Lab Exercise #4

Objective

The objective of this lab is to begin using a hardware description language (HDL) to describe circuits that will be implemented in an FPGA.

References

Handout: Logic Implementation for a Xilinx FPGA using VHDL
Xilinx hardware and software documentation on the class web page.

Design Flow

The flow for today's lab is:

- Do the initial design. Create suitable documentation, i.e. a block diagram of your circuit.
- Implement the design using VHDL
- Synthesize, place, route, and create bit map file using Xilinx ISE
- Download bit map to FPGA board
- Verify operation (and debug as needed)

Implement a simple circuit to learn VHDL driven design

When working with modern design tools (software) I highly recommend doing a simple design before doing a complex design. This allows you to learn what is called "the design flow", learn how to use each tool, and check that tools are working.

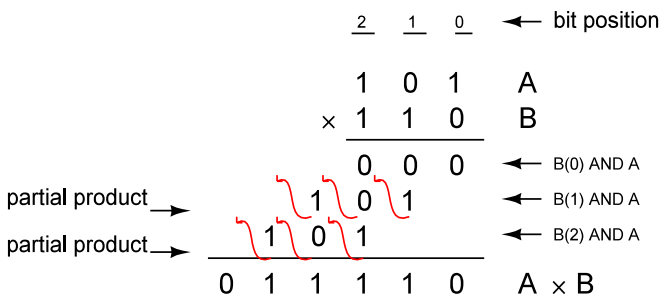
Thus, create a circuit where Led 0 lights up when sw0 or sw1 are asserted but not if both are asserted. Write a VHDL description, synthesize, and download to the FPGA.

Design Problem

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 Led 7 - 0 where the least significant bit is Led 0. Note that this circuit only uses combinational logic and for this lab you cannot use math operators in your VHDL description, only logical operators.

Start by considering how you do manual multiplication a digit at a time with pencil and paper:

Long hand multiplication of two 3-bit binary numbers A, B.



notes:
1) At each addition location a one-bit full binary adder will work.
2) The red 'S' shaped lines indicate a carry-out to the left

A one-bit full adder circuit has bits A and B and Carry_in as input. Output is 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 we call 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 just writing VHDL statements expressing the desired logic. Then use structural style VHDL to connect these modules together for the multiplier circuit.

To Turn In

By next week, a short report containing:

Abstract, design notes, source VHDL file(s), summary statement regarding results and problems encountered. One report per team.