Description of the Laws of Boolean Algebra

Annulment Law – A term AND‘ed with a “0” equals 0 or OR‘ed with a “1” will equal 1

$A . 0 = 0$ A variable AND’ed with 0 is always equal to 0

$A + 1 = 1$ A variable OR’ed with 1 is always equal to 1

Identity Law – A term OR‘ed with a “0” or AND‘ed with a “1” will always equal that term

$A + 0 = A$ A variable OR’ed with 0 is always equal to the variable

$A . 1 = A$ A variable AND’ed with 1 is always equal to the variable

Idempotent Law – An input that is AND‘ed or OR´ed with itself is equal to that input

$A + A = A$ A variable OR’ed with itself is always equal to the variable

$A . A = A$ A variable AND’ed with itself is always equal to the variable

Complement Law – A term AND‘ed with its complement equals “0” and a term OR´ed with its complement equals “1”

$A . A = 0$ A variable AND’ed with its complement is always equal to 0

$A + A = 1$ A variable OR’ed with its complement is always equal to 1

Commutative Law – The order of application of two separate terms is not important

$A . B = B . A$ The order in which two variables are AND’ed makes no difference

$A + B = B + A$ The order in which two variables are OR’ed makes no difference

Double Negation Law – A term that is inverted twice is equal to the original term

$A = A$ A double complement of a variable is always equal to the variable

de Morgan’s Theorem – There are two “de Morgan’s” rules or theorems,

(1) Two separate terms NOR‘ed together is the same as the two terms inverted (Complement) and AND‘ed for example: $A+B = A \, . \, B$

(2) Two separate terms NAND‘ed together is the same as the two terms inverted (Complement) and OR‘ed for example: $A.B = A + B$

Other algebraic Laws of Boolean not detailed above include:

Boolean Postulates – While not Boolean Laws in their own right, these are a set of Mathematical Laws which can be used in the simplification of Boolean Expressions.

$0 \, . \, 0 = 0$ A 0 AND’ed with itself is always equal to 0

$1 \, . \, 1 = 1$ A 1 AND’ed with itself is always equal to 1

$1 \, . \, 0 = 0$ A 1 AND’ed with a 0 is equal to 0

$0 + 0 = 0$ A 0 OR’ed with itself is always equal to 0

$1 + 1 = 1$ A 1 OR’ed with itself is always equal to 1

$1 + 0 = 1$ A 1 OR’ed with a 0 is equal to 1

$1 = 0$ The Inverse (Complement) of a 1 is always equal to 0

$0 = 1$ The Inverse (Complement) of a 0 is always equal to 1

Distributive Law – This law permits the multiplying or factoring out of an expression.

$A(B + C) = A.B + A.C$ (OR Distributive Law)

$A + (B.C) = (A + B).(A + C)$ (AND Distributive Law)

Absorptive Law – This law enables a reduction in a complicated expression to a simpler one by absorbing like terms.

A + (A.B) = (A.1) + (A.B) = A(1 + B) = A (OR Absorption Law)

A(A + B) = (A + 0).(A + B) = A + (0.B) = A (AND Absorption Law)

Associative Law – This law allows the removal of brackets from an expression and regrouping of the variables.

A + (B + C) = (A + B) + C = A + B + C   (OR Associate Law)

A(B.C) = (A.B)C = A . B . C   (AND Associate Law)

Boolean Algebra Functions

Using the information above, simple 2-input AND, OR and NOT Gates can be represented by 16 possible functions as shown in the following table.

what are glitches in digital logic circuits

A glitch happens sometimes as signals propagate through combinational logic.
Before reaching the final steady state value, sometimes, a wrong intermediate value shows up at the output during the settling phase. This is called a glitch.

The reason why it happens is because of the different signal paths have different delays. So when one input reaches the input of a gate while the other input is delayed, the wrong output may occur. The output signal will be corrected once all of the correct input has propagated through.

glitch in digital logic

gleaches are unwanted signals that occur in the circuit

the gleaches can observed in flipflops

gleach will damage the circuit at times.

How use two flip-flops on same clock

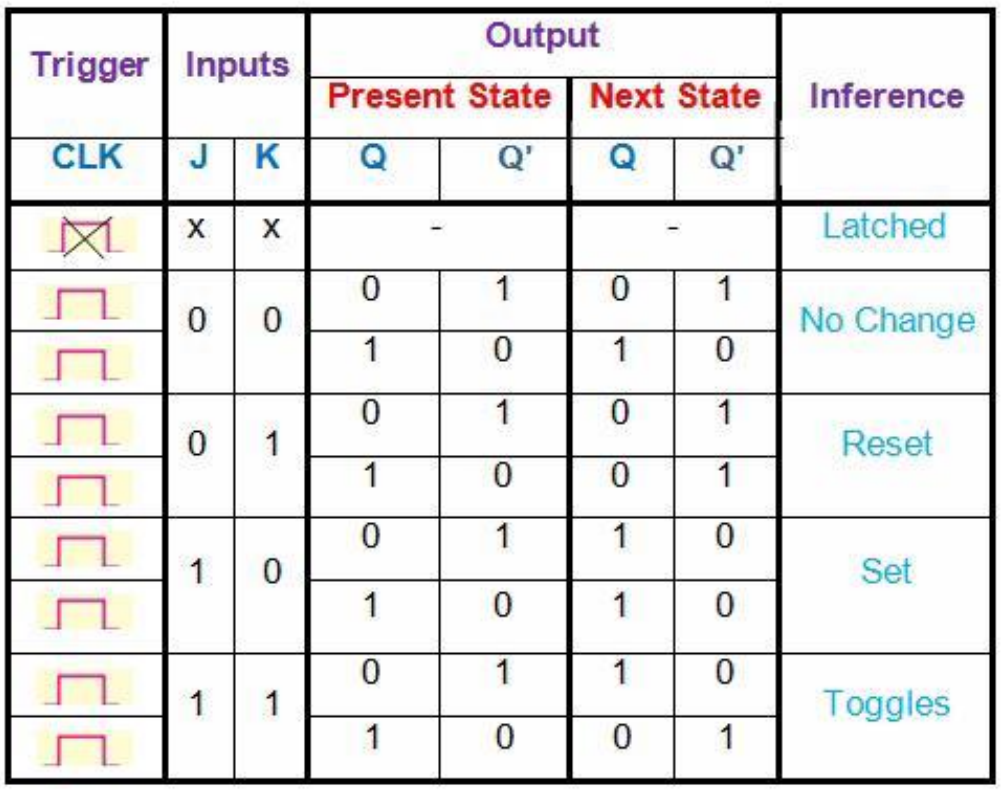
Master-Slave JK Flip-flop

The master-slave flip-flop eliminates all the timing problems by using two SR flip-flops connected together in a series configuration. One flip-flop acts as the “Master” circuit, which triggers on the leading edge of the clock pulse while the other acts as the “Slave” circuit, which triggers on the falling edge of the clock pulse. This results in the two sections, the master section and the slave section being enabled during opposite half-cycles of the clock signal.

Example

The TTL 74LS73 is a Dual JK flip-flop IC,

Table



Trigger	Inputs		Output				Inference
			Present State		Next State		
CLK	J	K	Q	Q'	Q	Q'	
⊠	x	x	-		-		Latched
⎍	0	0	0	1	0	1	No Change
⎍			1	0	1	0	
⎍	0	1	0	1	0	1	Reset
⎍			1	0	0	1	
⎍	1	0	0	1	1	0	Set
⎍			1	0	1	0	
⎍	1	1	0	1	1	0	Toggles
⎍			1	0	0	1	