EIE330 FPGA Based System Design

Report of Laboratory 1

FPGA Lab1: Full adder, Decoder, and Comparator

Source Code of this report can be found at: EIE330-FPGA-Laboratory

by

PENGRUI K. TANG
Student No: 1220031811
from EIE330 D1,
Faculty of Innovation Engineering,
Universidade de Ciência e Tecnologia de Macau

XIAOYANG S. ZHEN
Student No: 1220034170
from EIE330 D1,
Faculty of Innovation Engineering,
Universidade de Ciência e Tecnologia de Macau

Oct, 2024
(Coloane, Macao SAR)

This report is a part of the result for EIE330 FPGA Based System Design in U.C.T.M

This page is intentionally left blank.

Conteúdo

Capítulo I

Abstract

I Result

Source Code of this report can be found at the git-hub repository: EIE330-FPGA-Laboratory. And the demonstration video of certain codes can be found at the Github release Page or the Youtube video.

All relevant output in this report is shown in the following directory.

- **FPGA Lab1**
  1. 1-0. Report [T][Z]
     (a) Text [T][Z]
     (b) Video [T]
  2. 1-5. Full adder
     (a) Design [Z]
     (b) Coding [Z]
     (c) Test [Z]
  3. 1-6. Decoder
     (a) Design [T]
     (b) Coding [T]
     (c) Test [T]
  4. 1-7. Comparator
     (a) Design [Z]
     (b) Coding [Z]
     (c) Test [Z]

II Contribution

And the contribution of each member is labeled with their last name:

1. Pengrui Tang: [T]
2. Xiaoyang Zhen: [Z]

Capítulo II

Full-Adder

I Introduction

I: i Background

The full adder is the basic component in digital circuits to carry out arithmetic operations. Different from the half adder, the full adder handles the addition of three inputs(A, B, and Carry-in) and output the sum and carry-out of them.

I: ii Purpose

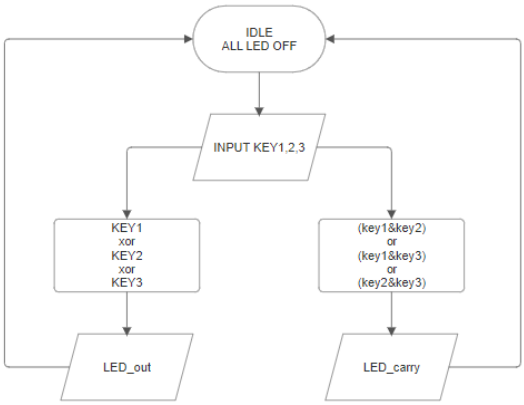
In this experiment, we want to design a full adder using the FPGA board that can handle three single-bit input additions.

I: iii Design and Key-results

The circuit of the full adder is implemented using Verilog on the FPGA board in this experiment. This circuit requires three single-bit inputs and gives out two single-bit outputs, they are Key1, Key2, Key3, LED_out, and LED_carry respectively. The outputs are displayed on the development board using LED lights.

II Materials and Methods

II: i Structure Block Diagram



II: ii Truth Table

Key_1	Key_2	Key_3	C_o	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

II: iii Resources in Board

We used LED0 to LED1 and KEY0 to KEY2 on the development board in this experiment.

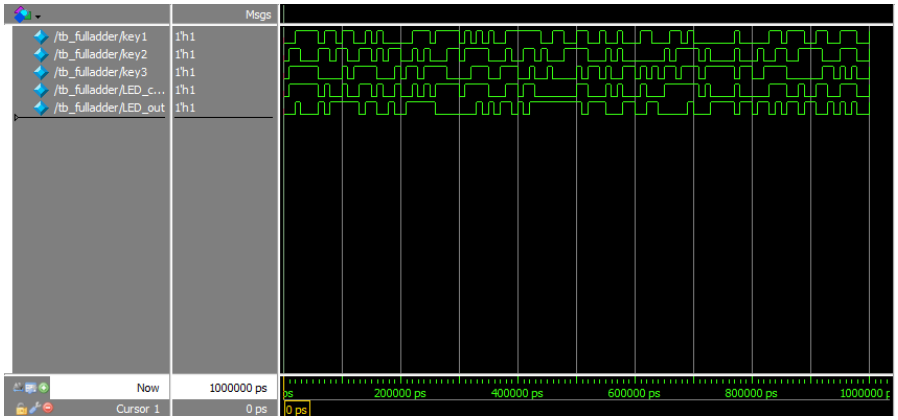
III Results

III: i Testbench

In this experiment, the testbench was built to accept three one-bit inputs. The testing inputs were randomly generated every 10 ns, and the outputs were presented by waveform. The expected results are listed below (carry, out):

1. All inputs are 0s. Output 0,0
2. One input is 1, and two are 0s. Output 0,1
3. Two inputs are 1, and one is 0. Output 1,0
4. ALL inputs are 1s. Output 1,1

III: ii Signal Waveforms



It turned out that the actual output value was the same as we predicted.

III: iii Development Board

On the development board, test results are the same as we predicted. Again, the video can be found at the Github release Page or Youtube.

IV Discussions and Conclusion

IV: i Discussions

We encountered the problem of the testbench simulation not giving out the waveform plot in this experiment. It turns out that it is the problem with the testbench codes, in which the initiation part has the wrong name.

IV: ii Conclusion

In this experiment, we learned how to use Quartus to create an FPGA project, write a Verilog code, and develop a testbench for the Verilog design.

V Appendices

V: i Reference

The code in this experiment referenced the following sources (IEEE Style):

Referências

[1] pikipity, Aug, 2024, "FPGA-Laboratory/../4_Half_Adder," distributed on Github, https://github.com/pikipity/FPGA-Laboratory/blob/main/Lab1/4_Half_Adder/RTL/half_adder.v

[2] pikipity, Aug, 2024, "FPGA-Laboratory/../4_Half_Adder," distributed on Github, https://github.com/pikipity/FPGA-Laboratory/blob/main/Lab1/4_Half_Adder/Sim/tb_half_adder.v

Capítulo III

2-4 Line Decoder

I Introduction

I: i Background

A decoder can take n-th input lines and then decode them into 2^n different output lines. One application of such a component is to drive a display array with relatively narrow input lines.

I: ii Purpose

In this experiment, we want to design a 2-4 line decoder with an enabler that can handle a two-bit input and decode them to light up 4 different LEDs.

I: iii Design and Key-results

The line decoder has one 2-bit input

II Materials and Methods

II: i Code Structure

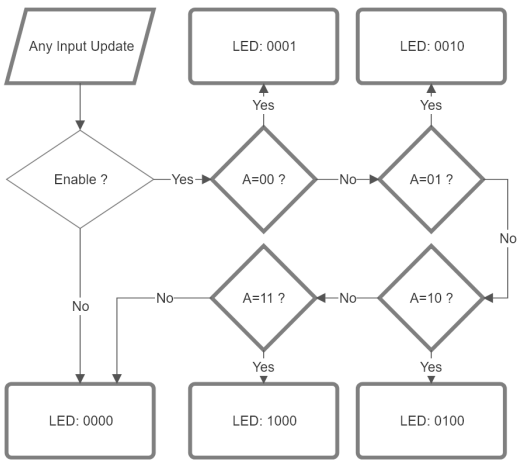


Figura 1: Block Diagram of the decoder

II: ii Truth Table

A	B_3	B_2	B_1	B_0
00	1	1	1	0
01	1	1	0	1
10	1	0	1	1
11	0	1	1	1

Tabela 1: Truth table of decoder

II: iii Signal Design

The decoder has two inputs a[0:1] and enable[0], and one output b[0:3]. For all of the signals, 1(high voltage) stands for "off"due to the design of circuit on the develop board.

- a[0:1] is the input of the decoder, corresponding to Key_2 and Key_3 on the board.
- enable[0] is the enable signal of the decoder, corresponding to Key_1 on the board.
- b[0:3] is connected to the LEDs on the board, with LED_3 being the most significant bit.

II: iv Resources in Board

We used LED_0 to LED_3 and Key_1 to Key_3 on the development board in this experiment.

III Results

III: i Testbench Design

The testbench is designed to have each bit of input *a* assigned randomly between 0 and 1 every 10ns. And the input *enable* is assigned randomly every 50ns.

III: ii Simulation Result

The simulation waveform report is shown in the following figure.

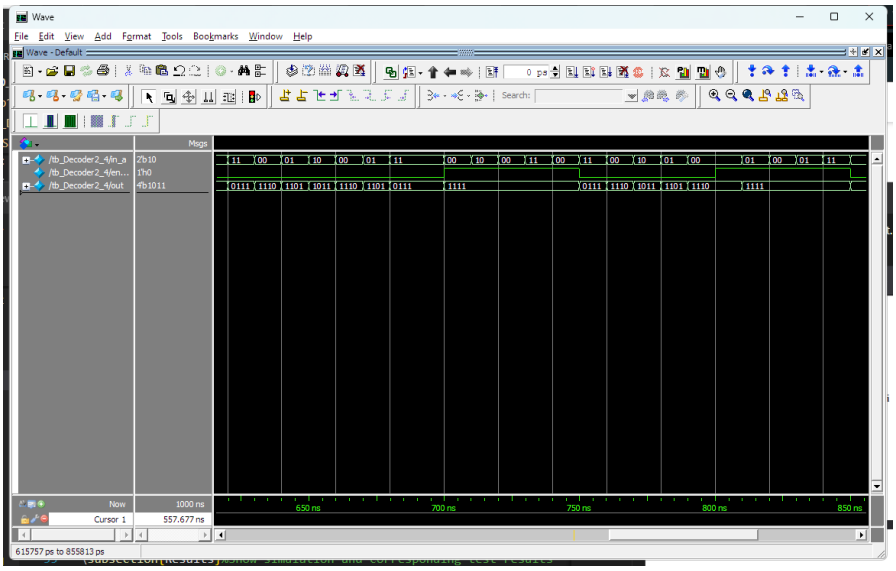


Figura 2: Waveform report of the decoder

As one may see from the waveform, when the enable signal is high, the decoder output 1111 whatever the input is. And when the enable signal is low, the output of the decoder is aligned

with the expected output according to the truth table.

III: iii Board Result

The board test shows the decoder works as expected as shown in the demonstration video. Again, the video can be found at the Github release Page or Youtube.

IV Discussions and Conclusion

IV: i Discussions

During the experiment, I have trouble connecting the word most significant bit with the index of i/o signals. And I find it helpful to align the variable with their physical location when designing the truth table.

IV: ii Conclusion

We successfully designed a 2-4 line decoder with an enabler in this experiment, and the decoder works as expected in the simulation as well as on the board.

V Appendices

The code in this experiment referenced the following sources (IEEE Style):

Referências

[1] pikipity, Aug, 2024, "FPGA-Laboratory/../3_Decoder," distributed on Github, https://github.com/pikipity/FPGA-Laboratory/blob/main/Lab1/3_Decoder/RTL/decoder3_8.v

Capítulo IV

One-bit Digital Comparator

I Introduction

I: i Background

The one-bit comparator can perform a logic comparison to judge the magnitude of two single-bit inputs. It outputs three different states to show the result which indicates greater, smaller, and equal.

I: ii Purpose

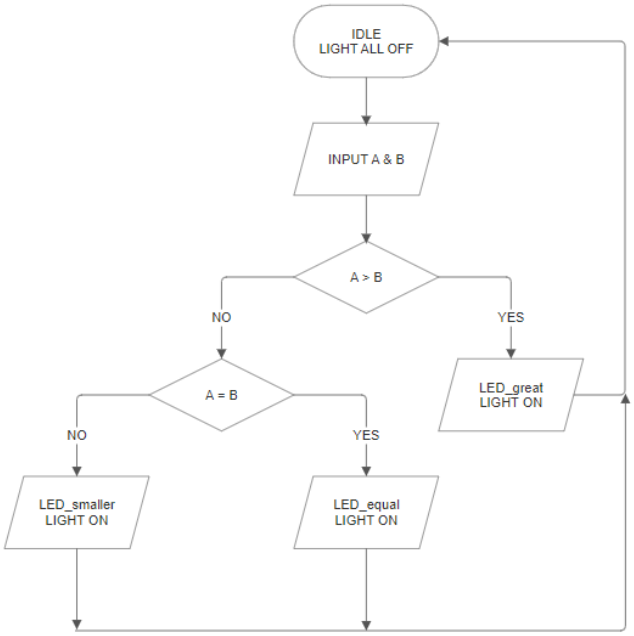
In this experiment, we aimed to design a circuit that is able to compare two one-bit inputs and give three different states of output, which shows the numerical relationship between two inputs.

I: iii Design and Key-results

The circuit of the comparator is implemented using Verilog on the FPGA board in this experiment. This circuit requires two single-bit inputs and gives out three single-bit outputs, they are A, B, LED_greater, LED_equal, and LED_smaller respectively. The outputs are displayed on the development board using LED lights.

II Materials and Methods

II: i Structure Block Diagram



II: ii Truth Table

A	B	LED_greater	LED_equal	LED_smaller
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

II: iii Resources in Board

We used LED0 to LED2 and KEY0 to KEY1 on the development board in this experiment.

III Results

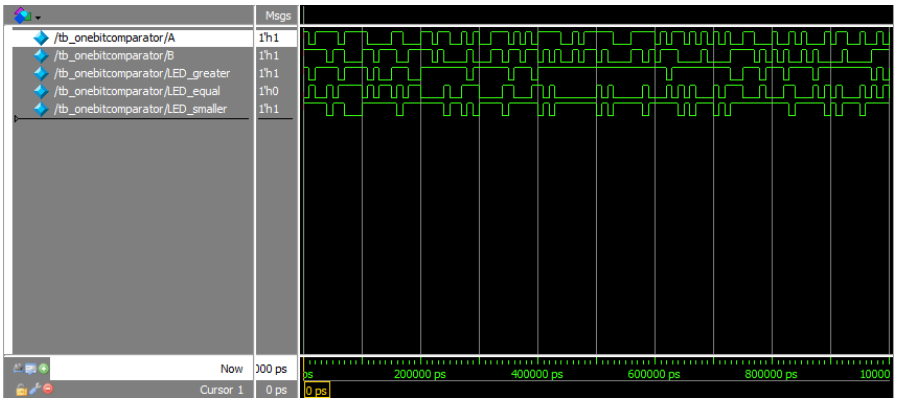
III: i Testbench

In this experiment, the testbench was built to accept two one-bit inputs. The testing inputs were randomly assigned every 10 ns, and the outputs having three different possibilities were presented

by waveform. The expected outputs are shown below:

1. Input A is 0, B is 0. Only LED_equal is 1
2. Input A is 0, B is 1. Only LED_smaller is 1
3. Input A is 1, B is 0. Only LED_greater is 1
4. Input A is 1, B is 1. Only LED_equal is 1

III: ii Simulation Waveforms



It turned out that the actual output value was the same as we predicted.

III: iii Development Board

On the development board, test results are the same as we predicted. Again, the video can be found at the Github release Page or Youtube.

IV Discussions and Conclusion

IV: i Discussions

During this experiment, we encountered a problem of differences in the recognition of signal inputs. When the button is pressed, we expect it is inputting signal 1 into the system. However, instead of entering signal 1, the chip recognizes the button pressing behaviour as sending signal 0. Eventually, the misinterpretation resulted in the opposite output. To fix this output error, we decided to reverse the logic judgment that was used to compare the input signals. Now, when the chip accepts the input A<B, which we originally predicted as A>B, it correctly produces the expected outcome of A>B.

IV: ii Conclusion

In this experiment, we met some unexpected errors. Eventually, we overcame them by discussion. We successfully designed a circuit that can compare two single-bit inputs and give a correct magnitude relationship outcome.

V Appendices

References and Appendices

V: i Reference

The code in this experiment referenced the following sources (IEEE Style):

Referências

[1] pikipity, Aug, 2024, "FPGA-Laboratory/../4_Half_Adder," distributed on Github, https://github.com/pikipity/FPGA-Laboratory/blob/main/Lab1/4_Half_Adder/Sim/tb_half_adder.v