ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Electronics for Information Technology

(Điện tử cho Công nghệ Thông tin)

IT3420E

Đỗ Công Thuần

Department of Computer Engineering

Email: thuandc@soict.hust.edu.vn

General Information

- Course: **Electronics for Information Technology**
- ID Number: IT3420
- Credits: 2 (2-1-0-4)
- Lecture/Exercise: 32/16 hours (48 hours, 16 weeks)
- Evaluation:
  - Midterm examination and weekly assignment: **50%**
  - Final examination: **50%**
- Learning Materials:
  - Lecture slides
  - Textbooks
    - ***Introductory Circuit Analysis*** (2015), 10th – 13th ed., Robert L. Boylestad
    - ***Electronic Device and Circuit Theory*** (2013), 11th ed., Robert L. Boylestad, Louis Nashelsky
    - ***Microelectronics Circuit Analysis and Design*** (2006), 4th ed., Donald A. Neamen
    - ***Digital Electronics: Principles, Devices and Applications*** (2007), Anil K. Maini

Contact Your Instructor

- You can reach me through office in **Room 802, B1 Building**, HUST.
  - You should make an appointment by email before coming.
  - If you have urgent things, just come and meet me!
- You can also reach me at the following **email** any time. This is the best way to reach me!
  - thuandc@soict.hust.edu.vn

Course Contents

- The Concepts of Electronics for IT
- **Chapter 1**: Passive Electronic Components and Applications
- **Chapter 2**: Semiconductor Components and Applications
- **Chapter 3**: Operational Amplifiers
- **Chapter 4**: Fundamentals of Digital Circuits
- **Chapter 5**: Logic Gates
- **Chapter 6**: Combinational Logic
- **Chapter 7**: Sequential Logic

Chapter 7:
Sequential Logic

1. Concepts
2. Flip Flop
3. Flip Flop Types
4. Finite State Machine (FSM)
5. Applications

References:

Digital electronics: Principles, Devices, and Applications, Anil Kumar Maini 2007 John Wiley & Sons

Fundamentals of Logic Design, Seventh Edition, Charles H. Roth, Jr. and Larry L. Kinney

Digital Fundamentals, Thomas L. Floyd, Eleventh Edition, Pearson Education Limited 2015

Contents

1. Concepts
2. Flip Flop
3. Flip Flop Types
4. Finite State Machine (FSM)
5. Applications

Definition

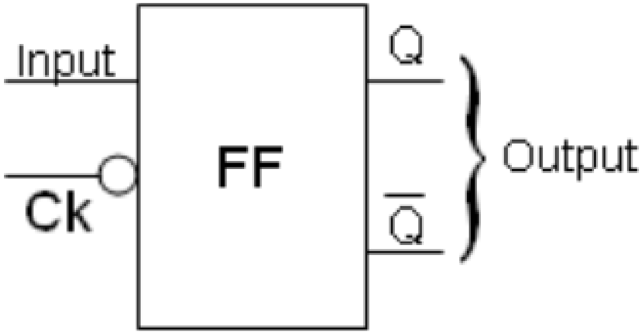
- A sequential logic circuit is one where the output depends upon **not only the present but also the past state of inputs**.
- Comprising both **logic gates** and **memory elements** (e.g. flip flops).

Contents

1. Concepts
2. Flip Flop
3. Flip Flop Types
4. Finite State Machine (FSM)
5. Applications

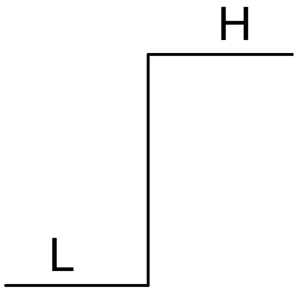
Flip Flop

- A flip flop is a bistable circuit, which remains in a particular output state indefinitely until something is done to change that output status.
- Most flip flops have both synchronous and asynchronous inputs:
  - **Asynchronous inputs** are those that operate independently of the synchronous inputs and the clock input.
  - **Synchronous inputs** are those whose effect on the flip flop output is synchronized with the clock input.
    - Level-triggered
    - Edge-triggered
    - Pulse-triggered



Level-Triggered Flip Flop

- High level:
  - The flip flop is triggered when the clock is at its HIGH state.
  - Otherwise, the flip flop's state is unchanged.

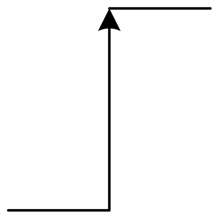


Level-triggered

- Low level:
  - The flip flop is triggered when the clock is at its LOW state.

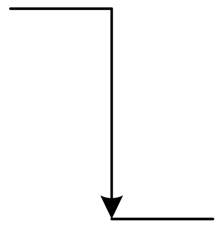
Edge-Triggered Flip Flop

- Rising edge:
  - The flip flop is triggered when the clock is changing from 0 to 1.
  - Otherwise, the flip flop's state is unchanged.



Rising edge triggered

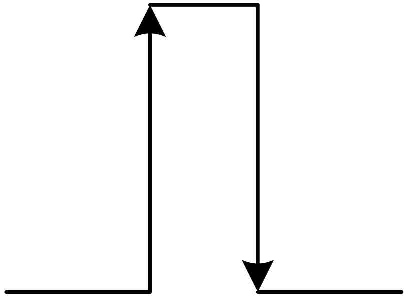
- Falling edge:
  - The flip flop is triggered when the clock is changing from 1 to 0.
  - Otherwise, the flip flop's state is unchanged.



Falling edge triggered

Pulse-Triggered Flip Flop

- The flip flop is triggered when the clock is changing from 0 to 1 for a small duration, and then is changing from 1 to 0.
- Otherwise, the flip flop’s state is unchanged.



Pulse-triggered

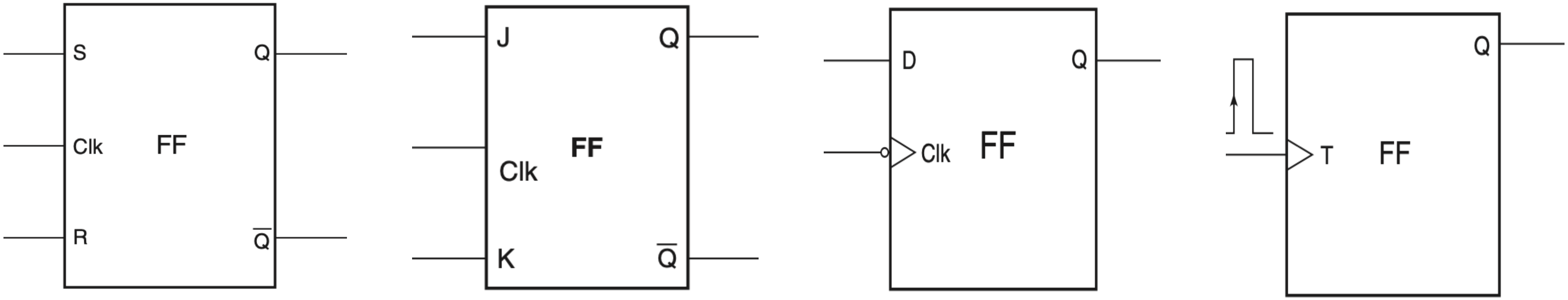
Contents

1. Concepts
2. Flip Flop
3. Flip Flop Types
4. Finite State Machine (FSM)
5. Applications

Flip Flop Types

- 4 types of flip-flops:

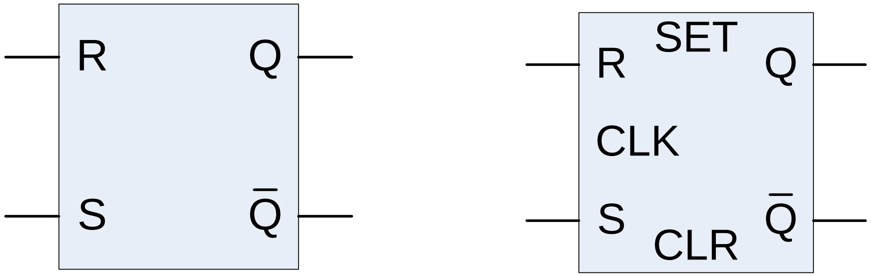
RS	Reset - Set
JK	Jordan & Kelly
D	Delay
T	Toggle



RS Flip Flop

- There are synchronous (clocked) and asynchronous RS flip flops.

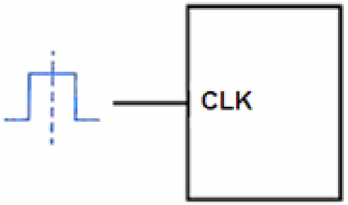
- Block diagram:



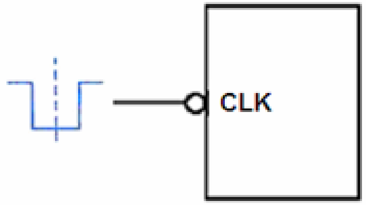
Clocked RS Flip Flop

- The outputs change states as per the inputs only on the occurrence of a clock pulse.
- Clocked flip flops could be:
    - Level-triggered (high, low)
    - Edge-triggered (rising, falling)

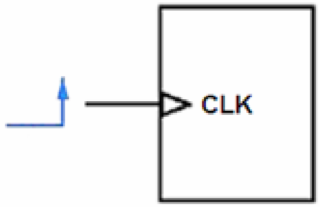
CLK	S	R	Q	Q'
'0'	x	x	Q	Q'
'1'	0	0	Q	Q'
	0	1	0	1
	1	0	1	0
	1	1	x	x



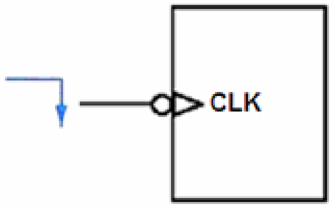
High-level triggered



Low-level triggered



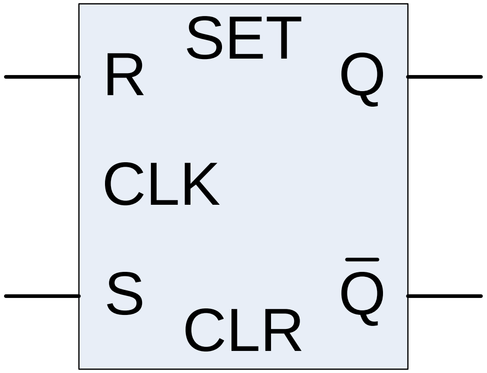
Rising-edge triggered

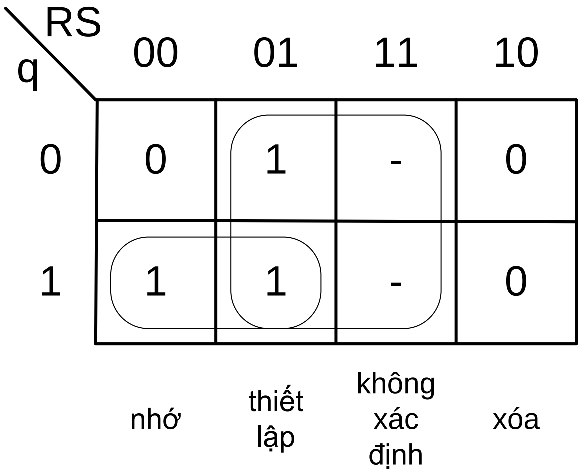


Falling-edge triggered

RS Flip Flop

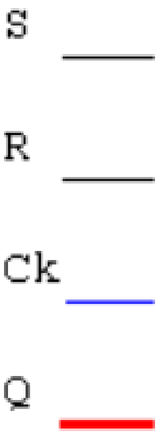
- Output waveform:

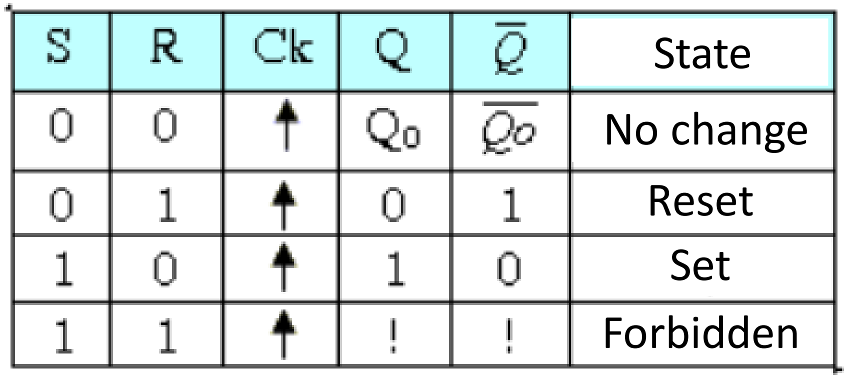




q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

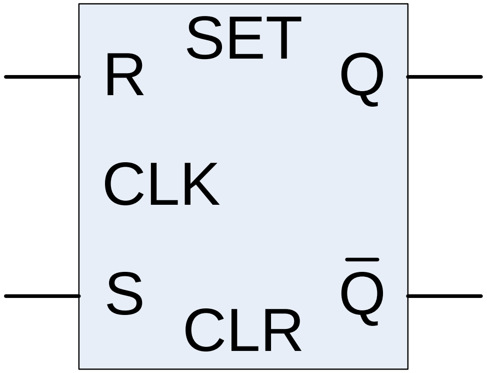




S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

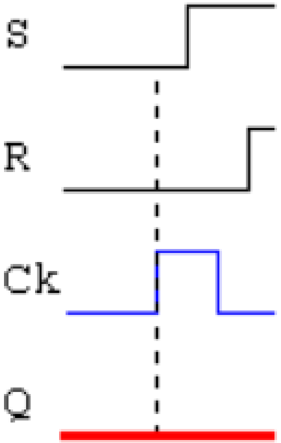
RS Flip Flop

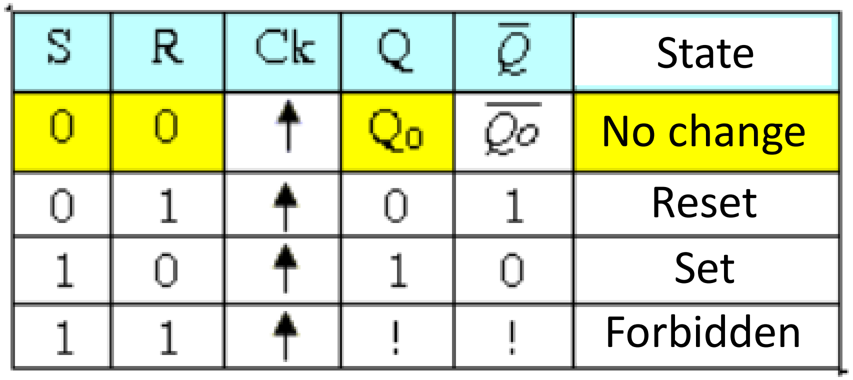
- Output waveform:



q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

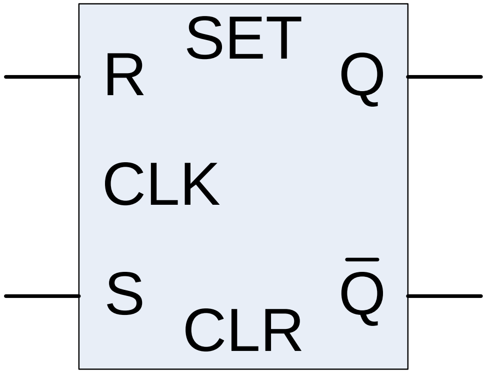


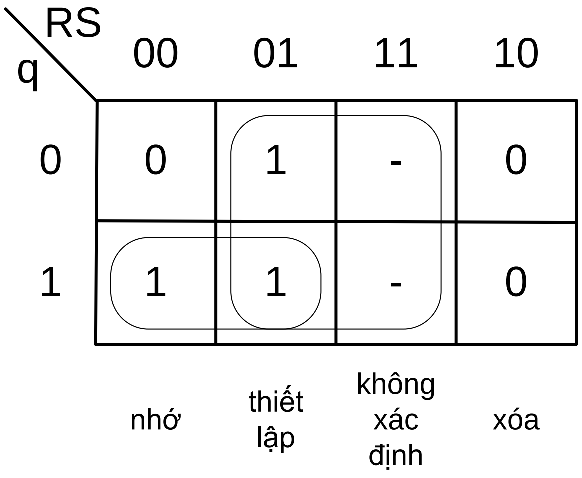


S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

RS Flip Flop

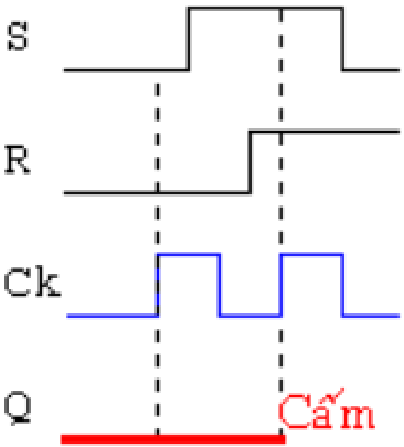
- Output waveform:

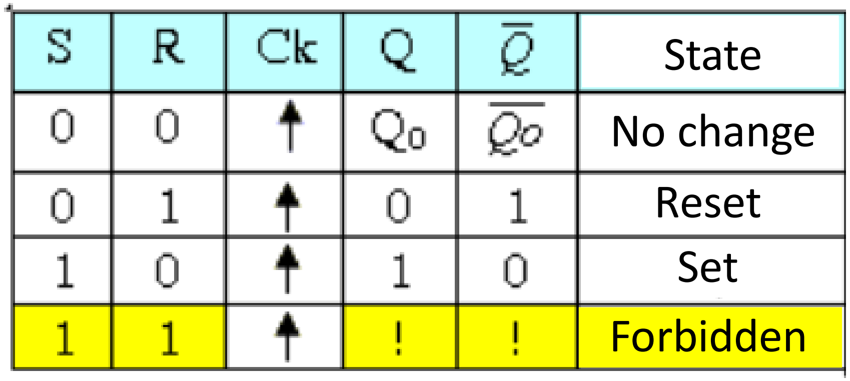




q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

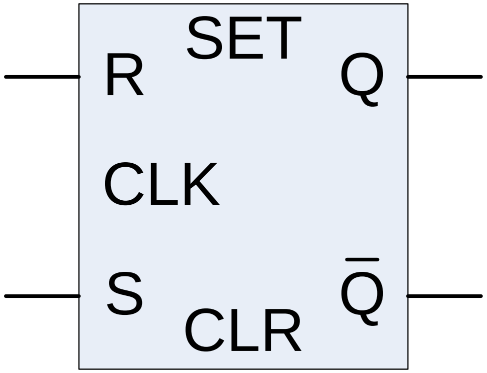


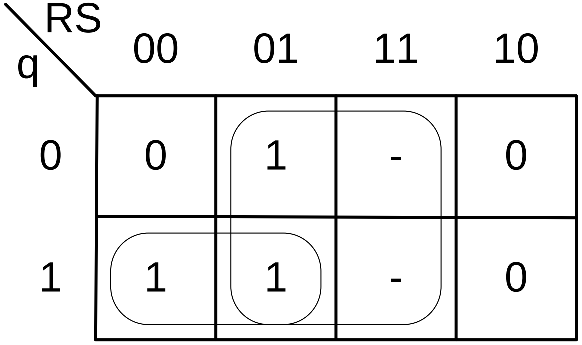


S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

RS Flip Flop

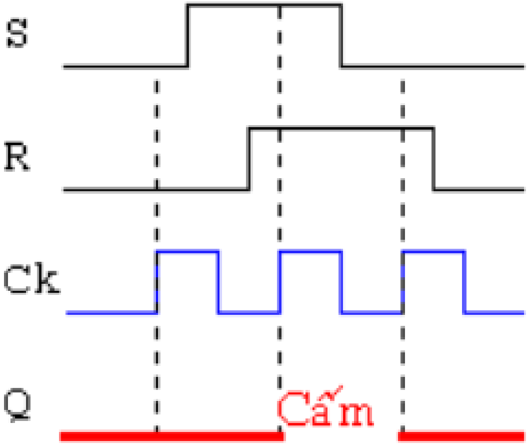
- Output waveform:





q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

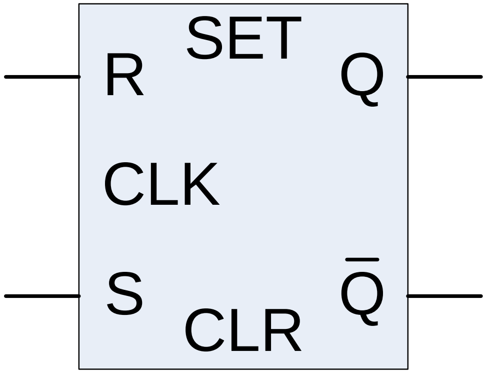
$$Q = S + q\overline{R}$$

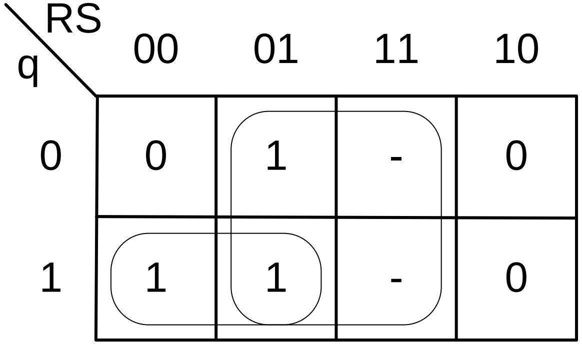


S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

RS Flip Flop

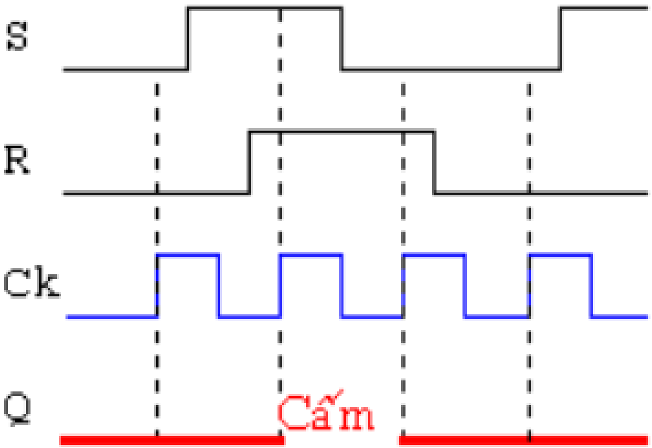
- Output waveform:

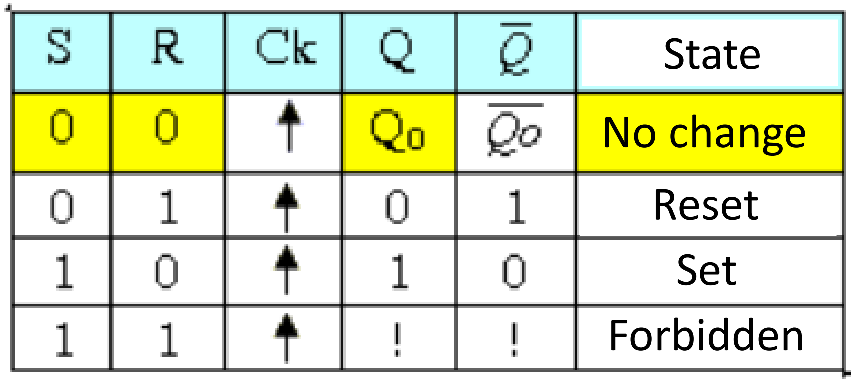




q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

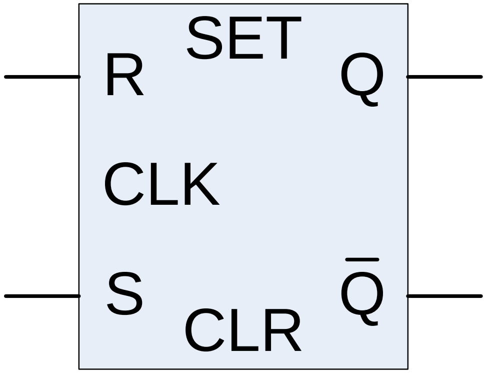


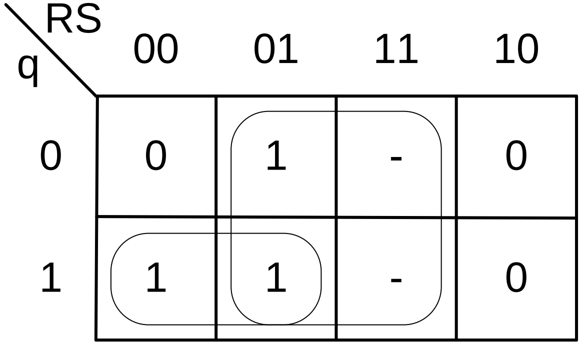


S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

RS Flip Flop

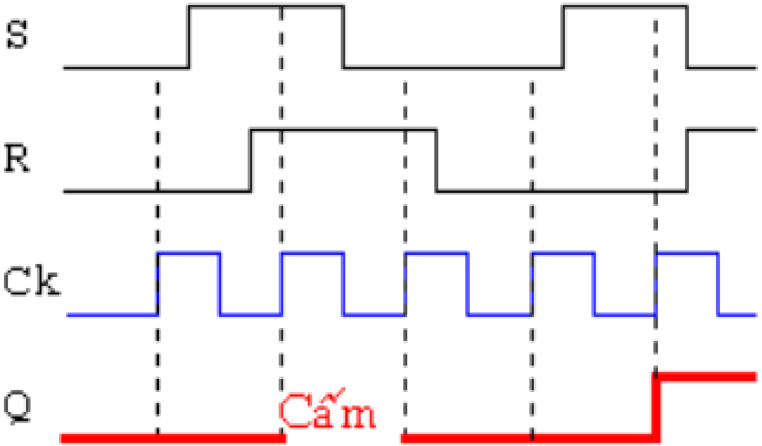
- Output waveform:

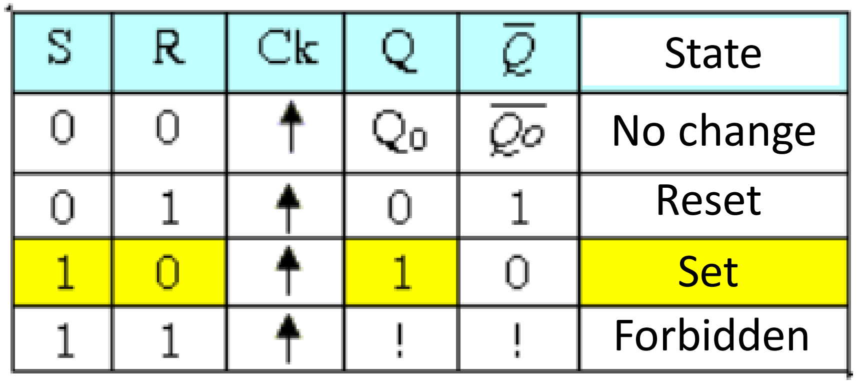




q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

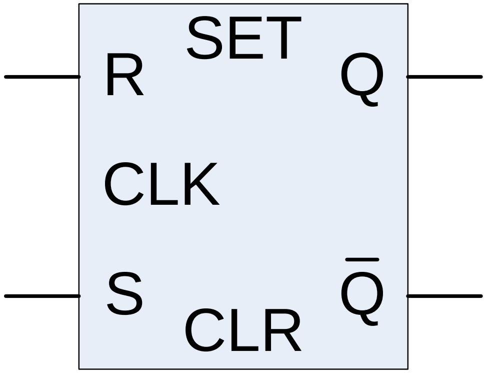


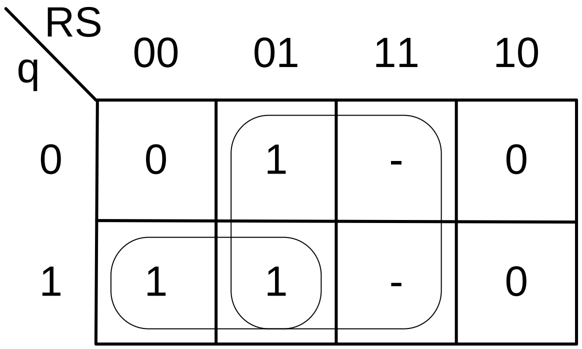


S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

RS Flip Flop

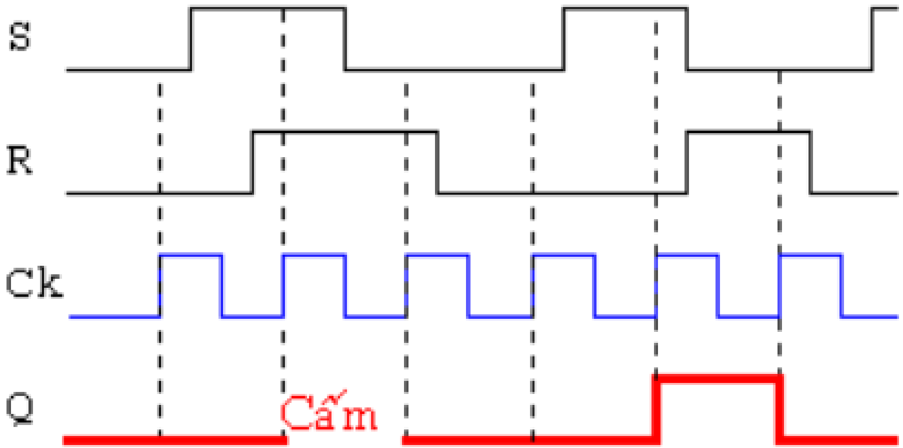
- Output waveform:

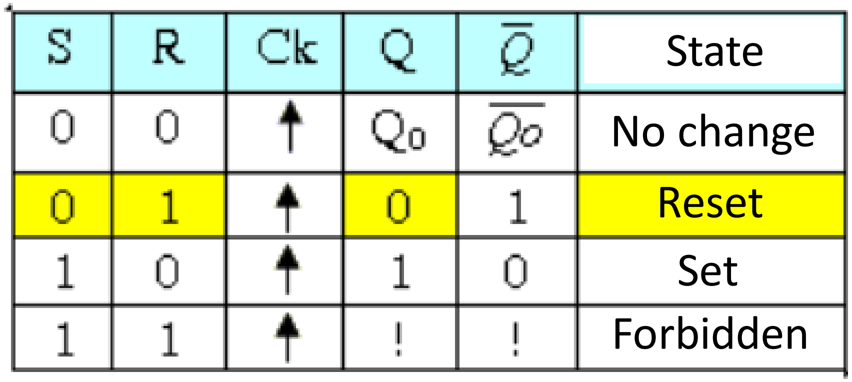




q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

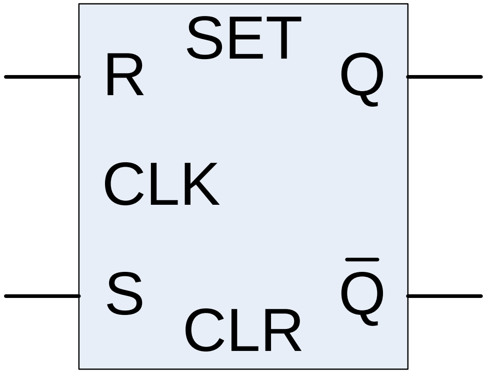




S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

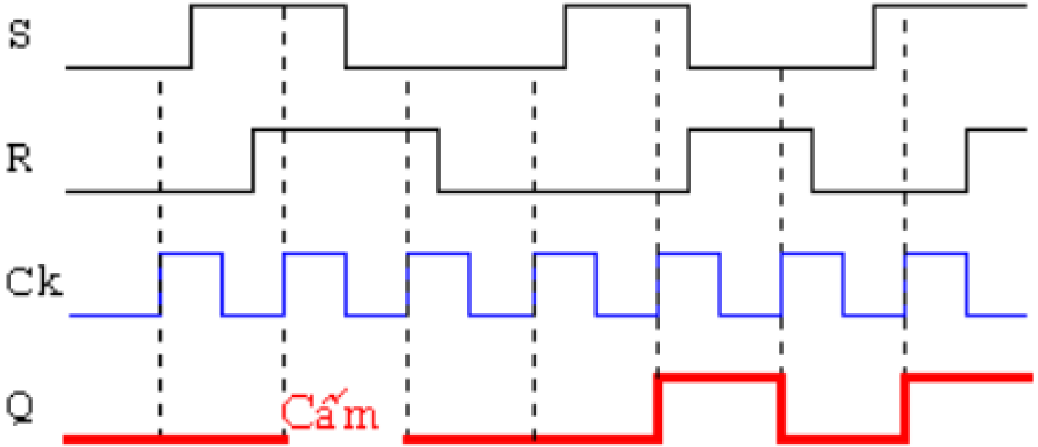
RS Flip Flop

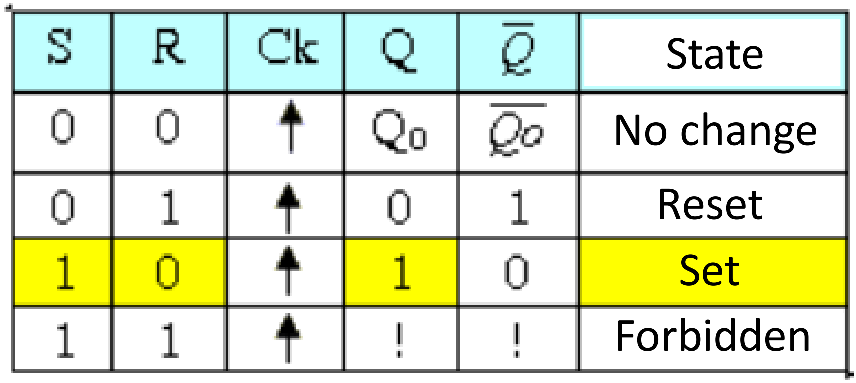
- Output waveform:



q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0
	nhớ	thiết lập	không xác định	xóa

$$Q = S + q\overline{R}$$

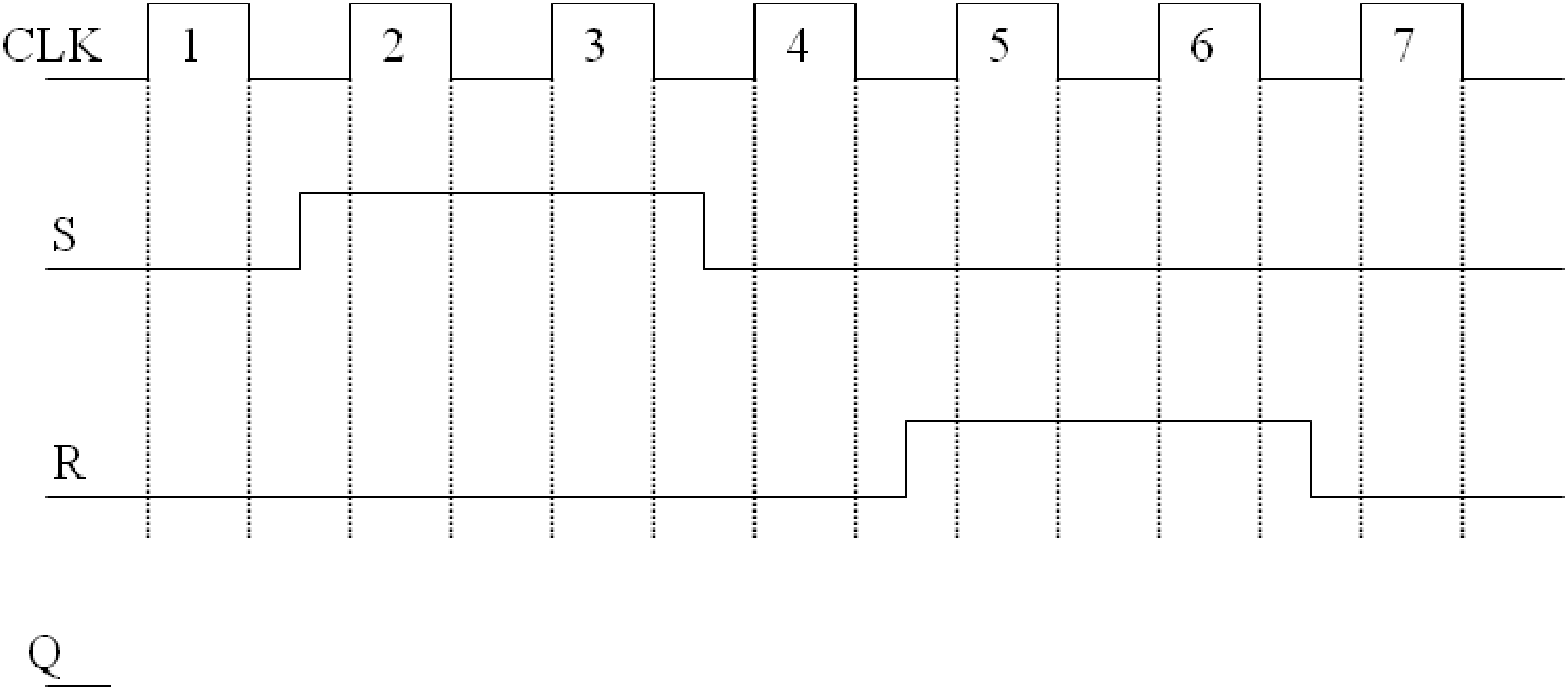




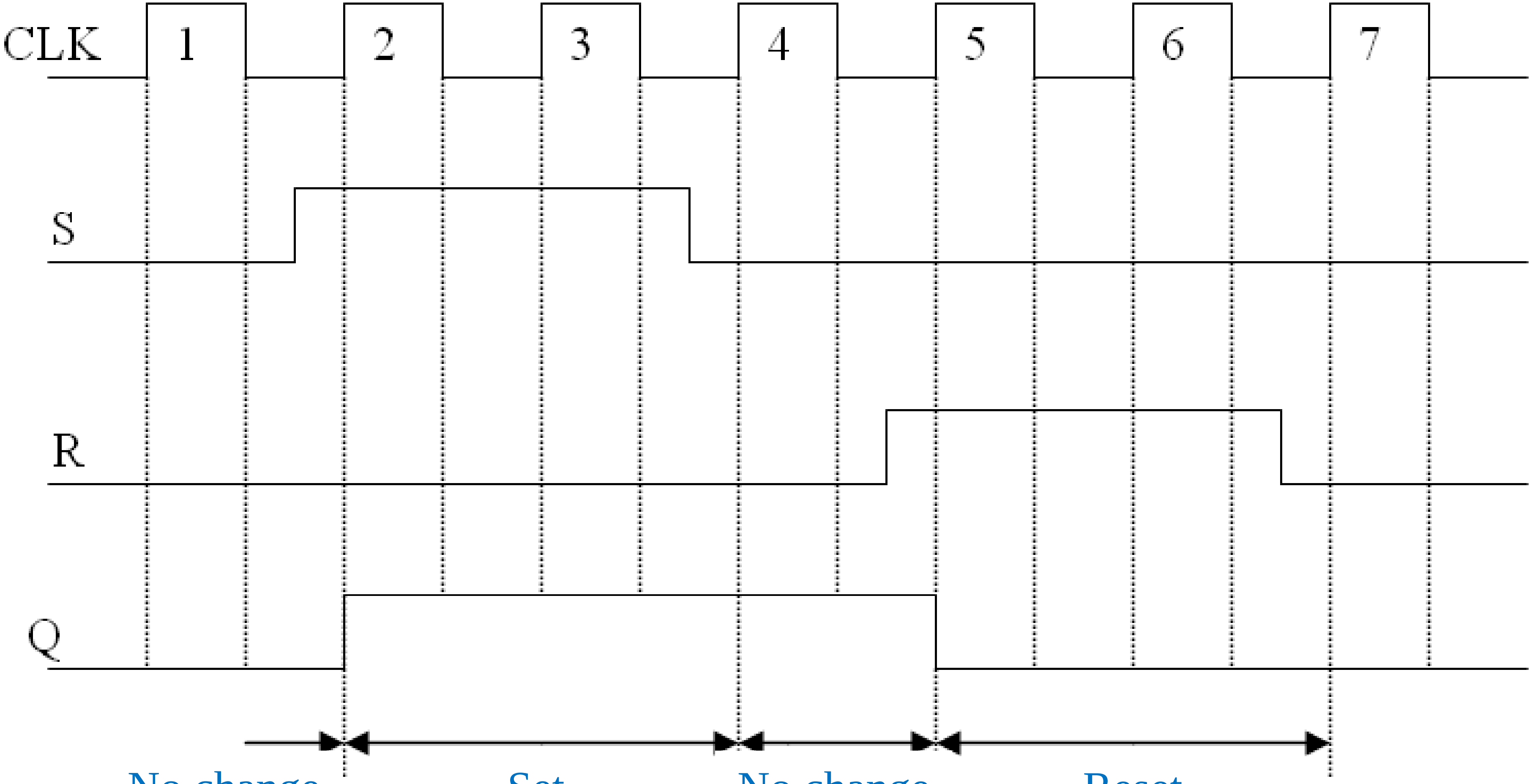
S	R	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	!	!	Forbidden

Example 1

- For a high-level triggered RS flip flop with R- and S-input waveforms, determine the Q-output waveform.



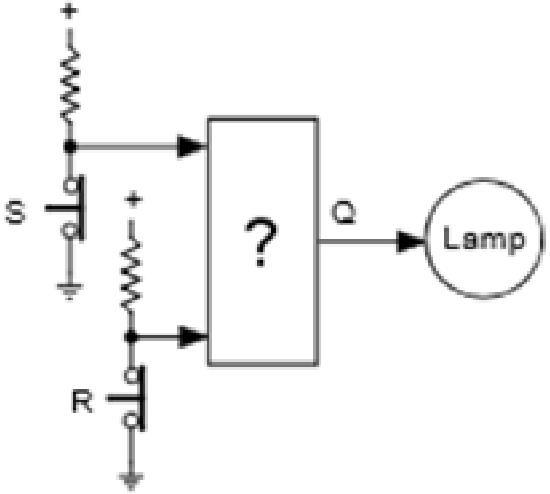
Example 1

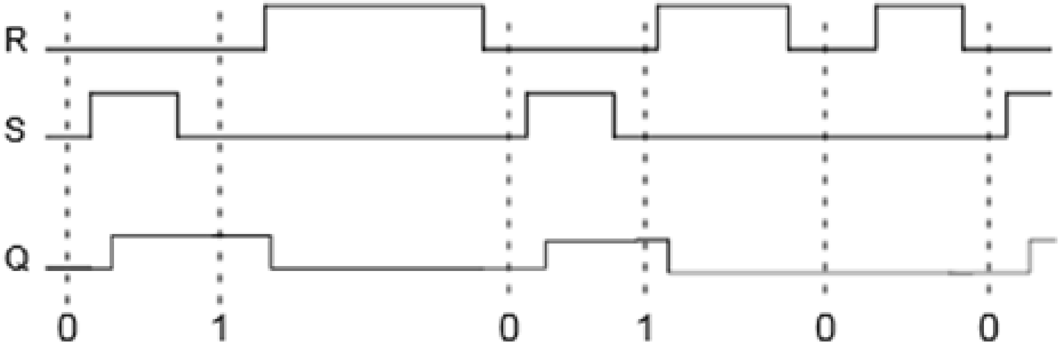


No change Set No change Reset

Example 2

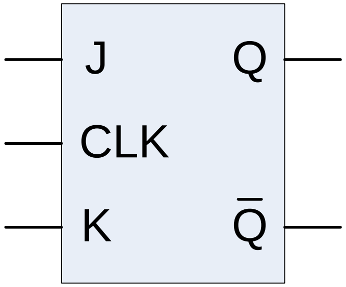
- Design a circuit to turn on/off a lamp.
  - When the button S is pressed, the lamp is ON. When the button S is released, the lamp is still ON.
  - When the button R is pressed, the lamp is OFF. When the button R is released, the lamp is still OFF.
  - The buttons S and R are not pressed at the same time.



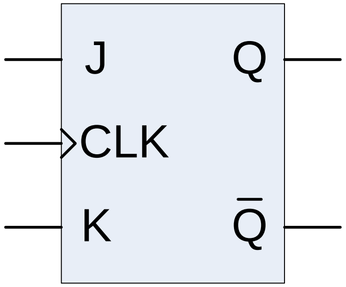


JK Flip Flop

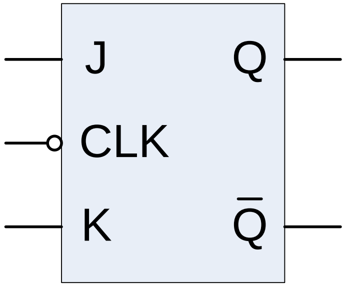
- A JK flip flop overcomes the problem of a forbidden input combination of an RS flip flop.
- Block diagram:



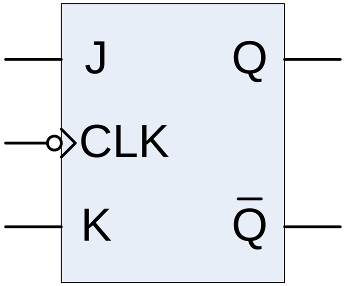
High-level triggered



Rising-edge triggered



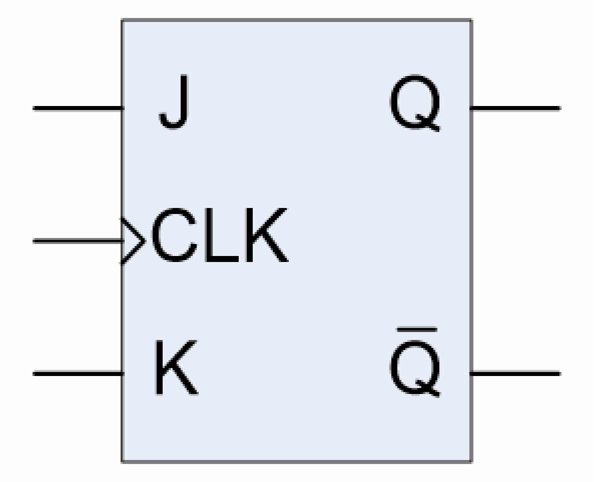
Low-level triggered



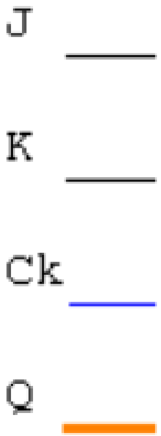
Falling-edge triggered

JK Flip Flop

- Output waveform:



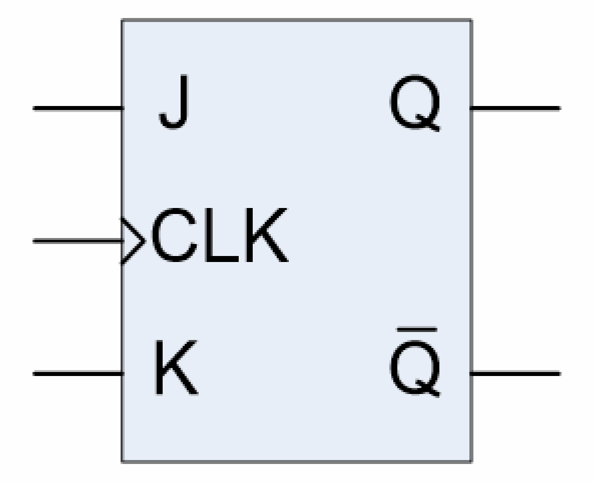
Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

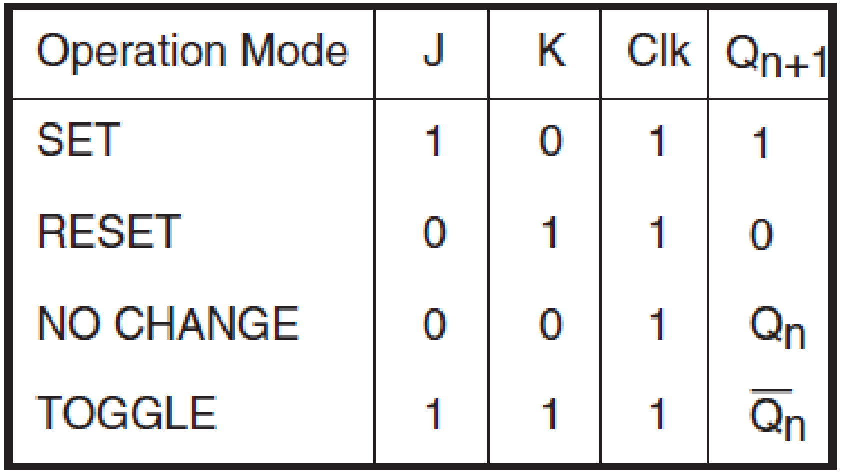


J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

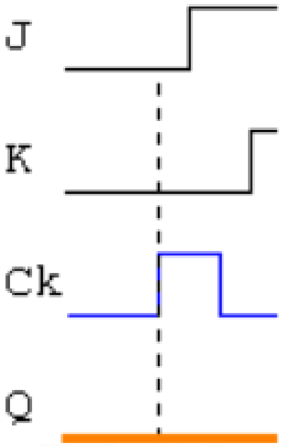
JK Flip Flop

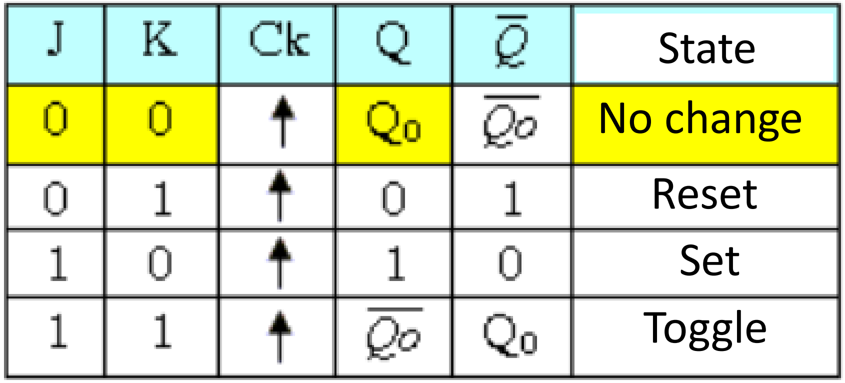
- Output waveform:





Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

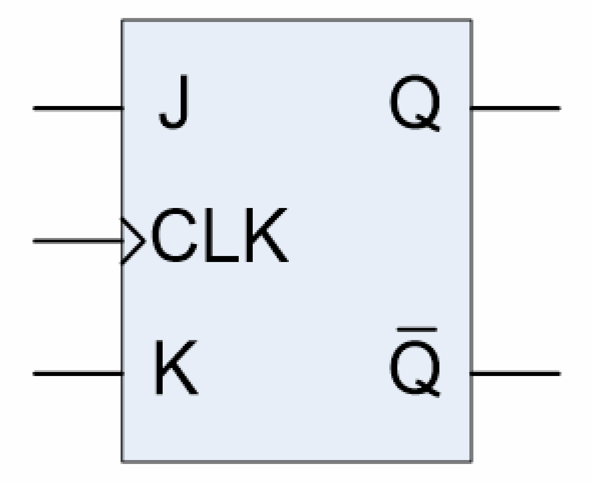


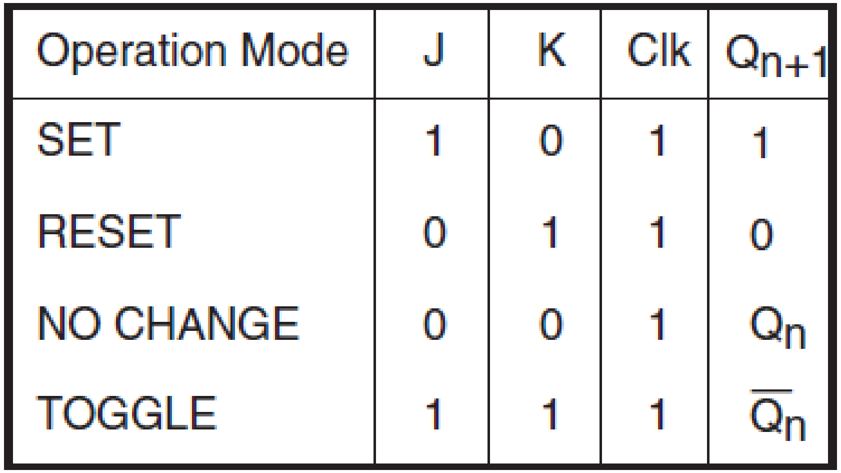


J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

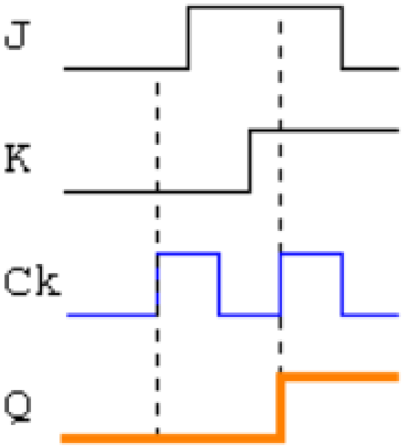
JK Flip Flop

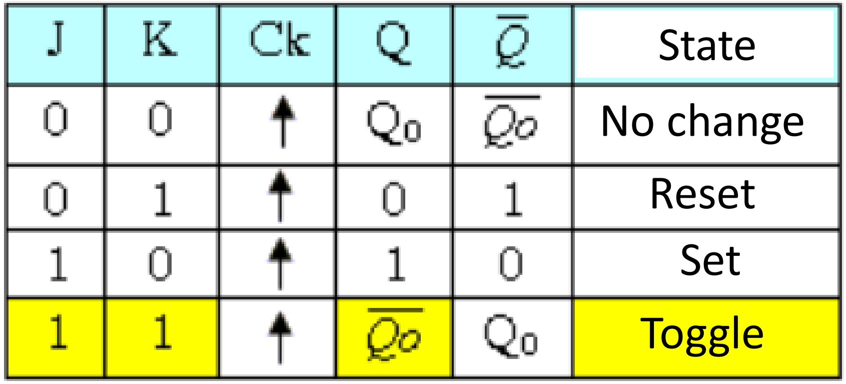
- Output waveform:





Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

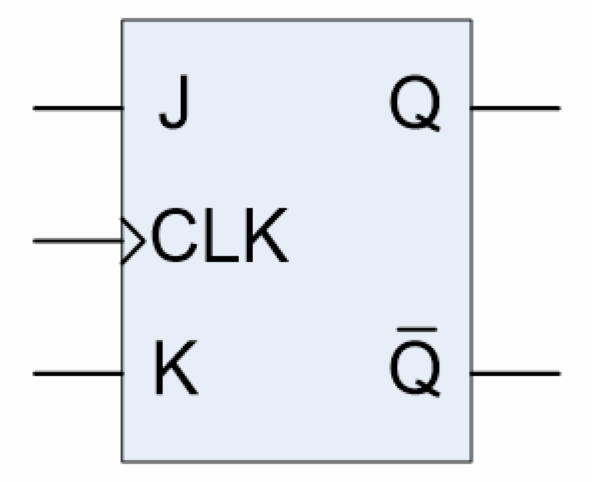


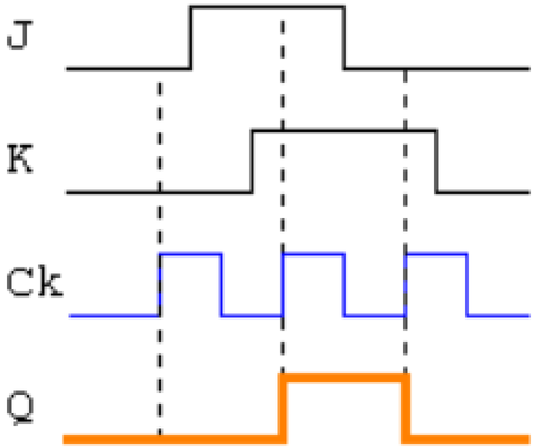


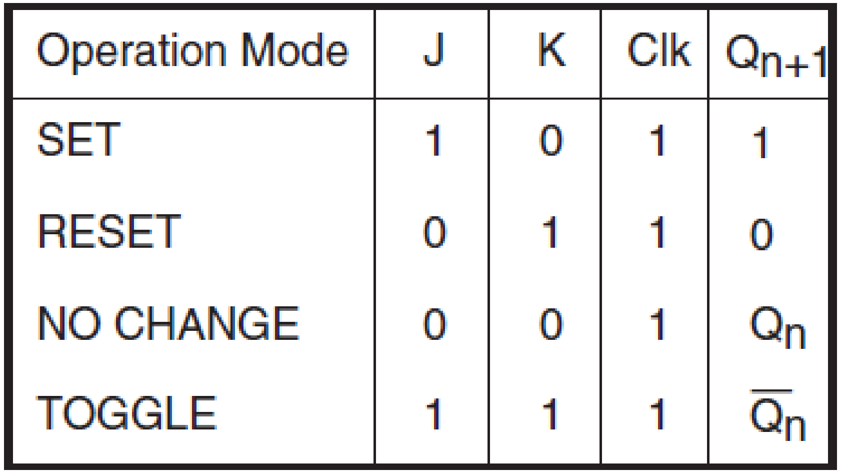
J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

JK Flip Flop

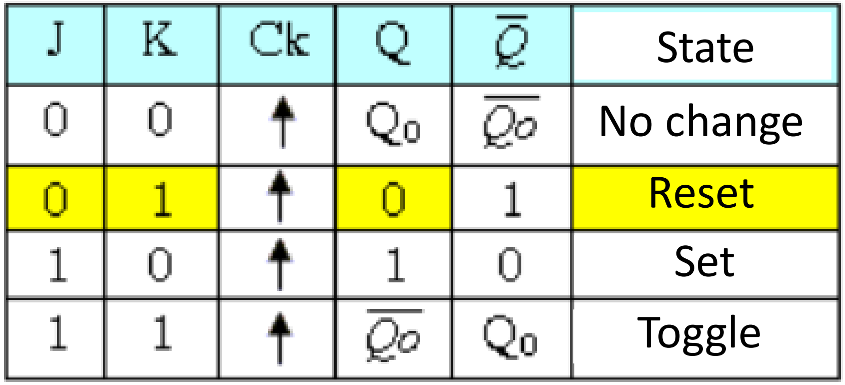
- Output waveform:







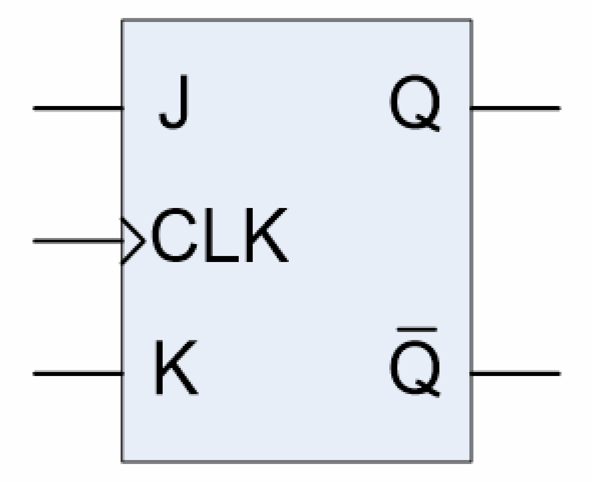
Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

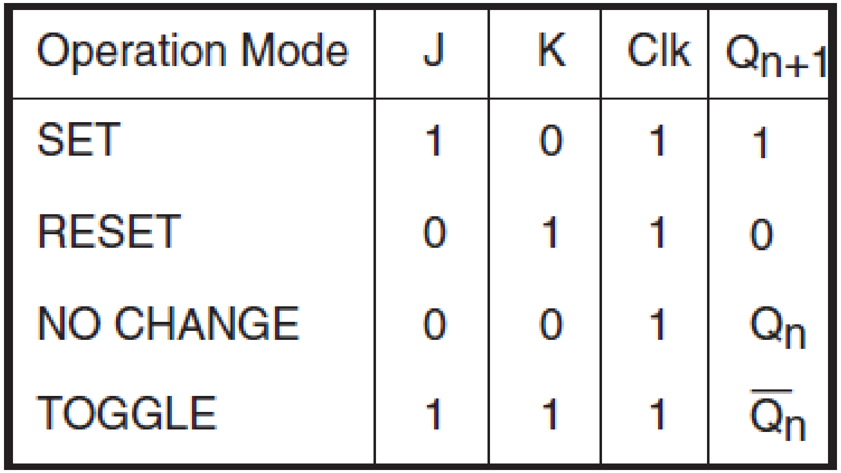


J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

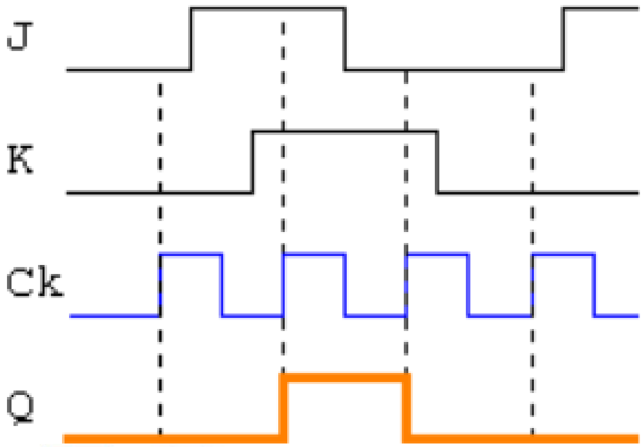
JK Flip Flop

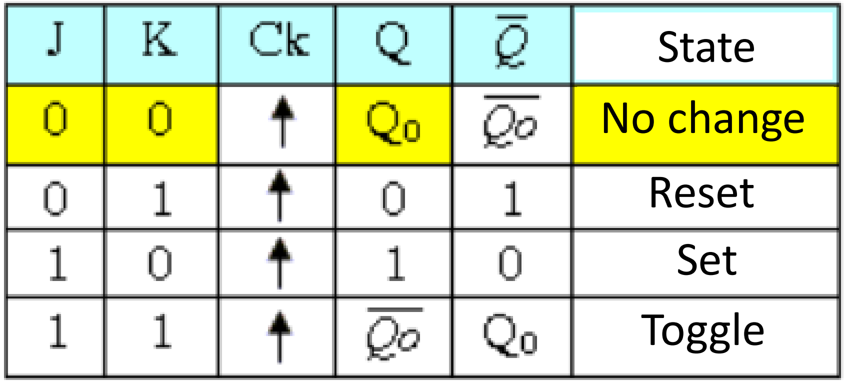
- Output waveform:





Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

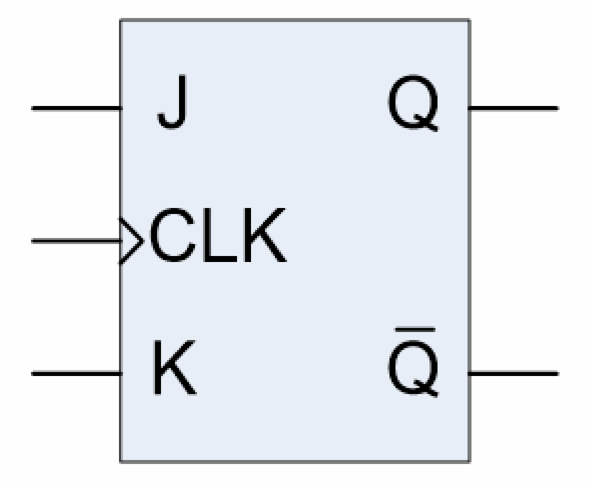


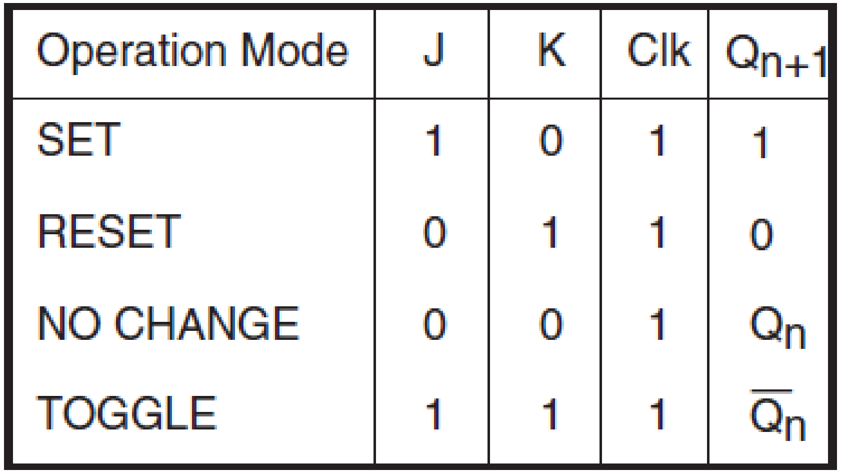


J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

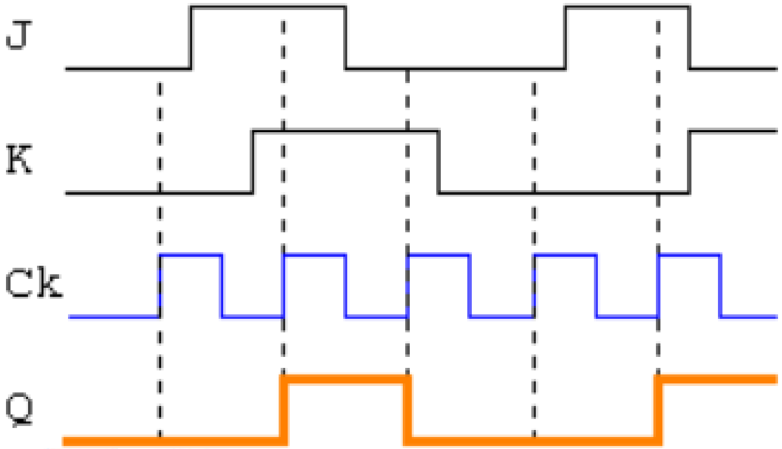
JK Flip Flop

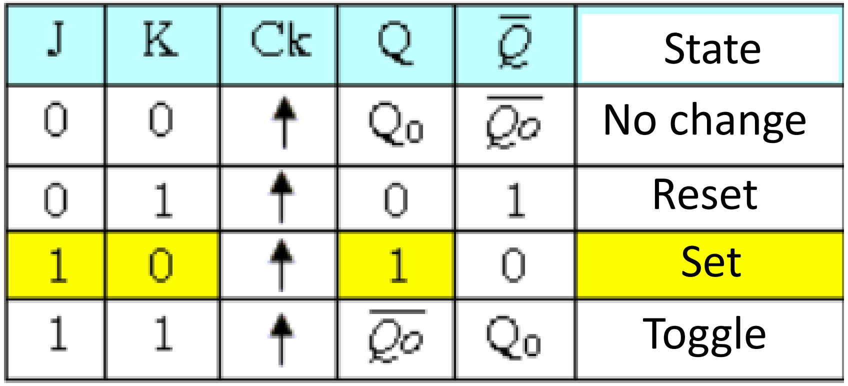
- Output waveform:





Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q}_n$

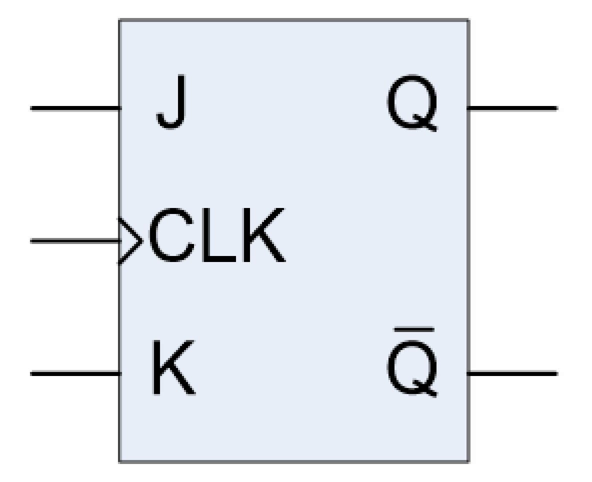


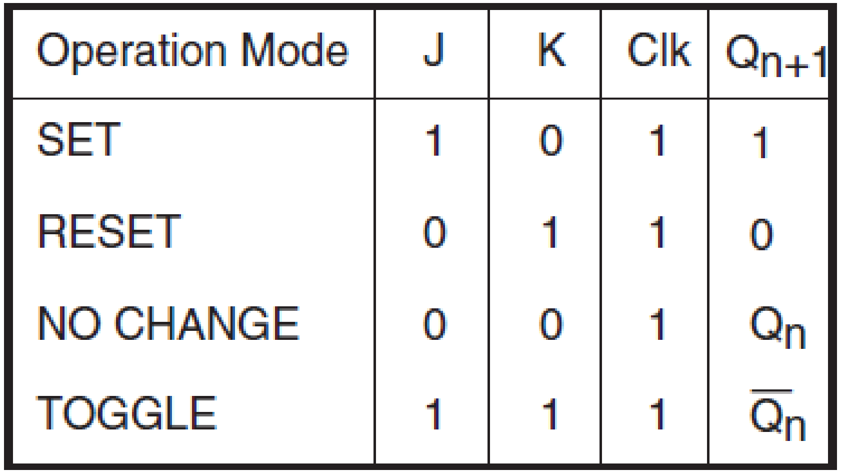


J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

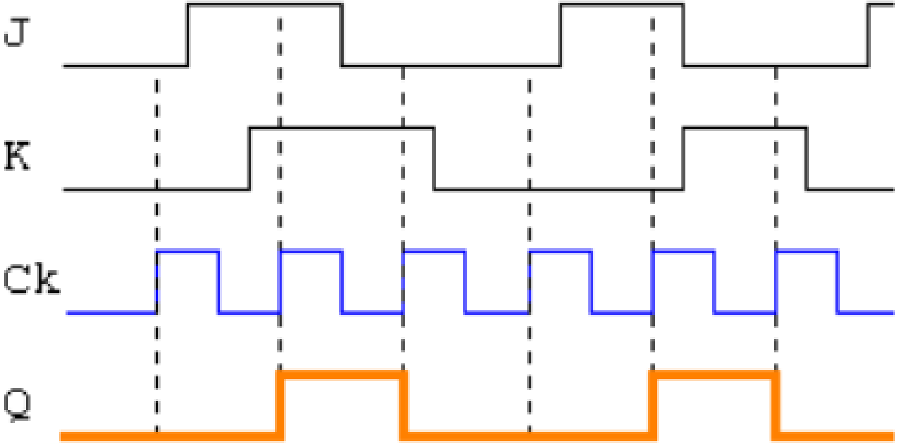
JK Flip Flop

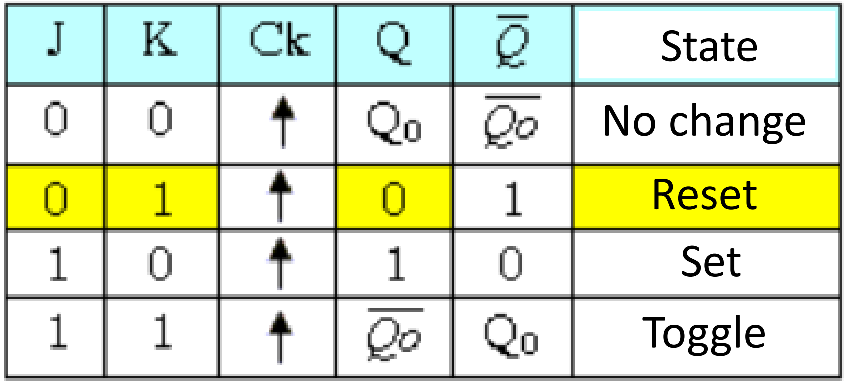
- Output waveform:





Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

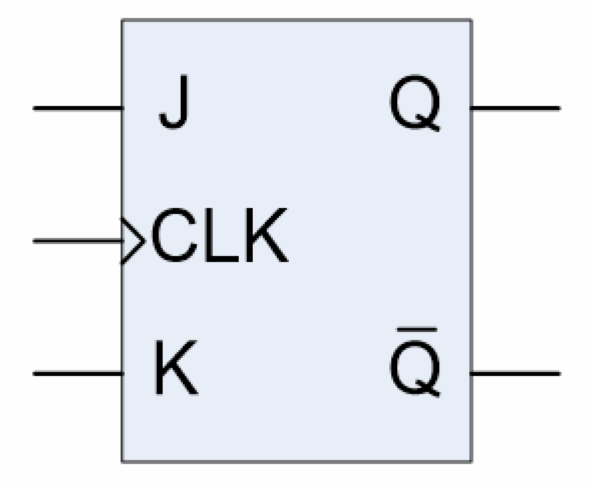


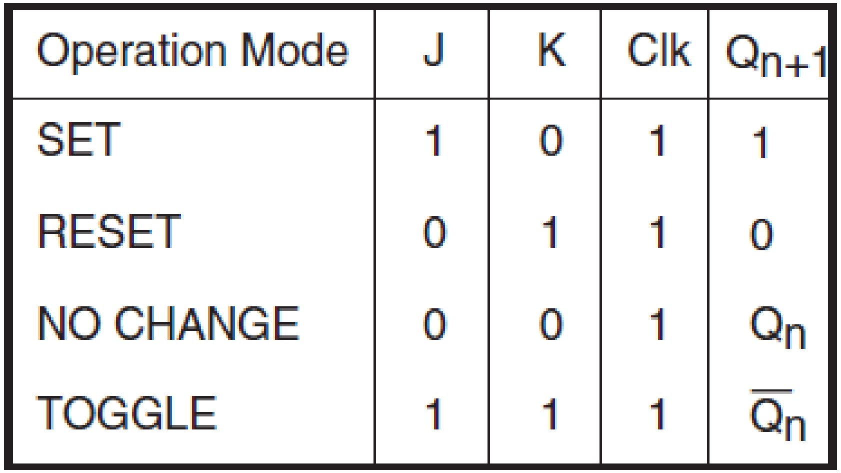


J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

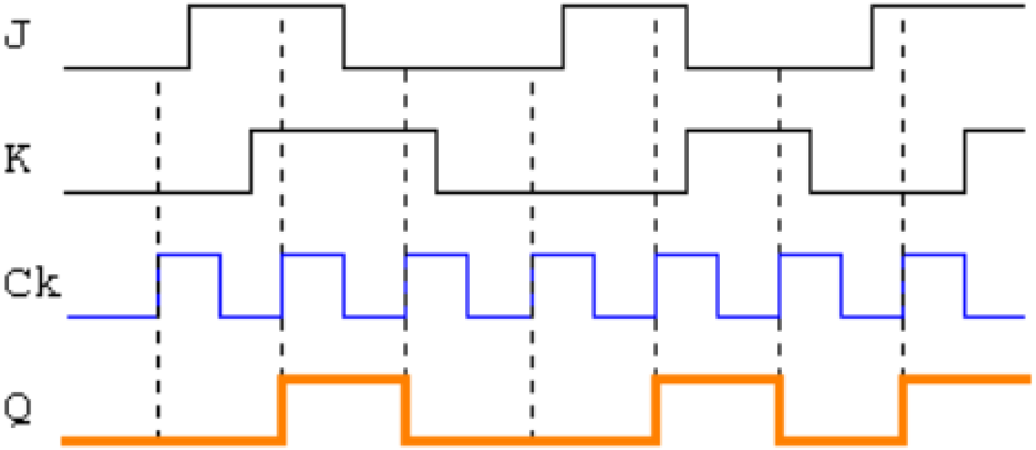
JK Flip Flop

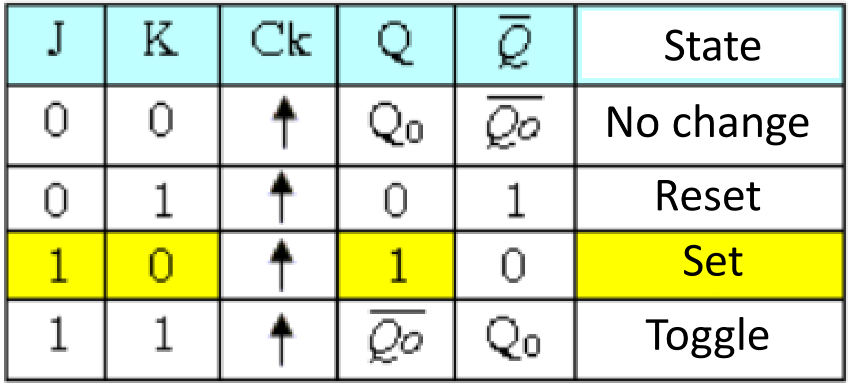
- Output waveform:





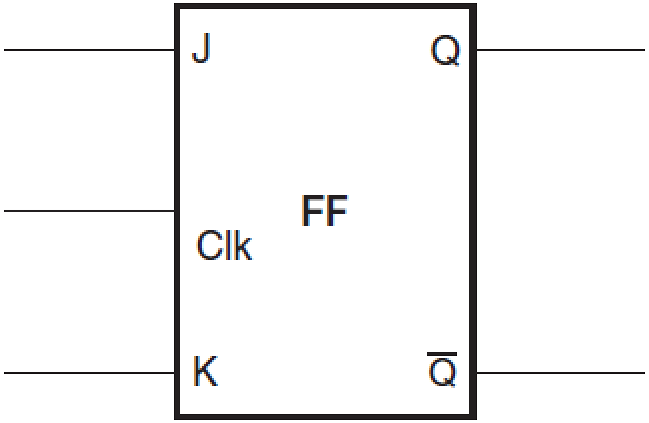
Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q}_n$





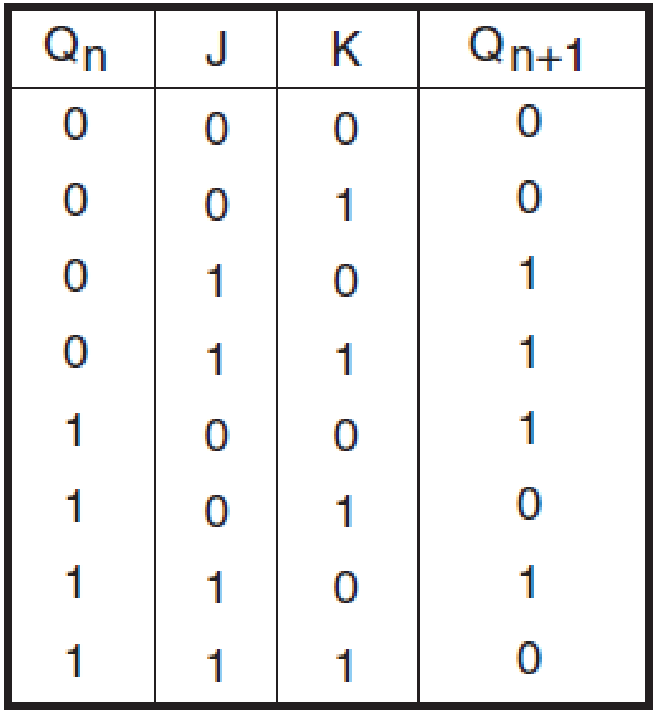
J	K	Ck	Q	$\overline{Q}$	State
0	0	↑	Q_0	$\overline{Q_0}$	No change
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	$\overline{Q_0}$	Q_0	Toggle

JK Flip Flop with Active HIGH Inputs

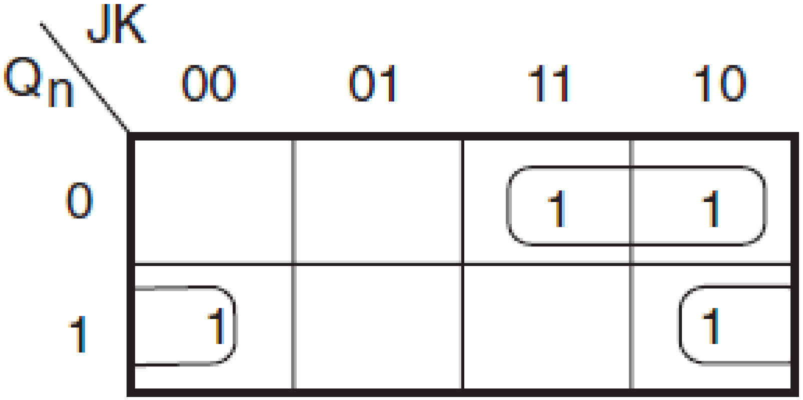


Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q_n}$

JK Flip Flop with Active HIGH Inputs



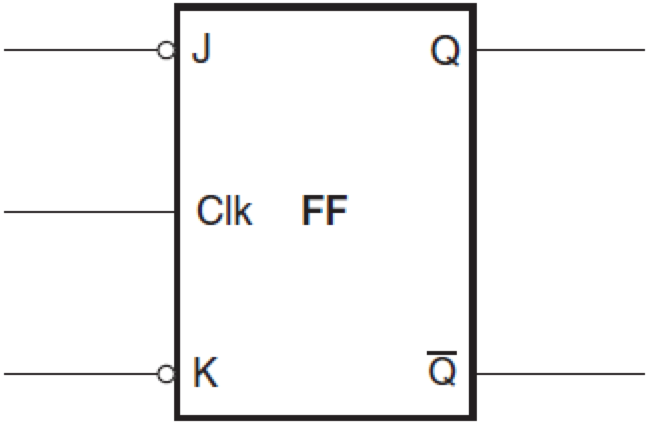
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



Q_n \ JK	00	01	11	10
0			1	1
1	1			1

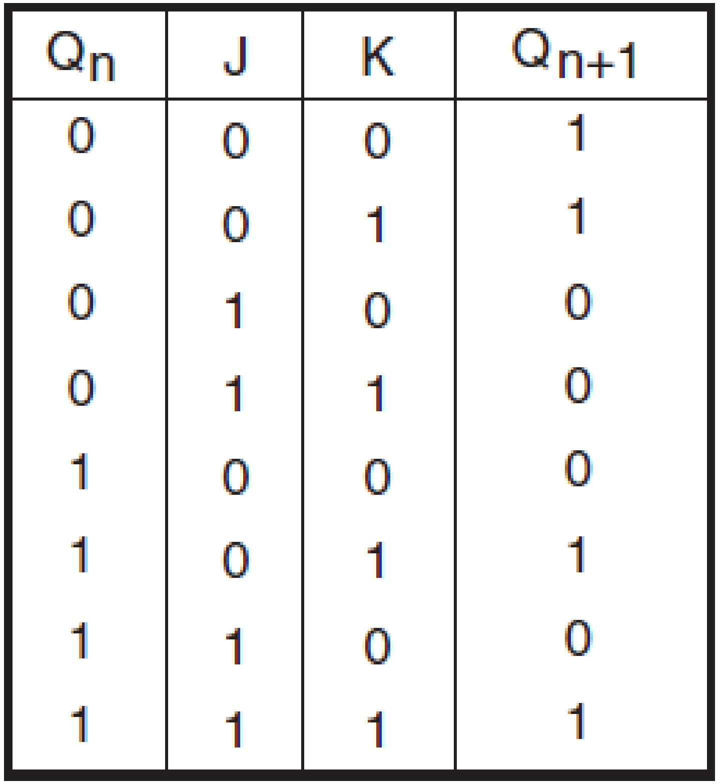
$$Q_{n+1} = J.\overline{Q_n} + \overline{K}.Q_n$$

JK Flip Flop with Active LOW Inputs

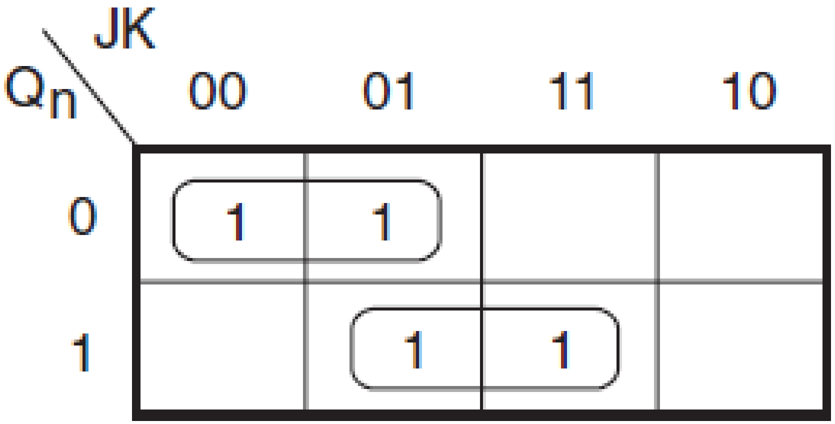


Operation Mode	J	K	Clk	Q_{n+1}
SET	0	1	1	1
RESET	1	0	1	0
NO CHANGE	1	1	1	Q_n
TOGGLE	0	0	1	$\overline{Q}_n$

JK Flip Flop with Active LOW Inputs



Q_n	J	K	Q_{n+1}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

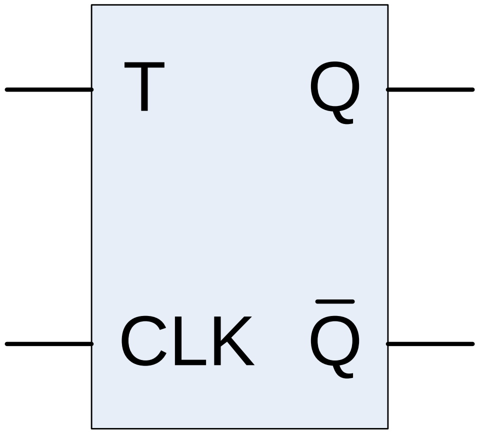


Q_n \ JK	00	01	11	10
0	1	1		
1		1	1	

$$Q_{n+1} = \overline{J}.\overline{Q_n} + K.Q_n$$

T Flip Flop

- T flip flop (or Toggle flip flop) changes state every time it is triggered at its T input.
- Block diagram:



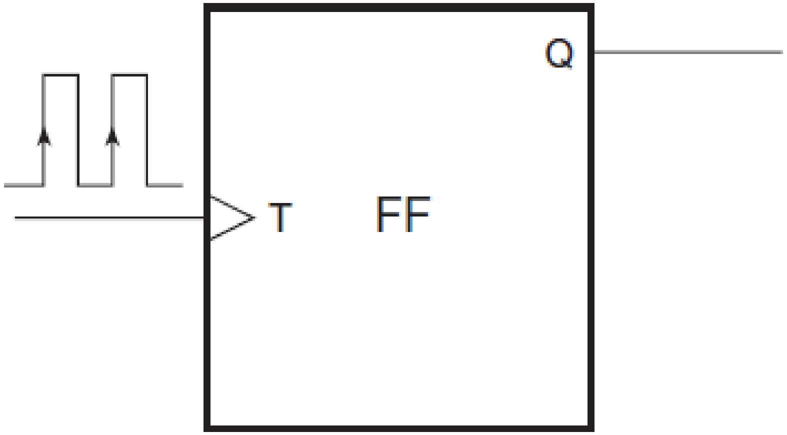
Positive Edge-Triggered T Flip Flop

- Function table:

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Q_n \ T	0	1
0		1
1	1	

$$Q_{n+1} = T.\overline{Q_n} + \overline{T}.Q_n$$



T	Q_n	Q_{n+1}
↑	0	1
↑	1	0

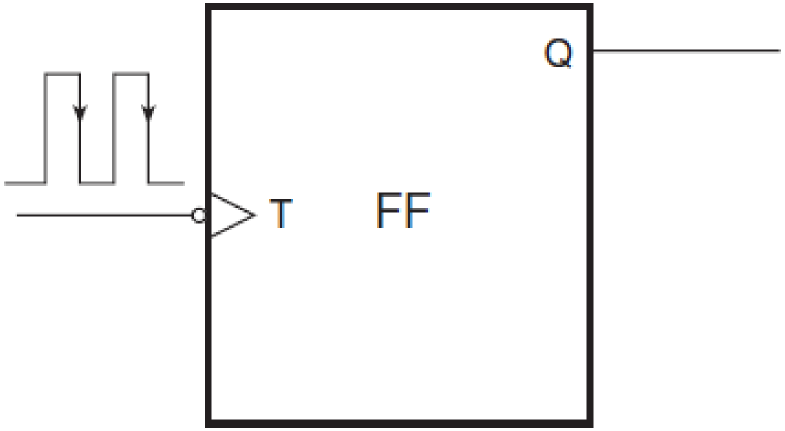
Negative Edge-Triggered T Flip Flop

- Function table:

Q_n	T	Q_{n+1}
0	0	1
0	1	0
1	0	0
1	1	1

Q_n \ T	0	1
0	1	
1		1

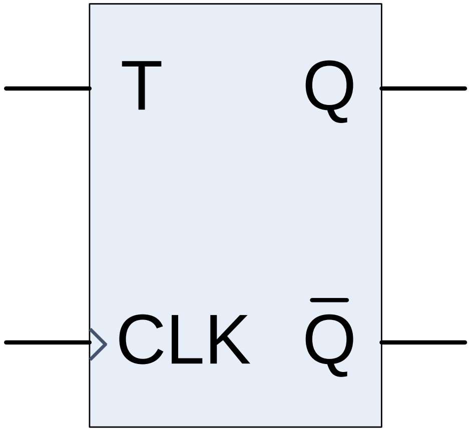
$$Q_{n+1} = \overline{T}.\overline{Q_n} + T.Q_n$$



T	Q_n	Q_{n+1}
↓	0	1
↓	1	0

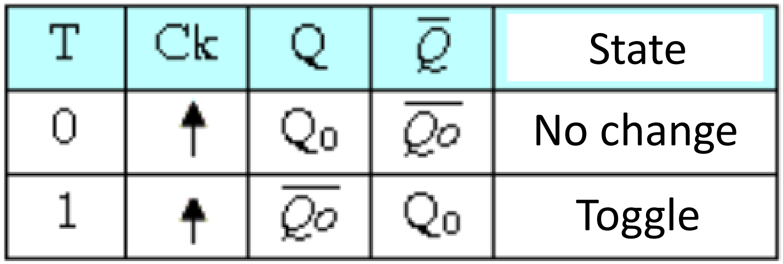
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

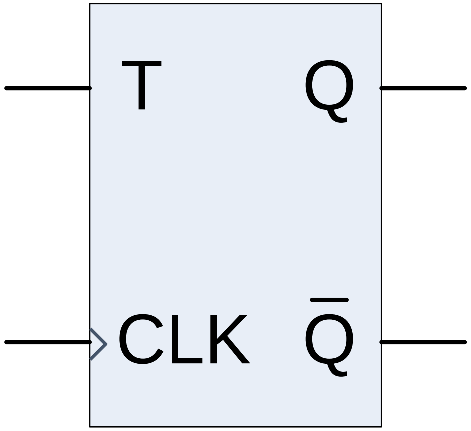
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

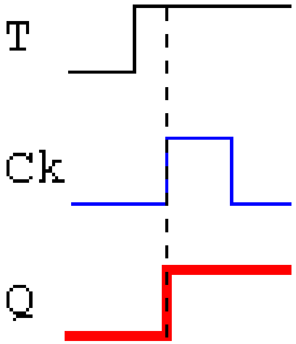
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

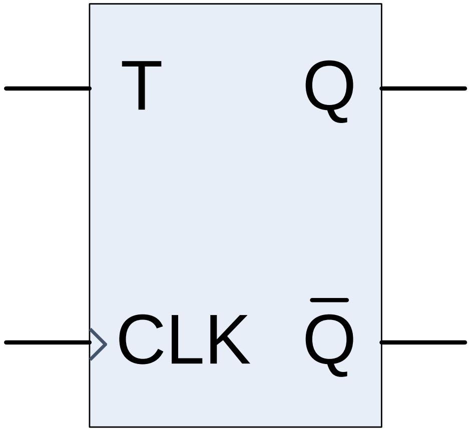
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

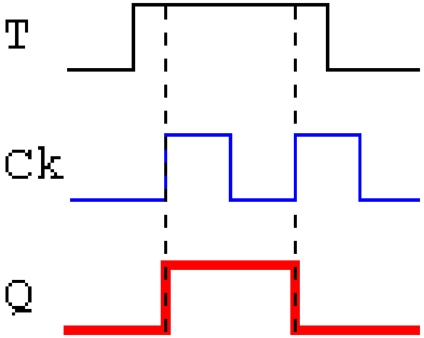
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

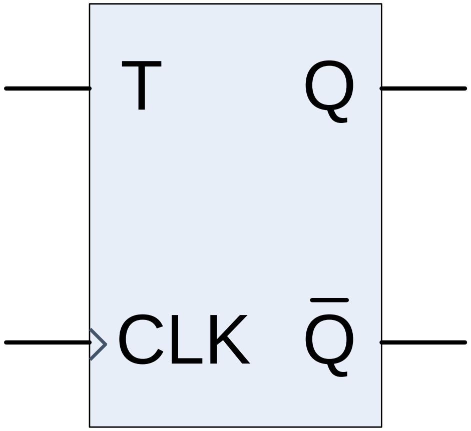
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

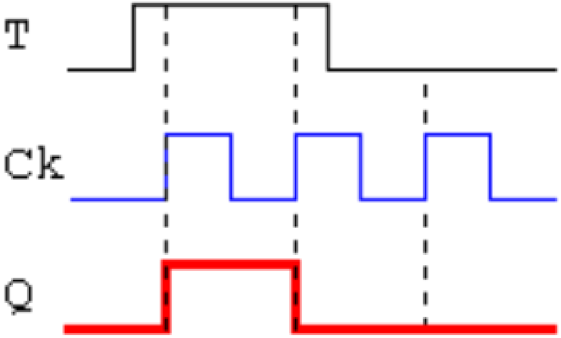
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

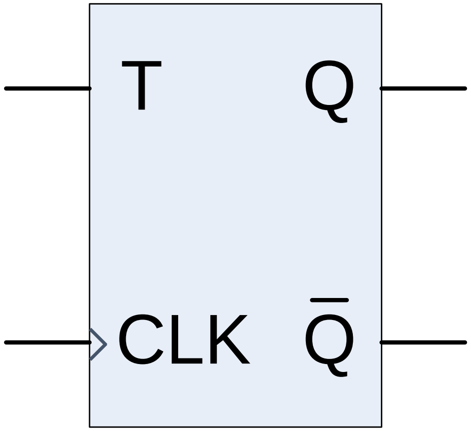
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

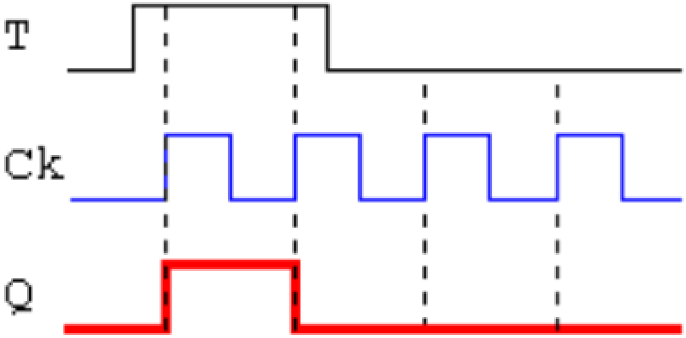
T Flip Flop

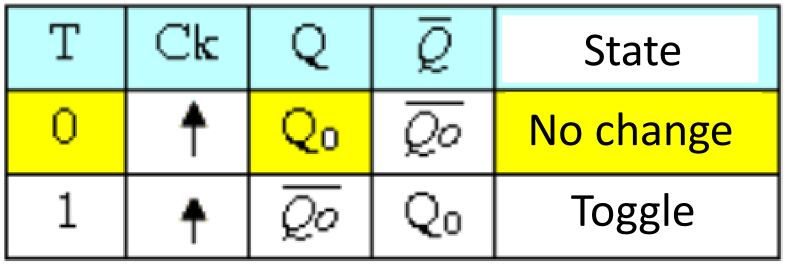
- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

$$Q = \overline{q}T + q\overline{T} = q \oplus T$$

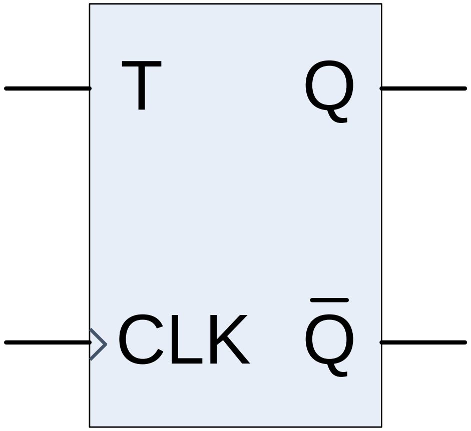




T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

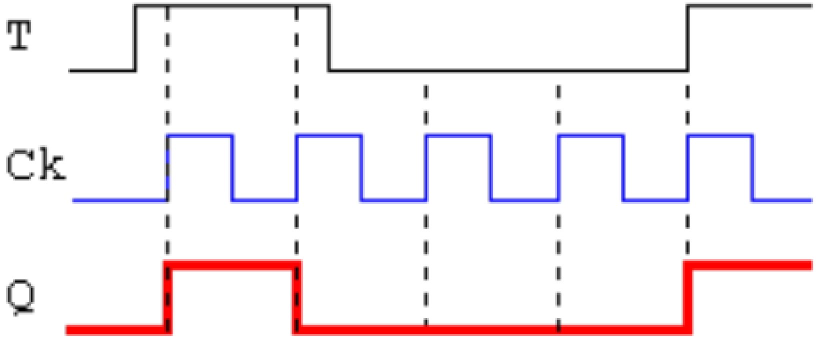
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

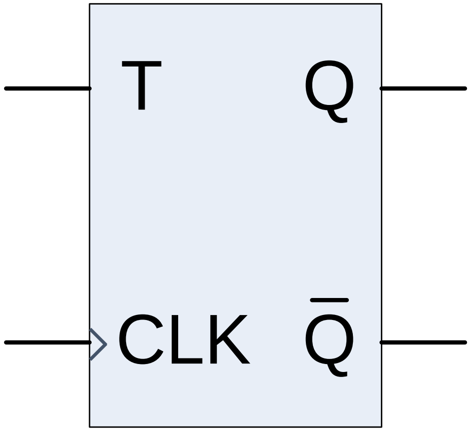
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

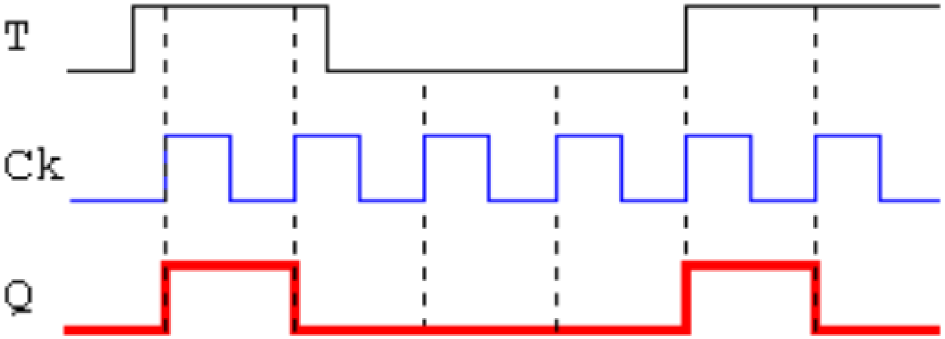
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

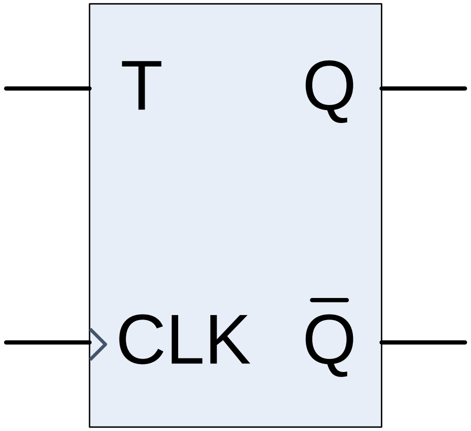
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



T	Ck	Q	$\overline{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

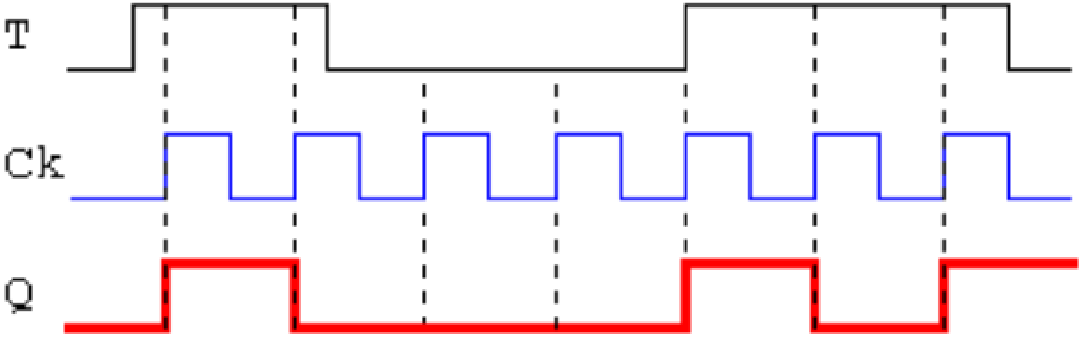
T Flip Flop

- Output waveform:



q \ T	0	1
0	0	1
1	1	0
	nhớ	lật

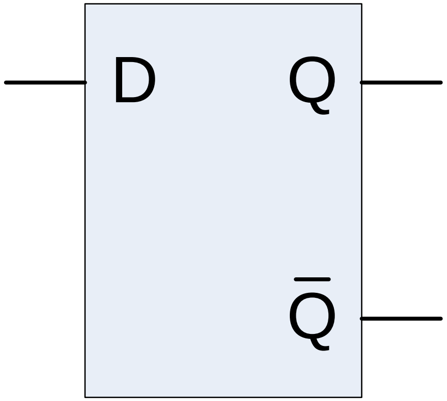
$$Q = \overline{q}T + q\overline{T} = q \oplus T$$



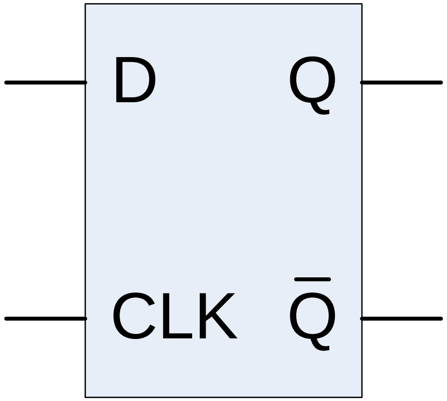
T	Ck	Q	$\bar{Q}$	State
0	↑	Q_0	$\overline{Q_0}$	No change
1	↑	$\overline{Q_0}$	Q_0	Toggle

D Flip Flop

- D FF (or Delay FF) transfers data from the D input to the output when the clock/enable is active.



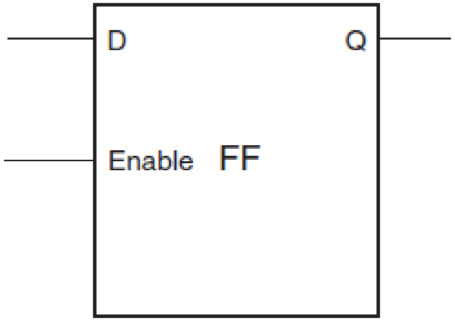
Unclocked D Flip Flop



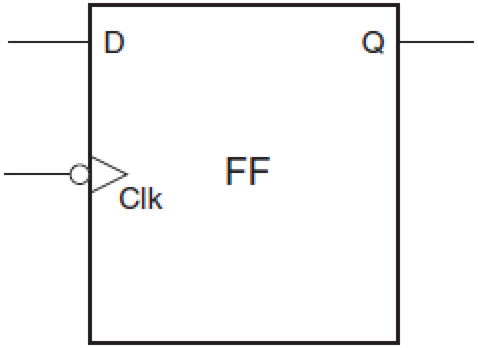
Clocked D Flip Flop

D Latch vs D Flip Flop

- D Latch:

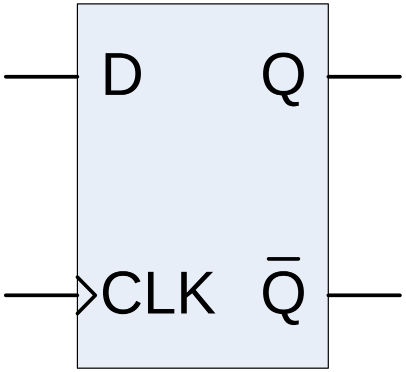


- D Flip Flop (falling-edge trigged):



D Flip Flop

- Output waveform:



Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

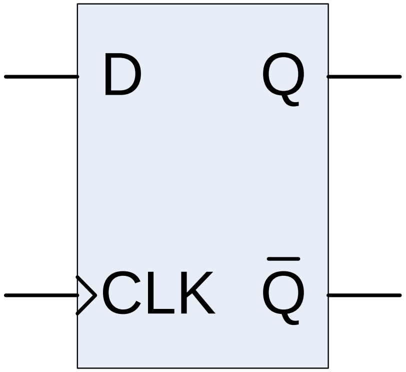
Q_n \ D	0	1
0		1
1		1

$$Q_{n+1} = D$$

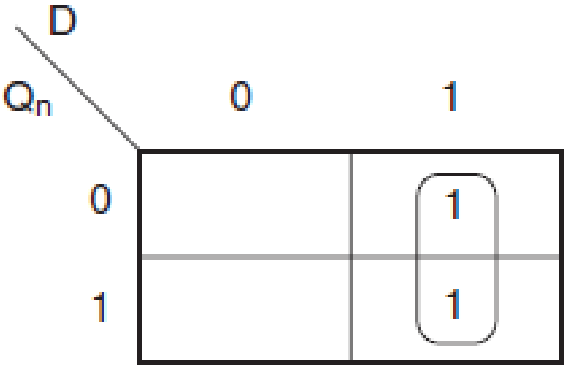
D	Ck	Q	$\overline{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

D Flip Flop

- Output waveform:

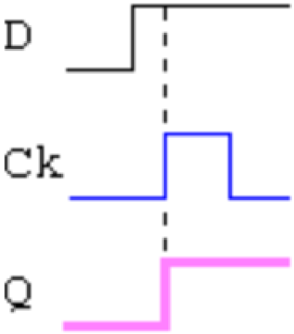


Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1



Q_n \ D	0	1
0		1
1		1

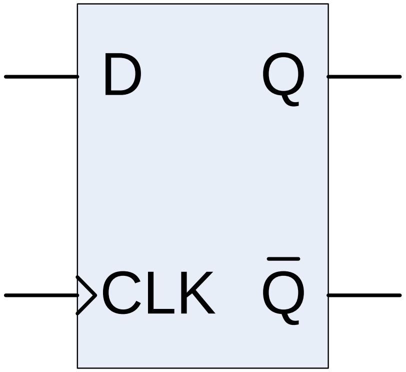
$$Q_{n+1} = D$$



D	Ck	Q	$\overline{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

D Flip Flop

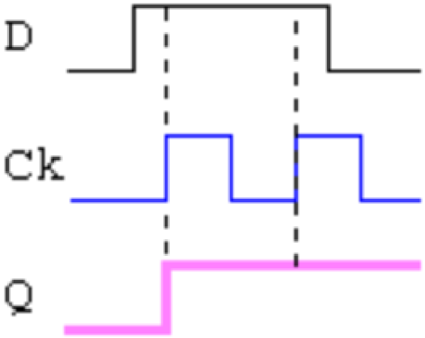
- Output waveform:



Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Q_n \ D	0	1
0		1
1		1

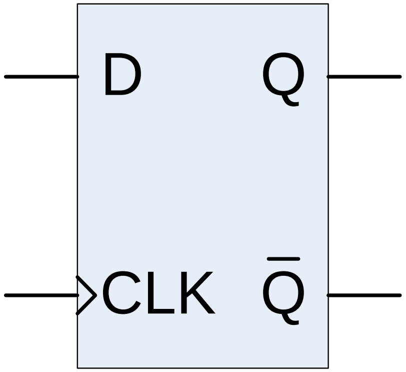
$$Q_{n+1} = D$$



D	Ck	Q	$\overline{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

D Flip Flop

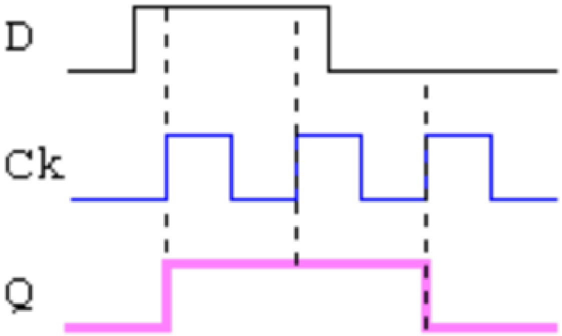
- Output waveform:



Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Q_n \ D	0	1
0		1
1		1

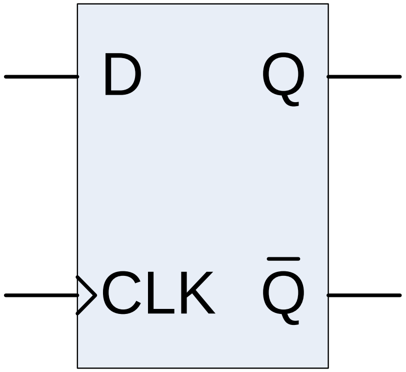
$$Q_{n+1} = D$$



D	Ck	Q	$\bar{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

D Flip Flop

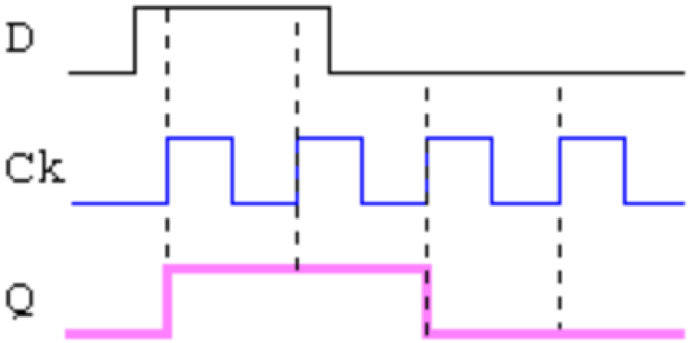
- Output waveform:



Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Q_n \ D	0	1
0		1
1		1

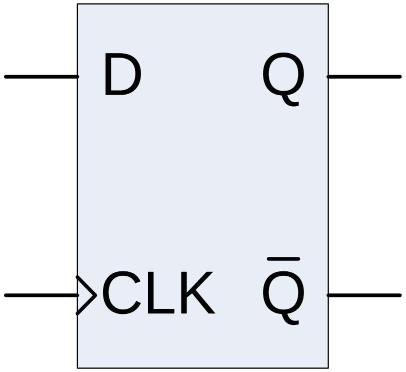
$$Q_{n+1} = D$$

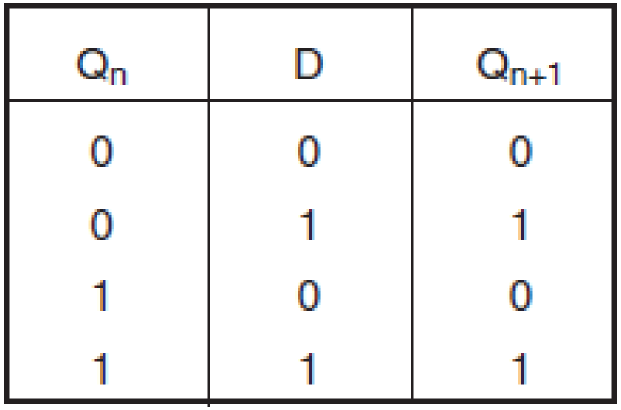


D	Ck	Q	$\bar{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

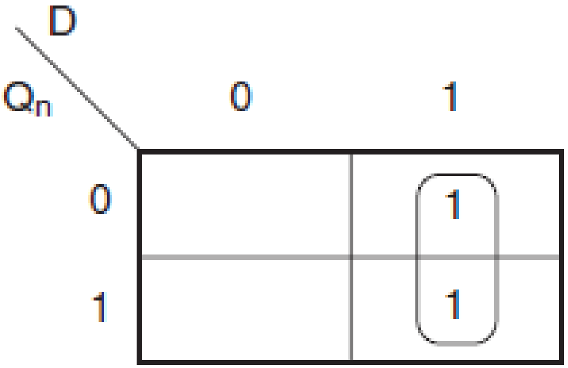
D Flip Flop

- Output waveform:



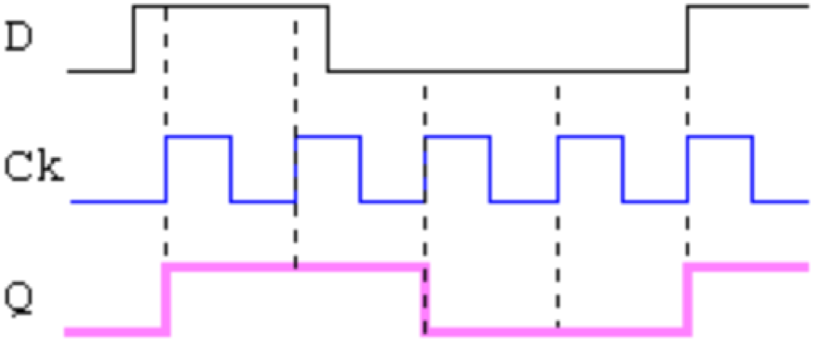


Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1



Q_n \ D	0	1
0		1
1		1

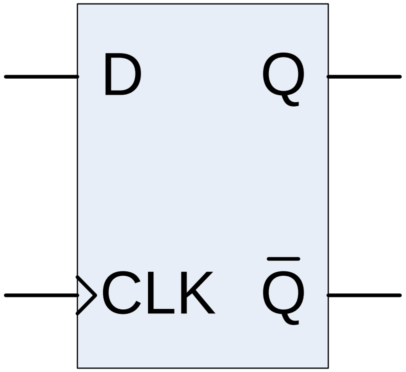
$$Q_{n+1} = D$$



D	Ck	Q	$\overline{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

D Flip Flop

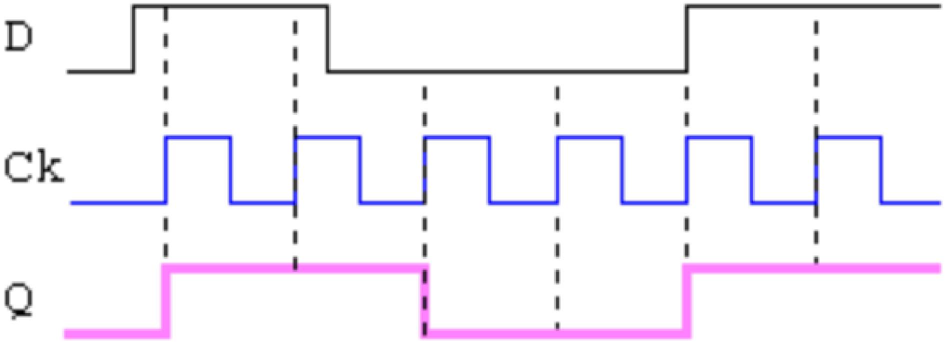
- Output waveform:



Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Q_n \ D	0	1
0		1
1		1

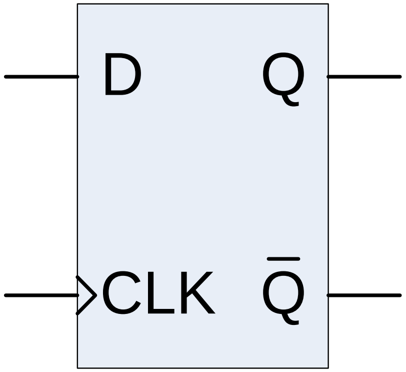
$$Q_{n+1} = D$$



D	Ck	Q	$\bar{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

D Flip Flop

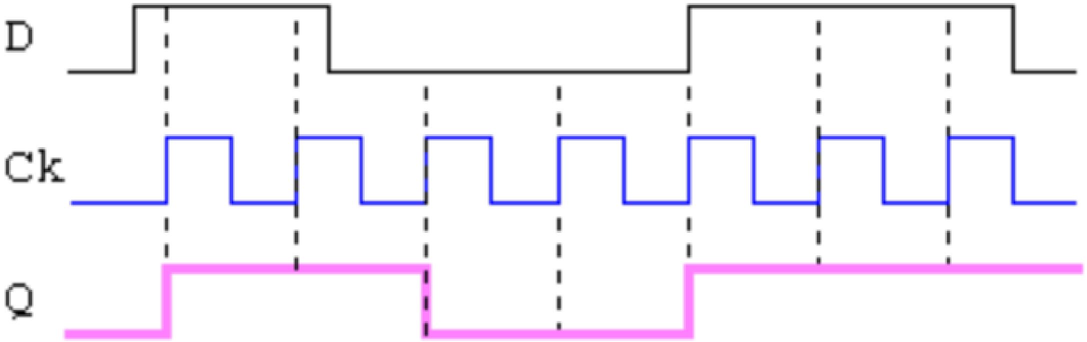
- Output waveform:



Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Q_n \ D	0	1
0		1
1		1

$$Q_{n+1} = D$$



D	Ck	Q	$\bar{Q}$	State
0	↑	0	1	Reset
1	↑	1	0	Set

Flip Flop Excitations

- The excitation table describes the inputs required to achieve a desired output state change.
- A flip flop has at most 4 responses: ‘S0’, ‘S1’, ‘T0’, and ‘T1’

Response		Excitation					
Symbol	**q → Q^+**	**S**	**R**	**J**	**K**	**T**	**D**
S0	**0 → 0**	**0**	**x**	**0**	**x**	**0**	**0**
T1	**0 → 1**	**1**	**0**	**1**	**x**	**1**	**1**
T0	**1 → 0**	**0**	**1**	**x**	**1**	**1**	**0**
S1	**1 → 1**	**x**	**0**	**x**	**0**	**0**	**1**

RS flip flop

CLK	S	R	Q	Q’
‘0’	x	x	q	q’
‘1’	0	0	q	q’
	0	1	0	1
	1	0	1	0
	1	1	x	x

T flip flop

q	T	Q	Q’
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

JK flip flop

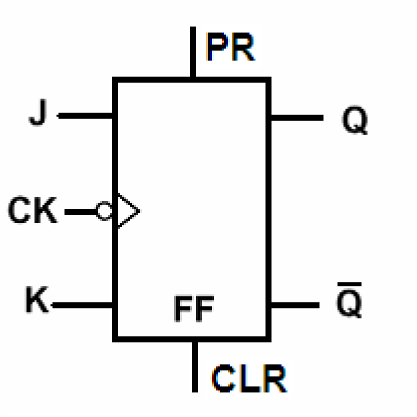
CLK	J	K	Q	Q’
‘0’	x	x	q	q’
‘1’	0	0	q	q’
	0	1	0	1
	1	0	1	0
	1	1	q’	q

D flip flop

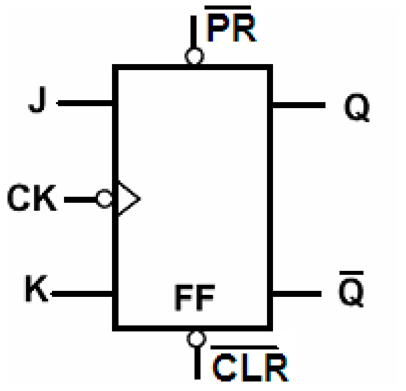
q	D	Q	Q’
0	0	0	1
0	1	1	0
1	0	0	1
1	1	1	0

Flip Flop with Preset and Clear Inputs

- A flip flop often has:
  - Inputs, e.g. J, K
  - Clock
  - Q output
- It is often necessary to clear or preset a flip flop:
  - Clear (CLR) → Q = 0
  - Preset (PR) → Q = 1

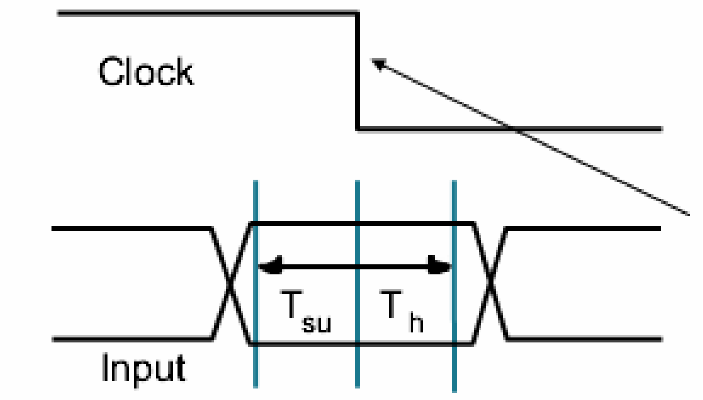


Clear and Preset are active HIGH



Clear and Preset are active LOW

Setup and Hold Time of Flip Flop

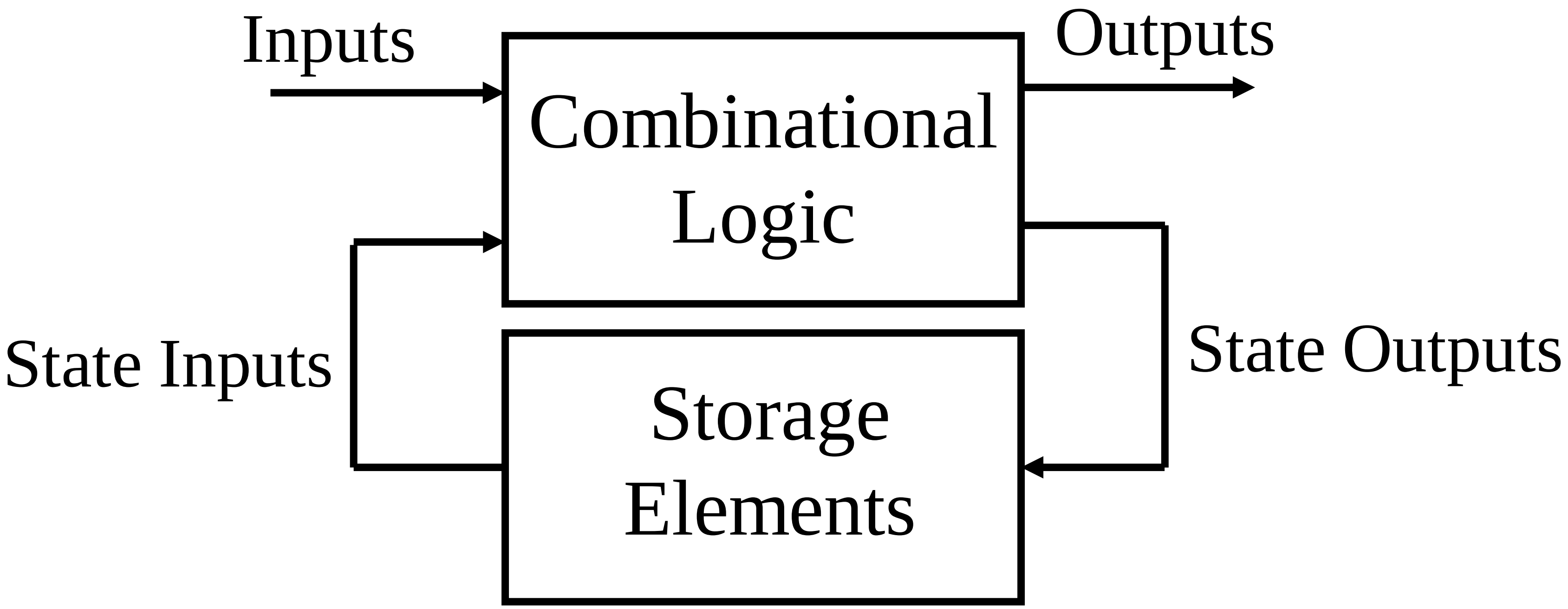


- t_{su}: Setup Time – the amount of time required for the input signal to be stable before a clock edge
- t_h: Hold Time – the minimum amount of time required for the input signal to be stable after a clock edge.
- Setup and hold time relate to propagation delay and clock frequency.

Contents

1. Concepts
2. Flip Flop
3. Flip Flop Types
4. Finite State Machine (FSM)
5. Applications

Models for Representing Sequential Logic



Forms of Sequential Logic

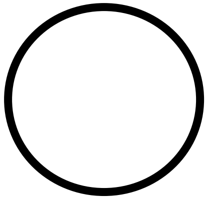
- **Asynchronous sequential logic**: state changes occur whenever input states change
  - Storage elements may be simple wires or delay elements
- **Synchronous sequential logic**: state changes occur in lock step across all storage elements
  - Using a periodic waveform - the clock

Finite State Machine (FSM)

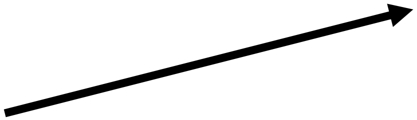
- A Finite State Machine is an abstract mathematical model of a sequential logic function.
  - A finite number of states, including one initial state (or start state)
  - A finite number of inputs
  - A finite number of outputs
  - A transition function – a function of its current state and its input
  - An output function – a function of its current state and its input (optional)

FSM Representations

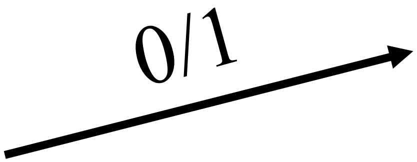
- **States**: determined by possible values in sequential storage elements



- **Transitions**: change of state

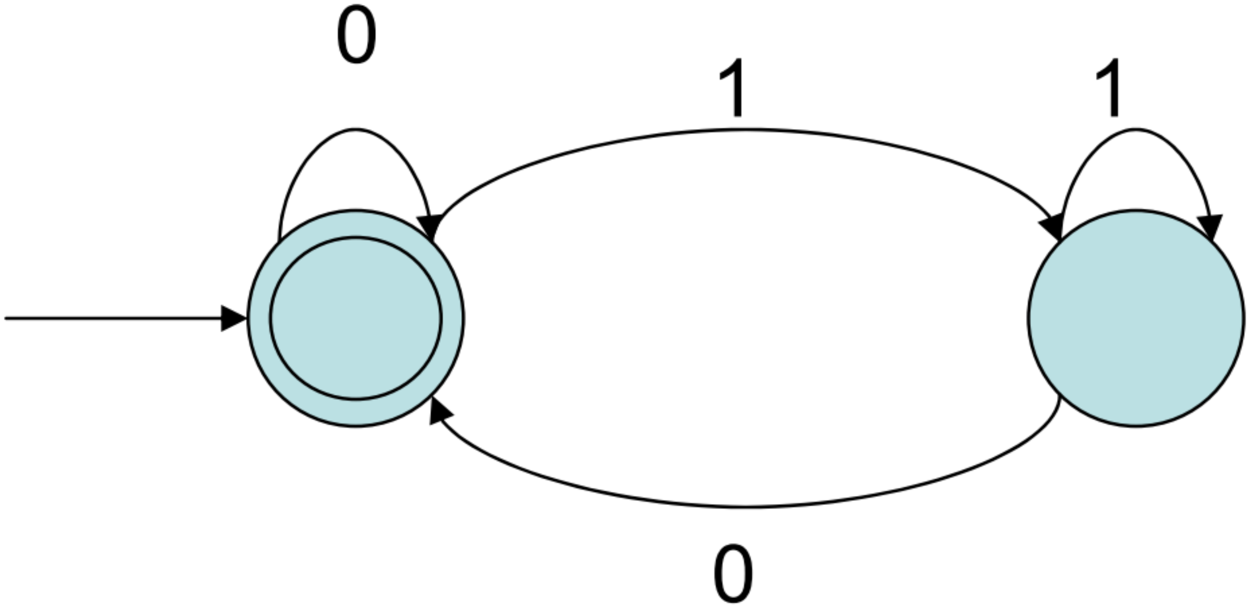


- **Inputs**:



Example 3

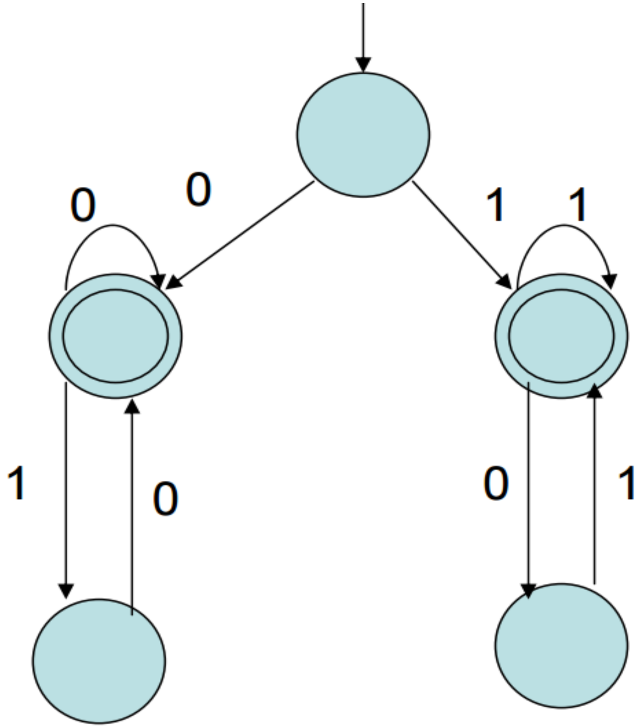
- What does this FSM do?



→ Accepts the empty string or any string that ends with ‘0’.

Example 4

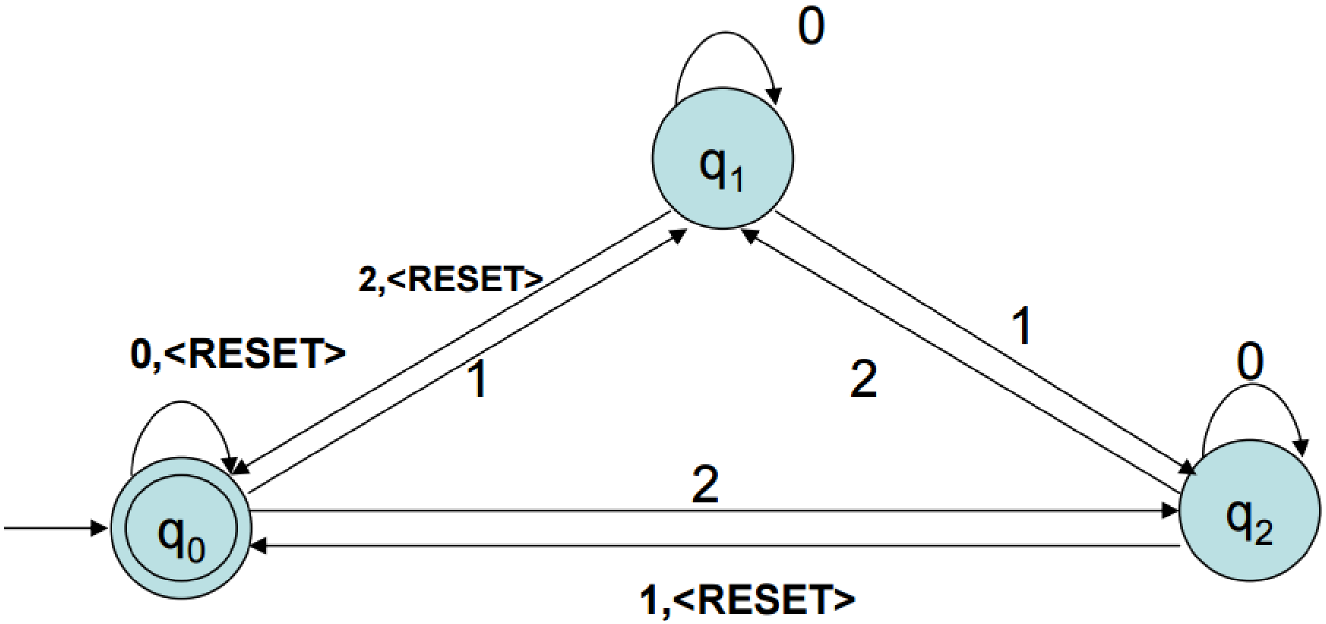
- What does this FSM do?



→ Accepts strings that starts and ends with the same bits.

Example 5

- What does this FSM do?

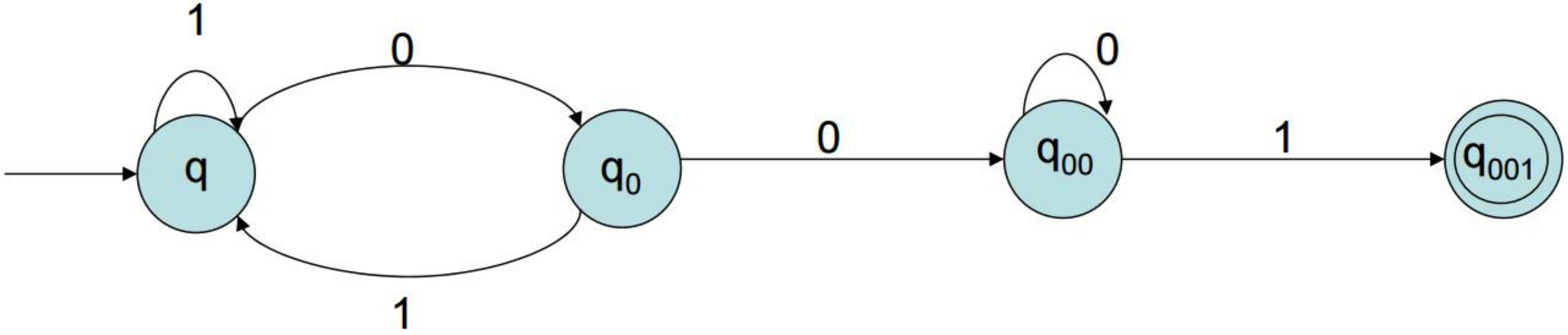


→ Accepts if the running sum of the input strings is a multiple of 3.

→ RESET symbol resets the running sum to 0.

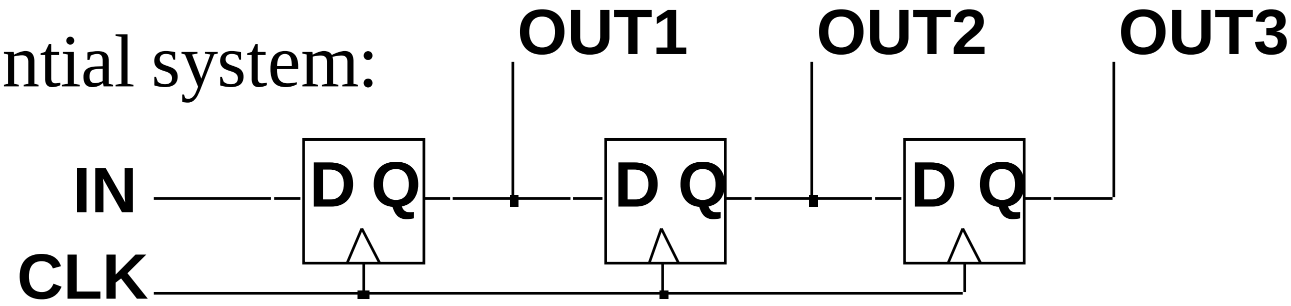
Example 6

- Design an FSM which accepts strings that contain 001 as substrings.
- There are 4 possibilities:
  - No string
  - Seen a 0
  - Seen a 00
  - Seen a 001

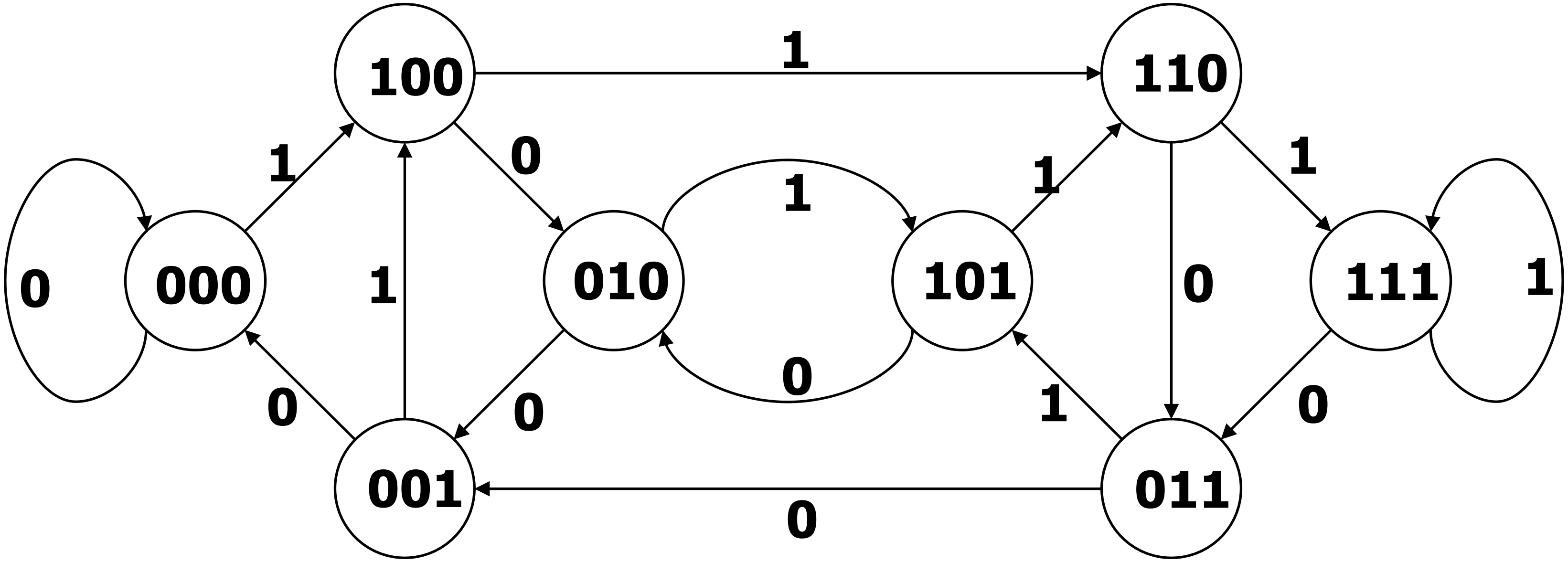


Example 7

- For a sequential system:

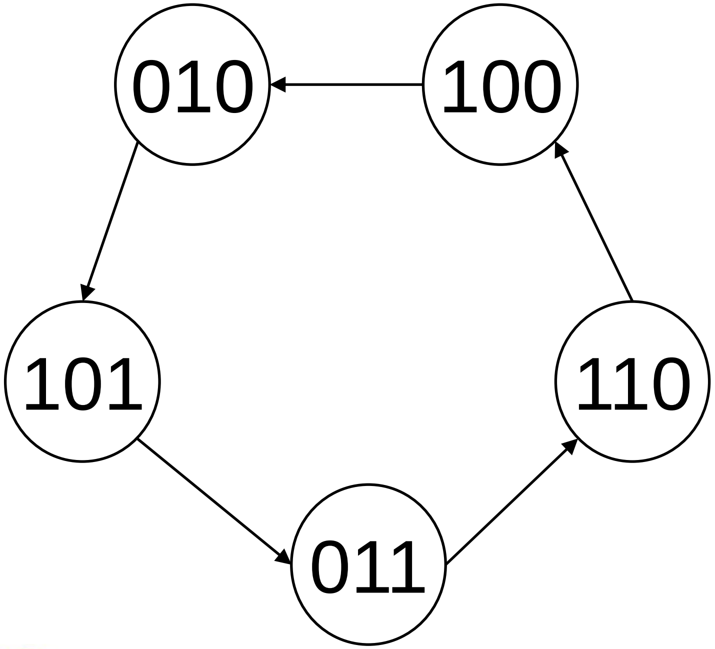


- Sketch the FSM of the above system:



Example 8

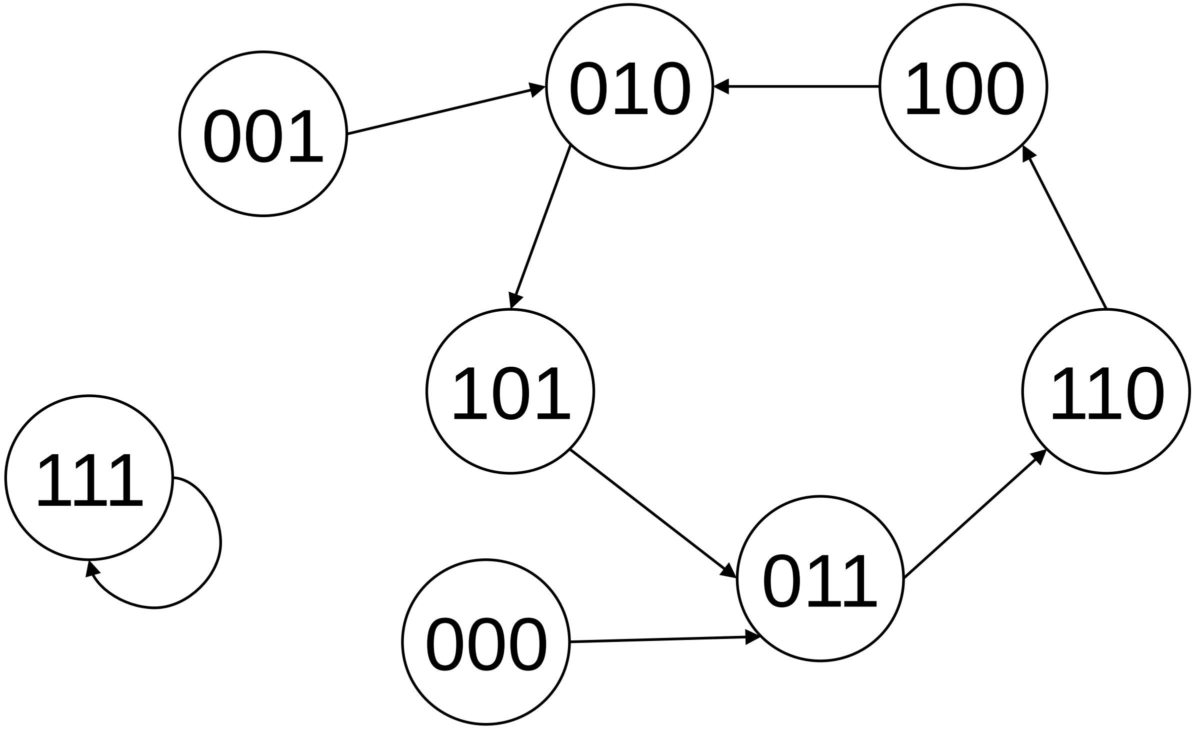
- Design a counter which is capable of counting: 100, 010, 101, 011, 110
  - 000, 001 and 111 are skipped.
- Sketch the FSM and write State Transition Table:



Present State			Next State		
C	B	A	C+	B+	A+
0	0	0	x	x	x
0	0	1	x	x	x
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	x	x	x

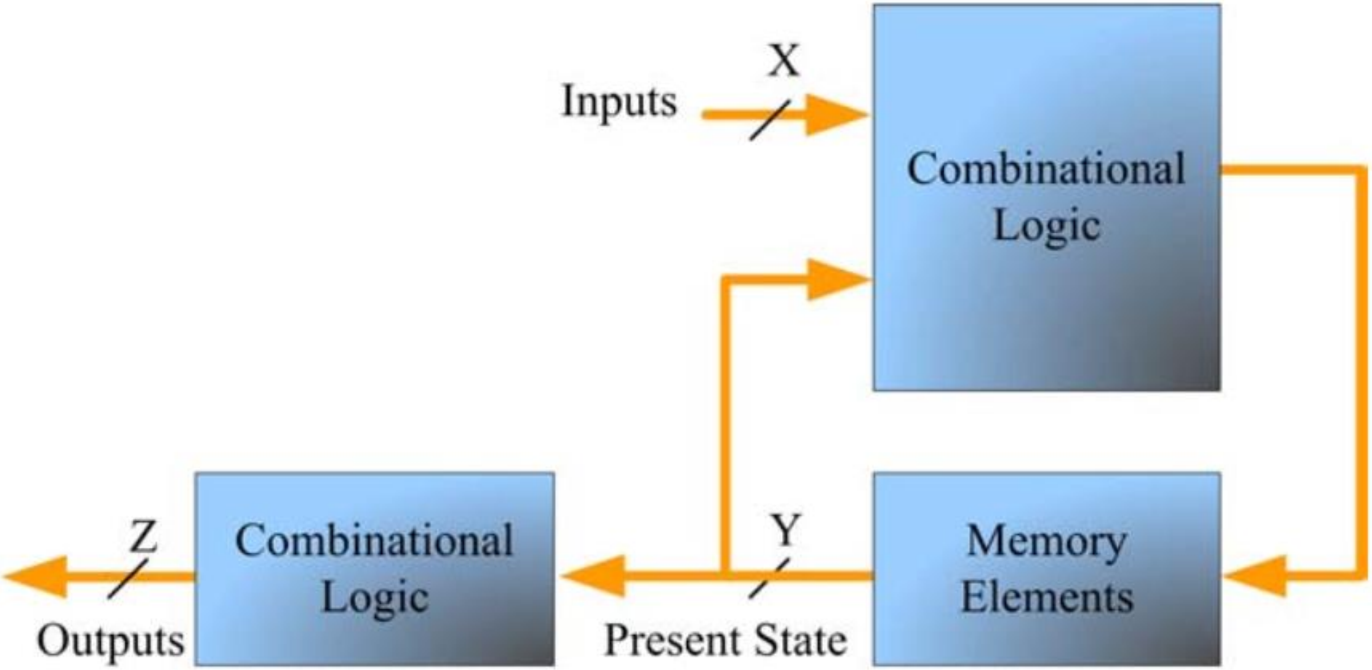
Example 8

- At power-up, the counter may be in an unused or invalid state → Must guarantee that it eventually enters a valid state.



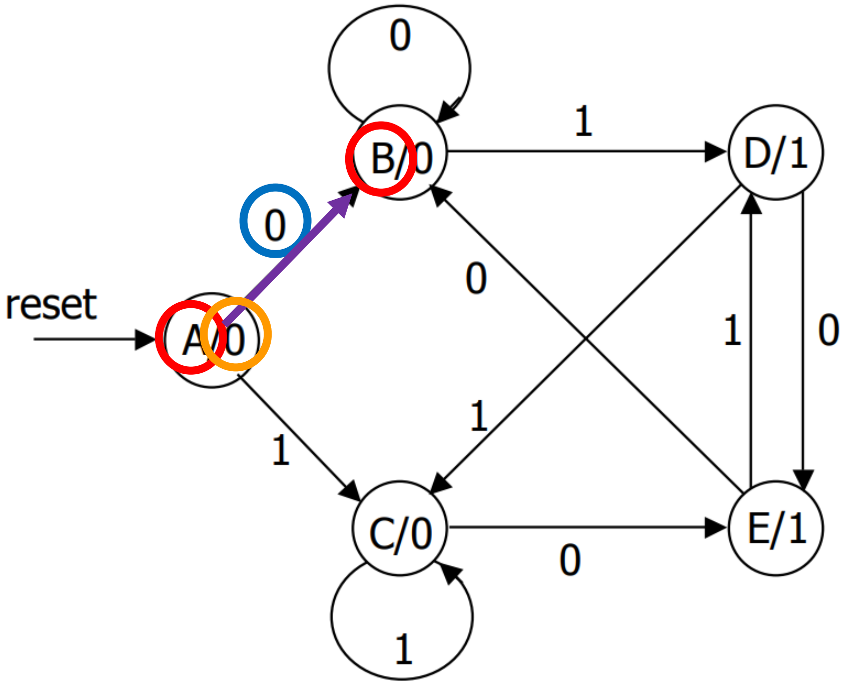
Moore FSM

- Its output is affected by only its current state, not its input.



Moore FSM Representation

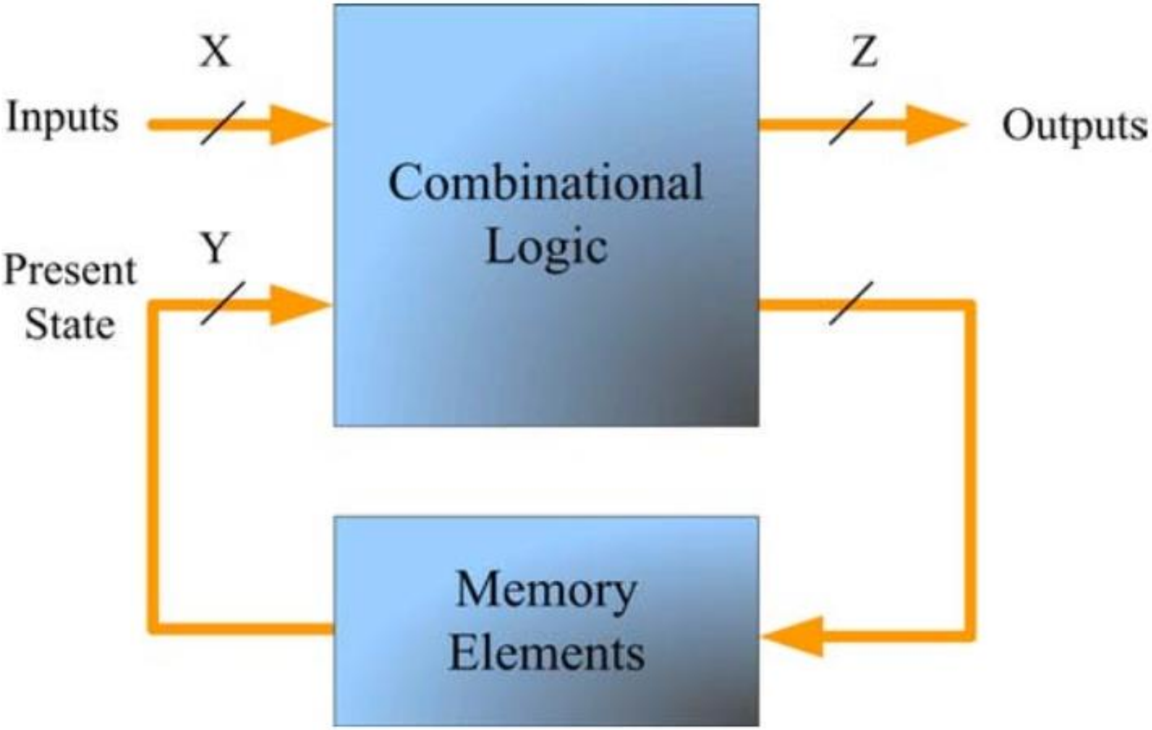
- Output is a function of the current state.



		current	next	
reset	input	state	state	output
1	–	–	A	
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	D	0
0	0	C	E	0
0	1	C	C	0
0	0	D	E	1
0	1	D	C	1
0	0	E	B	1
0	1	E	D	1

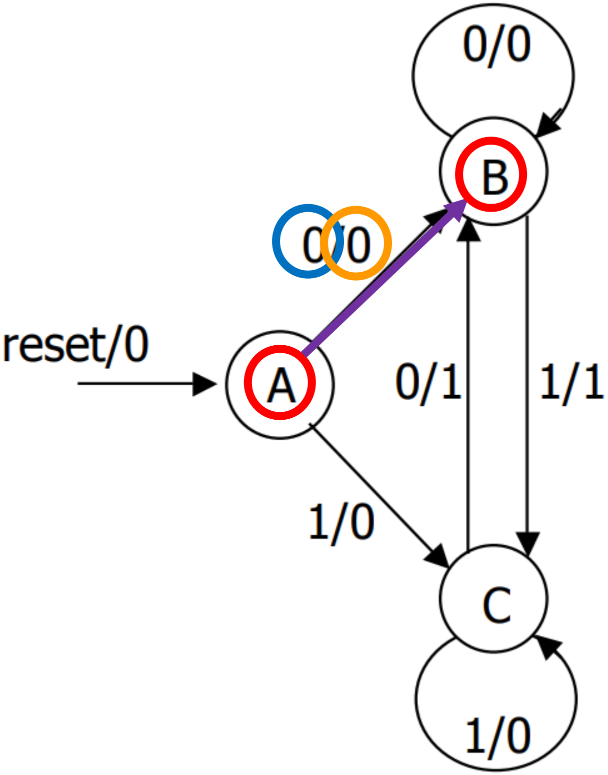
Mealy FSM

- Its output is affected by both its current state and input.



Mealy FSM Representation

- Output is a function of the current state and input.



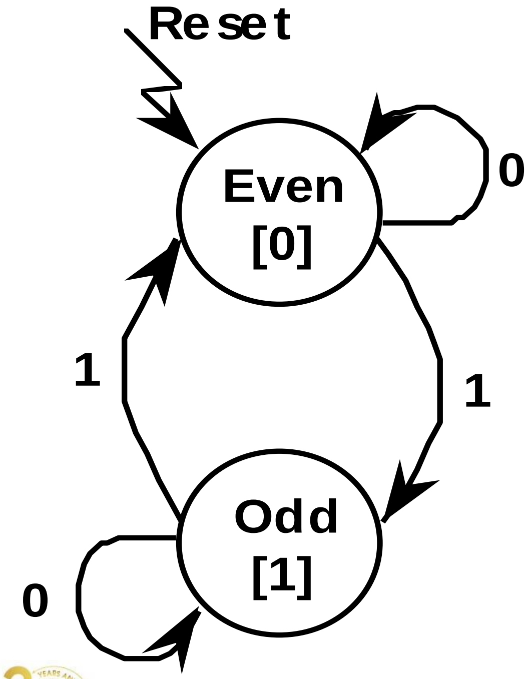
		current	next	
reset	input	state	state	output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0

Design Procedure

1. Describe the sequential circuit with simple but clear words
2. Derive the State Diagram
3. Encode the states and inputs (if any)
4. Fill the State Table (or Excitation Table)
5. Select the necessary number of flip-flops and the type of flip-flops
6. Determine the Boolean functions that produce the inputs of the flip flops and the output
7. Minimize the input/output functions
8. Design the circuit

Example 9

- Design a FSM which accepts 0,1 strings which has an odd number of 1's.
- It is required to remember whether there are odd 1's so far or even 1's so far.



Current State	Input	Next State	Output
Even	0	Even	0
Even	1	Odd	1
Odd	0	Odd	1
Odd	1	Even	0

Example 9

- Encode the states and inputs:

Current State	Input	Next State	Output
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

- Select the necessary number of flip flops:
  - 1 flip flop to hold 2 states

Example 9

- Select the type of flip flops: D flip flop
- Assume:
  - Q = Current state
  - Q^+ = Next state
  - X = Input

Response		Excitation					
Symbol	$Q \rightarrow Q^+$	S	R	J	K	T	D
S0	$0 \rightarrow 0$	0	x	0	x	0	0
T1	$0 \rightarrow 1$	1	0	1	x	1	1
T0	$1 \rightarrow 0$	0	1	x	1	1	0
S1	$1 \rightarrow 1$	x	0	x	0	0	1

Q	X	Q^+	Output	D
0	0	0	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	0	0

D = Q+

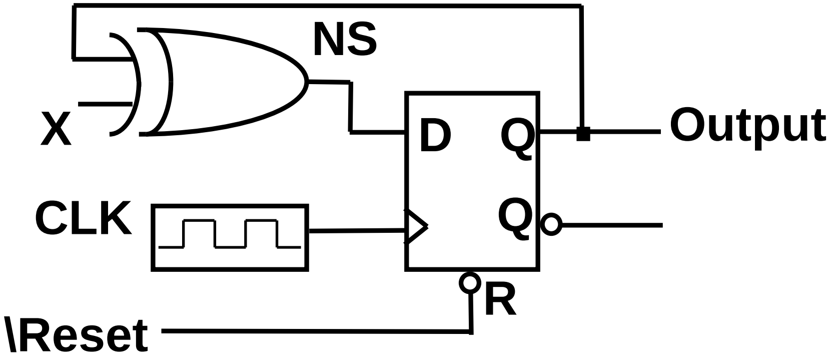
Example 9

- Logic simplification:

D X Q	0	1
0	0	1
1	1	0

$$D = \overline{Q}X + Q\overline{X} = Q \oplus X$$

- Logic circuit diagram:

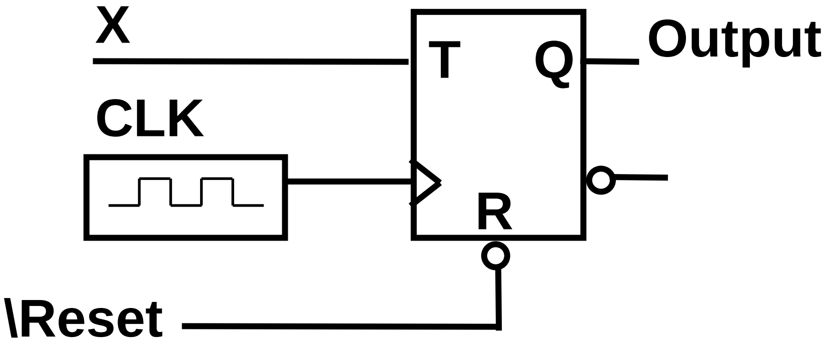


Example 9

- Select the type of flip flops: T flip flop
- Assume:
  - Q = Current state
  - Q^+ = Next state
  - X = Input
- Logic circuit diagram:

Q	X	Q^+	Output	T
0	0	0	0	0
0	1	1	1	1
1	0	1	1	0
1	1	0	0	1

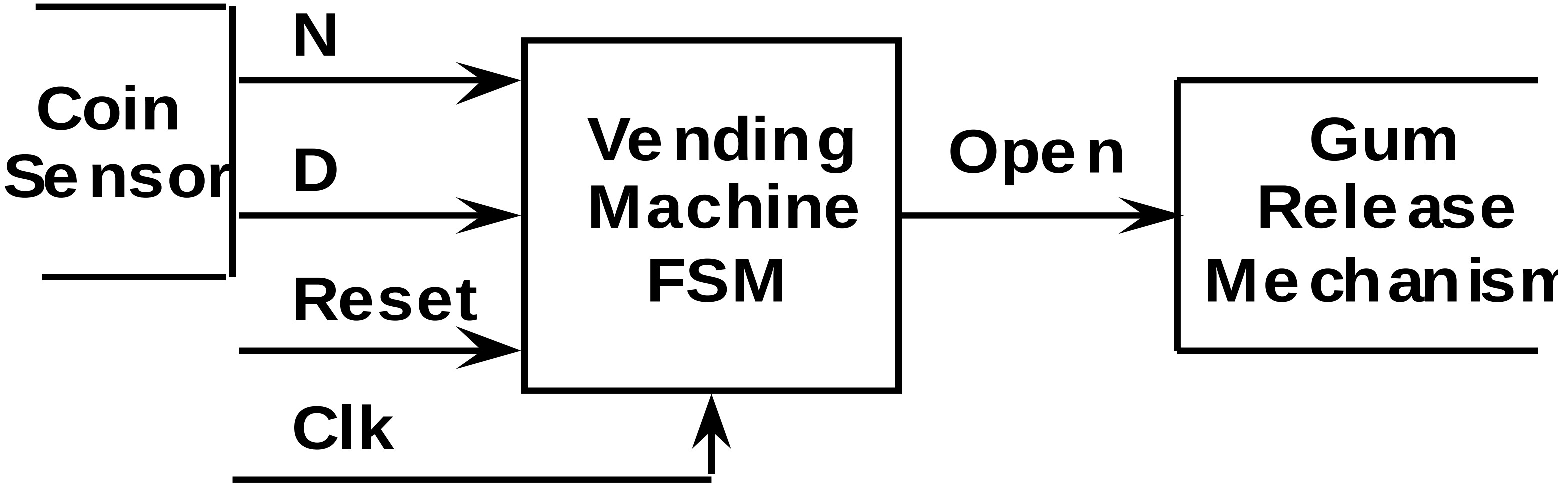
T = X



Example 10

- Design a vending machine:
  - Release item after 15 cents are deposited
  - Single coin slot for dimes (= 10 cents), nickels (= 5 cents)
  - No change

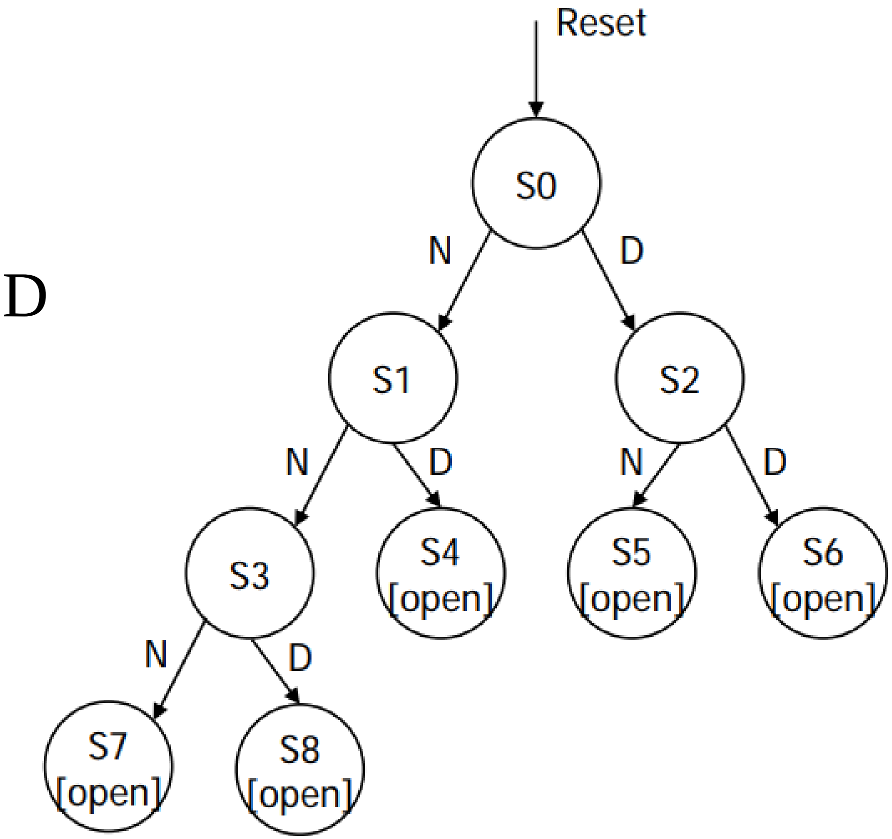
Example 10 – Step 1: Abstract Representation



- Input: **N**, **D**, **Reset**
- Output: **Open**

Example 10 – Step 2: Draw State Diagram/Table

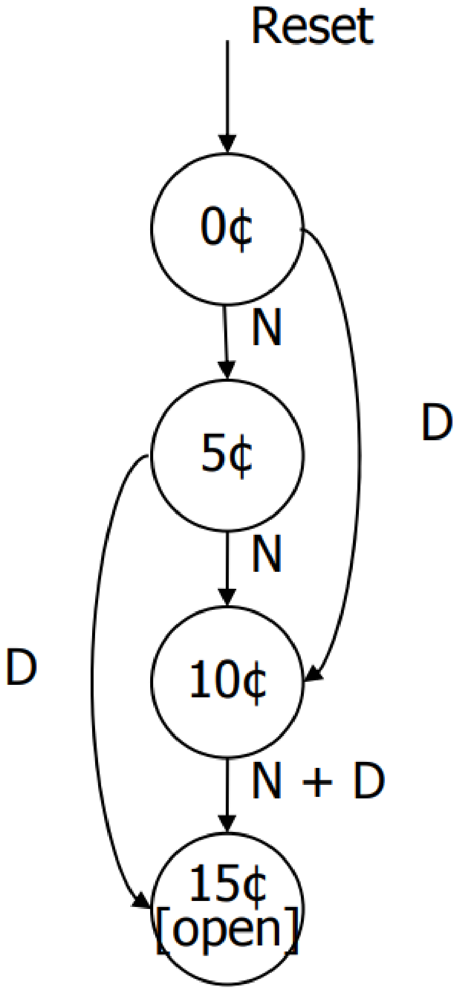
- State Diagram:
  - 3 nickels (= 5 cents): N, N, N
  - 1 nickel, 1 dime: N, D
  - 1 dime, 1 nickel: D, N
  - 2 dimes: D, D
  - 2 nickels, 1 dime: N, N, D



Example 10 – Step 2: Draw State Diagram/Table

- Moore machine

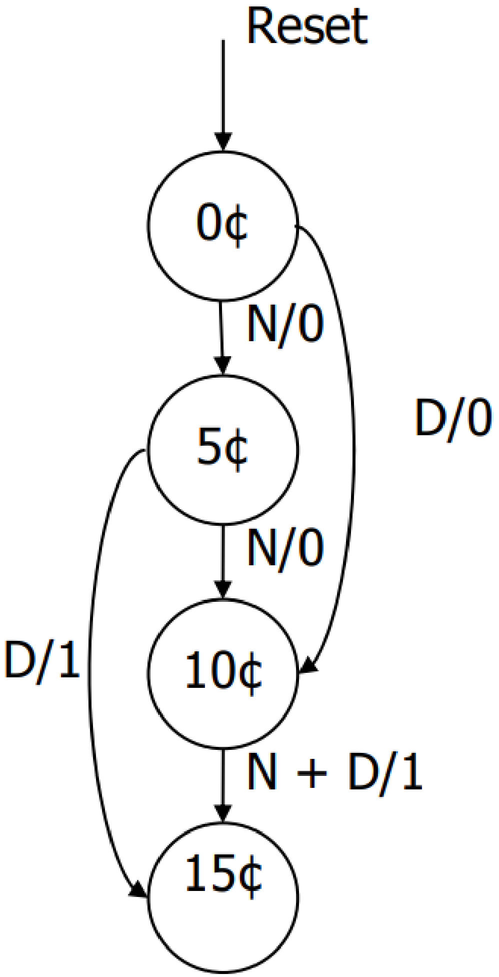
present state	inputs D	N	next state	present output
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	–	–
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	–	–
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	–	–
15¢	–	–	15¢	1



Example 10 – Step 2: Draw State Diagram/Table

- Mealy machine

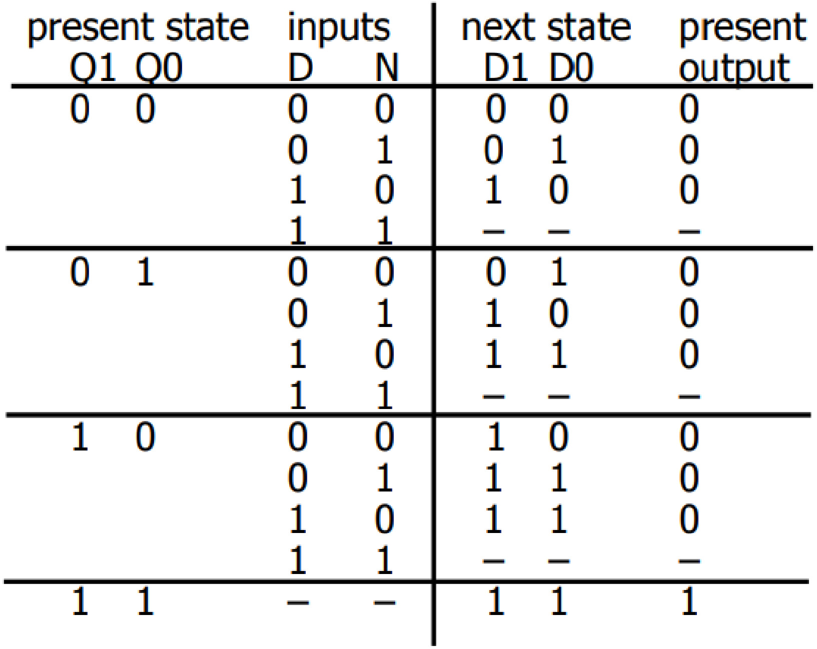
present	inputs		next	present
state	D	N	state	output
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	–	–
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	1
	1	1	–	–
10¢	0	0	10¢	0
	0	1	15¢	1
	1	0	15¢	1
	1	1	–	–
15¢	–	–	15¢	1



Example 10 – Step 3, 4, 5

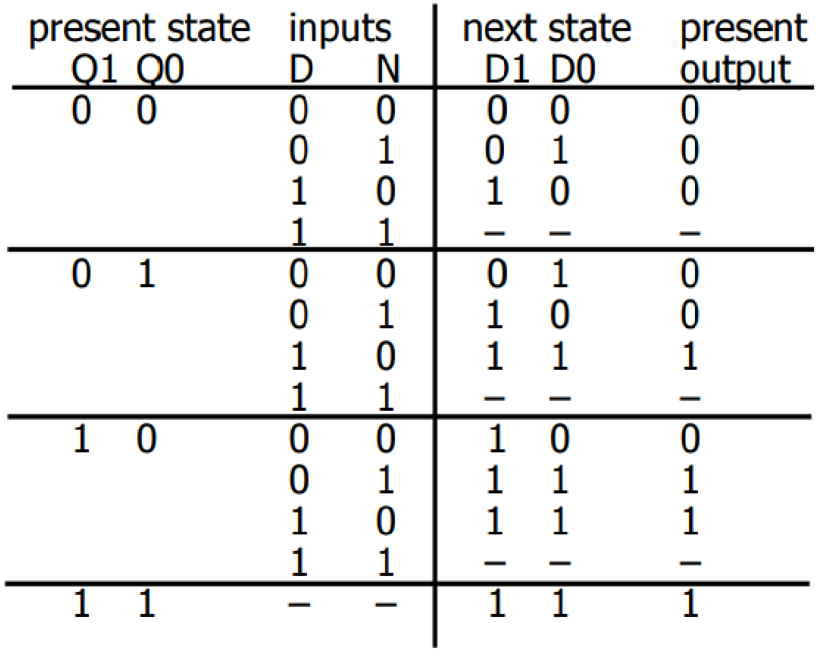
- Encode states/inputs
- Select necessary number/type of flip flops (02 D flip flops)

Moore



present state		inputs		next state		present
Q1	Q0	D	N	D1	D0	output
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	–	–	–
1	1	–	–	1	1	1

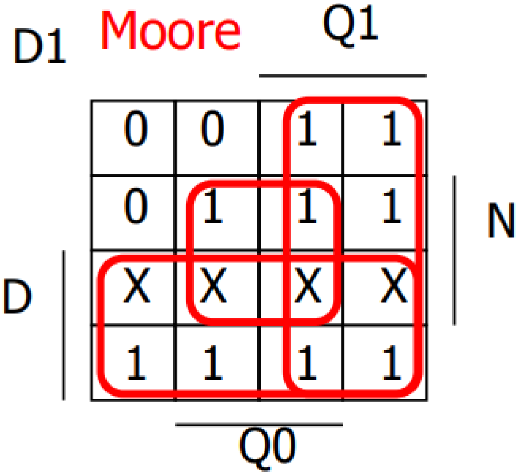
Mealy



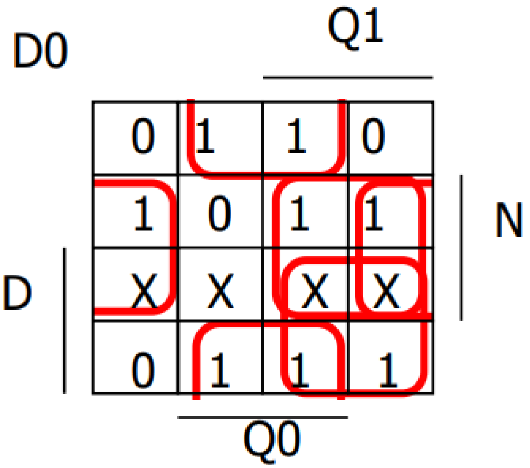
present state		inputs		next state		present
Q1	Q0	D	N	D1	D0	output
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	1
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	1
		1	0	1	1	1
		1	1	–	–	–
1	1	–	–	1	1	1

Example 10 - Step 6, 7

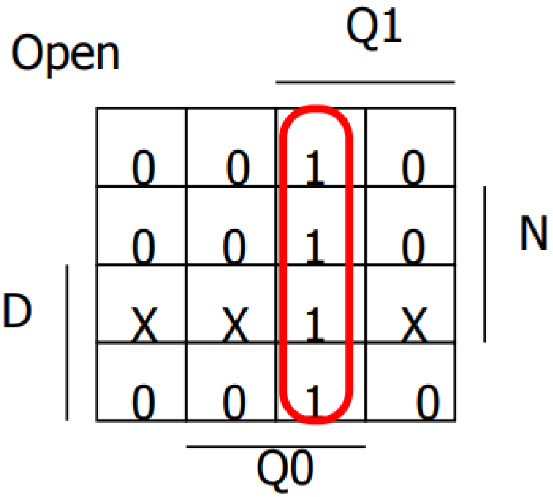
- Logic simplification:



0	0	1	1
0	1	1	1
X	X	X	X
1	1	1	1



0	1	1	0
1	0	1	1
X	X	X	X
0	1	1	1

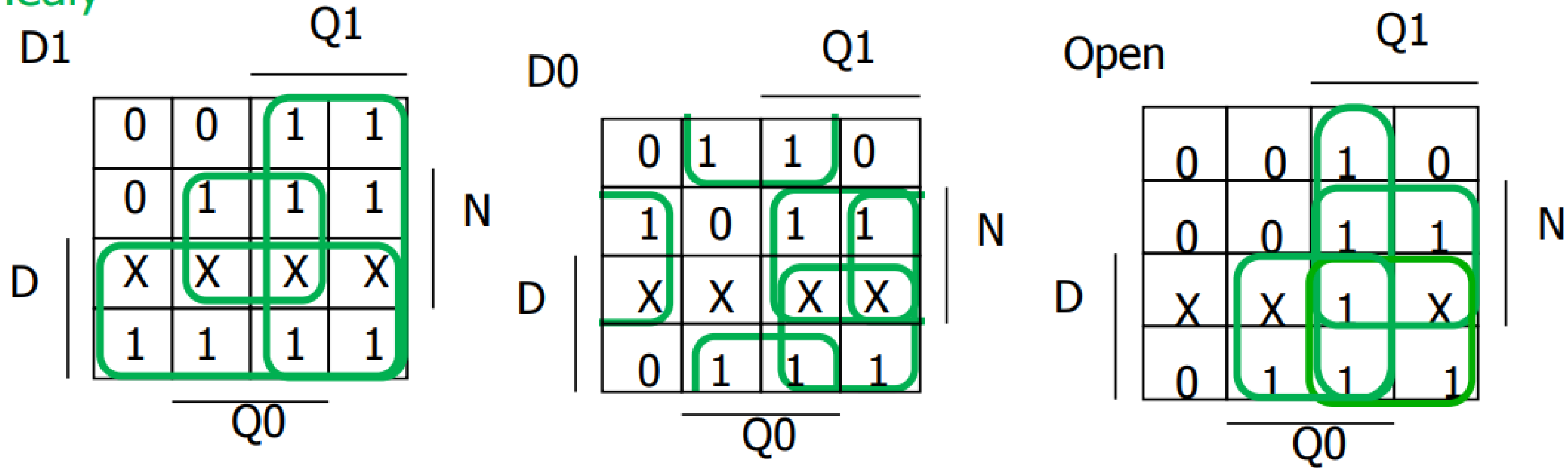


0	0	1	0
0	0	1	0
X	X	1	X
0	0	1	0

Example 10 - Step 6, 7

- Logic simplification:

Mealy



D1

0	0	1	1
0	1	1	1
X	X	X	X
1	1	1	1

D0

0	1	1	0
1	0	1	1
X	X	X	X
0	1	1	1

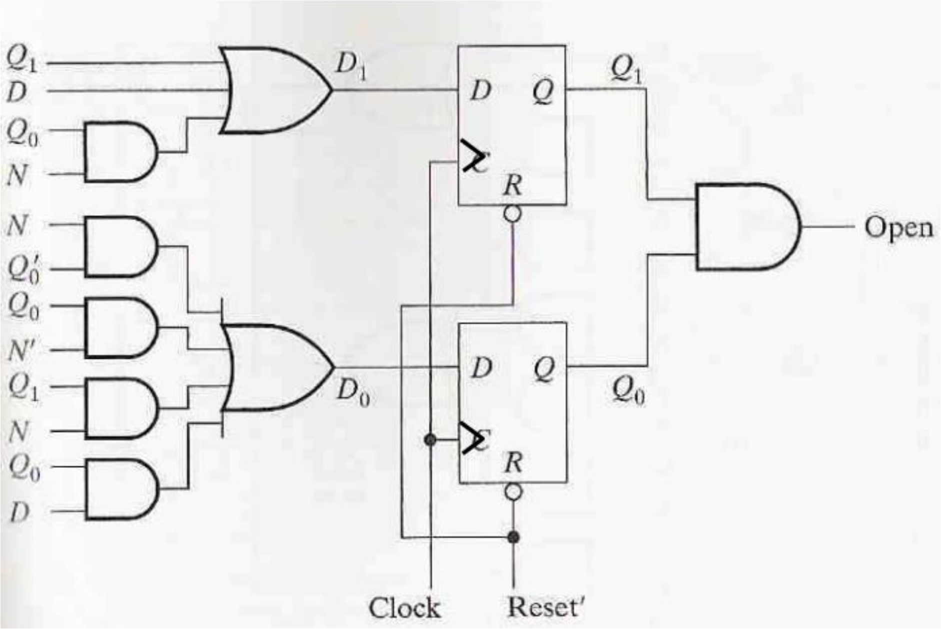
Open

0	0	1	0
0	0	1	1
X	X	1	X
0	1	1	1

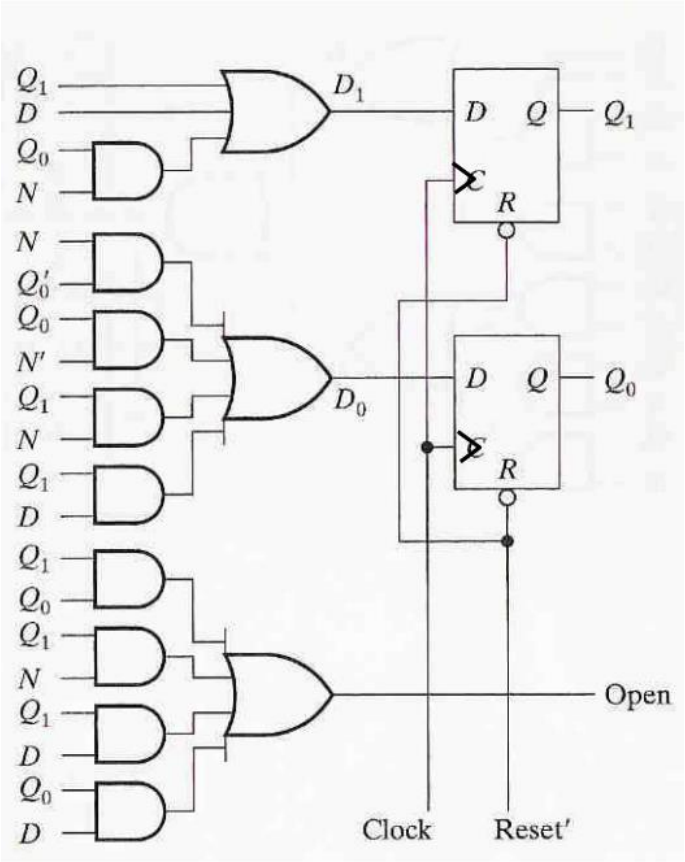
Example 10 – Step 8

- Logic circuit diagram:

Moore



Mealy



Contents

1. Concepts
2. Flip Flop
3. Flip Flop Types
4. Finite State Machine (FSM)
5. Applications

Applications of Sequential Logic

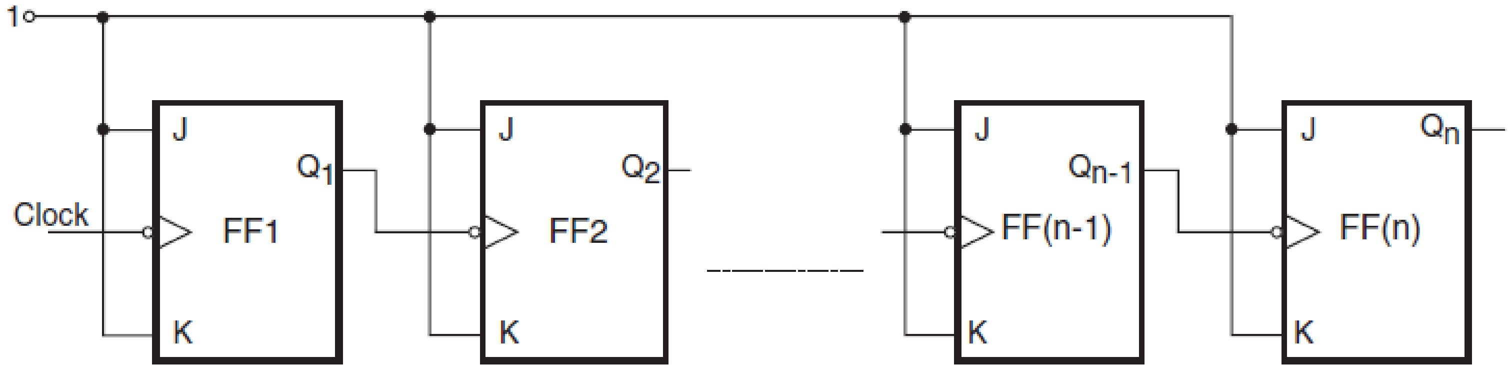
- Counter and Frequency Divider
- Shift Register

Counters

- Counters are mainly used in counting applications.
    - Measure the time interval between two unknown time instants, or
    - Measure the frequency of a given signal.
- 2 types of counters:
    - An asynchronous (or ripple) counter is a cascaded arrangement of flip-flops where **the output of one flip flop drives the clock input of the following flip flop**.
    - A synchronous (or parallel) counter is a cascaded arrangement of flip flops where **all the flip flops in the counter change state at the same time** in synchronism with the input clock signal.

Ripple Counter

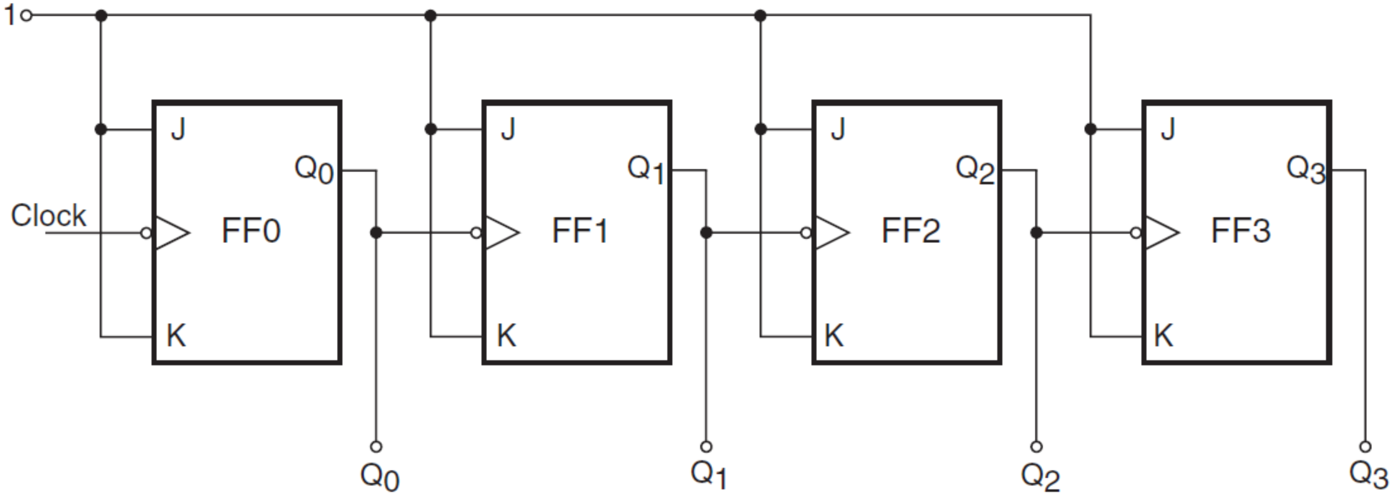
- Block schematic of n-bit binary ripple counter:



- Propogation delay:
  - The effective propagation delay in a ripple counter is equal to the sum of propagation delays due to different flip flops.

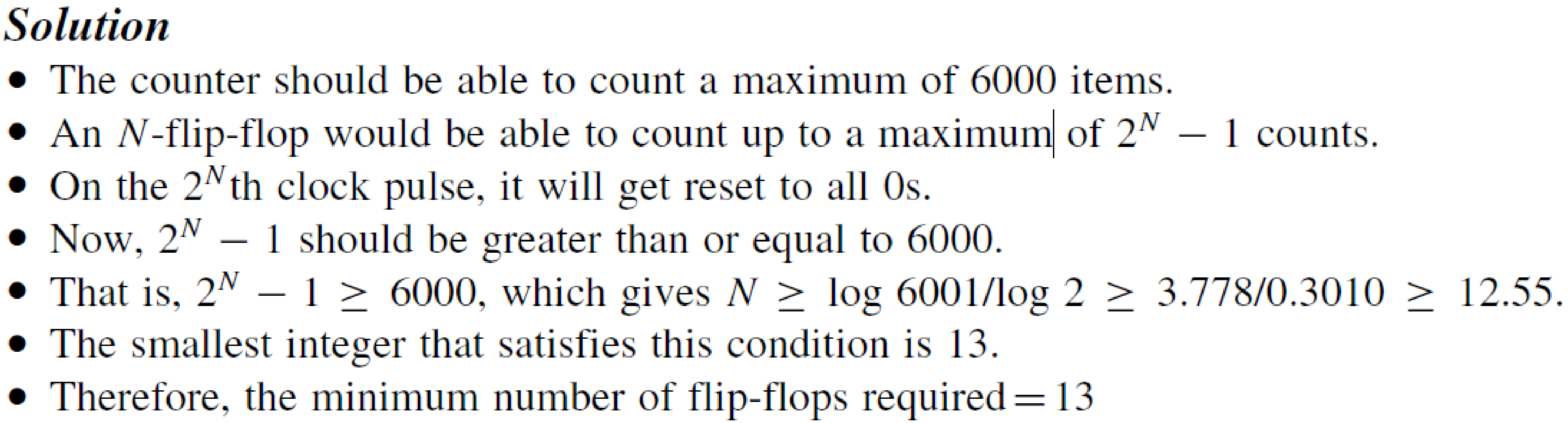
Mod-2^N Ripple Counter

- Mod-16 ripple counter
  - 16 states
  - 4-bit output (Q_3, Q_2, Q_1, Q_0)
  - Needs 4 flip flops (e.g. 4 JK flip flops)



Example 11

- A binary ripple counter is designed to count the number of items passing a vonveyor belt.
- Each time an item passes a given point, a pulse is generated that can be used as a clock input.
- If the maximum number of items to be counted is 6000, determine the number of flip flops required.

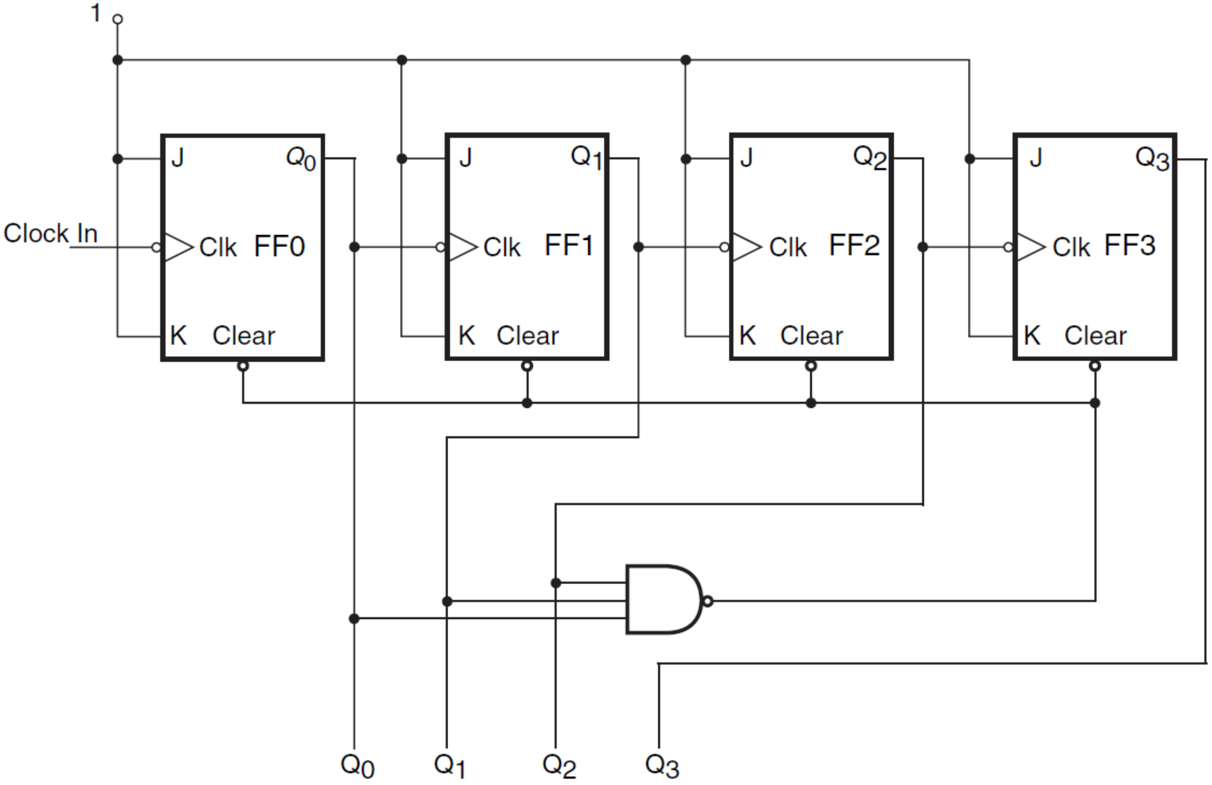


Solution

- The counter should be able to count a maximum of 6000 items.
- An N-flip-flop would be able to count up to a maximum of $2^N - 1$ counts.
- On the 2^Nth clock pulse, it will get reset to all 0s.
- Now, $2^N - 1$ should be greater than or equal to 6000.
- That is, $2^N - 1 \geq 6000$, which gives $N \geq \log 6001/\log 2 \geq 3.778/0.3010 \geq 12.55$.
- The smallest integer that satisfies this condition is 13.
- Therefore, the minimum number of flip-flops required = 13

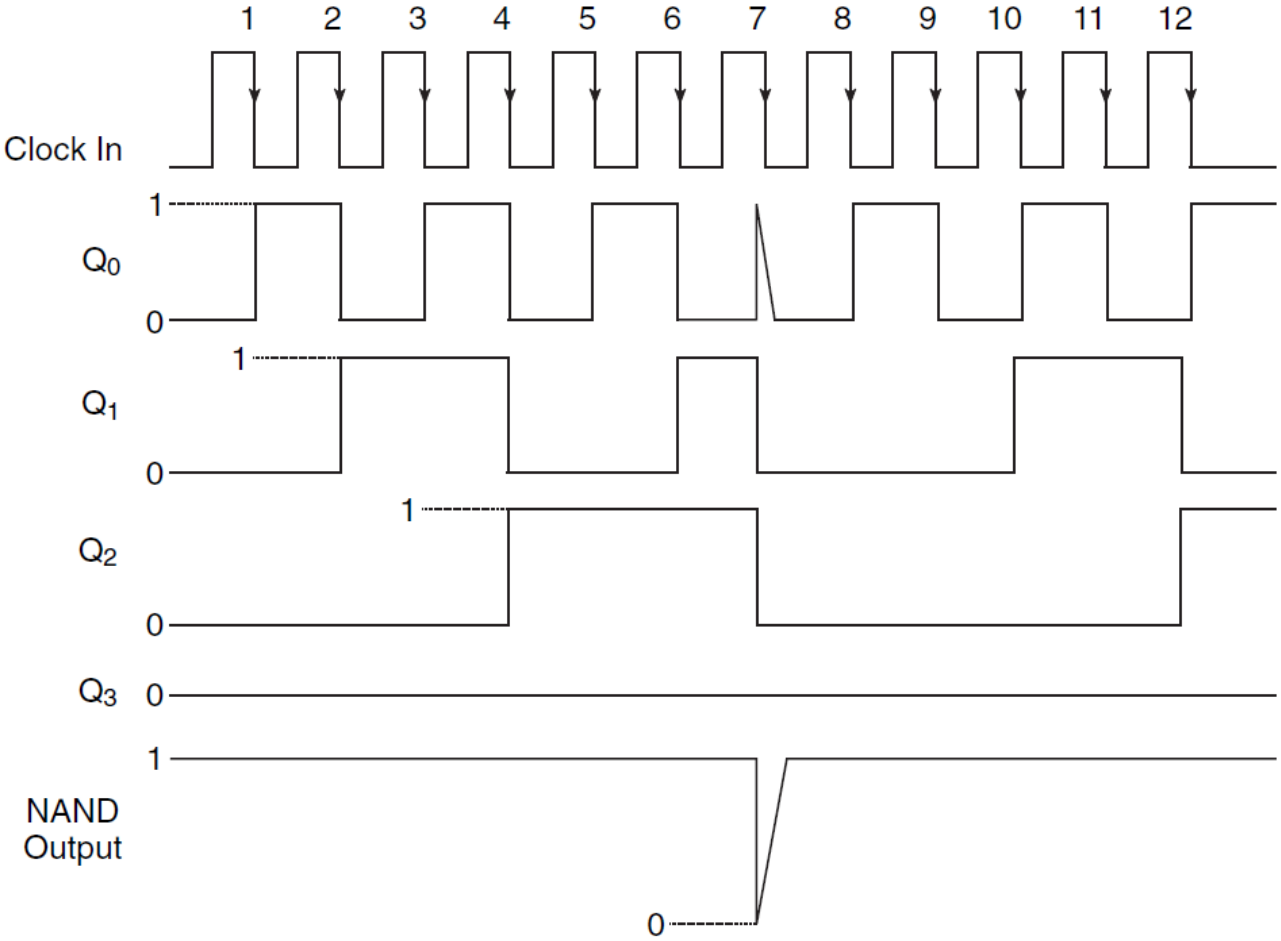
Binary Ripple Counters with a Modulus of < 2^N

- Consider an Mod-7 counter using 4 JK flip flops (CLEAR is active LOW).
- When the counter reaches 0110 (6_{10}), all flip flops are reset.



Binary Ripple Counters with a Modulus of < 2^N

- Output waveform:



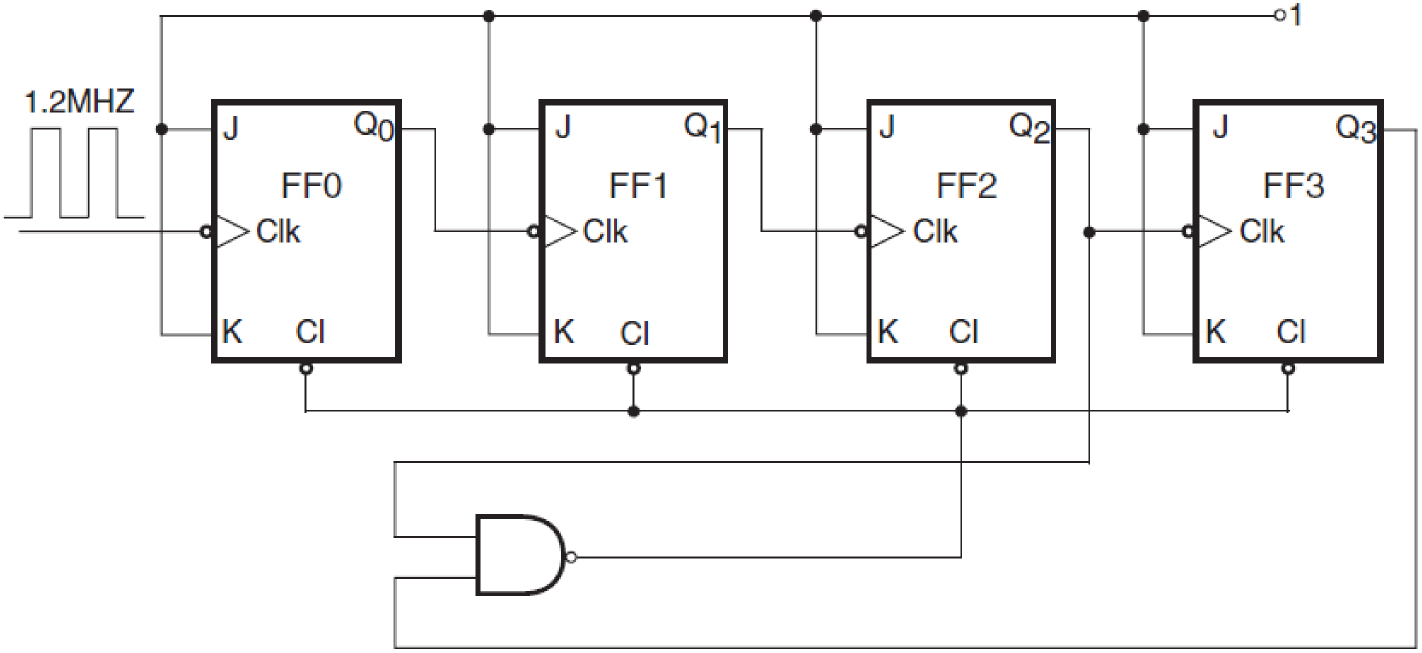
Binary Ripple Counters with a Modulus of < 2^N

Steps to design any binary ripple counter that starts from 0000 and has a modulus of X:

1. Determine the minimum number of flip flops N so that $2^N \geq X$. Connect these flip flops as a binary ripple counter. If $2^N = X$, do not go to steps 2 and 3.
2. Identify the flip flops that will be in the logic HIGH state at the count whose decimal equivalent is X. Choose a NAND gate with the number of inputs equal to the number of flip flops that would be in the logic HIGH state.
3. Connect the Q outputs of the identified flip flops to the inputs of the NAND gate and the NAND gate output to asynchronous clear inputs of all flip flops.

Example 12

- Determine the modulus of the counter and also the frequency of the flip-flop Q_3 output.

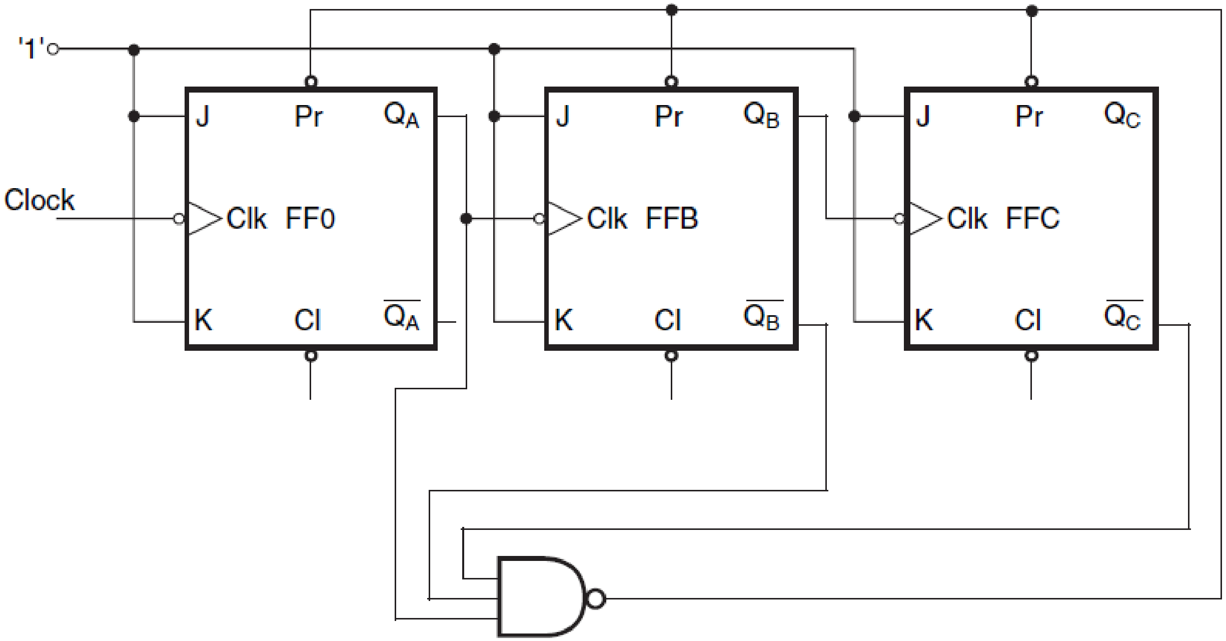


- Answer: the modulus = 12

  f_{Q3} = 100kHz

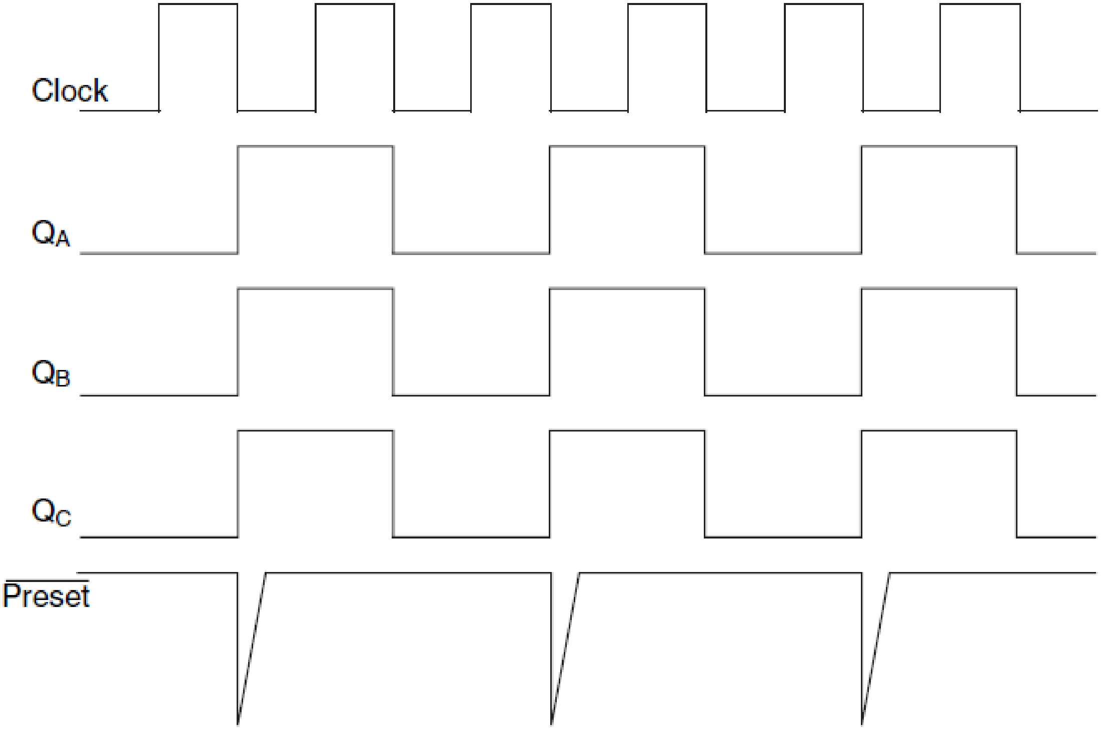
Example 13

- Design a binary ripple counter that counts 000 and 111 and skips the remaining six states.
- Use JK flip flops and draw the timing waveforms of the outputs.



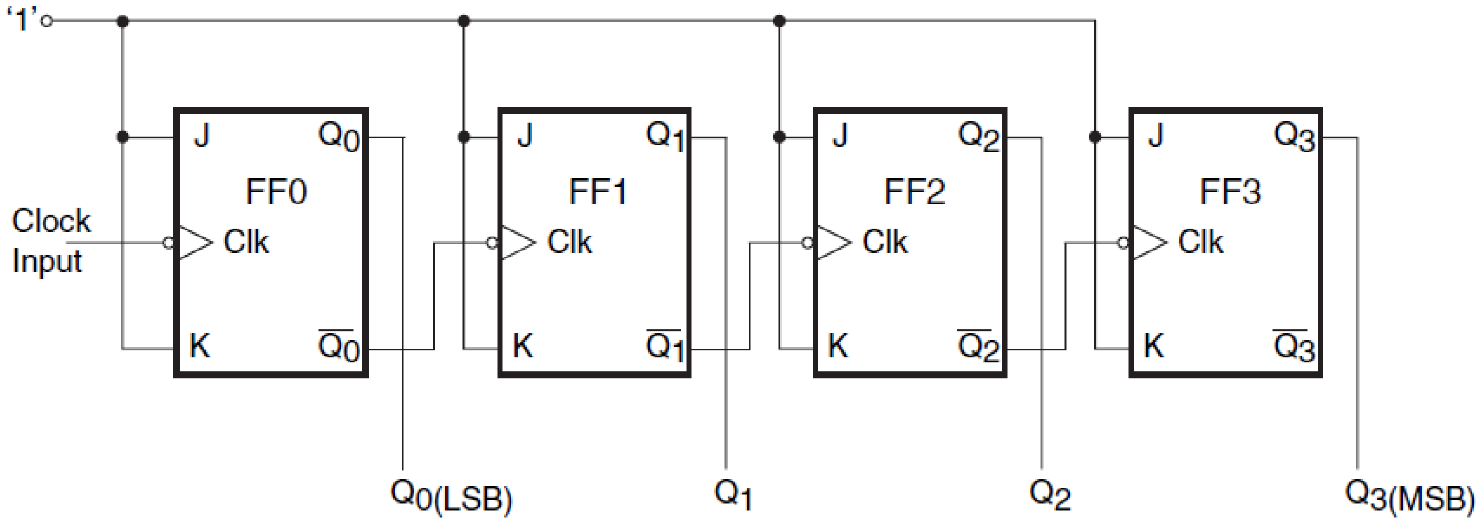
Example 13

- Output waveform:



Example 14

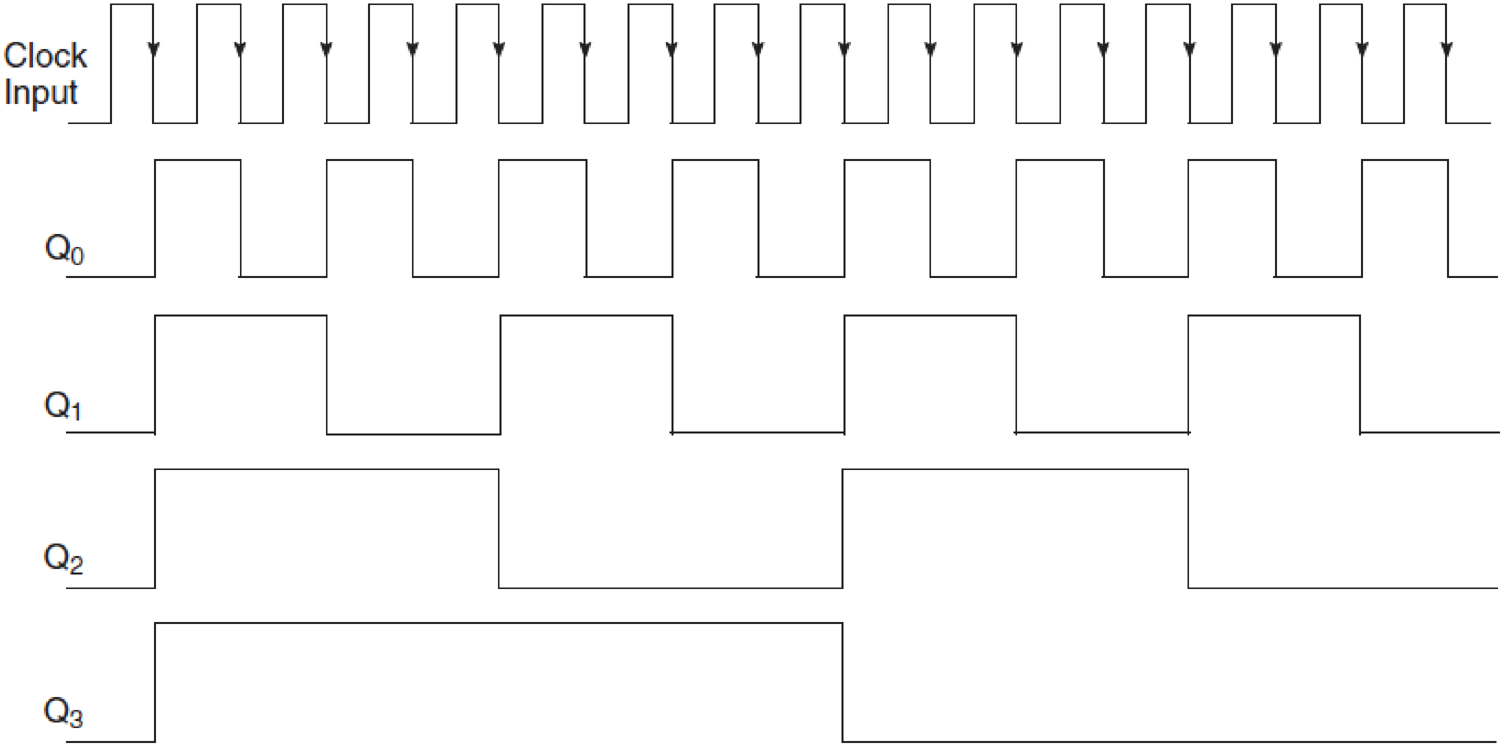
- Write its count sequence if it is initially in the 0000 state. Also draw the timing waveforms.



0000 → 1111 → 1110 → 1101 → 1100 → 1011 → 1010 → 1001 → 1000 → 0111 → 0110 → 0101 → 0100 → 0011 → 0010 → 0001 → 0000

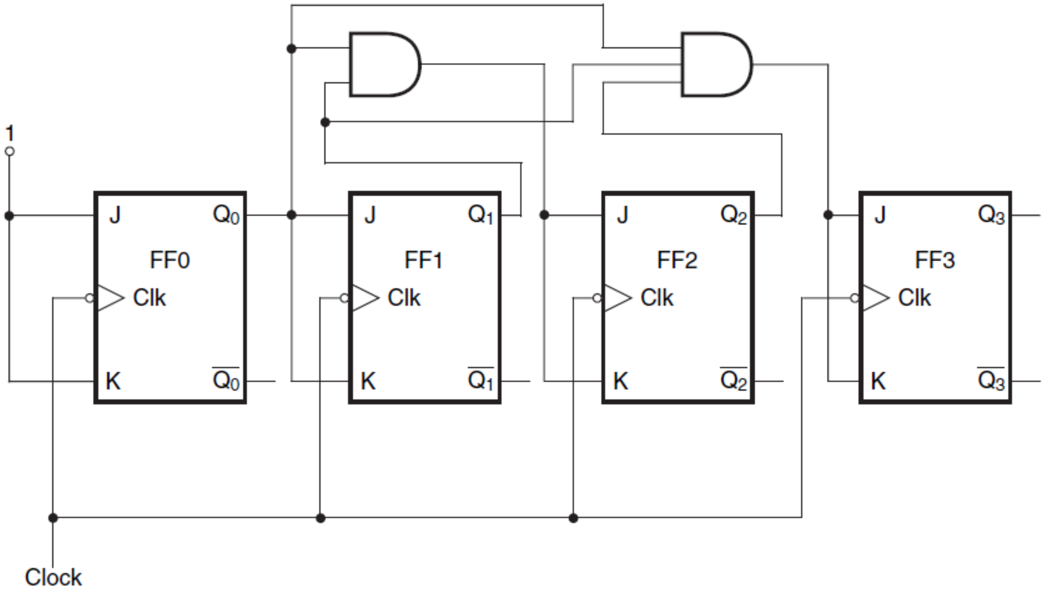
Example 14

- Output waveform:



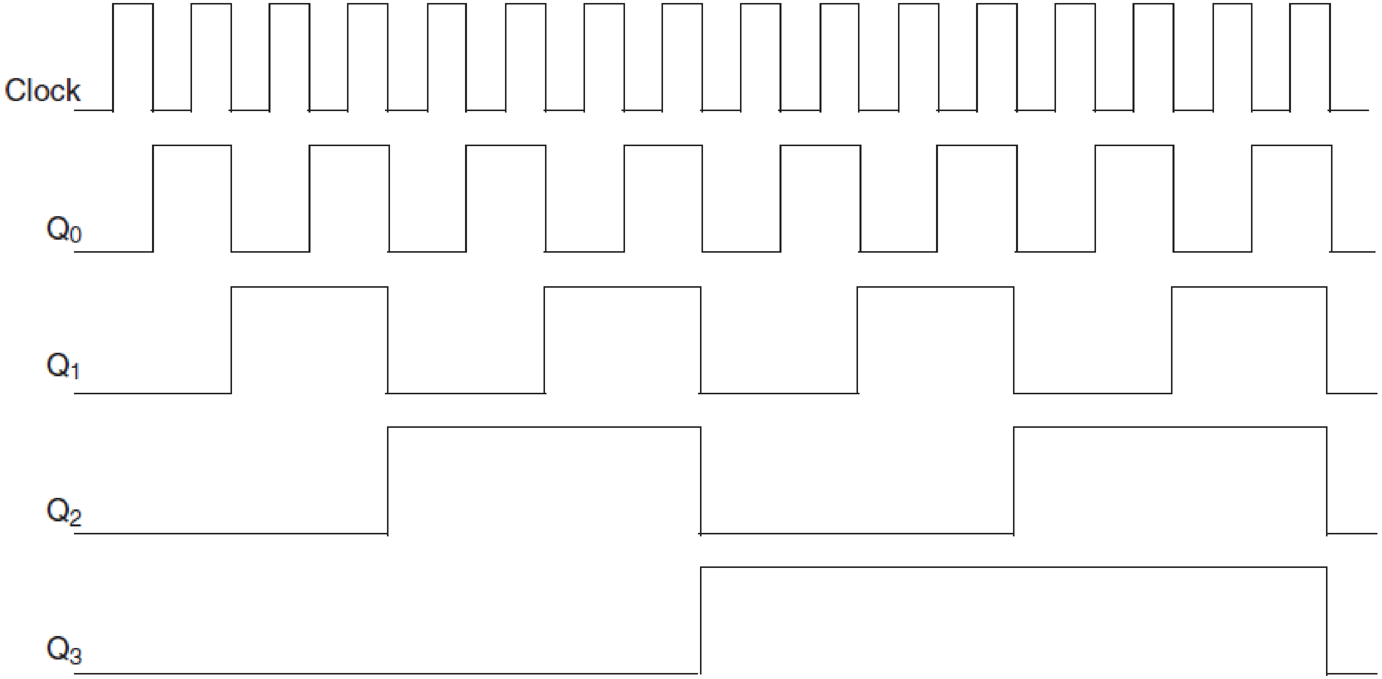
Synchronous Counter

- All the flip flops in the counter change state at the same time in synchronism with the input clock signal.
- The total propagation delay is equal to that of one flip flop.
- Example: a 4-bit synchronous up counter



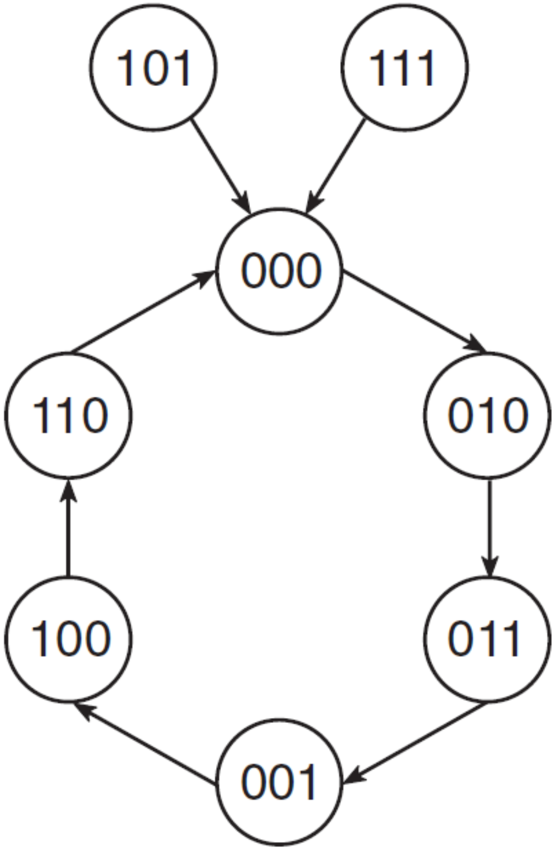
Synchronous Counter

- Output waveform of a 4-bit synchronous up counter



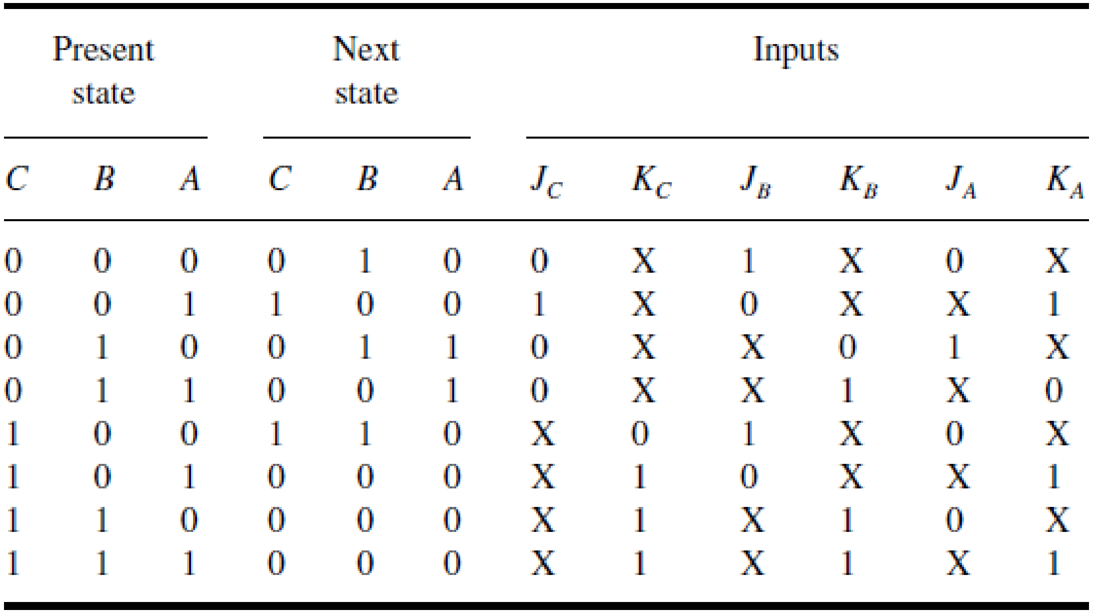
Designing Counters with Arbitrary Sequences

- **Design procedure:**
  - Determine the number of flip flops required for the purpose.
  - Identify the undesired states.
  - Draw the state transistion diagram (including undesired states).
  - The undesired states should be depicted to be transiting to any of the desired states.



Designing Counters with Arbitrary Sequences

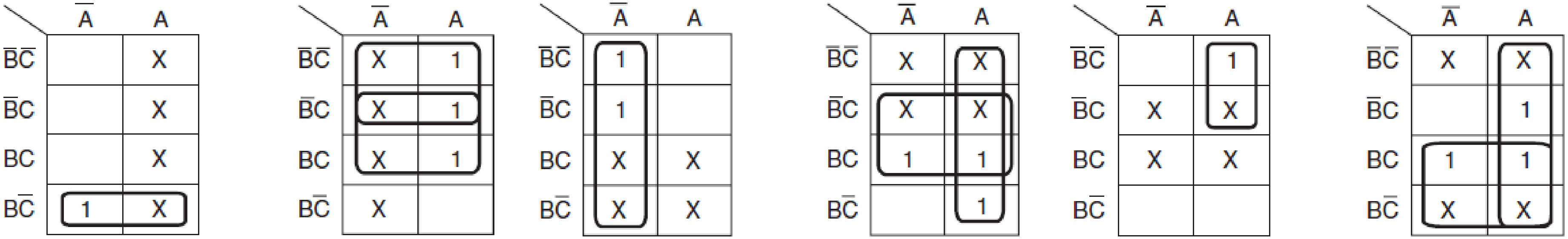
- **Design procedure:**
  - Draw the excitation table for the counter, listing:
    - The present states
    - The next states corresponding to the present states
    - The required logic status of the flip flop inputs



Present state			Next state			Inputs					
C	B	A	C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	0	0	X	1	X	0	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

Designing Counters with Arbitrary Sequences

- **Design procedure:**
  - Design the logic circuits for generating inputs (J_A, K_A, J_B, K_B, J_C, K_C) from available outputs (A, B, C…)
  - Can use the K-map for logic simplification
  - Example:



	$\overline{A}$	A
$\overline{B}\overline{C}$		X
$\overline{B}C$		X
BC		X
$B\overline{C}$	1	X

	$\overline{A}$	A
$\overline{B}\overline{C}$	X	1
$\overline{B}C$	X	1
BC	X	1
$B\overline{C}$	X	

	$\overline{A}$	A
$\overline{B}\overline{C}$	1	
$\overline{B}C$	1	
BC	X	X
$B\overline{C}$	X	X

	$\overline{A}$	A
$\overline{B}\overline{C}$	X	X
$\overline{B}C$	X	X
BC	1	1
$B\overline{C}$		1

	$\overline{A}$	A
$\overline{B}\overline{C}$		1
$\overline{B}C$	X	X
BC	X	X
$B\overline{C}$		

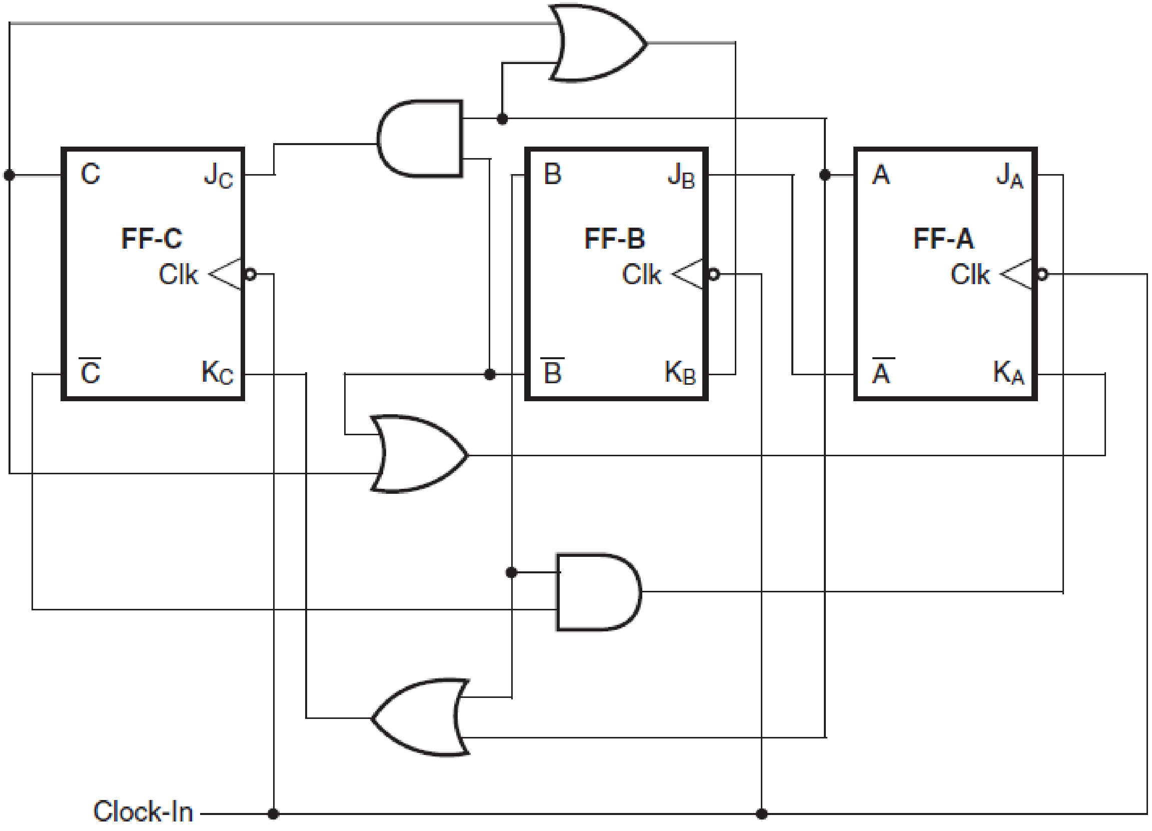
	$\overline{A}$	A
$\overline{B}\overline{C}$	X	X
$\overline{B}C$		1
BC	1	1
$B\overline{C}$	X	X

$$J_A = B.\overline{C} \qquad J_B = \overline{A} \qquad J_C = A.\overline{B}$$

$$K_A = \overline{B} + C \qquad K_B = A + C \qquad K_C = A + B$$

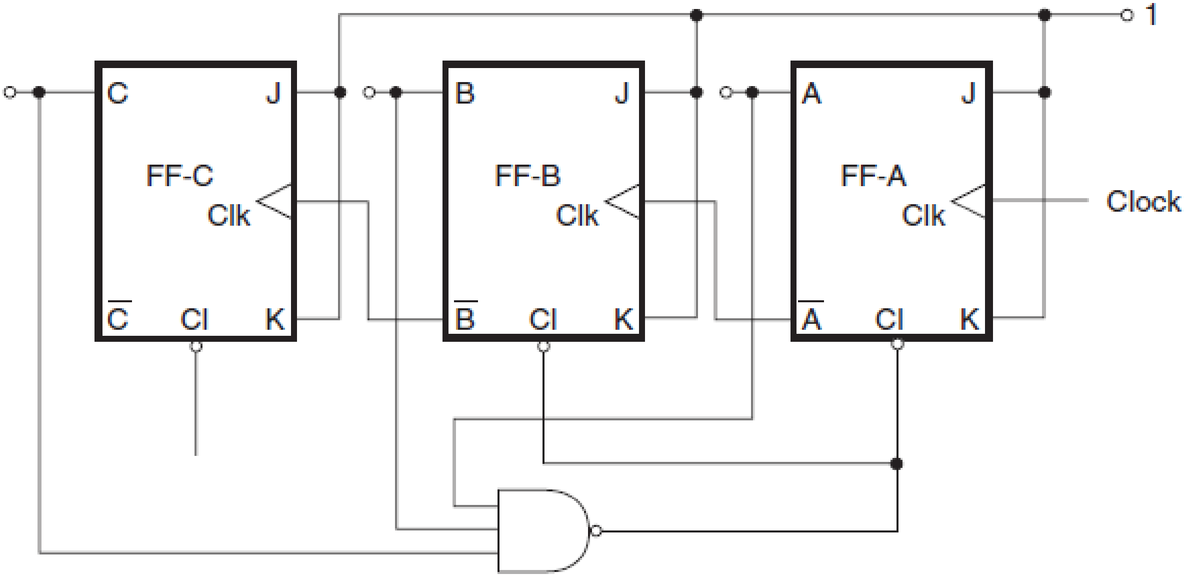
Designing Counters with Arbitrary Sequences

- Logic circuit diagram:



Example 15

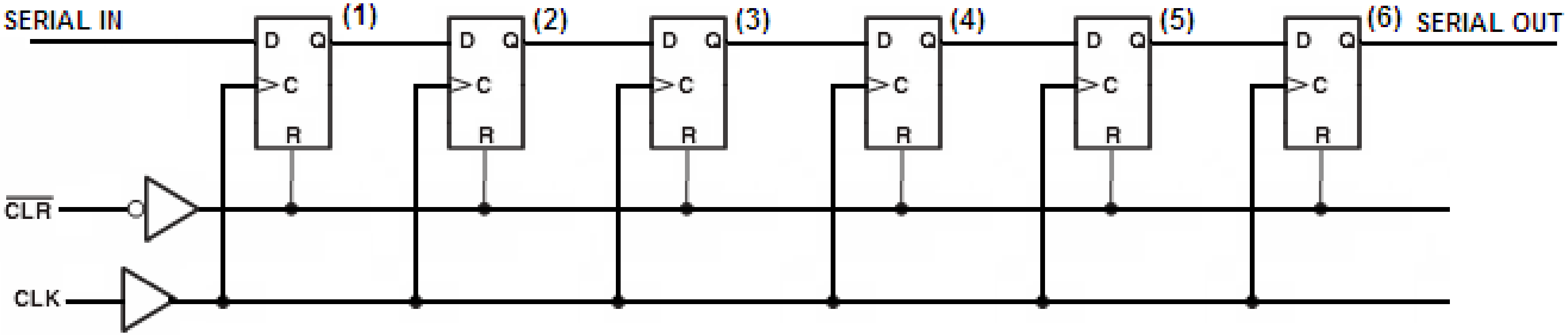
- Find the count sequence of the following counter:



000, 001, 010, 011, 100, 101, 110, 000, . . .

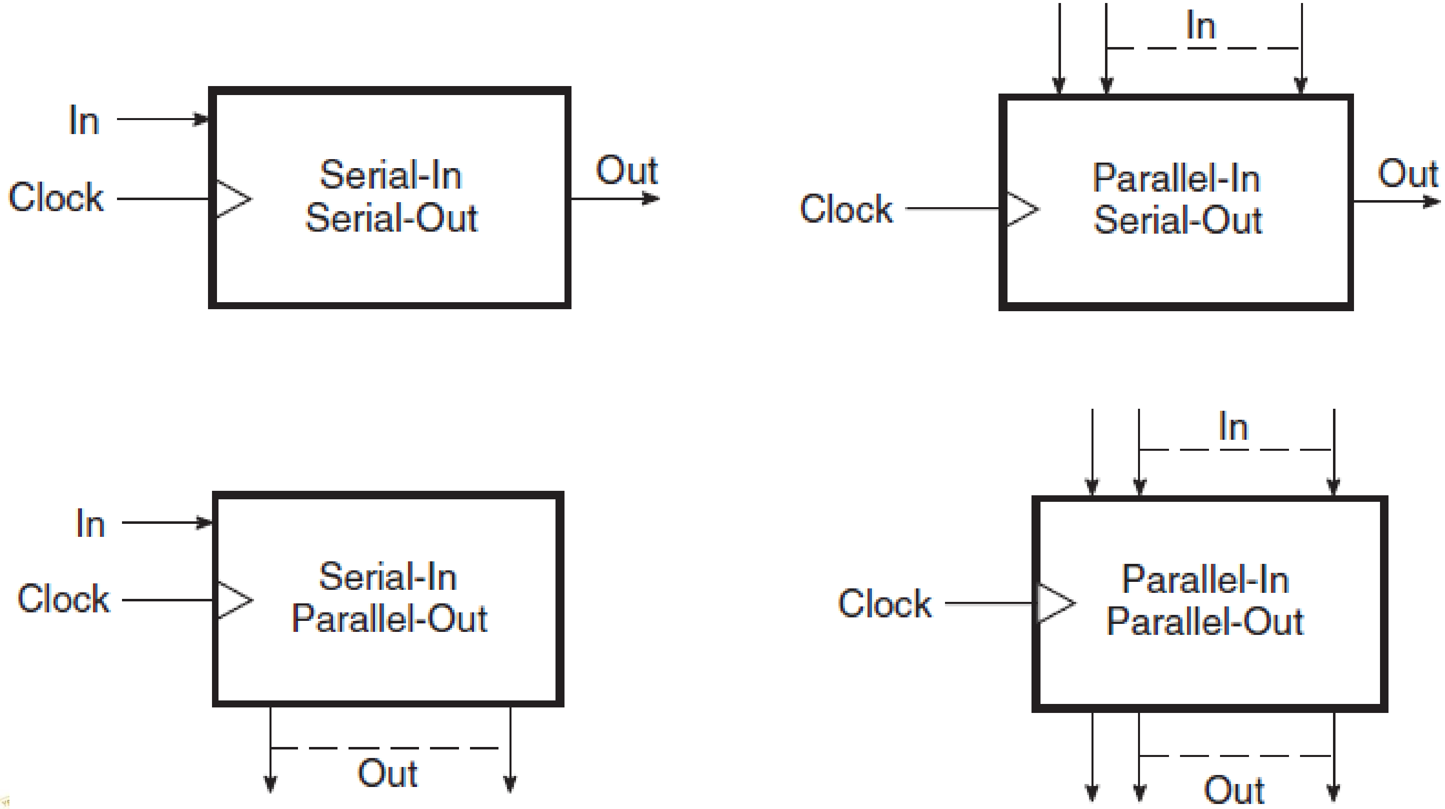
Shift Register

- A shift register is a digital device used for storage and transfer of data.
- Example:



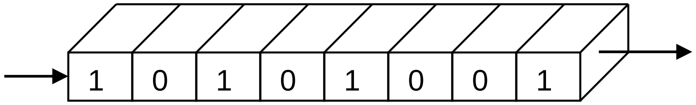
Shift Registers

- Based on the method used to load data onto and read data from shift registers, they can be classified as: SISO, SIPO, PISO, and PIPO.

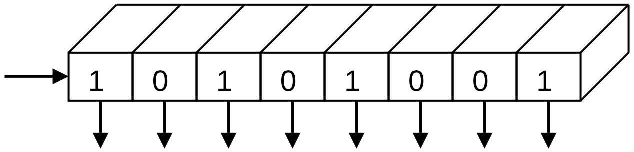


Shift Registers

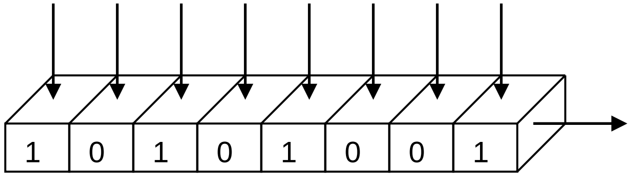
- Serial In, Serial Out (SISO):



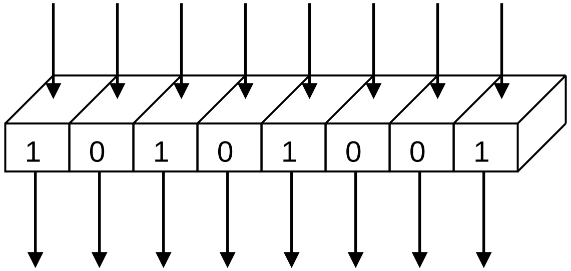
- Serial In, Parallel Out (SIPO):



- Parallel In, Serial Out (PISO):

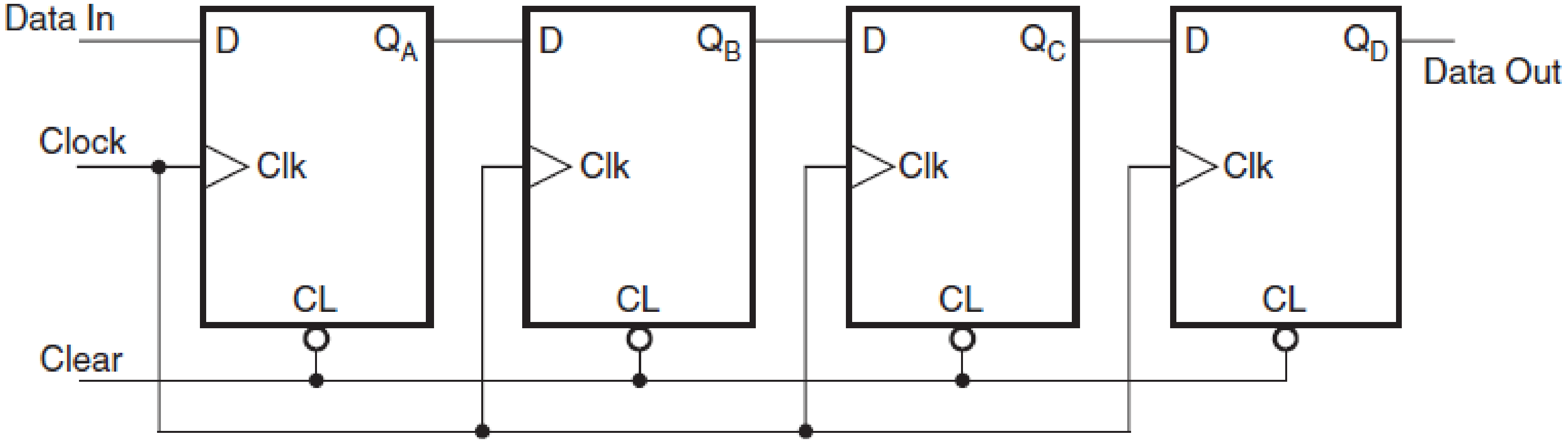


- Parallel In, Parallel Out (PIPO):



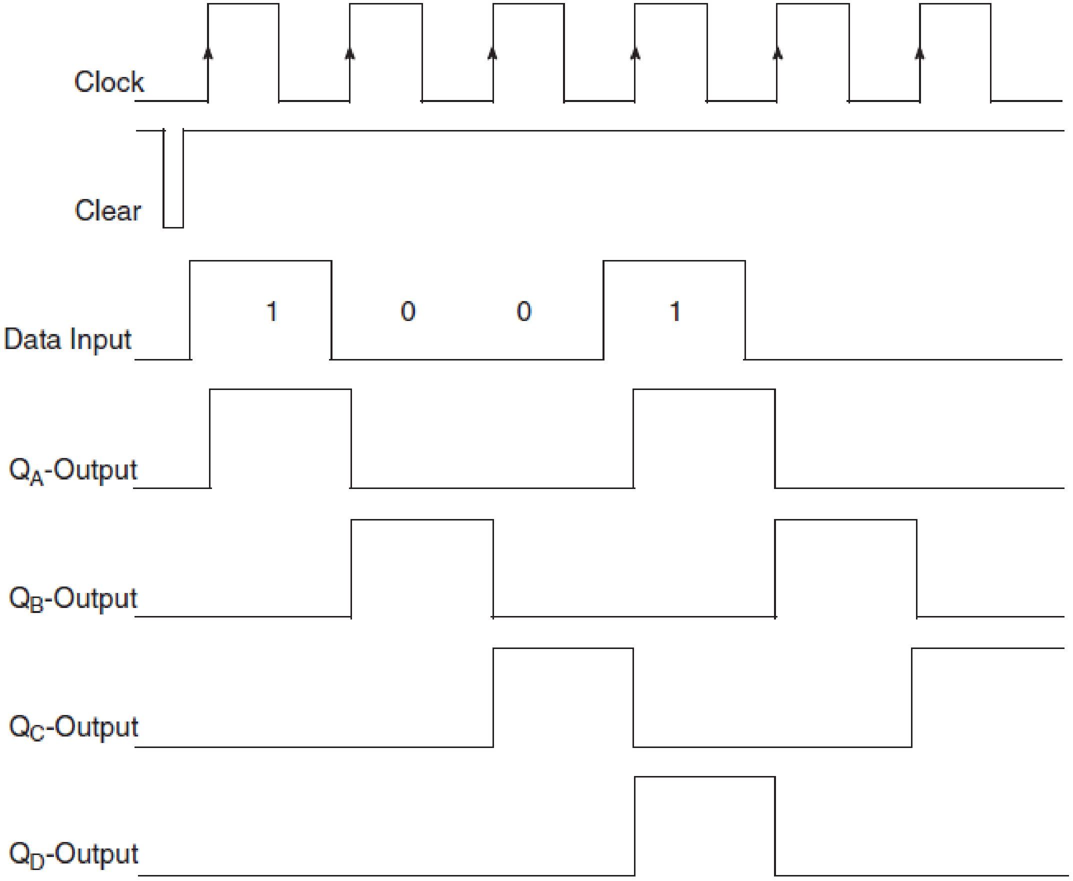
SISO

- Block diagram:



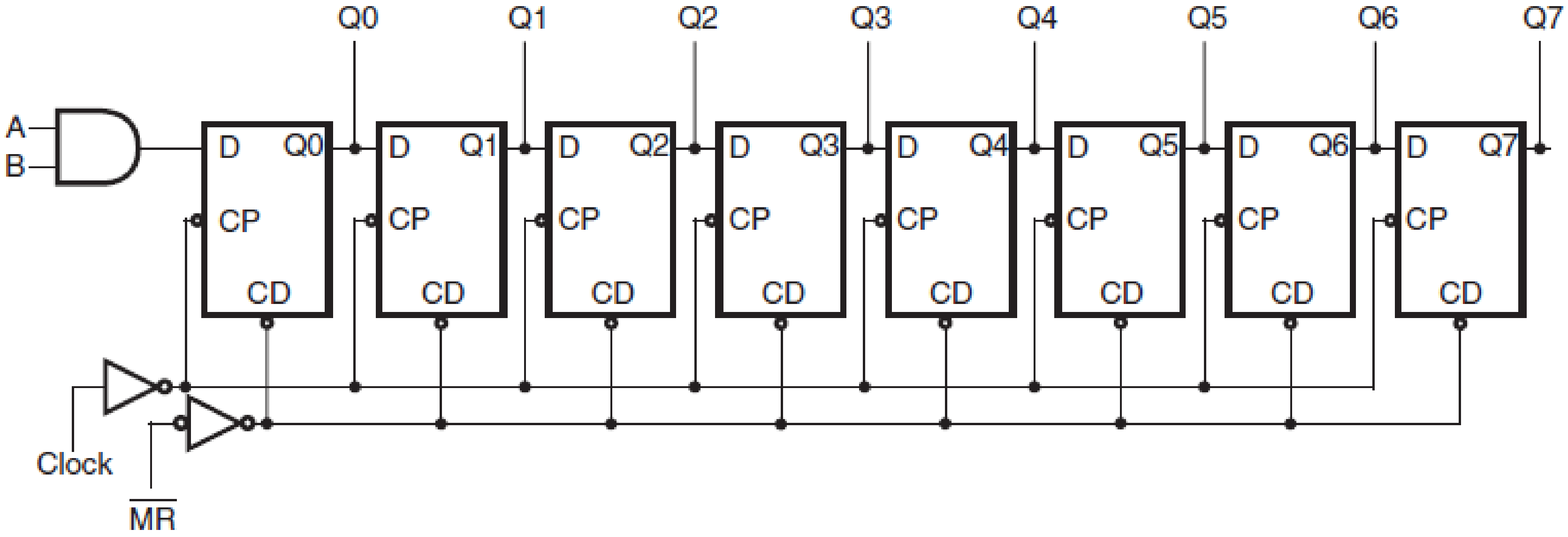
SISO

- Timing diagram:



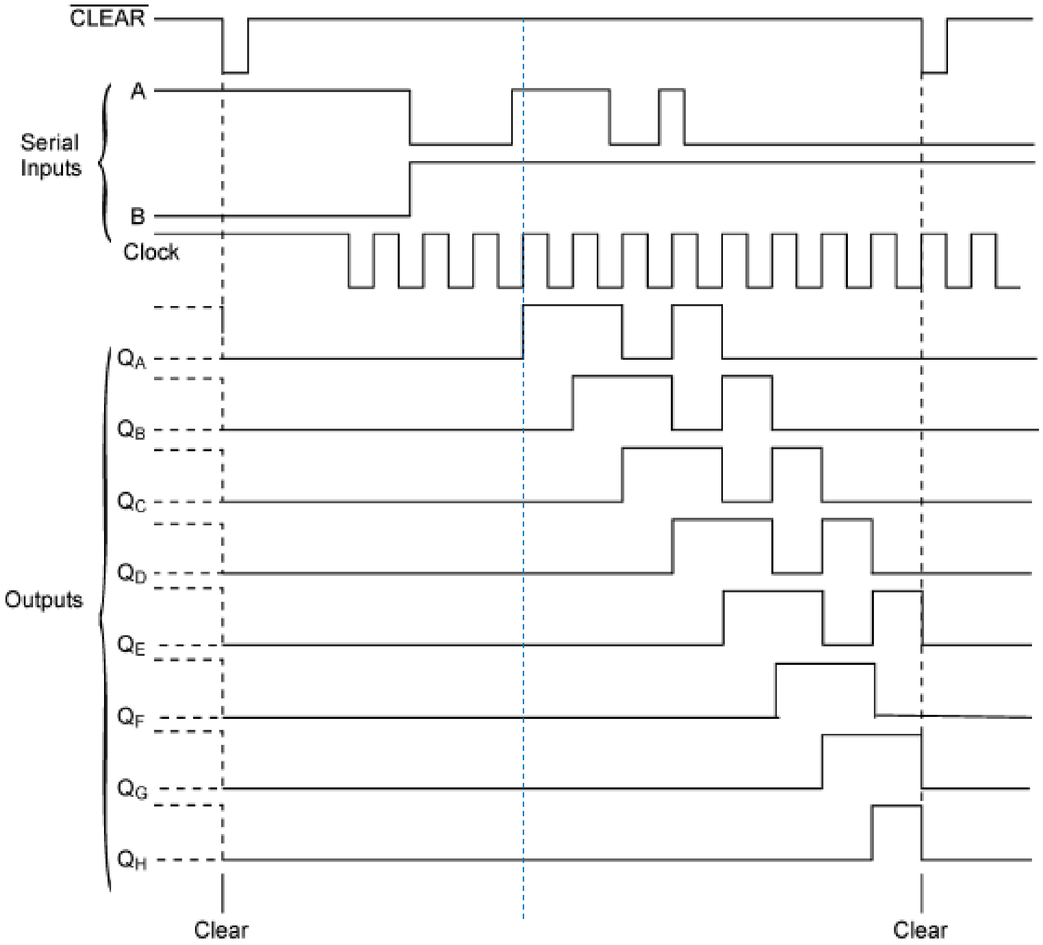
SIPO

- Block diagram:



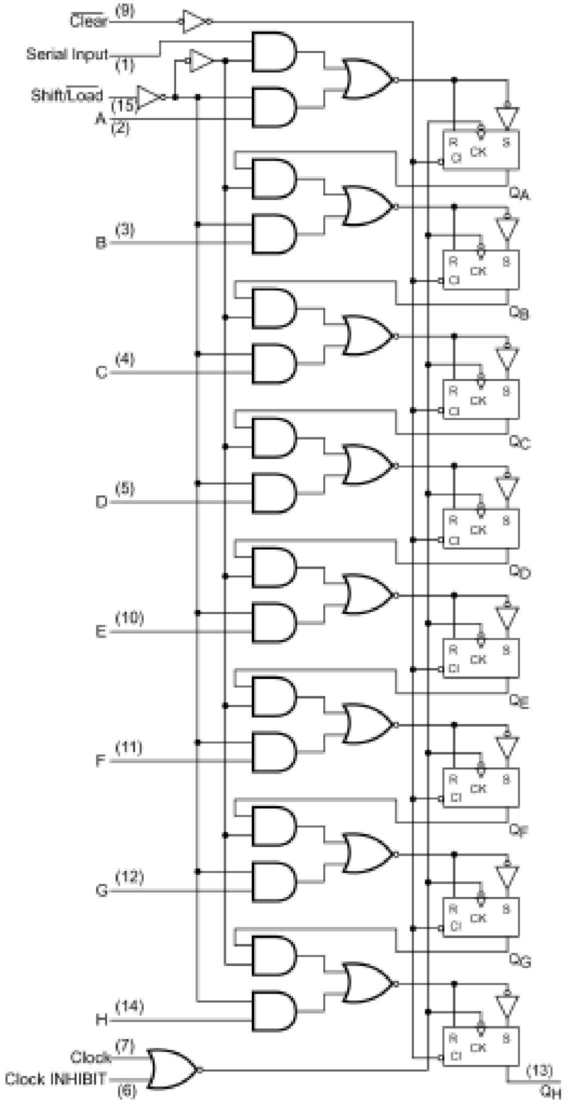
SIPO

- Timing diagram:



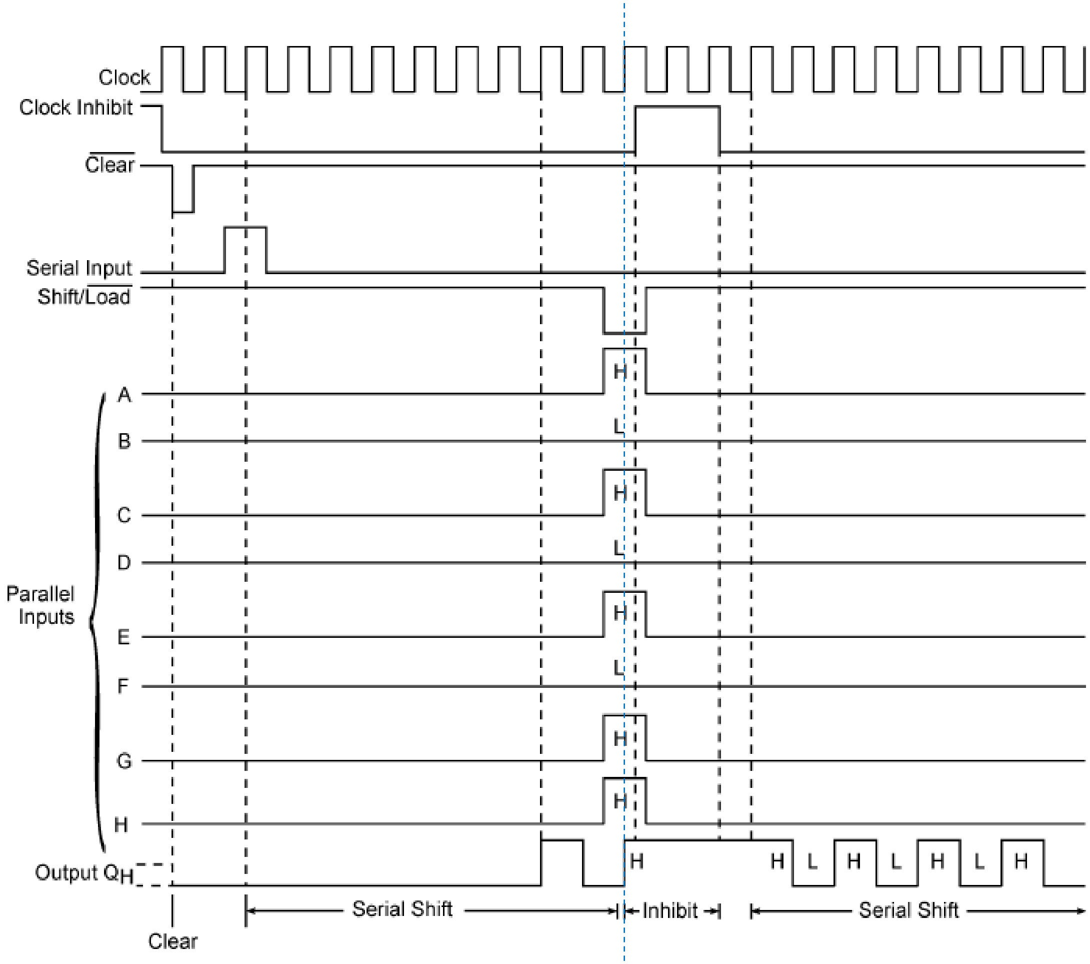
PISO

- Block diagram:



PISO

- Timing diagram:


Clock
Clock Inhibit
Clear
Serial Input
Shift/Load
A
B
C
D
E
F
G
H
Parallel Inputs
Output QH
H
L
H
L
H
L
H
H
H
H
L
H
L
H
L
H
Serial Shift
Inhibit
Serial Shift
Clear