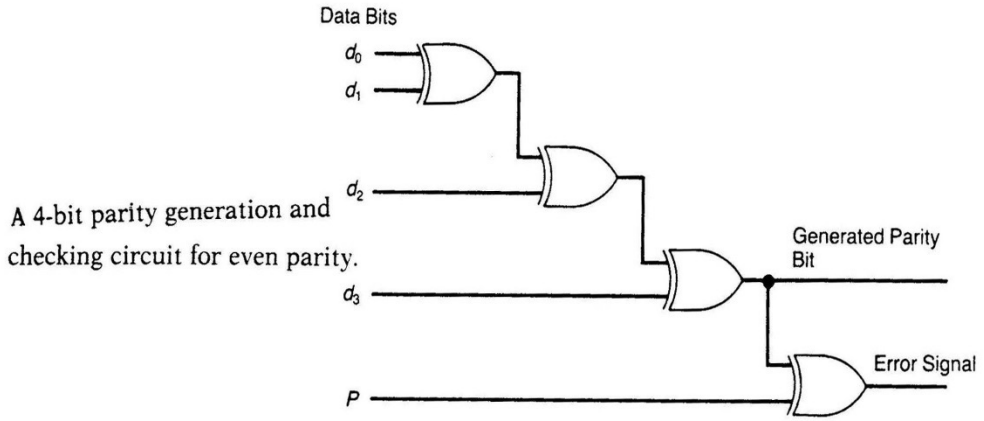
LAB - 03

陳培殷
國立成功大學　資訊工程系

Outline

- Combinational circuit
- HDL介紹
- Lab I – Full adder

Combinational Circuit



A 4-bit parity generation and checking circuit for even parity.

- 組合電路是一種邏輯電路，它任一時刻的輸出值，僅與目前的(present)輸入值有關，而與先前的(previous)輸入值無關
- 在電路結構上，組合電路由各種邏輯閘組成，電路中無記憶元件與反饋線
- 與之相對的則是循序邏輯電路，循序邏輯電路的輸出結果與目前的輸入及先前的輸入都有關係

Outline

- Combinational circuit
- **HDL介紹**
- Lab I – Full adder

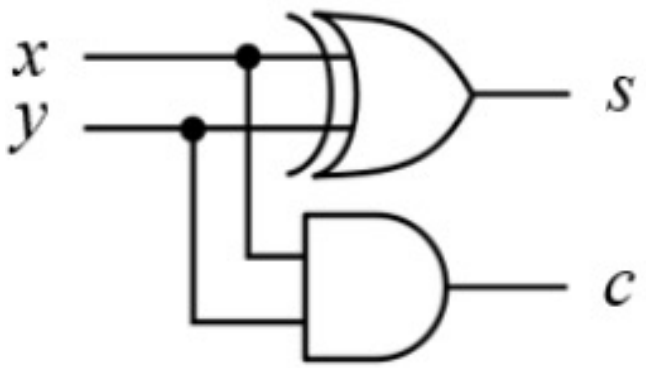
HDL介紹

- Verilog是一種用於描述、設計數位電路的硬體描述語言(HDL)
- 為了描述複雜的硬體電路，可以將功能劃分為不同模組，以Top-Down方式進行設計，提高開發效率

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5
6  and(carry, a, b);
7  xor(sum, a, b);
8
9  endmodule
```

HDL介紹

- 以半加器電路為例
- 半加器(Half adder)
  - 1 bit的半加器會接受兩個1 bit的輸入，並輸出兩個1 bit的訊號，分別為sum及carry



輸入		輸出	
A	B	C	S
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

HDL介紹

- 半加器之verilog電路
  - 輸入a、b；輸出sum、carry
  - 宣告時若未特別給定bit數，則該訊號為1 bit
  - 給定bit數，ex: input [2:0] a;

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5
6  and(carry, a, b);
7  xor(sum, a, b);
8
9  endmodule
```

模組宣告

輸入/輸出腳位宣告

HDL介紹

- Verilog可使用三種方式來描述電路
- Structural description:
  - Ex: and(out, in1, in2)、xor(out, in1, in2)
  - 有多個相同module時，需給定每個module名稱，ex: or u1(out, in1, in2)
  - 也可以呼叫使用者自訂的module，ex: HA half_adder(in1, in2, s, c)

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5
6  and(carry, a, b);      ——→ and閘
7  xor(sum, a, b);        ——→ xor閘
8
9  endmodule
```

HDL介紹

- Data flow description:
  - 使用continuous assignment，當右式變動時，左式的值會跟著隨時變化
  - 輸出不可包含輸入，ex: assign a = a + 1
  - 同一個訊號只能被assign一次

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5
6  assign carry = a&b;
7  assign sum = a^b;
8
9  endmodule
```

HDL介紹

- Behavior description:
  - 又稱作procedural assignment，會依照條件來觸發
  - 當always block的sensitivity list裡的訊號發生改變，就會執行一次該block的程式
  - 設計combination電路時，可用”*”取代sensitivity list內容

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5  reg sum, carry;
6
7  always@(a or b)
8  begin
9      sum = a^b;
10     carry = a&b;
11 end
12
13 endmodule
```

Sensitivity list

HDL介紹

- Data flow description vs Behavior description:
  - Data flow description中，運算結果須接至wire變數(input、output預設為wire)
  - Behavior description中，運算結果須存至reg變數

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5
6  assign carry = a&b;
7  assign sum = a^b;
8
9  endmodule
```

wire變數

Data flow description

```
1  module HA(a, b, sum, carry);
2
3  input a, b;
4  output sum, carry;
5  reg sum, carry;
6
7  always@(a or b)
8  begin
9      sum = a^b;
10     carry = a&b;
11 end
12
13 endmodule
```

宣告sum及carry為reg變數

Behavior description

HDL介紹

- Data flow description vs Behavior description:
  - Data flow description中，條件判斷以三元運算子進行
  - Behavior description中，可使用三元運算子、if-else、case進行條件判斷

```
15  module Mux(a, b, c);
16
17  input [2:0] a, b;
18  output [2:0] c;
19
20  assign c = (a > b)? a : b;
21
22  endmodule
```

Data flow description

```
15  module Mux(a, b, c);
16
17  input [2:0] a, b;
18  output reg [2:0] c;
19
20  always@(*)
21  begin
22      if(a > b)
23          c = a;
24      else
25          c = b;
26  end
27
28  endmodule
```

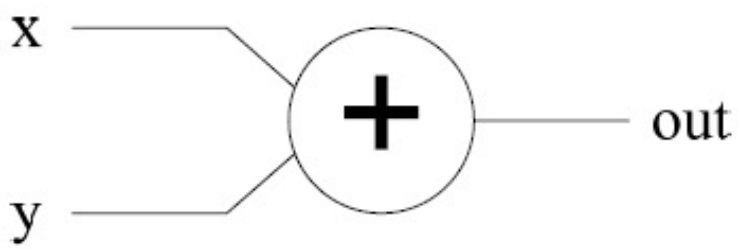
Behavior description

HDL介紹 – Example (1/4)

- 使用data flow description進行算數運算(+、-、*、/)

```
module  half_adder (x, y, out);

// port declaration
input           x, y;
output  [1:0]  out;

// functionality or structure
assign  out = x + y;

endmodule
```

2 bits wire變數



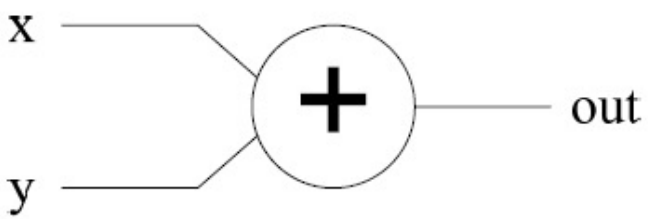
HDL介紹 – Example (2/4)

- 使用behavior description進行算數運算(+、-、*、/)

```
module  half_adder (x, y, out);

// port declaration
input           x, y;
output  [1:0]  out;

// I/O type
reg       [1:0]  out;

// functionality or structure
always @ (x or y)
begin
   out = x + y;
end
```

2 bits reg變數

Sensitivity list

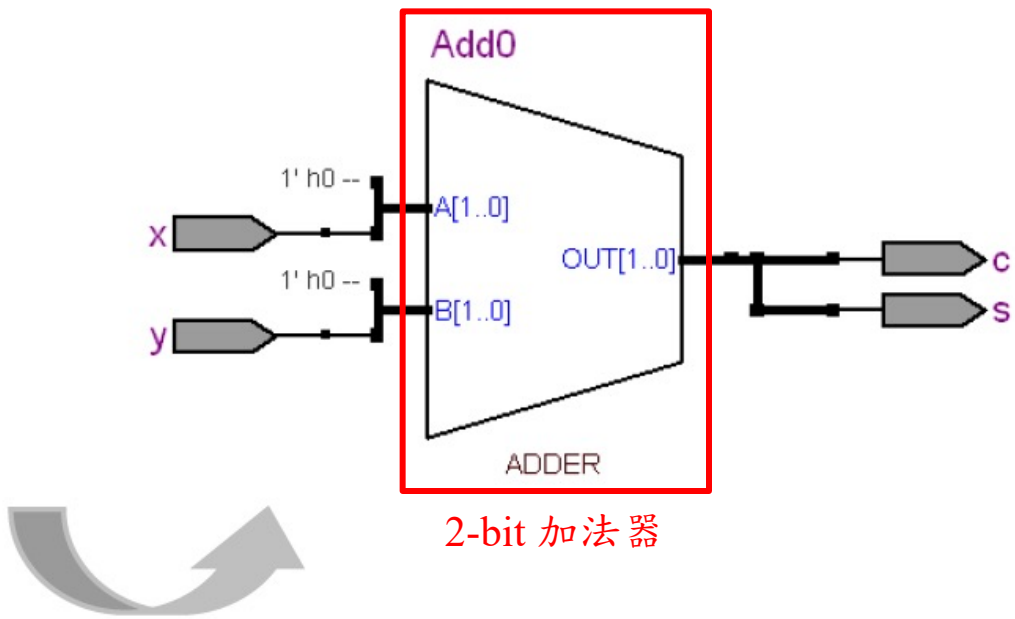


HDL介紹 – Example (3/4)

- Data flow description

```
module  half_adder (x, y, c, s);

// port declaration
input    x, y;
output  c, s;

// functionality or structure
assign {c, s} = x + y;

endmodule
```

將兩個1 bit wire變數串接為2 bits



2-bit 加法器

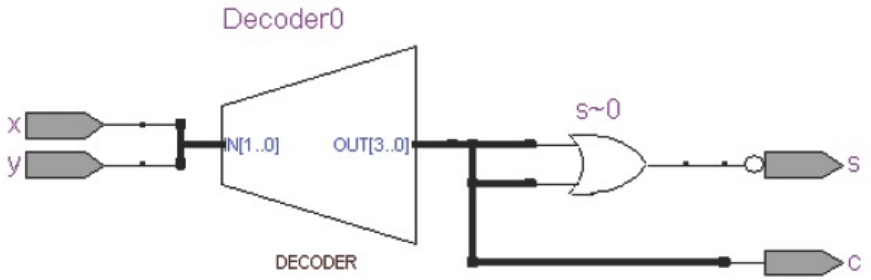
HDL介紹 – Example (4/4)

- 使用behavior description搭配truth table完成功能

```
// functionality or structure
always @ (x or y)
begin
   case ({x, y})
      2'b00: begin
         s = 0;
         c = 0;
      end
      2'b01: begin
         s = 1;
         c = 0;
      end
      2'b10: begin
         s = 1;
         c = 0;
      end
      2'b11: begin
         s = 0;
         c = 1;
      end
   endcase
end  //end always block
```

Truth table

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

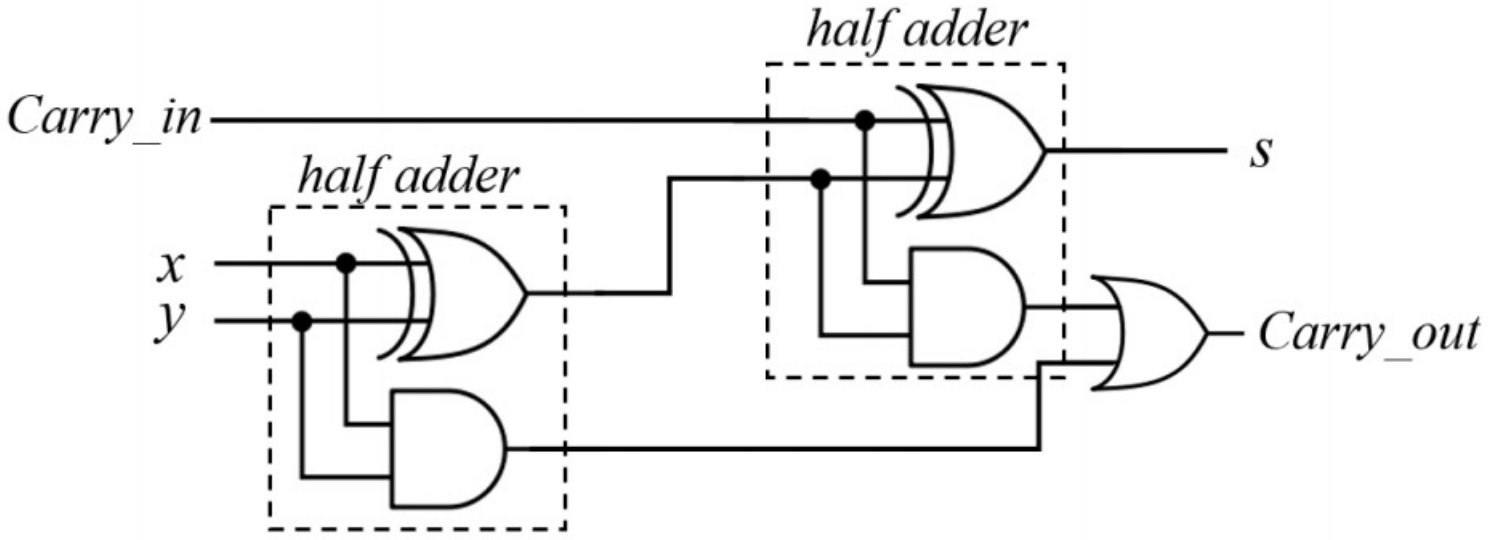


Outline

- Combinational circuit
- HDL介紹
- **Lab I – Full adder**

Lab 1 – Full Adder

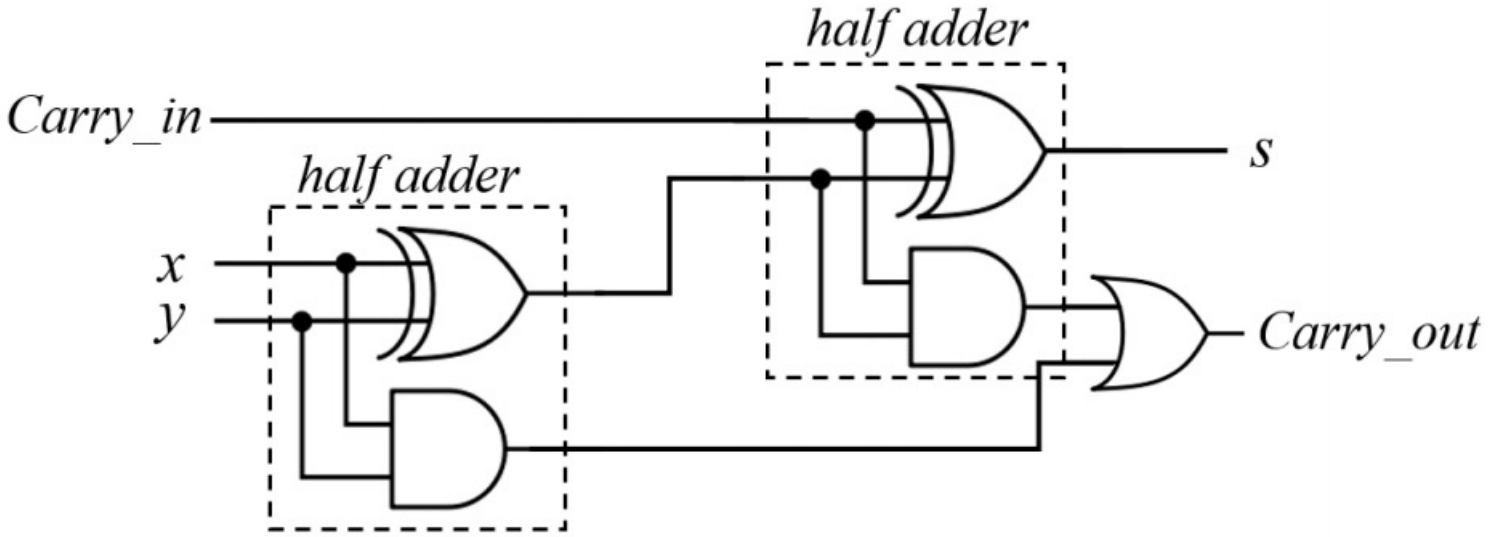
- **Full Adder**
  - 由 2 個 half adder 可組成一個full adder
  - 輸入兩個數及$carry_{in}$；輸出sum及$carry_{out}$



輸入			輸出	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Lab 1 – Full Adder

- 請分別以Structural、Data Flow、Behavior三種方式完成full adder
  - Input訊號為x、y及c_in；輸出訊號為sum及c_out
  - 輸入及輸出訊號皆為1 bit



輸入			輸出	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Notice for Lab I

- 使用Structural description撰寫時，需呼叫已完成的half adder模組
  - 呼叫方式分為兩種，可將訊號依據half adder模組腳位順序傳入，也可直接標明輸入訊號和腳位的對應方式

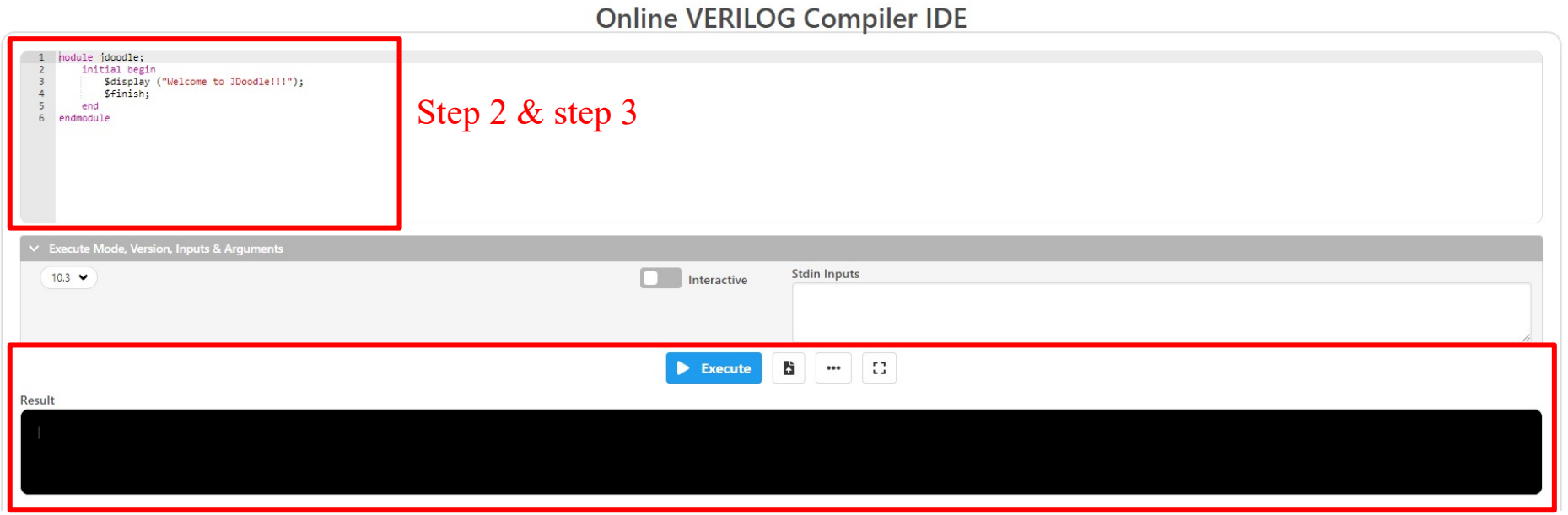
```
20  HA half_adder1(in1, in2, sum, carry);
21  HA half_adder2(.a(in1), .b(in2), .sum(sum), .carry(carry));
```

- 中間過程的結果可額外宣告wire變數儲存

```
17  input x,y,c_in;
18  output sum,c_out;
19  wire s1, c1, c2;
```

Notice for Lab I

- 驗證方式:
  - Step 1: 前往verilog線上模擬網站: https://www.jdoodle.com/execute-verilog-online/
  - Step 2: 將moodle上的testbench code整段貼上
  - Step 3: 修改full adder module
  - Step 4: 按下Execute按鈕，確認模擬結果是否正確



Notice for Lab I

- 模擬結果

Result

CPU Time: 0.00 sec(s), Memory: 7204 kilobyte(s)

```
X = 0 , Y = 0 , C_in = 0 , C_out = 0 , S = 0
X = 1 , Y = 0 , C_in = 0 , C_out = 0 , S = 1
X = 0 , Y = 1 , C_in = 0 , C_out = 0 , S = 1
X = 1 , Y = 1 , C_in = 0 , C_out = 1 , S = 0
X = 0 , Y = 0 , C_in = 1 , C_out = 0 , S = 1
X = 1 , Y = 0 , C_in = 1 , C_out = 1 , S = 0
X = 0 , Y = 1 , C_in = 1 , C_out = 1 , S = 0
X = 1 , Y = 1 , C_in = 1 , C_out = 1 , S = 1
```