Flip-Flops

VOLKAN KURSUN

Some material from McGraw Hill

Outline

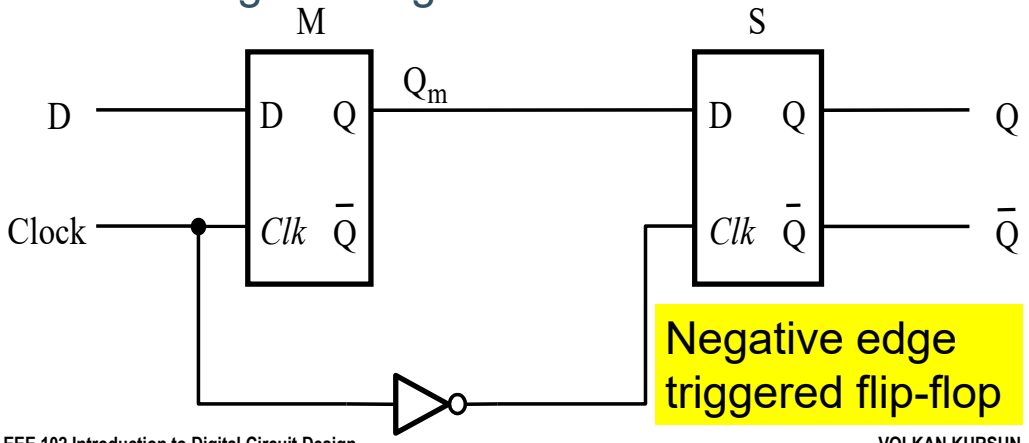
- Edge-Triggered D Flip-Flop
- T Flip-Flop
- JK Flip-Flop

Edge-Sensitive Data Storage

- **Level-sensitive** behavior: the state of the latch changes according to the values of the input signal during the clock period when the latch is transparent
- The **output state may change multiple times** during the phase of the clock signal when a level sensitive latch is transparent
- **Edge-sensitive** behavior: the state of a storage element cannot change more than once in a clock cycle
- **Positive edge triggered** flip-flop: the data is sampled and the output state may change only with the positive edges of the clock
- **Negative edge triggered** flip-flop: the data is sampled and the output state may change only with the negative edges of the clock

Master-Slave D Flip-Flop

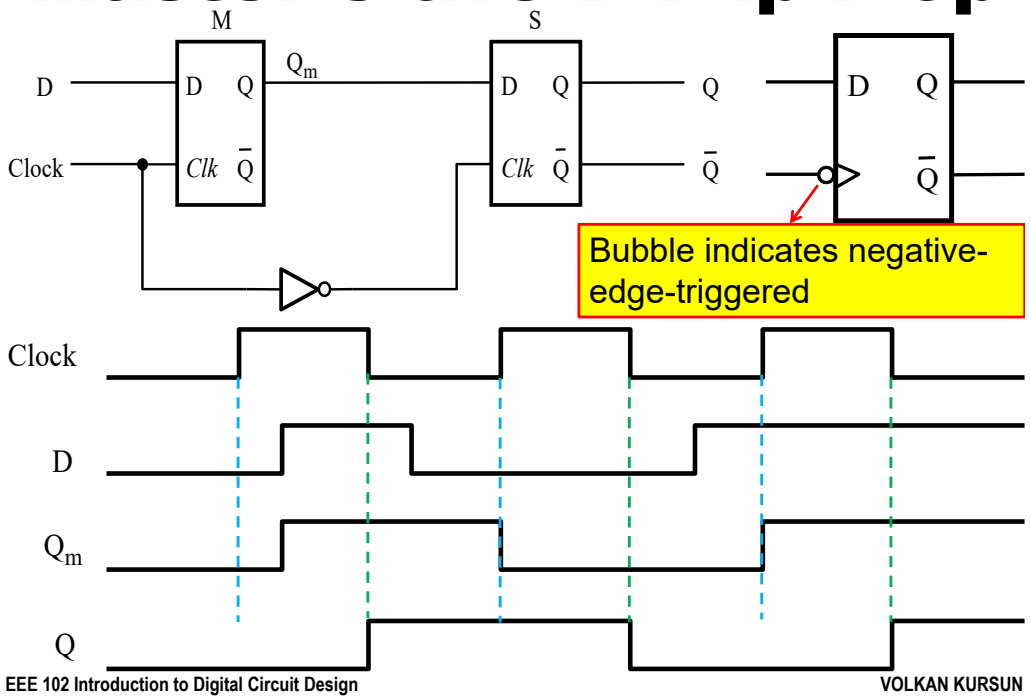
- Consists of two gated (level sensitive) D latches: master and slave
- Master stage: changes its state while Clock = 1
- Slave stage: changes its state while Clock = 0



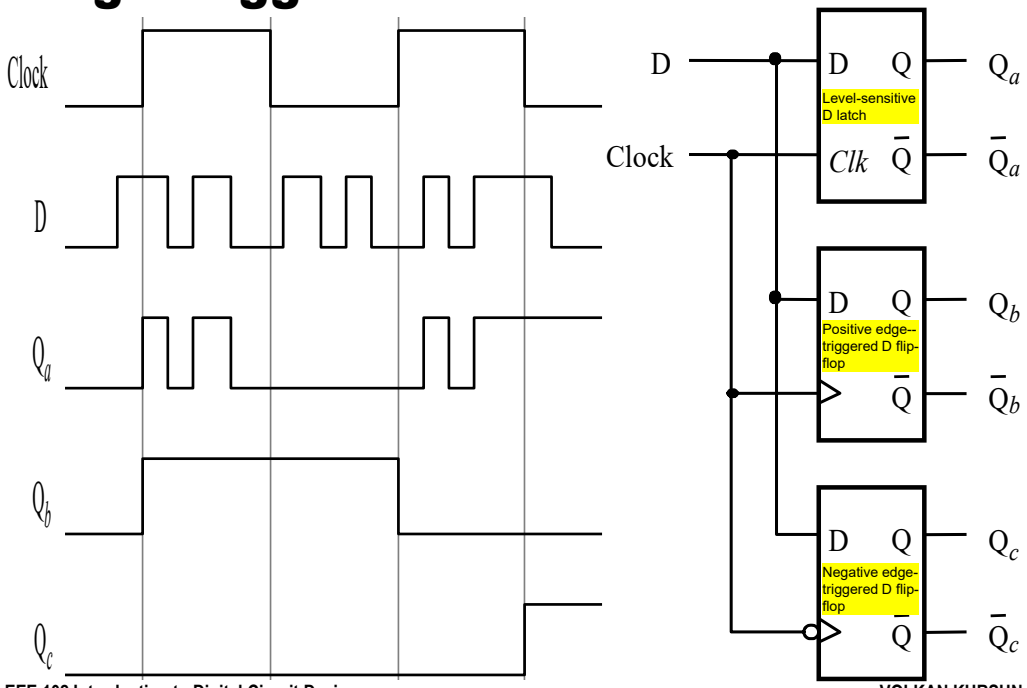
Master-Slave D Flip-Flop

- When Clock = 1: the master stage tracks the value of the D input while the slave stage does not change
- When Clock transitions from 1 to 0: master stage stops following the changes in the D input while the slave stage stores the value of the signal Q_m
- Since Q_m does not change while Clock = 0, the slave stage can undergo only one change of state (output switching) during a clock cycle: the slave stage changes state only at the negative edge of the clock signal
- Regardless of the number of changes in the D input to the master stage during one clock cycle, **there may be only one change at the Q output that corresponds to the D input stored at the negative edge of the clock signal**: **negative edge triggered**

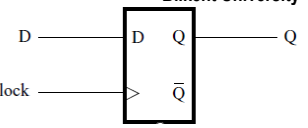
Master-Slave D Flip-Flop



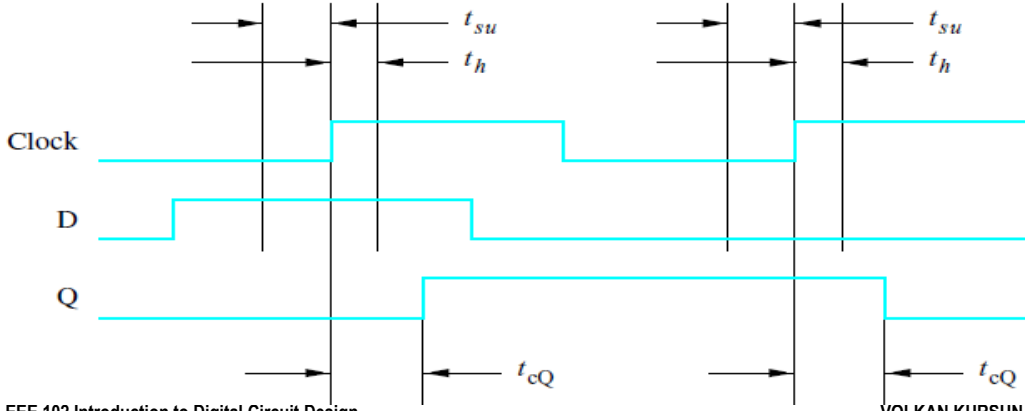
Edge-Triggered versus Level-Sensitive



D Flip-Flop Timing

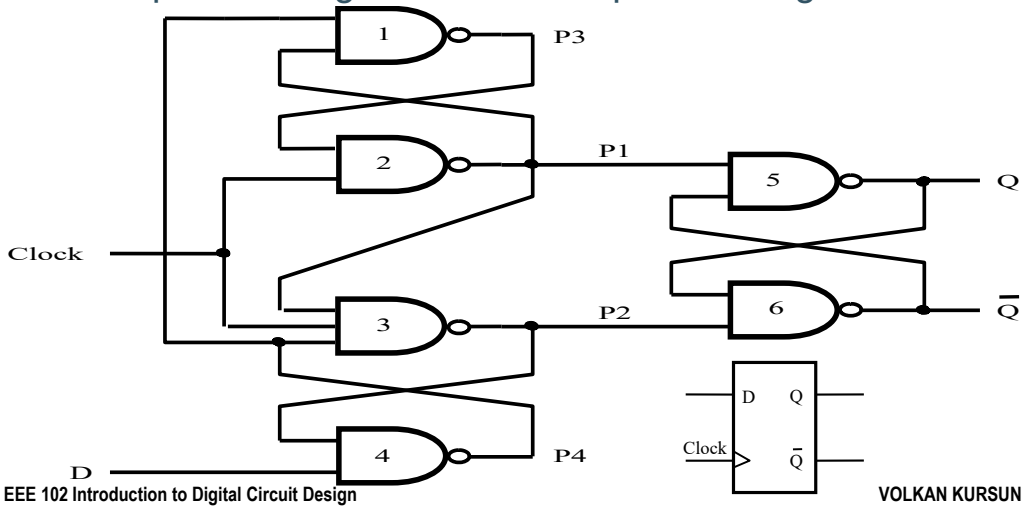


- Clock-to-Q propagation delay: t_{cQ}
- **Setup time**: D signal must be stable by at least the setup time before the Clk transitions from 0 to 1
- **Hold time**: D signal must remain stable by at least the hold time after the Clk 0 to 1 transition



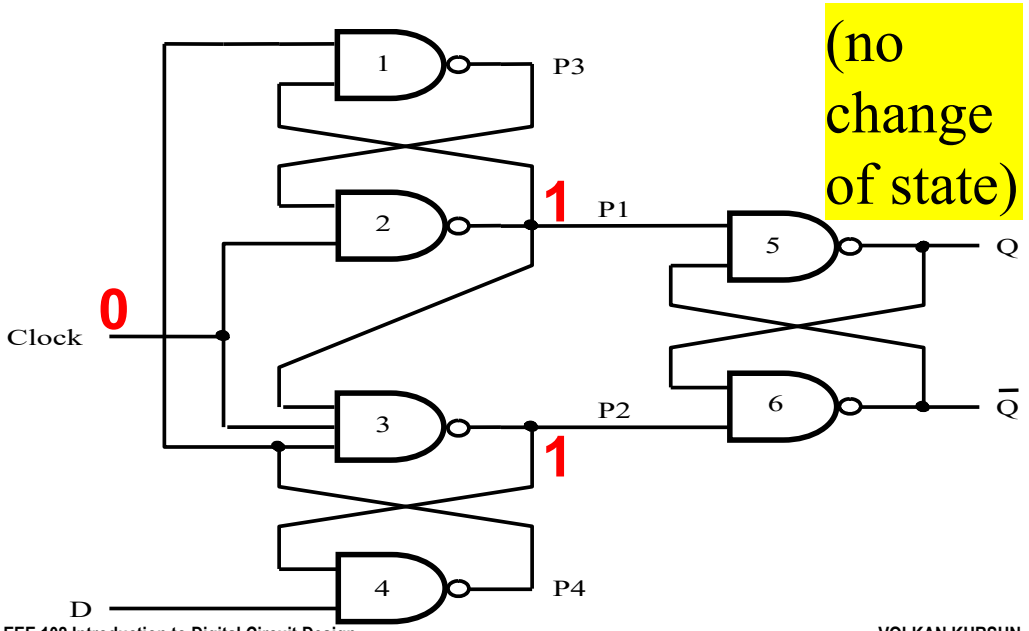
Edge-Triggered D: Alternative Design

- **Master-Slave D:** requires 8*2-input NAND gates and 3 inverters
- **Alternative design (positive edge triggered):** with only 5*2-input NAND gates and 1*3-input NAND gate



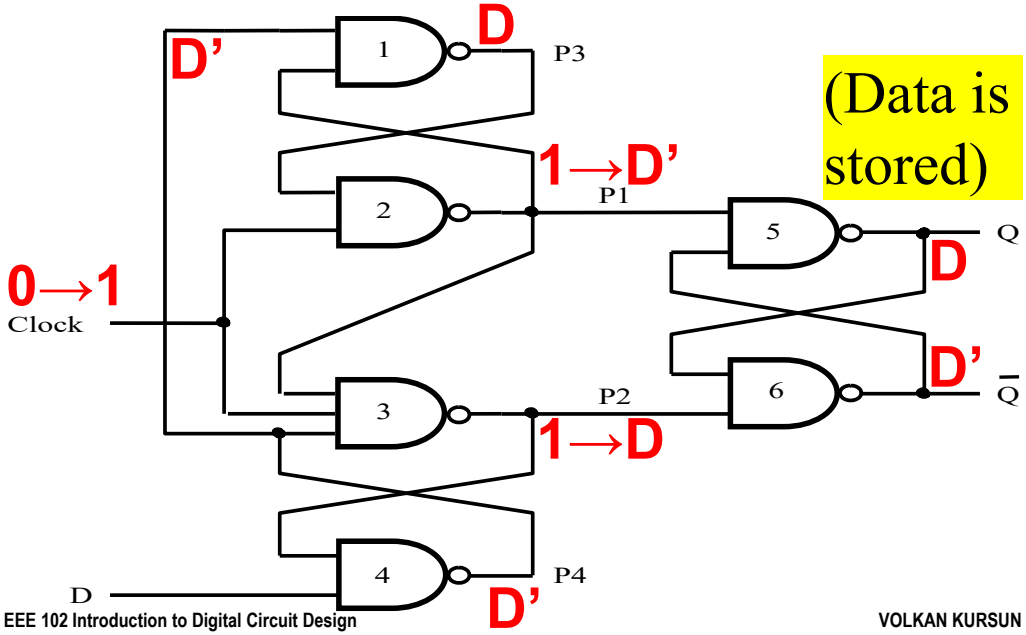
Edge-Triggered D: Alternative Design

- **When Clock = 0:** the flip-flop maintains its state



Edge-Triggered D: Alternative Design

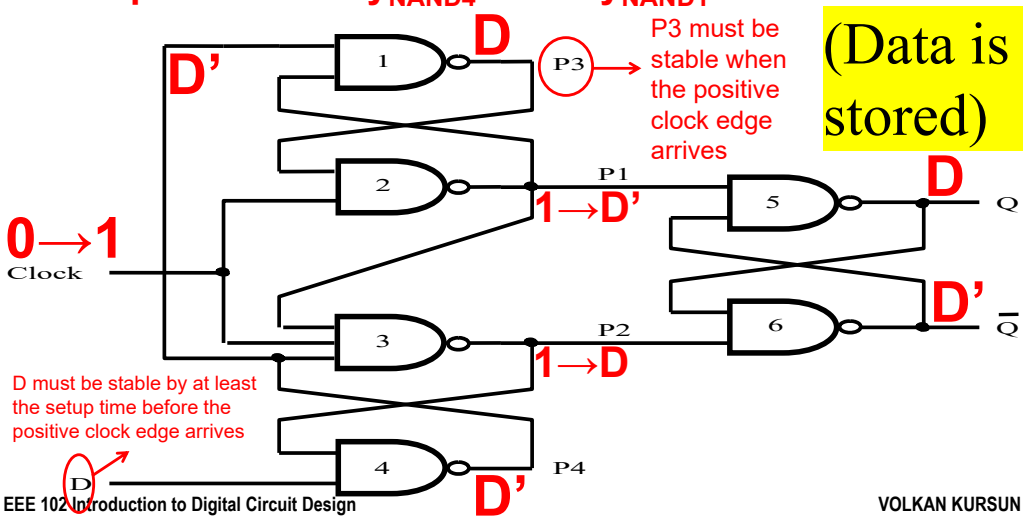
- **When Clock transitions from 0 to 1:** the data is stored



Edge-Triggered D: Setup Time

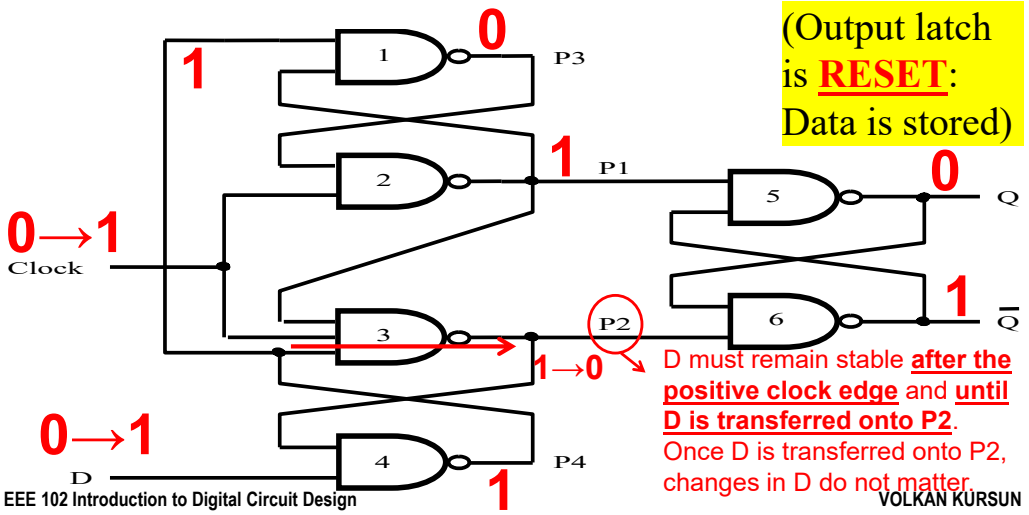
- **Setup time-1:** P3 is stable when the positive clock edge arrives which requires the input D to be stable by at least the setup time before the positive clock edge,

setup time = $Delay_{NAND4}$ + $Delay_{NAND1}$



Edge-Triggered D: Hold Time Scenario-1

- **Hold time scenario-1:** Assume D is 0 when the positive clock edge arrives. Once D (0) is transferred onto P2, P2 will keep P4 at 1 regardless of the subsequent changes in the D input: **Hold time scenario-1 = $Delay_{NAND3}$**



(Output latch is **RESET**: Data is stored)

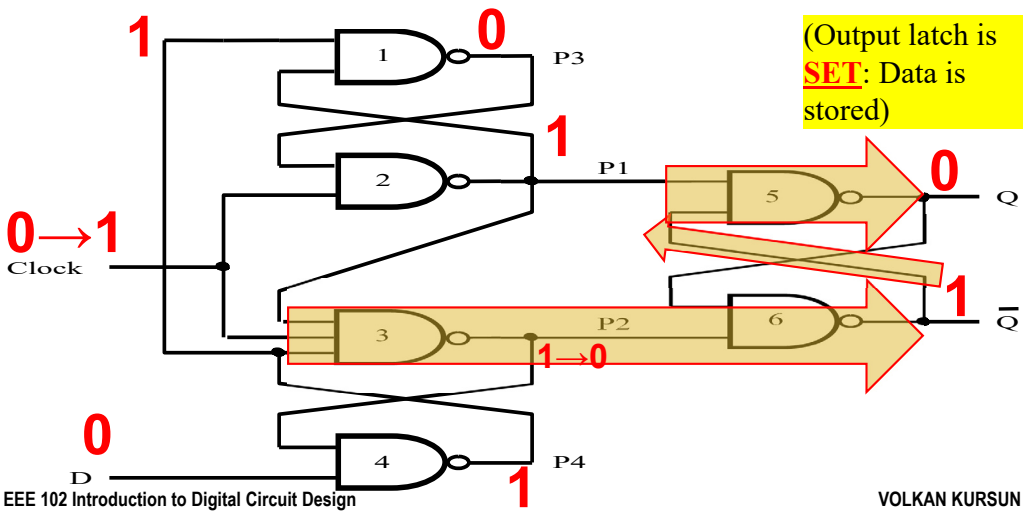
D must remain stable **after the positive clock edge** and **until D is transferred onto P2**. Once D is transferred onto P2, changes in D do not matter.

Edge-Triggered D: Delays Scenario-1

Delays with scenario-1 (D = 0):

Tclk-to-Q (scenario-1) = $Delay_{NAND3}$ + $Delay_{NAND6}$ + $Delay_{NAND5}$

Tclk-to-Qbar (scenario-1) = $Delay_{NAND3}$ + $Delay_{NAND6}$

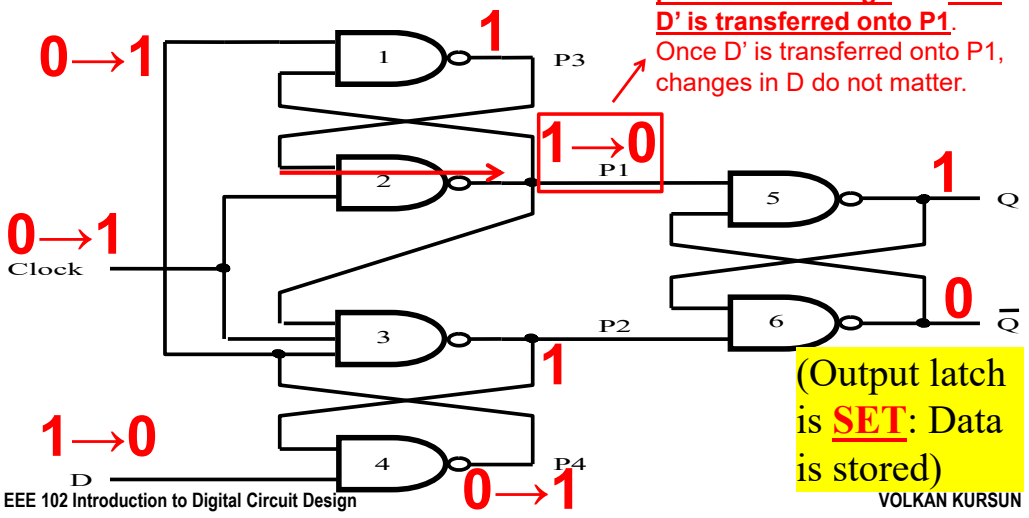


(Output latch is **SET**: Data is stored)

Edge-Triggered D: Hold Time Scenario-2

- **Hold time scenario-2:** Assume D is 1 when the positive clock edge arrives. With the current setup time requirement, when the positive clock edge arrives D is already transferred onto P3.

D must remain stable **after the positive clock edge** and **until D' is transferred onto P1**. Once D' is transferred onto P1, changes in D do not matter.



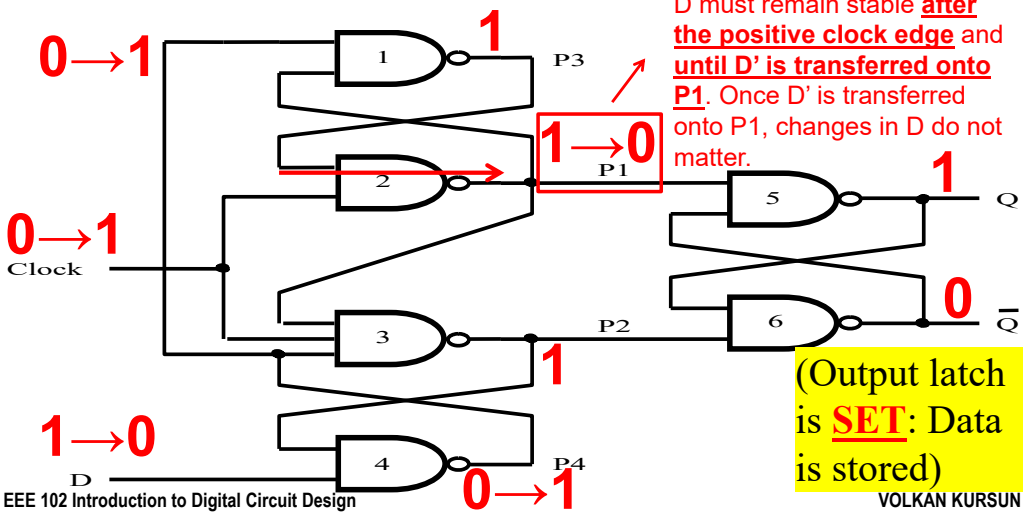
(Output latch is **SET**: Data is stored)

Edge-Triggered D: Hold Time Scenario-2

- **Hold time scenario-2:** Once the clock transitions from 0 to 1, D' (0) is transferred onto P1 and P1 keeps P2 and P3 stable at 1 regardless of the subsequent changes in the **D input and P4**.

Hold time scenario-2 = $Delay_{NAND2}$

D must remain stable **after the positive clock edge** and **until D' is transferred onto P1**. Once D' is transferred onto P1, changes in D do not matter.



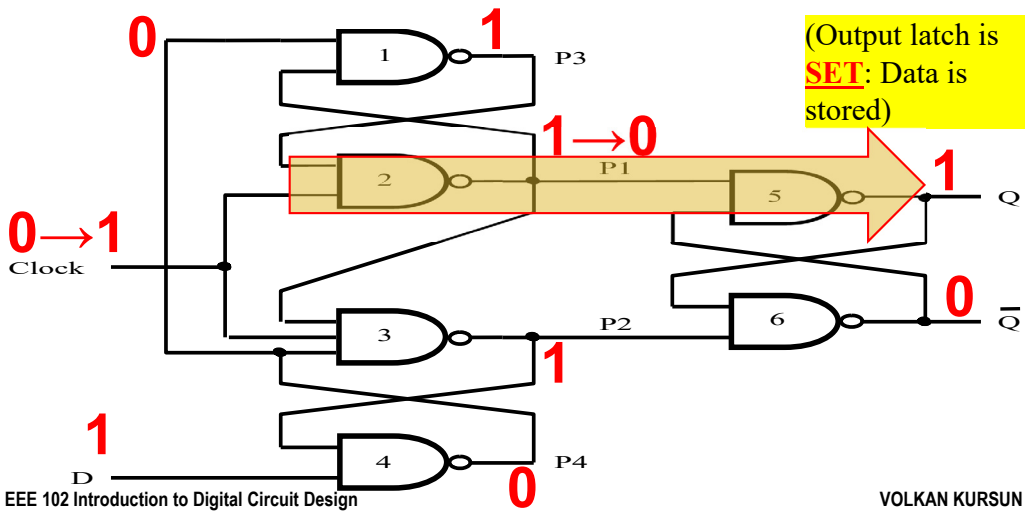
(Output latch is **SET**: Data is stored)

Edge-Triggered D: Delays Scenario-2

- Delays with scenario-2 (D = 1):

Tclk-to-Q (scenario-2) = $Delay_{NAND2}$ + $Delay_{NAND5}$

Tclk-to-Qbar (scenario-2) = Tclk-to-Q + $Delay_{NAND6}$

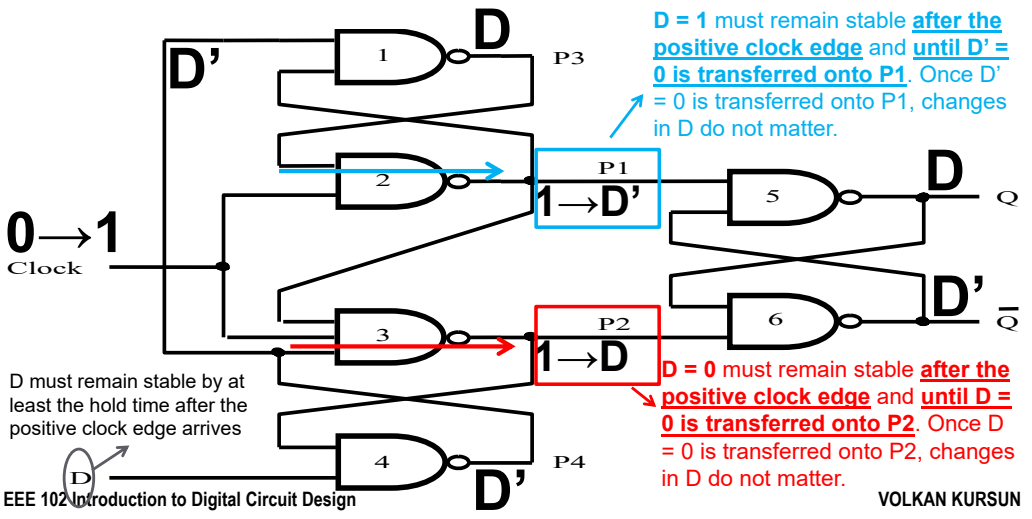


Edge-Triggered D: Hold Time

- Hold time is the maximum of the hold time requirements for scenarios 1 and 2

Hold time = Max (hold time scenario-1, hold time scenario-2)

Hold time = $Delay_{NAND3}$



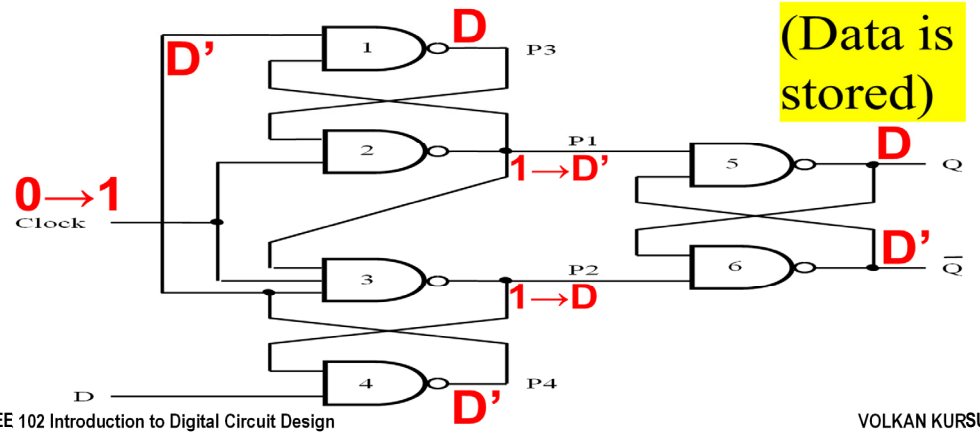
Edge-Triggered D: Delays

- Flip-flop delays with these setup and hold time requirements:

Tclk-to-Q = max(Tclk-to-Q (scenario-1), Tclk-to-Q (scenario-2))
= $Delay_{NAND3}$ + $Delay_{NAND6}$ + $Delay_{NAND5}$

Tclk-to-Qbar = max(Tclk-to-Qbar (scenario-1), Tclk-to-Qbar (scenario-2)) = $Delay_{NAND2}$ + $Delay_{NAND5}$ + $Delay_{NAND6}$

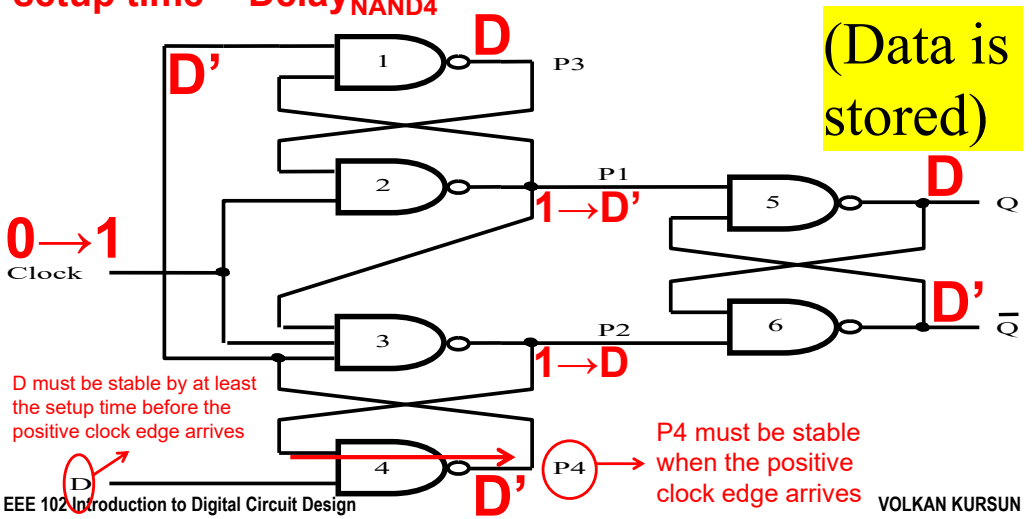
- Tclk-to-Q > Tclk-to-Qbar



Alternative Setup Time Restriction

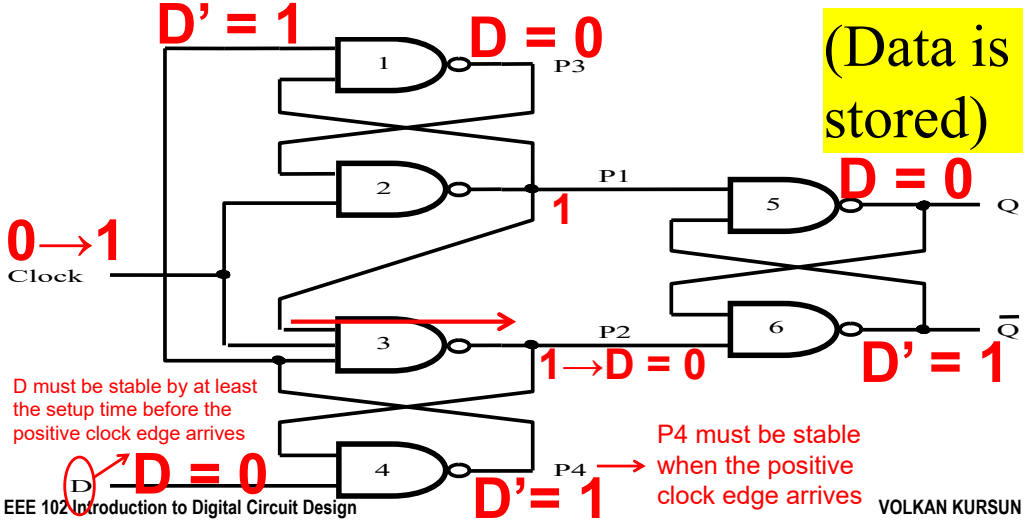
- Setup time-2: P4 is stable when the positive clock edge arrives which requires the input D to be stable by at least the setup time before the positive clock edge

setup time = $Delay_{NAND4}$



Alternative Hold Time Scenario-1

- **Hold time scenario-1:** Assume **D = 0** arrives by NAND4 delay before the positive clock edge. P1 is already 1 when the positive clock edge arrives. D = 0 will propagate to P2 after the delay of NAND3.

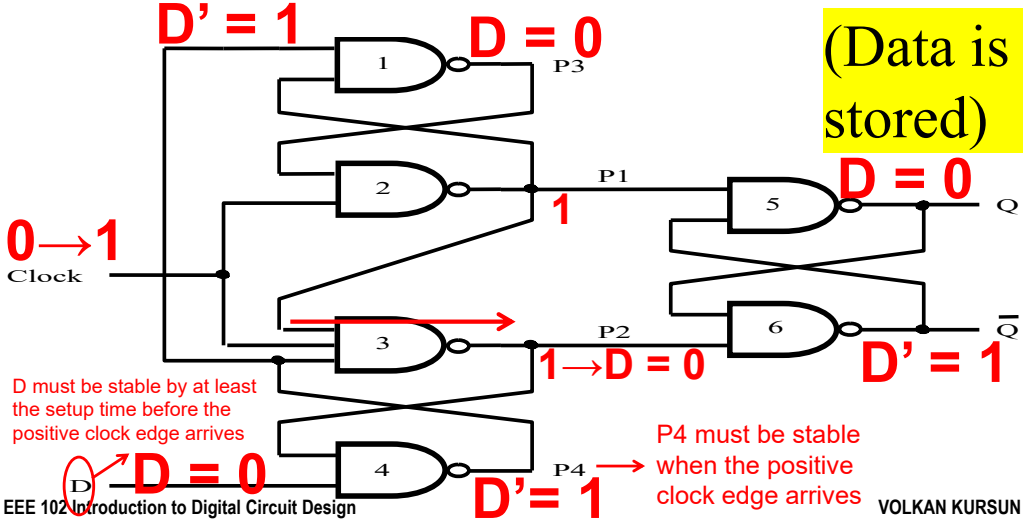


VOLKAN KURSUN Bilkent University

Alternative Hold Time Scenario-1

- **Hold time scenario-1:** Once D (0) is transferred onto P2, P2 will keep P4 at 1 regardless of the subsequent changes in the D input:

Hold time scenario-1 = $Delay_{NAND3}$

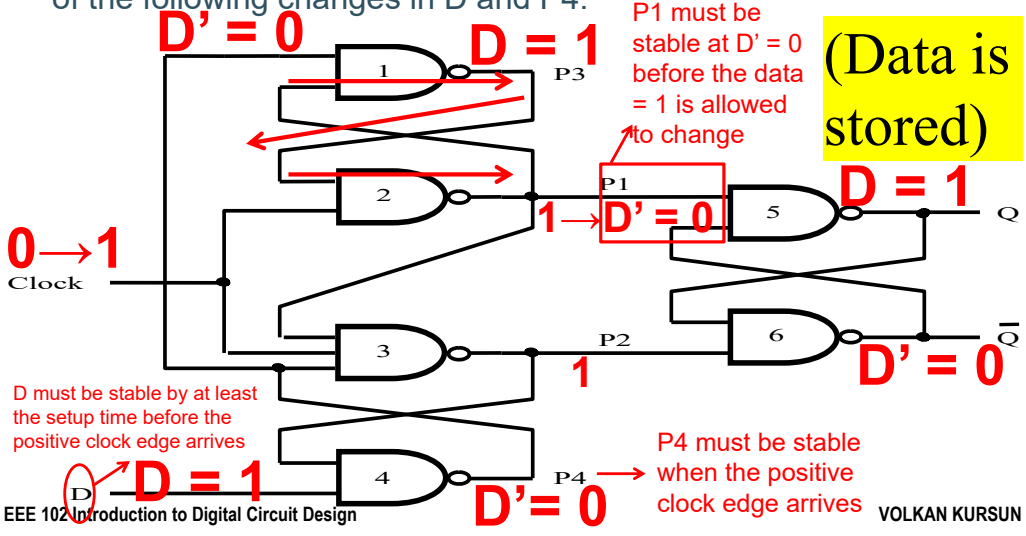


EEE 102 Introduction to Digital Circuit Design VOLKAN KURSUN

VOLKAN KURSUN Bilkent University

Alternative Hold Time Scenario-2

- **Hold time scenario-2:** Assume **D = 1** arrives by NAND4 delay before the positive clock edge. The data (D = 1) must not be allowed to change until P1 stabilizes to D' = 0. Once D' is transferred onto P1, P1 will keep P2 and P3 stable at 1 regardless of the following changes in D and P4.



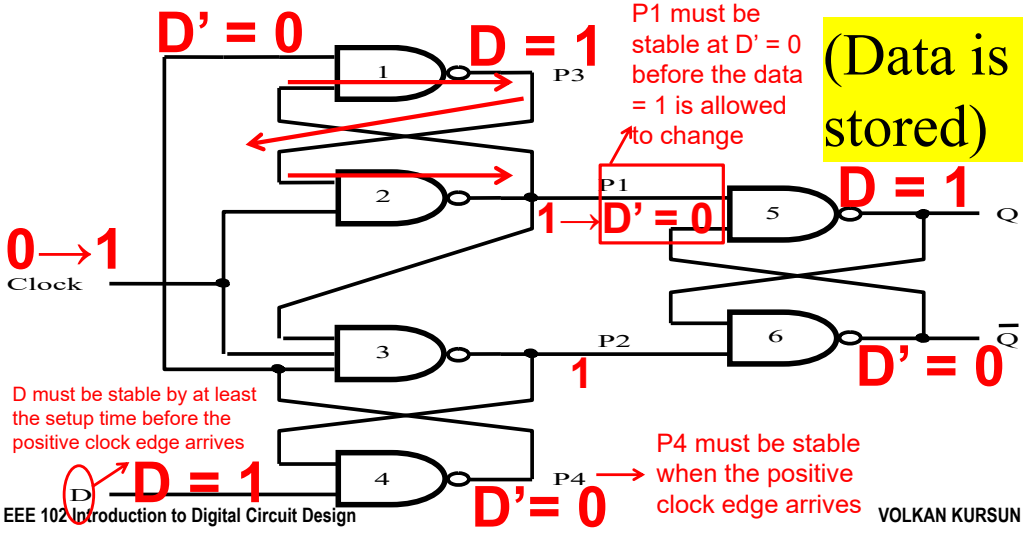
EEE 102 Introduction to Digital Circuit Design VOLKAN KURSUN

VOLKAN KURSUN Bilkent University

Alternative Hold Time Scenario-2

- **Hold time scenario-2:** Assume **D = 1** arrives by NAND4 delay before the positive clock edge. After the positive clock edge, D' must propagate through NAND1 and then through NAND2 before D is allowed to change.

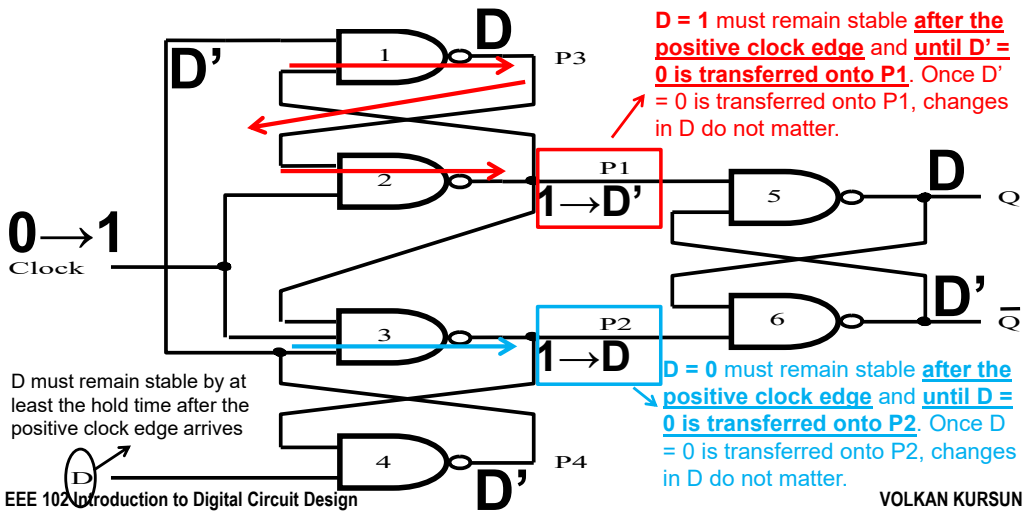
Hold time scenario-2 = $Delay_{NAND1}$ + $Delay_{NAND2}$



EEE 102 Introduction to Digital Circuit Design VOLKAN KURSUN

Alternative Hold Time Restriction

- Hold time is the maximum of the hold time requirements for the two possible data input scenarios: Hold time = Max (scenario-1, scenario-2) = $Delay_{NAND1} + Delay_{NAND2}$



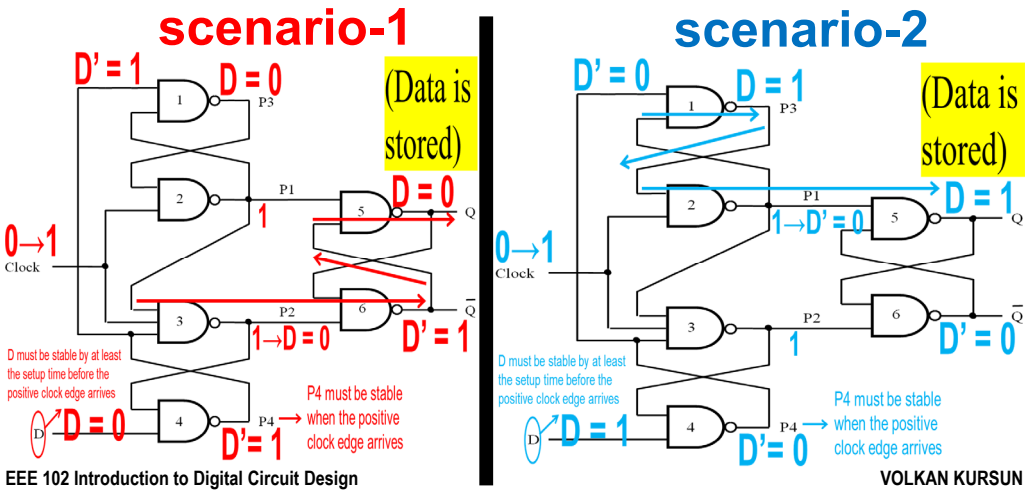
Alternative Setup Definition Delays

T_{CQ} scenario-1 = $Delay_{NAND3} + Delay_{NAND6} + Delay_{NAND5}$

T_{CQ} scenario-2 = $Delay_{NAND1} + Delay_{NAND2} + Delay_{NAND5}$

$T_{CQ} = \max(T_{CQ}\text{ scenario-1}, T_{CQ}\text{ scenario-2})$

$= Delay_{NAND3} + Delay_{NAND6} + Delay_{NAND5}$

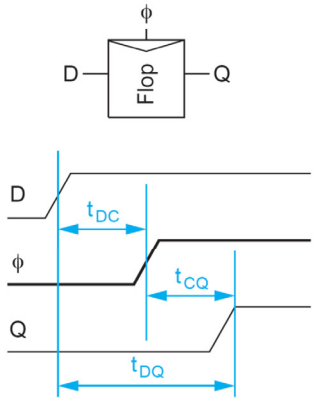


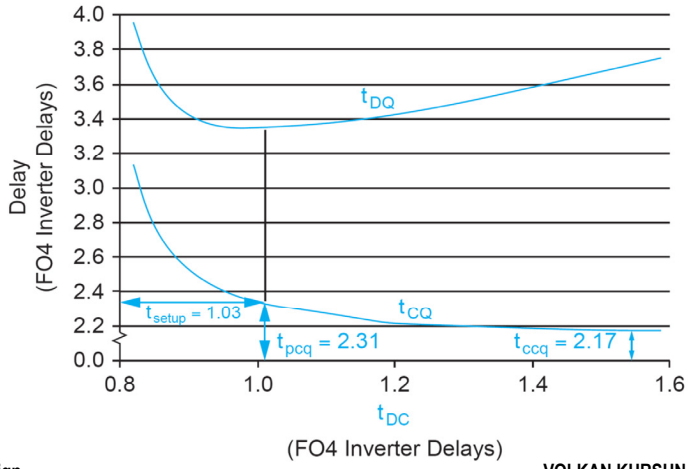
Comparison of the Two Setup/Hold Times

- Setup/Hold Times-1 (longer setup time but shorter hold time):
  - Setup time-1 = $Delay_{NAND4} + Delay_{NAND1}$
  - Hold time-1 = $Delay_{NAND3}$
  - $T_{CQ-1} = Delay_{NAND3} + Delay_{NAND6} + Delay_{NAND5}$
- Setup/Hold Times-2 (shorter setup time but longer hold time):
  - Setup time-2 = $Delay_{NAND4}$
  - Hold time-2 = $Delay_{NAND1} + Delay_{NAND2}$
  - $T_{CQ-2} = Delay_{NAND3} + Delay_{NAND6} + Delay_{NAND5}$
- The Clock-to-Q delays are identical for both scenarios
- However, the setup time and clock-to-Q delay are both parts of the clock period in a pipeline and **second setup-time requirement would allow a higher clock frequency pipeline**
- **Question: how small can the setup time be?**

Effect of Setup Time on Delay

- If the data arrival time gets too close to the clock edge, the delay starts to increase and eventually the flip-flop malfunctions
- Choose an appropriate setup time depending on clock-to-Q delay and pipeline clock frequency goals





Edge-Triggered D VHDL

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
    PORT (    D, Clock  : IN   STD_LOGIC ;
              Q         : OUT  STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS ( Clock )
    BEGIN
        IF Clock'EVENT AND Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Sensitivity list contains only the clock signal because clock is the only signal that can trigger an output change

'EVENT attribute refers to any change in the clock signal

If Clock'EVENT AND Clock = '1' checks if the clock has changed and if it has a value of 1 after the change: indicates a **positive clock edge**

Edge-Triggered D VHDL Code -2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY flipflop IS
    PORT (      D, Clock   : IN    STD_LOGIC ;
                Q          : OUT   STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        Q <= D ;
    END PROCESS ;
END Behavior ;
```

No sensitivity list: the VHDL code inside the process block will run continuously (the program loops back to the start of the block after executing the last line).

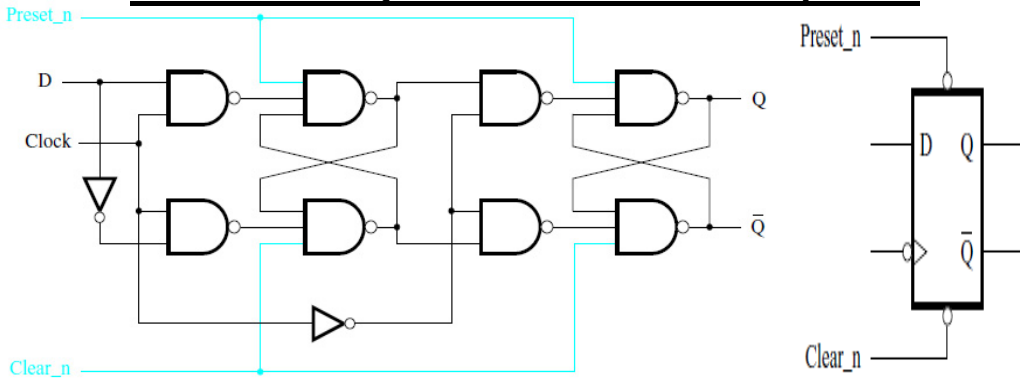
WAIT UNTIL construct implies that the sensitivity list includes only the Clock signal

WAIT UNTIL Clock'EVENT AND Clock = '1' to make an assignment to the Q output: indicates a **positive clock edge**

D Flip-Flop with Clear and Preset

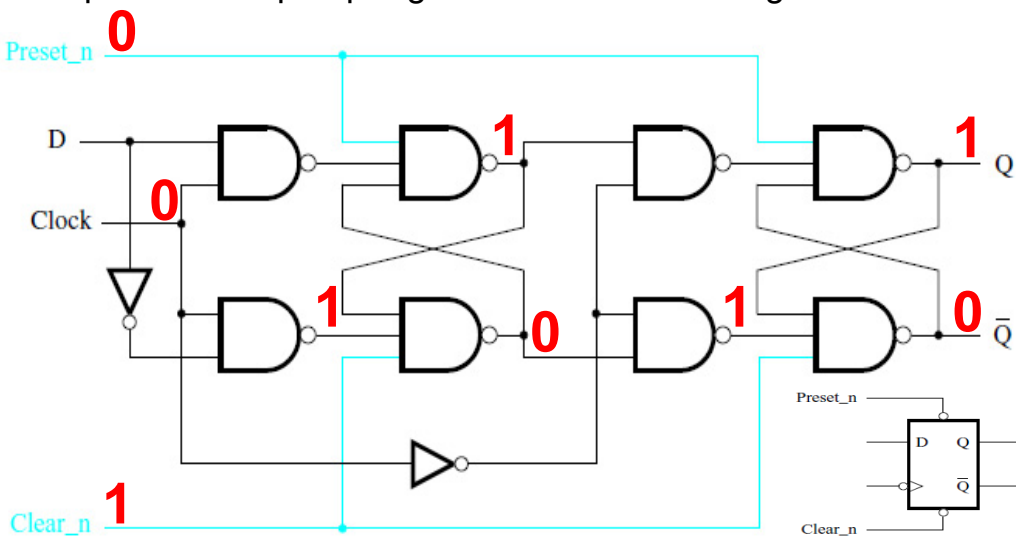
- Typically, it is necessary to be able to initialize sequential circuits to a specific state
- Clear = 0: Clear the Q output to 0
- Preset = 0: Set the Q output to 1
- Example: master-slave flip-flop based on NAND gates with **active-low asynchronous clear and preset**

Preset_n	Clear_n	Q	Q'
0	1	1	0
1	0	0	1
1	1	Normal op.	



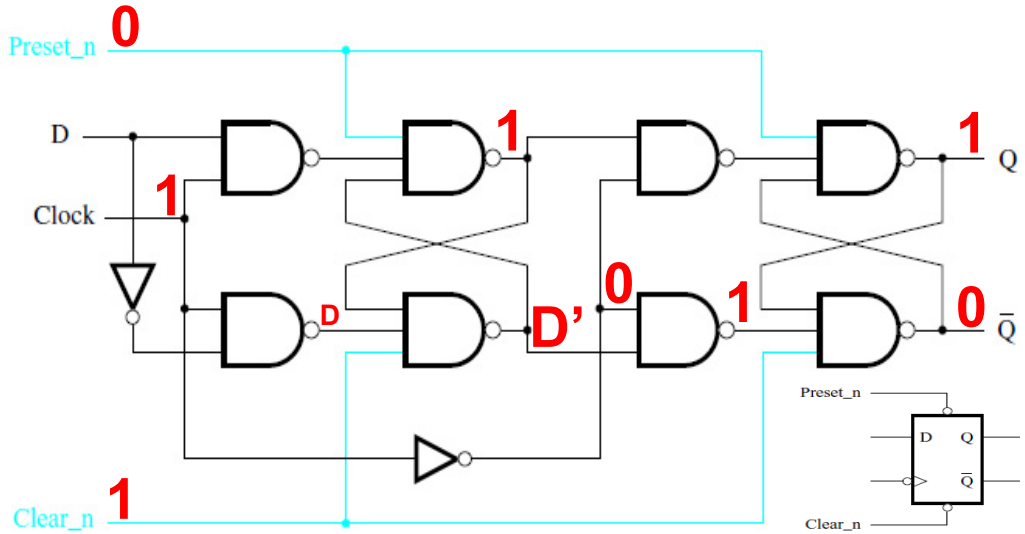
D Flip-Flop Preset with CLK = 0

- **Preset_n = 0**, Clear_n = 1: **Preset the Q output to 1**
- **Asynchronous active-low preset**: Preset_n signal is used to preset the flip-flop regardless of the clock signal



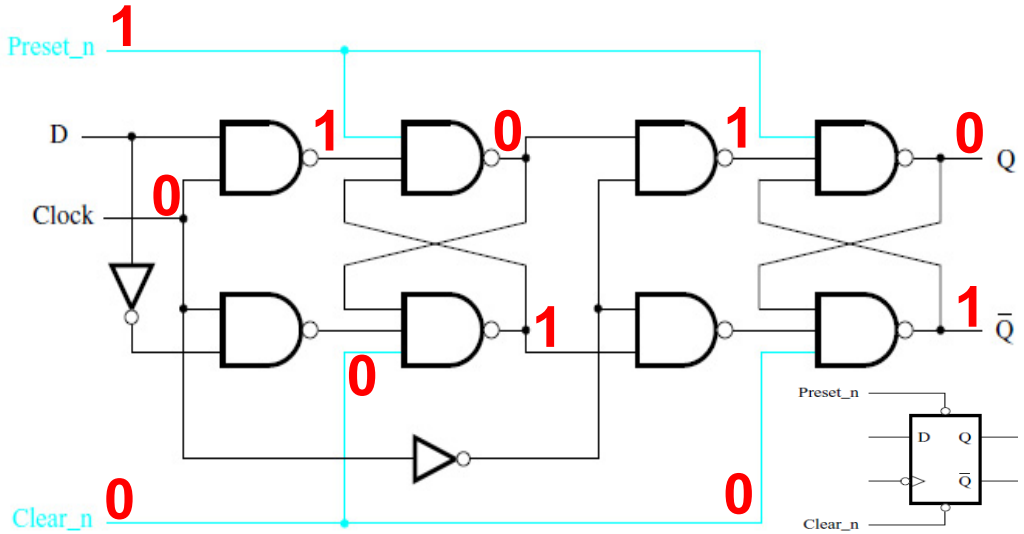
D Flip-Flop Preset with CLK = 1

- **Preset_n = 0**, Clear_n = 1: **Preset the Q output to 1**
- **Asynchronous active-low preset**: Preset_n signal is used to preset the flip-flop regardless of the clock signal



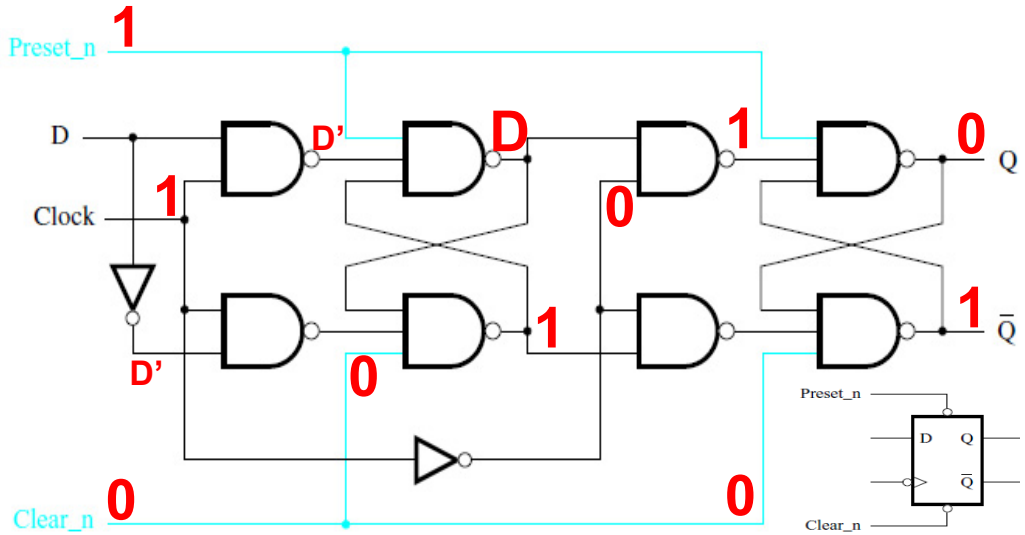
D Flip-Flop Clear with CLK = 0

- **Clear_n = 0**, Preset_n = 1: **Clear the Q output to 0**
- **Asynchronous active-low clear**: Clear_n signal is used to clear the flip-flop regardless of the clock signal



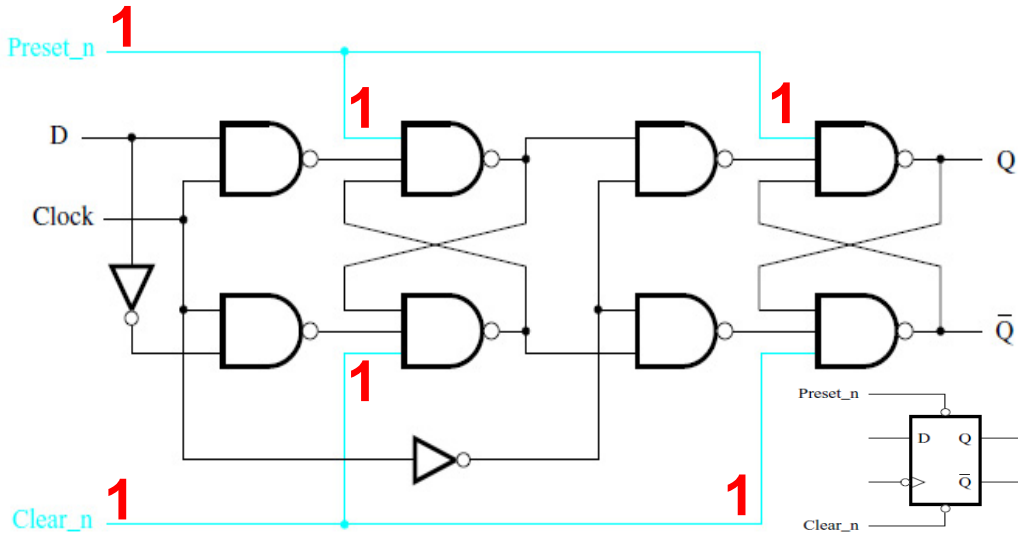
D Flip-Flop Clear with CLK = 1

- **Clear_n = 0**, Preset_n = 1: **Clear the Q output to 0**
- **Asynchronous active-low clear**: Clear_n signal is used to clear the flip-flop regardless of the clock signal



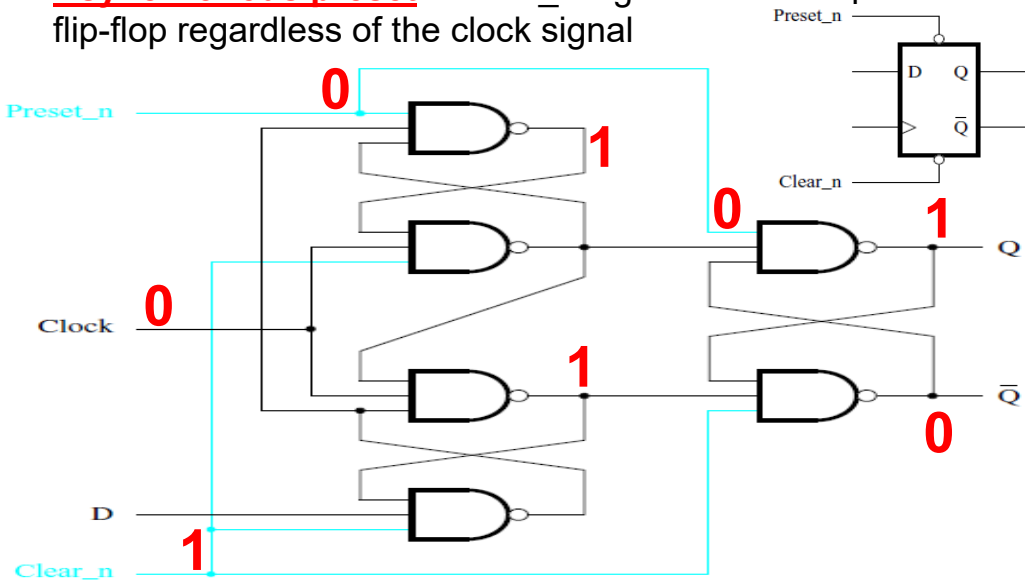
D Flip-Flop with Clear = Preset = 1

- **Clear_n = 1**, **Preset_n = 1**: clear and preset signals have no effect on the circuit operation, **normal negative edge-triggered master-slave flip-flop operation**



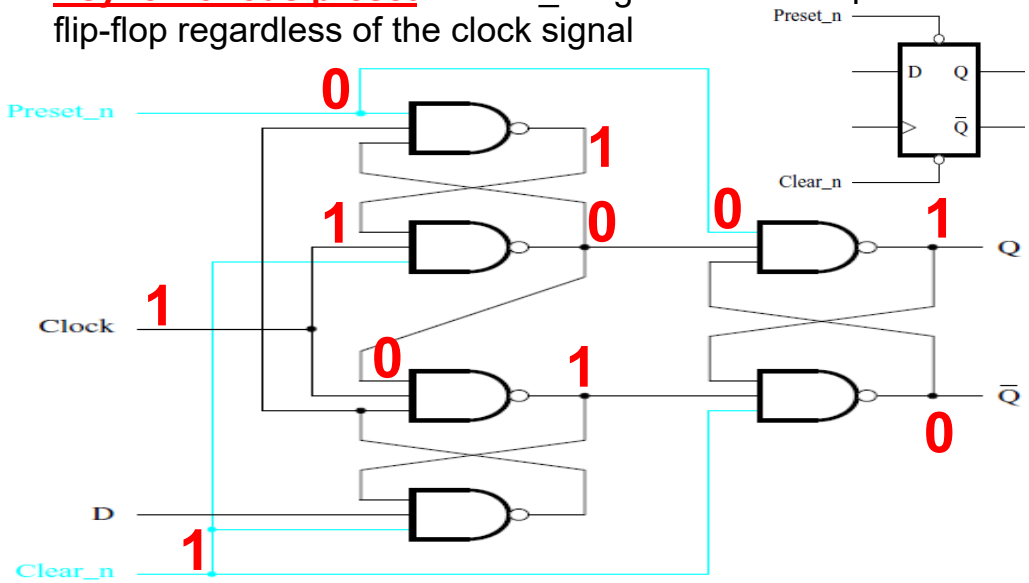
Alternative D Flip-Flop Preset with CLK = 0

- **Preset_n = 0**, Clear_n = 1: **Preset the Q output to 1**
- **Asynchronous preset**: Preset_n signal is used to preset the flip-flop regardless of the clock signal



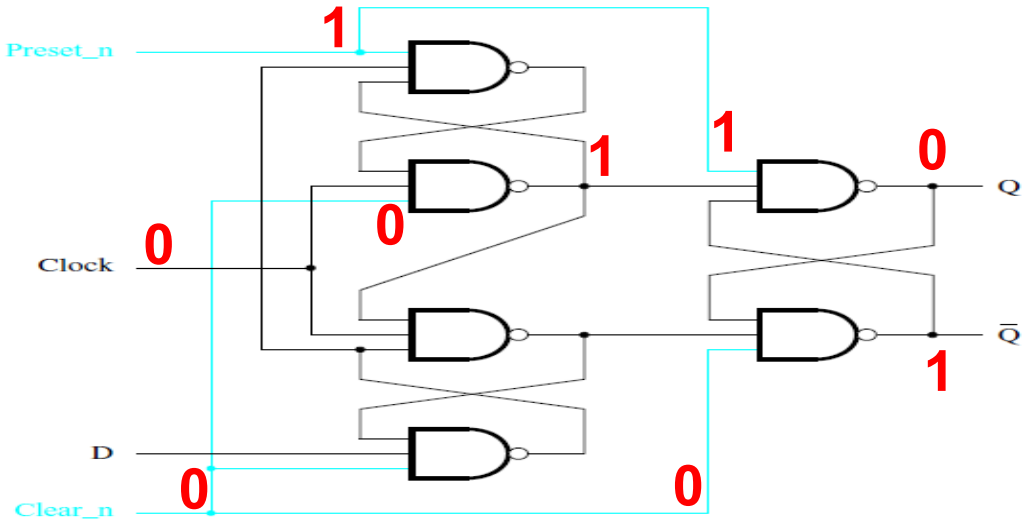
Alternative D Flip-Flop Preset with CLK = 1

- **Preset_n = 0**, Clear_n = 1: **Preset the Q output to 1**
- **Asynchronous preset**: Preset_n signal is used to preset the flip-flop regardless of the clock signal



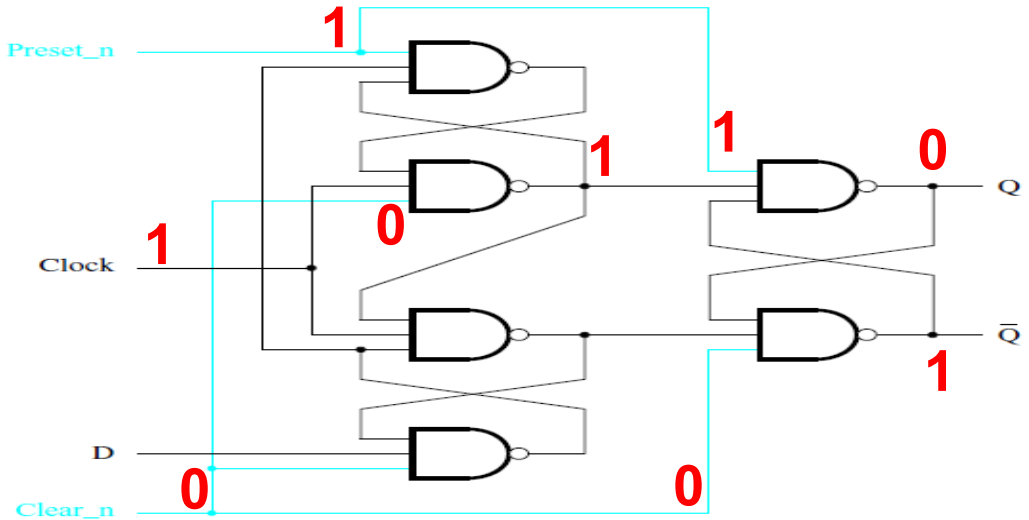
Alternative D Flip-Flop Clear with CLK = 0

- **Clear_n = 0**, Preset_n = 1: **Clear the Q output to 0**
- **Asynchronous active-low clear**: Clear_n signal is used to clear the flip-flop regardless of the clock signal



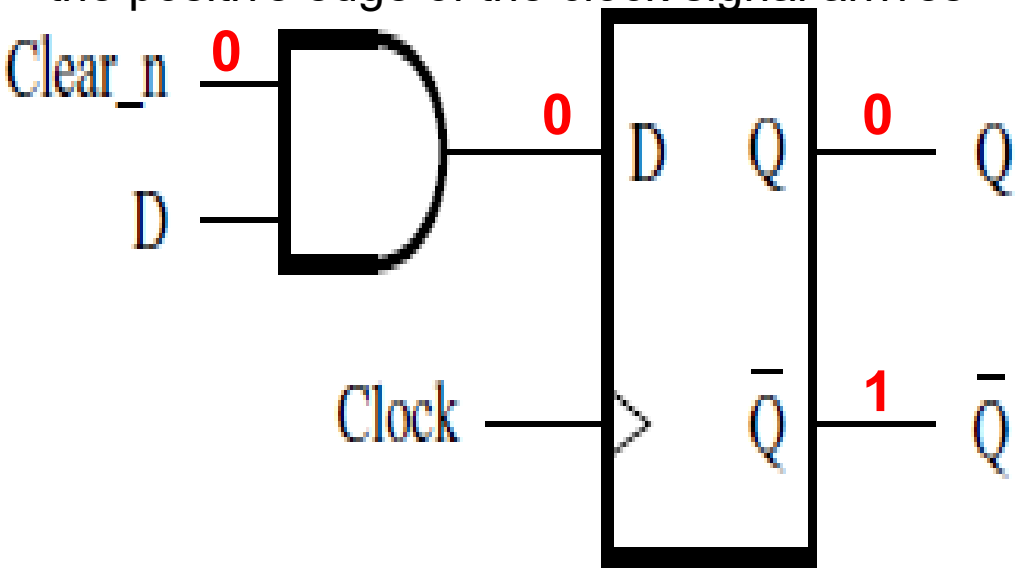
Alternative D Flip-Flop Clear with CLK = 1

- **Clear_n = 0**, Preset_n = 1: **Clear the Q output to 0**
- **Asynchronous active-low clear**: Clear_n signal is used to clear the flip-flop regardless of the clock signal



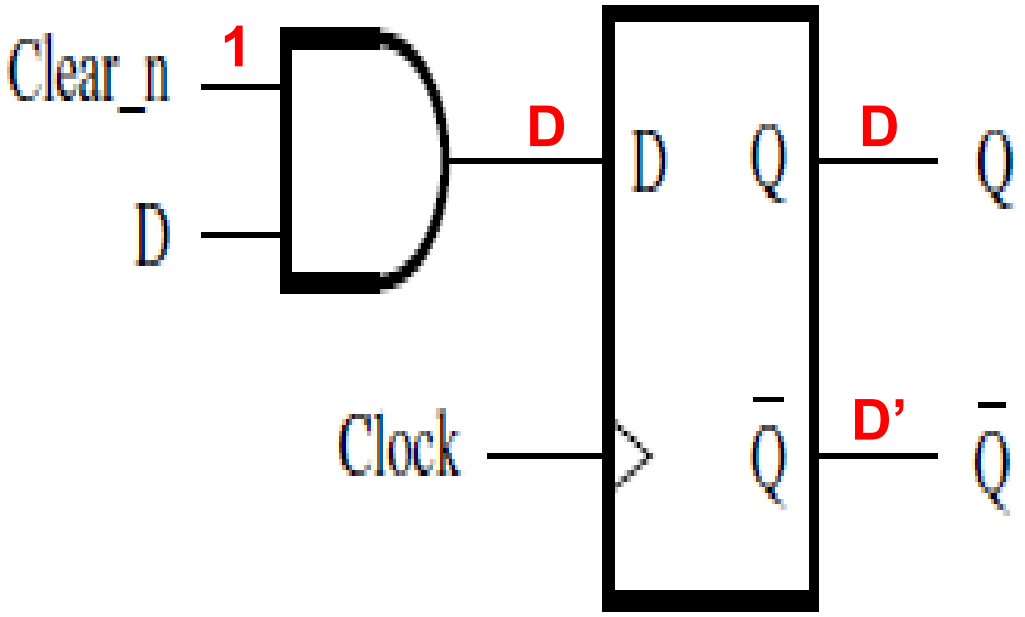
D Flip-Flop with Synchronous Clear

- **Clear_n = 0**: Clear the Q output to 0 when the positive edge of the clock signal arrives



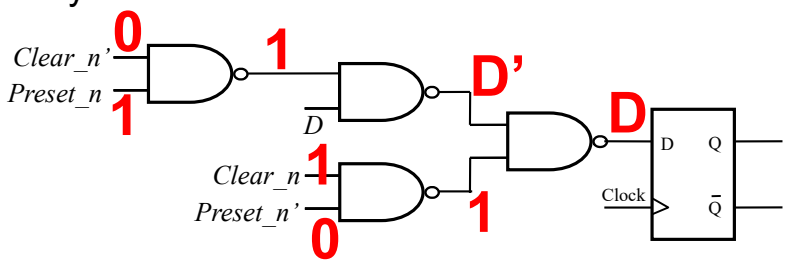
D Flip-Flop with Synchronous Clear

- When Clear_n = 1, the flip-flop operates normally

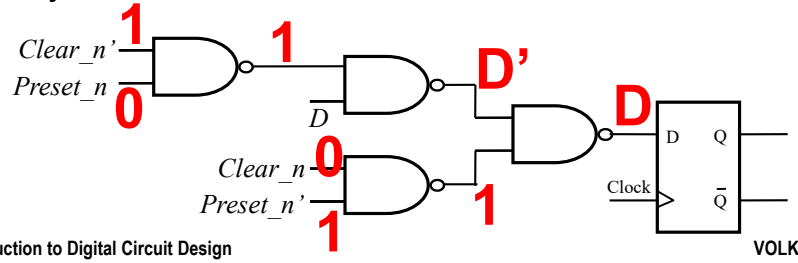


Synchronous Clear and Preset

- When Clear_n = 1, Preset_n = 1, the flip-flop operates normally

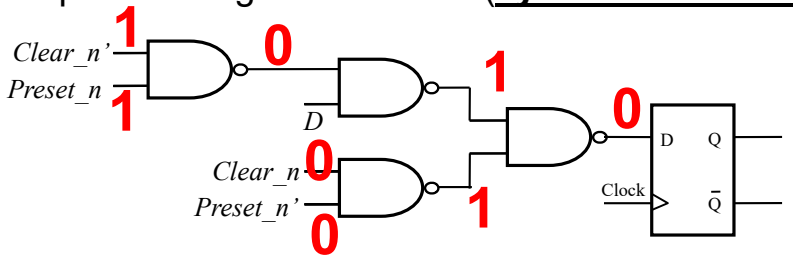


- When Clear_n = 0, Preset_n = 0, the flip-flop operates normally

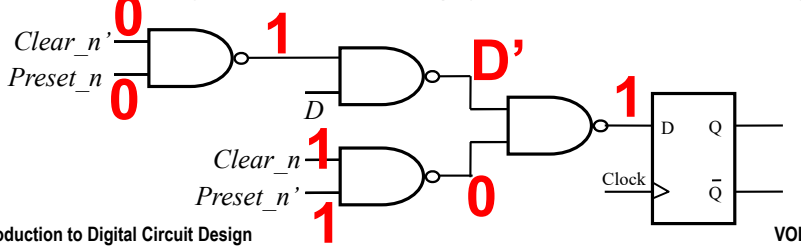


Synchronous Clear and Preset

- When **Clear_n = 0**, Preset_n = 1, the flip-flop is **cleared** with the positive edge of the clock (**synchronous clear**)



- When Clear_n = 1, **Preset_n = 0**, the flip-flop is **set** with the positive edge of the clock (**synchronous preset**)



Asynchronous Reset VHDL

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
    PORT ( D, Resetn, Clock      : IN     STD_LOGIC ;
           Q                     : OUT    STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            Q <= '0' ;
        ELSIF Clock'EVENT AND Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Whenever the process is triggered, check Resetn first (without checking the clock signal): **asynchronous reset**

Asynchronous Preset and Clear VHDL

```
process(CLK,CLEAR_L,PRESET_L)
begin
    if PRESET_L='0' then
        Q<='1';
    elsif CLEAR_L='0' then
        Q<='0';
    elsif falling_edge(CLK) then
        Q<=D;
    end if;
end process;
```


Synchronous Active-Low Reset VHDL

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
    PORT ( D, Resetn, Clock   : IN     STD_LOGIC ;
           Q                  : OUT    STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Resetn = '0' THEN
            Q <= '0' ;
        ELSE
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

No sensitivity list: the VHDL code inside the process block will run continuously (the program loops back to the start of the block after executing the last line).

WAIT UNTIL positive clock edge before checking the Resetn signal and clearing the output if Resetn = 0: **synchronous reset**

Synchronous Reset VHDL Code-2

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
    PORT (  D, Resetn, Clock    : IN     STD_LOGIC ;
            Q                   : OUT    STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS ( Clock ) IS
    BEGIN
        IF RISING_EDGE ( Clock )  THEN
            IF Resetn = '0'  THEN
                Q <= '0' ;
            ELSE
                Q <= D ;
            END IF ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

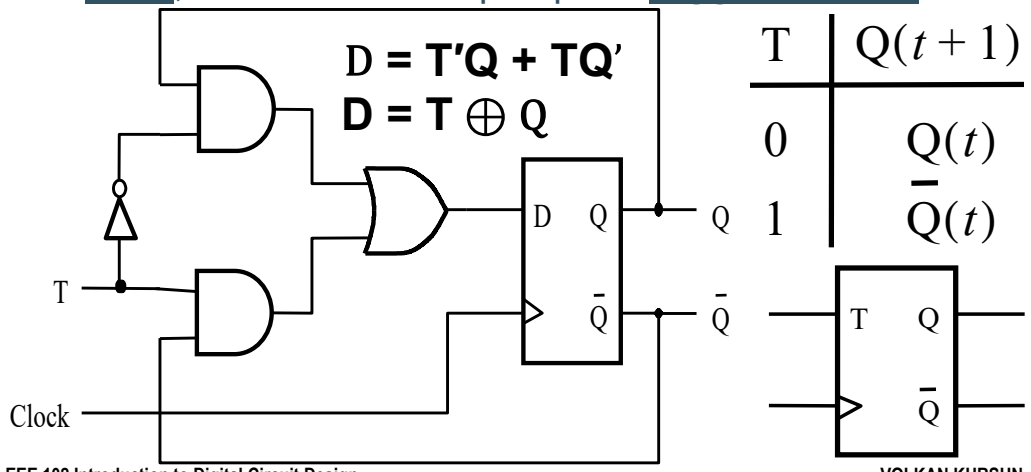
RISING_EDGE(Clock)

Outline

- Edge-Triggered D Flip-Flop
- T Flip-Flop
- JK Flip-Flop

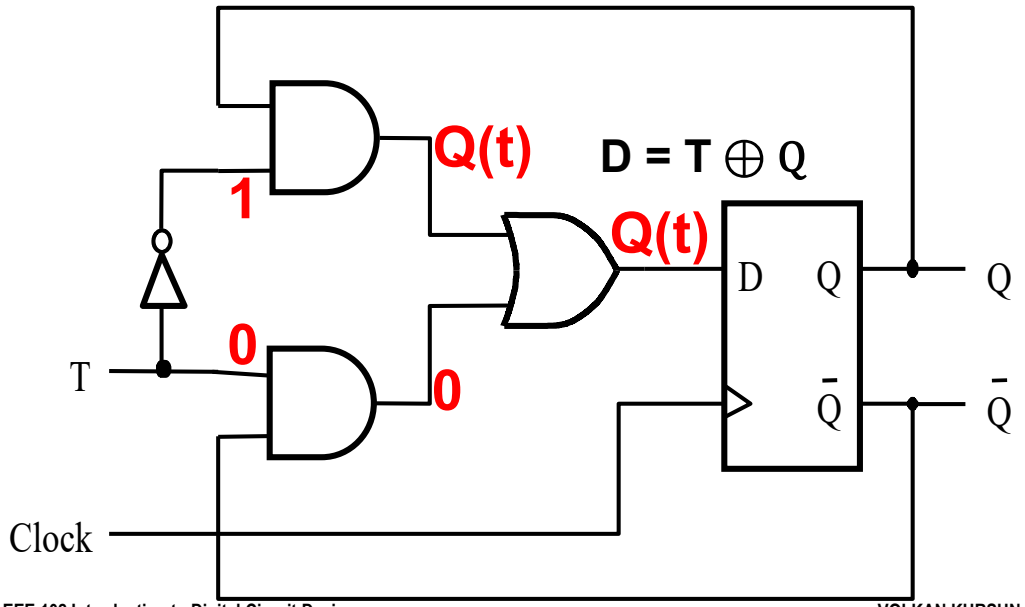
Toggle (T) Flip-Flop

- The input signal D is equal to either Q or Q' depending on the control signal labeled T
- If **T = 0**, D = Q and the flip-flop will **maintain its state**
- If **T = 1**, D = Q' and the flip-flop will **toggle its state**



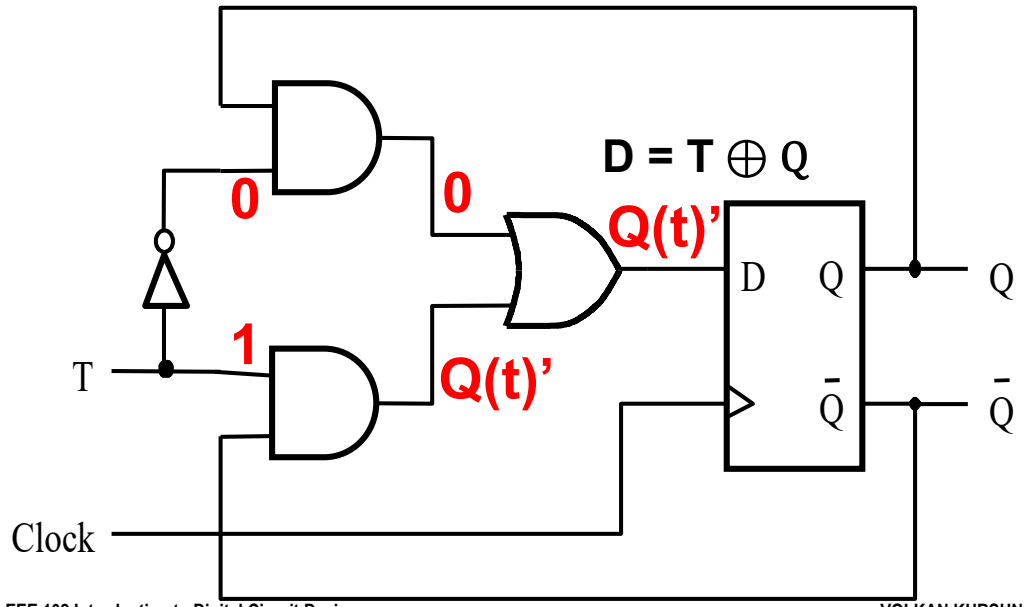
Toggle (T) Flip-Flop

- If **T = 0**, D = Q and the flip-flop will **maintain state**

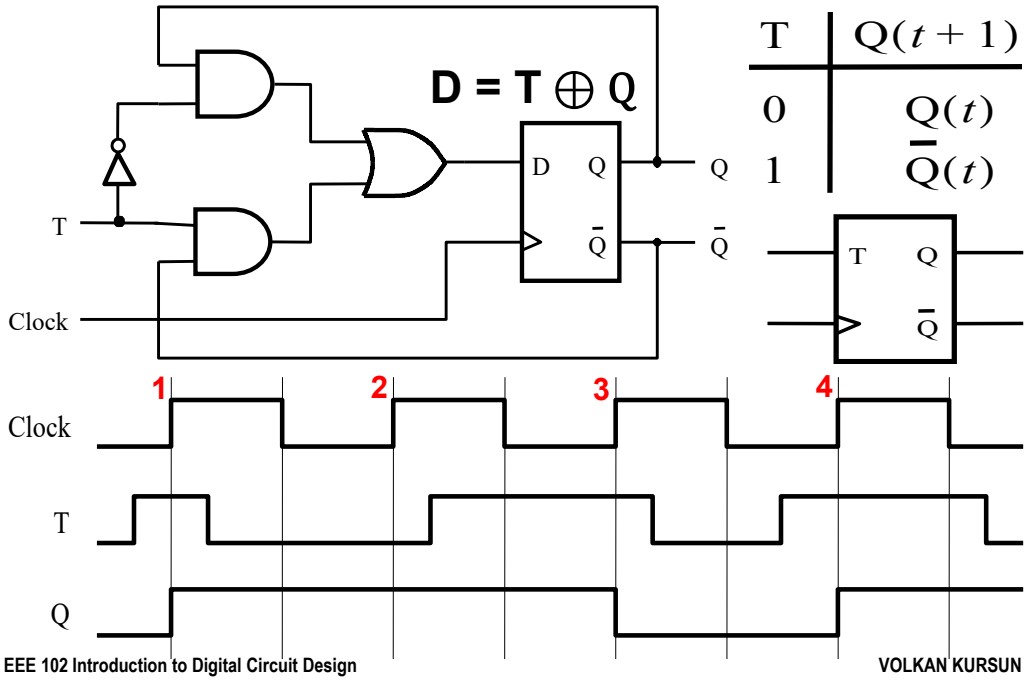


Toggle (T) Flip-Flop

- If **T = 1**, D = Q' and the flip-flop will **toggle** state



Toggle (T) Flip-Flop Timing



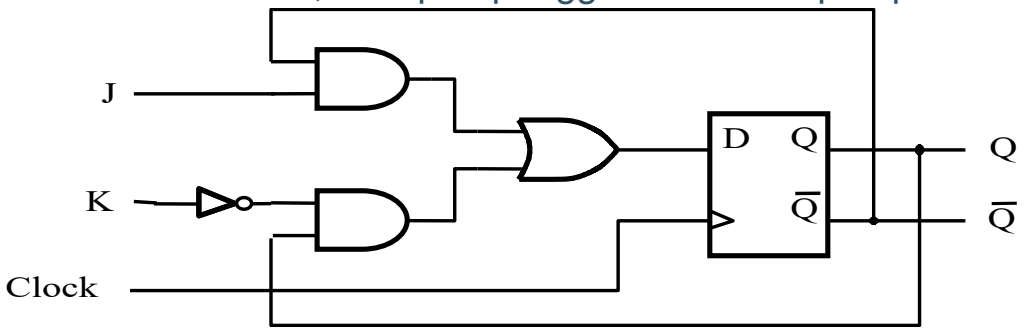
T	$Q(t+1)$
0	$Q(t)$
1	$\bar{Q}(t)$

Outline

- Edge-Triggered D Flip-Flop
- T Flip-Flop
- JK Flip-Flop

JK Flip-Flop

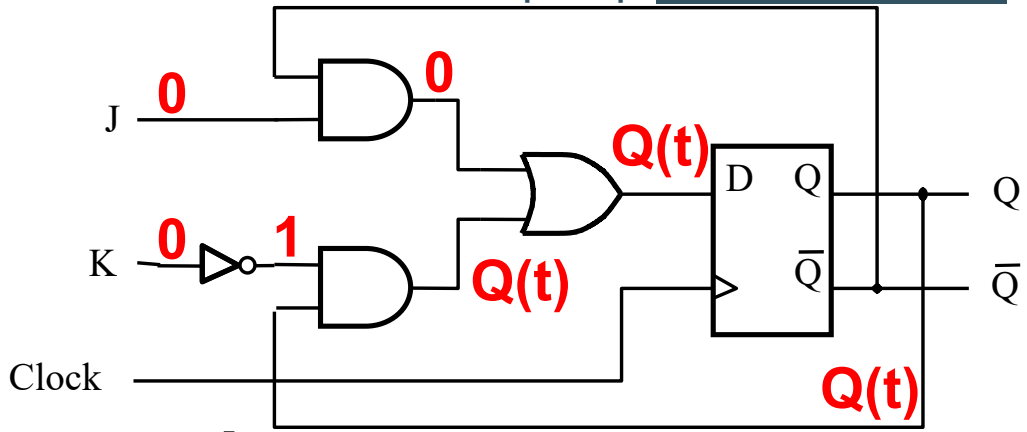
- Behaves as an SR flip-flop where J = S and K = R for all input values except J = K = 1
- When J = K = 1, JK flip-flop toggles like a T flip-flop



J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

JK Flip-Flop

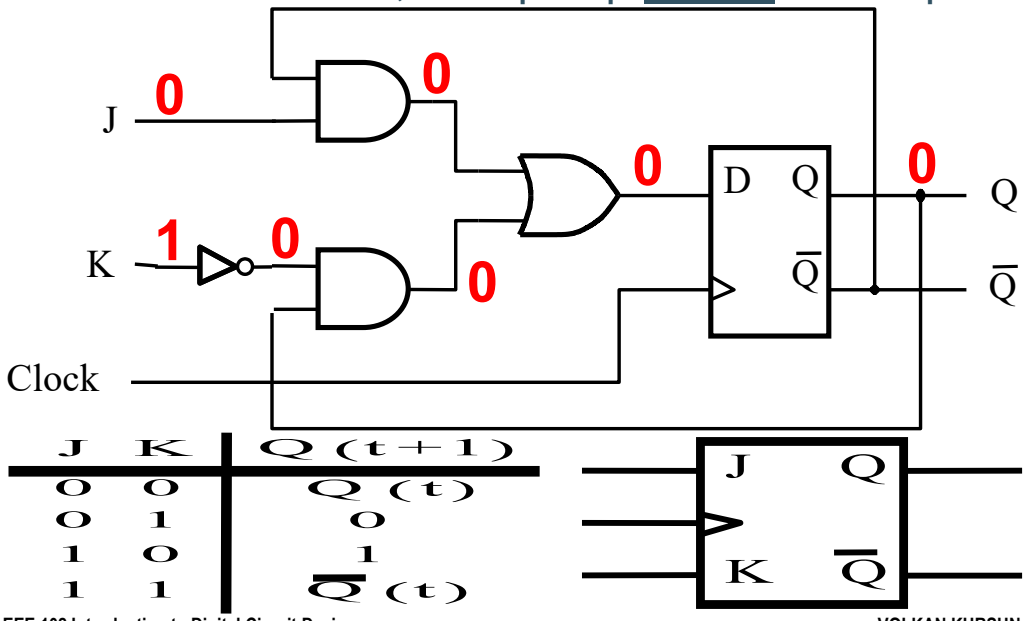
- When **J = K = 0**, JK flip-flop **maintains state**



J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

JK Flip-Flop

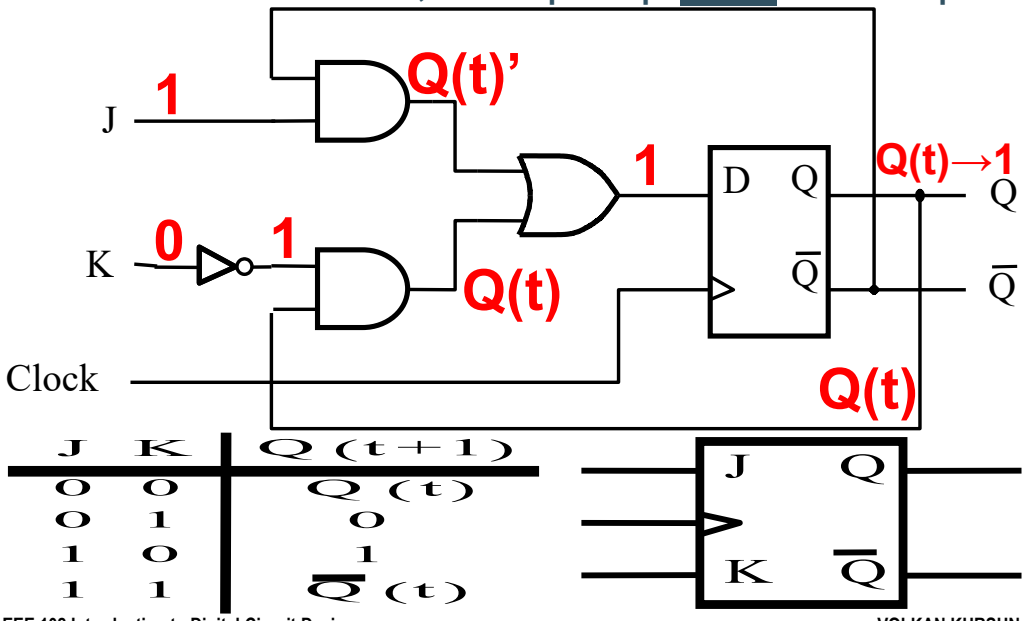
- When **JK = 0b01**, JK flip-flop **clears** the output



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

JK Flip-Flop

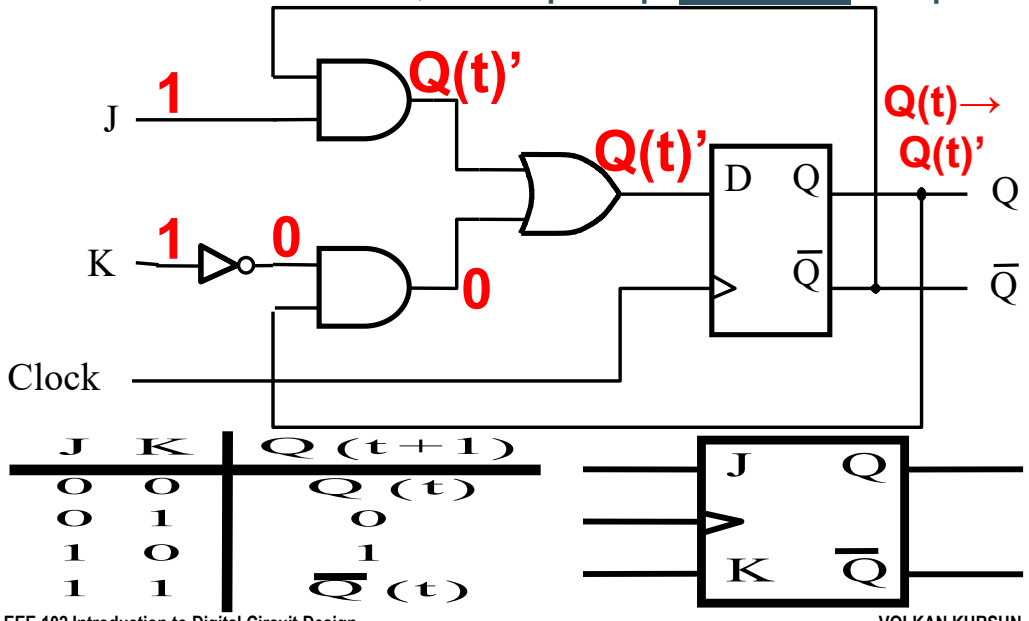
- When **JK = 0b10**, JK flip-flop **sets** the output



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

JK Flip-Flop

- When **JK = 0b11**, JK flip-flop **toggles** output



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

JK Flip-Flop VHDL

```
process(CLK) begin
    if rising_edge(CLK) then
        Q<=D;
    endif;
end process;
D<=Q when J='0' and K='0' else
   '1' when J='1' and K='0' else
   '0' when J='0' and K='1' else
   not Q;
```

Describes the D flip-flop

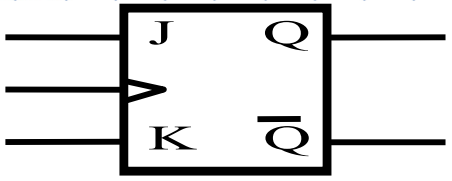
Describes the combinational circuit that produces the D input of the D flip-flop

Alternatively, we can use the concatenation operator

```
JK<=J&K;
with JK select
    D<=Q when "00",
       '1' when "10",
       '0' when "01",
       not Q when others;
```

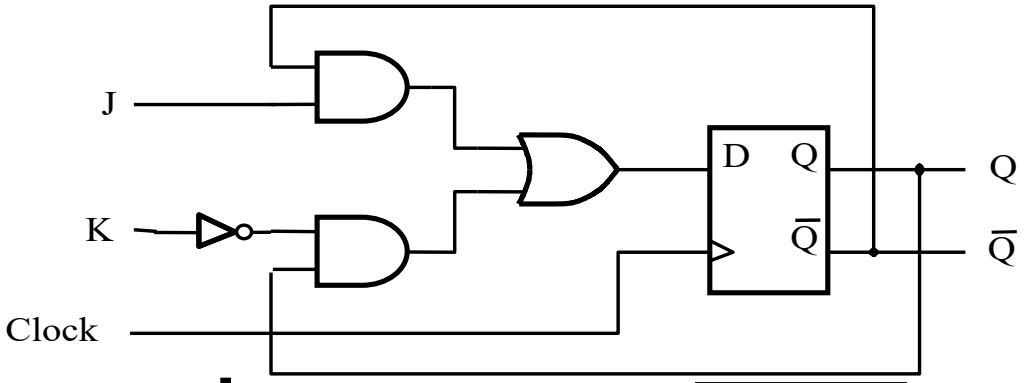
More compact code using the concatenation operator to define a new variable JK

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)



JK Flip-Flop: Inputs Tied

- If J and K are tied together, a JK flip-flop **turns into a T flip-flop**



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q̄(t)