I completed this assignment entirely on my own, except for discussions with Jiamin Xu

Ethan Wong
11/12/20

Assignment 7

2 4-bit inputs
1 start input → normally 0, initiates multiplication with pulse to 1
1 clock input
1 8-bit output

Demonstration of Binary Multiplication (with variables)

2bit:

		W1	W0	
	x	X1	X0	
	0	W1X1	W0X0	
+	W1X1	W0X1	0	
P3	P2	P1	P0	

P0 = W0X0
P1 = W1X0 + W0X1
P2 = W1X1 + C_o from P1
P3 = C_o from P_2

4 bit

					W3	W2	W1	W0	
				x	X3	X2	X1	X0	
	0	0	0	0	W3X0	W2X0	W1X0	W0X0	1
	0	0	0	W3X1	W2X1	W1X1	W0X1	0	2
	0	0	W3X2	W2X2	W1X2	W0X2	0	0	3
+	extra carry	W3X3	W2X3	W1X3	W0X3	0	0	0	4
	P7	P6	P5	P4	P3	P2	P1	P0	

Method of Multiplication

I have shown above two examples of binary multiplication. The logic for the 4-bit multiplication is very similar to that of the 2-bit multiplication. The basic multiplication of each bit is performed with an AND gate because the truth table for AND matches the truth table for multiplication. The addition will be perfomed by a full-adder.

Explanation of Overall Circuit Logic

The circuit does not do anything until a user presses the start button. ~~While the start button isn't clicked, the circuit will clear itself to ensure its registers are empty~~. Once the start button is pressed, the circuit will clear itself to make sure it starts fresh and a clock pulse is sent to the 4-bit Parallel load Right Shift Registers. W (W3 W2 W1 W0) gets placed in the top register and X (X3 X2 X1 X0) gets placed in the bottom register. After this is done, the circuit will be in S1.

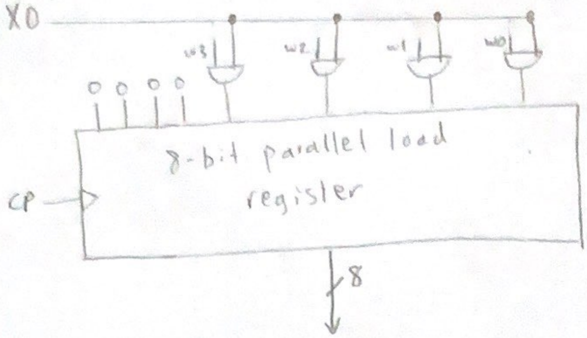
The input no longer matters for the rest of the states. It only mattered for Init. When changing from S1 to S2, the output includes "m", which will tell the circuit to shift the rightmost bit from X and every bit of W into the Multipliers. Since our select inputs S0 and S1 are both zero in this stage, the vector MUX receives the T0 input from Multiplier X0. The clock pulse allows the output to be added with whatever was previously stored in the 8-bit parallel-load register that is connected to the adder (00000000). The second 8-bit parallel register holds the value as well. This will be useful when it is time to output. After this is done, the circuit will be in S2.

When changing from S2 to S3, the process is almost identical. The only difference is that the S1 select input is now 1, so the vector MUX selects the T1 input from Multiplier X1. When changing from S3 to S4, the S0 select input is 1, so the vector Mux selects the T2 input from Multiplier X2.

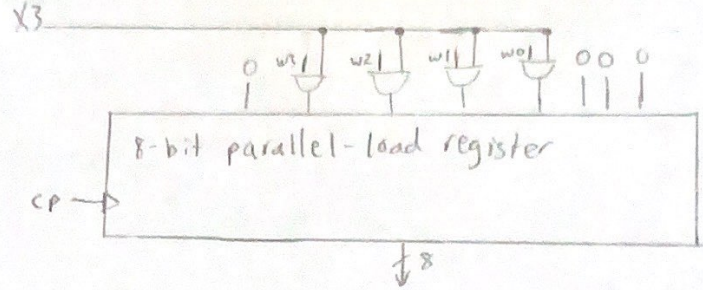
When changing from S4 back to initial, both the select inputs are 1. This means the vector MUX receives the T3 input from Multiplier X3.

After all of this is done, the second 8-bit Parallel load register will hold the final product as all the multiplication will have been completed. The multiplication will have been completed with shifts, ANDs, and an adder. We can then use the output clock to clock the product from the 8-bit parallel load register. The process takes 4 total clock pulses. The result will remain available in the register until the next multiplication (when the start button is pressed again.

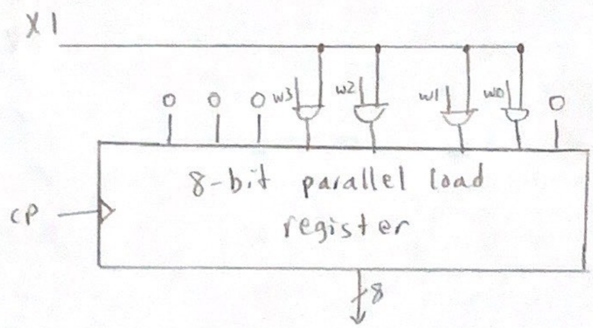
Multiplier X0



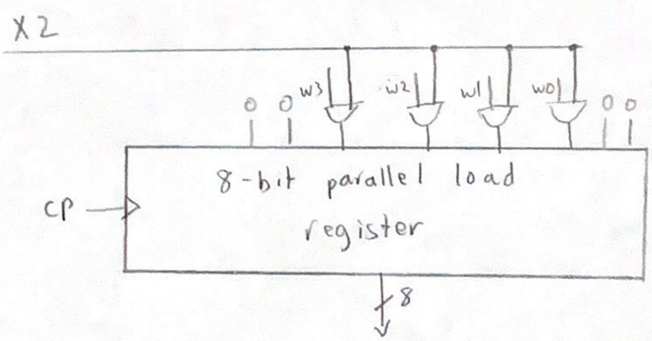
Multiplier X3



Multiplier X1

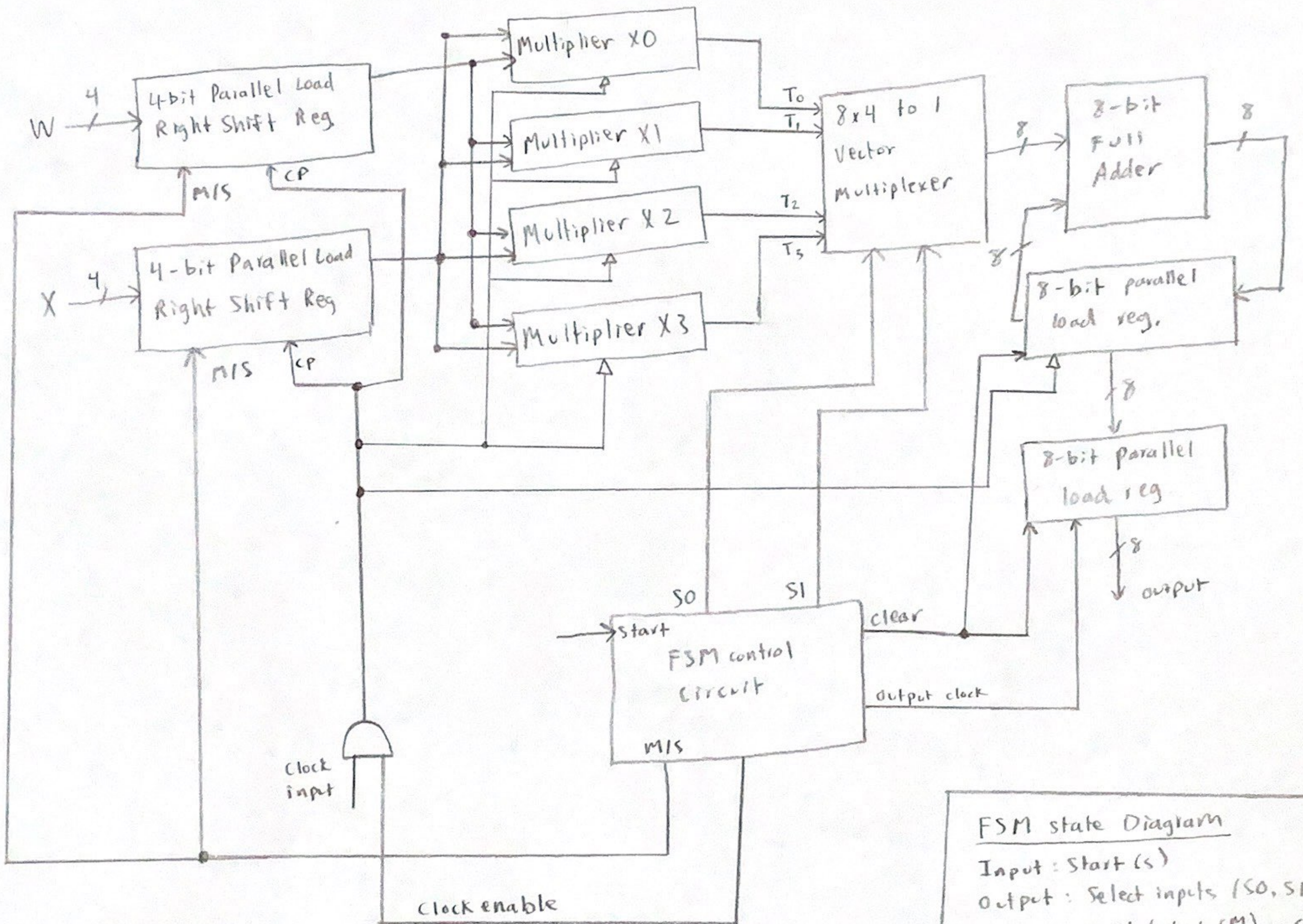


Multiplier X2



The presence of the zeros are so that each result is shifted to the right place to ensure multiplication works properly. See page 1 for example of multiplication; it demonstrates why the zeroes are needed.

FULL Circuit Diagram



Logic for Vector MUX

S0	S1	T
0	0	T0
0	1	T1
1	0	T2
1	1	T3

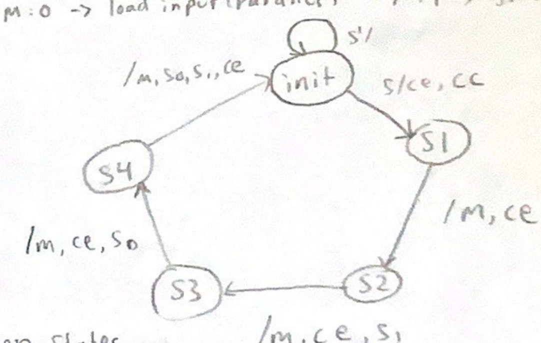
FSM state Diagram

Input : Start (s)

Output : Select inputs (S0, S1), output clock (O), clear (cc)

mode/select (M), clock enable (ce)

M:0 -> load input (parallel) M:1 -> Shift!



Blank input -> any input

Blank output -> no output

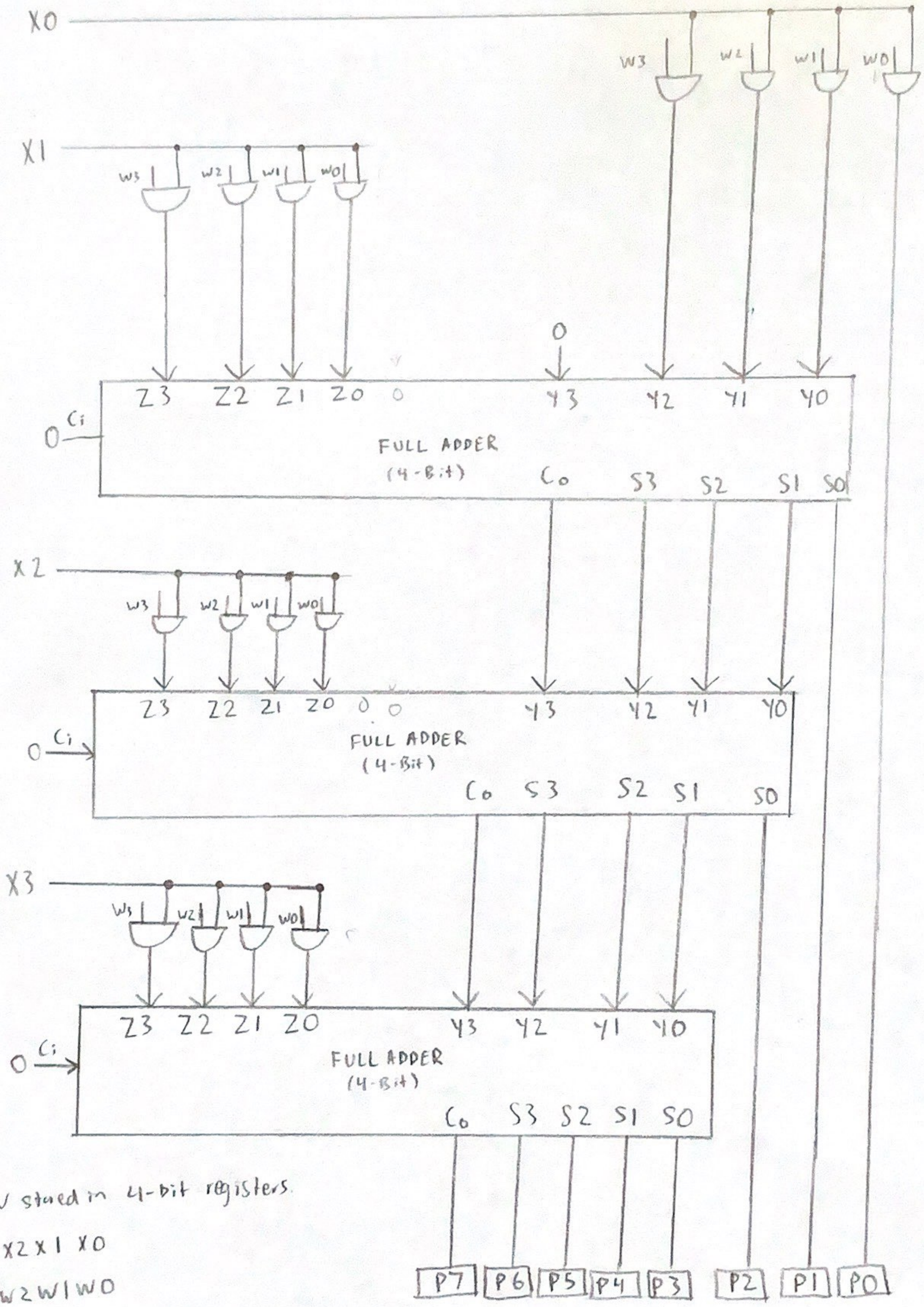
S1 : 1st shift S3 : 3rd shift

S2 : 2nd shift S4 : 4th shift

State reduction is impossible because there is never the same outputs shared between states.

Circuit Diagram for 4-bit Multiplier

-> This is extra, but I thought this could be another way to implement 4-bit multiplication. I feel like this is easier and simpler to understand.



Both X, W stored in 4-bit registers.

X = X3X2X1X0

W = W3W2W1W0

For the sake of organization and clarity I did not connect every single X3, X2, X1, X0 back to X and every single W3, W2, W1, W0 back to W. It would've made the diagram very cluttered and I believe my diagram effectively gets the point across.