`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 15:24:07 10/17/2016

// Design Name:

// Module Name: four\_bit\_mux\_behavioral

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module four\_bit\_mux(Y, A, B, S);

/\*output and input ports\*/

output reg [3:0]Y;

input wire [3:0]A;

input wire [3:0]B;

input wire S;

always@(A or B or S)

begin

if(S == 1'b0)

Y = A;

else

Y = B;

end

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 15:55:07 10/17/2016

// Design Name:

// Module Name: four\_two\_encoder

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module four\_two\_encoder(

input wire [3:0] W,

output wire zero,

output reg [1:0] Y

);

assign zero = (W == 4'b0000);

always@(W)

begin

case(W)

4'b0001: Y = 2'b00;

4'b0010: Y = 2'b01;

4'b0100: Y = 2'b10;

4'b1000: Y = 2'b11;

default: Y = 2'bXX;

endcase

end

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 15:35:42 10/17/2016

// Design Name:

// Module Name: mux\_4bit\_4tol

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module mux\_4bit\_4to1(Y, A, B, C, D, S);

output reg [3:0] Y;

input wire [3:0] A,B,C,D;

input wire [1:0] S;

always@(\*)

case(S)

2'b00: Y = A;

2'b01: Y = B;

2'b10: Y = C;

2'b11: Y = D;

endcase

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 16:02:53 10/17/2016

// Design Name:

// Module Name: priority\_encoder

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module priority\_encoder(input wire [3:0] W,

output wire zero,

output reg [1:0] Y

);

assign zero = (W == 4'b0000);

always@(W)

begin

casex(W)

4'b0001: Y = 2'b00;

4'b001X: Y = 2'b01;

4'b01XX: Y = 2'b10;

4'b1XXX: Y = 2'b11;

default: Y = 2'bXX;

endcase

end

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 15:47:05 10/17/2016

// Design Name:

// Module Name: two\_four\_decoder

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module two\_four\_decoder(

input wire [1:0] W,

input wire En,

output reg [3:0] Y

);

always@ (En or W)

begin

if(En == 1'b1)

case(W)

2'b00: Y = 4'b0001;

2'b01: Y = 4'b0010;

2'b10: Y = 4'b0100;

2'b11: Y = 4'b1000;

endcase

else

Y = 4'b0000;

end

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 14:51:14 10/17/2016

// Design Name:

// Module Name: two\_one\_mux\_behavioral

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module two\_one\_mux(Y, A, B, S);

/\*output and input ports\*/

output reg Y;

input wire A;

input wire B;

input wire S;

always@(A or B or S)

begin

if(S == 1'b0)

Y = A;

else

Y = B;

end

endmodule

PRIORITY

# User Push-Bottons

NET "W[0]" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN; #North

NET "W[1]" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN; #East

NET "W[2]" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN; #South

NET "W[3]" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN; #West

# LEDs

NET "Y[0]" LOC = "F12"; #LED0

NET "Y[1]" LOC = "E12"; #LED1

NET "zero" LOC = "E11"; #LED2

2-4 DECORDER

#This User Constraint File creates a mapping of

#the input and output ports from your top level

#module to the actual pins on the FPGA

#NET ---> specifies the net name within our design

#LOC ---> specifies the pin location on the FPGA

# User Switches

NET "W[0]" LOC = "L13"; #SW0

NET "W[1]" LOC = "L14"; #SW1

NET "En" LOC = "H18"; #SW2

# LEDs

NET "Y[0]" LOC = "F12"; #LD0

NET "Y[1]" LOC = "E12"; #LD1

NET "Y[2]" LOC = "E11"; #LD2

NET "Y[3]" LOC = "F11"; #LD3

4-2 ENCODER

# User Push-Bottons

NET "W[0]" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN; #North

NET "W[1]" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN; #East

NET "W[2]" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN; #South

NET "W[3]" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN; #West

# LEDs

NET "Y[0]" LOC = "F12"; #LED0

NET "Y[1]" LOC = "E12"; #LED1

NET "zero" LOC = "E11"; #LED2