CSC 211: DIGITAL ELECTRONICS II

- **MEDIUM SCALE INTEGRATED (MSI) DEVICES**

Dr. Ronoh Saweh
Department of Computer Science
KIBABII UNIVERSITY

INTRODUCTION

- There are several specialized MSI components that have extensive use in digital systems. These are classified as standard components.
- *These include adders, subtractors, comparators, decoders, encoders and multiplexers.*

Binary Adder–Subtractor

- The most basic arithmetic operation is the addition of two binary digits. This simple addition consists of four possible elementary operations:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Binary Adder–Subtractor

- The first three operations produce a sum (S) of one digit, but when both augend and addend bits are equal to 1 a carry (C) is also generated (this propagates to the next most significant stage of the addition).

Half Adder

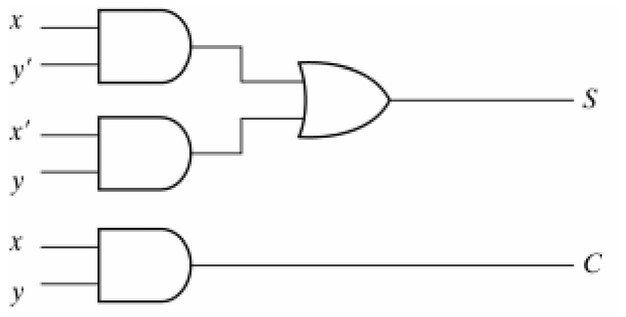
- Performs the addition of two bits.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = x'y + xy'$

$C = xy$

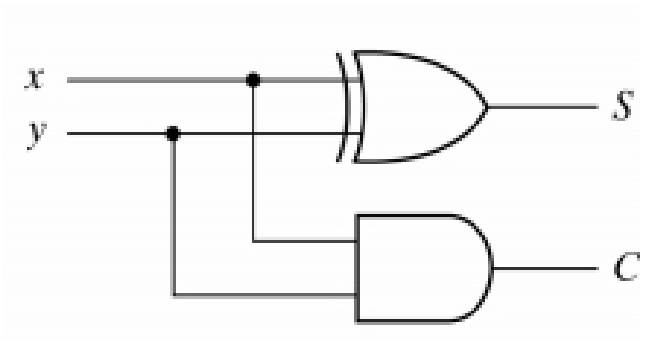
- **Implementation**



$S = x'y + xy'$

$C = xy$

Half Adder

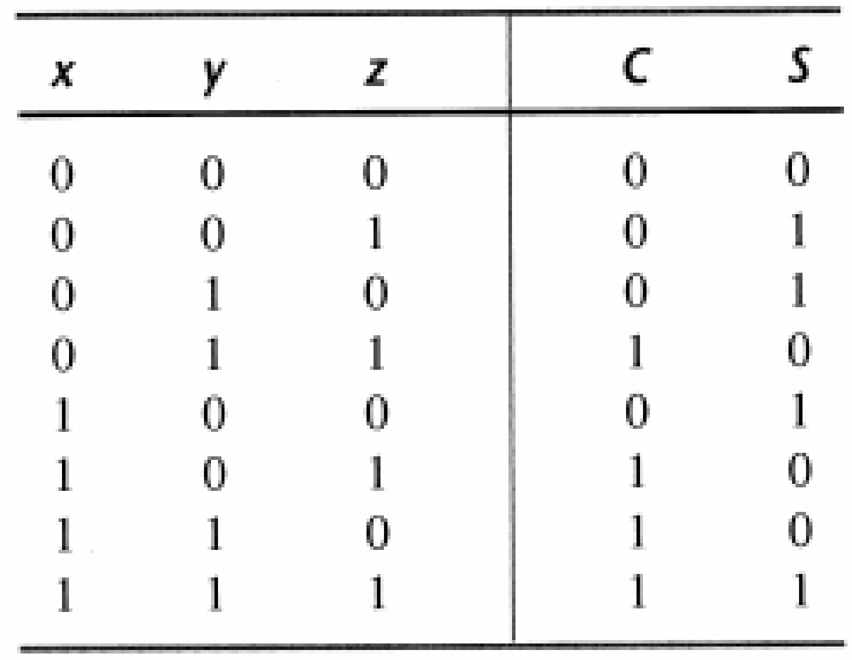


$S = x \oplus y$

$C = xy$

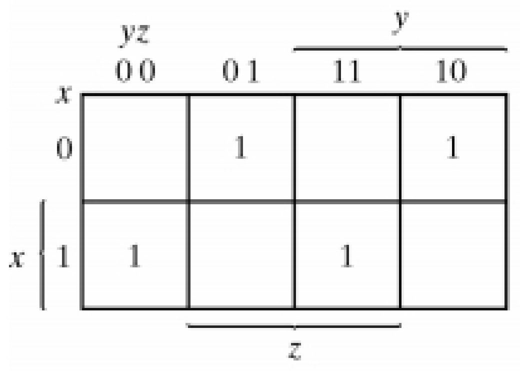
Full Adder

- Performs the arithmetic sum of three bits

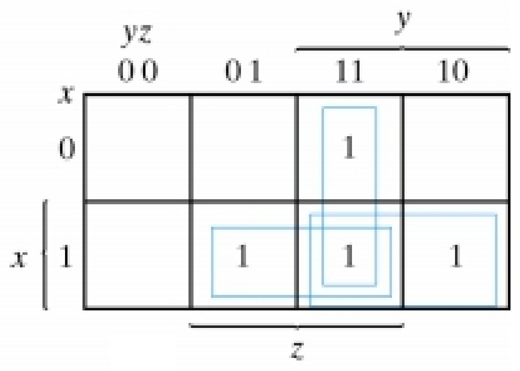


x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

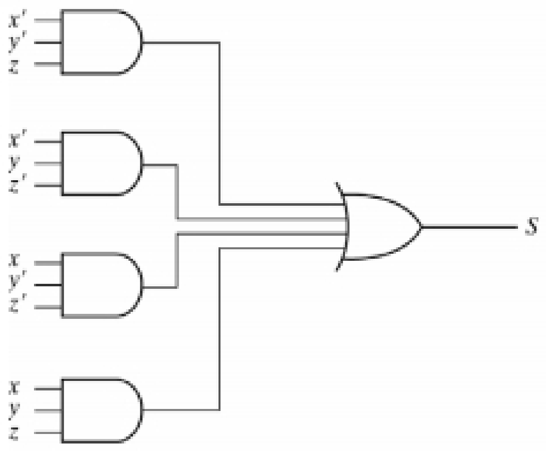
Full Adder

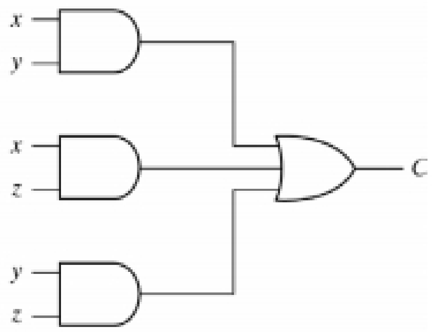


$S = x'y'z + x'yz' + xy'z' + xyz$



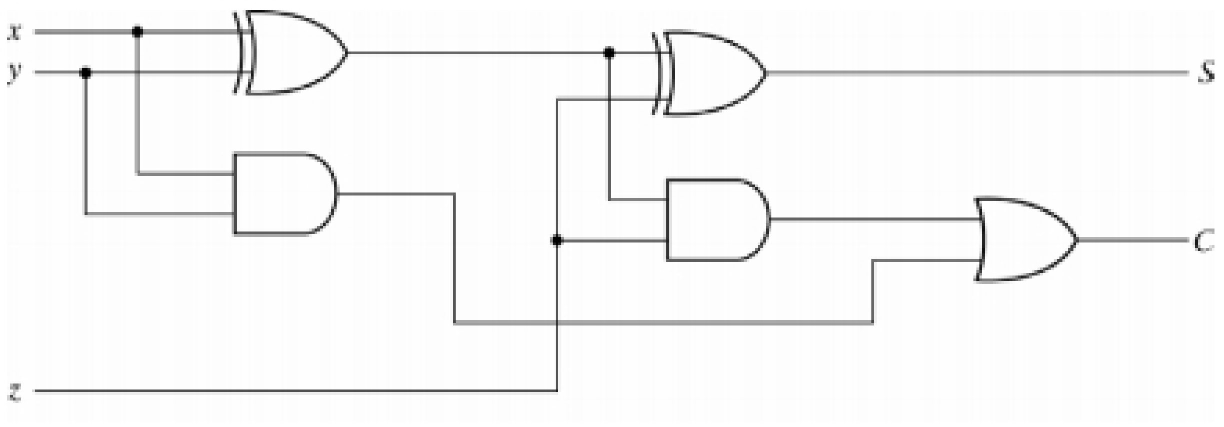
$C = xy + xz + yz$
$\quad = xy + xy'z + x'yz$





Alternative Implementation

- The full adder can also be realized with two half adders and one OR-gate:



- The output S from the second half adder is the X OR of z and the output of the first half adder, giving:

$$\begin{aligned} S &= z \oplus (x \oplus y) \\ &= z'(xy' + x'y) + z(xy' + x'y')' \\ &= z'(xy' + x'y) + z(xy + x'y') \\ &= xy'z' + x'yz' + xyz + x'y'z \end{aligned}$$

- The carry output is:

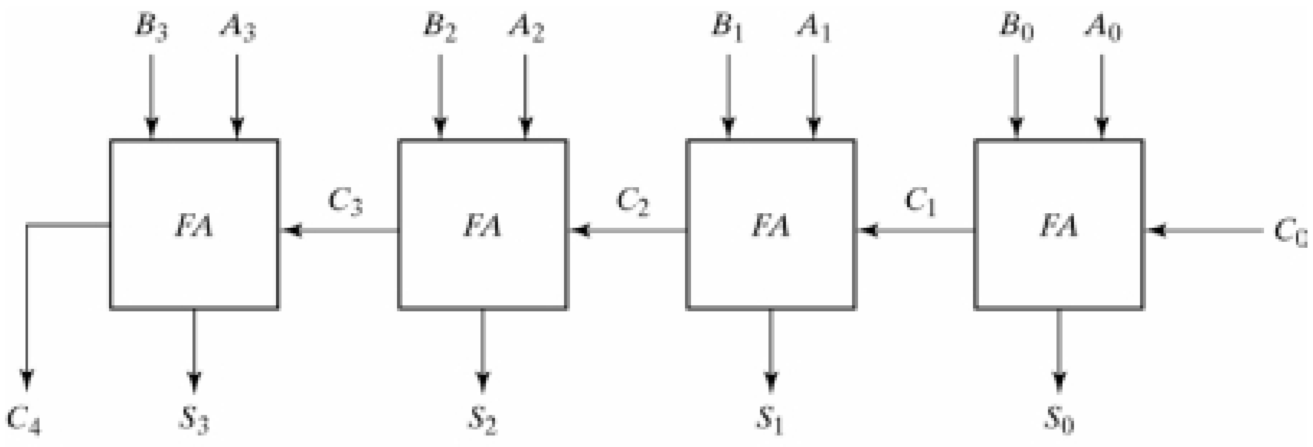
$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

Binary Adder

Binary Adder

- Produces the arithmetic sum of two binary numbers.
- It can be realized with full adders (FAs) connected in cascade.
- A 4-bit binary ripple adder is realized as shown below:

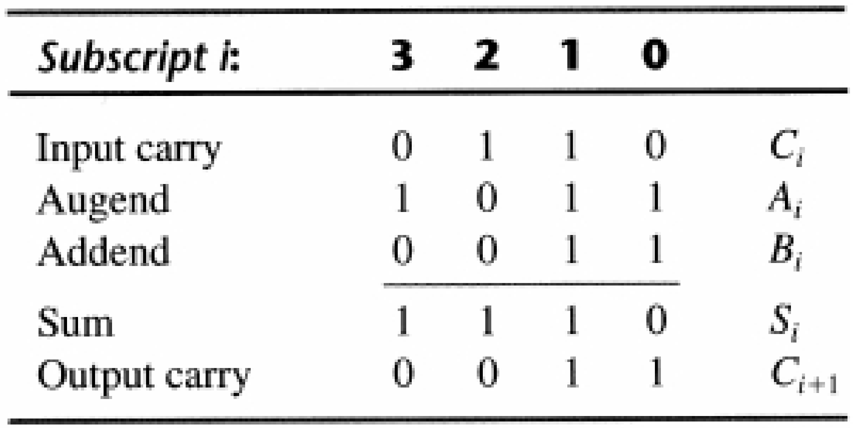
Binary Adder



- An n-bit adder requires n full adders with each output carry connected to the input carry of the next higher-order full adder.

Example

- Consider the two binary numbers, A = 1011 and B = 0011.
- Their sum S = 1110 is formed with the 4-bit adders as follows:

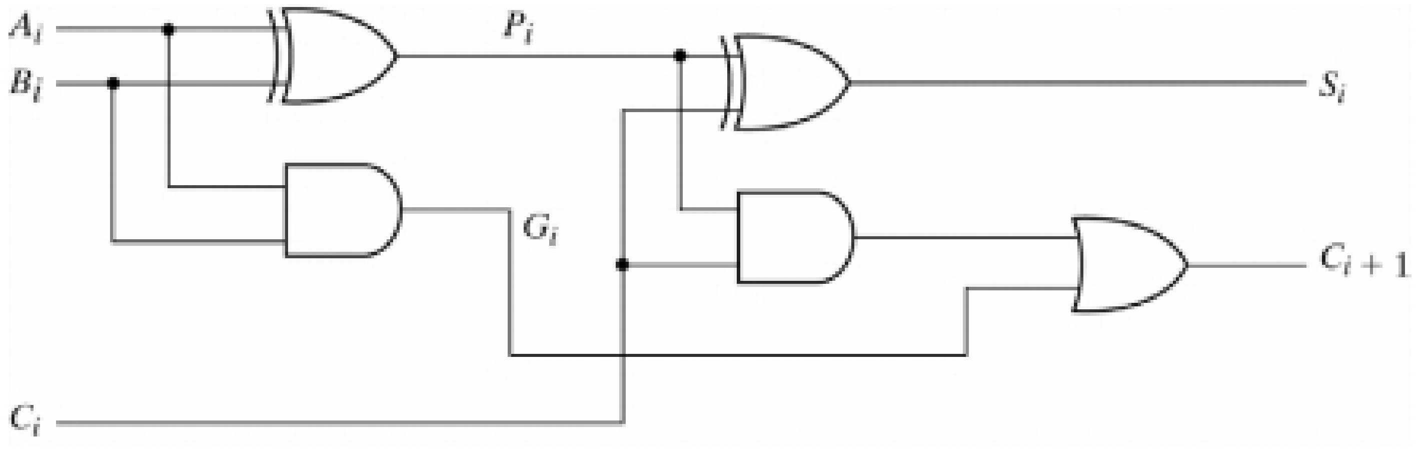


Subscript i:	**3**	**2**	**1**	**0**	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

Carry Propagation

- The longest propagation time in a binary ripple adder is the time it takes the carry to propagate through all full adders.
- The number of gate levels for the carry propagation can be found from the circuit of the full adder:

Carry Propagation



Carry Propagation

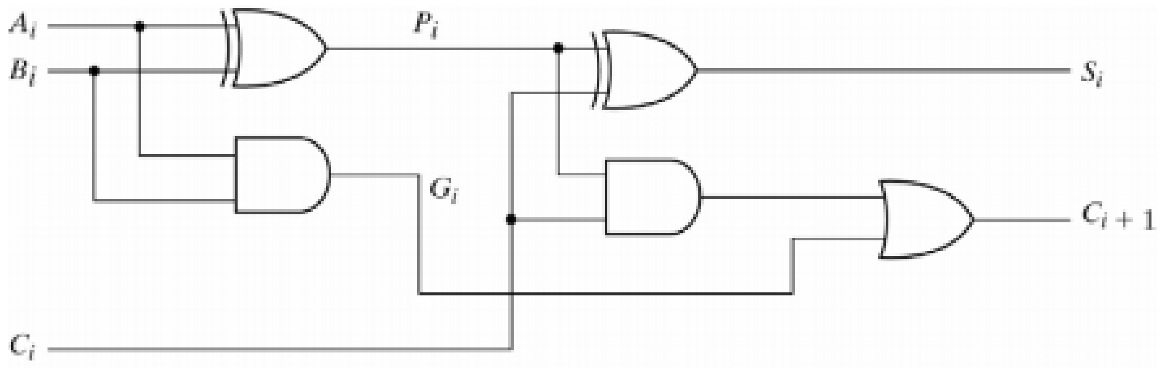
- The subscript i denotes a given stage in the adder.
- The signals P_i and G_i settle to their steady state values after they propagate through their gates.
- P_i and G_i are common to all full adders and depend only on the input augend and addend bits.

Carry Propagation

- The signal from the input carry C_i to the output carry C_i+1, propagates through two gates.
- If there are four full adders, the output carry C_4 would have 2 X 4 = 8 gate levels from C_0 to C_4.
- Clearly, carry propagation time the limiting factor on the speed with which two numbers are added.

Carry Propagation

- The most widely used technique for reducing the carry propagation time in a parallel adder employs the principle of carry lookahead. Consider again the circuit of the full adder:



Carry Propagation

- If two new binary variables are defined:

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

- the output sum and carry can be expressed as:

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

Carry Propagation

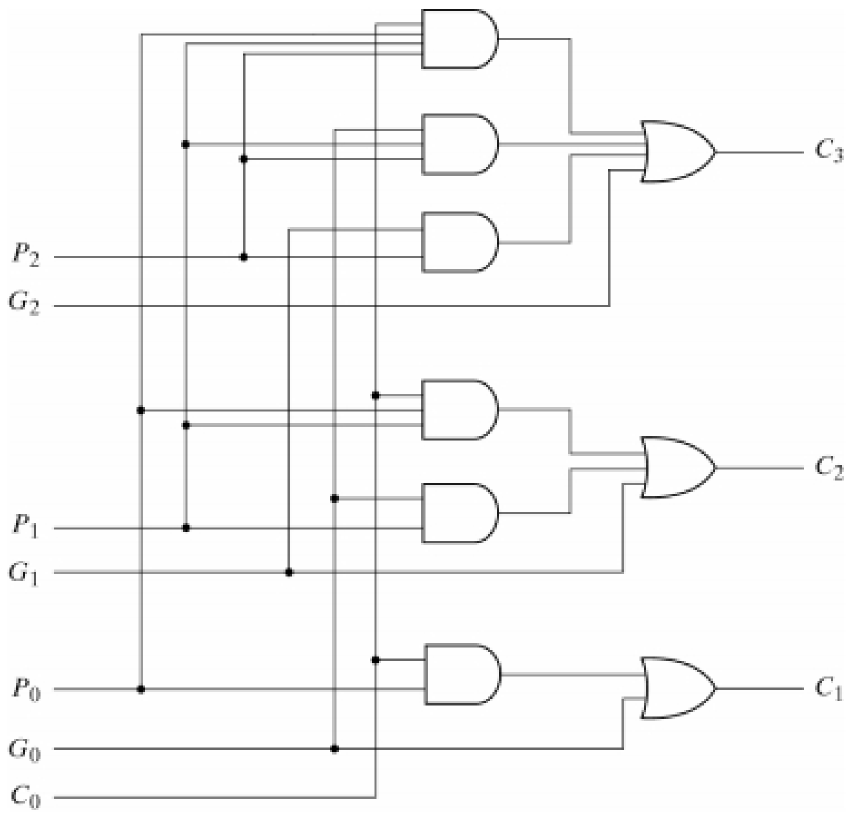
- G_i is called a carry generate and it produces a carry of 1, regardless of the input carry C_i.
- P_i is called a carry propagate because it is the term associated with the propagation of the carry from C_i to C_{i+1}.
- We can now write the Boolean functions for the carry outputs of each stage and substitute for each C_i its value from the previous equations

Carry Propagation

$$C_0 = \text{input carry}$$
$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

- These expressions are implemented in the following carry lookahead generator:

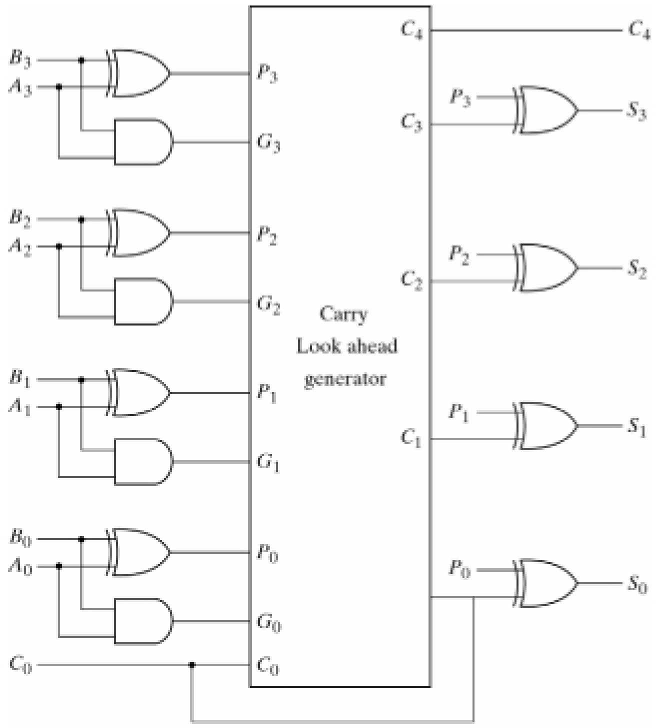
Carry Propagation



Carry Propagation

- The construction of a 4-bit adder with a carry lookahead scheme is shown below:
- All output carries are generated after a delay through two levels of gates. Thus outputs S_1 through S_3 have equal propagation times. The two level circuit for the output carry C_4 is not shown. This can be derived by the equation-substitution method.

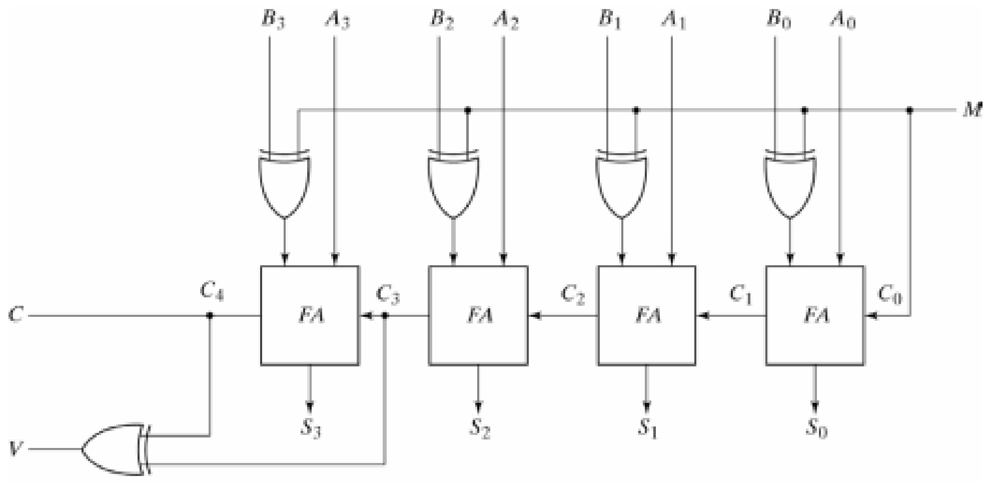
Carry Propagation



Binary Subtractor

- The subtraction of unsigned binary numbers can be simplified by means of complements.
- For example, A – B can be done by taking the 2’s complement of B and adding it to A.
- The 2’s complement can be obtained by taking the 1’s complement and adding 1 to the least significant pair of bits.
- The 1’s complement can be realized with inverters and a 1 can be added to the sum through the input carry.
- A 4-bit adder-subtractor circuit is shown below:

Binary Subtractor

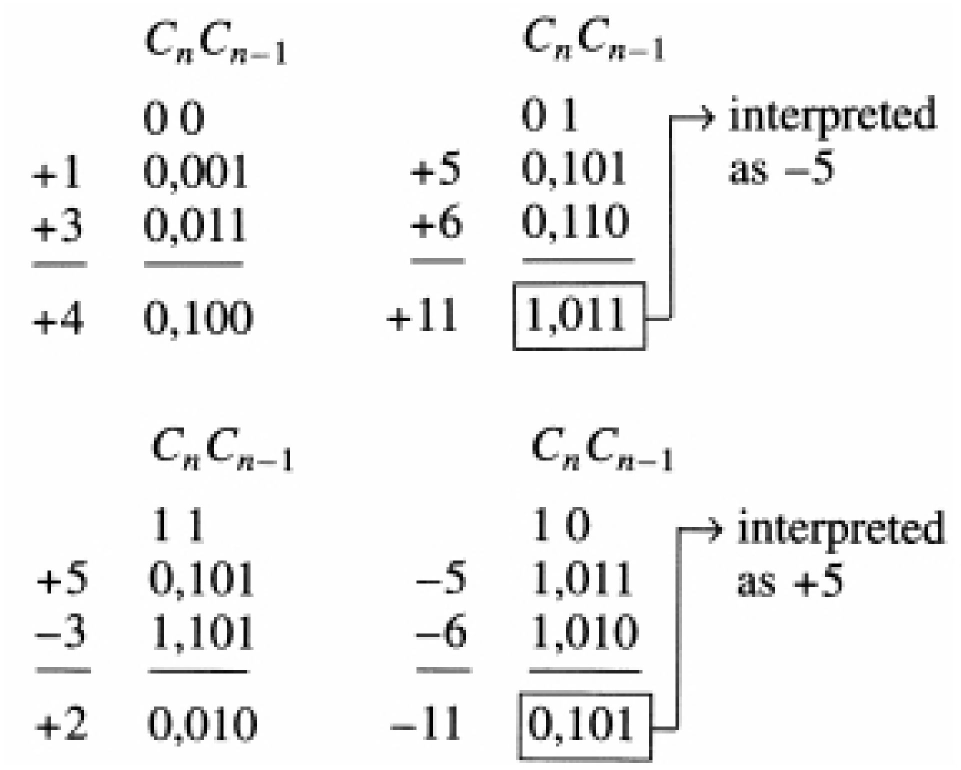


- M = 0; addition, M = 1; subtraction
- The V bit detects an overflow when the two binary numbers to be added are signed.

Overflow

- Sometimes, when an adder/subtractor is using signed arithmetic, there is arithmetic overflow from the most significant magnitude bit into the sign bit.
- An overflow may occur if the two numbers added are both positive or negative.
- An example of 4 possible situations that may arise is given below for a 4-bit (n = 4) word.
- For each case the carries C_{n-1} and C_n are recorded:

Overflow



$$\begin{array}{rl} & C_nC_{n-1} \\ & 0\ 0 \\ +1 & 0{,}001 \\ +3 & 0{,}011 \\ \hline +4 & 0{,}100 \end{array} \qquad \begin{array}{rl} & C_nC_{n-1} \\ & 0\ 1 \\ +5 & 0{,}101 \\ +6 & 0{,}110 \\ \hline +11 & \boxed{1{,}011} \end{array} \rightarrow \text{interpreted as } -5$$

$$\begin{array}{rl} & C_nC_{n-1} \\ & 1\ 1 \\ +5 & 0{,}101 \\ -3 & 1{,}101 \\ \hline +2 & 0{,}010 \end{array} \qquad \begin{array}{rl} & C_nC_{n-1} \\ & 1\ 0 \\ -5 & 1{,}011 \\ -6 & 1{,}010 \\ \hline -11 & \boxed{0{,}101} \end{array} \rightarrow \text{interpreted as } +5$$

Overflow

- In the case where the sum should +11 or -11, the corresponding binary sum is wrong.
- It is obvious that an overflow flag should be raised when C_{n-1} = 1 and C_n = 0, or, when C_n-1 = 0 and C_n = 1.
- Hence, the equation for overflow is:

$$\boxed{V = C_{n-1} \oplus C_n}$$

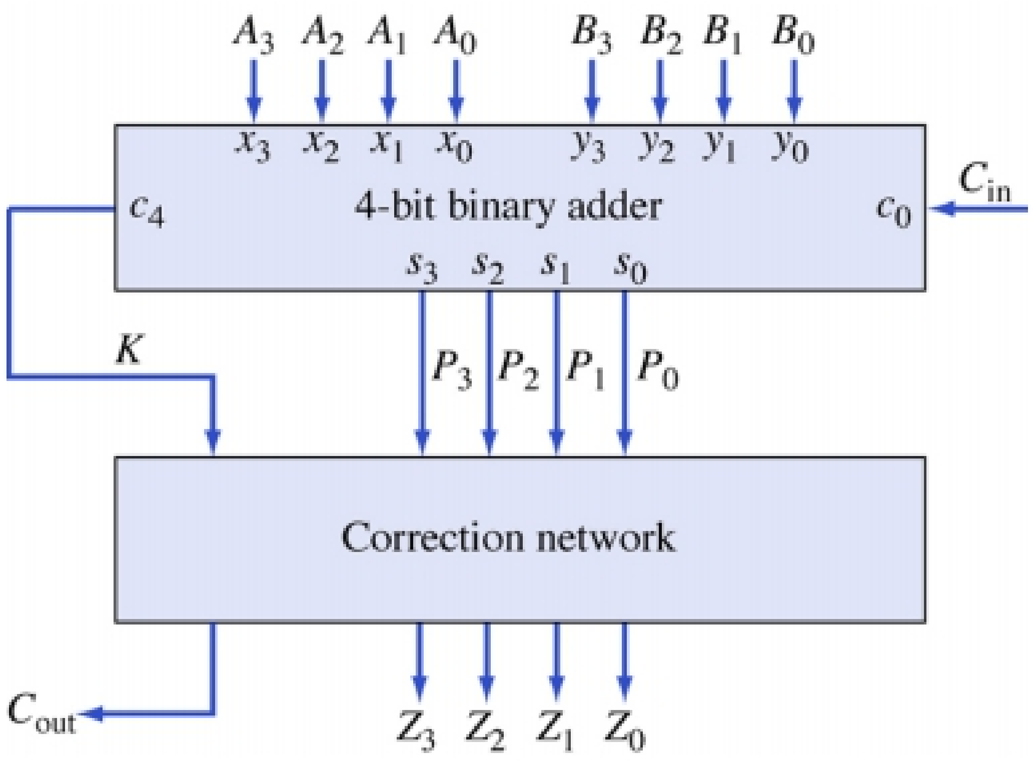
Decimal Adder

- Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary-coded form.
- The 8421 weighted coding scheme is the most commonly occurring in digital systems and is often referred to as simply BCD (binary-coded decimal).

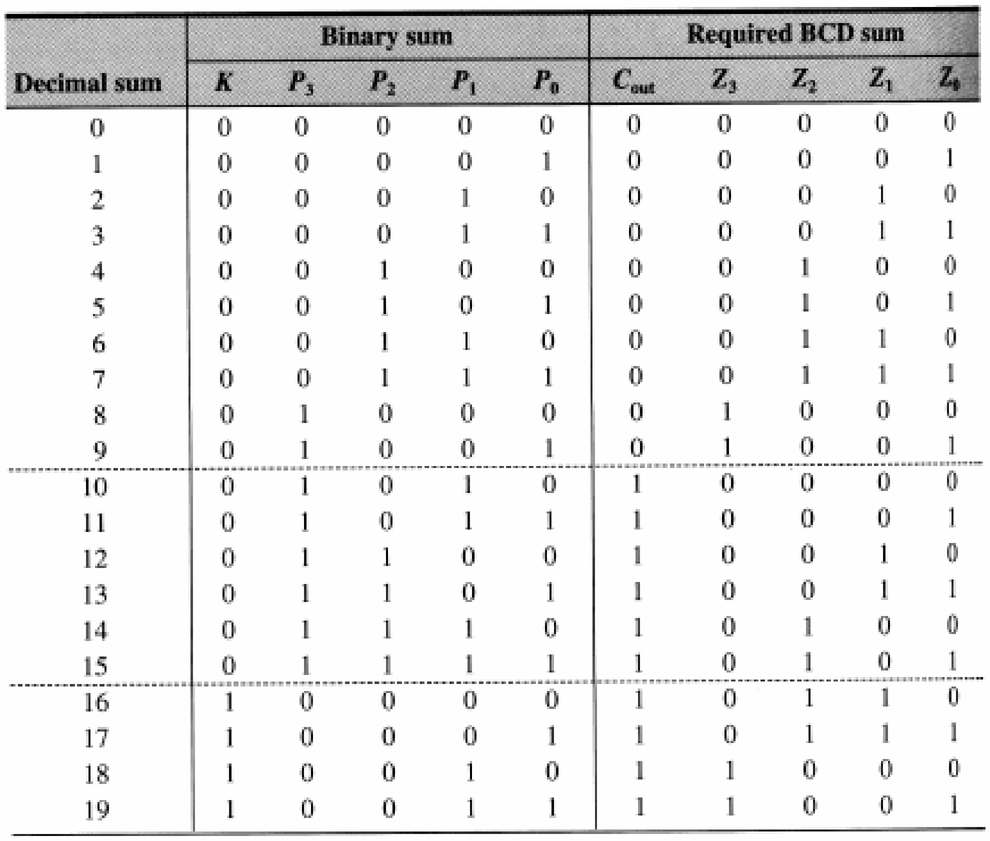
Decimal Adder

- When using BCD, a single-decade decimal adder can be realized by first performing conventional binary addition on two binary-coded operands and then applying a corrective procedure.
- This is illustrated below:

Decimal Adder



Decimal Adder

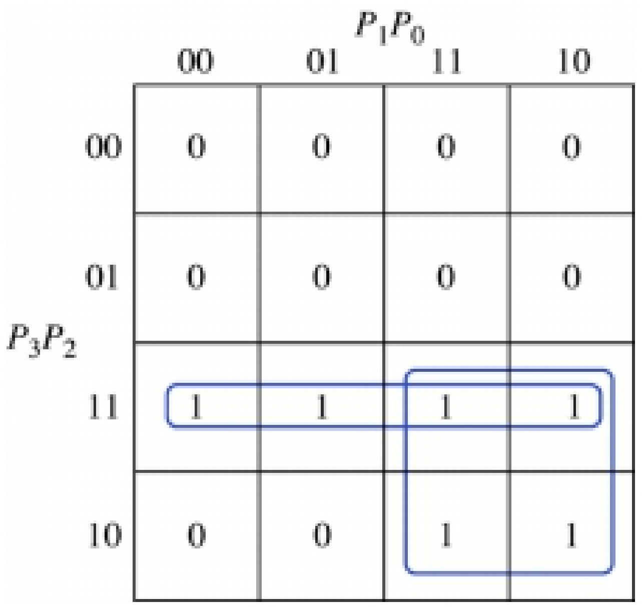


	Binary sum					Required BCD sum				
Decimal sum	K	P_3	P_2	P_1	P_0	C_{out}	Z_3	Z_2	Z_1	Z_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

Decimal Adder

- No correction to the binary sum is needed when
- $KP_3P_2P_1P_0 \leq 01001$.
- However, 0110 (decimal 6) must be added to
- $P_3P_2P_1P_0$ when $K\ P_3P_2P_1P_0 > 01001$.
- The logic circuit that detects the necessary correction can be derived from the table entries.
- Clearly, a correction is needed when the binary sum has an output carry $K = 1$.
- For the other six combinations from 1010 through 1111 (that also need a correction), a Boolean expression is required to detect them:

Decimal Adder



P_3P_2 \ P_1P_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$$f = P_3P_2 + P_3P_1$$

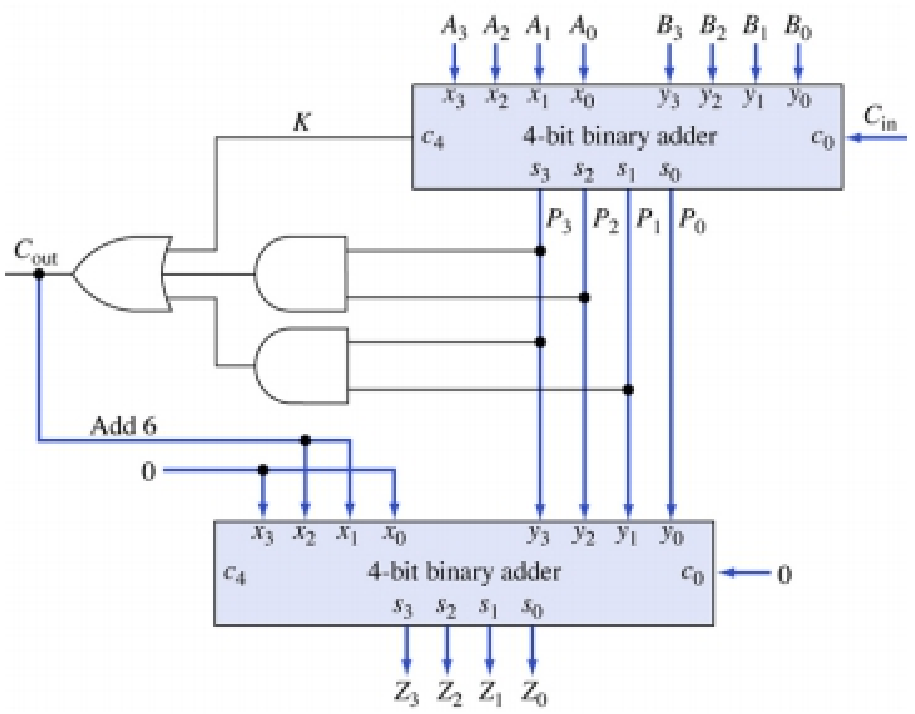
Decimal Adder

- The condition for a correction and an output carry can be expressed by the Boolean function:

$$\boxed{\text{Add}\,6 = K + P_3P_2 + P_3P_1}$$

- The logic diagram of a single-decade BCD adder is shown below:

Decimal Adder



Decimal Adder

- Whenever C_{out} = 0, the outputs from the upper 4-bit binary adder are sent to the lower 4-bit adder and decimal 0 is added.
- However, whenever C_{out} = 1, decimal 6 is added to the outputs of the upper 4-bit binary adder so that the correct sum digit is obtained.
- A decimal adder for two n-digit BCD numbers can be constructed by cascading the above system in much the same way as was done for the ripple binary adder.