數位系統導論實驗

Lab6 Verilog Modeling

Outline

- 課程目的
- 實驗工具
- Verilog Modeling 簡介
- 範例練習
- 作業說明及評分方式
- 附錄

課程目的

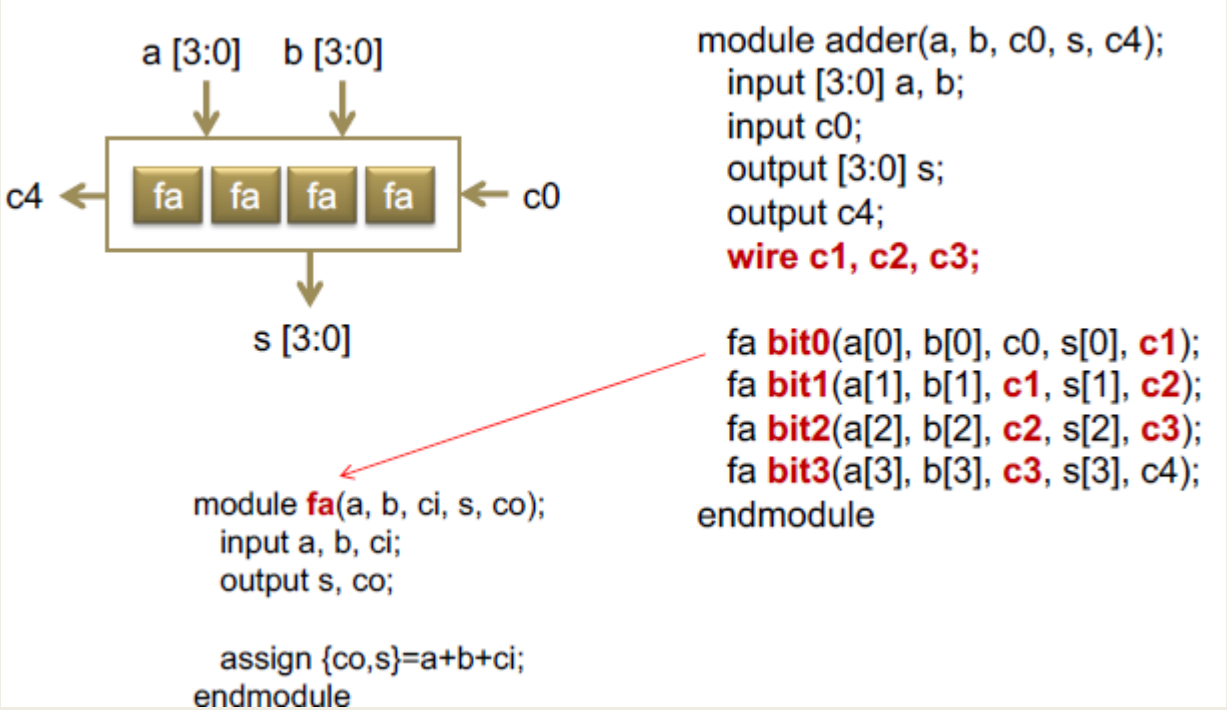
- 學習用 Verilog 完成生日碼產生器
- 學習以不同的 Verilog modeling 描述相同的硬體並了解 Verilog modeling 間的差異

實驗工具 – Icarus Verilog

- Icarus Verilog 是一套可進行Verilog 編譯與模擬的免費軟體，其內建GTKWave可顯示輸出波型。本次實驗課，會使用Icarus Verilog 中iverilog、vvp、gtkwave 來模擬硬體。

Verilog Modeling 簡介 (1/3)

- Verilog 電路的建模方式主要分為 Structural Modeling 與 Behavioral Modeling
- Structural Modeling：以邏輯閘、預定義模組及連結方式明確地描述數位電路



Verilog Modeling 簡介 (2/3)

- Behavioral Modeling：以行為描述數位電路，常用於設計 testbench，可分為 Continuous Assignment 與 Procedural Assignment
- Continuous Assignment: 以 assign 描述硬體的架構連結，位於 always 與 initial 外，且等式的左邊須為線

Continuous Assignment

```
module adder(a, b, c0, s, c4);
  input [3:0] a, b;
  input c0;
  output [3:0] s;
  output c4;

  assign {c4,s}=a+b+c0;

endmodule
```

Verilog Modeling 簡介 (3/3)

- Procedure Assignment: Verilog 的 Behavioral code 通常位於 Procedure Block 內，該 Assignment 賦值於型態為 reg、integer、時間變數…等的變數，但不能賦值於線，Procedure Block 的兩種型式為 always 與 initial

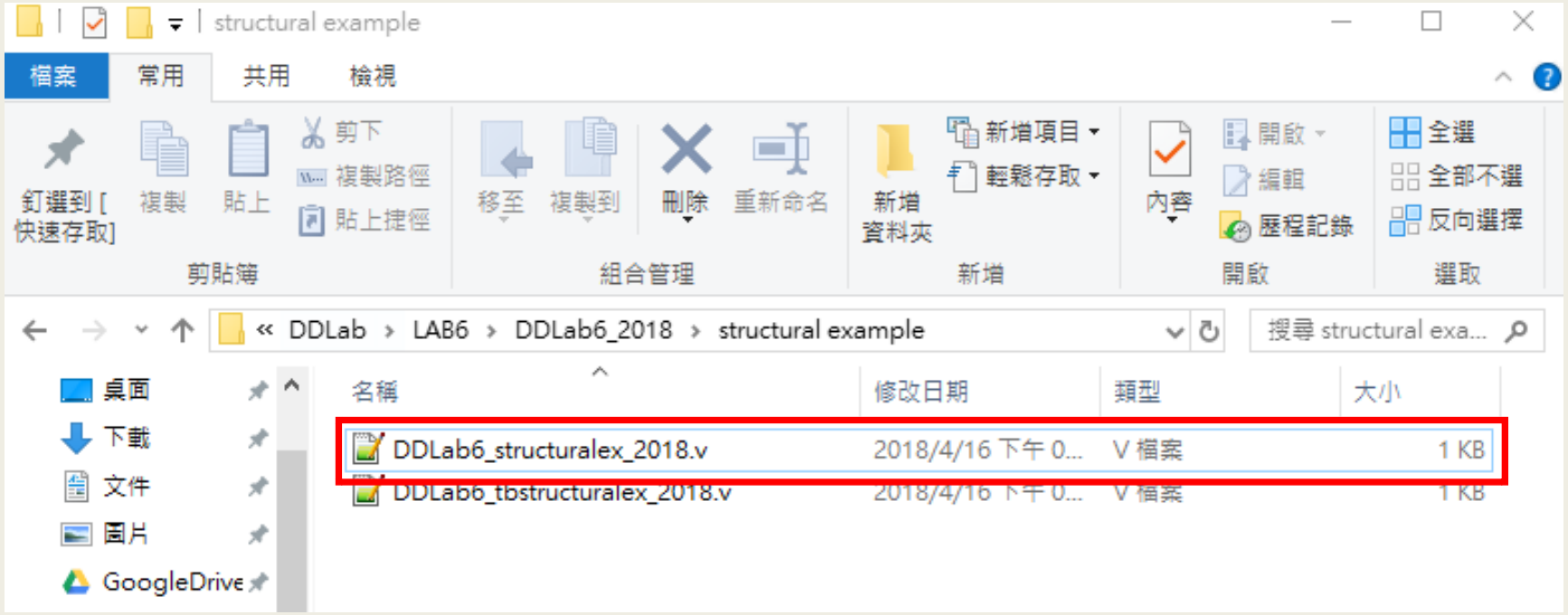
Procedural Assignment

```
module adder(a, b, c0, s, c4);
  input [3:0] a, b;
  input c0;
  output [3:0] s;
  output c4;
  reg [3:0] s;
  reg c4;

  always@(a or b or c0)
    {c4,s}=a+b+c0;

endmodule
```

範例練習一 – Structural Modeling (1/3)

- 將生日碼產生器以 Structural Modeling 完成，並以 testbench 驗證19760319

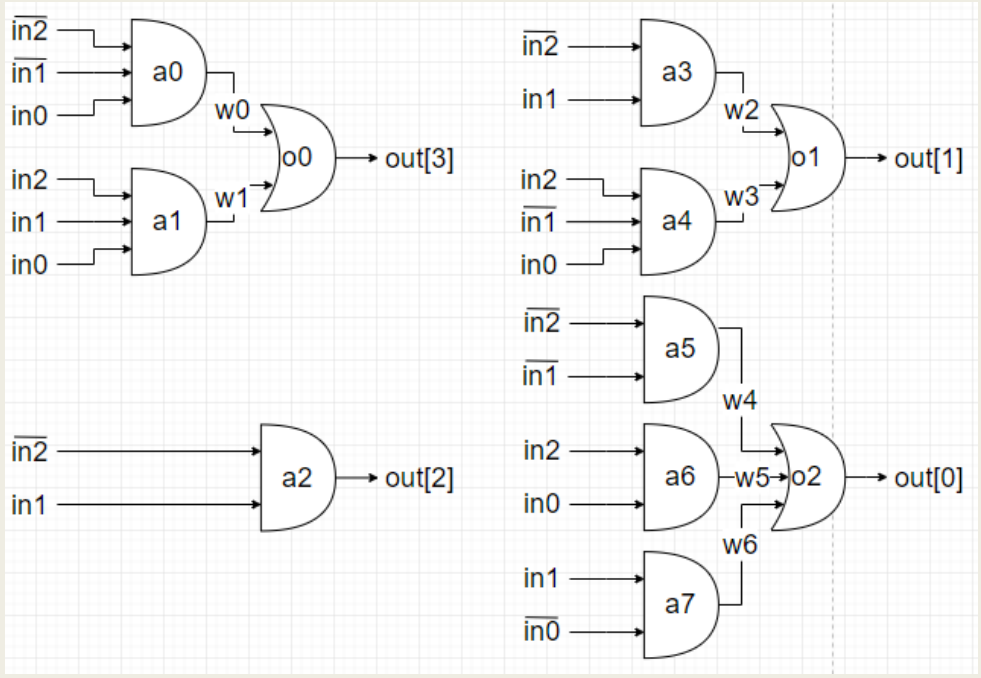
1. 請同學開啟課程壓縮包內的 DDLab6_structuralex_2018.v



範例練習一 – Structural Modeling (2/3)

2. 將生日碼產生器的邏輯圖內各個邏輯閘與線命名
3. 宣告輸入埠、輸出埠與線
4. 以 Verilog 內建的邏輯閘模組描述電路

※可於附錄查詢電路推導過程




```
//宣告區段
input [2:0]in; //宣告輸入埠 in ,其形態為 3bit wire
output  [3:0]out; //宣告輸出埠 out ,其形態為 4bit wire
wire nin2, nin1, nin0; //宣告線
wire w0, w1, w2, w3, w4, w5, w6;

//邏輯區段
//使用 not 將 in[2] 的補數訂為線 nin2
not n2(nin2, in[2]),
    n1(nin1, in[1]),
    n0(nin0, in[0]);

//使用 and 描述電路圖上的 and gate 與線的關係
and a0(w0, nin2, nin1, in[0]),
    a1(w1, in[2], in[1], in[0]),
    a2(out[2] , nin2, in[1]),
    a3(w2, nin2, in[1]),
    a4(w3, in[2], nin1, in[0]),
    a5(w4 , nin2, nin1),
    a6(w5, in[2], in[0]),
    a7(w6, in[1], nin0);

//使用 or 描述電路圖上的 or gate 與線的關係
or o0(out[3], w0, w1),
    o1(out[1], w2, w3),
    o2(out[0], w4, w5, w6);
```

範例練習一 – Structural Modeling (3/3)

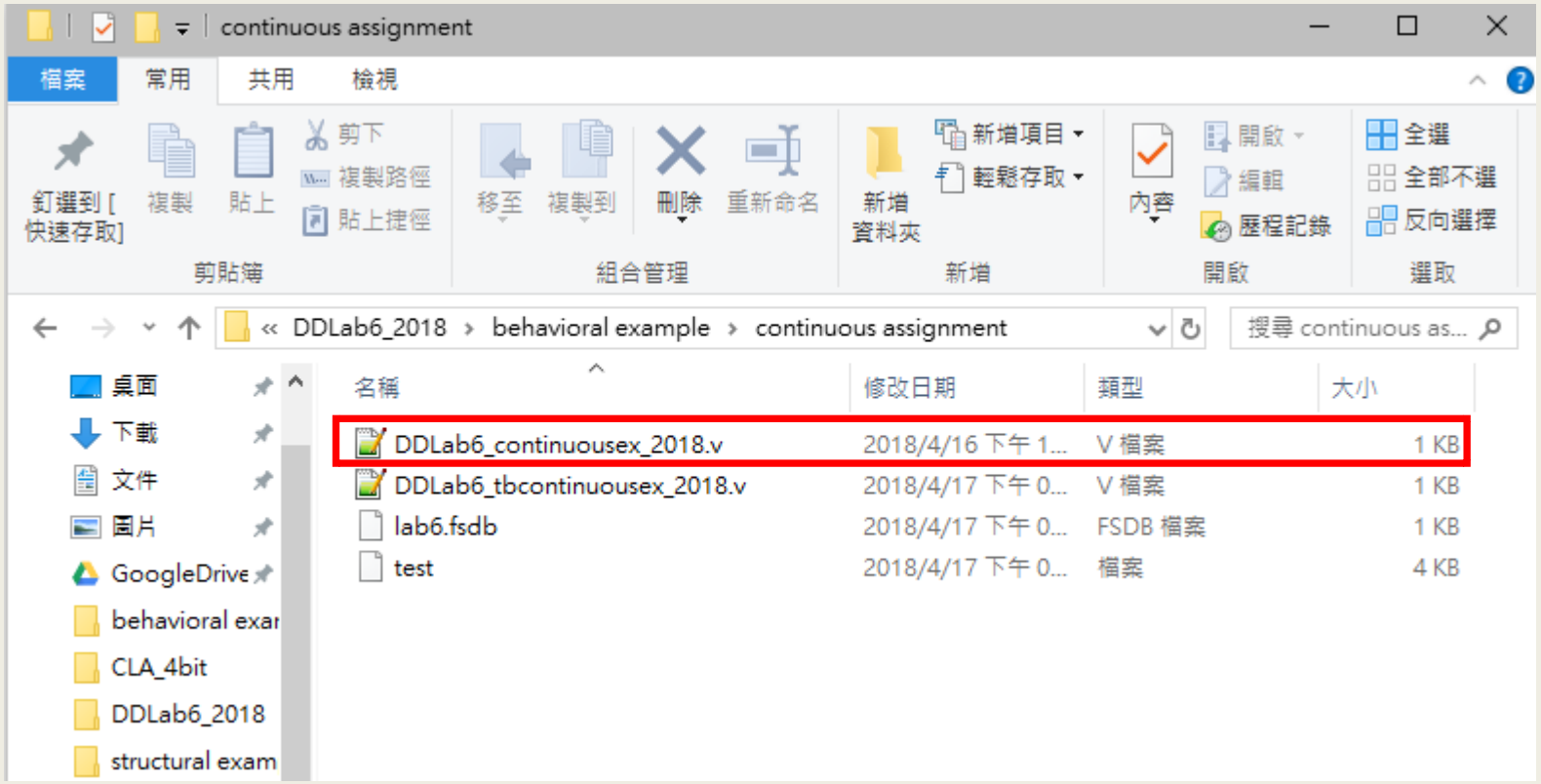
5. 開啟命令提示字元並進入當前資料夾
6. 輸入指令 “iverilog –o test DDLab6_tbstructuralex_2018.v ” 編譯程式
7. 輸入指令 ”vvp test” 執行程式並確認程式正確地依序輸出19760319
8. 輸入指令 “gtkwave lab6.fsdb” 觀察波形圖

```
D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\structural example>iverilog -o test DDLab6_tbstructuralex_2018.v

D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\structural example>vvp test
VCD info: dumpfile lab6.fsdb opened for output.
count =  0, out =  1
count =  1, out =  9
count =  2, out =  7
count =  3, out =  6
count =  4, out =  0
count =  5, out =  3
count =  6, out =  1
count =  7, out =  9

D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\structural example>gtkwave lab6.fsdb

GTKWave Analyzer v3.3.66 (w)1999-2015 BSI

[0] start time.
[160778] end time.
```

範例練習二 – Continuous Assignment (1/3)

- 將生日碼產生器以 Behavioral Modeling 的 Continuous Assignment 完成，並以 testbench 驗證19760319

1. 請同學開啟課程壓縮包內的 DDLab6_continuousex_2018.v



範例練習二 – Continuous Assignment (2/3)

2. 宣告輸入與輸出埠
3. 將生日碼產生器的 SOP 表示式以 assign 及 verilog 運算子描述

※可於附錄查詢 SOP 推導過程與常用運算子

```
module lab6(in, out); //模組 lab6(對應的輸入埠與輸出埠)

  //宣告區段
  input [2:0]in; //宣告輸入埠 in ,其形態為 3bit wire
  output [3:0]out; //宣告輸出埠 out ,其形態為 4bit wire

  //邏輯區段
  assign out[3] = (!in[2] & !in[1] & in[0]) | (in[2] & in[1] & in[0]),
    out[2] = (!in[2] & in[1]),
    out[1] = (!in[2] & in[1]) | (in[2] & !in[1] & in[0]),
    out[0] = (!in[2] & !in[1]) | (in[2] & in[0]) | (in[1] & !in[0]);
    //用 assign 敘述描述輸出埠 out 的各個位數值

endmodule //結束模組
```

$$out3 = (\overline{in2} * \overline{in1} * in0) + (in2 * in1 * in0)$$

$$out2 = (\overline{in2} * in1)$$

$$out1 = (\overline{in2} * in1) + (in2 * \overline{in1} * in0)$$

$$out0 = (\overline{in2} * \overline{in1}) + (in2 * in0) + (\mathrm{in1} * \overline{in0})$$

範例練習二 – Continuous Assignment (3/3)

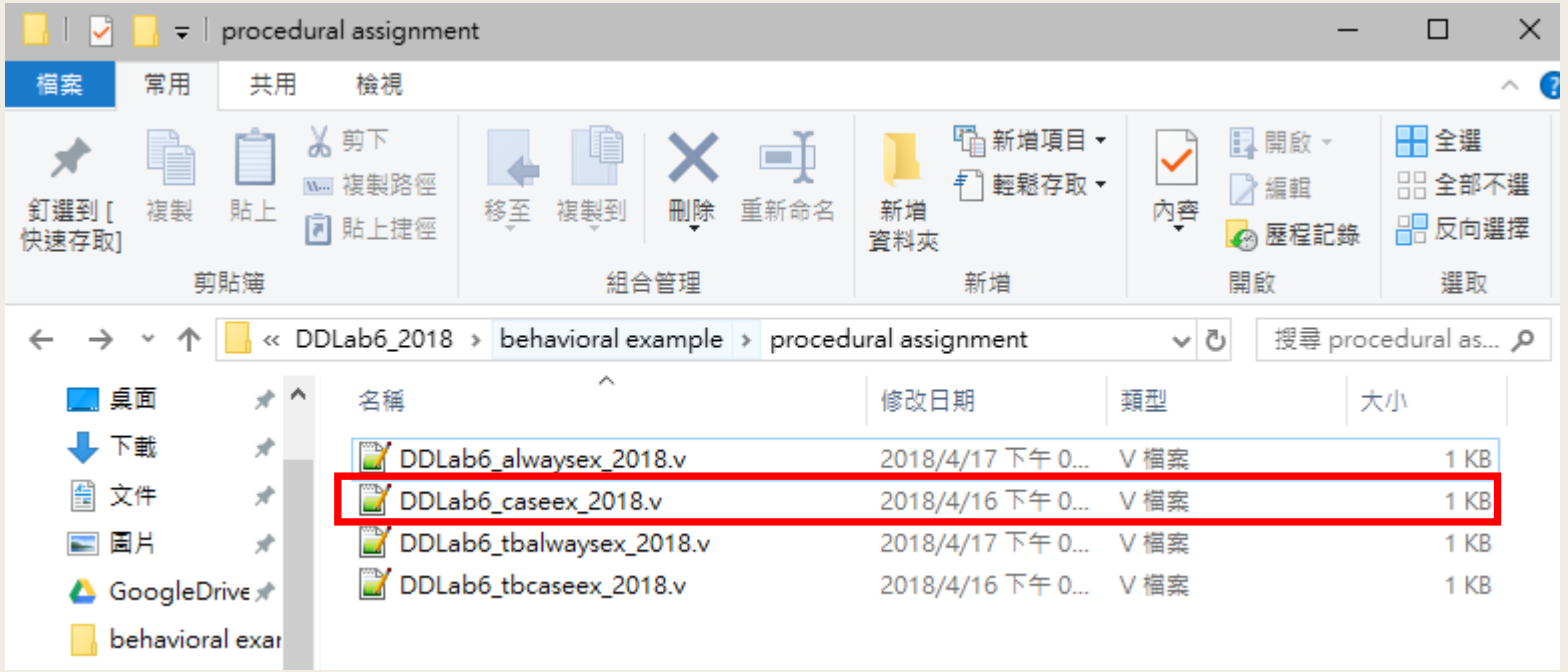
4. 開啟命令提示字元並進入當前資料夾
5. 輸入指令編譯程式 “iverilog –o test DDLab6_tbcontinuousex_2018.v ”
6. 輸入指令執行程式 ”vvp test” 並確認程式正確地依序輸出19760319
7. 輸入指令觀察波形圖 “gtkwave lab6.fsdb”

```
D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\behavioral example\continuous assignment>iverilog -o test DDLab6_tbcontinuousex_2018.v

D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\behavioral example\continuous assignment>vvp test
VCD info: dumpfile lab6.fsdb opened for output.
count =  0, out =  1
count =  1, out =  9
count =  2, out =  7
count =  3, out =  6
count =  4, out =  0
count =  5, out =  3
count =  6, out =  1
count =  7, out =  9

D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\behavioral example\continuous assignment>gtkwave lab6.fsdb

GTKWave Analyzer v3.3.66 (w)1999-2015 BSI

[0] start time.
[160778] end time.
```

範例練習三 – Procedural Assignment (1/3)

- 將生日碼產生器以 Behavioral Modeling 的 Procedural Assignment 完成，並以 testbench 驗證19760319

1. 請同學開啟課程壓縮包內的 DDLab6_caseex_2018.v



範例練習三 – Procedural Assignment (2/3)

2. 確認輸入與輸出之間的關係
3. 因 always 敘述內僅能對 reg 型態賦值，宣告輸入與輸出埠時還須令 out 為 reg
4. 藉由 always 敘述，描述硬體在輸入改變再次執行其內容，並以 case 描述輸入與輸出間的關係

```
//宣告區段
input [2:0]in; //宣告輸入埠 in ,其形態為 3bit wire
output reg [3:0]out; //宣告輸出埠 out ,其形態為 4bit reg

//邏輯區段
always @ (in) // 當變數 in 改變時，always 敘述會再次執行
case(in) //依照 in 值執行不同的敘述
   0 : out = 1; //當 in 為 0 , out 為1
   1 : out = 9;
   2 : out = 7;
   3 : out = 6;
   4 : out = 0;
   5 : out = 3;
   6 : out = 1;
   7 : out = 9;
   default : $display("Error in input"); //out 的預設敘述為 顯示錯誤訊息
endcase
```

input	output
0	1
1	9
2	7
3	2
4	0
5	3
6	1
7	9

範例練習三 – Procedural Assignment (3/3)

5. 開啟命令提示字元並進入當前資料夾
6. 輸入指令編譯程式 “iverilog –o test DDLab6_tbcaseex_2018.v ”
7. 輸入指令執行程式 ”vvp test” 並確認程式正確地依序輸出19760319
8. 輸入指令觀察波形圖 “gtkwave lab6.fsdb”

```
D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\behavioral example\procedural assignment>iverilog -o test DDLab6_tbcaseex_2018.v

D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\behavioral example\procedural assignment>vvp test
VCD info: dumpfile lab6.fsdb opened for output.
Error in input
count =  0, out =  1
count =  1, out =  9
count =  2, out =  7
count =  3, out =  6
count =  4, out =  0
count =  5, out =  3
count =  6, out =  1
Error in input
count =  7, out =  1

D:\GoogleDrive\106-2course\DDLab\LAB6\DDLab6_2018\behavioral example\procedural assignment>gtkwave lab6.fsdb

GTKWave Analyzer v3.3.66 (w)1999-2015 BSI

[0] start time.
[160778] end time.
```

作業說明及課程評分

- 作業Part1：正確呈現三個範例練習的結果
- 作業Part2：修改三個範例練習，以 minimal SOP 完成生日產生器並透過 testbench 顯示自己的生日 (驗收時請附上生日證明文件)
- 作業Part3：同學已於範例三練習 Procedural Assignment 的 case 敘述，請試著應用 for 敘述於 testbench，以 for loop 依序輸入 0~7 至生日碼產生器並顯示生日

作業說明及課程評分

- Demo 時間：測驗時間共分四梯次，分別為 19:20、19:40、 20:00 與 20:20
- Demo 梯次：與 Lab 1 相同
- Demo 地點：計中217
- 評分方式：part1 40%，part2 40%，part3 20%

附錄 - Verilog 資料型態簡介

- 常見的資料型態有連接線與暫存器
  - 連接線(net)：有wire、wand與wor，沒有記憶性，預設值為z(高抗阻)，用以將元件連結
  - 暫存器(reg)：有記憶性，且預設值為x(未定值)，用以儲存資料，程式碼使用reg時不代表一定存在一 register 或 flip-flop 於硬體

附錄 - 常見的運算子

+	加法	&&	邏輯 AND
-	減法	||	邏輯 OR
*	乘法	>	大於
**	次方	>=	大於等於
/	除法	<	小於
%	取餘數	<=	小於等於
~	NOT	==	等於
&	AND	!=	不等於
|	OR	<<	左移
^	XOR	>>	右移
!	邏輯 NOT	{}	連接變數

附錄 - 生日碼轉換 (1/5)

- 以下說明如何用真值表與卡諾圖完成生日碼(19720319)轉換，並以 SOP 式呈現

1. 設定輸入碼與輸出碼並將真值表畫出

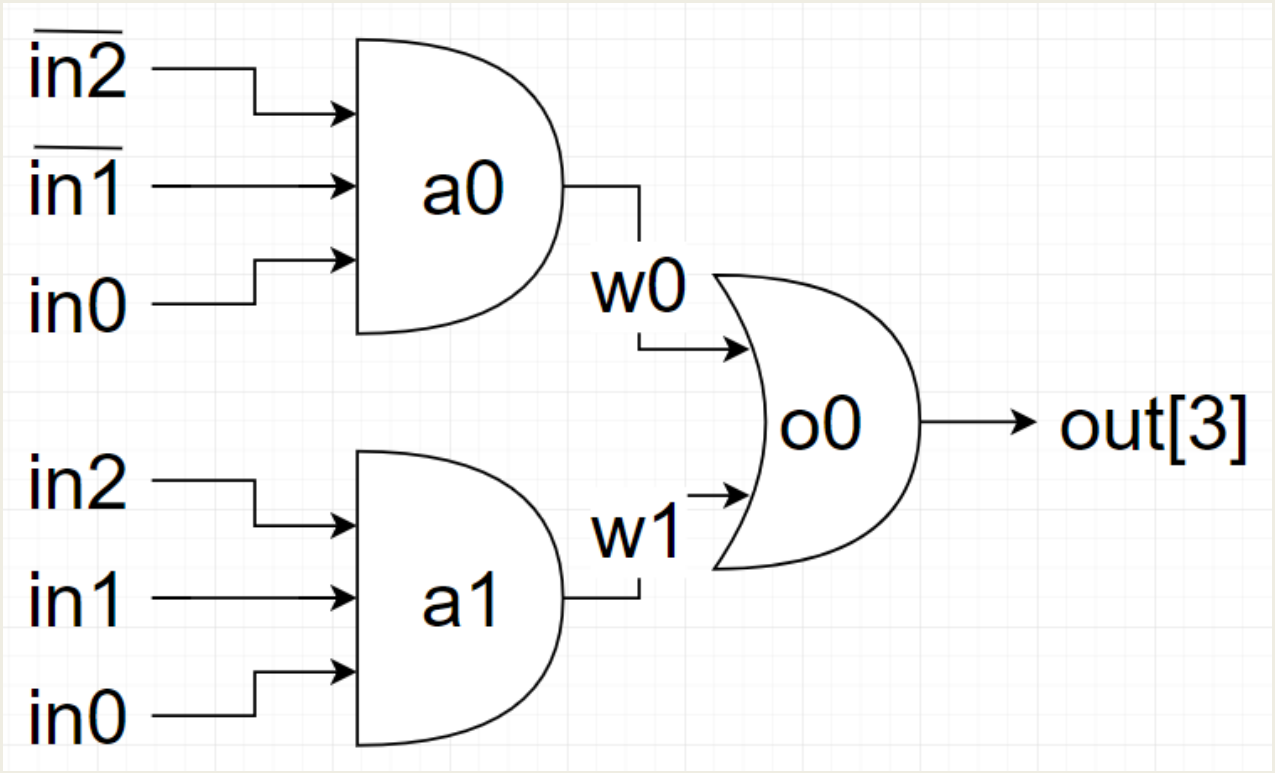
input	in2	in1	in0	out3	out2	out1	out0	output
0	0	0	0	0	0	0	1	1
1	0	0	1	1	0	0	1	9
2	0	1	0	0	1	1	1	7
3	0	1	1	0	1	1	0	2
4	1	0	0	0	0	0	0	0
5	1	0	1	0	0	1	1	3
6	1	1	0	0	0	0	1	1
7	1	1	1	1	0	0	1	9

附錄 - 生日碼轉換 (2/5)

2. 利用卡諾圖完成 minimal SOP

$$out3 = (\overline{in2} * \overline{in1} * in0) + (in2 * in1 * in0)$$

in1 in0 \ in2	0	1
00	0	0
01	1	0
11	0	1
10	0	0

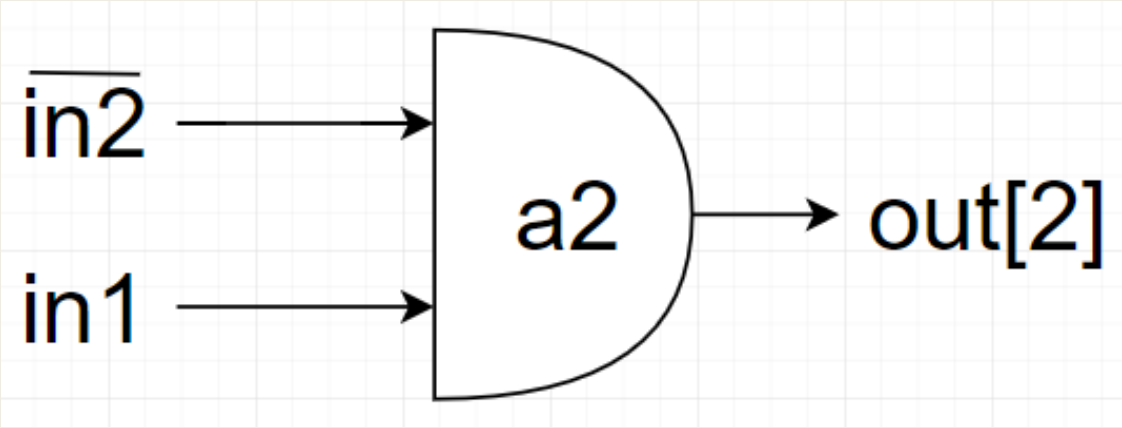


附錄 - 生日碼轉換 (3/5)

2. 利用卡諾圖完成 minimal SOP

$$out2 = (\overline{in2} * in1)$$

in1 in0 \ in2	0	1
00	0	0
01	0	0
11	1	0
10	1	0

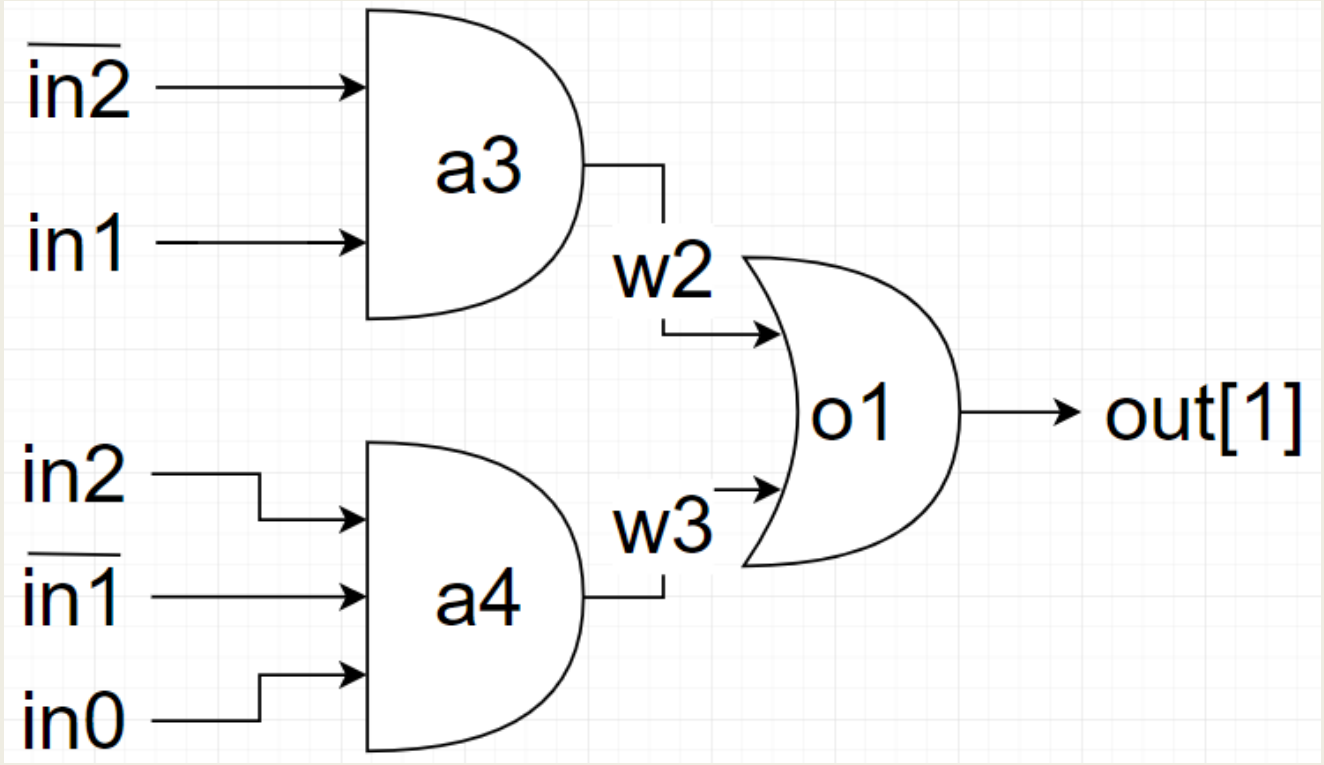


附錄 - 生日碼轉換 (4/5)

2. 利用卡諾圖完成 minimal SOP

$$out1 = (\overline{in2} * in1) + (in2 * \overline{in1} * in0)$$

in1 in0 \ in2	0	1
00	0	0
01	0	1
11	1	0
10	1	0



附錄 - 生日碼轉換 (5/5)

2. 利用卡諾圖完成輸出碼的邏輯運算式

$$out0 = (\overline{in2} * \overline{in1}) + (in2 * in0) + (\text{in1} * \overline{in0})$$

in1 in0 \ in2	0	1
00	1	0
01	1	1
11	0	1
10	1	1