CprE 381, Computer Organization and Assembly Level Programming

Lab 1 Report

Student Name Shivansh Patel

***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.

• 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.
• I expect the intermediate internal signal, s_W, to take on my new value of 5 at the
positive edge of the clock. The value of s_W prior to the clock edge may be anything
since this is an initialization. I expect that s_W will not change unless I assert iLdW to
1 across a positive edge of the clock while iW is a different value. Case 2: Perform
one MAC operation with average-case values
• With the above weight initialization to 5, I set the activation input, iX, to 10 and the
partial sum input, iY, to 2. 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 3 and the partial sum output, oY will be 52=5*10+2

iW set to 2, iLdW set to 1 for one clock cycle.
s_W is set to 2 to store the value of iW.
IX is then set to 2 and iY is set to 8 and we now set iLdW to 0 to make sure that s_W does not change values
in the next clock cycle we store the value of iX in s_X1 and we multiply iX and s_W to get a value of 4 which we then add to the value of s_Y1 to get the output oY of 12 and oX of 2.

iW set to 4, iLdW set to 1 for one clock cycle.
s_W is set to 4 to store the value of iW.
IX is then set to 4 and iY is set to 2 and we now set iLdW to 0 to make sure that s_W does not change values
in the next clock cycle we store the value of iX in s_X1 and we multiply iX and s_W to get a value of 16 which we then add to the value of s_Y1 to get the output oY of 18 and oX of 4.

[Part 1.e] For labels 1, 7, 22, and 28, 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

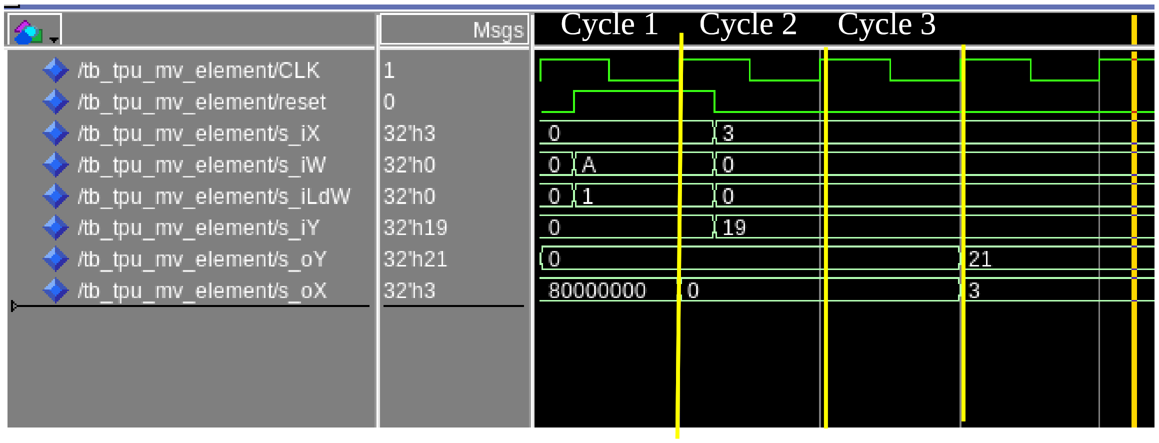
(1) is the entity name of
the multiplier module, which is defined as an entity on Line 23 of TPU_MV_Element.vhd and defined on Line 35. This is used to declare the entity's name and establish the component.

(7) is adder module used in the MAC system, which is defined as an entity on Line 26 of Adder.vhd and instanced on line 117 of TPU_MV_Element.vhd. For this specific instance of the adder module, the output oY is connected to a signal called oC on line 121 of TPU_MV_Element.vhd.

(22) is the oQ output of weight module which is defined as an output integer on line 31 of RegLD.vhd and called on line 82 of TPU_MV_Element.vhd. For this specific instance of the oQ output, the output stores the value of iW to prevent it from being changed during the process.

(28) is the iD input of
the delay3 module, which is defined as an in integer on line 29 of Reg.vhd and called on line 114 of TPU_MV_Element.vhd. For this specific instance of the iD input, the input holds the value o s_X1 through a clock cycle to output oX.

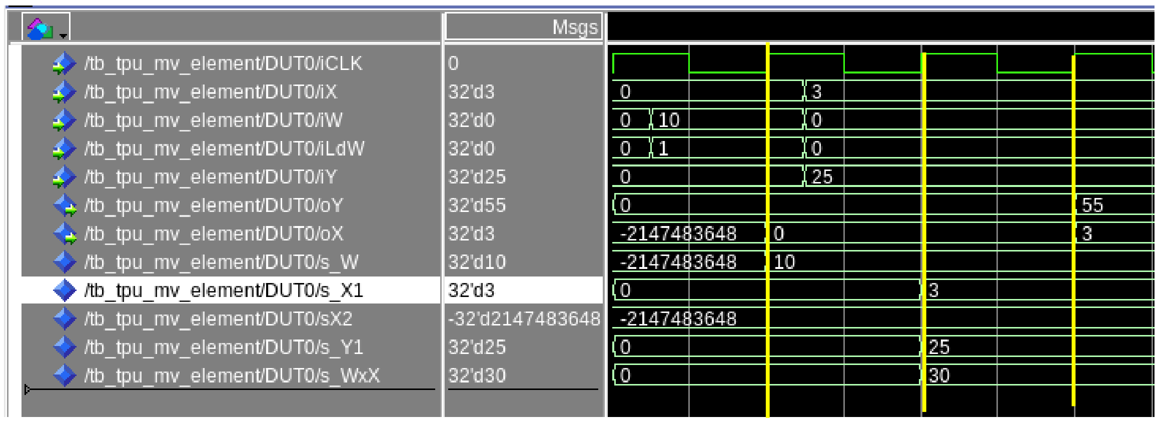
[Part 1.g.v] In your lab report, include a screenshot of the waveform. Describe, in plain English, any differences between what you expected and what the simulation showed.



	Msgs
/tb_tpu_mv_element/CLK	1
/tb_tpu_mv_element/reset	0
/tb_tpu_mv_element/s_iX	32'h3
/tb_tpu_mv_element/s_iW	32'h0
/tb_tpu_mv_element/s_iLdW	32'h0
/tb_tpu_mv_element/s_iY	32'h19
/tb_tpu_mv_element/s_oY	32'h21
/tb_tpu_mv_element/s_oX	32'h3

One discrepancy in what I expected versus what actually happened was the difference between the oY output and the expected result. Using iX, iW, and iY values of 3 A and 19, you would expect a value of 49: 3*10 + 19. However, our output y results in a value of 21.

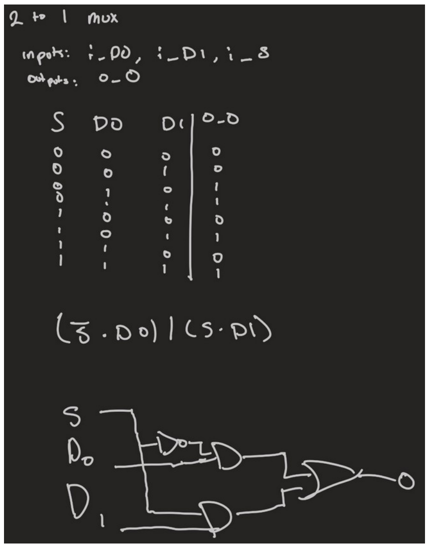
[Part 1.h] In your lab report, include a screenshot of the waveform. Describe, in plain English, how your waveform matches the expected result (e.g., reference the specific cycles and times). In your submission zip file, provide the completed *TPU_MV_Element.vhd* file in a folder called ‘MAC’.



	Msgs
/tb_tpu_mv_element/DUT0/iCLK	0
/tb_tpu_mv_element/DUT0/iX	32'd3
/tb_tpu_mv_element/DUT0/iW	32'd0
/tb_tpu_mv_element/DUT0/iLdW	32'd0
/tb_tpu_mv_element/DUT0/iY	32'd25
/tb_tpu_mv_element/DUT0/oY	32'd55
/tb_tpu_mv_element/DUT0/oX	32'd3
/tb_tpu_mv_element/DUT0/s_W	32'd10
/tb_tpu_mv_element/DUT0/s_X1	32'd3
/tb_tpu_mv_element/DUT0/sX2	-32'd2147483648
/tb_tpu_mv_element/DUT0/s_Y1	32'd25
/tb_tpu_mv_element/DUT0/s_WxX	32'd30

In this solution, I fixed the discrepancy through checking the delayed values of iX through iY. After fixing this delay, I got the expected value of 55 and 3 for oY and oX.

[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.



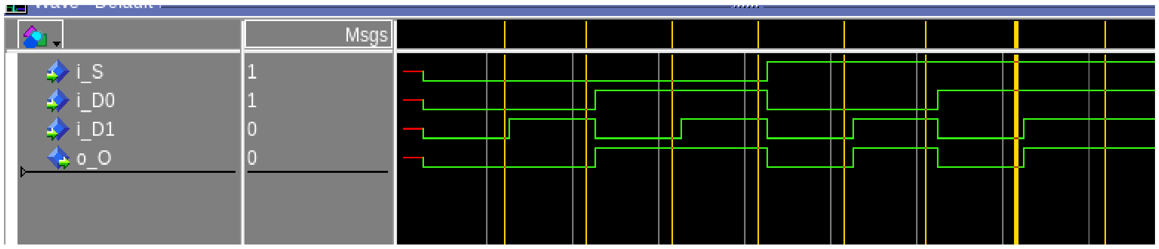
2 to 1 mux

inputs: i_D0, i_D1, i_S
outputs: o_O

S	D0	D1	o_O
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$(\overline{S} \cdot D0) \mid (S \cdot D1)$

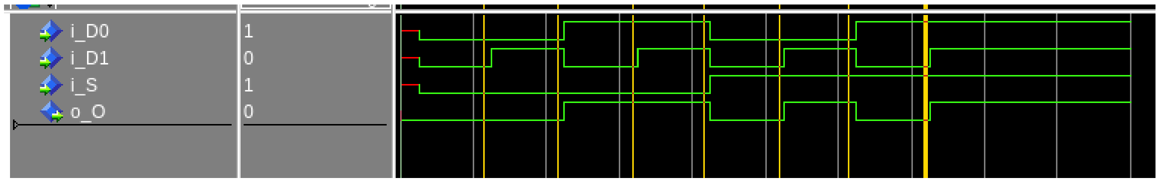
[Part 3.d] In your lab report, include a screenshot of the waveform. Make sure to label the screenshot with which module it is testing.



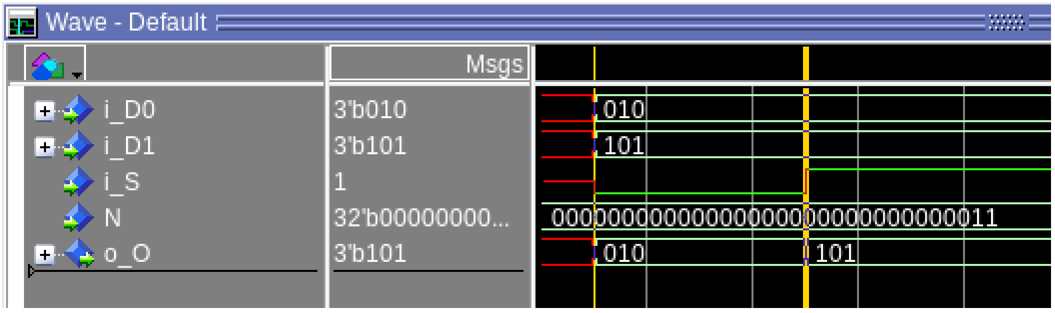
[Part 3.e] Again, in your lab report, include a labeled screenshot of the waveform showing the dataflow mux implementation working.

2 to 1 mux structural

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



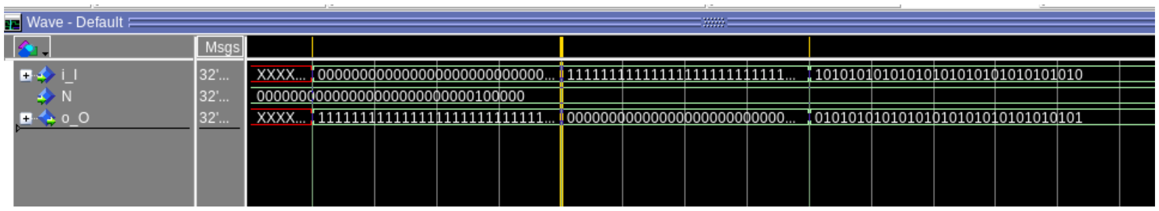
2 to 1 mux dataflow



	Msgs	
i_D0	3'b010	010
i_D1	3'b101	101
i_S	1	
N	32'b00000000...	00000000000000000000000000000011
o_O	3'b101	010 101

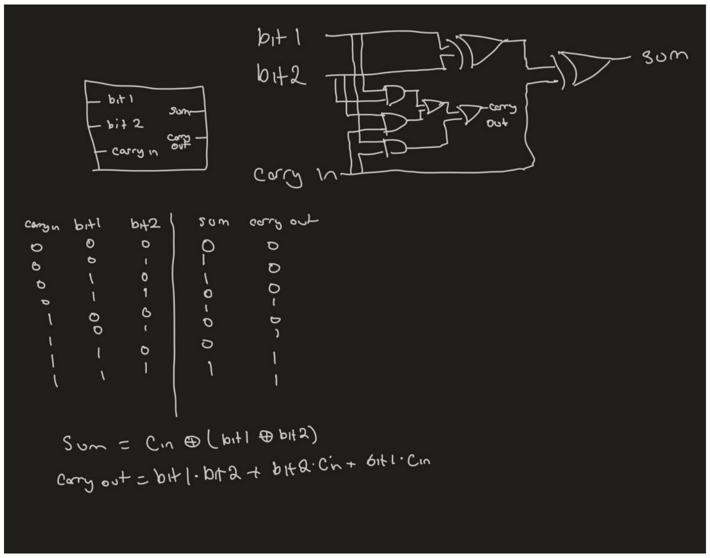
This is a 3 bit mux, we see it taking 010 for D0 and 101 for D1 and when i_S is 0, the output equals D0 and while i_S is 1, the output equals D1

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



	Msgs				
i_I	32'...	XXXX...	0000000000000000000000000000...	1111111111111111111111111111...	10101010101010101010101010101010
N	32'...	00000000000000000000000000100000			
o_O	32'...	XXXX...	1111111111111111111111111111...	0000000000000000000000000000...	01010101010101010101010101010101

[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.

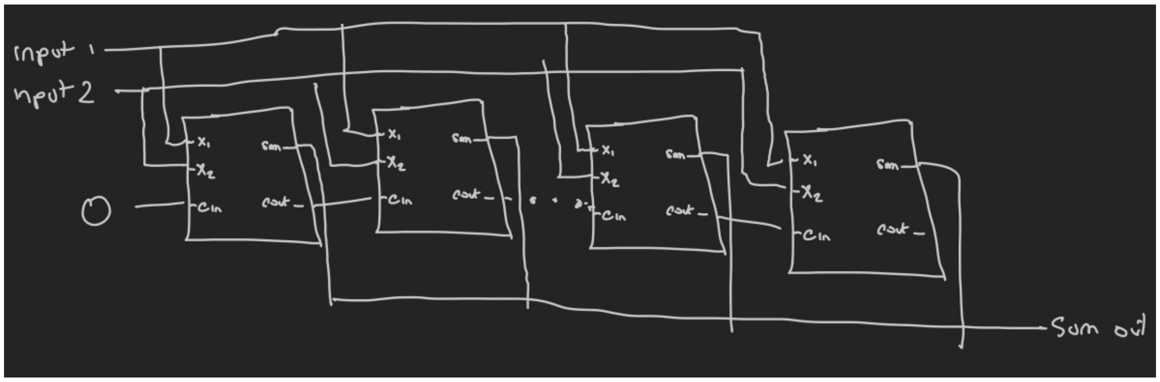


carry in	bit1	bit2	sum	carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

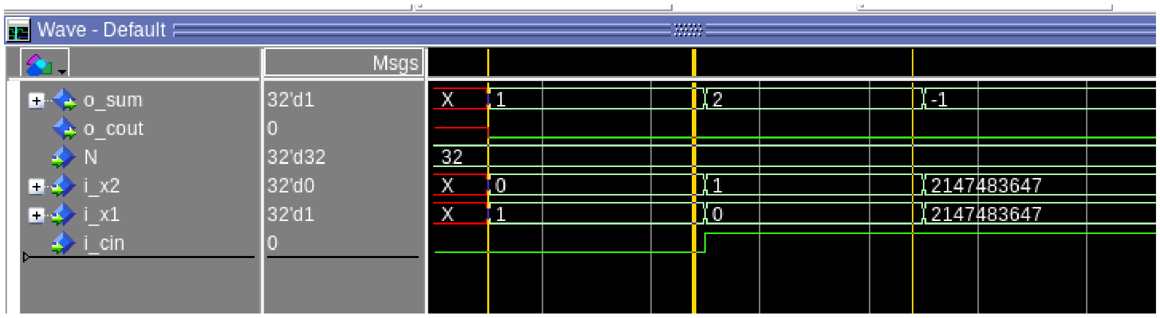
$$\text{Sum} = C_{in} \oplus (\text{bit1} \oplus \text{bit2})$$

$$\text{Carry out} = \text{bit1} \cdot \text{bit2} + \text{bit2} \cdot C_{in} + \text{bit1} \cdot C_{in}$$

[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.



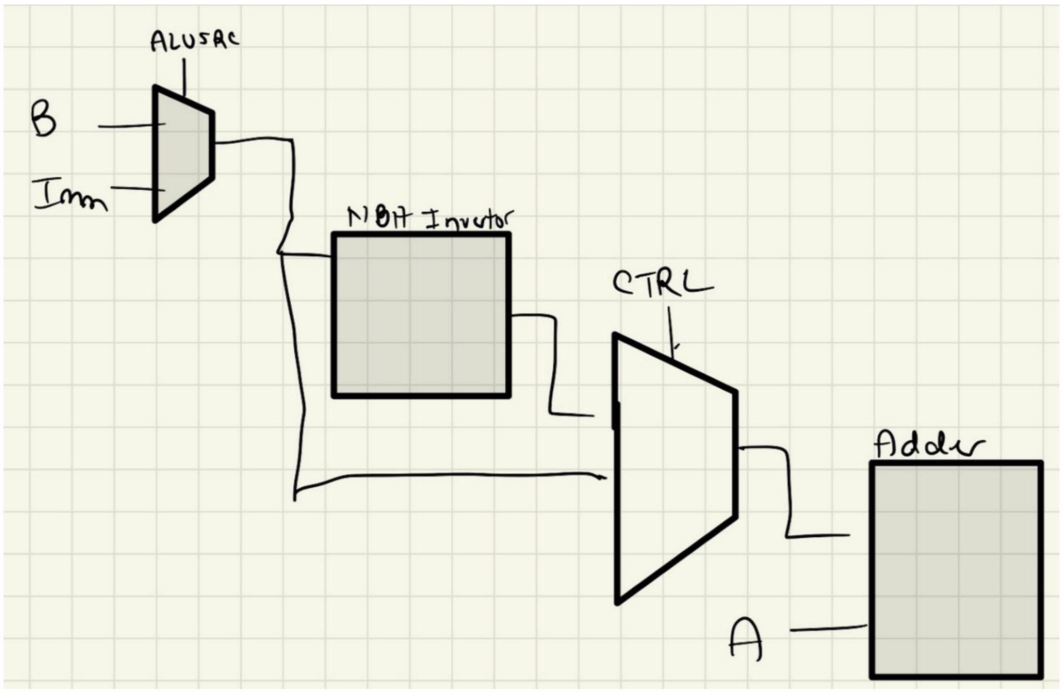
[Part 6.d] Include an annotated waveform screenshot in your write-up.



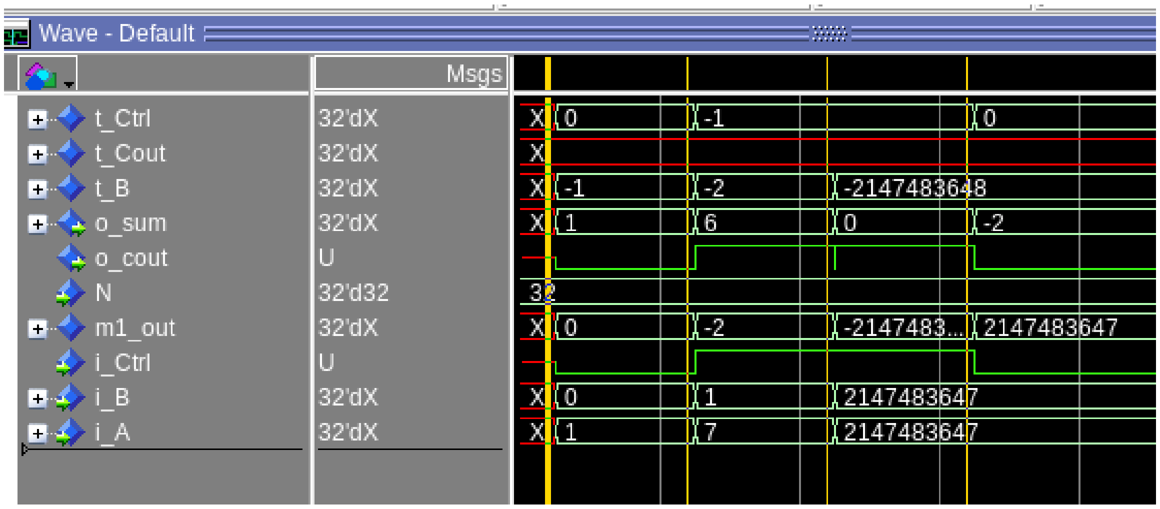
```
o_sum    32'd1    X  1            2            -1
o_cout   0
N        32'd32   32
i_x2     32'd0    X  0            1            2147483647
i_x1     32'd1    X  1            0            2147483647
i_cin    0
```

In the first cursor section we see that our inputs of 0 and 1 with no carry gives an out put of 1. In the second cursor, an input value of 0 and 1 with a carry in value of 1 will result in us having a 2 for the output. In the third cursor segment, we have a binary value of 0111… + 0111… and a carry in resulting in the maximum sum possible of 11111… which is signed to -1.

[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.



[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
t_Ctrl 32'dX X 0 -1 0
t_Cout 32'dX X
t_B 32'dX X -1 -2 -2147483648
o_sum 32'dX X 1 6 0 -2
o_cout U
N 32'd32 32
m1_out 32'dX X 0 -2 -2147483... 2147483647
i_Ctrl U
i_B 32'dX X 0 1 2147483647
i_A 32'dX X 1 7 2147483647

In this test case we see both the adder and subtractor functionality. The first cursor segment shows the ctrl is 0, b is 0, and a is 1. Therefore we should be doing the add operation of 0+1 and we get our o_sum value of 1. The next cursor chagnes this to ctrl value of 1 which means we subtract A-B 7-1 and get 6 in the output value. The third has us subtracting the two largest numbers, 01111… - 011111… and we get an output of 0. Then finally we add the two largest numbers with no carry in to get a signed value of -2.