

Sequential Logic Circuits

- Sequential circuits are specified by a *time sequence* of i/ps, o/ps, and internal states
- Need *storage elements* (**Registers**), *clock signals* (**CLK**), and feedback
- **Two types: Synchronous and Asynchronous**
- **Synchronous: O/ps change at discrete and predefined instants of time**
- **Asynchronous: Change of o/p not predefined, and function of orders of change of i/ps – uses time delay devices**


- * ***Storage elements (Memory)***

- form the *feedback path*

- * ***Memory stores binary***

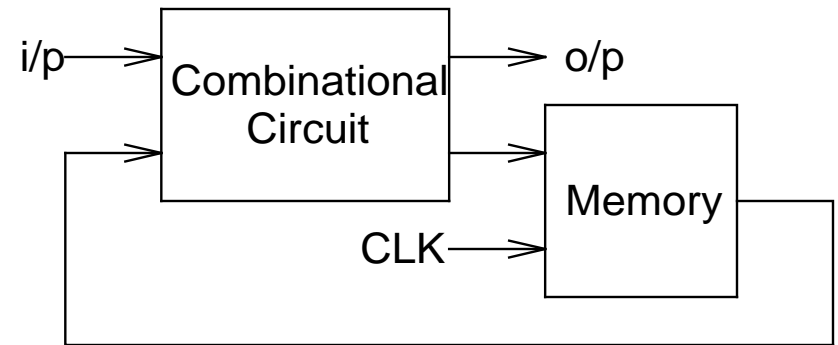
- information*

- * ***Defines the *state* of the sequential circuit at that time***

- * ***Next state determined by i / ps and present state of the memory*** 

- * ***These may change states of the memory elements as well***

- * ***Synchronous circuits employ signals that affect the storage elements only at discrete instants of time***



- * We will be focusing our attention on *Synchronous Circuits* first
- * The *timing device* that achieves this *synchronization*
 - ⇒ **Clock Generator Circuits**
 - ⇒ Provides a *periodic train* of clock pulses
- * *Storage elements affected only with the arrival of each such pulse*
- * Clock signals along with other i/ps cause the desired *change* of o/p in the system
- * These are known as **Clocked Sequential Circuits**

- * Most frequently used and very robust in terms of any kind of *instability* problems
- * Storage elements in clocked sequential circuits are known as **Flip-Flops** (F / Fs)
- * **F/F: *Binary storage device capable of storing 1 bit of information***
- * Many of these may be used to store *multiple bits* of information
- * **State of F/F can change only during a clock pulse transition**

- * Transition from one state to next occurs only at ***pre-determined*** time intervals dictated by the clock pulse
- * A F/F can maintain a binary state ***indefinitely*** (as long as power is supplied to it) until directed by an i/p signal to ***switch state***
- * Major ***differences*** between various types of F/Fs:
 - **Number of i/ps they have**
 - **The manner in which the i/ps affect the binary o/p state**
- * Most basic type of F/Fs operate with ***signal levels***
⇒ **Known as Latches**

Latches:

- * *Most basic and primitive form of F /F*
- * *Useful for storing binary information and in asynchronous sequential circuits, but not practical to be used in synchronous sequential circuits*

SR Latch:

- * Known as **Set-Reset Latch**
- * *Has the property of memory*
- * *Two types:*
 - **NOR Latch**
 - **NAND Latch**

NOR Latch:

* **Two cross-coupled NOR gates**

* **R: *Reset***

S: *Set*

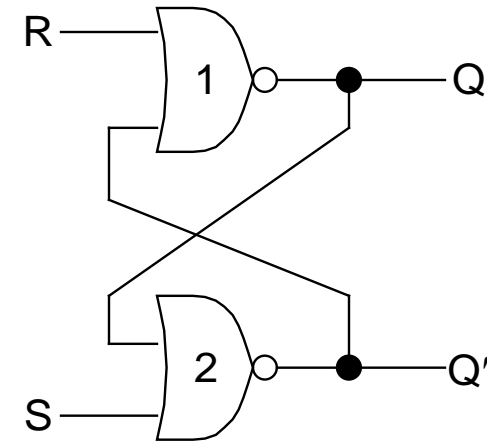
Q and Q': *Complementary o/ps*

* ***Q = 1 and Q' = 0*: Set state of the Latch**

* ***Q = 0 and Q' = 1*: Reset state of the Latch**

* ***Recall: For NOR gates:***

- ***Any (or both) i/p 1 will make the o/p 0***
- ***For o/p to be 1, both i/ps must be 0***



Operation:

* Initially, let $\mathbf{S} = \mathbf{R} = \mathbf{0}$, and Q and Q' in some state (*either 0 and 1 or 1 and 0*: the exact state is *immaterial*) – call these Q_n and Q'_n

* Now make $\mathbf{S} = \mathbf{1}$ (*keeping $R = 0$*)

\Rightarrow Immediately $\mathbf{Q}' = \mathbf{0}$, and it is fed back to NOR1, which already has one of its i/p (R) = 0

$\Rightarrow \mathbf{Q} = \mathbf{1}$ (and $\mathbf{Q}' = \mathbf{0}$)

\Rightarrow **Set** state of the latch

	S	R	Q	Q'
	0	0	Q_n	Q'_n
\downarrow	1	0	1	0
	0	0	1	0

* Changing S now to 0 will make *no difference* in the o/ps \Rightarrow *memory action*

* Now, start with $S = R = 0$, and some Q_n and Q'_n ,
and make $R = 1$ (*keeping $S = 0$*)

\Rightarrow Immediately $Q = 0$, and it is

fed back to NOR2, which

already has one of its i/p (S) = 0

$\Rightarrow Q' = 1$ (and $Q = 0$)

\Rightarrow **Reset** state of the latch

	S	R	Q	Q'
	0	0	Q_n	Q'_n
\downarrow	0	1	0	1
	0	0	0	1

* Changing R now to 0 will make *no difference*
in the o/ps \Rightarrow again a *memory action*

* Thus, with $S = R = 0$, Q may be *1 (or 0)*, and
corresponding Q' may be *0 (or 1)*, *depending*
on what was the last i/p \Rightarrow **memory** \Rightarrow **Latch**

* Now, make $S = R = 1$:

⇒ Irrespective of the previous i/p, *both Q and Q' will be 0*

⇒ Under this condition, Q and Q' are *not complementary* of each other

⇒ Causes serious ambiguity and anomaly

* *Now, if both S and R return to 0, Q and Q' will be completely undefined, since it will depend on which signal between Q and Q' propagated faster and reached the i/p to the NOR gate first, thus causing a change of state of that particular NOR gate before the other one!*

* Thus, for *NOR Latch*, $S = R = 1$ is a **NOT ALLOWED**
(or **NOT PERMITTED**) i/p combination

* *Complete Truth Table of a NOR Latch:*

S	R	Q	Q'	Remarks
0	0	Q_n	Q'_n	<i>No Change</i> , remains in <i>present state</i> (also known as HOLD state)
1	0	1	0	Set
0	1	0	1	Reset
1	1	0	0	<i>Invalid condition</i> (NOT PERMITTED) (also known as FORBIDDEN)

NAND Latch:

* **Two cross-coupled NAND gates**

* *Note the difference with NOR*

Latch : Here, S drives the Q o/p

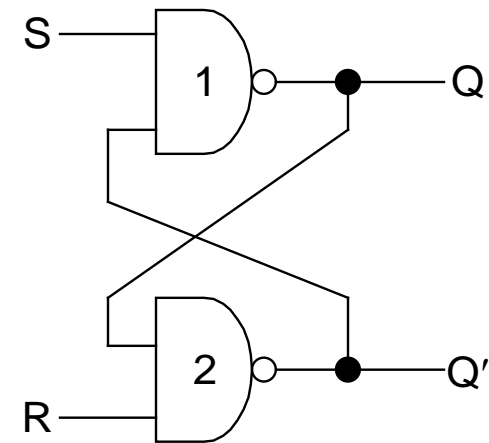
and R drives the Q' o/p

\Rightarrow *Exactly opposite to that of*

NOR Latch

* **Recall: For NAND gates:**

- *Any (or both) i/p 0 will make the o/p 1*
- *For o/p to be 0, both i/ps must be 1*



Operation:

- * Normally, both *S and R are kept high (1)*, and o/ps are at some Q_n and Q'_n
- * Now, if *S is made 0 (with R at 1)*, *Q becomes 1*, fed back to NAND2, and with R and Q both 1, $Q' = 0$
 \Rightarrow *Consistent*, and the latch is now **Set**
- * *Now, S can go back to 1 without Q and Q' changing state*
- * If *R is made 0 (with S at 1)*, *Q' becomes 1*, fed back to NAND1, and with S and Q' both 1, $Q = 0$
 \Rightarrow Again *consistent* \Rightarrow **Reset** state of the latch

* *Now, R can go back to 1 without any change of state*

* Next, make $S = R = 0 \Rightarrow Q = Q' = 1$

\Rightarrow *Same inconsistency as seen in NOR latch*

\Rightarrow Thus, this i/p combination is **NOT ALLOWED**

* *Complete Truth Table of a NAND Latch:*

S	R	Q	Q'	Remarks
0	0	1	1	FORBIDDEN (NOT ALLOWED)
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_n	Q'_n	HOLD (No Change)

* **NOR Latch:**

* ***Gets activated by high signals***

* ***$S = 1$ and $R = 0$ sets it ($Q = 1$ and $Q' = 0$)***

* ***$S = 0$ and $R = 1$ resets it ($Q = 0$ and $Q' = 1$)***

* **$S = R = 1$ is NOT ALLOWED**

* **Called SR Latch**

* **NAND Latch:**

* ***Gets activated by low signals***

* ***$S = 0$ and $R = 1$ sets it ($Q = 1$ and $Q' = 0$)***

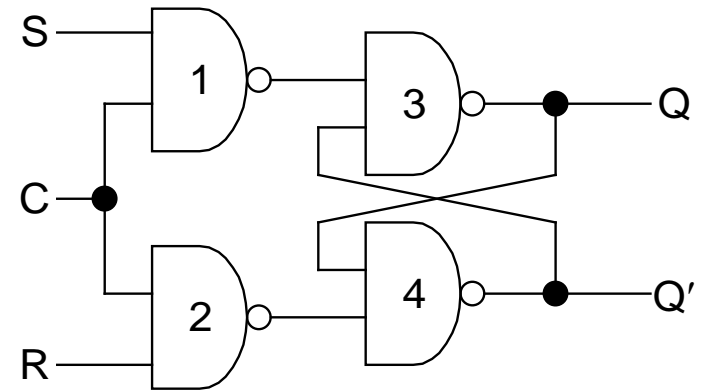
* ***$S = 1$ and $R = 0$ resets it ($Q = 0$ and $Q' = 1$)***

* ***$S = R = 0$ is NOT ALLOWED***

* **Called S'R' Latch**

Additional Control Input:

- * Determines when the state of the latch can be *changed*
- * **Control Signal C** acts as the *Enable signal*
- * When **C = 0**, o/ps of 1 and 2 stay at 1, which makes the o/ps of 3 and 4 stay at their *present states*, irrespective of the S and R i/ps \Rightarrow **Hold State**
- * When **C = 1**, then i/ps S and R can *affect* the o/p
- * If **S = R = 0**, o/ps of both 1 and 2 are 1, and Q and Q' maintain their states (**Hold** or *No Change* state)



* Now, when **S = 1 and R = 0**:

- O/p of 1 = i/p of 3 = 0 \Rightarrow o/p of 3 = **1 (Q)**
 - O/p of 2 = i/p of 4 = 1 \Rightarrow o/p of 4 = **0 (Q')**
- \Rightarrow **Set state**

- *Note that even if C goes to 0 now, this state will be maintained*

* Now, consider **S = 0 and R = 1**:

- Following the above logic, **Q = 0 and Q' = 1**
- \Rightarrow **Reset state**

* If **S = R = 1**, o/ps of both 1 and 2 = **0**, and o/ps of both 3 and 4 = **1 (NOT ALLOWED)**

- * *The control signal (C) can be clock as well*
- * This circuit is known as **Clocked or Gated SR Latch**
- * *Complete Truth Table of a Gated (Clocked) SR Latch:*

C	S	R	Q_{n+1}	Remarks
0	X	X	Q_n	Hold (No Change)
1	0	0	Q_n	Hold (No Change)
1	1	0	$Q = 1$	Set
1	0	1	$Q = 0$	Reset
1	1	1	?	NOT ALLOWED (INDETERMINATE)

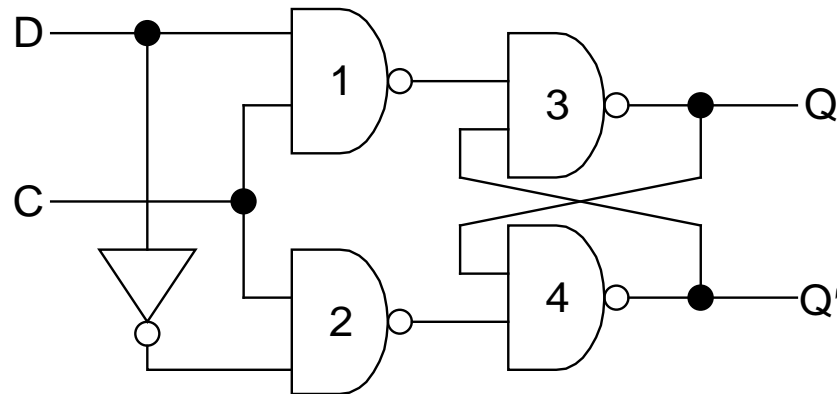
- * *Note: A clocked (or gated) NAND latch is of SR type*

D Latch:

- * Avoids the possibility of S and R ever becoming equal to 1 simultaneously, by putting an inverter (NOT gate) between the i/ps of 1 and 2

⇒ *Eliminates the indeterminate state*

- * *Has 2 i/ps: D (Data) and C (Control)*



- * If $C = 0$, o/ps of both 1 and 2 = 1, and the o/p *cannot change state* regardless of the value of D
 \Rightarrow **Hold (No Change)** State
- * If $C = 1$, then the circuit *samples D*
 - If $D = 1$, o/p of 1 = 0 $\Rightarrow Q = 1$, fed back to 4
 \Rightarrow I/ps of 2 = 1 and 0 \Rightarrow o/p of 2 = 1
 \Rightarrow I/ps of 4 = 1 and 1 \Rightarrow o/p of 4 = 0 (Q')
 $\Rightarrow Q = 1$ and $Q' = 0 \Rightarrow$ **Consistent (Set state)**
 - Similarly, if $D = 0$, $Q = 0$ and $Q' = 1$
 \Rightarrow **Reset state** (verify this condition)

**** Truth Table of D Latch:***

C	D	Remarks
0	X	Hold (<i>No Change</i>): Q = last value of D
1	0	$Q = 0$ (Reset)
1	1	$Q = 1$ (Set)

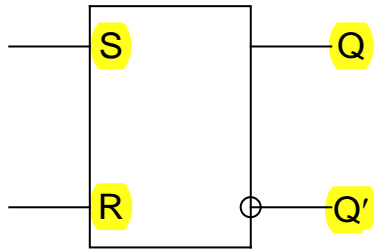
**** Name originates from its ability to hold data in its internal storage***

**** Used as a temporary storage element***

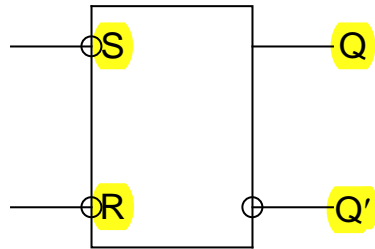
**** When C goes high, i/p D is sampled and sent to o/p***

**** Known as Transparent Latch, \therefore the o/p follows D so long as C = 1***

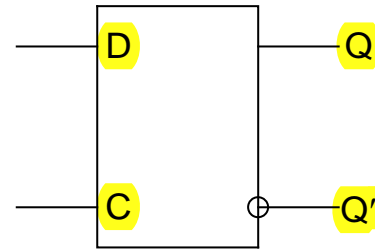
Symbols:



SR NOR Latch
Activated by High S/R



S'R' NAND Latch
Activated by Low S/R



D Latch

Flip-Flop (F /F):

- * *State of a latch can be changed by a change in the control i/p*
- * This momentary change is called a **trigger**, and the *transition it causes is said to trigger the latch*
- * If the control i/p of the D Latch is replaced by **pulses**, then each time the **pulse** goes **high**, the **state** of the D Latch may **change**, depending on the i/p
 \Rightarrow ***D Latch turns into a D F /F***
- * **As long as this control pulse remains high, any change in D would cause a corresponding change in Q**



Race Around Condition:

- * A **serious problem** in sequential circuits, which have feedback path
- * If clock pulse goes **high**, then o/ps of the storage elements **change**, and these are **fed back** to the i/p of the combinational circuit contained within
- * ***If these i/ps change while the clock signal is still high, then it may cause further change in the o/p***
- * Result is an **unpredictable situation**, **\therefore the o/p would keep on changing as long as the clock remains high**

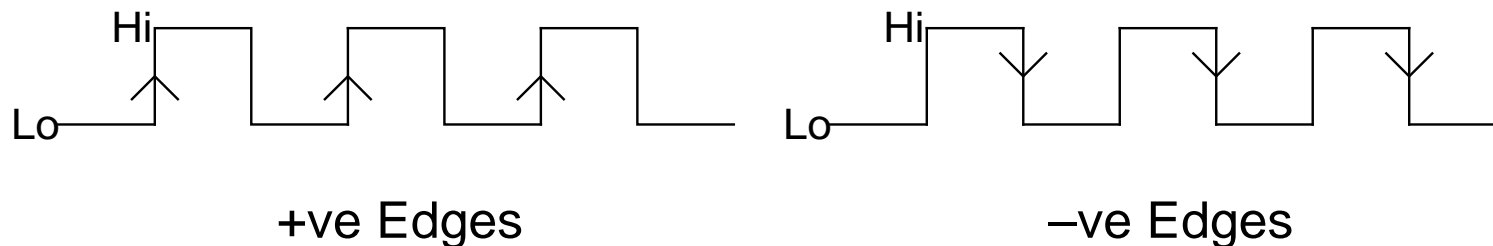
- * Known as the **Race Around Condition**
- * *Thus, the o/p of a latch cannot be applied directly or through some other combinational circuits to the i/p of another latch, if all these latches are triggered by a common clock*
- * **F/Fs, on the other hand, are designed in such a way that they operate properly even when used in a sequential circuit chain that uses a common clock**

* **Note:** Latches respond to *clock levels*

⇒ *If clock remains at high level, then changes in S/R or D would keep causing a change in D*

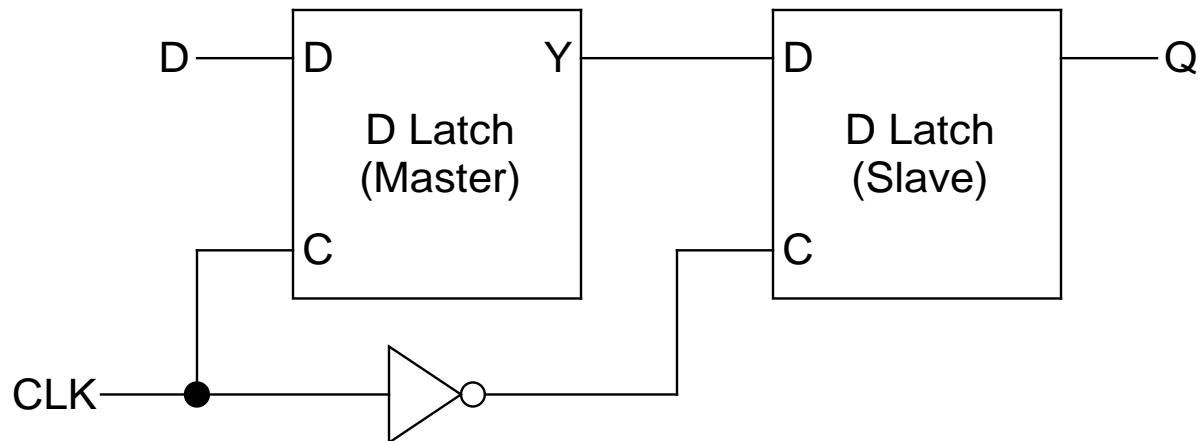
* **Remedy:** Trigger a F/F only during *clock transition*

* A clock pulse has *two transitions*: **Lo to Hi** (*0 to 1*) and **Hi to Lo** (*1 to 0*), which are known as **clock edges**



- * Thus, a Latch can be modified to a F/F *2 ways*:
- Employ *2 latches*, make the first one operate at **Hi** level of clock, and the second one operate at **Lo** level of clock (by the use of an *inverter*)
 - Produce a F/F that triggers only during a *signal transition* (**0 to 1** or **1 to 0**), and are disabled otherwise \Rightarrow known as **edge-triggered**

Edge Triggered D F / F (Master-Slave Architecture):

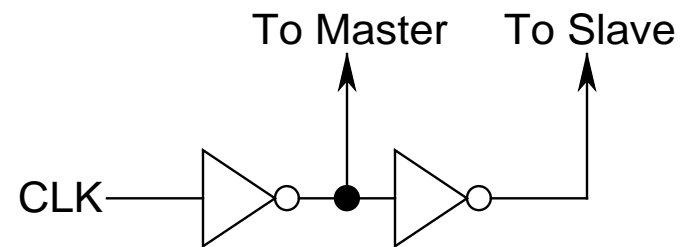


- * **Note:** Master D Latch is + ve level triggered, whereas Slave D Latch is – ve level triggered
- * When $CLK = 0$, $Q = Y$, and Master is disabled
- * When $CLK = 1$, Master is enabled, and Y takes on the value of D, while Slave remains disabled

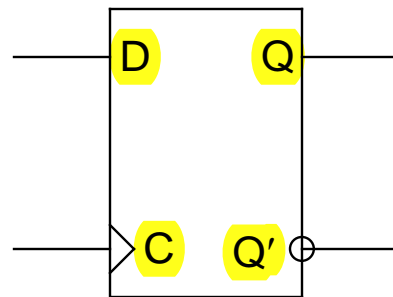
- * *Any change in i/p can change Y, but can't change Q*
- * Thus, *Q can change again only when the CLK transits from 1 to 0*

* Thus, this circuit is a **–ve Edge-Triggered D F/F**

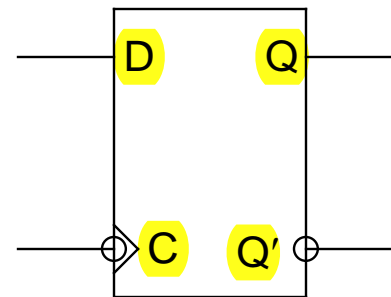
- * Can be easily converted to
+ve *Edge-Triggered* by \Rightarrow



* *Symbols:*



+ve Edge Triggered



–ve Edge Triggered

Three Important Time Definitions in Digital Design:

* **Set-Up Time:**

- *Before the CLK arrives, D must be maintained at a constant value for a specific time*

* **Hold Time:**

- *After the CLK arrives, D must be held constant for a specific time*

* **CLK-to-Q Delay:**

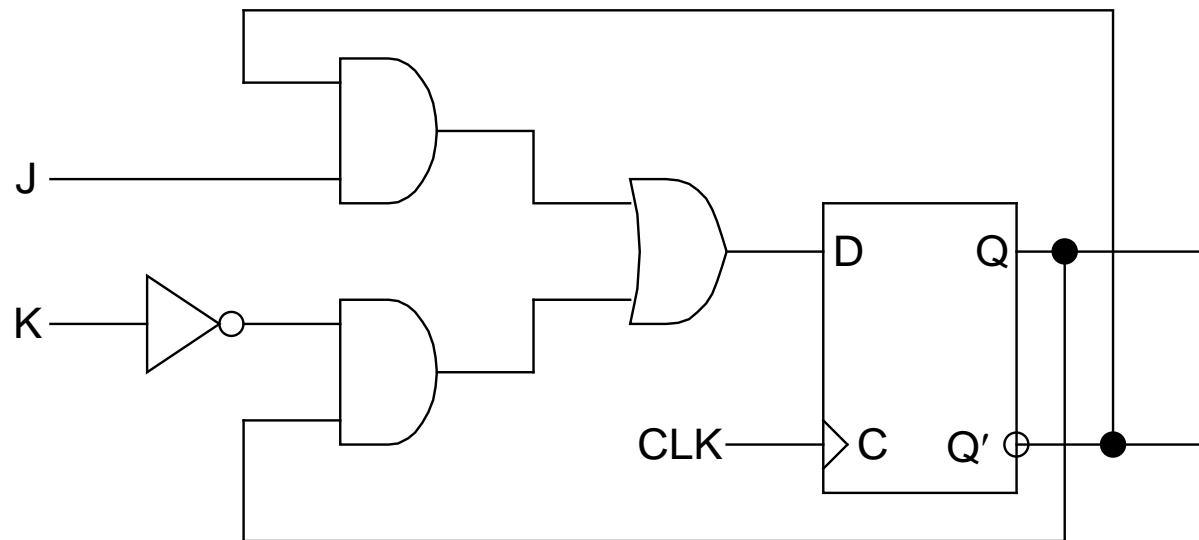
- *Propagation delay between the time the CLK arrives and the time that the o/p settles down (known as the o/p being valid)*

Variants of $D F/F$:

* **JK F/F**

* **TF F/F**

JK F/F:



- * **3 States:**
 - **Set to 1**
 - **Reset to 0**
 - **Complement the o/p (*Toggle*)**
- * **$J = K = 0 \Rightarrow$ *o/p holds on to the present state***
- * **$J = 1$ sets Q to 1**
- * **$K = 1$ resets Q to 0**
- * **$J = K = 1 \Rightarrow$ *o/p toggles***
- * **Note: $D = JQ' + K'Q$**
- * **When $J = 1$ and $K = 0$: $D = Q' + Q = 1$**
 \Rightarrow Next + ve CLK edge sets Q to 1

* When $J = 0$ and $K = 1$: $D = 0$

\Rightarrow Next + ve CLK edge resets Q to 0

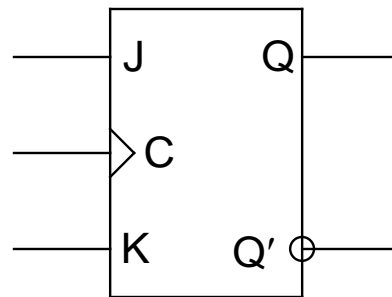
* When $J = K = 1$: $D = Q'$

\Rightarrow Next + ve CLK edge complements the o/p

* When $J = K = 0$: $D = Q$

\Rightarrow Next + ve CLK edge leaves the o/p unchanged

* *Symbol:*



Characteristic (or State Transition) Table:

J	K	Q(t + 1)	Remarks
0	0	Q(t)	Hold (<i>No Change</i>)
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Toggle (<i>Complement</i>)

* **Compare with D F/F, which does not have any**
Hold state: *O/p independent of present state and*
depends only on the value of D

- **D = 0: $Q(t + 1) = 0$ (Reset)**
- **D = 1: $Q(t + 1) = 1$ (Set)**

T F /F:

* **Complementing (Toggle) F/F**

* *Obtained from JK F /F by tying up J and K together, and using that as the toggle (T) i/p*

* When **T = 0** (**J = K = 0**):

• *O/p remains in Hold state:*

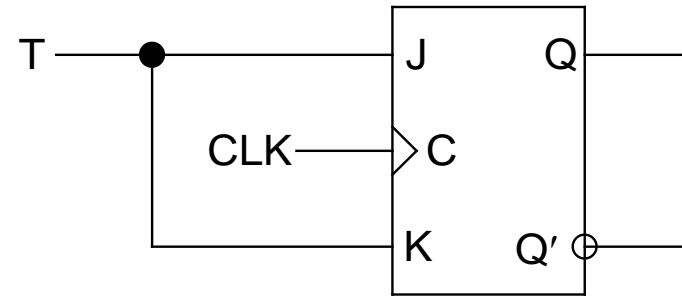
$$\Rightarrow Q(t + 1) = Q(t)$$

* When **T = 1** (**J = K = 1**):

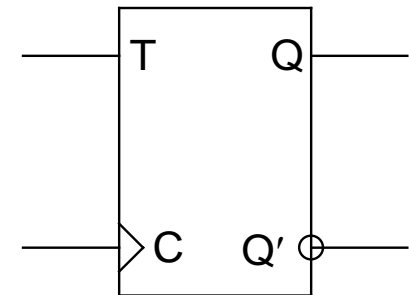
• *+ve CLK edge complements the o/p:*

$$\Rightarrow Q(t + 1) = Q'(t)$$

* *Extensively used for Binary Counters*



Conversion of a JK F/F to a T F/F



Symbol

* *Can also be constructed from a D F/F using an XOR gate*

* $D = T \oplus Q = TQ' + T'Q$

* When $T = 0$: $D = Q$

\Rightarrow No Change

* When $T = 1$: $D = Q'$

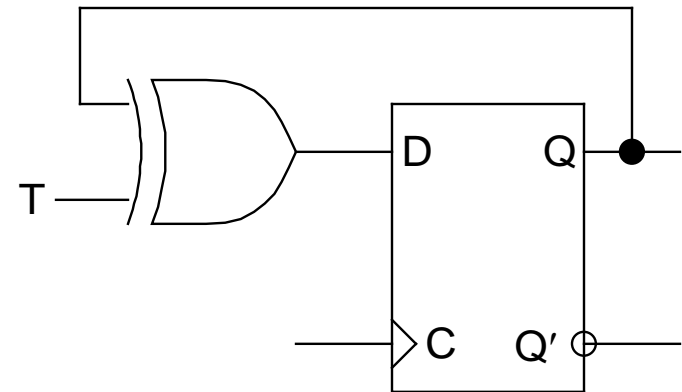
\Rightarrow Complement (*Toggle*)

Characteristic Equations:

* **D F/F:** $Q(t + 1) = D$

* **JK F/F:** $Q(t + 1) = JQ' + K'Q$

* **T F/F:** $Q(t + 1) = T \oplus Q = TQ' + T'Q$

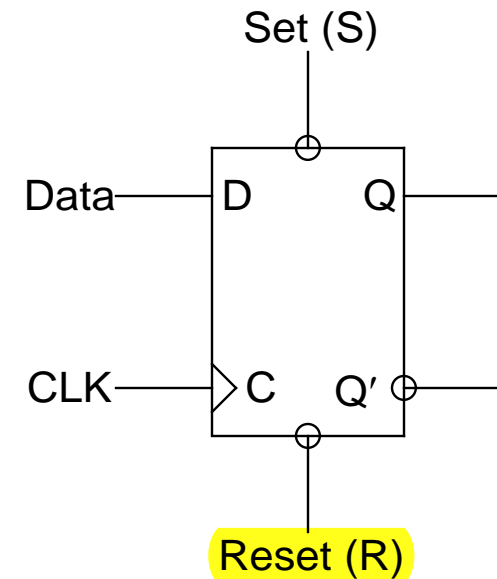


Conversion of a D F/F to a T F/F

Asynchronous Inputs:

- * **Overrides CLK signal**
- * *Forces the F /F to a particular state, independent of CLK*
- * I/p that *sets* the F/F to *1* is called a **Preset** or **Direct Set**
- * I/p that *resets* (or *clears*) the F/F to *0* is called a **Clear** or **Direct Reset**
- * When power is first turned on, states of the F/Fs are *unknown*
- * Thus, useful to bring all F/Fs in the system to a *known starting point* prior to the clocked operation

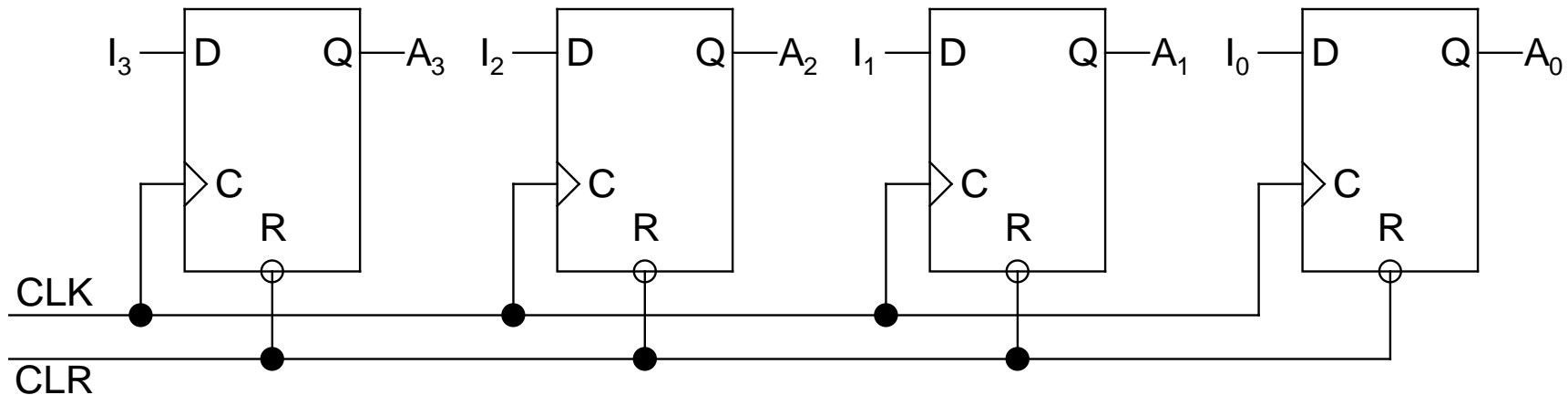
- * *Both Set (S) and Reset (R) are Active Low signals*
- * **S = 0 sets Q to 1**
- * **R = 0 resets (clears) Q to 0**
- * Normally, S and R are kept **high**, and the D F/F follows the **normal** Truth Table
- * **R and S should not be made low simultaneously**



Registers:

- * **Group of F/Fs**
- * *n-bit Register can store n bits of information*
- * Can also contain *gates* to perform certain data processing tasks
- * **Counter** is an example of a register that goes through a *predetermined sequence of states*

4-Bit Register:



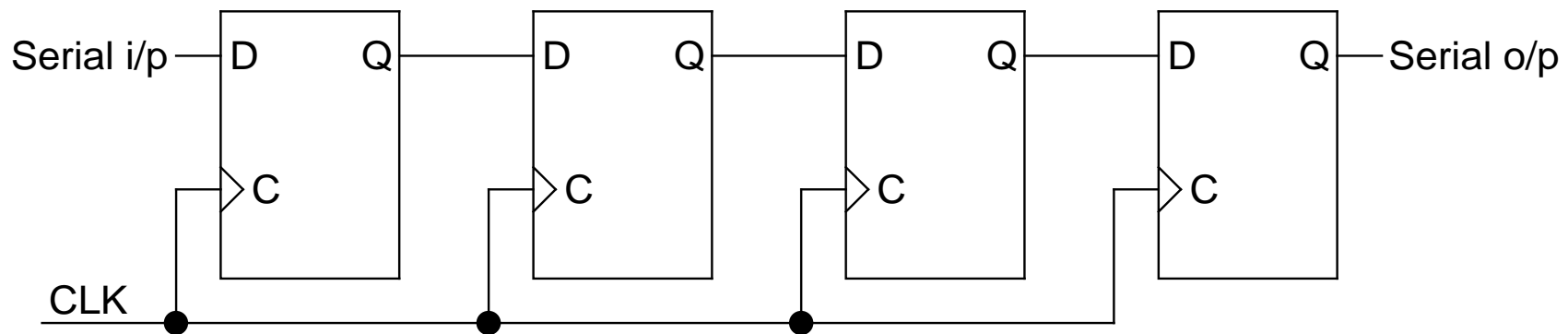
- * +ve **edge** triggered
- * Binary data at I_3-I_0 gets transferred into the 4-bit register at every +ve going clock edge
- * The 4 o/ps can be sampled *at any time* to obtain the binary information stored in the registers

- * When **CLR = 0**, all F/Fs are *reset* (or *cleared*) *asynchronously* (*nothing to do with clock*)
- * *Prior to normal clocked operation, generally all registers are cleared by pulling CLR line low*

Shift Register:

- * Capable of shifting the binary information contained within it either *right* or *left*
- * *Consists of a chain of F /Fs in cascade, with the o/p of one F /F connected to the i/p of the next F /F*
- * All F/Fs receive *common* CLK pulses, which activate the *shift* from one register to the next

4-Bit Shift Register:



- * Each clock pulse *shifts* the contents of the registers one bit position to the *right*
- * *Serial i/p* determines what goes into the leftmost *F/F* during the shift
- * *Serial o/p* is taken from the *rightmost F/F*

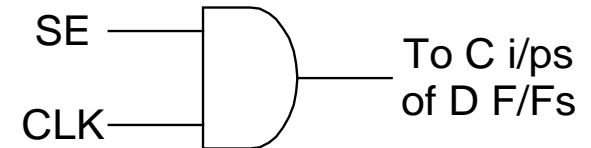
Controlled Shifting:

- * Shift would occur only with certain clock pulses, but not all

⇒ Prevents clock from reaching the F/Fs when shifting is not needed

- * **SE: Shift Enable**

- * If $SE = 1$, CLK gets transferred to C i/ps of F/Fs , and normal shifting takes place



- * If $SE = 0$, CLK is prevented from reaching the C i/ps of the F/Fs

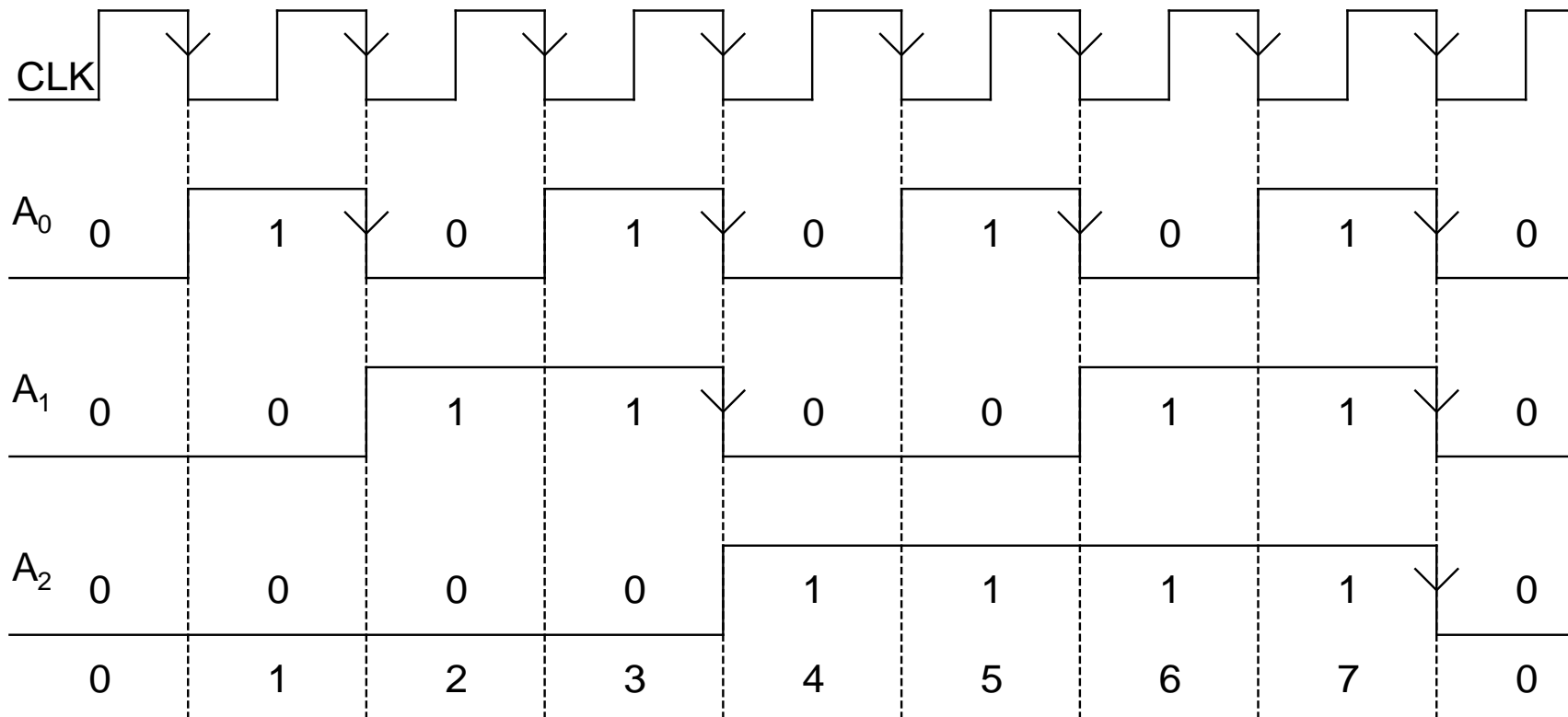
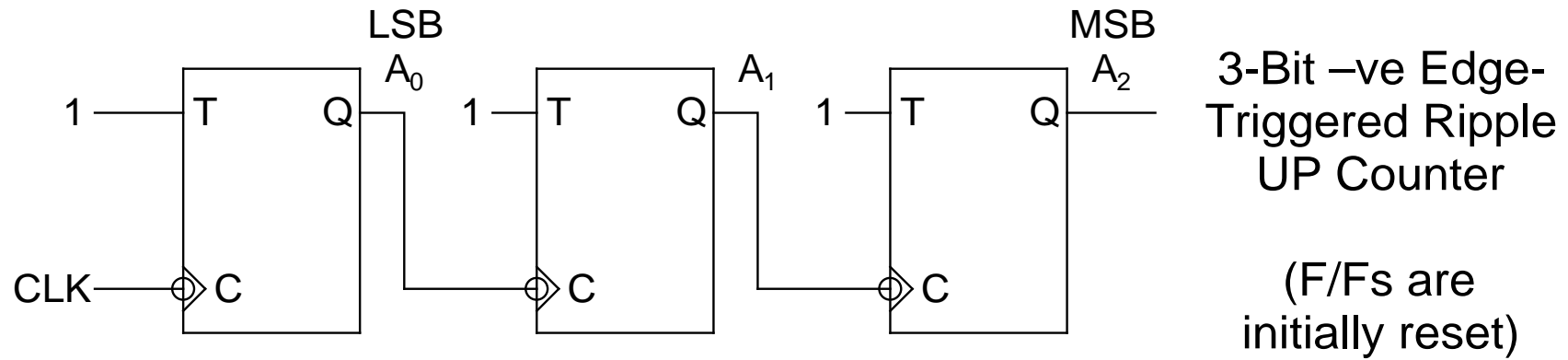
Counters:

- * **Registers going through a prescribed sequence of states upon application of i/p pulses**
- * *I/p pulses may either be clock pulses or originate from some external sources and occur either at a fixed interval of time or randomly*
- * **Sequency of states may either be a sequence of binary numbers or any other sequence of states**
- * *A counter following binary number sequence is known as a Binary Counter*

- * ***n -bit Binary Counter has n F/Fs and can count in binary from 0 to $(2^n - 1)$***
- * ***Two types: Ripple and Synchronous***
- * ***In Ripple Counters, F/F o/p transitions serve as a source of triggering for other F/Fs***
 - \Rightarrow The C i/ps of all the F/Fs (barring the first one) is not triggered by the common clock but by the o/ps of other F/Fs***
- * ***In Synchronous Counters, the C i/ps of all the F/Fs are connected to the common clock***

Binary Ripple Counter:

- * Consists of a series connection of T F/Fs, with o/p of each F/F connected to the C i/p of the next higher order F/F**
- * *The F /F handling the LSB receives the incoming clock pulses***
- * *Recall: T F/Fs can be replaced by JK F/Fs by tying $J = K = 1$ (Toggle mode)***
- * D F/F can also be used by connecting Q' back to D (*Toggle operation*)**



3-Bit Ripple UP Counter (–ve Edge-Triggered):

- * Counts from 0 to 7, and then goes back to 0
- * *The count increments by 1 for each clock pulse period*
- * Each – ve edge of CLK, A_0 , and A_1 complements A_0 , A_1 , and A_2 , respectively
- * *The F /Fs change their states one at a time in succession, and the signal propagates through the counter in a ripple fashion from one stage to the next*
- * *Exercise:* Show that if the F/Fs are *set* initially and they are *+ve edge-triggered*, then it becomes a **DOWN Counter**

BCD Ripple Counter:

*** BCD Number System:**

- *Binary Coded Decimal (BCD)*
- *Uses 4 bits to count decimal numbers from 0 to 9 (0000 to 1001) and then goes back to 0*
- **The rest of the numbers (10 to 15 – 1010 to 1111) are DON'T CARE (X) in K-map, and do not appear in Counters**
- *Apparent waste, however, earlier computers used this number system*

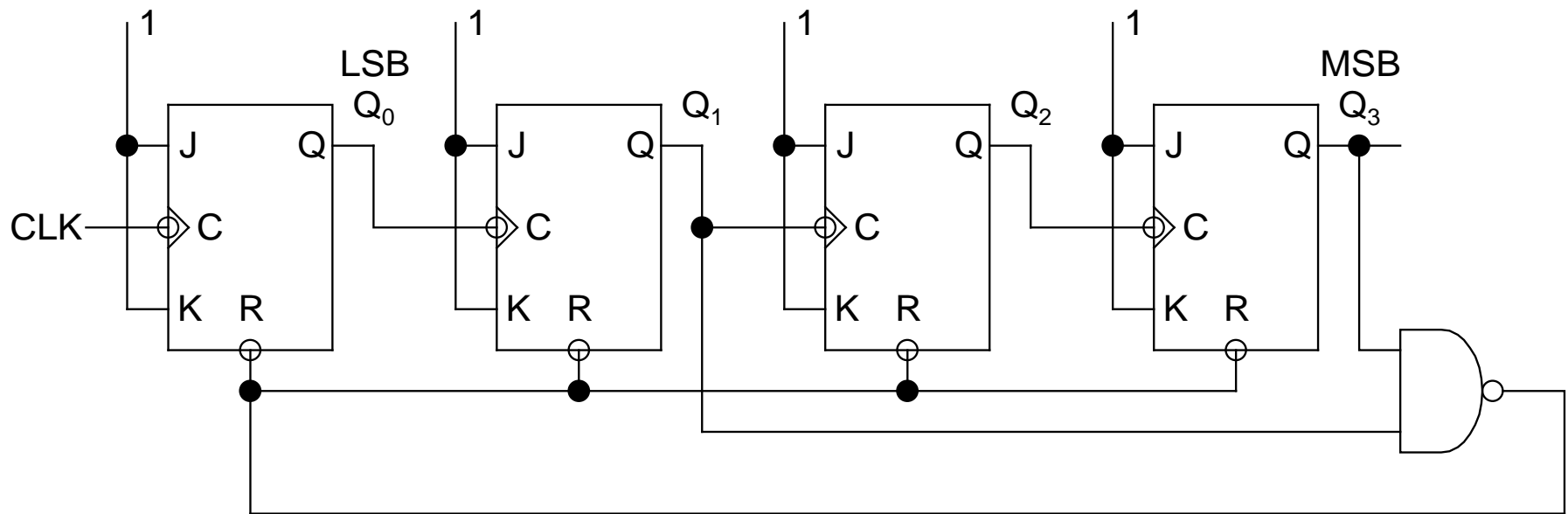
*** *Total 10 states represented by 4 bits \Rightarrow need 4 F/Fs***

Count Sequence:

MSB			LSB
Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

* **Note the automatic next state after 1001 is 1010, which should be able to asynchronously reset the F/Fs to 0000**

⇒ ***Decode Q_3 and Q_1 , such that when both of them go high, all the F /Fs are reset***



4-Bit BCD Ripple Counter

- * *During switch on, any arbitray state may appear at the o/p*
- * **For cases where Q_1 and Q_3 are both 1 (10, 11, 14, and 15), the o/p of the NAND gate will be zero, which would reset the F/Fs, and normal counting sequence (from 0 to 9) will start**
- * *Note that clearing of the F /Fs is done asynchronously*
⇒ **Example of asynchronous operation**
- * *Arbitray count sequence generation is extremely difficult (if not impossible) to design using Ripple Counters, and we need Synchronous Counters*

Synchronous Counter:

- * **Clock pulses are applied to i/ps of all F/Fs**
- * ***Whether the o/p of the F /F would change or not will depend on the state of the i/ps***
- * ***To design Synchronous Counters, necessary to understand F/F Excitation Table***

Excitation Table:

- * **Gives the i/ps required for a F/F to change the o/p state as desired**
- * ***Four possible transitions: 0 to 0, 0 to 1, 1 to 0, and 1 to 1***
- * **These tables are obtained from the Truth Table for the particular F/F**

JK F/F:

Truth Table of JK F/F:

J	K	Q_{n+1}	Remarks	* For 0 to 1, either <i>Set</i> or <i>Toggle</i>
0	0	Q_n	<i>Hold</i>	\Rightarrow J = 1 and K = 0 or
0	1	0	<i>Reset</i>	J = 1 and K = 1
1	0	1	<i>Set</i>	\Rightarrow J = 1 and K = X (<i>DON'T</i>
1	1	Q'_n	<i>Toggle</i>	<i>CARE</i>)

* For **1 to 0**, either *Reset* or *Toggle*

\Rightarrow **J = 0 and K = 1** or **J = 1 and K = 1**

\Rightarrow **J = X** (*DON'T CARE*) and **K = 1**

* To Hold **0**, J should not become 1 (i.e., **$J = 0$**)
and K is immaterial (i.e., $K = X$)

* To Hold **1**, K should not become 1 (i.e., **$K = 0$**)
and J is immaterial (i.e., $J = X$)

Excitation Table of JK F/F:

Q_n	Q_{n+1}	J	K
0	0	0	X
1	1	X	0
0	1	1	X
1	0	X	1

T F /F:

Truth Table of T F/F:

T	Q_{n+1}	Remarks
----------	-----------------------------	----------------

0	Q_n	<i>Hold</i>
---	-------	-------------

1	Q'_n	<i>Toggle</i>
---	--------	---------------

Excitation Table of T F/F:

Q_n	Q_{n+1}	T
-------------------------	-----------------------------	----------

0	0	0
---	---	---

1	1	0
---	---	---

0	1	1
---	---	---

1	0	1
---	---	---

3-Bit Synchronous UP Counter Using JK F /Fs:

*** General Algorithm:**

- *Write the Truth Table and the Excitation Table*
- *Prepare the State Transition Table (STT)*
- *Using STT and Excitation Table, prepare the list of required i/ps to the F /Fs*
- *Use K-map for minimizing the Boolean function for each such i/p*
- *Use required hardware to implement the Boolean functions*

*** Q_2 : MSB, Q_0 : LSB**

State Transition Table & Required F/F Inputs

Present State			Next State			Required F/F Inputs					
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	0	1	0
	1	X	X	X	X

$$J_2 = Q_1 Q_0$$

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	X	X	X
	1	0	0	1	0

$$K_2 = Q_1 Q_0$$

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	0	1	X
	1	0	1	X	X

$$J_1 = Q_0$$

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	X	X	1
	1	X	X	1	0

$$K_1 = Q_0$$

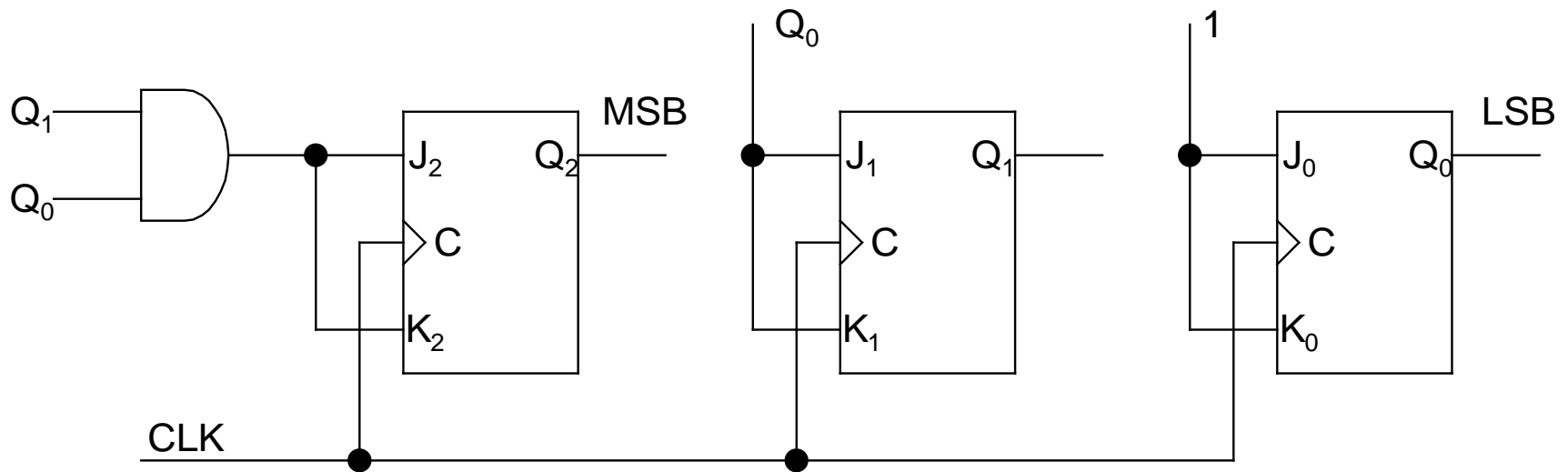
$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	1	X	X
	1	1	X	X	1

$$J_0 = 1$$

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	X	1	1
	1	X	1	1	X

$$K_0 = 1$$

K-Maps for Boolean Function Minimization



Implementation with +ve Edge Triggered JK F/Fs
(Note: Q' outputs of the F/Fs unused)

Exercise: If we want to count down from 15 to 0, and then back to 15, determine the number of JK F/Fs needed, and following the procedure outlined, find their necessary i/ps.

Modulo-m Counters:

- * **Divide by n Counter is known as Modulo-m Counter, with $m \leq 2^n$**
- * **Example: $n = 2, 3, 4$, etc. correspond to Modulo-4, Modulo-8, Modulo-16 Counters, etc., respectively**
- * **Note that a Modulo-m counter can also be made a Modulo-p counter, with $p < m$**
- * **Example: A Modulo-8 counter can be made to count only from 0 to 5, and thus, it becomes a Modulo-6 counter**

Arbitrary Sequence Generation:

- * Probably the **most complicated** of counter designs
- * Counts a **totally arbitrary sequence** specified by the user
- * *Note:* A counter using **n F/Fs** has **2^n** binary states, out of which, *all may not be used*
- * Generally, states that are **not used** are *not specified in the State Table*

- * ***Problem:*** During power up, if the F/F o/ps assume one of these states, then it may never come out of it!
- * ***This anomaly in operation must be taken care of***
- * **This makes the counter design little complicated and sometimes challenging**

Possibly the Most Complicated Design Example:

- * Design a **3-bit Synchronous Counter** that counts in the following sequence: **1, 0, 5, 2, 4, 1, 0, ...**
- * **Note:** States **3, 6, and 7** do not appear (*unused*)
- * ***If on start up, the counter goes to any of these states, then it will never be able to come out of it***
- * Thus, the design may have **two possible outcomes:**
 1. *If any of the states 3, 6, or 7 appear on start up, they all will go to a specific state (one of the designed states) at the next clock pulse*
 2. *They will go to different specific states*

- * ***Choose State 3 to go to State 1***

- State 6 to go to State 5***

- State 7 to go to State 4***

- during the next clock pulse***

- * **Once it goes to a specific designed state, the normal count sequence would follow**

- * ***Thus, the sequence:***

- $0 \rightarrow 5, 1 \rightarrow 0, 2 \rightarrow 4, 3 \rightarrow 1,$**

- $4 \rightarrow 1, 5 \rightarrow 2, 6 \rightarrow 5, \text{ and } 7 \rightarrow 4$**

- * **Let's implement this counter using T F/Fs**

State Transition Table & Required F/F Inputs

Present State			Next State			Required F/F Inputs		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	1	0	1	1	0	1
0	0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1	0
0	1	1	0	0	1	0	1	0
1	0	0	0	0	1	1	0	1
1	0	1	0	1	0	1	1	1
1	1	0	1	0	1	0	1	1
1	1	1	1	0	0	0	1	1

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	1	0	0
0		1	0	0	1
1		1	1	0	0

$$T_2 = Q_2 Q'_1 + Q'_2 Q'_0$$

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	0	0	1
0		0	0	1	1
1		0	1	1	1

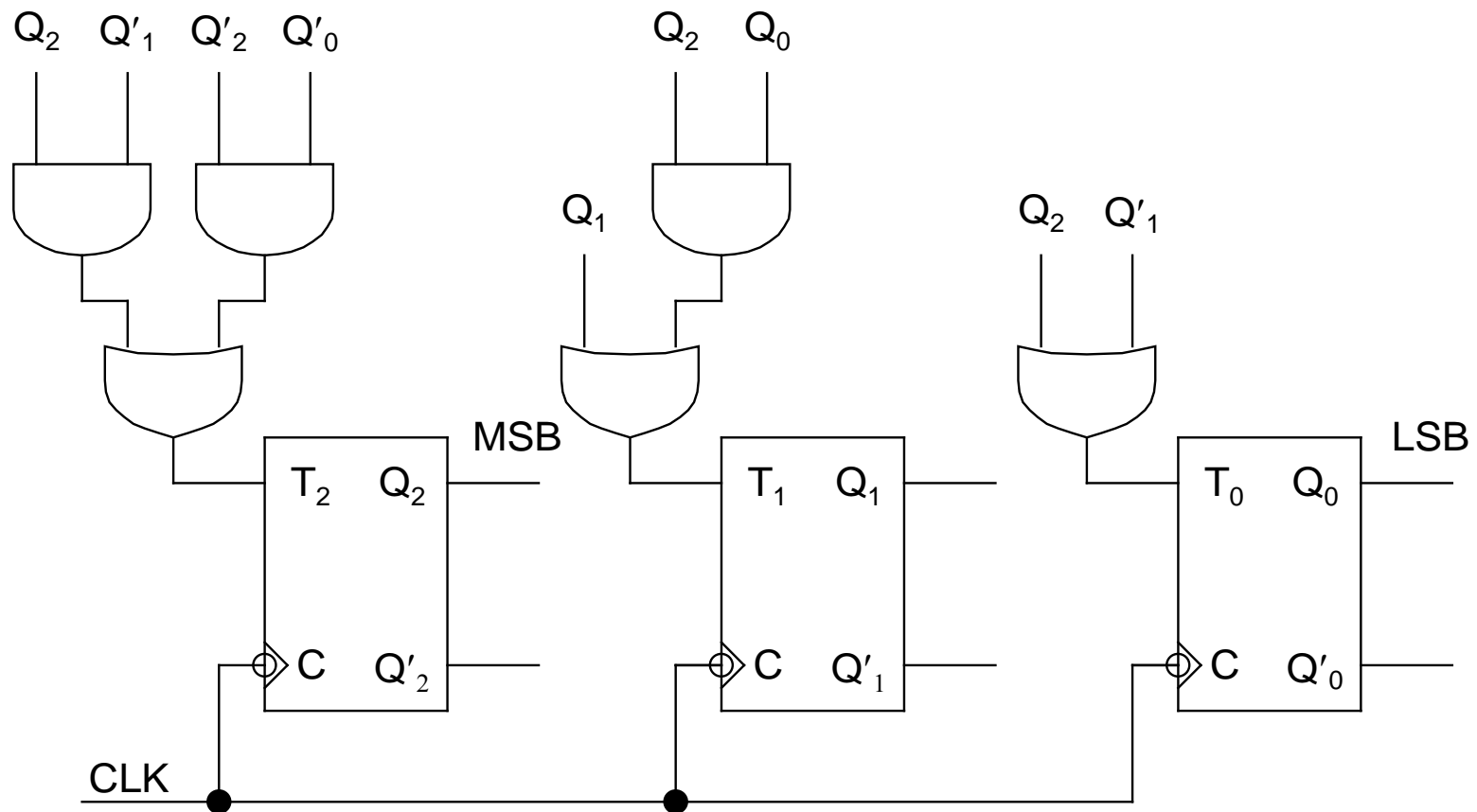
$$T_1 = Q_1 + Q_2 Q_0$$

$Q_2 \backslash Q_1 Q_0$		00	01	11	10
		0	1	1	0
0		1	1	0	0
1		1	1	1	1

$$T_0 = Q_2 + Q'_1$$

K-Maps for Boolean Function Minimization

Exercise: Implement the same design using JK F/Fs.



Implementation with –ve Edge Triggered T F/Fs

Self-Correcting Counters:

- * Counters that do not need to account for unused states*
- * If during power up, any state that is not designed for appears at the o/p, then either the next or the next two clock pulses put it to one of the states belonging to the sequence, and normal count sequence follows*
- * Known as **Self-Correcting Counters**
- * Obviously, these are very special cases, and not all counters are self-correcting