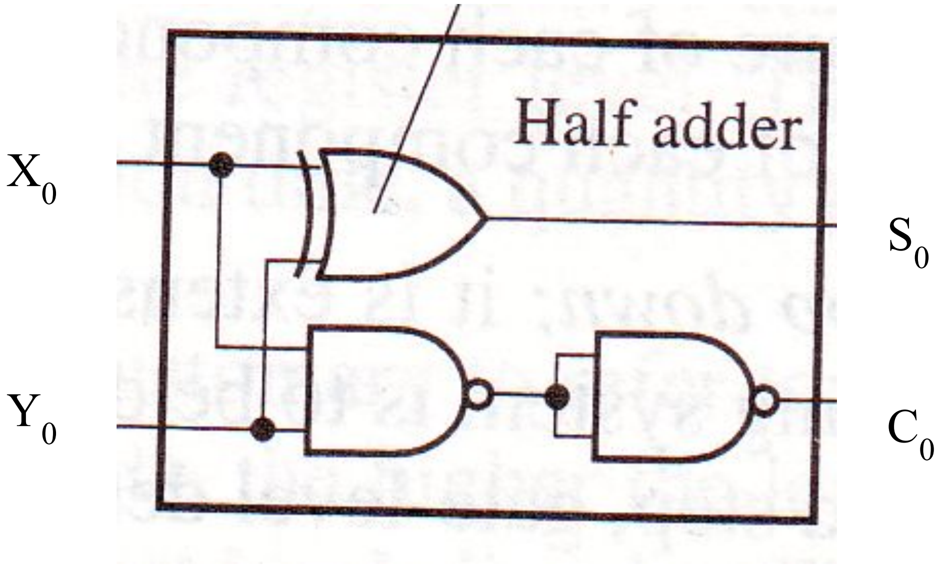
Lecture 2

Datapath Design
Book of John P. Hayes

Fixed-Point Arithmetic

- ✔ Addition
- ✔ Subtraction
- ✔ Multiplication
- ✔ Division

Half Adder

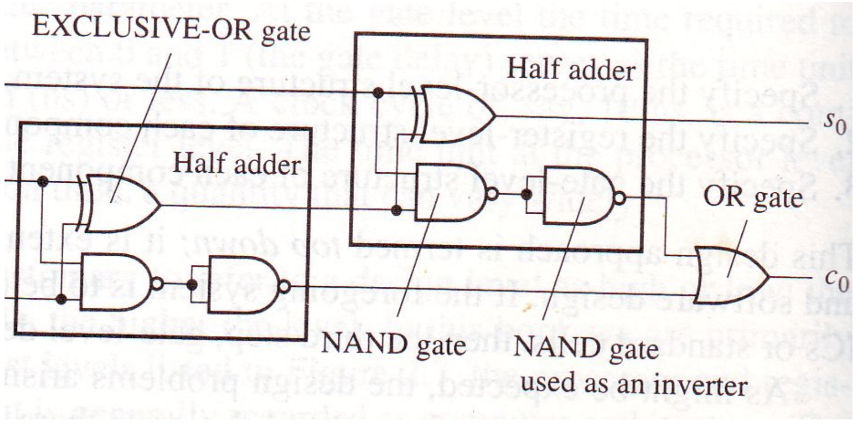


X_0 Y_0	S_0 C_0
0 0	0 0
0 1	1 0
1 0	1 0
1 1	0 1

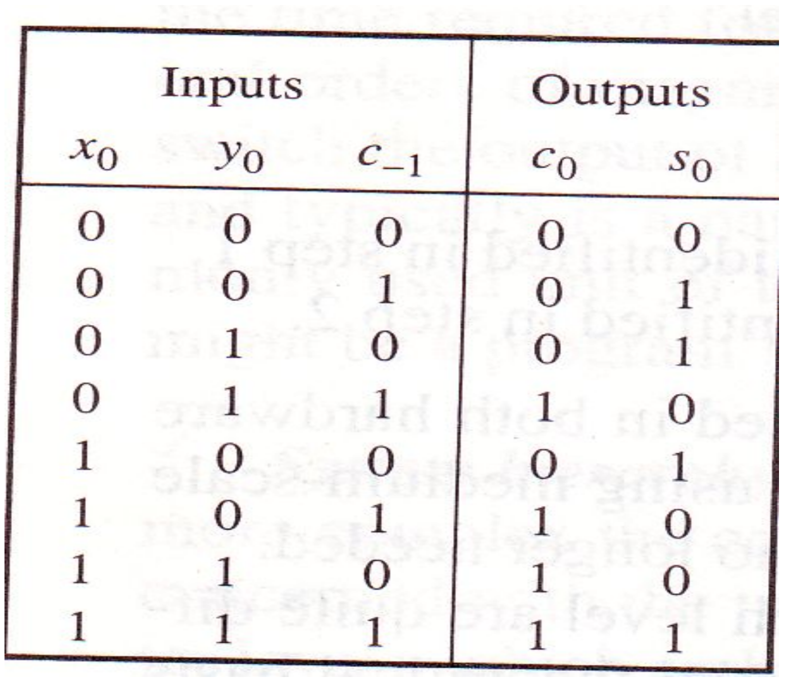
(a) Half Adder (b) Truth Table

$S_0 = X_0$ xor Y_0 and $C_0 = X_0Y_0$

Full Adder



(a) 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

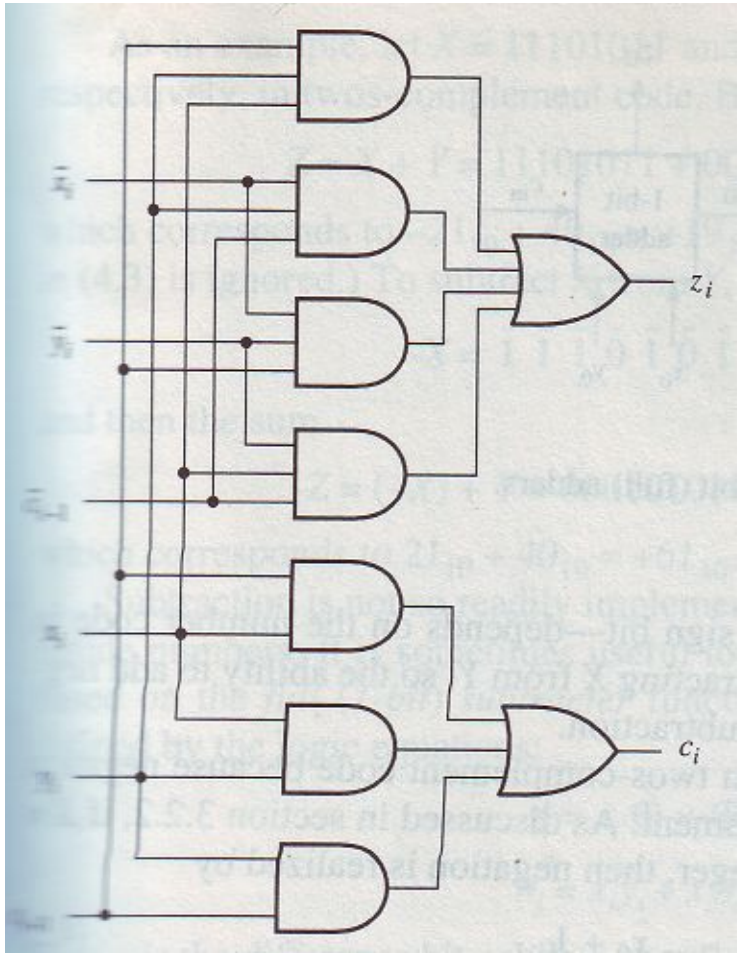
(b) Truth Table

$s_0 = x_0 \text{ xor } y_0 \text{ xor } c_{-1}$

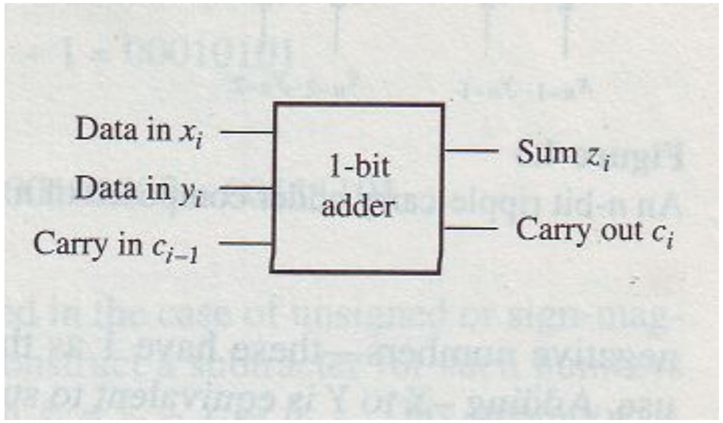
$c_0 = x_0y_0 + x_0c_{-1} + y_0c_{-1}$

$x_0y_0 + c(x \oplus y)$

Full Adder

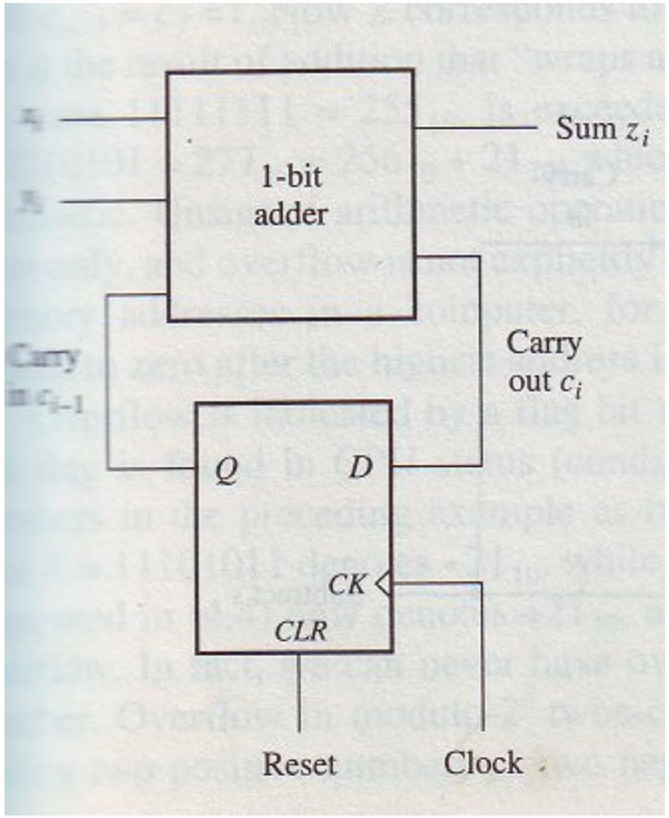


(a) Two-level AND-OR logic circuit



(b) Symbol

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 is independent of n.

Ripple Carry Adder / Parallel Adder

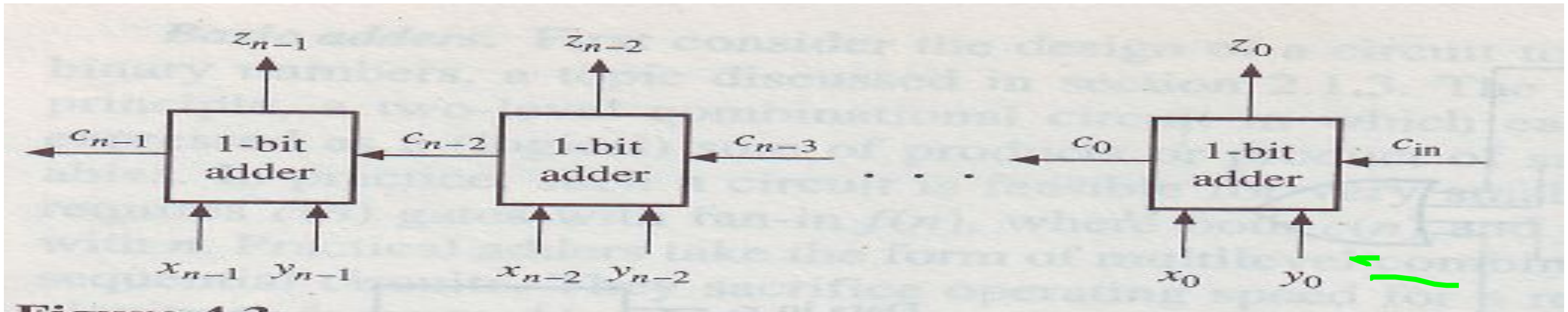


Figure 4.3
An n-bit ripple-carry adder composed of n 1-bit (full) adders.

✔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.

Subtracter for 2's Complement Number

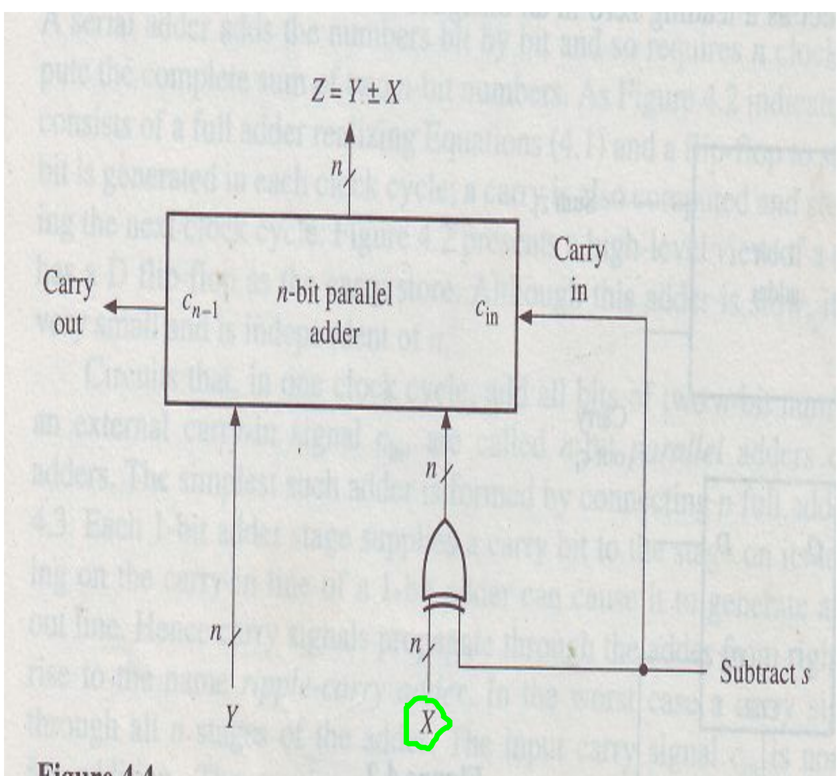
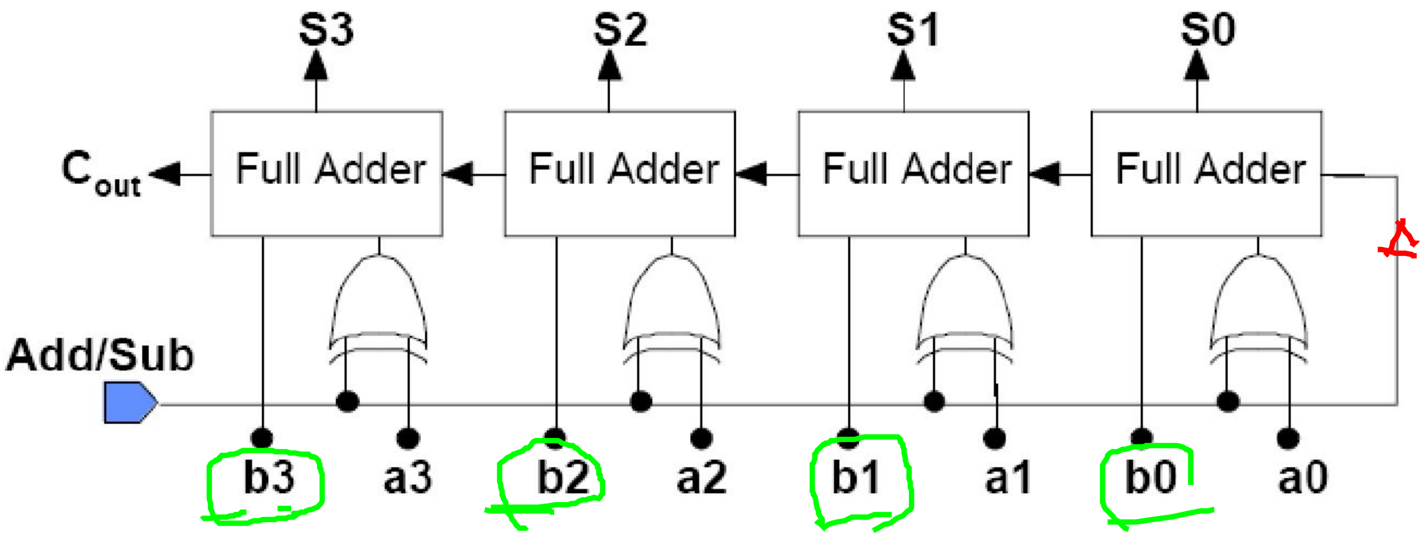


Figure 4.4
An n-bit twos-complement adder-subtracter.

- ✔ When s =0 then X xor s = X
- ✔ When s=1 then X xor s = $\overline{X}$

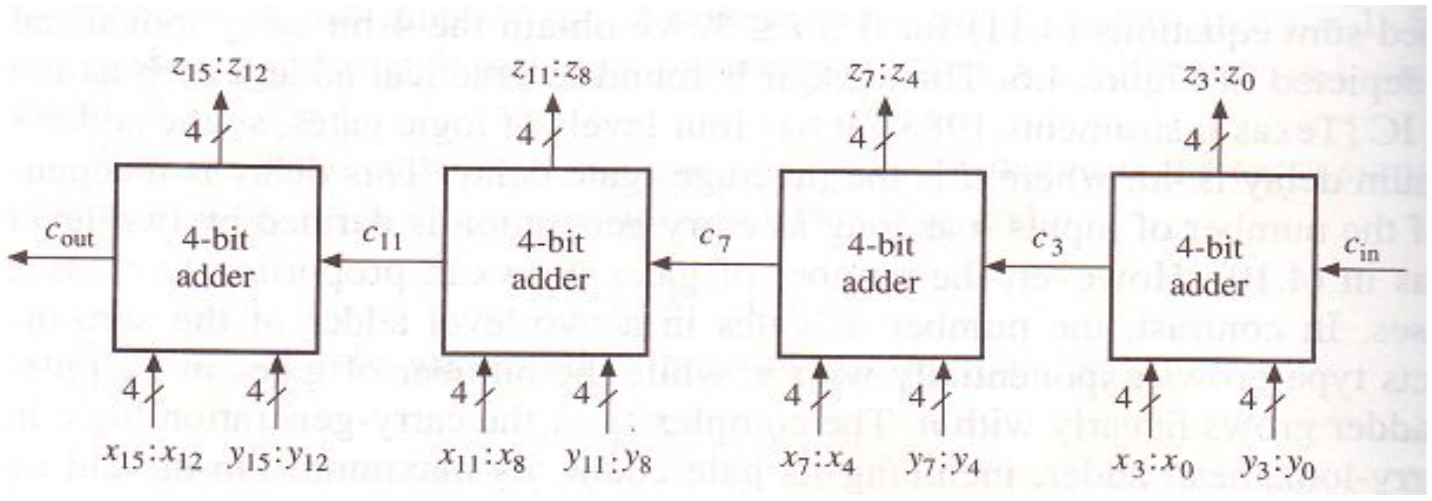
Subtracter for 2's Complement Number

Example: Adder/Subtractor



Subtracter

- ✔ For sign-magnitude number, it is useful to construct a subtracter on the full (1-bit) subtracter function $z_i = x_i \text{ xor } y_i \text{ xor } b_{i-1}$



16 bit Adder

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

$$
\begin{array}{r}
+\ 00101010 \\ \hline
00010101\ \mathbf{=21_{10}}
\end{array}
\qquad C_7 = 1
$$

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}** 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.
- ✔ **<u>Case 1:</u>** Two numbers are positive.

  Let n=4 +7 = 0111 +3 = 0011 so, 0111+0011 = $\overset{\downarrow}{1}010$ so, c_{n-2} = 1
- ✔ C_{n-2} =1 indicates that the magnitude of the sum exceeds the n-1 bits allocated to it.

Overflow

- ✔ **<u>Case 2:</u>** Two numbers are negative.

  Let n=4 -7 = 1001 -3 = 1101 so, 1001+1101 = $\overset{\downarrow}{1}\overset{\downarrow}{0}110$ so, $c_{n-2} = 0$
- ✔ C_{n-2} =0 indicates the overflow.
- ✔ For n the number for 2's complement number is $+(2^{n-1}-1)$ to -2^{n}.

Overflow

- ✔ $Z_{n-1}Z_{n-2}\ldots Z_0 := X_{n-1}X_{n-2}\ldots X_0 + Y_{n-1}Y_{n-2}\ldots Y_0$
- ✔ $v = X_{n-1}Y_{n-1}\overline{C_{n-2}} + \overline{X_{n-1}}\,\overline{Y_{n-1}}C_{n-2}$
- ✔ $v = C_{n-1}$ xor 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

Ripple Carry Adder

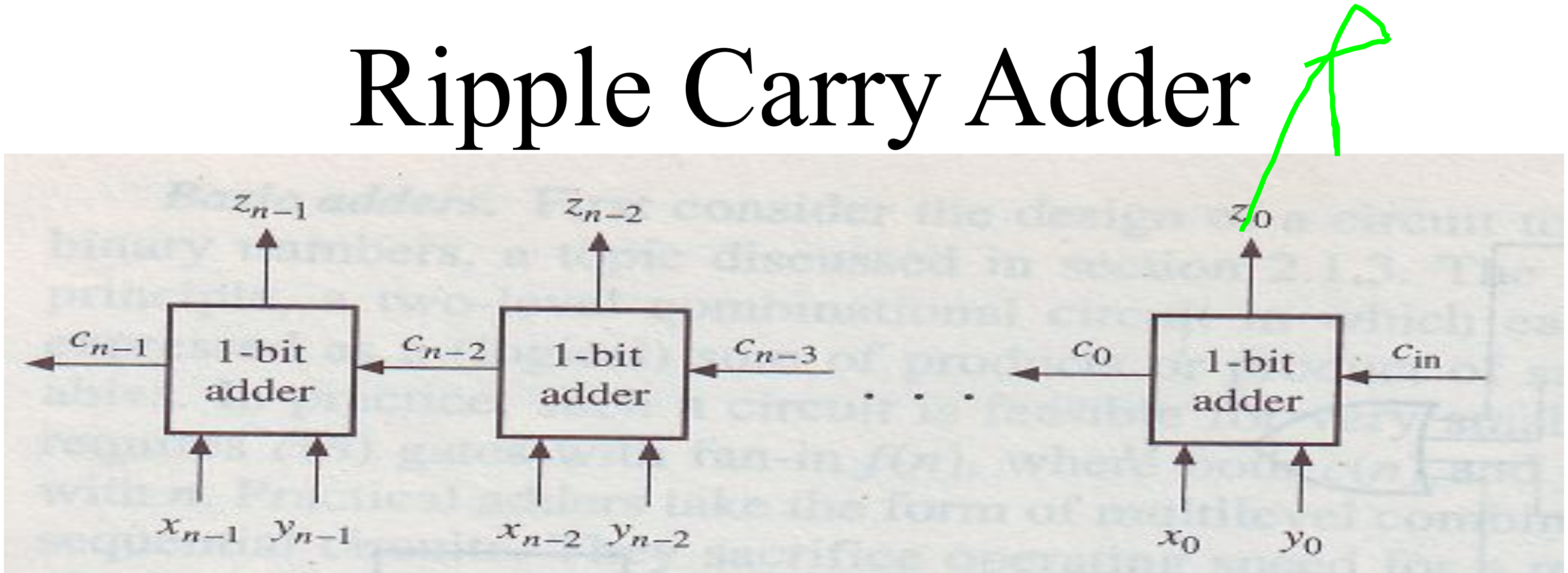


Figure 4.3
An n-bit ripple-carry adder composed of n 1-bit (full) adders.

- ✔ 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.

High Speed Adder

- ✔ Reduce the time required to form carry signals.
- ✔ **Approach:** To compute the input carry needed by stage *i* directly from carrylike signals obtained from all the preceeding stages i-1, i-2, ..,0.
- ✔ Adders that use this principle are called carry-lookahead adders.

Carry Lookahead Adder

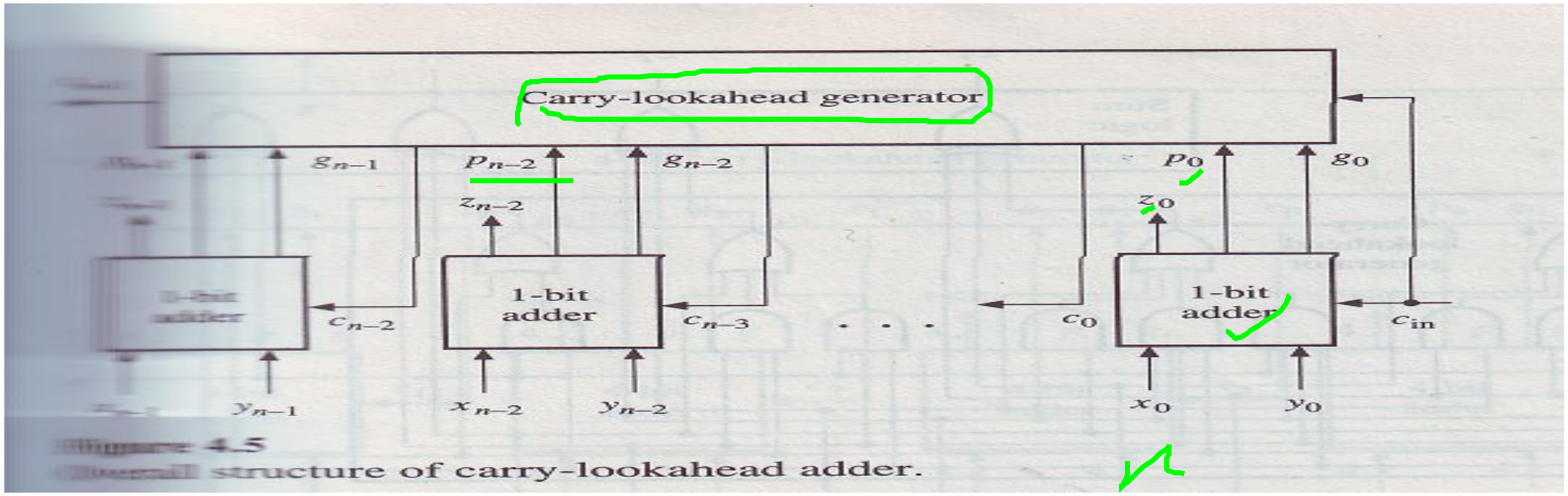


Figure 4.5
Overall structure of carry-lookahead adder.

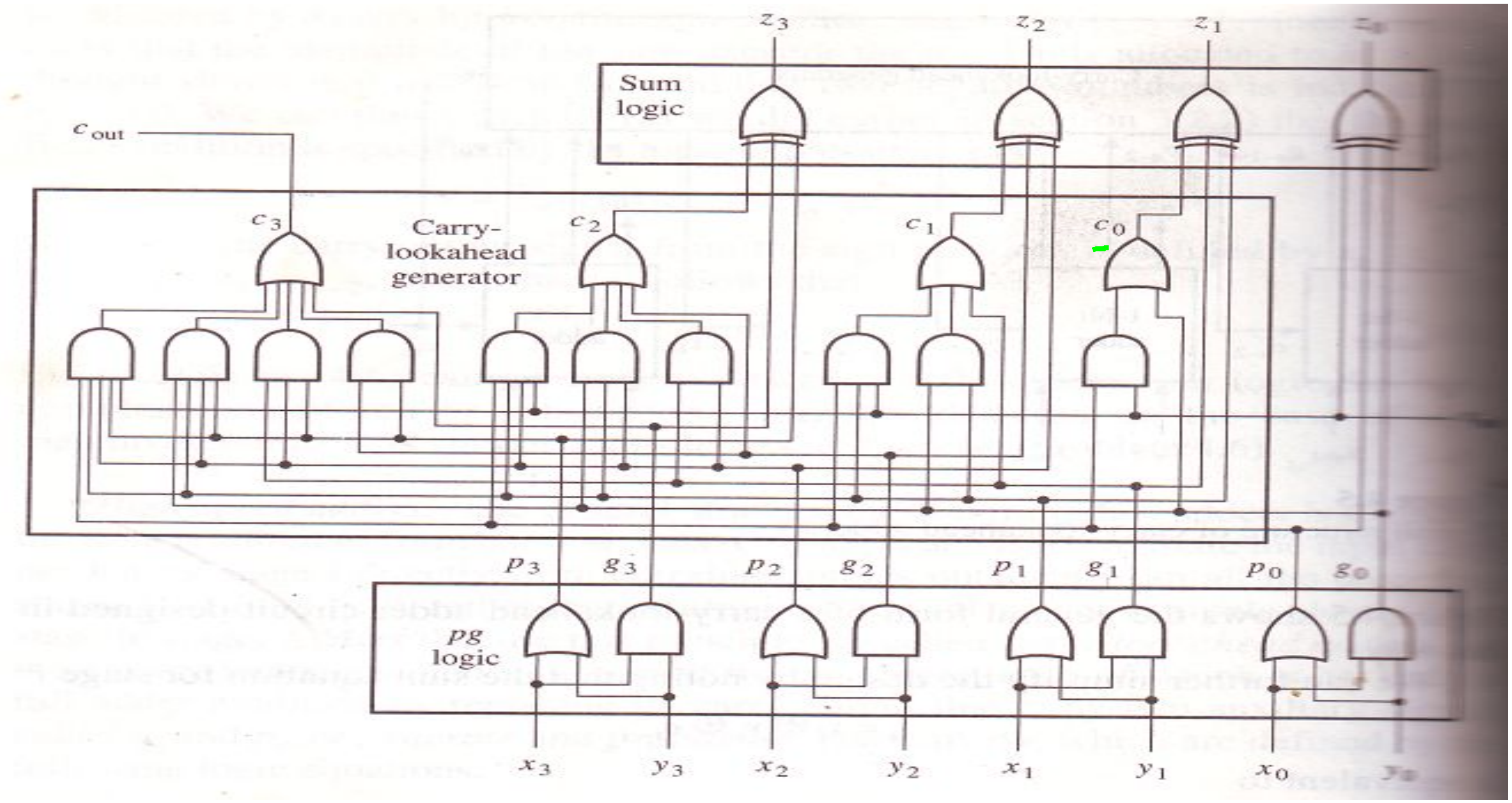
- ✔generate signal $g_i = x_i y_i$ and propagate signal $p_i = x_i + y_i$
- ✔$c_i = x_i y_i + x_i c_{i-1} + y_i c_{i-1}$ is the carry to be sent to the stage i+1.
- ✔$c_i = g_i + p_i c_{i-1}$
- ✔$c_{i-1} = g_{i-1} + p_{i-1} c_{i-2}$
- ✔$c_i = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-2}$

4-bit Carry Lookahead Adder

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

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

4-bit Carry Lookahead Adder



4-bit CarryLookahead Adder

- ✔ Maximum delay is 4d, where d is the average gate delay. It is independent of number of input n.
- ✔ 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.

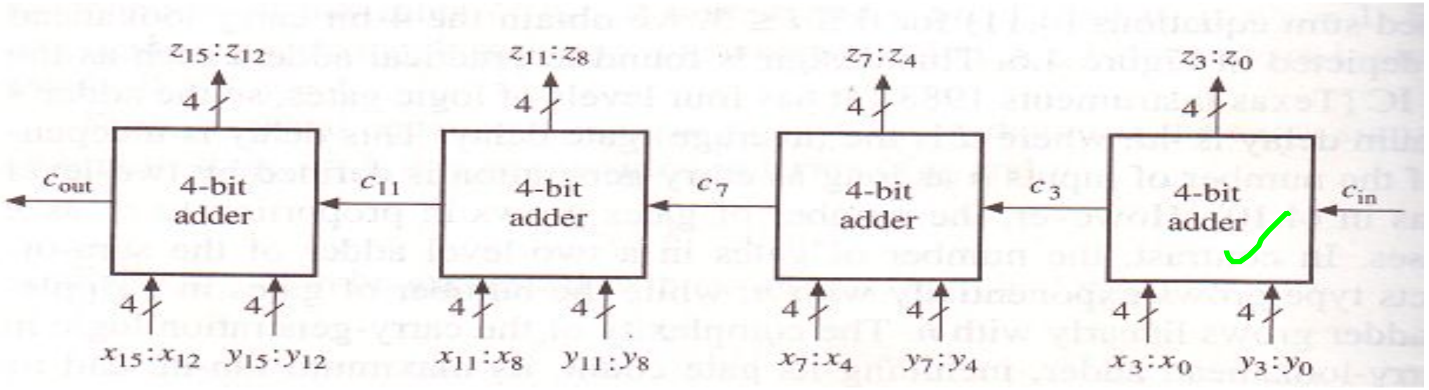
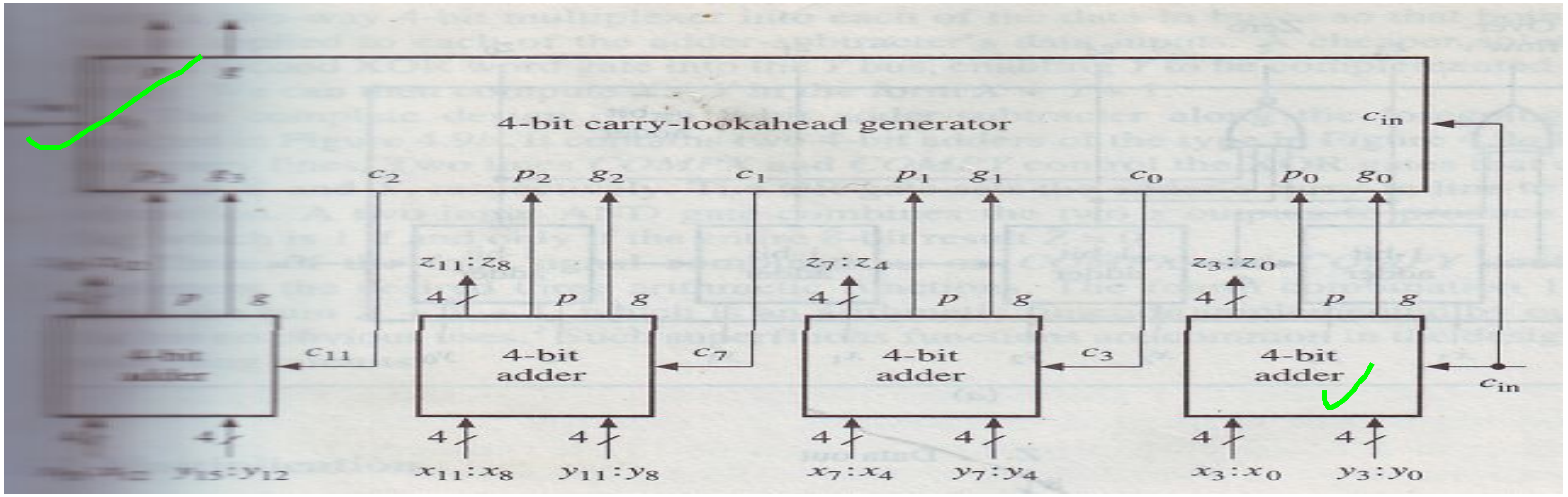


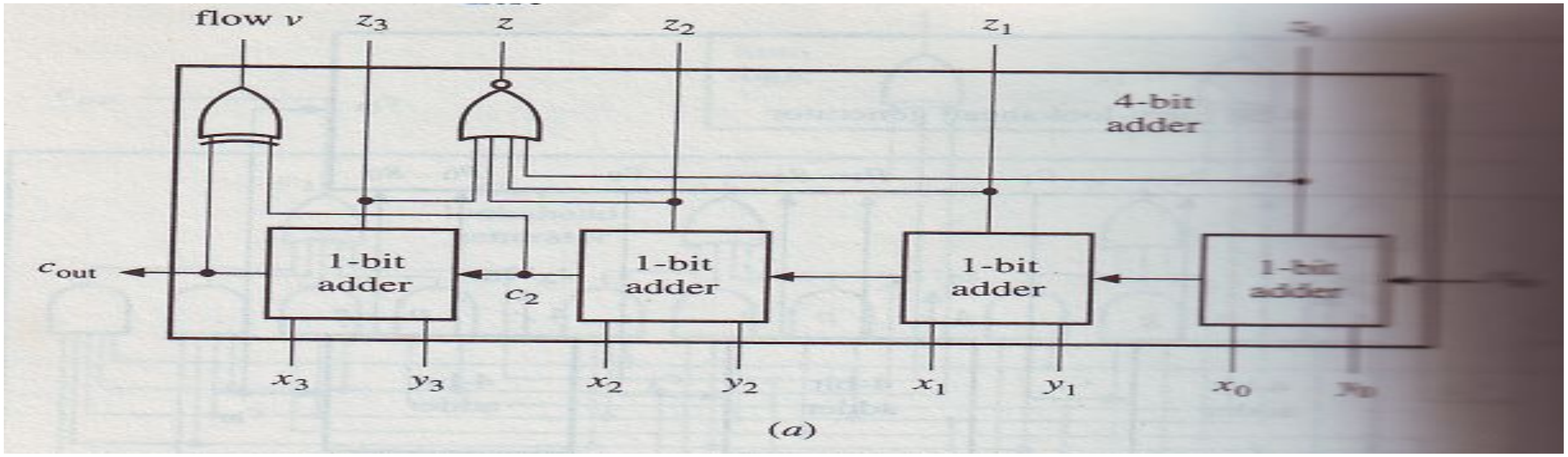
Figure: A 16-bit adder composed of 4-bit adders linked by ripple-carry propagation

Adder Expansion

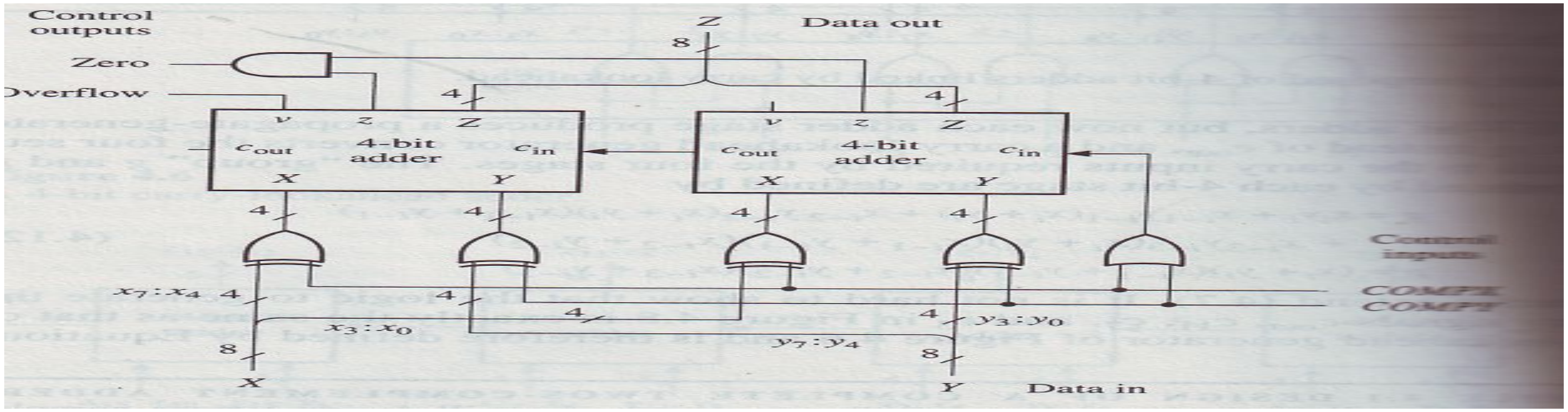


$$g = x_iy_i + x_{i-1}y_{i-1}(x_i + y_i) + x_{i-2}y_{i-2}(x_i + y_i)(x_{i-1} + y_{i-1})$$
$$+ x_{i-3}y_{i-3}(x_i + y_i)(x_{i-1} + y_{i-1})(x_{i-2} + y_{i-2})$$
$$p_3 = \underline{(x_i + y_i)}\underline{(x_{i-1} + y_{i-1})}\underline{(x_{i-2} + y_{i-2})}\underline{(x_{i-3} + y_{i-3})}$$

Design of a Complete 2's Complement Adder-Subtracter



(a)



Chapter 4, Section 4.1 of John P. Hayes Book