Stopwatch Design Using Seven-Segment Display and GPIO Buttons

Oviya Seeniraj

1. Purpose and Expected Goals:

The purpose of this lab is to design and implement a stopwatch using the seven-segment display and push buttons on the Nexys development board. The design goal is to create a stopwatch that starts, stops, resets, and counts up or down, based on user input through the buttons. The expected result is a fully functional stopwatch that interacts with real-time external events using interrupts.

2. Methodology:

a. Overview of Design Tasks:

- Design a timer-based system to update the stopwatch display at a frequency of 2 Hz (or more).
- Implement button interrupts to control stopwatch functions such as start, stop, reset, and direction of counting (up/down).
- Ensure that the seven-segment display updates without flickering, providing a smooth user experience.

b. Assumptions:

- The seven-segment display can be updated fast enough to prevent dimming but slow enough to avoid flickering.
- We can handle button inputs using debounce logic to avoid false triggers from switch bouncing.

c. Observations from Design Tasks:

- Using interrupts for buttons avoids the need for polling, allowing for real-time response to user inputs without affecting the stopwatch accuracy.
- The challenge was ensuring smooth updates to the display while keeping track of time.

d. Plan for Design Testing:

- Test each button function incrementally to confirm it performs the expected action (start, stop, reset, count up, count down).
- Monitor the display for flickering, and adjust the display refresh rate if necessary.
- Test the system for simultaneous button presses and observe the behavior.

3. Results:

a. Metric Results:

- Accuracy: The stopwatch successfully counts in increments of 0.01 seconds, meeting the goal of displaying fractions of a second (0.1s resolution).
- Button Functionality: All buttons function as expected, controlling start, stop, reset, count up, and count down.

b. Limitations of the Design:

- The stopwatch cannot count below zero when in the countdown mode, which meets the specification but is a limitation if extended functionality is desired.
- The timer's resolution is limited by the system's clock frequency and the need for a smooth display refresh rate - while the board has an oscillatory clock rate of 100 MHz, limiting our stopwatch to 0.01s measuring / display precision prevents flickering as visible to the human eye.

c. Design Test Results (if assumptions did not meet expectations):
The timing accuracy was as expected, but care had to be taken to ensure no visible glitches or irregularities in the display (not going into the thousandths or ten-thousandths of a second due to visible smoothness limitations).

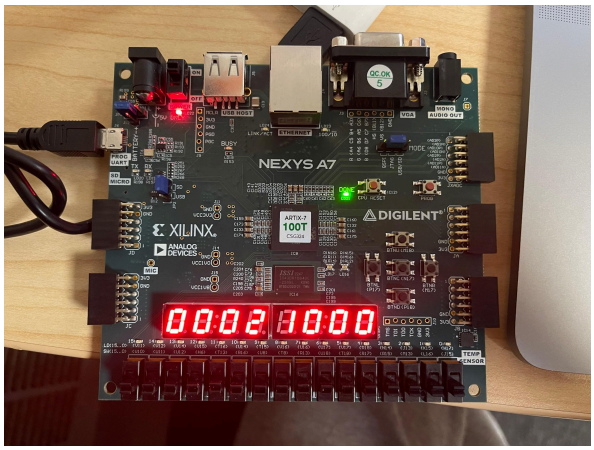
d. Roadblocks and Parts That Did Not Meet Spec:
Initially, there were challenges with debounce handling on the buttons, which caused multiple interrupts for a single button press. This was mitigated by adding button handling interrupts, but creating actual debounce logic would be a better fix.

e. Issues and Sources of Errors:
Flickering: At lower refresh rates from the initial sequence recommended in the lab (2 Hz), some digits on the display were brighter than others due to the multiplexing method used for the seven-segment display (e.g the first and last two digits on the 2 Hz). Increasing the display refresh rate to 100Hz by giving the timer interrupt a reset value of 0xF4240 (timer counts 1M cycles between each interrupt → 1M/100M → 0.01s between each stopwatch update → 100Hz refresh rate) eliminated the issue.

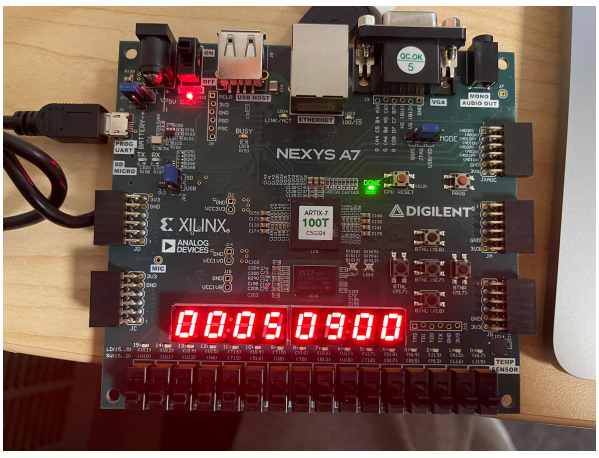
f. Suggested Improvements
- We could improve the accuracy and reliability of the stopwatch by implementing a software debounce mechanism for buttons.
- We can improve the display brightness by adding logic to adjust the time each digit is displayed.

4. Photos:
Two photos were captured to demonstrate the stopwatch functionality:

1. Stopped Timer: After stopping the timer, all digits remain clearly visible with no flicker.



2. Running Timer: While the timer is running, the digits update smoothly, and the illumination appears constant across all digits.



5. Reporting Questions:

1. How fast of a stopwatch are you able to build using your hardware and software configuration? What was the limiting factor in the accuracy of your stopwatch?

   I successfully built a stopwatch to count up to 0.01-second increments with the current hardware. The limiting factor in the accuracy is the timer resolution and display refresh rate. At lower refresh rates, display flickering becomes more noticeable. We need to balance the appearance of flickering due to overly high refresh rates with the dimness of digits due to low refresh rates.

2. How did you control the display update timing (i.e. loop period) and did this affect accurate timing measurement? If so, how? What display update period did you choose?

   I controlled the display update timing using a timer interrupt to ensure I consistently updated the seven-segment display. I chose a refresh rate of approximately 100Hz to avoid flickering and ensure that all tens and hundredths decimal places are accurately reflected in the stopwatch.

3. The push buttons in this stop-watch are implemented using interrupts. Would it be possible to instead check the values of your push buttons while executing the grand loop? Would this polling approach change the timing of your stop watch?

   Yes, but using a polling approach in the grand loop would most likely introduce delays in button response and affect the stopwatch's real-time performance. Delays are extremely irritating from a UX perspective.

4. What happens if two push buttons are pressed at once?

   If two buttons are pressed simultaneously, the stopwatch prioritizes the first detected interrupt due to a mixture of hardware or human error. Handling multiple button presses simultaneously would require additional logic to detect and respond to each button press independently, perhaps using interrupt priorities.

5. Describe the UI of the stopwatch. List the input and output devices.

   The stopwatch has a simple user interface, with the numbers of the Nexys's built-in 7-segment display stopwatch reacting to button presses (reset, start, stop, count up, count down)

   - Inputs: Five push buttons (Start, Stop, Reset, Count Up, Count Down).
   - Outputs: Eight-digit seven-segment display showing time in seconds and fractions of a second.

6. Describe anything unique that you did to make your stopwatch work better.

   This is not necessarily unique, but I handled interrupts for buttons to ensure real-time responsiveness. In order to improve my code further, I could implement debouncing.