Decoders and Encoders

Lab 7 245

11/7/22

Chris Tark

Objective: Learn to code a 3 by 8 decoder, an 8 by 3 encoder, and an 8 by 3 priority encoder in Verilog. Understand the capabilities and arguments between all three design specifications.

Design/Experiment:

Boolean algebra is data that computers use to function for the manufacturer. Our present day religiously relies on computers for everything imaginable. A bidet to stabilizing radioisotope thermoelectric generators, boolean algebra makes the world go round. Encoders and decoders are ways to control combinational logic. A decoder retrieves a binary numerical input and outputs a binary sequence of 1s and 0s. The initial data is 3-bit binary number and return a 8-bit binary integer.

Decoder Verilog Design:

The design has two inputs, the enabler and A with an array of length 3. One output of array length 8. In a loop, if enabler is on, the case will check each possible input of A. If an A is detected while the enabler is on then depending on the input, one output will display.

module 3-8decoder (input (E, A [2:0]), output Y [7:0]);

//loop for enabler, input, and output

always @(E,A,Y)

//check E

if(E)

case(A)

//check A for valid case

0: Y = b’800000001;

1: Y = b’800000010;

2: Y = b’800000100;

3: Y = b’800001000;

4: Y = b’800010000;

5: Y = b’800100000;

6: Y = b’801000000;

7: Y = b’810000000;

endcase

else

//else E==0

Y: b’800000000;

Endmodule

Decoder Functional Analysis:

The functional simulation verifies the written code in Verilog with testing user input values. A new file called Verilog Wave Form simulates the coded chip. This chip is imported into the VWF file to adjust desired inputs. A is clemently set for the whole array of A[2:0], values are form 0 to 7 in binary.

Functional simulation for the decoder:

Graphical user interface, application, table, Excel

Description automatically generated

Augment Decoder:

To display our data on the FPGA board we need to set the switches as input and lights as output. The port list is initialized as follows: input ([3:0]SW, [3:0] LEDR), output( [7:0] LEDG). The final out was “reg [7:0] Y;” The enabler was initialized as a wire along with A[2:0]. Now, assign the e to the last switch and A to the other switches. The lights are the only indication for a working word, assign the red to the switches and green to the output.

An encoder takes in a coded data strip and outputs a number in user’s format. In our case, this encoder takes in an 8-bit binary number and converts to a 3-bit binary number. Y (8-bit) and E (enabler) are inputs while the output is A (3-bit). In a loop, if e is true, and there is a valid input for a case then the corresponding output will follow.

Verilog Encoder:

module 8to3encoder(Y[7:0], A[2:0], E);

input Y, E;

output reg A,

//loop for enabler, input, and output

always @(E, Y, A)

//check enabler

if(E)

//valid cases

case(Y)

b’800000001: A = 0;

b’800000010: A = 1;

b’800000100: A = 2;

b’800001000: A = 3;

b’800010000: A = 4;

b’800100000: A = 5;

b’801000000: A = 6;

b’810000000: A = 7;

//multiple inputs return 0

default: A = 0;

endcase

endmodule

Encoder Functional Simulation with one input one at a time:

Graphical user interface, application, table

Description automatically generated

Encoder Functional Simulation with multiple inputs on:

Graphical user interface, table, Excel

Description automatically generated with medium confidence

In the figure above if the encoder has multiple inputs at once the results is 0 or in the binary final form 000. This is because multiple inputs is not a valid case (Y) and I implemented this by placing a default.

Priority Encoder:

The priority encoder has the same functionality of an encoder with the input gripe. Once the input has a 1 following number are ignored completely hence the name “priority” encoder.

module(A[7:0], Y[2:0], E);

//enabler and input

input A, E;

//final output

output reg Y;

//loop

always @(A,E,Y)

//if E is true

if(E)

casex(A)

8'b 1xxxxxxx: I = 7;

8'b 01xxxxxx: I = 6;

8'b 001xxxxx: I = 5;

8'b 0001xxxx: I = 4;

8'b 00001xxx: I = 3;

8'b 000001xx: I = 2;

8'b 0000001x: I = 1;

8'b 00000001: I = 0;

endcase

endmodule

Augment Priority Encoder:

Switches are inputs and red light (A), output is assigned to green light. E (wire) is assigned to the last switch. All are through a loop, check for enabler, and finally check for a valid A. Remember to keep in mind the priority aspect of the FPGA.

Hardware: n/a

Analysis:

The encoder can only take in one input at a time for a corresponding result. Multiple inputs call to action our default case. The priority encoder does not have a default case, instead rely on the input only. Once a valid input is entered all the digits after 1 are neglected. In this case there can be multiple inputs if, and only after the intended values is returned.

A new aspect I learned in Verilog was using x as a “don’t care” value. A decoder is used multiple times as an address decoder in CPU memory location. Also, encoders allow visual feedback for high-precision motion while under extreme conditions.

<https://www.ecstuff4u.com/2020/03/what-are-applications-of-decoders.html>

<https://www.encoder.com/encoder-applications>

Conclusion:

I designed and constructed a 3-8 decoder, 8-3 encoder, and 8-3 priority encoder all in Verilog. Testing all possible output in the Waveform editor for precise measurements to finally program a FPGA board for a complete analysis of decoders and encoders.