Lecture : Binary Multipliers

(EE5037 : VLSI Circuits for Signal Processing)

Presented by:

Dr. Vikramkumar Pudi
Assistant Professor
Electrical Engineering Department
IIT Tirupati

Binary Unsigned Array Multiplier Design

$N \times M$ **Binary Unsigned Multiplier** will multiply two binary numbers **A** and **B** of size N-bits and M-bits respectively, and produce the output **S** of size $N+M$-bits

$$A = a_3 a_2 a_1 a_0 \qquad B = b_3 b_2 b_1 b_0$$

$$S = A \times B = s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0$$

$$\begin{array}{cccccccc}
 & & & a_3a_2a_1a_0 & \times & b_3b_2b_1b_0 & & \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}$$

भारतीय प्रौद्योगिकी संस्थान तिरुपति

TIRUPATI

Binary Unsigned Array Multiplier Design

$\boldsymbol{N \times M}$ **Binary Unsigned Multiplier** will multiply two binary numbers **A** and **B** of size $\boldsymbol{N}$-bits and M-bits respectively, and produce the output **S** of size $N + M$-bits

$$\boxed{\begin{array}{c} A = a_3a_2a_1a_0 \qquad\qquad B = b_3b_2b_1b_0 \\[1em] S = A \times B = s_7s_6s_5s_4s_3s_2s_1s_0 \end{array}}$$

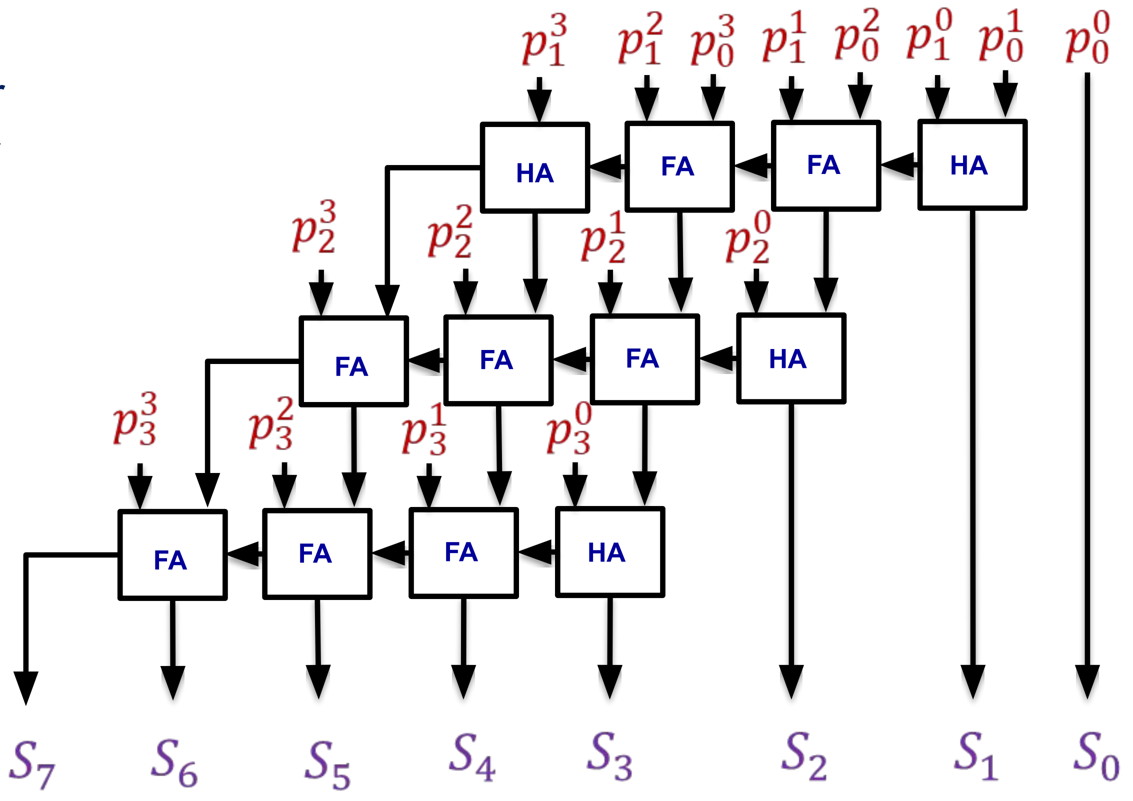
$$\begin{array}{rl}
a_3a_2a_1a_0 \quad \times \quad b_3b_2b_1b_0 & \\
\hline
\boxed{a_3b_0 \quad a_2b_0 \quad a_1b_0 \quad a_0b_0} & P_0 \\
\boxed{a_3b_1 \quad a_2b_1 \quad a_1b_1 \quad a_0b_1} \qquad\qquad & P_1 \\
\boxed{a_3b_2 \quad a_2b_2 \quad a_1b_2 \quad a_0b_2} \qquad\qquad\qquad\qquad & P_2 \\
\boxed{a_3b_3 \quad a_2b_3 \quad a_1b_3 \quad a_0b_3} \qquad\qquad\qquad\qquad\qquad\qquad & P_3 \\
\hline
\end{array}$$

P_0, P_1, P_2 and P_3 are called partial products. The size of each partial product depends upon the size of A . In case of 4×4 multiplier, the size of each partial product is 4-bits. We can define each partial product bits as below

$$P_i^j = a_j b_i \cdot 2^{i+j}$$

Binary Unsigned Array Multiplier Design

HA is *Half Adder*
FA is *Full Adder*



p_1^3 p_1^2 p_0^3 p_1^1 p_0^2 p_1^0 p_0^1 p_0^0
HA FA FA HA
p_2^3 p_2^2 p_2^1 p_2^0
FA FA FA HA
p_3^3 p_3^2 p_3^1 p_3^0
FA FA FA HA
S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0

Writing HDL code for Unsigned Array Multiplier

Generate Partial Products

wire [N-1:0]P[0:M-1]; // N = 4 , M = 4

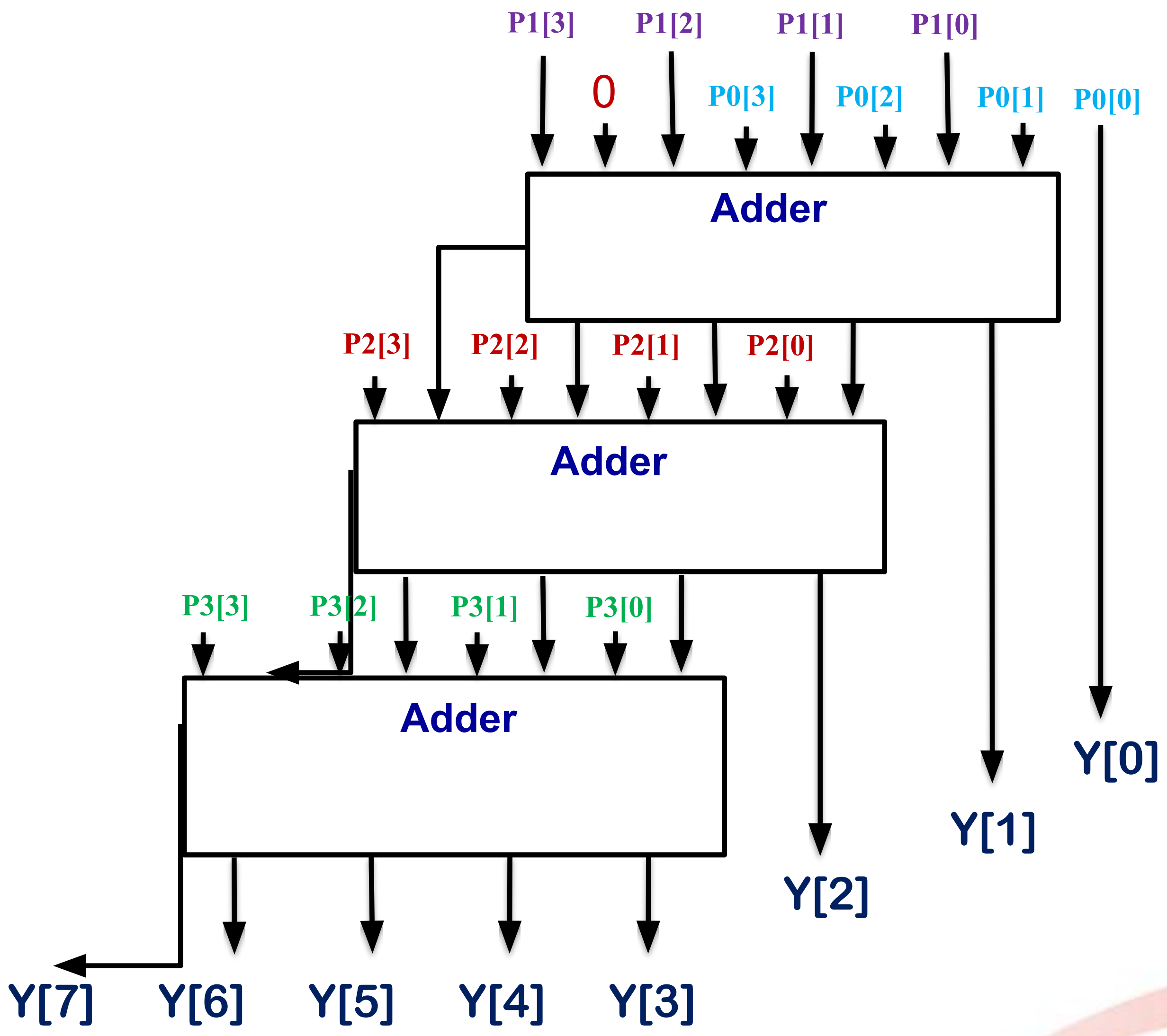
Note : we cannot access the individual bits of **P**.
Because **P** is two dimensional array

```
assign P[0] = B[0]?A:{N{1'b0}};

assign P[1] = B[1]?A:{N{1'b0}};

assign P[2] = B[2]?A:{N{1'b0}};

assign P[3] = B[3]?A:{N{1'b0}};
```



भारतीय प्रौद्योगिकी संस्थान तिरुपति
TIRUPATI

Writing HDL code for Array Multiplier Design

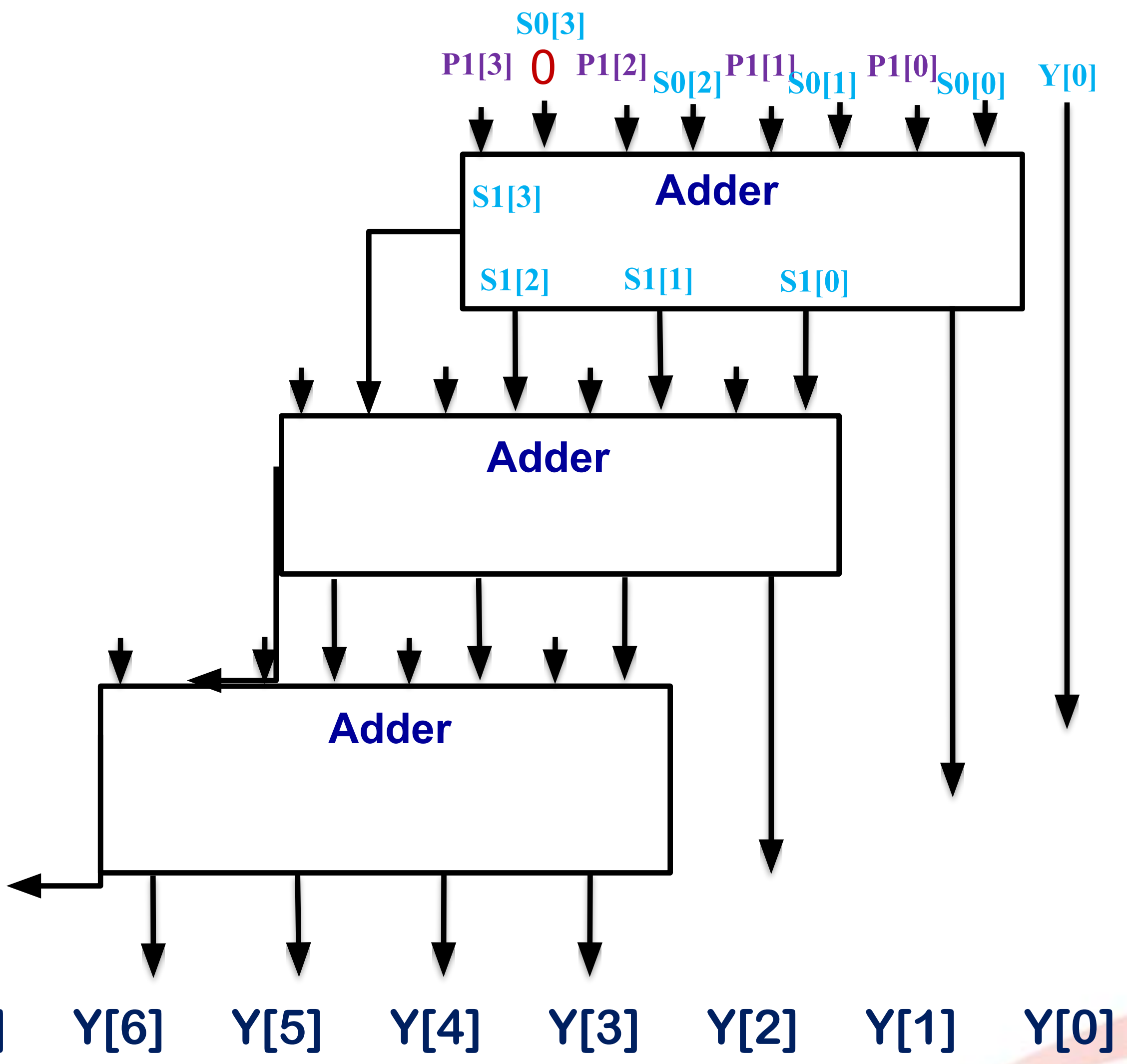
Add Partial Products

```
wire [3:0]S[0:2];
```

```
assign {S[0], Y[0]} = {1'b0, P[0]}
```

Note : we cannot access the individual bits of **S**.
Because **S** is two dimensional array

```
assign {S[1], Y[1]} = P[1] + S[0]
```



Writing HDL code for Array Multiplier Design

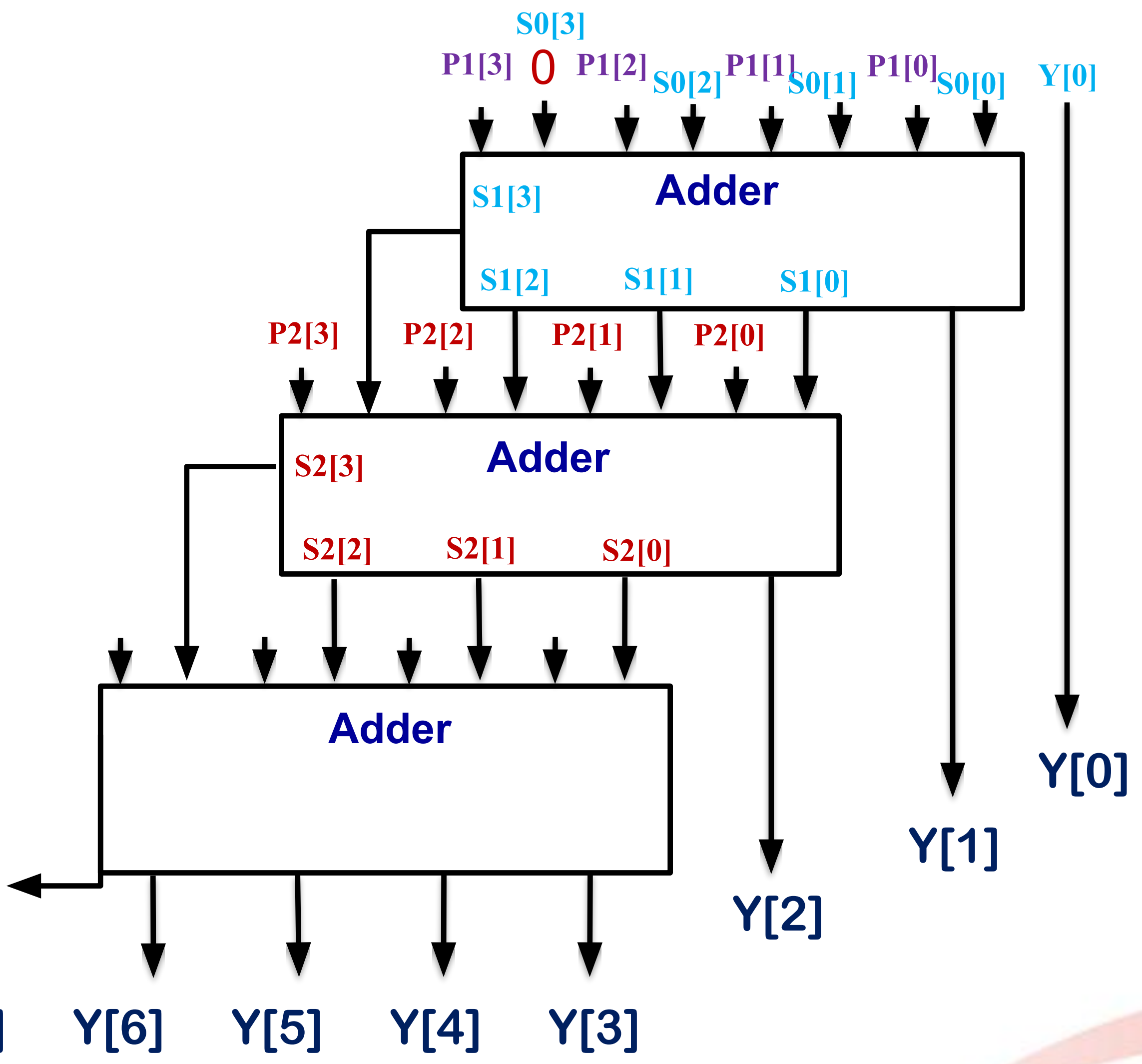
```
wire [3:0]S[0:2];
```

```
assign {S[0], Y[0]} = {1'b0, P[0]}
```

Note : we cannot access the individual bits of **S**.
Because **S** is two dimensional array

```
assign {S[1], Y[1]} = P[1] + S[0]

assign {S[2], Y[2]} = P[2] + S[1]
```



Writing HDL code for Unsigned Array Multiplier

Add Partial Products

```
wire [3:0]S[0:2];
```

Note : we cannot access the individual bits of **S**.
Because **S** is two dimensional array

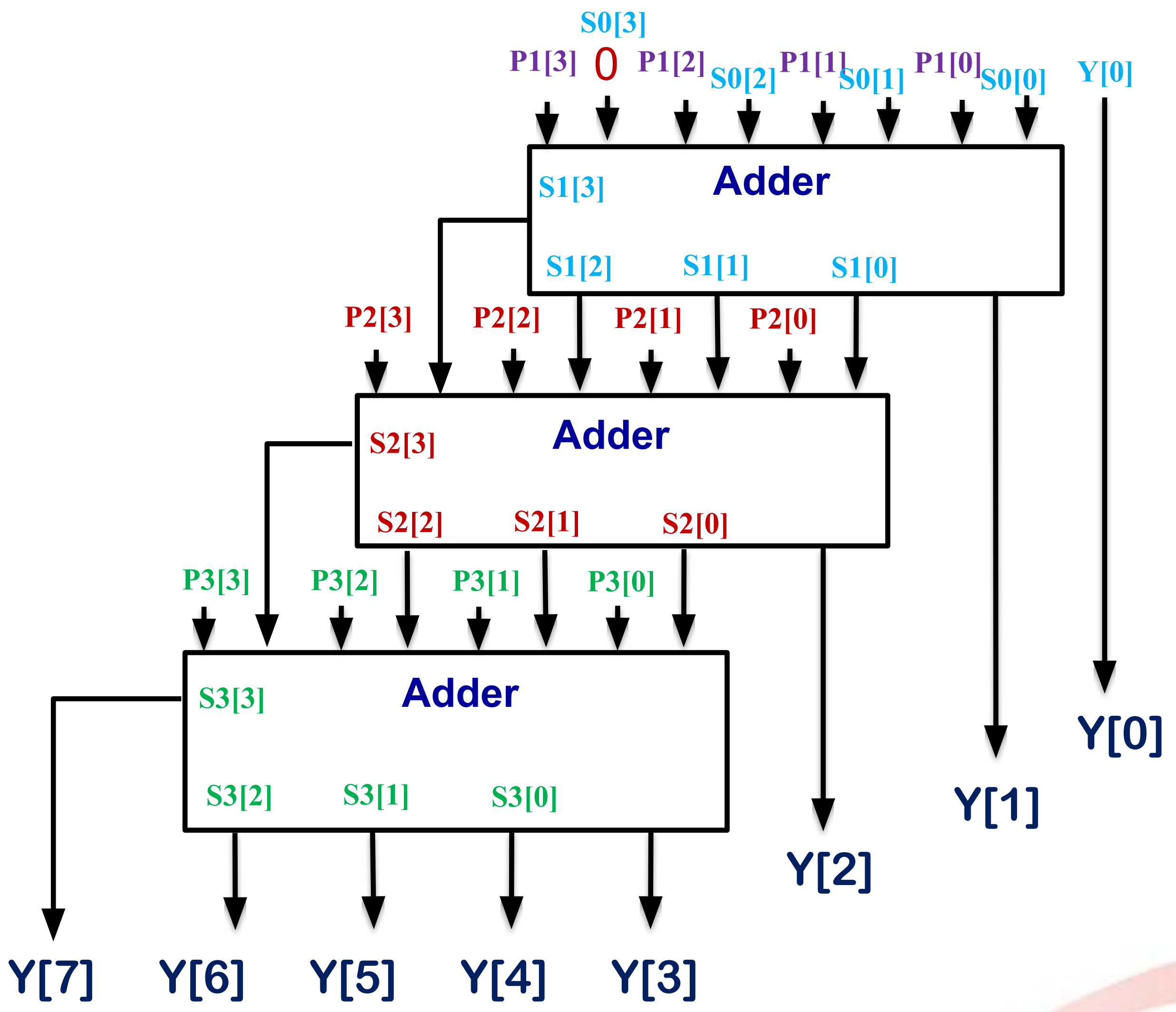
```
assign {S[0], Y[0]} = {1'b0, P[0]}

assign {S[1], Y[1]} = P[1] + S[0]

assign {S[2], Y[2]} = P[2] + S[1]

assign {S[3], Y[3]} = P[3] + S[2]

assign Y[7:4] = S[3]
```



भारतीय प्रौद्योगिकी संस्थान तिरुपति
TIRUPATI

Writing Generic HDL code for Unsigned Array Multiplier

Generate Partial Products

```
wire [N-1:0]P[0:M-1]; // N = 4 , M = 4
```

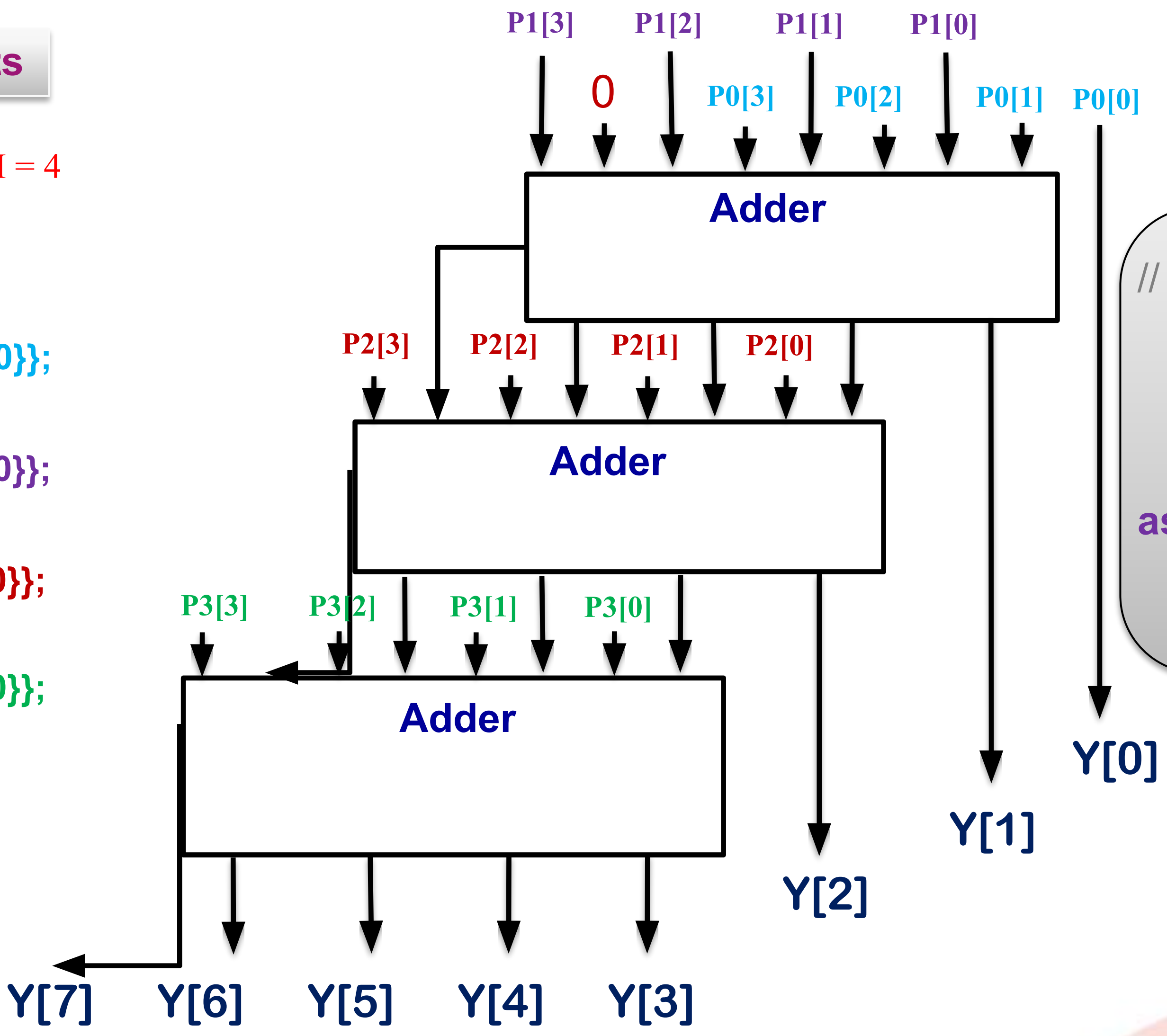
Note : we cannot access the individual bits of **P**.
Because **P** is two dimensional array

```
assign P[0] = B[0]?A:{N{1'b0}};

assign P[1] = B[1]?A:{N{1'b0}};

assign P[2] = B[2]?A:{N{1'b0}};

assign P[3] = B[3]?A:{N{1'b0}};
```



```
// Generate partial products
   genvar i;
   generate
   for(i=0;i<M;i=i+1)
   begin:partial_products_gen
assign P1[i] = B[i] ? A : {(N){1'B0}};
   end
   endgenerate
```

Writing Generic HDL code for Unsigned Array Multiplier

Add Partial Products

```
wire [3:0]S[0:2];
```

Note : we cannot access the individual bits of **S**.
Because **S** is two dimensional array

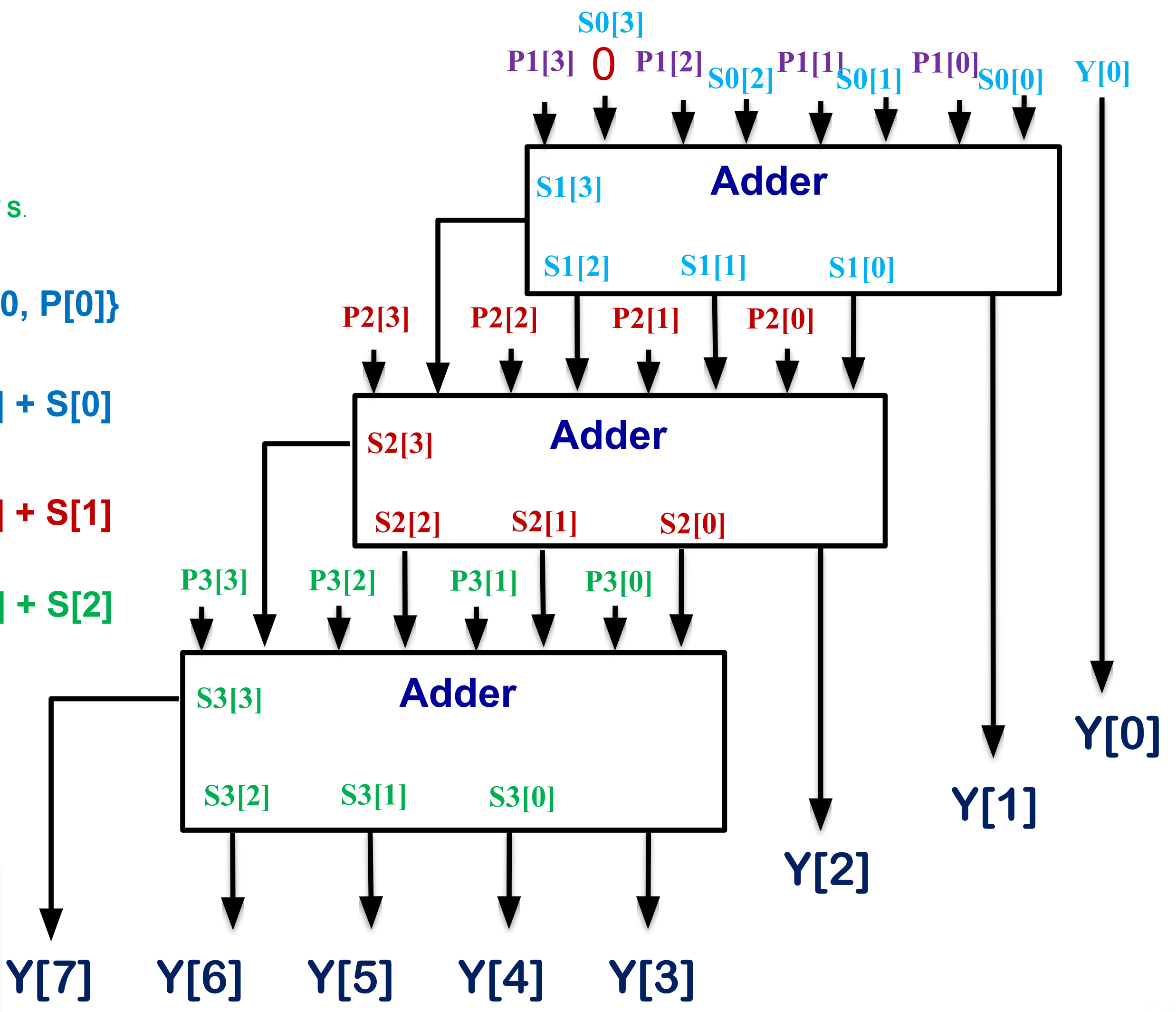
```
assign {S[0], Y[0]} = {1'b0, P[0]}

assign {S[1], Y[1]} = P[1] + S[0]

assign {S[2], Y[2]} = P[2] + S[1]

assign {S[3], Y[3]} = P[3] + S[2]

assign Y[7:4] = S[3]
```



```
assign {S[0], Y[0]} = {1'b0, P[0]}
```

Binary Signed Multiplier Design

$N \times M$ **Binary signed Multiplier** will multiply two binary numbers **A** and **B** of size N-bits and M-bits respectively, and produce the output **S** of size $N + M$-bits

$$A = a_3a_2a_1a_0 \qquad B = b_3b_2b_1b_0$$

$$S = A \times B = s_7s_6s_5s_4s_3s_2s_1s_0$$

$$a_3a_2a_1a_0 \quad \times \quad b_3b_2b_1b_0$$

$$\begin{array}{cccccccc}
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}$$

Binary Signed Multiplier Design (Baugh-Wooley Multiplier)

$N \times M$ **Binary signed Multiplier** will multiply two binary numbers **A** and **B** of size N-bits and M-bits respectively, and produce the output **S** of size $N+M$-bits

$$A = a_3a_2a_1a_0 \qquad B = b_3b_2b_1b_0$$

$$S = A \times B = s_7s_6s_5s_4s_3s_2s_1s_0$$

$$a_3a_2a_1a_0 \quad \times \quad b_3b_2b_1b_0$$

$$\begin{array}{cccccccc}
\hline
 & & & 1 & \overline{\boldsymbol{a_3b_0}} & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & \overline{\boldsymbol{a_3b_1}} & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & \overline{\boldsymbol{a_3b_2}} & a_2b_2 & a_1b_2 & a_0b_2 & & \\
1 & a_3b_3 & \overline{\boldsymbol{a_2b_3}} & \overline{\boldsymbol{a_1b_3}} & \overline{\boldsymbol{a_0b_3}} & & & \\
\hline
s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}$$

Binary Signed Multiplier Design (Baugh-Wooley Multiplier)

$N \times M$ **Binary signed Multiplier** will multiply two binary numbers **A** and **B** of size N-bits and M-bits respectively, and produce the output **S** of size $N + M$-bits

$$A = a_3a_2a_1a_0 \qquad B = b_3b_2b_1b_0$$

$$S = A \times B = s_7s_6s_5s_4s_3s_2s_1s_0$$

1 0 0 1 × 1 0 1 1

$$\begin{array}{cccccccc}
 & & & 1 & 0 & 0 & 0 & 1 \\
 & & & \overline{a_3b_1} & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & \overline{a_3b_2} & a_2b_2 & a_1b_2 & a_0b_2 & & \\
1 & a_3b_3 & \overline{a_2b_3} & \overline{a_1b_3} & \overline{a_0b_3} & & & \\
\hline
s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}$$

Binary Signed Array Multiplier Design (Baugh-Wooley Multiplier)

$N \times M$ **Binary signed Multiplier** will multiply two binary numbers **A** and **B** of size N-bits and M-bits respectively, and produce the output **S** of size $N + M$-bits

$$A = a_3a_2a_1a_0 \qquad B = b_3b_2b_1b_0$$

$$S = A \times B = s_7s_6s_5s_4s_3s_2s_1s_0$$

$$\begin{array}{cccccccc|c}
 & & & a_3a_2a_1a_0 & \times & b_3b_2b_1b_0 & & & \\
\hline
 & & & 1 & \overline{a_3b_0} & a_2b_0 & a_1b_0 & a_0b_0 & P_0 \\
 & & & \overline{a_3b_1} & a_2b_1 & a_1b_1 & a_0b_1 & & P_1 \\
 & & \overline{a_3b_2} & a_2b_2 & a_1b_2 & a_0b_2 & & & P_2 \\
1 & a_3b_3 & \overline{a_2b_3} & \overline{a_1b_3} & \overline{a_0b_3} & & & & P_3 \\
\hline
s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 &
\end{array}$$

भारतीय प्रौद्योगिकी संस्थान तिरुपति

TIRUPATI

Design Single Circuit for Binary Unsigned and Signed Multiplication

Unsigned Multiplication

$$a_3a_2a_1a_0 \quad \times \quad b_3b_2b_1b_0$$

$$\begin{array}{cccccccc}
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0
\end{array}$$

$\boldsymbol{S = 0}$

Signed Multiplication

$$a_3a_2a_1a_0 \quad \times \quad b_3b_2b_1b_0$$

$$\begin{array}{cccccccc}
 & & & 1 & \overline{\boldsymbol{a_3b_0}} & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & \overline{\boldsymbol{a_3b_1}} & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & \overline{\boldsymbol{a_3b_2}} & a_2b_2 & a_1b_2 & a_0b_2 & & \\
1 & a_3b_3 & \overline{\boldsymbol{a_2b_3}} & \overline{\boldsymbol{a_1b_3}} & \overline{\boldsymbol{a_0b_3}} & & & \\
\hline
Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0
\end{array}$$

$\boldsymbol{S = 1}$

Design Single Circuit for Binary Unsigned and Signed Multiplication

Unsigned and Signed Multiplication

$$a_3a_2a_1a_0 \quad \times \quad b_3b_2b_1b_0$$

$$
\begin{array}{cccccccc}
\hline
 & & & \boldsymbol{S} & \boldsymbol{S \oplus a_3b_0} & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & \boldsymbol{S \oplus a_3b_1} & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & \boldsymbol{S \oplus a_3b_2} & a_2b_2 & a_1b_2 & a_0b_2 & & \\
\boldsymbol{S} & a_3b_3 & \boldsymbol{S \oplus a_2b_3} & \boldsymbol{S \oplus a_1b_3} & \boldsymbol{S \oplus a_0b_3} & & & \\
\hline
Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0
\end{array}
$$

Design Single Circuit for Multiplier and Accumulator Unit (MAC)

Unsigned and Signed Multiplication

$$\mathbf{Y = C + A \times B}$$

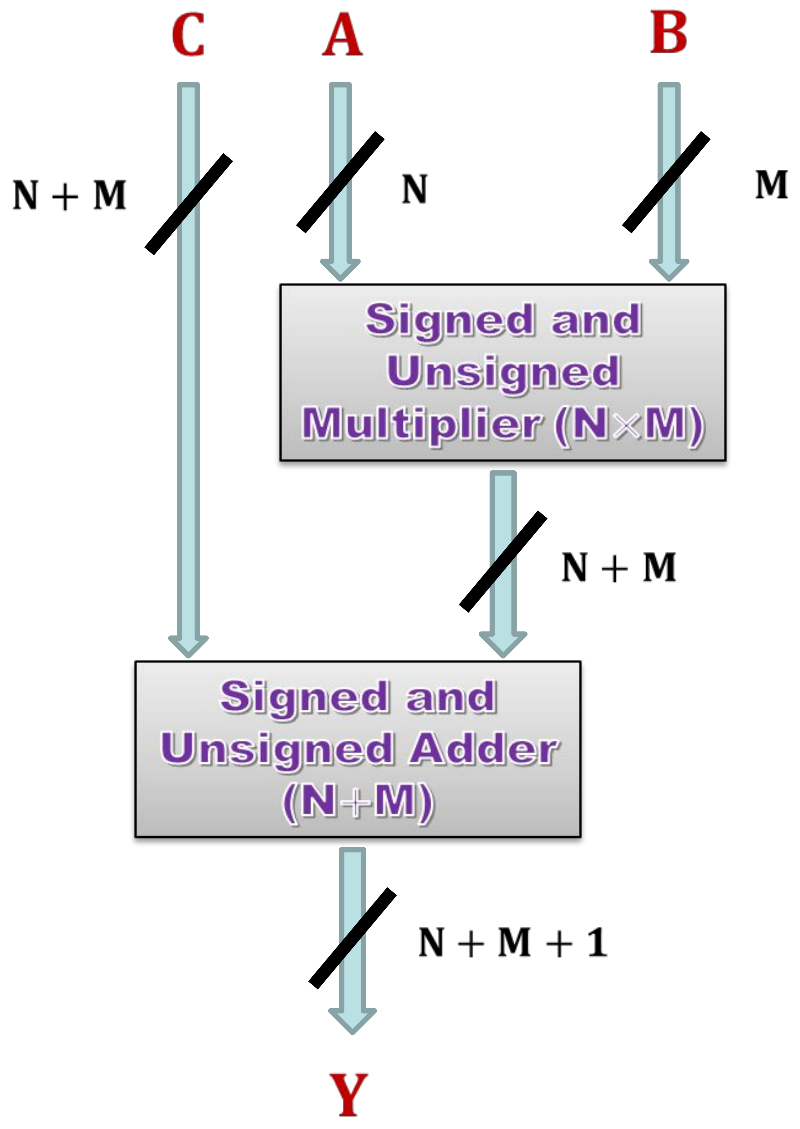
$\mathrm{A\ size\ is\ N - bits}$
$\mathrm{B\ size\ is\ M - bits}$
$\mathrm{C\ size\ is\ N + M - bits}$
$\mathrm{Y\ size\ is\ N + M - bits}$

$$\begin{array}{cccccccc}
 & & & & a_3a_2a_1a_0 & \times & b_3b_2b_1b_0 & \\
\hline
 & & & \boldsymbol{S} & \boldsymbol{S \oplus a_3b_0} & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & \boldsymbol{S \oplus a_3b_1} & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & \boldsymbol{S \oplus a_3b_2} & a_2b_2 & a_1b_2 & a_0b_2 & & \\
\boldsymbol{S} & a_3b_3 & \boldsymbol{S \oplus a_2b_3} & \boldsymbol{S \oplus a_1b_3} & \boldsymbol{S \oplus a_0b_3} & & & \\
\hline
Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0 \\
 & & & + & & & & \\
C_7 & C_6 & C_5 & C_4 & C_3 & C_2 & C_1 & C_0
\end{array}$$

Multiplier and Accumulator Unit (MAC) Architecture

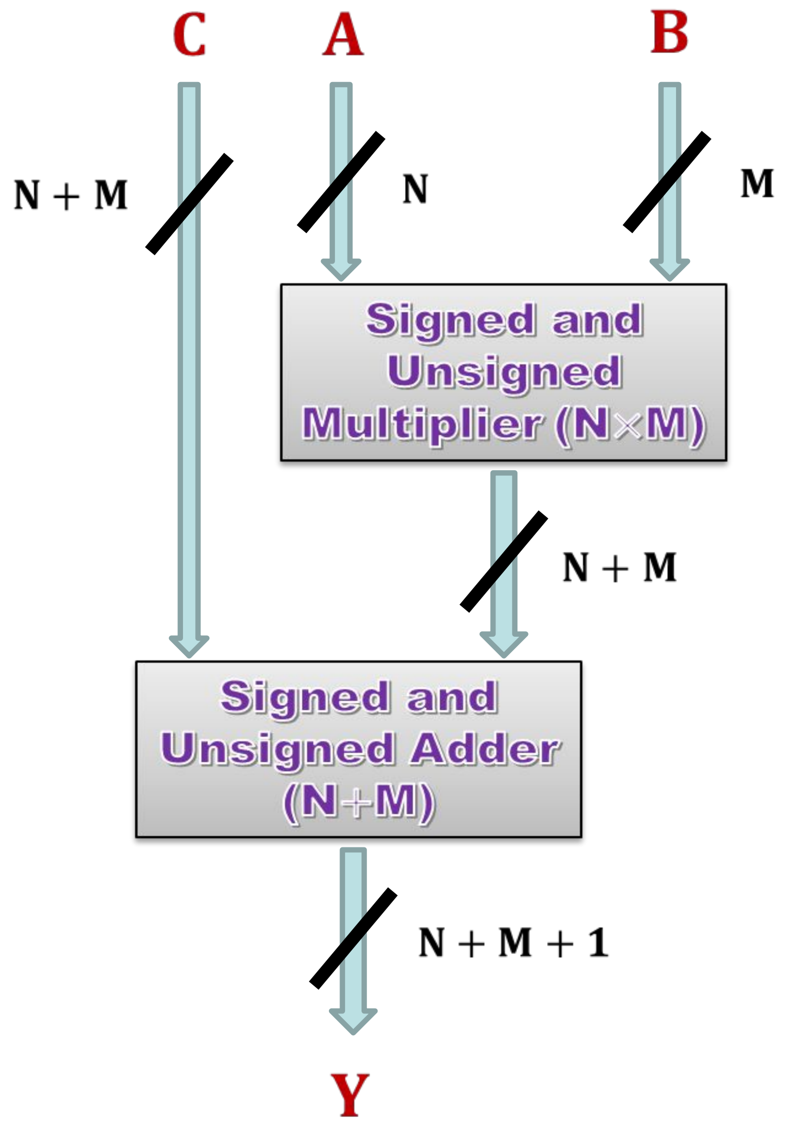
MAC

$$Y = C + A \times B$$



C
A
B
N + M
N
M
Signed and Unsigned Multiplier (N×M)
N + M
Signed and Unsigned Adder (N+M)
N + M + 1
Y

Multiplier and Accumulator Unit (MAC) Architecture


C
A
B
N + M
N
M
Signed and Unsigned Multiplier (N×M)
N + M
Signed and Unsigned Adder (N+M)
N + M + 1
Y

```
module MAC_Array_MUL_Sign #(parameter La=4,Lb=4, Lc = 8, Ly = 9)(A,B,C, sg,Y);
   input [La-1:0]A;
   input [Lb-1:0]B;
   input [Lc-1:0]C; // Lc = La + Lb
   input sg;
   output [Ly:0]Y; // Ly = La+Lb+1

wire [La+Lb-1:0]Ym;

// Calling Multiplier code
 Array_MUL_Sign #(.N(La),.M(Lb)) AMS1 (.A(A),.B(B),.sg(sg),.Y(Ym));

assign Y = {C[Lc-1]&sg,C } + {Ym[La+Lb-1]&sg, Ym};


endmodule
```