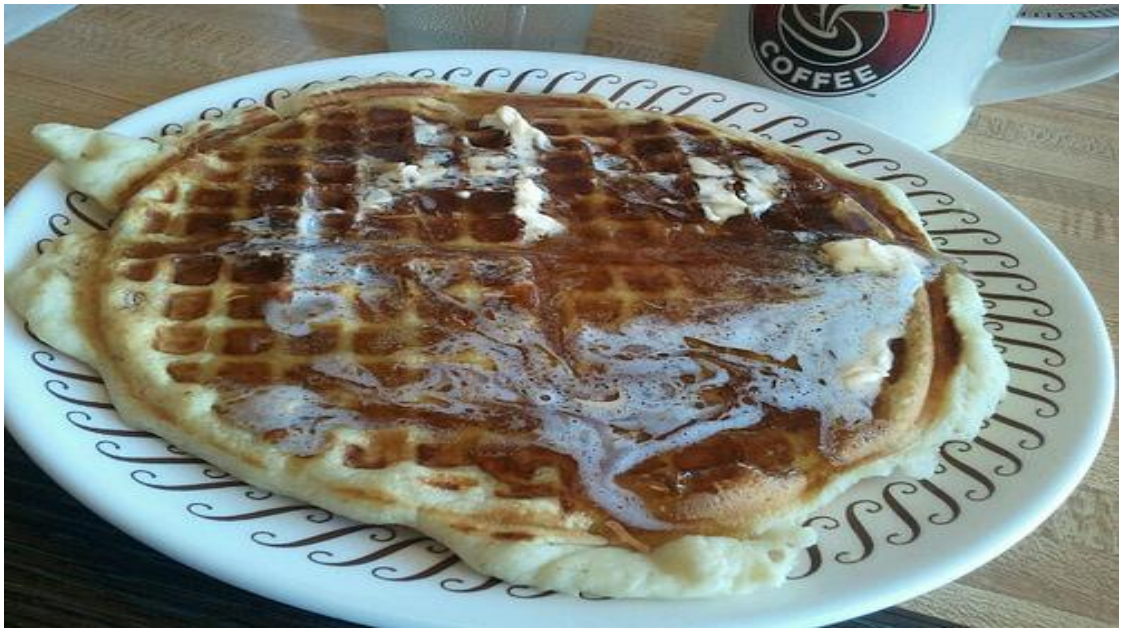
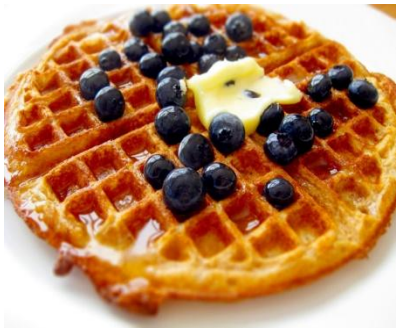
# Simple Logic




The only purpose of a waffle is to get the maple syrup and butter into my mouth.

One morning I was really hungry and made both of these:



Blueberry Waffle

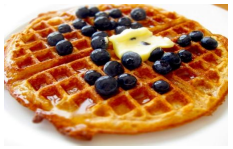
Pecan Waffle

I did not make either of these:

Pork and Kale Waffle

Masala Waffle

# NOT logic

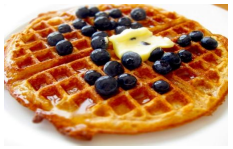
Did I NOT make a: 

Blueberry Waffle

?

# NOT logic

I did NOT make a:

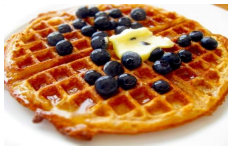


Blueberry Waffle

False

# NOT logic

I did NOT make a:



Blueberry Waffle

False

Did I NOT make a:

Pork and
Kale Waffle

?

# NOT logic

I did NOT make a:

Blueberry Waffle

False

I did NOT make a:

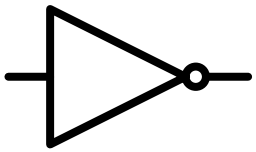
Pork and
Kale Waffle

True

# NOT logic

| | | |
|---|---|---|
| NOT True | is | False |
| NOT False | is | True |

Applying NOT to a statement flips its truth or falsity.



# NOT Truth Table

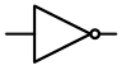
| NOT | |
|---|---|
| 0 | 1 |
| 1 | 0 |

NOT 0 is 1 (the negative of a false statement is a true statement)

NOT 1 is 0 (the negative of a true statement is a false statement)

A NOT Gate is like a light switch in a room that has been turned upside down. When you flip it up, normally turning it on, the light goes off. When you flip it down, normally turning it off, the light goes on.

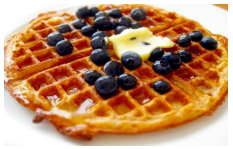
# Logic Gate Implementation

NOT Gate 

NOT closed is open
NOT 1 = 0

NOT open is closed
NOT 0 = 1

# AND logic

Did I make a: 

Blueberry Waffle

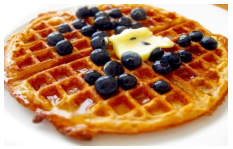
AND 

Pecan Waffle

?

# AND logic

I made a:



Blueberry Waffle

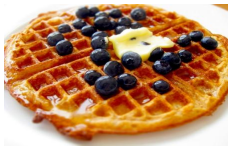
AND

Pecan Waffle

True

# AND logic

I made a:



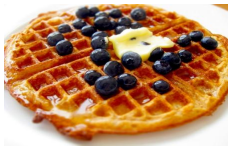
Blueberry Waffle

AND

Pecan Waffle

True

Did I make a:



Blueberry Waffle

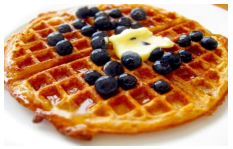
AND

Masala Waffle

?

# AND logic

I made a:



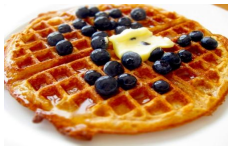
Blueberry Waffle

AND

Pecan Waffle

True

I made a:



Blueberry Waffle

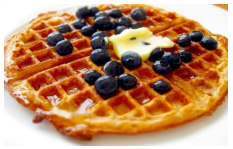
AND

Masala  Waffle

False

# AND logic

I made a:



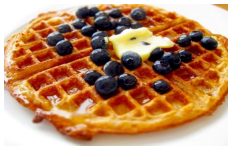
Blueberry Waffle

AND

Pecan Waffle

True

I made a:



Blueberry Waffle

AND

Masala Waffle

False

Did I make a:

Pork and
Kale Waffle

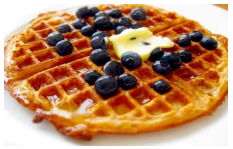
AND

Pecan Waffle

?

# AND logic

I made a:



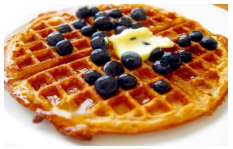
Blueberry Waffle

AND

Pecan Waffle

True

I made a:



Blueberry Waffle

AND

Masala Waffle

False

I made a:

Pork and
Kale Waffle

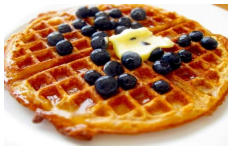
AND

Pecan Waffle

False

# AND logic

I made a:



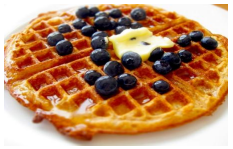
Blueberry Waffle

AND

Pecan Waffle

True

I made a:



Blueberry Waffle

AND

Masala Waffle

False

I made a:

Pork and
Kale Waffle

AND

Pecan Waffle

False

Did I make a:

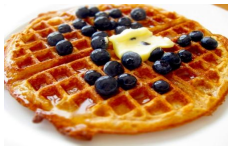
Pork and
Kale Waffle

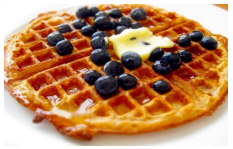
AND

Masala Waffle

?

# AND logic

I made a:  Blueberry Waffle AND  Pecan Waffle True

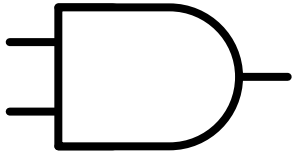
I made a:  Blueberry Waffle AND  Masala Waffle False

I made a:  Pork and Kale Waffle AND  Pecan Waffle False

I made a:  Pork and Kale Waffle AND  Masala Waffle False

# AND logic

| | | | | |
|---|---|---|---|---|
| True | AND | True | = | True |
| True | AND | False | = | False |
| False | AND | True | = | False |
| False | AND | False | = | False |



# AND Truth Table

| AND | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

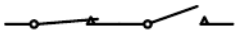
0 AND 0 is 0 (a false statement AND a false statement is a false combination)
0 AND 1 is 0 (a false statement AND a true statement is a false combination)
1 AND 0 is 0 (a true statement AND a false statement is a false combination)
1 AND 1 is 1 (a true statement AND a true statement is a true combination)

An AND Gate is two similar to draw bridges in a row.  If either one is up, then you won't be able to drive through.

# Logic Gate Implementation

AND Gate 

closed AND closed is closed
1 AND 1 = 1



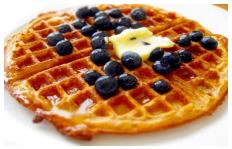
closed AND open is open
1 AND 0 = 0

open AND closed is open
0 AND 1 = 0

open AND open is open
0 AND 0 = 0

# OR logic

Did I make a:



Blueberry Waffle

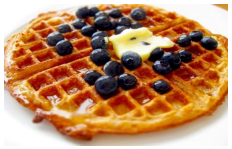
OR

Masala Waffle

?

# OR logic

I made a:



Blueberry Waffle

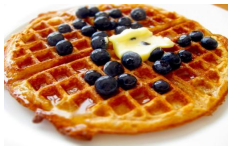
OR

Masala Waffle

True

# OR logic

I made a:



Blueberry Waffle

OR

Masala Waffle

True

Did I make a:

Pork and
Kale Waffle

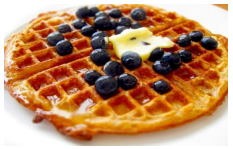
OR

Pecan Waffle

?

# OR logic

I made a:



Blueberry Waffle

OR

Masala Waffle

True

I made a:

Pork and
Kale Waffle

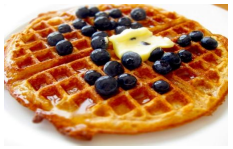
OR

Pecan Waffle

True

# OR logic

I made a:



Blueberry Waffle

OR

Masala Waffle

True

I made a:

Pork and
Kale Waffle

OR

Pecan Waffle

True

Did I make a:

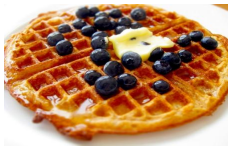
Pork and
Kale Waffle

OR

Masala Waffle

?

# OR logic

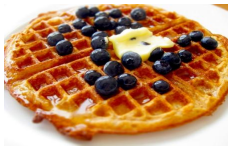
I made a:  Blueberry Waffle OR  Masala Waffle True

I made a:  Pork and Kale Waffle OR  Pecan Waffle True

I made a:  Pork and Kale Waffle OR  Masala Waffle False

# OR logic

Did I make a:



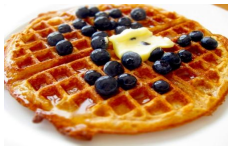
Blueberry Waffle

OR

Pecan Waffle

?

I made a:



Blueberry Waffle

OR

Masala Waffle

True

I made a:

Pork and
Kale Waffle

OR

Pecan Waffle

True

I made a:

Pork and
Kale Waffle

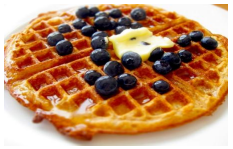
OR

Masala Waffle

False

# OR logic

I made a:



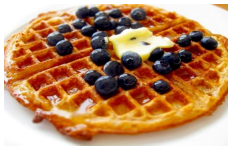
Blueberry Waffle

OR

Pecan Waffle

True

I made a:



Blueberry Waffle

OR

Masala Waffle

True

I made a:

Pork and
Kale Waffle

OR

Pecan Waffle

True

I made a:

Pork and
Kale Waffle

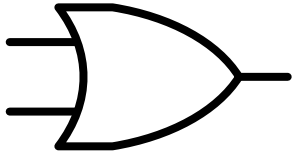
OR

Masala Waffle

False

# OR logic

| | | | | |
|---|---|---|---|---|
| True | OR | True | = | True |
| True | OR | False | = | True |
| False | OR | True | = | True |
| False | OR | False | = | False |



# OR Truth Table

Inclusive OR

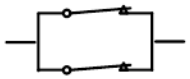
| OR | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

0 OR 0 is 0 (a false statement OR a false statement is a false combination)
0 OR 1 is 1 (a false statement OR a true statement is a true combination)
1 OR 0 is 1 (a true statement OR a false statement is a true combination)
1 OR 1 is 1 (a true statement OR a true statement is a true combination)

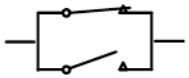
An OR Gate is two similar to draw bridges in parallel.  If either one is down, you will be able to drive through.

# Logic Gate Implementation

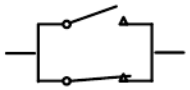
OR Gate 



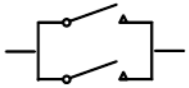
closed OR closed is closed
1 OR 1 = 1



closed OR open is closed
1 OR 0 = 1



open OR closed is closed
0 OR 1 = 1



open OR open is open
0 OR 0 = 0

# Logic Gate Implementation

To make a computer, all you need is Memorial Stadium for an afternoon, a few thousand chairs, a few thousand undergraduates, one pizza for each three undergraduates, and lots and lots of soda.

## AND Gate

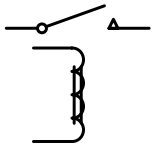
| | | | | |
|---|---|---|---|---|
|  | AND |  | IS |  |
| Hand Up | AND | Hand Up | is | Hand Up |
| 1 | AND | 1 | = | 1 |
|  | AND |  | IS |  |
| Hand Up | AND | Hand Down | is | Hand Down |
| 1 | AND | 0 | = | 0 |
|  | AND |  | IS |  |
| Hand Down | AND | Hand Up | is | Hand Down |
| 0 | AND | 1 | = | 0 |
|  | AND |  | IS |  |
| Hand Down | AND | Hand Down | is | Hand Down |
| 0 | AND | 0 | = | 0 |

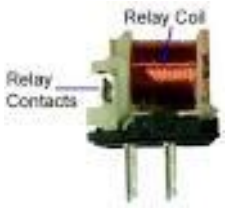
# Different Types of Switches

Mechanical Switch

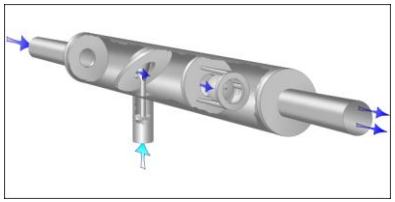


Electro-mechanical Relay



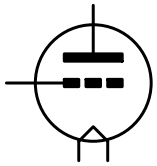



Fluid Pressure Value

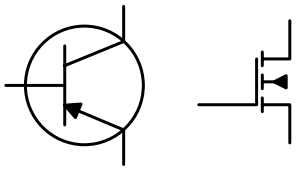


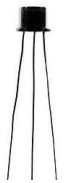
Electronic Tube

Called a “valve” in England



Transistor



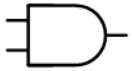


# Logic Gate Implementation

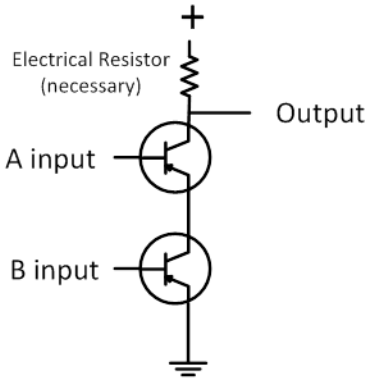
## using Transisters

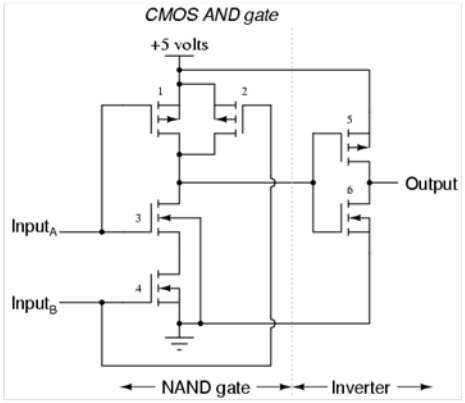
CMOS type transistors eliminate the need for resistors, allowing millions of transistors on a single chip

### AND Gate

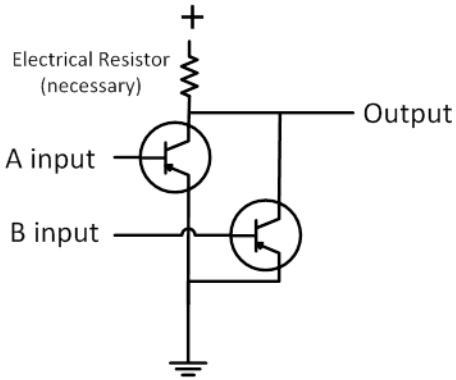


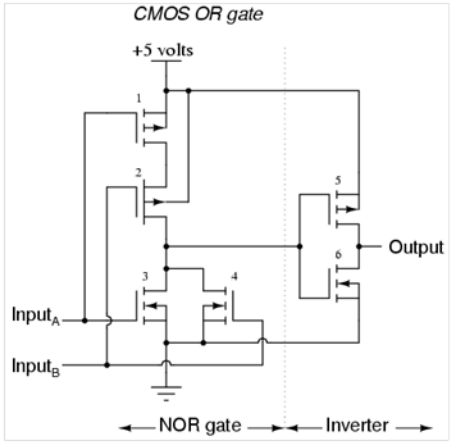
Older style individual transistors



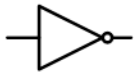



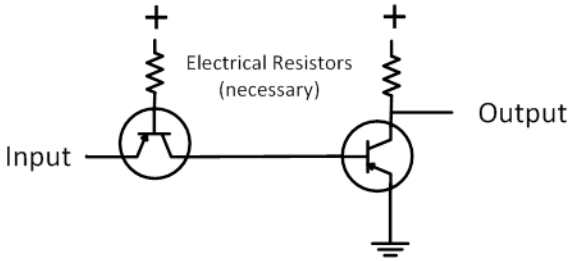

### OR Gate

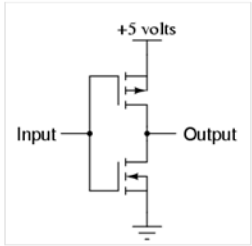



### NOT Gate

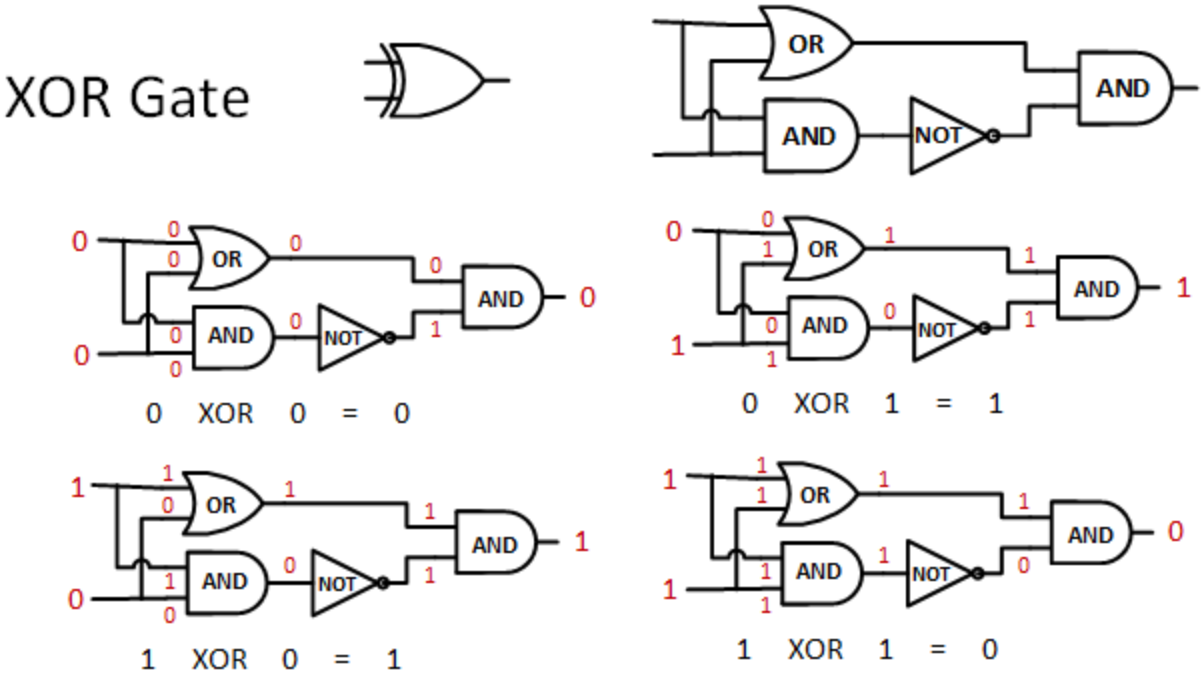




# Exclusive OR Gate (XOR)

Outputs True if either input is True, but not if both inputs are True



From Page 379 of Principia Mathematica

by Bertrand Russell and Alfred North Whitehead, 1910 - 1913

$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

*Dem.*

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

$[*51{\cdot}231] \qquad \equiv . \iota`x \cap \iota`y = \Lambda .$

$[*13{\cdot}12] \qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash .$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

# Five Bit Adder

**ADD opcode**

10010 + 00111

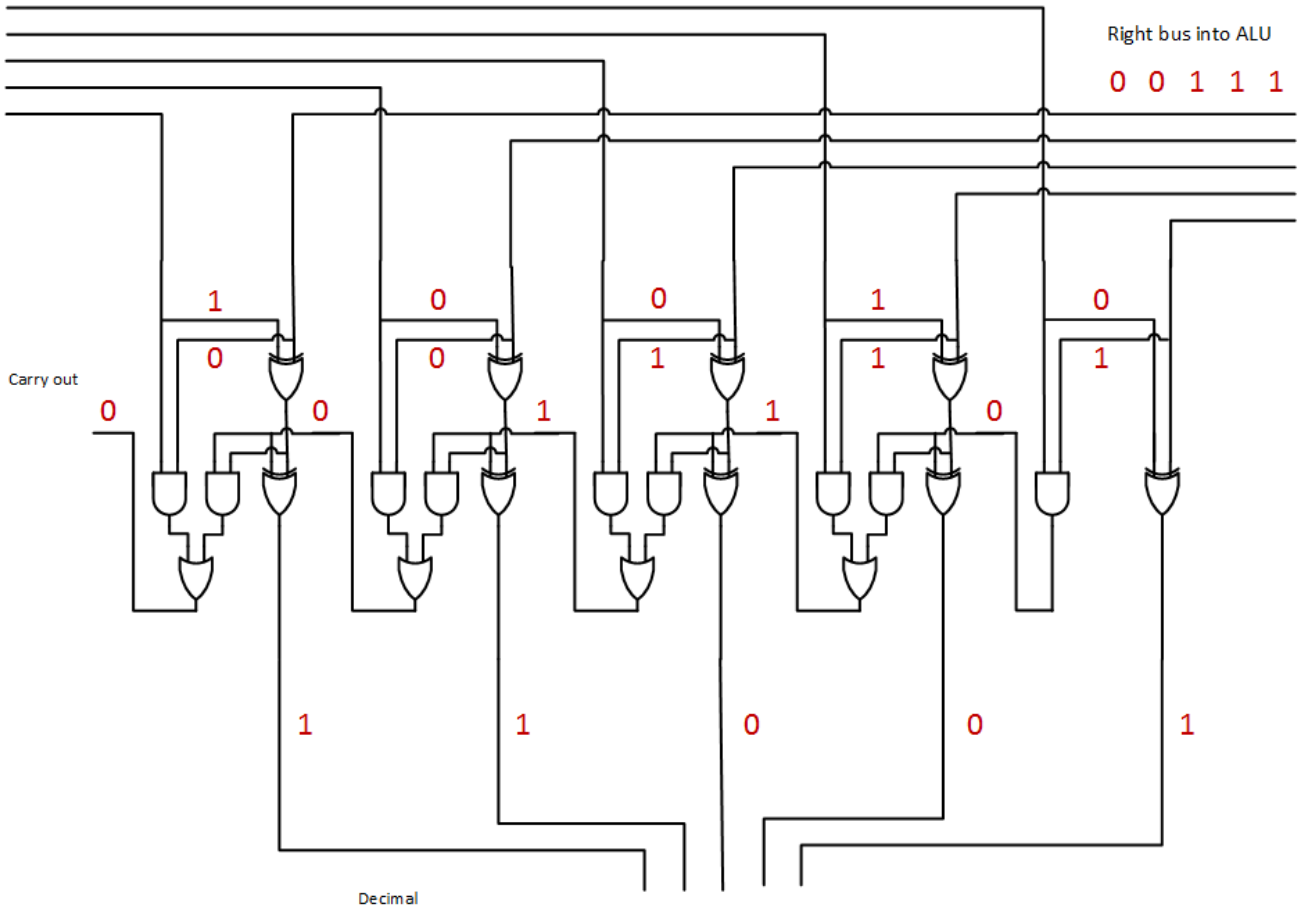
18 + 7

Left bus from Incrementor

1 0 0 1 0

Right bus into ALU

0 0 1 1 1



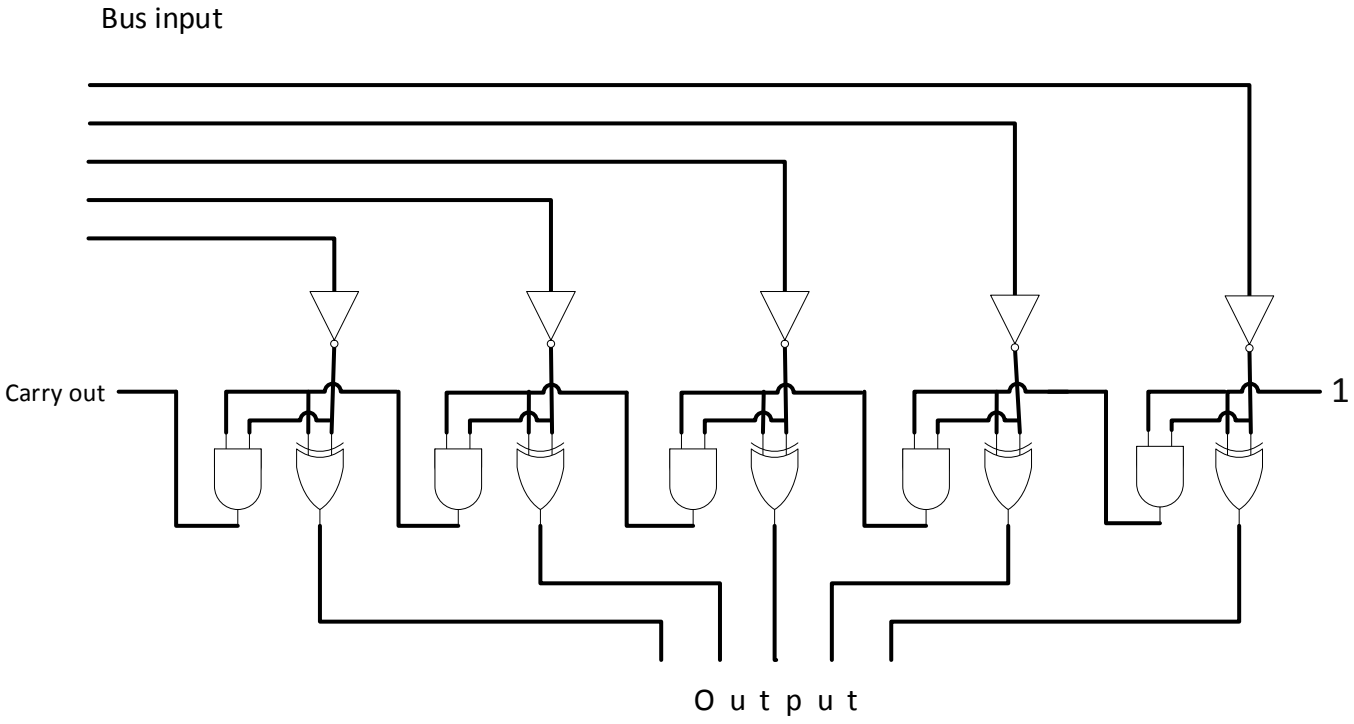

Output

Decimal equivalent

18 + 7 = 25

Accurate addition of two 5-bit binary numbers.

# Twos Complementor




twos complement of the input bus

# Five Bit Adder

**SUB opcode**

01110 – 00101

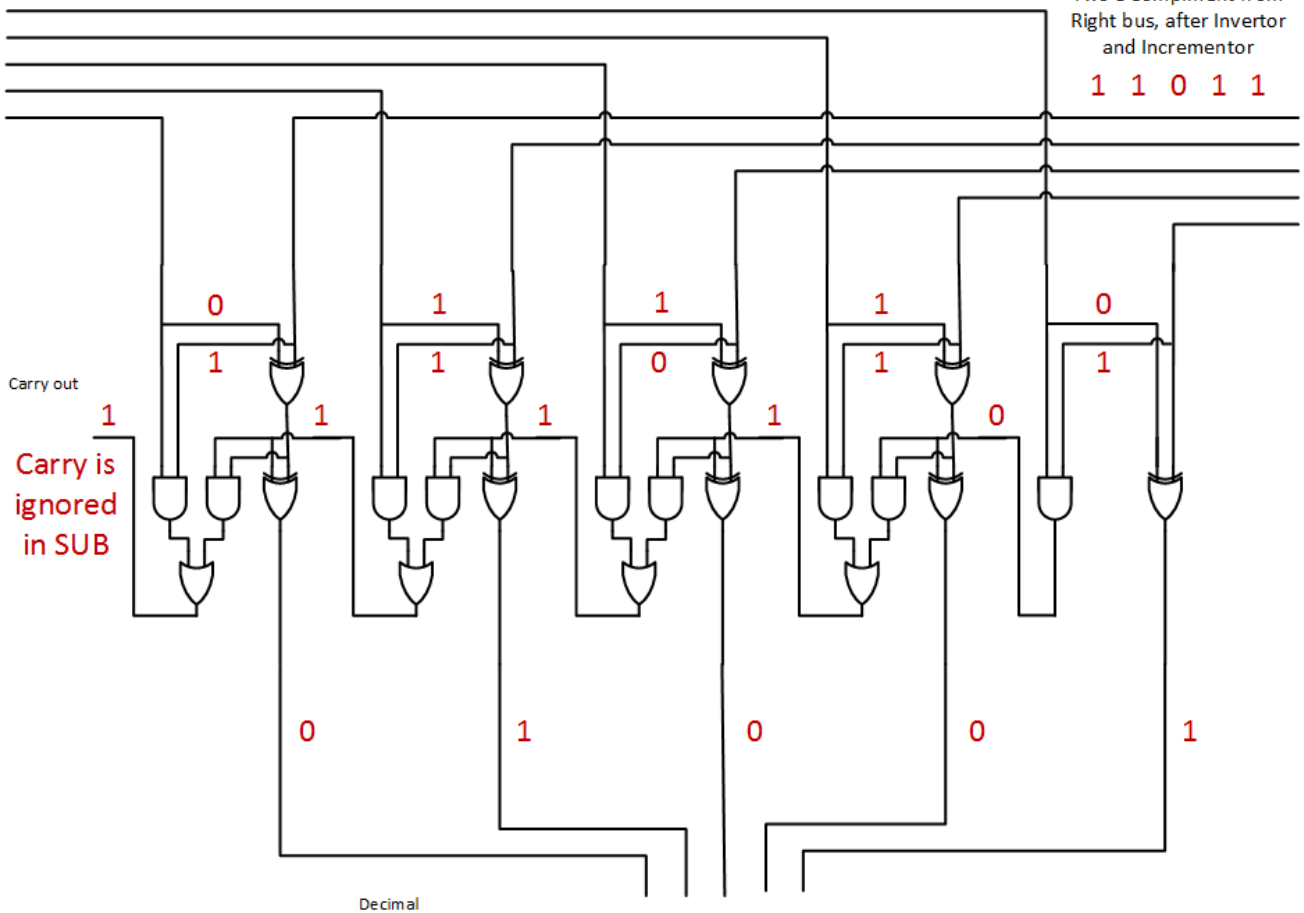
14 - 5

Left bus from ALU

0 1 0 1 0

Two's Compiment from Right bus, after Invertor and Incrementor

1 1 0 1 1




1 0 1 1 1

Output

Decimal equivalent

14

+ (-5)

9

0 1 1 1 0

+ 1 0 1 1 1

1 1 0 0 1

Accurate addition of two 5-bit binary numbers.

# Multiplication and Division

Multiplication and division is done in hardware by various logic gate designs. Typically, modern computers use about 4000 gates to multiply or divide two 64-bit binary numbers. This is the meaning of a “64-bit” computer. 4000 gates is minuscule compared to the approximately half billion gates on a typical modern Central Processing Unit (CPU) chip.

End of Presentation