計算機結構
(控制單元的設計)

陳鍾誠 於金門大學

第 8 章
Combinational Logic Applications

Computer Organization and Design Fundamental

書籍作者： David Tarnoff
投影片製作者：陳鍾誠

2 對 4 解碼器

Active High 版

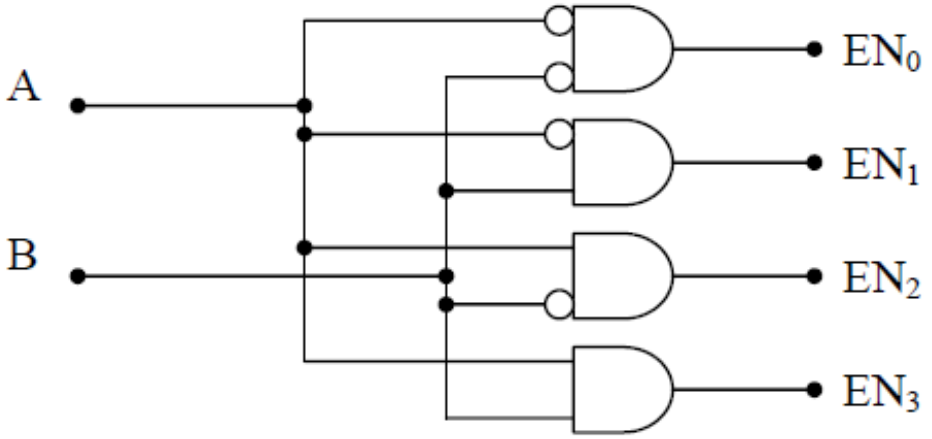


Figure 8-22 Digital Circuit for a 1-of-4 Decoder

Active Low 版

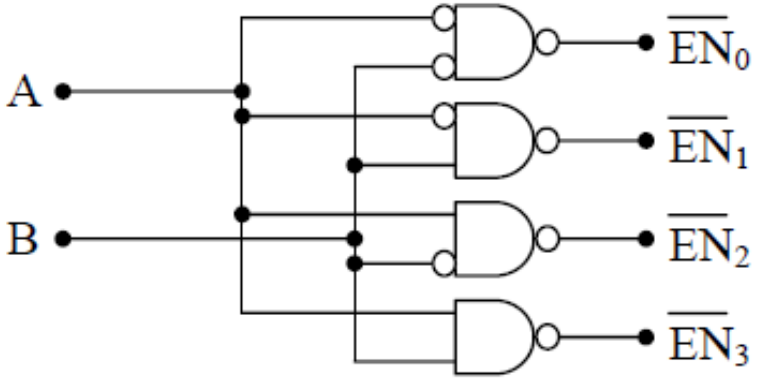
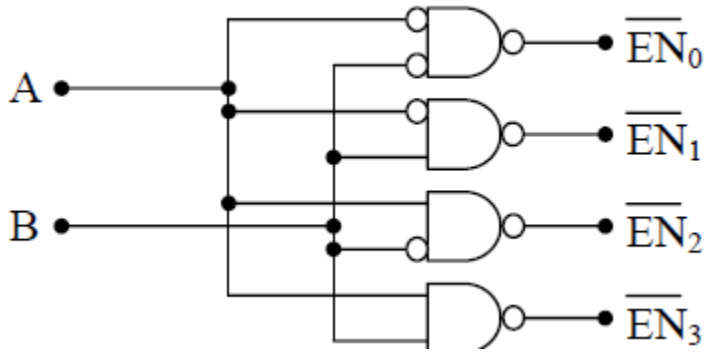


Figure 8-23 Digital Circuit for an Active-Low 1-of-4 Decoder

2 對 4 解碼器



A	B	C	EN_0	EN_1	EN_2	EN_3	EN_4	EN_5	EN_6	EN_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Figure 8-24 Truth Table for an Active-Low 1-of-8 Decoder

8.5 Multiplexers (多工器)

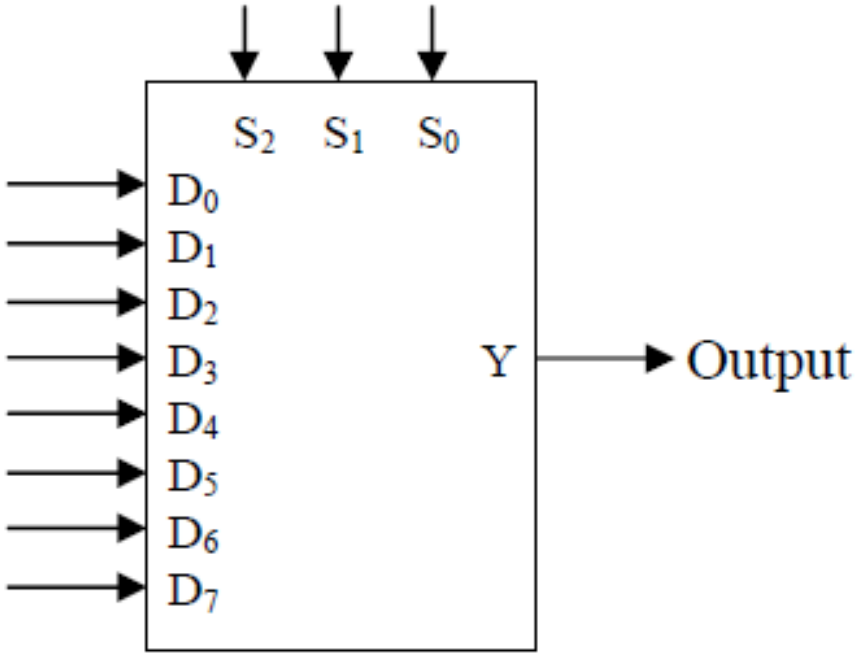
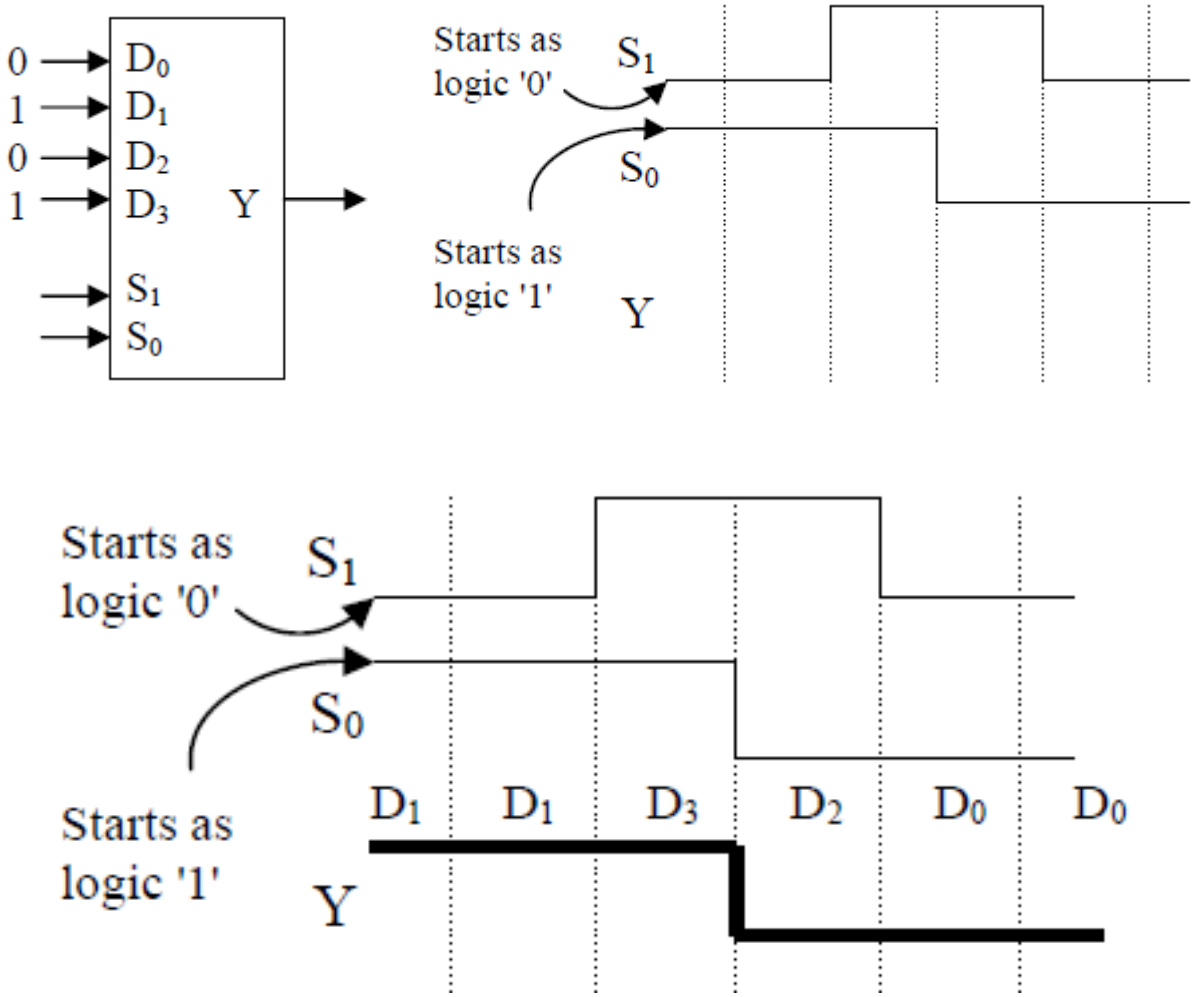


Figure 8-25 Block Diagram of an Eight Channel Multiplexer

多工器 -- 計算輸出


0
1
0
1
D0
D1
D2
D3
Y
S1
S0
Starts as logic '0'
Starts as logic '1'
S1
S0
Y
Starts as logic '0'
Starts as logic '1'
S1
S0
D1
D1
D3
D2
D0
D0
Y

8.6 Demultiplexers (解多工器)

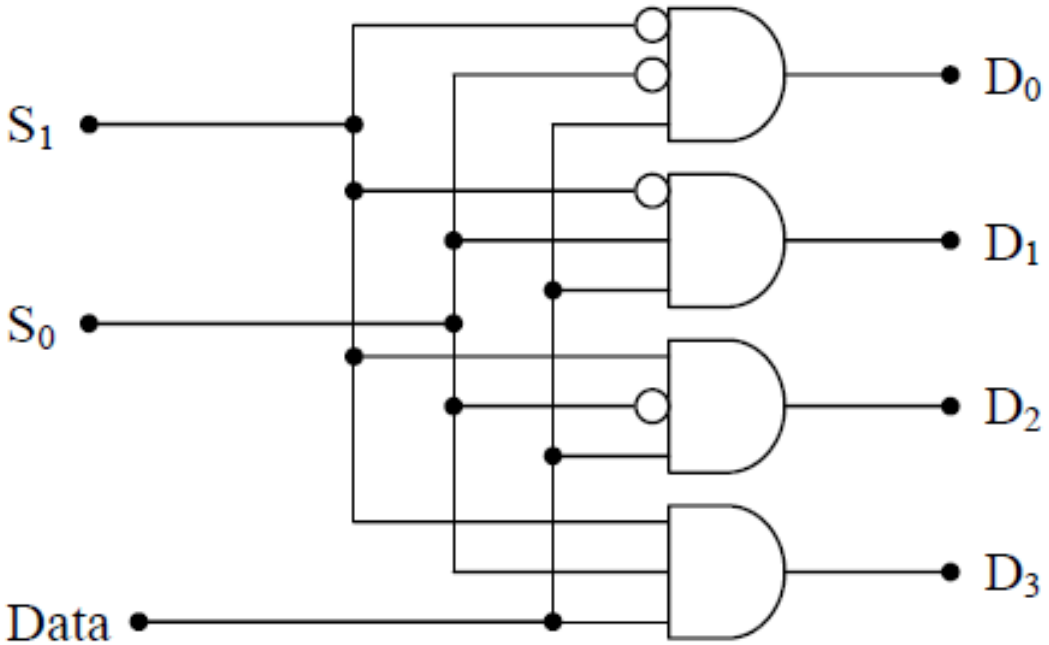
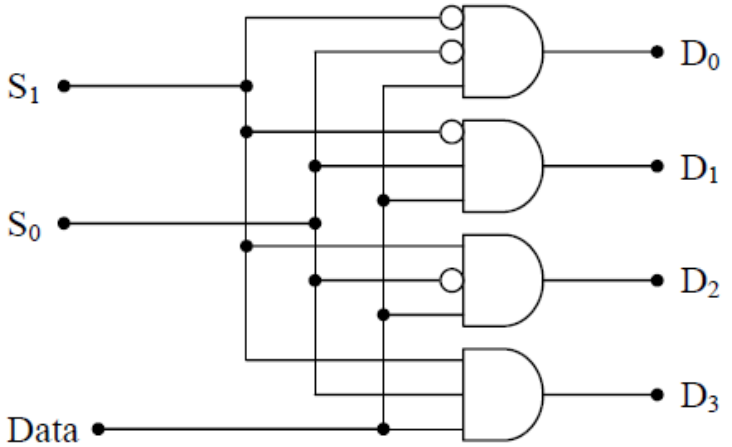


Figure 8-27 Logic Circuit for a 1-Line-to-4-Line Demultiplexer

解多工器 -- 真值表



S_1	S_0	Data	D_0	D_1	D_2	D_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

Figure 8-28 Truth Table for a 1-Line-to-4-Line Demultiplexer

解碼、編碼與多工器

陳鍾誠 於金門大學

授權限制

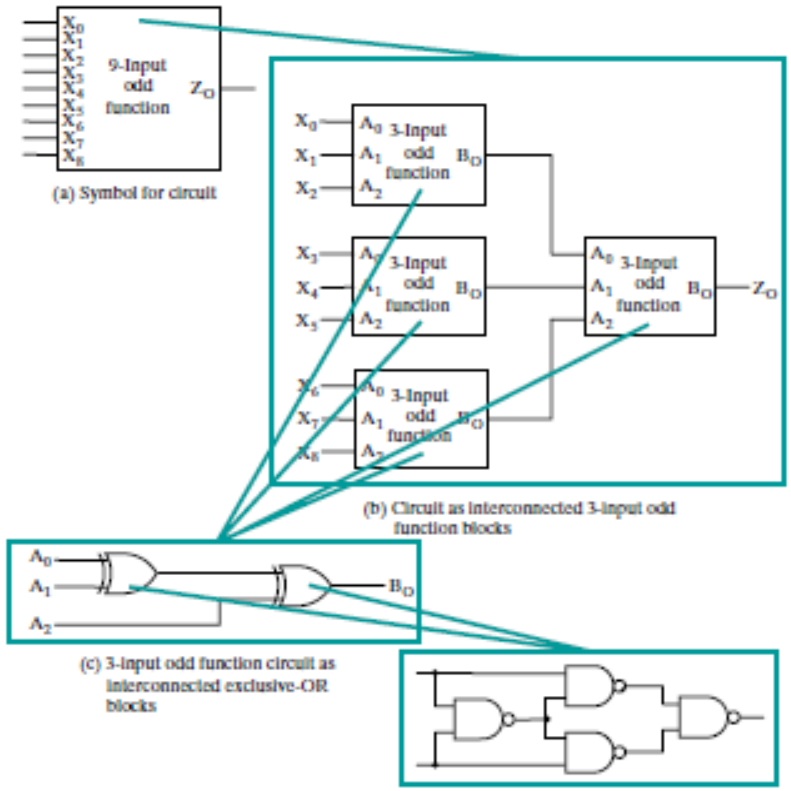
- 本投影片 部份取材自 邏輯與計算機設計 之投影片
  - 來源： http://writphotec.com/mano4/
  - 使用時請遵照下列條款

Terms of Use

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.

同位檢查 (Parity Check) 電路

Hierarchy for Parity Tree Example



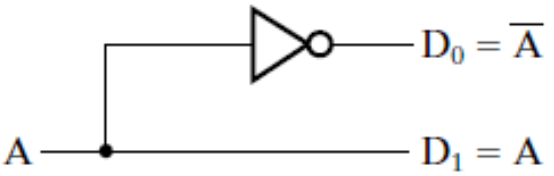
解碼器

Decoder Examples

- **1-to-2-Line Decoder**

A	D_0	D_1
0	1	0
1	0	1

(a)

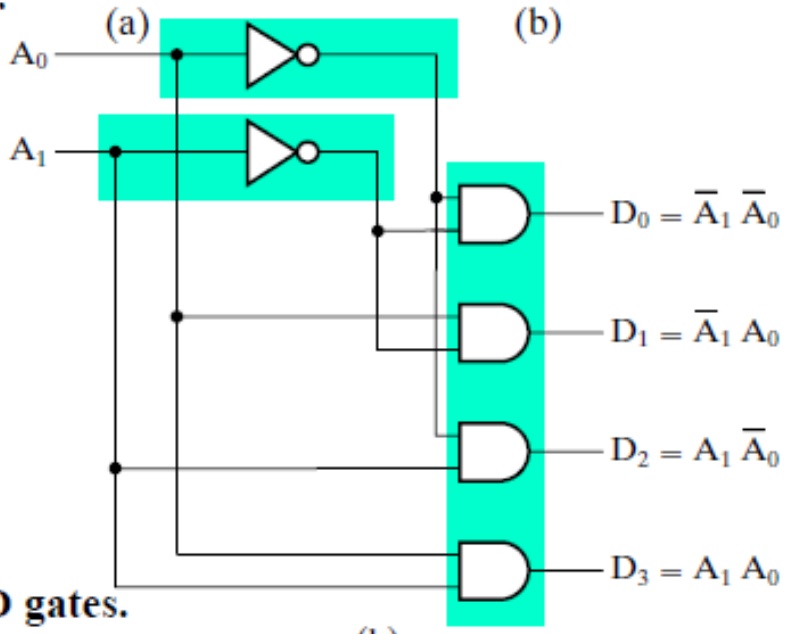


(b)

- **2-to-4-Line Decoder**

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)



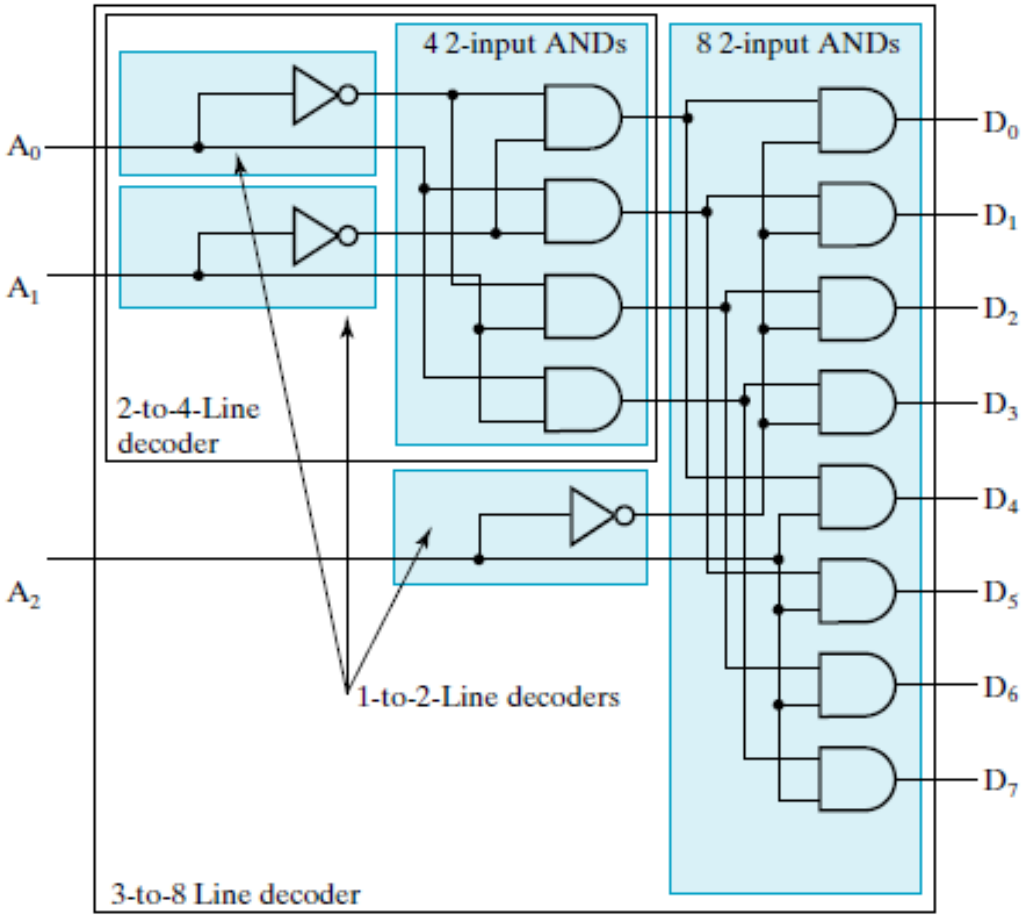
(b)

- **Note that the 2-4-line made up of 2 1-to-2-line decoders and 4 AND gates.**

解碼器的擴展

Decoder Expansion - Example 1

- **Result**



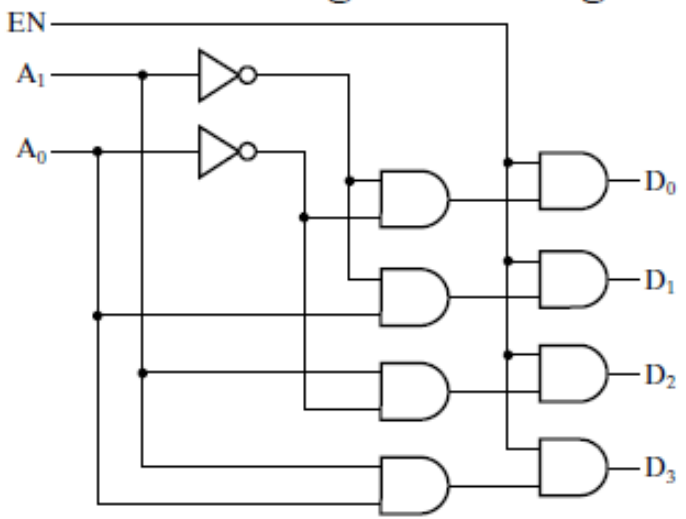
有 Enable 的解碼器

Decoder with Enable

- In general, attach *m*-enabling circuits to the outputs
- See truth table below for function
  - Note use of X's to denote both 0 and 1
  - Combination containing two X's represent four binary combinations
- Alternatively, can be viewed as distributing value of signal EN to 1 of 4 outputs
- In this case, called a *demultiplexer*

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a)



(b)

Priority Encoder Example

- Priority encoder with 5 inputs (D_4, D_3, D_2, D_1, D_0) - highest priority to most significant 1 present - Code outputs A2, A1, A0 and V where V indicates at least one 1 present.

No. of Min-terms/Row	Inputs					Outputs			
	D4	D3	D2	D1	D0	A2	A1	A0	V
1	0	0	0	0	0	X	X	X	0
1	0	0	0	0	1	0	0	0	1
2	0	0	0	1	X	0	0	1	1
4	0	0	1	X	X	0	1	0	1
8	0	1	X	X	X	0	1	1	1
16	1	X	X	X	X	1	0	0	1

- Xs in input part of table represent 0 or 1; thus table entries correspond to product terms instead of minterms. The column on the left shows that all 32 minterms are present in the product terms in the table

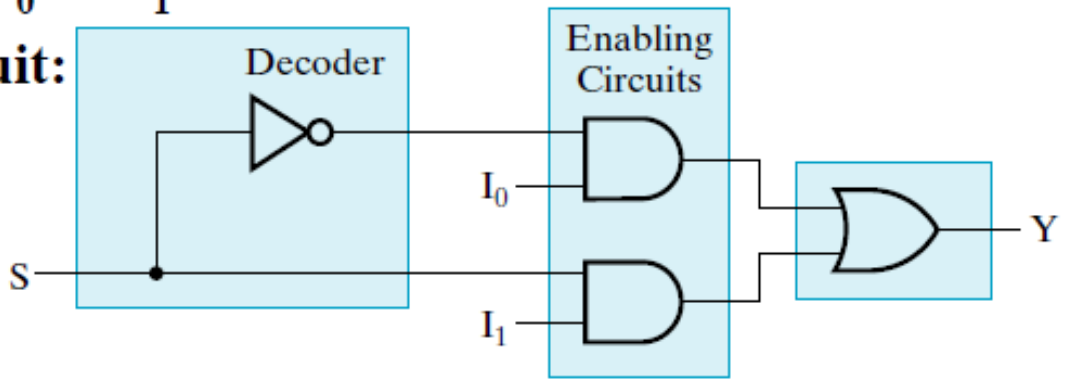
2 對 1 多工器

2-to-1-Line Multiplexer

- Since $2 = 2^1$, $n = 1$
- The single selection variable S has two values:
  - S = 0 selects input I_0
  - S = 1 selects input I_1
- The equation:

$$Y = \overline{S}I_0 + SI_1$$

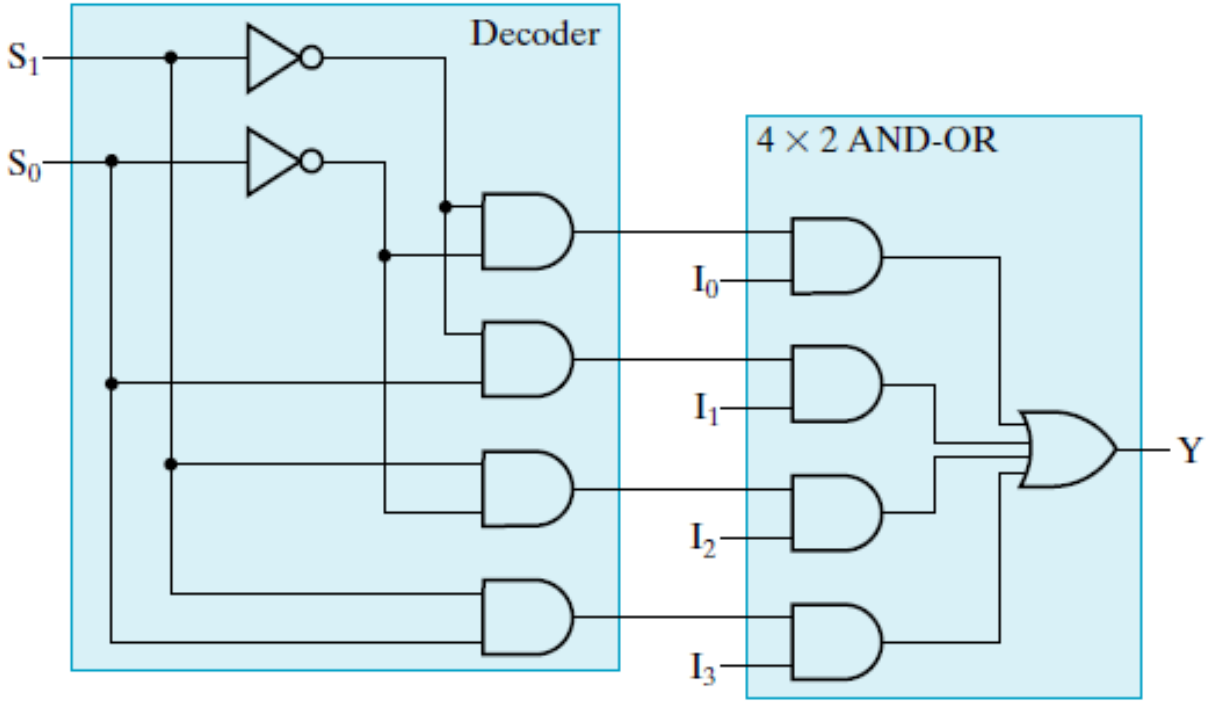
- The circuit:



4 對 1 多工器

Example: 4-to-1-line Multiplexer

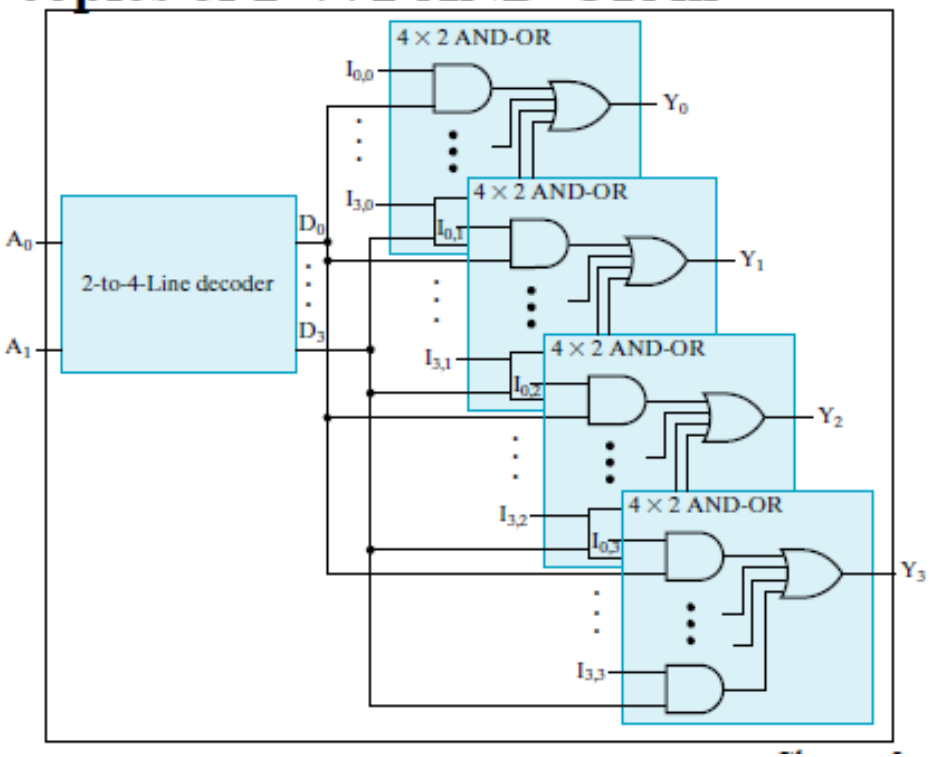
- 2-to-2^2-line decoder
- $2^2 \times 2$ AND-OR



多工器的擴展

Multiplexer Width Expansion

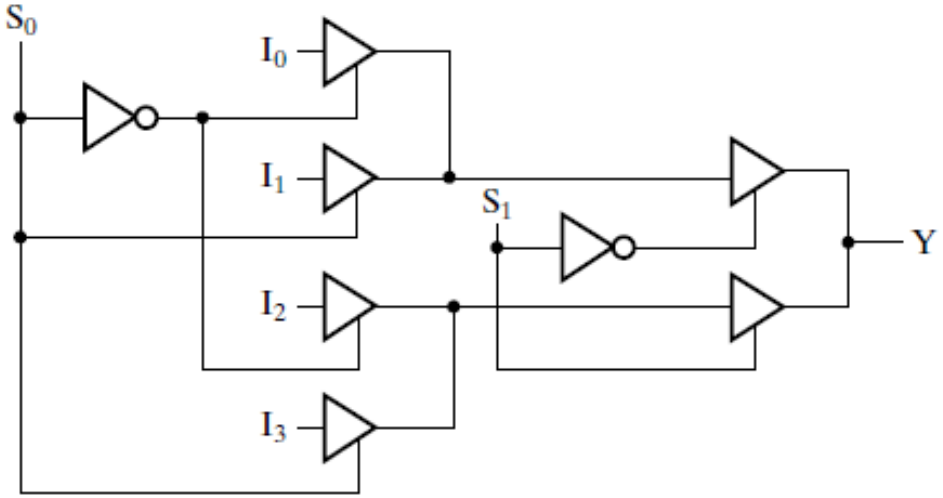
- Select "vectors of bits" instead of "bits"
- Use multiple copies of $2^n \times 2$ AND-OR in parallel
- Example: 4-to-1-line quad multi-plexer



使用三態緩衝器實作

Other Selection Implementations

- **Three-state logic in place of AND-OR**



(b)

- **Gate input cost = 14 compared to 22 (or 18) for gate implementation**

授權限制

- 本投影片 部份取材自 邏輯與計算機設計 之投影片
  - 來源： http://writphotec.com/mano4/
  - 使用時請遵照下列條款

Terms of Use

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.

第 11 章
State Machines (狀態機)

Computer Organization and Design Fundamental

書籍作者： David Tarnoff
投影片製作者：陳鍾誠

11.1 Introduction to State Machines

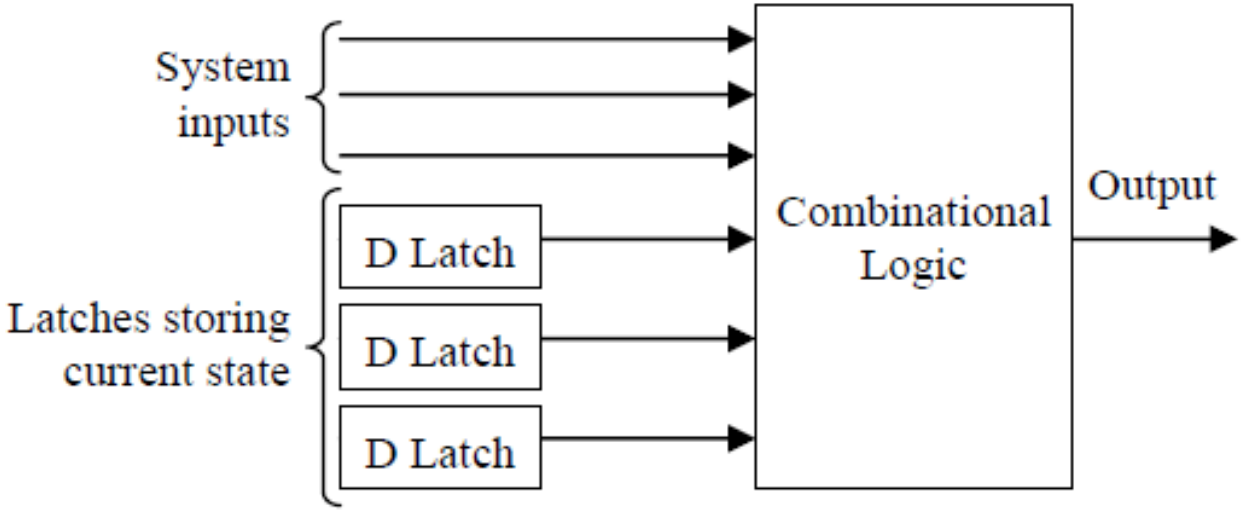


Figure 11-1 Adding Memory to a Digital Logic Circuit

11.1.1 States

traffic signal controls

- The most basic traffic signal controls an intersection with two directions, North-South and East-West for example

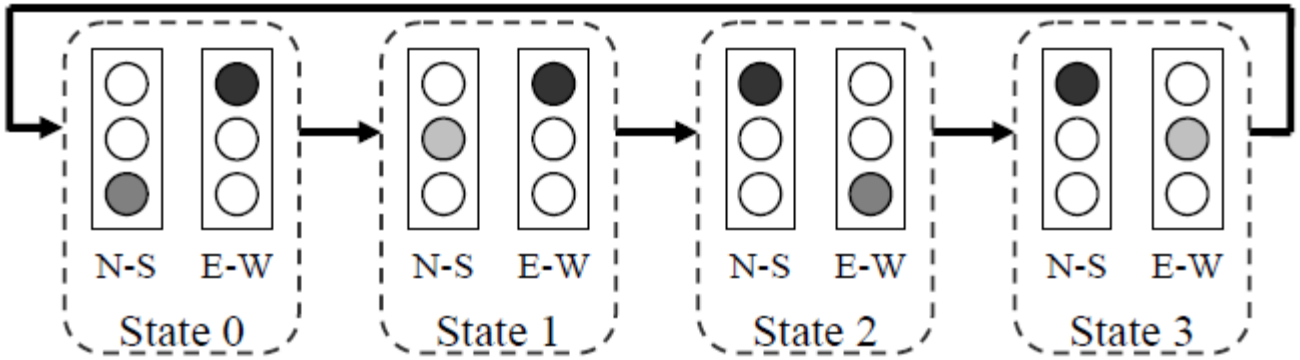


Figure 11-2 States of a Traffic Signal System

light bulb

- A state machine might also be as simple as a light bulb. The light bulb can have two states: on and off. The light switch moves the condition of the bulb from one state to another

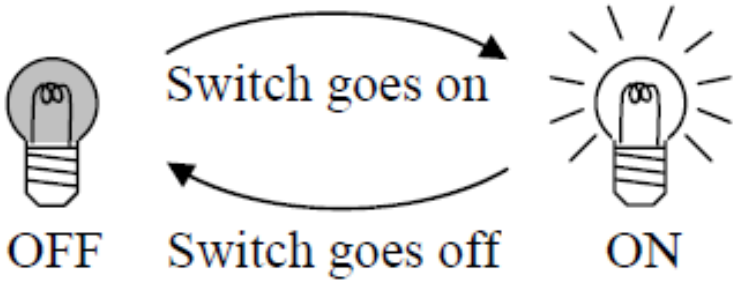


Figure 11-3 States of a Light Bulb

11.1.2 State Diagrams

State Machine : light bulb

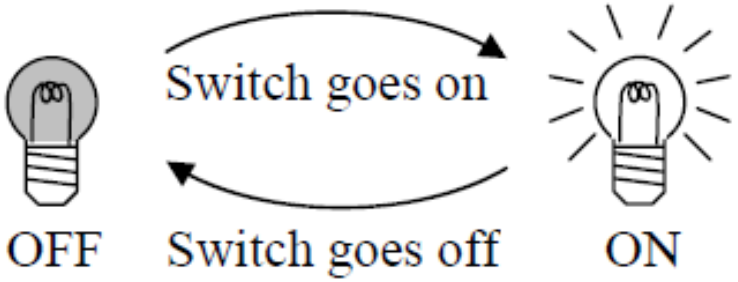


Figure 11-3 States of a Light Bulb

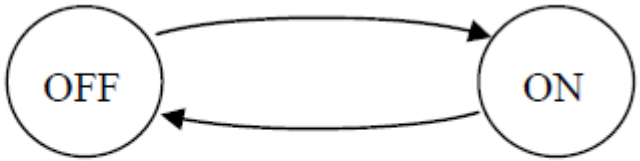


Figure 11-4 State Diagram for Light Bulb State Machine

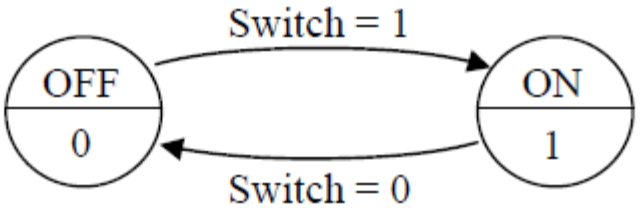


Figure 11-5 Complete State Diagram for Light Bulb State Machine

Up-Down Binary Counter

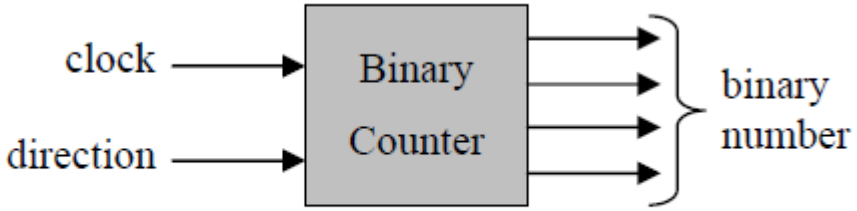


Figure 11-6 Block Diagram of an Up-Down Binary Counter

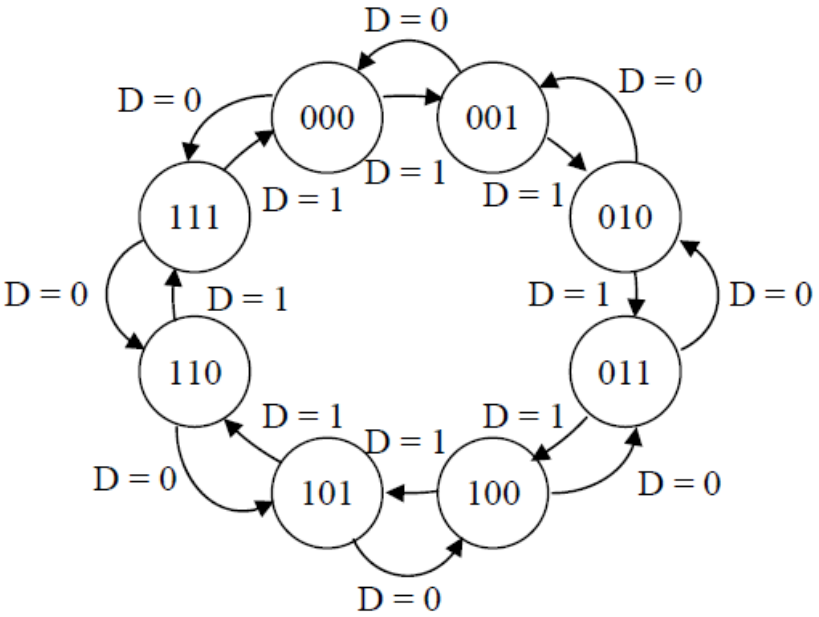


Figure 11-7 State Diagram for a 3-Bit Up-Down Binary Counter

Up-Down Binary Counter (加入 Reset)

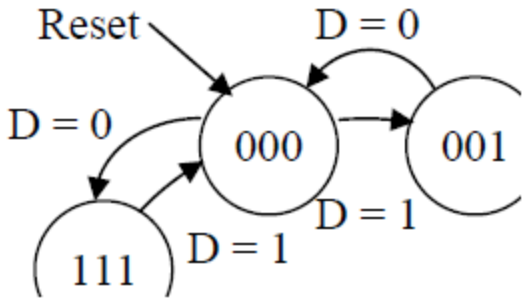


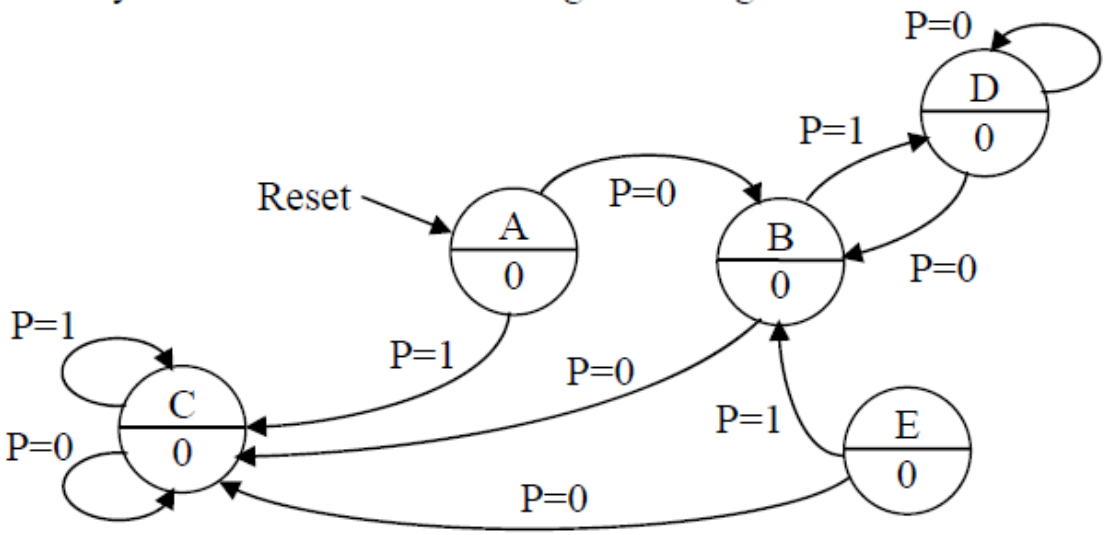
Figure 11-8 Sample of a Reset Indication in a State Diagram

11.1.3 Errors in State Diagrams

- Any state other than an initial state that has no transitions going ***into*** it should be removed since it is impossible to reach that state.
- For a system with n inputs, there should be exactly 2^n transitions coming ***out*** of every state, one for each pattern of ones and zeros for the n inputs. Some transitions may come back to the current state, but every input must be accounted for. Missing transitions should be added while duplicates should be removed.

Example

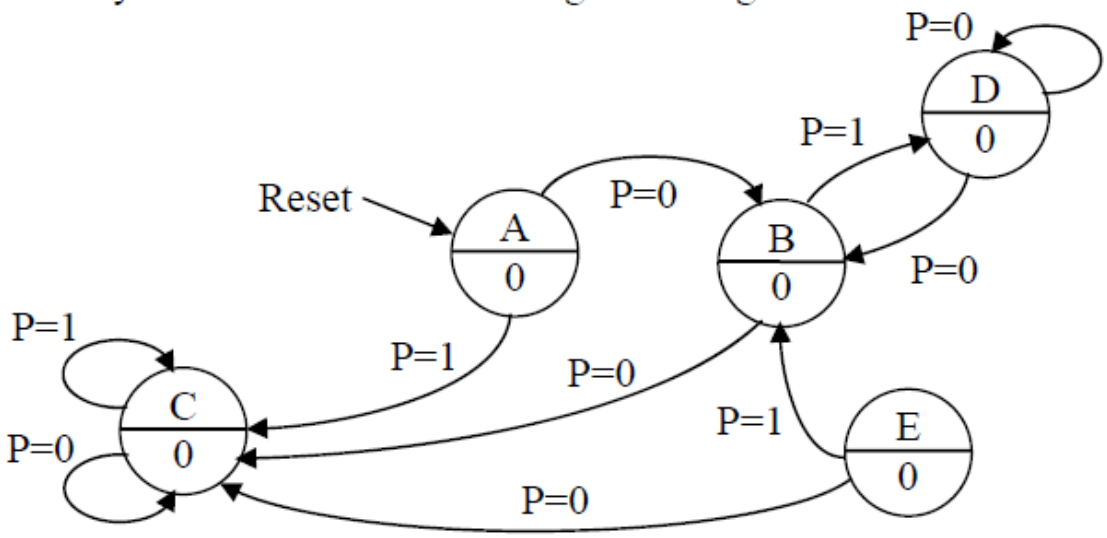
Identify the errors in the following state diagram.



Example : State Diagrams Error

Example

Identify the errors in the following state diagram.



Solution

Error 1 – There is no way to get to state E. It should be removed. Although state A has no transitions to it, it is not a problem because it is the initial state.

Error 2 – The transition from state D for P=0 is defined twice while the transition for P=1 is never defined.

State Machines 的電路

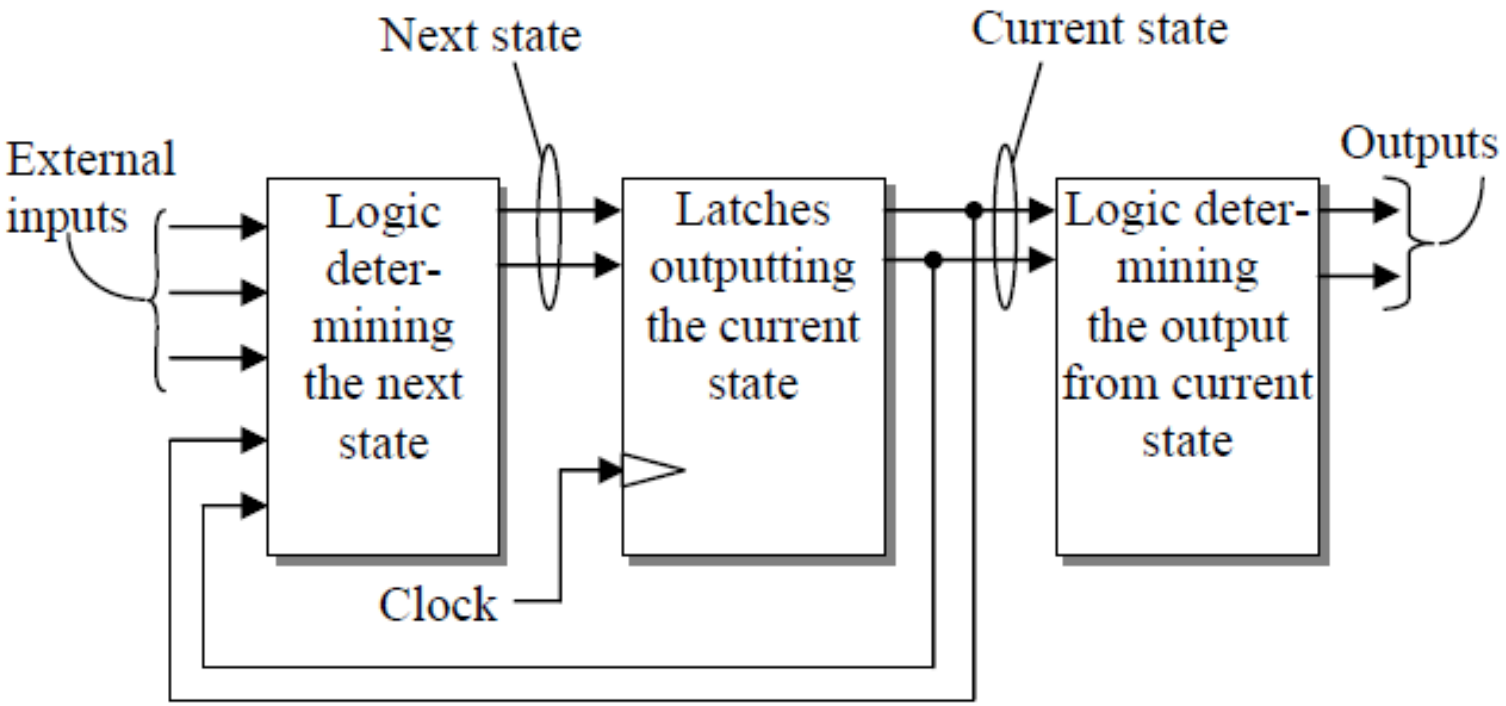


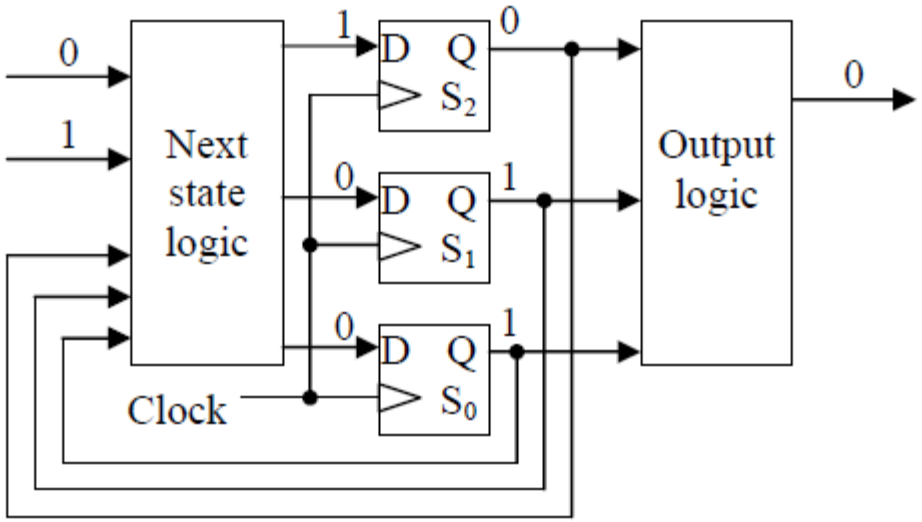
Figure 11-9 Block Diagram of a State Machine

State Machines 的電路 -- 範例 1

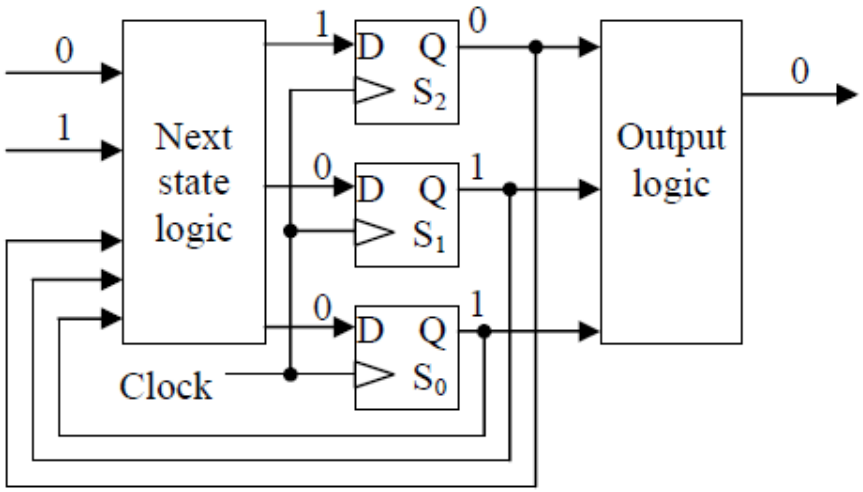
Example

The block diagram below represents a state machine. Answer the following questions based on the state machine's components and the digital values present on each of the connections.

- What is the maximum number of states this system could have?
- How many rows are in the truth table defining the output?
- How many rows are in the truth table defining the next state?
- What is the current state of this system?
- If the clock were to pulse right now, what would the next state be?



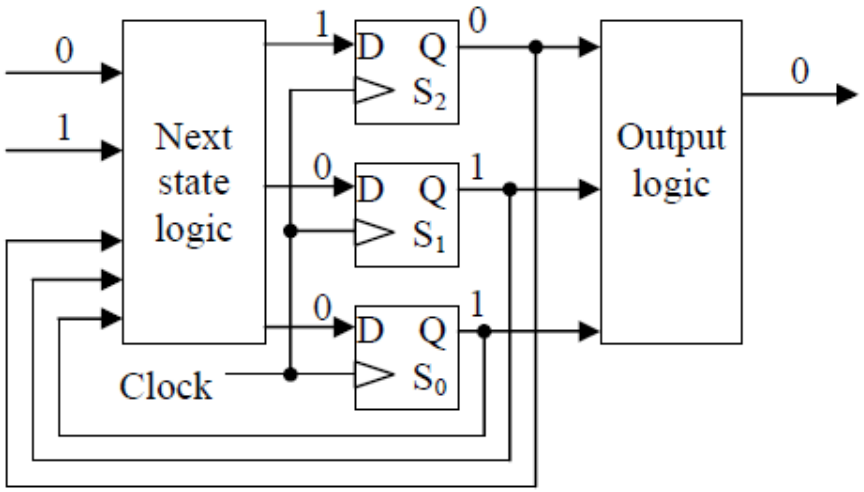
範例 1 -- 解答 (a,b)



What is the maximum number of states this system could have? Since the system has 3 latches, then the numbers 000_2, 001_2, 010_2, 011_2, 100_2, 101_2, 110_2, and 111_2 can be stored. Therefore, this state machine can have up to **eight states**.

How many rows are in the truth table defining the output? Since the output is based on the current state which is represented by the latches, and since there are three latches, the logic circuit for the output has three inputs. With three inputs, there are $2^3 = 8$ possible patterns of ones and zeros into the circuit, and hence, **8 rows** in the truth table.

範例 1 - 解答 (c,d,e)

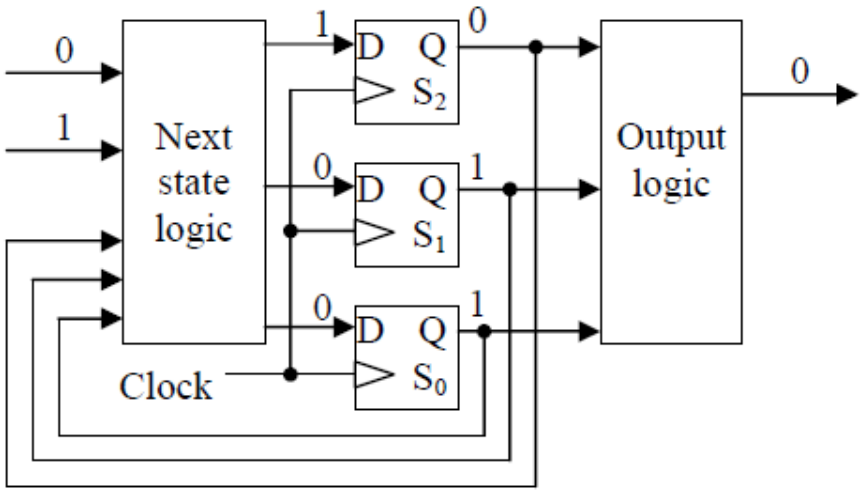


How many rows are in the truth table defining the next state? Since the next state of the state machine, i.e., the value on the input lines to the latches, depends on the current state fed back into the next state logic and the system inputs, then there are five inputs that determine the next state. Therefore, the inputs to the next state logic have $2^5 = 32$ possible patterns of ones and zeros. This means that the next state logic truth table has **32 rows**.

What is the current state of this system? The current state equals the binary value stored in the latches. Remembering that S_0 is the LSB while S_2 is the MSB, this means that the current state is $\mathbf{011_2 = 3_{10}}$.

If the clock were to pulse right now, what would the next state be? The next state is the binary value that is present at the D inputs to the latches. Once again, S_0 is the LSB and S_2 is the MSB. Therefore, the next state is $\mathbf{100_2 = 4_{10}}$.

範例 1 -- 解答 (a,b)



What is the maximum number of states this system could have? Since the system has 3 latches, then the numbers 000_2, 001_2, 010_2, 011_2, 100_2, 101_2, 110_2, and 111_2 can be stored. Therefore, this state machine can have up to **eight states**.

How many rows are in the truth table defining the output? Since the output is based on the current state which is represented by the latches, and since there are three latches, the logic circuit for the output has three inputs. With three inputs, there are $2^3 = 8$ possible patterns of ones and zeros into the circuit, and hence, **8 rows** in the truth table.

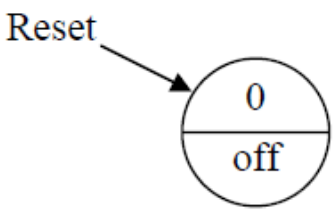
11.2 State Machine Design Process

燈號狀態機

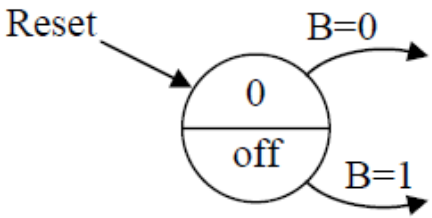
- 按钮亮燈，再按熄滅
  - The example we will be using is a push button circuit used to control a light bulb. When the button is pressed, the light bulb turns on. When the button is released, it stays on. Pushing the button a second time turns off the bulb, and the bulb stays off when the button is released

燈號狀態機 -- 設計過程

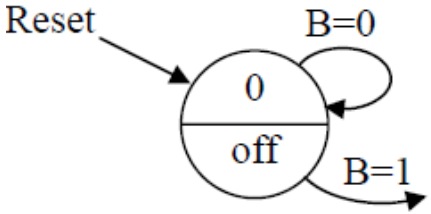
(1)



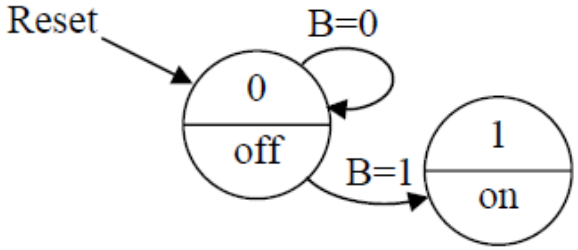
(2)



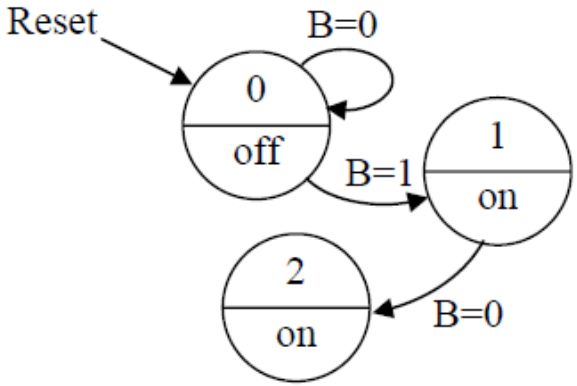
(3)



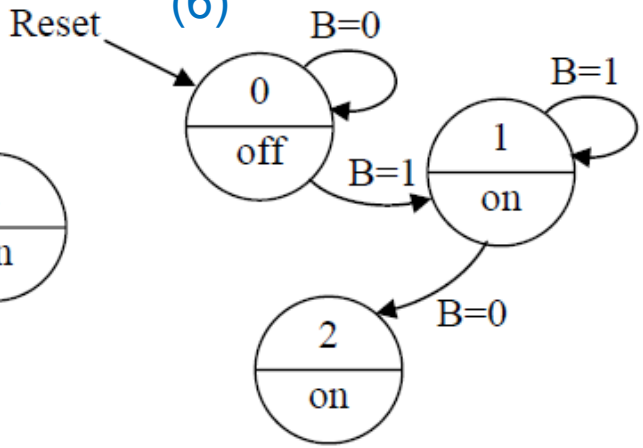
(4)



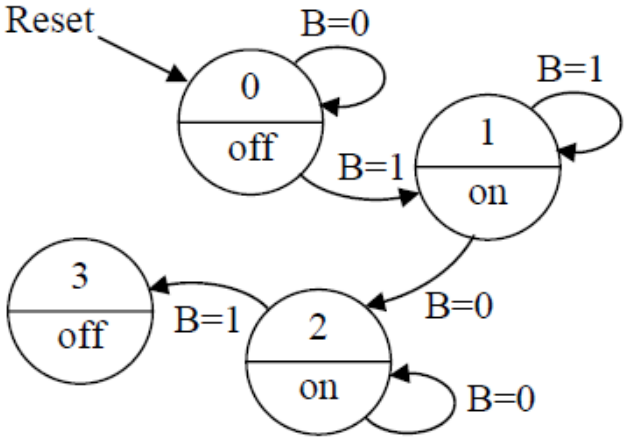
(5)



(6)



(7)



燈號狀態機 – 電路方塊圖

- 狀態數

Table 11-1 List of States for Push Button Circuit

State	Numeric Value	
	Decimal	Binary
Bulb off; button released	0	00
Bulb on; button pressed	1	01
Bulb on; button released	2	10
Bulb off; button pressed	3	11

- 電路

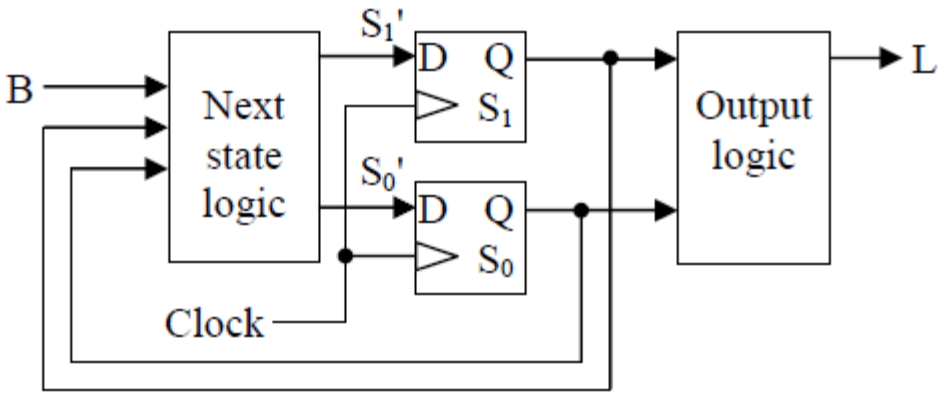


Figure 11-18 Block Diagram for Push Button Circuit

燈號狀態機 – 電路真值表

Table 11-2 Next State Truth Table for Push Button Circuit

S_1	S_0	B	S_1'	S_0'	
0	0	0	0	0	← State 0 stays in state 0 when B=0
0	0	1	0	1	← State 0 goes to state 1 when B=1
0	1	0	1	0	← State 1 goes to state 2 when B=0
0	1	1	0	1	← State 1 stays in state 1 when B=1
1	0	0	1	0	← State 2 stays in state 2 when B=0
1	0	1	1	1	← State 2 goes to state 3 when B=1
1	1	0	0	0	← State 3 goes to state 0 when B=0
1	1	1	1	1	← State 3 stays in state 3 when B=1

Table 11-3 Output Truth Table for Push Button Circuit

S_1	S_0	L	
0	0	0	← State 0: bulb is off
0	1	1	← State 1: bulb is on
1	0	1	← State 2: bulb is on
1	1	0	← State 3: bulb is off

燈號狀態機 – 化簡

- 卡諾圖化簡

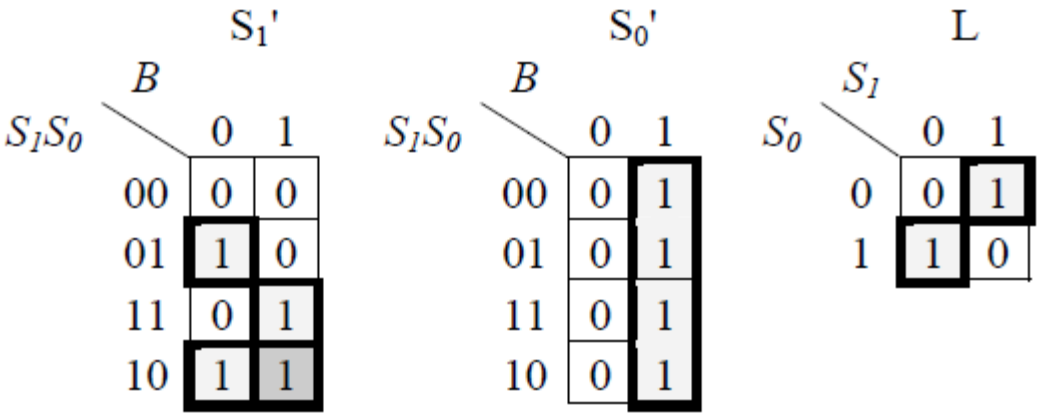


Figure 11-19 K-Maps for S_1', S_0', and L of Push Button Circuit

- 運算式

From these Karnaugh maps, we get the following boolean expressions:

$$S_1' = \overline{S_1} \cdot S_0 \cdot \overline{B} + S_1 \cdot B + S_1 \cdot \overline{S_0}$$

$$S_0' = B$$

$$L = \overline{S_1} \cdot S_0 + S_1 \cdot \overline{S_0} = S_1 \oplus S_0$$

燈號狀態機 – 最終電路圖

$$S_1' = \overline{S_1} \cdot S_0 \cdot \overline{B} + S_1 \cdot B + S_1 \cdot \overline{S_0}$$

$$S_0' = B$$

$$L = \overline{S_1} \cdot S_0 + S_1 \cdot \overline{S_0} = S_1 \oplus S_0$$

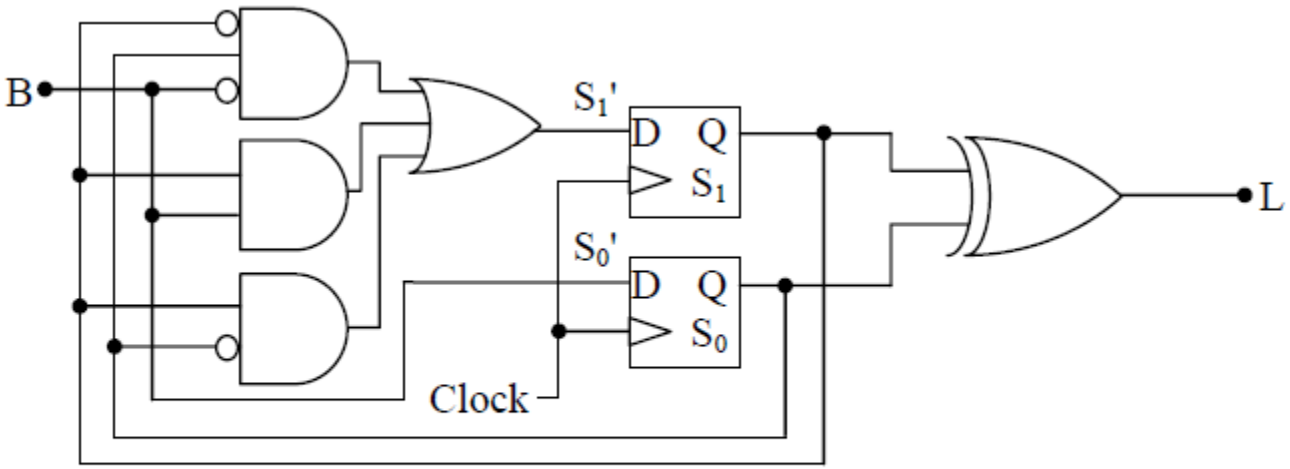


Figure 11-20 Finished Push Button Circuit

燈號狀態機 – 有這麼複雜嗎？

- 如果將 State 2, 3 反過來

Table 11-4 Revised List of States for Push Button Circuit

State	Numeric Value	
	Decimal	Binary
Bulb off; button released	0	00
Bulb on; button pressed	1	01
Bulb on; button released	3	11
Bulb off; button pressed	2	10

This gives us a new boolean expression for L.

S_1	S_0	L
0	0	0
0	1	1
1	0	0
1	1	1

S_0 \ S_1	0	1
0	0	1
1	0	1

$$L = S_1$$

Figure 11-21 Revised Truth Table and K Map for Push Button Circuit

11.3 Another State Machine Design: Pattern Detection

- 偵測 101

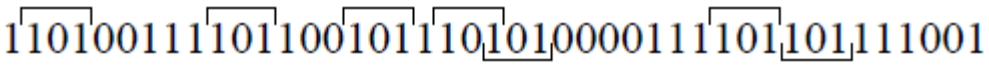


Figure 11-22 Identifying the Bit Pattern "101" in a Bit Stream

- 設計想法
  - The state machine used to detect the bit pattern "101" will have four states, each state representing the number of bits that we have received up to this point that match the pattern: 0, 1, 2, or 3.

偵測 101 – 狀態圖

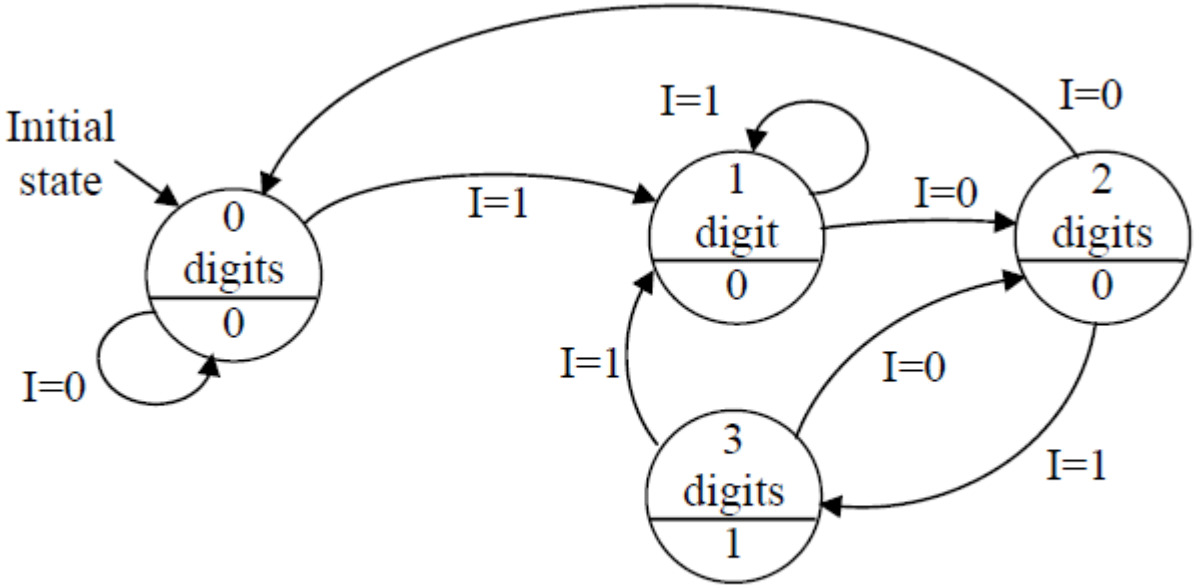


Figure 11-23 State Diagram for Identifying the Bit Pattern "101"

偵測 101 – 真值表

Table 11-5 List of States for Bit Pattern Detection Circuit

State	Numeric Value	
	Decimal	Binary
No bits of the pattern have been received	0	00
One bit of the pattern has been received	1	01
Two bits of the pattern have been received	2	10
Three bits of the pattern have been received	3	11

Next State

S_1	S_0	I	S_1'	S_0'
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	0
1	1	1	0	1

Output

S_1	S_0	P
0	0	0
0	1	0
1	0	0
1	1	1

Figure 11-24 Next State and Output Truth Tables for Pattern Detect

偵測 101 – 卡諾圖

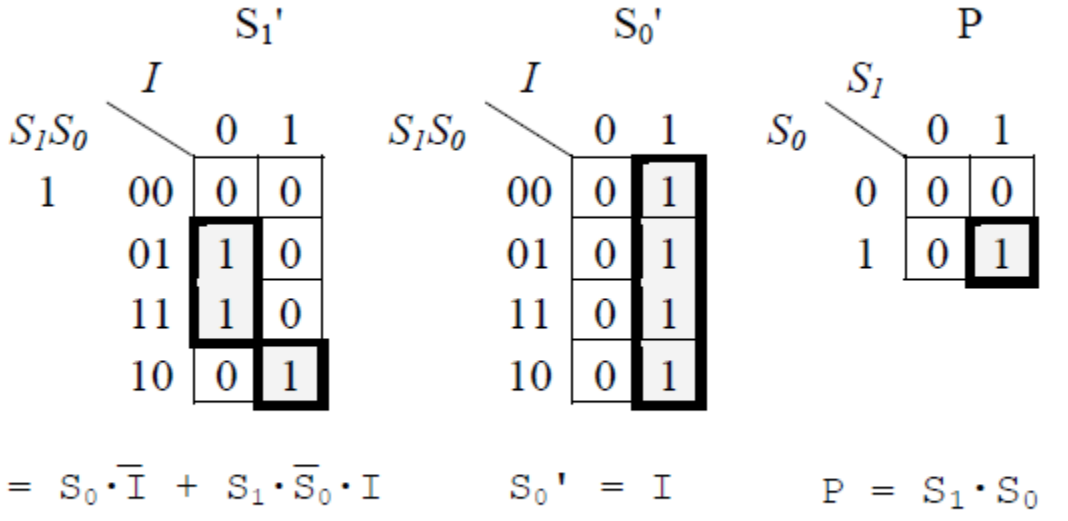


Figure 11-25 K-Maps for S_1', S_0', and P of Pattern Detect Circuit

偵測 101 – 最終電路

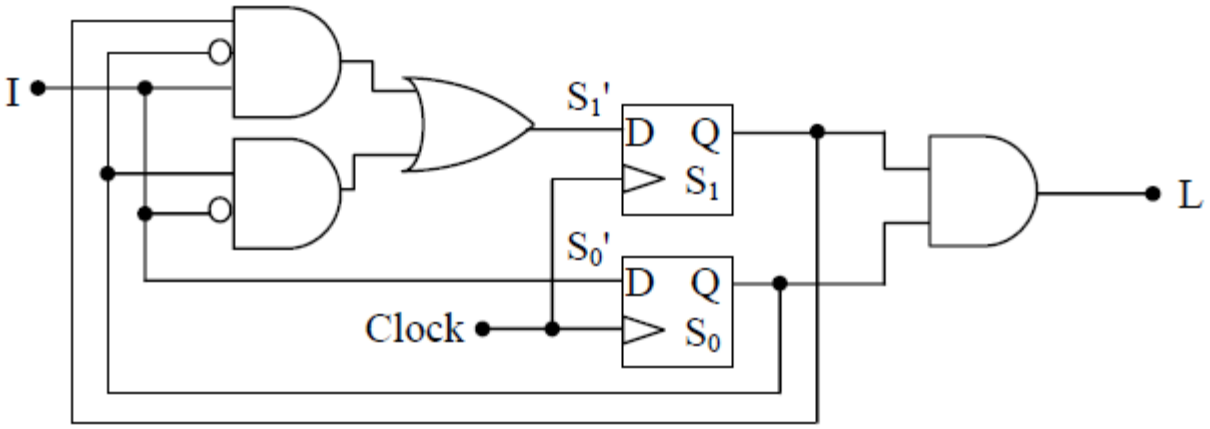


Figure 11-26 Final Circuit to Identify the Bit Pattern "101"

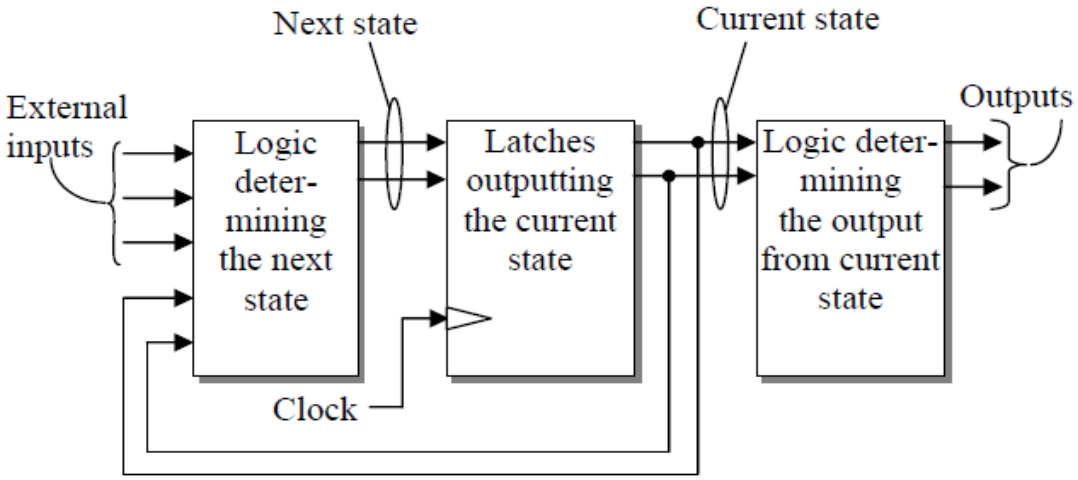
11.4 Mealy Versus Moore State Machines

- ***Moore machine***
  - The state machine design covered in the previous sections is referred to as a ***Moore machine. The distinguishing characteristic of a Moore*** machine is that its output is determined only by the current state.

- ***Mealy machine***
  - The output of the second type of state machine, the ***Mealy machine,*** is based on both the current state of the machine *and the system's input.*

Moore 與 Mealy 狀態機的不同 – 電路

- ***Moore machine***



- ***Mealy machine***

Figure 11-9 Block Diagram of a State Machine

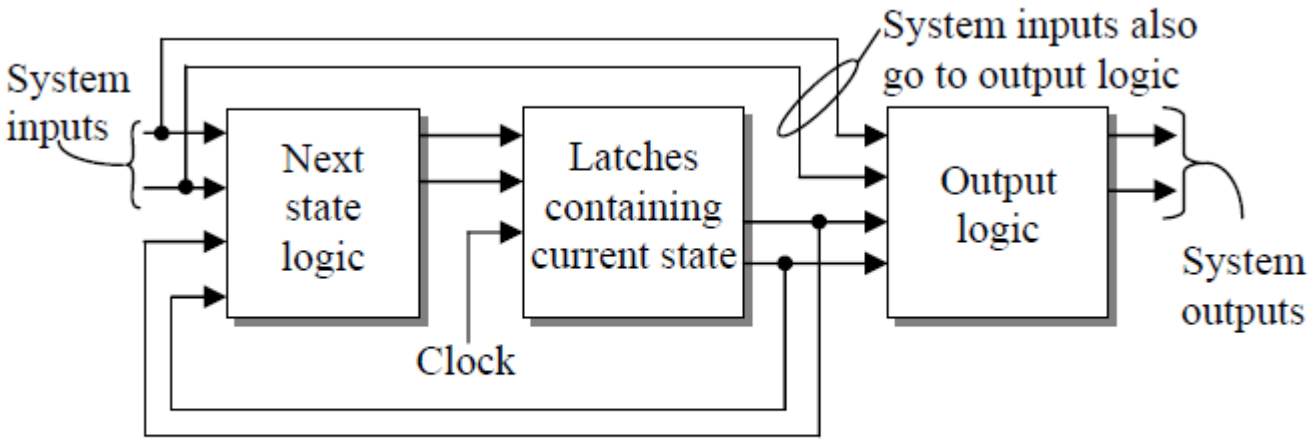


Figure 11-27 Basic Configuration of a Mealy Machine

Moore 與 Mealy 狀態機的不同 – 狀態圖

- ***Moore machine***

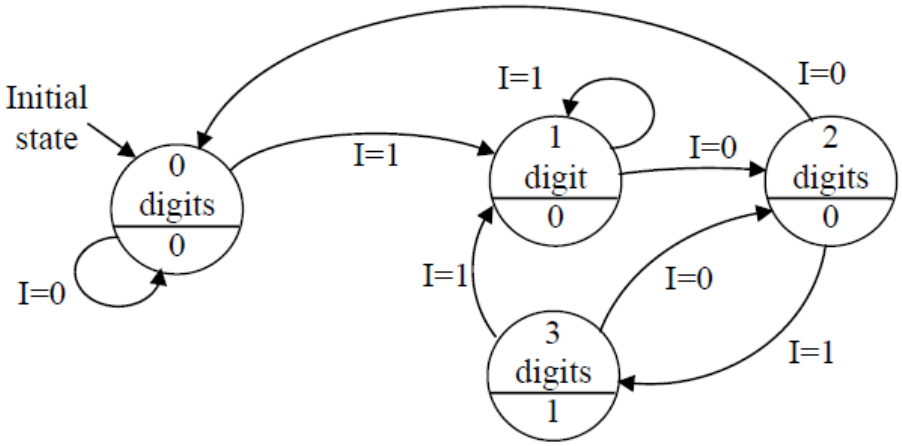


Figure 11-23 State Diagram for Identifying the Bit Pattern "101"

- ***Mealy machine***

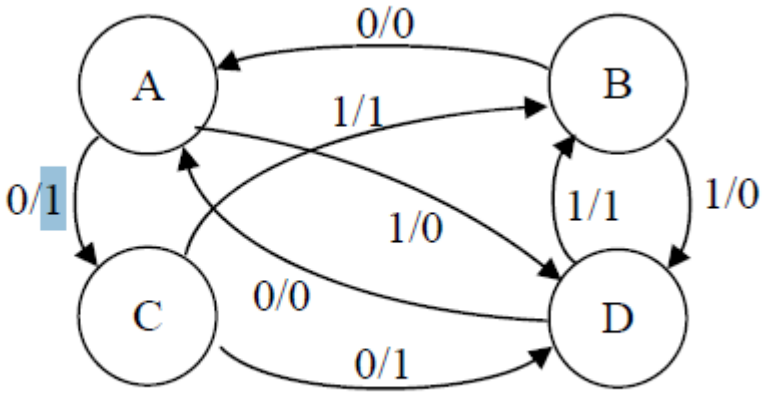


Figure 11-28 Sample State Diagram of a Mealy Machine

Mealy 狀態機 – 範例

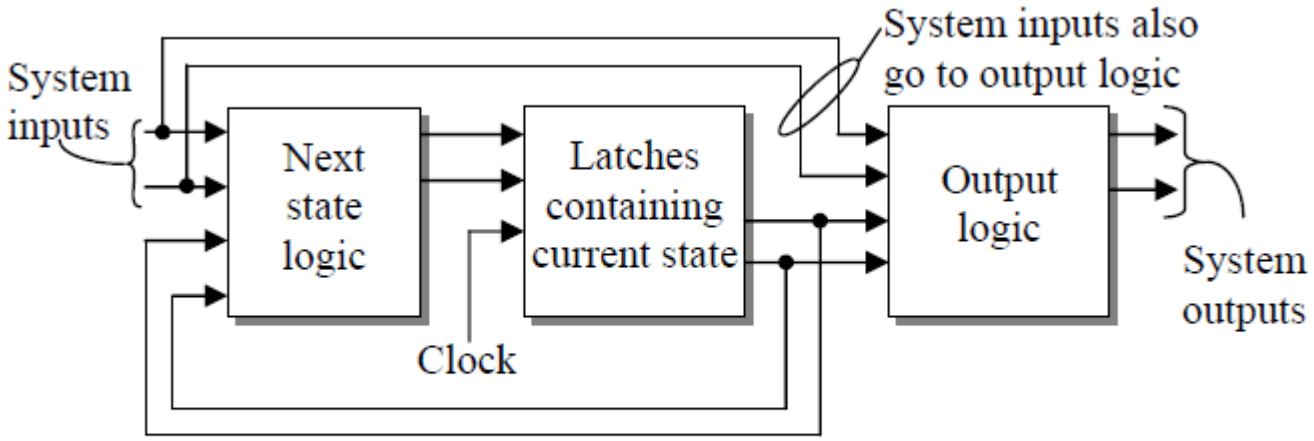


Figure 11-27 Basic Configuration of a Mealy Machine

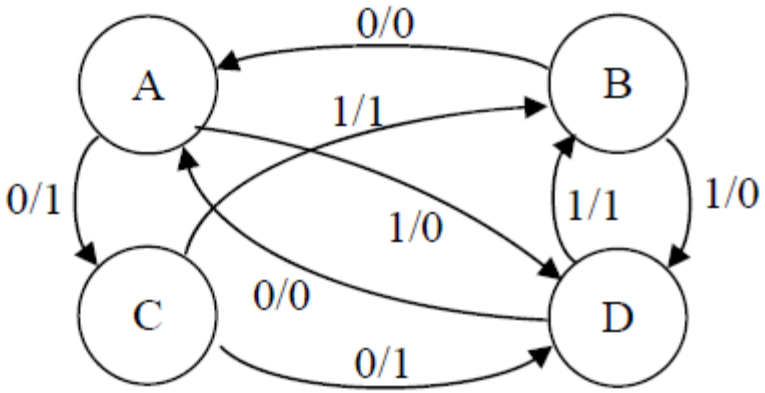


Figure 11-28 Sample State Diagram of a Mealy Machine

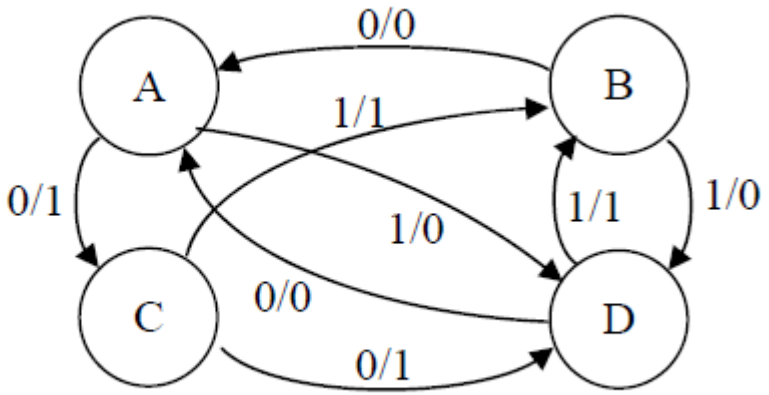


Figure 11-28 Sample State Diagram of a Mealy Machine

State	S_1	S_0	I	Y
A	0	0	0	1
	0	0	1	0
B	0	1	0	0
	0	1	1	0
C	1	0	0	1
	1	0	1	1
D	1	1	0	0
	1	1	1	1

Figure 11-29 Output Truth Table for Sample Mealy Machine