**HW #1 (p. 41)**

1. For each phase in the VHDL/PLD design methodology design flow in Fig 1.1.1 (p. 4) give a brief description (one or two sentences) of what is accomplished.
   1. Analyze Requirements and Develop Specification:  
       To take a plain language problem or goal, analyze and specify the details of the problem in concepts that translate more directly to a design (sketches, describing inputs and outputs, etc).   
       From there, develop the general shape of the end-product by specifying the type of interface for input and output and the broad strokes of internal logic.
   2. Write and Compile VHDL Design Description  
       Write a VHDL program that meets the specification developed in the previous step. It is important to remember that lines written are executed concurrently on the hardware, unless specified as sequential by being placed inside a process statement.  
       Compiling, or analyzing, a design description checks toe program for semantic and syntax errors and is done prior to simulation.
   3. Develop Verification Plan, Write and Compile VHDL Test Bench  
       Determine a set of test inputs and expected outputs that can efficiently test the wiring and logic of the design.  
       Write the verification plan in VHDL as a Test Bench that sequentially applies the test inputs to the device and can output error messages if the device generates an unexpected output.  
       Compile the test bench to check for semantic and syntax errors.
   4. Functional Simulation  
       Run a simulation of the test bench connected to the device. This simulation tests the logic of the designed device without any expected real-world constraints, such as timing errors or hardware limitations.
   5. Select PLD, Synthesize Logic  
       Select an appropriate PLD to use to implement the design. When considering the device, it is important to balance the cost of the device, the logic capacity of the device, and the speed of the device.  
       With the PLD chosen, the design can be synthesized. This converts the original VHDL design to function using the logic gates onboard the hardware, or a gate-level logic description.
   6. Post-Synthesis Simulation  
       A simulation is run with the previous test bench using the new gate-level logic description of the device instead of the original design. This step tests that the synthesis of the original design logic to hardware specific logic was completed successfully, and the output of this simulation should match that of the previous one.
   7. Place-and-Route Logic to PLD, Timing Simulation  
       In this step the software will determine where to place each of the necessary gates in the PLD architecture and how to route between each of the gates.  
       This description is used in the next simulation in order to determine any timing delays that will be present in this implementation of the design, as well as any errors that may arise due to propagation delay.
   8. Program PLD, Verify PLD’s operation  
       Implement the design on the target PLD using a device programmer.  
       Verifying the PLD can be done with equipment to directly test inputs and outputs or by implementing the PLD into the system and directly testing its operation.

11) Design a half-adder using the traditional approach. Create k-maps using the provided truth table, determine the sum-of-products for each output, and draw two logic diagram implementations. One using AND, OR, NOT gates and one using NAND only.  
 [Answer attached]

15) Explain the difference between syntax and semantic errors:  
 Syntax errors are errors in text of the language, such as spelling and punctuation errors or missing end statements.  
 Sematic errors are when the language is used incorrectly, such as writing a process statement that contains concurrent logic instead of sequential.

16) Explain the differences between Static and Dynamic Semantic Errors? Which tools may catch them?

Static Semantic Errors are often mismatching of logic, such as comparing objects of different types, having a static index that goes beyond the bounds, or using concurrent logic inside of a process statement. This can usually when compiling for simulation.

Dynamic Semantic Errors can only be caught when the design is running, so in simulation. Some examples would be having an index counter increment past the bounds of an array or an arithmetic function leading to over or underflow.

22) What is a testbench and what is the relationship of a design entity to its testbench? Can the same testbench be used for each of the three kinds of simulations?

A testbench is a VHDL program that is designed to apply a series of inputs to a designed entity and automatically evaluate outputs, throw appropriate errors for any outputs that do not match their expected value. They can be used for all three kinds of simulation.

23) What is an UUT?  
 UUT stands for Unit Under Test, and it is the design entity being tested with a testbench.

27) Examine the testbench in Listing 1.5.1 Create a table listing, in order, the stimulus values applied and the expected output values for each stimulus value.

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | Sum (a ⊕ b) | Carry out(a.b) |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

29) What is a Programmable Logic Device (PLD) and what are the three major categories.

PLDs are mass produced devices with prearranged logic that can be programmed by breaking and connecting lines to the gates. How rewritable they are and what type of logic is on the devices is entirely dependent on the specific technology.

The three major categories are Simple PLD, Complex PLD, and Field Programmable Gate Arrays (FPGA).

35) Why choose a PLD with slightly more logic capacity than is expected to be required for a design? Why not choose a PLD with substantially more logic capacity than what is expected to be required?

Cost. One shouldn’t overdesign their product and increase the cost of production unnecessarily, and spending extra for additional capacity that is, by design, expected to be unused is unnecessary.

44) List six advantages of the VHDL/PLD methodology over traditional digital design.

- Automatic Optimization

- Error checking for specifically logic errors (simulation)

- Error checking that is specific to a technology (post-synthesis)

- Easily test a variety of technology prior to purchase

- Rapid iteration of a project

- Easily importing and testing previous, functioning designs as modules in a new design

46) Can every VHDL program that can be simulated be synthesized? If not, explain why?

Synthesis is entirely bound by the target hardware. A program could be written that exceeds the capabilities of any target technology we could currently synthesize, e.g. a full-adder of an “infinite” length.