

Hardware Implementation

Raj Kohale

April 2024

1 Introduction

In this section I have done basic program implementation on a Hardware where we use ZYBO Z-7010. Here I have done two types of implementation

1. By using ZYBO kit input/output port
2. BY using Virtual input/output on FPGA.

2 Tools/Requirements

2.1 Xiling Vivado

To simulate and for observing behaviour of different logic gate we need Xiling Vivado tool. Vivado is a software for synthesis and analysis of hardware description language designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis.

2.2 ZYBO Z-7010

The ZYBO (ZYNq BOard) is a feature-rich, ready-to-use, entry-level embedded software and digital circuit development platform built around the smallest member of the Xilinx Zynq-7000 family, the Z-7010. The Z-7010 is based on the Xilinx All Programmable System-on-Chip (AP SoC) architecture, which tightly integrates a dual-core ARM Cortex-A9 processor with Xilinx 7-series Field Programmable Gate Array (FPGA) logic. When coupled with the rich set of multimedia and connectivity peripherals available on the ZYBO, the Zynq Z-7010 can host a whole system design.

3 What I have done?

3.1 Basic Overview

I have written verilog code Binary to Gray converter in Xiling Vivado then by doing simulation I have observed behaviour of program. After that I have set input/output Power supply i.e.(3.3V) and save constrained file with extension XDC and perform RUN Implementation and generate bit stream for program.

After creating bit stream I have connected ZYBO kit with system and turn on kit(power supply for kit 5V). Then from Open hardware manager I have choose target and connect ZYBO kit with Xiling Vivado. After that I have dump file on kit and by changing inputs from kit buttons I have observed output from LEDs.

3.2 Steps to be followed

3.2.1 Step 1

Write the verilog code and test bench for the given execution and save the file in Design Source and Simulation Source.

3.2.2 Step 2

Simulation

Perform the run simulation and observe the simulated result and verify with the inputs which is given from the testbench.

3.2.3 Step3

RTL analysis (Schematic)

Go for the I/O planning and select package pin and set the pin numbers which we are using for inputs and outputs.(pin numbers will be given on the kit). Similarly, set the values of I/O standards for the inputs and outputs. (refer data sheet for the I/O standards values for each input and output). Finally save the file with the extension of .xdc.

3.2.4 Step 4

Synthesis and Run Implementation

Go for the run synthesis, during synthesis in Xilinx Vivado, the tool translates Register Transfer Level (RTL) code, written in Hardware Description Languages (HDLs) (i.e. Verilog) into a netlist. This netlist represents a digital circuit composed of logic gates and flip-flops.

3.2.5 Step5

Hardware Connection

Connect the FPGA board to the Device.

3.2.6 Step6

Program and Debug

Click on Generate Bitstream, it will generate bitstream of program.

Bitstream The bitstream tells the FPGA how to connect its internal building blocks (logic gates and flip-flops) to achieve the desired functionality based on the program.

Select Program device from the open hardware manager and give the bitstream file to the device. It will dump bitstreams on the FPGA board.

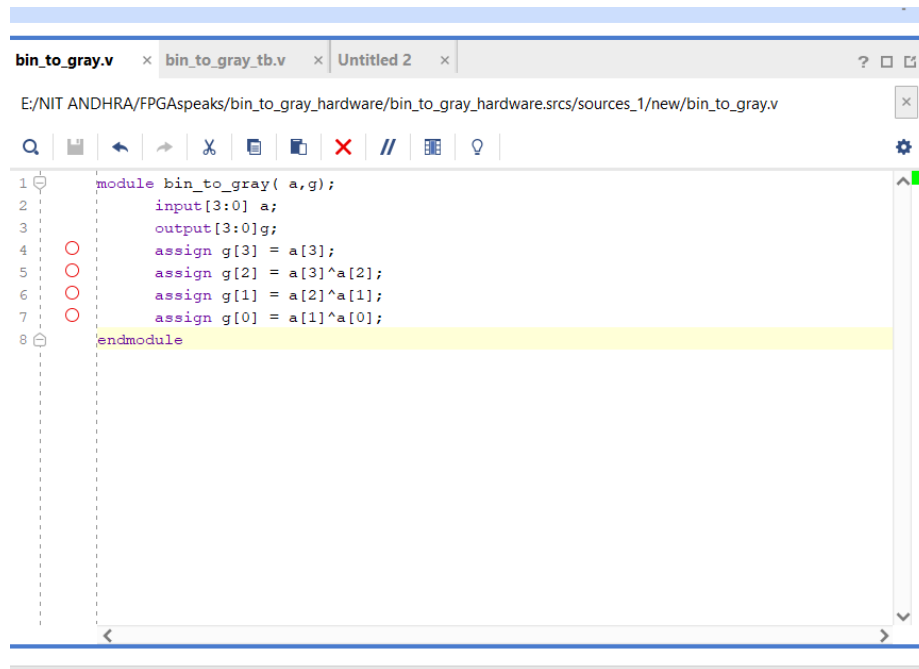
3.2.7 Step 6

Hardware Implementation

Give the different inputs from the inputs buttons and observe the output from the Leds.

4 Observations

4.1 Binary to Gray code

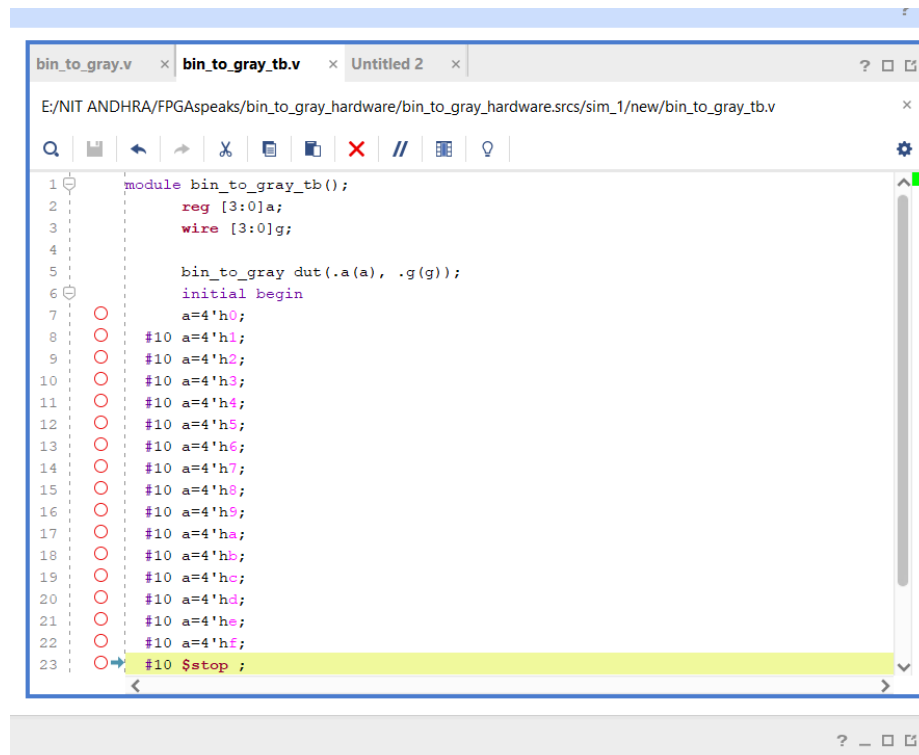


```
bin_to_gray.v  x  bin_to_gray_tb.v  x  Untitled 2  x
E:/NIT ANDHRA/FPGAspeaks/bin_to_gray_hardware/bin_to_gray_hardware.srscs/sources_1/new/bin_to_gray.v

1  module bin_to_gray( a,g);
2      input[3:0] a;
3      output[3:0]g;
4      assign g[3] = a[3];
5      assign g[2] = a[3]^a[2];
6      assign g[1] = a[2]^a[1];
7      assign g[0] = a[1]^a[0];
8  endmodule
```

Figure 1: Verilog code for binary to gray

4.2 Testbench



```
bin_to_gray.v  x  bin_to_gray_tb.v  x  Untitled 2  x
E:/NIT ANDHRA/FPGAspeaks/bin_to_gray_hardware/bin_to_gray_hardware.srscs/sim_1/new/bin_to_gray_tb.v

1  module bin_to_gray_tb();
2      reg [3:0]a;
3      wire [3:0]g;
4
5      bin_to_gray dut(.a(a), .g(g));
6      initial begin
7          a=4'h0;
8          #10 a=4'h1;
9          #10 a=4'h2;
10         #10 a=4'h3;
11         #10 a=4'h4;
12         #10 a=4'h5;
13         #10 a=4'h6;
14         #10 a=4'h7;
15         #10 a=4'h8;
16         #10 a=4'h9;
17         #10 a=4'ha;
18         #10 a=4'hb;
19         #10 a=4'hc;
20         #10 a=4'hd;
21         #10 a=4'he;
22         #10 a=4'hf;
23         #10 $stop ;
```

Figure 2: Testbench

4.3 Simulation

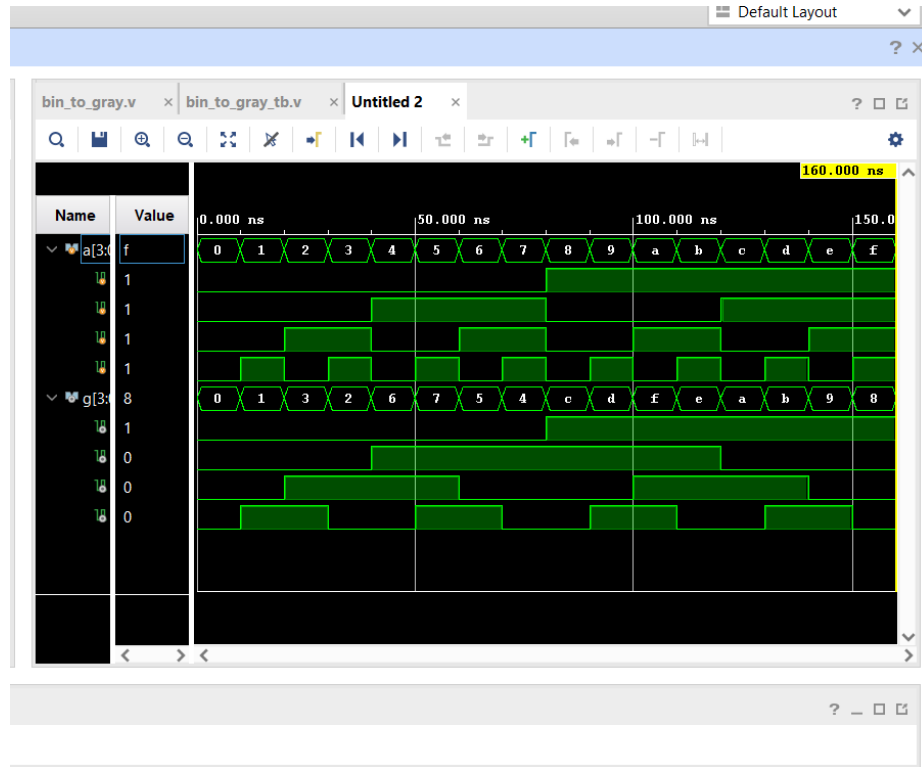


Figure 3: Simulation

4.4 RTL analysis schematic

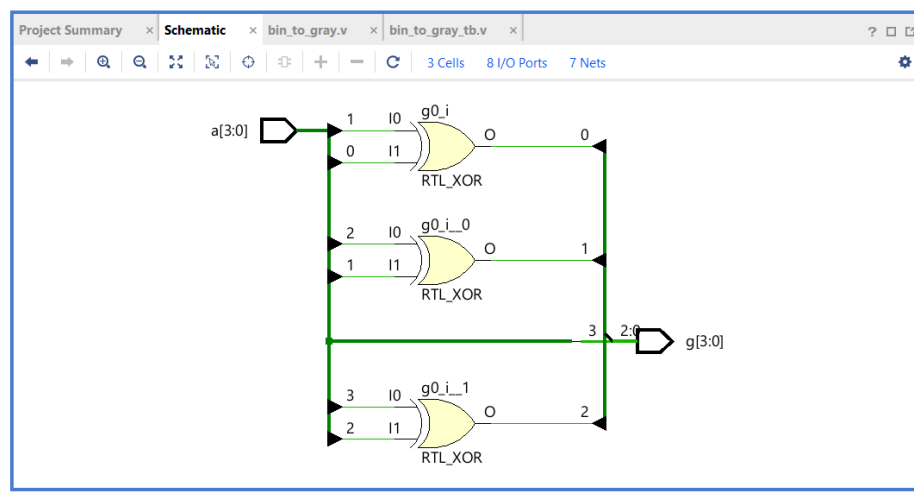


Figure 4: RTL analysis schematic

4.5 FPGA board connection



Figure 5: FPGA board

5 What I have learned?

1. How to write verilog code for the given functionality.
2. By creating testbench for the code observe the simulation output and check the given code matched with functionality or not.
3. Observe the RTL Schematic and do I/O floor planning.
4. How to generate bitstream from the code.
5. How the output will be coming by varying the inputs on the kit, which shows the desired functionality based on the our requirements.

6 Demonstration

[Hardware implementation](#)