數位系統設計實習

Lecture 2

Verilog HDL (Data Flow-Level)

指導老師：陳勇志 教授

實習課助教：鍾兆鋐

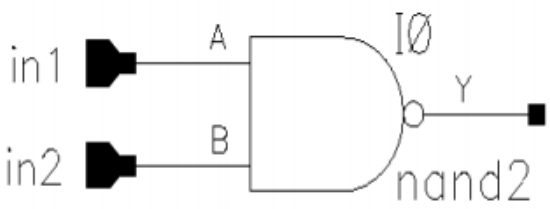
Outline

- Recap Gate-Level
- Verilog 描述式：Data Flow-Level
- Verilog 基礎語法與使用
- LAB 2-1 and 2-2

Chapter

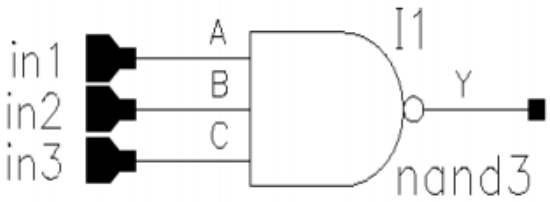
- Recap Gate-Level
- Verilog 描述式：Data Flow-Level
- Verilog 基礎語法與使用
- LAB 2-1 and 2-2

Recap Gate-Level

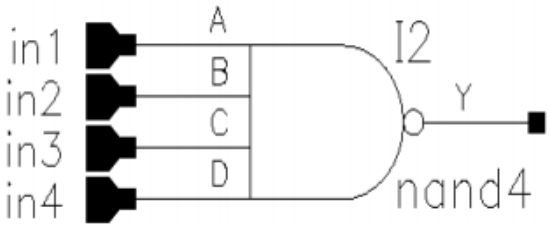


nand (y, in1, in2) ;

以宣告邏輯閘互相連接的方式，對設計電路進行描述。
是最初階的電路描述方法。



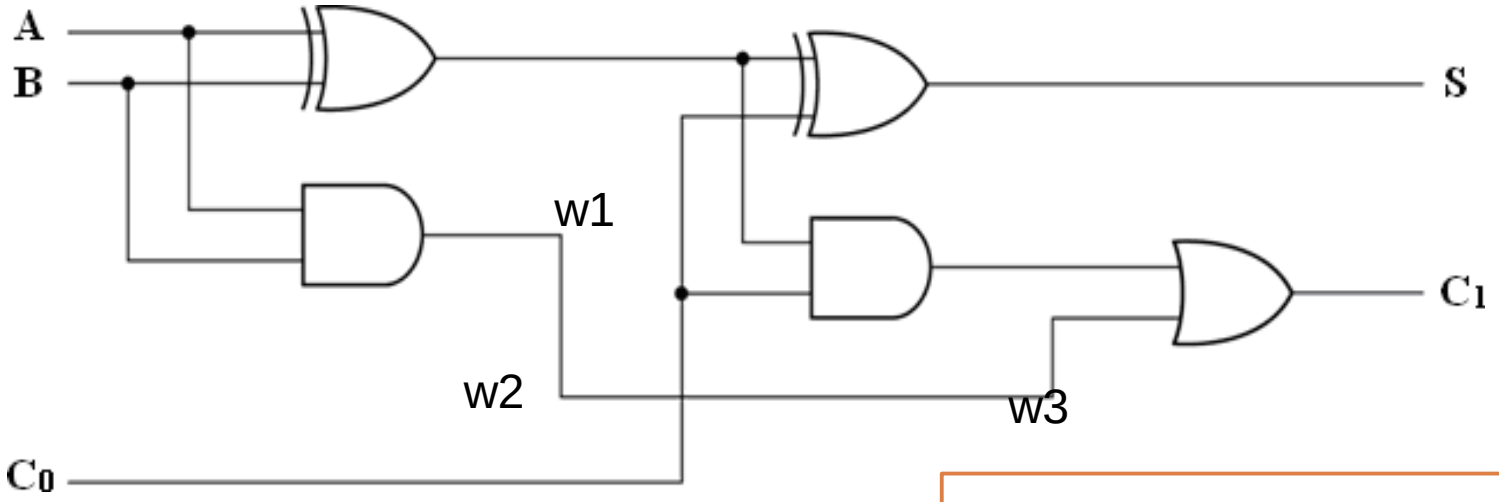
nand nand1(y, in1, in2, in3) ;



nand (y, in1, in2, in3, in4) ;

Gate-Level Modeling

Example code：Full Adder




```
xor     (w1,    A,
        B);
xor     (       S,
        w1, C0);
and     (w2,    A,
        B);
and     (w3     w1
```

Chapter

- Recap Gate-Level
- Verilog 描述式：Data Flow-Level
- Verilog 基礎語法與使用
- LAB 2-1 and 2-2

Verilog HDL(Data Flow modeling)


```
//Description of fulladder
module fulladder(S, C, A, B, C0);
input A, B, C0;
output S, C;
//Define combinational logic circuit
assign C = (A & B)|(B & Ci)|(Ci & A);
assign S = A ^ B ^ C0;
endmodule
```

根據Truth Table，能夠知道Cout(C),Sum(S)布林表示式。

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_i$$
$$= A\overline{B}\,\overline{C_i} + \overline{A}B\overline{C_i} + \overline{A}\,\overline{B}C_i + ABC_i$$

$$C_o = AB + BC_i + C_iA$$

Verilog basic Operator types

C	Verilog
* / + - %	* / + - %
! && ||	! && ||
> < >= <= == !=	> < >= <= == !=
~ & | ^ ~^	~ & | ^ ~^
>> <<	>> <<
?:	?:

(remainder)

Logical operation in statements

(xor)
(xnor)
(right shift)
(left shift)

Bit-level operation

Verilog的符號運算子與C語言的用法非常類似。

相關的符號運算子，可以參考左圖。

Bits vector

- 宣告 Input / Output 的位元寬度：

  input [3:0] A; (宣告4位元長度)

  output [3:0] B; (宣告4位元長度)

- wire [0:31] w1; // 32-bit wires with MSB = bit0
- Difference ? [2:0] vs [0:2]

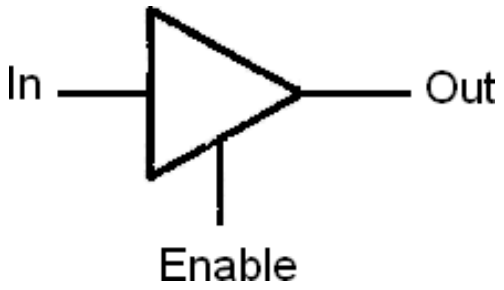
```
wire [2:0] a = 3'b110;
wire [0:2] b = 3'b110;
```

a[2]MSB	a[1]	a[0]LSB
1	1	0

b[2]LSB	b[1]	b[0]MSB
0	1	1

Data types

- 信號線代表的四種狀態值： 1, 0, X, Z



- 1:代表信號線在邏輯值1(ON)
- 0:代表信號線在邏輯值0(OFF)
- Z:代表信號線在高阻抗(High Impedance)
  - -輸入信號設定此值時,代表此信號為三態閘。
  - -輸出信號出現此值時,代表沒有信號源提供出邏輯值。
- X:代表信號線未知(Unknown)
  - -輸入信號設定此值時,代表隨意值(Dont care)
  - -輸出信號出現此值時,代表有兩個以上的信號源提供出不同的邏輯。

Number specification

- 明確指定常數值的位元寬度表示法：

  EX：

  二進制常數值格式：

  A = 8'b10101000;

  B = 8'b10XX100Z;

  十六進制常數值格式：

  x = 8'HFF;

  y = 8'HZF;

Number specification

- EX：

```
8'hAA            //h 十六進位，八位元
16'd1234         //d 十進制，十六位元
12'o7777         //o 八進制，十二位元
4'b1010          //b 二進制，四位元
```

*增加可讀性：

16'b0101_1010_1100_0000 (16’h6ac0)

(底線可以適當的使用，但請勿用在最高位元)

16'b_X0101_1010_1100_0000

Invaild!!

Chapter

- Recap Gate-Level
- Verilog 描述式：Data Flow-Level
- Verilog 基礎語法與使用
- LAB 2-1 and 2-2

Shift Operator

- 移位運算子：

  >> 右移 << 左 (空出的位元補0)

  ; 移

EX If b=1110 then

assign a=b >> 1 ; a=0111

assign a=b << 1 ; a=1100

Conditional Operator

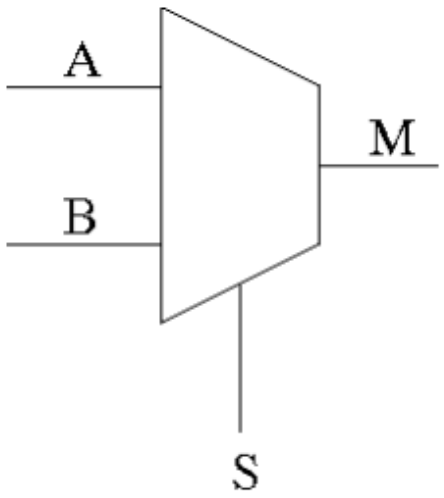
- 條件運算子： C語言也有此語法！

If a = 0110, b = 0101 then

assign m= (S) ? 1 : 0 ; m=1
assign m= (S) ? 1 : 0 ; m=0
assign m= (S) ? 1 : 0 ; m=1
assign m= (S) ? 1 : 0 ; m=0
assign m= (S) ? 0 : 1 ; m=1
assign m= (S) ? 0 : 1 ; m=1

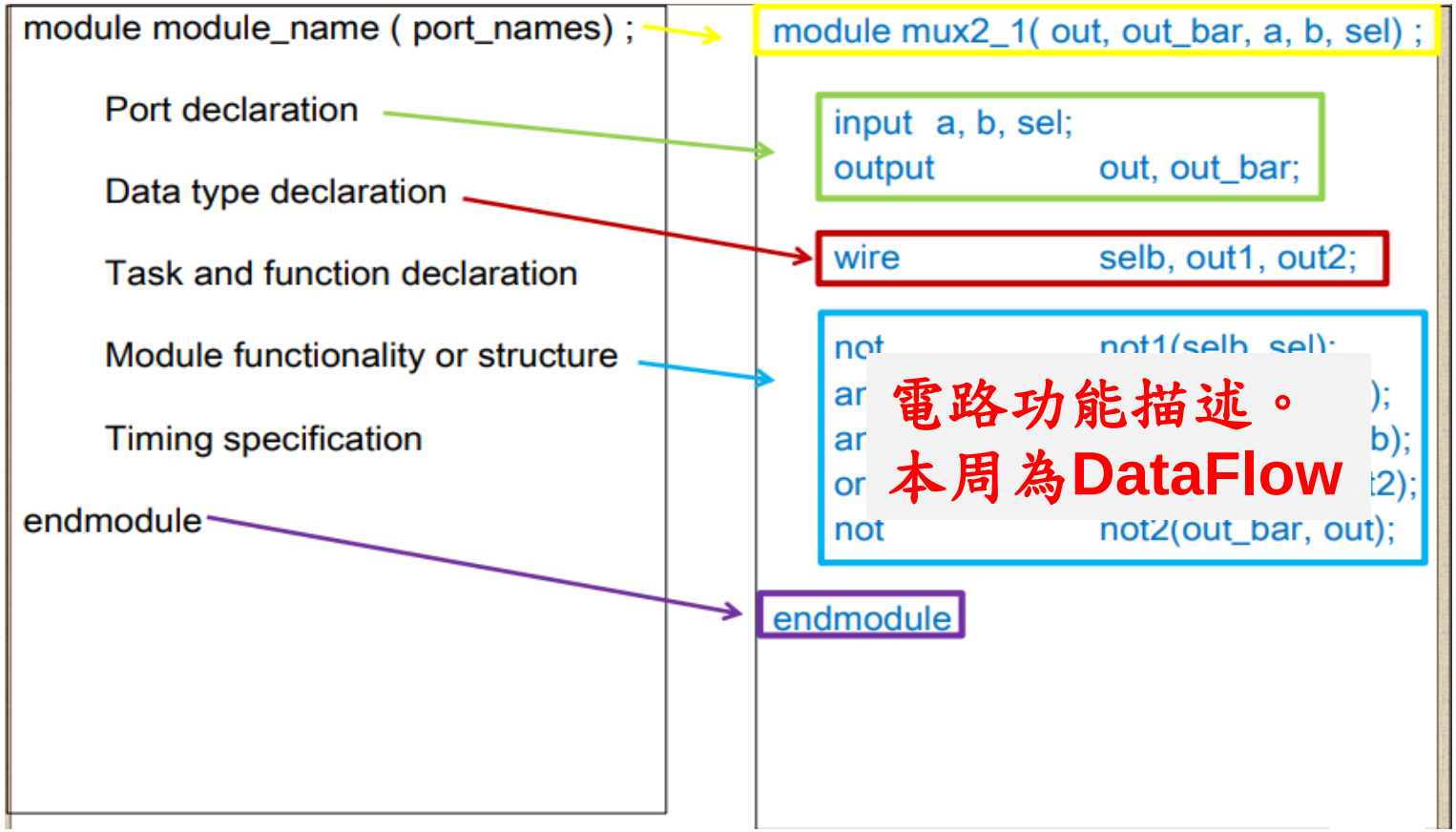
條件式

```
assign M = S ? A : B;
```



Recap Module Setup

回顧一下，上周的 Module 定義與宣告。



Module Instantiation

- Connecting ports to external signals
  - Connecting by ordered list, for example

```
module top;
input [3:0] A, B;
 input C_IN;
 output [3:0] SUM;
 output C_OUT;
fulladd4 mod (SUM, C_OUT, A, B, C_IN);
...
endmodule
```

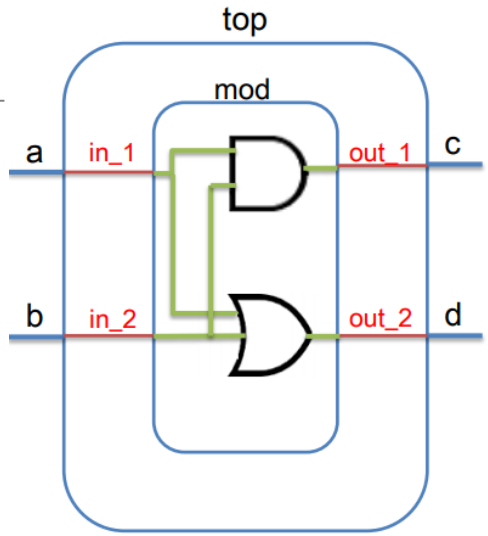
top-module

calling sub-module

可任意取名

```
module fulladd4(sum, c_out, a, b, c_in);
output [3:0] sum;
output c_cout;
input [3:0] a, b;
input c_in;
...
endmodule
```

sub-module



Module Instantiation

- Connecting ports by name：

```
fulladd4 fa_ordered
 (.c_out(C_OUT), .sum(SUM), .b(B), .c_in(C_IN), .a(A));
```

```
fulladd4 mod (SUM, C_OUT, A, B, C_IN);
```

```
module fulladd4(sum, c_out, a, b, c_in);
output [3:0] sum;
output c_cout;
input [3:0] a, b;
input c_in;
...
endmodule
```

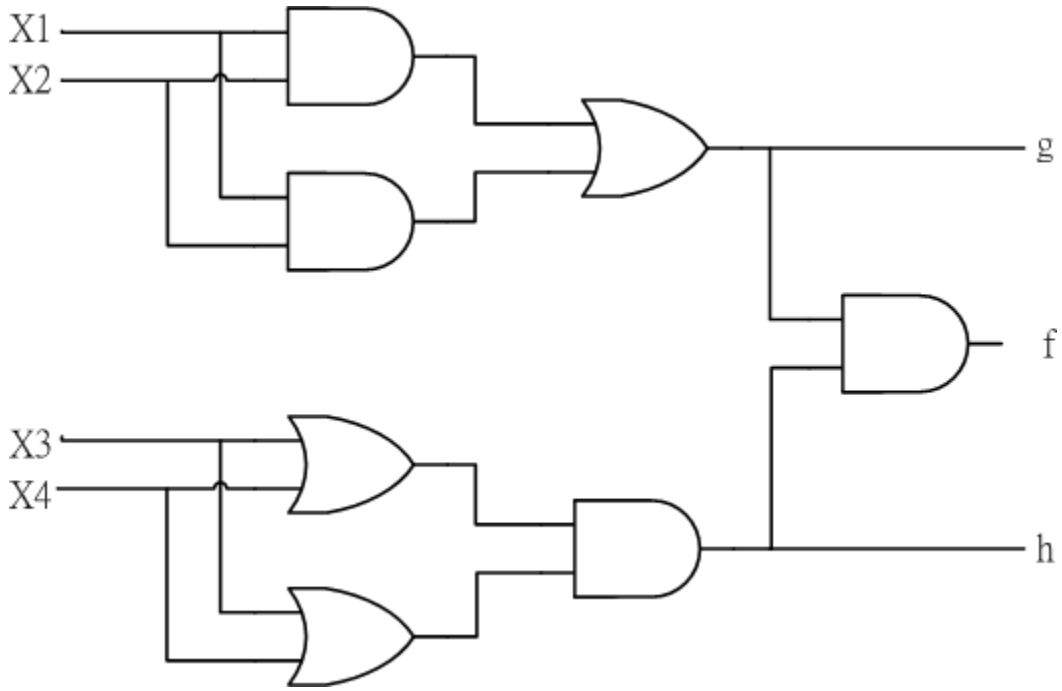
sub-module

Chapter

- Recap Gate-Level
- Verilog 描述式：Data Flow-Level
- Verilog 基礎語法與使用
- LAB 2-1 and 2-2

LAB 2-1

- 請使用**Data flow的方式**描寫出下圖之邏輯閘，並於 Quartus II 模擬訊號波型加以驗證結果。



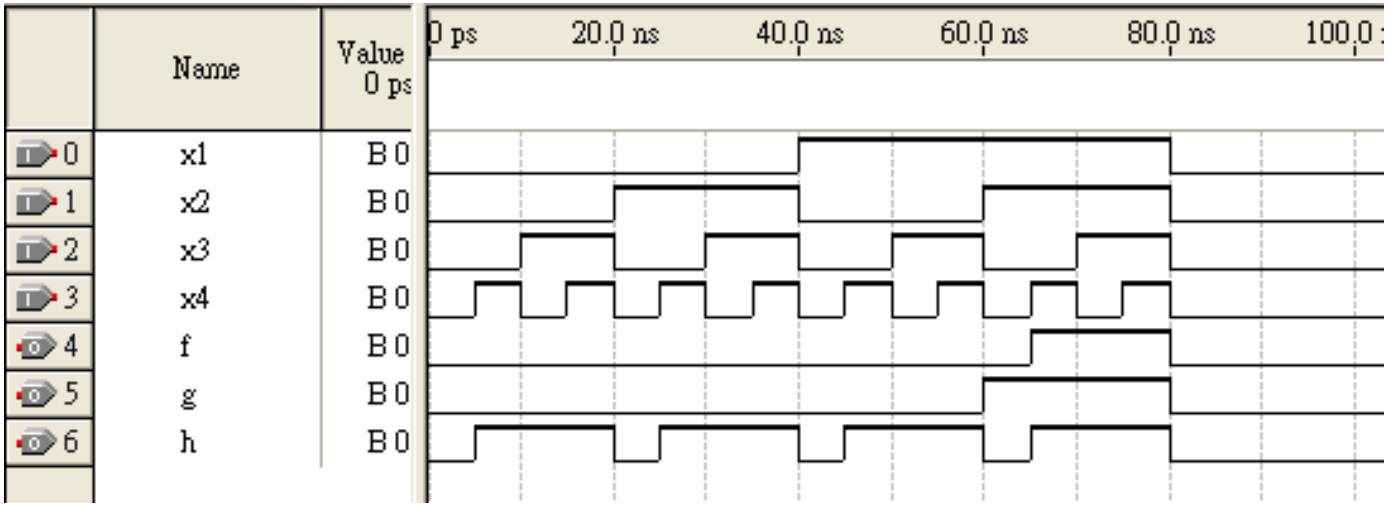
宣告wire：

Wire w1,w2…

Assign w1 = …?

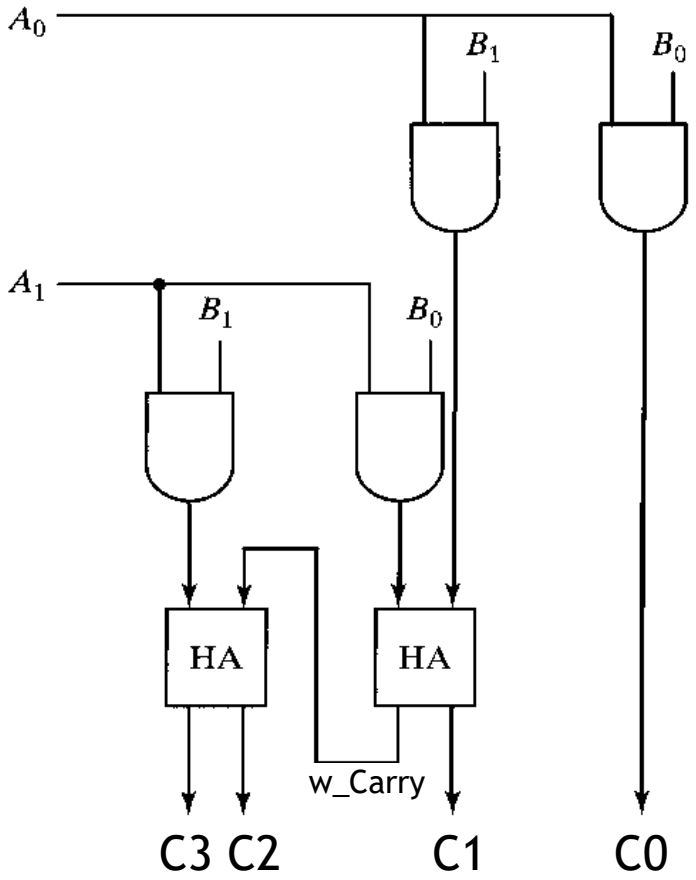
Assign g = …?

LAB 2-1


Name
Value
0 ps
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
0 x1 B 0
1 x2 B 0
2 x3 B 0
3 x4 B 0
4 f B 0
5 g B 0
6 h B 0

LAB 2-2

- 請使用**Data flow的方式**描寫出一2-bit multiplier 如下圖所示，並於Quartus II 模擬訊號波型加以驗證結果。

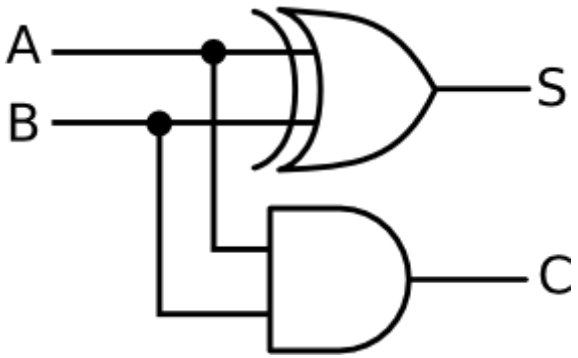


Note:
Write “HA “ as a sub-module

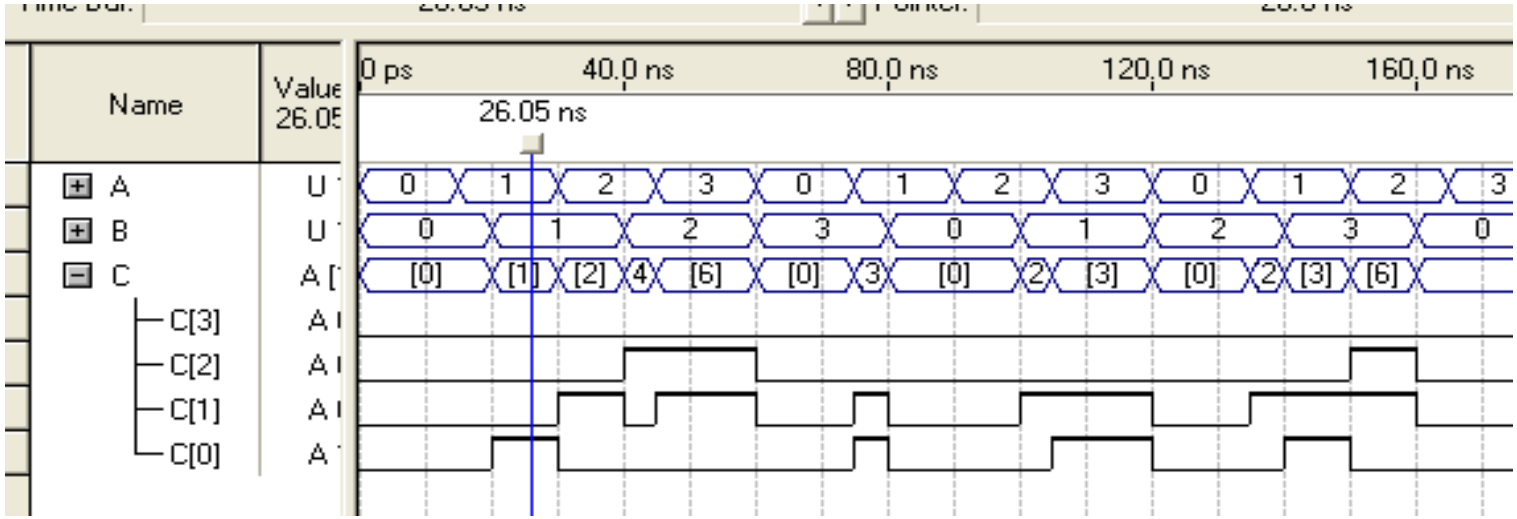
```
module XXX(A,B,C);
output [3:0] C;
input [1:0] A, B;

……

endmodule
```

Hint - HA

- Half Adder



LAB 2-2


Name
Value
26.05
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
26.05 ns
A
B
C
C[3]
C[2]
C[1]
C[0]

LAB 2

下課前繳交至moodle：

上傳verilog.v

波形截圖