Chapter 5
Number Representation & Arithmetic Circuits

Outline

- Review of Number Systems
- Adders
  - Ripple carry
  - Carry Lookahead (fast adders)
  - Carry Select
- Combinational Multipliers
- Arithmetic and Logic Unit (ALU)
- General Logic Function Units
- **READING**: Brown’s 5.1-5.4, 5.6-5.8

Numbers in different radix systems

used by human

used by Computers

Decimal	Binary	Octal	Hexadecimal
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10
17	10001	21	11
18	10010	22	12

Figure 5.1 Numbers in different systems.

HEX

6 7

01100111_2

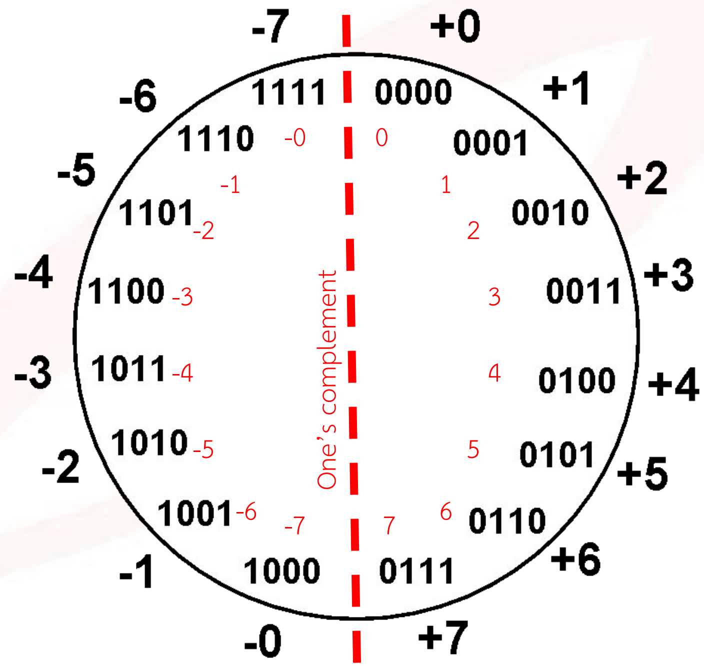
1 4 7

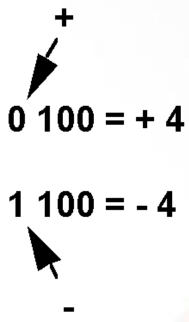
OCT

Review of Number Systems

- Differences in negative numbers
  - Three major schemes:
    - sign magnitude
    - one’s complement
    - two’s complement
- Assumptions:
  - we'll assume a 4 bit machine word
  - 16 different values can be represented
  - roughly half are positive, half are negative

Sign Magnitude/One's Complement Representation

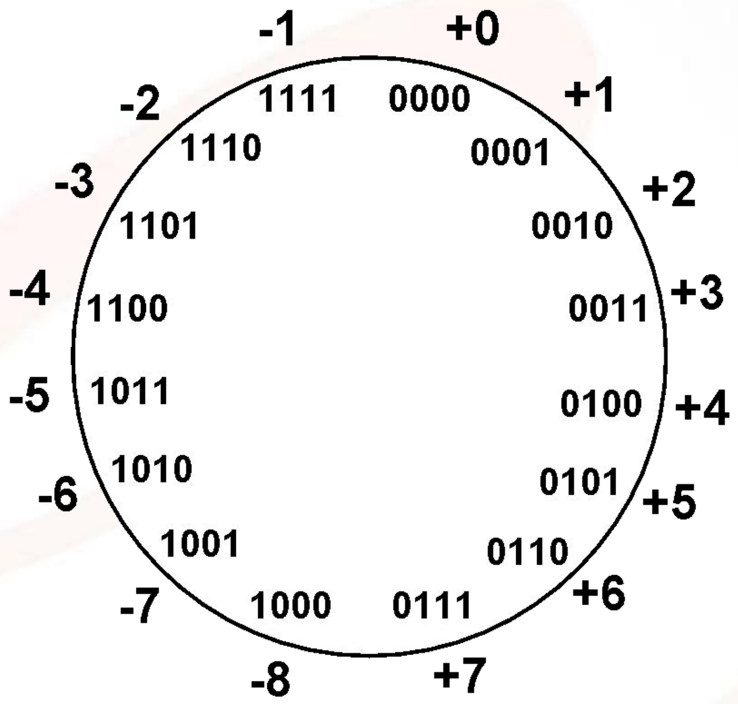




- High order bit is sign: 0 = positive (or zero), 1 = negative
- Three low order bits is the magnitude: 0 (000) thru 7 (111)
- Number range for n bits = +/-2^{n-1}-1
- Two representations for 0
- Cumbersome addition/subtraction
- Must compare magnitudes to determine sign of result

Two’s Complement Representation

like 1's comp except shifted one position clockwise



+

0 100 = + 4

1 100 = - 4

-

- Only one representation for 0
- numbers of positives = number of negatives

Two's Complement Number System

$N^* = 2^n - N$

Example: Two's complement of 7

$$
\begin{array}{rcl}
2^4 & = & 10000 \\
\text{sub } 7 & = & \underline{0111} \\
 & & 1001 \ [-7]
\end{array}
$$

Example: Two's complement of -7

$$
\begin{array}{rcl}
2^4 & = & 10000 \\
\text{sub } -7 & = & \underline{1001} \\
 & & 0111 \ [7]
\end{array}
$$

Shortcut method:

Two's complement = bitwise complement + 1

0111 -> 1000 + 1 -> 1001 (representation of -7)

1001 -> 0110 + 1 -> 0111 (representation of 7)

Addition and Subtraction of Numbers

Sign Magnitude

result sign bit is the same as the operands' sign

4	0100	-4	1100
+ 3	0011	+ (-3)	1011
7	0111	-7	1111

when signs differ, operation is subtract, sign of result depends on sign of number with the larger magnitude

4	0100	-4	1100
- 3	1011	+ 3	0011
1	0001	-1	1001

ELECTRICAL
Engineering KMITNB

Two’s Complement Addition and Subtraction

Two’s Complement Calculations

4	0100	-4	1100
+ 3	0011	+ (-3)	1101
7	0111	-7	11001

4	0100	-4	1100
- 3	1101	+ 3	0011
1	10001	-1	1111

Simpler addition scheme makes **two’s complement the most common choice for integer number systems** within digital systems

Arithmetic Overflow

- The result of addition or subtraction is supposed to fit within the significant bits used to represent the numbers
- If *n* bits are used to represent signed numbers, then the result must be in the range -2^{n-1} to $+2^{n-1}-1$
- If the result does not fit in this range, we say

  that arithmetic overflow has occurred
- To insure correct operation of an arithmetic circuit, it is important to be able to detect the occurrence of overflow

Arithmetic Overflow

For 4-bit numbers, there are 3 significant bits and the sign bit

(+7)	0111		(-7)	1001
+	+		+	+
(+2)	0010		(+2)	0010
(+9)	1001		(-5)	1011
	c4=0			c4=0
	c3=1			c3=0

$$\text{overflow} = c_{n-1} \oplus c_n$$

(+7)	0111		(-7)	1001
+	+		+	+
(-2)	1110		(-2)	1110
(+5)	10101		(-9)	10111
	c4=1			c4=1
	c3=1			c3=0

If the numbers have different signs, no overflow can occur

Circuits for Binary Half-Adder

Half Adder

With two's complement numbers, addition is sufficient

Ai	Bi	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

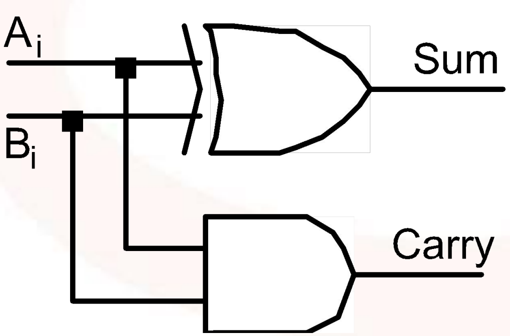
Bi \ Ai	0	1
0	0	1
1	1	0

Bi \ Ai	0	1
0	0	0
1	0	1

$$\text{Sum} = \overline{Ai}\,Bi + Ai\,\overline{Bi}$$

$$= Ai \oplus Bi$$

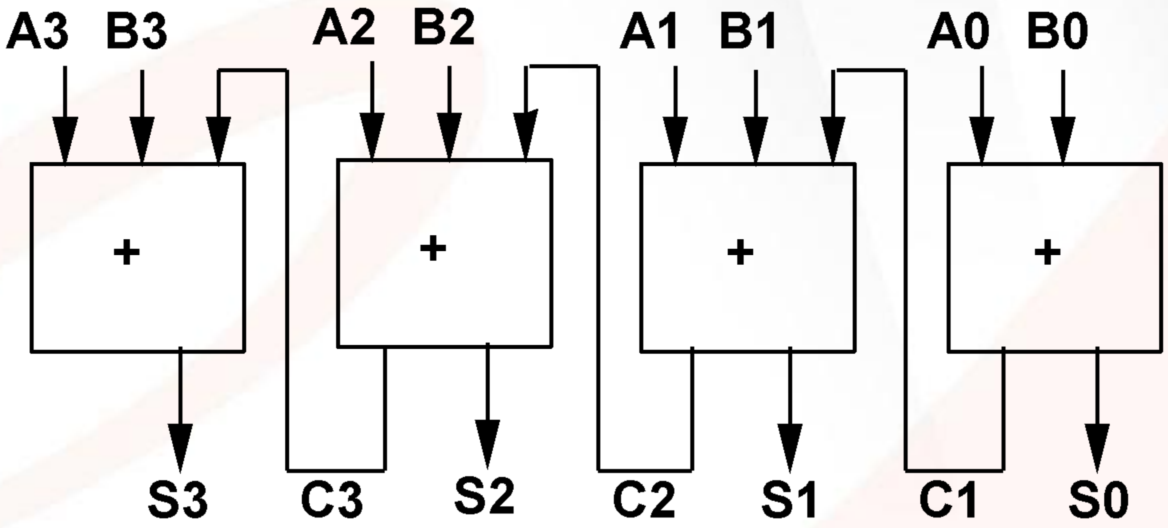
$$\text{Carry} = Ai\,Bi$$



Half-adder Schematic

Full Adder

Cascaded Multi-bit Adder



usually interested in adding more than two bits

this motivates the need for the full adder

Full Adder

A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S

CI \ A B	00	01	11	10
0	0	1	0	1
1	1	0	1	0

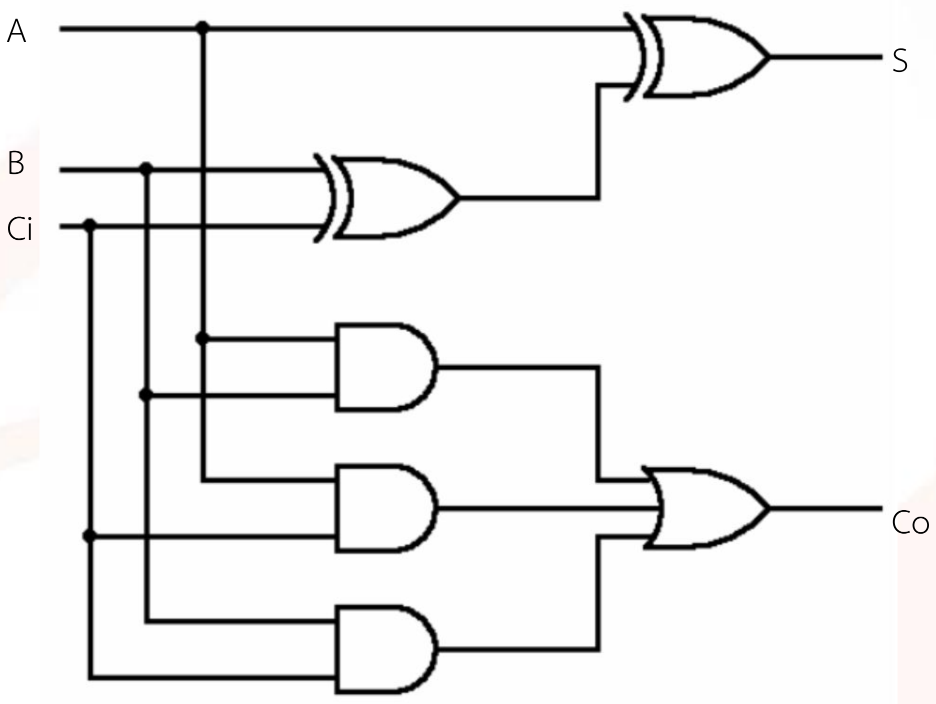
CO

CI \ A B	00	01	11	10
0	0	0	1	0
1	0	1	1	1

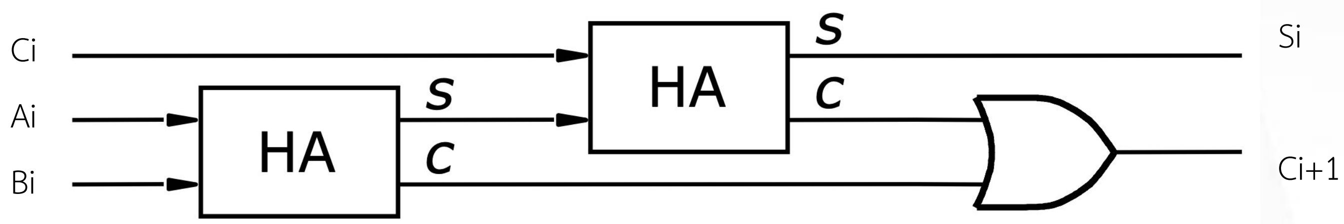
$S = CI \oplus A \oplus B$

$CO = B\,CI + A\,CI + A\,B = CI\,(A + B) + A$

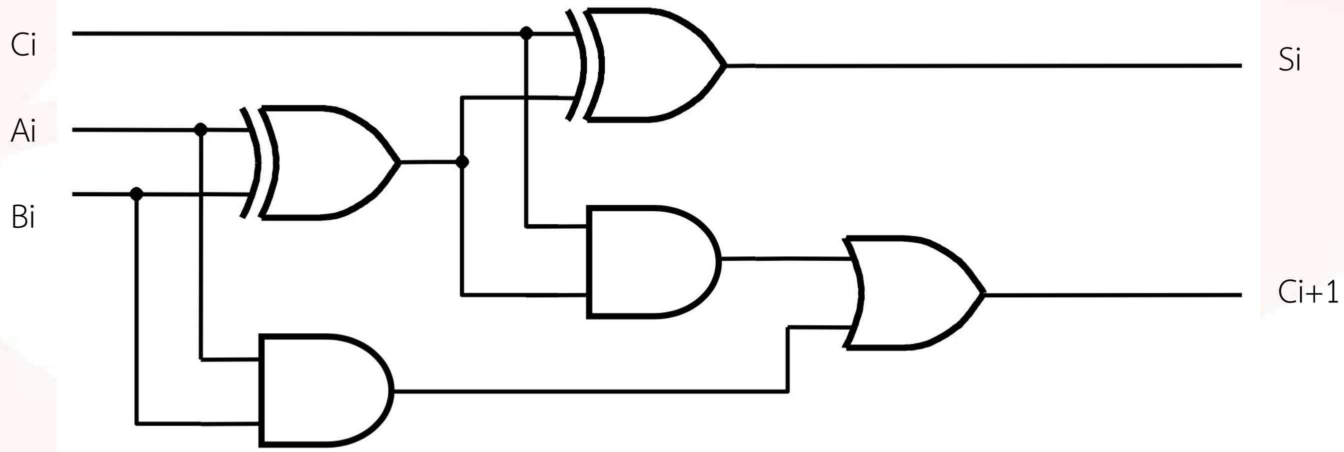
Full Adder Circuit



Full Adder Circuit (decomposed)

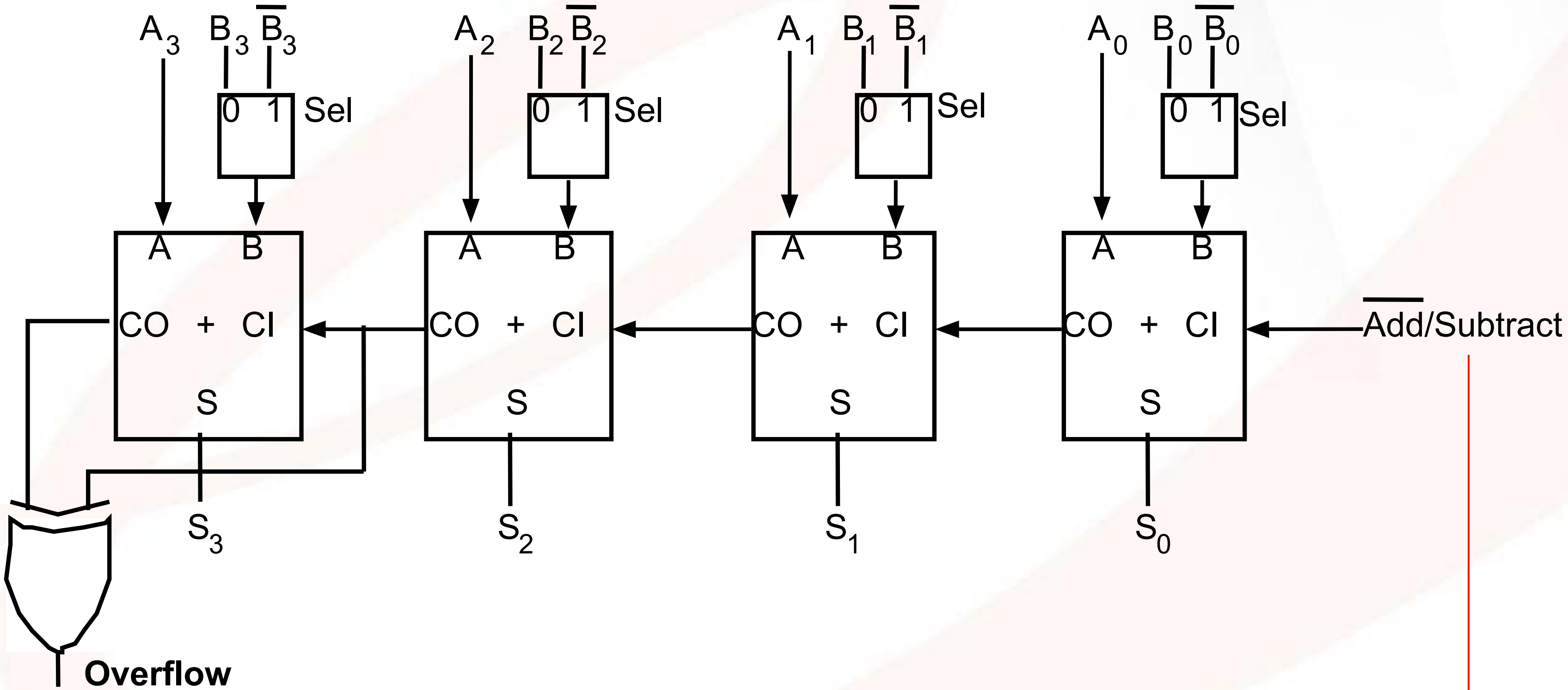


Block diagram



Detailed diagram

Adder/Subtractor



A - B = A + (-B) = A + B' + 1

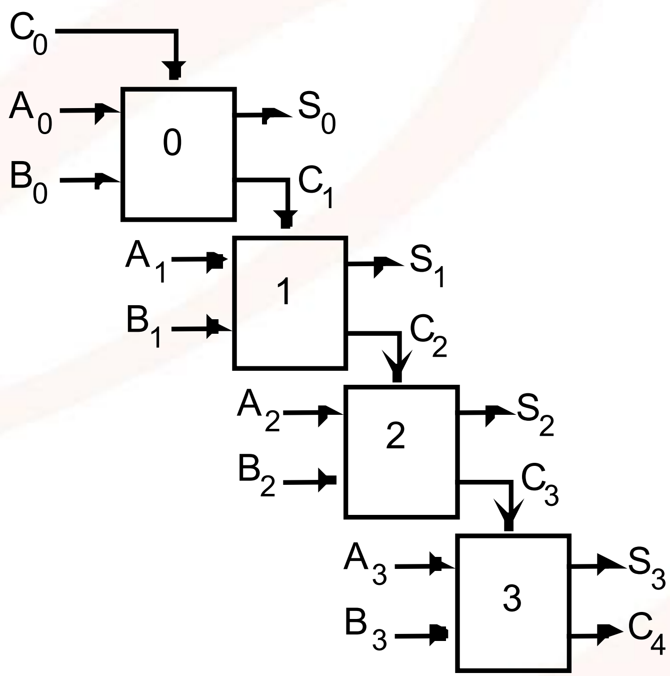
Delay Analysis of Ripple Adder

- Carry out of a single stage can be implemented in 2 gate delays
- For a 16 bit adder, the 16th bit carry is generated after 16 * 2 = 32 gate delays.
- Takes too long - need to investigate FASTER adders!

Carry Lookahead Adder

Critical delay: the propagation of carry from low to high order stages

4 stage adder



1111 + 0001 worst case addition

final sum and carry

Carry Lookahead Logic

Sum and Carry can be re-expressed in terms of generate/propagate:

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

$$= A_i B_i + C_i (A_i + B_i)$$

$$= A_i B_i + C_i (A_i \oplus B_i)$$

$$= G_i + C_i P_i$$

Carry Generate $G_i = A_i B_i$ ***must generate carry when A = B = 1***

Carry Propagate $P_i = A_i \oplus B_i$ ***carry in will equal carry out here***

Carry Lookahead Logic

Reexpress the carry logic as follows:

$$C1 = \underline{G0 + P0\ C0}$$

$$C2 = G1 + P1\ C1 = \underline{G1 + P1\ G0 + P1\ P0\ C0}$$

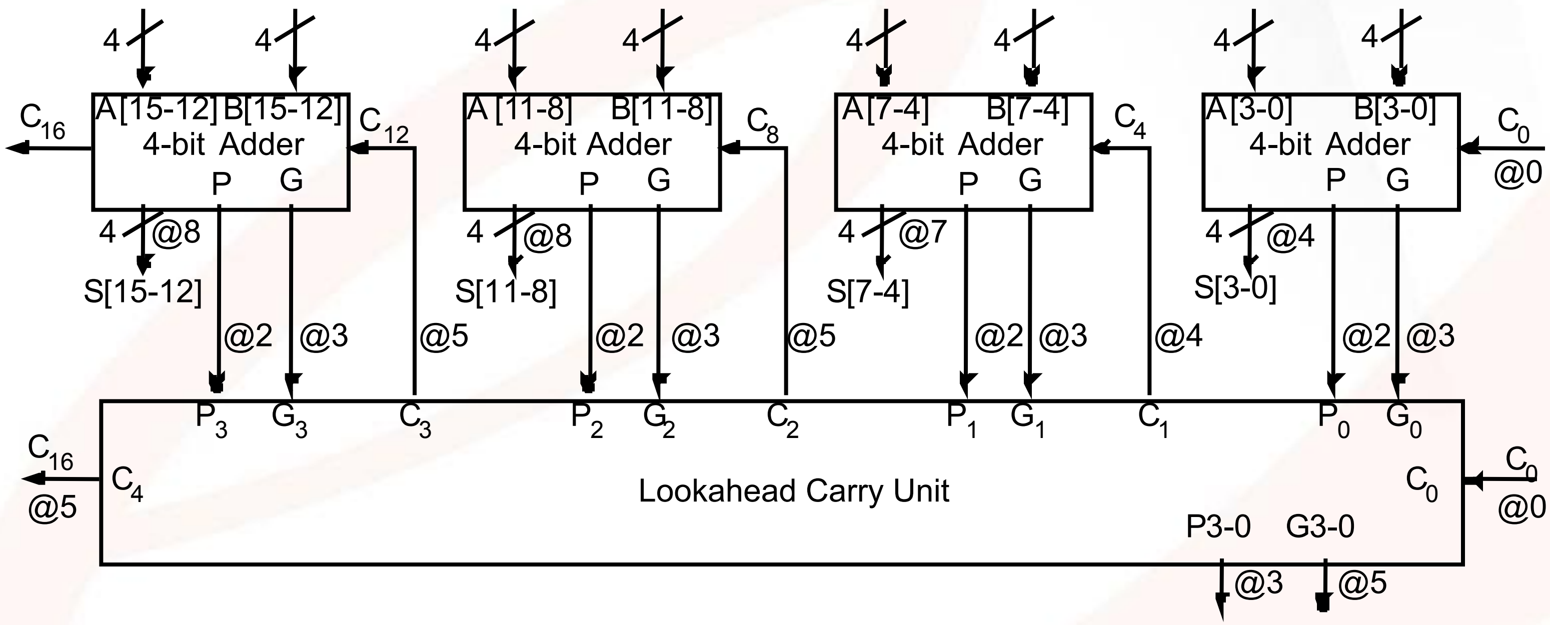
$$C3 = G2 + P2\ C2 = \underline{G2 + P2\ G1 + P2\ P1\ G0 + P2\ P1\ P0\ C0}$$

$$C4 = G3 + P3\ C3 = G3 + P3\ G2 + P3\ P2\ G1 + P3\ P2\ P1\ G0 + P3\ P2\ P1\ P0\ C0$$

- Each of the carry equations can be implemented in a two-level logic network
- Variables are the adder inputs and carry in to stage 0!

Carry Lookahead Logic

Cascaded Carry Lookahead



4 bit adders with internal carry lookahead

second level carry lookahead unit, extends lookahead to 16 bits

Delay Analysis of Carry Lookahead

- Consider a 16-bit adder
- Implemented with four stages of 4-bit adders using carry lookahead
- Carry in to the highest stage is available after 5 gate delays
- Sum from highest stage available at 8 gate delays
- 32 gate delays for a ripple carry adder

Theory of Multiplication

Basic Concept

multiplicand	**1101**	**(13)**
multiplier	*** 1011**	**(11)**
	1101	
	1101	
Partial products	**0000**	
	1101	
	10001111	**(143)**

product of 2 4-bit numbers is an 8-bit number

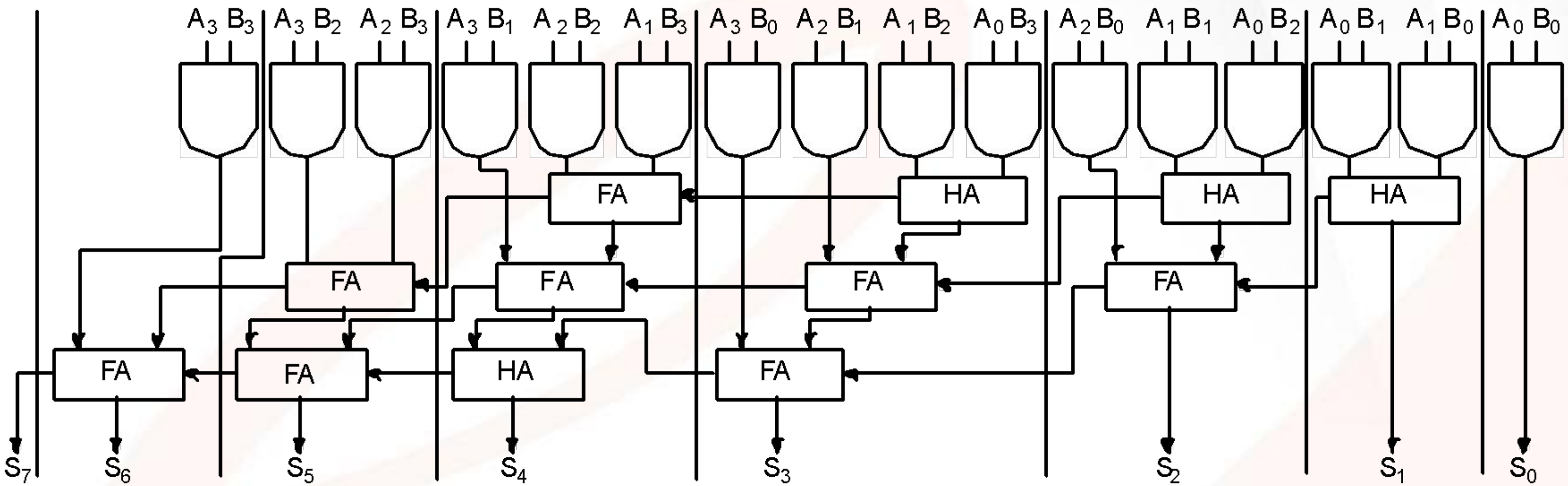
Combinational Multiplier

Partial Product Accumulation

				A3	A2	A1	A0
				B3	B2	B1	B0
				A2 B0	A2 B0	A1 B0	A0 B0
				A2 B1	A1 B1	A0 B1	
			A3 B1	A1 B2	A0 B2		
			A2 B2	A0 B3			
		A3 B2	A1 B3				
		A2 B3					
	A3 B3						
S7	S6	S5	S4	S3	S2	S1	S0

ELECTRICAL
Engineering KMITNB

Partial Product Accumulation



Note use of parallel carry-outs to form higher order sums

12 Adders, if full adders, this is 6 gates each = 72 gates

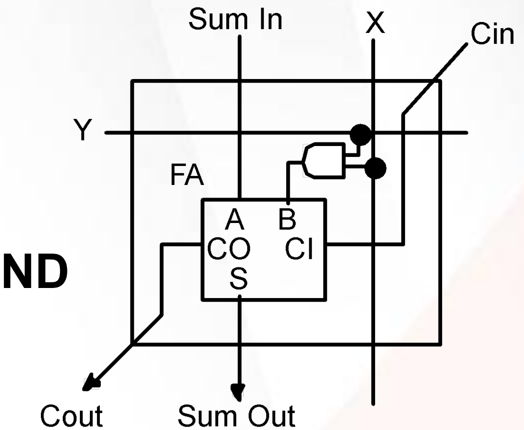
16 gates form the partial products

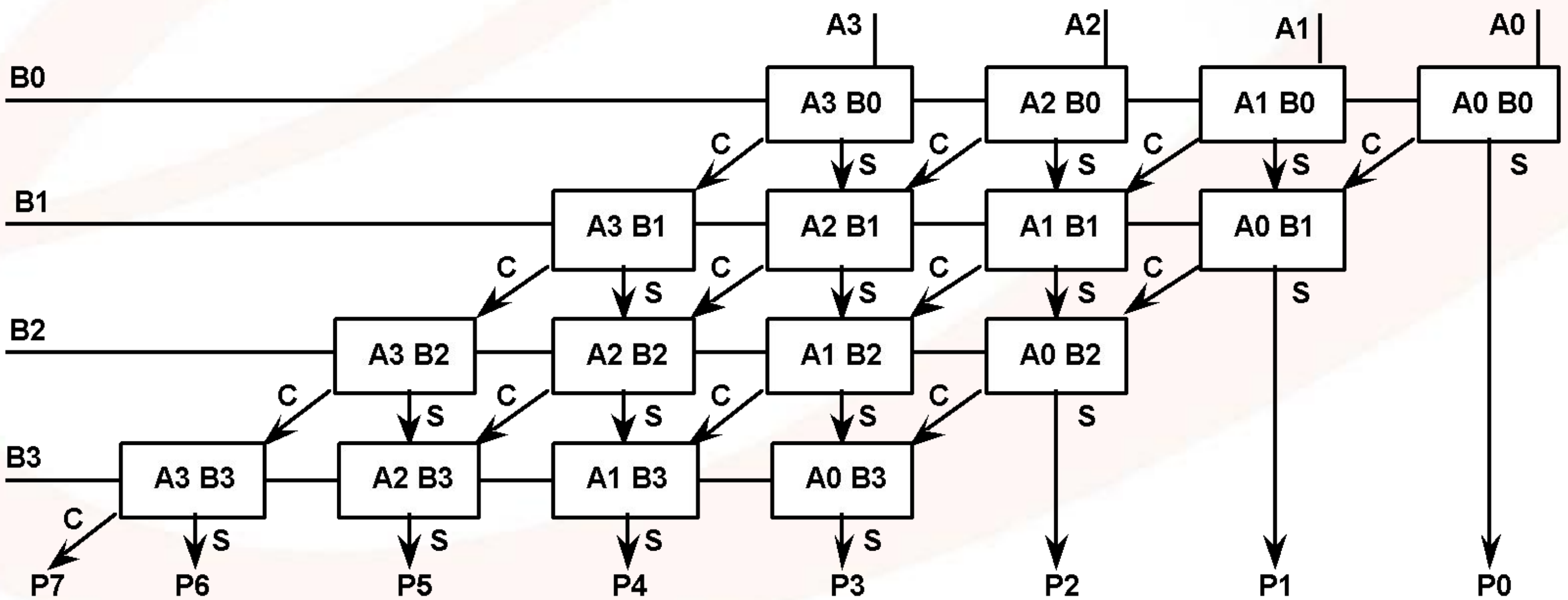
total = 88 gates!

Combinational Multiplier

Another Representation of the Circuit

Building block: Full Adder + AND





4 x 4 array of building blocks

Other number representations

Other number representations are also commonly used :

- **Fixed-point**: allows for fractional representation
- **Floating-point**: allows for high precision, very large and/or very small numbers
- **Binary-coded decimal (BCD)**: another form for integer representation
- **American Standard Code for Information Interchange (ASCII)**: represents information in computers is used for both numbers and letters and some control codes

Fixed-point number

- A fixed-point number consists of integer and fraction parts
- In positional notation, it is written as

$$B=b_{n-1}b_{n-2}...b_1b_0.b_{-1}b_{-2}...b_{-k}$$

- The position of the radix point is assumed to be fixed
- For example,

$$B=(01001010.10101)_2$$

$$B=1x2^6+1x2^3+1x2^1+1x2^{-1}+1x2^{-3}+1x2^{-5}$$

$$B=64+8+2+.5+.125+.03125$$

$$B=(74.65625)_{10}$$

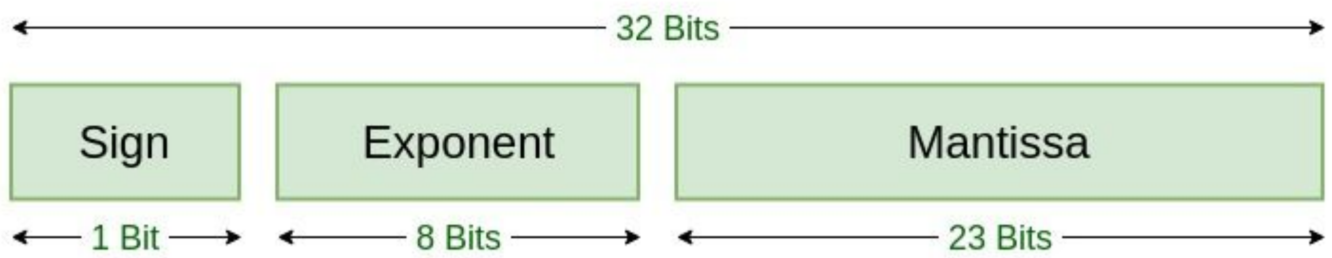
$$B=(8A.A8)_{16}$$

Floating Point numbers

- Fixed-point numbers have a range that is limited by the significant digits used to represent the number
- For some applications, it is often necessary to deal with numbers that are very large (or very small)
- For these cases, it is better to use a floating-point representation in which numbers are represented by a mantissa comprising the significant digits and an exponent of the radix R
- The format is Mantissa x R $^{\text{Exponent}}$
- The numbers are usually normalized such that the radix point is placed to the right of the first non-zero digit (for example, 5.234×10^{43} or 3.75×10^{-35})

Floating Point numbers

- The IEEE defines a 32-bit (single precision) format for floating point values
  - Sign bit (S): most significant bit
  - 8-bit exponent field (E): excess-127 exponent
    - True exponent = E-127
    - E=0 -> 32-bit value=0
    - E=255 -> 32-bit value=∞
  - 23-bit mantissa (M)



Floating Point numbers

- The IEEE 754 standard calls for a normalized mantissa, which means that the most significant bit is always set to 1.
- It is not necessary to include this bit explicitly in the mantissa field
  - If M is the value in the 23-bit mantissa field, the true (24-bit) mantissa is actually 1.M
  - The value of the floating point number is then

    Value = $(-1)^S.M \times 2^{E-127}$

Floating Point numbers

- For example,

  01000000011000000000000000000000

  $=+(1.11)_2 x2^{(128-127)}$

  $=+(1.11)_2 x2^{1}$

  $=+(11.1)_2$

  $=+(1x2^{1}+1x2^{0}+1x2^{-1})=(3.5)_{10}$

- What is the following?

  00111111011000000000000000000000

Binary-coded-decimal number

- It is possible to represent decimal numbers simply by encoding each decimal digit in binary form
  - Called binary-coded-decimal (BCD)
- Because there are 10 digits to represent, it is necessary to use four bits per digit
  - From 0=0000 to 9=1001
  - $(01111000)BCD=(78)_{10}$
- BCD representation was used in some early computers and many handheld calculators
  - Provides a format that is convenient when numerical

information is to be displayed on a simple digit-oriented

display