8-bit ALU:

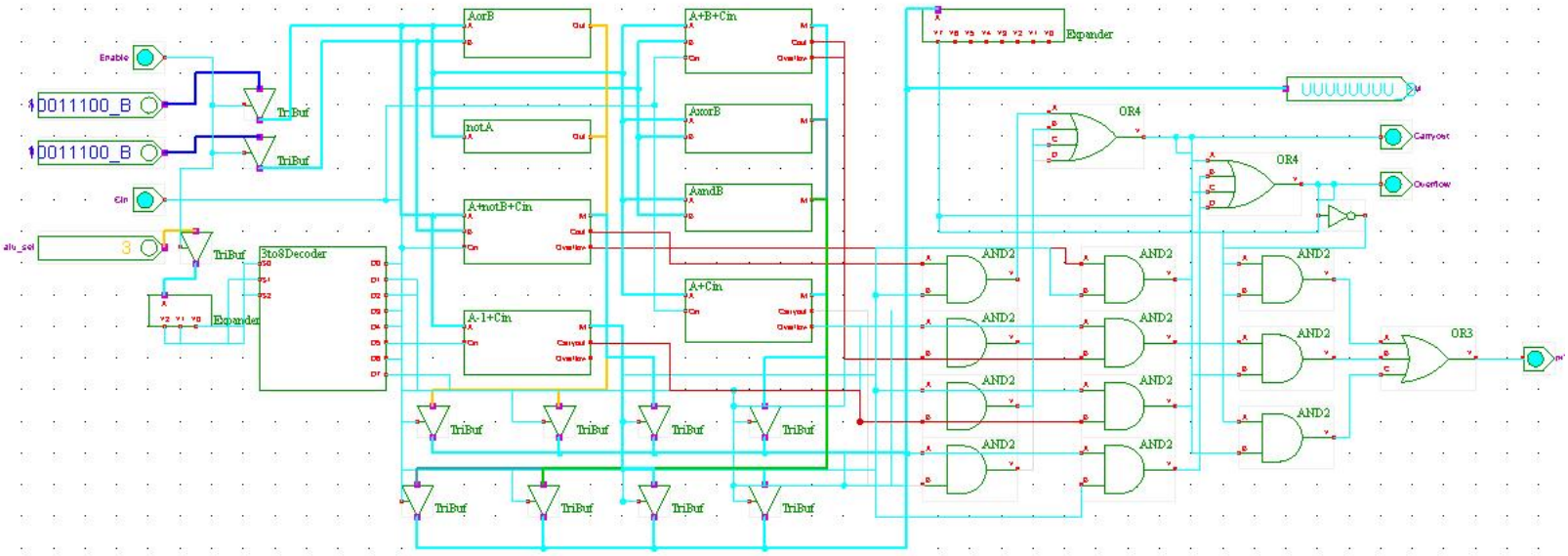
This 8-bit ALU takes two 8-bit inputs A and B, and performs an operation on them that is selected by the 3-bit selector ALU_SEL. The selection is done using a 3to8 decoder and 8 tri-state buffers. The output of the decoder enables the tri-state and allows the result of the selected to pass to M. The result of the operation is placed in the 8-bit output M. There is a single bit Cin that is used in function 2,3,6, and 7. There is an enable that must be set to 1 for the circuit to run.

In addition to the M output, there is also 3 single bit outputs Carryout, Overflow, and m7. Carryout is the carryout of the 4 functions that produce a carryout. The overflow tells us if we use the carryout or not. If overflow is 1 then carryout is the most significant bit. If overflow is 0 then the most significant bit is the 2^7 bit of M.

m7 is the sign bit of the resulting function. The sign bit is the most significant bit and tells us if the value is positive or negative in 2's compliment. 0 indicates a positive and 1 indicates a negative. If overflow is 1 then m7 is equal to the carryout. If overflow is 0 then m7 is equal to the most significant bit of M. After making a truth table and k-map, the function for m7 is Not(Overflow) *M7+Overflow*Cout.

The following functions are performed based on ALU_SEL:

ALU_SEL	Function
000	A or B
001	Not A
010	A+Not(B)+Cin
011	A+B+Cin
100	A xor B
101	A and B
110	A-1+Cin
111	A+Cin

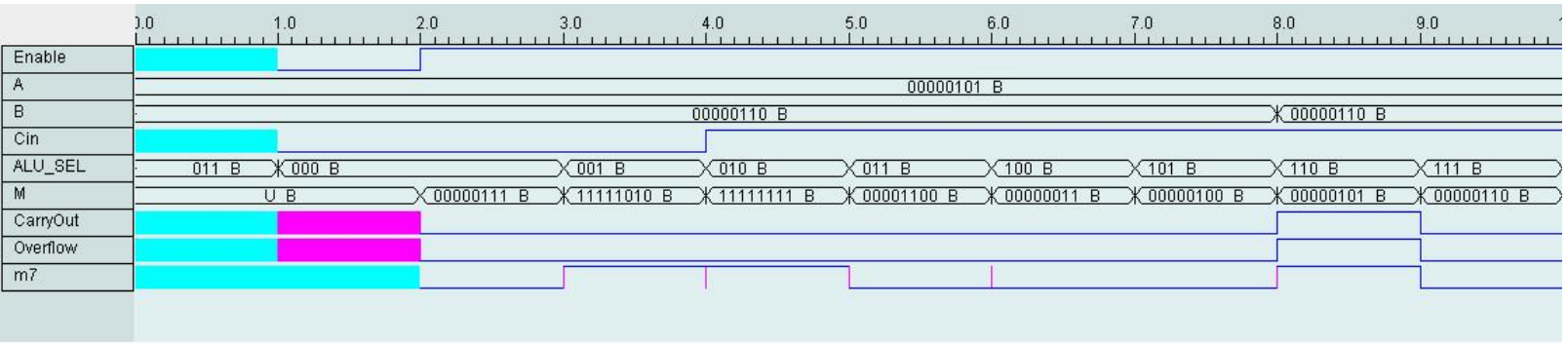


Testing:

For testing I did 2 sets of tests. For the first set I checked to make sure that each of the functions produced the proper result into M. When enable is 0, nothing should pass to M. When enable is 1, the function selected by ALU_SEL should pass to M. In total there were 9 tests for this set, 1 for the

enable and 1 for each function. The waveform matches the truth table.

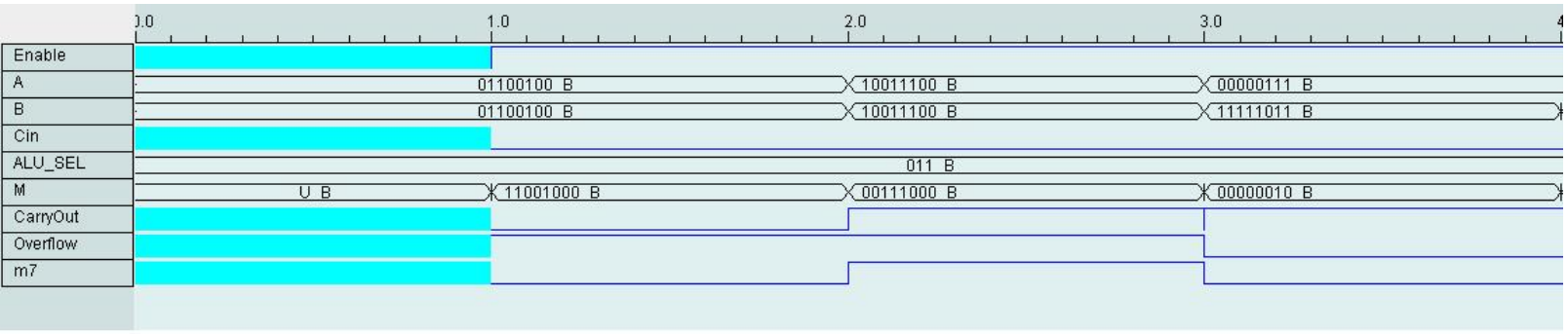
A	B	Cin	ALU_SEL	M	enable
x	x	x	x	UUU	0
00000101	00000110	0	000	00000111	1
00000101	00000110	0	001	11111010	1
00000101	00000110	1	010	11111111	1
00000101	00000110	1	011	00001100	1
00000101	00000110	1	100	00000011	1
00000101	00000110	1	101	00000100	1
00000101	00000110	1	110	00000101	1
00000101	00000110	1	111	00000110	1



For the second set of tests I tested to make sure that the Carryout, Overflow, and m7 outputs were working properly. I used function 3 A+B+Cin to test with various inputs of A and B. The waveform matches the truth table.

A	B	Cout	Overflow	m7
100	100	0	1	0
-100	-100	1	1	1

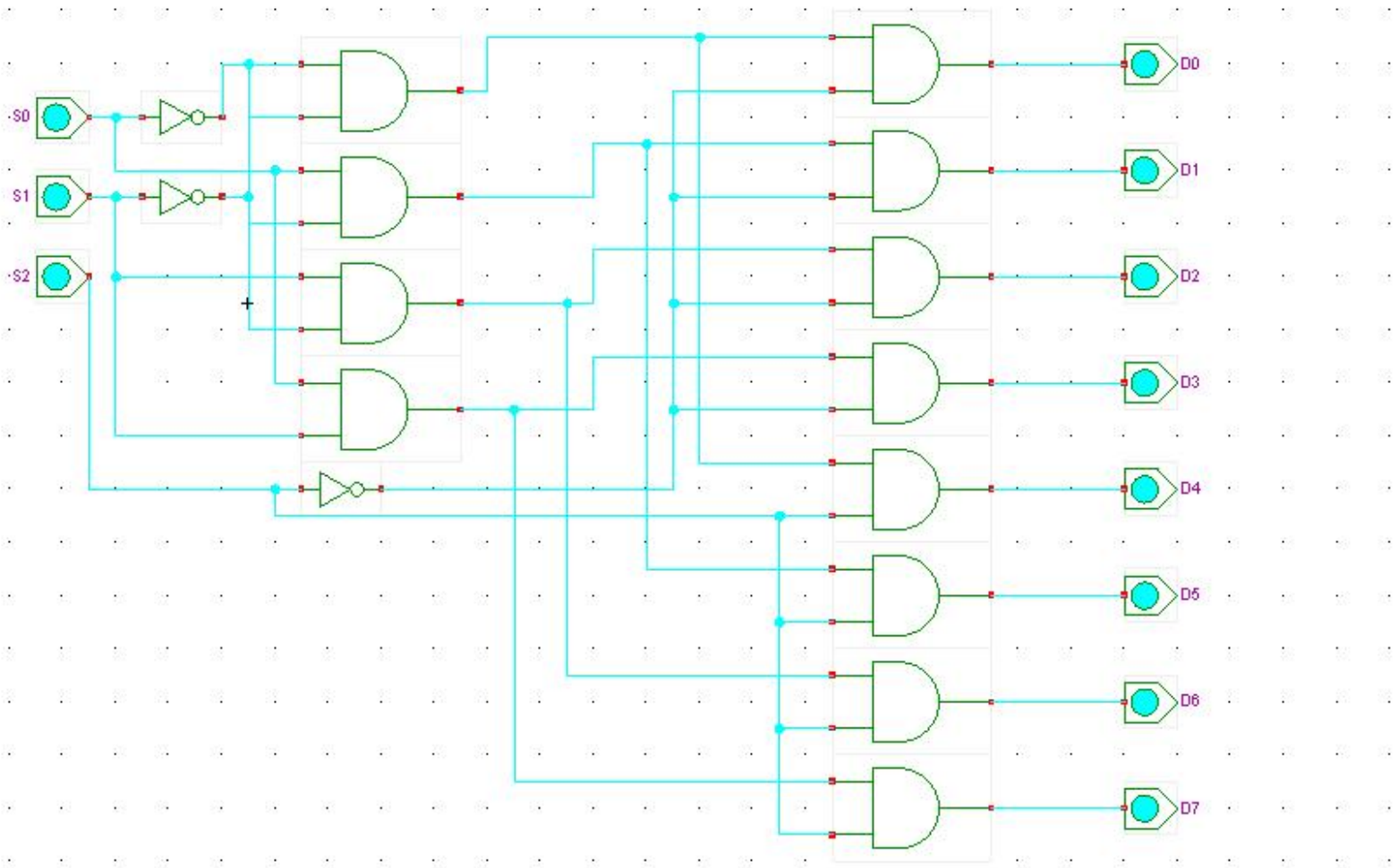
7 -5 1 0 0



3 to 8 Decoder:

The 3 to 8 Decoder takes 3 single bit inputs and selects one of 8 outputs based on the input combinations. With 3 inputs there will be 2^3 possible input combinations. This circuit is used in the ALU to decode the ALU_SEL. Each input of ALU_SEL will activate one output. That output is

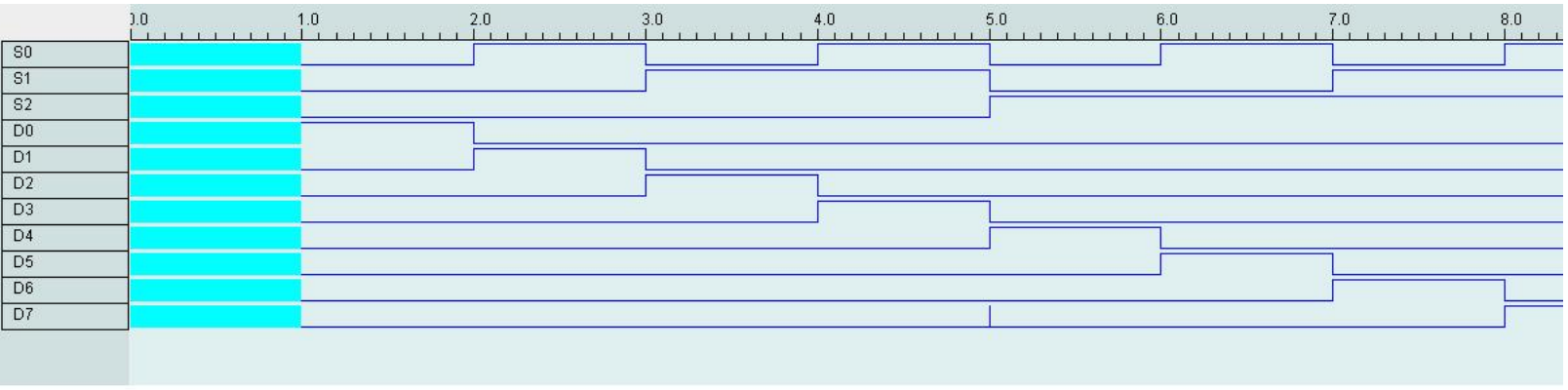
connected to the enable of a tri-state buffer. In this way the 3 to 8 decoder is used to select the functions.



Testing:

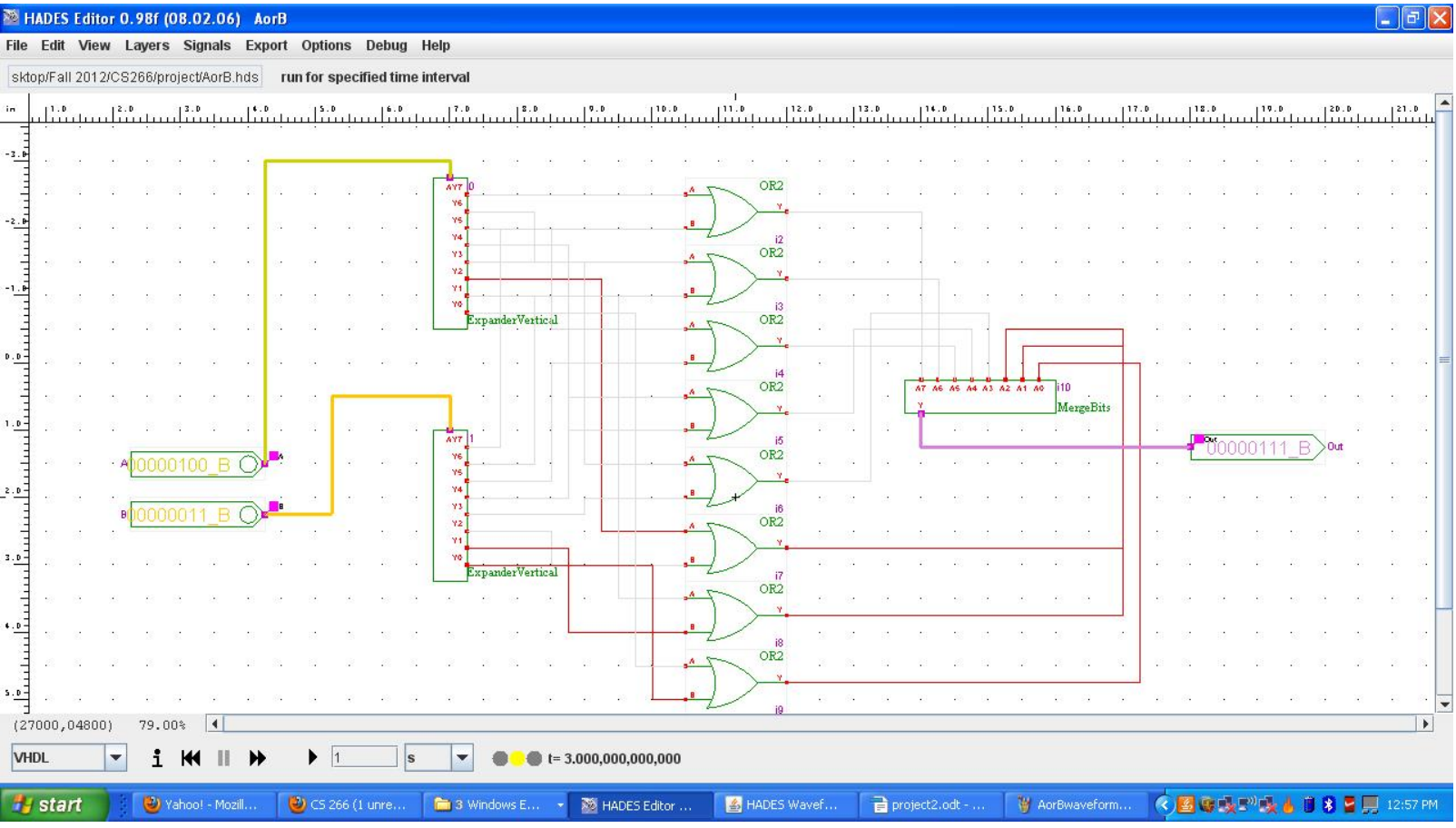
This is a simple circuit so I tested every possible input combination. The waveform matches the truth table and therefore the circuit works.

S2	S1	S0	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



A or B:

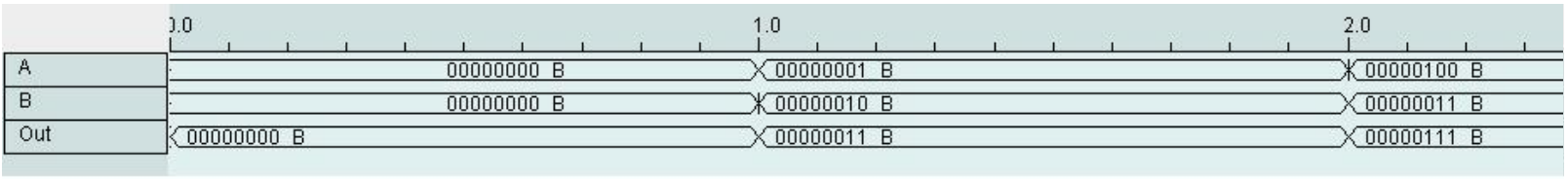
Description: The A OR B function takes two 8-bit binary numbers as inputs, and outputs the logical disjunction of the two inputs which displays a 1 whenever the corresponding bits of A or B are 1. The design uses two 8 bit busses that connect to designated expanders. These expanders have 8 outputs where each output connects to a 2-input OR-gate. The resulting OR-gates then connect to a 8-bit Merger which connects to the final 8-bit output resulting in an A OR B function.



Testing:

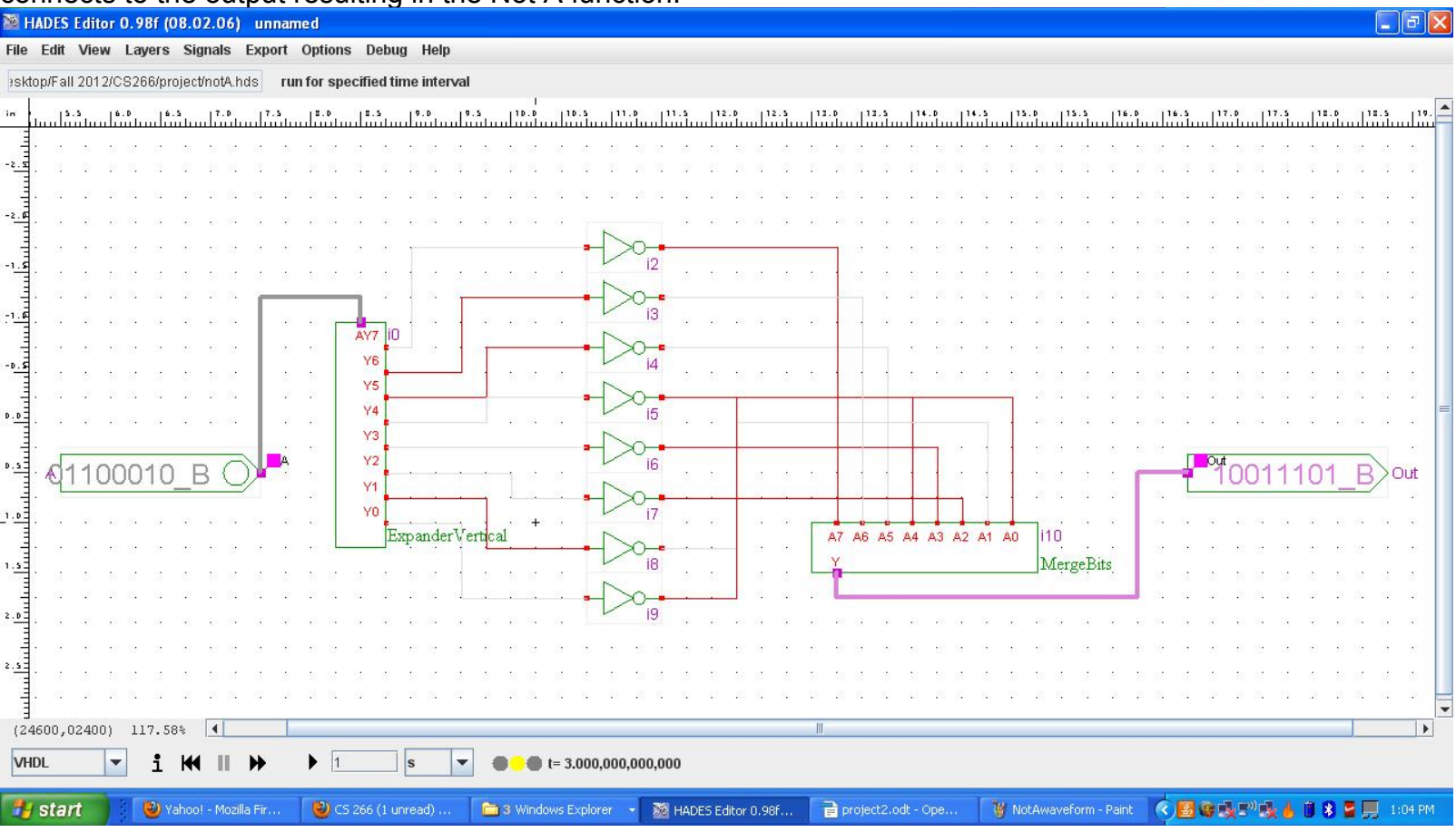
There were 3 tests. The waveform matches the truth table and shows the test work and therefore the circuit works.

A	B	Out
00000000	00000000	00000000
00000001	00000010	00000011
00000100	00000011	00000111



Not A:

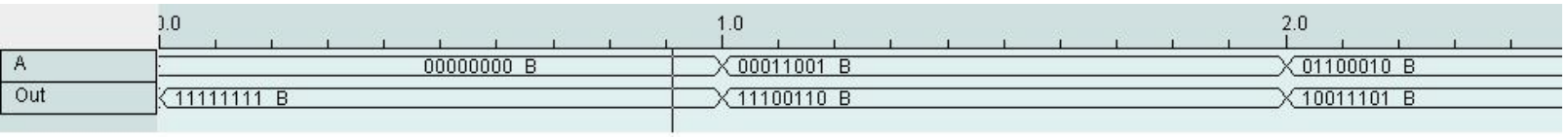
Description: The Not A function takes an 8-bit binary number as an input, and outputs the implemented 8-bit binary number which take the inverse of each bit from the input. The circuit design uses one 8-bit bus which is connected to an 8-bit expander. The expander has 8 outputs. Each of the outputs connect to 8 inverters. The outputs of the inverters then connect to an 8-bit Merger that finally connects to the output resulting in the Not A function.



Testing:

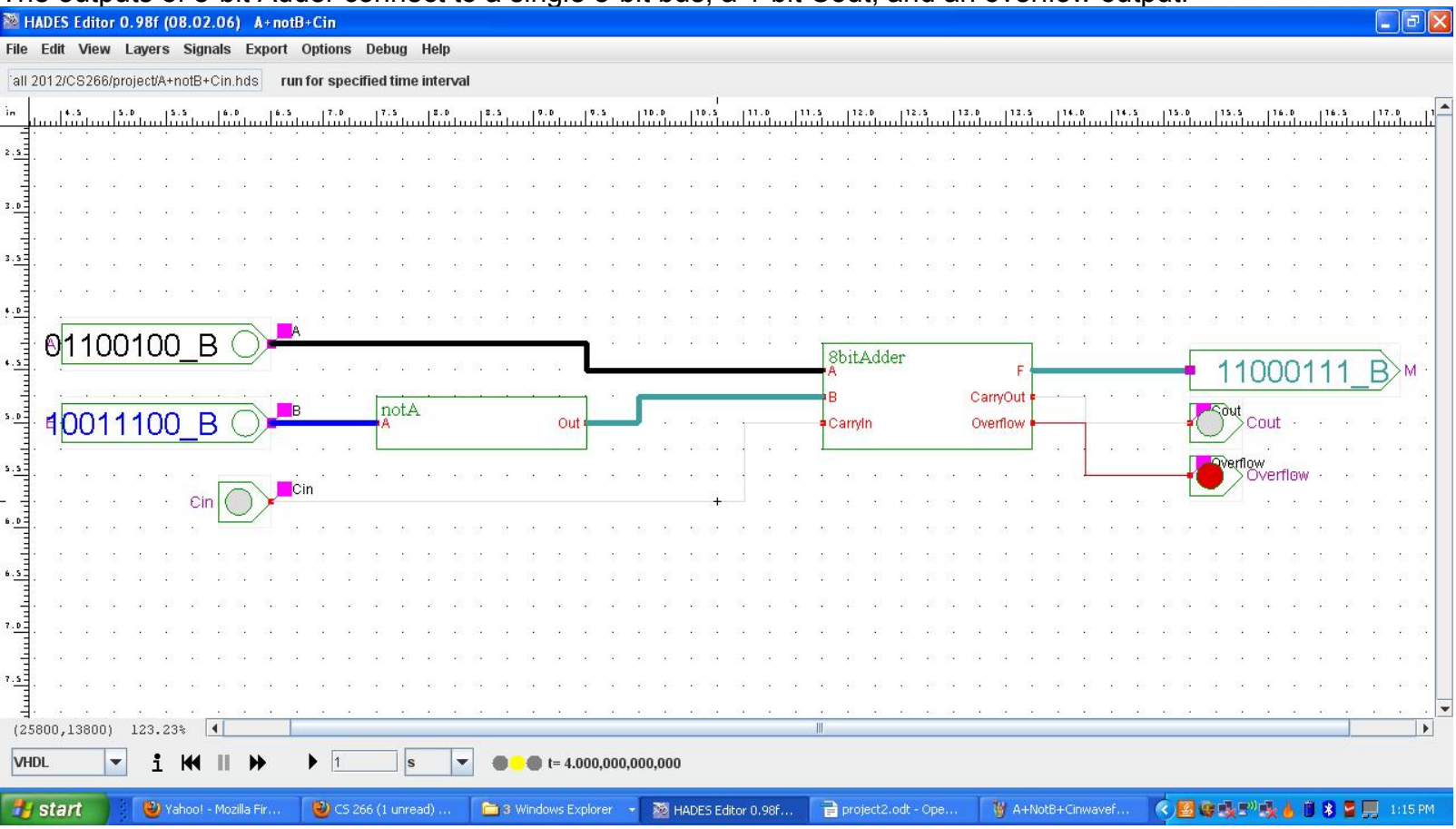
There were 3 test cases. The waveform matches the truth table and therefore the circuit works.

A	Out
00000000	11111111
00011001	11100110
01100010	10011101



A + Not(B)+Cin:

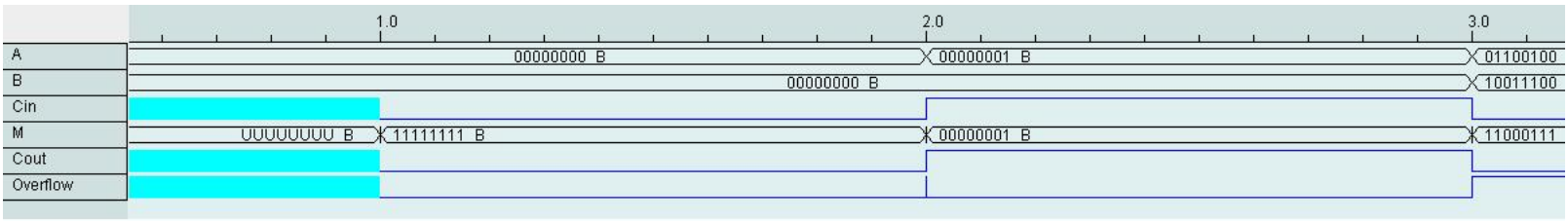
Description: The A+Not(B)+Cin function takes in two 8 bit values A and B as inputs, and a single 1-bit input Cin. The B input is sent through a not function which implements the inverse of B as the output. The output from the not function is connected as one of the inputs of an 8 bit Adder. The A and Cin are also connected to the 8bit Adder which adds the values of A and Not B taking into account Cin. The outputs of 8-bit Adder connect to a single 8-bit bus, a 1-bit Cout, and an overflow output.



Testing:

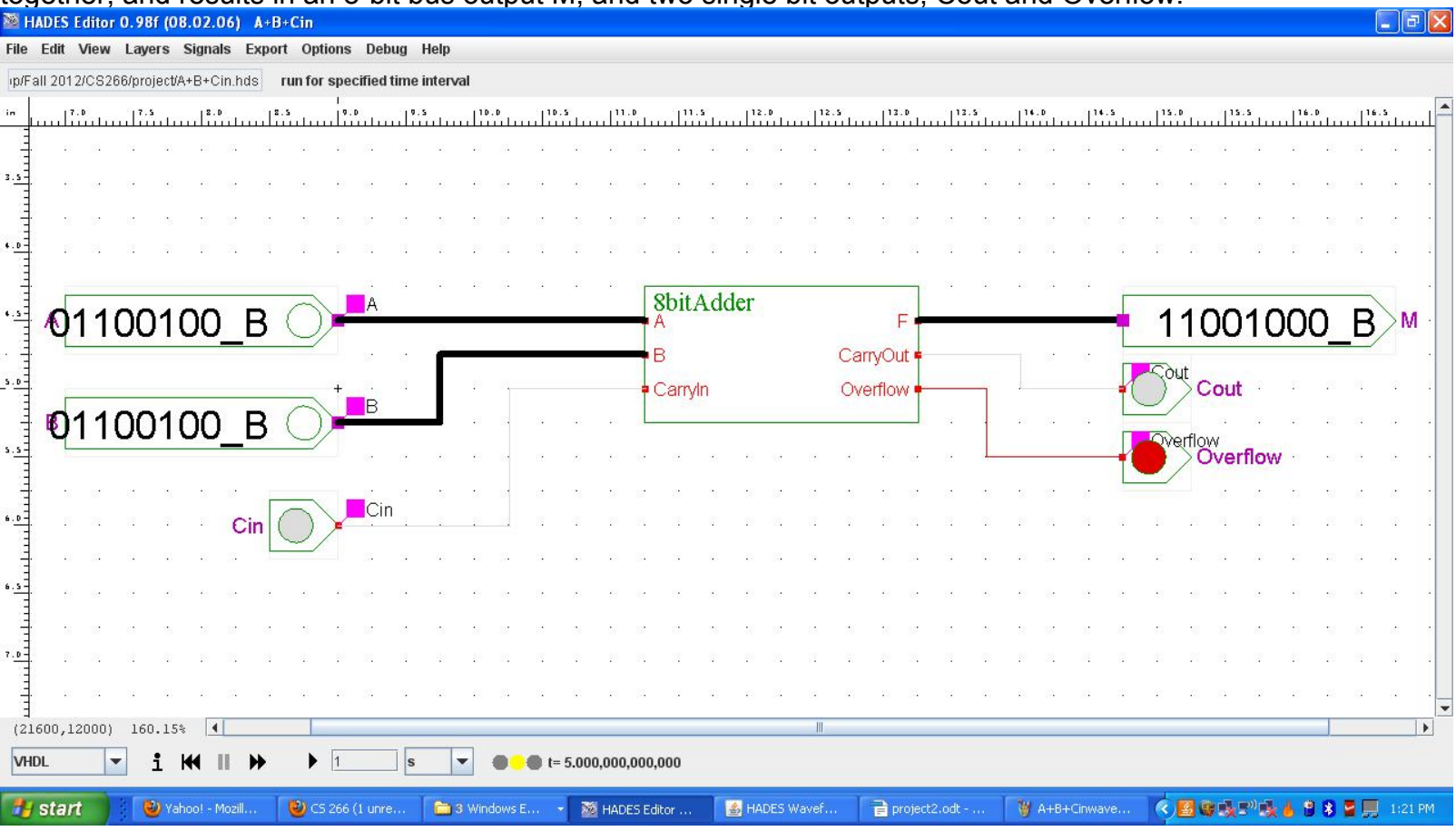
There were 3 tests. The waveform matches the truth table and shows the test work and therefore the circuit works.

A	B	Cin	M	Cout	Overflow
00000000	00000000	0	11111111	0	0
00000001	00000000	1	00000001	1	0
01100100	10011100	0	11000111	0	1


1.0
2.0
3.0
A
B
Cin
M
Cout
Overflow
00000000 B
00000001 B
01100100
00000000 B
10011100
UUUUUUUU B
11111111 B
00000001 B
11000111

A + B + Cin:

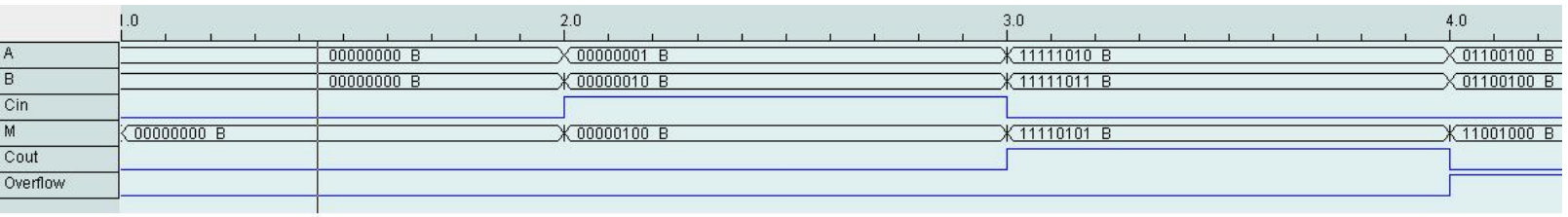
Description: The A + B + Cin function adds two 8-bit values together taking into account a 1-bit carry in. The function uses two 8-bit busses A and B as inputs which connect to an 8 bit Adder. The 1-bit Cin also connects to the 8bit Adder. The function of the 8bit Adder adds the values of its inputs together, and results in an 8-bit bus output M, and two single bit outputs, Cout and Overflow.



Testing:

There were 4 tests. The waveform matches the truth table and shows the test work and therefore the circuit works.

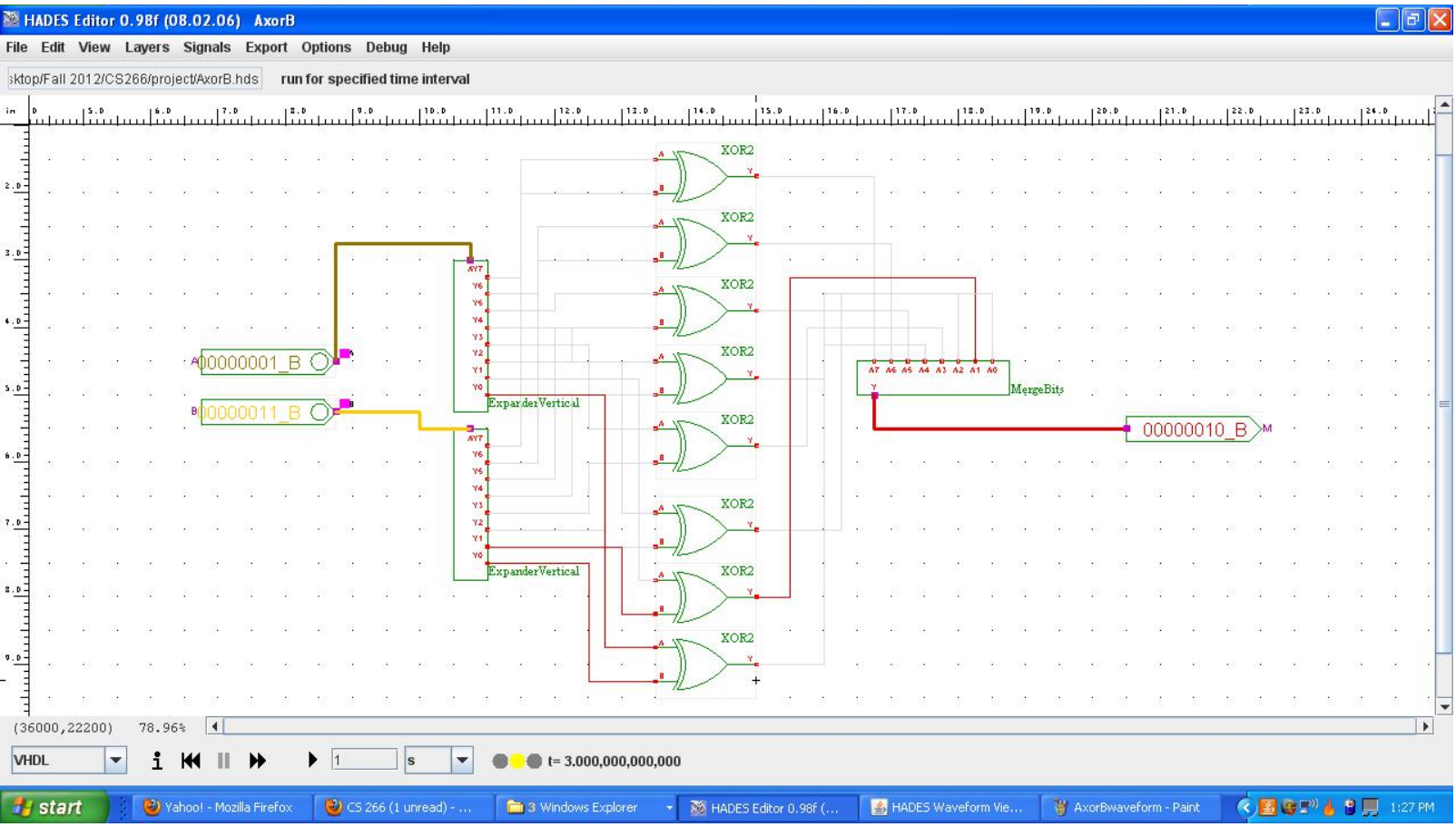
A	B	Cin	M	Cout	Overflow
00000000	00000000	0	00000000	0	0
00000001	00000010	1	00000100	0	0
11111010	11111011	0	11110101	1	0
01100100	01100100	0	11001000	0	1



A xor B:

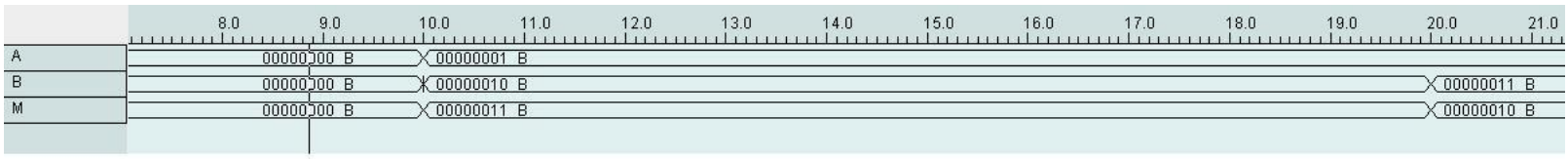
Description: The A xor B function shown below takes two 8-bit binary numbers as inputs and sends each bit through an xor gate which results in the corresponding 8-bit binary number. Each bit results in a true output (1) if one, and only if one or the other, of the inputs to the gate is true. The circuit design uses two 8-bit busses that each connect to an 8-bit expander. The expanders then connect

to 8 XOR gates which connect to the an 8-bit Merger. The Merger connects to an 8-bit output bus that displays the result of the A xor B function. The waveform shows a test case that verifies that the circuit functions correctly. When input A is 00000001 and input B is 00000010, then the output of the A xor B function is 00000011. This is the A xor B function.



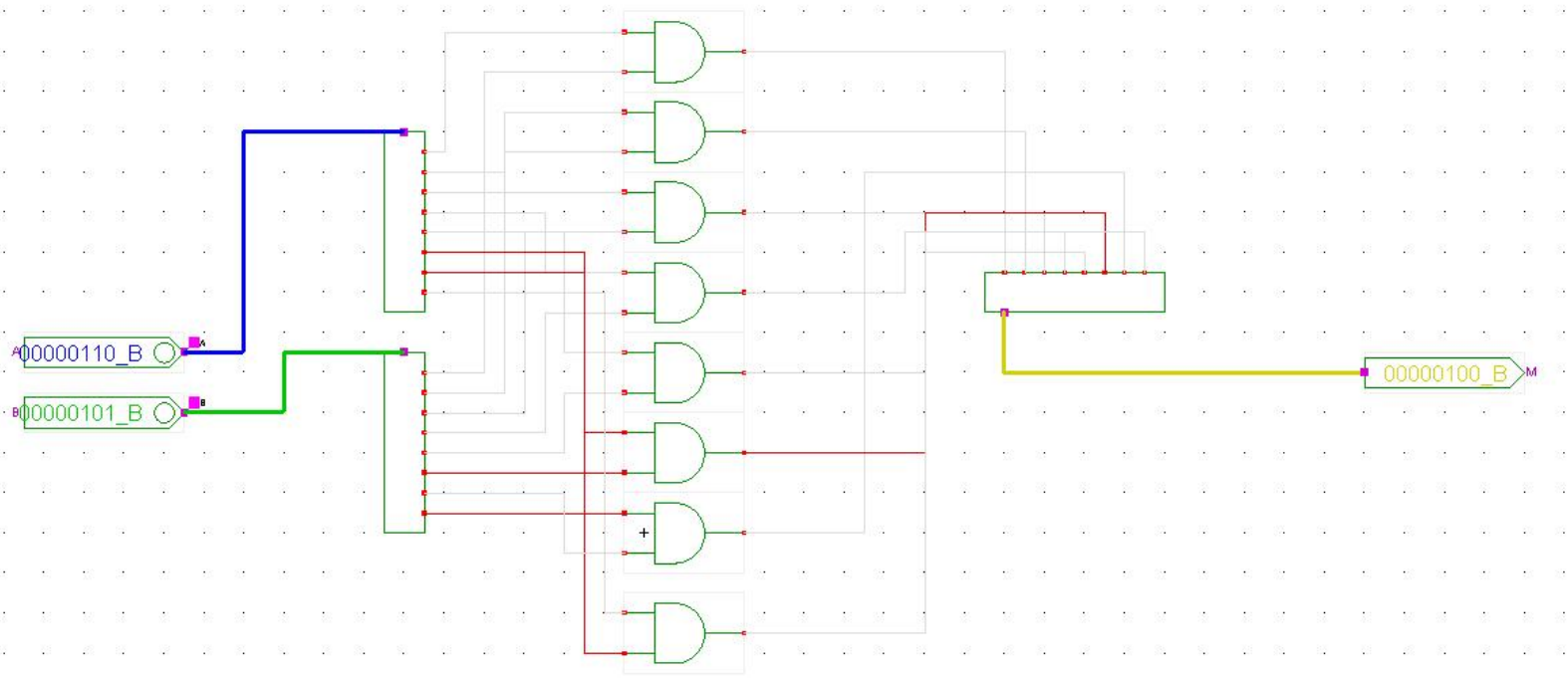
Testing:

A	B	M
00000000	00000000	00000000
00000001	00000010	00000011
00000001	00000011	00000010



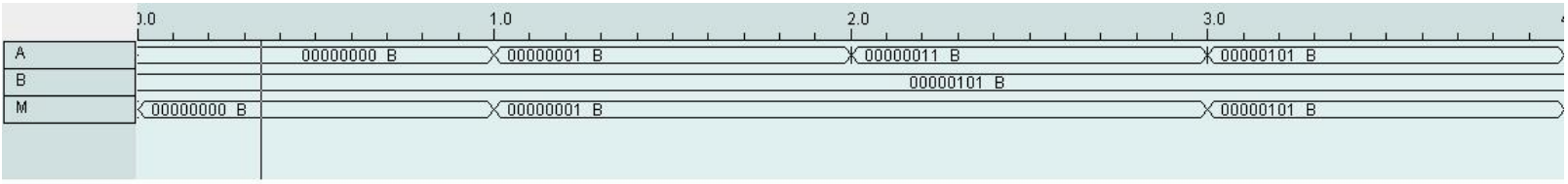
A and B:

Description: This circuit is a basic A and B function that takes two 8-bit binary numbers, and implements a logical conjunction between the two. This means that an output 1 results only if both the inputs to the AND gate are 1. The design uses two 8-bit busses that each connect to its corresponding 8-bit expander. The outputs of Expanders of A and B each connect to 8 AND gates. The output of each AND gate then connects to the 8-bit Merger which connects to the final 8-bit binary bus output of the A and B function. The waveform makes this circuit valid by showing that when input A is 00000011 and input B is 00000101 the resulting output is 00000001 because both the of least significant bits of A and B are the same. This is the A and B function.



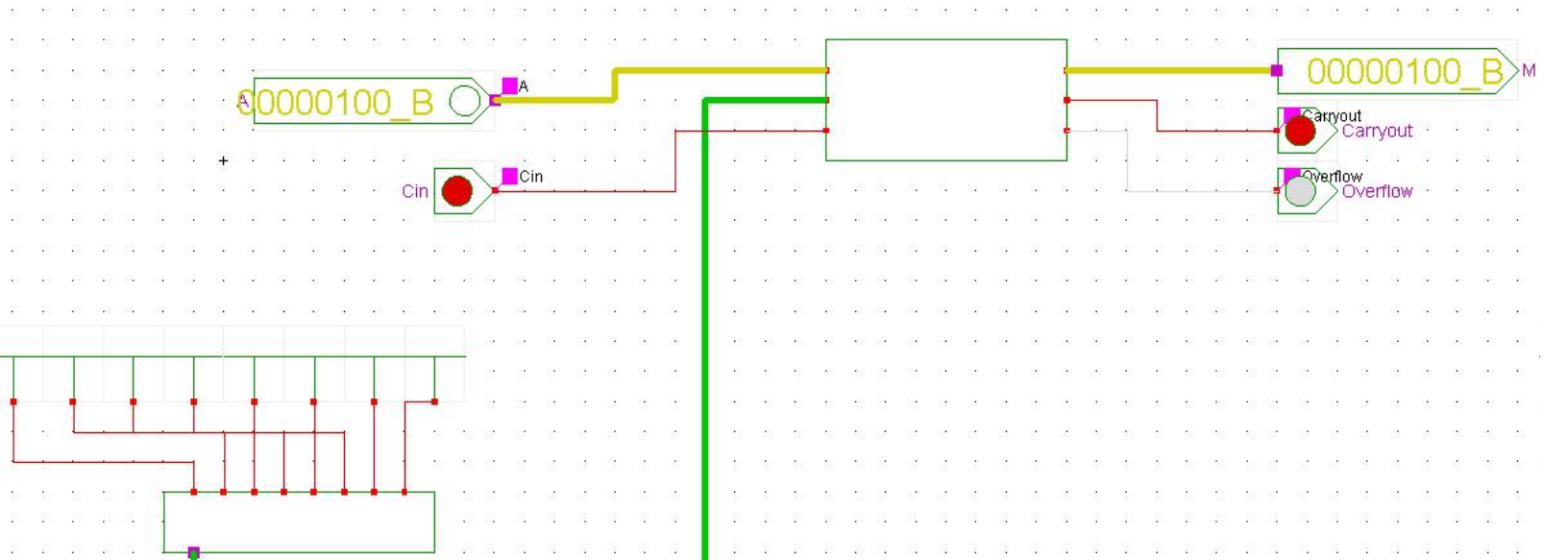
Testing:

A	B	M
00000000	00000101	00000000
00000001	00000101	00000001
00000011	00000101	00000001
00000101	00000101	00000101



A - 1 + Cin:

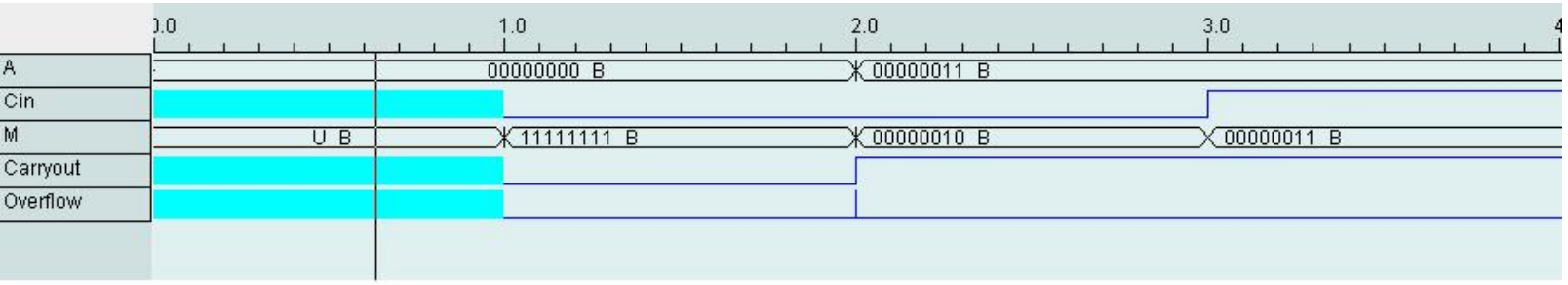
Description: This circuit takes an 8 bit input A and subtracts 1 from it and then adds Cin. Basically if Cin is 0 it does A-1. if Cin is 1 the result is just A. This circuit is build using an 8 bit adder with one of the inputs fixed to -1. The ouput is the 8 bit result M. While this circuit uses the 8 bit adder, it will never overflow.


00000100_B
A
Cin
Cin
00000100_B
M
Carryout
Carryout
Overflow
Overflow

Testing:

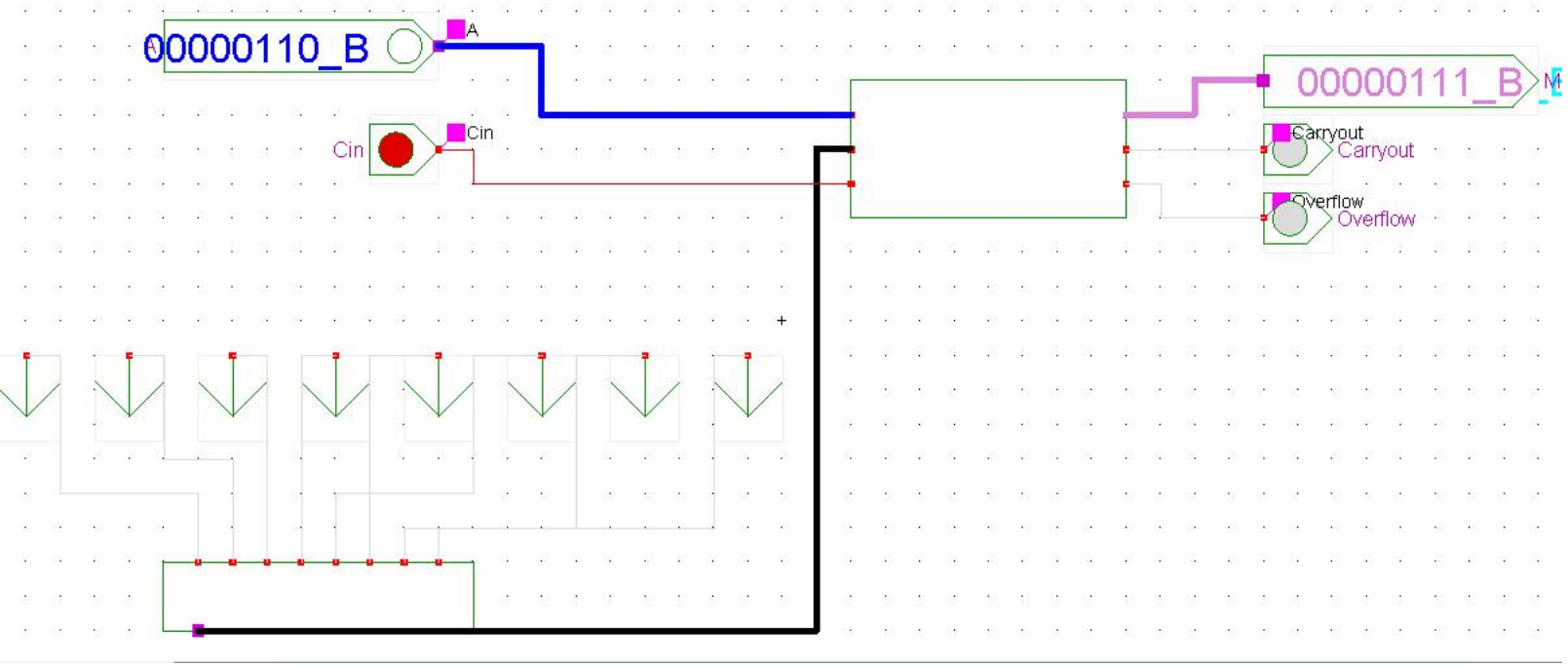
There were 3 test cases. The waveform matches the truth table and therefore the circuit works.

A	Cin	M
00000000	0	11111111
00000011	0	00000010
00000011	1	00000011



A + Cin:

Description: This circuit performs the function A+Cin. Basically if Cin is 0 the result is A. If Cin is 1 the result is A+1. This circuit is built using an 8 bit adder with one of the inputs fixed to 0. The output is the 8bit result M. This circuit will never overflow.



Testing:

There were 2 test cases. The waveform matches the truth table and therefore the circuit works.

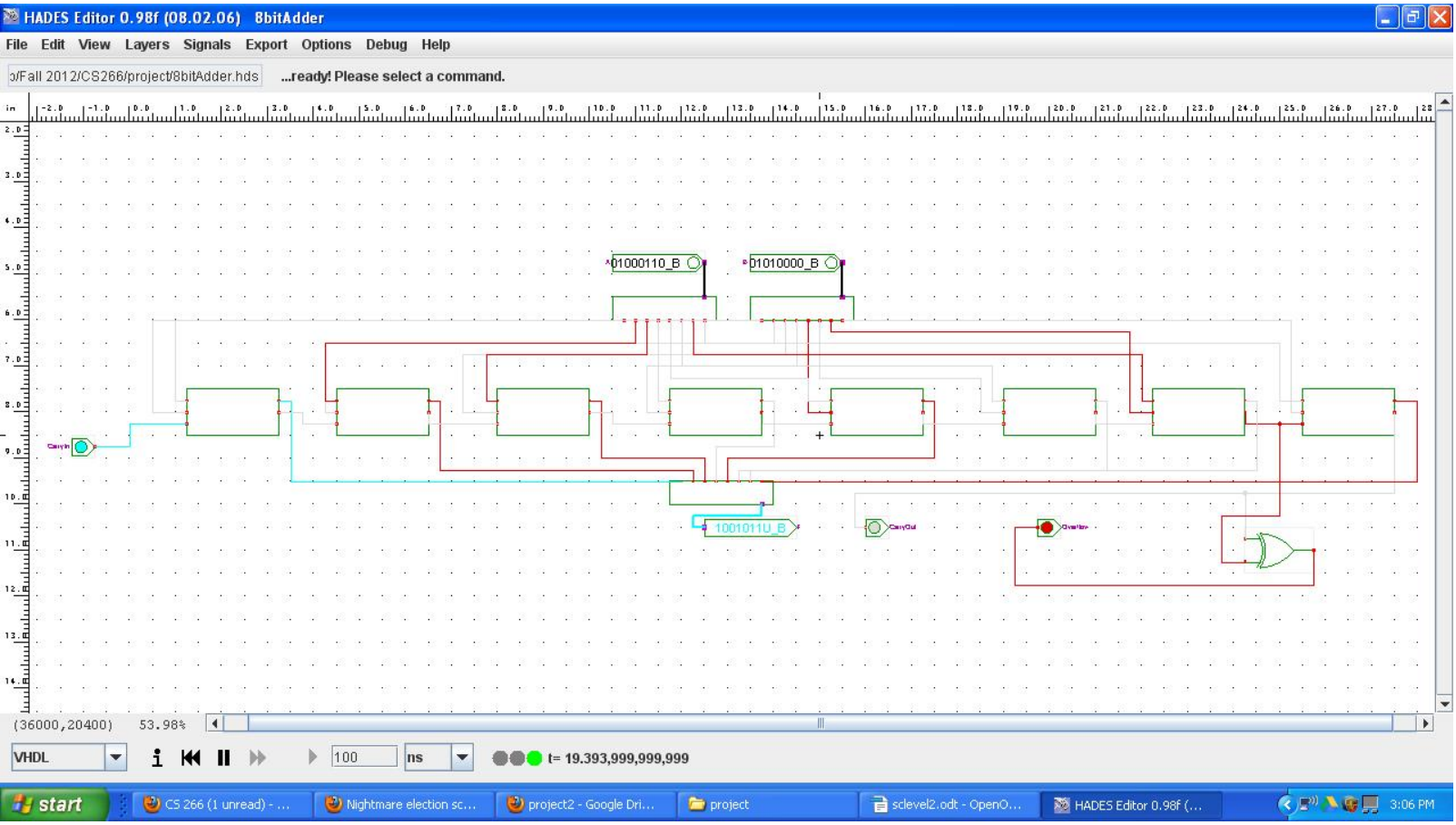
A	Cin	M
00000001	0	00000001
00000100	1	00000101

8 bit adder:

An 8 bit adder is a device that can add two 8 bit binary values. The output is 8 bits along with a carry out. The carry out is needed because its possible that the sum of two 8 bit numbers could be 9 bits. The carry out is the most significant bit, in this case the 2^8 bit. The carryout is ignored when overflow is 0. When overflow is 1, the carryout is used and the result is 9 bits. The adder is constructed using 8 full adders.

Inputs:
X1, X2- the two single bit inputs to be added
CarryIn- the single bit carry from the previous adder

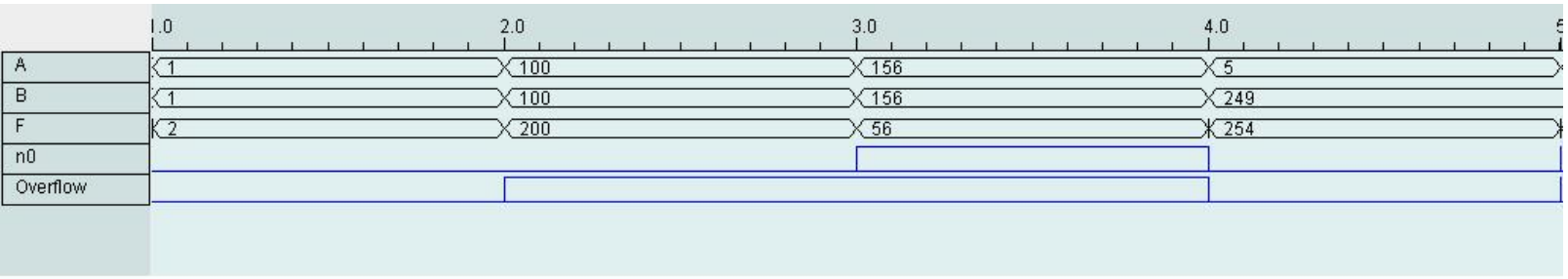
Output:
S- least significant bit
Cout- the carry out, most significant bit



Testing:
For testing I tested 1+1, 100+100, -100+-100, and 5+-7. These test were done to show that the adder works and the carryout and overflow activate properly. The waveform shows the circuit works.

A	B	M	Cout	Overflow
1	1	2	0	0
100	100	200	0	1

-100 -100 56 1 1

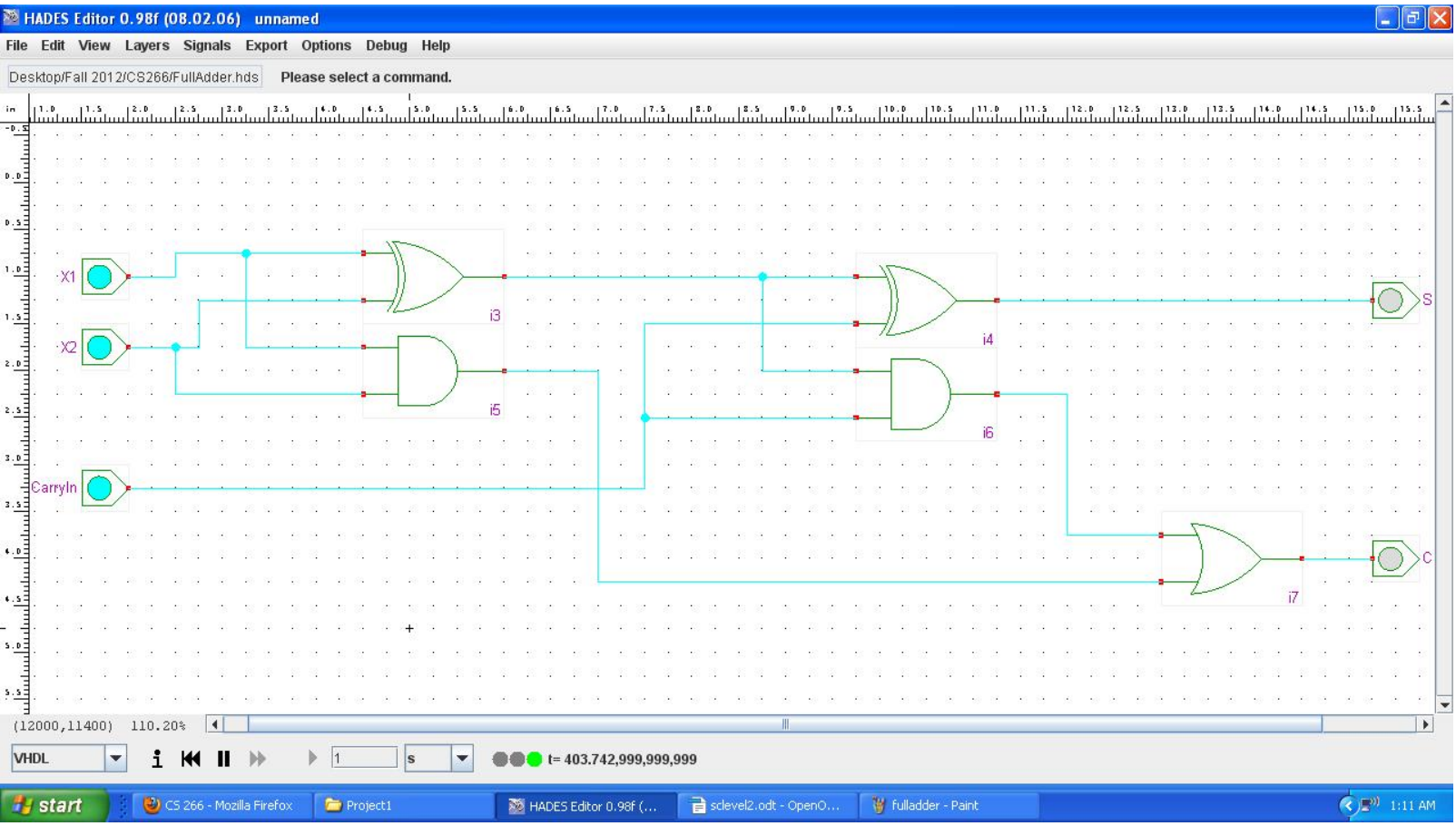


Full Adder:

A full adder is a device capable of adding 2 single bits plus a carry from the previous bit. It basically can add 1+1+1. The carry out from one full adder can be connected to the carry in to another full adder and produce a multipul bit adder.

Inputs:
X1, X2- the two single bit inputs to be added
CarryIn- the single bit carry from the previous adder

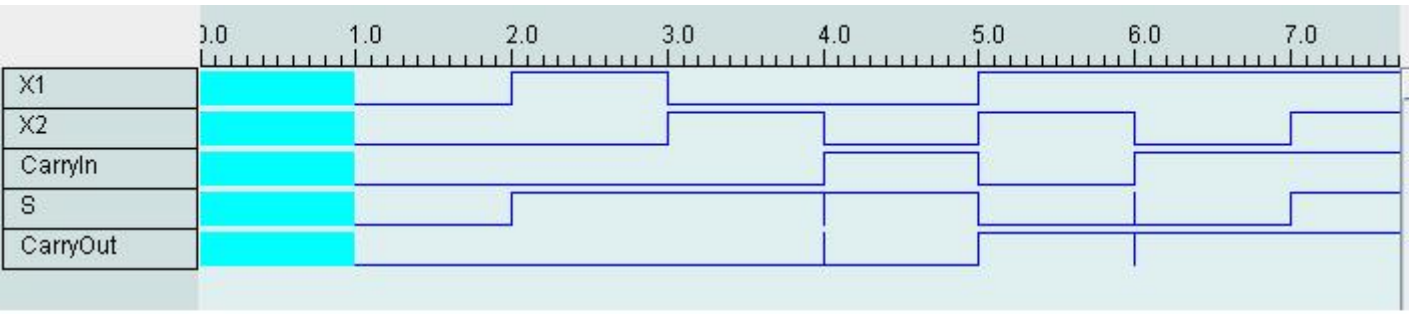
Output:
S- least significant bit
Cout- the carry out, most significant bit



Testing:

This is a simple circuit so I was able to test all possible inputs. The waveform matches the truth table.

X1	X2	Cin	S	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



Part 1a: 8-bit 4 to 1 Multiplexer:

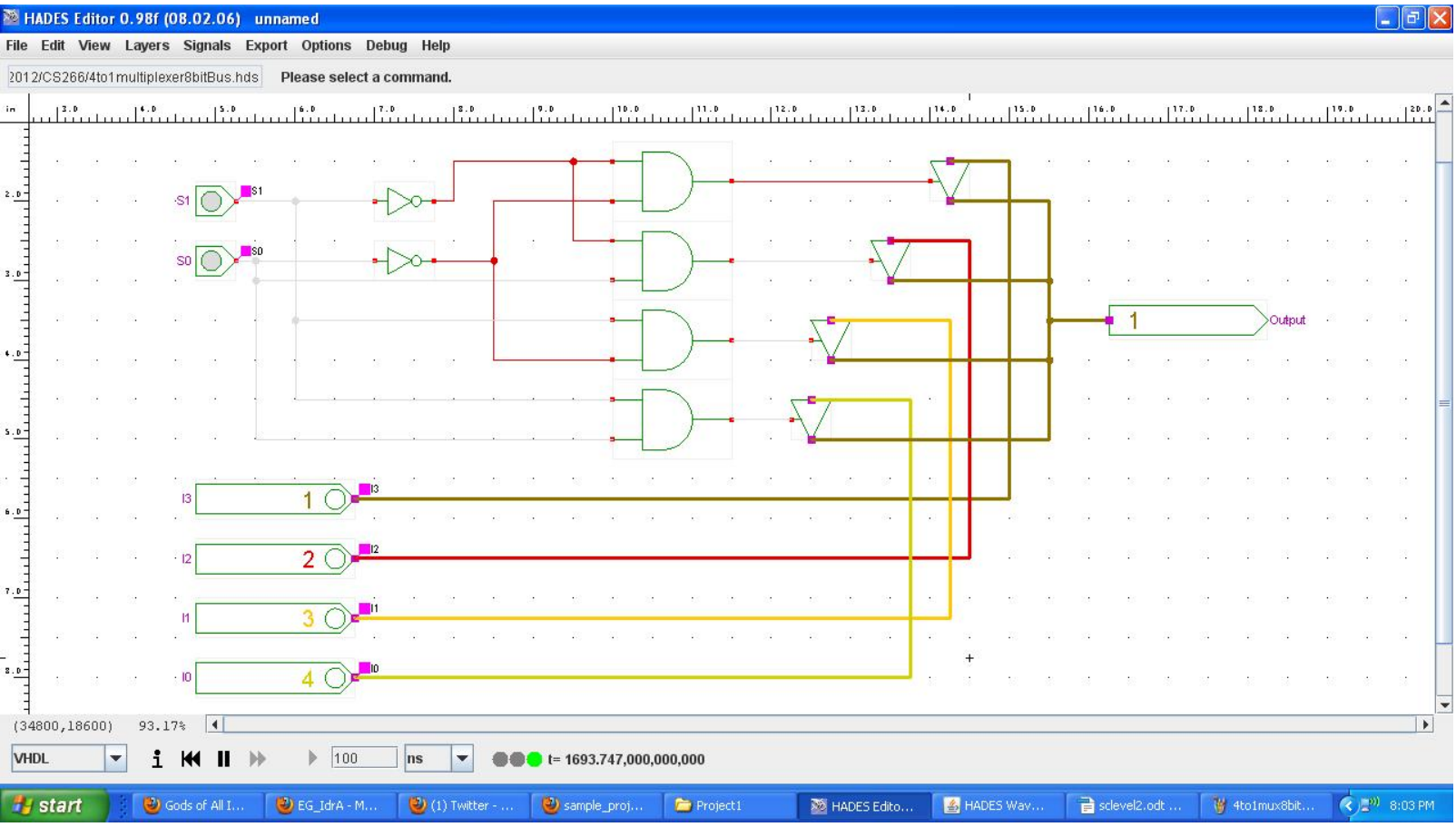
Part 1a – 4 to 1 multiplexer with 8-bit bus

A 4 to 1 multiplexer selects one of 4 inputs based on the input of a 2 bit selector. For each of the 4 possible bit combinations of the selector it passes the appropriate input to the output. For this particular multiplexer, the inputs and output will be 8-bit busses. In order to simplify the passing of the value from input to output, 8 bit tri-state buffers are used. The tri-states are enables by the selector bits.

Inputs:
S0, S1- selector bits
I0, I1, I2, I3- 8 bit busses

Outputs:
Output- 8 bit bus

Gates:
AND- 4
NOT- 2
TRI STATE- 4

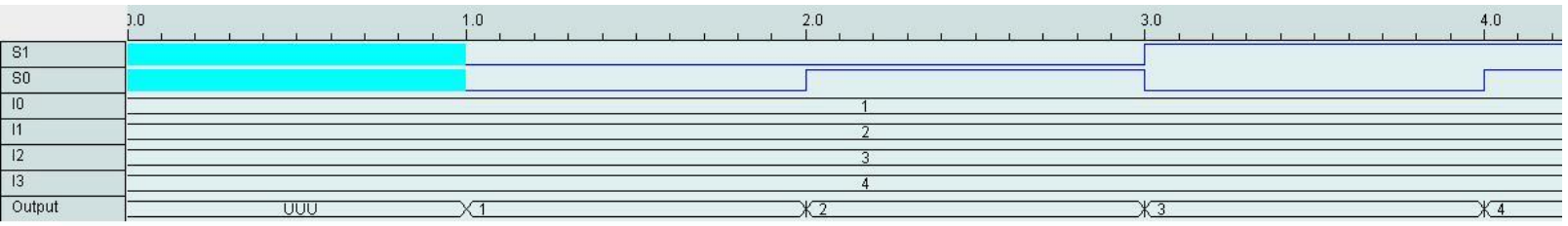


Testing:

I tested to show that each of the 4 bit combinations of the selector S0 and S1 passed the appropriate input bus to the output. S0S1= 00 should pass I0 to the output, 01 should pass I1, 10 should pass I2, and 11 should pass I3. For this test I made I0, I1, I2, I3 = 1,2,3,4 and checked that the value was passed to the output.

Truth Table:

S1	S0	I0	I1	I2	I3	Output
0	0	1	x	x	x	1
0	1	x	2	x	x	2
1	0	x	x	3	x	3
1	1	x	x	x	4	4



4 to 1 Multiplexer:

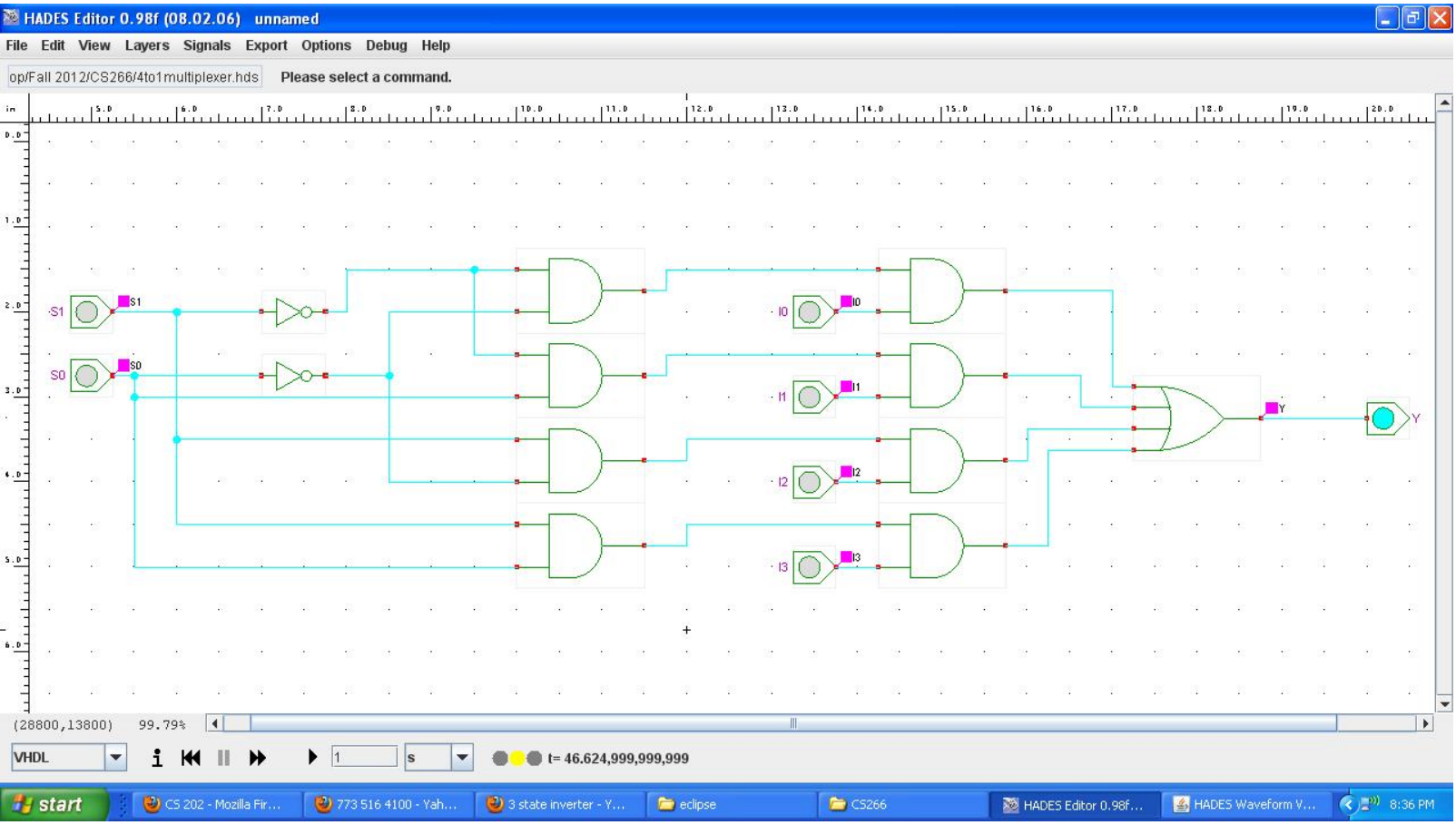
The 4-to-1 multiplexer takes 2 selector inputs, S0 and S1, and 4 single-bit inputs, I0, I1, I2, and I3. Each input combination of the selectors allows one of I0, I1, I2, or I3 to pass to the output Y.

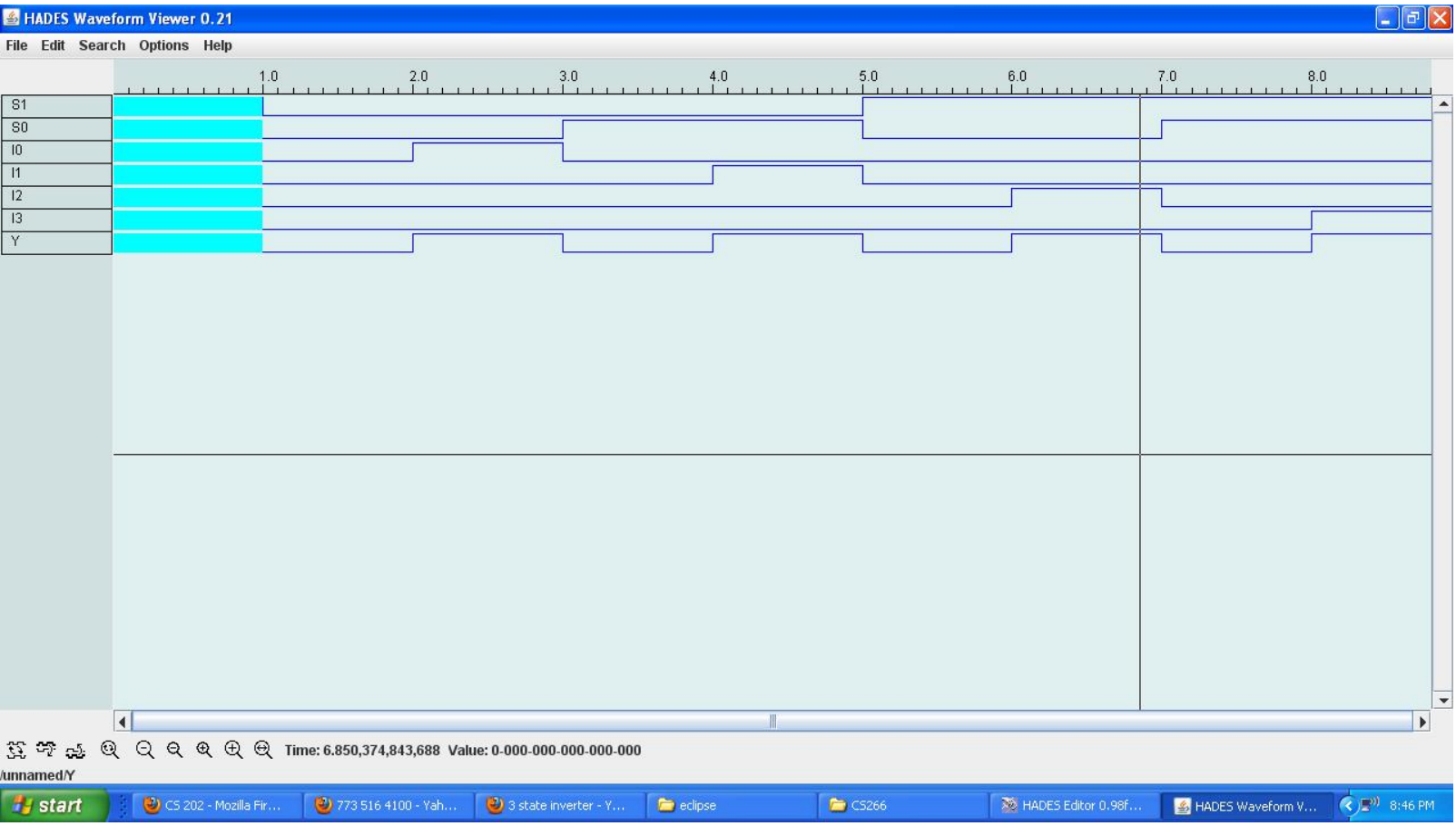
For testing, there are 64 possible input combinations but I think it is only necessary to test 8 in order to determine that the circuit works. For each bit combination of S0 and S1 I will test to make sure that it can pass a 1 and a 0 from the proper input. Since there are 4 selector bit combinations and both a 1 and 0 is tested at each one, it equals a total of 8 tests. These test will show that the

circuit can properly implement the function.

Truth Table:

S1	S0	I0	I1	I2	I3	Y
0	0	0	x	x	x	0
0	0	1	x	x	x	1
0	1	x	0	x	x	0
0	1	x	1	x	x	1
1	0	x	x	0	x	0
1	0	x	x	1	x	1
1	1	x	x	x	0	0
1	1	x	x	x	1	1





References:

Mano, M. Morris, and Charles R. Kime. *Logic and Computer Design Fundamentals*. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2007. Print