數位邏輯設計

4.2 Combinational Circuits

主講者：吳順德

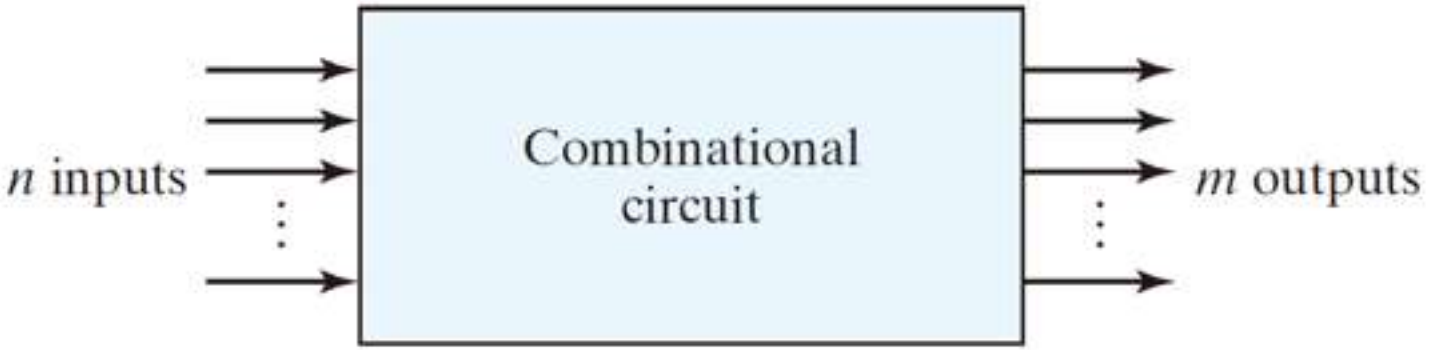
國立臺灣師範大學機電工程系 副教授

Combinational & Sequential Logic

- A *Combinational Circuit* consists of Logic gates whose **outputs at any time are determined from only the present value of inputs**.
- A *Sequential Circuit* consists of Logic gates whose **outputs at any time are determined from not only the present value of inputs, but also on past input**.
- Combinational circuits employ **only logic gates**.
- Sequential circuits employ **storage elements** in addition to **logic gates**.

Combinational Circuit

- The diagram of a combinational circuit has logic gates with **no feedback paths or memory elements**.
- Characteristics of the following diagram
  - 2^n possible combinations of input values.
  - Described by *m* Boolean functions and each function is expressed in terms of the n input variables.

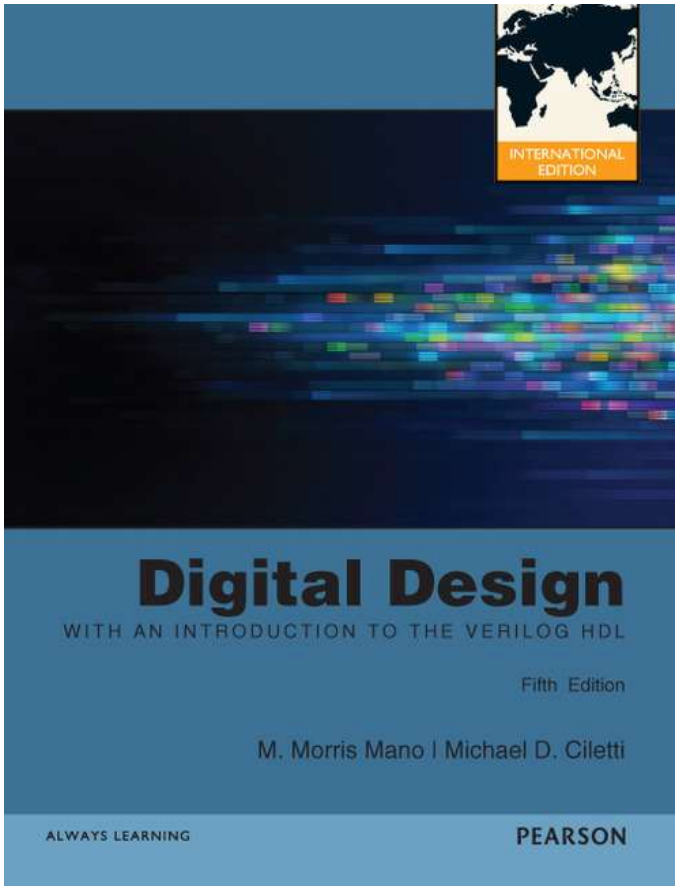


Important Standard Combinational Circuit

- Standard combinational circuit
  - adders, subtractors, multiplier
  - comparators,
  - decoders, encoders,
  - multiplexers
- The above components are available in integrated circuits as medium scale integration (MSI) circuit.
- They are also standard cells in complex very large scale integrated (VLSI) circuit.

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.3 Analysis Procedure

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

Analysis Procedures of Combinational Circuit

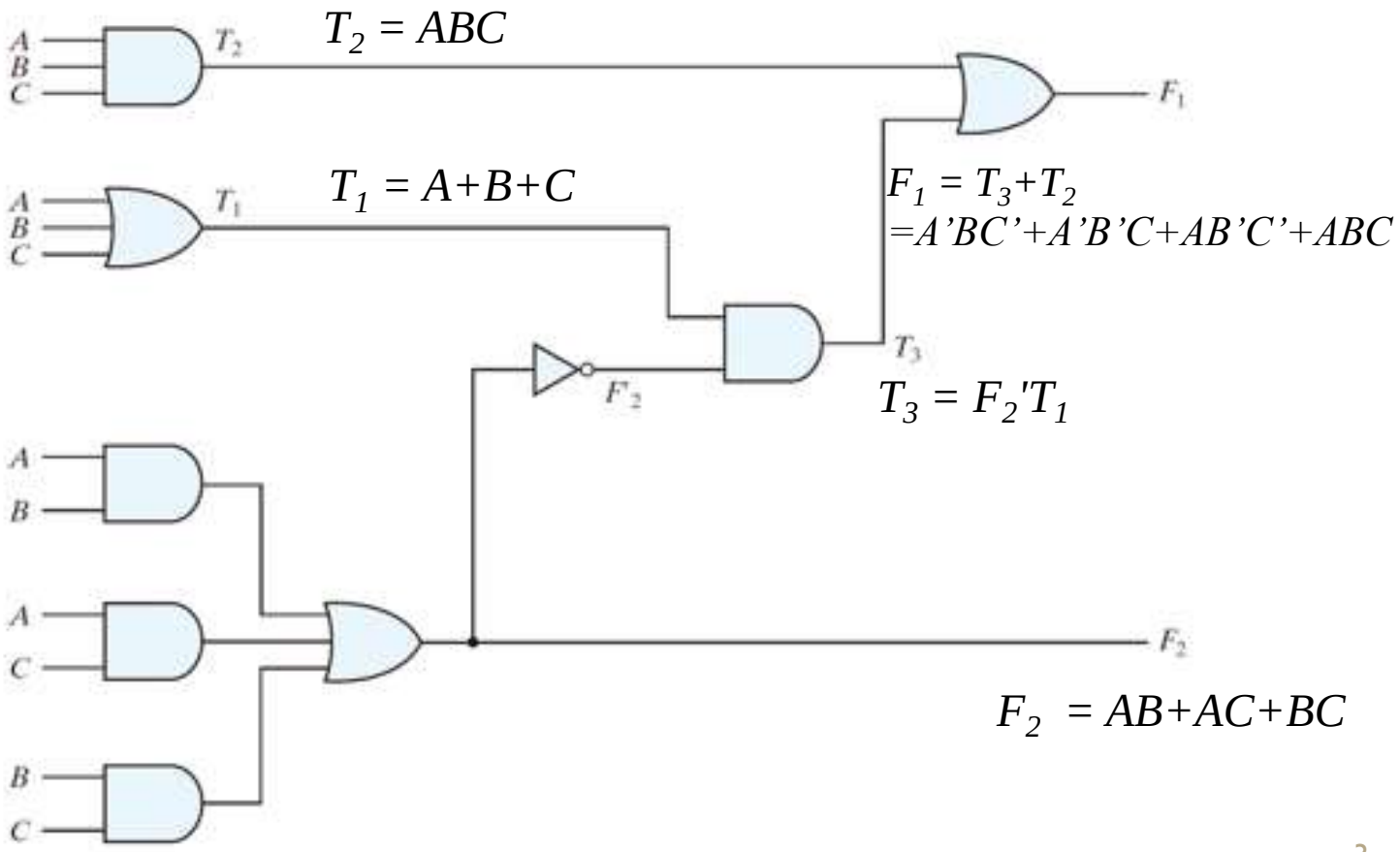
- Step 1:
  - make sure that it is combinational not sequential
    - No feedback path or memory elements.

- Step 2.
  - derive its Boolean functions or truth table.

Examples

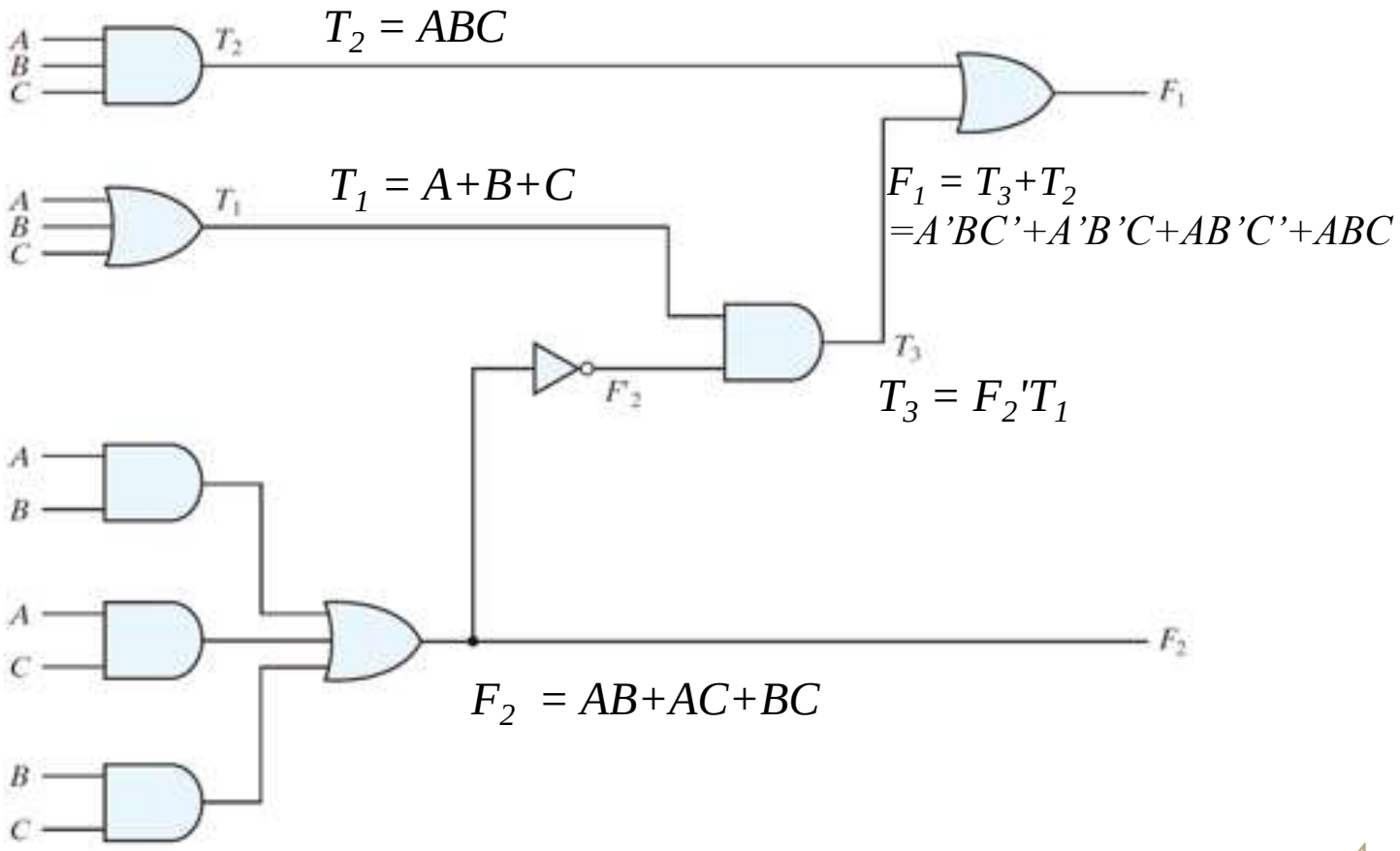
- Step 1: This circuit is a combinational circuit.

No feedback! No register!



Derive Boolean Function

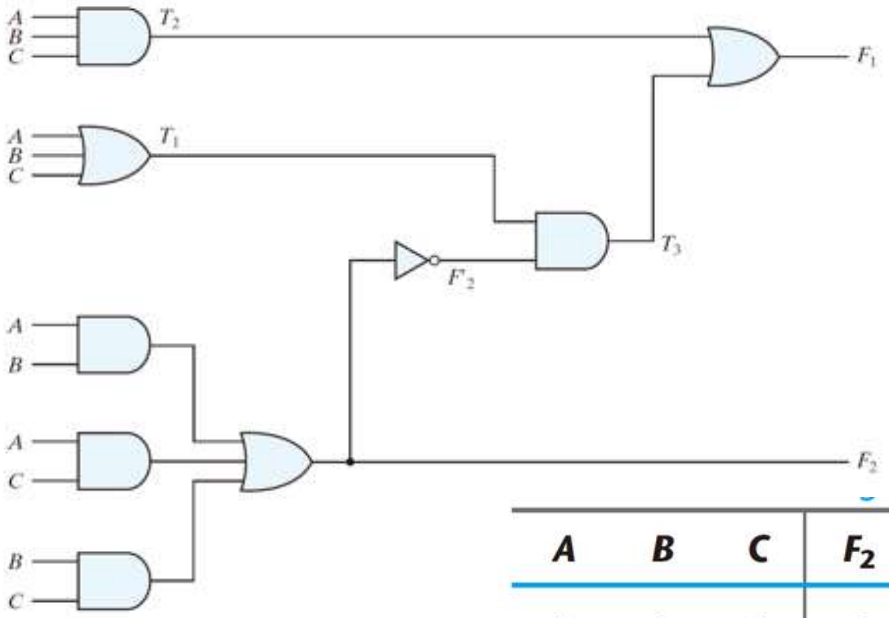
- Step 2:



Full Adder!

Derive Truth Table

- Step 2:

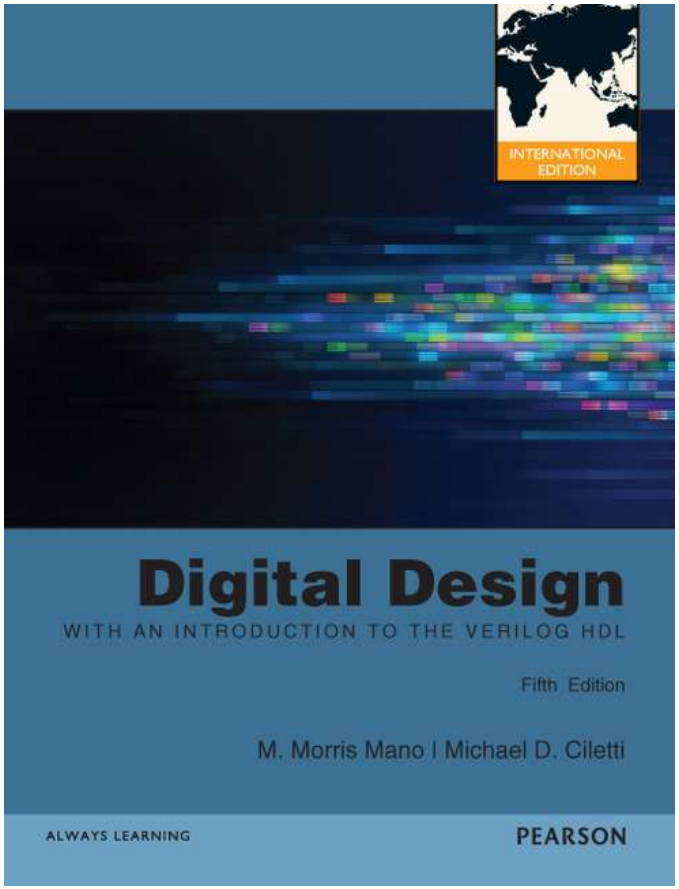


A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Full Adder!

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.4 Design Procedure

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

Design procedure of combinational circuit

1. Determine inputs and outputs from the specifications and assign a symbol to each.
2. Derive the truth table that defines the relationship between inputs and outputs.
3. Obtain the simplified Boolean functions for each output (by K-Map).
4. Draw the logic diagram and verify the correctness of the design (manually or by simulation).

Code Conversion Example

- Convert 4bits binary coded decimal (BCD) to excess-3 code.
- Inputs: bits of BCD code.
  - A,B,C,D
- Output: bits of excess-3 code.
  - x,y,z,w

Code Conversion Example

- Step 1:
  - Determine inputs and outputs from the specifications and assign a symbol to each.

Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*

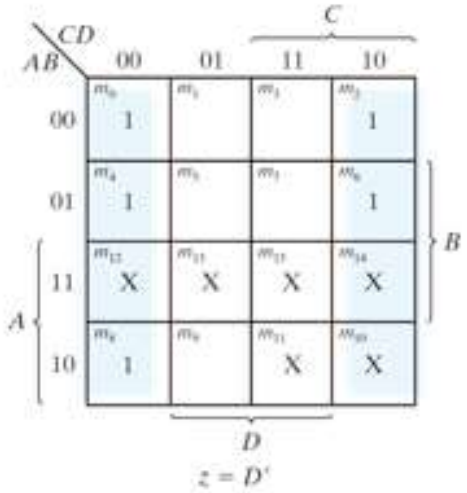
Code Conversion Example

- Step 2:
  - Derive the truth table that defines the relationship between inputs and outputs.

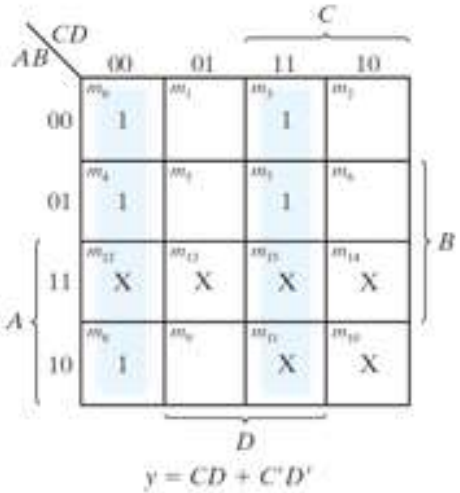
Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Code Conversion Example

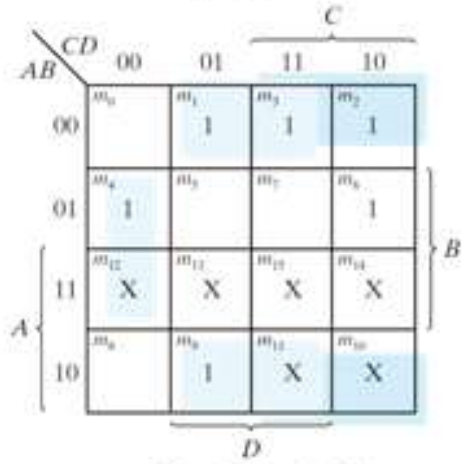
- Step3:
  - Obtain the simplified Boolean functions for each output.



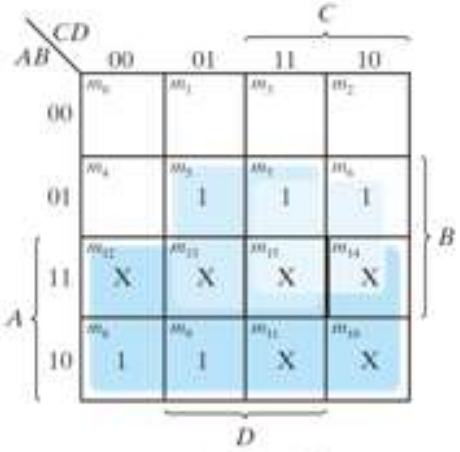
$z = D'$



$y = CD + C'D'$



$x = B'C + B'D + BC'D'$



$w = A + BC + BD$

Code Conversion Example

- The simplified functions
  - $z = D'$

    $y = CD + C'D'$

    $x = B'C + B'D + BC'D'$

    $w = A + BC + BD$

Two-level
AND gates: 7
OR gates: 3

- Another implementation
  - $z = D'$

    $y = CD + C'D' \quad = CD + (C+D)'$

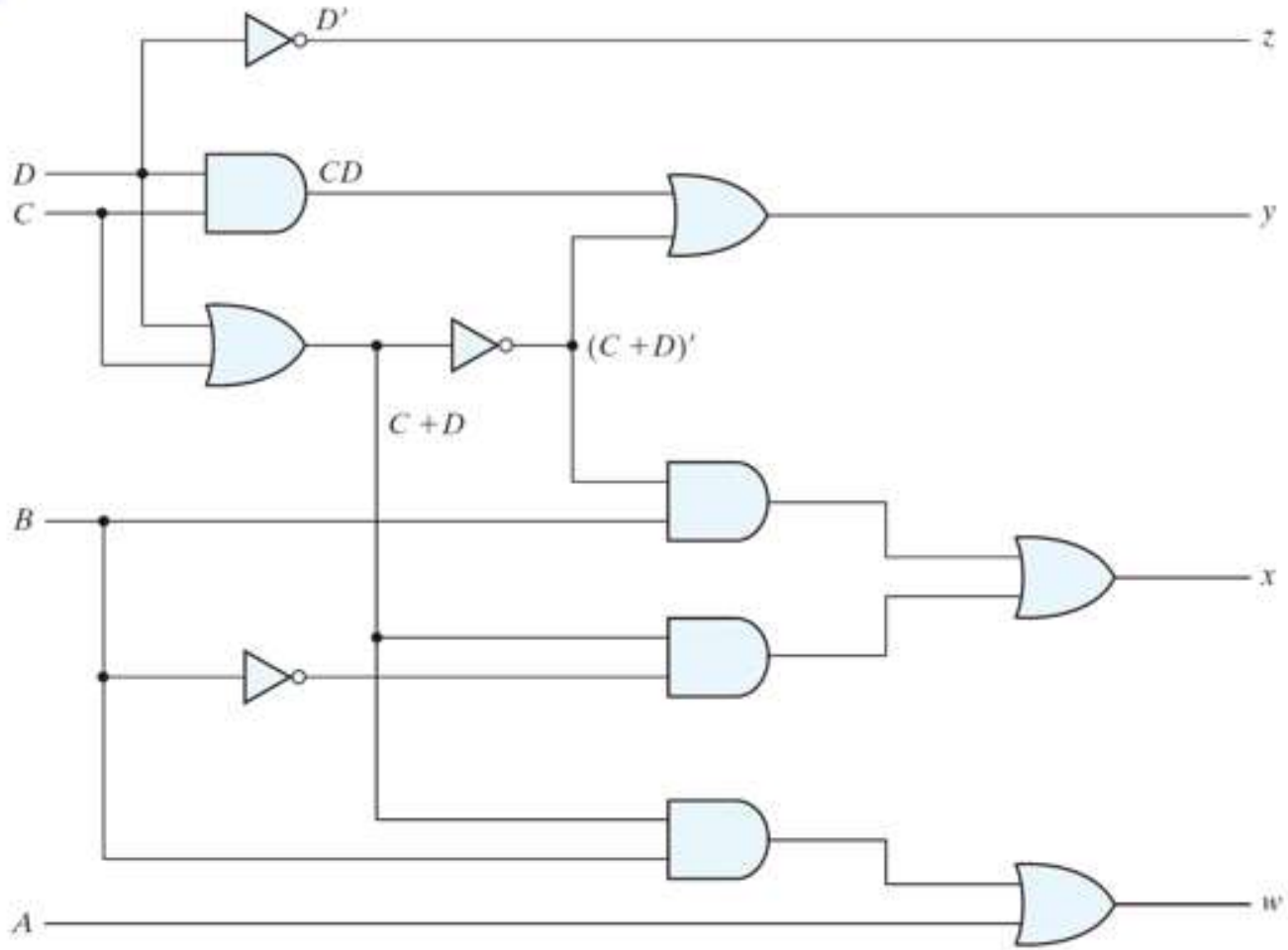
    $x = B'C + B'D + BC'D' = B'(C+D) + B(C+D)'$

    $w = A + BC + BD = A + B(C+D)$

Three-level
AND gates: 4
OR gates: 4

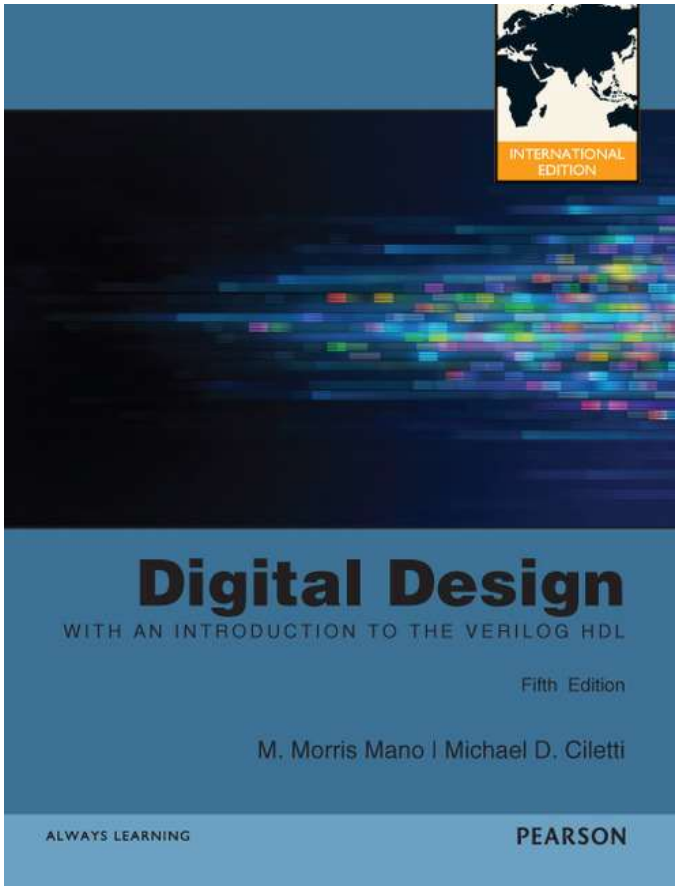
Code Conversion Example

- Logic Diagram for BCD-to-excess-3 Code Converter



Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.5.1 Half Adder and Full Adder

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

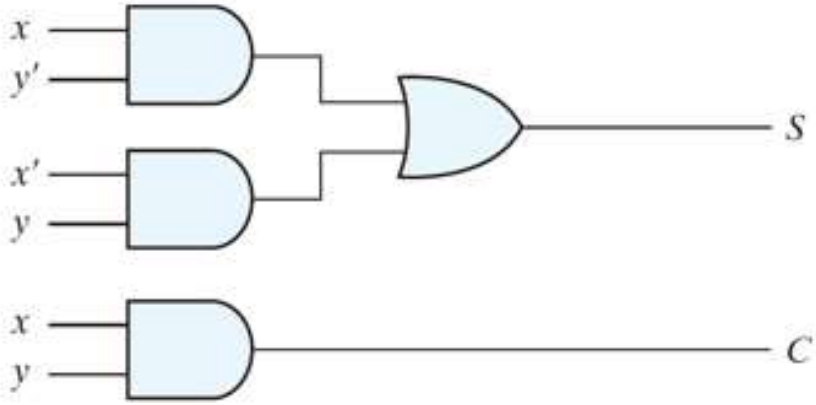
Half Adder

- Function
  - 0 + 0 = 0 ; 0 + 1 = 1 ; 1 + 0 = 1 ; 1 + 1 = 10
- Inputs
  - *x*, *y*
- outputs
  - *C* (carry)
  - *S* (sum)
- truth table

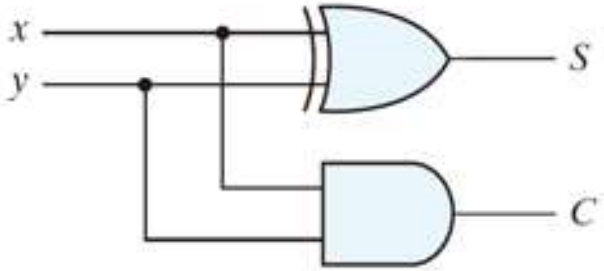
x	*y*	*C*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half Adder

- Boolean Functions
  - $S = x'y + xy' = x \oplus y$
  - $C = xy$
- Implementations



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

Full Adder

- The arithmetic sum of three input bits
- input bits
  - x, y: two significant bits
  - z: the carry bit from the previous lower significant bit
- output bits:
  - C (carry)
  - S (sum)

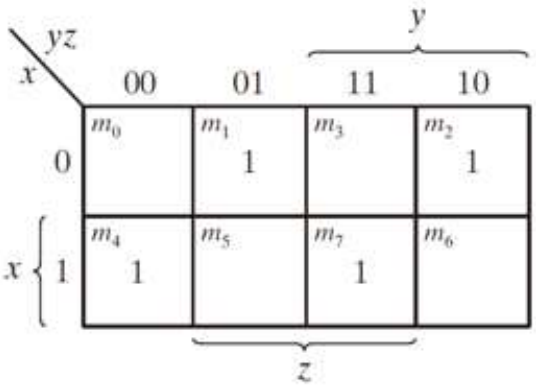
Full Adder

- Truth Table

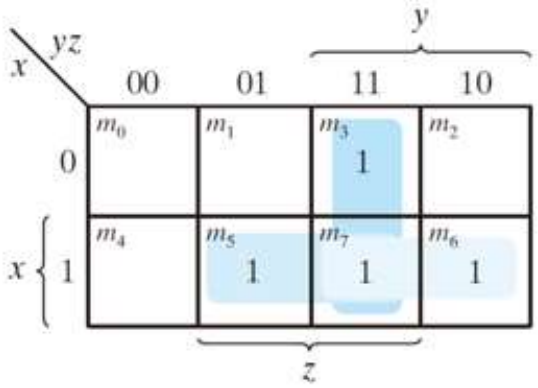
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full Adder

- Simplified the Boolean Functions

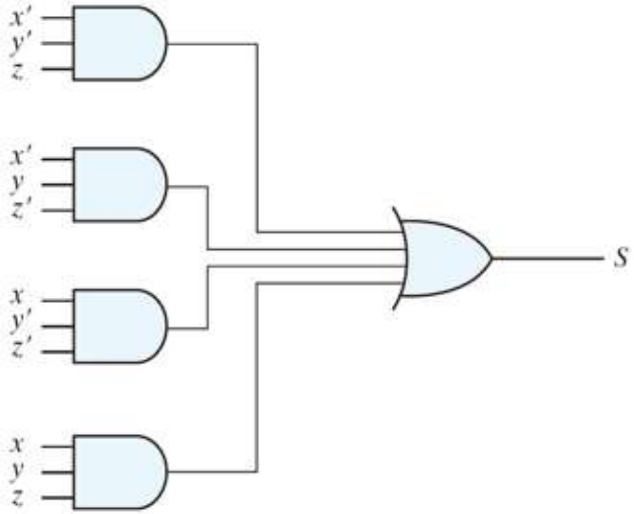


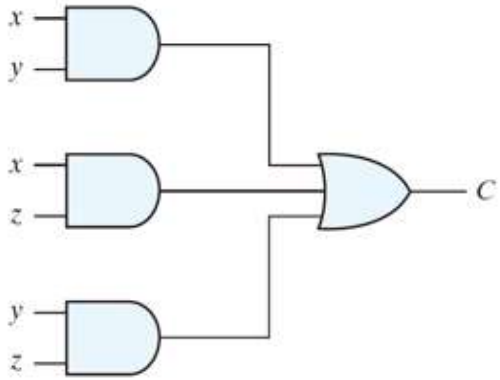
(a) $S = x'y'z + x'yz' + xy'z' + xyz$



(b) $C = xy + xz + yz$

- Implementation

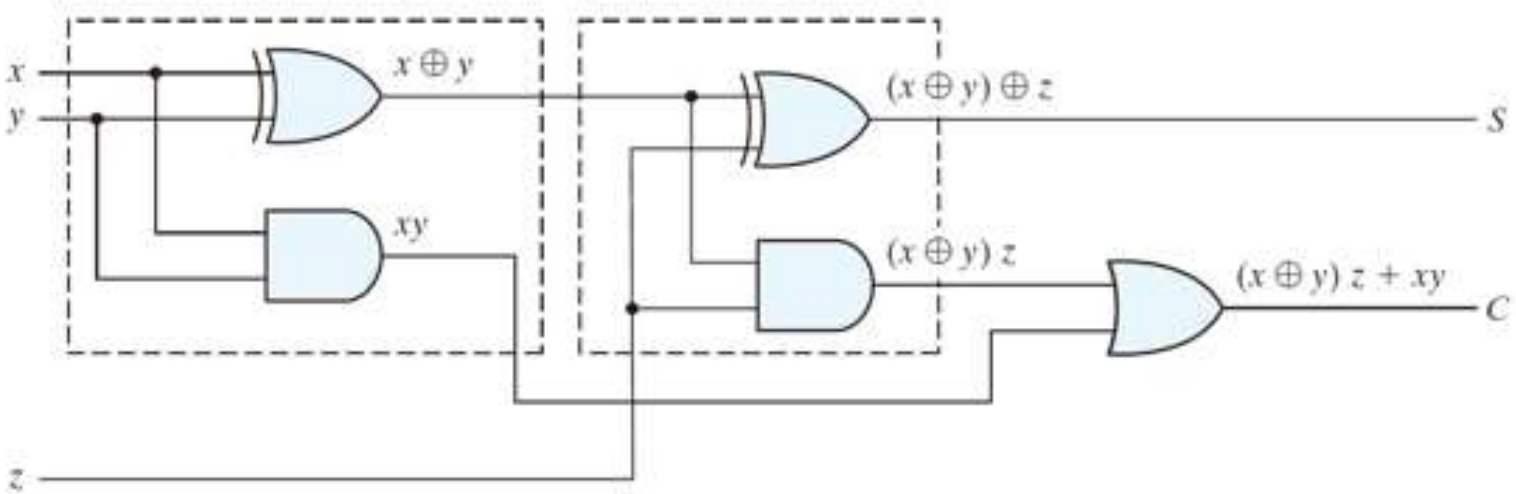




Full Adder

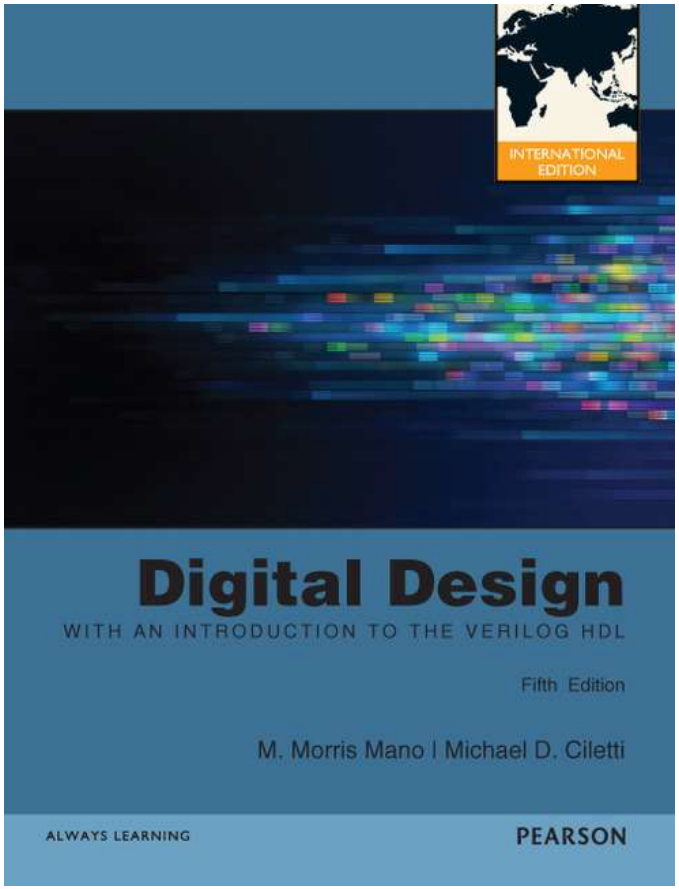
- Simplified the Boolean Functions
  - $S = x'y'z+x'yz'+ xy'z'+xyz = z\oplus(x\oplus y)$
  - $C = xy + xz + yz = z(x\oplus y) +xy$

- Implementation



Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

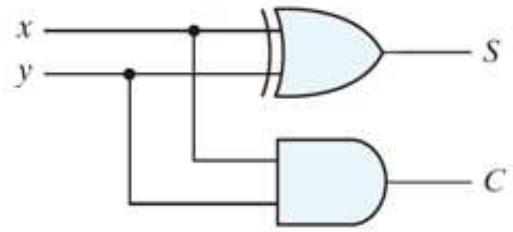
4.5.1(補充)Half Adder and Full Adder

主講者：吳順德

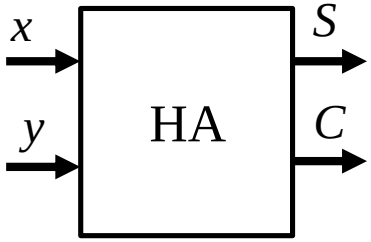
國立臺灣師範大學機電工程系 副教授

Full Adder & Half Adder

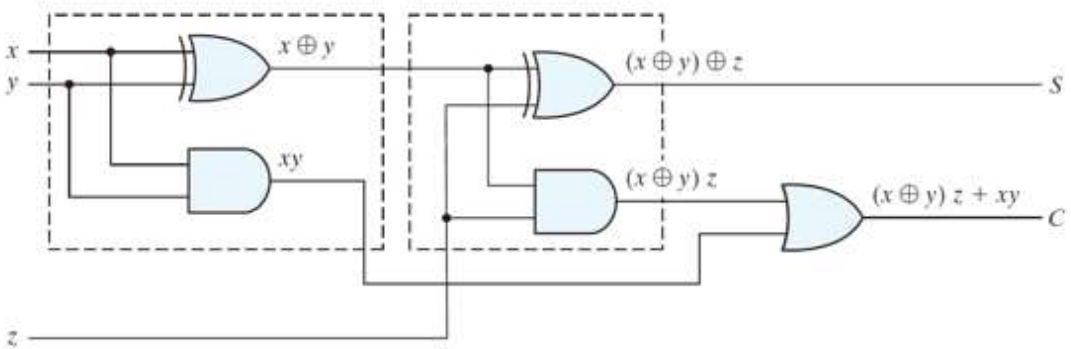
- Half Adder

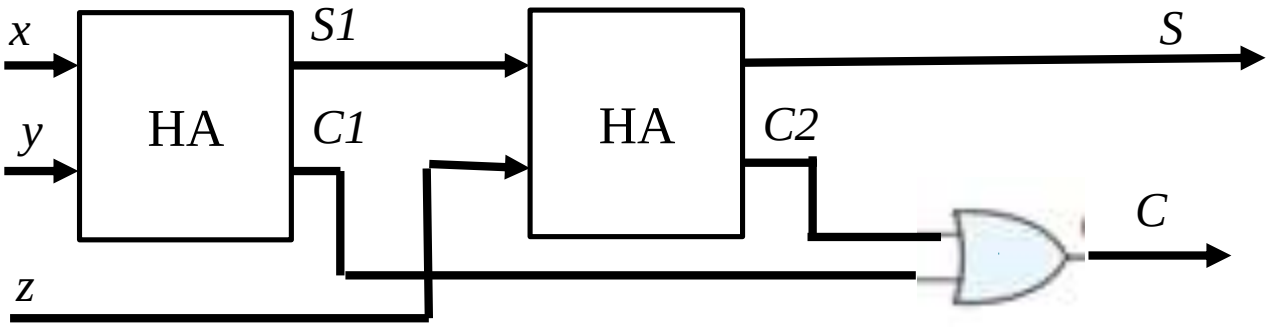


(b) $S = x \oplus y$
$C = xy$



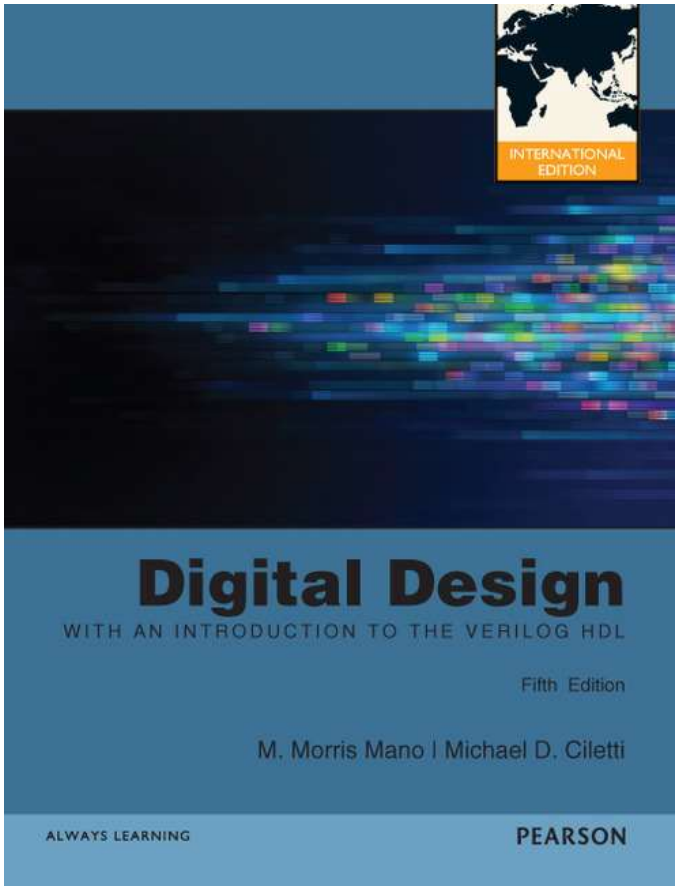
- Full Adder





Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.5.2 Binary Adder

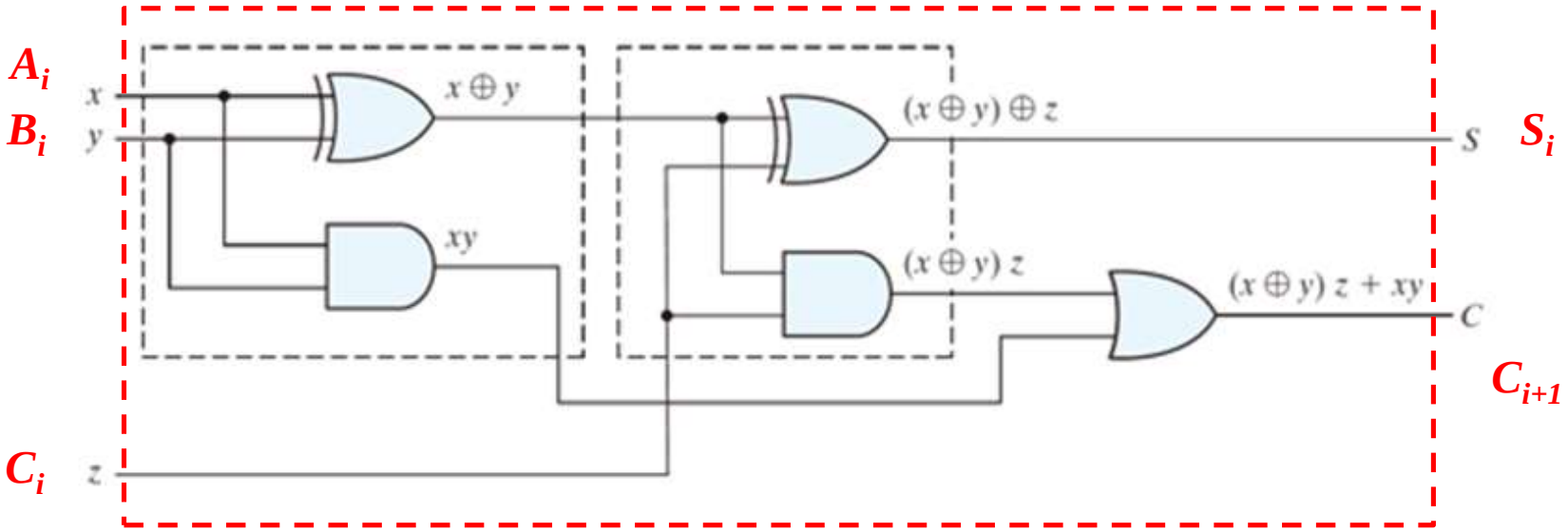
主講者：吳順德

國立臺灣師範大學機電工程系 副教授

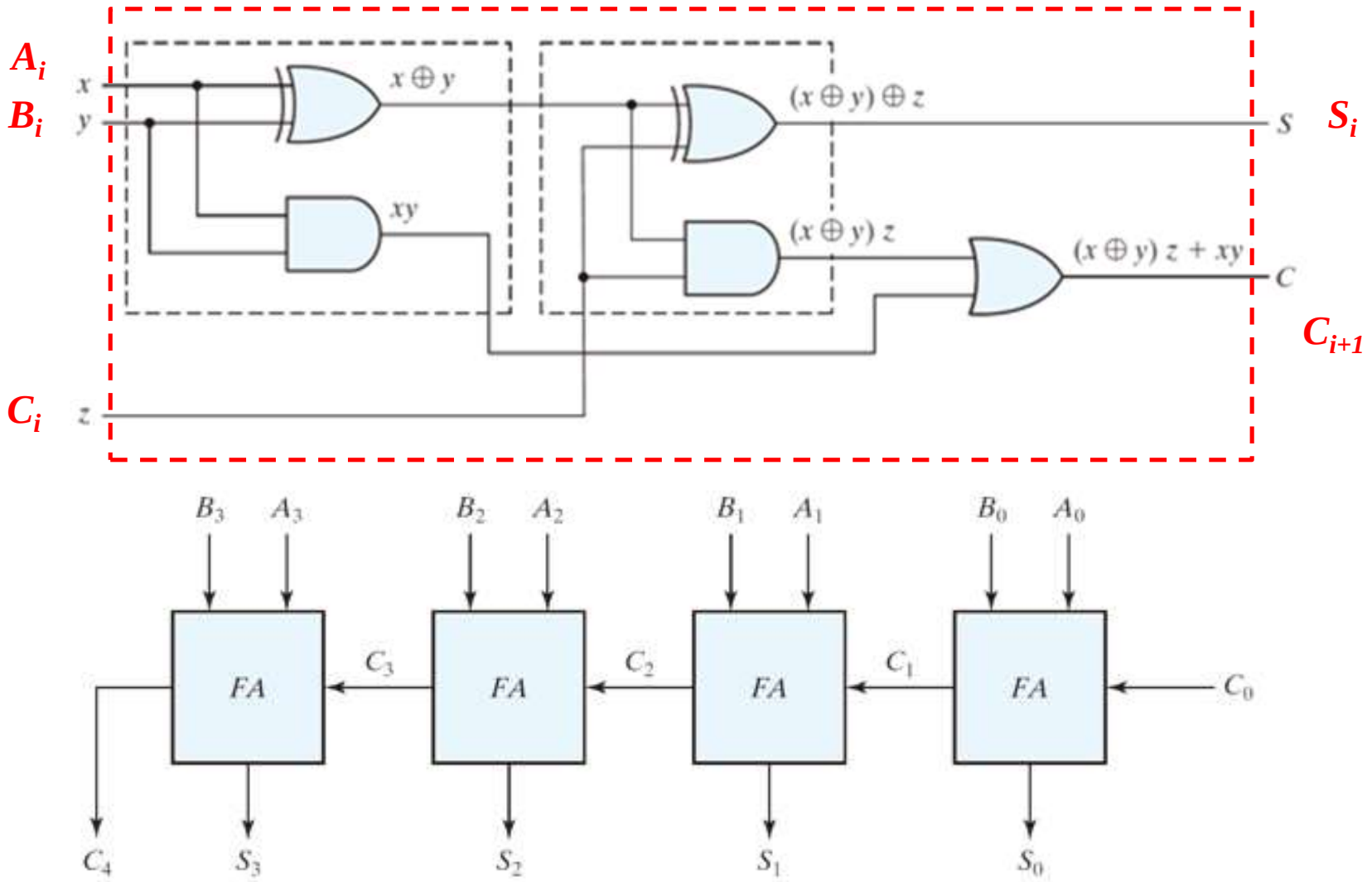
Full Adder & Binary Adder

- 4 bit adder

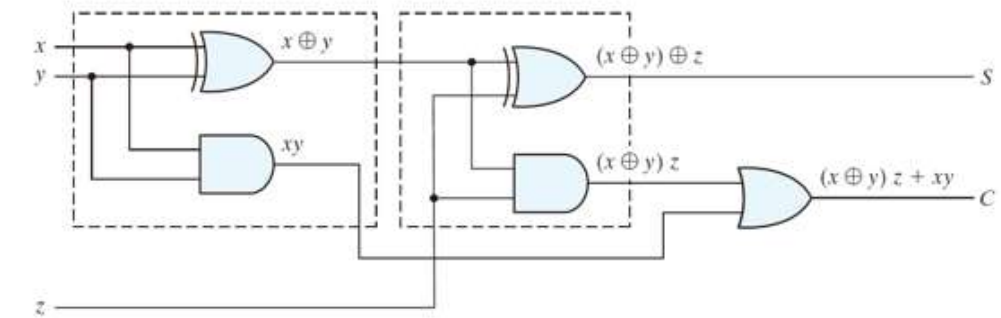
Subscript *i*:	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

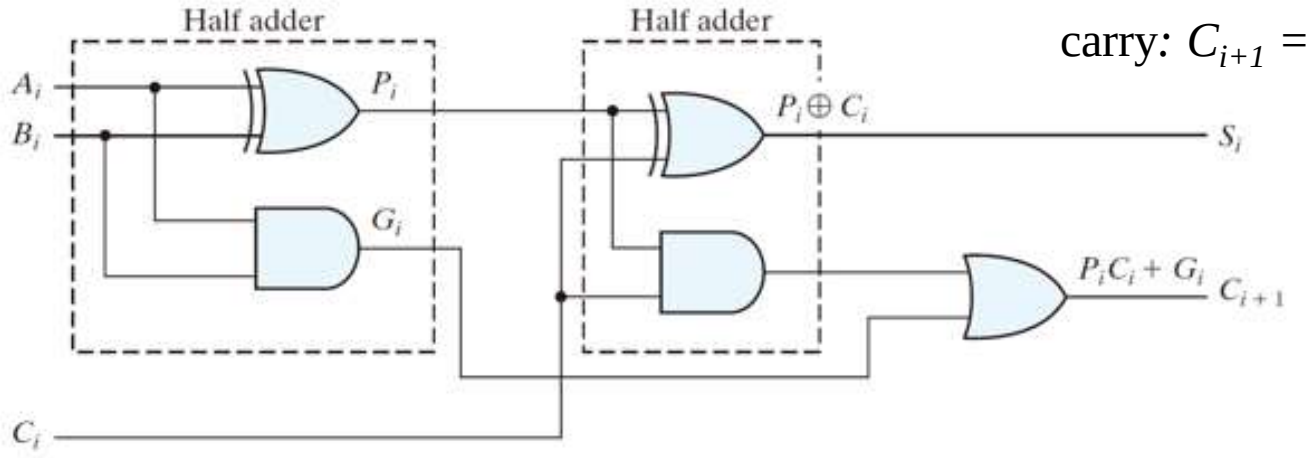


4 bits Binary Adder



Carry Propagation





carry propagate: $P_i = A_i \oplus B_i$

carry generate: $G_i = A_i B_i$

sum: $S_i = P_i \oplus C_i$

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

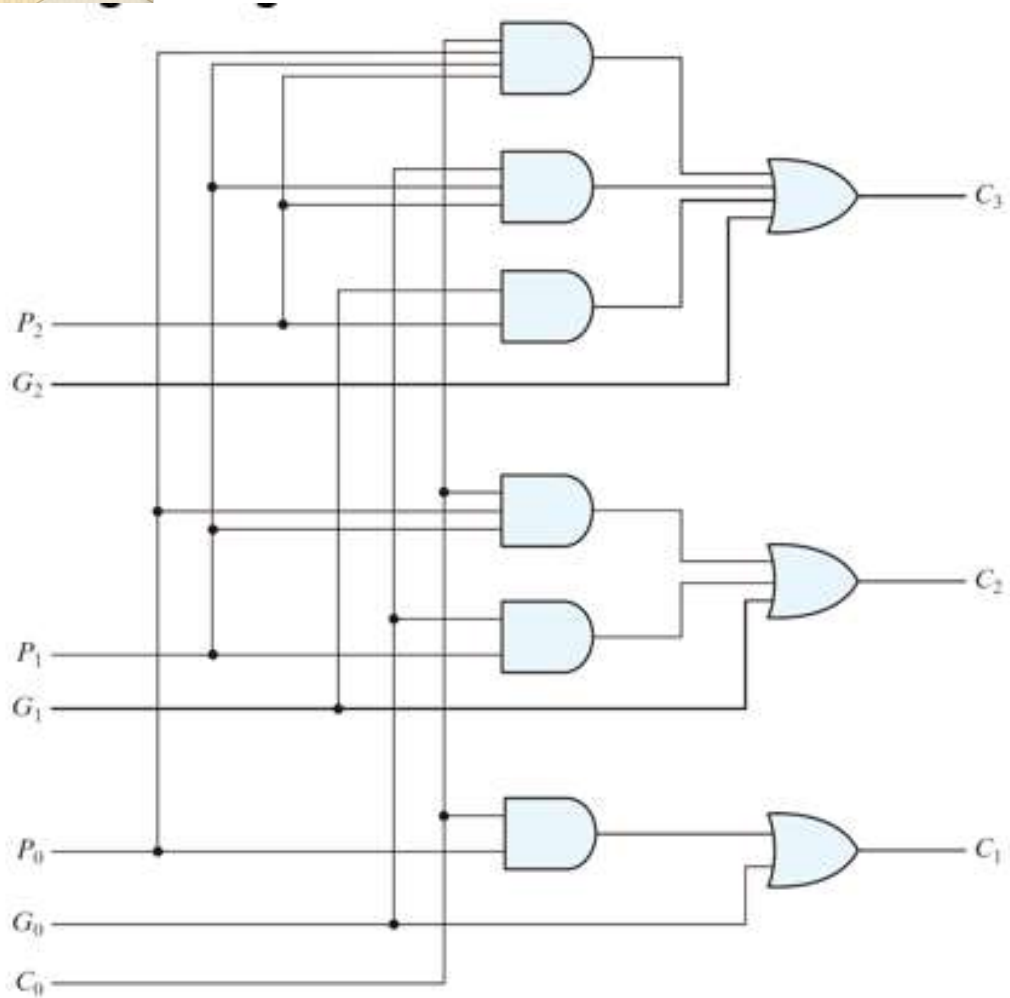
C_i to C_{i+1} : need 2 gate level; C_0 to C_4 : need 8 gate level

For an n-bits adder, there are 2n gate level for the carry to propagate from input to output.

Carry Look Ahead

- Purpose:
  - Reduce the carry propagation delay
- Formula
  - carry propagate: $P_i = A_i \oplus B_i$
  - carry generate: $G_i = A_i B_i$
  - sum: $S_i = P_i \oplus C_i$
  - carry: $C_{i+1} = G_i + P_i C_i$
- Boolean functions of Carries
  - C_0 = input carry
  - $C_1 = G_0 + P_0 C_0$
  - $C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$
  - $C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

Carry Lookahead Generator



C_0 = input carry

$C_1 = G_0+P_0C_0$

$C_2 = G_1+P_1C_1 = G_1+P_1(G_0+P_0C_0)$

$= G_1+P_1G_0+P_1P_0C_0$

$C_3 = G_2+P_2C_2$

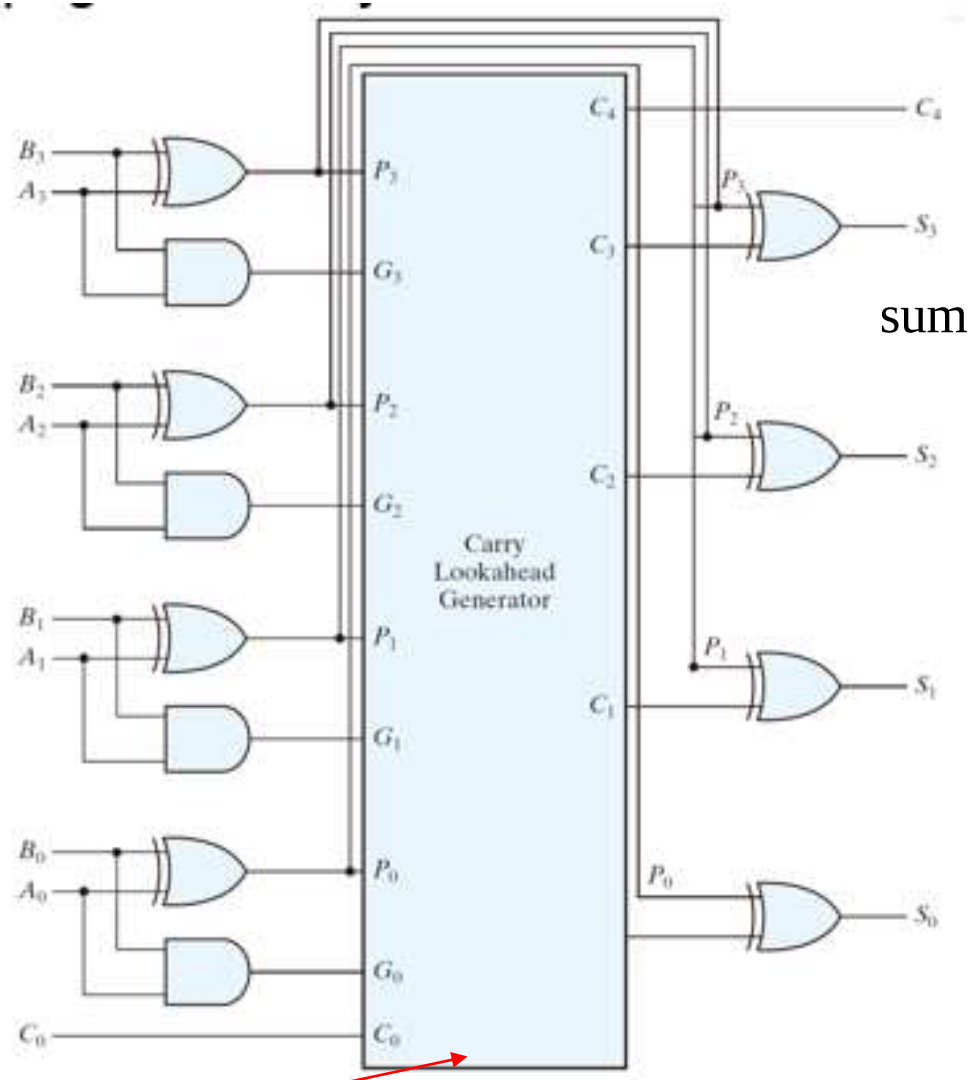
$= G_2+P_2G_1+P_2P_1G_0+ P_2P_1P_0C_0$

Only 2 gate levels for the carry to propagate from input to output!

4-bit carry-look ahead adder

carry propagate: $P_i = A_i \oplus B_i$

carry generate: $G_i = A_i B_i$



sum: $S_i = P_i \oplus C_i$

C_0 = input carry

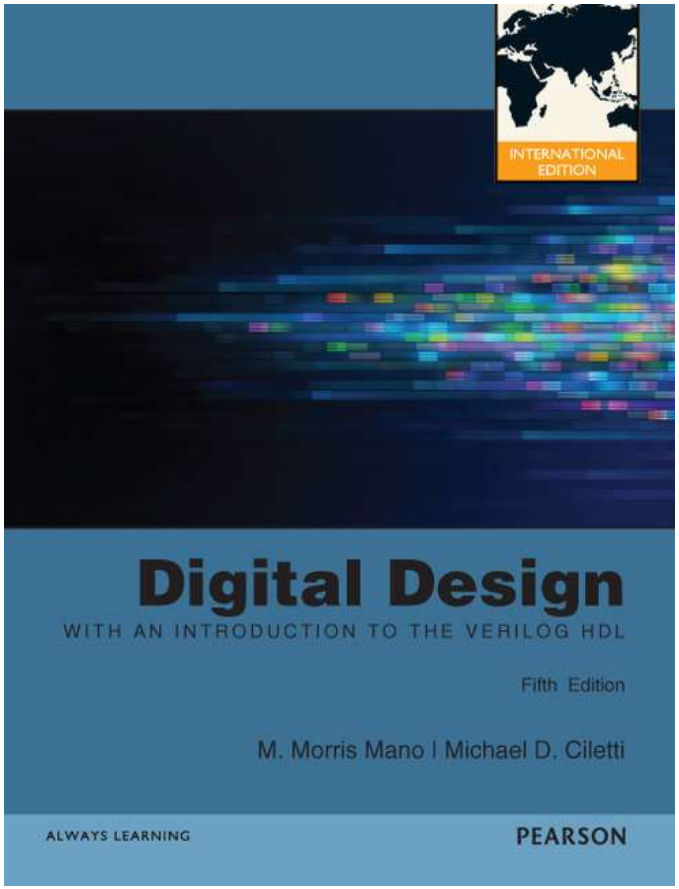
$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

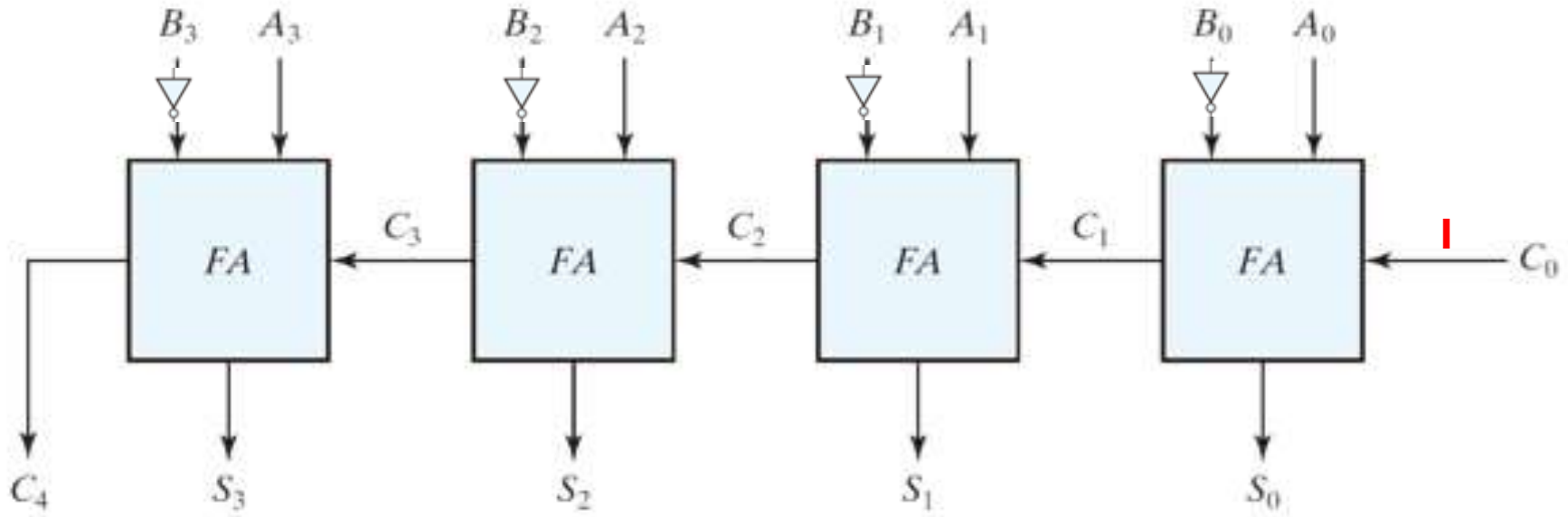
4.5.3 Binary Subtractor

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

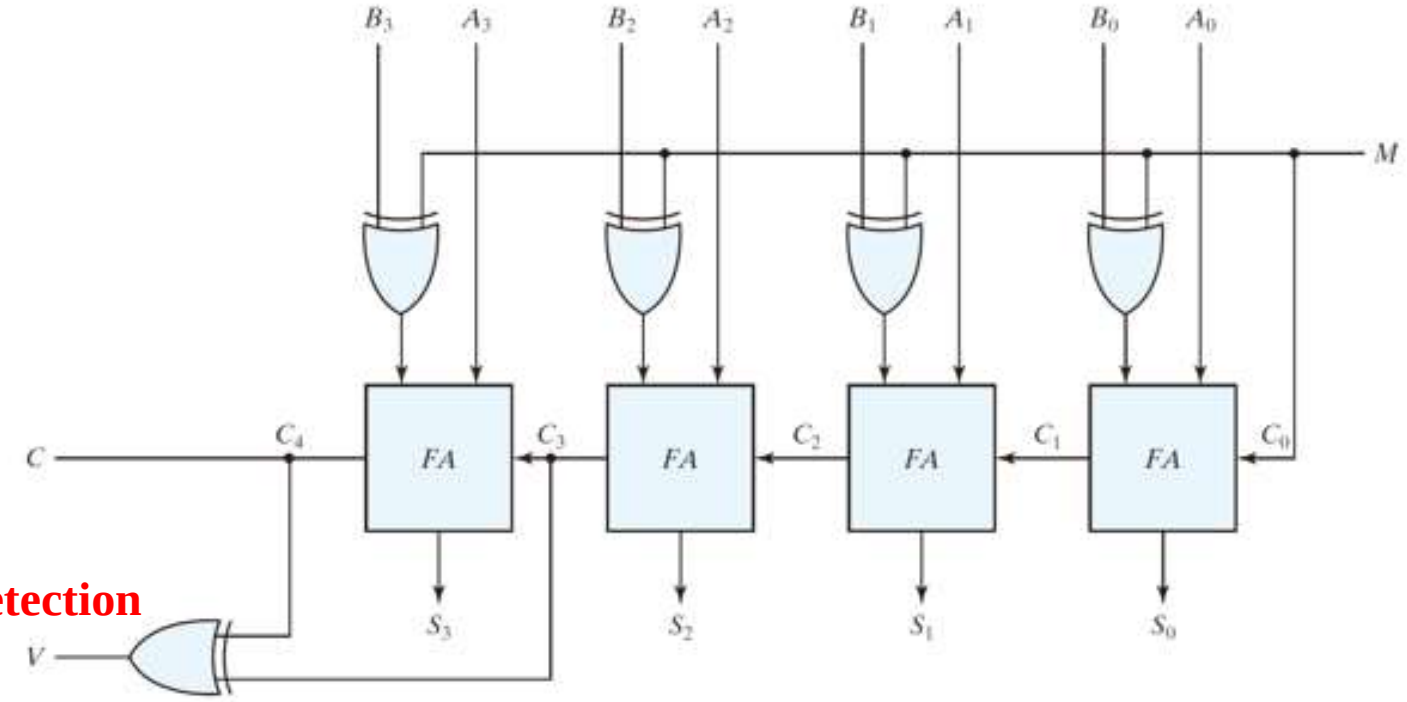
Binary Subtractor

- *A*-*B* = *A*+(2's complement of *B*)
  = A + B' + 1



Binary Subtractor

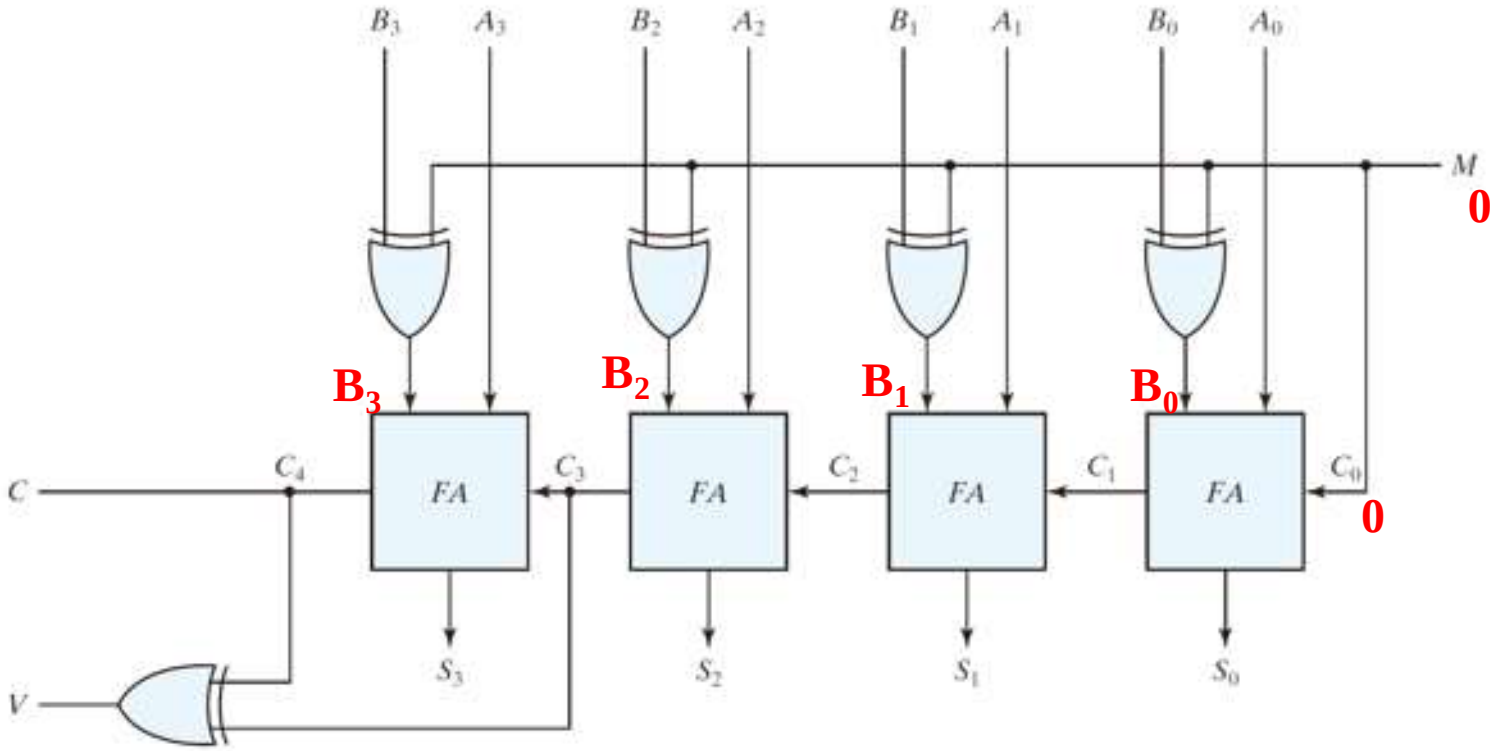
- *A*-*B* = *A*+(2's complement of *B*)
- 4-bit Adder Subtractor
  - *M=0, A+B;*
  - *M=1, A+B'+1*



Overflow Detection

Binary Adder

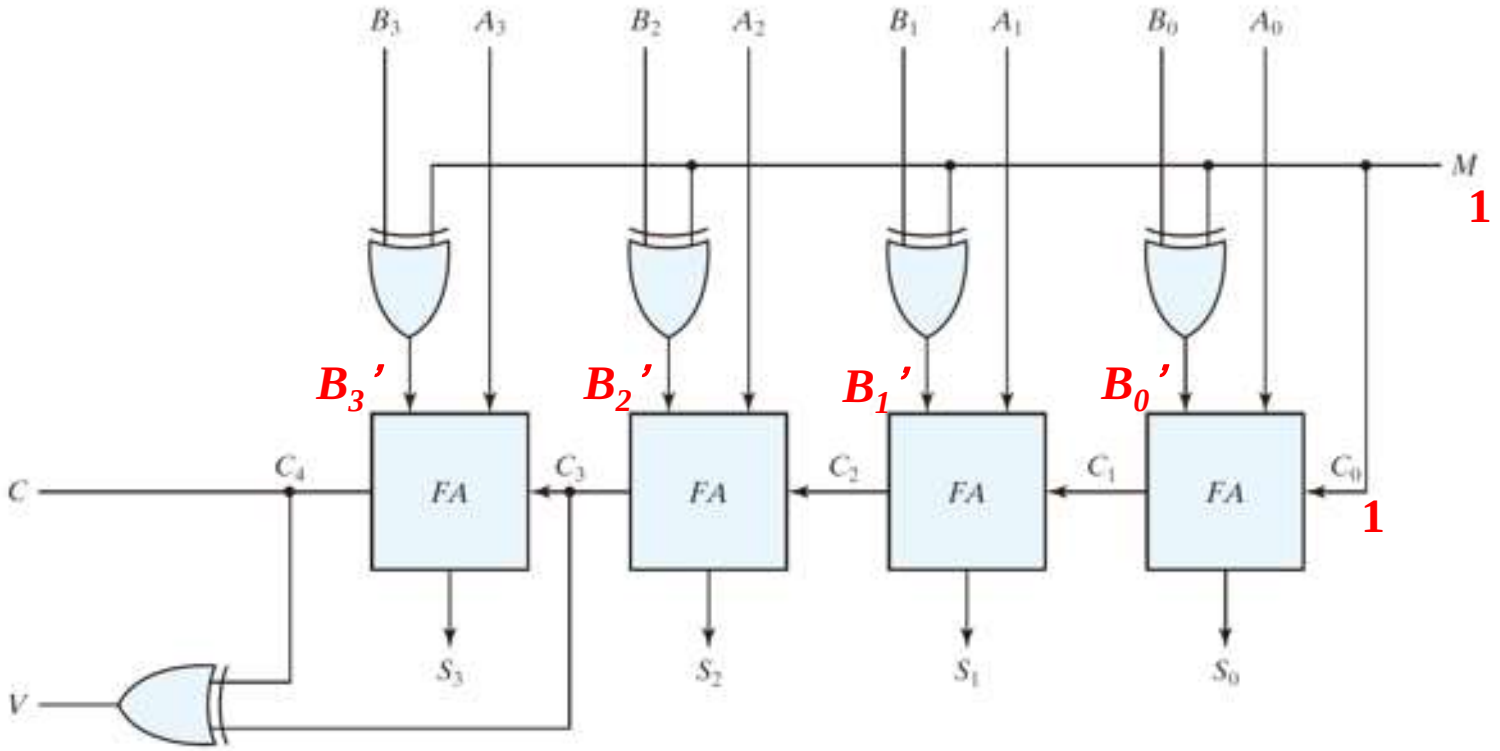
- M=0



Binary Subtractor

- M=1

A+B'+1



Overflow

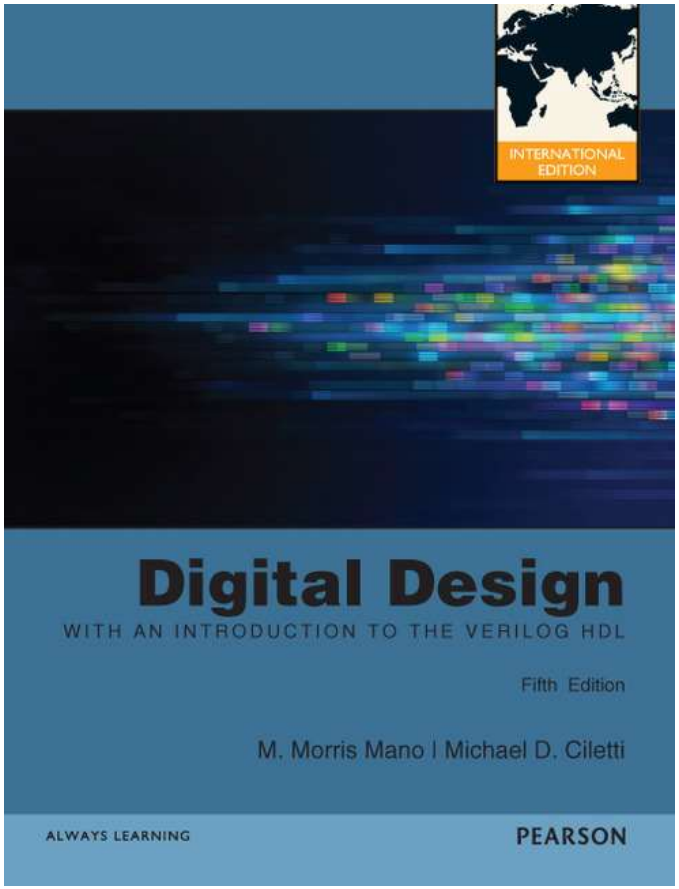
- The storage is limited
  - 4-bits : -8 ~7

6	0110	-6	1010	2	0010	-2	1110
5	0101	-5	1011	3	0011	-3	1101
11	01011	-11	10101	5	00101	-5	11011
C_3=1, C_4=0		**C_3=0, C_4=1**		**C_3=0, C_4=0**		**C_3=1, C_4=1**	

- Overflow Detection: $V=C_3 \oplus C_4$
  - V = 0 ➔ no overflow;
  - V = 1 ➔ overflow

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.6 Decimal Adder

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

BCD Adder

- Add two BCD's
  - 9 inputs: two BCD's and one carry-in
  - 5 outputs: one BCD and one carry-out
- Design approaches
  - A truth table with 2^9 entries → Bad idea!
  - use binary full Adders
    - the sum <= 9 + 9 + 1 = 19
    - binary to BCD

BCD Adder

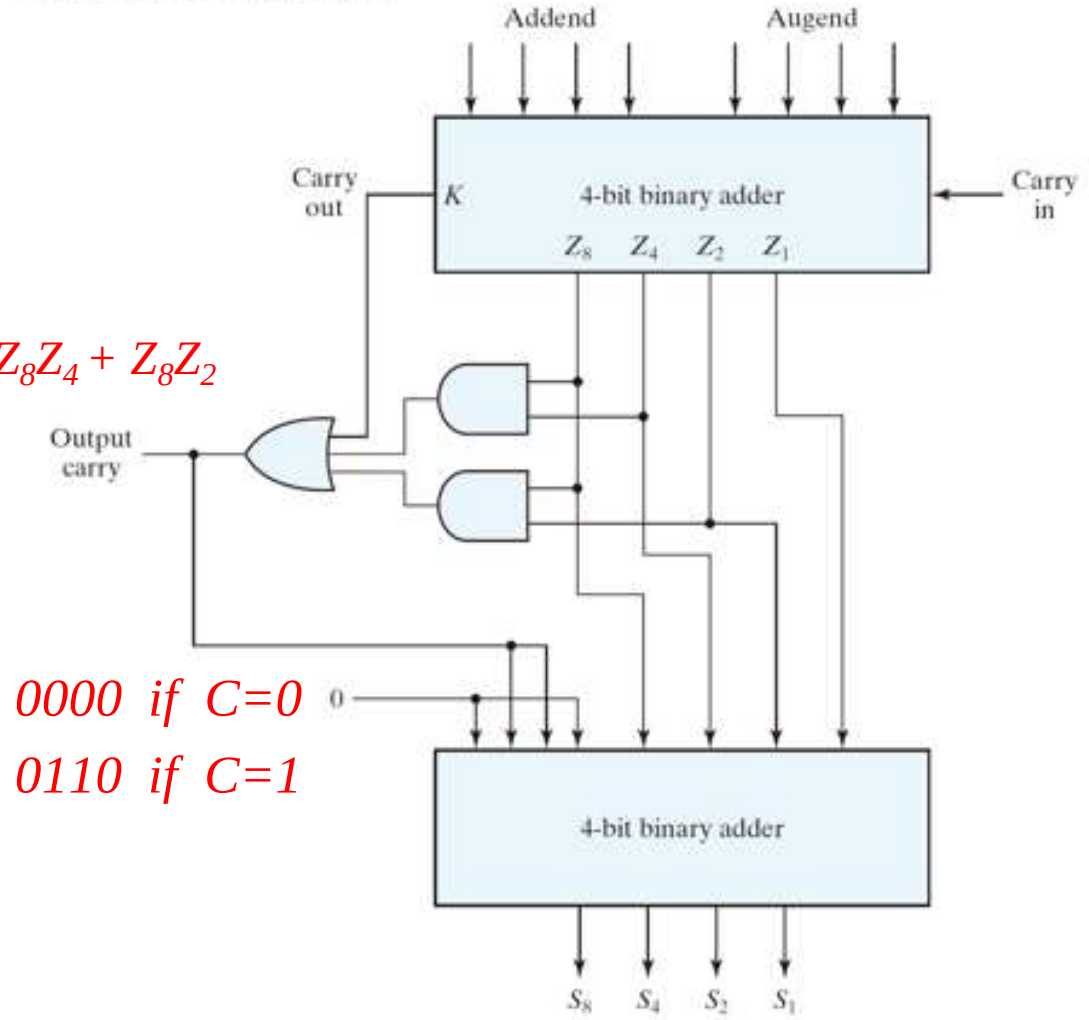
Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

$$C = K + Z_8Z_4 + Z_8Z_2$$

$$S = Z + 0000 \ \ if \ \ C=0$$

$$S = Z + 0110 \ \ if \ \ C=1$$

BCD Adder



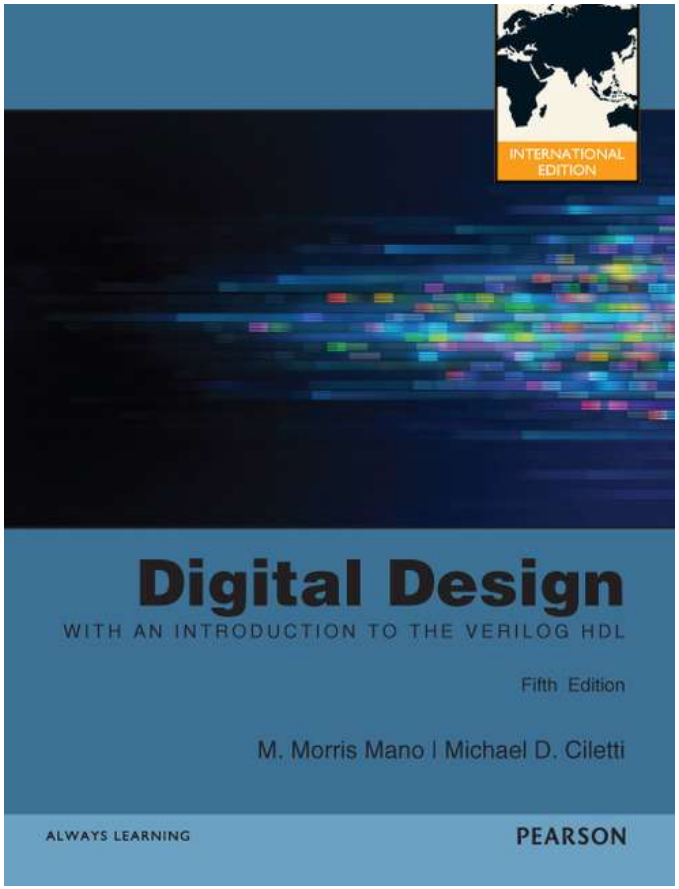
$C = K + Z_8Z_4 + Z_8Z_2$

$S = Z + 0000$ *if* $C=0$

$S = Z + 0110$ *if* $C=1$

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.7 Binary Multiplier

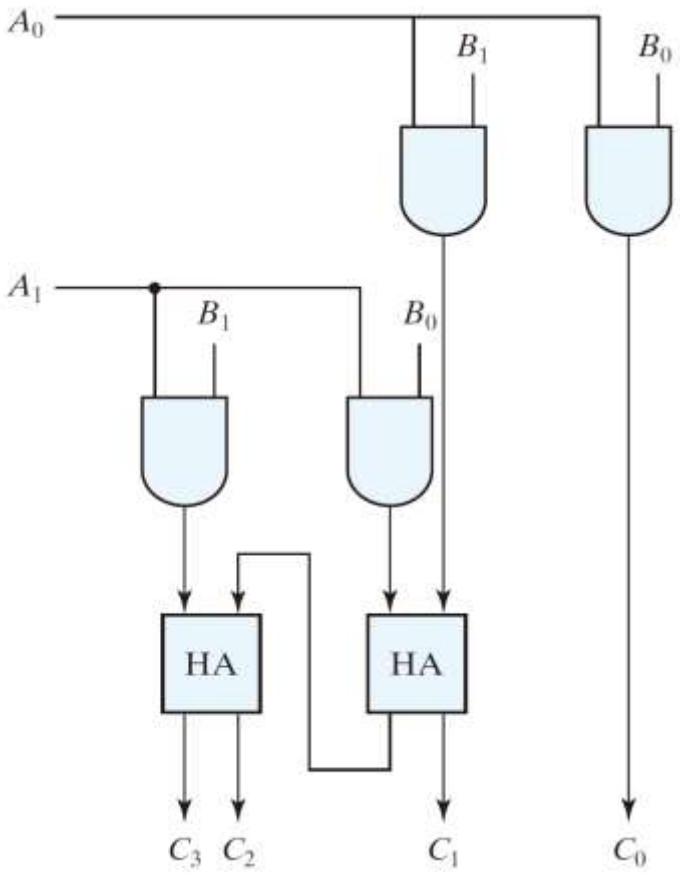
主講者：吳順德

國立臺灣師範大學機電工程系 副教授

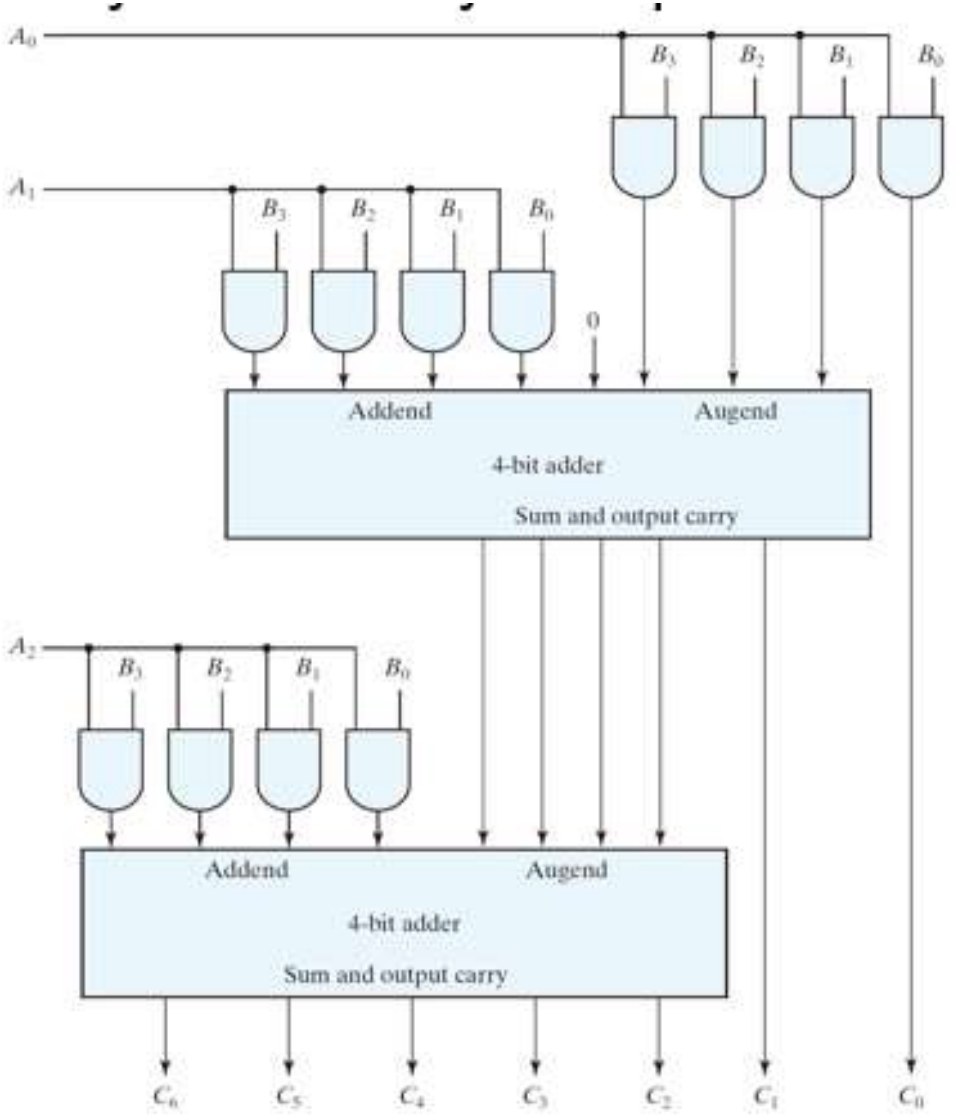
2 Bits Binary Multiplier

- Partial products
  - AND operations
- Sum of partial Products
  - Half Adder

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

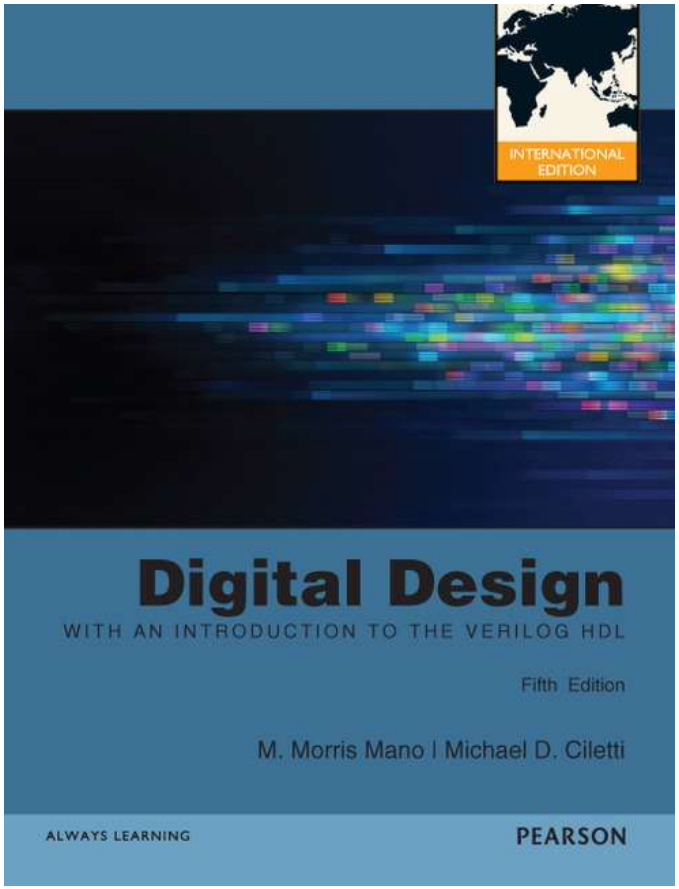


4-bit by 3-bit binary multiplier



Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.8 Magnitude Comparator

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

Design Concepts for Comparator

- The comparison of two numbers
  - outputs: A>B, A=B, A<B
- Design Approaches
  - truth table
    - 2^{2n} entries - too cumbersome for large *n*
  - use **inherent regularity of the problem**
    - reduce design efforts
    - reduce human errors

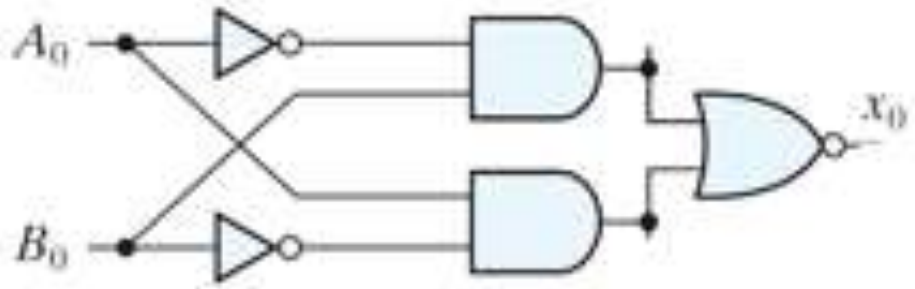
Algorithms

- Two binary number: ($A = A_3A_2A_1A_0$; $B = B_3B_2B_1B_0$)
- Equality for each bits: $x_i = A_iB_i + A_i'B_i'$, for i = 0, 1, 2, 3

- Algorithm:
  - $(A=B) = x_3x_2x_1x_0$ (**$A=B$ if $A_3=B_3$, $A_2=B_2$, $A_1=B_1$ and $A_0=B_0$**)
  - $(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$
  - $(A>B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$

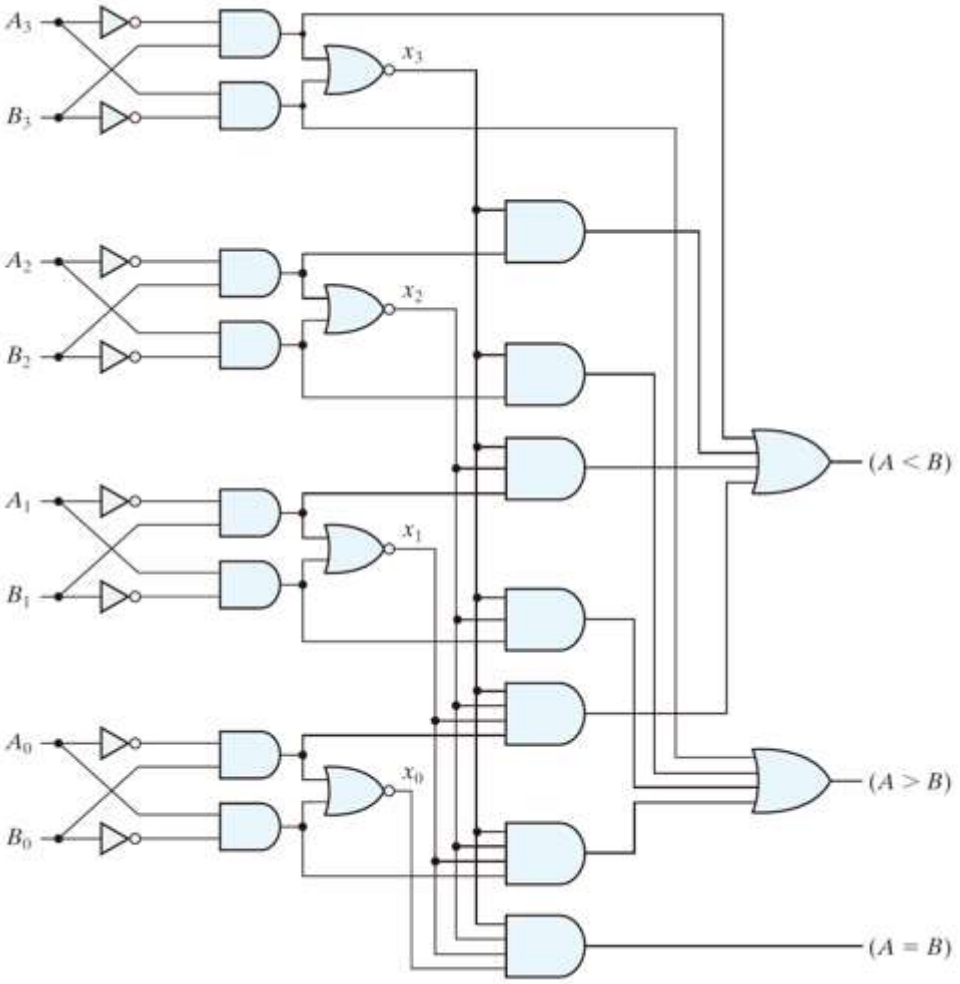
Implementation

- $x_i = A_iB_i + A_i'B_i' = (A_i'B_i + A_iB_i')'$



Exclusive-NOR

Four bit Magnitude Comparator



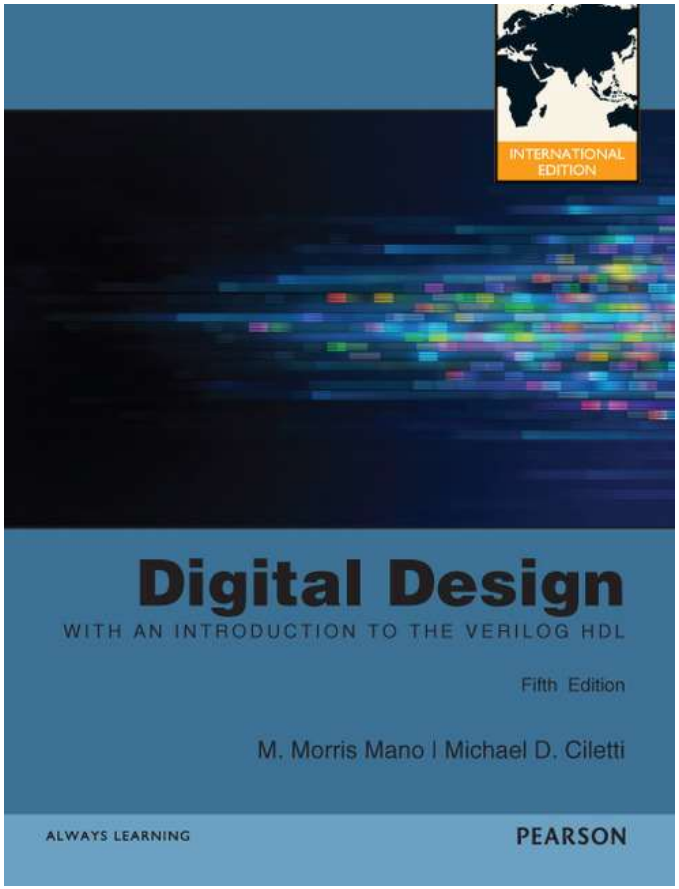
$A_3'B_3+x_3A_2'B_2+x_3x_2A_1'B_1+x_3x_2x_1A_0'B_0$

$A_3B_3'+x_3A_2B_2'+x_3x_2A_1B_1'+x_3x_2x_1A_0B_0'$

$x_3x_2x_1x_0$

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.9 Decoder

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

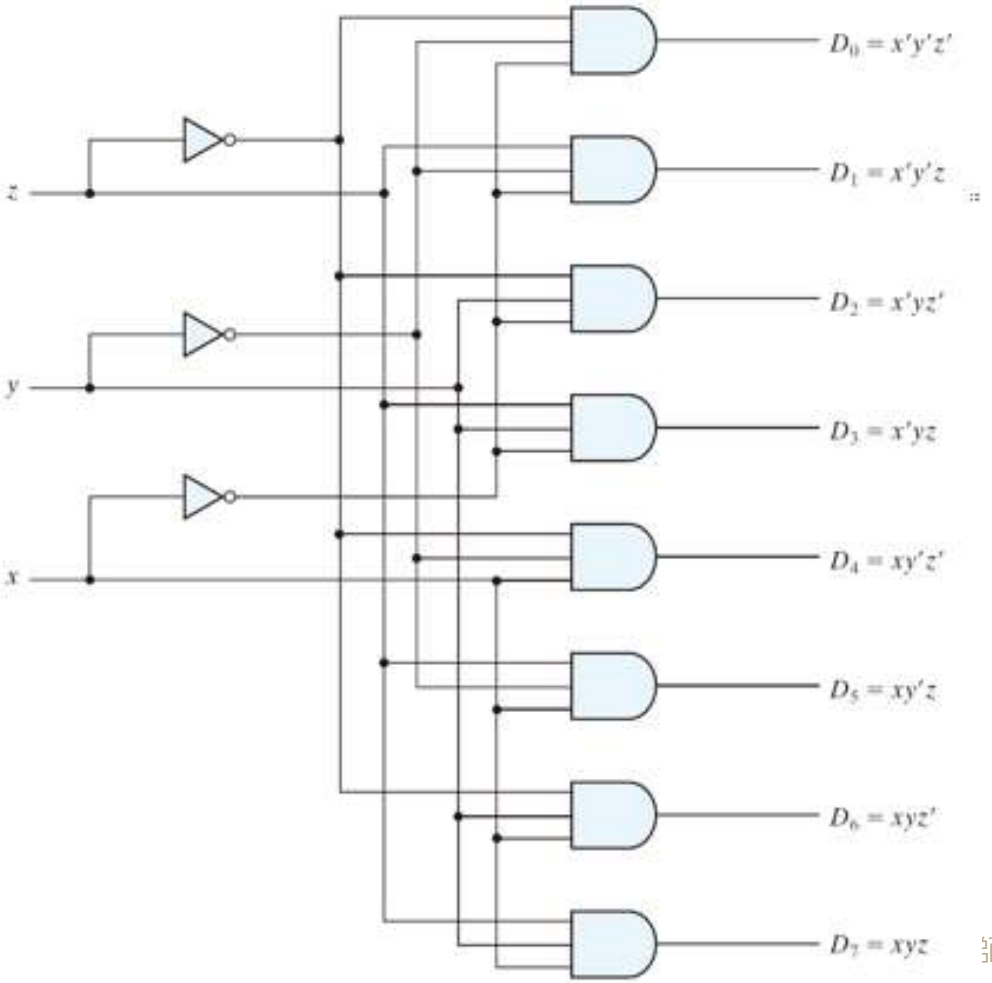
n-to-m decoder

- a binary code of n bits = 2^n distinct information
- A *decoder* is a combinational circuit that converts binary information from n input line to a m ($m \leq 2^n$) unique output lines.
- only one output can be active (D_i =1) at any time.
- Truth table of a 3 to 8 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

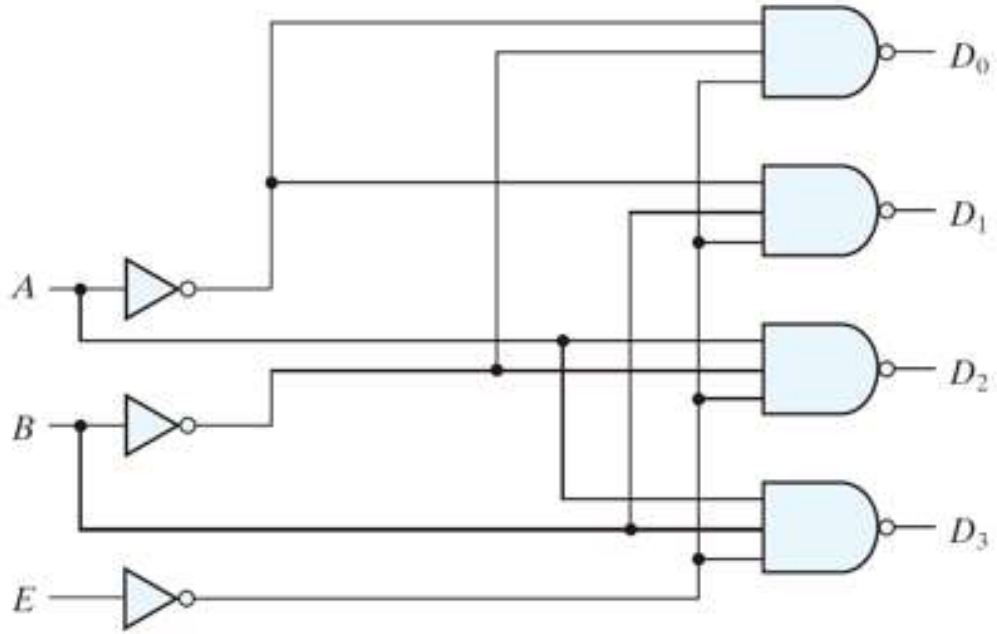
Implementation of 3 to 8 decoder

- Each one of the eight AND gates generates one of minterms.



2-to-4 line decoder with enable input

- It is more economical to generate the decoder minterms in their complement form by using NAND gates.
- Examples:
  - Enable when $E = 0$
  - Output is activated if $D_i = 0$



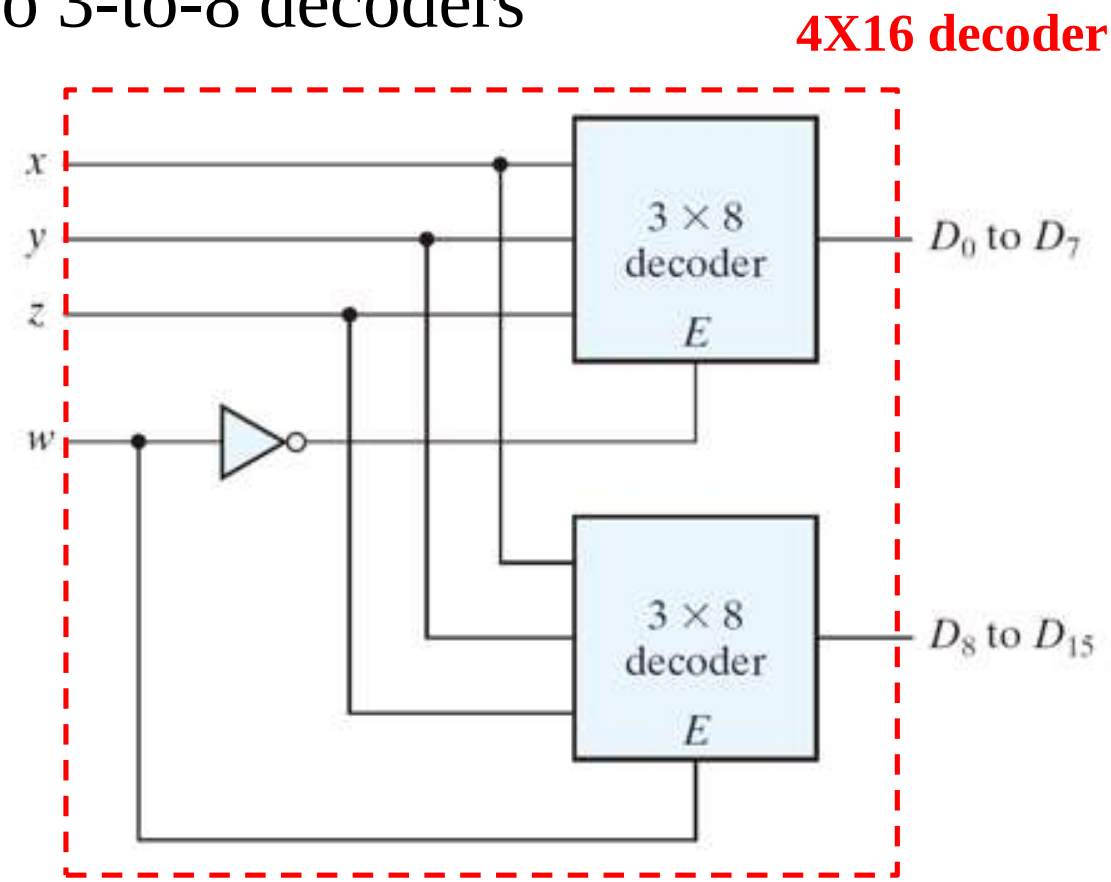
(a) Logic diagram

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	0

(b) Truth table

4-to-16 decoder

- A 4-to-16 decoder can be constructed by using two 3-to-8 decoders



Combination Logic Implementation

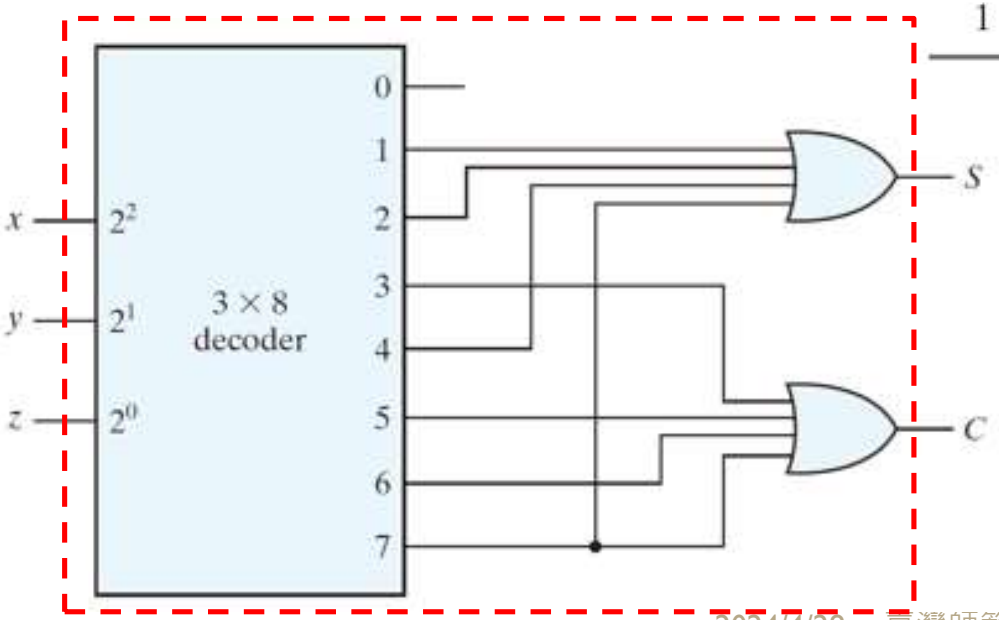
- each output = a minterm
- use a decoder and an external OR gate to implement any Boolean function of n input variables

Implementation of a Full Adder with a Decoder

- Boolean function of a full-adder
  - $S(x,y,x)=\Sigma(1,2,4,7)$
  - $C(x,y,z)=\Sigma(3,5,6,7)$

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

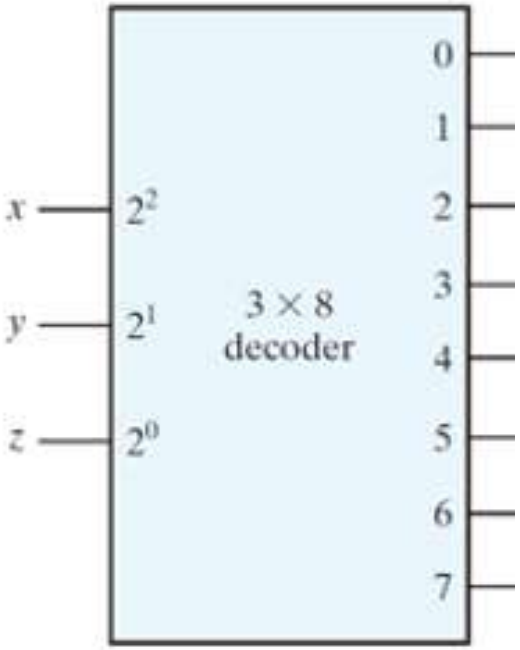
Full Adder

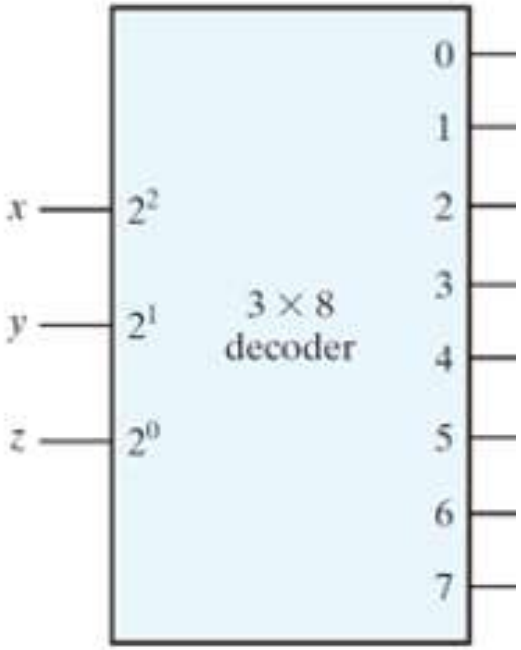


Implementation of Boolean Function

- two possible approaches using decoder
  - OR(minterms of F): k inputs
  - NOR(minterms of F'): $2^n - k$ inputs
- Example:

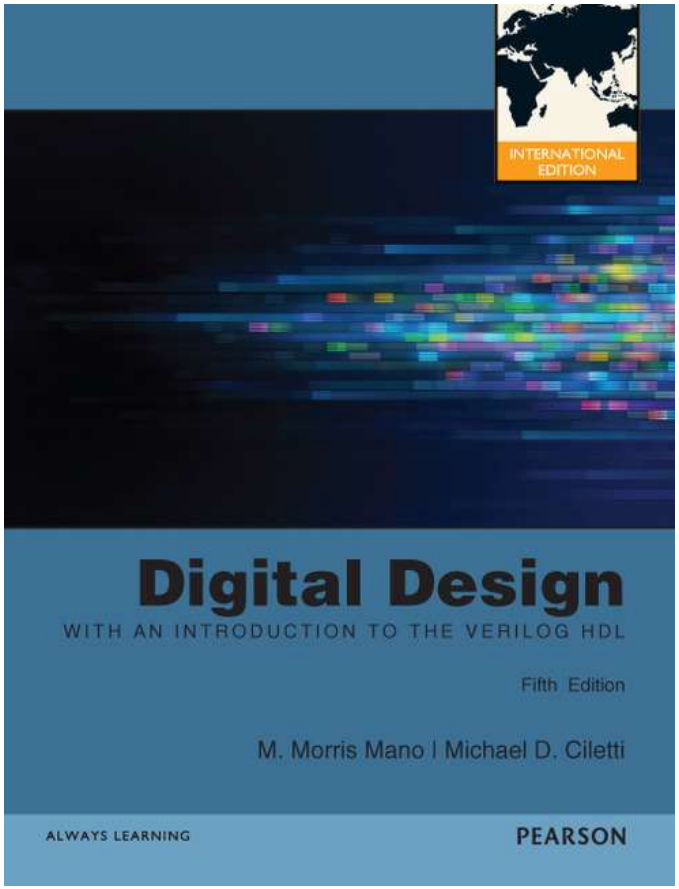
$F(x,y,x)=\Sigma(0,1,2,4,6,7)$ $F'(x,y,x)=\Sigma(3,5)$





Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



數位邏輯設計

4.10 Encoder

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

Encoder

- Encoder:
  - A digital circuit that perform the inverse operation of a decoder.
  - 2^n (or fewer) input lines and n output lines.
  - Only one input is activated ($D_i=1$)
- Example : 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

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Encoder

- Function of encoder fails
  - when multiple inputs are activated.
  - All inputs are 0.
- Example :
  - D_3=*1* & D_6=*1* ➔ $x = y = z = 1$;
  - Solution:
  - A higher priority for inputs with higher subscript numbers. (D_3=*1* & D_6=*1* ➔ D_6 is activated).

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Priority Encoder

- A ***priority encoder*** is an encoder circuit that includes the priority function.
- A priority encoder is used to resolve the ambiguity of illegal inputs (multiple inputs are 1 at the same time).
- Example:
  - V=0 if all inputs =0;
  - higher priority for inputs with higher subscript numbers

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

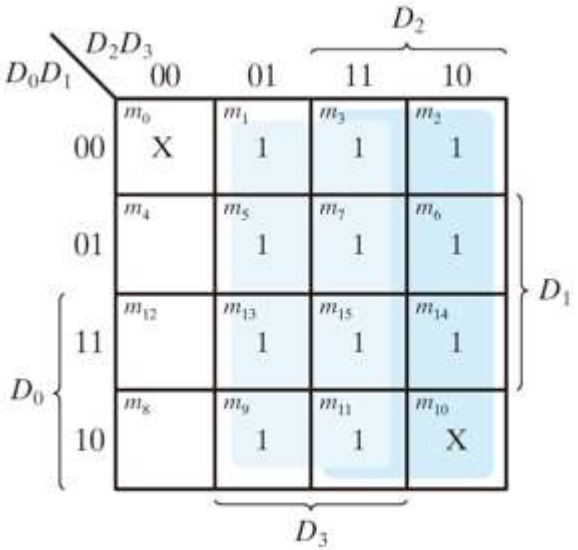
Maps for a priority encoder

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

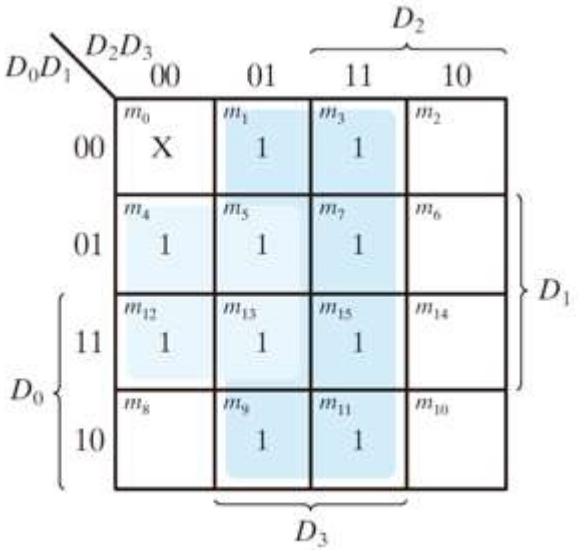
$$x = D_2 + D_3$$

$$y = D_3 + D_1 D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$



$x = D_2 + D_3$



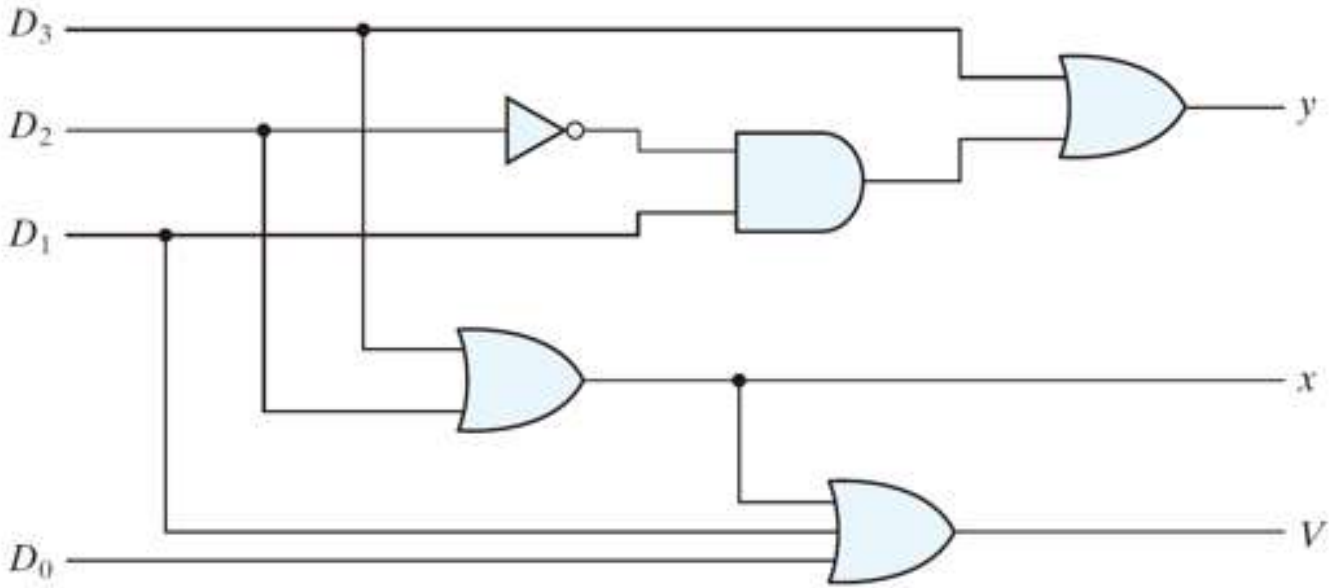
$y = D_3 + D_1 D'_2$

Implementation of a priority encoder

$$x = D_2 + D_3$$

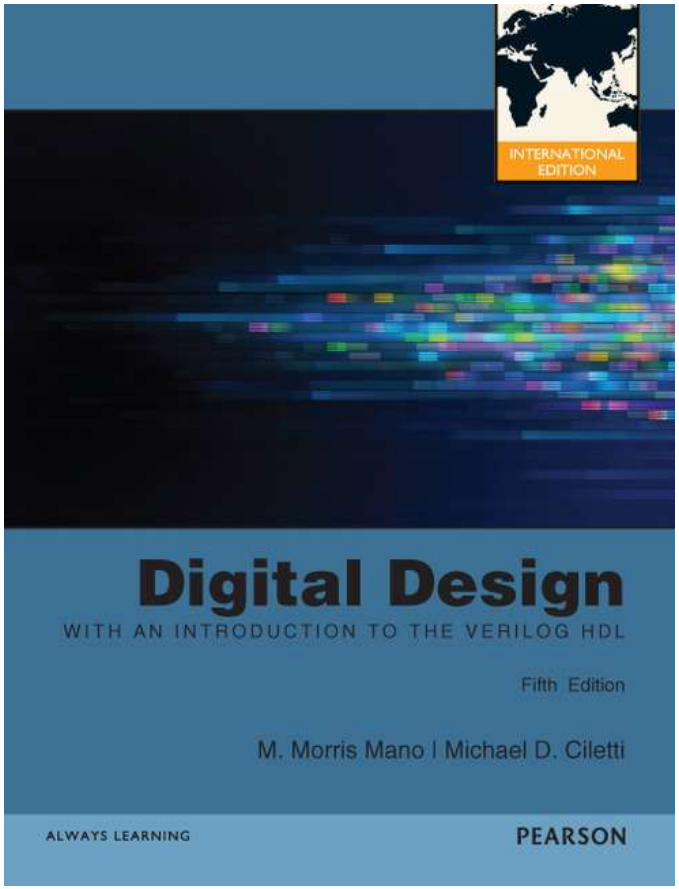
$$y = D_3 + D_1 D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$



Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



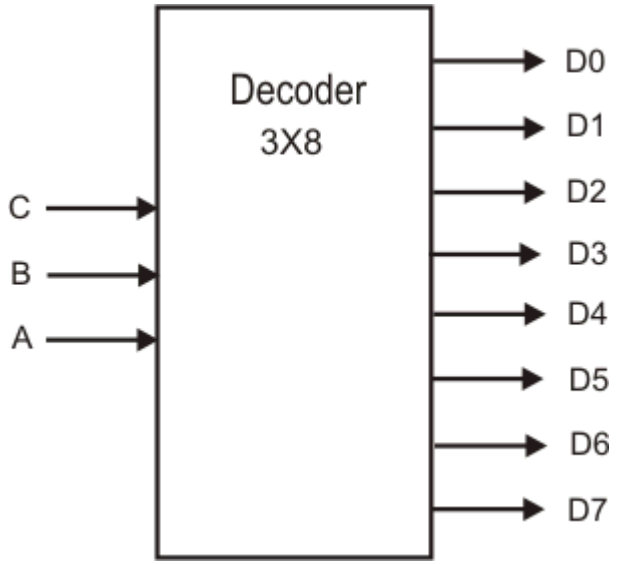
數位邏輯設計

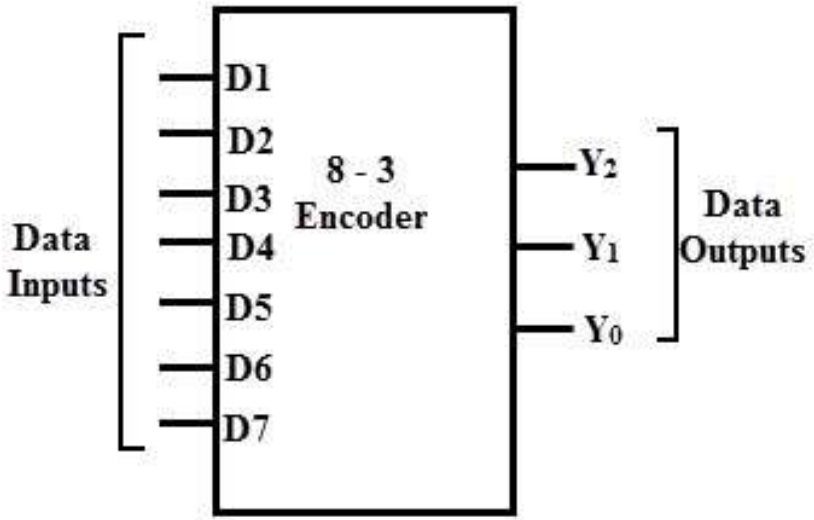
4.11.1 Multiplexer & Demultiplexer

主講者：吳順德

國立臺灣師範大學機電工程系 副教授

Recall : Decoder & Encoder



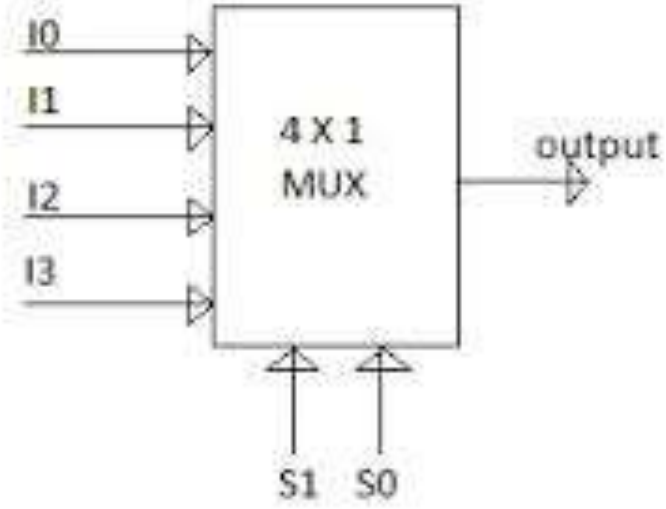


Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

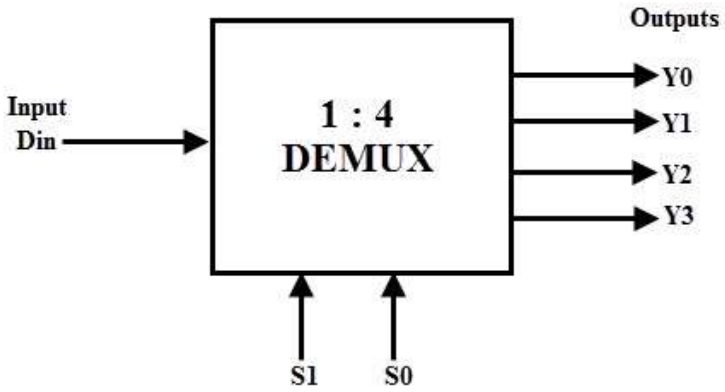
Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Multiplexer and Demultiplexer

- ***Multiplexer***: a circuit that selects binary information from one of many input lines and direct it to a single line.
- ***Demultiplexer***: a circuit that receive information from a single line and transmits it on one of many output lines.



Select lines

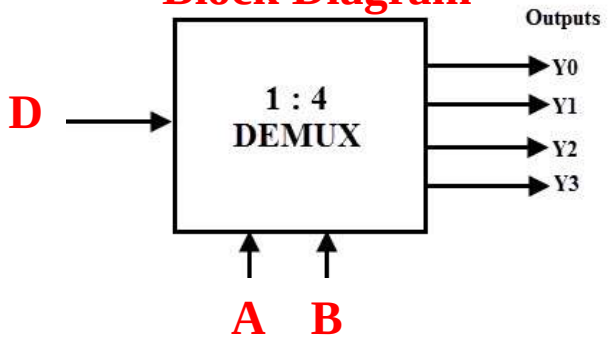


Select lines

Demultiplexers

- Example : 1 to 4 line demultiplexer
  - Data input : *D*
  - Selection input *A*, *B*
  - Output: Y_i

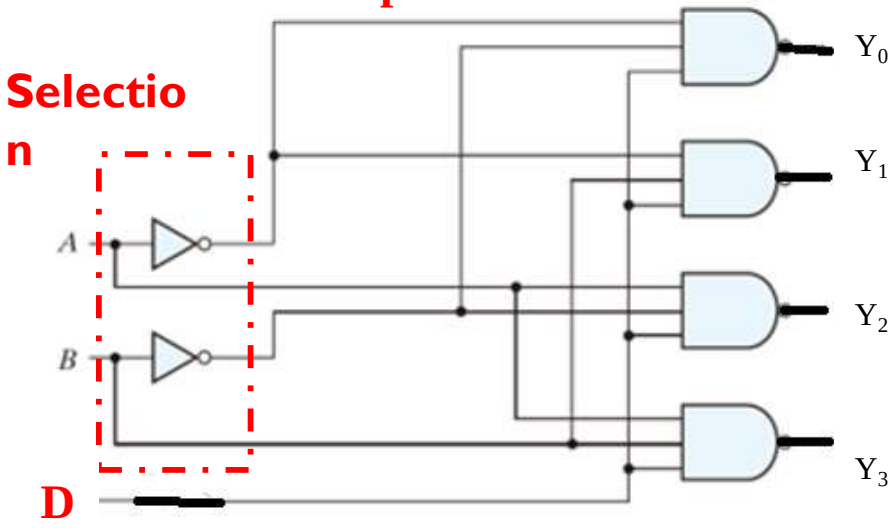
Block Diagram



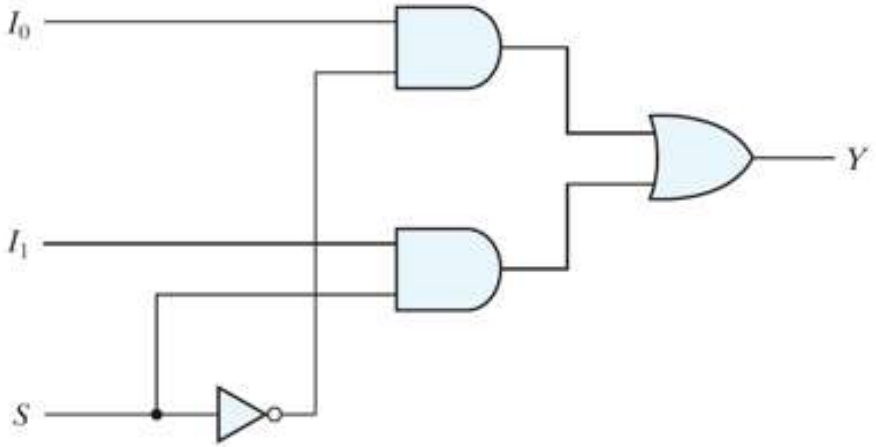
Truth Table

A	B		Y_0	Y_1	Y_2	Y_3
0	0		**D**	0	0	0
0	1		0	**D**	0	0
1	0		0	0	**D**	0
1	1		0	0	0	**D**

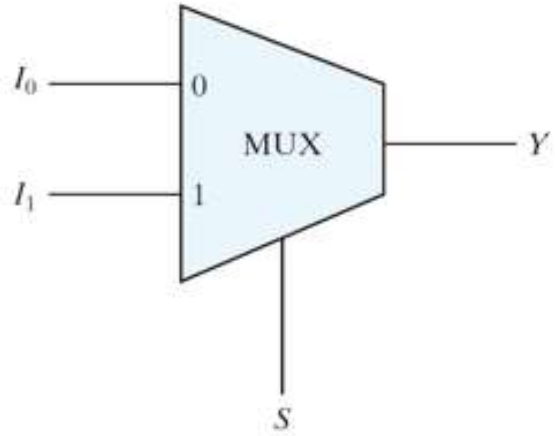
Implementation



2-to-1-line multiplexer



(a) Logic diagram



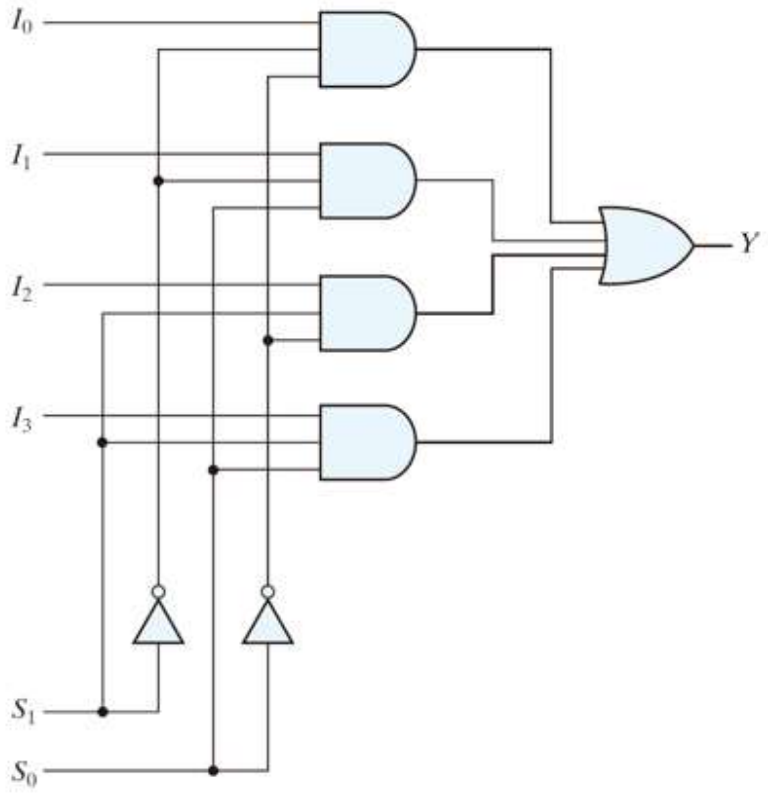
(b) Block diagram

4-to-1-line multiplexer

- Truth table

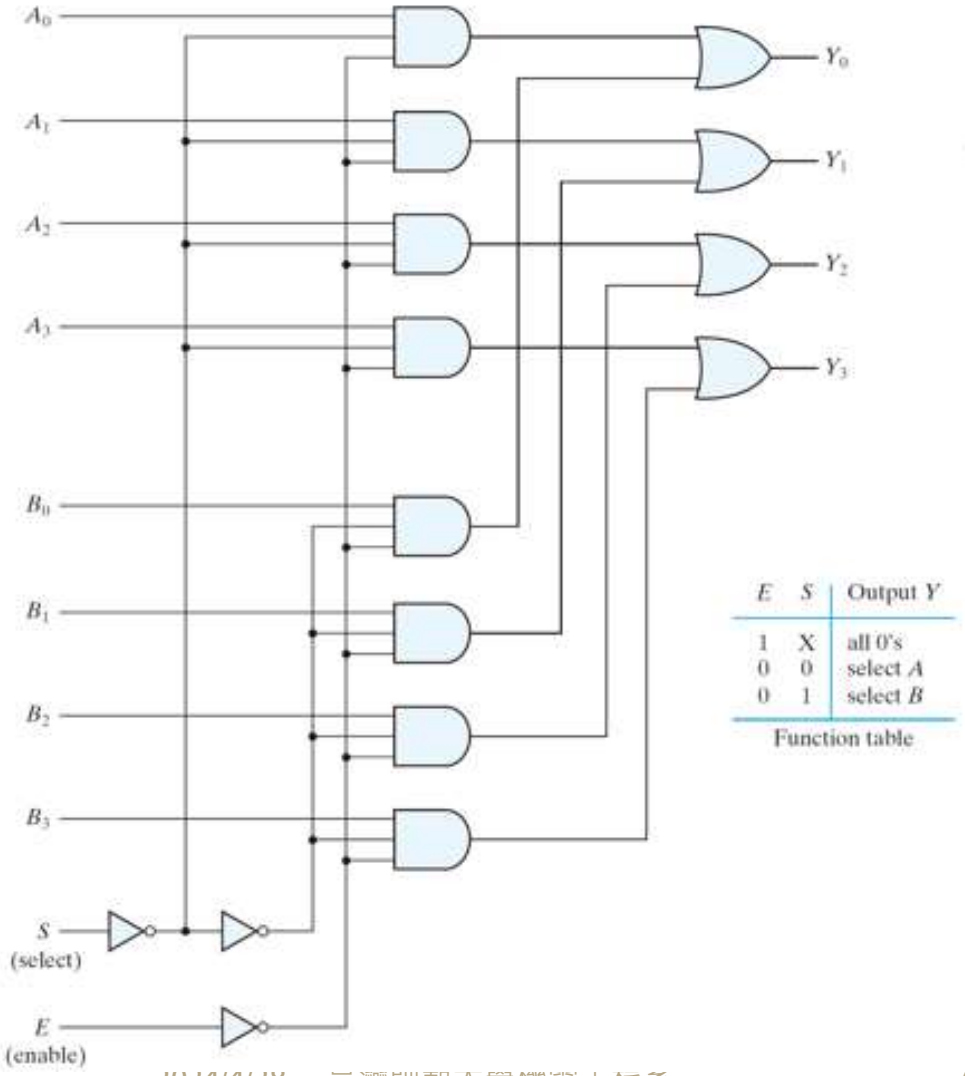
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

- Implementation



Multiple bits selection by using Multiplexer

- Data inputs:
  - A_i, B_i
- Selection input:
  - S
- Enable input:
  - E
- Outputs
  - Y_i

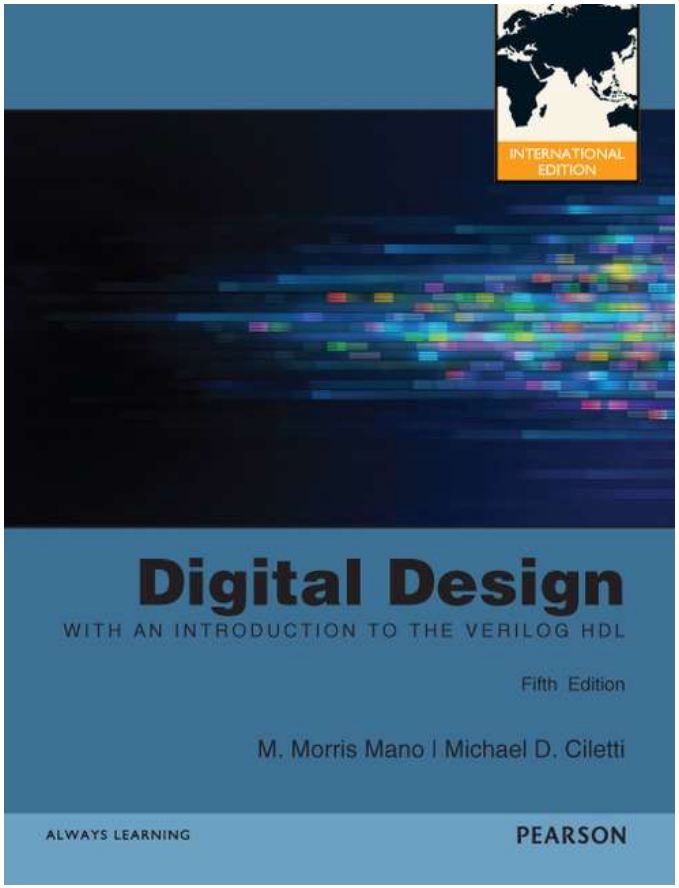


E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Function table

Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.



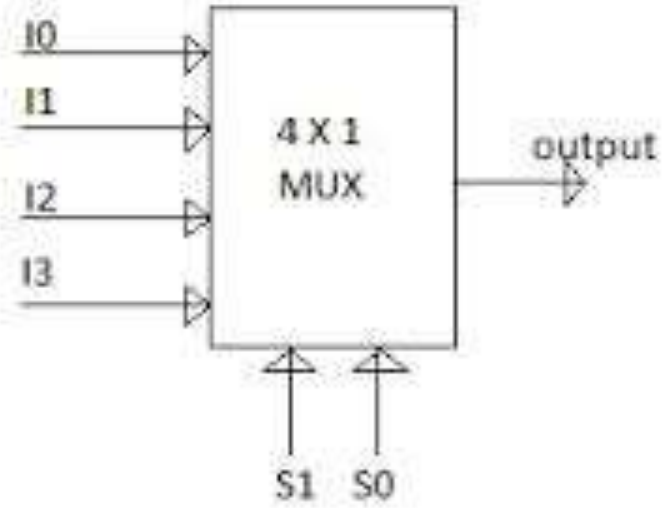
數位邏輯設計

4.11.2 Multiplexer

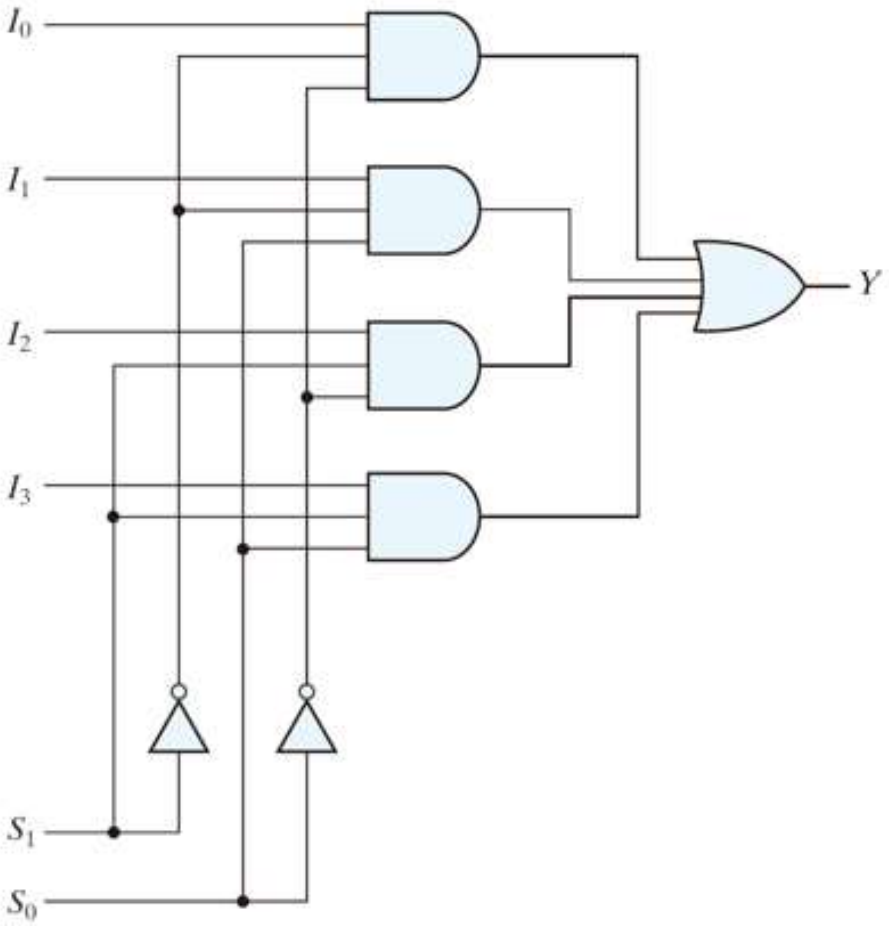
主講者：吳順德

國立臺灣師範大學機電工程系 副教授

Multiplexer

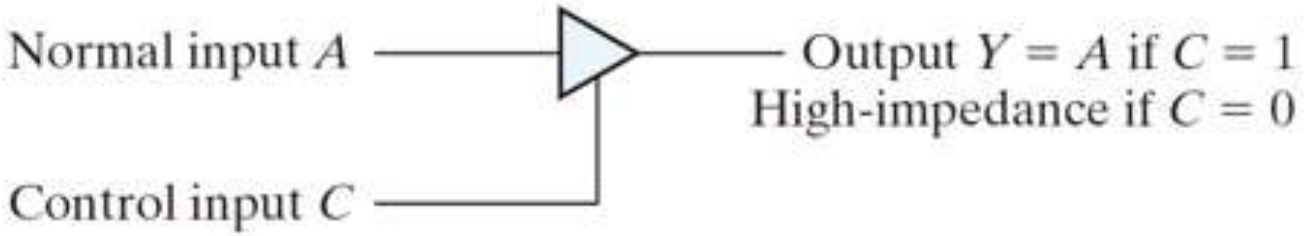


Select lines

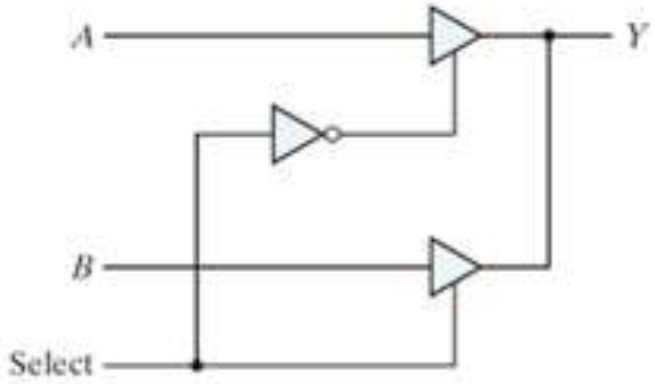


Three-State (Tri-State) Gates

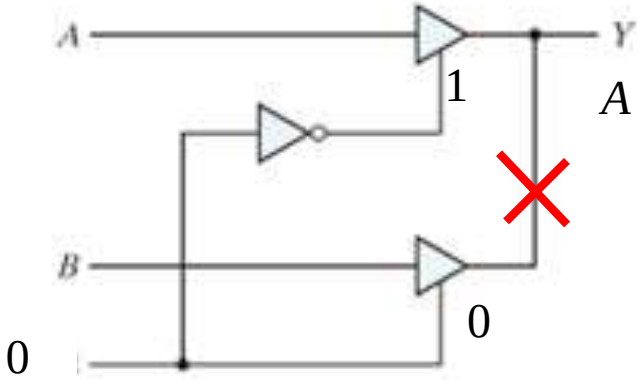
- Three-state gates:
  - Output state: 0, 1, and high-impedance.
- High impedance:
  - **The logic behaves like open circuit (output is disconnected).**
  - The circuit has no logic significant.
- Graph Symbol



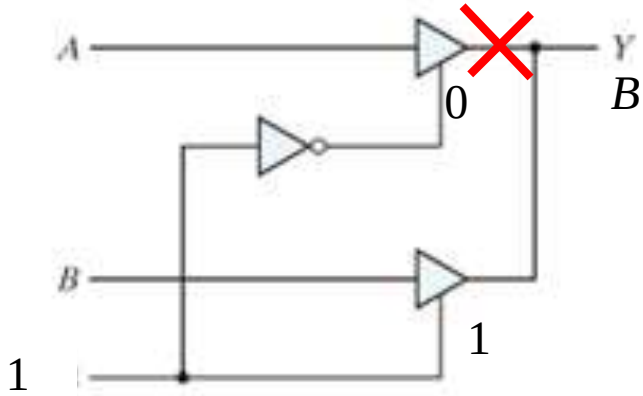
Construct Multiplexer by Three State Gates



(a) 2-to-1-line mux



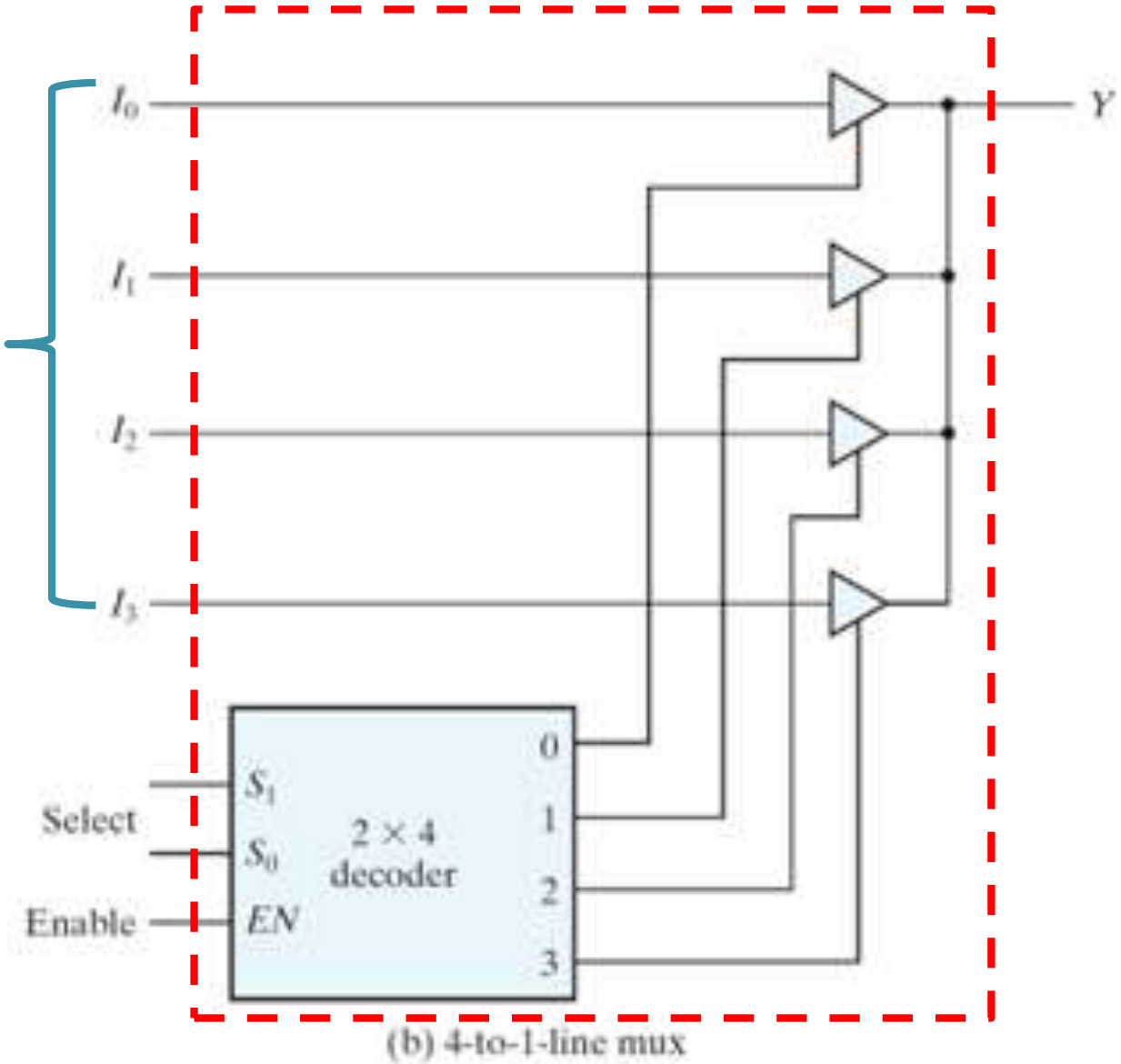
(a) 2-to-1-line mux



(a) 2-to-1-line mux

4-to-1 line multiplexer

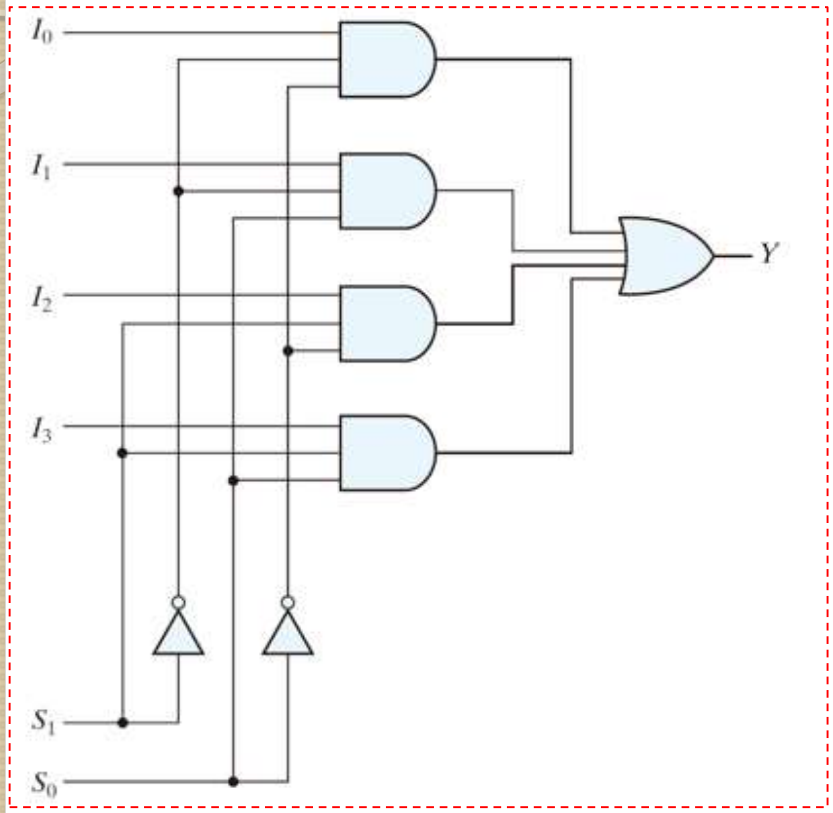
Data inputs



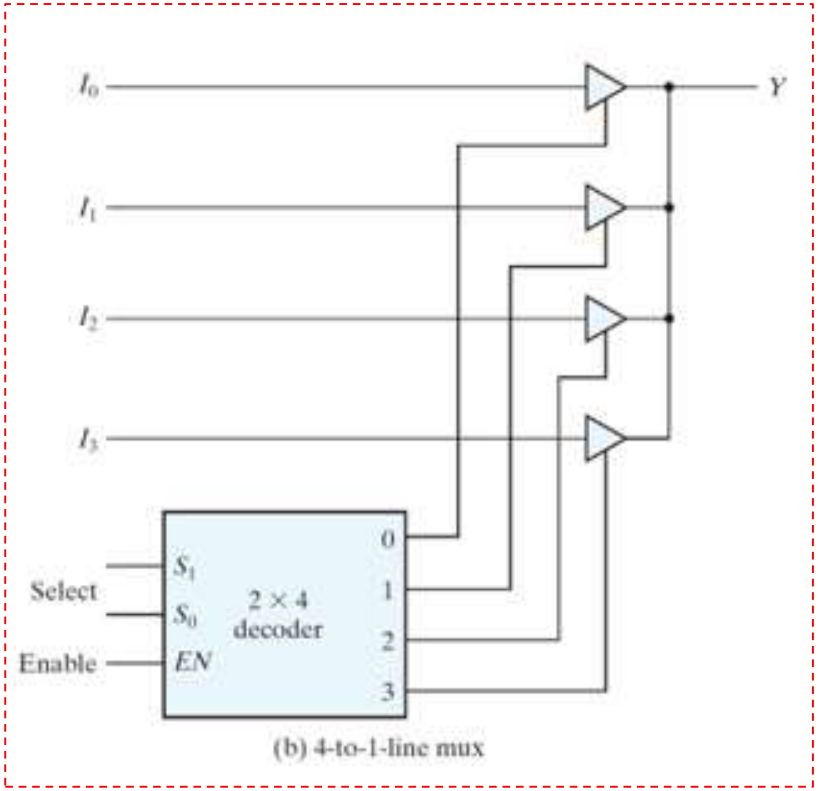
(b) 4-to-1-line mux

Ways for Implementing Multiplexer

AND – OR Gates



Three state gates + decoder



E=0 **➔ all of the three-state buffer are inactive ➔ Y high impedance**
E=1 **➔ only one of the three-state buffers will active.**

Boolean Function Implementation by using Multiplexer

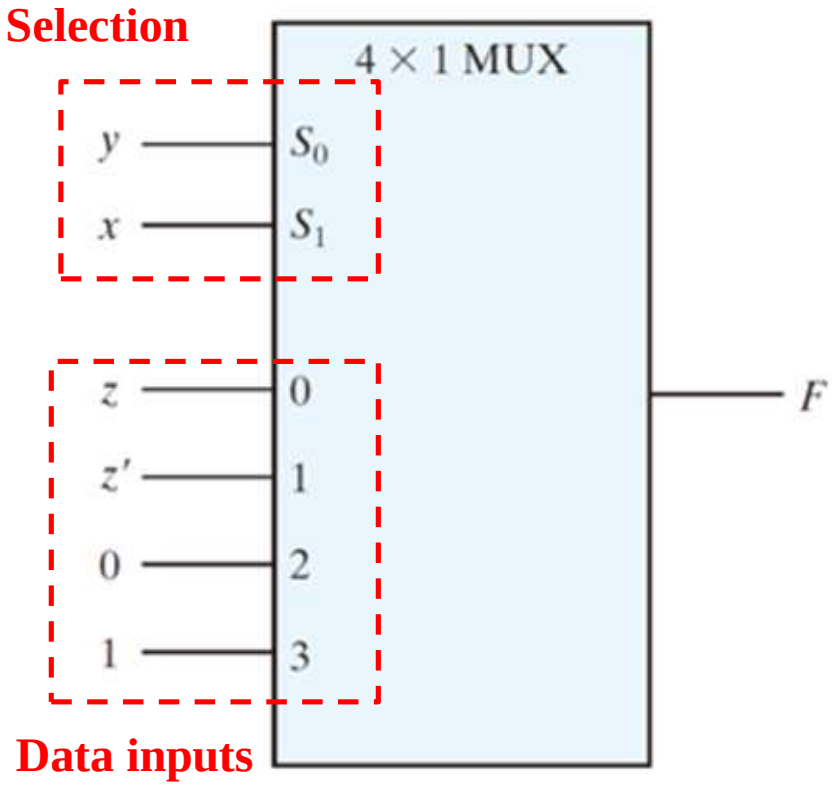
- *n* variable Boolean function implemented by using multiplexer:
  1. First *n*-1 variables are connected to the selection inputs of the multiplexer.
  2. The remaining single variable denoted by ***z*** is used for the data input.
  3. The data input of the multiplexer will be ***z***, ***z'***, *1* or *0*

Example:

$F(x,y,z) = \Sigma(1,2,6,7)$

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

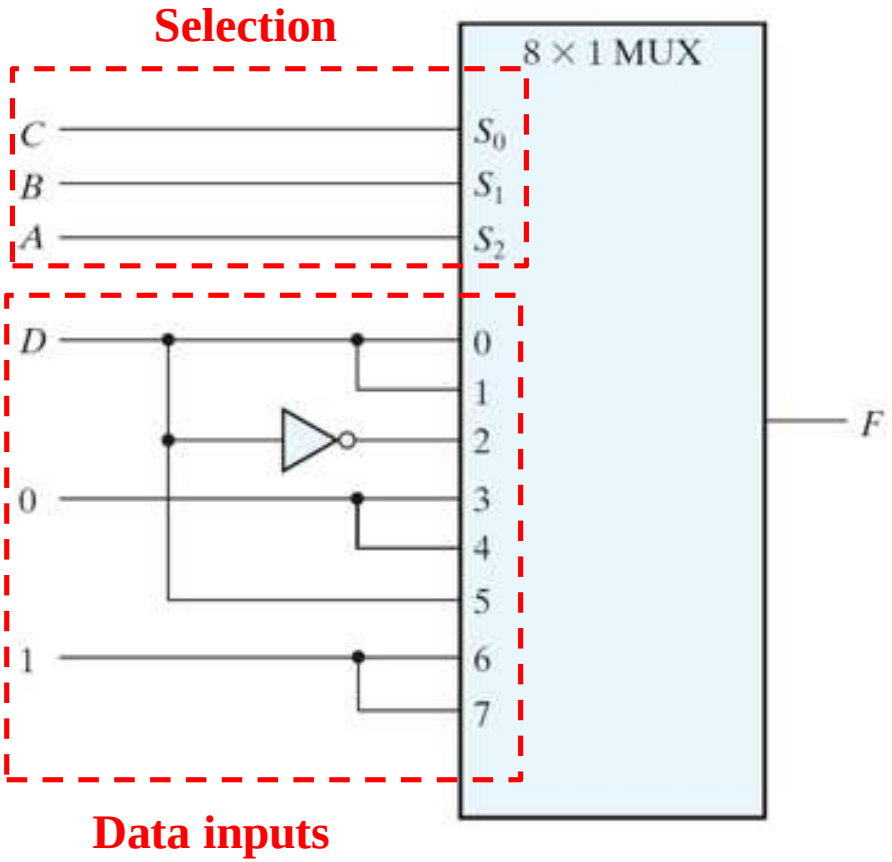


(b) Multiplexer implementation

Example:

$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



Reference

- M. M. Mano and M. D. Ciletti, “Digital Design," 5th Ed., Pearson Education Limited, 2013.