Experiment 11. Reaction Timer: Part 2. Response Time

Masud ul Hasan · Aiman El-Maleh · Ahmad Khayyat – Version 151, 29 October 2015

Table of Contents

1. Objectives

- Learn how to design a saturating BCD counter.
- Learn how to measure user response time.
- Finalize the reaction timer experiment.

2. Materials Required

- An FPGA prototyping board.
- Design and simulation software tools.
- Seven-segment display interface module.

3. Background

In the previous experiment, we have built a random delay counter, which waits for a random number of seconds before turning on an LED for the player to see and react by pushing a push button. Therefore, we need now to design the part that would measure the time between the LED turning on and the player pushing the push button. Then, we need to classify how fast the response of the player was.

3.1. Design Overview

The Functional Block Diagram of the Final System figure shows the functional design of the system. The white area highlights the blocks you are going to build in this part of the experiment.

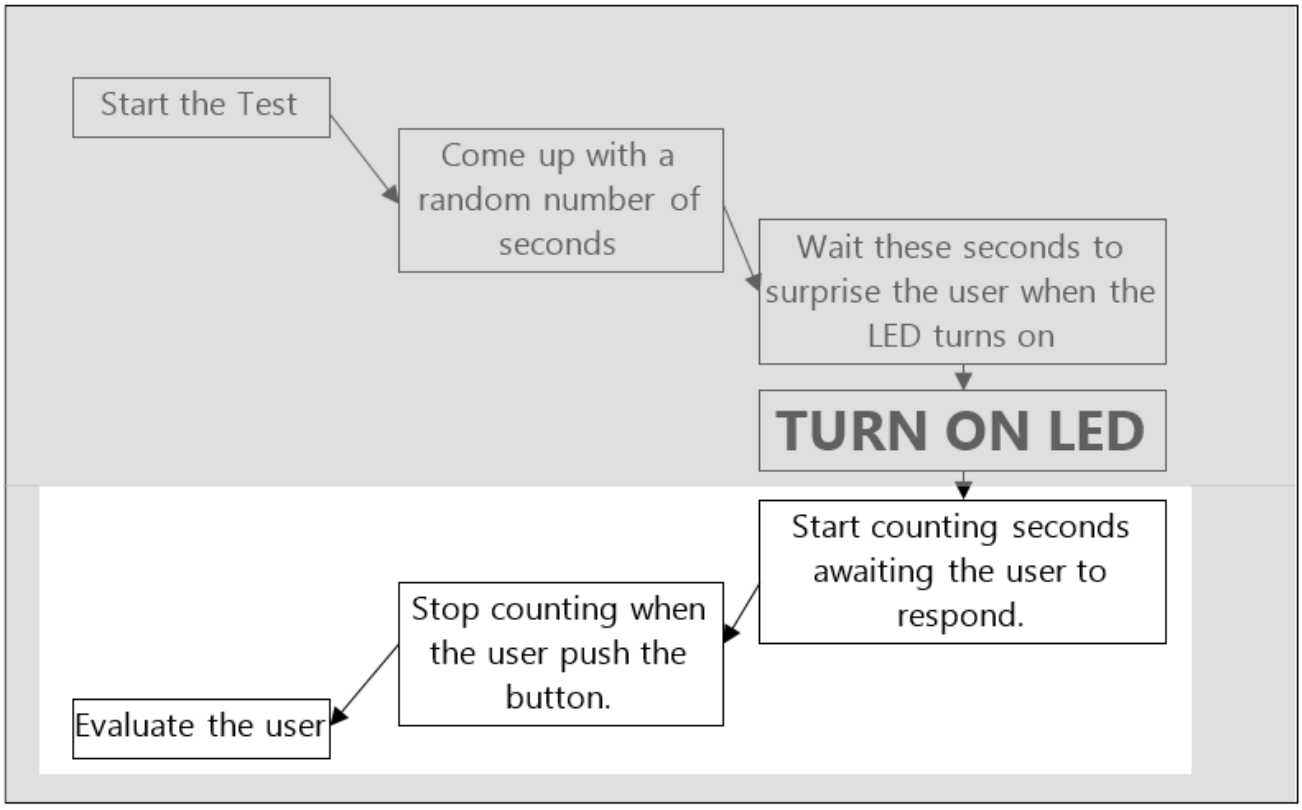

Figure 1. Functional Block Diagram of the Final System

We need to design a component that would count the number of seconds between the LED turning on and the time at which the push button is pressed by the user.

To simplify this task, we will measure the response time by counting using *binary coded decimal* (BCD) numbers rather than binary numbers, which means we can display the value of the counter to the player easily.

We will use a *saturating counter*, which is a counter that, upon reaching a preset maximum value, will stop counting rather than going back to zero. The counter is driven by a `two_khz` module that generates a 2-kHz signal; as a result, the counter has a half-a-millisecond accuracy, i.e. 500 µs. Once the counting stops, the output of the counter will be fed into a module that would classify the response time as *fast*, *medium*, or *slow*.

This design is illustrated in the Component Block Diagram figure below. In this experiment, we will implement these two components shown in the figure, and will integrate them with the previously implemented components to obtain a complete, functional system.

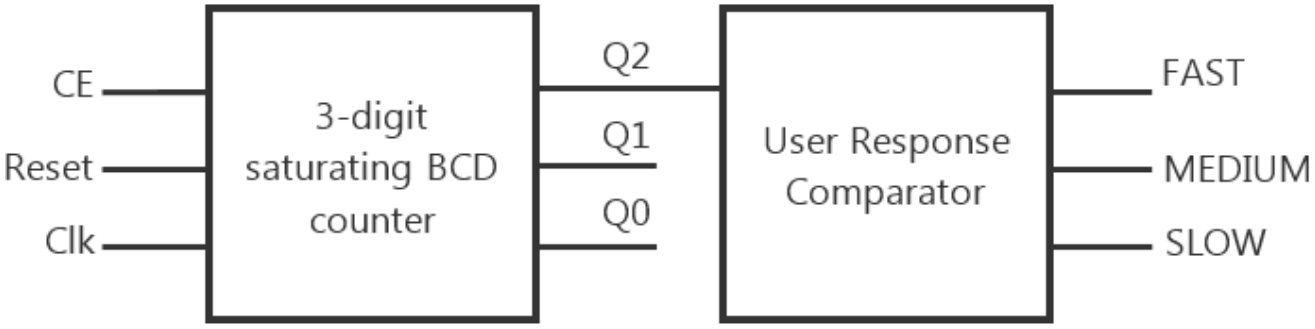

Figure 2. Component Block Diagram

3.2. Building a Saturating BCD Counter

Cascaded BCD Counters

Recall from experiment 9 (*Building a Digital Timer*) that a BCD counter is a mod-10 counter, and that BCD counters may be *cascaded*, such that when a digit counter reaches the maximum digit value of 9, it resets back to 0, and increments the counter of the next more significant digit. For more details, refer to experiment 9 manual.

The Verilog Module for a Two-Digit Saturating BCD Counter below shows how to build a two-digit saturating BCD counter, which will count from zero up to 99, and stop at 99 until it is reset using the `reset` input signal. Once this counter reaches the value of 99, the only way to get it out of that state is to reset it.

Observe how the two-digit counter in this example is built by cascading two one-digit BCD counters.

Verilog Module for a Two-Digit Saturating BCD Counter

```
module bcd_counter_1d (
    input clock, reset, enable,
    output eq9, reg [3:0] q);

    assign eq9 = q[3] & q[0];
    always @(posedge clock)
    if (reset)
        q <= 4'b0000;
    else if (enable)
        if (eq9)
            q <= 4'b0000;
        else
            q <= q + 1;
endmodule

module saturating_bcd_counter_2d (
    input clock, reset, enable,
    output eq99, [3:0] q1, q0);

    wire eq9_0, eq9_1, enable_0, enable_1;

    assign eq99 = eq9_0 & eq9_1;
    assign enable_0 = enable & ~eq99;
    assign enable_1 = enable_0 & eq9_0;

    bcd_counter_1d bcd_0 (clock, reset, enable_0, eq9_0, q0);
    bcd_counter_1d bcd_1 (clock, reset, enable_1, eq9_1, q1);
endmodule
```

In this experiment, we will need a three-digit saturating BCD counter.

3.3. User Response Comparator

The *response comparator* is the component that we will use to compare the value of the counter at the time of the user response against a standardized table that maps the response time to a response speed class (*fast*, *medium*, or *slow*).

Once the user presses the *measure* push button, the counter should stop counting, and the response comparator circuit will check the value of the counter and evaluates the user response time.

The comparator will generate the following three signals to categorize the user response time:

Fast	if the user response time is less than 400
Medium	if the user response time is 400 or greater but less than 800
Slow	if the user response time is between 800 and 999

Combinational logic can be designed to generate the three signals. Two K-maps can be used to derive the equations for the *medium* and *slow* signals. Then, the equation for the *fast* signal can be derived based on the other two signals.

> The input of this circuit is the most significant digit (MSD) only of the three-digit saturating BCD counter (i.e., `q2`).

---

4. Tasks

4.1. Design a Three-Digit Saturating BCD Counter

1. Write a Verilog module for a three-digit saturating BCD counter using the following module declaration:

```
module saturating_bcd_counter_3d (
    input clock, reset, enable,
    output eq999, [3:0] q2, q1, q0);
```

2. Simulate your three-digit saturating BCD counter and verify its correct functionality.

4.2. Build the Response Time Comparator

The *response time comparator* circuit determines whether the user's response is fast, medium, or slow. See the User Response Comparator section for details.

1. Using two K-maps, derive the equations of the *medium* and *slow* signals. Then, deduce the equation of the *fast* signal.
2. Write a Verilog module to model the response time comparator circuit using the following module declaration:

```
module response_speed (
    input [3:0] q,
    output fast, med, slow);
```

3. Simulate the `response_speed` module and verify its correct functionality.

4.3. Integrate the Reaction Timer Modules

The Reaction Timer Verilog Module, `reaction_timer`, below integrates the following modules: - `saturating_bcd_counter_3d` - `response_speed` - `delay_counter`, from experiment 10 (part 1 of this experiment).

along with other modules.

Reaction Timer Verilog Module

```
module reaction_timer (
```

```
    input clock, reset, start, measure,
    output led, [7:0] seg, [3:0] an);

    wire [3:0] q2, q1, q0;
    wire reset2 = reset | start;
    wire clock_2khz, measure_q, error, error_q, eq999, fast, med, slow;

    delay_counter m1 (clock, reset, start, led);
    two_khz m2 (clock, reset2, clock_2khz);

    assign error = ~led & measure;
    dff m3 (clock, reset2,  measure, 1'b1, measure_q);
    dff m4 (clock, reset2, error, 1'b1, error_q);

    assign enable = clock_2khz & led & ~measure_q;
    saturating_bcd_counter_3d m5 (clock, reset2, enable, eq999, q2, q1, q0);
    response_speed m6 (q2, fast, med, slow);
    display7seg m7 (clock, reset2, q2, 4'b0000, q0, q1, seg, an,
                    slow, med, fast, error_q, measure_q, 1'b0, 1'b1);
endmodule
```

For the `two_khz` module, you can use the following Verilog Module for Generating a 2 kHz Signal.

Verilog Module for Generating a 2 kHz Signal

```
module two_khz (
    input clock, reset,
    output clock_2khz);

    reg [15:0] counter;
    assign clock_2khz = (counter == 16'hc34f);
    always @(posedge clock)
    begin
        if (reset || clock_2khz)
            counter <= 0;
        else
            counter <= counter + 1;
        end
endmodule
```

For the `display7seg` module, use the `DISP7SEG.v` file that will be given to you in the lab.

1. Implement the `reaction_timer` module on the FPGA.
   a. The signals `seg` and `an` should be connected to the seven-segment display unit.
   b. The `clock` signal should be connected to the board's system clock (pin `V10` on the FPGA).
   c. The `start` and `measure` inputs are the two push buttons that the user will use.
2. Test the correct operation of your circuit.
   a. Press the `reset` button.
   b. Press the `start` button, which will load the down counter with a random number of seconds between 0 and 3. Once this random number of seconds elapses, the LED will light. Once the LED signal is 1, the counter will start counting and will stop once you hit the `measure` button.
   c. Press the `measure` button as soon as you see the LED light. Before pressing the `measure` button, you will see the count displayed on the seven-segment display unit. Once you push the `measure` button, the classification of your speed will be displayed on the seven-segment display unit. If you press the `measure` button before the LED is on, an error message will be displayed.
3. Answer the following questions:
   a. Why are the `measure` and `error` signals stored in flip-flops (`dff`) to obtain `measure_q` and

`error_q`?

b. Why is the `enable` signal, that is used to enable the counter, implemented as: `enable = clock_2khz & led & ~measure_q` ?

c. Why is the `reset2` signal, which is used for resetting all components except the `delay_counter`, implemented as: `reset2 = reset | start`?

---

5. Grading Sheet

Task	Points
Design a three-digit saturating BCD counter + Simulation	15 + 5
Build the response time circuit + Simulation	15 + 5
Integrate the reaction timer modules	35
Lab notebook and discussion	25

Version 151
Last updated 2015-11-23 14:23:16 AST