Digital Circuits

1. Inputs & Outputs are quantized at two levels.

2. Binary arithmetic, only digits are 0 & 1. Position indicates power of 2.

$$11001 = 2^4 + 2^3 + 0 + 0 + 2^0$$

$$16 + 8 + 0 + 0 + 1 = 25$$

Digital Circuits

3. Binary Coded Decimal (BCD) is an alternative binary representation still used for certain types of calculations where base-10 arithmetic is directly used where the rounding errors of binary cannot be tolerated (e.g. financial calculations, etc.) Also still often used for storing dates.

Bits are used in groups of 4 to represent a decimal (base 10) digit.

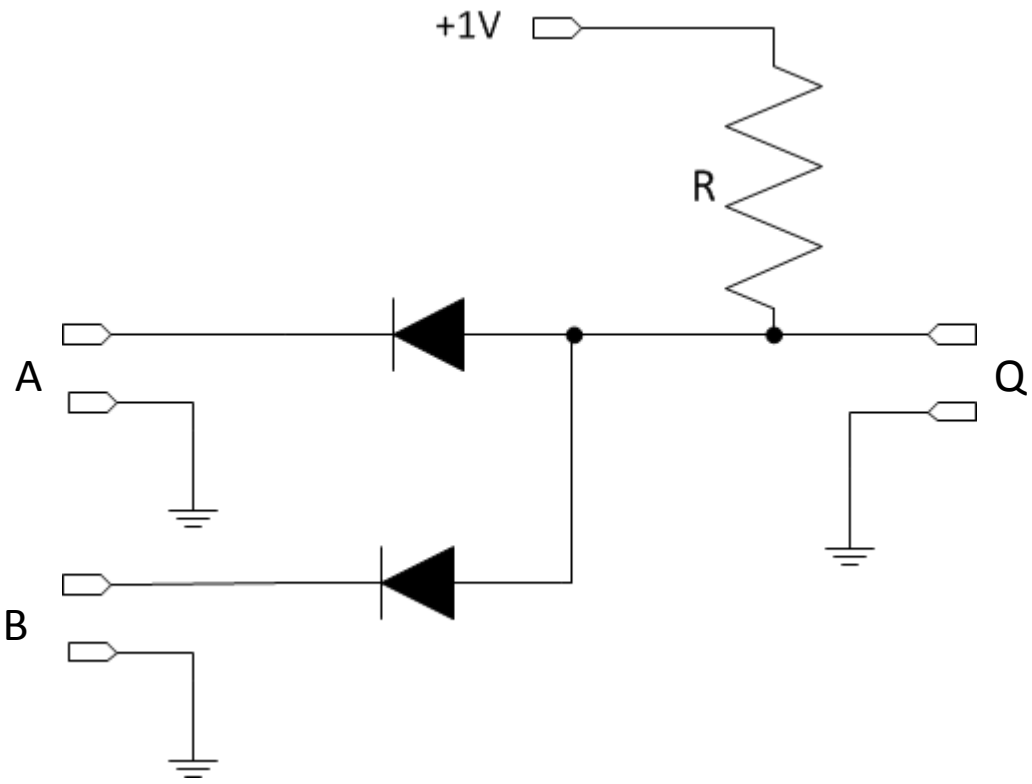
Decimal Digit	BCD 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

From Sept 18 2017 lecture

Semiconductors

Diode Circuits

AND LOGIC GATE (A and B)



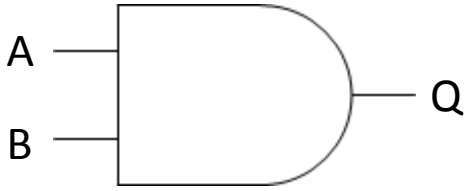
BC offers a Symbolic Logic course from the Philosophy Department (PHIL557701).

AND Truth Table

A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

Boolean Logic Gates

AND LOGIC GATE (A and B)



A AND B = Q

A • B = Q

AB = Q

Looks kind of a fat "times" dot.

AND Truth Table

A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

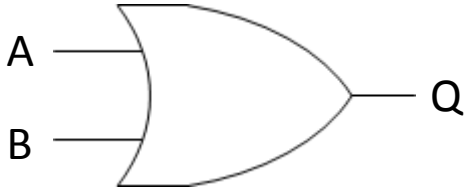
Not limited to two inputs. May have any number of inputs.

Notice: I have not shown the grounds nor the power supply connections.

This is a schematic symbol set for describing the circuit's logic, not necessarily how to implement it in actual circuitry.

Boolean Logic Gates

OR LOGIC GATE (A or B)



A OR B = Q
A + B = Q

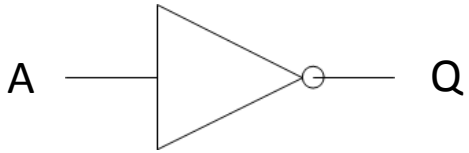
OR Truth Table

A	B	Q
0	0	0
1	0	1
0	1	1
1	1	1

Not limited to two inputs. May have any number of inputs.

Boolean Logic Gates

INVERT LOGIC GATE ($\overline{A}$)



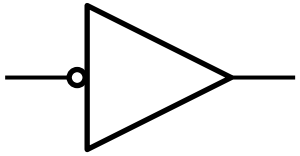
INVERT Truth Table

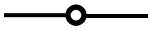
A	Q
0	1
1	0

NOT A = Q

$\overline{A}$ = Q

Alternate symbol has the “circle” on the input



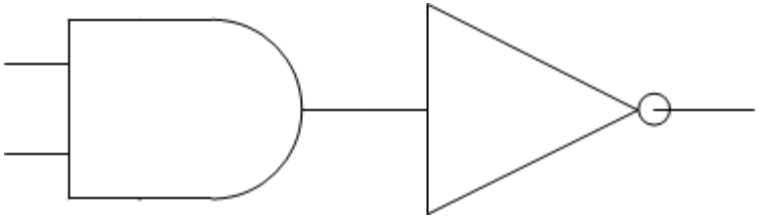
Sometimes just the circle inline 

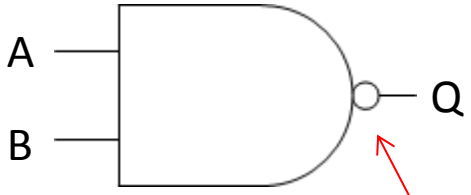
Boolean Logic Gates

With the three above modules, you can build ANY digital circuit.

Boolean Logic Gates

NAND LOGIC GATE (NOT AND)





A NAND B = Q

$\overline{A \bullet B} = Q$

NAND Truth Table

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

VS

AND Truth Table

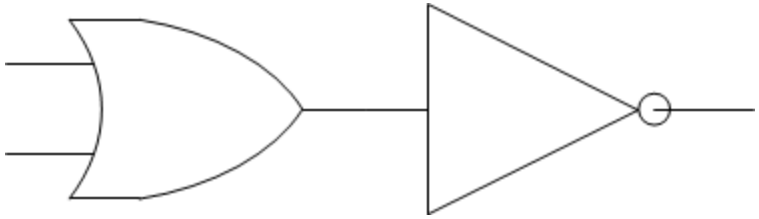
A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

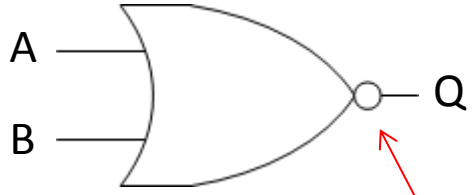
Not limited to two inputs. May have any number of inputs.

Note the little “NOT” circle

Boolean Logic Gates

NOR LOGIC GATE (NOT OR)





A NOR B = Q

$\overline{A + B} = Q$

NOR Truth Table

A	B	Q
0	0	1
1	0	0
0	1	0
1	1	0

VS

OR Truth Table

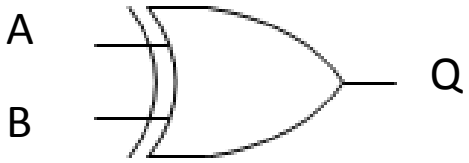
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	1

Not limited to two inputs. May have any number of inputs.

Note the little “NOT” circle

Boolean Logic Gates

EXCLUSIVE OR LOGIC GATE (XOR)



A XOR B = Q

$A \oplus B = Q$

XOR Truth Table

A	B	Q	
0	0	0	
1	0	1	$\leftarrow A \bullet \overline{B}$
0	1	1	$\leftarrow B \bullet \overline{A}$
1	1	0	

Not limited to two inputs. May have any number of inputs.

Notice what this truth table means:

Q = 1 if and only if A OR B = 1 ***but not both***.

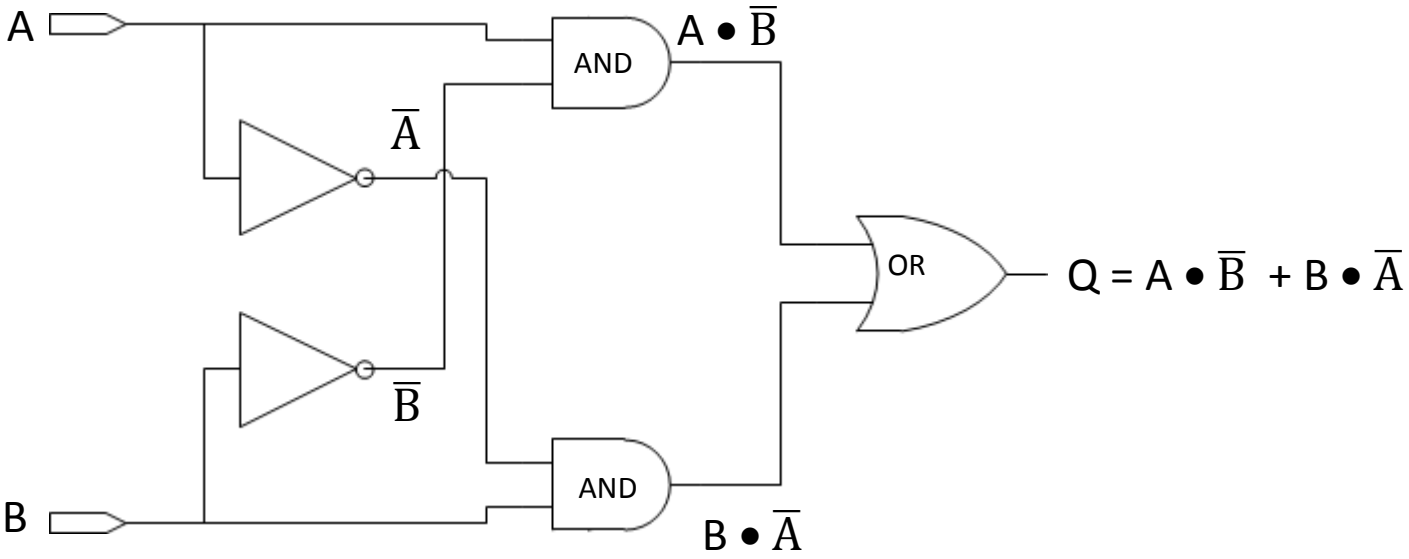
$$A \oplus B = A \bullet \overline{B} + B \bullet \overline{A}$$

So how do you make this out of AND, OR & NOT gates?

Boolean Logic Gates

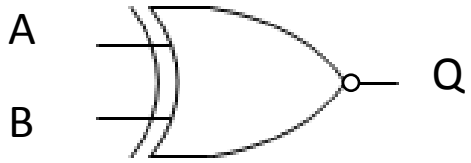
EXCLUSIVE OR LOGIC GATE (XOR)

Make a "$A \bullet \overline{B} + B \bullet \overline{A}$" circuit.



Boolean Logic Gates

EQUALITY (NOT XOR)



NOT(A XOR B) = Q

$\overline{A \oplus B} = Q$

NOT XOR Truth Table

A	B	Q
0	0	1
1	0	0
0	1	0
1	1	1

Q = 1 if and only if A OR B are the same.

Not limited to two inputs. May have any number of inputs.

$$Q = \overline{A \oplus B} = \overline{A \bullet \overline{B} + B \bullet \overline{A}}$$

Boolean Logic Gates

Logic Identities

$A \bullet B \bullet C = (A \bullet B) \bullet C = A \bullet (B \bullet C)$

$A \bullet B = B \bullet A$

$A \bullet A = A$

$A \bullet 1 = A$

$A \bullet 0 = 0$

$A \bullet (B+C) = A \bullet B + A \bullet C$

$A + A \bullet B = A$

$A + B \bullet C = (A + B) \bullet (A + C)$

$A + B + C = (A + B) + C = A + (B + C)$

$A + B = B + A$

$A + A = A$

$A + 1 = 1$

$A + 0 = A$

$\overline{1} = 0$

$\overline{0} = 1$

$A + \overline{A} = 1$

$A \bullet \overline{A} = 0$

$\overline{\overline{A}} = A$

$A + \overline{A} \bullet B = A + B$

$\overline{(A + B)} = \overline{A} \bullet \overline{B}$

$\overline{(A \bullet B)} = \overline{A} + \overline{B}$

DeMorgan's Theorem

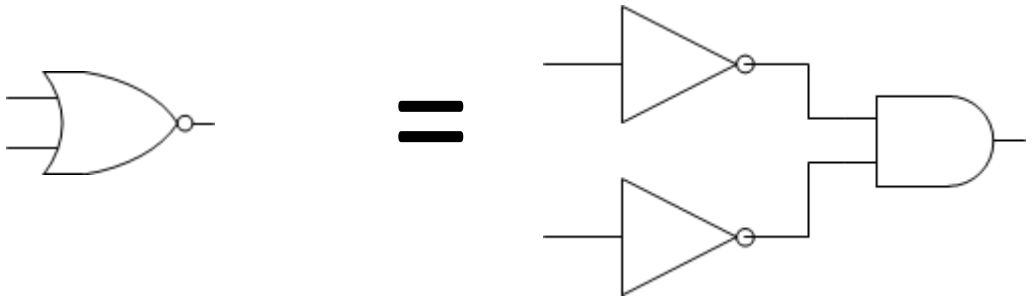
NOT (A or B) = NOT A and NOT B

NOT (A and B) = NOT A or NOT B

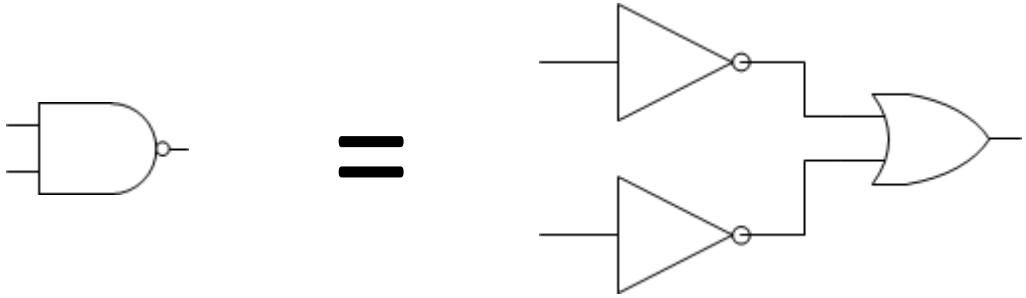
Boolean Logic Gates

DeMorgan's Theorem

$\overline{(A + B)} = \overline{A} \bullet \overline{B}$ which means NOT (A or B) = NOT A and NOT B

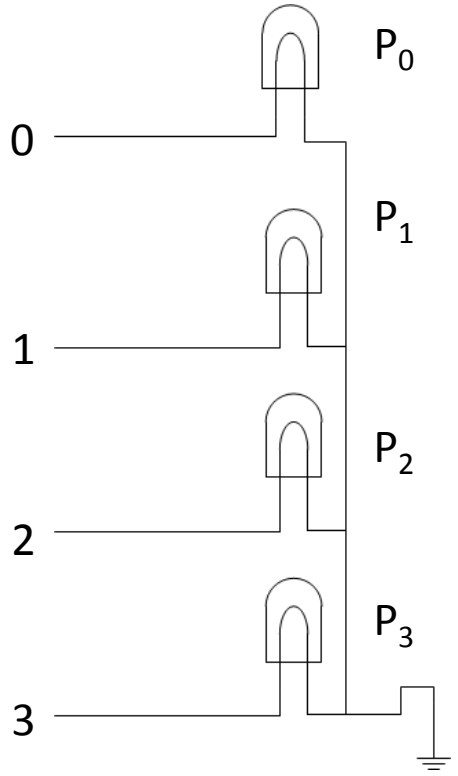


$\overline{(A \bullet B)} = \overline{A} + \overline{B}$ which means NOT (A and B) = NOT A or NOT B



Boolean Logic Gates

Example: Suppose we want to turn on one of four lights. We can do that by running 5 wires, one to each light and a common ground. Send the on signal along whichever wire goes to the lamp we want on.



Boolean Logic Gates

Let's achieve the same function using fewer wires.

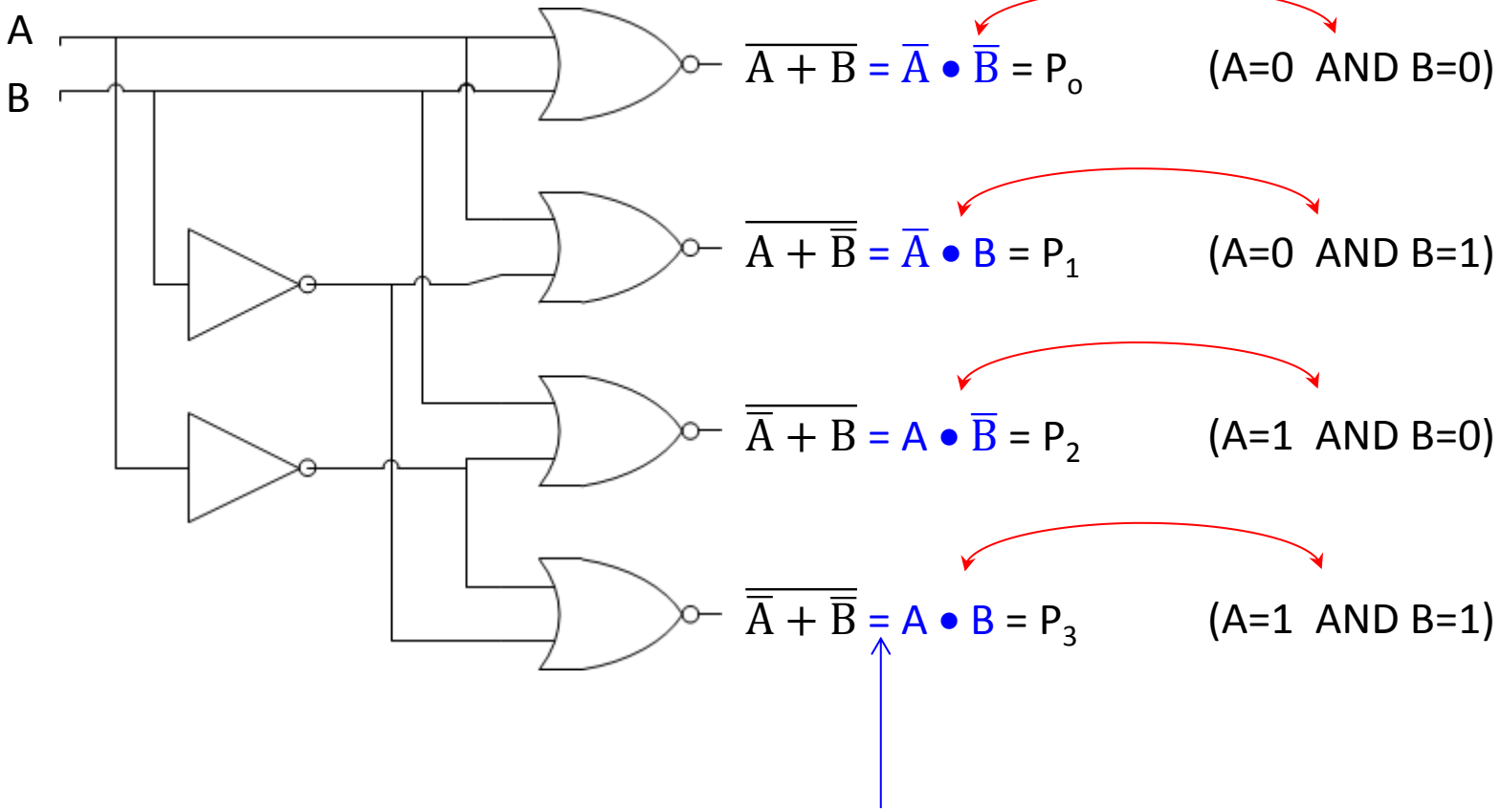
Two lines (plus ground) can carry binary digits 0-3.

D	A	B
0	0	0
1	1	0
2	0	1
3	1	1

But we need a way to decode this "2 wire" communication into "4 wires" to run the lights.

Boolean Logic Gates

Two wire decoder

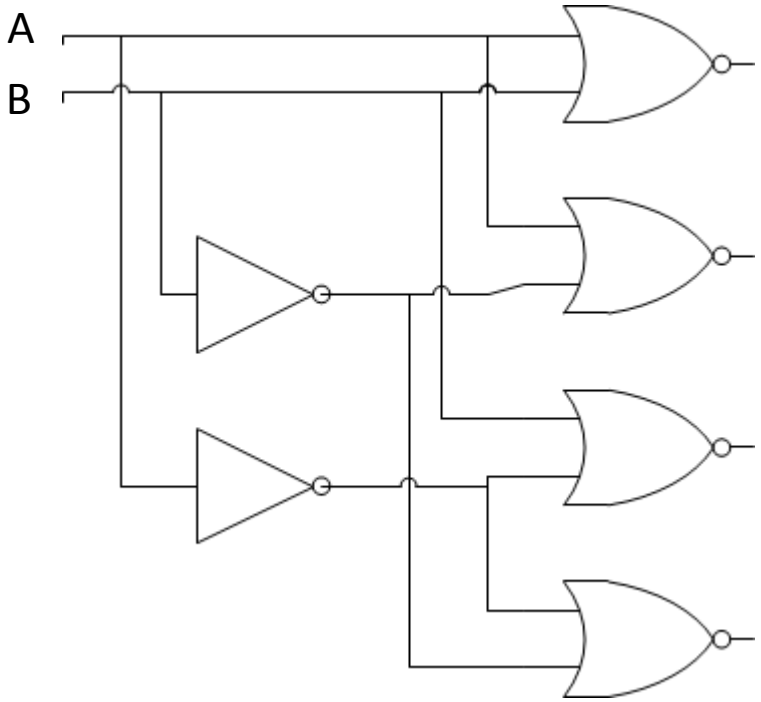


$\overline{(A + B)} = \overline{A} \bullet \overline{B}$ (DeMorgan's Theorem) & $\overline{\overline{A}} = A$

NOT (A or B) = NOT A and NOT B

Boolean Logic Gates

Two wire decoder



$\overline{A + B} = \overline{A} \bullet \overline{B} = P_o$

A	B	P_o
0	0	1
0	1	0
1	0	0
1	1	0

$\overline{A + \overline{B}} = \overline{A} \bullet B = P_1$

A	B	P_1
0	0	0
0	1	1
1	0	0
1	1	0

$\overline{\overline{A} + B} = A \bullet \overline{B} = P_2$

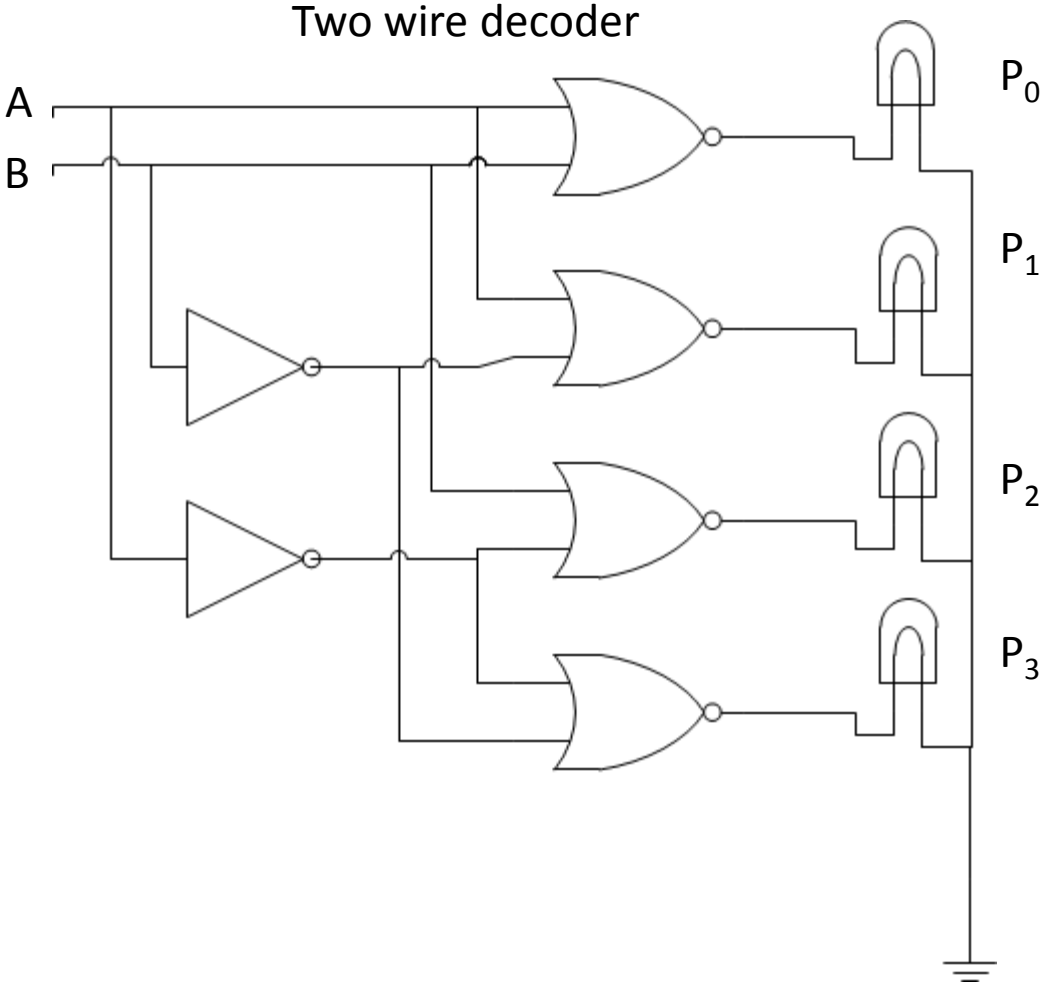
A	B	P_2
0	0	0
0	1	0
1	0	1
1	1	0

$\overline{\overline{A} + \overline{B}} = A \bullet B = P_3$

A	B	P_3
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Logic Gates

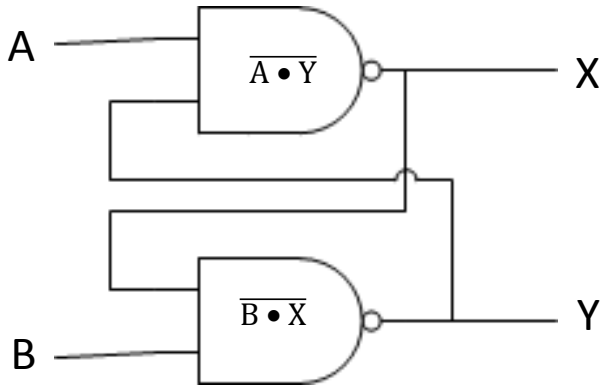
Two wire decoder



D	A	B	P_o	P_1	P_2	P_3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

Boolean Logic Gates

Flip-Flop



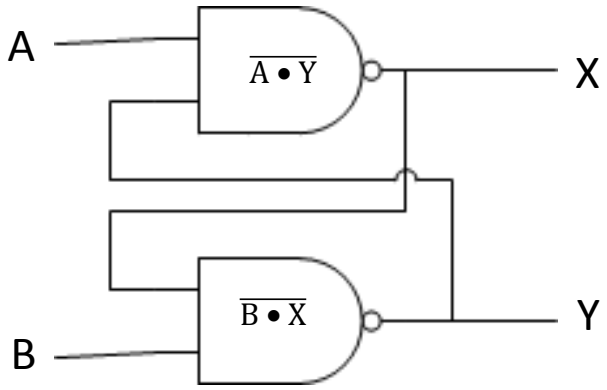
Assume A & B are 1.

If X = 1 then Y = 0	State 1
If Y = 1 then X = 0	State 2

Notice both states are consistent with A & B =1.

Boolean Logic Gates

Flip-Flop



Assume A & B are 1, X = 1 & Y = 0. (State 1)

Now, bring B = 0 $\rightarrow$ Y goes to Y = 1.

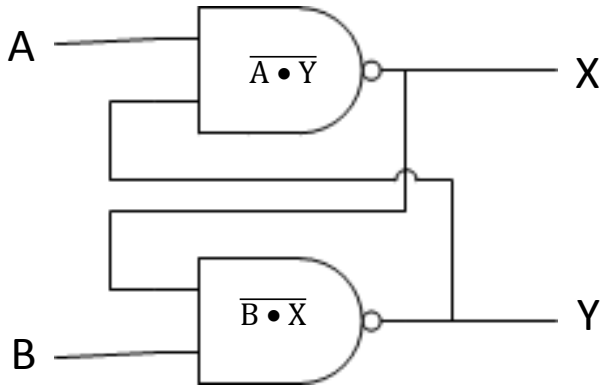
Y = 1 into the top gate throws X = 0.

We can bring B back to 1 (B = 1) and we are now in State 2.

A & B =1, X = 0 , Y = 1

Boolean Logic Gates

Flip-Flop



Assume A & B are 1, X = 0 & Y = 1. (State 2)

Now, bring A = 0 $\rightarrow$ X goes to X = 1.

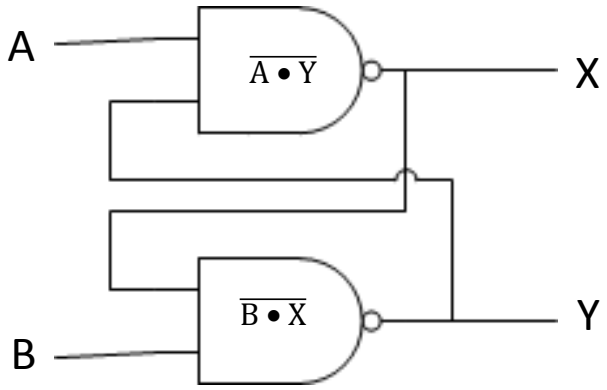
X = 1 into the bottom gate throws Y = 0.

We can bring A back to 1 (A = 1) and we are now back in State 1.

A & B =1, X = 1 , Y = 0

Boolean Logic Gates

Flip-Flop



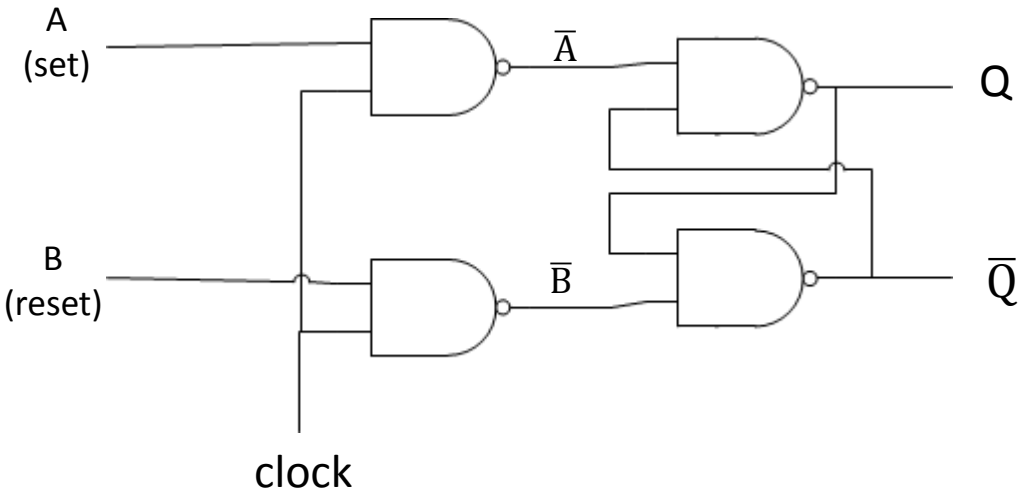
Pulsing A =0 (low) forces the X = 1 (high) and Y = 0 (low) state.

Pulsing B = 0 (low) forces the X = 0 (low) and Y =1 (high) state.

Circuits “flips” into one state, “flops” into the other.

Boolean Logic Gates

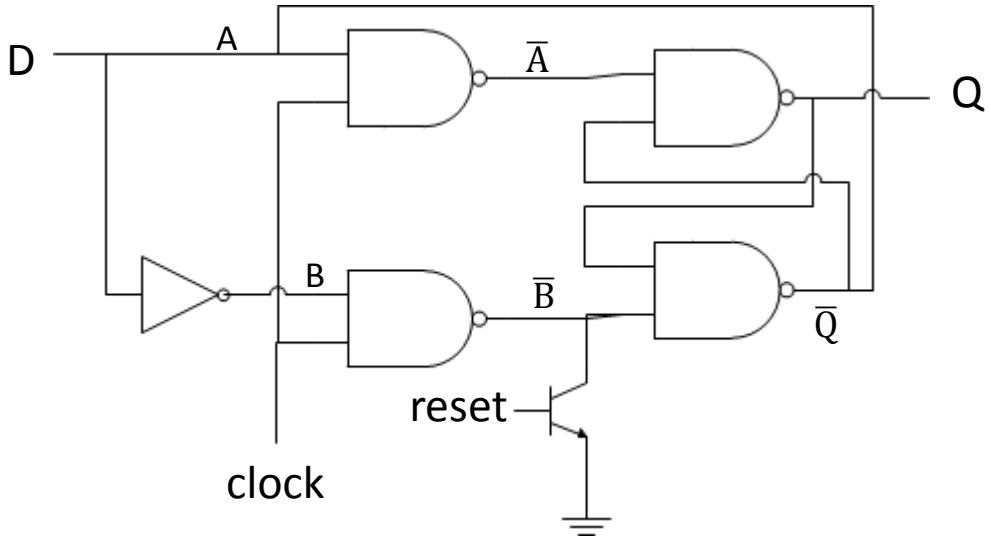
Clocked Flip-Flops



Same as before, only the two front end NAND gates only let the flip-flop “see” A & B when clock =1.

Boolean Logic Gates

Toggled Flip-Flops



Note: A = $\overline{B}$, A = $\overline{Q}$

Suppose Q = 1, then $\overline{Q}$ =0, A = 0 & B =1

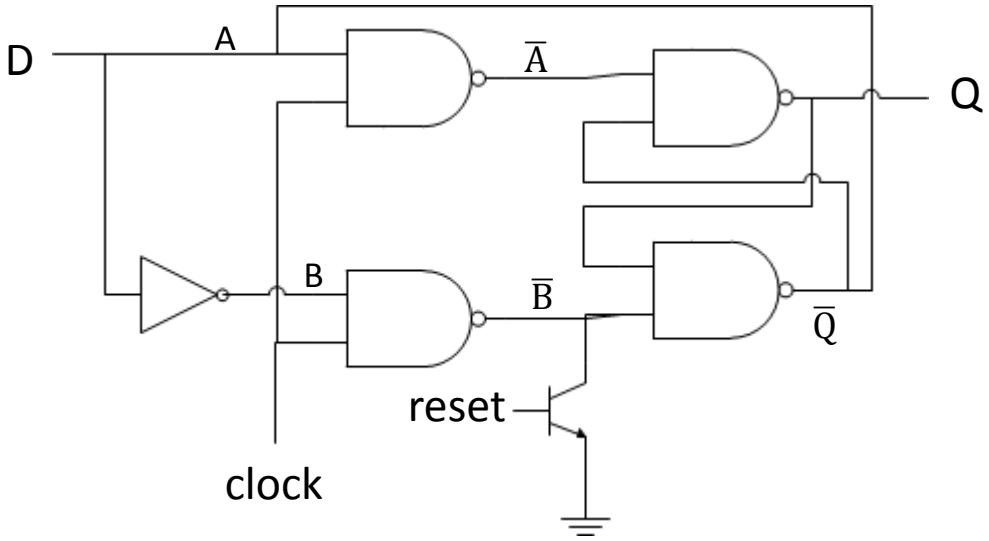
Then suppose clock =1 (high), then $\overline{A}$ = 1 & $\overline{B}$ = 0, which forces Q = 0.

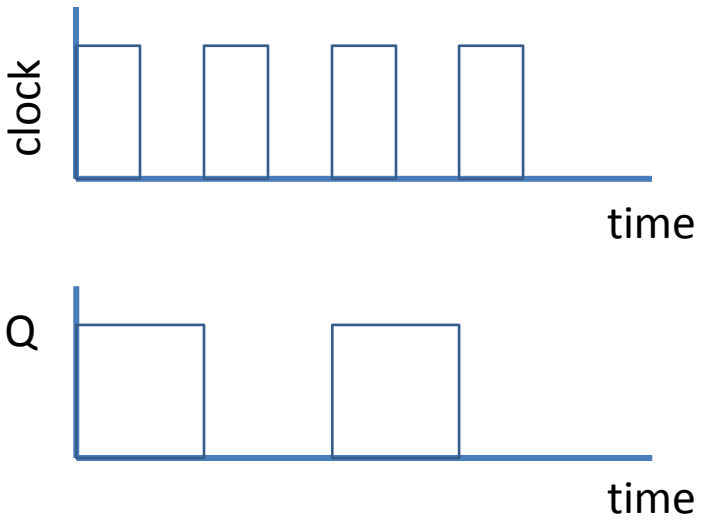
Since Q = 0, then $\overline{Q}$ =1, A = 1 & B = 0.

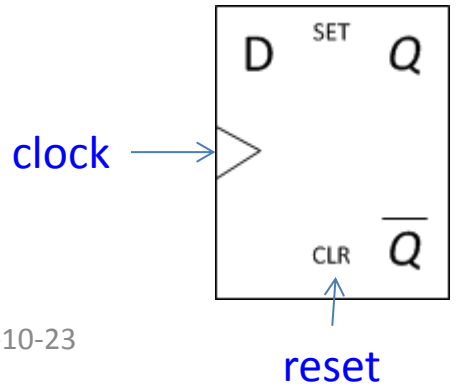
Then suppose clock =1 (high) again, then $\overline{A}$ = 0 & $\overline{B}$ = 1, which forces Q = 1.

Boolean Logic Gates

Toggled Flip-Flops





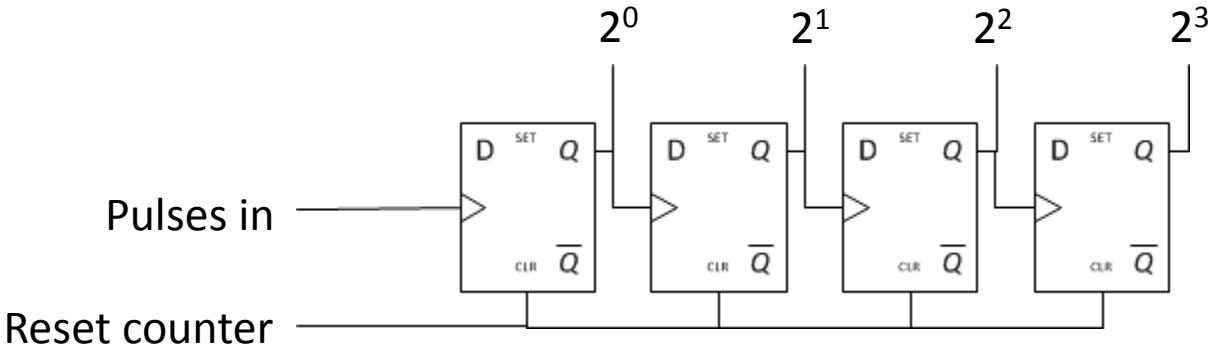


Acts as a divide by 2.

Boolean Logic Gates

Binary Counter (using toggled flip-flops)

If we take for example, four flip-flops, we can make a binary pulse counter.



Each flip-flops acts as a ÷2 and sends its output to the next flip-flop in the line.

This is a 4-bit binary counter. Counts from 0000 to 1111.

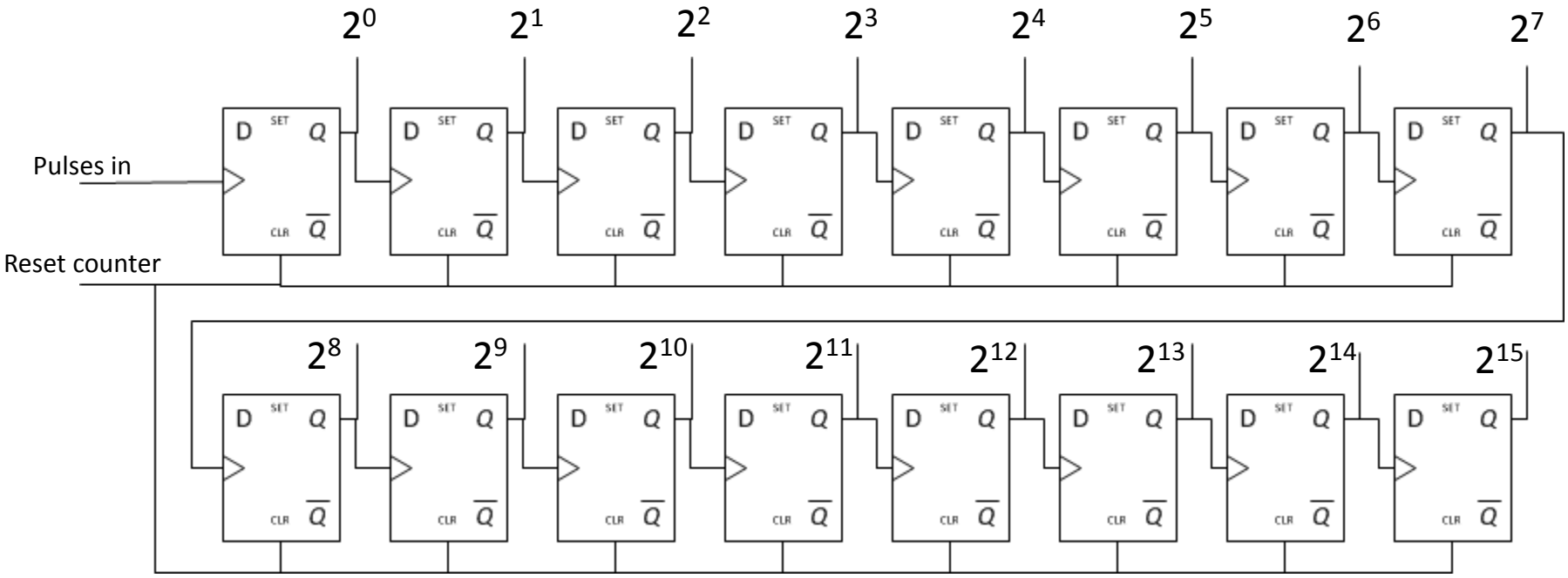
0 15

We reset to 0000 by pulsing the “reset counter” line.

Boolean Logic Gates

Binary Counter (using toggled flip-flops)

Easy to extend to however many bits we need.



This is a 16-bit binary counter. Counts from 0000000000000000 to 1111111111111111.

0 65535

If we doubled this again to make a 32-bit counter, we'd count from 0 to 4,294,967,295

4 Gb

Boolean Logic Gates

Digital Signals

Most common – Transistor Transistor Logic (TTL)

Nowadays, also includes TTL compatible such as CMOS

(CMOS – Complementary Metal Oxide Semiconductor)

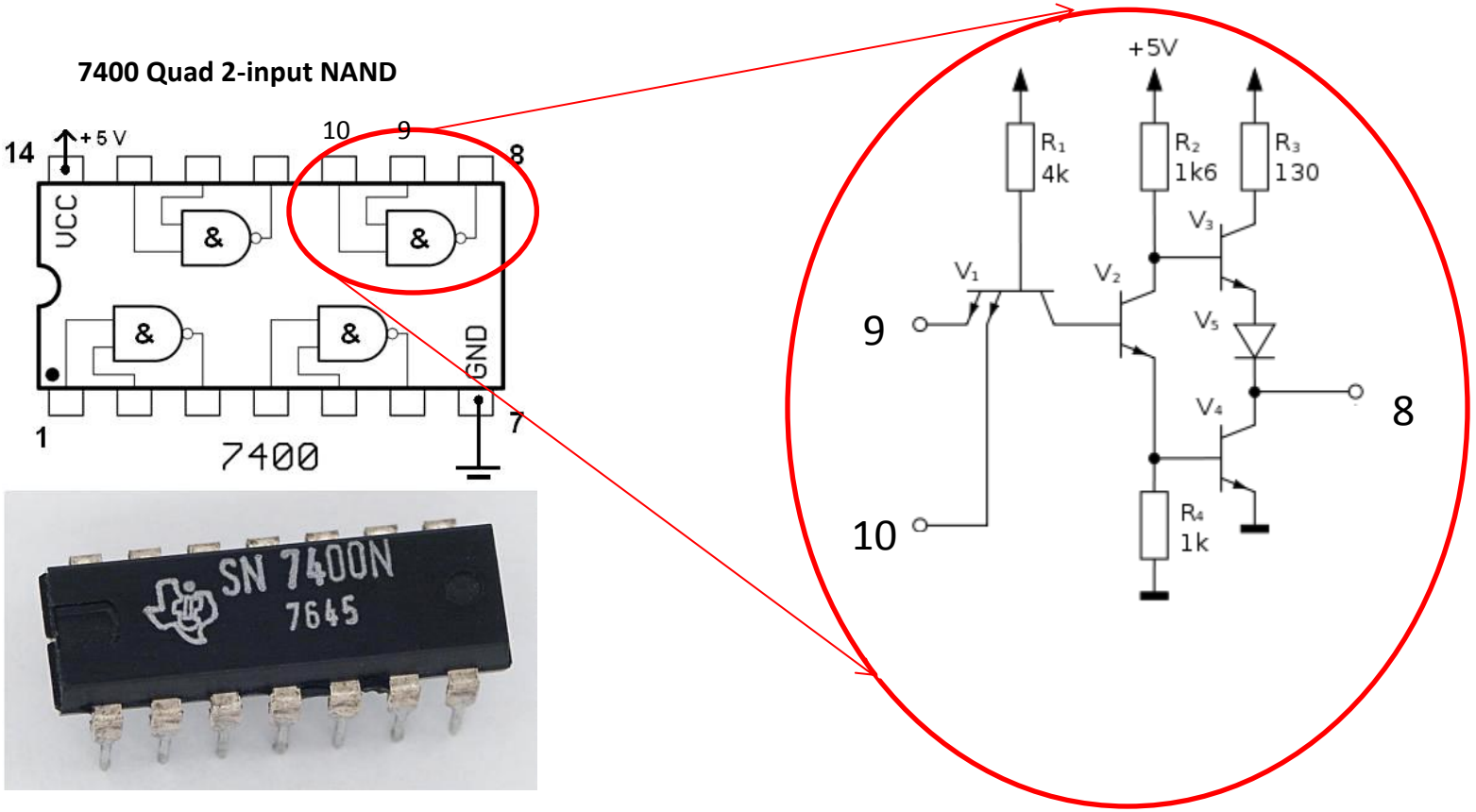
1. Positive Logic (voltage for 1's, no voltage for 0's)
2. 5 volt supply
3. Can source relatively large currents if need be (10's to 100's of mA), which means large fan out.
4. 1 = 5 volts (3.3 – 5 volts)
   0 = 0 volts (0 – 0.5 volts)
   Means that a signal rising from 0 needs to get to at least 3.3 volts to be considered a 1. A signal dropping from above 3.3 needs to drop below 0.5 volts to be considered a 0.
5. Propagation delay ~ 10^{-8} seconds.
   (i.e. Takes ~10nsec for output to reflect inputs.)

Modern CPU's cheat on this – often ~1 volt logic.

Boolean Logic Gates

Digital Signals - TTL

Are more complex functions as well. e.g.



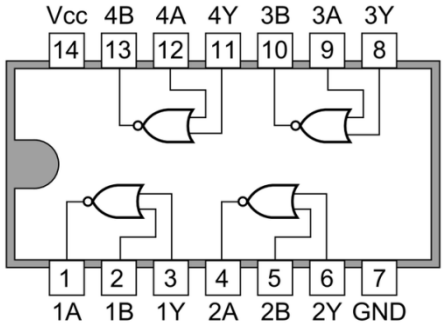
Most are 14-pin IC's, with pin 7 as ground and pin 14 as +5V

Boolean Logic Gates

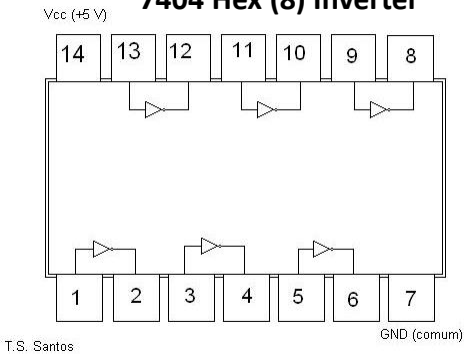
Digital Signals - TTL

Standardized chip set across manufacturers (7400's series):

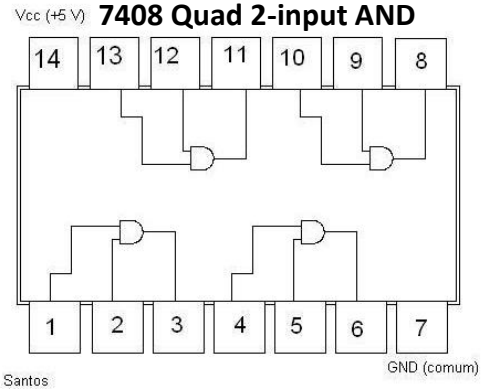
7402 Quad 2-input NOR



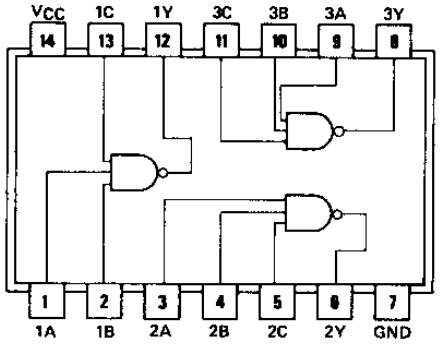
7404 Hex (8) Inverter



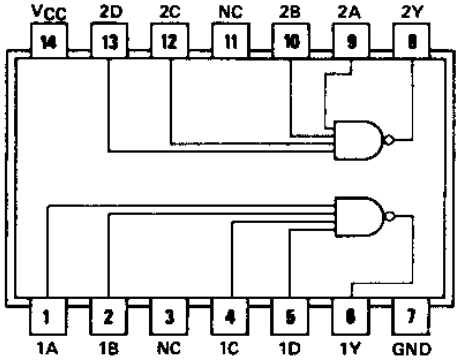
7408 Quad 2-input AND



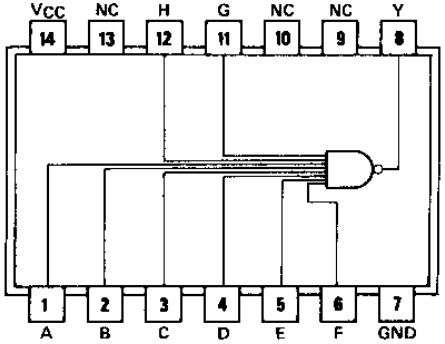
7410 Triple 3-input NAND



7420 Dual 4-input NAND



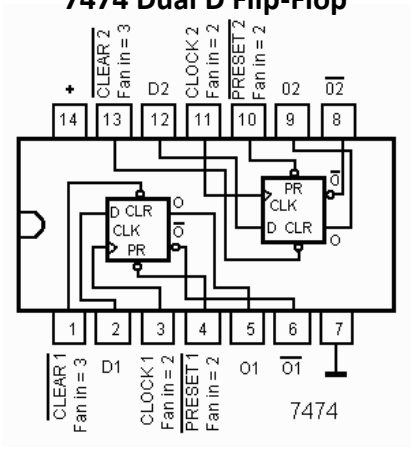
7430 8-input NAND



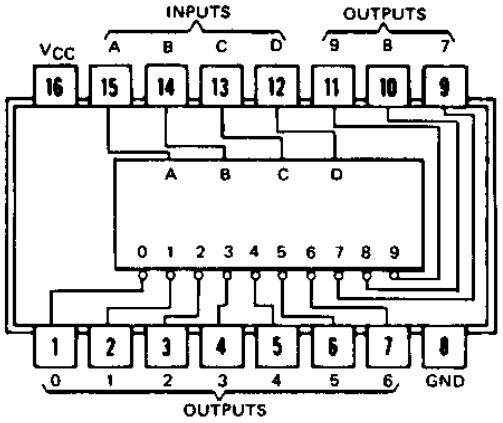
Boolean Logic Gates

Digital Signals - TTL Are more complex functions as well. e.g.

7474 Dual D Flip-Flop

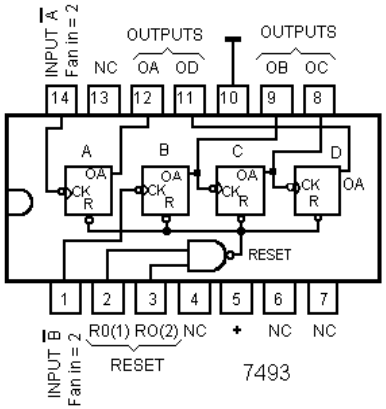


7442 Four Line to 10 line decoder



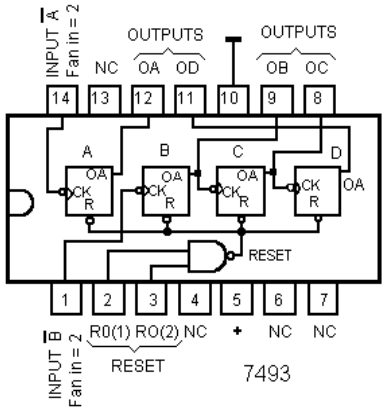
BCD digit input, selects 1 of 10 outputs

7493 4-bit binary counter



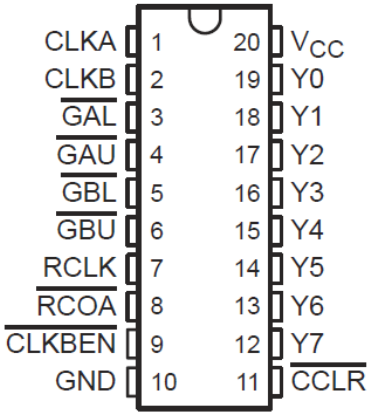
Boolean Logic Gates

7493 4-bit binary counter



So that 16-bit counter we showed earlier can be made with 4 chips.

There are nowadays, as you might expect, chips with “more stuff”.



e.g., SN74LV8154 – Dual 16-bit counter.

If you connect both counters together, form a 32-bit counter.

Only 20-pins: Shares all 32-bits on 8 output pins.

32-bits means counts from 0 to 4,294,967,295