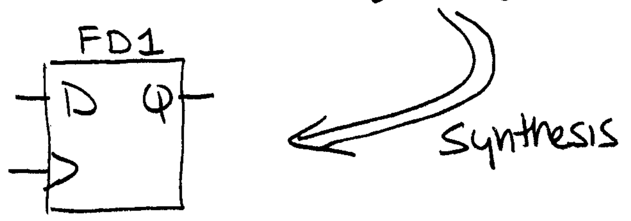
VHDL SYNTHESIS

* Synthesis takes an HDL model and maps it to a specific gate implementation.

```
IF( clk'EVENT AND clk='1') THEN q<=d; END IF;
```



* The input to the synthesis tool is HDL code, the output is either an HDL netlist (structural VHDL) or an EDIF netlist. (Electronic Design Interchange Format)

* Synthesis is a central link in top-down design.
- It is the most effective means of generating circuits
- 10-100x faster than drawing schematics
- very flexible to changes in design
- retargetable to new feature sizes or base technology

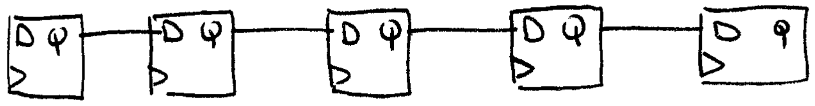
  $0.5\mu m \rightarrow 0.35\mu m \rightarrow 0.18\mu m$; CMOS, GaAs, ECL

  only the "backend" changes

Synthesis consists of two operations
1) Translation (Analyze, elaborate)
2) Optimization

Translation
- convert HDL to unoptimized netlist
- uses generic gates (17 input AND, 32 input OR, etc.)
- steps
  1. initial code processing
     - subprograms are in-line expanded
     - constant unfolding $a+3+5 \Rightarrow a+8$
     - loops unrolled

```
FOR i IN 0 TO 14 LOOP
    d <= q(i+1)
END LOOP;
```



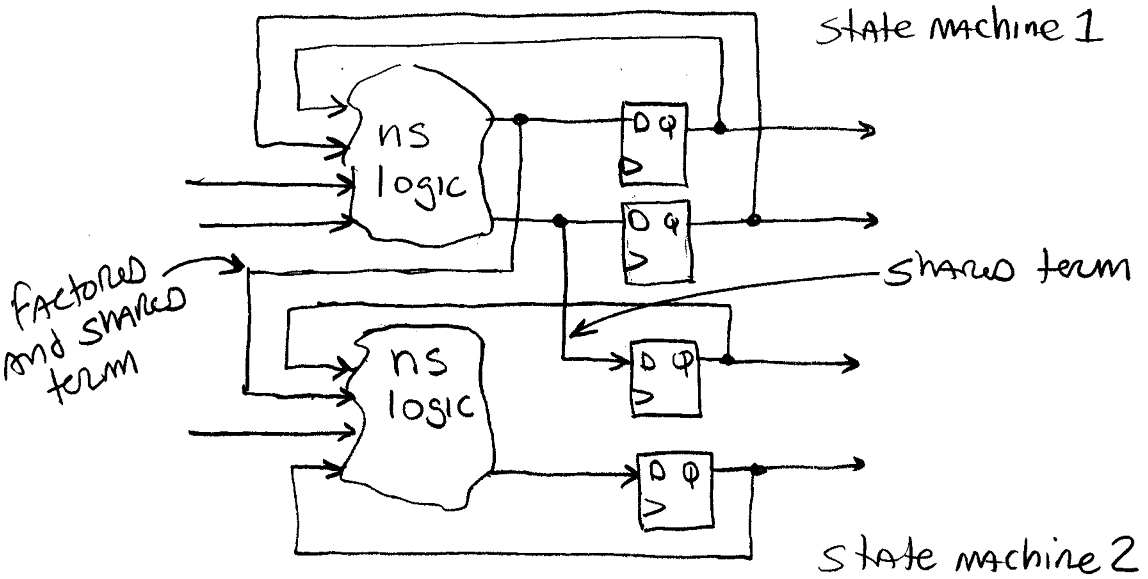
     - state encoding ; Assign vectors to enum. states
       binary, one hot, grey code ...

  2. Conversion to netlist
     - map language structures to generic gates
     - connect gates with internal format

Optimization

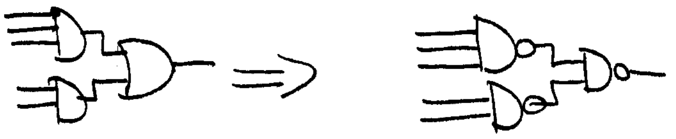
- Steps

1. Combinatorial logic optimized
   - logic minimization via Quine McCluskey
   - equation flattening (2 level realization)
   - factorization (term sharing) (Powerful technique)



State machine 1 and 2 share a term and a factored term

2. Mapping to target technology
   - map generic gates to a "best-fit" set of target gates
   - uses info from cell library timing files
     - cell delay (intrinsic, slew, loading)
     - wire table (estimate wire load)
   - Runs many trials to find the "best" combination for area or speed or power



Optimization is guided by constraints
- usually speed sets the constraint
- could be speed, area, or power

Constraints are the desired circuit characteristics or goals that must be met for the circuit to operate correctly.

Different Constraints → Different Circuits but Same Function

Operating Conditions set the <u>Global Constraints</u>
- process, voltage, temperature

AMI05_typ
-slow
-fast

For any cell: $\text{Delay} = K_P * K_T * K_V * T_{Pd}(\text{typ})$
$= K_F * T_{Pd}(\text{typ})$

K_P = process factor
K_T = temperature factor
K_V = voltage factor
K_f = max delay factor
$T_{Pd}(\text{typ})$ = typical delay from data book

From best case to worst, CMOS delay varies 4:1!

Toshiba slide, AMI slide

AMI SEMICONDUCTORS
n Microsystems, Inc.

ons using the formula:

ent; K_{VDC}, the DC voltage
ue to the ESD protection

ble manufacturing of the
s the "Target" fabrication,

All P-Channel (Voh = 2.4V)	
TYP	WCP
1.00	1.45

hannel 2.4V)	
0	5.5
00	1.21

Delay Derating Information

The propagation delays listed in the data sheets are for typical temperature, 25°C; typical supply voltage, 5.0V; and typical processing conditions. To calculate the delay at other conditions (including V_{DD} equals 3.0V) the following equation can be used:

$$T_{pdx} = T_{pdx}(typ) * K_P * K_V * K_T$$

where T_{pdx}(typ) is given in the data sheets. K_P, the process derating coefficient; K_T, the temperature derating coefficient; and K_V, the supply voltage derating coefficient, are described below.

Delay Variations with Temperature (K_T)

Delay varies linearly with temperature. The following formulas and common operating points can be used.

Temp	K_T
-55°C	0.79
-25°C	0.87
0°C	0.94
25°C	1.00
70°C	1.11
100°C	1.19
125°C	1.26

Temp. Range	K_T Formula
-55°C to 25°C	$K_T = 1.0 - (25 - T_J°C) * 2.58 \times 10^{-3}$
25°C to 140°C	$K_T = 1.0 + (T_J°C - 25) * 2.58 \times 10^{-3}$

Where T_J°C is the temperature at the silicon junction.

$(K_T * K_V * K_P)$ min typ max
.45 1 3.06

Delay Variations with Process (K_P)

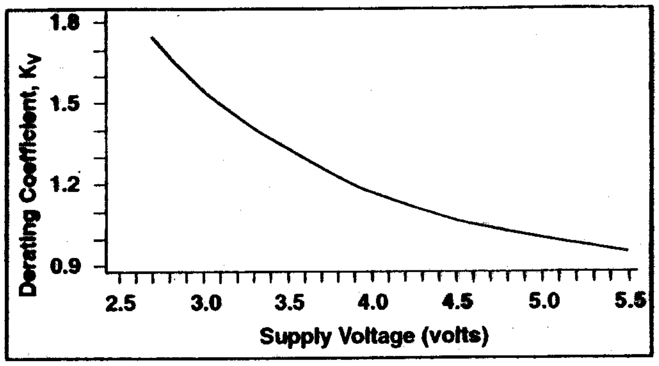
Delay variations with process are given as fixed constants determined at the limits of acceptable manufacturing of the process. These are described below.

Derating Coefficient (K_P)	Process Variation Point
1.40	Delay increase due to "Worst Case Speed" (WCS) fabrication
1.00	Typical delay; Fabrication target
0.61	Delay reduction due to "Worst Case Power" (WCP) fabrication

Delay Variations with Voltage (K_V)

Delay varies nonlinearly with voltage. Some common operating points and a characteristic curve are shown.

V_{DD}	K_V
2.7V	1.74
3.0V	1.54
3.3V	1.39
4.5V	1.07
4.75V	1.03
5.0V	1.00
5.25V	0.97
5.5V	0.94



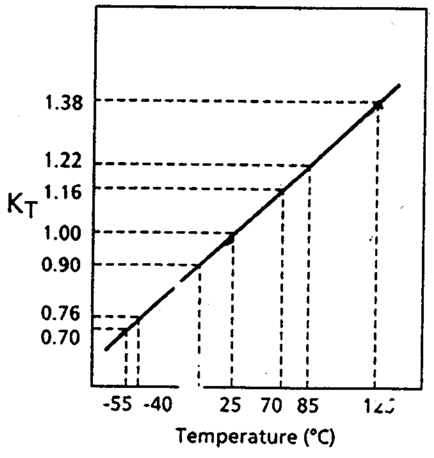


Figure1 K_T vs Temperature

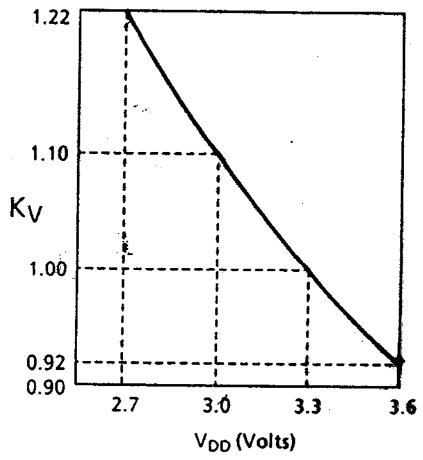


Figure2 K_V vs Supply Voltage

Note In this databook, performance information for a TC180G series macrocell is provided for nominal conditions. (Ta = 25°C, VDD = 3.3V, and typical process)

VDD	K_F (Ta = 0 °C~70 °C)		
	Best	Typ.	**Worst**
3.3V ± 3V	0.50	1.00	**1.86**
3.0V ± 3V	0.54	(1.10)	**2.07**

Table1 Maximum delay factor (K_F)

For example : AMI 0.8 μm <u>worst case</u>

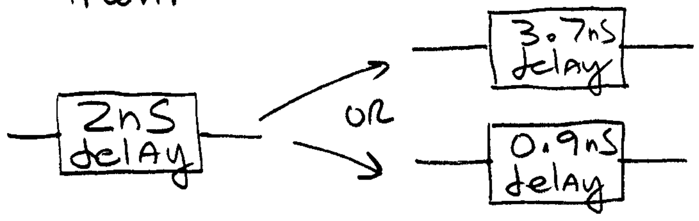
$$K_F = \underset{\substack{K_T \\ +125\,°C}}{1.26} * \underset{\substack{K_P \\ \text{process}}}{1.40} * \underset{\substack{K_V\,@ \\ 4.5V}}{1.07} = 1.89$$

more than 4:1 difference in delay

best case:

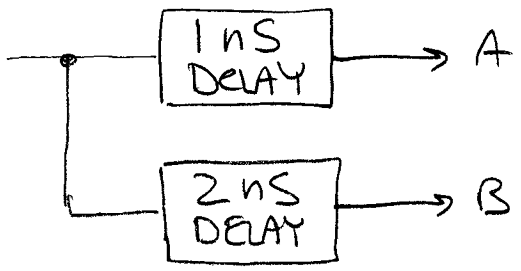
$$K_F = \underset{-55\,°C}{0.79} * \underset{\text{process}}{0.61} * \underset{5.5V}{0.94} = 0.45$$

This is why we <u>Almost</u> never use delay cells on chip. They do exist, but vendors will question your use of them.



Could be either you don't know

What about this?



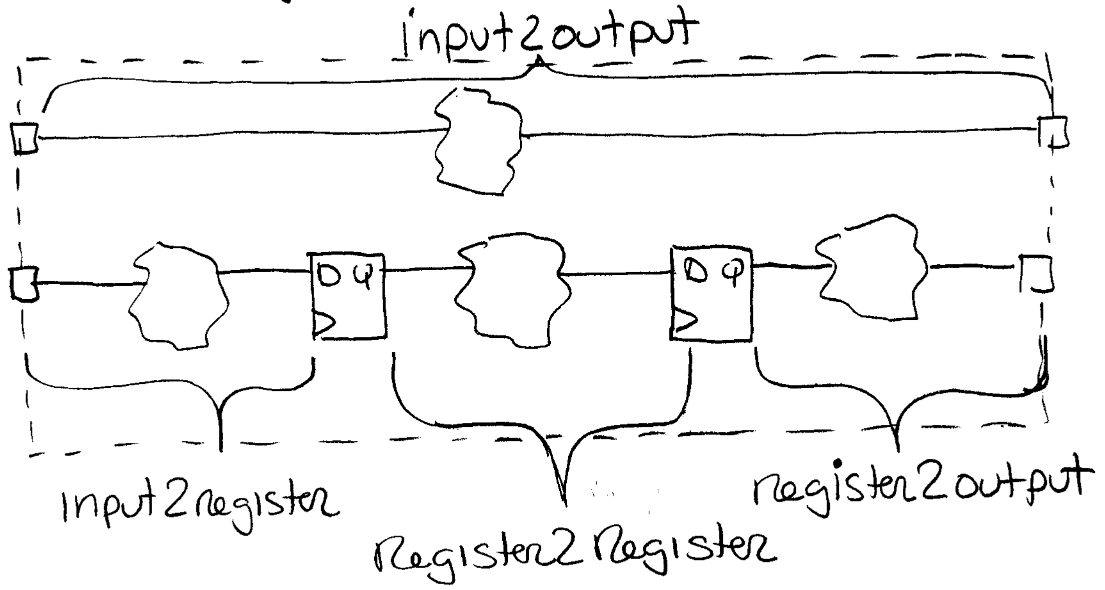
is this ok?
will A always lead B?

Delay cells used for generating hold time and little else.

Synthesis Constraints (contd)

Circuit specific Constraints

1. Area - minimize # gates or area
2. Timing (speed)
   - 4 major parts to this



The optimizer works on the logic clouds to meet timing and area requirements.

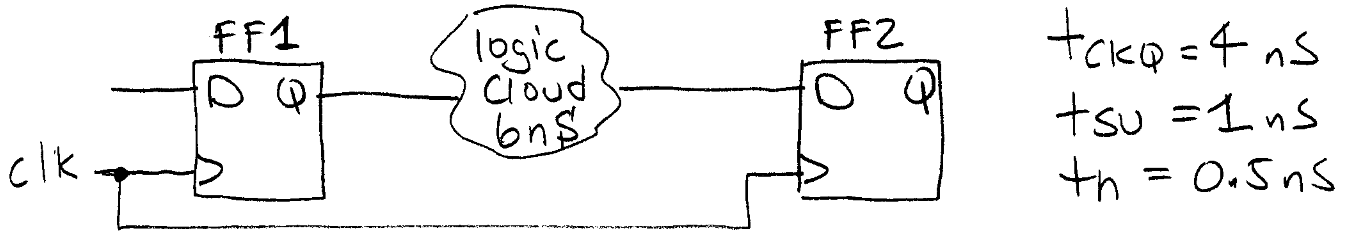
```
In elsyn script:  set register2register  20
                  set input2register     10
                  set register2output    10
                  set input2output       20
```

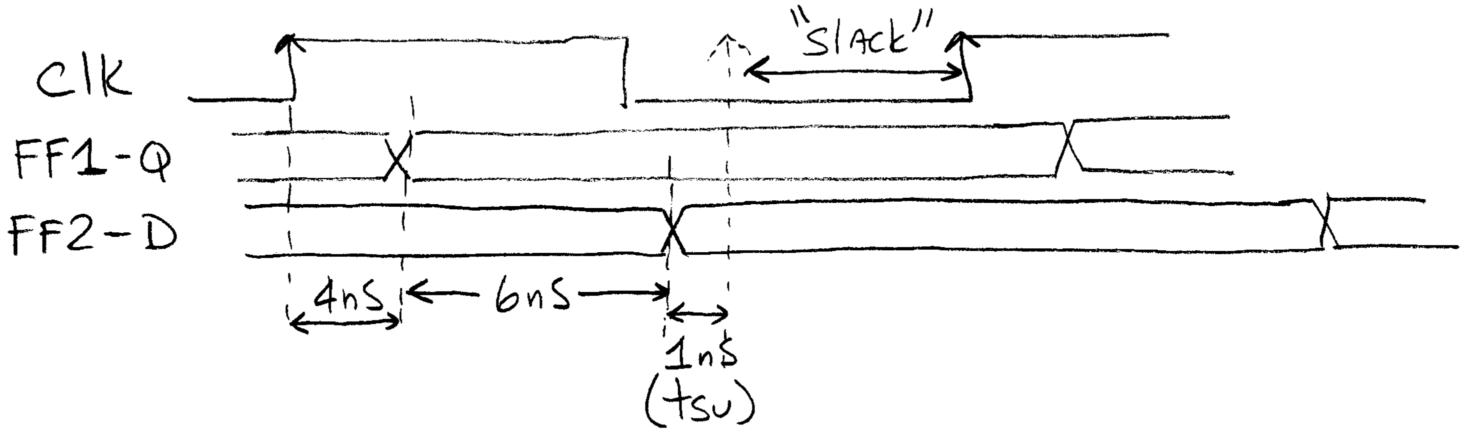
All these constraints except input2output are set relative to some clock. In fact, simply defining clock period will cause register2register optimization.

Register to Register Constraints

- Tells how much the synthesis tool must reduce delay between F/Fs.
- Optimizer can reduce logic depth, loading, rearrange state machine states, and with physical synthesis, change physical locations of cells
- Optimization is mostly about reducing logic delay so that clock cycle time is met.

- simple example





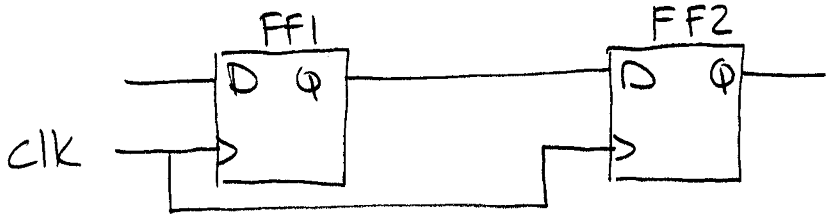
$t_{cycle(min)} = t_{CKQ} + t_{pd} + t_{su}$ (no clock skew)

slack $= t_{cycle} - t_{CKQ} - t_{pd} - t_{su}$ (slack must be $> \emptyset$ for ckt to work!)

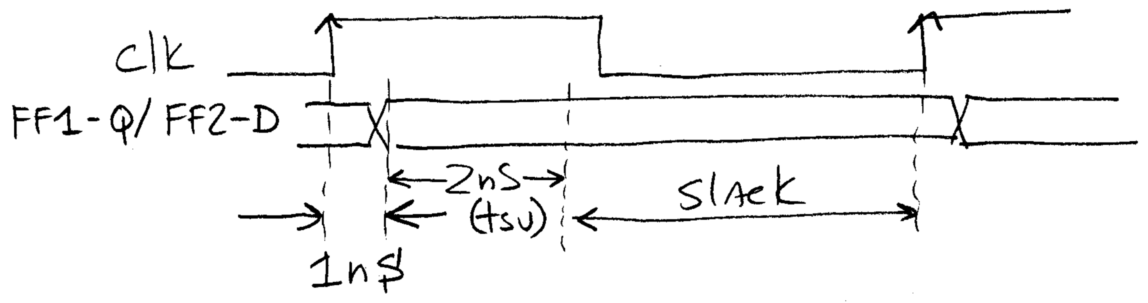
for this example $t_{cycle(min)} = 4 + 6 + 1 = \underline{\underline{11nS}}$

what would the slack be with a 15nS cycle time?

Optimizer must Also make sure hold time is met.



$t_{ckQ} = 1\text{nS}$
$t_{su} = 2\text{nS}$
$t_h = 0.5\text{nS}$
$t_{cycle} = 11\text{nS}$



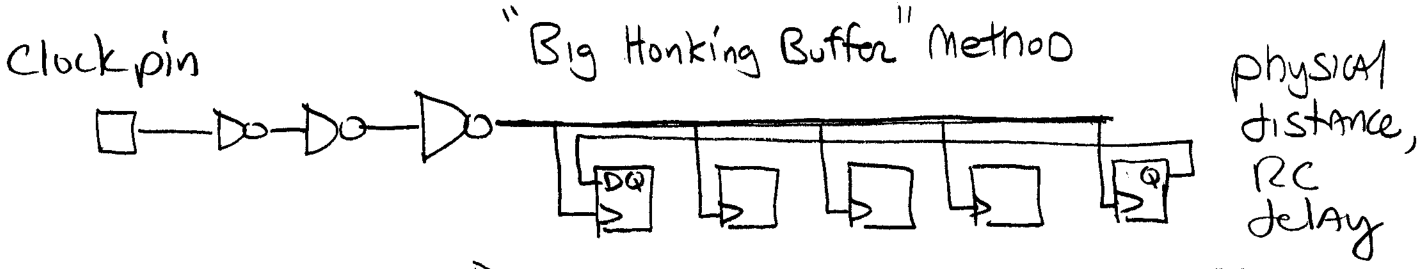
what is the slack?
is the hold time requirement met?

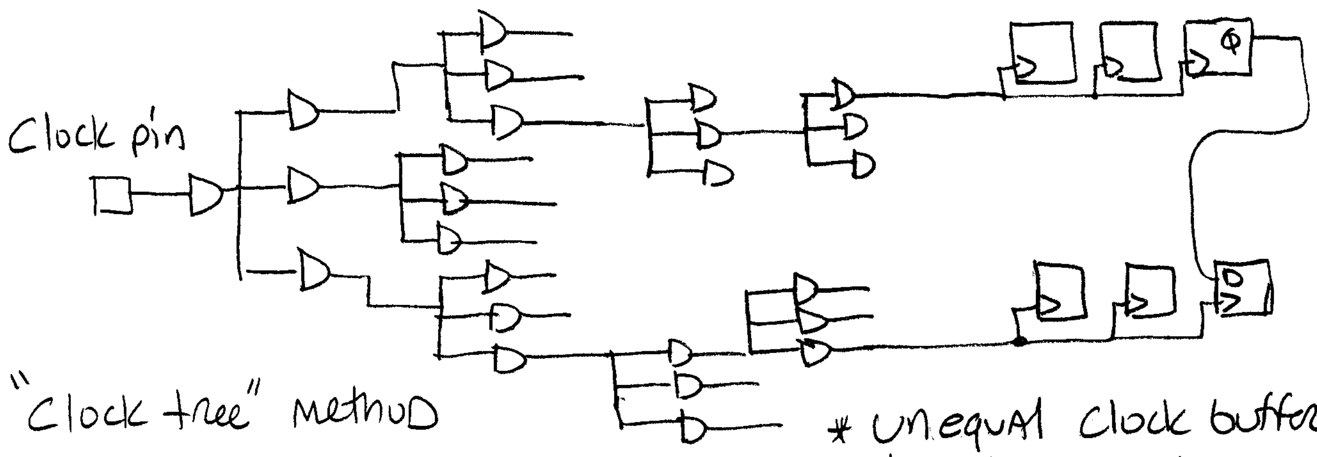
$t_{hold} \approx t_{ckQ} + t_{pd} \pm \text{skew}$

* Slowing the clock down helps slack for tsu
* the clock speed has no effect on hold time.

How does clock skew effect synthesis?

How does clock skew happen?



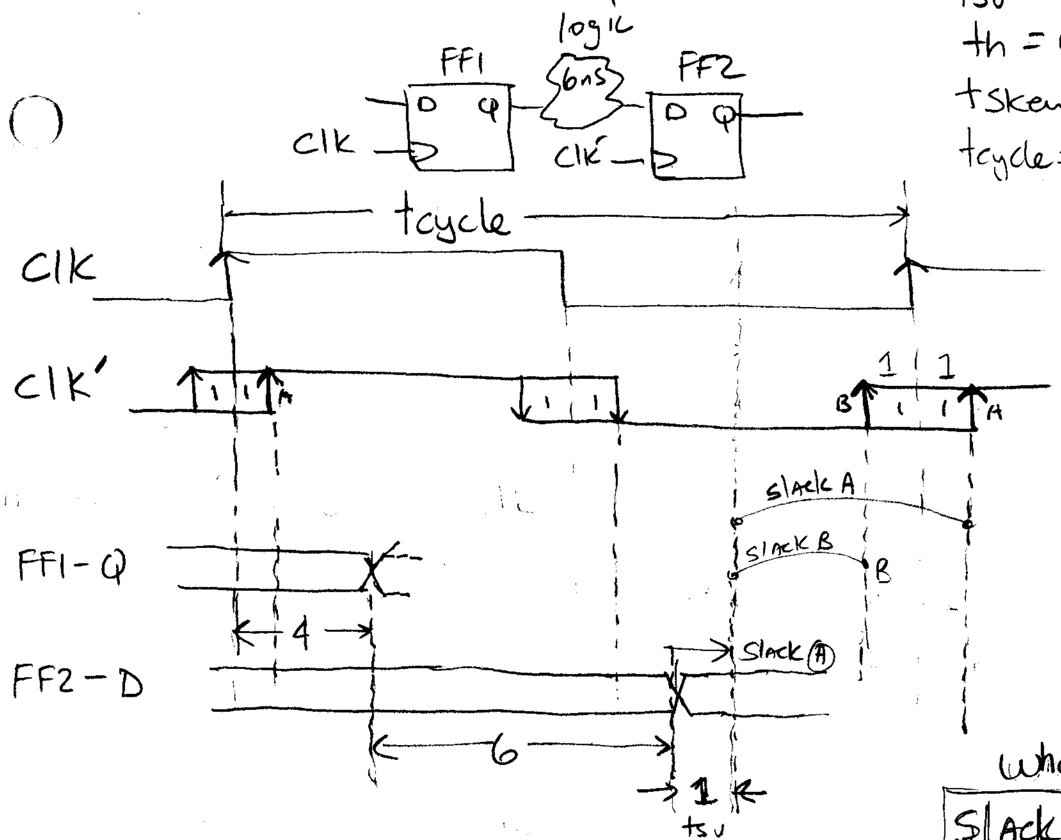


* unequal clock buffer delay due to variations of voltage + temperature across the die.
* unequal loading of clock drivers due to differing FF loads, or wiring

An example with clock skew

$t_{ckQ} = 4$
$t_{su} = 1$
$t_h = 0.5$
$t_{skew} = 1$ (spec'd as ± 1ns)
$t_{cycle} = 11$ns

Ⓐ clk' behind clk
Ⓑ clk' ahead of clk



when there is clock skew:

$$\boxed{\text{Slack} = t_{cycle} - t_{ckQ} - t_{pd} - t_{su} \pm \text{skew}}$$

For case Ⓐ $= t_{cycle} - t_{ckQ} - t_{pd} - t_{su} + \text{skew}$
$= 11 - 4 - 6 - 1 + 1$
$= 1$ns

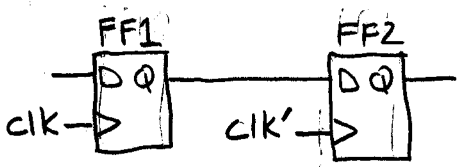
We have 1 extra ns to do the logic function... or could shave 1ns off the clock cycle

For case Ⓑ

$\text{slack} = t_{cycle} - t_{ckQ} - t_{pd} - t_{su} - \text{skew}$
$= 11 - 6 - 4 - 1 - 1$
$= \underline{-1}$ (in the hole)

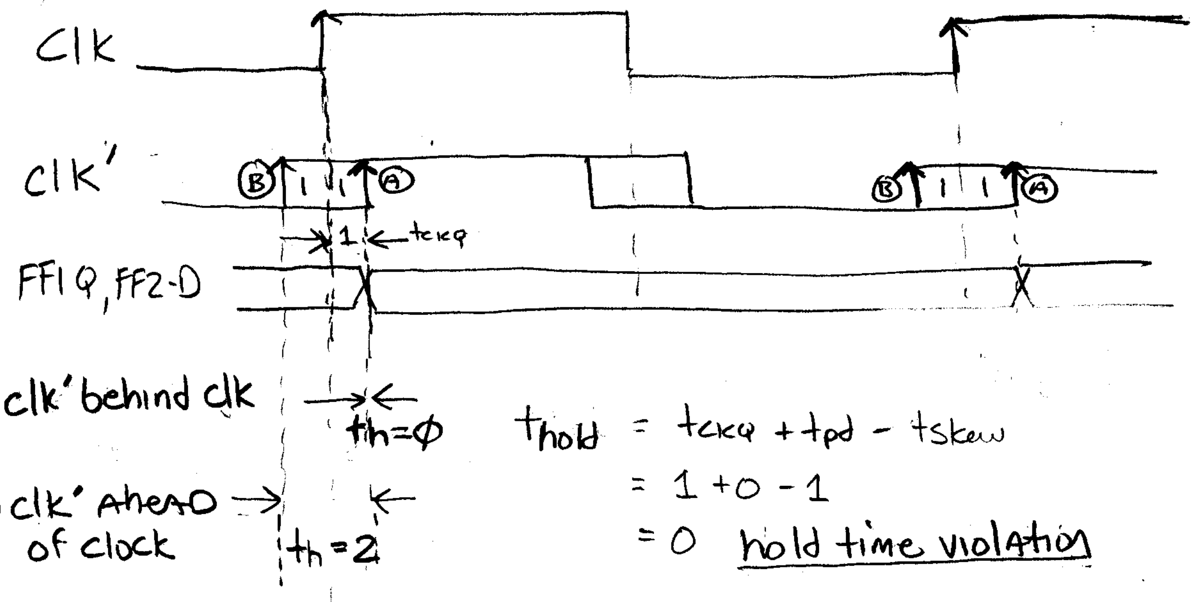
Case Ⓐ is often done intentionally in some high performance designs. The technique is called "cycle stealing" and lets you run faster if the following stage does not need as much tsu.

Hold time example with clock skew



$t_{ckq} = 1$
$t_{su} = 2$
$t_h = 0.5$
$t_{cycle} = 20$

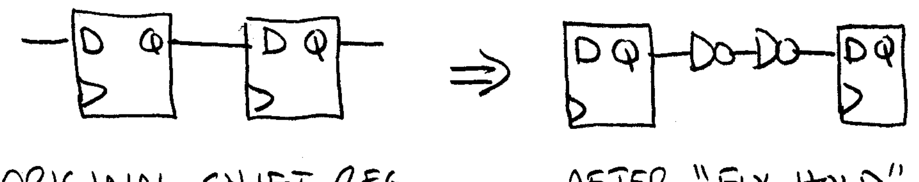
$t_{skew} = \pm 1ns$



$t_{hold} = t_{ckq} + t_{pd} - t_{skew}$
$= 1 + 0 - 1$
$= 0$ hold time violation

$t_{hold} = t_{ckq} + t_{pd} + t_{skew}$
$= 1 + 0 + 1$
$= 2ns$ ok!

- if clk' behind clk → tsu helped, th hurt
- If clk' ahead of clk → tsu hurt, th, helped
- In General, $t_{hold} = t_{ckq} + t_{pd} \pm skew$
- Hold time problems occur infrequently because tpd is usually present. Shift registers or the scan chain are exceptions.
- Synthesis tool can easily fix hold problems using a "Fix hold" directive.



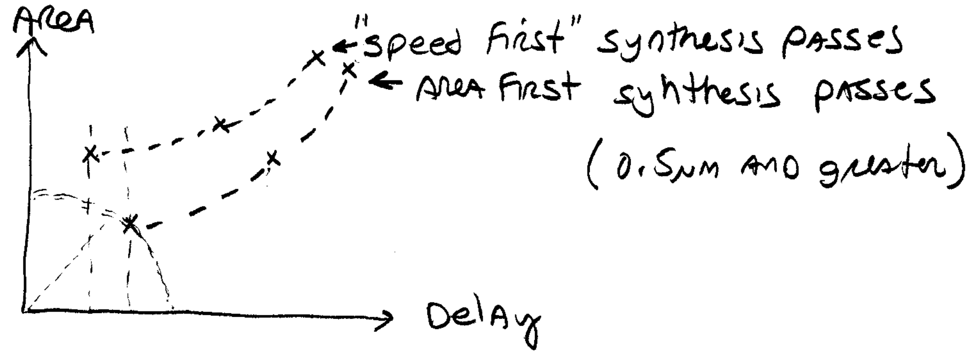
ORIGINAL SHIFT REG

AFTER "FIX HOLD" SYNTHESIS DIRECTIVE

Optimization Strategies

Synthesis tools have "knobs" to adjust
- Speed, Area, (Power?)

* Usually optimization aims for minimum area first. This approach usually converges towards the minimum area-delay product quickest.



* Deep submicron circuits exhibit a strong correlation between area and speed. Since wire delay dominates @ < .35μm the best approach is to go for minimum area.

* Speed optimization strives for 2 level logic.

* In the future, it is likely that only the area and power knobs will continue to exist.

* Why do we keep making feature sizes smaller?

1994 ⟶ 2004
0.7μm ⟶ 0.09μm

* What do these numbers mean?

→ Smallest feature actually constructed in the manufacturing process. Usually gate l

Scaling CMOS Circuits

* Scaling is the factor by which a smaller chip is obtained. (NEC slide)

* Why scale chips?

1. Cost/Function – more xistors on chip at less cost.
2. Performance - smaller feature size transistors go faster

$\frac{\text{Cost}}{\text{Perf}}$

* With smaller feature size...
  faster transistors, more transistors, more expensive

* MOSIS costs:

1.5μm process costs \$188/mm^2
0.5μm process costs \$860/mm^2 (4x the cost)

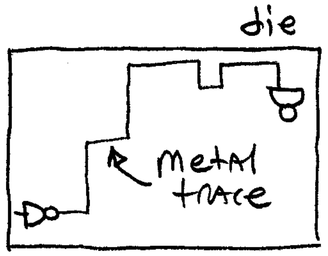
Suppose 1.5μm transistor is 3μm x 3μm = 9μm^2
0.5μm transistor is 1μm x 1μm = 1μm^2
$\therefore$ At 0.5μm we get 9x the transistors and they are 3x faster
$\therefore$ In going from 1.5μm to 0.5μm

$$\left(\frac{\text{cost}}{\text{performance}}\right) \Rightarrow \frac{\text{Factor of 4 more expensive}}{(\text{3 times faster}) * (\text{9 times the transistors})}$$

* Lets take a closer look at scaling, its not all "roses"

Scaling in More Detail

* When a process is scaled by a factor "S"
  - gate or transistor delay is reduced by roughly $1/S$
  - but, what happens to the interconnect?



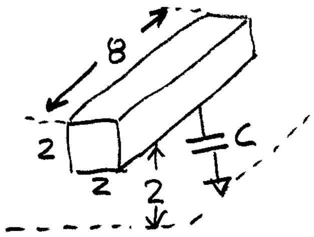
"wires" on die sometimes Al now, more commonly Cu, but also polysilicon for short paths.

* Some chips have up to 9 levels of metalization
  - upper layers are bigger, used for clock + power

* Lets look at a model of the interconnect

$R = \frac{l}{w * h} \overset{\text{material constant}}{e}$; let $e = 1$, C is proportional to parallel plate area + spacing

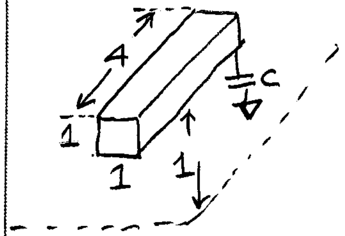
$C = \frac{l * w}{d}$



$R = \frac{8}{2 * 2} = \underline{2}$; $C = \frac{2 * 8}{2} = 8$

RC product $= 2 \cdot 8 = \underline{\underline{16}}$

scale all dimentions by 2X:

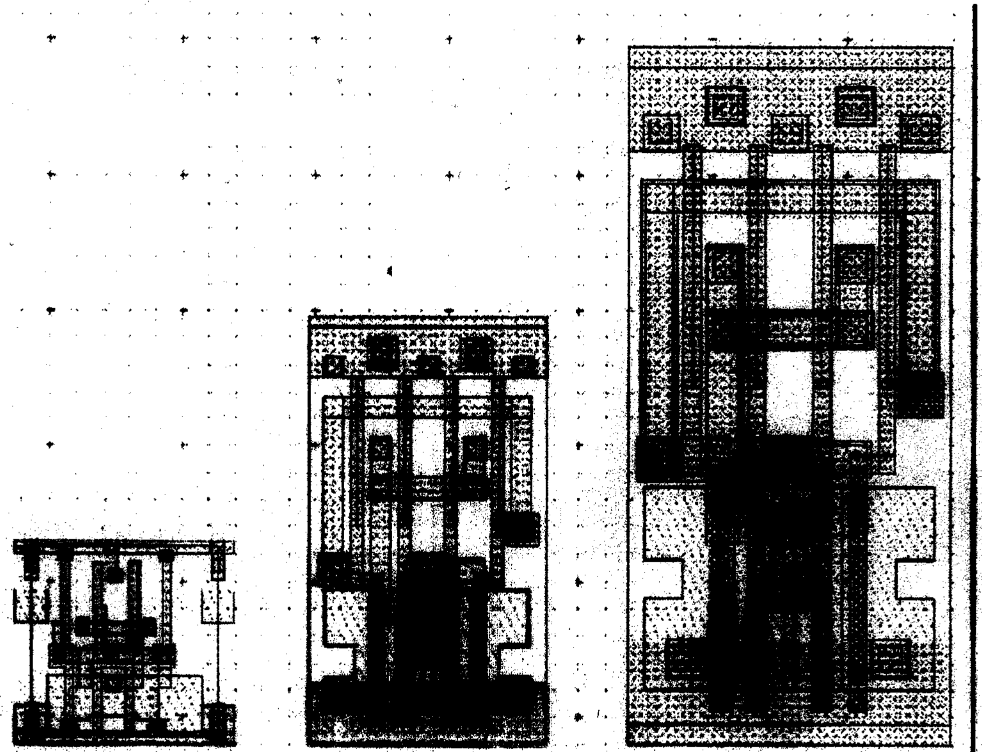


$R = \frac{4}{1 * 1} = \underline{4}$ $\quad C = \frac{4 * 1}{1} = \underline{4}$

RC Product $= 4 * 4 = \underline{\underline{16}}$

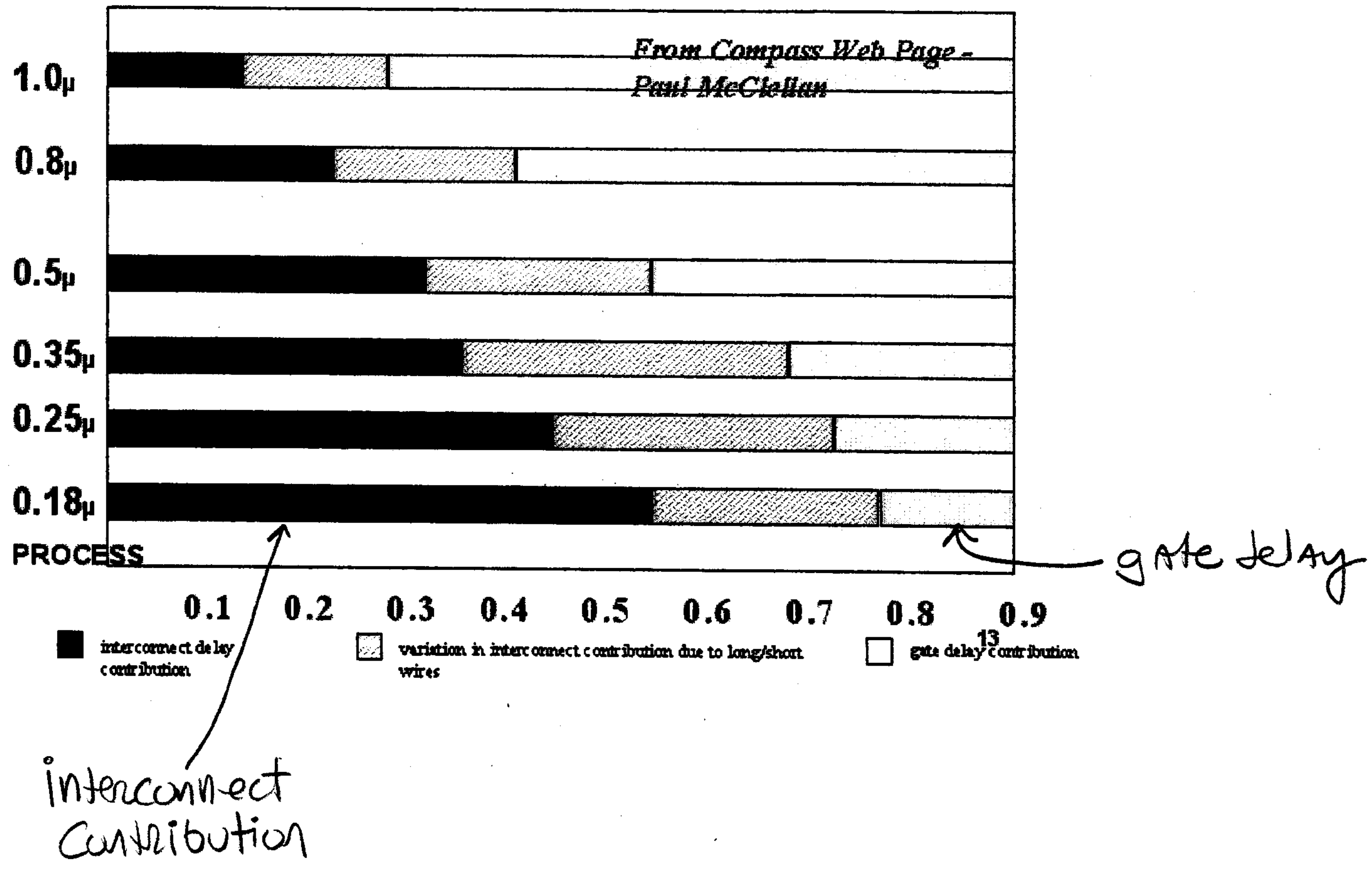
* At the <u>best</u> scaling does not improve the quality of the interconnect. (slide)

Artwork images (2-NAND)

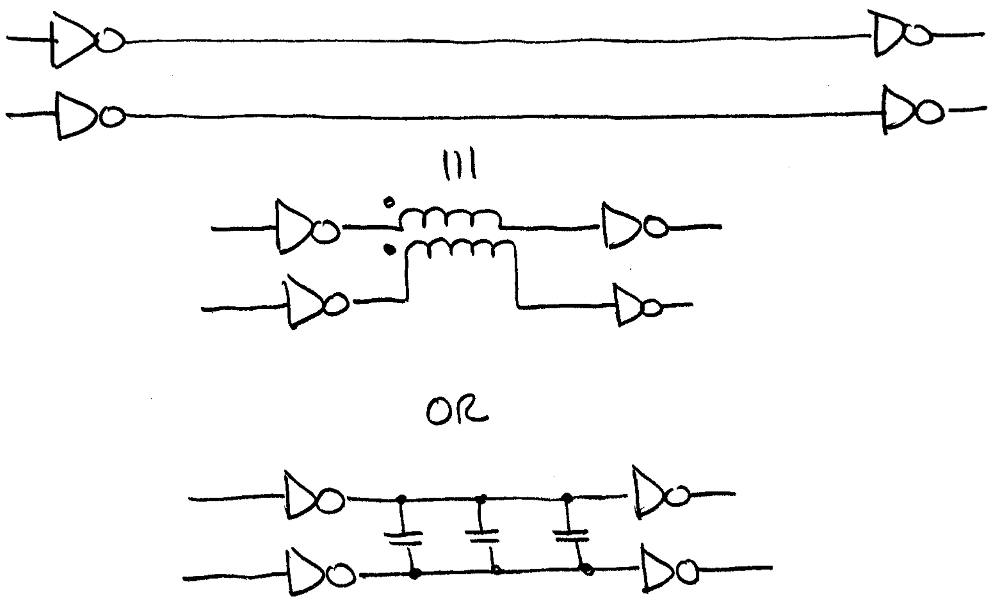


Series	CB11(0.18um)	CBC10 (0.25um)	CBC9VX (0.35um)
Width(um)	3.92	5.04	6.80
Height(um)	3.92	8.40	13.60
Size(um2)	15.36	42.34	92.48
Ratio	0.36	1.0	2.18

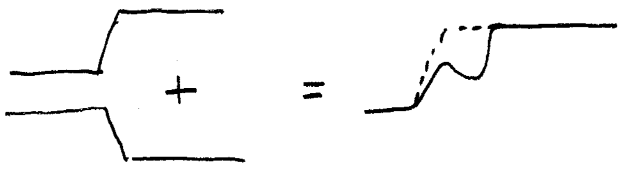
Challenge: Increasing Wire Delay



* Scaling can hurt interconnect performance
  - As wires get closer together, crosstalk becomes a factor



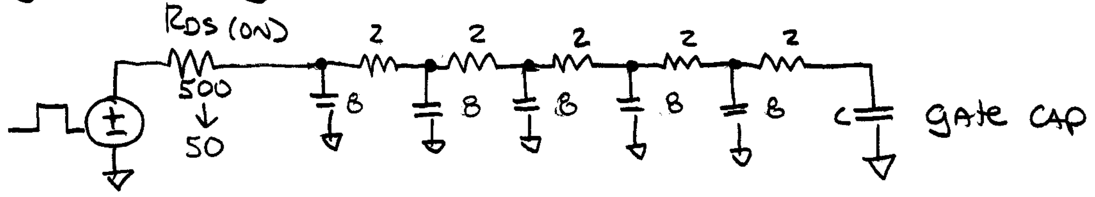
  - Adjacent buffers can launch a wavefront that opposes the edge of an incoming signal

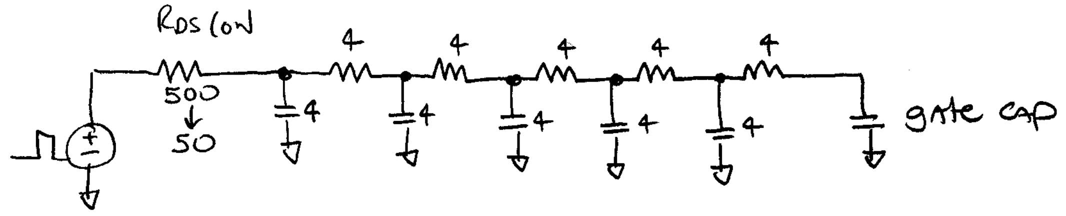


  - looks like a double edge or simply as a delayed signal

  - Adjacent buffers can launch a wavefront that speeds up an incoming signal
  - this effect is deadly to clock, reset signals

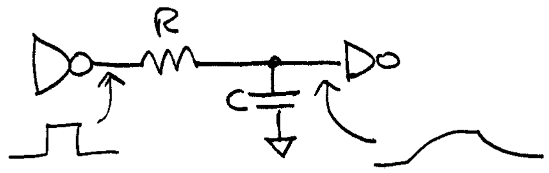
* In 0.5μm and larger, slow nets were sped up by increasing buffer size.





* In deep submicron sizes, the rds(on) resistance is less of a factor in RC delay, thus resizing of buffers is not as helpful.

* The interconnect has become a low-pass filter in small feature sizes and it won't scale.



* What can we do? Make R smaller, C smaller
  - Can't make interconnect wider, but taller?

    ? ⇐ this is actuall happening now @ < .25μm (@ < 0.25μm, 90% of C is line to line)
  - Now we have created another "C" by proximity to adjacent connectors.
  - Crosstalk is now worstened.

* Change materials
  - Copper interconnect reduces resistance by $\approx 40\%$
  - Copper has some electromigration problems and its hard to etch, but most problems are now solved. (Products don't have to last long)
  - Copper is a one-time improvement
  - Capacitance can be lowered by low "k" dielectrics
  - $Z_0 = \sqrt{\frac{L}{C}}$; lower C, higher Z, thus easier to drive
  - Low "k" dielectrics have been a real bugger!
  - SiO_2 : $k = 3.9 - 4.2$
  - low k : $k =< 3.0$