Functional Blocks: Addition

- Binary addition used frequently
- Addition Development:
  - *Half-Adder* (HA), a 2-input bit-wise addition functional block,
  - *Full-Adder* (FA), a 3-input bit-wise addition functional block,
  - *Ripple Carry Adder*, an iterative array to perform binary addition, and
  - *Carry-Look-Ahead Adder* (CLA), a hierarchical structure to improve performance.

Functional Block: Half-Adder

- A 2-input, 1-bit width binary adder that performs the following computations:

X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

- A half adder adds two bits to produce a two-bit sum
- The sum is expressed as a sum bit, S and a carry bit, C
- The half adder can be specified as a truth table for S and C ⇒

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Logic Simplification: Half-Adder

- The K-Map for S, C is:
- This is a pretty trivial map!
  By inspection:

S

	Y'	Y
X'	$_0$	1 $_1$
X	1 $_2$	$_3$

C

	Y'	Y
X'	$_0$	$_1$
X	$_2$	1 $_3$

$$\mathbf{S = XY' + X'Y = X \oplus Y}$$

$$\mathbf{S = (X+Y)\,(X'+Y')}$$

Recall that $S' = X'Y' + XY$

and

$$\mathbf{C = XY}$$

$$\mathbf{C = ((XY)')'}$$

- These equations lead to several implementations.

Five Implementations: Half-Adder

- We can derive following sets of equations for a half-adder:

(a) $S = XY' + X'Y$
$C = XY$

(b) $S = (X+Y)(X'+Y')$
$C = XY$

(c) $S = (C + X'Y')'$
$C = XY$

(d) $S = (X+Y)\,C'$
$C' = X'+Y'$

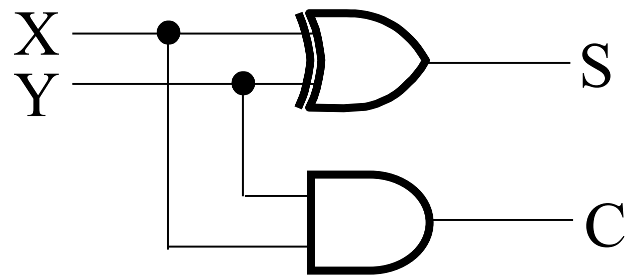
(e) $S = X \oplus Y$
$C = XY$

- (a), (b), and (e) are SOP, POS, and XOR implementations for S.
- In (c), the C function is used as a term in the AND-NOR implementation of S, and in (d), the C' function is used in a POS term for S.

Implementations: Half-Adder

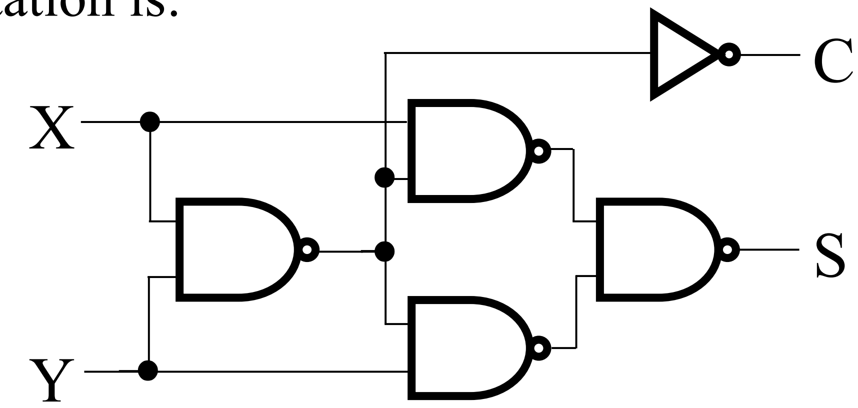
- The most common half adder implementation is: (e)

  $S = X \oplus Y$
  $C = XY$



- A NAND only implementation is:

  $S = (X+Y)\ C'$
  $C = ((XY)')'$



Functional Block: Full-Adder

- A full adder is similar to a half adder, but includes a carry-in bit from lower stages. Like the half-adder, it computes a sum bit, S and a carry bit, C.
  - For a carry-in (Z) of 0, it is the same as the half-adder:

Z	**0**	**0**	**0**	**0**
X	**0**	**0**	**1**	**1**
+ Y	**+ 0**	**+ 1**	**+ 0**	**+ 1**
C S	**0 0**	**0 1**	**0 1**	**1 0**

  - For a carry- in (Z) of 1:

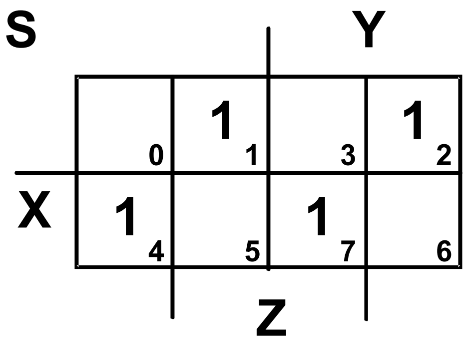
Z	**1**	**1**	**1**	**1**
X	**0**	**0**	**1**	**1**
+ Y	**+ 0**	**+ 1**	**+ 0**	**+ 1**
C S	**0 1**	**1 0**	**1 0**	**1 1**

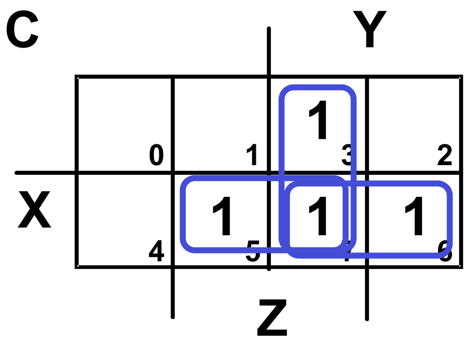
Logic Optimization: Full-Adder

- Full-Adder Truth Table:

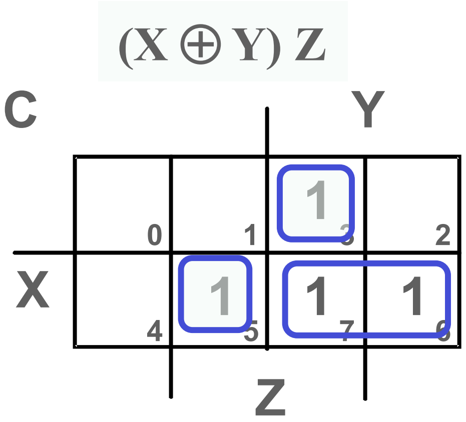
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- Full-Adder K-Map:





Equations: Full-Adder



- From the K-Map, we get:

$$\mathbf{S = XY'Z' + X'YZ' + X'Y'Z + XYZ}$$
$$\mathbf{C = XY + XZ + YZ}$$

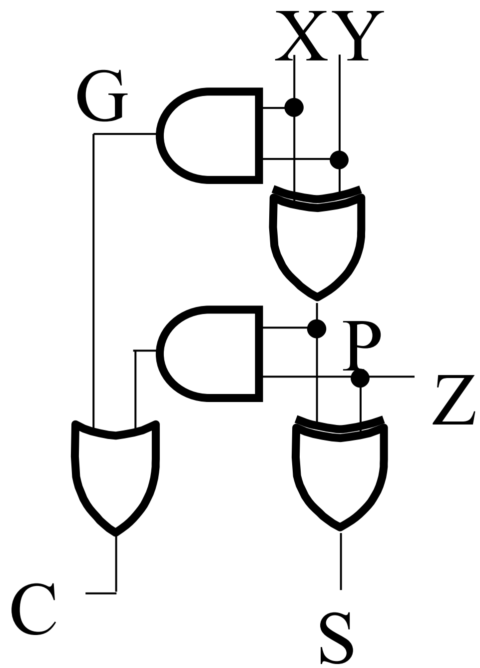
- The S function is the three-bit XOR function (Odd Function):

$$\mathbf{S = X \oplus Y \oplus Z}$$

- The Carry bit C is 1 if both X and Y are 1 (the sum is 2), or if the sum $(X \oplus Y)$ is 1 and a carry-in (Z) occurs. Thus C can be re-written as:

$$\mathbf{C = XY + (X \oplus Y)\, Z}$$

- The term $X \cdot Y$ is *carry generate (G)*.
- The term $X \oplus Y$ is *carry propagate (P)*.

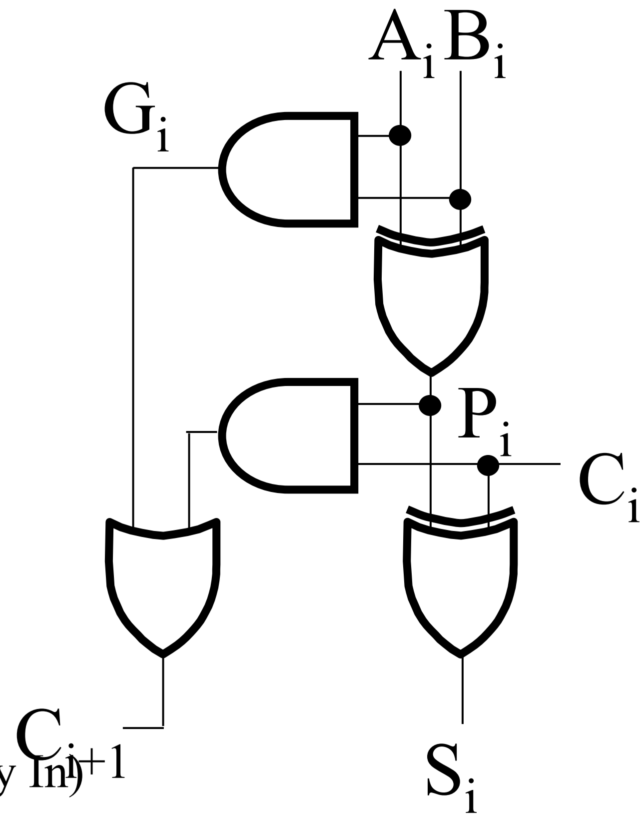


Implementation: Full Adder

- Full Adder Schematic
- Here X, Y, and Z, and C (from the previous pages) are A, B, C_i and C_o, respectively. Also,

  G = generate and
  P = propagate.

- Note: This is really a combination of a 3-bit odd function (for S)) and Carry logic (for C_o):



(G = Generate) OR (P =Propagate AND C_i = Carry In)

$C_o = G + P \cdot C_i$

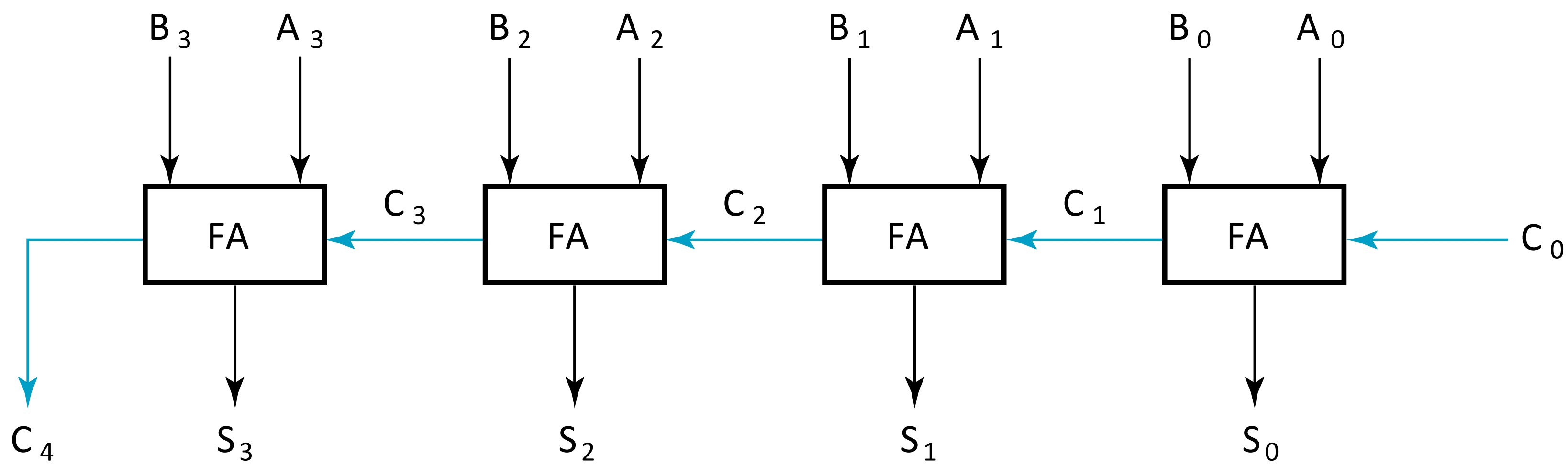
Binary Adders

- To add multiple operands, we "bundle" logical signals together into vectors and use functional blocks that operate on the vectors
- Example: 4-bit ripple carry adder: Adds input vectors A(3:0) and B(3:0) to get a sum vector S(3:0)
- Note: carry out of cell i becomes carry in of cell i + 1

Description	Subscript 3 2 1 0	Name
Carry In	**0 1 1 0**	C_i
Augend	**1 0 1 1**	A_i
Addend	**0 0 1 1**	B_i
Sum	**1 1 1 0**	S_i
Carry out	**0 0 1 1**	C_{i+1}

4-bit Ripple-Carry Binary Adder

- A 4-bit Ripple Carry Adder made from four 1-bit Full Adders:



Binary Adder/Subtractor

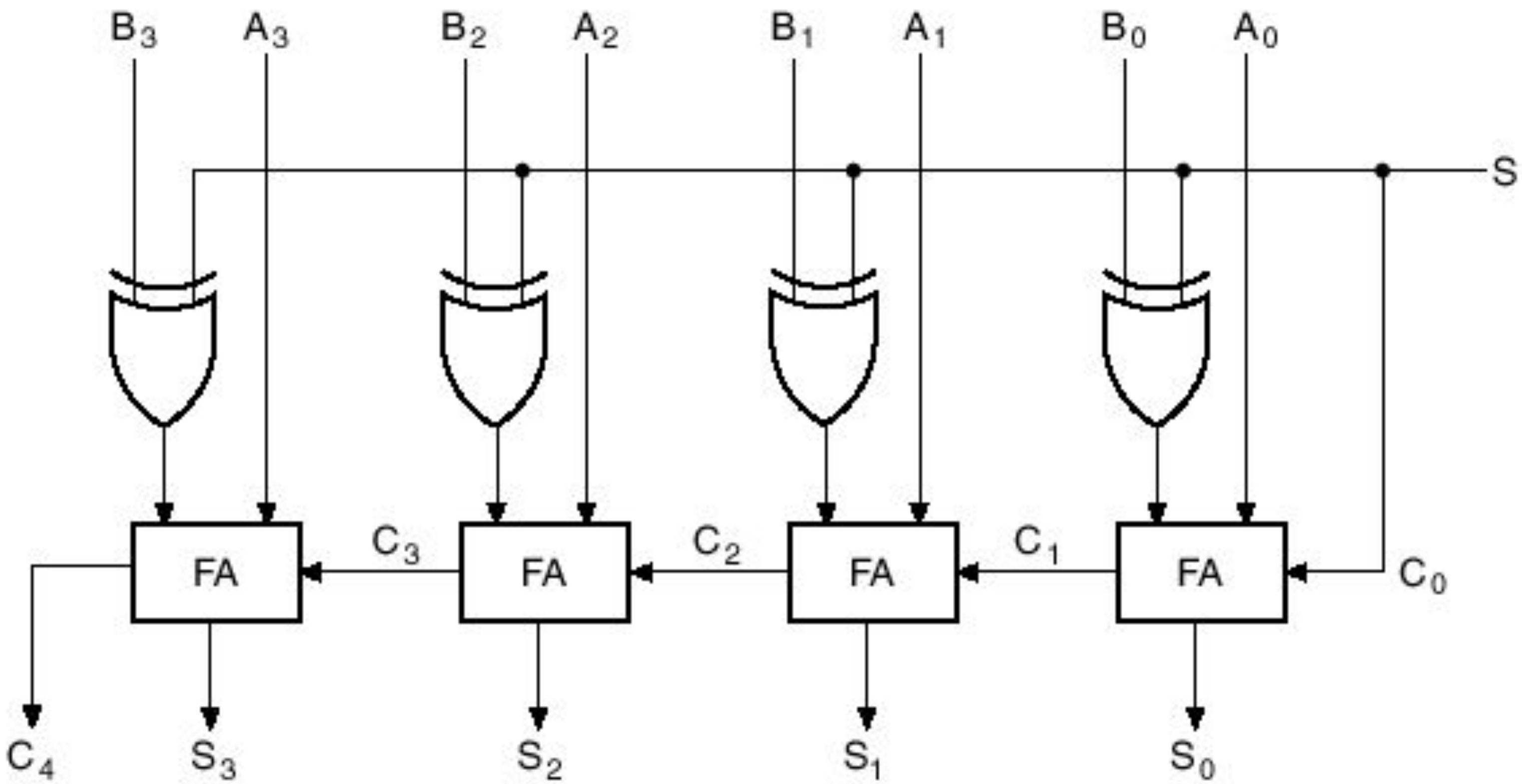


Fig. 3-31 Adder-Subtractor Circuit