Lab 5 Report

Matthew Crump A01841001
Nicholas Williams A02057223

Objectives

The objective of this lab is to implement an accumulator on our DE10-lite board by implementing a finite state machine into our design. One push button is designated as an accumulator, and the other push button is designated as a reset. The ten switches control the value to accumulate by when the accumulate button is pressed. The ten LEDs reflect the value of the ten switches. The six 7-segment displays show the total value that has been accumulated since the reset button was last pressed.

Procedure

We started out by reading the lab-5 requirements for the accumulator, and we reviewed the lecture notes on finite state machines. We then used the System Builder tool to initialize the Quartus Prime project. We determined that we needed the following I/0 for this lab: the system clock, the push buttons, the seven segment displays, the switches, and the LEDs.

We then designed the finite state machine that we were going to implement in VHDL for this lab. Figure 1 shows our finite state machine. For our design every button and transition between states occurs synchronously. Our accumulator waits in the IDLE state until the add button is pressed. When the add button is pressed, the accumulator moves to the ACCUM state, and it stays there until the add button is released. The reset button moves the state to the IDLE state and sets the accumulator value to zero.

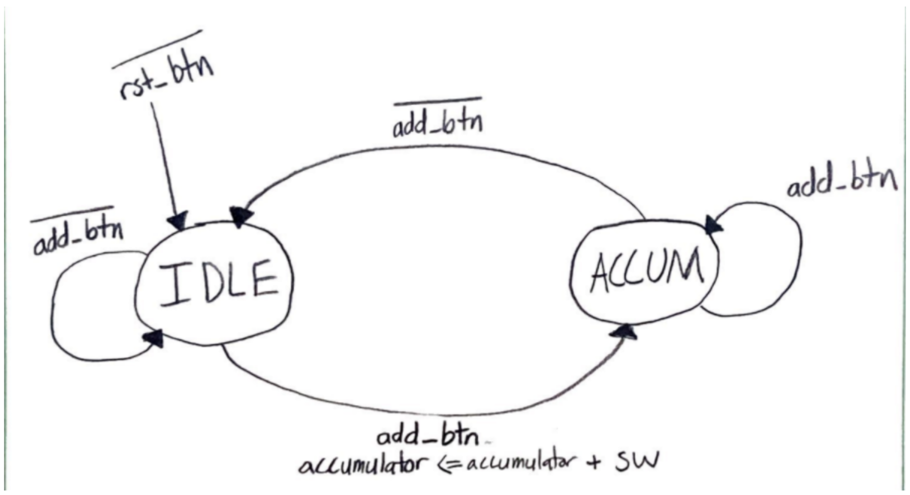
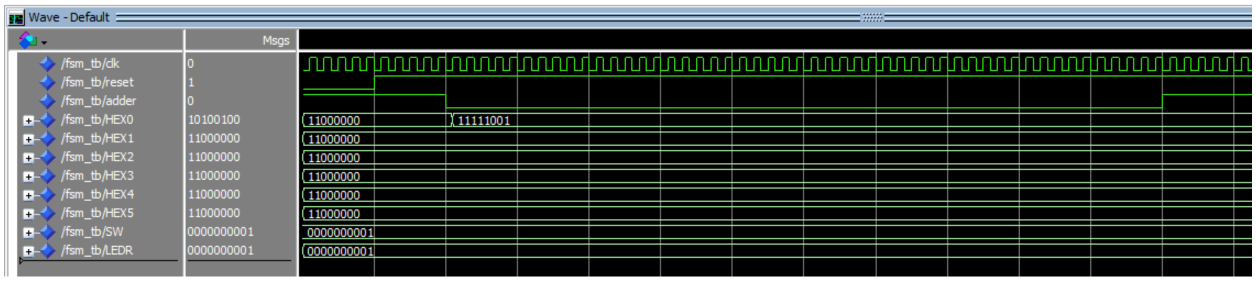

$\overline{\text{rst\_btn}}$
$\overline{\text{add\_btn}}$
add_btn
$\overline{\text{add\_btn}}$
IDLE
ACCUM
add_btn
accumulator <= accumulator + SW

Figure 1. Accumulator Finite State Machine

After designing our finite state machine we implemented our design into VHDL. We then simulated our finite state machine in ModelSim to determine if our design was ready to be programmed onto the FPGA.

Results

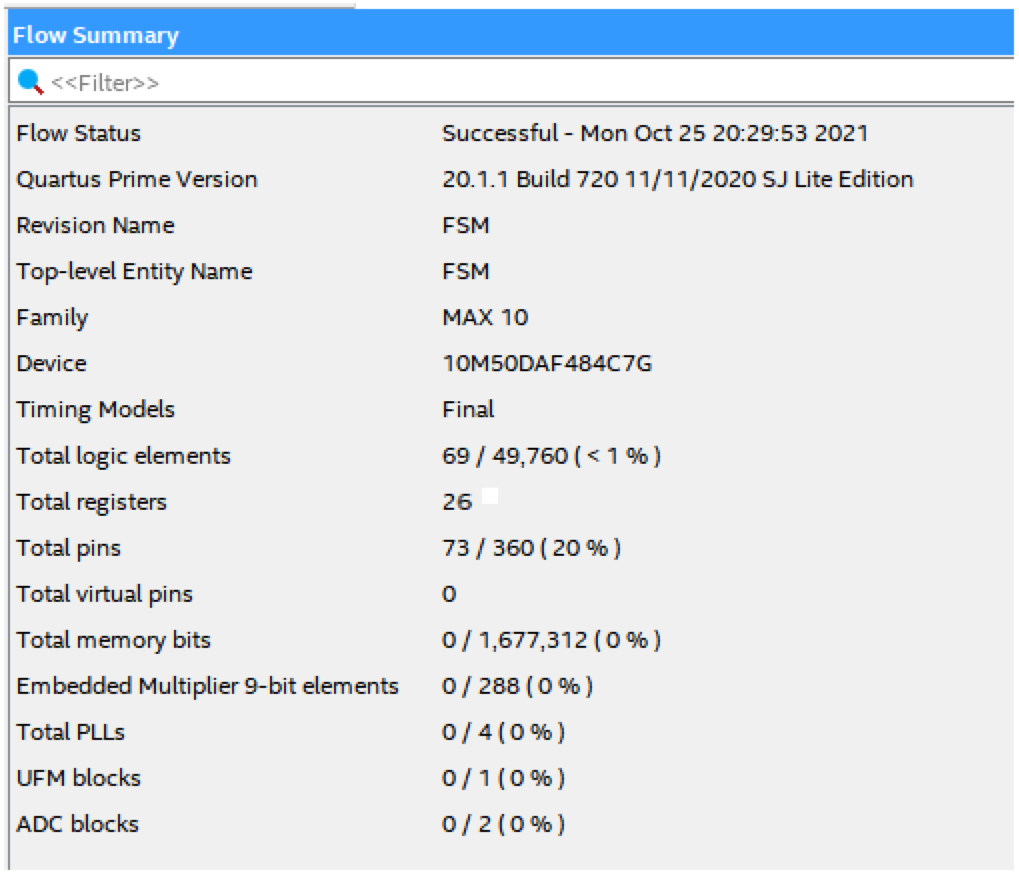
We created a testbench to ensure that our state machine worked as designed. Especially to ensure that even when the button is pressed for a long time, it still counts as one button press. This behavior can be seen in Figure 2. Despite this working in simulation, after 200 or so button presses, the count would skip a few. This meant that in our simulation either we didn't account for something or the testbench did not reflect the actual design. We eventually found the cause of the problem, the debounced signal from the debouncing circuit on the FPGA, is not synchronized to the FPGA internal clock. After synchronizing the input before using it in the state machine, the glitching never occurred again.



```
/fsm_tb/clk     0
/fsm_tb/reset   1
/fsm_tb/adder   0
/fsm_tb/HEX0    10100100     11000000   11111001
/fsm_tb/HEX1    11000000     11000000
/fsm_tb/HEX2    11000000     11000000
/fsm_tb/HEX3    11000000     11000000
/fsm_tb/HEX4    11000000     11000000
/fsm_tb/HEX5    11000000     11000000
/fsm_tb/SW      0000000001   0000000001
/fsm_tb/LEDR    0000000001   0000000001
```

Figure 2. Simulation Waveform

As expected, the compilation results show that 26 registers were used in the design: 24 counter registers, 1 state register, and 1 register to synchronize the debounced button input. A more comprehensive summarization is shown in Figure 3. We tried pressing the button very briefly or for long periods of time and observed that the accumulator only incremented once. We tried values on the switches ranging from 0 to all switches enabled. In all cases, the accumulator incremented by the correct amount and only cleared once the reset button was pressed.


Flow Summary

<<Filter>>

Flow Status	Successful - Mon Oct 25 20:29:53 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	FSM
Top-level Entity Name	FSM
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	69 / 49,760 (< 1 %)
Total registers	26
Total pins	73 / 360 (20 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Figure 3. Flow Summary

Conclusion

We learned how to design and create a finite state machine with its corresponding diagram. The importance of simulation does not always guarantee success, especially when some signals are asynchronous to the FPGA clock. This lab was the easiest so far. Perhaps the requirements could lean this lab to implement a slightly more complex state machine.