

## Lab 2 – Numerical Operations and Internal signal Analysis using Verilog

### Introduction

The previous lab work showed how to set up a simple Verilog source file – names for inputs and outputs were set, and logic described both behaviourally and structurally. This lab work illustrates the use of numbers in Verilog, described as signals with more than one bit. Analysis tools to investigate internal nodes are then used.

### Part 1 – ALU Using Verilog

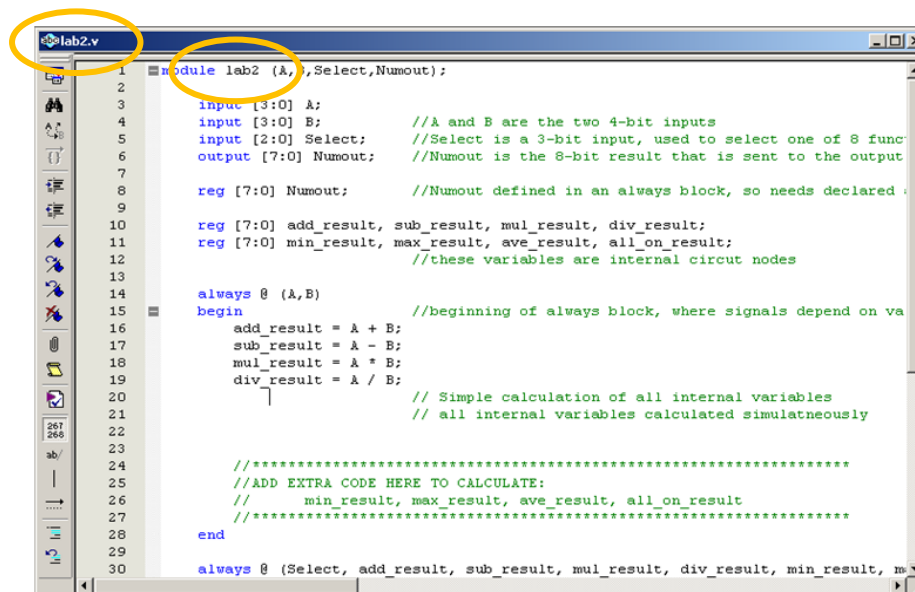
#### Start Quartus and Copy Pre-Written Code

Based on the detailed descriptions given in the worksheet for lab 1:

- Create a new project
- Create and save a new Verilog source file

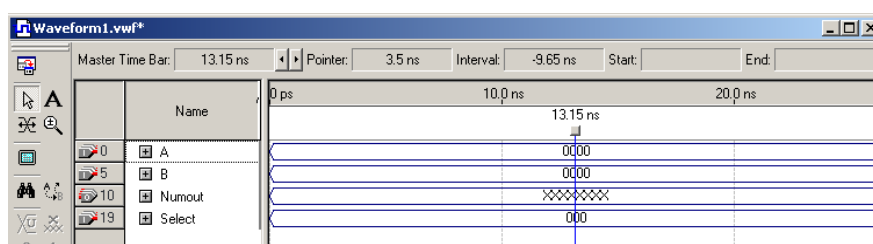
Copy all of the text from the source code appended to the end of this manual with the label “Lab2code.txt” and paste into the empty Verilog source file. **Compile**.

If an error is reported, check the name of the file and the name of the module – these must match. See below, where the source code file has been saved as ‘lab2.v’ and the name of the module is ‘lab2’. If you have saved the source code under a different name, simply change the name of the module in the first line of code to match this. Note that names are case sensitive.



### Simulate

Open a new Vector Waveform file, as described in lab 1. Insert all signals, in exactly the same way as in lab 1. The waveform window should appear as below.



Note that because A, B, Numout and Select were all defined as groups at the beginning of the code, they automatically appear as groups in the waveform window.

As before, change the end time to give a longer simulation interval:

Edit -> End Time...

Change the value to 100us.

In lab 1, group signals were set using binary values – this time we will use decimal values, which makes it easier to check that arithmetic is being performed correctly.

Click on 'A' to highlight the signal in blue, then right-click.

From the shortcut menu that appears, select 'Properties'.

In the properties box that appears, change the radix to 'Unsigned Decimal' and click 'OK'.

Do the same for signals 'B', 'Select' and 'Numout'.

Use the zoom tool to zoom out as far as possible.

Click on 'A' to highlight the signal. From the buttons on the left, click the one with the '?' symbol, as shown:



In the box for 'Numeric or named value', enter '13', and click 'OK'. It should be seen that a value of 13 is set for 'A' over the entire waveform interval.

Follow the same procedure to set a value of 7 for 'B'.

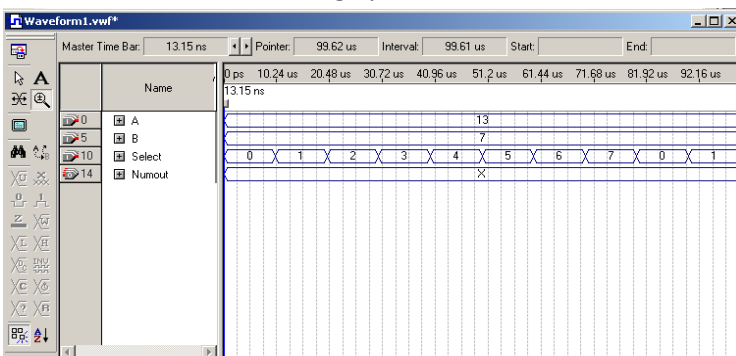
Ideally, we would wish to see how the circuit responded to all possible input combinations. However, with A and B both being 4 bits wide and the Select input being 3 bits wide, there are 11 input pins in total. To fully test therefore,  $2^{11}$  (2048) combinations would need to be set up – this is impractical in a single simulation. To make it more reasonable, A and B have been set to a fixed value throughout, and we will set all possible values of 'Select' for these specified values of 'A' and 'B'.

To do so, click on 'Select' to highlight the waveform.

Click the button on the left that applies a count sequence:

Click the 'timing' tab, and change to count every 10us. Note that it is necessary to change from ns to us in the dropdown list.

If the 'Select' waveform appears below 'Numout', click once on it to highlight it in blue. Release the mouse button, then click and hold, and drag upwards to move it above 'Numout'. The window should be as shown below.



Save the waveform file, and run the simulator.

Fill in the table below:

INPUT SIGNALS: A=13, B=7	
Select	Numout
0	
1	
2	
3	

What function has been performed in each case?

INPUT SIGNALS: A=13, B=7	
Select	function
0	
1	
2	
3	

Look again at the code – are these results what would be expected?

Click the '+' button next to each of the signals. What can be said about the decimal number shown and how it is made up?

What is the value of 'Numout' when 'Select' has values 4,5,6 and 7? Why is this? Does it matter?

Change the values of A and B to A=11, B=4, save, then run the simulator again. Fill in the table below.

INPUT SIGNALS: A=11, B=4	
Select	Numout
0	
1	
2	
3	

Does this give the expected results? Comment specifically on the result when Select=3. What is happening here?

## Modification

As should now be apparent, this is a simple ALU. The select input controls which function is implemented, i.e. if Select is 0, the output is A+B. If select is 1, the output is A-B, etc.

The design is to be modified. As before, it should accept two 4-bit numbers A and B, along with a 3-bit Select code. The output will remain an 8-bit number.

The functions to be implemented are shown in the table below. Note that 3'b is Verilog syntax for a 3-bit binary number.

Select code	Function	Description
3'b000 (0)	ADD	Numout=A+B
3'b001 (1)	SUBTRACT	Numout = A-B
3'b010 (2)	MULTIPLY	Numout = A*B
3'b011 (3)	DIVIDE	Numout = A/B
3'b100 (4)	MINIMUM	Numout = the smaller of A and B
3'b101 (5)	MAXIMUM	Numout = the larger of A and B
3'b110 (6)	AVERAGE	Numout = mean value of A and B
3'b111 (7)	ALL ON	Numout = 11111111 <sub>bin</sub>

Look in the code listing at the end of these notes for the following sections and also on Vision:

```
//*****
//ADD EXTRA CODE HERE TO CALCULATE:
//      min_result, max_result, ave_result, all_on_result
//*****

//*****
//ADD EXTRA CODE TO PASS min_result, max_result, ave_result, all_on_result
//TO NUMOUT BASED ON THE OTHER POSSIBLE VALUES OF 'Select'
//*****
```

Replace these with the necessary code to implement the 'MINIMUM', 'MAXIMUM', 'AVERAGE' and 'ALL ON' functions.

Compile, debug any errors, and simulate – in simulation set A=14 and B=8.

INPUT SIGNALS: A=14, B=8	
Select	Numout
0	
1	
2	
3	
4	
5	
6	
7	

Are these results as expected?

Using the procedure outlined in lab 1, assign pins and download the design to the DE1 board.

Use the following pin assignments:

Name	Pin number	Description
A[3]	PIN_L2	TOGGLE SWITCH 9
A[2]	PIN_M1	TOGGLE SWITCH 8
A[1]	PIN_M2	TOGGLE SWITCH 7
A[0]	PIN_U11	TOGGLE SWITCH 6
B[3]	PIN_U12	TOGGLE SWITCH 5
B[2]	PIN_W12	TOGGLE SWITCH 4
B[1]	PIN_V12	TOGGLE SWITCH 3
B[0]	PIN_M22	TOGGLE SWITCH 2
Select[2]	PIN_L21	TOGGLE SWITCH 1
Select[1]	PIN_L22	TOGGLE SWITCH 0
Select[0]	PIN_R22	PUSH BUTTON KEY SWITCH 0
Numout[7]	PIN_Y21	GREEN LED 7
Numout[6]	PIN_Y22	GREEN LED 6
Numout[5]	PIN_W21	GREEN LED 5
Numout[4]	PIN_W22	GREEN LED 4
Numout[3]	PIN_V21	GREEN LED 3
Numout[2]	PIN_V22	GREEN LED 2
Numout[1]	PIN_U21	GREEN LED 1
Numout[0]	PIN_U22	GREEN LED 0

The push button key switch is used simply because there are not enough toggle switches for all inputs. Note that when not pressed, the key switches have a logic level of 1 and when pressed they have a logic level of 0.

Set up some input combinations using the switches, and check that the outputs on the LEDs are as intended. Note that they will be shown in the form of binary numbers.

## **Part 2 – Using SignalTap II for Real Time Analysis**

### **Modify the Design to Make it Compatible with SignalTap**

Quartus allows the facility to include an embedded logic analyser within a design. This allows input and output signals to be monitored via the USB interface between the PC and the digital board.

When SignalTap is used, it automatically generates additional logic that it downloads to the board alongside the design. This additional logic requires a clock signal, which must be specified in the design.

To do this, add an extra clock input in the Verilog code, by changing the opening line to:

```
module lab2 (clock,A,B,Select,Numout);
```

Then add, immediately below this, the following line:

```
input clock;
```

Compile the design.

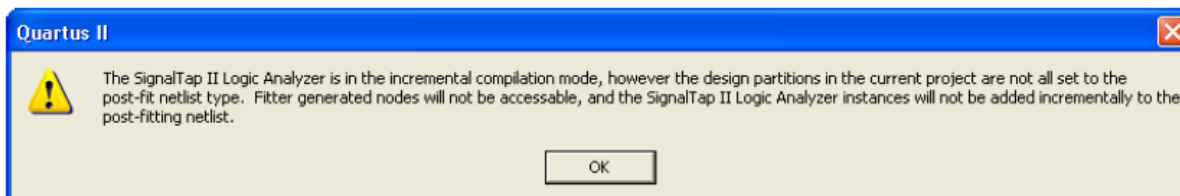
## Create a New SignalTap File

Select:

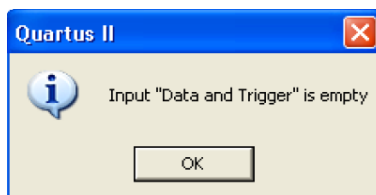
File > New

Click on the 'Other Files' tab, then choose 'SignalTap II Logic Analyzer File' and click OK.

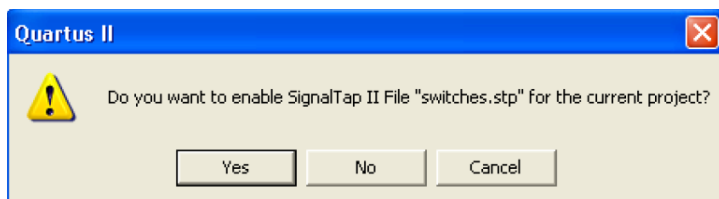
Save the file under the name 'lab2'. If the dialog box below appears, click OK. (Depending on settings and the version of the software used, this does not always appear).



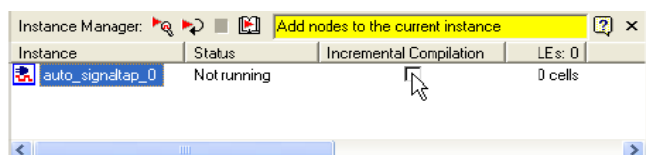
In the next dialogue box, also click OK, as shown below.



For the dialog "Do you want to enable SignalTap II file lab2.stp for the current project," click 'Yes' as shown below. The file lab2.stp is now the SignalTap file associated with the project.



For this project, we wish to turn off the incremental compilation feature of the Quartus II software. To do this, in the SignalTap II window uncheck the Incremental Compilation box, as shown below. NOTE – if you are using the web version on your own computer, the incremental compilation option is not available, so this step can be skipped.

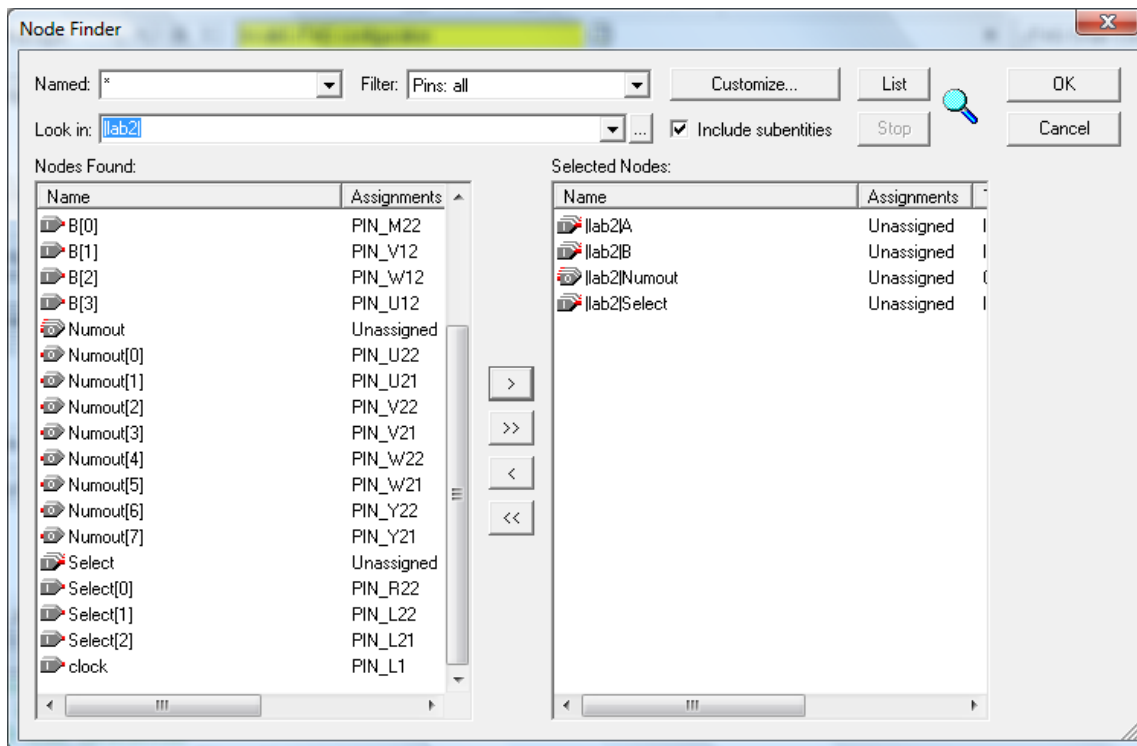


## Add Signals to Monitor

The nodes to be probed must now be added.

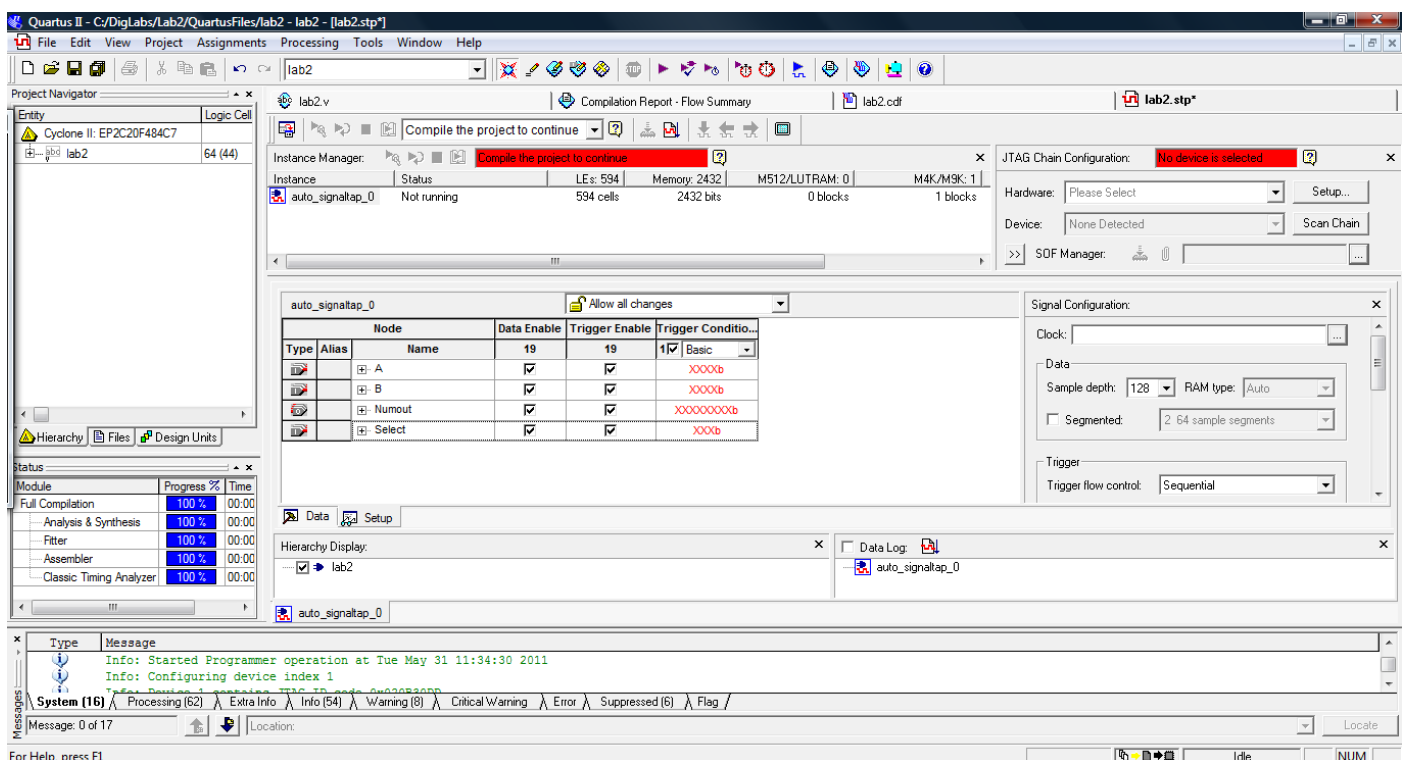
On the left-hand side of the SignalTap window, about 60% of the way down, you should see 'Data' and 'Setup' tabs. Ensure that the 'Setup' tab is selected by clicking it. Immediately above the 'Setup' tab, there should be a white space marked 'Double-Click to add nodes'. Double-click in this area to bring up the node finder window.

In the 'Filter' box at the top of the window, select 'Pins: all'. Click 'List'. This will now display all nodes that can be probed in the project. Hold down the 'ctrl' key on the keyboard while clicking 'A', 'B', 'Numout' and 'Select', and then click the '>' button to add the switches to be probed. The window should appear as shown below:



Click 'OK'.

The main SignalTap window should now appear as below, with the selected signals added:



Right-click on signal group 'A', and from the shortcut menu that appears, select:

Bus Display Format -> Unsigned Decimal

Do the same for B, Numout and Select.

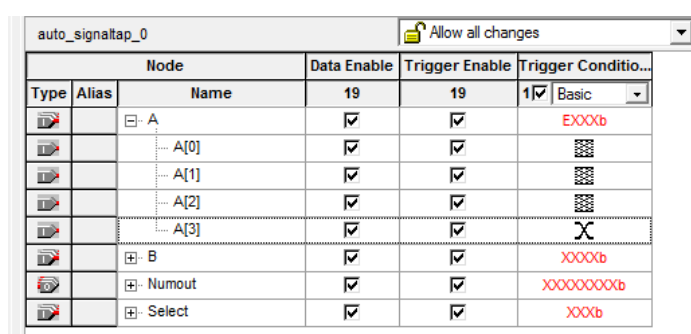
As noted, SignalTap needs a clock signal of some kind in order to run, so this signal must be selected. Using the pin assignment editor, **assign PIN\_L1 to the clock input.**

Returning to the SignalTap window, click the ‘...’ button beside the Clock box of the ‘Signal Configuration’ pane, on the right-hand side of the window. This brings up the node finder window again.

As before, select ‘Pins: all’ as the filter, and click ‘List’. From the list that appears, double-click ‘clock’. The ‘clock’ signal should now be seen on the right hand side of this window. Click ‘OK’.

It is now necessary to set up a trigger condition. The trigger condition is the setting that determines when the logic levels of the inputs and outputs will be captured. For now, we will set the trigger to be any time input pin A[3] changes.

To do this, click the ‘+’ beside ‘A’ in the large area in the middle of the window. This should expand the group ‘A’ into its four signals A[3], A[2], A[1] and A[0]. Click ‘A[3]’ to highlight it, then right-click in the right hand column (the trigger condition column) of this line. Select ‘Either Edge’ from the list that appears. This section should now appear as below.



Type	Alias	Name	Data Enable	Trigger Enable	Trigger Condition
		A	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1 Basic
		A[0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXb
		A[1]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXb
		A[2]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXb
		A[3]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	X
		B	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXb
		Numout	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXXXXXXXb
		Select	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXb


In the top-right hand corner of the SignalTap window, click in the ‘Hardware’ box. The USB-Blaster should appear – if it does so, select it. If it does not appear, click the ‘Setup’ button, and locate it there.

Compile again, and program to the board.

SignalTap should now be ready to run.

## Run SignalTap Analysis




Click the  button to begin signal analysis, and click the ‘Data’ tab (beside the ‘Setup’ tab used earlier). SignalTap is now waiting for the trigger condition to be met. The trigger condition was either a low-to-high or a high-to-low transition on A[3]. If pins were assigned as specified, A[3] will be connected to switch SW9 on the board, so flick this switch, and signals should appear, showing all signals both before and after the trigger. Flick some of the other switches, and observe that the output changes on the LEDs.

When switches other than A[3] ( on SW9) are flicked, is there any change in the signals shown in SignalTap? Explain why.



## Setting More Suitable Trigger Conditions

For a combinational circuit such as this, it is more useful to see how outputs change given a change in more than one input. The trigger condition. It is possible to set this up in SignalTap. We will change the trigger condition such that the signals shown in SignalTap update every time one of the A inputs change.

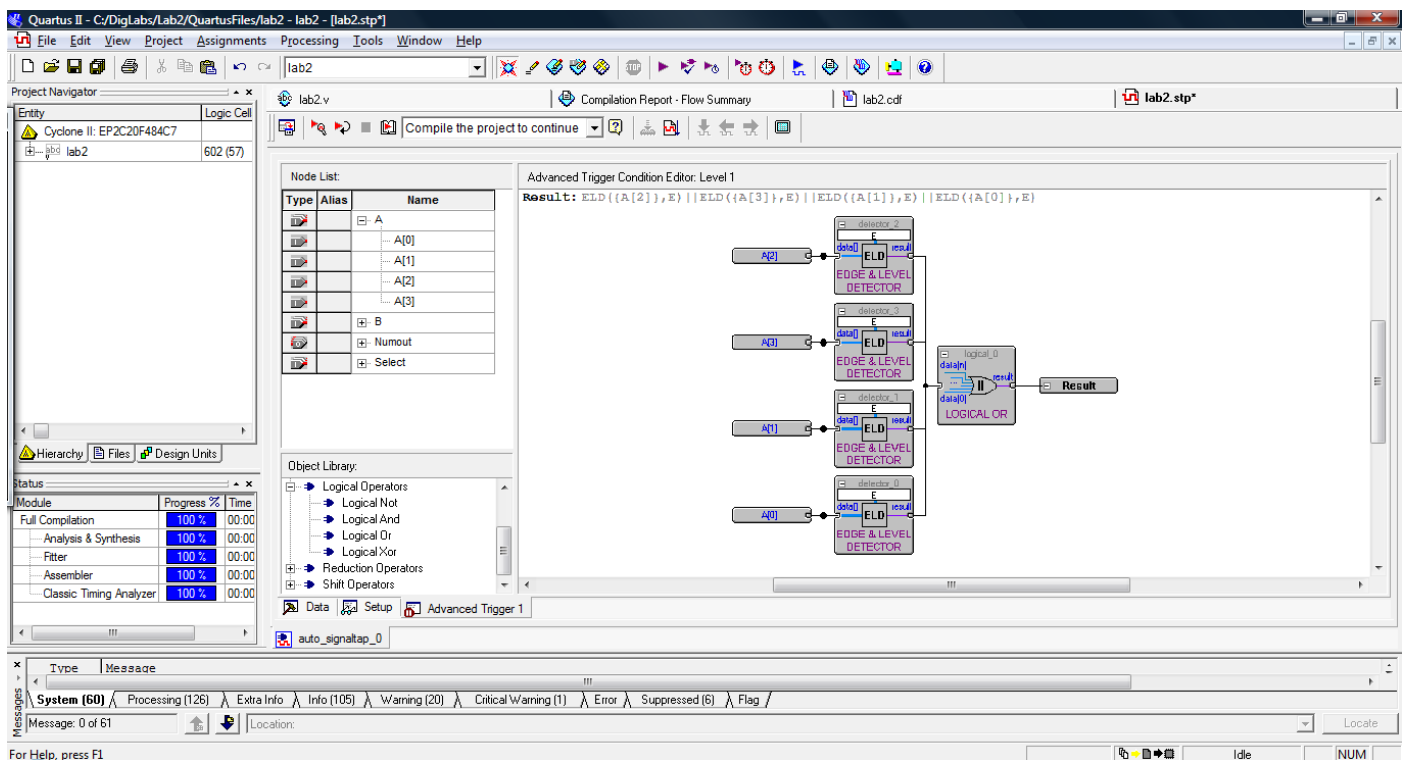
Firstly, stop the current analysis by clicking the  button.

Click the 'Setup' tab, and in the 'Trigger Condition' box, select 'Advanced', as shown below.


Node			Data Enable	Trigger Enable	Trigger Conditio...
Type	Alias	Name	19	19	1✓ Basic
		A	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Basic
		A[0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Advanced
		A[1]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		A[2]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		A[3]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	X
		B	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXb
		B[0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		B[1]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Data Setup

Now, click on the '+' beside 'A' to show the four signals that make up the group 'A'. Drag A[0] into the large white space to the right. Do the same with A[1], A[2], and A[3]. In the object library, below the signals, click 'Edge and Level Detector', and drag this into the large white space. Double-click on the symbol that appears, and type 'E' into the box, which means either edge. Add 3 more edge/level detectors, and enter 'E' for each of the in the same way. Finally, double-click 'Logical Operators', and a number of logical functions should appear. Click and drag 'Logical Or' into the white space to the right. Arrange the blocks in a similar way to that shown below.



Connect them together as shown, by dragging with the mouse.

Compile again, and program the design onto the board again. Start SignalTap analysis by clicking .

What happens to the signals shown when A[3], A[2], A[1] and A[0] are changed by flicking switches SW9 to SW6? Is this expected?

Modify the advanced trigger condition so that the SignalTap signals will change when any of the inputs (A, B or Select) are changed. Show this below. Does it give the expected result?

## Debugging Using SignalTap to Probe Internal Nodes

SignalTap can be useful in debugging, by monitoring internal signals. Sometimes these are available, but not always. When selecting signals, we have so far used only input and output pins, but others are available.

Click the 'Setup' tab, then double-click in the main area to add signals, as before. Set the filter to be 'SignalTap II: Post-Fitting' and click the 'List' button. It will be observed that many signals appear on the left, and these are all available for use. If you wish to monitor any of these, add them as before by clicking on them and then clicking the '>' button to move them to the right-hand side.

However, for debugging purposes, the most likely signals of interest will be those that were defined as internal nodes in the code. In this case, these nodes are:

```
reg [7:0] add_result, sub_result, mul_result, div_result;  
reg [7:0] min_result, max_result, ave_result, all_on_result;
```

It will be seen that these signals are all declared as 8 bits wide here, but do not appear as such in the Post-Fitting list. This is because of the way in which the design is physically implemented on the chip. If we wish to monitor these signals, we must declare them as outputs. To do so, modify the code to show these as outputs, by changing the first line to:

```
module lab2 (clock,A,B,Select,Numout,add_result, sub_result, mul_result, div_result,min_result, max_result,  
ave_result, all_on_result);
```

Then, below line 9 of the code, add the following:

```
output [7:0] add_result, sub_result, mul_result, div_result;  
output [7:0] min_result, max_result, ave_result, all_on_result;
```

Compile.

Note that we do not need to assign pins to these additional outputs, as they are only used for the logic analyser, rather than being real output signals.

In the SignalTap window, double click in the large area to add signals again. Set the filter to be 'Pins: All', and click the 'List' button. This time, the signals add\_result, sub\_result, mul\_result, div\_result, min\_result, max\_result, ave\_result, all\_on\_result should all appear. Add one or two of these.

Compile again, and program the design to the board. Run SignalTap analysis. Explain what happens to these signals as the inputs are changed.



**Prewritten Lab2code.txt text file to be copied into the Verilog source file and saved as "lab2.v".**

```
module lab2 (A,B,Select,Numout);    // lab2.txt on vision

    input [3:0] A;
    input [3:0] B;                //A and B are the two 4-bit inputs
    input [2:0] Select;           //Select is a 3-bit input, used to select one of 8 functions
    output [7:0] Numout;          //Numout is the 8-bit result that is sent to the output pins

    reg [7:0] Numout;              //Numout defined in an always block, so needs declared as register

    reg [7:0] add_result, sub_result, mul_result, div_result;
    reg [7:0] min_result, max_result, ave_result, all_on_result;
                                    //these variables are internal circuit nodes

    always @ (A,B)
    begin                            //beginning of always block, where signals depend on values of A and B
        add_result = A + B;
        sub_result = A - B;
        mul_result = A * B;
        div_result = A / B;

                                    // Simple calculation of all internal variables
                                    // all internal variables calculated simulatneously

        //*****
        //ADD EXTRA CODE HERE TO CALCULATE:
        //          min_result, max_result, ave_result, all_on_result
        //*****

    end

    always @ (Select, add_result, sub_result, mul_result, div_result, min_result, max_result, ave_result,
all_on_result)
    begin
        case (Select)
            3'b000: Numout = add_result;
            3'b001: Numout = sub_result;
            3'b010: Numout = mul_result;
            3'b011: Numout = div_result;

            //*****
            //ADD EXTRA CODE TO PASS min_result, max_result, ave_result, all_on_result
            //TO NUMOUT BASED ON THE OTHER POSSIBLE VALUES OF 'Select'
            //*****

        endcase

                                    //case statement based on value of 'Select' input
                                    //if (select==0) numout=add_result, etc

    end

endmodule
```