EE 421/621, Fall 2024

Project 1: ALU Design

October 29th, 2024

Michael Pittenger

Background

An Arithmetic Logic Unit (ALU) is the component in a computer's Central Processing Unit (CPU) that is responsible for carrying out arithmetic and logic operations on the binary data being processed. Depending on the instruction, the ALU will output different values corresponding to the various arithmetic and logic operations designed to work in the hardware. D Flip-Flops or D-Latches can be used to sync the loading of the input and output signals, allowing for the input and output signals to change only when valid.

Design

The structure of the ALU that we must design is shown in **Figure 1**, with the operations of the ALU shown in **Table 1**. We must design a circuit that efficiently incorporates all of the elements in our design and that is also able to execute the various functions given the specific instruction. The logic deriving the instructions to operations are shown in **Figure 2** and taken from the class slides to use as a base.

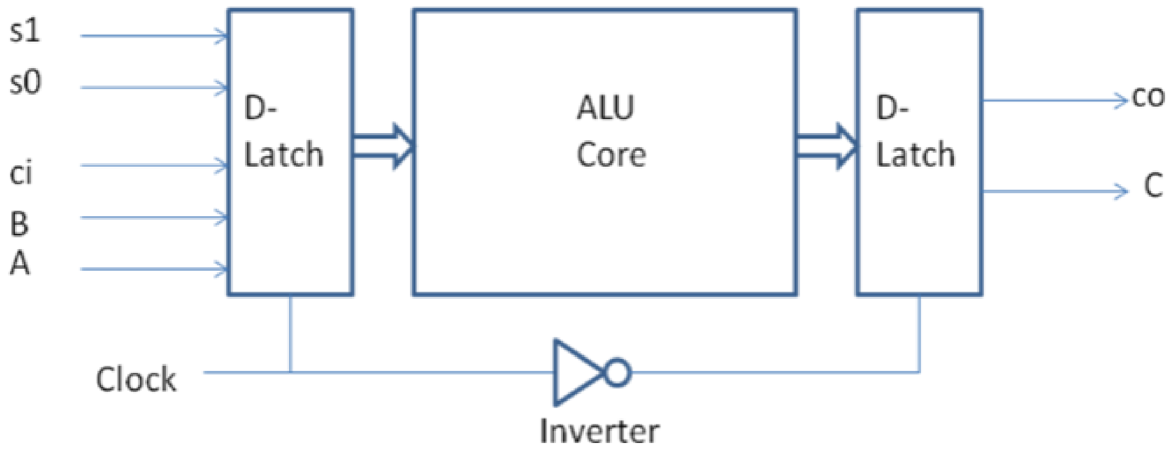
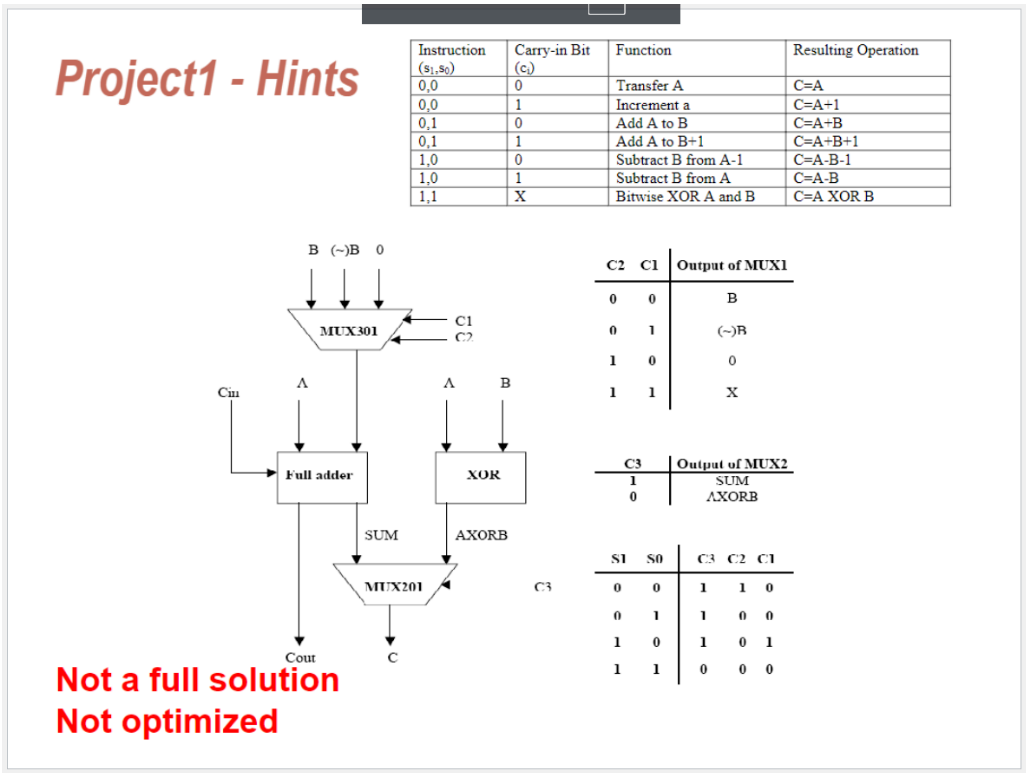


Figure 1: ALU Design

Table 1: ALU Operations

Instruction (s_1,s_0)	Carry-in Bit (c_i)	Function	Resulting Operation
0,0	0	Transfer A	C=A
0,0	1	Increment A	C=A+1
0,1	0	Add A to B	C=A+B
0,1	1	Add A to B+1	C=A+B+1
1,0	0	Subtract B from A-1	C=A-B-1
1,0	1	Subtract B from A	C=A-B
1,1	X	Bitwise XOR A and B	C=A XOR B



Project1 - Hints

Instruction (s_1,s_0)	Carry-in Bit (c_i)	Function	Resulting Operation
0,0	0	Transfer A	C=A
0,0	1	Increment a	C=A+1
0,1	0	Add A to B	C=A+B
0,1	1	Add A to B+1	C=A+B+1
1,0	0	Subtract B from A-1	C=A-B-1
1,0	1	Subtract B from A	C=A-B
1,1	X	Bitwise XOR A and B	C=A XOR B

C2	C1	Output of MUX1
0	0	B
0	1	(~)B
1	0	0
1	1	X

C3	Output of MUX2
1	SUM
0	AXORB

S1	S0	C3	C2	C1
0	0	1	1	0
0	1	1	0	0
1	0	1	0	1
1	1	0	0	0

Not a full solution
Not optimized

Figure 2: ALU Instruction Set Logic

The RTL Netlist of the completed and programmed VHDL ALU design is shown in **Figure 3**. This circuit implements all of the components and functions outlined in the project description.

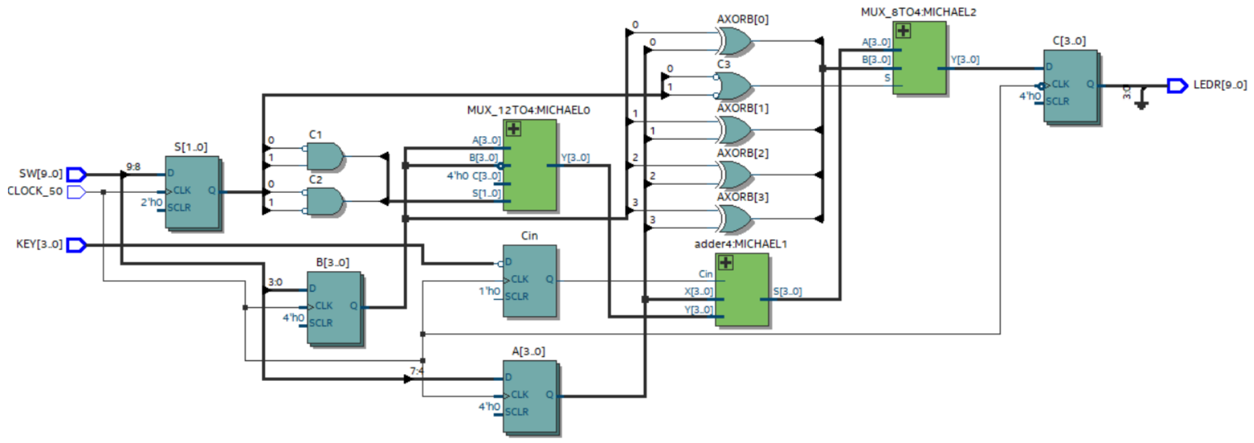


Figure 3: RTL Netlist

Arithmetic

All of the arithmetic is calculated with a single 4-bit ripple carry adder (adder4: MICHAEL1 in **Figure 3**), with the circuit design of the 4-bit adder shown in **Figure 4**. The 4-bit adder is composed of 4 separate full adders chained together, which is the concept behind a ripple carry adder. The circuit of a single full adder is shown in **Figure 5**.

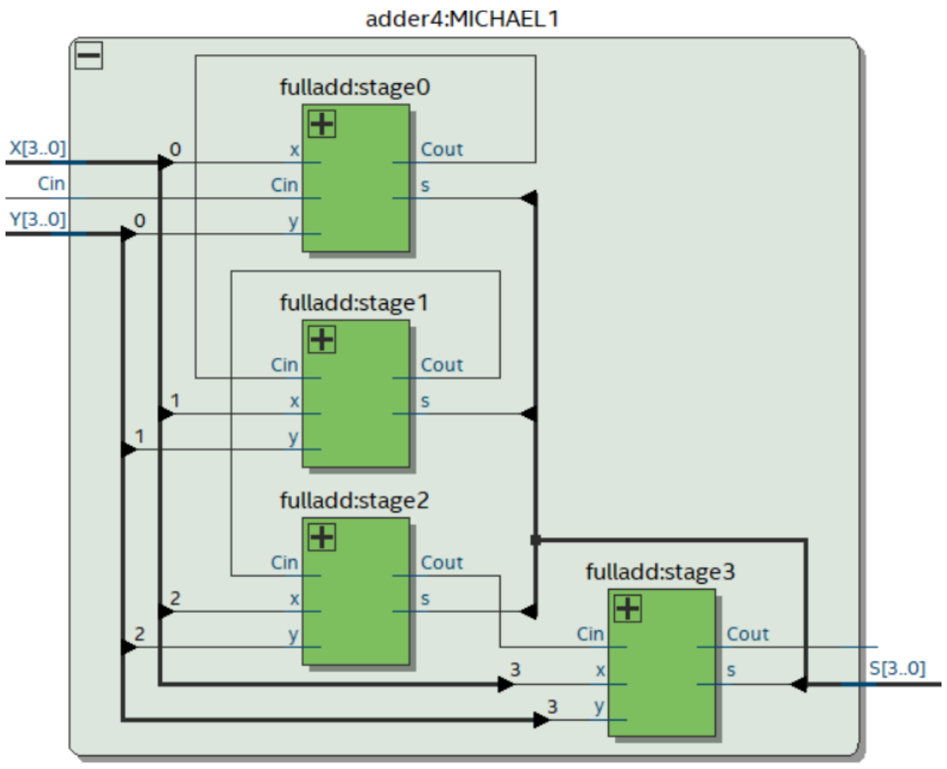


Figure 4: 4-bit Ripple Carry Adder Circuit

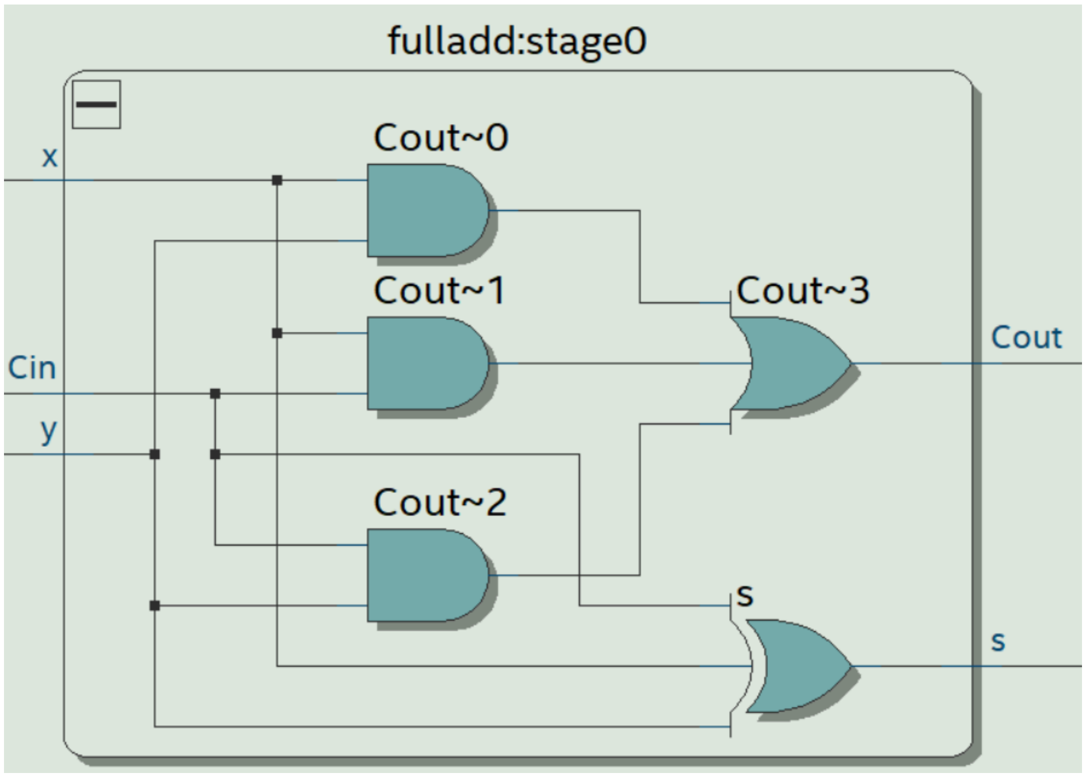


Figure 5: Full Adder Circuit

Arithmetic and Logic Selection

To allow for different arithmetic and logic to be executed, we must use a system of multiplexers (MUXs) to choose what outputs are seen and what binary signals are sent to the adder. **Figure 2** shows the logic behind the instruction bits and the intermediate carry signals in relation to the instruction and the operations being performed. The RTL implementation of the 4-bit 3:1 MUX is shown in **Figure 6** and the RTL implementation of the 4-bit 2:1 MUX is shown in **Figure 7**.

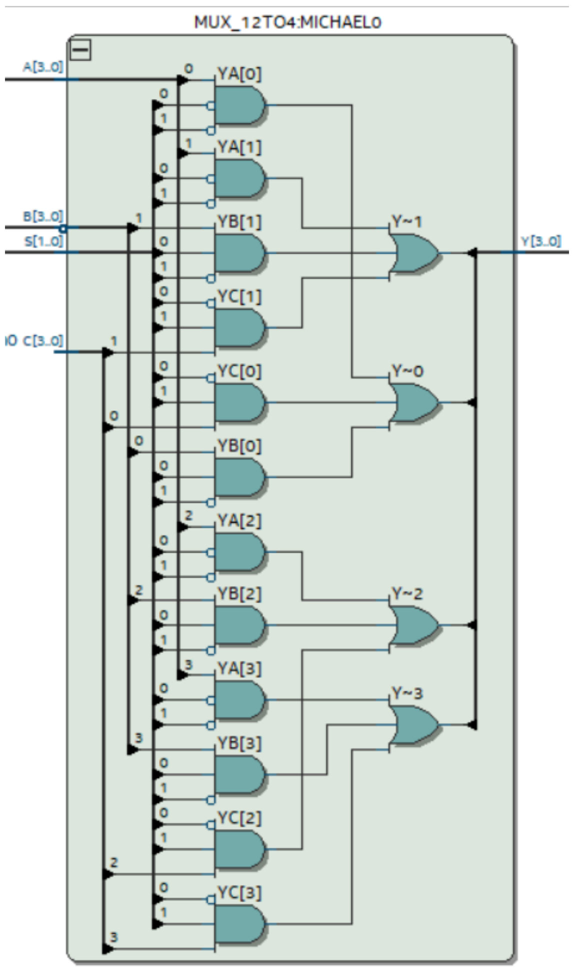


Figure 6: 3 to 1 4-bit MUX Circuit

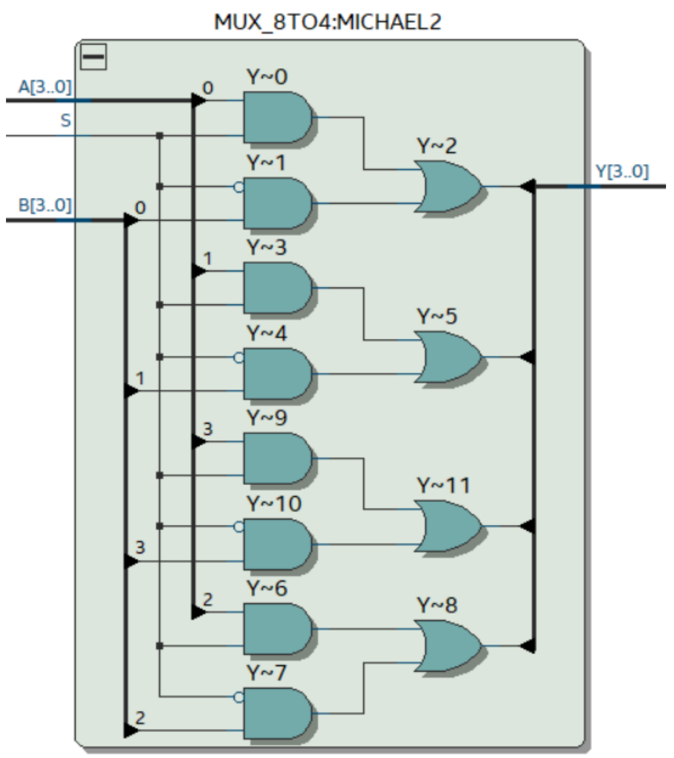


Figure 7: 2 to 1 4-bit MUX Circuit

D Flip Flops

D-latches only allow for a signal to be loaded into its output when the “clock” input is high. For this ALU implementation, D Flip-Flops were used, which only allow the carrying of the signal on the rising edge of the clock input. **Figure 8** shows input signals being stored in D Flip-Flops/registers, with the CLK enable (CLOCK_50 from the FPGA) allowing for the inputs from the switches to load into the A, B, S, and Cin signals only on the rising edge of the clock.

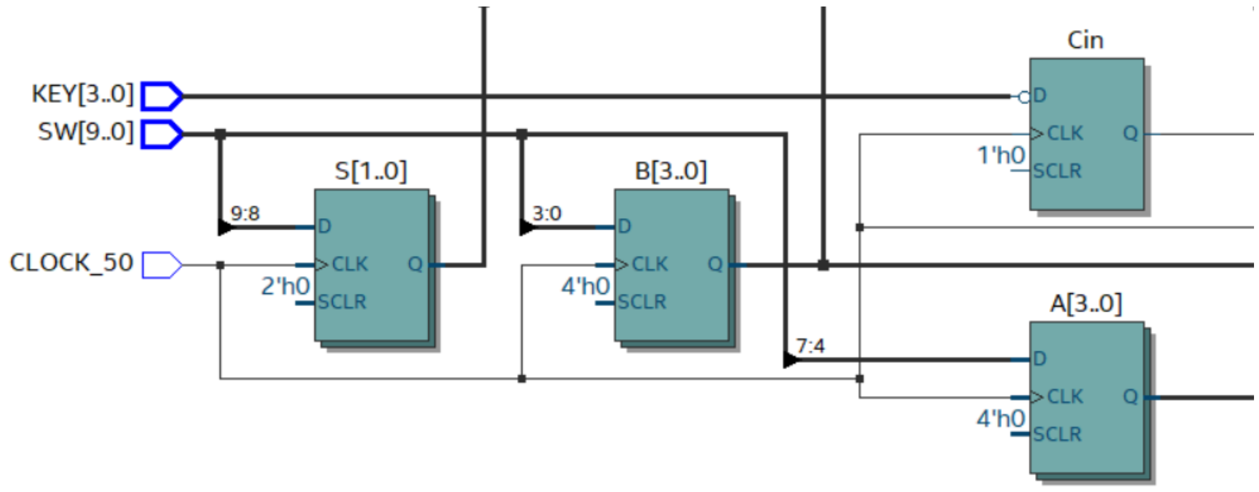


Figure 8: Input DFFs

The output D Flip Flop is used in a similar way for the output signal, but the clock signal is inverted so that it only updates on the falling edge, shown in **Figure 9**.. Every time the clock signal changes from ‘1’ to ‘0’, the value from the Y signal is sent to LEDs 3-0 (LEDR(3 DOWNTO 0)).

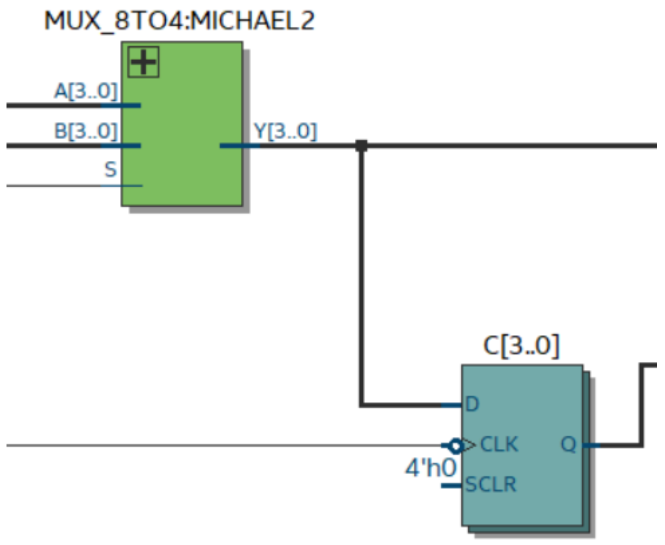


Figure 9: Output DFFs

Display

For better visibility in use and during testing, a circuit was added to the output that converts a 4-bit binary signal into binary coded decimal (BCD) and then outputs that value onto a seven segment display, shown in **Figure 10**. The value output is shown in an unsigned value, but the logic for the 2's complement input values is accurately displayed on the LEDs.

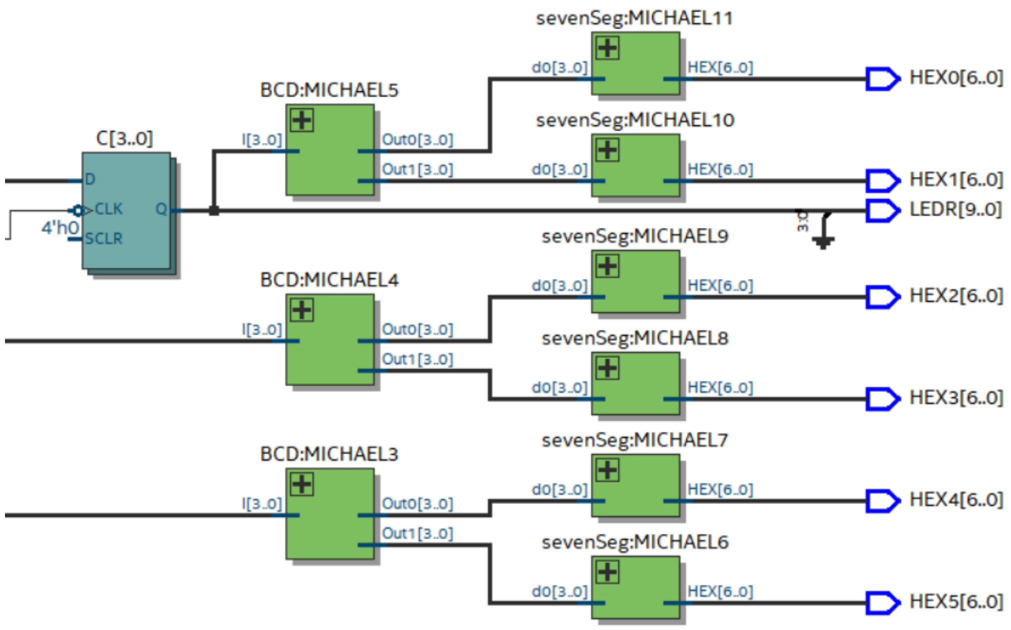


Figure 10: BCD to Seven Segment Display

Conclusion

In summary, this project successfully designs and implements an ALU using VHDL that can take the given instruction signals to execute the corresponding operations. The D Flip-Flops are added to ensure that the input and output signals are only loaded when the signals are valid, such as the rising/falling edge of a clock signal or the input of another enable signal. This project highlights key aspects of digital design and HDL tools, and it also builds a solid foundation for building more complex processing units in hardware.

The VHDL code will be attached as an Appendix.