CprE 381, Computer Organization and Assembly Level Programming

Lab 1 Report

Student Name : Varun Advani

Submit a typeset pdf version of this on Canvas by the due date. Refer to the highlighted language in the lab document for the context of the following questions.

[Part 1.c] Think of three more cases and record them in your lab report.
Case 1 oY value same as the day of my birth:
Initialize weight:
Initialize a weight value register by setting *iW* to the desired value of 10 and *iLdW* to 1 prior to a positive edge of the clock and holding them until after the positive edge of the clock.
Perform one MAC:
With the above weight initialization to 10, I set the activation input, *iX*, to 1 and the partial sum input, *iY*, to 10. I make sure the weight is not changed by setting *iLdW* to 0.
I expect that, after two positive edges of the clock (i.e., two cycles), the activation output, *oX*, will be 1 and the partial sum output, *oY* will be 20=10*1+10.
Case 2:
Initialize a weight value register by setting *iW* to the desired value of 5 and *iLdW* to 1 prior to a positive edge of the clock and holding them until after the positive edge of the clock.
Perform one MAC:
With the above weight initialization to 5, I set the activation input, *iX*, to 6 and the partial sum input, *iY*, to 0. I make sure the weight is not changed by setting *iLdW* to 0.
I expect that, after two positive edges of the clock (i.e., two cycles), the activation output, *oX*, will be 4 and the partial sum output, *oY* will be 30=5*6+0.

Case 3:
Initialize a weight value register by setting *iW* to the desired value of 10 and *iLdW* to 1 prior to a positive edge of the clock and holding them until after the positive edge of the clock.
Perform one MAC:
With the above weight initialization to 10, I set the activation input, *iX*, to 0 and the partial sum input, *iY*, to 10. I make sure the weight is not changed by setting *iLdW* to 0.
I expect that, after two positive edges of the clock (i.e., two cycles), the activation output, *oX*, will be 0 and the partial sum output, *oY* will be 10=10*0+10.

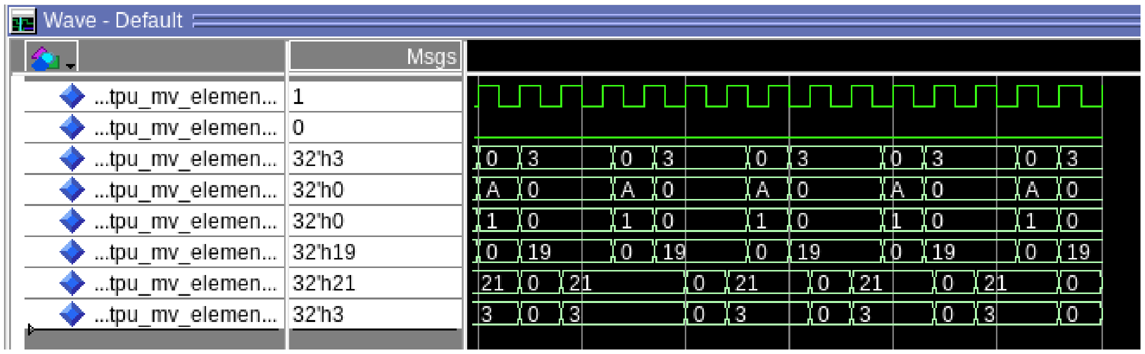
[Part 1.e] For labels 9, 20, 32, and 33, specify where (VHDL file and line number) these values are located – some will be found in more than one place. Also attempt to explain the functionality of each label as it occurs in the code
1. *in the attached diagram area, (9) is the oQ output of the regLd module which is a load register that loads the data value, which is defined as an out integer on Line 31 of RegLd.vhd and set on Line 51. For this specific instance of the RegLd module, the output oQ is connected to a signal called s_X1 on line 95 of TPU_MV_Element.vhd.*

2. *in the attached diagram area, (20) is the s_X1 output of the g_Delay1 module which is a module that delays the iX input by a clock cycle. It is defined on Line 70 of TPU_MV_element.vhd, and carries the signal of X.*

3. *in the attached diagram area, (32) is defined as g_add1 in the TPU vhd file, and is an adder that carries out the required addition operation in the given design. It is defined in the TPU_MV_element.vhd file on line 117 and is designed as the Adder.vhd file.*

4. *in the attached diagram area, (33) is the design of the TPU module in its entirety. It is designed as the TPU_MV_Element.vhd and contains all the logic and circuitry for the design.*

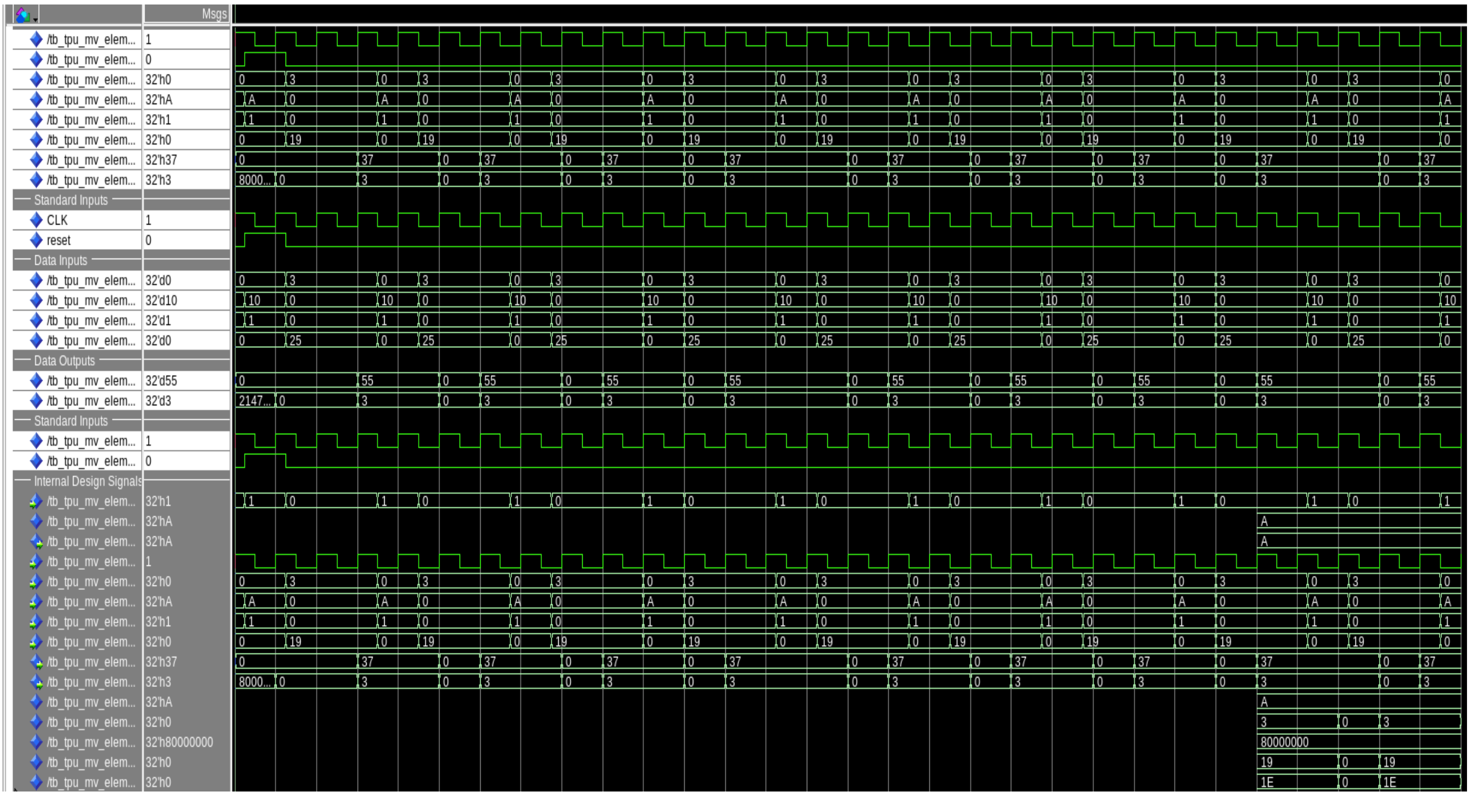
[Part 1.g.v] In your lab report, include a screenshot of the waveform. Describe, in plain English, any differences between what you expected and what the simulation showed.


Wave - Default

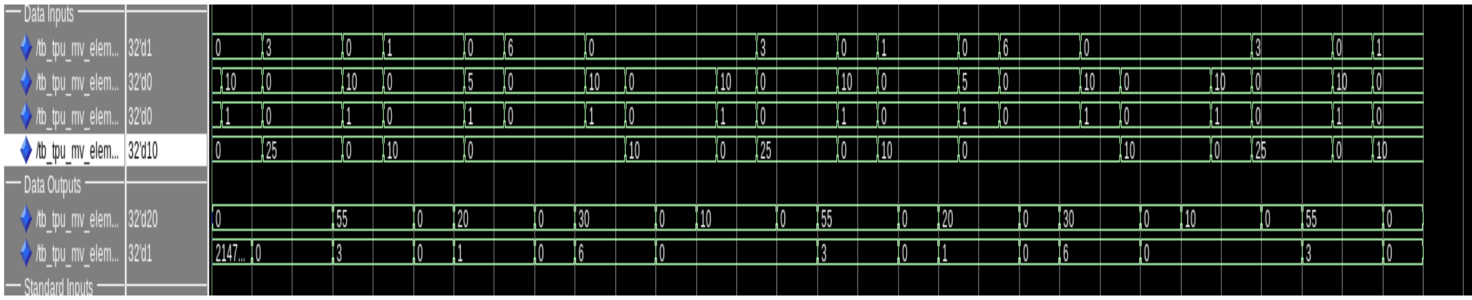
	Msgs
...tpu_mv_elemen...	1
...tpu_mv_elemen...	0
...tpu_mv_elemen...	32'h3
...tpu_mv_elemen...	32'h0
...tpu_mv_elemen...	32'h0
...tpu_mv_elemen...	32'h19
...tpu_mv_elemen...	32'h21
...tpu_mv_elemen...	32'h3

With the above simulation values for iX, iY, and iW, I expected oY to give me an output of 37 however, the output is 21 which is incorrect. This is because in level 2 of the TPU design, iD gets its value from iY however, the VHDL given to us has it setup in such a way that iD gets its value from iX.

[Part 1.h] In your lab report, include a screenshot of the waveform. Describe, in plain English, how your waveform matches the expected result (e.g., reference the specific cycles and times). In your submission zip file, provide the completed *TPU_MV_Element.vhd* file in a folder called ‘MAC’.



The waveform does match my expectation as the value computed by the TPU for the test case given in the lab document is 55 which matches the value that oY generates in the above waveform. The waveform for another test case is as follows:



The outputs for my 3 cases that is: 20 (my birthday), 30, and 10 can be seen in the waveform above.

[Part 3.a] Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 2:1 mux. Include this in your lab report.

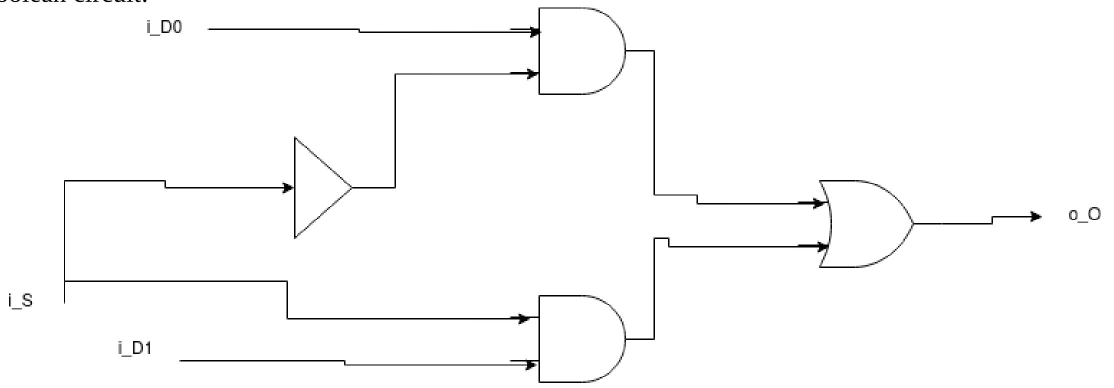
Truth Table:

S	D0	D1	Output
0	0	X	0
0	1	X	1
1	X	0	0

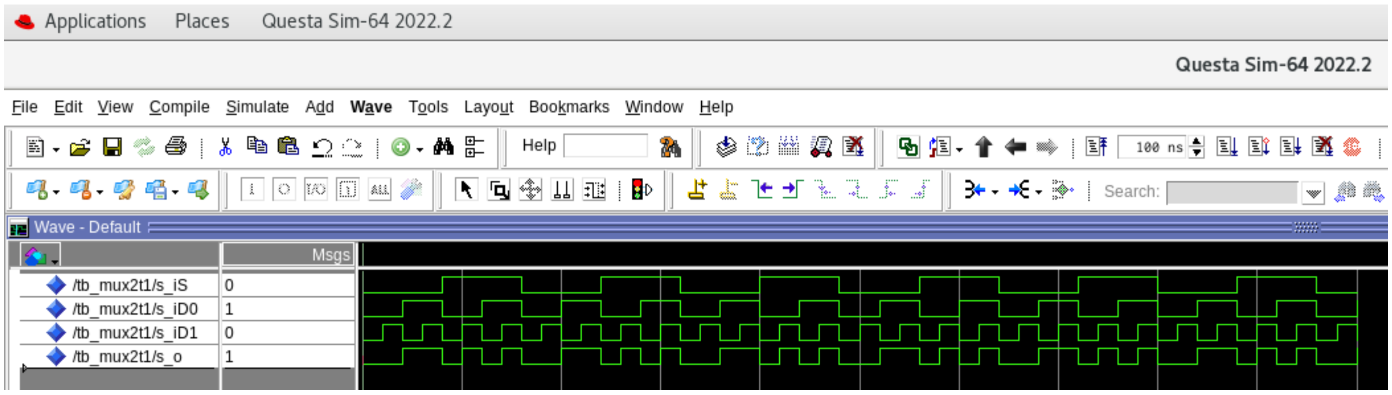
1	X	1	1

Boolean equation: i_S . ~i_D0 + i_S .i_D1 = o_O

Boolean circuit:



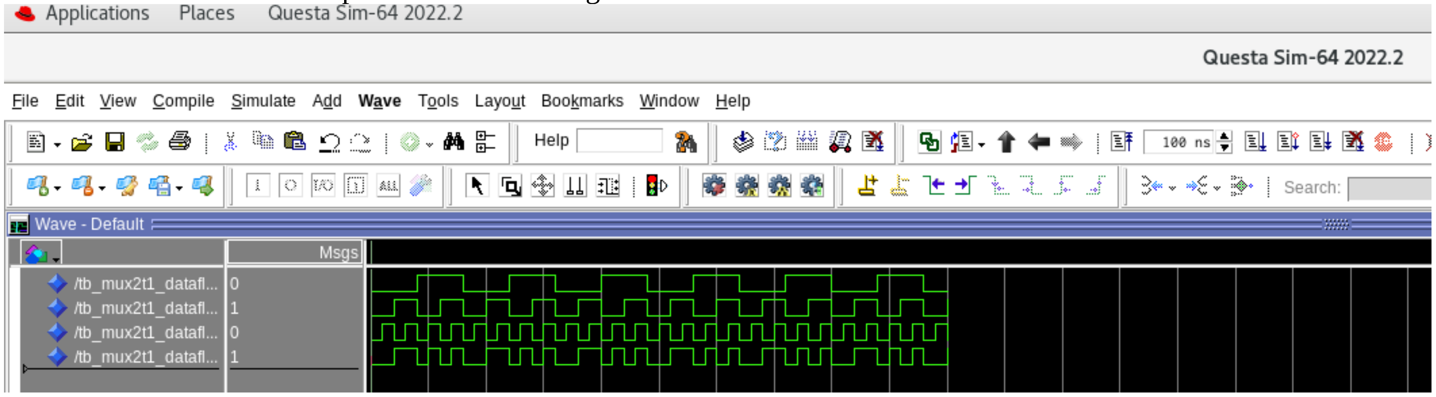
[Part 3.d] In your lab report, include a screenshot of the waveform. Make sure to label the screenshot with which module it is testing.


Applications Places Questa Sim-64 2022.2
Questa Sim-64 2022.2
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Help
100 ns
Search:
Wave - Default
Msgs

	Msgs
/tb_mux2t1/s_iS	0
/tb_mux2t1/s_iD0	1
/tb_mux2t1/s_iD1	0
/tb_mux2t1/s_o	1

2:1 MUX using structural VHDL

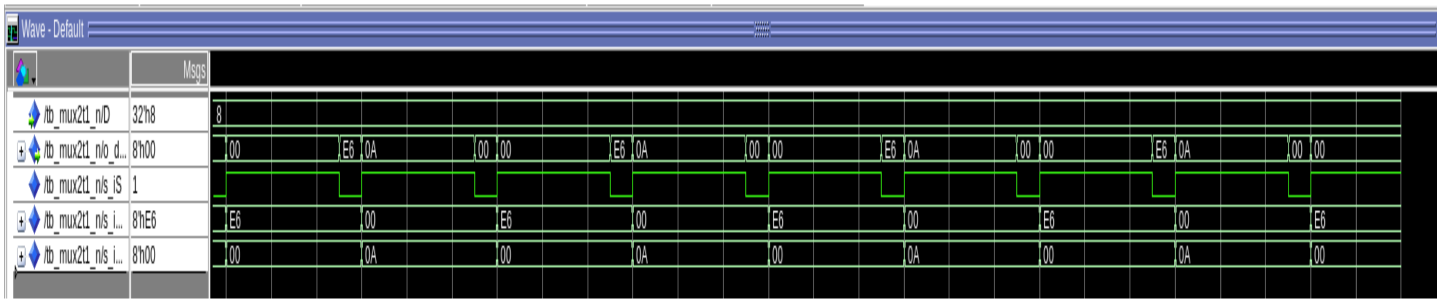
[Part 3.e] Again, in your lab report, include a labeled screenshot of the waveform showing the dataflow mux implementation working.


Applications Places Questa Sim-64 2022.2
Questa Sim-64 2022.2
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Help
100 ns
Search:
Wave - Default
Msgs

	Msgs
/tb_mux2t1_datafl...	0
/tb_mux2t1_datafl...	1
/tb_mux2t1_datafl...	0
/tb_mux2t1_datafl...	1

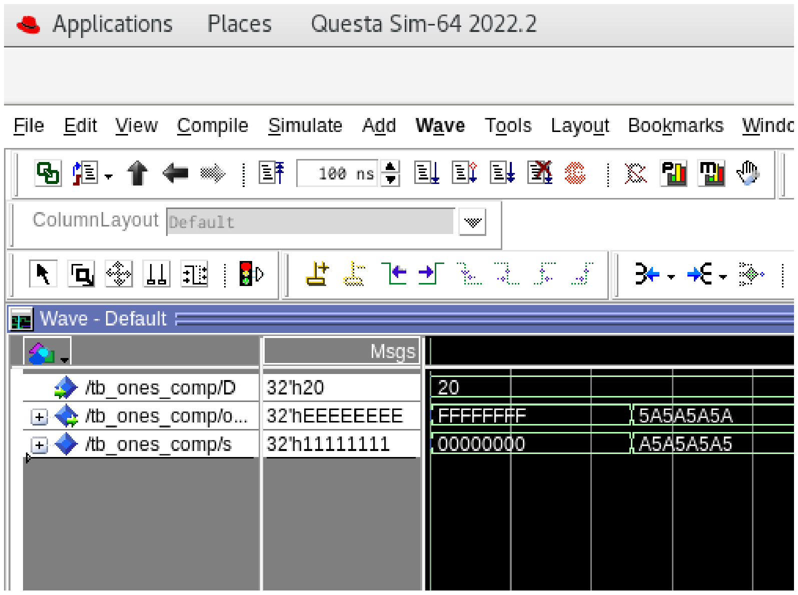
2:1 MUX using dataflow VHDL

[Part 4] Include a waveform screenshot and corresponding description demonstrating it is working correctly.


Wave - Default
Msgs
/tb_mux2t1_n/D 32'h8
/tb_mux2t1_n/o_d... 8'h00
/tb_mux2t1_n/s_iS 1
/tb_mux2t1_n/s_i... 8'hE6
/tb_mux2t1_n/s_i... 8'h00
8
00 E6 0A 00 00 E6 0A 00 00 E6 0A 00 00 E6 0A 00 00
E6 00 E6 00 E6 00 E6 00 E6
00 0A 00 0A 00 0A 00 0A 00

The above testbench simulates an N bit 2:1 MUX that selects between E6 and 00 in test 1, and 00 and 0A in test 2. The output data is as expected depending on the select line value that keeps alternating between 0 and 1. Data flow for the above operation was carried out in a file mux2t1_N_dataflow.

[Part 5.b] Include a waveform screenshot and description in your lab report.



One's compliment negate's each bit of the value as can be seen in the two test cases above.

[Part 6.a] A full adder takes three single-bit inputs and produces two single-bit outputs – a sum and carry for the addition of the three input bits. Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 1-bit full adder. Include this in your report.

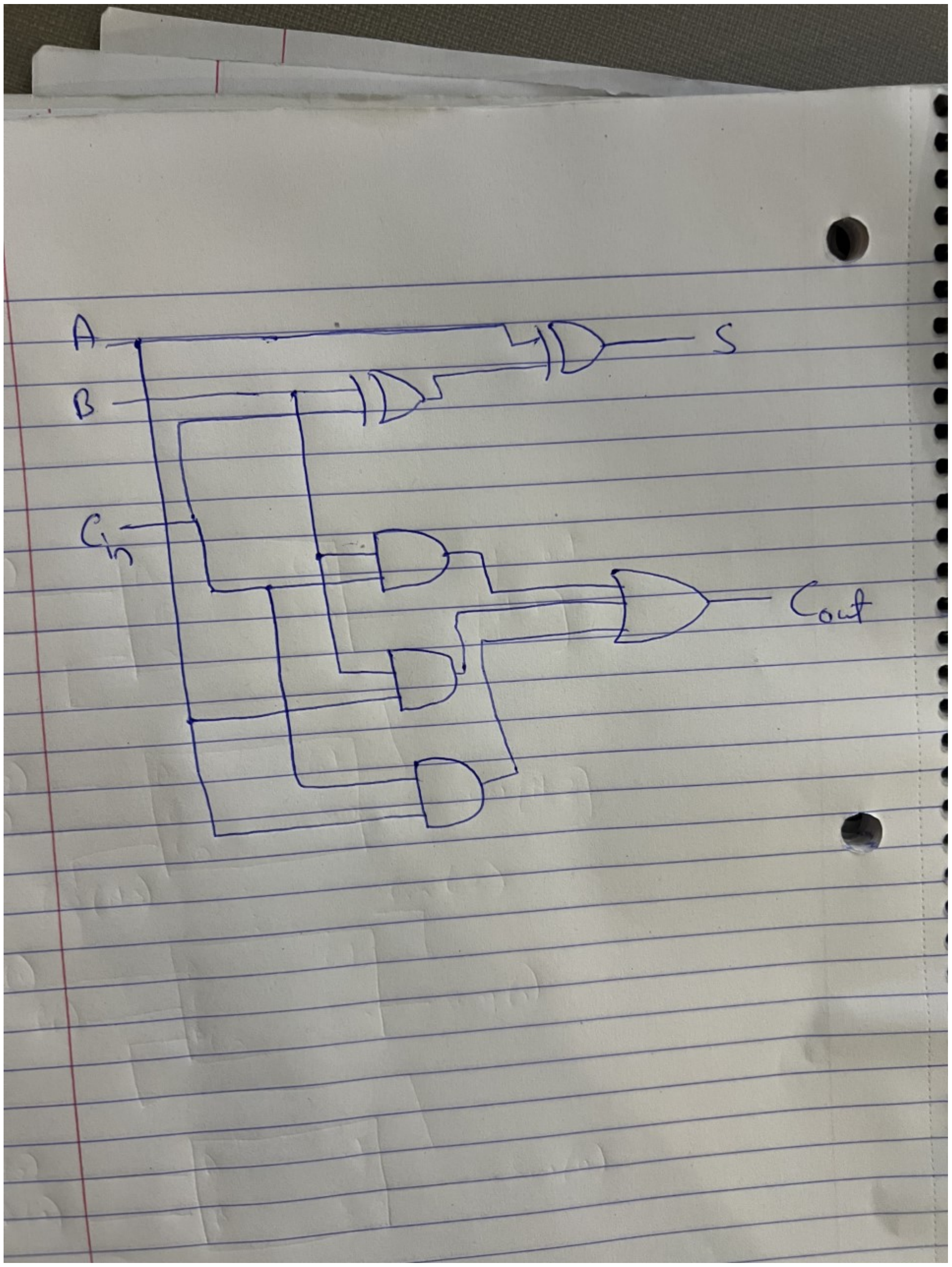
Truth table:

A	B	Cin	Sum	Cout
0	0	0	0	0

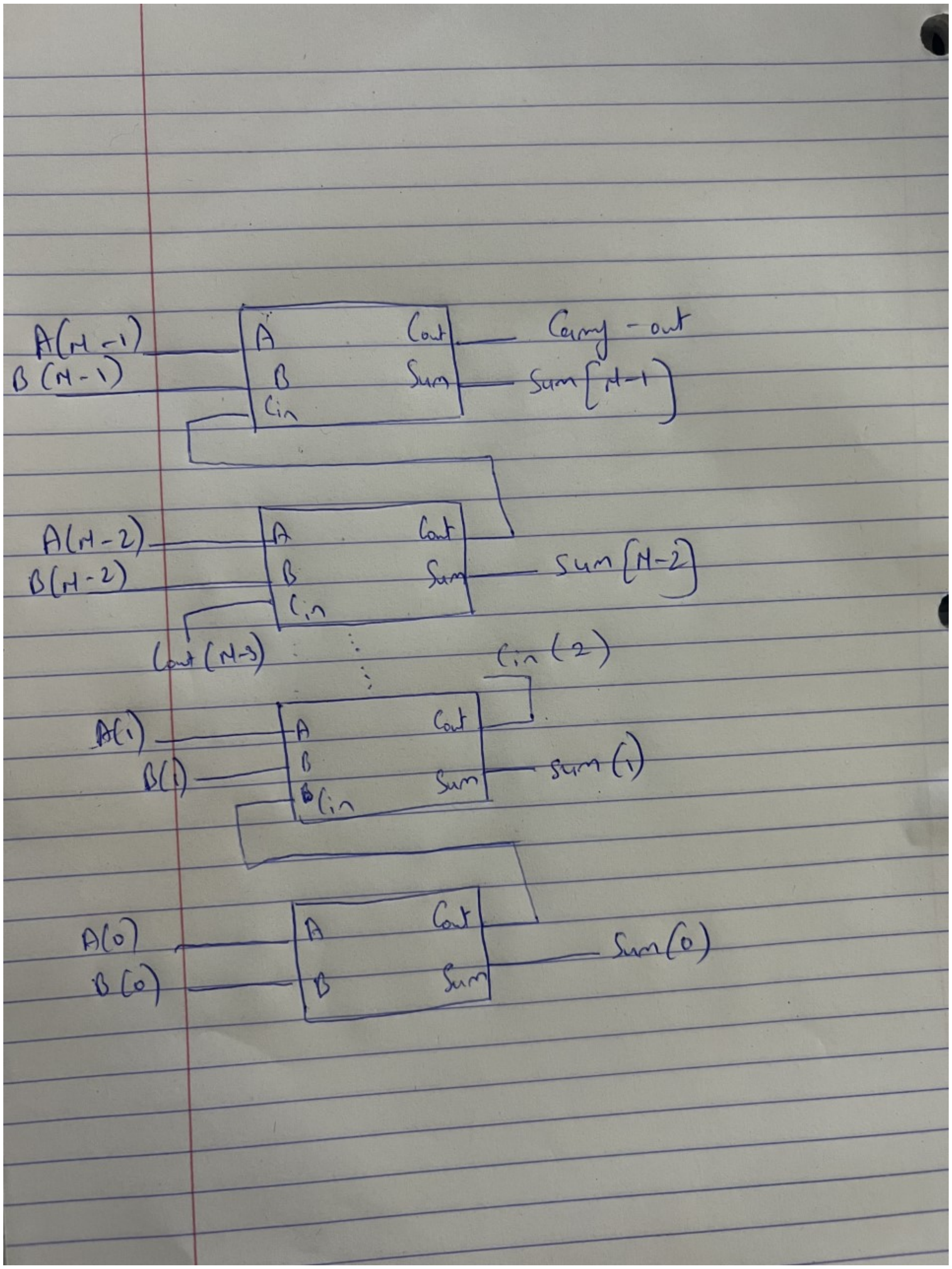
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean Expression:
Sum = A ^(B ^ C) [^ is XOR]
Cout = B*Cin + A*Cin + AB [* is AND and + is OR]

Boolean circuit:

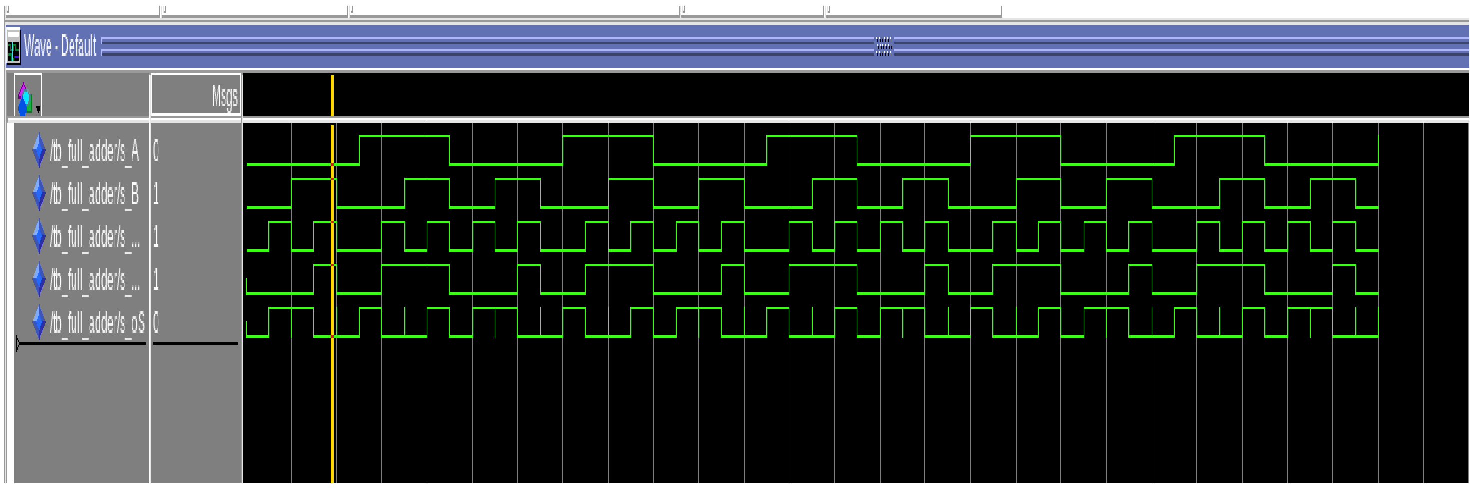


[Part 6.c] Then draw a schematic of the intended design, including inputs and outputs and at least the 0, 1, N-2, and N-1 stages. Include this in your report.

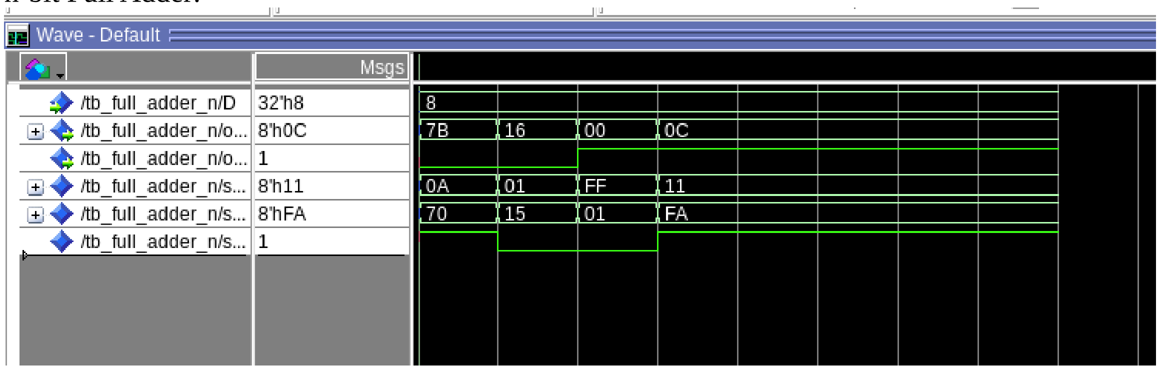


[Part 6.d] Include an annotated waveform screenshot in your write-up.

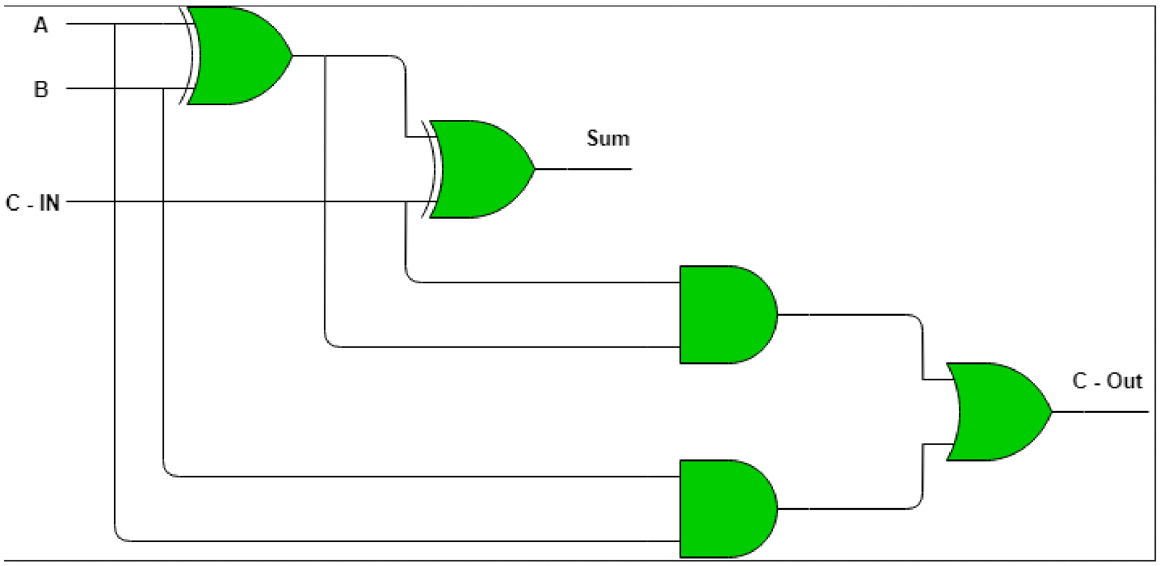
Full Adder:



n-bit Full Adder:

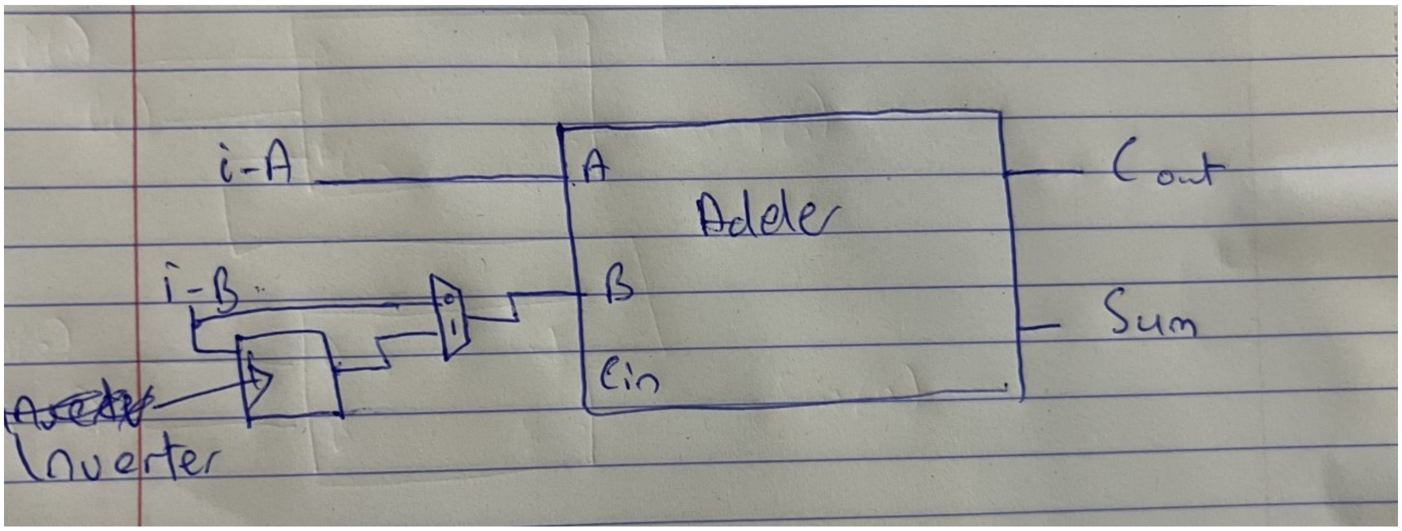


	Msgs				
/tb_full_adder_n/D	32'h8	8			
/tb_full_adder_n/o...	8'h0C	7B	16	00	0C
/tb_full_adder_n/o...	1				
/tb_full_adder_n/s...	8'h11	0A	01	FF	11
/tb_full_adder_n/s...	8'hFA	70	15	01	FA
/tb_full_adder_n/s...	1				

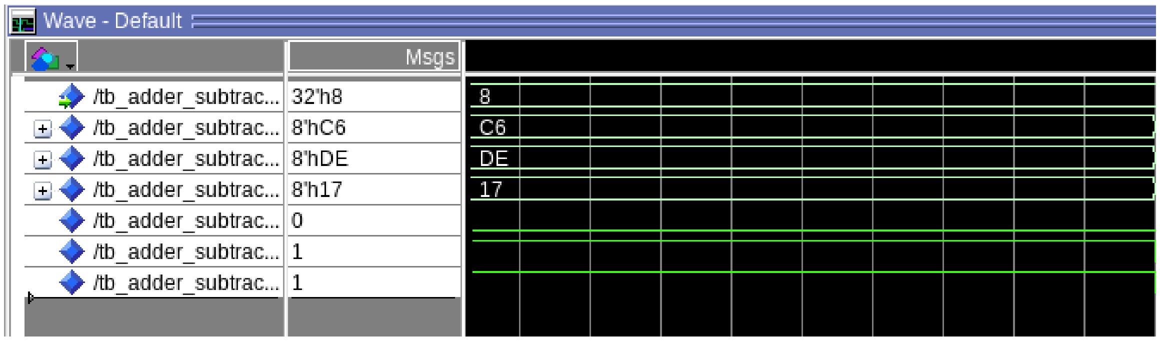


The waveform complies with the truth table written above. Since a 3 input or gate vhdl file was not available in the given zip file, I used the circuit diagram attached below for the 1-bit full adder:

[Part 7.a] Draw a schematic (don't use a schematic capture tool) showing how an N-bit adder/subtractor with control can be implemented using <u>only</u> the three main components designed in earlier parts of this lab (i.e., the N-bit inverter, N-bit 2:1 mux, and N-bit adder). How is the 'nAdd_Sub' bit used? Include this in your report.

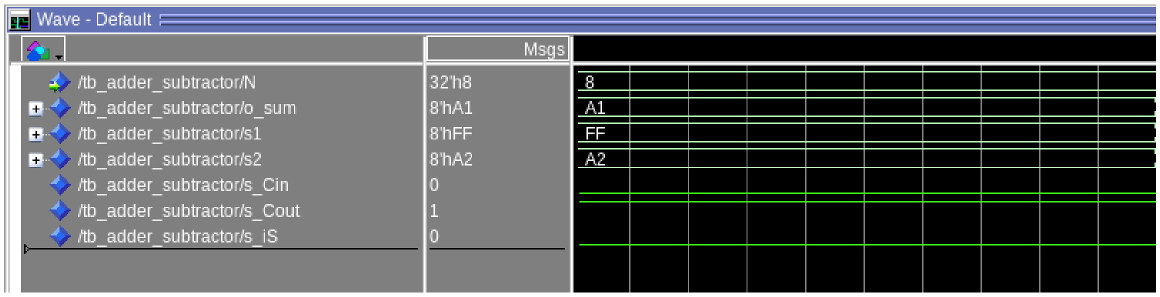


[Part 7.c] Provide multiple waveform screenshots in your write-up to confirm that this component is working correctly. What test-cases did you include and why?



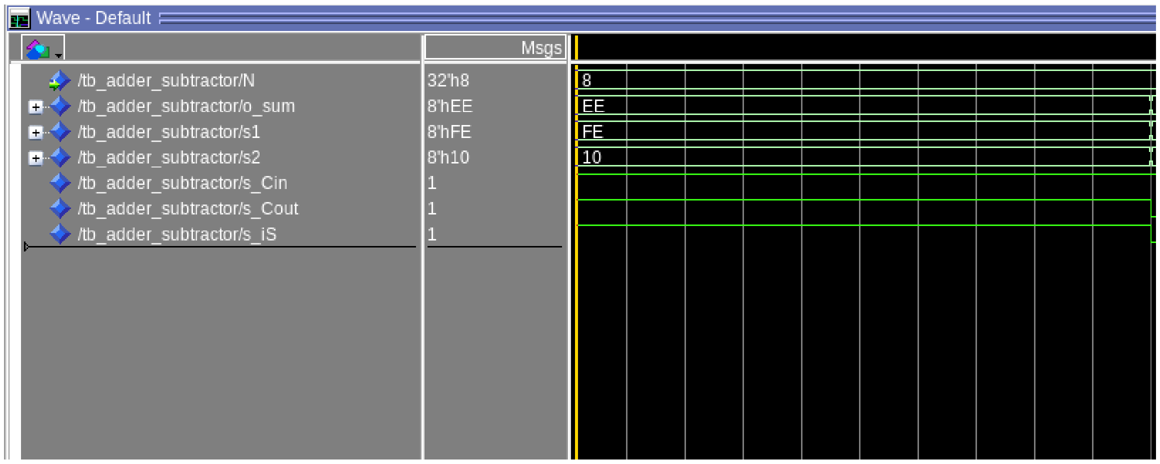
Wave - Default

	Msgs	
/tb_adder_subtrac...	32'h8	8
/tb_adder_subtrac...	8'hC6	C6
/tb_adder_subtrac...	8'hDE	DE
/tb_adder_subtrac...	8'h17	17
/tb_adder_subtrac...	0	
/tb_adder_subtrac...	1	
/tb_adder_subtrac...	1	



Wave - Default

	Msgs	
/tb_adder_subtractor/N	32'h8	8
/tb_adder_subtractor/o_sum	8'hA1	A1
/tb_adder_subtractor/s1	8'hFF	FF
/tb_adder_subtractor/s2	8'hA2	A2
/tb_adder_subtractor/s_Cin	0	
/tb_adder_subtractor/s_Cout	1	
/tb_adder_subtractor/s_iS	0	



Wave - Default

	Msgs	
/tb_adder_subtractor/N	32'h8	8
/tb_adder_subtractor/o_sum	8'hEE	EE
/tb_adder_subtractor/s1	8'hFE	FE
/tb_adder_subtractor/s2	8'h10	10
/tb_adder_subtractor/s_Cin	1	
/tb_adder_subtractor/s_Cout	1	
/tb_adder_subtractor/s_iS	1	

I inluded 1 test case with the select input 0 which indicates that the bits are not inverted by the inverter and hence the design acts as an adder. Similarly, I added 1 test case with select input 1 which indicates that the bits are inverted to ones complement and hence the circuit acts as a subtracter. The third test case I included was a case with the carry in bit being ‘1’.