第 2 講 設計與觀念

1. **參考文獻**:

[1]Online Verilog Simulator: edaplayground 教學:
https://www.youtube.com/watch?v=f9uwtAax4v0&t=249s

[2]金門大學陳鍾誠老師電子書: http://ccckmit.wikidot.com/ve:main

[3] https://www.chipverify.com/verilog/verilog-operators

[4]林灶生，verilog 晶片設計，全華 05579037

2. **基本觀念**

- 基本語法[2]

```
module <name> // 模組名稱
 parameter ... // 參數宣告
 port ... // 腳位宣告
 wire ... // 線宣告
 reg  ... // 暫存器宣告

 initial begin // 初始化設定區塊
 end

 assign ... // 資料處理層級之描述

 ... // 引用較低階模組別名

 always begin // 行為層級之描述區塊
   // 資料處理與指定等描述
   // task與function的使用
 end

 function  // 函數宣告
 task      // 作業宣告

endmodule
```

- 字符[4]
  - 運算子

    Y = ~X　// ~ 單一運算子

    W = X | Y // | 雙重運算子

    X = W?X:Y　//三重運算子

運算子型態	運算子	說　　明
算術運算子	+, - , * , / %	算術加減、乘、除 求餘數
邏輯運算子	! && ||	邏輯 not 邏輯 and 邏輯 or
關係運算子	>> >= < <=	大於 大於等於 小於 小於等於
等於運算子	= = != = = = != =	等於 不等於 狀況等於(case equality) 狀況不等於(case inequality)
位元邏輯 運算子	~ & | ^ ^~或~^	位元反邏輯 位元 and 位元 or 位元 xor 位元 nxor
精簡邏輯 運算子	& | ~& ~| ^ ^~ 或~^	精簡 and 精簡 or 精簡 nand 精簡 nor 精簡 xor 精簡 xnor
移位運算子	<< >>	左移 右移
條件運算子	? :	條件指定
連結運算子	{ }	連結

◆ 註解說明[4]

單行註解: //

多行註解: /* */

◆ 數字[4]

- < 長度 >'< 格式 (h, d, o, b)>< 數值 >

例如: 8'hec = 8 位元 16 進位 ec = ec_{16} = 14*16+12= 236_{10} = $1110,1100_2$

16'd2048 = 16 位元 10 進位 2048 = 2048_{10} = $0000,1000,0000,0000_2$

12'o2347 = 12 位元 8 進位 2347= 2347_8

= $2x8^3 + 3x8^2 + 4x8^1 + 7x8^0 = 1255_{10}$ = $0100,1110,0111_2$

4'b0101 = 4 位元 2 進位 0101 = 0101_2

-8'd79 = 8 位元 10 進位 -79 = -79_{10} = $-100,1111_2$ = $1011, 0001_2$

z: 高阻抗

x: 未知數值

例如: 4'b0x10 = $xx10_2$

8'h3z = 8 位元 16 進位 3z = $0011, zzzz_2$

12'oz732 = 12 位元 8 進位 z732 = $z732_8$ = zzz+ $7x8^2$+ 3x8 + 2

= $zzz474_{10}$ = zzz $1,1101,1010_2$

- 當固定長度取消時，如下:

788 = 內定 32 位元 10 進位 788

'b1101 = 內定 32 位元 2 進位 1101 = 1101_2

'o37 = 內定 32 位元 8 進位 37 = 37_8 =3*8 + 7 = 31_{10} = $1111,1111_2$

'hef = 內定 32 位元 18 進位 ef = ef_{16} =

◆ 字串 = “abcdefg”

跳脫字串	產生之字元
\n	字元(new line)
\t	字元 (tab)
\\	字元(\)
*	字元(*)
\ooo	由 1-3 個八進制數字所定義之字元 (0≦o≦7)
%%	字元(%)
\”	字元(”)

◆ 識別字[4]

例如: output [7:0] Qout; //Qout 是一個識別字

wire w; // w 是一個識別字

reg bus1; // bus1 是一個識別字

◆ 暫存器內存: 1bit 內容 有 0, 1, x(unkown value), z(高阻抗)。

◆ 保留關鍵字

Table: Verilog Reserved Keywords

always	and	assign	automatic
begin	buf	bufif0	bufif1
case	casex	casez	cell*
cmos	config*	deassign	default
defparam	design*	disable	edge
else	end	endcase	endconfig*
endfunction	endgenerate	endmodule	endprimitive
endspecify	endtable	endtask	event
for	force	forever	fork
function	generate	genvar	highz0
highz1	if	ifnone	incdir*
include*	initial	inout	input
instance*	integer	join	larger
liblist*	library*	localparam	macromodule
medium	module	nand	negedge
nmos	nor	noshow-cancelled*	not
notif0	notif1	or	output
parameter	pmos	posedge	primitive
pull0	pull1	pullup*	pulldown*
pulsestyle_ondetect*	pulsestyle_onevent*	rcmos	real
realtime	reg	release	repeat
rnmos	rpmos	rtran	rtranif0
rtranif1	scalared	show-cancelled*	signed
small	specify	specpa	strong0
strong1	supply0	supply1	table
task	time	tran	tranif0
tranif1	tri	tri0	tri1
triand	trior	trireg	use*
vectored	wait	wand	weak0
weak1	while	wire	wor
xnor	xor		

- 訊息顯示於標準輸出[4]:

  $display, $displayb, $displayh, $displayo, $write, $writeb, $writeh, $writeo

預設顯示格式	
任　務	預設值
$display	十進制
$displayb	二進制
$displayh	十六進制
$displayo	八進制
$write	十進制
$writeb	二進制
$writeh	十六進制
$writeo	八進制

```
$display("Hello Prof. Lin");          //顯示 Hello Prof. Lin
$display($time);                      //顯示市目前的時間
dat1= 4'b0101;
$display("The data is %b", dat1);     //顯示 The data is 0101
```

```
$write(4'b1111);
$write(" ", 5'b10101);
$write(" ",5'b01111, "\n");
則顯示：  15  21  0F
```

格　　式	資 料 顯 示
%d 或 %D	十進制
%b 或 %B	二進制
%h 或 %H	十六進制
%c 或 %C	ASCII 字元
%o 或 %O	八進制
%m 或 %M	階層名子(不須引數)
%s 或 %S	字串
%t 或 %T	目前時間格式
%e 或 %E	科學格式之實數
%f 或 %F	十進制格式之實數
%g 或 %G	十進制 / 科學格式之短實數
%v 或 %V	電壓強度

- 模擬監視[4]

$monitor、$monitoron及$monitoroff

```
module monitest ;
  integer s, t;
  initial begin
    s = 4;
    t = 6;
    forever begin
      #5 s = t + s;
      #5 t = s – 1;
    end   //forever begin
  end   //initial begin
  initial # 40 $finish;
  initial begin
   $monitor ($time, "s = %d ,t=%d," s, t);
  end //initial begin
endmodule
```

輸出:

```
0     S=  4 ,   t =  6
5     S=  10,   t =  6
10    S=  10,   t =  9
15    S=  19,   t =  9
20    S=  19,   t =  18
25    S=  37,   t =  18
30    S=  37,   t =  36
35    S=  73,   t =  36
```

- 閃控監視[4]: $strobe

```
forever @(posedge clk)
    $strobe("Time = %d, data = %h", $time, data);
```

在每一個時脈訊號 clk 正緣(由 0 -> 5 V 的上升邊緣)，$strobe 將時間與資料寫入標準的輸出(log 檔案與影幕)。

- 結束模擬[4]: $stop, $finish

```
initial
  begin
    clock = 1'b0 ;
    reset = 1'b0;
     #300   $stop ;     //懸住模擬，並將其設爲交談模式
     #600   $finish ;   //結束模擬
  end
```

範例 1. 左移 << and 右移 >>

```
module des;
   reg[7:0] data;
   integer i=0;

   initial begin
      data = 8'h1;
      $display ("orginal data = 'b%8b", data);
      for(i=0; i<10; i=i+1) begin
        // $display ("current loop#%0d, data << i = 'b%0b", i, data<<i);
          $display ("data << %0d = 'b%8b", i, data<<i);
      end

      data = 8'h80;
      $display ("orginal data = 'b%8b", data);
      for(i=0; i<9; i=i+1) begin
          $display ("data << %0d = 'b%8b", i, data>>i);
      end
```

```
        data = 8'h1;
        $display ("orginal data = 'b%8b", data);
        $display ("data >> 1 = 'b%8b", data>>1);

    end
endmodule
```

範例 **2[4].**

```
module equ_inequ();
    reg[3:0] a,b,c,d,e,f;
    initial begin
        a = 3; b=6;
        c = 4'b011;
        d = 4'bx11;
        e = 4'bx110;
        f = 4'bxx10;

        $display(a == b); //logical equality ==> 0
        $display(c != d); //logical inequality ==> x
        $display(c != f); //logical equality ==> 1
        $display(d === f); //conditional equality ==> 0
        $display(c !== d); //conditional inequality ==> 1
    end
endmodule
```

範例 **3.**

```
module test();
    reg[3:0] data;
    initial begin
        data = 4'bxxxx;
        $display("4'bxxxx == 2'bxx ? =>", data == 2'bxx); // ==> x
        $display("4'bxxxx === 2'bxx ? =>", data === 2'bxx); // ==> 0
    end
endmodule
```

3. 作業 2-1: 運算子 ^ 與 ^~ 執行結果的差異

(1) 寫一個簡單程式，請執行**^ 與 ^~**，將結果貼到作業報告中。