Class Report 2: 6.5.6 Reaction Timer

Samuel Hussey

1 Introduction

In this project, a reaction timer circuit was designed and implemented in which a user may initiate a test of their reflexes by pressing a button and waiting for an led to illuminate during a random interval between 2 and 15 seconds. During this time, an led onboard the Nexys4 will illuminate and the user must push a button as quickly as possible to record their time. The timer display will halt if the user takes longer than 1 second to react, but if the button is pressed before the led turns on the display will show '9999' and the test must be restarted. Source files can be found here and a video demonstration here.

2 Implementation

The implementation of the circuit began with the design of 4 individual timer modules, each to control the count of a digit on the 7-segment display. This approach was chosen over a binary to binary-coded decimal converter for simplicity's sake. Each timer counts up to 9 at millisecond, centisecond, decisecond, and second speeds while driving the output to a time-multiplexed 7-segment display driver. A final timer was then used to coordinate the display with the 7-segment display output. A state machine is used to implement the rules of the test and coordinate the user interface buttons with the timers and display. A design sketch of the circuit is shown in figure 1.

3 Results

The simulated timing diagram is shown in figure 2. This figure demonstrates the 7-segment outputs changing in conjunction with the button input stimulus. The display shows the message 'HI' until btn[0] is pressed. This is followed by the display turning off while the simulated user waits for the led to illuminate. Pressing the 'clear button' returns the cricuit to its original state.

4 Discussion

The use of 5 separate timers made for a crude implementation and proved difficult to coordinate during state transitions. In the future, a single timer counter and decimal converter would be a more elegant and simple approach. Issues were also encountered with random interval generation. Ultimately, the solution was to utilize a constantly running mod-m timer and check its value against a hard coded constant. Though not random, the impression of randomness is given depending on how the user interacts with the application.

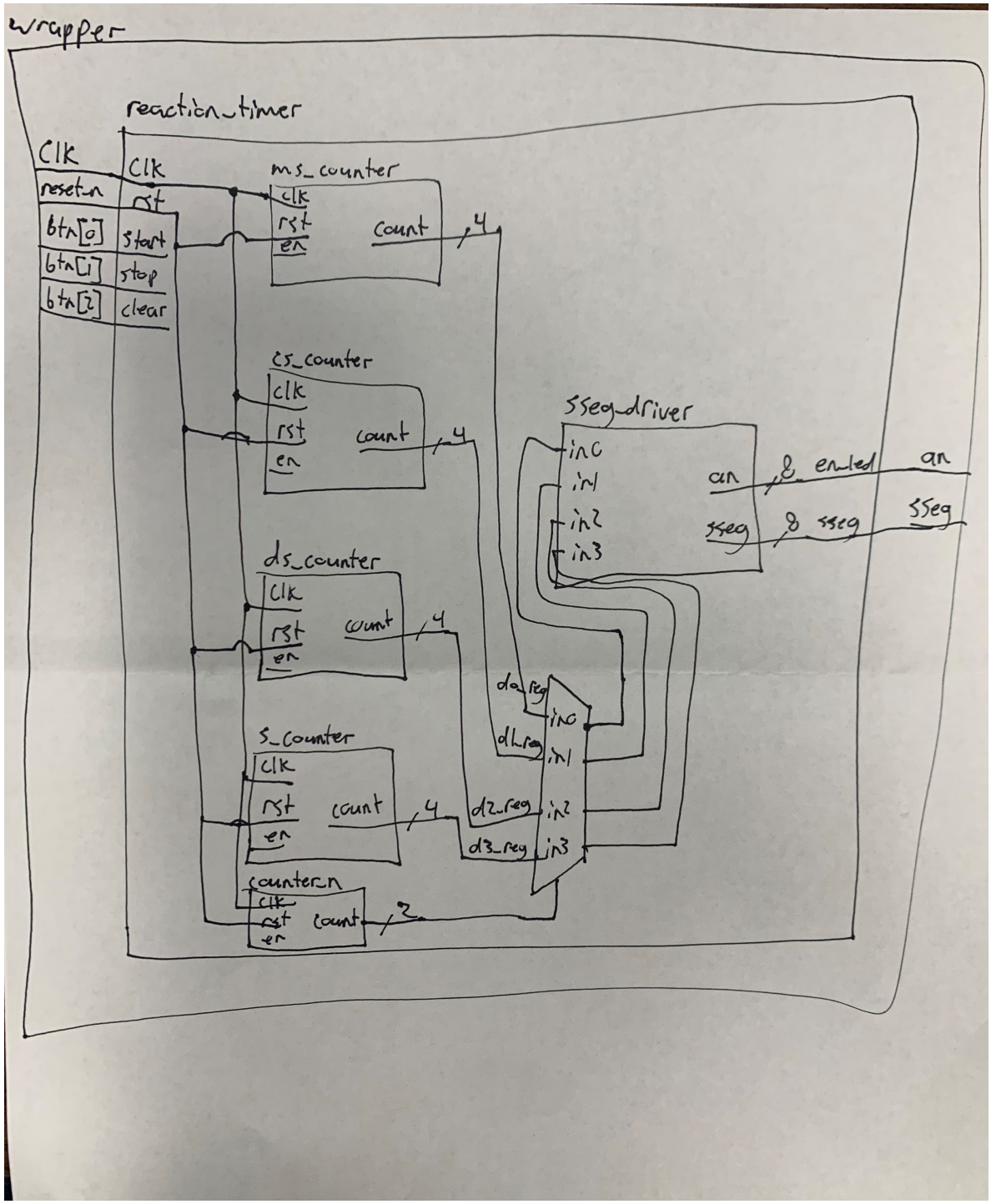

Figure 1: Design sketch of rotating square circuit.

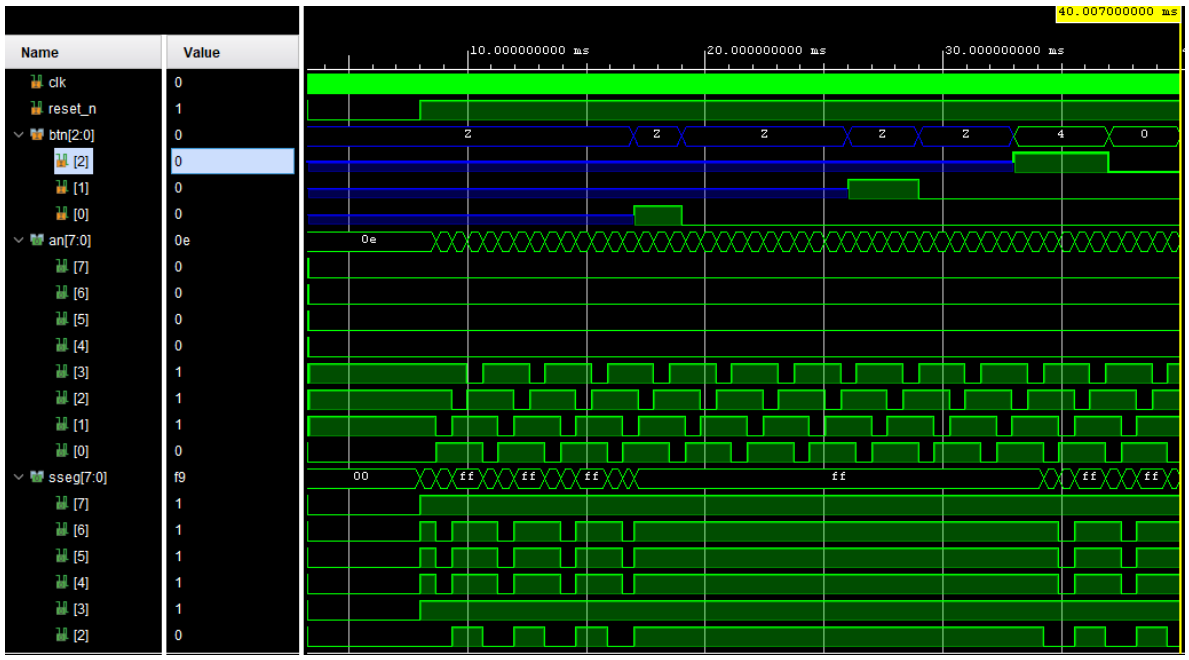

Figure 2: Timing diagram of testbench.