Arithmetic

COMP375 Computer Architecture and Organization

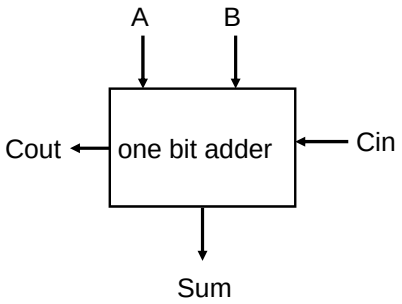
Elementary School

17	010001
+ 7	+000111
24	011000

You add two numbers together. If the sum is greater than the number base, you add one to the next column. When you add two numbers, you may also have to add the carry from the column to the right.

1 Bit Adder

- A one bit adder has three inputs, numbers A and B and Carry in. There are two outputs, the Sum and Carry out.



1 bit Adder Truth Table

Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0+0+0=00_{two}$
0	0	1	0	1	$0+0+1=01_{two}$
0	1	0	0	1	$0+1+0=01_{two}$
0	1	1	1	0	$0+1+1=10_{two}$
1	0	0	0	1	$1+0+0=01_{two}$
1	0	1	1	0	$1+0+1=10_{two}$
1	1	0	1	0	$1+1+0=10_{two}$
1	1	1	1	1	$1+1+1=11_{two}$

figure from textbook

Addition Sum

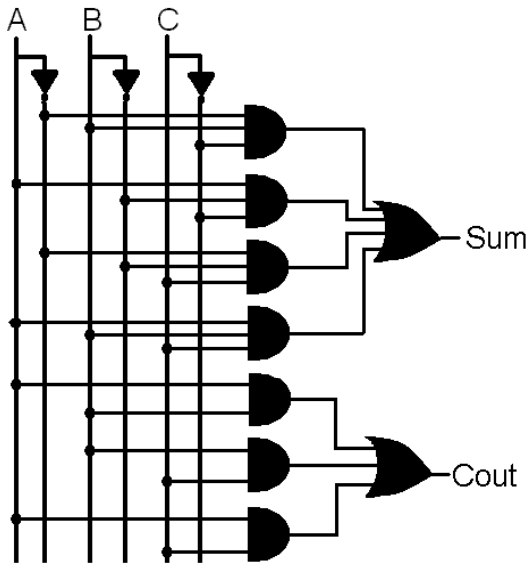
	~A ~B	~A B	A B	A ~B
~Cin	0	1	0	1
Cin	1	0	1	0

Sum = ~AB~C + A~B~C + ~A~BC + ABC

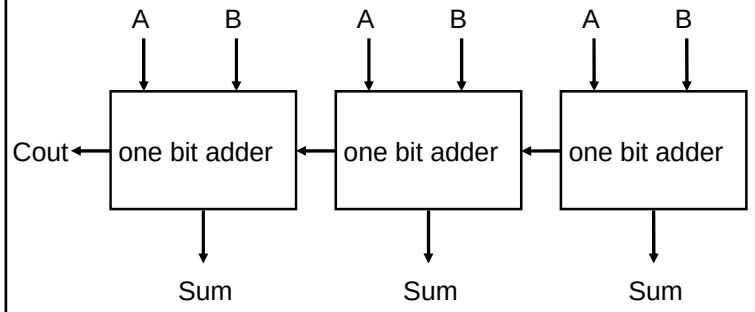
Addition Carry Out

	~A ~B	~A B	A B	A ~B
~Cin	0	0	1	0
Cin	0	1	1	1

Cout = AB + BC + AC



Multiple Bit Adder



Subtraction

- Instead of building a separate subtraction circuit, you can add the negative of the operand.
- To make a twos complement number negative, you must invert the bits and add one
- A NOT gate can be used to invert all the bits.
- Setting the Carry In on the rightmost bit will add one to the result.

A+~B+1 = A + (~B+1) = A + (-B) = A - B

Propagation Delay

- To make a 32 bit adder, you can use 32 one bit adders.
- The left most bit cannot be computed until all of the other bits are computed so that the Carry In value will be known.
- Each one bit adder requires the signal to go through two gates. Each gate takes a small amount of time to react.
- This limits the speed of the adder.

Look Ahead Carry

- The Carry Out is determined by

$$C_{i+1} = A_i*B_i + A_i*C_i + B*C_i$$

$$C_{i+1} = A_i*B_i + C_i*(A_i + B_i)$$

- The A*B term is true when this bit generates a carry out. Call it G_i
- The other term is true when a carry propagates from an earlier bit. Call it P_i

Carry Equations

- We can express the carry as:

$$C_{i+1} = G_i + (P_i * C_i)$$

$$C_{i+2} = G_{i+1} + (P_{i+1} * G_i) + (P_{i+1} * P_i * C_i)$$

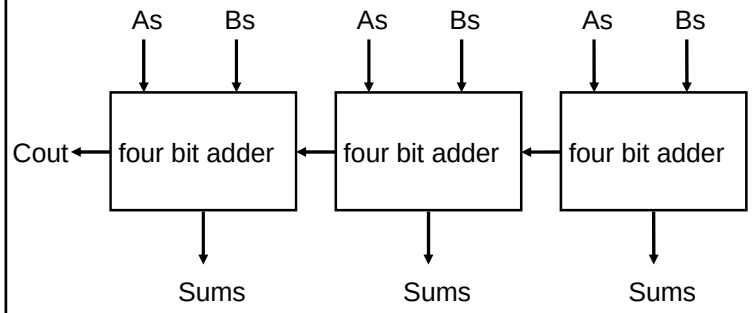
$$C_{i+4} = G_{i+3} + (P_{i+3} * G_{i+2}) + (P_{i+3} * P_{i+2} * G_{i+1}) + (P_{i+3} * P_{i+2} * P_{i+1} * G_i) + (P_{i+3} * P_{i+2} * P_{i+1} * P_i * C_i)$$

Reduced Propagation

- The simple n bit ripple adder took O(2n) time to add n bits due to carry propagation.
- With carry look ahead, it takes O(3) time to propagate the carry. The look ahead requires more circuitry.

Further Simplification

- Creating a big adder out of groups of adders can reduce propagation and circuitry



Multiplication Tables

Decimal

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Binary

	0	1
0	0	0
1	0	1

Multiplication

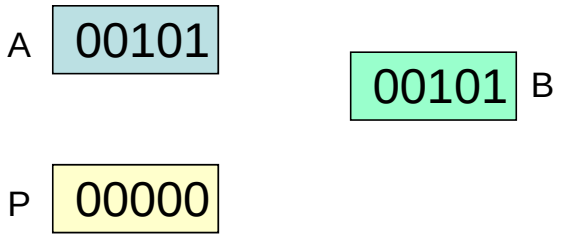
543	110
*312	*101
1086	110
543	000
1629	110
169416	11110
decimal	binary

Add and Shift

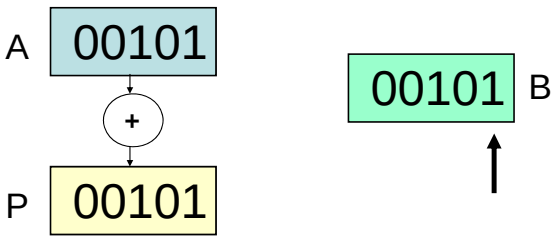
- Multiplication can be done by a series of adds and left shifts.
- Assume you have operands A & B and Product P (initially zero).

```
for each bit i in B {
 if (Bi is one)
    add A to P
 shift A left by one bit
 }
```

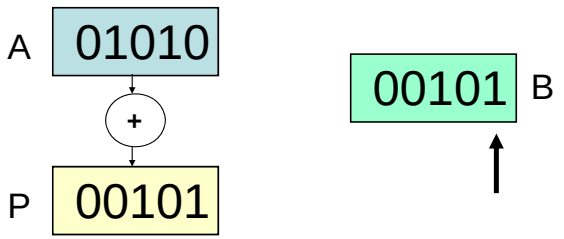
5 x 5 initial setup



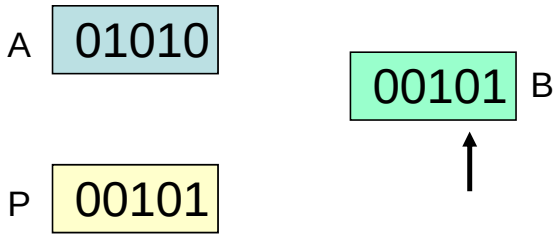
B_0 is 1 so Add



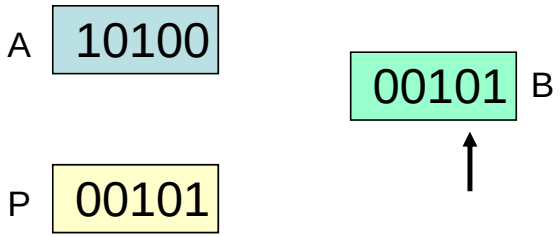
Shift A left one bit



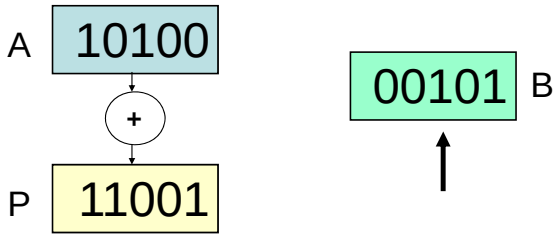
B_1 is zero so do nothing



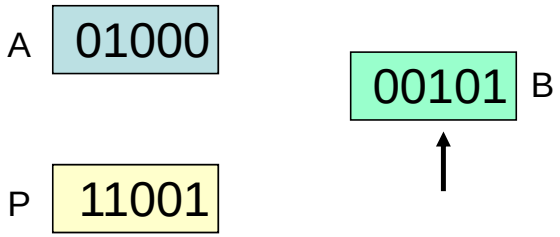
Shift A left one bit



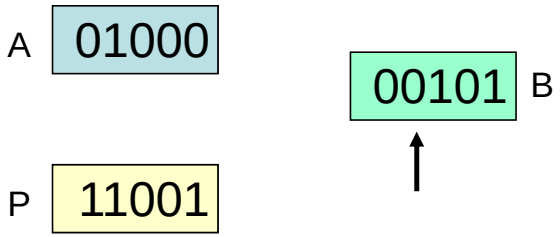
B_2 is 1 so Add



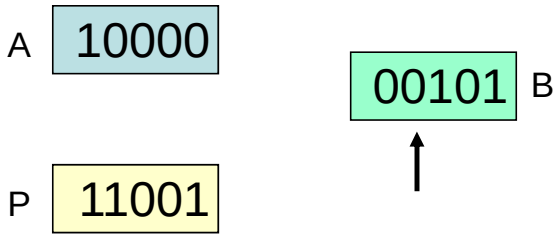
Shift A left one bit



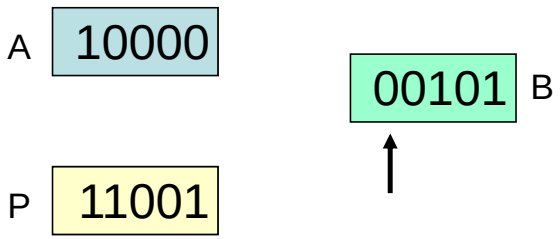
B_3 is zero so do nothing



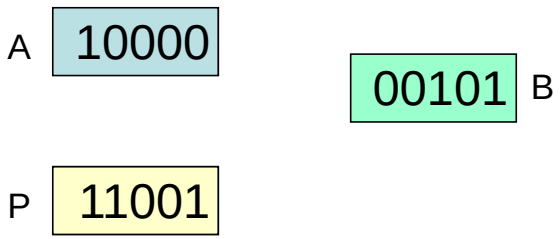
Shift A left one bit



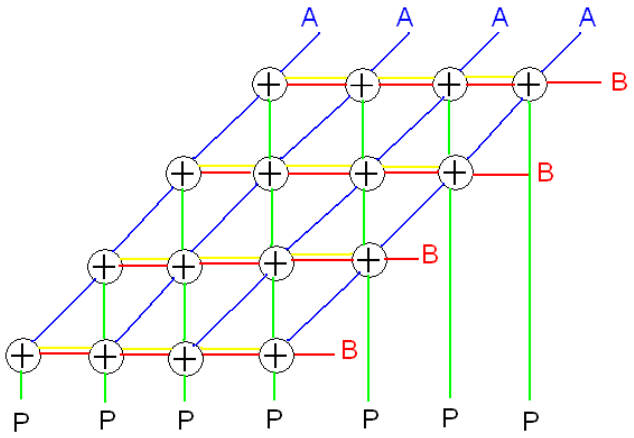
B_4 is zero so do nothing



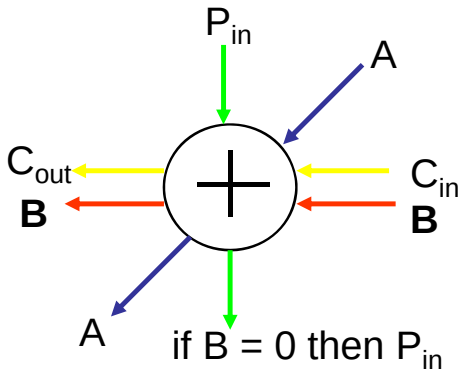
Done



Combinatorial Multiplier



One Bit Multiplier



if B = 0 then P_{in}

if B = 1 then P_{in} + A

Division

- Division is difficult. (Every elementary school child knows that.)
- Some small computers do not have multiplication or division.
- Some have multiplication but not division.