R·I·T | Kate Gleason College of Engineering

Engineering Lab Report

From: Gautham Mutyala

Date: 28 February 2024

Subject: Bit Serial Adder Subtractor

Abstract

The lab assignment was to design a fully working and functional bit serial adder and subtractor circuit that could add with two's complements and subtract as well. The bit serial adder subtractor would have 4 key parts. The results were then taken with the software Model Sim to test if the schematic would actually work.

Theory

There are 4 main parts of the bit serial adder and subtractor: X accumulator, Y register, serial adder, and a control finite state machine. Each part was designed and tested individually to ensure that the final schematic would work as best as it could. The first part that was designed was the serial adder. The first thing that was done was implementing a full adder in order to start adding binary values. The full adder circuit can be visualized in Figure 6. The adder that was used was a 1 bit full adder. There are 2 outputs for a one bit adder; carry and sum. The sum is when both inputs are added, there usually is one extra output that is over one bit, which makes it impossible for the sum to output since it would be 2 bits instead of one, so the carry out is outputted as its own output. The equation for a 1 bit full adder would be Sum = Cin + Xi + Yi and $Cout$ = XiYi + (Xi $*$ Cin) + (Yi $*$ Cin). A two-bit full adder (Figure 6) works in the same way as the one bit adder, however there are some differences. A two-bit adder has 4 inputs, A has A1 and A0, B has B1 and B0, there is still one carry in. In Figure 6 Xi and Yi have only 2 wires connecting them, and 2 sums to one sum output. This is due to the fact that BUS wires are being used. Bus wires work by using specific syntax on each wire that will assign it a version of Xi, Yi, or Sum. There are now 2 sum outputs, and still one carries out. The full adder is then connected to a D flip flop. The wiring is done so that the carry out value will then become an input for the carry in during the next clock cycle (Figure 7). Since the adder is connected to a D flip flop it can now add and subtract consecutively, being able to follow a clock cycle thanks to a clock being wired into the clock input in the d flip flop (Figure 7), the clock is wire through a NOT gate to make it go off of a negative edge rather than a positive one. The D flip flop follows an asynchronous reset on a negative edge clear. It also has a preset input also being wired through a not gate. After the serial adder was designed, it is turned into a block symbol, then the X accumulator was then designed next.

The X accumulator is a 5 bit shift register that can load a value and then shift it to an output, that output is then sent to the serial adder to be added or subtracted depending on what the output of the Y register is going to be. The X accumulator and the Y register are made with a logic gate called a "d flip flop", the flip flop works by allowing a value in and being stored when the clock input is equal to 0, and when the clock is equal to 1, the clock will then finally allow the stored value to be released, or loaded into whatever it is wired into. The D flip flops are made

with 2 D latches, where each latch acts as a door, where if the clock is one then the value will be let through, and if the clock into the latch is 0 then the latch blocks the information from going through. A data register can then be made from a set of multiple flip flops(depending on how many bits need to be stored), the .The X accumulator circuit is based on a modified MOD 11 circuit that was asynchronous, the mod 11 circuit was made synchronous using multiplexer division, from the 9th lab assignment (Figure 8). The problem with the original MOD 11 4 bit counter is that it is asynchronous, this means that the counter will be way too fast, which is not preferable. Another problem with asynchronous counters is that they are very prone to glitches, which is a problem for the lab since the QUARTUS software can't detect glitches unless the circuit is buffered, buffered circuits are not good as they are severely slowed down, ruining the efficiency of the circuit. The schematic (Figure 8) has 2 inputs and 7 outputs. The first input is a "Dip Switch Counter In" meaning that the input is tied to a DIP switch on the field programmable gate array board (FGPA), this is so that the tester can have random points to turn off the active high, and test if the circuit works when the DIP switch is changed. The second input is the clock input, this is pretty straight forward. The clock when having a high signal means that it won't let data through but will release stored data. When the clock has a low signal, it won't release data but can receive new data, and then store said new data. The outputs were "Counter Input Output", clock output, sync reset, and q0, q1, q2, q3. Counter input output is just the output of the DIP switch. The clock output does the same, showing the clock square wave output. The sync reset shows where the reset is done on the waveform, usually on the rising edge of the clock, this also shows on the waveform where the values count up to 10. Q0, q1, q2, and q3 are the outputs from each part of the up counter and get toggled at different clock cycles. Q0 is toggled every clock cycle, q1 is every 2 cycles, q2 is every 4 cycles, and q3 is every 8 cycles. This can generate different number values allowing the total outputs to count up to 10. The AND gate in the bottom right of the schematic (Figure 8) resets the counter every time the binary values count up to 10, this works by connecting the AND gate to the selector lines to each 2 to 1 multiplexer gate. This is signified since the inputs to the AND gate is q1 and q3, these flip flops are where the clock division cycles add up to 10. This is also where the sync output gets the signal from. The outputs q0, q1, q2, and q3 are BUS 'ed to their output because it is more efficient than creating 4 different outputs for each q value from every toggle flip flop The X accumulator is different than the MOD 11 counter because the desired circuit needed is a shift register/accumulator, not a counter. This was done by removing the AND gates, and having an actual input of bits, since the d flip flops are going to store said bits. The selector AND gate are also removed, with selectors having named inputs rather than certain wires to count in, also removing the NOT gates for the non-selector inputs. The reset for the d flip flops is going to be asynchronous rather than synchronous. The X accumulator is different than this modified MOD 11 counter in that it will need more inputs, and therefore will need 4 to 1 multiplexers instead of 2 to 1 multiplexers. The bit values come from "X" rather than "Q." Also, like the serial adder, the clock signal and reset(also may be referred to as "clear") are wired through NOT gates in order to be negative edge triggered rather than positive edge. The inputs for the 4 to one mux gates are as follows(visualized in Figure 10, Figure 11); D0 is a bit from the "X" input, with the least significant bit on the left, with most significant bit on the right, D1 is the "si" signal for the first mux which is the sum of the serial adder, then it is the output of the d flip flop before the MUX, D2 & D3 are the following output of the D flip

flop, S1 is the “done” signal, and S0 is the “sh” or shift signal. The final output of from the last d flip flop is then used as the output Xi which then gets shifted into the serial adder to be added or subtracted depending on the sub signal. The schematic for the X accumulator is split into 2 images for the purpose of clarity in the circuit. Each bit of X gets their own output into a BUS ’ed output of Xout[4..0], this is so that testing for the accumulator becomes easier, it also makes troubleshooting easier, for example if the X accumulator is having errors when loading. Another problem could be when the X accumulator is shifting values even though the shift value is equal to zero. The Y register is similar to the X register, but with some key differences. The first difference is the “Si” input, it may look the same within the Y register circuit(Figure 12,Figure 13,Figure 14), however the Si input is the output Yi as opposed to the sum of the serial adder, this can be seen in the final bit serial adder subtractor schematic (Figure 2). Another difference is the Y register decides whether the bit serial adder subtractor is going to use the addition operation or subtraction operation by using the 4 XOR gates and the “sub” input from the control finite state machine to create a 2’s complement number which is then shifted into the serial adder. Other than the Y inputs going through XOR gates, the rest of the inputs stay the same, same thing with the outputs, though the letters are changed from X to Y. In short, most of the circuit is derived from the X accumulator, with some differences to change its function. The functions of both the Y register and the X accumulator can be seen in a truth table (

Table 5). It is very important that when designing both the Y and X accumulators that the most significant bit output will be at the end of the circuit, meaning that the output of the last d flip flop should be X[0] or Y[0], and the left most d flip flop output should be X[4] or Y[4], this error was not caught until both the X accumulator and Y register were tested in the university waveform program simulator. After the X accumulator and Y register are tested, both of the schematics are turned into a block symbols. The final piece of the whole circuit and the most challenging part was the finite state machine. The challenge was not testing the circuit but designing the circuit and coming up with the equations for the different d flip flops for each Y value. The type of state machine used for the bit serial adder subtractor was a Moore finite state machine that is grey coded. The circuit is derived from a state diagram (Figure 16). The state diagram is a visual indication of the sequences and which sequence will give the outputs “sh”, “done”, and “sub”. The diagram is then used to draw a state assignment table. The state assignment tables (Table 1) will show the current state and will show the state that will come after when the conditions for n=0 or n=1 is met. If the desired sequence is made, then the outputs will have certain values assigned depending on whether n=1 or n=0, however some states will happen regardless of if n=0 or n=1. For example, in state B, when the “Nadd” input is 1, then it will start outputting the “sh” output to 1 for 5 states, after which “sh” will become 0 and “done” will output 1 .From the state assignment table, Karnaugh maps can then be created, 5 K maps are made for the grey coded circuit, one for Y_3, $Y2$, $Y1$, “sh”, and “done” this will create the equations that will determine the next states for each variable, and the output. The K maps are good because they create a simplified Boolean expression for each Y value, and for the outputs. The equation for sub was already developed as $sub = \overline{y_3 * y_2 * y_1} * nADD$. The state machine was done in grey code because grey coding the control finite state machine would significantly simplify the design process of the finite state machine and make the circuit much more simpler than if the finite state machine was coded in binary. A Moore state diagram was used as opposed

to a mealy state machine, this is due to having the most knowledge on hand with the Moore state machine. The mealy state machine is very good in that the outputs would be a function of the present state and an input at the same time, this means that the mealy finite state machine would line up with the detection waveform exactly, however the mealy finite state machine is prone to glitches, the control finite state machine desired for the bit serial adder subtractor circuit must be a stable one, even though it would be one clock cycle behind detection. After the state machine is done being designed and tested it is turned into a block symbol. After the following circuits are tested, they are wired together with some additional logic gates (Figure 2). The whole circuit as a whole will have 6 outputs, and 5 outputs. Some notable features of the circuit are that the clock input for the serial adder is the “sh” input from the finite state machine and the clock input is put through an AND gate and that becomes the clock input for the serial adder. An AND gate simple means that both inputs are being multiplied together, meaning that if at any point one of the inputs is 0, then the output is going to be 0 as well, but if both inputs are 1 then the output will return 1, this is the fundamental part of the majority of the circuits, and more complicated sequences made in circuits and digital logic as a whole, a truth table for an AND gate can be seen in Table 8. The sub input from the finite state machine becomes the preset signal for the serial adder, though the preset signal is labeled “sub” to make wiring easier. After the block schematic for the bit serial adder subtractor are done, then the whole circuit is then made into one symbol (Figure 1), this is done so that the whole bit serial adder subtractor circuit can be properly simulated in the Model Sim software without errors.

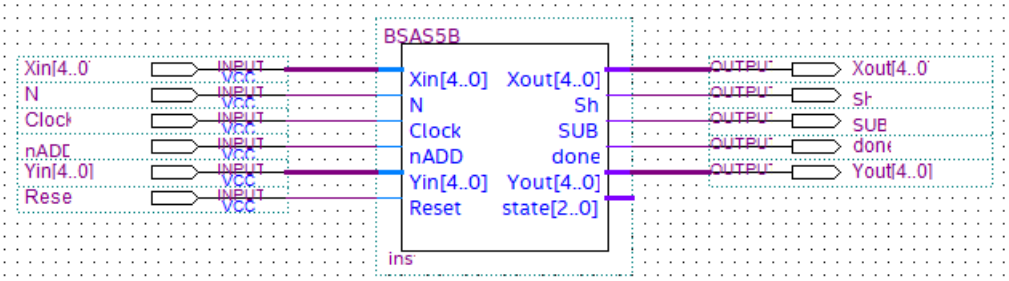

Figure 1 The top most level schematic, which is a symbol of the entire bit serial adder subtractor circuit

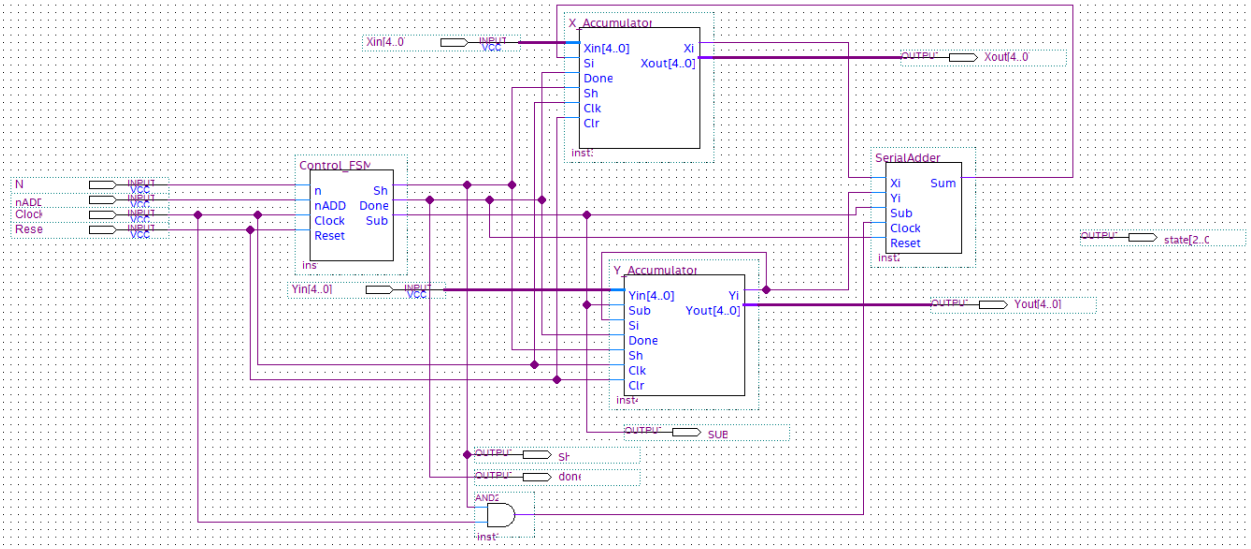

Figure 2 A block level circuit diagram of the entire bit serial adder subtractor circuit, the inputs are N, nADD, Clock, Reset, Xin[4..0], Yin[4..0], and the outputs are Yout[4..0], Xout[4..0], SUB, Sh, Done. The "done" output is used to check if the bit serial adder subtractor circuit is working by comparing it to the "equal" waveform in the Model Sim waveform software. The sub, and sh outputs are included as well to check if those waveforms are functioning as they are made to.

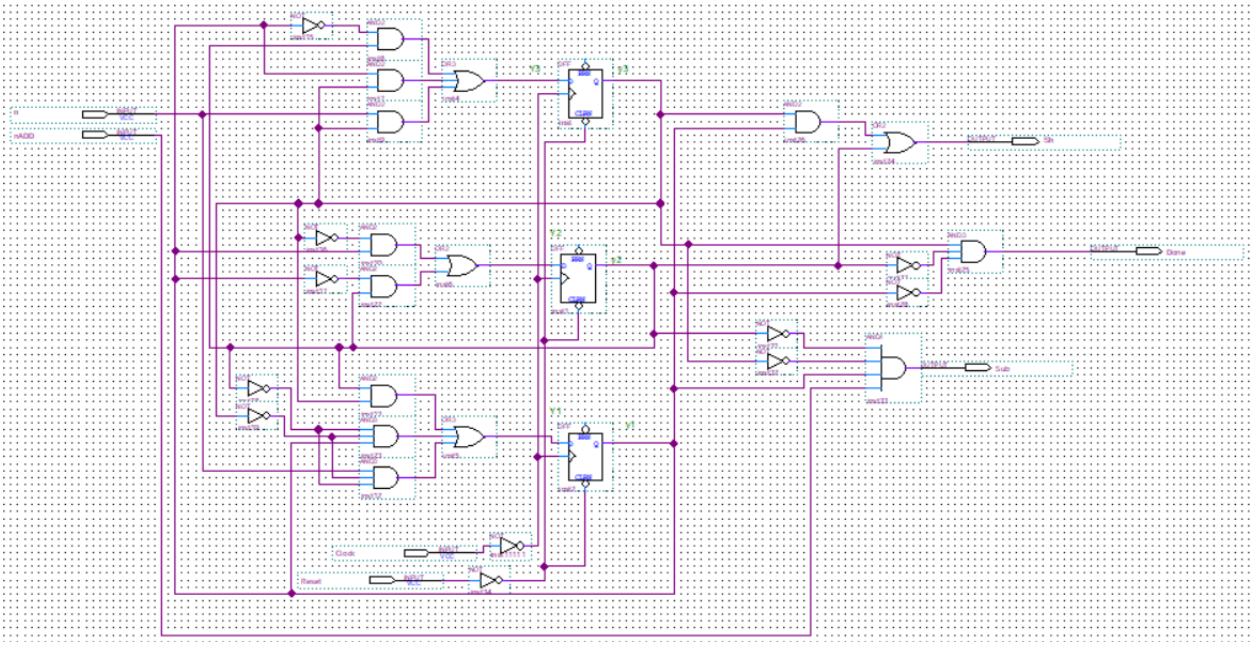

Figure 3 The whole circuit for the Moore grey coded finite state machine, the Y values for each d flip flop are labeled with the green text, each equation is matched with the k map that its associated with.

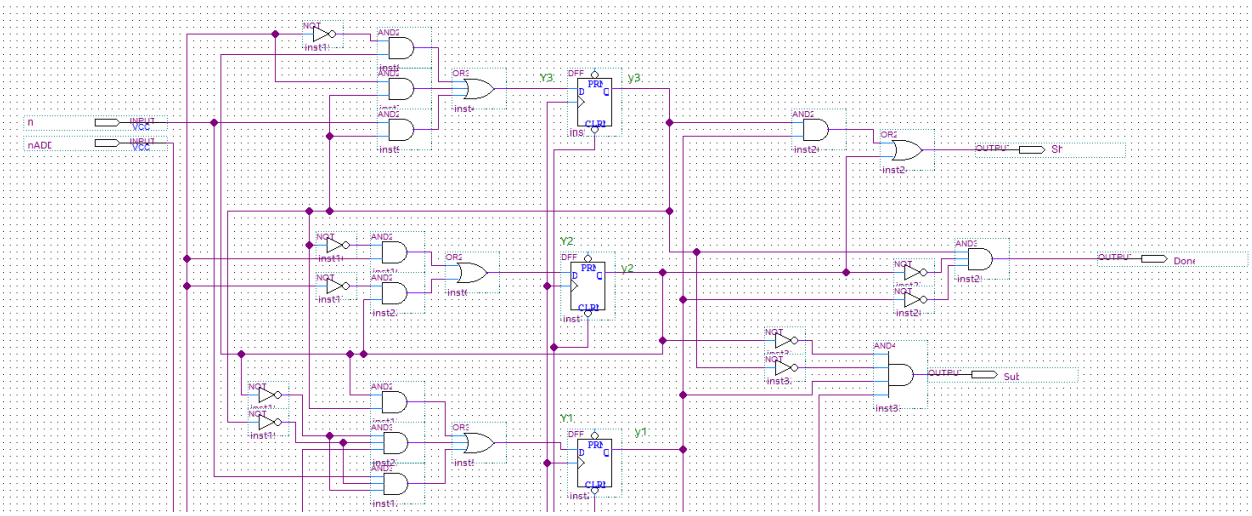

Figure 4 The top part of the finite state machine, split for clarity. Showing the circuit for each DFF

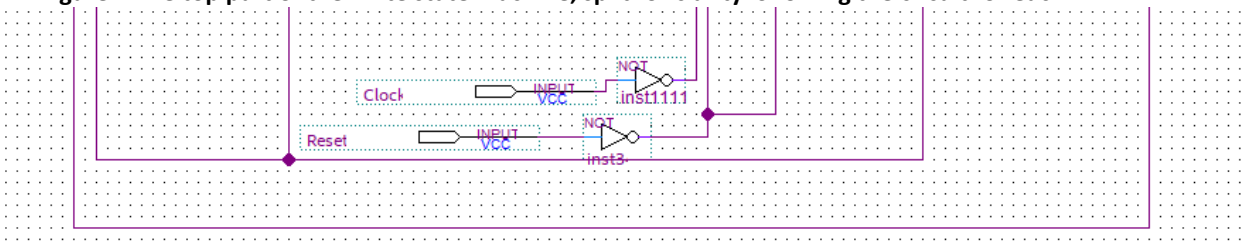

Figure 5 The bottom part of the finite state machine, split for clarity, displaying the clock and reset inputs and showing the wiring throught the NOT gates. The clock and reset are sent through NOT gates to make both inputs negative edge triggered

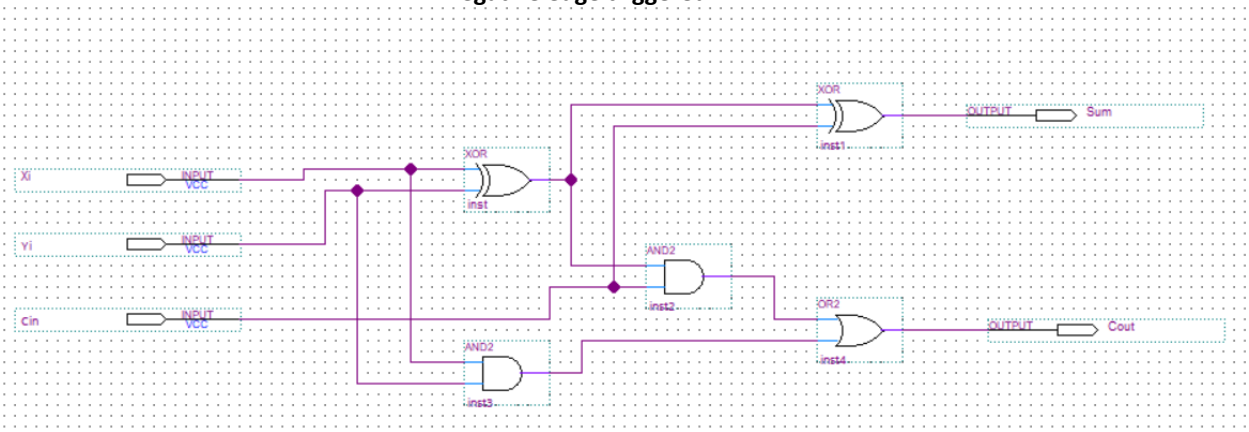

Figure 6 The schematic for a full adder, this is then changed into a symbol and used into the serial adder.

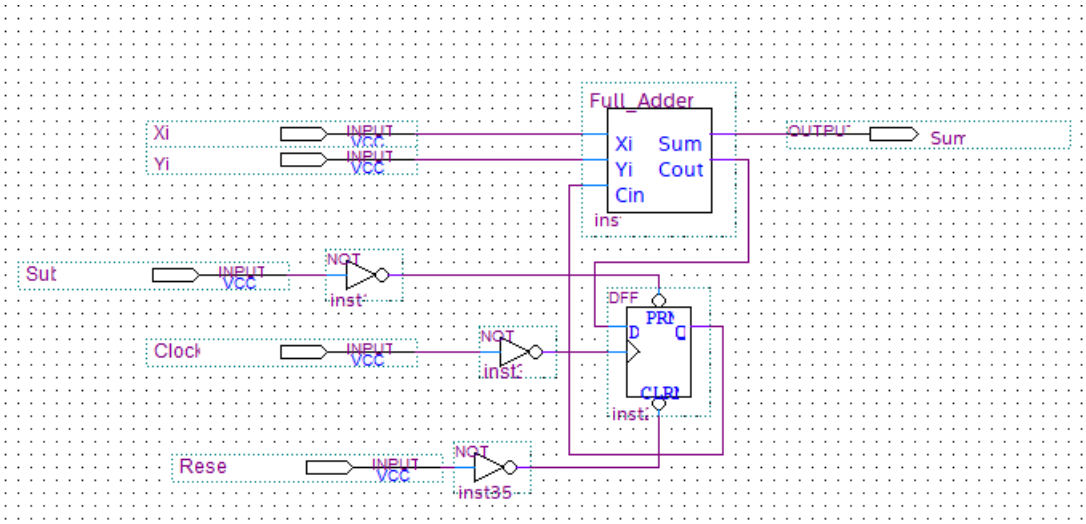

Figure 7 This is the serial adder schematic, it possesses a negative edge triggered clock, with a positive triggered reset and preset.

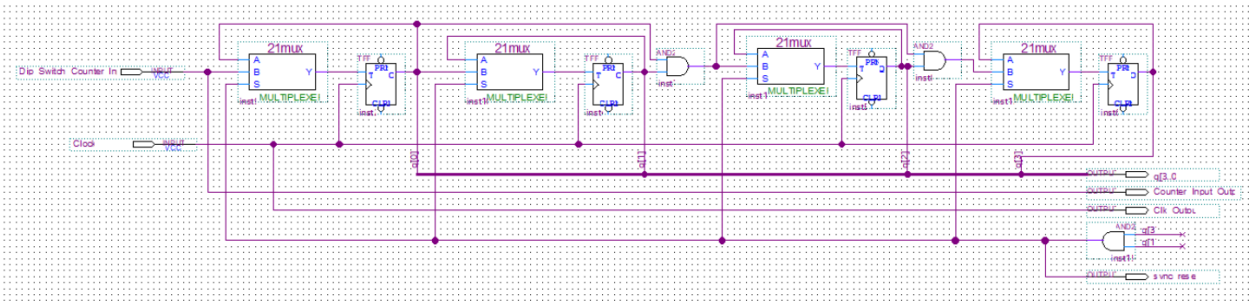

Figure 8 MOD 11 circuit that the X accumulator design and Y register design were initially based off of.

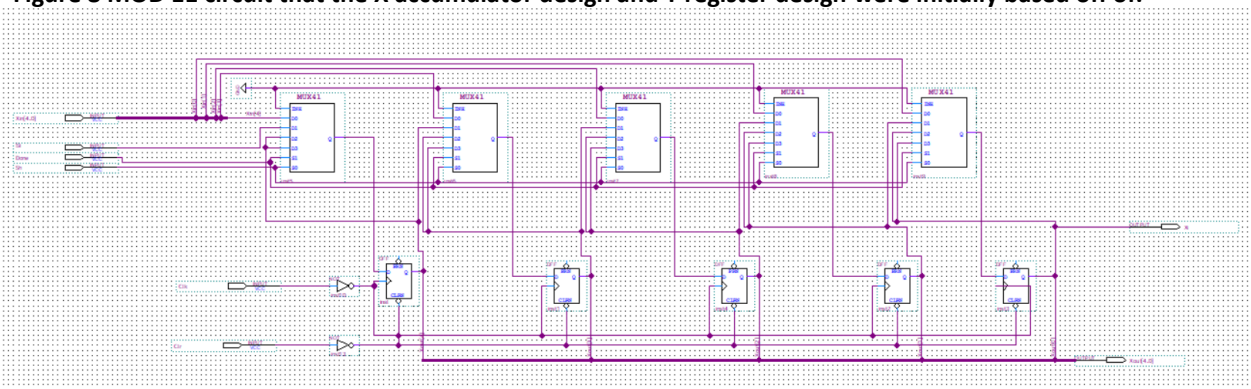

Figure 9 Full schematic of the X accumulator.

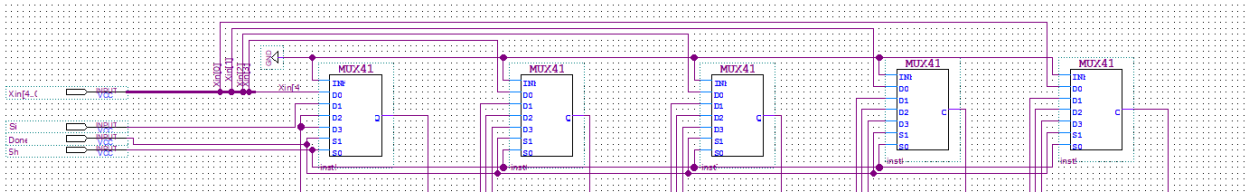

Figure 10 Top half of the X accumulator schematic, split for clarity, this part shows the MUX gates and how the inputs are wired to the mux gates.

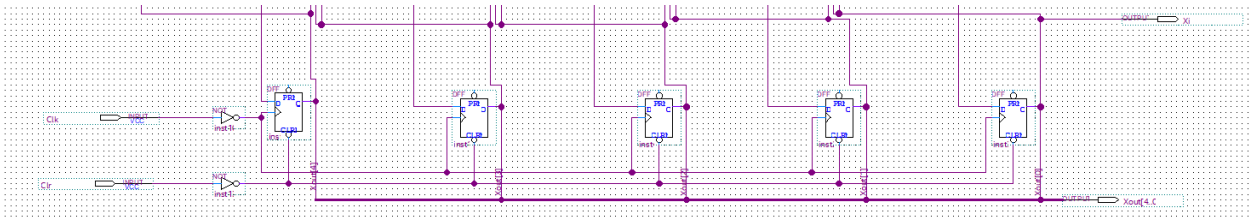

Figure 11 Bottom half of the X accumulator circuit, showing the D flip flops and their outputs, and showing the clock and reset signals with their inverted signals. The clock and reset are sent through NOT gates to make both inputs negative edge triggered

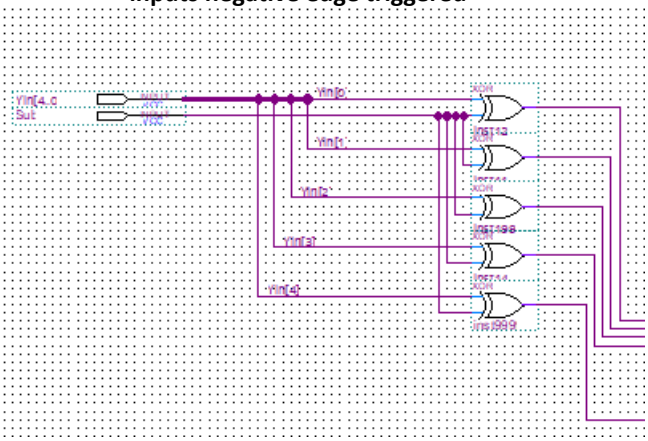

Figure 12 Top left portion of the Y register, showing the Y inputs being XOR 'ed with the sub input

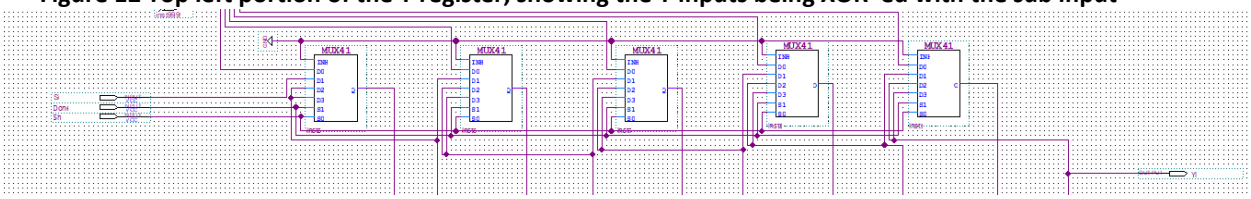

Figure 13 Middle part of the Y register, showing the MUX gates and how the si, done, and shift signals are wired into the MUX gates

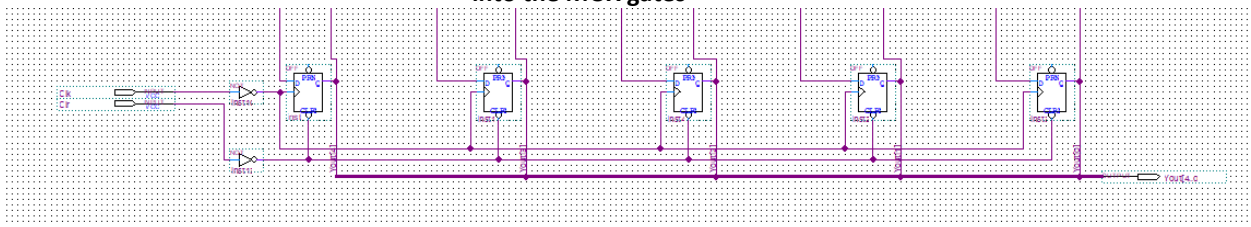

Figure 14 Bottom part of the circuit, showing how the outputs are wired and the clock and reset signals being inverted. The clock and reset are sent through NOT gates to make both inputs negative edge triggered

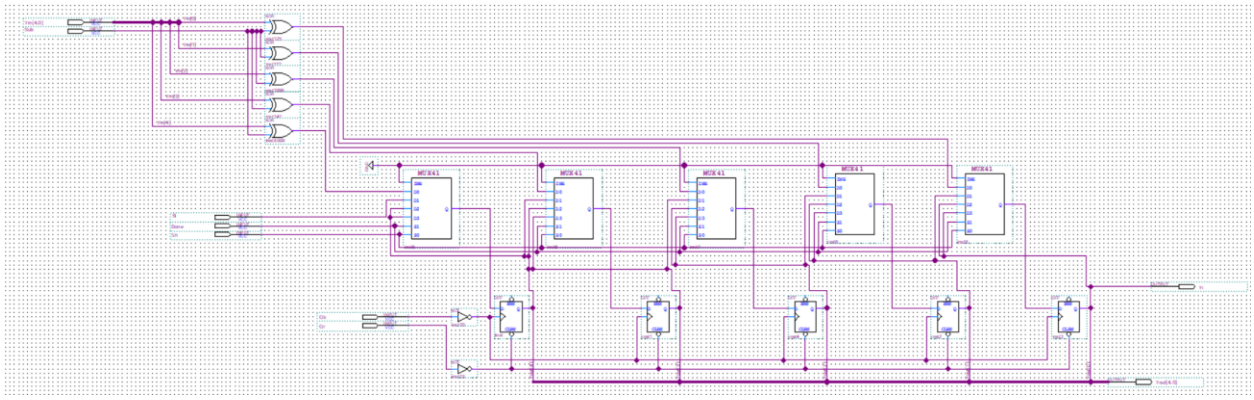

Figure 15 Entire Circuit of the Y accumulator

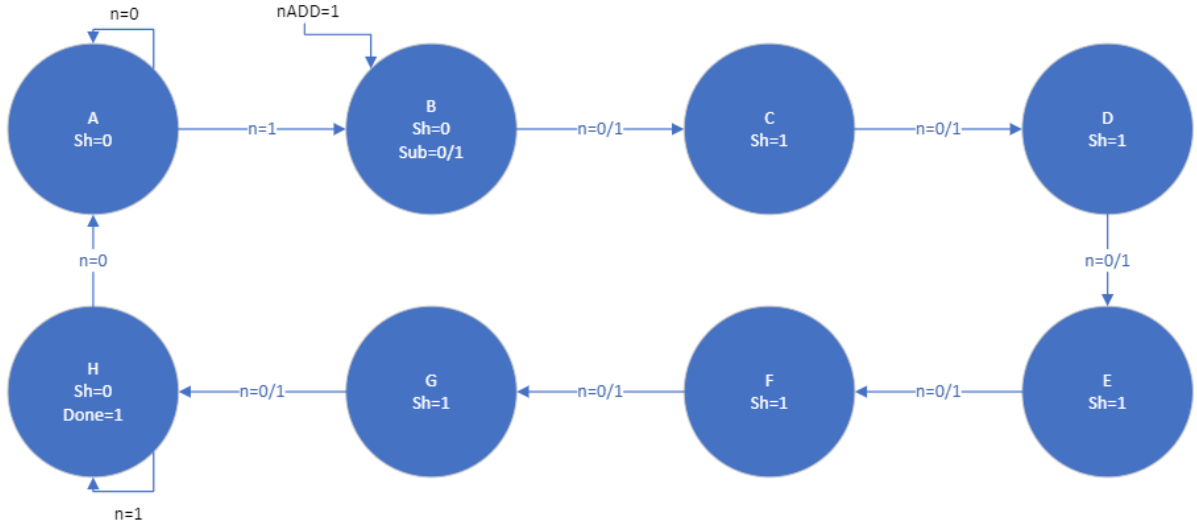

Figure 16 State Diagram for the Control finite state machine. Each state can be thought of as a clock cycle. The finite state machine starts with state A, meaning that sh=0 and so long as n=0 then state A will remain for the clock cycle, once n=1 however, then the finite state machine moves on to state B. State B will start, sub will can be 0 or 1, but once nADD is one then it will move onto the next state. State C is when shift will become 1, then for the next 4 states, sh will be 1 regardless of whether n=1 or n=0. Since sh=1 then the accumulators will start shifting their binary values, then state H is reached after the 5 cycles where sh=1. The state drawing diagram was drawn using the Microsoft Visio software.

Results and Discussion

During the design process, each part of the bit serial adder subtractor circuit was tested using a timing simulation waveform, using the university waveform tool in the QUARTUS software. The timing simulations are a form of simulation that emulates the real world, specifically it emulates the propagation delay that would exist only in physical circuits, versus a theoretical one. The timing simulation for the control finite state machine (Figure 17 shows that once the "n" input goes high then the sh input will go active high for 5 clock cycles, with sub going high for one cycle right before sh goes high, and showing that the done input will equal 1 after sh goes back to 0, this perfectly visualizes what was shown in the state diagram for the finite state machine (Figure 16), this was set up by having the clock cycles reset every 40 nano seconds, with an asynchronous reset at a point in the beginning, with n being high for 1 and a half clock cycles, though it does not need to be that exact. After the finite state

machine is tested to work, then the design and simulation process for the X accumulator and Y register starts. The Timing simulations for the X and Y accumulator show that each of the accumulator and register works as intended showing the shifting values for all the binary numbers. For the X accumulator and the Y register, the clock was set to 40 nanoseconds for each clock cycle, this makes keeping track of clock cycles easier due to the cycles lining up with the vertical dotted lines in the university waveform program perfectly. The reset is asynchronous, so it can be high for a short timespan in the beginning of the circuit. The “sh” and “done” inputs for both the X accumulator and Y register should emulate the output waveforms from the finite state machine (Figure 17) where sh is high for 5 clock cycles, with done high for 1 clock cycle. The “si” input can be high at an arbitrary point in the waveform, though it is recommended to set it to test all scenarios of “sh” and “done”, meaning when either one is high and one isn’t, or when both are high or low, this enables that the whole accumulator or adder will function through all scenarios. After, the serial adder is tested. The serial adder is tested by setting inputs for Yi and Xi as said inputs are going to be added and subtracted together simultaneously (Figure 18) , it is preferable to set Xi and Yi as consistent inputs throughout the waveform as it can be easier to see if the serial adder is adding and subtracting as intended. After the serial adder is designed and simulated, each part: Control finite state machine, X accumulator, Y register, and serial adder are all turned into symbols and wired to make a bit serial adder subtractor circuit, which is then made into 1 final block with only inputs and outputs on the block diagram schematic. After this final design step, it is all testing for the bit serial adder subtractor. The simulation of the final circuit is done in model sim and can be seen in Figure 21. Model sim is a superior waveform tool compared to the university waveform tools because of 2 features, the existence of color coding the waveforms make troubleshooting significantly easier to do, and the existence of the “equal” wave is used to test if the circuit should be operating correctly. For example, while working on the Model sim waveform simulation, there were many errors found in the circuit when testing, and due to the detection on Model Sim, some problems were easily fixed. In one instance, the “sub” and “done” output wires somehow became bridged, and went undetected, though thanks to Model Sim showing that the sub and done waves were equal, this problem was then fixed. At an earlier point in testing, most of the waveforms were red, this indicated that something was creating errors from the serial adder, as the other parts were tested to work, it was then realized that the serial adder did not have the preset triggered on a negative edge, therefore creating multiple errors. Model Sim was able to contribute to the troubleshooting process significantly. This lab properly showed how to utilize model sim to the fullest.

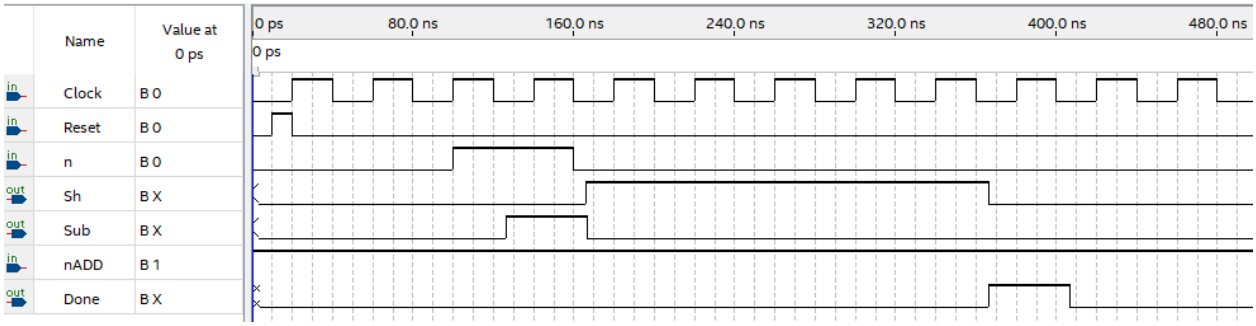

Figure 17 Timing simulation for the finite state machine. This waveform shows the finite state machine working perfectly as intended, as n=1, sh will go high for 5 clock cycles, with sub going high for 1 clock cycle right before sh goes high and showing that the done signal outputs 1 for one clock cycle as intended from the state diagram drawing.

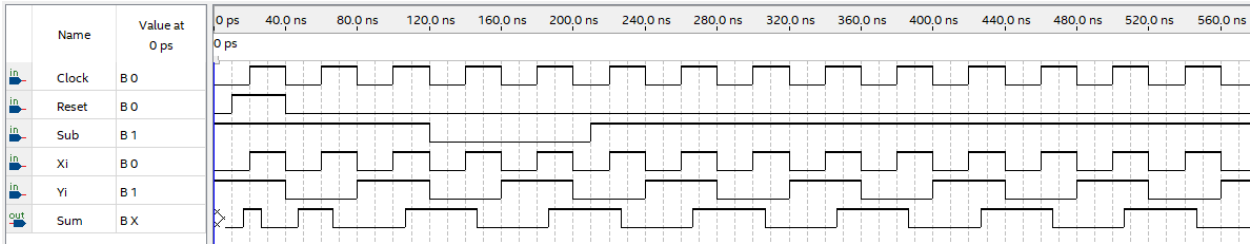

Figure 18 Serial Adder timing simulation.

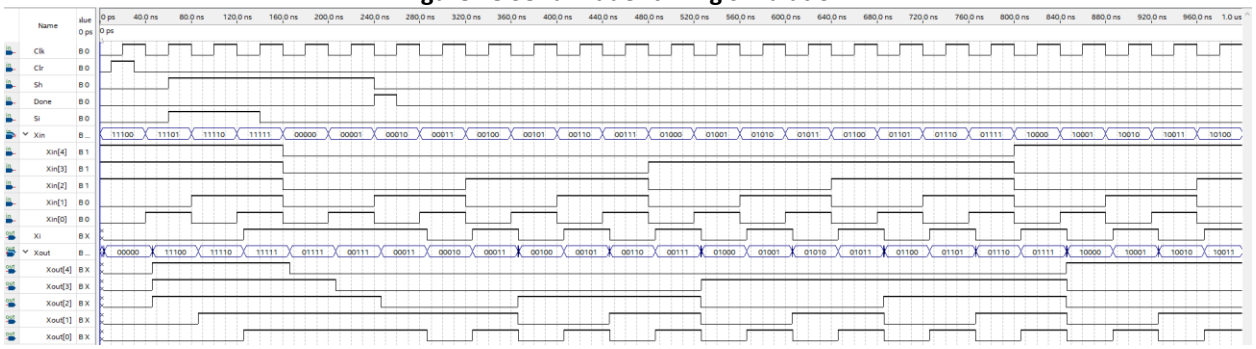

Figure 19 X Accumulator Timing simulation, it can be seen that the binary values are being shifted over and held at certain positions throughout the entire waveform.

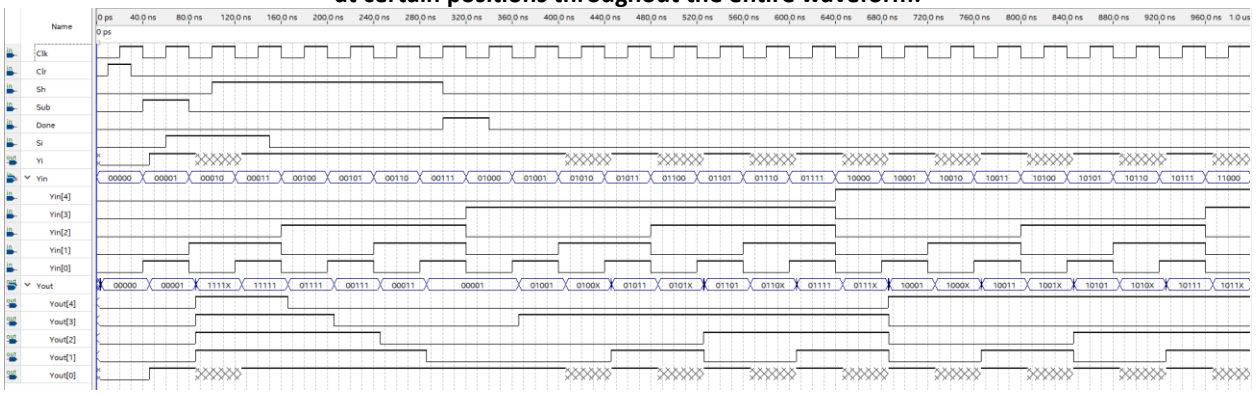

Figure 20 Y Register Timing simulation, it may be hard to tell, but the binary values are being loaded, and shifted throughout the waveform, and they are being held as well.

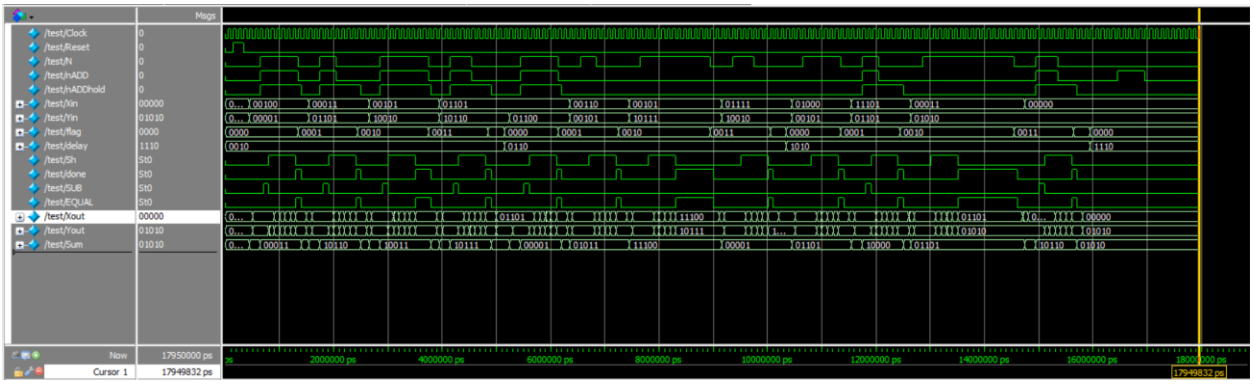

Figure 21 Model Sim waveform, displaying the bit serial adder subtractor circuit functioning as intended.

Conclusions

The nature of the lab assignment was to learn to create a complicated circuit with multiple advanced logic functions within the circuit. Another objective of the assignment is to learn to use the Model Sim software as it can be very beneficial to learn in the future. Like other labs, there was a theoretical approach taken, but there was no practical approach taken after. First a serial adder was drawn (Figure 7) and then simulated to check if it functioned properly (Figure 18). After the serial adder was finished, then the X accumulator and register was conceptualized, first a truth table was drawn to determine what the accumulator or register would do when the selector line values “done” and “shift” were either 1 or 0. This can be visualized in

Table 5. After, the X accumulator was drawn using the MOD 11 counter (Figure 8) as a reference point, changing the necessary schematic components to change the circuit from a 4 bit counter to a proper shifting data register, meaning that selector lines were changed to inputs, the flip flops changed from toggle flip flops to D flip flops, and the mux gates changed to 4 to 1 MUXes, these mux gates are then wired to “done”, “si”, “shift”, and the outputs of the following and previous d flip flops for the rest of the mux gates. The X accumulator was then tested using the university waveform simulation tools. After the X accumulator gate is troubleshooted and completed, the Y register (Figure 15) is then designed. The design of the Y register is derived from the X accumulator, with some key differences, there is a “sub” input added, this is because the inputs for each MUX gate from Y is XOR ‘ed, this enables the use of subtraction in the entire circuit. The sub signal controls whether the number for the Y register is going to be positive or negative, determining the operand used in the serial adder. After the Y register is then designed, tested, and troubleshooted, then the large task of making the control finite state machine is started. The first thing that is done when designing the finite state machine is deciding which type of state machine is going to be used (Moore vs Mealy) and which type of coding is going to be used (binary, grey, or one hot). It was decided that the finite state machine would be a Moore finite state machine that would be grey coded. As stated previously this was done to ensure that the circuit would be as simple as possible, and be resistant to glitching, as mealy is prone to glitching due to its speed difference. After the type of finite state machine is chosen, then a state diagram is drawn (Figure 16) to determine how it would function and what conditions would be met to change between the states. After the state diagram is drawn, then a state assignment table is drawn (Table 1) to visualize the present state and the next state, this allows for the synthesis of the equations via Karnaugh maps, not to mention it shows when each output is going

to be 1, which will help for waveform testing. After the state assignment table is drawn, then 5 Karnaugh maps are created for the following outputs: Y3 (Table 2), Y2 (

Table 3), Y1 (Table 4), Sh (

Table 6), and Done (Table 7). After the K maps are drawn, then equations are derived from each one. There are 5 in total: $\boldsymbol{Y_3 = y_2\,\overline{y_1} + y_3\,y_1 + n\,y_3}$, $\boldsymbol{Y_2 = \overline{y_3}\,y_1 + y_2\,\overline{y_1}}$, $\boldsymbol{Y_1 = \overline{y_3\,y_2}\,y_1 + y_3\,y_2 + n\,\overline{y_3\,y_2}}$, $\boldsymbol{sh = y_2 + y_3\,y_1}$, $\boldsymbol{done = y_3\,\overline{y_2\;\;y_1}}$. The equation for sub is developed as $\boldsymbol{sub = \overline{y_3 * y_2 * y_1} * nADD}$. After then the schematic for the control finite state machine is drawn (Figure 3). Then the schematic is tested in a simulation, set up with nADD as active high for the entirety of the simulation, as nADD being high is necessary to trigger the events that would enable the shift signal to be high for 5 consecutive clock cycles. The clock being set to 40 nano seconds, and the asynchronous reset being high for a show period in the beginning. The sub output should be 1 for one clock cycle, and the done signal should be 1 for one clock cycle, if both these conditions are met then the finite state machine is correct. Then, after the finite state machine is tested to be correct, then all the parts are re tested and are troubleshooted if any mistakes were not detected before, this ensures debugging of the final schematic will be simpler and less convoluted. Finally, all four core parts of the bit serial adder subtractor are then wired together, they are wired together according to the specifications of the lab assignment PDF. After the circuit is put together, it is compiled to check for small errors, then one big block symbol is made for the bit serial adder subtractor, this enables it to work in the Model Sim software. Then it is run through the Model Sim software that comes packaged with QUARTUS. Then the circuit is debugged if there is anything wrong with the waveforms. This is a tedious process that can take many hours. However, the process is made easier due to Model Sim being a very good software, with the color coded waveforms, and the equal outputs making debugging much easier. The theory and experiment lined up perfectly, this was predicted as if both the theory and the experimental portions of the lab assignment did not line up, the bit serial adder subtractor circuit would simply just not work. The theory of the of the finite state machine lined up perfectly with the experimental outcome, the waveforms were the same as the predicted outcomes that were theorized in the state diagram drawing and the state assignment table. However, the QUARTUS software made testing the experimental data very difficult, this is due to the QUARTUS software not being well optimized for modern computer technology, during the testing periods the university waveforms would frequently crash the entire Quartus software, not to mention even with very exceptionally good hardware, compilation times are exceptionally slow. There are even some problems with model Sim, for example with model Sim one cannot have 2 model Sim waveform simulations open at the same time. The model sim software also cannot work if the regular bit serial adder circuit is set as the top level, it needs the full block as the top level in order to simulate properly. All in all, the lab assignment of building a bit serial adder subtractor is a beneficial one as it teaches how to design multiple advanced components, and use the components in tandem to create one large, advanced circuit that can perform complicated or consecutive operations.

Appendix

Table 1 The state assignment table for the Moore grey coded control finite state machine.

Present State	Next State		Output		
$y_3y_2y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	Sh	Done	Sub
	n = 0	n = 1	0	0	0
A 000	000	001	0	0	1
B 001	011	011	1	0	0
C 011	010	010	1	0	0
D 010	110	110	1	0	0
E 110	111	111	1	0	0
F 111	101	101	1	0	0
G 101	100	100	1	0	0
H 100	000	100	0	1	0

Table 2 This is the Karnaugh map for the value of Y_3, the equation is $Y_3 = y_2\overline{y_1} + y_3y_1 + ny_3$

Y_3		$y_2\ y_1$			
		00	01	11	10
$n\ y_3$	00	0	0	0	1
	01	0	1	1	1
	11	1	1.	1.	1.
	10	0	0	0	1

Table 3 This is the Karnaugh map for the value of Y_2, the equation is $Y_2 = \overline{y_3}y_1 + y_2\overline{y_1}$

Y_2		$y_2\ y_1$			
		00	01	11	10
$n\ y_3$	00	0	1	1	1
	01	0	0	0	1
	11	0	0	0	1
	10	0	1	1	1

Table 4 This is the Karnaugh map for the value of Y_1, the equation is $Y_1 = \overline{y_3y_2}y_1 + y_3y_2 + n\overline{y_3}\overline{y_2}$

Y_1		$y_2\ y_1$			
		00	01	11	10
$n\ y_3$	00	0	1	0	0
	01	0	0	1	1
	11	0	0	1	1
	10	1	1	0	0

Table 5 This is the truth table for both accumulators regarding their behaviors with bit storage. If done and shift(also known as "sh") are both 0, then they will load the value stored into the serial adder, if done is 0 and sh is 1, then the X accumulator or the Y register is then going to shift the bits, if done is 1 and shift is 0, or if both shift and done are 1, then the binary values are going to be held in place inside the register.

Done	Shift	Output
0	0	Load
0	1	Shift
1	0	Hold
1	1	Hold

Table 6 The Karnaugh map for the Sh output of the finite state machine, the equation is $sh = y_2 + y_3 y_1$

Sh					
		$y_2 y_1$			
		00	01	11	10
y_3	0	0	0	1	1
	1	0	1	1	1

Table 7 The Karnaugh map for the done output of the finite state machine, the equation is $done = y_3 \overline{y_2\ y_1}$

Done					
		$y_2 y_1$			
		00	01	11	10
y_3	0	0	0	0	0
	1	1	0	0	0

Table 8 This is the truth table for an AND logic function. The equation is $f = X_1 * X_2$

X1	X2	f
0	0	0
0	1	0
1	0	0
1	1	1

References

[1] G. Mutyala, *"gsm9263_ee120_Lab5,"* Mar. 2023.
[2] G. Mutyala, *"gsm9263_ee120_Lab9Report,"* Apr. 2023.
[3] G. Mutyala, *"gsm9263_ee120_Lab10,"* Apr. 2023.