Lab 5 Report

Goal: The objective of this lab is to design complex arithmetic logic using Cadence.

Function:

Logarithmic multiplier performs multiplication of a pair of input numbers A and B by adding their logarithms as follows:

$A*B=\log^{-1}(\log A+\log B)$

Inputs:

A, B: 5-bit inputs to the multiplier.

Outputs:

M: Output from the multiplier

F: Overflow flag

Steps:

1. Made 5:32 decoder which is used for making log rom which is used to find the log values of the number to be multiplied.
2. 8 bit Carry look ahead adder designed using two 4 bit carry look ahead adder.
3. Designed -100 and -200 filters to reduce these values to the desired range i.e. the number above 100 are subtracted from -100 filter and numbers above 200 are subtracted from -200 filter and the numbers below 100 are just passed as it is to the antilog rom.
4. Designed a 6:64 decoder using 3:8 decoders which used make the antilog rom to find the antilog of the summation of two log numbers i.e. the numbers which are to be multiplied.
5. Assembled the individual circuits to design a final top level schematic.

Function of the circuit:

The two numbers to be multiplied are given to the two log rom to find the log of those numbers. These log values are then given to the carry look ahead adders to find the sum and carry of those log numbers. These values are given to the filters to get the values within the values within the desired range. These values are then passed to antilog rom to get the output which is approximately closed to the desired output.

Results:

Case1: a=7, b=9

When the inputs are 7 and 9 the output should be 63. When we give these as the input to the terminal, the output we get is 617 with flags as 010. Which mean there is a decimal point after 1 bit which indicates the result is 61.7 which is close to 63.

Case 2: a=31, b=31

When both the inputs are 31 the output should be 961. The output we get is 955 with flag bits as 001 which says that there is no decimal point and the result is valid.

Case 3: a=0, b=0:

When we give this as the input the answer we get is invalid as shown in the graph because log of 0 is not defined.

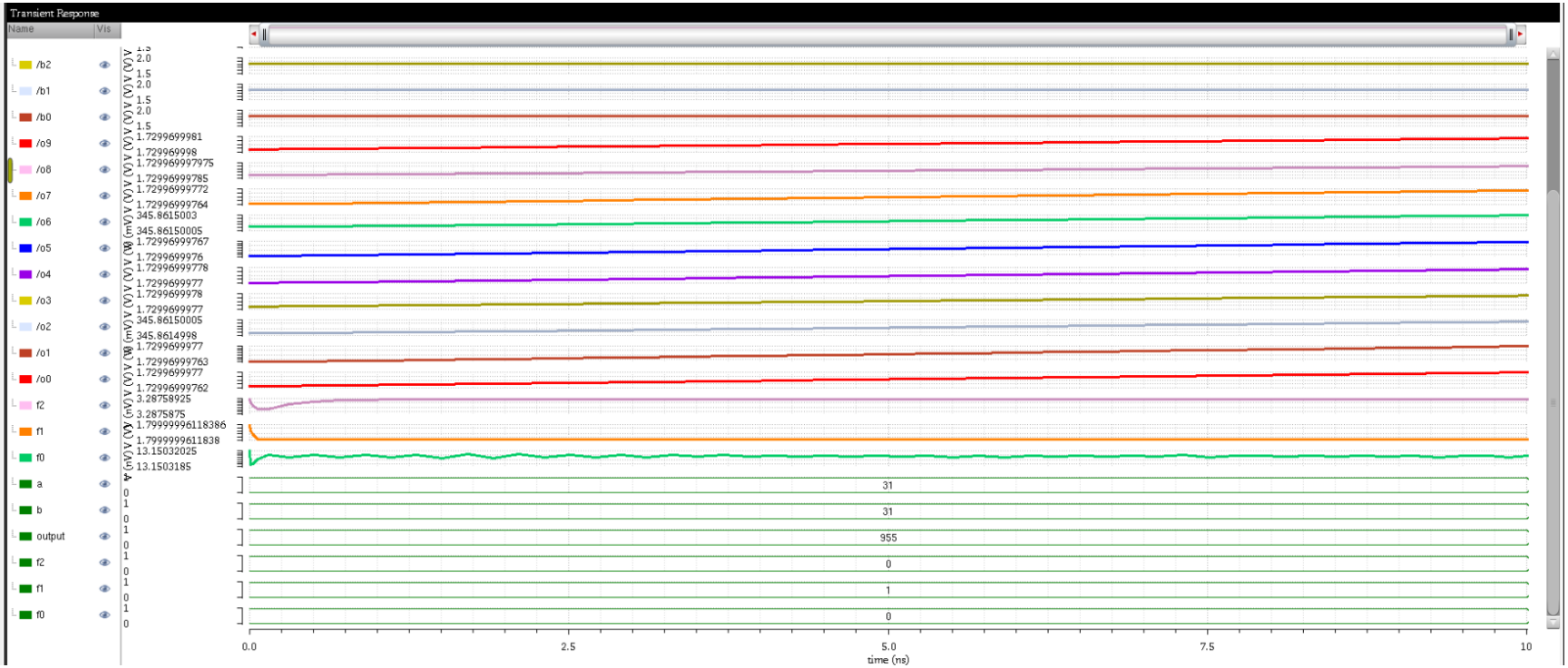
Conclusion:

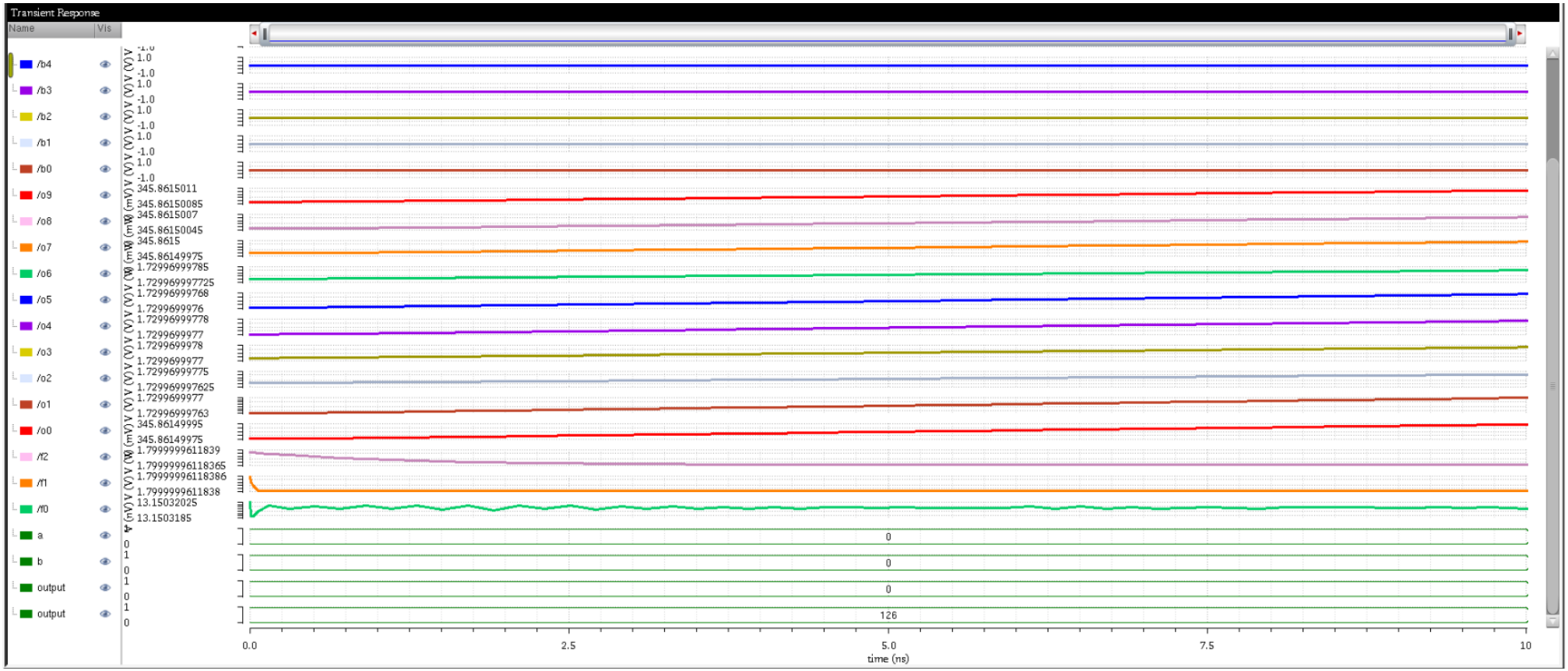
The multiplier we designed gives us outputs which are close to the desired result but not exact. This because of the rounding off that we did to reduce the number of values from 99 to 64. This gives us inaccuracies while using it in some applications where accurate results are required.

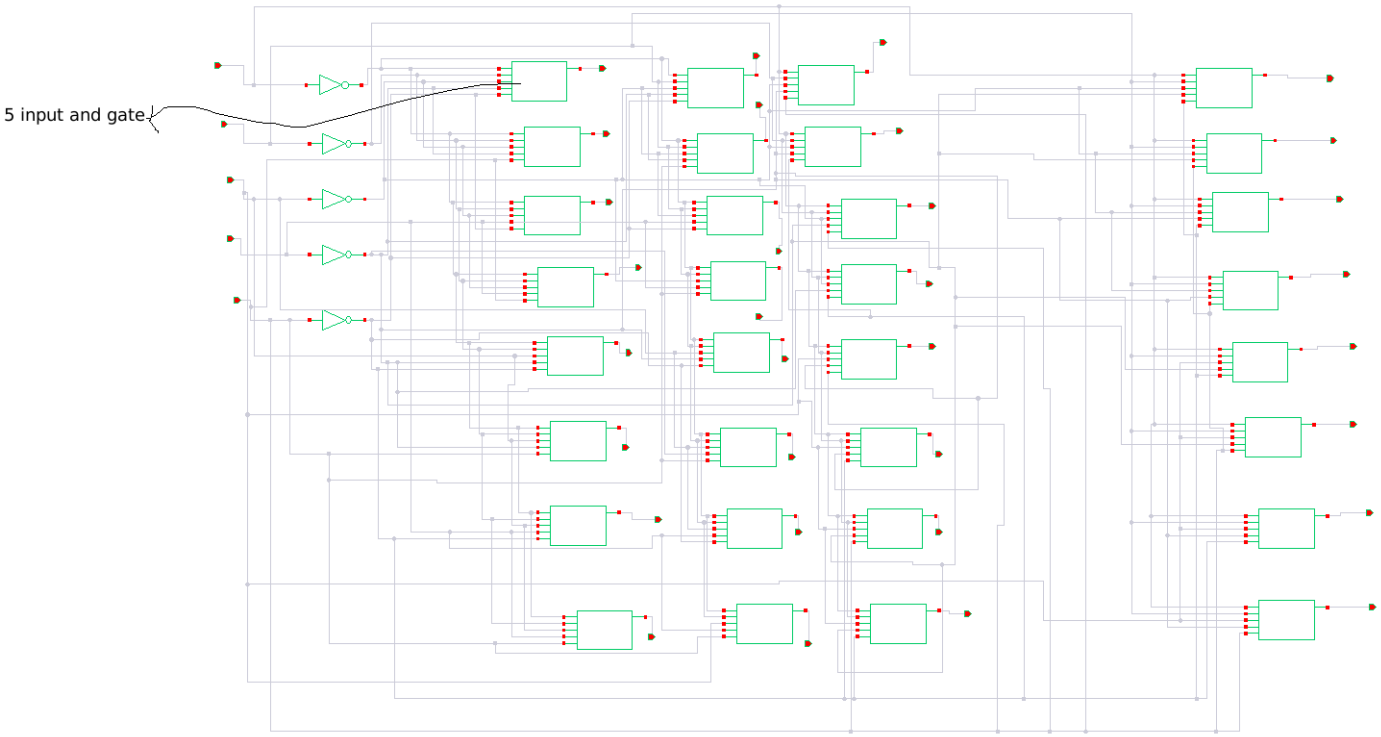
Question:

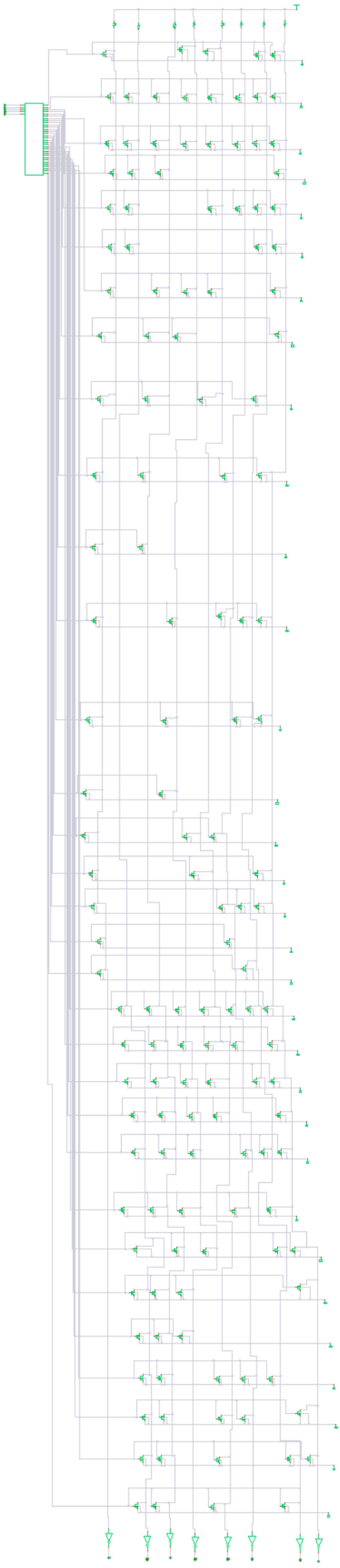
Multiplication being a repetitive method can be performed using adders. But the disadvantage of this is that if we are to multiply two large numbers, the number of bits that we would need to add would increase and that would in turn increase the delay.

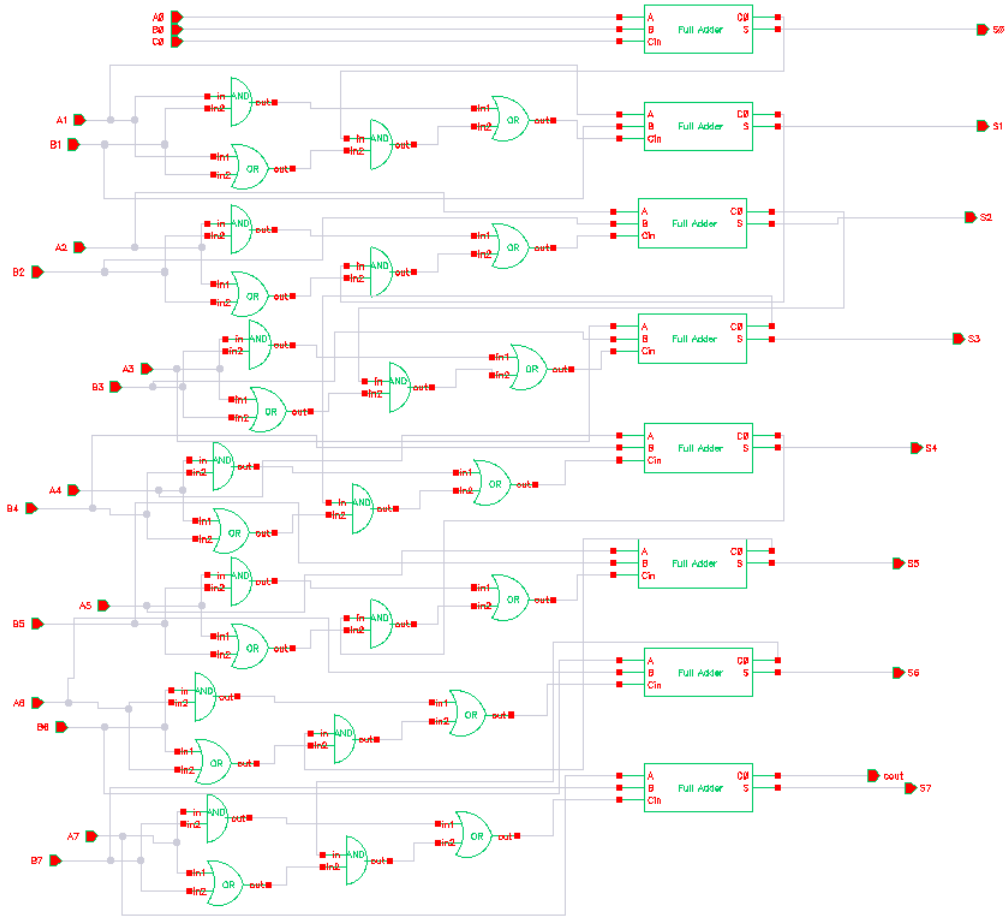

Transient Response
Name
Vis
/b1
/b0
/o9
/o8
/o7
/o6
/o5
/o4
/o3
/o2
/o1
/o0
f2
/f1
/f0
a
a
b
output
1.0
-1.0
2.0
1.5
1.7299699981
1.729969998
345.8615007
345.86150045
345.8615
345.86149975
1.72996999785
1.729969997725
1.72996999768
1.7299699976
345.86150015
345.86149995
1.7299699978
1.7299699977
345.86150005
345.8614998
345.86149995
345.86149975
1.7299699977
1.72996999763
3.28758925
3.2875875
3.2875805
3.287578
1.79999996118386
1.7999999611838
0
7
9
617

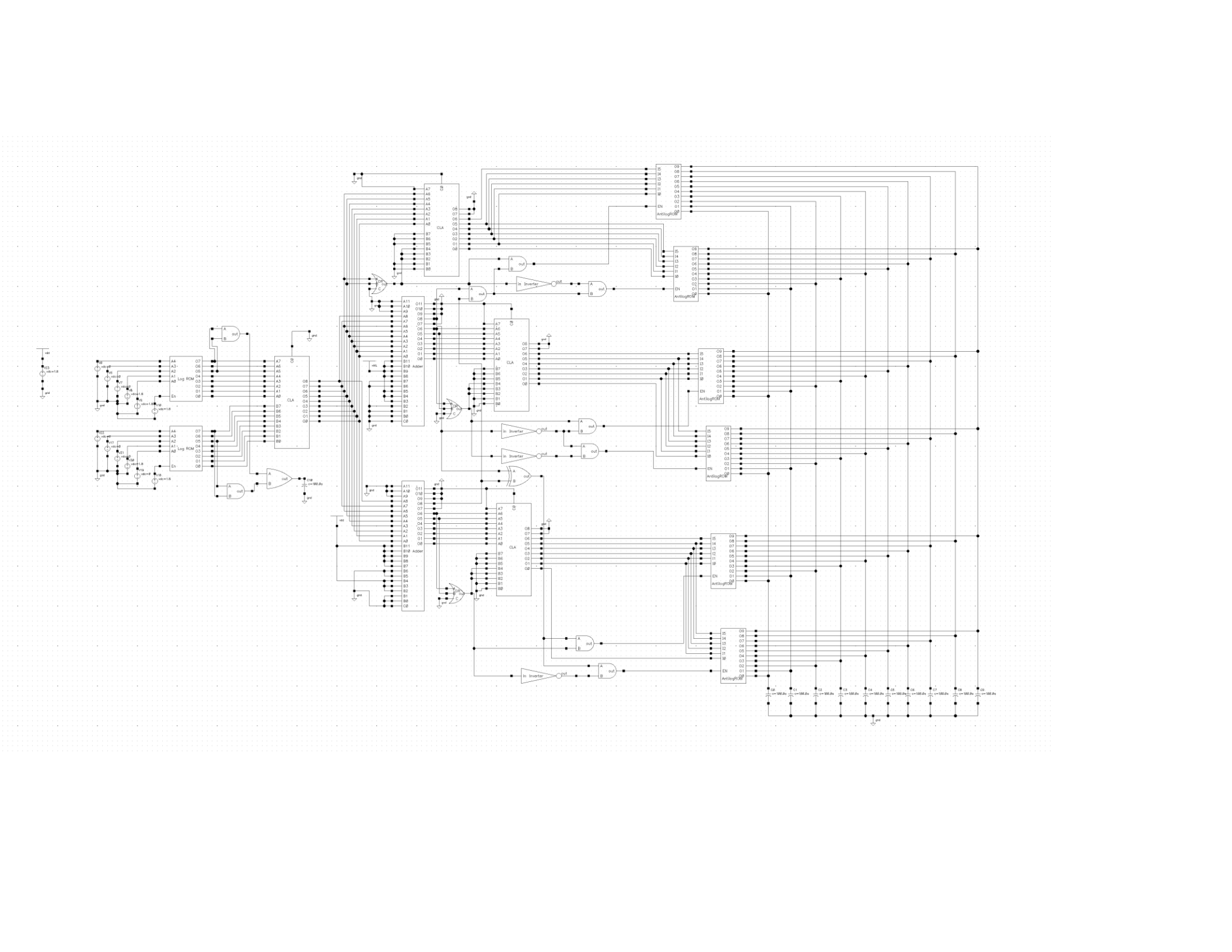

Transient Response
Name
Vis
/b2
/b1
/b0
/o9
/o8
/o7
/o6
/o5
/o4
/o3
/o2
/o1
/o0
f2
f1
f0
a
b
output
f2
f1
f0
1.5
2.0
1.5
2.0
1.5
2.0
1.5
1.7299699981
1.729969998
1.729969997975
1.72996999785
1.72996999772
1.72996999764
345.8615003
345.86150005
1.72996999767
1.7299699976
1.72996999778
1.7299699977
1.7299699978
1.7299699977
345.86150005
345.8614998
1.7299699977
1.72996999763
1.7299699977
1.72996999762
3.28758925
3.2875875
1.799999961183 86
1.79999999611838
13.15032025
13.1503185
31
31
955
0
1
0
0.0
2.5
5.0
7.5
10
time (ns)

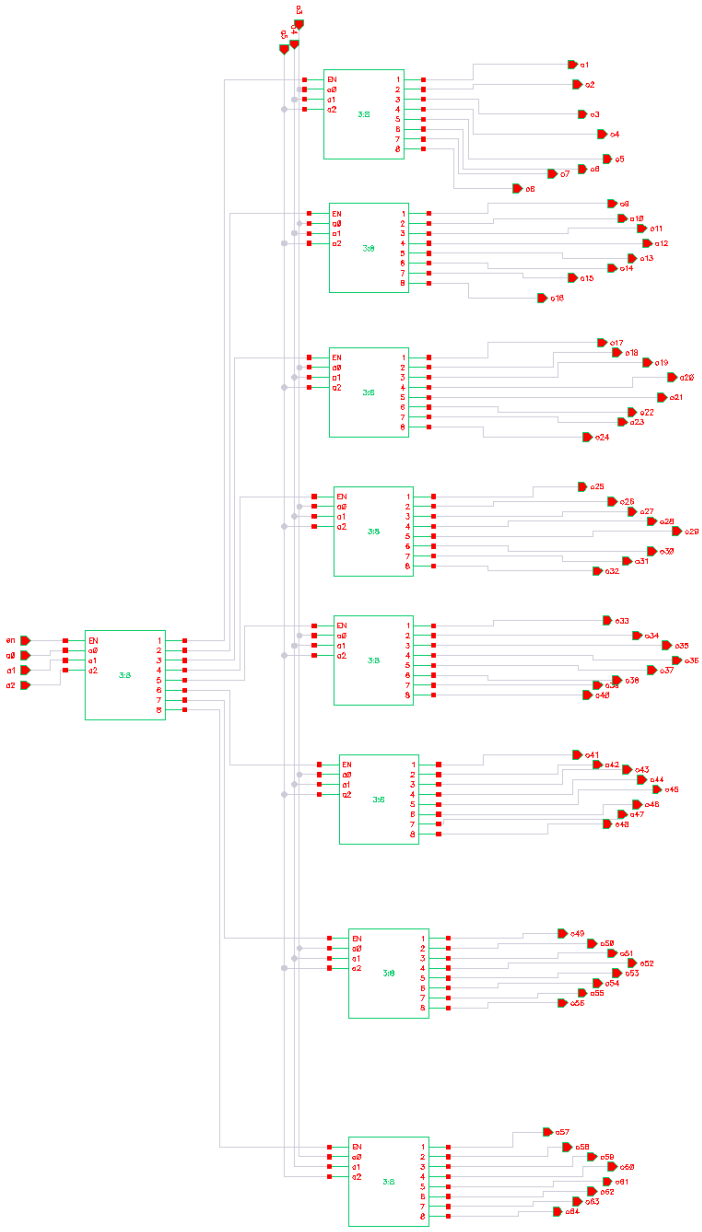

Transient Response
Name
Vis
/b4
/b3
/b2
/b1
/b0
/o9
/o8
/o7
/o6
/o5
/o4
/o3
/o2
/o1
/o0
/f2
/f1
/f0
a
b
output
output
1.0
-1.0
345.8615011
345.86150085
345.8615007
345.86150045
345.8615
345.86149975
1.72996999785
1.729969997725
1.72996999768
1.7299699976
1.72996999778
1.7299699977
1.7299699978
1.7299699977
1.72996999775
1.729969997625
1.7299699977
1.72996999763
345.86149995
345.86149975
1.7999999611839
1.79999996118365
1.79999996118386
1.7999999611838
13.15032025
13.1503185
0
1
0
1
0
1
0
0
0
0
126
0.0
2.5
5.0
7.5
10
time (ns)


5 input and gate




A0
B0
C0
A1
B1
A2
B2
A3
B3
A4
B4
A5
B5
A6
B6
A7
B7
Full Adder
A
B
Cin
Cout
S
AND
OR
in1
in2
out
S0
S1
S2
S3
S4
S5
S6
cout
S7




EN
a0
a1
a2
3:8
o1
o2
o3
o4
o5
o6
o7
o8
o9
o10
o11
o12
o13
o14
o15
o16
o17
o18
o19
o20
o21
o22
o23
o24
o25
o26
o27
o28
o29
o30
o31
o32
o33
o34
o35
o36
o37
o38
o40
o41
o42
o43
o44
o45
o46
o47
o48
o49
o50
o51
o52
o53
o54
o55
o56
o57
o58
o59
o60
o61
o62
o63
o64