ENEL373: Digital Electronics and Devices

University of Canterbury

This is an Awesome Title for Our First Report

Philip Brand *(15776928)*

April 9, 2025

Contents

Introduction

Human reaction time is the interval between the a stimulus and the response to that stimulus. There are two main types of reaction time; simple, and choice. Simple reaction involves reacting to a singular stimulus, while choice reaction involves choosing the correct response from multiple stimuli. This project aims to implement a simple reaction time recorder on an FPGA in VHDL.

The reaction stimulus is three LEDs that turn off sequentially, with a random delay between each LED turning off. Once the last LED turns off, the user must press a button, and the delay is displayed in milliseconds on 8 7-segment displays. The FPGA stores the reaction delays from the last three trials, and can display the fastest, slowest, and average reaction times. These values can be reset by the user.

Design Summary

Design Details

Finite State Machine

The reaction timer was managed by a finite state machine with the following states: (change these to match the ones in the code)

- **Idle**: The system is waiting for the user to press the start button. The LEDs are on, and the 7-segment displays are blank.
- **Countdown**: The system is counting down from a random number between 1 and 5 seconds. The LEDs are off, and the 7-segment displays are blank.
- **Reaction**: The system is waiting for the user to press the button after the last LED turns off. The LEDs are off, and the 7-segment displays are blank.
- **Display**: The system is displaying the reaction time on the 7-segment displays. The LEDs are off, and the 7-segment displays show the reaction time in milliseconds.
- **Store**: The system is storing the reaction time in the memory. The LEDs are off, and the 7-segment displays are blank.
- **Reset**: The system is resetting the memory. The LEDs are off, and the 7-segment displays are blank.

The reaction delays were stored in a three-element circular buffer.

Pseudo-Random Number Generation

The pseudo-random delay between the sequential deactivation of the LEDs was generated using a linear feedback shift register (LFSR) and a clock divider. An eight-bit LFSR was used to generate pseudo-random numbers, and a selection of those bits were assigned to the most significant eight bits of the upper bound for the clock divider. The next pseudo-random number was generated on a rising clock edge from that clock divider, as seen in Listing ??.

```vhdl
--  Generate next "random" number from the LFSR in random_number_generator
  ff9: random_number_generator port map (CLK_IN => clk_var_hz,
                                         RAND_OUT => rand_num);

-- Set the upperbound for the variable clk based on the random number
  clk_var_hz_divider_bound(27 downto 20) <= rand_num;

-- Generate another clk square wave to trigger a new random number
  ff10: clk_divider port map(CLK100MHZ_IN => CLK100MHZ,
                              SLOWCLK_OUT => clk_var_hz,
                              UPPERBOUND_IN => clk_var_hz_divider_bound);
```

Code Listing 1: Behavioural VHDL implementation of pseudo-random number generation.

If all eight bits of the LFSR set the most significant bits of the upper bound, the minimum delay between LEDs turning off would be,

$$\text{Minimum Delay} = \frac{2^{20} - 1}{100\ \text{MHz}} = 11\ \text{ms}.$$

It was decided this would be too short a delay. Instead of reducing the number of bits in the LFSR, the four most significant bits of the upper bound were set by the middle four bits of the LFSR. This increased the minimum delay to,

$$\text{New Minimum Delay} = \frac{2^{24} - 1}{100\ \text{MHz}} = 167\ \text{ms},$$

which was acceptable. The maximum delay remained unchanged, at

$$\text{Maximum Delay} = \frac{2^{28} - 1}{100\ \text{MHz}} = 2.68\ \text{s}.$$

The location of the taps in the LFSR were chosen to ensure a maximum period of $2^8 - 1 = 255$ cycles. These locations were 8, 6, 5, and 4 [1], as seen in Lisiting **??**.

```
process (CLK_IN)
begin
    if rising_edge(CLK_IN) then
        current_rand(7 downto 1) <= current_rand(6 downto 0);
        current_rand(0) <= current_rand(7) XOR current_rand(5) ...
        ... XOR current_rand(4) XOR current_rand(3);
    end if;
end process;

RAND_OUT <= current_rand;
```

Code Listing 2: Dataflow VHDL implementation of LFSR.

Module Testing

Conclusions

References

[1] U. o. O. Department of Physics, “Electronics technical report 2012-1,” University of Otago, Tech. Rep., 2012, Accessed: 2023-10-04. [Online]. Available: `https://www.physics.otago.ac.nz/reports/electronics/ETR2012-1.pdf`.

Here is how we include a figure in the report. Figure **??** shows an example of a figure. A figure.

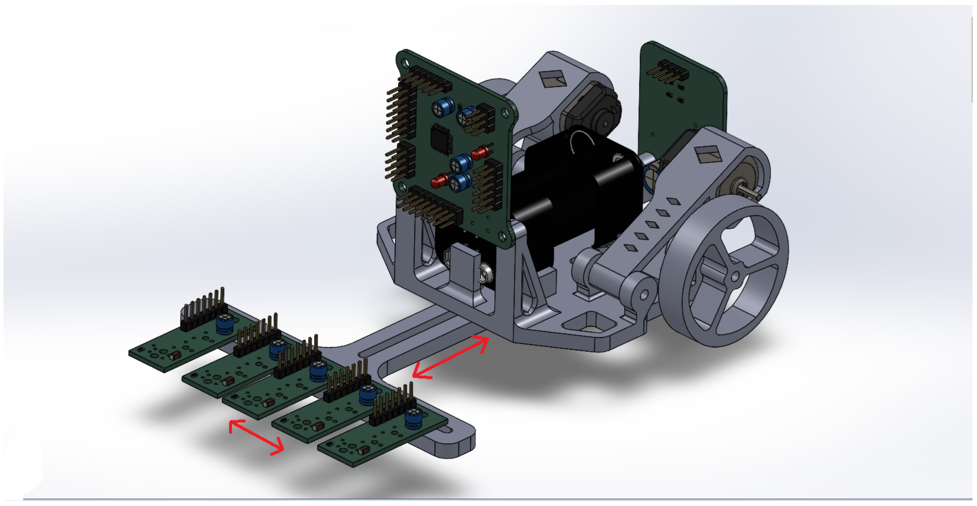

Figure 1: This is an example of a figure.