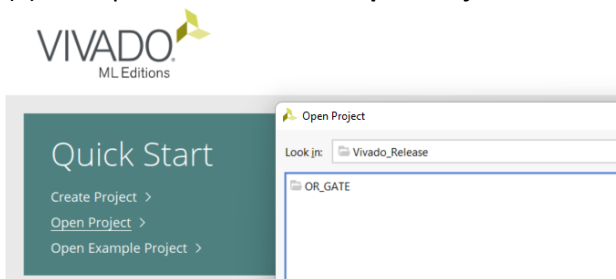


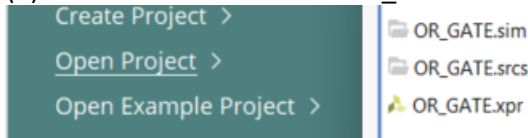
Purpose	(i) To learn how to navigate Vivado, (ii) To learn some basic VHDL.
Files Required	Vivado software (on machines, free download) and zipped projects
ILO Contribution	LO 5
Send to Me	nix
Homework	Read chapter 13

1. Getting to know Vivado – This will be our workflow.

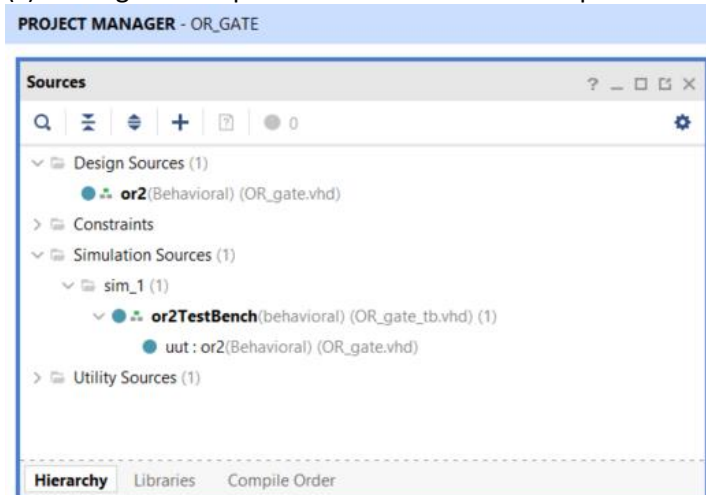
(a) Fire up Vivado and select **Open Project**. You will see a list of project folders like this (you'll get more)



(b) Double click on the folder OR_GATE and select the file OR_GATE.xpr and hit OK

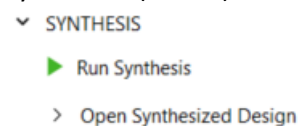


(c) You'll get a complex UI. Look for the Sources pane and expand so it looks like this

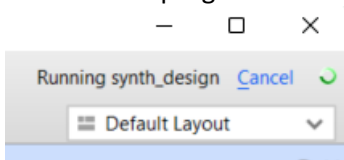


(d) In Design Sources you will see **or2**. Right-click and Open. Then in Simulation Sources you will see **or2TestBench**. Right-click and open. So now you have two VHDL files open. Just park them for the moment.

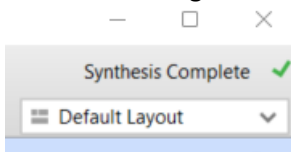
(e) Let's say we have made some changes to the design source **or2**. The next step in the workflow is to synthesize (create) the electronic circuit. In the PROJECT MANAGER tab on the left click on **Run Synthesis**



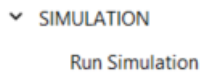
and at the top-right of the window you will see the following indication



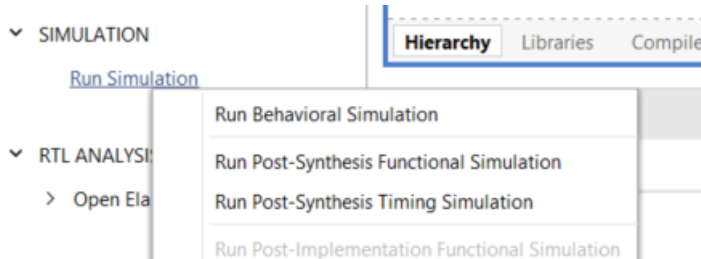
which will change to this when the synthesis is complete



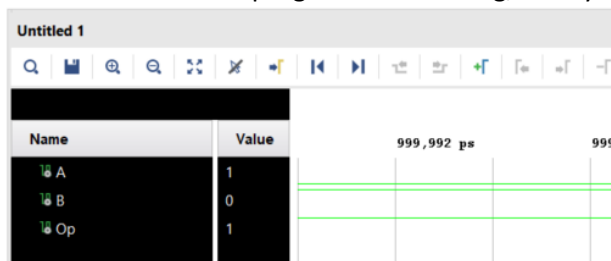
(f) Now we simulate our synthesized circuit by hitting Run Simulation in the PROJECT MANAGER tab,




and choose Run Behavioral Simulation



You'll see some nice progress bars moving, then you will get a window which looks like this.

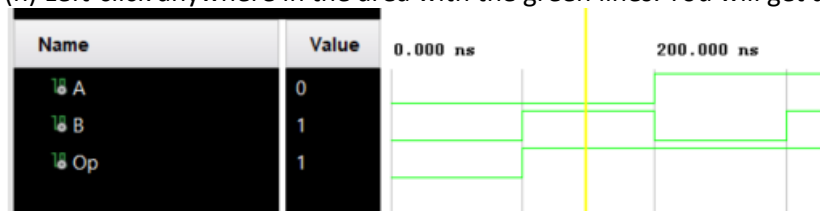


(g) Click on the Zoom Fit icon  and you'll get an interesting testbench waveform which looks like this



The block on the left indicates the signals: A and B are inputs to our OR-gate and Op is the output. The green lines show how these signals change with time. The testbench provided values for A and B, the Op is computed by our device simulation *as a function of time, moving to the right*. You will see the input values of A,B are 0,0 then 0,1 then 1,1

(h) Left-click anywhere in the area with the green lines. You will get a yellow indicator of the signals at any time



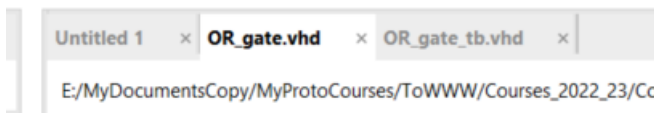
The numbers in the Value field will change to tell you the values here, so above I got A=0, B=1 and Op=1. Yep, that's an OR-gate!

(i) Now explore the waveform and convince yourself that you are really looking at an OR-gate.

2. Reading VHDL for the OR-gate

To quickly learn a new language, it's often best to learn to read that language. So let's have a go. You opened both the design source and the simulation source and parked both.

(a) In the right-most window select the design source **OR_gate.vhd** to give you a code editor



(b) Look at the code block lines 15-21. This is where the OR-gate *functionality* is defined. Line 19 describes how the output **Op** signal is derived from the two input signals **A** and **B**. Note the special operator **<=** which reminds us that we are assigning signals not conventional program variables.

(c) Look at the code block lines 9 – 13 which defines the OR-gate *structure*, it has two inputs A and B and one output Op. Please notice they have both a 'type' (std_logic) and whether they are in or out (inputs or outputs). You can think of std_logic as either HIGH '1' or LOW '0'.

(d) Now comment out line 19 and replace it with the following alternative way to code an OR-gate. This is discussed in 13.2.1

```
Op <= '0' when A = '0' and B = '0' else  
      '1' when A = '1' or B = '1';
```

(e) Change the code in OR_gate.vhd to another gate type (e.g. an AND gate). Think, what do I need to change? The *structure* in the **entity** block or the functionality in the **Behavioral** block?

(f) Now run through our workflow established in Activity 1 (e), (f), (g), (h) to check out your chosen gate.

(g) Try the alternative way to code as in part (d).

3. Synthesizing a Digital Circuit for a 'Logic and Language' problem.

Remember in our previous sessions we explore Logic and Language and investigated a number of problems. Most of these involved 'sum of products'. Here you can choose any Boolean sum of products expression and code this

(a) Open the project **NoC_LOGIC_PROBLEM** and open the design source **NoC_Logic_Problem.vhd**. Here you will find the code for the problem

$$L = A \cdot \sim B + A \cdot B$$

which in VHDL looks like this

```
Op <= (A and not(B) or (A and B)
```

Change this line so it codes your chosen problem, synthesize the circuit and run the testbench.

(b) Check the testbench waveform and make sure it agrees with your original truth table.

(c) [optional] An alternative way of programming the expression by using variables explicitly to represent mini-terms is shown in Chapter 13 page 7. You could try this out if you like.
