Hardware Design Language (Verilog)

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- 硬體描述語言的基本架構
- 事件基礎時間控制
- 行為描述語法
- 模組行為驗證

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- 硬體描述語言的基本架構
- 事件基礎時間控制
- 行為描述語法
- 模組行為驗證

模組描述的基本格式

```
module MyCPU(R_Wb, Data, Address, Clock, Reset);
   input  Reset, Clock;
   input  [31:0] Address;
   inout  [31:0] Data;
   output R_Wb;

      .
      .
      .

      .
      .
      .
endmodule
```

I/O port declarations

Resource / variable declarations

Structural / behavioral modeling

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- 硬體描述語言的基本架構
- 事件基礎時間控制
- 行為描述語法
- 模組行為驗證

輸入輸出埠描述

輸入埠：input
輸出埠：output

Example 1:

```
module fulladd(sum, c_out, a, b, c_in);
    output sum, c_out;
    input a, b, c_in;

// perform addition here.
endmodule
```

Example 2:

```
module fulladd(
    output sum,
    output c_out,
    input a,
    input b,
    input c_in
);

// perform addition here.
endmodule
```

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- **資料型態**
- 描述格式
- 硬體描述語言的基本架構
- 事件基礎時間控制
- 行為描述語法
- 模組行為驗證

資料狀態及型態

資料狀態

值	意義	說明
0	低電位	邏輯 0
1	高電位	邏輯 1
X	未定值	線路未初始化之前，或有 0、1 衝突的線路值
Z	高阻抗	三態緩衝器的輸出，高阻抗斷線 基本上代表斷路

資料型態

- 連接線Net (wire)：代表一條接線，沒有記憶性。預設值為Z。
- 暫存器Register(reg)：具記憶性，主要用於保持住電路中的資料。預設值為X。

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- **描述格式**
- 硬體描述語言的基本架構
- 事件基礎時間控制
- 行為描述語法
- 模組行為驗證

描述格式-註解

- 單行註解

```
module add(a,b,ci,sum,co);
      input a,b,ci;
      output sum,co;
// 註解在這裡
endmodule
```

- 多行註解

```
module add(a,b,ci,sum,co);
      input a,b,ci;
      output sum,co;
/*
註解在這裡
*/
endmodule
```

描述格式–數字格式

<size>'<base format> <number>

<size>: the number of bits

'<base>: 'd (decimal),
'h (hexadecimal),
'b (binary),
'o (octal)

' is the single quote symbol, ***not*** back quote symbol ` which will be used in compiler directive, Ex: `define

e.g.

4'b1111 // 4-bit binary number $1111_2=15_{10}$

12'habc // 12-bit hexadecimal number = $1010\_1011\_1100_2$

16'd255 // 16-bit decimal number = 0000000011111111_2

向量表示

宣告多位元的wire及reg時可使用向量(Vector)格式

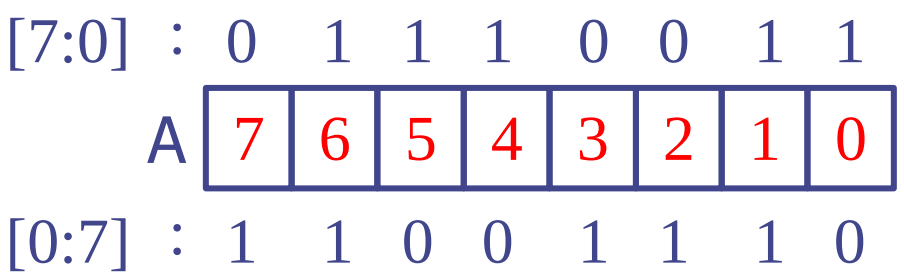
Example 1:

- wire [7:0] A; //宣告一8位元的匯流排
- reg [15:0] out; //宣告一16位元寬度的向量暫存器變數out

Example 2:

- wire [0:7] A;
- reg [0:15] out;

$115_{10} = 0111\_0011_2$



陣列

Verilog所提供陣列的儲存內容可以是整數、暫存資料、時間及向量，但不能為實數而且只適用於一維陣列。
表示格式為<array_name>[<subscript>]

integer A[0:15]; ⟶ 16個變數A的陣列

reg [3:0] B[0:15]; ⟶ 16個變數B的陣列，每一個B的位元寬度為4位元

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- **硬體描述語言的基本架構**
- 事件基礎時間控制
- 行為描述語法
- 模組行為驗證

硬體描述語言的基本架構

RTL

Behavioral

Dataflow

Gate

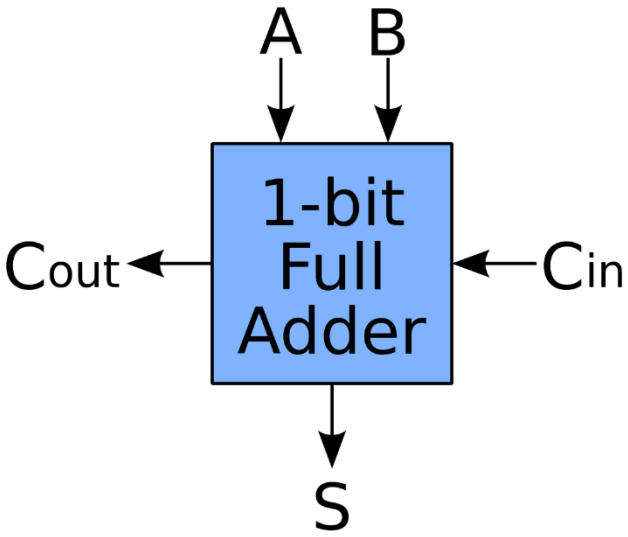
Switch

- Behavioral-Level
  描述模組功能
- Dataflow-Level
  描述電路資料如何在暫存器中儲存與傳輸
- Gate-Level
  描述以邏輯閘為基礎之連接形式
- Switch-Level(Transistor)
  描述元件開關及儲存點

```
module T_FF (q, clock, reset);
... <functionality of TFF>
endmodule
```

Behavioral-Level

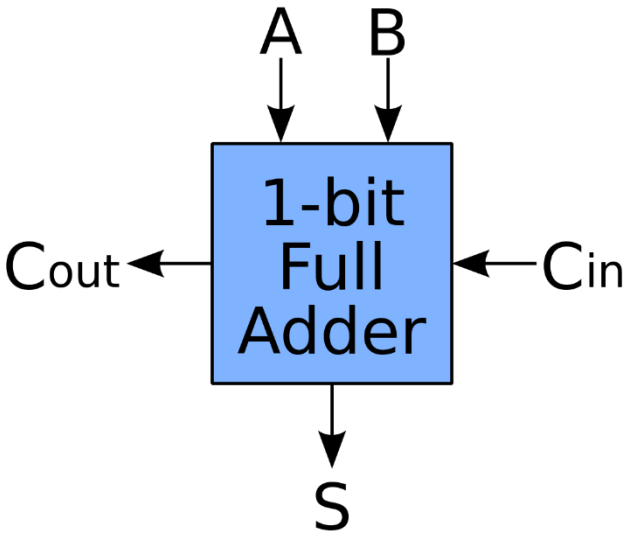
單純描述電路模組功能，
忽略硬體之低階電路架構。




```
// behavioral level modeling

module fulladd(sum, c_out, a, b, c_in);
output  reg  sum, c_out;
input a, b, c_in;

always @ (a or b or c_in)
     {c_out, sum} = a + b + c_in;

endmoudle
```

Dataflow-Level

說明資料如何在電路中傳送及運算。




```
// dataflow modeling

module fulladd(sum, c_out, a, b, c_in);
output sum, c_out;
input a, b, c_in;

assign {c_out, sum} = a + b + c_in;

endmoudle
```

運算子

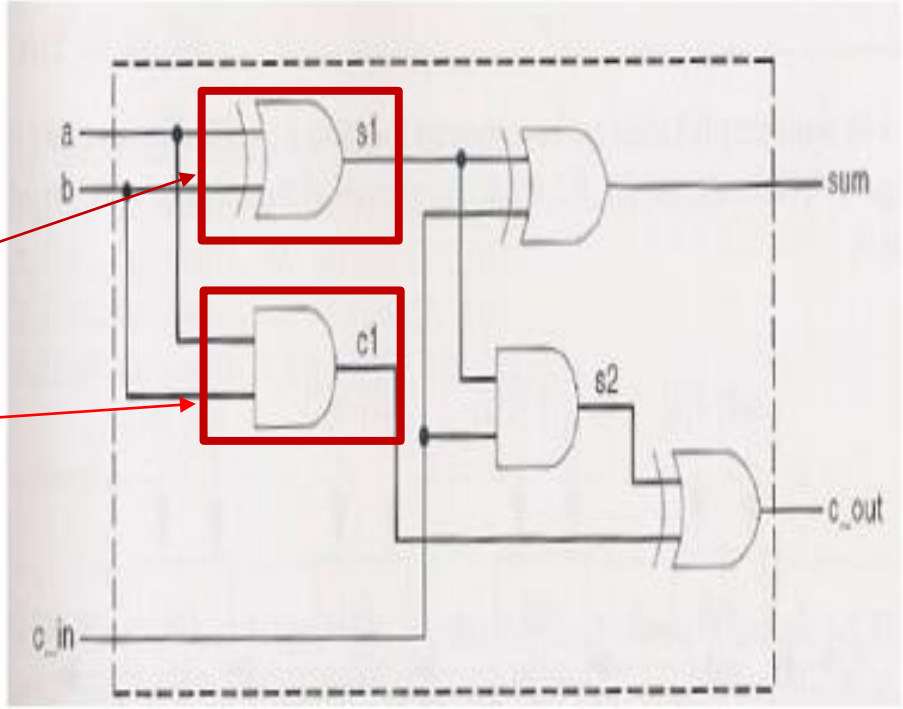
Type of operation	Operator symbol	Description	Number of operands
Arithmetic	+	addition	2
	-	subtraction	2
	*	multiplication	2
	/	division	2
	%	modulus	2
	**	exponentiation	2
Shift	>>	logical right shift	2
	<<	logical left shift	2
	>>>	arithmetic right shift	2
	<<<	logical left shift	2
Relational	>	greater than	2
	<	less than	2
	>=	greater than or equal to	2
	<=	less than or equal to	2
Equality	==	equality	2
	!=	inequality	2
	===	case equality	2
	!==	case inequality	2
Bitwise	~	bitwise negation	1
	&	bitwise and	2
	|	bitwise or	2
	^	bitwise xor	2
Reduction	&	reduction and	1
	|	reduction or	1
	^	reduction xor	1

Operator	Precedence
! ~ + - (unary)	highest
** * / % + - (binary)	
>> << >>> <<<	
< <= > >= == != === !==	
& ^ |	
&& ||	
?:	lowest

Gate-Level

由基本邏輯閘所組合而成的電路模組。

```
// structural level modeling

module fulladd(sum, c_out, a, b, c_in);
output sum, c_out;
input a, b, c_in;
wire s1, c1, c2;

      xor (s1, a, b);
      and (c1, a, b);
      xor (sum, s1, c_in);
      and (c2, s1, c_in)
      xor (c_out, c2, c1);

endmoudle
```



Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- 硬體描述語言的基本架構
- **事件基礎時間控制**
- 行為描述語法
- 模組行為驗證

事件基礎時間控制

事件發生於訊號發生改變時，可用來觸發一個敘述或包含多個敘述的區塊，且模組的輸入埠接收到一個新值也算是一個事件。

代表符號為@，表示當訊號產生正緣(posedge)，負緣(negedge)，轉換(transition)或數值改變時，其相關敘述才會被執行。

Example：

- always @(clock) Q=J;
  當clock訊號轉換時(1->0或0->1) 執行 Q=J 敘述
- always @(posedge clock) Q=J;
  當clock訊號正緣觸發時(0->1) 執行 Q=J 敘述
- always @(negedge clock) Q=J;
  當clock訊號負緣觸發時(1->0) 執行 Q=J 敘述

事件基礎時間控制

若一個always block可藉由多種訊號或事件其中之一觸發，則將這些訊號或事件以or(或)來區隔。

Example:

```
always @(reset or clock or A or B)
    begin
      if (reset)
         F=1'b0;
      else
         if (clock)
           F=A+B;
    end
```

Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- 硬體描述語言的基本架構
- 事件基礎時間控制
- **行為描述語法**
- 模組行為驗證

行為描述語法

同一硬體架構內之所有always block會平行(parallel)執行

always@(事件1 or 事件2 or … or 事件n)
begin
　　<敘述區>
end

always block內任一輸入訊號變化即執行敘述：

always@(*)
begin
　　<敘述區>
end

行為描述語法

```
// D Flip flop example

always @(posedge clk or negedge rst)    //發生clk正緣觸發或是rst負緣觸發事件時執行以下敘述
begin                                   //如同C語言的‘{‘
        if (!rst)                       //如果此always block觸發時rst為0即執行以下敘述
                q=1’b0;                 //q reset為0。因為只有一行敘述，所以begin end可加可不加
        else                            //若rst不為0，則執行以下敘述
        begin                           //因為else block敘述式不只一個，故要加begin end
                q=d;                    //將d的資料傳遞給q
        end
end                                     //如同C語言的‘}’
```

行為描述語法

If-elseif-else 為條件判斷式，達成該判斷條件則執行該敘述區塊，否則執行else內部的敘述區塊。
由上而下依次判斷條件，只要條件成立並執行完敘述區塊則不再繼續判斷。

```
If (判斷條件1)
begin
        <敘述區塊>
end
else if (判斷條件2)
begin
        <敘述區塊>
end
else if (判斷條件3)
begin
        <敘述區塊>
end
  .
  .
  .
else
begin
        <敘述區塊>
end
```

行為描述語法

Case 類似if-else的寫法，依照case後面的條件狀況判斷式，來判斷要執行哪一個狀況的敘述式，狀況無一符合時執行default裡面的敘述式。

*不用加break

```
case (條件狀況判斷式)
        狀況1:
        begin
                <敘述區塊1>
        end
        狀況2:
        begin
                <敘述區塊2>
        end
                .
                .
                .
        default:
        begin
                <敘述區塊n>
        end
endcase
```

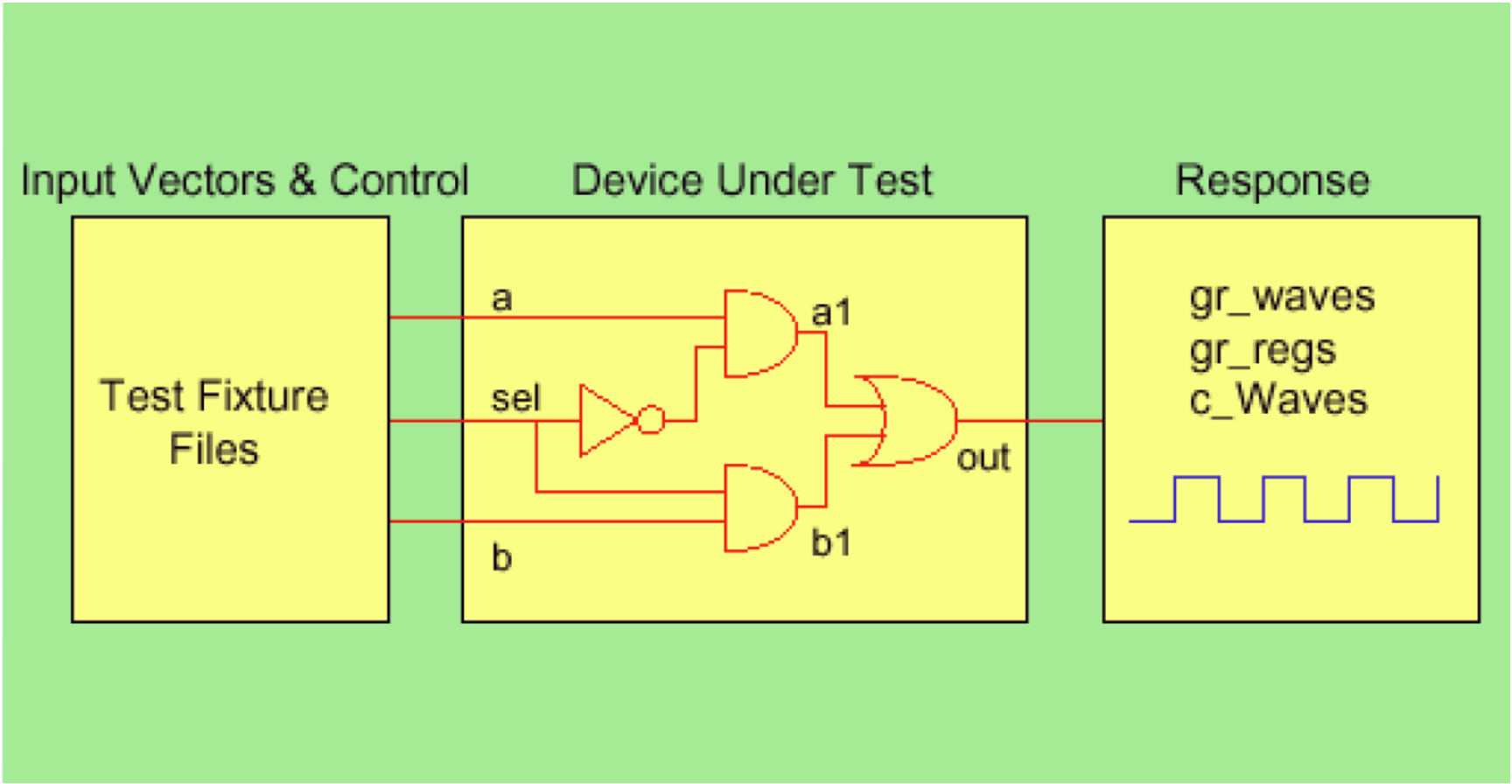
行為描述語法

```
Case (s)                    //判斷S的訊號
    2'b00:                  //若S的狀態是2bit二進制00則執行以下敘述區塊
    begin
        y = i[0];
        count = count+1;
    end
    2'b01:                  //若S的狀態是2bit二進制01則執行以下敘述區塊
    begin
        y = i[1];
        count = count+1;
    end
    2'b10:                  //若S的狀態是2bit二進制10則執行以下敘述區塊
    begin
        y = i[3];
        count = count+1;
    end
    default:                //以上狀況皆不符合時執行以下敘述區塊
    begin
        y = i[4];
        count = count+1;
    end
endcase
```

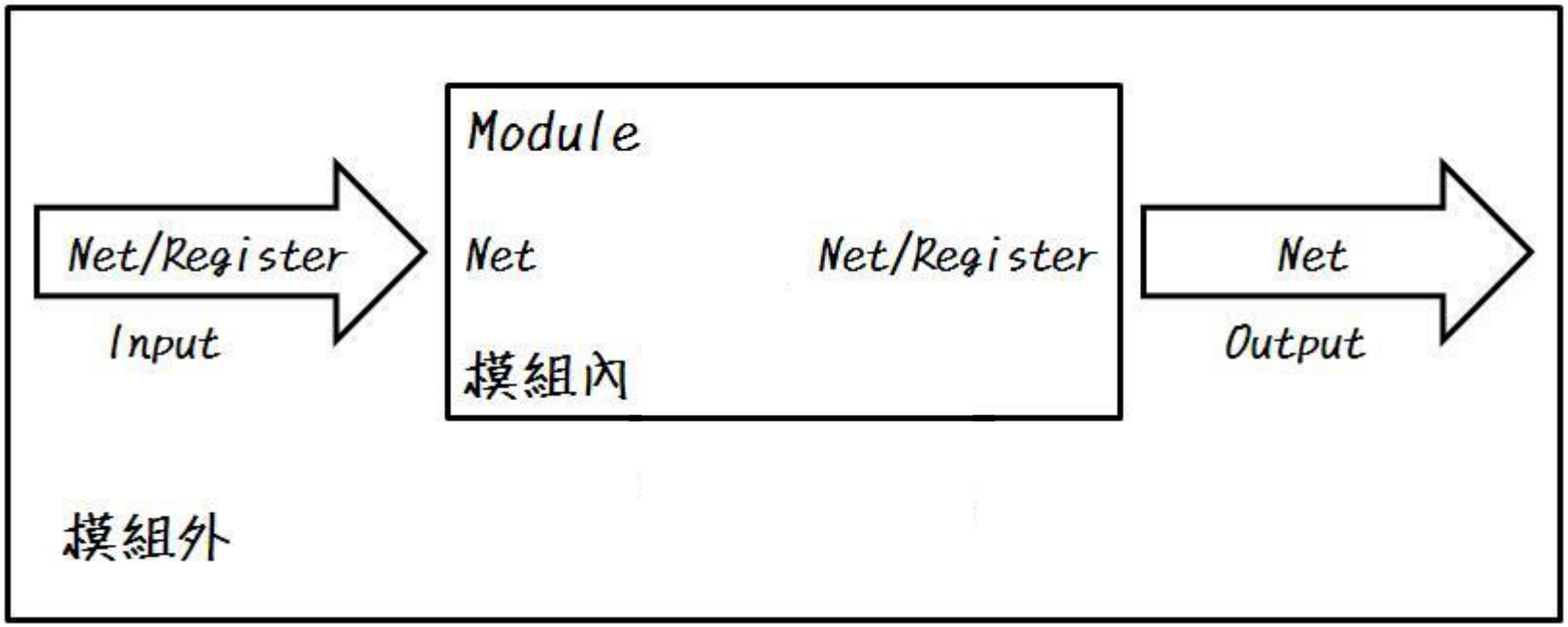
Outline

- 模組描述的基本格式
- 輸入輸出埠描述
- 資料型態
- 描述格式
- 硬體描述語言的基本架構
- 事件基礎時間控制
- 行為描述語法
- **模組行為驗證**

Testbench (Test Fixture)



Testbench (Test Fixture)



Testbench (Test Fixture)

```
module and(A, B, Y);

    input A;
    input B;
   output Y;

   AND(Y,A,B);
endmodule
```

```
1  // Verilog test fixture cre
2
3  `timescale 1ns / 1ps
4
5  module test_test_sch_tb();
6
7  // Inputs
8     reg B;
9     reg A;
10
11 // Output
12    wire Y;
13
14 // Bidirs
15
16 // Instantiate the UUT
17    and UUT (
18        .A(A),
19        .B(B),
20        .Y(Y)
21    );
22
23 initial begin
24        B = 0;
25        A = 0;
26        #5;
27        A = 1;
28        B = 0;
29        #5;
30        A = 0;
31        B = 1;
32        #5
33        A = 1;
34        B = 1;
35        #5
36        $finish;
37 end
38 endmodule
```