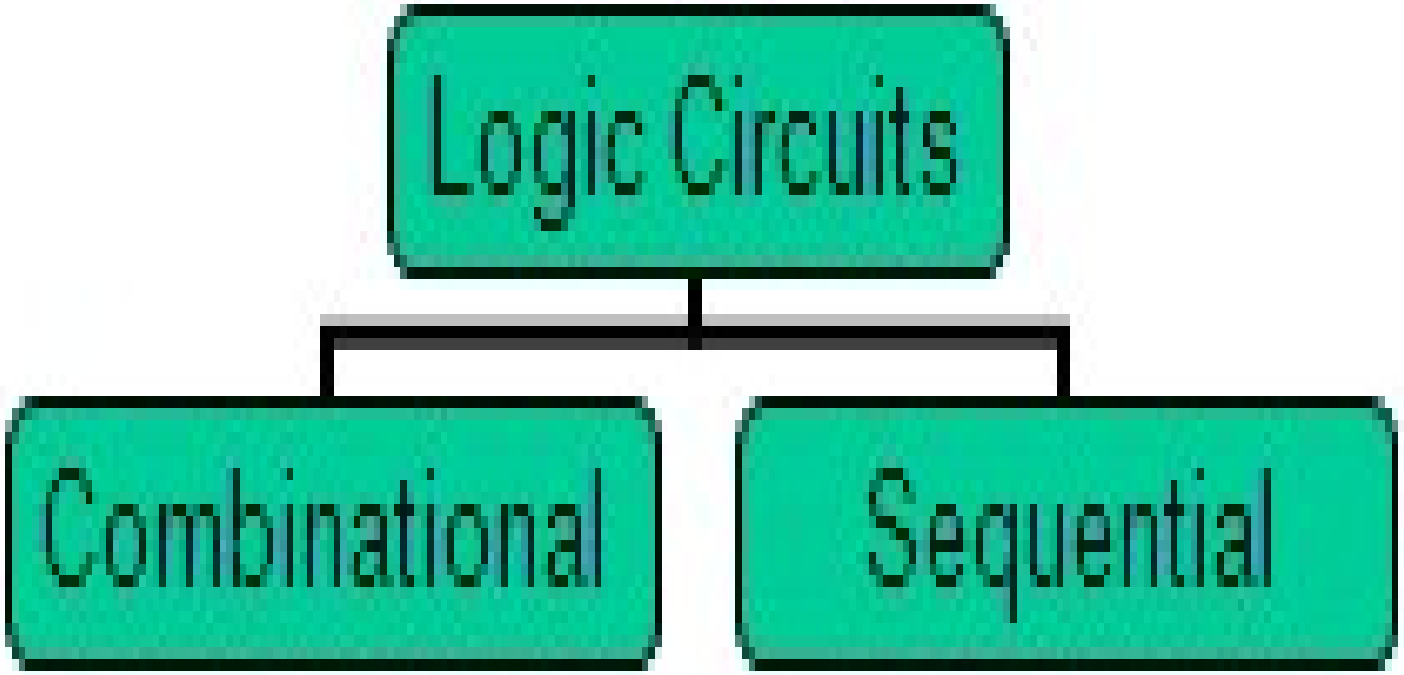
Chapter 5
Sequential Circuits
(FlipFlops, Registers and Counters)

Outline

- Flip Flops : RS, D, T, JK, Asynchronous, Synchronous and Master slave.
- Shift registers(shift left and shift right), Bidirectional shift register with parallel load.
- Counters : Synchronous and ripple counter (BCD and Binary)
- Simple arithmetic and logic circuits.

Logic Circuits



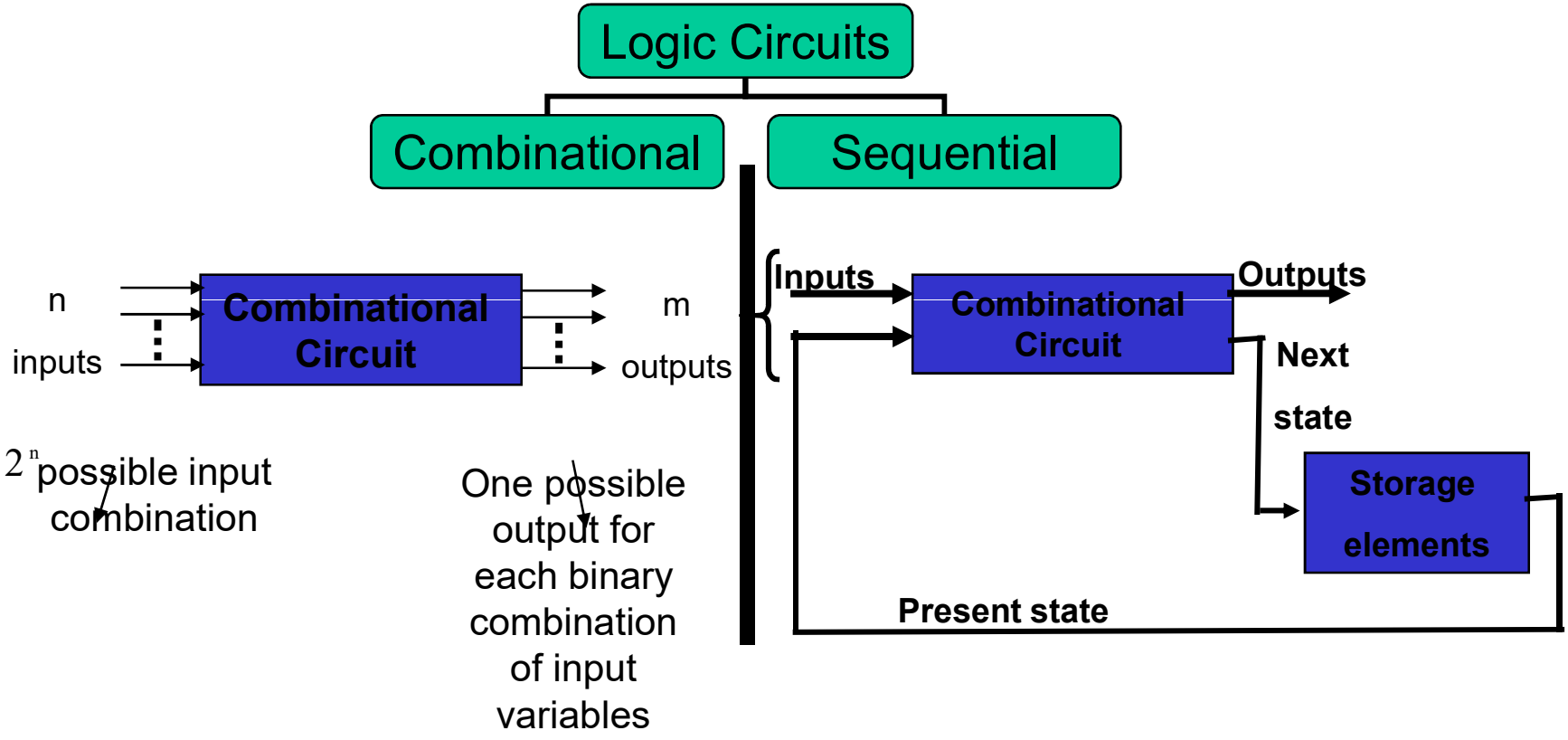
Logic Circuits

- Switching circuits may be combinational switching circuits or sequential switching circuits.
- Combinational
  - Combinational switching circuits are those whose output levels at any instant of time are dependent only on the levels present at the inputs at that time.
  - It has no memory and consists of only logic gates.
- Sequential
  - Sequential switching circuit are those whose output levels at any instant of time are dependent not only on the levels present at the inputs at that time, but also on the state of the circuit, i.e. on the prior input level condition (i.e. on its past inputs)
  - The past history is provided by feedback from the output back to the input.
  - It has memory and are made of combinational circuits and memory elements.

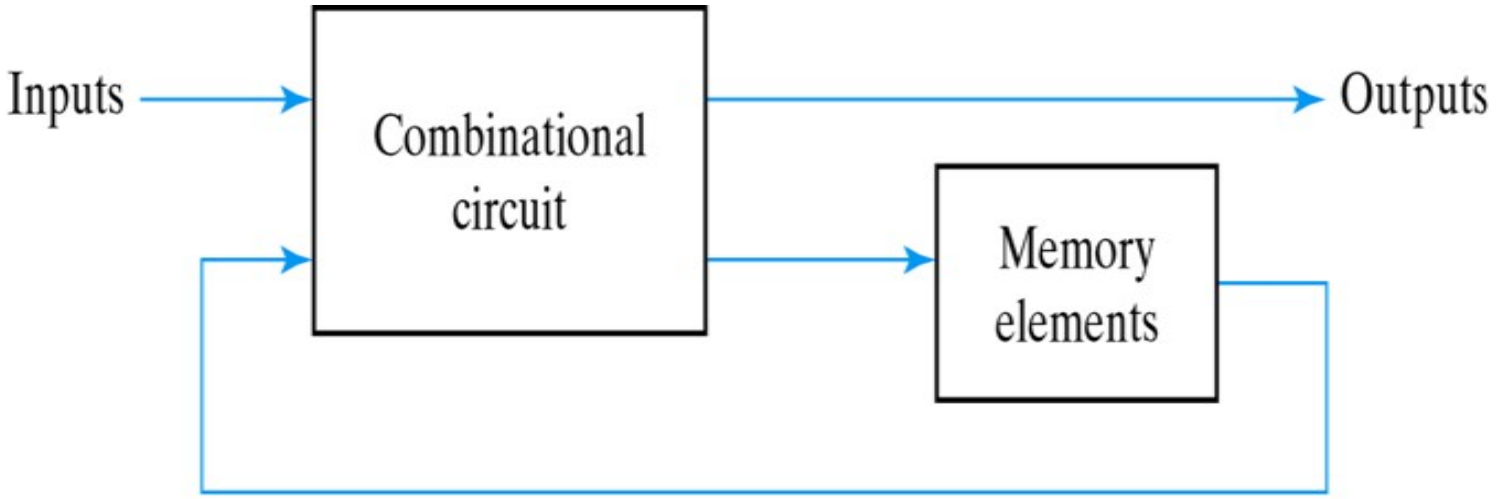
Sequential Circuits

- Examples of combinational circuits are :
  - Parallel adder
  - Subtractor
  - Encoder
  - Decoders
  - Code converters
  - Parity bit generator.
- Examples of sequential circuits are :
  - Counters
  - Shift registers
  - Serial adders
  - Sequence generators
  - Logic function generators

Logic Circuits



Block diagram of a Sequential Circuits



Block Diagram of Sequential Circuit

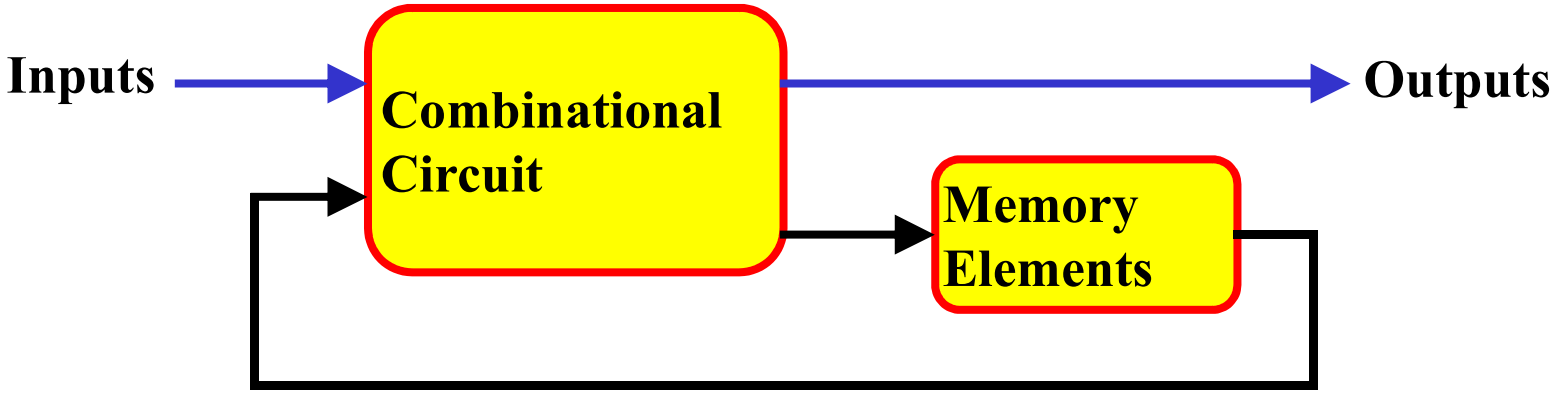
- A sequential circuit is an interconnection of combinational circuits and storage elements.
- The storage elements, called flip-flops, store binary information that indicates the state of sequential circuit at that time.

Classification of Sequential Circuits

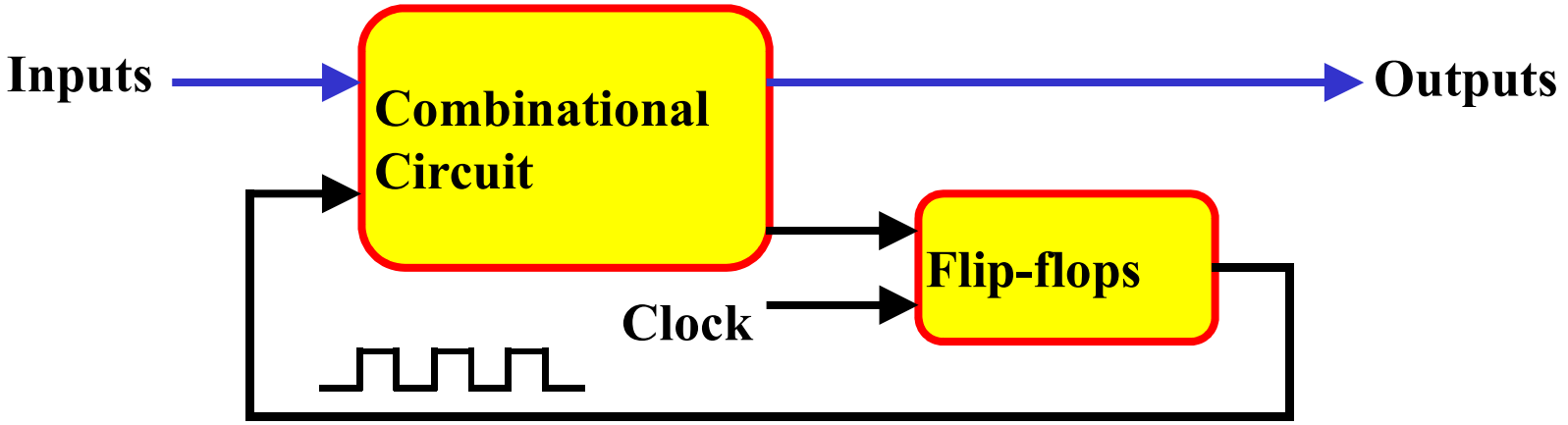
- The sequential circuit may be classified depending on the timing of their signal as
  - Synchronous sequential circuits
  - Asynchronous sequential circuits
- The sequential circuits which are controlled by a clock are called synchronous sequential circuits. These circuit will be active only when clock signal is present.
- The sequential circuit which are not controlled by a clock are called asynchronous sequential circuits, i.e. the sequential circuits in which events can take place any time the inputs are applied are called asynchronous sequential circuits.
- Periodically, recurring pulse is called a clock. It is generated by a pulse generator.

Sequential Circuits

- Asynchronous



- Synchronous



Classification of Sequential Circuits

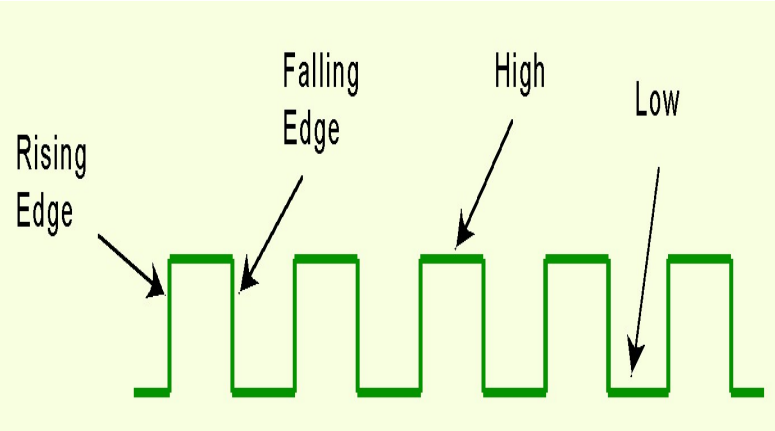
- The clock signal performs the synchronization of the state of memory elements in the case of synchronous sequential circuits.
- The output, in this case, is either stored in **latches** (memory devices) or **flip-flops (FFs).**
- The output gets synchronized with the clock's only positive edges or only the negative edges.

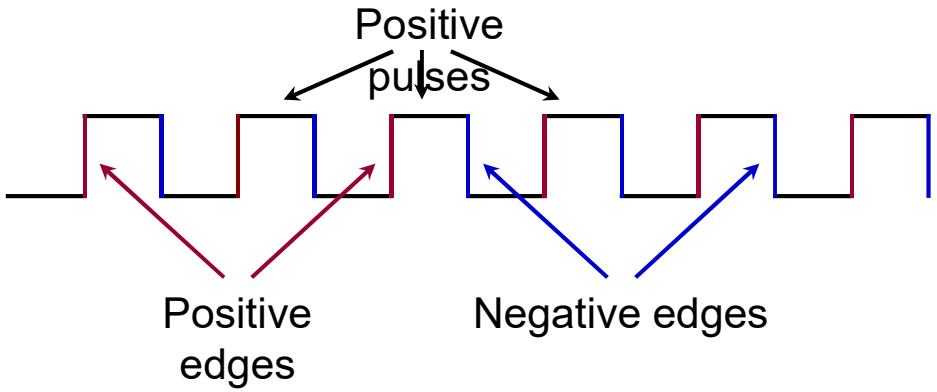
Comparison between synchronous and asynchronous sequential Circuits

Synchronous sequential circuits	**Asynchronous Sequential circuits**
Memory elements are clocked FFs.	Memory elements are either unlocked FFs or time delay element
The change in input signal can affect memory elements upon activation of clock signal	Change in input signal can affect memory elements at any instant of time
The maximum operating speed of the clock depends on time delays involved	Because of the absence of the clock, asynchronous circuits can operate faster than synchronous circuits
Easier to design	More difficult to design

Sequential Logic Circuits

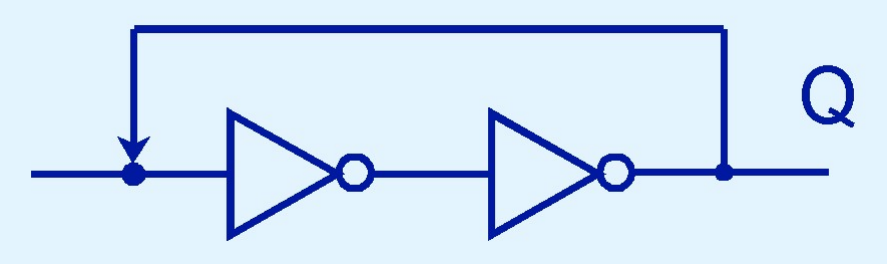
- Clock is usually a square wave.
- State changes occur in sequential circuits only when the clock ticks.
- Circuits can change state on the rising edge, falling edge, or when the clock pulse reaches its highest voltage.
- Circuits that change state on the rising edge, or falling edge of the clock pulse are called *edge-triggered.*
- *Level-triggered circuits* change state when the clock voltage reaches its highest or lowest level





Sequential circuit

- **A sequential circuit is a type of digital circuit in electronics that contains memory units to store and propagate information, and its output depends on both current inputs and internal state.**
- To retain/store their state values, sequential circuits rely on *feedback.*
- Feedback in digital circuits occurs when an output is looped back to the input.
- A simple example of this concept is shown below.
  - If Q is 0 it will always be 0, if it is 1, it will always be 1.
    Why? (double-inverter)
  - A one-bit buffer (memory)



Analysis of Combinational vs Sequential Circuits

Combinational :

- Boolean Equations
- Truth Table

- Output as a function of inputs

Sequential :

- State Equations
- State Table
- State Diagram

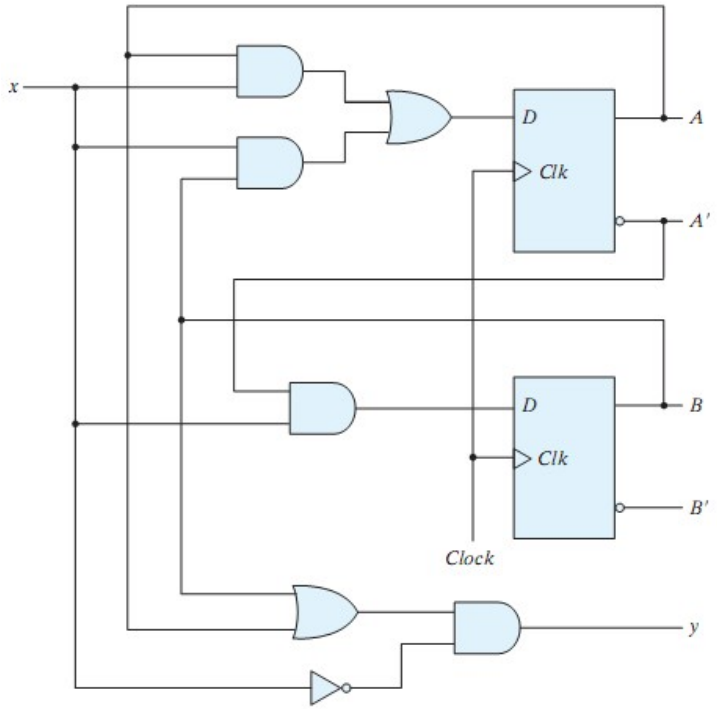
- Output as a function of input and previous state
- Next state as a function of inputs and current state.

State Equations

- A **state equation** is a Boolean expression which specifies the next state and output as a function of the present state and inputs.

Example:

- The shown circuit has two D-FFs (A,B), an input x and output y.
- The D input of a FF determines the next state
  - A(t+1) = A(t)x+B(t)x = Ax+Bx
  - B(t+1) = A'(t)x = A'x
- Output:
  - y = (A+B)x'



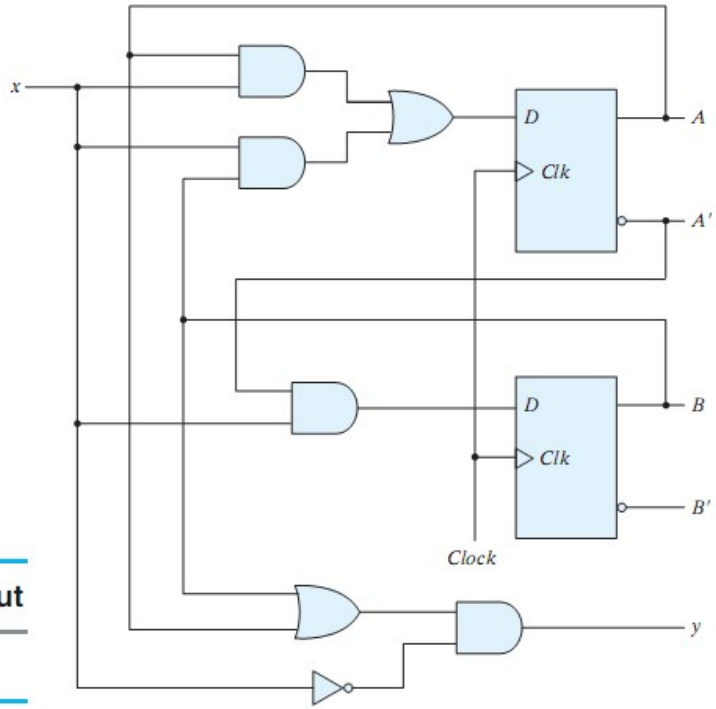
State Table

A **state table** is a table enumerating all present states, inputs, next states and outputs.

- Present state, inputs: list all combinations
- Next states, outputs: derived from state equations

4 sections

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

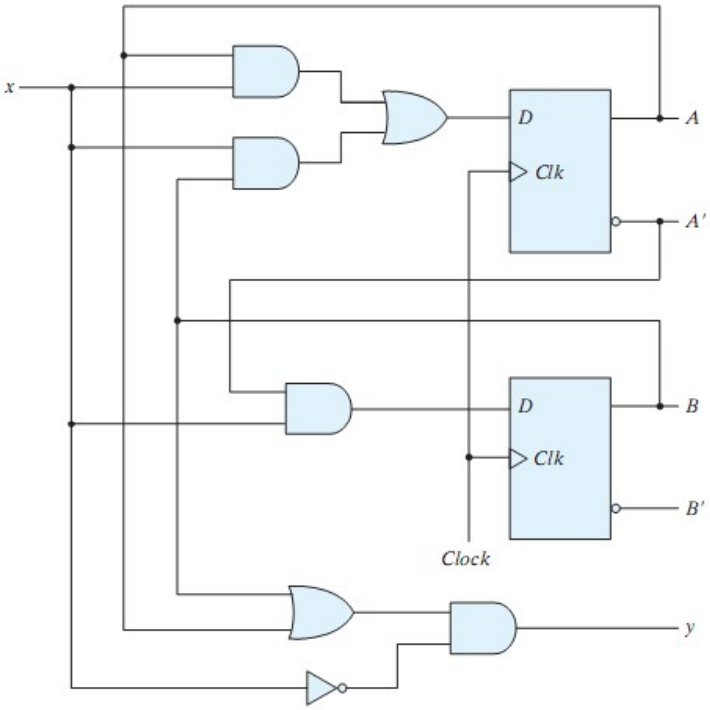


State Table

A **state table** is a table enumerating all present states, inputs, next states and outputs.

- Present state, inputs: list all combinations
- Next states, outputs: derived from state equations

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

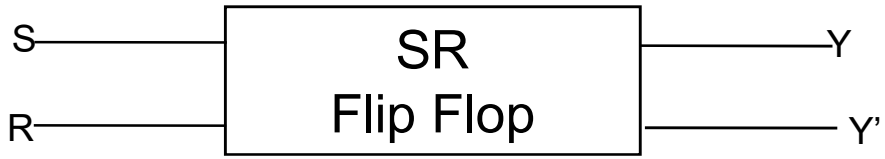


Flip-flops

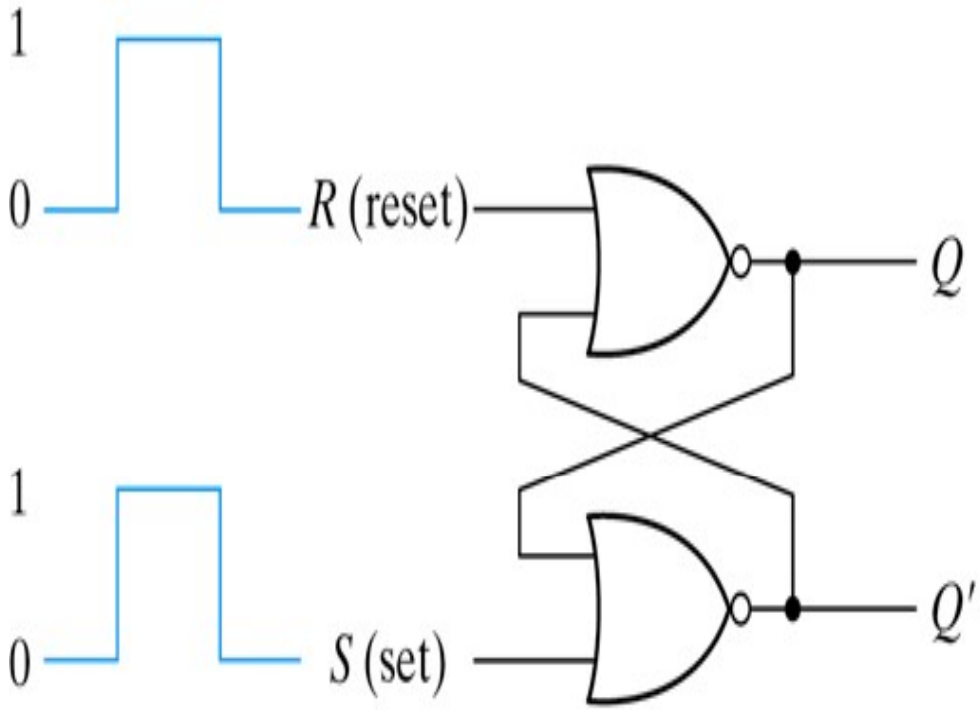
- A flip-flop is a binary cell, which stores 1-bit of information. It itself is a sequential circuit.
- It has two outputs, one for the normal value and one for the complement value.
- Flip-flop can change its state when clock pulse occurs but when? When the clock transitions from 0 to 1 (rising edge) or from 1 to 0 (falling edge) and not when clock is 1.
- If the storage element changes its state when clock is exactly at 1 then it is called ***latch.***
- ***In simple words, flip-flop is edge-triggered and latch is level triggered.***
- A latch or a flip-flop can be constructed using two NOR or NAND gates.

SR **Flip-flops**

- This flip flop is constructed from two NOR gates connected back to back. It can also be constructed from two NAND gates.
- The crossed coupled connection from the output of one gate to the input of the other gate continues a feedback path.
- Each flip flop has two outputs y and y' (or Ỹ) and two inputs S for 'setting' and R for 'resetting' (i.e.; clearing).
- This type of flip flop is called a direct coupled RS flipflop, the R and S being the first letters of the two inputs names.
- Block diagram



SR Flip-flops Circuit diagram



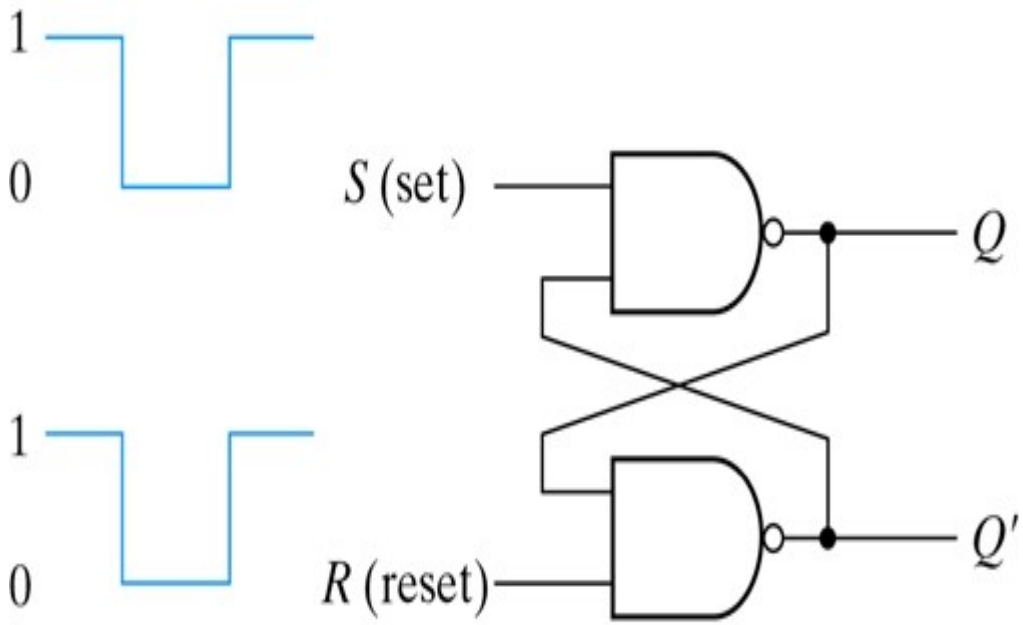
(a) Logic diagram

S	R	Q	Q′	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	

(b) Function table

SR Latch with NOR Gates

SR **Flip-flops Circuit diagram**



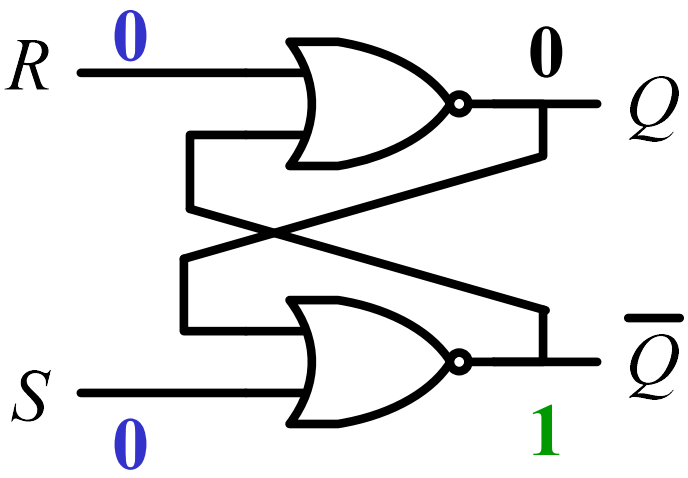
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	

(a) Logic diagram (b) Function table

SR Latch with NAND Gates

Latches

- *SR* Latch



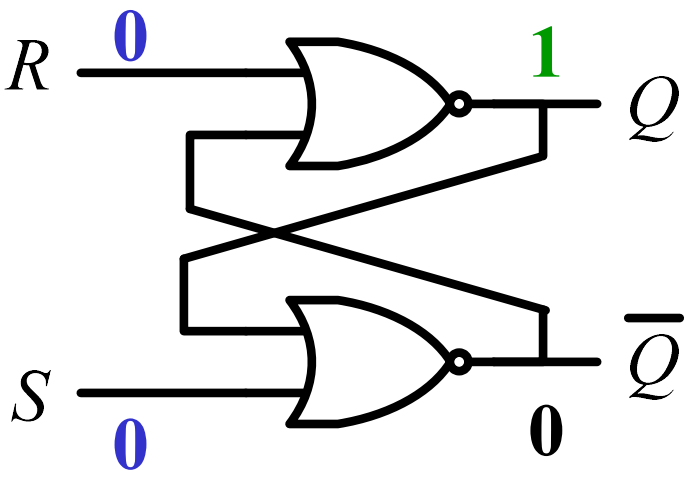
S	R	Q_0	Q	Q'
0	0	0	0	1

$Q = Q_0$

Initial Value

Latches

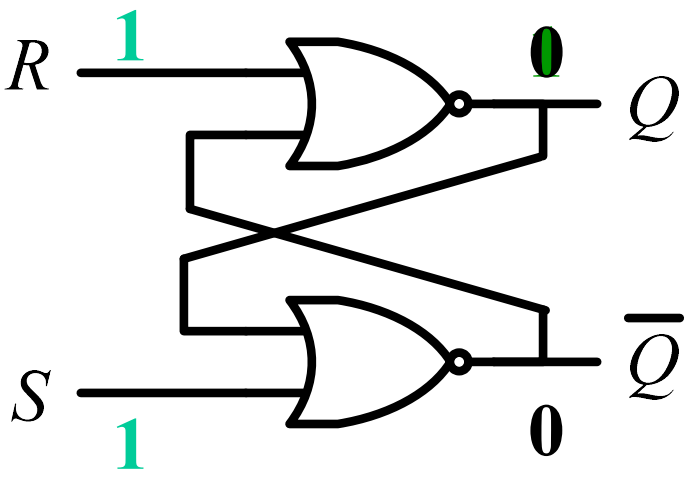
- *SR* Latch



S	*R*	Q_0	*Q*	*Q'*	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	$Q = Q_0$

Latches

- *SR* Latch



S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$
1	1	1	0	0	$Q = Q'$

Latches

- *SR* Latch

S	*R*	*Q*	
0	0	Q_0	No change
0	1	0	Reset
1	0	1	Set
1	1	*Q*=*Q*'=0	Invalid

S	*R*	*Q*	
0	0	*Q*=*Q*'=1	Invalid
0	1	1	Set
1	0	0	Reset
1	1	Q_0	No change

SR **Flip-flops behavior**

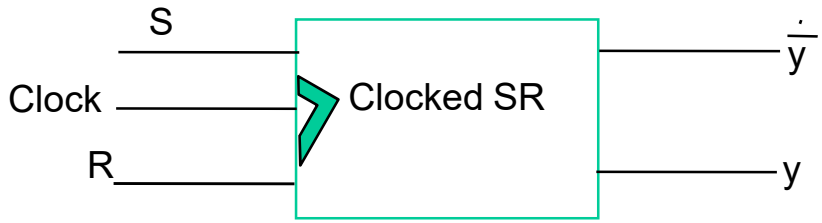
- Race condition : When SR = 11, Y_{next} = 0 . In this case, the value of y is set to 0 (irrespective of its previous value).
- However in this case the value of y will alter from 0 or 1 and from 1 to 0 again and again. So this situation is called as “Race condition”.
- When SR = 11 in SR flip flop, the value of y, after some time becomes unpredictable and circuit randomly stores 1 or 0 into y. This situation is known as “Race condition”
- Assume that y = 0 , y’ = 1 and SR = 11. so new y’ = (y+s)’ = (0+1)’ = 1’ = 0 ∴next y = (y’+R)’ = (0+1)’ = 1’ = 0 so y = 0 and y’ = 0
- The output will remain until SR = 11. now when SR becomes 00 the u and y’ will become 00, 11, 00, 11, ….
- And finally depending on delay of both gates, yy’ will become either 01 or 10. This result is unpredictable.

Comparing SR with NOR and NAND gate

- Comparing the SR NAND-Latch and the SR NOR-Latch, we note that the input signals for the NAND required the complement values of those used for the NOR-Latch.
- Because the NAND-Latch require a “0” signal to change its state, it is sometimes referred to as an S-R Latch, the bar above the letters designates the fact that the inputs must be in their complement form to activate the circuit.

Clocked SR

- By adding gates to the inputs of the latch, the FF can be made to respond only during the occurrence of clock pulse.
- This is clocked SR FF.
- It is often convenient to prevent the latch from changing state except at certain specified times. To achieve this SR FF circuit is modified slightly.
- The circuit has an additional input the (the clock) which is normally zero.
- In clocked SR FF main part of the circuit is same but input signals are coming through AND gate i.e. the clock controls the effect of SR in the circuit through AND gate.
- **Block diagram**



Clocked SR

- When clock is zero the basic circuit have both input as 00 (i.e. both AND gates output is 0) independent of S and R value and the latch does not change state.
- Thus clock 00 disable the operation of S and R.
- When clock is 1 the circuit behaves same as unclocked SR FF.
- i.e. effect of the AND gates vanishes and the lath becomes sensitive to S and R. Thus clock controls the operations of SR.

Case i : clock = 0;
•R = Rprev * clock = R * 0 = 0;
•S = Sprev * clock = S * 0 = 0.
•So the basic FF has SR = 00 as input so the output of y and y' will not change. Thus FF remains in the same state.

Case ii : clock = 1;
•R = Rprev * clock = R * 1 = R;
•S = Sprev * clock = S * 1 = S.
•So the basic FF has SR as input and accordingly it will change the value of y and y'. Thus FF behaves as simple SR FF

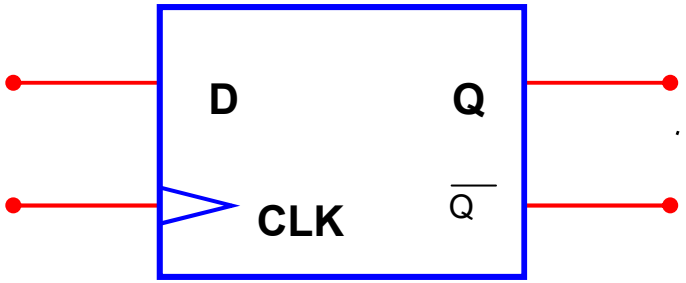
Controlled Latches

- *SR* Latch with Control Input/ Clocked SR flip flop

C *S* *R*	*Q*	
0 x x	Q_0	**No change**
1 0 0	Q_0	**No change**
1 0 1	0	**Reset**
1 1 0	1	**Set**
1 1 1	*Q*=*Q'*	**Invalid**

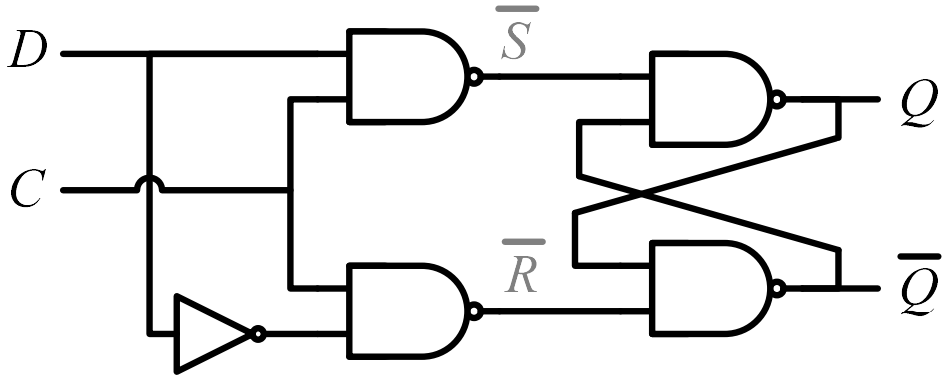
Clocked D FF

- The problem of undefined output in SR flip-flop when both R & S become 1 gets avoided in D flip-flop by providing just a single input.
- Thus, the non-clocked inputs to AND gates (S &R) are guaranteed to be opposite of each other by inserting an **inverter between them.**
- The clocked D FF (D stand for delay)
- This FF has only one input D, and the input to the lower AND gate is always the complement of the both input to the upper one.
- The problem of both inputs being 1 never arises.
- Block diagram

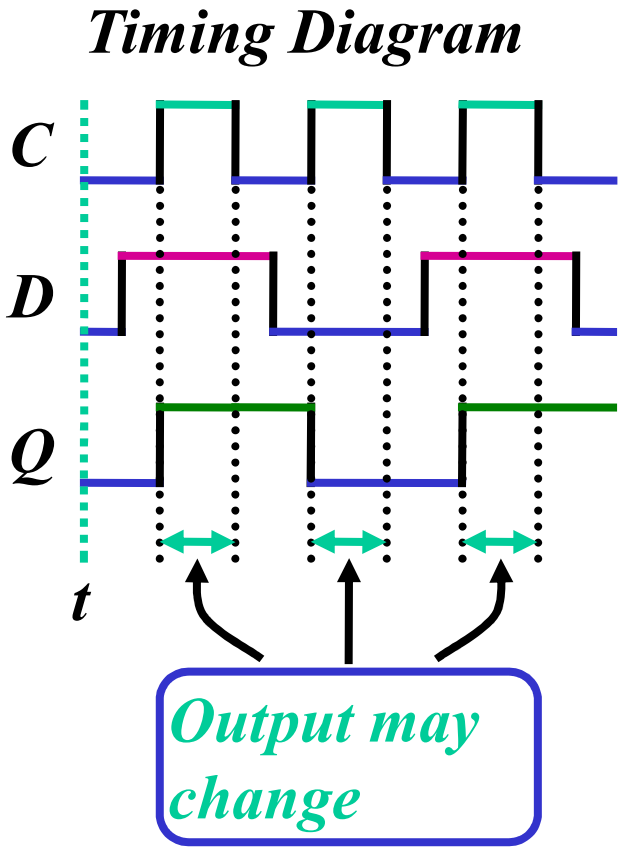


Controlled Latches

- *D* Latch (*D* = *Data*)

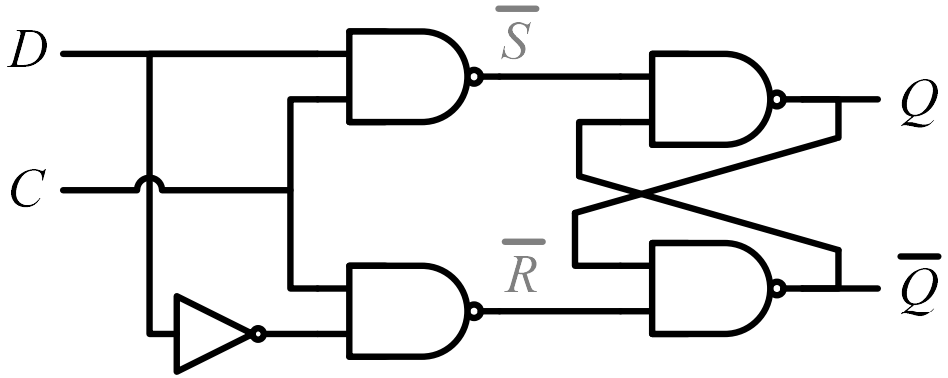


C D	Q	
0 x	Q_0	No change
1 0	0	Reset
1 1	1	Set

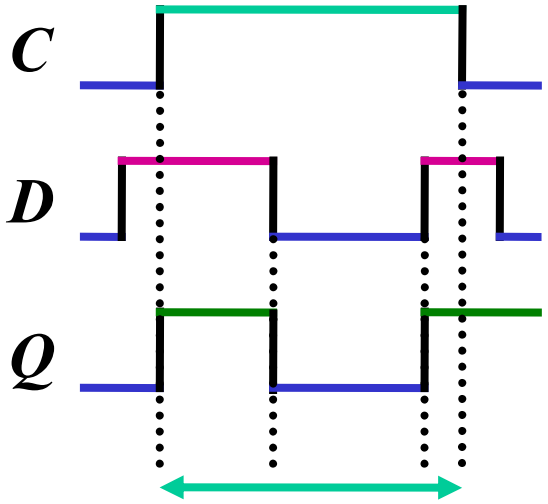


Controlled Latches

- *D* Latch (*D* = *Data*)



Timing Diagram



C D	Q	
0 x	Q_0	No change
1 0	0	Reset
1 1	1	Set

Output may change

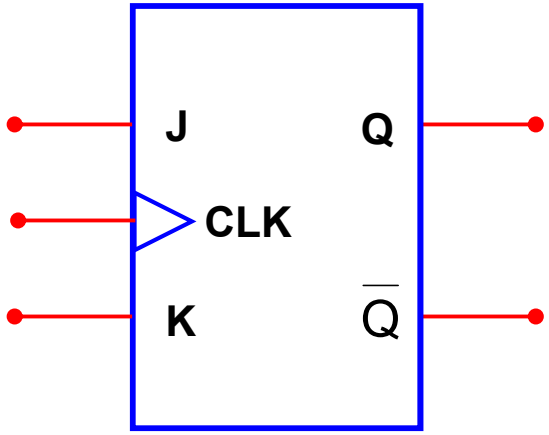
Clocked D FF

- D flip-flop is also referred as Delay flip-flop because it delays the 0 or 1 applied to its input by a single clock pulse.
- When D = 0 and the clock is 1 the latch is driven into state logical 0. In other terms, when the clock is 1, the current value of D is sampled and stored in the latch.
- This circuit is a true 1-bit memory.
- The value stored is always available at y.
- To load the current value of D into memory, a positive pulse is put on the clock line.
- Behavior :
  - When clock = 0, the FF remains in the same state (i.e. value of y remains unchanged)
  - When clock = 1 the FF changes the value of y depending on D. if D = 0 the y = 0. If D = 1 then y =1

Clocked JK FF

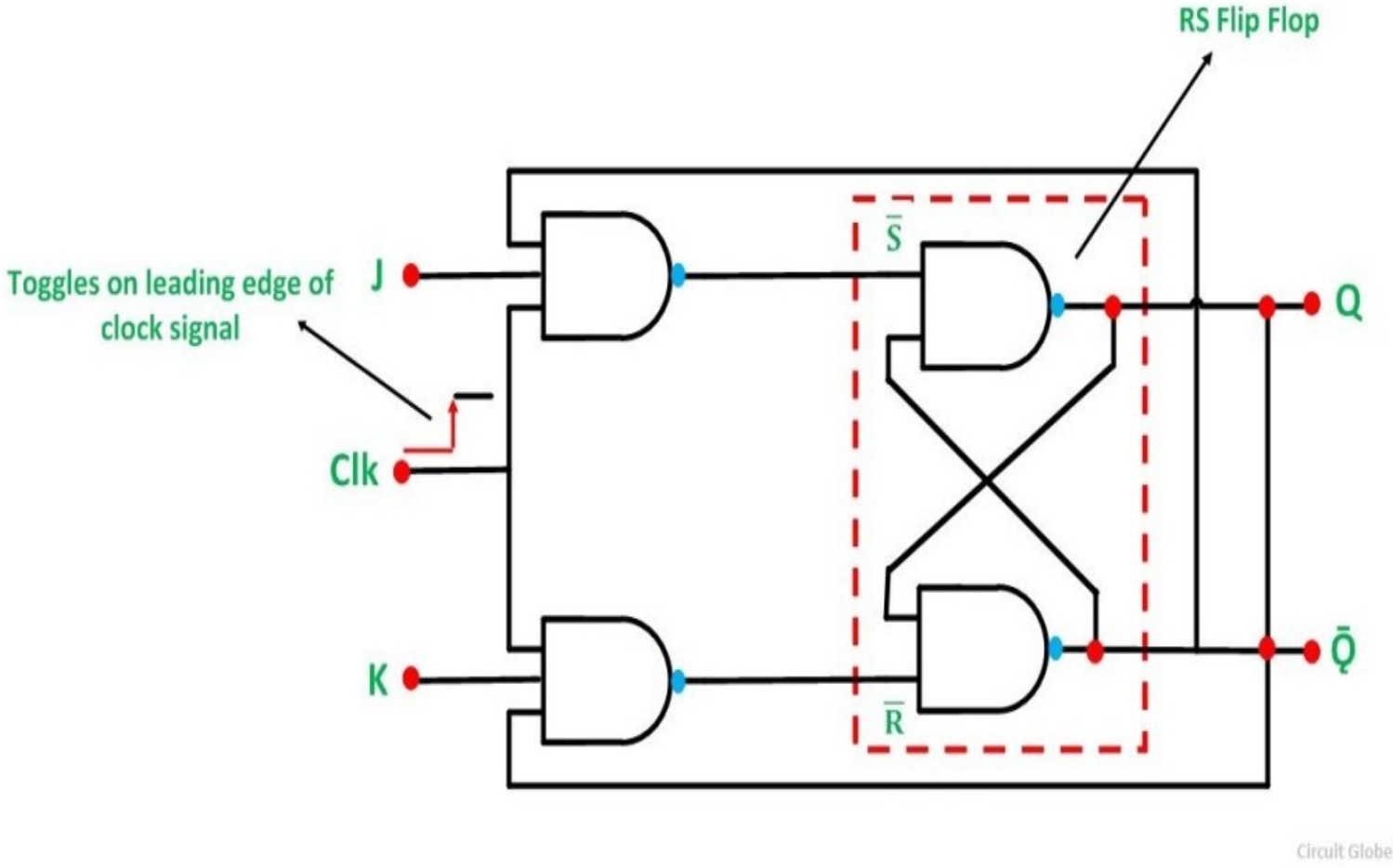
- The J-K flip-flop is also a modification of SR flip-flop, it has 2 inputs like S & R and all possible inputs combinations are valid in J K flip-flop.
- The difference is that the JK Flip Flop does not have the invalid input states of the SR Latch (when S and R are both 1).
- The JK Flip Flop name has been kept on the inventor name of the circuit known as Jack Kilby. The basic symbol of the JK Flip Flop is shown below.

- Block diagram

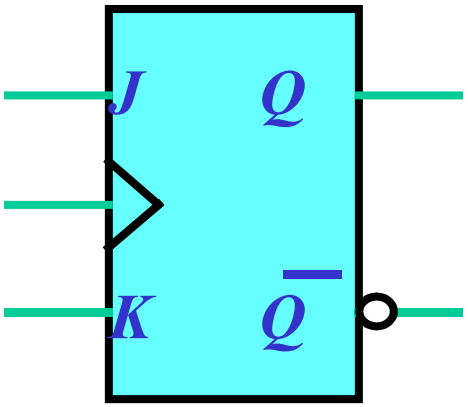


Clocked JK FF

- Clocked JK Flip Flop



Flip-Flop Characteristic Tables



J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Toggle

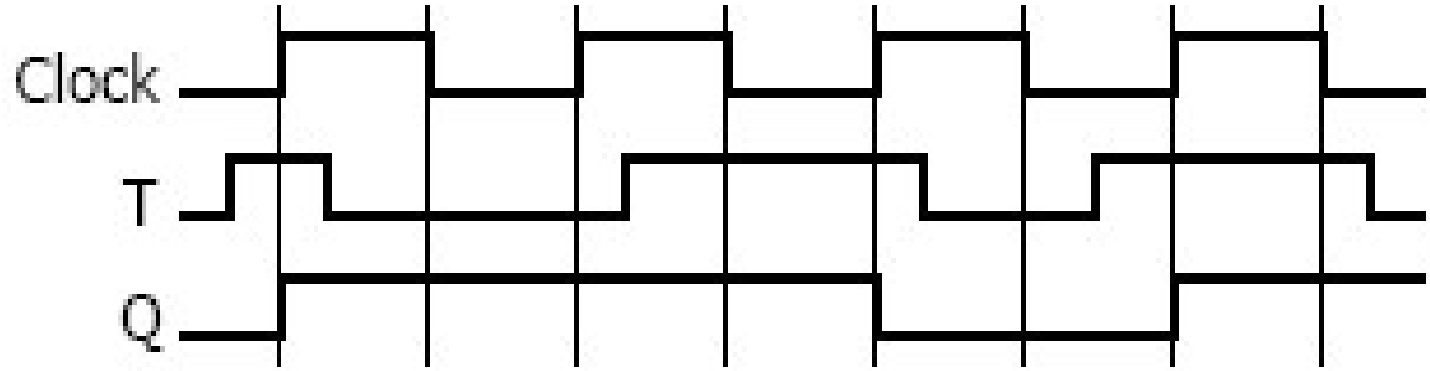
T Flip-Flop

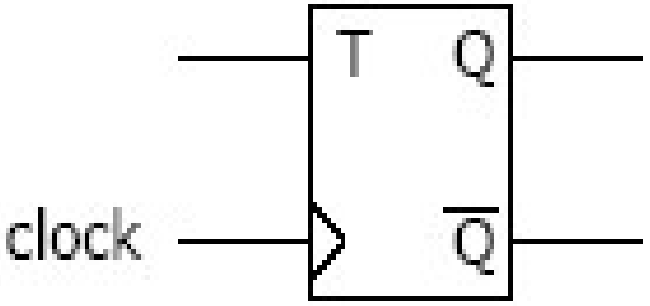
- The T flip-flop is a single input version of the JK flip-flop.
- The T flip-flop is obtained from the JK type if both inputs are tied together.
- The output of the T flip-flop "toggles" with each clock pulse.

T flip-flop

- The name T derives from the behavior of the circuit, which 'toggles' its state when T=1
  - This feature makes the T flip-flop a useful element when constructing counter circuits

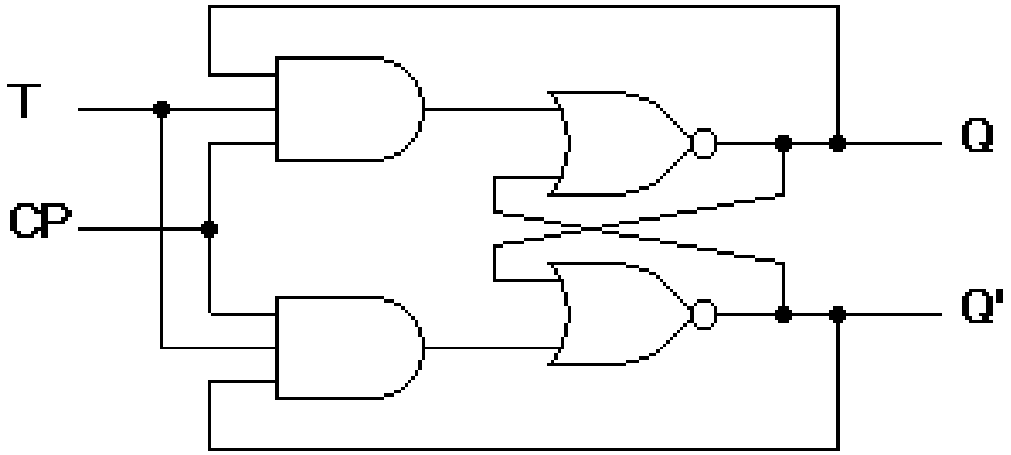
T	Q(t+1)
0	Q(t)
1	Q'(t)



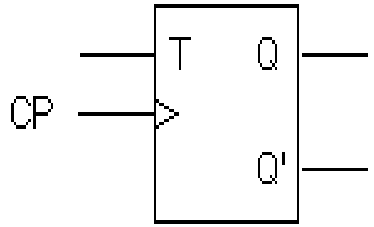


Positive edge triggered

T Flip-Flop



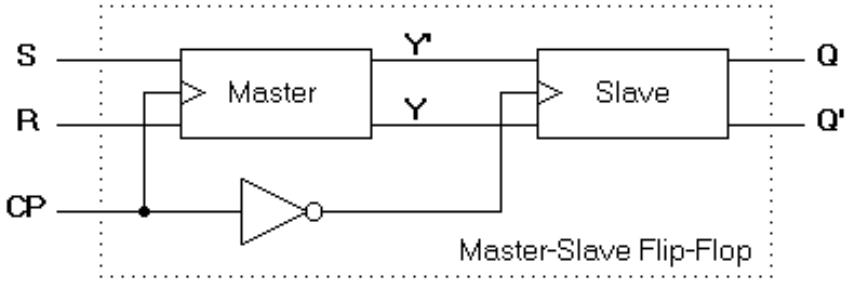
C	T	Next state of Q
0	X	No change
1	0	No change
1	1	complement

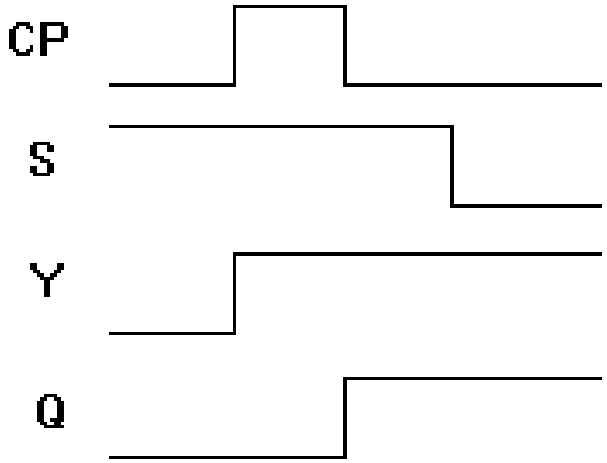


Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Master-Slave Flip Flop

- The timing relationship is shown in Figure and is assumed that the flip-flop is in the clear state prior to the occurrence of the clock pulse.
- The output state of the master-slave flip-flop occurs on the negative transition of the clock pulse.
- Some master-slave flip-flops change output state on the positive transition of the clock pulse by having an additional inverter between the CP terminal and the input of the master.



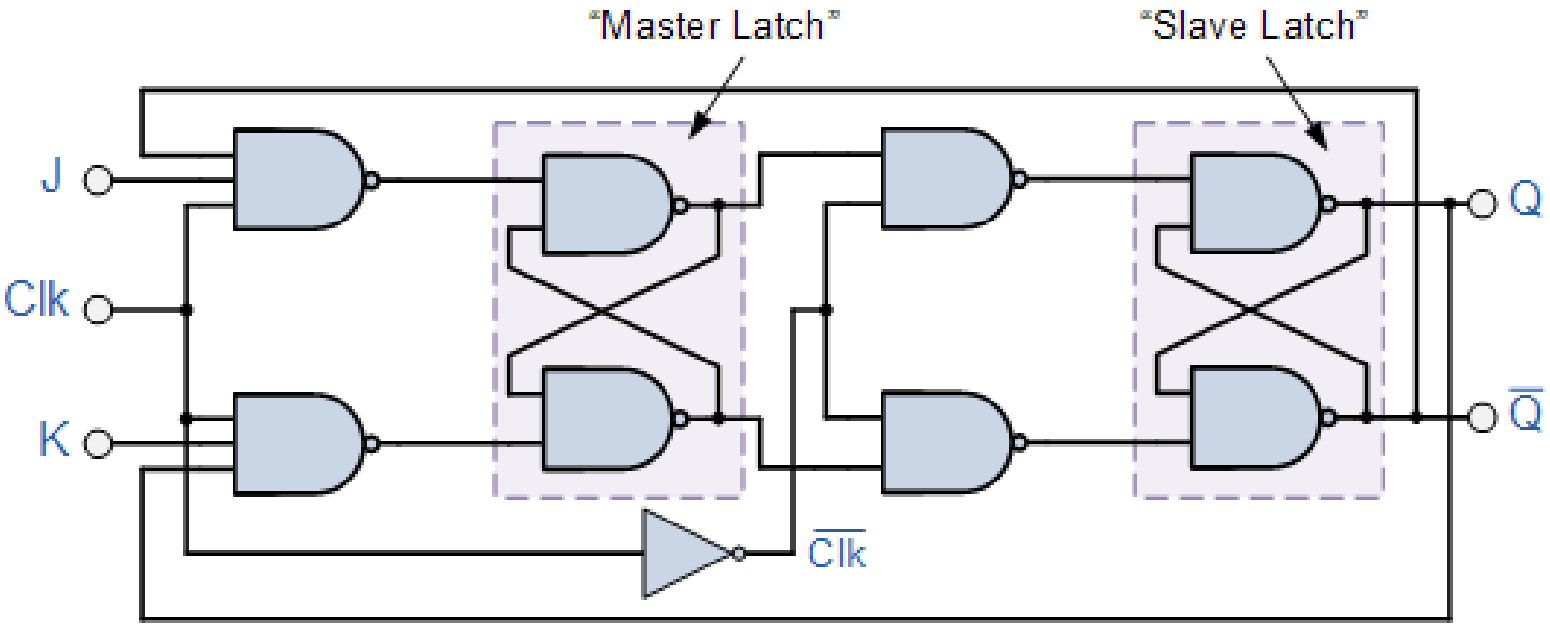


Master-Slave with a JK Flip-Flop

Truth Table

Trigger	Inputs		Output						Inference
			Present State		Intermediate		Next State		
CLK	J	K	Q	$\bar{Q}$	M_1	M_2	Q	$\bar{Q}$	
↑	0	0	0	1	0	1	Latched		No Change
↓			0	1	Latched		0	1	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	0	1	0	1	0	1	Latched		Reset
↓			0	1	Latched		0	1	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	
↑	1	0	0	1	1	0	Latched		Set
↓			0	1	Latched		1	0	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	1	1	0	1	1	0	Latched		Toggles
↓			0	1	Latched		1	0	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	

Master-Slave with a JK Flip-Flop



Master-Slave with a JK Flip-Flop

- The input signals J and K are connected to the gated "master" SR flip-flop which "locks" the input condition while the clock (Clk) input is "HIGH" at logic level "1".
- As the clock input of the "slave" flip-flop is the inverse (complement) of the "master" clock input, the "slave" SR flip-flop does not toggle.
- The outputs from the "master" flip-flop are only "seen" by the gated "slave" flip-flop when the clock input goes "LOW" to logic level "0".
- When the clock is "LOW", the outputs from the "master" flip-flop are latched and any additional changes to its inputs are ignored.
- The gated "slave" flip-flop now responds to the state of its inputs passed over by the "master" section.
- Then on the "Low-to-High" transition of the clock pulse the inputs of the "master" flip-flop are fed through to the gated inputs of the "slave" flip-flop and on the "High-to-Low" transition the same inputs are reflected on the output of the "slave" making this type of flip-flop edge or pulse-triggered.

Introduction: Registers

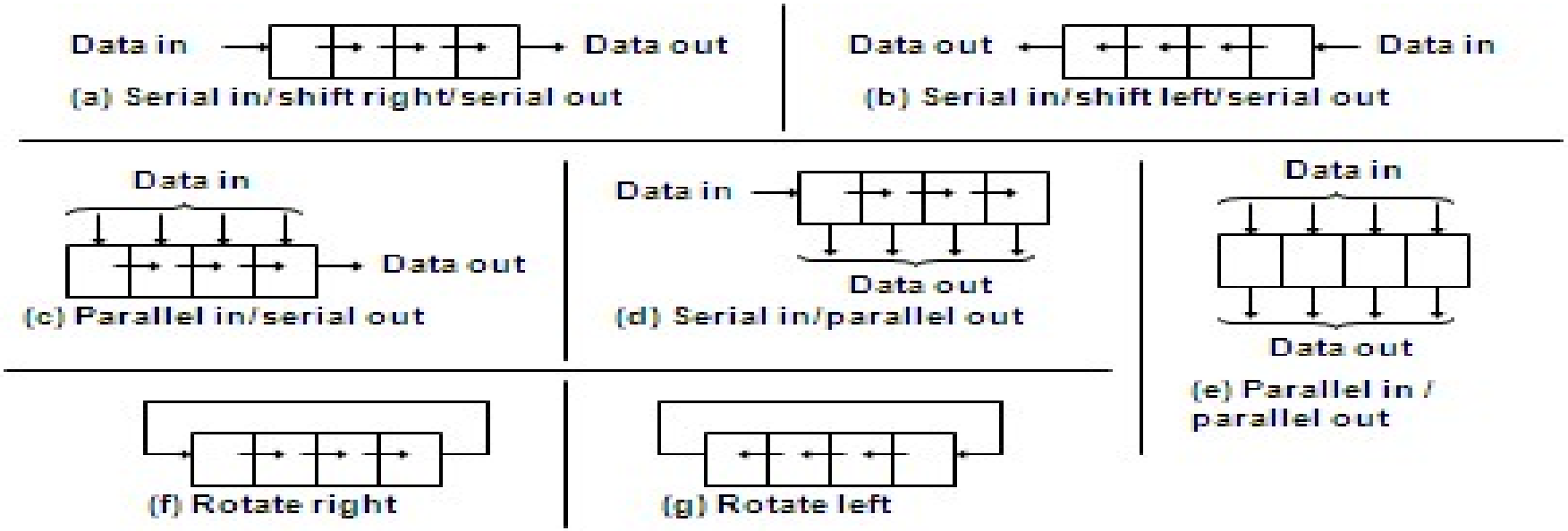
- Multiple flip flops can be combined to form a data register
- Shift registers allow data to be transported one bit at a time
- Registers also allow for parallel transfer
  - Many bits transferred at the same time
- Shift registers can be used with adders to build arithmetic units. For example the operations of complementation, multiplication and division are frequently implemented by means of a register.
- A register capable of shifting in one direction only is a unidirectional shift register. One that can shift in both directions is bidirectional shift register.
- Remember: most digital hardware can be built from combinational logic (and, or, invert) and flip flops
  - Basic components of most computers

Introduction: Registers

- An *n*-bit register has a group of *n* flip-flops and some logic gates and is capable of storing *n* bits of information.
- The flip-flops store the information while the gates control when and how new information is transferred into the register.
- Some functions of register:
  - ❖ Retrieve data from register
  - ❖ Store/load new data into register (serial or parallel)
  - ❖ Shift the data within register (left or right)
- Another function of a register, besides storage, is to provide for data movements.

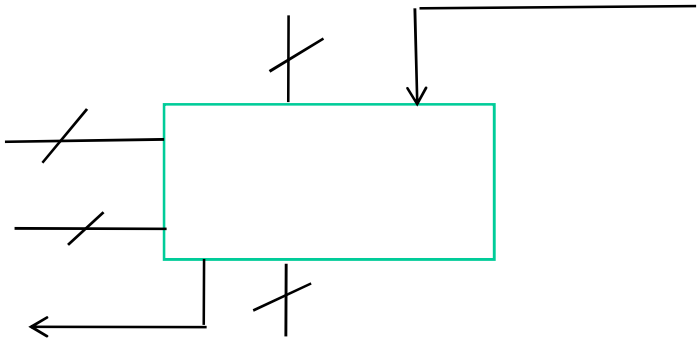
Shift Registers

- Each *stage* (flip-flop) in a shift register represents one bit of storage, and the shifting capability of a register permits the movement of data from stage to stage within the register, or into or out of the register upon application of clock pulses.
- Basic data movement in shift registers (four bits are used for illustration).



Shift Left

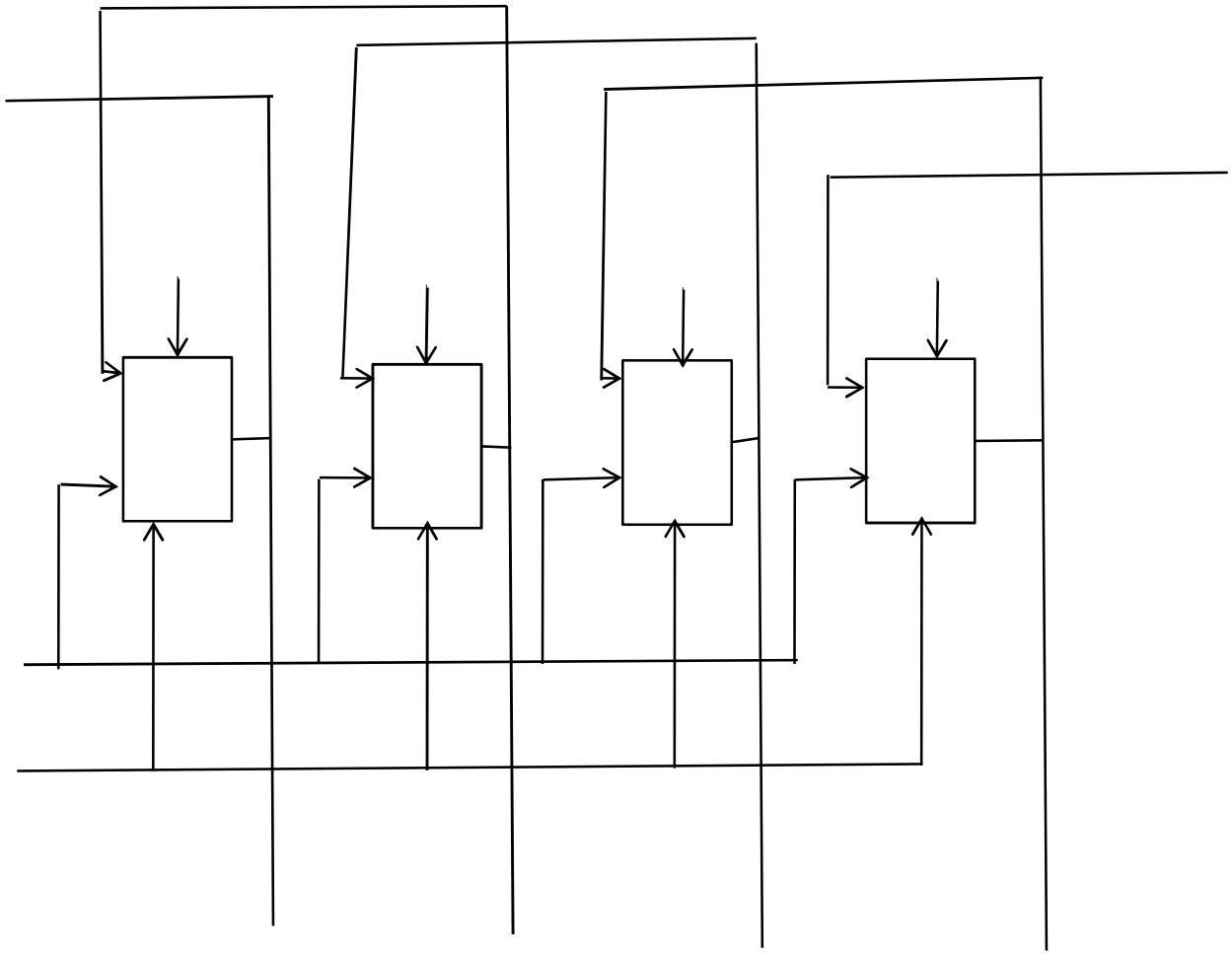
- A register capable of shifting its binary information either to the right or to the left is called a shift register.
- An n –bit shift register consists of n FF and the gates that control shift operations.
- Shift left register
  - This circuit shifts all bits to its left position by 1 bit.
  - Block diagram



Shift Left

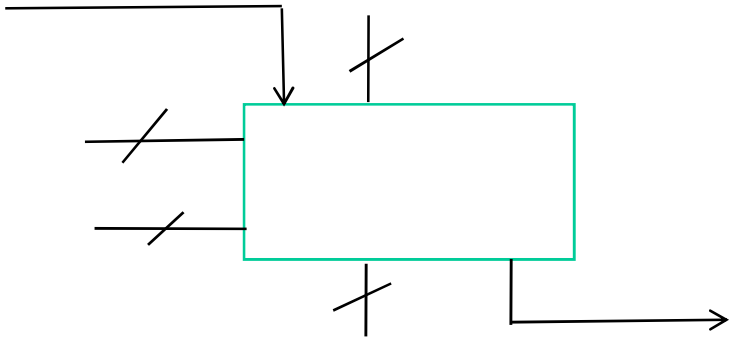
- The SHL data output is leftmost bit of the register.
- The SHL data input is inserted in rightmost bit of the register.
- Suppose y = 0110 and SHL data input = 1. After shifting y = 1101 and SHL data output = 0.
- Characteristics
  - Preset and clear are used to initialize the value of FF.
  - Clock is used to unable and disable the FF circuit
  - Timing of clock is very important in this circuit. It is set to one “1” for small amount of time, then immediately set to zero “0”.
  - If it continues for “1” for more time the circuit will again shift value. So timing is important in this circuit to shift the data by exactly one bit

Shift Left Circuit



Shift Right

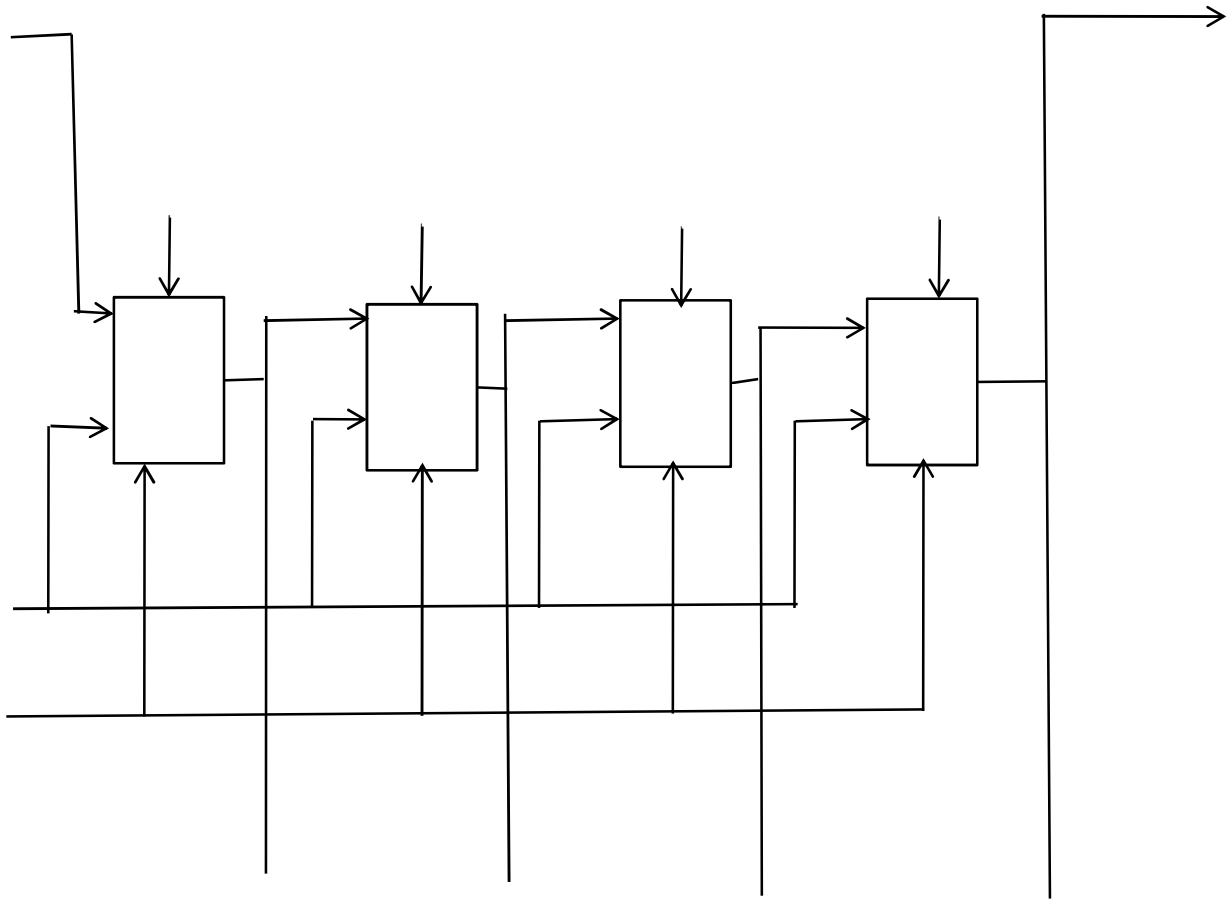
- This register shifts all the bits to its right position by 1 bit
- Block diagram



Shift Right

- The SHR data output is rightmost bit of the register.
- The SHR data input is inserted in leftmost bit of the register.
- Suppose y = 0110 and SHR data input = 1. After shifting y = 1011 and SHR data output = 0.
- Characteristics
  - Preset and clear are used to initialized the value of FF.
  - Clock is used to unable and disable the FF circuit

Shift Right Circuit



Universal Left right shift register with parallel input

- A Universal shift register is a register which has both the right shift and left shift with parallel load capabilities. Universal shift registers are used as memory elements in computers.
- The Universal shift register is a combination design of **bidirectional** shift register and a **unidirectional** shift register with parallel load provision.

Universal Left right shift register with parallel input

- A n-bit universal shift register consists of n flip-flops and n 4×1 multiplexers.
- All the n multiplexers share the same select lines(S1 and S0)to select the mode in which the shift register operates. The select inputs select the suitable input for the flip-flops.

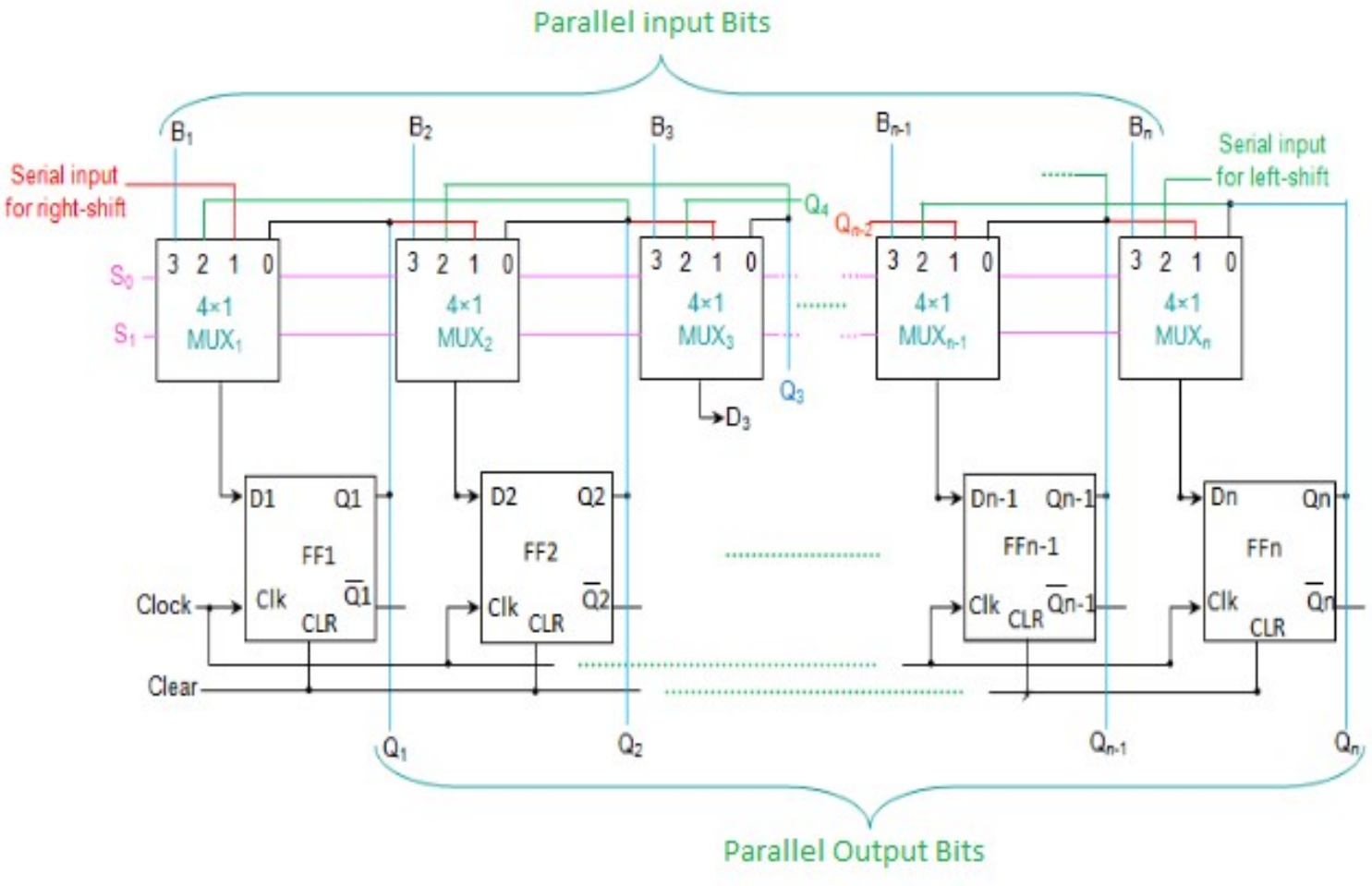
Universal Left right shift register with parallel input

- The most general shift register has the following capabilities:
  - A clear control to clear the register to 0
  - A clock input to synchronize the operations.
  - A shift-right control to enable the shift-right operation and serial input and output lines associated with the shift-right.
  - A shift-left control to enable the shift-left operation and serial input and output lines associated with the shift-left.
  - A parallel load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
  - N parallel output lines
  - A control state that leaves the information in the register unchanged in the presence of the clock

Universal Left right shift register with parallel input

- This register performs 3 functions
  - Parallel input
  - Shift left
  - Shift right
- When S_0S_1 = 00 this circuit is disable (Clock signal for all bits is 0)
- When S_0S_1 = 01 this circuit performs shift left operation
- When S_0S_1 = 10 this circuit performs shift right operation
- When S_0S_1 = 11 this circuit performs parallel input operation
- This circuit allows a binary number placed on parallel input pins, to be loaded into the internal registers.
- With parallel load shift register, this parallel input data appears immediately on the parallel outputs, but with the other a clock pulse is required to move the parallel-loaded data to the output. It may be parallel load, shift left, shift right or do nothing, as determined by the programming on the S_0S_1 inputs.

Universal Shift Register



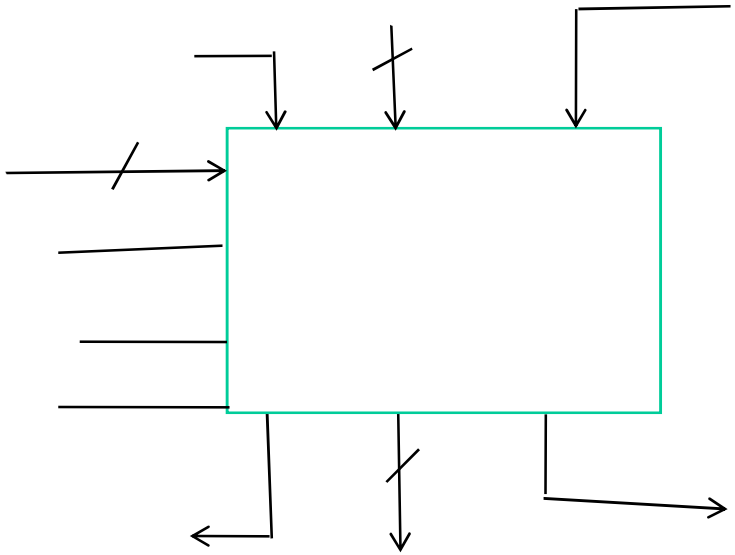
Applications of shift registers

1. The first input (zeroth pin of multiplexer) is connected to the output pin of the corresponding flip-flop.
2. The second input (first pin of multiplexer) is connected to the output of the very-previous flip flop which facilitates the right shift.
3. The third input (second pin of multiplexer) is connected to the output of the very-next flip-flop which facilitates the left shift.
4. The fourth input (third pin of multiplexer) is connected to the individual bits of the input data which facilitates parallel loading.

The working of the Universal shift register depends on the inputs given to the select lines.
The register operations performed for the various inputs of select lines are as in the table.:

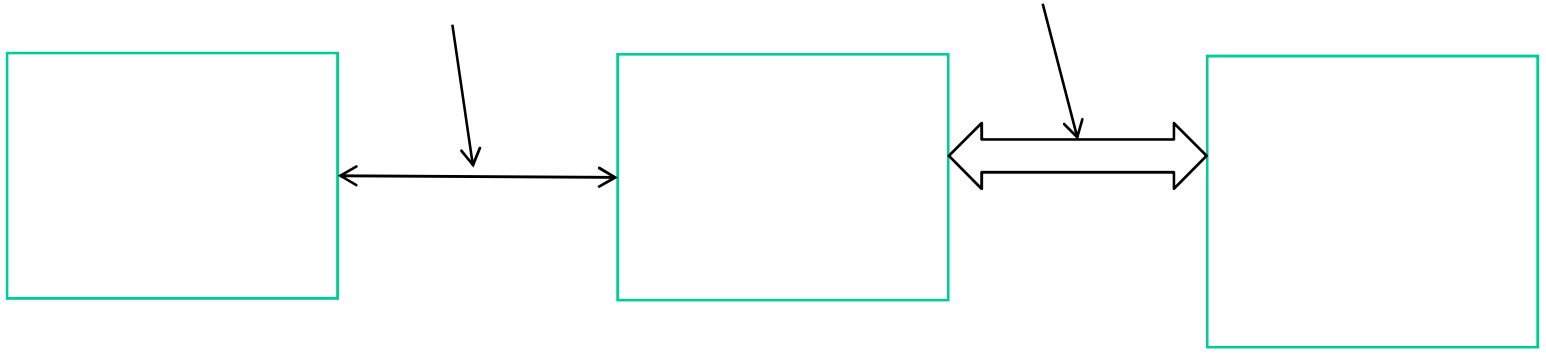
S1	s0	Register operation
0	0	No changes
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Universal Left right shift register with parallel input



Applications of shift registers

- Time delays
- Serial/Parallel data conversion
- Ring counters
- Universal asynchronous receiver transmitter (URAT) as an interface device
- Serial transfer used in modems and computer peripherals (e.g. mouse)



An Application – Serial Addition

- A = 0100; B = 0111. $A + B$ = 1011 is stored in A after 4 clock pulses.

		0 1 0 0 0 1 1 1	0
1	0	1 0 1 0 0 1 1	0
1	0	1 1 0 1 0 1	0
0	1	0 1 1 0 0	1
1	0	1 0 1 1	0

Introduction: Counters

- A **Counter** is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.
- Counters are used in digital electronics for counting purpose, they can count specific event happening in the circuit.
- For example, in *UP counter* a counter increases count for every rising edge of clock.
- Not only counting, a counter can follow the certain sequence based on our design like any random sequence 0,1,3,2… They can also be designed with the help of flip flops.
- Counter works in two modes
  - Up counter
  - Down counter

Introduction: Counters

- A register that goes through a predetermined sequence of states upon the application of input pulses is called a counter
- Counters are circuits that cycle through a specified number of states.
- Two types of counters:
  - ❖ synchronous (parallel) counters
  - ❖ asynchronous (ripple) counters
- Ripple counters allow some flip-flop outputs to be used as a source of clock for other flip-flops.
- Parallel counters apply the same clock to all flip-flops.

Counters

- Ripple counters triggered by initial Count signal
- Applications:
  - Watches
  - Clocks
  - Alarms
  - Web browser refresh

Synchronous versus asynchronous counter

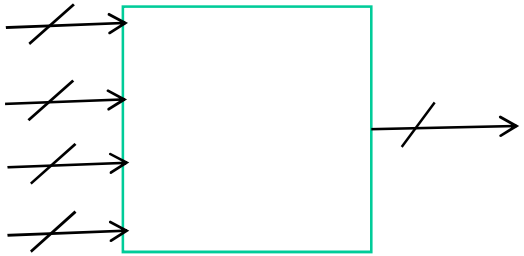
Asynchronous counter	**Synchronous Counter**
In this type of counter FFs are connected in such a way that the output of first FF drives the clock for the second the clock of the third and so on	In this type of counter there is no connect ion between the output of first FF and clock input of the next FF and so on
All the FF are not clocked simultaneously	All the FF are clocked simultaneously
Design and implementation is very simple even for more number of states	Design and implementation becomes tedious and complex as the number of states increases.
Main draw back of these counter is their low speed as the clock is propagated through a no. of FF before it reaches the last FF	Since clock is applied to all the FF simultaneously the total propagation delay is equal delay of only one FF. hence they are faster.

Asynchronous (Ripple) Counters

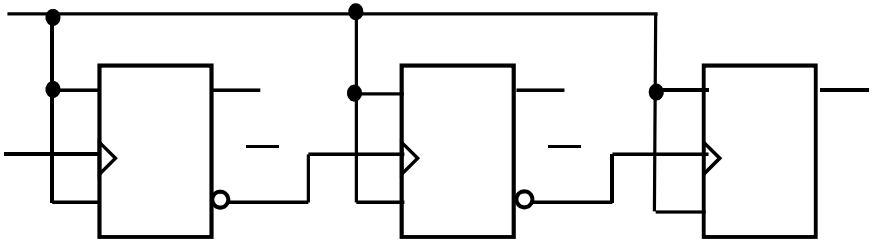
- Asynchronous counters: the flip-flops do not change states at exactly the same time as they do not have a common clock pulse.
- In asynchronous counter we don’t use universal clock, only first flip flop is driven by main clock and the clock input of rest of the following flip flop is driven by output of previous flip flops.
- Also known as ripple counters, as the input clock pulse “ripples” through the counter – cumulative delay is a drawback.

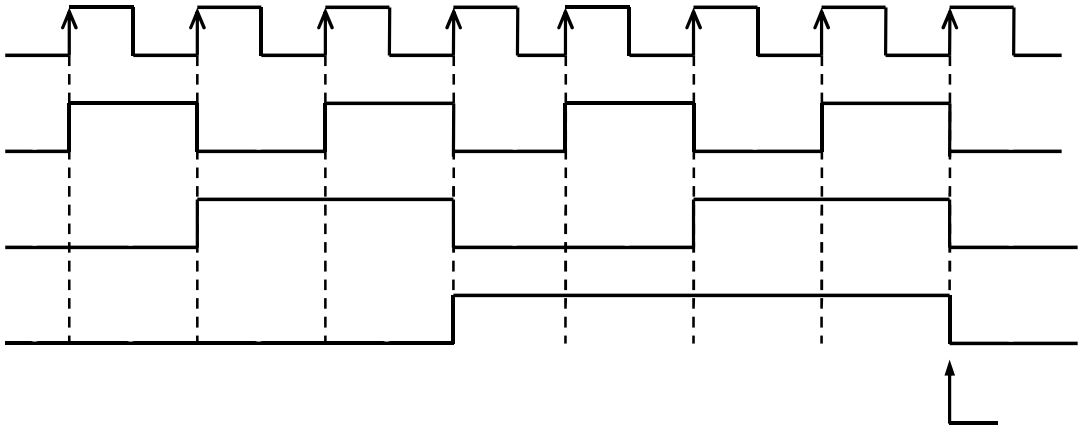
Asynchronous (Ripple) Counters

- Example: 3-bit ripple binary counter.
- Block diagram



Circuit





Asynchronous (Ripple) Counters

- Example: 3-bit ripple binary counter Truth Table.

Clock Pulse	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Asynchronous (Ripple) Counters

- It is evident from timing diagram that Q0 is changing as soon as the rising edge of clock pulse is encountered, Q1 is changing when rising edge of Q0 is encountered(because Q0 is like clock pulse for second flip flop) and so on.
- In this way ripples are generated through Q0,Q1,Q2,Q3 hence it is also called **RIPPLE counter and serial counter.**
- A ripple counter is a cascaded arrangement of flip flops where the output of one flip flop drives the clock input of the following flip flop

Synchronous counters

- Unlike the asynchronous counter, synchronous counter has one global clock which drives each flip flop so output changes in parallel.
- The one advantage of synchronous counter over asynchronous counter is, it can operate on higher frequency than asynchronous counter as it does not have cumulative delay because of same clock is given to each flip flop. It is also called as parallel counter.

Synchronous counters

- Synchronous(parallel) counters
  - All of the FFs are triggered simultaneously by the clock input pulses.
  - All FFs change at same time
- Remember
  - If J=K=0, flop maintains value
  - If J=K=1, flop toggles
- Most counters are synchronous in computer systems.
- Can also be made from D flops
- Value increments on positive edge

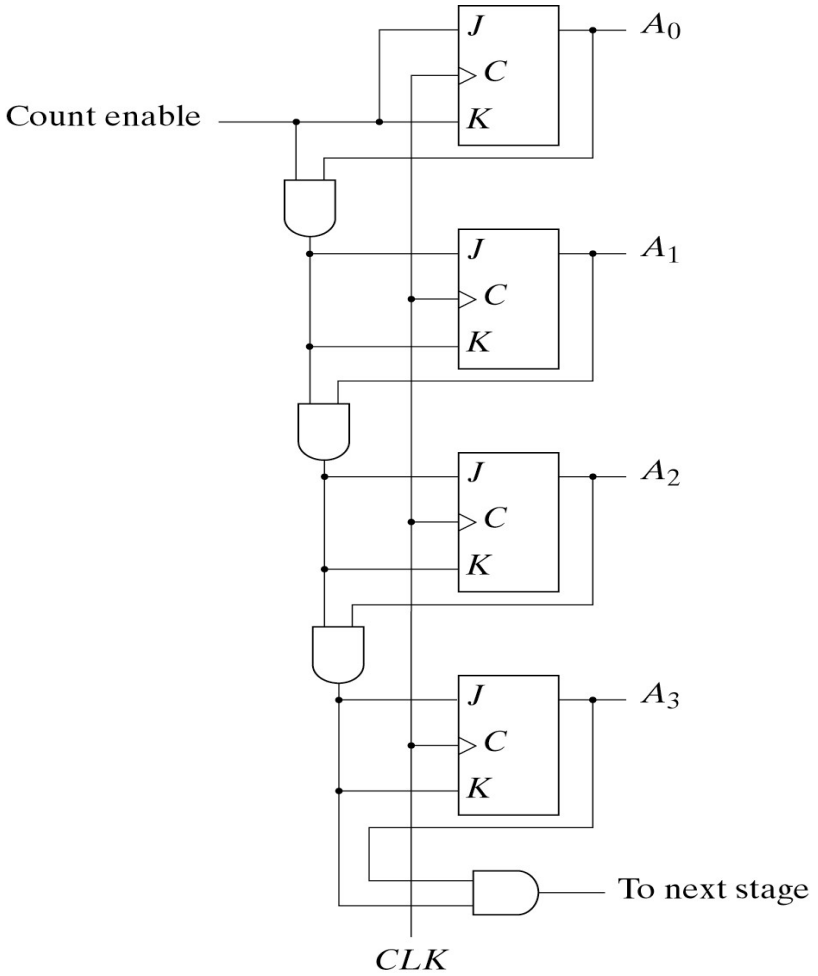
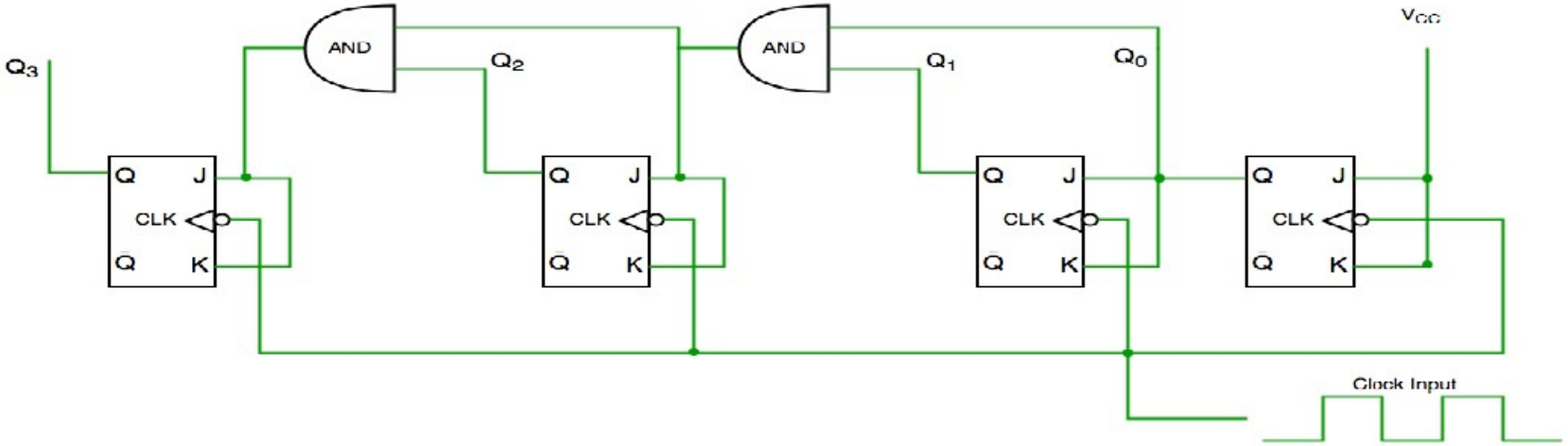


Fig. 6-12 4-Bit Synchronous Binary Counter

Synchronous counters

- Same counter as previous slide except Count enable replaced by J=K=1
- Note that clock signal is a square wave
- Clock fans out to all clock inputs



Circuit operation

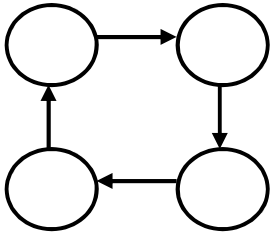
Count	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.
.	.	.	.	.
.	.	etc.	.	.

(b)

- negative edge
- (A)

Synchronous (Parallel) Counters

- Synchronous (parallel) counters: the flip-flops are clocked at the same time by a common clock pulse.
- We can design these counters using the sequential logic design process .
- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

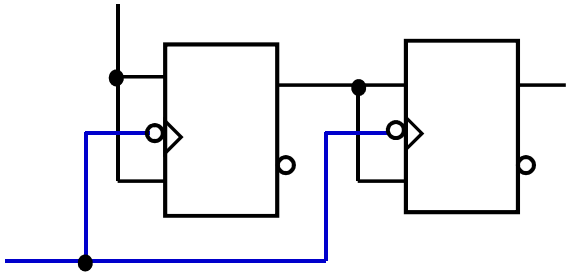


Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

Synchronous (Parallel) Counters

- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

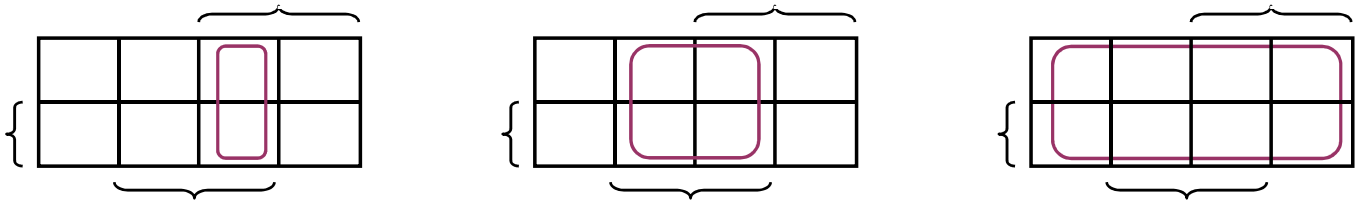
Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1



Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$