UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
INTRODUÇÃO A MICROELETRONICA

RELATÓRIO II

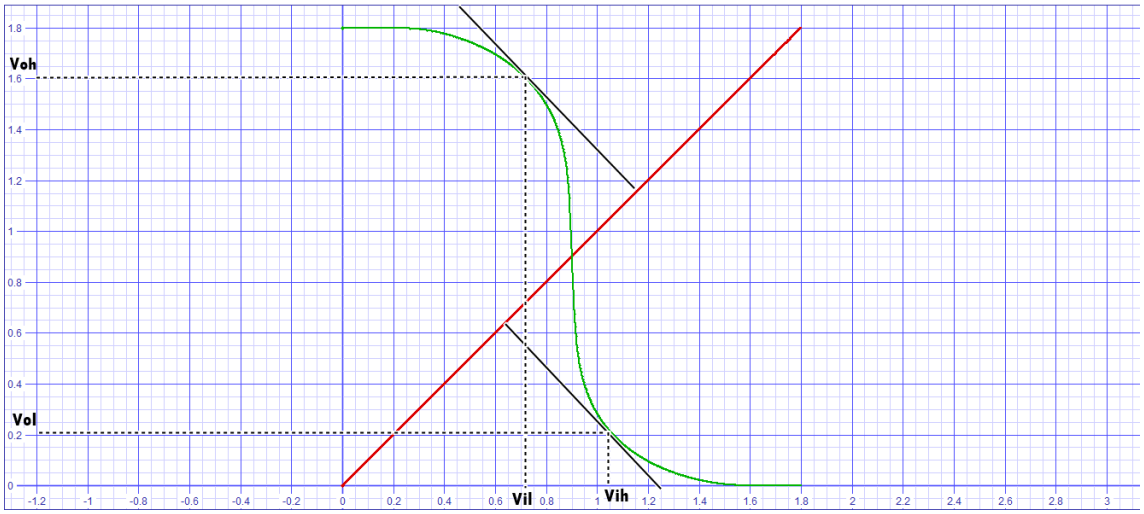
THIAGO ALVES DE ARAUJO
MATRICULA: 2016019787

João Pessoa
2018

1 – Margem de ruído

A margem de ruído é um parâmetro que permite que se determine o ruído de tensão na entrada de modo que a saída não seja prejudicada. A especificação mais usada para se caracterizar a margem de ruído utiliza dois parâmetros: a margem de ruído baixa, NML, e margem de ruído alta, NMH. NML é a diferença entre o máximo do nível de entrada baixa reconhecido pelo inversor, e o máximo do nível de saída baixa produzida pela saída do inversor (NML = VIL - VOL). Já o MHH é a diferença entre o mínimo do nível de saída alta e o mínimo de entrada alta do inversor (NMH = VOH - VIH).

Para calcular a margem de ruído, deve-se procurar na curva o ponto onde sua derivada(dVs/dVe) for igual a -1, ou seja, quando a reta tangente for ortogonal com a reta que representa o sinal da entrada. Para facilitar o cálculo, podemos utilizar Ctrl+R para deixar o aspecto da imagem em 1:1. Depois, com uma régua, podemos aproximar os pontos onde a reta ortogonal toca a curva como mostrado na imagem abaixo:

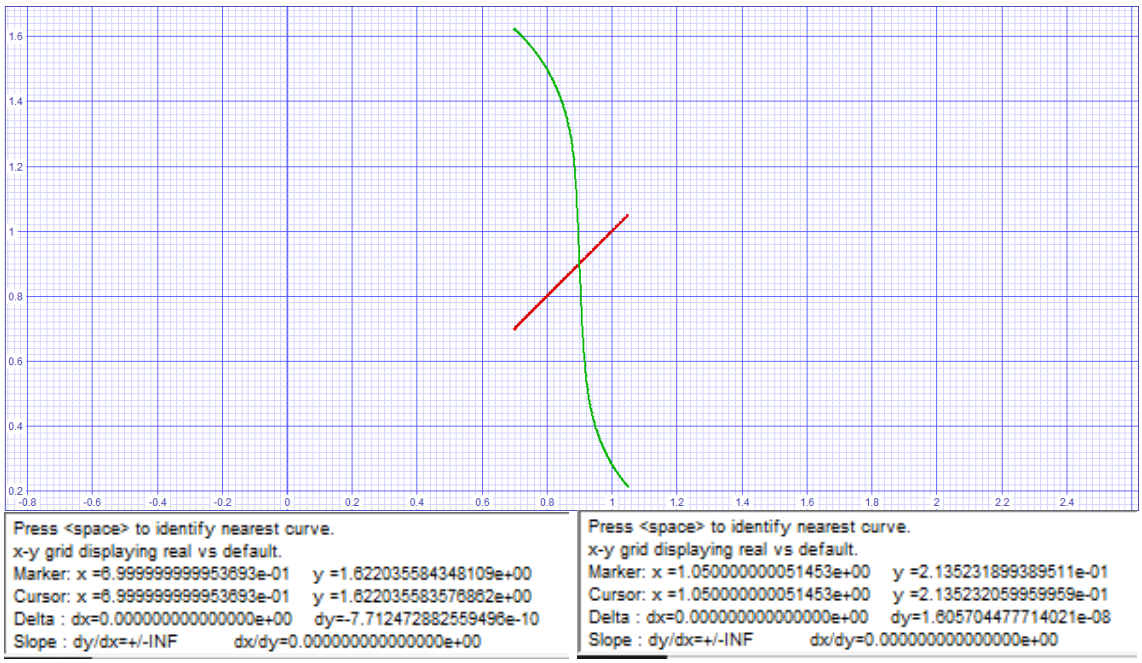


Valores da margem de ruído

Com isso, determinamos que para o inversor ideal projetado, os valores de Vil e Vih são aproximadamente 0,7 e 1,05 respectivamente. Para verificar se os valores encontrados estão corretos, podemos fazer uma nova simulação alterando o sinal de entrada para os valores encontrados.

```
.include inversor.spi

*   a   b   vdd vss
X1  20  40  10   30 inversor

V1  10  30  1.8V DC
V2  30  0   0V DC
V3  20  30  0V

.model tp pmos level=54
.model tn nmos level=54

.dc V3 0.7 1.05 .001
.end
```

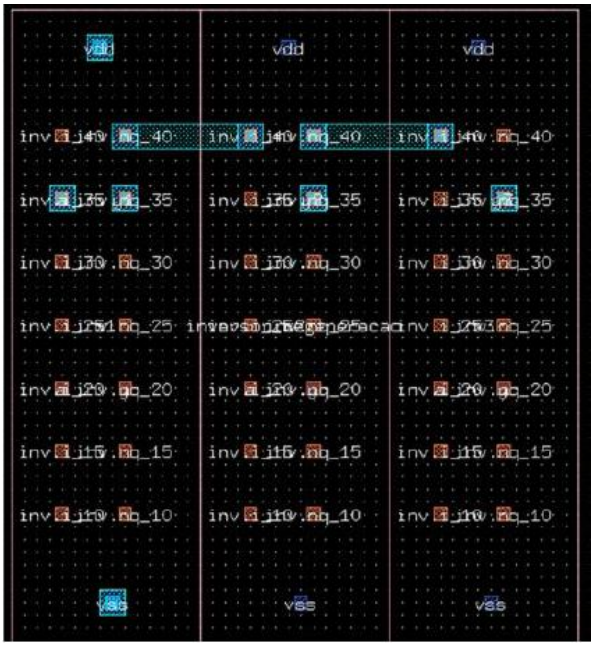
Após a simulação, percebemos que os valores de Vol e Voh encontrados anteriormente estão condizentes com os valores da simulação.



Voh – Vil Vol - Vih

2 - Regeneração da família CMOS - regeneração estática

Para analisarmos o comportamento do inversor ideal com uma entrada entre os valores da margem, podemos conectar três inversores em série para simularmos o inversor numa situação de regeneração estática.



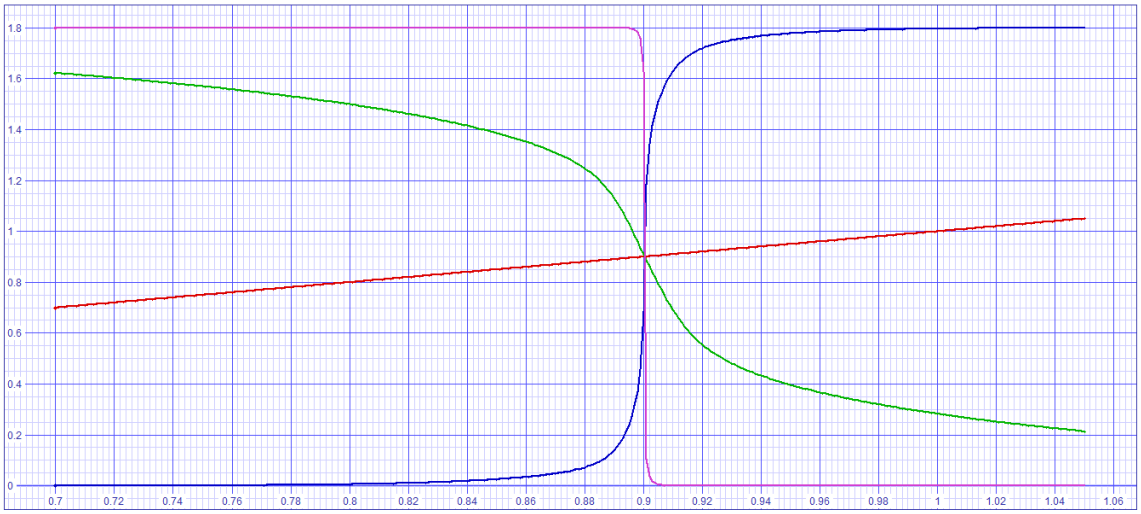
Layout Regeneração

Para isso, podemos criar um novo circuito no *graal* colocando os três *inversores* ideais em serie e criando um novo arquivo *".spi"'* no *cougar* como descrito anteriormente. Após isso, criamos um novo arquivo "*.cir"* para a simulação.

```
.include inversor_regeneracao.spi

*   a vdd vss y1 y2 y3

X1 20 10 30 40 41 42 inversor_regeneracao

V1 10 30 1.8V DC
V2 30 0 0V DC
V3 20 30 0V

.model tp pmos level=54
.model tn nmos level=54

.dc V3 0.7 1.05 .001
.end
```

Arquivo .cir utilizado para a simulação

Após executar o comando no Spice Opus *plot v(10) v(40) v(41) v(42)* obtivemos:



Regeneração estática

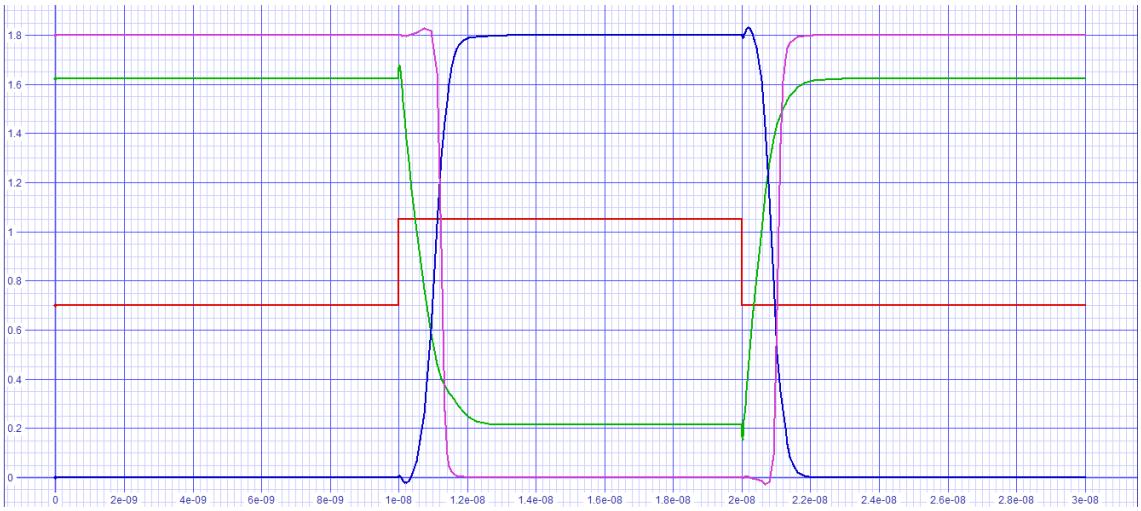
3 - Regeneração da família CMOS - Regeneração dinâmica

Utilizando o mesmo circuito da simulação anterior, alteramos a sinal de entrada para um pulso com amplitude de Vil até Vih.

```
.include inversor_regeneracao.spi

*  a vdd vss y1 y2 y3

X1 20 10 30 40 41 42 inversor_regeneracao

V1 20 30 pulse(0.7 1.05V 10ns 1ps 1ps 10ns 20ns)
V2 30 0 0V
V3 10 30 1.8V

.model tp pmos level=54
.model tn nmos level=54

.tran 0.000001 30ns
.end
```

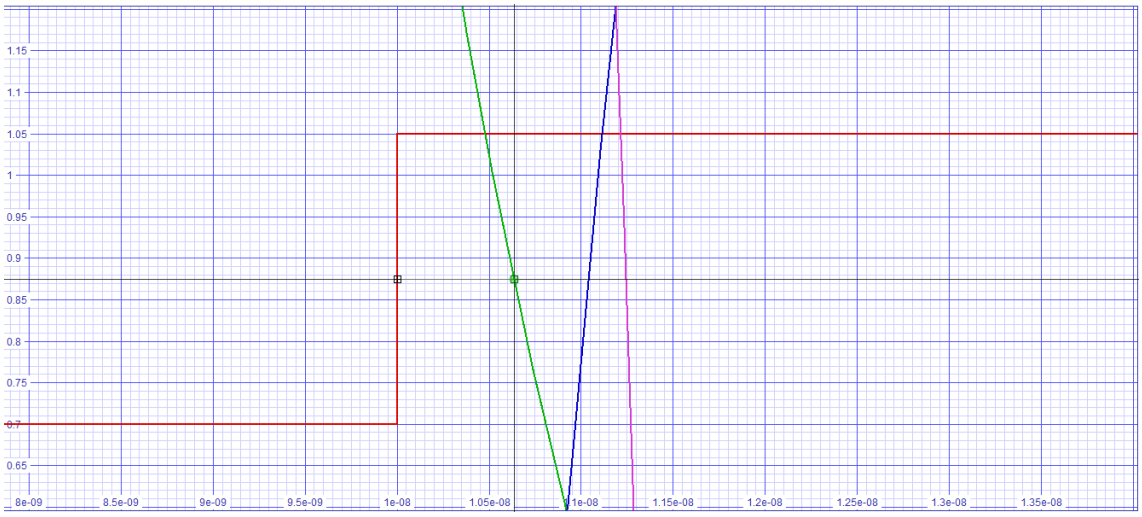
Arquivo .cir para simulação dinâmica

Então, vamos no *Spice Opus* e executamos o comando *plot v(10) v(40) v(41) v(42)* para assim obter os sinais da saída:



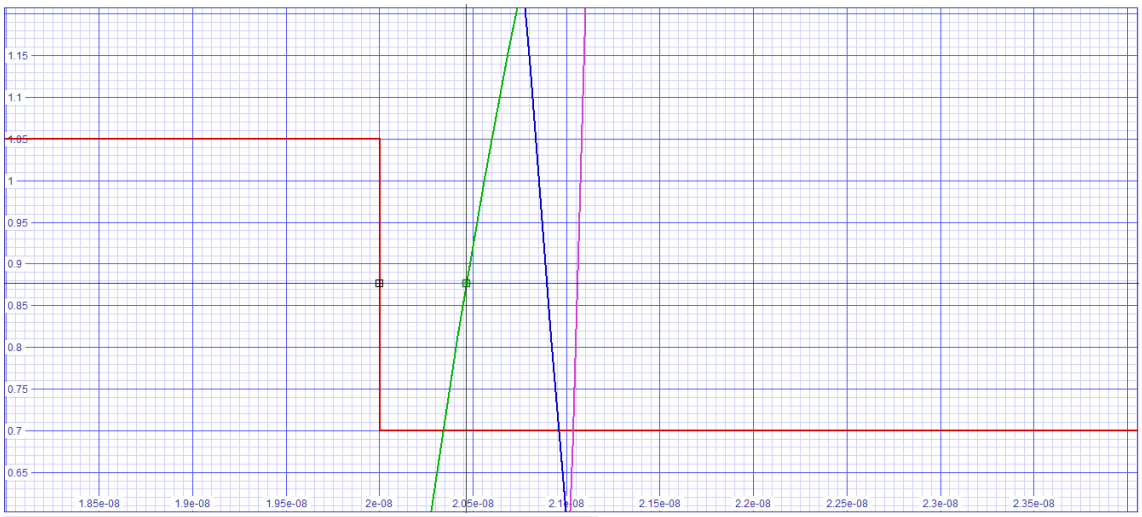
Regeneração dinâmica

Com isso, podemos medir o tempo de propagação de cara inversor.



Nearest curve: v(40)
x-y grid displaying real vs default.
Marker: x =1.000188281214172e-08 y =8.746351064459434e-01
Cursor: x =1.064131536975985e-08 y =8.746351064459434e-01
Delta : dx=6.394325576181280e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

HL v40 – 6,393e-10

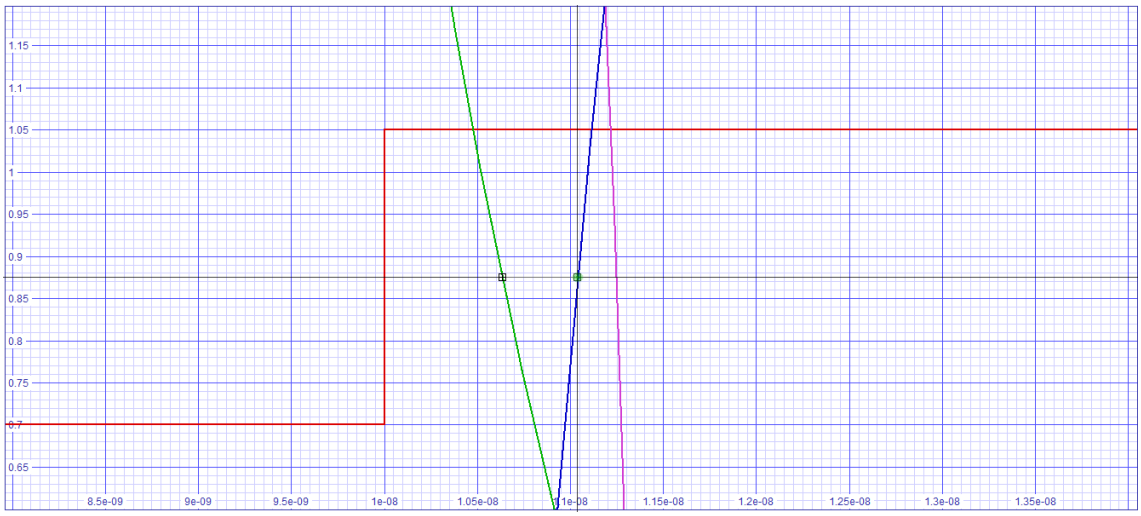


Nearest curve: v(40)
x-y grid displaying real vs default.
Marker: x =1.999798640162923e-08 y =8.769161716898093e-01
Cursor: x =2.046298749717621e-08 y =8.769161716898093e-01
Delta : dx=4.650010955469822e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

LH v40 – 4,65e-10

Nearest curve: v(41)
x-y grid displaying real vs default.
Marker: x =2.045386982863607e-08 y =8.738473158720309e-01
Cursor: x =2.089151791856264e-08 y =8.738473158720309e-01
Delta : dx=4.376480899265704e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

HL v41 – 4,376e-10

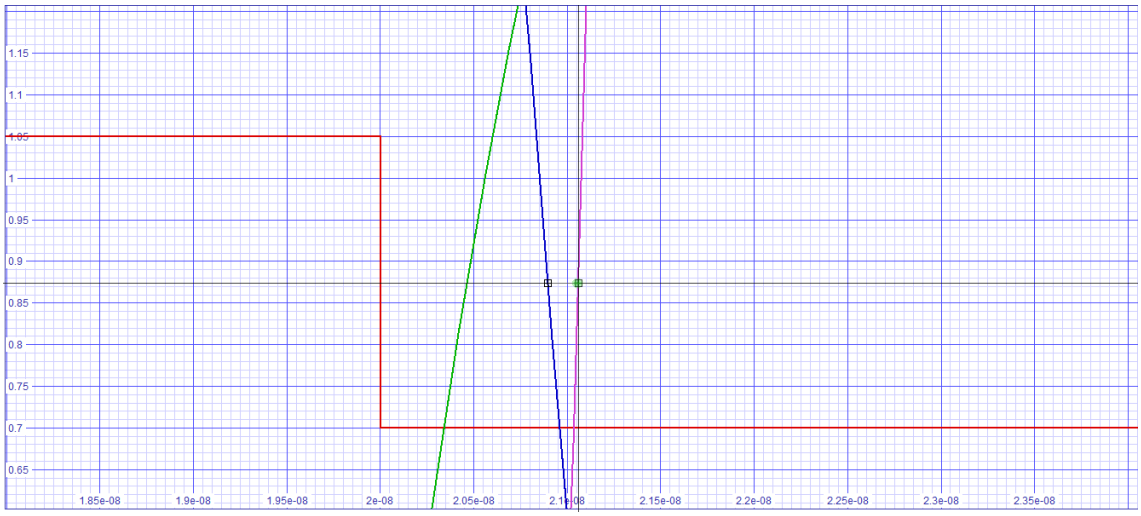


Nearest curve: v(41)
x-y grid displaying real vs default.
Marker: x =1.063376713596842e-08 y =8.753641066785189e-01
Cursor: x =1.104116623948827e-08 y =8.753641066785189e-01
Delta : dx=4.073991035198468e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

LH v41 – 4,074e-10

Nearest curve: v(42)
x-y grid displaying real vs default.
Marker: x =1.104574375750535e-08 y =8.743529128075269e-01
Cursor: x =1.125173206827381e-08 y =8.743529128075269e-01
Delta : dx=2.059883107684621e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

HL v42 – 2,060e-10

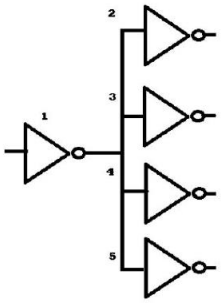


Nearest curve: v(42)
x-y grid displaying real vs default.
Marker: x =2.089607675283271e-08 y =8.738473158720309e-01
Cursor: x =2.105563595228511e-08 y =8.738473158720309e-01
Delta : dx=1.595591994523944e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

LH v42 – 1,596e-10

4 - FO4 (Fan-Out of 4)

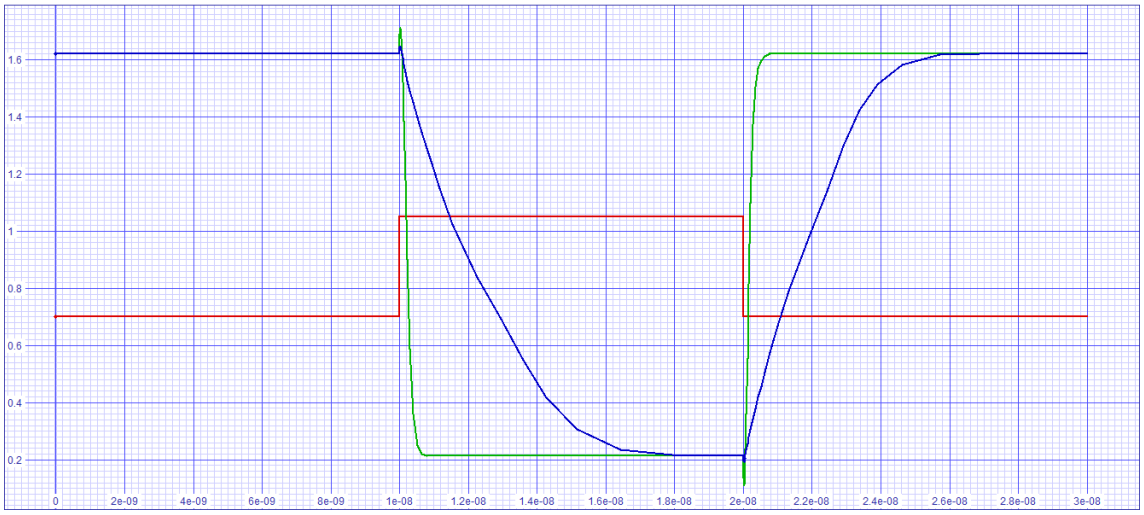
O FO4 (fan-out of 4) consiste em um método para medição do tempo de atraso de uma porta lógica em que são conectados quatro inversores na saída de um inversor para que seja analisado sua performance.



Abrindo o *graal*, vamos em *new/create/instance/no sym* e selecionamos o ponto de coordenadas (0,0). Selecionamos o arquivo "*.ap"* do inversor ideal e nomeamos ele de "X1". Após isso, clicamos no canto inferior direito do inversor e inserimos outro inversor ideal, fazendo o processor até q tenhamos 5 inversores. Com os inversores inseridos, ajustamos o tamanho do *abutment box* para que todos os 5 inversores fiquem dentro do mesmo e então começamos a fazer as conexões. Vamos em *create/via/via1-2* e selecionamos os pontos onde vai se encontrar as conexões (Vdd, Vss, A1, Y1, Y2, Y3, Y4, Y5). Após isso colocamos os seguimentos (*create/seguimento/alu2*) entre a saída o primeiro inversor e a entrada dos demais. O circuito final é mostrado a seguir:

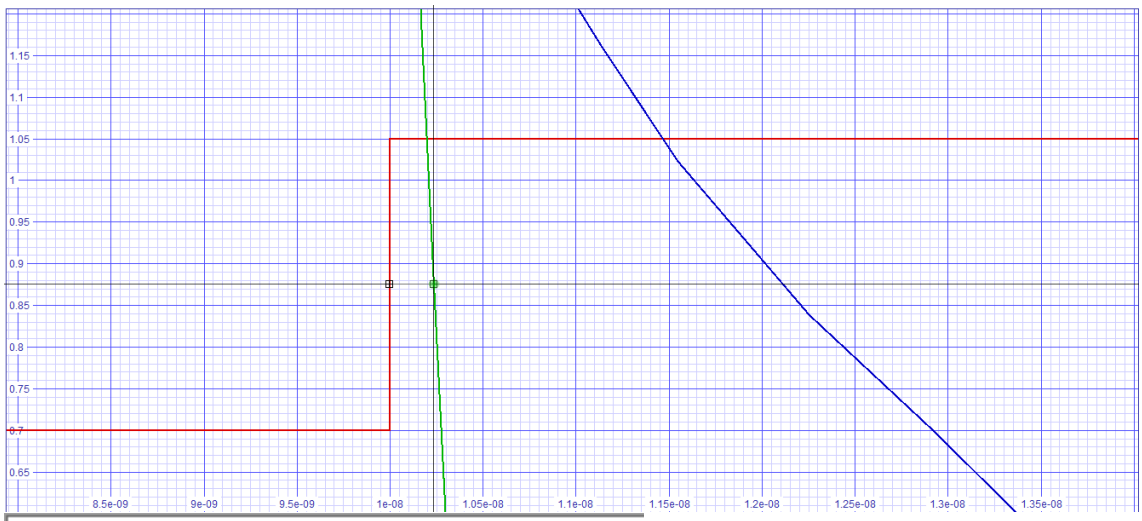
Agora vamos compara-lo com o inversor equilibrado.

```
.include inversor.spi
.include fo4.spi

* a vdd vss y1 fo4
* a b vdd vss  inversor

X1 20 10 30 41 fo4
X2 20 40 10 30 inversor

V1 20 30 pulse(0.7 1.05V 10ns 1ps 1ps 10ns 20ns)
V2 30 0 0V
V3 10 30 1.8V

.model tp pmos level=54
.model tn nmos level=54

.tran 0.00001 30ns
.end
```



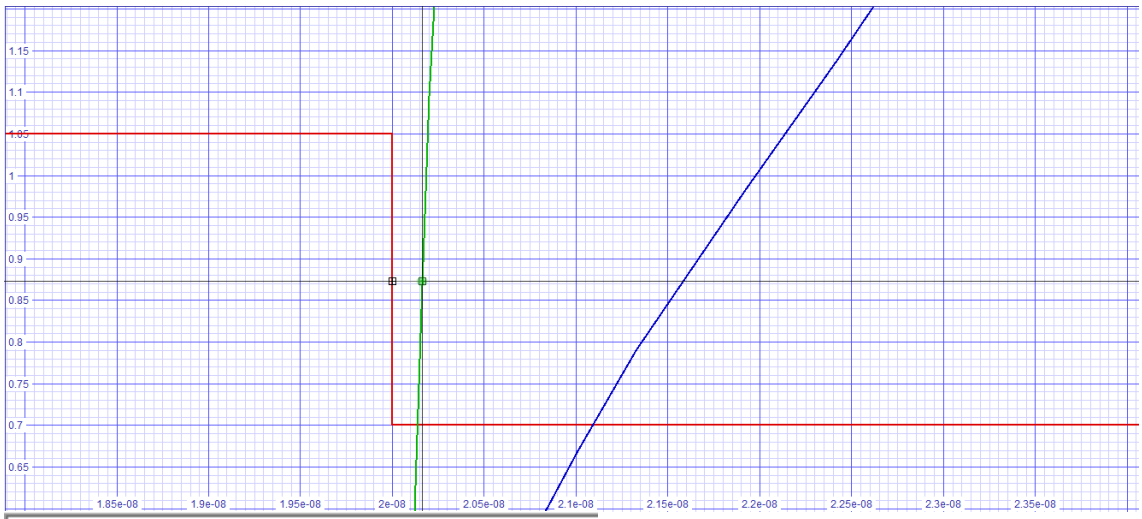
Linha azul – FO4 v(41)
Linha verde – inversor equilibrado v(40)

Agora vamos calcular os tempos de propagação:



Nearest curve: v(40)
x-y grid displaying real vs default.
Marker: x =9.995549021905844e-09 y =8.756376510655353e-01
Cursor: x =1.023815747681092e-08 y =8.756376510655353e-01
Delta : dx=2.426084549050778e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

HL Inversor – 2,426e-10



Nearest curve: v(40)
x-y grid displaying real vs default.
Marker: x =2.000024713501725e-08 y =8.730705327219543e-01
Cursor: x =2.016307599019613e-08 y =8.730705327219543e-01
Delta : dx=1.628288551788806e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

LH inversor 1,628e-10

João Pessoa
2018

Nearest curve: v(41)
x-y grid displaying real vs default.
Marker: x =9.999435241283607e-09 y =8.994151416798483e-01
Cursor: x =1.199981061474623e-08 y =9.014404815958885e-01
Delta : dx=2.000375373462623e-09 dy=2.025339916040259e-03
Slope : dy/dx=1.012479928971742e+06 dx/dy=9.876738998822265e-07

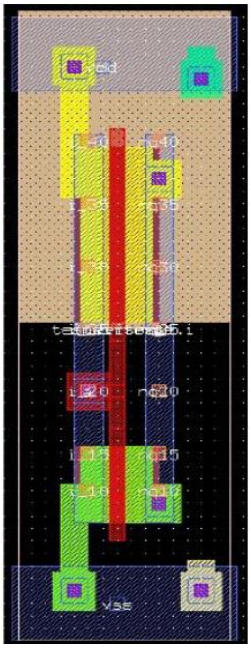
HL FO4 – 2,000e-10

Nearest curve: v(41)
x-y grid displaying real vs default.
Marker: x =1.999740720547196e-08 y =9.002657844445852e-01
Cursor: x =2.166582844583497e-08 y =9.002657844445852e-01
Delta : dx=1.668421240363010e-09 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

LH FO4 – 1,665e-10

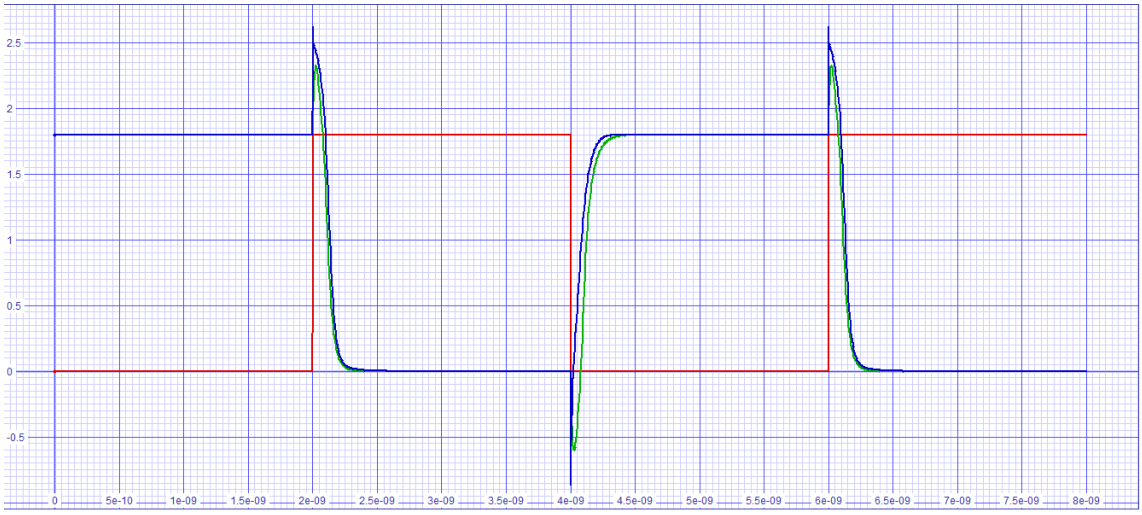
5 - Inversor gordo

Agora vamos dobrar o tamanho dos transistores do inversor criado anteriormente. Para isso, voltamos ao *graal* e dobramos o tamanho do dos transistores p e n mantendo o equilíbrio. Depois, vamos no *cougar* e geramos um novo arquivo arquivo "*.spi*"



Para observar o comportamento desse novo inversor, podemos simula-lo e compara-lo com o inversor antigo utilizado até agora. Abaixo esta o resultado da comparação.

```
.include inversor.spi
.include inversor_gordo.spi

* inve a b vdd vss
* invg a b vdd vss

X1 20 40 10 30 inversor
X2 20 41 10 30 inversor_gordo

V3 10 30 1.8V DC
V2 30 0 0V DC
V1 20 30 pulse(0V 1.8V 2ns 1ps 1ps 2ns 4ns)

.model tp pmos level=54
.model tn nmos level=54
.tran 0.001ns 8ns
*.dc V3 0 1.8 .001
.end
```



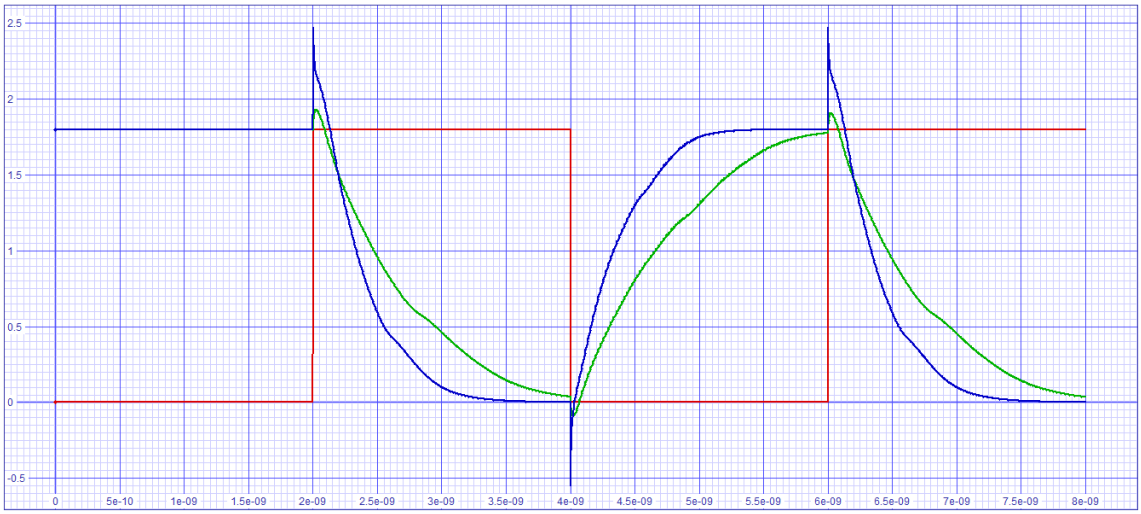
Linha azul – inversor "gordo" equilibrado
Linha verde – inversor equilibrado

Agora vamos criar outro inversor FO4 substituindo o 1° inversor pelo inversor gordo. Para isso, vamos fazer o mesmo processo descrito anteriormente. Para facilitar o trabalho, podemos abrir o circuito do inversor gordo, inserir o FO4 com instancia e apenas fazer as conexões. Abaixo está a imagem do circuito:

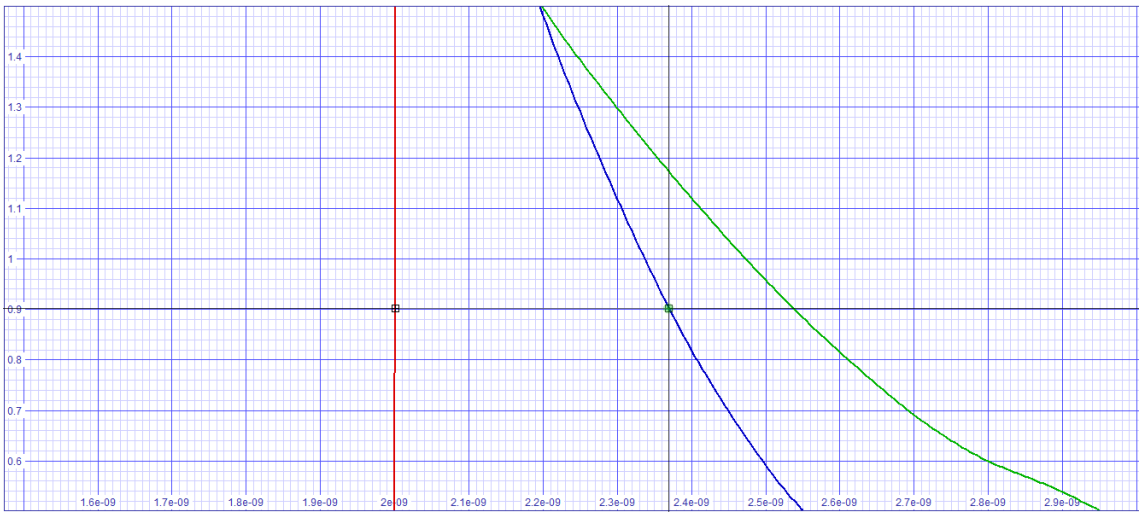
Agora podemos simular e comparar o FO4 com o FO4 “gordo”

```
.include fo4.spi
.include fo4_gordo.spi

* fo4  a vdd vss y1
* fo4g a vdd vss y1

X1 20 40 10 30  fo4
X2 20 40 10 31  fo4_gordo

V1 20 10 pulse(0V 1.8V 2ns 1ps 1ps 2ns 4ns)
V2 10 0  0V DC
V3 40 10 1.8V

.model tp pmos level=54
.model tn nmos level=54

.tran 0.001ns 8ns
.end
```

-



Linha verde – FO4
Linha azul – FO4 “gordo”

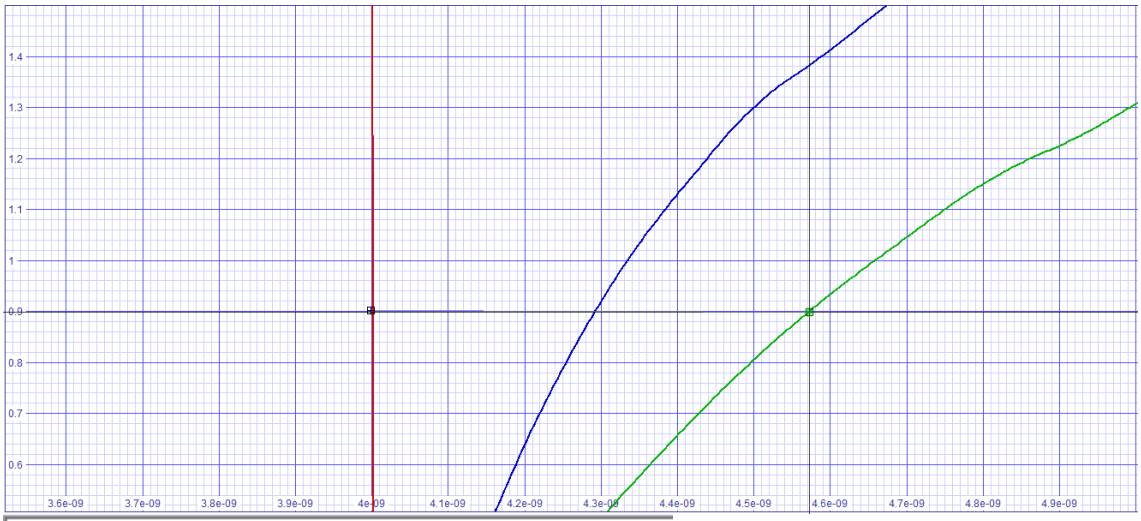


Nearest curve: V(31)
x-y grid displaying real vs default.
Marker: x =2.001010960565404e-09 y =9.019800973712271e-01
Cursor: x =2.368154062780787e-09 y =9.019800973712271e-01
Delta : dx=3.671431022153826e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

HL FO4 gordo – 3,671E-10

Nearest curve: V(31)
x-y grid displaying real vs default.
Marker: x =3.998617402721373e-09 y =9.019991853750402e-01
Cursor: x =4.291204880874128e-09 y =9.019991853750402e-01
Delta : dx=2.925874781527552e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

LH FO4 gordo – 2,925E-10



Nearest curve: V(30)
x-y grid displaying real vs default.
Marker: x =3.998617402721373e-09 y =9.019991853750402e-01
Cursor: x =4.571461086001287e-09 y =9.003194410394795e-01
Delta : dx=5.728436832799148e-10 dy=-1.679744335560729e-03
Slope : dy/dx=-2.932290927854986e+06 dx/dy=-3.410302813068807e-07

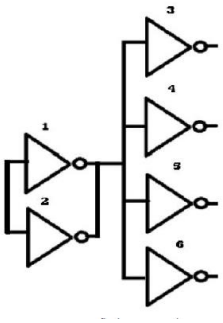
LH FO4 – 5,728E-10

Nearest curve: V(30)
x-y grid displaying real vs default.
Marker: x =2.001010960565404e-09 y =9.019800973712271e-01
Cursor: x =2.536188209876229e-09 y =9.019800973712271e-01
Delta : dx=5.351772493108243e-10 dy=0.000000000000000e+00
Slope : dy/dx=0.000000000000000e+00 dx/dy=+/-INF

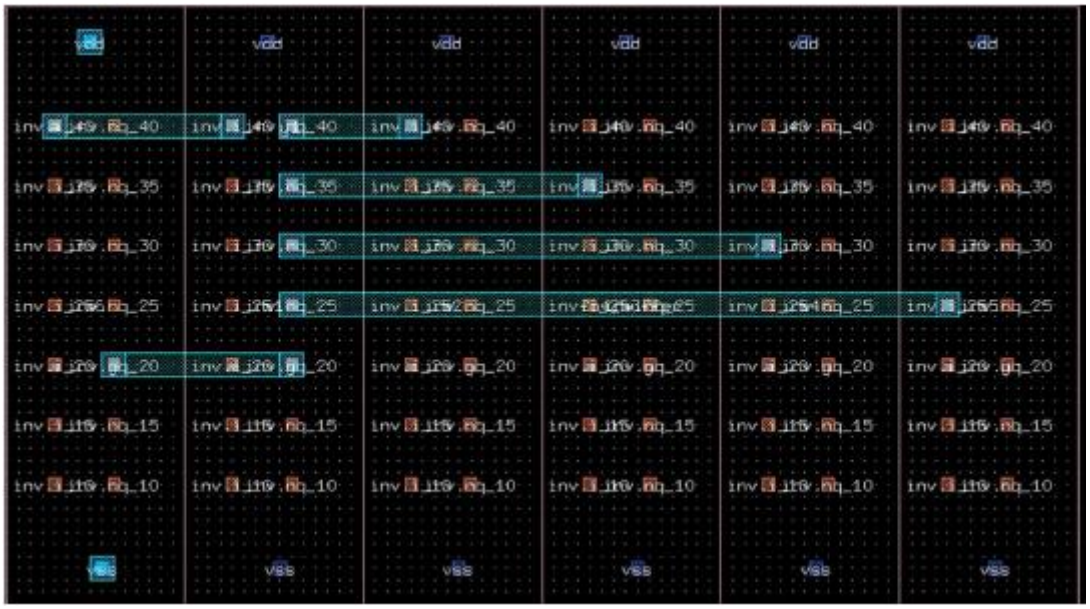
HL FO4 – 5,351E-10

6 – Bufferização

Agora vamos criar outro circuito FO4 substituindo o inversor gordo por dois inversores equilibrados ligados em paralelo.



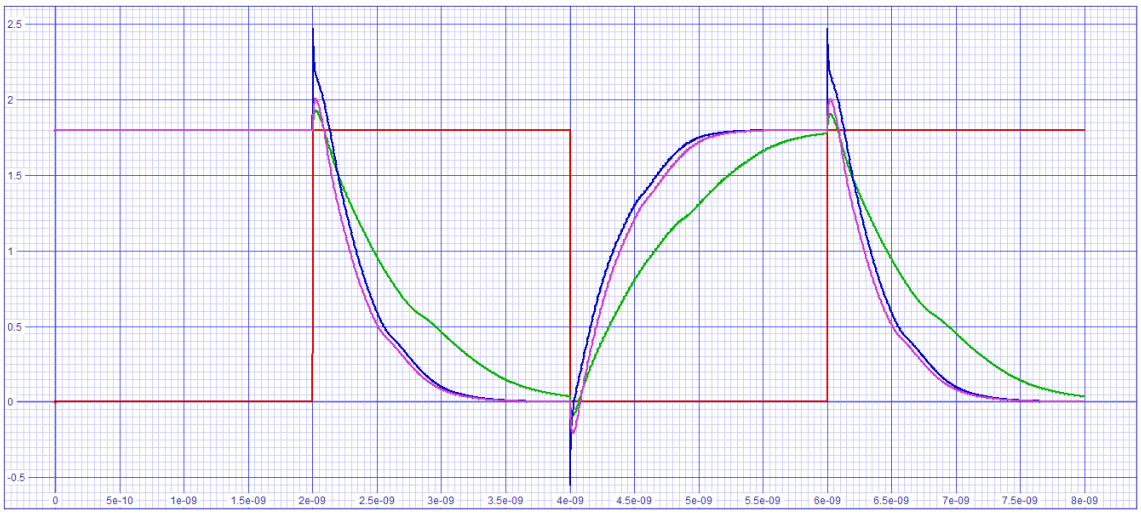
Abrindo o graal e fazendo as devidas conexões temos o seguinte circuito:



Por fim, vamos simular os três circuitos criados até agora: FO4, FO4 gordo e Buffer. Abaixo está o arquivo ".cir" utilizado

```
.include fo4.spi
.include fo4_gordo.spi
.include fo4_buffer.spi


* fo4  a vdd vss y1
* fo4g a vdd vss y1
* fo4b a vdd vss y1

X1 20 40 10 30  fo4
X2 20 40 10 31  fo4_gordo
X3 20 40 10 32  fo4_buffer

V1 20 10 pulse(0V 1.8V 2ns 1ps 1ps 2ns 4ns)
V2 10 0  0V DC
V3 40 10 1.8V

.model tp pmos level=54
.model tn nmos level=54

.tran 0.001ns 8ns
.end
```

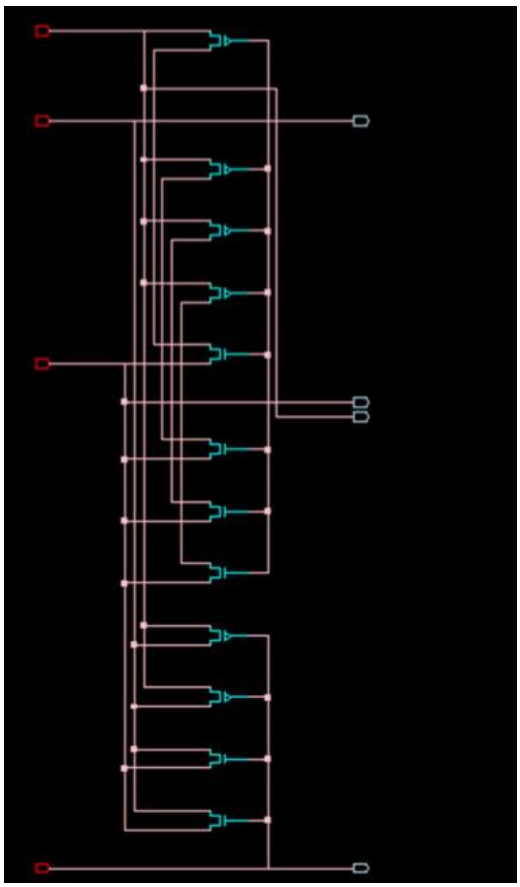
João Pessoa
2018



Linha verde – FO4 v(30)
Linha azul – FO4 gordo v(31)
Linha roxa – FO4 buffer v(32)

Por fim, podemos utilizar o do visualizador de *net list* em forma de diagrama esquemático (circuito com transistores) do buffer utilizando os seguintes comandos no *cougar*:

```
export MBK_IN_LO = spi
xsch -l ‘nome‘ &
```



João Pessoa
2018