Chapter 6 Summary
Digital Circuits

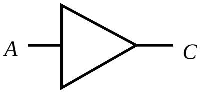
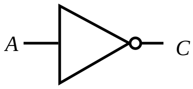
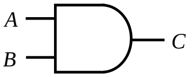
base b number:

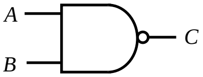
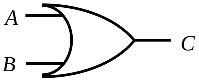
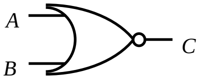
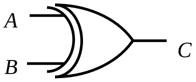
$$(d_{n-1}\ldots d_3 d_2 d_1 d_0)_b = (d_{n-1} \cdot b^{n-1} + \cdots + d_2 \cdot b^2 + d_1 \cdot b^1 + d_0 \cdot b^0)$$

Common base equivalents:

Binary	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Combinational logic operations:

Gate	Operation	Symbol	Expression	Truth table
buffer	increase output signal current		$C = A$	A C 0 0 1 1
inverter (INV, NOT)	invert signal (complement)		$C = \bar{A}$	A C 0 1 1 0
AND gate	AND logic		$C = A \cdot B$	A B C 0 0 0 0 1 0 1 0 0 1 1 1

NAND gate	inverted AND logic		$C = \overline{A \cdot B}$	$A\ B\ C$ 0 0 1 0 1 1 1 0 1 1 1 0
OR gate	OR logic		$C = A + B$	$A\ B\ C$ 0 0 0 0 1 1 1 0 1 1 1 1
NOR gate	inverted OR logic		$C = \overline{A + B}$	$A\ B\ C$ 0 0 1 0 1 0 1 0 0 1 1 0
XOR Gate	exclusive OR logic		$C = A \oplus B$	$A\ B\ C$ 0 0 0 0 1 1 1 0 1 1 1 0

Boolean Algebra Laws and Identities:

Fundamental Laws:

OR	AND	NOT
$A + 0 = A$	$A \cdot 0 = 0$	
$A + 1 = 1$	$A \cdot 1 = A$	$\overline{\overline{A}} = A$
$A + A = A$	$A \cdot A = A$	(double inversion)
$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$	

Commutative Laws:

$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

Associative Laws:

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

Distributive Laws

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

DeMorgan's Laws:

$$\overline{A+B+C+\cdots} = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\cdots$$
$$\overline{A\cdot B\cdot C\cdot\cdots} = \overline{A}+\overline{B}+\overline{C}+\cdots$$
$$A+B+C+\cdots = \overline{\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\cdots}$$
$$A\cdot B\cdot C\cdot\cdots = \overline{\overline{A}+\overline{B}+\overline{C}+\cdots}$$

steps in basic logic design:

1. Define the problem in words.
2. Write quasi-logic statements in English that can be translated into Boolean expressions.
3. Write the Boolean expressions.
4. Simplify and optimize the Boolean expressions if possible.
5. Write an all-AND or all-NAND or all-OR or all-NOR realization of the circuit to minimize the number of required logic gate IC components.
6. Draw the logic schematic for the electronic realization of the circuit.

converting truth tables to logic expressions:

sum-of-products method: form a product for every row in the truth table that results in an output of 1 and take the sum of the products

product-of-sums method: form a sum for every row in the truth table that results in an output of 0 and take the product of the sums

RS flip-flop:

Inputs		**Outputs**	
S	***R***	***Q***	$\overline{Q}$
0	0	Q_0	$\overline{Q_0}$
1	0	1	0
0	1	0	1
1	1	NA	

positive edge-triggered RS flip-flop:

S	***R***	***CK***	***Q***	$\overline{Q}$
0	0	↑	Q_0	$\overline{Q_0}$
1	0	↑	1	0
0	1	↑	0	1
1	1	↑	NA	
x	x	0,1	Q_0	$\overline{Q_0}$

positive edge-triggered D flip-flop:

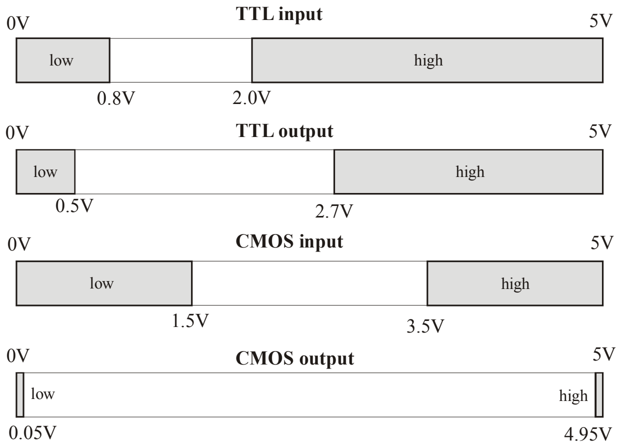
D	***CK***	***Q***	$\overline{Q}$
0	↑	0	1
1	↑	1	0
x	0	Q_0	$\overline{Q_0}$
x	1	Q_0	$\overline{Q_0}$

negative edge-triggered JK flip-flop:

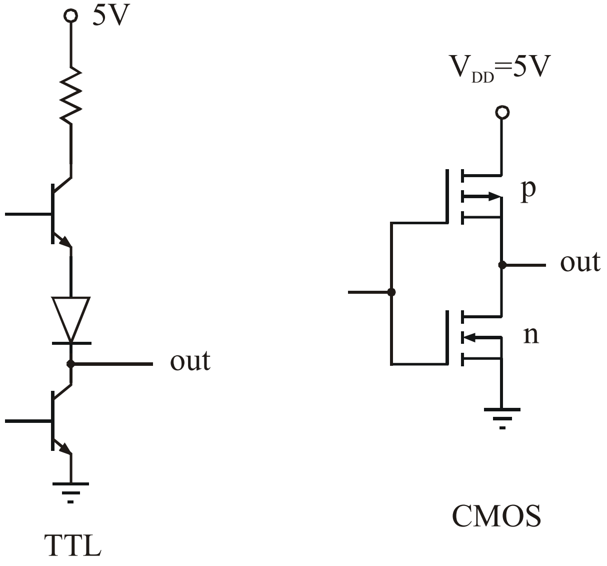
$\overline{\textbf{Preset}}$	$\overline{\textbf{Clear}}$	***CK***	***J***	***K***	***Q***	$\overline{Q}$
0	1	x	x	x	1	0
1	0	x	x	x	0	1
0	0	NA				
1	1	↓	0	0	Q_0	$\overline{Q_0}$
1	1	↓	1	0	1	0
1	1	↓	0	1	0	1
1	1	↓	1	1	toggle	
1	1	0, 1	x	x	Q_0	$\overline{Q_0}$

positive edge-triggered T flip-flop:

T	$\overline{\textbf{Preset}}$	$\overline{\textbf{Clear}}$	***Q***	$\overline{Q}$
↑	1	1	$\overline{Q_0}$	Q_0
0	1	1	Q_0	$\overline{Q_0}$
1	1	1	Q_0	$\overline{Q_0}$
x	0	1	1	0
x	1	0	0	1



TTL and CMOS input and output levels



TTL and CMOS output circuits