10/29/2024

**Lab 06 Report**

Ryan Beck

Jared Bronson

**Objectives**

Implement a revised version of the Lab 5 Accumulator on the DE10-Lite development board. The design will use the two on-board push buttons. One push button will reset the board, and clear the accumulated value. The other push button will store the value on the 10-toggle switches in a FIFO. When five items are in the FIFO, it will be drained and the values from the FIFO will be added to the accumulated value. The 24-bit accumulated value will be displayed on the six 7-segment displays. The value of the toggle switches will be reflected on the 10 LEDs. Each Finite State Machine(FSM) will run on different clocks using a Phase-Locked Loop(PLL). The first FSM will run on 5 MHz clock and will manage the write side of the FIFO and Accumulator. The second FSM will run on a 12.5 MHz clock and will manage the read side of the FIFO and Accumulator. This must be implemented entirely in VHDL.

**Procedures**

Map out the first FSM with states for Clear, Waiting, Debounce, Pressed, Check, and Write. Map out the second FSM with states for Clear, Empty, Waiting, Accumulate, and Display. Next, create a Quartus Project File with the system builder to include the on-board clock, seven-segment displays, LEDs, push buttons, and toggle switches. Create a VHDL file with port names to match what is in the generated Verilog file, then remove the Verilog file from the project. In the generic, create an integer to use as a delay in Debounce to make sure the button does not add after false triggering.

Next, create a 10x128 FIFO to accept the value for the switches when the store button is pushed. Make variables that will be taken in by the FIFO to enable the writing process for the first FSM. Variables to enable the reading process for the second FSM will also be needed.

Create a PLL with needed variables to output the clock frequencies needed for each of the state machines. Initialize any other needed variables for the design of the FIFO Accumulator in the architecture. For the seven-segment display, make a lookup table to display each hex value. Last, create a type to be able to move through the states of each state machine.

Start by instantiating the FIFO and the PLL. Then, create a process sensitive to the reset key, to clear the FIFO. Set one process sensitive to the 5 MHz clock and one sensitive to the 12.5 MHz clock. In these two processes set the corresponding FSM to go to the clear state when reset is pushed, and set the current state to the next state.

In the next process, the sensitivity list will include KEY, fifo\_full, and the 5 MHz clock. This process will control the first state machine. In the clear state, all values will be cleared and the state machine goes to Waiting. The FSM goes to the Clear state if reset is pushed, and Debounce if store is pushed. A timer is incremented in the Debounce state. When the timer equals the delay value, if the button is still pushed, the FSM will go to the Pressed state. If the button is not pushed when the timer equals delay, the FSM will go back to the Waiting state. While the FSM is in the Pressed state, it waits for the store button to be released and will go to the Check state. The Check state looks at the amount of input in the FIFO. If the FIFO is full, the FSM goes to the back to the Waiting state, otherwise it will go to the writing state. Once in the Writing state, the value given by the 10 toggle switches is stored in the FIFO, and the FSM goes back to Waiting.

The next process will control the second FSM and will be sensitive to the push buttons, fifo\_empty, fifo\_full, rd\_en, and the 12.5 MHz clock. The Clear state will set everything to zero including the seven-segment display, and go into the Empty state. The FIFO is cleared in the Empty state by reading what is in the FIFO until it is empty. Once the FIFO is empty, the FSM goes to Waiting. If reset is pushed, the FSM goes back to the Clear state. This Waiting state will send the FSM to Accumulate only when the FIFO has 5 inputs. The Accumulate state will take an input out of the FIFO and add it to the accumulated value. Once the FIFO is empty, the FSM will navigate to the Display state. The Display state takes the 24-bit accumulated value and shows it on the six 7-segment displays. After the value is displayed, the FSM goes back to Waiting.

Define one more process sensitive to the 50 MHz clock to continuously update the LEDs to show the state of the 10 toggle switches. Implement the design on the development board to ensure it works as intented.

**Results**

The FIFO\_Accumulator file shown in figures successfully implements the design. There were a lot of issues implementing the two state machines, and processing data in the FIFO properly. There were problems with reading from the FIFO at the proper time, inferred latches, and the FSMs not going between states properly. To debug the design, a test bench was used to run simulations, as well as the signal tap logic analyzer in Quartus. Because of the issues, the design instantiates the PLL, but the FSMs are running off of the same 50 MHz clock.

**Conclusion**

In conclusion, we were able to implement a design using two state machines. The design took input from the 10 toggle switches and stored it in a FIFO when the store button was pushed. When the FIFO was full, the values were read and added to the accumulated value. The clear button resets all the values, including the 7-segment displays. The value of the toggle switches displays on the LEDs. While we instantiated the PLL were not able to implement it into the design of the two FSMs.

**Figures**A screen shot of a computer

Description automatically generated

Figure 1: FIFO\_Accumulator.vhd Pt 1

A screenshot of a computer program

Description automatically generated

Figure 2: FIFO\_Accumulator.vhd Pt 2

A screen shot of a computer

Description automatically generated

Figure 3: FIFO\_Accumulator.vhd Pt 3

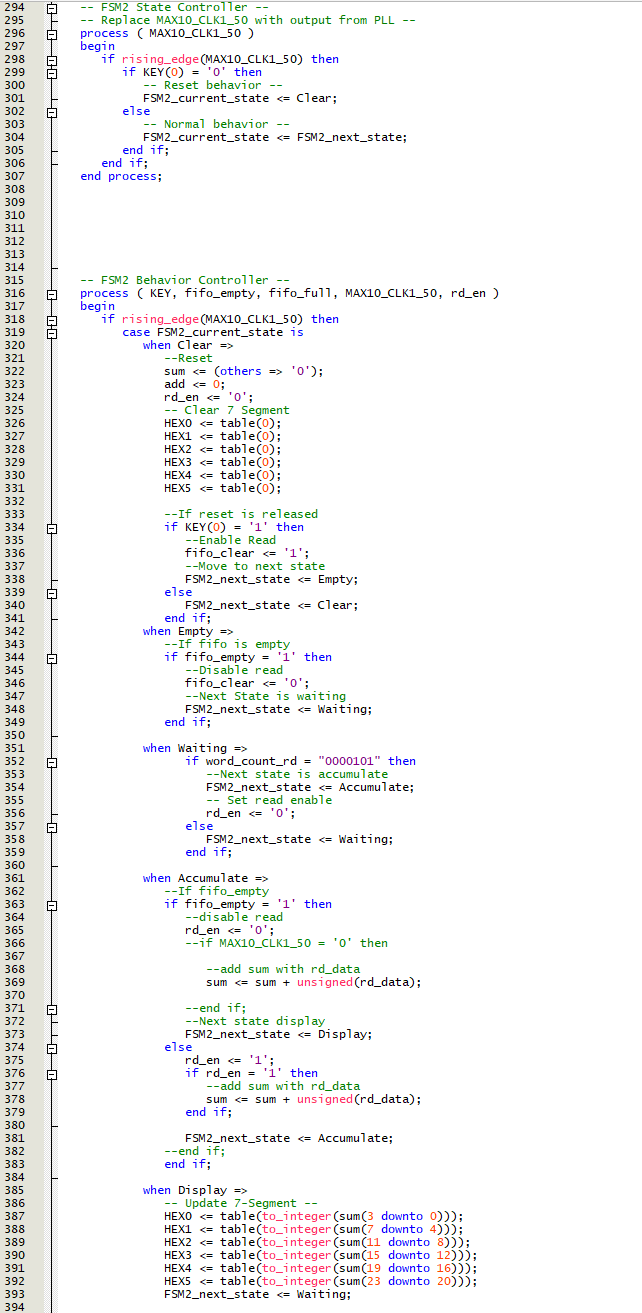


Figure 4: FIFO\_Accumulator.vhd Pt 4

A computer code with text

Description automatically generated

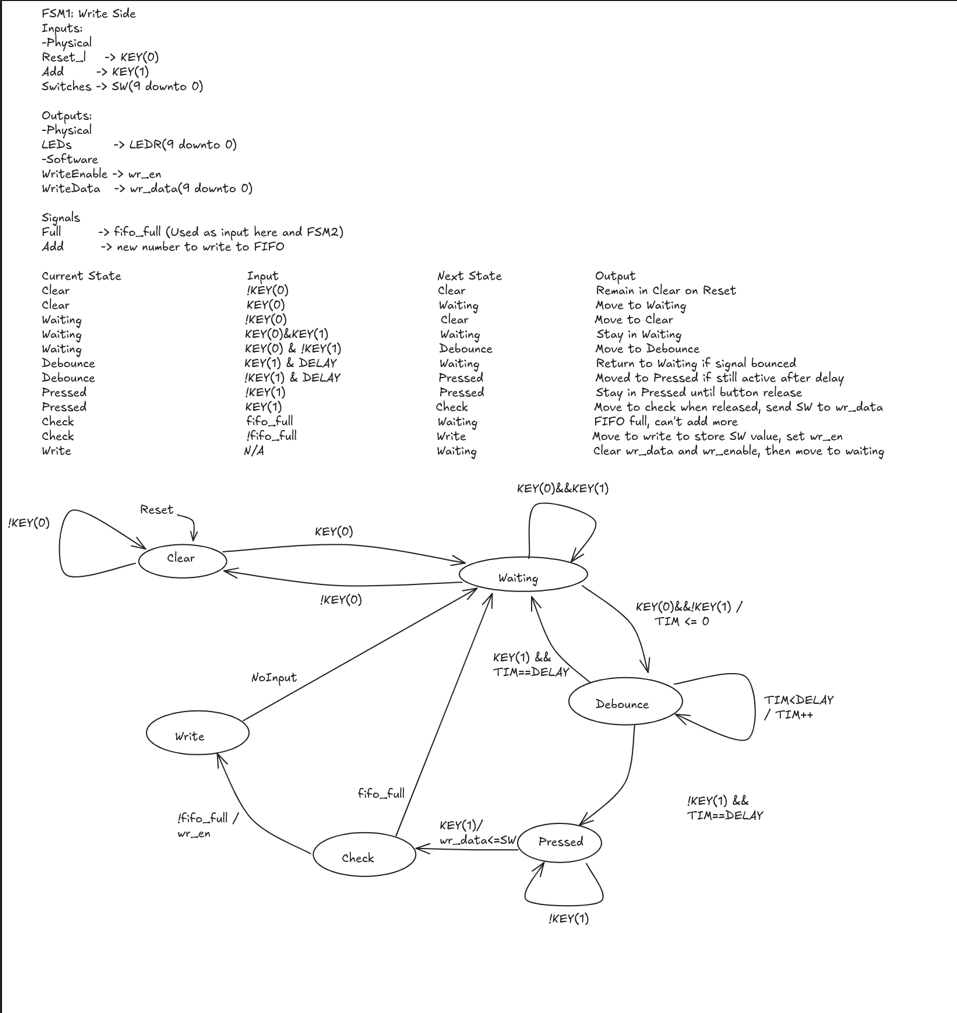
Figure 5: FIFO\_Accumulator.vhd Pt 5

Figure 6: FIFO\_Accumulator FSM1

A diagram of a diagram

Description automatically generated

Figure 7: FIFO\_Accumulator FSM2