KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

COE 272
Digital Systems

Sequential Circuits

(Registers, Counters, FSM)

Registers

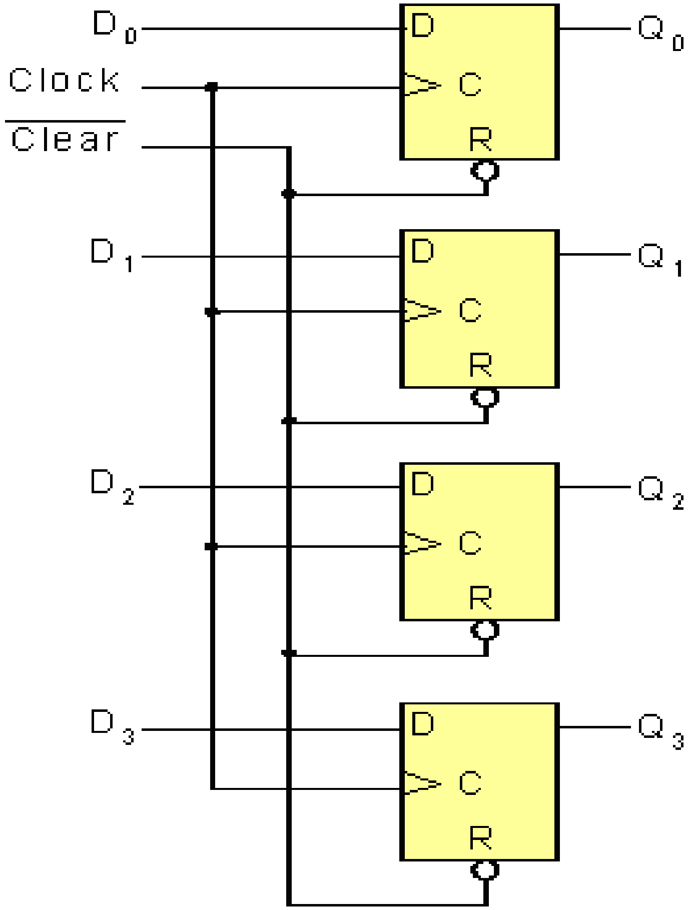
- A register is a circuit capable of storing data.

- An n-bit register consists of n FFs, together with some other logic that allows simple processing of the stored data.

- All FFs are triggered *synchronously* by the same clock signal; new data are latched into all FFs at the same time.

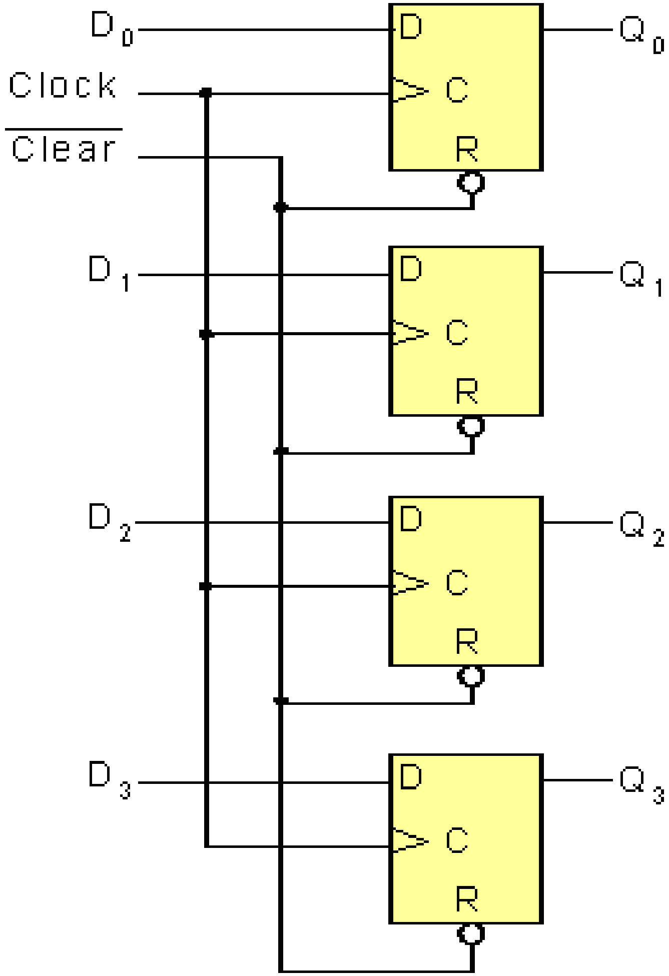
4-bit Register with D Flip Flops

- Figure shows a 4-bit register constructed with four D-type FFs.
  - Inputs D0 to D3
  - Clock
  - $\overline{\text{Clear}}$
  - Outputs Q_0 to Q_3
- The Clock input is common to all the four D FFs.
- It triggers all FFs on the rising edge of each clock pulse
- and the binary data available at the four D inputs are latched into the register.



4-bit Register with D Flip Flops

- The Clear input is an *active-low asynchronous* input which clears the register
  - *content to all 0's when asserted low, independent of the clock pulse.*
- During normal operation, Clear is maintained at logic 1.
- The transfer of new information into a register is referred to as Loading
- The term Parallel Loading is used if all the input bits are transferred into the register simultaneously, with the common clock pulse.



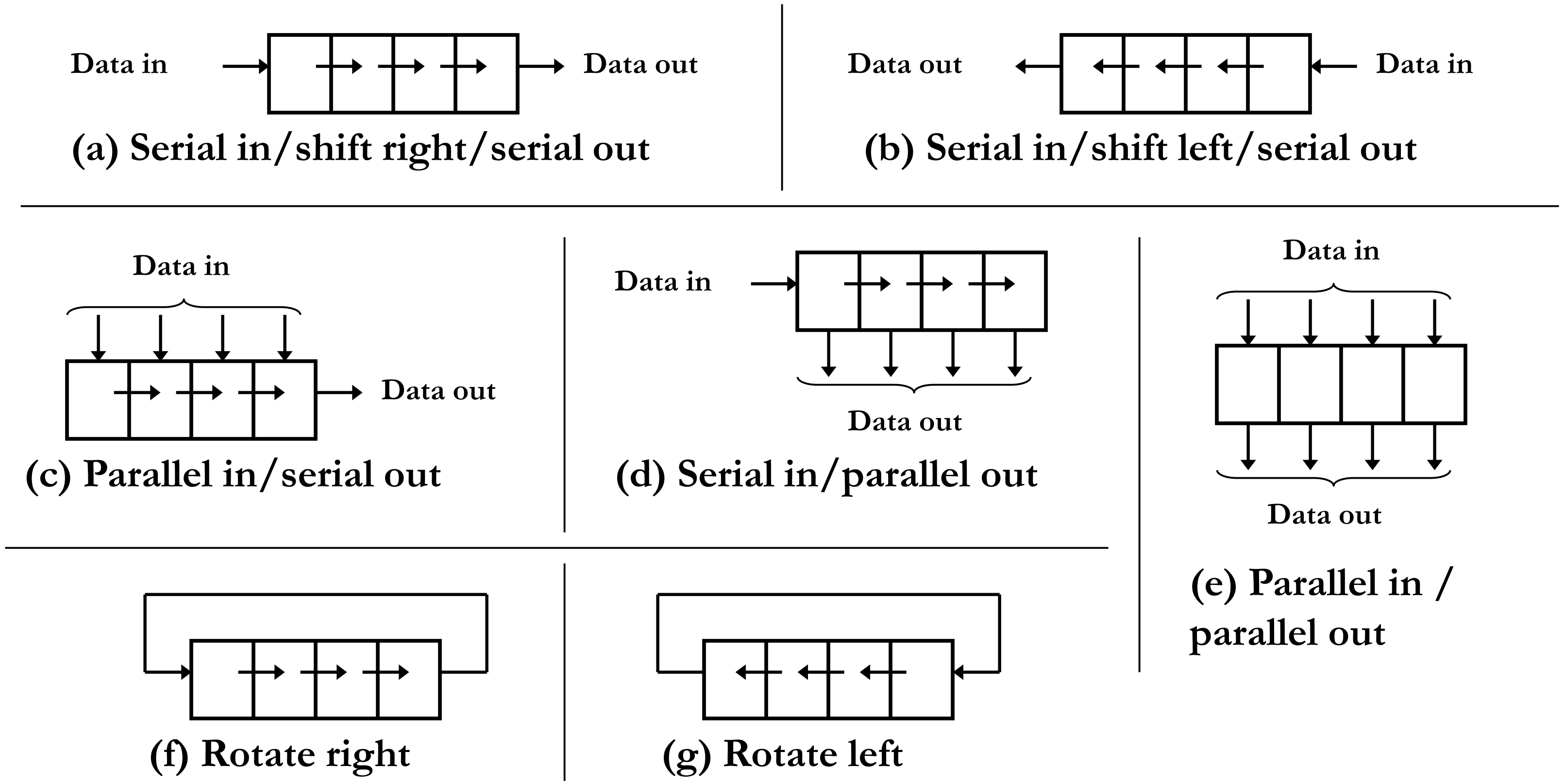
Shift Registers

- Another function of a register, besides storage, is to provide for *data movements*.

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

Shift Registers

- Basic data movement in shift registers (four bits are used for illustration).



(a) Serial in/shift right/serial out

(b) Serial in/shift left/serial out

(c) Parallel in/serial out

(d) Serial in/parallel out

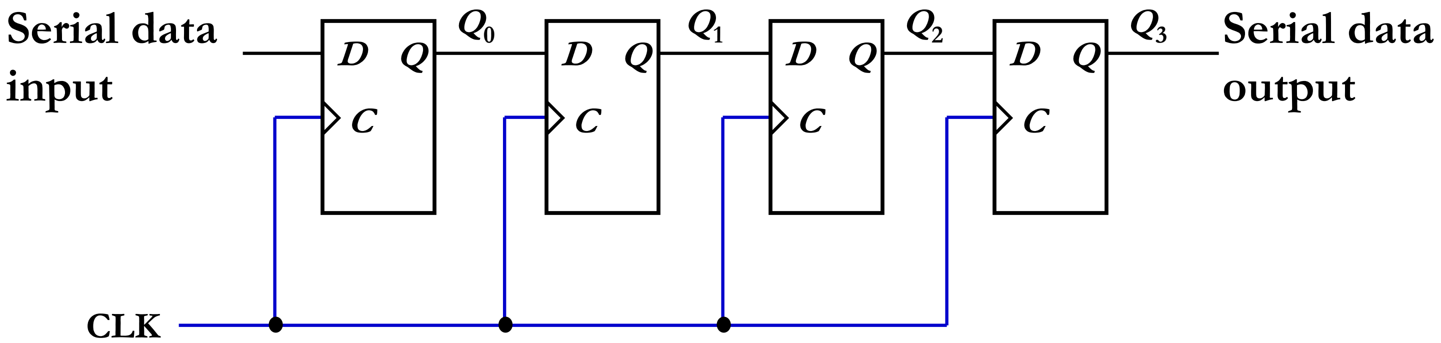
(e) Parallel in / parallel out

(f) Rotate right

(g) Rotate left

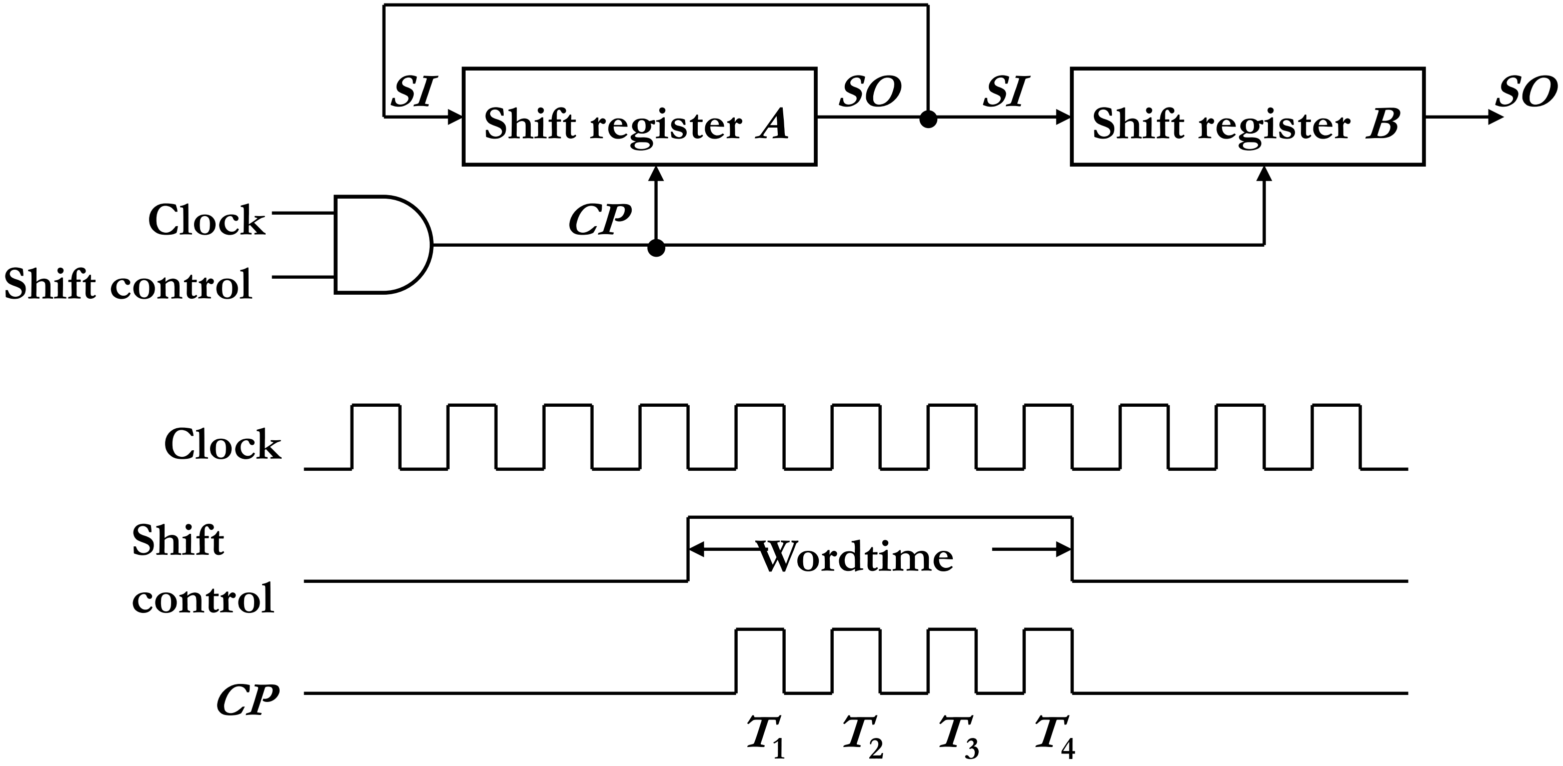
Serial In/Serial Out Shift Registers

- Accepts data serially – one bit at a time – and also produces output serially.



Serial In/Serial Out Shift Registers

- Application: Serial transfer of data from one register to another.



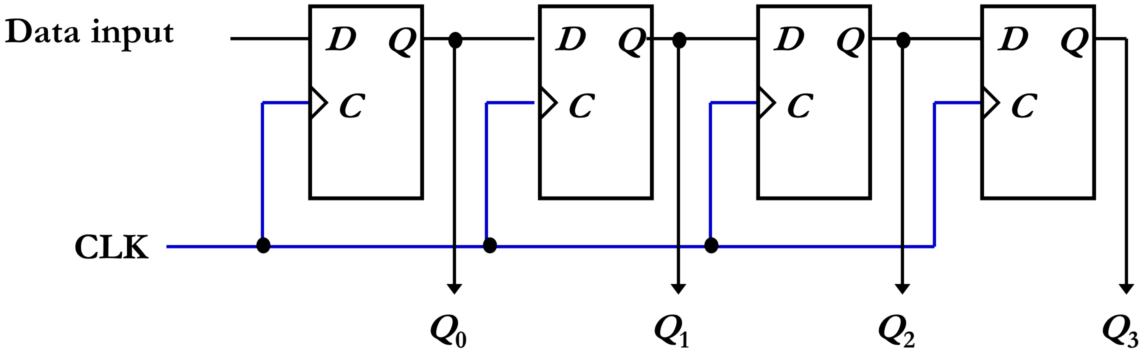
Serial In/Serial Out Shift Registers

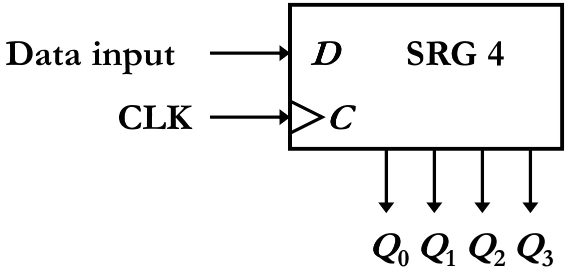
- Serial-transfer example.

Timing Pulse	Shift register *A*				Shift register *B*				Serial output of *B*
Initial value	1	0	1	1	0	0	1	0	0
After T_1	1	1	0	1	1	0	0	1	1
After T_2	1	1	1	0	1	1	0	0	0
After T_3	0	1	1	1	0	1	1	0	0
After T_4	1	0	1	1	1	0	1	1	1

Serial In/Parallel Out Shift Registers

- Accepts data serially.
- Outputs of all stages are available simultaneously.

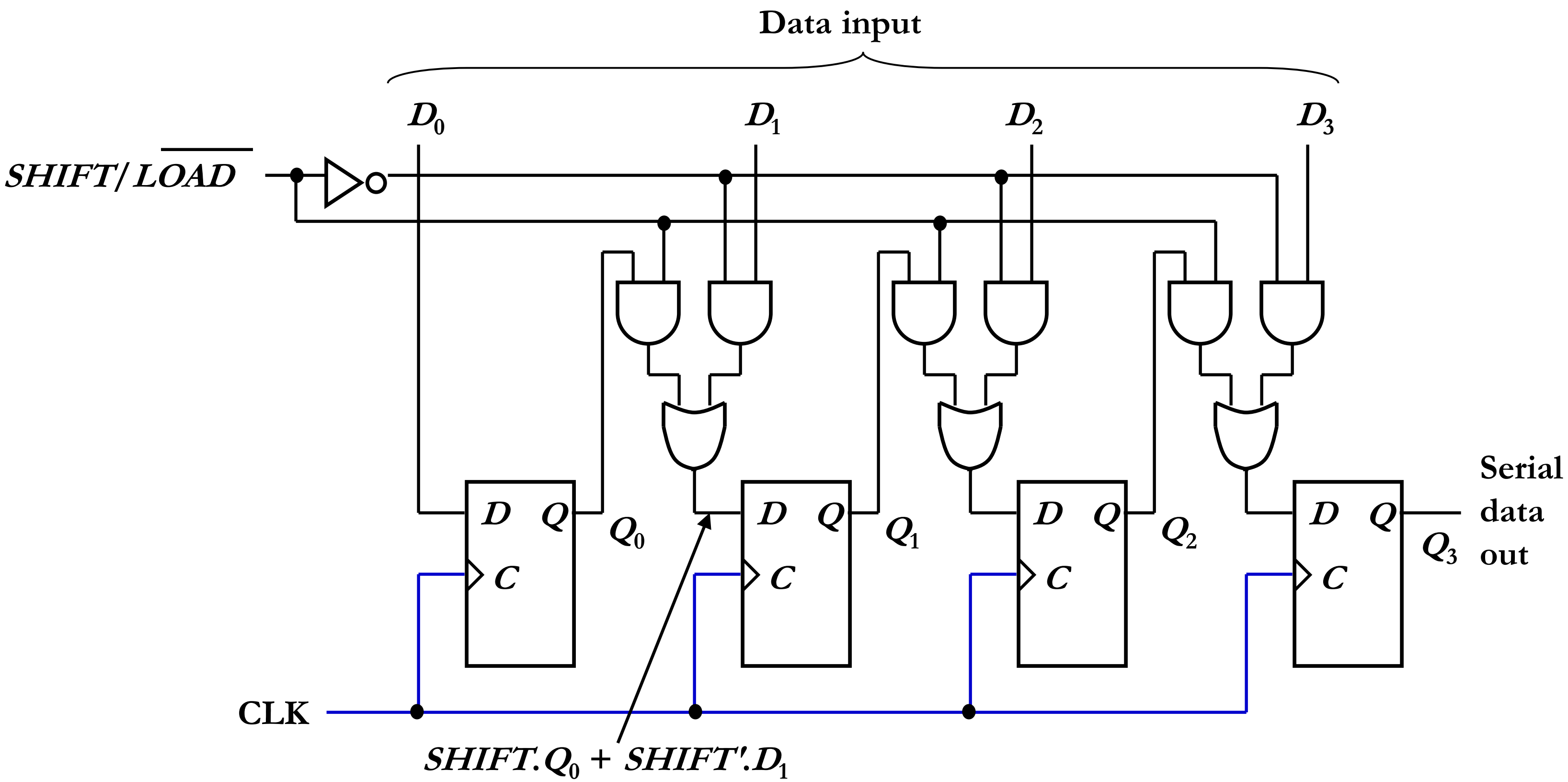




Logic symbol

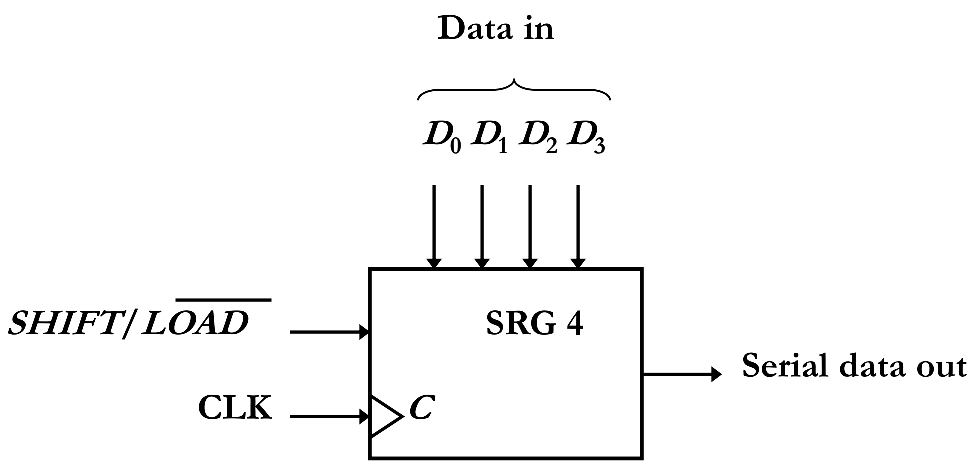
Parallel In/Serial Out Shift Registers

- Bits are entered simultaneously, but output is serial.



Parallel In/Serial Out Shift Registers

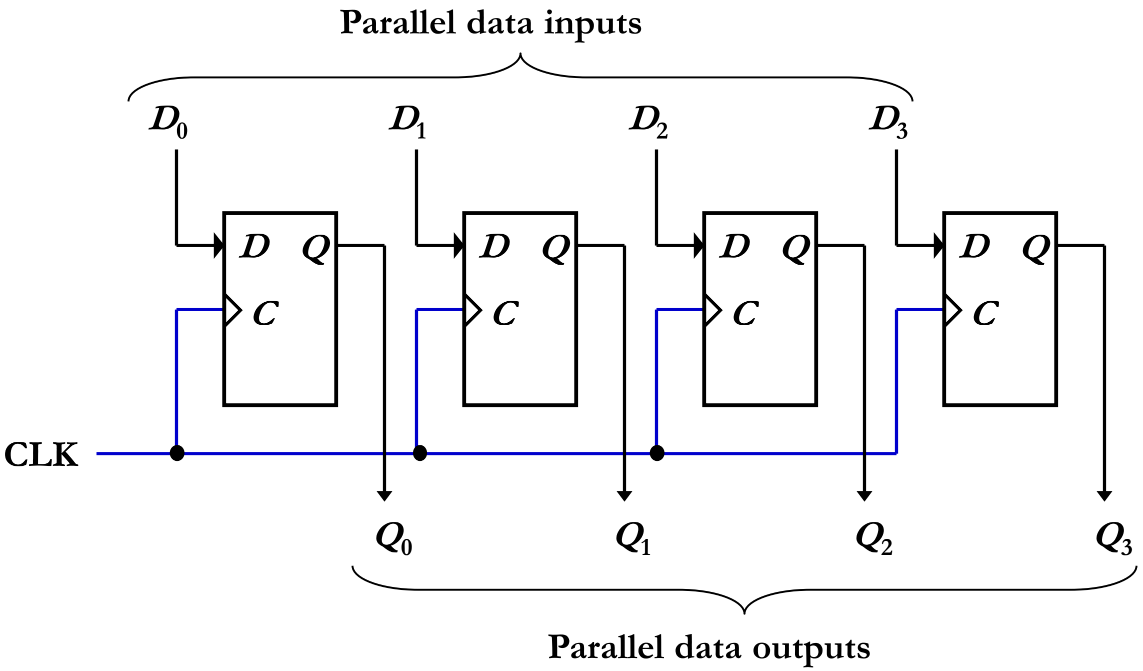
- Bits are entered simultaneously, but output is serial.



Logic symbol

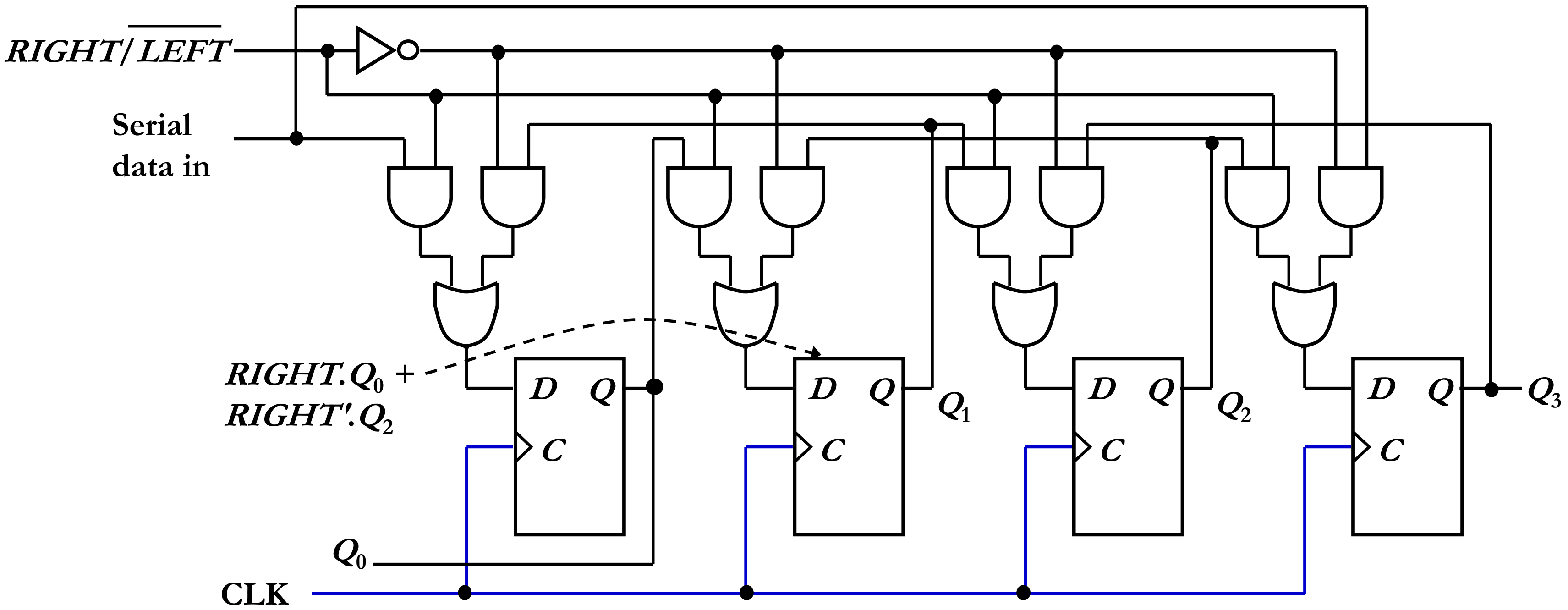
Parallel In/Parallel Out Shift Registers

- Simultaneous input and output of all data bits.



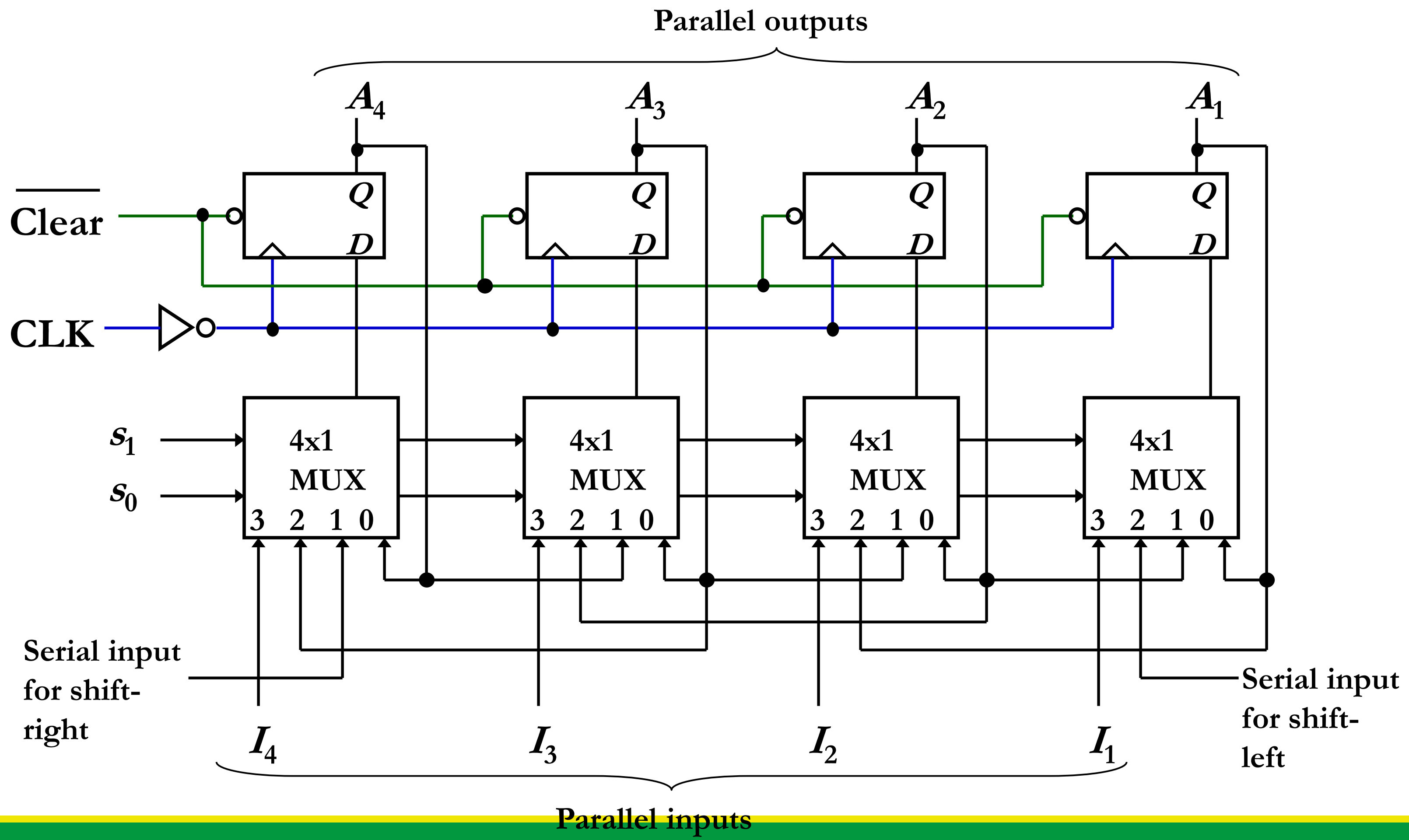
Bidirectional Shift Registers

- Data can be shifted either left or right, using a control line $RIGHT/\overline{LEFT}$ (or simply $RIGHT$) to indicate the direction.



Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.



Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.

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

Shift Register Counters

- Shift register counter: a shift register with the serial output connected back to the serial input.

- They are classified as counters because they give a specified sequence of states.

- Two common types: the *Johnson counter* and the *Ring counter*.

Ring Counters

- One flip-flop (stage) for each state in the sequence.

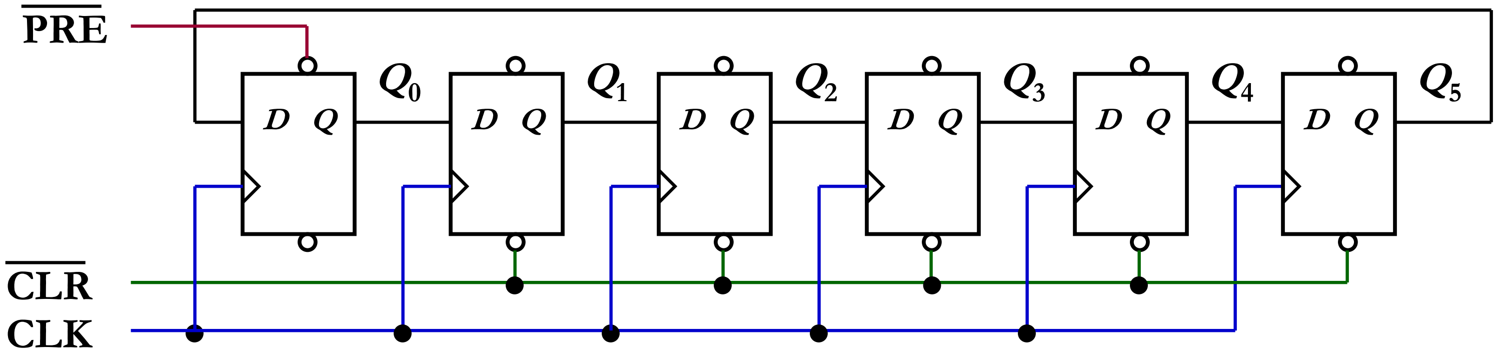
- The output of the last stage is connected to the D input of the first stage.

- An *n*-bit ring counter cycles through *n* states.

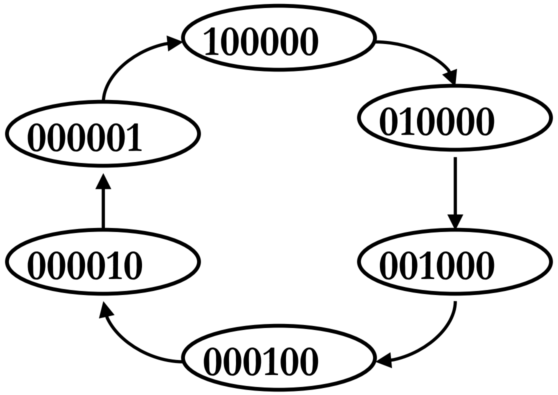
- No decoding gates are required, as there is an output that corresponds to every state the counter is in.

Ring Counters

- Example: A 6-bit ring counter.



Clock	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

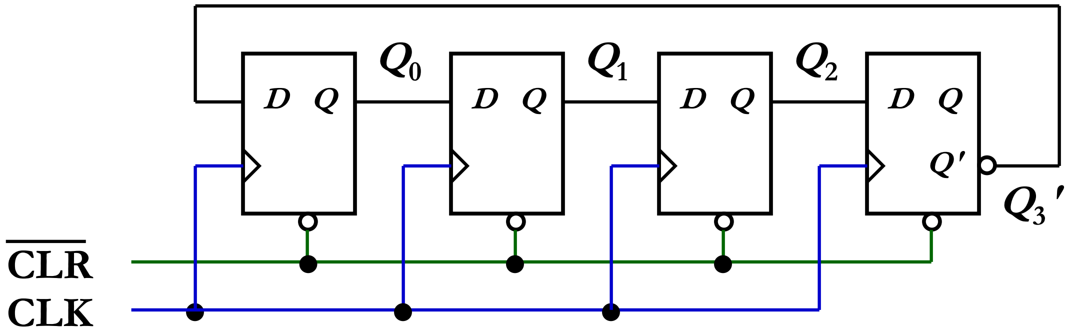


Johnson Counters

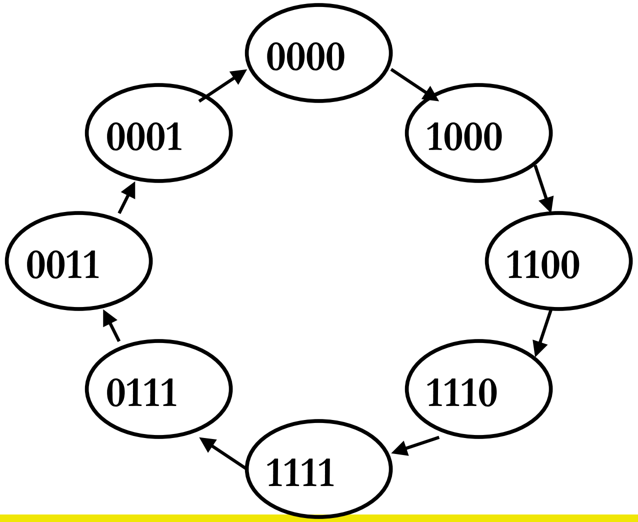
- The complement of the output of the last stage is connected back to the D input of the first stage.
- Also called the *twisted-ring counter*.
- Require fewer flip-flops than ring counters but more flip-flops than binary counters.
- An *n*-bit Johnson counter cycles through 2*n* states.
- Require more decoding circuitry than ring counter but less than binary counters.

Johnson Counters

- Example: A 4-bit (MOD-8) Johnson counter.



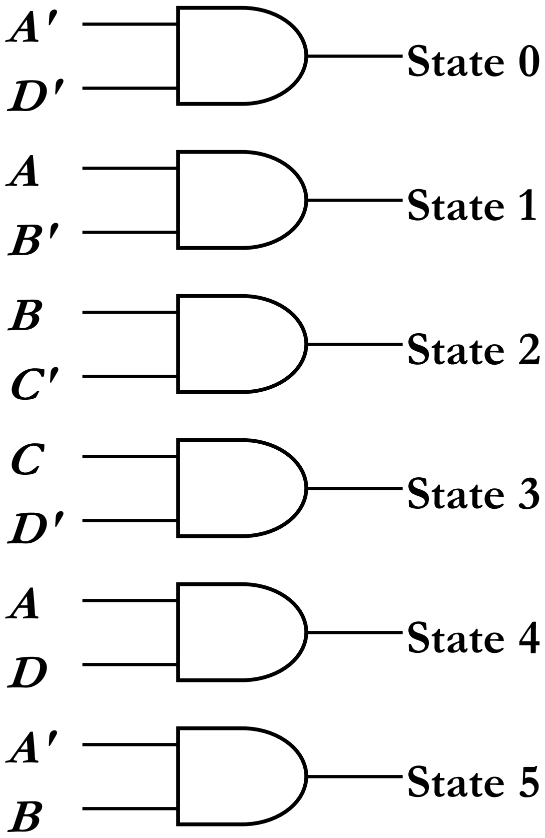
Clock	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

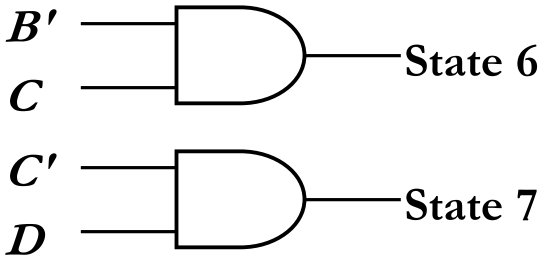


Johnson Counters

- Decoding logic for a 4-bit Johnson counter.

Clock	*A*	*B*	*C*	*D*	Decoding
0	0	0	0	0	*A'.D'*
1	1	0	0	0	*A.B'*
2	1	1	0	0	*B.C'*
3	1	1	1	0	*C.D'*
4	1	1	1	1	*A.D*
5	0	1	1	1	*A'.B*
6	0	0	1	1	*B'.C*
7	0	0	0	1	*C'.D*





Counters

- Counters are circuits that cycle through a specified number of states.

- Two types of counters:
  - synchronous (parallel) counters
  - asynchronous (ripple) counters

- Ripple counters allow some flip-flop outputs to be used as a source of clock for other flip-flops.

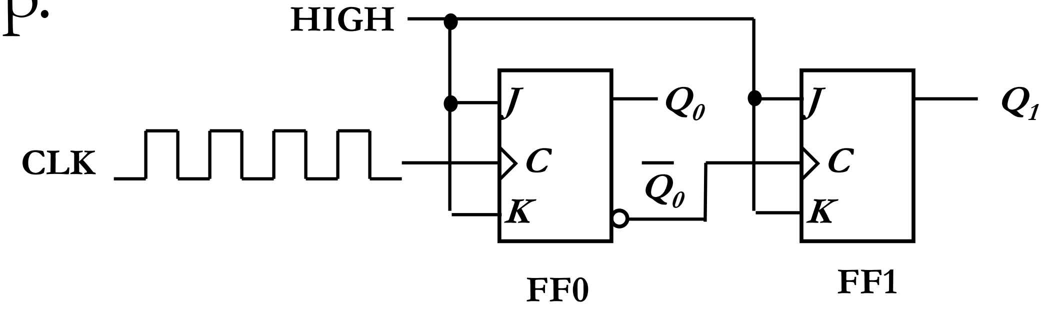
- Synchronous counters apply the same clock to all flip-flops.

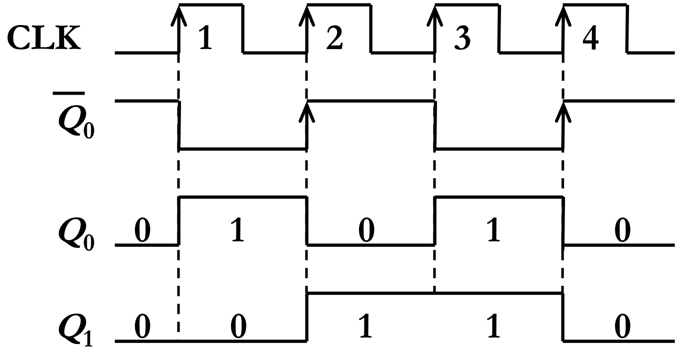
Asynchronous (Ripple) Counters

- Asynchronous counters: the flip-flops do not change states at exactly the same time as they do not have a common clock pulse.
- Also known as ripple counters, as the input clock pulse “ripples” through the counter – cumulative delay is a drawback.
- n flip-flops → a MOD (modulus) 2^n counter. (Note: A MOD-x counter cycles through x states.)
- Output of the last flip-flop (MSB) divides the input clock frequency by the MOD number of the counter, hence a counter is also a *frequency divider*.

Asynchronous (Ripple) Counters

- Example: 2-bit ripple binary counter.
- Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.



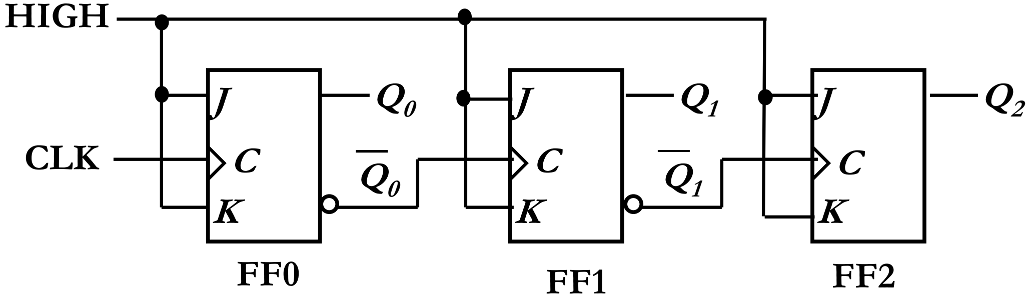


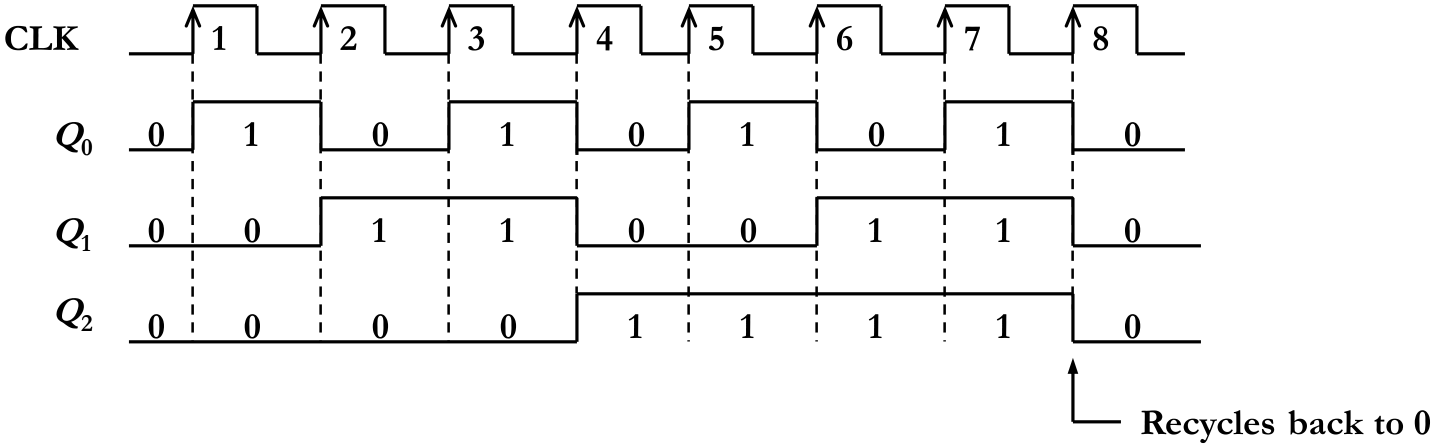
Timing diagram

00 → 01 → 10 → 11 → 00 ...

Asynchronous (Ripple) Counters

- Example: 3-bit ripple binary counter.

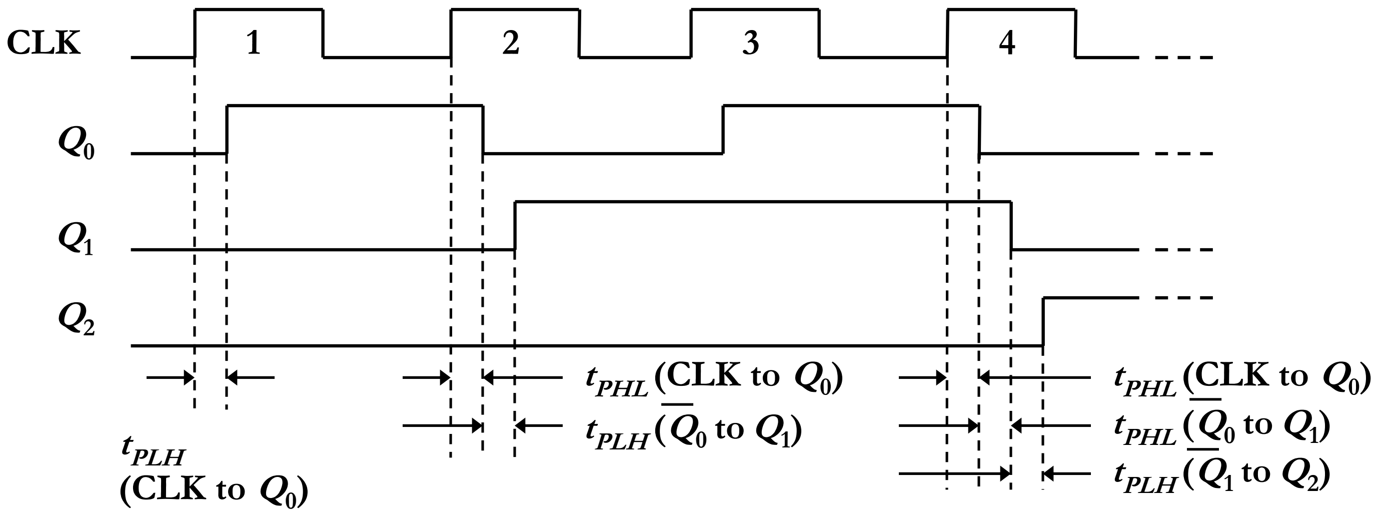




Asynchronous (Ripple) Counters

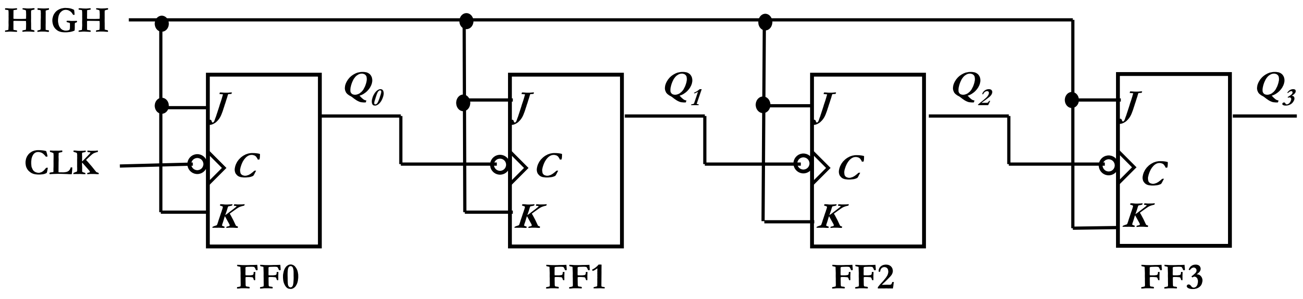
- Propagation delays in an asynchronous (ripple-clocked) binary counter.

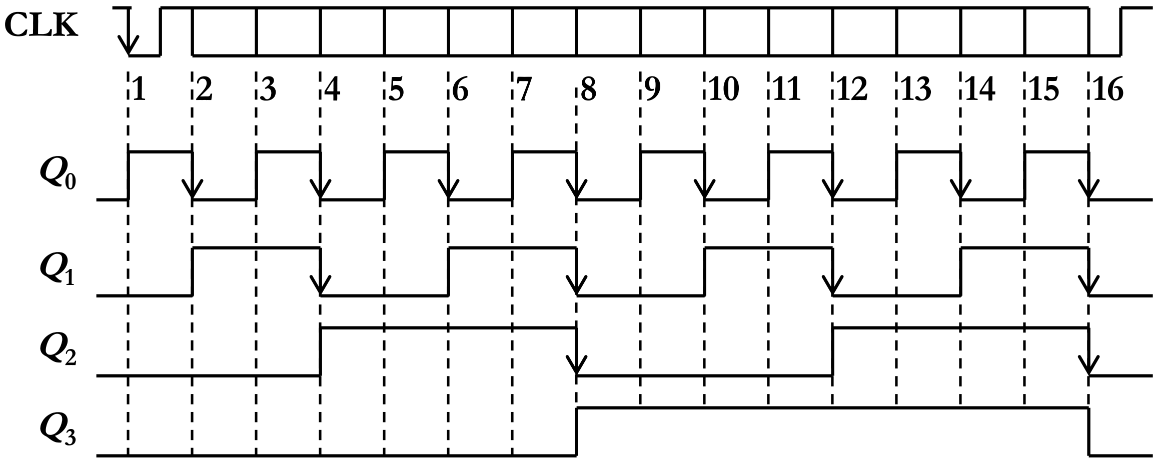
- If the accumulated delay is greater than the clock pulse, some counter states may be misrepresented!


CLK
1
2
3
4
Q_0
Q_1
Q_2
t_{PHL} (CLK to Q_0)
t_{PLH} ($\overline{Q}_0$ to Q_1)
t_{PHL} (CLK to Q_0)
t_{PHL} ($\overline{Q}_0$ to Q_1)
t_{PLH} ($\overline{Q}_1$ to Q_2)
t_{PLH}
(CLK to Q_0)

Asynchronous (Ripple) Counters

- Example: 4-bit ripple binary counter (negative-edge triggered).

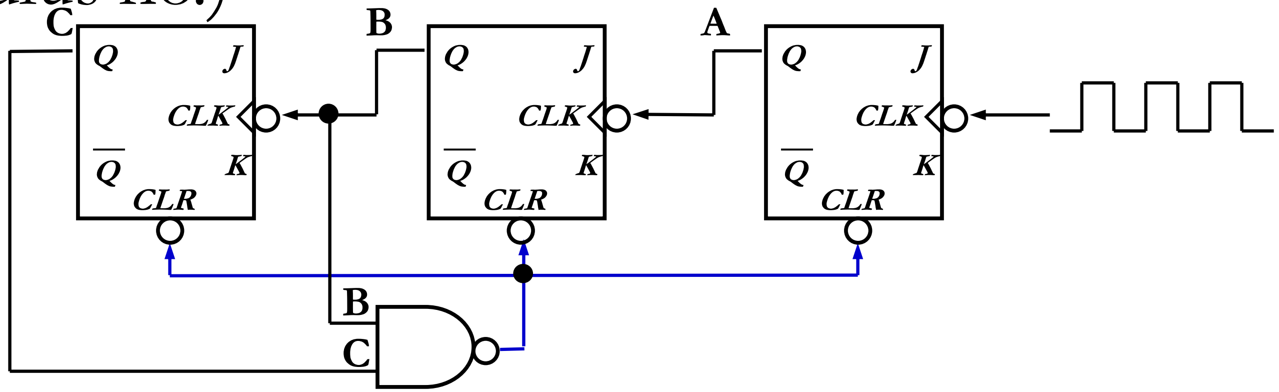




Asyn. Counters with MOD no. < 2^n

- States may be skipped resulting in a truncated sequence.
- Technique: force counter to *recycle before going through all of the states* in the binary sequence.
- Example: Given the following circuit, determine the counting sequence (and hence the modulus no.)

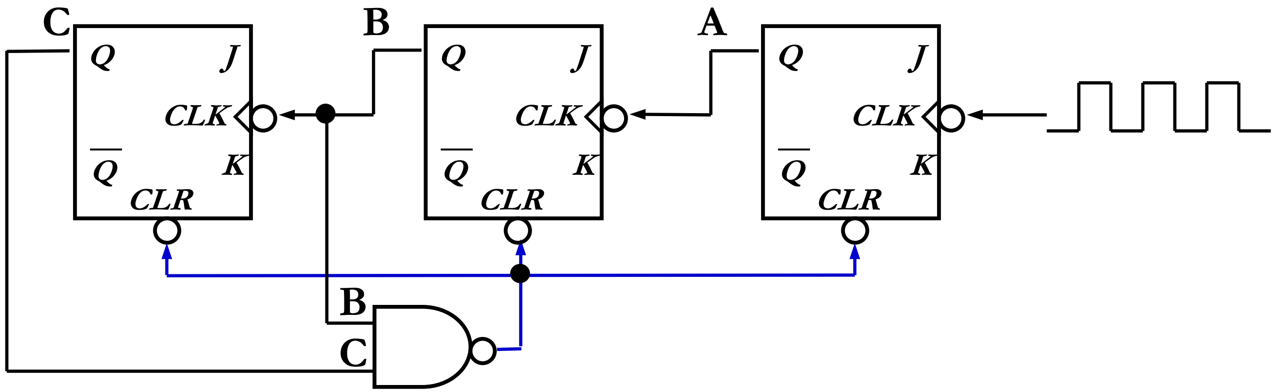
All *J*, *K* inputs are 1 (HIGH).

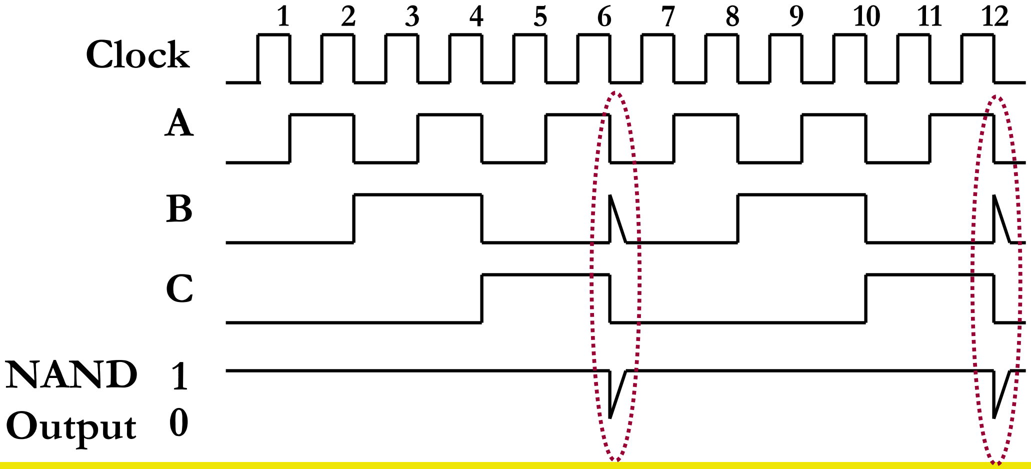


Asynchronous Counters with MOD no. < 2^n

- Example (cont'd):

All *J*, *K* inputs are 1 (HIGH).

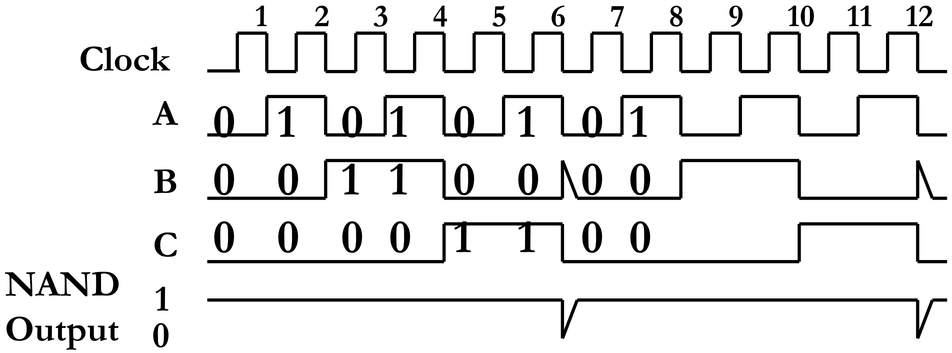


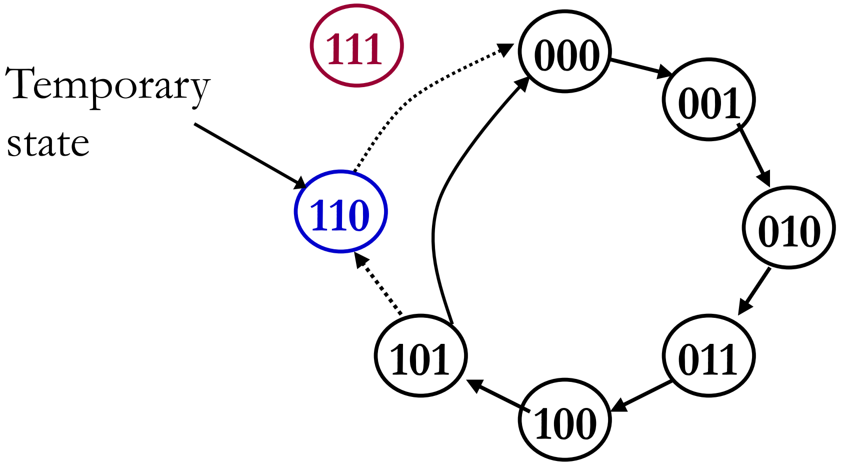


MOD-6 counter produced by clearing (a MOD-8 binary counter) when count of six (110) occurs.

Asynchromous Counters with MOD no. < 2^n

- Example (cont'd): Counting sequence of circuit (in CBA order).



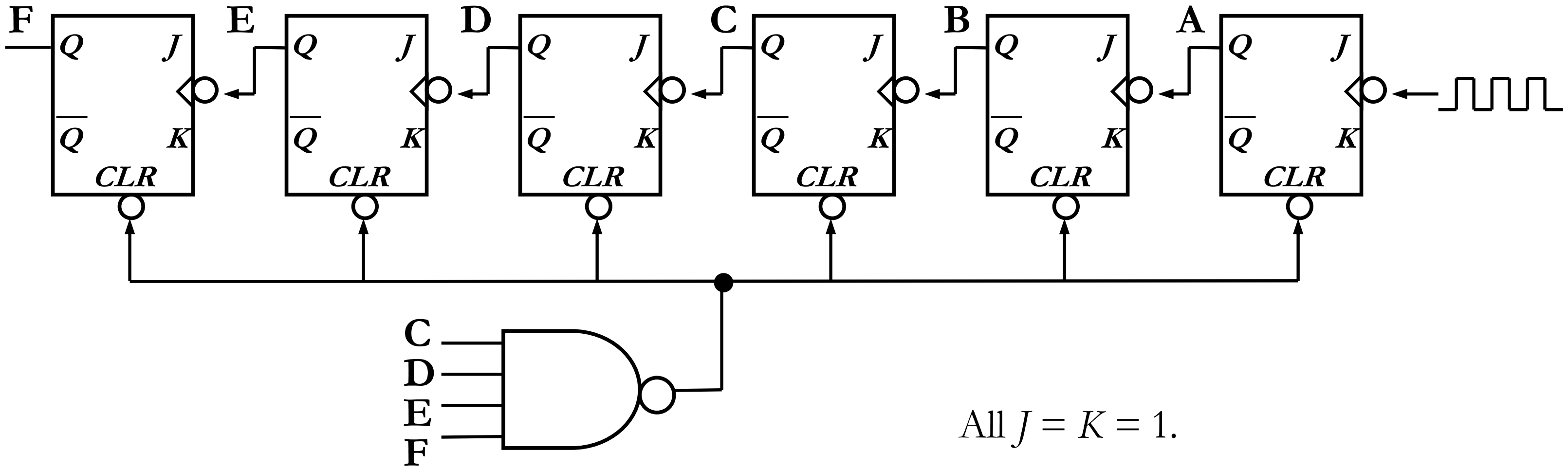


Counter is a MOD-6 counter.

Asynchronous Counters with MOD no. < 2^n

- *Exercise:* How to construct an asynchronous MOD-5 counter? MOD-7 counter? MOD-12 counter?

- *Question:* The following is a MOD-? counter?

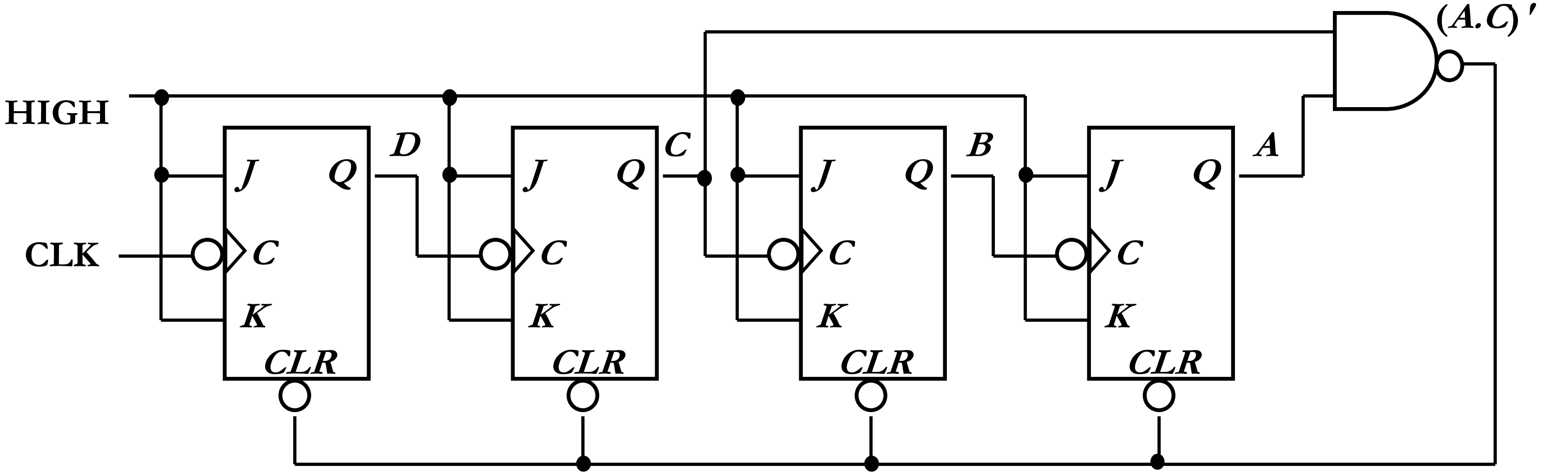


All $J = K = 1$.

Asynchronous Counters with MOD no. < 2^n

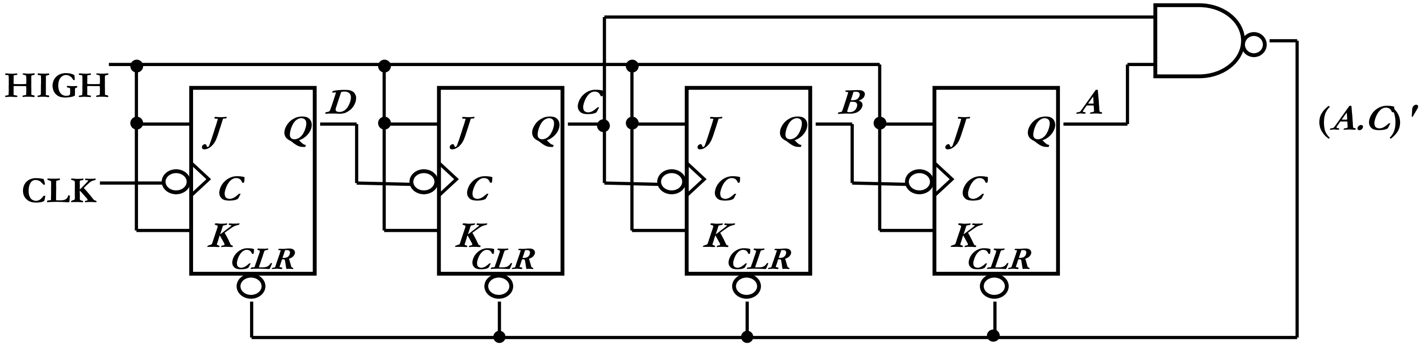
- Decade counters (or BCD counters) are counters with 10 states (modulus-10) in their sequence. They are commonly used in daily life (e.g.: utility meters, odometers, etc.).

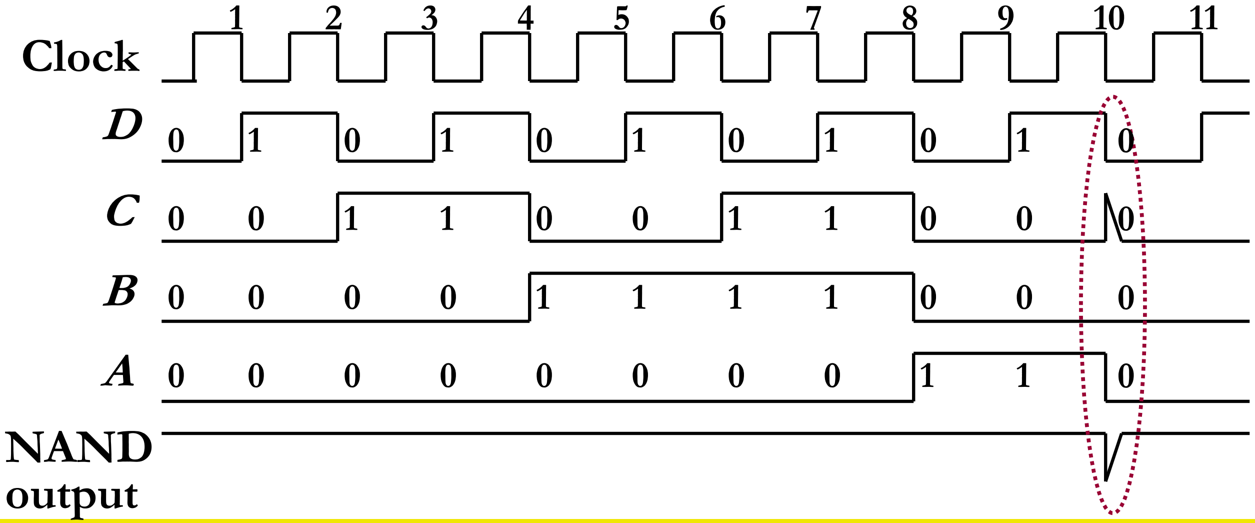
- Design an asynchronous decade counter.



Asynchronous Counters with MOD no. < 2^n

- Asynchronous decade/BCD counter (cont'd).

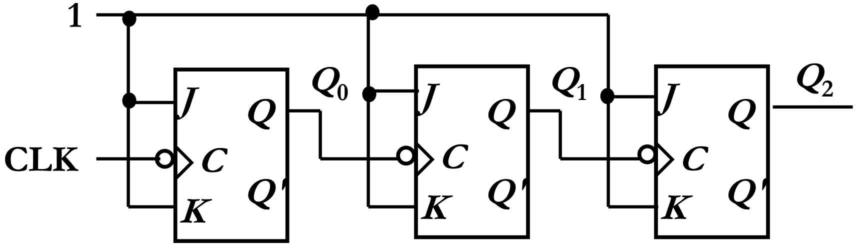




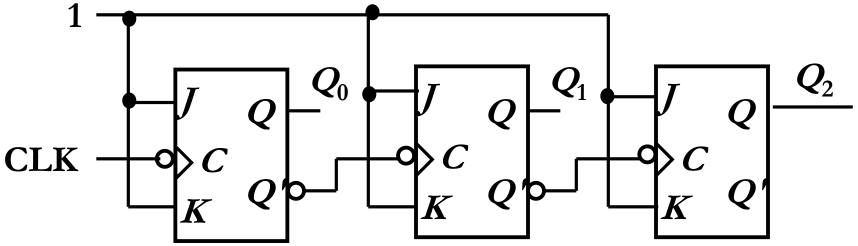
Asynchronous Down Counters

- So far we are dealing with *up counters*. *Down counters*, on the other hand, count downward from a maximum value to zero, and repeat.

- Example: A 3-bit binary (MOD-2^3) down counter.



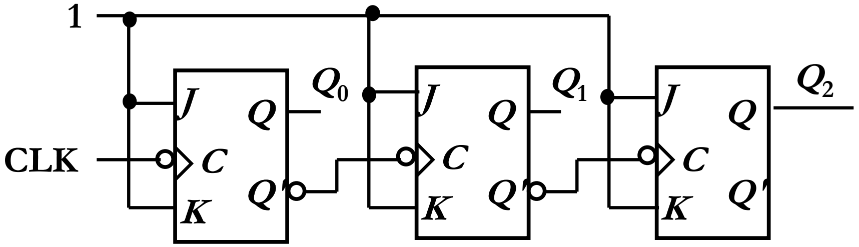
3-bit binary up counter

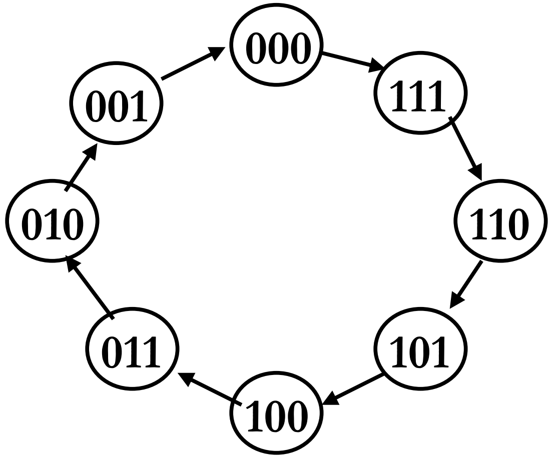


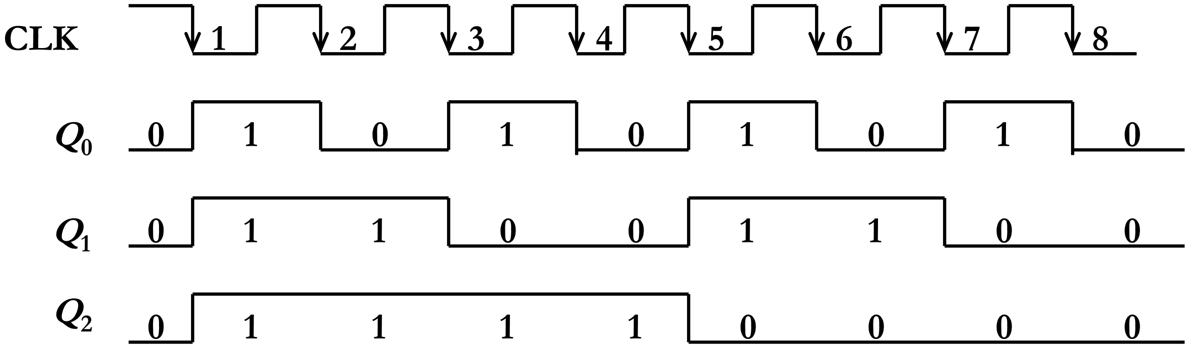
3-bit binary down counter

Asynchronous Down Counters

- Example: A 3-bit binary (MOD-8) down counter.

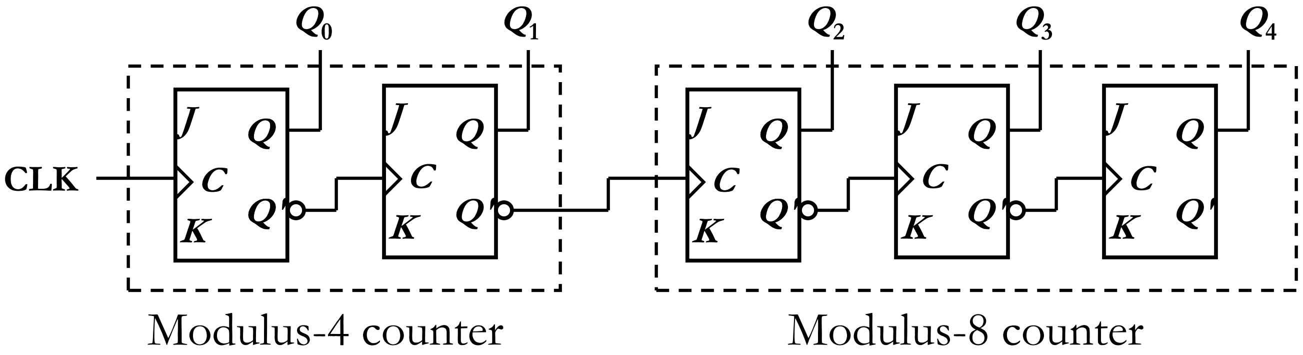






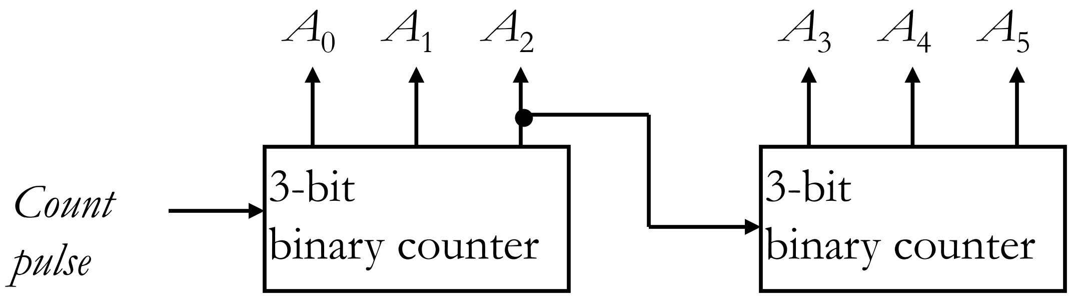
Cascading Asynchronous Counters

- Larger asynchronous (ripple) counter can be constructed by cascading smaller ripple counters.
- Connect last-stage output of one counter to the clock input of next counter so as to achieve higher-modulus operation.
- Example: A modulus-32 ripple counter constructed from a modulus-4 counter and a modulus-8 counter.



Cascading Asynchronous Counters

- Example: A 6-bit binary counter (counts from 0 to 63) constructed from two 3-bit counters.

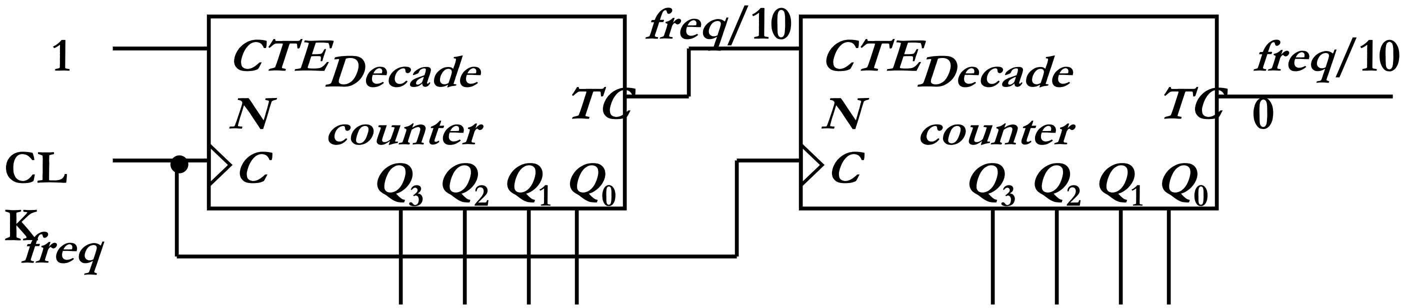


A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	:	:	:
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
:	:	:	:	:	:

Cascading Asynchronous Counters

- If counter is a not a binary counter, requires additional output.

- Example: A modulus-100 counter using two decade counters.



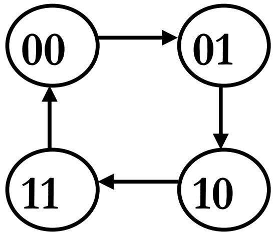
$TC = 1$ when counter recycles to 0000

Synchronous (Parallel) Counters

- Synchronous (parallel) counters: the flip-flops are clocked at the same time by a common clock pulse.

- We can design these counters using the sequential logic design process.

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



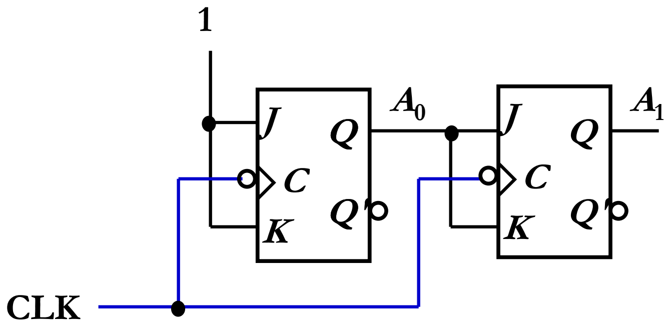
Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

Synchronous (Parallel) Counters

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

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

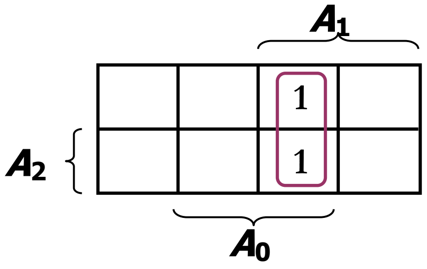
$$TA_1 = A_0$$
$$TA_0 = 1$$



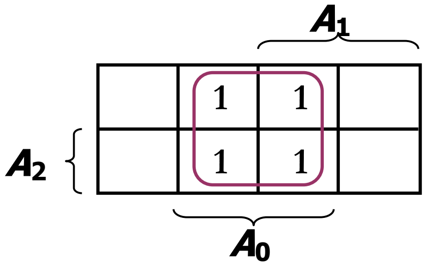
Synchronous (Parallel) Counters

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

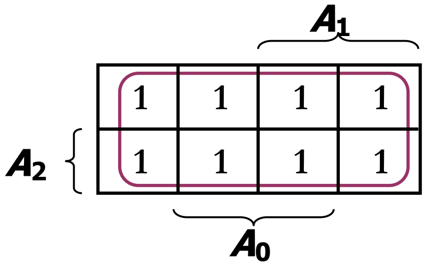
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$



$TA_1 = A_0$

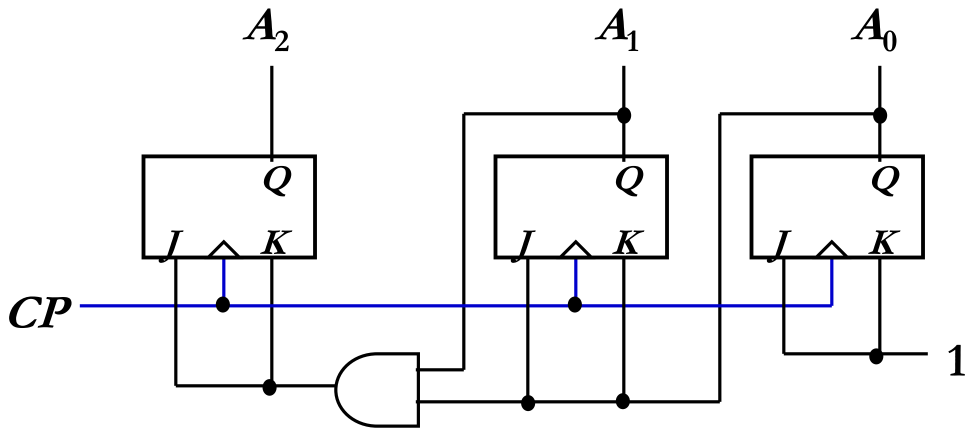


$TA_0 = 1$

Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (cont'd).

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

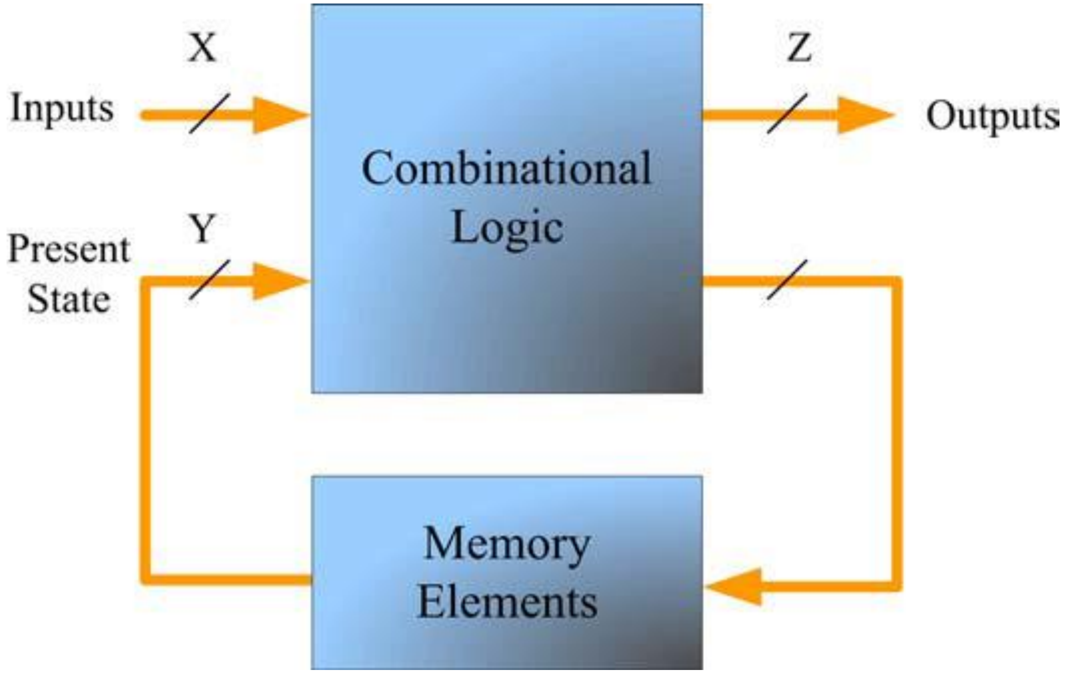


Finite State Machines

- There are two basic ways to design clocked sequential circuits. These are using:
    - Mealy Machine
    - Moore Machine

Mealy Type Machine

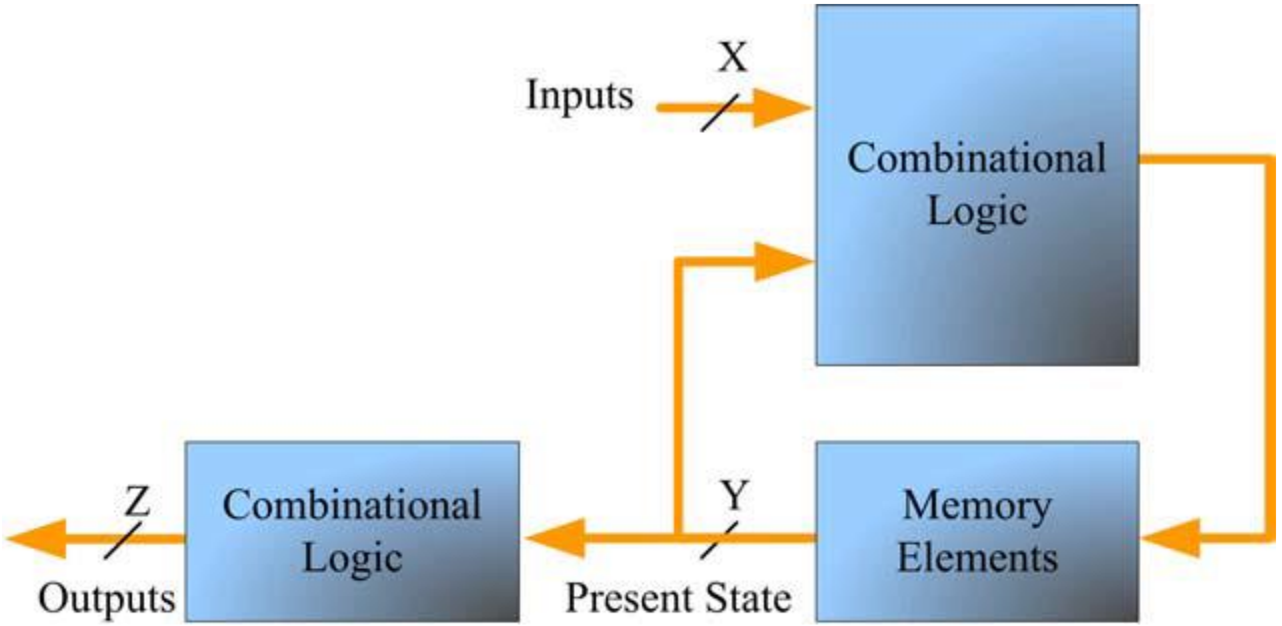
- The outputs are a function of the present state and the value of the inputs
- The output may change asynchronously in response to any change in the inputs



Moore Type Machine

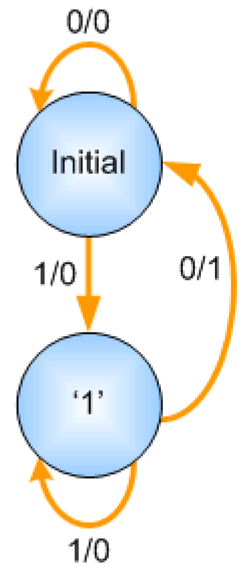
- In a Moore machine the outputs depend only on the present state.
- A combinational logic block maps the inputs and the current state into the necessary flip-flop inputs to store the appropriate next state just like Mealy machine.
- However, the outputs are computed by a combinational logic block whose inputs are only the flip-flops state outputs.
- The outputs change synchronously with the state transition triggered by the active clock edge.

Moore Type Machine

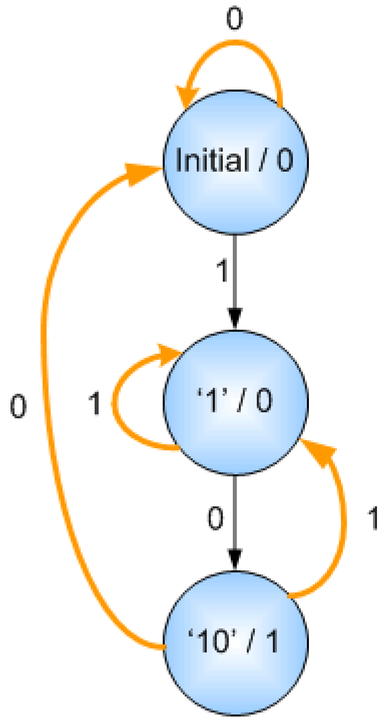


Comparison of the two machine types

- Consider a finite state machine that checks for a pattern of '10' and asserts logic high when it is detected.
- The state diagram representations for the Mealy and Moore machines are shown in the Figure.
- The state diagram of the Mealy machine lists the inputs with their associated outputs on state transitions arcs.
- The value stated on the arrows for Mealy machine is of the form Zi/Xi where Zi represents input value and Xi represents output value.

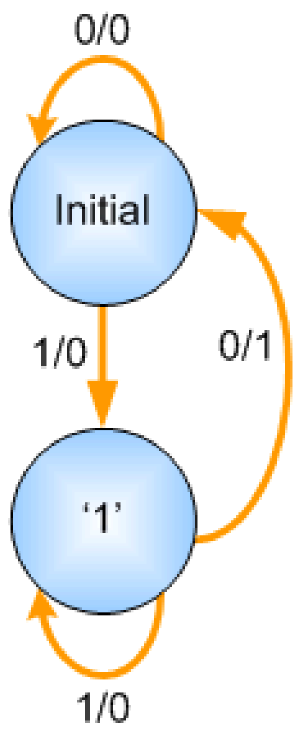


Mealy Machine

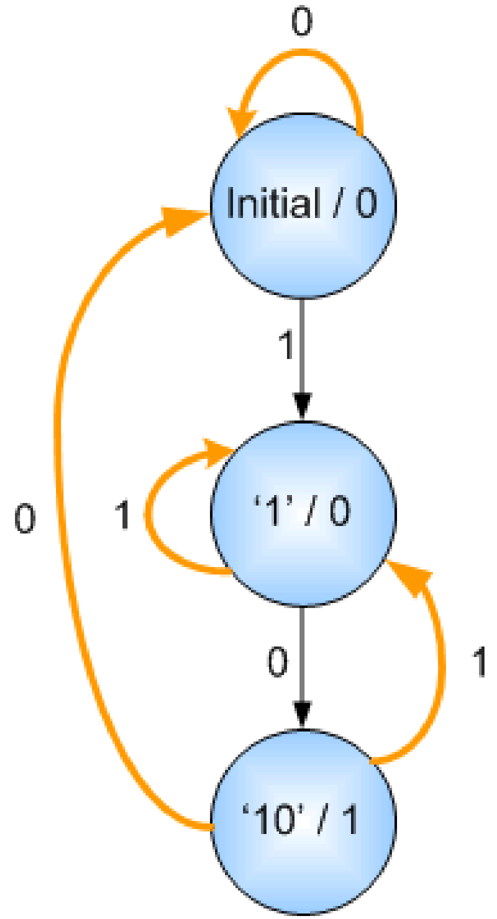


Moore Machine

- A Moore machine produces a unique output for every state irrespective of inputs.
- Accordingly the state diagram of the Moore machine, it associates the output with the state in the form state-notation/output-value.
- The state transition arrows of Moore machine are labeled with the input value that triggers such transition.
- Since a Mealy machine associates outputs with transitions, an output sequence can be generated in fewer states using Mealy machine as compared to Moore machine.

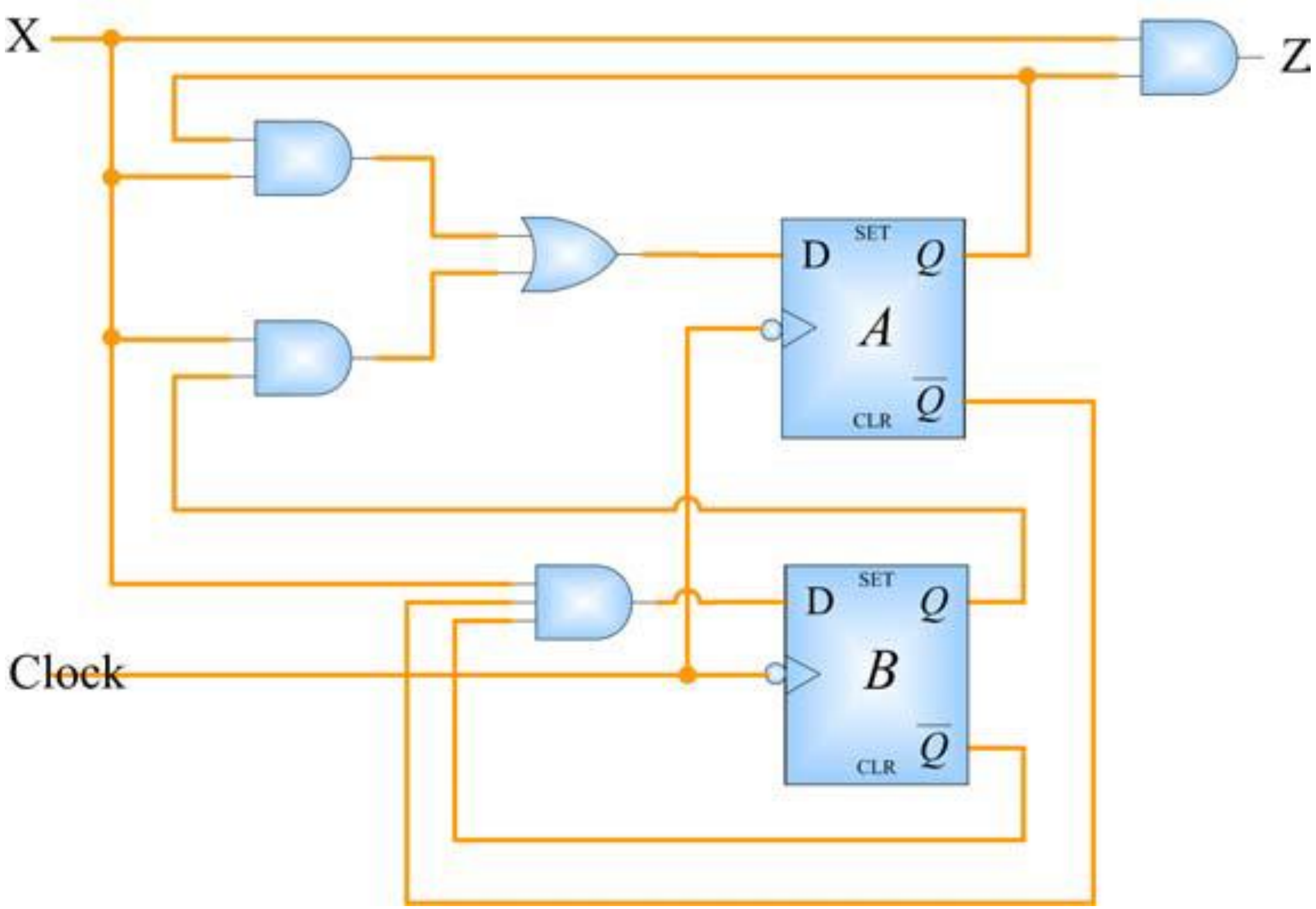


Mealy Machine

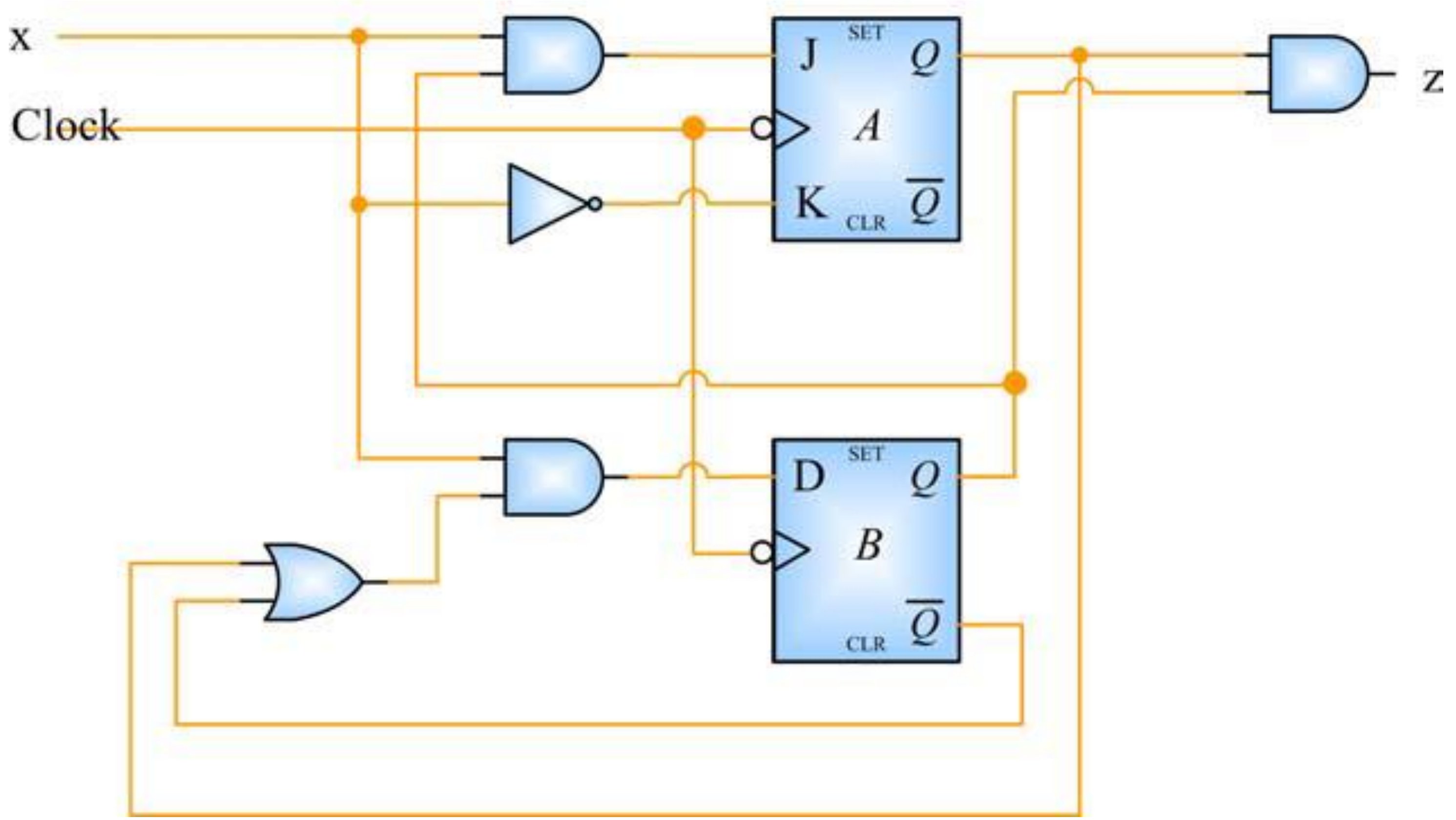


Moore Machine

Mealy State Machine Implementation



Moore State Machine Implementation



Design Procedure for Sequential Circuits

- The design procedure consists of a series of steps that start with a state diagram or description of a circuit and ends with a circuit diagram and or the flip flop input equations and the circuit output equations.

- The design procedure are outlined in the next slide

Design Procedure for Sequential Circuits

- From the information given, create the headings of the state table.
  - Remember that a state table consists of present state, input, next state and output.
  - From the question you must decide the following
    - The number and type of flip flop
    - The number of inputs. There may be no inputs
    - The number of outputs. There may be no outputs
  - Base on these information label each column in the state table.

Design Procedure for Sequential Circuits

- Fill each entry in the state table in binary order based upon the combinational columns of the present state and input.
  - Filling in the entire chart is a must, else it would be impossible to design the circuit
  - To the right hand of the state table, create the column headings for the flip-flop inputs

Design Procedure for Sequential Circuits

- Fill out the flip-flop inputs. (using the present and next state values from corresponding columns, i.e. the A column of each and the B column of each, etc.
- For each column of Flip-flop inputs, transfer the minterms (only the 1's and X's (don't cares) are necessary) to a K-map and reduce
- For each column of the output, transfer the minterms to a K-map and reduce.

NB: the last two steps produce the Equations which sufficiently describe the sequential circuit

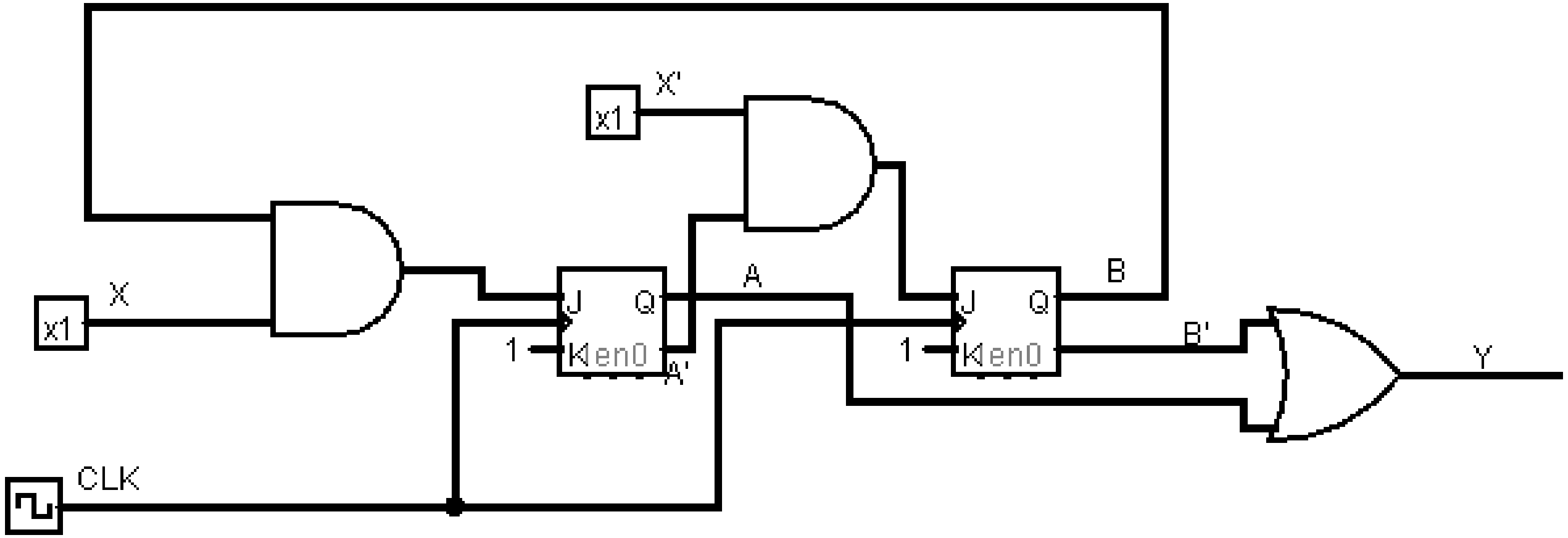
Design Procedure for Sequential Circuits

- If requested draw the circuit

- Note: it is always wise to test out your equations and/or circuit to see if specifications of problem are met

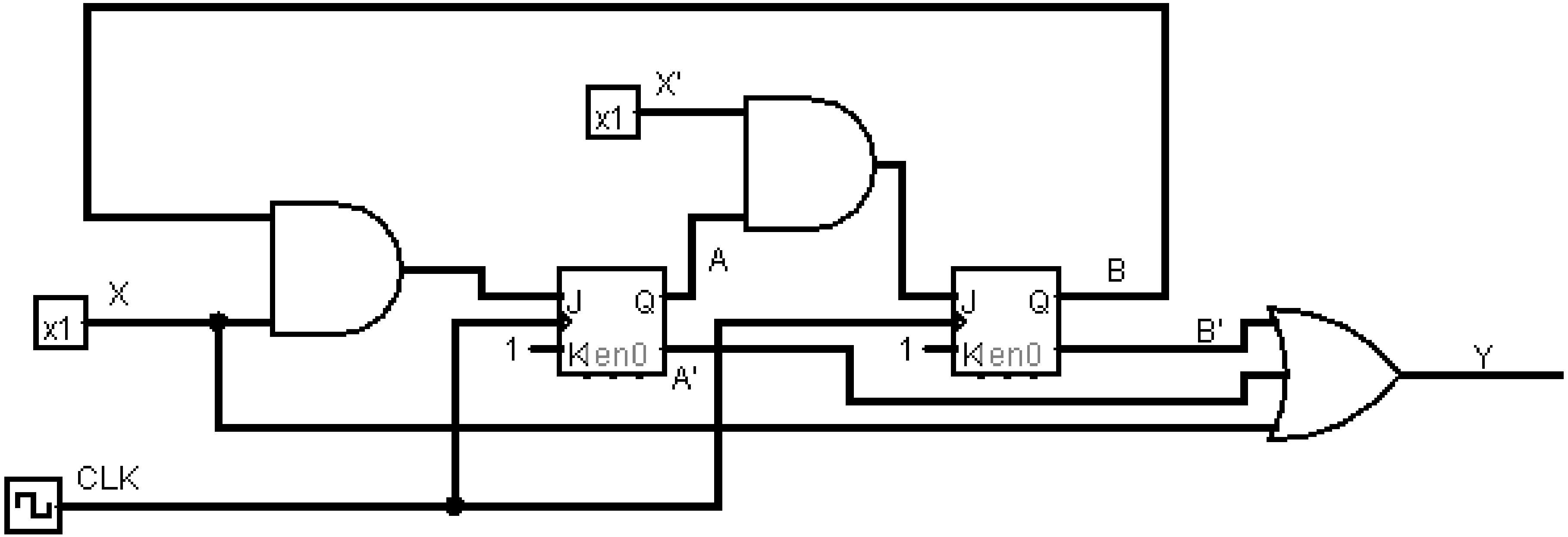
Exercise

Identify the type of circuit given. Write the state table and draw the state diagram for the same



Exercise

Identify the type of circuit given. Write the state table and draw the state diagram for the same



Exercise

- Design a sequential circuit with two D flip-flops A and B and one input X. when X=0, the state of the circuit remains the same. When X=1, the circuit goes through the transition from 00 to 01 to 11 to 10 back to 00, repeat