Introduction:

The lab required to design a Full Calculator via system integration. The lab was a combinational lab which included pipeline-able integer multiplier (lab 5), small calculator (lab 7), and integer divider (lab 8) as subsystems. The goal of the lab was to implement addition, subtraction, multiplication, division, increment, decrement, and square. To operate those functions, a top-level Datapath (DP), and a top-level Control Unit (CU) were required to design. The top-level DP design, different from previous lab that only connected blocks, connected the both sub-CU and sub-DP. The top-level CU controlled the top-level DP via en_f, en_x, en_y, go_cal, op_cal, go_div, sel_H, sel_L, en_out_H, and en_out_L. Meanwhile, the CU would receive two flags: Done_Calc and Done_DIV from the top-level DP.

Design Methodology:

FullCalc DP:

The top-level DP contained three inputs: X, Y, and F. All the three inputs were controlled by CU via en_f, en_x, en_y. Then, the DP would execute the following operations on 4-bit operands X and Y as show in Table 1.

Opcode	Functions
000	Addition: X+Y
001	Subtraction: X -Y
010	Multiplication: X*Y
011	Division: X / Y
100	Increment X + 1
101	Decrement X-1
110	Square X^2
111	Not Defined

Table 1: DP Operations

Lab 7 built a small calculator which can do addition and subtraction, Lab 5 had a pipeline module integer multiplier, and Lab 8 has integer divider. Thus, there are three blocks named **Calc**, **MUL**, and **DIV** were created. Since **Calc** and **DIV** were FSM modules, each of them has sub-control unit, and should return a flag **Done**. **Calc** would return a **Done_Calc** and **DIV** would return a **Done_DIV** to the top level CU. **MUL** was a parallel pipeline design and it does not a clock to control it. However, the product of two 4-bit operands was 8-bit. Thus, there should be extra design for **MUL**. The complete design could be review from Lab instruction, however, there were two extra blocks were added for increment and decrement functions. Thus, the new design can be shown in Figure 1.

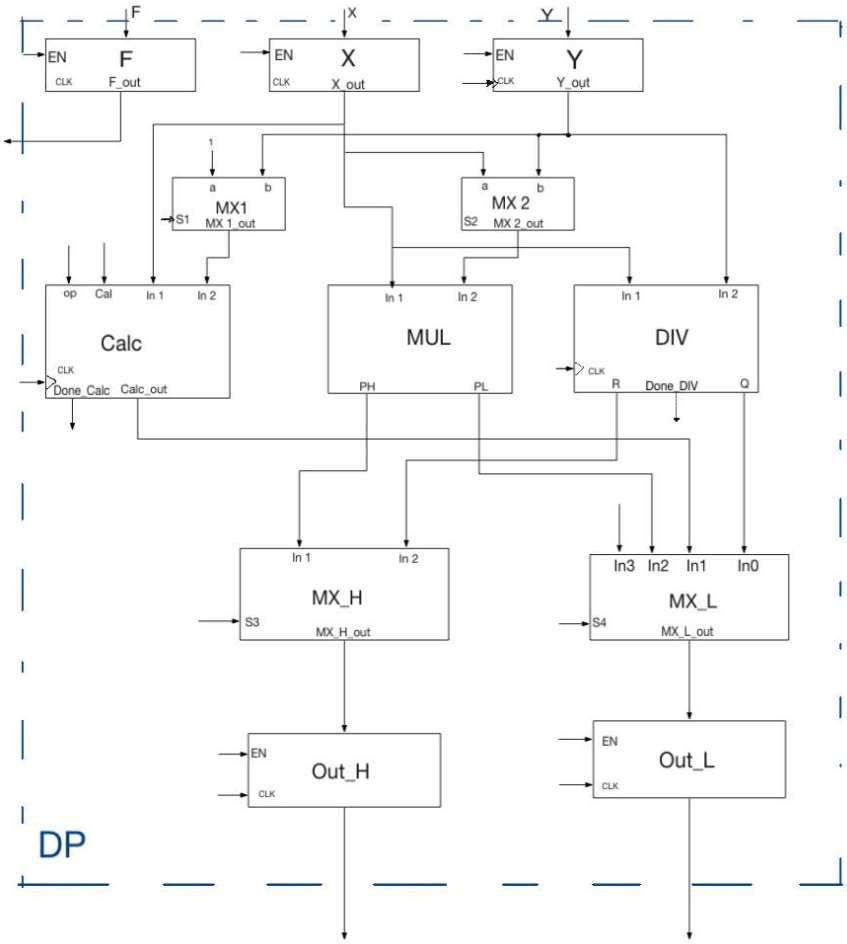

Figure 1: DP Structure Based on Description Above.

FullCalc CU:

The top-level CU controlled the top-level DP. Based on Figure 1. The whole CU system was divided into 18 states like Figure 2 showed:

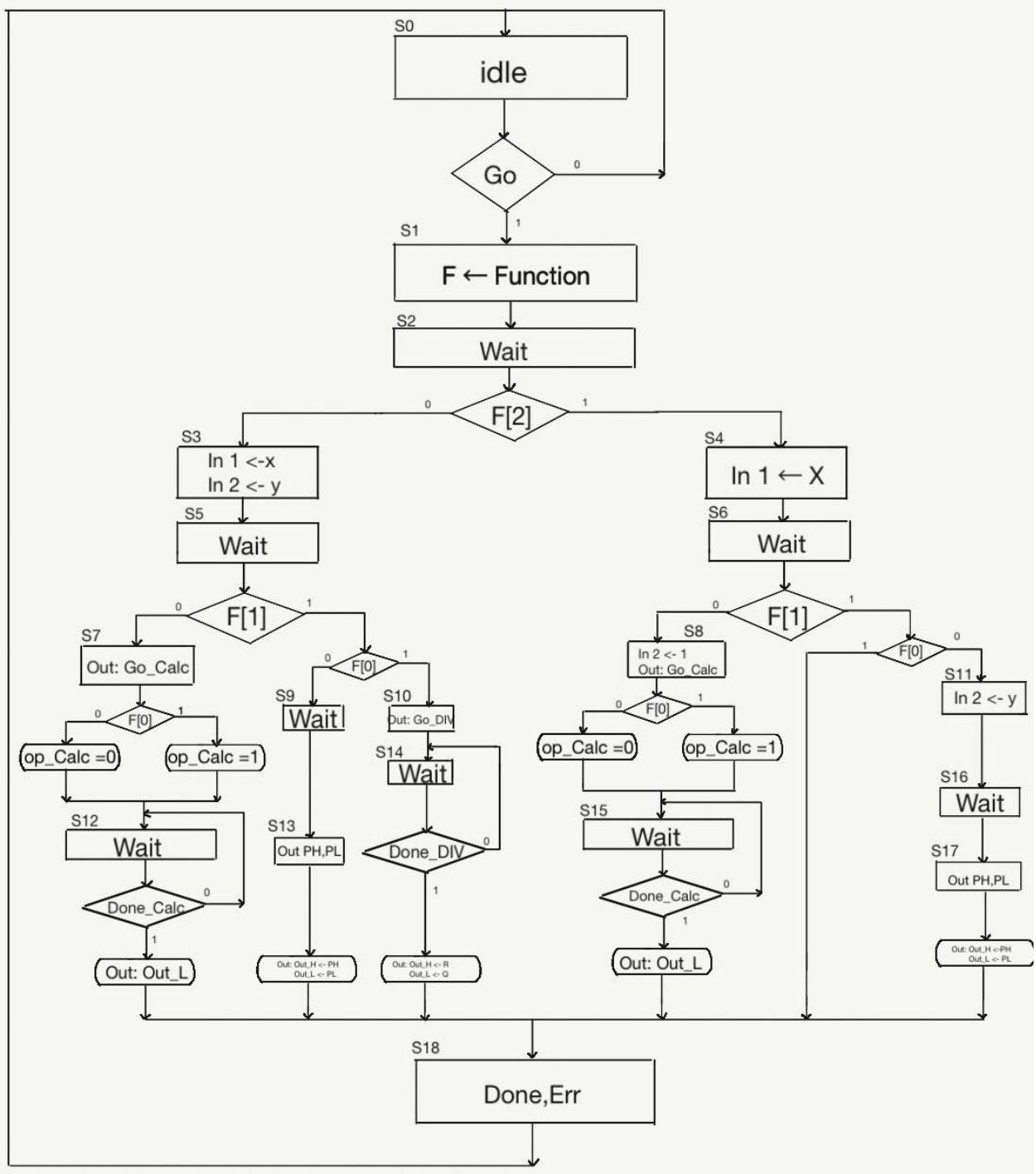

Figure 2: ASM Chart for Top-Level Control

As Figure 2 showed, when the system stayed in S0, it was idle. At the state, CU was waiting for *GO* signal activated. Once the *Go* turned to 1, the system soon went to S1. At S1, function would load through F and then passed through *Op_Calc* to DP. Then, at state 2, while the system was waiting, it detected the MSB of function F. if the MSB is 0, the system would go to S3, otherwise, it would go to S4. Using Table 1 as reference, the following bits of F would lead the

system to either do addition or other operations. Meanwhile, as mentioned in DP section, there were two signals would return from DP to CU, which were **Done_Calc** and **Done_DIV**. For state 12, 14, and 15, which were operating small calc operations and integer division, need to wait these two return signals. When one of them was turned to 1, it meant the operation was done. Then, the outcome could be output from DP by using controlling signal **En_out_H** and **En_out_L**. The state transfer diagram was shown in Figure 3, and truth table of CU was shown in Table 2:

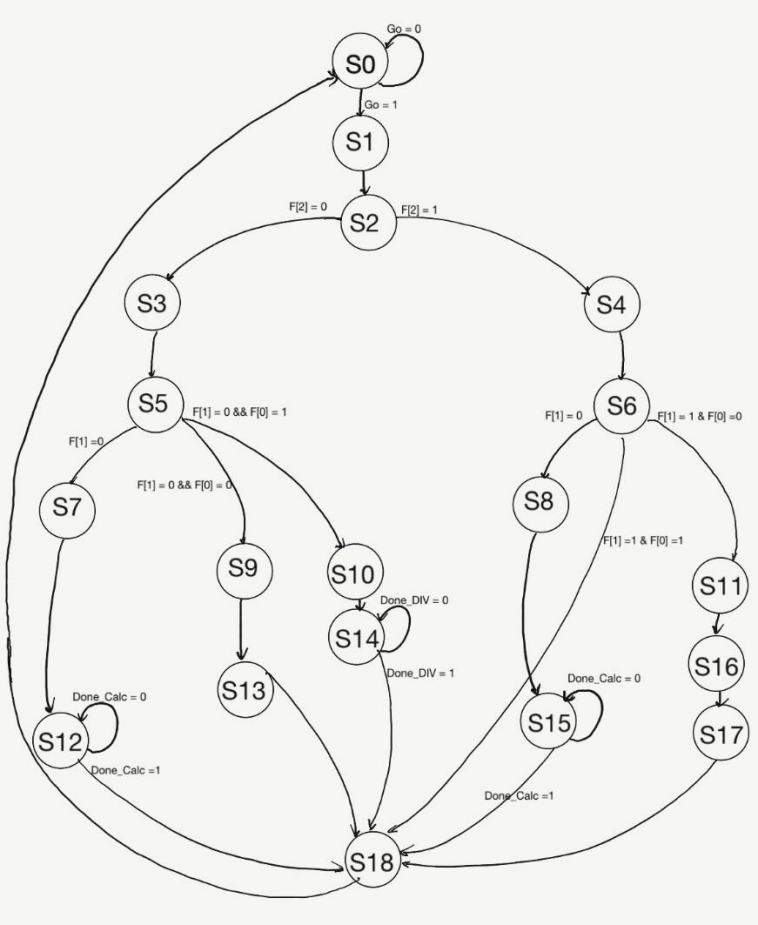

Figure 3: State Transition Diagrams

	rst_out	en_f	en_x	en_y	sel1	sel2	sel_H	sel_L	go_cal	go_div	en_h	en_l	done
s0	1	0	0	0	0	0	0	0	0	0	0	0	0
s1	0	1	0	0	0	0	0	0	0	0	0	0	0
s2	0	1	0	0	0	0	0	0	0	0	0	0	0
s3	0	1	1	1	1	0	0	0	0	0	0	0	0
s4	0	1	1	0	0	0	0	0	0	0	0	0	0
s5	0	1	1	1	1	0	0	0	0	0	0	0	0
s6	0	1	1	0	0	0	0	0	0	0	0	0	0
s7	0	1	1	1	1	0	0	10	1	0	0	1	0
s8	0	1	1	1	0	0	0	0	1	0	1	1	0
s9	0	1	1	1	1	0	0	0	0	0	0	0	0
s10	0	1	1	1	1	0	1	0	0	1	1	1	0
s11	0	1	1	1	0	1	0	0	0	0	0	0	0
s12	0	1	1	1	1	0	0	0	1	0	0	0	0
s13	0	1	1	1	1	0	0	1	0	0	1	1	0
s14	0	1	1	1	1	0	1	0	0	1	1	1	0
s15	0	1	1	1	0	0	0	0	1	0	1	1	0
s16	0	1	1	1	0	1	0	0	0	0	0	0	0
s17	0	1	1	1	0	0	0	0	0	0	1	1	0
s18	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 2: Implementation for Each State

FullCalc Combination:

The final design for the lab was to combine the top-level CU and top-level DP together as shown in Figure 4.

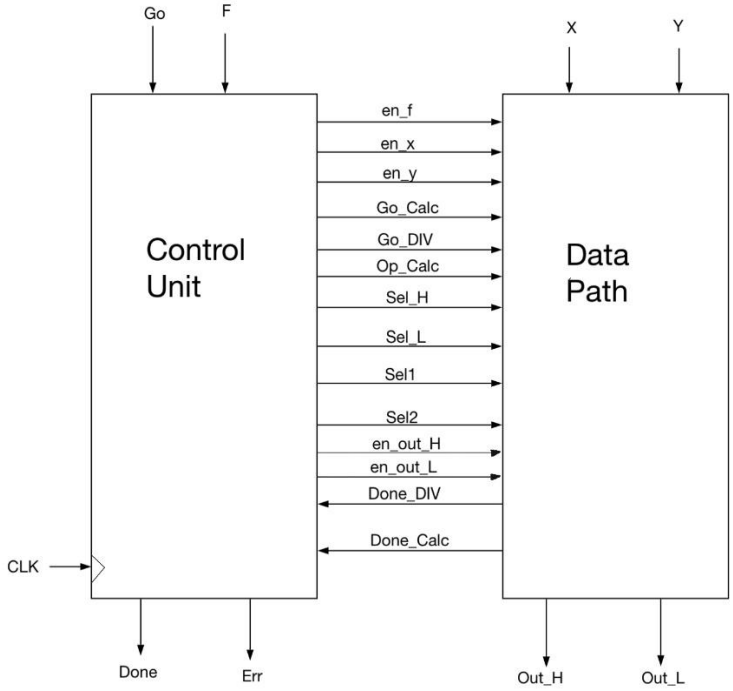

Figure 4: Complete FullCalc Design

Since there were two extra blocks added for DP, there were two extra control signals were added for CU. Also, due to the property of division, which the divisor cannot be zero, there were one *Err* flag was added for CU.

FPGA Setup:

Figure 5 showed the FPGA setup on Board via the constrain file. The three buttons with peach rectangle were the rest, go, and button. The right side four switches were the digit for In2, next In2 was In1. The LEDs with yellow rectangle were the state display. The left side of the board were the Done and Err LED. The 7-seg display would show the final answers.

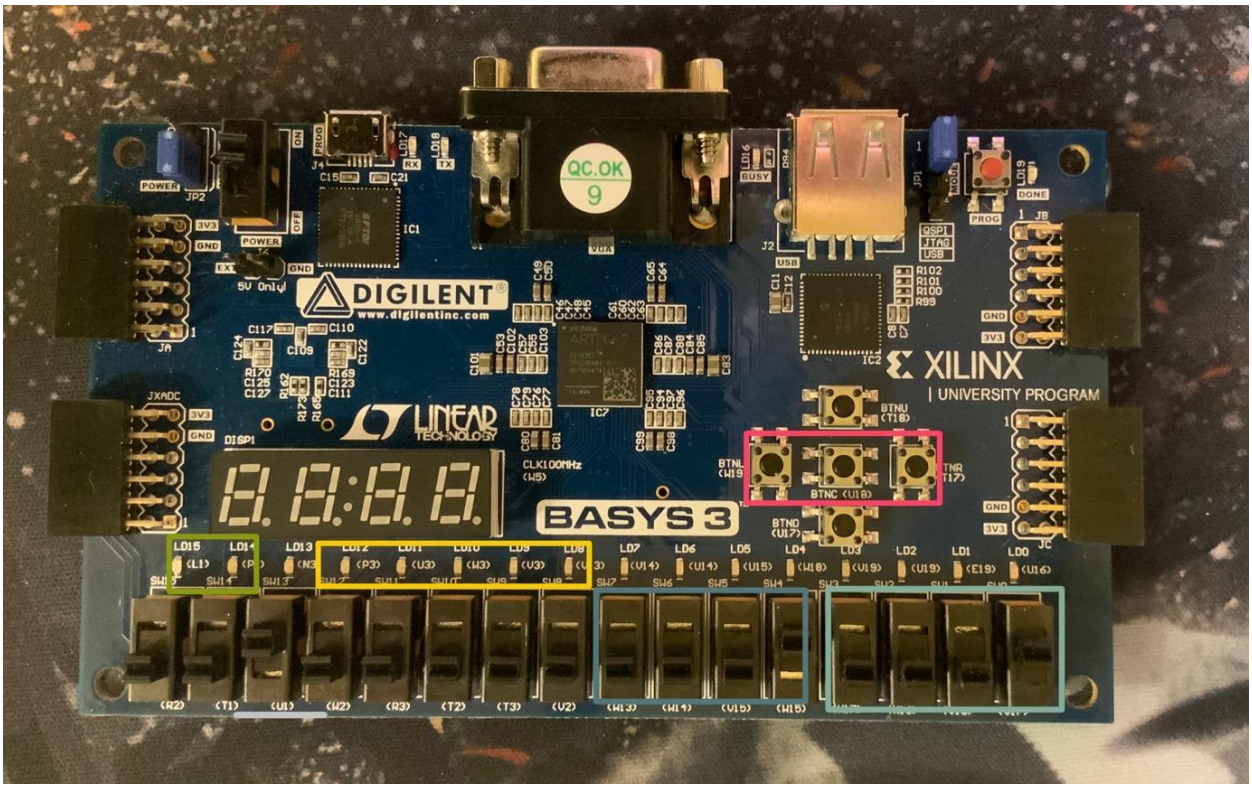

Figure 5: FPGA Setup.

Test Simulation:

This was the part we did not finish. By looking at the schematic design, we noticed that the final issue was some variables in DP did not connect with sub-system. However, we did not have more time to correct it and the by looking through the code, it supposed to be connected. Due to this issue, when we run simulation, there were some variables showed X.

Conclusion:

Also, we do not have time to show how much we really done at the end of class. However, this lab provided a deep learning of top-level design. In addition, we learnt a lot from this semester. Now, we could interpret waveforms, do self-debugging, and use schematic from RTL analysis to check block connection. For this lab, we designed the top-level structure but cannot demonstrate on waveform and FPGA board.