Lab 14 Submission

**FSM-Based 5-Bit Multiplier**

CPE 133 - 03

Michael Hegglin

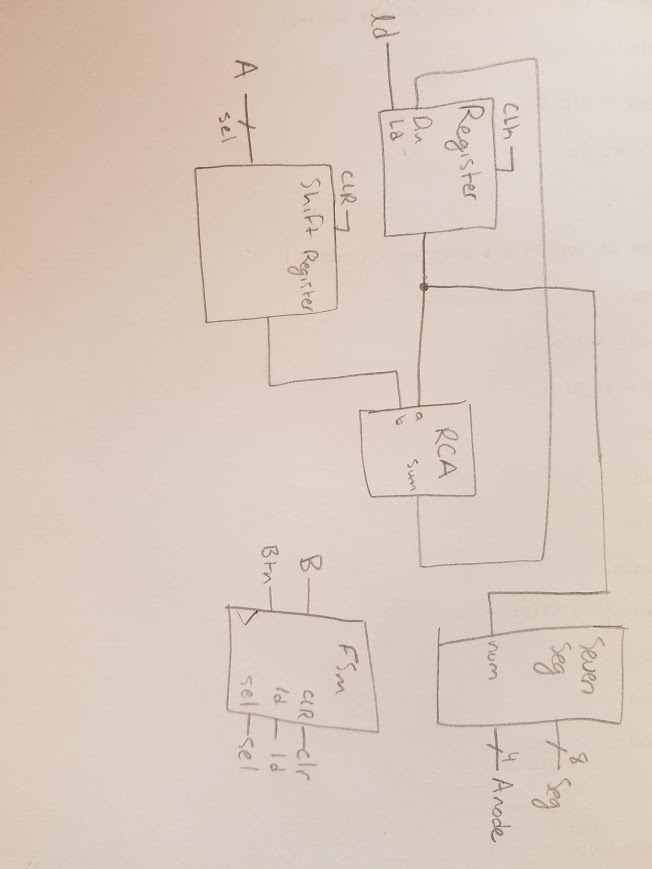
Jonathan Skelly

**Executive Summary:**

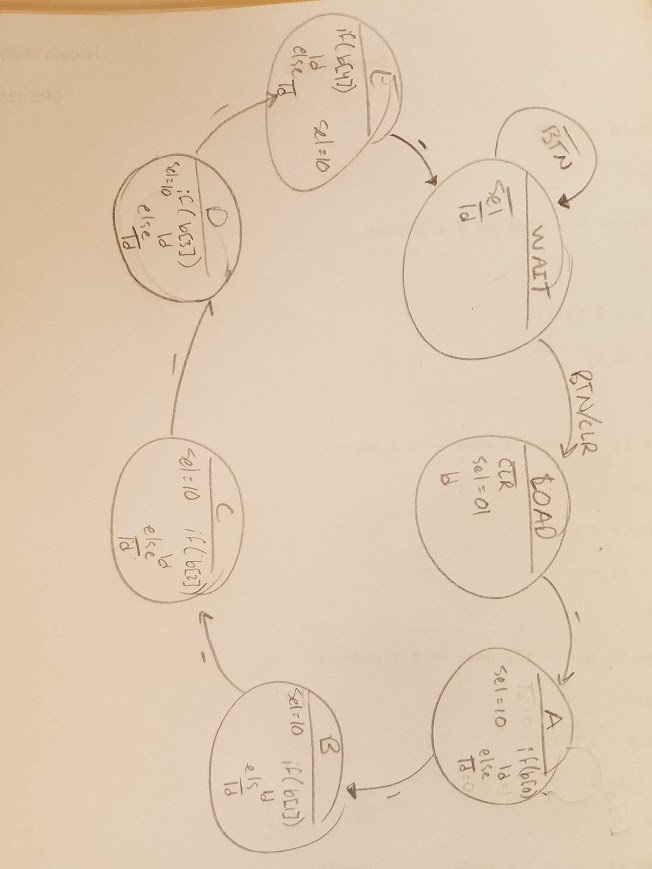
We designed an 5-bit number multiplier with an unsigned binary number. The multiplier used an FSM, and shift registers to multiply two 5-bit numbers together. The logic decided which way the number(bit-position) would be shifted to multiply.



**High Level BBD**

****

**Low Level BBD**



**FSM Diagram**

**Questions:**

1. How would you approach designing a circuit if one of the design constraints was to make the circuit invisible?

* I would build all the logic in a programming language so that there was physical circuit; thus it would be “invisible”.

2. What are the four operations you can perform on a single bit?

* The four operations you can perform on a single bit are: toggle, set, clear, and hold.

3. Pretend this is an EE course and write a closed form formula that relates the maximum value a result can be from multiplying two n-bit numbers. Show your calculations for this problem.

* The maximum value that can result from 2 n-bit numbers is 2n-bit output.

4. Attack of the SAT questions: State diagrams are to FSMs such as “X’s” are to circuits. For this question, what does “X” refer to?

* The “X” refers to a circuit diagram.

5. Another attack of the SAT questions: State diagrams are to FSMs such as “X’s” are to computer programming code. For this question, what does “X” refer to?

* The “X” refers to a UML diagram.

6. One way to perform multiplication is to add continually. Which approach to multiplication would be faster: the continual adding approach or the approach you used in this lab activity? Answer for the best case, worst case, and average case. Briefly but complete describe and support your answer.

* The approach we used in the lab was better due to the fact that when the number of bits increases, the number of additions increases. This means multiplying each bit of the first number by each bit of the second number is faster. It is faster in all cases.

7. Based on the answer to the previous problem, approximately how much slower would it be to do a multiplication using the continuous addition method as opposed to the approach you used in this lab activity? For this problem, assume you’re multiplying two 8-bit values.

* 2^n - n
* 2^n is the addition method
* N is the multiplication method

8. Based on you FSM implementation in this lab activity and a 100MHz system clock, how long will it take to generate the results of your 4-bit multiply operation after the button is pressed? Support your answer with such things as actual calculations.

* To make the calculations it takes 7 clock cycles. This means it will take 7\*10^-6s.

9. So, I ask my higher-level computer programming language to execute an unsigned integer multiplication algorithm (meaning, I use the “\*” operator in an expression), and it does it for me. Is there a way to know what algorithm the computer uses to perform that operation? Briefly but completely explain.

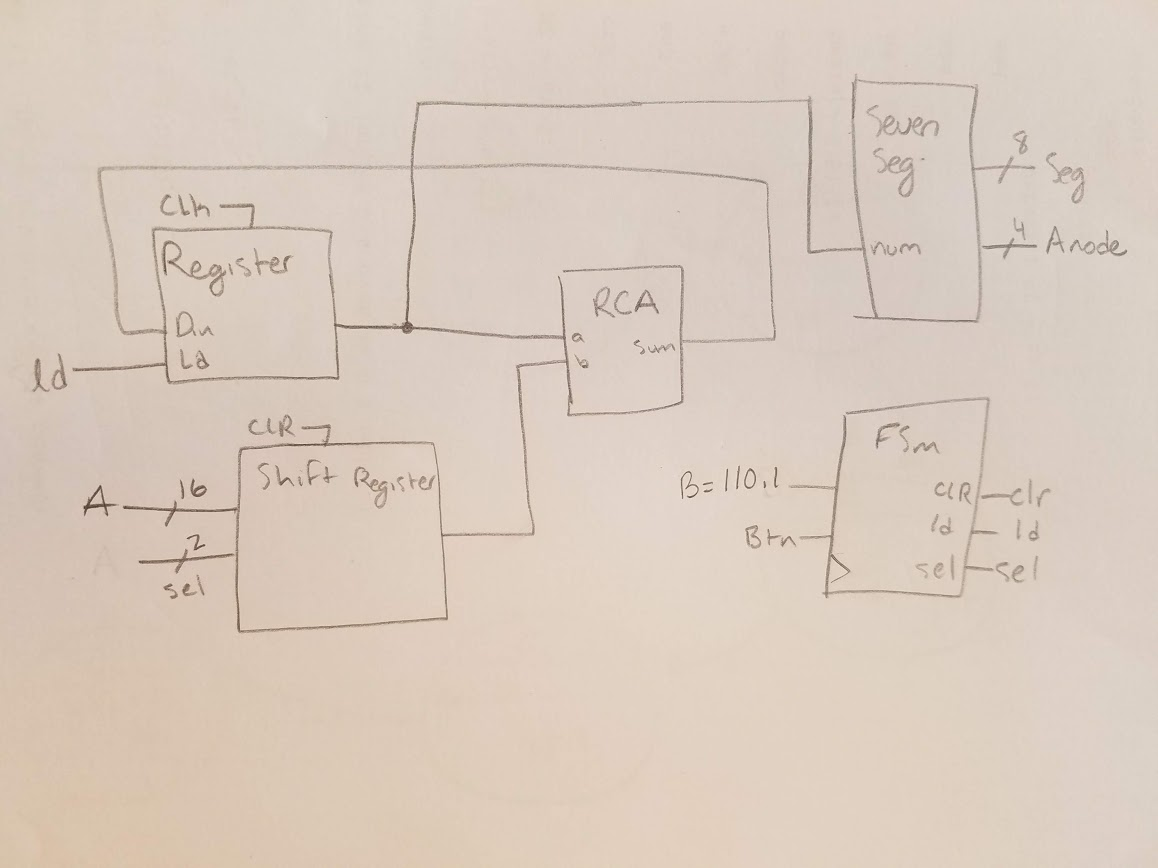
* You could step through the code bit by bit and see what is happening.

10. List the four types of information provided by a state diagram.

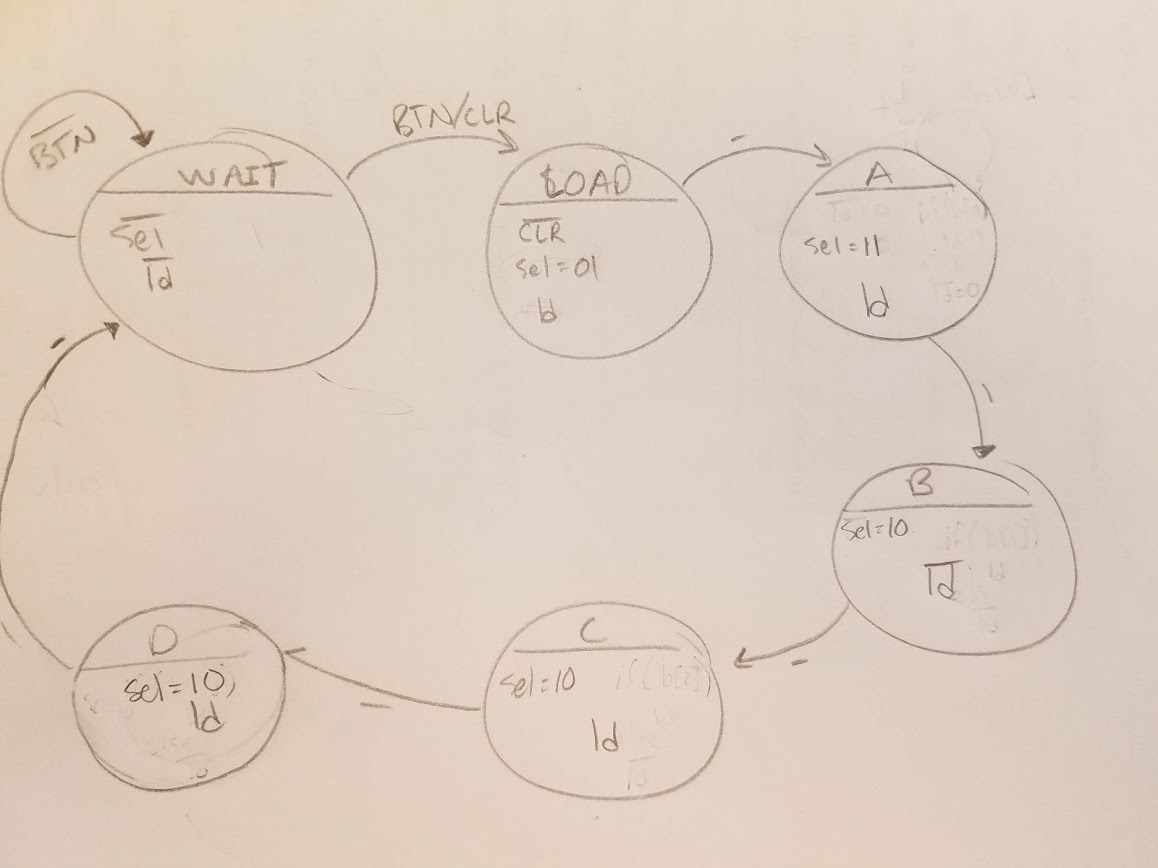
* The four types of information are: outputs within the state (moore), variables that make an operation in the circuit happen(changing states, mealy), starting conditions for a circuit, and ending conditions for a circuit.

**Design Problems:**

1. Using only one RCA, one register, and one shift register, design a circuit that could effectively multiply a given 16-bit unsigned number by 6.5. Also, provide a state diagram describing an FSM that would drive your circuit. Consider the multiply operation to start when a FSM detects a button press; after that, the algorithm continues independently of button presses until the algorithm completes. When the algorithm completes, the FSM verifies the button is released before waiting for another button press. For this problem, consider the output persistent after the algorithm completes. Minimize your use of hardware in this design. State how the circuit is controlled.

****

**Black Body Diagram**



**FSM Diagram**

**Source Code**

`timescale 1ns / 1ps

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

// Company: Ratner Surf Designs

// Engineer: James Ratner

//

// Create Date: 12/07/2018 08:05:03 AM

// Design Name: FSM module

// Module Name: fsm\_template

// Project Name: 5-bit multiplier

// Target Devices:

// Tool Versions:

// Description: Generic FSM model with both Mealy & Moore outputs.

// Note: data widths of state variables are not specified

//

// Dependencies:

//

// Revision:

// Revision 1.00 - File Created (07-07-2018)

// Additional Comments:

//

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

module fsm\_template(reset\_n, sel, ld, clk, b, btn, LED, clr);

input reset\_n, clk, btn;

input [4:0] b;

output reg ld, LED, clr;

output reg [1:0] sel;

//- next state & present state variables

reg [2:0] NS, PS;

//- bit-level state representations

parameter [2:0] st\_WAIT=3'b000, st\_0=3'b001, st\_1=3'b010, st\_2=3'b011, st\_3=3'b100, st\_4=3'b101, st\_LOAD=3'b111;

//- model the state registers

always @ (negedge reset\_n, posedge clk)

if (reset\_n == 0)

PS <= st\_WAIT;

else

PS <= NS;

//- model the next-state and output decoders

always @ (\*)

begin

case(PS)

st\_WAIT:

begin

sel = 00;

LED = 0;

ld = 0;

if(btn == 1)

begin

clr = 1;

NS = st\_LOAD;

end

else

begin

NS = st\_WAIT;

end

end

st\_LOAD:

begin

clr = 0;

sel = 01;

ld = 1;

NS = st\_0;

end

st\_0:

begin

ld = 0;

clr = 0;

sel = 10;

if(b[0] == 1)

begin

ld = 1;

end

else

begin

ld = 0;

end

NS = st\_1;

end

st\_1:

begin

ld = 0;

sel = 10;

if(b[1] == 1)

begin

ld = 1;

end

else

begin

ld = 0;

end

NS = st\_2;

end

st\_2:

begin

ld = 0;

sel = 10;

if(b[2] == 1)

begin

ld = 1;

end

else

begin

ld = 0;

end

NS = st\_3;

end

st\_3:

begin

ld = 0;

sel = 10;

if(b[3] == 1)

begin

ld = 1;

end

else

begin

ld = 0;

end

NS = st\_4;

end

st\_4:

begin

ld = 0;

LED = 1;

sel = 10;

if(b[4] == 1)

begin

ld = 1;

end

else

begin

ld = 0;

end

NS = st\_WAIT;

end

default: NS = st\_WAIT;

endcase

end

endmodule

`timescale 1ns / 1ps

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

// Company: DogsWithJobs

// Engineer: Hegglin Skelly

//

// Create Date: 12/01/2018 08:05:03 AM

// Design Name: Multiplier

// Module Name: Main

// Project Name: FSM-Based 5-Bit Multiplier

// Target Devices:

// Tool Versions:

// Description:Circuit takes 2 5-bit numbers and multiplies them.

//

// Dependencies:

//

// Revision:

// Revision 1.00 - File Created (07-07-2018)

// Additional Comments:

//

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

module Main(a, b, seg, an, clk, btn, LED);

input [4:0] a, b;

input clk, btn;

output [7:0] seg;

output [3:0] an;

output LED;

wire [1:0] sel;

wire nclk, clr, ld;

wire [9:0] shift, sum, reg\_out;

clk\_divder\_nbit #(.n(24)) MY\_DIV (

.clockin (clk),

.clockout (nclk)

);

usr\_nb #(.n(10)) MY\_USR (

.data\_in (a),

.dbit (0),

.sel (sel),

.clk (nclk),

.clr (clr),

.data\_out (shift)

);

reg\_nb #(10) MY\_REG (

.data\_in (sum),

.ld (ld),

.clk (nclk),

.clr (clr),

.data\_out (reg\_out)

);

rca\_nb #(.n(10)) MY\_RCA (

.a (reg\_out),

.b (shift),

.cin (0),

.sum (sum),

.co (0)

);

fsm\_template my\_fsm(

.reset\_n (1),

.ld (ld),

.sel (sel),

.clk (nclk),

.b (b),

.btn (btn),

.LED (LED),

.clr (clr)

);

univ\_sseg my\_univ\_sseg (

.cnt1 (reg\_out),

.cnt2 (0),

.valid (1),

.dp\_en (0),

.dp\_sel (0),

.mod\_sel (10),

.sign (0),

.clk (clk),

.ssegs (seg),

.disp\_en (an) );

endmodule