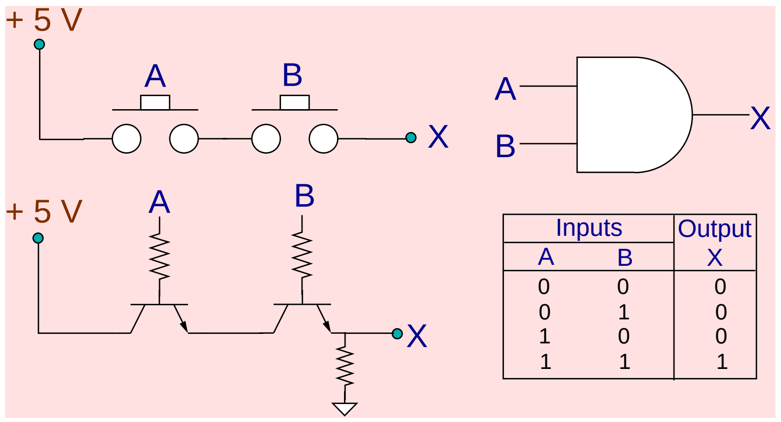
Digital Electronics

Prof. Dr. M. Zahurul Haq
zahurul@me.buet.ac.bd
http://teacher.buet.ac.bd/zahurul/

Department of Mechanical Engineering
Bangladesh University of Engineering & Technology

ME 475: Mechatronics

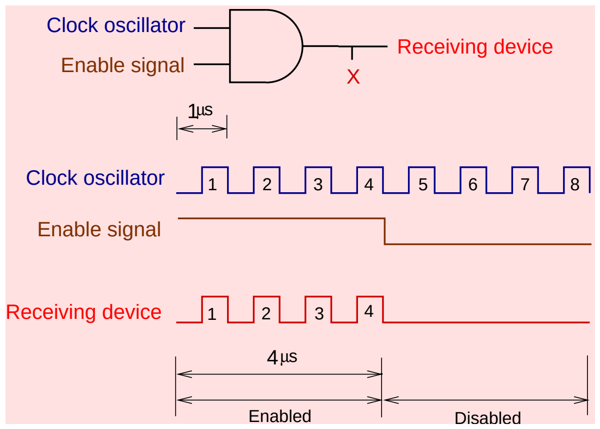
AND Gate



Inputs		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

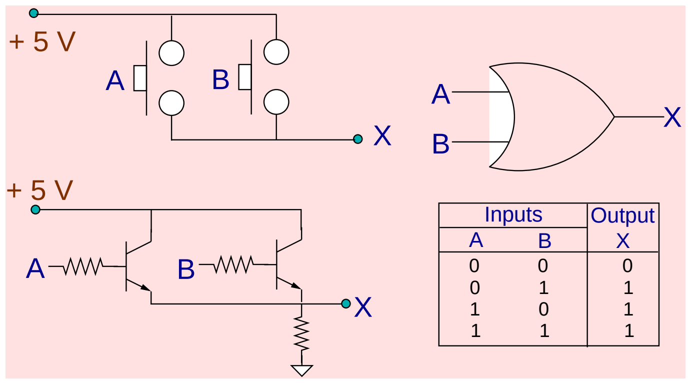
- Output, X, is HIGH only if inputs A and B are both HIGH.
- Boolean Equation: $X = A \text{ AND } B = A \cdot B = AB$

. . . contd.



- AND gate can be used to **enable** a waveform to transmit from one point to another. The LOW value disables the clock from reaching the X-output.

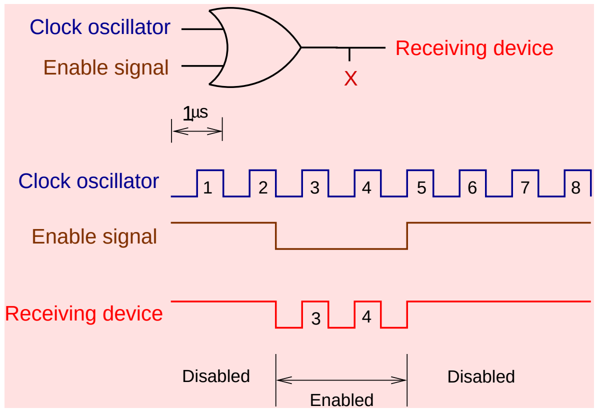
OR Gate



Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

- Output, X, is HIGH if input A or input B is HIGH or both are HIGH.
- Boolean Equation: $X = A \text{ OR } B = A + B$

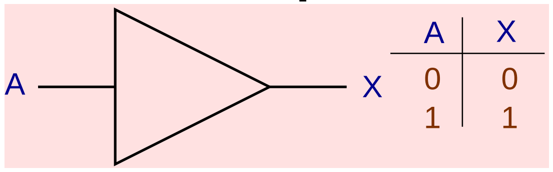
. . . contd.



- OR gate can be used to **disable** a waveform from transmitting from one point to another.

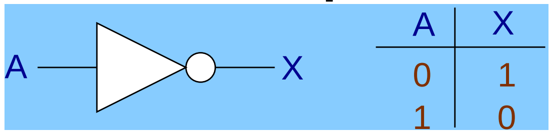
Buffer & Inverter ICs

- **Buffer IC**: Boolean Equation: $X = A$



A	X
0	0
1	1

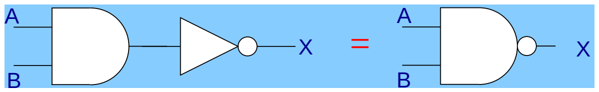
- **Inverter IC**: Boolean Equation: $X = \overline{A}$



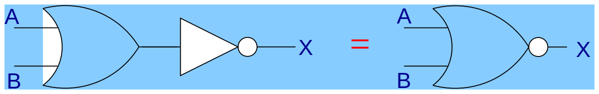
A	X
0	1
1	0

NAND & NOR Gates

- **NAND**: Boolean Equation: $X = \overline{AB}$
- Output is always HIGH unless both inputs are HIGH.



- **NOR**: Boolean Equation: $X = \overline{A + B}$
- Output is always LOW unless both inputs are LOW.



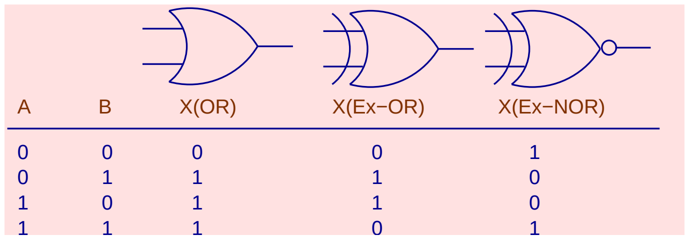
Ex-OR/Ex-NOR Gates

- Exclusive-OR (Ex-OR) gate provides a HIGH output if one input or the other input is HIGH, but not the both.

Ex-OR: $X = A \oplus B = \overline{A}\,B + A\,\overline{B}$

- Ex-NOR is the compliment of the Ex-OR. It provides a HIGH output for both inputs HIGH or both inputs LOW.

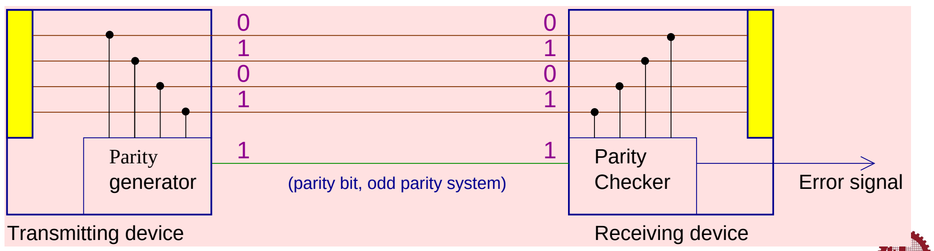
Ex-NOR: $X = \overline{A \oplus B} = AB + \overline{A}\,\overline{B}$



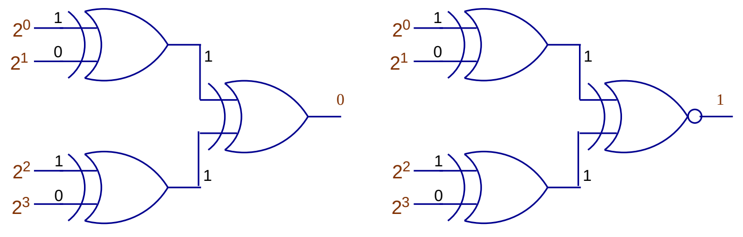
A	B	X(OR)	X(Ex−OR)	X(Ex−NOR)
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

Parity Generator/Checking

- In digital data transmission external noise can cause an error. Parity system is used to recognise the error and take corrective measures (retransmission).
- Parity system puts an extra bit to the digital transmission.
- odd parity: sum of all bits is odd.
- even parity: sum of all bits is even.

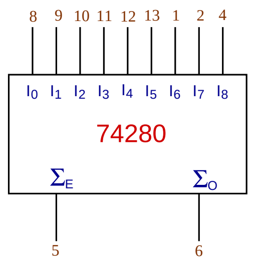


. . . contd.



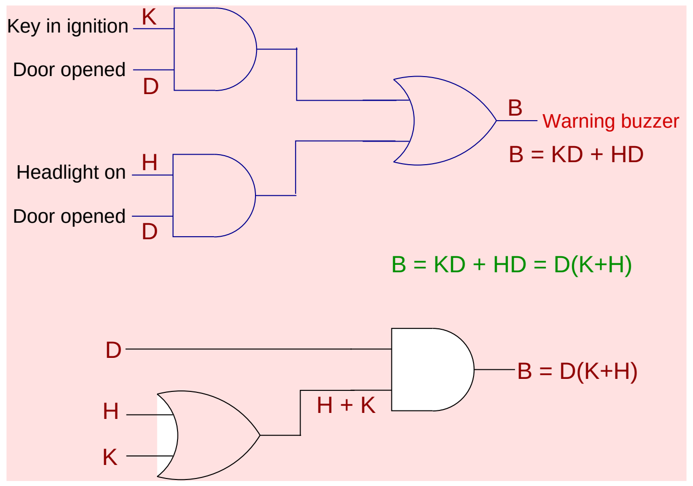
Even parity generator

Odd parity generator



I_0 – I_8	Σ_E	Σ_o
Even	HIGH	LOW
Odd	LOW	HIGH

Combinational Logic Example



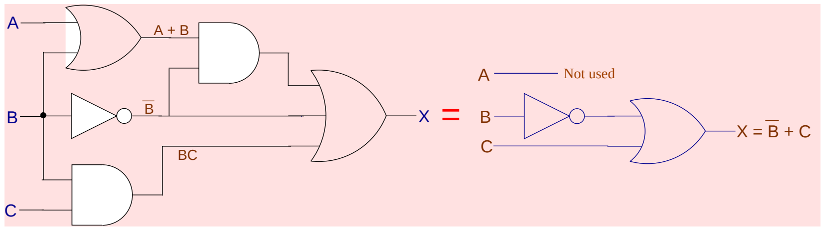
Boolean Algebra Laws

1. Commutative law of addition: $A + B = B + A$, and multiplication: $AB = BA$.
   These laws mean that the order of ORing and ANDing does not matter.
2. Associative law of addition: $A + (B + C) = (A + B) + C$, and multiplication: $A(BC) = (AB)C$.
   These laws mean that the grouping of several variables ORed or ANDed together does not matter.
3. Distributive law: $A(B + C) = AB + BC$, and $(A + B)(C + D) = AC + AD + BC + BD$.
   These laws show methods for expanding and equation containing ORs and ANDs.

Boolean Algebra Rules

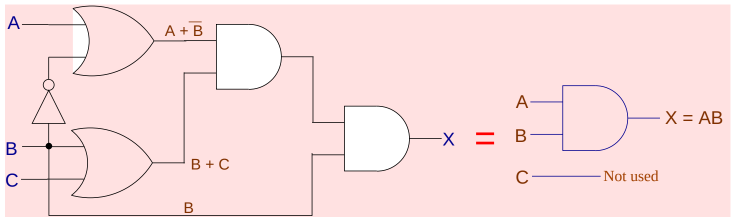
1. Anything ANDed with a 0 is equal to 0 ($A \cdot 0 = 0$).
2. Anything ANDed with a 1 is equal itself ($A \cdot 1 = A$).
3. Anything ORed with a 0 is equal itself ($A + 0 = A$).
4. Anything ORed with a 1 is equal to 1 ($A + 1 = 1$).
5. Anything ANDed with itself is equal itself ($A \cdot A = A$).
6. Anything ORed with itself is equal itself ($A + A = A$).
7. Anything ANDed with its own compliment equals 0 ($A \cdot \overline{A} = 0$).
8. Anything ORed with its own compliment equals 1 ($A + \overline{A} = 1$).
9. A variable that is complemented twice will return to its original logic level ($\overline{\overline{A}} = A$) .
10. (a) $A + \overline{A}B = A + B$
    (b) $\overline{A} + AB = \overline{A} + B$

Reduction of Logic Circuits: Example 1



$$\begin{aligned}
X &= (A+B)\overline{B} + \overline{B} + BC & \\
&= A\overline{B} + B\overline{B} + \overline{B} + BC & \text{Law 3} \\
&= A\overline{B} + 0 + \overline{B} + BC & \text{Rule 7} \\
&= A\overline{B} + \overline{B} + BC & \text{Rule 3} \\
&= \overline{B}(A+1) + BC & \text{Factorisation} \\
&= \overline{B} \cdot 1 + BC & \text{Rule 4} \\
&= \overline{B} + BC & \text{Rule 2} \\
&= \overline{B} + C & \text{Rule 10(b)}
\end{aligned}$$

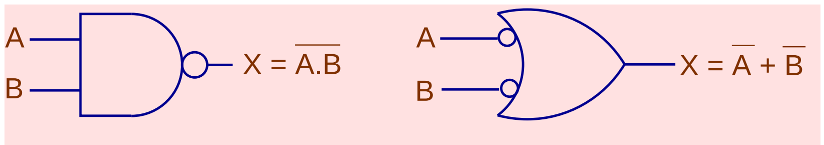
Reduction of Logic Circuits: Example 2



$$\begin{aligned}
X &= [(A+\overline{B})(B+C)]B & \\
&= (AB + AC + \overline{B}B + \overline{B}C)B & \text{Law 3} \\
&= (AB + AC + 0 + \overline{B}C)B & \text{Rule 7} \\
&= (AB + AC + \overline{B}C)B & \text{Rule 3} \\
&= ABB + ACB + \overline{B}CB & \text{Law 3} \\
&= ABB + ABC + \overline{B}BC & \text{Law 1} \\
&= AB + ABC = AB(1+C) & \text{Rule 5, 7 \& 1; \& then factorisation} \\
&= AB & \text{Rule 2}
\end{aligned}$$

De Morgan's Theorem 1

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



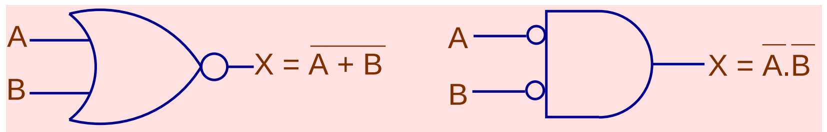
A	B	X = $\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X = $\overline{A}$ + $\overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

$\longleftarrow$ Equivalent Result $\longrightarrow$

De Morgan's Theorem 2

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

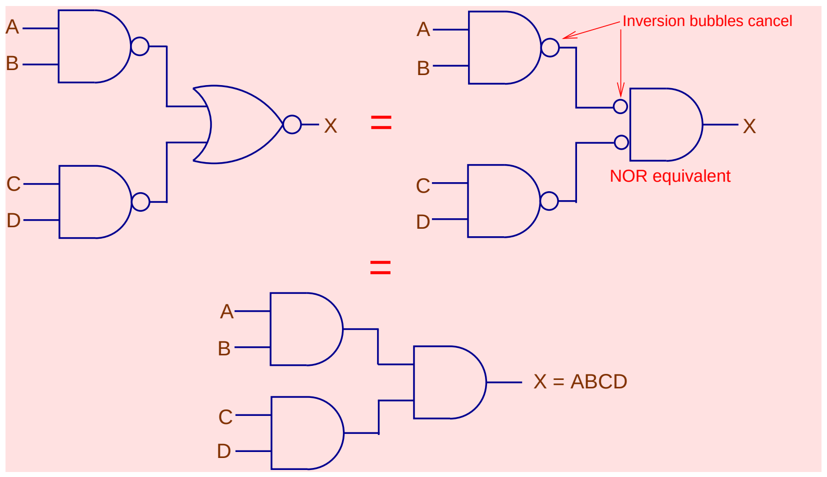


A	B	X = $\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

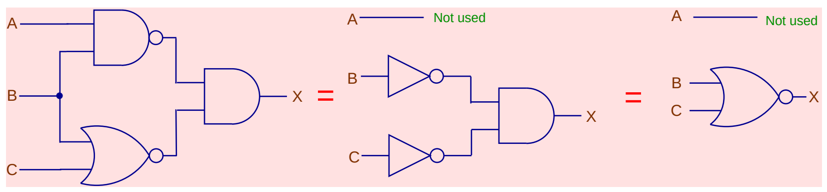
A	B	X = $\overline{A}.\overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

Equivalent Result

De Morgan's Theorem: Application



De Morgan's Theorem: Application

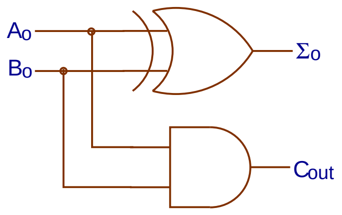


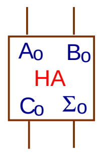
$$\begin{aligned}
X &= \overline{\overline{AB}\cdot\overline{B+C}} \\
&= (\overline{A}+\overline{B})\cdot\overline{B}\;\overline{C} && \text{De Morgan's theorem} \\
&= \overline{A}\;\overline{B}\;\overline{C}+\overline{B}\;\overline{B}\;\overline{C} \\
&= \overline{A}\;\overline{B}\;\overline{C}+\overline{B}\;\overline{C} \\
&= \overline{B}\;\overline{C}(\overline{A}+1) \\
&= \overline{B}\;\overline{C} \\
&= \overline{B+C}
\end{aligned}$$

Binary Addition of LSB

$$\boxed{A_o + B_o = \Sigma_o + C_{out}}$$

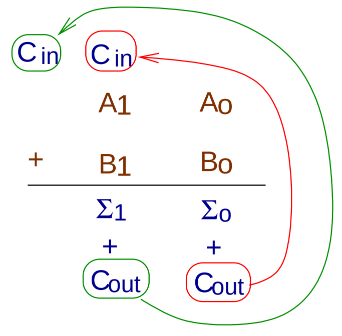
A_o	B_o	Σ_o	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





- Addition of binary & decimal numbers are similar. The binary sum is made up of only 1's and 0's.
- For LSB bit, half adder (HA) circuit is used which does not have a carry-in bit from a previous digit.
- In other bits, if a carry-out is produced, it is added to the next-more-significant column as carry-in, C_{in}.

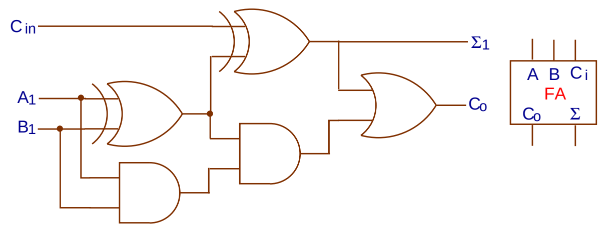
. . . contd.



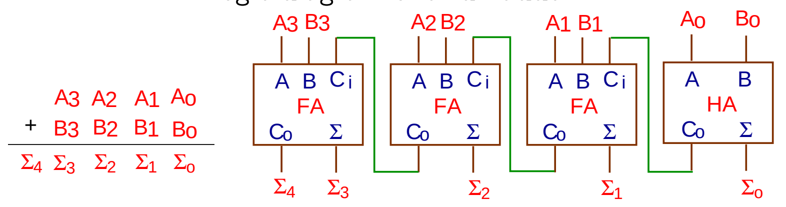
A_1	B_1	C_{in}	Σ_1	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Decimal	Binary
31	0001 1111
+7	+0000 0111
38	0010 0110

Binary Addition contd.



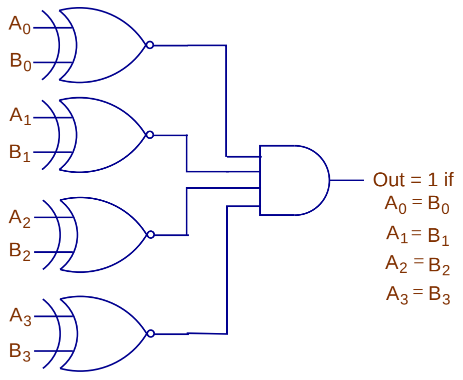
Logic diagram of a full-adder.



Block diagram of a 4-bit binary adder.

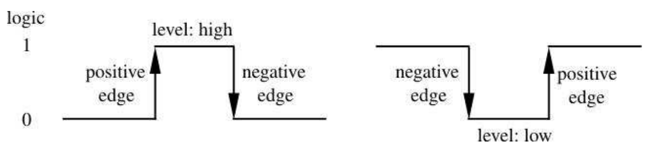
Comparators

- The basic comparator evaluates two binary strings bit by bit and outputs 1 if they are exactly the same.
- Ex-NOR is the easiest way to compare equality of bits.



Combinational vs. Sequential Logic

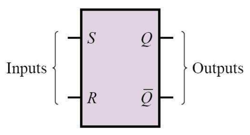
- In combinational logic ckt, state depends upon the actual signals being applied to their inputs at that time.
- Sequential logic circuits have some form of inherent memory and these are able to take into account their previous input state as well as those actually present.
- Sequential logic ckts can be divided into 3 main groups:
  1. Clock driven: ckts are synchronised to specific CLK signal.
  2. Even driven: asynchronous ckts to react when external event occurs.
  3. Pulse driven: combination of synchronous & asynchronous.



- +ve edge trigger devices respond to low-to-high transition.
- -ve edge trigger devices respond to high-to-low transition.

S-R Flip-Flop

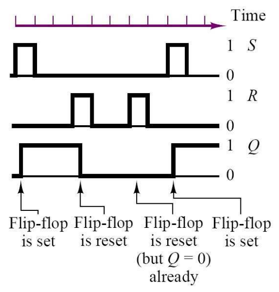
- SR flip-flop be constructed using two NOR or NAND gates.
- $S = 1,\ R = 0 \implies Q = 1\ \&\ \bar{Q} = 0$: Set condition.
- S removed: $S = 0,\ R = 0 \implies Q = 1\ \&\ \bar{Q} = 0$: Hold.
- $S = 0,\ R = 1 \implies Q = 0\ \&\ \bar{Q} = 1$: Reset
- $S = 1,\ R = 1$: Not used.



S	R	Q
0	0	Present state
0	1	Reset
1	0	Set
1	0	Disallowed

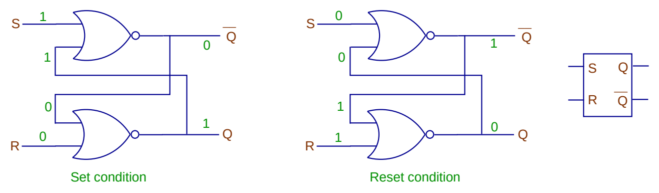
e587.eps

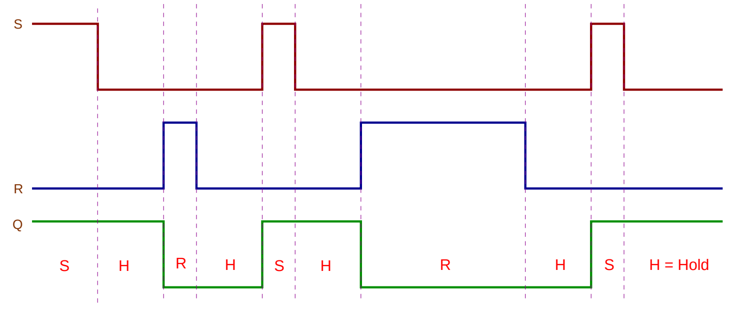
S	R	Q
1	0	1
0	0	1
0	0	1
0	1	0
0	0	0
0	0	0
0	1	0
0	0	0
1	0	1
0	0	1



e588.eps

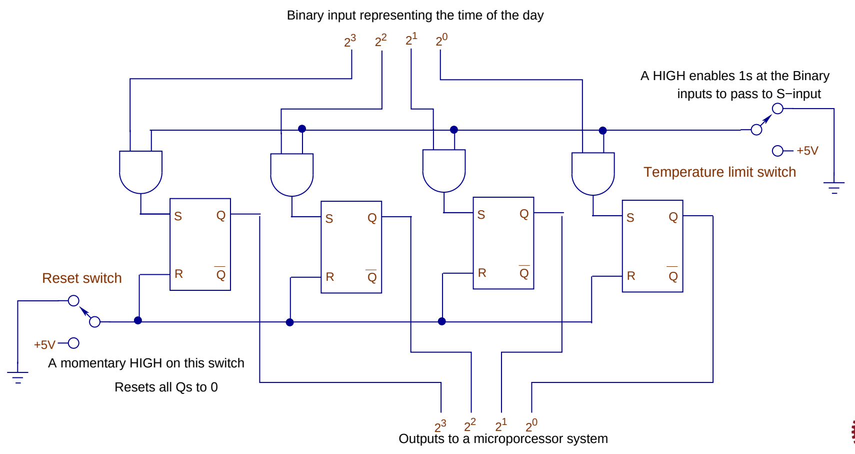
. . . contd.





. . . Applications

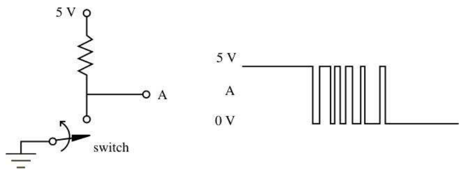
To register the binary value representing the time when a temperature limit switch goes into a HIGH.



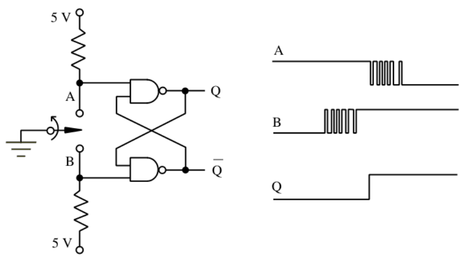
Binary input representing the time of the day
2³ 2² 2¹ 2⁰
A HIGH enables 1s at the Binary inputs to pass to S–input
+5V
Temperature limit switch
Reset switch
+5V
A momentary HIGH on this switch Resets all Qs to 0
S Q R Q̄
2³ 2² 2¹ 2⁰
Outputs to a microporcessor system

. . . Switch Bounce & De-bouncer

When switches are opened or closed, there are brief current oscillations due to mechanical bouncing or electrical arcing.

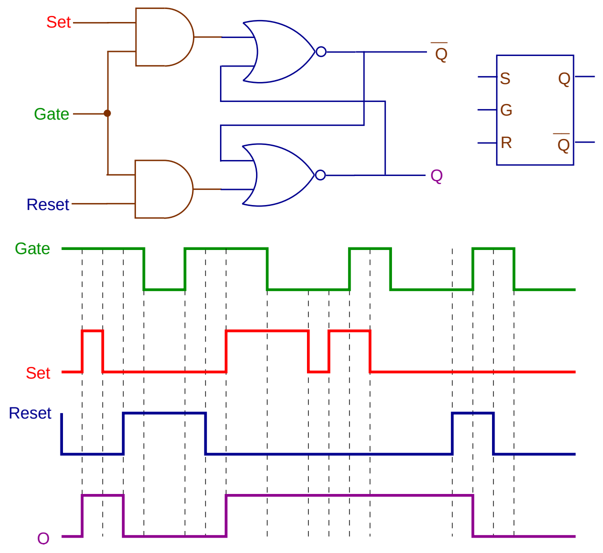


e586.eps



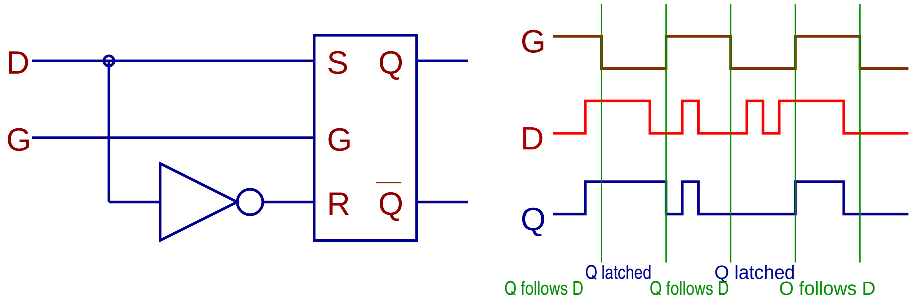
e291.eps

Gated S-R Flip Flop



Data Latch

A level-triggered flip-flop is a latch. Data latch can be formed from the gated S-R Flip-flop by the addition of an inverter; this enables just a single input (D) to latch the previous input value.



. . . contd.

- Level-triggered devices respond to their inputs while the clock signal is at a high level and retain their output values after the level changes.
- The output Q tracks the input D while CK is high. At the negative edge (i.e. when CK goes low), the flip-flop output will hold or latch the value D had at the edge transition.

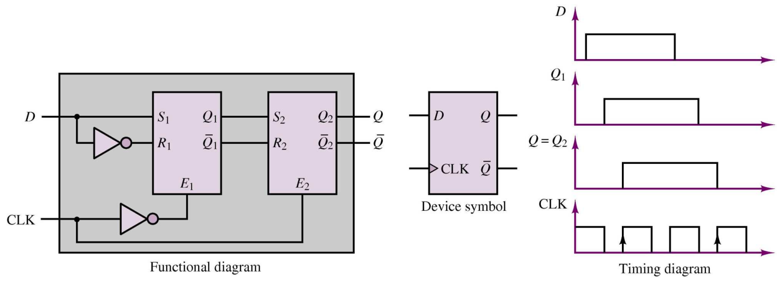
D	CK	Q	$\bar{Q}$
0	1	0	1
1	1	1	0
x[1]	0	Q_o	$\bar{Q}_o$

[1] the value of D has no effect on the output as long as CK is low

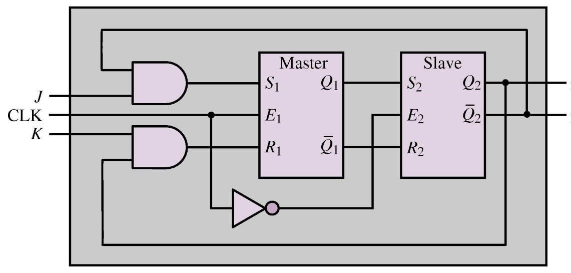
D Flip-Flop

- It has a single input D whose value is stored and presented at the output Q at the +ve or -ve edge of CLK.

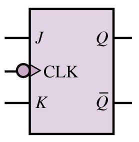
D	CK	Q	$\bar{Q}$
0	$\uparrow$	0	1
1	$\uparrow$	1	0
x	0	Q_o	$\bar{Q}_o$
x	1	Q_o	$\bar{Q}_o$



JK Flip-flop



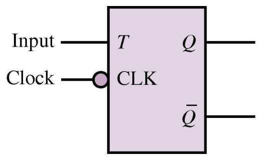
Functional diagram



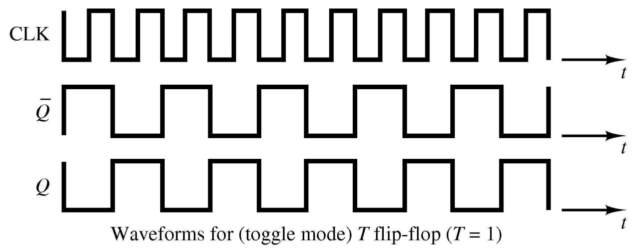
Device symbol

- When J and K are both low, no change in state occurs.
- When J = 0 and K = 1, the flip-flop is reset to 0.
- When J = 1 and K = 0, the flip-flop is set to 1.
- When both J and K are high, the flip-flop will toggle between states at every -ve edge of CLK.

T Flip-flop



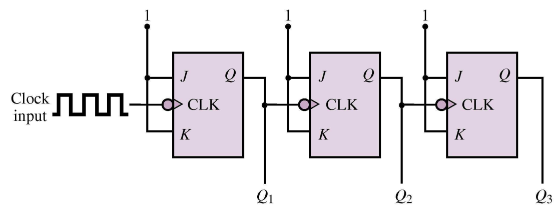
T flip-flop



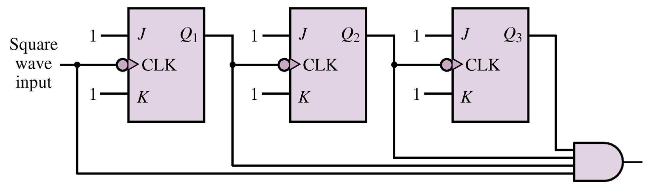
Waveforms for (toggle mode) *T* flip-flop ($T = 1$)

T flip-flop is a JK flip-flop with its inputs tied together. It toggles between the high and low state at half of the clock frequency. It can be used as a *divide by 2 counter.*

3-bit Binary Counter

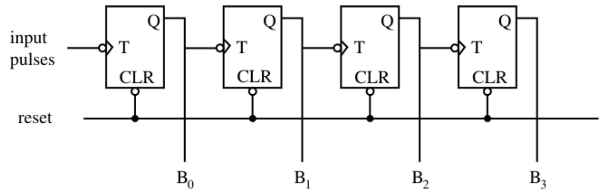


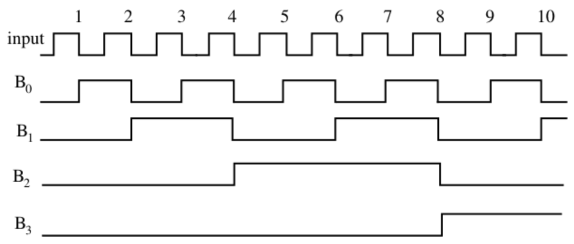
Input	Q_3	Q_2	Q_1
⎍	0	0	0
⎍	0	0	1
⎍	0	1	0
⎍	0	1	1
⎍	1	0	0
⎍	1	0	1
⎍	1	1	0
⎍	1	1	1



A 3-bit ripple counter can be configured as a divide-by-8 mechanism, simply by adding an AND gate.

4-bit Binary Counter



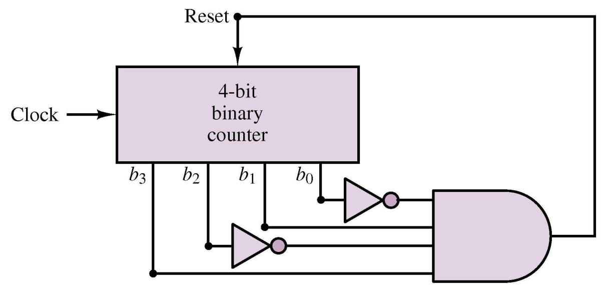


The ckt may be used as a **frequency divider**. Output B_0, B_1, B_2, B_3 are divide by -2, -4, -8 & -16 outputs, respectively.

Decade Counter

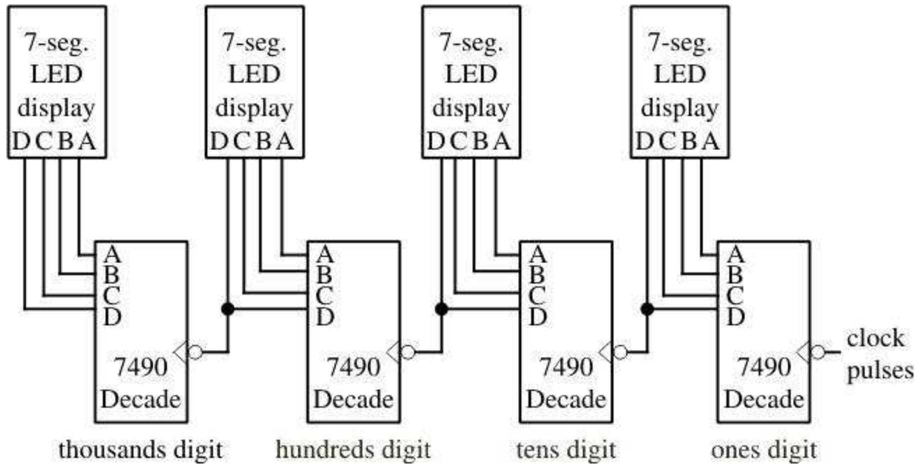
Input pulses	b_3	b_2	b_1	b_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0

Reset



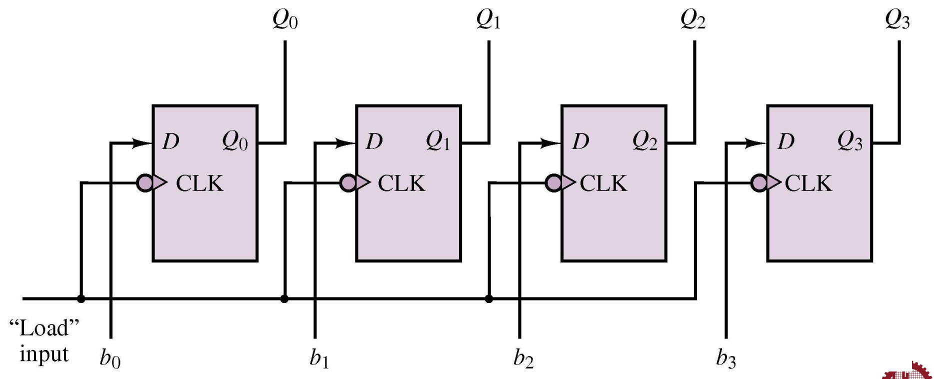
A decade counter counts from 0 to 9 and then resets.

Cascaded Decade Counter

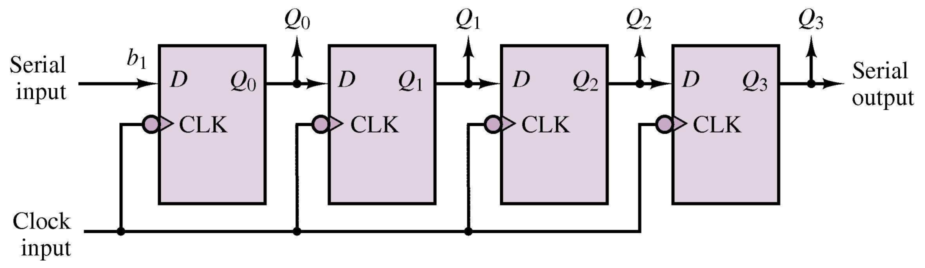


4-bit Parallel Register

In the register, the *load input pulse acts on all clocks simultaneously causing the parallel inputs to be transferred to the respective flip-flops to store the binary data.*



4-bit Shift Register



Same basic structure of parallel register applies to the shift register, except that the input is now applied to the first flip-flop and shifted along at each clock pulse. Note that, this type of register provides both a serial and a parallel output.