UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

ENEL373: Digital Electronics and Devices

University of Canterbury

This is an Awesome Title for Our First Report

Philip Brand *(15776928)*
Michael Brown *(48571923)*

April 19, 2025

Contents

1 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 test starts with 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.

2 Design Summary

The FPGA reaction stimulus test depends on four major components, those being the 8-digit counter, the ALU, the reaction countdown, and the output select and override. These components are shown in Figure 1. Each of these components are enabled and disabled via a finite state machine.

Figure 1: Overview of VHDL reaction timer structure

An 8-digit counter is used to track the reaction time of each new stimulus test. This is directly connected to a circular buffer and ALU system which stores the last three reaction times and calculates the minimum, maximum, and average. Both components, in addition to the random delay countdown component, are tied

to a 8x5-to-1 mux that is synchronised to the anode select, allowing each system to output the actively displayed digit. These outputs are then tied to an additional 8x5-to-5 mux that selects which component to display, with any additional text being added before being output to hardware.

3 Expanded Design Summary

3.1 Design Methods

The primary focus for the reaction timer project project was to develop VHDL code that is easy to maintain and develop. This led to a heavy focus on modular design and code reuse. Since the project was developed as a group, the other focus was for individual components to be developed and tested by individually, and then put together with minimal issues.

3.1.1 Diagramming

The primary program development method used in this project was to draw program or component flow diagrams before VHDL was written. These digrams detailed the communication between between individual modules, and outlined the overall program structure. Figure **??** shows one such digram outlining the ...

Philip - Michael, do you want to cover some of this?

3.1.2 Naming and Syntax Conventions

Another important aspect of the project ensuring high code quality, readability, and maintainability was strict rules around naming and syntax. This included consistent whitespace, indentation, case, and naming style. For example, all signals within the entity was postfixed with an underscore and whether that signal was an input or output. This can be seen in Listing 1 in Appendix A. This ensured when the components were used in the top-level Behavioural architecture with port mapping, it was easy to see which signals were inputs and outputs.

Add more examples here ...

3.1.3 Component Reuse

Philip - I believe you Boston were going to write this?

3.1.4 Programming Techniques

Another method used to implement the FPGA design was pair programming. This had to be done with care, as since the project was done in three-person groups, it would have been easy to leave one person out of the process and result in a lack of their understanding of the code. However, pair programming did help ensure that minimal time was spent debugging, as the observer frequently caught subtleties of VHDL that the programmer missed. Pair programming was used to develop the circular buffer, counters, and FSM.

Since the project the significantly modularised, it was important that individual components could be developed and tested without hindering the development of other components. The method used to achieve this was to use git branching and merging. Individual components were developed on separate branches, tested, and then merged into the main branch. This ensured that the main branch was always stable, and that broken changes could be easily rolled back.

3.2 Design Justification

1. it did what it was meant to do

2. it was not coded into an unchangeable spaghetti
3. it was incredibly modular

4 Module Testing

Philip - I believe this was going to the random number generator and then we can discuss maximal outputs? (to do on Tuesday?)

5 Design & Implementation Problems

1. Steven Weddell
2. Vivado
3. Vivado
4. Vivado
5. VHDL
6. Steven Weddell
7. Steven Weddell
8. Vivado
9. Vague Error messages
10. Decimal point issue
11. Randomisation delay insufficient
12. Not case sensitive code
13. Steven Weddell
14. Vivado

One issue that arose during the development of the seven segment display module which mapped a BCD input to a combination of segments on a display to represent that BCD number was that the provided mappings did not represent the numbers correctly. The solution approach was to first determine whether the BCD input or mapping was incorrect. This was done by tying the BCD input to the green LEDs on the FPGA. This asserted BCD input was correct. To rectify the incorrect mappings, individual segments on a board were tied to the FPGA user switches. Combinations of switches were tested, and when the segments represented a number, the combination of switches was recorded as a mapping.

However, once all the mappings were implemented, some numbers still failed to display correctly. @Michael - what appraches did we use to solve this issue?

6 Conclusions

7 Appendix A: Code Listings

```
-- Define module IO
entity counter_decade is
    Port ( EN_IN : in STD_LOGIC;
           RESET_IN : in STD_LOGIC;
           INCREMENT_IN : in STD_LOGIC;
           COUNT_OUT : out STD_LOGIC_VECTOR (3 downto 0);
           TICK_OUT : out STD_LOGIC);
end counter_decade;
```

Code Listing 1: Entity naming convention example.