Chapter-7

Registers, Counters and Memory units

Introduction

A circuit with only flip-flops is considered a sequential circuit even in the absence of combinational gates. Certain MSI circuits that include flip-flops are classified by the operation that they perform rather than the name sequential circuit. Two such MSI components are registers and counters.

Registers

A register is a group of binary cells suitable for holding binary information. A group of flip-flops constitutes a register.

An n-bit register has a group of n flip-flops and is capable of storing any binary information containing n bits.

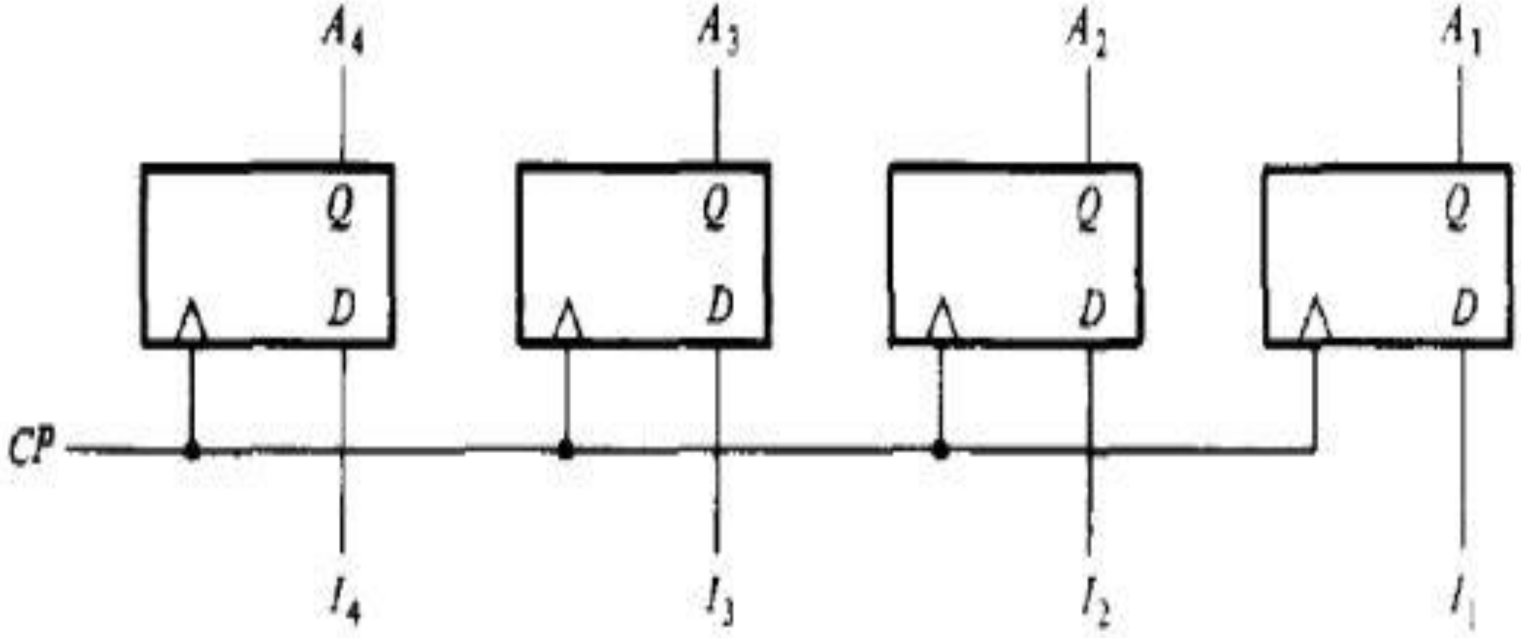
In addition to the flip-flops, a register may have combinational gates that perform certain data processing tasks.

Various types of registers are available in MSI circuits. The simplest possible register is one that consists of only flip-flops without any external gates. Following fig. shows such a register constructed with four D-type flip-flops and a common clock-pulse input.

The clock pulse input, CP, enables all flip-flops, so that the information presently available at the four inputs can be transferred into the 4-bit register.

The four outputs can be sampled to obtain the information presently stored in the register.

4-bit Register



parallel load and shift register

The transfer of new information into a register is referred to as loading the register. If all the bits of the register are loaded simultaneously with a single clock pulse, we say that the loading is done in parallel. A pulse applied to the CP input of the register of Fig. above will load all four inputs in parallel. When CP goes to 1, the input information is loaded into the register. If CP remains at 0, the content of the register is not changed. Note that the change of state in the outputs occurs at the positive edge of the pulse.

Shift Registers

A register capable of shifting its binary information either to the right or to the left is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip flop. All flip-flops receive a common clock pulse that causes the shift from one stage to the next.

The Shift Register is used for data storage or data movement and are used in calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock signal making them synchronous devices. Shift register IC's are generally provided with a clear or reset connection so that they can be "SET" or "RESET" as required.

Mode of operation of shift reg.

Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.

Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.

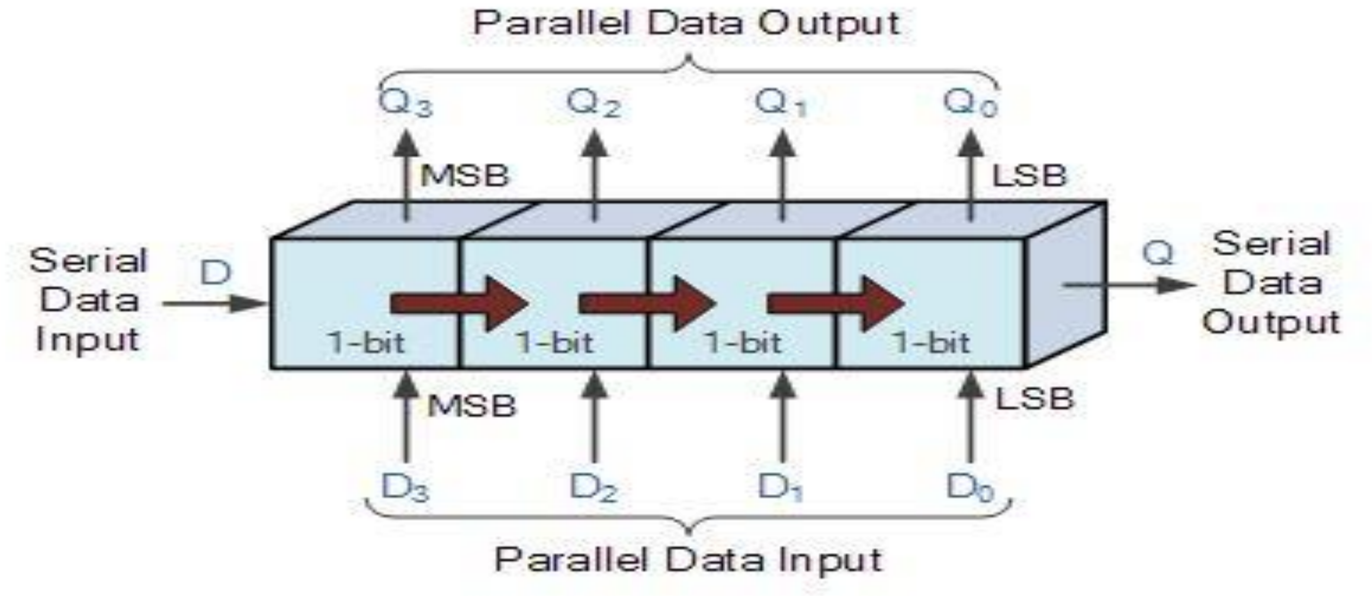
Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.

Parallel-in to parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

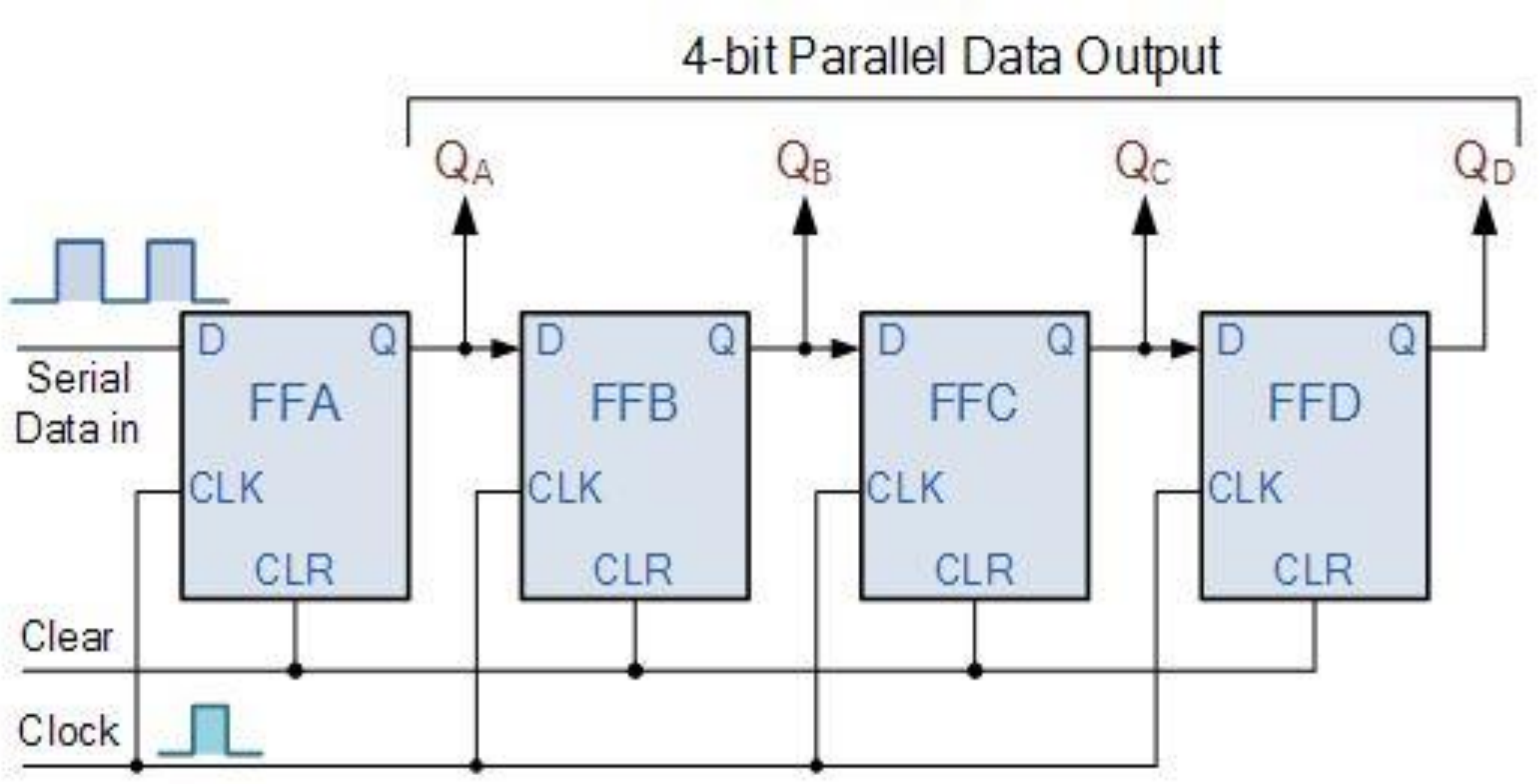
The effect of data movement from left to right through a shift register can be presented graphically as:

Mode of operation of shift reg.

Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it bidirectional. In this tutorial it is assumed that all the data shifts to the right, (right shifting).



Serial-in to Parallel-out Shift Reg



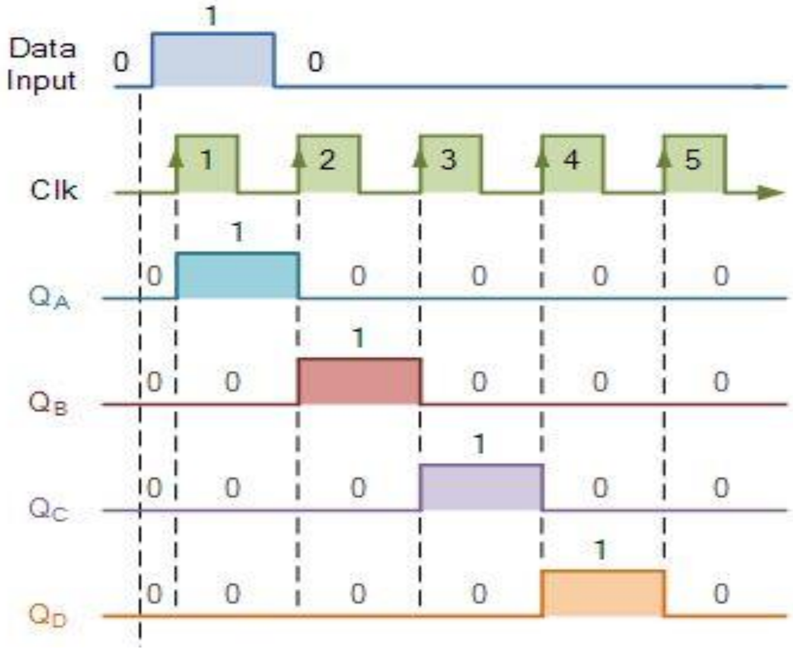
Serial-in to Parallel-out Shift Reg

Operation

- Let’s assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs QA to QD are at logic level "0" i.e., no parallel data output.
- If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting QA will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0".
- Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.
- The second clock pulse will change the output of FFA to logic "0" and the output of FFB and QB HIGH to logic "1" as its input D has the logic "1" level on it from QA. The logic "1" has now moved or been "shifted" one place along the register to the right as it is now at QA. When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q) and so on until the arrival of the fifth clock pulse which sets all the outputs QA to QD back again to logic level "0" because the input to FFA has remained constant at logic level "0".
- The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register.

Serial-in to Parallel-out Shift Reg.

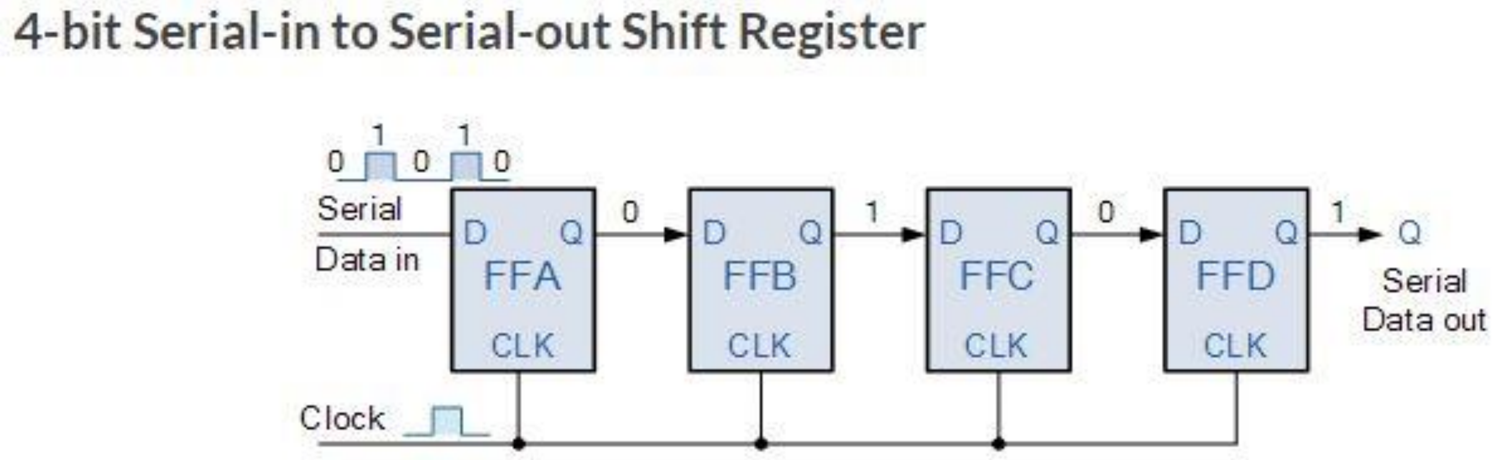
Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0



Serial-in to Serial-out (SISO)

This shift register is very similar to the SIPO above, except were before the data was read directly in a parallel form from the outputs QA to QD, this time the data is allowed to flow straight through the register and out of the other end. Since there is only one output, the DATA leaves the shift register one bit at a time in a serial pattern, hence the name Serial-in to Serial-Out Shift Register or SISO.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal . The logic circuit diagram below shows a generalized serial-in serial-out shift register.

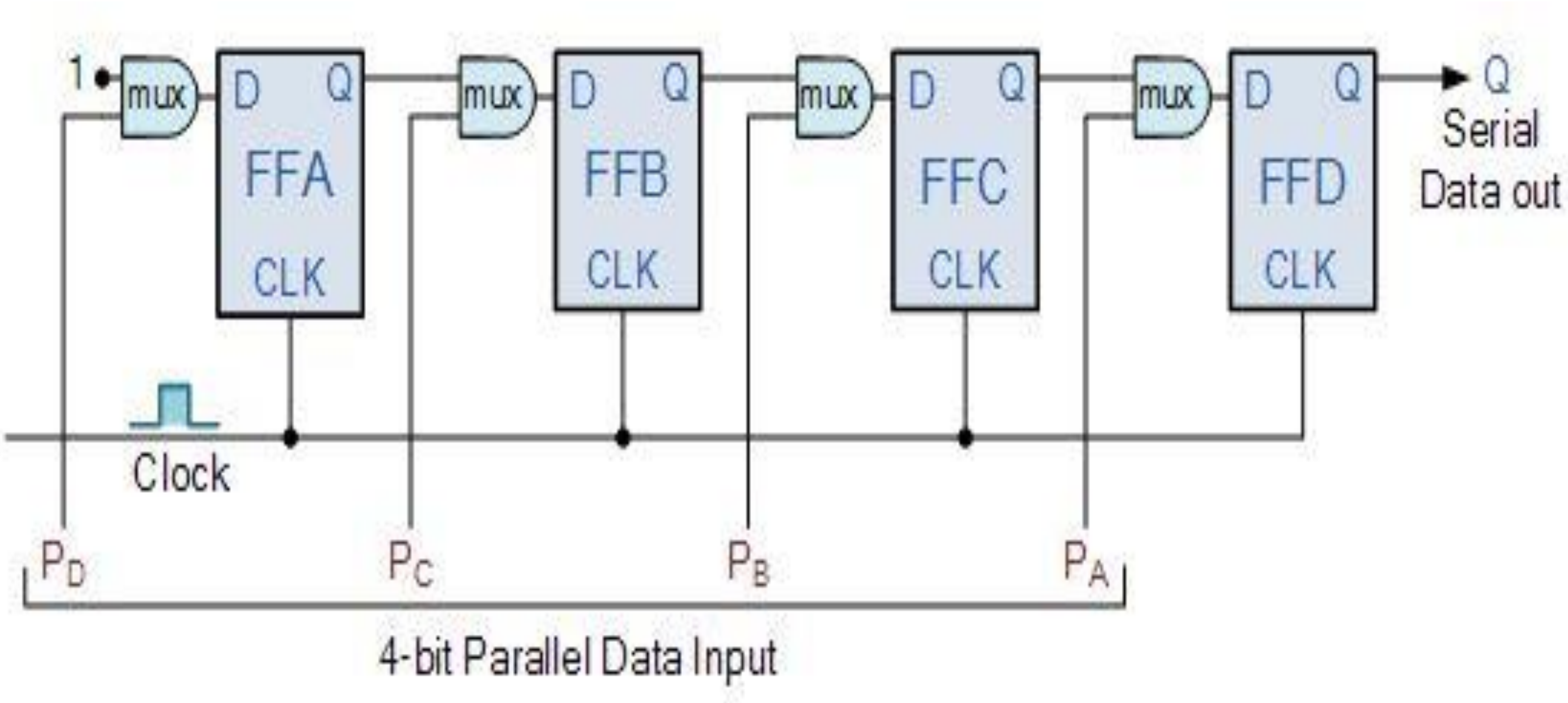


Parallel-in to Serial-out (PISO)

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format i.e. all the data bits enter their inputs simultaneously, to the parallel input pins PA to PD of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at PA to PD. This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this system a clock pulse is not required to parallel load the register as it is already present, but four clock pulses are required to unload the data.

As this type of shift register converts parallel data, such as an 8-bit data word into serial format, it can be used to multiplex many different input lines into a single serial DATA stream which can be sent directly to a computer or transmitted over a communications line.

Parallel-in to Serial-out (PISO)

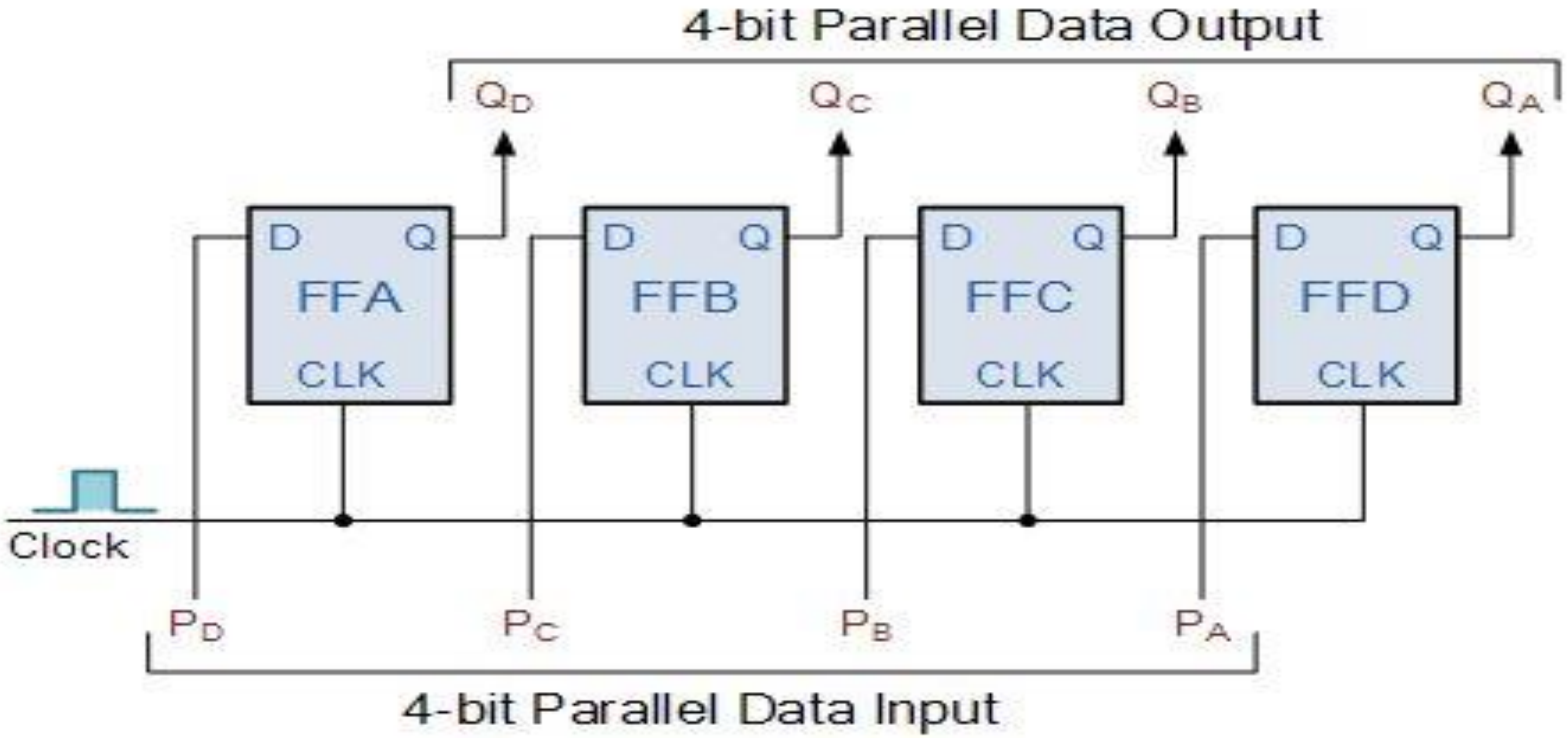


Parallel-in to Parallel-out (PIPO)

The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. The data is presented in a parallel format to the parallel input pins PA to PD and then transferred together directly to their respective output pins QA to QD by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk). Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

Parallel-in to Parallel-out (PIPO)



Ripple Counters (Asynchronous Counters)

- MSI counters come in two categories: **ripple counters and synchronous counters.**
- In a ripple counter (Asynchronous Counter); flip-flop output transition serves as a source for triggering other flip-flops. In other words, the CP inputs of all flip-flops (except the first) are triggered not by the incoming pulses, but rather by the transition that occurs in other flip-flops.
- Synchronous counter, the input pulses are applied to all CP inputs of all flip-flops. The change of state of a particular flip-flop is dependent on the present state of other flip-flops.

Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing flip-flops (T or JK type), with the output of each flip-flop connected to the CP input of the next higher-order flip-flop. The flip-flop holding the least significant bit receives the incoming count pulses. The diagram of a 4-bit binary ripple counter is shown in Fig. below. All J and K inputs are equal to 1.

Binary Ripple Counter

All J and K inputs are equal to 1. The small circle in the CP input indicates that the flip-flop complements during a negative-going transition or when the output to which it is connected goes from 1 to 0.

To understand the operation of the binary counter, refer to its count sequence given in Table. It is obvious that the lowest-order bit A1 must be complemented with each count pulse. Every time A1 goes from 1 to 0, it complements A2. Every time A2 goes from 1 to 0, it complements A3, and so on.

For example: take the transition from count 0111 to 1000. The arrows in the table emphasize the transitions in this case. A1 is complemented with the count pulse. Since A1 goes from 1 to 0, it triggers A2 and complements it. As a result, A2 goes from 1 to 0, which in turn complements A3. A3 now goes from 1 to 0, which complements A4. The output transition of A4, if connected to a next stage, will not trigger the next flip-flop since it goes from 0 to 1. The flip-flops change one at a time in rapid succession, and the signal propagates through the counter in a ripple fashion

Binary Ripple Counter

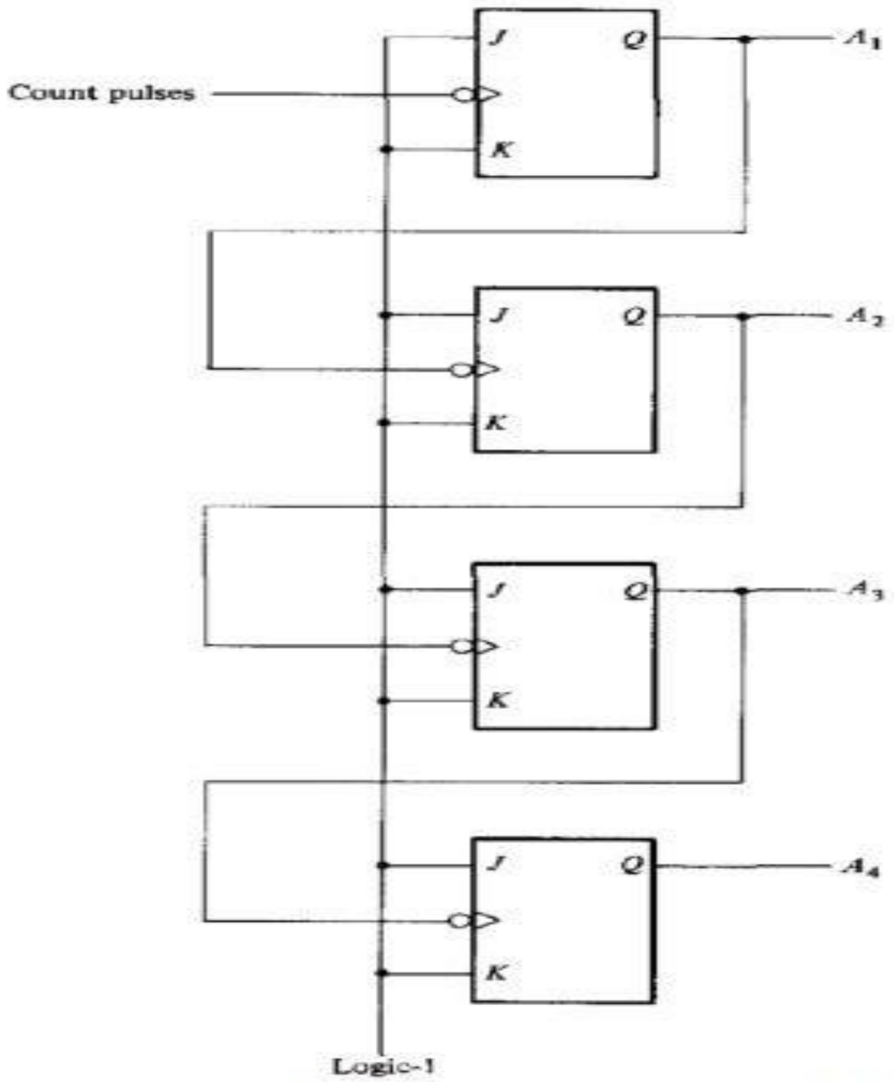


Fig: 4-bit binary ripple counter

Binary Ripple Counter

Count Sequence				Conditions for Complementing Flip-Flops	
A_4	A_3	A_2	A_1		
0	0	0	0	Complement A_1	
0	0	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	0	1	0	Complement A_1	
0	0	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3
0	1	0	0	Complement A_1	
0	1	0	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2
0	1	1	0	Complement A_1	
0	1	1	1	Complement A_1	A_1 will go from 1 to 0 and complement A_2; A_2 will go from 1 to 0 and complement A_3; A_3 will go from 1 to 0 and complement A_4
1	0	0	0	and so on . . .	

BCD Ripple Counter (Decade Counter)

BCD Counter - Digital counters count upwards from zero to some pre-determined count value on the application of a clock signal. Once the count value is reached, resetting them returns the counter back to zero to start again.

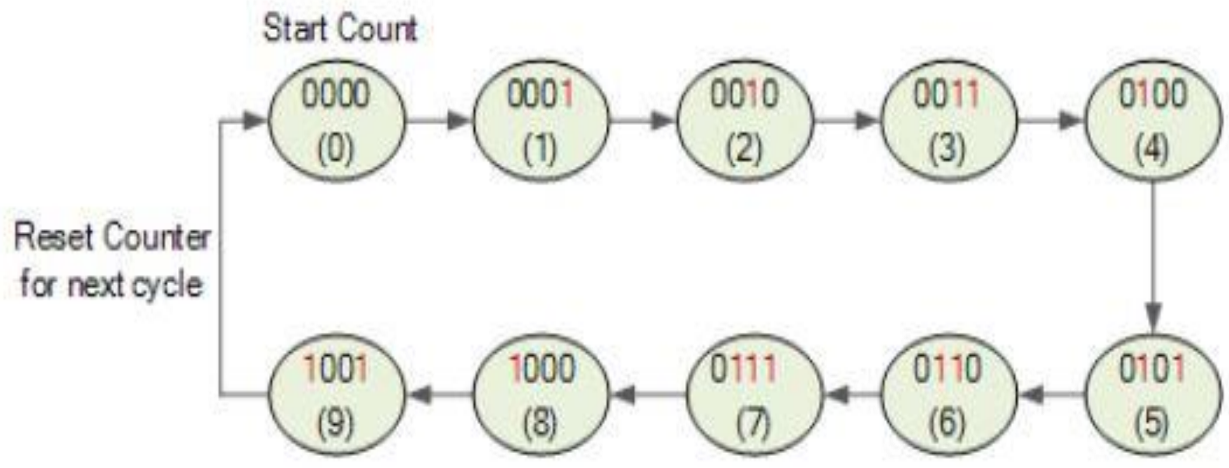
A counter which resets after ten counts with a divide-by-10 count sequence from binary 0000 (decimal "0") through to 1001 (decimal "9") is called a "binary-coded-decimal counter" or BCD Counter for short and a MOD-10 counter can be constructed using a minimum of four toggle flip-flops. BCD Counter is a devices that goes through a sequence of ten states when it is clocked and returns to 0 after the count of 9.BCD Counter is a devices that goes through a sequence of ten states when it is clocked and returns to 0 after the count of 9.

A BCD counter counts in a sequence of ten and then returns back to zero after the count of nine. Obviously to count up to a binary value of nine, the counter must have at least four flip-flops within its chain to represent each decimal digit. It is called a BCD counter because its ten state sequence is that of a BCD code and does not have a regular pattern, unlike a straight binary counter.

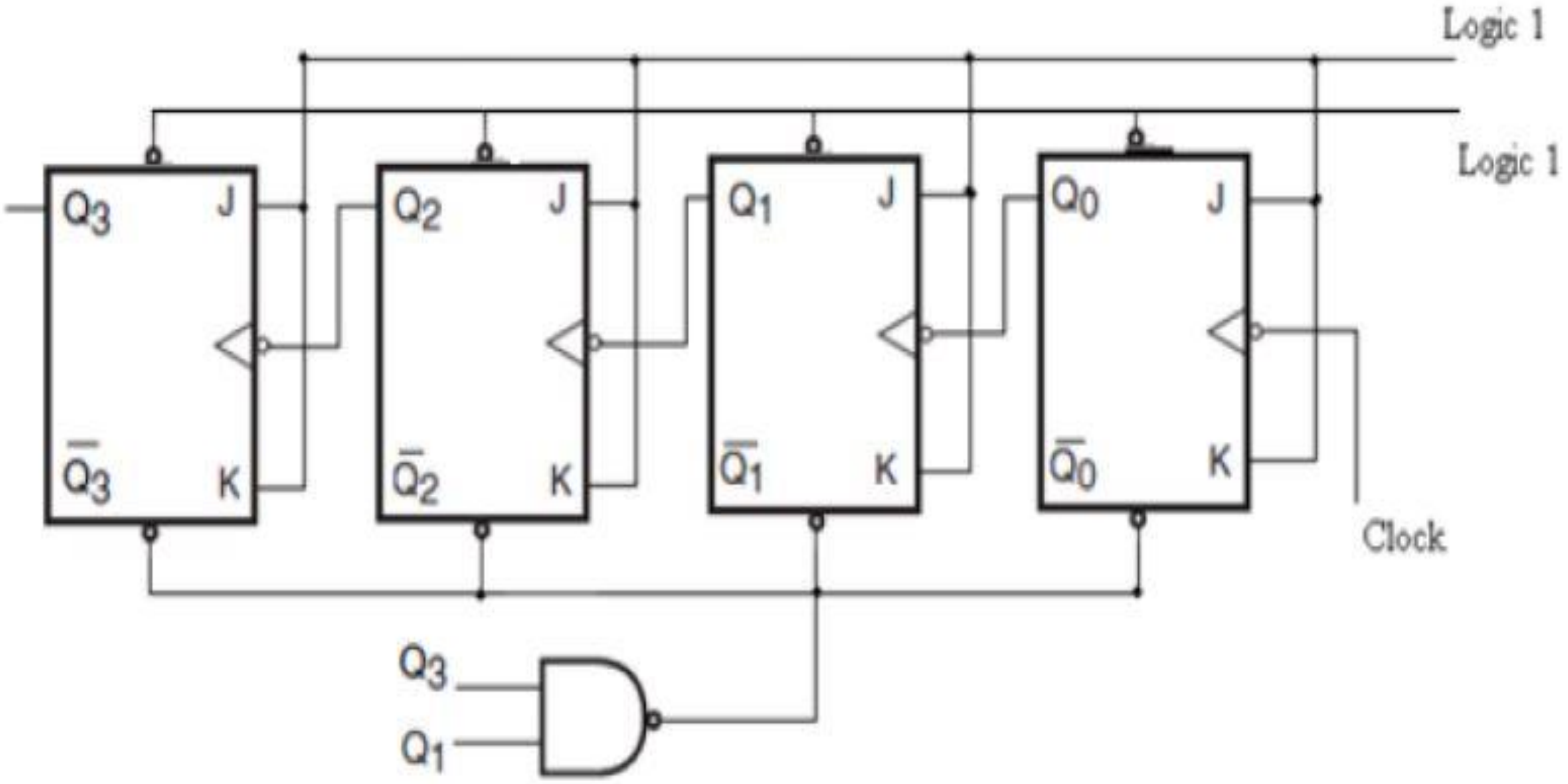
BCD Ripple Counter(State Diagram)

A decade counter has four flip-flops and 16 potential states, of which only 10 are used and if we connected a series of counters together we could count to 100 or 1,000 or to whatever final count number we choose.

BCD counters follow a sequence of ten states and count using BCD numbers from 0000 to 1001 and then returns to 0000 and repeats. 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 giving a MOD-10 count.



BCD Ripple Counter(Block Diagram)



Synchronous Counters

Synchronous counters are distinguished from ripple counters in that clock pulses are applied to the CP inputs of all flip-flops. The common pulse triggers all the 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 or not is determined from the values of the J and K inputs at the time of the pulse. If J = K = 0, the flip-flop remains unchanged. If J = K = 1, the flip-flop complements.

Different synchronous counter:

Binary counter

Binary up/down counter

BCD counter

Binary Counter

The design of synchronous binary counters is so simple that there is no need to go through a rigorous sequential-logic design process. In a synchronous binary counter, the flip-flop in the lowest-order position is complemented with every pulse. This means that it’s J and K inputs must be maintained at logic-1. A flip-flop in any other position is complemented with a pulse provided all the bits in the lower-order positions are equal to 1, because the lower-order bits (when all 1's) will change to 0's on the next count pulse.

Synchronous binary counters have a regular pattern and can easily be constructed with complementing flip-flops and gates. The regular pattern can be clearly seen from the 4-bit counter depicted in Fig:

The CP terminals of all flip-flops are connected to a common clock-pulse source. The first stage A1 has its J and K equal to 1 if the counter is enabled. The other J and K inputs are equal to 1 if all previous low-order bits are equal to 1 and the count is enabled. The chain of AND gates generates the required logic for the J and K inputs in each stage. The counter can be extended to any number of stages, with each stage having an additional flip-flop and an AND gate that gives an output of 1 if all previous flip-flop outputs are 1's.

Binary Counter

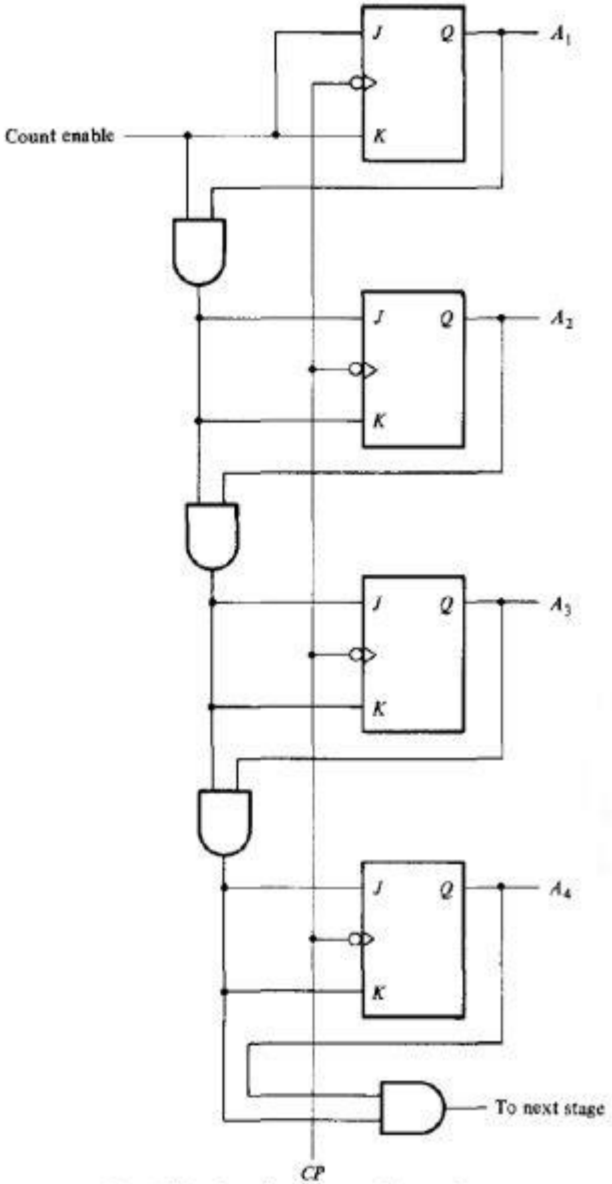


Fig: 4-bit Synchronous Binary Counter

Exercise

Q. Design 4-bit synchronous up counter.

Q. Design 4-bit synchronous down counter.

Q. Design 3-bit synchronous up/down counter.

Q. Design 4-bit synchronous up counter using RS flip flop.

Q. Design 3-bit ring counter using JK flip flop.

Q. Design mod 7 synchronous counter using JK flip flop.

Q. Design mod 10 synchronous counter using T flip flop

Timing Sequences

The sequences of operations in a digital system are specified by a control unit. The control unit that supervises the operations in a digital system would normally consist of timing signals that determine the time sequence in which the operations are executed. The timing sequences in the control unit can be easily generated by means of counters or shift registers.

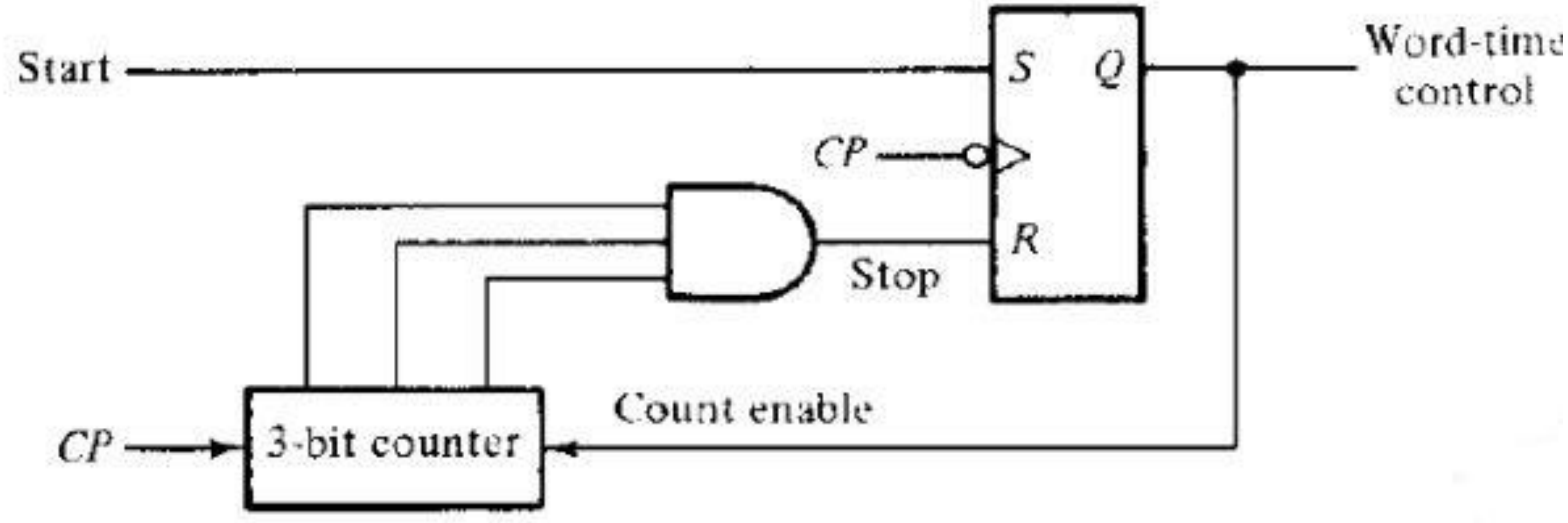
Word-Time Generation

The control unit in a serial computer must generate a word-time signal that stays on for a number of pulses equal to the number of bits in the shift registers. The word-time signal can be generated by means of a counter that counts the required number of pulses.

Example:

Word-Time Generation

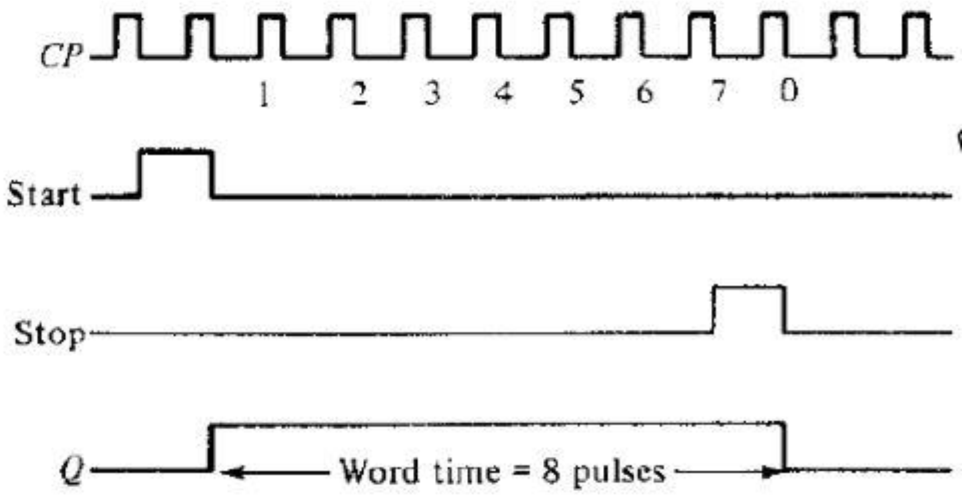
Assume that the word-time signal to be generated must stay on for a period of eight clock pulses. Fig. shows a counter circuit that accomplishes this task. Initially, the 3-bit counter is cleared to 0. A start signal will set flip-flop Q. The output of this flipflop supplies the word-time control and also enables the counter. After the count of eight pulses, the flip-flop is reset and Q goes to 0.



(a) Circuit diagram

Word-Time Generation

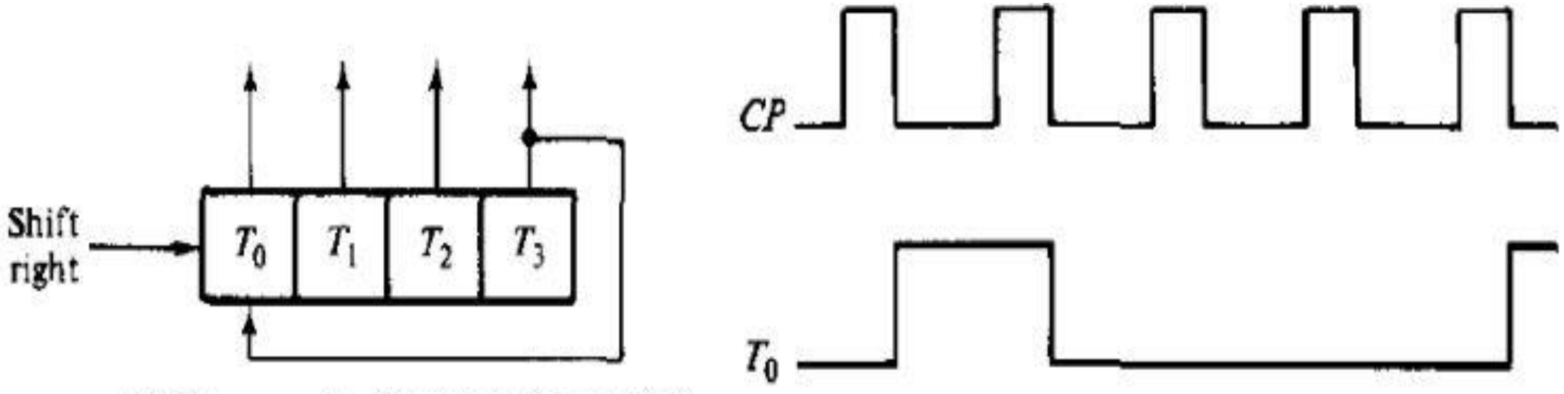
The timing diagram demonstrates the operation of the circuit. The start signal is synchronized with the clock and stays on for one clock-pulse period. After Q is set to 1, the counter starts counting the clock pulses. When the counter reaches the count of 7 (binary 111), it sends a stop signal to the reset input of the flip-flop. The stop signal becomes a 1after the negative-edge transition of pulse 7. The next clock pulse switches the counter to the 000 state and also clears Q. Now the counter is disabled and the word-time signal stays at 0.s



(b) Timing diagram

Timing Signals

The control unit in a digital system that operates in the parallel mode must generate timing signals that stay on for only one clock pulse period. Timing signals that control the sequence of operations in a digital system can be generated with a shift register or a counter with a decoder. A ring counter is a circular shift register with only one flip-flop being set at any particular time; all others are cleared. The single bit is shifted from one flip-flop to the other to produce the sequence of timing signals.



(a) Ringcounter (initial value = 1000)

Figure (a) shows a 4-bit shift register connected as a ring counter. The initial value of the register is 1000, which produces the variable T0. The single bit is shifted right with every clock pulse and circulates back from T3 to T0. Each flip-flop is in the 1 state once every four clock pulses and produces one of the four timing signals shown in Fig.Each output becomes a 1 after the negative-edge transition of a clock pulse and remains 1 during the next clock pulse.

Johnson Counter

The Johnson counter is similar to the Ring counter. The only difference between the Johnson counter and the ring counter is that the outcome of the last flip flop is passed to the first flip flop as an input. But in Johnson counter, the inverted outcome Q' of the last flip flop is passed as an input. The remaining work of the Johnson counter is the same as a ring counter. The Johnson counter is also referred to as the Creeping counter.

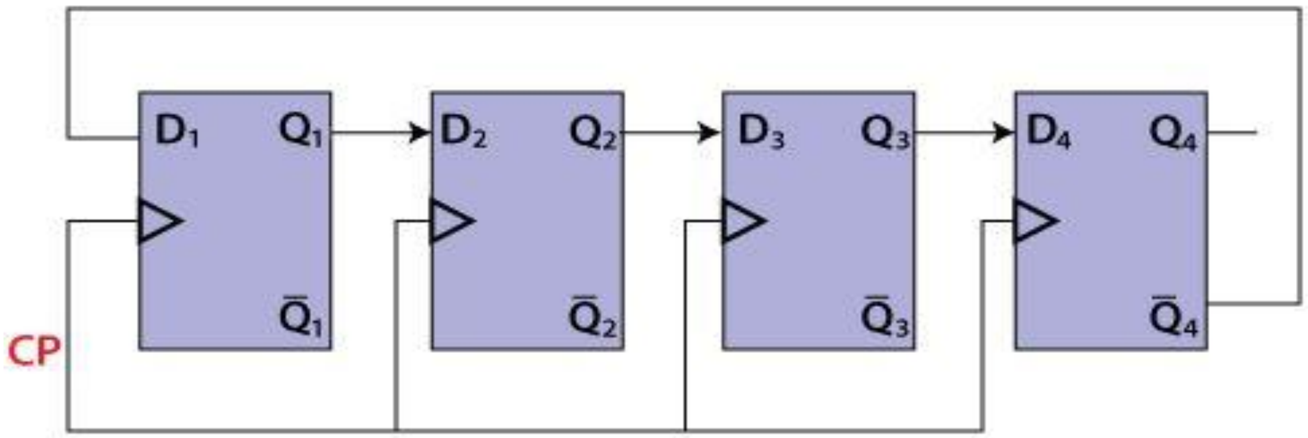
No. of states in Johnson counter = No. of flip-flop used

Number of used states=2n

Number of unused states=2n - 2*n

Below is the diagram of the 4-bit Johnson counter. Like Ring counter, four D flip flops are used in the 4-bit Johnson counter, and the same clock pulse is passed to all the input of the flip flops.

Johnson Counter



CP	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	1	1	1

Memory unit (Random Access Memory-RAM)

A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of the device. Memory cells can be accessed for information transfer to or from any desired random location and hence the name random access memory, abbreviated RAM.

A memory unit stores binary information in groups of bits called words. A word in memory is an entity of bits that move in and out of storage as a unit. A memory word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters, or any other binary-coded information

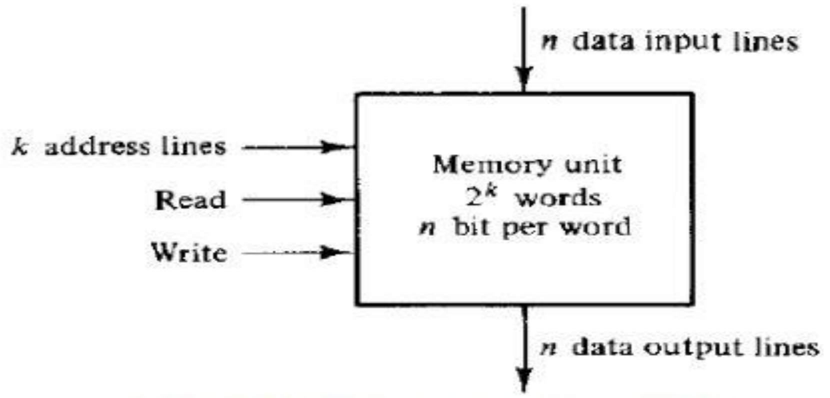


Fig: Block Diagram of a memory unit

Memory unit (Random Access Memory-RAM)

The communication between a memory and its environment is achieved through

n data input lines : provide information to be stored in memory

n data output lines: supply the information coming out of memory.

k address lines: specify particular word chosen among the many available.

two control inputs: specify the direction of transfer desired

Each word in memory is assigned an identification number, called an address, starting from 0 and continuing with 1, 2, 3, up to 2k- 1, where k is the number of address lines.

Computer memories may range from 1024 words, requiring an address of 10 bits, to 2^{32} words, requiring 32 address bits.

Conventions for Memory storage:

K (kilo) = 2^{10}

M (mega) = 2^{20}

G (giga) = 2^{30}

Thus, 64K = 2^{16}, 2M = 2^{21}, and 4G = 2^{32}

Example

Memory unit with a capacity of 1K words of 16 bits each. Since 1K = 1024 = 2^{10} and 16 bits constitute two bytes, we can say that the memory can accommodate 2048 = 2K bytes.

Each word contains 16 bits, which can be divided into two bytes. The words are recognized by their decimal address from 0 to 1023. The equivalent binary address consists of 10 bits. The first address is specified with ten 0's, and the last address is specified with ten 1's. A word in memory is elected by its binary address. When a word is read or written, the memory operates on all 16 bits as a single unit.

Memory address register (MAR): It is CPU register which contains the address of the memory words.
If memory has k address lines, then MAR is of k bits.

Memory Buffer Register (MBR): It contains the word-data pointed by the MAR.

Memory address		
Binary	decimal	Memory content
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Fig: Possible content of 1024 x 16 memory

Write and Read Operations

The two operations that a random-access memory can perform are the write and read operations. The write signal specifies a transfer-in operation and the read signal specifies a transfer-out operation. On accepting one of these control signals, the internal circuits inside the memory provide the desired

function.

Write Operation: transferring a new word to be stored into memory

1. Transfer the binary address of the desired word to the address lines.

2. Transfer the data bits that must be stored in memory to the data input lines.

3. Activate the write input.

Read Operation: transferring a stored word out of memory

1. Transfer the binary address of the desired word to the address lines.

2. Activate the read input.

Commercial memory components available in IC chips sometimes provide the two control inputs for reading and writing in a somewhat different configuration. The memory operations that result from these control inputs are specified in Table below.