1.2: FPGA Design Levels

- Gate Level: using simple gates (AND, OR, · · ·). This allow hardware designers to implement small circuits on small FPGPA.

- RTL: with more advanced in resources and capacity on FPGA, RTL became the fundamental concept in desinging ciruit. RTL main languages are VHDL and Verilog.

However, these languages are complex because they require deep hardware knowledge, and its very close to the gate level. It is the equivalent of **assembly** in the software world.

- HLS: it is using C and C++ to map complex algorithm to the hardware, with low knowdlege on the hardware.

The HLS is not simply a lanuage, but also an optimzation technique and some toolset, which can automatically generate the RTL code for us

Design Level

Design Level:

- *To review and Adapt the design level points later*
- *see section 2, video 5, part 1*

1.3.2 FGPA and CPU

We hilight the differences between CPU and FPGA

- CPU: a computing platform with fixed architecture, with fixed hardware architecture, controlled by a set of commands or instructions.
- FPGA: also a computing platform, but with no dedicated architecture.

Using HLS, each C or C++ code has its own optmized hardware architecture, which leads high performance execution of some algorithm using FGPA.

Also, when CPU tends to translate the programming code into assembly code and executed *sequentialy*, whereas FGPA tools translate the code into a set of hardware block and execut the code in *parallel*.

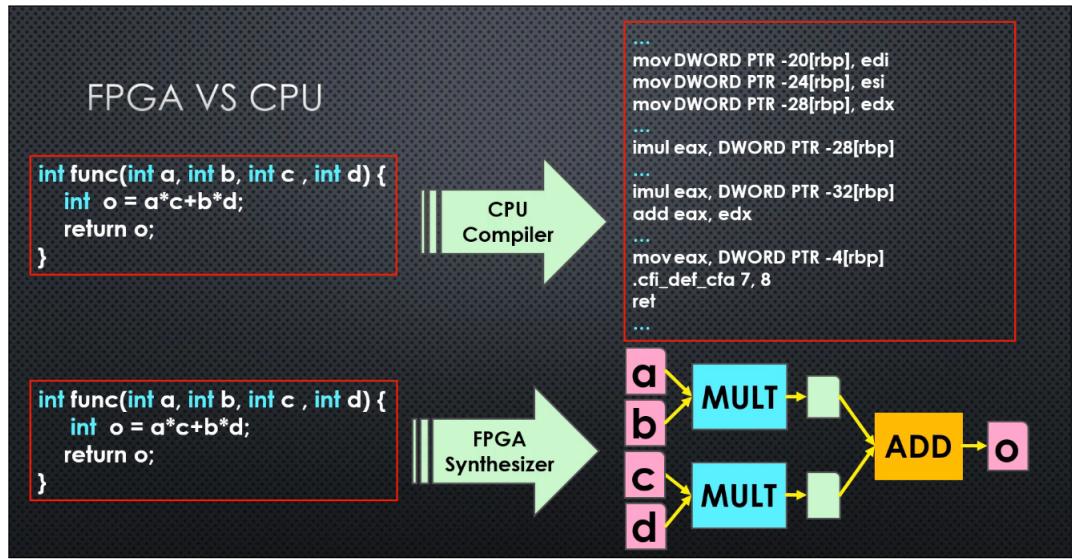


Figure 1.3: FPGA Vs CPU: Sequential vs Parallelism

PA vs CPU
allelism
. Sequential

FGPA vs CPU Parallelism and Sequential:

- Source: section 2, video 6, course 1
- To reinsert the picture where we illusrtate adding operation using CPU model (a register and an ALU)
- To read this model from the microcontroller book, how a program is executed on a CPU

1.3.3 FPGA Structure

Internal FGPA structure is invisible to the user in HLS programming, but we explore the main components so we can understand later the tools reports.

From a concept (or software point of view), an FPGA consist of 2 main layer (as shown in Figure 1.4):

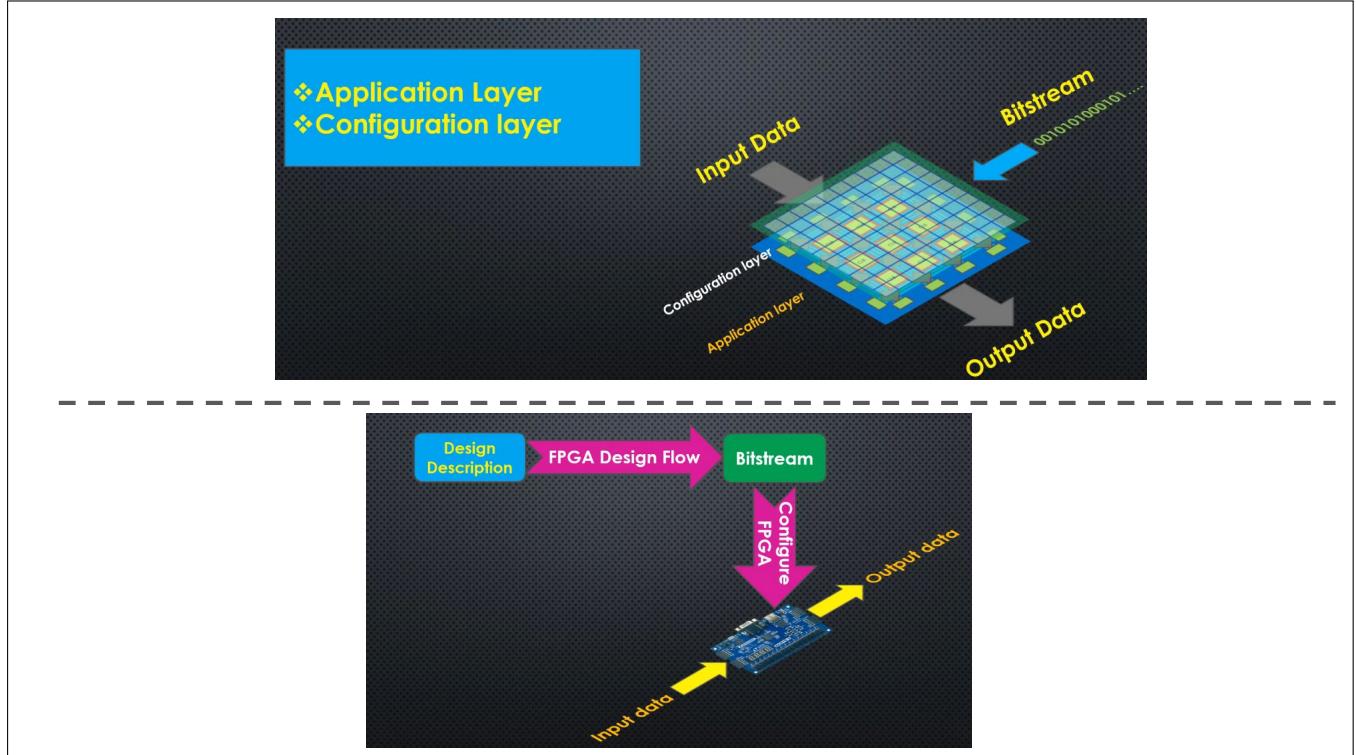


Figure 1.4: FPGA Structure: layer view

1. Application layer: contains all the hardware module to implement some algorithm
Once this layer is configured we can apply the input data to obtain some output
2. Configuration layer: consist of memory cells to configure the application layer.
The configuration data saved in this module is called *bit stream*, and it is generated by synthesis tools.

The role of software design suit is to transform the description language into bitstream, then use this bitstream to implement some circuit using the FPGA, as shown in 2nd picture of Figure 1.4

synthesis tool and bitstream: To review later this idea after doing some applications.

synthesis tool
and bitstream

Now from a physical side, an FGPA composition is shown in [Figure 1.5](#).

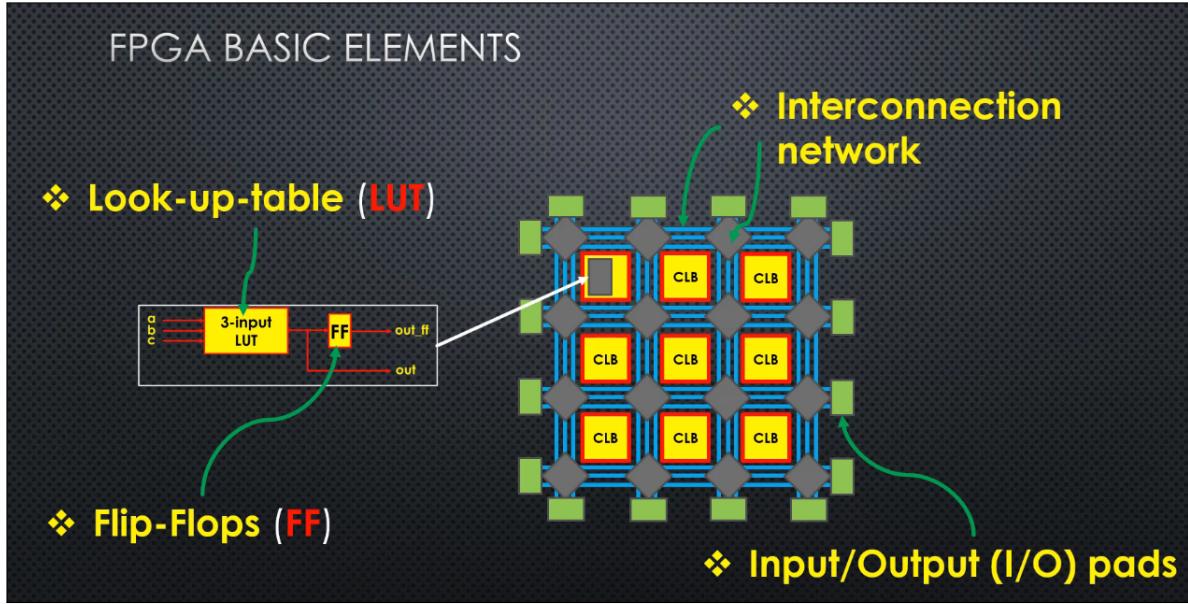


Figure 1.5: FPGA Structure: pyhsical composition

- A look up table (LUT): to implement some logic function
- A flip flop: a register which saves the output of LUT
- IO ports: for data manipulation
- Interconnection network: to connect different resources of FGPA

1.3.4 Look Up Table

A LUT keeps the output of some logic function for all the input combinations. Examples are shown in [Figure 1.6](#)

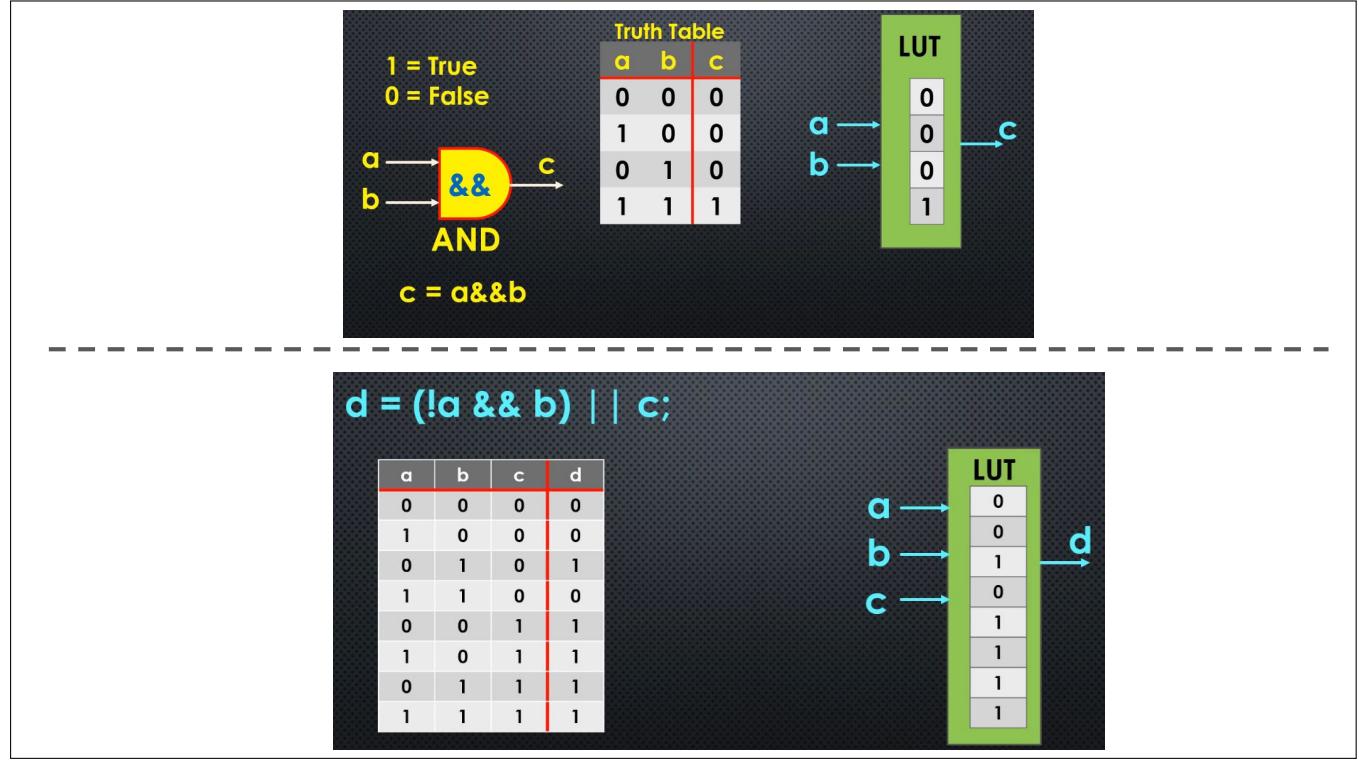


Figure 1.6: LUT Examples

- For a basic AND gate, a LUT needs only 2 inputs to track down different output
- For 2nd example, we have 3 inputs leading to 8 outputs. The LUT need 3 inputs to track down the output.

1.3.5 LUT in FPGA

Now in FPGA, a LUT has a configuration memory that keeps all the possible outputs of a logic function.

It also has a set of inputs to select the proper cell in the memory corresponding to the inputs logic.

LUT in FPGA:

LUT in FGI

- Source: course 1, section 2, video 8
- To redo the writing later
- Main idea
 - A LUT in FPGA uses multiplexer circuit to select the proper output needed, depending on the selection line of the MUX circuit
 - Write down the idea of MUX later

1.3.6 Flip Flop

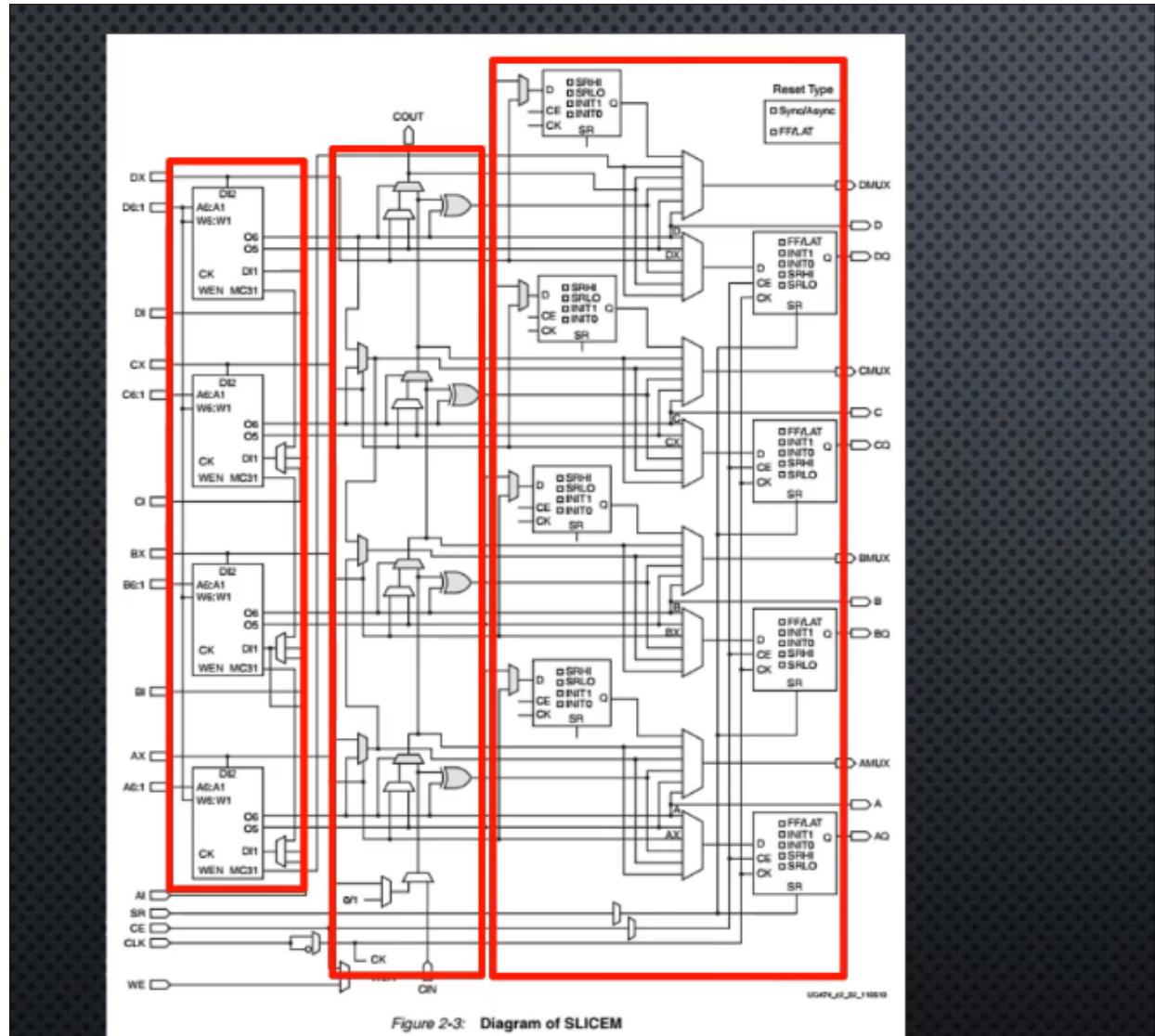
In addition to LUT, an FGPA consist also of flip flop, which is a register which can save output into memory. By having flip flop, we can implement design which depends also on the *past history*, and not only of the present states of the input as in pure combinational logic.

1.4 FPGA Example

Main idea:

- An FPGA is consisting of *many slices*
- Each slice consist of LUT, flip flop and some basic logic gates

Figure 1.7 present a slice in Xilinx 7-series, which is one of the most common Xilinx FPGA families.



Source: Xilinx “7 Series FPGAs Configurable Logic Block: User Guide” UG474

Figure 1.7: FPGA Slice Example

It has three layers of elements:

1. 6 LUT
2. An arithmetic logic circuit
3. Flip flop
4. A MUX

Now a ***set of 2 slices and a switch matrix*** to connect the slices to the routing on the FPGA is called a ***configurable logic block*** or ***CLB*** for short. **An FPGA** can be considered **as a 2D array of CLBs** as shown in Figure 1.8.

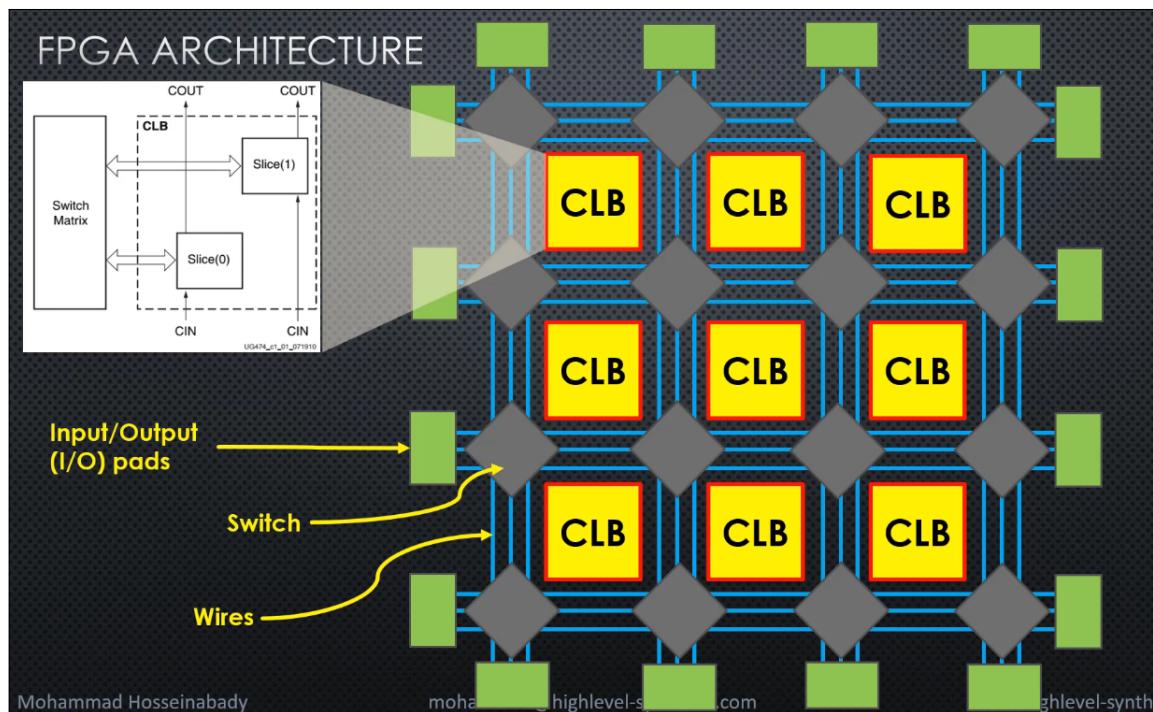


Figure 1.8: FPGA as a 2D array of CLB

Different CLB are connected through switches and wires to form different complex logic circuit.

Now an to a better efficient circuit to implement complex algorithm, an modern FGPA structure ***comes with different additional elements*** such as:

- Block RAM (BRAM) to save large amount of data
- PLL for driving fpga at different clock rate
- High speed serial transceivers
- Multiply and Accumulate DSP block

The FGPA ship along with these extra elements form an development board. The Basys 3 development board will be explored next in 1.5.

1.5 Basys 3 Board Anatomy

Figure 1.9 presents Basys 3 development board, which we will use throughout our development HLS software.

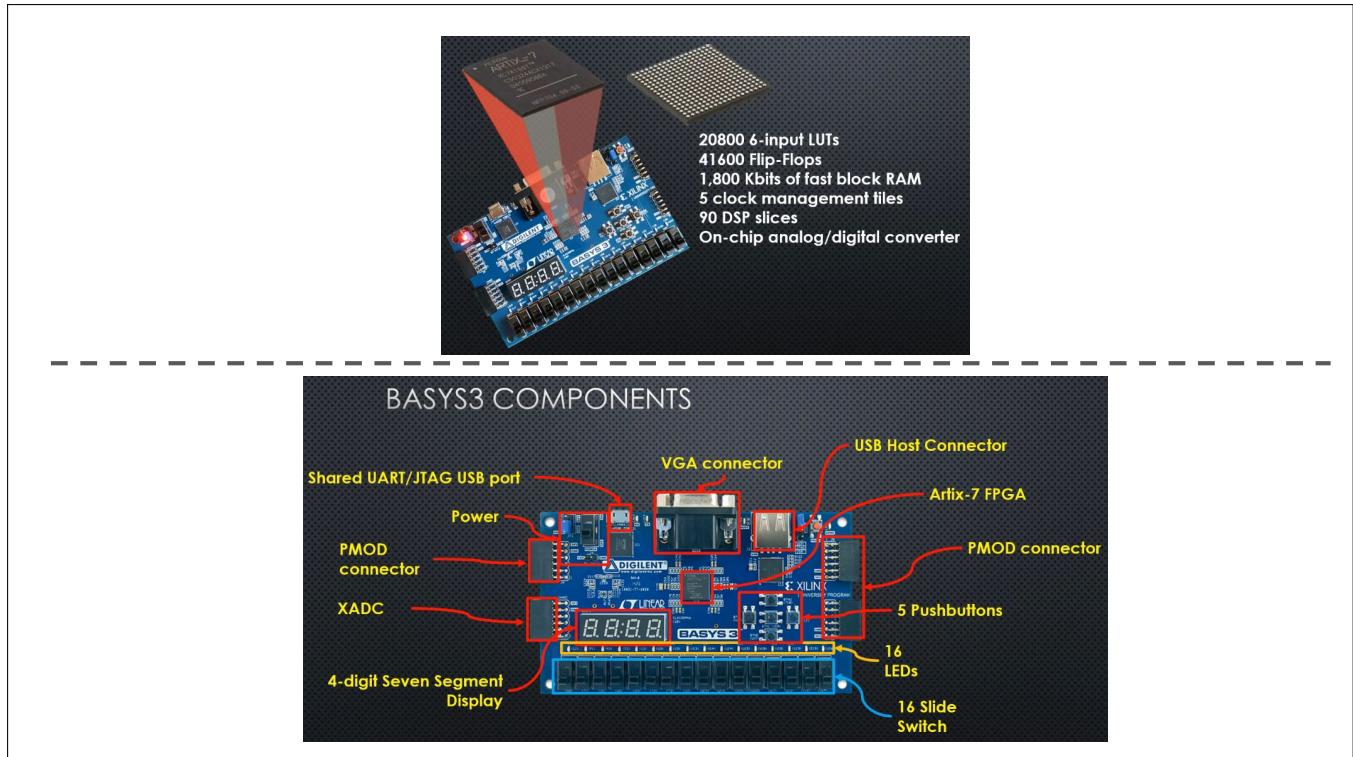


Figure 1.9: Basys 3 board

- The LED are arranged in a common cathode structure
- The LED of the board are connected to a 330 ohm resistors to prevent damage in case of short circuits

cathod and anode:

- To review these concepts later, and insert them into an appendix

cathod and
anode

1.5.1 Power Source

Figure 1.10 present the power circuit for the fpga.

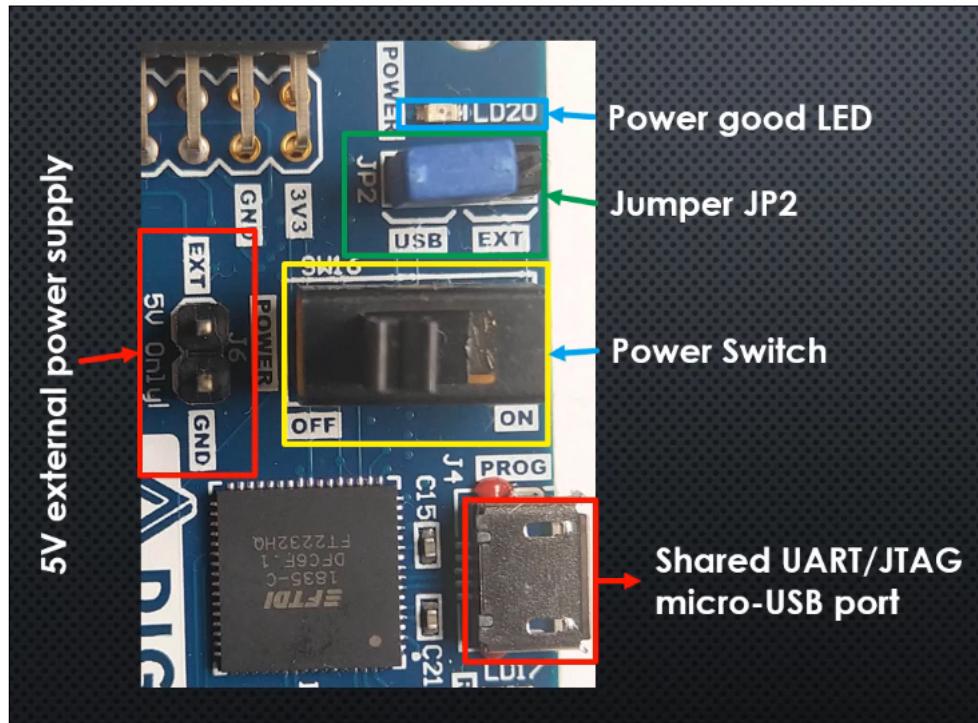


Figure 1.10: Basys 3 power circuit

The board can take power either from shared UART/JTAG port, or from 5v external board. A jumper j2 is used to select which source we choose, and a power switch can be used to turn on or off the board. The LD20 Led is used to indicate if power is at normal state throughout the board.

1.5.2 FPGA Configuration

After powering up the FPGA, it must be configured. We have 3 modes for configuration as shown in [Figure 1.11](#).

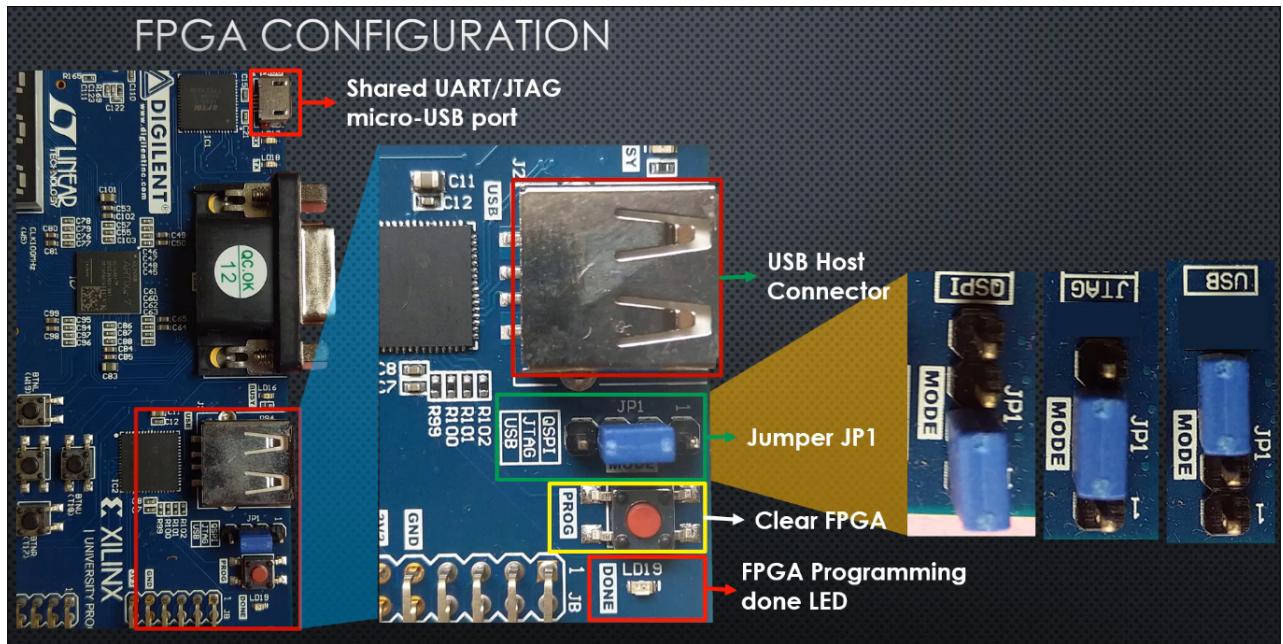


Figure 1.11: Configuration Modes

1. Through the shared JTAG usb port
2. Using a file from a usb connector
3. Using the SPI board

FPGA configuration:

- *Source: course 1, section 2, video 10*
- *To review this section later after doing some examples*

FPGA configuration

Once the programming done, the LD19 will be on to indicate firmware flashing has been done.