LABORATORY MANUAL

SC1005 : Digital Logic

Lab 3 :
Combinational Logic Design with
Structural Verilog

COMPUTER ENGINEERING COURSE
COMPUTER SCIENCE COURSE

COLLEGE OF COMPUTING AND DATA SCIENCE
NANYANG TECHNOLOGICAL UNIVERSITY

COMBINATIONAL LOGIC DESIGN WITH STRUCTURAL VERILOG

1.   OBJECTIVES

1.1   To obtain minimum-cost SOP Boolean expressions using Karnaugh Maps

1.2   To explore logic circuit design with the Xilinx FPGA design tool chain

1.3   To design a 7-segment decoder with structural Verilog statements

1.4   To simulate, synthesise and implement the designed circuit on Xilinx FPGA

1.5   To selectively display a digit using a push button and multiplexer

2.   LABORATORY

There are two lab venues. A student will go to the same venue for all 5 lab sessions.

Hardware Laboratory 3, N4-B1a-05

Hardware Project Lab, N4-01C-09A

3.   EQUIPMENT AND SOFTWARE

Personal computer
Xilinx FPGA tool chain (Vivado Design Suite)
Xilinx University Programme (XUP) Artix-7 Basys3 development board

4.   INTRODUCTION

4.1   In Lab 1 and 2, you have implemented logic circuits using SSI and MSI logic components mounted on a breadboard. That is a quick and low-cost method to build and test simple circuits; but it can become messy for more complex circuits. In this lab, you will implement a 7-segment decoder circuit on the Xilinx FPGA (Field-programmable Gate Array), which comprises logic components that can be programmed to implement the desired logic [Ref 8.4].

Figure 1 shows the diagram of a 7-segment decoder and display. The segments on the display unit are labeled a, b, c, d, e, f and g. By lighting up the correct segments (e.g. a, b and c in Figure 1), different numeric digits (e.g. "7" in Figure 1) can be displayed. Table 1 lists the hexadecimal digits that can be displayed by controlling the 4-bit input x[3:0].

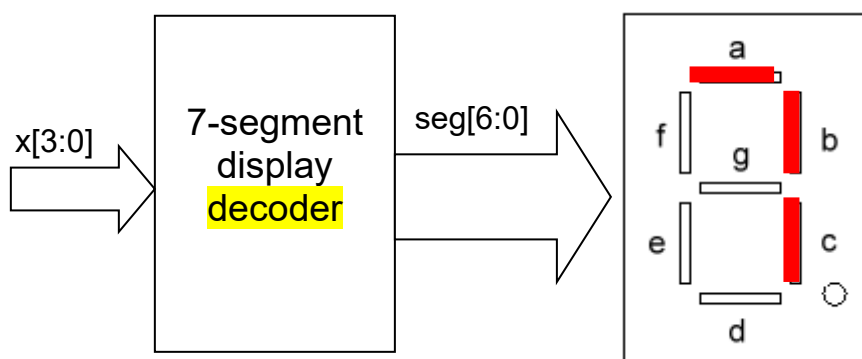You will use the Xilinx Vivado design suite [Ref 8.3] to design and implement the decoder.

Figure 1: 7-segment decoder and display (e.g. displaying the digit "7")

4.2   Fill in the remaining values in Table 1 to achieve the desired digit display. After that, draw a Karnaugh map for each of the 7 outputs: a, b, c, d, …, and g; obtain the minimized Sum-of-Product (SOP) expression for each output.

For example, the minimized SOP expression for a is:

a = **x3'x2x0** + **x2x1** + **x3'x1** + **x3x0'** + **x3x2'x1'**
        + **x2'x0'**

where x3, x2, x1, x0 are the 4 bits
of the hexadecimal digit (x3 is msb, x0 is lsb)

Kmap for a		x1, x0			
		00	01	11	10
x3, x2	00	1	0	1	1
	01	0	1	1	1
	11	1	0	1	1
	10	1	1	0	1

Note that in this case a total of 6 loops are required to minimize the SOP: 4 of them cover 4 minterms each, and 2 of them cover 2 minterms each.

4.3   Obtain the minimized SOP expressions for each of the remaining segments: b to g. You will need to obtain these 6 logic expressions before the lab class.

You may make use of an online Kmap tool [Ref 8.5] to verify your expressions but make sure you are able to construct Kmaps with pen and paper on your own.

4.4   You will design the 7-segment decoder using simple structural Verilog HDL (Hardware Description Language) statements to describe logic expressions. Essential information on structural Verilog is included in the Appendix.
For example, the expression in 4.2

a = **x3'x2x0** + **x2x1** + **x3'x1** + **x3x0'** + **x3x2'x1'** + **x2'x0'**

can be written as follows in Verilog syntax:

assign a = **~x[3]&x[2]&x[0]** | **x[2]&x[1]** | **~x[3]&x[1]** | **x[3]&~x[0]** | **x[3]&~x[2]&~x[1]** | **~x[2]&~x[0]**;

Table 1: Truth table of a 7-segment display decoder

Inputs x[3:0]		Active High Outputs seg[0:6]						
Hexadecimal digits (binary)	Display	Segments (1: on, 0: off)						
		a	b	c	d	e	f	g
0 (0000)		1	1	1	1	1	1	0
1 (0001)		0	1			0		0
2 (0010)		1	1			1		1
3 (0011)		1	1			0		1
4 (0100)		0	1	1	0	0	1	1
5 (0101)		1	0			0		1
6 (0110)		1	0			1		1
7 (0111)		1	1			0		0
8 (1000)		1	1	1	1	1	1	1
9 (1001)		1	1			0		1
A (1010)		1	1			1		1
b (1011)		0	0			1		1
C (1100)		1	0	0	1	1	1	0
d (1101)		0	1			1		1
E (1110)		1	0			1		1
F (1111)		1	0			1		1

Note: seg[0] is segment a, seg[1] is segment b, …, seg[6] is segment g
In digital circuits, a multi-bit signal e.g. x[3:0] is also known as a bus.

4.5   In this lab, you may also make use of a 2-input multiplexer. Figure 2 shows its logic symbol and truth table. By controlling the logic input Sel, either the input D0 or the input D1 is directed to the output X at any time.
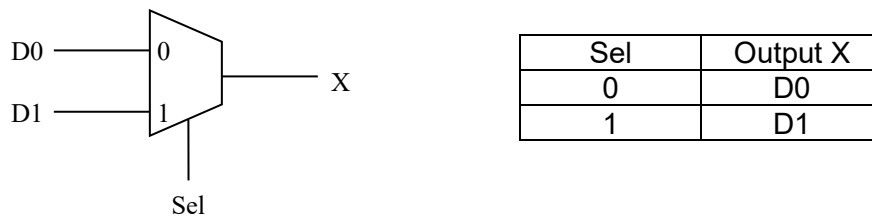

Sel	Output X
0	D0
1	D1

Figure 2: Logic symbol and truth table of a 2-input multiplexer

4.6   Following a number of steps, you will design the decoder using the Boolean expressions obtained from 4.3, then **simulate** it on the PC. The test bench file (given to you on NTULearn) contains input test data for the decoder. The function of the 7-segment decoder circuit is verified by viewing the simulation result on the PC. Note that for simulation, an actual FPGA board is not required.

4.7   After the decoder's function has been verified to be correct using simulation, you will proceed to **implement** the circuit on physical hardware. The inputs are mapped to switches on the FPGA board, and outputs are mapped to the 7-segment display. The mapping is specified in an XDC (Xilinx Design Constraints) file. The circuit is then synthesized to generate a bitstream file which is used to program the FPGA. The actual circuit implemented on the FPGA is then ready for testing. You may refer to the Appendix for the typical workflow.

4.8   You may contrast this modern method of logic circuit design and implementation with the traditional breadboard+ICs+wires approach used in lab 1 and 2.

5.   PROCEDURE

5.1   SET UP DESIGN ENVIRONMENT

5.1.1   After logging in to the PC with your NTU account, open File Explorer. Go to the project location (e.g. J) specified by the lab executive, create a New Folder and name it **Lab3**. If somebody has previously created a folder named Lab3, you should first delete that old folder if you do not want unpleasant errors from the Vivado software later in the procedure.

Copy the 3 given files *vsevenseg.v, vsevenseg_tb.v* and *vsevenseg.xdc* from NTULearn and place them in the newly created folder **Lab3**.

Check that the file extensions (e.g. *filename.v, filename.xdc*) are not modified. Also, if any filename contains brackets, e.g. *filename(1).v*, rename the file to get rid of the brackets.

You may refer to steps 1 to 9 of *Lab3_Vivado_guide.pdf* for a pictorial guide for the steps below.

> Vivado is a comprehensive software and we only use a small part of it.
> **It is IMPORTANT that you follow every step carefully and NOT try to take shortcuts.**
> Any wrong shortcuts taken are likely to lead to errors and cause you to spend FAR MORE time on figuring out what has gone wrong.

5.1.2   Double click the **Vivado 2022.2** shortcut to start. Wait patiently as the software may take a while to open. <u>DO NOT</u> click the shortcut multiple times.

5.1.3   Create a new project (**Quick Start > Create Project**). This opens the *New Project Wizard*.

5.1.4   In the *New Project* window, specify the <u>Project Location</u> (e.g. J) and <u>Project Name</u> (e.g. Lab3). They must follow the ones you specified in step 5.1.1 above.

5.1.5   Add the design file *vsevenseg.v* to the project – it should be added as a **Design Souce**.

5.1.6   Specify the correct part for the FPGA board. Check the *Project Summary* then click *Finish*.

xc7a35tcpg236-1

5.2   DESIGN THE 7-SEGMENT DECODER MODULE WITH STRUCTURAL VERILOG STATEMENTS

You may refer to steps 10 to 12 of *Lab3_Vivado_guide.pdf* for a pictorial guide.

5.2.1   Double click on the **Design Source** file *vsevenseg.v* to view its content. Comments have been added to make it easier to see what the circuit does. Take note of the circuit's input and active-low outputs. **Note that Verilog syntax is case-sensitive.**

5.2.2   Compare the logic expressions of segments a, b, e and g given in the file with the ones you have obtained in 4.3.
Pause and think: Are they exactly the same? If not, why?

Refer to the Appendix for the essential Verilog operators.

5.2.3   Using the logic expressions that you have obtained earlier in 4.3, <mark>key in the Verilog expressions of segments c, d and f into the file *vsevenseg.v*.</mark> Follow the same syntax as the given expressions and don't forget the semi colon at the end of each line. No statement must be inserted after **endmodule**. Click **SAVE**. Fix all syntax errors (if any) before proceeding.

5.3   SIMULATE WITH A TEST BENCH

You may refer to <mark>steps 13 to 22 of *Lab3_Vivado_guide.pdf*</mark> for a pictorial guide.

5.3.1   Add the given test bench file *vsevenseg_tb.v* to the project – it should be added as **Simulation Source**.

5.3.2   Double click on the **Simulation Source** file *vsevenseg_tb.v* to view its content. Note that the values of input x are varied from 0h to Fh to observe the corresponding changes in the active-low output seg_L.

5.3.3   On the *Flow Navigator* (left hand panel) click **Run Simulation > Run Behavioral Simulation** to launch the simulator.

5.3.4   On the tool bar, change the run duration to 300 ns (nanosecond). Click **Restart** followed by **Run for time duration specified**. On the simulation window, click the **Zoom Fit** button to get a good view of the simulation result.

5.3.5   Verify that the decoder outputs are correct for each value of input x. Note that seg on Table 1 is active-high whereas seg_L is active-low (_L is deliberately included in the output name to highlight that it is active-low). Also, segment g is the msb (seg_L[6]) while segment a is the lsb (seg_L[0]).

If your observations are incorrect, it probably means that you have made one or more mistakes when entering the Boolean expressions for segment c, d and f into the vsevenseg.v file. You should correct the mistakes before proceeding to 5.4.

Assessment (a):

Student to show simulation results to instructor

5.4   MAP INPUTS/OUTPUTS AND IMPLEMENT CIRCUIT ON FPGA

You may refer to steps 23 to 33 of *Lab3_Vivado_guide.pdf* for a pictorial guide.

5.4.1   You will implement the decoder as a physical circuit on the FPGA. Connect the USB cable of the Basys3 board [Ref 8.2] to the PC and slide the power switch to ON position. The onboard 7-segment display will flash numeric digits to show that it is working.

5.4.2   To see the physical effect of your decoder circuit, you will make use of the onboard switches and 7-segment display. Figure 3 illustrates how the decoder inputs/outputs (I/O) are mapped to the   switches and display. The exact mappings are specified by the constraints file *vsevenseg.xdc*.
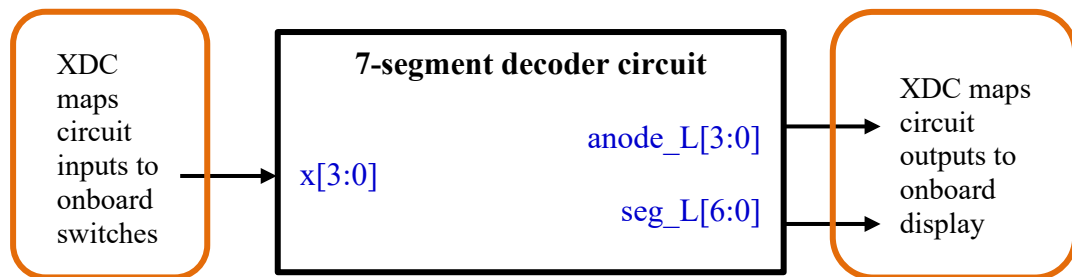

Figure 3: XDC file maps circuit I/O to onboard switches and display

5.4.3   Figure 4 shows the I/O schematic diagram and Figure 5 shows the physical layout of the Basys3 board. The I/O mappings specified by *vsevenseg.xdc* are indicated in these figures.

5.4.4   Add the given constraints file *vsevenseg.xdc* to the project – it should be added as **Constraint.**

5.4.5   Double click on the added **Constraint** XDC file to open it and examine the I/O mappings of *vsevenseg.v* to the FPGA board. Note that for every I/O pin, two statements are required:

```
set_property PACKAGE_PIN pin_id [get_ports {port_name}]
set_property IOSTANDARD LVCMOS33 [get_ports {port_name}]
```

The port_name must match the input/output name specified in the circuit design.

The pin_id is the unique identifier for a pin, which is usually printed on the board. For example, the id for SW3-SW0 are W17, W16, V16 and V17 respectively (see below).
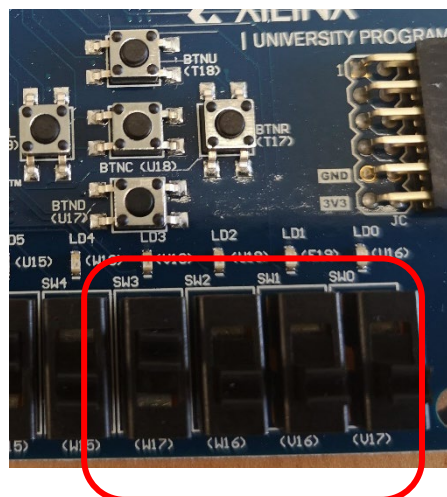
5.4.6   On the *Flow Navigator* click **Run Implementation**. It will take a while to complete the synthesis. If there is error in the process, look at the error messages and fix the error (e.g. a wrong FPGA part might have been selected in 5.1) before proceeding further.

5.4.7   On the *Flow Navigator,* generate the bitstream and program the FPGA board.

5.4.8   Test the circuit by sliding the switches SW3-SW0 to different values. Are all the 16 digits displayed correctly? If not, check the following:
- the logic expressions in *vsevenseg.v*
- the mappings in the *vsevenseg.xdc* file

Assessment (b):
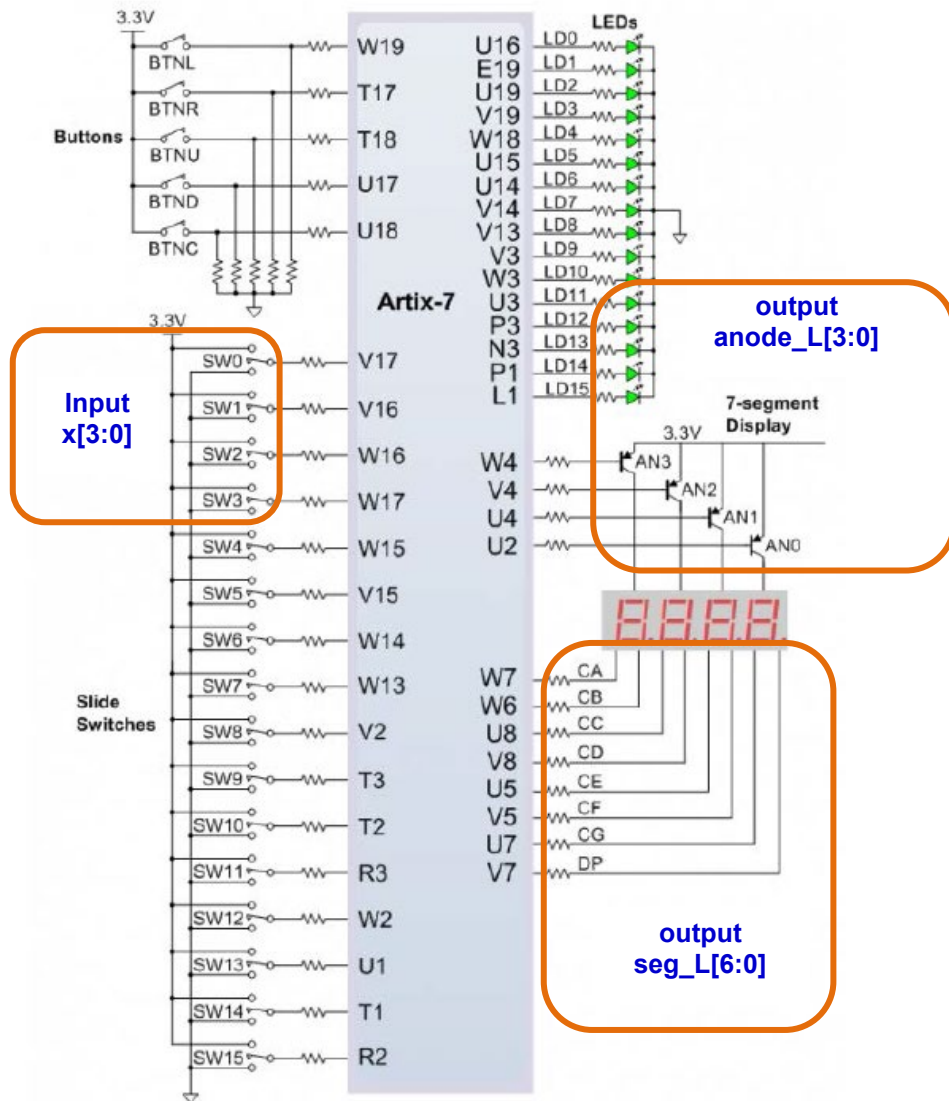
Student to show working 7-segment display to instructor

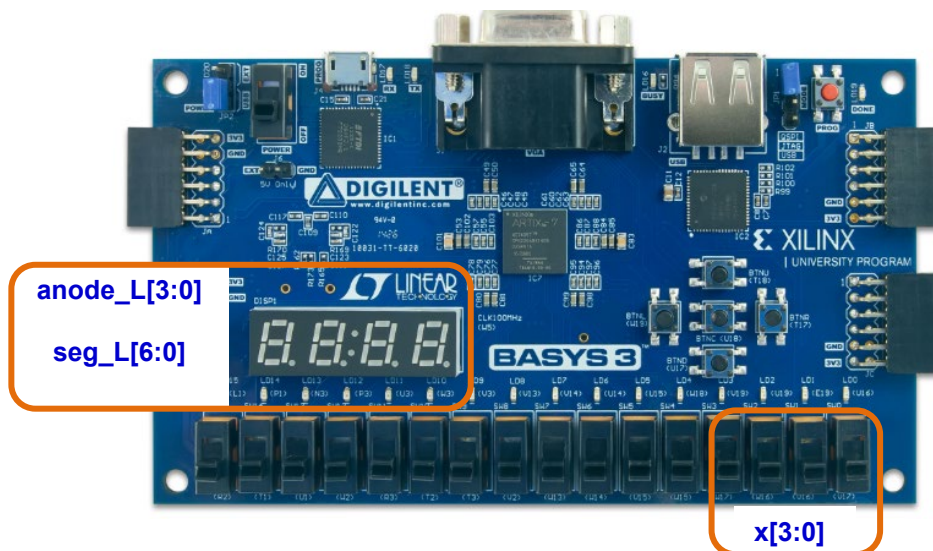Figure 4: Basys3 basic input/output (I/O) schematic (source: Ref 8.2)

Figure 5: Basys3 board physical layout (source: Ref 8.2)

Attempt the optional part only if you have completed and understood 5.1 - 5.4.

5.5   OPTIONAL: MODIFY CIRCUIT TO DISPLAY EITHER LEFT OR RIGHT DIGIT

You may refer to steps 34 to 37 of *Lab3_Vivado_guide.pdf* for a pictorial guide.

5.5.1   In section 5.4, the two digits that are lighted up always display the same value. This is because the digits share the same value of CA-CG (mapped to seg_L in Figure 4). However, Figure 4 also shows that the two digits can be turned on or off separately by AN1 and AN0.

The two digits can be made to display 2 different values (e.g. m, n) by making seg_L=m when AN1 is asserted, and seg_L=n when AN0 is asserted.

Side note: If AN1 and AN0 takes turn to be asserted rapidly (say 60 Hz), the two digits will flash so rapidly that they appear to be lighted up all the time. You will explore this in Lab 4.

5.5.2   Open *vsevenseg.v* to edit the **design**. The new circuit will have two 4-bit inputs: a and b. The input "left" will make the left digit light up instead of the right digit. Figure 6 shows how a multiplexer can be used to selectively display the digit.
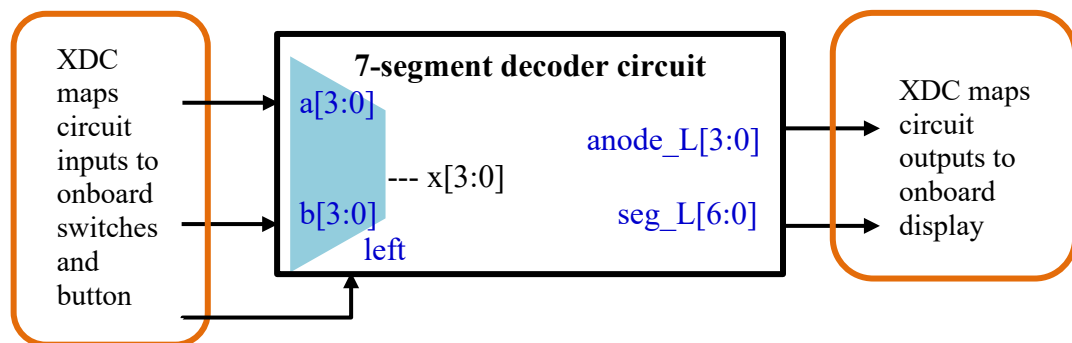

Figure 6: Multiplexer selectively displays left or right digit

5.5.3   Open *vsevenseg.xdc* to edit the **constraints**. Map input a to SW7-SW4, b to SW3-SW0, and left to the button BTNC. Save all the changes.

5.5.4   Follow the same steps in 5.4.6 and 5.4.7 to generate the new bitstream file and program the FPGA.

5.5.5   Select the left or right digit to be displayed by using the pushbutton BTNC. Note that when the pushbutton is **pressed and held down**, it generates a logic 1 signal, otherwise it is logic 0. i.e. it produces an <u>active high</u> signal.

6.   PREPARATION AND OBSERVATIONS

Maintain proper records of your preparation (e.g. Kmaps, derivations of logic expressions, sketches of logic circuit diagrams, etc.) and observations.

7.   ASSESSMENT AND QUIZ

Each student is required to demonstrate the completion of **sections 5.3 and 5.4** to the lab instructor.

There will be a 10-minute online quiz at the end of this lab. The quiz will include (but not limited to) materials covered in this lab.

8.   REFERENCES

8.1   SC1005 Lab3 laboratory materials, Digital Logic Couse Site, NTULearn.

8.2   https://reference.digilentinc.com/basys3/refmanual (FPGA board reference manual)

8.3   https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-

Design/2013x/Nexys4/Verilog/docs-pdf/Vivado_tutorial.pdf (Vivado tutorial)

8.4   http://en.wikipedia.org/wiki/Seven-segment_display

8.5   http://www.32x8.com/   (online Kmap tool)

Appendix

Note on Xilinx Vivado:
For design (.v) and constraints (.xdc) files, it is not sufficient to place them in the same working directory as the project. They need to be added to the design using **Add Source.** Similarly, these files can be **removed** from the design when they are not needed. Removing the files from a design will <u>not</u> delete the files from the working directory.

Verilog bitwise operators (can be use for single- or multi-bit signals)

Operator	Logic	Example	Effect
~	NOT	~A	If A=1, then ~A=0 If A=0, then ~A=1 If A=1010, then ~A=0101
&	AND	A&B	0&1=0; 1&1=1 0101&0011 = 0001
|	OR	A|B	0|1 = 1; 0|0 =0 0101|0011 = 0111

Note: usual order of precedence applies

A typical Structural Verilog module declaration

```verilog
module module_name (
    input input_name1,        // comment…
    input input_name2,
    input input_name3,
    etc.,
    output output_name1,
    output output_name2,
    etc.);

    // comment for better clarity
    assign output_name1 = logic expression1;
    // comment for better clarity
    assign output_name2 = logic expression2;

    … etc.

Endmodule
```

Note: each **assign** statement must end with a semi colon (;)

Typical workflow when using Vivado to design/simulate/implement circuits

1. Prepare the design file(s) (filename.v), test bench file (filename_tb.v) and constraints file (filename.xdc) in advance.
2. Use Windows Explorer to create a new folder and give it a simple name (e.g. Lab3).
3. Copy the files prepared in step 1 to the folder created in step 2.
4. Open Vivado, create a new project by selecting the folder created in step 2 as the working directory.
5. Add the design file(s) as **Design Source** to the project
6. Add the test bench file as **Simulation Source** to the project
7. Add the xdc file as **Constraints** to the project
Note: adding a file as a wrong type of source will give error further down the design/implementation process
8. Select the correct FPGA hardware part – in the case of SC1005, it is **xc7a35tcpg236-1**.
9. View the content of the added files to make sure that they are indeed the correct ones.
10. If there are more than one design files (i.e. there is a top module that instantiates one or more other modules), make sure that the correct module is set as the **top module**.
11. Fix all syntax errors in the design file(s).
12. Click Save to make sure that all files are up to date.
Note: if you made some changes without saving, the old files are still being used for the design/implementation process
13. Run simulation and check that the circuit is behaving correctly. If not, make changes to the design file(s), repeat steps 12 & 13 until the circuit behaves correctly.
14. Implement the circuit by generating a bitstream. Program the FPGA with the bitstream and verify that the circuit is working correctly.
Note: if you made changes to the design file(s) and/or constraint file, you need to repeat step 14 to implement the new circuit.

Design files: they collectively make up the circuit that you wish to construct. Each filename.v file defines one module.
Test bench file: it simulates the input signals to your circuit and captures the output signals from your circuit. By observing the output signals (in response to the inputs) you should be able to tell whether the circuit is behaving correctly.
XDC constraint file: it specifies the connections between the FPGA hardware (e.g. switches, pushbuttons, LEDs) and the inputs/outputs of your circuit (specifically the top module).

<p style="text-align:center; color:red">DO NOT OPEN MORE THAN ONE INSTANCE OF VIVADO</p>