Objectives:

Digital IC designer need to be perfect of HDL tools such as (VHDL –VERILOG –System Verilog –System C) as these tools used extensively when we design a complete Digital system that is consist of million or billions of transistors in a single die actually each group of transistors are connected to perform a simple block function this simple block with other blocks complete our system so the task of the digital designer is to write an HDL code that describe these blocks and the connection between them to make the system. Then the next task is to implement this HDL code on an FPGA or as an ASIC as needed.

1.1 Introduction

Developing a large FPGA-based system is an involved process that consists of many complex transformations and optimization algorithms. Software tools are needed to automate some of the tasks. We will use the Web version of the Xilinx ISE package for synthesis and implementation.

1.2 Classification of Device Technologies:

- Discrete logic
- Programmable devices (FPGAs)
- ASIC (Application specific IC)

1.3 System representation and abstraction:

1.3.1 System representation

- Behavioral level
- Structural level
- Physical level

1.3.2 System abstraction

- Transistor level
- Gate level
- Register transfer level (RTL)
- Processor Level

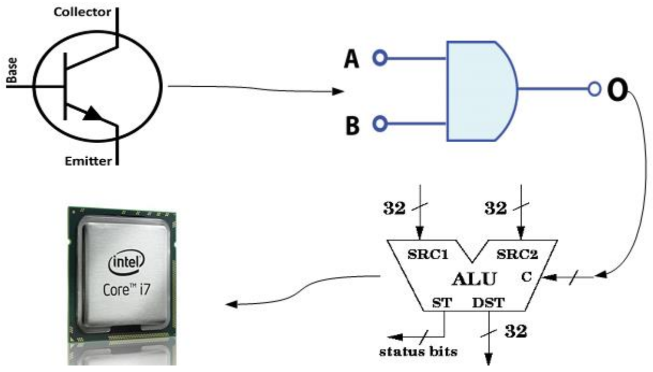

Fig.1. System abstraction levels

2.1 What is FPGA?

FPGA stands for "Field Programmable Gate Array". FPGA essentially is a huge array of gates which can be programmed and reconfigured any time anywhere. "Huge array of gates" is an oversimplified description of FPGA. FPGA is indeed much more complex than a simple array of gates. Some FPGAs has built-in hard blocks such as Memory controllers, high-speed communication interfaces etc. But the point is, there are a lot of gates inside the FPGA which can be arbitrarily connected together to make a circuit of your choice. Like connecting individual logic gate ICs (again oversimplified but a good mental picture nonetheless). FPGAs are manufactured by companies like Xilinx, Altera, Microsemi etc.

2.2 What is FPGA programming?

FPGA programming or FPGA development process is the process of planning, designing and implementing a solution on FPGA. The amount and type of planning vary from application to application. But creating a requirements document that captures all specific requirements and creating a design document that explains how the proposed solution would be implemented can be very helpful to enumerate potential problems and plan around them. A little bit of time spent creating a quality design document will save tons of time in refactoring, debugging and bug fixing later. Implementing a solution on FPGA includes building the design using one of the design entry methods such as schematics or HDL code such as Verilog or VHDL, Synthesizing the design (Synthesis, netlist generation, place and route etc..) in to output files that FPGAs can understand and program the output file to the physical FPGA device using programming tools. Entering the design using schematics is not used in the industry widely anymore. So, we will keep the discussion limited to design entry using HDL (Hardware Description Language), specifically Verilog. All necessary steps to be taken by the user as part of design entry, synthesis and programming will be explained in subsequent sections.

2.3 What are the applications of FPGA?

Architecturally FPGAs are essentially a sea of gates which can be reconfigured to build almost any digital circuit that one can imagine. This great flexibility along with the ability to reconfigure the device with different designs at-will makes FPGA a better choice compared to ASICs (Application Specific Integrated Circuit) for a lot of applications. In certain applications, the number of individual units manufactured would be very small. Designing and manufacturing ASICs for these applications can be prohibitively expensive. In such situations, FPGA can offer very cost effective but robust solutions. Below are some of the potential applications of FPGAs: Cryptography, ASIC prototyping, Industrial, medical and Scientific Instruments, Audio/Video and Image processing and broadcasting, High-performance computing, AI, and Deep Learning, Military and Space applications, Networking, packet processing, and other communications.

2.4 Design flows

The designer facing a design problem must go through a series of steps between initial ideas and final hardware. This series of steps is commonly referred to as the 'design flow'. First, after all the requirements have been spelled out, a proper digital design phase must be carried out. It should be stressed that the tools supplied by the different FPGA vendors to target their chips do not help the designer in this phase. They only enter the scene once the designer is ready to translate a given design into working hardware. The most common flow nowadays used in the design of FPGAs involves the following subsequent phases:

1) Architecture design. This stage involves analysis of the project requirements, problem decomposition and functional simulation (if applicable). The output of this stage is a document which describes the future device architecture, structural blocks, their functions and interfaces.

2) HDL design entry. This step consists in transforming the design ideas into some form of computerized representation. This is most commonly accomplished using Hardware Description Languages (HDLs). The two most popular HDLs are Verilog and the Very High Speed Integrated Circuit HDL (VHDL). It should be noted that an HDL, as its name implies, is only a tool to describe a design that pre-existed in the mind, notes, and sketches of a designer. It is not a tool to design electronic circuits. Another point to note is that HDLs differ from conventional software programming languages in the sense that they don't support the concept of sequential execution of statements in the code.

3) Test environment design. This stage involves writing of test environments and behavioral models (when applicable). They are later used to ensure that the HDL description of a device is correct.

4) Behavioral simulation. This is an important stage that checks HDL correctness by comparing outputs of the HDL model and the behavioral model (being put in the same conditions).

5) Synthesis. This stage involves conversion of an HDL description to a so-called netlist which is basically a formally written digital circuit schematic. Synthesis is performed by a special software called synthesizer. For an HDL code that is correctly written and simulated, synthesis shouldn't be any problem. However, synthesis can reveal some problems and potential errors that can't be found using behavioral simulation, so, an FPGA engineer should pay attention to warnings produced by the synthesizer.

6) Implementation. A synthesizer-generated netlist is mapped onto particular device's internal structure. The main phase of the implementation stage is place and routeor layout, which allocates FPGA resources (such as logic cells and connection wires). Then these configuration data are written to a special file by a program called bitstream generator.

7) Timing analysis. During the timing analysis special software checks whether the implemented design satisfies timing constraints (such as clock frequency) specified by the user.

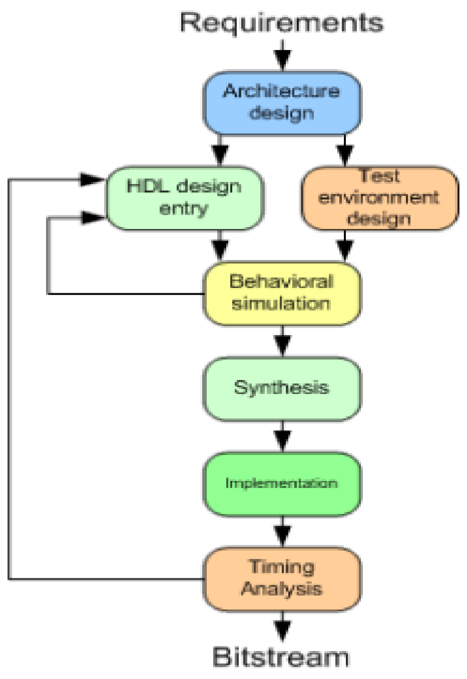

Fig.2. Design flow steps