计算机体系结构

胡伟武、汪文祥

二进制与逻辑电路

- 计算机中数的表示
- **CMOS门电路及工艺**
- **CMOS组合逻辑与时序逻辑**
- **CMOS电路延迟**
- **从Verilog到GDSII**
- 其它“**0**”和“**1**”表示方法

计算机中数的表示

计算机中数的表示

- 二进制
  - 最容易逻辑实现
  - 自然界中的二值系统较多
- “1”和“0”的表示
  - 用电压的高低表示，半导体工艺，CMOS
  - 用磁通量的有无表示，超导体工艺
  - 用能级的高低表示，量子计算机
  - 用基因序列表示，A, G, C, T, DNA计算机，非二进制？

定点数的表示（1）

- 原码： $A=a_{n-1}\ a_{n-2}\ldots\ a_1\ a_0$表示
  - 最高位a_{n-1}为符号位，0表示正，1表示负。
  - 其它位$a_{n-2}\ldots\ a_1\ a_0$表示数值。
  - 原码的问题：加减法效率低，两个“0”
- 补码
  - 本质是取模运算，如-2%12=10
  - 最高位a_{n-1}为符号位，0表示正，1表示负。
  - $A=a_{n-1}\ a_{n-2}\ldots\ a_1\ a_0$表示$(-2^{n-1}\ a_{n-1}+a_{n-2}\ldots\ a_1\ a_0)$
    - $a_{n-1}==0$时，补码和原码一样， A表示正$a_{n-2}\ldots\ a_1\ a_0$ 。
    - $a_{n-1}==1$时， A表示$(-2^{n-1}+a_{n-2}\ldots\ a_1\ a_0)$ 。
- 原码与补码的转换
  - 最高位为0时，一样
  - 最高位为1时，最高位不变，其余位“按位取反加一”。

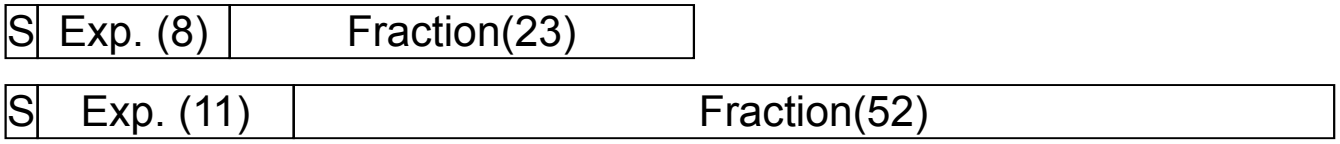
定点数的表示（2）

- 补码运算
  - 取负数，每一位(包括符号位)求补，即按位取反加一。
  - **A-B=A+B的负数=A+(B求补)**
  - 加法溢出判断：**A和B的最高位一样，且结果的最高位与A和B的最高位不一样。**
  - **1001+0101(-7+5)=1110, 1100+0100(-4+4)=0000,**
  - **0011+0100(3+4)=0111, 1100+1111(-4-1)=1011,**
  - **0101+0100(5+4)=1001, 1001+1010(-7-6)=0011**

	+18	-18
8 位原码	00010010	10010010
8 位补码	00010010	11101110
16 位原码	0000000000010010	1000000000010010
16 位补码	0000000000010010	1111111111101110

浮点数的表示（1）

- 定点数的不足
  - 表示范围有限，太大或太小的数都不能表示
  - 除法不精确
- 浮点数的表示：**IEEE 754**标准
  - 三部分组成：符号位，阶码**(exponent)**，尾数**(fraction)**
  - 最高位是符号位
  - 阶码的移码表示，底为**2**，$2^{(\text{阶码-偏移值})}$
  - 规格化表示，尾数的最高位总为**1**，因此可以不存
  - 单精度和双精度
  - 扩展的单双精度



浮点数的表示（2）

- **IEEE 754 浮点格式参数**

参数	单精度	扩展单精度	双精度	扩展双精度
字长	32	>=43	64	>=79
阶码位数	8	>=11	11	>=15
阶码偏移量	127	未定	1023	未定
最大阶码	127	>=1023	1023	>=16383
最小阶码	-126	<=-1022	-1022	<=-16382
表示范围	10^{-38}, 10^{+38}	未定	10^{-308}, 10^{+308}	未定
尾数位数	23	>=31	52	>=63
阶码个数	254	未定	2046	未定
尾数个数	2^{23}	未定	2^{52}	未定
值的个数	$1.98*2^{31}$	未定	$1.99*2^{63}$	未定

浮点数的表示（3）

- **IEEE 754 浮点格式**

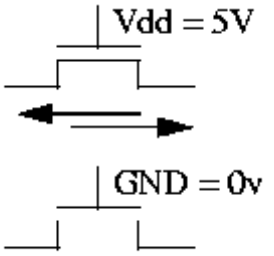
	单精度				双精度			
	符号位	阶码	尾数	值	符号位	阶码	尾数	值
正无限	0	255	0	∞	0	2047	0	∞
负无限	1	255	0	-∞	1	2047	0	-∞
Quiet NaN	0 或 1	255	高位=0	NaN	0 或 1	2047	高位=0	NaN
Signaling NaN	0 或 1	255	高位=1	NaN	0 或 1	2047	高位=1	NaN
正规格化非 0	0	0<e<255	f	$2^{e-127}(1.f)$	0	0<e<2047	f	$2^{e-1023}(1.f)$
负规格化非 0	1	0<e<255	f	$-2^{e-127}(1.f)$	1	0<e<2047	f	$-2^{e-1023}(1.f)$
正非规格化非 0	0	0	f!=0	$2^{e-126}(0.f)$	0	0	f!=0	$2^{e-1022}(0.f)$
负非规格化非 0	1	0	f!=0	$-2^{e-126}(0.f)$	1	0	f!=0	$-2^{e-1022}(0.f)$
正 0	0	0	0	0	0	0	0	0
负 0	1	0	0	-0	1	0	0	-0

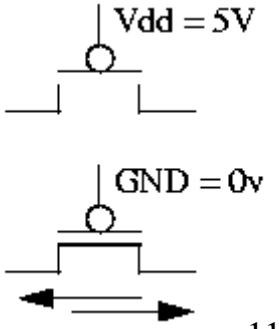
- 把 Singaling NaN 当作操作数引起例外，把 Quiet NaN 当作操作数结果也为 Quiet NaN，可以用尾数部分区分它们。
- 非规格化数用于阶码下溢的情况，填补最小数和 0 之间的“空隙”。

CMOS电路及工艺

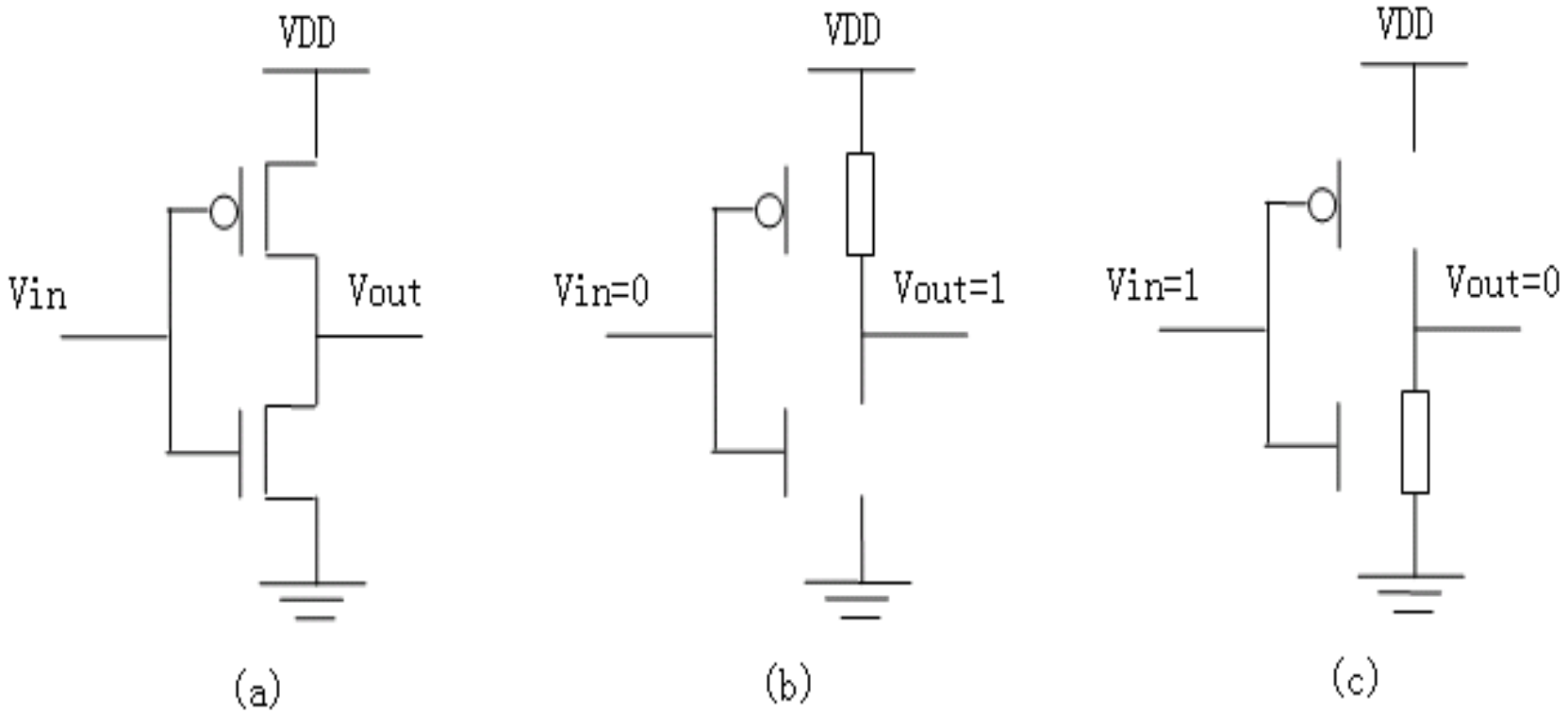
CMOS电路---P管与N管

- **CMOS: Complementary Metal Oxide Semiconductor**
  - **NMOS: N-type Metal Oxide Semiconductor**
  - **PMOS: P-type Metal Oxide Semiconductor**
- **NMOS管**
  - 门电压为高时导通
  - 门电压为低时关闭
- **PMOS管**
  - 门电压为高时关闭
  - 门电压为低时导通



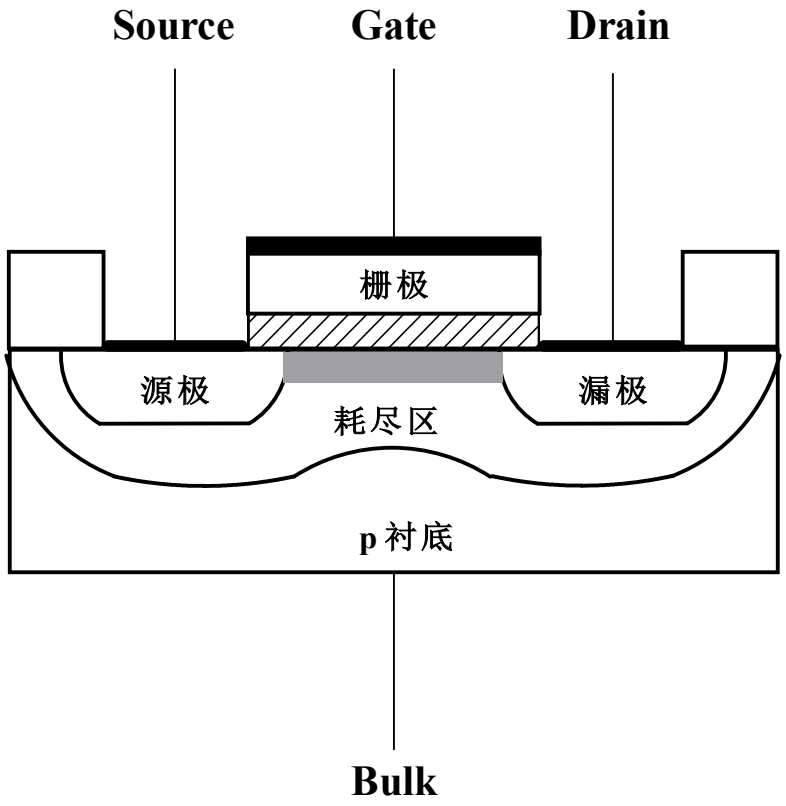


CMOS电路----反相器

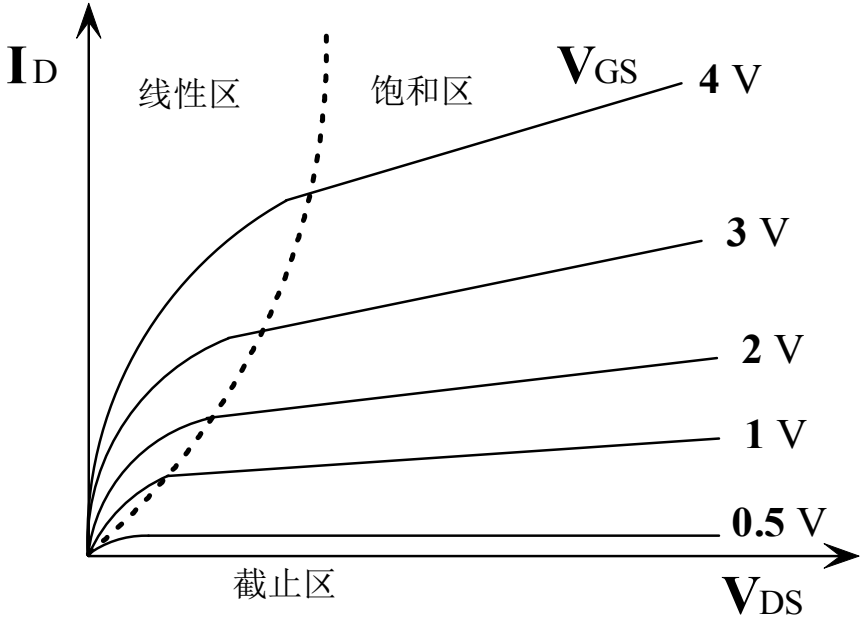


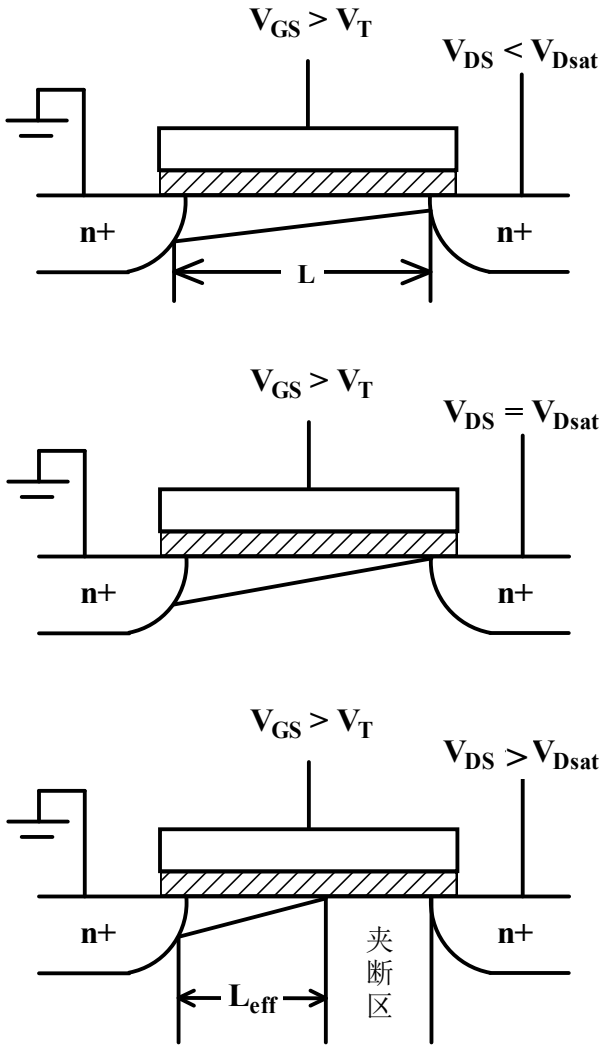
N沟道MOS晶体管的示意图

- 栅极不加电时，源漏之间是一对PN结，不导通
- 栅极加电后，吸引电子到栅氧化层下面，形成导电的沟道。
  - 附近形成耗尽区

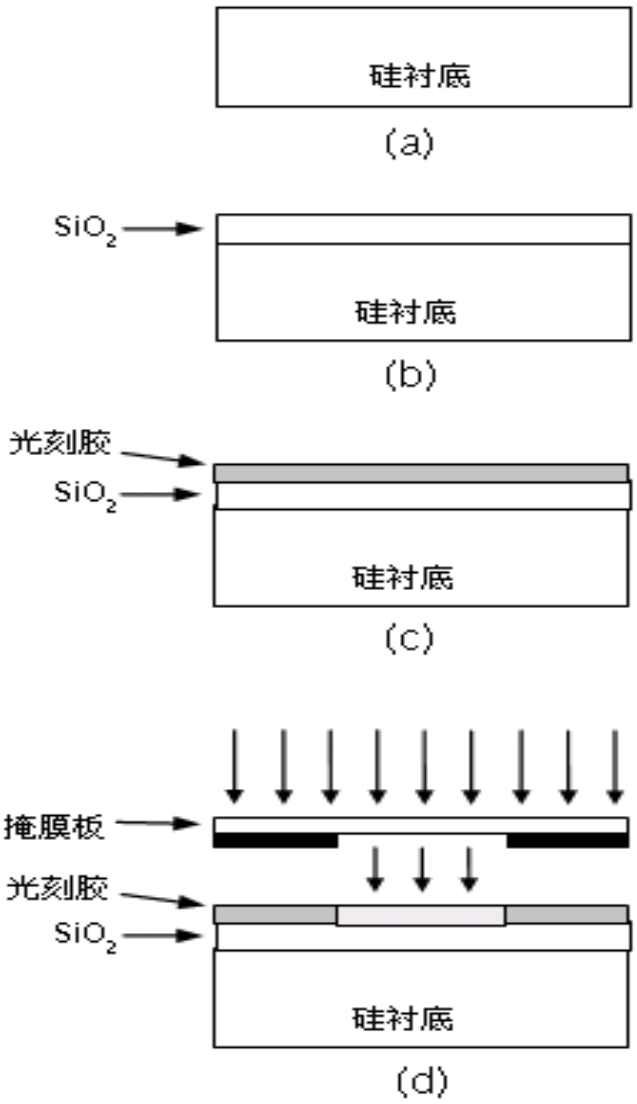


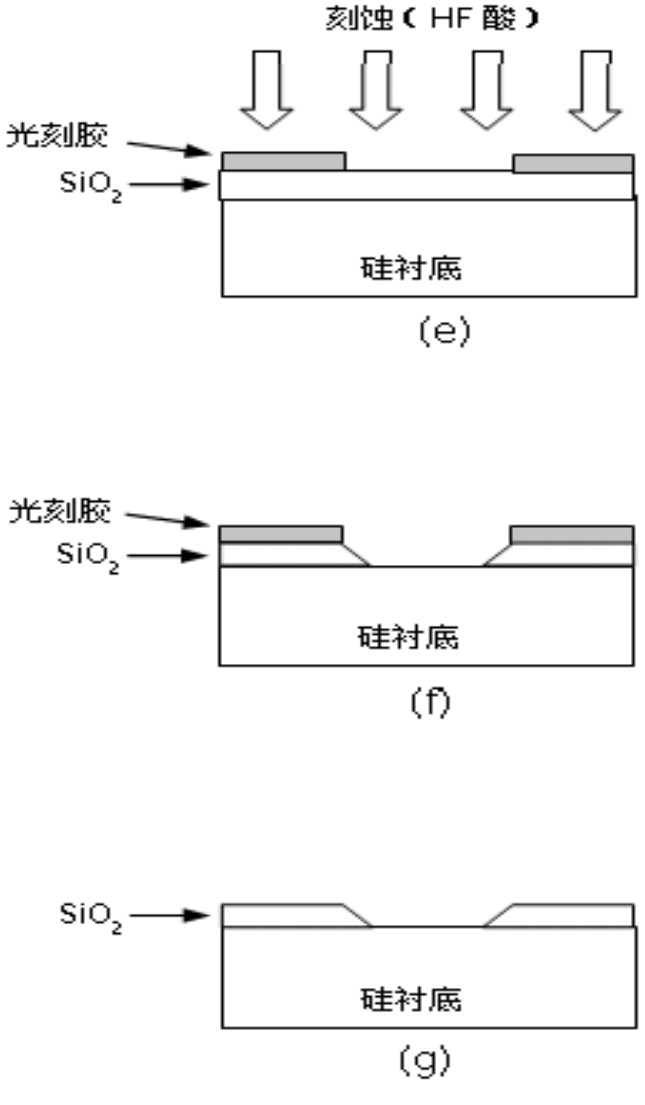
MOS晶体管的工作状态



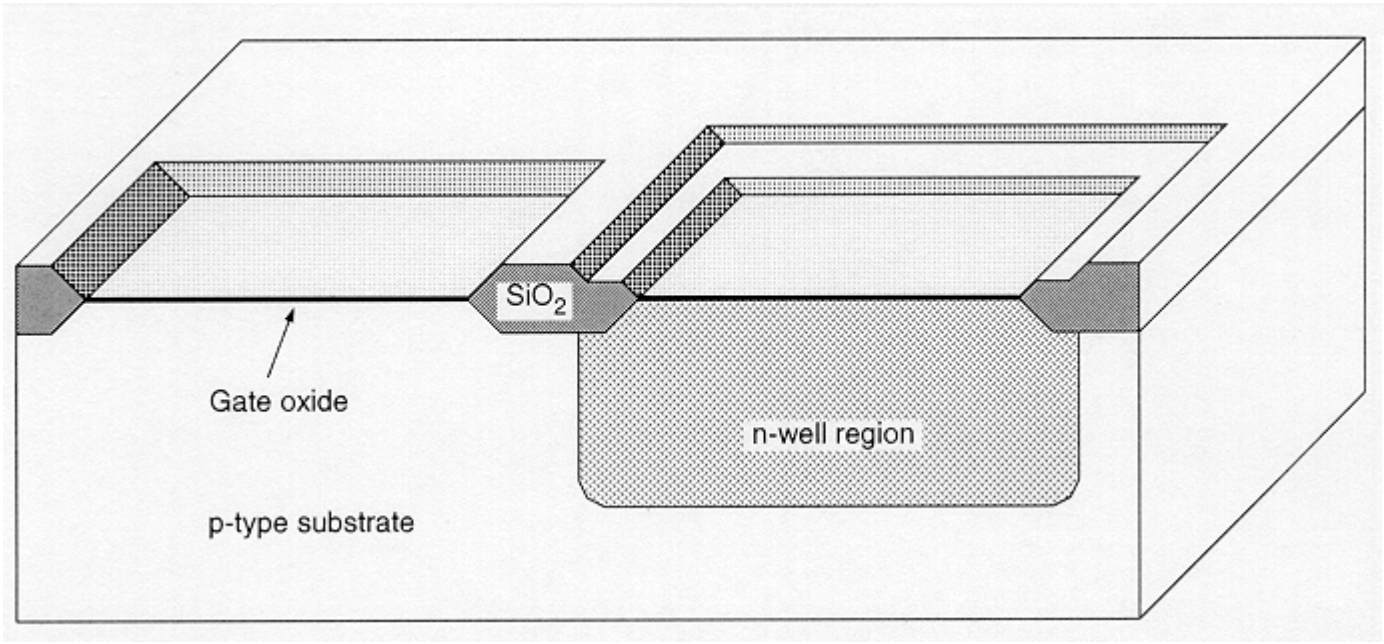


MOS基本工艺---光刻

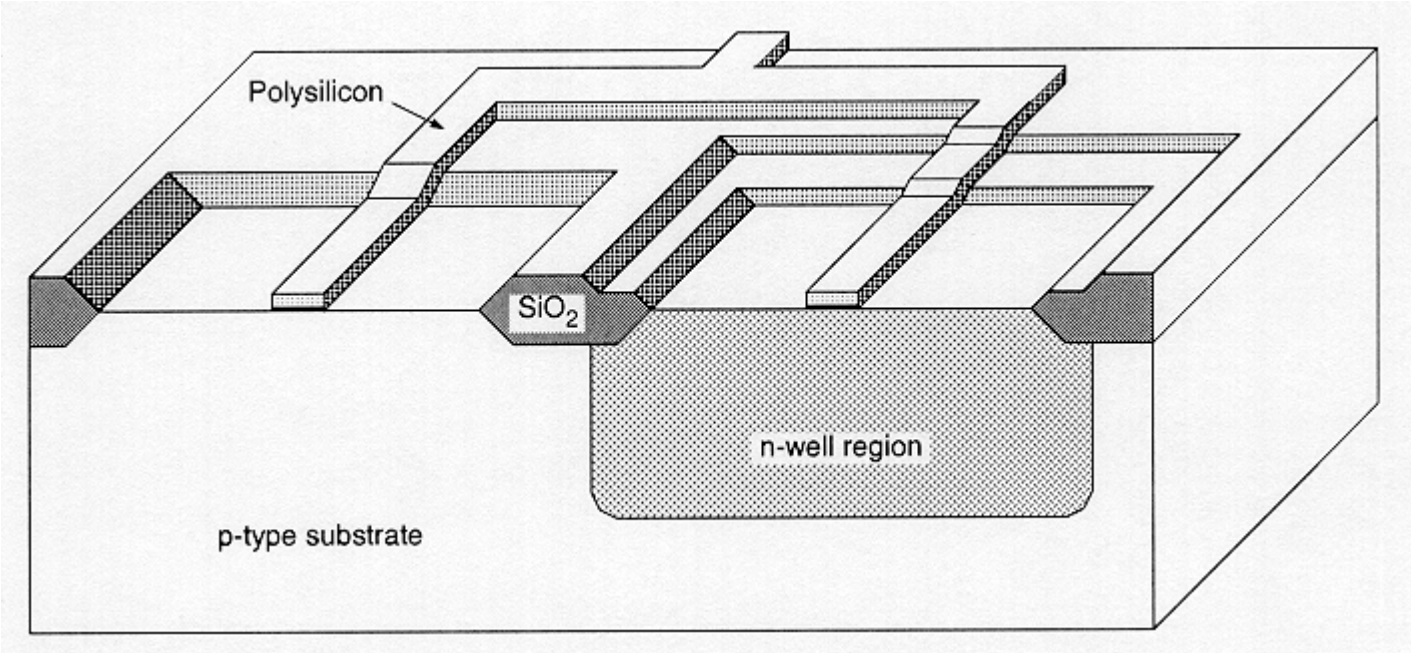

硅衬底
(a)
SiO_2
硅衬底
(b)
光刻胶
SiO_2
硅衬底
(c)
掩膜板
光刻胶
SiO_2
硅衬底
(d)


刻蚀（HF 酸）
光刻胶
SiO_2
硅衬底
(e)
光刻胶
SiO_2
硅衬底
(f)
SiO_2
硅衬底
(g)

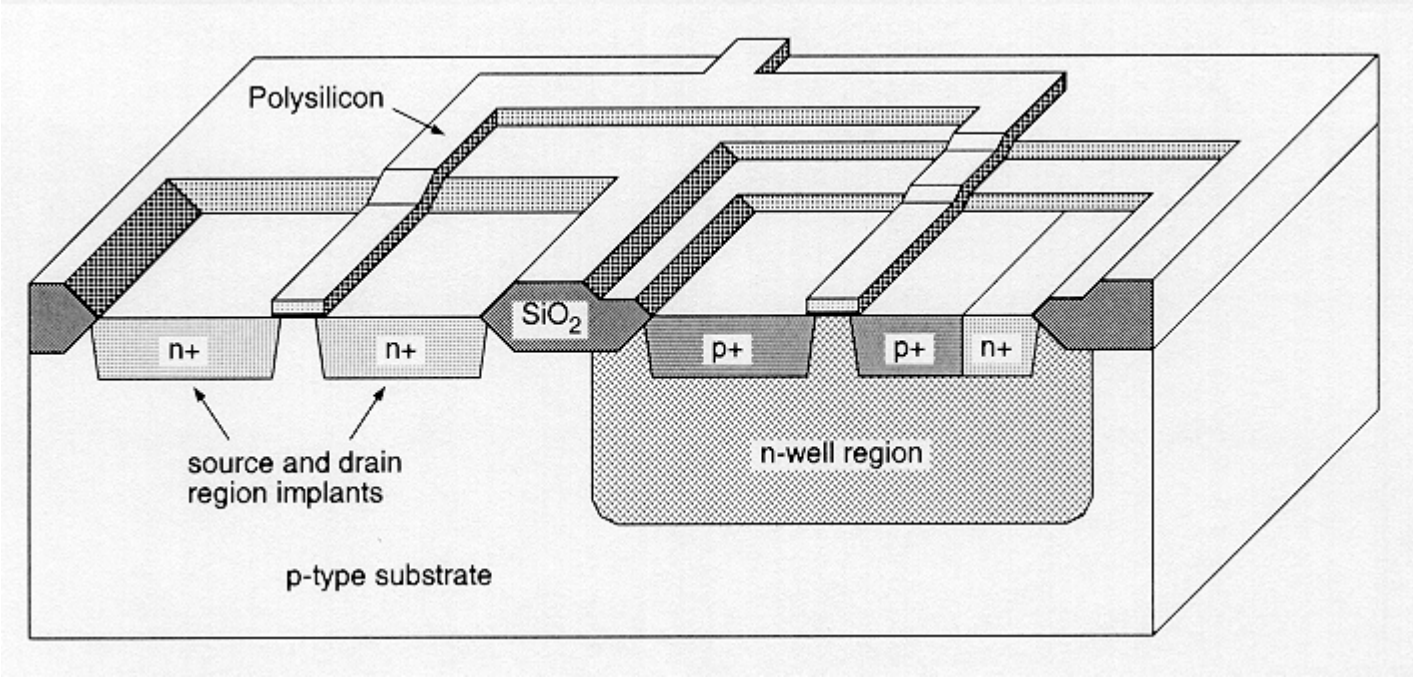
P衬底 nWell CMOS工艺（1）



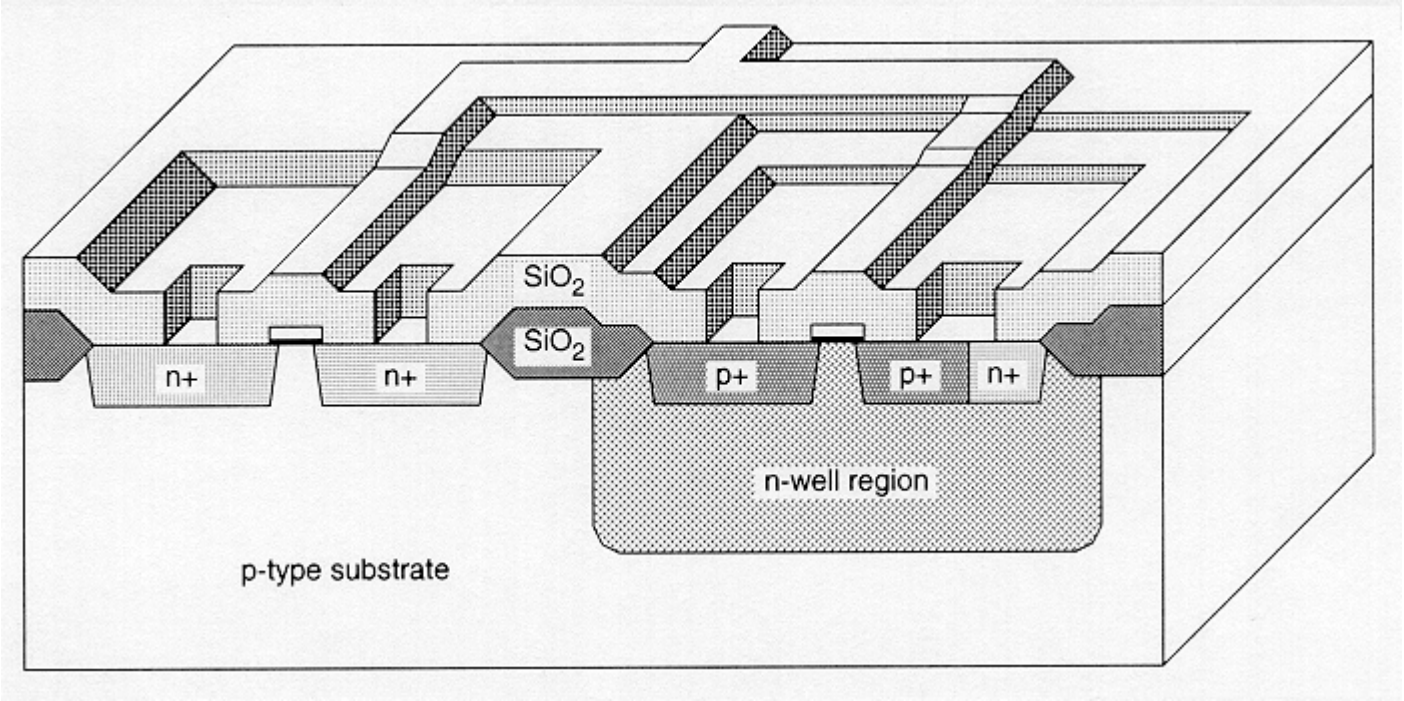
P衬底 nWell CMOS工艺（2）



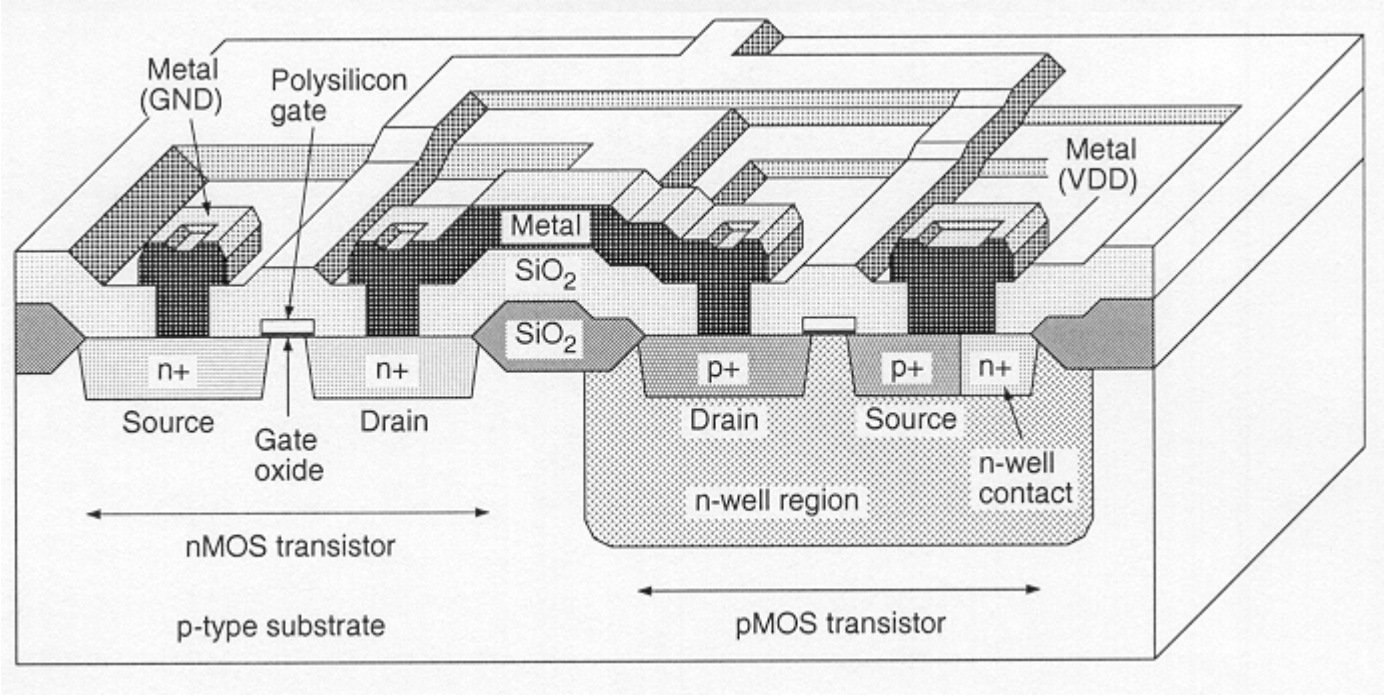
P衬底 nWell CMOS工艺（3）



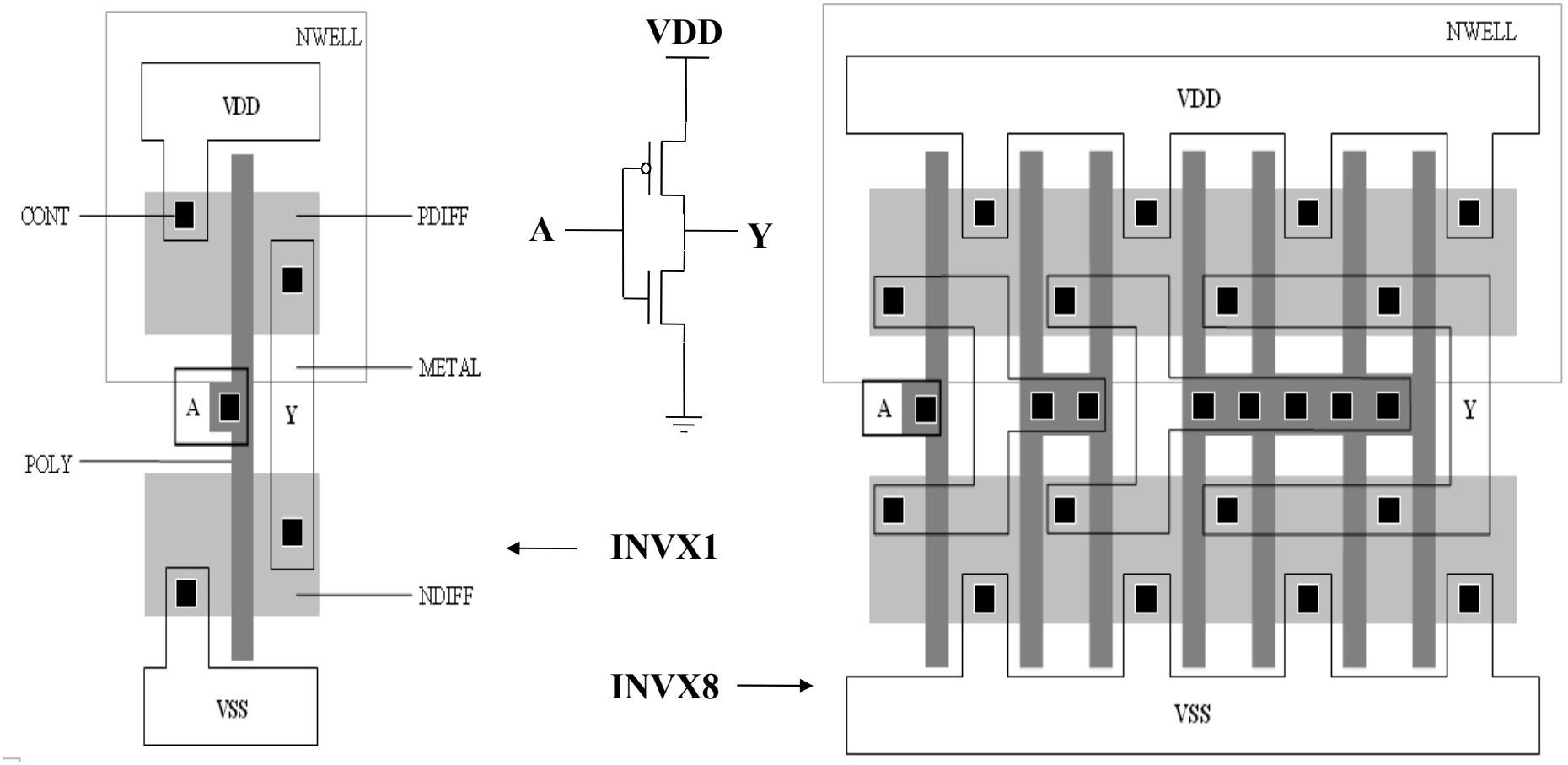
P衬底 nWell CMOS工艺（4）



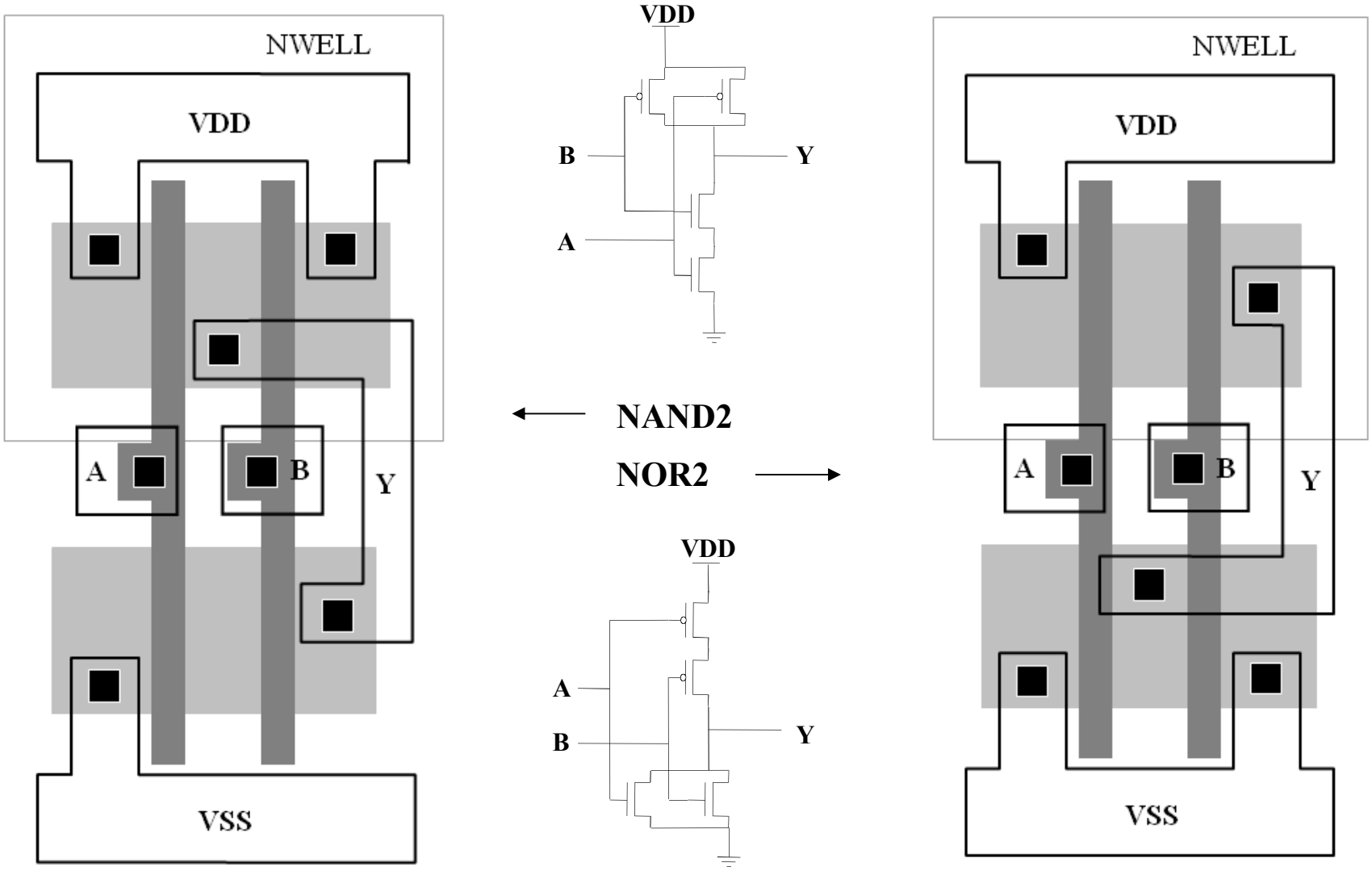
P衬底 nWell CMOS工艺（5）



反相器的版图

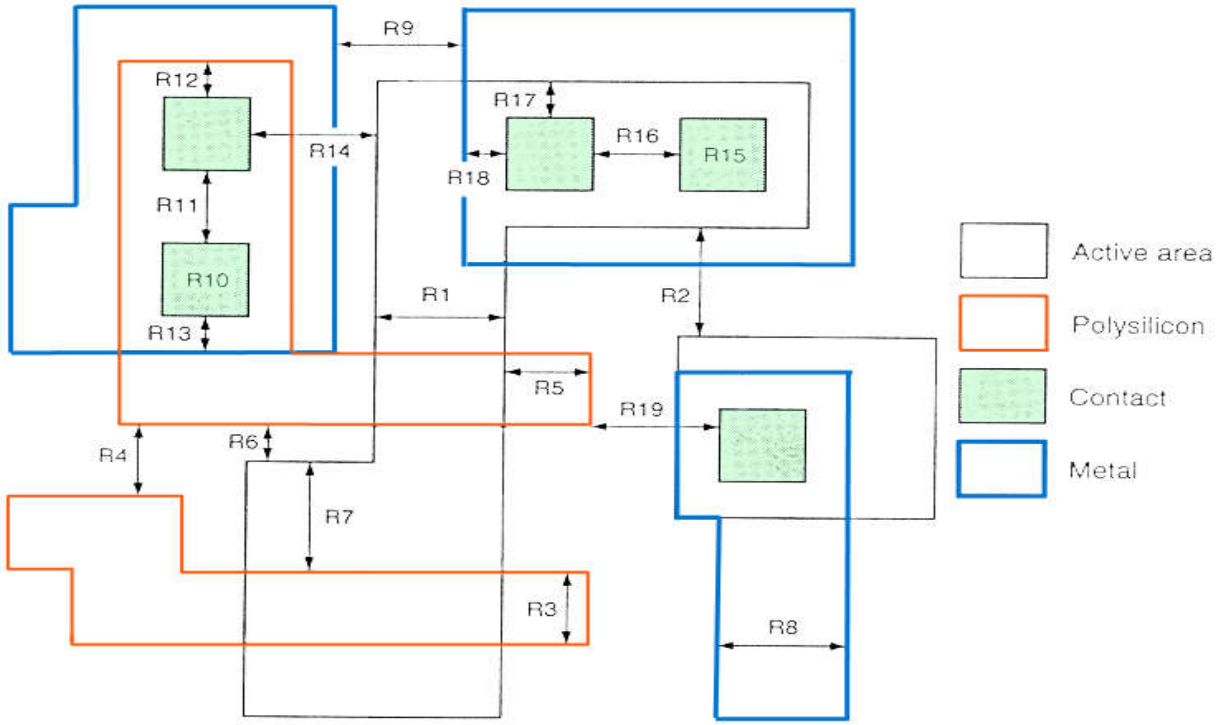

NWELL
VDD
CONT
PDIFF
METAL
A
Y
POLY
NDIFF
VSS
VDD
A
Y
INVX1
INVX8
NWELL
VDD
A
Y
VSS

NAND2 和 NOR2 的版图



版图的设计规则

R1	Minimum active area width	0.22
R2	Minimum active area spacing	0.38
R3	Minimum poly width	0.18
R4	Minimum poly spacing	0.25
R5	Minimum gate extension of poly over active	0.22
R6	Minimum poly-active edge spacing (poly outside active area)	0.10
R7	Minimum poly-active edge spacing (poly inside active area)	0.10
R8	Minimum metal width	0.23
R9	Minimum metal spacing	0.23
R10	Poly contact size	0.22
R11	Minimum poly contact spacing	0.25
R12	Minimum poly contact to poly edge spacing	0.10
R13	Minimum poly contact to metal edge spacing	0.06
R14	Minimum poly contact to active edge spacing	0.10
R15	Active contact size	0.22
R16	Minimum active contact spacing (on the same active region)	0.22
R17	Minimum active contact to active edge spacing	0.10
R18	Minimum active contact to metal edge spacing	0.06
R19	Minimum active contact to poly edge spacing	0.16



CMOS逻辑电路

基本逻辑电路

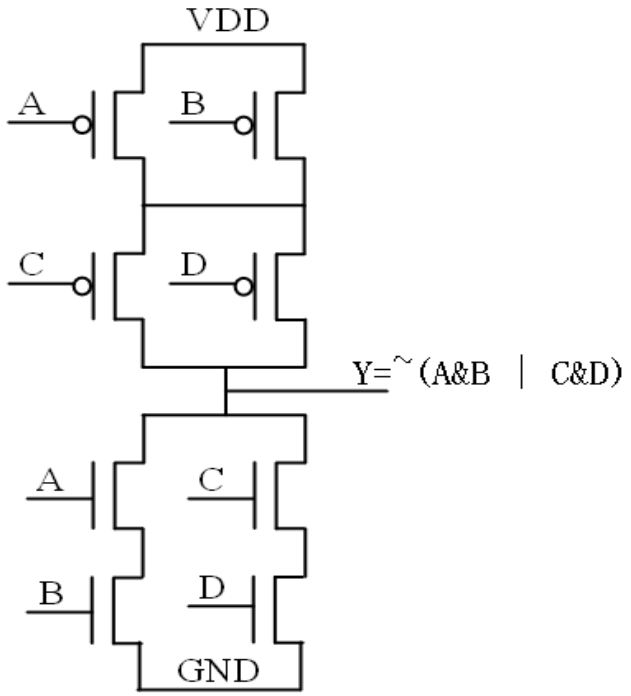
- 组合逻辑电路
  - 电路中没有存储单元，逻辑电路的输出完全由当前的输入决定。
- 时序逻辑电路
  - 电路中有存储单元，逻辑电路的输出由原来状态和当前的输入决定。

逻辑表达式

- **Boolean代数的基本操作：与(*)、或(+)、非(^)**
- 一些基本定律
  - **A+0=A, A*1=A, A+1=1, A*0=0**
  - **A+A=A, A*A=A, A+^A=1, A*^A=0**
  - **A+B=B+A, A*B=B*A**
  - **A+(B+C)=(A+B)+C, A*(B*C)=(A*B)*C**
  - **A*(B+C)=A*B+A*C, A+(B*C)=(A+B)(A+C)**
  - **^(A*B)=^A+^B, ^(A+B)=^A*^B**
  - **A+A*B=A, A+^A*B=A+B**
- **CMOS中，最基本的门是非、与非、或非门，因此逻辑表达式最好写成上述操作的组合**
  - **如，^(^(A*B)*^(C*D))=A*B+C*D**

CMOS组合电路的组成

- 用NMOS组成正逻辑
  - 串联表示与
  - 并联表示并
- 用PMOS组成反逻辑
  - ^(A*B)=^A+^B
  - ^(A+B)=^A*^B
- 正反逻辑串联
  - 由于N网络导通时接地，因此输出是反向的
  - 例： ~(A&B | C&D)



真值表

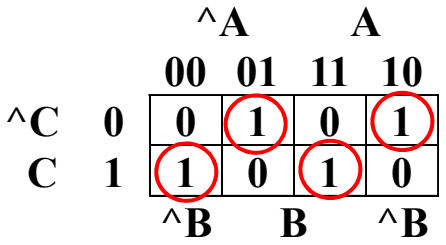
- 对于每一种可能的输入组合，给出输出值。对于一个 **n**输入的电路，有2^n项。
  - 与项之或
- 以一位全加器为例
  - 输入：**A, B, Cin**；输出：**S, Cout**
  - **S=^A*^B*Cin+^A*B*^Cin+A*^B*^Cin+A*B*Cin**
  - **Cout=^A*B*Cin+A*^B*Cin+A*B*^Cin+A*B*Cin**

=A*B+A*Cin+B*Cin

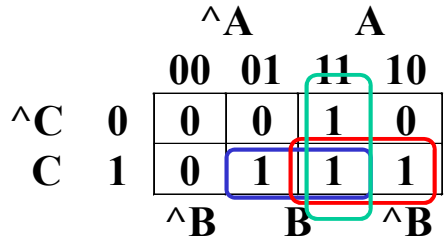
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

卡诺图

- 便于逻辑表达式优化的输入输出关系表示。
- 在图中相临的“1”或“X”合并成一项。
- 以一位全加器为例



S= ^A*^B*Cin+^A*B*^Cin+A*^B*^Cin+A*B*Cin

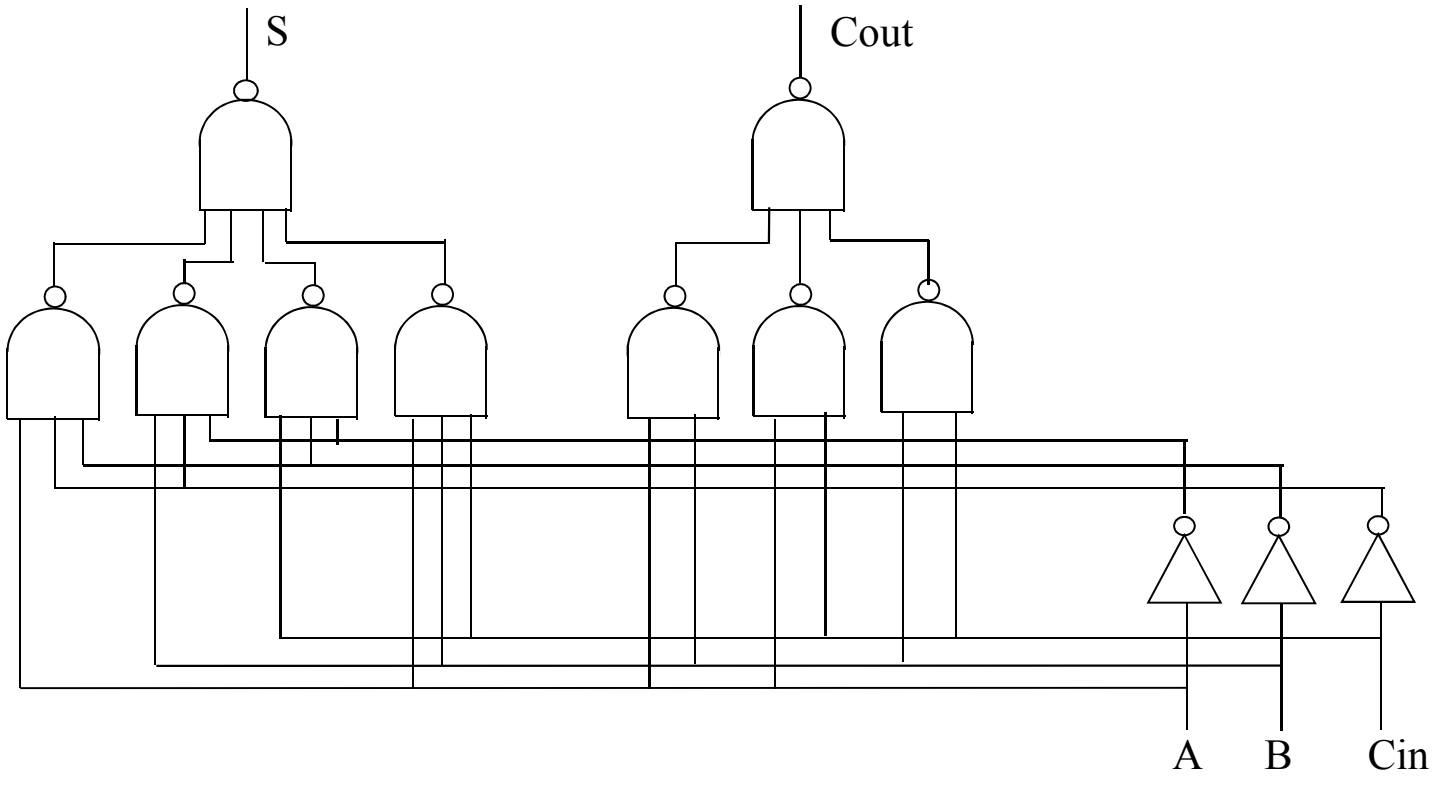


Cout= A*B+A*Cin+B*Cin

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

逻辑图

- 直接用门及互连表示输入输出的逻辑关系
- 以一位全加器为例



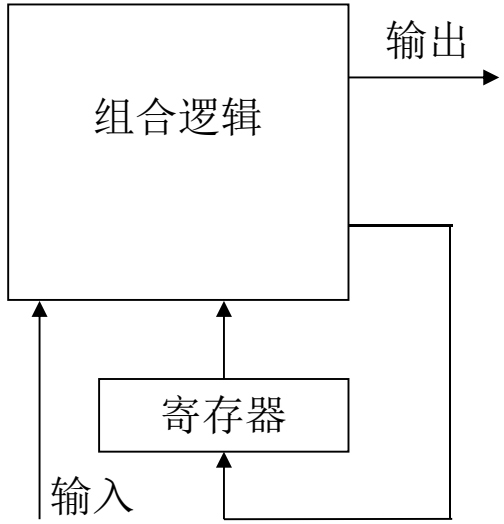
常见组合逻辑：译码器和选择器

输入			输出							
C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

输入			输出
C	B	A	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

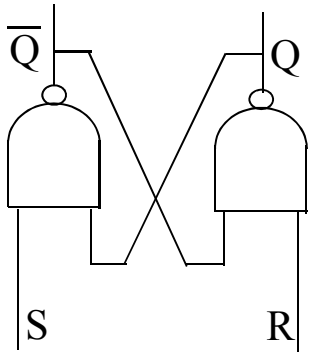
时序逻辑电路

- 时序逻辑电路内部有存储单元，其行为由输入和内部单元的值共同决定
  - 可分为同步时序逻辑电路和异步时序逻辑电路，计算机中主要用同步电路
  - 在同步时序电路中，所有存储单元的变化由时钟统一触发

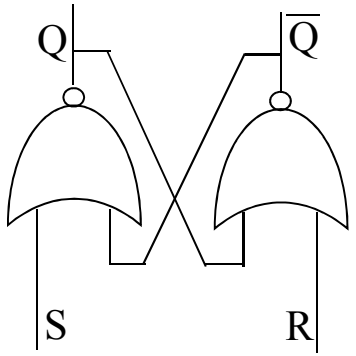


RS触发器

- 其它寄存器电路的基础



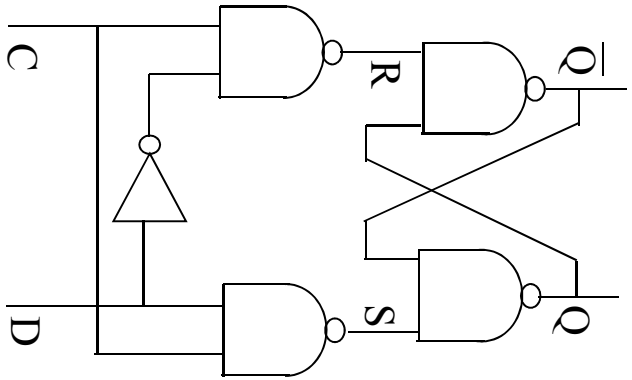
R	S	Q	~Q
0	0	nc	nc
0	1	1	0
1	0	0	1
1	1	Q	~Q

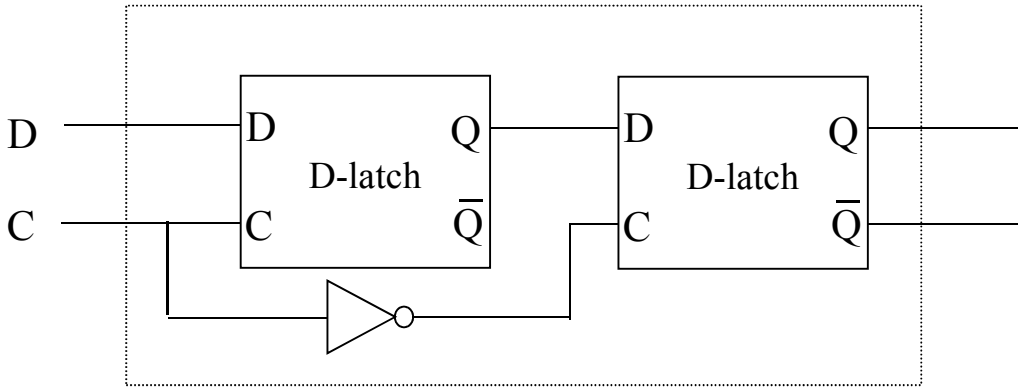


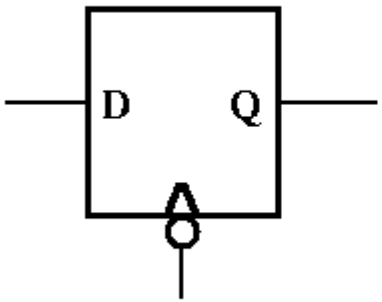
R	S	Q	~Q
0	0	Q	~Q
0	1	1	0
1	0	0	1
1	1	nc	nc

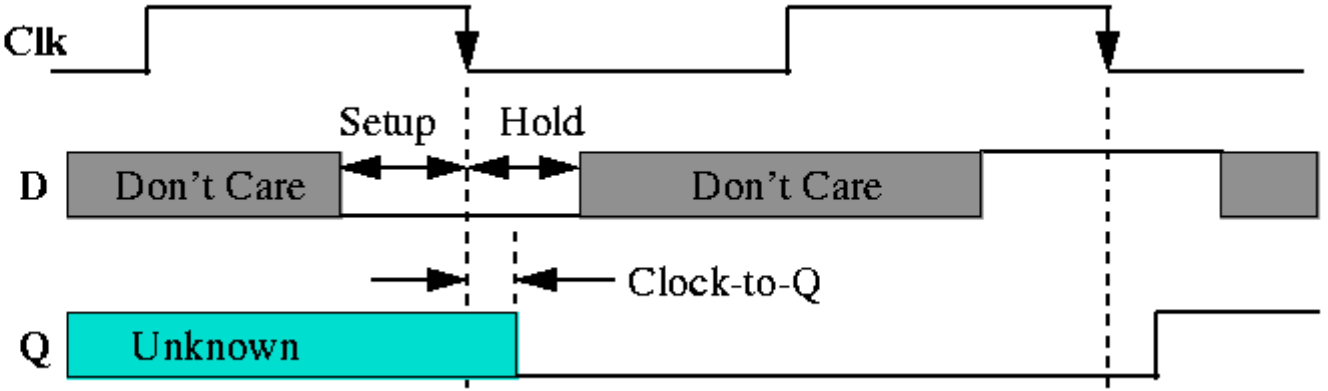
D门闩和D触发器

- **D门闩和D触发器**
  - **三个重要参数：Setup、Hold、CLK-to-Q**
  - **触发器的延迟=Setup+CLK-to-Q**



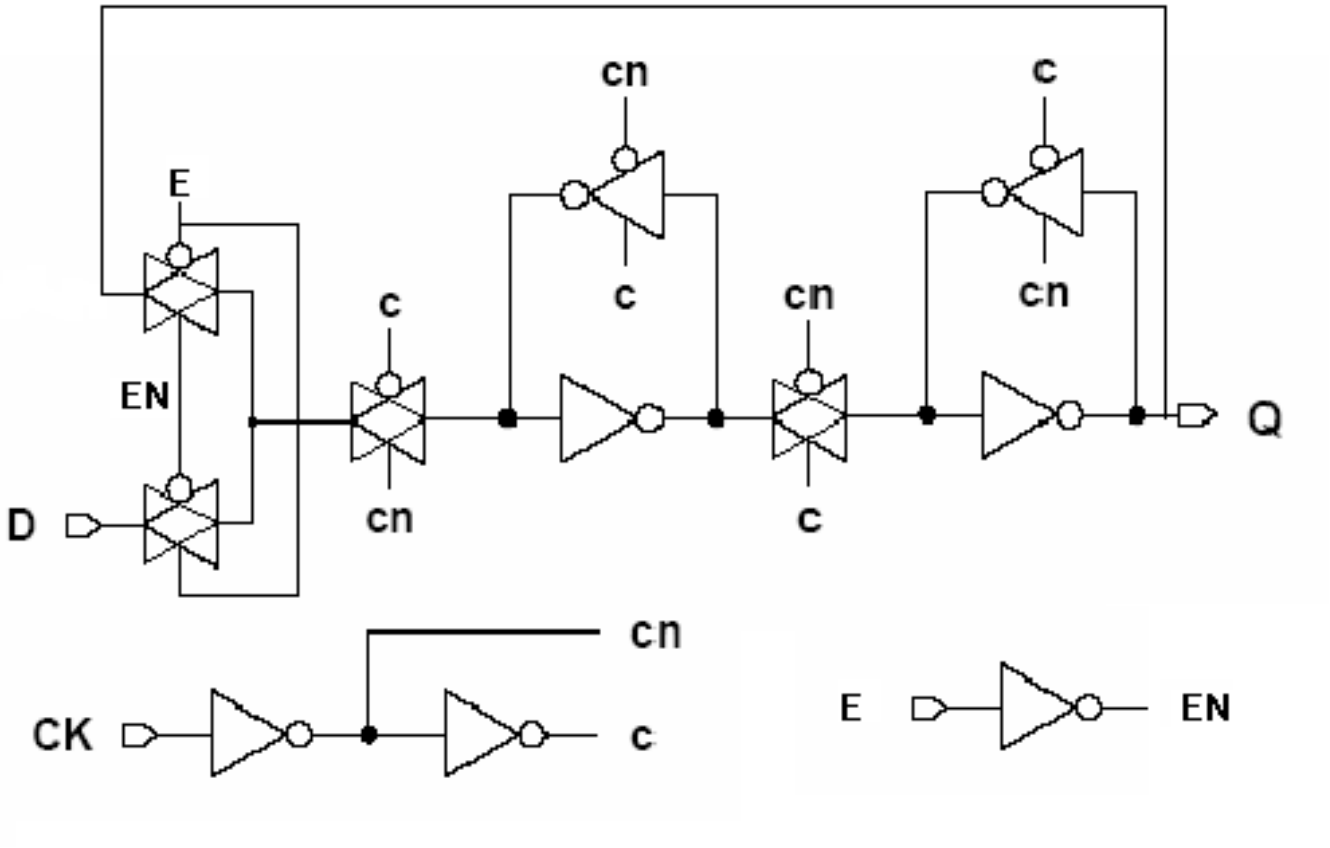


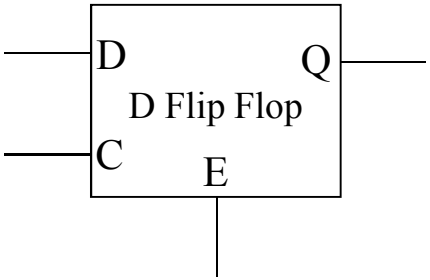




EDFF电路

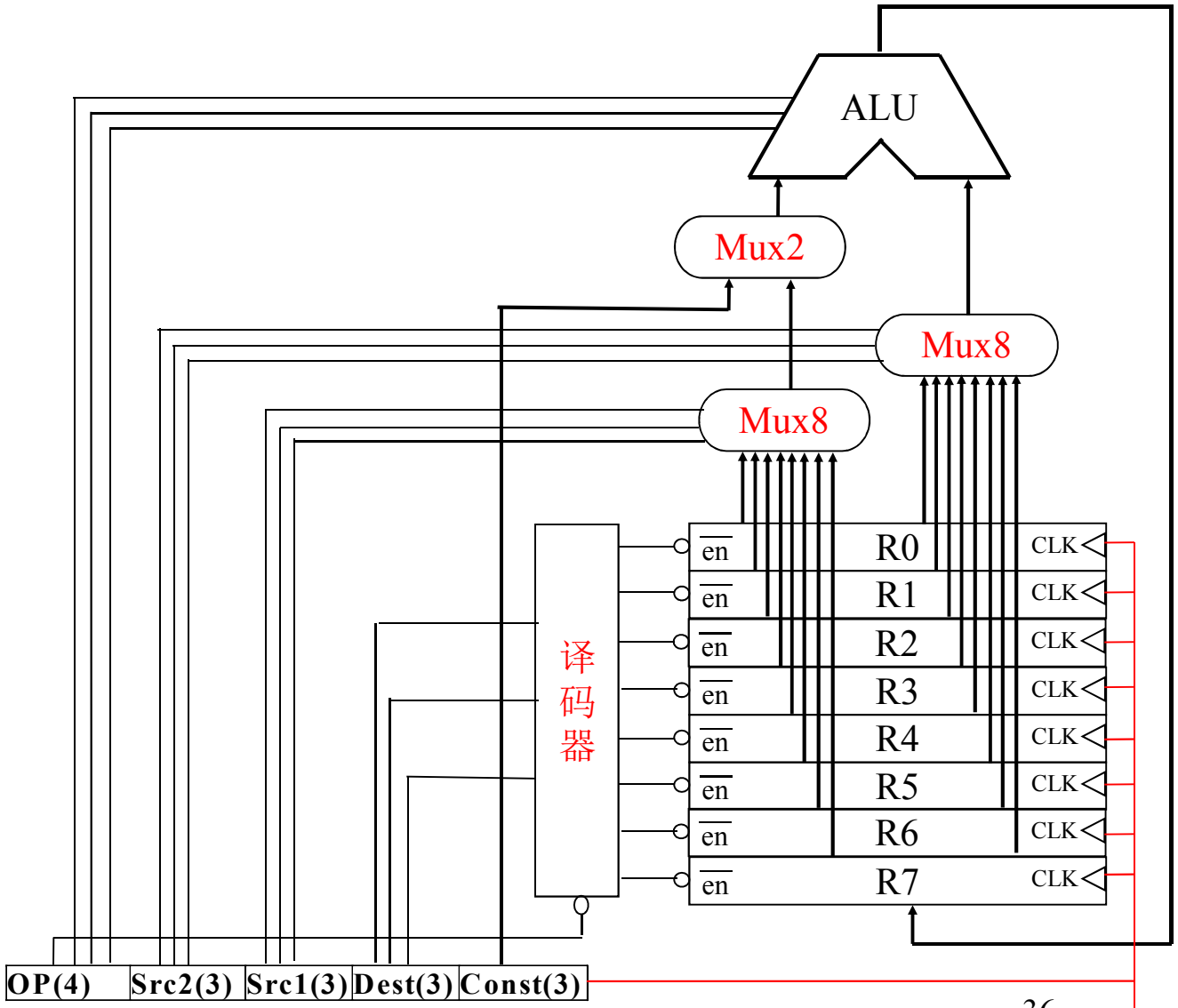
- 带输入使能的D触发器
  - 只有E有效，才能输入





例：简单CPU的运算电路

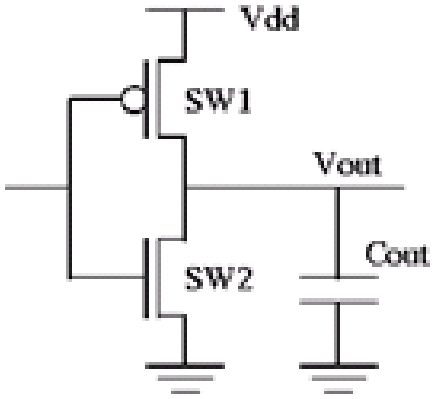
- 16位IR
- 8个16位GPR
- 4位OP
- 三地址
- Load-store结构

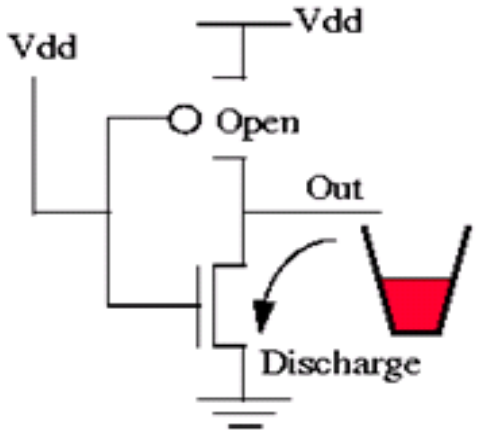


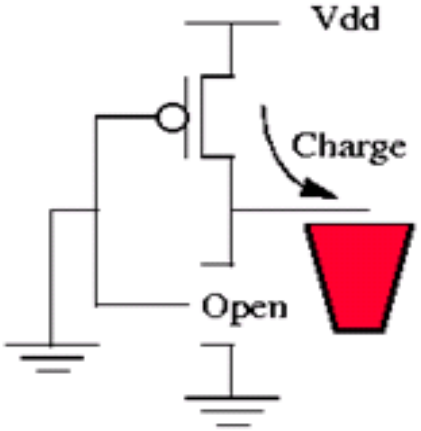
CMOS电路延迟

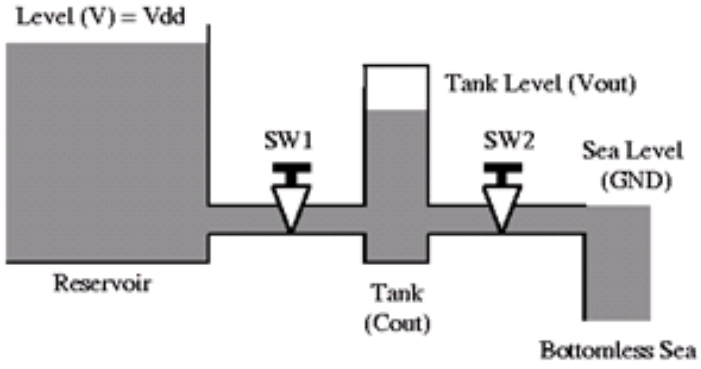
CMOS电路延迟原理

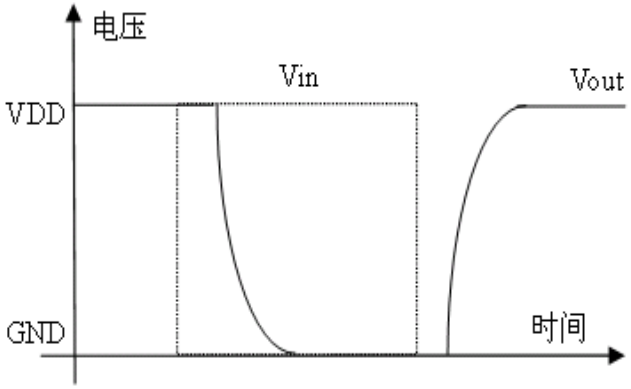
- 输入电压变化时，输出电压的变化不是瞬时完成的











CMOS电路延迟模型

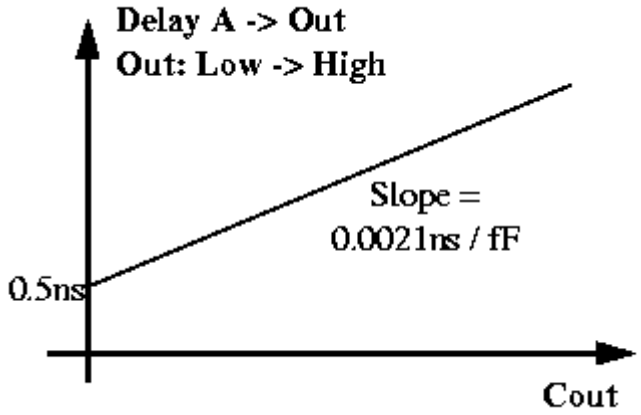
- 与延迟相关的因素
  - 输入**transition**、内部延迟、输出负载有关
  - 负载包括下一级负载、连线延迟等
  - 内部延迟**(ns)**+与负载（包括负载）有关的延迟（**ns/fF**）
- 二输入与非门为例

For A and B: Input Load = 61 fF

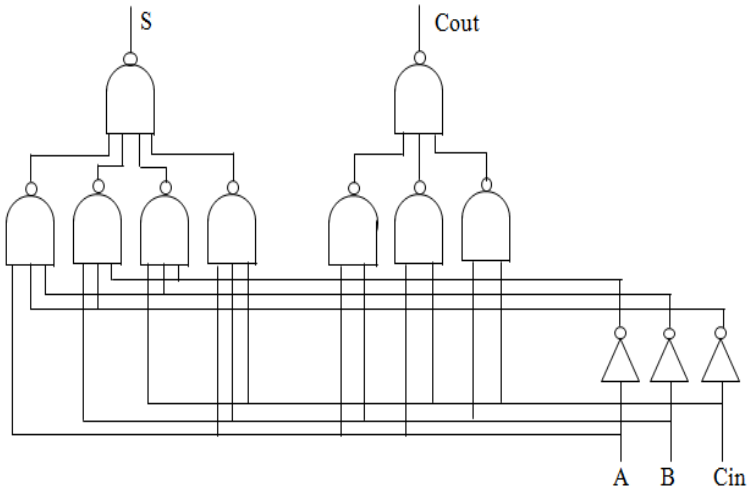
For either A -> Out or B -> Out:

TPlh = 0.5ns Tplhf = 0.0021ns / fF

TPhl = 0.1ns TPhlf = 0.0020ns / fF



CMOS电路延迟例子



TPhl(INV)	= 0.10ns	TPlh(INV)	= 0.10ns
TPhl(NAND2)	= 0.10ns	TPlh(NAND2)	= 0.20ns
TPhl(NAND3)	= 0.15ns	TPlh(NAND3)	= 0.20ns
Tlhf(INV)	= 0.010ns/fF	Thlf(INV)	= 0.010ns/fF
Tlhf(NAND2)	= 0.010ns/fF	Thlf(NAND2)	= 0.015ns/fF
Tlhf(NAND3)	= 0.010ns/fF	Thlf(NAND3)	= 0.020ns/fF
Cil(INV)	= 20fF	Cil(NAND2/3)	= 30fF
Cwire	= 50fF		

- 假设Cout负载为60fF.

Tlh(Cin->Cout) = TPhl(NAND2) + Thlf(NAND2)*(Cil(NAND3)+Cwire) +
TPlh(NAND3) + Tlhf(NAND3)*Col
= 0.10 + 0.015*(30+50) + 0.10 + 0.010 * 60
= 2.0ns

标准单元延迟的计算

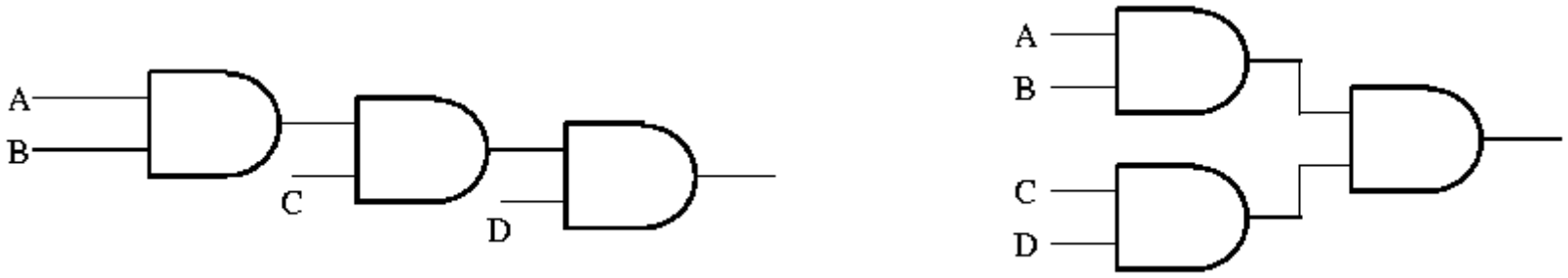
- 深亚微米电路中连线延迟占总延迟的大部分
  - 以**TSMC18**的**NAND2X1**为例：内部延迟**0.034ns**，每**pf**延迟为**4.5ns**左右，输入电容**0.004pf**
- 连线延迟的计算是一个不断迭代求精的过程，
  - 综合时根据负载个数估计线长，非常初略
  - 布局后根据距离估计线长，比较准确了
  - 布线后进一步考虑具体连线、互相干扰、过孔等精确连线信息
- **TSMC 0.18**微米二输入与门的特征
  - 面积、输入电容、最大负载、功耗
  - 延迟（非线性）：根据输入**transition**和输出负载查表
  - 库单元的定义

降低延迟的方法

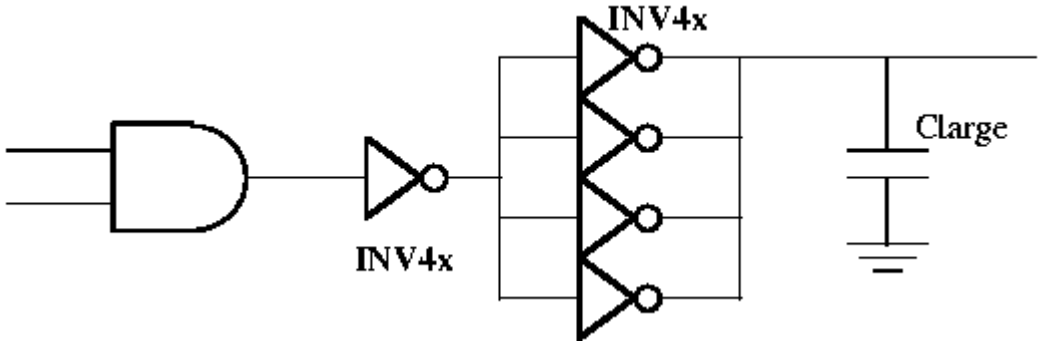
- 通过结构设计降低延迟
  - 如流水级的划分、**CACHE**读出和命中比较是否在同一级、浮点流水级设计、**Pipeline stalling signal** 的设置
- 逻辑设计
  - 如多选一通路的选择次序、加法器算法、流水级间的局部调整等
- 物理设计
  - 动态电路、**clock skew**、**clock jitter**、**transition time**、布局布线

通过逻辑设计降低延迟

- 减少门的级数



- 负载平衡：**CMOS**的负载能力较低，分推



从Verilog到版图

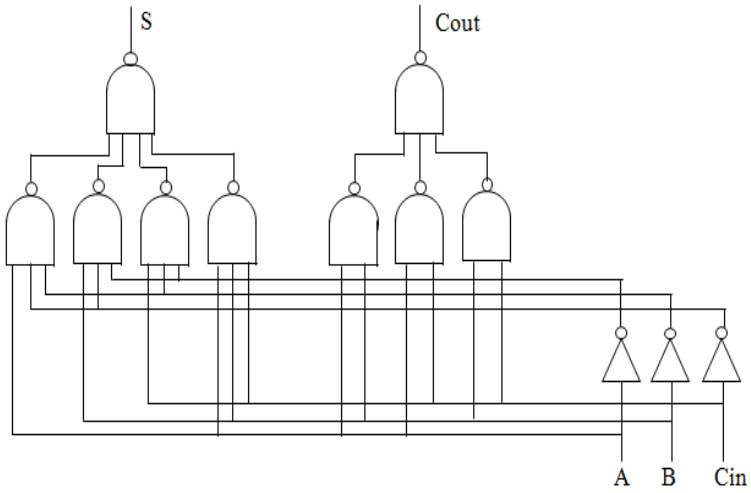
Verilog语言

- 做结构设计时要明白逻辑上是什么样子
- 做逻辑设计时要明白物理上是什么样子
- 硬件设计人员和软件设计人员对**Verilog**有不同认识
- 我只用**Verilog**中三种语句
  - 组合逻辑：**assign**
  - 时序逻辑：**always (@posedge clock)**
  - 模块调用

Verilog 语言---组合逻辑

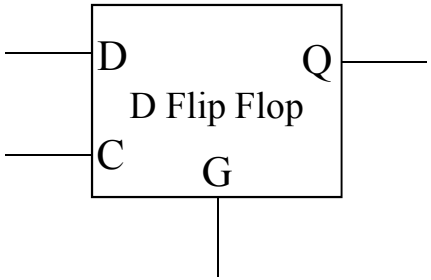
- 组合逻辑---assign语句
  - 以一位全加器为例

```
module full_adder(a,b,c,s,cout);
  input a,b,c;
  output s,out;

  assign s=a^b^c;
  assign cout=a&b | b&c | c&a
endmodule
```



Verilog语言----时序逻辑

- 时序逻辑---触发器
  - 以一位触发器为例




```
module dflipflop(D,C,G,Q);
input D,C,G;
output Q;
reg Q;
always (@posedge C)begin
  if (G) Q<=D;
end
endmodule
```

Verilog语言----模块调用

- 以四位串行加法器为例

```
module adder4(a,b,cin,s,cout);
input [3:0] a;
input [3:0] b;
input cin;
output [3:0] s;
output cout;
wire carry[2:0]
full_adder  bit0(.a(a[0]),.b(b[0]),.c(cin[0]),.s(s[0]),.cout(carry[0]));
full_adder  bit1(.a(a[1]),.b(b[1]),.c(carry[0]),.s(s[1]),.cout(carry[1]));
full_adder  bit2(.a(a[2]),.b(b[2]),.c(carry[1]),.s(s[2]),.cout(carry[2]));
full_adder  bit3(.a(a[3]),.b(b[3]),.c(carry[2]),.s(s[3]),.cout(cout));
endmodule
```

Verilog语言

- 四位计数器

```
module counter4(reset,en,clock,out);
input reset,en,clock;
output [3:0] out;
reg [3:0] counter;
wire [3:0] c_in;
assign out = counter;
adder4 counter(.a(counter),.b(4'b0),.cin(1'b1),.s(c_in),.cout());
always (@posedge clock) begin
counter <= reset ? 0 : en ? c_in : counter;
//counter<=reset ? 0 : en ? counter +1 : counter;
end
endmodule
```

从Verilog 到GDSII

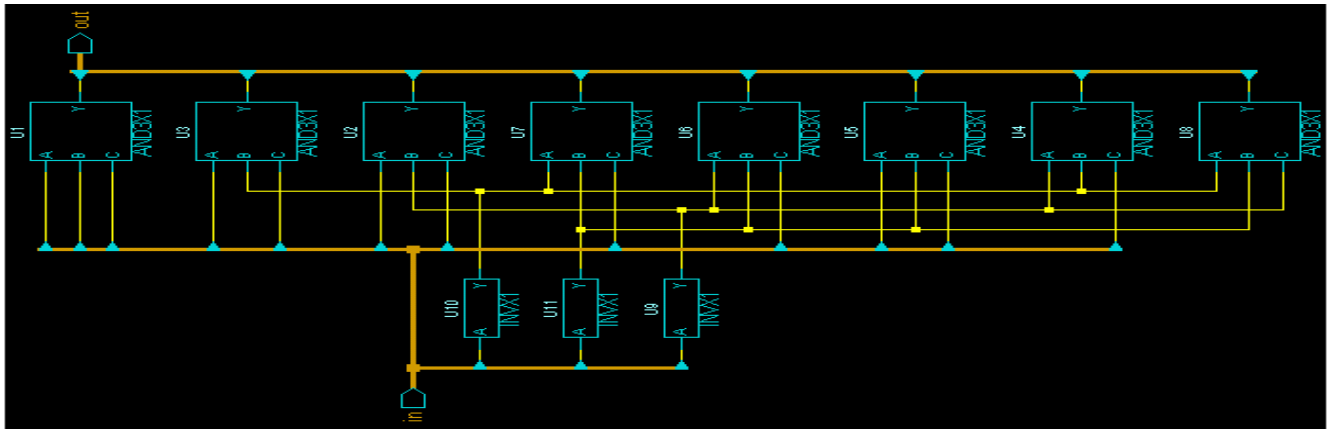
- 以3-8译码器为例
  - **Verilog**
  - 网表
  - 布线
  - **GDSII**

Verilog

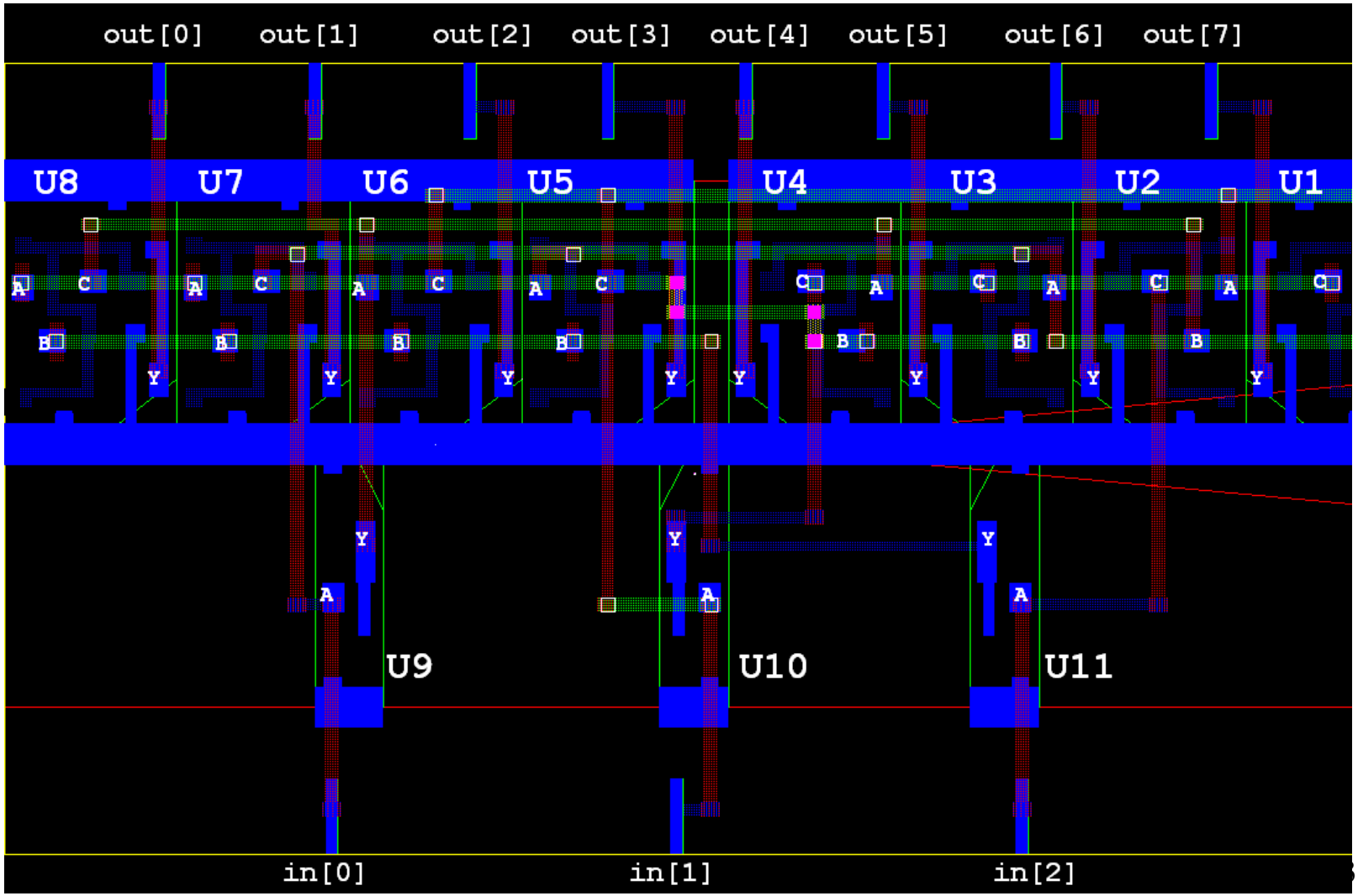
```
module decode(in, out);
input [2:0] in;
output [7:0] out;

assign out[0] = (!in[2]) && (!in [1]) && (!in[0]); // (in==3'b000)?1:0;
assign out[1] = (!in[2]) && (!in [1]) && ( in[0]); // (in==3'b001)?1:0;
assign out[2] = (!in[2]) && ( in [1]) && (!in[0]); // (in==3'b010)?1:0;
assign out[3] = (!in[2]) && ( in [1]) && ( in[0]); // (in==3'b011)?1:0;
assign out[4] = ( in[2]) && (!in [1]) && (!in[0]); // (in==3'b100)?1:0;
assign out[5] = ( in[2]) && (!in [1]) && ( in[0]); // (in==3'b101)?1:0;
assign out[6] = ( in[2]) && ( in [1]) && (!in[0]); // (in==3'b110)?1:0;
assign out[7] = ( in[2]) && ( in [1]) && ( in[0]); // (in==3'b111)?1:0;

endmodule
```

网 表

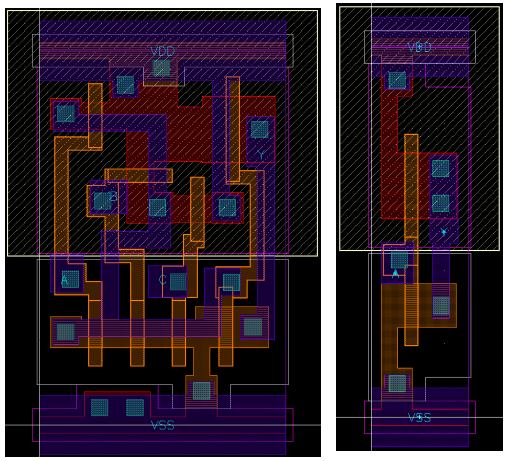
```
module decode ( in, out );
input  [2:0] in;
output [7:0] out;
    wire n1, n2, n3;
    AND3X1 U1 ( .A(in[1]), .B(in[0]), .C(in[2]), .Y(out[7]) );
    AND3X1 U2 ( .A(in[1]), .B(n1),    .C(in[2]), .Y(out[6]) );
    AND3X1 U3 ( .A(in[0]), .B(n2),    .C(in[2]), .Y(out[5]) );
    AND3X1 U4 ( .A(n1),    .B(n2),    .C(in[2]), .Y(out[4]) );
    AND3X1 U5 ( .A(in[0]), .B(n3),    .C(in[1]), .Y(out[3]) );
    AND3X1 U6 ( .A(n1),    .B(n3),    .C(in[1]), .Y(out[2]) );
    AND3X1 U7 ( .A(n2),    .B(n3),    .C(in[0]), .Y(out[1]) );
    AND3X1 U8 ( .A(n2),    .B(n3),    .C(n1),    .Y(out[0]) );
    INVX1  U9 ( .A(in[0]), .Y(n1) );
    INVX1 U10 ( .A(in[1]), .Y(n2) );
    INVX1 U11 ( .A(in[2]), .Y(n3) );
endmodule
```

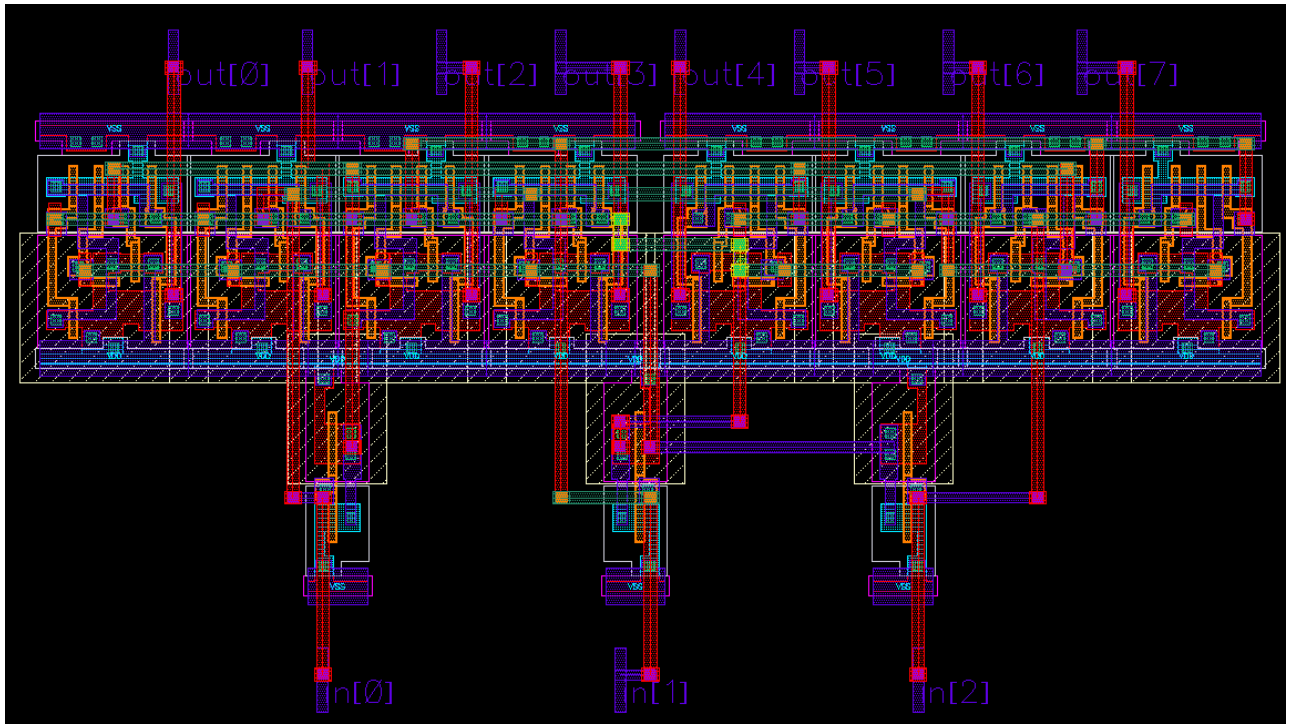


布线后


out[0]
out[1]
out[2]
out[3]
out[4]
out[5]
out[6]
out[7]
U8
U7
U6
U5
U4
U3
U2
U1
U9
U10
U11
in[0]
in[1]
in[2]

版图





其它“0”和“1”的表示方法

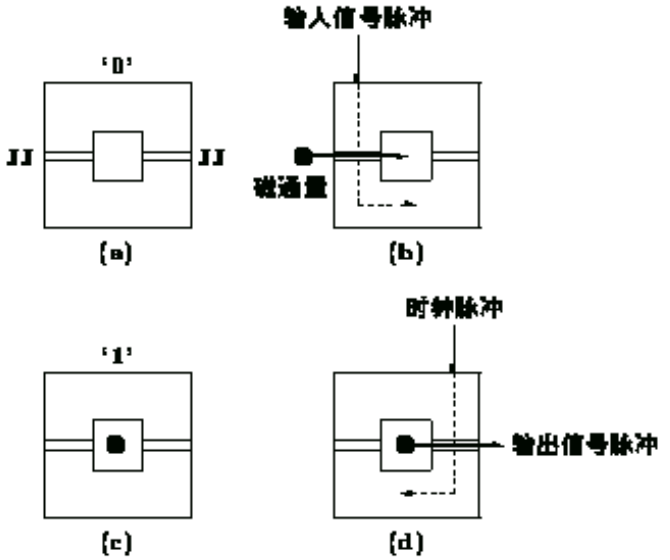
其它“0”和“1”表示方法

- 用磁通量的有无表示，超导体工艺
- 用能级的高低表示，量子计算机
- 用基因序列表示，**A, G, C, T, DNA**计算机，非二进制？

超导计算---RSFQ技术

(Rapid Single Flux Quantum)

- 基本原理
  - 超导(4-5K)环中的磁通量具有量子化特性，设计电路使超导环中的磁通量只变化一个磁通量，用磁通量的有无来表示二进制数位的“1”和“0”。磁通量的变化由外加电流控制。
  - **SQUID (Superconducting Quantum Interface Device)**
- 电路特点
  - 工作频率高--**100 GHz**（实验室已达**370 GHz, 1.5μ** 工艺）。
  - 每个门的功率——**0.1μW**。
  - 工艺比较简单。



量子计算（1）

- 量子力学特性
  - 量子叠加态(superposed state)：量子器件的信息位称为量子位(qubit)，它可处于叠加态。叠加态可以是 “0” 也可以是 “1”。通过测量或与其他物体发生相互作用可呈现出 “0” 态或 “1” 态。由于每个量子位都可以是 “0” 或 “1”， n 个量子位就可以表示 2^n 个 n 位数。常规计算机的一个 n 位存储单元只能存放一个 n 位数，而 n 个量子位可以存放 2^n 个 n 位数，可以实现超大容量的存储器。
  - 量子纠缠态(entangled state)——除了叠加态以外，用作运算的多个量子位还应处于纠缠态，即所有量子位的状态紧密相关。当测量某个量子位时，会影响其他量子位的测量结果。
  - 量子并行 (quantum parallelism) ——计算 f(x) 时，可同时计算出 x 的所有值的 f(x)。所以不需要多次循环，也不需要多个处理机并行计算。

量子计算（2）

- 应用领域
  - 大数 N 因子分解——令 $n = \log_2 N$, 经典算法所需步骤为 $2^{n/2}$ ， Shor 量子并行算法所需步骤为 Poly(n)，Poly(n) 为 n 的多项式。该算法将 NP 问题转换为 P 问题。
  - 搜寻算法—— 在 N 个元素的集合中搜寻某个元素，经典算法搜寻 N/2 次后，找到的概率为 1/2， Grover 量子搜寻算法则只需 $N^{1/2}$ 次，即可达到同样概率。
  - 量子系统模拟——常规计算机不可能有效地模拟量子系统，因为它们的物理机制不同。用常规计算机模拟量子系统，所需的信息量和时间都远大于模拟经典系统。量子计算可用于研究高温高密度等离子体、量子色动力学、晶体固态模型、分子行为的量子模型等。

量子计算（3）

- 物理实现
  - 量子位的实现——任何两态的量子系统都可作为量子位，如原子的能级、电子或原子核的自旋、光子的正交偏振态等。
  - 量子计算机的实现——有多种可能，包括：核磁共振（用磁场中的原子核自旋作为量子位）、离子阱（用被俘获在线性量子阱中的离子作为量子位）、硅基半导体量子器件（杂质核自旋与电子自旋相互作用）等。
- 存在问题
  - 量子位的缠结态容易崩溃，位数越多，越难实现。
  - 量子器件之间的连接。
  - 为了维持量子逻辑的一致性，量子系统和环境的隔离。
  - 设备缺陷所引起的逻辑错误。

分子计算机

- 原理
  - DNA 计算机利用 DNA 分子保存信息。DNA 分子是由 A, G, C, T 四种核苷酸（碱基）组成的序列。不同的序列可用来表示不同的信息。通过 DNA 分子之间的一系列生化反应来进行运算，可产生表示结果的 DNA 分子。已解决了 7 个城市的旅行售货员等问题。
- 优点
  - 高度并行——所有 **DNA** 分子同时运算。
  - 能耗低——半导体计算机的 10^{10} 之一。
  - 存储密度大——磁存储器的 10^{12} 倍。
- 缺点
  - 生化反应慢、操作有随机性、**DNA** 分子容易水解、**DNA** 分子之间难以通信。

作 业