

CprE 381, Computer Organization and Assembly Level Programming

Lab 1 Report

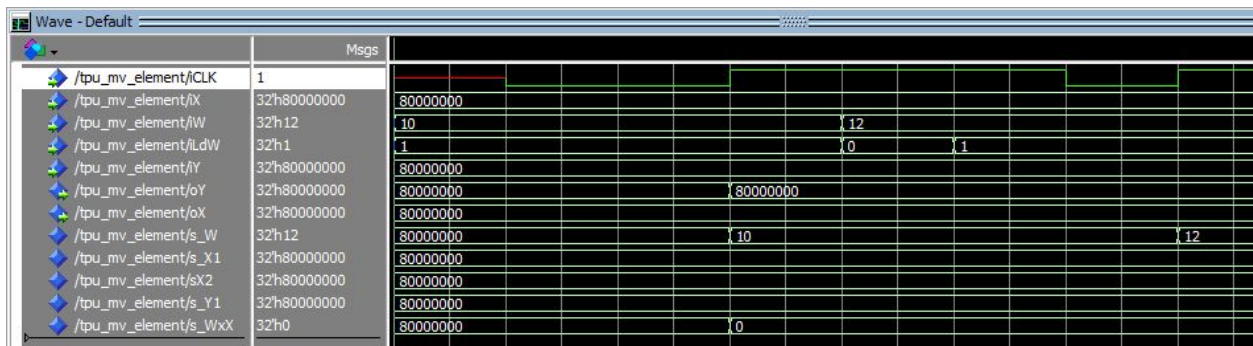
Student Name Declan Costello

Student Name Adnan Salihovic

Lab Partners # 2

Submit a typeset pdf version of this on Canvas by the due date (i.e., the start of your next lab section). Refer to the highlighted language in the lab 1 instruction for the context of the following questions.

EXAMPLE CASES:

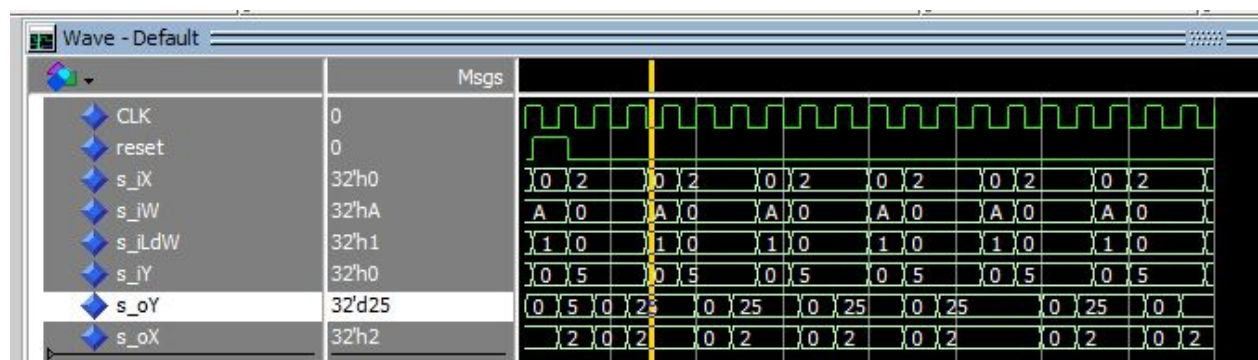


[Part 2.c] Think of three more cases and record them in your lab report.

CASE 3: THE ONE THAT EQUALS X (25)

```
--OUR CASE 3 |
-- Test case 3:
-- Initialize weight value to 10.
s_iX <= 0; -- Not strictly necessary, but this makes the testcases easier to read
s_iW <= 10;
s_iLdW <= 1;
s_iY <= 0; -- Not strictly necessary, but this makes the testcases easier to read
wait for gCLK_HPER*2;
-- Expect: s_W internal signal to be 10 after positive edge of clock

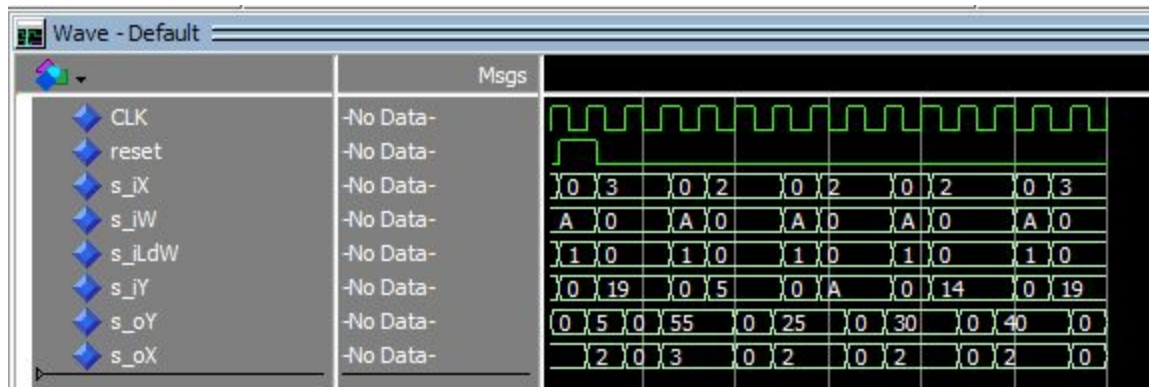
-- Test case 3.2:
-- Perform average example of an input activation of 3 and a partial sum of 25. The weight is still 10.
s_iX <= 2;
s_iW <= 0; -- Not strictly necessary, but this makes the testcases easier to read
s_iLdW <= 0; -- Make sure we don't continue to load.
s_iY <= 5;
wait for gCLK_HPER*2;
wait for gCLK_HPER*2;
```



CASE 4 AND 5:

4: THE ONE THAT EQUALS X (30)

5: THE ONE THAT EQUALS X (40) (Both cases 4 and 5 can be seen in the wave)



^30 and ^40 right above

in second to last line

```

-----
--OUR CASE 4; 20+10 =30
-- Test case 4:
-- Initialize weight value to 10.
s_iX  <= 0; -- Not strictly necessary, but this makes the testcases easier to read
s_iW  <= 10;
s_iLdW <= 1;
s_iY  <= 0; -- Not strictly necessary, but this makes the testcases easier to read
wait for gCLK_HPER*2;
-- Expect: s_W internal signal to be 10 after positive edge of clock

-- Test case 4.2:
-- Perform average example of an input activation of 3 and a partial sum of 25. The weight is still 10.
s_iX  <= 2;
s_iW  <= 0; -- Not strictly necessary, but this makes the testcases easier to read
s_iLdW <= 0; -- Make sure we don't continue to load.
s_iY  <= 10;
wait for gCLK_HPER*2;
wait for gCLK_HPER*2;
-----
--OUR CASE 5; 20+20 =40
-- Test case 5:
-- Initialize weight value to 10.
s_iX  <= 0; -- Not strictly necessary, but this makes the testcases easier to read
s_iW  <= 10;
s_iLdW <= 1;
s_iY  <= 0; -- Not strictly necessary, but this makes the testcases easier to read
wait for gCLK_HPER*2;
-- Expect: s_W internal signal to be 10 after positive edge of clock

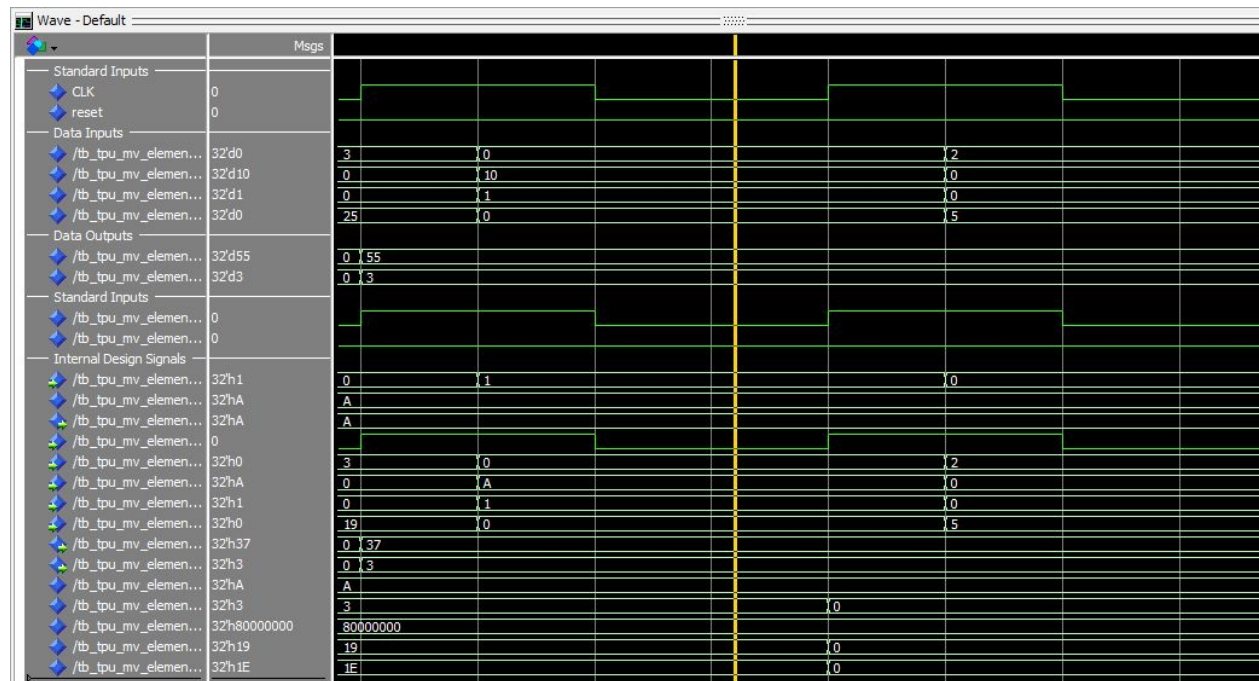
-- Test case 5.2:
-- Perform average example of an input activation of 3 and a partial sum of 25. The weight is still 10.
s_iX  <= 2;
s_iW  <= 0; -- Not strictly necessary, but this makes the testcases easier to read
s_iLdW <= 0; -- Make sure we don't continue to load.
s_iY  <= 20;
wait for gCLK_HPER*2;
wait for gCLK_HPER*2;

```

[Part 2.e] For labels 19, 24, 30, 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

- in the attached diagram area, (19) is the s_Y1 signal of the g_Delay2 module, which is defined as an integer on Line 31 of TPU_MV_ELEMENT.vhd and set on Line 72. For this specific instance of the multiplier module, the signal s_Y1 is connected to an input called iB on line 120 of TPU_MV_Element.vhd
- in the attached diagram area, (24) is the oC output of the g_Add1 module, which is defined as an integer on Line 31 of Adder.vhd and set on Line 41. For this specific instance of the Adder module, the output oC is connected to an output called oY on line 121 of TPU_MV_Element.vhd
- in the attached diagram area, (30) is the g_Delay2 module which is defined as an entity on Line 26 of Reg.vhd and set on lines 34-44. For this specific instance of the module, the g_Delay2 module is connected to a signal called s_Y1 on line 100 of the TPU_MV_ELEMENT.vhd.
- in the attached diagram area, (33) is the TPU_MV_ELEMENT entity. It is defined across lines 23-124. The specific instance of the entity is constructed of the modules mentioned before as well as many others.

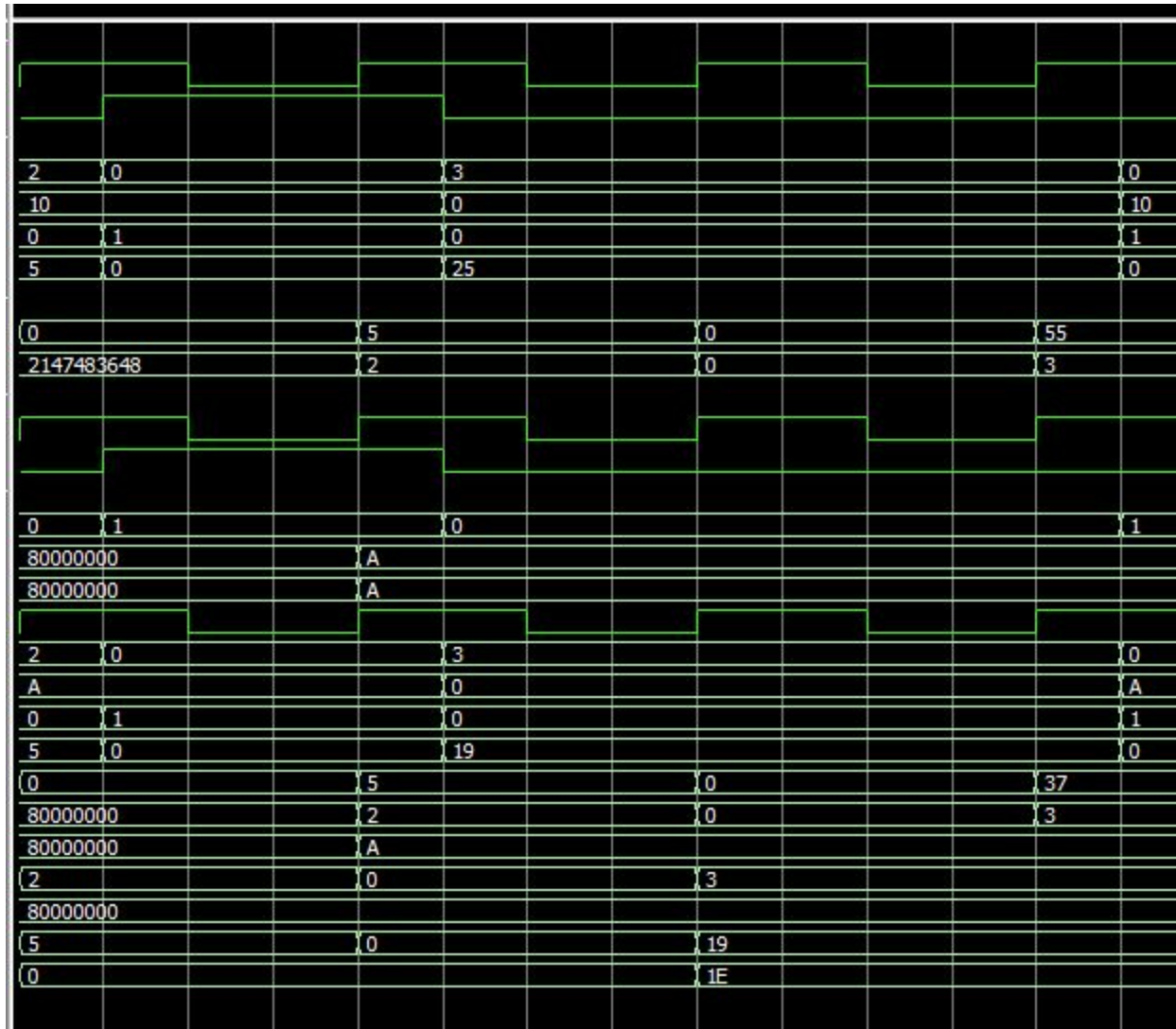
[Part 2.g.v] In your lab report, include a screenshot of the waveform. Describe, in plain English, any differences between what you expected and what the simulation showed.



[Part 2.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'.

EXPLANATION: the do file compiled test case by test case. Each individual test case took 3 full clock cycles to compile out test case to the correct result shown below.

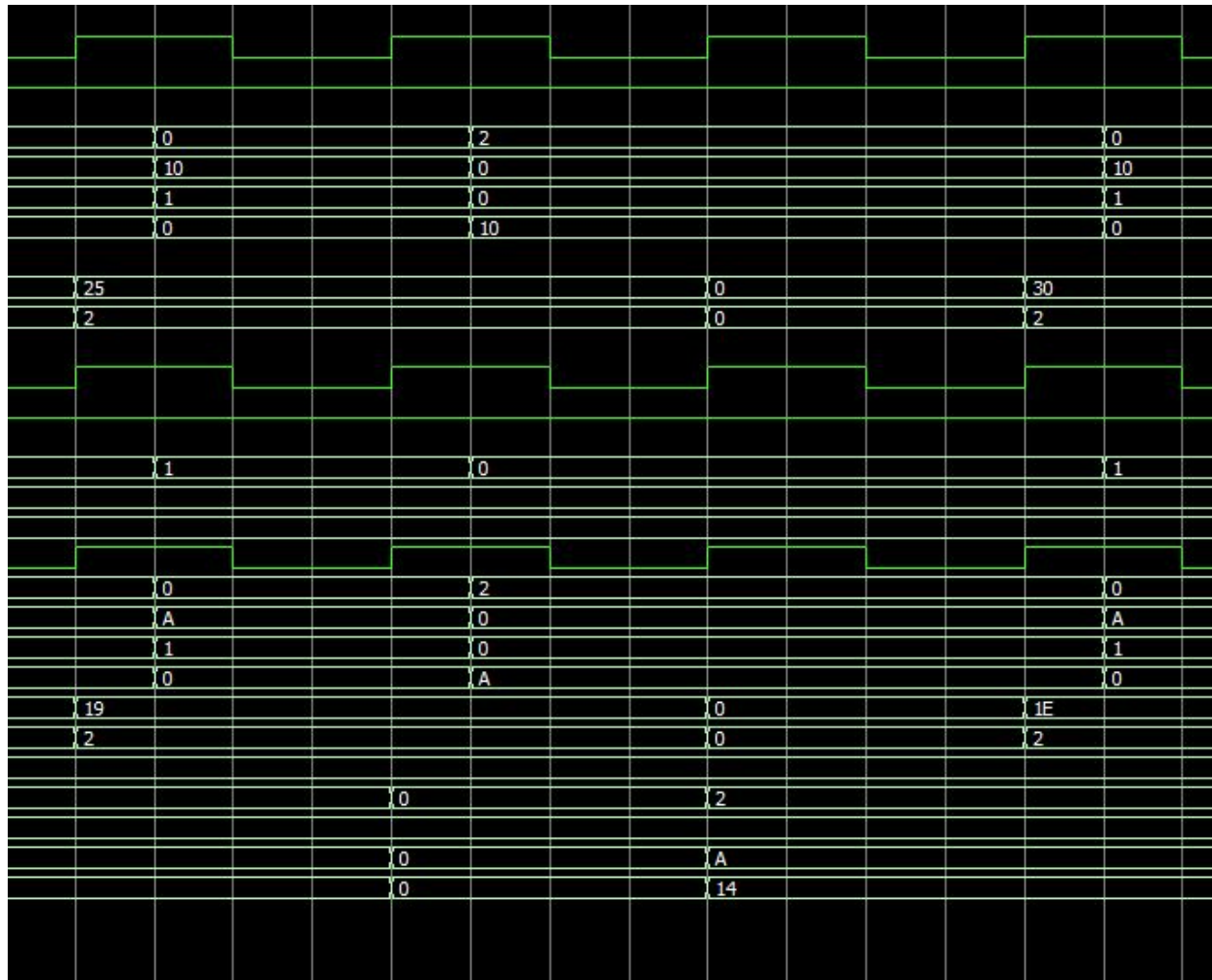
RESULT 1:



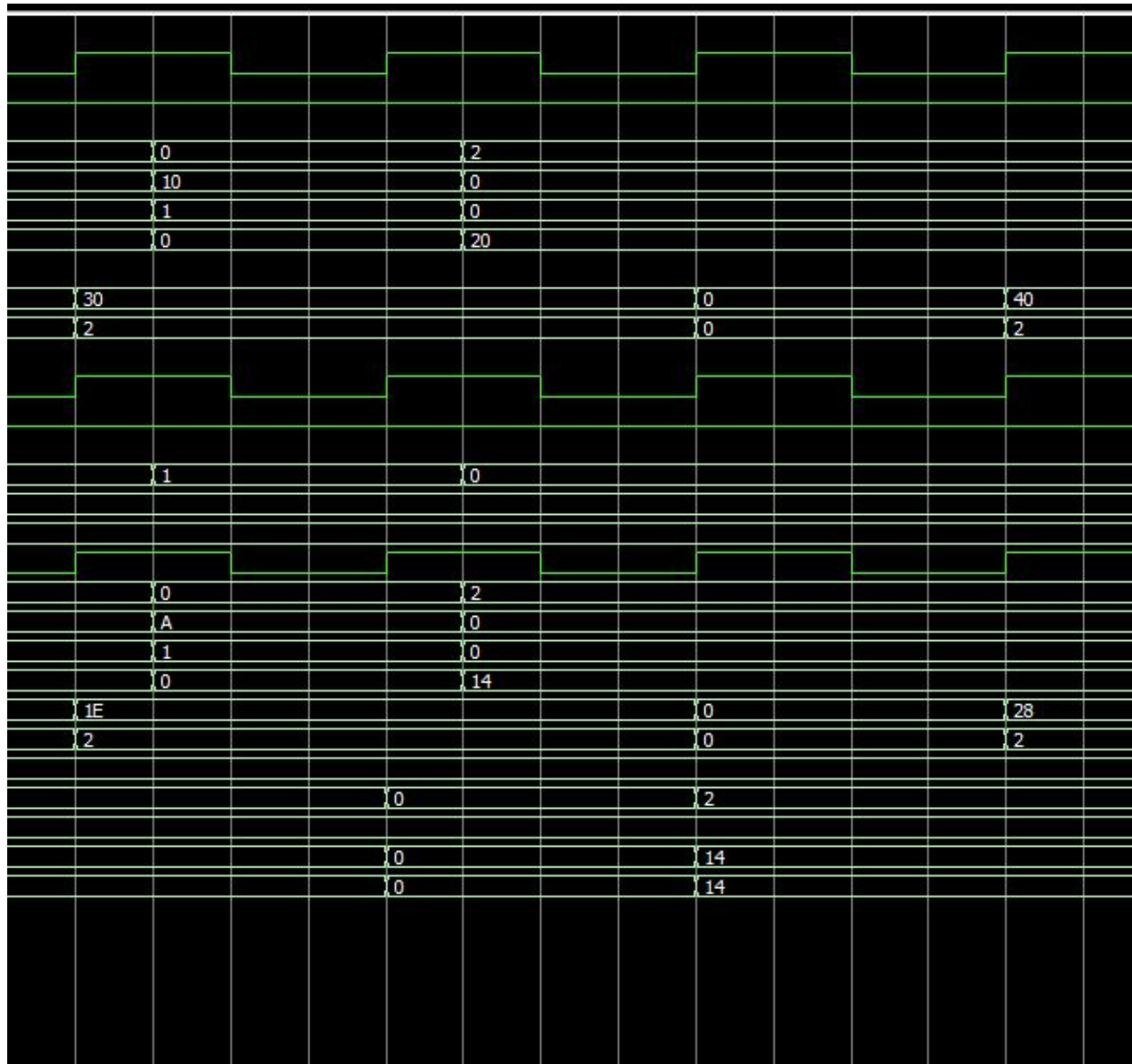
RESULT 2:

		0			2							0		
		10			0							10		
		1			0							1		
		0			5							0		
	55							0				25		
	3							0				2		
		1			0							1		
		0			2							0		
		A			0							A		
		1			0							1		
		0			5							0		
	37							0				19		
	3							0				2		
					0			2						
					0			5						
					0			14						

RESULT 3:



RESULT 4:



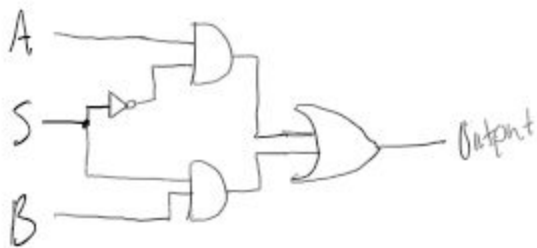
[Part 3.a] Draw the truth table, Boolean equation, and Boolean circuit equivalent (using only two-input gates) that implements a 2:1 mux. Include this in your lab report.

S	A	B	out	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\bar{S}A\bar{B}$
0	1	1	1	$\bar{S}AB$
1	0	0	0	
1	0	1	1	$S\bar{A}B$
1	1	0	0	
1	1	1	1	SAB

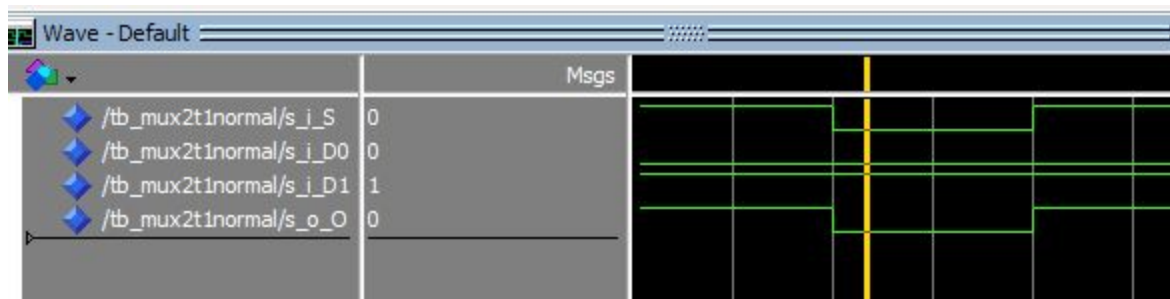
$\bar{S}A$

SB

$$\text{out} = \bar{S}A + SB$$



[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.



```

P_TEST_CASES: process
begin
wait for 100 ns ; -- for waveform c.

-----

-- Test case 1:
-- Initialize weight value to 10.
s_i_S   <= '1'; -- Not strictly 1
s_i_D0  <= '0';
s_i_D1 <= '1';
-- --do we need o_o output? -- No
wait for 100 ns;
-- Expect: s_W internal signal to

-- Test case 2:
-- Perform average example of an :
s_i_S   <= '0'; -- Not strictly 1
s_i_D0  <= '0';
s_i_D1 <= '1';
--do we need o_o
-- wait for gCLK_HPER*2;
-- wait for gCLK_HPER*2;

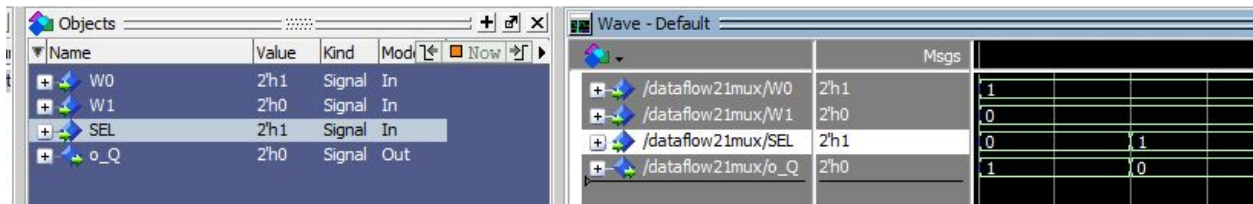
-- Expect: o_Y output signal to be

-- TODO: add test cases as needed
end process;

end mixed;
```

As you can see in the waveform above, when s_i_S is 1 the output is equal to s_i_D1 and when s_i_S is 0 the output is equal to s_i_D0.

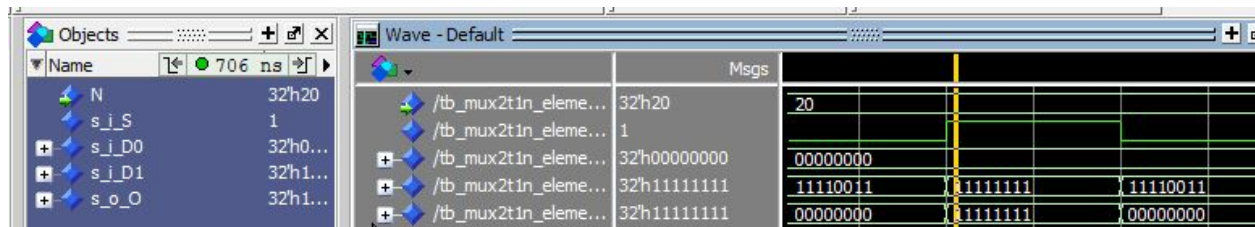
[Part 3.e] Again, in your lab report, include a labeled screenshot of the waveform showing the dataflow mux implementation working.



As shown in the wave form, when SEL is set to 0 the output o_o is set to the value of W0 and when SEL is 1 the value of the output is the value of W1.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity dataflow21mux is
5
6  port( W0, W1: in std_logic_vector(1 downto 0);
7        SEL: in std_logic_vector(1 downto 0);
8        o_Q : out std_logic_vector(1 downto 0));
9  end dataflow21mux;
10
11 architecture dataflow of dataflow21mux is
12
13 begin
14   with SEL select
15     o_Q <=
16     W0 when "00",
17     W1 when "01",
18
19     (others=>'0') when others;
20 end dataflow;
```

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



```
-- TODO: add test cases as needed.
P_TEST_CASES: process
begin
    wait for 100 ns ; -- for waveform c

    -----

    -- Test case 1:
    -- Initialize weight value to 10.
    s_i_S    <= '1'; -- Not strictly ne
    s_i_D0   <= X"0000000000";
    s_i_D1  <= X"1111111111";
    -- --do we need o_o output? -- Not
    wait for 100 ns;
    -- Expect: s_W internal signal to b

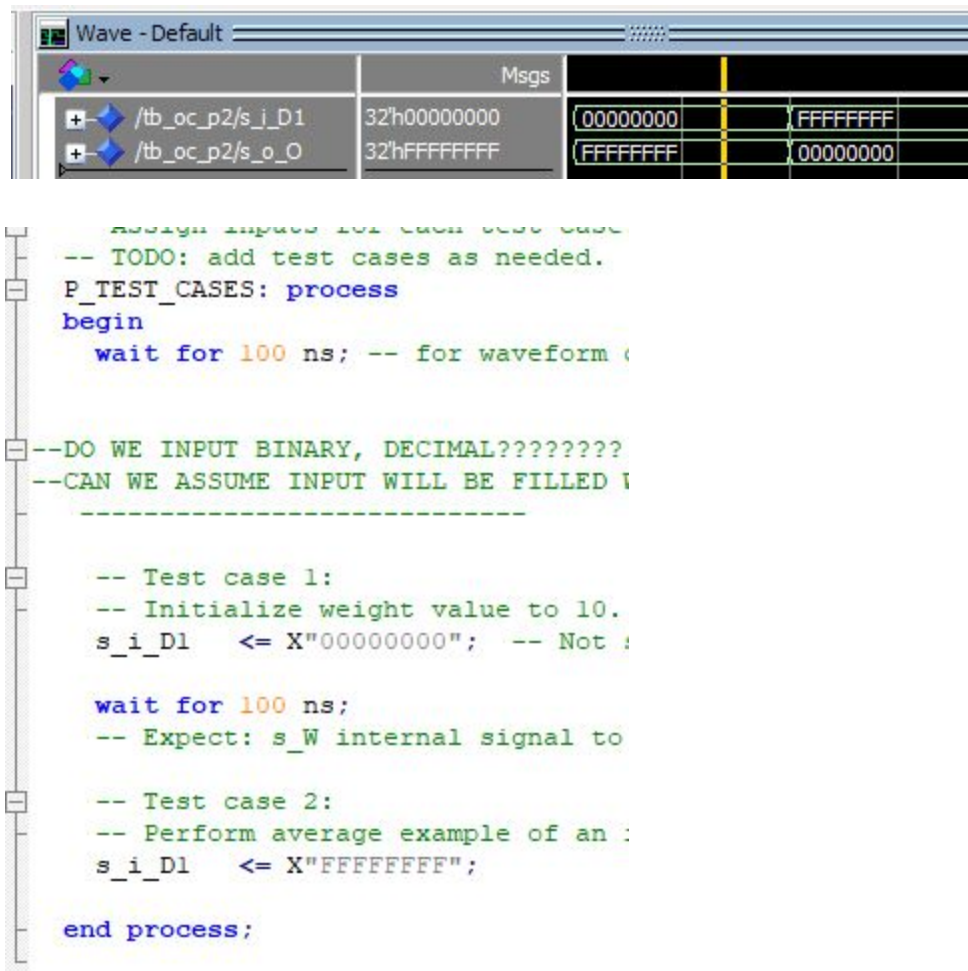
    -- Test case 2:
    -- Perform average example of an in
    s_i_S    <= '0'; -- Not strictly ne
    s_i_D0   <= X"0000000000";
    s_i_D1  <= X"1111100111";
    --do we need o_o
    -- wait for gCLK_HPER*2;
    -- wait for gCLK_HPER*2;

    -- Expect: o_Y output signal to be

    -- TODO: add test cases as needed (
```

When select is 0 the output s_o_O is outputting s_i_D0 and when the select is 1 the output is s_i_D1. This can be seen in the waveform above.

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



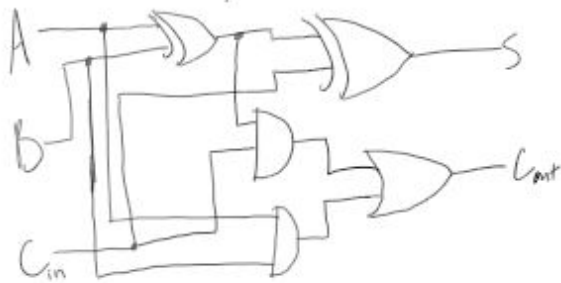
The Ones Complementor takes a given 32bit input and inverses all of the individual bits. This can be seen in the waveform above where the beginning input is 0 and when the converter is triggered the 0's becoming F's signifying that each individual bit was changed to a 1 in binary.

[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.

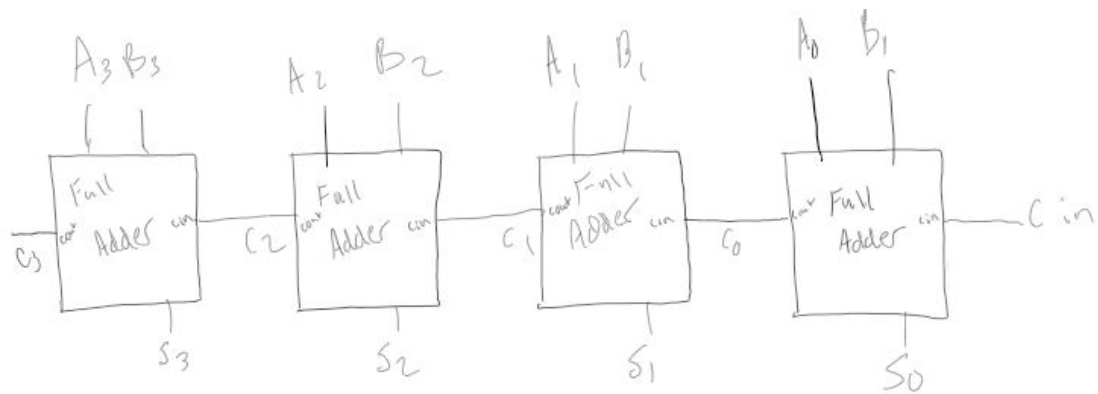
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = (A \oplus B) \oplus C_{in}$$

$$C_{out} = (A \oplus B)C_{in} + AB$$



[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.



```

P_TEST_CASES: process
begin
    wait for 100 ns; -- for waveform clarity,

--DO WE INPUT BINARY, DECIMAL???????
--CAN WE ASSUME INPUT WILL BE FILLED WITH 0S
    -----
    --HOW TO TEST
    -- Test case 1:
    -- Initialize weight value to 10.
    s_i_A    <= X"00000001";
    s_i_B    <= X"00000000";
    s_i_Cin  <= '0';
    --Expected sum to be 00000001 and Cout to b

    --Expected sum to be 00000001 and Cout to b
    wait for 100 ns;

    -----
    -- Test case 2:
    -- Perform average example of an input ac
    s_i_A    <= X"0000000A";
    s_i_B    <= X"0000000B";
    s_i_Cin  <= '0';
    --expectinf sum to be 00000015 and cout not

    -----
    -- Expect: s_W internal signal to be 10 a
    wait for 100 ns;
    -- Test case 2:
    -- Perform average example of an input ac
    s_i_A    <= X"11111111";
    s_i_B    <= X"00000000";
    s_i_Cin  <= '1';
    --sum will equal.....

    --Expected sum to be 00000001 and Cout to b
    wait for 100 ns;

    s_i_A    <= X"F0083940";
    s_i_B    <= X"F343ABC0";
    s_i_Cin  <= '0';

    -----

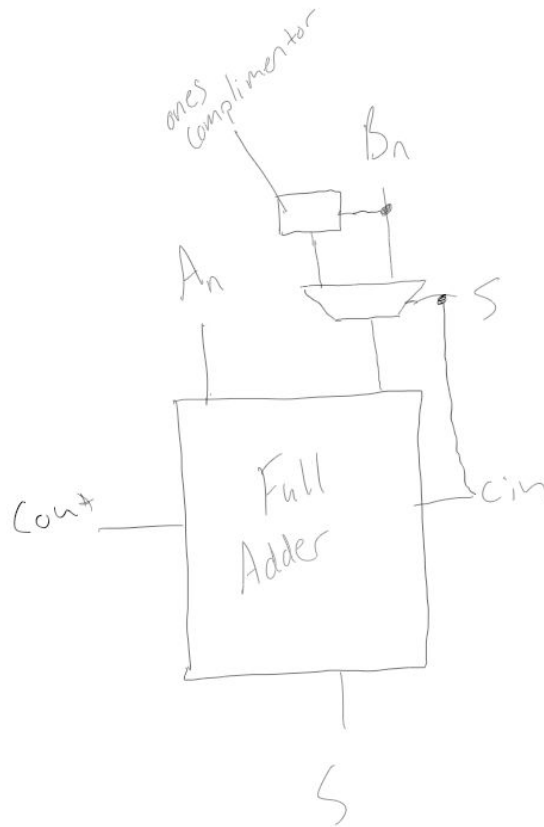
end process;
end mixed;

```

The waveform indicates three of our test cases being done correctly, the first one being a simple addition, the second one being A + B and the last one being a much more difficult input to compute by hand. Each test case is split up with a bit of space so you can see the individual test cases that were run.

*****RED AT THE BEGINNING WERE WAITS WE PUT AS SEEN IN THE TEST BENCH ODE SCREEN SHOT

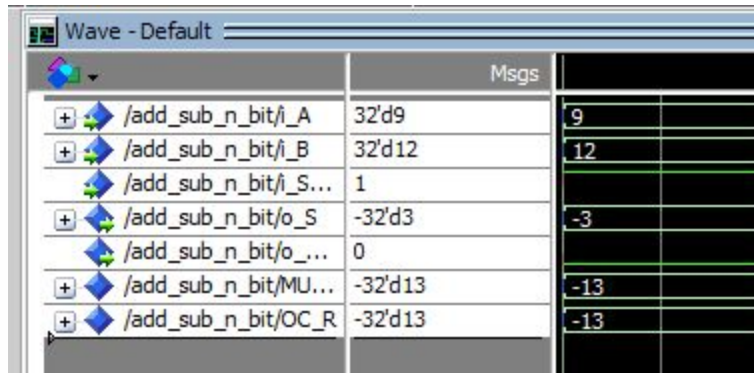
[Part 7.a] Draw a schematic (don't use a schematic capture tool) showing how an N-bit adder/subtractor with control can be implemented using only the three main components designed in earlier parts of this lab (i.e., the N-bit inverter, N-bit 2:1 mux, and N-bit adder). How is the 'nAdd_Sub' bit used?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



If the nAdd_Sub bit is referring to the signal that the mux takes to switch between the inputs, it is used as select bit to switch between addition and subtraction. This is done by selecting a normal input B_n or the inverted input produced by the 1's complement which will then subtract the two.

[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?

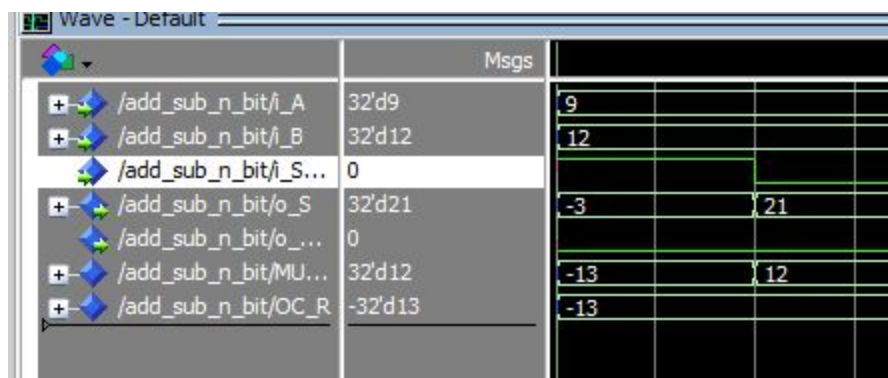
First Case: $9 - 12 = -3$ ($0x9 - 0xC$) We included this because it was an easy test for subtraction



Wave - Default

	Msgs	
/add_sub_n_bit/i_A	32'd9	9
/add_sub_n_bit/i_B	32'd12	12
/add_sub_n_bit/i_S...	1	
/add_sub_n_bit/o_S	-32'd3	-3
/add_sub_n_bit/o_...	0	
/add_sub_n_bit/MU...	-32'd13	-13
/add_sub_n_bit/OC_R	-32'd13	-13

Second Case: $9 + 12 = 21$ ($0x9 + 0xC$) We included this because it was an easy test for addition



Wave - Default

	Msgs	
/add_sub_n_bit/i_A	32'd9	9
/add_sub_n_bit/i_B	32'd12	12
/add_sub_n_bit/i_S...	0	
/add_sub_n_bit/o_S	32'd21	-3
/add_sub_n_bit/o_...	0	21
/add_sub_n_bit/MU...	32'd12	-13
/add_sub_n_bit/OC_R	-32'd13	12
		-13

Third Case: $0x1234 - 0xABCD = 0xFFFF6667$ We included this test because it was a difficult subtraction problem.



Wave - Default

	Msgs	
/add_sub_n_bit/i_A	32'h00001234	00000009
/add_sub_n_bit/i_B	32'h0000ABCD	0000000C
/add_sub_n_bit/i_S...	1	
/add_sub_n_bit/o_S	32'hFFFF6667	FFFFFFFFD
/add_sub_n_bit/o_...	0	00000015
/add_sub_n_bit/MU...	32'hFFFF5432	FFFFFFFF3
/add_sub_n_bit/OC_R	32'hFFFF5432	FFFFFFFF3

Fourth Case: $0x1234 + 0xABCD = 0xBE01$ We included this test because it was a difficult addition problem

Wave - Default		Msgs			
+ /add_sub_n_bit/i_A	32'h00001234	00000009		00001234	
+ /add_sub_n_bit/i_B	32'h0000ABCD	0000000C		0000ABCD	
+ /add_sub_n_bit/i_S...	0				
+ /add_sub_n_bit/o_S	32'h0000BE01	FFFFFFFD	00000015	FFFF6667	0000BE01
+ /add_sub_n_bit/o_...	0				
+ /add_sub_n_bit/MU...	32'h0000ABCD	FFFFFFF3	0000000C	FFFF5432	0000ABCD
+ /add_sub_n_bit/OC_R	32'hFFFF5432	FFFFFFF3		FFFF5432	