Reaction Timer

Jordan Reeser

September 14, 2018

1 Executive Summary

In this lab, students were responsible for creating a reaction timer on the Nexys 4 DDR board. This reaction timer begins by displaying "hi" on the seven-segment display. Once the user presses the start button, a random amount of time between 2-15 seconds will pass before an LED turns on and a count in milliseconds (ms) begins and is displayed on the seven-segment display. When the user sees the LED turn on, they hit the stop button as quickly as possible stopping the timer, and their reaction time in ms will be displayed. To enable this operation, 6 modules were created/utilized inside of a top module that combined the function in a state machine. The top reaction timer, BCD_13bit, and normal counter modules were created by the designer, Jordan Reeser; the LFSR_fib168pi module was created by the instructor, Dr. Schubert; and the mod_m_counter, hex_to_sseg, and disp_mux modules were downloaded from the class textbook's companion site and created by the book's author, Pong Chu. The hex_to_sseg module was modified for this lab's specific needs. Utilizing these modules, a reaction timer was sucessfully implemented.

2 Board Setup

For this reaction timer, 3 buttons on the Nexys 4 DDR board are utilized as the ready, start, and stop buttons, and a board LED is usedd as the stimulus for reaction. For ease of use, the designer decided to use the 3 horizontal buttons on the board (btnL, btnC, and btnR) for ready, start, and stop, and picked the adjacent tricolor led (in green) as the reaction stimulus.

2.1 Ready, Start, and Stop

The designer decided to use the three horizontal buttons as ready, start, and stop since this setup appeared to be the most intuitive design.

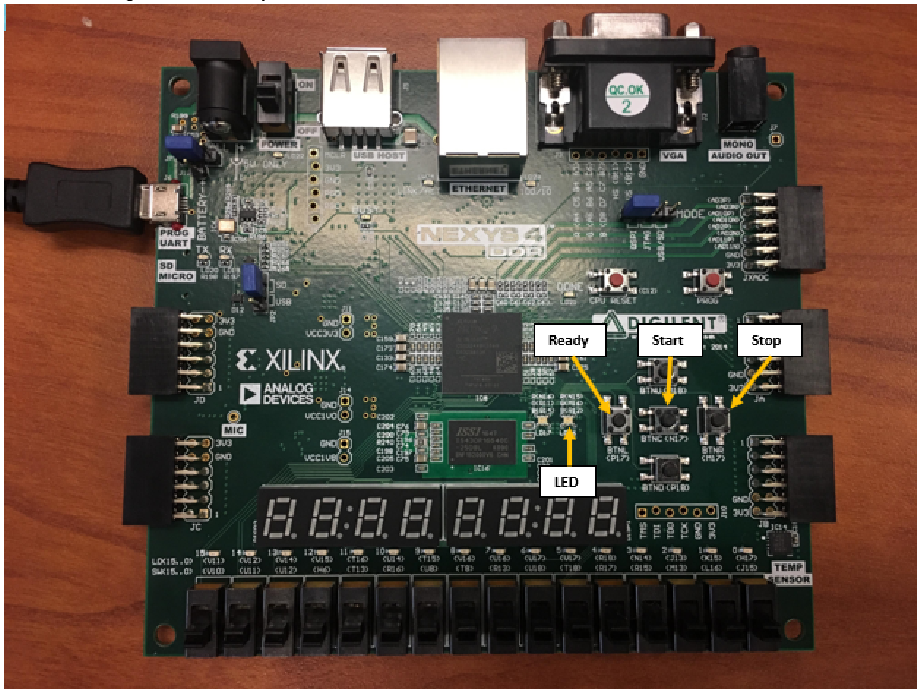
1. Ready: btnL
2. Start: btnC
3. Stop: btnR

2.2 LED

The tricolor LED adjacent to the ready, start, and stop buttons on the board was picked as the reaction stimulus. The LED will shine green when the reaction time begins.

1. LED: LD16 in green

Figure 1: Nexys 4 DDR board with buttons and LED labeled.



3 Functionality

To control the functionality of the project, a state machine was used with the following states: READY, START0, START1, STOP, ERROR1, and ERROR2.

3.1 READY

This state is similar to a reset state; it sets the initial conditions to prepare for the reaction timer to begin. In this phase, the seven-segment display shows the message "Hi" on the first two displays. All counts are held at 0, and the LED is off. The next state logic sets next_state to START0 when the start button is pushed

3.2 START0

This state is where the random delay before the reaction stimulus occurs. A value is picked from the random number generator module (LFSR_fib168pi) to be the random delay time; this variable is set to be 4 bits so it will pick a maximum value of 15, and an if statement ensures that the value is at least 2. The seconds are counted with a mod_m counter connected to a basic counter until the count value equals that of the delay value. During this entire state the seven-segment display will be blank. The next state logic sets next_state to ERROR1 if the stop button is pushed prematurely or to START1 when the count equals the delay time.

3.3 START1

The reaction time begins in this state. The stimulus LED is turned on, and the ms counter begins. The value of the ms counter is displayed on the seven-segment display. The next state logic sets next_state to ERROR2 if the user does not press the stop button before 1 second has passed or to STOP when the stop button is pressed.

3.4 STOP

This state simply stops the ms counter and displays the count value on the seven-segment display. This value should be the user's reaction time in ms. The next state logic sets next_state to READY is the user presses the ready (reset) button.

3.5 ERROR1

The first error state is the error from the user pressing the stop button before the reaction time (START1 state) begins. It displays "9999" on the display and returns to the READY state when the ready (reset) button is pressed.

3.6 ERROR2

This error occurs if the user does not press the stop button before the ms counter reaches 1000, or 1 second. The value "1000" is displayed and the design returns to the READY state when ready (reset) is pressed.

4 Modules

To implement the reaction timer design, the following modules were used:

1. reaction_timer
2. LFSR_fib168pi

3. mod_m_counter
4. counter
5. BCD_13bit
6. hex_to_sseg
7. disp_mux

4.1 reaction_timer

The reaction_timer module is the top module for the project. It connects the inputs and outputs to the board and includes all of the state logic and other module instantiation.

4.2 LFSR_fib168pi

This module is a linear-feedback shift register used to generate a random number for the delay. it uses a 28 bit seed and 4 keys that are the binary digits of pi. This module was created by Dr. Schubert.

4.3 mod_m_counter

This module is a simple mod m counter that is used to generate a tick after a specified amount of clock cycles. Two of these modules were used for the lab–one to generate a tick every second for the random delay time and the second to generate a tick every millisecond for the reaction time. This code was written by Pong Chu and levereged from his book *FPGA Prototyping by SystemVerilog Examples.*

4.4 counter

This module is a basic counter module. It increments count every time a tick is present. Reset sets the count value back to zero, and a stop input holds the counter's value. Two instances of this module were needed for the lab–one to count the seconds and one to count milliseconds. Both of these instances were connected to the corresponding mod m counter.

4.5 BCD_13bit

This module converts binary numbers to the corresponding decimal value to be displayed. The ms count value is fed into this module to display the count in decimal instead of hex.

4.6 hex_to_sseg

The hex_to_sseg module converts hex values fed into the module to binary values that will turn on the proper segments of the seven-segment display to show the hex value. This module was originally created by Pong Chu in his book *FPGA Prototyping by SystemVerilog Examples*, and was modified to include "h" and blank display values for use in this lab.

4.7 disp_mux

This module is used in conjunction with the hex_to_sseg module to display the values in the proper display units on the board. It multiplexes the display annodes to turn on one display at a time with the appropriate display value, and cycles through these displays at a rate that is too fast for a human viewer to discern. This module was created by Pong Chu and levereged from his book *FPGA Prototyping by SystemVerilog Examples*.

5 Code Appendix

Listing 1: Verilog code for implementing the reaction timer.

```verilog
//////////////////////////////////////////////////////////
// Author: Jordan Reeser
//
// Create Date: 09/03/2018 01:29:53 PM
// Module Name: Reaction_Timer
//
// Description: Module to instantiate a reaction timer
//
//////////////////////////////////////////////////////////


module Reaction_Timer(
    input logic start, stop, reset, clk,
    output logic led,
    output logic [7:0] an,
    output logic [7:0] sseg
    );

    logic [15:0] value;
    logic [31:0] display;
    logic [3:0] delay, thou, hun, tens, ones, new_delay;
    logic [12:0] count_ms, count_s;
    logic [27:0] seed = 28'b1110110001111110011101101100;
    logic tick_s, tick_ms, clr_ms, clr_s, hold, start_s;
```

```
// Set up state variables for program states
typedef enum {READY, START0, START1, STOP, ERROR1, ERROR2} TimerStates;
TimerStates state, next_state;

// Flip flop to move between states
always_ff @(posedge clk, posedge reset)
    if (reset)
        state <= READY;
    else
        state <= next_state;


// State logic
always_comb
begin
    case (state)
        READY:
        begin
        led = 0; // Initialize off (active high)
        clr_s = 1; // Hold count at 0
        clr_ms = 1;
        hold = 0;
        new_delay = 6;

        // sseg display "HI"
        value[7:4] = 4'hf; // f has been changed to display "h"
        value[3:0] = 4'h1;
        value[15:8] = 8'hee; // e has been changed to blank display

       // Next state logic
        if (start)
        begin
            next_state = START0;
            start_s = 1;
        end
        else
            next_state = state;
        end

        START0:
        begin
        clr_s = 0; // Allow counter to start
        hold = 0;

        // Blank display on sseg
        value = 16'heeee;
        if (start_s)
            begin
            new_delay = delay;
            start_s = 0;
            end
        else
            new_delay = new_delay;

        //Delay value has to be at least 2
        if (new_delay < 2)
            new_delay = new_delay | 4'b0010;

        // Next state logic
        if (stop)
            next_state = ERROR1;
        else if (count_s == {9'b0, new_delay})
            next_state = START1;
        else
            next_state = state;
        end

        START1:
        begin
        led = 1; // Turn on LED
        clr_ms = 0; // Start reaction counter
        clr_s = 1; // Clear delay counter
        hold = 0;

        value [15:0] = {thou, hun, tens, ones}; // Display count on sseg

        // Next state logic
        if (count_ms == 1000)
            next_state = ERROR2;
        else if (stop)
            begin
            next_state = STOP;
            hold = 1; // Hold count value
            end
        else
            next_state = state;
        end

        STOP:
        begin
        // Display count
        hold = 1;

        // Next state logic
        if (reset)
            next_state = READY;
        else
            next_state = state;
        end

        ERROR1:
        begin
        // Display "9999" on sseg
        value = 16'h9999;

        // Next state logic
        if (reset)
            next_state = READY;
        else
            next_state = state;
        end

        ERROR2:
        begin
        // Display "1000" on sseg
        value = 16'h1000;

        // Next state logic
        if (reset)
            next_state = READY;
        else
            next_state = state;
        end
    endcase
end

// Turn off second set of sseg
assign an[7:4] = 4'b1111;

LFSR_fib168pi rand_gen(.clk(clk), .reset(reset), .seed(seed), .r(delay)); //
mod_m_counter #(.M(100000000)) rand_delay(.clk(clk), .reset(reset), .max_tic
mod_m_counter #(.M(100000)) react_time(.clk(clk), .reset(reset), .max_tick(t
counter#(.SIZE(13)) s_count(.clk(clk), .tick(tick_s), .reset(reset | clr_s),
counter#(.SIZE(13)) ms_count(.clk(clk), .tick(tick_ms), .reset(reset | clr_m
BCD_13bit ms_binary_to_dec(.in(count_ms), .thousands(thou), .hundreds(hun),
hex_to_sseg sseg_for_disp0(.hex(value[3:0]), .dp(1), .sseg(display[7:0])); /
hex_to_sseg sseg_for_disp1(.hex(value[7:4]), .dp(1), .sseg(display[15:8]));
```

```
    hex_to_sseg sseg_for_disp2(.hex(value[11:8]), .dp(1), .sseg(display[23:16]))
    hex_to_sseg sseg_for_disp3(.hex(value[15:12]), .dp(1), .sseg(display[31:24])
    disp_mux sseg_control(.clk(clk), .reset(reset), .in3(display[31:24]), .in2(d

endmodule
```