

MODULE-5

Registers and Counters

Books Referred

- **Text Book:**

- Donald P Leach, Albert Paul Malvino & Goutam Saha: Digital Principles and Applications, 7th Edition, Tata McGraw Hill, 2015

- **Reference Books:**

- Anil K. Maini: Digital Electronics Principles, Devices and Applications, 2007, John Wiley & Sons, Ltd.
- Thomas L. Floyd: Digital Fundamentals, 9th Edition., Pearson International Edition.
- M. Morris Mano, Michael D. Ciletti : Digital Design With an Introduction to the Verilog HDL, 5th Edition, Pearson Education, Inc

Objective

• Register

- Understand serial in-serial out shift registers and be familiar with the basic features of the 74LS91 register
- Understand serial in-parallel out shift registers and be familiar with the basic features of the 74164 register
- Understand parallel in-serial out shift registers and be familiar with the basic features of the 74166 register
- Understand parallel in-parallel out shift registers and be familiar with the basic features of the 74174 and 7495A registers
- Understand working of Universal shift register with the basic features of the 74194 register.
- State various uses of shift registers

• Counter

- Describe the basic construction and operation of an asynchronous counter
- Determine the logic circuit needed to decode a given state from the output of a given counter
- Describe the synchronous counter and its advantages
- See how the modulus of a counter can be reduced by skipping one or more of its natural counts

REGISTERS

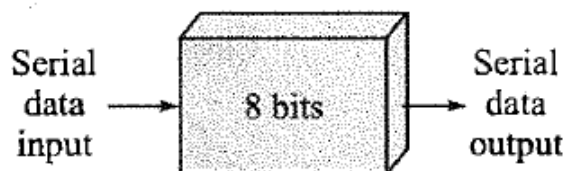
Introduction

- A register is simply a group of flip-flops that can be used to store a binary number. There must be one flip-flop for each bit in the binary number.
- A register is a digital circuit with two basic functions: data storage and data movement.
 - E.g. a register used to store an 8-bit binary number must have eight flip-flops.
- Naturally the flip-flops must be connected such that the binary number can be entered (shifted) into the register and possibly shifted out.
 - A group of flip-flops connected to provide either or both of these functions is called a **shift register**.
- Applications:
 - A data register is often used to momentarily store binary information e.g. RAM.
 - A register used in a microprocessor chip. E.g. Processor Register
 - Shift Register is a sequence generator and sequence detector and also as parallel to serial converters offers very distinct advantages.

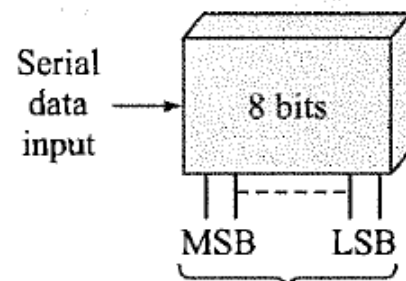
TYPES OF REGISTERS-1

- The bits in a binary number can be moved from one place to another in either of two ways.
- The first method involves shifting the data 1 bit at a time in a serial fashion, beginning with either the most significant bit (MSB) or the least significant bit (LSB). This technique is referred to as *serial shifting*.
- The second method involves shifting all the data bits simultaneously and is referred to as *parallel shifting*.
- There are two ways to shift data into a register (serial or parallel) and similarly two ways to shift the data out of the register.
- This leads to the construction of four basic register types:
 - Serial-in-serial out (SISO) 54/74LS91, 8 bits
 - Serial in-parallel out (SIPO) 54/74164, 8 bits
 - Parallel in-serial out (PISO) 54/74165, 8 bits
 - Parallel in-parallel out (PIPO) 54/74198, 8 bits

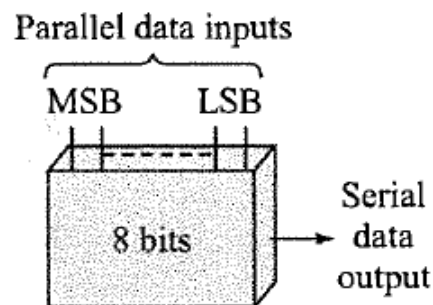
TYPES OF REGISTERS-2



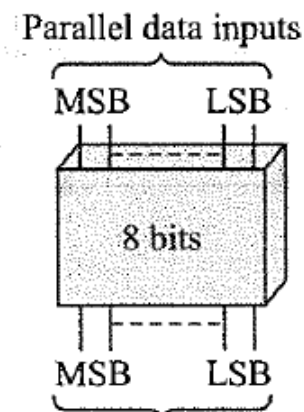
(a) Serial in-serial out



Parallel data outputs
(b) Serial in-parallel out



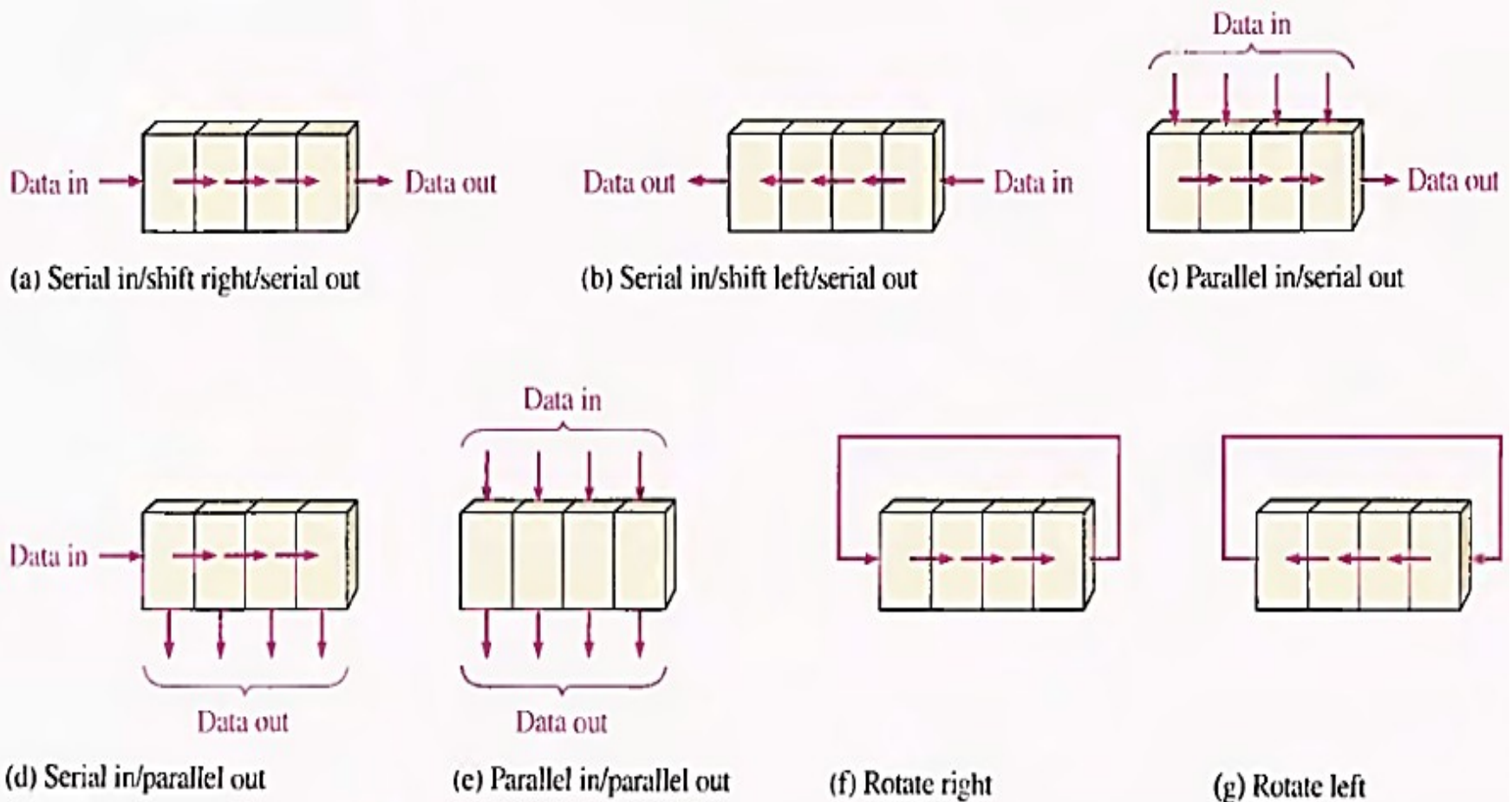
(c) Parallel in-serial out



(d) Serial in-parallel out

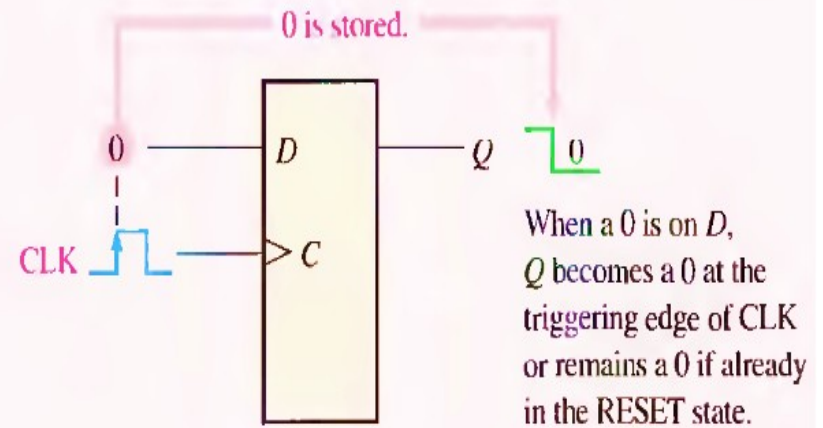
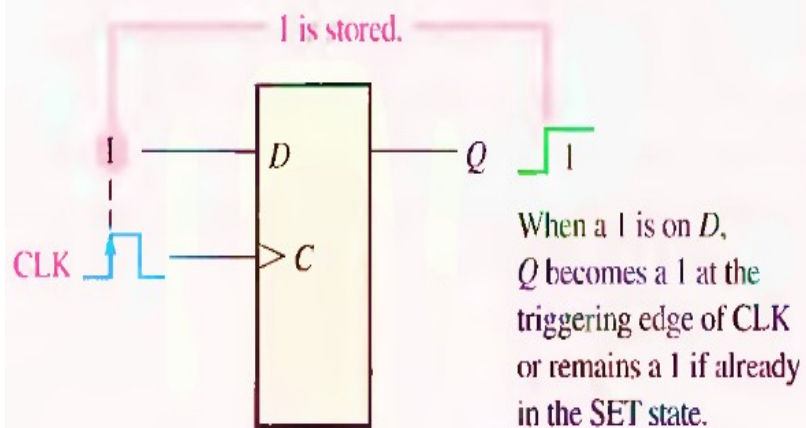
Shift register types

TYPES OF REGISTERS-3



Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

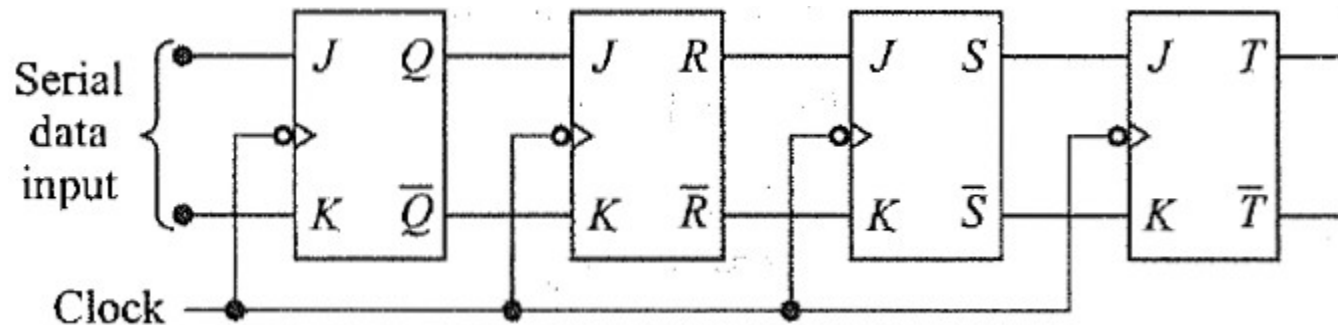
TYPES OF REGISTERS-4



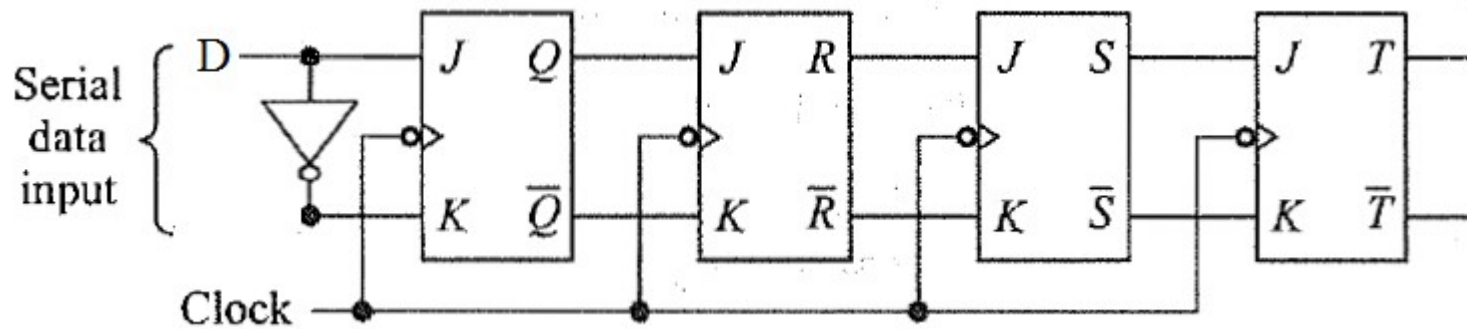
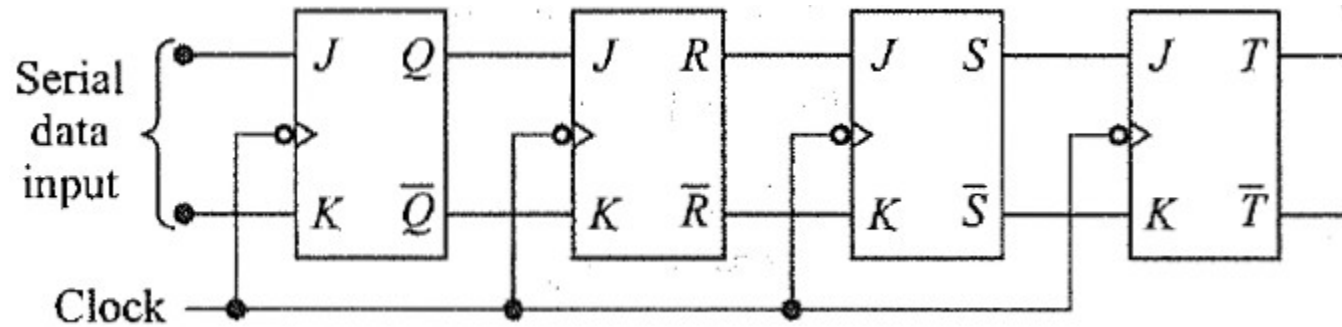
The flip-flop as a storage element.

SERIAL IN-SERIAL OUT-1

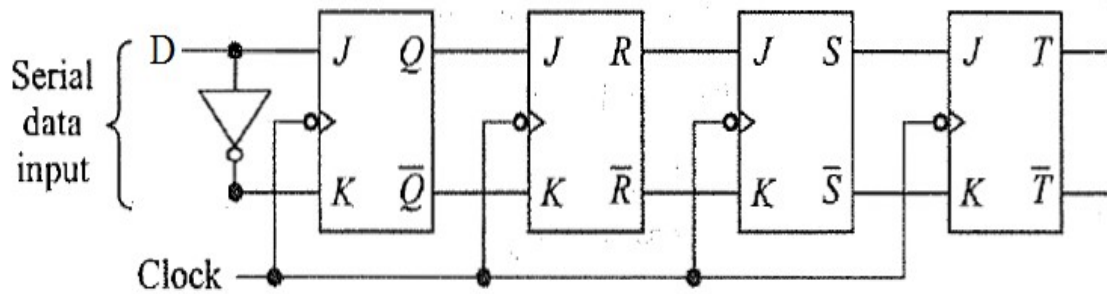
- The flip-flops used to construct registers are usually edge-triggered JK , SR or D types.
- D flip-flops connected as shown in figure forming 4-bit shift register. A common clock provides trigger at its negative edge to all the flip-flops.
- As output of one D flip-flop is connected to input of the next at every Clock trigger data stored in one flip-flop is transferred to the next.
- For this circuit transfer takes place like this $Q \rightarrow R$, $R \rightarrow S$, $S \rightarrow T$ and *serial data input* is transferred to Q .



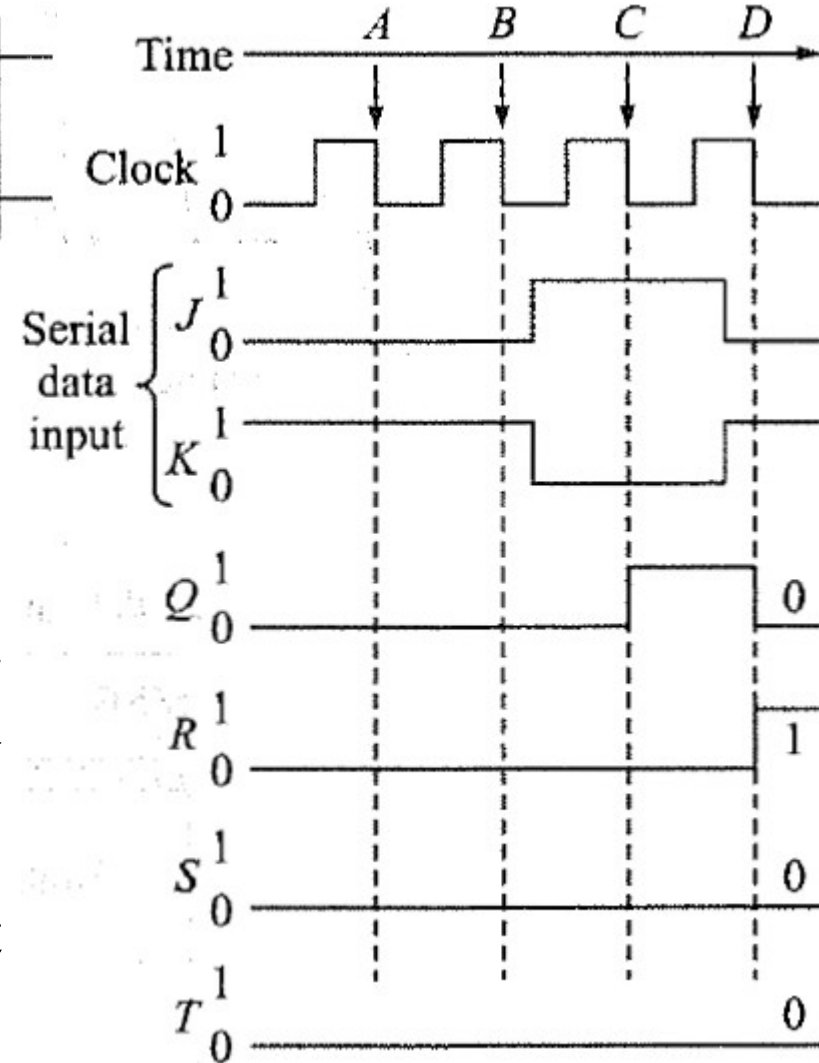
SERIAL IN-SERIAL OUT-1



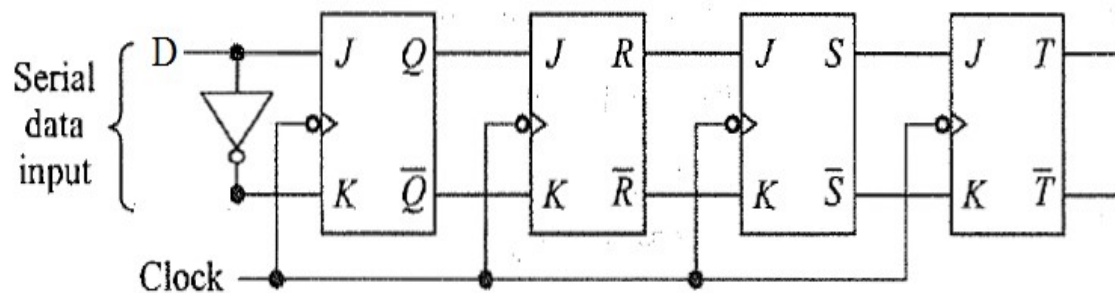
SERIAL IN-SERIAL OUT-2



- **At clock edge A ,**
- $D = 0 \rightarrow Q, Q \rightarrow R, R \rightarrow S, S \rightarrow T$
- When clock triggers, these inputs get transferred to corresponding flip-flop outputs simultaneously so that $QRST = 0000$.
- Thus at clock trigger, values at $DQRS$ is transferred to $QRST$.



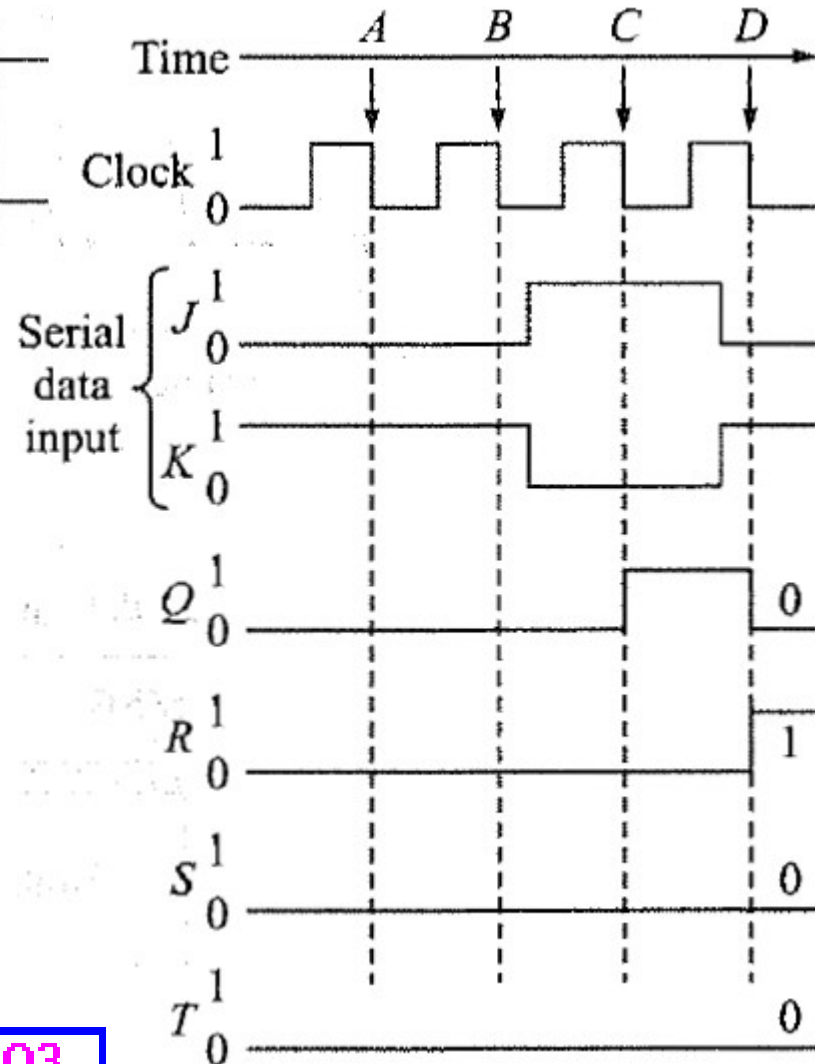
SERIAL IN-SERIAL OUT-3



- At clock edge **B**, serial data in $D=0$, i.e. $DQRS = 0000$. So after NT at **B**, $QRST = 0000$.
- Serial data becomes 1 in next clock cycle.
- At clock edge **C**, $DQRS = 1000$ and after NT $QRST = 1000$.
- Serial data goes to 0 in next clock cycle
- At clock edge **D**, $DQRS = 0100$ and after NT $QRST =$

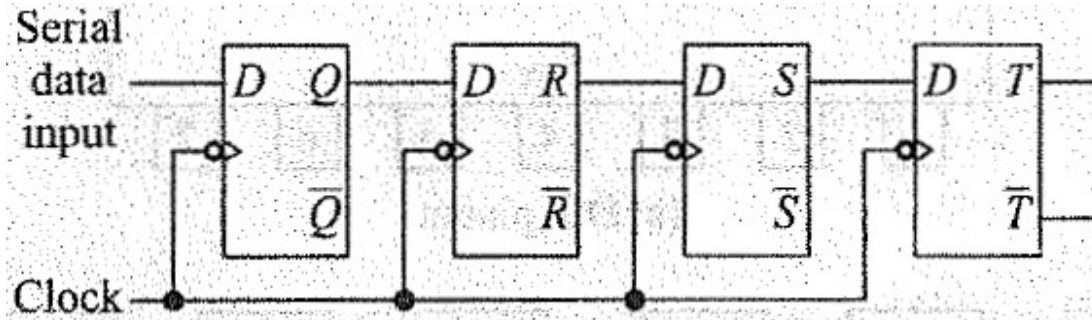
CLEAR
1001

Q0	Q1	Q2	Q3
0	0	0	0



SERIAL IN-SERIAL OUT-4

- Example: Show how a number 0100 is entered serially in a 4-bit shift register using D flip-flop. Also write state table.

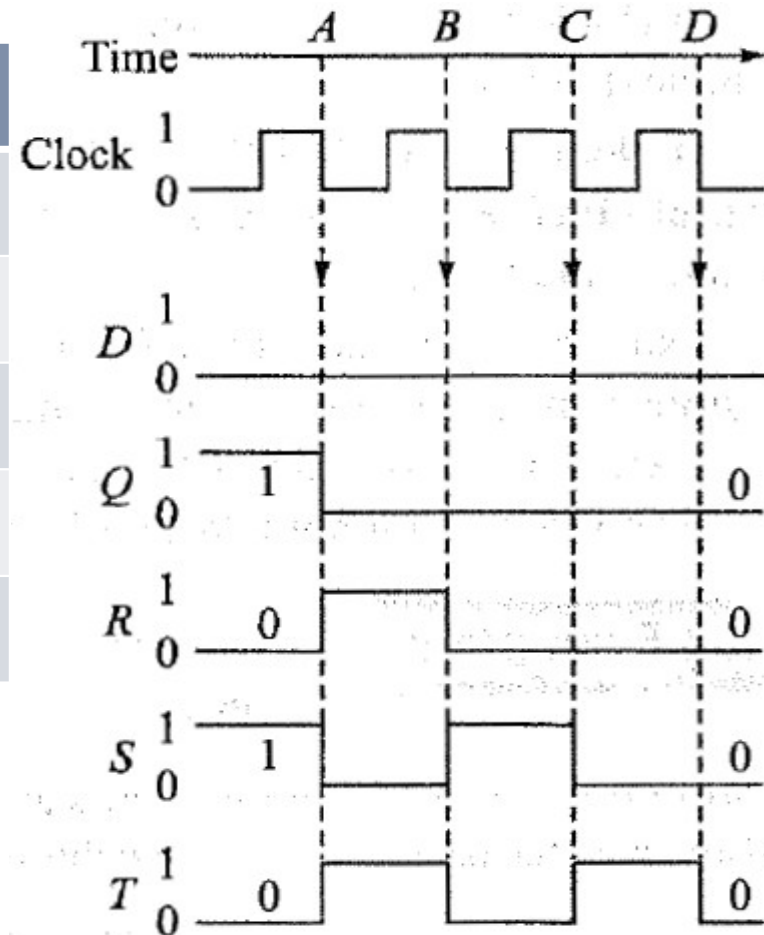


Clock	Serial input	Q	R	S	T
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4		0	0	1	0

SERIAL IN-SERIAL OUT-5

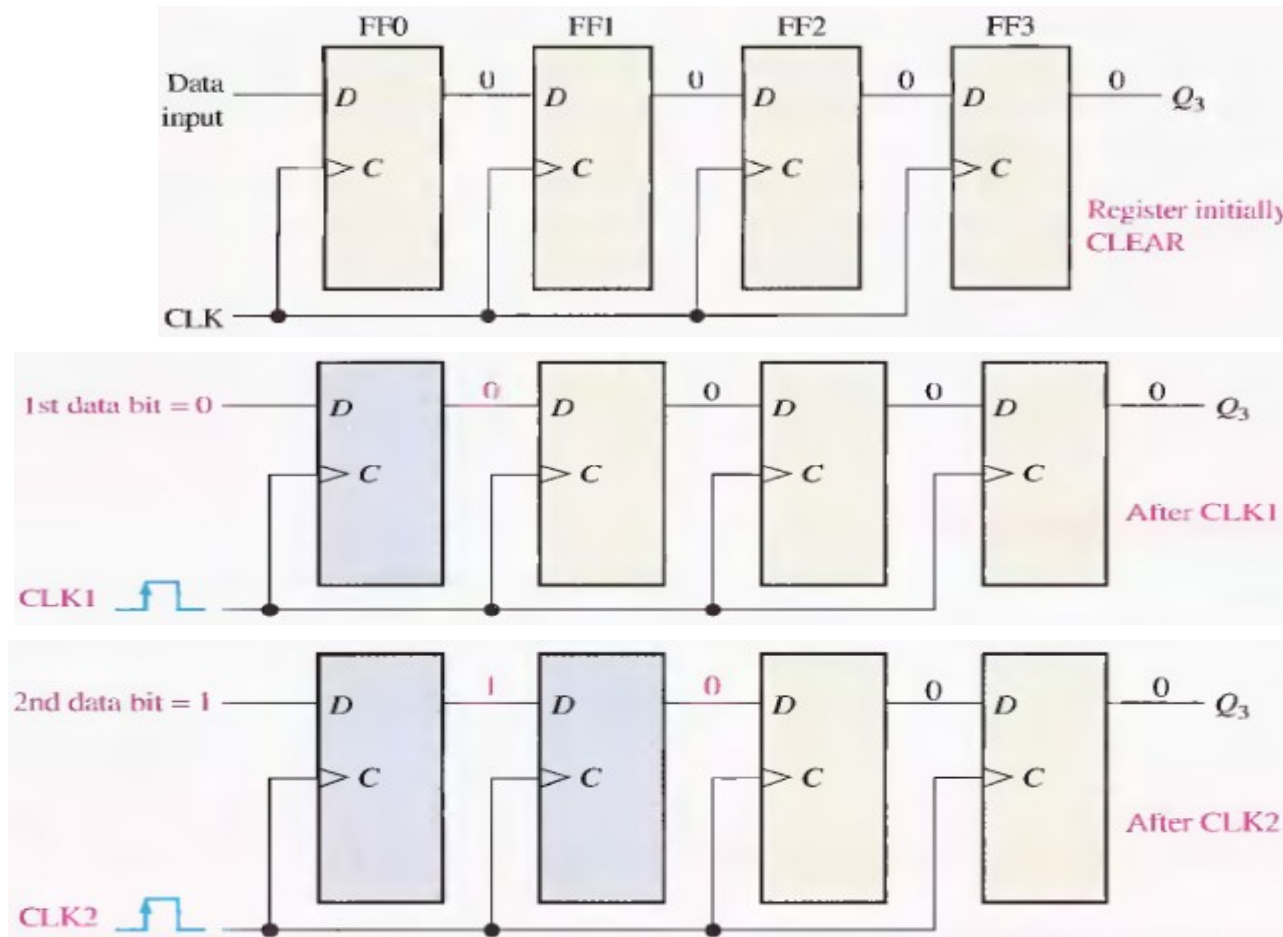
- Example: Suppose that it has the 4-bit number $QRST = 1010$ stored in it so draw the waveform

Clk	Serial In	Q	R	S	T
0	0	1	0	1	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0



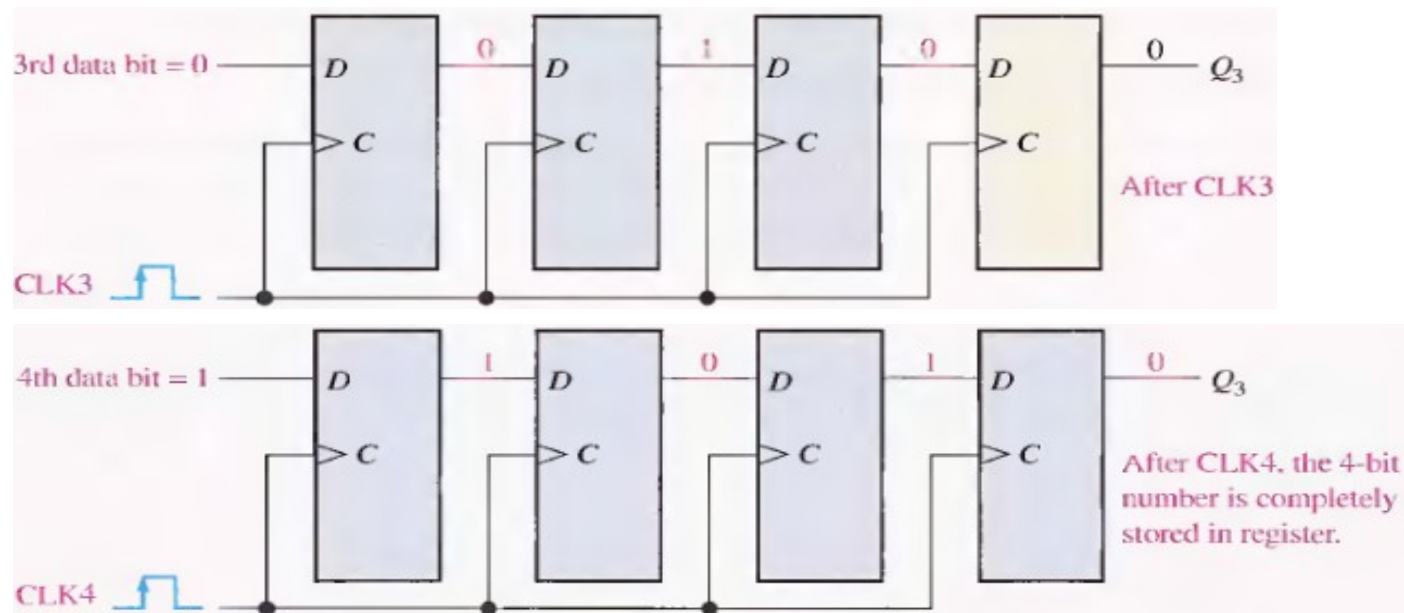
SERIAL IN-SERIAL OUT-6

- Example: 1010 Serial Input Entering serially into register



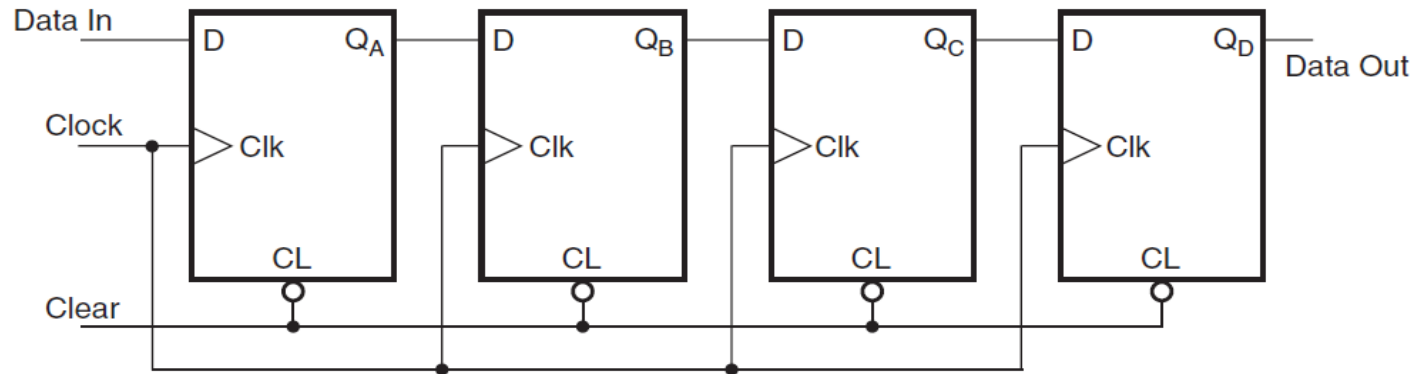
SERIAL IN-SERIAL OUT-7

- Example: 1010 Serial Input Entering serially into register



SERIAL IN-SERIAL OUT-8

- Example
- Data In 1001

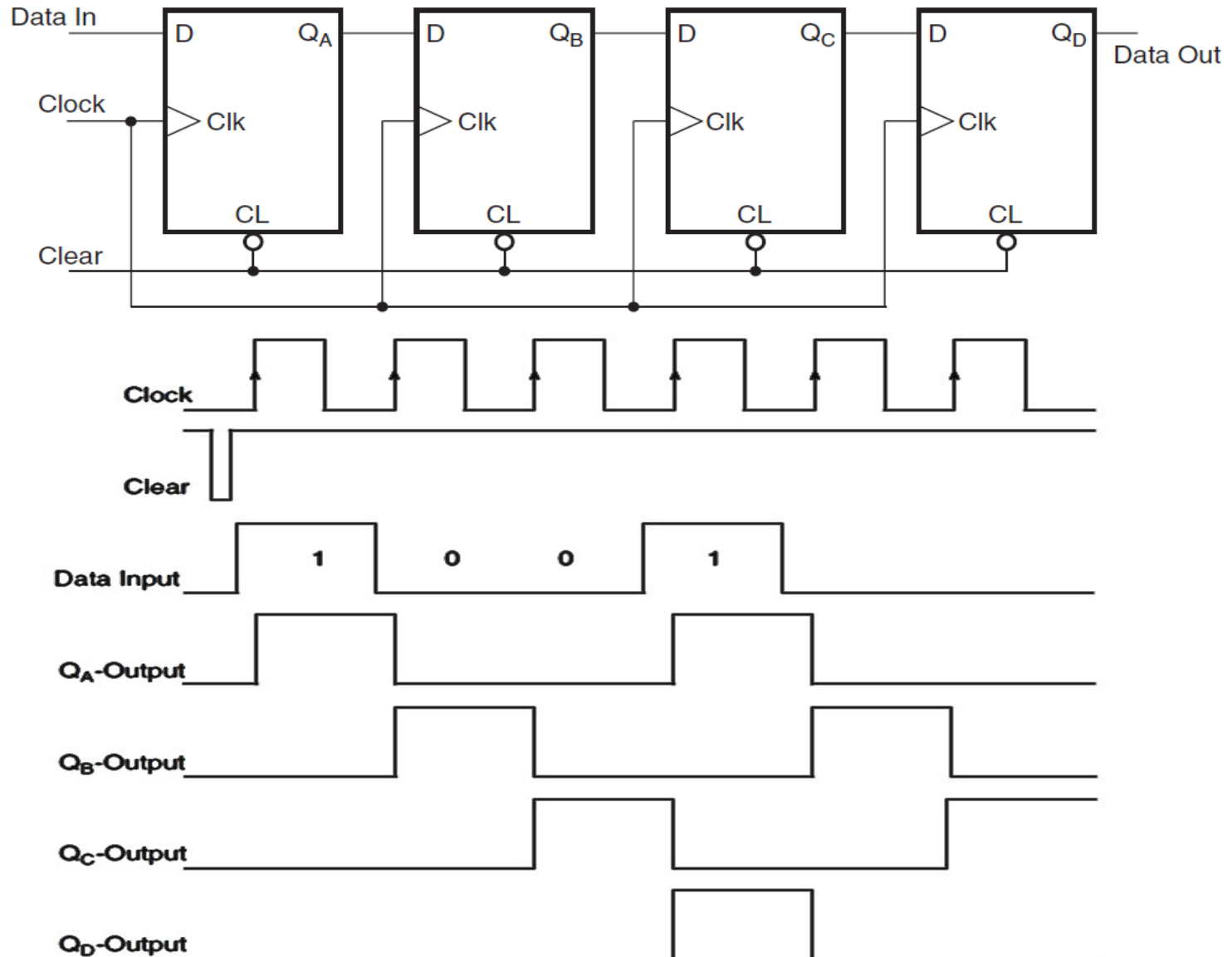


Contents of four-bit serial-in serial-out shift register for the first eight clock cycles.

Clock	Q_A	Q_B	Q_C	Q_D
Initial contents	0	0	0	0
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	0	0	1	0
After fourth clock transition	1	0	0	1
After fifth clock transition	0	1	0	0
After sixth clock transition	0	0	1	0
After seventh clock transition	0	0	0	1
After eighth clock transition	0	0	0	0

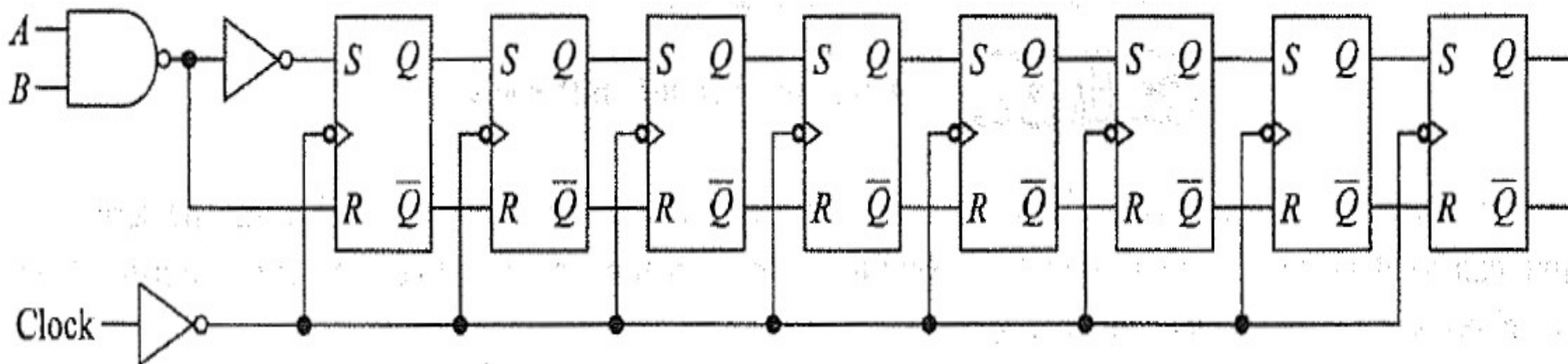
SERIAL IN-SERIAL OUT-9

- Example



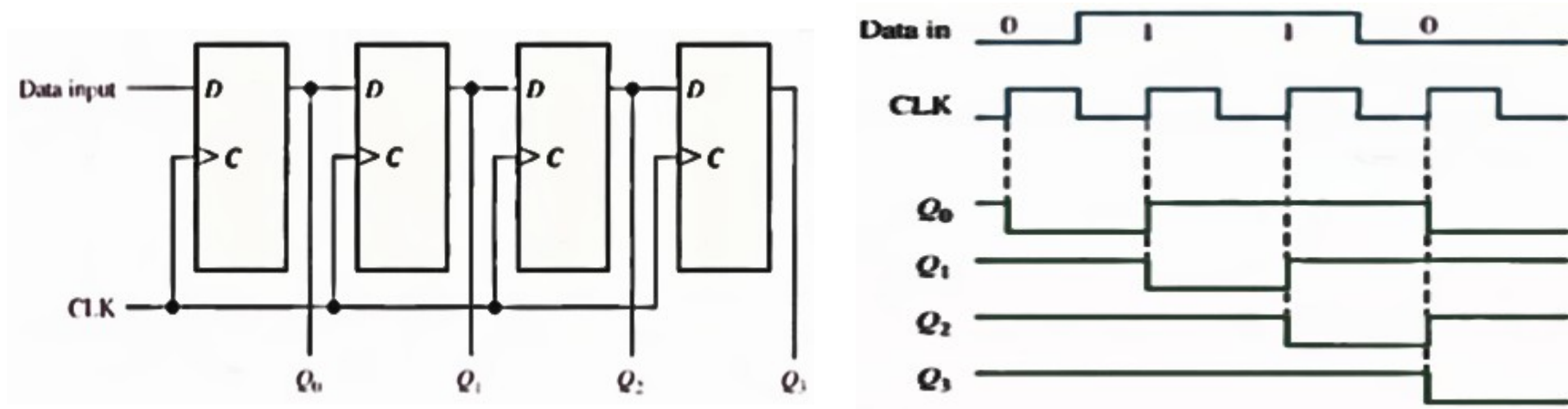
SERIAL IN-SERIAL OUT-10

- **74LS91 8-bit shift register**
- The data input is applied at either A (pin 10) or B (pin 12). Notice that a data level at A (or B) is complemented by the NAND gate and then applied to the R input of the first flip-flop.
- The same data level is complemented by the NAND gate and then complemented again by the inverter before it appears at the S input. So, a 1 at input A will set the first flip-flop (in other words, this 1 is shifted into the first flip-flop) on a positive clock transition.
- The NAND gate with inputs A and B simply provides a gating function for the input data stream if desired.
- If gating is not desired, simply connect pins 10 and 12 together and apply the input data stream to this connection.



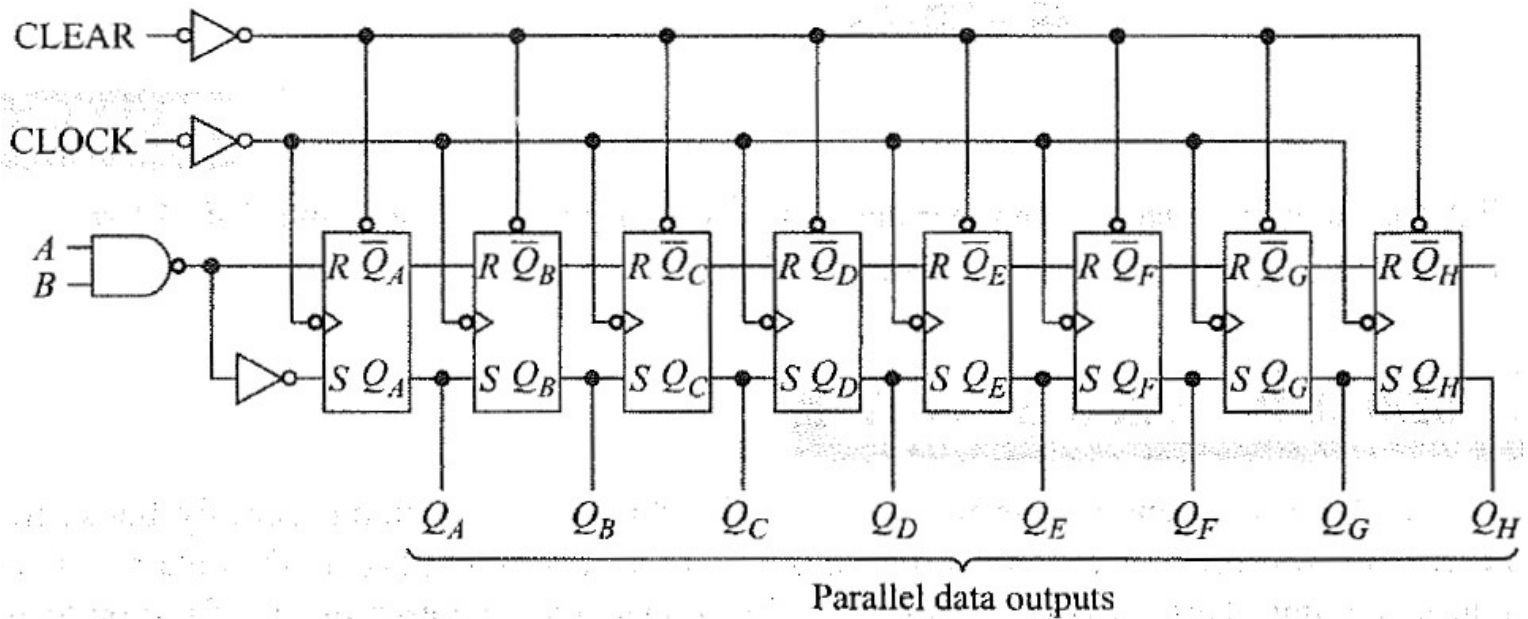
SERIAL IN-PARALLEL OUT-1

- Data is shifted in serially, but shifted out in parallel.
- In order to shift the data out in parallel, it is simply necessary to have all the data bits available as outputs at the same time.
- E.g. 4-bit shift register



SERIAL IN-PARALLEL OUT-1

- E.g. 8-bit shift register



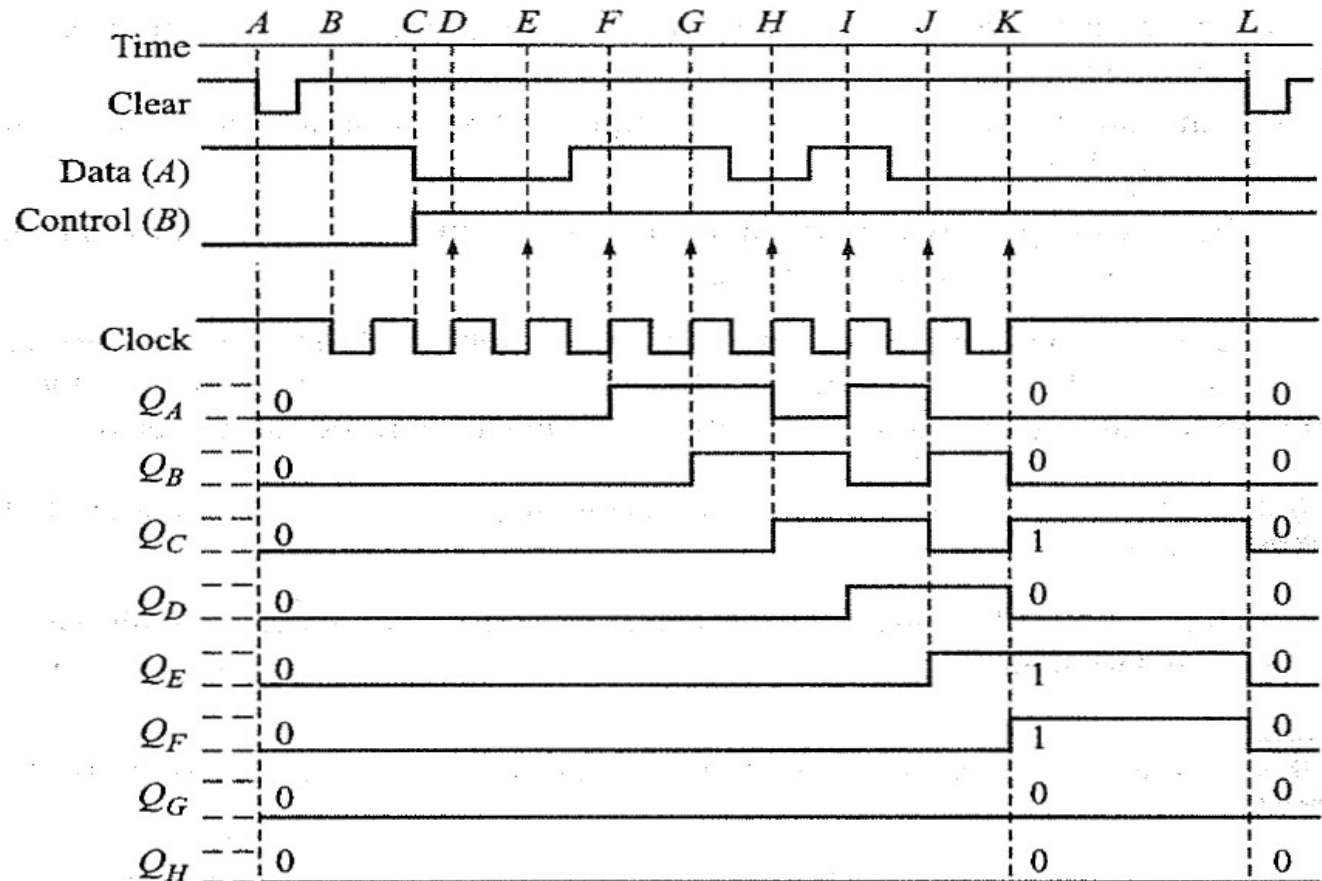
(b) Logic diagram

SERIAL IN-PARALLEL OUT-2

- How long will it take to shift an 8-bit number into a 8-bit shift register if the clock is set at 10 MHz?
- A minimum of eight clock periods will be required since the data is entered serially. One clock period is 100 ns, so it will require 800 ns minimum.

SERIAL IN-PARALLEL OUT-3

- The waveforms shown below, show the typical response of a 54/74164. The serial data is input at A (pin 1), while a gating control signal is applied at B (pin 2). The first clear pulse occurs at time A and simply resets all flip-flops to 0.

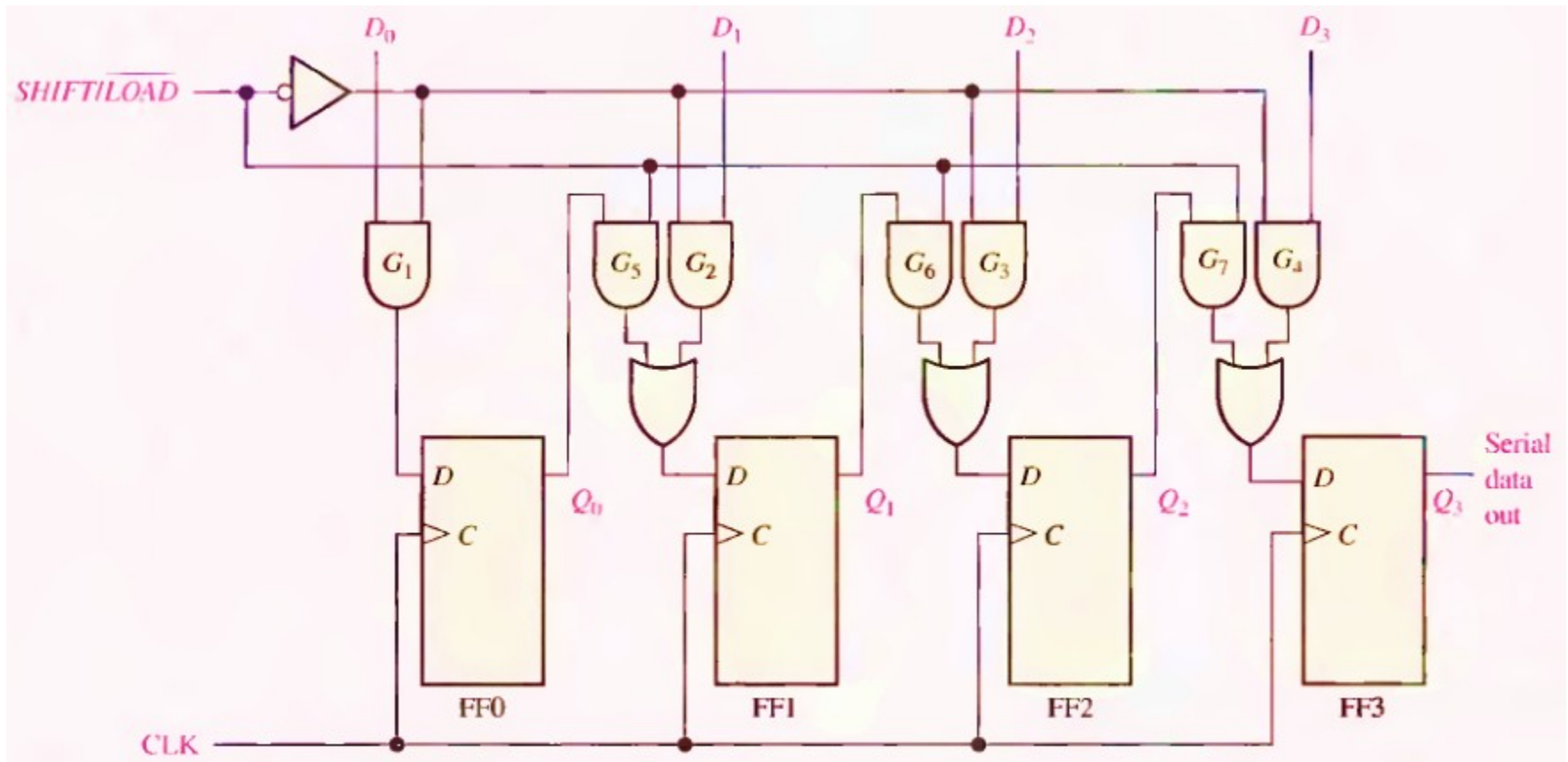


PARALLEL IN-SERIAL OUT-1

- For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs.
- Next slide illustrates a 4-bit parallel in/serial out shift register and a typical logic symbol.
- Notice that there are four data-input lines, D_0 , D_1 , D_2 and D_3 and a SHIFT/LOAD input, which allows four bits of data to load in parallel into the register.
- When SHIFT/LOAD is LOW, gates G1 through G4 are enabled, allowing each data bit to be applied to the D input of its respective flip-flop. When a clock pulse is applied, the flip-flops with $D = 1$ will set and those with $D = 0$ will reset. thereby storing all four bits simultaneously.
- When SHIFT/LOAD is HIGH, gates G1 through G4 are disabled and gates G5 through G 7 are enabled, allowing the data bits to shift right from one stage to the next.
- The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the SHIFT/LOAD input.

PARALLEL IN-SERIAL OUT-2

- 4-bit parallel in/serial out shift register.



PARALLEL IN-SERIAL

- The 54/74166 is an 8-bit shift register.

Inputs					Internal Levels		Outputs
Clear	Shift/load	Clock inhibit	Clock	Serial	Parallel $A \dots H$	Q_A and Q_B	Q_H
L	X	X	X	X	X	L L	L
H	X	L	L	X	X	Q_{AO} Q_{BO}	Q_{HO}
H	L	L	↑	X	$a \dots h$	a b	h
H	H	L	↑	H	X	H Q_{An}	Q_{Gn}
H	H	L	↑	L	X	L Q_{An}	Q_{Gn}
H	X	H	↑	X	X	Q_{AO} Q_{BO}	Q_{HO}

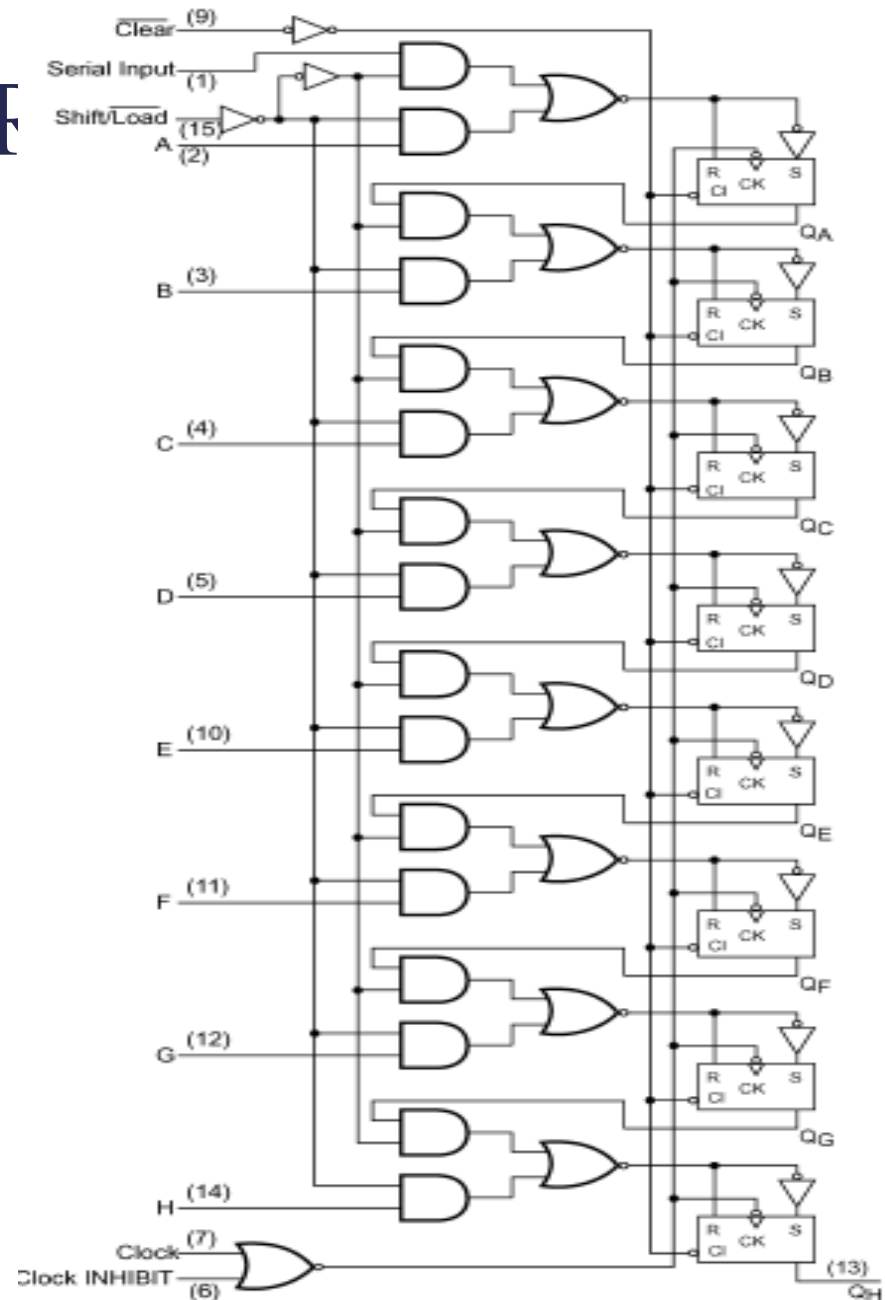
X = Irrelevant, H = High level, L = Low level

↑ = Positive transition

$a \dots h$ = Steady state input level at $A \dots H$ respectively

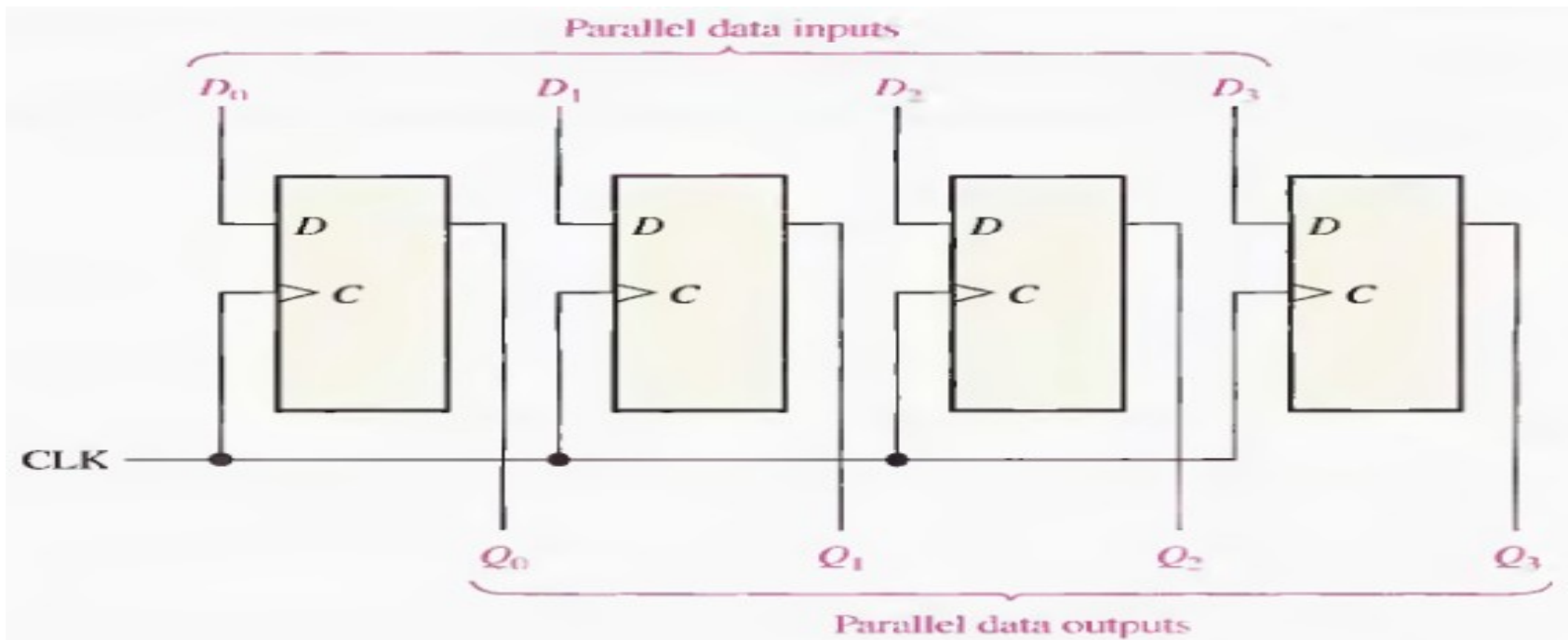
Q_{AO}, Q_{BO} = Level at $Q_A, Q_B \dots$ before steady state

Q_{An}, Q_{Gn} = Level of Q_A or Q_B before most recent transition () ↑



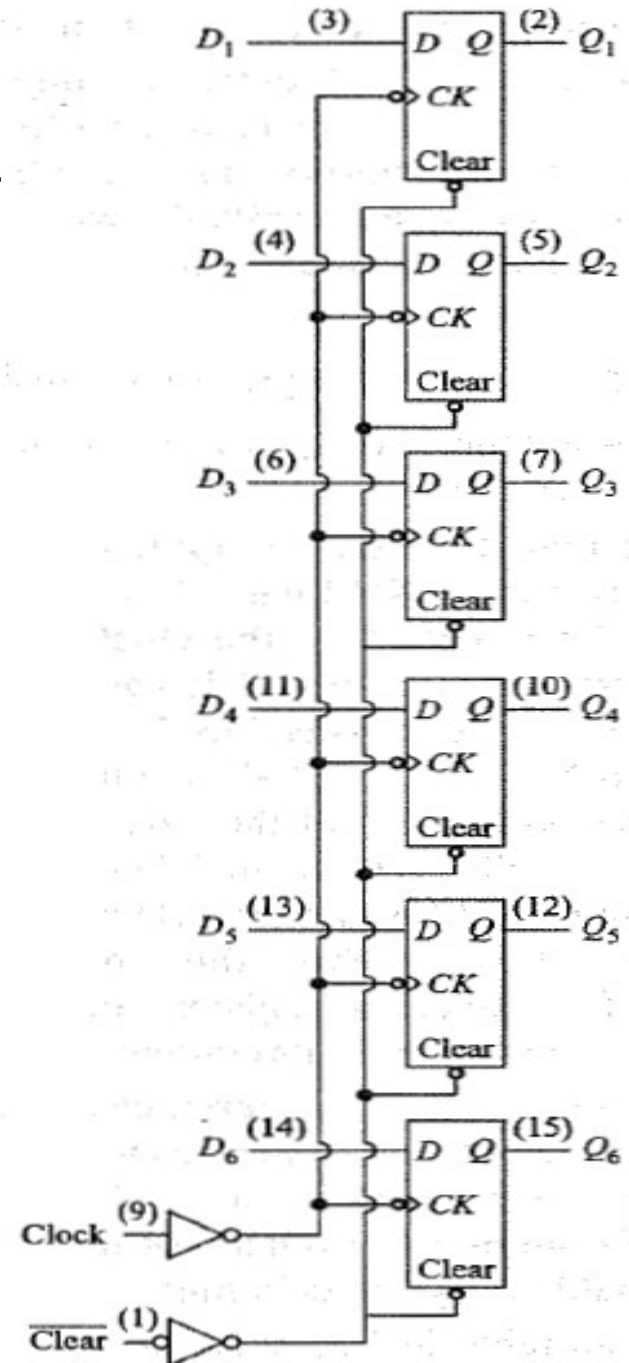
PARALLEL IN- PARALLEL OUT-1

- The parallel in/parallel out register employs both methods. Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs.



PARALLEL IN- PARALLEL

- The 74174 is an example of a 6- bit parallel in-parallel out register.

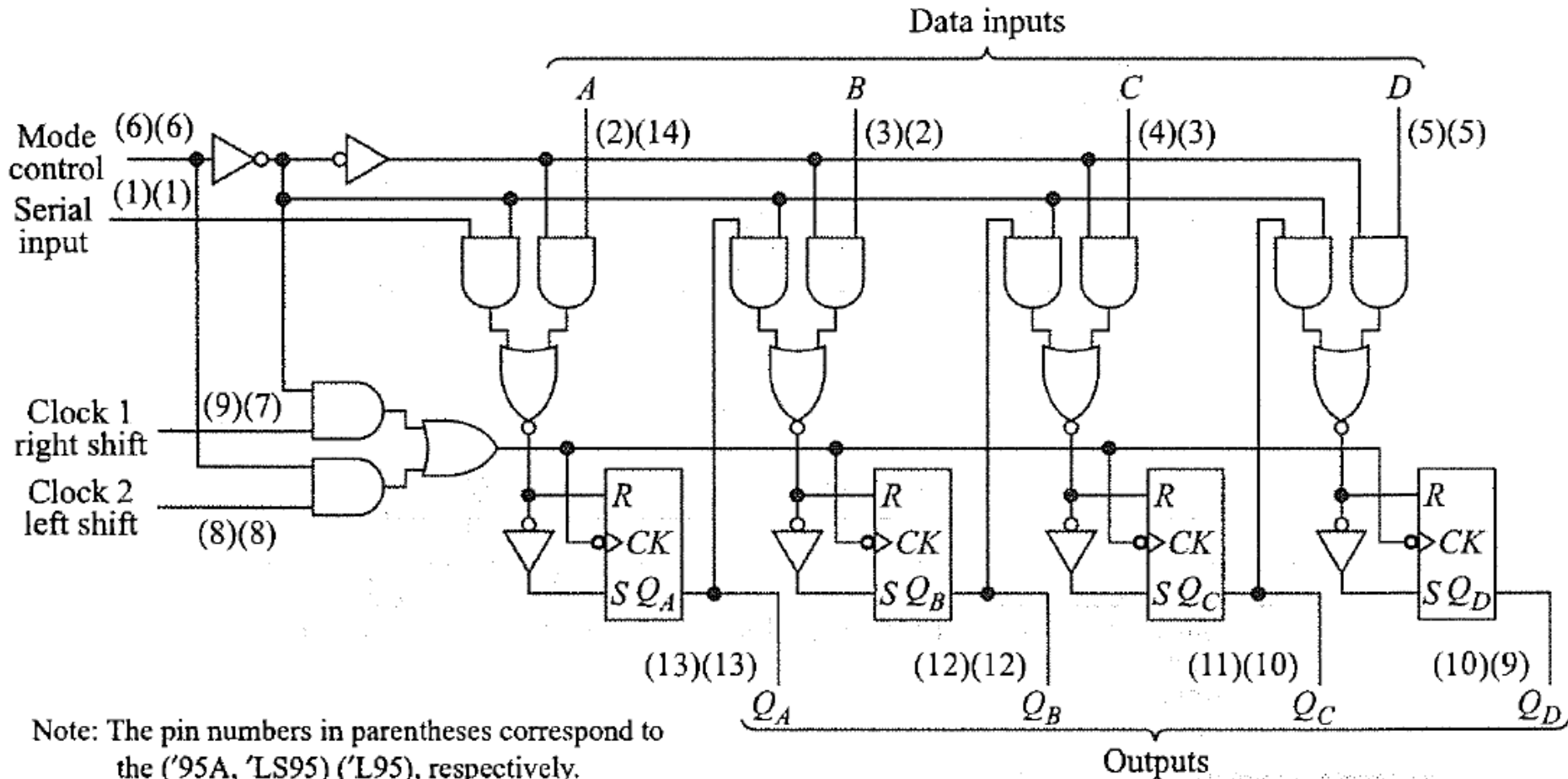


PARALLEL IN- PARALLEL OUT-3

- The 74LS174 data sheet gives a setup time of 20 ns and a hold time of 5 ns. What is the minimum required width of the data input levels (*D1* ... *D6*) for the 74LS174?

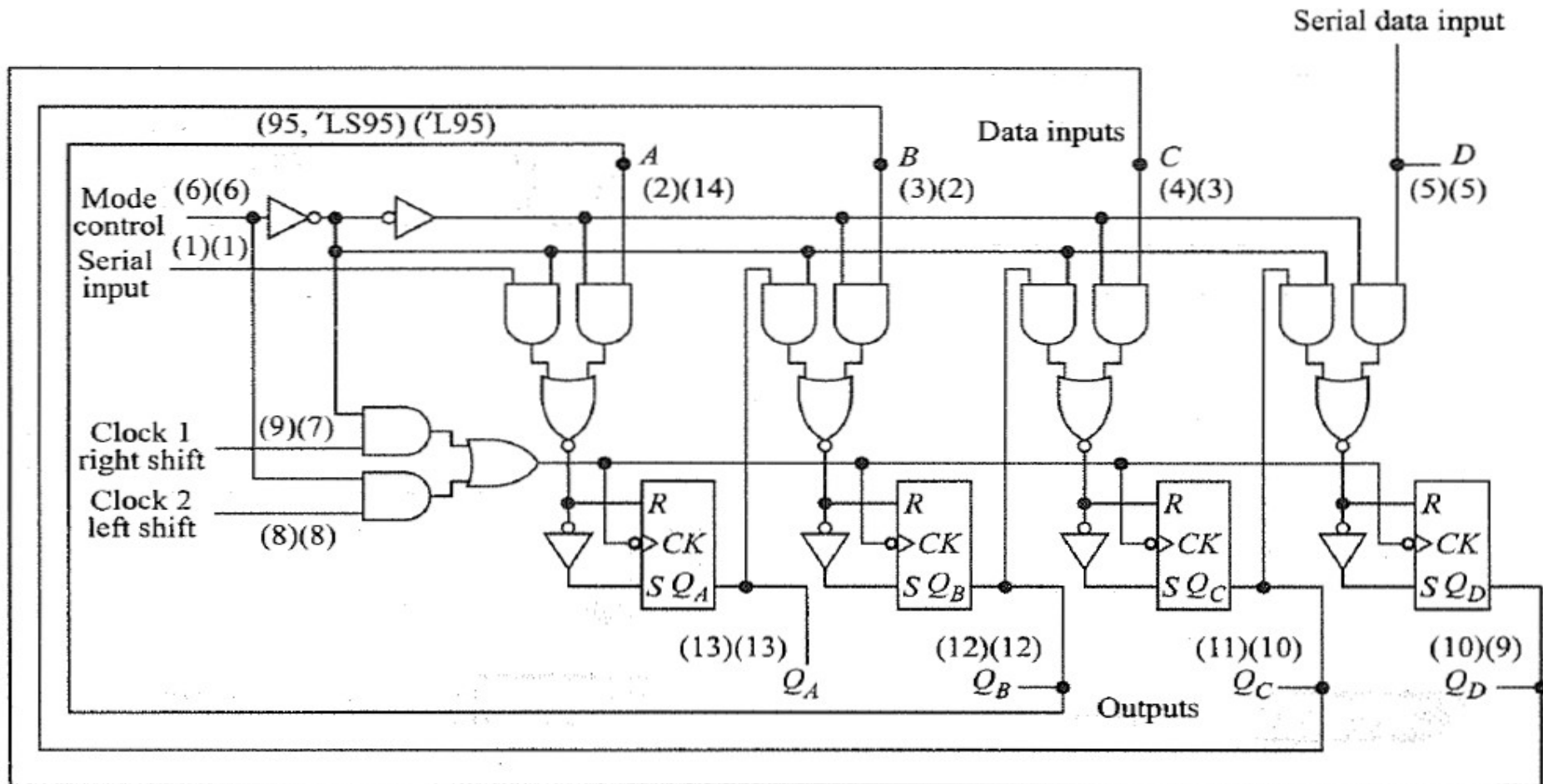
PARALLEL IN- PARALLEL OUT-4

- The 5417495A describes it as a 4-bit parallel-access shift register. It also has serial data input and can be used to shift data to the right (from Q_A toward Q_B) and in the opposite direction to the left.



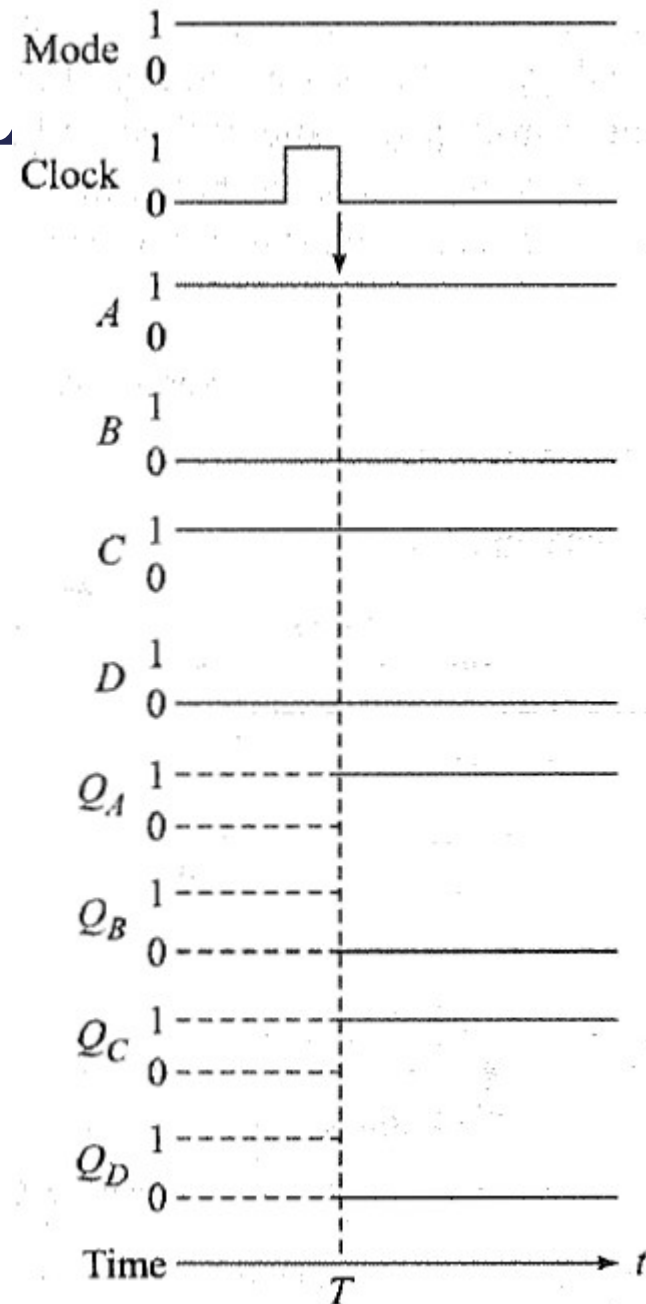
PARALLEL IN- PARALLEL OUT-4

- The 5417495A describes it as a 4-bit parallel-access shift register. It also has serial data input and can be used to shift data to the right (from Q_A toward Q_B) and in the opposite direction to the left.



PARALLEL IN- PARALL

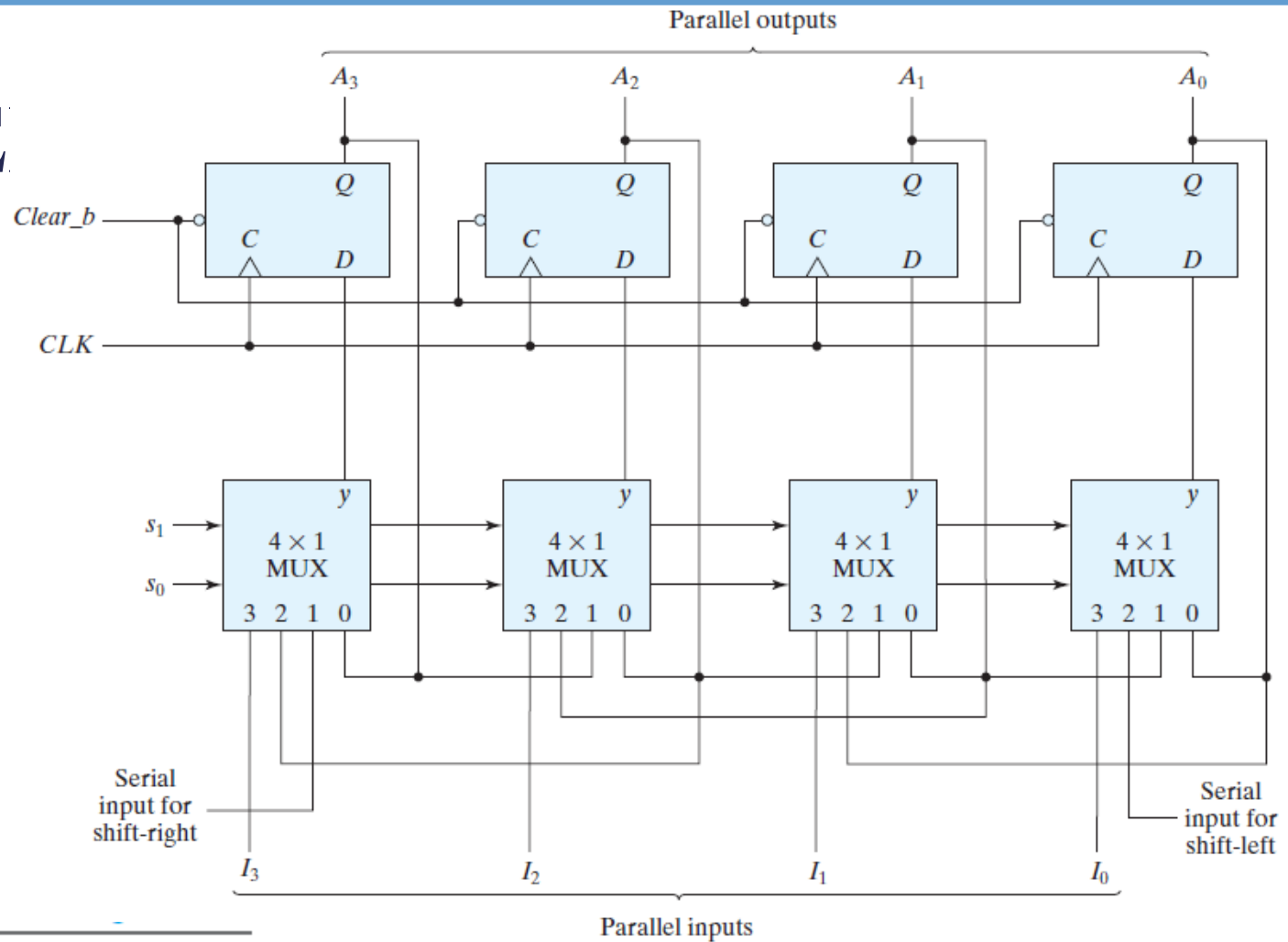
- Draw the waveforms you would expect if the 4-bit binary number 1010 were shifted into a 5417495A in parallel.
- The mode control line must be high, The data. input line must be stable for more than 10 ns prior to the clock NTs (setup time for the data sheet information). A single clock NT will enter the data. If the clock is stopped after the transition time T , the levels on the input data lines may be changed, However, if the clock is not stopped, the input data line levels must be maintained.



UNIVERSAL SHIFT REGISTER-1

- Basic types of shift register ,the following operations are possible-serial in-serial out, serial in-parallel out, parallel in-serial out, and parallel in-parallel out. Serial in or serial out again can be made possible by shifting data in any of the two directions, left shift ($QA \leftarrow QB \leftarrow QC \leftarrow QD \leftarrow \text{Data in}$) and right shift ($\text{Data in} \rightarrow QA \rightarrow QB \rightarrow QC \rightarrow QD$). A universal shift register can perform all the four operations and is also bidirectional in nature.

UNIVE



Mode Control

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

APPLICATIONS Of SHIFT REGISTERS-1

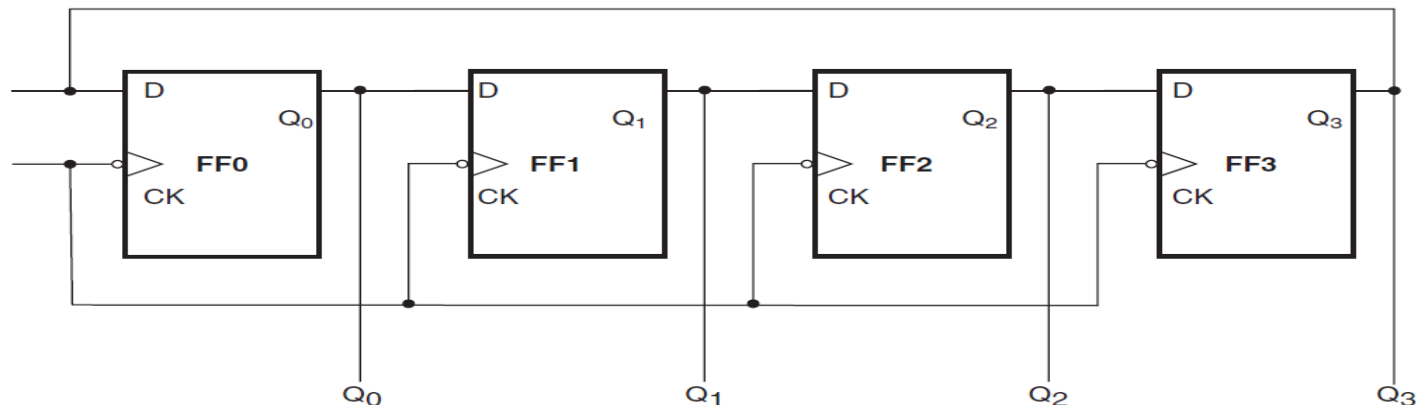
- Shift register can be used to count number of pulses entering into a system as ring counter or switched-tail counter. As ring counter it can generate various control signals in a sequential manner.
- Shift register can also generate a prescribed sequence repetitively or detect a particular sequence from data input. It can also help in reduction of hardware by converting parallel data feed to serial one.
- Serial adder is one such application discussed in this section.

APPLICATIONS OF SHIFT REGISTERS-2

• Ring Counter

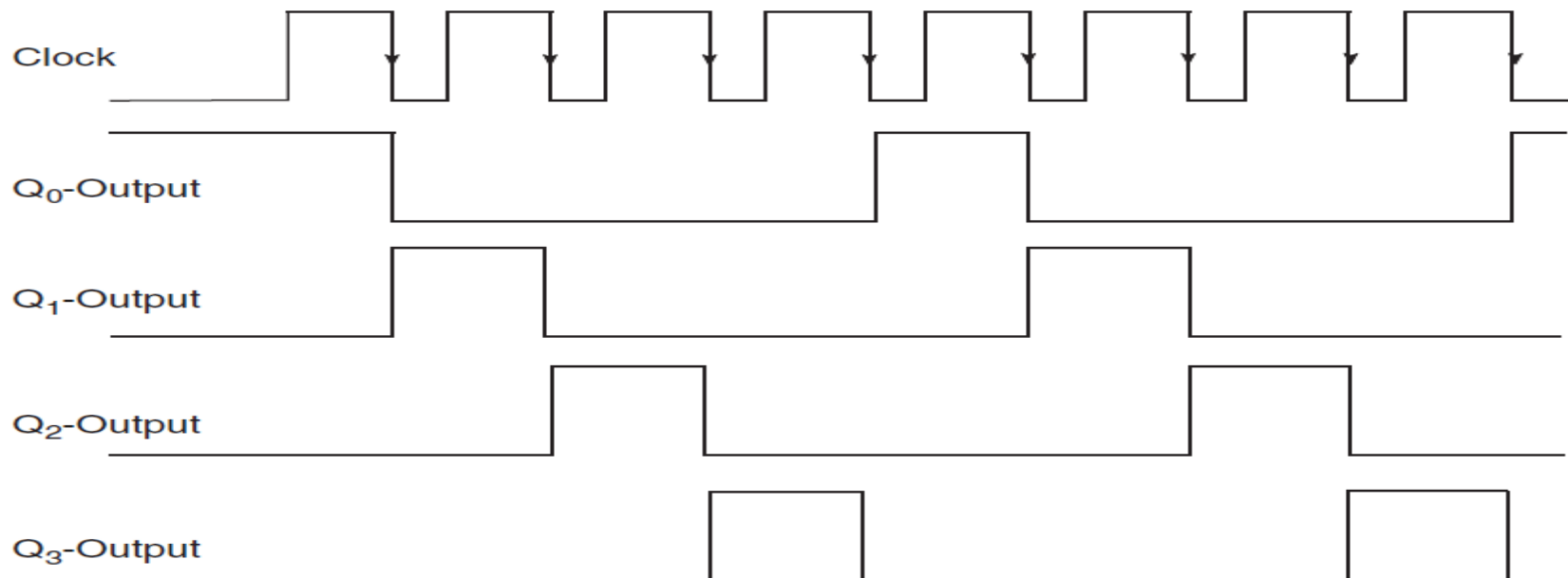
- A *ring counter* is obtained from a shift register by directly feeding back the true output of the output flip-flop to the data input terminal of the input flip-flop.
- If D flip-flops are being used to construct the shift register, the ring counter, also called a circulating register, can be constructed by feeding back the Q output of the output flip-flop back to the D input of the input flip-flop.

Clock Pulse	Q0	Q1	Q2	Q3
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	1	0	0	0
6	0	0	0	
7	0	0	1	0



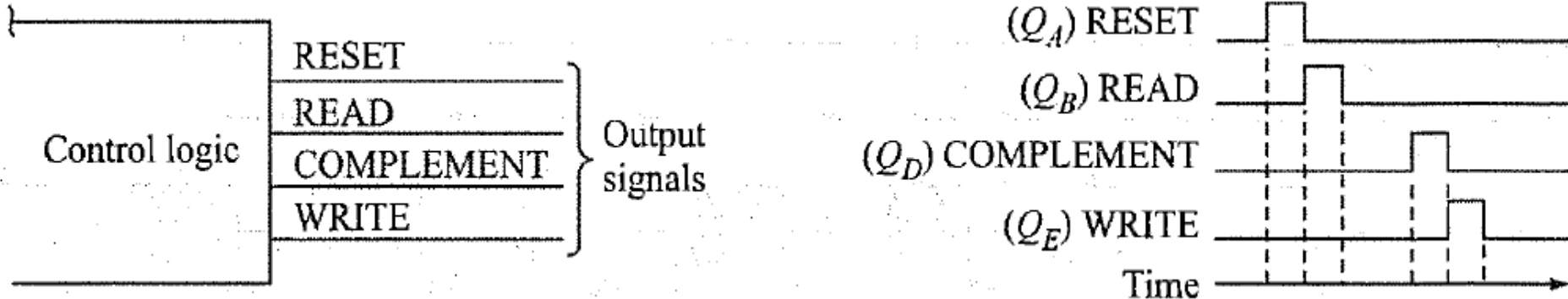
APPLICATIONS OF SHIFT REGISTERS-3

- Flip-flop FF0 is initially set to the logic '1' state and all other flip-flops are reset to the logic '0' state.
- The counter output is therefore 1000. With the first clock pulse, this '1' gets shifted to the second flip-flop output and the counter output becomes 0100.
- Similarly, with the second and third clock pulses, the counter output will become 0010 and 0001. With the fourth clock pulse, the counter output will again become 1000. The count cycle repeats in the subsequent clock pulses.



APPLICATIONS Of SHIFT REGISTERS-4

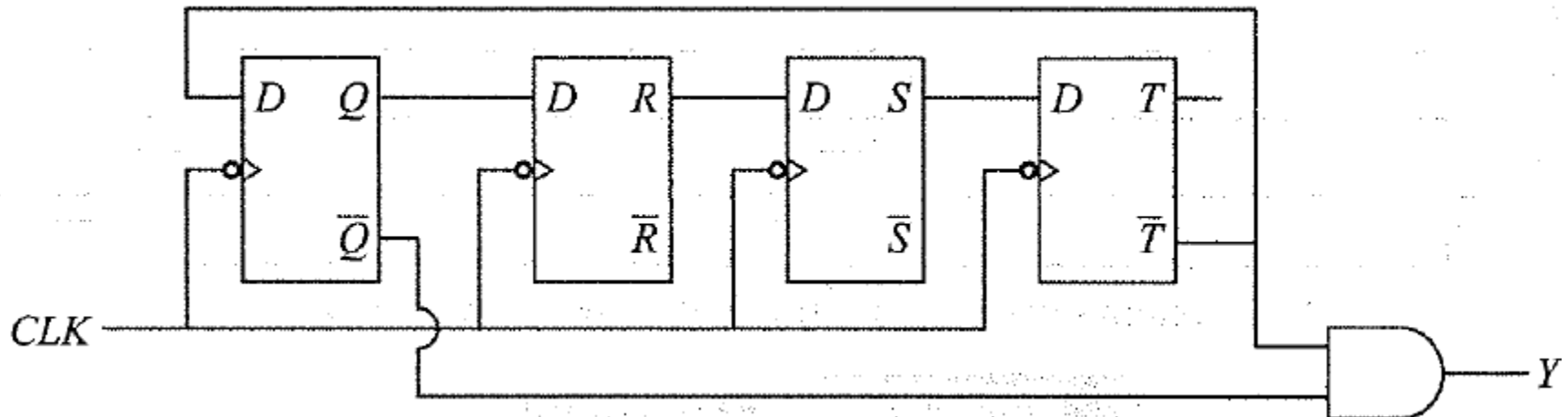
- Waveforms of this type are frequently used in the control section of a digital system. They are ideal for controlling events that must occur in a strict time sequence-that is, event *A*, then event *B*, then *C*, and so on.
- For instance, the logic diagram in figure shows how to generate RESET, READ, COMPLEMENT, and WRITE (a fictitious set of control signals) as a set of control pulses that occur one after the other sequentially.
- There is, however, a problem with such ring counters. In order to produce the waveforms shown in figure, the counter should have one, and only one, 1 in it. The chances of this occurring naturally when power is first applied are very remote indeed. If the flip-flops should all happen to be in the reset state when power is first applied it will not work at all



APPLICATIONS Of SHIFT REGISTERS-5

- **Switched-Tail Counter or Johnson Counter**
- In Ring counter what happens if non-inverting output of the first flip-flop is fed back to first flip flop of the shift register. If we instead feed inverting output back (or switch the tail) as shown in figure for a 4-bit shift register we get *switched tail counter*, also known as *twisted tail counter* or *Johnson counter*.

Clock	Serial in = T	Q	R	S	T	$Y = Q'T'$
0	1	0	0	0	0	1
1	1	1	0	0	0	0
2	1	1	1	0	0	0
3	1	1	1	1	0	0
4	0	1	1	1	1	0
5	0	0	1	1	1	0
6	0	0	0	1	1	0
7	0	0	0	0	1	0
8	1	0	0	0	0	1
9	1	1	0	0	0	0
						repeats



APPLICATIONS OF SHIFT REGISTERS-6

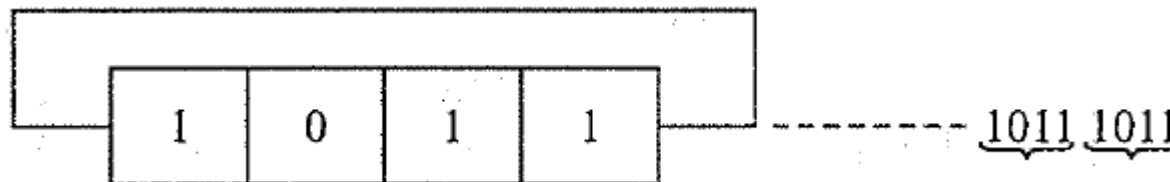
- From state table similar. Assume all the flip-flops are cleared in the beginning. Then all the flip-flop inputs have 0 except the first one, *serial data* in which is complement of the last flip-flop, i.e. 1.
- When clock trigger occurs flip-flop stores $QRST$ as 1000. This makes 1100 at the input of $QRST$ when the next clock trigger comes and that gets transferred to output at NT.
- Note that output $Y = Q'T'$ and state of the circuit repeats every eighth clock cycle. Thus this 4-bit shift register circuit can count 8 clock pulses or called modulo-8 counter.
- Following above logic and preparing state table for any N-bit shift register we see switched-tail configuration can count up to $2N$ number of clock pulse and gives modulo- $2N$ counter. The output Y , derived similarly by AND operation of first and last flip-flop inverting outputs gives a logic high at every $2N$ -th clock cycle.
- This two-input AND gate which decodes states repeating in the memory units to generate output that signals counting of a given number of clock pulses is called decoding gate.

APPLICATIONS Of SHIFT REGISTERS-7

- **Sequence Generator and Sequence Detector**
- Sequence generator is useful in generating a sequence pattern repetitively.
- It may be the synchronizing hit pattern sent by a digital data transmitter or it may be a control word directing repetitive control task.
- Sequence detector checks binary data stream and generates a signal when a particular sequence is detected.

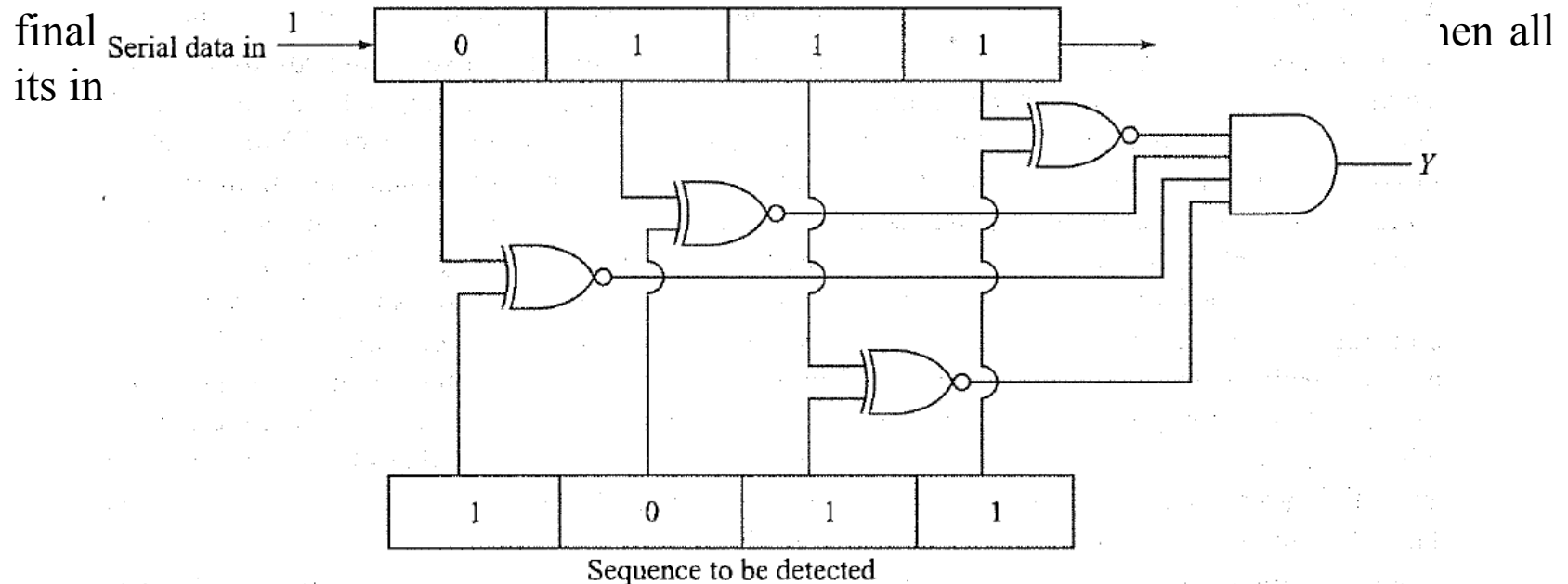
APPLICATIONS Of SHIFT REGISTERS-8

- **Sequence Generator**
- Figure gives the basic block diagram of a sequence generator where shift register is presented as pipe full of data and each flip-flop represents one compartment of it.
- The leftmost flip-flop is connected to serial data in and rightmost provides serial data out. The clock is implied and data transfer takes place only when a clock trigger arrives.
- Note that the shift register is connected like a ring counter and with triggering of clock the binary word stored in the clock comes out sequentially from serial out but does not get lost as it is fed back as serial in to fill the register all over again. Sequence generated for binary word 1011 is shown in the figure and for any n -bit long sequence to be generated for this configuration we need to store the sequence in an n -bit shift register.



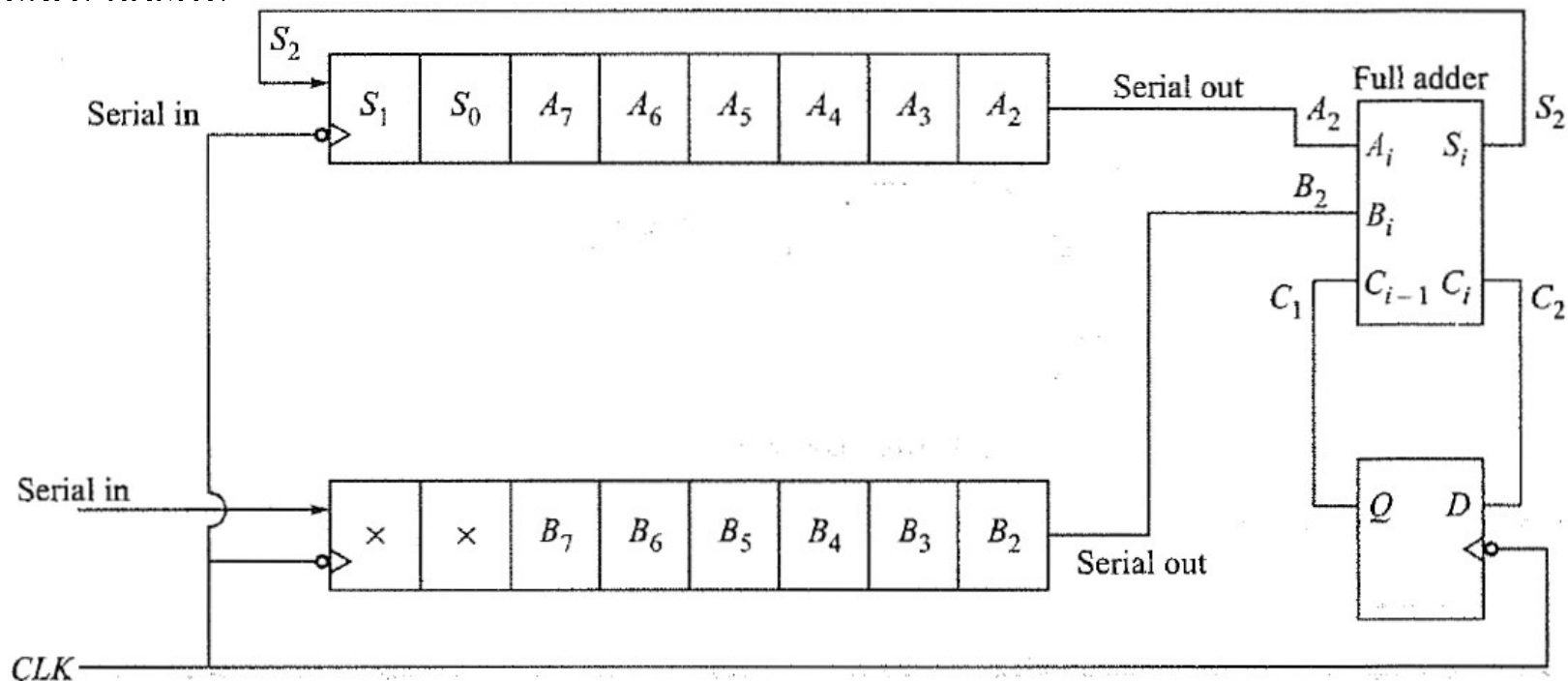
APPLICATIONS Of SHIFT REGISTERS-9

- **Sequence detector**
- The circuit that can detect a 4-bit binary sequence is shown in figure. It has one register to store the binary word we want to detect from the data stream. Input data stream enters a shift register as serial data in and leaves as serial out.
- At every clocking instant, bit-wise comparisons of these two registers are done through Ex-NOR gate as shown in the figure. Two input Ex-NOR gives logic high when both inputs are low or both of them are high, i.e. when both are equal. The



APPLICATIONS OF SHIFT REGISTERS-10

- **Serial Adder**
- For 8 bit full adder (FA) circuit need 8 FA units. There the addition is done in parallel. Using shift register we can convert this parallel addition to serial one and reduce number of FA units to only one. The benefit of this technique is more pronounced if the hardware unit that's needed to be used in parallel is very costly.



APPLICATIONS OF SHIFT REGISTERS-11

- The LSBs of two numbers (A_0 and B_0) appearing at serial out of respective registers are added by FA during 1st clock cycle and generate sum (S_0) and carry (C_0). S_0 is available at serial data input of register A and C_0 at input of D flip-flop.
- At NT of clock shift registers shift its content to right by one unit. S_0 becomes MSB of A and C_0 appears at D flip-flop output. Therefore in the second clock cycle FA is fed by second bit (A_1 and B_1) of two numbers and previous carry (C_0).
- In second clock cycle, S_1 and C_1 are generated and made available at serial data in of A register and input of D flip-flop respectively. At NT of clock S_1 becomes MSB of A and S_0 occupies next position. A_2 and B_2 now appear at FA data input and carry input is C_1 .
- In 3rd clock cycle, S_2 and C_2 are generated and they get transferred similarly to register and flip-flop. This process goes on and is stopped by inhibiting the clock after 8 clock cycles. At that time shift register A stores the sum bits, S_7 in leftmost (MSB) position and S_0 in rightmost (LSB) position. The final carry is available at D flip-flop output.
- The limitation of this scheme is that the final addition result is delayed by eight clock cycles. In parallel adder the result is obtained almost instantaneously, after nanosecond order propagation delay of combinatorial circuit. However, using a high frequency clock the delay factor can be reduced considerably.

REGISTER IMPLEMENTATION IN HDL-1

The PIPO, When Clear is activated (active LOW) all 6 outputs of Q are reset.

```

module regpipo
(D,clock,clear,Q);
input Clock, clear;
input [5:0] D;
output [5:0] Q;
reg [5:0] Q;
always@ (negedge Clock
or negedge Clear)
if (~Clear) Q=6'b0;
//Q stores 6 binary 0
else Q=D;
endmodule

```

Shift right register , where T is the final output and Q, R, S are internal outputs.

```

module SRreg (D, Clock, T);
input Clock, D;
output T;
reg T;
reg Q,R,S;
always @ (negedge Clock)
begin
    Q<=D;
    R<=Q;
    S<=R;
    T<=S;
end
endmodule

```

SIPO

```

module SR2(D,Clock,Q);
input Clock, D;
output [3:0] Q;
reg [3 :0] Q;
always @ (negedge clock)
begin
    Q[0]<=D;
    Q[1]<=Q[0];
    Q[2]<=Q[1];
    Q[3]<=Q[2];
end
endmodule

```

REGISTER IMPLEMENTATION IN HDL-2

- Assignment operator `<=` within **always** block which unlike `=` operator executes all associated statements concurrently.
- Assignment operator must start with **begin**

REGISTER IMPLEMENTATION IN HDL-3

- Write Verilog code for switched tail counter

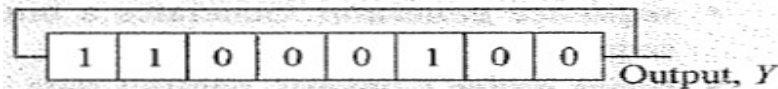

```

module STC(Clock,Clear,Y); //Switched Tail Counter
input Clock, Clear;
output Y;
reg Q,R,S,T; //internal outputs of flip-flops
assign Y= (~Q)&(~T);
always@ (negedge Clock)
begin
if (~Clear) Q=6'b0; //Q stores 6
else
begin
Q <= ~T; //Tail is switched and connected to input
R <= Q;
S <= R;
T <= S;
end
endmodule

```

PROBLEM SOLVING WITH MULTIPLE METHODS

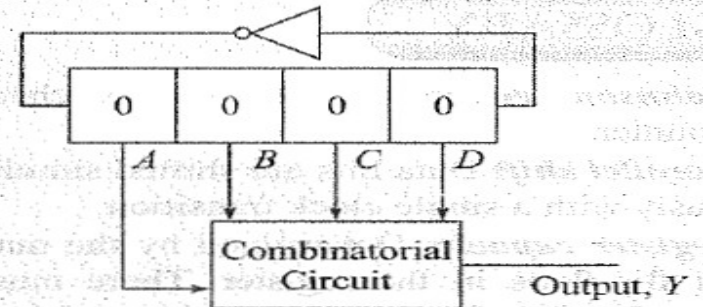
- Design an 8-bit sequence generator that generates the sequence 11000100 repetitively using shift register.



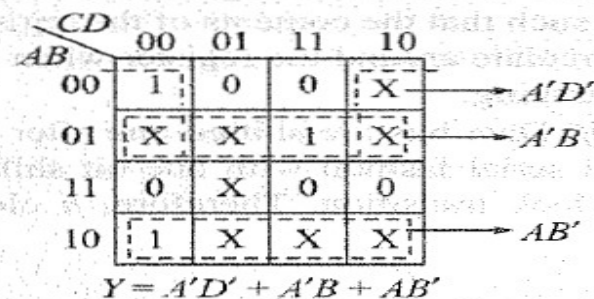
(a)

Counter				Sequence Generator
A	B	C	D	Output, Y
0	0	0	0	1
1	0	0	0	1
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0
0	1	1	1	1
0	0	1	1	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	0
...repeats...				

(c)



(b)



(d)

(a) Solution with Method-1, (b)-(d) Solution with Method-2, (b) Targeted realization, (c) Counter sequence vs. sequence generator output, (d) Karnaugh Map to generate logic equation

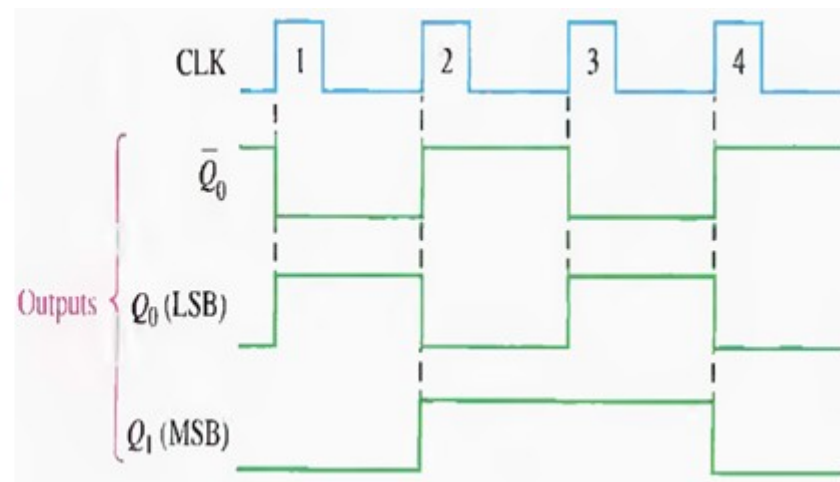
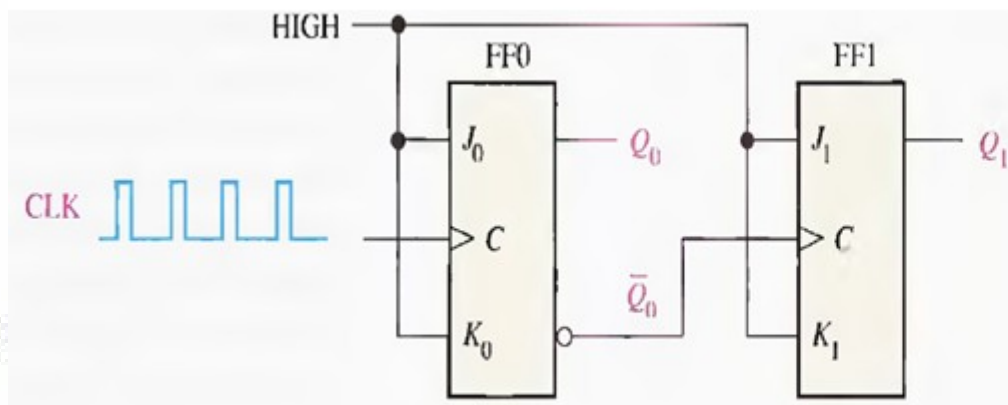
COUNTERS

Introduction

- A counter driven by a clock can be used to count the number of clock cycles.
- Since the clock pulses occur at known intervals, the counter can be used as an instrument for measuring time and therefore period or frequency.
- There are basically two different types of counters-synchronous and asynchronous.
- Serial, or asynchronous counter is defined as each flip-flop is triggered by the previous flip-flop, and thus the counter has a cumulative settling time.
- An increase in speed of operation can be achieved by use of a parallel or synchronous counter. Here, every flip-flop is triggered by the clock (in synchronism), and thus settling time is simply equal to the delay time of a single flip-flop.
- Serial and parallel counters are used in combination to compromise between speed of operation and hardware count.

ASYNCHRONOUS COUNTERS-1

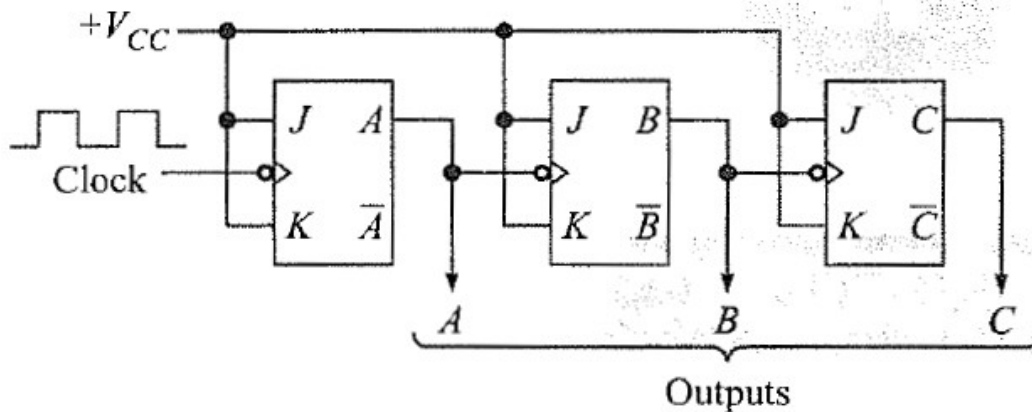
- The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time.
- An asynchronous counter is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse.
- **A 2-Bit Asynchronous Binary Counter**
- Figure shows a 2-bit counter connected for asynchronous operation. Notice that the clock (CLK) is applied to the clock input (C) of only the first flop-flop, FF0, which is always the least significant bit (LSB). The second flip-flop, FF1, is triggered by the out-put of FF0. FF0 changes state at the positive-going edge of each clock pulse, but FF1 changes only when triggered by a negative-going transition of the Q0 output of FF0.



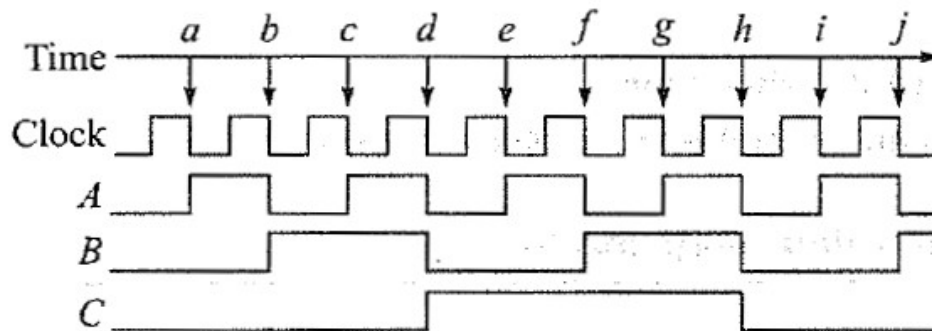
ASYNCHRONOUS COUNTERS-2

- **Ripple Counters (Up Counter)**
- Figure shows (Next slide) three negative edge- triggered, JK flip-flops connected in cascade.
- The system clock, a square wave, drives flip-flop A . The output of A drives B , and the output of B drives flip-flop C . All the J and K inputs are tied to $+V_{CC}$. This means that each flip-flop will change state (toggle) with a negative transition at its clock input.
- When the output of a flip-flop is used as the clock input for the next flip-flop, we call the counter a *ripple counter*, or *asynchronous counter*.
- The A flip-flop must change state before it can trigger the B flip-flop, and the B flip-flop has to change state before it can trigger the C flip-flop.
- If each flip-flop in this three-flip-flop counter has a propagation delay time of 10 ns, the overall propagation delay time for the counter is 30 ns.

ASYNCHRONOUS COUNTERS-3



(a) Three-bit binary ripple counter



(b) Waveforms

Negative clock transitions	C	B	A	State or count
---	0	0	0	0
a	0	0	1	1
b	0	1	0	2
c	0	1	1	3
d	1	0	0	4
e	1	0	1	5
f	1	1	0	6
g	1	1	1	7
h	0	0	0	0

(c) Truth table

ASYNCHRONOUS COUNTERS-4

- The waveform at the output of flip-flop A is one-half the clock frequency.
- The waveform at the output of flip-flop B is one-half the frequency of A and one-fourth the clock frequency.
- The frequency of the waveform at C is one-half that at B , but it is only one-eighth the clock frequency.
- What is the clock frequency, if the period of the waveform at C is $24 \mu\text{s}$?
- Since there are eight clock cycles in one cycle of C , the period of the clock must be $24/8 = 3 \mu\text{s}$. The clock frequency must then be $1/(3 \times 10^{-6}) = 333 \text{ kHz}$.

ASYNCHRONOUS COUNTERS-5

- A binary ripple counter counts in a straight binary sequence, a counter having n flip-flops will have 2^n output conditions.
- For instance, the three-flip-flop counter just discussed has $2^3 = 8$ output conditions (000 through 111). Five flip-flops would have $2^5 = 32$ output conditions (00000 through 11111), and so on.
- The largest binary number that can be represented by n cascaded flip-flops has a decimal equivalent of 2^{n-1} .
- For example, the three-flip-flop counter reaches a maximum decimal number of 2^{3-1} .
- The maximum decimal number for five flip-flops is $2^{5-1} = 31$, while six flip-flops have a maximum count of 63.
- A three-flip-flop counter is often referred to as a modulus-8 (or mod-8) counter since it has eight states. Similarly, a four-flip-flop counter is a mod-16 counter, and a six-flip-flop counter is a mod-64 counter.
- The *modulus* of a counter is the total number of states through which the counter can progress.

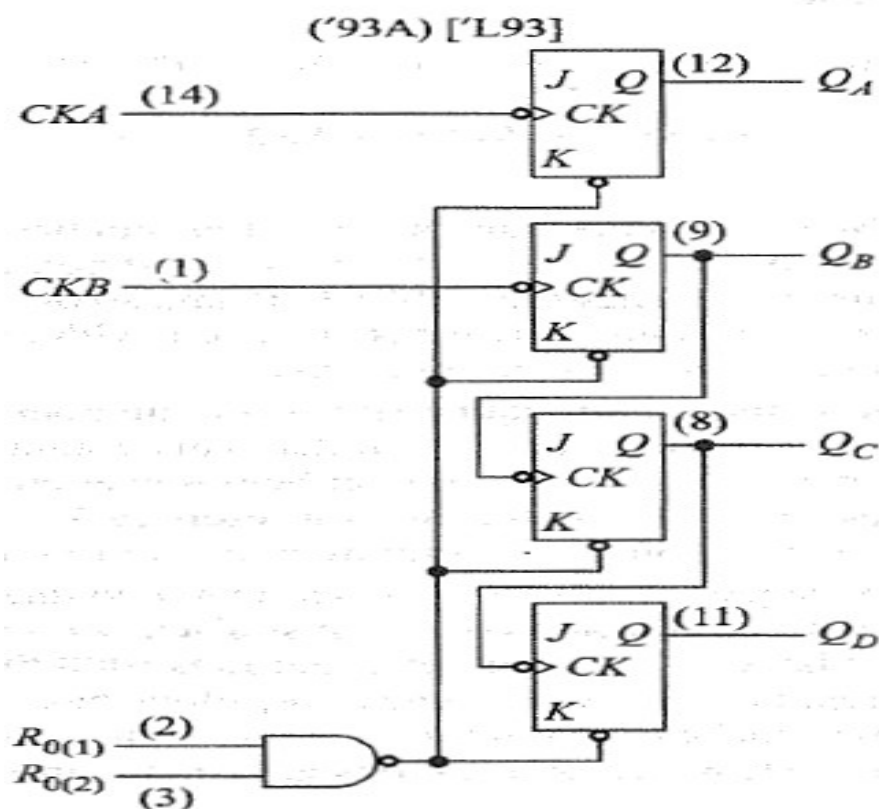
ASYNCHRONOUS COUNTERS-6

- How many flip-flops are required to construct a mod-128 counter? A mod-32? What is the largest decimal number that can be stored in a mod-64 counter?
- A mod-128 counter must have seven flip-flops, since $2^7 = 128$.
- Five flip-flops are needed to construct a mod-32 counter.
- The largest decimal number that can be stored in a six-flop flip counter (mod-64) is $111111 = 63$.
- Note carefully the difference between the modulus (total number of states) and the maximum decimal number.

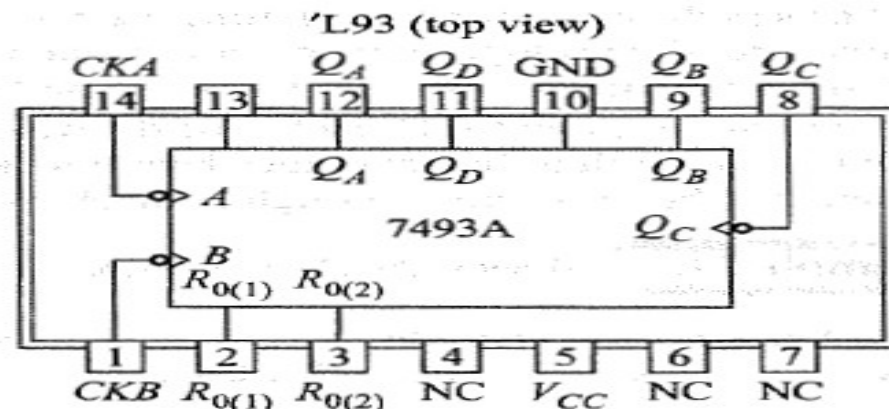
ASYNCHRONOUS COUNTERS-8

- **The 54/7493A**
- A 4-bit binary counter that can be used in either a mod-8 or a mod-16 configuration. If the clock is applied at input CKB , the outputs will appear at QB , QC , and QD , and this is a mod-8 binary ripple counter.
- If the clock is applied at input CKA and flip-flop QA is connected to input CKB , have a mod-16, 4-bit binary ripple counter. The outputs are QA , QB , QC , and QD .

ASYNCHRONOUS COUNTERS-9



(a) Logic diagram



Positive logic: see function tables
NC - No internal connection

(b) DIP pinout

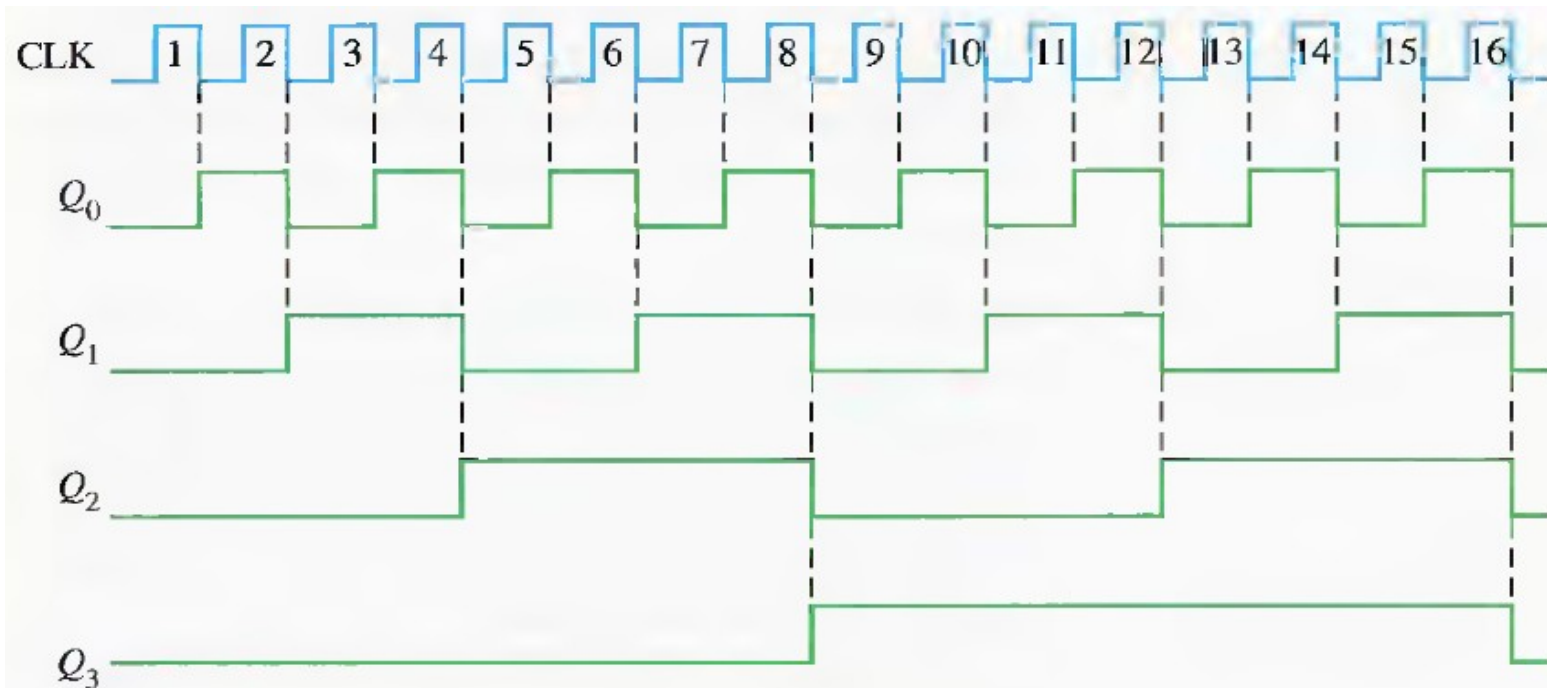
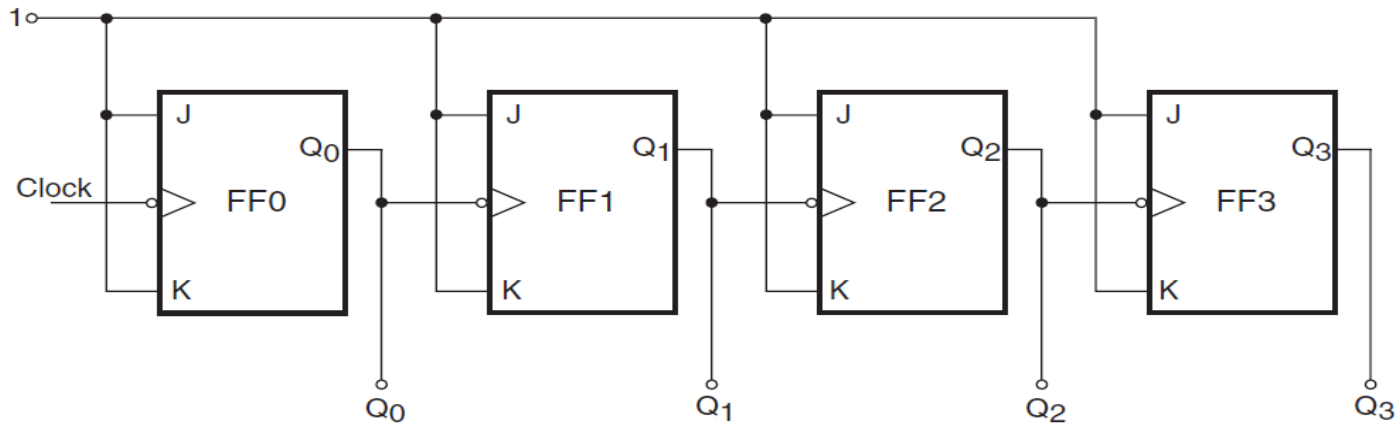
'L93A, 'L93, 'LS93 Count sequence

Count	Output			
	Q _D	Q _C	Q _B	Q _A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

(c) Truth table

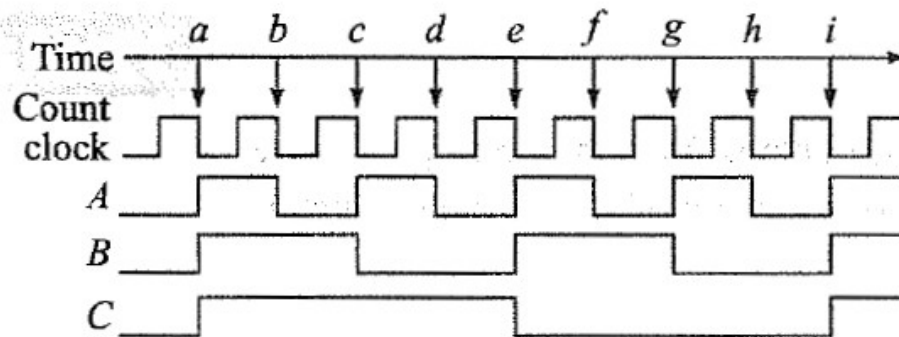
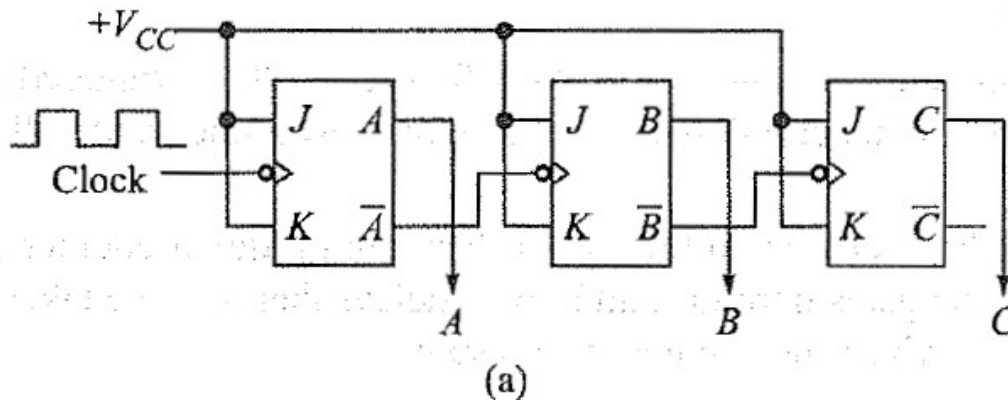
ASYNCHRONOUS COUNTERS-8

- Example



ASYNCHRONOUS COUNTERS-9

- **Ripple Counters (Down Counter)**
- In 3-bit ripple up counter the system clock is still used at the clock input to flip-flop A , but the complement of A , A' , is used to drive flip-flop B , likewise; B' is used to drive flip-flop C .

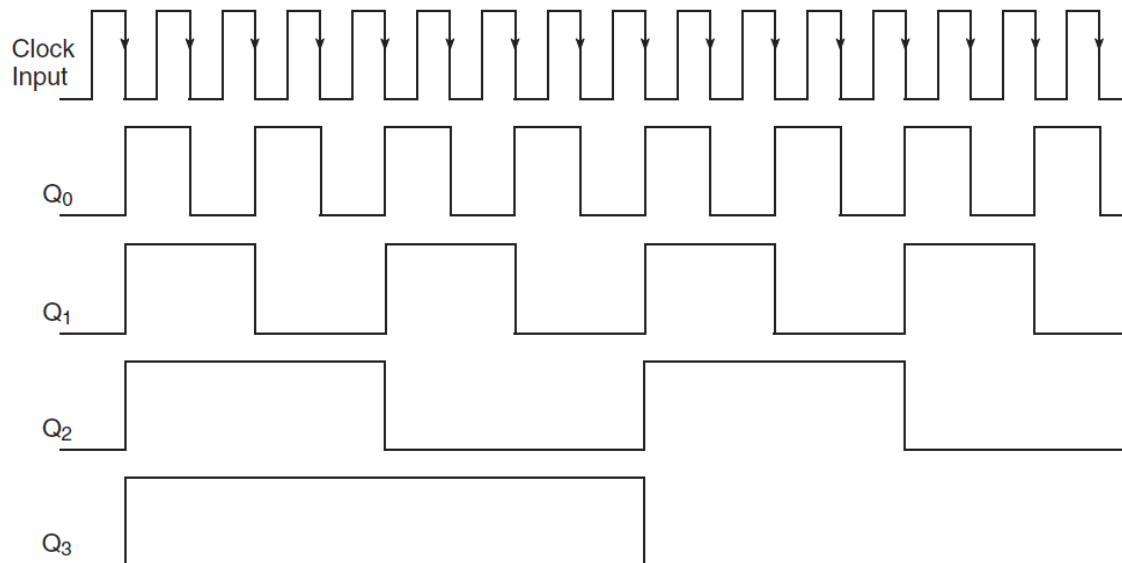
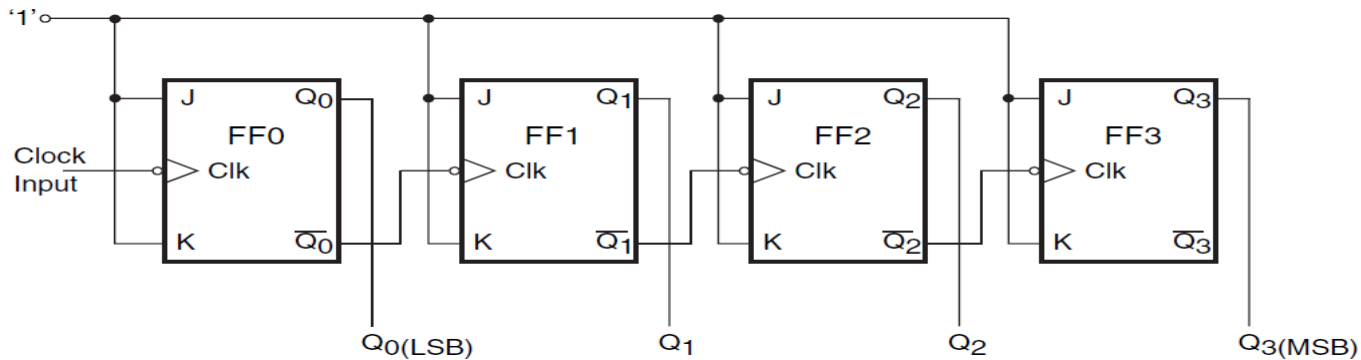


Count	C	B	A
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0
7	1	1	1

(c)

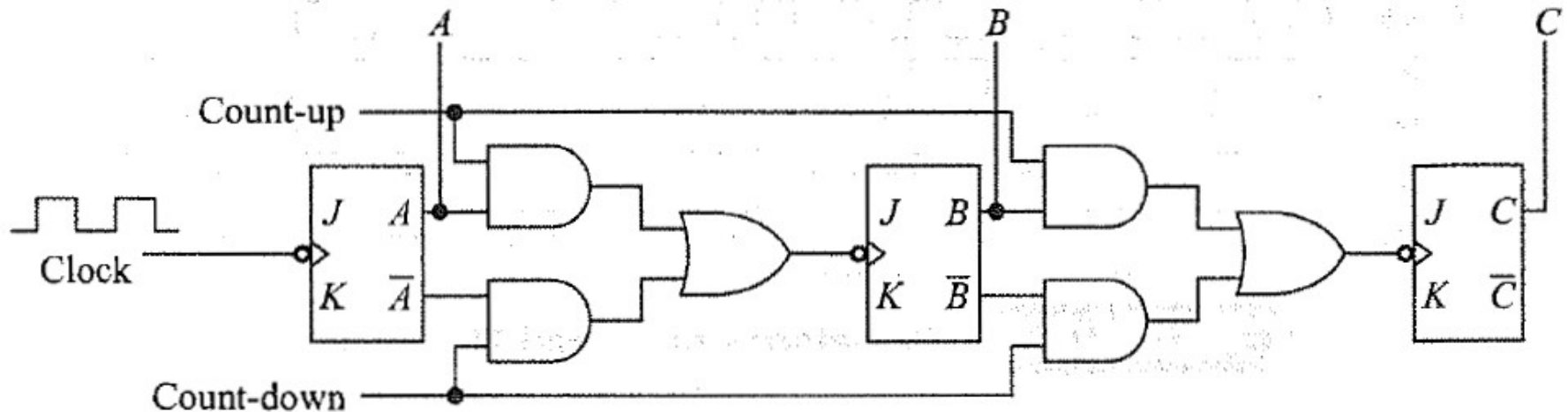
ASYNCHRONOUS COUNTERS-10

- Ripple Counters (4-bit Down Counter)**



ASYNCHRONOUS COUNTERS-11

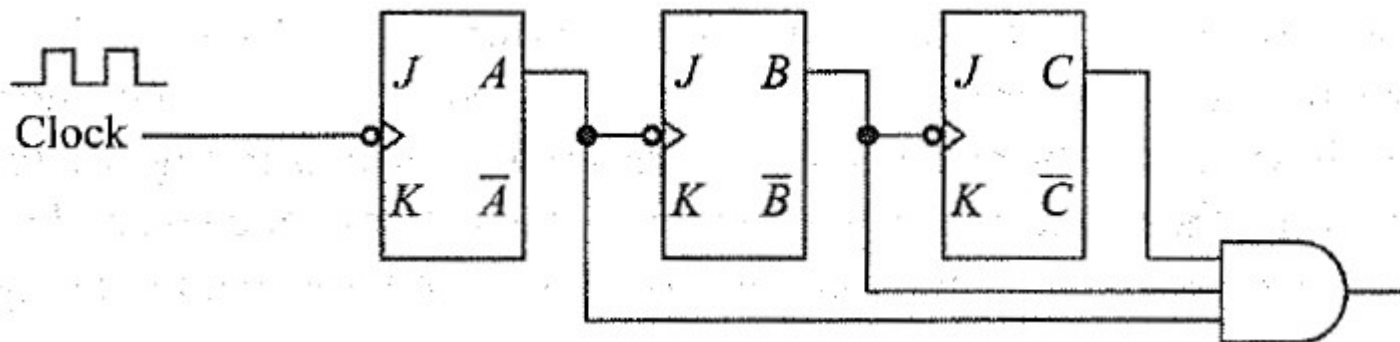
- **Asynchronous up-down counter**
- Count-up=1 works as Up counter
- Count-down=1 works as down counter
- Both should not set to 1



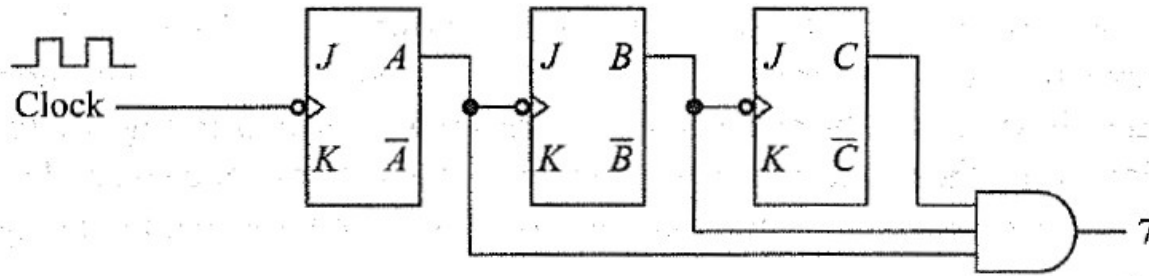
Note: The J and K inputs are all tied to $+V_{CC}$.
The counter outputs are A , B , and C .

DECODING GATES-1

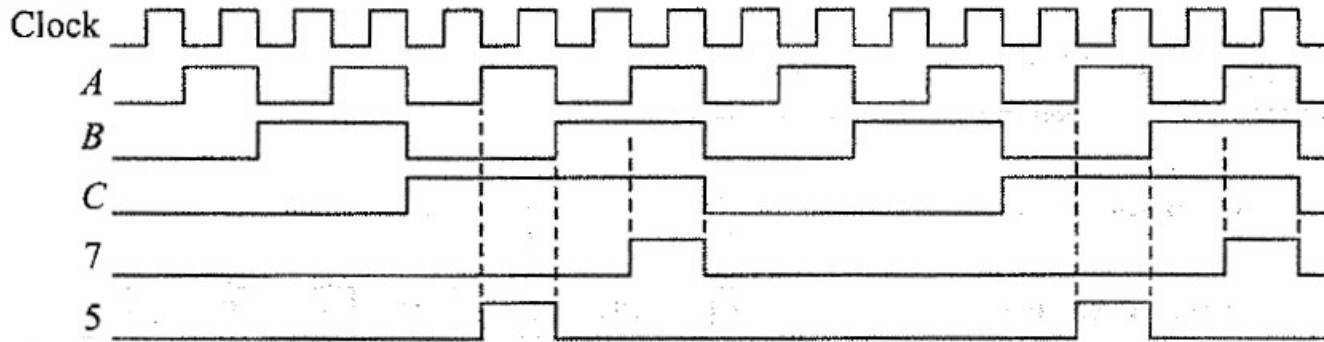
- A **decoding gate** can be connected to the outputs of a counter in such a way that the output of the gate will be high (or low) only when the counter contents are equal to a given state.
- For instance, the decoding gate connected to the 3-bit ripple counter in below figure will decode state 7 ($CBA = 111$). Thus the gate output will be high only when $A = 1$, $B = 1$, and $C = 1$ and the waveform appearing at the output of the gate is labelled 7. The Boolean expression for this gate can be written $7 = CBA$.



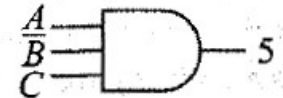
DECODING GATES-2



(a) Decoding gate for state 7



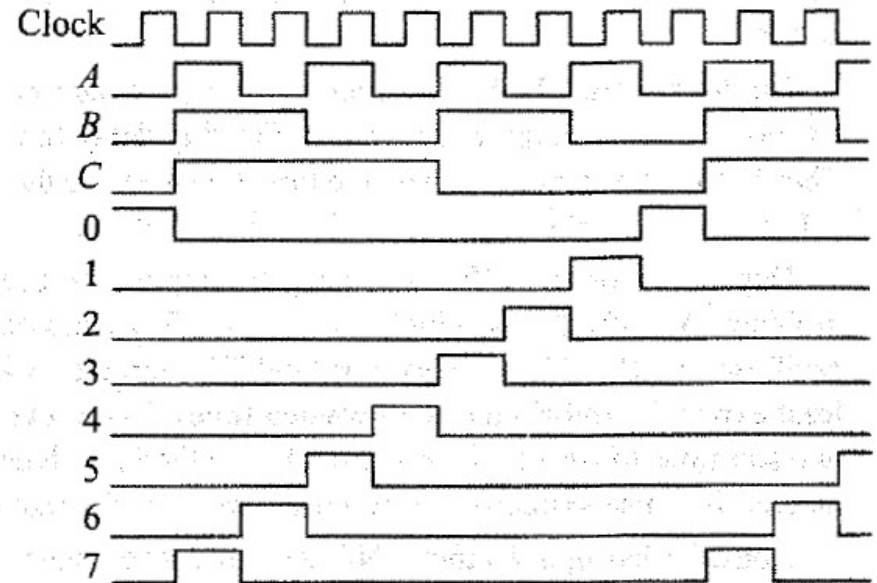
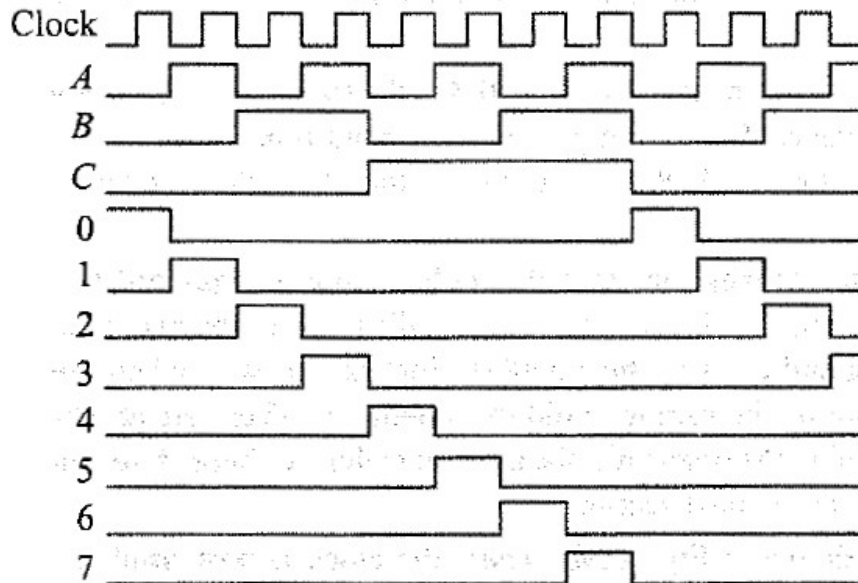
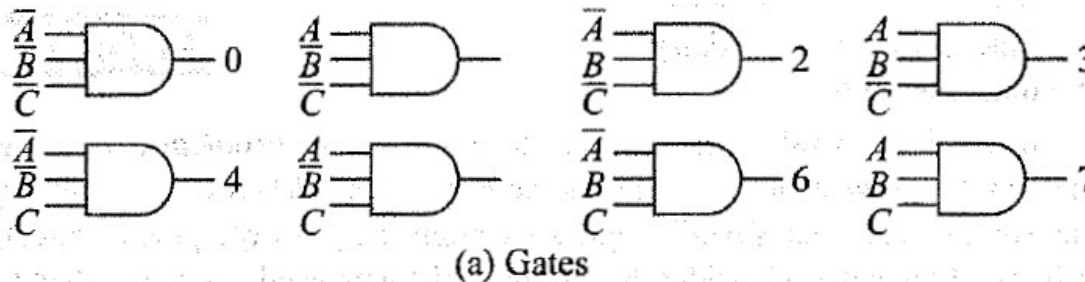
(b) Waveforms



(c) Gate to decode state 5

DECODING GATES-3

- Decoding gates for a 3-bit binary ripple counter



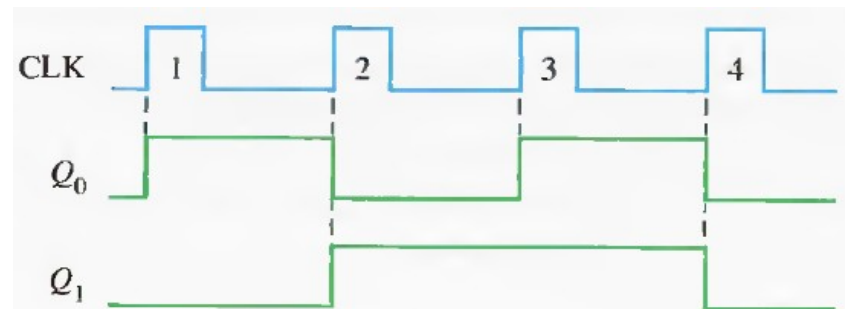
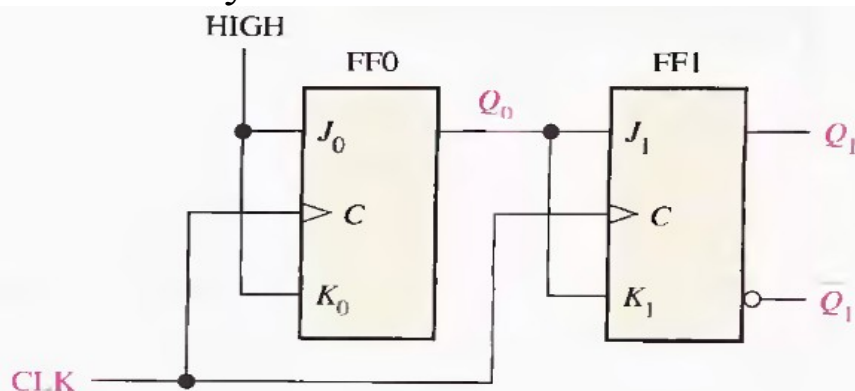
SYNCHRONOUS COUNTERS-1

- The ripple counter is the simplest to build, but there is a limit to its highest operating frequency. As each flip-flop has a delay time.
- In a ripple counter these delay times are additive, and the total "settling" time for the counter is approximately the delay time times the total number of flip-flops.
- There is the possibility of glitches occurring at the output of decoding gates used with a ripple counter.
- To overcome by the use of a synchronous parallel counter.
- The main difference here is that every flip-flop is triggered in synchronism with the clock.
- The term synchronous refers to events that have a fixed time relationship with each other.
- A synchronous counter is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse.

SYNCHRONOUS COUNTERS-2

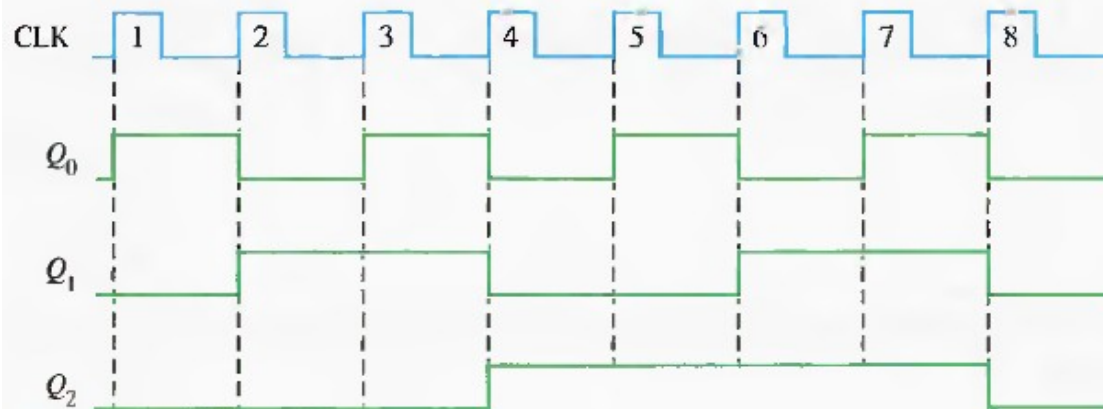
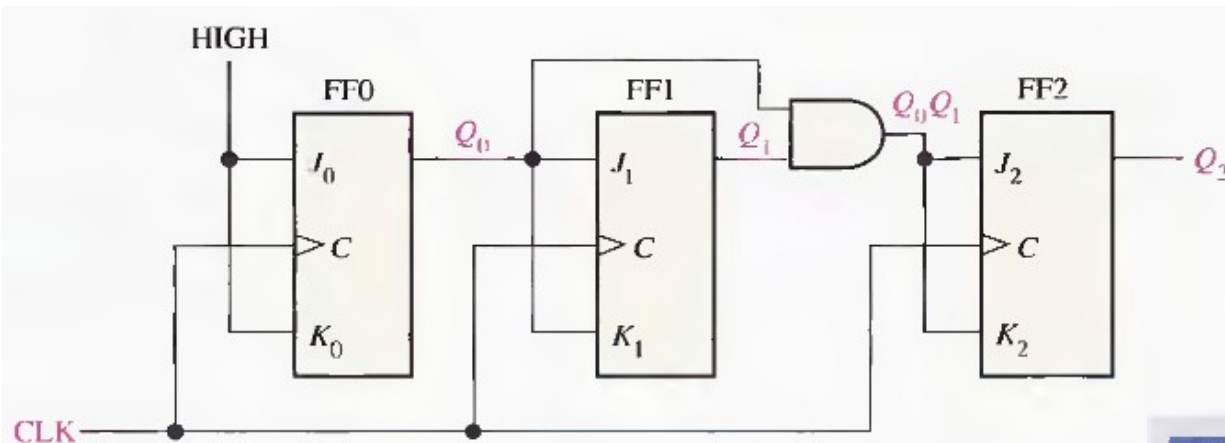
• A 2-Bit Synchronous Binary Counter

- Notice that an arrangement different from that for the asynchronous counter must be used for the J and K inputs of FF1 in order to achieve a binary sequence.
- The operation of this synchronous counter is as follows: First, assume that the counter is initially in the binary 0 state that is both flip-flops are RESET. When the positive edge of the first clock pulse is applied, FF0 will toggle and Q_0 will therefore go HIGH. What happens to FF1 at the positive-going edge of CLK1? To find out, let's look at the input conditions of FF1. Inputs J_1 and K_1 are both LOW because Q_0 , to which they are connected, has not yet gone HIGH. Remember, there is a propagation delay from the triggering edge of the clock pulse until the Q output actually makes a transition.



SYNCHRONOUS COUNTERS-3

- A 3-Bit Synchronous Binary Counter



CLOCK PULSE	Q_2	Q_1	Q_0
Initially	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 (recycles)	0	0	0

SYNCHRONOUS COUNTERS-4

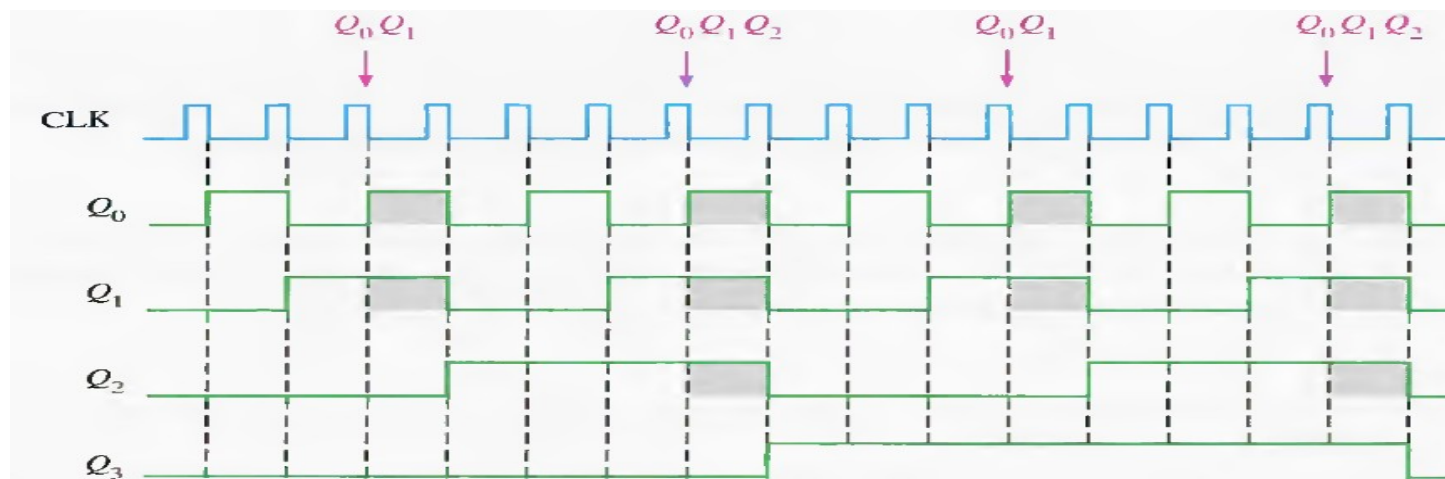
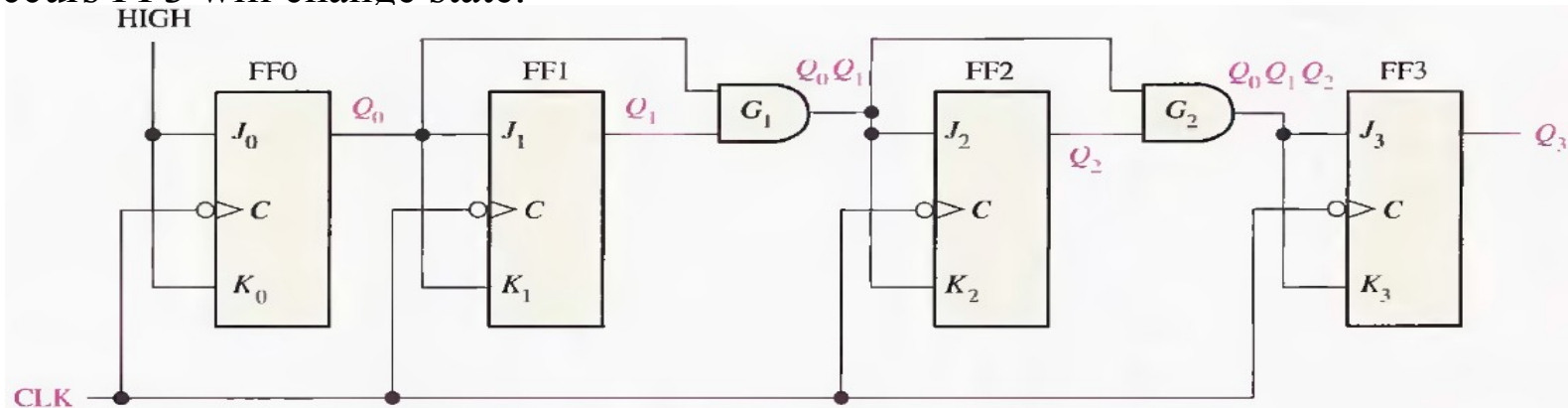
- First, let's look at Q0. Notice that Q0 changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state.
- Next, let's see how FF2 is made to change at the proper times according to the binary sequence. Notice that both times Q2 changes state it is preceded by the unique condition in which both Q0 and Q1 are HIGH.
- This condition is detected by the AND gate and applied to the J2 and K2 inputs of FF2. Whenever both Q0 and Q1 are HIGH, the output of the AND gate makes the J2 and K2 inputs of FF2 HIGH, and FF2 toggles on the following clock pulse.
- At all other times, the J2 and K2 inputs of FF2 are held LOW by the AND gate output, and FF2 does not change state.

SYNCHRONOUS COUNTERS-4

- **Each flip-flop should have its J and K inputs connected such that they are HIGH only when the outputs of all lower order flip-flops are in the HIGH state.**

SYNCHRONOUS COUNTERS-5

- A 4-Bit Synchronous Binary Counter
- Notice that both of these transitions occur following the times that Q_0 , Q_1 and Q_2 are all HIGH. This condition is decoded by AND gate G_2 so that when a clock pulse occurs FF3 will change state.

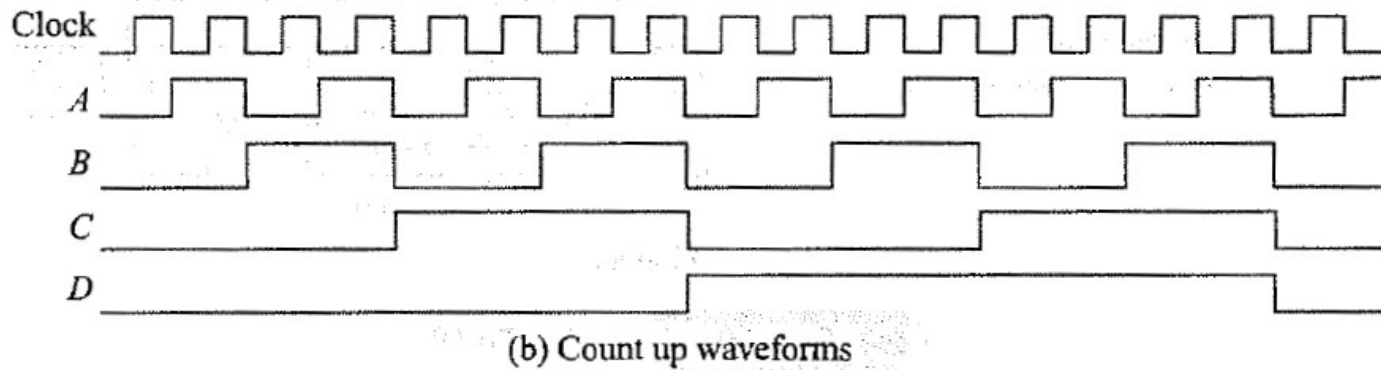
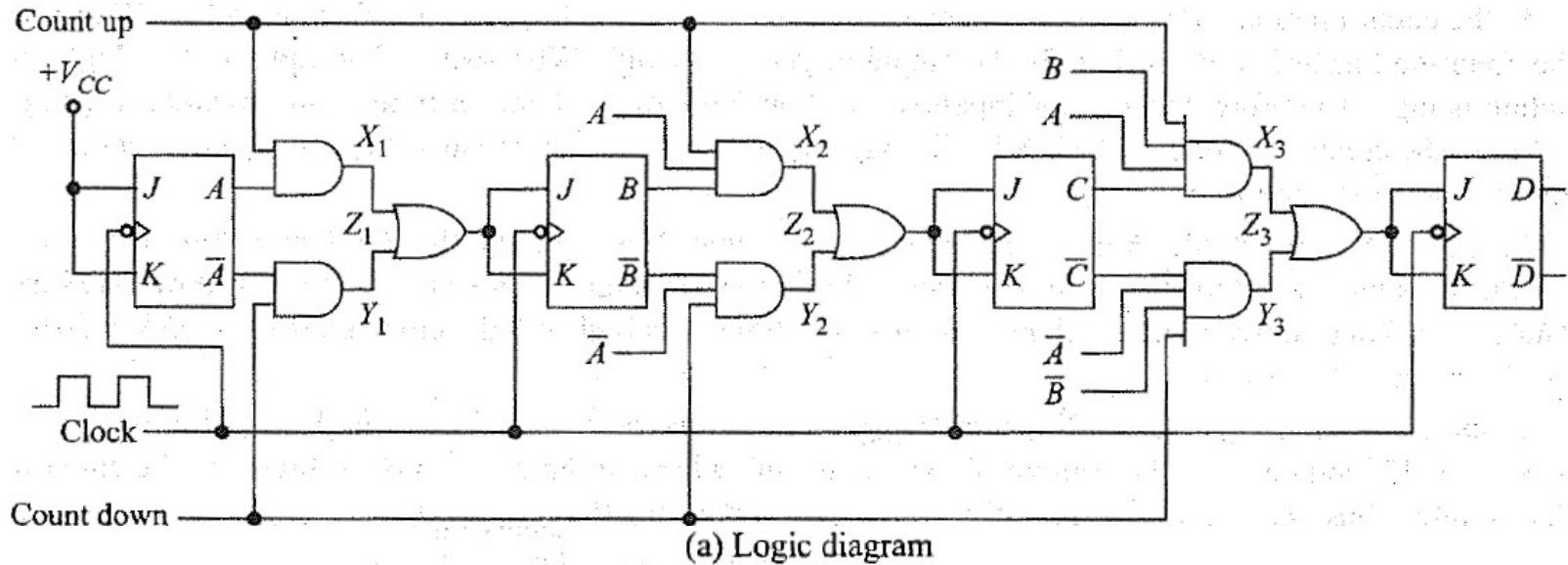


SYNCHRONOUS COUNTERS-6

- **UP/DOWN SYNCHRONOUS COUNTERS**
- Below shows Synchronous 4-bit up-down counter.
- To operate in the count-up mode, the logic High is applied at the count-up input, while the count-down input is held low.
- To operate in the count-down mode, the logic Low is applied at the count-down input while holding the count-up input low.

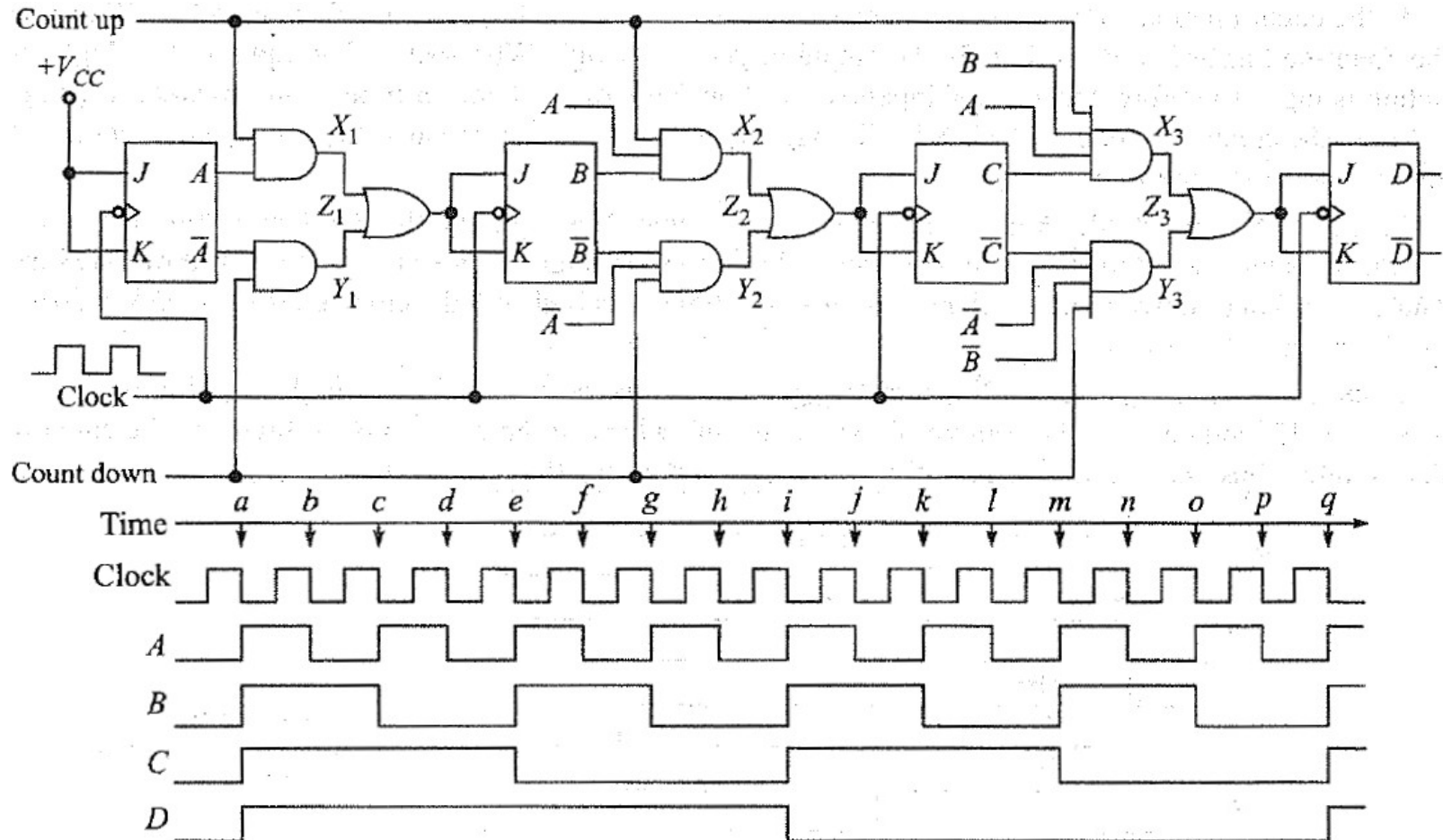
SYNCHRONOUS COUNTERS-7

• UP/DOWN SYNCHRONOUS COUNTERS



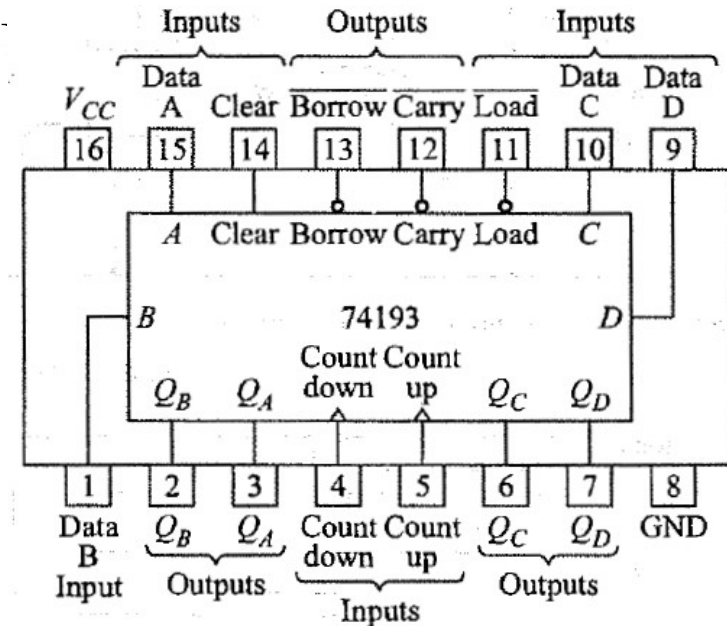
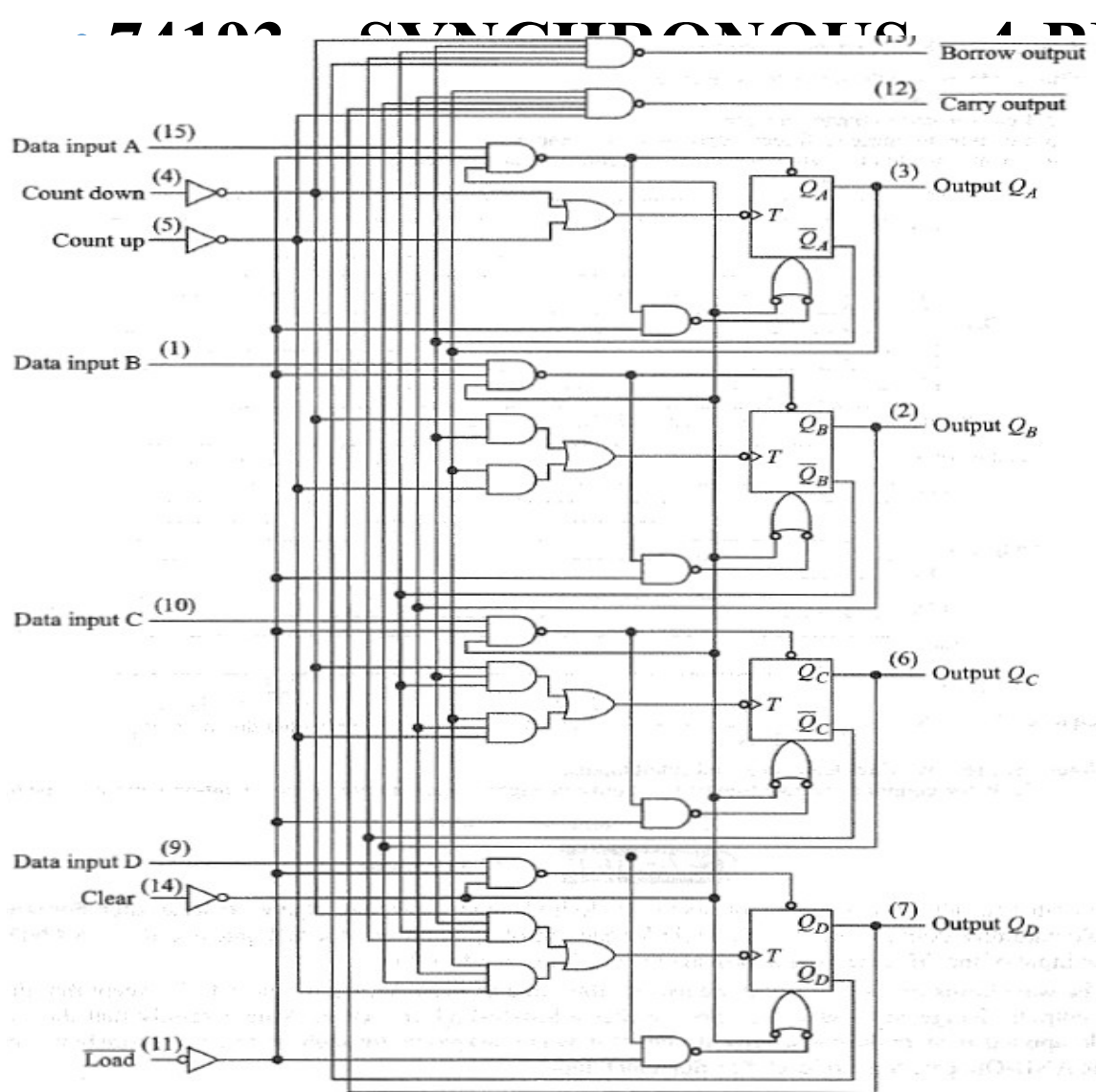
SYNCHRONOUS COUNTERS-8

• UP/DOWN SYNCHRONOUS COUNTERS



(c) Count down waveforms

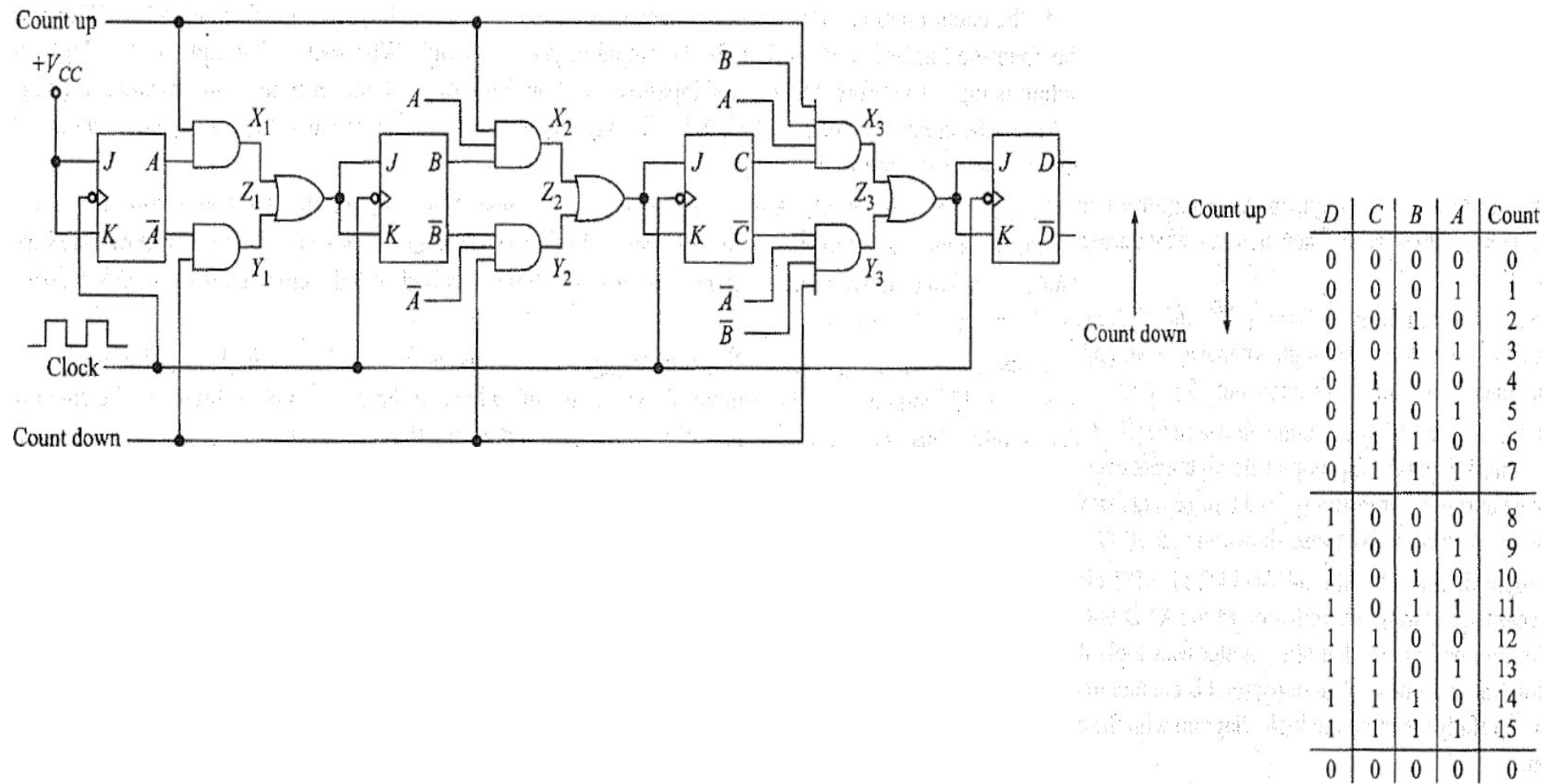
SYNCHRONOUS COUNTERS-8



- Counter that can also be cleared and preset to any desired count-attributes that we discuss later. Now, should carefully examine the steering logic for each flip-flop and study the OR gate and the two AND gates at the input of the OR gate used to provide the clock to each flip-flop.

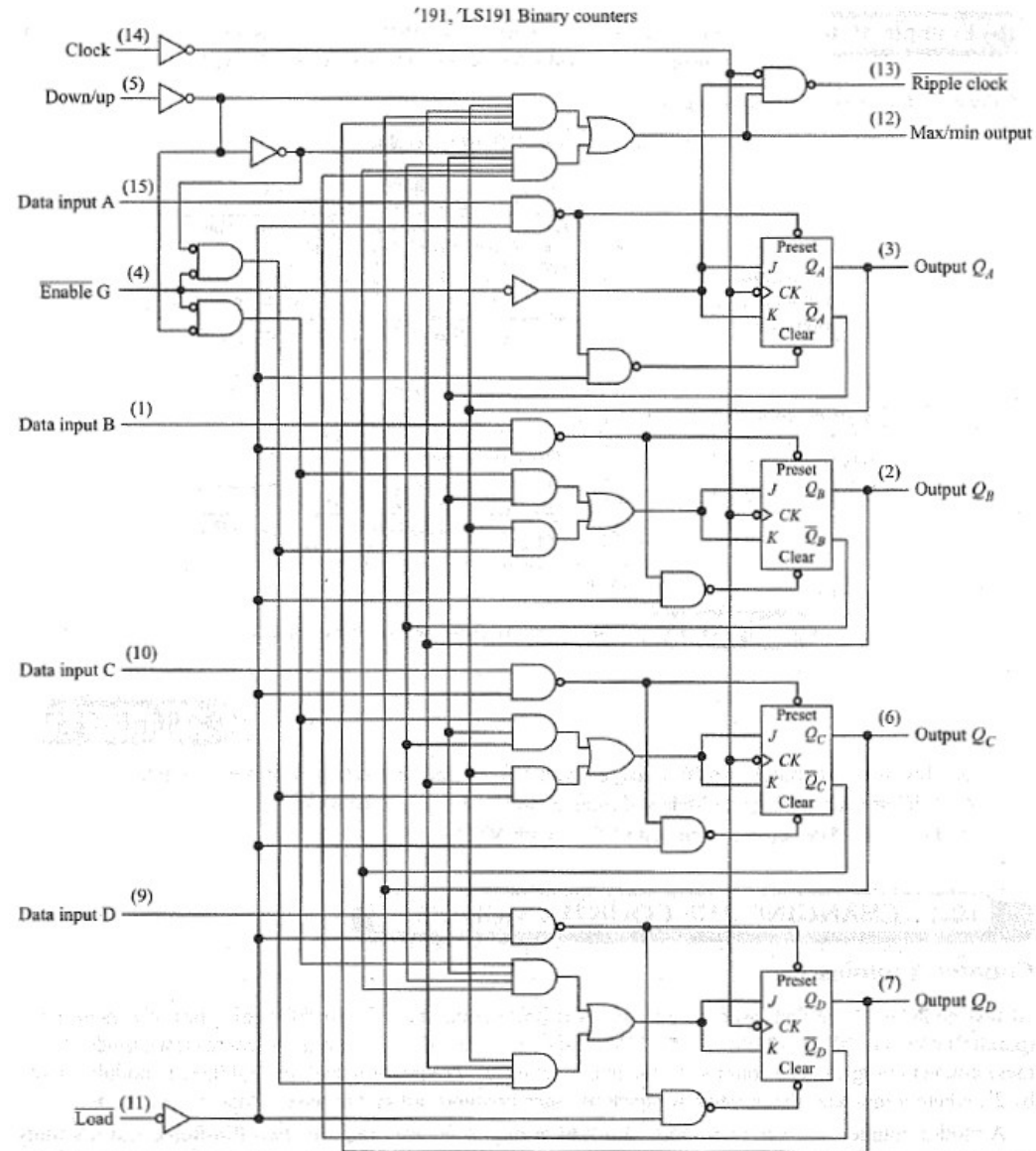
SYNCHRONOUS COUNTERS-9

• Parallel up-down counter



SYNCHRONOUS COUNTERS-9

- **54/74191 synchronous up-down counter**
- A careful examination of the AND-OR-gate logic used to precondition the J and K inputs to each flip-flop will reveal that this counter uses look-ahead logic.
- Additional logic allows one to clear or preset this counter to any desired count.



CHANGING THE COUNTER MODULUS-1

- **Counter Modulus**

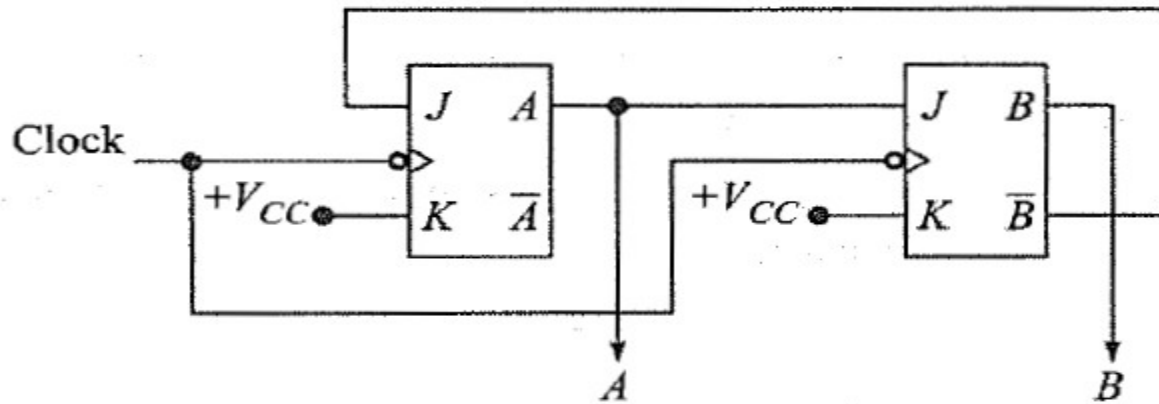
- A modulus given by 2^n , where n indicates the number of flip-flops. Such counters are said to have a "natural count" of 2^n .
- A mod-2 counter consists of a single flip-flop; a mod-4 counter requires two flip-flops, and it counts through four discrete states. Three flip-flops form a mod-8 counter, while four flip-flops form a mod-16 counter.
- Thus we can construct counters that have a natural count of 2, 4, 8, 16, 32, and so on by using the proper number of flip-flops.
- It is often desirable to construct counters having a modulus other than 2, 4, 8, and so on. For example, a counter having a modulus of 3, or 5, would be useful. A small modulus counter can always be constructed from a larger modulus counter by skipping states. Such counters are said to have a *modified count*.
- It is first necessary to determine the number of flip-flops required. The correct number of flip-flops is determined choosing the lowest natural count that is greater than the desired modified count.
- For example, a mod-7 counter requires three flip-flops, since 8 is the lowest natural count greater than the desired modified count of 7.
 - MOD number = 2^N
 - where N is the number of flip-flops connected

CHANGING THE COUNTER MODULUS-2

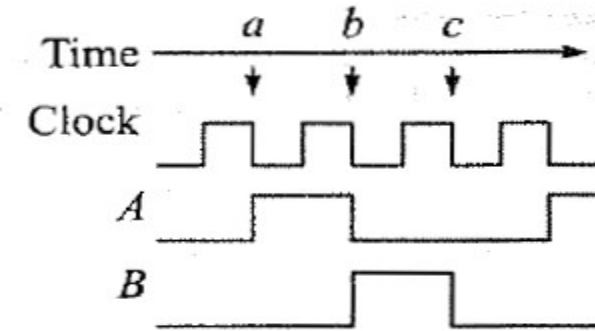
- **Question**
- Indicate how many flip-flops are required to construct each of the following counters: (a) mod-3, (b) mod-6, and (c) mod-9.
- The lowest natural count greater than 3 is 4. Two flip-flops provide a natural count of 4. Therefore, it requires at least two flip-flops to construct a mod-3 counter.
- Construction of a mod-6 counter requires at least three flip-flops, since 8 is the lowest natural count greater than 6.
- A mod-9 counter requires at least four flip-flops, since 16 is the lowest natural count greater than 9.

CHANGING THE COUNTER MODULUS-3

- A Mod-3 Counter



(a) Logic diagram

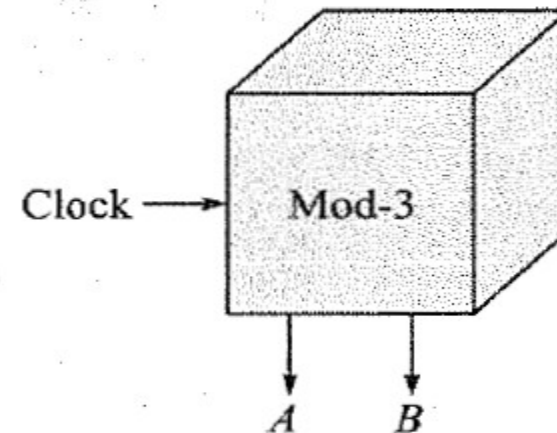


(b) Waveforms

B	A	Count
0	0	0
0	1	1
1	0	2

0	0	0

(c) Truth table



(d) Logic block

CHANGING THE COUNTER MODULUS-4

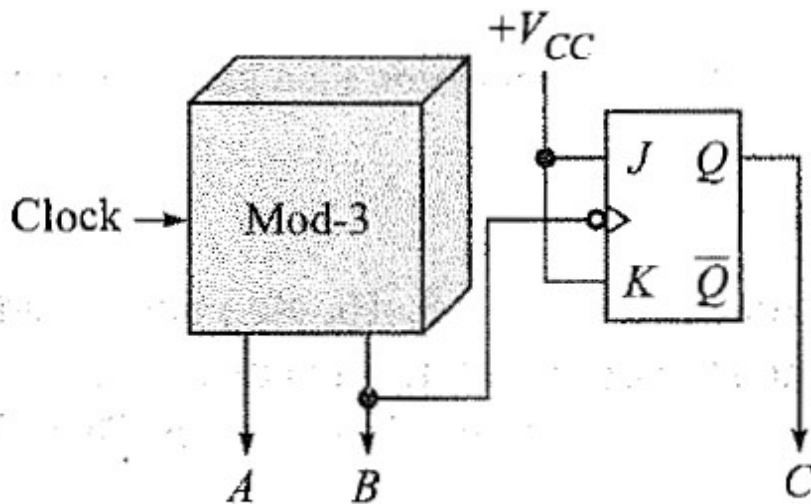
- **A Mod-3 Counter**
- Above circuit considered as a divide-by-3 block, since the output waveform at B (or at A) has a period equal to three times that of the clock-in other words, this counter
- divides the clock frequency by 3.

CHANGING THE COUNTER MODULUS-5

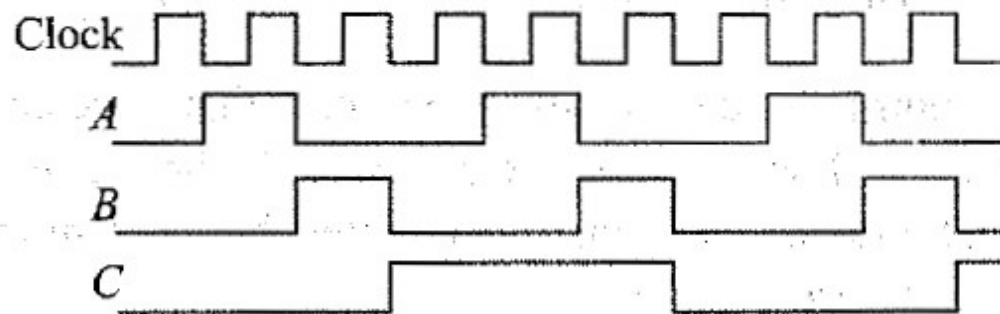
- If we consider a basic flip-flop to be a mod-2 counter, we see that a mod-4 counter (two flip-flops in series) is simply two mod-2 counters in series.
- Similarly, a mod-8 counter is simply a 2 x 2 x 2 connection, and so on.
- Thus a great number of higher-modulus counters can be formed by using the product of any number of lower modulus counters.

CHANGING THE COUNTER MODULUS-6

- **A Mod-6 Counter**
- Mod-6 counter is a ($3 \times 2 = 6$).
- The output of the single flip-flop is labeled C. Notice that it is a symmetrical waveform, and it also has a frequency of one-sixth that of the input clock.
- Also, this can no longer be considered a synchronous counter since flip flop C is triggered by flip-flop B; that is, the flip-flops do not all change status in synchronism with the clock.



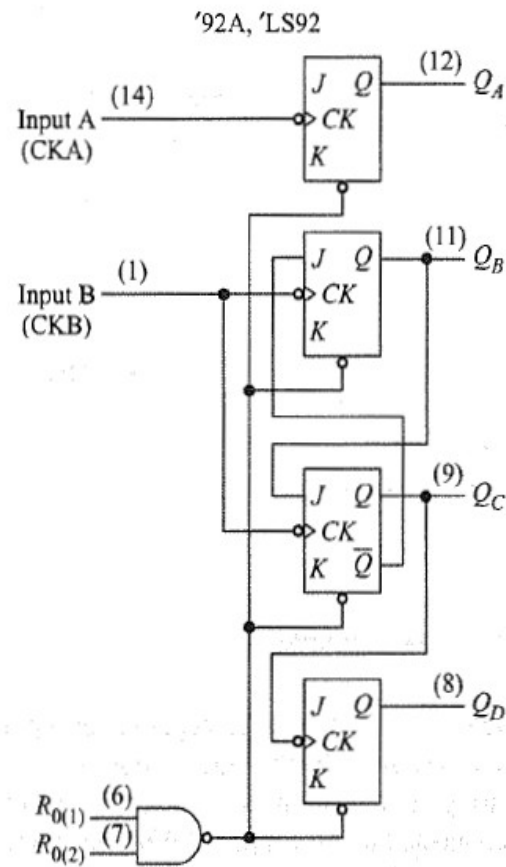
(a) 3×2 Mod-6 counter



(b) Waveforms

CHANGING THE COUNTER MODULUS-6

- **The 54/7492A**
- The 5417492A ('92A) is a divide-by-12. A careful examination of the logic diagram will reveal that flip-flops Q_B , Q_C , and Q_D are exactly the same as the 3 x 2 counter.
- Thus if the clock is applied to input B of the '92A and the outputs are taken at Q_B , Q_C , and Q_D , this is a mod-6 counter.
- On the other hand, if the clock is applied at input A and Q_A is connected to input B , we have a 2 x 3 x 2 mod-12 counter.

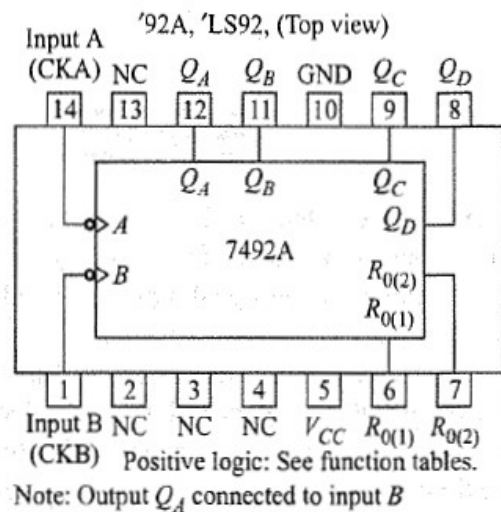


(a) Logic

'92A, 'LS92 Count sequence (See Note C)

Count	Output			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	H	L	L	L
7	H	L	L	H
8	H	L	H	L
9	H	L	H	H
10	H	H	L	L
11	H	H	L	H

(b) Truth table



(c) Pinout

Note: Output Q_A connected to input B