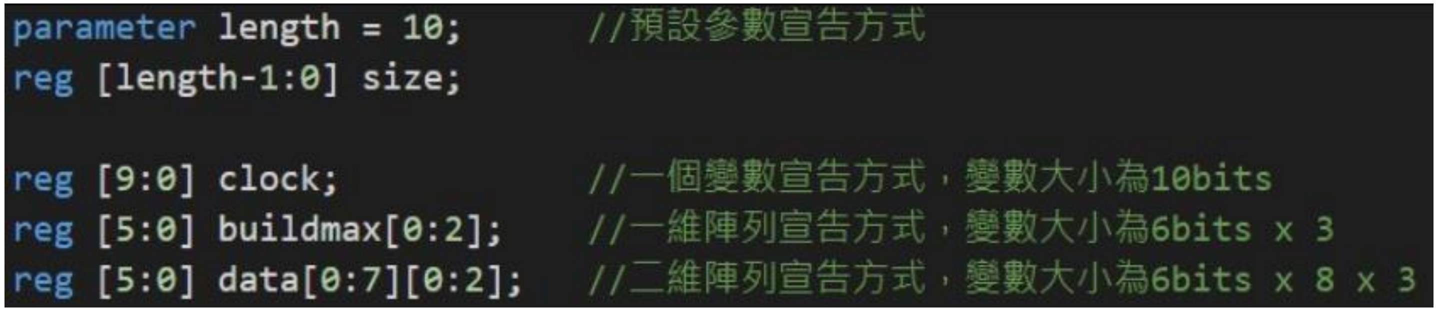
數位系統導論實驗 LAB 6

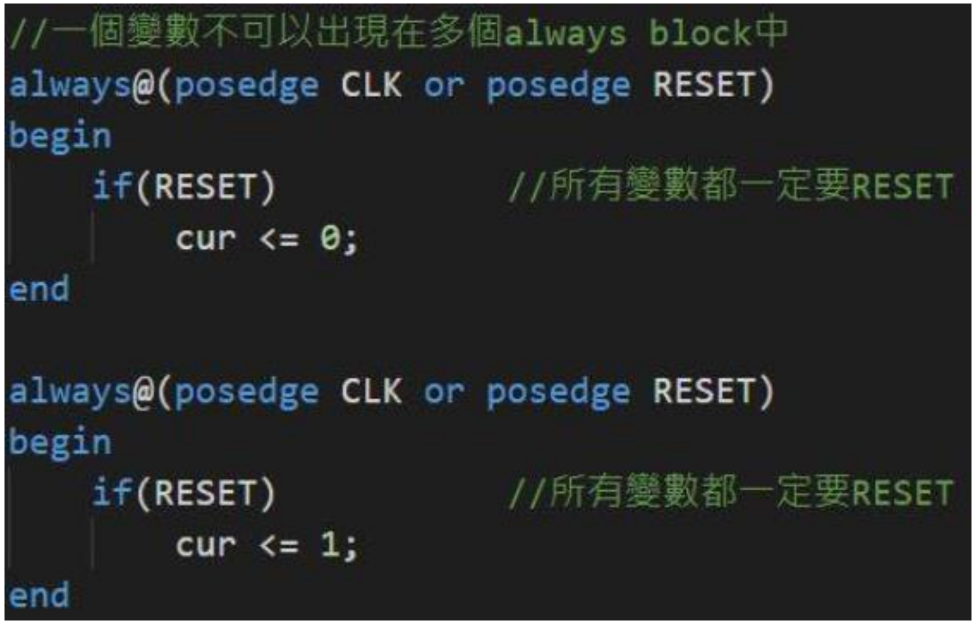
梁郁珮

2022/04/25
2022/04/27

Verilog coding Styles(1/3)



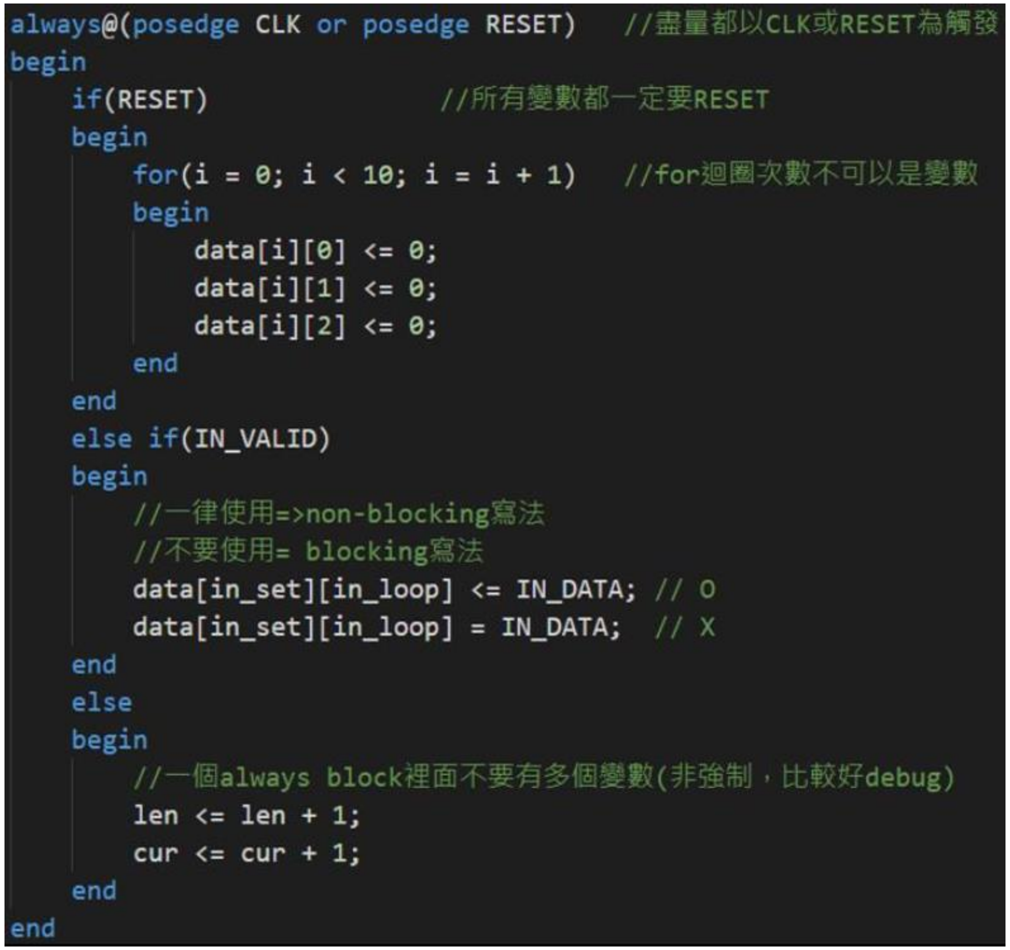
```
parameter length = 10;        //預設參數宣告方式
reg [length-1:0] size;

reg [9:0] clock;              //一個變數宣告方式，變數大小為10bits
reg [5:0] buildmax[0:2];      //一維陣列宣告方式，變數大小為6bits x 3
reg [5:0] data[0:7][0:2];     //二維陣列宣告方式，變數大小為6bits x 8 x 3
```



```
//一個變數不可以出現在多個always block中
always@(posedge CLK or posedge RESET)
begin
    if(RESET)              //所有變數都一定要RESET
        cur <= 0;
end

always@(posedge CLK or posedge RESET)
begin
    if(RESET)              //所有變數都一定要RESET
        cur <= 1;
end
```

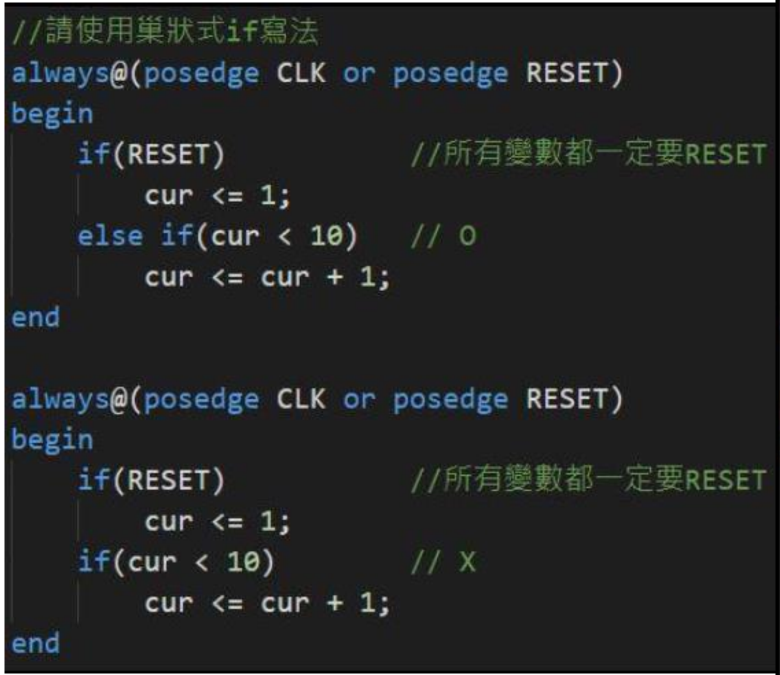
國立中正大學 National Chung Cheng University

Verilog coding Styles(2/3)

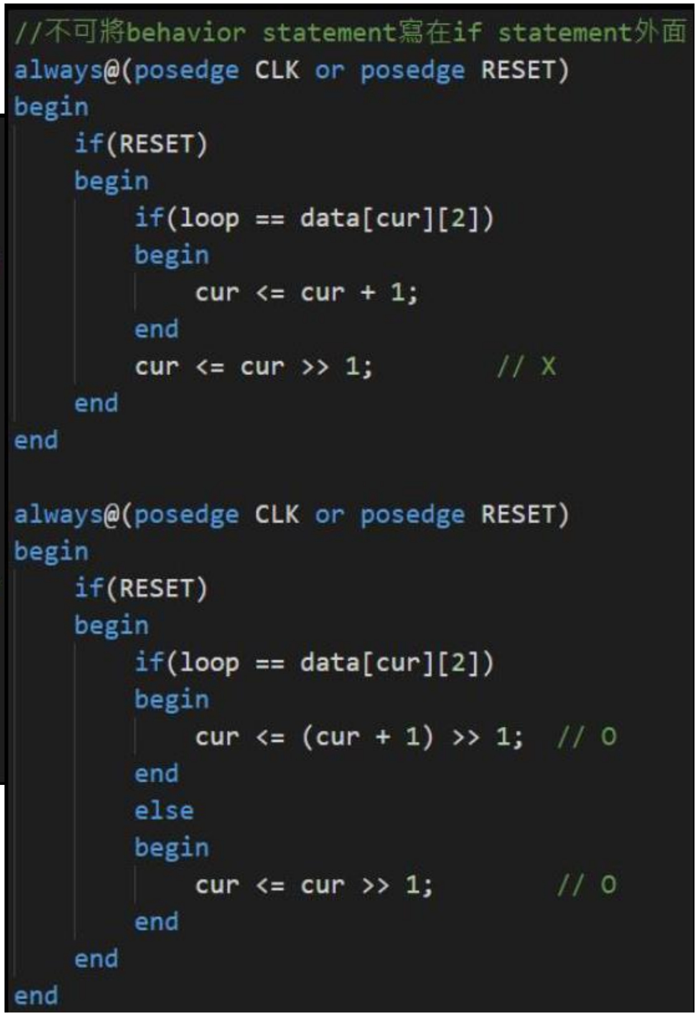


```
always@(posedge CLK or posedge RESET)   //盡量都以CLK或RESET為觸發
begin
    if(RESET)                  //所有變數都一定要RESET
    begin
        for(i = 0; i < 10; i = i + 1)   //for迴圈次數不可以是變數
        begin
            data[i][0] <= 0;
            data[i][1] <= 0;
            data[i][2] <= 0;
        end
    end
    else if(IN_VALID)
    begin
        //一律使用=>non-blocking寫法
        //不要使用= blocking寫法
        data[in_set][in_loop] <= IN_DATA; // O
        data[in_set][in_loop] = IN_DATA;  // X
    end
    else
    begin
        //一個always block裡面不要有多個變數(非強制，比較好debug)
        len <= len + 1;
        cur <= cur + 1;
    end
end
```

Verilog coding Styles(3/3)



```verilog
//請使用巢狀式if寫法
always@(posedge CLK or posedge RESET)
begin
    if(RESET)           //所有變數都一定要RESET
        cur <= 1;
    else if(cur < 10)   // O
        cur <= cur + 1;
end

always@(posedge CLK or posedge RESET)
begin
    if(RESET)           //所有變數都一定要RESET
        cur <= 1;
    if(cur < 10)        // X
        cur <= cur + 1;
end
```



```verilog
//不可將behavior statement寫在if statement外面
always@(posedge CLK or posedge RESET)
begin
    if(RESET)
    begin
        if(loop == data[cur][2])
        begin
            cur <= cur + 1;
        end
        cur <= cur >> 1;        // X
    end
end

always@(posedge CLK or posedge RESET)
begin
    if(RESET)
    begin
        if(loop == data[cur][2])
        begin
            cur <= (cur + 1) >> 1;  // O
        end
        else
        begin
            cur <= cur >> 1;        // O
        end
    end
end
```

國立中正大學 National Chung Cheng University

Verilog logic

```
reg [3:0]count, temp;
always@(posedge clk)
begin
  if(!rst)              count <= 0;
  else if(counter > 5)  count <= 0;
  else                  count <= count + 1;
end

always@(posedge clk)
begin
  if(!rst)              temp <= 0;
  else if(counter < 5)  temp <= temp + 1;
end
```

```
reg [3:0]count, temp;
always@(posedge clk)
begin
  if(!rst)              count <= 0;
  else if(counter < 5)  count <= count + 1;
end

always@(posedge clk)
begin
  if(!rst)              temp <= 0;
  else                  temp <= temp + 1;
end
```

```
int count = 0;
int temp = 0;
While(true)
{
  count = (count < 5) ? count + 1 : 0;
  temp = (temp <= 15) ? temp + 1 : 0;
}
```

```
int count = 0;
int temp = 0;
While(true)
{
  count = (count < 5) ? count + 1 : count;
  temp = (temp <= 15) ? temp + 1 : 0;
}
```

```
for(count = 0; count < 5; count ++)
{
  temp = (temp <= 15) ? temp + 1 : 0;
}
```

Verilog logic

casez

```
always@(posedge clk)
begin
  casez(<condition>)
    1’b1??? : temp <= 0;
    1’b?1?? : temp <= 1;
    1’b??1? : temp <= 2;
    1’b???1 : temp <= 3;
    default : temp <= temp;
  endcase
end
```

casex

```
always@(posedge clk)
begin
  casex(<condition>)
    1’b1xxx : temp <= 0;
    1’bx1xx : temp <= 1;
    1’bxx1x : temp <= 2;
    1’bxxx1 : temp <= 3;
    default : temp <= temp;
  endcase
end
```

Verilog logic

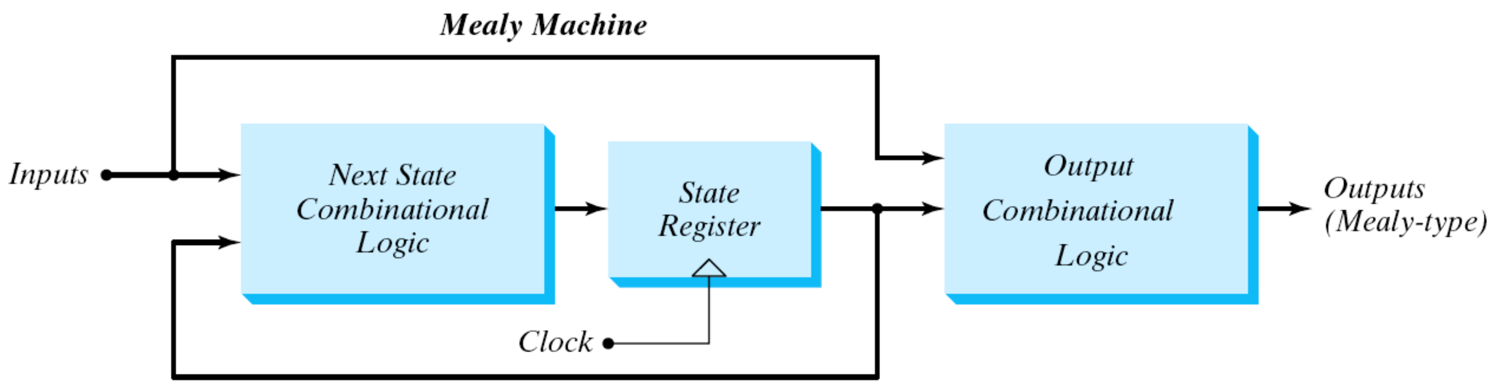
```
temp <= 4’b1001 << 1;         temp : 0010

temp <= 4’sb1001 >> 1;        temp : 0100

temp <= 4’sb1001 >>> 1;       temp : 1100
```

```
A : 4’b1101
B : 4’b0110

C <= {A, B}
C : 8’b1101_0110

C <= {10{1’b1}}
C : 10’b11_1111_1111

C <= {2'b1,{2{A}}}
C : 10’b11_1101_1101
```

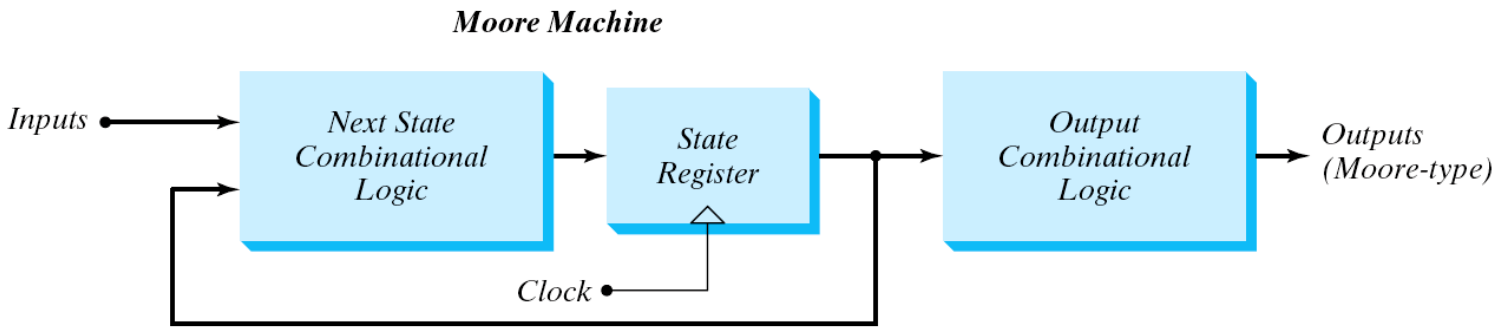
優先順序	
! ~	
* / %	
+ -	
<< >>	最高優先順序別
< <= > >=	↓
== !== === !===	↓
&	↓
^ ^~	↓
|	最低優先順序別
&&	
||	
? :	

National Chung Cheng University

Finite State Machine

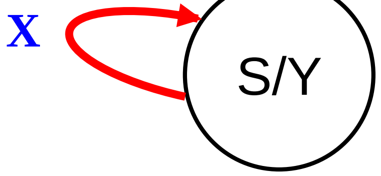


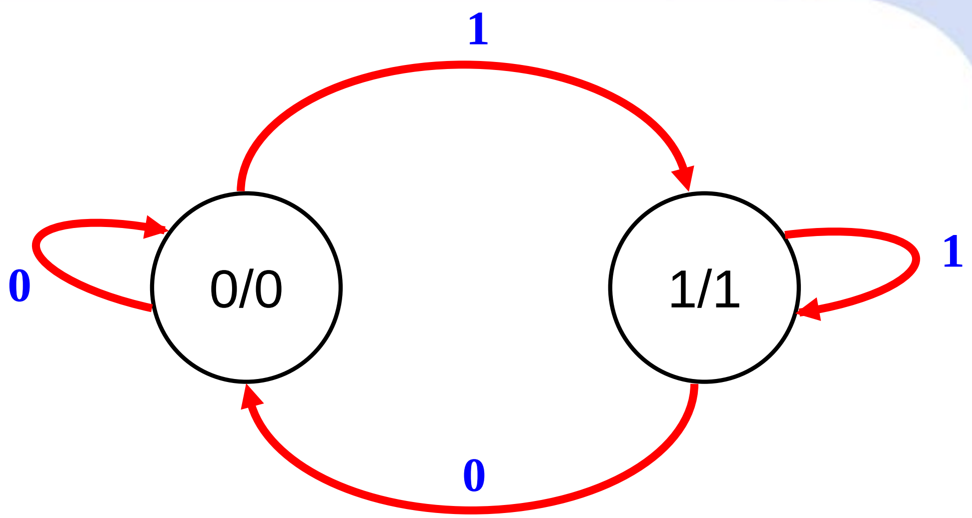
(a)

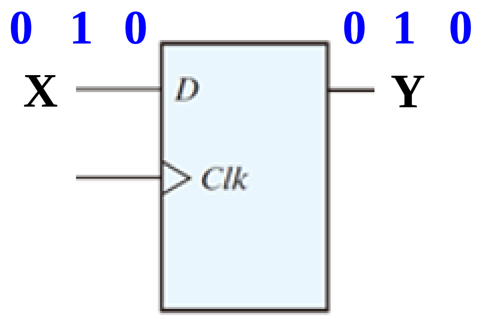


(b)

FSM

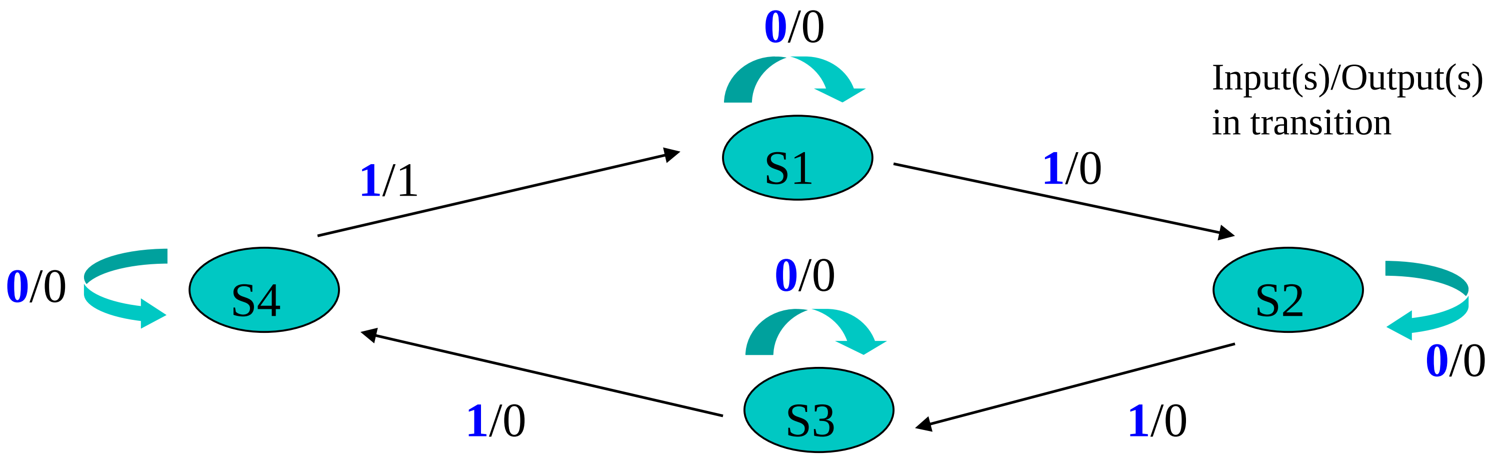






Present State (t)	input	Next State (t+1)	output
A	X	A	Y
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	1

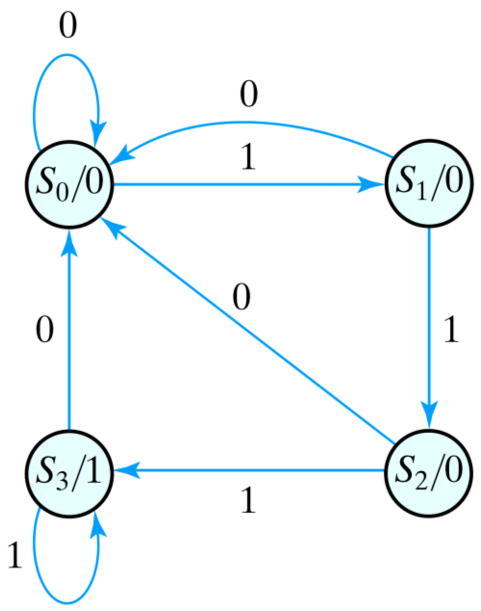
FSM(Mealy.)



Input(s)/Output(s) shown in transition

FSM(Moore.)

Present State	Input x	Next State	Output y
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_0	0
S_1	1	S_2	0
S_2	0	S_0	0
S_2	1	S_3	0
S_3	0	S_0	1
S_3	1	S_3	1


$$\begin{tikzcd}
S_0/0 \arrow[loop above, "0"] \arrow[r, "1"] & S_1/0 \arrow[l, bend right, "0"'] \arrow[d, "1"] \\
S_3/1 \arrow[u, "0"] \arrow[loop below, "1"] & S_2/0 \arrow[l, "1"] \arrow[ul, "0"]
\end{tikzcd}$$

- Present state indicates current value of flip flops
- Next state indicates state after next rising clock edge
- Output is output value on next rising clock edge

實 驗 範 例

component	I/O	size	pin name	instructions
seven_seg	input	1	clk	return 7-segment code from seg_number at posedge clk while rst = 1
	input	1	rst	
	input	4	seg_number	
	output	8	seg_data	
div_clk	input	1	clk	return slow clock (1hz) from input clock (50MHz)
	input	1	rst	
	output	1	clk_1hz	
bin2dec	input	8	A	return 3 BCD code D2 for 10^2, D1 for 10^1 and D0for 10^0, (direct to 7-segment) from 8-bit A
	output	4	D0	
	output	4	D1	
	output	4	D2	

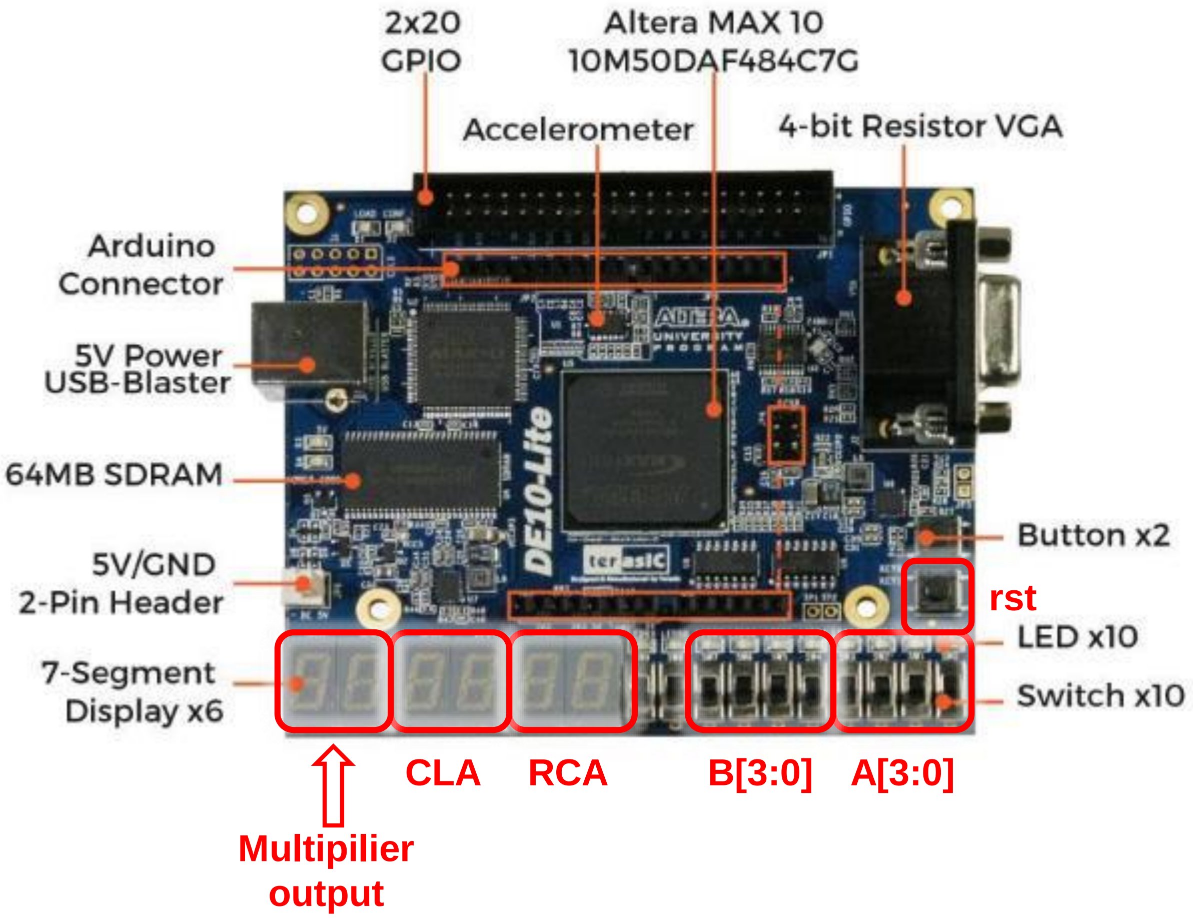
component	I/O	size	pin name	instructions
RCA	input	1	clk	A Ripple Carry Adder
	input	1	rst	
	input	4	A	
	input	4	B	
	input	1	carry	
	output	5	Y	
CLA	input	1	clk	Lab 6-1 Carry Look Ahead
	input	1	rst	
	input	4	A	
	input	4	B	
	input	1	carry	
	output	5	Y	
multiplier	input	1	clk	Lab 6-2 Bitwise multiplier
	input	1	rst	
	input	4	A	
	input	3	B	
	output	8	Y	

國立中正大學 National Chung Cheng University

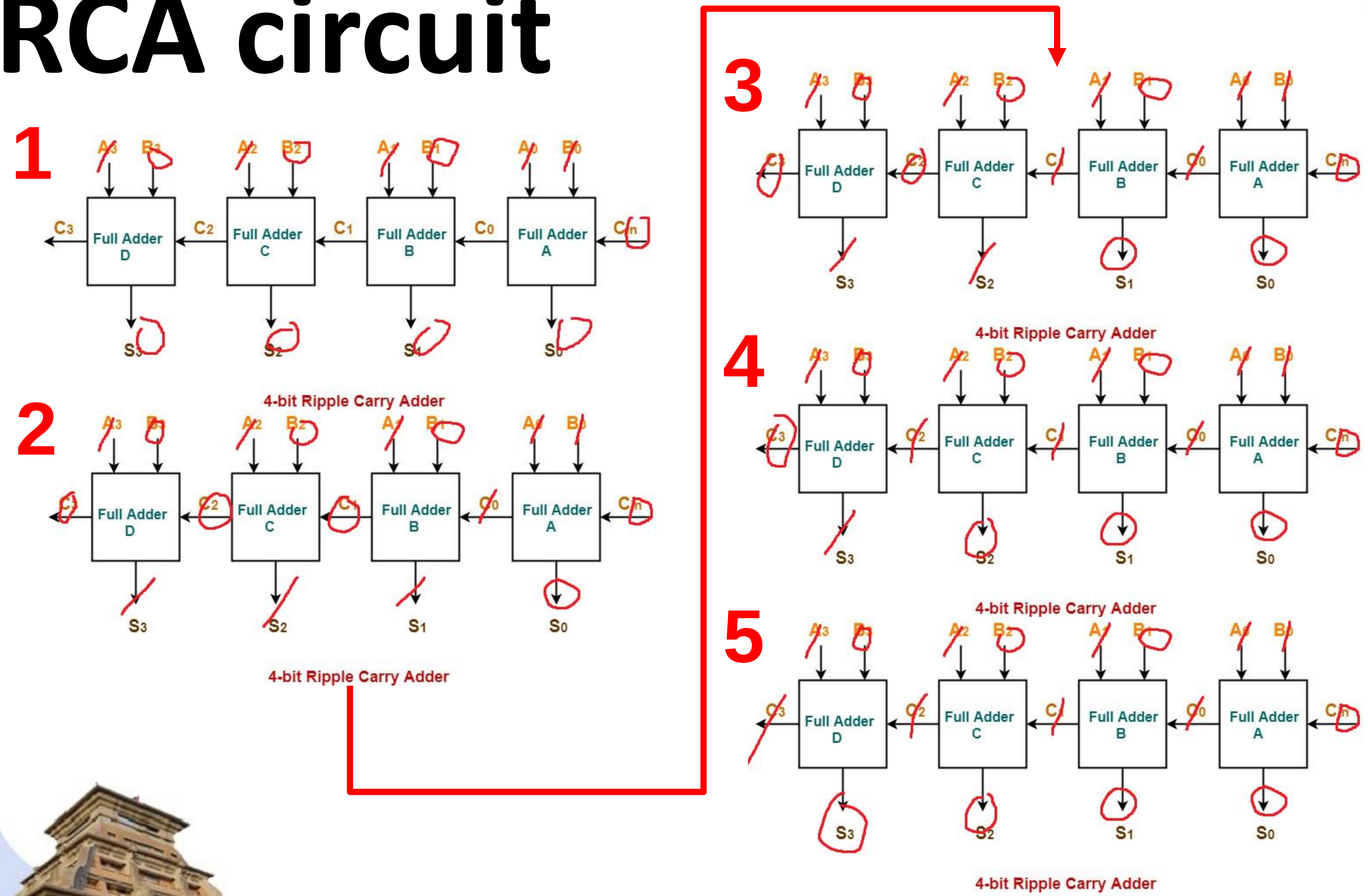
實 驗 項 目

- 6-1. 請參考範例程式碼並使用內建50MHz時脈，實作出CLA的硬體描述語言
- 6-2. 同lab4，請實作4-bit by 3-bit binary multiplier
  (※B[3]為0)
  (※不可以直接寫A×B)

實 驗 項 目


2x20
GPIO
Altera MAX 10
10M50DAF484C7G
Accelerometer
4-bit Resistor VGA
Arduino
Connector
5V Power
USB-Blaster
64MB SDRAM
5V/GND
2-Pin Header
7-Segment
Display x6
Button x2
rst
LED x10
Switch x10
CLA
RCA
B[3:0]
A[3:0]
Multipilier
output

RCA circuit


1
2
3
4
5
Full Adder D
Full Adder C
Full Adder B
Full Adder A
A3
B3
A2
B2
A1
B1
A0
B0
C3
C2
C1
C0
Cin
S3
S2
S1
S0
4-bit Ripple Carry Adder

CLA circuit

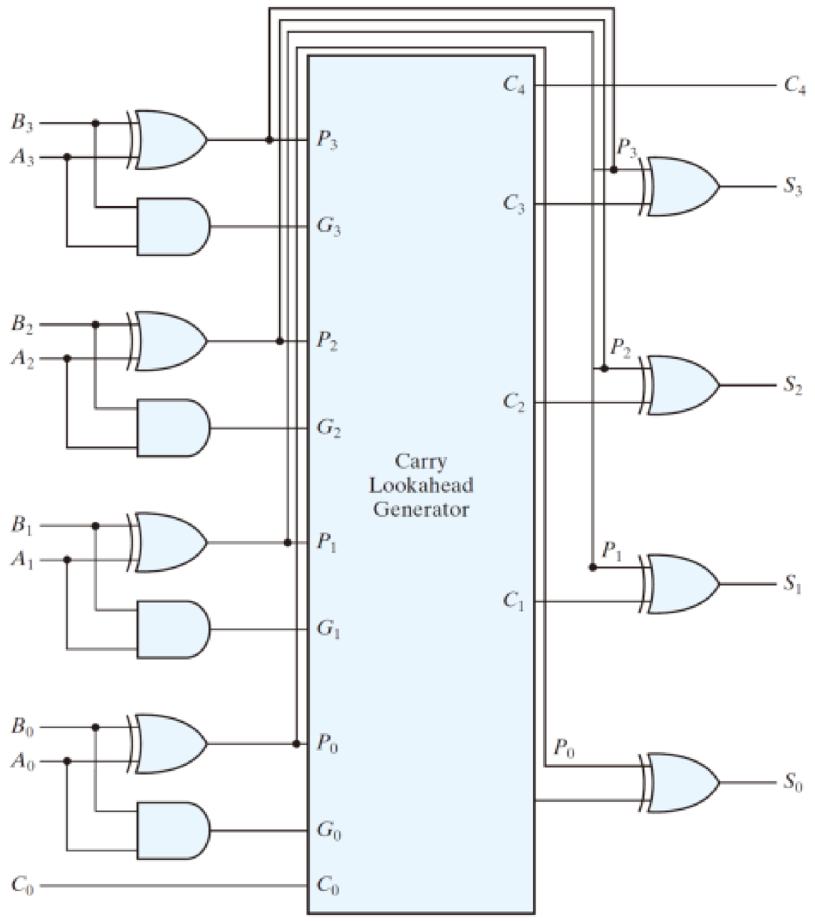
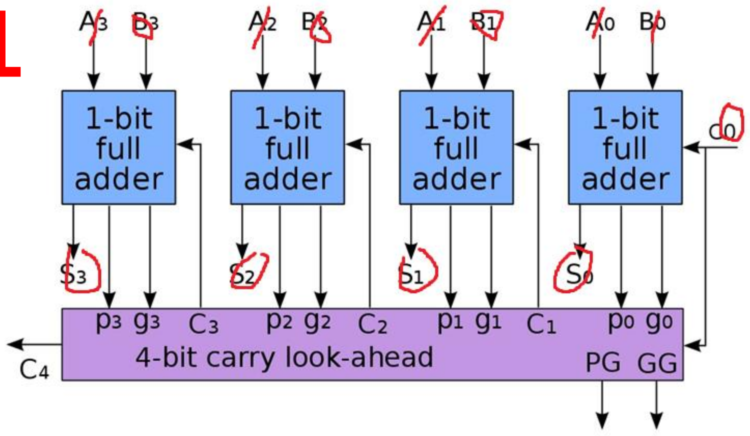


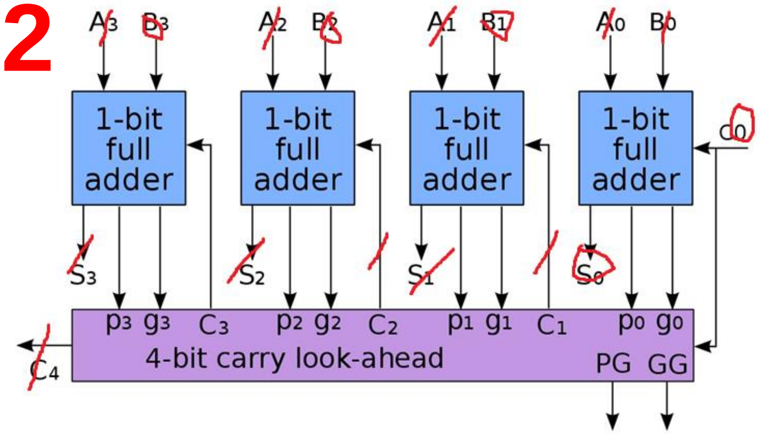
FIGURE 4.12
Four-bit adder with carry lookahead

CLA circuit

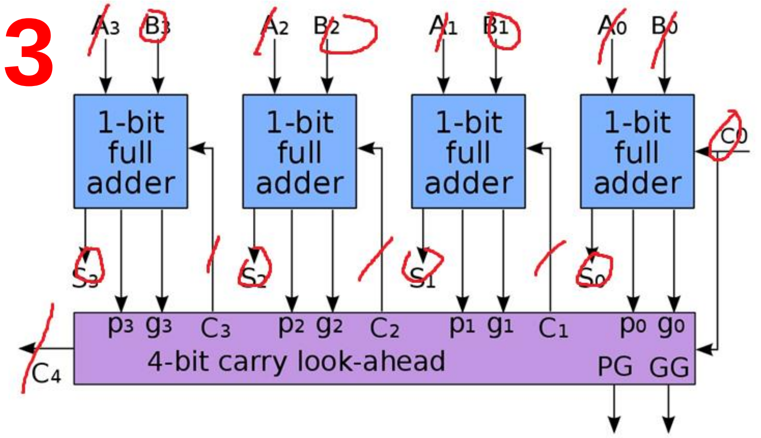
1



2



3



Binary multiplier

4-bit by 3-bit binary multiplier

$$\begin{array}{rrrrrr}
 & & B3 & B2 & B1 & B0 \\
 & & X) & A2 & A1 & A0 \\
\hline
 & & A_0B_3 & A_0B_2 & A_0B_1 & A_0B_0 \\
 & A_1B_3 & A_1B_2 & A_1B_1 & A_1B_0 & \\
A_2B_3 & A_2B_2 & A_2B_1 & A_2B_0 & & \\
\hline
\end{array}$$

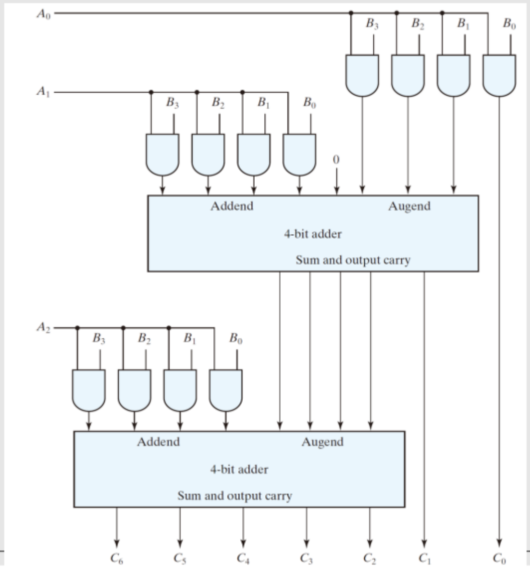


FIGURE 4.16

REPORT OF LAB 6 (4%)

The report should include …

1. 小明在路上撿到一塊FPGA板子，好奇的按下reset後，發現RCA數字順序為0, 12, 10, 18，請問A和B為何？(全列出)
2. 呈1題，CLA數字順序會怎麼顯示？誰比較快？
3. 呈2題，如果把CLA和RCA裡的 <= 全部寫成 = assignment，誰比較快？為什麼？
4. 小明參加心算比賽，他想用RCA的方法計算7+9，只見他第三秒就寫出答案，請問他寫的是多少？(reset後的第一狀態是第一秒)
5. 心得與討論
6. 題目難度: 超難/難/普/易/超易

- File type: pdf
- File name: Lab6_(Number of team)_report
- Deadline: 2022/05/04 23:59