VLSI DESIGN

PROJECT

Name: Anushka Agrawal

Roll No: 2021102023

4x4 MULTIPLIER

NGSPICE:

The verification for subcircuits used for the construction of 4x4 Multiplier:

The ngspice netlist used for the verification of the subcircuits:

testinggates.sp

```
.include TSMC_180nm.txt
.include and.sub
.include or.sub
.include xor.sub
.include fadder.sub
** Parameters **
.PARAM Lmin=180n
.PARAM Wmin=180n
.PARAM XX = 1
.PARAM tr=0.1p
.PARAM pvdd = 1
.temp 25
** Input Voltages **
VDS vdd 0 dc='pvdd'
GRD gnd 0 dc=0

V1 in1 gnd pulse pvdd 0 0 100p 100p 10n 20n
V2 in2 gnd pulse pvdd 0 0 100p 100p 20n 40n
V3 in3 gnd pulse pvdd 0 0 100p 100p 40n 80n

** Circuit Description **
* xand in1 in2 out vdd gnd and
* xor in1 in2 out vdd gnd or
* xxor in1 in2 out vdd gnd xor
xfadd in1 in2 in3 cout sum vdd gnd fadder

** Analysis **
.tran 1p 80n

** Plotting **
.control
run
* plot out in1+2 in2+4
plot cout sum+2 in1+4 in2+6 in3+8
.endc
.end

```

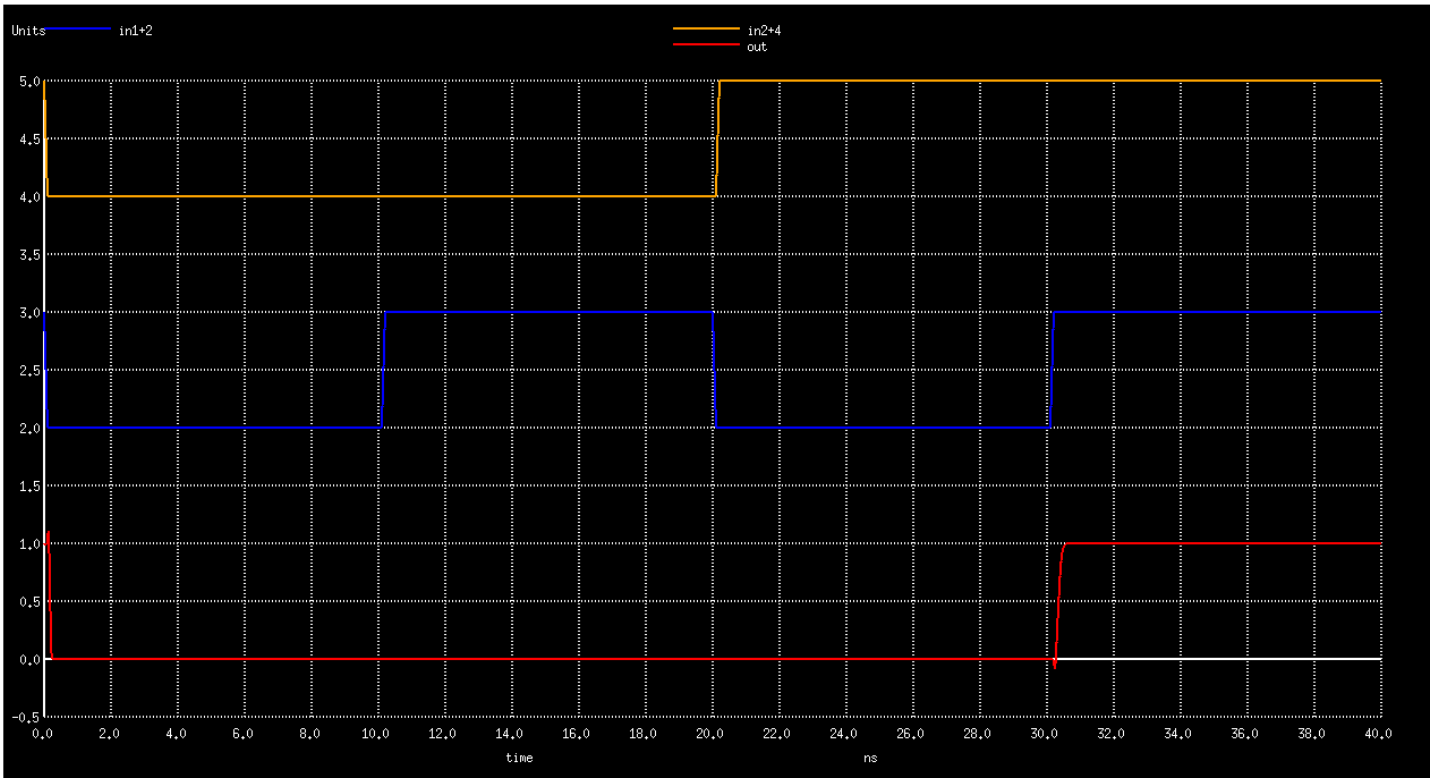
AND GATE:

The ngspice subcircuit for AND Gate:

and.sub

```
.subckt and inp1 inp2 out vdd gnd

Mp1 outn    inp1    vdd     vdd     CMOSP W={2*XX*Wmin} L={Lmin}
Mp2 outn    inp2    vdd     vdd     CMOSP W={2*XX*Wmin} L={Lmin}
Mn1 outn    inp1    nodez   gnd     CMOSN W={2*XX*Wmin} L={Lmin}
Mn2 nodez   inp2    gnd     gnd     CMOSN W={2*XX*Wmin} L={Lmin}

Mp3 out  outn   vdd     vdd     CMOSP W={2*XX*Wmin} L={Lmin}
Mn3 out  outn   gnd     gnd     CMOSN W={2*XX*Wmin} L={Lmin}

Cout out gnd 3f

.ends
```

The simulation results observed from this AND gate:



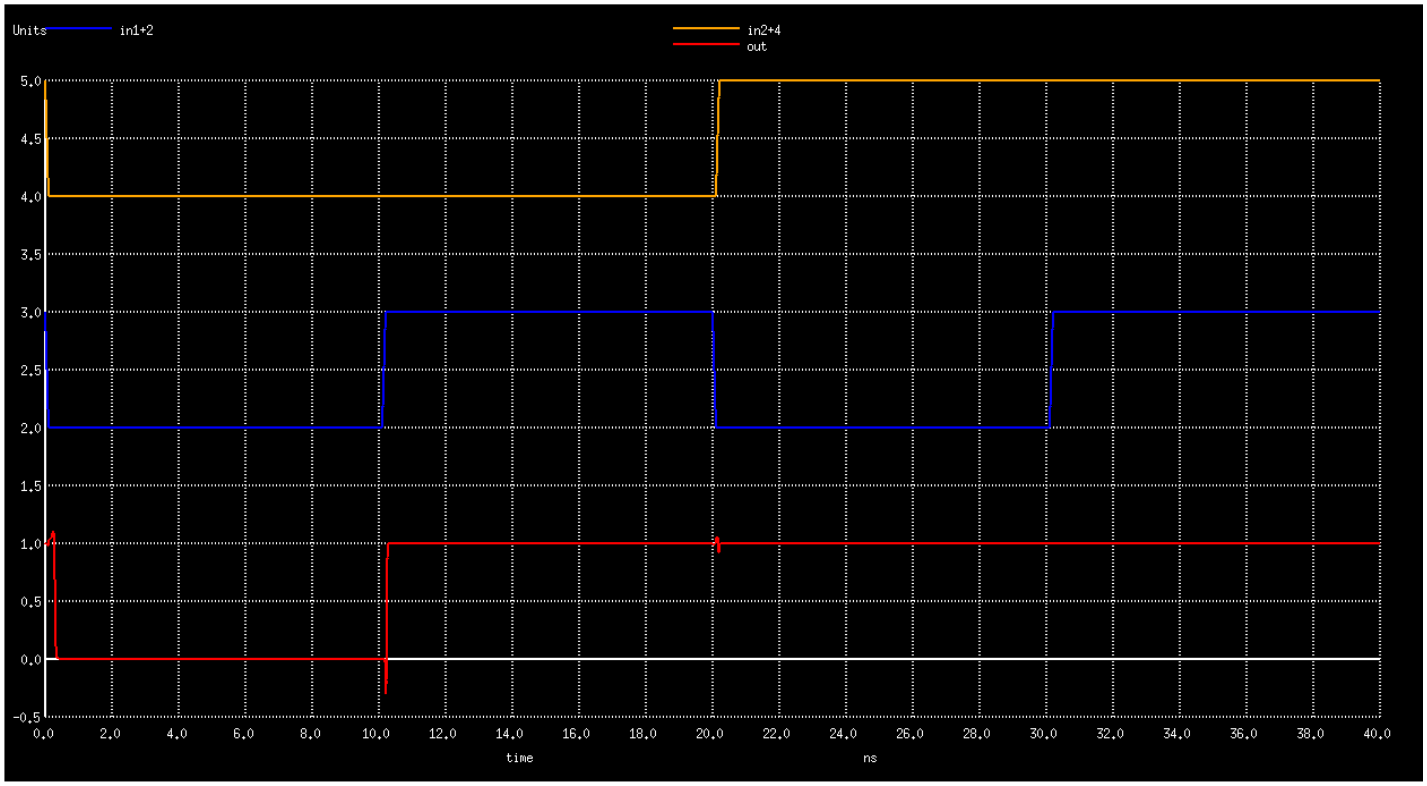
OR GATE:

The ngspice subcircuit for OR Gate:

or.sub

```
.subckt or in1 in2 out vdd gnd

MN1 n01 in1 gnd gnd CMOSN W={2*XX*Wmin} L={Lmin}
MN2 n01 in2 gnd gnd CMOSN W={2*XX*Wmin} L={Lmin}
MP1 n01 in1 n02 n02 CMOSP W={2*XX*Wmin} L={Lmin}
MP2 n02 in2 vdd vdd CMOSP W={2*XX*Wmin} L={Lmin}


MN3 out n01 gnd gnd CMOSN W={2*XX*Wmin} L={Lmin}
MP3 out n01 vdd vdd CMOSP W={2*XX*Wmin} L={Lmin}
.ends

```

The simulation results observed from this OR gate:



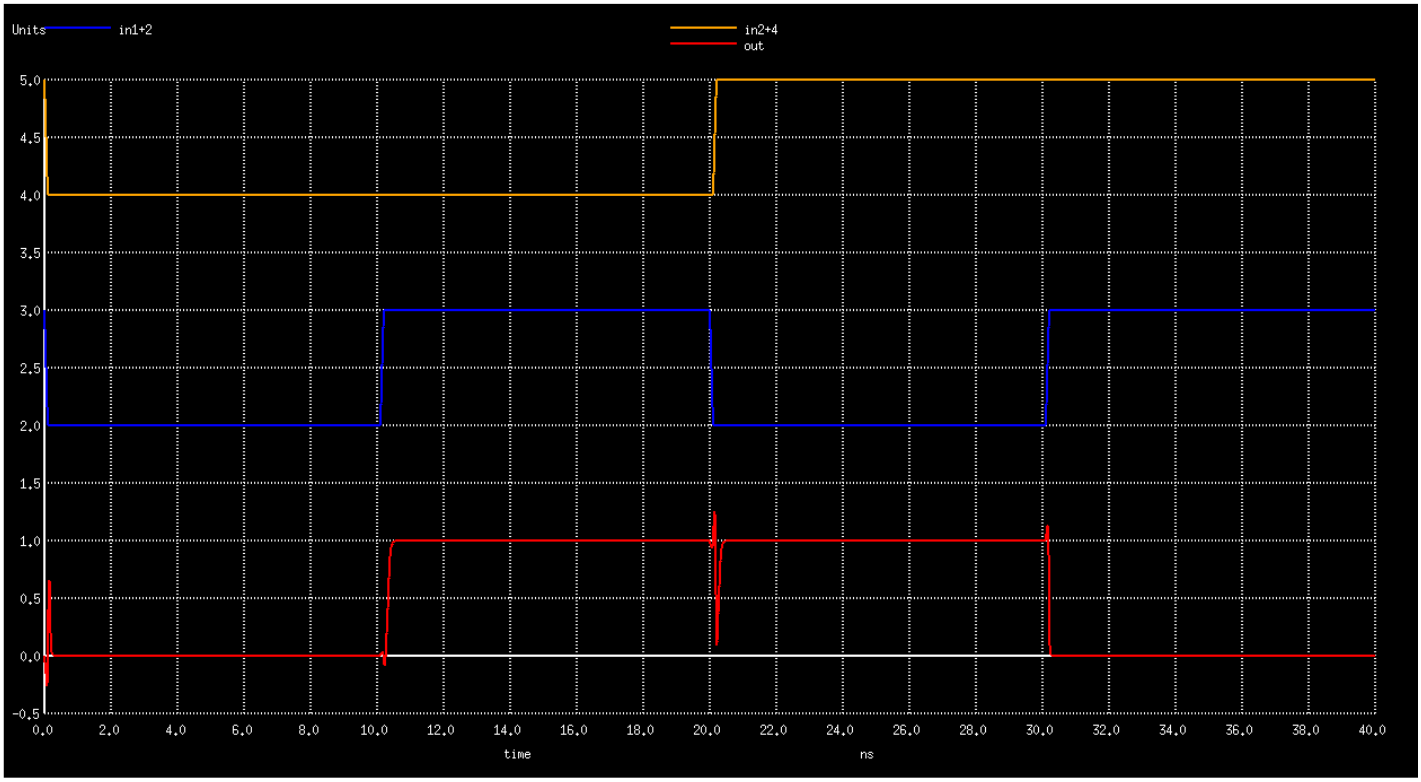
XOR GATE:

The ngspice subcircuit for XOR Gate:

xor.sub

```
.subckt not out in1 vdd gnd
MN1 out in1 gnd gnd CMOSN W={2*XX*Wmin} L={Lmin}
MP1 out in1 vdd vdd CMOSP W={2*XX*Wmin} L={Lmin}
.ends

.subckt twopmos n01 n02 in1 in2
MP1 n03 in1 n01 n01 CMOSP W={2*XX*Wmin} L={Lmin}
MP2 n02 in2 n03 n03 CMOSP W={2*XX*Wmin} L={Lmin}
.ends

.subckt twonmos n01 n02 in1 in2
MP1 n01 in1 n03 n03 CMOSN W={2*XX*Wmin} L={Lmin}
MP2 n03 in2 n02 n02 CMOSN W={2*XX*Wmin} L={Lmin}
.ends

.subckt xor in1 in2 out vdd gnd
xnot1 nn1 in1 vdd gnd not
xnot2 nn2 in2 vdd gnd not
xtwopmos1 vdd out nn1 in2 twopmos
xtwopmos2 vdd out in1 nn2 twopmos
xtwonmos1 out gnd in1 in2 twonmos
xtwonmos2 out gnd nn1 nn2 twonmos
.ends
```

The simulation results observed from this XOR gate:



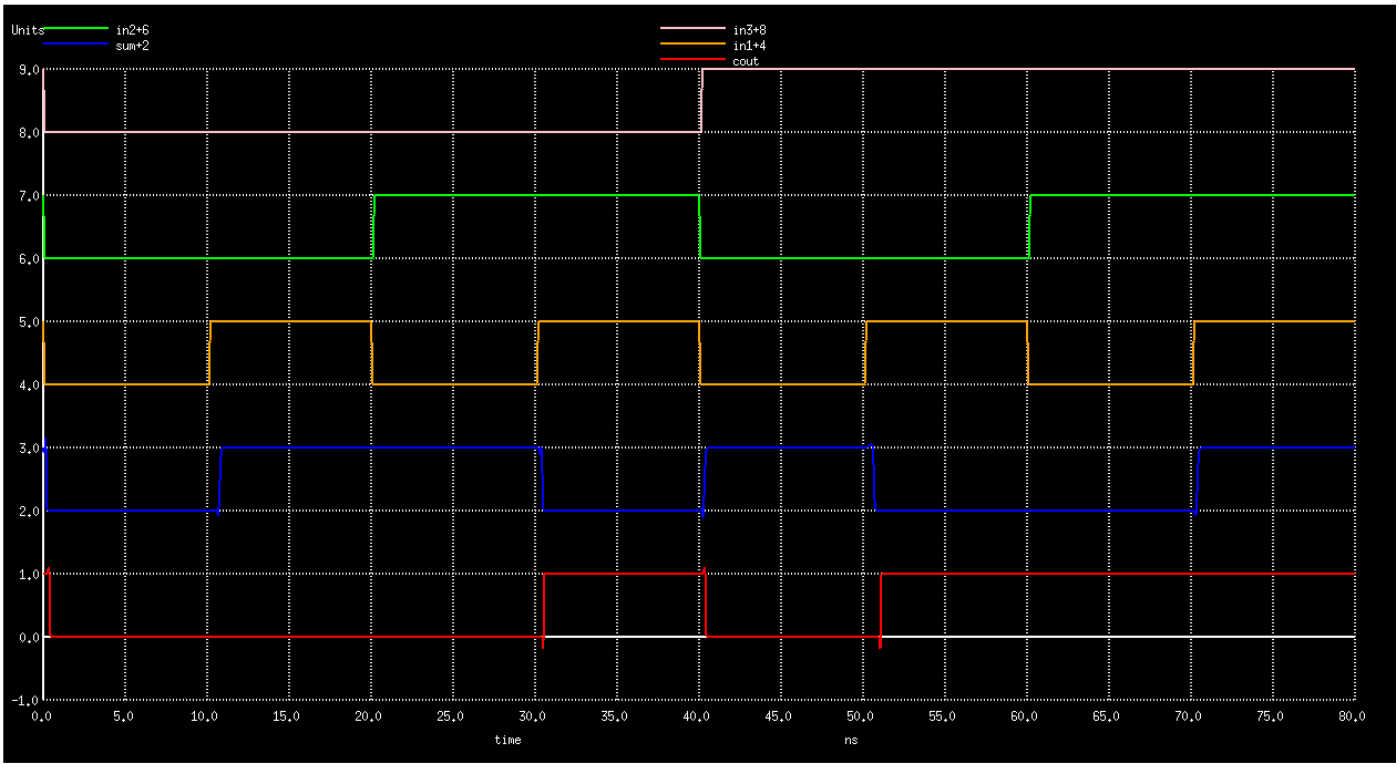
FULL ADDER:

The ngspice subcircuit for Full Adder:

fadder.sub

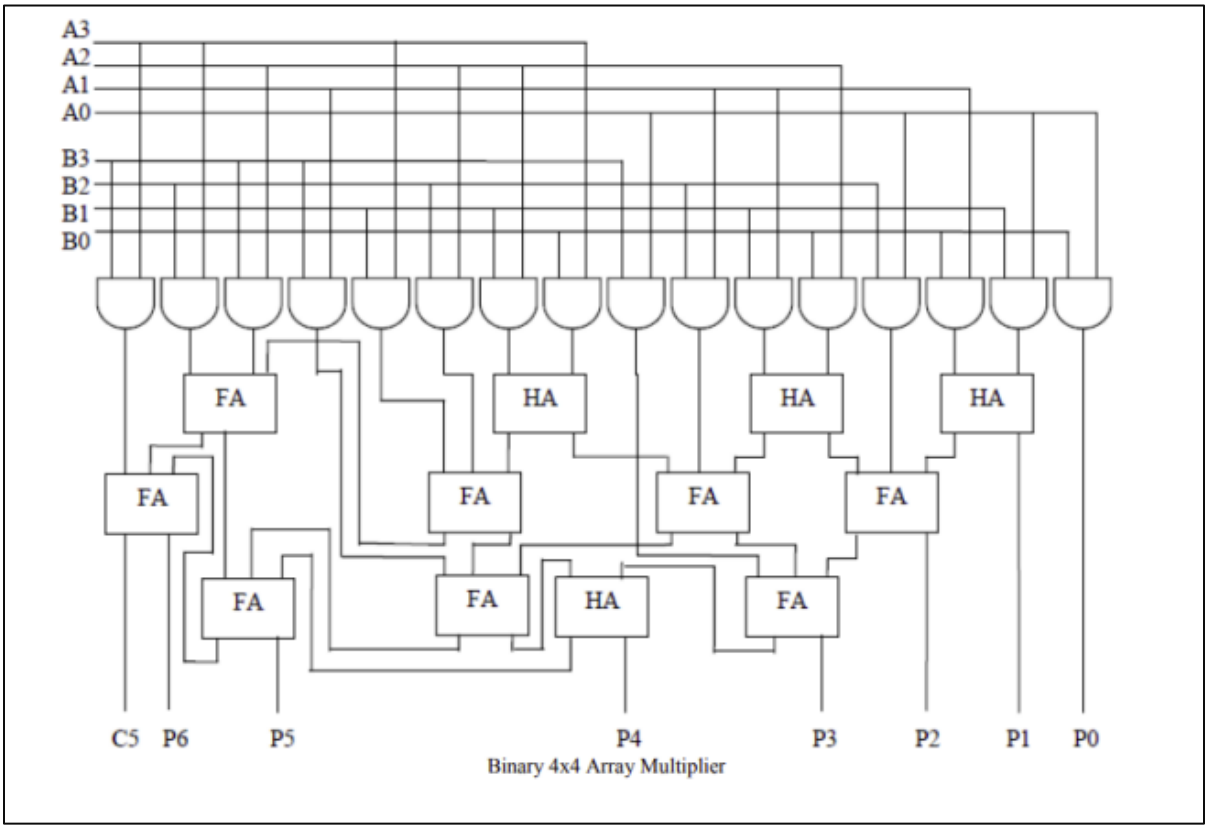
```
.include xor.sub
.include and.sub
.include or.sub

.subckt halfadd in1 in2 cout sum vdd gnd

xxor in1 in2 sum vdd gnd xor
xand in1 in2 cout vdd gnd and

.ends

.subckt fadder in1 in2 cin cout sum vdd gnd

xhalfadder1 in1 in2 x y vdd gnd halfadd
xhalfadder2 cin y z sum vdd gnd halfadd
x_or x z cout vdd gnd or

.ends
```

The simulation results observed from this Full Adder:



4x4 MULTIPLIER

The above subcircuits were used to make the final 4x4 multiplier. We design the circuit of the multiplier in the following way:



Binary 4x4 Array Multiplier

The ngspice netlist for the above circuit diagram for 4x4 Multiplier:

4x4mul.sp

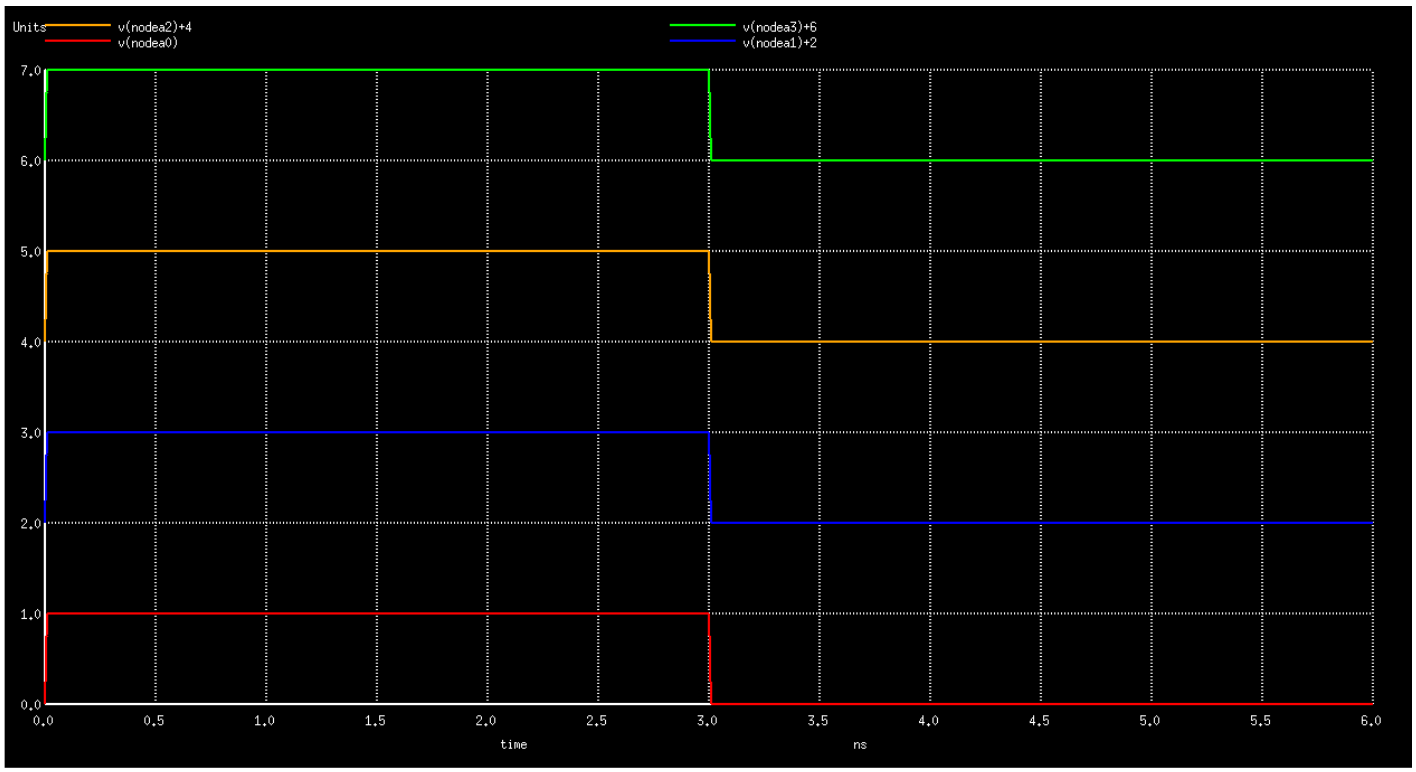
```
.INCLUDE TSMC_180nm.txt
.include and.sub
.include fadder.sub
.include xor.sub
.include or.sub

*****Parameters*****
.PARAM Lmin=180n
.PARAM Wmin=180n
.PARAM XX = 1
.PARAM tr=10p
.PARAM pvdd = 1
.temp 25

VDS vdd 0 dc='pvdd'
GRD gnd 0 dc=0


VinA3 nodeA3 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinA2 nodeA2 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinA1 nodeA1 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinA0 nodeA0 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinB3 nodeB3 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinB2 nodeB2 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinB1 nodeB1 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)
VinB0 nodeB0 gnd PWL ( {0*6000p} 0 {0*6000p+tr} 1 {0*6000p+3000p} 1 {0*6000p+3000p+tr} 0)

```

4x4mul.sp

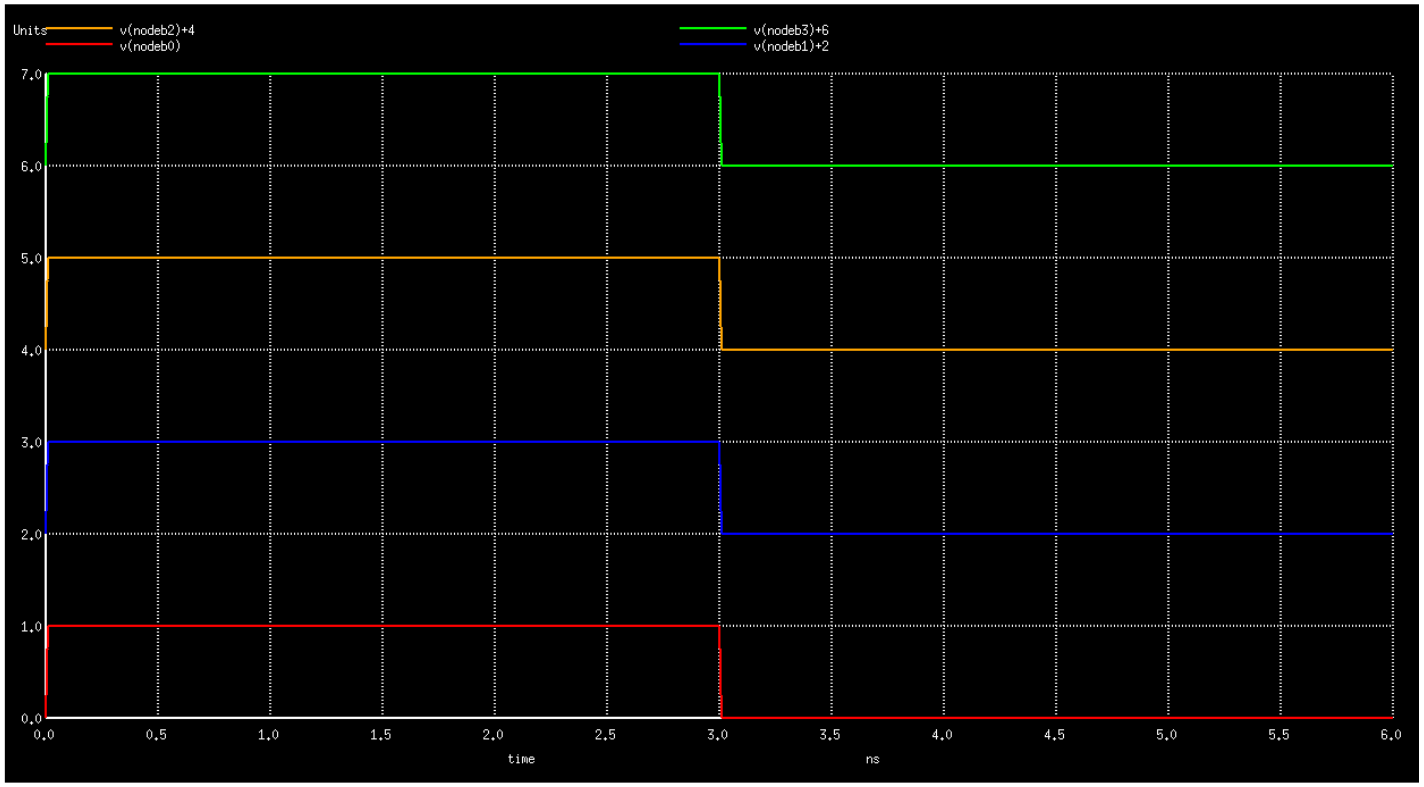
```
Xand1 nodeB3 nodeA3 a1 vdd gnd and
Xand2 nodeB2 nodeA3 a2 vdd gnd and
Xand3 nodeB3 nodeA2 a3 vdd gnd and
Xand4 nodeB3 nodeA1 a4 vdd gnd and
Xand5 nodeB1 nodeA3 a5 vdd gnd and
Xand6 nodeB2 nodeA2 a6 vdd gnd and
Xand7 nodeB1 nodeA2 a7 vdd gnd and
Xand8 nodeB0 nodeA3 a8 vdd gnd and
Xand9 nodeB3 nodeA0 a9 vdd gnd and
Xand10 nodeB2 nodeA1 a10 vdd gnd and
Xand11 nodeB1 nodeA1 a11 vdd gnd and
Xand12 nodeB0 nodeA2 a12 vdd gnd and
Xand13 nodeB2 nodeA0 a13 vdd gnd and
Xand14 nodeB0 nodeA1 a14 vdd gnd and
Xand15 nodeB1 nodeA0 a15 vdd gnd and
Xand16 nodeB0 nodeA0 P0 vdd gnd and

Xfadd1 a7 a8 0 C_fa1 S_fa1 vdd gnd fadder
Xfadd2 a11 a12 0 C_fa2 S_fa2 vdd gnd fadder
Xfadd3 a14 a15 0 C_fa3 P1 vdd gnd fadder
Xfadd4 C_fa1 a5 a6 C_fa4 S_fa4 vdd gnd fadder
Xfadd5 a2 a3 C_fa4 C_fa5 S_fa5 vdd gnd fadder
Xfadd6 a10 C_fa2 S_fa1 C_fa6 S_fa6 vdd gnd fadder
Xfadd7 a13 S_fa2 C_fa3 C_fa7 P2 vdd gnd fadder
Xfadd8 a9 C_fa7 S_fa6 C_fa8 P3 vdd gnd fadder
Xfadd9 a4 S_fa4 C_fa6 C_fa9 S_fa9 vdd gnd fadder
Xfadd10 0 C_fa8 S_fa9 C_fa10 P4 vdd gnd fadder
Xfadd11 C_fa10 C_fa9 S_fa5 C_fa11 P5 vdd gnd fadder
Xfadd12 C_fa11 C_fa5 a1 C5 P6 vdd gnd fadder

.tran 0.1p {1*6000p}

.control
run
plot V(nodeA3) V(nodeA2)+2 V(nodeA1)+4 V(nodeA0)+6
plot V(nodeB3) V(nodeB2)+2 V(nodeB1)+4 V(nodeB0)+6
plot V(P0) V(P1)+2 V(P2)+4 V(P3)+6 V(P4)+8 V(P5)+10 V(P6)+12 V(C5)+14
.endc
```

We provide an input of A = 1111 and B = 1111, and we observe the following results:

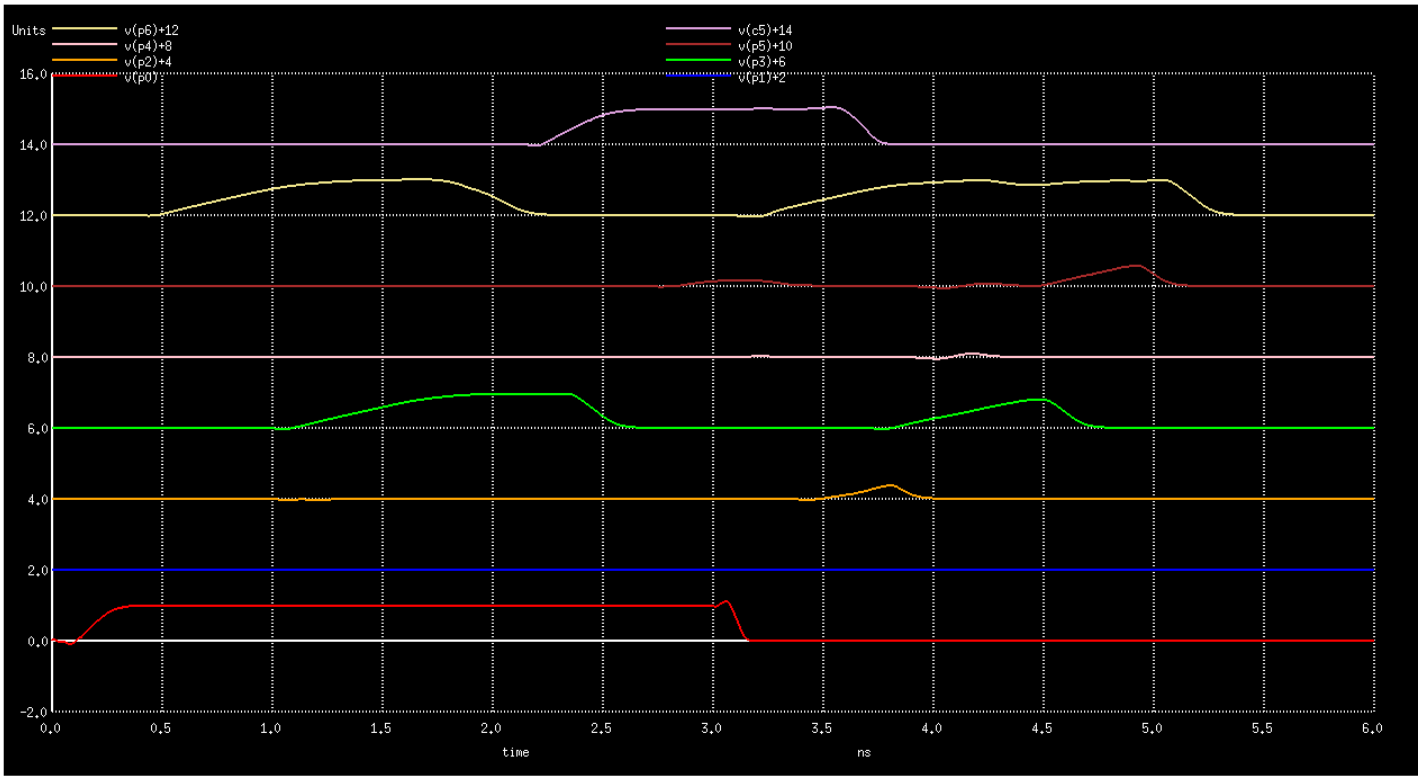
The plot of Input A:



The plot of Input B:



The output plot:



We measure the worst-case propagation delay for some 50 sample inputs and printed it out in a text file named "delay_output.txt". The worst propagation delay is observed between any one input bit and the most significant non-zero bit. Thie 4x4 multiplier is the cascade of multiple full adders. The delay appears in a signal where it passes through a gate. Therefore, the output that is obtained after the input bit has gone through most no. of gates will have the highest propagation delay.

The highest propagation delay observed for 25 sample inputs is: 2.84128E-09.

Now for the leakage power calculation, we give dc supply to all the inputs. We measure the power consumed by all the input sources (VinA3, VinA2, etc.) and the supply source (Vdd).

Power Consumption:- $P = VinA3 \cdot InA3 + VinA2 \cdot InA2 + VinA1 \cdot InA1 + VinA0 \cdot InA0 +$

$$VinB3 \cdot InB3 + VinB2 \cdot InB2 + VinB1 \cdot InB1 + VinB0 \cdot InB0 + Vdd \cdot Id$$

A python script named "main.py" has been written to print leakage power consumption for all the input combinations. These values of power Consumption are printed in a text file named "power_output.txt". We can run the following command to get the leakage power:

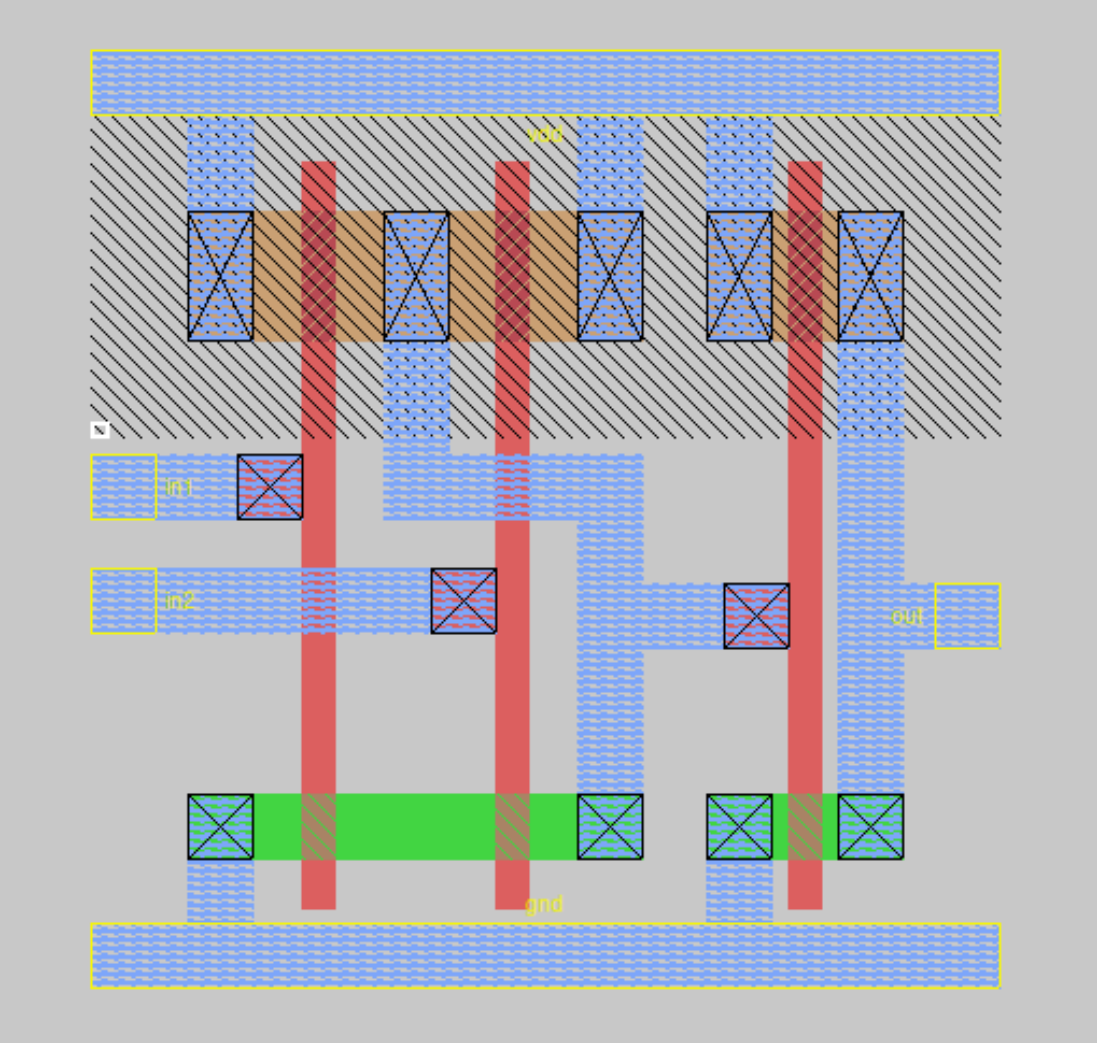
"python3 main.py"

MAGIC:

The verification of the subcircuits used for making the 4x4 multiplier:

AND GATE:

The magic layout for AND gate:



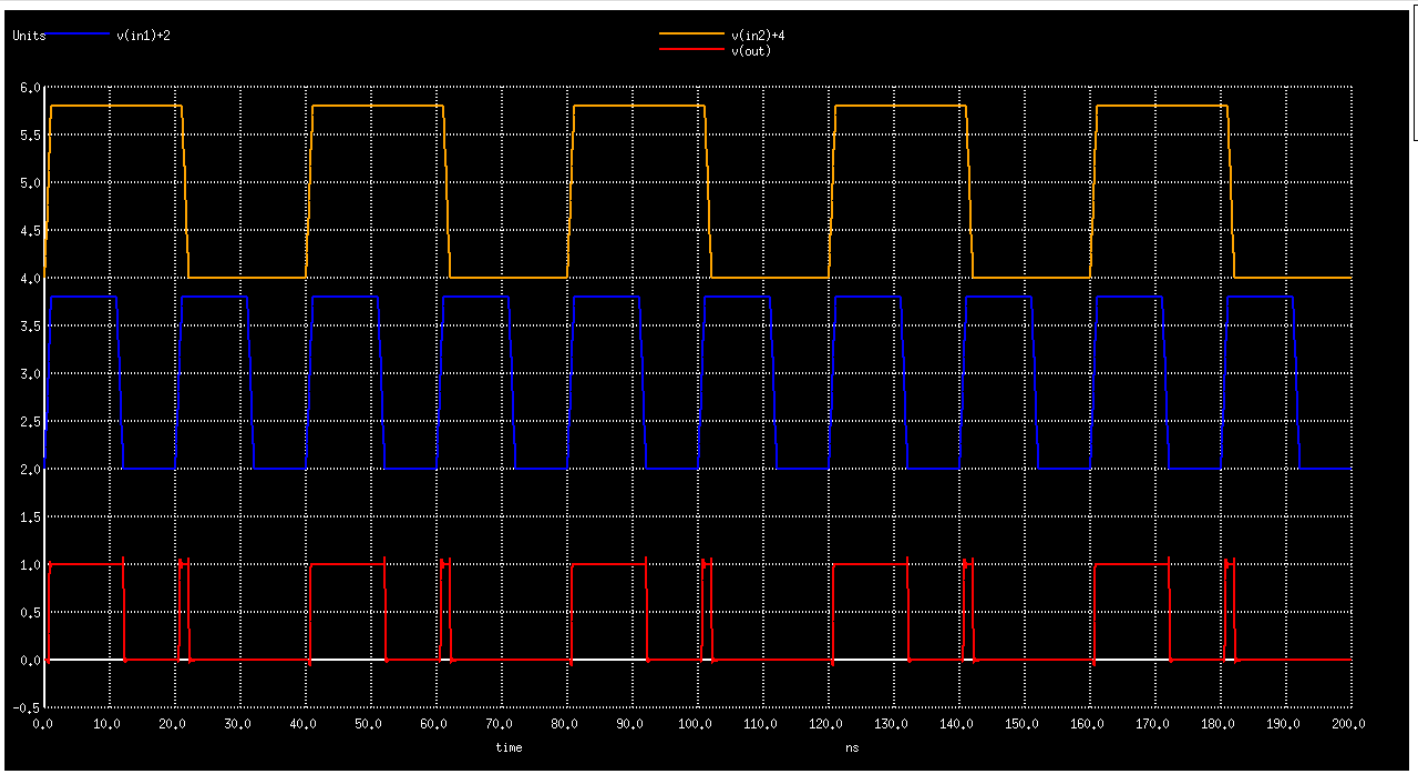
The spice extracted from this layout:

Magic > and.spice

```
1   .INCLUDE TSMC_180nm.txt
2   * SPICE3 file created from and.ext - technology: scmos
3   .option scale=0.09u
4   .PARAM pvdd = 1
5   .global gnd vdd
6
7   VDS vdd 0 dc='pvdd'
8   GRD gnd 0 dc=0
9
10  Vin1 in1 gnd pulse 0 1.8 0ns 1ns 1ns 10ns 20ns
11  Vin2 in2 gnd pulse 0 1.8 0ns 1ns 1ns 20ns 40ns
12
13  M1000 a_15_6# in2 a_15_n26# Gnd CMOSN w=4 l=2
14  +  ad=28 pd=22 as=40 ps=28
15  M1001 a_15_6# in1 vdd w_0_0# CMOSP w=8 l=2
16  +  ad=80 pd=36 as=152 ps=86
17  M1002 out a_15_6# gnd Gnd CMOSN w=4 l=2
18  +  ad=20 pd=18 as=48 ps=40
19  M1003 out a_15_6# vdd w_0_0# CMOSP w=8 l=2
20  +  ad=40 pd=26 as=0 ps=0
21  M1004 vdd in2 a_15_6# w_0_0# CMOSP w=8 l=2
22  +  ad=0 pd=0 as=0 ps=0
23  M1005 a_15_n26# in1 gnd Gnd CMOSN w=4 l=2
24  +  ad=0 pd=0 as=0 ps=0
```

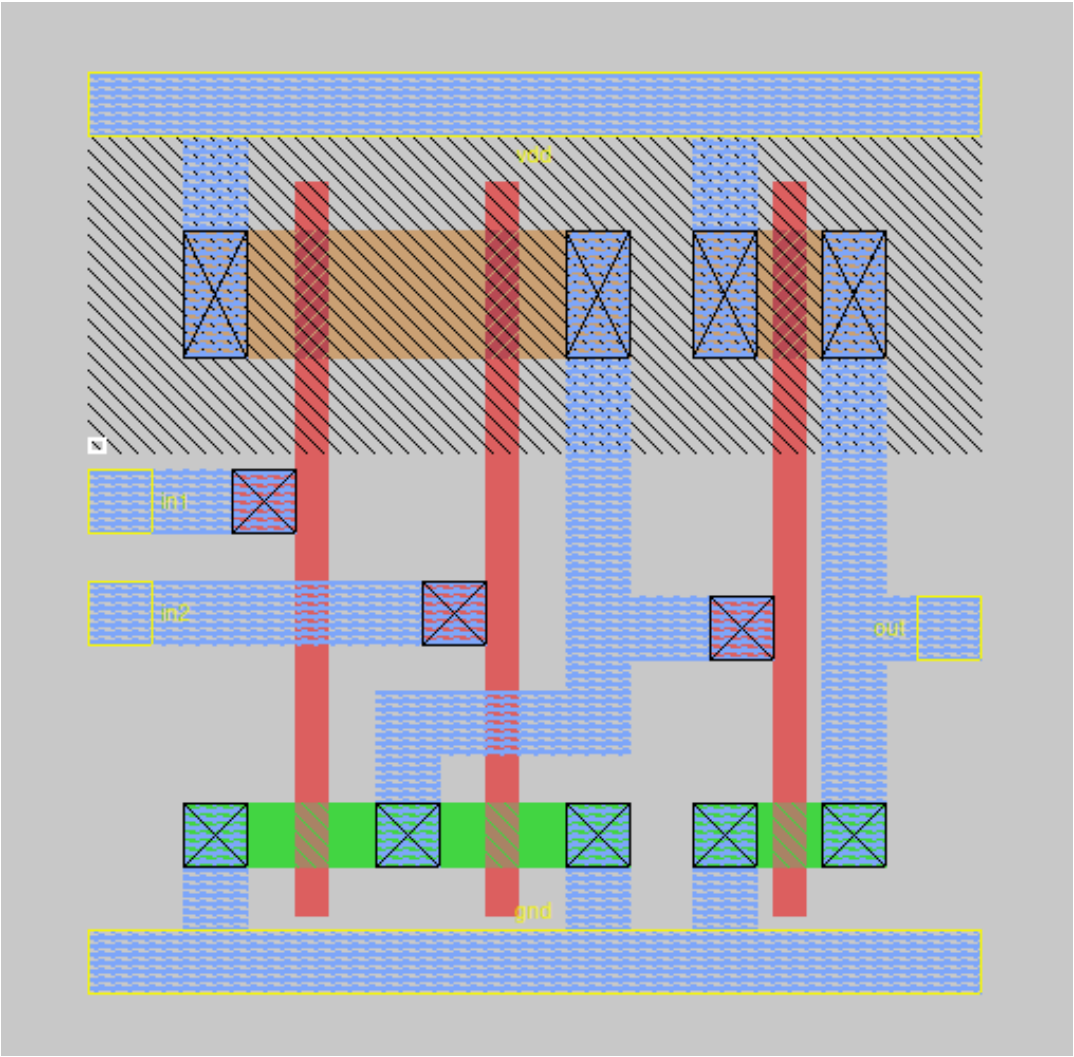
```
Magic > ≡ and.spice
25   C0 out vdd 0.11fF
26   C1 gnd a_15_6# 0.08fF
27   C2 w_0_0# vdd 0.14fF
28   C3 in1 a_15_6# 0.03fF
29   C4 in1 vdd 0.02fF
30   C5 in2 a_15_6# 0.21fF
31   C6 out w_0_0# 0.03fF
32   C7 out gnd 0.08fF
33   C8 w_0_0# in1 0.06fF
34   C9 w_0_0# in2 0.06fF
35   C10 in1 in2 0.27fF
36   C11 a_15_6# vdd 0.05fF
37   C12 out a_15_6# 0.05fF
38   C13 w_0_0# a_15_6# 0.09fF
39   C14 gnd Gnd 0.23fF
40   C15 out Gnd 0.10fF
41   C16 vdd Gnd 0.13fF
42   C17 a_15_6# Gnd 0.32fF
43   C18 in2 Gnd 0.26fF
44   C19 in1 Gnd 0.23fF
45   C20 w_0_0# Gnd 1.12fF
46
47   .tran 0.1n 200n
48
49   .control
50   run
51   plot v(out) v(in1)+2 v(in2)+4
52   .endc
53
54   .end
```

The output from the above spice file:



OR GATE:

The magic layout for OR gate:



The spice extracted from this layout:

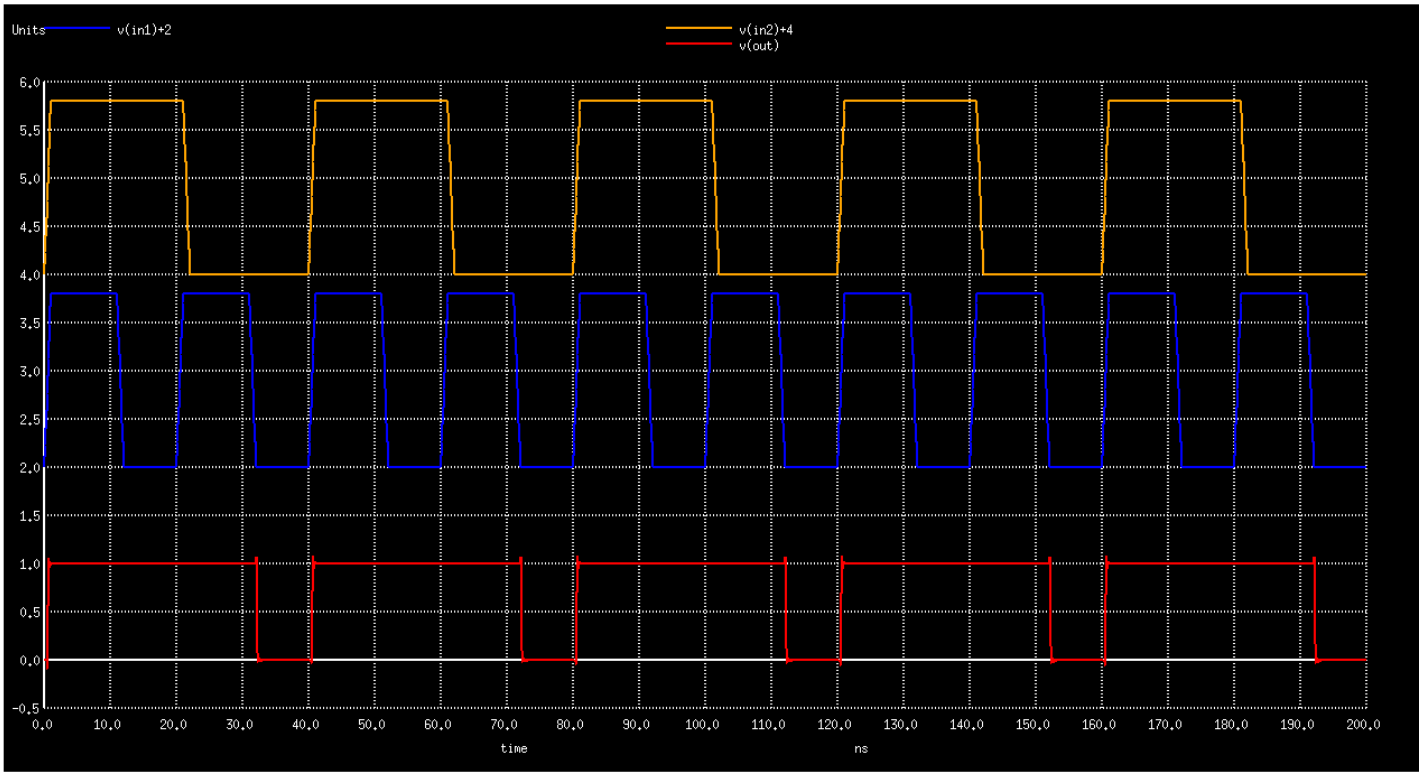
Magic > or.spice

```
.INCLUDE TSMC_180nm.txt
* SPICE3 file created from or.ext - technology: scmos

.option scale=0.09u
.PARAM pvdd = 1
.global gnd vdd

VDS vdd 0 dc='pvdd'
GRD gnd 0 dc=0

Vin1 in1 gnd pulse 0 1.8 0ns 1ns 1ns 10ns 20ns
Vin2 in2 gnd pulse 0 1.8 0ns 1ns 1ns 20ns 40ns

M1000 gnd in2 a_15_n26# Gnd CMOSN w=4 l=2
+  ad=76 pd=62 as=40 ps=28
M1001 a_15_6# in1 vdd w_0_0# CMOSP w=8 l=2
+  ad=80 pd=36 as=96 ps=56
M1002 out a_15_n26# gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1003 out a_15_n26# vdd w_0_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1004 a_15_n26# in2 a_15_6# w_0_0# CMOSP w=8 l=2
+  ad=56 pd=30 as=0 ps=0
M1005 a_15_n26# in1 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
```

Magic > or.spice

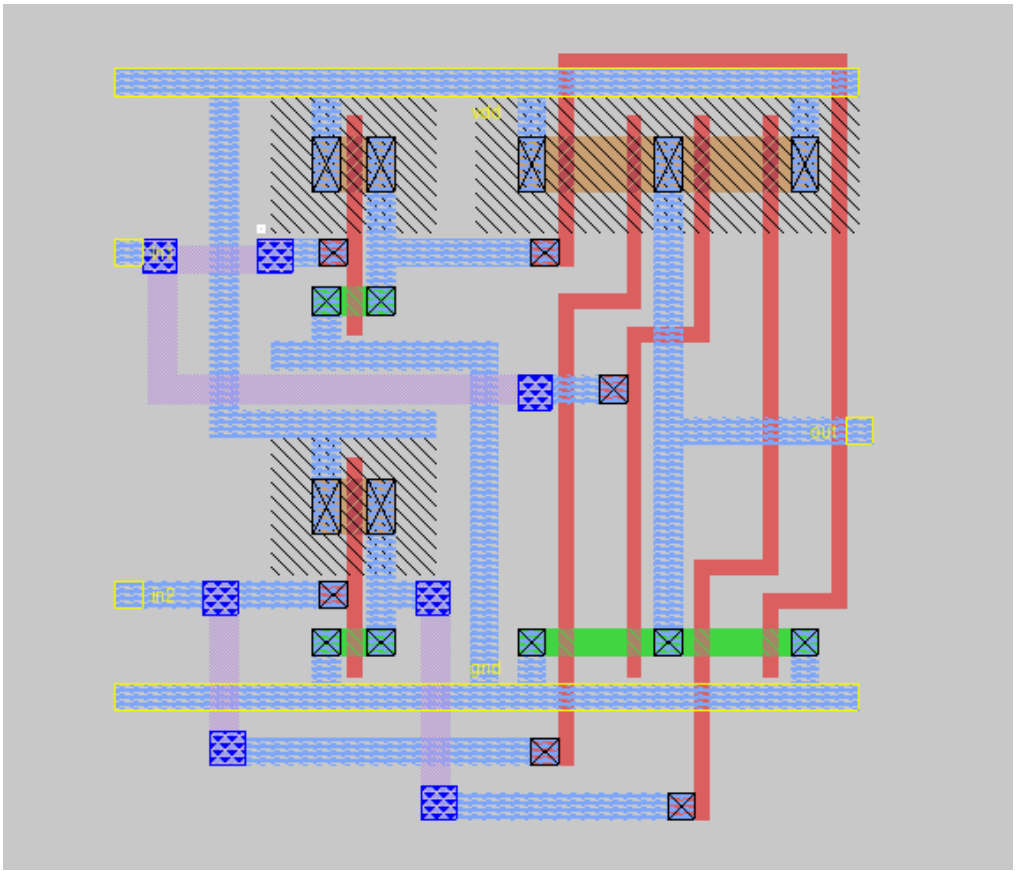
```
C0 a_15_n26# w_0_0# 0.10fF
C1 in1 vdd 0.02fF
C2 in2 a_15_n26# 0.21fF
C3 vdd w_0_0# 0.11fF
C4 a_15_n26# vdd 0.11fF
C5 out gnd 0.08fF
C6 w_0_0# out 0.03fF
C7 a_15_n26# out 0.05fF
C8 in1 w_0_0# 0.06fF
C9 in1 in2 0.27fF
C10 a_15_n26# gnd 0.10fF
C11 vdd out 0.11fF
C12 in2 w_0_0# 0.06fF
C13 gnd Gnd 0.24fF
C14 out Gnd 0.10fF
C15 vdd Gnd 0.13fF
C16 a_15_n26# Gnd 0.32fF
C17 in2 Gnd 0.26fF
C18 in1 Gnd 0.23fF
C19 w_0_0# Gnd 1.12fF

.tran 0.1n 200n

.control
run
plot v(out) v(in1)+2 v(in2)+4
.endc

.end
```

The output from the above spice file:



XOR GATE:

The magic layout for XOR gate:



The spice extracted from this layout:

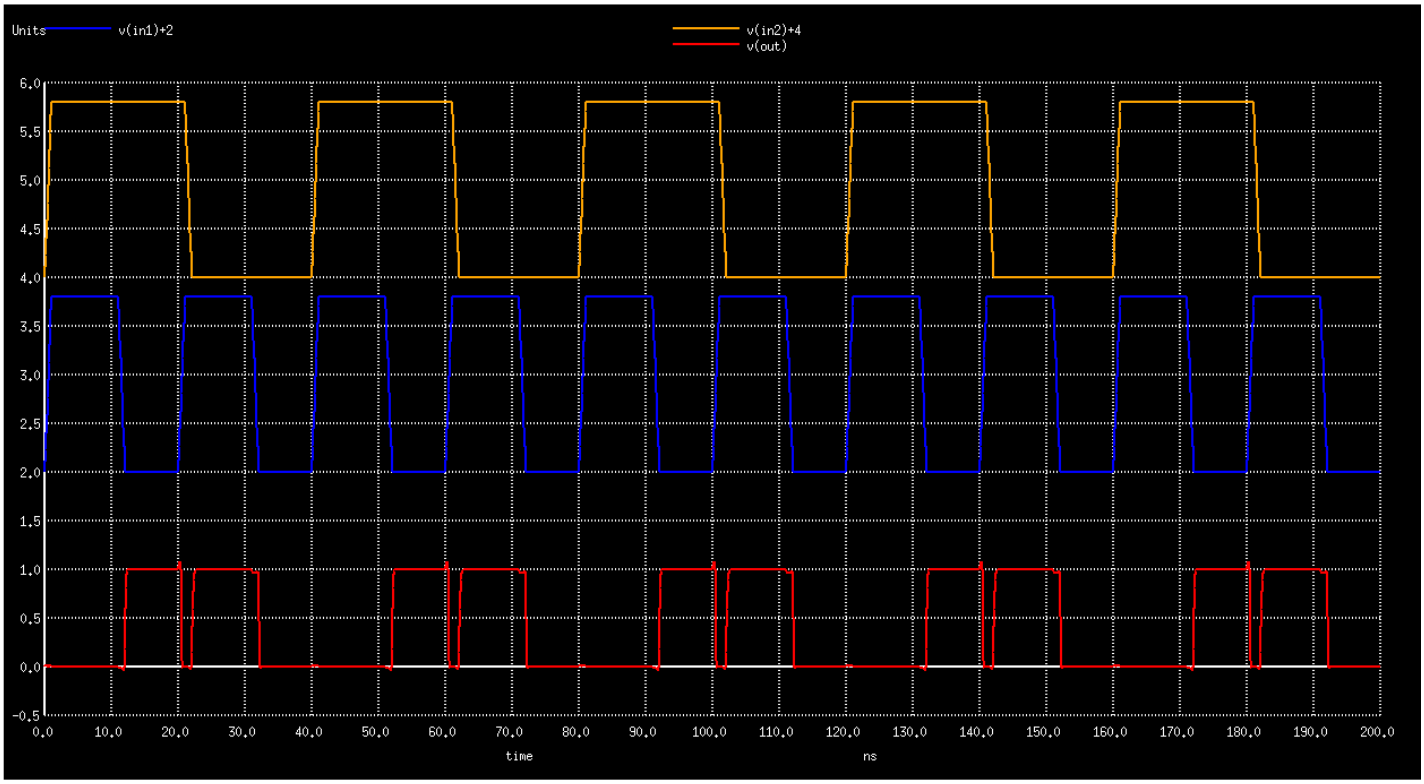
Magic > xor.spice

```
.INCLUDE TSMC_180nm.txt
* SPICE3 file created from xor.ext - technology: scmos
.option scale=0.09u
.PARAM pvdd = 1
.global gnd vdd

VDS vdd 0 dc='pvdd'
GRD gnd 0 dc=0

Vin1 in1 gnd pulse 0 1.8 0ns 1ns 1ns 10ns 20ns
Vin2 in2 gnd pulse 0 1.8 0ns 1ns 1ns 20ns 40ns

M1000 a_15_n62# in2 vdd w_2_n50# CMOSP w=8 l=2
+  ad=40 pd=26 as=176 ps=108
M1001 gnd a_15_n12# a_66_n62# Gnd CMOSN w=4 l=2
+  ad=88 pd=76 as=32 ps=24
M1002 a_46_6# a_15_n12# vdd w_32_0# CMOSP w=8 l=2
+  ad=64 pd=32 as=0 ps=0
M1003 a_15_n12# in1 vdd w_2_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1004 a_15_n12# in1 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1005 a_66_n62# a_15_n62# out Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1006 vdd a_15_n62# a_66_6# w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
M1007 out in1 a_46_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1008 a_46_n62# in2 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1009 a_15_n62# in2 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1010 a_66_6# in1 out w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
M1011 out in2 a_46_6# w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
```

Magic > xor.spice

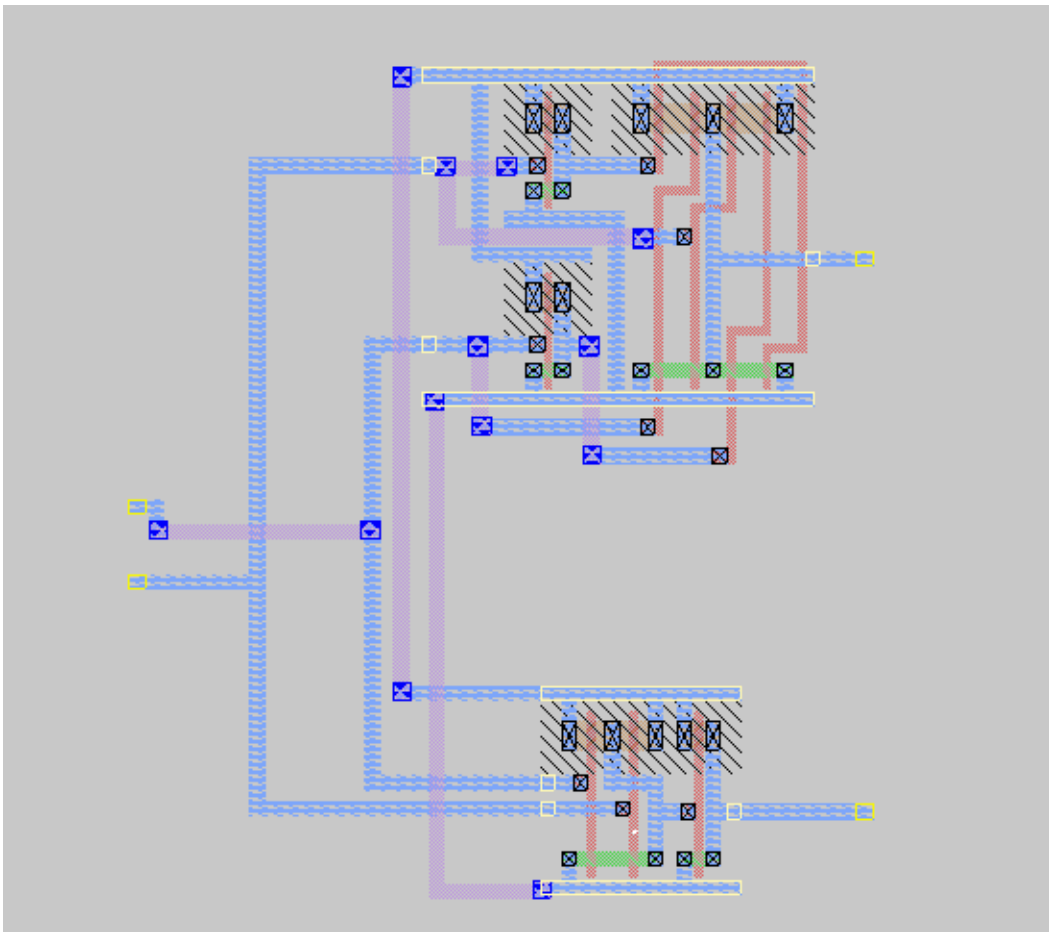
```
C0 a_15_n12# gnd 0.08fF
C1 w_2_0# a_15_n12# 0.03fF
C2 vdd out 0.03fF
C3 a_15_n62# out 0.08fF
C4 in1 gnd 0.21fF
C5 w_2_0# in1 0.06fF
C6 w_32_0# a_15_n12# 0.19fF
C7 in2 gnd 0.76fF
C8 w_32_0# in1 0.06fF
C9 vdd gnd 0.23fF
C10 a_15_n62# gnd 0.31fF
C11 w_2_0# vdd 0.05fF
C12 w_32_0# in2 0.06fF
C13 w_32_0# vdd 0.11fF
C14 a_15_n12# in1 0.06fF
C15 w_2_n50# in2 0.06fF
C16 w_32_0# a_15_n62# 0.06fF
C17 out gnd 0.04fF
C18 w_2_n50# vdd 0.05fF
C19 w_2_n50# a_15_n62# 0.03fF
C20 a_15_n12# in2 0.02fF
C21 a_15_n12# vdd 0.74fF
C22 w_32_0# out 0.02fF
C23 a_15_n12# a_15_n62# 0.02fF
C24 in1 in2 0.11fF
C25 in1 vdd 0.30fF
C26 a_15_n12# out 0.08fF
C27 in2 vdd 0.02fF
C28 in2 a_15_n62# 0.36fF
C29 in1 out 0.12fF
C30 a_15_n62# vdd 0.11fF
C31 gnd Gnd 0.64fF
C32 out Gnd 0.23fF
C33 vdd Gnd 0.17fF
C34 a_15_n62# Gnd 0.26fF
C35 in2 Gnd 0.39fF
C36 in1 Gnd 1.62fF
C37 a_15_n12# Gnd 0.17fF
C38 w_2_n50# Gnd 0.48fF
C39 w_32_0# Gnd 1.12fF
C40 w_2_0# Gnd 0.48fF

.tran 0.1n 200n

.control
run
plot v(out) v(in1)+2 v(in2)+4
.endc

.end
```

The output from the above spice file:



HALF ADDER:

The magic layout for half adder:



The spice extracted from this layout:

Magic > hadd(copy).spice

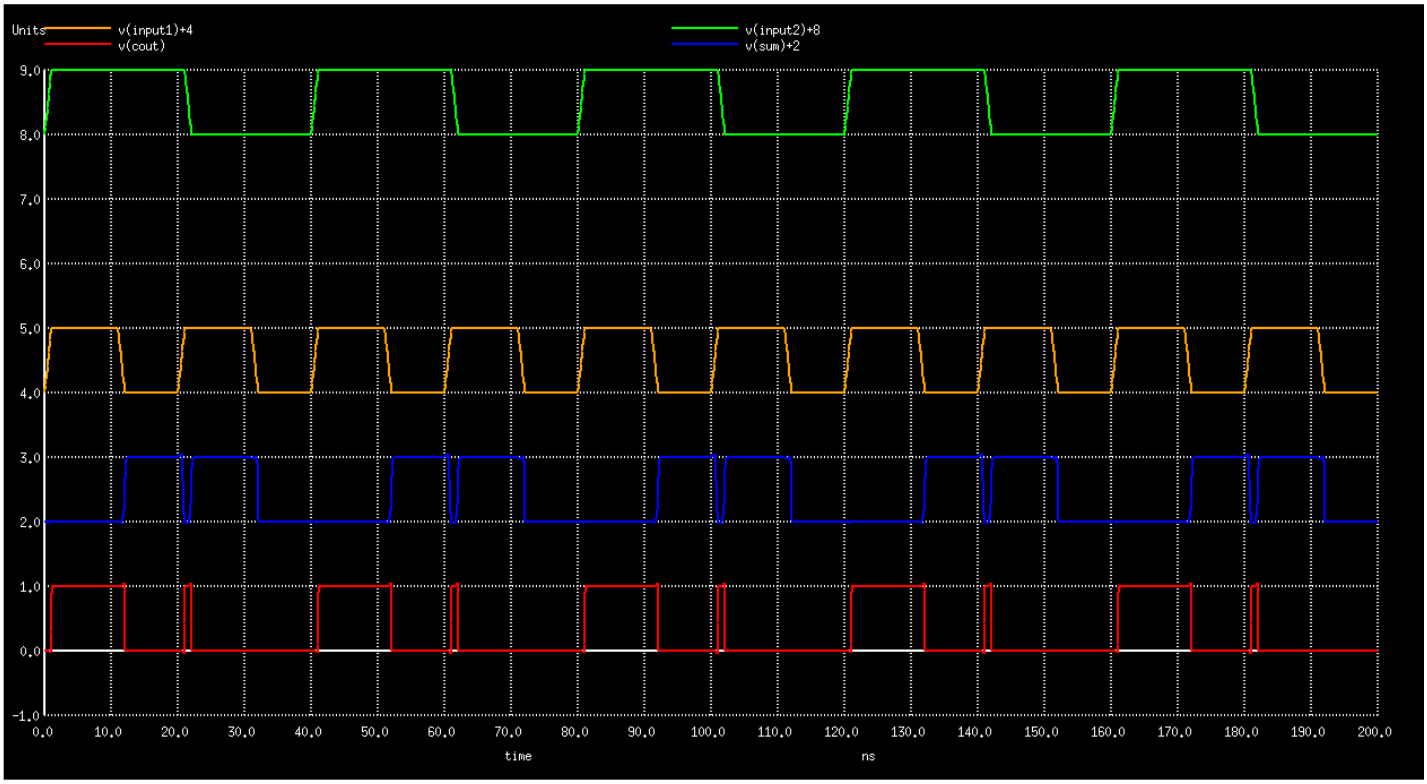
```
.INCLUDE TSMC_180nm.txt
* SPICE3 file created from hadd.ext - technology: scmos
.option scale=0.09u
.PARAM pvdd = 1
.global gnd vdd

VDS vdd 0 dc='pvdd'
GRD gnd 0 dc=0

Vin1 input1 gnd pulse 0 1 0ns 1ns 1ns 10ns 20ns
Vin2 input2 gnd pulse 0 1 0ns 1ns 1ns 20ns 40ns

M1000 and_0/a_15_6# input2 and_0/a_15_n26# Gnd CMOSN w=4 l=2
+  ad=28 pd=22 as=40 ps=28
M1001 and_0/a_15_6# input1 vdd and_0/w_0_0# CMOSP w=8 l=2
+  ad=80 pd=36 as=328 ps=194
M1002 cout and_0/a_15_6# gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=136 ps=116
M1003 cout and_0/a_15_6# vdd and_0/w_0_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1004 vdd input2 and_0/a_15_6# and_0/w_0_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
M1005 and_0/a_15_n26# input1 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1006 xor_0/a_15_n62# input1 vdd xor_0/w_2_n50# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1007 gnd xor_0/a_15_n12# xor_0/a_66_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1008 xor_0/a_46_6# xor_0/a_15_n12# vdd xor_0/w_32_0# CMOSP w=8 l=2
+  ad=64 pd=32 as=0 ps=0
M1009 xor_0/a_15_n12# input2 vdd xor_0/w_2_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1010 xor_0/a_15_n12# input2 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1011 xor_0/a_66_n62# xor_0/a_15_n62# sum Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1012 vdd xor_0/a_15_n62# xor_0/a_66_6# xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
```

Magic > hadd(copy).spice

```
M1013 sum input2 xor_0/a_46_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1014 xor_0/a_46_n62# input1 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1015 xor_0/a_15_n62# input1 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1016 xor_0/a_66_6# input2 sum xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
M1017 sum input1 xor_0/a_46_6# xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
C0 gnd xor_0/a_15_n12# 0.08fF
C1 cout vdd 0.11fF
C2 gnd vdd 0.47fF
C3 input2 and_0/a_15_6# 0.21fF
C4 gnd sum 0.04fF
C5 cout and_0/w_0_0# 0.03fF
C6 gnd input1 0.85fF
C7 gnd xor_0/a_15_n62# 0.31fF
C8 input2 xor_0/a_15_n12# 0.06fF
C9 and_0/a_15_6# vdd 0.05fF
C10 input2 vdd 0.39fF
C11 input2 sum 0.12fF
C12 and_0/a_15_6# input1 0.03fF
C13 and_0/a_15_6# and_0/w_0_0# 0.09fF
C14 xor_0/a_15_n12# vdd 0.74fF
C15 input2 input1 1.14fF
C16 input2 and_0/w_0_0# 0.06fF
C17 xor_0/a_15_n12# sum 0.08fF
C18 sum vdd 0.03fF
C19 xor_0/a_15_n12# input1 0.02fF
C20 input2 xor_0/w_32_0# 0.06fF
C21 xor_0/a_15_n12# xor_0/a_15_n62# 0.02fF
C22 vdd input1 0.20fF
C23 and_0/w_0_0# vdd 0.14fF
C24 xor_0/a_15_n62# vdd 0.11fF
C25 xor_0/a_15_n12# xor_0/w_32_0# 0.19fF
C26 input2 xor_0/w_2_0# 0.06fF
C27 sum xor_0/a_15_n62# 0.08fF
```

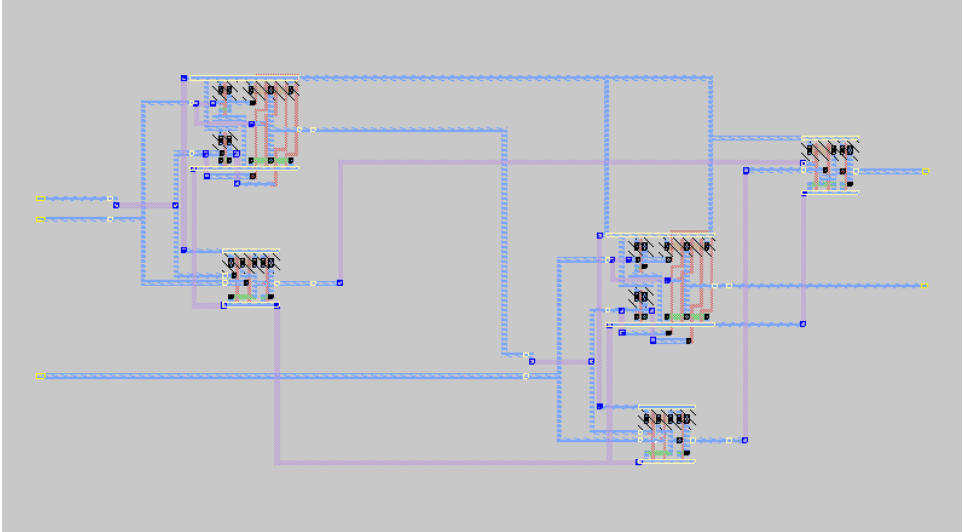
```
Magic > hadd(copy).spice
 77   C28 and_0/w_0_0# input1 0.06fF
 78   C29 xor_0/w_32_0# vdd 0.11fF
 79   C30 xor_0/a_15_n62# input1 0.36fF
 80   C31 xor_0/a_15_n12# xor_0/w_2_0# 0.03fF
 81   C32 sum xor_0/w_32_0# 0.02fF
 82   C33 gnd cout 0.08fF
 83   C34 xor_0/w_32_0# input1 0.06fF
 84   C35 xor_0/w_2_0# vdd 0.05fF
 85   C36 xor_0/w_2_n50# vdd 0.05fF
 86   C37 xor_0/a_15_n62# xor_0/w_32_0# 0.06fF
 87   C38 xor_0/w_2_n50# input1 0.06fF
 88   C39 cout and_0/a_15_6# 0.05fF
 89   C40 gnd and_0/a_15_6# 0.08fF
 90   C41 xor_0/a_15_n62# xor_0/w_2_n50# 0.03fF
 91   C42 gnd input2 0.29fF
 92   C43 gnd Gnd 1.31fF
 93   C44 sum Gnd 0.30fF
 94   C45 vdd Gnd 0.82fF
 95   C46 xor_0/a_15_n62# Gnd 0.26fF
 96   C47 input1 Gnd 0.66fF
 97   C48 input2 Gnd 2.42fF
 98   C49 xor_0/a_15_n12# Gnd 0.17fF
 99   C50 xor_0/w_2_n50# Gnd 0.48fF
100   C51 xor_0/w_32_0# Gnd 1.12fF
101   C52 xor_0/w_2_0# Gnd 0.48fF
102   C53 cout Gnd 0.23fF
103   C54 and_0/a_15_6# Gnd 0.32fF
104   C55 and_0/w_0_0# Gnd 1.12fF
105
106   .tran 0.1n 200n
107
108   .control
109   run
110   plot v(cout) v(sum)+2 v(input1)+4 v(input2)+8
111   .endc
112
113   .end
```

The output from the above spice file:



FULL ADDER:

The magic layout for full adder:



The spice extracted from this layout:

Magic > fadd.spice

```
.INCLUDE TSMC_180nm.txt
* SPICE3 file created from fadd.ext - technology: scmos
.option scale=0.09u
.PARAM pvdd = 1
.global gnd vdd

VDS vdd 0 dc='pvdd'
GRD gnd 0 dc=0
Vin1 in1 gnd pulse 0 1.8 0ns 1ns 1ns 10ns 20ns
Vin2 in2 gnd pulse 0 1.8 0ns 1ns 1ns 20ns 40ns
Vin3 in3 gnd pulse 0 1.8 0ns 1ns 1ns 40ns 80ns

M1000 hadd_0/and_0/a_15_6# in2 hadd_0/and_0/a_15_n26# Gnd CMOSN w=4 l=2
+  ad=28 pd=22 as=40 ps=28
M1001 hadd_0/and_0/a_15_6# in1 vdd hadd_0/and_0/w_0_0# CMOSP w=8 l=2
+  ad=80 pd=36 as=752 ps=444
M1002 or_0/in1 hadd_0/and_0/a_15_6# gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=348 ps=294
M1003 or_0/in1 hadd_0/and_0/a_15_6# vdd hadd_0/and_0/w_0_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1004 vdd in2 hadd_0/and_0/a_15_6# hadd_0/and_0/w_0_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
M1005 hadd_0/and_0/a_15_n26# in1 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1006 hadd_0/xor_0/a_15_n62# in1 vdd hadd_0/xor_0/w_2_n50# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1007 gnd hadd_0/xor_0/a_15_n12# hadd_0/xor_0/a_66_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1008 hadd_0/xor_0/a_46_6# hadd_0/xor_0/a_15_n12# vdd hadd_0/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=64 pd=32 as=0 ps=0
M1009 hadd_0/xor_0/a_15_n12# in2 vdd hadd_0/xor_0/w_2_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1010 hadd_0/xor_0/a_15_n12# in2 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1011 hadd_0/xor_0/a_66_n62# hadd_0/xor_0/a_15_n62# hadd_0/sum Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1012 vdd hadd_0/xor_0/a_15_n62# hadd_0/xor_0/a_66_6# hadd_0/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
```

Magic > fadd.spice

```
M1013 hadd_0/sum in2 hadd_0/xor_0/a_46_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1014 hadd_0/xor_0/a_46_n62# in1 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1015 hadd_0/xor_0/a_15_n62# in1 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1016 hadd_0/xor_0/a_66_6# in2 hadd_0/sum hadd_0/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
M1017 hadd_0/sum in1 hadd_0/xor_0/a_46_6# hadd_0/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
M1018 hadd_1/and_0/a_15_6# in3 hadd_1/and_0/a_15_n26# Gnd CMOSN w=4 l=2
+  ad=28 pd=22 as=40 ps=28
M1019 hadd_1/and_0/a_15_6# hadd_0/sum vdd hadd_1/and_0/w_0_0# CMOSP w=8 l=2
+  ad=80 pd=36 as=0 ps=0
M1020 or_0/in2 hadd_1/and_0/a_15_6# gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1021 or_0/in2 hadd_1/and_0/a_15_6# vdd hadd_1/and_0/w_0_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1022 vdd in3 hadd_1/and_0/a_15_6# hadd_1/and_0/w_0_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
M1023 hadd_1/and_0/a_15_n26# hadd_0/sum gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1024 hadd_1/xor_0/a_15_n62# hadd_0/sum vdd hadd_1/xor_0/w_2_n50# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1025 gnd hadd_1/xor_0/a_15_n12# hadd_1/xor_0/a_66_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1026 hadd_1/xor_0/a_46_6# hadd_1/xor_0/a_15_n12# vdd hadd_1/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=64 pd=32 as=0 ps=0
M1027 hadd_1/xor_0/a_15_n12# in3 vdd hadd_1/xor_0/w_2_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1028 hadd_1/xor_0/a_15_n12# in3 gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1029 hadd_1/xor_0/a_66_n62# hadd_1/xor_0/a_15_n62# sum Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
M1030 vdd hadd_1/xor_0/a_15_n62# hadd_1/xor_0/a_66_6# hadd_1/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
M1031 sum in3 hadd_1/xor_0/a_46_n62# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=32 ps=24
```

Magic > fadd.spice

```
M1032 hadd_1/xor_0/a_46_n62# hadd_0/sum gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
M1033 hadd_1/xor_0/a_15_n62# hadd_0/sum gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1034 hadd_1/xor_0/a_66_6# in3 sum hadd_1/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=64 ps=32
M1035 sum hadd_0/sum hadd_1/xor_0/a_46_6# hadd_1/xor_0/w_32_0# CMOSP w=8 l=2
+  ad=0 pd=0 as=0 ps=0
M1036 gnd or_0/in2 or_0/a_15_n26# Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=40 ps=28
M1037 or_0/a_15_6# or_0/in1 vdd or_0/w_0_0# CMOSP w=8 l=2
+  ad=80 pd=36 as=0 ps=0
M1038 cout or_0/a_15_n26# gnd Gnd CMOSN w=4 l=2
+  ad=20 pd=18 as=0 ps=0
M1039 cout or_0/a_15_n26# vdd or_0/w_0_0# CMOSP w=8 l=2
+  ad=40 pd=26 as=0 ps=0
M1040 or_0/a_15_n26# or_0/in2 or_0/a_15_6# or_0/w_0_0# CMOSP w=8 l=2
+  ad=56 pd=30 as=0 ps=0
M1041 or_0/a_15_n26# or_0/in1 gnd Gnd CMOSN w=4 l=2
+  ad=0 pd=0 as=0 ps=0
C0 hadd_0/xor_0/a_15_n62# hadd_0/xor_0/w_2_n50# 0.03fF
C1 in1 hadd_0/xor_0/a_15_n12# 0.02fF
C2 in3 hadd_1/and_0/w_0_0# 0.06fF
C3 or_0/in2 hadd_1/and_0/a_15_6# 0.05fF
C4 vdd hadd_0/sum 0.22fF
C5 sum gnd 0.13fF
C6 hadd_1/xor_0/a_15_n12# hadd_1/xor_0/w_32_0# 0.19fF
C7 or_0/in2 or_0/w_0_0# 0.06fF
C8 sum hadd_1/xor_0/a_15_n12# 0.08fF
C9 hadd_0/and_0/a_15_6# hadd_0/and_0/w_0_0# 0.09fF
C10 in1 hadd_0/and_0/w_0_0# 0.06fF
C11 hadd_0/sum hadd_1/and_0/a_15_6# 0.03fF
C12 hadd_0/xor_0/a_15_n62# hadd_0/xor_0/w_32_0# 0.06fF
C13 in2 hadd_0/xor_0/a_15_n12# 0.06fF
C14 in1 hadd_0/xor_0/w_2_n50# 0.06fF
C15 gnd hadd_0/xor_0/a_15_n62# 0.31fF
C16 vdd hadd_1/and_0/a_15_6# 0.05fF
C17 or_0/a_15_n26# gnd 0.10fF
```

Magic > fadd.spice

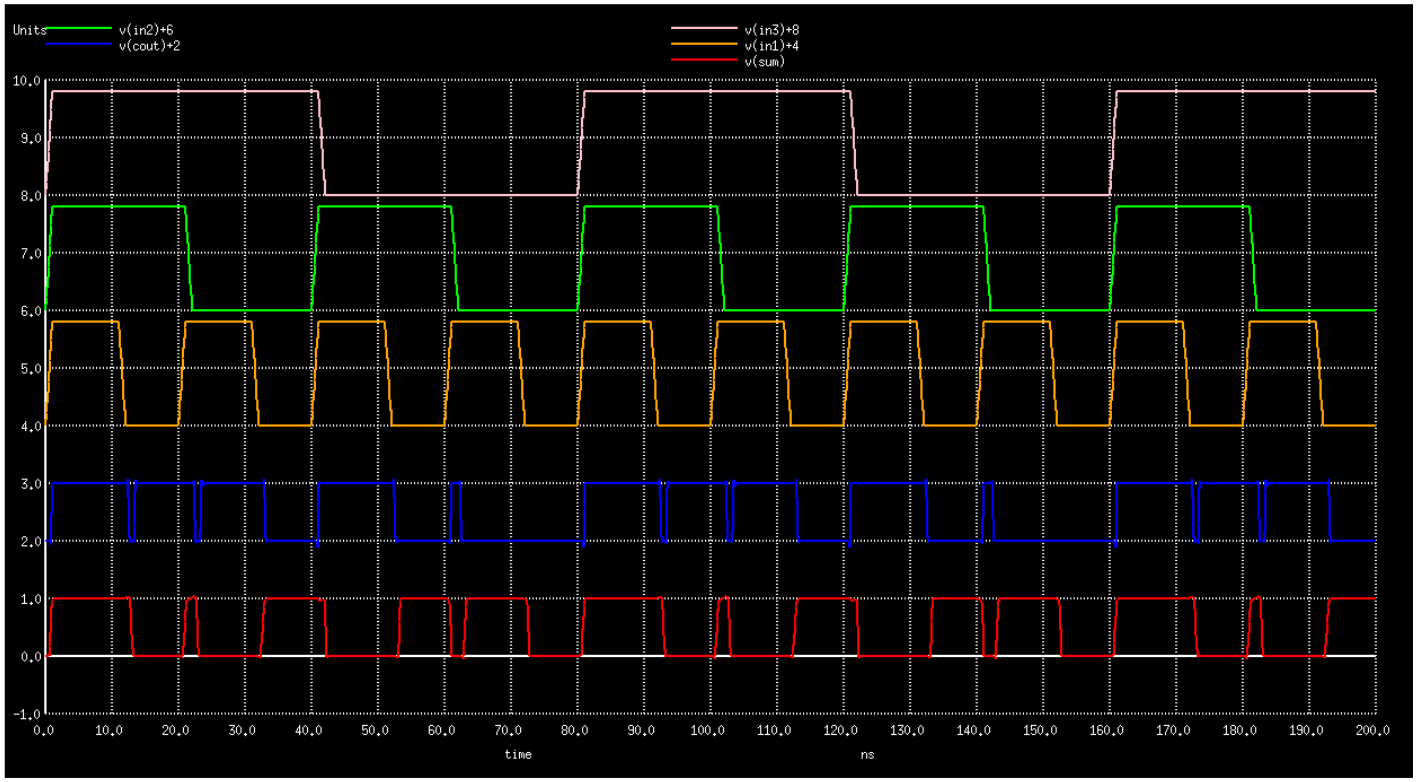
```
C18 or_0/in1 hadd_0/and_0/a_15_6# 0.05fF
C19 in3 hadd_1/xor_0/w_2_0# 0.06fF
C20 vdd or_0/w_0_0# 0.11fF
C21 or_0/in2 sum 0.09fF
C22 in2 hadd_0/and_0/w_0_0# 0.06fF
C23 in1 hadd_0/xor_0/w_32_0# 0.06fF
C24 gnd hadd_0/and_0/a_15_6# 0.08fF
C25 gnd in1 0.85fF
C26 or_0/in2 hadd_1/and_0/w_0_0# 0.03fF
C27 hadd_1/xor_0/w_32_0# hadd_0/sum 0.06fF
C28 hadd_1/xor_0/a_15_n12# hadd_1/xor_0/w_2_0# 0.03fF
C29 or_0/in2 or_0/a_15_n26# 0.21fF
C30 vdd hadd_1/xor_0/w_32_0# 0.11fF
C31 vdd sum 0.03fF
C32 hadd_0/sum hadd_1/and_0/w_0_0# 0.06fF
C33 in2 hadd_0/xor_0/w_32_0# 0.06fF
C34 hadd_0/sum hadd_0/xor_0/a_15_n62# 0.08fF
C35 gnd in2 0.29fF
C36 vdd hadd_1/and_0/w_0_0# 0.14fF
C37 vdd hadd_0/xor_0/a_15_n62# 0.11fF
C38 hadd_1/xor_0/a_15_n62# gnd 0.31fF
C39 vdd or_0/a_15_n26# 0.11fF
C40 cout gnd 0.08fF
C41 hadd_1/xor_0/a_15_n62# hadd_1/xor_0/a_15_n12# 0.02fF
C42 hadd_1/and_0/a_15_6# hadd_1/and_0/w_0_0# 0.09fF
C43 in2 hadd_0/xor_0/w_2_0# 0.06fF
C44 hadd_0/xor_0/a_15_n12# hadd_0/xor_0/w_32_0# 0.19fF
C45 gnd hadd_0/xor_0/a_15_n12# 0.08fF
C46 vdd hadd_0/and_0/a_15_6# 0.05fF
C47 or_0/in1 hadd_0/and_0/w_0_0# 0.03fF
C48 vdd in1 0.20fF
C49 or_0/w_0_0# or_0/a_15_n26# 0.10fF
C50 vdd hadd_1/xor_0/w_2_0# 0.05fF
C51 sum hadd_1/xor_0/w_32_0# 0.02fF
C52 hadd_1/xor_0/a_15_n62# hadd_1/xor_0/w_2_n50# 0.03fF
C53 hadd_0/xor_0/a_15_n12# hadd_0/xor_0/w_2_0# 0.03fF
C54 hadd_0/sum in2 0.12fF
C55 vdd in2 0.39fF
```

Magic > fadd.spice

```
C56 hadd_1/xor_0/a_15_n62# hadd_0/sum 0.36fF
C57 in3 gnd 0.38fF
C58 in3 hadd_1/xor_0/a_15_n12# 0.06fF
C59 or_0/in1 gnd 0.08fF
C60 vdd hadd_1/xor_0/a_15_n62# 0.11fF
C61 vdd cout 0.11fF
C62 hadd_0/sum hadd_0/xor_0/a_15_n12# 0.08fF
C63 vdd hadd_0/xor_0/a_15_n12# 0.74fF
C64 hadd_1/xor_0/a_15_n12# gnd 0.08fF
C65 cout or_0/w_0_0# 0.03fF
C66 or_0/in2 or_0/in1 0.31fF
C67 hadd_0/xor_0/a_15_n62# in1 0.36fF
C68 vdd hadd_0/and_0/w_0_0# 0.14fF
C69 vdd hadd_0/xor_0/w_2_n50# 0.05fF
C70 in3 hadd_0/sum 1.14fF
C71 or_0/in2 gnd 0.17fF
C72 or_0/in1 hadd_0/sum 0.09fF
C73 hadd_1/xor_0/a_15_n62# hadd_1/xor_0/w_32_0# 0.06fF
C74 vdd in3 0.39fF
C75 sum hadd_1/xor_0/a_15_n62# 0.08fF
C76 vdd or_0/in1 0.30fF
C77 in1 hadd_0/and_0/a_15_6# 0.03fF
C78 hadd_0/sum hadd_0/xor_0/w_32_0# 0.02fF
C79 in3 hadd_1/and_0/a_15_6# 0.21fF
C80 gnd hadd_0/sum 0.89fF
C81 vdd hadd_0/xor_0/w_32_0# 0.11fF
C82 hadd_1/xor_0/a_15_n12# hadd_0/sum 0.02fF
C83 vdd gnd 0.94fF
C84 vdd hadd_1/xor_0/a_15_n12# 0.76fF
C85 or_0/in1 or_0/w_0_0# 0.08fF
C86 cout or_0/a_15_n26# 0.05fF
C87 in2 hadd_0/and_0/a_15_6# 0.21fF
C88 gnd hadd_1/and_0/a_15_6# 0.08fF
C89 hadd_0/xor_0/a_15_n62# hadd_0/xor_0/a_15_n12# 0.02fF
C90 in1 in2 1.14fF
C91 vdd hadd_0/xor_0/w_2_0# 0.05fF
C92 hadd_1/xor_0/w_2_n50# hadd_0/sum 0.06fF
C93 in3 hadd_1/xor_0/w_32_0# 0.06fF
```

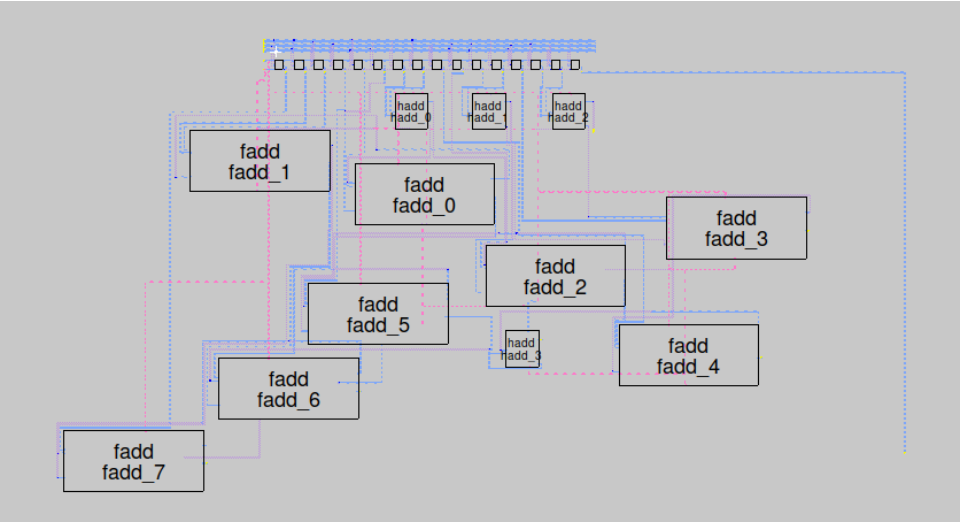
```
Magic > fadd.spice
C94 vdd hadd_1/xor_0/w_2_n50# 0.05fF
C95 sum in3 0.12fF
C96 or_0/in2 vdd 0.11fF
C97 cout Gnd 0.35fF
C98 vdd Gnd 1.96fF
C99 or_0/a_15_n26# Gnd 0.32fF
C100 or_0/in2 Gnd 0.73fF
C101 or_0/in1 Gnd 0.70fF
C102 or_0/w_0_0# Gnd 1.12fF
C103 sum Gnd 0.61fF
C104 hadd_1/xor_0/a_15_n62# Gnd 0.26fF
C105 in3 Gnd 2.45fF
C106 hadd_1/xor_0/a_15_n12# Gnd 0.17fF
C107 hadd_1/xor_0/w_2_n50# Gnd 0.48fF
C108 hadd_1/xor_0/w_32_0# Gnd 1.12fF
C109 hadd_1/xor_0/w_2_0# Gnd 0.48fF
C110 hadd_1/and_0/a_15_6# Gnd 0.32fF
C111 hadd_1/and_0/w_0_0# Gnd 1.12fF
C112 gnd Gnd 3.79fF
C113 hadd_0/sum Gnd 1.31fF
C114 hadd_0/xor_0/a_15_n62# Gnd 0.26fF
C115 in1 Gnd 0.91fF
C116 in2 Gnd 2.60fF
C117 hadd_0/xor_0/a_15_n12# Gnd 0.17fF
C118 hadd_0/xor_0/w_2_n50# Gnd 0.48fF
C119 hadd_0/xor_0/w_32_0# Gnd 1.12fF
C120 hadd_0/xor_0/w_2_0# Gnd 0.48fF
C121 hadd_0/and_0/a_15_6# Gnd 0.32fF
C122 hadd_0/and_0/w_0_0# Gnd 1.12fF

.tran 0.1n 200n

.control
run
plot v(sum) v(cout)+2 v(in1)+4 v(in2)+6 v(in3)+8
.endc
```

The output from the above spice file:



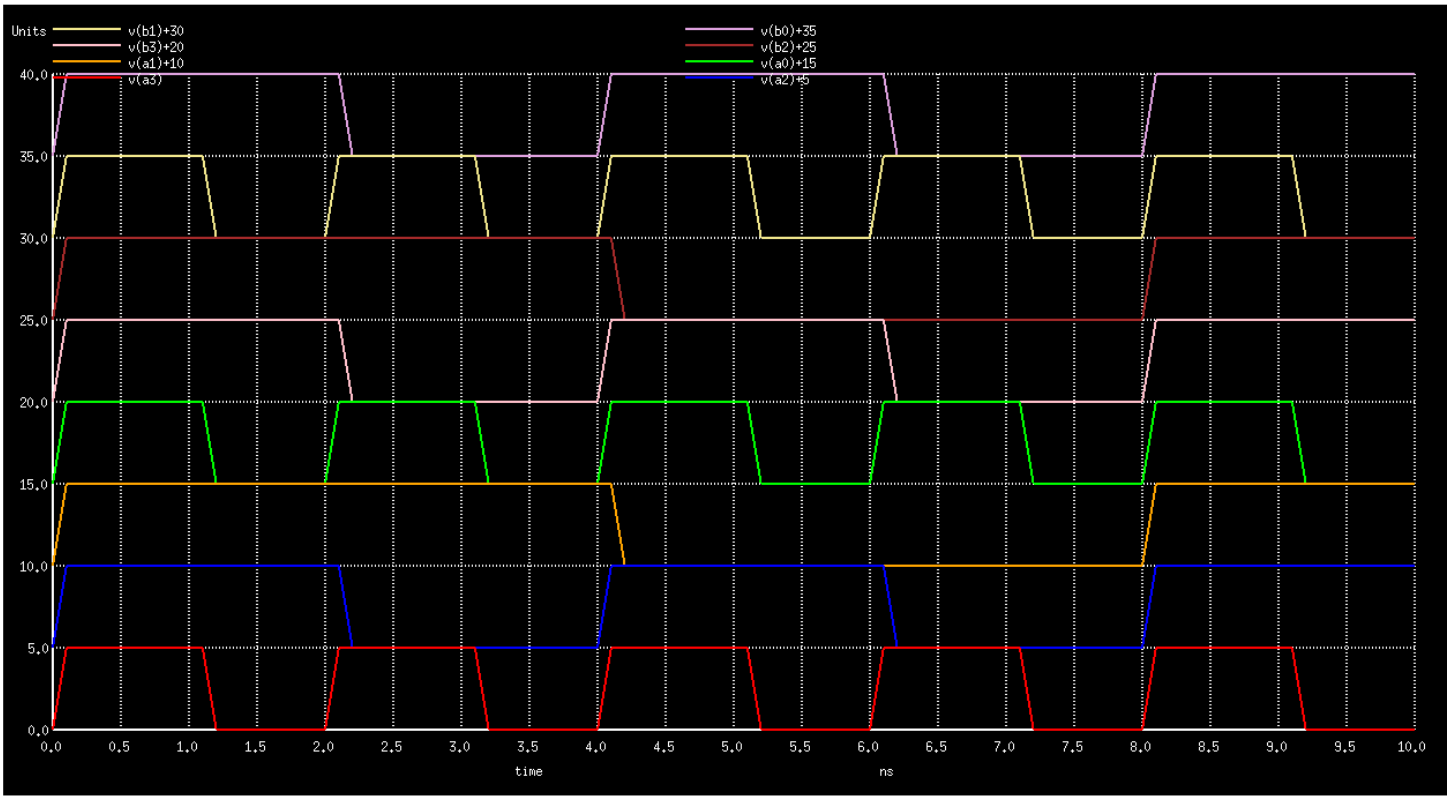
4x4 MULTIPLIER:

The magic layout for 4x4 multiplier:

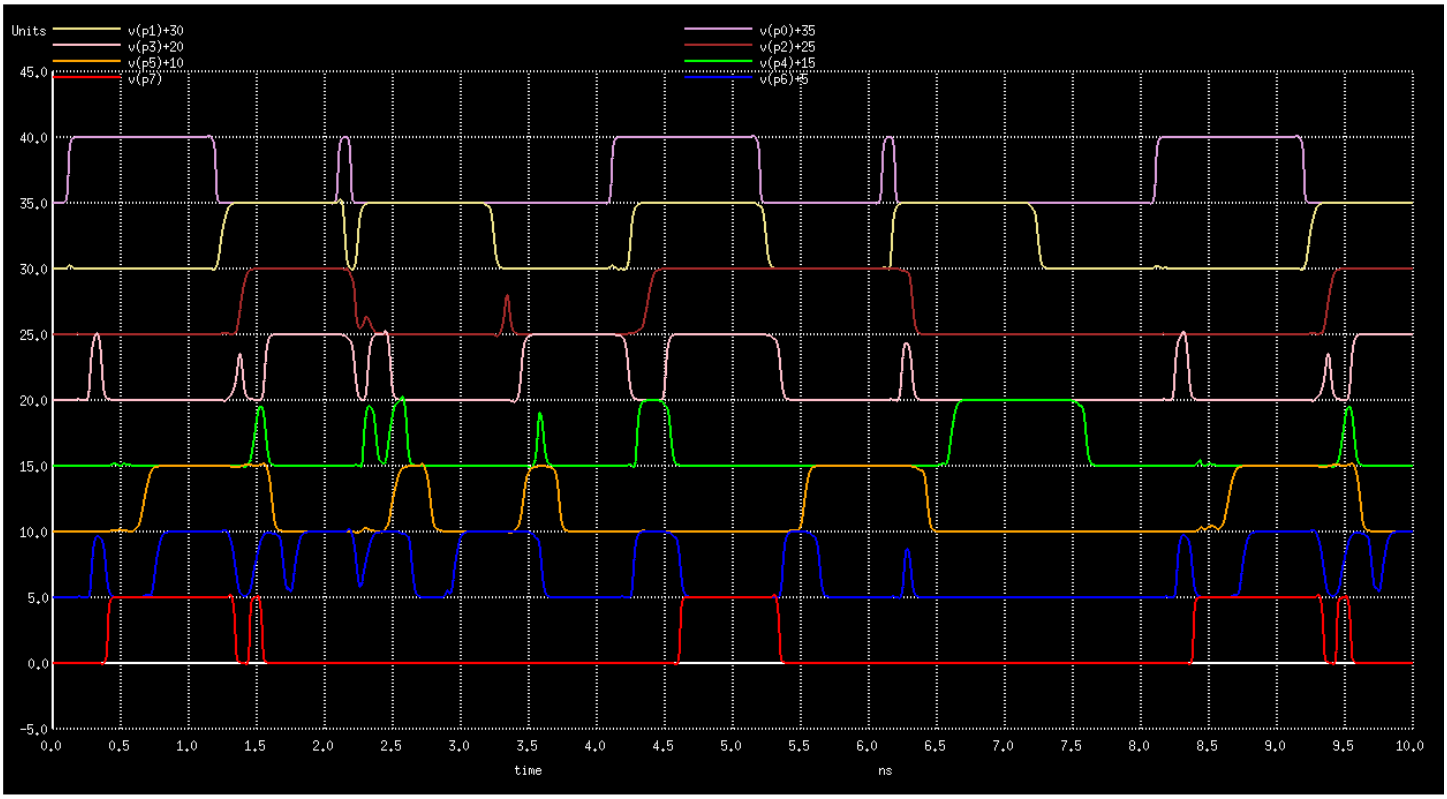


The stimulation results from the above spice file:

The input waveform:



The output waveform:



We observe from the waveform that the propagation delays and power leakages have hugely increased in the layout as compared to the circuit simulation generated using ngspice netlist.

We measure the power consumption in similar way as we need in case of ngspice netlist. We provide a dc input to the circuit for measuring leakage power. The power consumption for each input combination is printed in a text file named “power_output_magic.txt”. We write a python script to change the inputs in the spice file so that power consumption for all the input combinations is printed. We see that the power consumption has significantly increased from some 10nW to 10mW.

We also find the propagation delay for some 25 sample input and print them on a text file named “delay_output_magic.txt”. The propagation delay also increases from some 3-4ns to 30-40ns. In case of magic layout we get the worst case delay as: 3.98514E-08

VERILOG:

The Verilog code to test the working of our 4x4 multiplier circuit:

MULTIPLIER MODULE:

```
`timescale 1ns/10ps
module f_add(a, b, cin, cout, sum);
  input a,b,cin;
  output sum,cout;
  wire x, y, z;

  h_add h1(.a(a),.b(b),.s(x),.c(y));
  h_add h2(.a(x),.b(cin),.s(sum),.c(z));
  or o1(cout,y,z);
endmodule

module h_add(a, b, c, s);
  input a,b;
  output s,c;

  xor x1(s,a,b);
  and a1(c,a,b);
endmodule

```

```
module multiplier(p, a, b);
    input [3:0] a, b;
    output [7:0] p;
    wire a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16;
    wire  sum1, sum2, sum3, sum4, sum5, sum6, sum7, sum8, sum9, sum10, sum11, sum12;
    wire  c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12;

    and and1(a1, a[3], b[3]);
    and and2(a2, a[3], b[2]);
    and and3(a3, a[2], b[3]);
    and and4(a4, a[1], b[3]);
    and and5(a5, a[3], b[1]);
    and and6(a6, a[2], b[2]);
    and and7(a7, a[2], b[1]);
    and and8(a8, a[3], b[0]);
    and and9(a9, a[0], b[3]);
    and and10(a10, a[1], b[2]);
    and and11(a11, a[1], b[1]);
    and and12(a12, a[2], b[0]);
    and and13(a13, a[0], b[2]);
    and and14(a14, a[1], b[0]);
    and and15(a15, a[0], b[1]);
    and and16(a16, a[0], b[0]);

    h_add h1(a7, a8, c1, sum1);
    h_add h2(a11, a12, c2, sum2);
    h_add h3(a14, a15, c3, sum3);
    f_add f4(c1, a5, a6, c4, sum4);
    f_add f5(a2, a3, c4, c5, sum5);
    f_add f6(a10, c2, sum1, c6, sum6);
    f_add f7(a13, sum2, c3, c7, sum7);
    f_add f8(a9, c7, sum6, c8, sum8);
    f_add f9(a4, sum4, c6, c9, sum9);
    h_add h10(c8, sum9, c10, sum10);
    f_add f11(c10, c9, sum5, c11, sum11);
    f_add f12(c11, c5, a1, c12, sum12);

```

```
    assign p[0] = a16;
    assign p[1] = sum3;
    assign p[2] = sum7;
    assign p[3] = sum8;
    assign p[4] = sum10;
    assign p[5] = sum11;
    assign p[6] = sum12;
    assign p[7] = c12;

endmodule
```

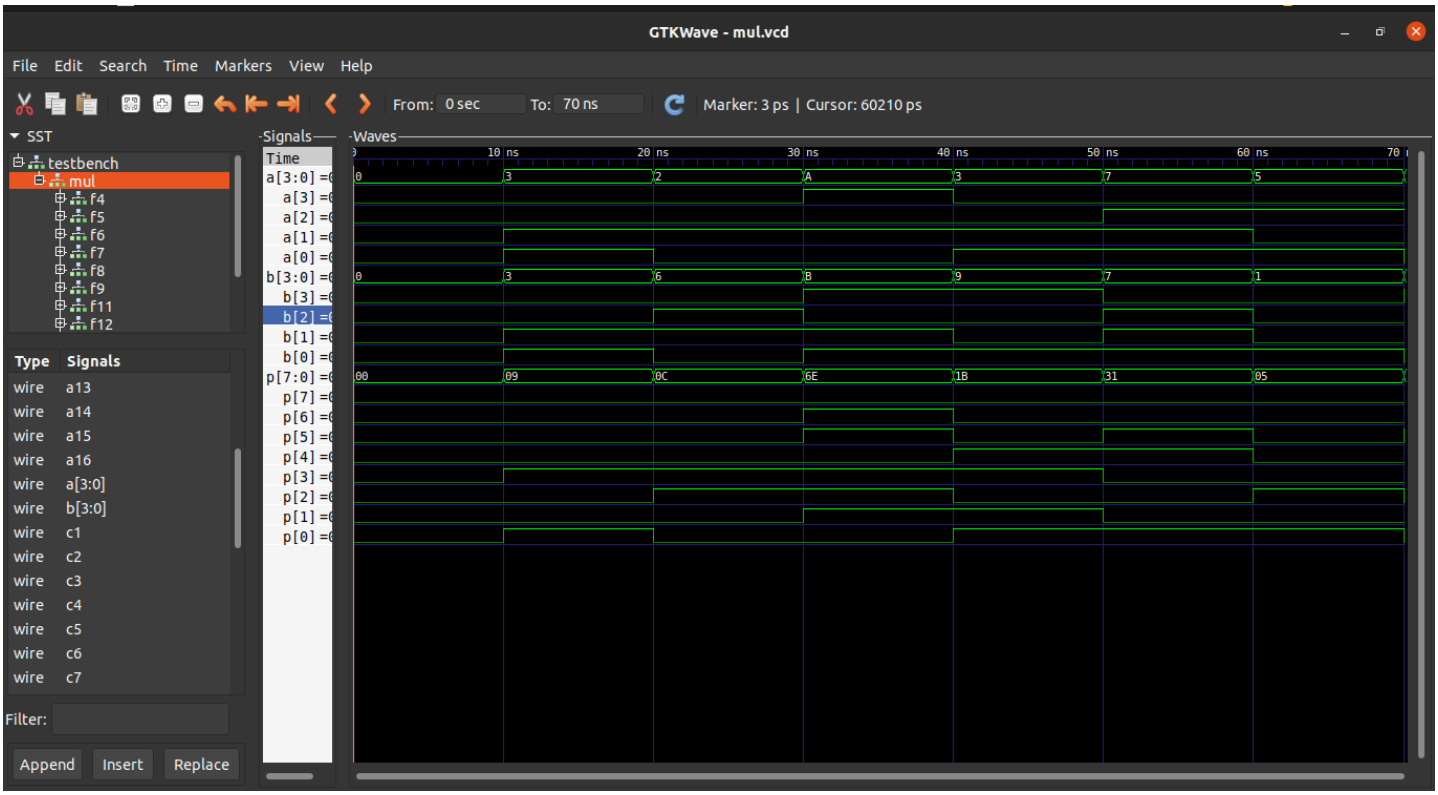
TESTBENCH:

```
`timescale 1ns / 1ps

module testbench();
    reg [3:0] a, b;
    output [7:0] p;

    multiplier mul(p, a, b);

    initial begin
        $dumpfile("mul.vcd");
        $dumpvars(0,mul);

        $monitor($time, "a = %d, b = %d, p = %d", a, b, p);
        a = 4'd0; b = 4'd0;

        #10 a = 4'd3; b = 4'd3;;
        #10 a = 4'd2; b = 4'd6;
        #10 a = 4'd10; b = 4'd11;
        #10 a = 4'd3; b = 4'd9;
        #10 a = 4'd7; b = 4'd7;
        #10 a = 4'd5; b = 4'd1;
        #10 a = 4'd4; b = 4'd8;

        $finish;
    end
endmodule
```

The output obtained for the given set of inputs:

```
 0a =  0, b =  0, p =   0
10a =  3, b =  3, p =   9
20a =  2, b =  6, p =  12
30a = 10, b = 11, p = 110
40a =  3, b =  9, p =  27
50a =  7, b =  7, p =  49
60a =  5, b =  1, p =   5
70a =  4, b =  8, p =  32
```

The GTKWAVE obtained for the above set of inputs:



A python script named “script.py” has also been written to give the output for all combination of inputs. The output for all the input combinations will be printed on the terminal by running the command: -

“python3 script.py”

SUBMISSION SPECIFICTIONS:

i. The codes and files for different implementation of 4x4 multiplier have been submitted in 3 different folders: NGSPICE, MAGIC, VERILOG

ii. For Ngspice, two netlists for the final 4x4 multiplier have been submitted, one where we give dc input to measure power consumption and the other where we give PWL input to measure propagation delay. A python script named “main.py” has also been added that changes the input values to measure the propagation delay for all possible input combinations. There are also two text files named “power_output.txt” and “delay_output.txt” that contains the values of leakage power and propagation delay for different inputs. This folder also contains subcircuits required for designing the multiplier.

iii. For Magic, two spice have we made for final 4x4multiplier, one where we give dc input to measure power consumption and the other where we give PWL input to measure propagation delay. A python script named “main_magic.py” has also been added that changes the input values to measure the propagation delay for all possible input combinations. There are also two text files named “power_output_magic.txt” and “delay_output_magic.txt” that contains the values of leakage power and propagation delay for different inputs. The folder contains all the .mag, .ext and .spice files for all the subcircuits used (like and gate, or gate, xor gate, half adder, full adder) and the final 4x4 multiplier.

iv. The Verilog file contains the main module file, two testbenches, and one python script. The python script changes the inputs in one of the testbench named “testbench.v” to print the output for all combinations of input. The other testbench named “testbench1.v” is used to print the output for few inputs.