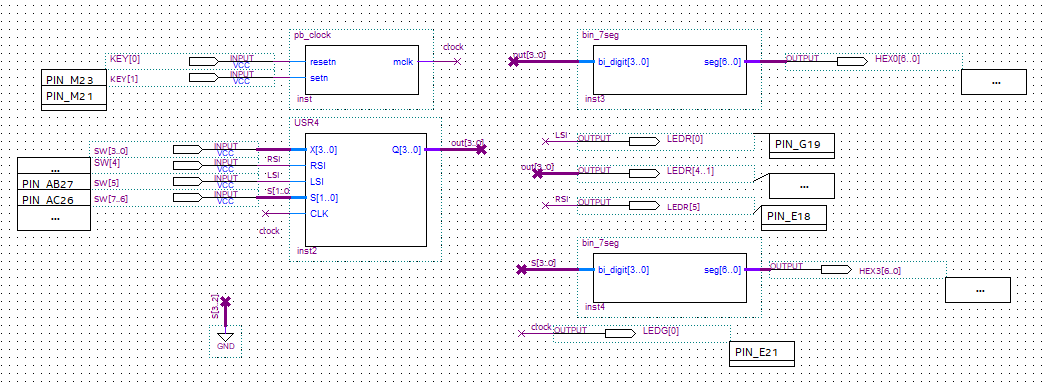
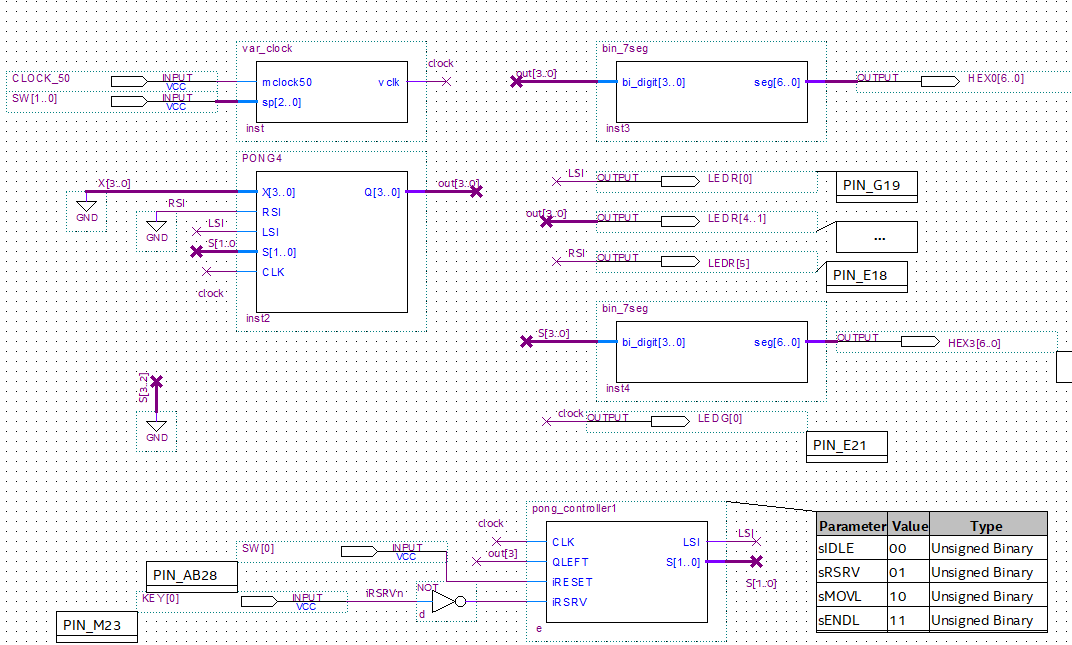
Jay Alag, Tim Lu, Jason Fong

**Lab 8 Report:**

**1. Include schematics, test plan, Verilog, and observed results.**

USR4PONG4

module USR4(X,RSI,LSI,S,CLK,Q);

input[3:0]X;

input[1:0]S;

input RSI,LSI,CLK;

output reg [3:0]Q;

always @(posedge CLK)

case(S)

0: begin Q[3]<= Q[3];

Q[2]<= Q[2];

Q[1]<= Q[1];

Q[0]<= Q[0]; end

1: begin Q[3]<= RSI;

Q[2]<= Q[3];

Q[1]<= Q[2];

Q[0]<= Q[1]; end

2: begin Q[3]<= Q[2];

Q[2]<= Q[1];

Q[1]<= Q[0];

Q[0]<= LSI; end

3: begin Q[3]<= X[3];

Q[2]<= X[2];

Q[1]<= X[1];

Q[0]<= X[0]; end

endcase

endmodule

module PONG4(X,RSI,LSI,S,CLK,Q);

input[3:0]X;

input[1:0]S;

input RSI,LSI,CLK;

output reg [3:0]Q;

always @(posedge CLK)

case(S)

0: begin Q[3]<= Q[3];

Q[2]<= Q[2];

Q[1]<= Q[1];

Q[0]<= Q[0]; end

1: begin Q[3]<= RSI;

Q[2]<= Q[3];

Q[1]<= Q[2];

Q[0]<= Q[1]; end

2: begin Q[3]<= Q[2];

Q[2]<= Q[1];

Q[1]<= Q[0];

Q[0]<= LSI; end

3: begin Q[3]<= 0;

Q[2]<= 0;

Q[1]<= 0;

Q[0]<= 0; end

endcase

endmodule

module pong\_controller1 (input CLK, QLEFT, iRESET, iRSRV,

output reg LSI,

output reg [1:0] S);

reg [1:0] currState, nextState;

parameter sIDLE = 2'b00, sRSRV = 2'b01, sMOVL = 2'b10, sENDL = 2'b11;

always @ (posedge CLK) currState <= nextState;

always @ (\*) begin

nextState = sIDLE;

case(currState)

sIDLE: begin

if(iRSRV) nextState = sRSRV;

{LSI, S} = 3'b011;

end

sRSRV: begin

if(!iRESET) nextState = sMOVL;

{LSI, S} = 3'b110;

end

sMOVL: begin

if(iRESET) nextState = sIDLE;

else if(QLEFT) nextState = sENDL;

else nextState = sMOVL;

{LSI, S} = 3'b010;

end

sENDL: begin

nextState = sIDLE;

{LSI, S} = 3'b000;

end

default: begin

nextState = sIDLE;

{LSI, S} = 3'b000;

end

endcase

end

endmodule

**2. Describe the most challenging part of this lab.**

One hurdle we faced was getting our FPGA board to display the board correctly, but the most difficult part was understanding how it would be displayed given certain inputs.

**3. For lab part 1, if you loaded the register with 1101, what value would that represent if you interpreted it as an unsigned integer? With RSI=0, if you shifted to the right what binary pattern would you see and how would it be interpreted as an unsigned number? With RSI=0, if you shifted to the right a second time what binary pattern would you see and how would it be interpreted as an unsigned number? Explain how right shifting is related to dividing by 2.**

An unsigned integer 1101 has the decimal value of 13. With RSI = 0, shifting it to the right gives the binary value 0110, with decimal value 6. Shifting right again gives the value 0011, which translates to 3. Essentially, shifting right divides the value by two, though whether it is even or odd depends on if the second bit is 1 or 0.

**4. In the previous question, for unsigned interpretation of the values, RSI was set to zero. To divide by 2 using a right shift for signed integer interpretation using 2’s complement representation, what should RSI be? Why?**

For a signed integer, RSI should be set to 1 if it is negative and 0 if it is positive in order to preserve the sign and calculate the negative value correctly.

**5. How would two 4-bit universal shift registers be connected to form an 8-bit universal shift register? Show a schematic in which each 4-bit register is shown as a block component with inputs and outputs. Do not show the internal components and connections of the 4-bit universal shift register.**

The LSB of the first shift register should be used as the input for the RSI of the second and the MSB of the second shift register being used as the LSI of the first. Otherwise, both registers should use their standard inputs, with the first bit taking 7..4 inputs and the second using 3..0 inputs, the first register taking the external RSI of the 8-bit shift register and the second taking the external LSI.

**6. For lab part 2, if the controller is in the idle state and the right serve input is high during an active clock edge, what is the next state? For the next 6 active clock edges, specify the state that the controller is in after each clock edge.**

sRSRV, sMOVL, sMOVL, sMOVL, sMOVL, sIDLE

**7. If by mistake QLEFT were connected to Q[0] instead of Q[3], how would your answer to the previous question change?**

sRSRV, sMOVL, sENDL, sIDLE, sRSRV, sMOVL

**8. Show how you would modify your state transition diagram to allow both players to serve. How many additional states would you need? What additional inputs and outputs would be needed?**

To allow both players to serve, we would need 2 more states to act as the counterparts of sRSRV and sMOVL. Two additional inputs are required, one to determine whether the left or right side is serving and the other connected to the RSI of the register. For outputs, we would just need to assign RSI and S[1..0] similarly to how LSI and S[1..0] were assigned for a right server.