Project 3 Basic Logic Circuits & Combinational Logic Circuits

Introduction

This project provides further experience with logic circuits and presents several circuit requirements that are described with higher-level, natural-worded descriptions. Your job is to create circuits that behave according to the descriptions.

When starting circuit designs from higher-level, purely behavioral descriptions like those presented below, it's a good idea to identify and follow a set of good design practices. In general, good design practices follow several steps: first, be sure you clearly understand the design intent, and exactly what is being asked; second, cast the worded description into formalisms like a block diagram and truth table that shows all inputs & outputs and their functional relationships; third, capture a circuit based on the formalisms (that is, write a Verilog description); and fourth, implement the circuit and verify its performance. Each of the requirements below presents problems that are involved enough that all of these steps should be followed.

Before you begin, you should:

- Have the Xilinx Vivado tool installed;
- Have a Boolean board and know how to program it;
- Know how to start a Vivado project and construct a basic logic circuit;
- Understand logic gates and basic logic circuits.

After you are done, you should:

- Be comfortable creating new designs in Vivado;
- Be able to translate a worded problem description into Verilog;
- Be able to write more complex Verilog descriptions;
- Be able to analyze more complex descriptions, and design circuits to implement the behavior;

Background

In this project, you will take some significant steps towards designing your own digital logic circuits. The design process involves many steps that are best learned through doing actual designs, but some fundamental background knowledge will help that process – read the theory topics! Each of the problems in this project describes a two-state output (either on or off, true or false, 1 or 0, etc.) that is a logic function of some number of two-state (or binary) inputs. It follows that each of these problems can be represented using a truth table, and the truth table forms the specification for a circuit. The truth table exactly specifies the behavior of a circuit, but not the structure. Before a physical circuit can be constructed, its structure must be defined.

Requirements

☑ 1. Drive an RGB LED using pushbuttons

Connect **3 pushbuttons** to the **3 LEDs (Red, Green, and Blue)** that are included in one of the RGB LEDs. Drive the individual colors only when the corresponding pushbutton is pressed. Notice how you can produce a variety of colors by pressing multiple buttons simultaneously!

Note that Blue, Green, and Red LEDs are available in the RGD LEDs as individual signals (check the **booleanxdc.txt** file).

☑ 2. RGB LED Controller Using Switches

Illuminate a **<u>red</u> LED** only when **exactly two** of the first four slide switches **(SW0, SW1, SW2, SW3)** are set to **"1"**. Illuminate a **<u>blue</u> LED** when **exactly three** of the next four switches **(SW4, SW5, SW6, SW7)** are set to **"0".**

☑ 3. Design odd-number and even-number detectors

Implement a circuit that illuminates a **<u>red</u> LED** when **an odd** number of the **eight slide switches** are set to **"1",** and illuminates a **<u>green</u> LED** when **an even** number of **slide switches** are set to **"1".**

☑ 4. Enhance the even number detector with pushbutton inputs

Illuminate a **<u>blue</u> LED** when the **even <u>green</u> LED** in the circuit above is **illuminated, and 1 or 3** of the **pushbuttons** are pressed.

***** For requirements 2 & 3, you can find example videos of tasks implemented on a Blackboard (notBoolean Board, but similar) in the following links:

https://www.youtube.com/watch?v=O8V1ILoQyJ8
https://www.youtube.com/watch?v=1AcZvzaGROk