18CS 33

# ANALOG AND DIGITAL ELECTRONICS

## MODULE 4

# VHDL, LATCHES AND FLIP-FLOPS

Mahesh Prasanna K.
Dept. of CSE, VCET.

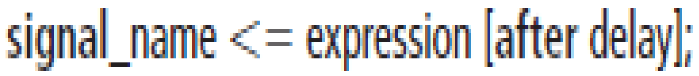
## INTRODUCTION TO VHDL

* ***VHDL*** stands for ***VHSIC-HDL*** (*Very High Speed Integrated Circuit-Hardware Description Language*).
* ***VHDL*** is a hardware description language that is used to describe the behavior and structure of digital systems.
* ***VHDL*** is a general-purpose hardware description language which can be used to describe and simulate the operation of a wide variety of digital systems, ranging in complexity from a few gates to an interconnection of many complex integrated circuits.
* ***VHDL*** was originally developed to allow a uniform method for specifying digital systems. The VHDL language became an IEEE standard in 1987, and it is widely used in industry. IEEE published a revised VHDL standard in 1993.

* ***VHDL*** can describe a digital system at several different levels—*behavioral*, *data flow*, and *structural*. For example,
  * A binary adder could be described at the *behavioral level* in terms of its function of adding two binary numbers, without giving any implementation details.
  * The same adder could be described at the *data flow level* by giving the logic equations for the adder.
  * Finally, the adder could be described at the *structural level* by specifying the interconnections of the gates which make up the adder.

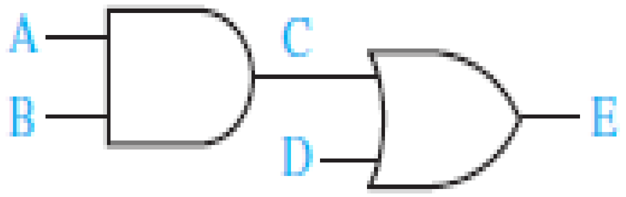
# VHDL DESCRIPTION OF COMBINATIONAL CIRCUITS

* In VHDL, a signal assignment statement has the form:
  * *C <= A* ***and*** *B* ***after*** *5 ns;*


signal_name <= expression [after delay];

  * A VHDL signal (are *concurrent*) is used to describe a signal in a physical system .
  * Square brackets indicate that after delay is *optional*.
  * If after delay is omitted, then the signal is scheduled to be updated after a *delta delay, Δ (infinitesimal delay)*.
  * VHDL is *not case sensitive*.
  * VHDL *identifiers* may contain *letters, numbers, and the underscore character* .
  * An *identifier must start with a letter*, and it cannot end with an underscore.
  * VHDL statement can be continued over several lines.
  * anything following a *double dash* (**--**) is treated as a comment.
  * Words such as *and*, *or*, and *after* are reserved words (or *keywords*).

* The gate circuit of the following Figure has five signals: *A, B, C, D, and E*. The symbol " **<=** " is the *signal assignment operator* which indicates that the value computed on the right-hand side is assigned to the signal on the left side.



* Dataflow Description

*C <= A* ***and*** *B* ***after*** *5 ns;*

*E <= C* ***or*** *D* ***after*** *5 ns;*

* Behavioral Description

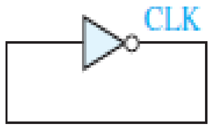
*E <= D* ***or*** *(A* ***and*** *B);*

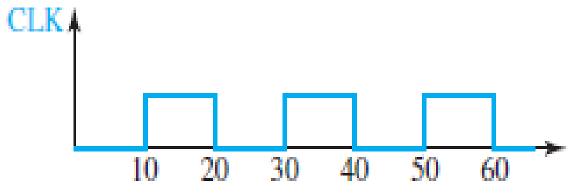
* Structural Description

*Gate1: AND2* ***port map*** *(A, B, D);*

*Gate2: OR2* ***port map*** *(C, D, E);*

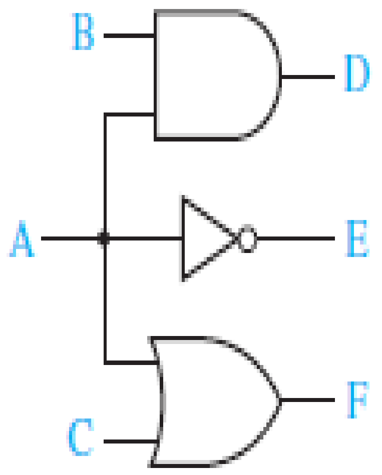
***Inverter*** with output connected back to the input



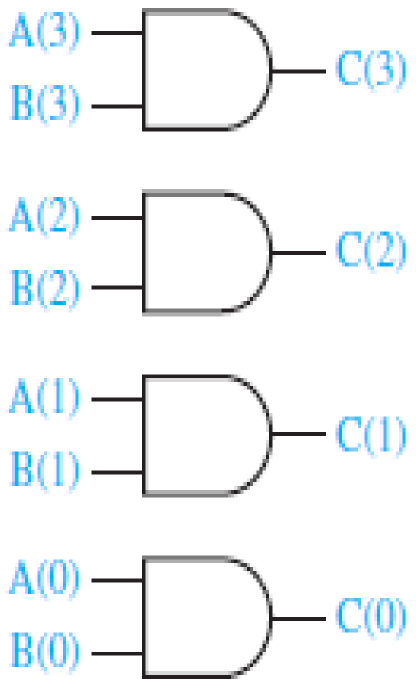


```
CLK <= not CLK after 10 ns;
```

## * Bit & Vector



```
-- when A changes, these concurrent
-- statements all execute at the same time
D <= A and B after 2 ns;
E <= not A after 1 ns;
F <= A or C after 3 ns;
```



```
-- the hard way
C(3) <= A(3) and B(3);
C(2) <= A(2) and B(2);
C(1) <= A(1) and B(1);
C(0) <= A(0) and B(0);
```

```
-- the easy way
C <= A and B;
```

## * Inertial Delay Model

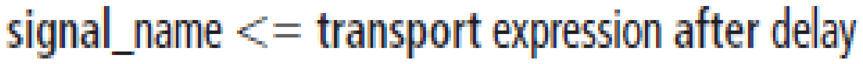
A device with an inertial delay of *D* time units filters out output changes that would occur in less than or equal to D time units.

* Signal assignment statements containing "*after delay*" create what is called an *inertial delay model*.
  * Consider a device with an inertial delay of *D* time units.
  * If an input change to the device will cause its output to change, then the output changes *D* time units later.
  * But, if the device receives two input changes within a period of *D* time units; the device output does not change in response to either input change.
  * *Example:* Consider the signal assignment *C <= A **and** B **after** 10 ns;*

Assume A and B are initially 1, and A changes to 0 at 15 ns, to 1 at 30 ns, and to 0 at 35 ns. Then C changes to 1 at 10 ns and to 0 at 25 ns, but C does not change in response to the A changes at 30 ns and 35 ns.

## * Ideal (Transport) Delay Model

- * Output changes caused by input changes to a device exhibiting an ideal (transport) delay of *D* time units are delayed by *D* time units, and the output changes occur even if they occur within *D* time units.

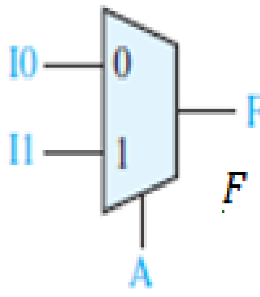
- * The VHDL signal assignment statement that models ideal (transport) delay is 

```
signal_name <= transport expression after delay
```

- * *Example:* consider the signal assignment *C <=* ***transport*** *A* ***and*** *B* ***after*** *10 ns;*

Assume A and B are initially 1 and A changes to 0 at 15 ns, to 1 at 30 ns, and to 0 at 35 ns. Then C changes to 1 at 10 ns, to 0 at 25 ns, to 1 at 40 ns, and to 0 at 45 ns. Note that the last two changes are separated by just 5 ns.

# VHDL MODELS FOR MULTIPLEXERS



$F = A'I_0 + AI_1$

The corresponding VHDL statement is

F <= (**not** A **and** I0) **or** (A **and** I1);

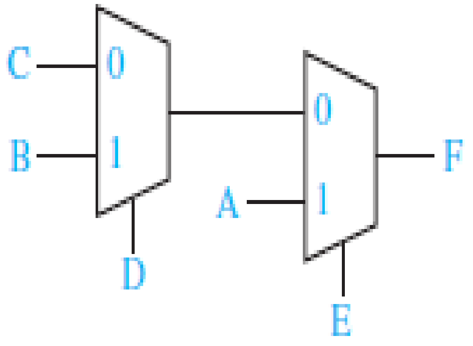
MUX by a conditional signal assignment statement.

F <= I0 **when** A = '0' **else** I1;

The general form of a conditional signal assignment statement is

```
signal_name <= expression1 when condition1
               else expression2 when condition2
               [else expressionN];
```

Two Cascaded MUXes



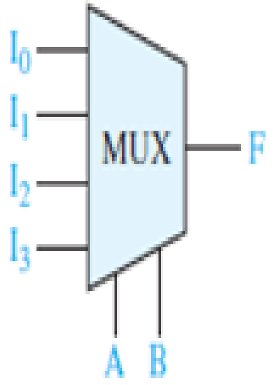
```
F <= A when E = '1'
     else B when D = '1'
     else C;
```

## * 4-to-1 MUX

$$F = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3.$$

*F <= (not A and not B and I0) or (not A and B and I1) or*

*(A and not B and I2) or (A and B and I3);*



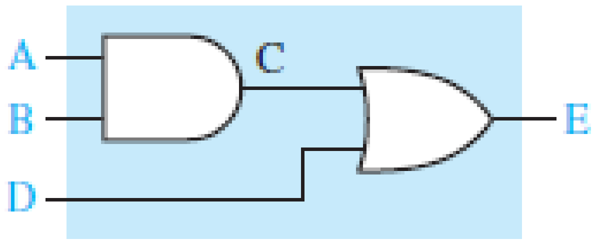
```
sel <= A&B;
-- selected signal assignment statement
with sel select
  F <= I0 when "00",
       I1 when "01",
       I2 when "10",
       I3 when "11";
```

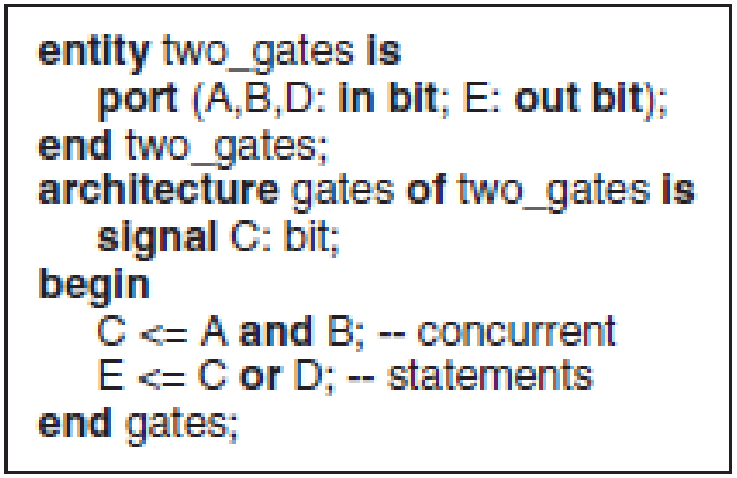
```
F <= I0 when A = '0' and B = '0'
  else I1 when A = '0' and B = '1'
  else I2 when A = '1' and B = '0'
  else I3;
```

The general form of a selected signal assignment statement is

```
with expression_s select
   signal_s <= expression1 [after delay-time] when choice1,
               expression2 [after delay-time] when choice2,
               . . .
               [expression_n [after delay-time] when others];
```

# VHDL MODULES



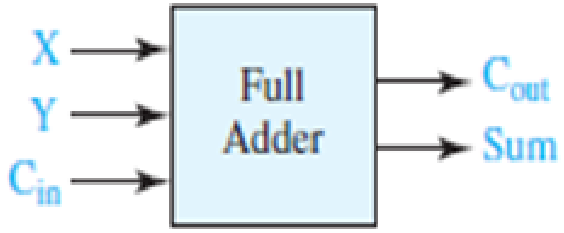


```
entity two_gates is
    port (A,B,D: in bit; E: out bit);
end two_gates;
architecture gates of two_gates is
    signal C: bit;
begin
    C <= A and B; -- concurrent
    E <= C or D; -- statements
end gates;
```

- When we describe a system in VHDL, we must specify an *entity* and *architecture* at the top level.
- The *entity* declaration gives the name "***two_gates***" to the module.
- The *port* declaration specifies the inputs and outputs to the module. *A*, *B*, and *D* are input signals of type bit, and *E* is an output signal of type bit.
- The *architecture* is named "***gates***".
- The signal *C* is declared within the architecture because it is an internal signal. The two concurrent statements that describe the gates are placed between the keywords *begin* and *end*.

***Example:*** *To write the entity and architecture for a full adder module.*

*The entity specifies the inputs and outputs of the adder module, as shown in the following Figure. The port declaration specifies that X, Y and Cin are input signals of type bit, and that Cout and Sum are output signals of type bit.*



```
entity FullAdder is
    port (X,Y,Cin: in bit;          -- Inputs
              Cout, Sum: out bit);  -- Outputs
end FullAdder;
```

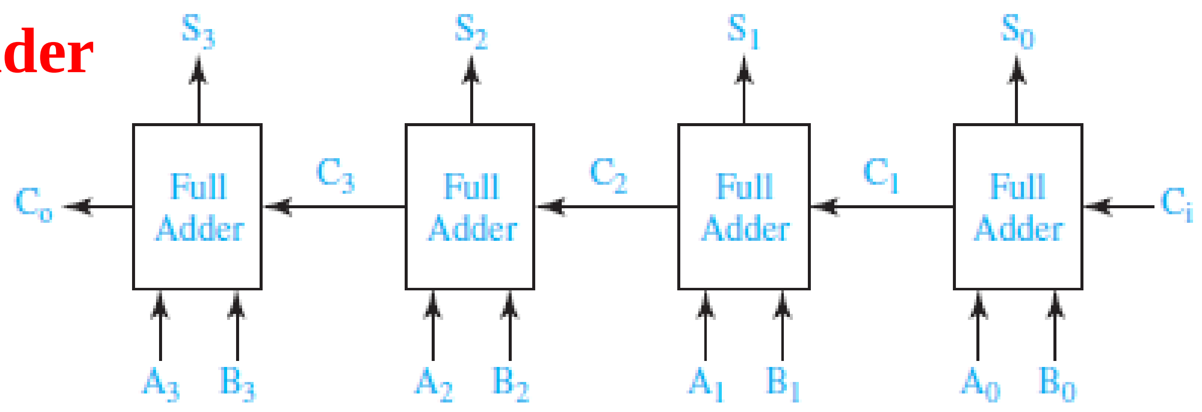
*The operation of the full adder is specified by an architecture declaration:*

```
architecture Equations of FullAdder is
begin           -- concurrent assignment statements
    Sum <= X xor Y xor Cin after 10 ns;
    Cout <= (X and Y) or (X and Cin) or (Y and Cin) after 10 ns;
end Equations;
```

*In this example, the architecture name (Equations) is arbitrary, but the entity name (FullAdder) must match the name used in the associated entity declaration.*

*The VHDL assignment statements for Sum and Cout represent the logic equations for the full adder. Several other architectural descriptions such as a truth table or an interconnection of gates could have been used instead. In the Cout equation, parentheses are required around (X and Y) because VHDL does not specify an order of precedence for the logic operators.*

## * Four-Bit Full Adder



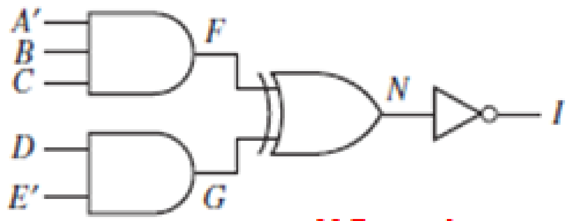
```
entity Adder4 is
     port (A, B: in bit_vector(3 downto 0); Ci: in bit;    -- Inputs
          S: out bit_vector(3 downto 0); Co: out bit);    -- Outputs
end Adder4;
architecture Structure of Adder4 is
component FullAdder
     port (X, Y, Cin: in bit;     -- Inputs
          Cout, Sum: out bit);     -- Outputs
end component;
signal C: bit_vector(3 downto 1);
begin     -- instantiate four copies of the FullAdder
     FA0: FullAdder port map (A(0), B(0), Ci, C(1), S(0));
     FA1: FullAdder port map (A(1), B(1), C(1), C(2), S(1));
     FA2: FullAdder port map (A(2), B(2), C(2), C(3), S(2));
     FA3: FullAdder port map (A(3), B(3), C(3), Co, S(3));
end Structure;
```

***Homework:***

*1] Write VHDL statements that represent the following circuit:*

*a) Write a statement for each gate.*

*b) Write one statement for the whole circuit.*



*2] Draw the circuit represented by the following VHDL statements:*

*F <= E* ***and*** *I;*

*I <= G* ***or*** *H;*

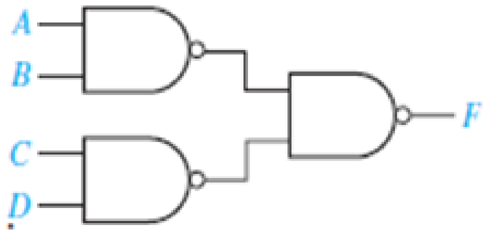
*G <= A* ***and*** *B;*

*H <=* ***not*** *C* ***and*** *D;*

*3] Write*

*a) a complete VHDL module for a two-input NAND gate with 4-ns delay.*

*b) Write a complete VHDL module for the following circuit that uses the NAND gate module of Part (a) as a component.*

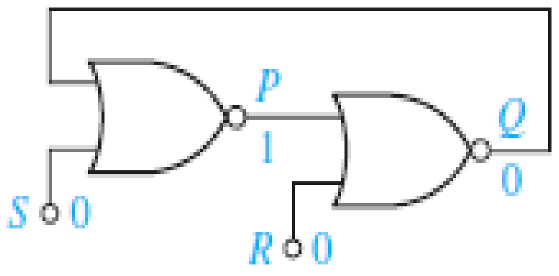


## LATCHES & FLIP-FLOPS

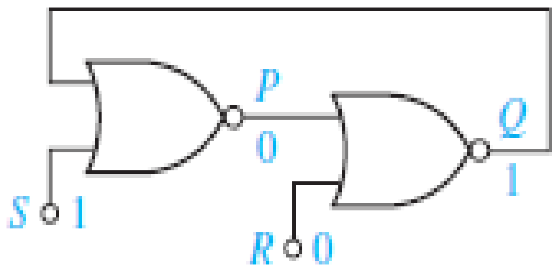
- Sequential switching circuits have the property that the *output depends not only on the present input but also on the past sequence of inputs*.
- In effect, these circuits must be able to "*remember*" something about the past history of the inputs in order to produce the present output.
- Latches and flip-flops are commonly used *memory devices* in sequential circuits.
- Basically, latches and flip-flops are memory devices which
  - *can assume one of two stable output states* and
  - *have one or more inputs that can cause the output state to change.*

## SET RESET LATCH

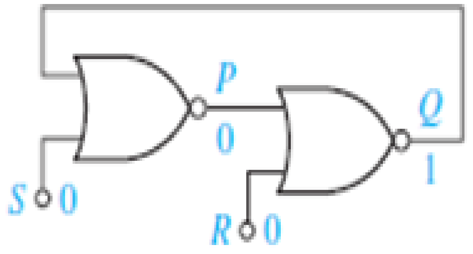
* A simple latch can be constructed by introducing feedback into a NOR-gate circuit, as given in the following Figure (a).
* As indicated, if the inputs are S = R = 0, the circuit can assume a stable state with Q = 0 and P = 1.



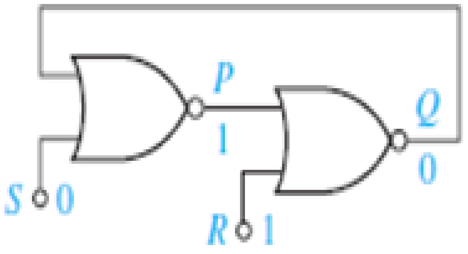
(a)



(b)

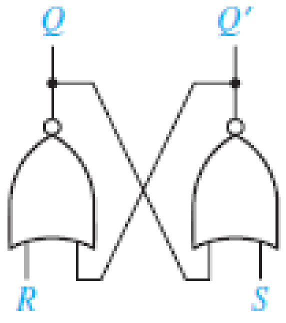


(c)

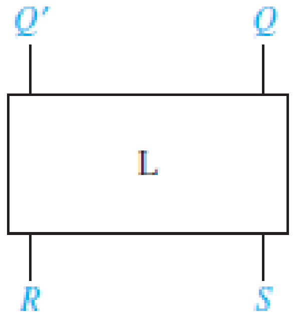


(d)

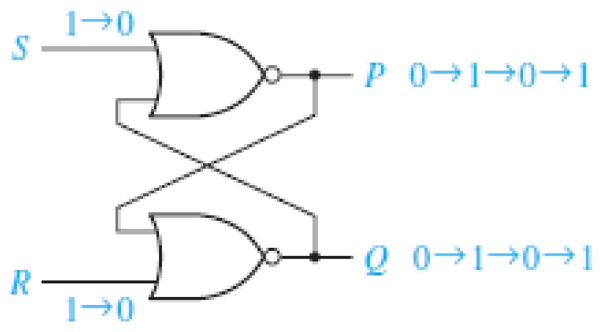
* The circuit is often drawn in cross-coupled form, as shown in Figure (a).



(a)



(b)

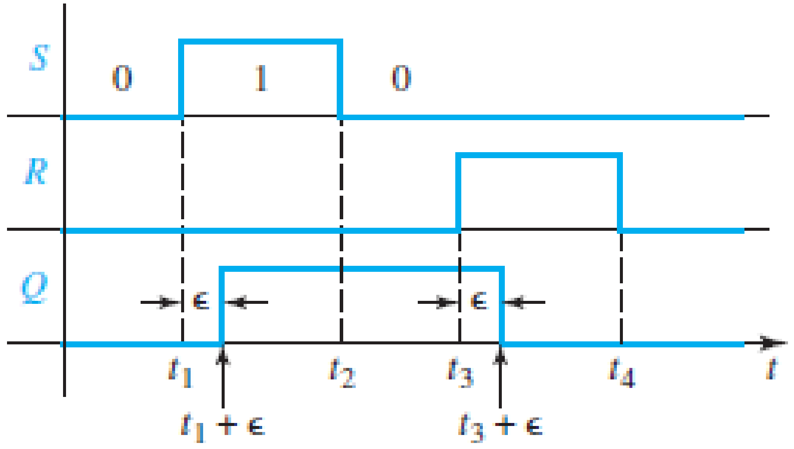


(c)

$$Q(t+\epsilon) = R(t)'[S(t) + Q(t)] = R(t)'S(t) + R(t)'Q(t) \quad \text{or} \quad Q^+ = R'S + R'Q$$

$$P(t) = S(t)'Q(t)' \quad \text{or}$$

$$P = S'Q'$$



| RQ \ S | 0 | 1 |
|---|---|---|
| 00 | 0 | 1 |
| 01 | 1 | 1 |
| 11 | 0 | X |
| 10 | 0 | X |

(a) $Q^+$ map

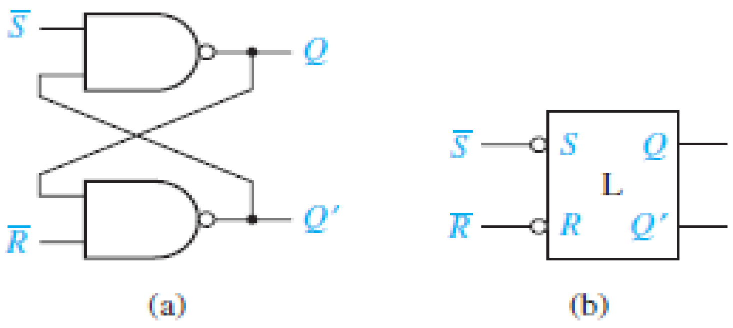
| S | R | Q | $Q^+$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | – } Inputs not allowed |
| 1 | 1 | 1 | – |

(b) Truth table

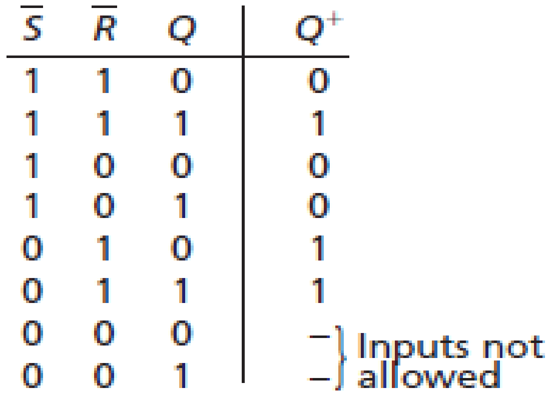
$$Q^+ = S + R'Q$$

| Present State Q | Next State $Q^+$ SR 00 | SR 01 | SR 11 | SR 10 | Present Output P SR 00 | SR 01 | SR 11 | SR 10 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

* An alternative form of the S-R latch uses NAND gates, as shown in the following Figure.



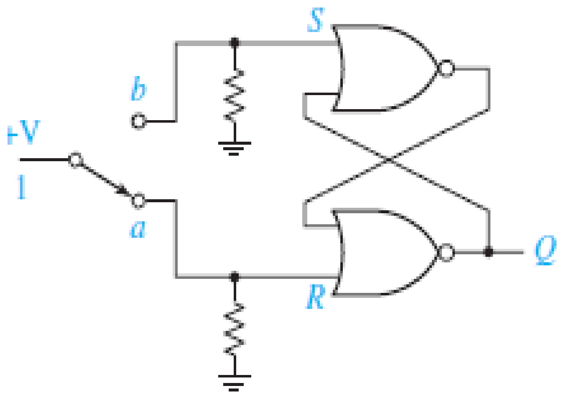
$\bar{S}$-$\bar{R}$ Latch

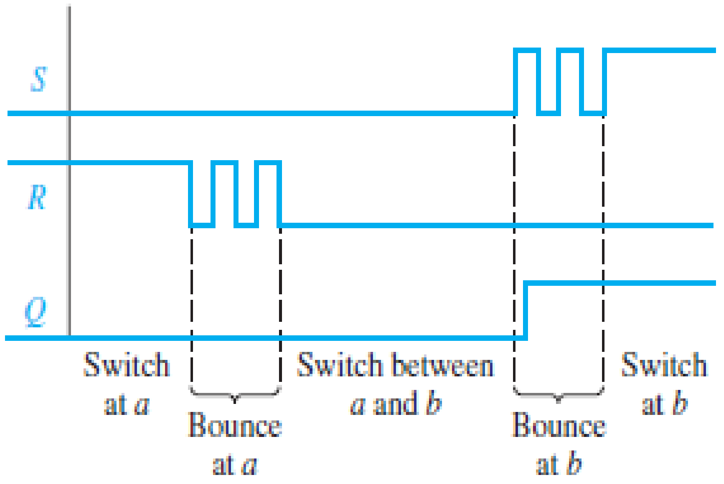


| $\bar{S}$ | $\bar{R}$ | $Q$ | $Q^+$ |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | – } Inputs not allowed |
| 0 | 0 | 1 | – |

(c)

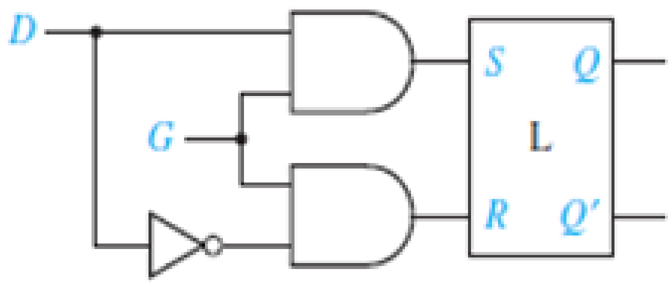
* Applications of S-R Latch:



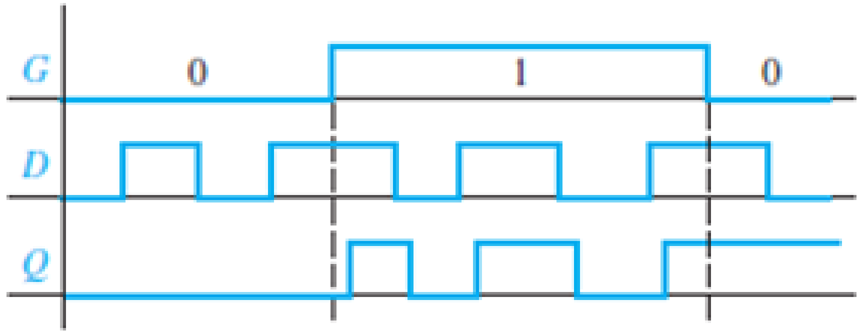


# GATED D LATCH

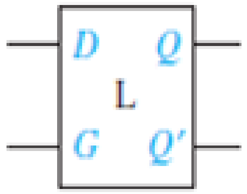
* A gated *D* latch has two inputs—a data input (*D*) and a gate input (*G*).
  * When *G = 0*, *S = R = 0*, so *Q* does not change.
  * When *G = 1*
    * *D = 1*, *S = 1* and *R = 0*, so *Q* is set to *1*.
    * *D = 0*, *S = 0* and *R = 1*, so *Q* is reset to *0*.

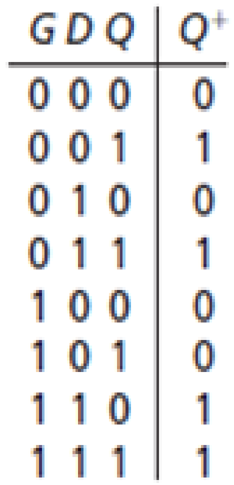


(a)

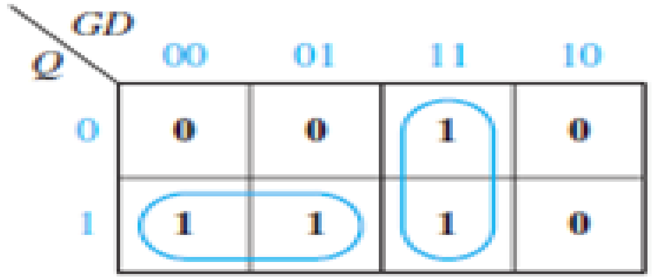


(b)





| G | D | Q | $Q^+$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

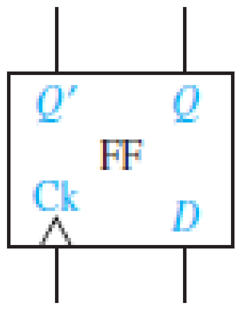


| Q \ GD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

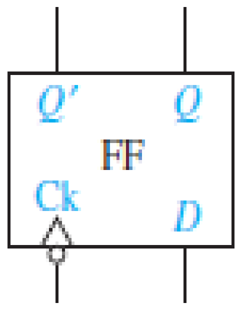
$Q^+ = G'Q + GD$

## EDGE TRIGGERED D FLIP-FLOP

- A *D* flip-flop has two inputs, *D* (data) and *Clk* (clock).
- Unlike the *D* latch, the flip-flop output changes only in response to the clock, not to a change in *D*.
  - If the output can change in response to
    - a *0* to *1* transition on the clock input, we say that the flip-flop is triggered on the *rising edge* (or *positive edge*) of the clock.
    - a 1 to *0* transition on the clock input, we say that the flip-flop is triggered on the *falling edge* (or *negative edge*) of the clock.
  - An *inversion bubble* on the clock input indicates a *falling-edge trigger* (Figure (b)), and no bubble indicates a rising-edge trigger (Figure (a)).
  - The term *active edge* refers to the clock edge (rising or falling) that triggers the flip-flop state change.



(a) Rising-edge trigger



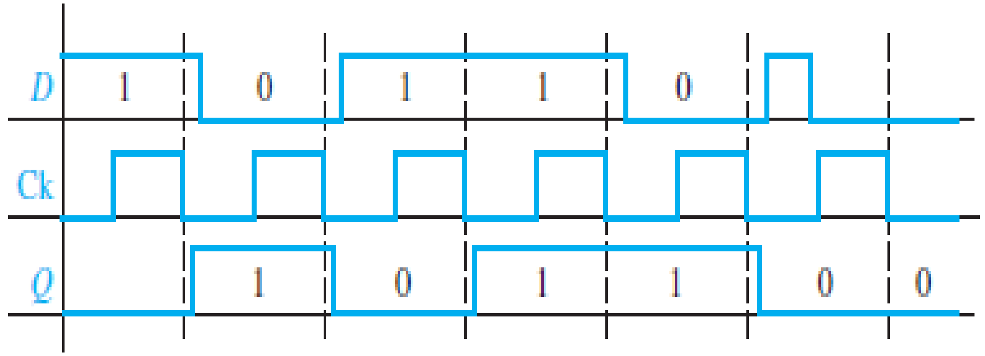
(b) Falling-edge trigger

| D | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

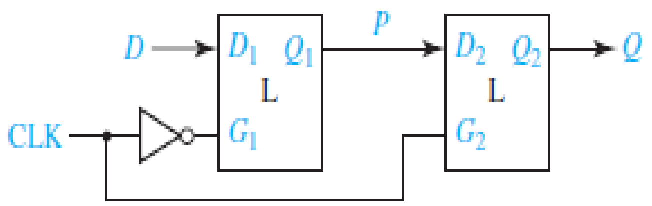
(c) Truth table

$$\boxed{Q^+ = D}$$

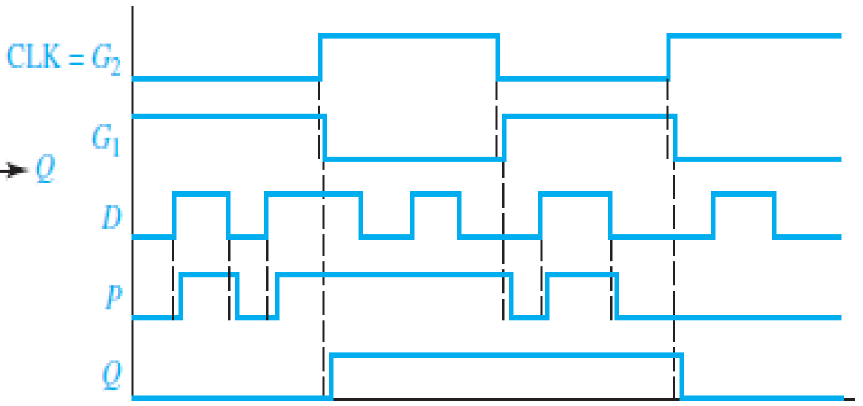
Timing for D Flip-Flop (Falling-Edge Trigger)



D Flip-Flop (Rising-Edge Trigger)



(a) Construction from two gated D latches

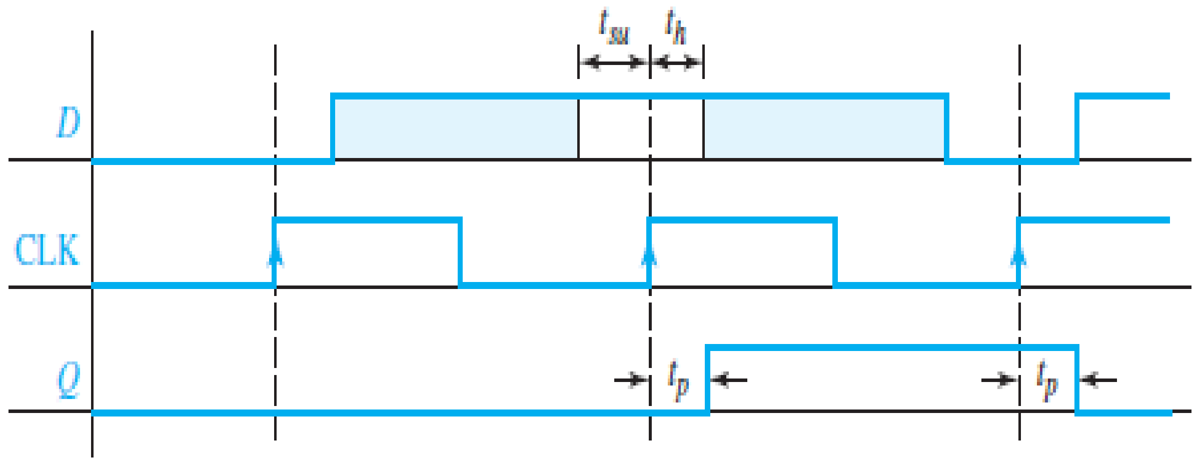


(b) Timing analysis

* A flip-flop changes state only on the active edge of the clock; the ***propagation delay*** (***tp***) of a flip-flop is the time between the active edge of the clock and the resulting change in the output.
* There are also timing issues associated with the D input.
    * To function properly, the *D* input to an edge-triggered flip-flop must be held at a constant value for a period of time before and after the active edge of the clock.
    * If D changes at the same time as the active edge, the behavior is unpredictable.

* The amount of time that the *D* input must be stable before the active edge is called the ***setup time*** (***tsu***),
* The amount of time that the *D* input must hold the same value after the active edge is the ***hold time*** (***th***).

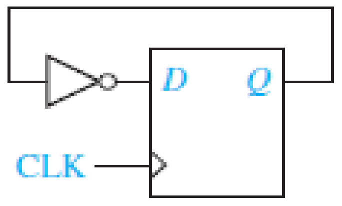
Setup and Hold Times for an Edge-Triggered D Flip-Flop



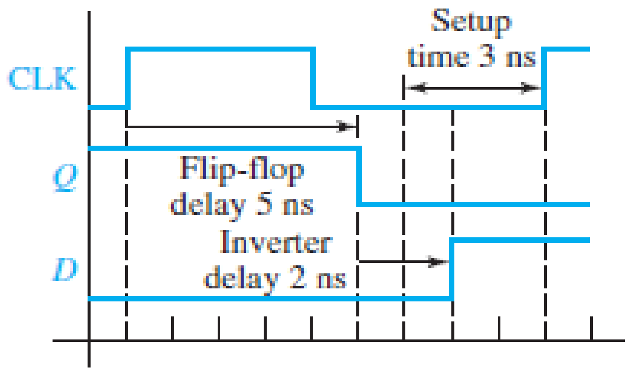
* Using these timing parameters, we can determine the minimum clock period for a circuit which will not violate the timing constraints.

* Figure (a): Inverter has a propagation delay of 2 ns, and the flip-flop has a propagation delay of 5 ns and a setup time of 3 ns.

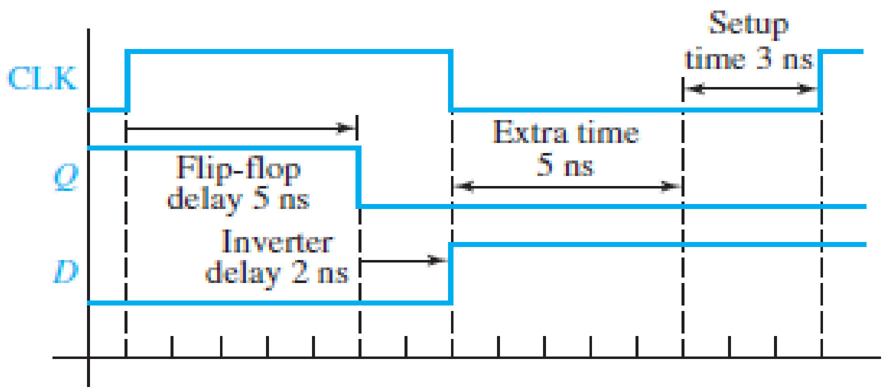
Determination of Minimum Clock Period



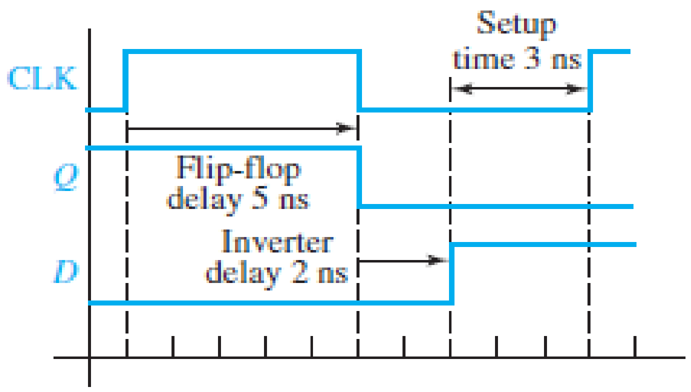
(a) Simple flip-flop circuit



(b) Setup time not satisfied



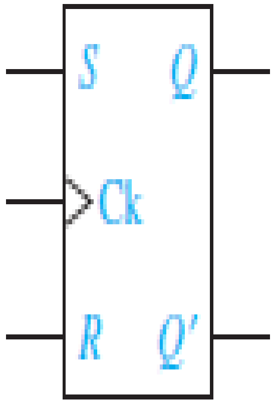
(c) Setup time satisfied



(d) Minimum clock period

## SR FLIP-FLOP

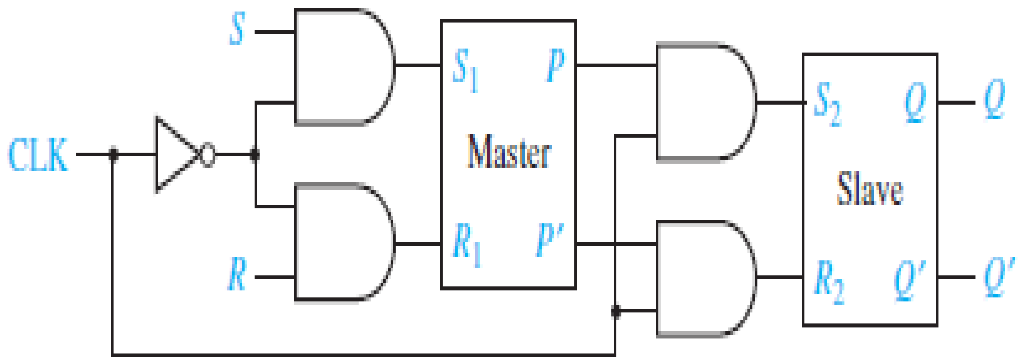
- An S-R flip-flop is similar to an S-R latch in that S = 1 sets the Q output to 1, and R = 1 resets the Q output to 0.
- The difference is that the flip-flop has a clock input; and the Q output can change only after an active clock edge.



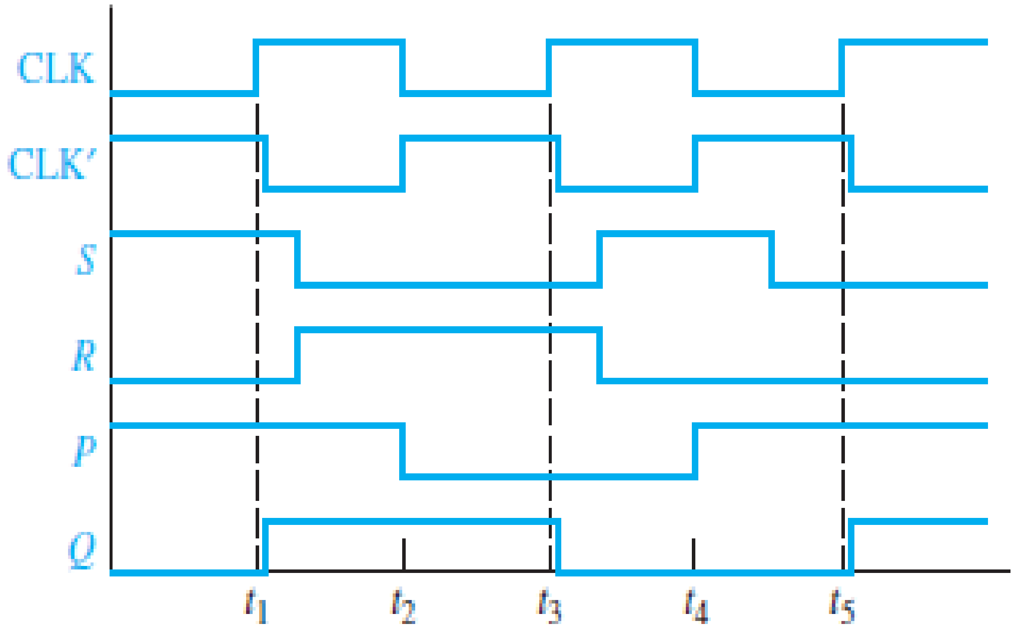
Operation summary:

| | |
|---|---|
| $S = R = 0$ | No state change |
| $S = 1, R = 0$ | Set $Q$ to 1 (after active Ck edge) |
| $S = 0, R = 1$ | Reset $Q$ to 0 (after active Ck edge) |
| $S = R = 1$ | Not allowed |

* The following Figure (a) shows an S-R flip-flop constructed from two S-R latches and gates. This flip-flop changes state after the rising edge of the clock. The circuit is often referred to as a ***master-slave flip-flop***.

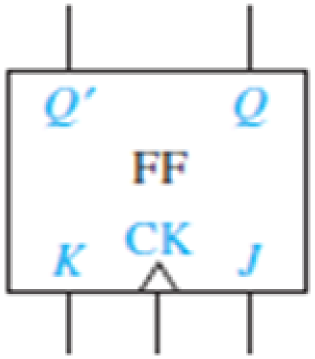


(a) Implementation with two latches



(b) Timing analysis

## JK FLIP-FLOP

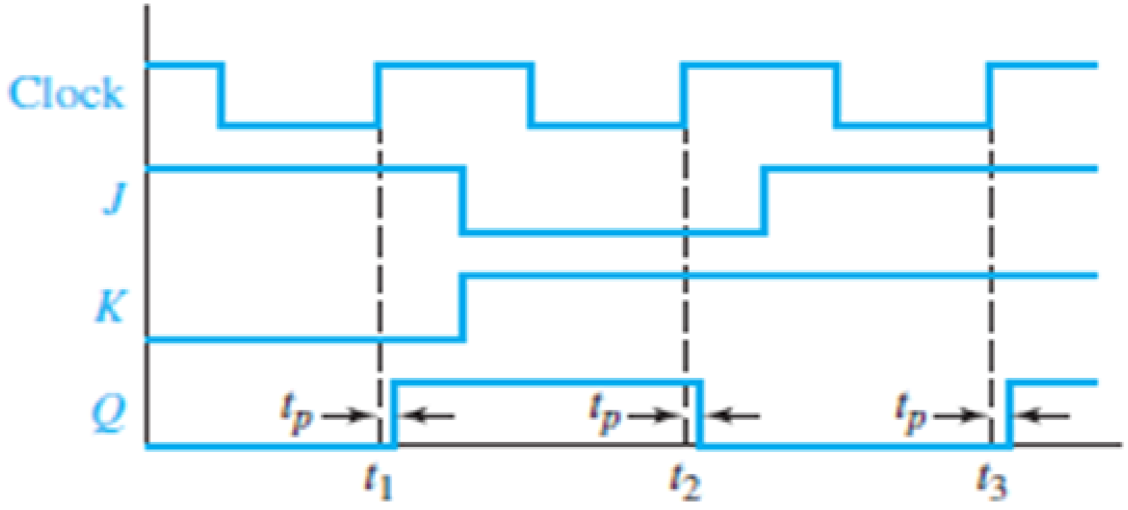


(a) J-K flip-flop

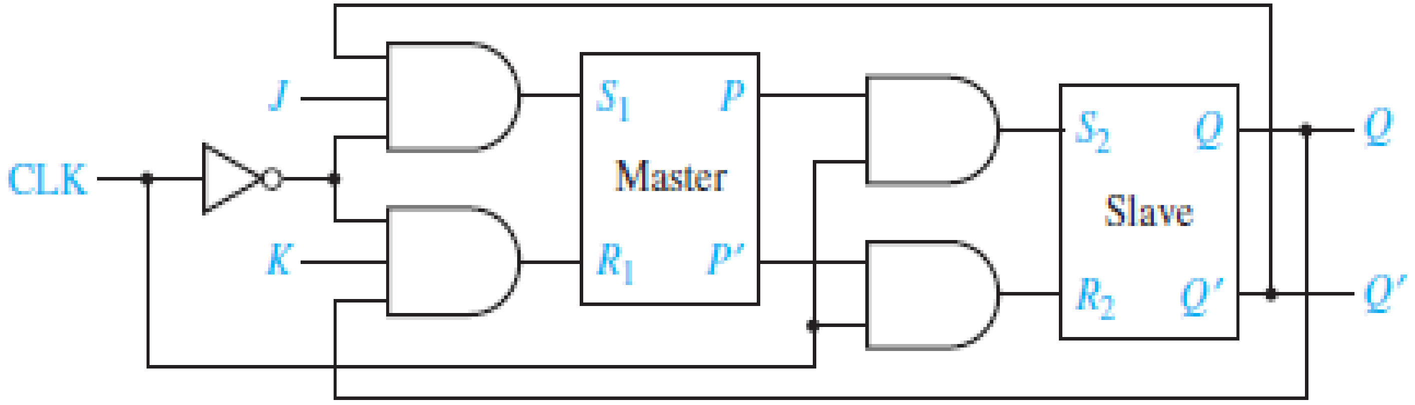
$$\boxed{Q^+ = JQ' + K'Q}$$

| J | K | Q | $Q^+$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

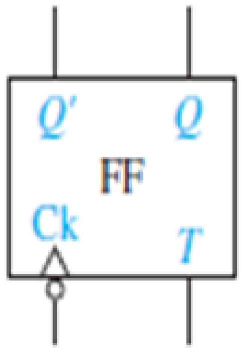
(b) Truth table



(c) J-K flip-flop timing



## T FLIP-FLOP

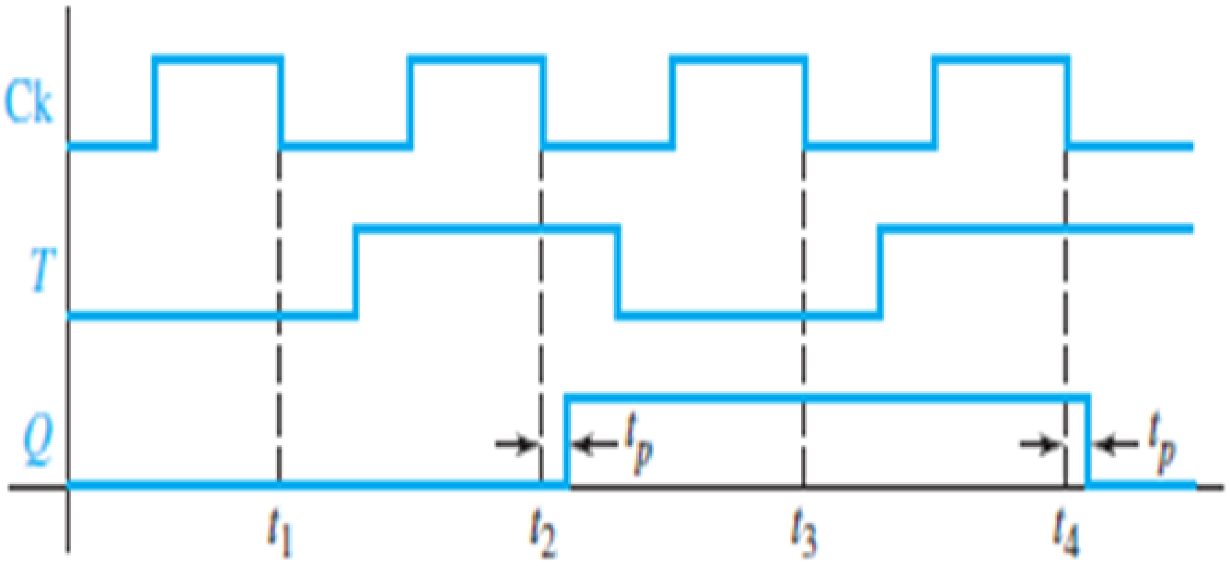


(a)

| T Q | $Q^+$ |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

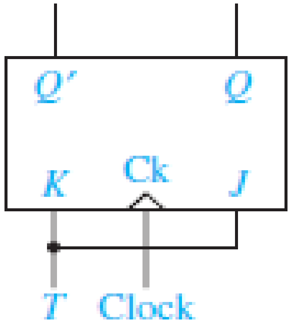
(b)

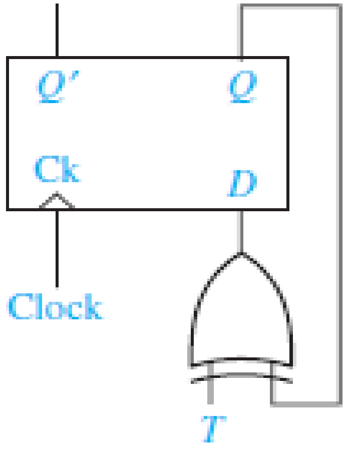
$$\boxed{Q^+ = T'Q + TQ' = T \oplus Q}$$



Timing Diagram for T Flip-Flop (Falling-Edge Trigger)

Implementation of T Flip-Flops



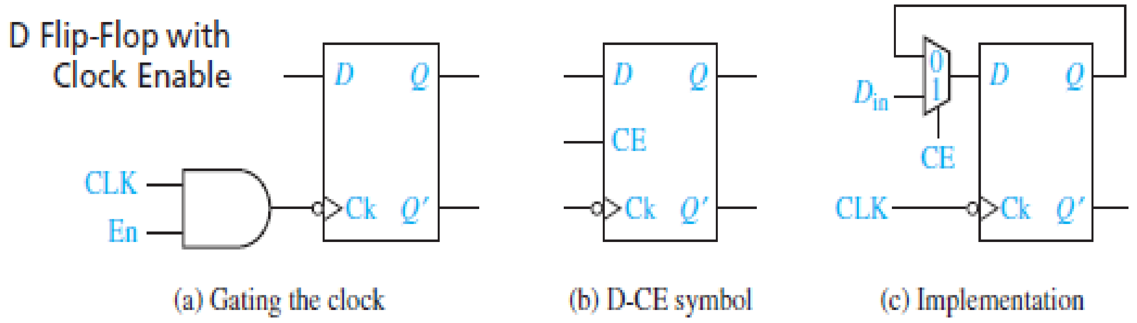


$$Q^+ = JQ' + K'Q = TQ' + T'Q$$

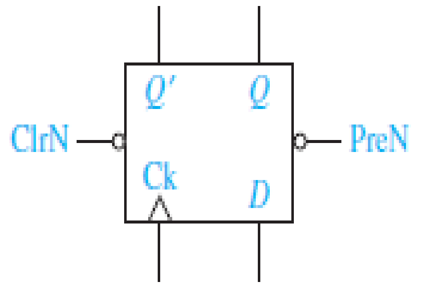
$$Q^+ = Q \oplus TQ' + T'Q$$

(a) Conversion of J-K to $T$ (b) Conversion of $D$ to $T$

# FLIP-FLOP WITH ADDITIONAL INPUTS



(a) Gating the clock (b) D-CE symbol (c) Implementation



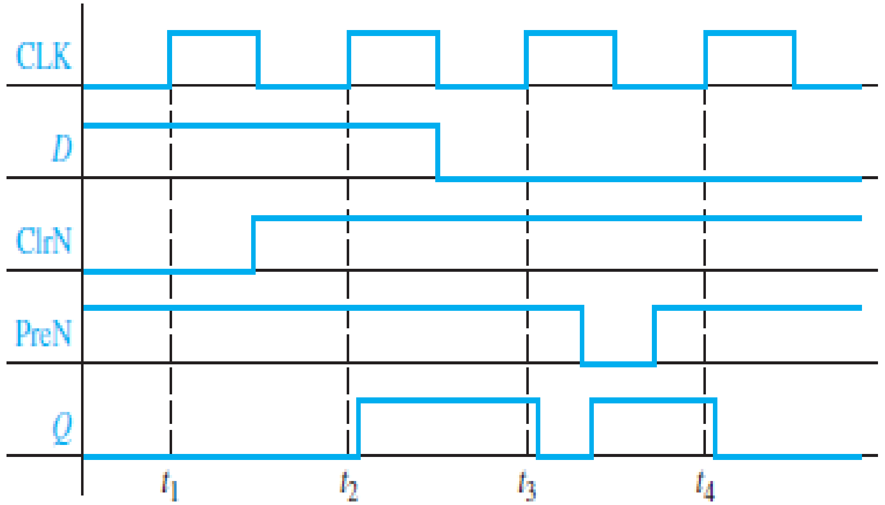
(a)

| Ck | D | PreN | ClrN | $Q^+$ |
|---|---|---|---|---|
| x | x | 0 | 0 | (not allowed) |
| x | x | 0 | 1 | 1 |
| x | x | 1 | 0 | 0 |
| ↑ | 0 | 1 | 1 | 0 |
| ↑ | 1 | 1 | 1 | 1 |
| 0,1,↓ | x | 1 | 1 | Q (no change) |

(b)

* Flip-flops often have additional inputs which can be used to set the flip-flops to an initial state independent of the clock.

Timing Diagram for D Flip-Flop with Asynchronous Clear and Preset



## * Characteristic Equations of Flip-Flops

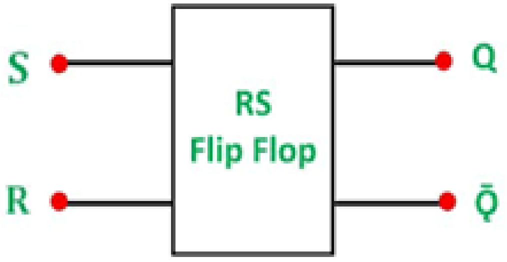
| S | R | $Q^+$ |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | ? |

| D | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| J | K | $Q^+$ |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\bar{Q}$ |

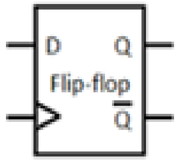
| T | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

***SR Flip-Flop:***



$Q^+ = S + R'Q\ (SR = 0)$

***D Flip-Flop:***



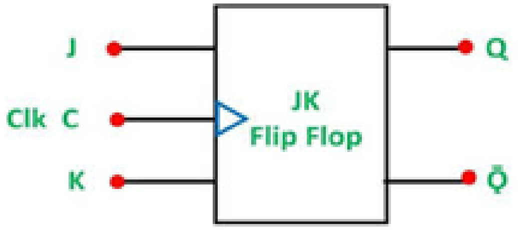
$Q^+ = D$

$Q^+ = S + R'Q\ (SR = 0)$ (S-R latch or flip-flop)

$Q^+ = GD + G'Q$ (gated D latch)

$Q^+ = D$ (D flip-flop)

***JK Flip-Flop:***



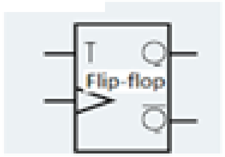
$Q^+ = JQ' + K'Q$

$Q^+ = D{\cdot}CE + Q{\cdot}CE'$ (D-CE flip-flop)

$Q^+ = JQ' + K'Q$ (J-K flip-flop)

$Q^+ = T \oplus Q = TQ' + T'Q$ (T flip-flop)

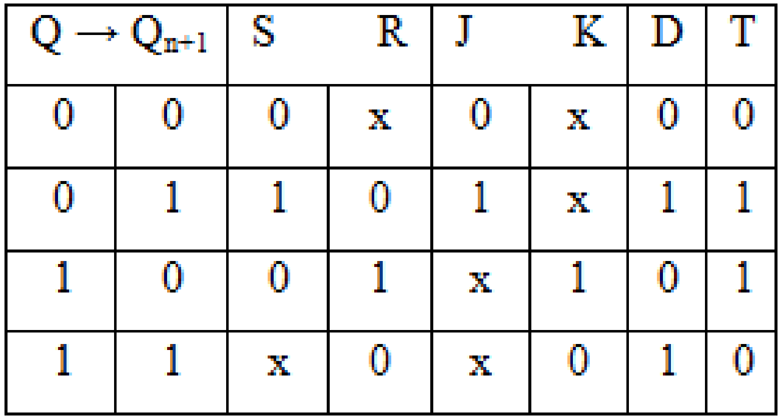
***T Flip-Flop:***



$Q^+ = T \oplus Q = TQ' + T'Q$

## * Flip-Flop as Finite State Machine

* State Transition Diagrams of SR, D, JK & T Flip-Flops

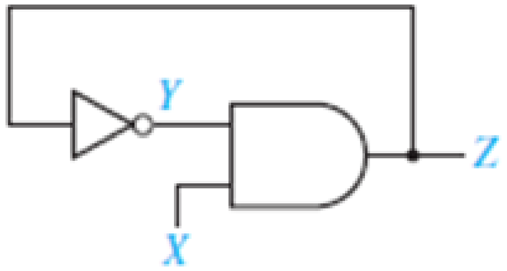


| $Q \rightarrow Q_{n+1}$ | | S | R | J | K | D | T |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | x | 0 | x | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | x | 1 | 1 |
| 1 | 0 | 0 | 1 | x | 1 | 0 | 1 |
| 1 | 1 | x | 0 | x | 0 | 1 | 0 |

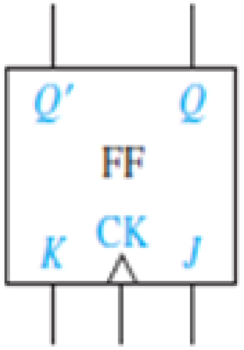
Excitation Table of Flip-Flops

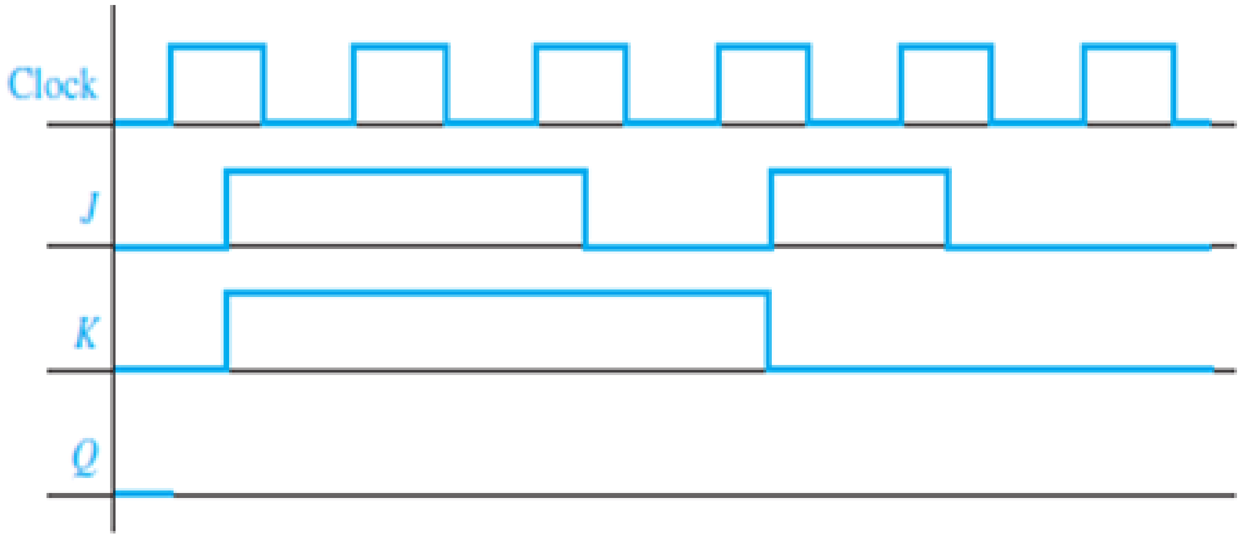
***Homework:***

*1] Assume that the inverter in the given circuit has a propagation delay of 5 ns and the AND gate has a propagation delay of 10 ns. Draw a timing diagram for the circuit showing X, Y, and Z. Assume that X is initially 0, Y is initially 1, after 10 ns X becomes 1 for 80 ns, and then X is 0 again.*

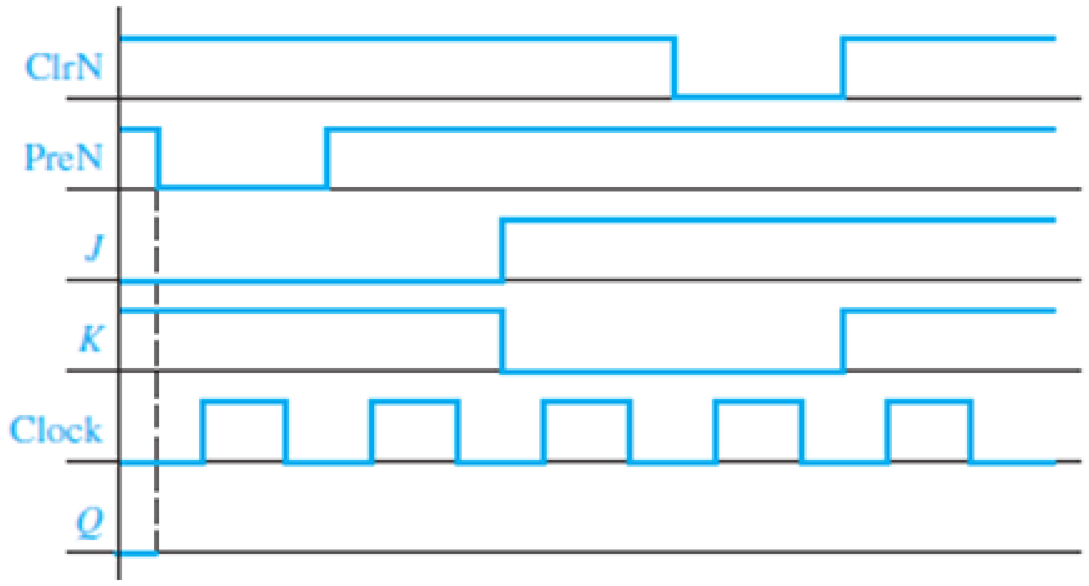


*2] Complete the following timing diagram for the flip-flop:*

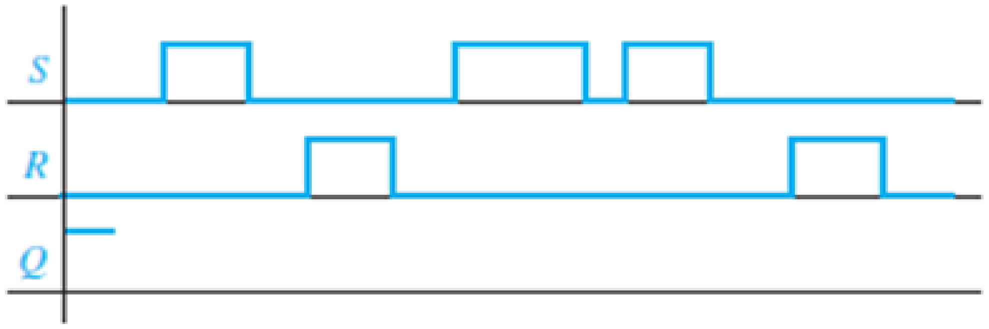




3] Complete the following timing diagram for a J-K flip-flop with a falling-edge trigger and asynchronous ClrN and PreN inputs.



4] Complete the following timing diagram for an S-R latch. Assume Q begins at 1.



5] Convert by adding external gates: (a) a D flip-flop to a J-K flip-flop; (b) a T flip-flop to a D flip-flop; (c) a T flip-flop to a D flip-flop with clock enable.