Engr433 Lab #5

VHDL Implementation Using an FPGA – Basic

Name _____

Partner _____ Grade _____/10

Objectives

- Learn how to implement a circuit in a Xilinx FPGA using VHDL
- Refine your
  - Design and functional partitioning strategies;
  - Testing approach;
  - Debugging skills.

References

- This handout;
- Xilinx documents on the class web page including *VHDL Entry Using ISE 14.7 Tutorial*

Tools Required

- Linux computer with Xilinx software;
- FPGA boards with download and power cables;
- Instruments as necessary.

Design Flow

The design flow for today's lab is

1) Confirm understanding of the problem statement;
2) Partition the design into subsystems;
3) Create a system block diagram;
4) Implement the design using VHDL;
5) Synthesize, place, route, and create a bit-map file using Xilinx ISE;
6) Download the bit-map file to the FPGA board;
7) Test and debug as needed.

Task 1

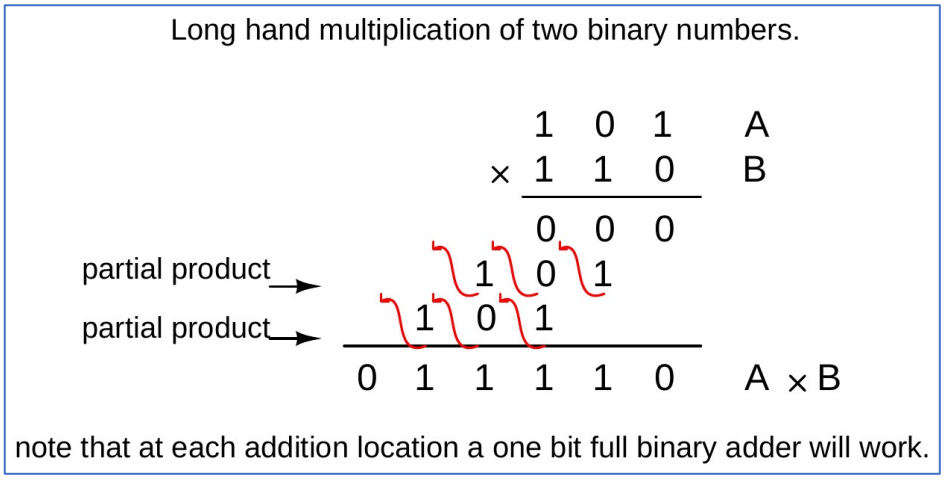
Create a circuit where Led0 lights up when sw0 or sw1 are asserted but not if both are asserted. Write a VHDL description, synthesize, and download to the FPGA, and verify correct operation.

Task 2

Design a circuit that will multiply two 4-bit, unsigned, binary numbers and display the result in binary on 8 LEDs. Mathematically, P = A * B where A will be switches 3 - 0 and B will be switches 7 - 4. Product P will be Leds 7 - 0 where the least significant bit is Led0. Note that this circuit only uses combinational logic and for this lab you are **not allowed** to use math operators in your VHDL description, only logical operators.

Refer to figure at the right to remember how to do manual multiplication.

A one-bit full adder circuit has bits A and B and Carry_in as inputs. Outputs are Sum and Carry_out.



A circuit design strategy is to create two module types that can be repeatedly used to implement what we call an array multiplier. Each module will have a full adder circuit plus additional logic. To describe the full adder and other logic in a module, use what are called behavioral-type VHDL statements, i.e. use basic logic operators (AND, OR, NOT, XOR, etc.) to describe the operation of the module. Within the module you will not be placing instances of these logic operators as if they were logic parts but simply writing VHDL statements expressing the desired logic. Then, use structural-type VHDL to connect these modules together for the multiplier circuit.

To Turn In

By next lab period, a short report containing abstract, design notes, source VHDL, summary statement regarding results, and problems encountered. One report per team please.