Exp 2: Full Adder - Reduced SOP Form

Objectives:

- To receive more exposure to the Xilinx Design Methodology
- To practicing reducing your circuit

Background: One of the previous lab activities was to design a full adder in both SOP and POS form. Now you'll design a full adder in reduced SOP form and verify that it's operation is the same as your non-reduced full adder.

Assignment: Design both a full adder in reduced SOP form (see if you can use XOR gates to reduce the equation for the sum). Use the simulator to verify that your FA implementations works properly. You do not need to test your full adder on the board.

Questions:

1. Was there any advantage to using the reduced form of the FA equation when implementing your equations? Briefly explain.

2. Since a FA does everything a HA does and more, briefly describe why you feel that HAs are still hanging around in digital design land.

Exp 3: Implementing and Simulating Functions Forms

Objectives

- To analyze a digital circuit implemented on a programmable logic device
- To reduce a function to generate several useful function forms
- To use Verilog Testbenches and Xilinx ISim to verify Verilog model correctness

Procedures

Procedure Overview: In this experiment, you are provided with a programming file that contains the implementation of a function which is downloaded to you development board. Your task is to analyze this circuit in order to generate a truth table model of the circuit. You'll then implement this circuit using both the NAND/NAND and NOR/ NOR circuit forms to prove that these two forms are functionally equivalent.

1. Download the appropriate programming file *exp3_function.bit* for your Basys board from PolyLearn.

2. Figure 10 shows a black-box diagram of the circuit implemented in the FPGA while Table 0.1 shows the switch to signal mappings for the four input signals on the development board. Using the switches and LEDs, analyze the input / output behavior of the circuit programmed in the FPGA. Create a truth table with your findings.
   - The up and down positions of the switch represent 1's and 0's, respectively.
   - The lit and unlit conditions of the left-most LED on the development board represent 1's and 0's of the function's output, respectively.

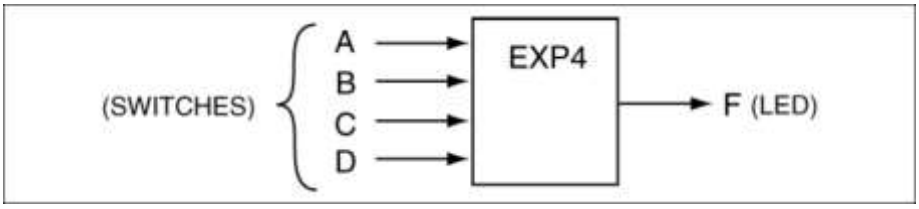

Figure 1: Black-box diagram of circuit represented by the provided programming file.

Development Board	A	B	C	D	F
Basys	SW7	SW6	SW5	SW4	LD7

Table 0.1: Switch and output mappings for the development boards.

3. Reduce the function described in the truth table and write down the NAND/NAND and NOR/NOR equations of the circuit.

4. Generate a Verilog model that models both the NAND/NAND and NOR/NOR forms of the circuit (your entity should have four inputs A, B, C, and D and 2 outputs (myNANDNAND and myNORNOR)).

5. Test your Verilog model using the ISim simulator.

6. Implement your Verilog model on the Basys board (map your inputs to switches and outputs to LEDs) and verify that it has the same output as the original circuit you analyzed and the Verilog model you simulated in the previous step.

Questions

1. Write down the reduced SOP, reduced POS, NAND/NAND and NOR/NOR forms of the circuit designed in this experiment. How many logic gates are used in each form? For this question, assume that the inputs to the system are all non-inverted; therefore, include the number of inverters required in your gate count. Assume you have the ability to use any type of gate with any number of inputs (such as 2,3,4-input NANDs, NORs, ANDs, ORs, *etc.*).

2. Speculate on why integrated circuit manufacturers don't generally make items such as 7-input NAND gates.