CSE1003-Digital Logic Design

Module-3 COMBINATIONAL CIRCUITS-I

Module:3	COMBINATIONAL CIRCUIT - I	4 hours
Adder - Subtractor - Code Converter - Analyzing a Combinational Circuit		
Module:4	COMBINATIONAL CIRCUIT –II	6 hours
Binary Parallel Adder- Look ahead carry - Magnitude Comparator - Decoders – Encoders - Multiplexers –Demultiplexers.		

Overview

- Part 1 – Design Procedure
- Part 2 – Combinational Logic
- Part 3 – Arithmetic Functions
  - Iterative combinational circuits
  - Binary adders
    - Half and full adders
    - Ripple carry and carry lookahead adders
  - Binary subtraction
  - Binary adder-subtractors
    - Signed binary numbers
    - Signed binary addition and subtraction
    - Overflow
  - Binary Multiplication

Remember

- **Combinational**
  - The outputs depend only on the current input values
  - It uses only logic gates
- **Sequential**
  - The outputs depend on the current and past input values
  - It uses logic gates and storage elements

Notes

- If there are *n* input variables, there are 2^n input combinations
- For each input combination, there is one output value
- Truth tables are used to list all possible combinations of inputs and corresponding output values

Basic Combinational Circuits

- Adders
- Subtractors
- Multipliers
- Multiplexers
- Decoders
- Encoders
- Comparators

Part-1-Design Procedure

- Determine the inputs and outputs
- Assign a symbol for each
- Derive the truth table
- Get the simplified boolean expression for each output
- Draw the network diagram

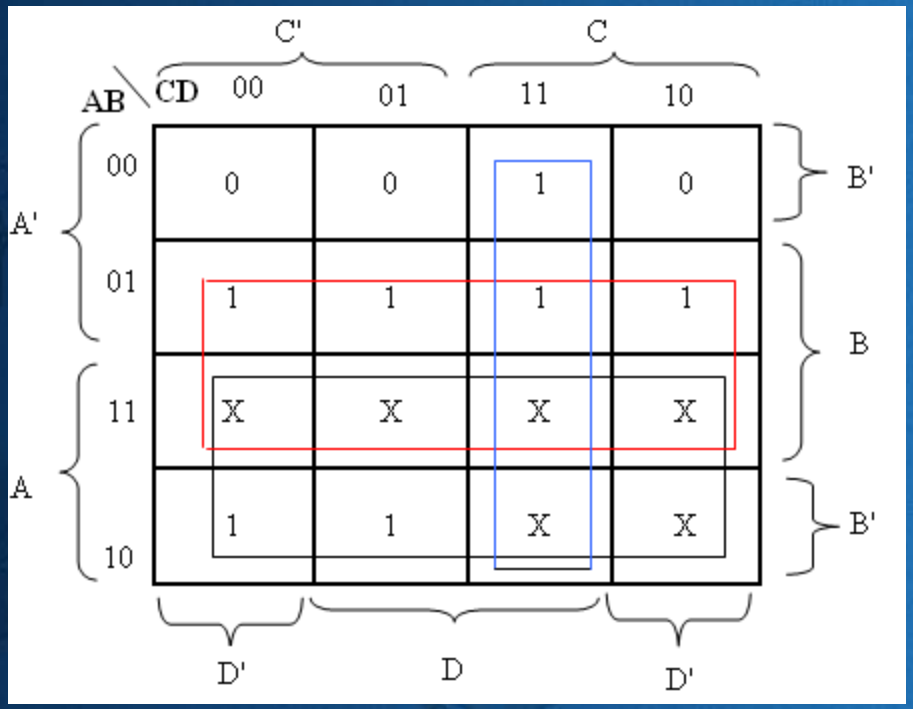
Example

- Conversion from BCD to excess-5

INPUT				OUTPUT			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	1	0	1
0	0	0	1	0	1	1	0
0	0	1	0	0	1	1	1
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	1	1	1	0

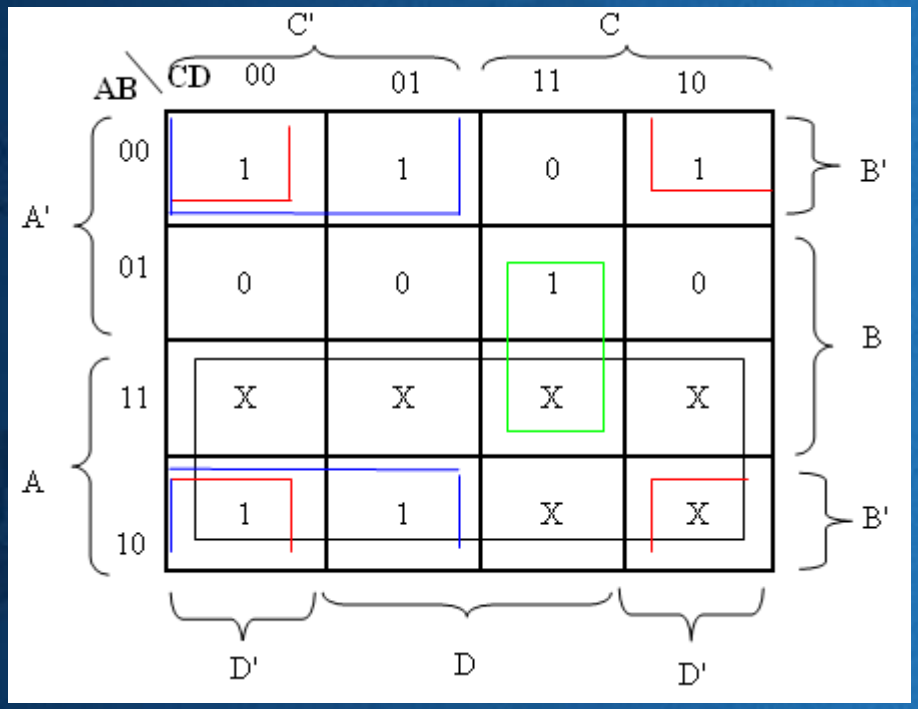
Example (Cont.)

$$W = A + B + CD$$



Example (Cont.)

$$X = A + B'D' + B'C' + BCD$$



Example (Cont.)

Find Y and Z

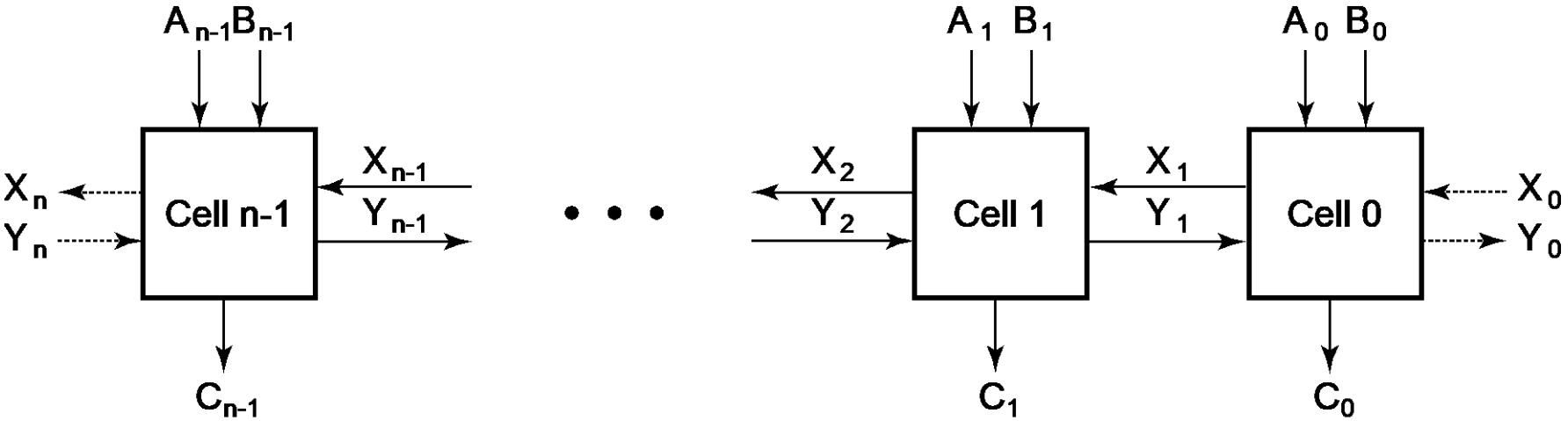
Draw the network diagram

For more details read the text Book

Iterative Combinational Circuits

- **Arithmetic functions**
  - **Operate on binary vectors**
  - **Use the same subfunction in each bit position**
- **Can design functional block for subfunction and repeat to obtain functional block for overall function**
- ***Cell* - subfunction block**
- ***Iterative array* - a array of interconnected cells**
- **An iterative array can be in a <u>single</u> dimension (1D) or <u>multiple</u> dimensions**

Block Diagram of a 1D Iterative Array



- **Example: n = 32**
  - **Number of inputs = ?**
  - **Truth table rows = ?**
  - **Equations with up to ? input variables**
  - **Equations with huge number of terms**
  - **Design impractical!**
- **Iterative array takes advantage of the regularity to make design feasible**

Adders

- Essential part of every CPU
- Half adder (Ignore the carry-in bit)
  - It performs the addition of two bits
- Full adder
  - It performs the addition of three bits

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 <u>sum bit</u> , S and a <u>carry bit</u>, 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	
		0	1 1
X	1 2		3

C		Y	
		0	1
X		2	1 3

$$S = X \times \overline{Y} + \overline{X} \times Y = X \oplus Y$$

$$S = (X + Y) \times \overline{(X + Y)}$$

- **and**

$$C = X \times Y$$

$$C = \overline{(\overline{(X \times Y)})}$$

- **These equations lead to several implementations.**

Five Implementations: Half-Adder

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

(a) $S = X \times \overline{Y} + \overline{X} \times Y$
$C = X \times Y$

(b) $S = (X + Y) \times (\overline{X} + \overline{Y})$
$C = X \times Y$

(c) $S = \overline{(C + \overline{X} \times \overline{Y})}$
$C = X \times Y$

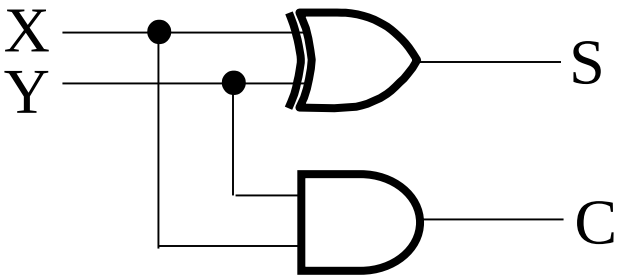
(d) $S = (X + Y) \times \overline{C}$
$C = (\overline{X} + \overline{Y})$

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

- (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 $\overline{C}$ function is used in a POS term for S.

Implementations: Half-Adder

- **The most common half adder implementation is:**

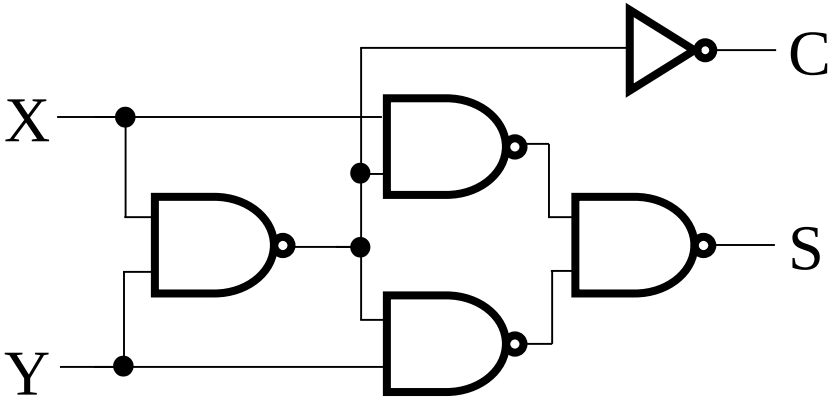


(e)

$$\mathbf{S = X \oplus Y}$$
$$\mathbf{C = X \times Y}$$

- **A NAND only implementation is:**

$$S = (X + Y) \times \overline{C}$$
$$= X \times \overline{C} + Y \times \overline{C}$$
$$= \overline{\overline{(\overline{X \times \overline{C}}) \times (\overline{Y \times \overline{C}})}}$$
$$C = \overline{\overline{X \times Y}}$$



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:

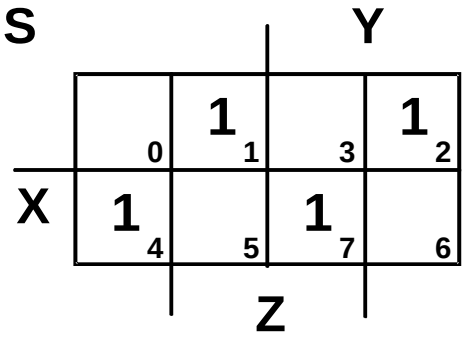
Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

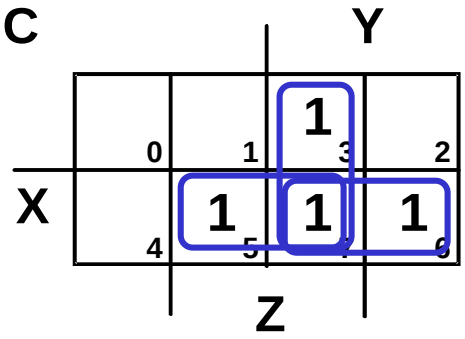
Logic Optimization: Full-Adder

- **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:

$$S = X\,\overline{Y}\,\overline{Z} + \overline{X}\,Y\,\overline{Z} + \overline{X}\,\overline{Y}\,Z + X\,Y\,Z$$

$$C = X\,Y + X\,Z + Y\,Z$$

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

$$S = X \oplus Y \oplus Z$$

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

$$C = X\,Y + (X \oplus Y)\,Z$$

- The term $X \cdot Y$ is *carry generate*.
- The term $X \oplus Y$ is *carry propagate*.

Logic diagram of full adder

$$S = X \oplus Y \oplus Z$$

$$C = XY + (X \oplus Y)Z$$

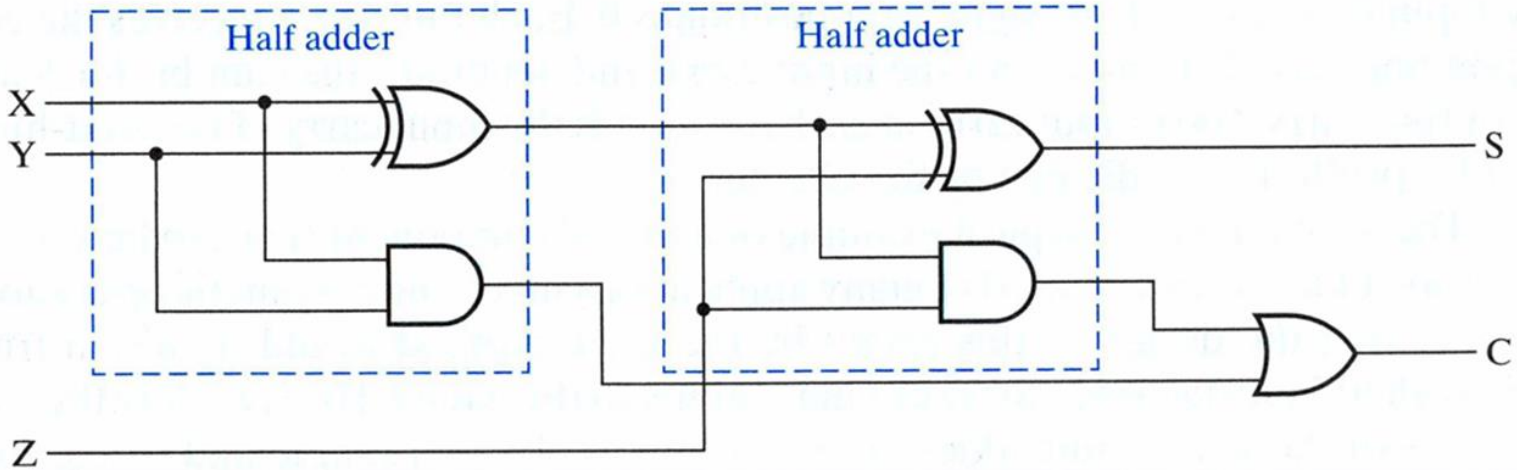


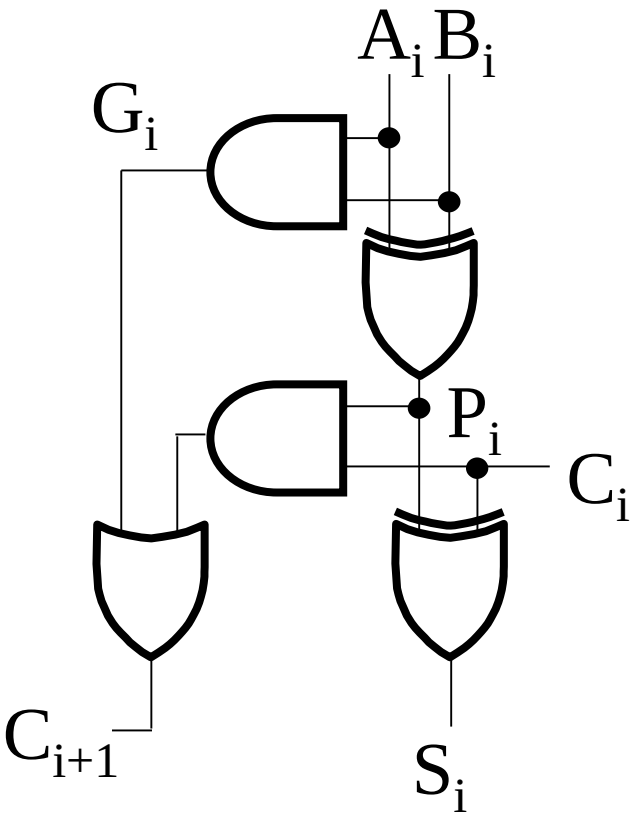
FIGURE 3-42
Logic Diagram of Full Adder

Another Implementation of full adder

- **Full Adder Schematic**
- **Here X, Y, and Z, and C (from the previous pages) are A_i, B_i, C_i and C_{i+1}, 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_{i+1}):**



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

$$C_{i+1} = G + P \cdot Ci$$

Binary Adders

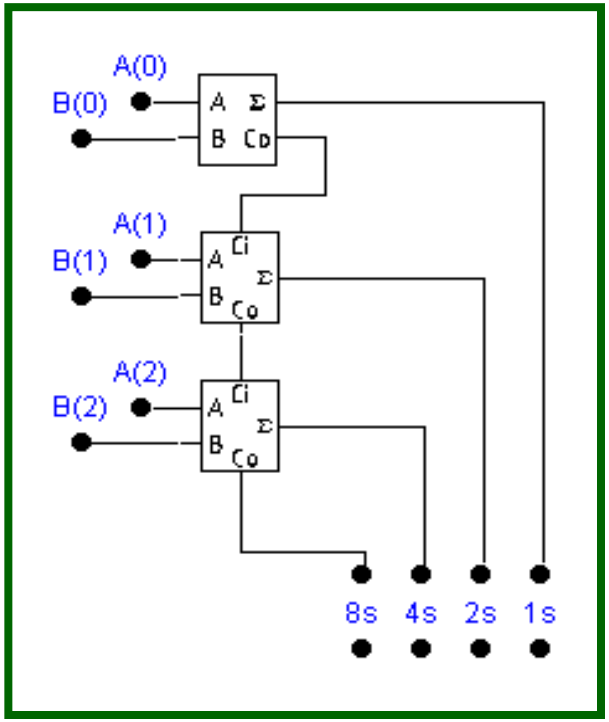
- To add multiple operands, we "bundle" logical signals together into vectors and use functional blocks that operate on the vectors
- Example: <u>4-bit ripple carry adder:</u> Adds input vectors A(3:0) and B(3:0) to get a sum vector S(3:0)
- Note: carry out of cell i becomes carry in of cell i + 1

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}

BINARY PARALLEL ADDING

- Use half adder for LSD
- Use full adder for other digits

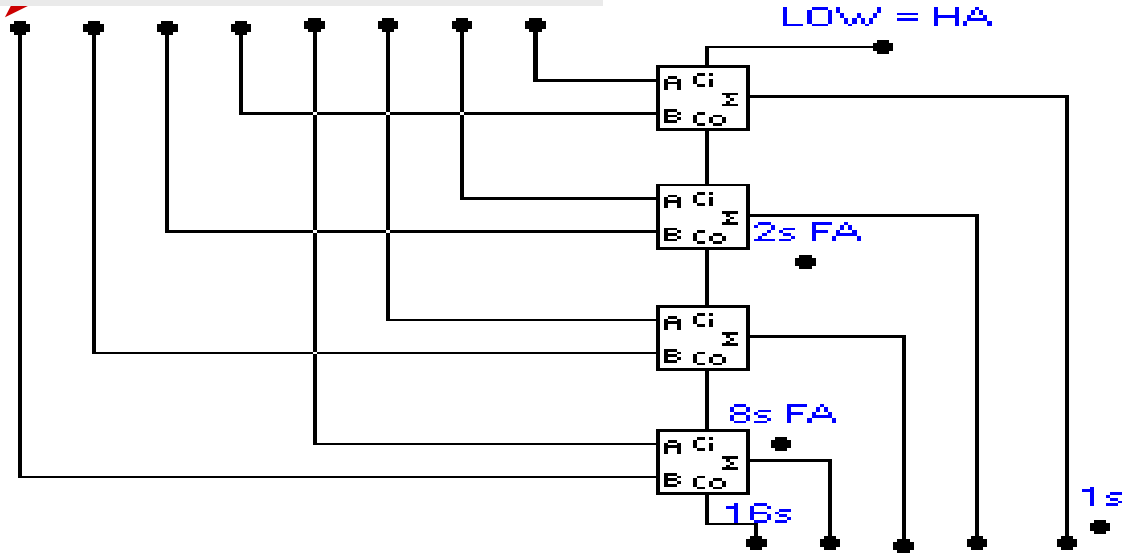
$$\begin{array}{rrrr} & A_2 & A_1 & A_0 \\ + & B_2 & B_1 & B_0 \\ \hline \end{array}$$



4-BIT PARALLEL ADDER

Enter binary numbers to be added

1 1 1 0 + 0 1 1 0



1 0 1 0 0

Sum appears here

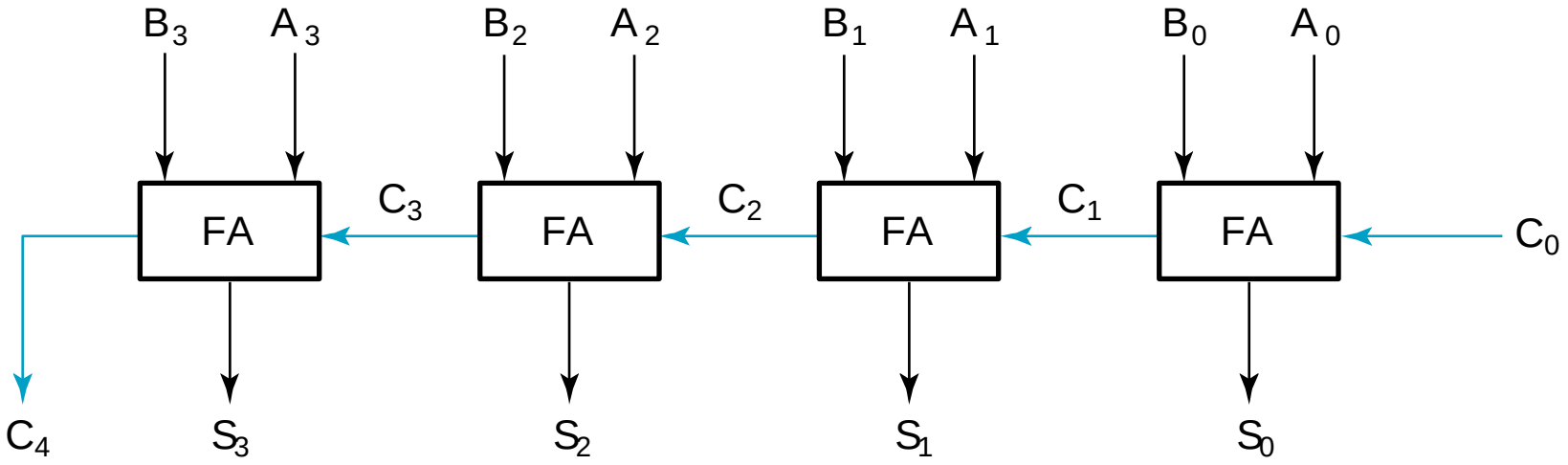
Parallel adders are available in IC form.

1s place uses half-adder

2s, 4s, 8s places use full adders

4-bit Ripple-Carry Binary Adder

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



BINARY SUBTRACTION

Example: **Subtract binary number 101 from 1011**

(borrow)

$$
\begin{array}{rcccc}
 & 0 & 1 & & \\
 & \cancel{1} & 0 & 1 & 1 \\
- & & 1 & 0 & 1 \\
\hline
 & 0 & 1 & 1 & 0
\end{array}
$$

TEST

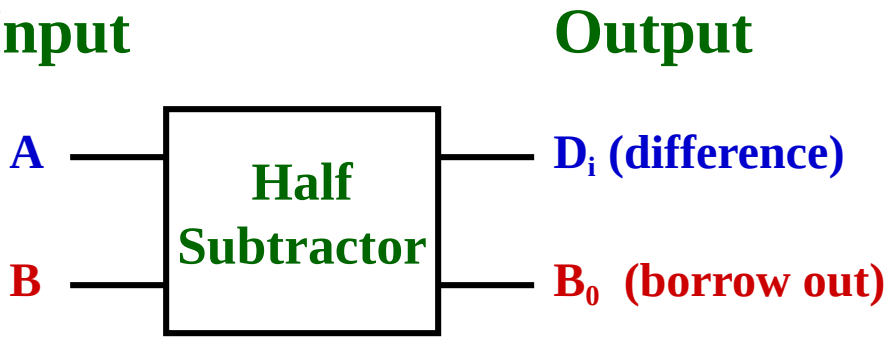
Subtract binary number 11 from 1010

$$
\begin{array}{cccc}
 & 01 & & \\
0 & \cancel{10} & 10 & \\
\cancel{1} & \cancel{0} & \cancel{1} & {}^{1}0 \\
- & & 1 & 1 \\
\hline
0 & 1 & 1 & 1
\end{array}
$$

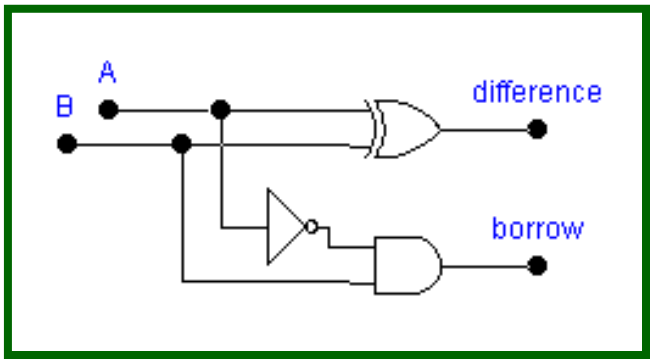
HALF SUBTRACTOR

Subtracts LSD column in binary subtraction

Logic Symbol:



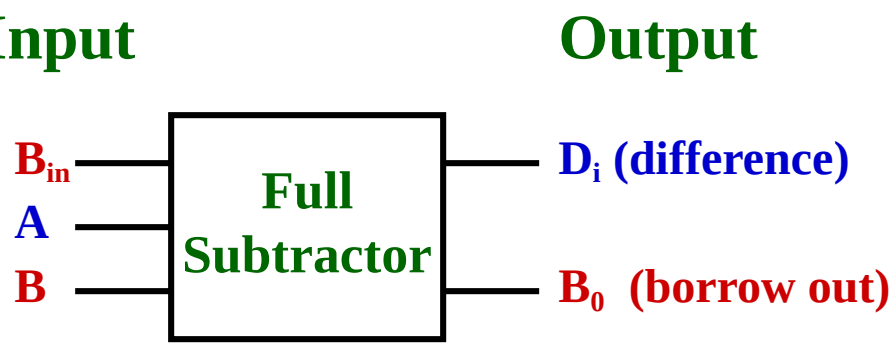
Logic Diagram:



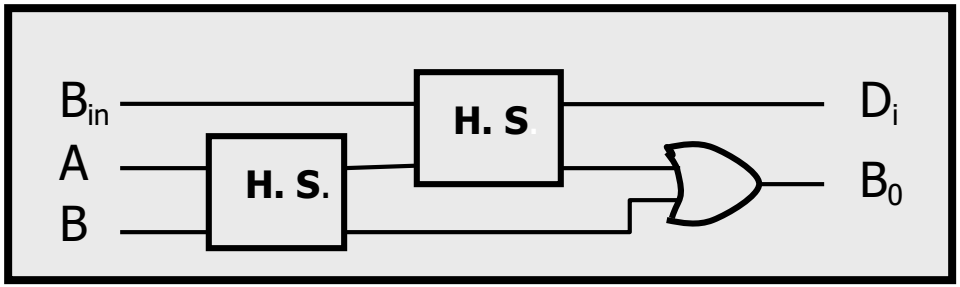
FULL SUBTRACTOR

Used for subtracting binary place values other than the 1s place

Logic Symbol:

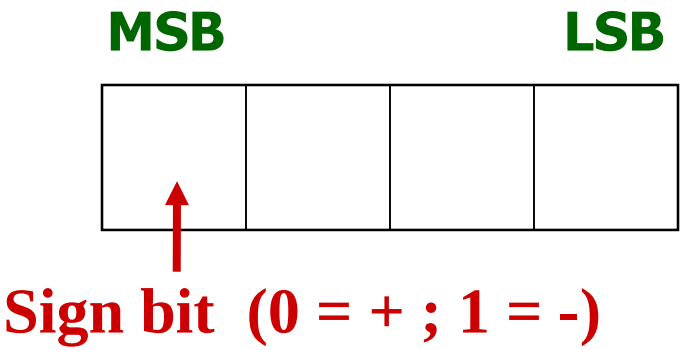


Logic Diagram:



2s COMPLEMENT NOTATION

- **2s complement representation - widely used in microprocessors.**
- **Represents *sign* and *magnitude***



Decimal:	**+7**	**+4**	**+1**	**0**	**-1**	**-4**	**-7**
2s Complement:	**0111**	**0100**	**0001**	**0000**	**1111**	**1100**	**1001**

2s COMPLEMENT - CONVERSIONS

- **Converting positive numbers to 2s complement:**
  - **Same as converting to binary**
- **Converting negative numbers to 2s complement:**

Decimal to 2s Complement

- 4 (decimal)

↓ **Convert decimal to binary**

0 1 0 0

↓ **1s complement**

1 0 1 1

↓ **Add 1**

- 4 = 1 1 0 0 (2s Complement)

2s Complement to Binary

1 1 0 0 (2s C)

↓ **1s complement**

0 0 1 1

↓ **Add 1**

0 1 0 0 (Binary)

ADDING/SUBTRACTING IN 2s COMPLEMENT

2s complement notation makes it possible to add and subtract signed numbers

(Decimal)	2s Complement	
(- 1)	1 1 1 1	
+ (- 2)	+ 1 1 1 0	
(- 3)	$\boxed{1}$ 1 1 0 1	2s complement

Discard

(+1)	0 0 0 1	
+ (- 3)	+ 1 1 0 1	
(- 2)	1 1 1 0	2s complement

TEST

Add the following 2s complement numbers:

$$
\begin{array}{r}
(+5) \\
+\ (-4) \\
\hline
(+1)
\end{array}
\qquad
\begin{array}{r}
0\ 1\ 0\ 1 \\
+\ 1\ 1\ 0\ 0 \\
\hline
\boxed{1}\ 0\ 0\ 0\ 1
\end{array}
$$

Discard

EXAMPLE 3-20 Unsigned Binary Subtraction by 2s Complement Addition

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction $X - Y$ and $Y - X$ using 2s complement operations. We have

$$
\begin{aligned}
X &= 1010100 \\
\text{2s complement of } Y &= 0111101 \\
\text{Sum} &= 10010001 \\
\text{Discard end carry } 2^7 &= -\underline{10000000} \\
\textit{Answer: } X - Y &= 0010001
\end{aligned}
$$

$$
\begin{aligned}
Y &= 1000011 \\
\text{2s complement of } X &= \underline{0101100} \\
\text{Sum} &= 1101111
\end{aligned}
$$

There is no end carry.

Answer: $Y - X = -(\text{2s complement of } 1101111) = -0010001.$

Binary Subtractor

- **Remember**
  - **You need to take 2's complement to represent negative numbers**
  - **A-B**
    - ✓ **Take 2's complement of B and add it to A**
      - **First take 1's complement and add 1**

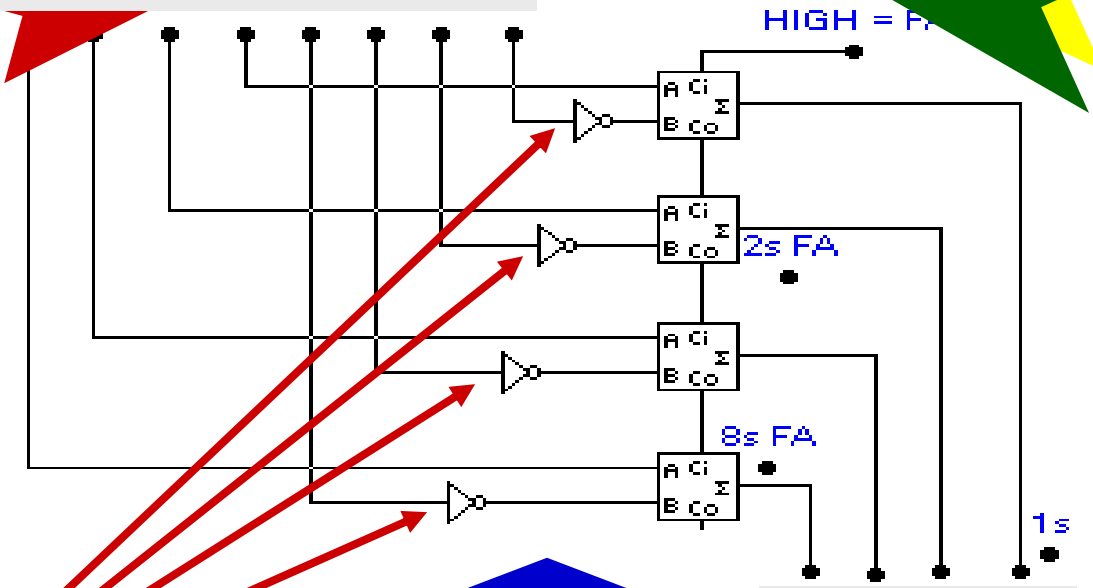
PARALLEL SUBTRACTOR USING FULL ADDERS

Binary numbers to subtracted are input

1 0 0 1 - 0 1 1 1

HIGH at Carry in input acts like adding +1 to a 1s C number to form the 2s complement.

1sC is formed by four inverters.



Inverters

0 0 1 0

The result (difference) of the subtraction problem will appear here.

Note the us

Also notice the addition of four inverters on the B inputs to the FAs

4-Bit Adder and Subtractor

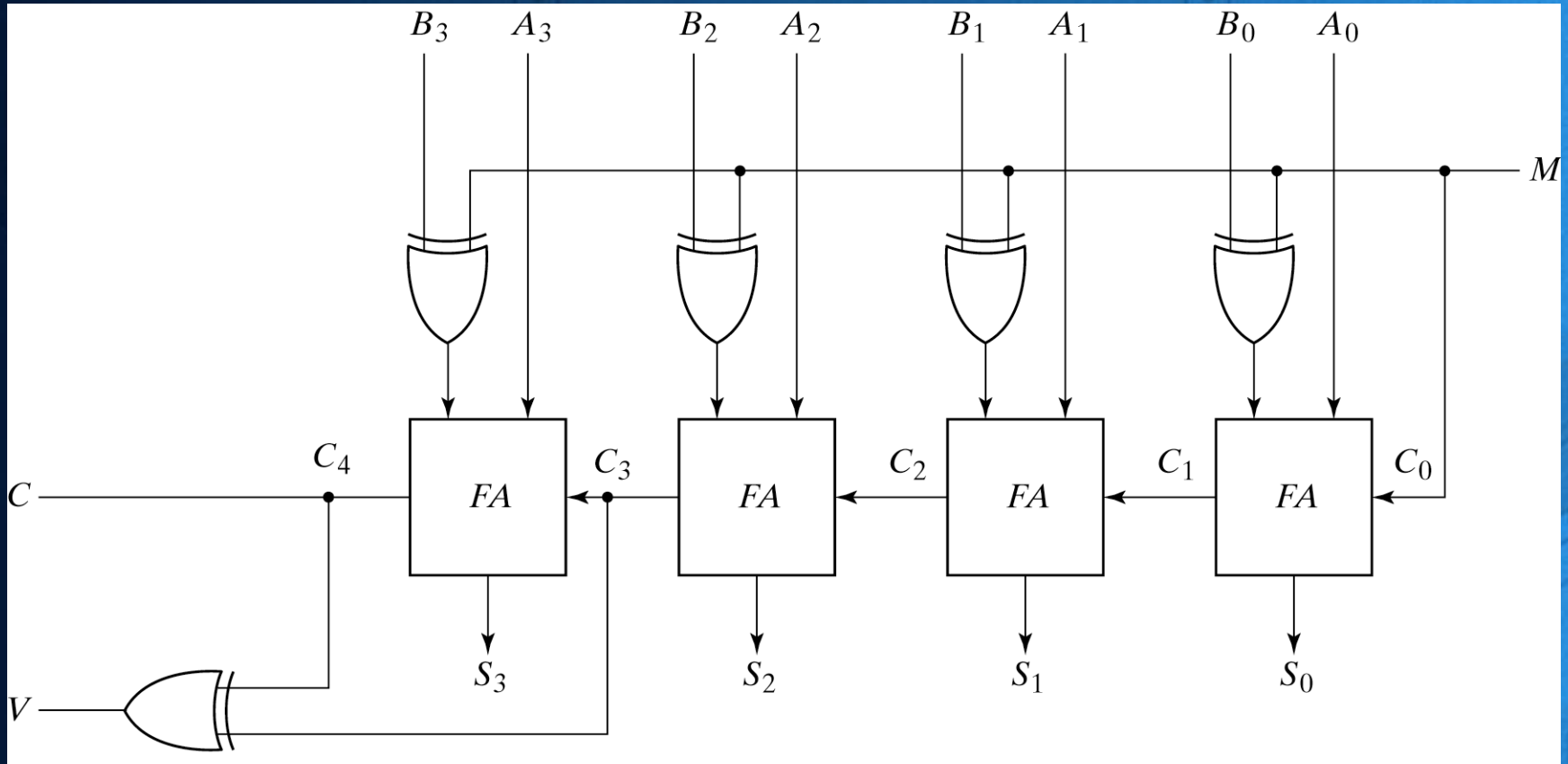


Fig. 4-13 4-Bit Adder Subtractor

$$M = 0(Adder)$$
$$M = 1(Subtractor)$$
$$V(Overflow)$$

From course text book

Signed Integers

- **Positive numbers and zero can be represented by unsigned n-digit, radix *r* numbers. We need a representation for negative numbers.**
- **To represent a sign (+ or –) we need exactly one more bit of information (1 binary digit gives $2^1 = 2$ elements which is exactly what is needed).**
- **Since computers use binary numbers, by convention, the most significant bit is interpreted as a sign bit:**

$$s\ a_{n-2} \ldots a_2 a_1 a_0$$

where:

s = 0 for Positive numbers

s = 1 for Negative numbers

and a_i = 0 or 1 represent the magnitude in some form.

Signed Integer Representations

- ***Signed-Magnitude*** **– here the n – 1 digits are interpreted as a positive magnitude.**
- ***Signed-Complement*** **– here the digits are interpreted as the rest of the complement of the number. There are two possibilities here:**
  - ***Signed 1's Complement***
    - **Uses 1's Complement Arithmetic**
  - ***Signed 2's Complement***
    - **Uses 2's Complement Arithmetic**

Signed Integer Representation Example

- r =2, n=3

Number	Sign -Mag.	1's Comp.	2's Comp.
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
– 0	100	111	—
– 1	101	110	111
– 2	110	101	110
– 3	111	100	101
– 4	—	—	100

Signed Integer Representation

TABLE 3-13
Signed Binary Numbers

Decimal	Signed 2s Complement	Signed Magnitude
+ 7	0111	0111
+ 6	0110	0110
+ 5	0101	0101
+ 4	0100	0100
+ 3	0011	0011
+ 2	0010	0010
+ 1	0001	0001
+ 0	0000	0000
− 0	—	1000
− 1	1111	1001
− 2	1110	1010
− 3	1101	1011
− 4	1100	1100
− 5	1011	1101
− 6	1010	1110
− 7	1001	1111
− 8	1000	—

Signed-Magnitude Arithmetic

- **If the parity of the two signs is 0:**
  1. Add the magnitudes.
  2. Check for overflow (a carry out of the MSB)
  3. The sign of the result is the same as the sign of the first operand.
- **If the parity of the two signs is 1:**
  1. Subtract the second magnitude from the first.
  2. If a borrow occurs:
     - take the two's complement of result
     - and make the result sign the complement of the sign of the first operand.
  3. Overflow will never occur.

Sign-Magnitude Arithmetic Examples

- **Example 1:**

$$\begin{array}{r} 0010 \\ +\underline{0101} \end{array}$$

- **Example 2:**

$$\begin{array}{r} 0010 \\ +\underline{1101} \end{array}$$

- **Example 3:**

$$\begin{array}{r} 1010 \\ -\underline{0101} \end{array}$$

Signed-Complement Arithmetic

- **Addition:**

  1. Add the numbers including the sign bits, discarding a carry out of the sign bits (2's Complement), or using an end-around carry (1's Complement).

  2. If the sign bits were the same for both numbers and the sign of the result is different, an overflow has occurred.

  3. The sign of the result is computed in step 1.

- **Subtraction:**

  Form the complement of the number you are subtracting and follow the rules for addition.

Signed 2's Complement Examples

- **Example 1:**

$$\begin{array}{r} 1101 \\ +\underline{0011} \end{array}$$

- **Example 2:**

$$\begin{array}{r} 1101 \\ -\underline{0011} \end{array}$$

Signed Binary Addition

EXAMPLE 3-21 Signed Binary Addition Using 2s Complement

+6	00000110	−6	11111010	+6	00000110	−6	11111010
+13	00001101	+13	00001101	−13	11110011	−13	11110011
+19	00010011	+7	00000111	−7	11111001	−19	11101101

In each of the four cases, the operation performed is addition, including the sign bits. Any carry out of the sign bit position is discarded, and negative results are automatically in 2s complement form. ■

Signed Binary Subtraction

EXAMPLE 3-22 Signed Binary Subtraction Using 2s Complement

-6	11111010	11111010	$+6$	00000110	00000110
$-(-13)$	$-$ 11110011	$+$ 00001101	$-(-13)$	$-$ 11110011	$+$ 00001101
$+7$		00000111	$+19$		00010011

The end carry is discarded.

Overflow Detection

- *Overflow* occurs if $n + 1$ bits are required to contain the result from an n-bit addition or subtraction
- Overflow can occur for:
  - Addition of two operands with the same sign
  - Subtraction of operands with different signs
- Signed number overflow cases with correct result sign

$$\begin{array}{rrrr} 0 & 0 & 1 & 1 \\ +\underline{0} & -\underline{1} & -\underline{0} & +\underline{1} \\ 0 & 0 & 1 & 1 \end{array}$$

- Detection can be performed by examining the result signs which should match the signs of the top operand

Overflow Detection

- **Signed number cases with carries C_n and C_{n-1} shown for correct result signs:**

$$\begin{array}{cc|cc|cc|cc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & & 0 & & 1 & & 1 & \\ +\,0 & & -\,1 & & -\,0 & & +\,1 & \\ \hline 0 & & 0 & & 1 & & 1 & \end{array}$$

- **Signed number cases with carries shown for erroneous result signs (indicating overflow):**

$$\begin{array}{cc|cc|cc|cc} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & & 0 & & 1 & & 1 & \\ +\,0 & & -\,1 & & -\,0 & & +\,1 & \\ \hline 1 & & 1 & & 0 & & 0 & \end{array}$$

- **Simplest way to implement overflow $V = C_n + C_{n-1}$**

Overflow Detection

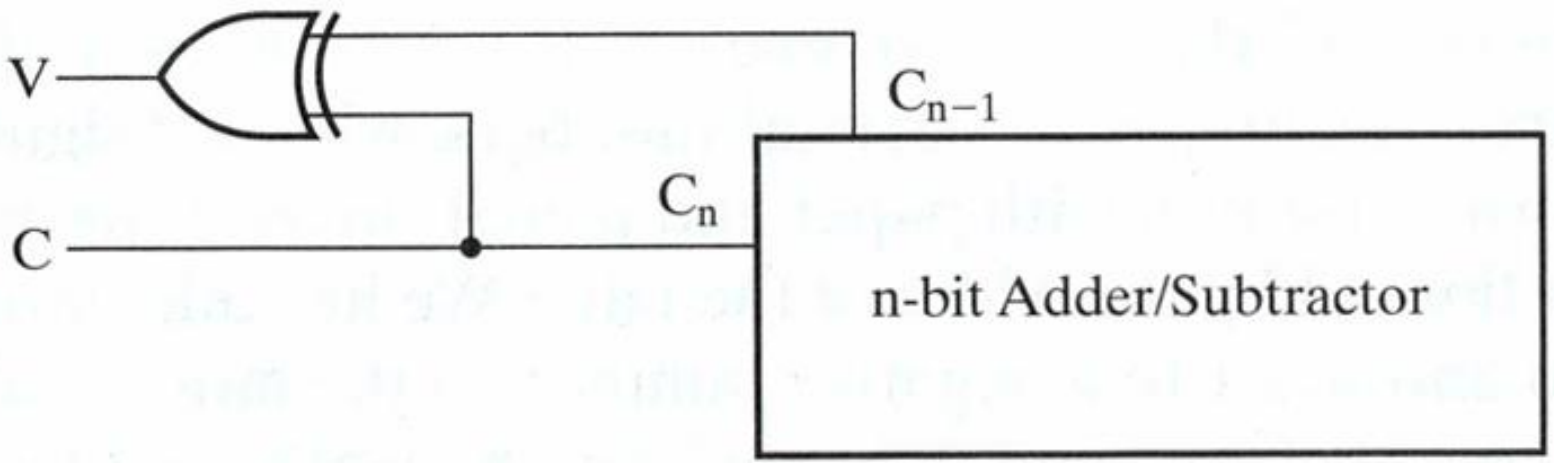


FIGURE 3-46
Overflow Detection Logic for Addition and Subtraction

BINARY MULTIPLICATION

Example: **Multiply the binary numbers 111 and 101.**

				1	1	1	**Multiplicand**
			x	1	0	1	**Multiplier**
				1	1	1	**1st partial product**
			0	0	0		**2nd partial product**
		1	1	1			**3rd partial product**
	1	0	0	0	1	1	**Product**

111 x 101 can also be calculated: 111 + 111 + 111 + 111 + 111

TEST

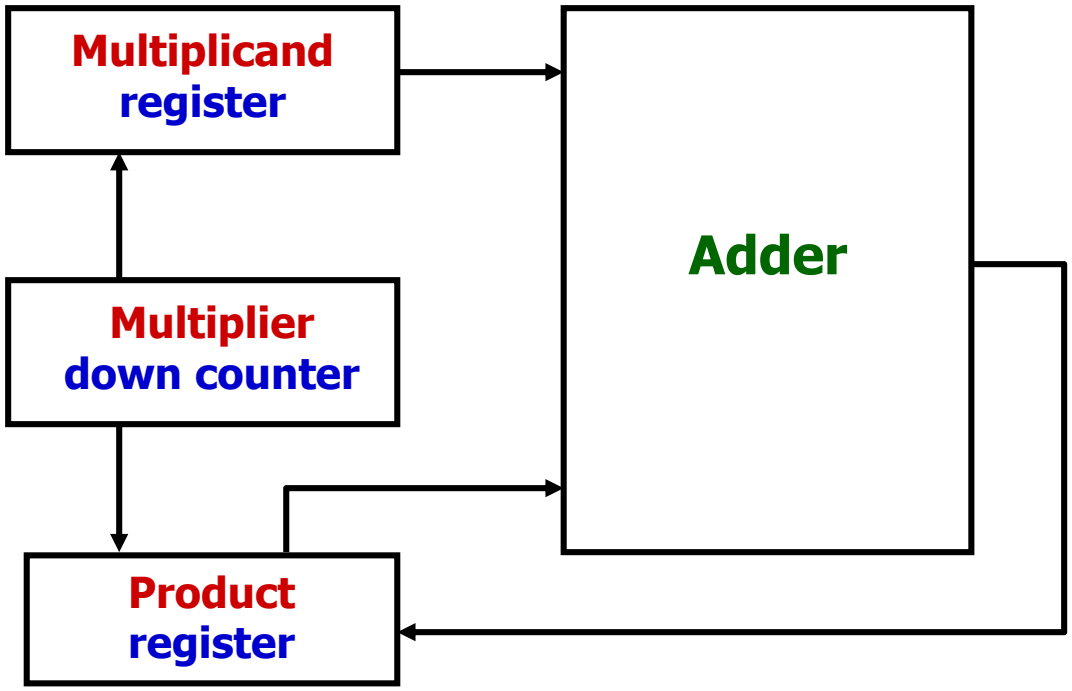
Multiply the binary numbers 101 and 100.

```
      1 0 1
    x 1 0 0
  ---------
      0 0 0
    0 0 0
  1 0 1
  ---------
  1 0 1 0 0
```

BINARY MULTIPLIERS

Binary multiplier circuits - utilize repeated addition.

Block Diagram:



Binary Multiplier

		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0

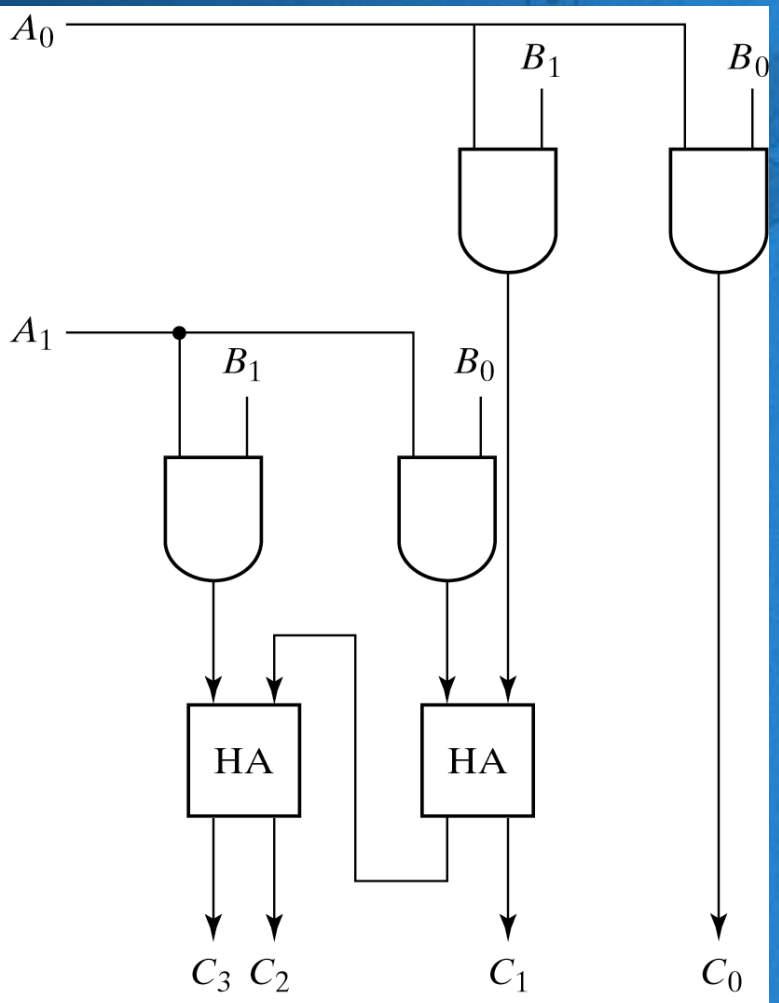


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

From course text book