北京大學
PEKING UNIVERSITY

人工智能的硬件基石

从物理器件到计算架构

第二讲：半导体晶体管与逻辑设计

主讲：陶耀宇

2025年春季

注意事项

- 课程作业情况

- **第1次作业将在下周一，3月3号课后放出**

  之后大致每隔2-3周一次

- **第1次lab时间：3月10-4月10**
- **第2次lab时间：4月10-6月10**

注意事项

- **课程作业情况**

- 请各位选课同学在第二周周四（2.27）23:59前，**使用学号登陆CLab平台**，并同意用户协议以激活账号
- 激活账号是后续lab能够进行的必要条件，请务必完成！
- Clab网址：clab.pku.edu.cn
- **Clab问题请联系助教詹喆同学**

目录

CONTENTS

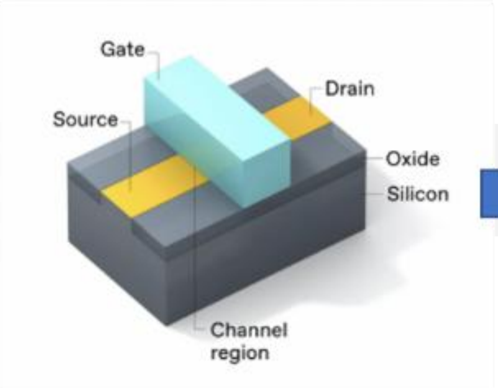
MOSFET晶体 – 现代芯片的基石

- 本课程从MOSFET开始，从基础微电子器件出发

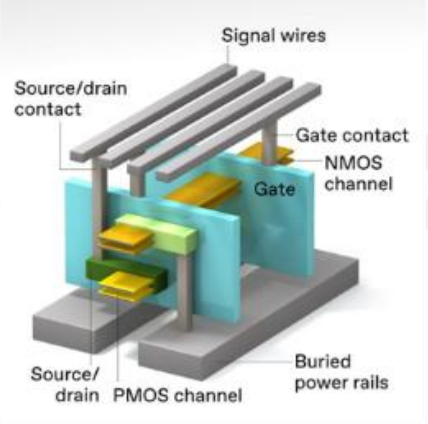
软件
- 问题 — 用户
- 算法
- 编程语言 — 程序设计
- 操作系统/环境

- 体系结构（指令集等）— 架构设计
- 微体系结构

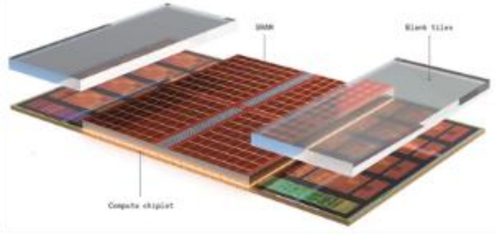
硬件
- 功能模块（AND等）— 电路设计
- 电路（数字/模拟）
- 微电子器件 — 器件工艺
- 材料工艺



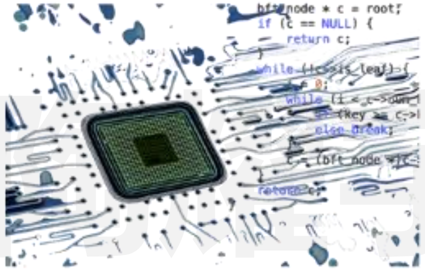
半导体物理与晶体管



数字逻辑与复杂电路



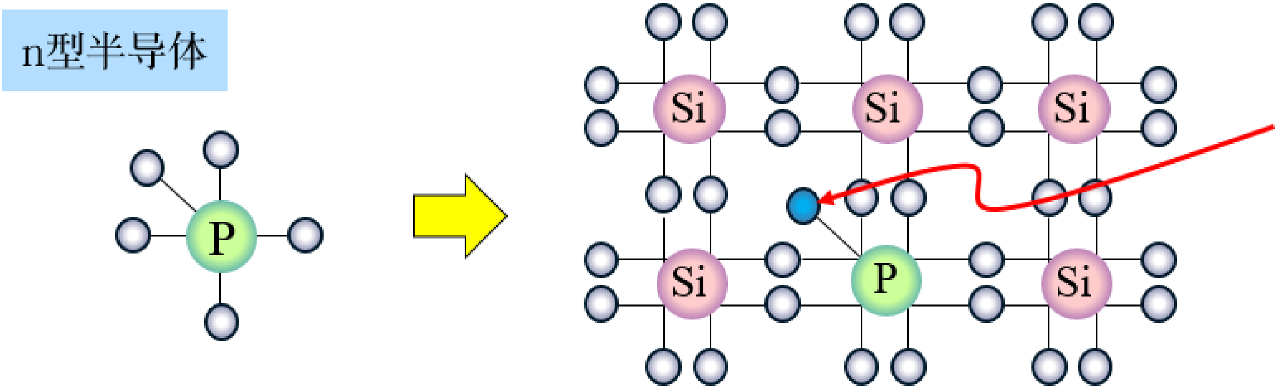
数据流、指令与芯片架构



软硬件协同设计

MOSFET晶体 – 现代芯片的基石

- N型与P型半导体的概念

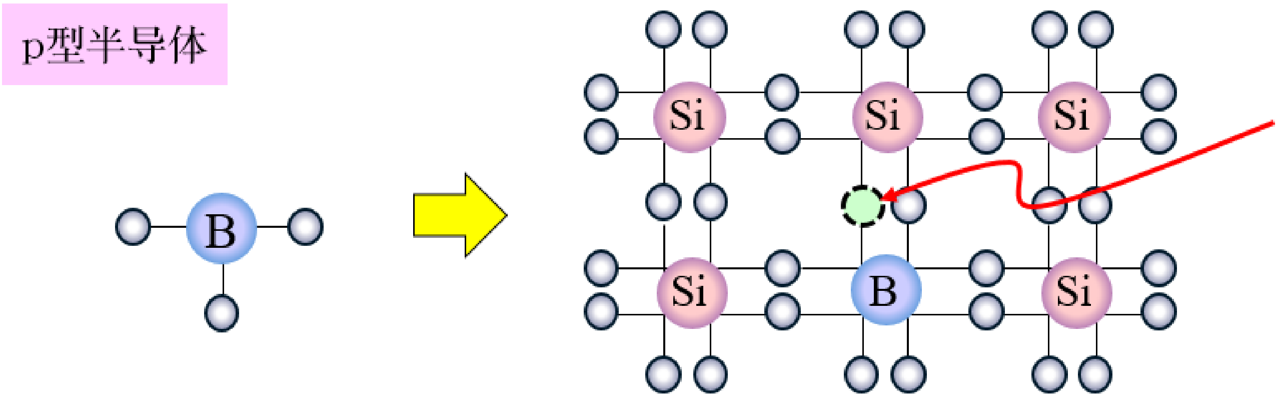


电子处于“多余”状态。

半导体内富含<u>自由电子</u>。

外加电压后电子就会被吸向+极

半导体变为导电的状态

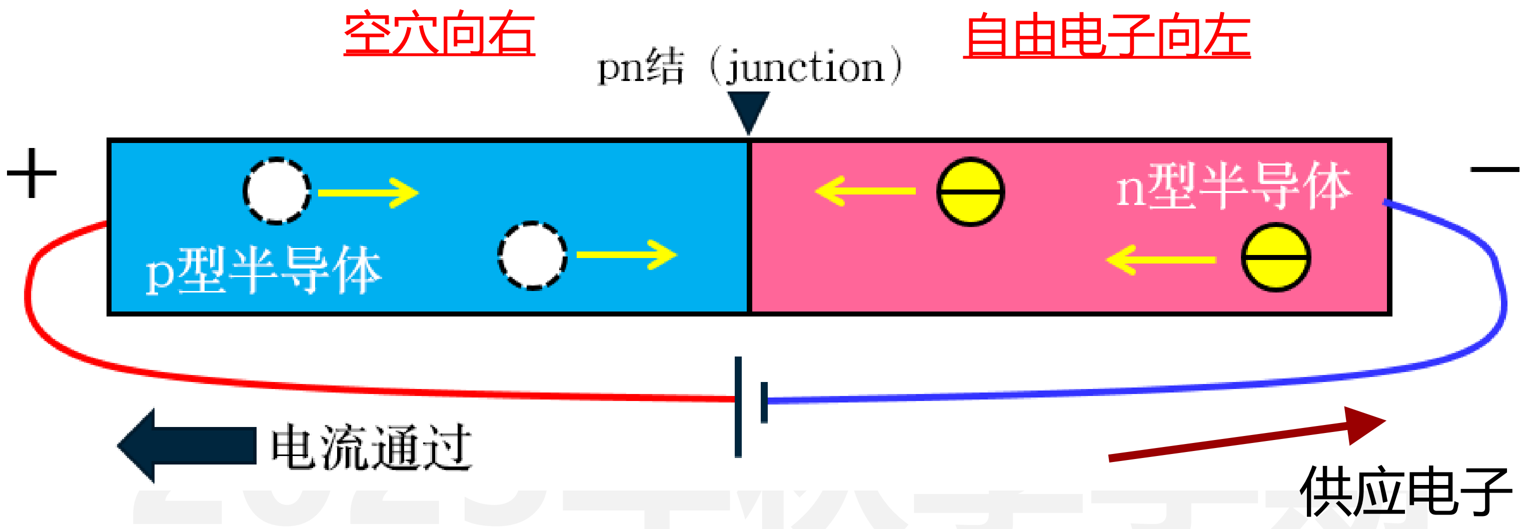


没有电子的“空位”状态

这种空位叫空穴，也就是说<u>空穴</u>中无实体，也叫虚拟粒子

MOSFET晶体 – 现代芯片的基石

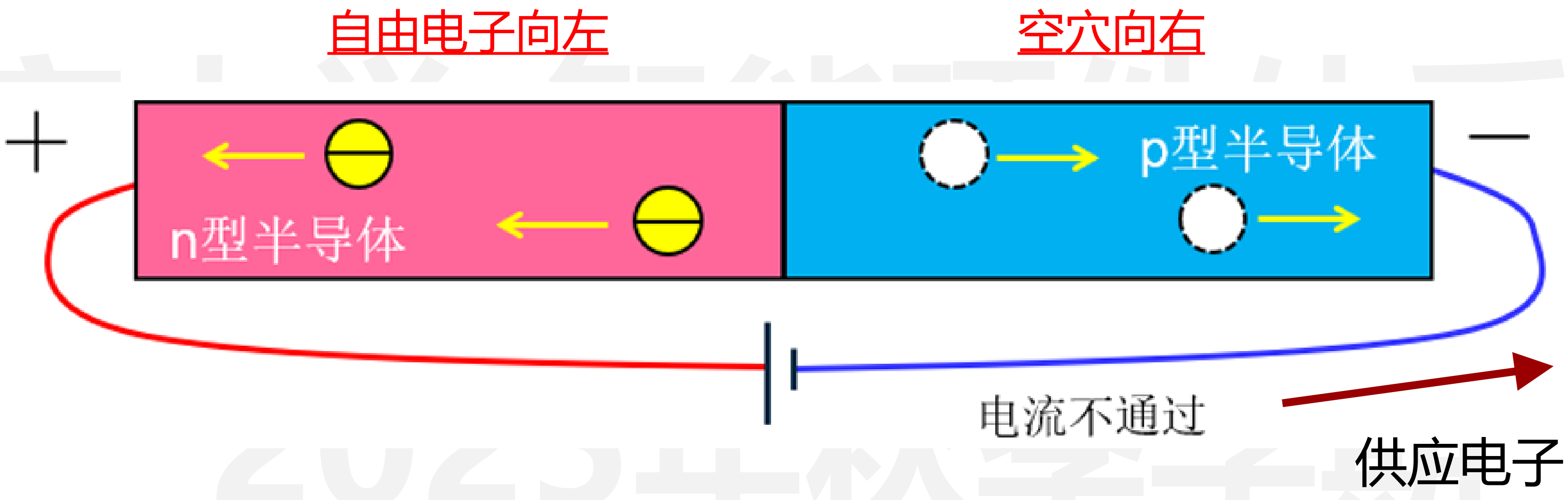
- **PN结的概念 – 导通状态**



- 对PN结外加电压使P为正极，空穴和电子都向结面移动
  当空穴与电子在结面（Junction）相遇时，电子飞入空穴，两者抵消
- **相应的新电子从电源补充流入n层**，同时电子从p层流出而产生新的空穴，如此反复，电流不断通过

MOSFET晶体 – 现代芯片的基石

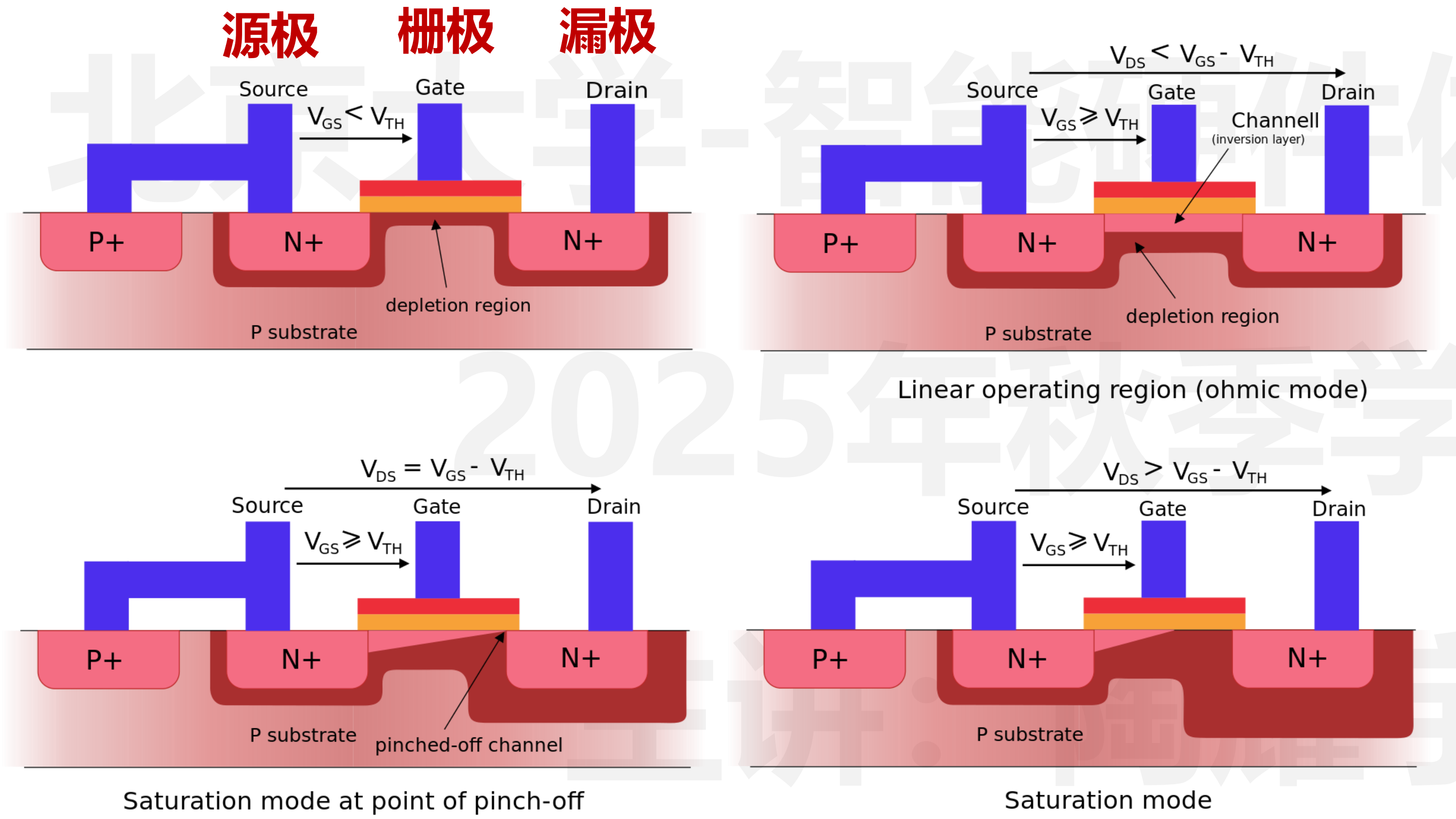
- **PN结的概念 – 断开状态**



- 对PN结外加电压使N为正极。
- **空穴和电子向相互远离的方向移动**，因此不会在结面相遇，电流无法通过
- **在结面附近会形成既无空穴又无电子存在的区域，叫做耗尽层**，它会产生耐压。
- 从上述可知pn结**具有整流作用**

MOSFET晶体 – 现代芯片的基石

- MOSFET结的概念 – 栅极(gate)、漏极(drain)和源极(source)

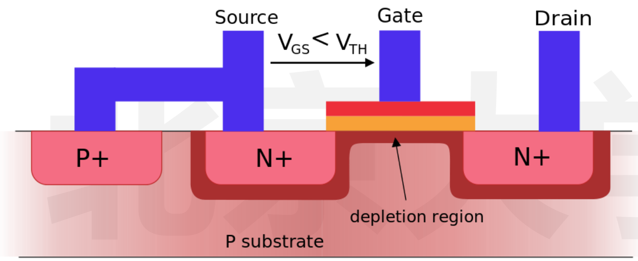


- 当栅极-源极间的电压Vgs<Vth时，N接了电源正极无论漏极-源极间的电压Vds为多少，**因为PN结的单向导通性**，不会有电流从漏极流向源极

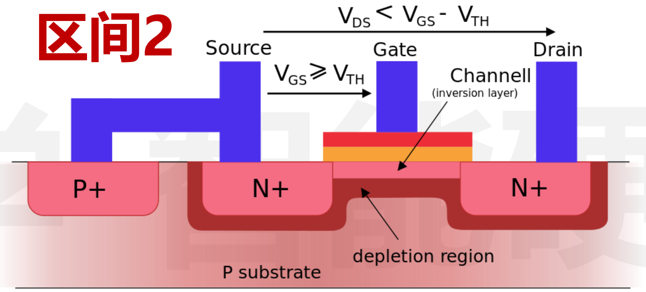
- 当Vgs超过阈值电压Vth后，**会形成一个横跨二氧化硅层的电场**，在这种电场的作用下，**SiO2层和P区交界处附近的电子会被吸引至SiO2侧**，在SiO2侧形成一个局部电子浓度相对较高的带状区域(沿着SiO2表面)，即**N型导电沟道**(depeletion)

MOSFET晶体 – 现代芯片的基石

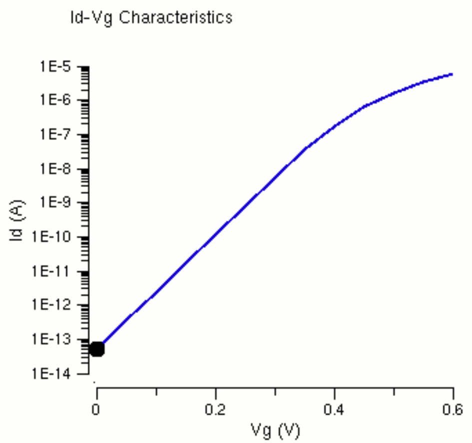
- MOSFET有三个工作区间：断开、线性（欧姆区间）、饱和（电压不随电流线性增加）

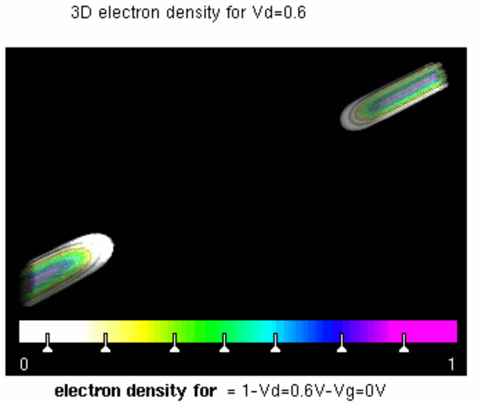


区间2



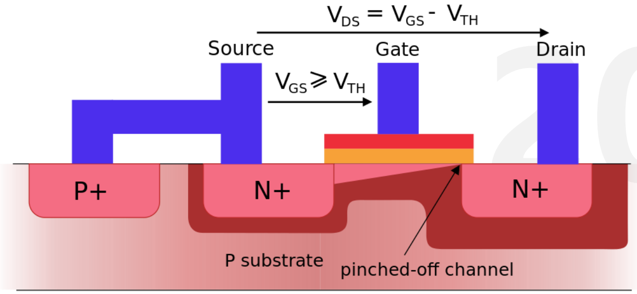
Linear operating region (ohmic mode)



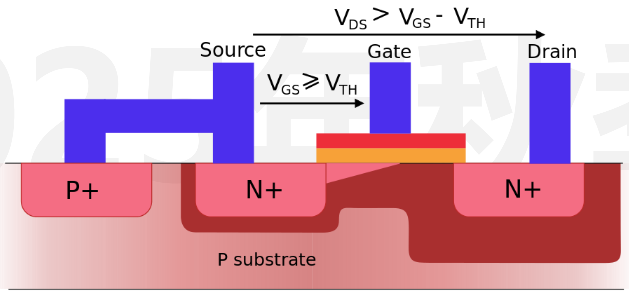


区间1 核心：Vg不足

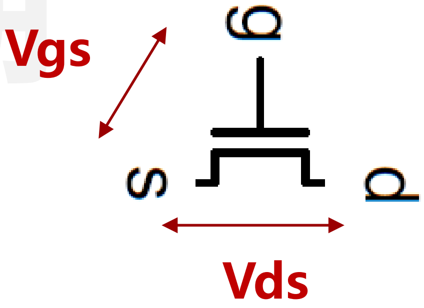
核心：Vd不足



Saturation mode at point of pinch-off



Saturation mode



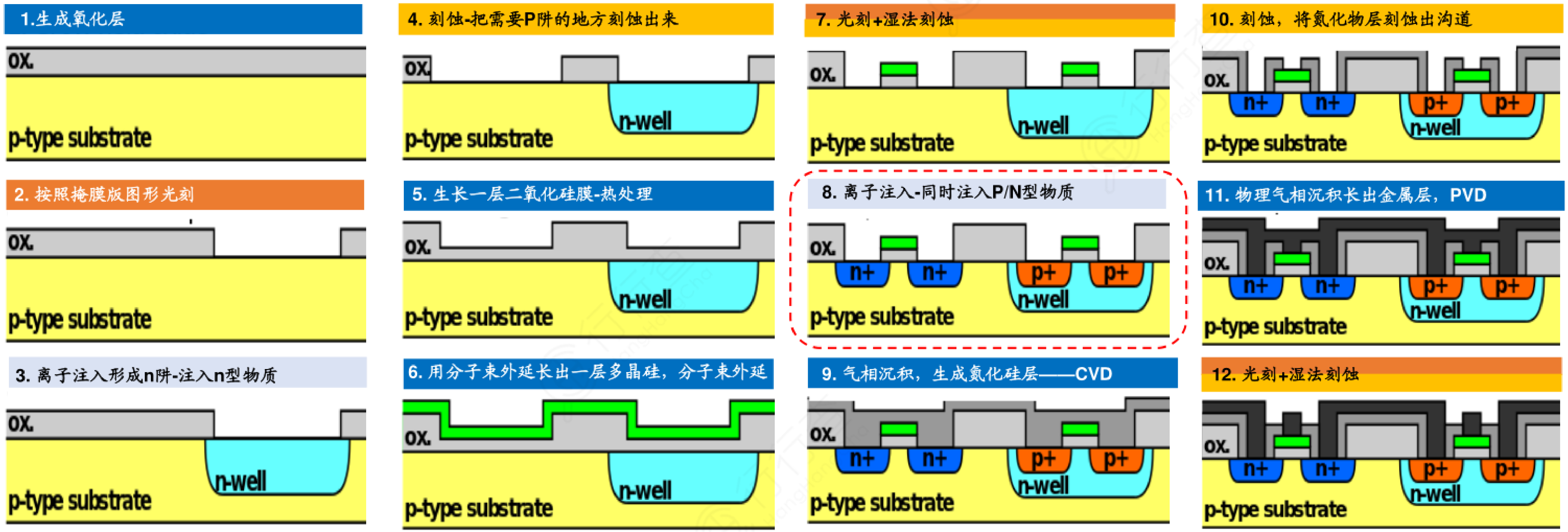
区间3 核心：Vd过大

Vth：MOSFET的阈值电压与工艺相关

1、断开区间：Vgs < Vth 2、线性区间：Vds < Vgs - Vth 3、饱和区间：Vds > Vgs - Vth

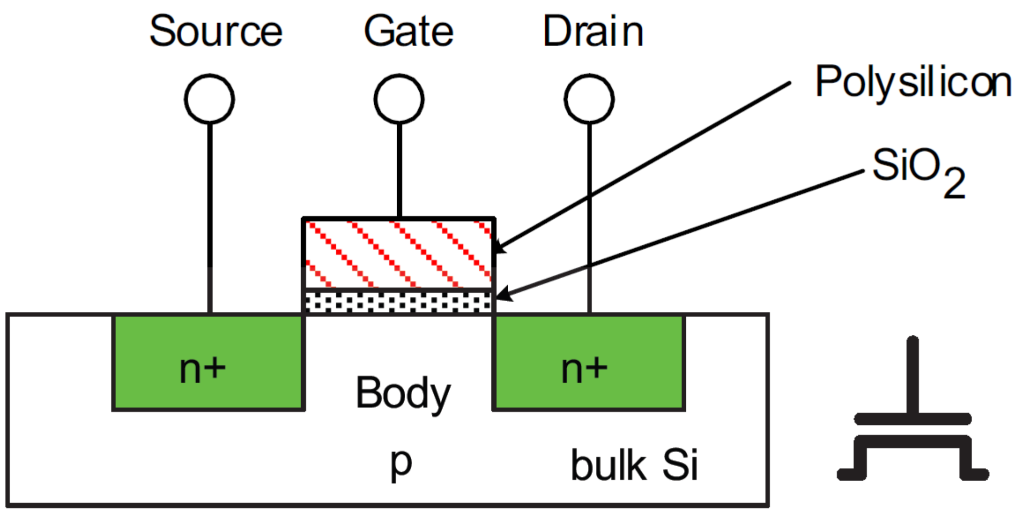
MOSFET晶体 – 现代芯片的基石

- MOSFET的制造过程总结

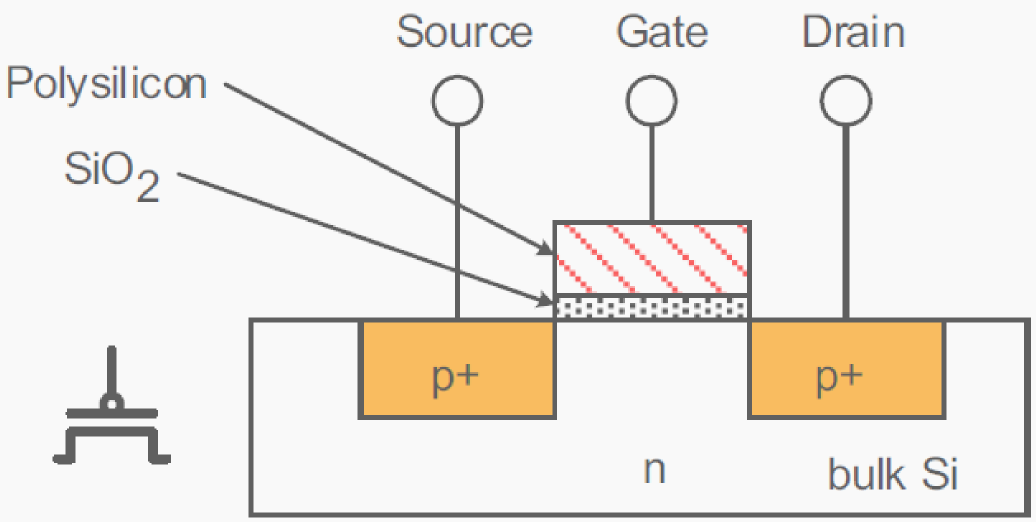


MOSFET晶体 – 现代芯片的基石

- **NMOS与PMOS**



NMOS



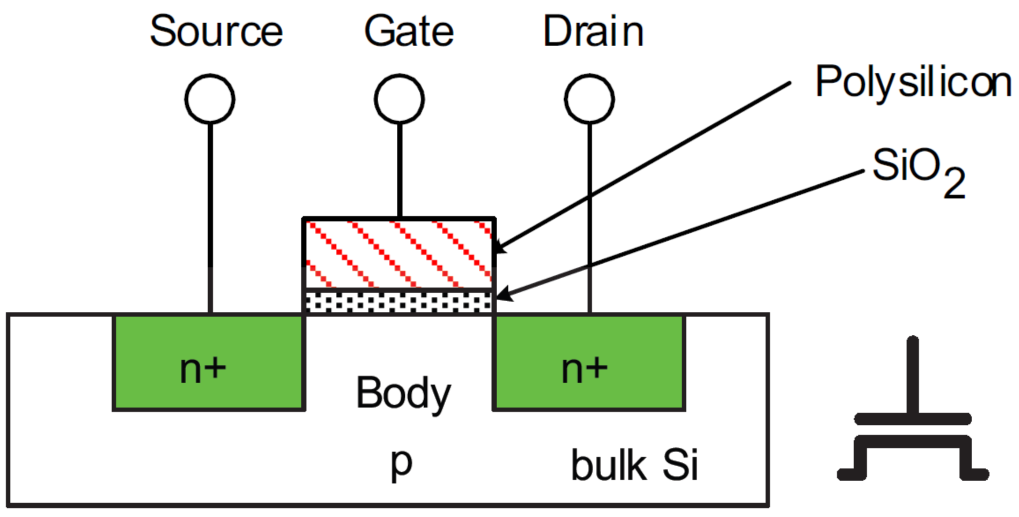
PMOS

当**栅极处于低电压：**

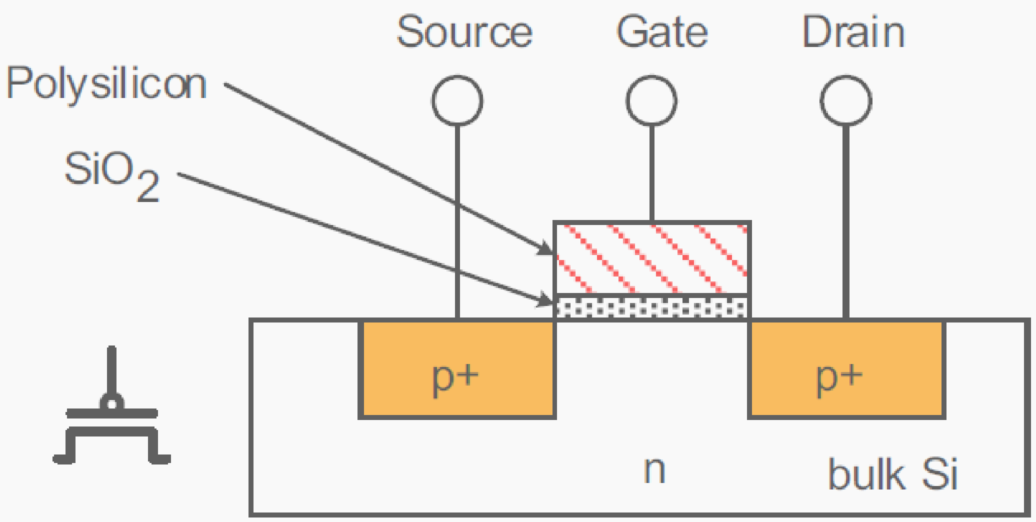
§ 体端（body）接地，即低电压
§ 栅极（gate）下通道中不存在导电通路
§ 无电流流动，晶体管关闭

MOSFET晶体 – 现代芯片的基石

- **NMOS与PMOS**



NMOS



PMOS

当栅极处于高电压：

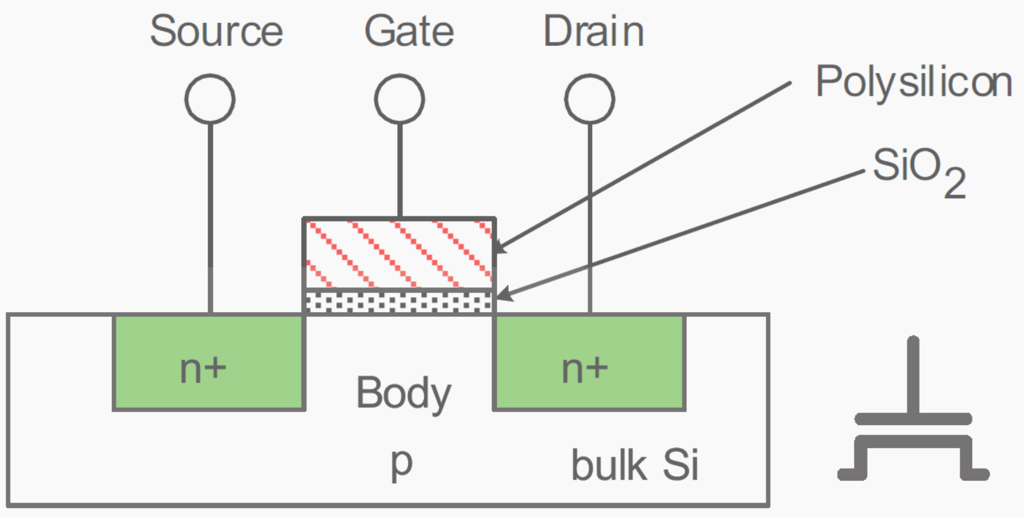
§MOS 电容栅极上有正电荷，体端则有负电荷

§将栅极下的通道反转为 n 型

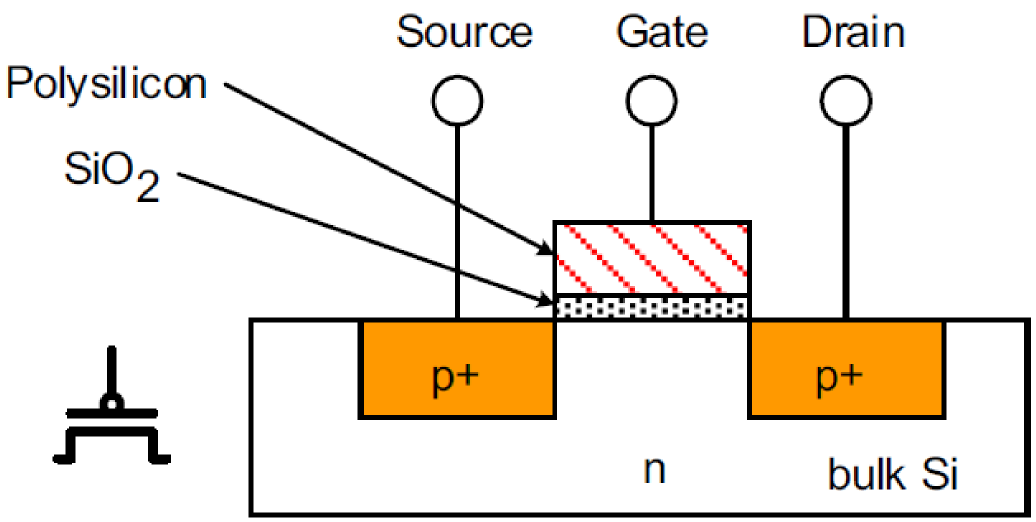
§现在电流通过通道流过n型硅，晶体管导通

MOSFET晶体 – 现代芯片的基石

- **NMOS与PMOS**



NMOS



PMOS

类似NMOS，但掺杂和电压相反

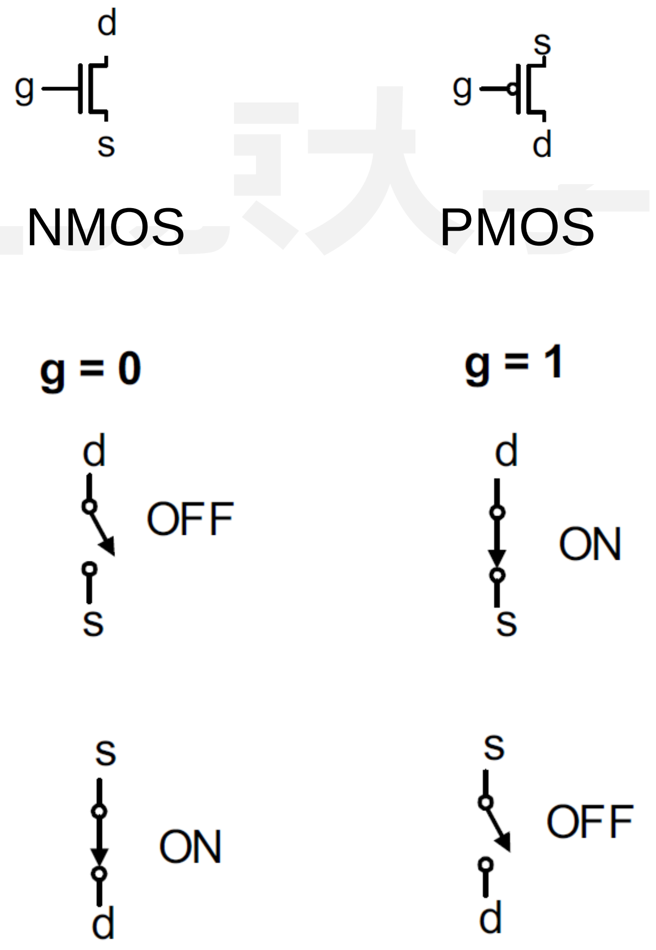
§体端与高电平相连（VDD）

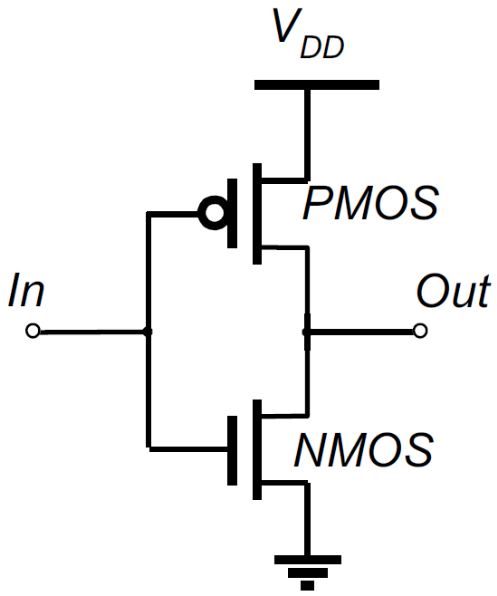
§**栅极低电平**：晶体管导通

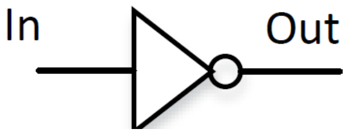
§**栅极高电平**：晶体管关闭

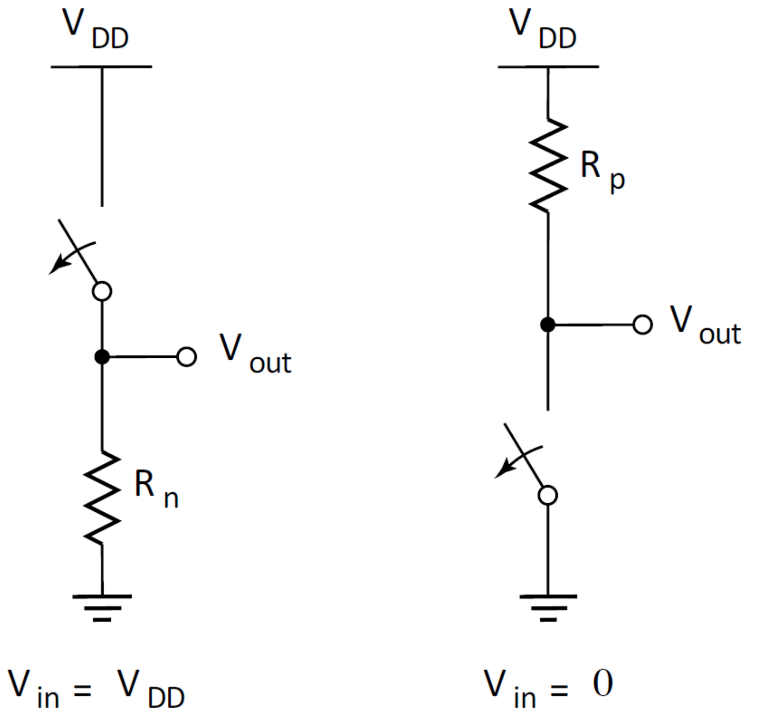
MOSFET作为开关的行为级模型

- **NMOS、PMOS和简单反相器**



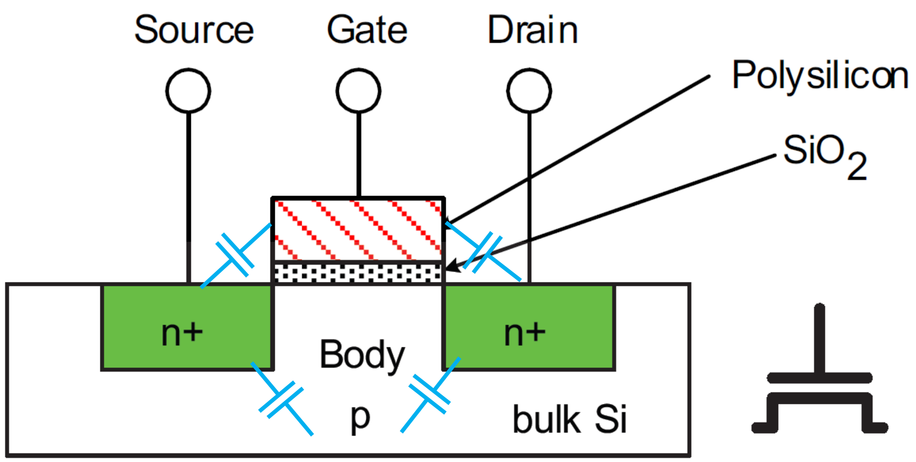




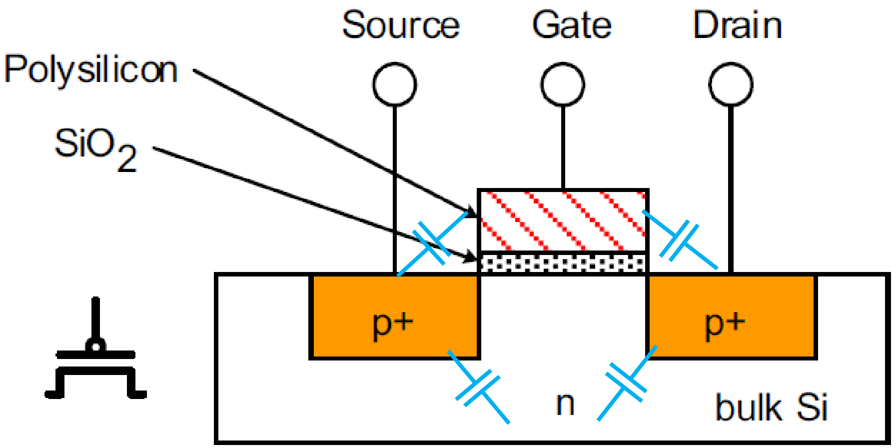


MOSFET晶体 – 现代芯片的基石

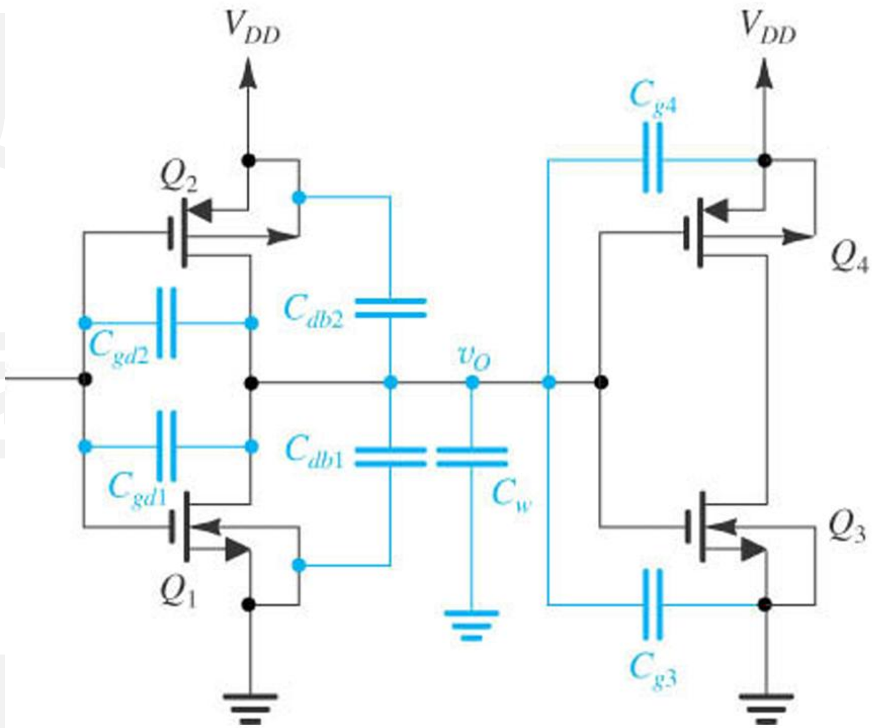
- **NMOS与PMOS的寄生电容**



NMOS

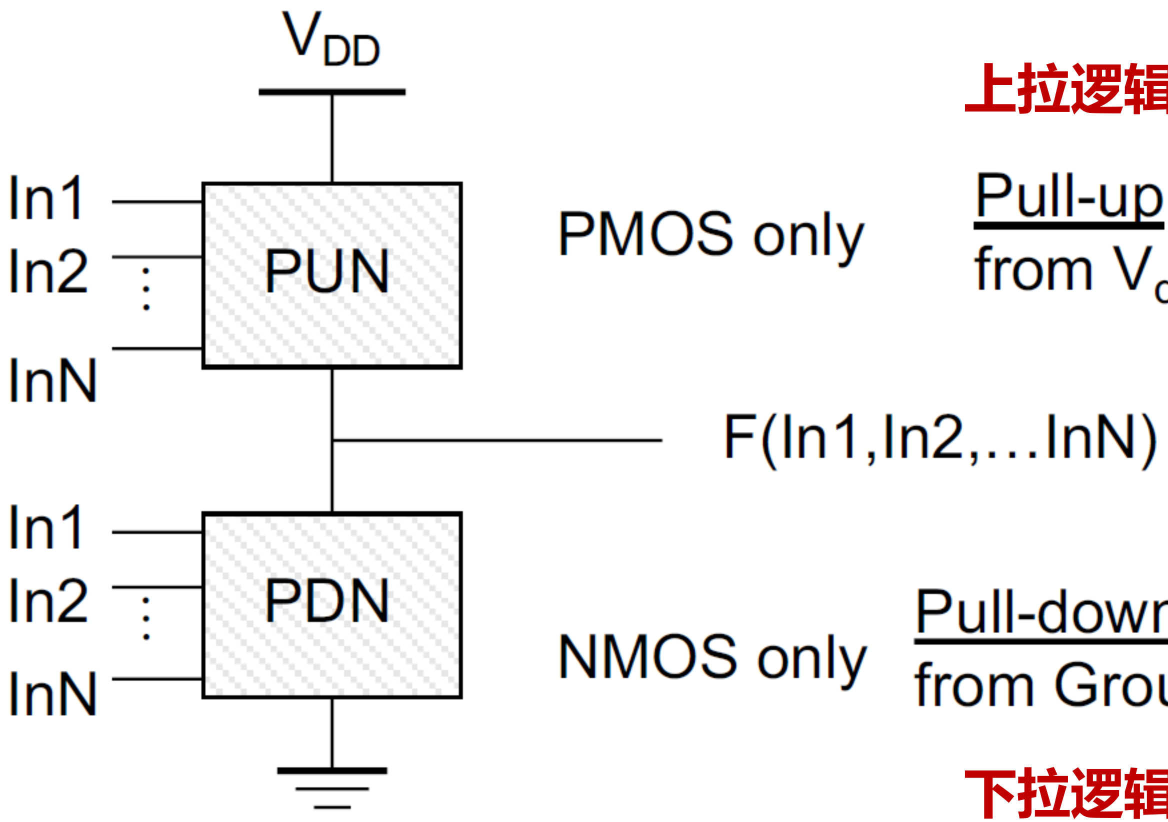


PMOS



经典静态逻辑电路

- NMOS、PMOS可以组成PDN和PUN



上拉逻辑设计：

Pull-up network: make a connection from V_{dd} to F when F(In1,In2…)=1

Pull-down network: make a connection from Ground to F when F(In1,In2…)=0

下拉逻辑设计：

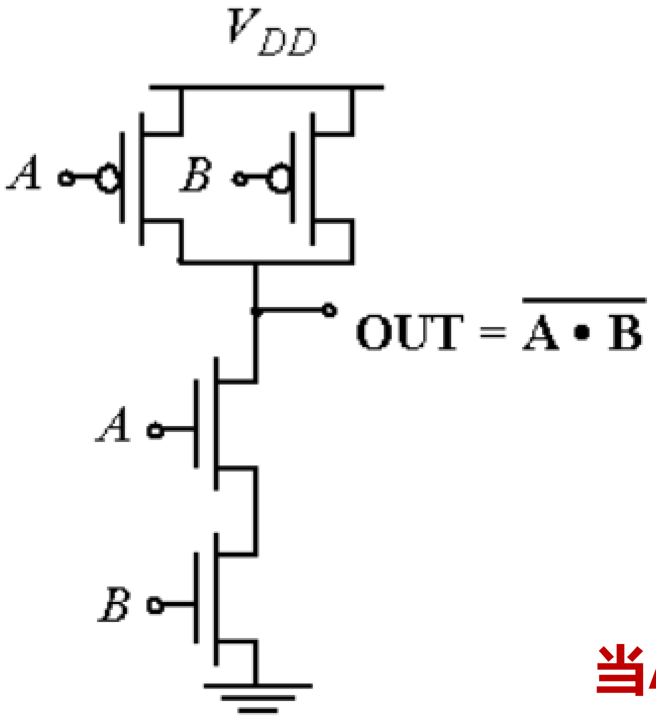
PUN and PDN are *dual* networks

经典静态逻辑电路

- 由PDN和PUN组成的NAND门电路

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate



NAND门

当A和B同为1时，输出为0

当A和B有0时，输出为1

PDN: Connects OUT to ground when A●B=1

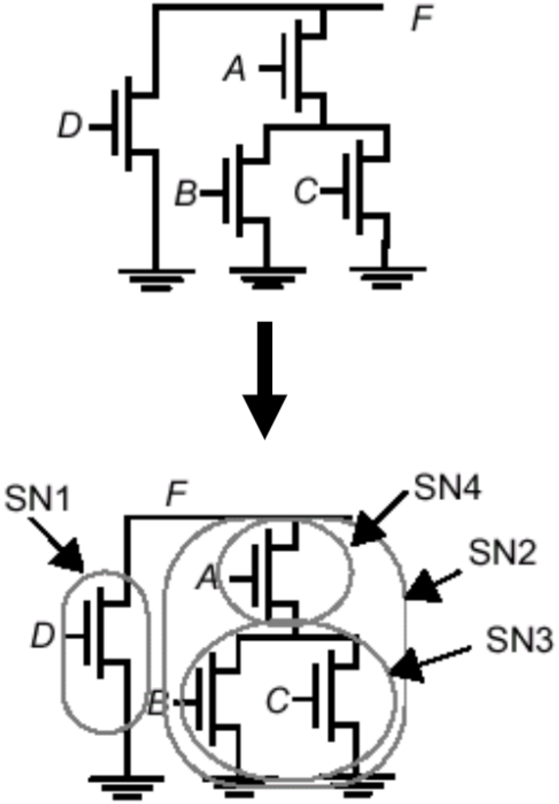
PUN: Connects OUT to V_{dd} when $\overline{A} + \overline{B} = 1$

> So OUT = Complement of PDN function
> Also OUT = PUN function with each input inverted

经典静态逻辑电路

- **由PDN和PUN组成的复杂静态逻辑电路**

$F = \overline{D + (A(B + C))}$ **什么情况下F为0 -> D+A(B+C) = 1**



忽略取非操作，直接构建PDN

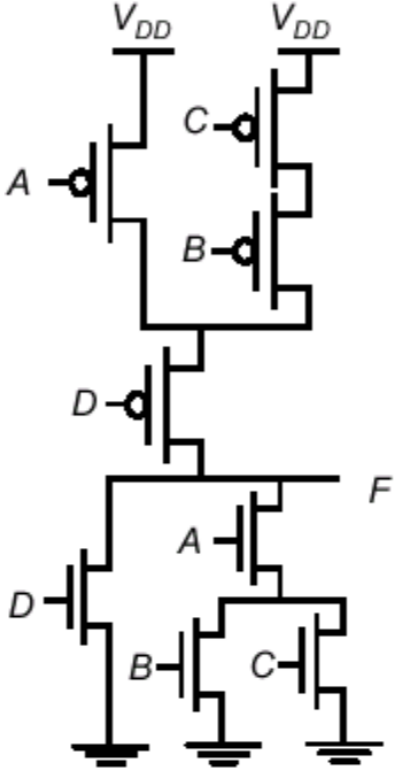
Ex：D+X意味着D与X并联

构建PDN的亚网络

在SN3内，B和C是并联的，

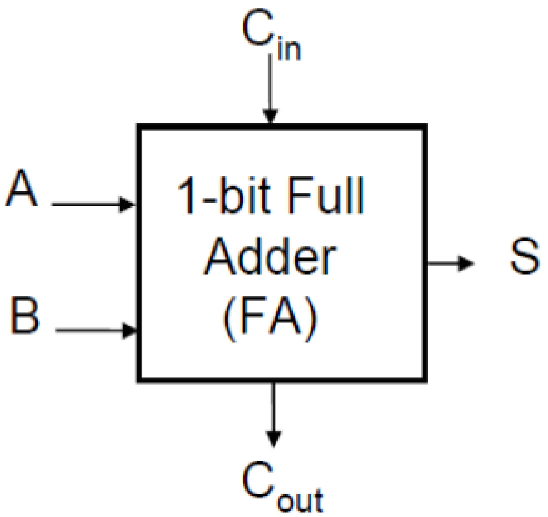
则在PUN中，他们串联

自下向上构建



- 简单1bit加法器电路

  - A[n-1:0] + B[n-1:0] = S[n-1:0]



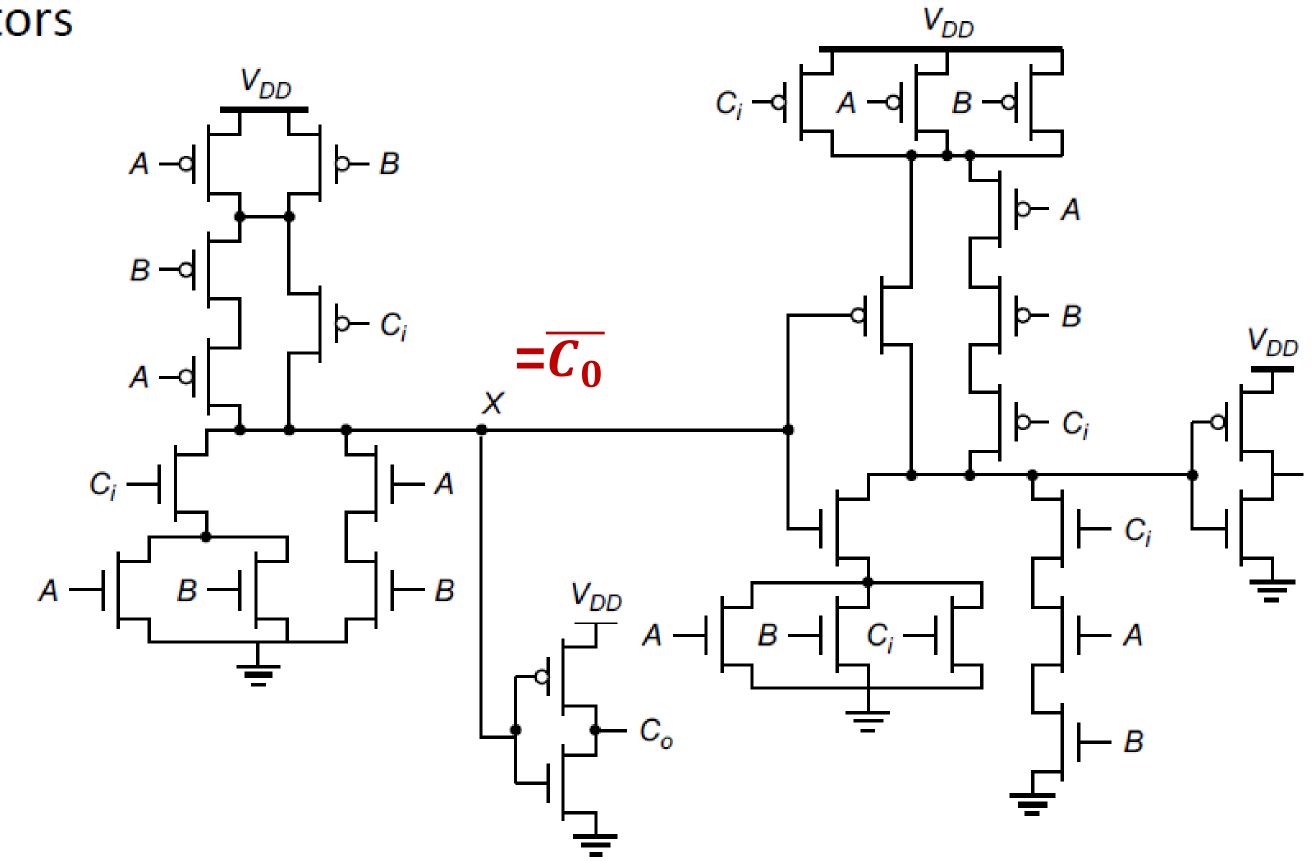
A	B	C_{in}	C_{out}	S	carry status
0	0	0	0	0	kill
0	0	1	0	1	kill
0	1	0	0	1	propagate
0	1	1	1	0	propagate
1	0	0	0	1	propagate
1	0	1	1	0	propagate
1	1	0	1	0	generate
1	1	1	1	1	generate

单比特加法器逻辑设计

  - $C_o = AB + BC_i + AC_i = AB + (A + B)C_i$
  - $S = A \oplus B \oplus C_i = ABC_i + \overline{C_o}(A + B + C_i)$
  - $G = AB,\ K = \overline{A}\,\overline{B},\ P =$ A⊕B

经典静态逻辑电路

- 简单1bit加法器电路
  - $C_o = AB + BC_i + AC_i = AB + (A + B)C_i$
  - $S = A \oplus B \oplus C_i = ABC_i + \overline{C_o}(A + B + C_i)$
  - 28 transistors

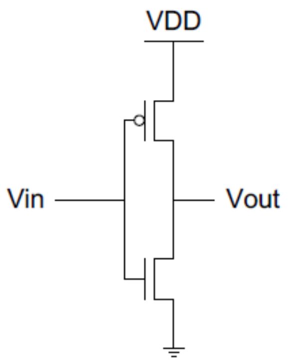


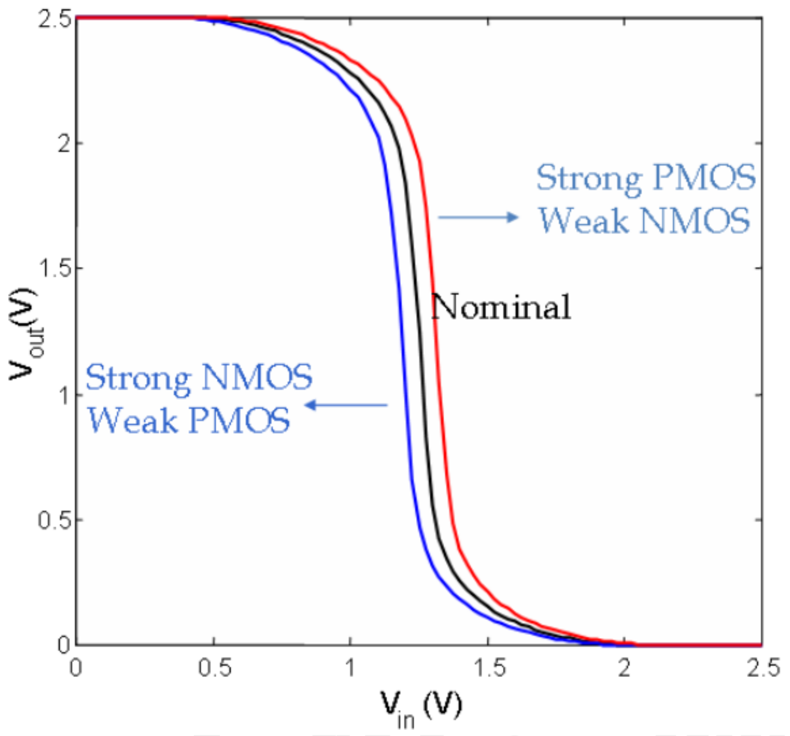
目 录

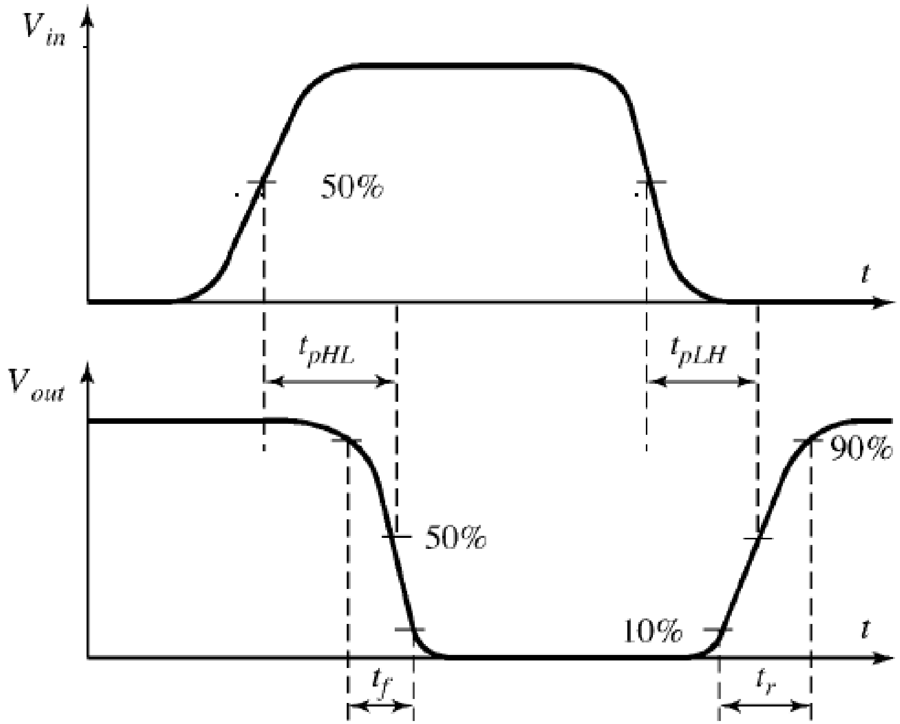
CONTENTS

什么是电路的延迟

- 以Inverter反相器为例

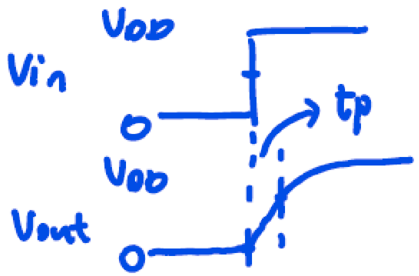


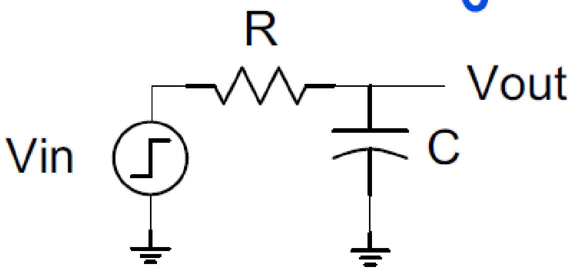




- 一阶RC延迟分析

A First-Order RC Network





$$V_{out}(t) = (1 - e^{-t/\tau})V_{DD} = \frac{1}{2}V_{DD}$$

$$e^{-t/\tau} = \frac{1}{2}$$

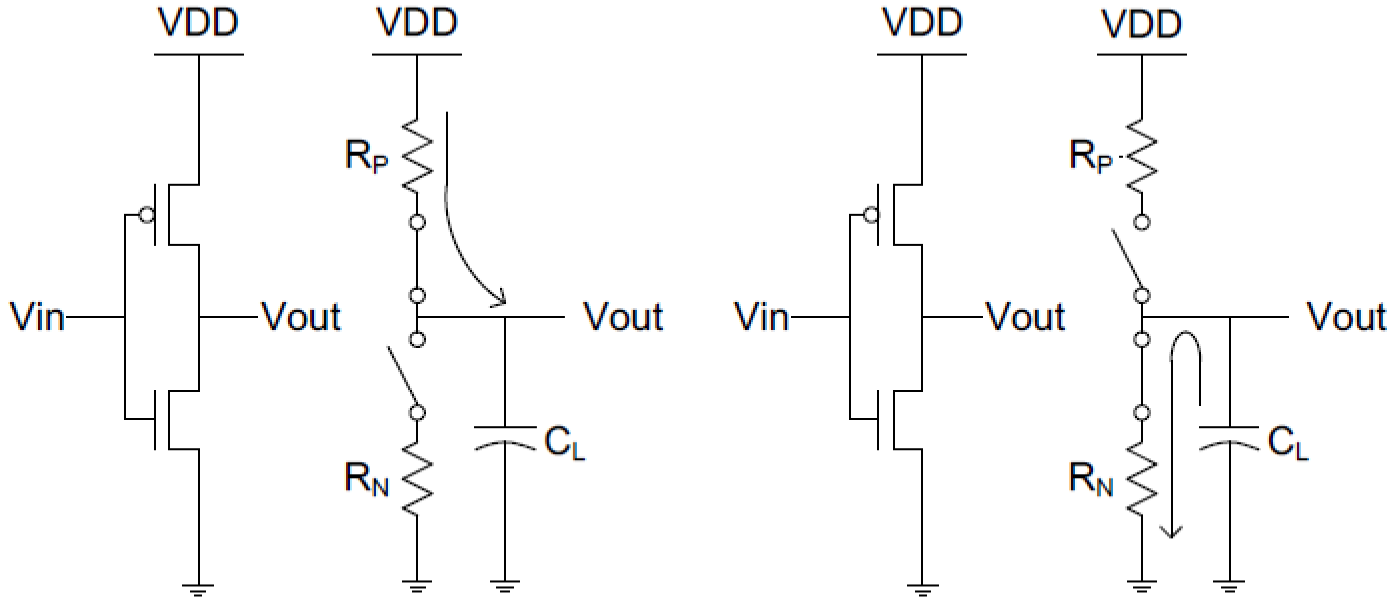
$$\tau = R \times C$$

$$-t/\tau = -\ln 2$$

$$t_p = \ln(2)\tau = 0.69 R \times C \qquad t = \ln(2)\tau$$

反相器延迟

- **利用一阶RC延迟分析方法**



$$t_{pHL} = f(R_N \times C_L)$$

$$t_{pHL} = 0.69 R_N \times C_L$$

$$t_{pLH} = 0.69 R_P \times C_L$$

$V_{in} = 0$

(a) Low-to-high

$V_{in} = V_{DD}$

(b) High-to-low

降低延迟的设计方法

- 利用一阶RC延迟分析方法

$$t_{pHL} = f(R_N \times C_L)$$

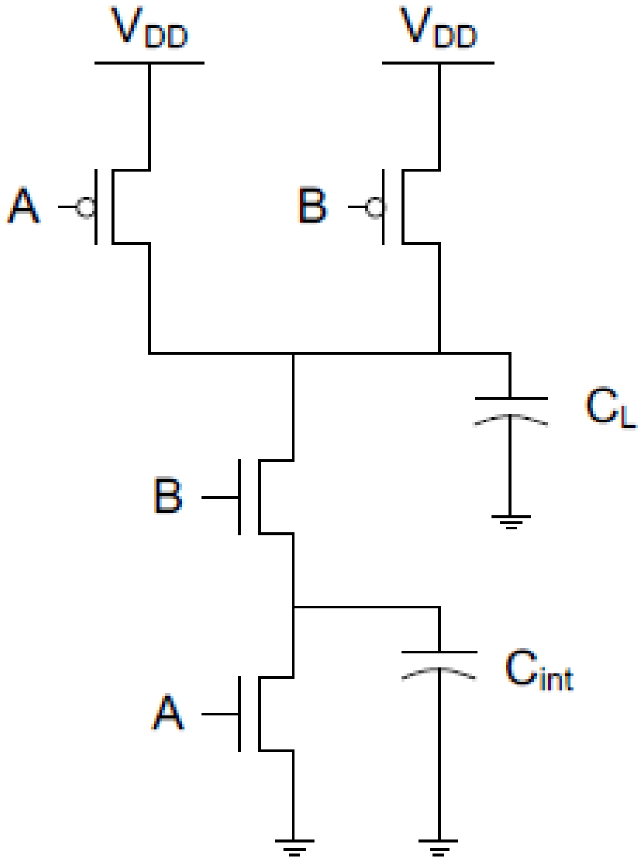
$$t_{pHL} = 0.69 R_N \times C_L$$

$$t_{pLH} = 0.69 R_P \times C_L$$

- **利用较小的电容 – 降低C**
  - – 版图紧凑, 布局合理
  - - 保持较短走线&减少diffusion routing
- **增加晶体管尺寸 – 降低 R**
  - –避免self-loading出现，否则会导致寄生电容增大
- **增加电源电压**
  - – 同时会影响可靠性与功耗，因而一般不采用

输入Pattern对延迟的影响

- 利用一阶RC延迟分析方法

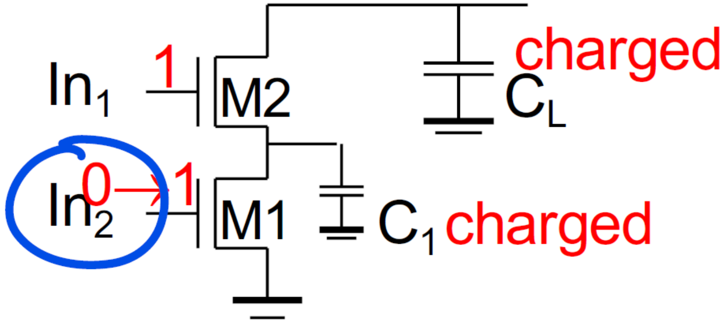


电路延迟与输入的顺序有关！

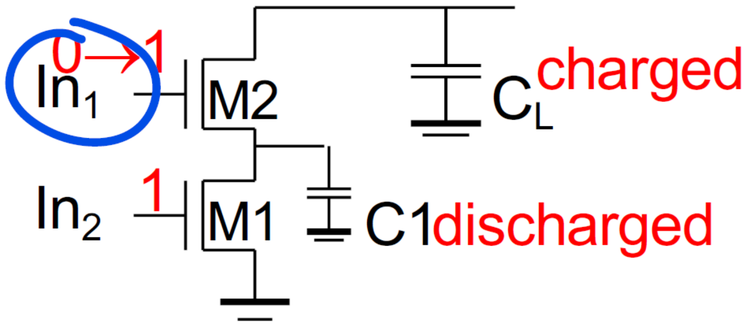
- *Ignore C_{int} for the moment!*
- Low to high transition
  - both inputs go low
    - delay is 0.69 $R_p/2$ C_L
  - one input goes low
    - delay is 0.69 R_p C_L
- High to low transition
  - both inputs go high
    - delay is 0.69 $2R_n$ C_L

利用Transistor Ordering提升逻辑速度

- 复杂的Transistor Ordering需要仿真工具支持



延迟由CL与C1放电时间决定



延迟由CL放电时间决定

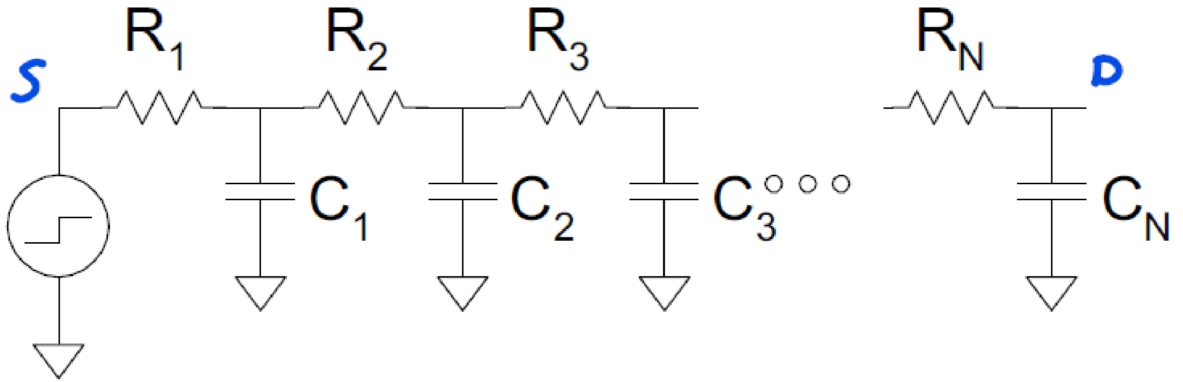
逻辑电路的Elmore Delay模型

- **拓展多级的RC模型**

  - 导通晶体管看作电阻
  - 电路网络建模为RC阶梯
  - RC阶梯的Elmore延迟
  - Apply to complex gates （i.e.,stacks),also interconnect （later)

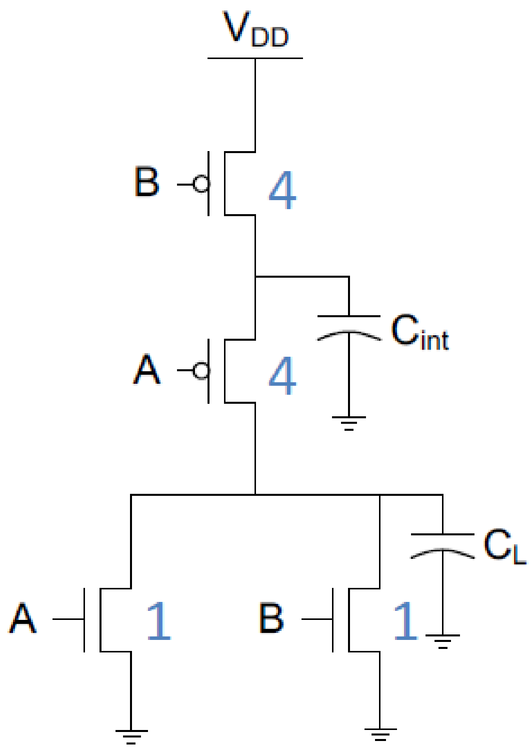
$$t_{pd} \approx 0.69 \sum_{\text{nodes } i} R_{i-to-source} C_i$$

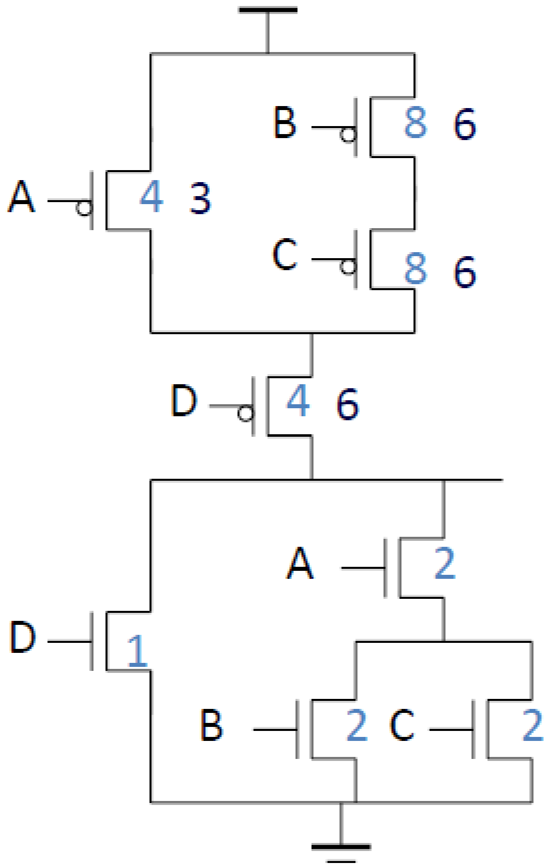
$$= 0.69\left(R_1 C_1 + (R_1 + R_2) C_2 + \ldots + (R_1 + R_2 + \ldots + R_N) C_N\right)$$



逻辑电路的Transistor Sizing

- 目标是将PDN和PUN的延迟进行匹配

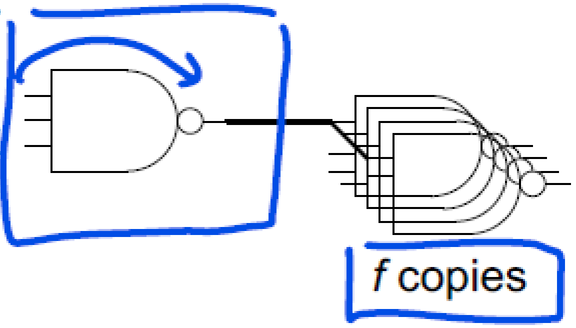


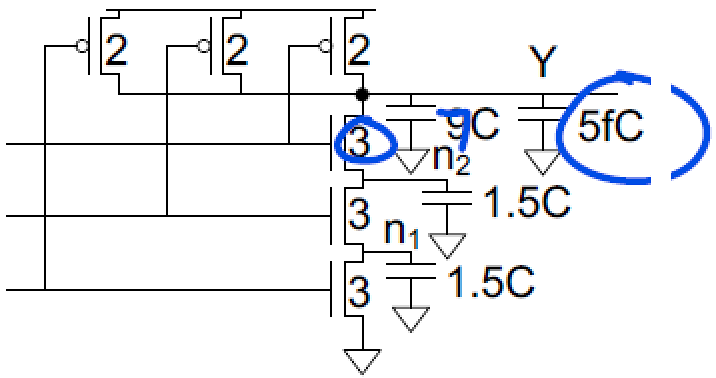


$$OUT = \overline{D + A \bullet (B + C)}$$

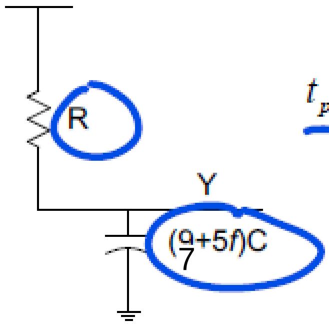
逻辑电路的Elmore Delay模型

- 拓展多级的RC模型 – 3-input NAND gates
  - Estimate worst-case rising and falling delay of 3-input NAND driving *f* identical gates.
  - R=13KΩ/unit
  - C=0.21fF/unit

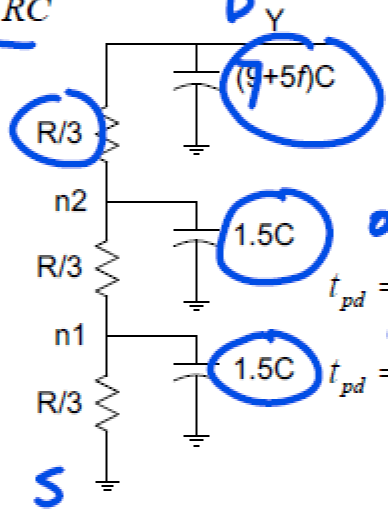




Worst-case pull up



$$t_{pu} = (9+5f)RC$$



Worst-case pull down

$$t_{pd} = 1.5C\frac{R}{3} + 1.5C\left(\frac{R}{3}+\frac{R}{3}\right) + \left[(9+5f)C\right]\left(\frac{R}{3}+\frac{R}{3}+\frac{R}{3}\right)$$

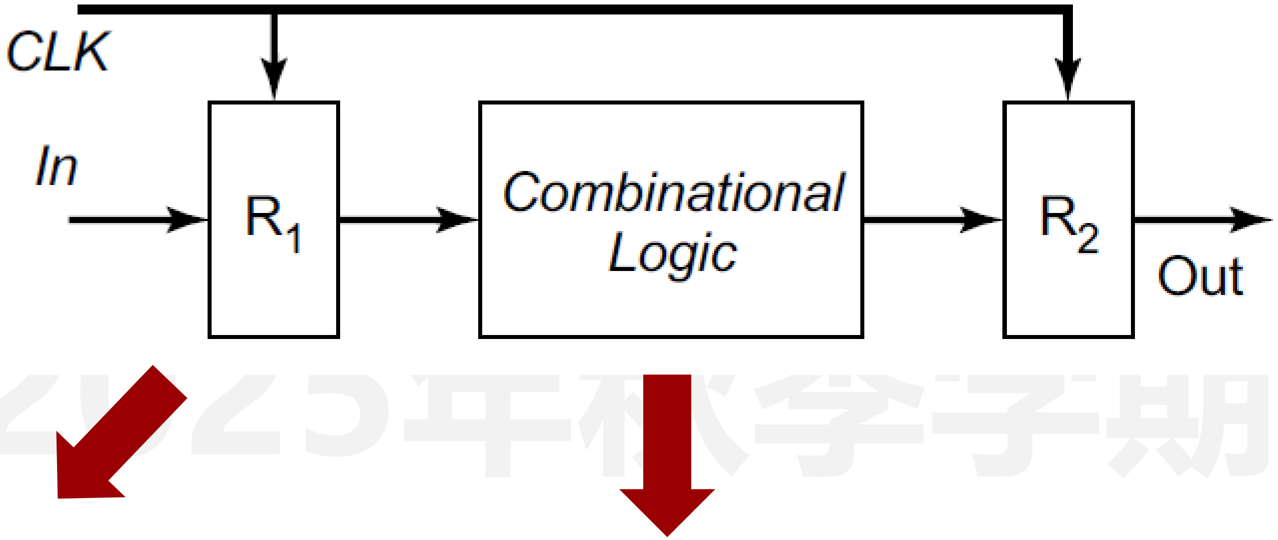
$$t_{pd} = (10.5+5f)RC$$

目 录

CONTENTS

电路时序的基本概念

- 同步时序（Synchronous Timing）

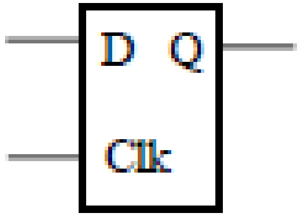


触发器
(Register)

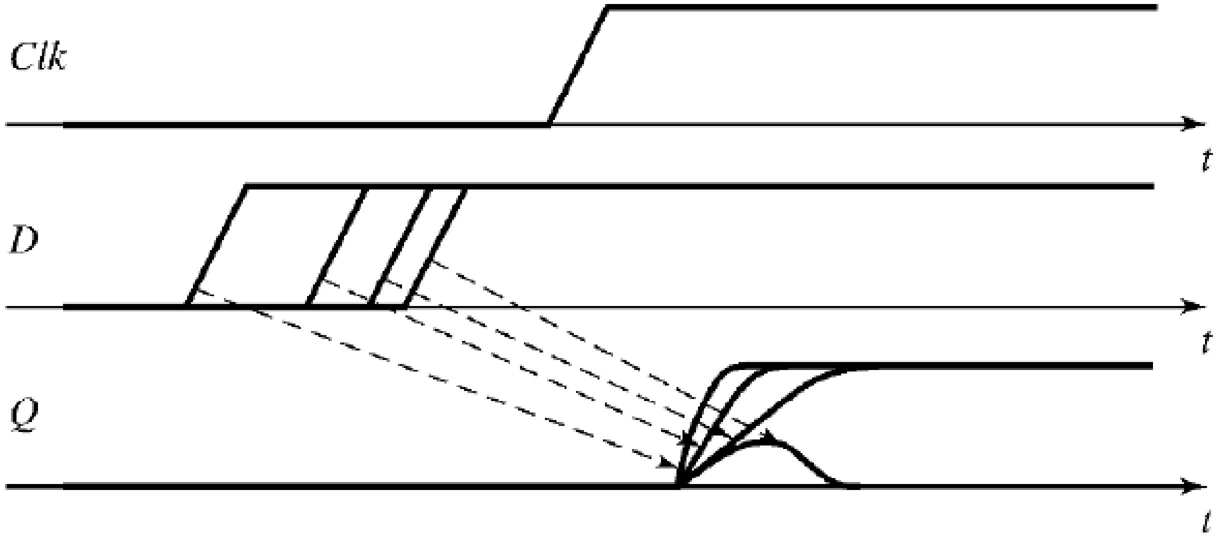
组合逻辑（各种逻辑门电路）

电路时序的基本概念

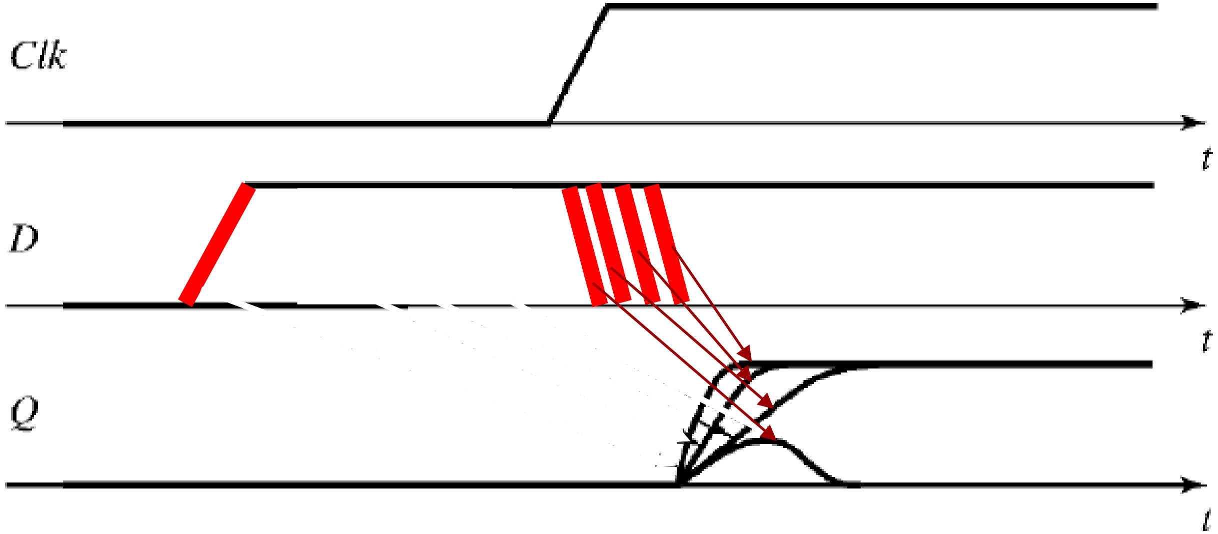
- 触发器的Setup Time和Hold Time



Setup Time

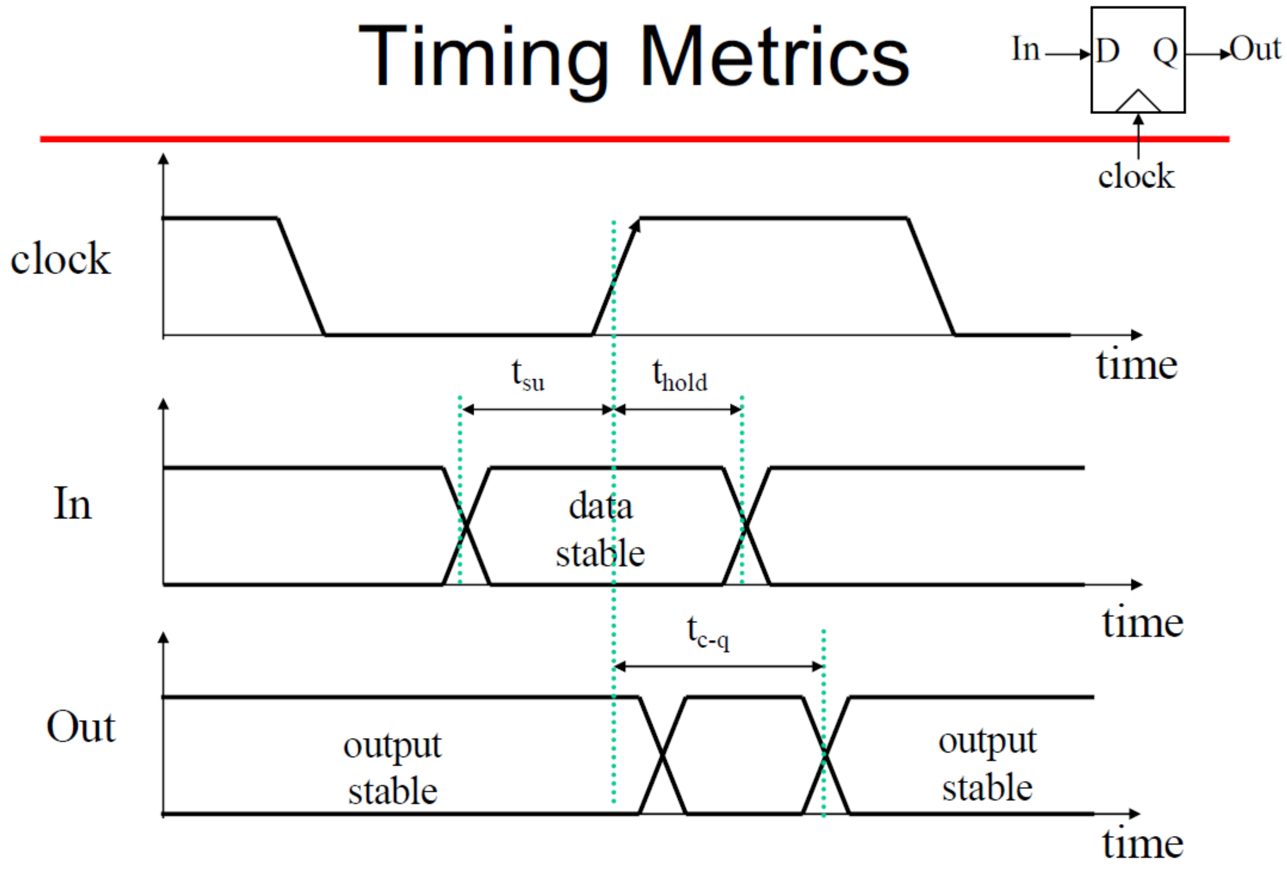


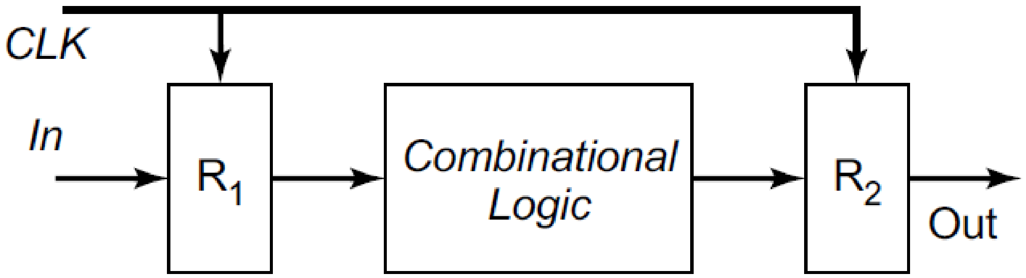
Hold Time



电路时序的基本概念

- 同步时序（Synchronous Timing）





$$T_{c\text{-}q} + t_{plogic,min} \geq t_{hold}$$

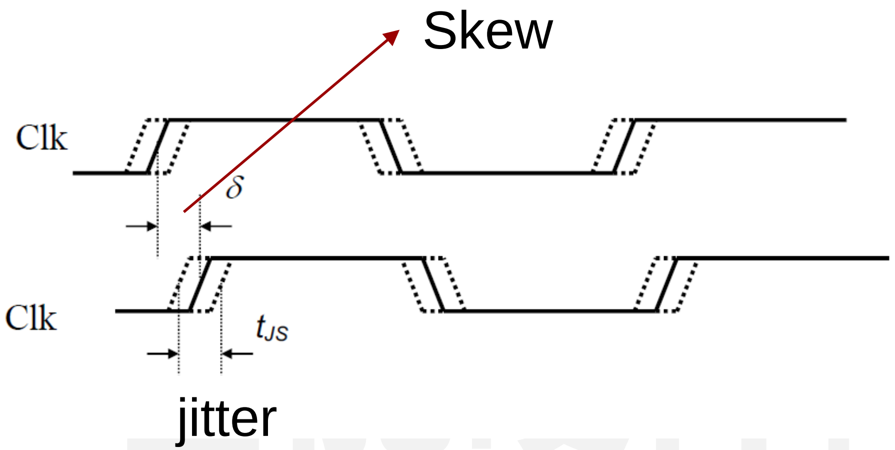
min

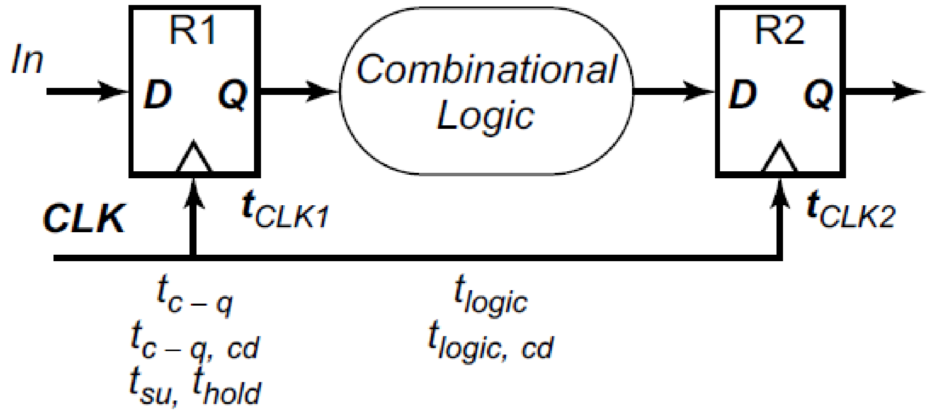
$$T \geq t_{c\text{-}q} + t_{plogic,max} + t_{su}$$

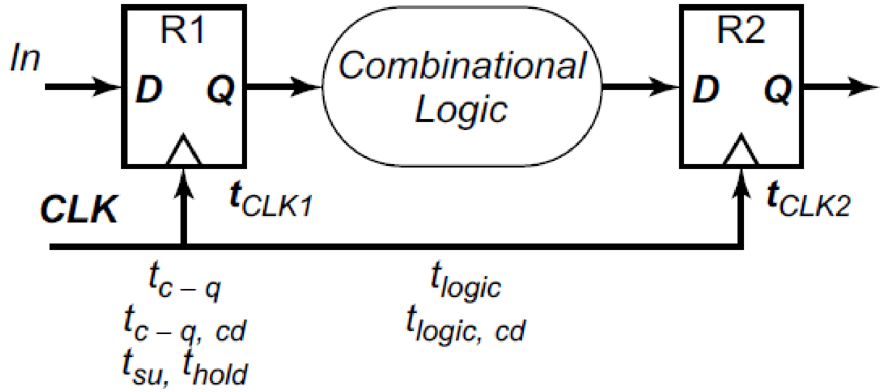
max

电路时序的基本概念

- **时钟的不稳定性**







Minimum cycle time:

$T \geq t_{c\text{-}q} + t_{su} + t_{logic} - \delta$

最坏情况为接收边沿过早到达（negative δ）

Hold time constraint:

$t_{(c\text{-}q,\ cd)} + t_{(logic,\ cd)} > t_{hold} + \delta$

最坏情况为接收边沿过晚到达（正偏差）

数据和时钟之间的竞争

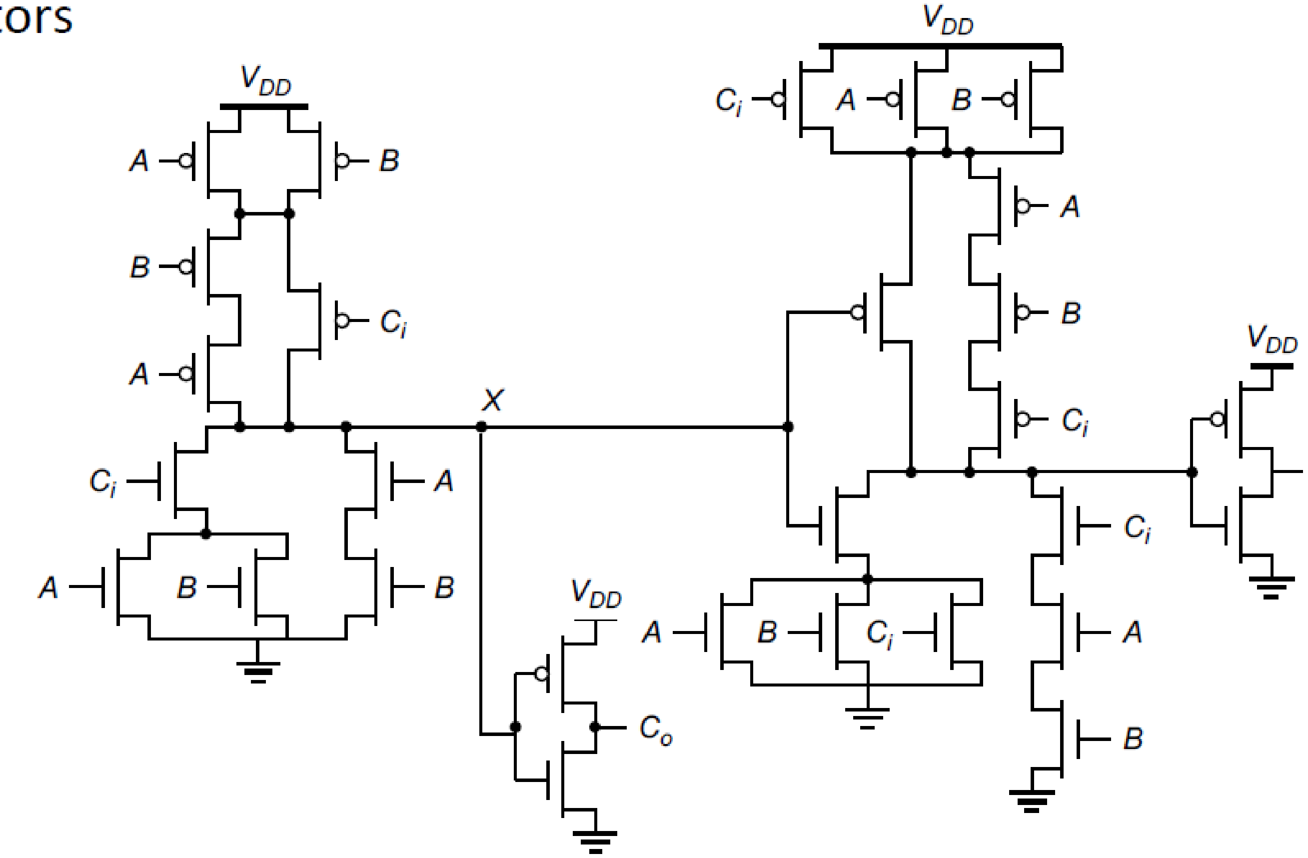
Cd：contamination delay（最快可能延迟）

目 录

CONTENTS

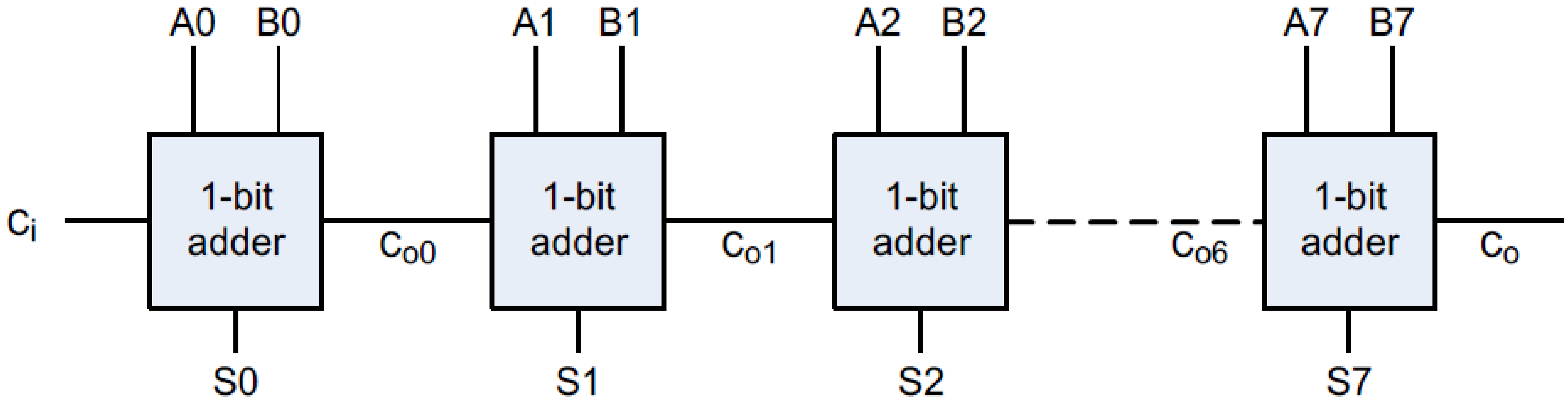
加法器设计

- 简单1bit加法器电路
  - $C_o = AB + BC_i + AC_i = AB + (A + B)C_i$
  - $S = A \oplus B \oplus C_i = ABC_i + \overline{C_o}(A + B + C_i)$
  - 28 transistors



加法器设计

- Ripple Carry加法器电路



最差延迟与比特数呈线性关系

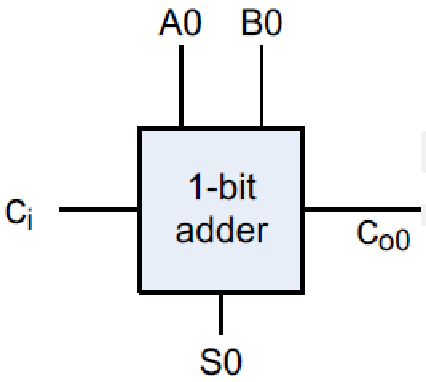
$$t_d = O(N)$$

$$t_{adder} = (N-1)t_{carry} + t_{sum}$$

Goal: Make the fastest possible carry path circuit

加法器设计

- 基于PGK的加法器设计方法



Generate (G) = AB

Propagate (P) = A ⊕ B

$$C_o(G,P) = G + PC_i$$

$$S(G,P) = P \oplus C_i$$

$\{ P = A \oplus B$; $P = A + B$

- Generate: C_{out} = 1 independent of C_{in}
  - G = A • B
- Propagate: $C_{out} = C_{in}$
  - P = A ⊕ B
- Kill: C_{out} = 0 independent of C_{in}
  - K = ~A • ~B

加法器设计

- **基于PGK的加法器设计方法**

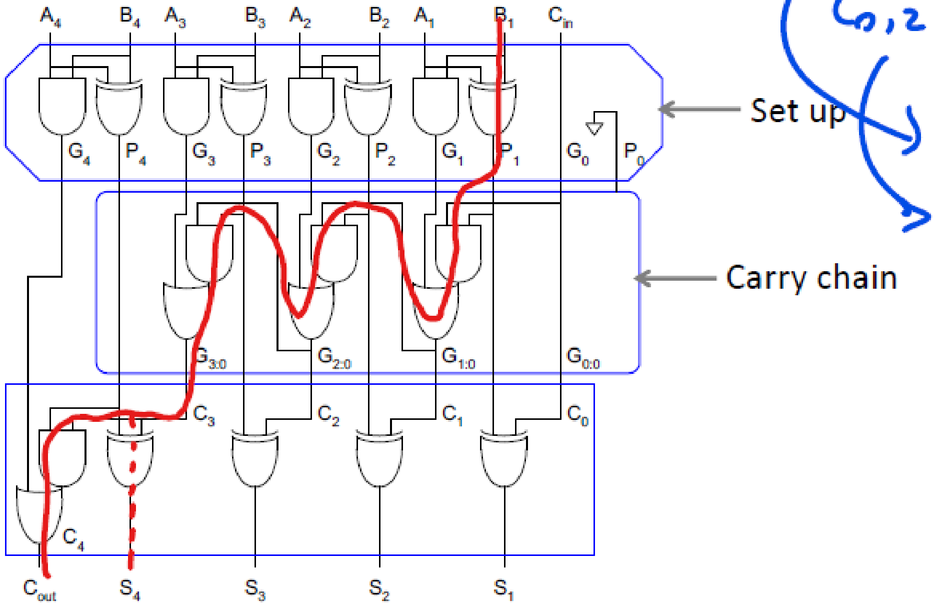
Carry-Ripple using P and G

$$C_{i:0} = G_i + P_i \cdot C_{i-1:0}$$

$$G_{0:0} = C_{in}$$

$$P_{0:0} = 0$$

$$C_{out,i} = G_{i:0}$$



$C_{0,1} = G_1 + P_1 C_{in}$

$C_{0,2} = G_2 + P_2 C_{0,1}$

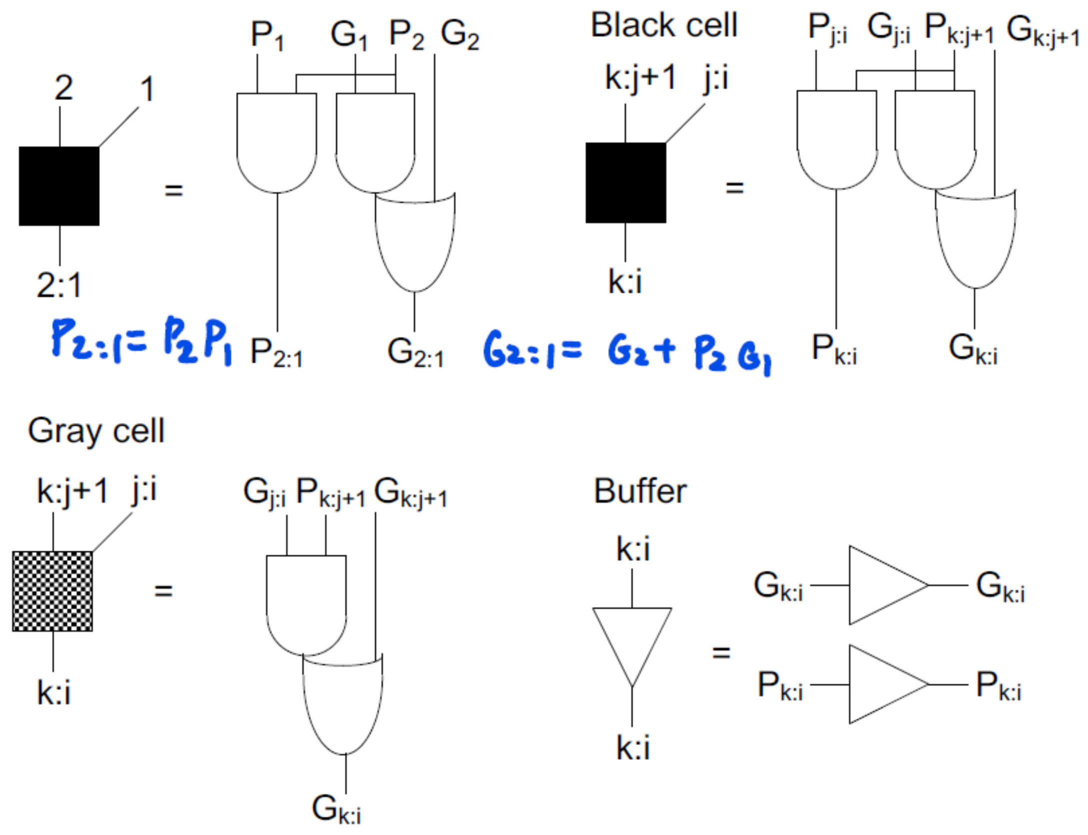
$G_{1:0} = G_1 + P_1 G_0$

$G_{2:0} = G_2 + P_2 G_{1:0}$

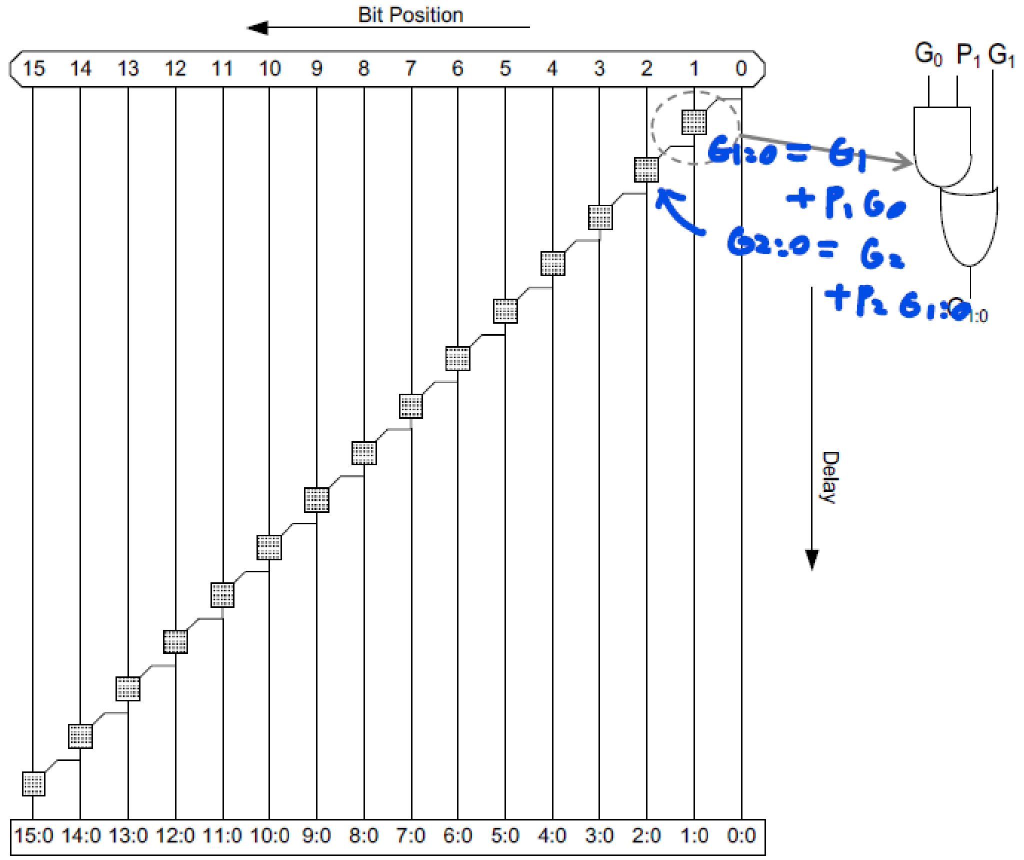
$$t_{adder} = t_{setup} + (N-1)\, t_{carry} + \max(t_{carry}, t_{sum})$$

加法器设计

- **基于PGK的加法器设计方法**



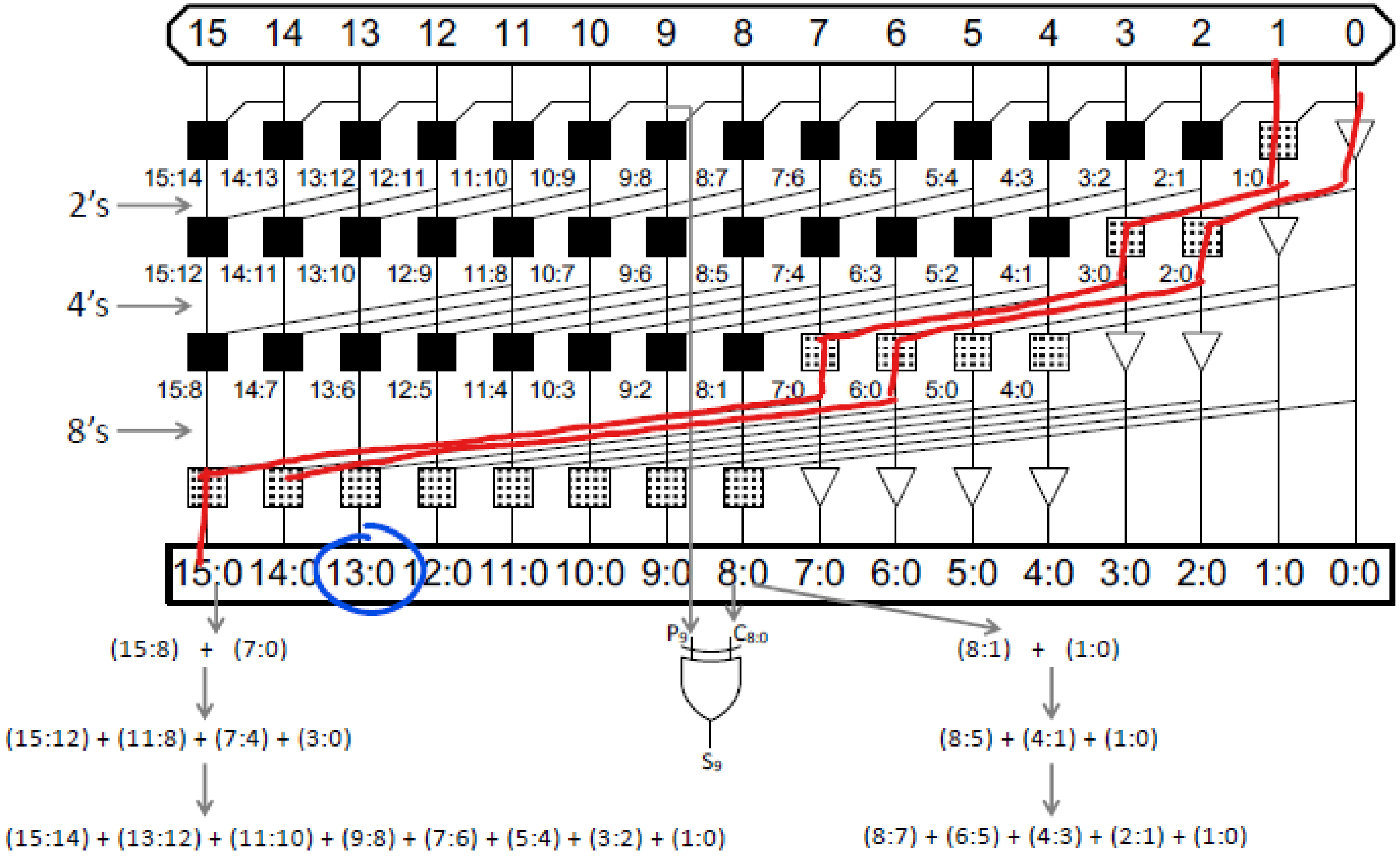
PG生成逻辑



Carry Ripple的PG图

下节课

- 基于PGK的加法器设计方法 – 复杂PG树加法器



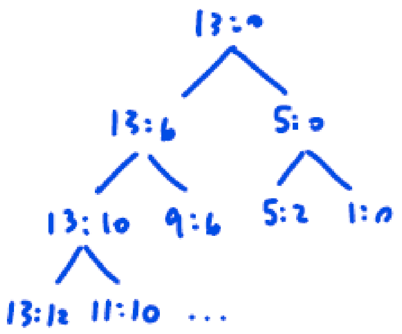
$\log_2(N)$

$G_{13:0} = G_{13:6} + P_{13:6}\, G_{5:0}$

$G_{13:6} = G_{13:10} + P_{13:10}\, G_{9:6}$

$P_{13:6} = P_{13:10}\, P_{9:6}$

$G_{5:0} = G_{5:2} + P_{5:2}\, G_{1:0}$

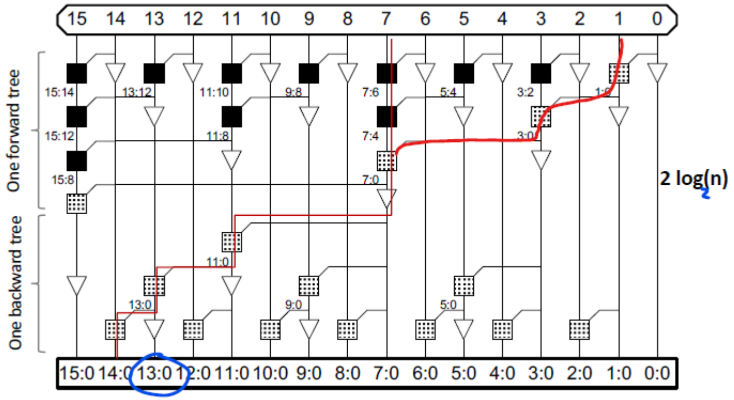


作业题

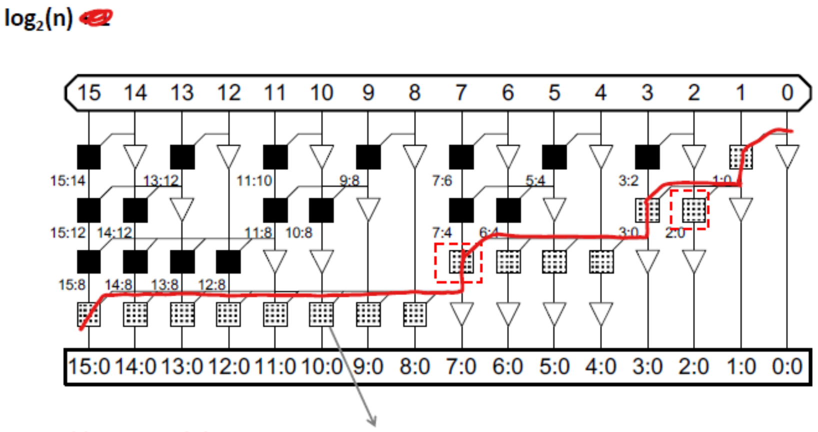
加法器设计

- **基于PGK的加法器设计方法 – 复杂PG树加法器**

Brent-Kung

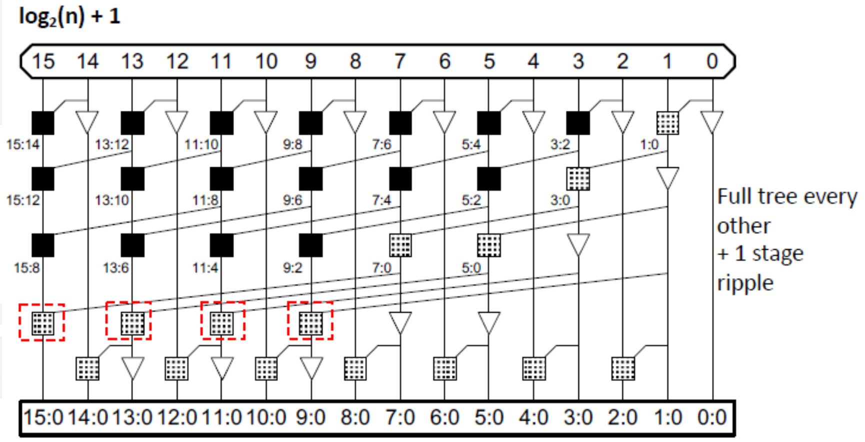


Sklansky



- Uneven sizing
- Large fanout

Han-Carlson



Low fanout, tradeoff between logic levels and wiring

Reduces wire length by half ! → half power compared to Kogge Stone

- Kogge-Stone: low logic levels, low fanout, high wiring
- Brent-Kung: low fanout, low wiring, high logic levels
- Sklansky: low logic levels, low wiring, high fanout

乘法器设计

- 乘法器设计的核心是部分和累加

Example:

```
    1100   : 12_10   multiplicand
    0101   : 5_10    multiplier
    ----
    1100             ┐
   0000              │ partial
  1100               │ products
 0000                ┘
--------
00111100   : 60_10   product
```

M x N比特乘法

- 产生N个M比特部分乘积
- 求和得到M+N比特的结果

乘法器设计

- **乘法器设计的核心是部分和累加**

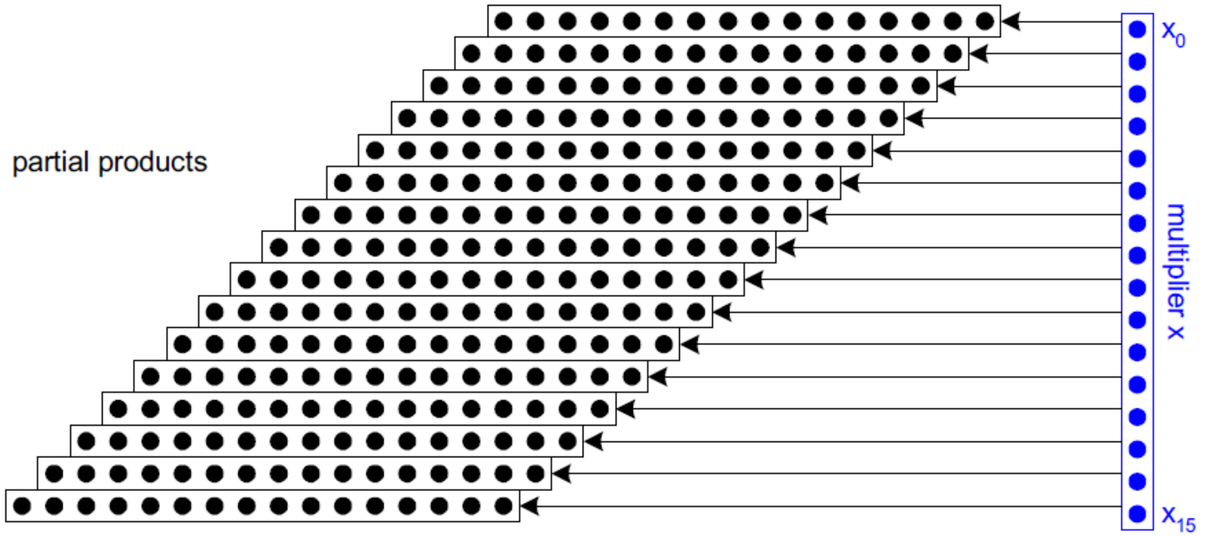
Multiplicand: $Y = (y_{M-1}, y_{M-2}, \ldots, y_1, y_0)$

Multiplier: $X = (x_{N-1}, x_{N-2}, \ldots, x_1, x_0)$

Product: $$P = \left(\sum_{j=0}^{M-1} y_j 2^j\right)\left(\sum_{i=0}^{N-1} x_i 2^i\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

						y_5	y_4	y_3	y_2	y_1	y_0	multiplicand
						x_5	x_4	x_3	x_2	x_1	x_0	multiplier
						x_0y_5	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0	partial products
					x_1y_5	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0		
				x_2y_5	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0			
			x_3y_5	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0				
		x_4y_5	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0					
	x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_5y_1	x_5y_0						
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	product

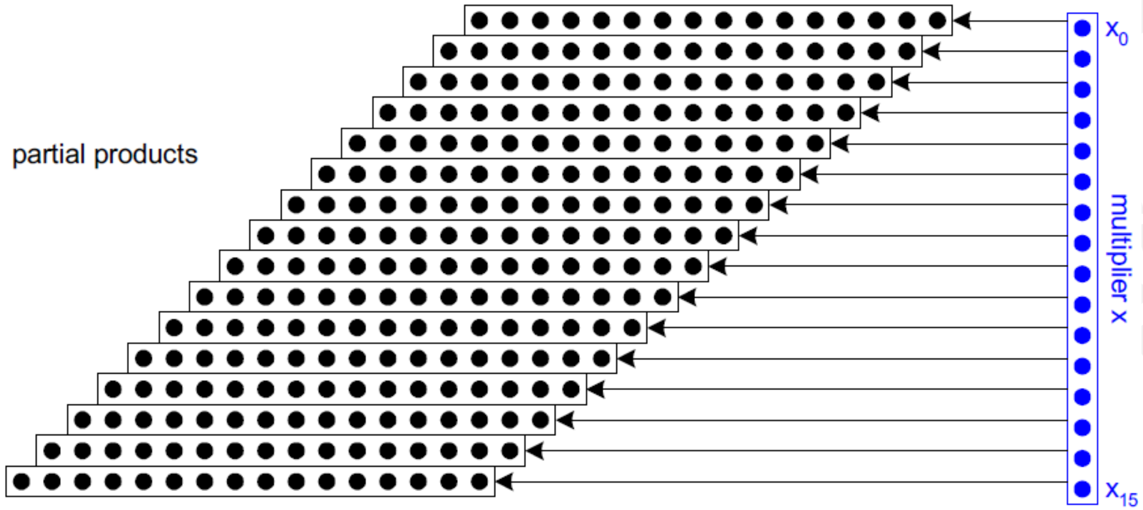
Each dot represents a bit

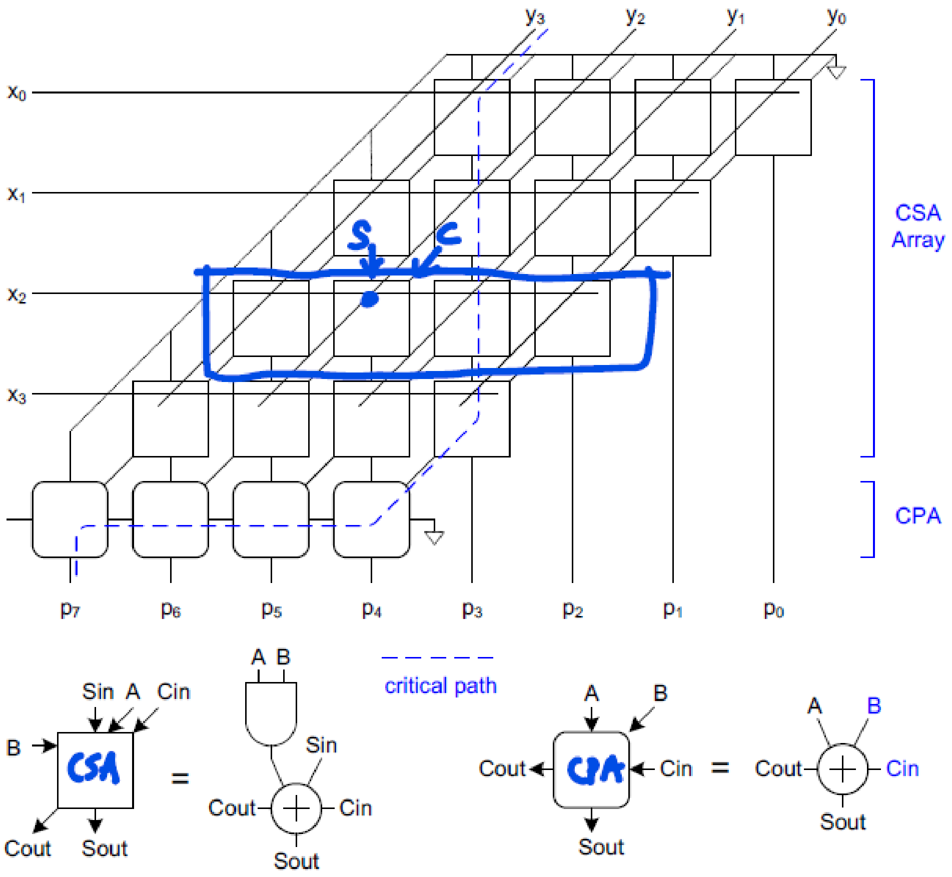


乘法器设计

- 乘法器设计的核心是部分和累加

Each dot represents a bit





乘法器设计

- 如何减少部分和累加的次数?

- Array multiplier requires N partial products
- If we looked at groups of r bits, we could form N/r partial products.
  - Faster and smaller?
  - Called radix-2^r encoding
- Ex: r = 2: look at pairs of bits
  - Form partial products of 0, Y, 2Y, 3Y
  - First three are easy, but 3Y requires adder ☹

```
        1   1   0   0
      ( 0   1)( 0   1
      ---------------
        a   a   a   a
    b   b   b   b
    ---------------
```

x	PP
(0 0)	0
(0 1)	y
(1 0)	2y (4y-2y)
(1 1)	3y = (4y - y)

乘法器设计

- 如何减少部分和累加的次数 – 布斯编码（Radix-2^r)

  - Instead of 3Y, try –Y, then increment next partial product to add 4Y
  - Similarly, for 2Y, try –2Y + 4Y in next partial product

Inputs			Partial Product	Booth Selects		
x_{2i+1}	x_{2i}	x_{2i-1}	PP_i	$SINGLE_i$	$DOUBLE_i$	NEG_i
0	0	0	0	0	0	0
0	0	1	Y	1	0	0
0	1	0	Y	1	0	0
0	1	1	$2Y$	0	1	0
1	0	0	$-2Y$	0	1	1
1	0	1	$-Y$	1	0	1
1	1	0	$-Y$	1	0	1
1	1	1	$-0\ (=0)$	0	0	1

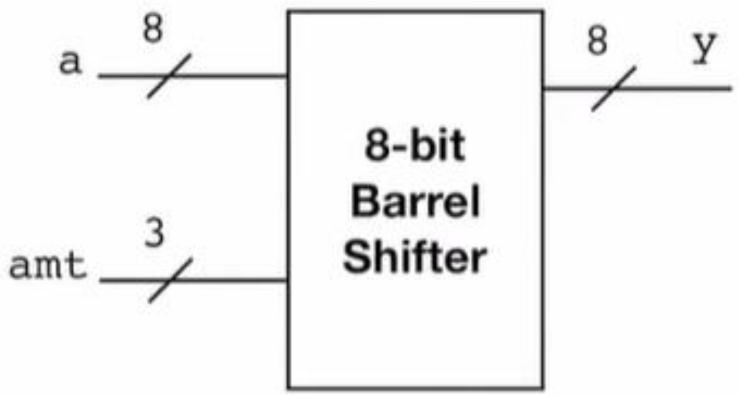
作业题

位移器设计

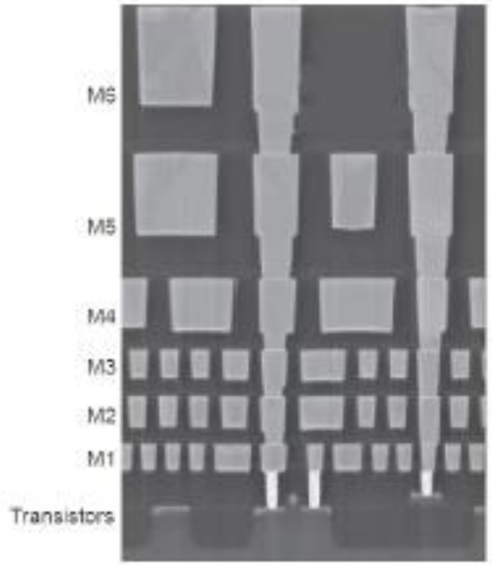
- Shifter也是重要的数字电路模块之一

```
module barrel_shifter
    (
        input logic [7:0] a,
        input logic [2:0] amt,
        output logic [7:0] y
    );

    always_comb
        case(amt)
            3'b000: y = a;
            3'b001: y = {a[0], a[7:1]};
            3'b010: y = {a[1:0], a[7:2]};
            3'b011: y = {a[2:0], a[7:3]};
            3'b100: y = {a[3:0], a[7:4]};
            3'b101: y = {a[4:0], a[7:5]};
            3'b110: y = {a[5:0], a[7:6]};
            3'b111: y = {a[6:0], a[7]};
            default: y = a;
        endcase
endmodule
```



- **Wire Geometry**



Intel 90 nm Stack

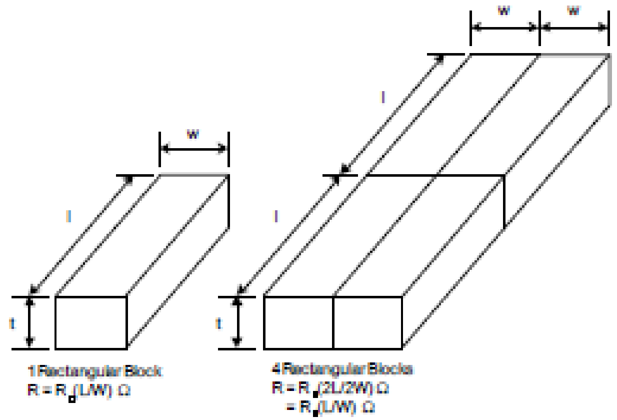
Intel 45 nm Stack

线电阻的计算方式

- ρ = *resistivity* (Ω*m)

$$R = \frac{\rho}{t}\frac{l}{w} = R_{\square}\frac{l}{w}$$

- R□ = *sheet resistance* (Ω/□)
  - □ is a dimensionless unit(!)
- Count number of squares
  - R = R□ * (# of squares)

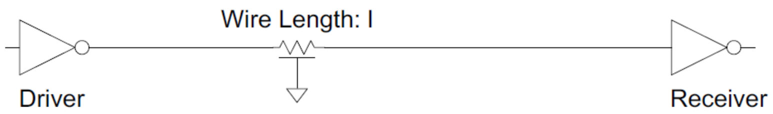


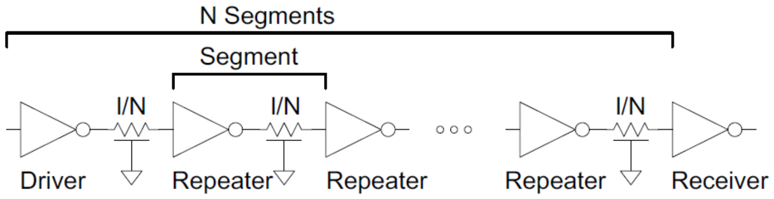
线路分析

• Wire Repeaters

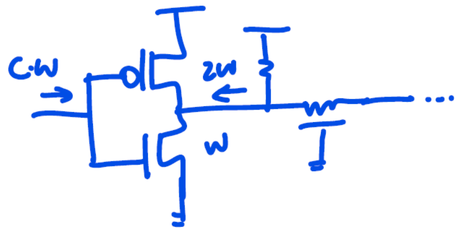
- R和C正比与l
- RC延迟正比于l平方
  --对于长走线是完全不可接受的

- 将长走线切分为N段更短的走线
  --通过反相器或者缓冲器来驱动每一段走线

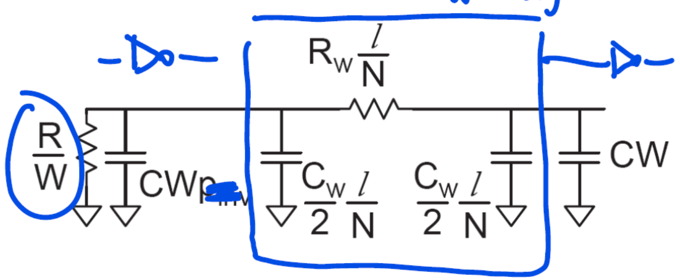




- How many repeaters should we use?
- How large should each one be?
- Equivalent Circuit
  – Wire length l/N
    - Wire Capacitance C_w*l/N, Resistance R_w*l/N
  – Inverter width W (nMOS = W, pMOS = 2W)
    - Gate Capacitance C*W, Resistance R/W



$t_p = N\left(0.69\left(\frac{R}{W}CW + \frac{R}{W}\frac{C_w}{2}\frac{l}{N} + \left(\frac{R}{W} + R_w\frac{l}{N}\right)\left(\frac{C_w}{2}\frac{l}{N} + CW\right)\right)\right)$



- **Wire Repeaters**

- Write equation for Elmore Delay
  - Differentiate with respect to W and N
  - Set equal to 0, solve

$$\frac{l}{N} = \sqrt{\frac{2RC'}{R_w C_w}}$$

unit wire segment

unit inv resistance

unit inv cap

unit wire cap

unit wire res

$C' = C(1 + p_{inv})$

$$W = \sqrt{\frac{RC_w}{R_w C'}}$$

作业题