DIC
ID: 519021911248
Name: ZhuoHao Li

Lab 3

2021-12-1

1 Introduction

Chip designers face a bewildering array of choices.

1. What is the best circuit topology for a function?
2. How large should the transistors be?
3. How many stages of logic give least delay?

Logical Effort is a method of answering these questions:

1. Uses a very simple model of delay
2. Back of the envelope calculations and tractable optimization
3. Gives new names to old ideas to emphasize remarkable symmetries

Who cares about logical effort?

1. Circuit designers waste too much time simulating and tweaking circuits
2. High speed logic designers need to know where time is going in their logic
3. CAD engineers need to understand circuits to build better tools

Delay of a gate d has two components. A fixed part called parasitic delay p. A part proportional to the load on the output called the effort delay or stage effort f. Total delay is measured in units of , and is sum of these delays. d = f + p.

The effort delay (due to load) can be further broken down into two terms: f = g * h. g = logical effort which captures properties of the gate' s structure. h = electrical effort which captures properties of load and transistor sizes. h = Cout/Cin. Cout is capacitance that loads the output. Cin is capacitance presented at the input. So, d = gh + p.

2 Lab Procedures

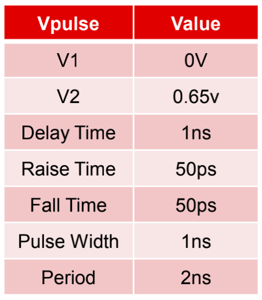
2.1 Use Logic Effort to Size Gates

Size each transistor in the circuit below properly to optimize the speed for FinFET, and construct the circuit with HSPICE. Usually a minimized FinFET device is equipped with an equivalent Fin number of 1 for both nfet and pfet. From this, we design a whole chain, using HSPICE to simulate it.

We'll design a combine logic like Fig.1:

The impulse parameter has been set as the left of Fig.1 shown downside.

I design **3 sub circuits**, 1 for FinFET inverter, it has an equivalent default Fin number for nfet and pfet. (1). It's just like ordinary CMOS inverter. The only defference is that CMOS' ratio of pmos and noms is 2:1, but FINFET is 1:1.



Vpulse	Value
V1	0V
V2	0.65v
Delay Time	1ns
Raise Time	50ps
Fall Time	50ps
Pulse Width	1ns
Period	2ns

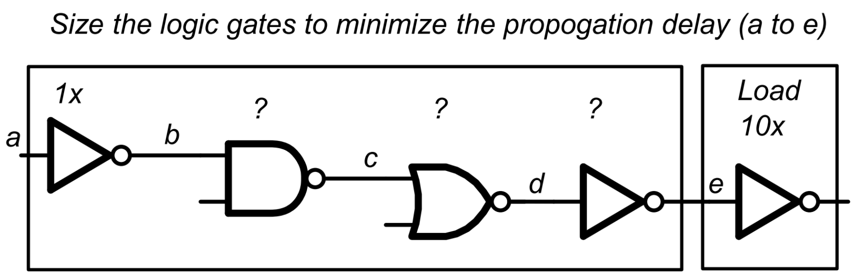

Size the logic gates to minimize the propogation delay (a to e)
1x
?
?
?
Load
10x
a
b
c
d
e

图 1: task1

As for the NAND and NOR, are also really similar to CMOS implement. Ther are tremendous implement methods to that. Here I list some latest NAND implement using different FinFET tech(Fig.2).

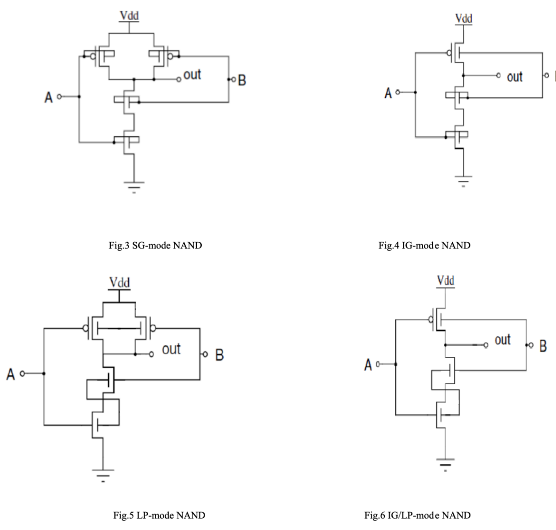


图 2: many different NAND

compared to that, the ordinary CMOS NAND are shown as belows:

The core difference is the minimized FinFET NAND2' nfet size is 2, and pfet size is 1. With the same way, the minimized FinFET NOR's nfet is 2, and pfet is 2. There's 1 trick: **We link the body of nfet to the gnd and pfet to the vdd, unlike those results shown in papers**. FinFET is not liek bulk-silicon CMOS we learned in class. FinFET's causes of controlling is merely **FINNUM**. So that's because in every step of our simulation, we should set the fin number as an **INTEGER**!. In manual calculation, we may get a float number, then we need to evaluate it and select a closed integer to the float! We must be clear that device property details when we are considering some possible effects in our study or research! One possible structure is like this(Fig.4):

.

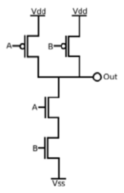


图 3: ordinary CMOS NAND

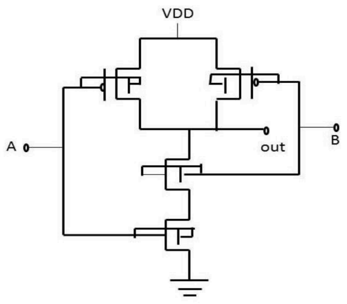


图 4: FinFET NAND

The core module is shown below:
Let's make an explain to make the code more clear! For the most important part, is which pin should we connect to the correct pin. As for an inverter, it's quite easily to be shown that drain is connected to the output, and gate is connceted to the input. Also, the body is connected to the gnd for nfet and vdd for pfet, the source is connected to the gnd or middle node if possible. Also, we know that our design is not advanced so there's some simplification in our lab.

```
.subckt inv in out l=length num=n
xnmos out in gnd gnd nfet l=length NFIN=num
xpmos out in vdd vdd pfet l=length NFIN=num
.ends //inverter

.subckt NAND2 a b out l=length num=n
xnfet1 out b net gnd nfet l=length NFIN='2*num'
xnfet2 net a gnd gnd nfet l=length NFIN='2*num'
xpfet1 out a vdd vdd pfet l=length NFIN=num
xpfet2 out b vdd vdd pfet l=length NFIN=num
.ends //NAND2

.subckt NOR2 a b out l=length num=n
xnfet1 out b gnd gnd nfet l=length NFIN=num
xnfet2 out a gnd gnd nfet l=length NFIN=num
xpfet1 out a net vdd pfet l=length NFIN='2*num'
xpfet2 net b vdd vdd pfet l=length NFIN='2*num'
.ends //NOR2
```

inv nand2 nor2.sp

Before formal lab, I tested **inverter, NAND2, NOR2** separately. You can check the source code in the ***test*** folder.Here are the results:(Figure 5)

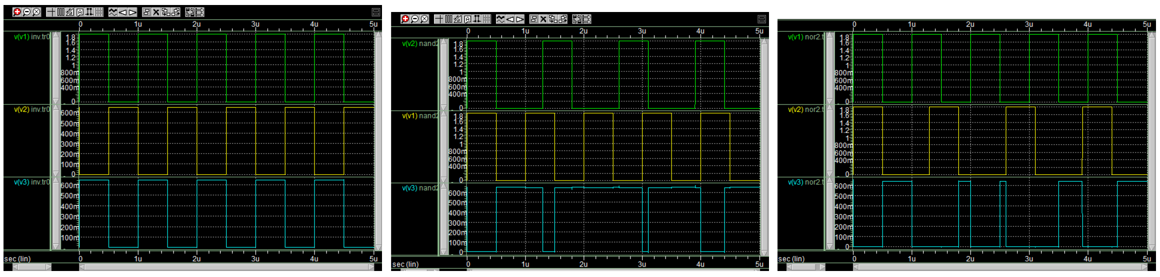


图 5: inverter, NAND2, NOR2

Due to the circuit in Fig.1, using the code below to connect them:

```
x1 va vb inv
x2 vb vdd vc NAND2 num=n
x3 vc gnd vd NOR2 num=n
x4 vd ve inv num=n
x5 ve vo inv num=10

vs va gnd pulse(0,0.65,1ns,50ps,50ps,1ns,2ns)
vdd vdd gnd supply

.tran 0.1n 10n
.option post accurate probe
.probe v(va) v(vb) v(vc) v(vd) v(ve) v(vo)
.meas tran tpLH
+trig v(va)='0.5*supply' rise=2
+targ v(ve)='0.5*supply' rise=2
.meas tran tpHL
+trig v(va)='0.5*supply' fall=2
+targ v(ve)='0.5*supply' fall=2
.measure tp param='(tpLH+tpHL)/2'
```

t1.sp

2.2 Use Logic Effort to Optimize 4x16 Decoder

Backgroud of decoder:
- Logic function: convert N-bit coded input to 2N unique outputs
- Can be used as address decoder in memory
–Address line (input) has large fanout
–Decoders typically have high electrical effort and branching effort
–Decoder may have many stages, so the fastest design is the one that minimizes the logical effort

task2 is using logic effort to optimize a decoder:
For a 4x16 decoder, we just need to simulate the critical path to minimize the propagration delay of decoder

- Use the logical effort method
- Assume that the input inverter/buffer is minimum sized
- Assume that each output of decoder has a load of 256X minimum sized Finfet inverter
- insert buffer at the output if necessary

the critical path needs to be the longest path in the circuit, that is:

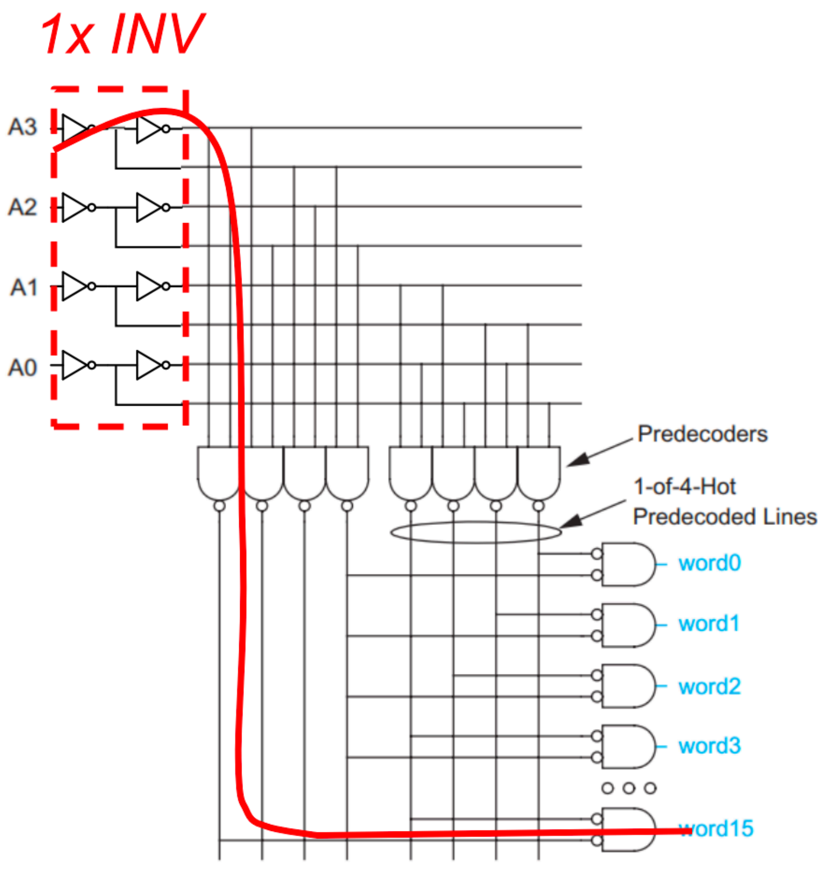


图 6: task2

The critical path is marked by the red line above

So, what we need to do is simulating the critical path. It has a dual buffer, 2 NAND2, 1 NOR2. The $256 \times C_{load}$ is not shown above! The main core is like this:

```
xa3 va3 v3p inv
xa2 va2 v2p inv
xa1 va1 v1p inv
xa0 va0 v0p inv
xb3 v3p v3n inv
xb2 v2p v2n inv
```

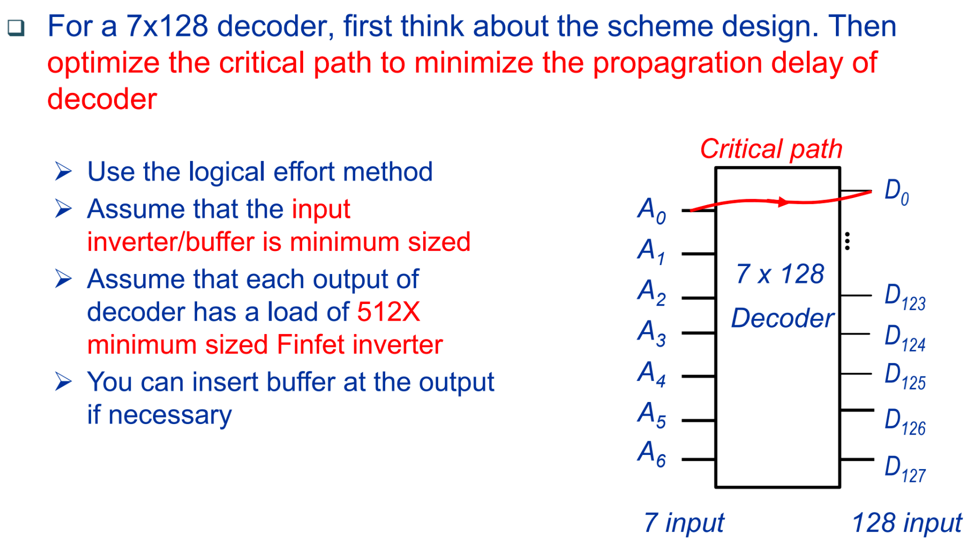
```
xb1 v1p v1n inv
xb0 v0p v0n inv
xnand1 v3n v2n vo1 NAND2 num= //opt size
xnand2 v1n v0n vo2 NAND2 num= //opt size
xnor2 vo1 vo2 vo NOR2 num= //opt size
//possible buffers
xo1 vo voo inv num=//opt size
xo2 voo vooo inv num=//opt size
xo3 vooo vpo inv num=//opt size
xo4 vpo vpp inv num=//opt size
xout vpp vout inv num=//opt size

.meas tran tpLH
+trig v(va3)='0.5*supply' rise=2
+targ v(vpp)='0.5*supply' rise=2
.meas tran tpHL
+trig v(va3)='0.5*supply' fall=2
+targ v(vpp)='0.5*supply' fall=2
.measure tp param='(tpLH+tpHL)/2'
```

t2.sp

2.3 Use Logical Effort to Optimize 7x128 Decoder

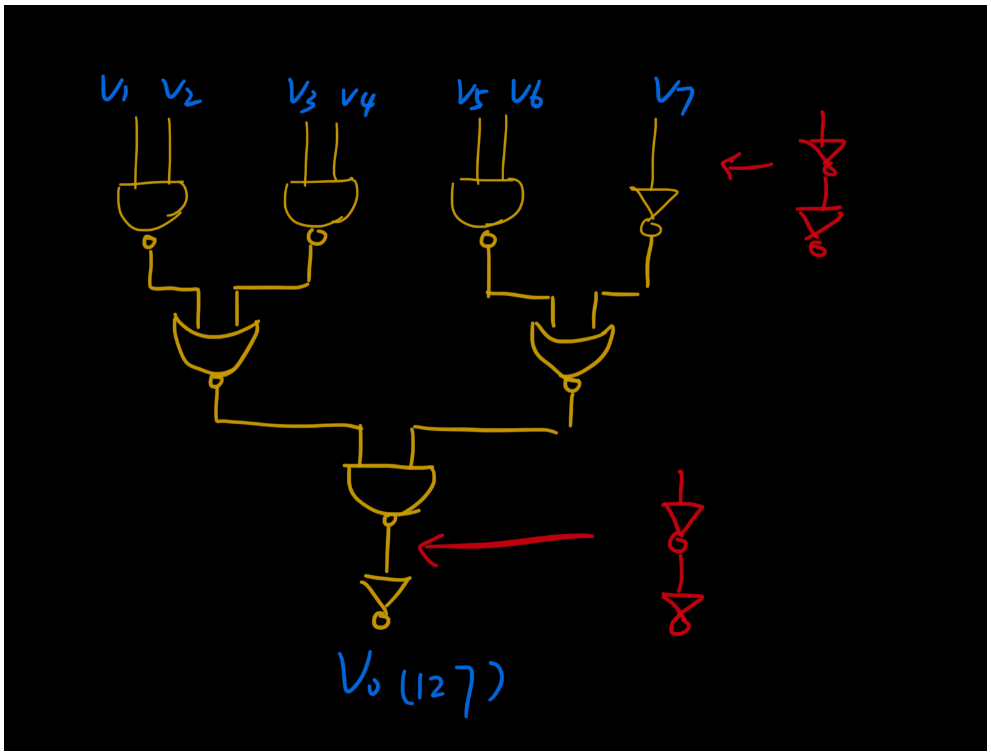
Here's the task:



- For a 7x128 decoder, first think about the scheme design. Then optimize the critical path to minimize the propagration delay of decoder
  - Use the logical effort method
  - Assume that the input inverter/buffer is minimum sized
  - Assume that each output of decoder has a load of 512X minimum sized Finfet inverter
  - You can insert buffer at the output if necessary

Critical path
A_0
A_1
A_2
A_3
A_4
A_5
A_6
7 x 128 Decoder
D_0
D_{123}
D_{124}
D_{125}
D_{126}
D_{127}
7 input
128 input

In this task, I derive the idea in task2. I scheme the logic myself, to some extent, it's instinctive that we should construct a **tree** structure to make the logic high-efficient. Here's my design:



In my diagram, it's clear that the input is 7-side and they're all contributed to a single output in 128 possible pins. We only consider the **Vout 127** to see the critical path. Obviously, the logic in the path is completely an **AND** logic. So as I show in the design figure, every input passes a NAND2 or an buffer, then pass NOR2 2 by 2. Last, they pass NAND2 gate and a buffer to ensure the whole logic path is the same as what we want.
I can also insert some buffer(s) at the beginning or at the end of the path. Be sure that every step of inserting buffer(s) needs at least 2, and for an even number.
When scheming the input, 2 buffers are necessary. They are the input which capacitance is a unit(1).
The core code is following:

```
x1b1 v1b1 v1b2 inv
x1b2 v1b2 v1 inv
x2b1 v2b1 v2b2 inv
x2b2 v2b2 v2 inv
x3b1 v3b1 v3b2 inv
x3b2 v3b2 v3 inv
x4b1 v4b1 v4b2 inv
x4b2 v4b2 v4 inv
x5b1 v5b1 v5b2 inv
x5b2 v5b2 v5 inv
x6b1 v6b1 v6b2 inv
x6b2 v6b2 v6 inv
x7b1 v7b1 v7b2 inv
x7b2 v7b2 v7 inv
//input buffers (7 groups)
x1 v1 v2 v12 NAND2 num=//opt size
x2 v3 v4 v34 NAND2 num=//opt size
```

```
x3 v5 v6 v56 NAND2 num=//opt size
x4 v7 v7i inv num=//opt size

x5 v12 v34 v1234 NOR2 num=//opt size
x6 v56 v7i v567i NOR2 num=//opt size

x7 v1234 v567i vo1 NAND2 num=//opt size

x8 vo1 vo2 inv num=//opt size

xo1 vo2 vp1 inv num=//opt size
xo2 vp1 vp2 inv num=//opt size
xo3 vp2 vp3 inv num=//opt size
xo4 vp3 vp inv num=//opt size
//ouput buffers

xload vp vout inv num=512

.meas tran tpLH
+trig v(v1b1)='0.5*supply' rise=2
+targ v(vp)='0.5*supply' rise=2
.meas tran tpHL
+trig v(v1b1)='0.5*supply' fall=2
+targ v(vp)='0.5*supply' fall=2
.measure tp param='(tpLH+tpHL)/2'
```

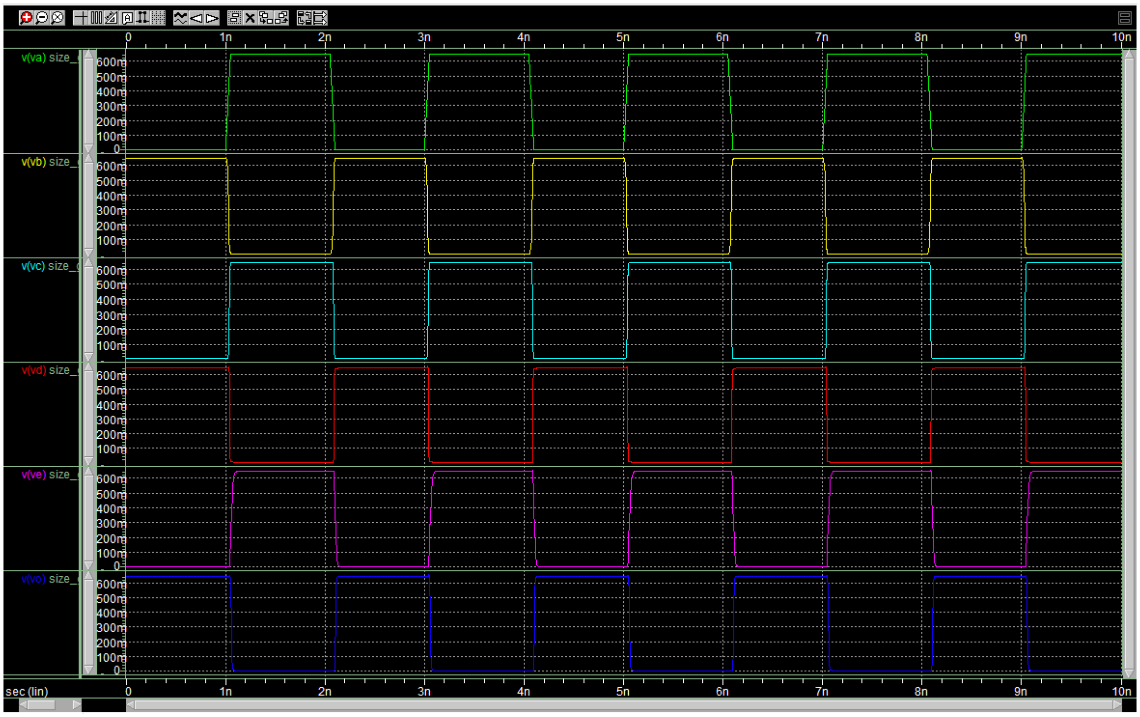
t3.sp

3 Lab Results

3.1 Use Logic Effort to Size Gates

First, let's check out the function of the circuit is correct or not? In task1, we preset NAND2's another input as 1 and NOR2's another input as 0. The merit if the action is to prevent the static output due to the logic fraction! We cannot overlap the user's input from 'a', so that's the way we can check the function of the logic is correct or not.

Then in HSPICE, the output wave are shown as below:

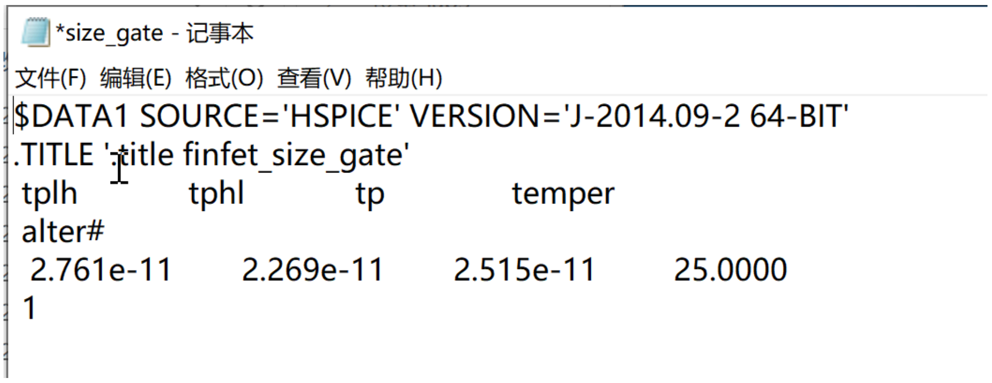


we can directly derive the function of the logic from the circuit itself, which is $e = a$, that's the same as the wave formed in the HSPCIE. We aer correct!

Due to the theory analysis, I tested 4 circumstances schedule design idea:**1.naive 2.size array:1,1,3,10 3.size array:1,2,3,10, size array:1,1,3,5**.

the results are separately like these:

for the optimal result:


*size_gate - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

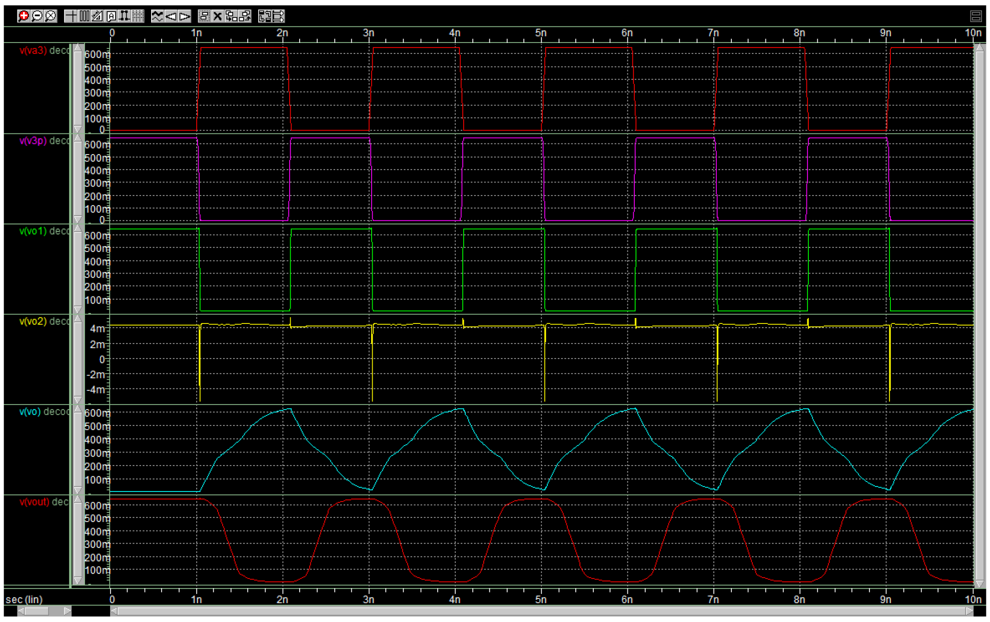
```
$DATA1 SOURCE='HSPICE' VERSION='J-2014.09-2 64-BIT'
.TITLE '.title finfet_size_gate'
 tplh        tphl        tp          temper
 alter#
  2.761e-11   2.269e-11   2.515e-11   25.0000
 1
```

conclusion:

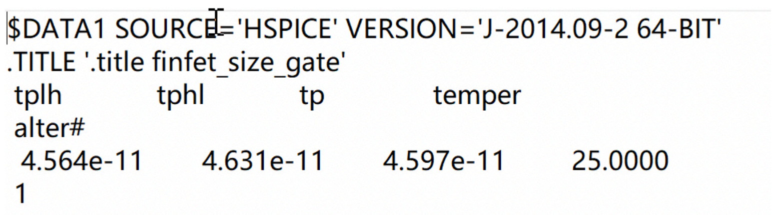
So the optimal size is 1 for NAND2, 3 for the NOR2, 5 for the inverter

3.2 Use Logic Effort to Optimize 4x16 Decoder

First, we check whether the function is correct or not. The wave is like:



From the wave, we. can derive that the logic actually perform the **AND**, vout is invesrse of the vin(va3) because I use a 256 buffer as the load. The logic function is correct.
As the optimal size, after the manual calculation, I select 4 options: **1. naive (for comparison) 2. 5times for NAND, 46times for NOR. 3. 6times for NAND, 46times for NOR 4. 7levels, 1 for NAND2, 1 for the NOR2, the 4 additional buffers are 6,14,57,227 times size**. The optimal result is as:



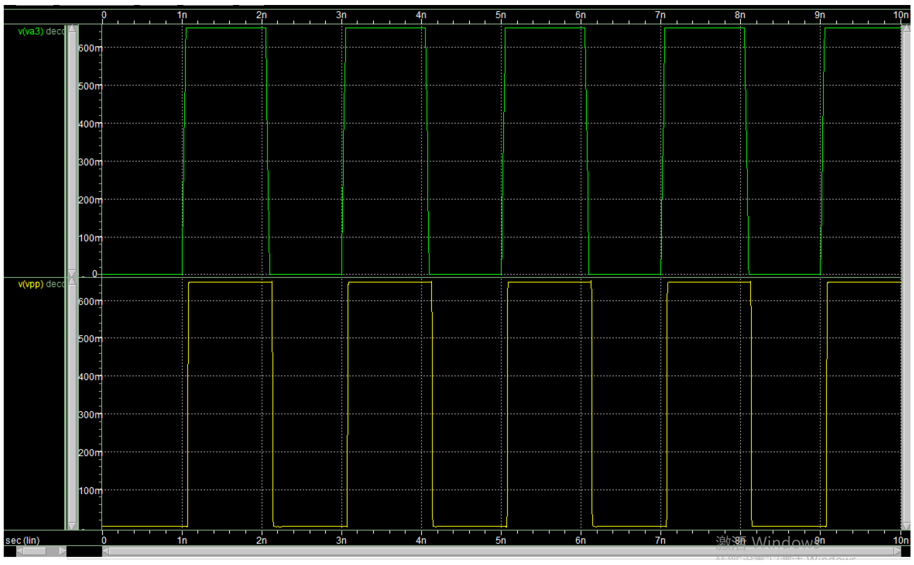
```
$DATA1 SOURCE='HSPICE' VERSION='J-2014.09-2 64-BIT'
.TITLE '.title finfet_size_gate'
 tplh          tphl          tp            temper
 alter#
  4.564e-11     4.631e-11     4.597e-11     25.0000
 1
```

conclusion:

So the optimal size is 1 for NAND2, 1 for the NOR2, the 4 additional buffers are 6,14,57,227 times size, 7 levels in total

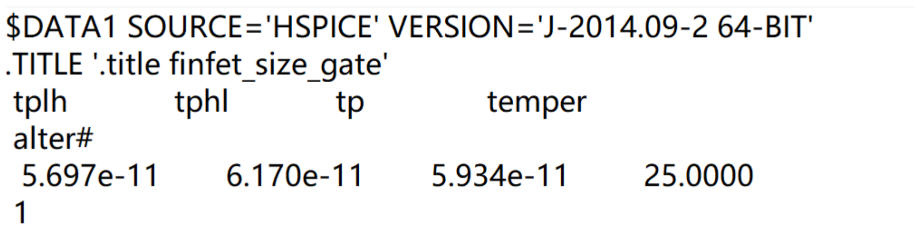
3.3 Use Logical Effort to Optimize 7x128 Decoder

As before, we evaluate the logic function first. If we only use an impulse at one single input and maintain the others as 1, the final output is OUT127 with a total AND logic.



The function is correct!

I design several options to test. And the optimal one is following:



```
$DATA1 SOURCE='HSPICE' VERSION='J-2014.09-2 64-BIT'
.TITLE '.title finfet_size_gate'
 tplh            tphl            tp              temper
 alter#
  5.697e-11       6.170e-11       5.934e-11       25.0000
 1
```

conclusion:

So the optimal size is 1 for NAND2, 2 for the seconde level inverter, 1 for the NOR2, 1 for the NAND2(4 level), 1 for the 5-level buffer, 5 fo the 6-level buffer, 19 for the 7-level buffer, 38 for the 8-level buffer, 154 for the 9-level buffer. 9 levels in total

4 Technical Analysis of the Simulation Results

4.1 task1

We can do manual calculation about it.

The $F = 10, G = (\frac{3}{2})^2, B = 1$, So, $H = GBF = 22.5, h = H^{\frac{1}{4}} = 2.17$.

So, from the last load backwards to the beginning, we get the sizes for each gate: **1,1.4,3.15,4.6. for integer: 1,1,3,5**. That's the same as the lab result.

4.2 task2

We can do manual calculation about it.

The $F = 256, G = (\frac{3}{2})^2, B = 8$, So, $H = GBF = 4608, n = log_4 H = 7(or\ 6), h = 4 = gfb$.

So, from the last load backwards to the beginning, we get the sizes for each gate: **1,1.33,0.88,3.55,14.2,56.8,227.2. for integer: 1,1,1,4,14,57,227**. That's the same as the lab result.

4.3 task3

We can do manual calculation about it.

The $F = 512, G = (\frac{3}{2})^3, B = 64$, So, $H = GBF = 110592, n = log_4 H = 8.4, h = 4,$.

So, from the last load backwards to the beginning, we get the sizes for each gate: **1,1.33(2 for the inverter),0.88,0.30,1.2,4.8,19.2,38.4,153.6, for integer: 1,1,(2 for inverter),1,1,1,5,19,38,154**. That's the same as the lab result.

5 Observations and conclusions

1. The buffers at the beginning is necessary because we should set the input load as a unit inverter. And make sure it must be equipped with 2 buffers.
2. Notice that the FInFET NAND and NOR is different from ordinary CMOS. It's logic effort is $\frac{3}{2}$ instead of $\frac{4}{3}, \frac{5}{3}$. The mirror properties can produce an excellent convenience in design.
3. In manual process, we can evaluate the H by H=GFB. Notice that if there is a **N** outputs, the B=**N/2**. F is the output/input and G is the product of all the logic efforts in each level.
4. Notice that we should put the function of a logic at the prior position. So when we're considering the buffer inserting, it must be an even number.