Introduction to Arithmetic Operation

Instructor : Pei-Yun Tsai

Unsigned Addition/Subtraction

```
  8        1000
+9        +1001
-----     --------
  1        0001
```

```
  8        01000
+9        +01001
-----     --------
 17        10001
```

```
 14        1110
-   5     - 0101
-----     --------
  9        1001
```

```
  5        0101      0101
- 14     - 1110     +0010
-----    --------   --------
  7                  0111
```

Signed Addition/Subtraction

8	01000
+9	+01001
-----	----------
-15	10001

8	001000
+9	+001001
-----	------------
17	010001

(-5)	11011	11011
- 14	- 01110	+10010
-----	---------	----------
13		01101

(-5)	111011	111011
- 14	- 001110	+110010
------	-----------	-----------
-19		101101

Signed/Unsigned Addition

- If *N*-bit addition/subtraction is required, usually we use
  - Unsigned addition/subtraction
    - **wire** [N-1:0] In1,In2;
    - **wire** [N:0] AddOut;
    - **assign** AddOut={1’b0,In1}+{1’b0,In2};
  - Signed addition/subtraction
    - **wire** [N-1:0] In1,In2;
    - **wire** [N:0] AddOut;
    - **assign** AddOut={In1[N-1],In1}+{In2[N-1],In2};

Unsigned Multiplication

- Multiplicand: $Y = (y_{M-1}, y_{M-2}, \ldots, y_1, y_0)$
- Multiplier: $X = (x_{N-1}, x_{N-2}, \ldots, x_1, x_0)$
- Product: $$P = \left(\sum_{j=0}^{M-1} y_j 2^j\right)\left(\sum_{i=0}^{N-1} x_i 2^i\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

							y_5	y_4	y_3	y_2	y_1	y_0	multiplicand
							x_5	x_4	x_3	x_2	x_1	x_0	multiplier
						x_0y_5	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0	
					x_1y_5	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0		
				x_2y_5	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0			partial products
			x_3y_5	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0				
		x_4y_5	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0					
	x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_5y_1	x_5y_0						
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	product

Two's Complement Multiplication

- A binary two's-complement number is formulated as

$$X = -X_{N-1}2^{N-1} + X_{N-2}2^{N-2} + X_{N-3}2^{N-3} \cdots + X_0 2^0$$

$$= -X_{N-1}2^{N-1} + \sum_{i=0}^{N-2} X_i 2^i$$

- Similarly for *Y* and the product of *X* and *Y* is

$$XY = (-X_{N-1}2^{N-1} + \sum_{i=0}^{N-2} X_i 2^i)(-Y_{N-1}2^{N-1} + \sum_{j=0}^{N-2} Y_j 2^j)$$

$$= X_{N-1}Y_{N-1}2^{2N-2} + \sum_{i=0}^{N-2}\sum_{j=0}^{N-2} X_i Y_j 2^{i+j} - Y_{N-1}\sum_{i=0}^{N-2} X_i 2^{N+i-1} - X_{N-1}\sum_{j=0}^{N-2} Y_j 2^{N+j-1}$$

- The last two terms can both be expressed as

$$-\sum_{i=0}^{N-2} Y_{N-1}X_i 2^{N+i-1} = -2^{2N-2} + (\sum_{i=0}^{N-2}(1-Y_{N-1}X_i)2^{N+i-1}) + 2^{N-1}$$

$$= -2^{2N-2} + (\sum_{i=0}^{N-2}\overline{Y_{N-1}X_i}\,2^{N+i-1}) + 2^{N-1}$$

Two's Complement Array Multiplication (1/2)

							y_5	y_4	y_3	y_2	y_1	y_0
							x_5	x_4	x_3	x_2	x_1	x_0
								x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0
							x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0	
$\sum_{i=0}^{N-2}\sum_{j=0}^{M-2} x_i y_j 2^{i+j}$						x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0		
					x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0			
				x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0				
$x_{N-1} y_{M-1} 2^{M+N-2}$		x_5y_5										
$-\sum_{i=0}^{N-2} x_i y_{M-1} 2^{i+M-1}$	1	1	$\overline{x_4y_5}$	$\overline{x_3y_5}$	$\overline{x_2y_5}$	$\overline{x_1y_5}$	$\overline{x_0y_5}$	1	1	1	1	1
												1
$-\sum_{j=0}^{M-2} x_{N-1} y_j 2^{j+N-1}$	1	1	$\overline{x_5y_4}$	$\overline{x_5y_3}$	$\overline{x_5y_2}$	$\overline{x_5y_1}$	$\overline{x_5y_0}$	1	1	1	1	1
												1
	p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Two’s Complement Array Multiplication (2/2)

- Modified Baugh-Wooly multiplier

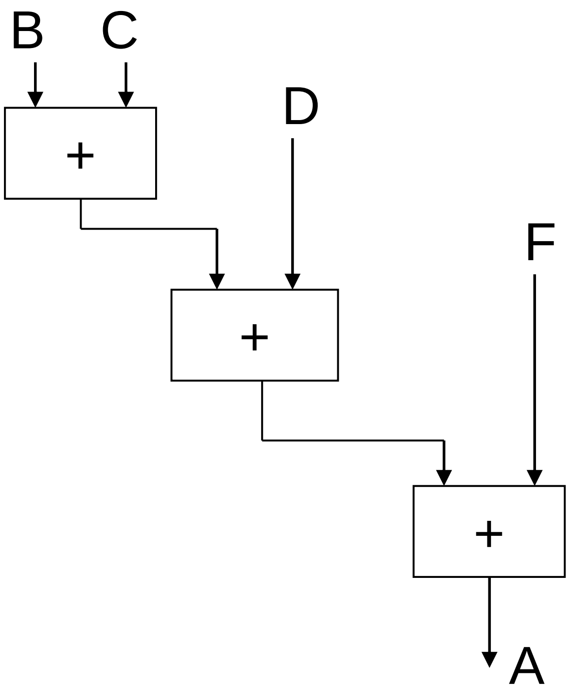
						y_5	y_4	y_3	y_2	y_1	y_0
						x_5	x_4	x_3	x_2	x_1	x_0
					1	$\overline{x_5 y_0}$	$x_0 y_4$	$x_0 y_3$	$x_0 y_2$	$x_0 y_1$	$x_0 y_0$
					$\overline{x_5 y_1}$	$x_1 y_4$	$x_1 y_3$	$x_1 y_2$	$x_1 y_1$	$x_1 y_0$	
				$\overline{x_5 y_2}$	$x_2 y_4$	$x_2 y_3$	$x_2 y_2$	$x_2 y_1$	$x_2 y_0$		
			$\overline{x_5 y_3}$	$x_3 y_4$	$x_3 y_3$	$x_3 y_2$	$x_3 y_1$	$x_3 y_0$			
		$\overline{x_5 y_4}$	$x_4 y_4$	$x_4 y_3$	$x_4 y_2$	$x_4 y_1$	$x_4 y_0$				
1	$x_5 y_5$	$\overline{x_4 y_5}$	$\overline{x_3 y_5}$	$\overline{x_2 y_5}$	$\overline{x_1 y_5}$	$\overline{x_0 y_5}$					
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Signed/Unsigned Multiplication

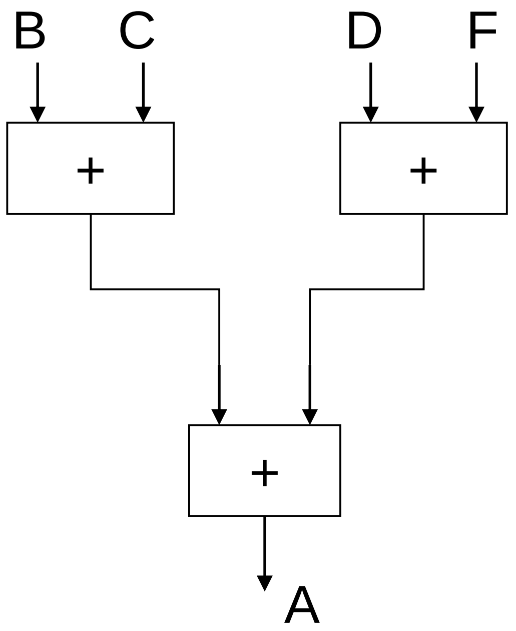
- If *N*-bit multiplication is required, usually we use
  - Unsigned multiplication
    - **wire** [N-1:0] In1,In2;
    - **wire** [2N-1:0] MulOut;
    - **assign** MulOut=In1*In2;
  - Signed multiplication
    - **wire signed** [N-1:0] In1,In2;
    - **wire signed** [2N-1:0] MulOut;
    - **assign** MulOut=In1*In2;

Using Parenthesis

A=B+C+D+F

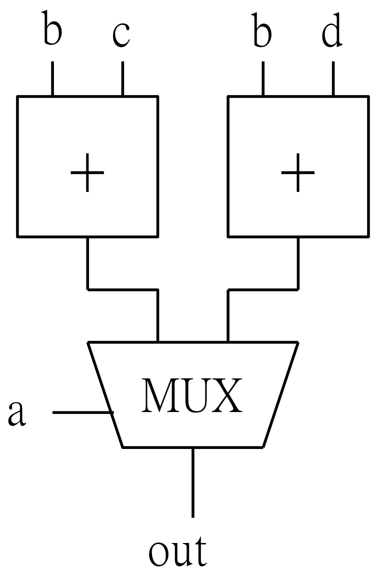


A=(B+C)+(D+F)



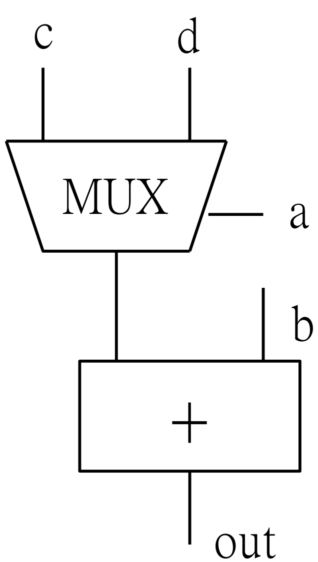
Resource Sharing (1/2)

```
always@(a or b or c or d)
  out =(a) ? (b+c):(b+d);
```



Keep sharable resource in the
- ✓ same conditional statement
- ✓ same always block
- ✓ same module

```
always@(a or b or c or d)
  if (a)  out = b+c;
  else out = b+d ;
```



without resource sharing

with resource sharing

Resource Sharing (2/2)

- The operators that can share resources must be in the mutual exclusive paths

```
if (sel1)  out1=a1+b1;
else begin
  out1 = c1+d1 ;
  if ( sel2) out2=a2+b2;
  else out2=c2+d2;
end
```

operator "c1+d1" and "a2+b2" or "c2+d2" are not in the mutual exclusive paths

```
if (sel1)  out1=a1+b1;
else begin
  out1 = c1+d1 ;
  if ( sel2) out2=a2+b2;
  else out2=c2+d2;
end
```

All operators are in the mutual exclusive paths

Resource sharing

Explicit Resource Sharing

- Original

```
if (F)
  A=B+C;
else
  A=B+20;
```

- Modified

```
if(F)
    T=C;
else
    T=20;
A=B+T;
```