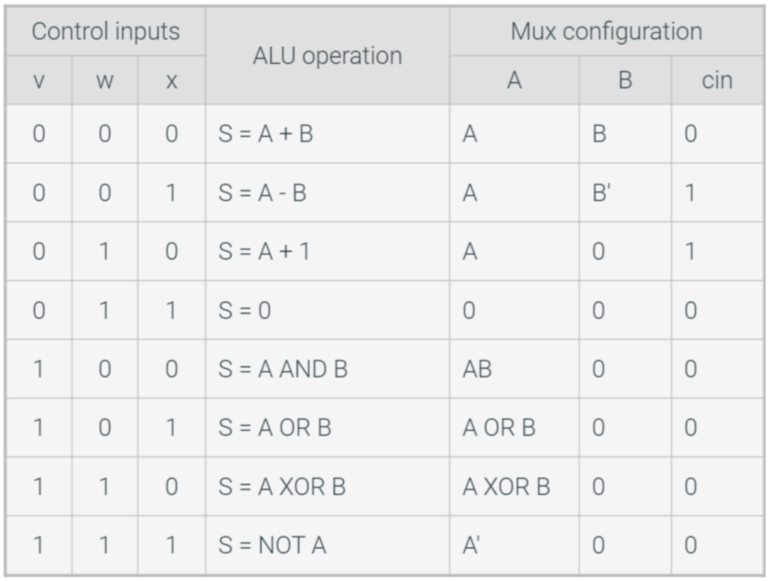
Joshua Pollock

EE 310 – Lab 4 Report

NAU, 1 March 2020

Problem Description

In this lab, we have been asked to design an arithmetic logic unit (ALU) with the operation table given to us. Input A, input B, and output S from the ALU will be 3 bits in size. Control inputs V, W and X are 1 bit in size. V, W, and X will select which operation to take place with V being the MSB and X being the LSB. The inputs A and B are 3 bits and will contain the numbers that will be used in the operations (A+B, A-B, etc.). The output S will also be 3 bits and will be where the operation result is stored.



Control inputs			ALU operation	Mux configuration		
v	w	x		A	B	cin
0	0	0	S = A + B	A	B	0
0	0	1	S = A - B	A	B'	1
0	1	0	S = A + 1	A	0	1
0	1	1	S = 0	0	0	0
1	0	0	S = A AND B	AB	0	0
1	0	1	S = A OR B	A OR B	0	0
1	1	0	S = A XOR B	A XOR B	0	0
1	1	1	S = NOT A	A'	0	0

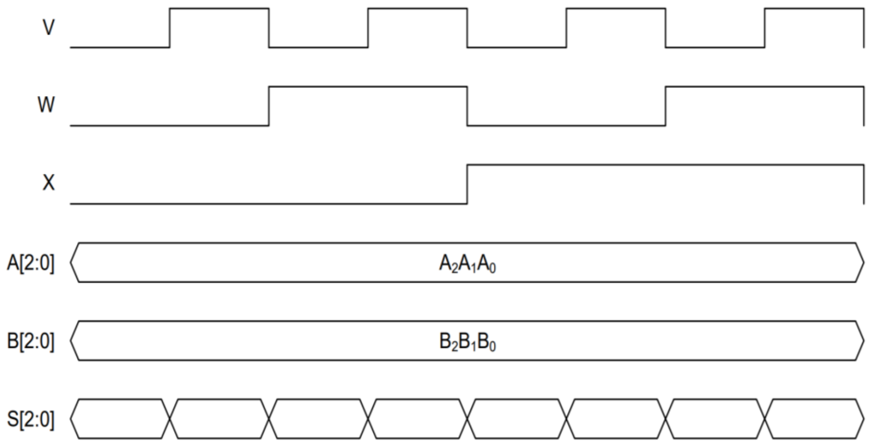


Figure 1. Expected behavior of the circuit

Solution Plan

In order to solve the problem explained above, I created one case statement for all 8 operations based off the given table (Figure 1). This was done using {V, W, X}, where V is the MSB and X is the LSB. For example, if V = 1, W = 0, and X=1 the result would be the binary value 101 or decimal 5. This example would OR input A with input B and store the result in S. This case statement is kept within the "always @ (V, W, X, A, B) begin" and will wait for change in V, W, X, A, or B. If one of these values change, the case statement will execute.

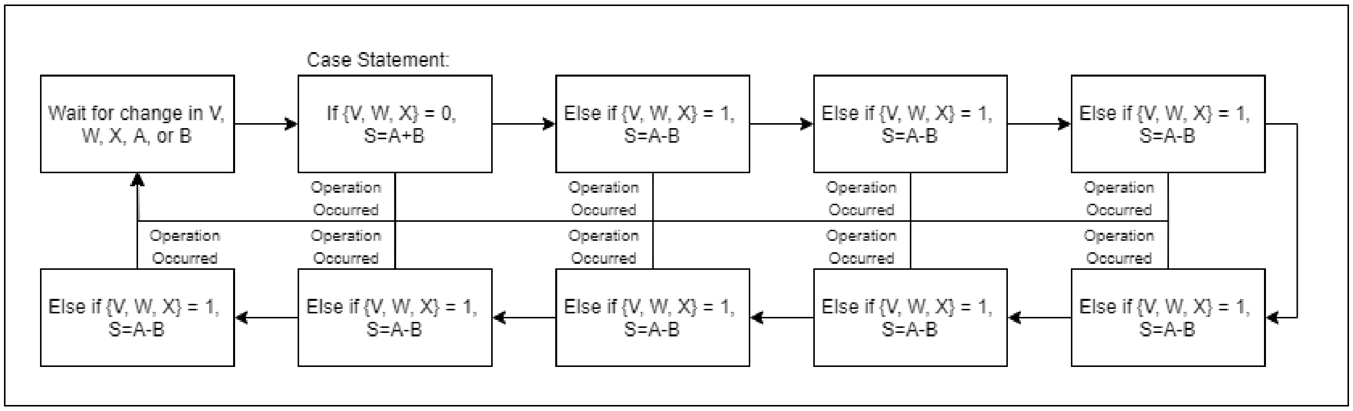

Case Statement:
Wait for change in V, W, X, A, or B
If {V, W, X} = 0, S=A+B
Else if {V, W, X} = 1, S=A-B
Else if {V, W, X} = 1, S=A-B
Else if {V, W, X} = 1, S=A-B
Operation Occurred
Else if {V, W, X} = 1, S=A-B
Else if {V, W, X} = 1, S=A-B
Else if {V, W, X} = 1, S=A-B
Else if {V, W, X} = 1, S=A-B
Else if {V, W, X} = 1, S=A-B

Figure 2. State diagram for the proposed solution

Implementation and Test Plan

I have implemented the solution plan explained above, by first completing the prelim of the lab, creating the main program and a testbench to simulate the program. The testbench program for this lab was very simple and easy to write. First, we needed to convert the last two numbers of our student ID to binary. My last two numbers were 0 and 2, resulting in 0000 0010. Converting this to two 3-bit numbers resulted in A=000 and B=010. From here using the given timing diagram, I manipulated V, W, and X based on the waveforms (The states of {V, W, X} were 000, 100, 010, 110, 001, 101, 011, and 111). Once the prelimainary lab was completed, all that was left was to plug in the FPGA and run the code. In the baseline file the inputs were bound to various switches. Input V was bound to switch 8, W was bound to switch 7, X was bound to switch 6, A was bound to switches 5 through 3, and X was bound to switches 2 through 0. Lastly the output S was bound to green LEDs 2 through 0. Testing the code on the FPGA gave me no errors and the lab was completed quite quickly.

Lab4.v
```
        module Lab4 ( V , W , X , A , B , S ) ;

        input  V, W , X ;
        input  [2:0]  A , B ;
        output reg  [2:0]  S ;

        always  @ (V, W, X, A, B)  begin
```

```
		case  ({ V, W , X })
			0  :  S  =  A + B;
			1  :  S  =  A - B;
			2  :  S  =  A + 1;
			3  :  S  =  0;
			4  :  S  =  A & B;
			5  :  S  =  A | B;
			6  :  S  =  A ^ B;
			7  :  S  =  ~A;
		endcase
	end

	endmodule
```

Lab4_tb.v

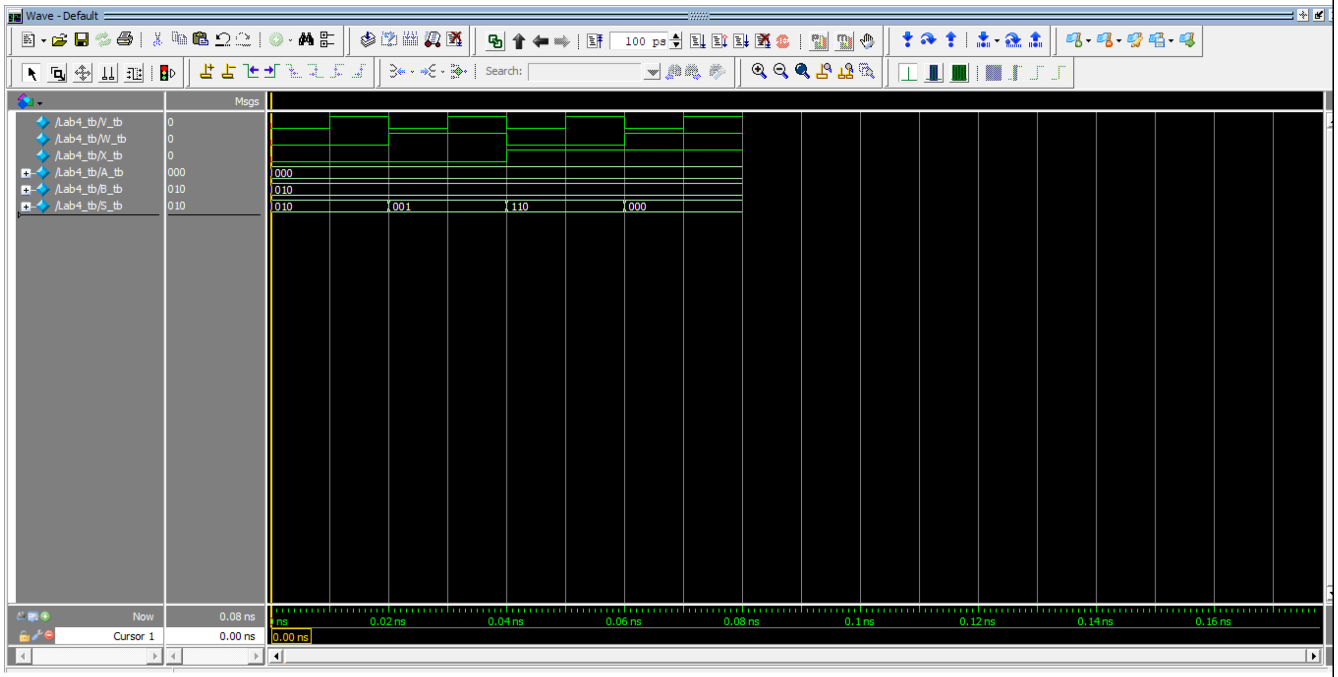
```
	module  Lab4_tb;

	reg   V_tb, W_tb, X_tb;
	reg   [2:0]  A_tb , B_tb;
	wire  [2:0]  S_tb;

	localparam  PER  =  10;

	Lab4  dut  (V_tb, W_tb, X_tb, A_tb, B_tb, S_tb);

	initial begin
		// 5209702 -> 02 -> 000 010 -> A=000 B=010
		A_tb  =  3'b000;
		B_tb  =  3'b010;

		V_tb = 1'b0;
		W_tb  =  1'b0;
		X_tb  =  1'b0;
		#(PER);
		V_tb = 1'b1;
		W_tb  =  1'b0;
		X_tb  =  1'b0;
		#(PER);
		V_tb = 1'b0;
		W_tb  =  1'b1;
		X_tb  =  1'b0;
		#(PER);
		V_tb = 1'b1;
		W_tb  =  1'b1;
		X_tb  =  1'b0;
		#(PER);
```

```
                V_tb = 1'b0;
                W_tb  =  1'b0;
                X_tb  =  1'b1;
                #(PER);
                V_tb = 1'b1;
                W_tb  =  1'b0;
                X_tb  =  1'b1;
                #(PER);
                V_tb = 1'b0;
                W_tb  =  1'b1;
                X_tb  =  1'b1;
                #(PER);
                V_tb = 1'b1;
                W_tb  =  1'b1;
                X_tb  =  1'b1;
                #(PER);
                $stop;
        end
        endmodule
```

baseline_c5gx.v (Only one line added)

```
        Lab4 dut ( SW[8] , SW[7] , SW[6] , SW[5:3] , SW[2:0] , LEDG[2:0]);
```

Figure 3. Verilog code for the proposed solution



```
/Lab4_tb/V_tb   0
/Lab4_tb/W_tb   0
/Lab4_tb/X_tb   0
/Lab4_tb/A_tb   000    000
/Lab4_tb/B_tb   010    010
/Lab4_tb/S_tb   010    010  001  110  000

Now       0.08 ns
Cursor 1  0.00 ns
```

Figure 4. Lab pictures of the running solution