APPENDIX-A

ITU CHANNEL MODEL

A.1 CHANNEL MODELS FOR LTE

Channel models are heavily dependent upon the radio architecture. A scalable multi-cell architecture with Non-Line-of- Sight (NLOS) conditions becomes necessary. Typically, the scenario is as follows:

- Cells are less than 10 km in radius, variety of terrain and tree density types
- Under-the-eave/window or rooftop installed directional antennas (2 10 m) at the receiver
- 15 - 40 m BTS antennas
- High cell coverage requirement.

The wireless channel is characterized by:

- Path loss (including shadowing)
- Multipath delay spread
- Fading characteristics
- Doppler spread
- Co-channel and adjacent channel interference

It is to be noted that these parameters are random and only a statistical characterization is possible. The above propagation model parameters depend upon terrain, tree density, antenna height and beam width, wind speed, and season (time of the year). The wireless channels can be modeled using AWGN (Additive White Gaussian noise, ITU Vehicular and Pedestrian models [96]. The multipath fading is modeled as a tapped-delay line with 6 taps with non-uniform delays. The gain associated with each tap is characterized by a distribution (Rician with a K-factor > 0, or Rayleigh with K-factor=0) and the maximum Doppler frequency. For each tap, the method of filtered noise is used to generate channel coefficients

with the specified distribution and spectral power density. The definition of the 4 specific ITU channels [97] is shown in the following table A.1 and A.2

Where P = Power in each tap in dB, K = Rician K-factor in linear scale, Tau = tap delay in μs, Doppler = Doppler maximal frequency parameter in Hz,

K-factor: The k-factor is defined as the ratio of the power in the fixed component to the power in the variable component. If it is zero, the channel is Rayleigh. For larger values, the channel is assumed to be Rician in nature.

Table A.1 Extended Pedestrian Model of ITU

Tap	***Tap Delay (μs)***	***Power (P) in dB***
1	0	0
3	0.03	-1.0
4	0.07	-2..0
5	0.09	-3.0
6	0.11	-8.0
7	0.19	-17.2
8	0.41	-20.8

Table A.2 Extended Vehicular Model of ITU

Tap	***Tap Delay (μs)***	***Power (P) in dB***
1	0	0
3	0.03	-1.4
4	0.15	-1.5
5	0.31	-3.6
6	0.37	-0.6
7	0.71	-9.1
8	1.09	-7.0
9	1.73	-12.0
10	2.51	-16.9

APPENDIX-**B**

SYSTEM DESIGN ON FPGA

B.1 INTRODUCTION TO FPGA

Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The Xilinx family offers densities ranging from 100,000 to 1.6 million system gates. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

B.1.1 ***Architectural Overview***

The FPGA architecture consists of five fundamental programmable functional elements as shown in Fig B.1 (a) [98]:

- **Configurable Logic Blocks (CLBs)** contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.
- **Input/ Output Blocks (IOBs)** control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Supports a variety of signal standards, including four high-performance differential standards. Double Data-Rate (DDR) registers are included.
- **Block RAM** provides data storage in the form of 18-Kbit dual-port blocks.
- **Multiplier Blocks** accept two 18-bit binary numbers as inputs and calculate the product. The FPGA's provide 4 to 36 dedicated multiplier

blocks per device. The multipliers are located together with the block RAM in one or two columns depending on device density.

- **Digital Clock Manager (DCM) Blocks** provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase-shifting clock signals.

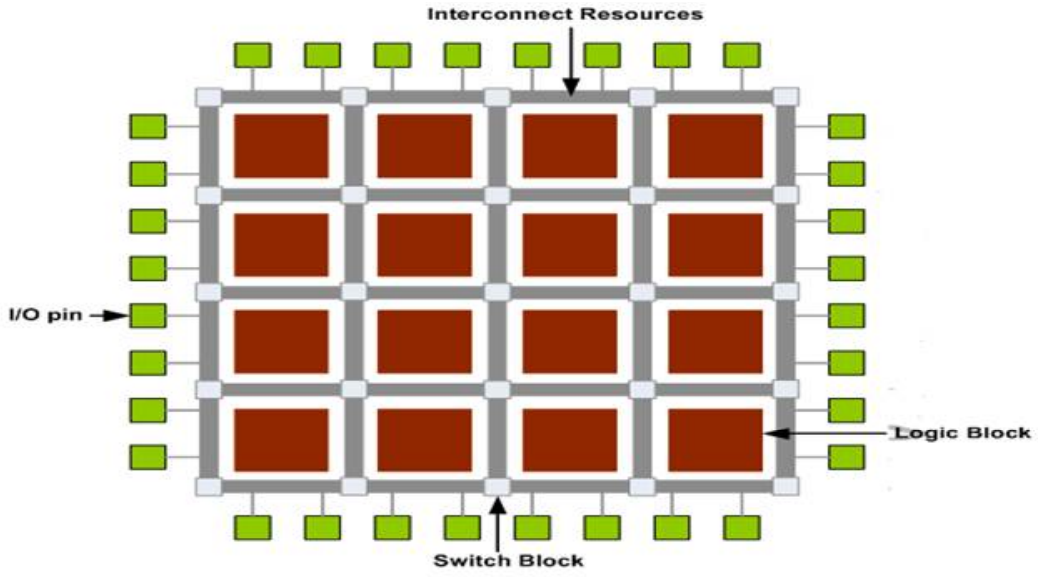

Fig B.1 FPGA Architecture

B.2 Xilinx ISE TOOL FLOW

The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). FPGAs can be used to implement any logical function that an ASIC could perform. The Integrated Software Environment (ISE) is the Xilinx design software suite that allows you to take the design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes the design through the following steps in the ISE design flow.

B.2.1 Design Entry:

Design entry is the first step in the ISE design flow. During design entry, you create the source files based on the design objectives. You can create the top-level design file using a Hardware Description Language (HDL), such as VHDL, Verilog, or ABEL, or using a schematic. You can use multiple formats for the lower-level source files in the design. If work starts with a synthesized EDIF or NGC/NGO file, design entry and synthesis steps can be skipped and start with the implementation process.

B.2.2 Synthesis:

After design entry and optional simulation, you run synthesis. During this step, VHDL, Verilog, or mixed language designs become netlist files that are accepted as input to the implementation step.

B.2.3 Implementation:

After synthesis, you run design implementation, which converts the logical design into a physical file format that can be downloaded to the selected target device. From Project Navigator, you can run the implementation process in one step, or you can run each of the implementation processes separately. Implementation processes vary depending on whether you are targeting a Field Programmable Gate Array (FPGA) or a Complex Programmable Logic Device (CPLD).

B.2.4 Verification:

You can verify the functionality of the design at several points in the design flow. You can use simulator software to verify the functionality and timing of the design or a portion of the design. The simulator interprets VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. Simulation allows you to create and verify complex functions in a relatively small amount of time. You can also run in-circuit verification after programming the device.

B.2.5 Device Configuration:

After generating a programming file, you configure the device. During configuration, you generate configuration files and download the programming files from a host computer to a Xilinx device.

Synthesis using Xilinx can be done by following steps:

1) Now create a new project and select device VertexX5pro and then XC5VLX130T

2) Now add source code.

3) Go to Implementation

4) Synthesis XST

5) Now to change to Behavioral simulation

6) Run the source code

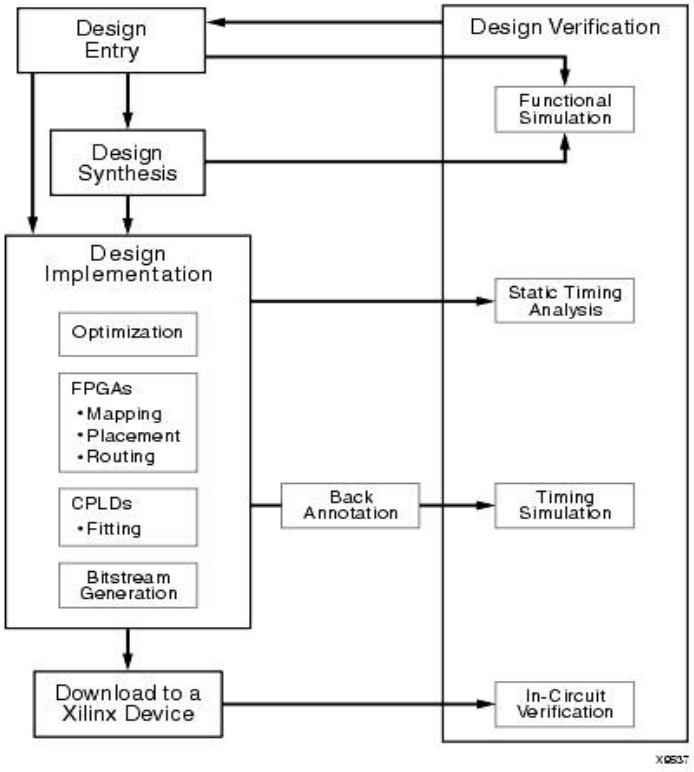

Fig B.2 Xilinx ISE Design Flow

B.3 CHIPSCOPE

Chipscope is an embedded, software based logic analyzer. By inserting an "integrated controller core" (icon) and an "integrated logic analyzer" (ila) into your design and connecting them properly, you can monitor any or all of the signals in your design. Chipscope provides you with a convenient software based interface for controlling the "integrated logic analyzer," including setting the triggering options and viewing the waveforms. Below Figure shows a block diagram of a Chip Scope Pro system. Users can place the ICON, ILA, VIO, and ATC2 cores (collectively called the Chip Scope Pro cores) into their design by generating the cores with the Core Generator and instantiating them into the HDL source code. We can also insert the ICON, ILA, and ATC2 cores directly into the synthesized design netlist using the Core Inserter tool. The design is then placed and routed using the ISE 14.2 implementation tools. Next, we download the bit stream into the device under test and analyze the design with the Analyzer software.

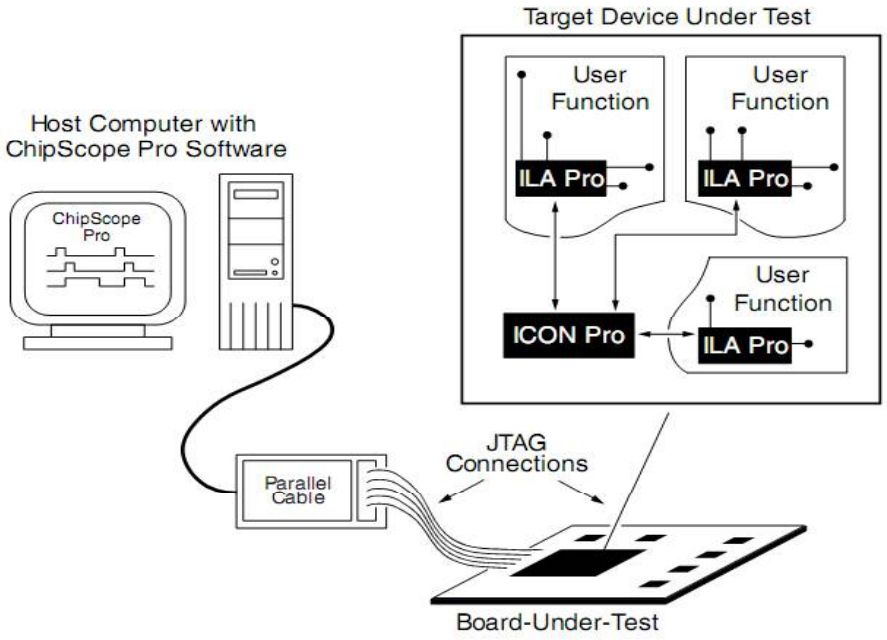

Fig B.3 Chip Scope Pro Cores Description

B.3.1 ICON Core:

All of the cores use the JTAG Boundary Scan port to communicate to the host computer via a JTAG download cable. The ICON core provides a communications path between the JTAG Boundary Scan port of the target FPGA and up to 15 ILA, IBA/OPB, IBA/PLB, VIO, and/or ATC2 core. For devices not of the Virtex-4 or Virtex-5 families, the ICON core uses either the USER1 or USER2 JTAG Boundary Scan instructions for communication via the BSCAN_VIRTEX primitive. The unused USER1 or USER2 scan chain of the BSCAN_VIRTEX primitive can also be exported for use in your application, if needed.

For Virtex-4 and Virtex-5 devices, the ICON core uses any one of the USER1, USER2, USER3 or USER4 scan chains available via the BSCAN_VIRTEX primitives. In Virtex-4 and Virtex-5 devices, it is not necessary to export unused USER scan chains because each BSCAN_VIRTEX primitive implements a single scan chain.

B.3.2 ILA Core:

The ILA core is a customizable logic analyzer core that can be used to monitor any internal signal of your design. Since the ILA core is synchronous to the design being monitored, all design clock constraints that are applied to your design are also applied to the components inside the ILA core. The ILA core consists of three major components:

- Trigger input and output logic:
  - ♦ Trigger input logic detects elaborate trigger events.
  - ♦ Trigger output logic triggers external test equipment and other logic.
- Data capture logic:
  - ♦ ILA cores capture and store trace data information using on-chip block RAM Resources.
- Control and status logic:
  - ♦ Manages the operation of the ILA core.

B.4 MODELSIM FOR SIMULATION

Modelsim is a verification and simulation tool for VHDL, Verilog, SystemVerilog, systemC, and mixed-language designs. Modelsim optimizations are automatically performed on all designs. These optimizations are designed to

maximize simulator performance, yielding improvements above 10X, in some Verilog designs, over non-optimized runs.

The following diagram shows the basic steps for simulating a design in ModelSim.

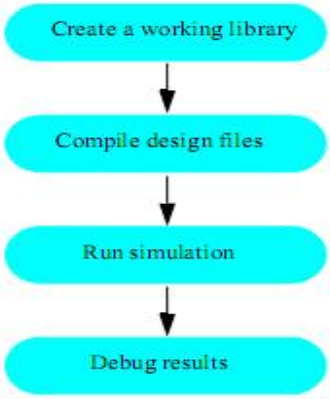

Fig B.4.Modelsim Simulation Flow

- *Creating the working library*: - In ModelSim, all designs, be they VHDL, Verilog, or some combination thereof, are compiled into a library. You typically start a new simulation in ModelSim by creating a working library called "work". "Work" is the library name used by the compiler as the default destination for compiled design units.
- *Compiling your design*:- After creating the working library, you compile your design units into it. The ModelSim library format is compatible across all supported platforms. You can simulate your design on any platform without having to recompile your design.
- *Running the simulation*:- With the design compiled, you invoke the simulator on a top-level module (Verilog) or a configuration or entity/architecture pair (VHDL). Assuming the design loads successfully, the simulation time is set to zero, and you enter a run command to begin simulation.

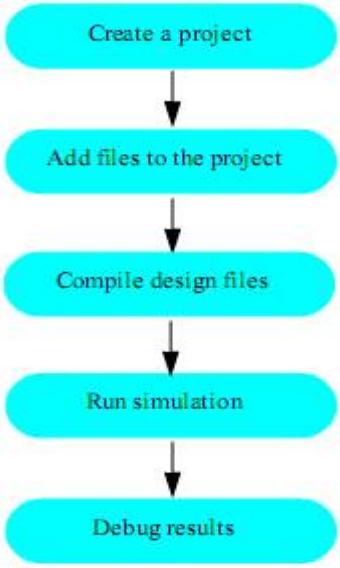

Fig B.5 Project Flow

B.5 VHDL

VHDL is a high level description language for system and circuit design. The language supports various levels of abstraction. In contrast to regular net list formats that supports only structural description and a Boolean entry system that supports only dataflow behavior, VHDL supports a wide range of description styles. These include structural descriptions, dataflow descriptions and behavioral descriptions.

The structural and dataflow descriptions show a concurrent behavior. That is, all statements are executed concurrently, and the order of the statements is not relevant. On the other hand, behavioral descriptions are executed sequentially in processes, procedures and functions in VHDL. The behavioral descriptions resemble high-level programming languages.

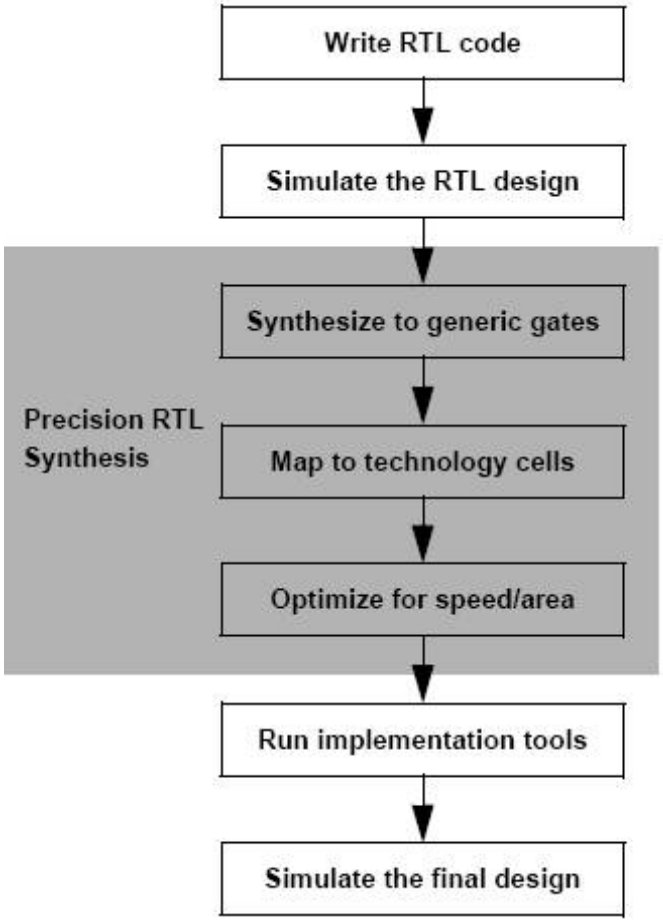

Fig B.6.Top-Down Design Flow with Precision RTL Synthesis

VHDL allows a mixture of various levels of design entry abstraction. Precision RTL Synthesis Synthesizes will accept all levels of abstraction, and minimize the amount of logic needed, resulting in a final net list description in the technology of your choice. The Top-Down Design Flow is shown in Figure B.6.