[Labs 3-4: Fayetteville Bike Crossing/Timers]

[Blake Williams]

[09/29/2024]

**Summary**

This is the combination of Labs 3 and 4. The overall goal of the Labs were to use a State Machine to simulate a bike crossing light. Lab 3 goal was to implement the State Machine and use a delay\_ms() function which delayed the program by the amount passed in. Then in Lab 4 a Timer was introduced allowing for a more asynchronous execution. Allowing the program to continue running, while the timer executes alongside the program. Taking the place of the delay\_ms() function.

**Design and Implementation**

**Lab 3 Design and Implementation**

For this lab we were provided with starter code consisting of a skeleton of a state machine, and a while(1) infinite loop in main. From here the defines used in previous labs were exported to this project. Those of the GPIO\_BASE\_ADDR, PUSH\_BTN\_BASE\_ADDR, PUSH\_BTN\_REG, RGB\_LEDS\_BASE\_ADDR, and RGB\_LEDS\_REG. Then in the main loop the buttons Tri state was set to input and the rgbLEDs were set to output. Next the provided delay\_ms() function was imported into the code. After completing the boilerplate, the first area of work was the GREEN state. If a button is pressed, then the state machine must recognize this and transition from the GREEN state to the FLASH\_RED\_START state. To do this current button state and previous button state variables were declared. The current button state variable stores either a 1 or 0 representing if the button is turned on or off. The previous button state stores the state of the button from the previous cycle. The GREEN state checks to see if the current button state is a 1 and the previous button state is a 0. If this is true then a state transition occurs. Once in the FLASH\_RED\_START state, if the user presses the button again it must add 6 seconds onto the current timer. Along with the button press logic, the LED lights were required to blink red for the duration of the time in the FLASH\_RED\_START state. To achieve both goals three variables were declared. These being the state timer which stores how much time remains before a state transition, elapsed time which stores how long has the current state run, and time in cycle which is the elapsed time mod 1000. When the state is first entered the state timer variable is set to 6000ms or 6 seconds, and the elapsed time variable is reset. Then the time in cycle variable is utilized by turning on the red LEDs if it is less than 500 otherwise turning them off. This simulates flashing red lights. After the lights flash the state timer is decremented by 10 and the elapsed time is incremented by 10. Finally, the same check that was used in the GREEN state is used here to check for button presses. If the button is pressed 6000ms are added to the state timer. Once the state timer goes to 0 then a state transition occurs to RED for 4 seconds and to FLASH\_RED\_END where no user input is taken, and the lights flash for 6 seconds total. After FLASH\_RED\_END finishes it causes a state transition back to GREEN where the cycle is primed to start over again.

**Lab 4 Timers**

Building off Lab 3 an external timer was added to the VIVADO block design, and a new application was created utilizing it. Building upon the same code from last time, the delay\_ms function was completely removed and replaced with the timer. To the previous code new defines were added being the AXI\_TIMER\_CONTROL\_BASE and AXI\_TIMER\_LOAD\_BASE which pointed to the base addresses of the timer control register and the timer load register. The new design would utilize the timer to increment a count variable. Each increment of this variable would represent a half second, allowing for more precise timing of the state transitions than the previous delay\_ms() function. To achieve this the timer was needing to be initialized. The timer runs at 81.42MHz or 81247000 operations per second. If this number was passed into the load register then the timer would trigger the interrupt bit every one second. Therefore, this number is divided by two when it is passed into the register, allowing for the timer to represent 0.5 seconds. Then into the timer control register the value 00000110010 was passed in to Enable interrupts, enable the load register, and make the timer count down from the loaded value. Then using a XOR operator the Load register was toggled off and the timer was enabled. Inside the main while loop there are two if statements. The first checks if the interrupt bit has gone high, this indicates the timer has ended and restarted. When this occurs, a count variable is incremented and a 1 is written to bit 8 resetting the interrupt. The second if statement checks for the first button press by seeing if the button was pressed, and if the current state is GREEN. If both of these checks pass, then the a state transition occurs to the FLASH\_RED\_START state. This state was modified from the previous lab by removing the state time, elapsed time, and time in cycle variables and replacing them with the count. Since the count increments once every cycle, then the lights now blink on if count is even and off it count is odd. Then if the button is pressed while in the FLASH\_RED\_STATE the count is reset back to 0, simulating adding 6 seconds onto the timer. Once count reaches 12 then a state transition occurs to RED. In red once count reaches 8 (4 seconds) it transitions to FLASH\_RED\_END then to GREEN restarting the cycle.

**Conclusion**

Overall, both labs were great and rolled into each other very smoothly. Lab 3 reinforced what was learned in Lab2 and more. It also left the question of “Is this the best way to do this” in the back of my mind. This question was answered with Lab 4 in which that question was refuted with timers. Timers simplified the entire process by reducing all the variables that were used in Lab 3 to only a single variable. This made the implementation process a lot smoother, and because a timer was being utilized the timekeeping was more accurate than the delay\_ms(). After finishing lab 4 the question of “Is this the best way to do this” came back, which is promptly disputed with lab 5 and interrupts.

***NB:*** *While the headings listed above satisfy the minimum requirements of the assignment. You are allowed to deviate from this template where necessary. For example, wherever you believe adding additional sections and or sub-sections is necessary.*