Digital Integrated Circuits

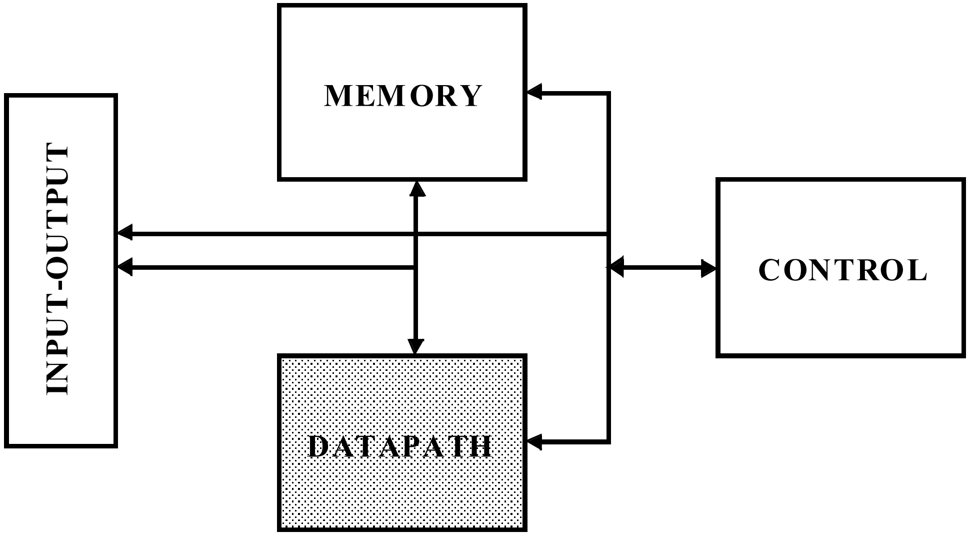
Arithmetic Circuits

Fuyuzhuo

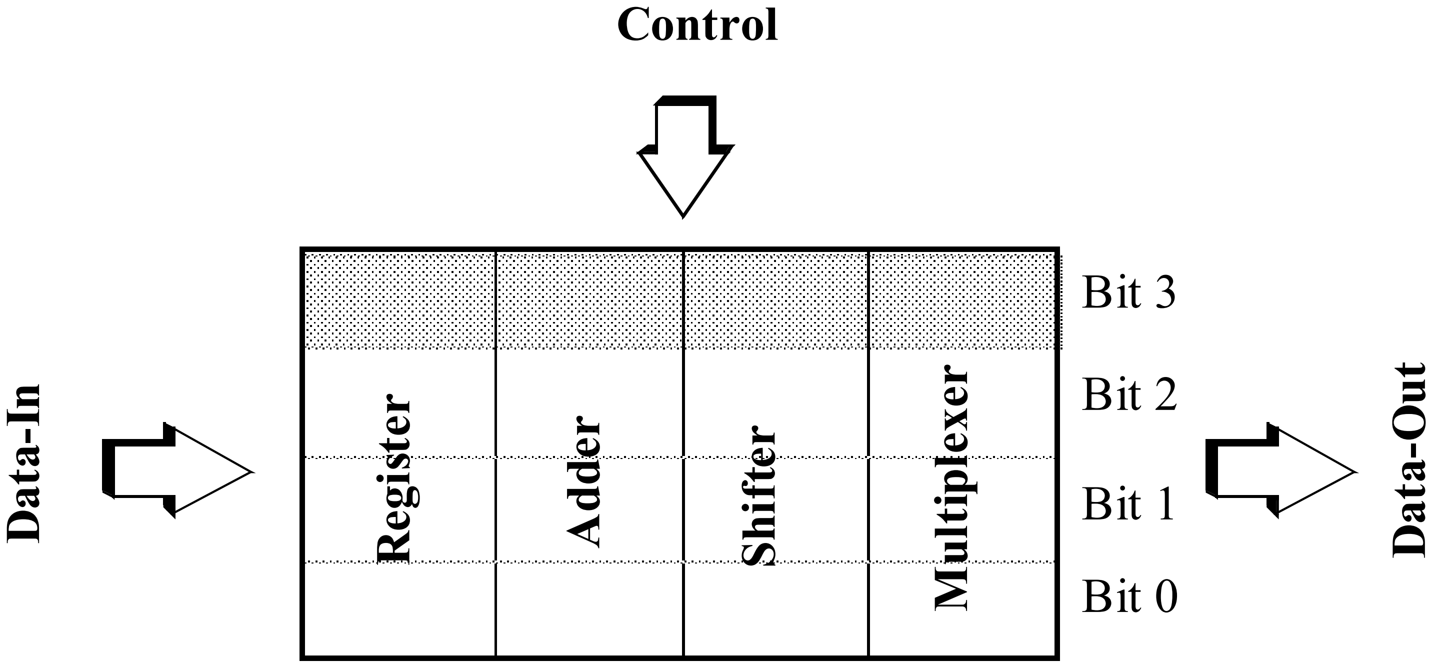
Chapter 5
Arithmetic Circuits

Building Blocks for Digital Architectures

- Arithmetic unit
  - Bit-sliced datapath (adder, multiplier, shifter, comparator, etc.)
- Memory
  - RAM, ROM, Buffers, Shift registers
- Control
  - Finite state machine (PLA, random logic.)
  - Counters
- Interconnect
  - Switches\Arbiters\ Bus



Bit-Sliced Design



Tile identical processing elements

outline

- ***Adder***
- ***Datapath functional unit***
  - ***Comparators***
  - ***Shifters***
  - ***Multi-input Adders***
- ***Multipliers***

Adders

Multitudes of contrivances were designed,and almost endless drawings made, for the purpose of economizing the time and simplifying the mechanism of carriage

__**charles babbage**, on difference engine No.1,1864

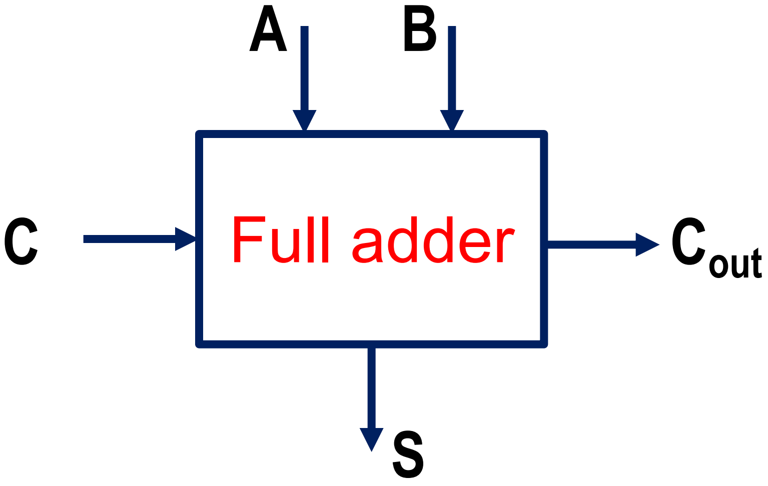
Outline

- Single-bit Addition architecture
- Group Definition
- Manchester Carry Chain
- Classic adders
- Tree Adder

Outline

- ***Single-bit Addition architecture***
- Group Definition
- Manchester Carry Chain
- Classic adders
- Tree Adder

The Binary Adder



$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC =$$
$$\bar{A}\,(\bar{B}C + B\bar{C})\ +A\,(\bar{B}\bar{C} + BC)$$
$$= \bar{A}\,(B \oplus C)\ +A\,(\overline{B \oplus C})\ =$$
$$A \oplus B \oplus C$$

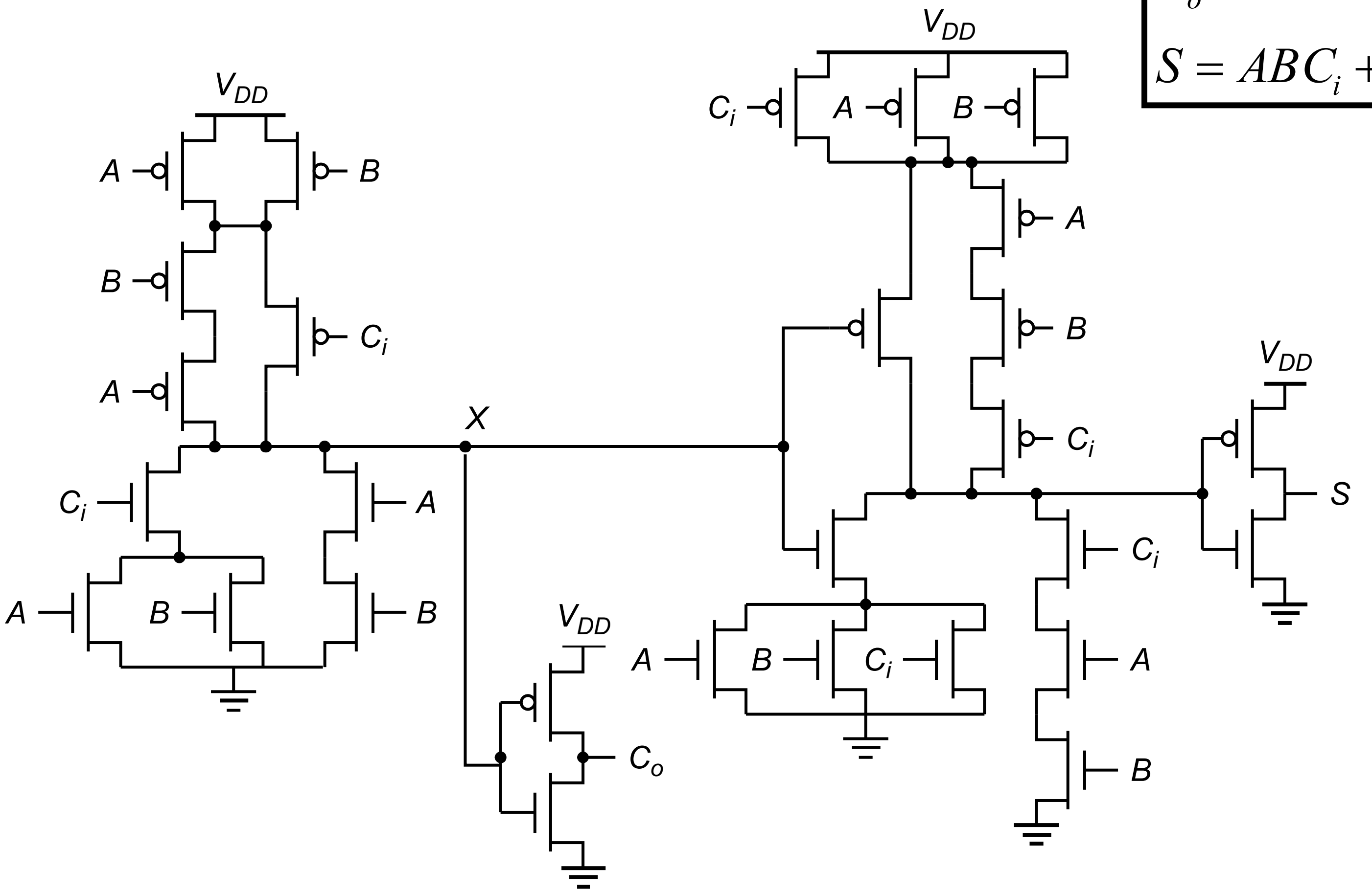
$$C_{out} = \bar{A}BC + A\bar{B}C + AB\bar{C} +$$
$$ABC =\ (\bar{A}B + A\bar{B})\ C + AB =$$
$$(A \oplus B)\ C + AB =\ (A + B)$$
$$C + AB$$

A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full Adder Design I

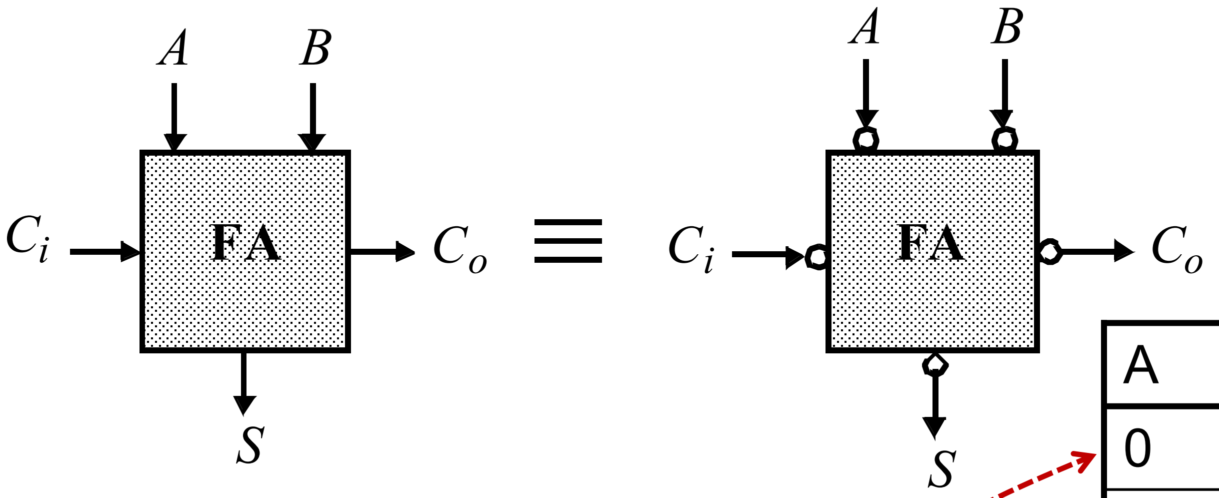
$$C_o = AB + (B + A)C_i$$

$$S = ABC_i + \overline{C_o}(A + B + C_i)$$



Complimentary Static CMOS Full Adder 28 Transistors

Inversion Property



$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\overline{C_i})$$

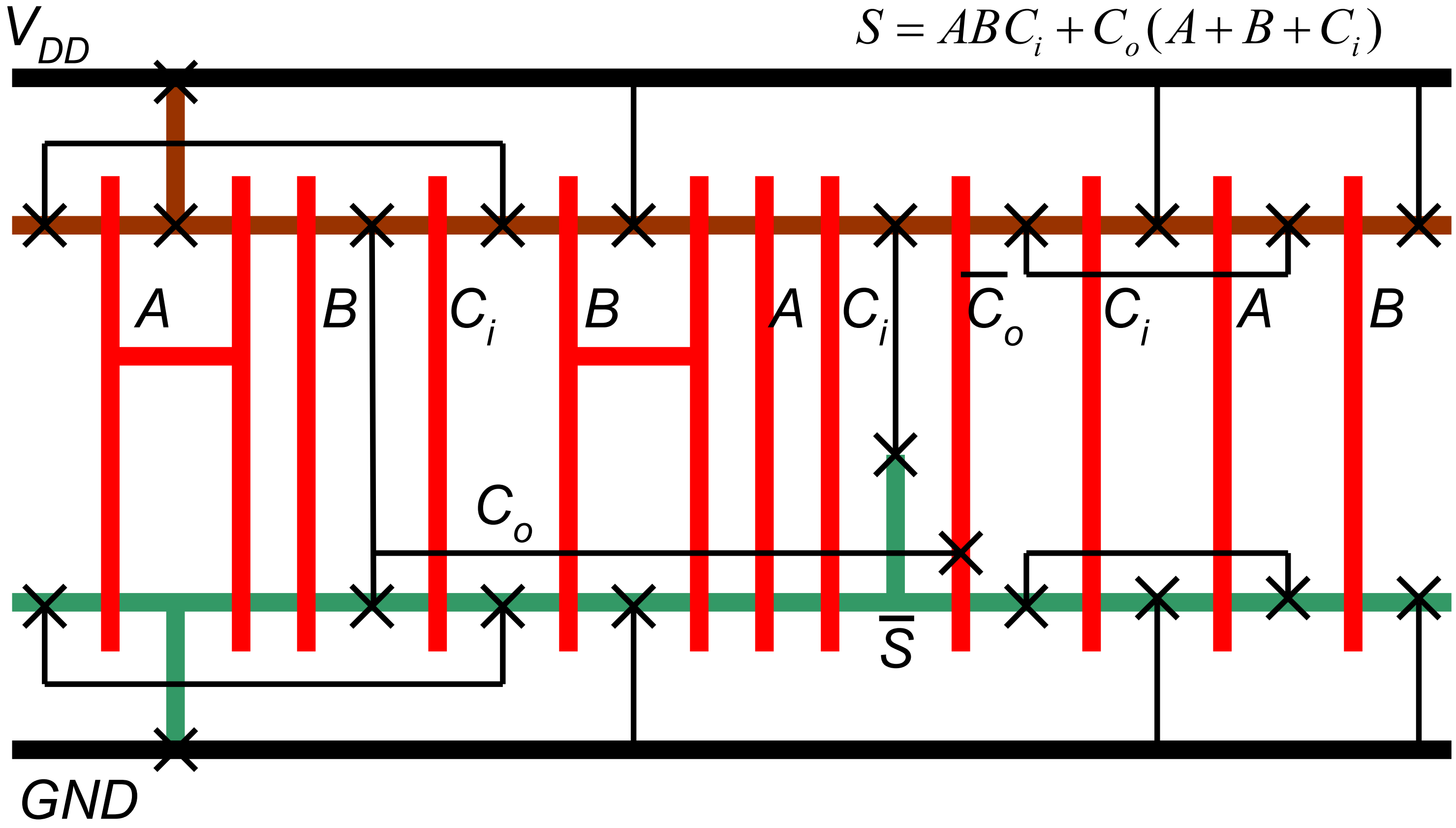
$$\overline{C_o}(A,B,C_i) = C_o(\bar{A},\bar{B},\overline{C_i})$$

A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Mirror Adder-Stick Diagram

$$C_o = AB + BC_i + AC_i$$

$$S = ABC_i + \overline{C_o}(A + B + C_i)$$

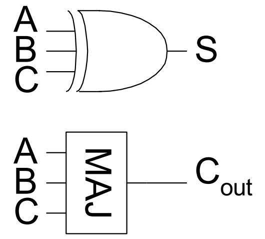


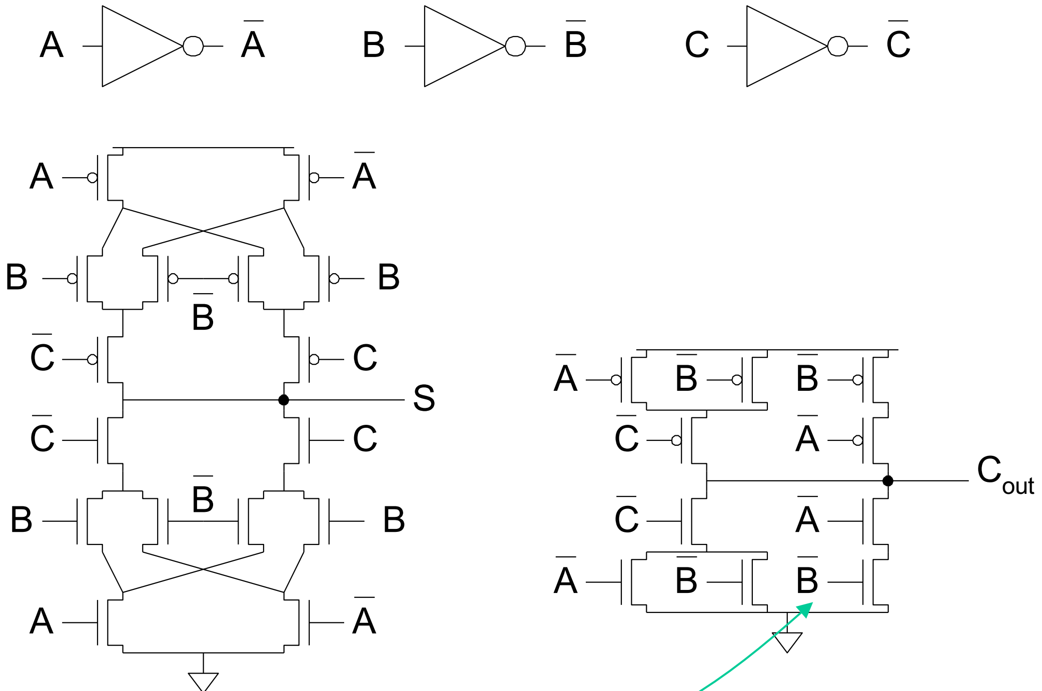
Full Adder Design for mirror

- implementation from eqns

$$S = A \oplus B \oplus C$$

$$C_{out} = MAJ(A,B,C)$$





$$S = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC$$

$$= A\oplus B\oplus C = P\oplus C$$

$$C_{out} = AB + (A+B)C$$

$$= \overline{\bar{A}\bar{B} + (\bar{A}+\bar{B})\bar{C}} = MAJ(A,B,C)$$

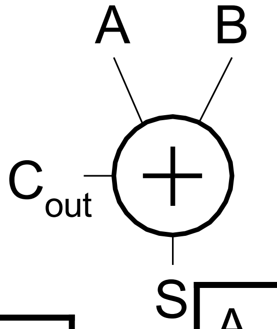
Fewer than mentioned above because of sharing some transistors for XOR gate

Single-Bit Addition

Half Adder

$$S = A \oplus B$$

$$C_{out} = AB$$



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC$$

$$= A \oplus B \oplus C = P \oplus C$$

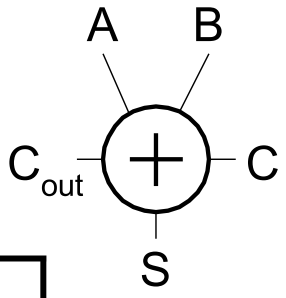
$$C_{out} = AB + (A + B)C$$

$$= \overline{\bar{A}\bar{B} + (\bar{A} + \bar{B})\bar{C}} = MAJ(A, B, C)$$

Full Adder

$$S = A \oplus B \oplus C$$

$$C_{out} = MAJ(A, B, C)$$



A	B	C	C_{out}	S	P	G
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	0
1	0	0	0	1	1	0
1	0	1	1	0	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	1

Delete/Kill

propagate

generate

PGK

- For a full adder, define what happens to carries
  - **Generate:** C_{out} = 1 independent of C
    - G = A • B
  - **Propagate:** C_{out} = C
    - P = A ⊕ B
  - **Kill:** C_{out} = 0 independent of C
    - K /D= ~A • ~B
- *Note that we will be sometimes using an alternate definition for … … when using P for carry chain(not for Summ)*

$$\boxed{\begin{array}{c} C_o(G,P) = G + PC_i \\ S(G,P) = P \oplus C_i \end{array}}$$

Propagate (P) = A+B

Full Adder Design IV

$S = A \oplus B \oplus C = P \oplus \mathrm{C}$

$C_{out} = AB + (A \oplus B)C = AAB + A\bar{A}\bar{B} + (A \oplus B)C = A\bar{P} + PC$

$\overline{C_{out}} = (\bar{A} + P)(\bar{P} + \bar{C}) = \bar{A}\bar{P} + \bar{A}\bar{C} + P\bar{C} = \bar{A}\bar{P} + \bar{A}\bar{C}\bar{P} + \bar{A}\bar{C}P + P\bar{C} = \bar{A}\bar{P} + P\bar{C}$

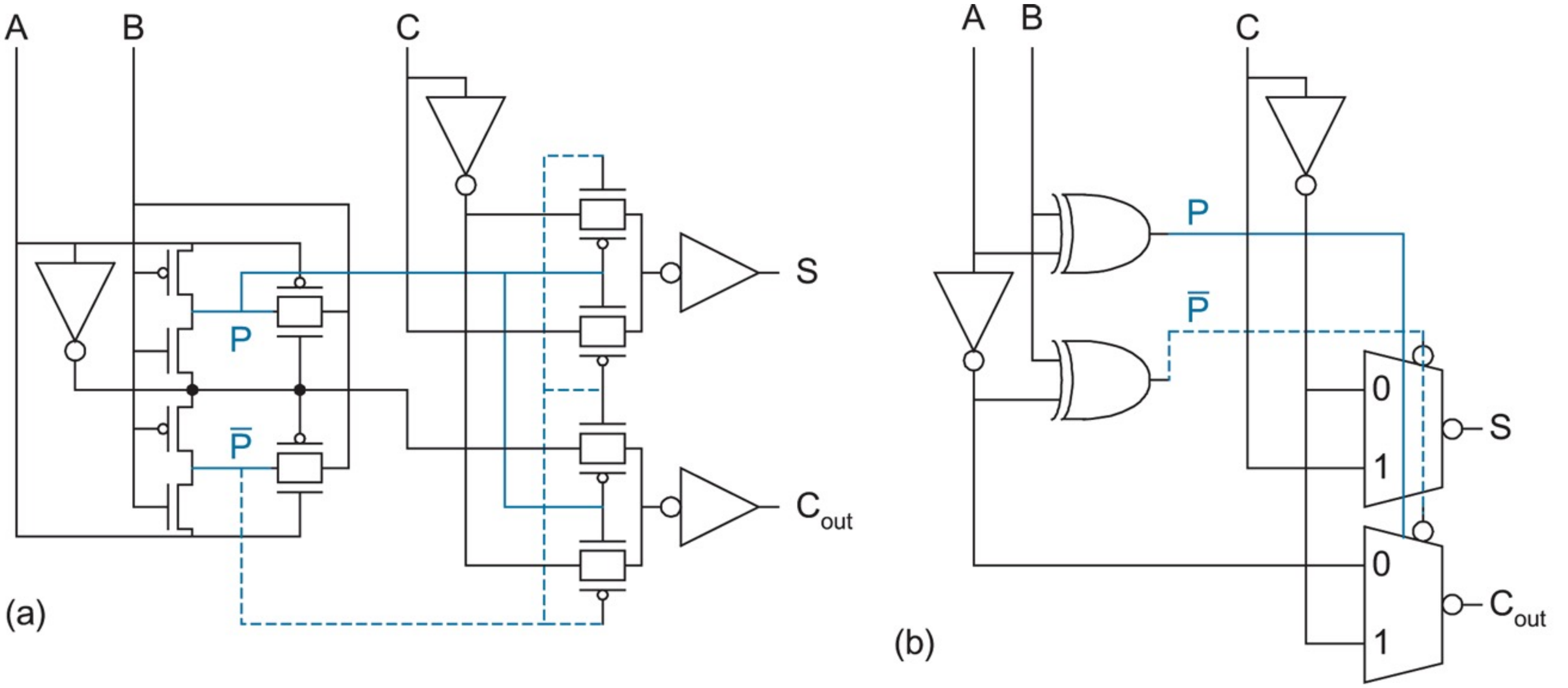
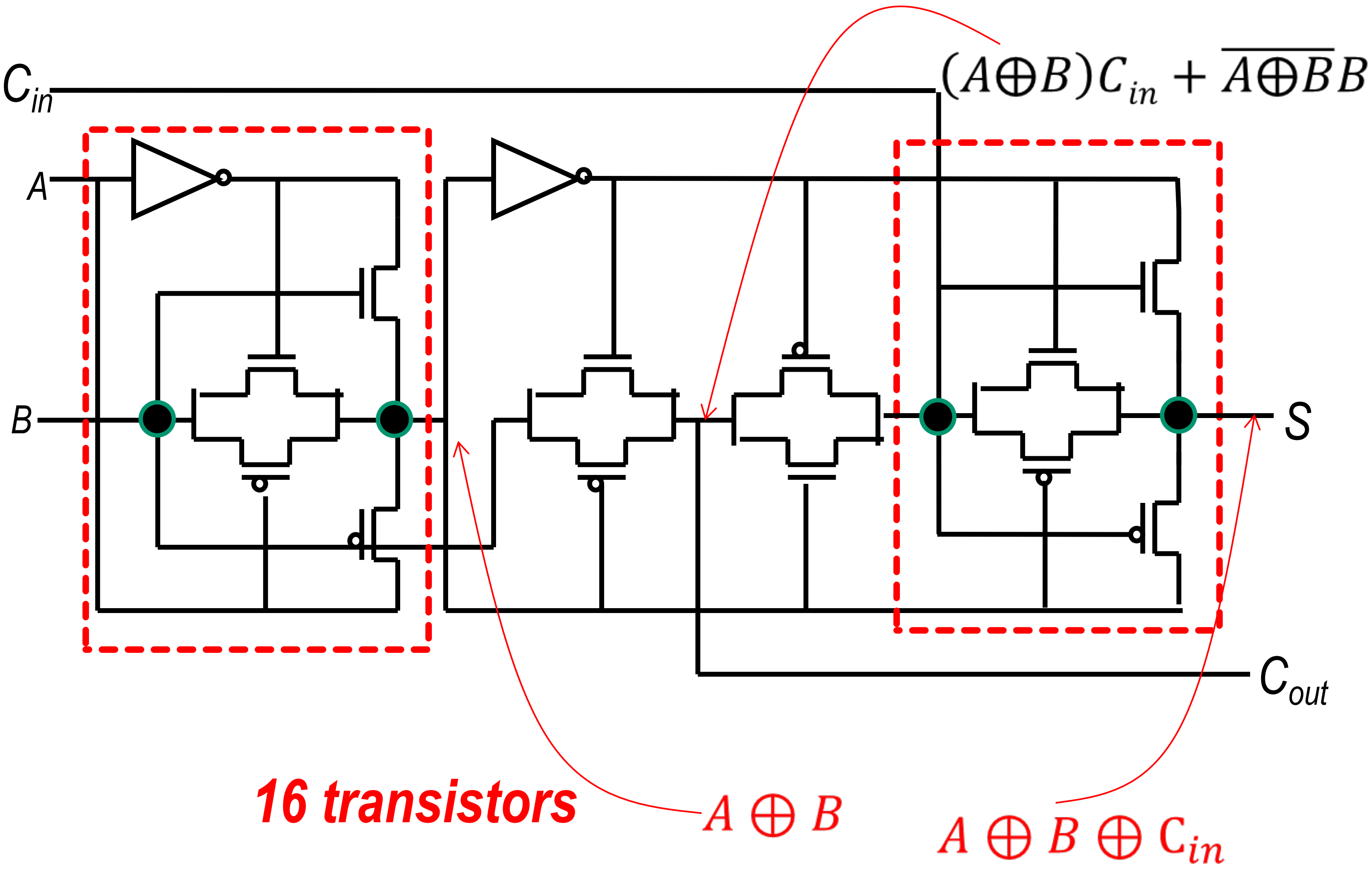


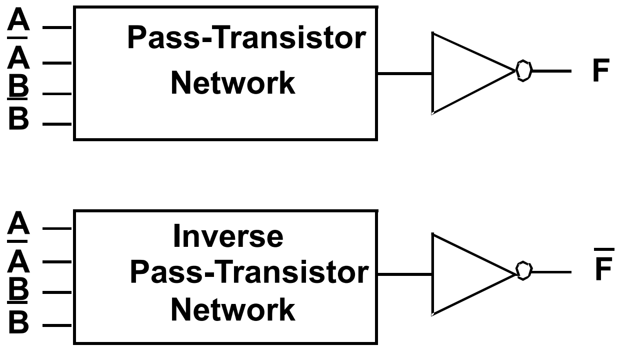
FIG 10.6 Transmission gate full adder

Review: XOR FA

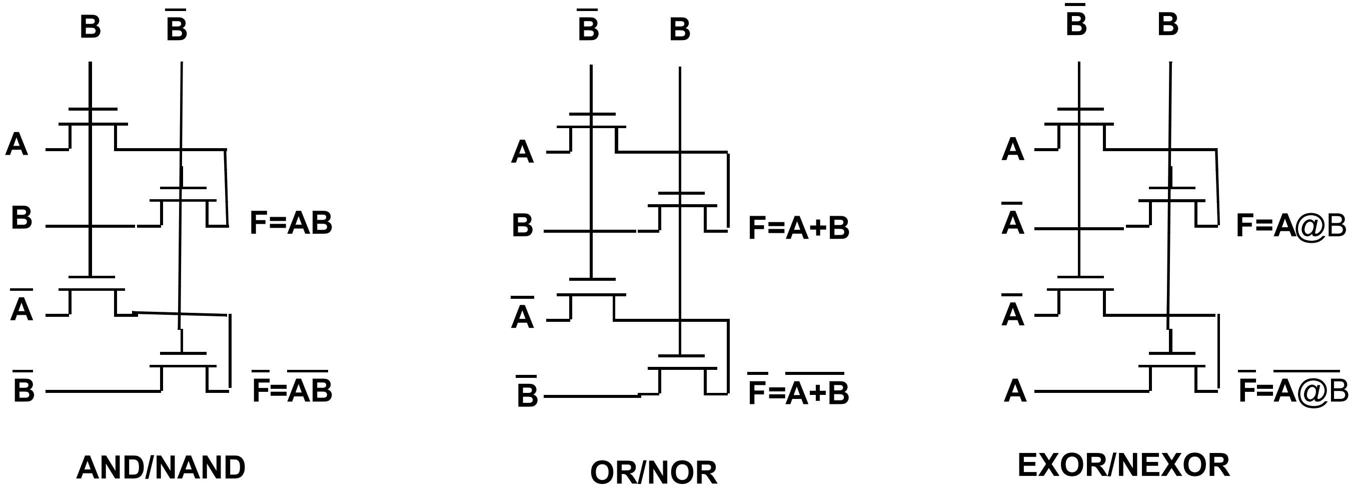

$(A \oplus B)C_{in} + \overline{A \oplus B}B$
C_{in}
A
B
S
C_{out}
$A \oplus B$
$A \oplus B \oplus C_{in}$

16 transistors

Complementary Pass Transistor Logic

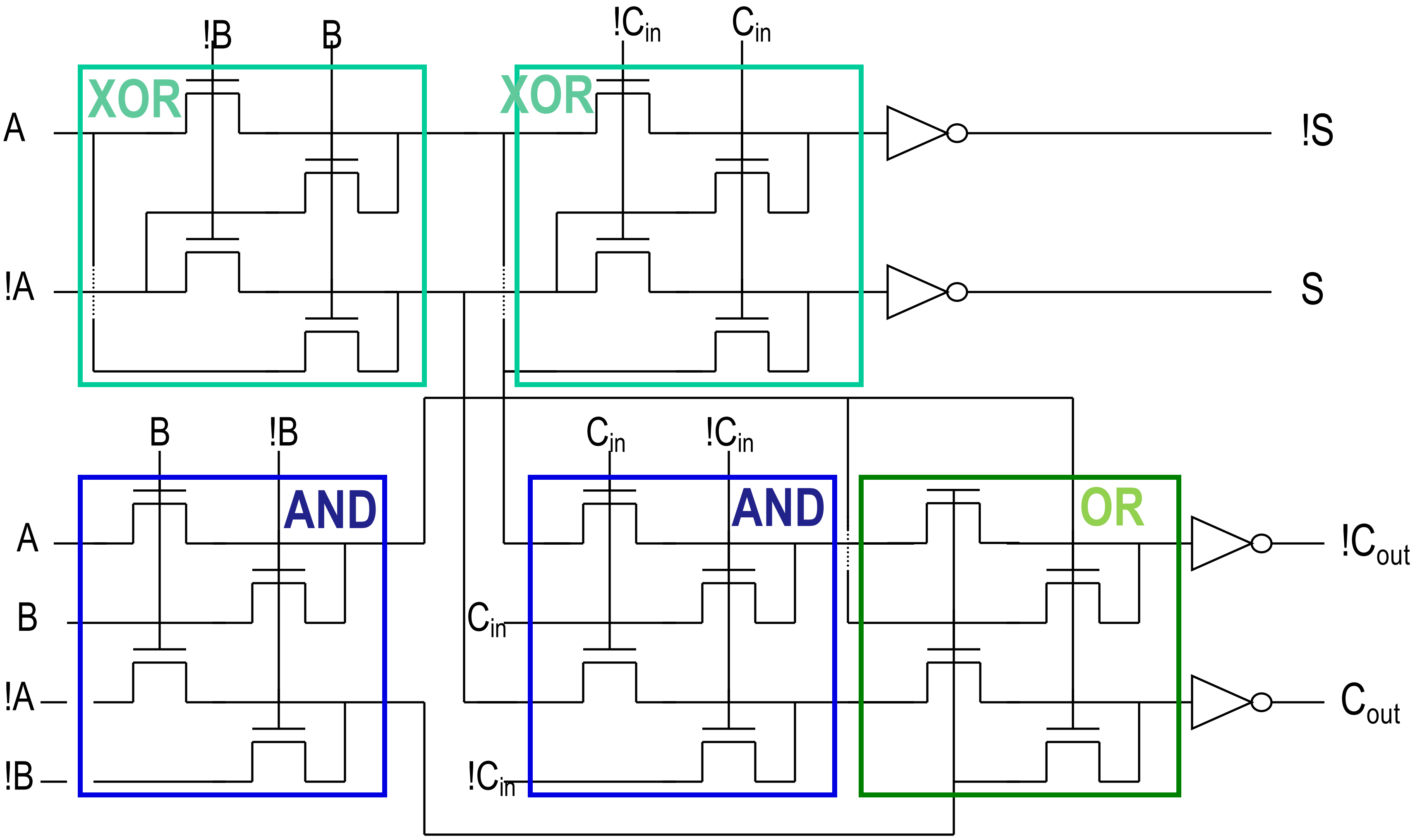


(a)



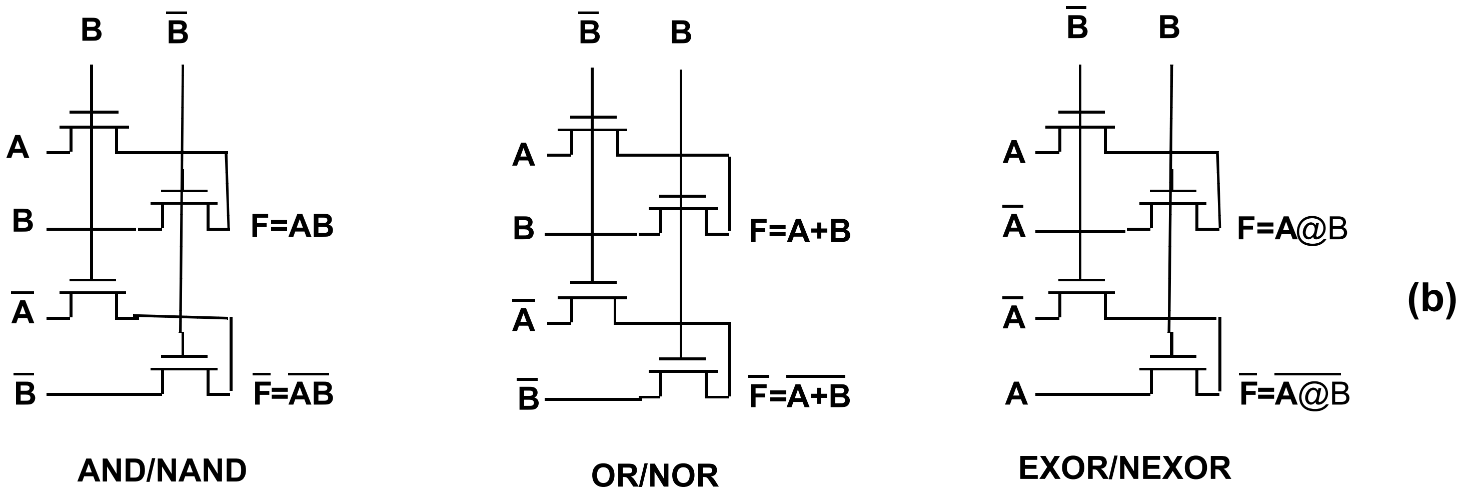
(b)

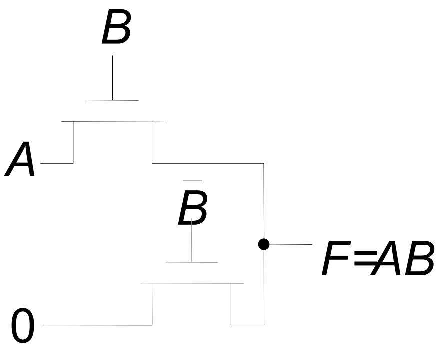
Review: CPL FA

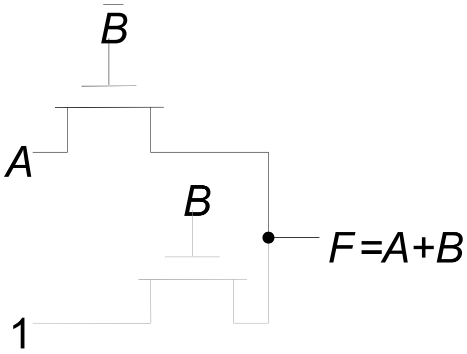


20+8 transistors, dual rail – beware of threshold drops

Complementary Pass Transistor Logic

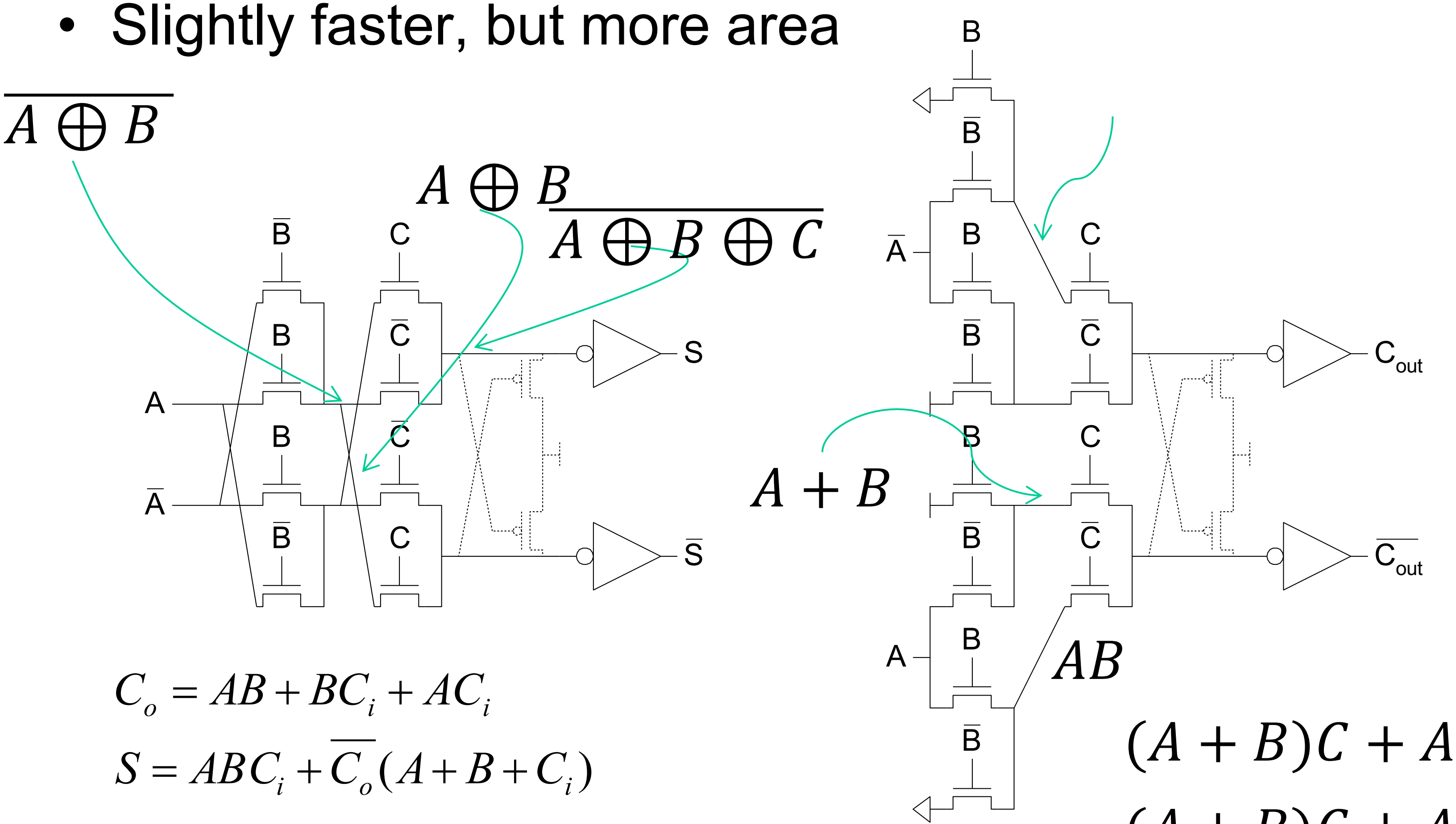






Full Adder Design III

- Complementary Pass Transistor Logic (CPL)
  - Slightly faster, but more area



$\overline{A \oplus B}$

$A \oplus B$

$\overline{A \oplus B \oplus C}$

$A + B$

AB

$C_o = AB + BC_i + AC_i$

$S = ABC_i + \overline{C_o}(A + B + C_i)$

$(A+B)C + AB\bar{C}$

$(A+B)C + AB$

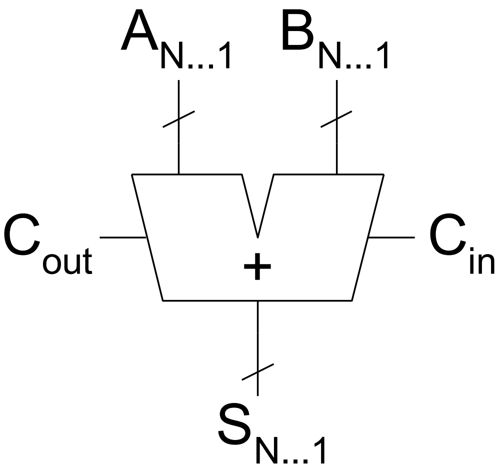
Bit to datapath

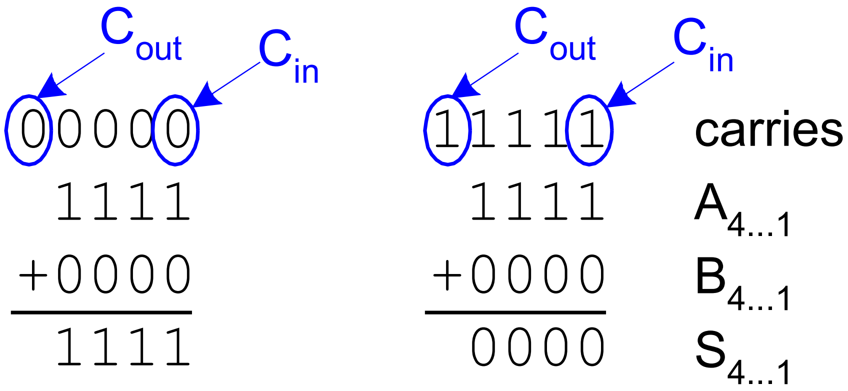
1. ***How can we use it to build a 64-bit adder?***
2. ***How can we modify it easily to build an adder/subtractor?***
3. ***How can we make it better (faster, lower power, smaller)?***

?

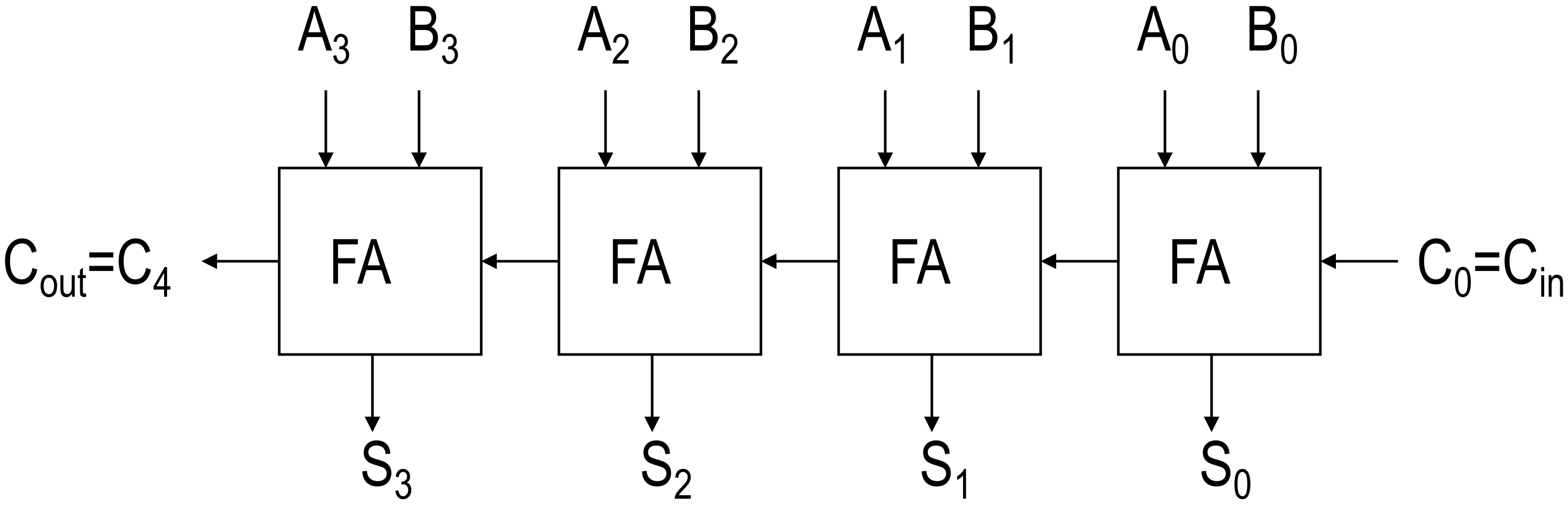
Carry Propagate Adders

- N-bit adder called CPA
  - Each sum bit depends on all previous carries
  - How do we compute all these carries quickly?





Ripple Carry Adder (RCA)



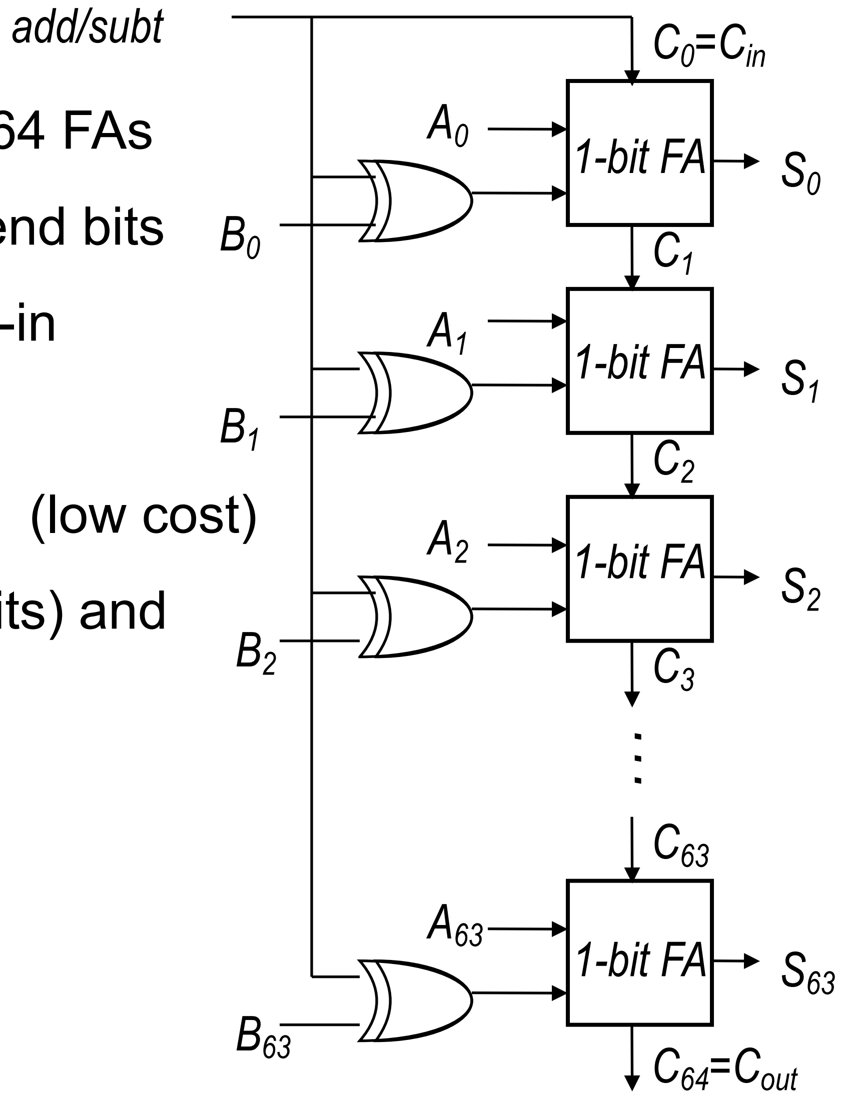
$T_{adder} \approx (N\text{-}1)\, T_{carry} + T_{sum}$

T = O(N) worst case delay

Real Goal: Make the fastest possible carry path

A 64-bit Adder/Subtractor

- Ripple Carry Adder (RCA) built out of 64 FAs
- Subtraction – complement all subtrahend bits (xor gates) and set the low order carry-in
- RCA
  - advantage: simple logic, so small (low cost)
  - disadvantage: slow (O(N) for N bits) and lots of glitching (so lots of energy consumption)



Lookahead adder

$Ci=gi+piCi-1$

$Cn=gn+pnCn-1$

$=gn+pn(gn-1+pn-1Cn-2)$

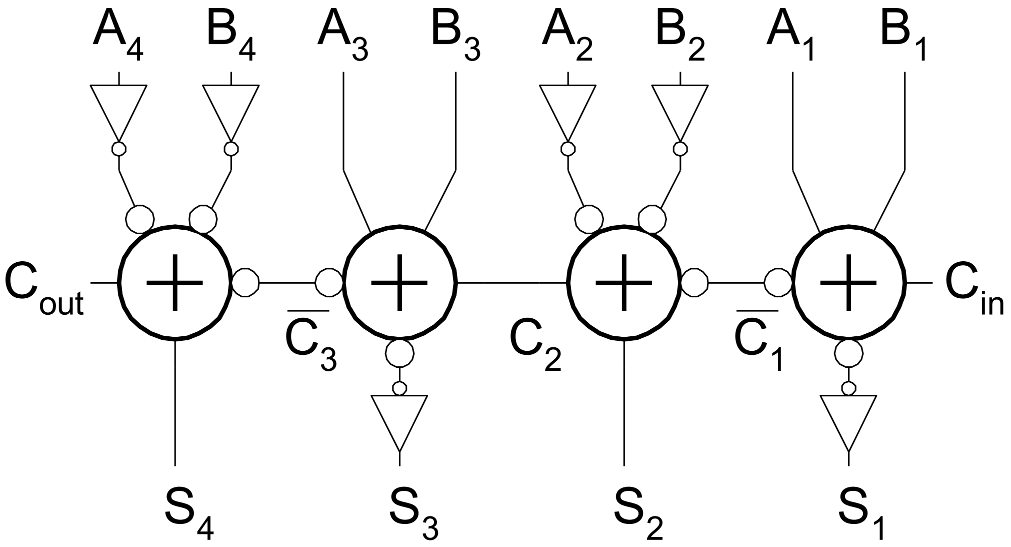
$= gn+pn(gn-1+pn-1(gn-2+pn-2Cn-3)$

$=gn+pn(gn-1+pn-1(gn-2+pn-2(gn-3+pn-3Cn-4)$

…. ….

Inversions

- Critical path passes through majority gate
  - Built from minority + inverter
  - Eliminate inverter and use inverting full adder



Outline

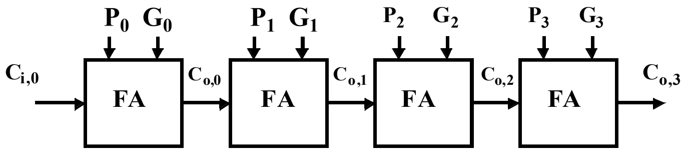
- Single-bit Addition architecture
- ***Group Definition***
- Manchester Carry Chain
- Classic adders
- Tree Adder

Why group?

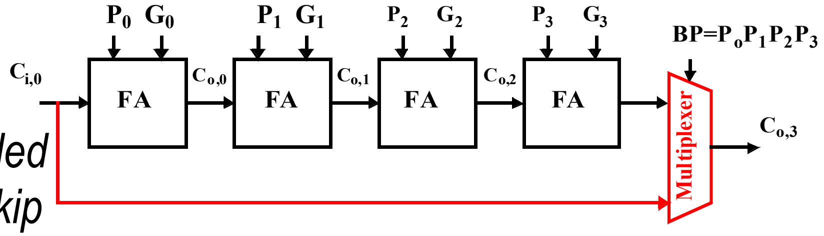
$C_i=g_i+p_iC_{i-1}$

$C_{o,3}=g_3+p_3C_2=g_3+p_3(g_2+p_2C_1)=g_3+p_3(g_2+p_2(g_1+p_1(g_0+p_0C_{i,0})))=$

$$\underbrace{g_3+p_3g_2+p_3p_2g_1+p_3p_2p_1g_0}_{\textit{Independent with carry in}}+\underbrace{p_3p_2p_1p_0C_{i,0}}_{\textit{bypass}}$$



Also called Carry-Skip



Idea: If (P0 and P1 and P2 and P3 = 1) then C_{o3} = C_0, else “kill” or “generate”.

Generate / Propagate

- Equations often factored into G and P
- Generate and propagate for **groups** spanning i:j

$$G_{i:j} = G_{i:k} + P_{i:k} G_{k-1:j}$$
$$P_{i:j} = P_{i:k} P_{k-1:j}$$

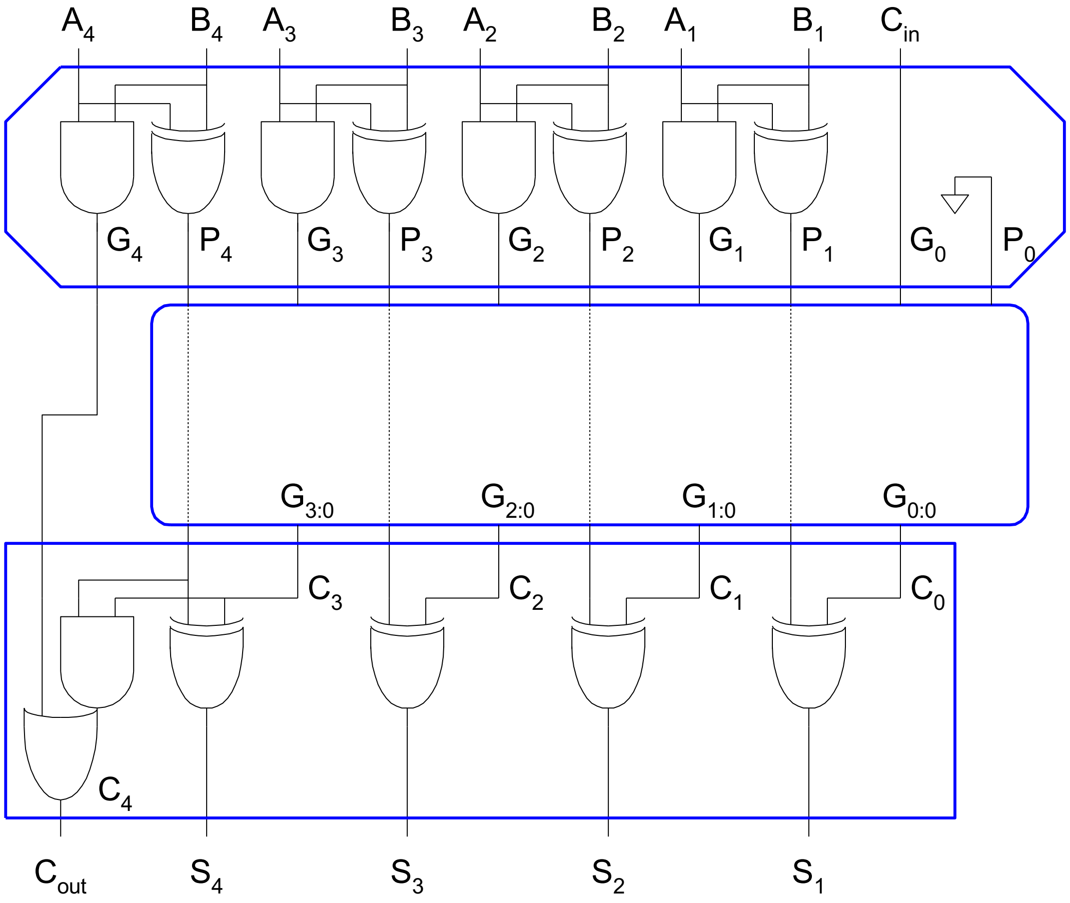
- Base case

$$G_{i:i} \equiv G_i = A_i B_i \qquad G_{0:0} \equiv C_0 = C_{in}$$
$$P_{i:i} = P_i = A_i \oplus B_i \qquad P_{0:0} = P_0 = 0$$

- Sum:

$$S_i = P_i \oplus G_{i-1:0}$$
$$C_i = G_{i:0} = G_i + P_i G_{i-1:0}$$

PG Logic



$$G_{0:0} \equiv C_0 = C_{in}$$
$$P_{0:0} = P_0 = 0$$

1: Bitwise PG logic

$$G_{i:i} \equiv G_i = A_i B_i$$
$$P_{i:i} = P_i = A_i \oplus B_i$$

2: Group PG logic

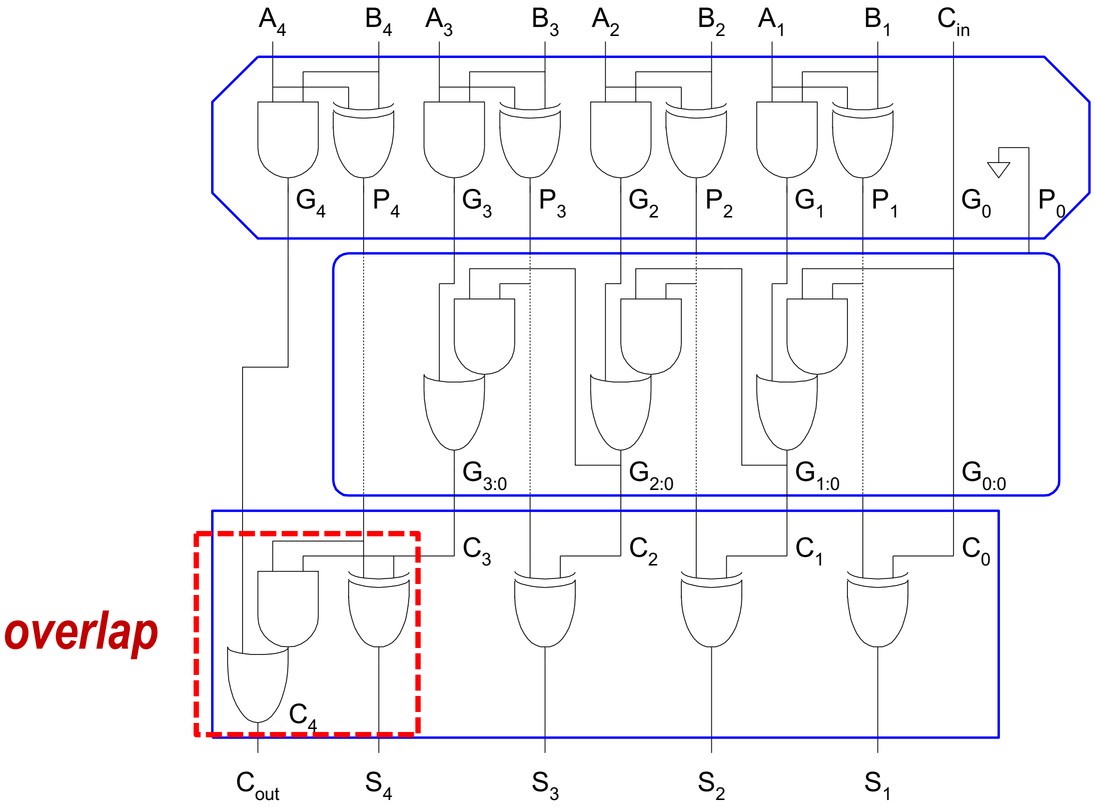
$$G_{i:j} = G_{i:k} + P_{i:k} G_{k-1:j}$$
$$P_{i:j} = P_{i:k} P_{k-1:j}$$

3: Sum logic

$$S_i = P_i \oplus G_{i-1:0}$$

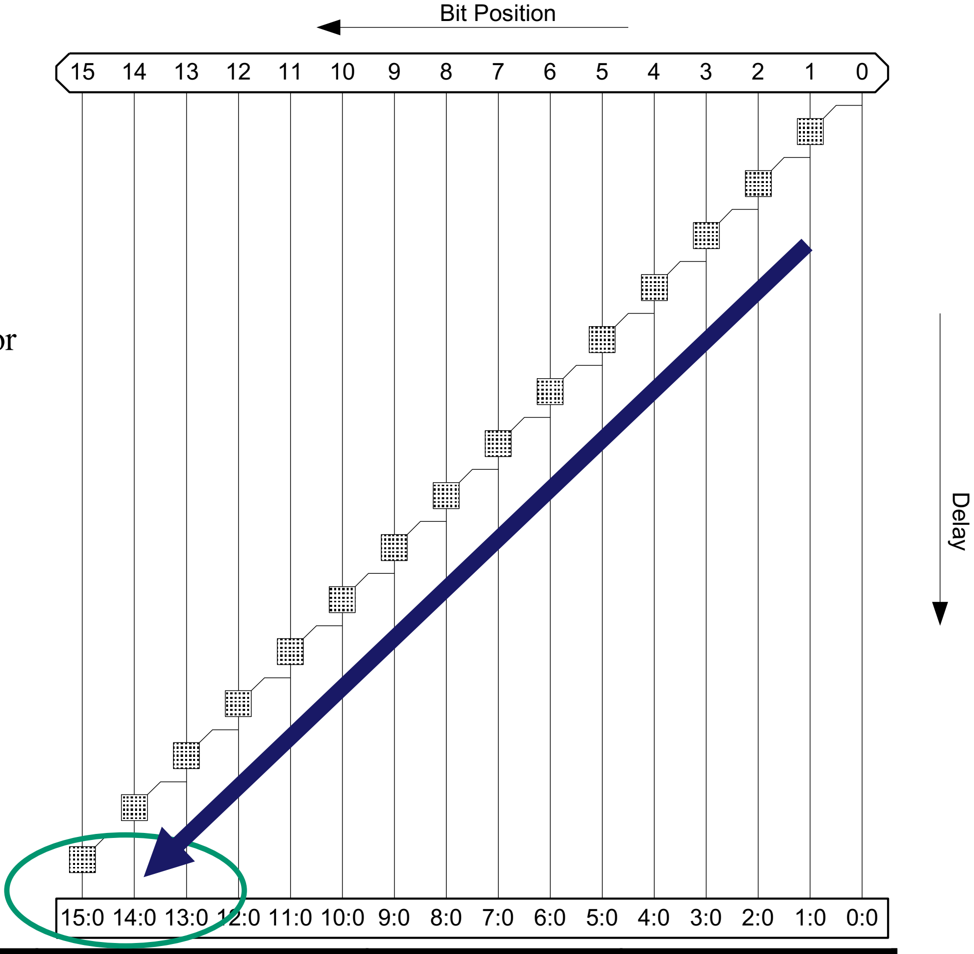
Carry-Ripple Revisited

$$C_i = G_{i:0} = G_i + P_i G_{i-1:0}$$



Carry-Ripple PG Diagram

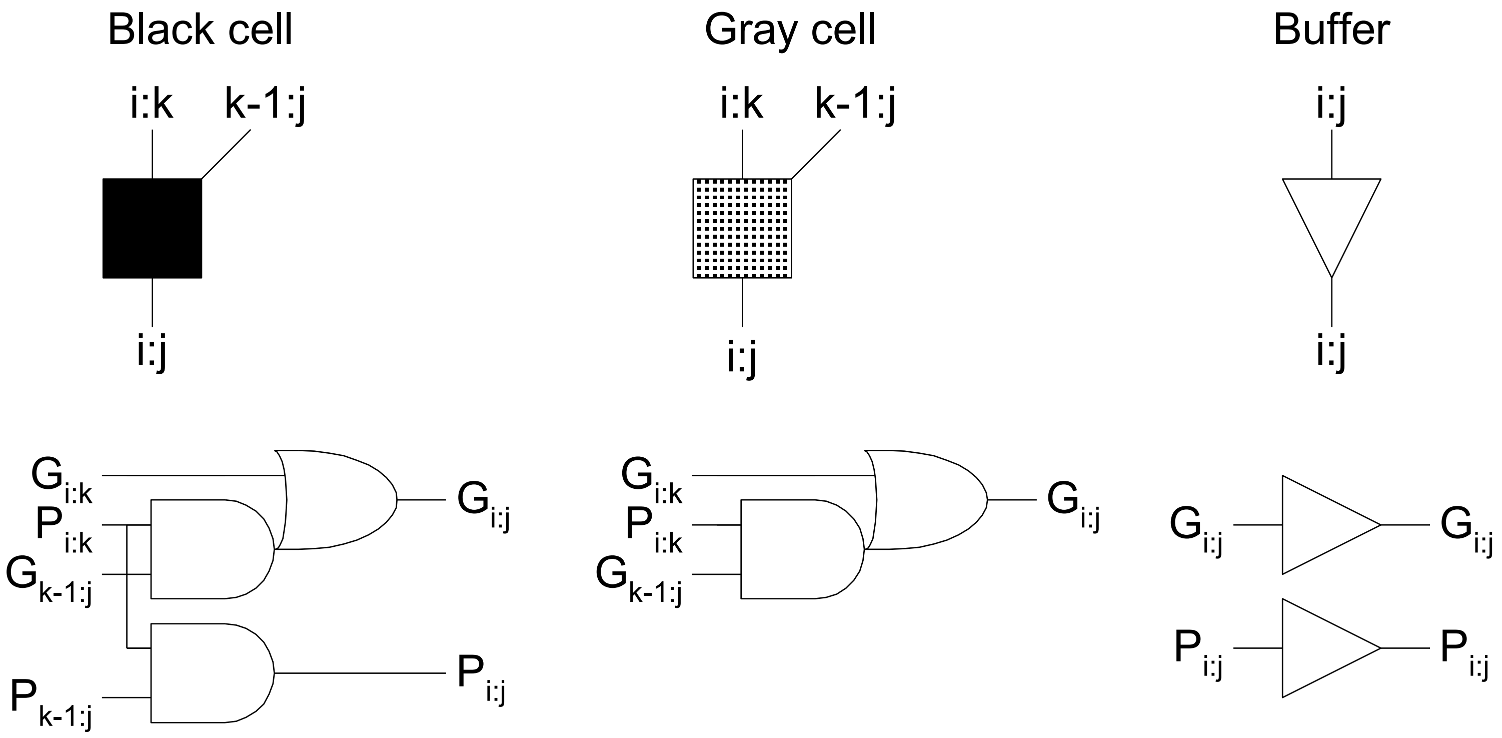
$$t_{\text{ripple}} = t_{pg} + (N-1)t_{AO} + t_{\text{xor}}$$



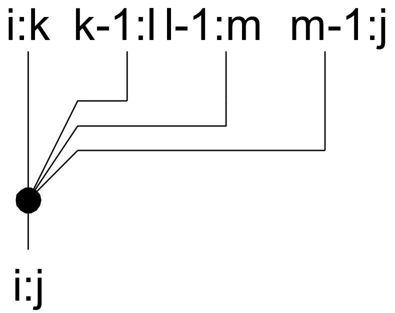
Overlap time

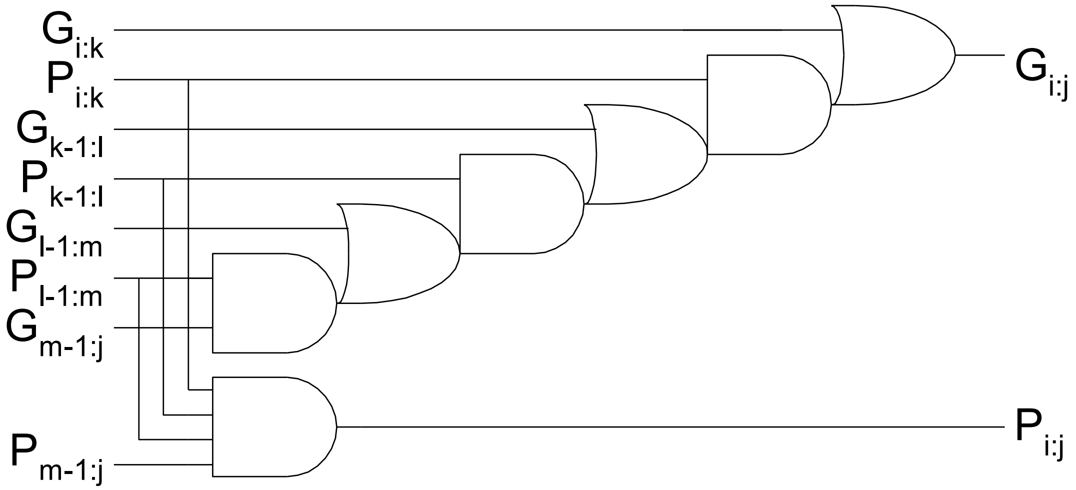
Architecture	Logic Levels	Max Fanout	Tracks	Cells
Carry-Ripple	N-1	1	1	N

PG Diagram Notation



Higher-Valency Cells





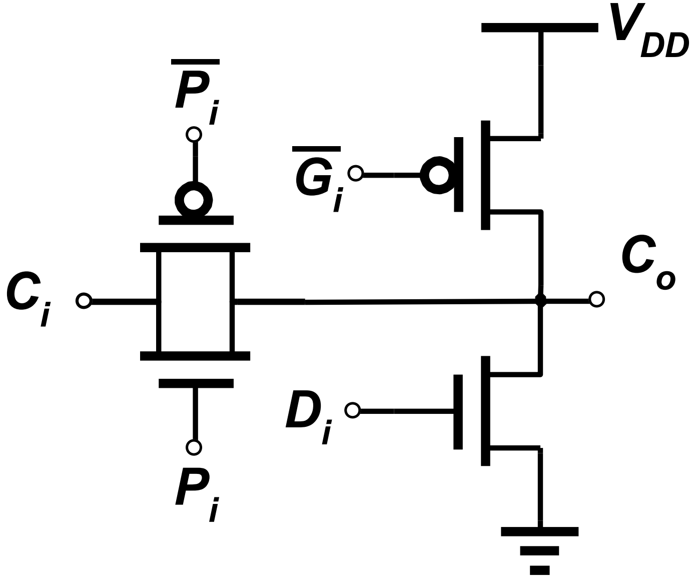
$$G_{i:j} = G_{i:k} + P_{i:k}\left(G_{k-1:l} + P_{k-1:l}\left(G_{l-1:m} + P_{l-1:m}G_{m-1:j}\right)\right)$$
$$= G_{i:k} + P_{i:k}G_{k-1:l} + P_{i:k}P_{k-1:l}G_{l-1:m} + P_{i:k}P_{k-1:l}P_{l-1:m}G_{m-1:j}$$

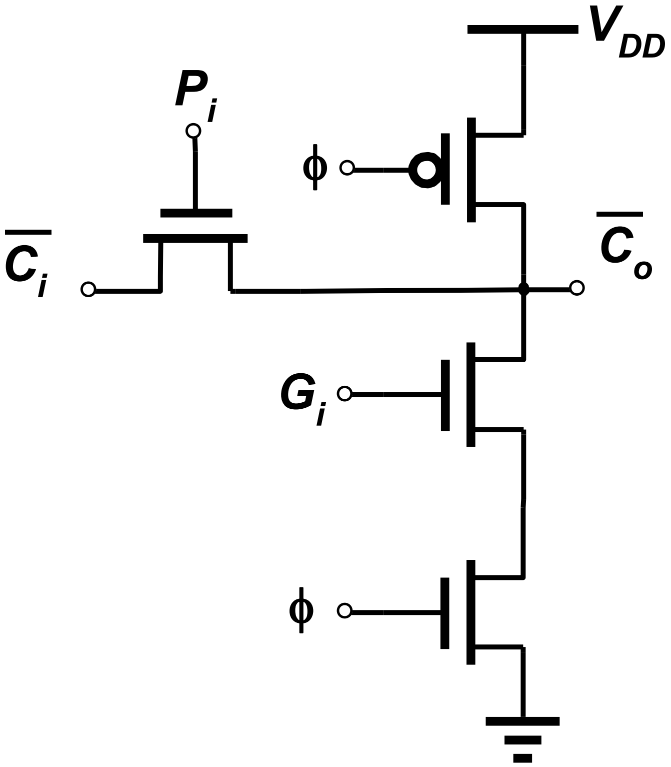
Outline

- Single-bit Addition architecture
- Group Definition
- ***Manchester Carry Chain***
- Classic adders
- Tree Adder

Manchester Carry Chain

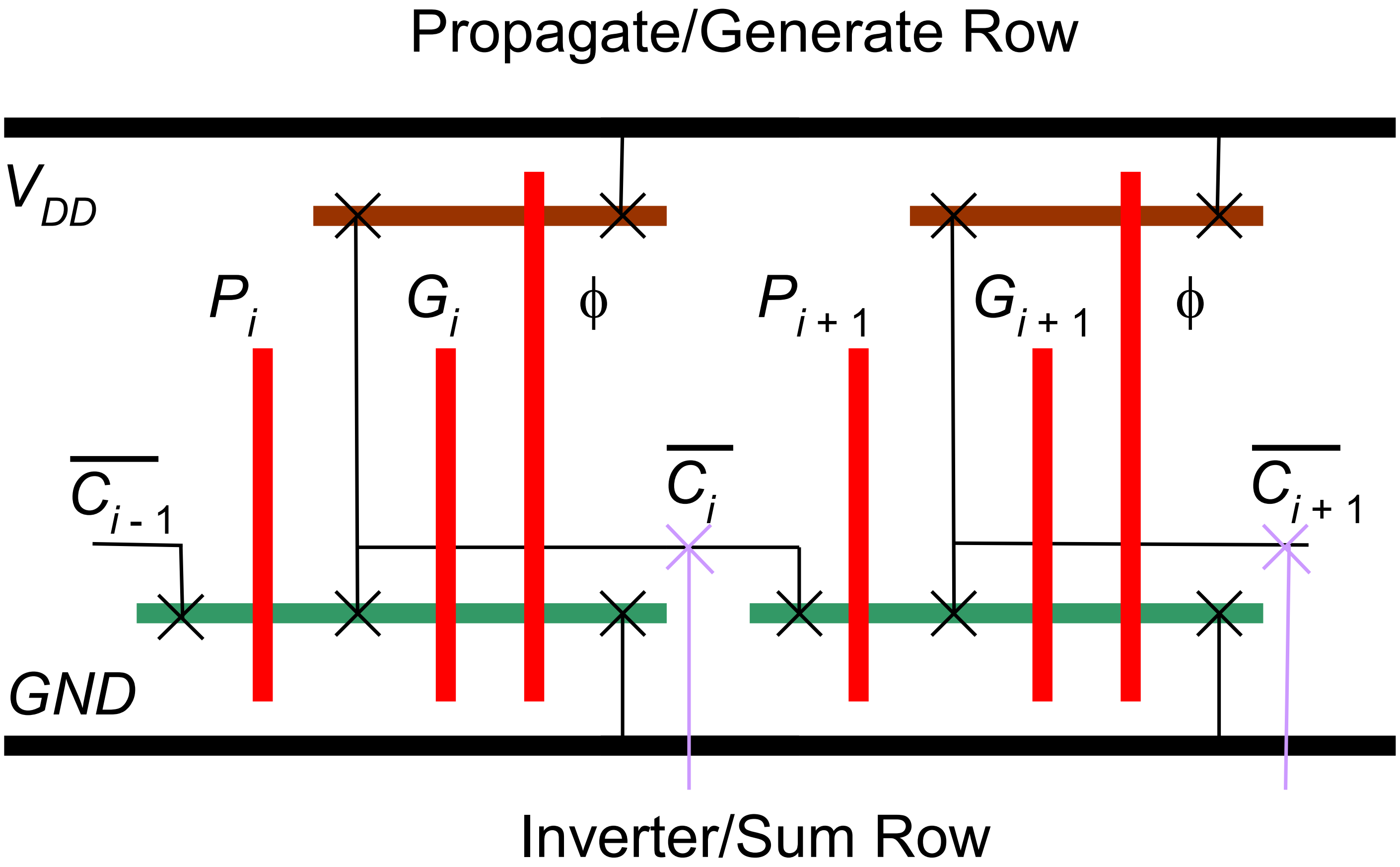
Only one line valid





$$C_{out} = G + PC$$

MCC Stick Diagram



Manchester Carry Chain

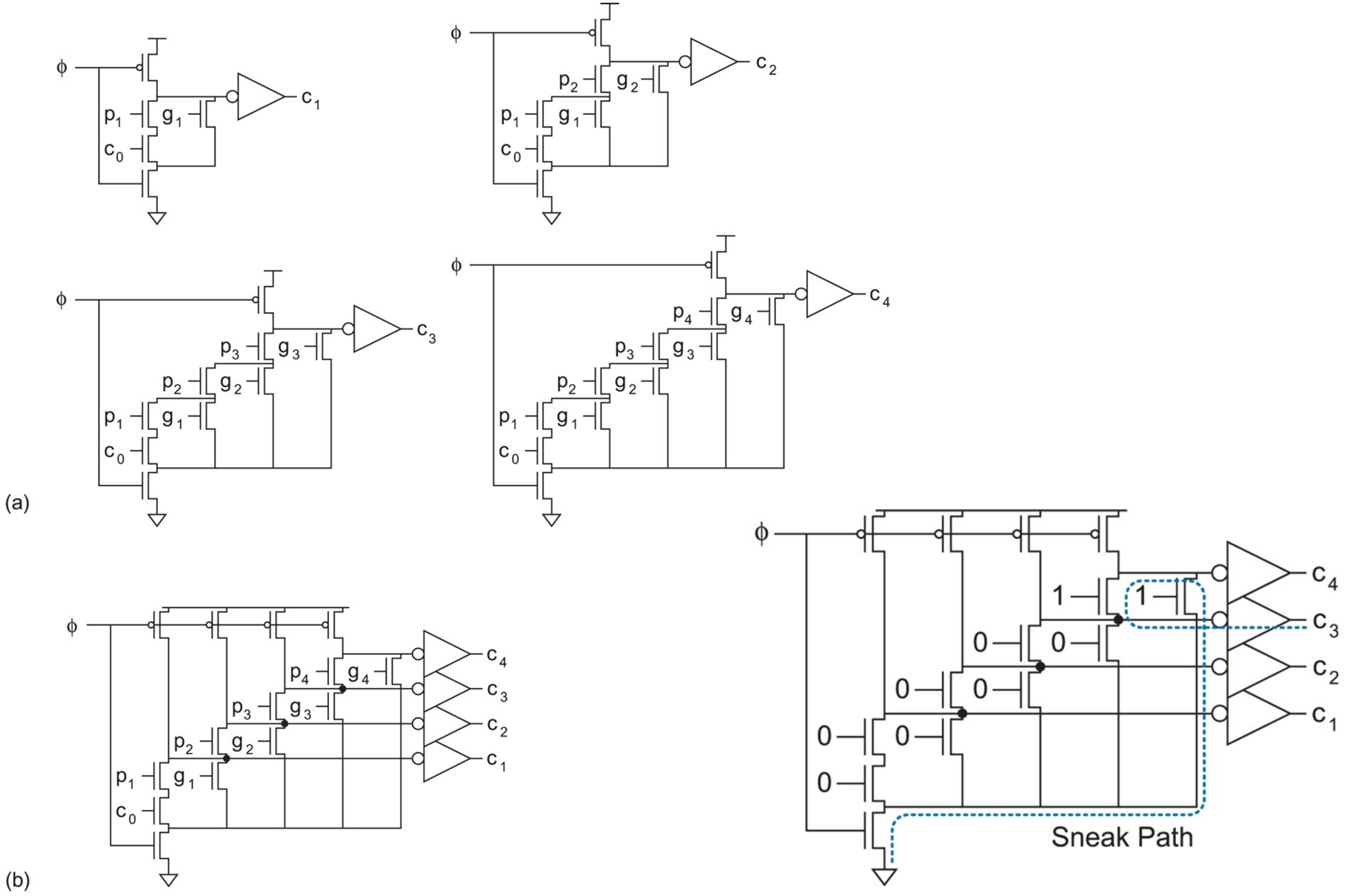
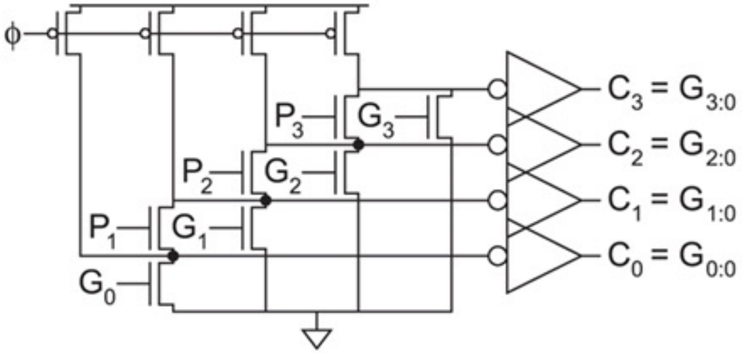
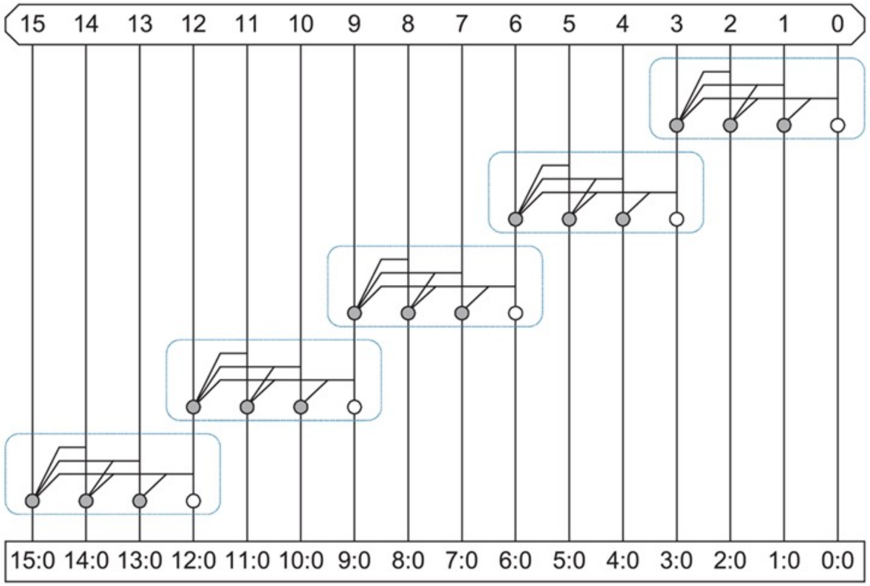


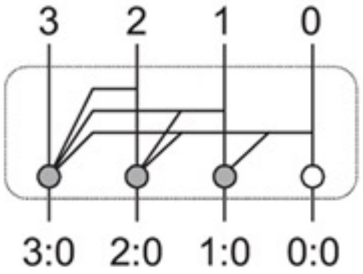
FIG 6.44 Conventional and MODL carry chains

FIG 6.45 Sneak path

Manchester Carry Chain


ϕ
P_3
G_3
P_2
G_2
P_1
G_1
G_0
$C_3 = G_{3:0}$
$C_2 = G_{2:0}$
$C_1 = G_{1:0}$
$C_0 = G_{0:0}$


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
15:0 14:0 13:0 12:0 11:0 10:0 9:0 8:0 7:0 6:0 5:0 4:0 3:0 2:0 1:0 0:0


3 2 1 0
3:0 2:0 1:0 0:0