DD LAB8

Array Multiplier & Signed handling

助教:陳冠良、方泰翔、郭政頡、鄭東昇、陳憲億

Outline

- 課程目的
- Array Multiplier
- Sign Extension
- 教材說明
- 課堂練習
- LAB作業
- Demo事項

課程目的

學習完先前的LAB，我們已經了解如何透過多個全加器設計RCA與CLA架構，在數位系統導論課程中也學習了無號乘法與有號乘法的原理。此LAB會教大家：

- 透過全加器與半加器設計無號乘法器Array Multiplier
- 透過signed handling及sign extension 設計有號乘法器

Array Multiplier

- Array Multiplier 是一個計算方式與直式乘法相似的硬體架構
- 可分為RCA array multiplier和CSA (Carry Save Adder) array multiplier
- 先將被乘數與各個乘數的bit相乘得到該部分的乘積，再透過加法器將各個部分乘積相加得到結果

```
A          0 1 0 1 (5)
B        x 1 1 0 0 (12)
-------------------------
           0 0 0 0
         0 0 0 0
       0 1 0 1
   +)  0 1 0 1
-------------------------
A*B  0 0 1 1 1 1 0 0 (60)
```

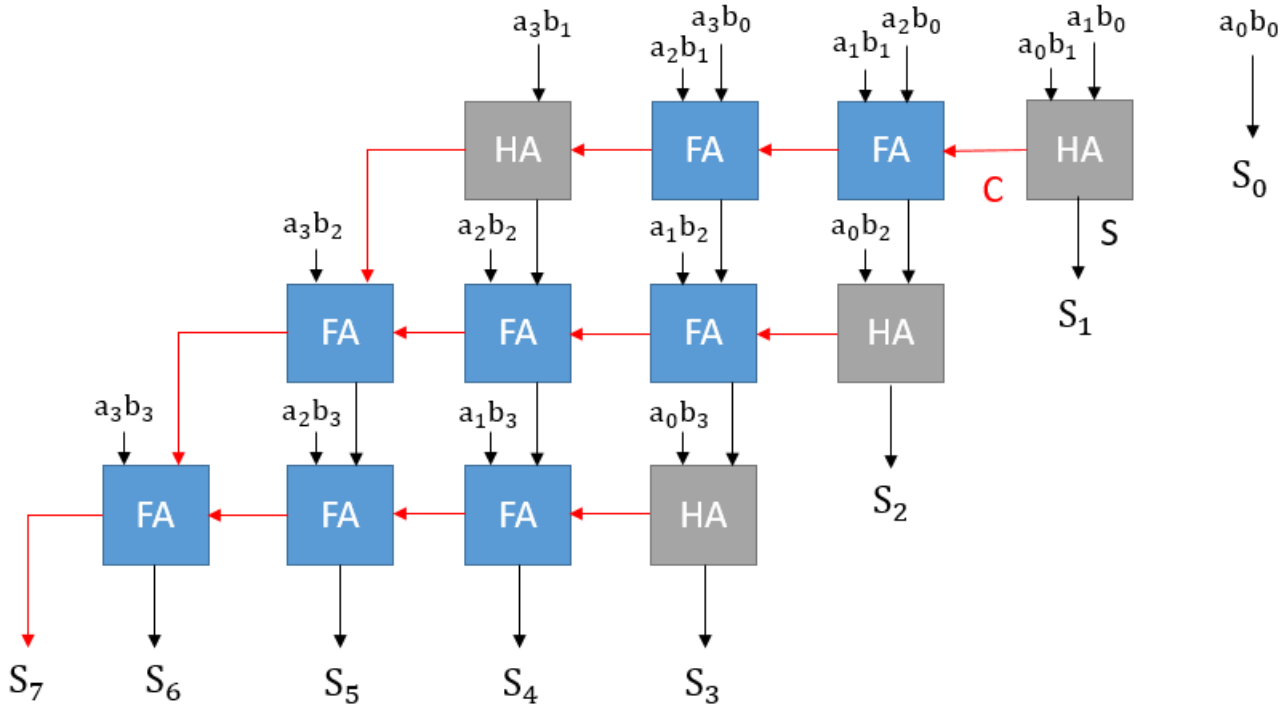
直式乘法範例

$$
\begin{array}{cccccccc}
 & & & a_3 & a_2 & a_1 & a_0 & \\
 & & \times & b_3 & b_2 & b_1 & b_0 & \\
\hline
 & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & \\
 & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & \\
 & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & \\
a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & & \\
\hline
S_7 & S_6 & S_5 & S_4 & S_3 & S_2 & S_1 & S_0
\end{array}
$$

直式乘法示意圖

RCA Array Multiplier

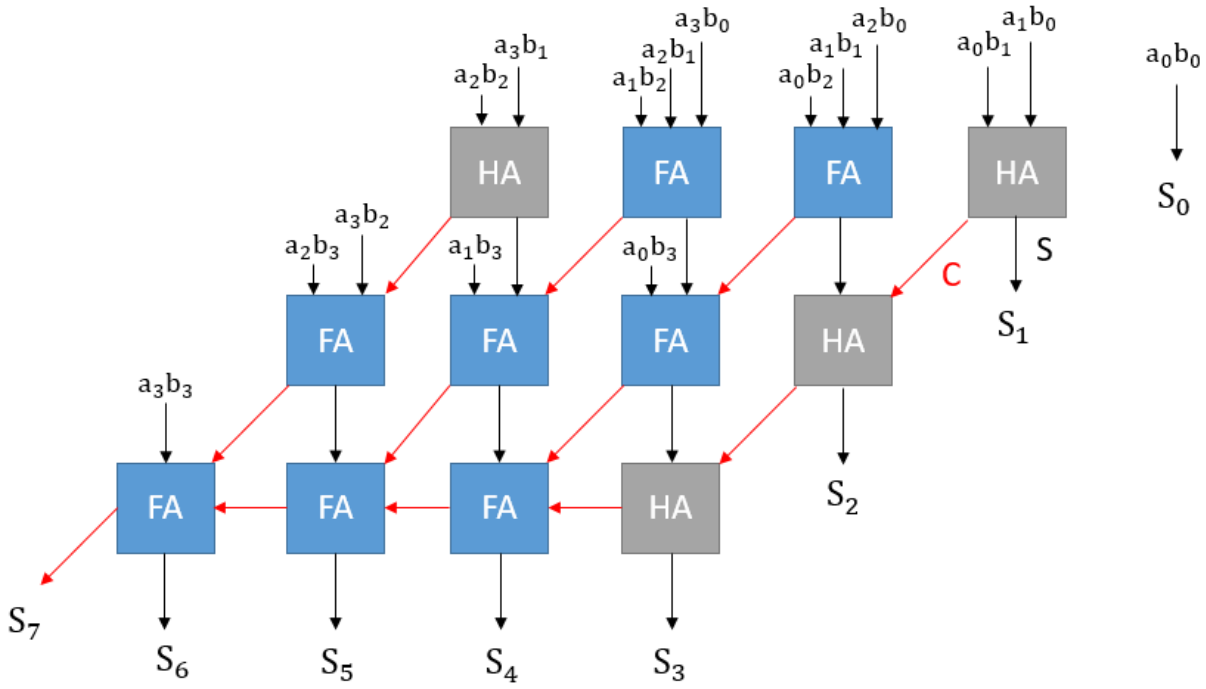
- 對於m bit * n bit的array multiplier將需要：(1) m*n 個and (2) n個HA (3) m*n – m – n個FA
- 這邊以4bit MPY 為範例，總共需要16個and，4個HA，8個FA來實現



4bit RCA Array Multiplier架構

CSA Array Multiplier

- 將RCA array multiplier的進位連接至斜下角的加法器，即CSA array multiplier
- 將carry與sum分別計算，不必計算該層的carry，省去了RCA需要carry傳遞的部分



4bit CSA Array Multiplier架構

Signed Handling

■ 若是有號數運算要使用Array Multiplier的方法時，需進行以下步驟：

1. 乘數或被乘數的sign bit 為1(即為負數)時，取二補數後進行乘法運算。

ex. if(a[3] == 1) Multiplicand = ～a + 1;

ex. if(b[3] == 1) Multiplier = ～b + 1;

2. 運算完後，若乘數或被乘數其中一個為負數，則乘積為負，需再將運算結果取二補數轉換。

ex. sign = a[3] ^ b[3];

if(sign == 1) Product = ～Product + 1;

Sign Extension(1/2)

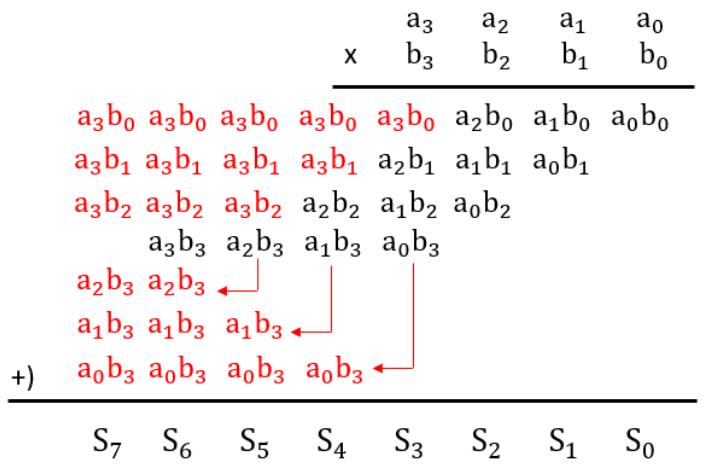
- 在4-bit Array Multiplier直接進行有號數乘法時，運算結果不正確，其原因為有號數的MSB(最高位元)代表正負號。若其為負(MSB=1)，則代表中間計算結果也會是負值，在後續進行加法後會得到錯誤的結果。需進行sign extension得到正確的結果。
- 下頁投影片介紹兩種sign extension的方法

```
        1 1 0 1   A = -3
      x) 1 1 0 0   B = -4
-------------------
        0 0 0 0
      0 0 0 0
    1 1 0 1
+)  1 1 0 1
-------------------
  1 0 0 1 1 1 0 0   A*B = -100 (X)
```

沒有進行sign extension範例

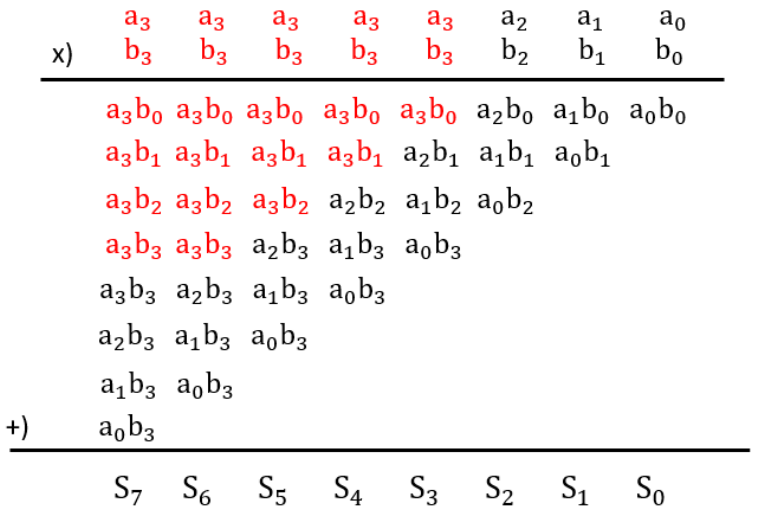
Sign Extension(2/2)

方法一:將乘到MSB的值進行sign extension，若為0則補0至8bit，若為1則補1至8bit。




```
          1 1 0 1   A = -3
       x) 1 1 0 0   B = -4
  ---------------
    0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0
    1 1 1 1 0 1
      1 1 0 1
    1 1
    0 0 0
+)  1 1 1 1
  ---------------
    0 0 0 0 1 1 0 0   A*B = 12 (V)
```

方法二:將被乘數與乘數的MSB進行sign extension到乘積所需bit數(e.g. 4bit * 4bit = 8bit)後進行運算。在與乘數每bit相乘至8bit時即截斷。




```
          1 1 1 1 1 1 0 1   A = -3
       x) 1 1 1 1 1 1 0 0   B = -4
  ---------------
    0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0
    1 1 1 1 0 1
    1 1 1 0 1
    1 1 0 1
    1 0 1
    0 1
+)  1
  ---------------
    0 0 0 0 1 1 0 0   A*B = 12 (V)
```

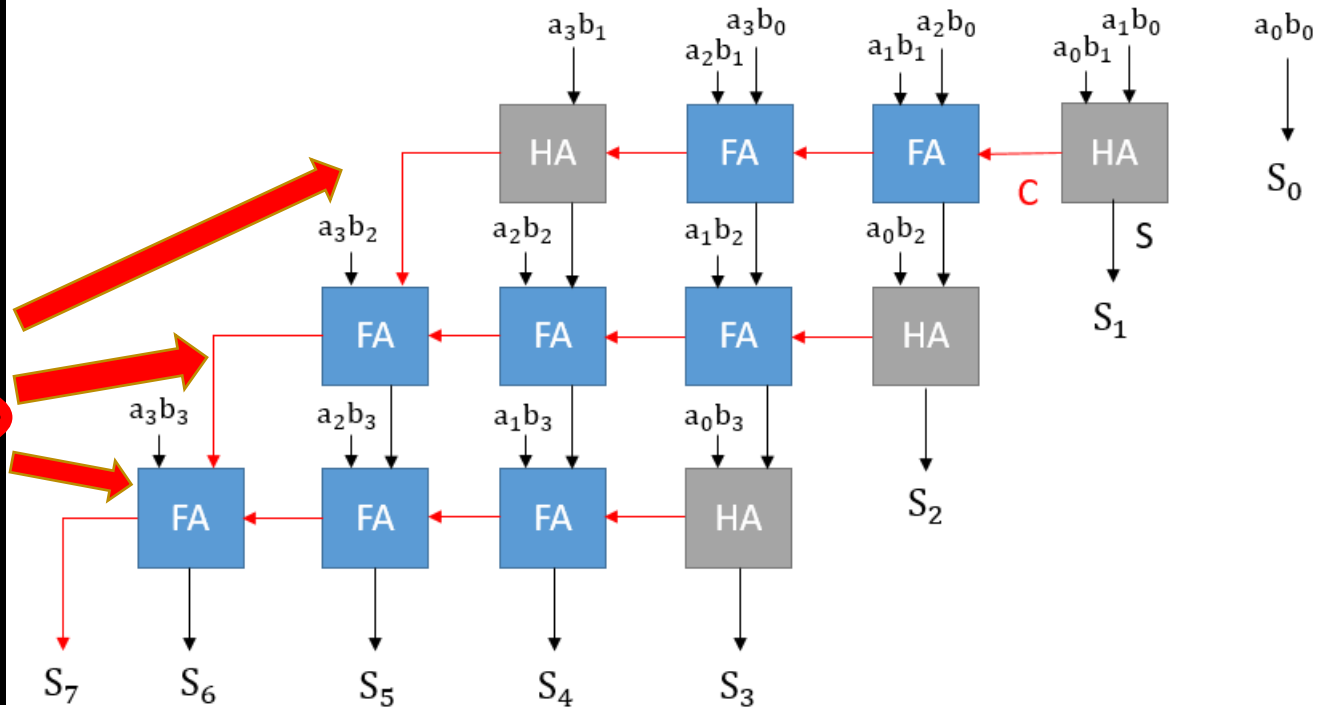
教材說明

教材內容

- 4bit RCA Array Multiplier及對應的testbench
- 4bit CSA Array Multiplier及對應的testbench
- 4bit Sign Extension Multiplier(方法一)及對應的testbench
- 4bit Sign Extension Multiplier(方法二)及對應的testbench

教材說明 – 4bit RCA Array Multiplier

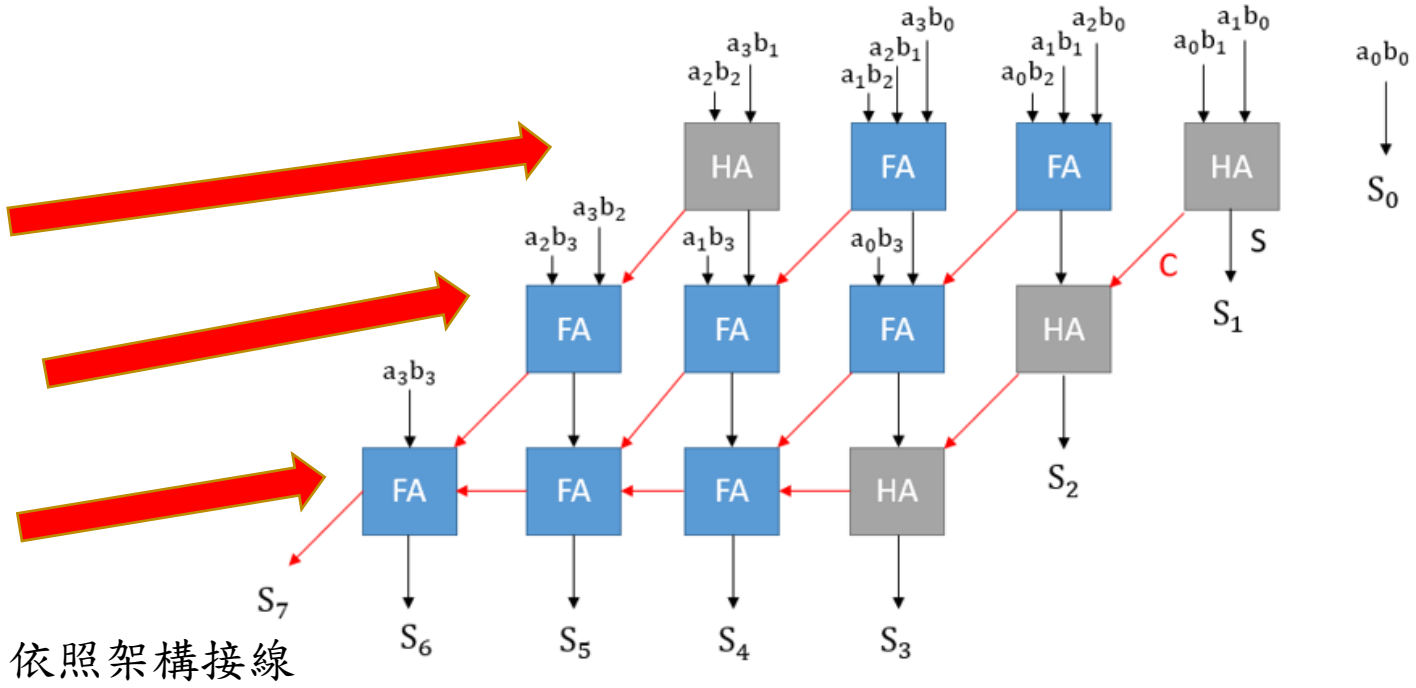
```
module MPY(clk, a, b, p);
    input clk;
    input [3:0] a, b;
    output [7:0] p;
    wire [3:0] ab0, ab1, ab2, ab3;
    wire [2:0] carry;
    wire [3:0] s0, s1, s2;

    arrand and0(a, b[0], ab0);
    arrand and1(a, b[1], ab1);
    arrand and2(a, b[2], ab2);
    arrand and3(a, b[3], ab3);

    HA2FA2 HA2FA2_u1(clk, ab1, ab0[3:1], carry[0], s0);
    HA1FA3 HA1FA3_u1(clk, ab2, {carry[0],s0[3:1]}, carry[1], s1);
    HA1FA3 HA1FA3_u2(clk, ab3, {carry[1],s1[3:1]}, carry[2], s2);

    assign p[0] = ab0[0];
    assign p[1] = s0[0];
    assign p[2] = s1[0];
    assign p[6:3] = s2;
    assign p[7] = carry[2];
endmodule
```



依照架構接線

教材說明 – 4bit CSA Array Multiplier

```
module MPY(clk, a, b, p);
    input clk;
    input [3:0] a, b;
    output [7:0] p;
    wire [3:0] ab0, ab1, ab2, ab3;
    wire [3:0] carry0, carry1, carry2;
    wire [3:0] s0, s1, s2;

    arrand and0(a, b[0], ab0);
    arrand and1(a, b[1], ab1);
    arrand and2(a, b[2], ab2);
    arrand and3(a, b[3], ab3);

    HA HA1_1(clk, ab1[0], ab0[1], s0[0], carry0[0]);
    FA FA1_1(clk, ab2[0], ab1[1], ab0[2], s0[1], carry0[1]);
    FA FA1_2(clk, ab2[1], ab1[2], ab0[3], s0[2], carry0[2]);
    HA HA1_2(clk, ab2[2], ab1[3], s0[3], carry0[3]);

    HA HA2_1(clk, s0[1], carry0[0], s1[0], carry1[0]);
    FA FA2_1(clk, ab3[0], s0[2], carry0[1], s1[1], carry1[1]);
    FA FA2_2(clk, ab3[1], s0[3], carry0[2], s1[2], carry1[2]);
    FA FA2_3(clk, ab3[2], ab2[3], carry0[3], s1[3], carry1[3]);

    HA HA3_1(clk, s1[1], carry1[0], s2[0], carry2[0]);
    FA FA3_1(clk, s1[2], carry1[1], carry2[0], s2[1], carry2[1]);
    FA FA3_2(clk, s1[3], carry1[2], carry2[1], s2[2], carry2[2]);
    FA FA3_3(clk, ab3[3], carry1[3], carry2[2], s2[3], carry2[3]);

    assign p[0] = ab0[0];
    assign p[1] = s0[0];
    assign p[2] = s1[0];
    assign p[6:3] = s2[3:0];
    assign p[7] = carry2[3];

endmodule
```

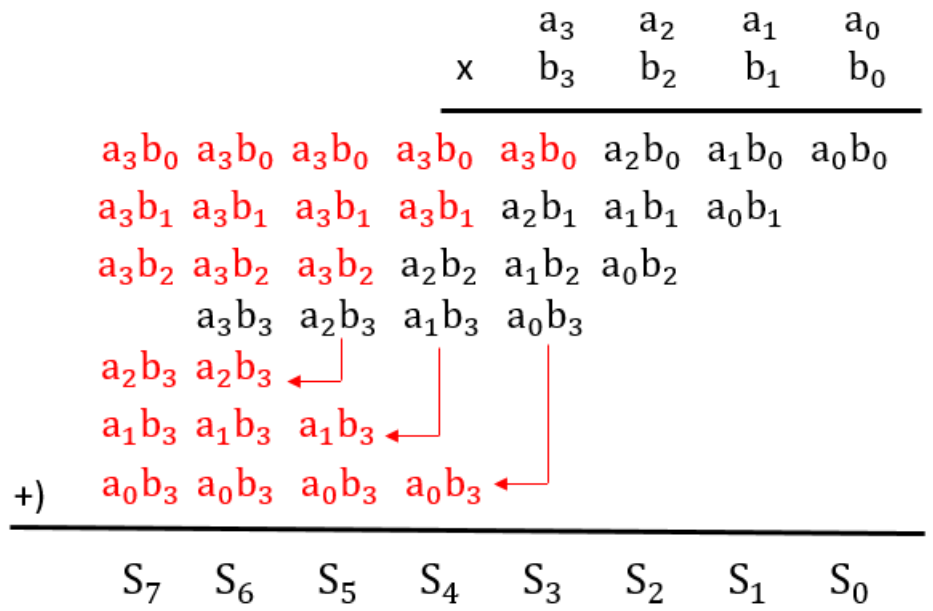


依照架構接線

教材說明 – Sign Extension(方法一)

```
module MPY(clk, a, b, product);
    input clk;
    input [3:0] a, b;
    wire [3:0] ab0, ab1, ab2, ab3;
    wire [7:0] add0, add1, add2, add3;
    wire [7:0] ext0, ext1, ext2, ext3;
    wire [7:0] sum0, sum1, sum2, sum3, sum4;
    output [7:0] product;

    arrand and0(a, b[0], ab0);
    arrand and1(a, b[1], ab1);
    arrand and2(a, b[2], ab2);
    arrand and3(a, b[3], ab3);

    assign add0 = {{4{ab0[3]}}, ab0};
    assign add1 = {{3{ab1[3]}}, ab1, 1'b0};
    assign add2 = {{2{ab2[3]}}, ab2, 2'b0};
    assign add3 = {1'b0, ab3, 3'b0};
    assign ext0 = {{2{ab3[2]}}, 6'b0};
    assign ext1 = {{3{ab3[1]}}, 5'b0};
    assign ext2 = {{4{ab3[0]}}, 4'b0};

    adder adder1(clk,add0,add1,sum0);
    adder adder2(clk,sum0,add2,sum1);
    adder adder3(clk,sum1,add3,sum2);
    adder adder4(clk,sum2,ext0,sum3);
    adder adder5(clk,sum3,ext1,sum4);
    adder adder6(clk,sum4,ext2,product);

endmodule
```

將乘到MSB的bit
sign extension至8bit



教材說明 – Sign Extension(方法二)

```
module MPY(clk, a, b, product);
    input clk;
    input [3:0] a, b;
    wire [7:0] ab0, ab1, ab2, ab3, ab4, ab5, ab6, ab7;
    // wire [7:0] add0, add1, add2, add3;
    // wire [7:0] ext0, ext1, ext2, ext3;
    wire [7:0] sum0, sum1, sum2, sum3, sum4, sum5;
    wire [7:0] ext_a, ext_b;
    output [7:0] product;

    assign ext_a = {{4{a[3]}}, a};
    assign ext_b = {{4{b[3]}}, b};

    arrand and0(ext_a, ext_b[0], ab0);
    arrand and1(ext_a, ext_b[1], ab1);
    arrand and2(ext_a, ext_b[2], ab2);
    arrand and3(ext_a, ext_b[3], ab3);
    arrand and4(ext_a, ext_b[4], ab4);
    arrand and5(ext_a, ext_b[5], ab5);
    arrand and6(ext_a, ext_b[6], ab6);
    arrand and7(ext_a, ext_b[7], ab7);

    adder adder1(clk, ab0, {ab1[6:0], 1'b0}, sum0);
    adder adder2(clk, sum0, {ab2[5:0], 2'b0}, sum1);
    adder adder3(clk, sum1, {ab3[4:0], 3'b0}, sum2);
    adder adder4(clk, sum2, {ab4[3:0], 4'b0}, sum3);
    adder adder5(clk, sum3, {ab5[2:0], 5'b0}, sum4);
    adder adder6(clk, sum4, {ab6[1:0], 6'b0}, sum5);
    adder adder7(clk, sum5, {ab7[0], 7'b0}, product);

endmodule
```

被乘數及乘數MSB sign extension至8bit

被乘數與乘數每bit進行乘積至8bit即截斷

	a_3	a_3	a_3	a_3	a_3	a_2	a_1	a_0
x)	b_3	b_3	b_3	b_3	b_3	b_2	b_1	b_0
	a_3b_0	a_3b_0	a_3b_0	a_3b_0	a_3b_0	a_2b_0	a_1b_0	a_0b_0
	a_3b_1	a_3b_1	a_3b_1	a_3b_1	a_2b_1	a_1b_1	a_0b_1	
	a_3b_2	a_3b_2	a_3b_2	a_2b_2	a_1b_2	a_0b_2		
	a_3b_3	a_3b_3	a_2b_3	a_1b_3	a_0b_3			
	a_3b_3	a_2b_3	a_1b_3	a_0b_3				
	a_2b_3	a_1b_3	a_0b_3					
	a_1b_3	a_0b_3						
+)	a_0b_3							
	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

課堂練習

- 使用RCA Array Multiplier設計一8bit無號乘法器

1. 架構上第一列需更改成2HA、6FA，其他列為1HA、7FA，並依照架構接線

2. 依照架構接線找到對應的output

3. 使用所提供的testbench驗證功能是否正確

1.

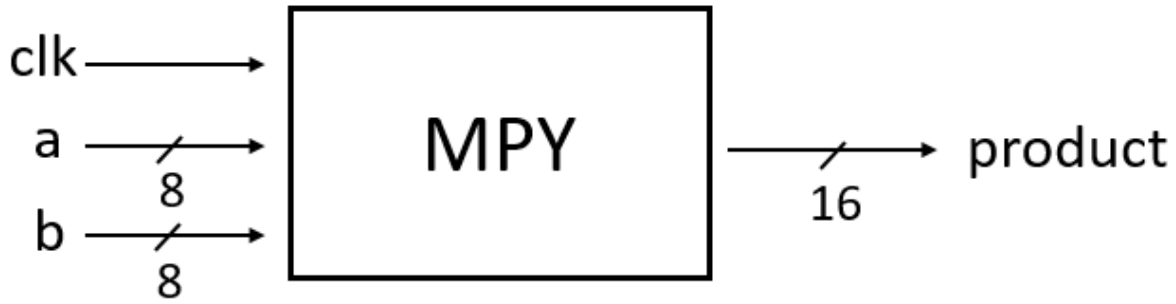
```
HA2FA6 HA2FA6_u1(clk, ab1, ab0[7:1], carry[0], s0);
HA1FA7 HA1FA7_u1(clk, ab2, {carry[0],s0[7:1]}, carry[1], s1);
HA1FA7 HA1FA7_u2(clk, ab3, {carry[1],s1[7:1]}, carry[2], s2);
HA1FA7 HA1FA7_u3(clk, ab4, {carry[2],s2[7:1]}, carry[3], s3);
HA1FA7 HA1FA7_u4(clk, ab5, {carry[3],s3[7:1]}, carry[4], s4);
HA1FA7 HA1FA7_u5(clk, ab6, {carry[4],s4[7:1]}, carry[5], s5);
HA1FA7 HA1FA7_u6(clk, ab7, {carry[5],s5[7:1]}, carry[6], s6);
```

2.

```
assign p[0] = ab0[0];
assign p[1] = s0[0];
assign p[2] = s1[0];
assign p[3] = s2[0];
assign p[4] = s3[0];
assign p[5] = s4[0];
assign p[6] = s5[0];
assign p[14:7] = s6;
assign p[15] = carry[6];
```

LAB作業

使用sign extension其一方法，設計一8bit有號乘法器，並使用所提供的testbench驗證結果是否正確。(驗證結果全對才算對，100%)



作業繳交&demo事項

- 作業繳交時間
  - 2021/6/2(三) 23:59前，上傳至eCourse2
- 作業上傳檔案
  1. sign_ext_mpy_8bit.v
  2. testbench.v
- Demo時間
  - 於社團日後公告
- Demo地點
  - 資工館501A實驗室
- 可在Demo時間前5分鐘至Demo地點準備
- 請攜帶作業相關檔案(使用隨身碟，不要使用雲端)
- 可使用自己的筆電demo