King Saud University

جامعة الملك سعود

CSC 220: Computer Organization

Unit 5
COMBINATIONAL CIRCUITS-1

Prepared by:

Md Saiful Islam, PhD

Department of Computer Science

College of Computer and Information Sciences

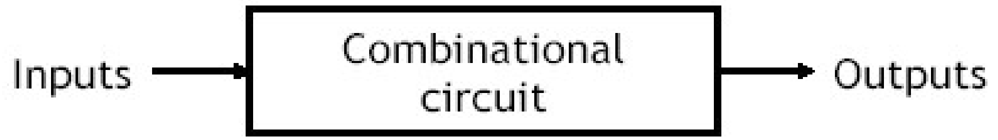
Overview

- Introduction to Combinational Circuits
- Adder
- Ripple Carry Adder
- Subtraction
- Adder/Subtractor

Chapter-3

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

Combinational circuits



- So far we've only worked with combinational circuits, where applying the same inputs always produces the same outputs.
  - This corresponds to a mathematical function, where every input has a single, unique output.
  - In programming terminology, combinational circuits are similar to "functional programs" that do not contain variables and assignments.
- Such circuits are comparatively easy to design and analyze.

Binary addition by hand

- You can add two binary numbers one column at a time starting from the right, just like you add two decimal numbers.
- But remember it's binary. For example, 1 + 1 = 10 and you have to carry!

The initial carry in is implicitly 0

	1	1	1	0		Carry in
		1	0	1	1	Augend
+		1	1	1	0	Addend
	1	1	0	0	1	Sum

most significant bit (MSB)

least significant bit (LSB)

Adder

- Design an Adder for 1-bit numbers?
- **1. Specification:**

  2 inputs (X,Y)

  2 outputs (C,S)

Adder …

- Design an Adder for 1-bit numbers?
- **1. Specification:**

  2 inputs (X,Y)

  2 outputs (C,S)
- **2. Formulation:**

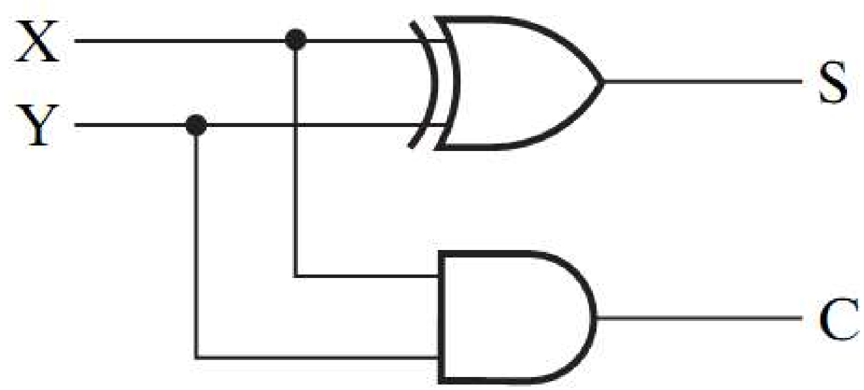
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Adder …

- Design an Adder for 1-bit numbers?
- **1. Specification:**
  2 inputs (X,Y)
  2 outputs (C,S)
- **2. Formulation:**

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

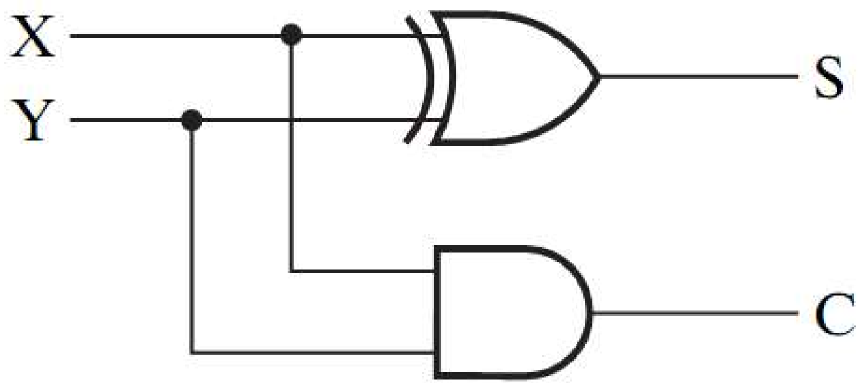
3. Optimization/Circuit



Half Adder …

- This adder is called a Half Adder
- **Q: Why?**

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



Full Adder

- A combinational circuit that adds 3 input bits to generate a Sum bit and a Carry bit

- A truth table and sum of minterm equations for C and S are shown below.

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

$0 + 1 + 1 = 10$

$1 + 1 + 1 = 11$

$C(X,Y,Z) = \Sigma m(3,5,6,7)$

$S(X,Y,Z) = \Sigma m(1,2,4,7)$

Full Adder

- A combinational circuit that adds 3 input bits to generate a Sum bit and a Carry bit

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

Sum

X \ YZ	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$S = X'Y'Z + X'YZ' + XY'Z' + XYZ$

$= X \oplus Y \oplus Z$

Carry

X \ YZ	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C = XY + YZ + XZ$

Full Adder = 2 Half Adders

Manipulating the Equations:

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

$$C = XY + XZ + YZ$$

$$= XY + XZ(Y + Y') + YZ(X + X')$$

$$= XY + XYZ + XY'Z + X'YZ + X\cancel{Y}Z$$

$$= XY(1 + Z) + Z(XY' + X'Y)$$

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

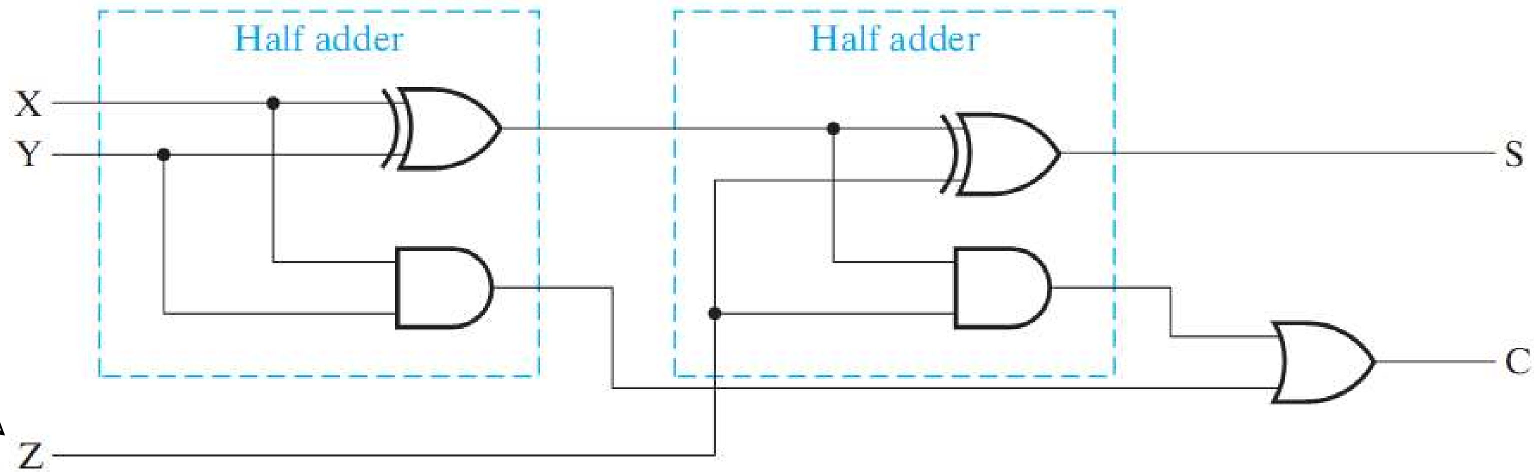
Full Adder = 2 Half Adders

Manipulating the Equations:

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

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

Think of Z as a carry in



Src: Mano's Book

n-bit Adder

- How to build an adder for n-bit numbers?
  - Example: 4-Bit Adder
    - Inputs ?
    - Outputs ?
    - What is the size of the truth table?
    - How many functions to optimize?

n-bit Adder …

- How to build an adder for n-bit numbers?
  - Example: 4-Bit Adder
    - Inputs ? 9 inputs
    - Outputs ? 5 outputs
    - What is the size of the truth table? 512 rows!
    - How many functions to optimize? 5 functions

Binary Parallel Adder

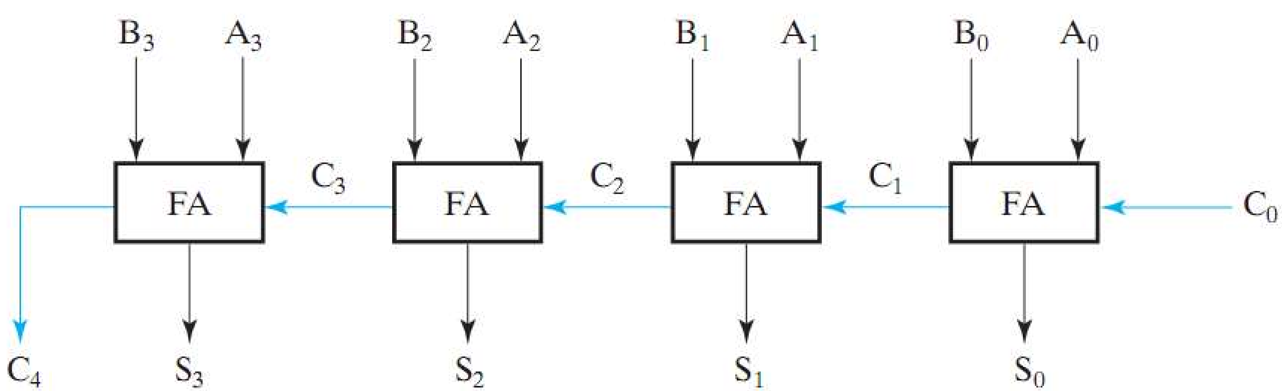
- To add n-bit numbers:
- Use n Full-Adders in parallel
- The carries propagates as in addition by hand
- Use Z in the circuit as a C_{in}

Example

$$
\begin{array}{r}
0\ 1\ 0\ 0\ 0 \\
0\ 1\ 0\ 1 \\
\underline{0\ 1\ 1\ 0} \\
1\ 0\ 1\ 1
\end{array}
$$

Binary Parallel Adder ..

- To add n-bit numbers:
- Use n Full-Adders in parallel
- The carries propagates as in addition by hand



C	0 1 0 0 0
A	0 1 0 1
B	0 1 1 0
S	1 0 1 1

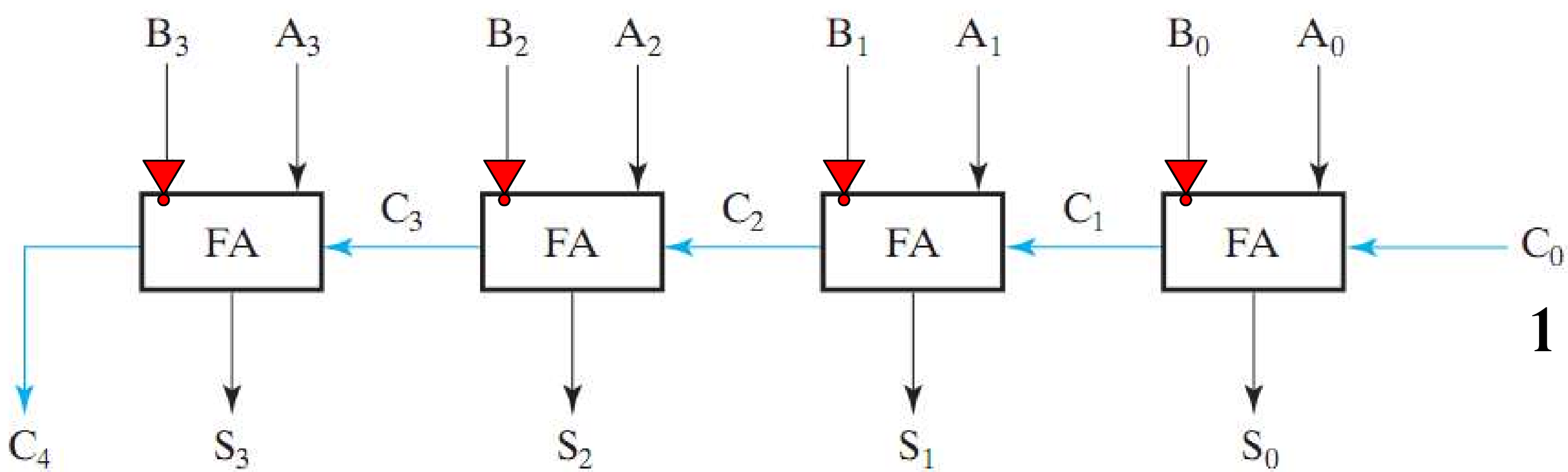
This adder is called *ripple carry adder*

Src: Mano's Book

Subtraction (2's Complement)

- How to build a subtractor using 2's complement?

$$S = A - B$$
$$= A + (-B)$$



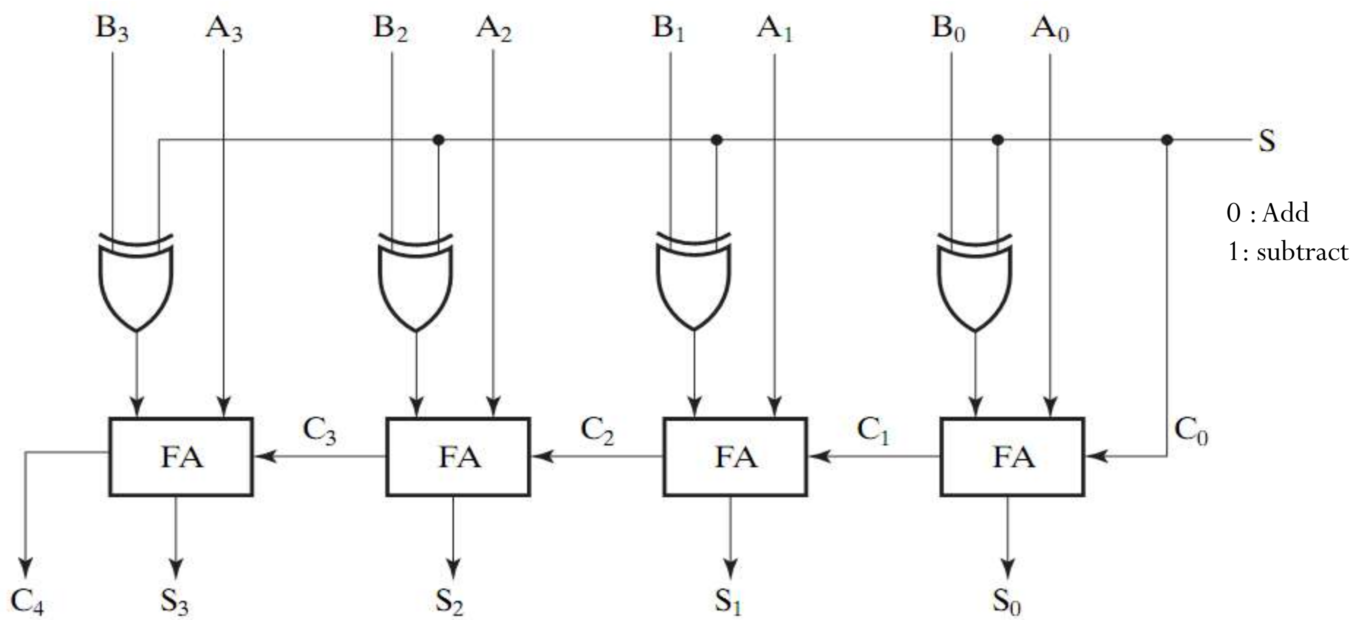
Src: Mano's Book

A	0 1 0 1
B	0 1 1 0

C	0 0 0 0 0
A	0 1 0 1
-B	1 0 1 0
S	1 1 1 1

Adder-Subtractor

- How to build a circuit that performs both addition and subtraction?

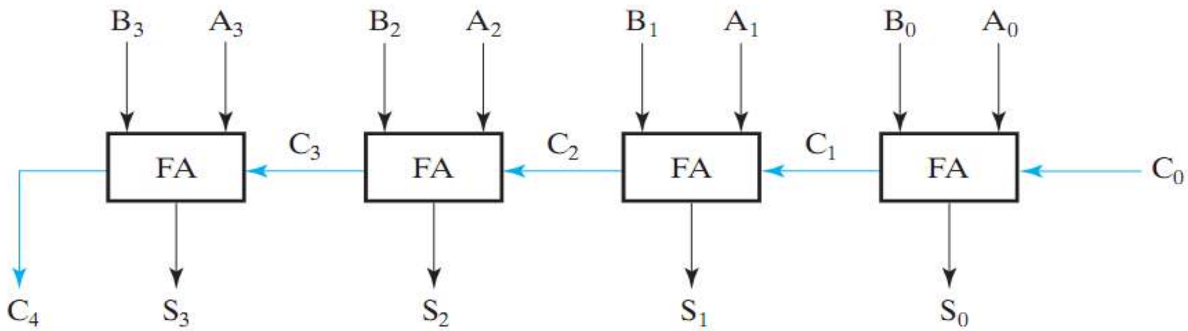


C	0	1	0	0	0
A		0	1	0	1
B		0	1	1	0
S		1	0	1	1

Src: Mano's Book

Using full adders and XOR we can build an Adder/Subtractor!

Carry Look Ahead Adder



- How to reduce propagation delay of ripple carry adders?
- **Carry look ahead adder:** All carries are computed as a function of C_0 (independent of n !)
- It works on the following standard principles:
  - A carry bit is generated when both input bits Ai and Bi are 1, or
  - When one of input bits is 1, and a carry in bit exists

Carry bits → Cn Cn-1.......Ci..........C2C1C0

An-1.......Ai..........A2A1A0

Bn-1.......Bi..........B2B1B0

Carry Out →

Sn Sn-1.......Si..........S2S1S0

Detecting signed overflow

- The easiest way to detect signed overflow is to look at all the sign bits.

$$
\begin{array}{rr}
 & 0\,1\,0\,0 \\
+ & 0\,1\,0\,1 \\
\hline
0 & 1\,0\,0\,1
\end{array}
\qquad
\begin{array}{rr}
 & (+4) \\
+ & (+5) \\
\hline
 & (-7)
\end{array}
\qquad\qquad
\begin{array}{rr}
 & 1\,1\,0\,0 \\
+ & 1\,0\,1\,1 \\
\hline
1 & 0\,1\,1\,1
\end{array}
\qquad
\begin{array}{rr}
 & (-4) \\
+ & (-5) \\
\hline
 & (+7)
\end{array}
$$

- Overflow occurs only in the two situations above.
  1. If you add two *positive* numbers and get a *negative* result.
  2. If you add two *negative* numbers and get a *positive* result.
- Overflow can never occur when you add a positive number to a negative number. (Do you see why?)

Overflow

Example1:

$$\begin{array}{rl}
\texttt{0100000} & \\
\texttt{0110101}_2 & (= 53_{10}) \\
\underline{+\texttt{0101010}}_2 & (= 42_{10}) \\
\texttt{1011111}_2 & (=-33_{10})
\end{array}$$

Example2:

$$\begin{array}{rl}
\texttt{1000000} & \\
\texttt{1010101}_2 & (=-43_{10}) \\
\underline{+\texttt{1001010}}_2 & (=-54_{10}) \\
\texttt{0011111}_2 & (= 31_{10})
\end{array}$$

Example3:

$$\begin{array}{rl}
\texttt{1100000} & \\
\texttt{0110101}_2 & (= 53_{10}) \\
\underline{+\texttt{1101010}}_2 & (=-22_{10}) \\
\texttt{0011111}_2 & (= 31_{10})
\end{array}$$

Example4:

$$\begin{array}{rl}
\texttt{0000000} & \\
\texttt{0010101}_2 & (= 21_{10}) \\
\underline{+\texttt{0101010}}_2 & (= 42_{10}) \\
\texttt{0111111}_2 & (= 63_{10})
\end{array}$$

Detecting Sign Overflow …