California State University, Long Beach
Computer Engineering and Computer Science Department
CECS 361 - Computer Logic Design II
Lab 2 - 3-SAT Detection Circuit

Objectives:

- To implement and test a simple 3-Sat detection circuit and display it on the Nexys A7 board
- To learn how to use the Nexys A7 basic I/O devices
- To familiarize students with implementing HDL designs for a target technology

Theory:

This lab assignment will help solidify the concepts covered in "Topic 2: Introduction to Design Verification." The Nexys A7 trainer board comes equipped with a variety of input and output peripherals such as buttons, switches, LEDs etc. You are to utilize these to output the results of your design.

Assignment:

You are to implement a simple 3-SAT detection circuit that will indicate whether a given CNF function is satisfiable or not. The CNF functions are to be defined as modules, where the number of input variables is 3. If the given function is satisfiable (SAT), a green light on your board will be turned on. If the given module is not satisfiable (UNSAT), a red light will be turned on. Your detection circuit must be able to check all the possible input/output values (3 variables ➔ 8 cases). If the output is '0' for all the cases, then the given CNF function is UNSAT. Likewise, if the output is '1' for any case, then the CNF function is SAT. If the function is found to be SAT, you must report one of the satisfiable assignments on the onboard LEDs. To do so, you must use three of the LEDs to be able to express the 3-variable solution.

Steps:

1. Create a project and name it lab2.
2. Select the Nexys A7-100T (xc7a100tcsg324-1) board when prompted to select a target device.
3. Start by adding the CNF module that has been provided for you on BeachBoard (CNF.v). This module contains 4 different CNF functions that can be independently selected via the *sel* input, which controls the select line of an internal 4-to-1 multiplexer. This will allow you to test 4 different CNF functions on your board.

4. Create the function-solving module and name it *Solver.* This module will contain the logic for iterating through all the possible input values to CNF, and will check if the current input value is a solution to the function. Also, this module will provide the signals that will control an RGB LED (to display either Red for UNSAT or green for SAT), and 3 of the onboard LEDs (to display one of the expressions that satisfies the function). The logic within this module must be synchronous (clocked), so as to simplify your design. The following are the global inputs/outputs to this module:
    a. clk: 1-bit input that will be driven by the board's 100Mhz clock
    b. reset: 1-bit input that will serve as the reset to the synchronous logic
    c. in: 1-bit input that will be driven by the output value of the CNF function
    d. RGB: 3-bit output that will drive one of the onboard RGB LEDs.
    e. LED: 3-bit output that will drive 3 different LEDs to display the solution.
    f. solution: 3-bit output that will drive inputs A, B, and C of the CNF module.

```
module Solver(
    input clk,
    input reset,
    input in,
    output reg [2:0] RGB,
    output reg [2:0] solution,
    output reg [2:0] LED
    );


    always@(posedge clk, posedge reset) begin
        if(reset) begin
            solution <= 3'b0;
            LED <= 3'b000;
            RGB <= 3'b001;
        end
        else if(in) begin
            // Insert logic here for the case when a solution is found (update LEDs)

        end
        else begin
            // Insert logic here to iterate to the next test expression
            // when a solution has not yet been found.
            // Hint: update LEDs here as well

        end
    end
endmodule
```

Figure 1. Sample code that you can use to build your Solver.v module

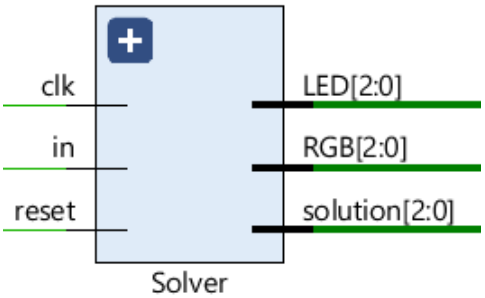

Figure 2. Block Diagram of solver

5. Create a testbench for the *Solver* module to verify its functionality through simulation.
6. Create a top module to connect the *Solver* module with the *CNF* module. The top module should have the following inputs/outputs
    a. clk: 1-bit input driven by the onboard 100Mhz clock
    b. reset: 1-bit input that will serve as the reset to the synchronous logic. Driven by an onboard switch.
    c. sel: 2-bit input that will drive the select line of the CNF module to select a function. Driven by 2 onboard switches.
    d. RGB: 3-bit output that will drive one of the onboard RGB LEDs.
    e. LED: 3-bit output that will drive 3 different LEDs to display the solution.

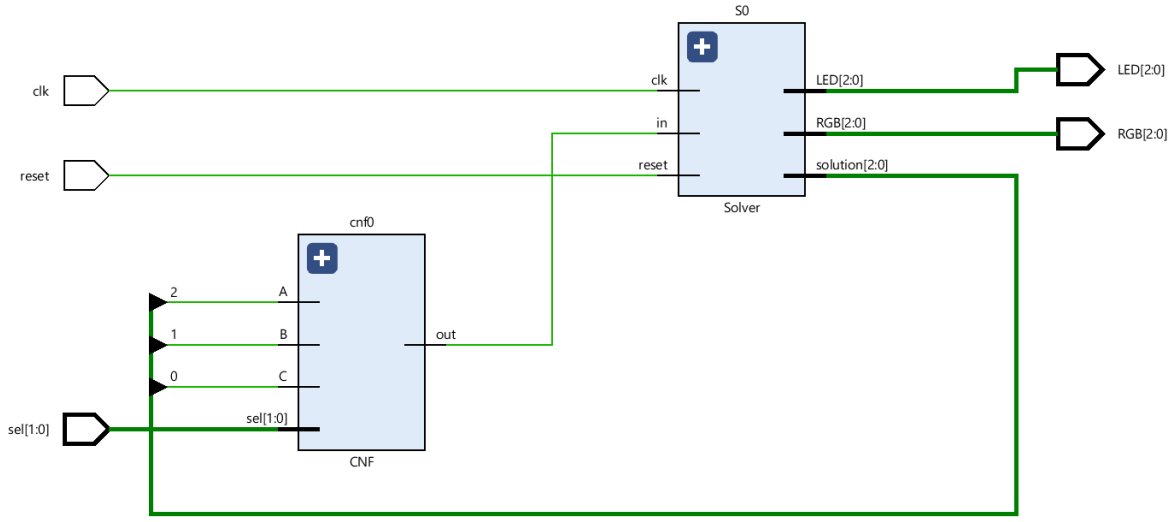

Figure 3. Diagram of the top level connections

7. Create a testbench for the top module to verify its functionality through simulation.
8. Add the NexysA7-100T-Lab2.xdc constraints file from BeachBoard into your project. The constraints file maps the various inputs and outputs of your top module to the components for which they are meant to drive. The constraints file has been filled out for you in accordance to the port namings that have been suggested. If you have different identifiers for your global inputs/outputs, you will need to make the necessary changes to the constraints file to ensure the correct mappings. To add the constraints file, click on **Add Sources** in the Flow Navigator and select **Add or create Constraints**. Then, add the file into your project by clicking **Add Files**, find its location in your drive, and selecting it. Click Finish when done.

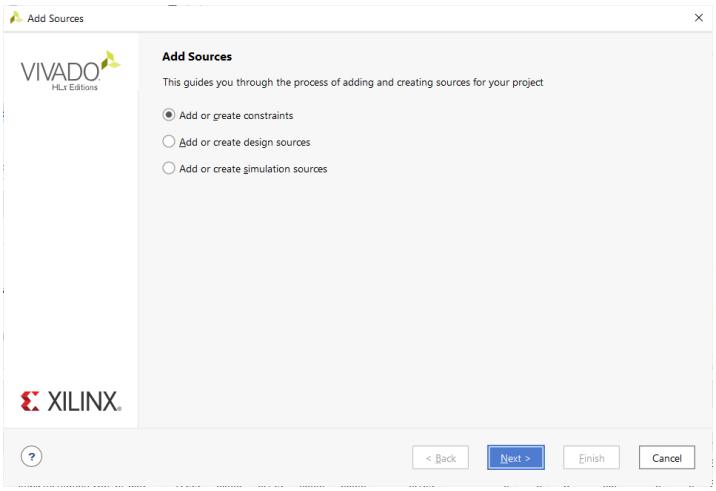

Figure 4. In the Add Sources pane, select Add or create constraints

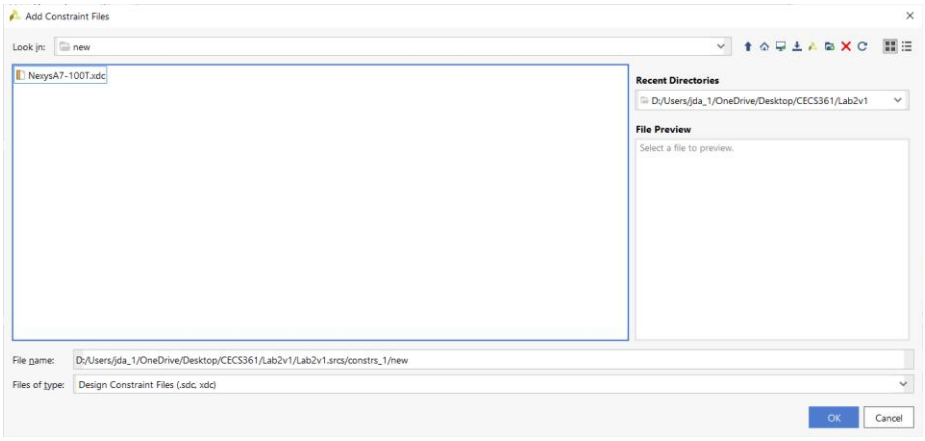

Figure 5. Find the constraints file in your drive and select it.

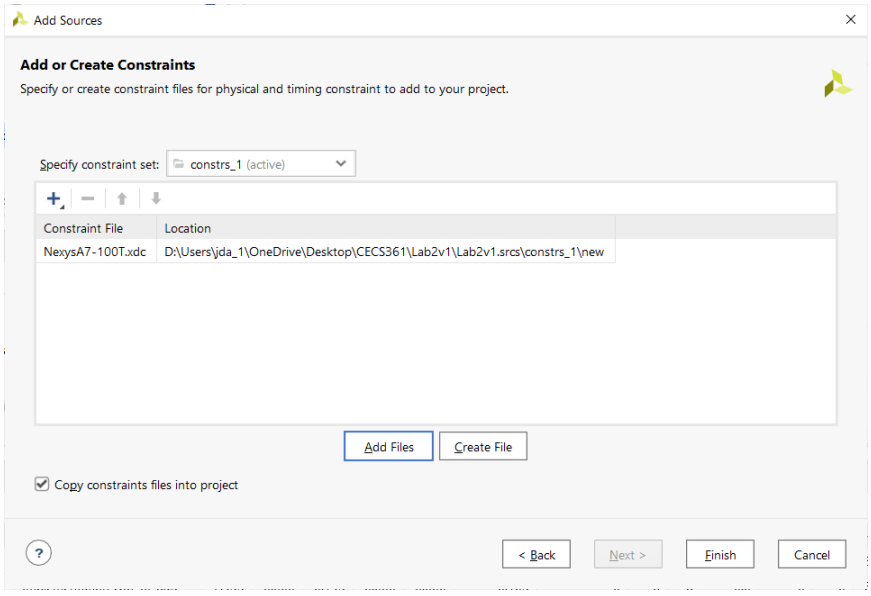

Figure 6. Click Finish to add the constraints file to your project.

9. Generate the bitstream. Under the Flow Navigator pane, click on **Generate Bitstream** to generate the bitstream that will map your design to the FPGA. You will be prompted to run the synthesis and implementation steps that are required to generate the bitstream. Leave all the prompts with the default selections and continue.

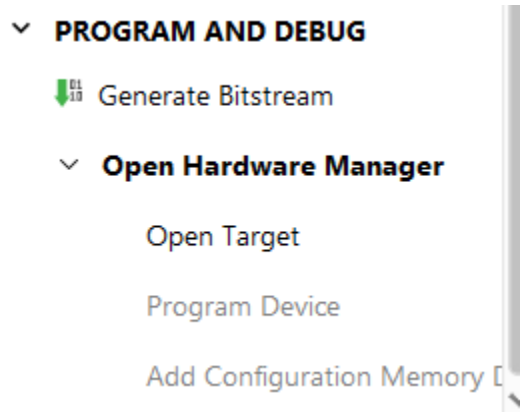

Figure 4. Generate Bitstream under the Flow Navigator Pane

10. Program your device. Connect your Nexys board to a USB port on your computer and power it on. Connect to the board by clicking on **Program and Debug > Open Hardware Manager > Open Target > Auto Connect** to automatically connect to your board. Once connection has been achieved, click on **Program Device** to program the board with the bitstream that you have generated. Observe behavior of your design on the board.

Lab Deliverables:

Submit a brief report including the snapshots of your simulation waveforms and the snapshots of your running board. Also, you need to demonstrate your work and the completed steps in the demo time. Upload your solutions (one .zip file) including a lab report (.pdf file) and CNF module, solver module and testbench, and top module and testbench (.v files) to the Dropbox labeled Lab 2.

Submission (Report & Verilog files) due date: Oct 5th 2020, 11:30PM

Demo: Oct 6th 2020, 6:00PM - 7:15PM & Oct 8th 2020, 6:00PM - 7:15PM