第六章

編輯器指令與狀態機器

編輯器指令

define 與 undefine 敘述：

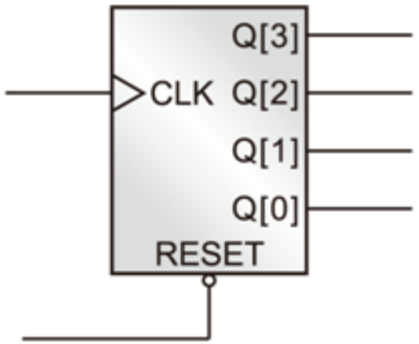
`define 巨集變數 <字串內容>

`undef 巨集變數

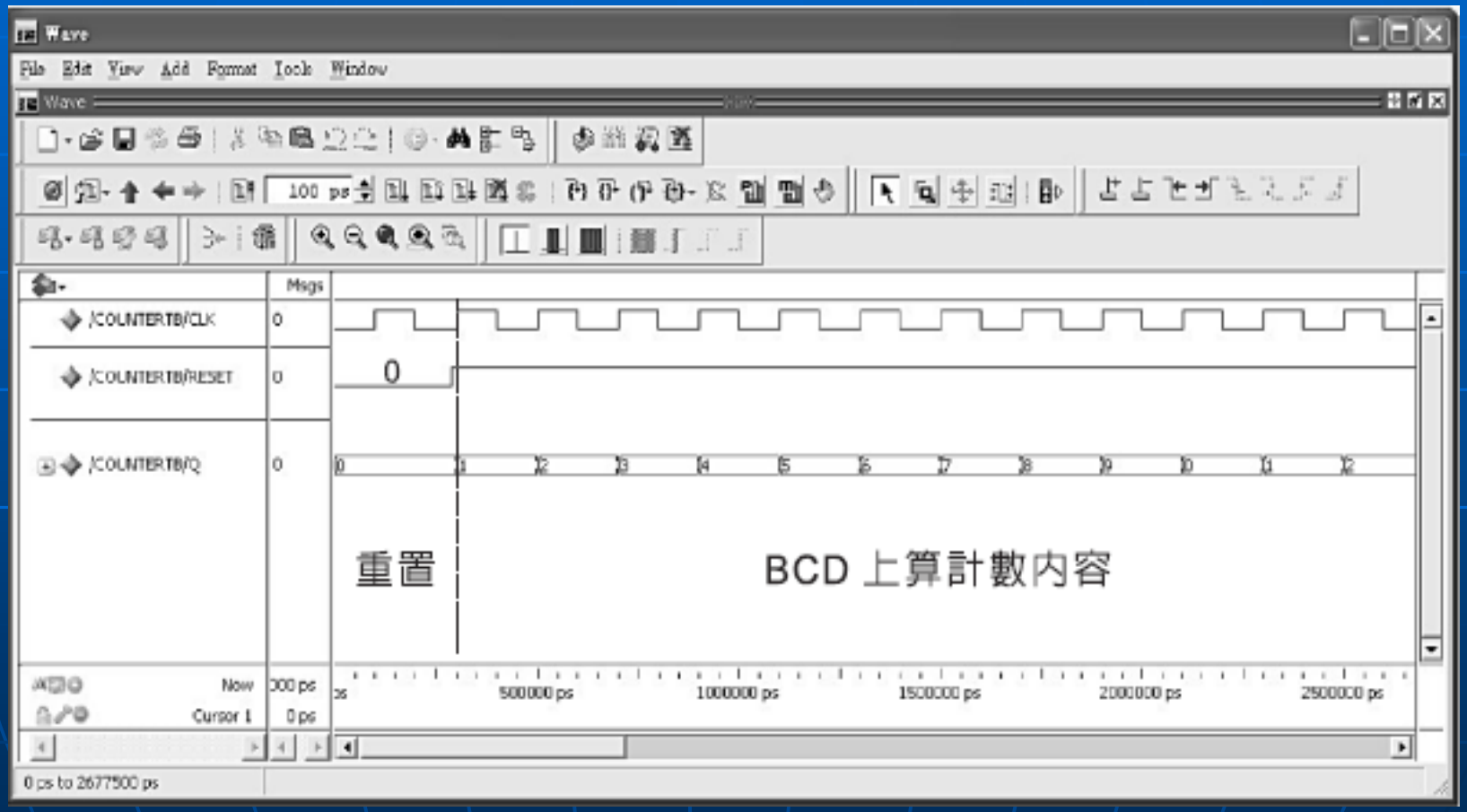
範例	檔名：UP_COUNTER_1DIG_DEFINE
以 `define 指令，設計一個從 0 上算到 N 的 BCD 計數器 (N 為 `define 後面的字串內容)，也就是當我們改變字串內容時，BCD 上算計數器的上限就會跟著改變。	

原始程式 (source program) :

```
 1  /*************************
 2   *    1 digital counter    *
 3   *      using define       *
 4   *  Filename : COUNTER.v   *
 5   *************************/
 6
 7  `define LAST 9
 8
 9  module COUNTER(CLK, RESET, Q);
10
11    input   CLK, RESET;
12    output  [3:0] Q;
13
14    reg [3:0] Q;
15
16    always @(posedge CLK or negedge RESET)
17      if (!RESET)
18        Q <= 4'h0;
19      else
20        if (Q == `LAST)
21          Q <= 4'h0;
22        else
23          Q <= Q + 1'b1;
24
25  endmodule
```



功能模擬 ：


Wave
File Edit View Add Format Tools Window
Msgs
/COUNTERTB/CLK
/COUNTERTB/RESET
/COUNTERTB/Q
0
重置
BCD 上算計數內容
Now
Cursor 1
500000 ps
1000000 ps
1500000 ps
2000000 ps
2500000 ps
0 ps to 2677500 ps

`ifdef, `else, `endif 敘述

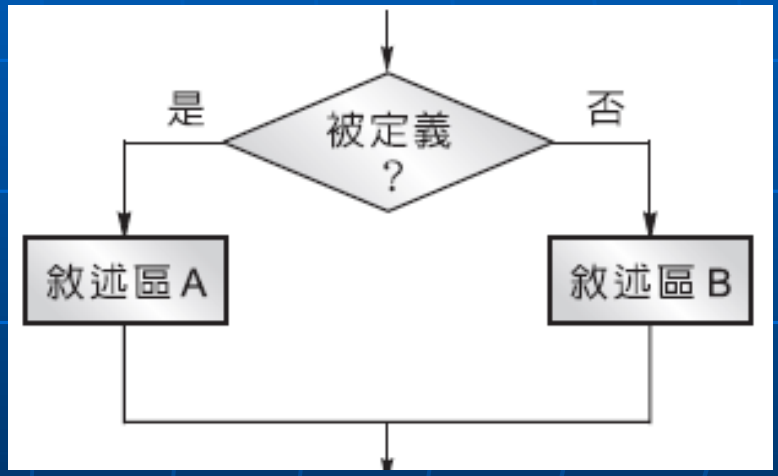
`ifdef 巨集變數　　流程圖：

敘述區 A；

`else

敘述區 B；

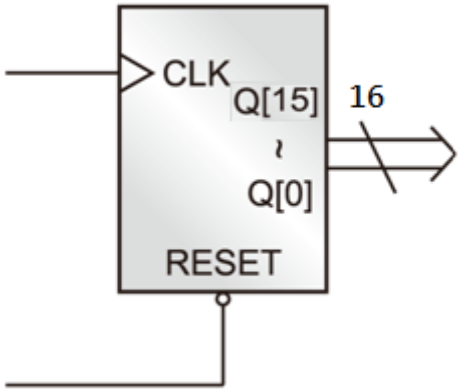
`endif



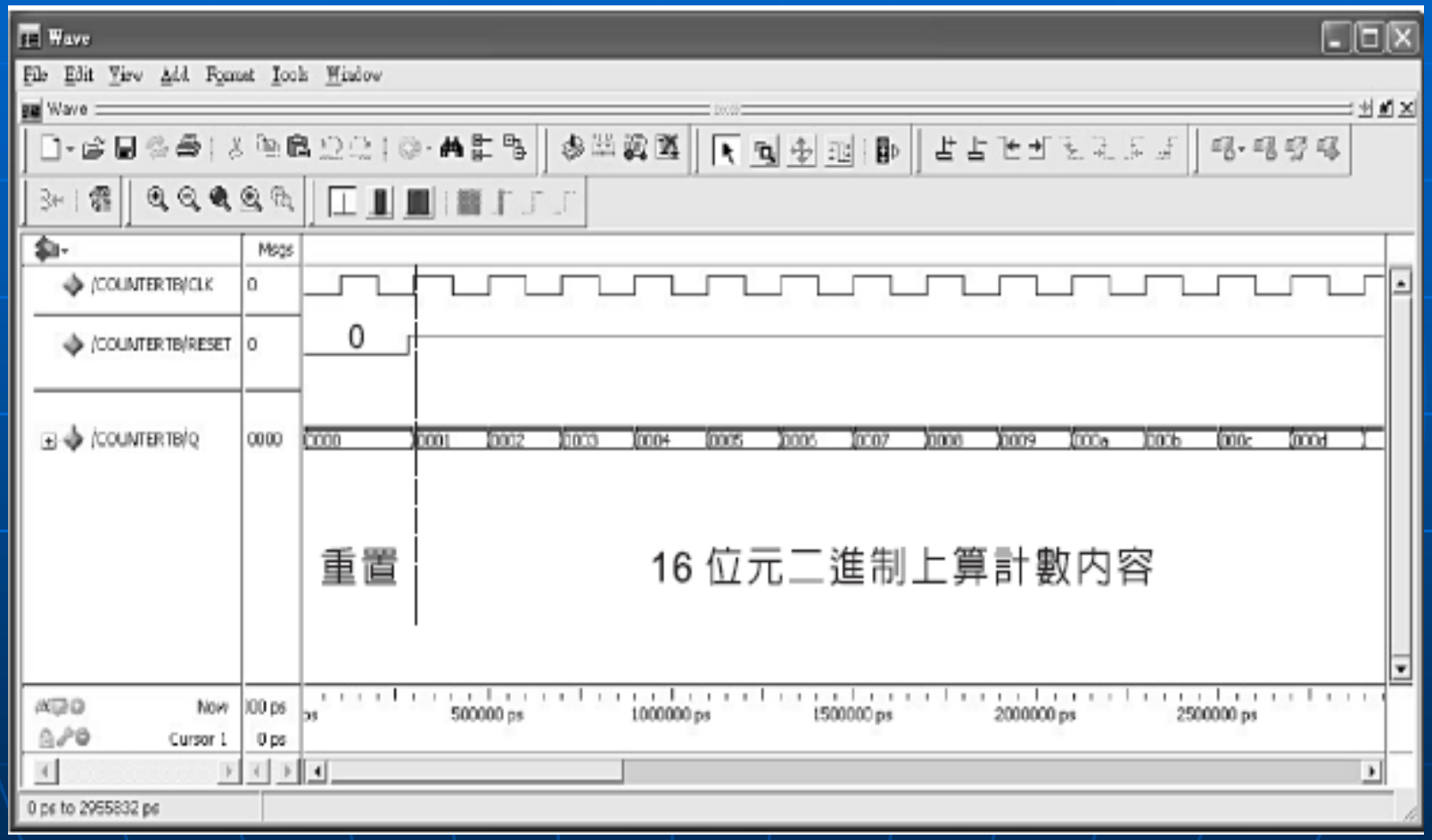
範例一	檔名：UP_COUNTER_LENGTH_IFDEF
說明下面原始程式合成的結果為何？	

原始程式 (source program) :

```
 1  /*************************
 2   *   1 digital counter    *
 3   *  using define, ifdef   *
 4   * Filename : COUNTER.v   *
 5   *************************/
 6
 7   `define LONG
 8
 9   `ifdef LONG
10      `define LENGTH 16
11   `else
12      `define LENGTH 8
13   `endif
14
15 module COUNTER(CLK, RESET, Q);
16
17    input   CLK, RESET;
18    output  [`LENGTH-1:0] Q;
19
20    reg [`LENGTH-1:0] Q;
21
22    always @(posedge CLK or negedge RESET)
23      if (!RESET)
24        Q <= 0;
25      else
26        Q <= Q + 1;
27
28 endmodule
```



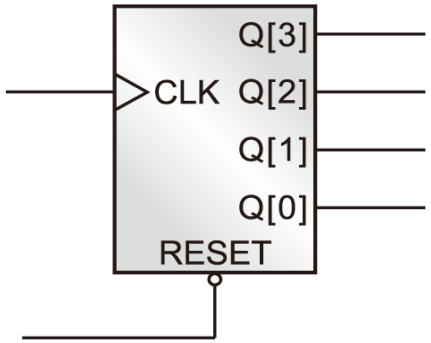
功能模擬 ：


Wave
File Edit View Add Format Tools Window
Wave
Msgs
/COUNTERTB/CLK
0
/COUNTERTB/RESET
0
/COUNTERTB/Q
0000
0
0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000a 000b 000c 000d
重置
16 位元二進制上算計數內容
Now
100 ps
Cursor 1
0 ps
500000 ps
1000000 ps
1500000 ps
2000000 ps
2500000 ps
0 ps to 2955832 ps

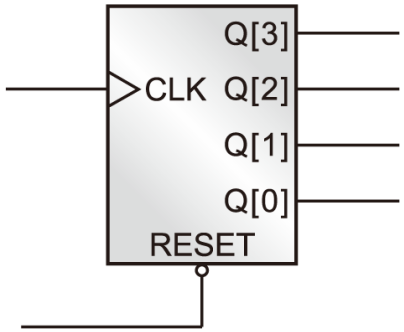
範例二	檔名：UP_COUNTER_4BITS_DEFINE_IFDEFINE
說明下面原始程式合成的結果為何？	

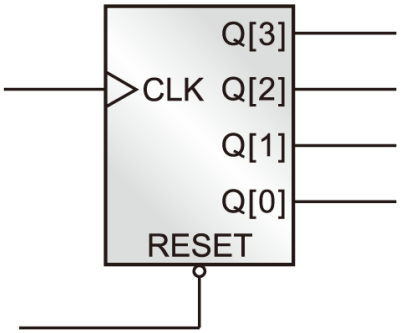
原始程式 (source program) :

```
 1  /*******************************
 2   *      4 bits up counter      *
 3   * using define, ifdef, else *
 4   *    Filename : COUNTER.v    *
 5   *******************************/
 6
 7  module COUNTER(CLK, RESET, Q);
 8
 9    input   CLK, RESET;
10    output  [3:0] Q;
11
12    reg [3:0] Q;
13
14   `define BINARY
15
16   `ifdef BCD
17
18   // 4 bits BCD counter
19
```

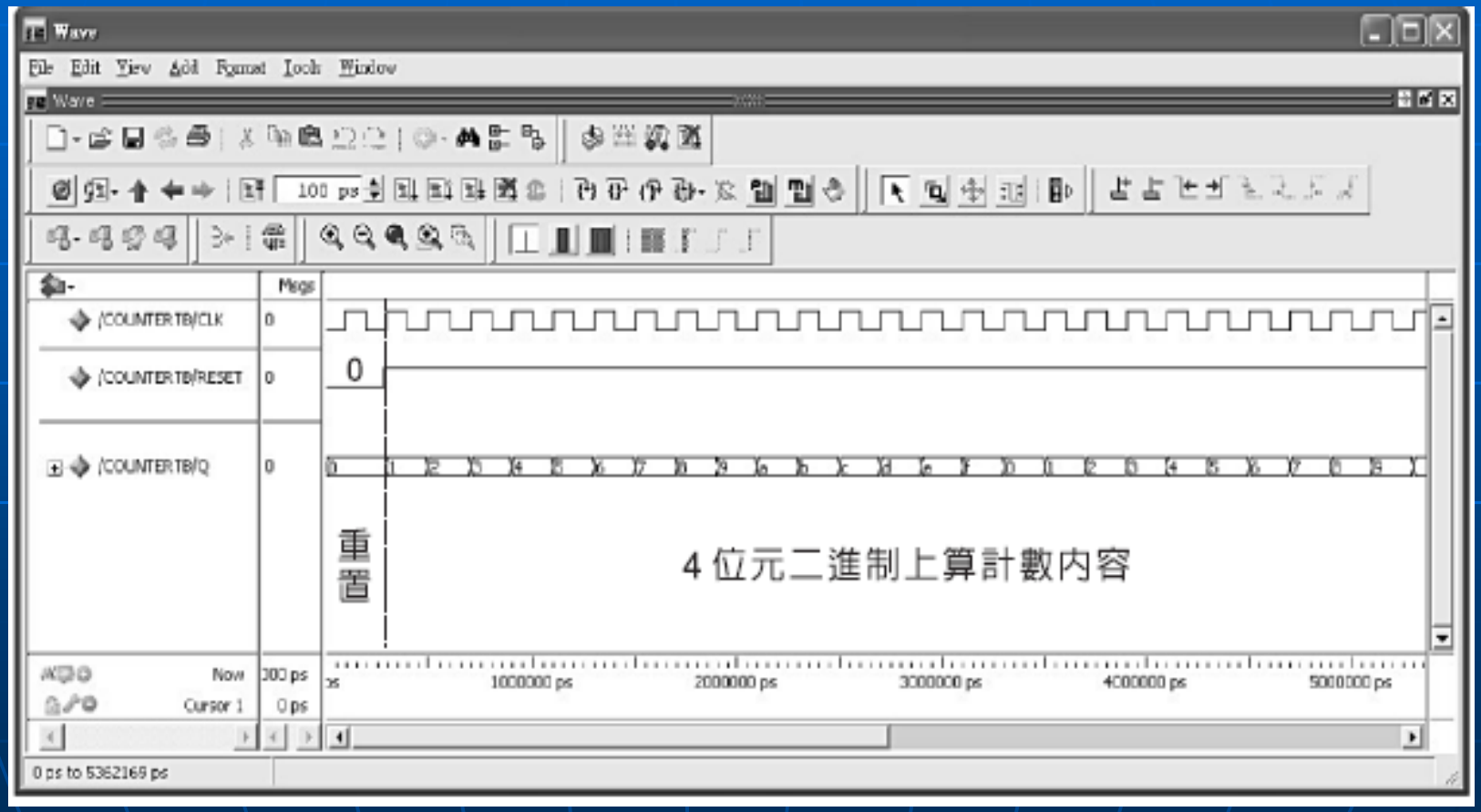


```
20   always @(posedge CLK or negedge RESET)
21     if (!RESET)
22       Q <= 4'h0;
23     else
24       if (Q == 4'h9)
25         Q <= 4'h0;
26       else
27         Q <= Q + 1'b1;
28
29  `else
30    `ifdef BINARY
31
32  // 4 bits binary counter
33
34   always @(posedge CLK or negedge RESET)
35     if (!RESET)
36       Q <= 4'h0;
37     else
38       Q <= Q + 1'b1;
39
40     `endif
41
42  `endif
43
44  endmodule
```





功能模擬 ：

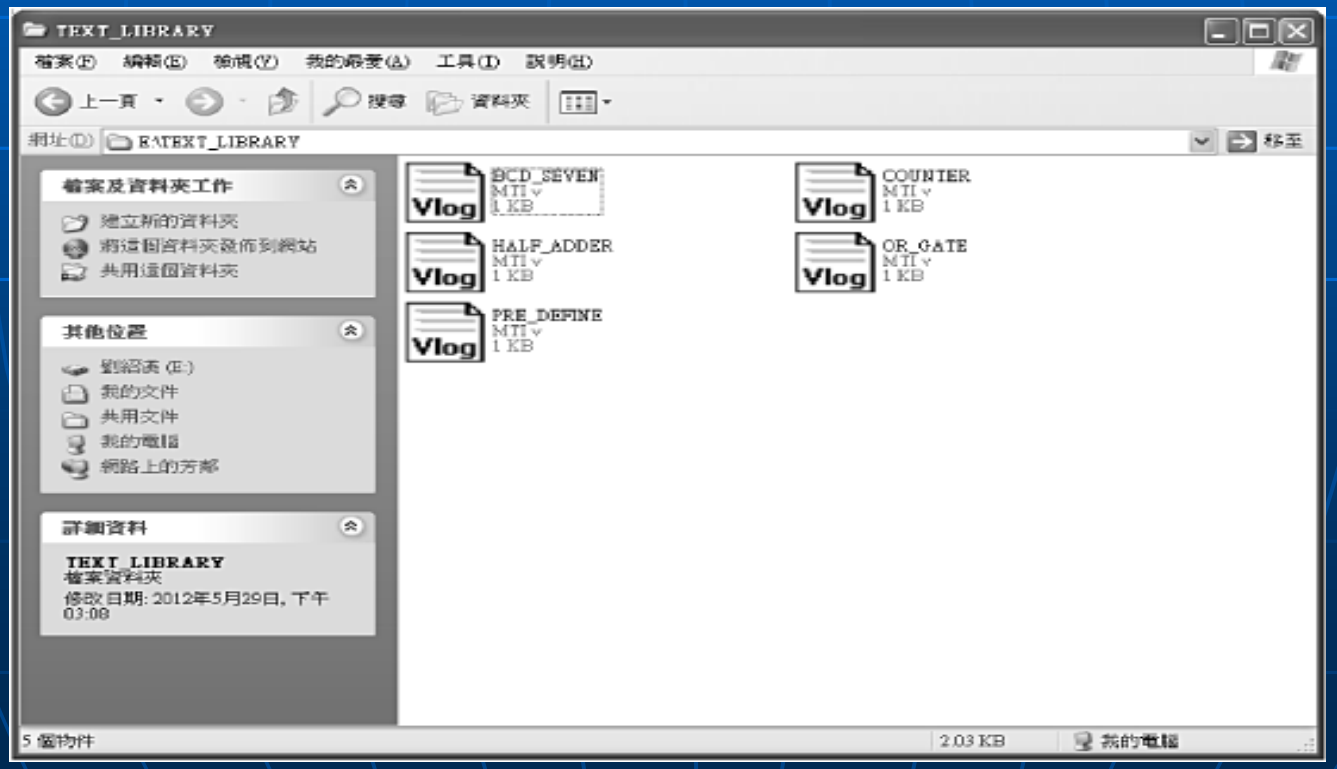

Wave
File Edit View Add Format Tools Window
Wave
100 ps
Msgs
/COUNTERTB/CLK
/COUNTERTB/RESET
/COUNTERTB/Q
0
0
0
重置
4 位元二進制上算計數內容
Now
Cursor 1
1000000 ps
2000000 ps
3000000 ps
4000000 ps
5000000 ps
0 ps to 5362169 ps

`include 敘述

基本語法：

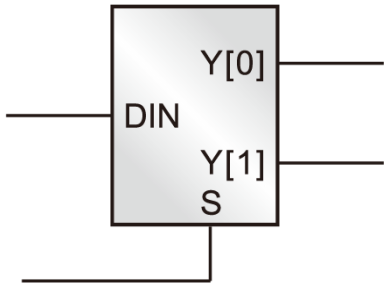
`include “儲存路徑 / 標頭檔案名稱”

TEXT_LIBRARY 目錄內建立如下的元件檔案 (D:/TEXT_LIBRARY)：



PRE_DEFINE.v 的檔案內容：

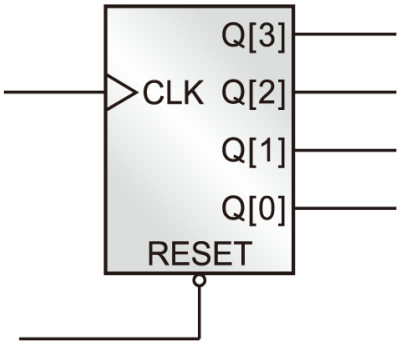
```
 1  /*****************************************
 2   *  define data, component and store in  *
 3   *      D:\TEXE_LIBRARY\PRE_DEFINR.v     *
 4   *          FILENAME : PRE_DEFINE.v      *
 5   *****************************************/
 6
 7   `ifdef WORD
 8     `define LENGTH 16
 9   `else
10     `define LENGTH 8
11   `endif
12
13   `define LAST 4'd9
14
15 module DEMUL1_2(DIN, S, Y);
16
17    input  DIN, S;
18    output [0:1] Y;
19
20    reg [0:1] Y;
21
22    always @(DIN or S)
23      case (S)
24        1'b0    : Y = {DIN, 1'b1};
25        default : Y = {1'b1, DIN};
26      endcase
27
28 endmodule
```



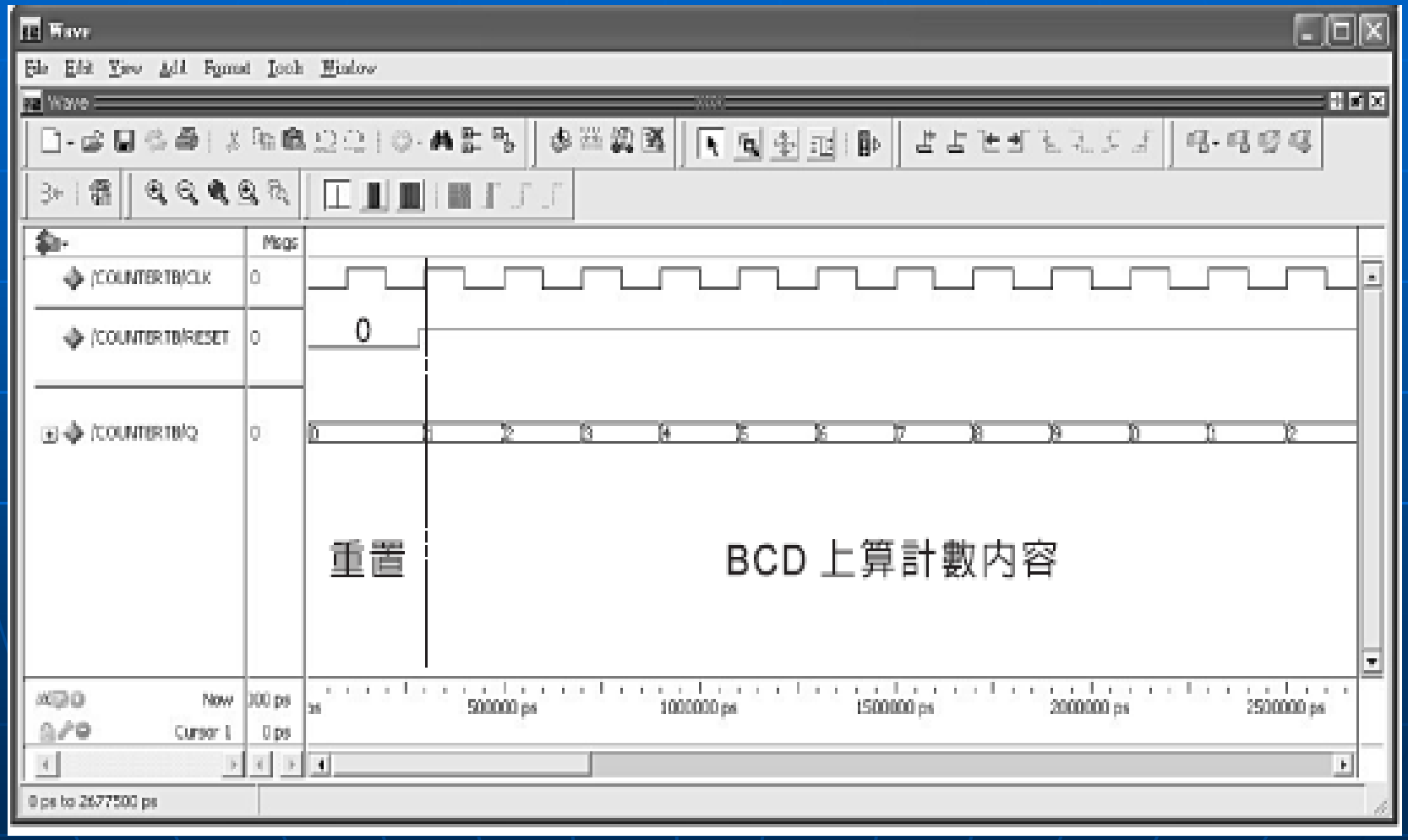
範例一	檔名：UP_COUNTER_1DIG_DEFINE_INCLUDE
以 `include 敘述，配合儲存在 D:/TEXT_LIBRARY/PRE_DEFINE.v 的檔案內容，設計一個功能與前面檔名為 UP_COUNTER_1DIG_DEFINE 相同的電路（從 0 上算到 N 的 BCD 計數器）。	

原始程式(source program) :

```
 1  /*************************
 2   *    1 digital counter   *
 3   *  using define library  *
 4   *  Filename : COUNTER.v  *
 5   ************************* /
 6
 7   `include "D:/TEXT_LIBRARY/PRE_DEFINE.v"
 8
 9  module COUNTER(CLK, RESET, Q);
10
11    input   CLK, RESET;
12    output  [3:0] Q;
13
14    reg [3:0] Q;
15
16    always @(posedge CLK or negedge RESET)
17      if (!RESET)
18        Q <= 4'h0;
19      else
20        if (Q == `LAST)
21          Q <= 4'h0;
22        else
23          Q <= Q + 1'b1;
24
25  endmodule
```



功能模擬 ：

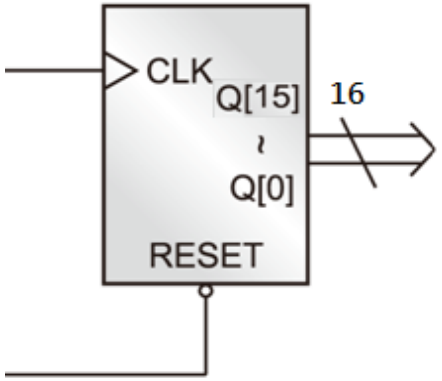

Wave
File Edit View Add Format Tools Window
Wave
Msgs
/COUNTER1B/CLK
/COUNTER1B/RESET
/COUNTER1B/Q
0
0
0
0 1 2 3 4 5 6 7 8 9 10 11 12
重置
BCD 上算計數內容
Now
Cursor 1
500000 ps
1000000 ps
1500000 ps
2000000 ps
2500000 ps
0 ps to 2677500 ps

範例二 檔名：UP_COUNTER_LENGTH_IFDEF_INCLUDE

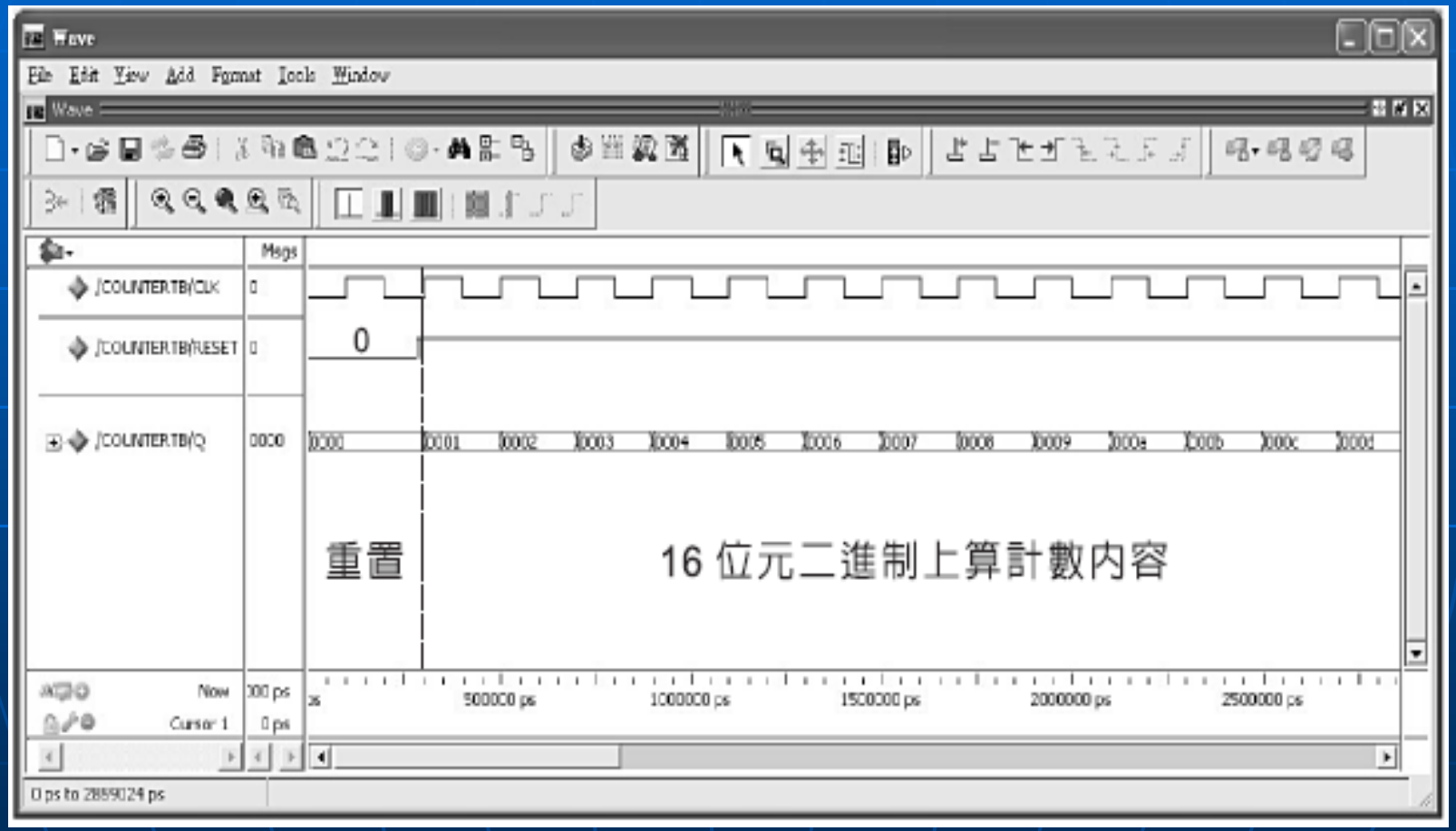
配合儲存在 D:/TEXT_LIBRARY/PRE_DEFINE.v 的檔案內容，說明下面原始程式合成的結果為何？

原始程式(source program) :

```
 1  /************************************
 2   *     1 digital counter using      *
 3   *      library define, ifdef       *
 4   *      Filename : COUNTER.v        *
 5   ************************************/
 6
 7  `define WORD
 8
 9  `include "D:/TEXT_LIBRARY/PRE_DEFINE.v"
10
11  module COUNTER(CLK, RESET, Q);
12
13    input  CLK, RESET;
14    output [`LENGTH-1:0] Q;
15
16    reg [`LENGTH-1:0] Q;
17
18    always @(posedge CLK or negedge RESET)
19      if (!RESET)
20        Q <= 0;
21      else
22        Q <= Q + 1'b1;
23
24  endmodule
```

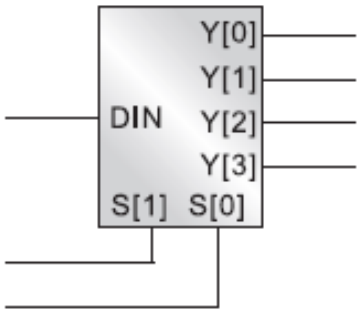


功能模擬 ：

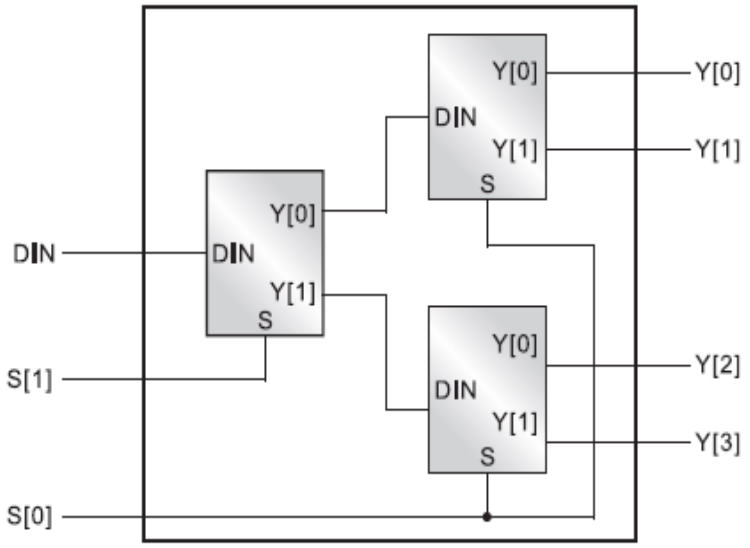

Wave
File Edit View Add Format Tools Window
Wave
Msgs
/COUNTER_TB/CLK
/COUNTER_TB/RESET
/COUNTER_TB/Q
0
0000
0001 0002 0003 0004 0005 0006 0007 0008 0009 000a 000b 000c 000d
重置
16 位元二進制上算計數內容
Now
Cursor 1
500000 ps
1000000 ps
1500000 ps
2000000 ps
2500000 ps
0 ps to 2859024 ps

範例三	檔名：DEMULTIPLEXER1_4_POSITION_INCLUDE
以`include 敘述，連續叫用前面我們所建立儲存在“D:/TEXT_LIBRARY”底下“PRE_DEFINE.v”檔案內的 1 對 2 解多工器元件 DEMUL1_2 (行號 15～28) 3 次，並以位置對應的連線方式完成一個 1 對 4 的解多工器電路。	

1. 方塊圖：

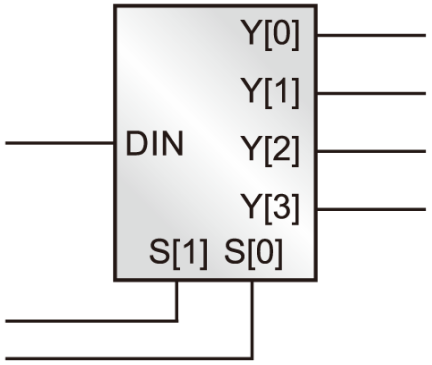


2. 結構圖：

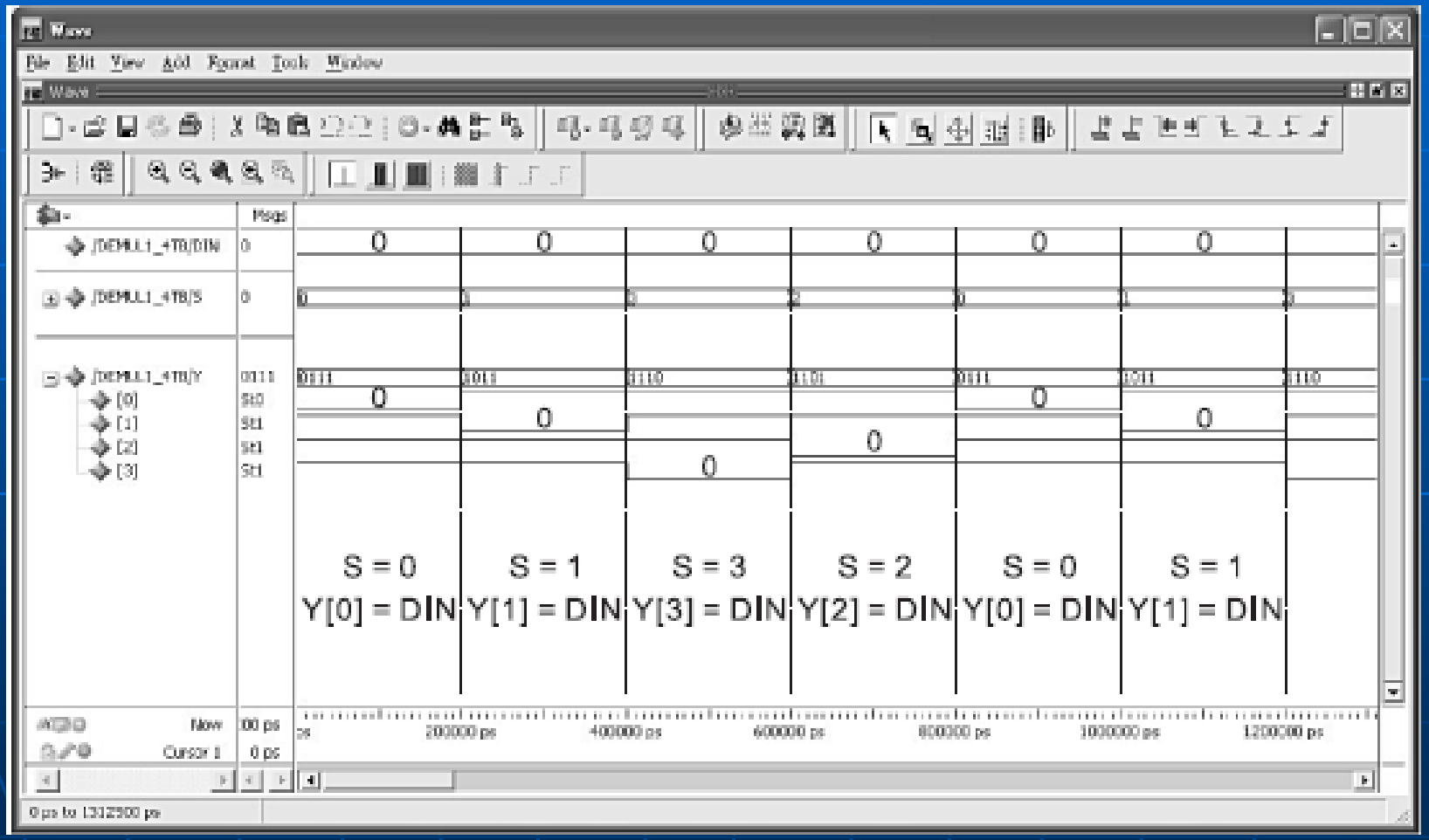


原始程式 (source program) :

```
 1  /*********************************
 2   *   1 to 4 demultiplexer using   *
 3   *  include library by position   *
 4   *      Filename : DEMUL1_4.v     *
 5   *********************************/
 6
 7  `include "D:/TEXT_LIBRARY/PRE_DEFINE.v"
 8
 9  module DEMUL1_4(DIN, S, Y);
10
11    input  DIN;
12    input  [1:0] S;
13    output [0:3] Y;
14
15    wire [0:1] X;
16
17    DEMUL1_2 A1 (DIN, S[1], X);
18    DEMUL1_2 A2 (X[0], S[0], Y[0:1]);
19    DEMUL1_2 A3 (X[1], S[0], Y[2:3]);
20
21  endmodule
```



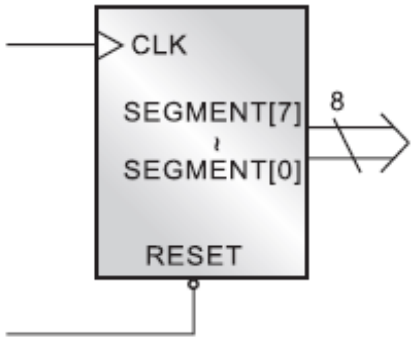
功能模擬 ：


Wave
File Edit View Add Format Tools Window
/DEMUL1_4TB/DIN
/DEMUL1_4TB/S
/DEMUL1_4TB/Y
[0]
[1]
[2]
[3]
S = 0
Y[0] = DIN
S = 1
Y[1] = DIN
S = 3
Y[3] = DIN
S = 2
Y[2] = DIN
S = 0
Y[0] = DIN
S = 1
Y[1] = DIN

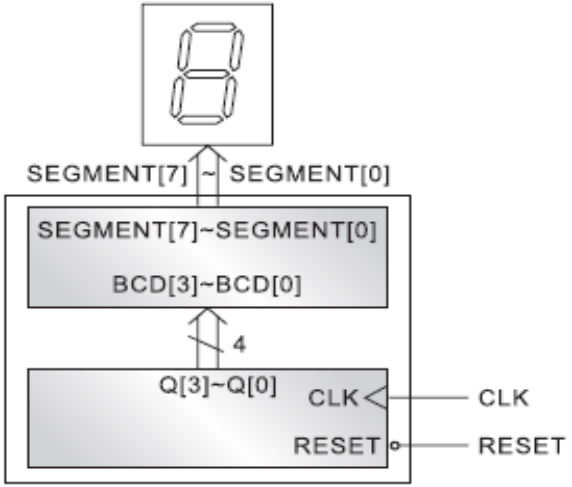
範例四	檔名：BCD_SEVEN_SEGMENT_1DIG_NAME_INCLUDE

以 `include 敘述，叫用前面我們所建立儲存在“D:/TEXT_LIBRARY”底下的兩個元件“COUNTER.v”與“BCD_SEVEN.v”，並以名稱對應的連線方式進行連線，以便完成一個帶有共陽極七段解碼器的 1 位數 BCD 上算計數器。

1. 方塊圖：

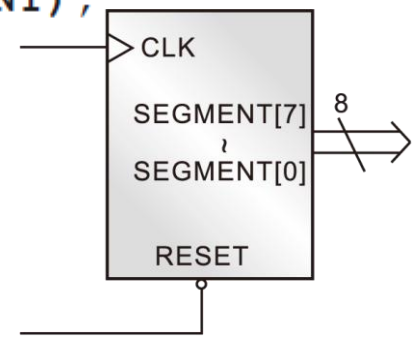


2. 結構圖：

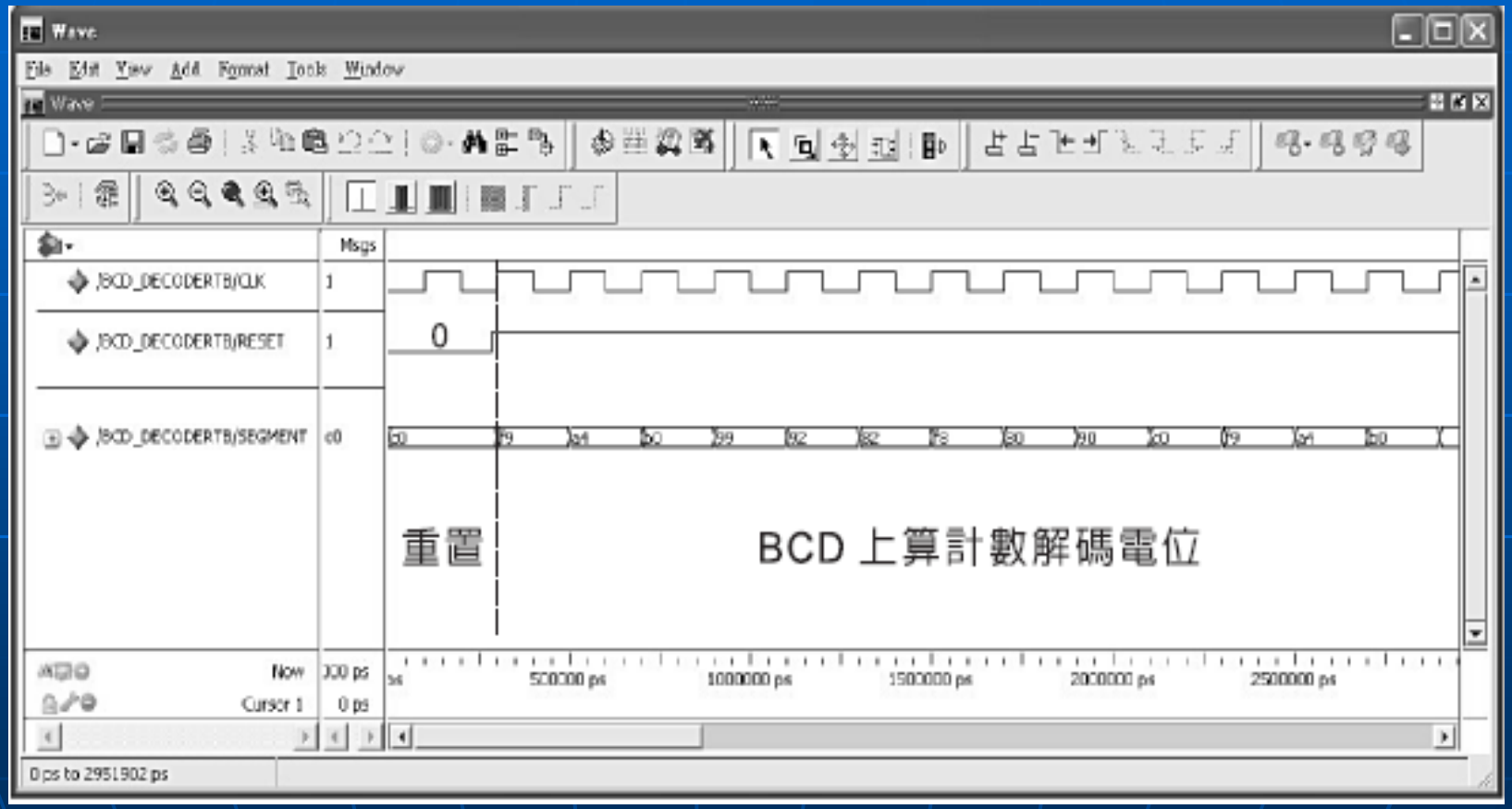


原始程式(source program)：

```
 1  /**********************************
 2   *  1 dig up counter and decoder  *
 3   * using include library by name *
 4   *     Filename : BCD_DECODER.v   *
 5   **********************************/
 6
 7   `include "D:/TEXT_LIBRARY/COUNTER.v"
 8   `include "D:/TEXT_LIBRARY/BCD_SEVEN.v"
 9
10  module BCD_DECODER(CLK, RESET, SEGMENT);
11
12    input  CLK;
13    input  RESET;
14    output [7:0] SEGMENT;
15
16    wire [3:0] BCD;
17
18    COUNTER    A1 (.CLK(CLK), RESET(RESET), Q(BCD));
19    BCD_SEVEN  A2 (.BCD(BCD), SEGMENT(SEGMENT));
20
21  endmodule
```



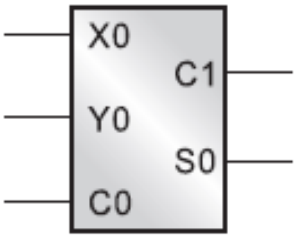
功能模擬 ：


Wave
File Edit View Add Format Tools Window
Msgs
/BCD_DECODERTB/CLK
/BCD_DECODERTB/RESET
/BCD_DECODERTB/SEGMENT
0
重置
BCD 上算計數解碼電位
Now
Cursor 1
500000 ps
1000000 ps
1500000 ps
2000000 ps
2500000 ps
0 ps to 2951902 ps

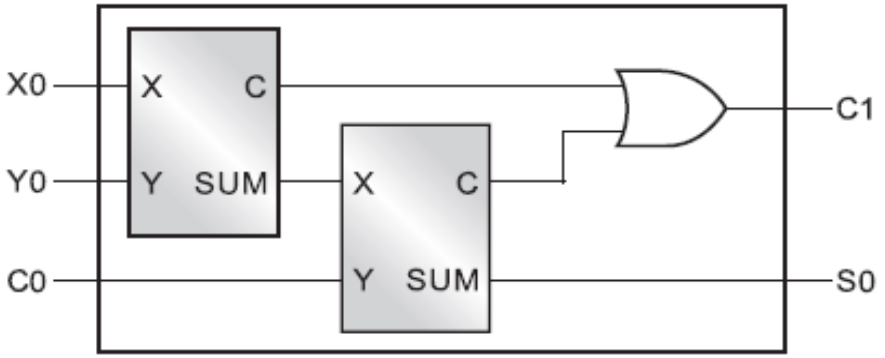
範例五	檔名：FULL_ADDER_POSITION_INCLUDE

以 `include 敘述，叫用我們所建立儲存在“D:/TEXT_LIBRARY”底下的兩個元件“HALF_ADDER.v”與“OR_GATE.v”，並以位置對應的連線方式進行連線，以便完成一個 1 位元的全加器電路。

1. 方塊圖：

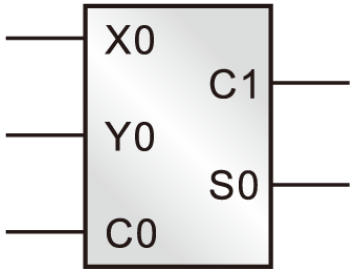


2. 結構圖：

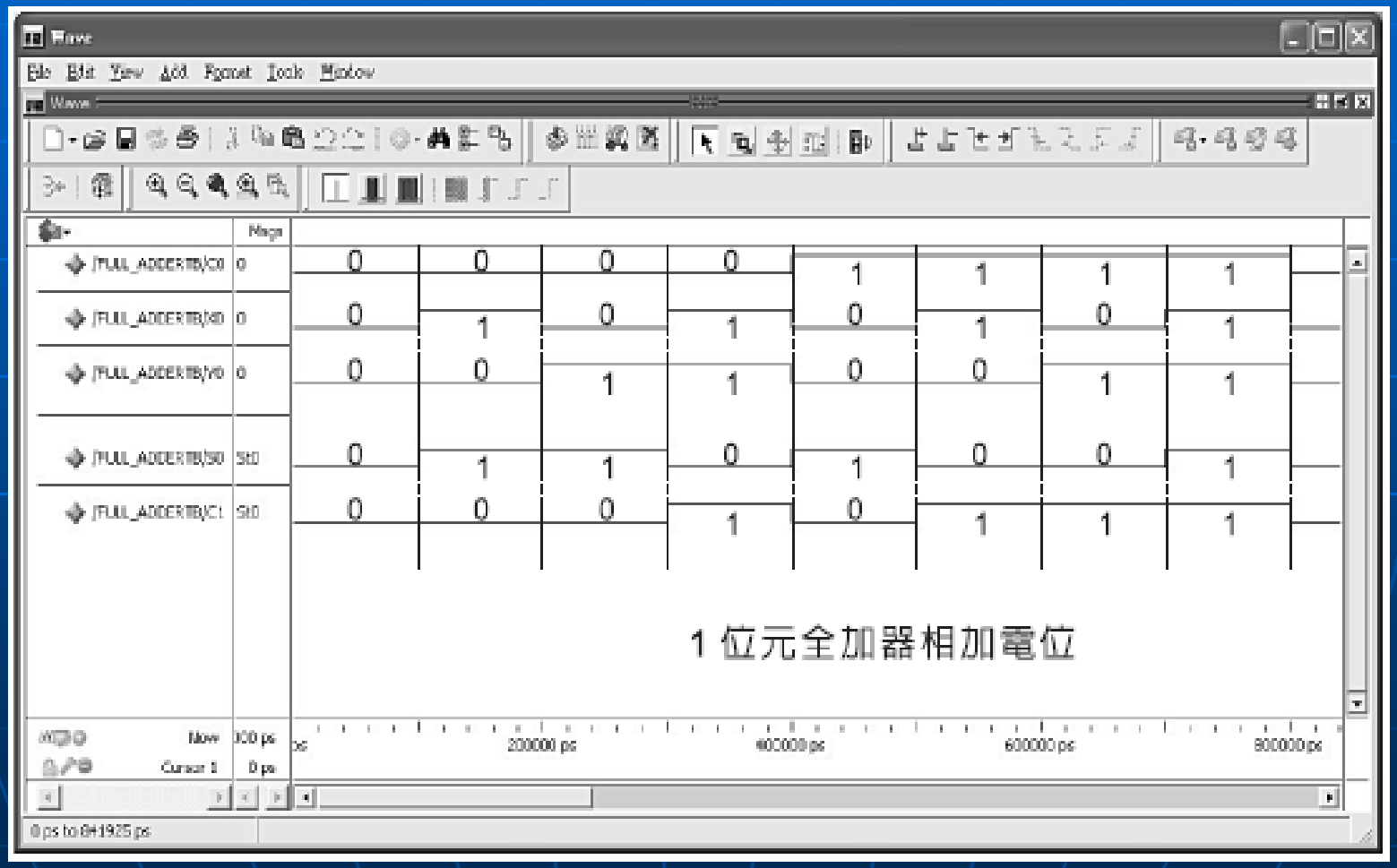


原始程式(source program) :

```
 1  /**********************************
 2   *     1 bit full adder using     *
 3   *  include library by position   *
 4   *     Filename : FULL_ADDER.v    *
 5   **********************************/
 6
 7  `include "D:/TEXT_LIBRARY/HALF_ADDER.v"
 8  `include "D:/TEXT_LIBRARY/OR_GATE.v"
 9
10  module FULL_ADDER(X0, Y0, C0, S0, C1);
11
12    input  X0, Y0, C0;
13    output S0, C1;
14
15    wire CT, ST, CA;
16
17    HALF_ADDER A1 (X0, Y0, CT, ST);
18    HALF_ADDER A2 (ST, C0, CA, S0);
19    OR_GATE    A3 (CT, CA, C1);
20
21  endmodule
```



功能模擬 ：


Wave
File Edit View Add Format Tools Window
Wave
Msgs
/FULL_ADDER_TB/C0
/FULL_ADDER_TB/X0
/FULL_ADDER_TB/Y0
/FULL_ADDER_TB/S0
/FULL_ADDER_TB/C1
Now
Cursor 1
200000 ps
400000 ps
600000 ps
800000 ps
0 ps to 841925 ps
1 位元全加器相加電位

Moore 與 Mealy 狀態機

Moore Machine：

狀態機的輸出電位只與目前所記錄的狀態有關，與輸入訊號無立即的關係。

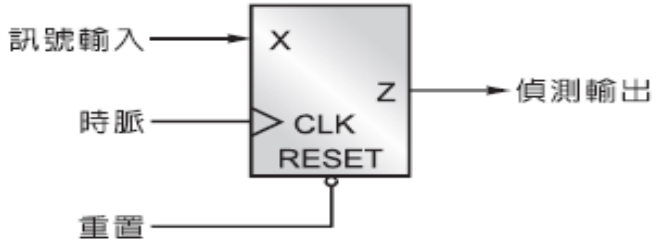
Mealy Machine：

狀態機的輸出電位不只與目前所記錄的狀態有關，而且與輸入訊號有立即的關係。

Moore 有限狀態機器

範例	檔名：MOORE_MACHINE_DETECT_101
以 Moore Machine 設計一個從一連串的輸入訊號 X 中，偵測出連續的 101 訊號。	

方塊圖：



動作狀況：

從 X 端輸入一連串 0 與 1 的訊號中：

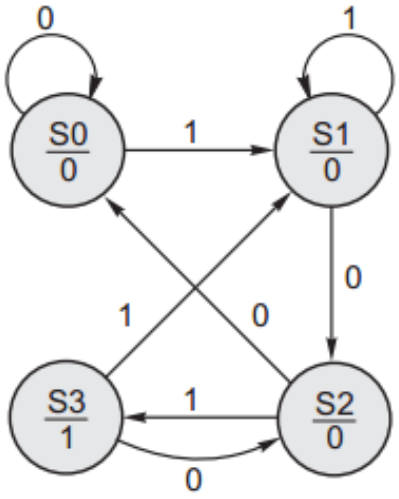
偵測到 101 時，輸出端 Z 為高電位 1。

沒有偵測到 101 時，輸出端 Z 為低電位 0。

訊號輸入 X：11001011001010100

偵測輸出 Z：000000**1**00000**1**0**1**0**1**00

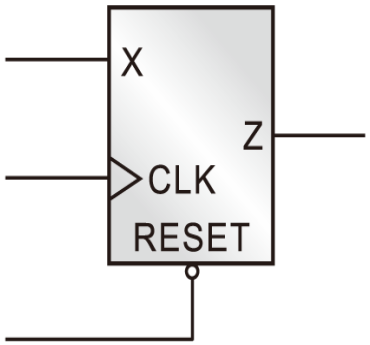
<步驟一> 繪出電路的狀態圖 (圖形說明請參閱本人所寫的“最新數位邏輯電路設計”一書)：



<步驟二> 整理出狀態表：

目前狀態 PS	下一個狀態		輸出 Z
	X = 0	X = 1	
S0	S0	S1	0
S1	S2	S1	0
S2	S0	S3	0
S3	S2	S1	1

原始程式 (source program) :

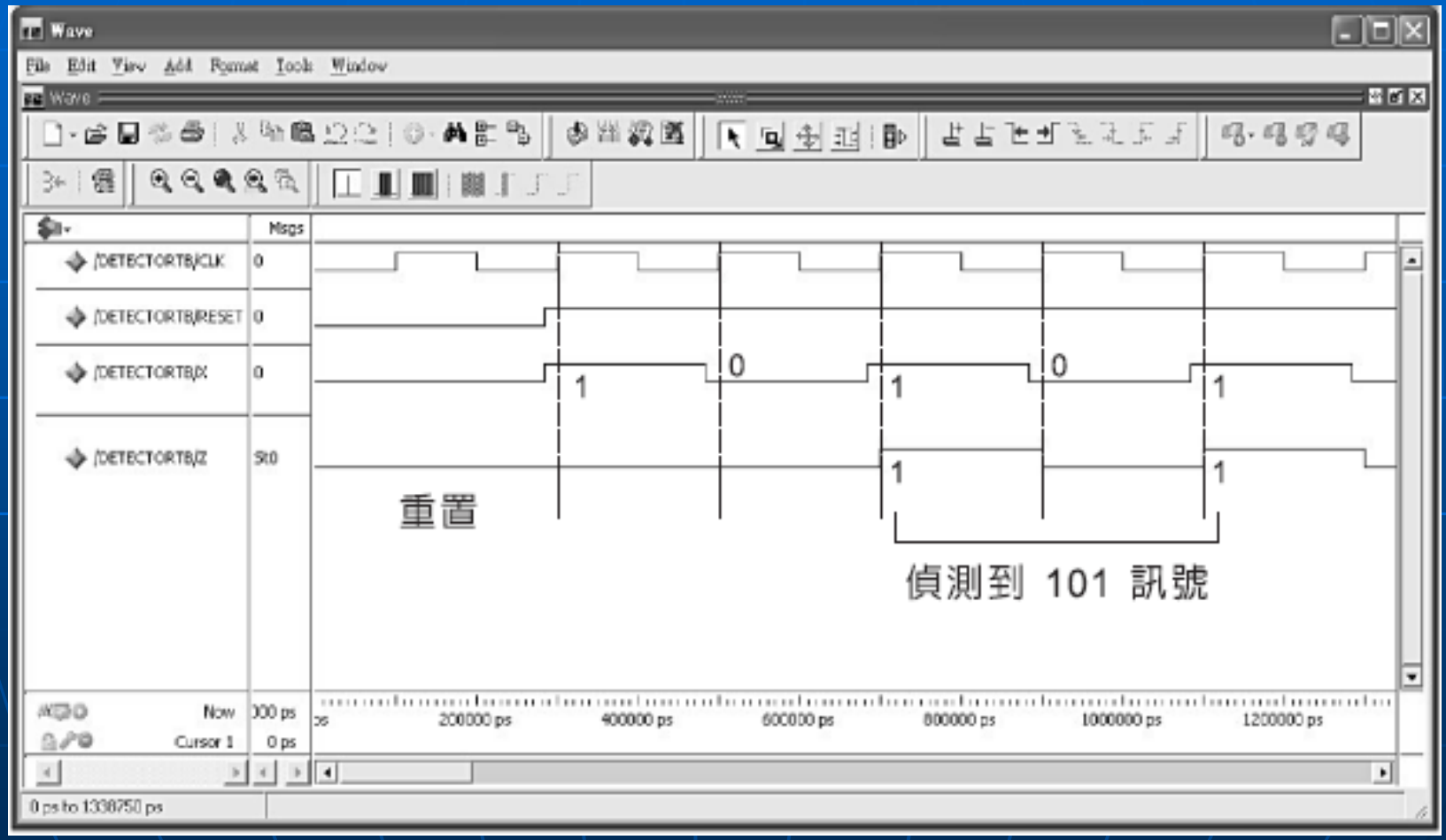



```
 1  /*********************************
 2   *    detect "101" from series    *
 3   *   input signal (moor machine)  *
 4   *       Filename : DETECTOR.v    *
 5   *********************************/
 6
 7  module DETECTOR(CLK, RESET, X, Z);
 8    input  CLK, RESET, X;
 9    output Z;
10
11    parameter  S0 = 2'b00, S1 = 2'b01,
12               S2 = 2'b10, S3 = 2'b11 ;
13
14    reg Z;
15    reg [1:0] present_state, next_state;
16
17    always @(posedge CLK or negedge RESET)
18      begin
19        if (!RESET)
20          present_state <= S0;
21        else
22          present_state <= next_state;
23      end
24
```

```
25    always @(present_state or X)
26      begin
27        case(present_state)
28          S0 :
29            begin
30              if (X)
31                next_state <= S1;
32              else
33                next_state <= S0;
34              Z <= 1'b0;
35            end
36          S1 :
37            begin
38              if (X)
39                next_state <= S1;
40              else
41                next_state <= S2;
42              Z <= 1'b0;
43            end
```

```
44          S2 :
45            begin
46              if (X)
47                next_state <= S3;
48              else
49                next_state <= S0;
50              Z <= 1'b0;
51            end
52          default :
53            begin
54              if (X)
55                next_state <= S1;
56              else
57                next_state <= S2;
58              Z <= 1'b1;
59            end
60        endcase
61    end
62
63  endmodule
```

功能模擬 ：

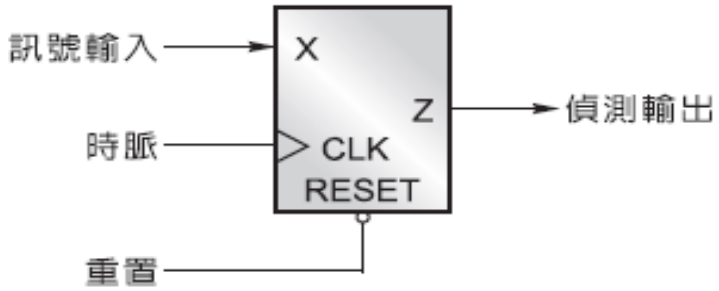

Wave
File Edit View Add Format Tools Window
Msgs
/DETECTORTB/CLK 0
/DETECTORTB/RESET 0
/DETECTORTB/X 0
/DETECTORTB/Z St0
1
0
1
0
1
1
1
重置
偵測到 101 訊號
Now 300 ps
Cursor 1 0 ps
200000 ps
400000 ps
600000 ps
800000 ps
1000000 ps
1200000 ps
0 ps to 1338750 ps

Mealy有限狀態機器

範例	檔名：MEALY_MACHINE_DETECT_101

以 Mealy Machine 設計一個，從一連串的輸入訊號 X 中，偵測出連續的 101 訊號。(讀者可以將它與前面範例二做個比較，就可以知道 Moore 與 Mealy Machine 之間的差別)

方塊圖：



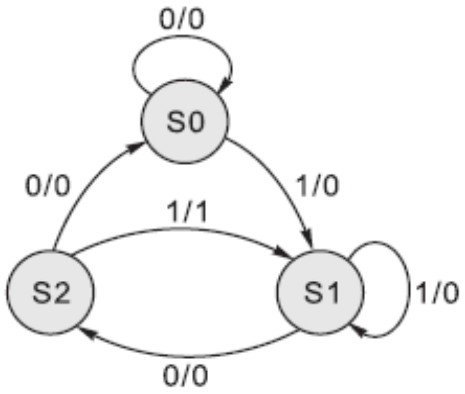
動作狀況：

從 X 端輸入一連串 0 與 1 的訊號中：

偵測到 101 時，輸出端 Z 為高電位 1。

沒有偵測到 101 時，輸出端 Z 為低電位 0。

<步驟一> 繪出電路的狀態圖 (圖形說明請參閱本人所寫的“最新數位邏輯電路設計”一書)：

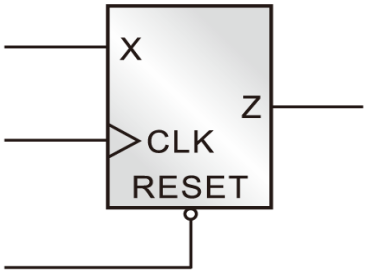


<步驟二> 整理出狀態表：

目前狀態 PS	下一個狀態	
	X = 0	X = 1
S0	S0，0	S1，0
S1	S2，0	S1，0
S2	S0，0	S1，1

原始程式(source program) :

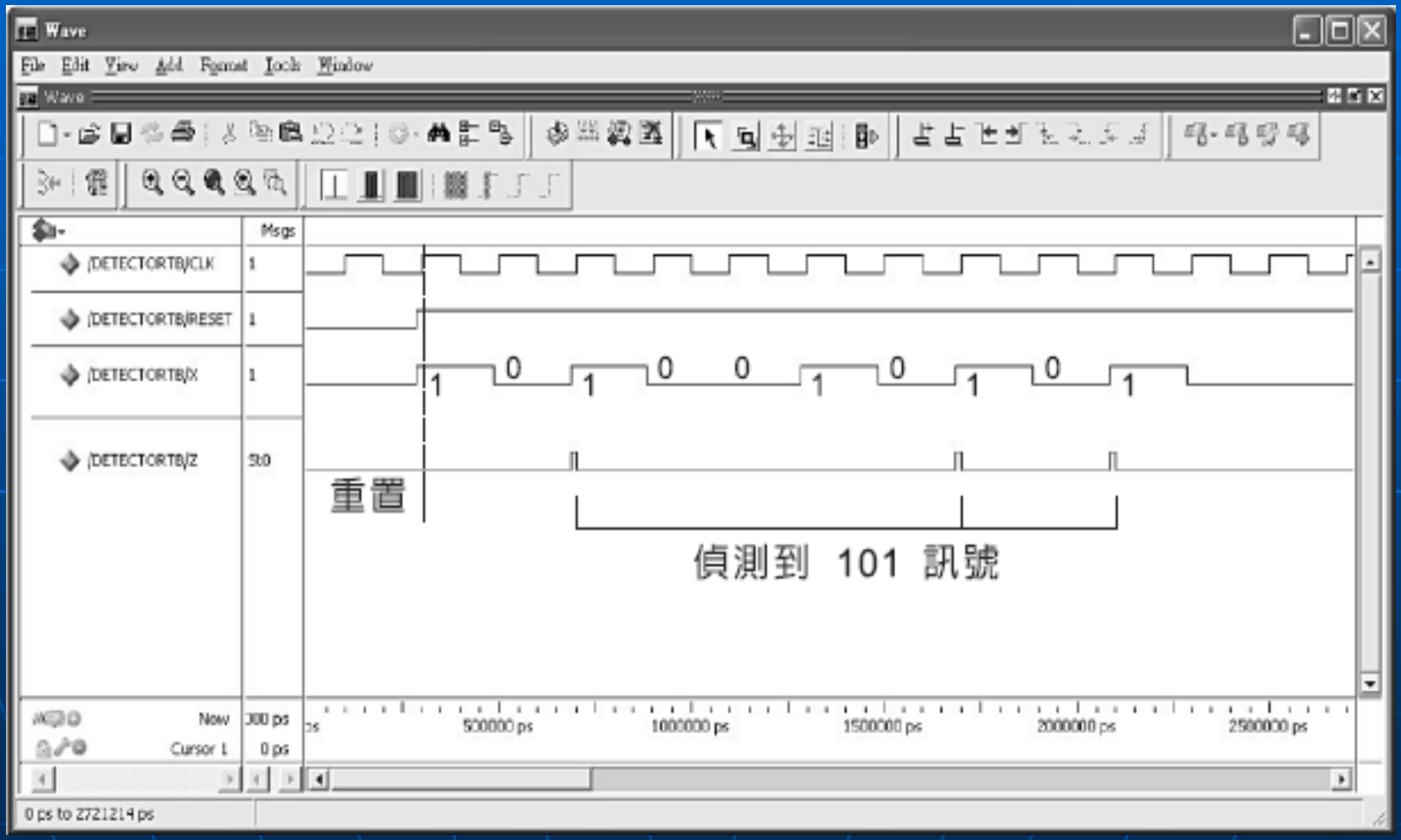
```
 1  /*********************************
 2   *    detect "101" from series    *
 3   *  input signal (mealy machine) *
 4   *      Filename : DETECTOR.v     *
 5   *********************************/
 6
 7  module DETECTOR(CLK, RESET, X, Z);
 8
 9    input   CLK, RESET, X;
10    output  Z;
11
12    parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
13
14    reg Z;
15    reg [1:0] present_state, next_state;
16
17    always @(posedge CLK or negedge RESET)
18      begin
19        if (!RESET)
20          present_state <= S0;
21        else
22          present_state <= next_state;
23      end
24
```



```
25    always @(present_state or X)
26      begin
27        case(present_state)
28          S0 :
29            begin
30              if (X)
31                begin
32                  next_state <= S1;
33                  Z <= 1'b0;
34                end
35              else
36                begin
37                  next_state <= S0;
38                  Z <= 1'b0;
39                end
40            end
```

```
41          S1 :
42            begin
43              if (X)
44                begin
45                  next_state <= S1;
46                  Z <= 1'b0;
47                end
48              else
49                begin
50                  next_state <= S2;
51                  Z <= 1'b0;
52                end
53            end
54         default :
55            begin
56              if (X)
57                begin
58                  next_state <= S1;
59                  Z <= 1'b1;
60                end
61              else
62                begin
63                  next_state <= S0;
64                  Z <= 1'b0;
65                end
66            end
67        endcase
68    end
69
70  endmodule
```

功能模擬 ：


Wave
File Edit View Add Format Tools Window
Msgs
/DETECTORTB/CLK
/DETECTORTB/RESET
/DETECTORTB/X
/DETECTORTB/Z
1
1
1
0
1
0
0
1
0
1
0
1
重置
偵測到 101 訊號
Now
Cursor 1
500000 ps
1000000 ps
1500000 ps
2000000 ps
2500000 ps
0 ps to 2721214 ps