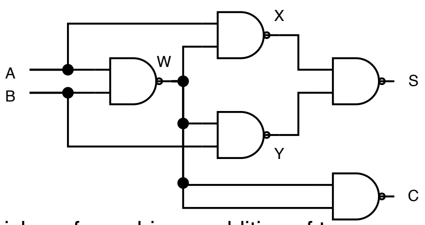
ALevel CS C19 Logic Circuits and Boolean algebra

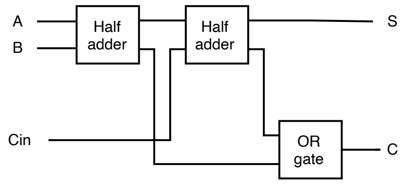
The half adder: a circuit which performs binary addition of two individual bits
sum bit(S) and **carry bit (C)**

Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



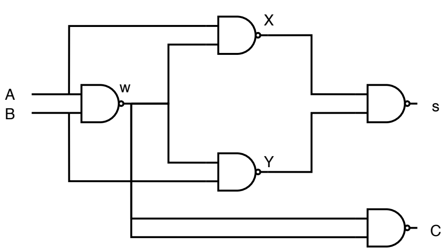
The full adder: a circuit which performs binary addition of two individual bits and an input carry bit. **carry in bit (Cin)**

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



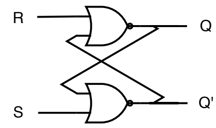
Boolean algebra: a circuit in which the output is dependent only on the input values

Identity/Law	AND form	IR form
Identity	$1.A=A$	$0+A=A$
Null	$0.A=0$	$1+A=1$
Idempotent	$A.A=A$	$A+A=A$
inverse	$A.\overline{A}=0$	$A+\overline{A}=1$
Commutative	$A.B=B.A$	$A+B=B+A$
Associative	$(A.B).C=A.(B.C)$	$(A+B)+C=A+(B+C)$
Distributive	$A+B.C=(A+B).(A+C)$	$A.(B+C)=A.B+A.C$
Absorption	$A.(A+B)=A$	$A+A.B=A$
De Morgan's	$\overline{A.B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}.\overline{B}$
Double Component	$\overline{\overline{A}}=A$	$\overline{\overline{A}}=A$



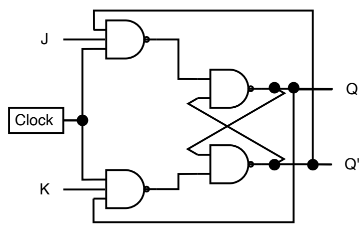
combinational circuits: a circuit in which the output is dependent only on the input values
Sequential circuit: a circuit in which the output depends on the input values and the previous output
SR flip-flop:
S - set R - reset
set state(self-consistent): Q = 1, Q' = 0
unset state: Q = 0, Q' = 1
These properties explain why the SR flip-flop can be used as a storage device for 1 bit and therefore could be used as a component in RAM because a value is stored but can be altered.

Input signals		Initial state		Final state	
S	R	Q	Q'	Q	Q'
0	0	1	0	1	0
1	0	1	0	1	0
0	1	1	0	0	1
0	0	0	1	0	1
1	0	0	1	1	0
0	1	0	1	0	1



JK flip-flop:
difference: if both J and K are input as a 1 then the values for Q and Q' are toggled (they switch value).
This makes the JK flip-flop a more reliable device because there is no combination of input states that leave uncertainty as to which values are stored.

Input signals		Initial state	Final state
J	K	Clock	Q
0	0	↑	Q unchanged
1	0	↑	1
0	1	↑	0
1	1	↑	Q toggles



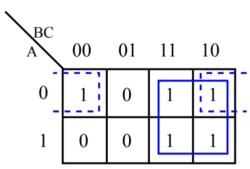
difference SR & JR:
1. SR flip-flop has an invalid combination of S and R. SR allow both Q and Q' to have the same value. SR inputs may arrive at different times.
2.JK flip-flop all four combination of values for J and K are valid.JK does not allow Q and Q' to have the same value.JK incorporates a clock pulse for synchronisation

The role of flip-flop in a computer:
1. A flip-flop can store either a 0 or a 1
2. Computers use bits to store data
3. Flip-flops can therefore be used to store bits (of data)
4. Memory can be created from flip-flops

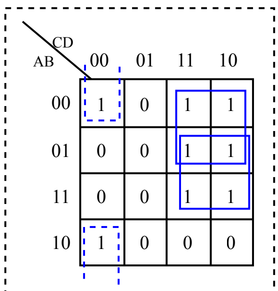
Karnaugh maps(K-maps): a method of creating a Boolean algebra expression from a truth table.
A K-map can make the process much easier than if you use sum-of-products to create minterms. If applied correctly a K-map produces the simplest possible form for the Boolean algebra expression.

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$\overline{A}.\overline{B}.\overline{C}+\overline{A}.B.\overline{C}+\overline{A}.B.C+A.B.C$



$\overline{A}.\overline{C}+B$



Step 1 NAND Truth Table

NAND Truth Table		
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Step 2 Truth Table -> Logic Expression

$$W=\overline{A}.\overline{B}+\overline{A}.B+A.\overline{B}$$

Step 3 Logic Expression X and Y

$$X=\overline{A.W}$$
$$X=\overline{A.(\overline{A}.\overline{B}+\overline{A}.B+A.\overline{B})}$$
$$X=\overline{0+0+A.A.\overline{B}}$$
$$X=\overline{A.\overline{B}}$$
$$X=A+\overline{B}$$
$$Y=\overline{A}+B$$

Step 4 Logic Expression S

$$S=\overline{X.Y}$$
$$S=\overline{X}+\overline{Y}$$
$$S=\overline{(A+\overline{B})}+\overline{(\overline{A}+B)}$$
$$S=\overline{A}.B+A.\overline{B}$$