Digital Circuit
&
Logic Design
(CSPC-201)

Registers and Counters

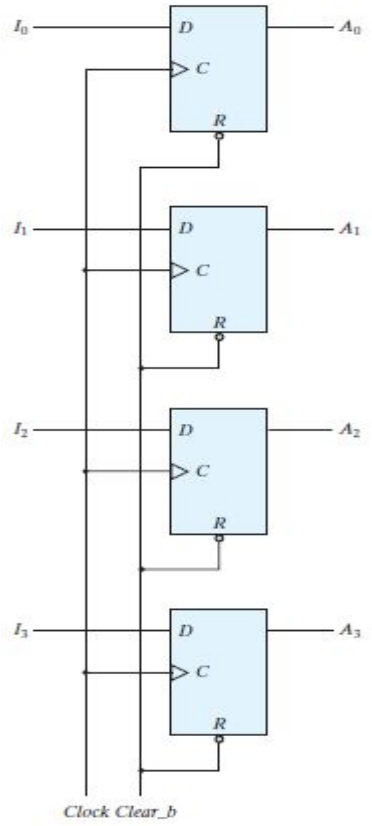
A clocked sequential circuit consists of a group of flip-flops and combinational gates.

A circuit with flip-flops is considered a sequential circuit even in the absence of combinational gates.

A **register** is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information. An n-bit register consists of a group of n flip-flops capable of storing n bits of binary information.

In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.

A **counter** is essentially a register that goes through a predetermined sequence of binary states. The gates in the counter are connected in such a way as to produce the prescribed sequence of states.



Four-bit register

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Register with Parallel Load

Registers with parallel load are a fundamental building block in digital systems.

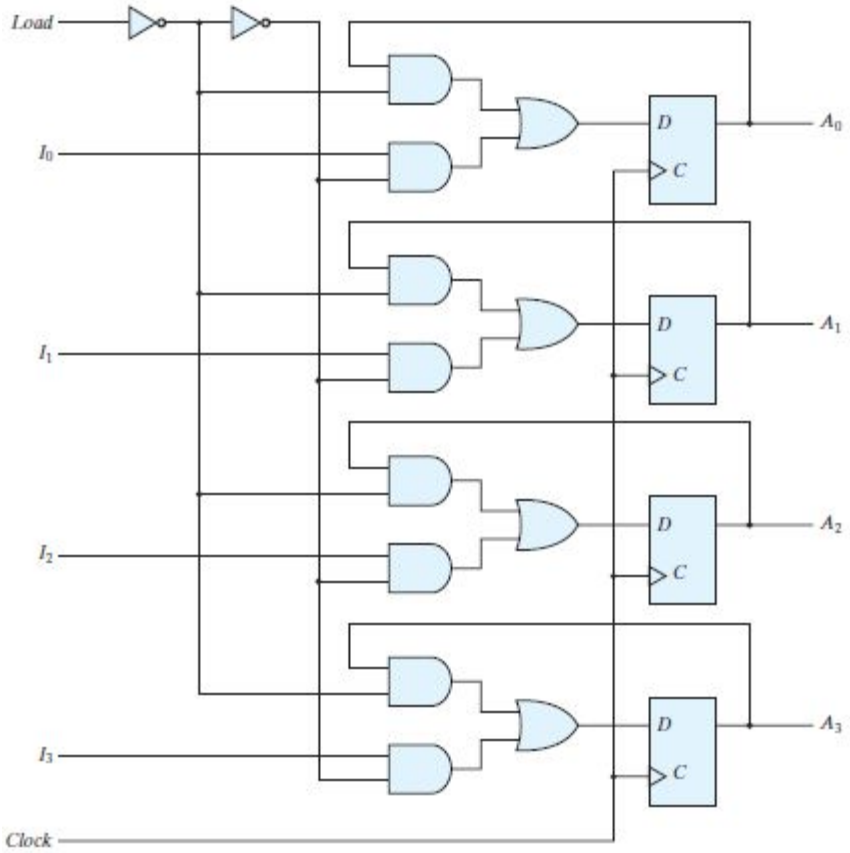
Synchronous digital systems have a master clock generator that supplies a continuous train of clock pulses. The pulses are applied to all flip-flops and registers in the system. The master clock acts like a drum that supplies a constant beat to all parts of the system.

A separate control signal must be used to decide which register operation will execute at each clock pulse.

The transfer of new information into a register is referred to as *loading or updating* the register. If all the bits of the register are loaded simultaneously with a common clock pulse, we say that the loading is done *in parallel* .

To fully synchronize the system, we must ensure that all clock pulses arrive at the same time anywhere in the system, so that all flip-flops trigger simultaneously. Performing logic with clock pulses inserts variable delays and may cause the system to go out of synchronism.

For this reason, it is advisable to control the operation of the register with the *D inputs, rather* than controlling the clock in the *C inputs of the flip-flops. This creates the effect of a* gated clock, but without affecting the clock path of the circuit



Four-bit register with parallel load

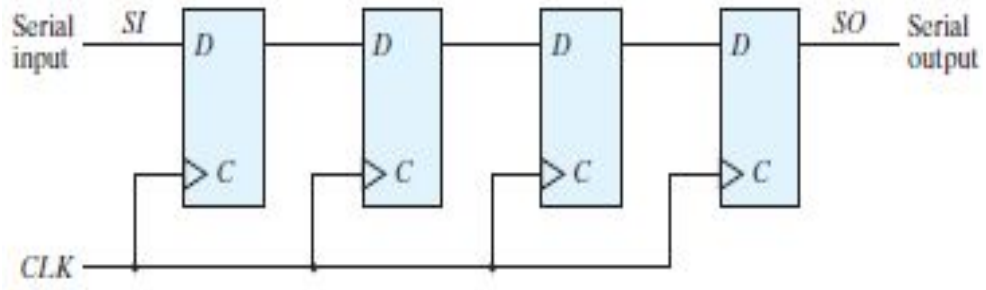
SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Shift Registers

A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a *shift register.*

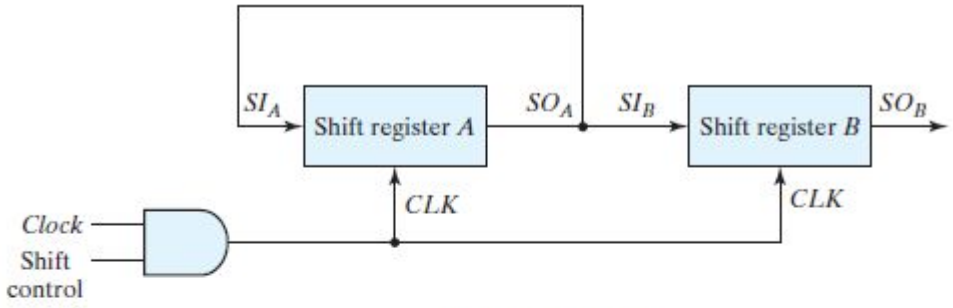
The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop.

All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next.

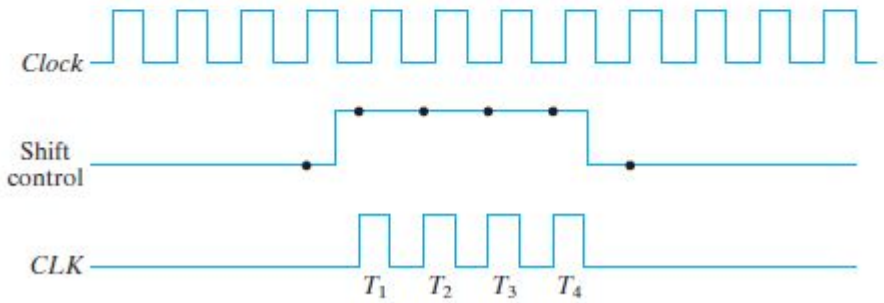


Four-bit shift register
(left to right shift)

Serial Transfer



(a) Block diagram



(b) Timing diagram

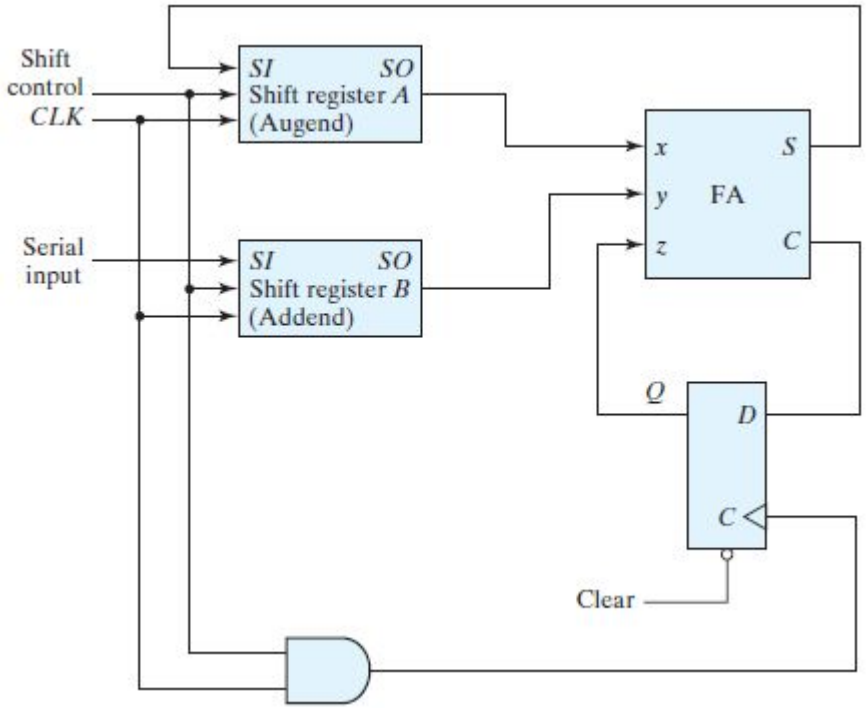
Serial-Transfer Example

Timing Pulse	Shift Register *A*				Shift Register *B*			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

Serial transfer from register A to register B

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Serial Addition

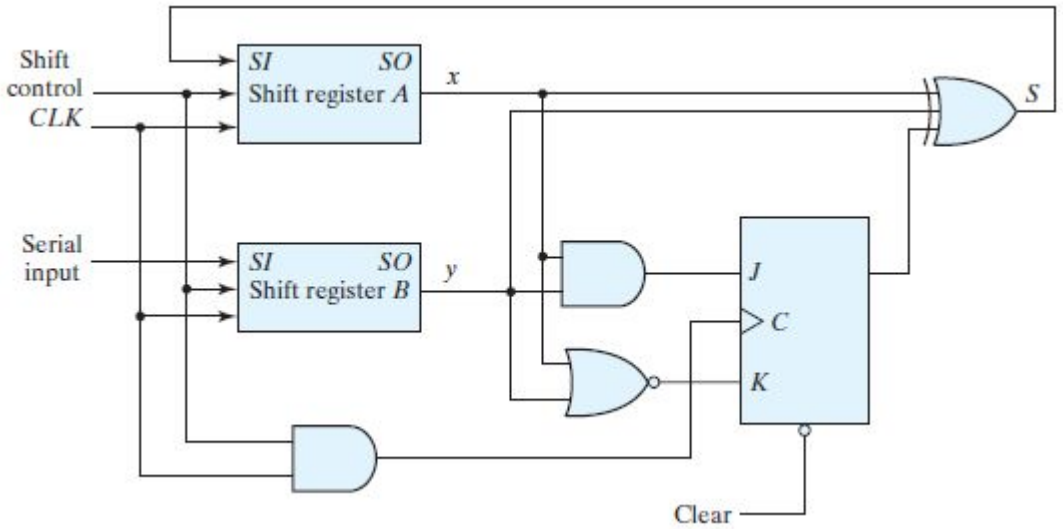


State Table for Serial Adder

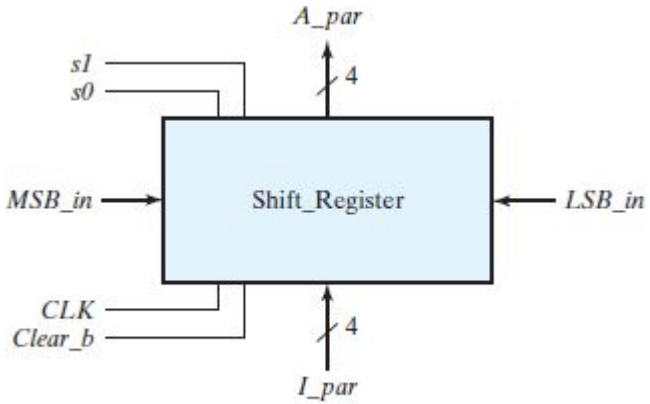
Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Second form of serial adder

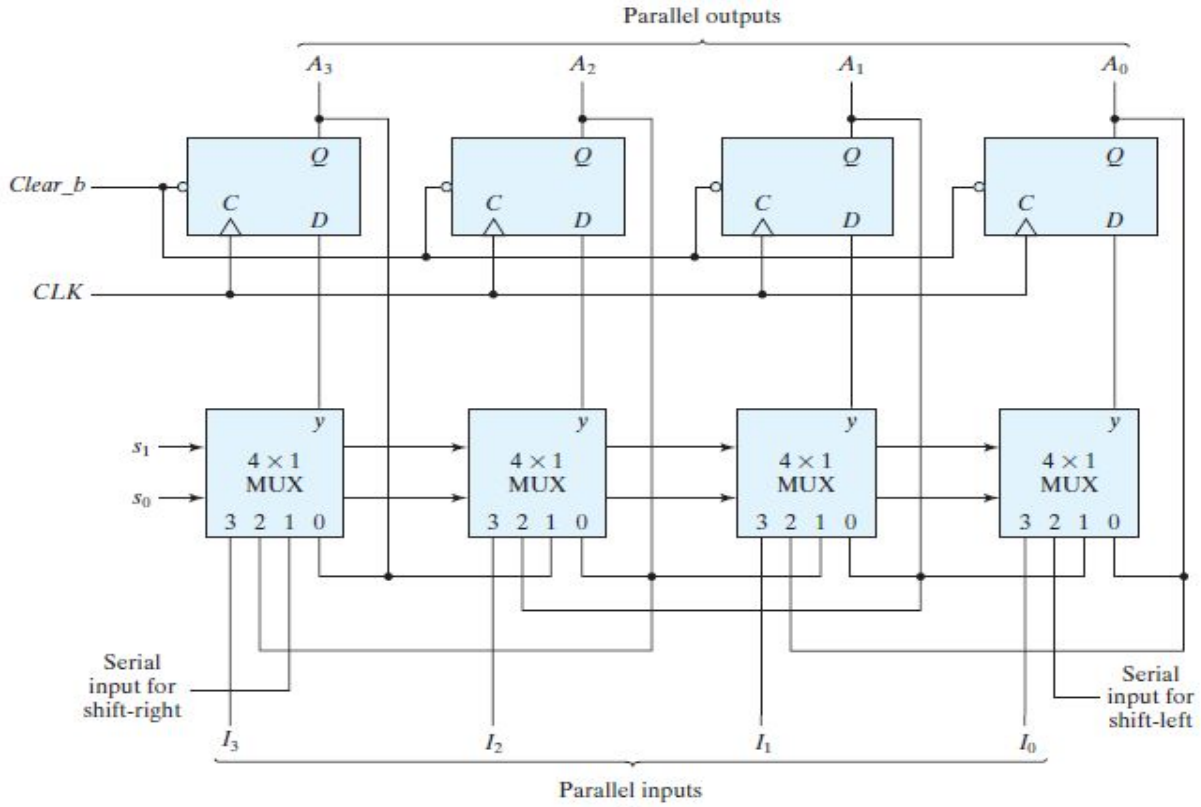


Universal Shift Register



(a)

Mode Control		
s_1	s_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load



(b)

A register capable of shifting in one direction only is a **unidirectional shift register**. One that can shift in both directions is a **bidirectional shift register**. If the register has both shifts and parallel-load capabilities, it is referred to as a **universal shift register**.

The most general shift register has the following capabilities:

1. A clear control to clear the register to 0.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.
4. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift left.
5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. n parallel output lines.
7. A control state that leaves the information in the register unchanged in response to the clock. Other shift registers may have only some of the preceding functions, with at least one shift operation.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Ripple Counter

A register that goes through a prescribed sequence of states upon the application of input pulses is called a *counter*.

The sequence of states may follow the binary number sequence or any other sequence of states. A counter that follows the binary number sequence is called a binary counter . An n-bit binary counter consists of n flip-flops and can count in binary from 0 through 2^n - 1.

Counters are available in two categories: *ripple counters and synchronous counters*.

In a **ripple counter**, a flip-flop output transition serves as a source for triggering other flip-flops. In other words, the C input of some or all flip-flops are triggered, not by the common clock pulses, but rather by the transition that occurs in other flip-flop outputs.

In a **synchronous counter**, the C inputs of all flip-flops receive the common clock.

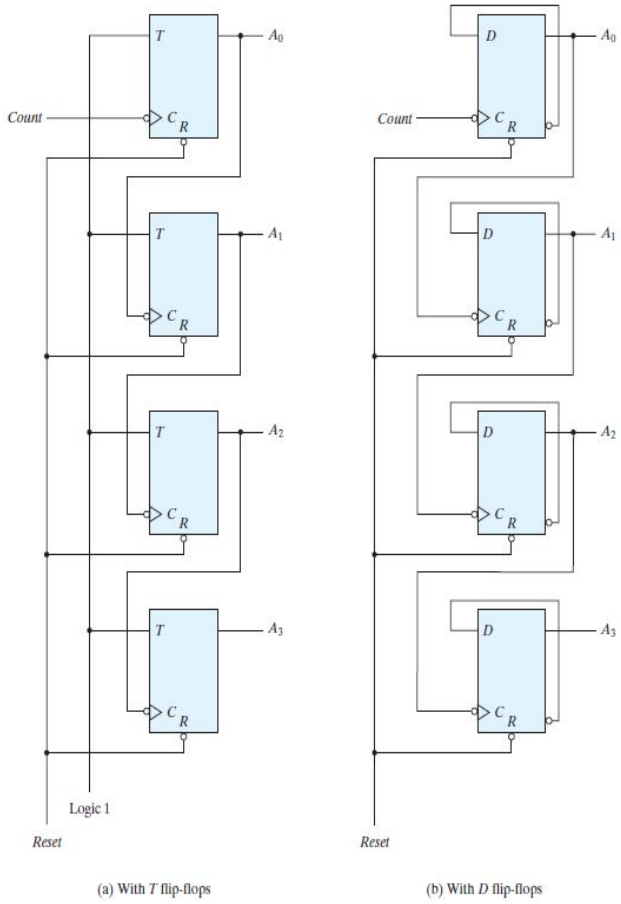
SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the C input of the next higher order flip-flop.

The flip-flop holding the least significant bit receives the incoming count pulses. A complementing flip-flop can be obtained from a JK flip-flop with the J and K inputs tied together or from a T flip-flop. A third possibility is to use a D flip-flop with the complement output connected to the D input. In this way, the D input is always the complement of the present state, and the next clock pulse will cause the flip-flop to complement.

A binary counter with a reverse count is called a *binary countdown counter*.



(a) With *T* flip-flops

(b) With *D* flip-flops

A_3	A_2	A_1	A_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

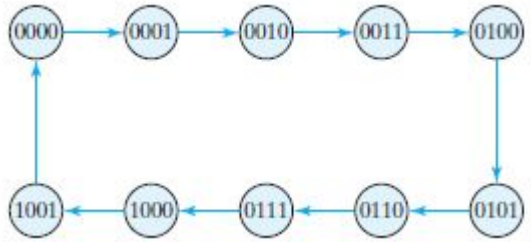
SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

BCD Ripple Counter

A decimal counter follows a sequence of 10 states and returns to 0 after the count of 9.

Such a counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits. The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit.

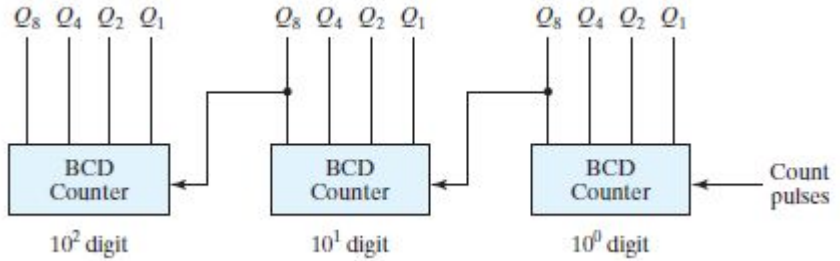
A decimal counter is similar to a binary counter, except that the state after 1001 (the code for decimal digit 9) is 0000 (the code for decimal digit 0).



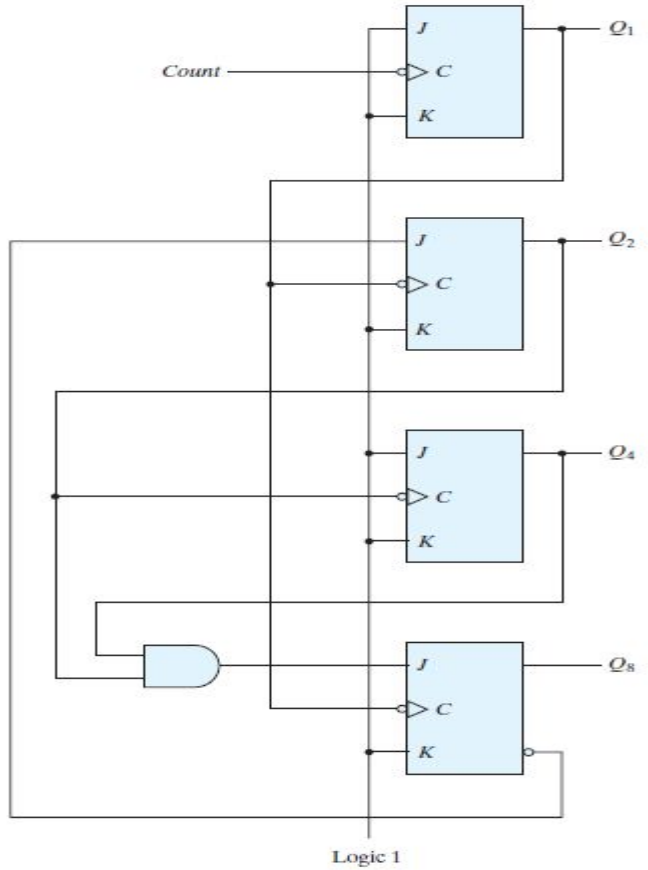
SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

A ripple counter is an asynchronous sequential circuit. Signals that affect the flip-flop transition depend on the way they change from 1 to 0. The operation of the counter can be explained by a list of conditions for flip-flop transitions. These conditions are derived from the logic diagram and from knowledge of how a JK flip-flop operates.

Remember that when the C input goes from 1 to 0, the flip-flop is set if J = 1, is cleared if K = 1, is complemented if J = K = 1, and is left unchanged if J = K = 0.



Three-decade decimal BCD counter



BCD ripple counter

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Synchronous Counters

Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.

A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter.

The decision whether a flip-flop is to be complemented is determined from the values of the data inputs, such as T or J and K at the time of the clock edge. If T = 0 or J = K = 0, the flip-flop does not change state. If T = 1 or J = K = 1, the flip-flop complements.

The synchronous counter can be triggered with either the positive or the negative clock edge.

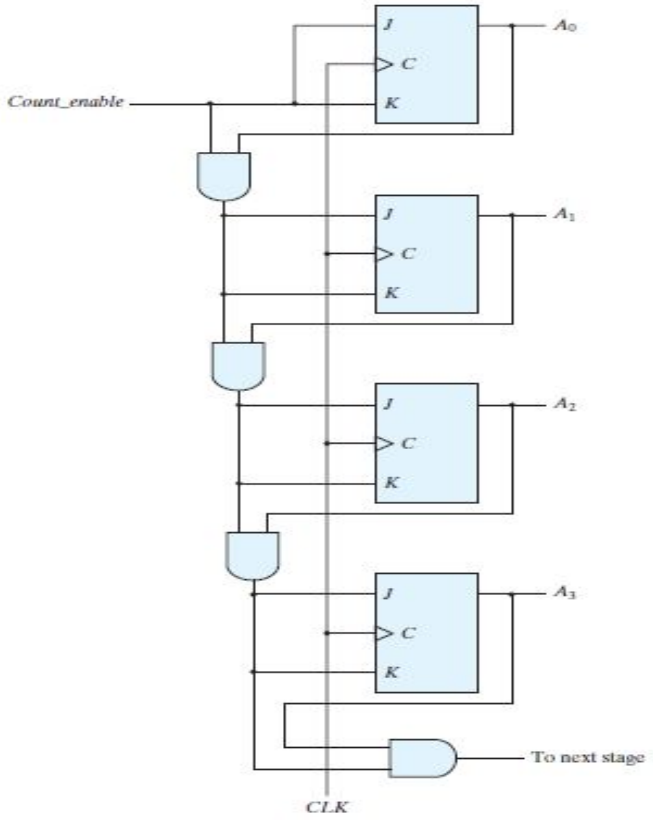
SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Binary Counter

In a synchronous binary counter, the flip-flop in the least significant position is complemented with every pulse. *A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1*.

Synchronous binary counters have a regular pattern and can be constructed with complementing flip-flops and gates.

The complementing flip-flops in a binary counter can be of either the JK type, the T type, or the D type with XOR gates.

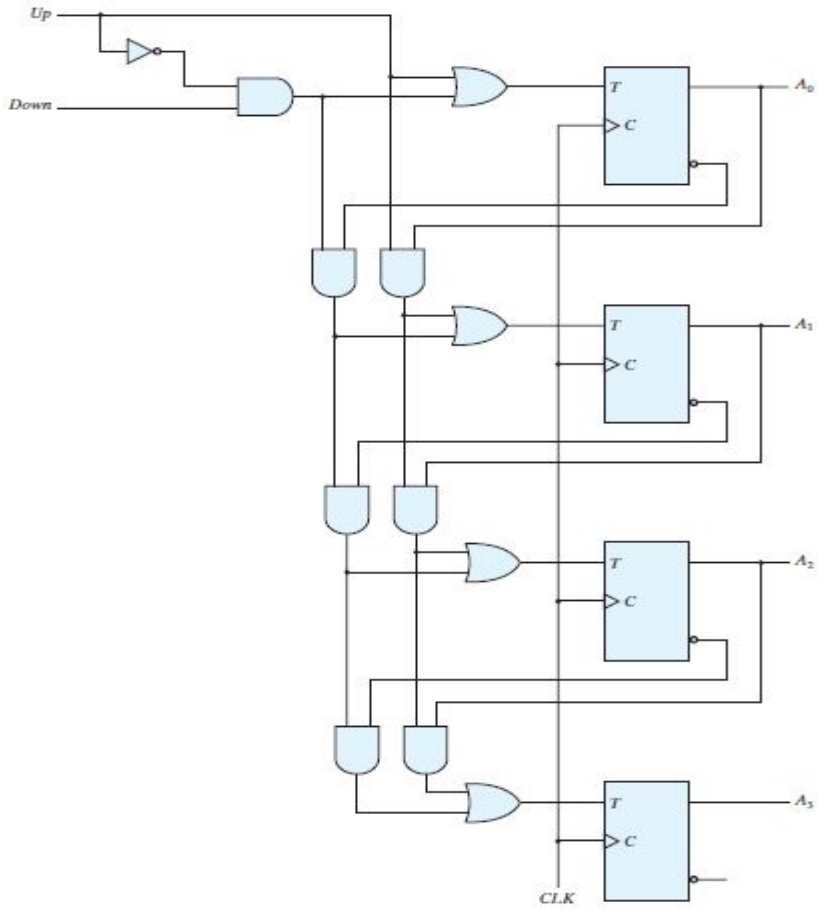


SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Up–Down Binary Counter

A synchronous countdown binary counter goes through the binary states in reverse order, from 1111 down to 0000 and back to 1111 to repeat the count. It is possible to design a countdown counter in the usual manner, but the result is predictable by inspection of the downward binary count. The bit in the least significant position is complemented with each pulse. A bit in any other position is complemented if all lower significant bits are equal to 0.

The least significant bit is always complemented. The second significant bit is complemented because the first bit is 0. The third significant bit is complemented because the first two bits are equal to 0. But the fourth bit does not change, because not all lower significant bits are equal to 0.



SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

BCD Counter

A BCD counter counts in binary-coded decimal from 0000 to 1001 and back to 0000.

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

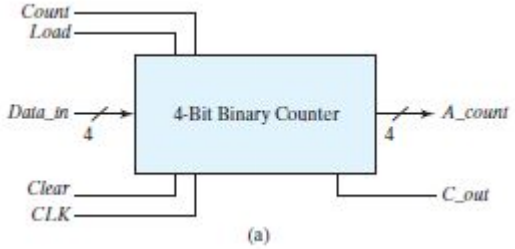
$$T_{Q1} = 1$$
$$T_{Q2} = Q'_8 Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

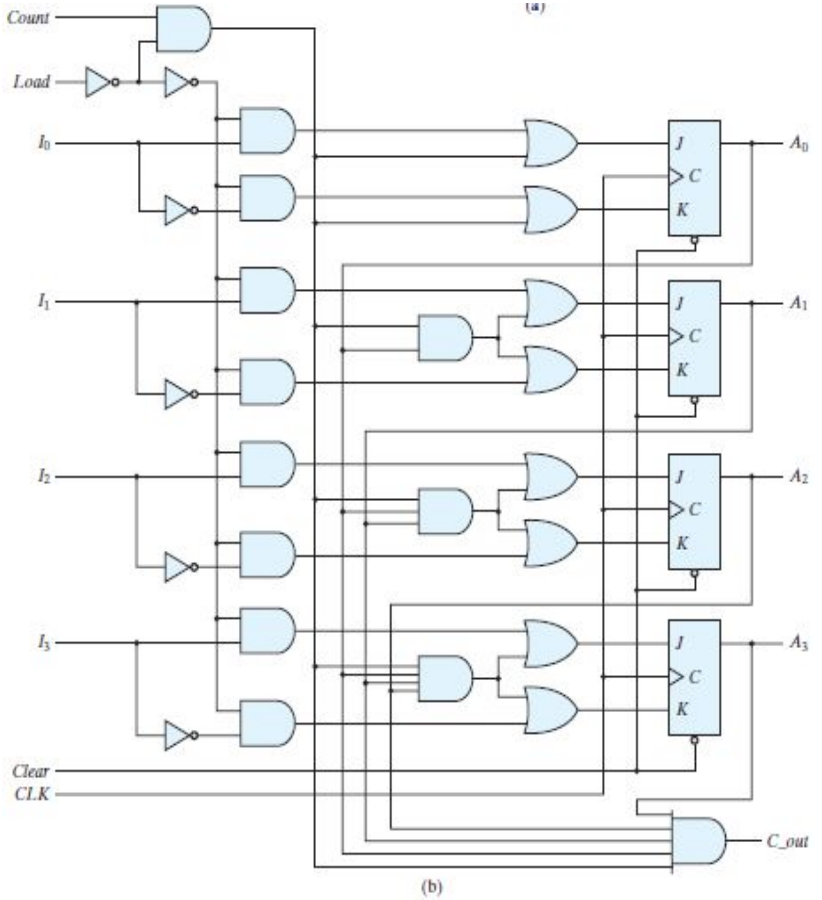
Binary Counter with Parallel Load

Counters employed in digital systems quite often require a parallel-load capability for transferring an initial binary number into the counter prior to the count operation.

When equal to 1, the input load control disables the count operation and causes a transfer of data from the four data inputs into the four flip-flops. If both control inputs are 0, clock pulses do not change the state of the register.



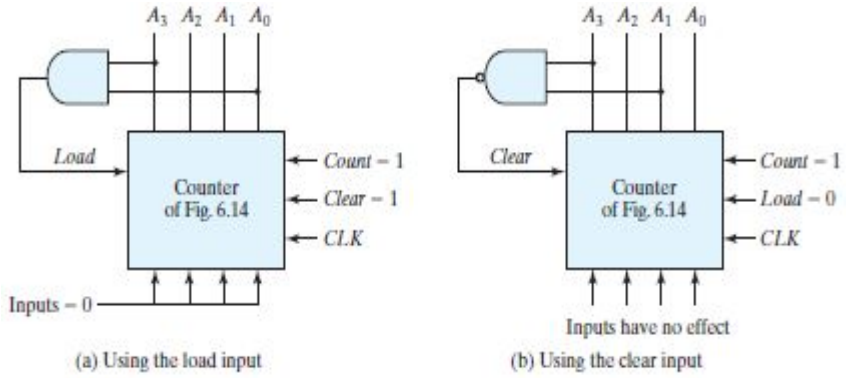
Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change



SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

The AND gate in Fig. (a) detects the occurrence of state 1001. The counter is initially cleared to 0, and then the Clear and Count inputs are set to 1, so the counter is active at all times. As long as the output of the AND gate is 0, each positive-edge clock increments the counter by 1. When the output reaches the count of 1001, both A0 and A3 become 1, making the output of the AND gate equal to 1. This condition activates the Load input; therefore, on the next clock edge the register does not count, but is loaded from its four inputs. Since all four inputs are connected to logic 0, an all-0's value is loaded into the register following the count of 1001. Thus, the circuit goes through the count from 0000 through 1001 and back to 0000, as is required in a BCD counter.

In Fig. (b), the NAND gate detects the count of 1010, but as soon as this count occurs, the register is cleared. The count 1010 has no chance of staying on for any appreciable time, because the register goes immediately to 0. A momentary spike occurs in output A0 as the count goes from 1010 to 1011 and immediately to 0000. The spike may be undesirable, and for that reason, this configuration is not recommended. If the counter has a synchronous clear input, it is possible to clear the counter with the clock after an occurrence of the 1001 count.



Two ways to achieve a BCD counter using a counter with parallel load

Error Detection and Correction

Hamming Codes

Hamming code is a set of error-correction codes that can be used to **detect and correct the errors** that can occur when the data is moved or stored from the sender to the receiver.

Redundant bits: Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.
The number of redundant bits can be calculated using the following formula:

$2^r \geq m + r + 1$ where, r = redundant bit, m = data bit

Parity Bits: A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd.

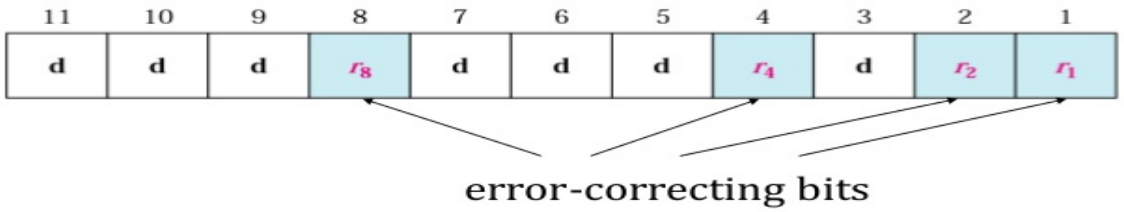
The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
   Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
   Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
   etc.

4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

SOURCE: INTERNET



Example of redundancy bit calculation

d	d	d	r8	d	d	d	r4	d	r2	r1

r1=1,3,5,7,9,11

r2=2,3,6,7,10,11

r4=4,5,6,7

r8=8,9,10,11

Input Data 1001101

	11	10	9	8	7	6	5	4	3	2	1
Adding r1	1	0	0	r8	1	1	0	r4	1	r2	1
Adding r2	1	0	0	r8	1	1	0	r4	1	0	1
Adding r4	1	0	0	r8	1	1	0	0	1	0	1
Adding r8	1	0	0	1	1	1	0	0	1	0	1

Output data:10011100101

example

Error detecting using hamming code

error

	11	10	9	8	7	6	5	4	3	2	1	
	1	0	0	1	0	1	0	0	1	0	1	
r1=1												
	1	0	0	1	0	1	0	0	1	0	1	
r2=1												
	1	0	0	1	0	1	0	0	1	0	1	
r4=1												
	1	0	0	1	0	1	0	0	1	0	1	
r8=0												

If no.1,s is even 0

If no.1,s is odd 1

8 4 2 1

0 1 1 1

7

It mean the 7 bit is corrupted

- The Hamming code can detect and correct only a single error.
- By adding another parity bit to the coded word, the Hamming code can be used to correct a single error and detect double errors.
- If we include this additional parity bit, then the previous 12-bit coded word becomes 001110010100P_{13}, where P_{13} is evaluated from the exclusive-OR of the other 12 bits. This produces the 13-bit word 0011100101001 (even parity).
- When the 13-bit word is read from memory, the check bits are evaluated, as is the parity P over the entire 13 bits. If $P = 0$, the parity is correct (even parity), but if $P = 1$, then the parity over the 13 bits is incorrect (odd parity).

If $C = 0$ and $P = 0$, no error occurred.
If $C \neq 0$ and $P = 1$, a single error occurred that can be corrected.
If $C \neq 0$ and $P = 0$, a double error occurred that is detected, but that cannot be corrected.
If $C = 0$ and $P = 1$, an error occurred in the P_{13} bit.

Master-Slave Flip Flop

Race Around Condition

- For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain
- Can be avoided by ensuring that the clock input is at logic “1” only for a very short time.

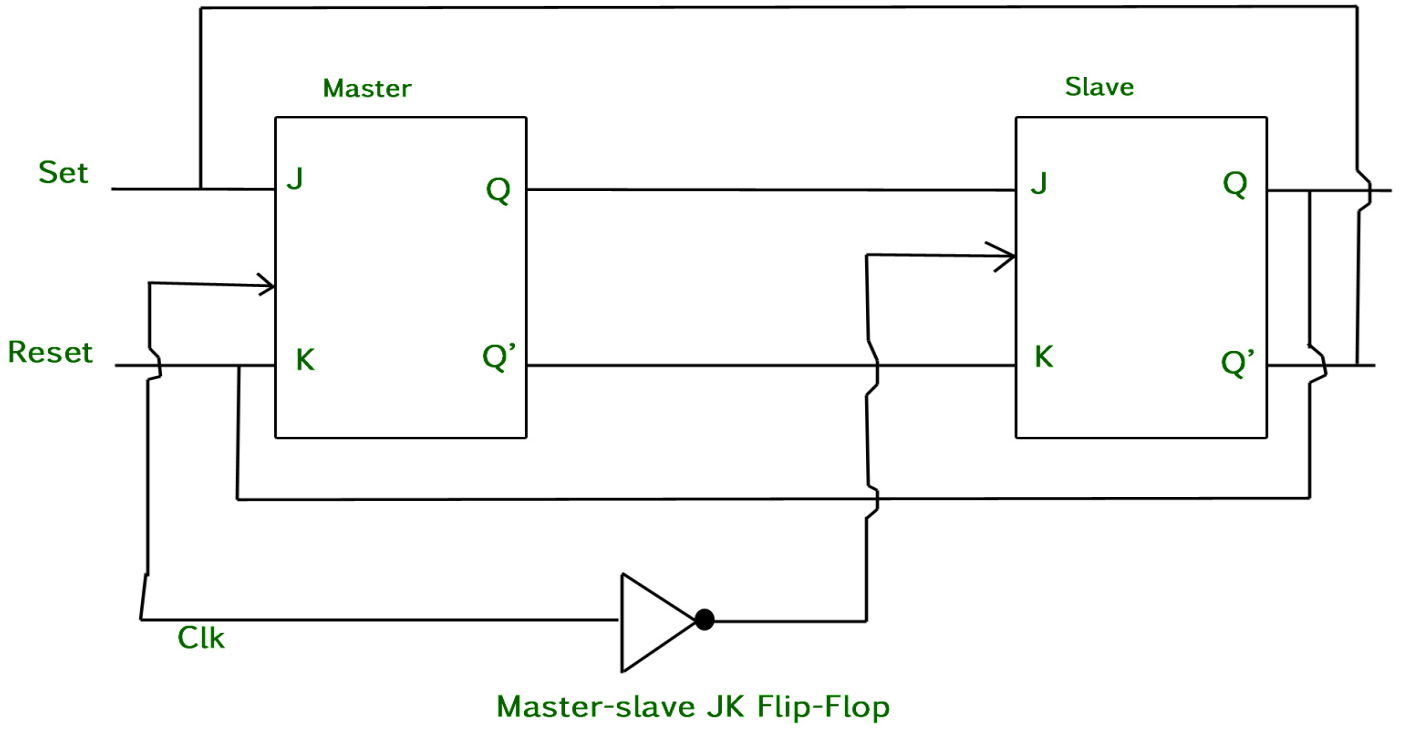
Solution:-

Master Slave JK flip flop

- The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration.
- Out of these, one acts as the “master” and the other as a “slave”. The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.
- In addition to these two flip-flops, the circuit also includes an inverter. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop.
- In other words if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Master-Slave Flip Flop



Master-slave JK Flip-Flop

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1

Working of a master slave flip flop –

1. When the clock pulse goes to 1, the slave is isolated; J and K inputs may affect the state of the system. The slave flip-flop is isolated until the CP goes to 0. When the CP goes back to 0, information is passed from the master flip-flop to the slave and output is obtained.
2. Firstly the master flip flop is positive level triggered and the slave flip flop is negative level triggered, so the master responds before the slave.
3. If J=0 and K=1, the high Q' output of the master goes to the K input of the slave and the clock forces the slave to reset, thus the slave copies the master.
4. If J=1 and K=0, the high Q output of the master goes to the J input of the slave and the Negative transition of the clock sets the slave, copying the master.
5. If J=1 and K=1, it toggles on the positive transition of the clock and thus the slave toggles on the negative transition of the clock.
6. If J=0 and K=0, the flip flop is disabled and Q remains unchanged.

SOURCE: HTTPS://DOI.ORG/10.1007/978-3-319-66775-1_1