8-Bit Calculator

Jaskaran Singh
John Nasir
Alyssa Lafever
Nhi Vu

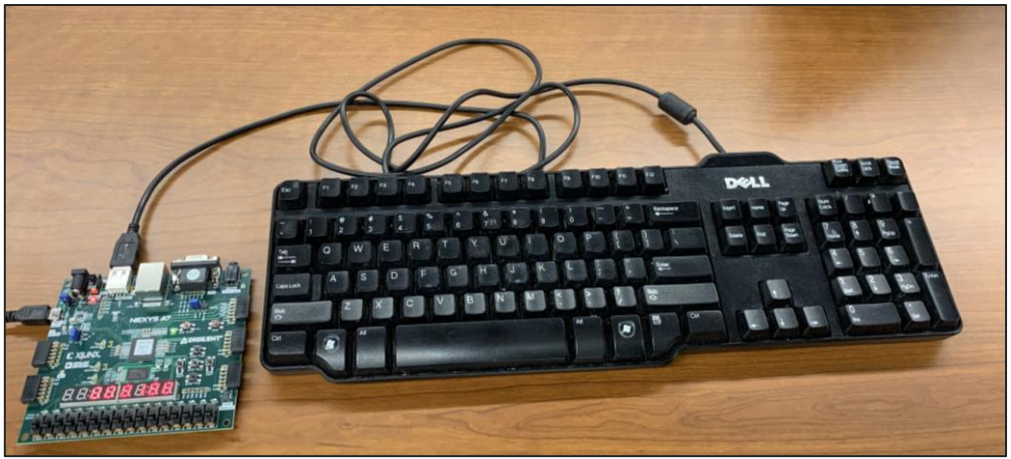
Component Used

Hardware

- Nexys A7 FPGA board
- 7 segment display
- USB computer keyboard

Software

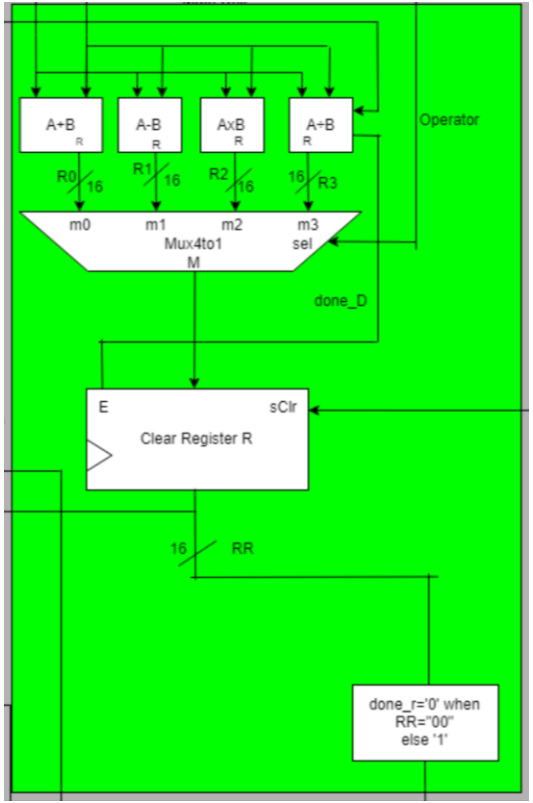
- Vivado



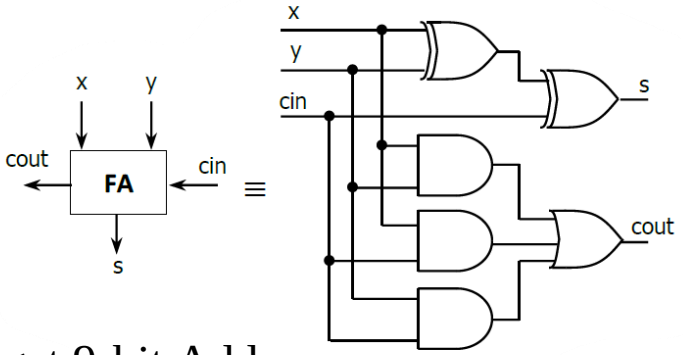
The Operations

Math Unit

- Top-file of the 4 math operations: +, -, x, /
- Contains 4 inputs: operand A, operand B, math operator, enable
- Data goes into each math function
- Result is outputted through a 4-to-1 bus MUX with operator select line

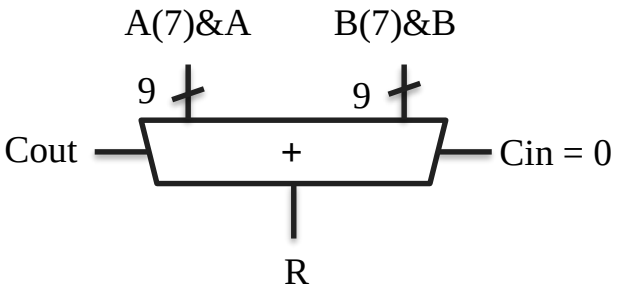


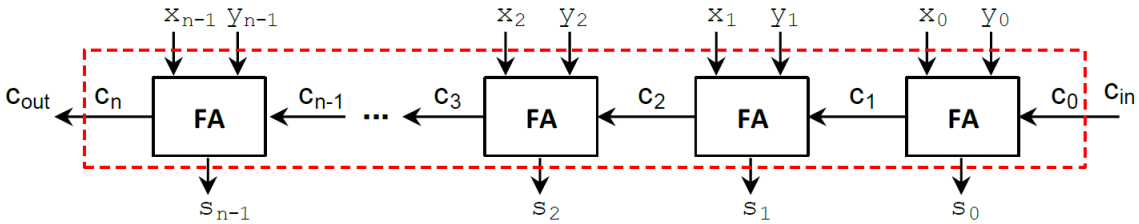
Adder



- Link 9 full-adders through carry ports to get 9-bit Adder
- Sign extend the sum to create 16-bit output

(Notes - Unit 4, Page 13)

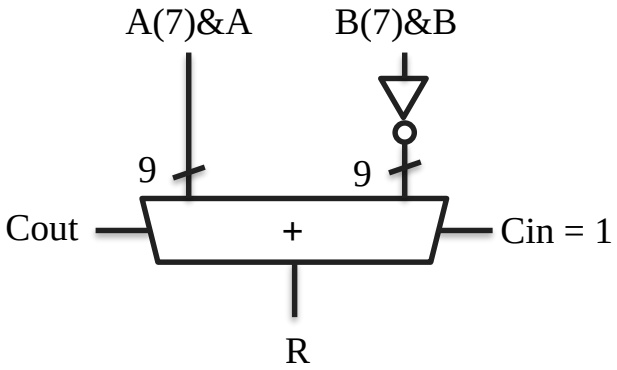




(Notes - Unit 4, Page 13)

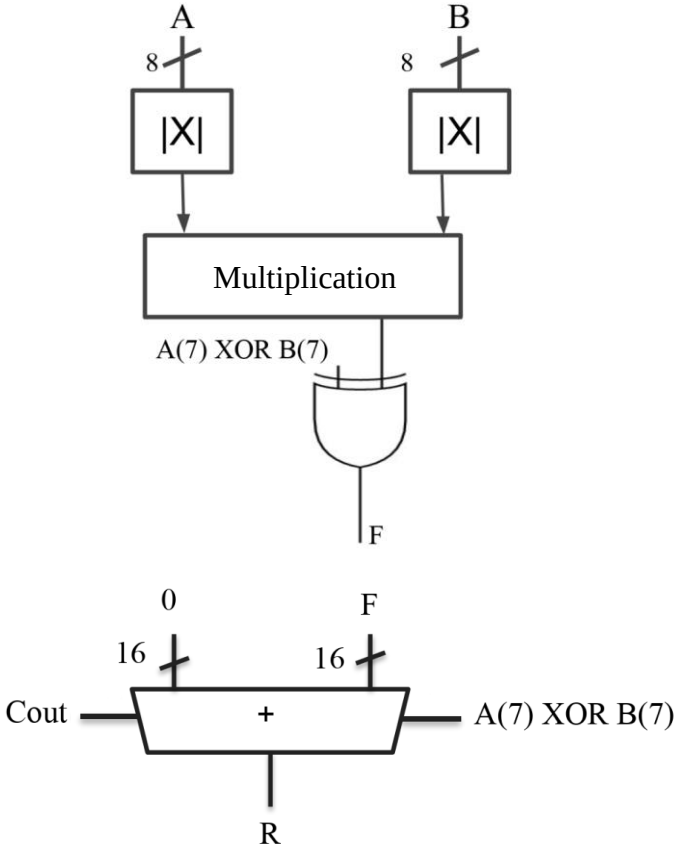
Subtractor

- Link 9 full-adders through carry ports to get 9-bit Subtractor
- Not gate is used to invert operand B before inputting
- Carry-in is 1 instead of 0



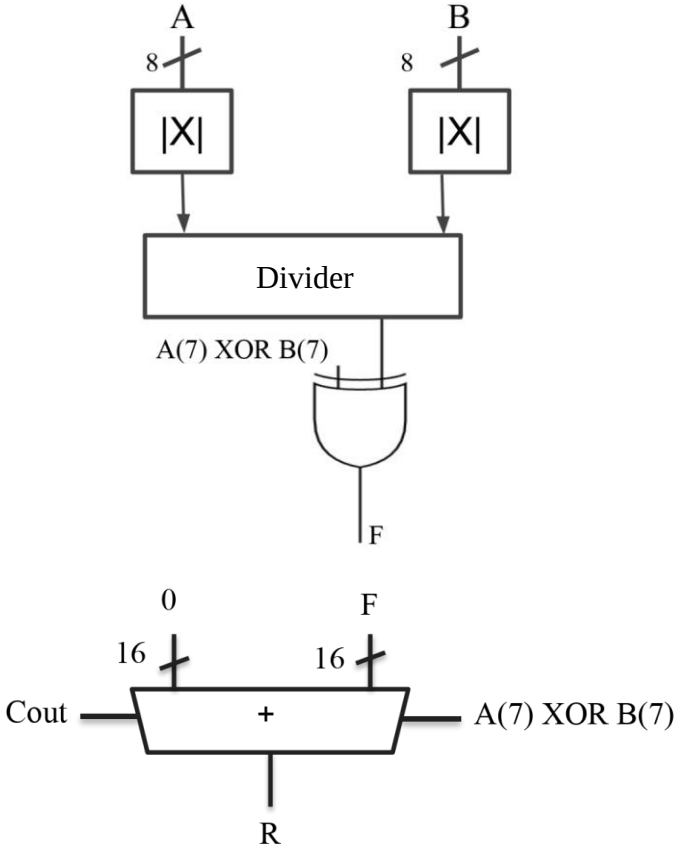
Multiplier

- Takes the magnitude of A and B to convert the numbers from signed to unsigned
- Multiplies the magnitudes of A and B
- Then the output is converted to 2’s complement binary to get the signed output



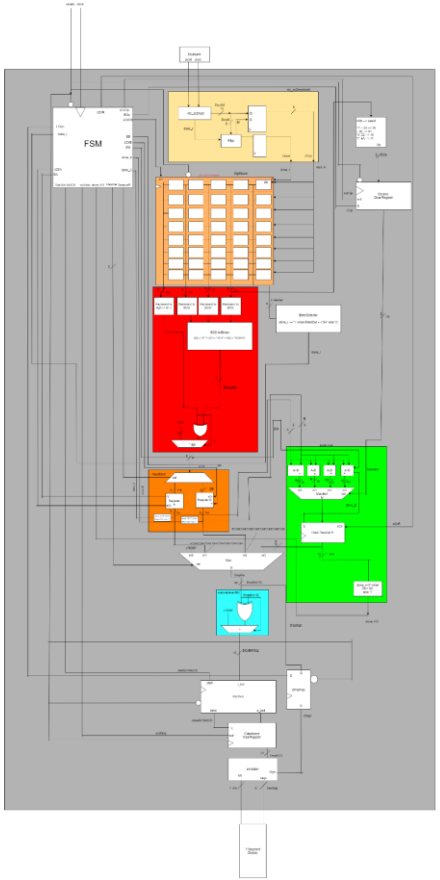
Divider

- Takes the magnitude to convert the numbers into unsigned
- One input goes into a normal register, the other goes into a left shift register
- Divider includes FSM, register file, n-bit adder, another left shift register (one has synchronous clear) and a counter



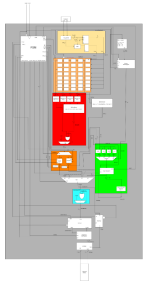
Top File Components

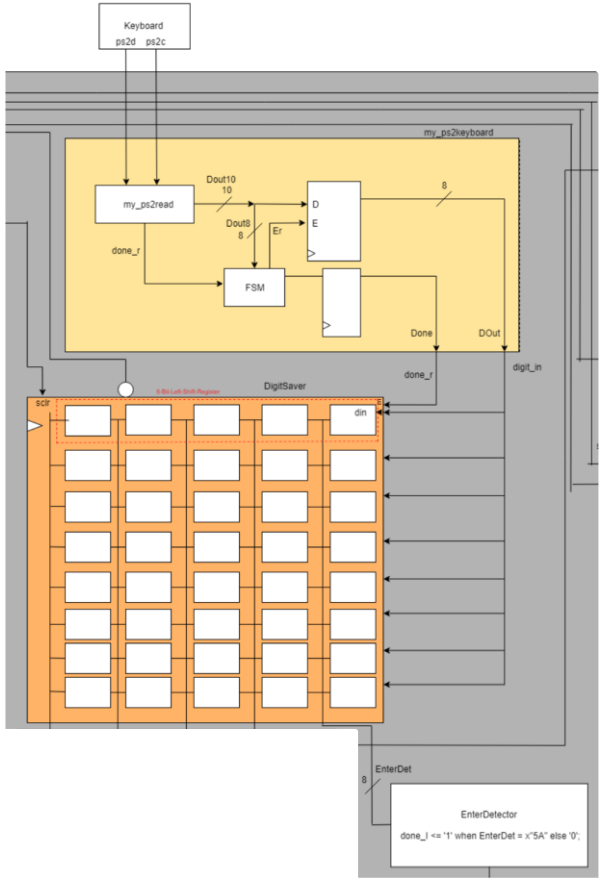
Top File



Input – Digit Saver

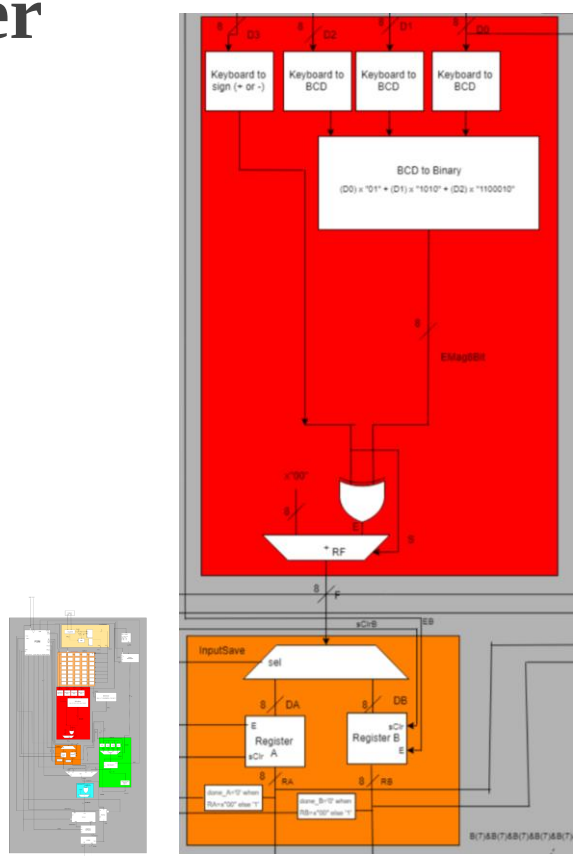
- FPGA receives input data into Dr. Llamocca's my_ps2keyboard component
- 8 shift registers shift in each digit's 8-bit keycode
- Enter detector detects when Enter keycode is inputted





Input – Input Saver

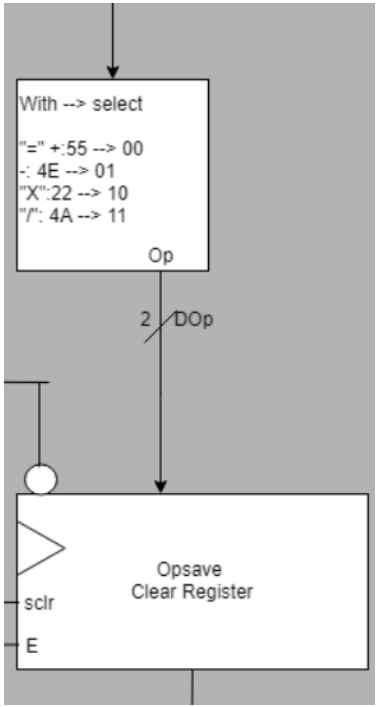
- Converts keycode to 2’s complement binary
- Demultiplexer stores operands in appropriate registers

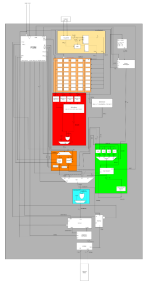


Key	Hex Keycode	Keycode
0	45	0100 0101
1	16	0001 0110
2	1E	0001 1110
3	26	0010 0110
4	25	0010 0101
5	2E	0010 1110
6	36	0011 0110
7	3D	0011 1101
8	3E	0011 1110
9	46	0100 0110
+	55	0101 0101
-	4E	0100 1110
x	22	0010 0010
/	4A	0100 1010
Enter	5A	0101 1010

Input – Operator Saver

- Operator gets encoded to a 2-bit binary number
- Encoded operator is saved in a 2-bit register



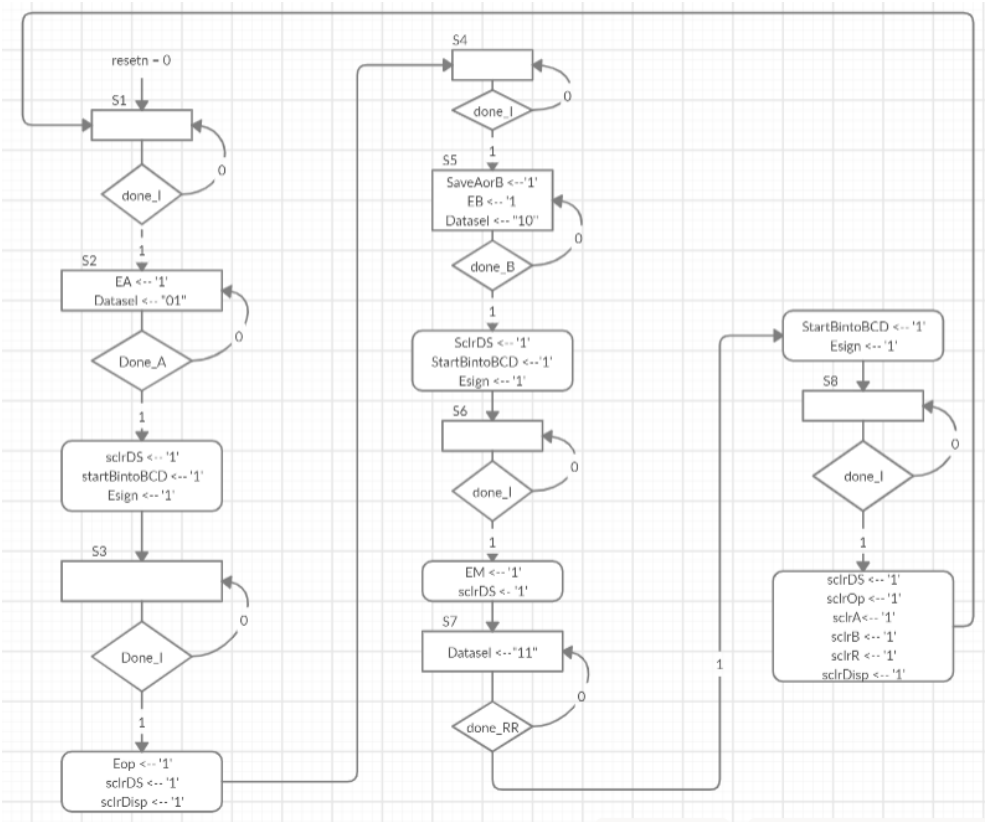


FSM

- The FSM controls all of the data path components in the calculator
- After all the inputs have been entered and output is shown, it will return to original state

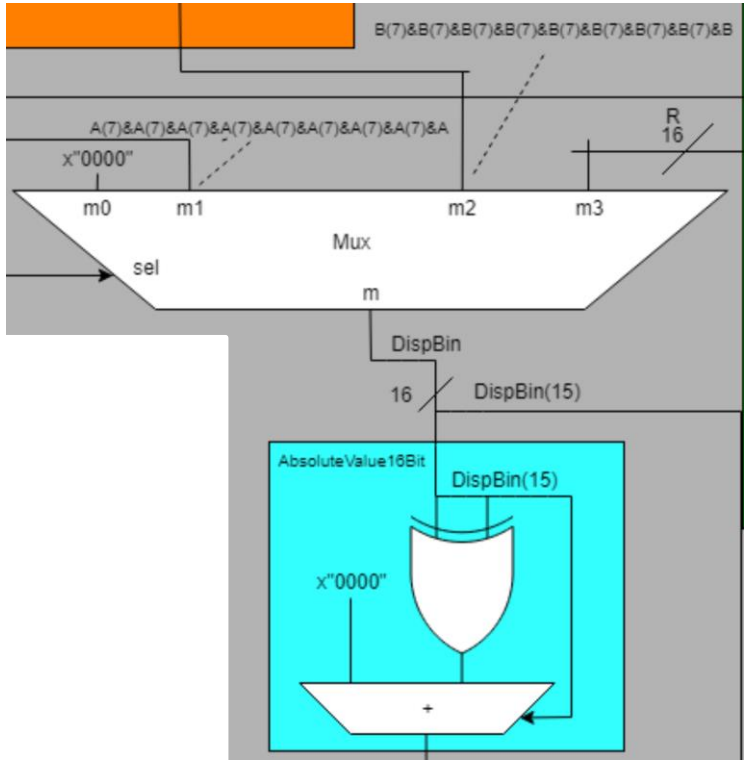
State	Description	Notes
S1	Receive Operand A	
S2	Save Operand A	We'll be stuck in S2 if Operand A is 0
S3	Display Operand A on 7Seg; Receive Operator, Save Operator	
S4	Receive Operand B	
S5	Save Operand B	We'll be stuck in S5 if Operand B is 0
S6	Display Operand B on 7Seg; Receive Calculate Command	
S7	Execute Math Operation, Convert Output to BCD	We'll be stuck in S7 if Result R is 0
S8	Display Output on 7Seg	

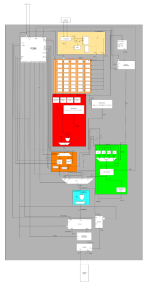
ASM


resetn = 0
S1
done_I
0
1
S2
EA <-- '1'
Datasel <-- "01"
Done_A
0
1
sclrDS <-- '1'
startBintoBCD <-- '1'
Esign <-- '1'
S3
Done_I
0
1
Eop <-- '1'
sclrDS <-- '1'
sclrDisp <-- '1'
S4
done_I
0
1
S5
SaveAorB <--'1'
EB <-- '1
Datasel <-- "10"
done_B
0
1
SclrDS <-- '1'
StartBintoBCD <--'1'
Esign <-- '1'
S6
done_I
0
1
EM <-- '1'
sclrDS <- '1'
S7
Datasel <--"11"
done_RR
0
1
StartBintoBCD <-- '1'
Esign <-- '1'
S8
done_I
0
1
sclrDS <-- '1'
sclrOp <-- '1'
sclrA<-- '1'
sclrB <-- '1'
sclrR <-- '1'
sclrDisp <-- '1'

Output

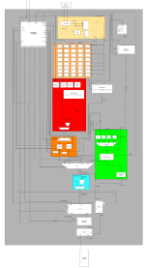
- Multiplixer outputs either operand A, operand B, or the result R.
- 16-bit absolute value

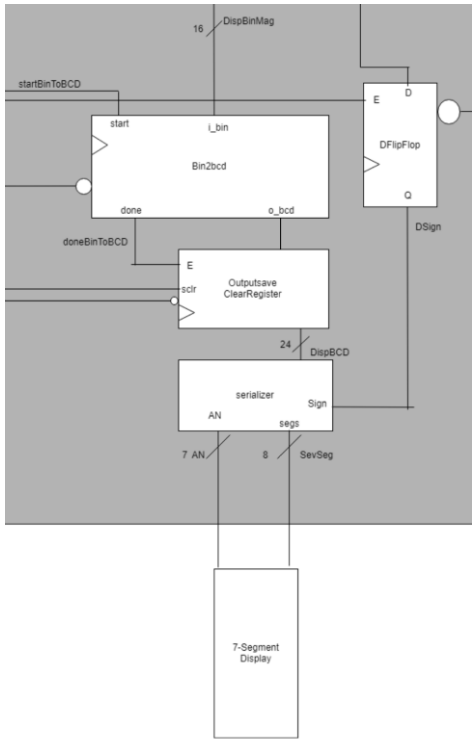




Output

- Binary to BCD converter
- Register stores output value as 24-bit BCD
- D flip flop stores 1-bit sign
- Serializer sequentially cycles data and power to seven-segment displays





Challenges

- Designing the input and output data path circuit
- Multiplier didn’t properly function at first for negative numbers
- Showing the negative sign on the seven-segment display

Constraints

- Number range is from -128 to 127
- The operands A and B cannot equal zero
- The result R cannot equal zero
- Negative numbers must be inputted as three digit numbers
  (e.g. -5 would be inputted as -005)
- Divider outputs truncated integers only

Demonstration

- (-060) + (090) = 30
- (-025) – (-012) = -13
- (-040) x (127) = -5080
- (-070) / (-004) = 17 (The decimal is truncated)

Questions?