Indian Institute of Technology Tirupati

EE536P: FPGA Laboratory

Credit: L-T-P-C : 0-0-3-2
Jan-June 2023

Experiment 1: Logic Gates and Combinational Circuits
Date: 5th Jan 2023
Class: M.Tech

In this exercise we will be learning about how does the Verilog language be used to describe the digital gates and circuits. The Verilog describes the circuit in different ways at behavior level, data flow level and gate level. The behavioral description of a circuit is the top level of abstraction where the circuit will be described using the input and output relationship, it does not describe the structure of a given circuit as such. Whereas the data flow level which is also known as register transfer level describes the flow of data among different components/variables. In register transfer level description most of the variables are of register type. The RTL description provides more details than the behavioural description. Similarly to describe a circuit with still more detail the Verilog provides gate level description of the design. The gate level description clearly provides the structure of a given design. In this way, the Verilog has provision for three different ways of describing a given design. So, in a sense, all the three descriptions ranges from behavior to structure in order to describe a given design. All these three level of description can be used simultaneously as needed to describe a given design.

The Lab has divided into two parts:

1. **Part-A:**Design and verification of the digital circuits using Verilog HDL and using Testbench
2. **Part-B:**Verification of the functionality of the above designs on FPGA by dumping the generated bit-stream file.

Part-A: In this lab exercise, we will design and implement the following using Verilog HDL.

1. Logic gates such as NOT, AND, OR, XOR, XNOR, NAND, NOR gates and $M \times 1$ Multiplexer.
2. As we know that that **any circuit logic can be implemented using NAND and NOR gates**. Prove this statement by implementing all the logic gates using only NAND and NOR gates (separately) by writing its corresponding Boolean expressions and implement the same the by writing it's equivalent verilog HDL and verify the same by writing it's corresponding test-bench. Compare your simulation results with your theoretical truth table.
3. Write a gate-level Verilog HDL code to design a 8-input 1-output XOR gate using 2-input XOR gates. Verify the correctness of your design for all the possible inputs. Write a test bench program to generate all the 2^8 inputs and apply them to the design and check whether the design is producing the correct output.
4. Design and Implement N-bit Adder and write the test bench program to verify the functionality of the adder design by applying the random inputs to the test design. (**Note: It needs to perform both signed and unsigned number addition**)
5. Design and Implement $M \times N$ Signed and Unsigned Array multiplier. Write the test bench program to verify the functionality of the multiplier design by applying the random inputs to the test design.
6. Design and implement by incorporating all the above experiments into one top level module, which is having 3 input ports A, B, C and one output port Z. Where A and B are N-bit size and choose an appropriate size of the selection line C and Z is N+1 bit size. The top level module needs to work on the following basis:
   - if C is 0, the output must be Logical AND operation
   - if C is 1, the output must be Logical OR operation

- if C is 2, the output must be Logical NAND operation
- if C is 3, the output must be Logical NOR operation
- if C is 4, the output must be Logical XOR operation
- if C is 5, the output must be Logical XNOR operation
- if C is 6, the output must be $M \times 1$ Mux operation
- if C is 7, the output must be Adder operation
- if C is 8, the output must be multiplier operation

Based on the above analysis, name the top level module and verify its functionality by writing its test bench program.

Part-B: Generate the Bit-stream file for the design stated in Part-A, 6^{th} question, and dump the generated bit file to Edge Artix-7 FPGA board. Verify the obtained simulation results in Part-A on FPGA by applying the test vector by choosing the proper sliding switches and LEDs (If required, select a seven-segment display also). (Choose N = 3)

Part-C: Design and implement the operations mentioned and Table-1 using verilog hdl and generate the bitstream file and do the following:

1. Display the Output Y using LEDs
2. write verilog code to convert the binary output to BCD number
3. Display the same by interfacing Seven Segment Display.

Where the inputs A and B are 5-bit size and the selection line is 2 bit size and output Y is maximum of 10 bits. **Note that A and B are signed numbers**

Select	Operation
0	Y = A + B
1	Y = A - B
2	Y = $A \times B$
3	Y = max(A,B)

Table 1: Simple ALU Operations

Guidelines and Reporting:

- Use mixed level of modelling for writing the verilog codes
- Each design must be verified using the test bench
- The output and input should be displayed in the waveform format
- Each exercise should be reported in the report which would contain the gate-level diagram of the design (or block diagram), the truth table, and waveforms, and description of the functionality.
- The the report should have following sections:
  1. **Objective:** Write a paragraph on the objective of the exercise.
  2. Design Description,
     Block diagram,
     Truth Table.
  3. Experimental Results
     Waveforms: take the screen short from the tool and put in a proper visible format with figure numbering. Describe what it depicts.

4. **Summary and Lesson:** Summarise the exercise done and what are the lesson that you have learned.

1 Resources

1.1 Textbooks and video lectures

We may refer the following resources to learn the Verilog HDL

1. Hardware Modelling using Verilog HDL.
   `https://www.youtube.com/playlist?list=PLJ5C_6qdAvBELELTSPgzYkQg3HgclQh-5`
2. Verilog HDL: A Guide to Digital Design and Synthesis, Second Edition By Samir Palnitkar
   `https://d1.amobbs.com/bbs_upload782111/files_33/ourdev_585395BQ8J9A.pdf`
3. `https://www.chipverify.com/verilog/verilog-tutorial`
4. `http://www.csit-sun.pub.ro/courses/Masterat/Xilinx%20Synthesis%20Technology/toolbox.xilinx.com/docsan/xilinx4/data/docs/xst/hdlcode2.html`
5. Digital Design: An Embedded Systems Approach Using Verilog by Peter Ashenden
6. Digital Signal Processing with Field Programmable Gate Arrays by Uwe Meyer-Baese
7. FPGA PROTOTYPING BY VERILOG EXAMPLES by Pong P. Chu

1.2 Short References for Verilog coding

The following link gives the short introduction to the verilog basics and implementation of basic programs such as fulladder etc... in VIVADO, MODELSIM softwares.

`https://www.youtube.com/watch?v=u8Mg-80YJfc&list=PLyBufKJju4dv5NRmTPwTAHtrQsvHIlica`

1.3 Download and Installation of softwares

We may refer the following resources to download and install the softwares such as VIVADO, MODELSIM STUDENT EDITION.

1. Vivado download and installation (Take the software from VLSI LAB JTS (Mr. Kumar)) or follow the below link to download and install
   `https://youtu.be/fBFn32Al0yw`
2. To add the FPGA board files, follow the below-specified link
   `https://youtu.be/cNnhXN1NldI`
3. To install the Quartus and ModelSim student edition, follow the below-specified link
   `https://youtu.be/TA4WEichF_k`