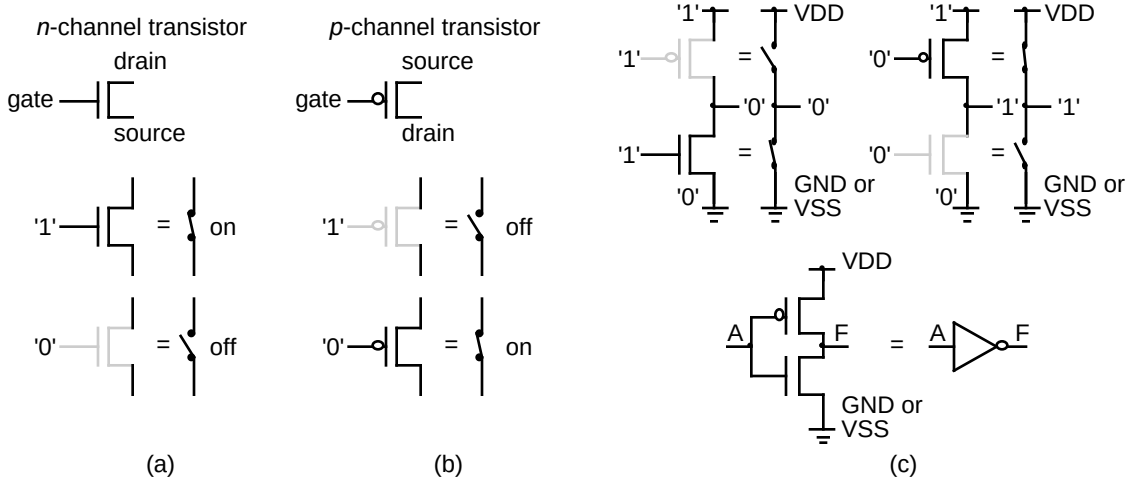
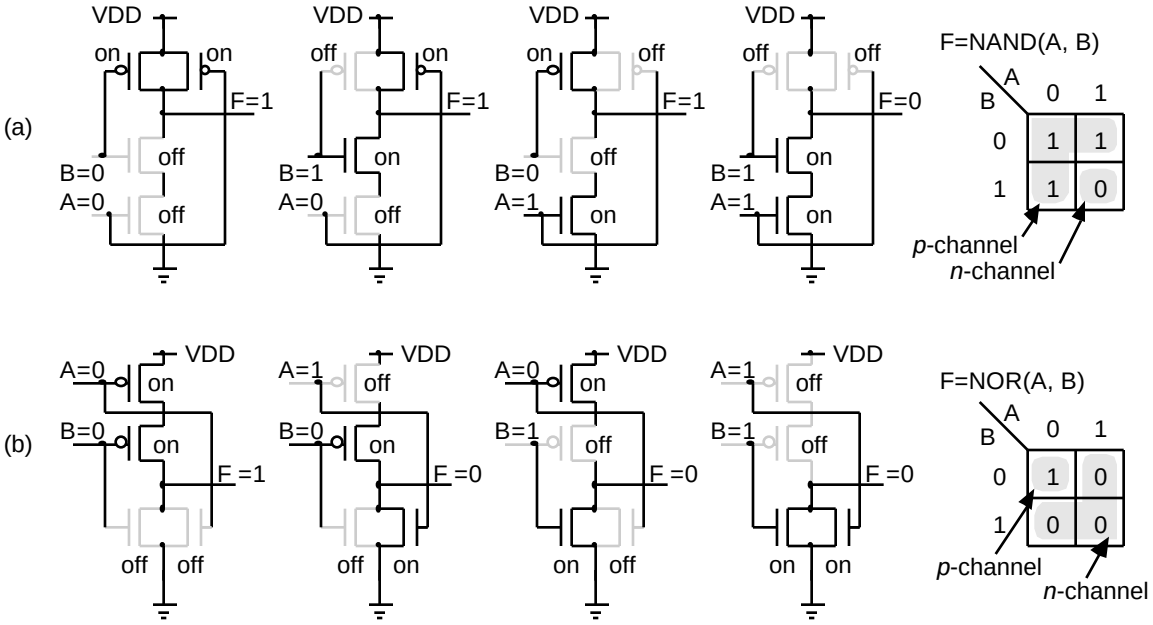
CMOS LOGIC 2

Key concepts: The use of transistors as switches • The difference between a flip-flop and a latch • Setup time and hold time • Pipelines and latency • The difference between datapath, standard-cell, and gate-array logic cells • Strong and weak logic levels • Pushing bubbles • Ratio of logic • Resistance per square of layers and their relative values in CMOS • Design rules and λ

- **CMOS transistor** (or device)
- A transistor has three terminals: **gate**, **source**, **drain** (and a fourth that we ignore for a moment)
- An MOS transistor looks like a switch (conducting/on, nonconducting/off, not open or closed)

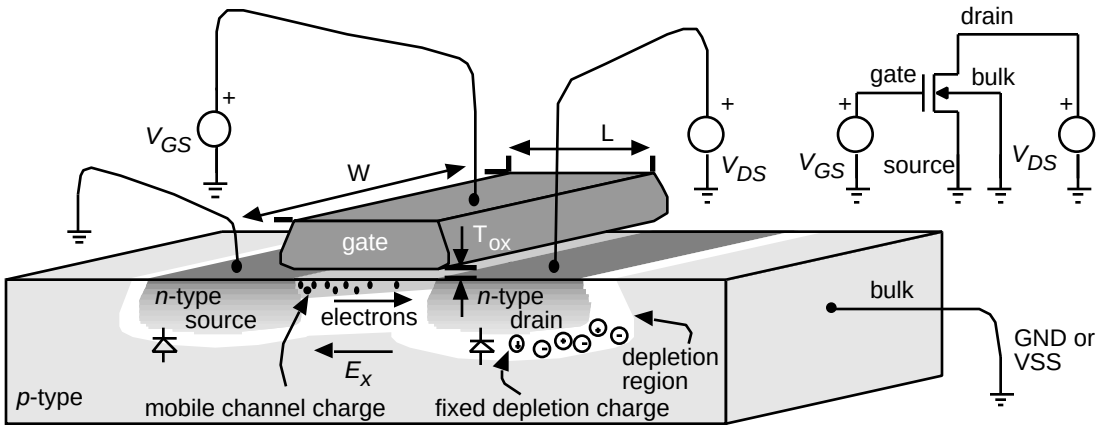


CMOS transistors viewed as switches • a CMOS inverter



CMOS logic • a two-input **NAND gate** • a two-input **NOR gate** • **Good '1's** • **Good '0's**

2.1 CMOS Transistors



An n-channel transistor • **channel** • **source** • **drain** • **depletion region** • **gate** • **bulk**

current (amperes) = charge (coulombs) per unit time (second)

- Channel charge = Q (imagine taking a picture and counting the electrons)
- t_f is **time of flight** or **transit time**

The drain-to-source current $I_{DSn} = Q/t_f$

The (vector) velocity of the electrons $\mathbf{v} = -\mu_n \mathbf{E}$

- μ_n is the **electron mobility** (μ_p is the **hole mobility**)
- **E** is the electric field (units Vm^{-1})

$$t_f = \frac{L}{v_x} = \frac{L^2}{\mu_n V_{DS}}$$

$$Q = C(V_{GC} - V_{tn}) = C\,[\,(V_{GS} - V_{tn}) - 0.5\,V_{DS}\,] = WLC_{ox}\,[\,(V_{GS} - V_{tn}) - 0.5\,V_{DS}\,]$$

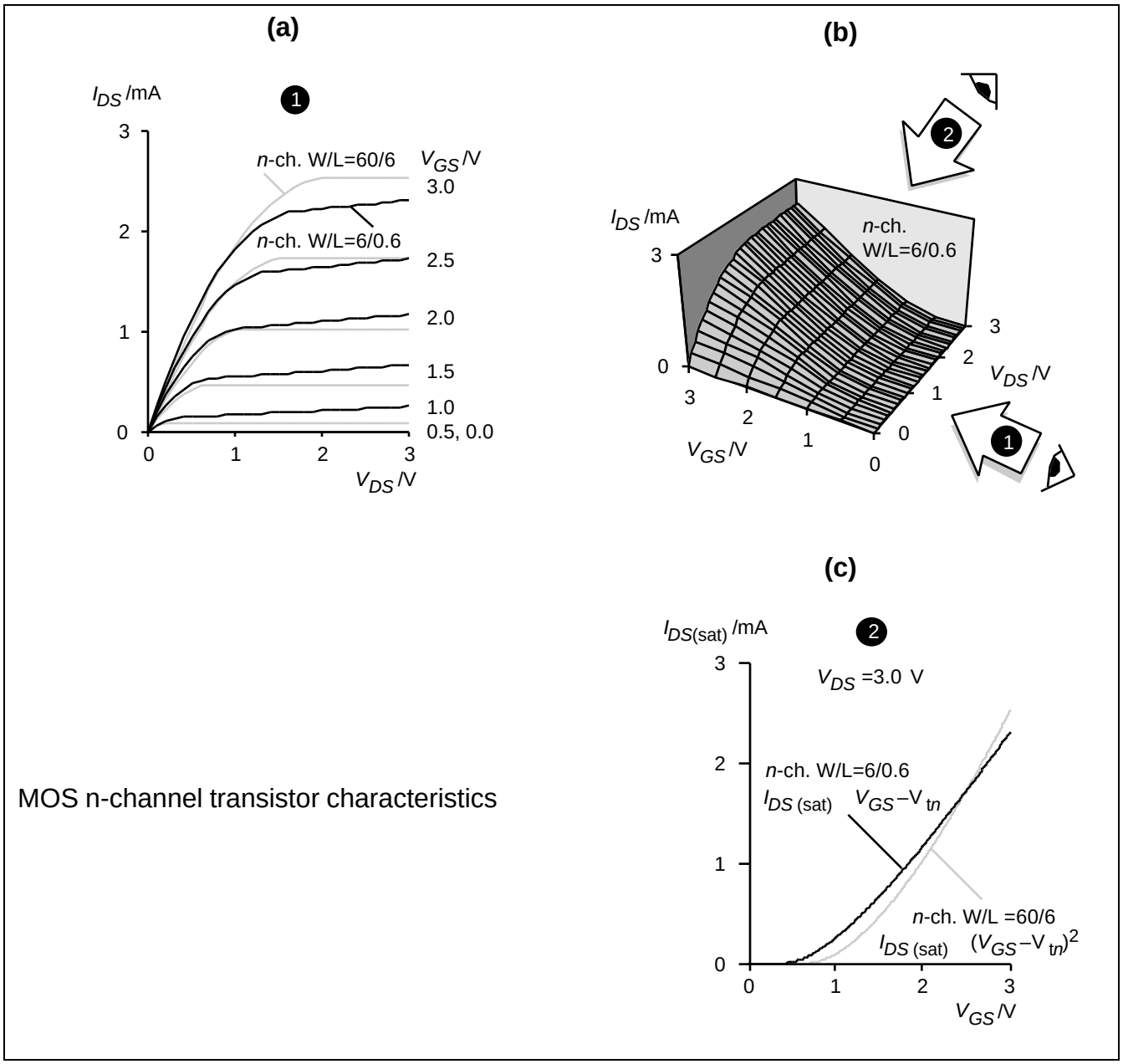
$$I_{DSn} = Q/t_f$$
$$= (W/L)\mu_n C_{ox}[\,(V_{GS} - V_{tn}) - 0.5\,V_{DS}\,]V_{DS} = (W/L)k'_n\,[\,(V_{GS} - V_{tn}) - 0.5\,V_{DS}\,]V_{DS}$$

$k'_n = \mu_n C_{ox}$ is the process transconductance parameter (or **intrinsic transconductance**)

$\beta_n = k'_n(W/L)$ is the **transistor gain factor** (or just **gain factor**)

- The **linear region** (triode region) extends until $V_{DS} = V_{GS} - V_{tn}$
- $V_{DS} = V_{GS} - V_{tn} = V_{DS(sat)}$ (**saturation voltage**)
- $V_{DS} > V_{GS} - V_{tn}$ (the **saturation region**, or pentode region, of operation)
- **saturation current**, $I_{DSn(sat)}$

$$I_{DSn(sat)} = (\beta_n/2)(V_{GS} - V_{tn})^2 \;;\quad V_{GS} > V_{tn}$$



MOS n-channel transistor characteristics

2.1.1 P-Channel Transistors

$$I_{DSp} = -k'_p(W/L)[\,(V_{GS} - V_{tp}) - 0.5\,V_{DS}\,]V_{DS}\;;\quad V_{DS} > V_{GS} - V_{tp}$$

$$I_{DSp(sat)} = -\beta_p/2\,(V_{GS} - V_{tp})^2\;;\quad V_{DS} < V_{GS} - V_{tp}\;.$$

- V_{tp} is negative
- V_{DS} and V_{GS} are normally negative (and –3V<–2V)

2.1.2 Velocity Saturation

- $v_{maxn}=10^5 ms^{-1}$
- **velocity saturation**
- $t_f = L_{eff}/v_{maxn}$
- **mobility degradation**

$I_{DSn(sat)} = Wv_{maxn}C_{ox}(V_{GS} - V_{tn})$; $V_{DS} > V_{DS(sat)}$ (velocity saturated).

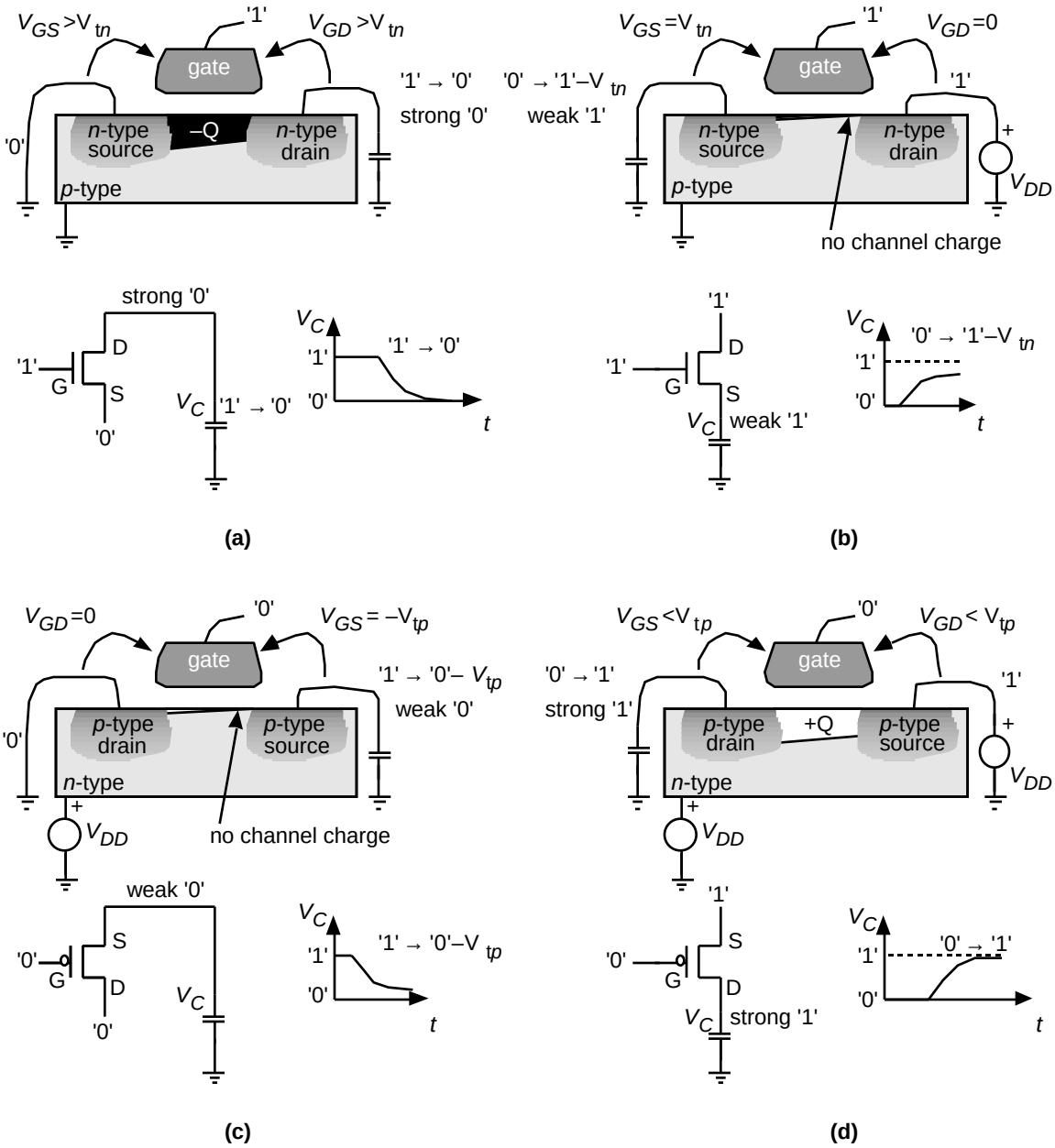
2.1.3 SPICE Models

- KP (in μAV^{-2}) = k'_n (k'_p)
- VT0 and TOX = V_{tn} (V_{tp}) and T_{ox}
- U0 (in $cm^2V^{-1}s^{-1}$) = μ_n (and μ_p)

SPICE parameters

```
.MODEL CMOSN NMOS LEVEL=3 PHI=0.7 TOX=10E-09 XJ=0.2U TPG=1 VTO=0.65
DELTA=0.7
+ LD=5E-08 KP=2E-04 UO=550 THETA=0.27 RSH=2 GAMMA=0.6 NSUB=1.4E+17
NFS=6E+11
+ VMAX=2E+05 ETA=3.7E-02 KAPPA=2.9E-02 CGDO=3.0E-10 CGSO=3.0E-10
CGBO=4.0E-10
+ CJ=5.6E-04 MJ=0.56 CJSW=5E-11 MJSW=0.52 PB=1
.MODEL CMOSP PMOS LEVEL=3 PHI=0.7 TOX=10E-09 XJ=0.2U TPG=-1 VTO=-
0.92 DELTA=0.29
+ LD=3.5E-08 KP=4.9E-05 UO=135 THETA=0.18 RSH=2 GAMMA=0.47
NSUB=8.5E+16 NFS=6.5E+11
+ VMAX=2.5E+05 ETA=2.45E-02 KAPPA=7.96 CGDO=2.4E-10 CGSO=2.4E-10
CGBO=3.8E-10
+ CJ=9.3E-04 MJ=0.47 CJSW=2.9E-10 MJSW=0.505 PB=1
```

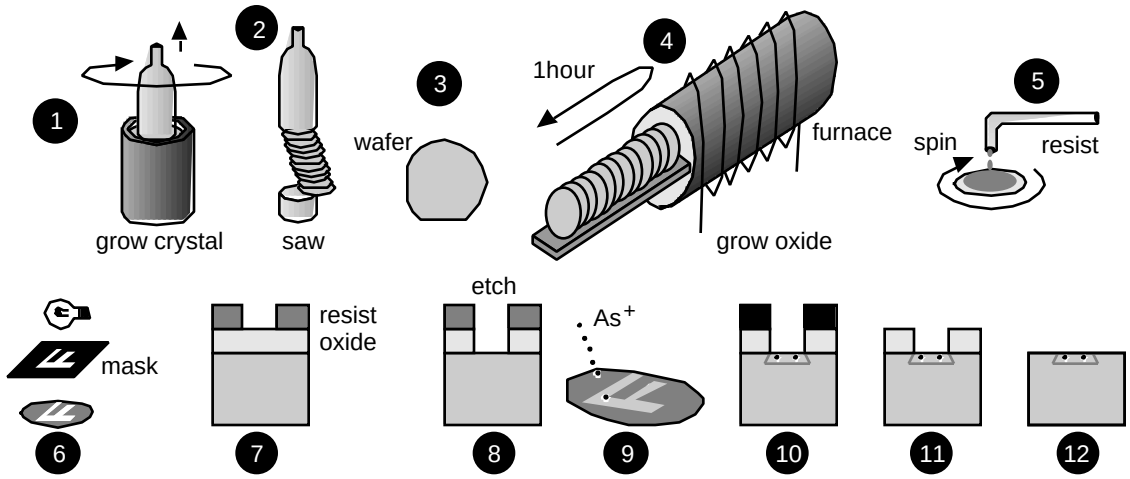
2.1.4 Logic Levels



CMOS logic levels

- V_{SS} is a strong '0' • V_{DD} is a strong '1'
- degraded logic levels: $V_{DD}-V_{tn}$ is a weak '1' ; $V_{SS}-V_{tp}$ (V_{tp} is negative) is a weak '0'

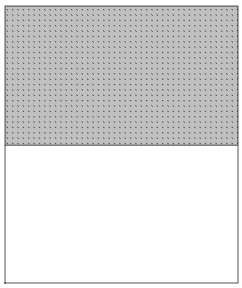
2.2 The CMOS Process



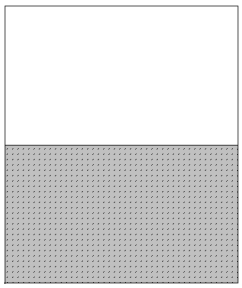
The CMOS manufacturing process

Key words: boule • wafer • boat • silicon dioxide • resist • mask • chemical etch • isotropic • plasma etch • anisotropic • ion implantation • implant energy and dose • **polysilicon** • chemical vapor deposition (CVD) • sputtering • photolithography • submicron and deep-submicron process • n-well process • p-well process • twin-tub (or twin-well) • triple-well • **substrate contacts** (well contacts or tub ties) • **active** (**CAA**) • **gate oxide** • **field** • field implant or channel-stop implant • **field oxide** (**FOX**) • bloat • dopant • **self-aligned process** • positive resist • negative resist • drain engineering • LDD process • lightly doped drain • LDD diffusion or LDD implant • stipple-pattern

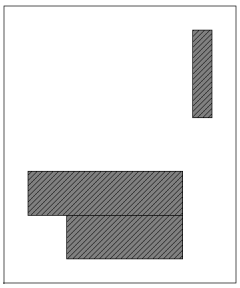
Mask/layer name	Derivation from drawn layers	Alternative names for mask/layer	Mask label
n-well	=nwell	bulk, substrate, tub, n-tub, moat	CWN
p-well	=pwell	bulk, substrate, tub, p-tub, moat	CWP
active	=pdiff+ndiff	thin oxide, thinox, island, gate oxide	CAA
polysilicon	=poly	poly, gate	CPG
n-diffusion implant	=grow(ndiff)	ndiff, n-select, nplus, n+	CSN
p-diffusion implant	=grow(pdiff)	pdiff, p-select, pplus, p+	CSP
contact	=contact	contact cut, poly contact, diffusion contact	CCP and CCA
metal1	=m1	first-level metal	CMF
metal2	=m2	second-level metal	CMS
via2	=via2	metal2/metal3 via, m2/m3 via	CVS
metal3	=m3	third-level metal	CMT
glass	=glass	passivation, overglass, pad	COG



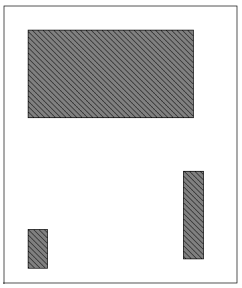
(a) nwell



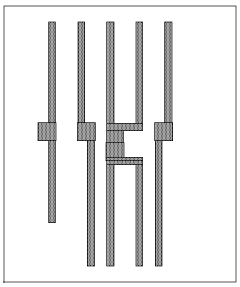
(b) pwell



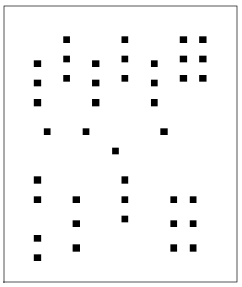
(c) ndiff



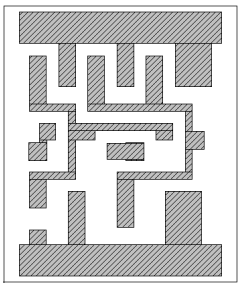
(d) pdiff



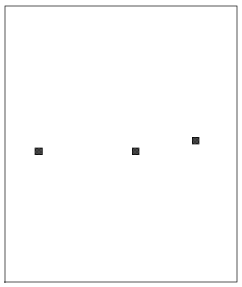
(e) poly



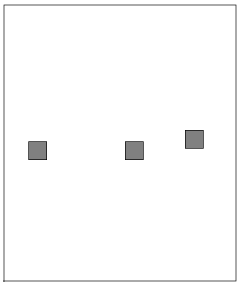
(f) contact



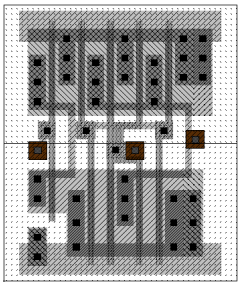
(g) m1



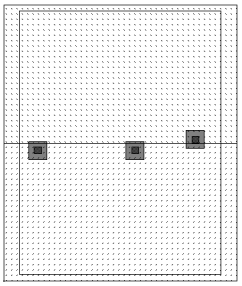
(h) via



(i) m2



(j) cell



(k) phantom

The mask layers of a standard cell

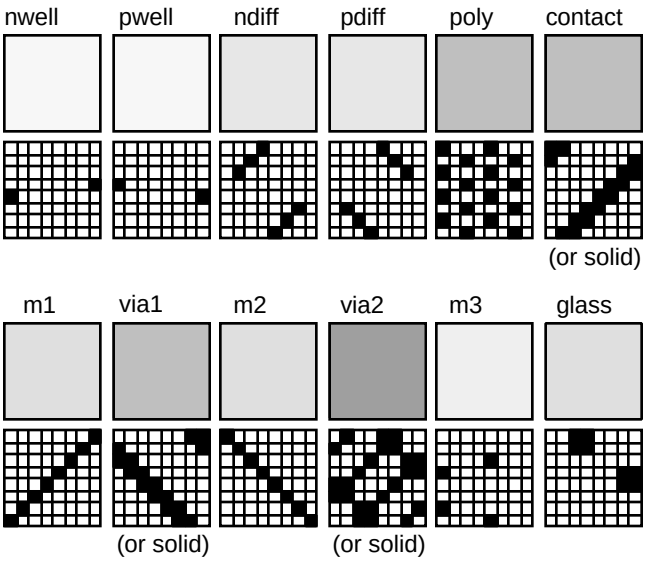
Active *mask*
CAA (mask) = ndiff (drawn) pdiff (drawn)

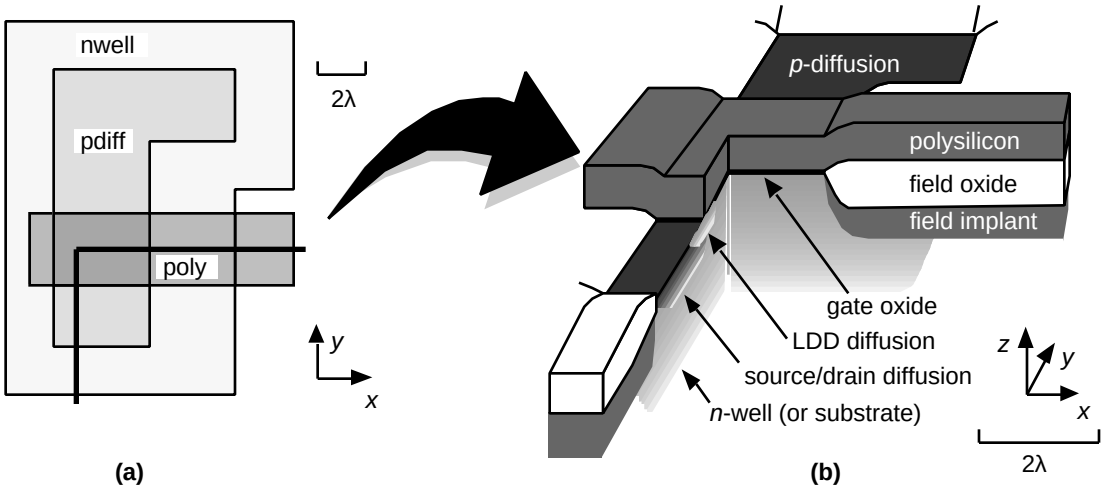
Implant select *masks*
CSN (mask) = grow (ndiff (drawn)) and
CSP (mask) = grow (pdiff (drawn))

Source and drain diffusion (on the *silicon*)
n-diffusion (silicon) = (CAA (mask) CSN (mask)) (¬ CPG (mask)) and
p-diffusion(silicon)=(CAA(mask) CSP(mask)) (¬ CPG(mask))

Source and drain diffusion (on the *silicon*) in terms of *drawn* layers
n-diffusion (silicon) = (ndiff (drawn)) (¬ poly (drawn)) and
p-diffusion (silicon) = (pdiff (drawn)) (¬ poly (drawn))

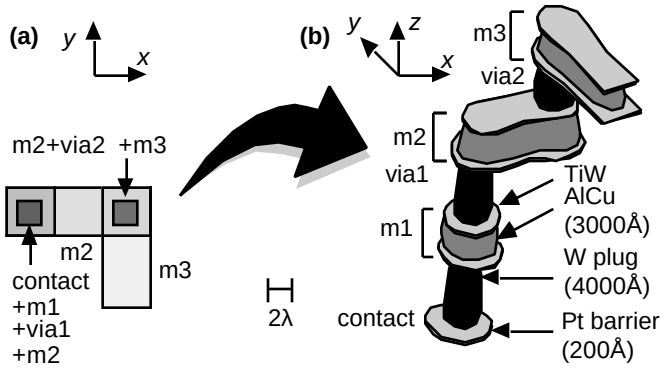
Drawn layers and stipple patterns





The transistor layers

The interconnect layers

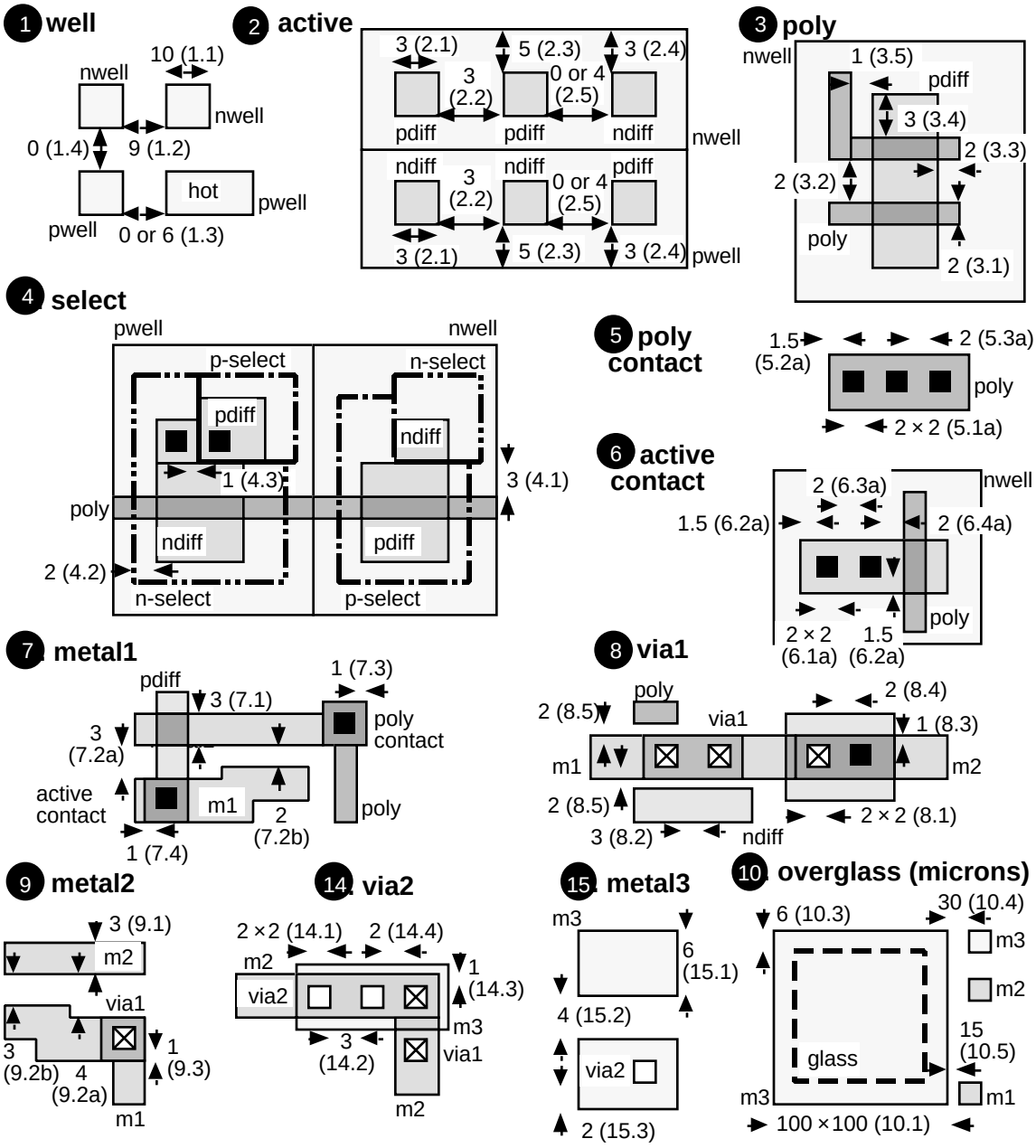


2.2.1 Sheet Resistance

Sheet resistance (1μm)			**Sheet resistance (0.35μm)**		
Layer	**Sheet resistance**	**Units**	**Layer**	**Sheet resistance**	**Units**
n-well	1.15± 0.25	kΩ/square	n-well	1± 0.4	kΩ/square
poly	3.5± 2.0	Ω/square	poly	10± 4.0	Ω/square
n-diffusion	75± 20	Ω/square	n-diffusion	3.5± 2.0	Ω/square
p-diffusion	140± 40	Ω/square	p-diffusion	2.5± 1.5	Ω/square
m1/2	70± 6	mΩ/square	m1/2/3	60± 6	mΩ/square
m3	30± 3	mΩ/square	metal4	30± 3	mΩ/square

Key words: **diffusion** • **Ω/square** (ohms per square) • **sheet resistance** • **silicide** • self-aligned silicide (**salicide**) • **LI**, white metal, local interconnect, metal0, or m0 • **m1** or metal1 • **diffusion contacts** • **polysilicon contacts** • **barrier metal** • **contact plugs** (**via plugs**) • **chemical–mechanical polishing** (**CMP**) • **intermetal oxide** (**IMO**) • **interlevel dielectric** (**ILD**) • **metal vias**, **cuts**, or **vias** • **stacked vias** and **stacked contacts** • **two-level metal** (**2LM**) • **3LM** (**m3** or metal3) • **via1** • **via2** • **metal pitch** • **electromigration** • **contact resistance** and **via resistance**

2.3 CMOS Design Rules



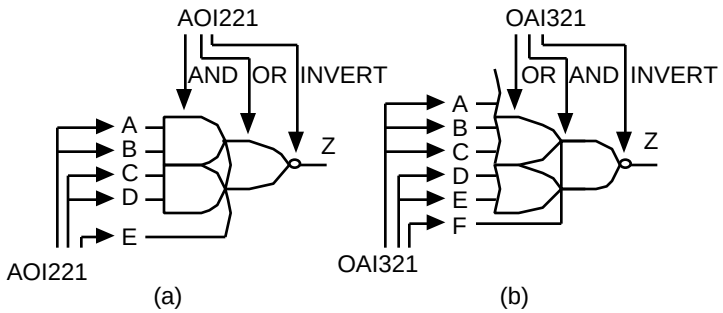
Scalable CMOS design rules

2.4 Combinational Logic Cells

The AOI family of cells with three index numbers or less		
Cell type[1]	**Cells**	**Number of unique cells**
Xa1	X21, X31	2
Xa11	X211, X311	2
Xab	X22, X33, X32	3
Xab1	X221, X331, X321	3
Xabc	X222, X333, X332, X322	4
Total		14

[1]Xabc: X={AOI, AO, OAI, OA}; a, b, c = {2, 3}; {} means “choose one.”

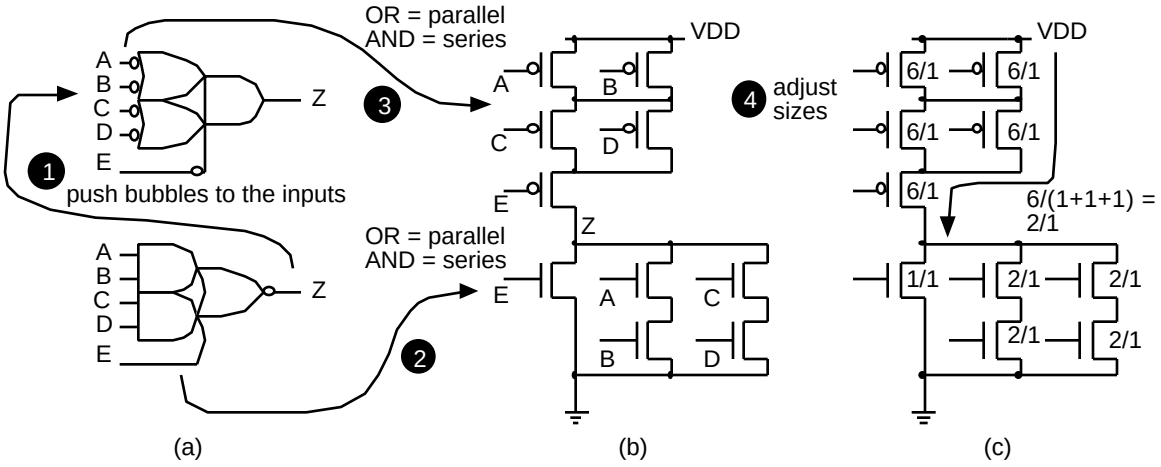
Naming of complex CMOS combinational logic cells



2.4.1 Pushing Bubbles

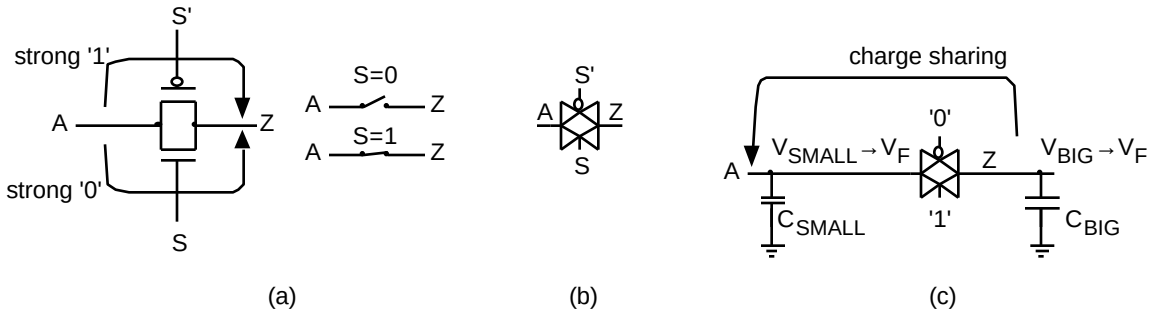
2.4.2 Drive Strength

We **ratio** a cell to adjust its **drive strength** and make $\beta_n = \beta_p$ to create equal rise and fall times



Constructing a CMOS logic cell—an AOI221 • pushing bubbles • **de Morgan's theorem** • **network duals**

2.4.3 Transmission Gates



CMOS transmission gate (TG, TX gate, pass gate, coupler)

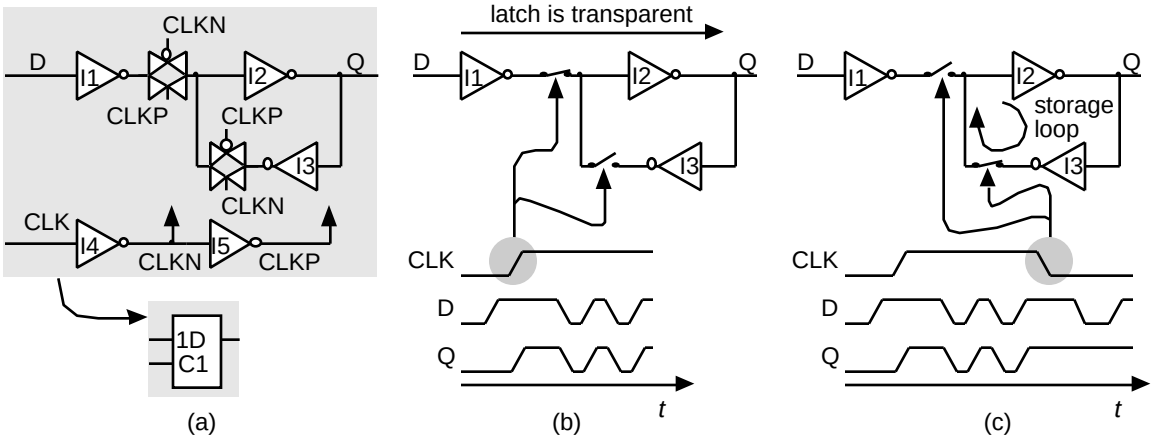
Charge sharing: suppose C_{BIG}=0.2pF and C_{SMALL}=0.02pF, V_{BIG}=0V and V_{SMALL}=5V; then

$$V_F = \frac{(0.2 \times 10^{-12})\,(0) + (0.02 \times 10^{-12})\,(5)}{(0.2 \times 10^{-12}) + (0.02 \times 10^{-12})} = 0.45\ \text{V}$$

2.5 Sequential Logic Cells

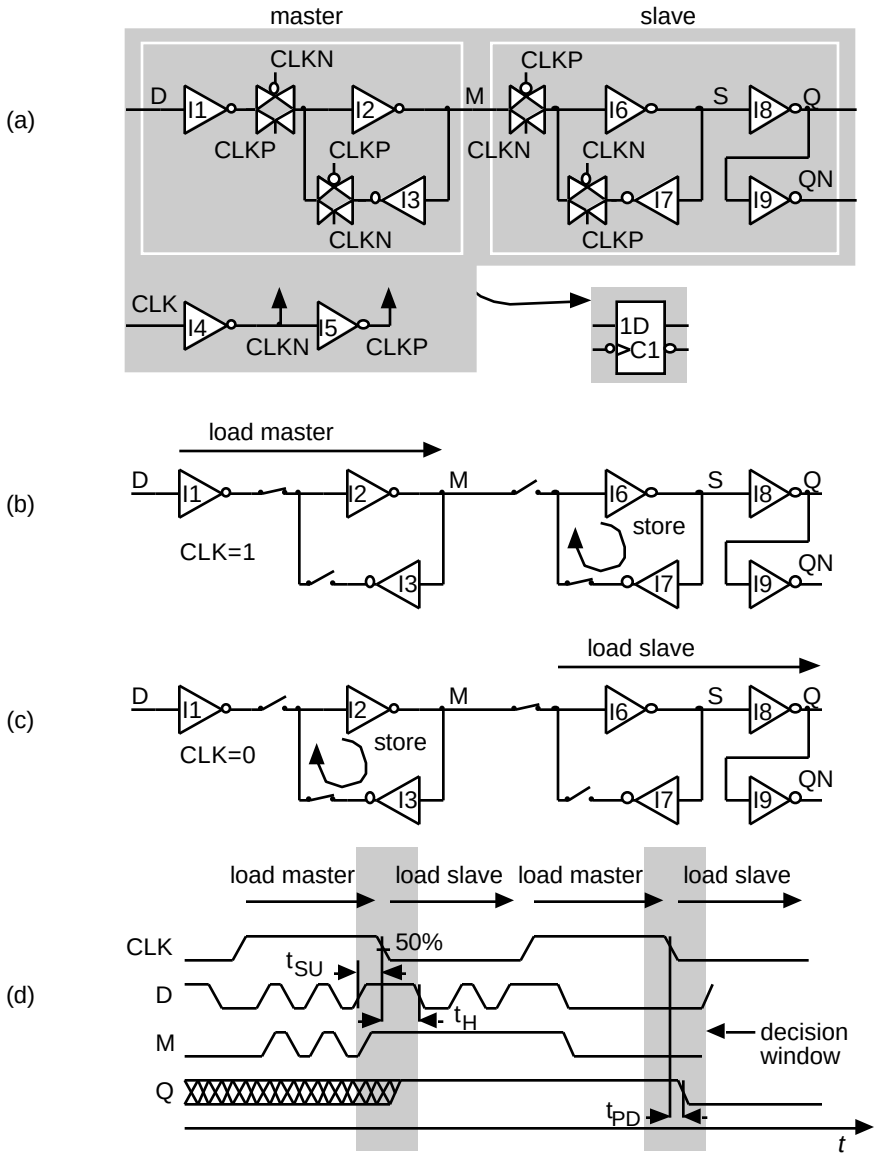
Two choices for sequential logic: **multiphase clocks** or **synchronous design**. We choose the latter.

2.5.1 Latch



CMOS latch • **enable** • **transparent** • **static** • **sequential logic cell** • **storage** • **initial value**

2.5.2 Flip-Flop



CMOS flip-flop

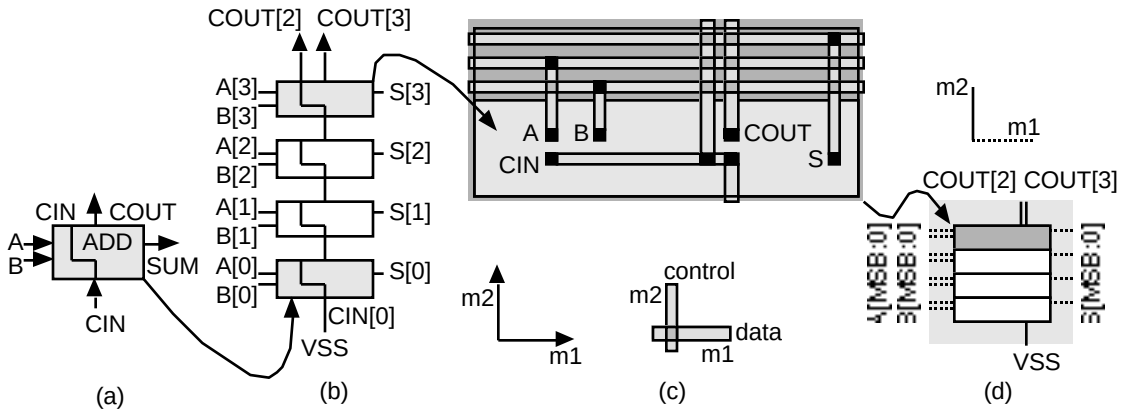
- **master latch • slave latch**
- **active clock edge • negative-edge–triggered flip-flop**
- **setup time** (t_{SU}) • **hold time** (t_H) • clock-to-Q propagation delay (t_{PD})
- decision window

2.6 Datapath Logic Cells

full adder (**FA**): SUM = A B CIN = SUM(A, B, CIN) = PARITY(A, B, CIN) ,
COUT = A · B + A · CIN + B · CIN = MAJ(A, B, CIN).

- **parity function** ('1' for an odd numbers of '1's)
- **majority function** ('1' if the majority of the inputs are '1')

S[*i*] = SUM (A[*i*], B[*i*], CIN)
COUT = MAJ (A[*i*], B[*i*], CIN)



A datapath adder

- Ripple-carry adder (RCA)
- **Data** signals • **control** signals • **datapath** • **datapath cell** or **datapath element**
- Datapath advantages: predictable and equal delay for each bit • built-in interconnect
- Disadvantages of a datapath: overhead • harder design • software is more complex

2.6.1 Datapath Elements

Binary arithmetic

Operation	Binary Number Representation			
	Unsigned	**Signed magnitude**	**Ones' complement**	**Two's complement**
	no change	**if** positive **then** MSB=0 **else** MSB=1	**if** negative **then** flip bits	**if** negative **then** {flip bits; add 1}
3=	0011	0011	0011	0011
–3=	NA	1011	1100	1101
zero=	0000	0000 or 1000	1111 or 0000	0000
max. positive=	1111=15	0111=7	0111=7	0111=7
max. negative=	0000=0	1111=–7	1000=–7	1000=–8
addition= S= A+B =addend+augend SG(A)=sign of A	S=A+B	**if** SG(A)=SG(B) **then** S=A+B **else** {**if** B<A **then** S=A–B **else** S=B–A}	S= A+B+COUT[MSB] COUT is carry out	S=A+B
addition result: OV=overflow, OR=out of range	OR=COUT[MSB] COUT is carry out	**if** SG(A)=SG(B) **then** OV=COUT[MSB] **else** OV=0 (impossible)	OV= XOR(COUT[MSB], COUT[MSB–1])	OV= XOR(COUT[MSB], COUT[MSB–1])
SG(S)=sign of S S= A+B	NA	**if** SG(A)=SG(B) **then** SG(S)=SG(A) **else** {**if** B<A **then** SG(S)=SG(A) **else** SG(S)=SG(B)}	NA	NA
subtraction= D= A–B =minuend –subtrahend	D=A–B	SG(B)=NOT(SG(B)); D=A+B	Z=–B (negate); D=A+Z	Z=–B (negate); D=A+Z

subtraction result: OV=overflow, OR=out of range	OR=BOUT[MSB] BOUT is borrow out	as in addition	as in addition	as in addition
negation: Z=–A (negate)	NA	Z=A; SG(Z)=NOT(SG(A))	Z=NOT(A)	Z=NOT(A)+1

2.6.2 Adders

Generate, G[*i*] and **propagate**, P[*i*]

method 1

G[i] = A[i] · B[i]

P[*i*] = A[*i*] B[*i*

C[*i*] = G[*i*] + P[*i*] · C[*i*–1]

S[*i*] = P[*i*] C[*i*–1]

method 2

G[*i*] = A[*i*] · B[*i*]

P[*i*] = A[*i*] + B[*i*]

C[*i*] = G[*i*] + P[*i*] · C[*i*–1]

S[*i*] = A[*i*] B[*i*] C[*i*–1]

Carry signal:

either C[*i*] = A[*i*] · B[*i*] + P[*i*] · C[*i* – 1]

or C[*i*] = (A[*i*] + B[*i*]) · (P[*i*]' + C[*i* – 1]), where P[*i*]'=NOT(P[*i*])

Carry chain using two-input NAND gates, one per cell:

even stages

C1[*i*]' = P[*i*] · C3[*i* – 1] · C4[*i* – 1]

C2[*i*] = A[*i*] + B[*i*]

C[*i*] = C1[*i*] · C2[*i*]

odd stages

C3[*i*]' = P[*i*] · C1[*i* – 1] · C2[*i* – 1]

C4[*i*]' = A[*i*] · B[*i*]

C[*i*] = C3[*i*]'+ C4[*i*]'

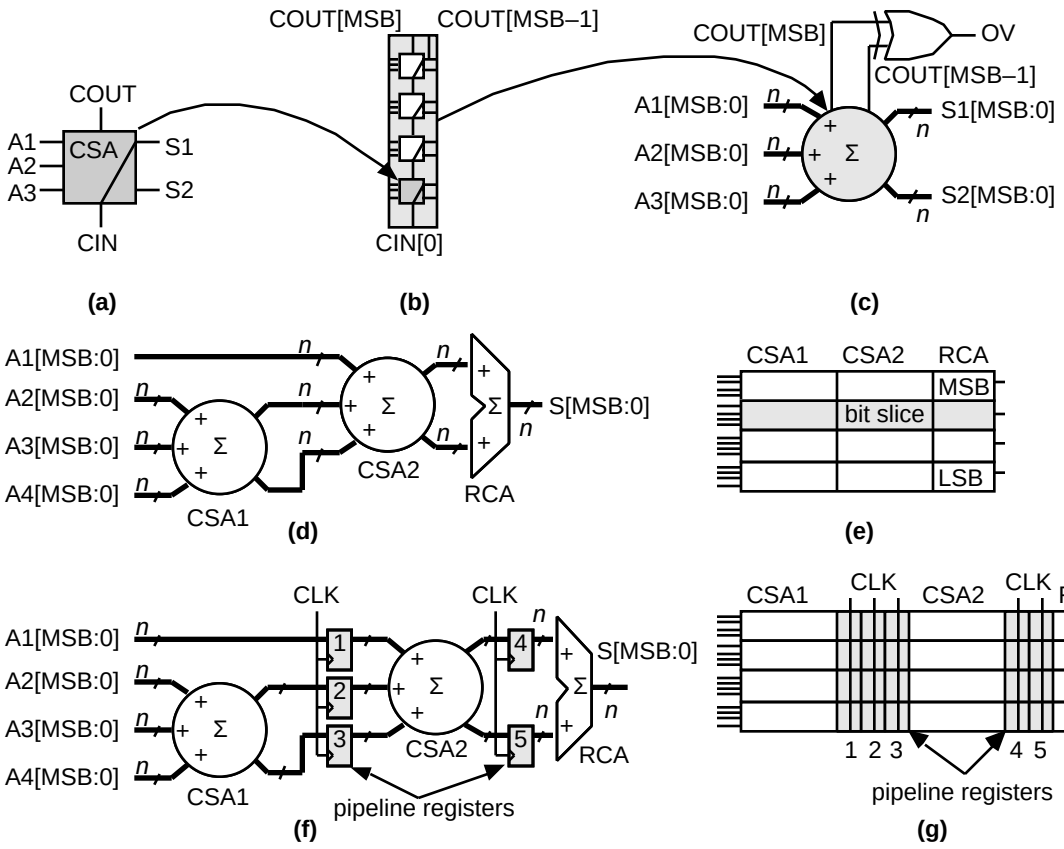
Carry-save adder (**CSA**) cell CSA(A1[*i*], A2[*i*], A3[*i*], CIN, S1[*i*], S2[*i*], COUT) has three outputs:

S1[*i*] = CIN ,

S2[*i*] = A1[*i*] A2[*i*] A3[*i*] = PARITY(A1[*i*], A2[*i*], A3[*i*])

COUT = A1[*i*] · A2[*i*] + [(A1[*i*] + A2[*i*]) · A3[*i*]] = MAJ(A1[*i*], A2[*i*], A3[*i*])

Carry-propagate adder (**CPA**)



The carry-save adder (CSA) • **pipeline** • **latency** • **bit slice**

carry-bypass adders (**CBA**):

$C[7]=(G[7]+P[7]\cdot C[6])\cdot BYPASS'+C[3]\cdot BYPASS$

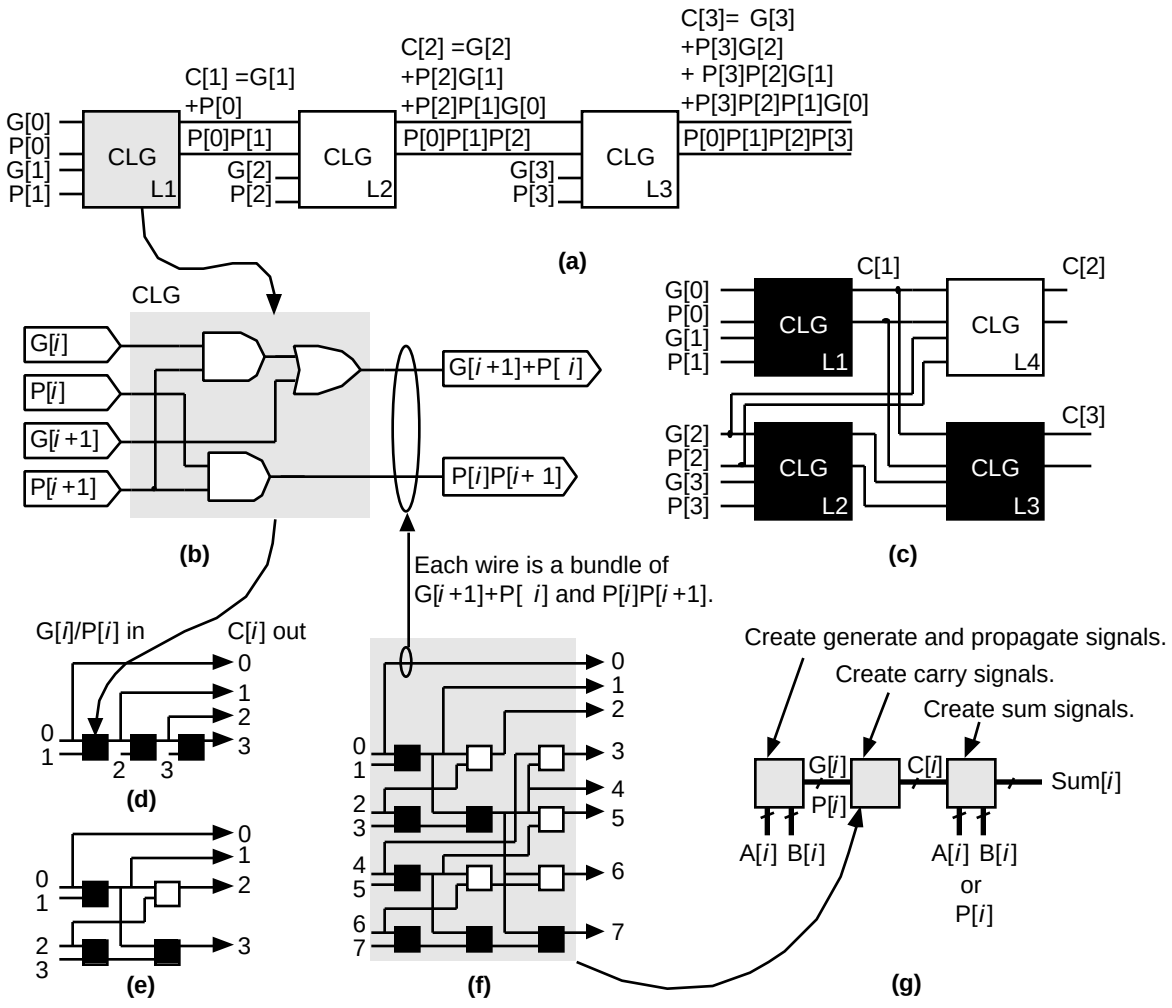
carry-skip adder:

$CSKIP[i] = (G[i] + P[i] \cdot C[i-1]) \cdot SKIP' + C[i-2] \cdot SKIP$

Carry-lookahead adder (**CLA**, for example the **Brent–Kung adder**):

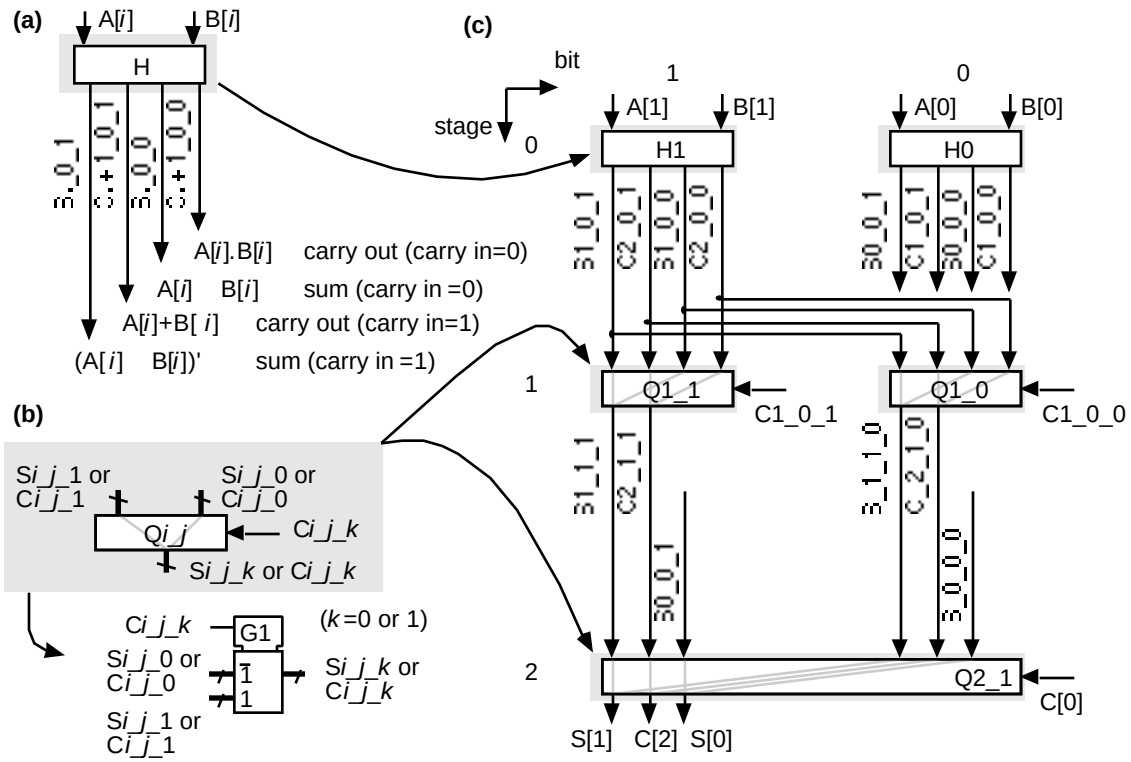
$$
\begin{aligned}
C[1] &= G[1] + P[1] \cdot C[0] \\
&= G[1] + P[1] \cdot (G[0] + P[1] \cdot C[-1]) \\
&= G[1] + P[1] \cdot G[0]
\end{aligned}
$$

$$
\begin{aligned}
C[2] &= G[2] + P[2] \cdot G[1] + P[2] \cdot P[1] \cdot G[0] , \\
C[3] &= G[3] + P[2] \cdot G[2] + P[2] \cdot P[1] \cdot G[1] + P[3] \cdot P[2] \cdot P[1] \cdot G[0]
\end{aligned}
$$



The Brent–Kung carry-lookahead adder

Carry-select adder duplicates two small adders for the cases CIN='0' and CIN='1' and then uses a MUX to select the case that we need



Ci_j_k =carry in to the *i*th bit assuming the carry in to the *j*th bit is *k* (*k* =0 or 1)
Si_j_k =sum at the *i*th bit assuming the carry in to the *j*th bit is *k* (*k* =0 or 1)

The conditional-sum adder

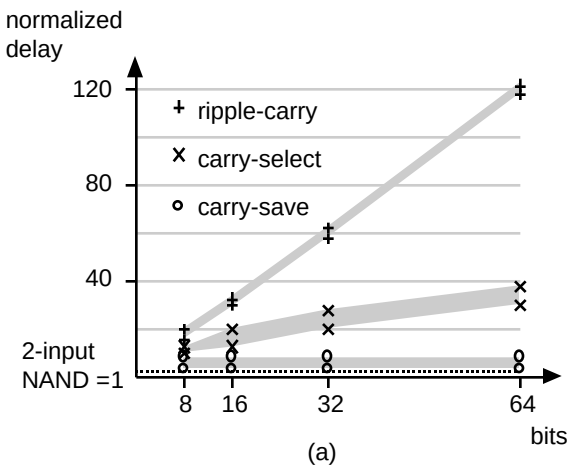
2.6.3 A Simple Example

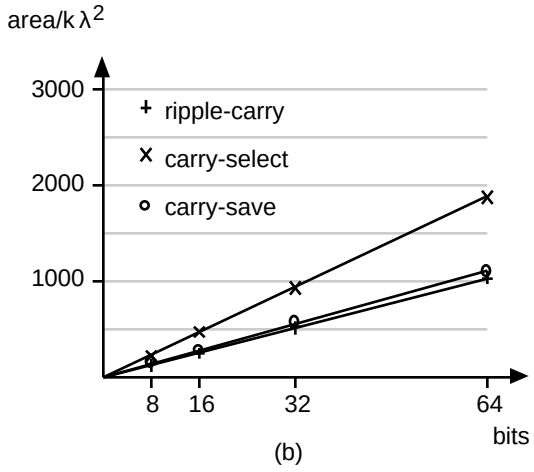
An 8-bit conditional-sum adder

```
module m8bitCSum (C0, a, b, s, C8); // Verilog conditional-sum adder
for an FPGA                                                    //1
input [7:0] C0, a, b; output [7:0] s; output C8;               //2
wire
A7,A6,A5,A4,A3,A2,A1,A0,B7,B6,B5,B4,B3,B2,B1,B0,S8,S7,S6,S5,S4,S3,S2
,S1,S0;                                                        //3
wire C0, C2, C4_2_0, C4_2_1, S5_4_0, S5_4_1, C6, C6_4_0, C6_4_1,
C8;                                                            //4
assign {A7,A6,A5,A4,A3,A2,A1,A0} = a;assign
{B7,B6,B5,B4,B3,B2,B1,B0} = b;                                 //5
assign s = { S7,S6,S5,S4,S3,S2,S1,S0 };                       //6
assign S0 = A0^B0^C0 ; // start of level 1: & = AND, ^ = XOR, | =
OR, ! = NOT                                                    //7
assign S1 = A1^B1^(A0&B0|(A0|B0)&C0) ;                         //8
assign C2 = A1&B1|(A1|B1)&(A0&B0|(A0|B0)&C0) ;                 //9
assign C4_2_0 = A3&B3|(A3|B3)&(A2&B2) ;assign  C4_2_1 =
A3&B3|(A3|B3)&(A2|B2) ;                                       //10
assign S5_4_0 = A5^B5^(A4&B4) ;assign S5_4_1 = A5^B5^(A4|B4) ; //11
assign C6_4_0 = A5&B5|(A5|B5)&(A4&B4) ;assign C6_4_1 =
A5&B5|(A5|B5)&(A4|B4) ;                                       //12
assign S2 = A2^B2^C2 ; // start of level 2                    //13
assign S3 = A3^B3^(A2&B2|(A2|B2)&C2) ;                        //14
assign S4 = A4^B4^(C4_2_0|C4_2_1&C2) ;                        //15
assign S5 = S5_4_0&
!(C4_2_0|C4_2_1&C2)|S5_4_1&(C4_2_0|C4_2_1&C2) ;               //16
assign C6 = C6_4_0|C6_4_1&(C4_2_0|C4_2_1&C2) ;                //17
assign S6 = A6^B6^C6 ; // start of level 3                    //18
assign S7 = A7^B7^(A6&B6|(A6|B6)&C6) ;                        //19
assign C8 = A7&B7|(A7|B7s)&(A6&B6|(A6|B6)&C6) ;               //20
endmodule                                                      //21
```

2.6.4 Multipliers

- Mental arithmetic: 15 (**multiplicand**) × 19 (**multiplier**) = 15×(20–1) = 15×$2\bar{1}$
- Suppose we want to multiply by B=00010111 (decimal 16+4+2+1=23)
- Use the canonical signed-digit vector (**CSD vector**) D=$0010\bar{1}001$ (decimal 32–8+1=23)
- B has a **weight** of 4, but D has a weight of 3 — and saves hardware





Datapath adders

To **recode** (or encode) any binary number, B, as a CSD vector, D: $D_i = B_i + C_i - 2C_{i+1}$, where C_{i+1} is the carry from the sum of $B_{i+1} + B_i + C_i$ (we start with $C_0=0$).

If B=011 ($B_2=0$, $B_1=1$, $B_0=1$; decimal 3), then: $D_0 = B_0 + C_0 - 2C_1 = 1 + 0 - 2 = \bar{1}$,

$D_1 = B_1 + C_1 - 2C_2 = 1 + 1 - 2 = 0$,

$D_2 = B_2 + C_2 - 2C_3 = 0 + 1 - 0 = 1$,

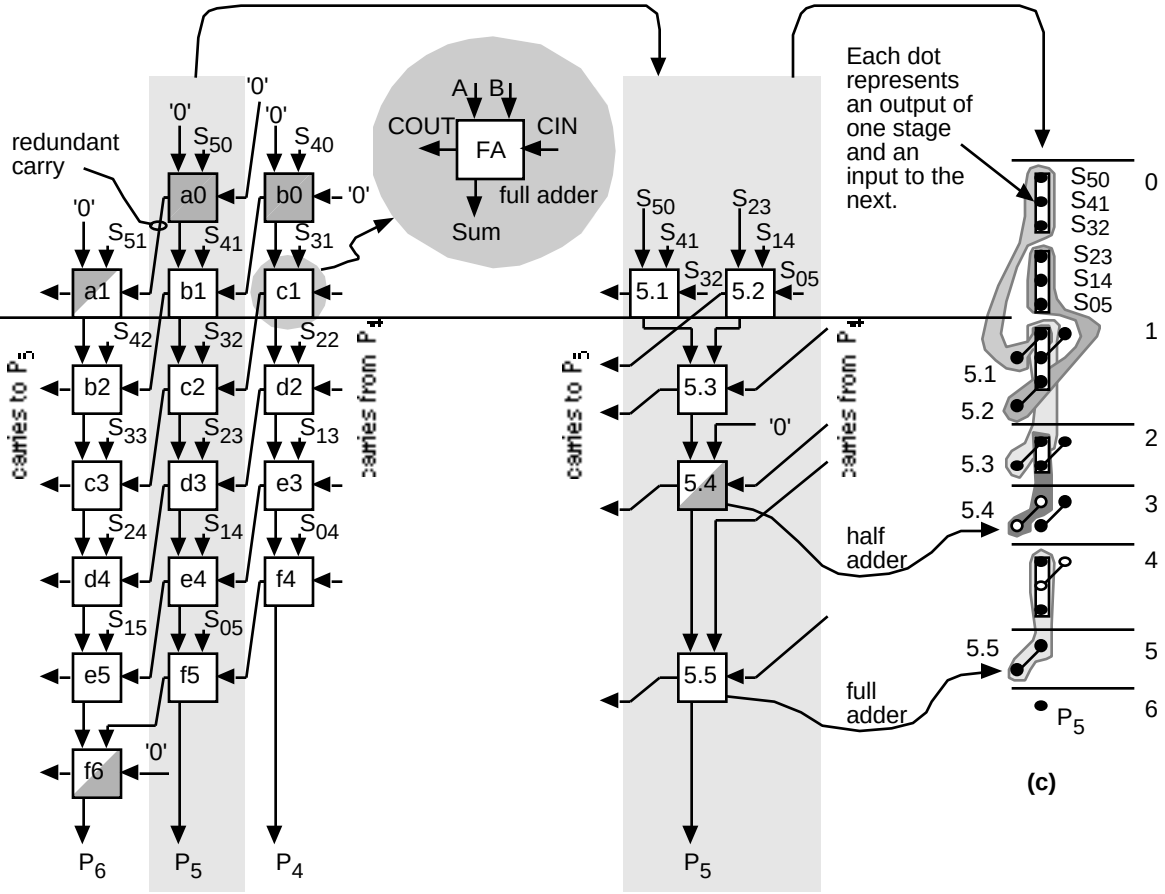
so that D= $10\bar{1}$ (decimal 4–1=3).

We can use a **radix** other than 2, for example **Booth encoding** (radix-4):

B=101001 (decimal 9–32=–23) E= $\bar{1}\,\bar{2}1$ (decimal –16–8+1=–23)

B=01011 (eleven) E= $1\bar{1}\,\bar{1}$ (16–4–1)

B=101 E= $\bar{1}1$

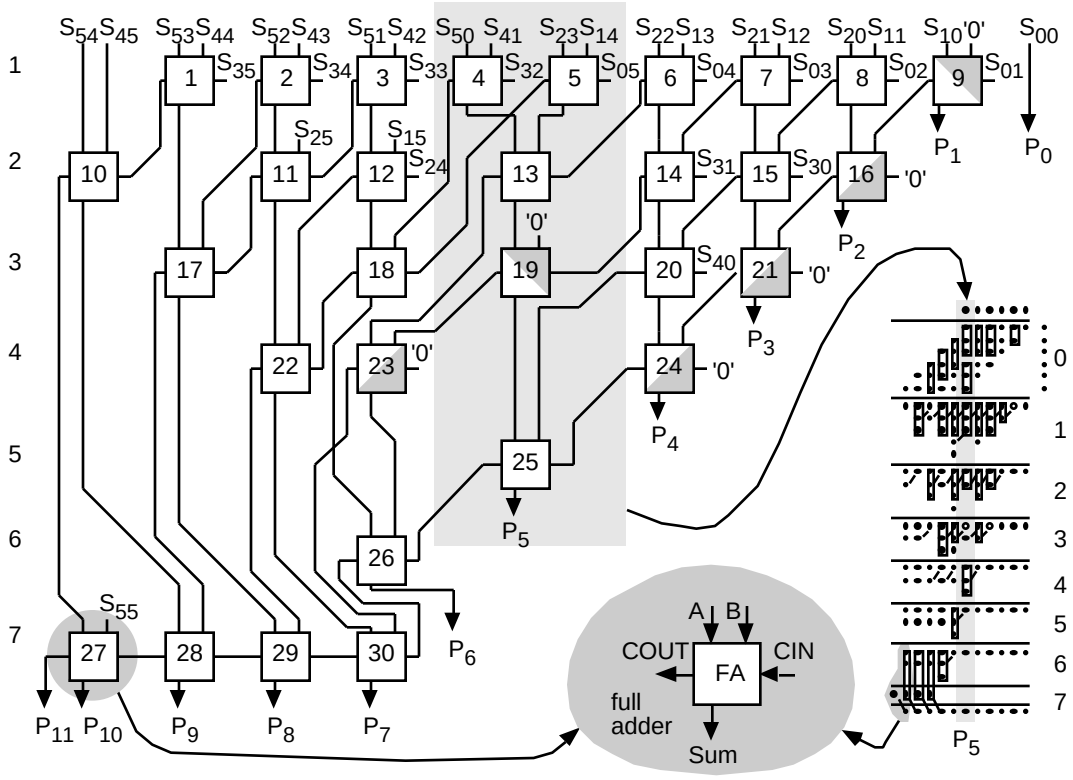


Tree-based multiplication – at each stage we have the following three choices:

(1) sum three outputs using a full adder

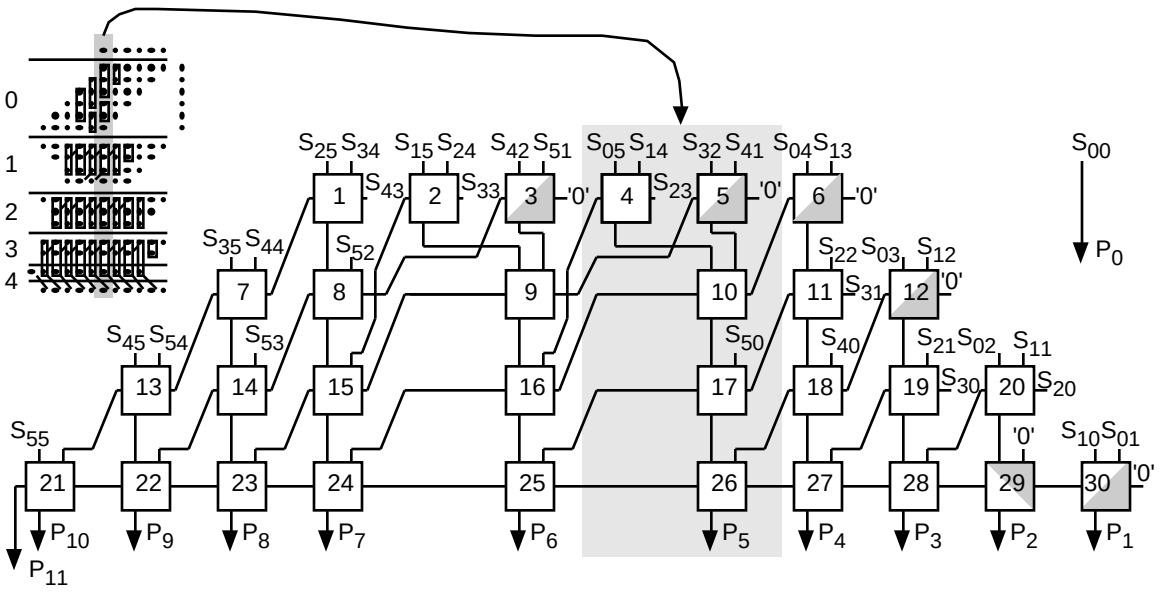
(2) sum two outputs using a half adder

(3) pass the outputs to the next stage



A Wallace-tree multiplier works forward from the multiplier inputs

- Full adder is a **3:2 compressor** or **(3, 2) counter**
- Half adder is a **(2, 2) counter**

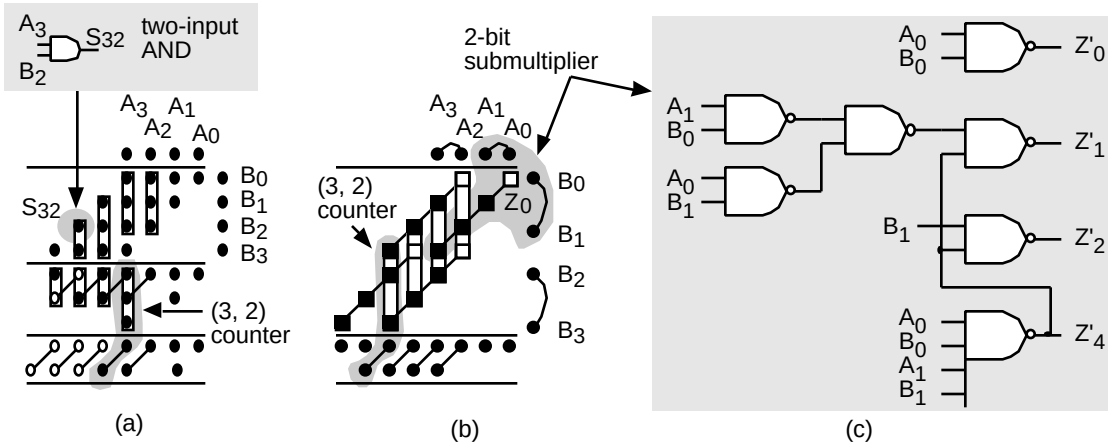


The Dadda multiplier works backward from the final product

• Each stage has a maximum of 2, 3, 4, 6, 9, 13, 19, ...outputs (each successive stage is 3/2 times larger—rounded down to an integer

The number of stages and thus delay (in units of an FA delay—excluding the CPA) for an *n*-bit tree-based multiplier using (3, 2) counters is

$\log_{1.5} n = \log_{10} n / \log_{10} 1.5 = \log_{10} n / 0.176$



Ferrari–Stefanelli architecture “nests” multipliers

2.6.5 Other Arithmetic Systems

binary	decimal	redundant binary	CSD vector	
1010111	87	$10\bar{1}0\bar{1}00\bar{1}$	$10\bar{1}0\bar{1}00\bar{1}$	addend
+ 1100101	101	+ $1\bar{1}100\bar{1}1\bar{1}$	+ 01100101	augend
		$0100\bar{1}\bar{1}10$	= $\bar{1}\bar{1}00\bar{1}100$	intermediate sum
		$1\bar{1}000\bar{1}0\bar{1}$	11000000	intermediate carry
= 10111100	= 188	$1\bar{1}1000\bar{1}00$	$10\bar{1}00\bar{1}100$	sum

Redundant binary addition • redundant binary encoding avoids carry propagation					
A[*i*]	**B[*i*]**	**A[*i*–1]**	**B[*i*–1]**	**Intermediate sum**	**Intermediate carry**
$\bar{1}$	$\bar{1}$	x	x	0	$\bar{1}$
$\bar{1}$	0	A[i–1]=0/1 and B[i–1]=0/1		$\bar{1}$	0
0	$\bar{1}$	A[i–1]=$\bar{1}$ or B[i–1]=$\bar{1}$		1	$\bar{1}$
$\bar{1}$	1	x	x	0	0
1	$\bar{1}$	x	x	0	0
0	0	x	x	0	0
0	1	A[i–1]=0/1 and B[i–1]=0/1		$\bar{1}$	1
1	0	A[i–1]=$\bar{1}$ or B[i–1]=$\bar{1}$		1	0
1	1	x	x	0	1

- 101 (decimal) is 1100101 (in binary and CSD vector) or $1\bar{1}100111$
- 188 (decimal) is 10111100 (in binary), $1\bar{1}1000\bar{1}00$, $10\bar{1}00\bar{1}100$, or $10\bar{1}000\bar{1}00$ (CSD vector)
- $10\bar{1}$ is represented as 010010 (using sign magnitude) — rather wasteful

Residue number system

- 11 (decimal) is represented as [1, 2] residue (5, 3)
- $11R_5$=11 mod 5=1 and $11R_3$=11 mod 3=2
- The **size** of this system is 3×5=15
- We can now add, subtract, or multiply without using any carry

4	[4, 1]	12	[2, 0]	3	[3, 0]
+ 7	+ [2, 1]	– 4	– [4, 1]	× 4	× [4, 1]
= 11	= [1, 2]	= 8	= [3, 2]	= 12	= [2, 0]

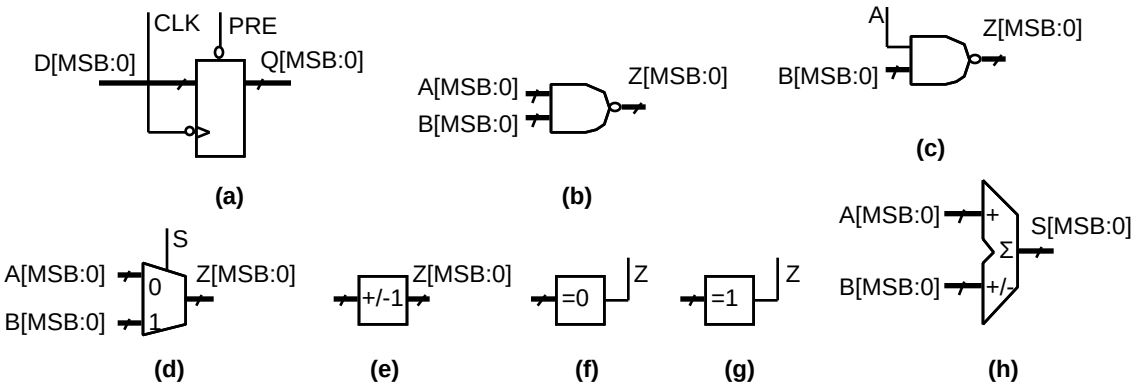
The 5, 3 residue number system

n	residue 5	residue 3	n	residue 5	residue 3	n	residue 5	residue 3
0	0	0	5	0	2	10	0	1
1	1	1	6	1	0	11	1	2
2	2	2	7	2	1	12	2	0
3	3	0	8	3	2	13	3	1
4	4	1	9	4	0	14	4	2

2.6.6 Other Datapath Operators

Full subtracter

$$DIFF = A \quad NOT(B) \quad NOT(BIN)$$
$$= SUM(A, NOT(B), NOT(BIN))$$
$$NOT(BOUT) = A \cdot NOT(B) + A \cdot NOT(BIN) + NOT(B) \cdot NOT(BIN)$$
$$= MAJ(NOT(A), B, NOT(BIN))$$



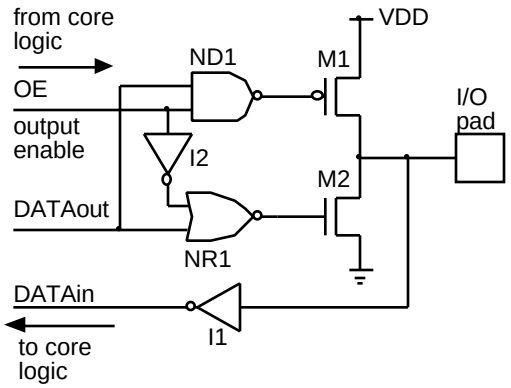
Symbols for datapath elements

Keywords: adder/subtracter • barrel shifter • normalizer • denormalizer • leading-one detector • priority encoder • exponent correcter • accumulator • multiplier–accumulator (MAC) • incrementer • decrementer • incrementer/decrementer • all-zeros detector • all-ones detector • register file • first-in first-out register (FIFO) • last-in first-out register (LIFO)

2.7 I/O Cells

*Keywords:*Tri-State® is a registered trademark of National Semiconductor) • **drivers** • **contention** • **bus keeper** or **bus-hold** cell (TI calls this Bus-Friendly logic) • **slew rate** • **power-supply bounce** • **simultaneously switching outputs** (SSOs) • **quiet-I/O** • **bidirectional I/O** • **open-drain** • **level shifter** • **electrostatic discharge**, or **ESD** • **electrical overstress** (EOS) • **ESD implant** • **human-body model** (**HBM**) • **machine model** (**MM**) • **charge-device model** (**CDM**, also called device charge–discharge) • **latch-up** • **undershoot** • **overshoot** • **guard rings**

A three-state bidirectional output buffer



2.8 Cell Compilers

Keywords: silicon compilers • RAM compiler • multiplier compiler • single-port RAM • dual-port RAMs • multiport RAMs • asynchronous • synchronous • model compiler • netlist compiler • correct by construction

2.9 Summary

- The use of transistors as switches
- The difference between a flip-flop and a latch
- The meaning of setup time and hold time

- Pipelines and latency
- The difference between datapath, standard-cell, and gate-array logic cells
- Strong and weak logic levels
- Pushing bubbles
- Ratio of logic
- Resistance per square of layers and their relative values in CMOS
- Design rules and λ

2.10 Problems

Suggested homework: 2.1, 2.2, 2.38, 2.39 (from *ASICs... the book*)