Combinational Circuits

- Logic circuits for digital systems may be combinational or sequential.
- A combinational circuit - logic gates whose outputs at any time are determined from only the present combination of inputs.
- A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.
- In contrast, sequential circuits has storage elements in addition to logic gates.
- Their outputs are a function of the inputs and the state of the storage elements.
- Sequential circuits are the building blocks of digital systems.

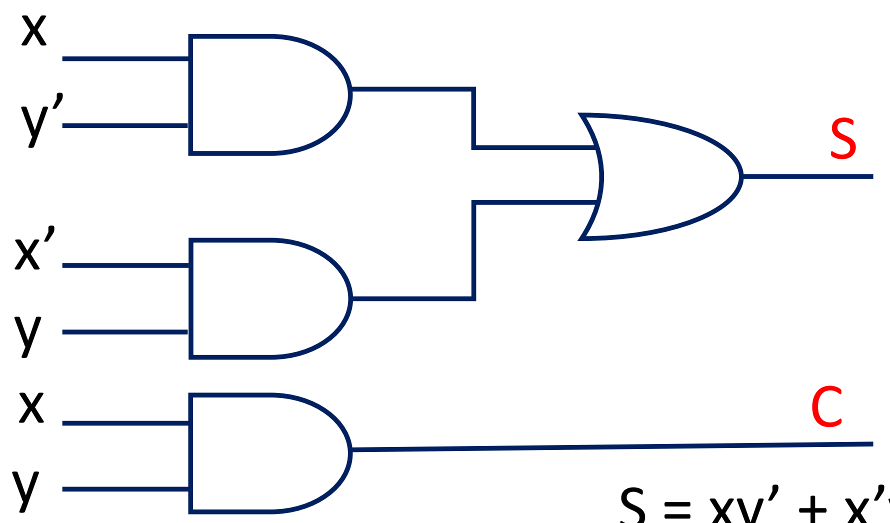
BINARY ADDER–SUBTRACTOR

- Addition of two binary digits: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 10.
- First three operations produce a sum of one digit.
- When both augend and addend bits are equal to 1, the binary sum consists of two digits.
- The higher significant bit of this result is called a *carry* .
- When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits.
- A combinational circuit that performs addition of two bits is called a *half adder* .
- One that performs the addition of three bits (two significant bits and a previous carry) is a *full adder*.
- A binary Adder–Subtractor - a combinational circuit that performs arithmetic operations of addition and subtraction with binary numbers.

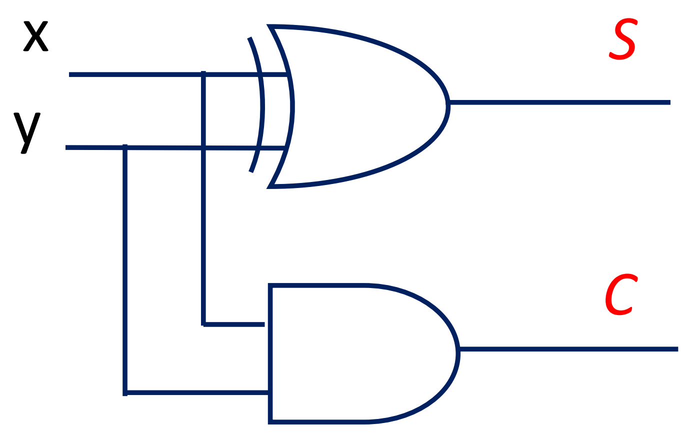
Half ADDER

- *x* and *y* are two inputs and *S* (for sum) and *C* (for carry) to the outputs.
- *C* output is 1 only when both inputs are 1.
- *S* output represents least significant bit of the sum.
- Simplified sum-of-products expressions are
  - S = xy’ + x’y
  - C = xy

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



S = xy’ + x’y
C = xy



S = x ⊕ y
C = xy

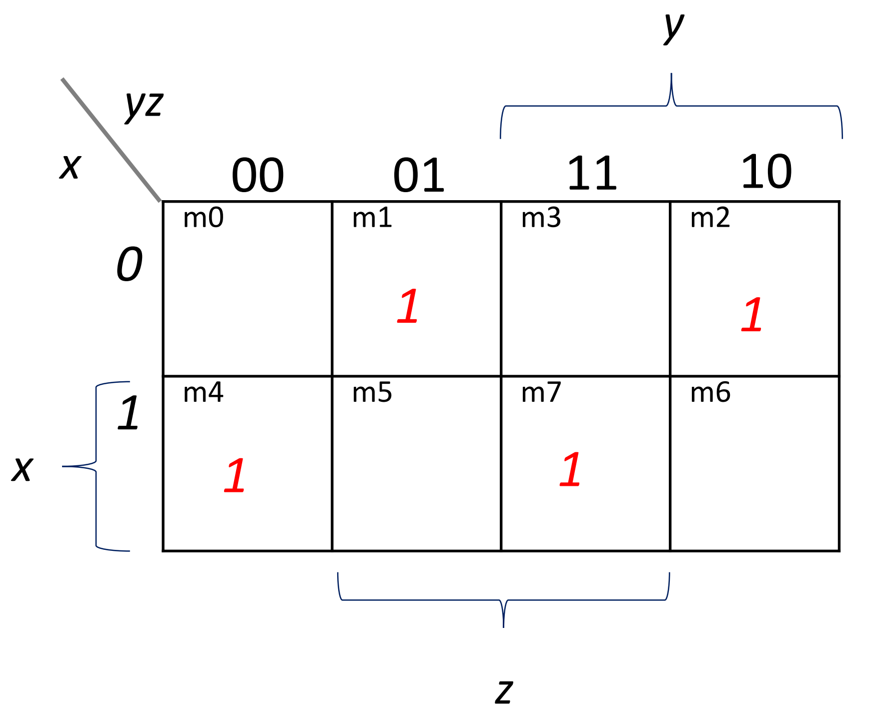
Full ADDER

- A full adder is a combinational circuit that forms the arithmetic sum of three bits.
- It consists of three inputs and two outputs.
- Two input variables represent the two significant bits to be added.
- Third input represents carry from the previous lower significant position.
- Addition of *n*-bit binary numbers requires use of a full adder,
- Addition proceeds on a bit-by-bit basis, right to left, beginning with the least significant bit.
- After the least significant bit, addition at each position adds not only the respective bits of the words, but must also consider a possible carry bit from addition at the previous position.
- Two outputs are necessary because arithmetic sum of three binary digits ranges in value from 0 to 3, and binary representation of 2 or 3 needs two bits.

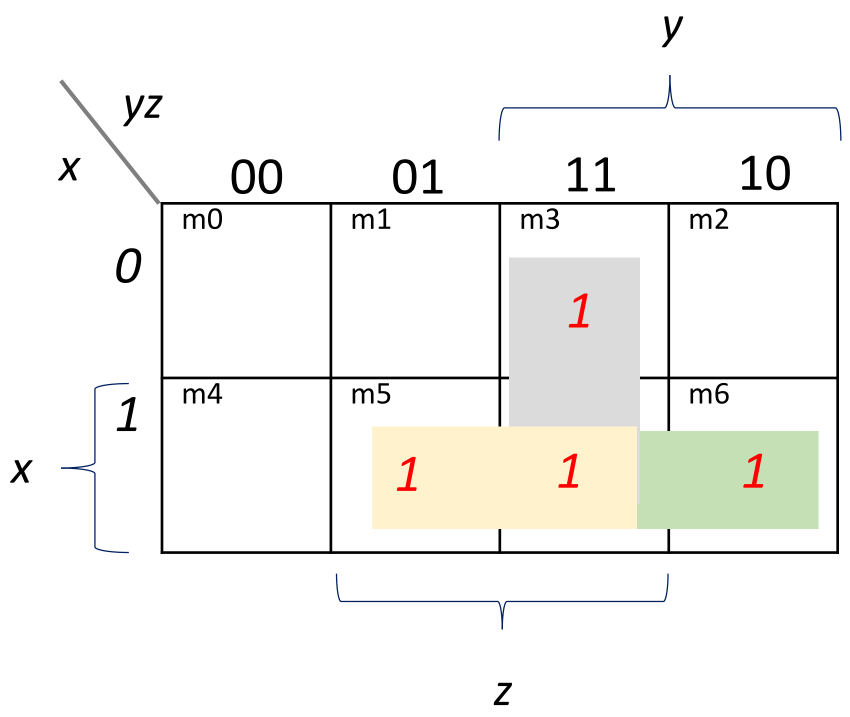
Full ADDER

- *S* output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to 1.
- *C* output has a carry of 1 if two or three inputs are equal to 1.

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



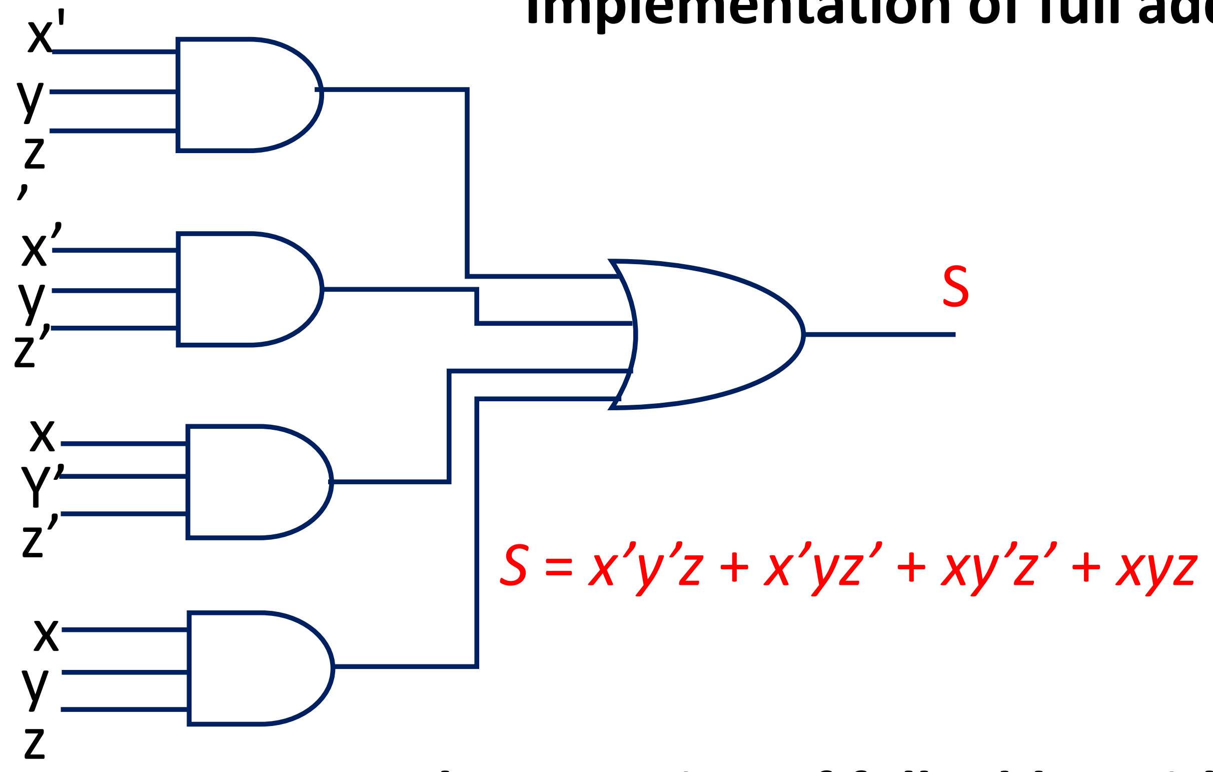
$S = x'y'z + x'yz' + xy'z' + xyz$



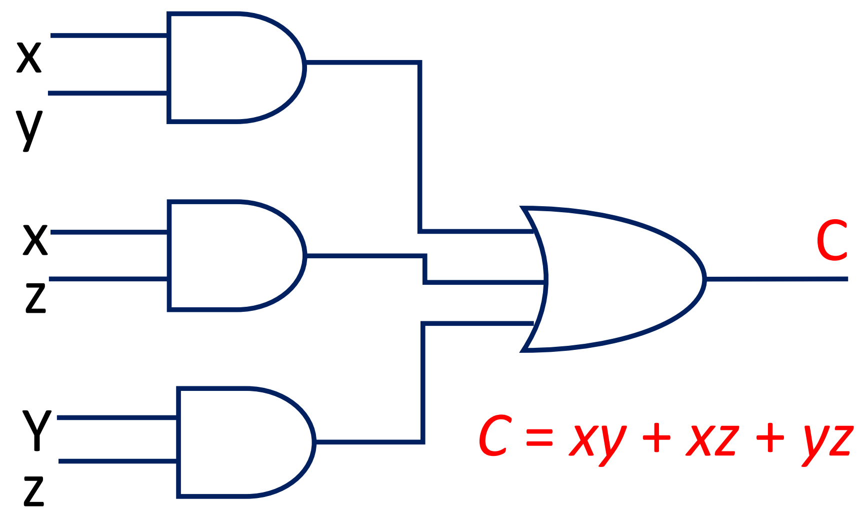
$C = xy + xz + yz$

x	y	Z	S	C
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**

Implementation of full adder in sum-of-products form

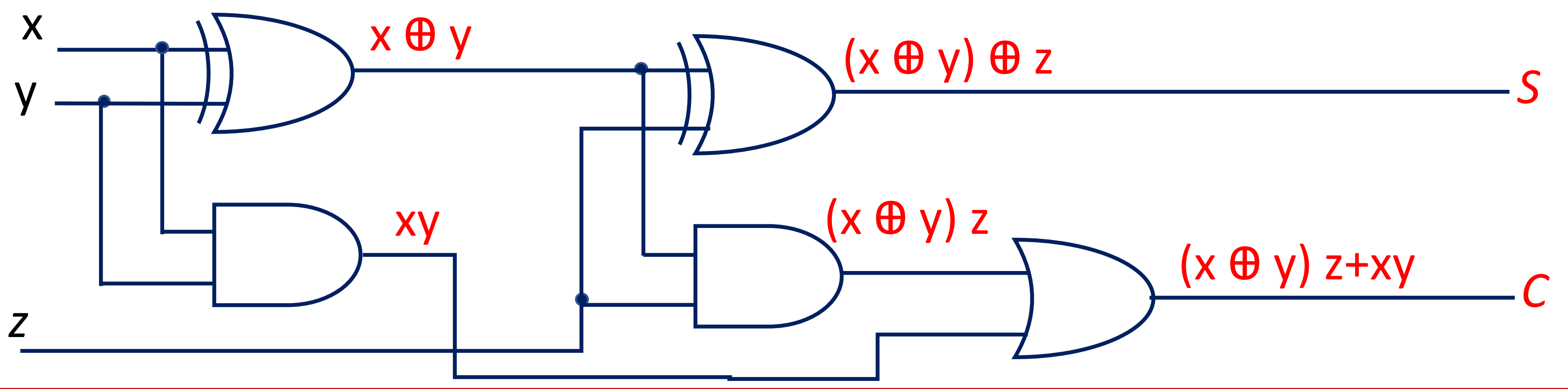


$S = x'y'z + x'yz' + xy'z' + xyz$



$C = xy + xz + yz$

Implementation of full adder with two half adders and an OR gate



Full ADDER

- Output from the second half adder is the exclusive-OR of *z* and the output of the first half adder

$S = (x \oplus y) \oplus z$

$= (xy' + x'y) \oplus z$

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

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

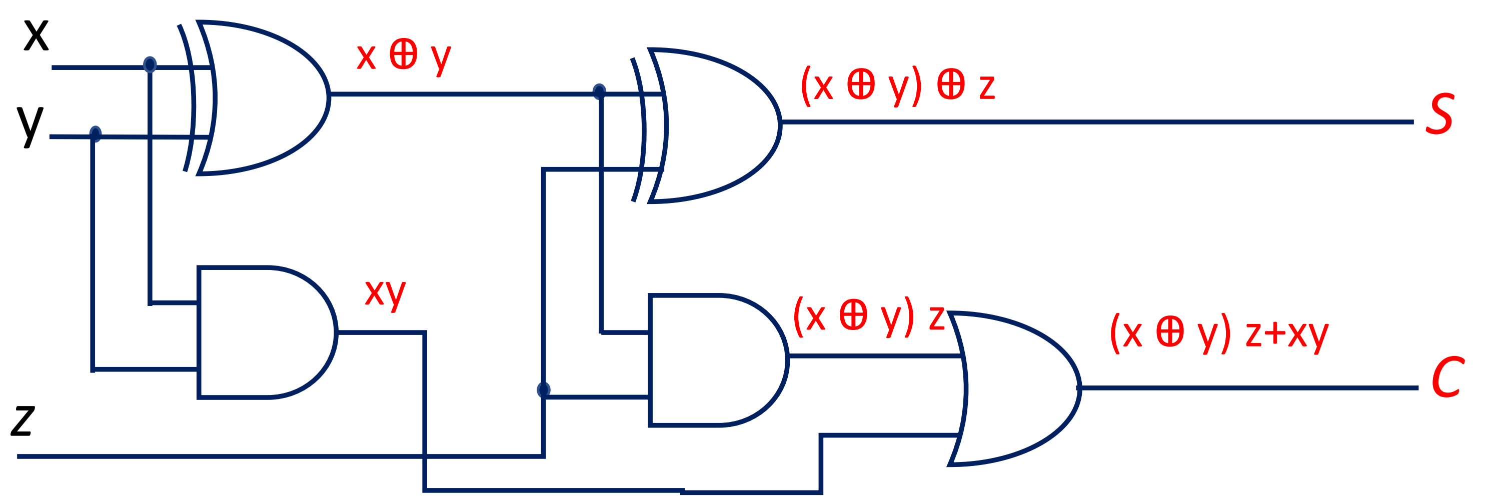
$= xy'z' + x'yz' + xyz + x'y'z$

- The carry output is

$C = (x \oplus y) z + xy$

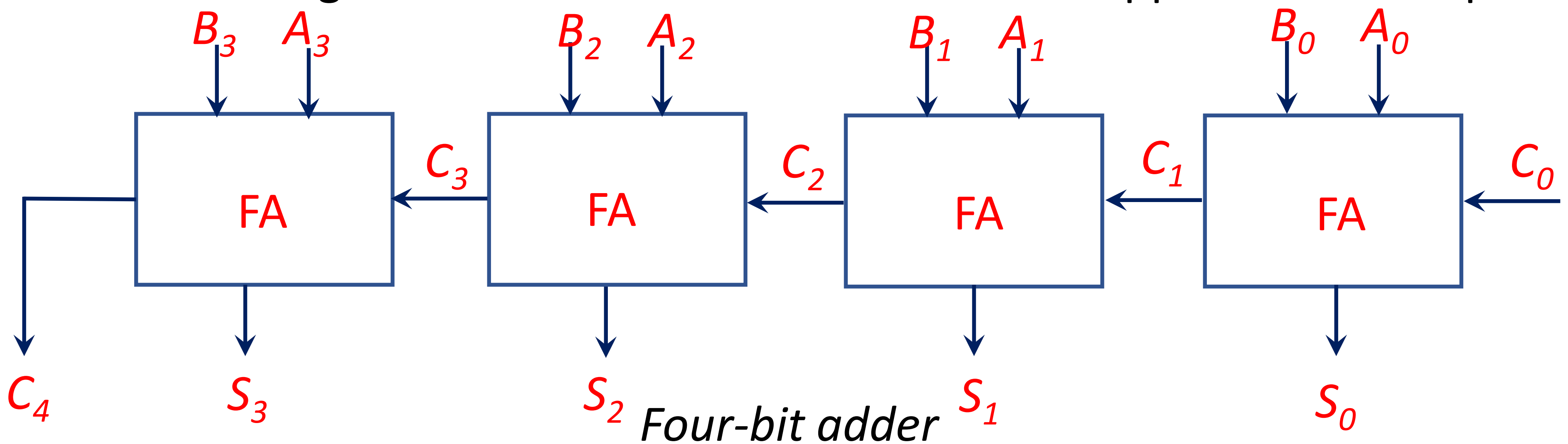
$= (xy' + x'y) z + xy$

$= xy'z + x'yz + xy$



Binary Adder

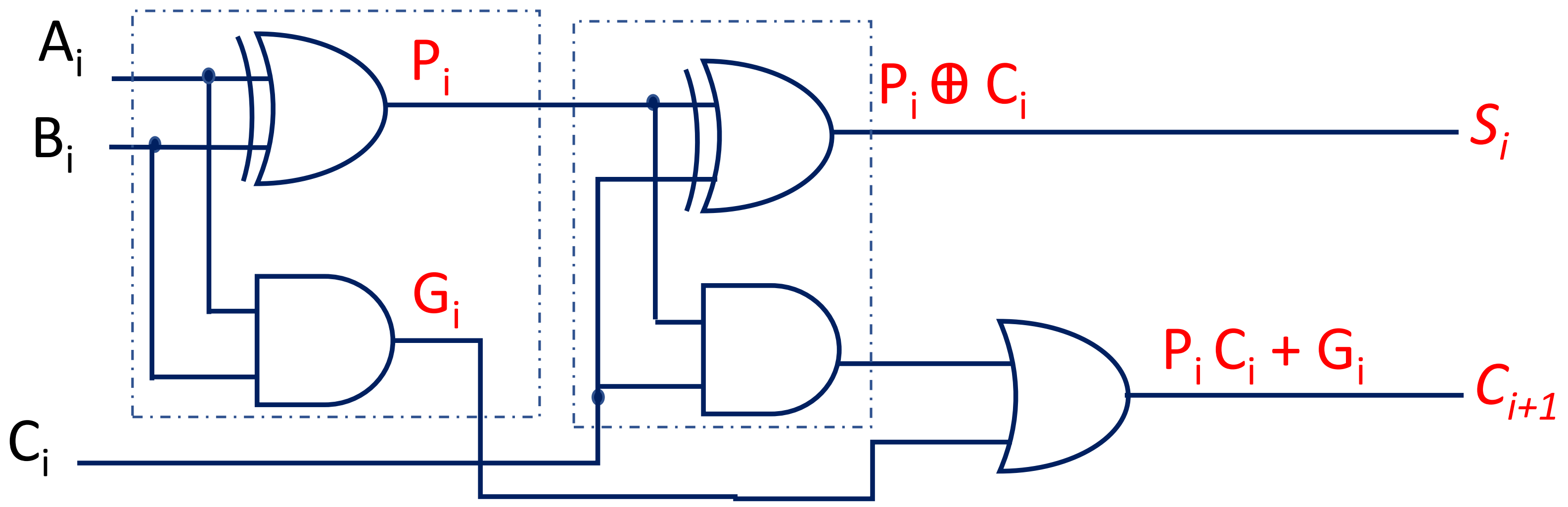
- A binary adder is a digital circuit that produces arithmetic sum of two binary numbers.
- It can be constructed with full adders connected in cascade, with output carry from each full adder connected to input carry of next full adder in the chain.
- Addition of *n*-bit numbers requires a chain of *n* full adders or a chain of one-half adder and (n -1) full adders.
- Input carry to the least significant position is fixed at 0.
- The sum bits are generated starting from the rightmost position and are available as soon as the corresponding previous carry bit is generated.
- All the carries must be generated for the correct sum bits to appear at the outputs.



Four-bit adder

Carry Propagation

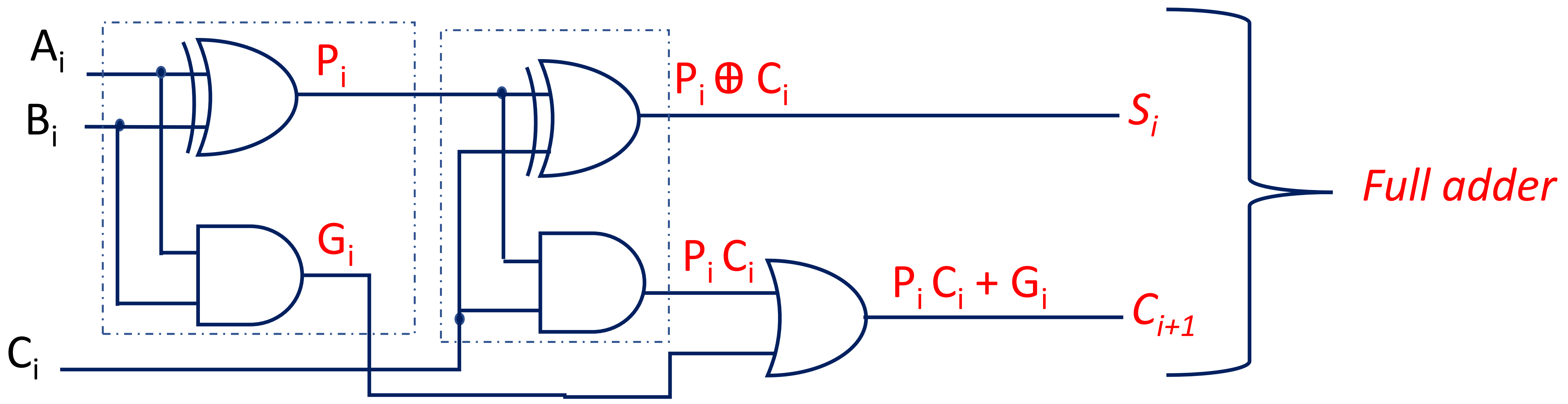
- The total propagation time is equal to the propagation delay of a typical gate, times the number of gate levels in the circuit.
- The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adders.
- Since each bit of the sum output depends on the value of the input carry, the value of S_i at any given stage in the adder will be in its steady-state final value only after the input carry to that stage has been propagated.



Carry lookahead logic

- Carry propagation time limits the speed with which two numbers are added.
- Output of any combinational circuit will not be correct unless the signals are given enough time to propagate through the gates connected from the inputs to the outputs.
- Since all other arithmetic operations are implemented by successive additions, time consumed during the addition process is critical.
- An obvious solution for reducing the carry propagation delay time is to employ faster gates with reduced delays. However, physical circuits have a limit to their capability.
- Another solution is to increase the complexity of the equipment in such a way that carry delay time is reduced.
- There are several techniques for reducing carry propagation time in a parallel adder.
- The most widely used technique employs the principle of *carry lookahead logic*.

Carry lookahead logic



$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

- Output sum and carry can respectively be expressed as

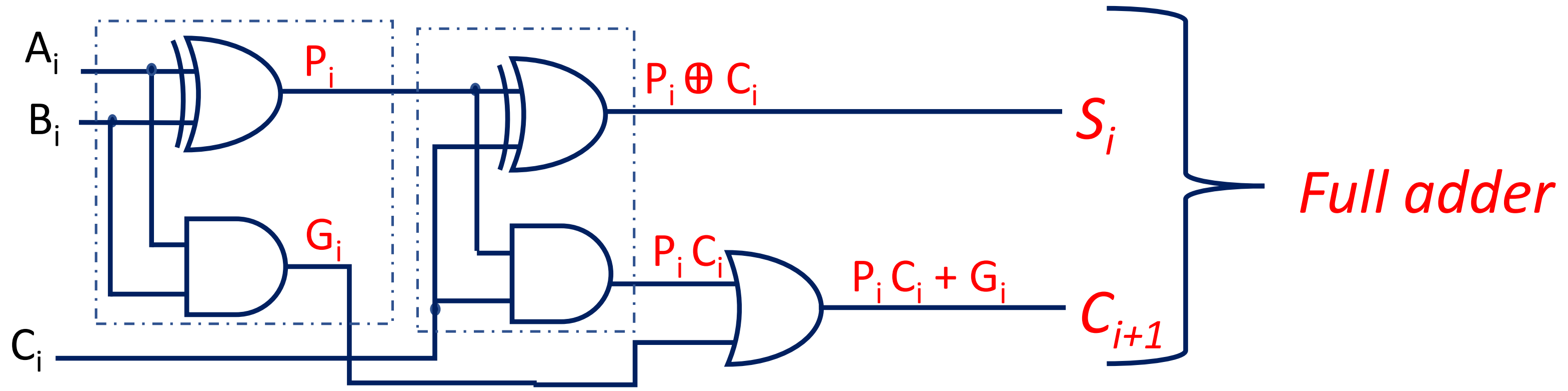
  $S_i = P_i \oplus C_i$

  $C_{i+1} = G_i + P_i C_i$

- G_i is called a *carry generate*, and it produces a carry of 1 when both A_i and B_i are 1, regardless of input carry C_i.
- P_i is called a *carry propagate*, because it determines whether a carry into stage i will propagate into stage (i + 1).

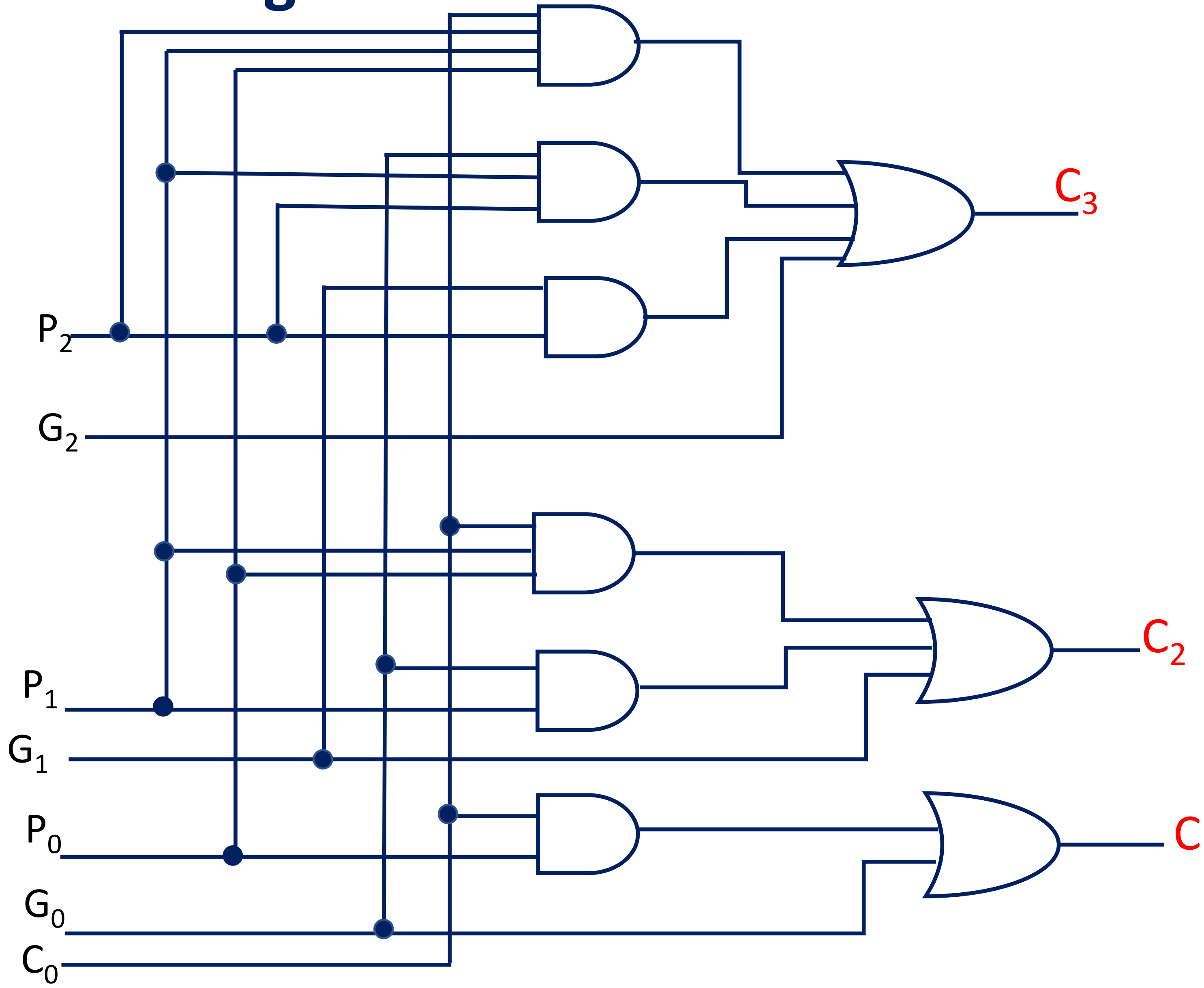
Carry lookahead logic

- Boolean functions for the carry outputs of each stage can be obtained by substituting value of each C_i from previous equations:
  - C_0 = input carry
  - $C_1 = G_0 + P_0C_0$
  - $C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$
  - $C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
- Boolean function for each output carry is expressed in sum-of-products form.
- Each function can be implemented with one level of AND gates followed by an OR gate (or by a two-level NAND).



Carry lookahead logic

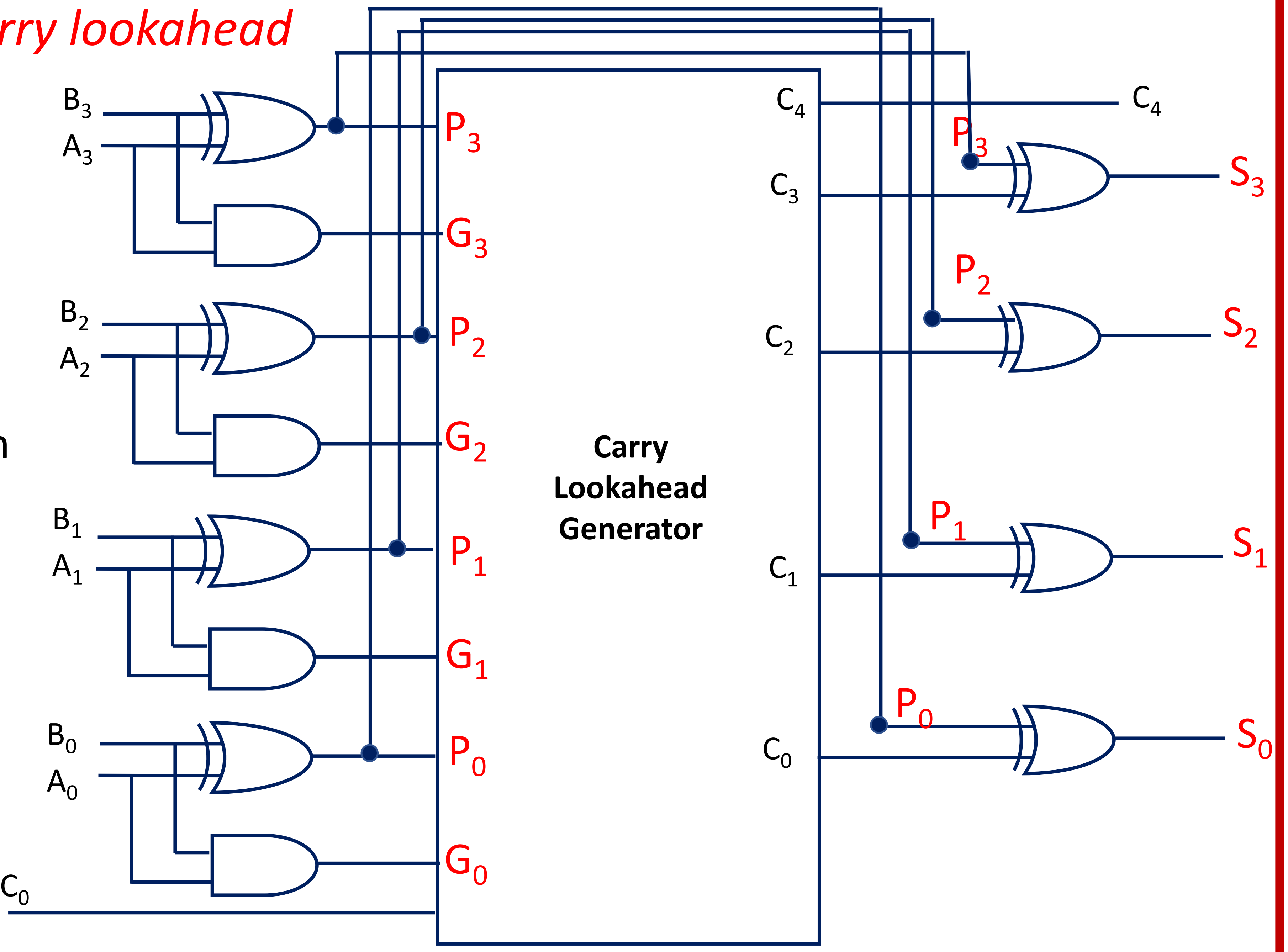
- This circuit can add in less time because C_3 does not have to wait for C_2 and C_1 to propagate.
- C_3 is propagated at the same time as C_1 and C_2.
- This gain in speed of operation is achieved at the expense of additional complexity (hardware).



Logic diagram of carry lookahead generator

Four-bit adder with carry lookahead

- Outputs S_1 through S_3 have equal propagation delay times.



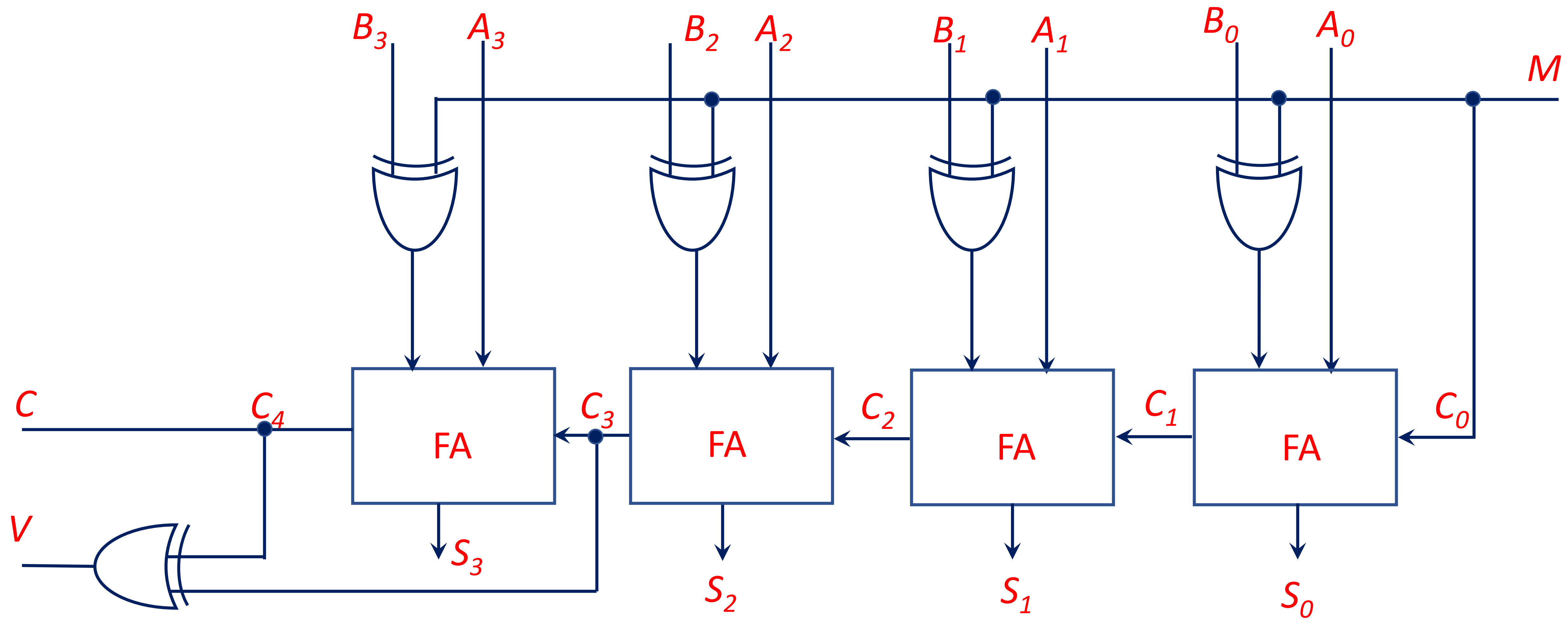
Binary Subtractor

- Subtraction of unsigned binary numbers can be done most conveniently by means of complements.
- Circuit for subtracting A - B consists of an adder with inverters placed between each data input B and the corresponding input of the full adder.
- The input carry C_0 must be equal to 1 when subtraction is performed.
- The operation thus performed becomes A , plus the 1’s complement of B , plus 1 equal to A plus the 2’s complement of B .
- For unsigned numbers, that gives A - B if $A \geq B$ or the 2’s complement of (B – A) if $A < B$.
- For signed numbers, the result is A - B, provided that there is no overflow.

Binary Subtractor

- Addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder.
- The mode input *M* controls the operation.
- When $M = 0$, circuit is an adder, and when $M = 1$, circuit becomes a subtractor.
- Each exclusive-OR gate receives input *M* and one of the inputs of *B* .
- When $M = 0$, we have $B \oplus 0 = B$. The full adders receive the value of *B* , input carry is 0, and the circuit performs *A* plus *B* .
- When $M = 1$, we have $B \oplus 1 = B'$ and $C_0 = 1$. The *B* inputs are all complemented and a 1 is added through the input carry.
- The circuit performs the operation *A* plus 2's complement of *B* .
- The exclusive-OR with output *V* is for detecting an overflow.
- Binary numbers in the signed-complement system are added and subtracted by the same basic addition and subtraction rules as are unsigned numbers.

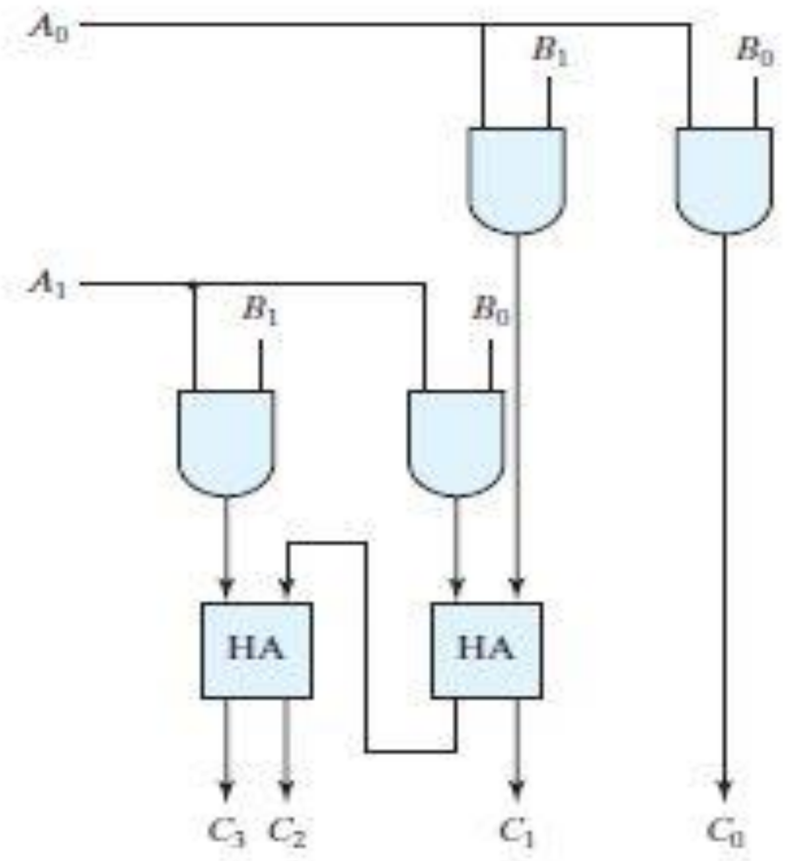
Binary Subtractor



BINARY MULTIPLIER

- The multiplicand is multiplied by each bit of the multiplier, starting from the least significant bit.
- Each such multiplication forms a partial product.
- Successive partial products are shifted one position to the left.
- The final product is obtained from the sum of the partial products.

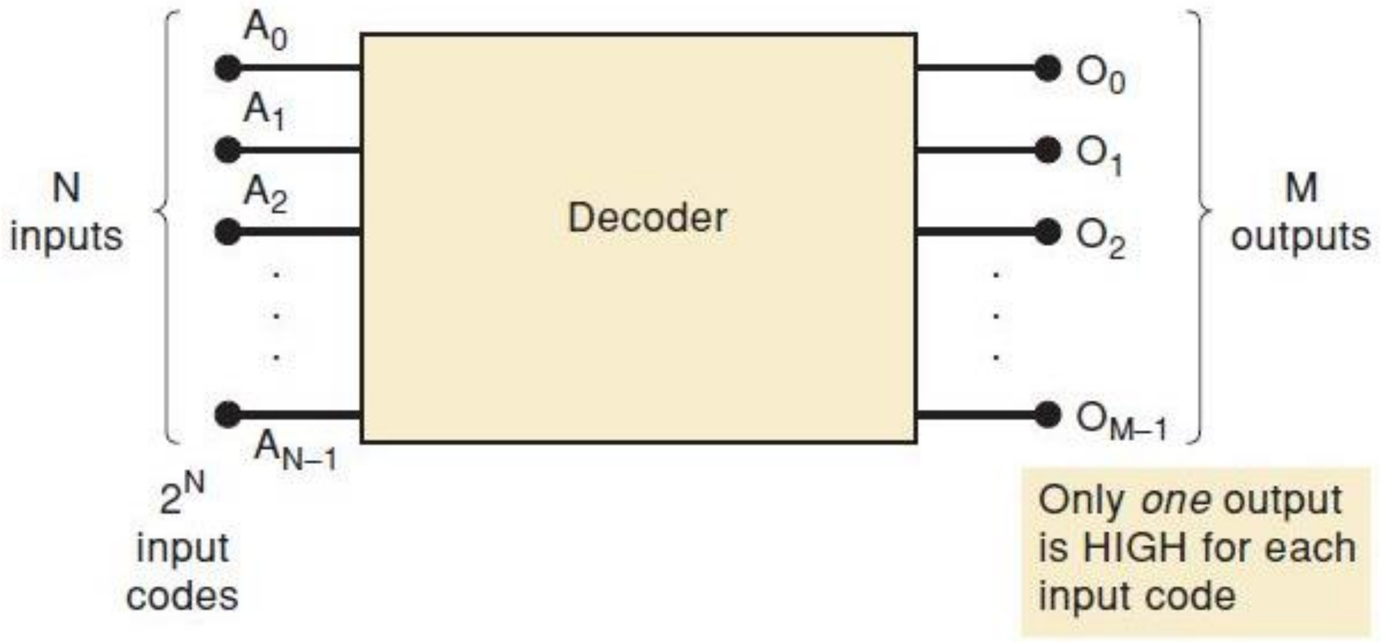
$$\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\cline{3-4}
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}$$



- Digital systems obtain binary-coded data and information that are continuously being operated on in some manner. Some of the operations include:
    - *decoding and encoding,*
    - *multiplexing,*
    - *demultiplexing,*
    - *comparison,*
    - *code conversion,*
    - *data busing.*

All of these operations and others have been facilitated by the availability of numerous ICs in the MSI (medium-scale-integration) category.

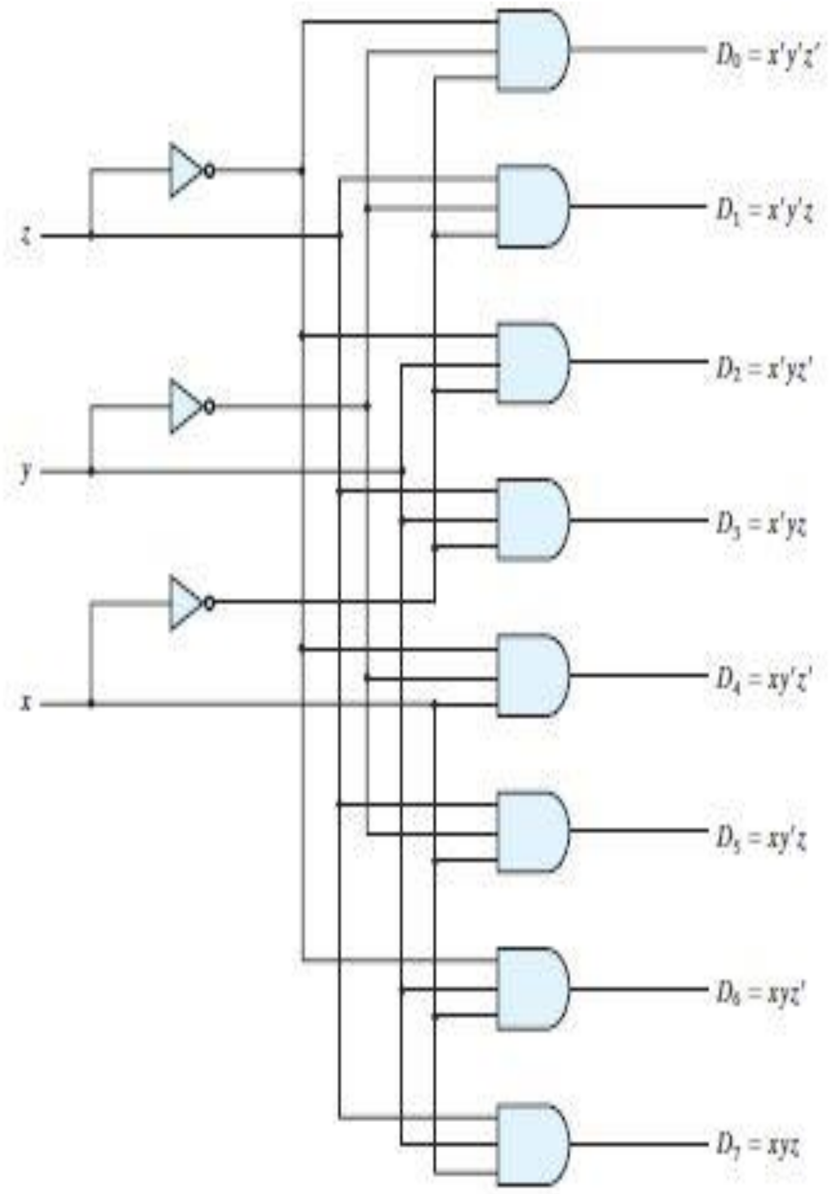
DECODERS



- A **decoder** is a logic circuit that accepts a set of inputs that represents a binary number and activates only the output that corresponds to that input number.
- A decoder circuit looks at its inputs, determines, which binary number is present there, and activates the one output that corresponds to that number; all other outputs remain inactive.

DECODERS

- Discrete quantities of information are represented in digital systems by binary codes.
- A binary code of *n*-bits is capable of representing up to 2^n distinct elements of coded information.
- A *decoder* is a combinational circuit that converts binary information from *n* input lines to a maximum of 2^n unique output lines.
- If the *n*-bit coded information has unused combinations, the decoder may have fewer than 2^n outputs.
- The decoders presented as ***n*** -to- ***m*** -line decoders.

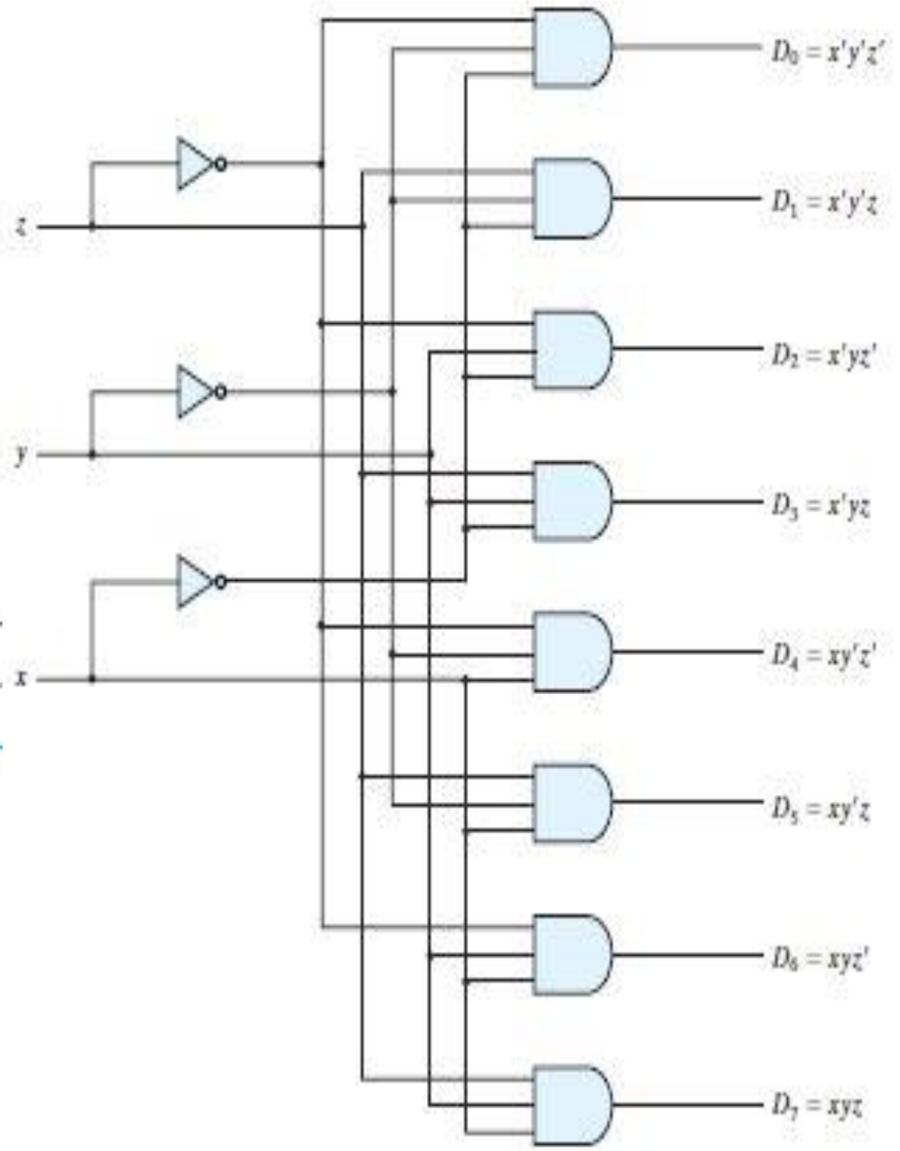


DECODERS

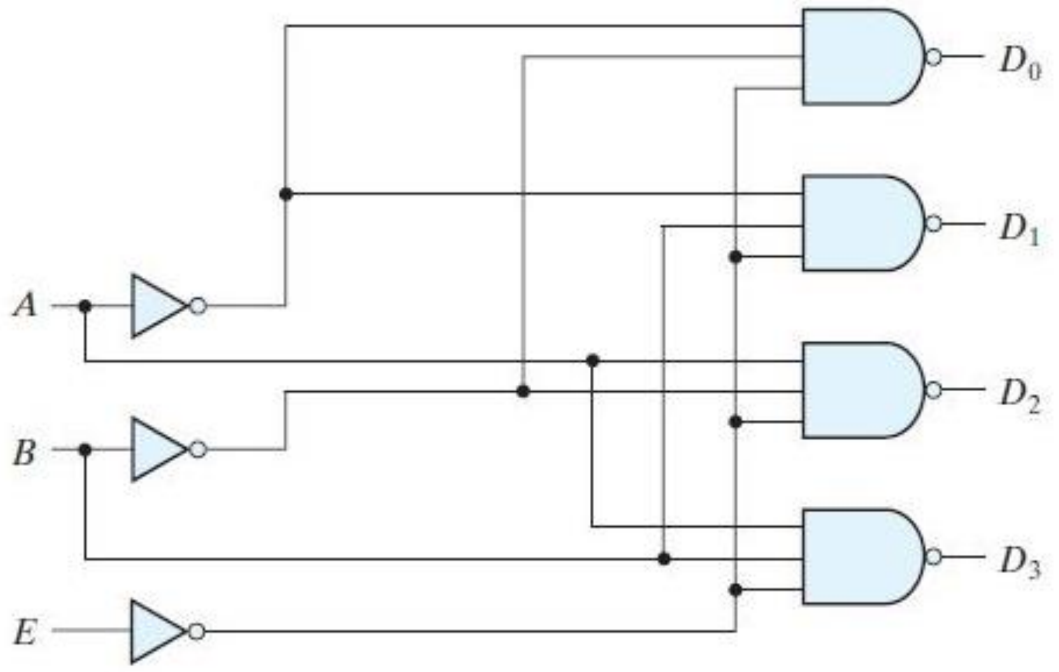
- For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1.
- The output whose value is equal to 1 represents the minterm equivalent of the binary number currently available in the input lines.

Truth Table of a Three-to-Eight-Line Decoder

Inputs			**Outputs**							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



DECODERS ARE CONSTRUCTED WITH NAND GATES



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	1

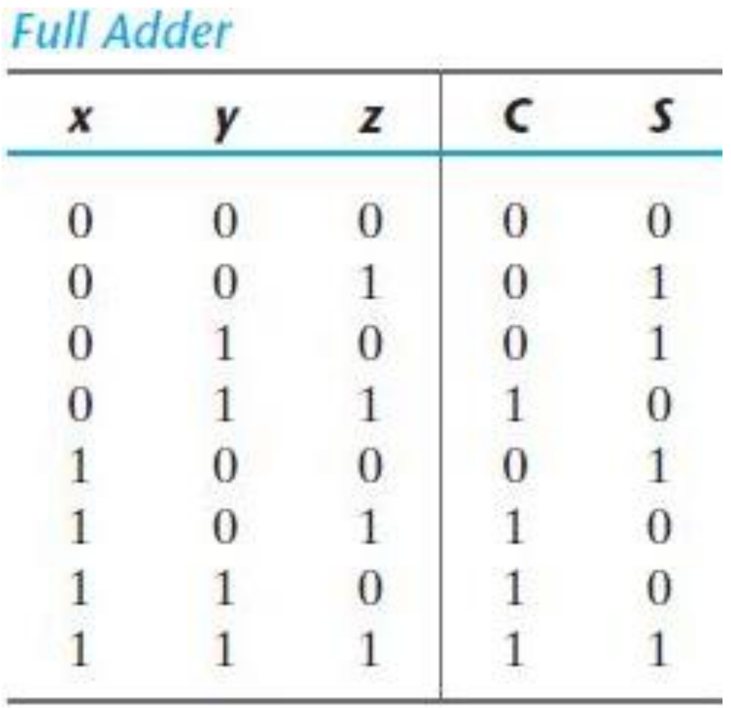
A decoder may operate with complemented or uncomplemented outputs. The enable input may be activated with a 0 or with a 1 signal.

- NAND gate produces decoder minterms in their complemented form.
- Decoders include one or more enable inputs to control circuit operation.
- Circuit operates with complemented outputs and a complement enable input.
- Decoder is enabled when E is equal to 0 (i.e., active-low enable).
- Output 0 represents minterm selected by inputs A and B .
- Circuit is disabled when E is equal to 1, regardless of the values of the other two inputs.
- When circuit is disabled, none of the outputs are equal to 0 and none of the minterms are selected.

COMBINATIONAL LOGIC IMPLEMENTATION

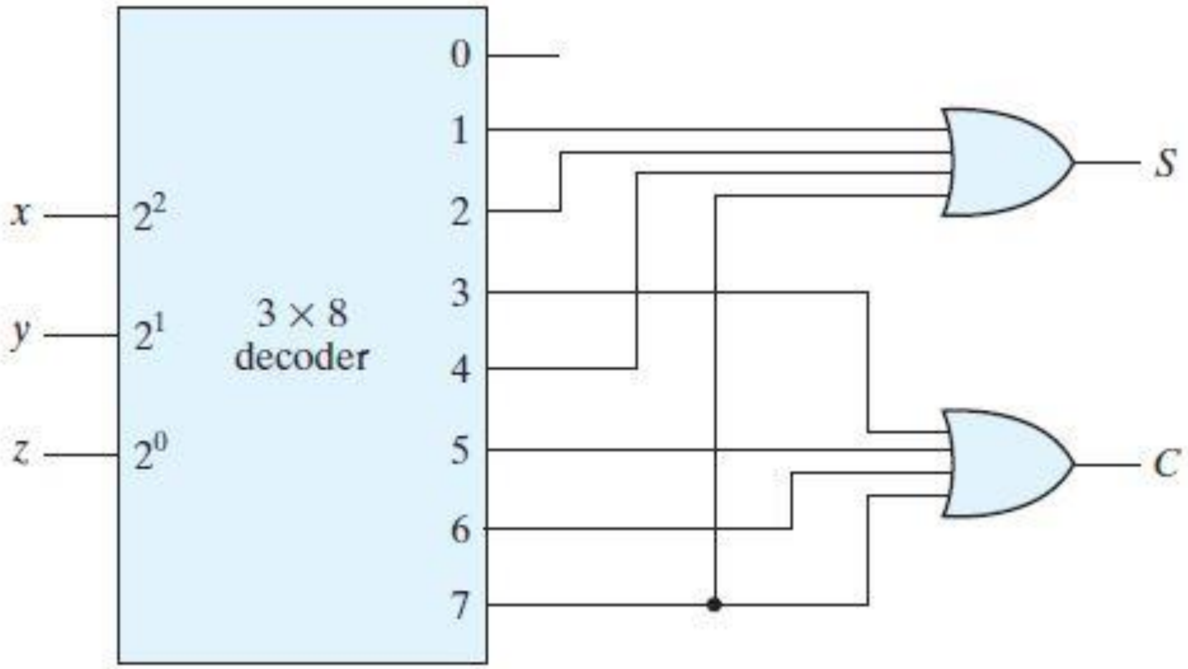
- A decoder provides the 2^n minterms of n-input variables.
- Since any Boolean function can be expressed in sum-of-minterms form, a decoder that generates the minterms of the function, together with an external OR gate that forms their logical sum, provides a hardware implementation of the function.
- *Any combinational circuit with n-inputs and m-outputs can be implemented with an n -to-2^n -line decoder and m OR gates*.
- Implementation of a combinational circuit using decoder and OR gates requires Boolean function for the circuit expressed as a sum of minterms.
- A decoder is then chosen that generates all the minterms of the input variables.
- The inputs to each OR gate are selected from the decoder outputs according to list of minterms of each function.

IMPLEMENTATION OF FULL-ADDER CIRCUIT



Full Adder

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



- Functions for full adder combinational circuit in sum-of minterms form:

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$

$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

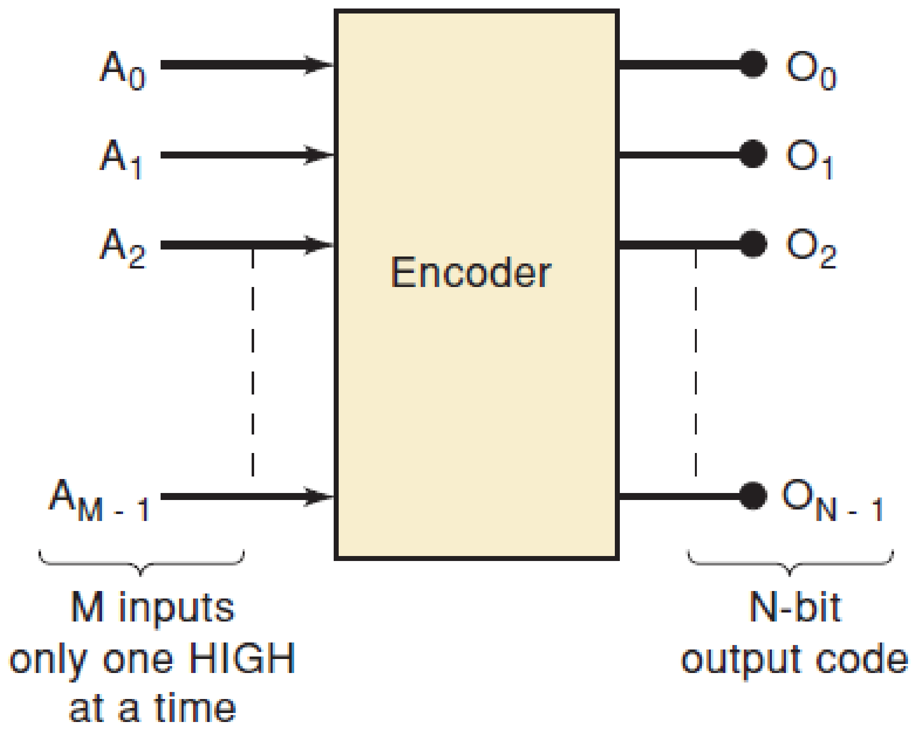
- Decoder generates eight minterms for *x* , *y* , and *z* . The OR gate for output *S* forms the logical sum of minterms 1, 2, 4, and 7.
- The OR gate for output *C* forms the logical sum of minterms 3, 5, 6, and 7.

DECODER APPLICATIONS

- Decoders are used whenever an output or a group of outputs is to be activated only on the occurrence of a specific combination of input levels.
- These input levels are often provided by the outputs of a counter or a register.
- Decoders are widely used in the memory system of a computer where they respond to the address code generated by the central processor to activate a particular memory location.
- Each memory IC contains many registers that can store binary numbers (data).
- Each register needs to have its own unique address to distinguish it from all the other registers.
- A decoder is built into the memory IC’s circuitry and allows a particular storage register to be activated when a unique combination of inputs (i.e., its address) is applied.
- In a system, there are usually several memory ICs combined to make up the entire storage capacity.
- A decoder is used to select a memory chip in response to a range of addresses by decoding the most significant bits of the system address and enabling (selecting) a particular chip

ENCODERS

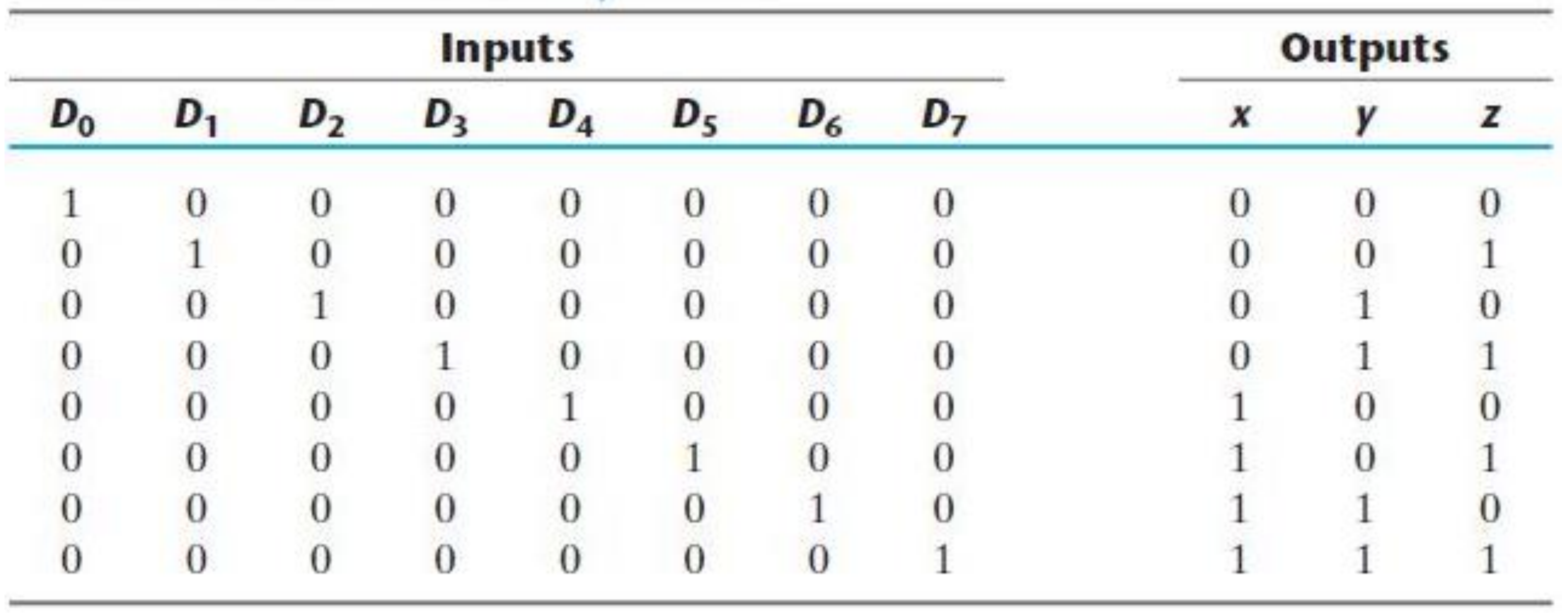
- Decoders accept an input code and produce a HIGH (or a LOW) at *one and only one* output line.
- Decoder identifies, recognizes, or detects a particular code.
- The opposite of this decoding process is called **encoding** and is performed by a logic circuit called an **encoder**.
- An encoder has a number of input lines, only one of which is activated at a given time, and produces an *N*-bit output code, depending on which input is activated.



- A *binary-to-octal decoder (3-line-to-8-line decoder)* accepts a three-bit input code and activates one of eight output lines corresponding to that code.
- An *octal-to-binary encoder (8-line-to-3-line encoder)* performs the opposite function: it accepts eight input lines and produces a three-bit output code corresponding to the activated input.

ENCODERS

- An encoder has 2^n (or fewer) input lines and n output lines.
- The output lines, as an aggregate, generate the binary code corresponding to the input value. Ex. 8-to-3 encoder.
- It has eight inputs (one for each of the octal digits) and three outputs that generate the corresponding binary number.
- It is assumed that only one input has a value of 1 at any given time.


Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

ENCODERS

- Encoder can be implemented with OR gates whose inputs are determined directly from the truth table.
- Output *z* is equal to 1 when the input is 1, 3, 5,or 7.
- Output *y* is 1 for 2, 3, 6, or 7, and output *x* is 1 for digits 4, 5, 6, or 7.
- Boolean output functions:

$x = D_4 + D_5 + D_6 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$

Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

ENCODERS

- Encoder (8-to-3 line) has limitation that only one input can be active at any given time.
- If two inputs are active simultaneously, output produces an undefined combination.
- For example, if D_3 and D_6 are 1 simultaneously, output of encoder will be 111 because all three outputs are equal to 1.
- The output 111 does not represent either binary 3 or binary 6.
- To resolve this, encoder circuits must establish an input priority to ensure that only one input is encoded.
- If we establish a higher priority for inputs with higher subscript numbers, and if both D_3 and D_6 are 1 at the same time, the output will be 110 because D_6 has higher priority than D_3.
- Another ambiguity in the 8-to-3 encoder is that an output with all 0's is generated when all the inputs are 0; but this output is the same as when D_0 is equal to 1.
- The discrepancy can be resolved by providing one more output to indicate whether at least one input is equal to 1.

Priority Encoder

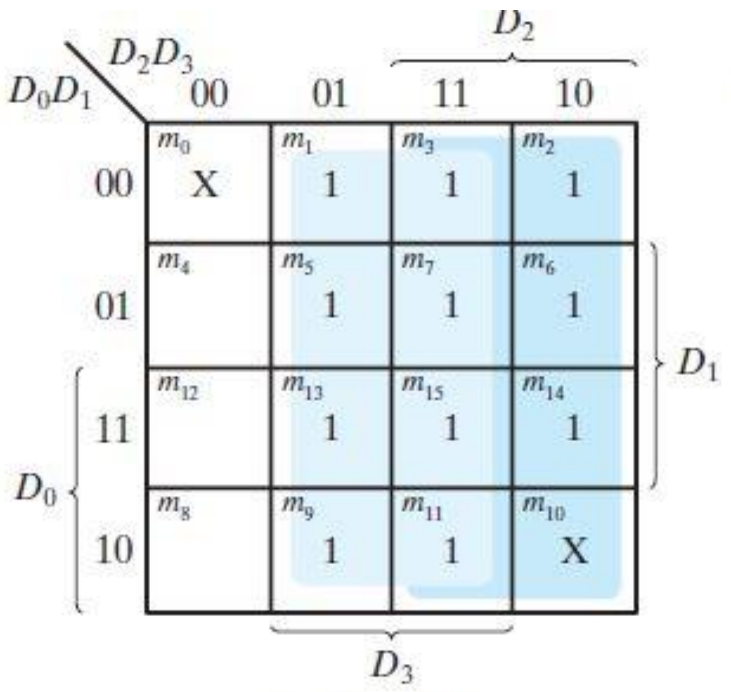
- A priority encoder is an encoder circuit that includes the priority function.
- If two or more inputs are equal to 1 at the same time, input having the highest priority will take precedence.
- In addition to two outputs *x* and *y*, circuit has a third output designated by *V.*

Truth Table of a Priority Encoder

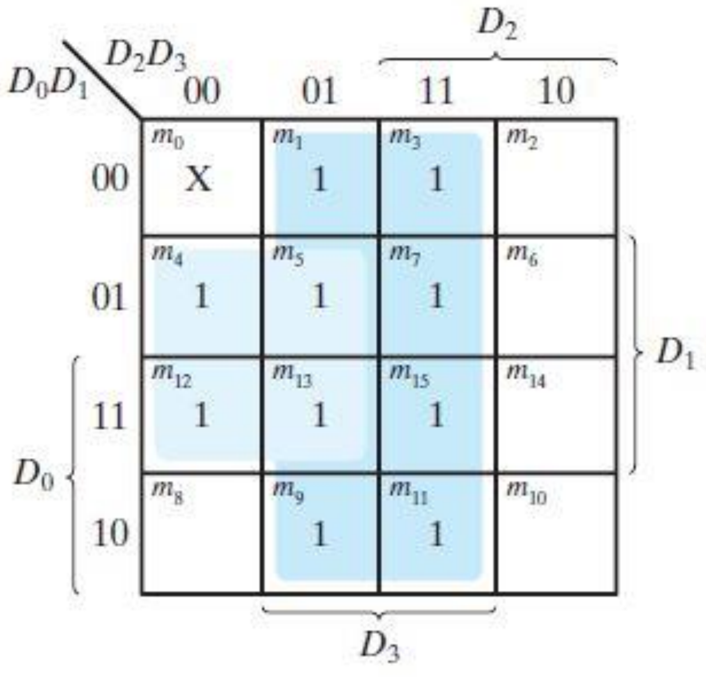
Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

- *Valid* bit indicator is set to 1, when one or more inputs are equal to 1.
- If all inputs are 0, there is no valid input and *V* is equal to 0.
- Other two outputs are not inspected when V equals 0 and are specified as don't-care conditions.
- X 's in output columns represent don't-care conditions.
- X 's in the input columns are useful for representing a truth table in condensed form.
- Instead of listing all 16 minterms of four variables, truth table uses an X to represent either 1 or 0. Ex. X100 represents two minterms 0100 and 1100.

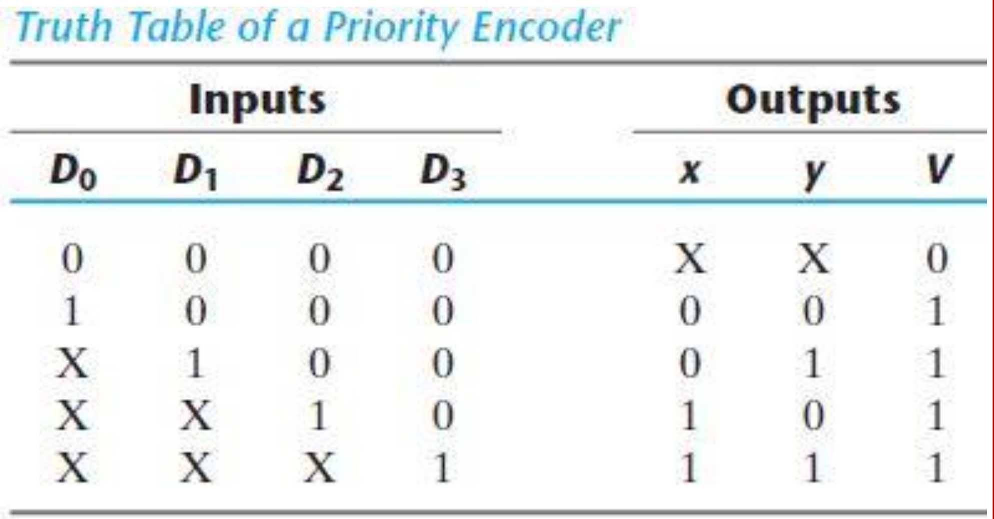
Priority Encoder



$x = D_2 + D_3$

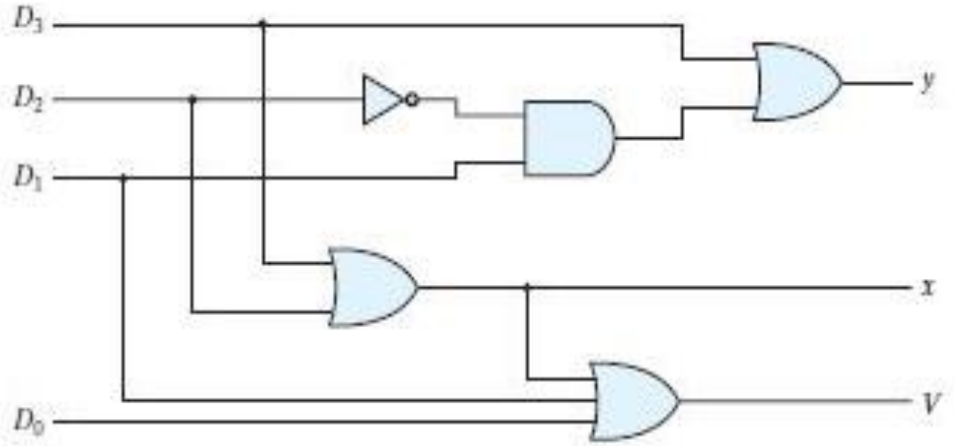


$y = D_3 + D_1D'_2$



Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1



- The simplified Boolean expressions for the priority encoder are obtained from the maps.

  $x = D_2 + D_3$

  $y = D_3 + D_1 D_2'$

  $V = D_0 + D_1 + D_2 + D_3$
- Condition for output *V* is an OR function of all input variables.