Short Ques
Logical Ques

CSE-2103

Computer Architecture

Lecture – 1, 2

Text Books

- ***Computer Architecture and Organization***
  Hayes J.P., McGraw-Hill.

- ***Computer organization and design: The hardware/software interface***
  Patterson D.A., Hennessy J.L., Morgan Kaufmann.

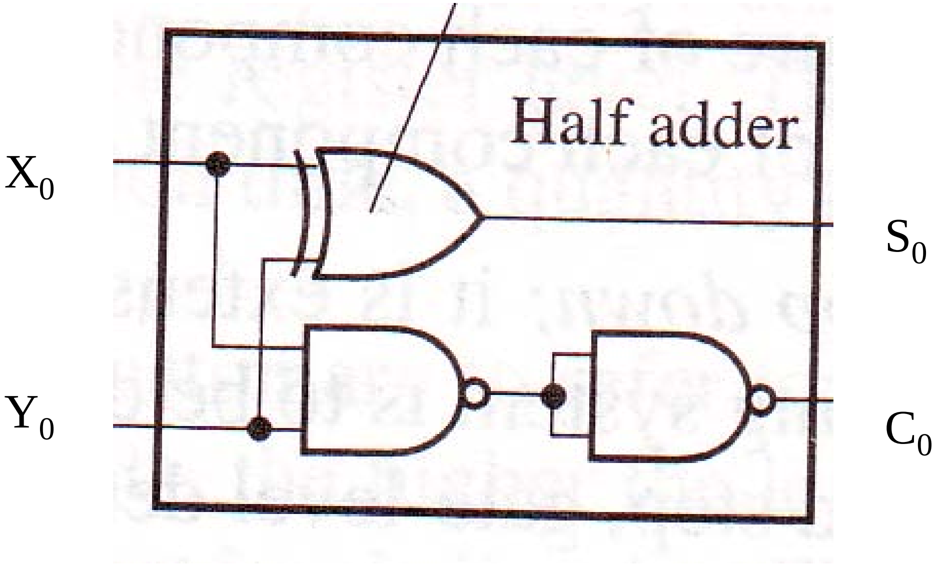
Reference Book

- ***Computer Architecture: A Quantitative Approach***
  Patterson D.A., Hennessy J.L., Morgan Kaufmann.

Fixed-Point Arithmetic

- Addition
- Subtraction
- Multiplication
- Division

Half Adder

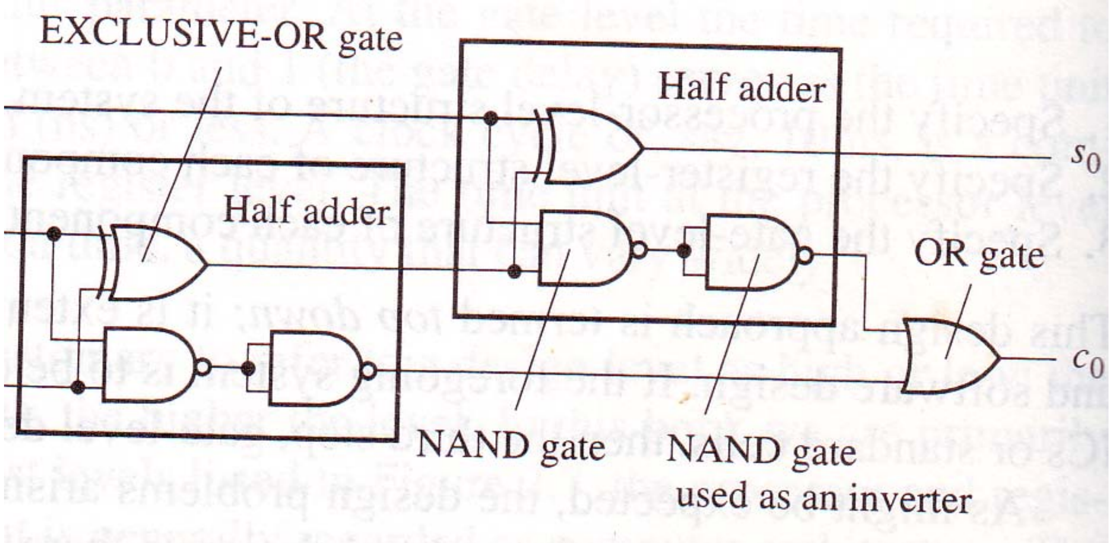


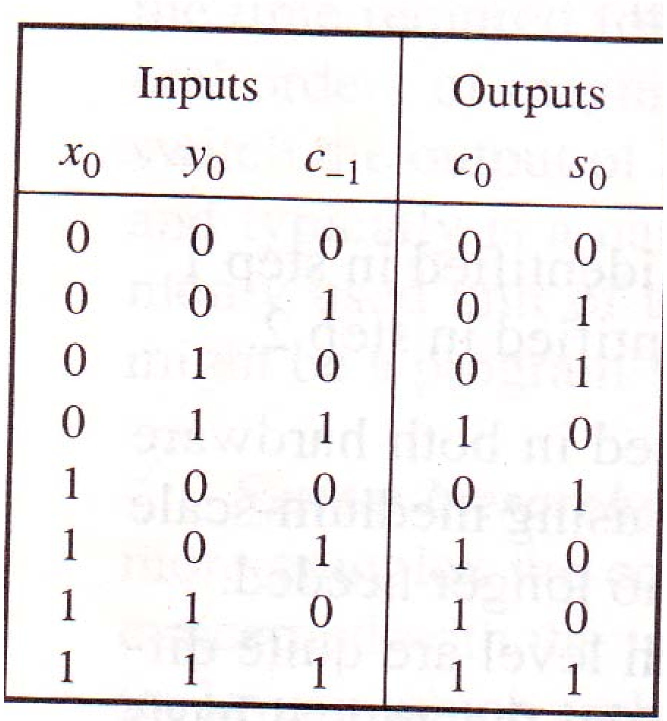
X_0 Y_0	S_0 C_0
0 0	0 0
0 1	1 0
1 0	1 0
1 1	1 1

$$S_0 = X_0 \oplus Y_0$$

$$C_0 = X_0 Y_0$$

Full Adder



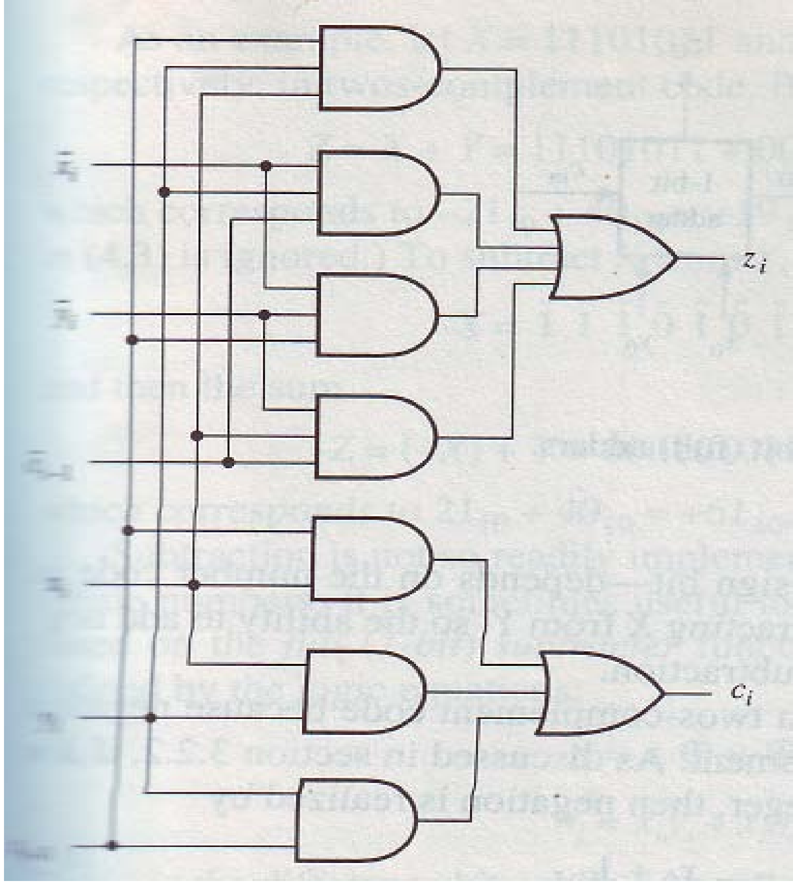


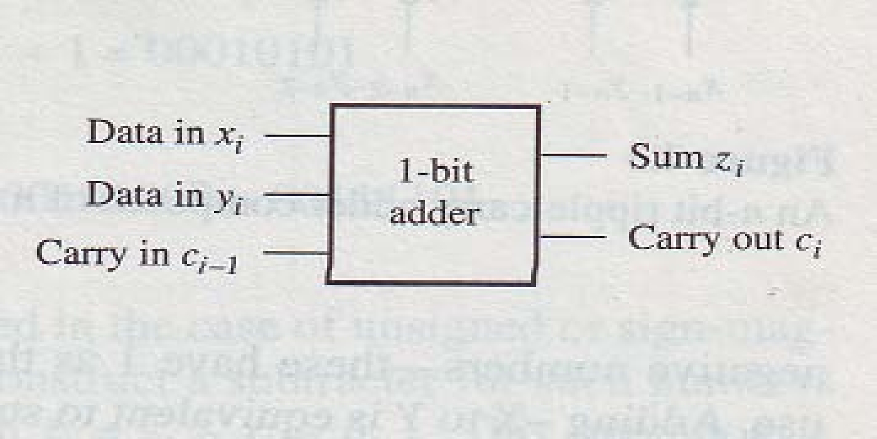
Inputs			Outputs	
x_0	y_0	c_{-1}	c_0	s_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S_0 = X_0 \oplus Y_0 \oplus C_{-1}$$

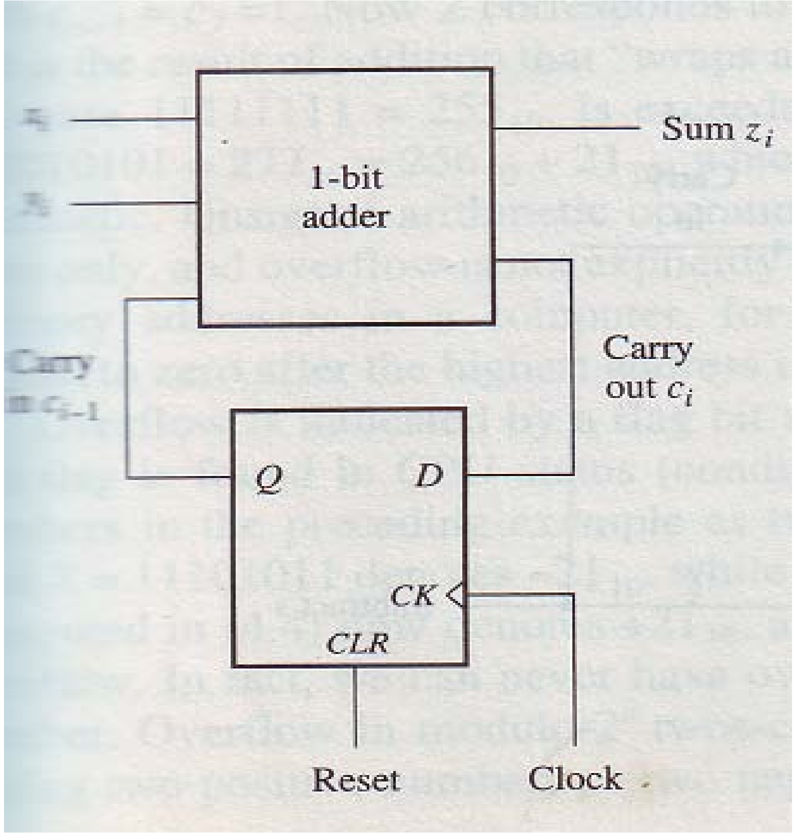
$$C_0 = X_0Y_0 + X_0C_{-1} + Y_0C_{-1}$$

Full Adder


$\bar{x}_i$
$\bar{y}_i$
$\bar{c}_{i-1}$
x_i
y_i
c_{i-1}
z_i
c_i

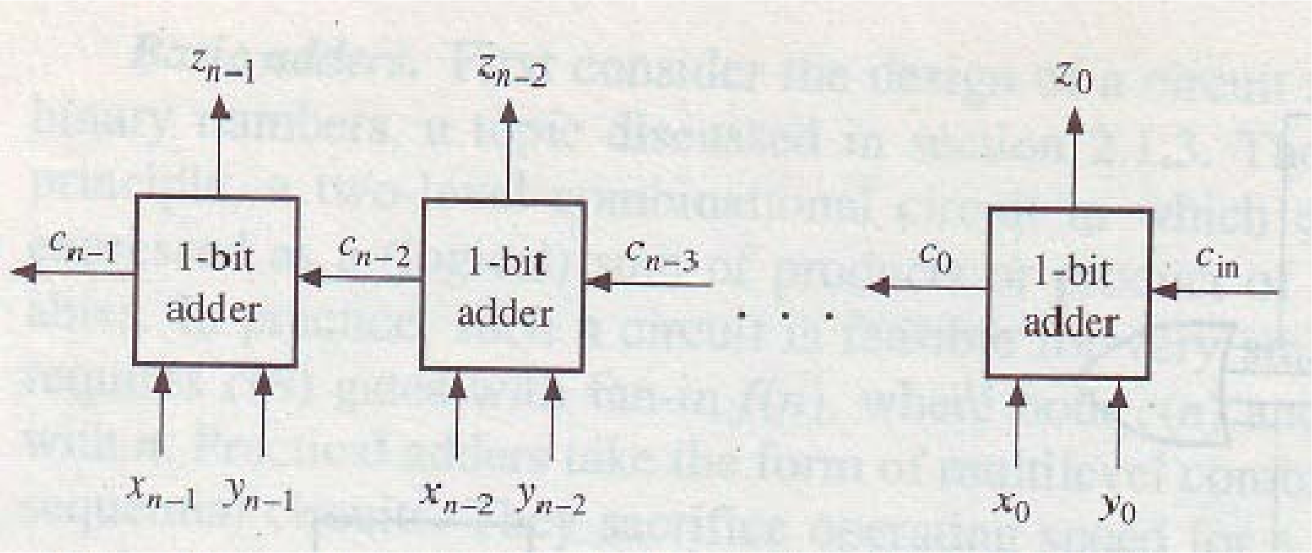

Data in x_i
Data in y_i
Carry in c_{i-1}
1-bit adder
Sum z_i
Carry out c_i

Serial Binary Adder



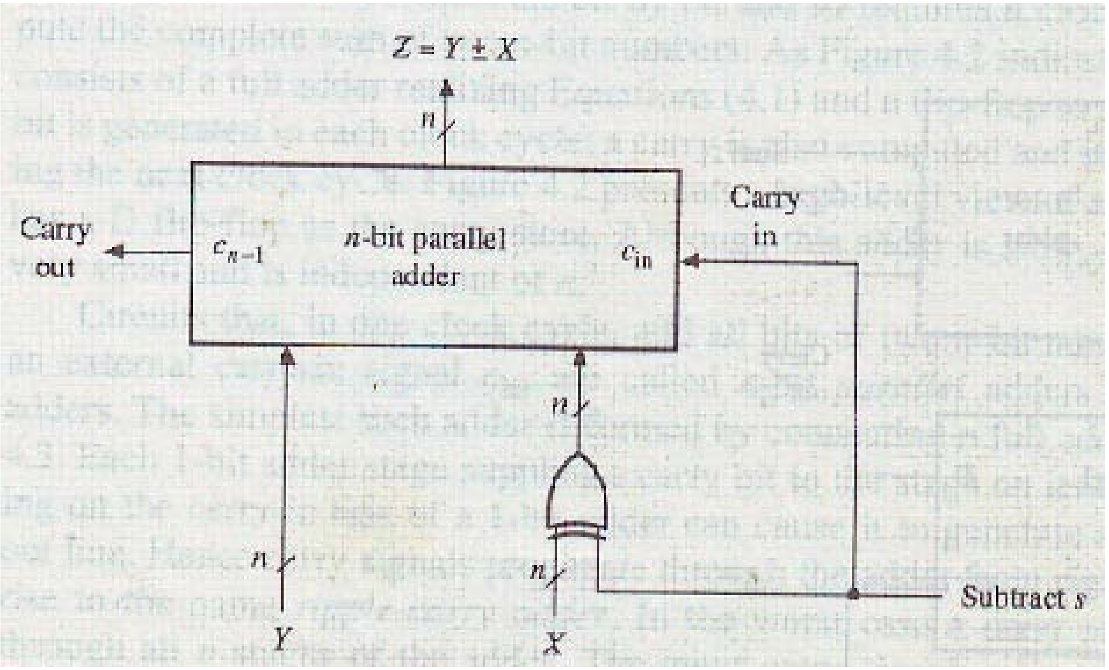
- Least expensive circuit in terms of hardware cost.
- It adds the numbers bit by bit and so requires n clock cycle to compute the sum of two n-bit numbers.
- Circuit size independent of n.

Ripple Carry Adder



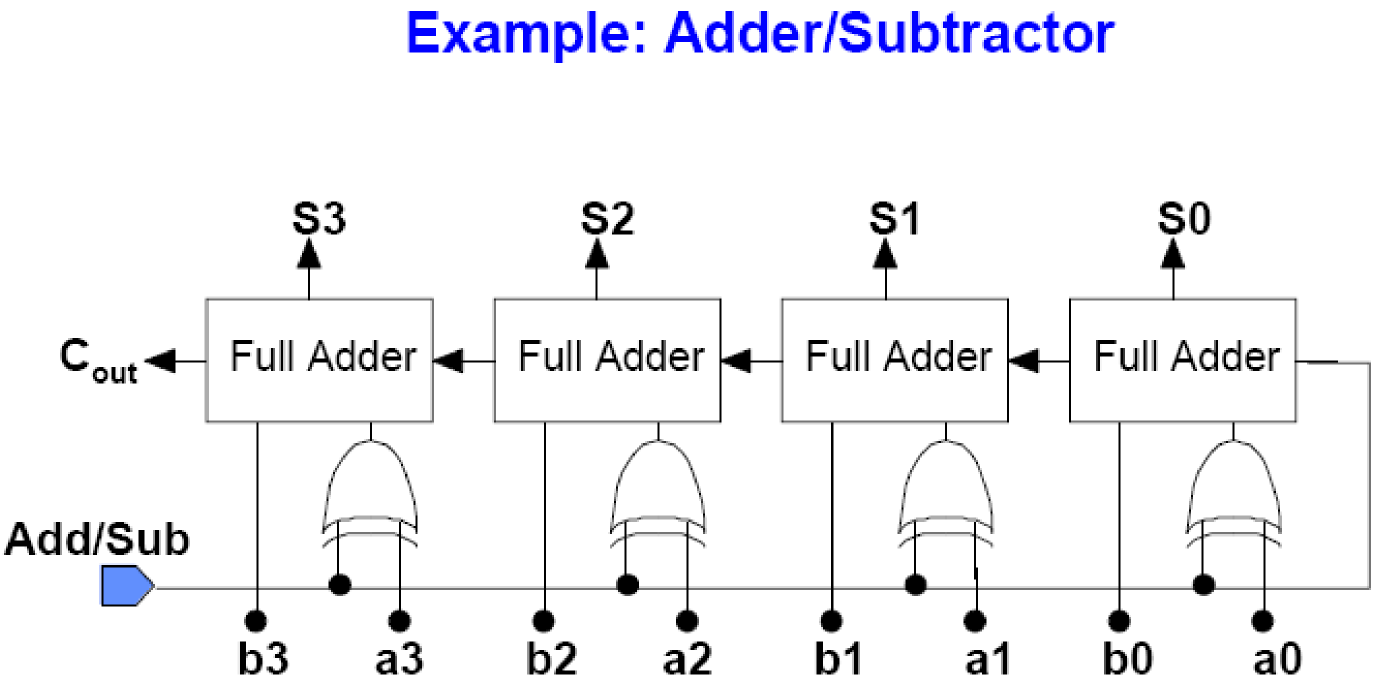
- A 1 appearing on the carry in line of a 1-bit adder cause it to generate a 1 on its carry out line. So, the carry signal propagate through the adder from right to left.
- The maximum signal propagation delay is nd, where d is the delay of a full-adder stage.
- The amount of hardware increase linearly with n.

2's Complement Adder-Subtracter



- When s =0, then X ⊕ s = X
- When s=1, then X ⊕ s = $\overline{X}$

2’s Complement Adder-Subtracter



Overflow

- When the result of an arithmetic operation exceeds the standard word size n, overflow occurs.

- Example: let n=8, $X=11101011=235_{10}$ and $Y=00101010=42_{10}$
  $Z = X+Y = 11101011 + 0101010 = 00010101 = 21_{10}$
  with $C_{n-1} = C_7 = 1$.
  $C_7Z = 100010101 = 277_{10} = 256_{10} + 21_{10}$

- The result of an addition simply wraps around when the largest number 2^n-1 is exceeds.

- For n, the number range for unsigned number is 0 to 2^n-1

Overflow

- We can never have overflow on adding a negative and positive number.

- Example: let n=8 X=11101011=-21$_{10}$ and Y=00101010=+42$_{10}$

  Z = X+Y = 00010101 = 21$_{10}$ and C_7 = 1.

  So, C_{n-1} = 1 does not indicate overflow.

- Overflow in 2's complement addition can result from adding

  1) two positive numbers or

  2) two negative numbers.

Overflow

Case 1: Two numbers are positive.

Let n = 4, 7 + 3 = 0111+0011 = 1010 so, c_{n-2} = 1

C_{n-2} =1 indicates that the magnitude of the sum exceeds the n-1 bits allocated to it.

Case 2: Two numbers are negative.

Let n=4, -7 = 1001, -3 = 1101

so, 1001+1101 = 10110 so, c_{n-2} = 0

C_{n-2} =0 indicates the overflow.

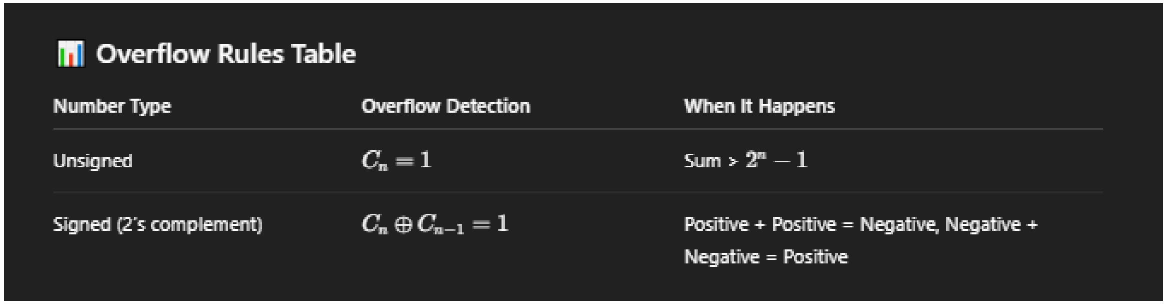
Overflow

$$Z_{n-1}Z_{n-2}.....Z_0 := X_{n-1}X_{n-2}.....X_0 + Y_{n-1}Y_{n-2}.....Y_0$$

$$v = \overline{X_{n-1}}\,\overline{Y_{n-1}}C_{n-2} + X_{n-1}Y_{n-1}\overline{C_{n-2}}$$

$$v = C_{n-1} \oplus C_{n-2}$$

X_{n-1}	Y_{n-1}	C_{n-2}	Z_{n-1}	v
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0



📊 Overflow Rules Table

Number Type	Overflow Detection	When It Happens
Unsigned	$C_n = 1$	Sum > $2^n - 1$
Signed (2's complement)	$C_n \oplus C_{n-1} = 1$	Positive + Positive = Negative, Negative + Negative = Positive

Carry-Lookahead Adder

- It reduce the time required to form carry signals.

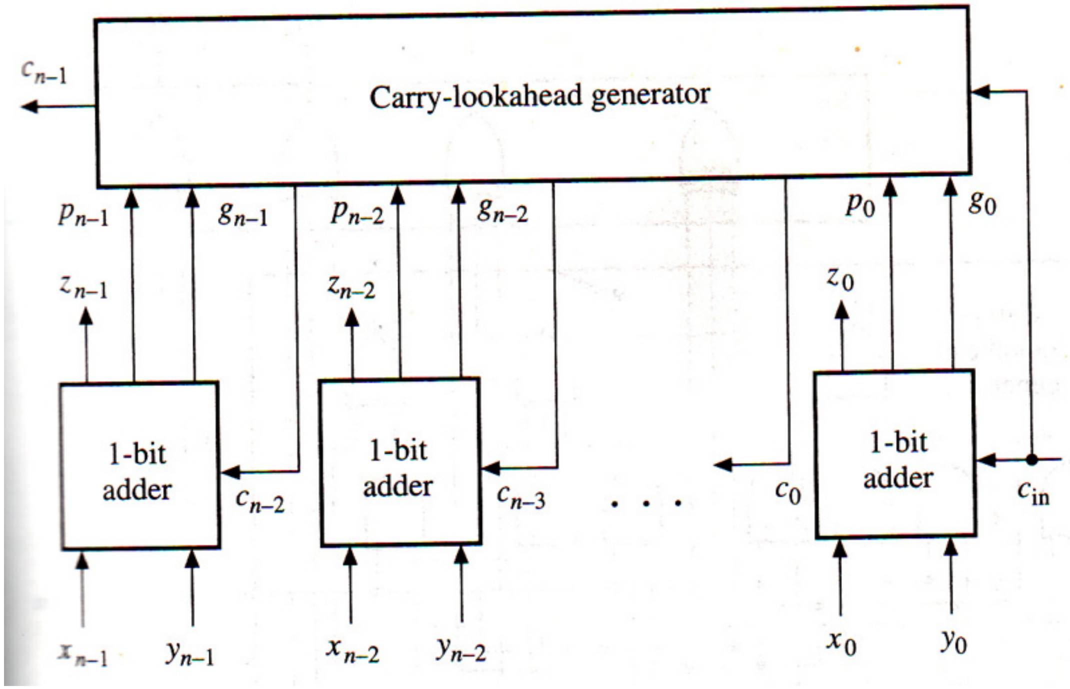
- It computes the input carry needed b y stage I directly from carrylike signals obtained from all the preceding stages i-1, i-2, ….., 0, rather than waiting for normal carries to ripple slowly from stage to stage.

- Adders that use this principle are called carry-lookahead adders.

Carry-Lookahead Adder

- Two signals:
  generate signal, $g_i = x_i y_i$
  propagate signal, $p_i = x_i + y_i$

- $c_i = x_i y_i + x_i c_{i-1} + y_i c_{i-1}$
- $c_i = g_i + p_i c_{i-1}$
- $c_{i-1} = g_{i-1} + p_{i-1} c_{i-2}$
- $c_i = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-2}$



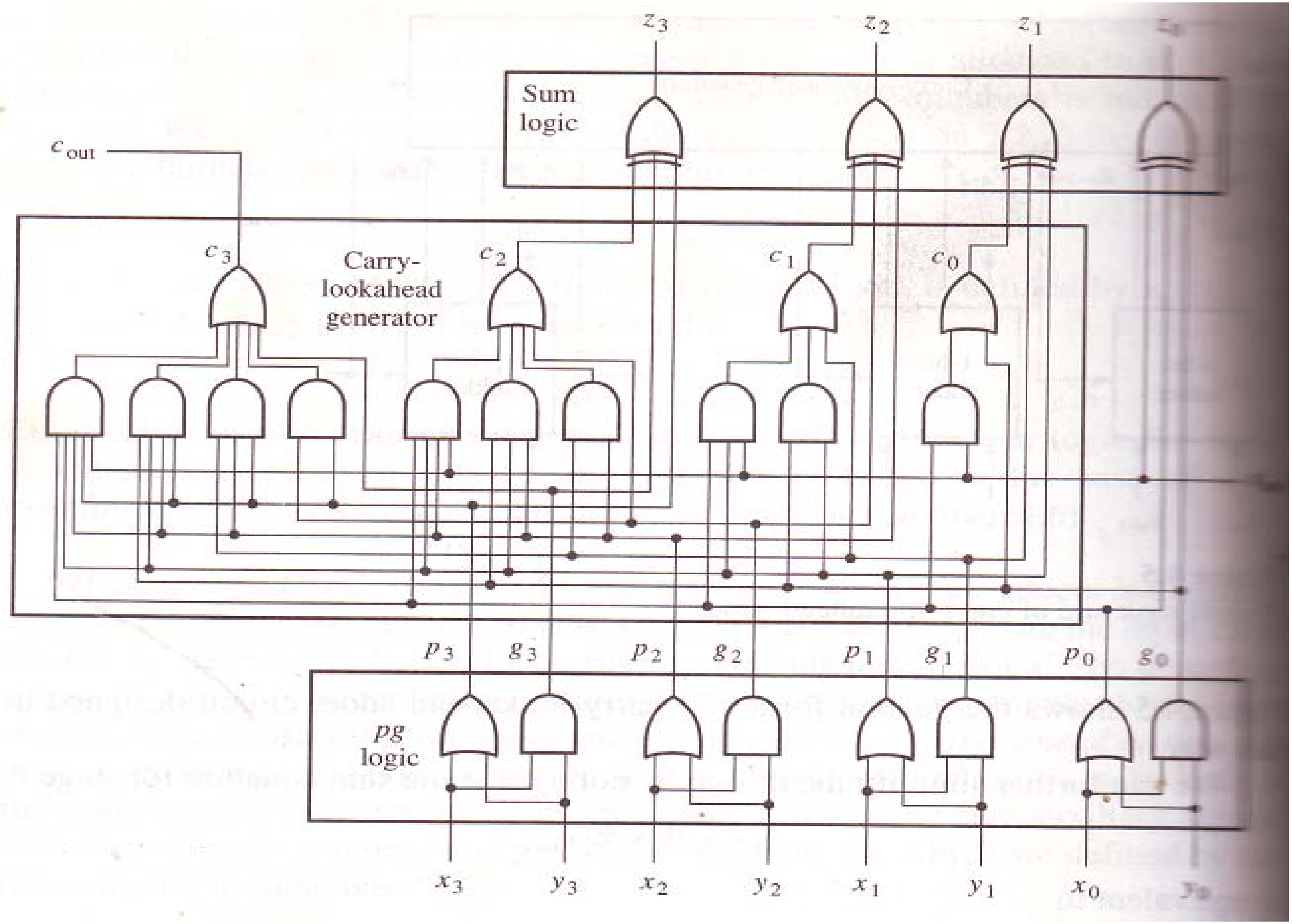
4-bit Carry-Lookahead Adder

- $c_i = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-2}$

- $c_0 = g_0 + p_0 c_{in}$
  $c_1 = g_1 + p_1 g_0 + p_1 p_0 c_{in}$
  $c_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{in}$
  $c_3 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_{in}$

- $z_i = x_i \oplus y_i \oplus c_{i-1}$ can be written as $z_i = p_i \oplus g_i \oplus c_{i-1}$

4-bit Carry-Lookahead Adder

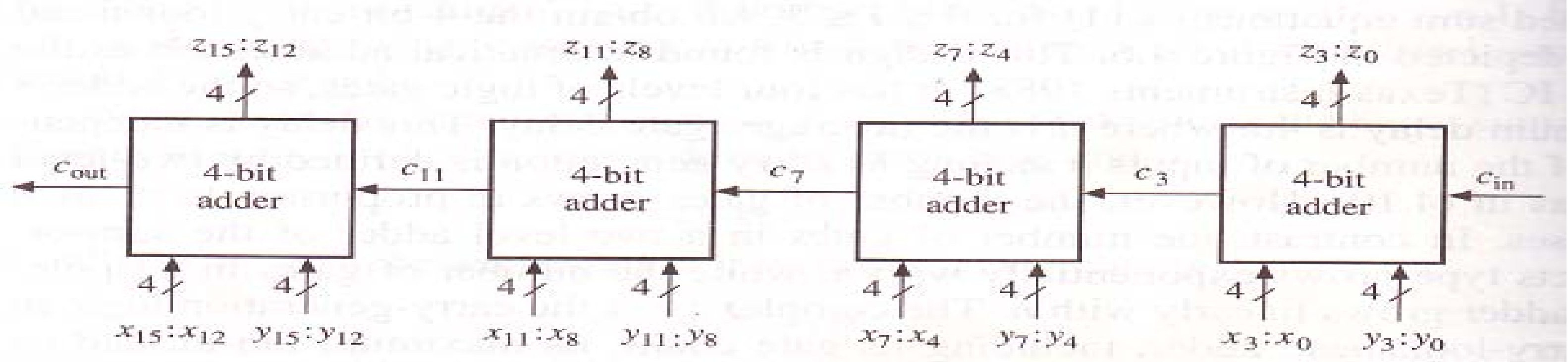


4-bit Carry-Lookahead Adder

- Maximum delay is 4d, where d is the average gate delay. It is independent of number of input n.
- The number of gates grows in proportion to n^2 as n increases.
- The complexity of the carry generation logic in the carry lookahead adder, including its gate count, its maximum fan-in, and its maximum fan-out, increase steadily with n.
- It limits n to 4.

Adder Expansion

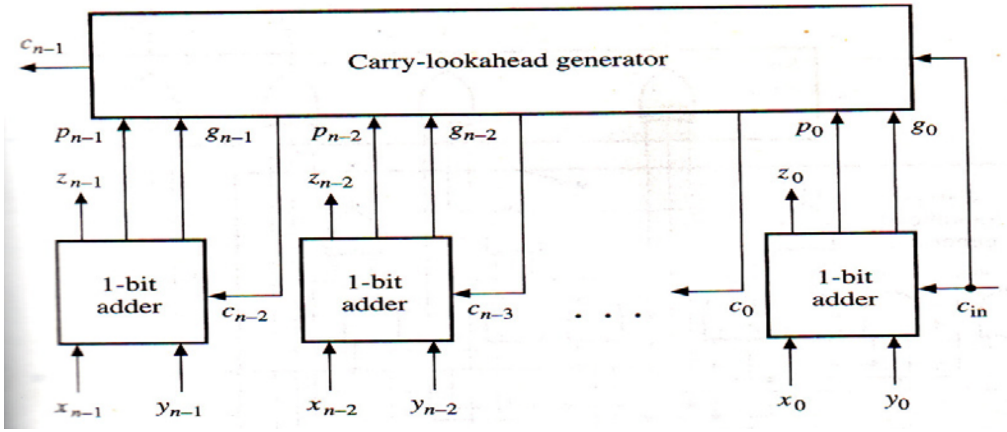
- If we replace n 1-bit adder stages in the n-bit ripple carry adder with n k-bit adders, we obtain an nk-bit adder.



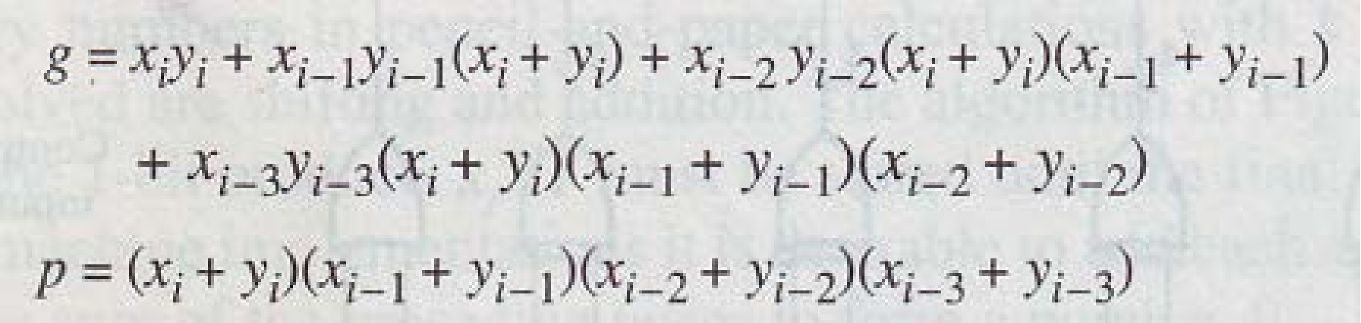
16-bit adder composed of 4-bit adders linked by ripple-carry propagation

Adder Expansion

- If we replace n 1-bit adder stages in the n-bit carry look-ahead adder with n k-bit adders, we obtain an nk-bit adder.

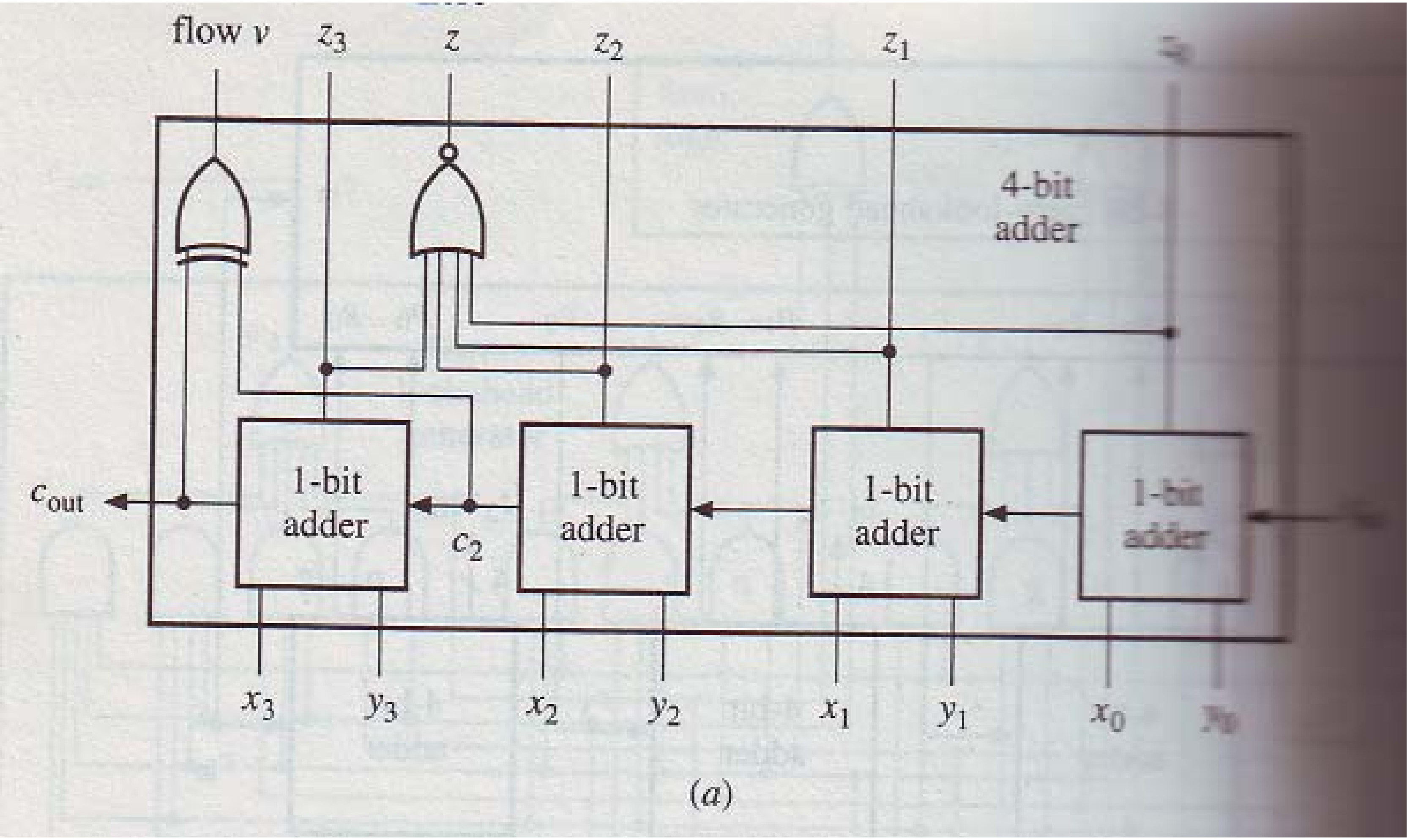


16-bit adder composed of 4-bit adders linked by carry look-ahead

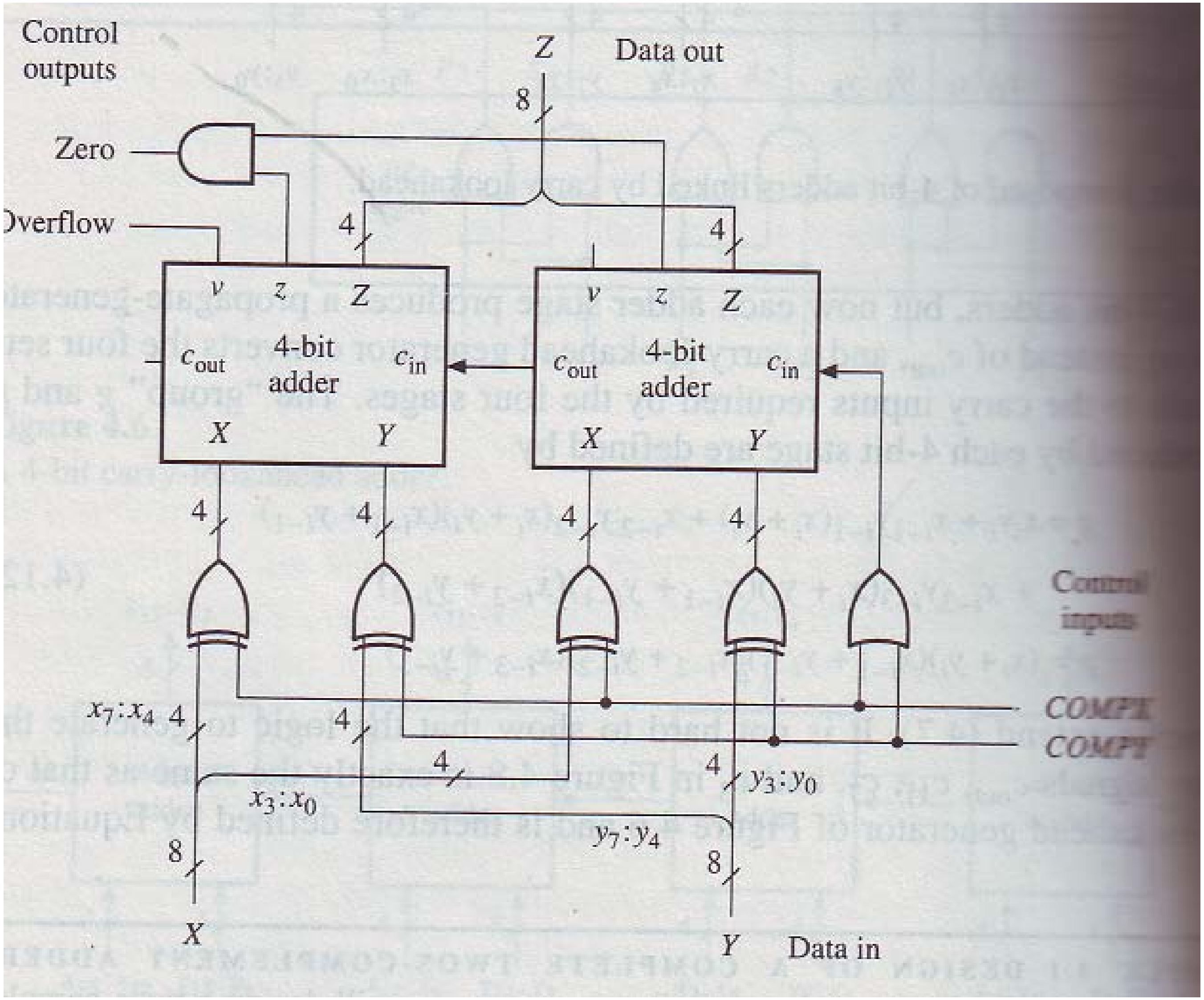


$$g = x_i y_i + x_{i-1} y_{i-1}(x_i + y_i) + x_{i-2} y_{i-2}(x_i + y_i)(x_{i-1} + y_{i-1})$$
$$+ x_{i-3} y_{i-3}(x_i + y_i)(x_{i-1} + y_{i-1})(x_{i-2} + y_{i-2})$$
$$p = (x_i + y_i)(x_{i-1} + y_{i-1})(x_{i-2} + y_{i-2})(x_{i-3} + y_{i-3})$$

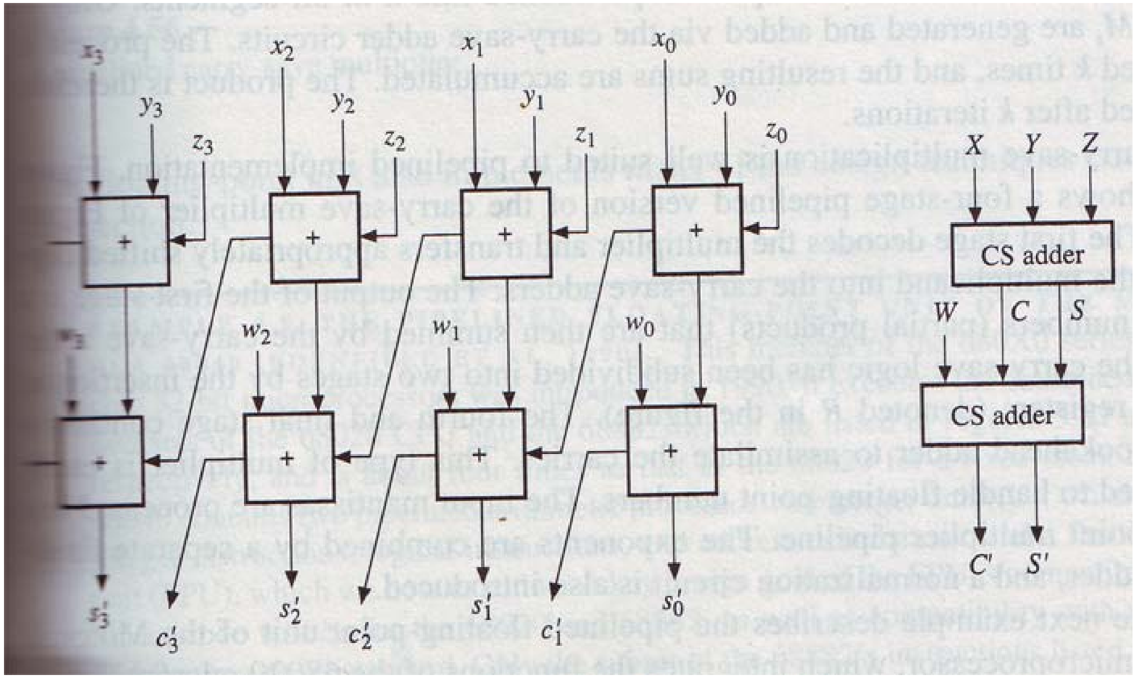
Complete 2’s Complement Adder-Subtracter



Complete 2's Complement Adder-Subtracter



Carry-Save Adder



Carry-Save Adder

- One of the major speed enhancement techniques used in modern digital circuits is the ability to add numbers with minimal carry propagation.

- The basic idea is that three numbers can be reduced to 2, in a 3:2 compressor, by doing the addition while keeping the carries and the sum separate.

```
         10111001
         00101010
         00111001
     Sum: 10101010
   Carry: 00111001
  Result: 100011100
```

no prop delay

no prop delay

only 1 propagation delay

A
+ B
+ C
+ D
+ E
+
S =
C =

no 4 prop delay

only 1 prop delay

- The sum and carry can then be recombined in a normal addition to form the correct result.