Project 4 Basic Combinational Building Blocks

Multiplexors, Decoders, Encoders

Introduction

This project presents the design of logic circuits that are commonly used as components in larger, more complex circuits. Althoughthey can be used by themselves in simpler settings, these components are more typically thought of as basic building blocks in larger, more involved circuits.

Although it is easier, simpler, and more natural to define circuits behaviorally, any given circuit description must be translated into a detailed structural form before it can be implemented. For many years, the main job of a design engineer was translating loose behavioral descriptions into detailed structural specifications. In more recent times, logic synthesizer tools took over that task. Today, virtually all digital circuit designs use logic synthesizers, and all modern design tools (like Vivado) include synthesizers as essential components.

Before you begin, you should:

- Be able to find a minimum circuit for any given behavioral description;
- Understand the general design flow for designing any digital circuit;
- Be comfortable using Vivado and the Boolean board to define and implement designs.

After you're done, you should:

- Understand the design and use of multiplexors, decoders, and encoders;
- Be comfortable defining slightly larger circuits;
- Be comfortable using the simulator for routine circuits.

Requirements

Each of the requirements involves the design, simulation, and implementation of an independent combinational circuit. If you have **multiple source files** and/or test benches in a single project, you can select which one is "**active**" by **right-clicking on the name** and selecting "**set as top**" from the pull-down menu. Three small boxes in a triangle pattern next to a source file name indicate which source file isactive. When you run a process like synthesis or simulation, the tools will use the currently active source file(s). Before clicking **Run Behavioral Simulation** , right-click on the simulation source file you want to simulate and select Set as Top.

Note: You can follow the **tutorial and simulation** files to help you complete your tasks.

☑ 1. Multiplexor

(a) *Design* and *simulate* a **4:1** multiplexor, and then *implement* it in the Boolean board. Use **2 slide switches SW0 and SW1 to select** the multiplexor input, connect the **4 slide switches SW10, SW11, SW12, and SW13 to the multiplexor input** signals, and connect the **output to an LED**. Verify the circuit works correctly.

(b) Next, modify your code to create a **4:1** multiplexor where the **inputs** and **output** are **4 bits each**. Connect the inputs of thefour multiplexor channels to four unique slide switches (**16 switches**), connect the **outputs** to **four LEDs**, and control the **select** signals with two **push buttons**. *Simulate* the circuit to verify that it works correctly, and then *program* it into your Boolean board.

Hint: instead of simulating every single possible input, assign an arbitrary value to each data input. Then just show that the selectsignals change the output appropriately. E.g.,

```
reg [3:0] I0 = 4'h1;
reg [3:0] I1 = 4'h3;
reg [3:0] I2 = 4'hD;
reg [3:0] I3 = 4'hF;
```

☑ 2. Decoder

Implement a **2:4** decoder circuit that based on the binary value of the input, **enables one of four LEDs when a corresponding pushbutton is pressed**. Use **two slide switches as inputs.**

The decoder's outputs are AND'ed with push buttons. The AND gate outputs should drive four LEDs. When the circuit is implemented, the **four pushbuttons should turn on the LEDs, but only if the slide switches have been set to enable that channel.**

Note: You can find example video of the task implemented on a Blackboard (not Boolean) in the following link:
https://www.youtube.com/watch?v=dnK9ccSNm4g

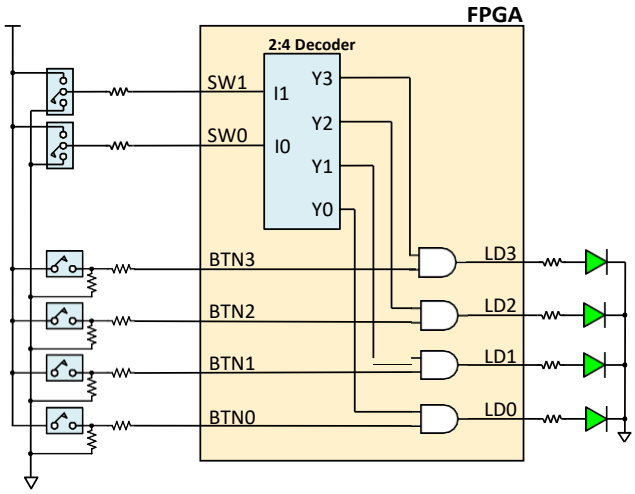

Figure 1. Circuit schematic

☑ 3. Encoder

Create a Verilog description of a **4:2** priority **encoder** and a **test bench that checks for all possible input patterns**. *Simulate* the encoder to verify that it works properly. You do **not** need to implement this circuit.