Arithmetic

COMP375 Computer Architecture and Organization

Goal for Today

- Create logic gates that perform arithmetic

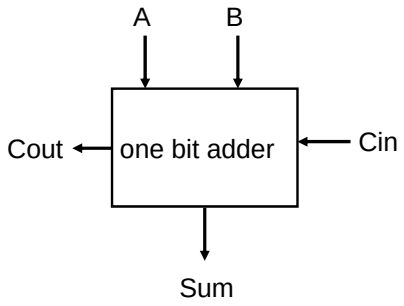
Elementary School

17	010001
+ 7	+000111
24	011000

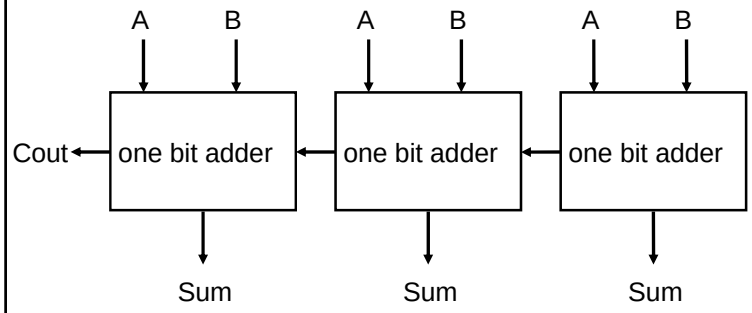
You add two numbers together. If the sum is greater than the number base, you add one to the next column. When you add two numbers, you may also have to add the carry from the column to the right.

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.



Multiple Bit Adder



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

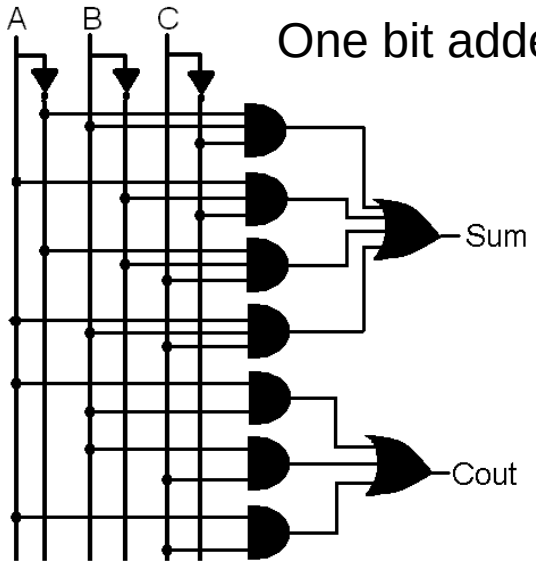
Sum = A'BC' + AB'C' + A'B'C + ABC

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

One bit adder



Overflow

- If operands are too big, then sum cannot be represented as an *n*-bit 2's complement number.

carry into sign bit

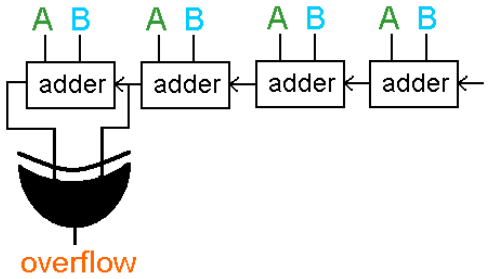
	01000	(8)
+	01001	(9)
	10001	(-15)

carry out of sign bit

	11000	(-8)
+	10111	(-9)
	01111	(+15)

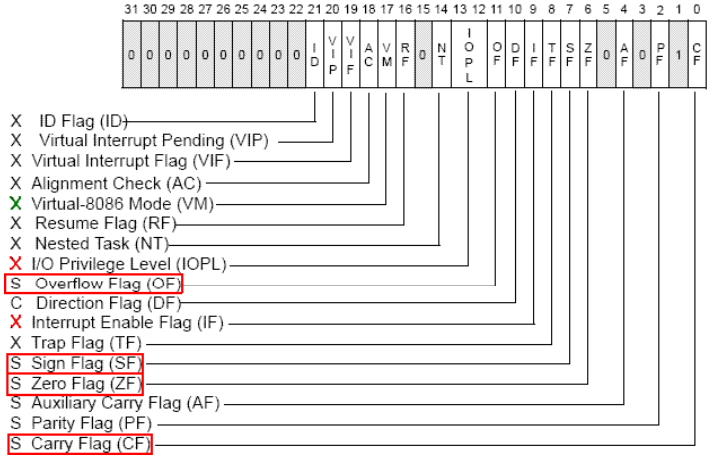
- We have overflow if:
  - signs of both operands are the same, and
  - sign of sum is different.
- Another test -- easy for hardware:
  - carry into left most bit does not equal carry out

Detecting Overflow



- When the carry into the sign bit does not match the carry out, there is an overflow

Intel Status Register



Saving Status

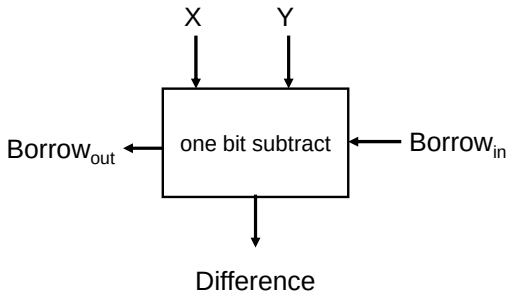
Status flag	Detection
Overflow	XOR of carry into and out of the sign bit
Sign	Copy of the sign bit
Zero	NOR of all result bits
Carry	Carry into the sign bit

Intel Status Register

- The status register records the results of executing the instruction.
- Performing arithmetic sets the status register.
- The compare instruction does a subtraction, but doesn't store the results. It just sets the status flags.
- All jump instructions are based on the status register

1 Bit Subtracter

- A one bit subtracter has three inputs, numbers X and Y and Borrow in. There are two outputs, the Difference and Borrow out.



X-Y Subtracter Truth Table

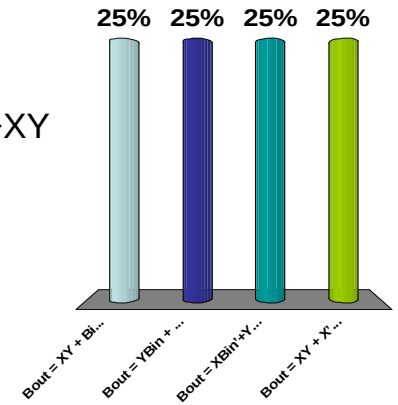
X	Y	B_{in}	B_{out}	Diff
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$Diff = X'YB_{in}' + XY' B_{in}' + X'Y' B_{in} + XYB_{in}$

	X' Y'	X' Y	X Y	X Y'
Bin'	0	1	0	1
Bin	1	0	1	0

What is the equation for B_{out}?

1. $B_{out} = XY + B_{in}$
2. $B_{out} = YB_{in} + XB_{in}$
3. $B_{out} = XB_{in}'+YB_{in}'+XY$
4. $B_{out} = XY + X'B_{in}$



The equation for B_{out}

X	Y	Bin	Bout	Diff
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$B_{out} = XB_{in}' + YB_{in}' + XY$

	X' Y'	X' Y	X Y	X Y'
Bin'	0	1	1	1
Bin	0	0	1	0

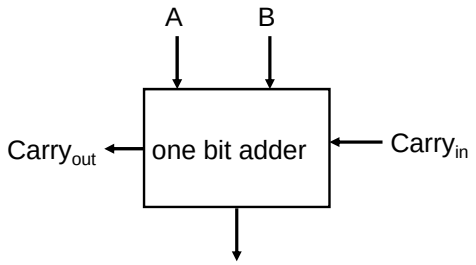
Subtraction

- 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.

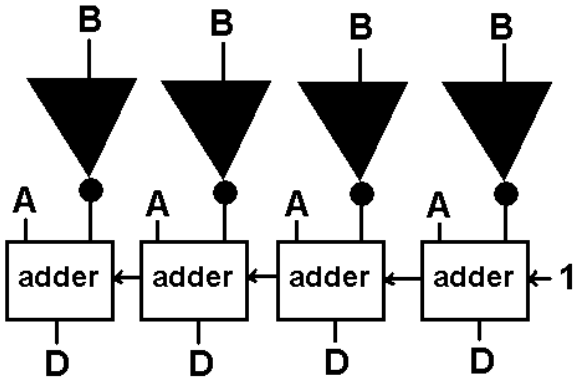
A+B'+1 = A + (B'+1) = A + (-B) = A - B

Draw a Subtraction Circuit

- Using one bit adder boxes and OR, AND or NOT gates, draw the logic diagram for a four bit twos complement A - B subtraction circuit.



Subtraction Circuit



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.

Timing for Ripple Adder

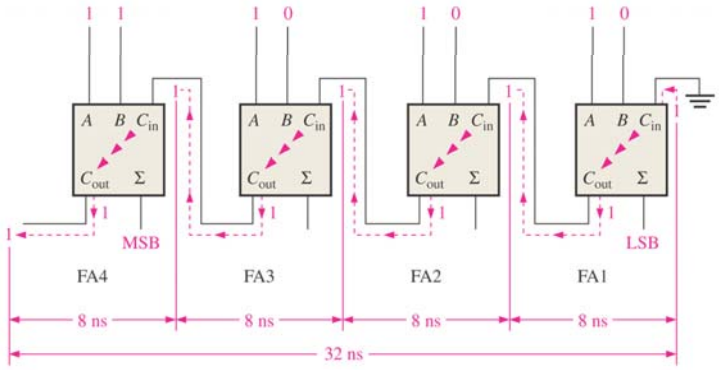


diagram from Digital Fundamentals by Thomas Floyd

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:

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

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

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

4 bit Adder with Look Ahead Carry

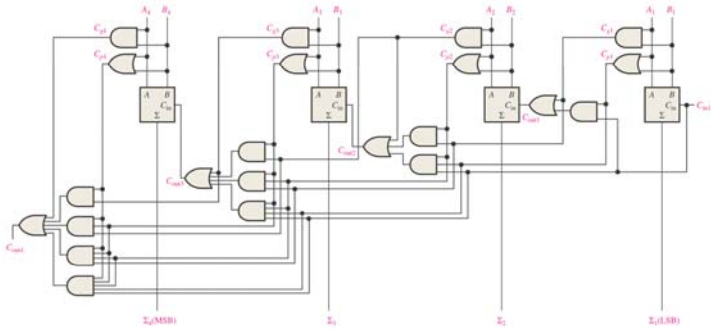


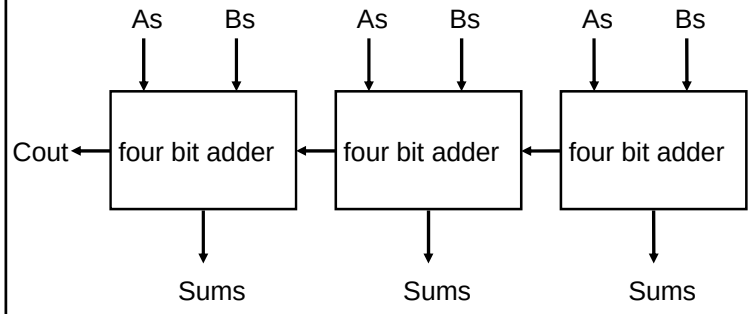
diagram from Digital Fundamentals by Thomas Floyd

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

Binary Multiplication

```
  110
 *101
  110     101 copy number when bit is one
  000     101 add zero when bit is zero
110       101 copy number when bit is zero
11110     add the results in proper column
```

Add and Shift

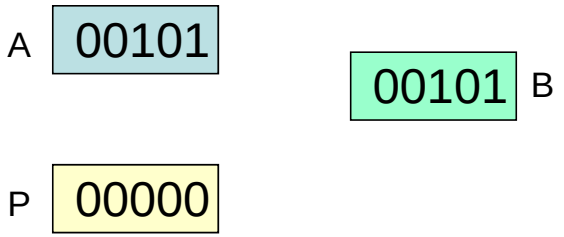
- Multiplication can be done by a series of adds and left shifts.
- Assume you have operands A & B and Product P (initially zero).

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

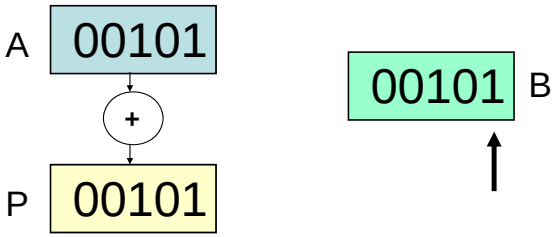
Sequential Logic

- The Add and Shift multiplier is not a combinatorial logic circuit.
- **Combinatorial logic** has no memory. The output is determined by the input.
- **Sequential logic** has memory. The output is determined by the input and what happened previously.
- Add and Shift multiplier is sequential logic.

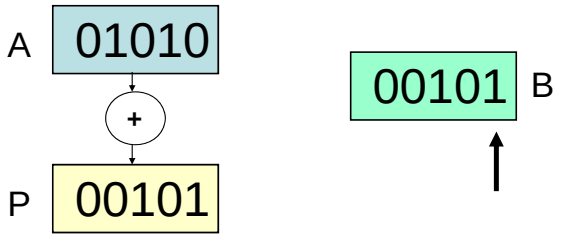
5 x 5 initial setup



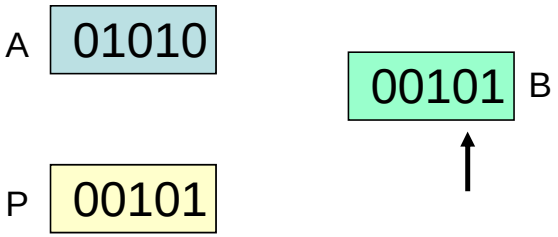
B_0 is 1 so Add



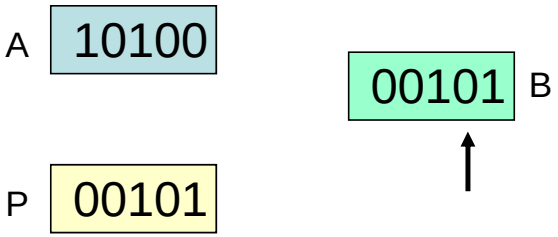
Shift A left one bit



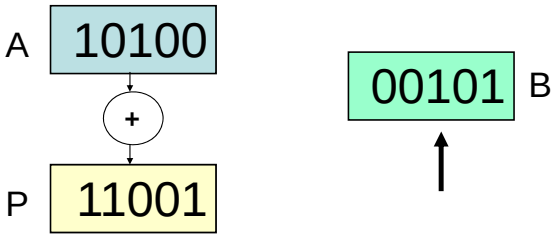
B_1 is zero so do nothing



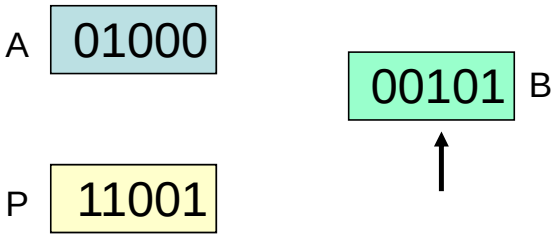
Shift A left one bit



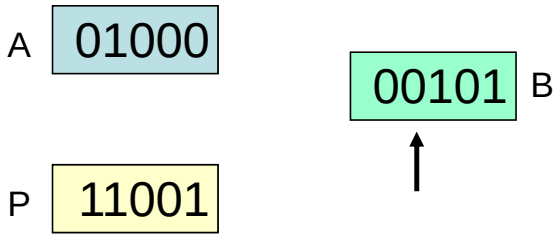
B_2 is 1 so Add



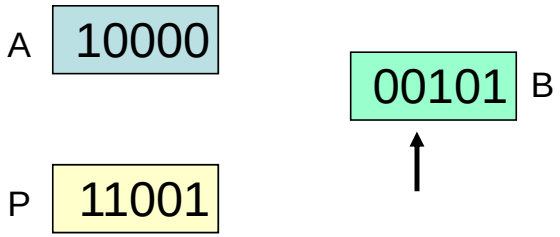
Shift A left one bit



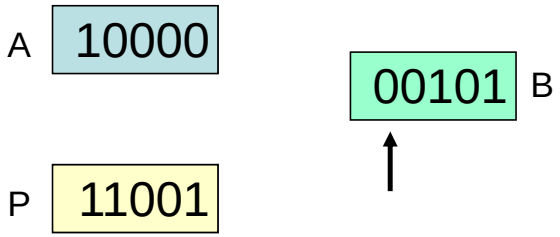
B_3 is zero so do nothing



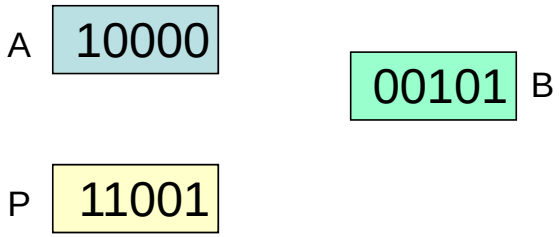
Shift A left one bit



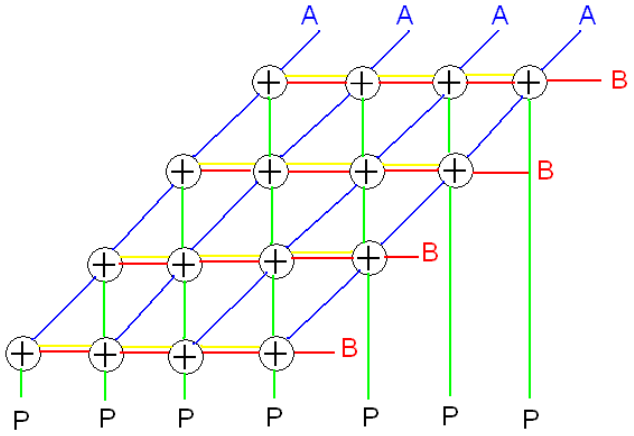
B_4 is zero so do nothing



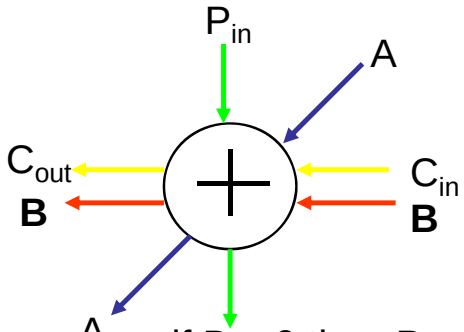
Done



Combinatorial Multiplier



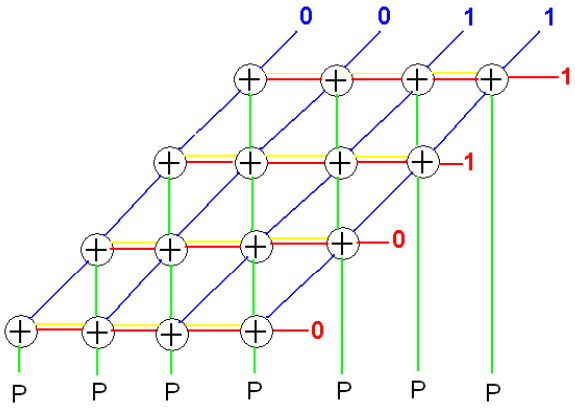
One Bit Multiplier



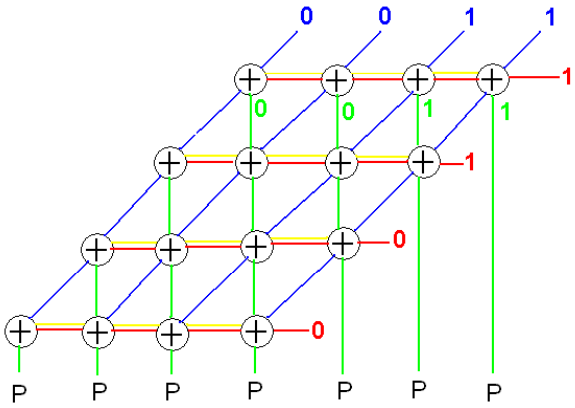
if B = 0 then P_{in}

if B = 1 then P_{in} + A

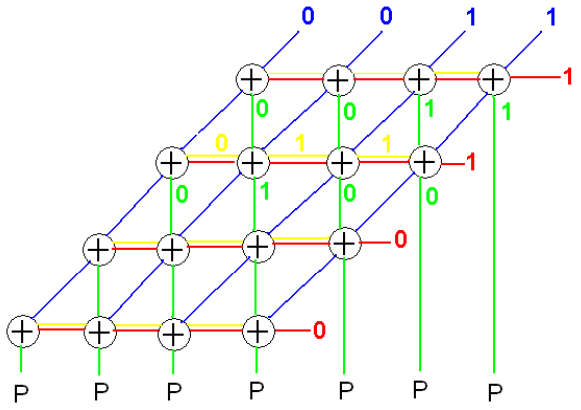
3 x 3 Example



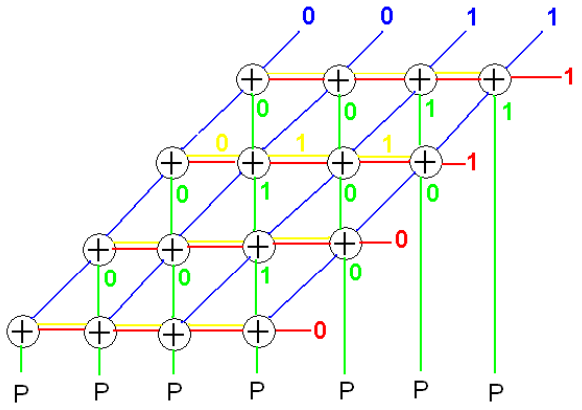
3 x 3 Example



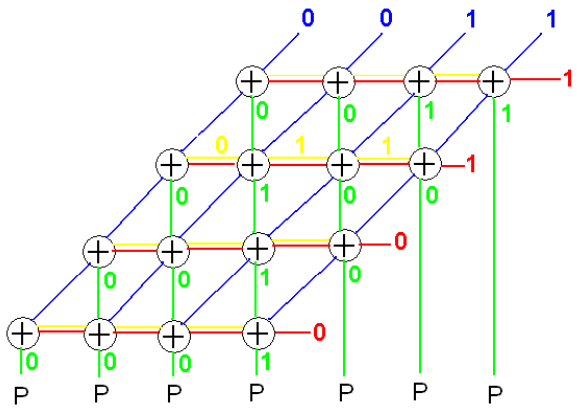
3 x 3 Example



3 x 3 Example



3 x 3 Example



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.