



# A whole new world – SPICE

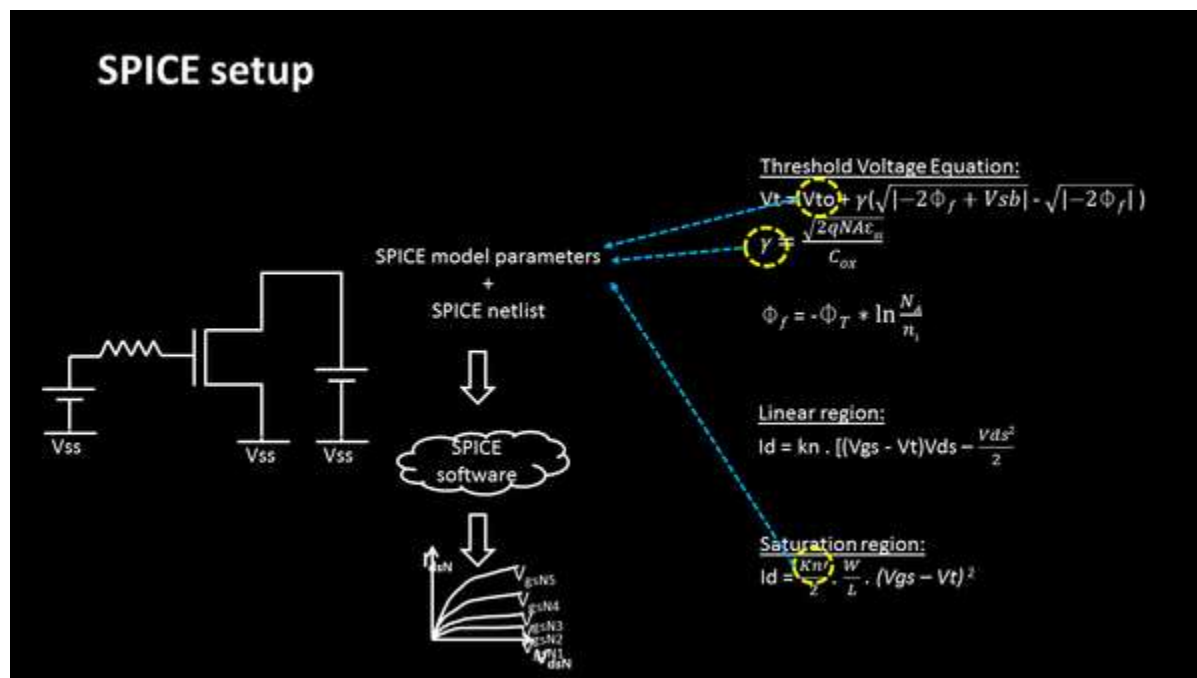
Kunal Ghosh

This was exactly the feeling when I was told to do SPICE .... but you know what, after practically applying the concepts on real design, I felt, it wasn't a new world. It was an extension towards accuracy, that guarantees your chip will not fail, in timing.

And, there it all started. I will be announcing my 5th course – “**VLSI Academy – Circuit Design and SPICE simulations**”, along with my existing 4 courses on Udemy, **which covers right from Physical Design flow to clock tree synthesis to crosstalk to parasitic to timing.**

**SPICE comes exactly after these topics. Please follow this [link](#) to get details of my existing courses**

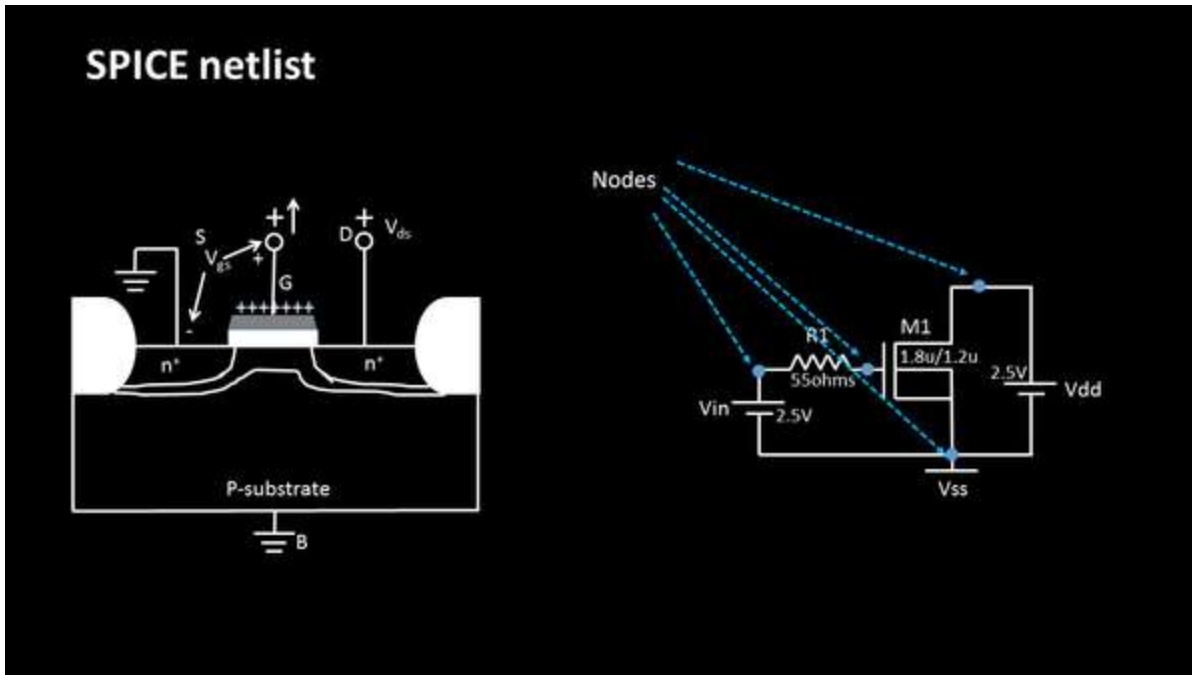
So, let me brief you what exactly we are going to do in SPICE. To start with, below image clearly shows the basic flow of SPICE



The complex equations on the **right side of the image, are not so complex**, once I go through them in my way in the courses. These equations model a MOSFET, the **left had side is one basic SPICE netlist** (In the course, we will be considering more complex one's, but I will make sure, you go through the basic one's first :))

When we provide both to SPICE engine, the output are some current waveforms (which eventually converts to a timing waveform, I will show you 'how' in the course, when I publish)

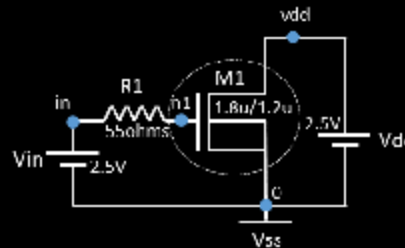
The below MOSFET device, with potential applied to drain terminal and source/substrate being grounded, can be represented by a circuit shown in right side of image. **The nodes help us to create the SPICE deck or SPICE netlist.** Stay with me to see ‘how’



Let’s write the SPICE deck for below MOSFET. The name is M1. The nodes are Vdd, n1, 0 (follow the blue dots in the image) and the syntax is “**mosfet\_name drain gate source substrate** “. When we write M1 Vdd n1 0 0, it means drain is connected node Vdd (blue dot), gate is connected to node n1, source and substrate is connected to node ‘0’. (since its ground, I named it ‘0’)

## SPICE netlist

```
M1 vdd n1 C 0 nmos w=1.8u l=1.2u
```

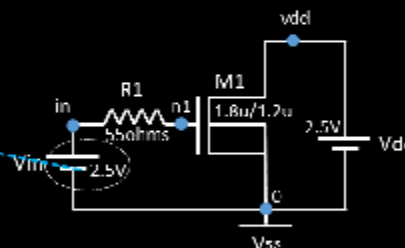


And, we do similar thing for resistor R1, input supply Vin, and main supply Vdd, like below

## SPICE netlist

```
M1 vdd n1 C 0 nmos w=1.8u l=1.2u  
R1 in n1 55  
vdd vdd 0 2.5  
Vin in 0 2.5
```

Voltage source value:



**Netlist is ready.** Now, we need to supply the model parameters, basically, the one's encircled in yellow. Once we give that, **SPICE engine internally has the equations or models that evaluates to give the output current values.** The list is huge. Maybe, I will introduce them to you in the course

## SPICE netlist Technology file

```

*** NETLIST Description ***
M1 vdd n1 0 0 nmos w=1.8u l=1.2u
R1 in n1 55
vdd vdd 0 2.5
Vin in 0 2.5

*** .include xxxx_1um_model.mod ***
.LIB "xxxx_025um_model.mod" CMOS_MODELS

```

### Threshold Voltage Equation:

$$V_t = V_{t0} + \gamma(\sqrt{|-2\Phi_f + V_{sb}|} - \sqrt{|-2\Phi_f|})$$

$$\gamma = \frac{2qN_A\epsilon_s}{C_{ox}}$$

$$\Phi_f = -\Phi_j + \ln \frac{N_A}{n_i}$$

### Linear region:

$$I_d = k_n \cdot [(V_{gs} - V_t)V_{ds} - \frac{V_{ds}^2}{2}]$$

### Saturation region:

$$I_d = \frac{k_n'}{2} \cdot \frac{W}{L} \cdot (V_{gs} - V_t)^2$$

Once, you have them, **pack all the technology related things, into a file, say, .mod file and point to that file, something like below.** This helps maintenance of the files, as, we write SPICE netlist for complex gates.

## SPICE netlist Technology file

```

M1 vdd n1 0 0 nmos w=1.8u l=1.2u
R1 in n1 55
vdd vdd 0 2.5
Vin in 0 2.5

```

```

.lib cmos_models
.MODEL nmos NMOS (TOX = ..
  VTH0 = .. 1.0 = .. GAMMA1 = ..)
.MODEL pmos PMOS (TOX = ..
  VTH0 = .. 1.0 = .. GAMMA1 = ..)
.end

```



Package in a file

xxxx\_025um\_model.mod

### Threshold Voltage Equation:

$$V_t = V_{t0} + \gamma(\sqrt{|-2\Phi_f + V_{sb}|} - \sqrt{|-2\Phi_f|})$$

$$\gamma = \frac{2qN_A\epsilon_s}{C_{ox}}$$

$$\Phi_f = -\Phi_j + \ln \frac{N_A}{n_i}$$

### Linear region:

$$I_d = k_n \cdot [(V_{gs} - V_t)V_{ds} - \frac{V_{ds}^2}{2}]$$

### Saturation region:

$$I_d = \frac{k_n'}{2} \cdot \frac{W}{L} \cdot (V_{gs} - V_t)^2$$

Now, we provide **circuit + technology file to SPICE engine along with below simulation commands ....** (sentence continued after this image)

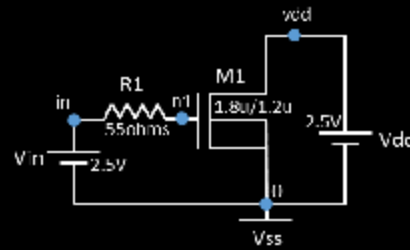
## SPICE netlist Simulation Commands

```

*** NETLIST Description ***
M1 vdd n1 0 0 nmos W=1.8u L=1.2u
R1 in n1 55
vdd vdd 0 2.5
Vin in 0 2.5

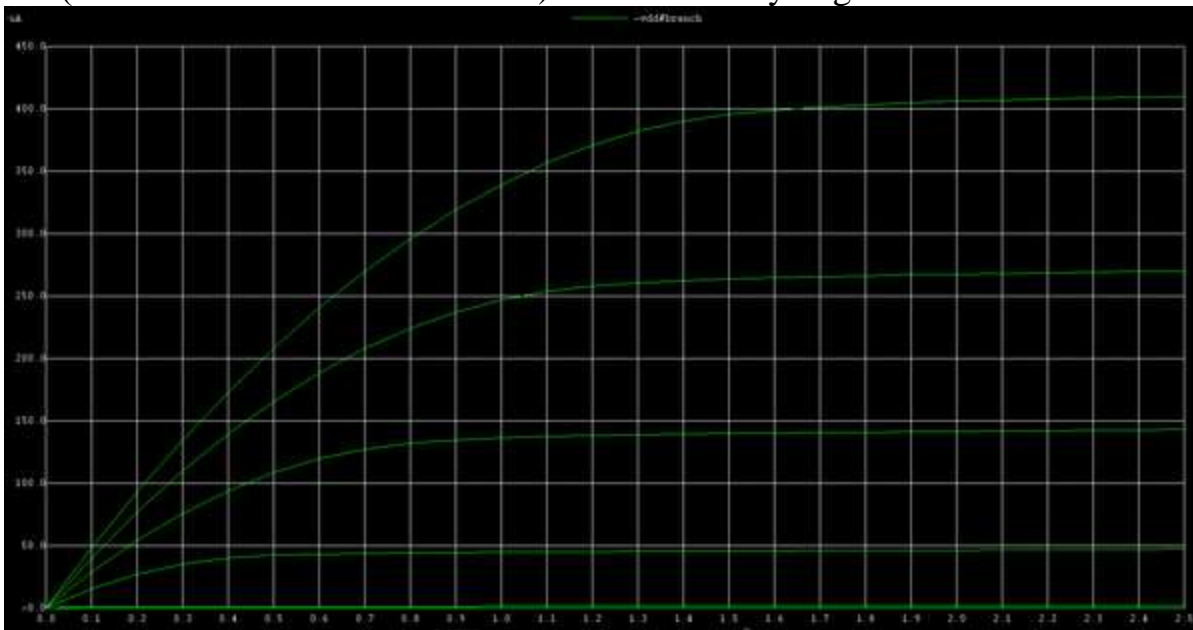
*** .include xxxx_1um_model.mod ***
.LIB "xxxx_025um_model.mod" CMOS_MODELS
*** SIMULATION Commands ***
.top
.dc Vdd 0 2.5 0.1 Vin 0 2.5
.end

```



$V_{gs} = 2.5V$   
 $V_{ds} = \text{sweep from } 0 \text{ to } 2.5$

.... (sentence continued from above) and this what you get

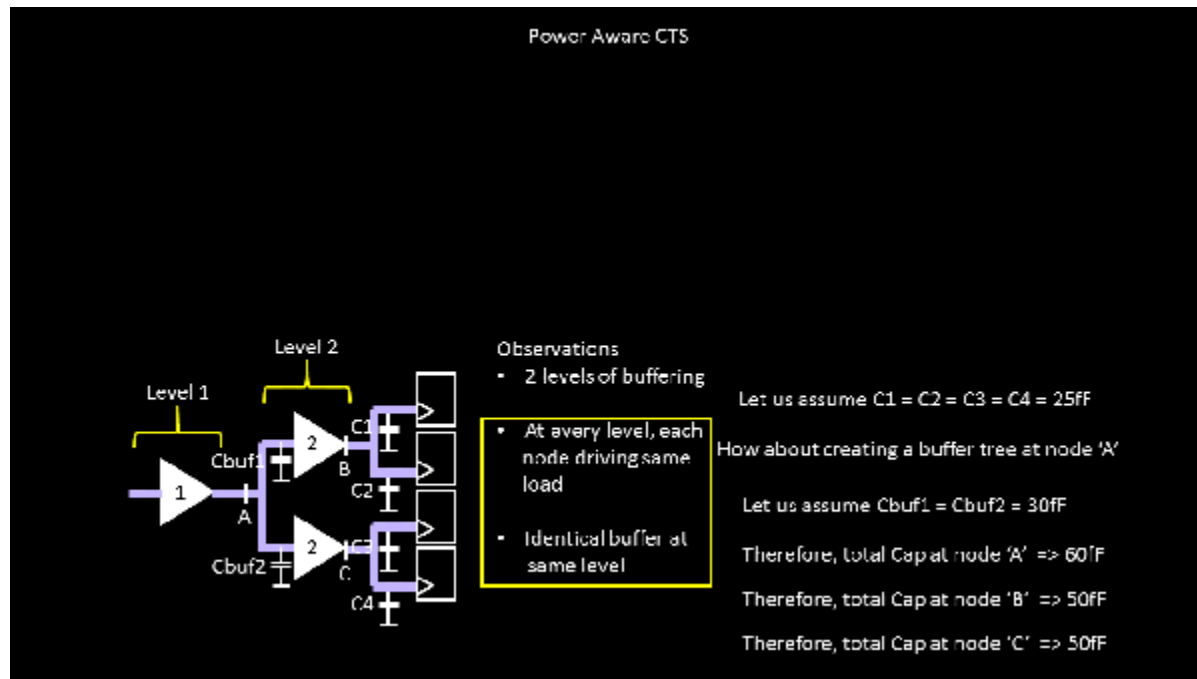


Does that interest you? :). If yes, I will send you the link to my new course very soon. Be ready for another adventure.

I was overwhelmed by the very common question being asked to me after my previous post, and these were from all levels of people (even people from verification and embedded background). “Why do we need to learn SPICE?”

And, that was exactly what I was expecting. “For the things, *we must learn before we can do them, we learn by doing them*”. Though, this will be discussed in detail in my **upcoming course of SPICE**, let me try to give a glimpse of it here itself

Below is a snippet from my existing course on [Clock Tree Synthesis](#)



It has 2 kinds of clock buffers, say ‘1’ and ‘2’. Now, to calculate the delay of these buffers, the only input, we have been delay models. **These could be “non-linear delay models” or “constant-current source models”.**

Let’s take a simple example of below “**non-linear delay model.**” This is how it looks. It has a “**input slew**” one side and output capacitance on other side.

A more complex model (to be discussed separately), and time units on one side and normalized voltage on other side.

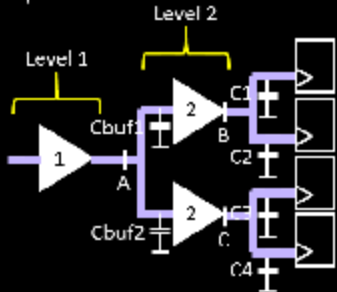
Power Aware CTS

Delay Table for CBUF '1'

Input slew	Output load					
	10fF	30fF	50fF	70fF	90fF	110fF
20ps	x1	x2	x3	x4	x5	x6
40ps	x7	x8	x9	x10	x11	x12
60ps	x13	x14	x15	x16	x17	x18
80ps	x19	x20	x21	x22	x23	x24

Delay Table for CBUF '2'

Input slew	Output load					
	10fF	30fF	50fF	70fF	90fF	110fF
20ps	y1	y2	y3	y4	y5	y6
40ps	y7	y8	y9	y10	y11	y12
60ps	y13	y14	y15	y16	y17	y18
80ps	y19	y20	y21	y22	y23	y24



Observations

- 2 levels of buffering

- At every level, each node driving same load
- Identical buffer at same level

Let us assume  $C1 = C2 = C3 = C4 = 25fF$

How about creating a buffer tree at node 'A'?

Let us assume  $Cbuf1 = Cbuf2 = 30fF$

Therefore, total Cap at node 'A' => 60fF

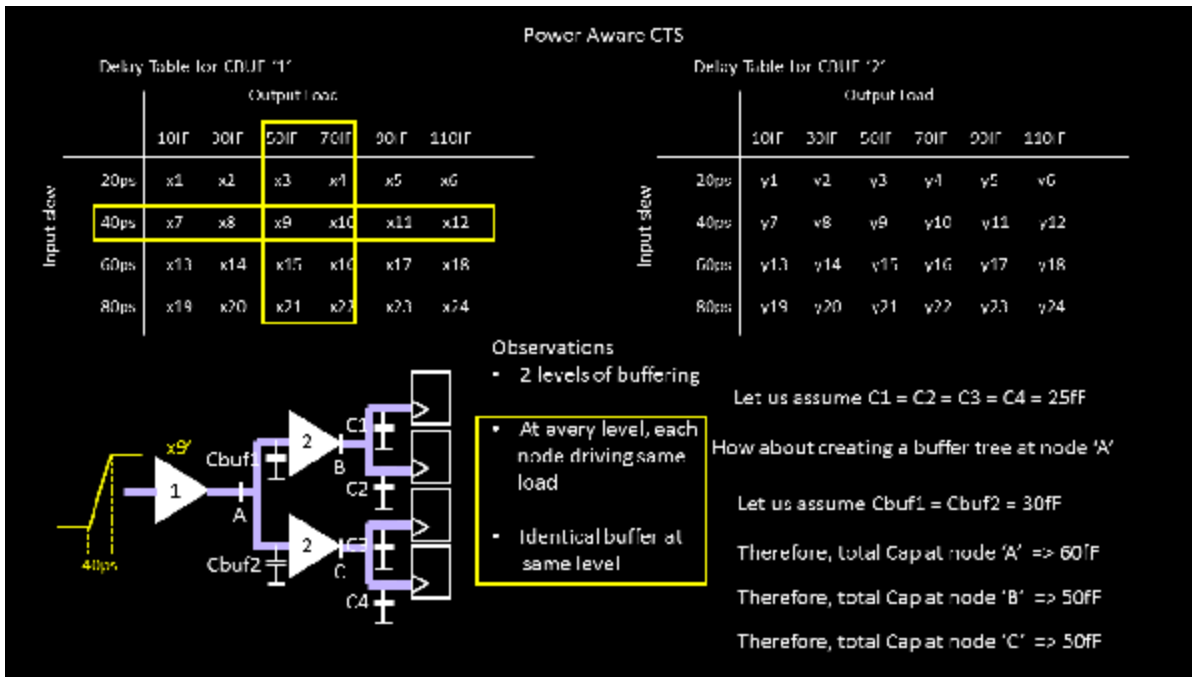
Therefore, total Cap at node 'B' => 50fF

Therefore, total Cap at node 'C' => 50fF

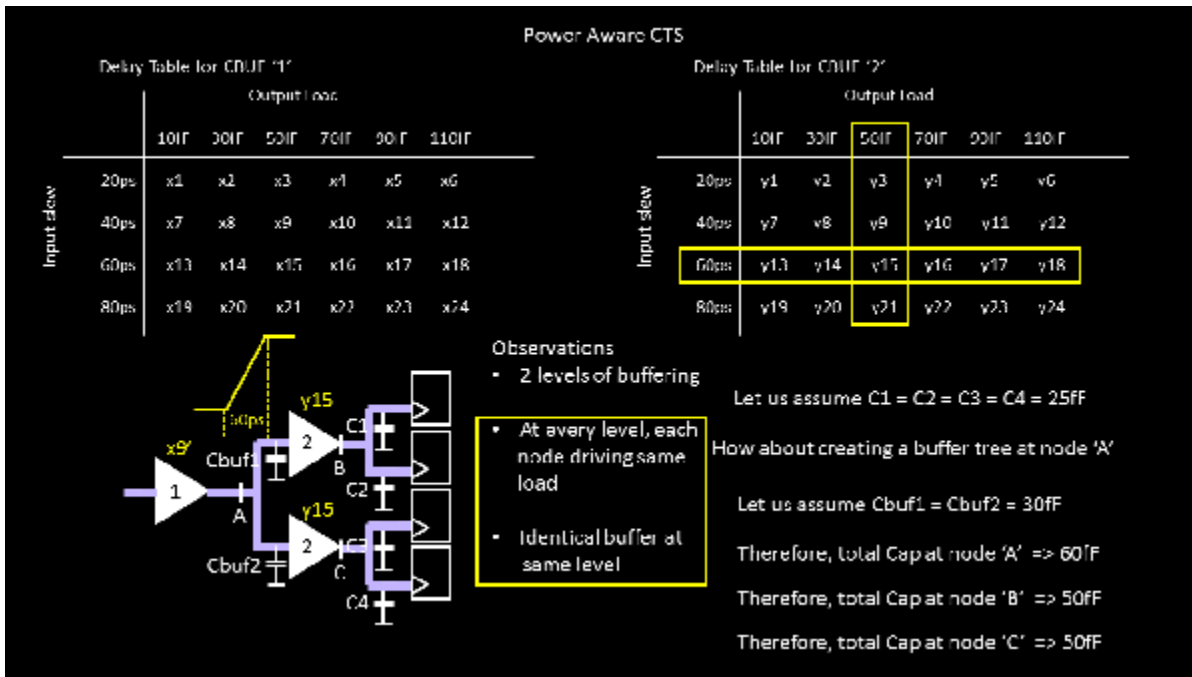
**Both buffers can be of different type (or different sizes and drive strength), so each buffer '1' and '2' has their own table.**

Now, say, output capacitance of buffer '1' is 60fF and input slew is 'say' 40ps, below is the section of the table that we need to consider computing its delay. **This technique is called 'interpolation'**. We need to interpolate the delay value between 50fF and 70fF, for an output cap of 60fF



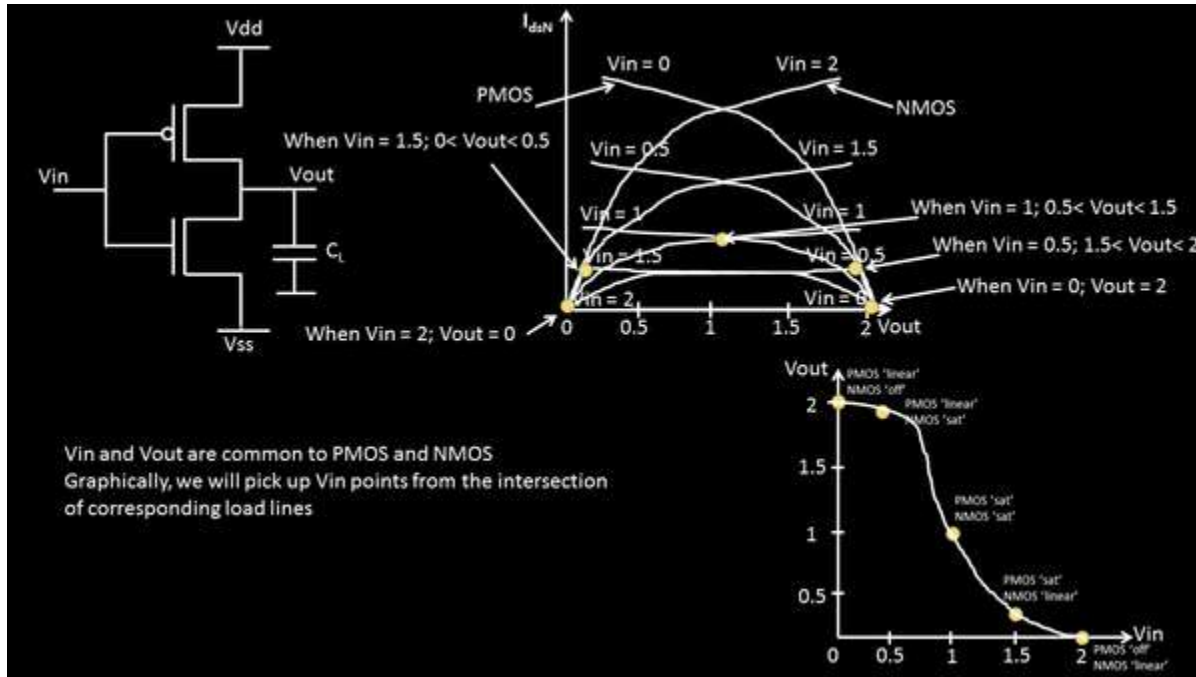


Similarly, for an input slew of 'say' 60ps and output load of '50fF', the delay value of buffer '2' will be about 'y15', as per below table



I haven't yet answered your question. **Why do we need to learn SPICE?** I will answer it now. **Where, do you think, the delay values in the table come from?**

Let's say we have this below inverter (simple buffer is 2 inverters connected back to back), with the below IO characteristics.



Now, observe very carefully to all above waveforms (specially the last one present in bottom right). Do they remind you of look-alike waveforms, for which we just calculated the delay?

Does the waveform at the top-middle, remind you of the waveforms we derived in last post using a **SPICE simulator**? Do they look like **Id-Vds curves for NMOS**?

**Do you relate how delay of cells are derived from NMOS-PMOS-CMOS transfer characteristics?**

I feel, learning or working in VLSI domain without transistor model knowledge is like flying a helicopter with low fuel in a deserted region, not knowing where the hell you will land up.

Here how the course goes. I start with asking few questions, on why do we need SPICE

Power Aware CTS

Delay Table for COIIF '1'

Input slew	Output load					
	10fF	30fF	50fF	70fF	90fF	110fF
20ps	x1	x2	x3	x4	x5	x6
40ps	x7	x8	x9	x10	x11	x12
60ps	x13	x14	x15	x16	x17	x18
80ps	x19	x20	x21	x22	x23	x24

Delay Table for COIIF '2'

Input slew	Output load					
	10fF	30fF	50fF	70fF	90fF	110fF
20ps	y1	y2	y3	y4	y5	y6
40ps	y7	y8	y9	y10	y11	y12
60ps	y13	y14	y15	y16	y17	y18
80ps	y19	y20	y21	y22	y23	y24

- "Did you know, where does the delay of a cell actually comes from?"
- "We have learnt about delay models, but are the models accurate?"
- "How do you verify, if what you are doing in static timing analysis, is correct?"

<http://www.mcsys.com/>

Then, I introduce and (literally) derive models for threshold voltage and .... (sentence continued after below image)

**Threshold Voltage Equation:**  

$$Vt = Vt0 + \gamma(\sqrt{|-2\phi_f + Vsb|} - \sqrt{|-2\phi_f|})$$
 Where  
 $Vt0$  = Threshold voltage at  $Vsb = 0$ , and is a function of manufacturing process  
 $\gamma$  = body effect coefficient, expresses the impact of changes in body bias  $Vsb$  (Unit is  $V^{0.5}$ )  
 $\phi_f$  = Fermi Potential (Covered in Semiconductor Physics Slides)  

$$\gamma = \frac{\sqrt{2qNA_s\epsilon_{si}}}{C_{ox}}$$
 $\epsilon_{si}$  = relative permittivity of silicon = 11.7  
 $N_A$  = doping concentration  
 $q$  = charge of the electron  
 $C_{ox}$  = oxide capacitance  

$$\phi_f = -\phi_T + \ln \frac{N_A}{n_i}$$
 $n_i$  = intrinsic doping parameter for the substrate

$Vsb = +ve$  value

Semiconductor surface inverts to n type material at voltage  $Vgs = Vt0 + V1$

..... (sentence continued from above) drain current models for resistive and saturation region of operation

**First order analysis:**

$$Q_i(x) \propto -([V_{gs} - V(x)] - V_t)$$

i.e.,

$$Q_i(x) = -C_{ox} [(V_{gs} - V(x)) - V_t]$$

$$I_d = -V_n(x) \cdot Q_i(x) \cdot W$$

$$= \mu_n \cdot dV/dx \cdot C_{ox} [(V_{gs} - V(x)) - V_t] \cdot W$$

$$I_d \cdot dx = \mu_n \cdot C_{ox} \cdot W [(V_{gs} - V(x)) - V_t] \cdot dV$$

Integrate over Length 'L' on LHS and over drain-source voltage  $V_{ds}$  on RHS, we get the below

$$I_d = \mu_n \cdot C_{ox} \cdot (W/L) [(V_{gs} - V_t)V_{ds} - V_{ds}^2/2]$$

$$I_d = k_n' \cdot (W/L) [(V_{gs} - V_t)V_{ds} - V_{ds}^2/2]$$

Where  $k_n'$  = process transconductance

$$I_d = k_n \cdot [(V_{gs} - V_t)V_{ds} - V_{ds}^2/2]$$

Where  $k_n = k_n' \cdot (W/L)$  = gain factor

Top View  
<http://vlsi.systemdesign.com/>

Effective conductive channel length is modulated by applied  $V_{ds}$   
 $V_{ds} \uparrow$  Depletion region at drain  $\uparrow$   
 Effective channel length  $\downarrow$

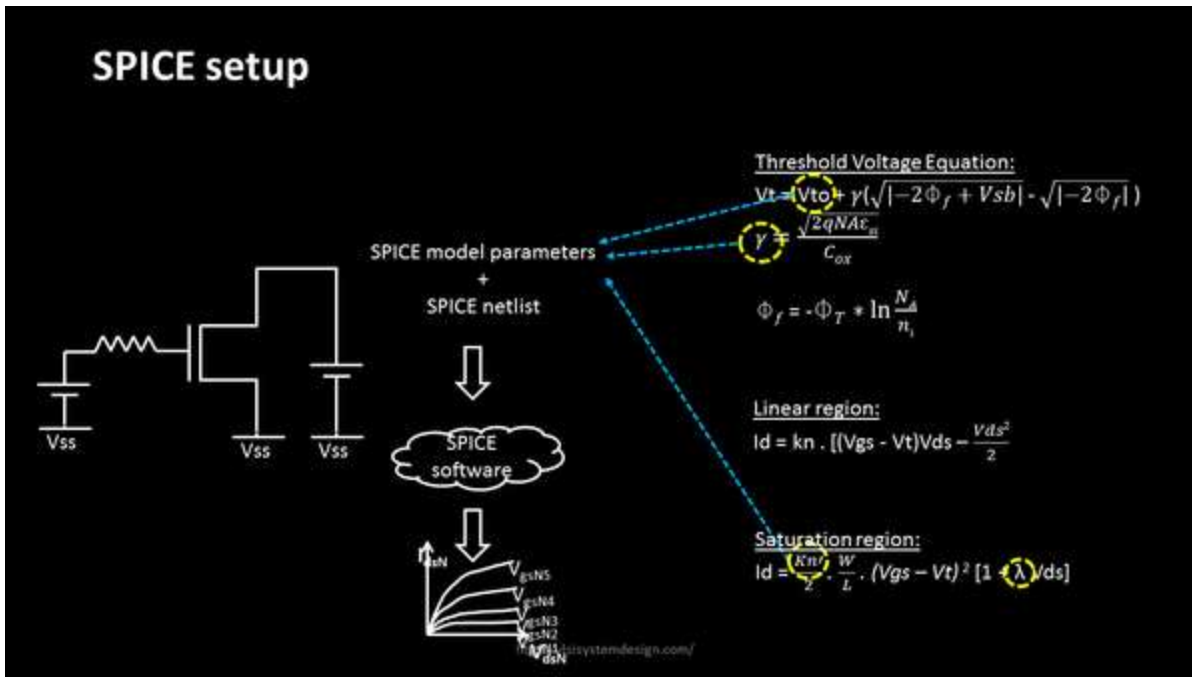
Looks like perfect current source  
 i.e current is constant.  
 Not 'correct'

$$I_d = \frac{k_n'}{2} \cdot \frac{W}{L} \cdot (V_{gs} - V_t)^2 [1 + \lambda V_{ds}]$$

More accurate equation, and  $\lambda$  = channel length modulation

<http://vlsi.systemdesign.com/>

Eventually, I end up, developing the SPICE setup, and from there on-wards, there is no stopping us from reaching the final goal, where we derive delays for cells.



There's difference in 'Online SPICE simulator' and 'Online SPICE simulations'. I can happily say that, I was able to achieve the latter, through my **online videos on Circuit design**.

Let me think, and get back on how do I do the 'former'.

Don't you think, it would be great, that while you learn basics on **Circuit Design & SPICE simulations**, parallel you should even be able to simulate it 'Online', without needing to download any simulator on your PC?

That's the 'VISION' I have, and let's see, if I will be able to achieve it.

For now, the first step towards this vision, has been achieved. Look for yourself in the below demo video, how am I doing this

<https://youtu.be/KIaAEhpC-ak>

**Happy Learning!!**