APPENDIX – A

FPGA DESIGN

A.1 INTRODUCTION TO FPGA

Field Programmable Gate Arrays (FPGAs) are configurable integrated circuits that can be used to design digital circuits and programmable chips. Hardware description languages such as VHDL or Verilog HDL are used to specify the configuration of FPGA. FPGA offers significant advantages in digital logics and integrated circuits because of the reconfigurability feature as well as non-recurring engineering (NRE) cost. It is not like Application Specific Integrated Circuits (ASICs) where designers do not have the flexibility of design modifications after the chip is manufactured by fabrication unit. FPGA is treated like a blank canvas where the design is “painted” according to the constraints of the designer’s needs & the FPGA’s capabilities. A FPGA is a device in which the final logic structure can be directly configured by the end user, without the use of an integrated circuit fabrication. FPGA is similar to a Programmable Logic Device (PLD), but PLDs are generally limited to hundreds of gates, FPGAs support thousands or more gates.

FPGA contains a two dimensional arrays of logic blocks and interconnections between logic blocks. The logic blocks and interconnects both are programmable. These logic blocks are programmed to implement a desired Boolean function and interconnects are programmed using the switch boxes to
connect the logic blocks. Let us consider, if we want to implement a complex design, then the design is divided into small sub functions and each sub function is implemented using one logic block. Now, to get our desired design (CPU), all the sub functions implemented in logic blocks must be connected and this is done by programming interconnects. The internal structure of an FPGA is shown in the figure A.1. In FPGA designer can put the design according to its capability to reprogram it and the device functions according the program is loaded in the FPGA device.

Fig. A.1 FPGA Architecture [102]
FPGAs are alternative approach to the custom ICs, are be used to implement an entire System On one Chip (SOC). The main advantage of FPGA is ability to reprogram. Designer can reprogram an FPGA device to implement a design and this is done after the FPGA is manufactured. It beings the name custom ICs which is due to field programmability. These ICs are expensive and takes longer time to design so they are useful when produced in bulk amounts. But FPGAs follows the shortest time to market and easy to implement within a short time with the help of Computer Aided Designing (CAD) tools.

Xilinx logic block consists of one Look up Table (LUT) [7] and one Flip Flop. An LUT [7] is used to implement number of different functionality. LUTs handle the input lines to the logic block to go into and enable its port. The output of the LUT gives the result of the logic function that it implements and the output of logic block is registered or unregistered output from the LUT. SRAM [7] [21] is used to implement a LUT. For an example a k-input logic function is implemented using $2^k \times 1$ size SRAM. Total number of possible different functions for k input LUT is $2^{2^k}$. The main advantage of such architecture is that it supports implementation of so many logic functions, although the disadvantage is unusually large number of memory cells required to implement such a logic block in case number of inputs is large. Figure A.2 shows a 4-input LUT based implementation of logic block of FPGA. LUT based design provides for better logic block utilization. A logic block having k-input LUT can be implemented in number of different ways with trade off between performance and logic density.
An n-LUT can be shown as a direct implementation of a function truth-table. Each of input combination is hold by the input latch that holds the value of the function corresponding to every input based on truth table. For Example: 2 input LUT can be used to implement 16 types of functions like AND, OR, A+ not B.... etc.

![Fig A.2 Xilinx LUT][102]

Table A.1 Truth table for logic design

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A.1.1 INTERCONNECTS

A wire segment can be described as two end points of an interconnect [8] with no programmable switch between them. There is a sequence of one or more wire segments in an FPGA can be termed as a track. An FPGA has logic blocks, interconnects and switch blocks (Input/output blocks) which are typically interconnected. In FPGA, switch blocks lie in the periphery of logic blocks and interconnect. Switch blocks have wire segments which are connected to logic blocks through switch blocks. Based on the required design, one logic block is connected to another and so on.

Since clock signals are normally routed via special-purpose dedicated routing networks in commercial FPGAs, clock and other signals are separately managed. In the architecture, the locations of the FPGA logic block [8] [9] pins are shown below. Each input of FPGA is accessible from one side of the logic block, although the output pin can connect to routing wires in both the channel to the right and the channel below the logic block. He output pin of each logic block pin can connect to any of the wiring segments in the channels adjacent to it. In the same way, an I/O pad can connect to any one of the wiring segments in the channel adjacent to it. For an example, I/O pad at the top of the chip can connect to any of the W wires (W is the channel width) in the horizontal channel immediately below it. FPGA routing also depends on the number of logic inputs assigned to the LUT.
Generally, the FPGA routing is not in segment form in which each wiring segment spans only one logic block before it terminates in a switch box. With the help of turning on some of the programmable switches within a switch box, it is possible to construct longer paths. Some FPGA architectures use longer routing lines that span multiple logic blocks for higher speed interconnects. There exists a switch box, whenever a vertical and a horizontal channel intersect. In the architecture, when a wire enters a switch box. In switch matrix, there are three programmable switches that allow it to connect to three other wires in adjacent channel segments. The topology or pattern, of switches used in this architecture is the planar or domain-based switch box topology.

In the switch box topology, a wire in track number one connects only to wires in track number one in adjacent channel segments and wires in track number 2 connect only to other wires in track number 2 and so on. The figure A.4 shown below illustrates the connections in a switch box. Switch box consist of wire segments and programmable structure which can be reconfigured many times.
Modern FPGA families expand upon the above capabilities to include higher level functionality fixed into the silicon. The feature of having these common functions embedded into the silicon reduces the requirement of area and gives those functions increased speed compared to building them from primitives. Examples of such logics include logic gates, multipliers, generic DSP blocks, high speed IO logic, embedded processors and embedded memories modules. FPGAs are also widely used for systems validation including pre-silicon validation pre pre synthesis, post-silicon validation or post synthesis, and firmware development. It allows chip fabrication companies to validate their design before the chip is produced in the fabrication plant, reducing the time-to-market.

To shrink the size and power consumption of FPGAs, different vendors such as Tabula and Xilinx have introduced new 3D or stacked architectures [8] following the introduction of its 28 nm 7-series FPGAs [7]. Xilinx revealed that
several of the highest-density parts in those FPGA product lines will be constructed using multiple dice in one package, which employs technology development for 3D construction and stacked-die assemblies. The technology stacks several active FPGA dice side by side on a silicon interposer; a single piece of silicon that carries passive interconnects.
APPENDIX –B

SIMULATION TOOLS

The designing of chip and FPGA implementation includes the following software
development tools.

B.1 XILINX ISE 14.2

Xilinx [7] [8] has been a semiconductor industry leader at the forefront of
technology, market and business achievement. It is a tool to design the IC and to
view their RTL (Register Transfer Level) schematic. It is a tool to test the code on
FPGA environment and the values of all parameters details required to implement
the Chip. The detail of the synthesized results includes the hardware details,
utilization of hardware parameters, memory utilization and timing information.
Device utilization report gives the percentage utilization [13] of device hardware
for the chip implementation. Device hardware includes No of slices, No of flip
flops, No of input LUTs, No. of bounded IOBs, and No of gated clocks (GCLKs)
used in the implementation of design. Timing [13] details provides the
information of delay, minimum period, maximum frequency, minimum input
arrival time before clock and maximum output required time after clock

B.2 MODELSIM 10.1 B

Mentor Graphics [9] was the first to combine single kernel simulator
(SKS) technology with a unified debug environment for Verilog HDL, VHDL,
and System C. The simulation and synthesis combination of industry-leading and native SKS performance with the best integrated debug and analysis environment make Modelsim the simulator of choice for both ASIC and FPGA design. The design platform and standards support in the industry make it easy to adopt in the majority of process and tool flows.

B.2.1 MODELSIM: SIMULATION AND DEBUG

ModelSim EE [9] is the industry-leading, Windows-based simulator for VHDL, Verilog, or mixed-language simulation environments.

B.2.1.1 ModelSim EE Features

- Partial VHDL 2008 support
- It has transaction wlf logging support in all languages including VHDL
- Windows7 a 32 Support
- Secure IP support
- System C option
- RTL and Gate-Level Simulation
- Integrated Debug Environment
- Verilog, VHDL and SystemVerilog Design
- Mixed-HDL Simulation option
- Code Coverage option

B.2.1.2 ModelSim EE Benefits

Model Sim EE has proven the following benefits

- ModelSim EE is a Cost-effective HDL simulation solution,
- It has intuitive GUI for efficient interactive debug,
- Supports to integrated project management simplifies managing project data,
- It is easy to use with outstanding technical support,
- Support for sign-off for popular ASIC libraries,
- It supports hardware debugging
- Used in functional simulation

**B.2.1.3 Simulation and Design Steps**

The diagram has shown in figure B.1 the basic steps for simulating a design in ModelSim.

![Simulations and Design Steps](image)

- **Creating the Working Library:** In ModelSim SE, all programs are designed and compiled into a library. Typically start a new simulation in ModelSim by creating a working library called "work," which is the default library name used by the compiler as the default destination for compiled design units.
• **Compiling Design:** After creating the working library, design is being compiled into it. The ModelSim library format is compatible across all supported platforms.

• **Loading and Running the Simulator with the Design:** With the design compiled, we load the simulator with design by invoking the simulator on a top-level module (Verilog) or a configuration or entity/architecture pair (VHDL). Assuming the design is loaded successfully, and the simulation time is set to zero. There is the requirement to enter a run command to begin simulation.

• Debugging: ModelSim’s robust debugging environment is used to track down the cause of the problem.

**B.3 DESIGN VERIFICATION**

The design is developed in Xilinx tool and functionally checked in Modelsim simulation tool. After the simulation, the design is verified. Verification can be done at different stages of the process steps.

**B.3.1 BEHAVIOURAL SIMULATION (RTL SIMULATION)**

The functional simulation is performed before synthesis process to verify RTL (behavioral) code and to confirm that the design is functioning as intended. Behavioral modeling design and simulation can be performed on either VHDL or Verilog HDL designs. In the simulation process, signals and variables are used to pass intermediate values, procedures and functions are traced and breakpoints are set. In fast simulation designer is allowed to change the HDL code if the required
functionality is not met with in a short time period. Timing and resource usage properties are still unknown, since the design is not yet synthesized to gate level.

**B.3.2 FUNCTIONAL SIMULATION (POST TRANSLATE SIMULATION)**

Functional simulation gives information about the logic operation of the circuit design. Designers can verify the functionality of the design using this process after the Translate process. If the developed code is not meeting the expected functionality, then the designer has to made changes in the code and again follow the design flow steps. Changes in the code and again follow the design flow steps.

**B.3.3 STATIC TIMING ANALYSIS (STA)**

This can be done after MAP or PAR processes. Timing report relating to post MAP lists signal path delays of the design derived from the design logic. Timing report relating to post Place and Route incorporates timing delay information to provide a comprehensive timing summary of the design.

**B.4 IP BASED DESIGN, HARD AND SOFT MACROS**

Logical primitives [18] using cell libraries are usually provided by the device manufacturer as part of the service. They will incur no additional cost and their release will be covered by the terms of a Non Disclosure Agreement (NDA) [29] and they will be regarded as intellectual property by the manufacturer. The physical design based on it is predefined so they could be termed "hard macros". What most engineers understand as IP cores, designs purchased from a third party as sub components of a larger ASIC design. These design can be provided as an HDL description [11], or as a fully routed design that could be printed directly
onto an ASIC’s mask. Many organizations now sell such predesigned cores CPUs, Ethernet, USB or telephone interfaces [88] and larger organizations may have an entire department or division to produce cores for the rest of the organization. There are a lot of functions available in IP design, as a core takes a lot of time and investment to create, its further development cuts product cycle times and reuse dramatically and creates better products.

Additionally, organizations such as open cores are collecting free IP cores paralleling the open source movement in software. Soft macros are often process independent because they can be fabricated on a wide range of manufacturing processes and different manufacturers. Hard core macros are limited and usually further design effort must be invested to migrate to a different process or manufacturer. ASIC design [8] is based on a design flow that uses HDL. Most Electronic Design Automation (EDA) tools used for ASIC flow are compatible with both Verilog HDL and VHDL. In the design flow, the code is synthesized with the help of Xilinx tool. In this process the RTL code is converted into logic gates and flip flops, multiplexers, memory devices etc. The logic gates synthesized will have the same logic functionality as described in the RTL code [102]. In next step, a synthesis tool is required to convert the RTL code to logic gates. Most common tools used in the ASIC industry include Synopsys’s Design Compiler, Mentor Graphics, Xilinx and Cadence’s Ambit. The synthesis process requires two other input files to make the conversion from RTL to logic gates.