10/22/2024

**Lab 05 Report**

Ryan Beck

Jared Bronson

**Objectives**

Implement a 24-bit accumulator on the DE10-Lite board. The accumulator must be implemented as a VHDL Finite State Machine. One push button will be used to “add” a new value to the accumulated value. The accumulated value will be displayed as a hexadecimal value on the six 7-segment displays. The other push button will clear the accumulated value and reset the six 7-segment displays to show all zeros. Using the 10 toggle switches on the board, the 10-bit unsigned value to be added to the accumulator will be provided. The accumulator can be used by setting the toggle switches to the desired number, then pressing “add” to update the accumulated value. The 10 LEDs reflect the state of the 10 toggle switches. A logic ‘1’ turns the LED on, and ‘0’ turns it off. The “add” button will be de-bounced using the State Machine. One push of the “add” button will result in only one accumulate operation, regardless of how long the button is pushed.

**Procedures**

Map out a Finite State Machine (FSM) with states for Clear, Waiting, Debounce, Pressed, and Accumulate. Using the system builder, create a Quartus Project File that uses the on-board clock, 7-segment display, push buttons, toggle switches, and LEDs.

Next, create the architecture of the project. Create a 10-bit signal to take the input value from the toggle switches, a 24-bit signal for the accumulated value, and an integer to count clock cycles when “add” is pushed. Then, create a type with each of the states outlined in the FSM, and a signal to go through each of the states. Define a table with values to be called so the accumulator value can be shown on the 7-segment.

Define a process for the on-board clock. In this process, on the rising edge of the clock, the next state of the state machine becomes the current state. Define another process for the on-board clock. This process will have a case statement to define what inputs move states in the FSM. Implement the designed FSM using the case statement. The Debounce state will have a delay to ensure the FSM does not go to the next state if the button is falsely triggered. Create another process sensitive to the clock. This process will update the LEDs to show the state of the switches, and update the 7-segment display to show the accumulated value.

**Results**

The accumulator file, shown in figures, successfully implements the desired 24-bit accumulator on the DE10-Lite board. The design takes an unsigned 10-bit number from the toggle switches and displays the state on the LEDs. While the “add” button is pushed the accumulated value does not change. Once the “add” button is released, the 10-bit value from the toggle switches is added to the accumulated value. When the reset button is pushed, the display and the accumulated value is cleared.

**Conclusion**

In conclusion, we were able to implement an accumulator on the development board. One push button adds a value from the toggle switches to the accumulated value and shows it on the six 7-segment display. The other push button will clear the accumulated value and show all zeros on the display. If the “add” button is held, none of the values change. Once the button is released, only one accumulate operation is performed.

**Figures**

A screenshot of a computer program

Description automatically generated Figure 1: Accumulator.vhd Pt 1

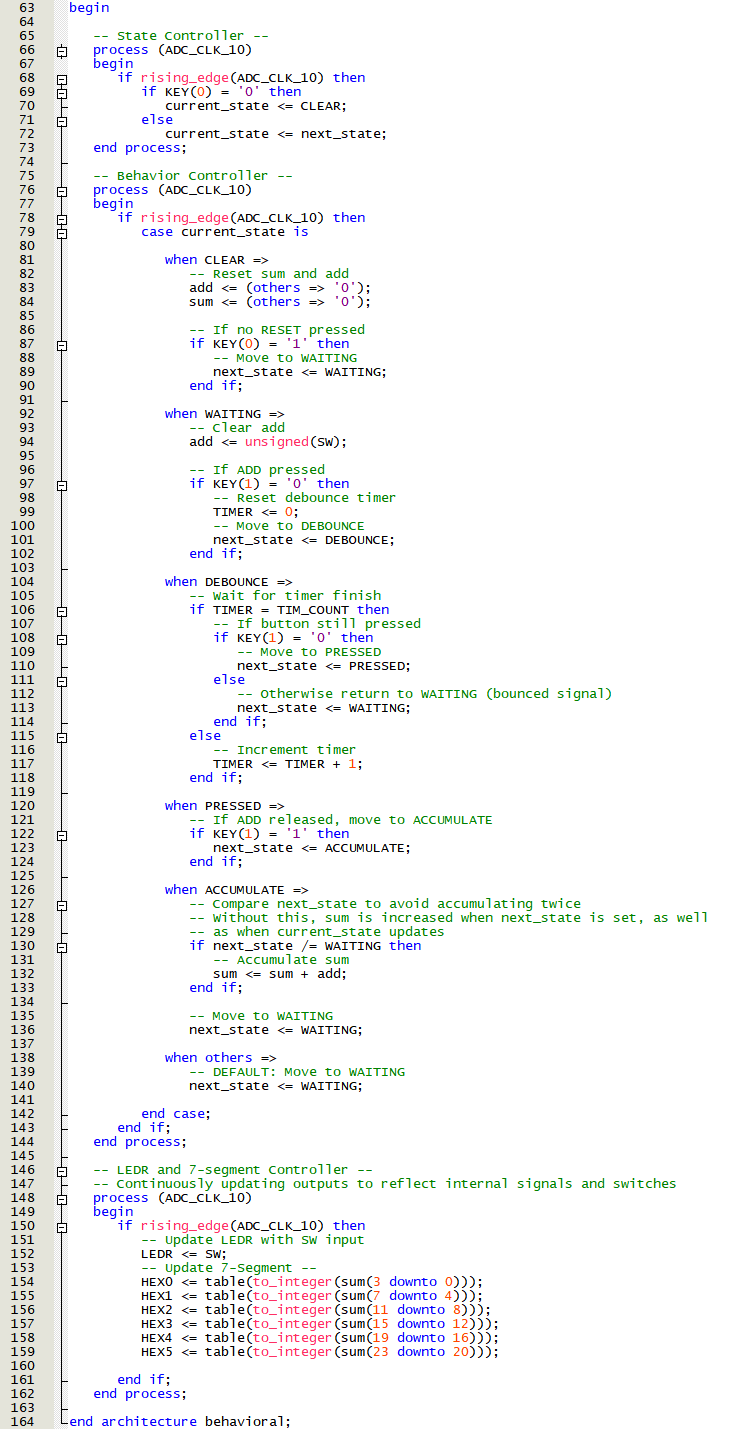


Figure 2: Accumulator.vhd Pt 2

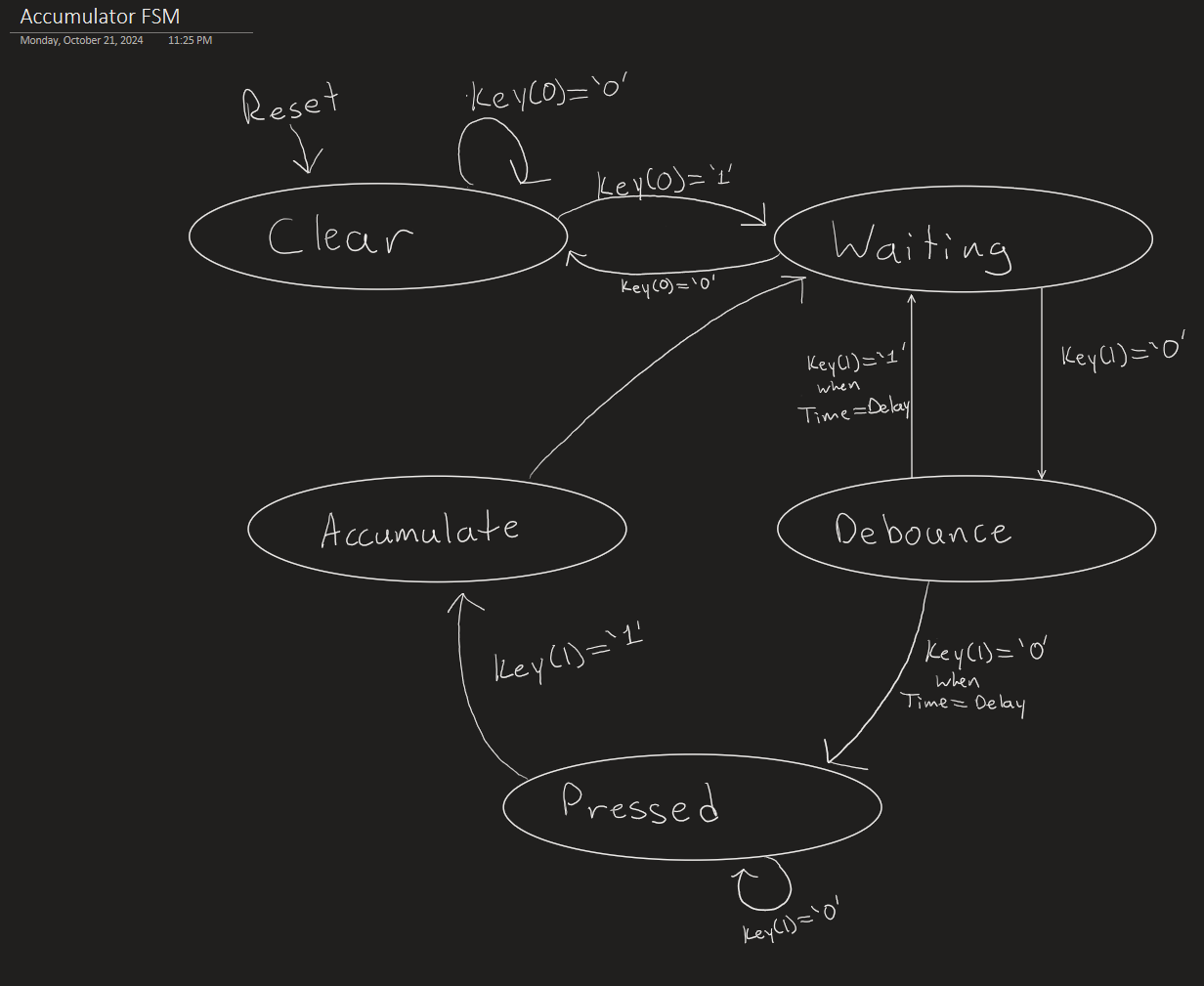


Figure 3: Accumulator FSM