Simple Processor Documentation

Anjelo Gana

Hardware:

I used The Go Board from Nandland.com

https://nandland.com/the-go-board/

Project Description:

Convert the last 4 digits of our student number into 2 separate 8 bit registers A and B. Convert that Hex number into Binary.

Student Number: 501085**972**

$A = (59)_{16}$ $B = (72)_{16}$

$A = (0101\ 1001)_2$ $B = (0111\ 0010)_2$

Read the Finite State Machine section for more info about states, and Decoder for more info.

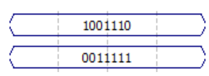
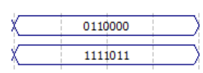
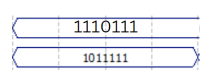
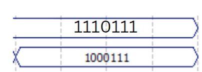
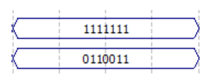
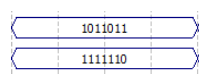
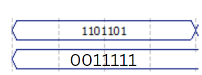
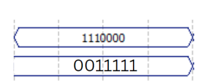
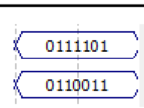
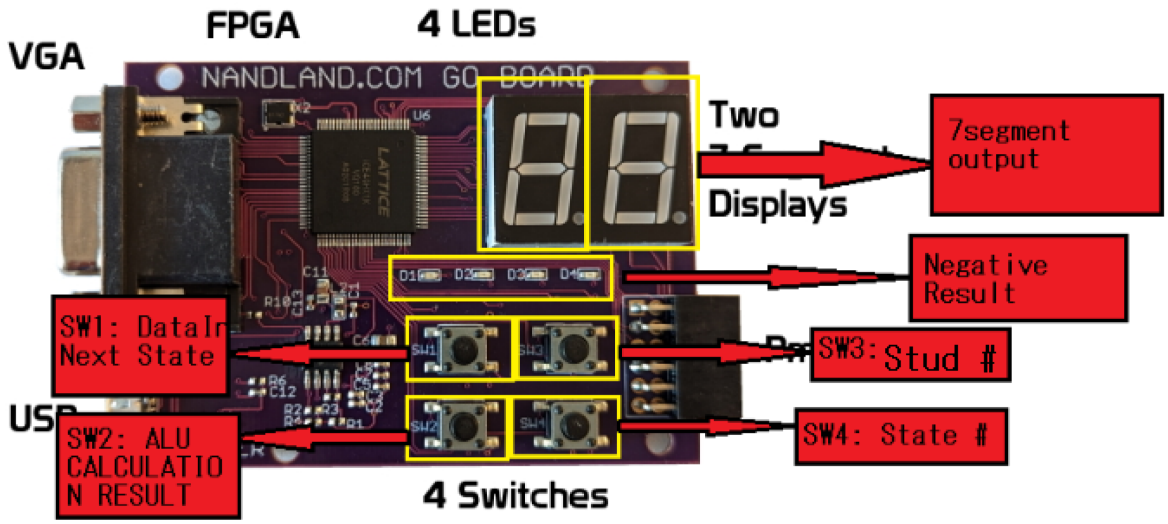
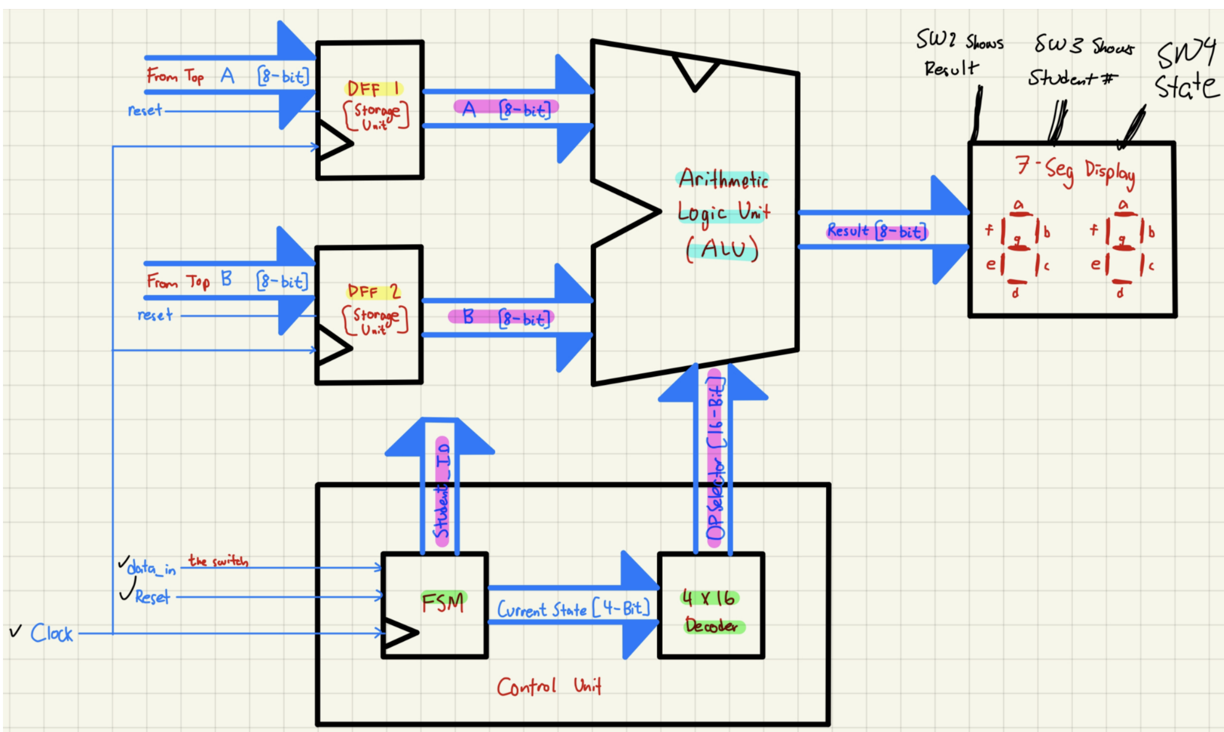
Microcode	Boolean Operation	7sseg	Binary to 7sseg	Binary Calculation
0000000000000001	$sum\ (A, B)$		"1100" => "1001110" "1011" => "0011111"	11001011
0000000000000010	$diff\ (A, B)$		"0001" => "0110000" "1001" => "1111011"	-00011001
0000000000000100	$\overline{A}$		"1010" => "1110111" "0110" => "1011111"	10100110
0000000000001000	$\overline{A \bullet B}$		"1010" => "1110111" "1111" => "1000111"	10101111
0000000000010000	$\overline{A + B}$		"1000" => "1111111" "0100" => "0110011"	10000100
0000000000100000	$A \bullet B$		"0101" => "1011011" "0000" => "1111110"	01010000
0000000001000000	$A \oplus B$		"0010" => "1101101" "1011" => "0011111"	00101011
0000000010000000	$A + B$		"0111" => "1110000" "1011" => "0011111"	01111011
0000000100000000	$\overline{A \oplus B}$		"1101" => "0111101" "0100" => "0110011"	11010100

Table 1.0: Part 1 Results

Place and Route:



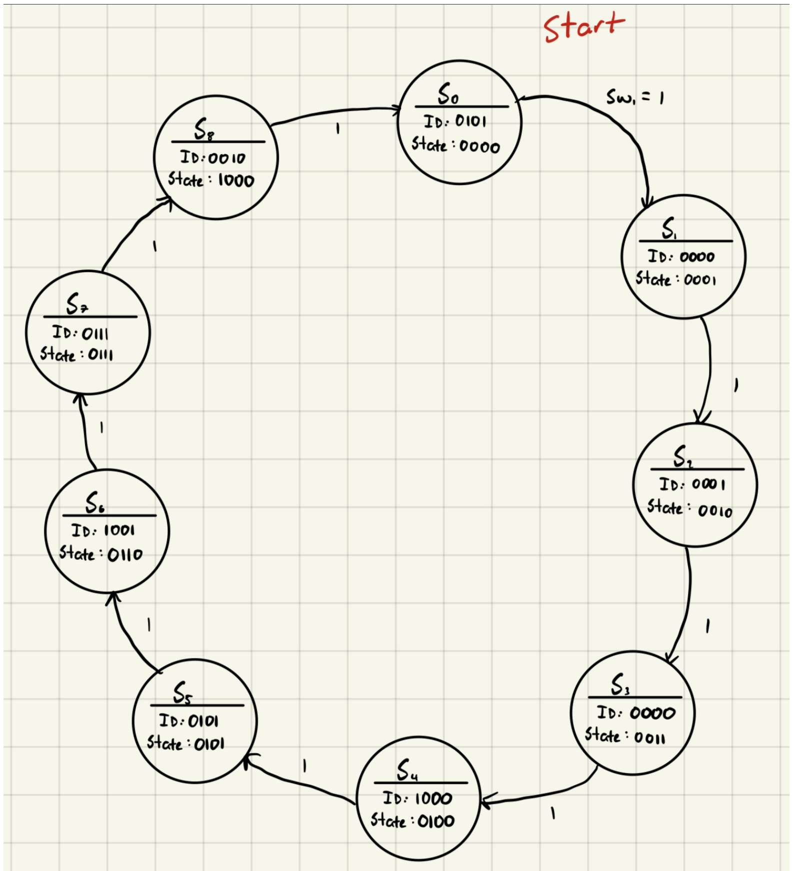
Schematic Diagram:

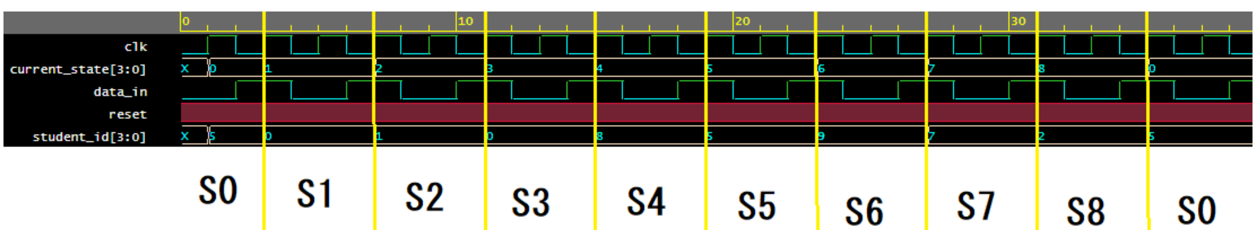


I used the 4 main components to make a simple processor:

- **Control Unit** [FSM, 4x16 Decoder] - *Green Highlight*
- **Storage Unit** [D-Flip-Flop] - *Yellow Highlight*
- **ALU** - *Cyan Highlight*
- **Control Bus** - *Pink Highlight*

Finite State Machine:
https://edaplayground.com/x/edMV





From the schematic diagram, the **current_state** goes to the input of the decoder to decode which microcode to use.

Outputs:

SW1: Goes to the next state, and outputs state number,
SW2: Shows the ALU Results
SW3: Shows the Student #
SW4: Shows the state again

Decoder:

```
case(current_state)
    4'b0000 : selector = 16'b0000000000000001;
    4'b0001 : selector = 16'b0000000000000010;
    4'b0010 : selector = 16'b0000000000000100;
    4'b0011 : selector = 16'b0000000000001000;
    4'b0100 : selector = 16'b0000000000010000;
    4'b0101 : selector = 16'b0000000000100000;
    4'b0110 : selector = 16'b0000000001000000;
    4'b0111 : selector = 16'b0000000010000000;
    4'b1000 : selector = 16'b0000000100000000;
    default: selector  = 16'b0000000000000000;
endcase
```

From **Table 1.0** we can see that if we are in state 0, we do sum(a,b) since it's the direct microcode for it. We follow this general rule for the other 8 states.