ESC201: Lecture 17

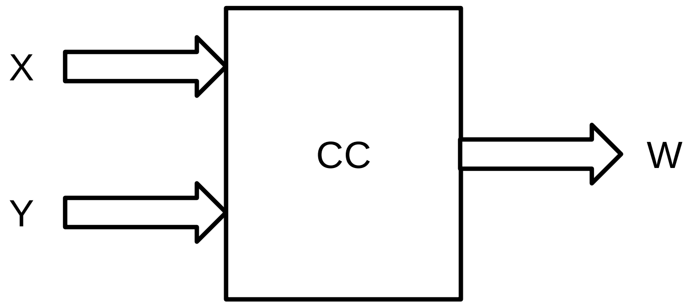
Dr. Imon Mondal

Assistant Professor,
Electrical Engineering, IIT Kanpur

2024-25 Sem-I | **ESC201** Introduction to Electronics

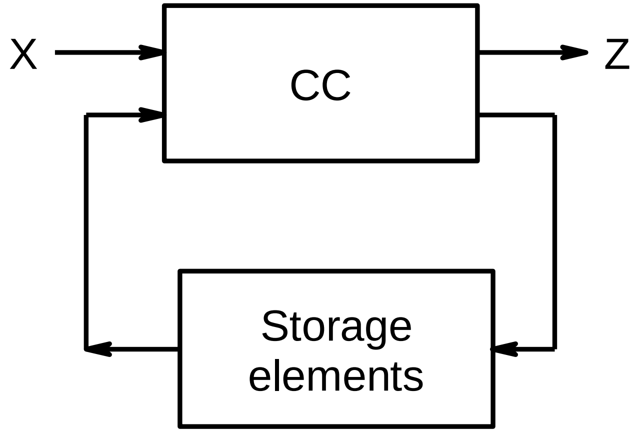
Digital Circuits

Combinational Circuits

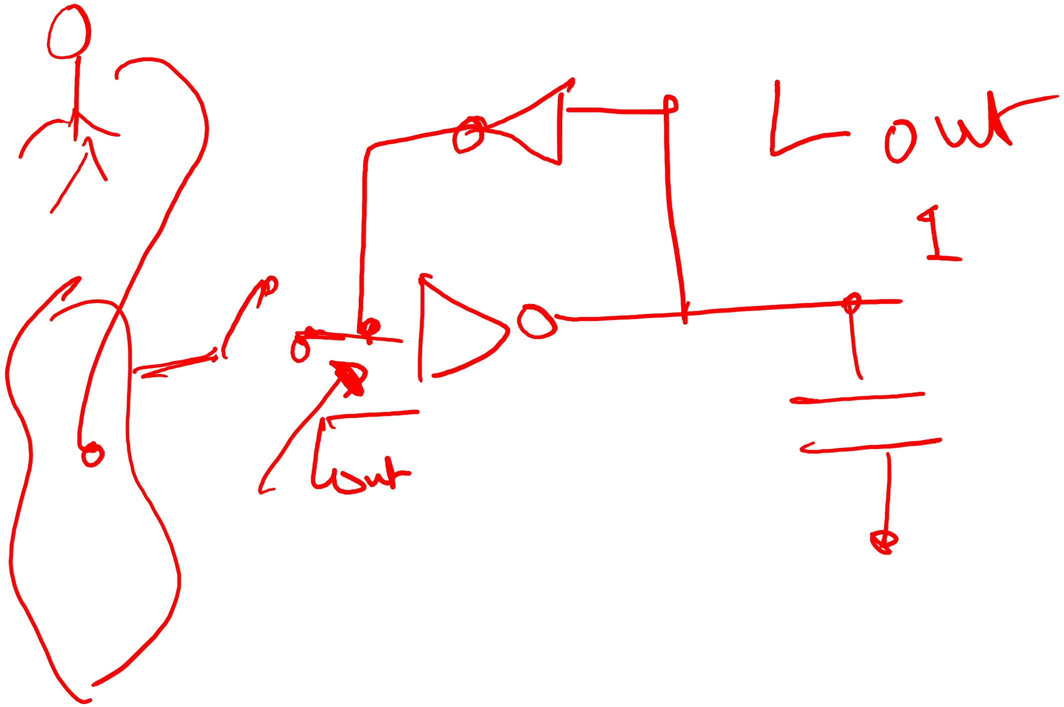


Output is determined by current values of inputs only.

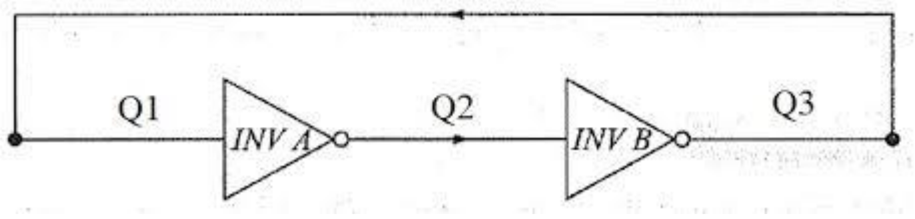
Sequential Circuits



Output is determined in general by current values of inputs and past values of inputs/outputs as well.

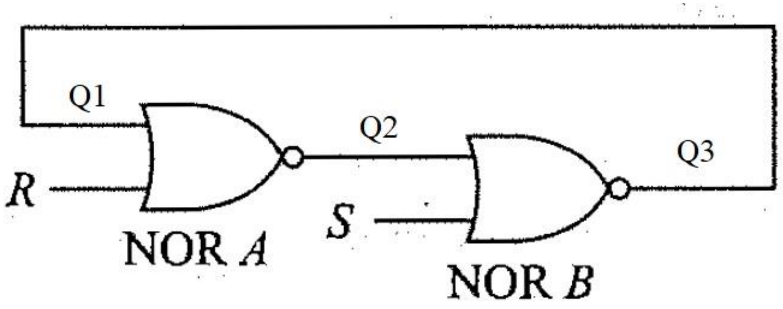

Lout
1
Lout

Memory with Set/Reset Knob

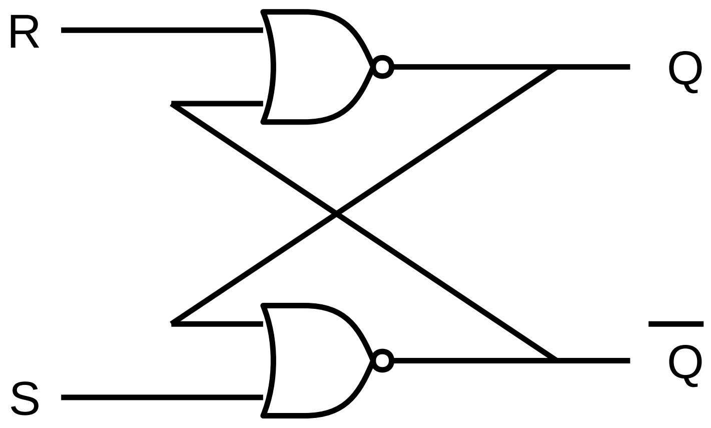


Two memory states are possible:
1. Q1=Q3=0 (Low), Q2=1 (High)
2. Q1=Q3= 1 (High), Q2= 0 (Low)

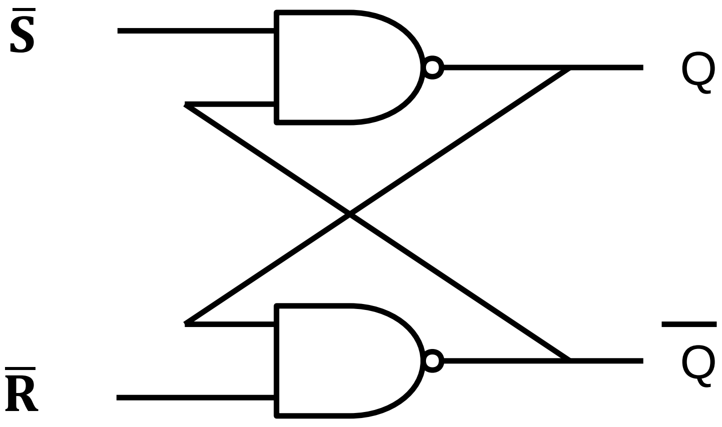
But how will one change the state?



We need at least 2 i/p logic gates
- NOR gate behave like INV if one i/p is 0
- NAND behaves like INV if one i/p is 1

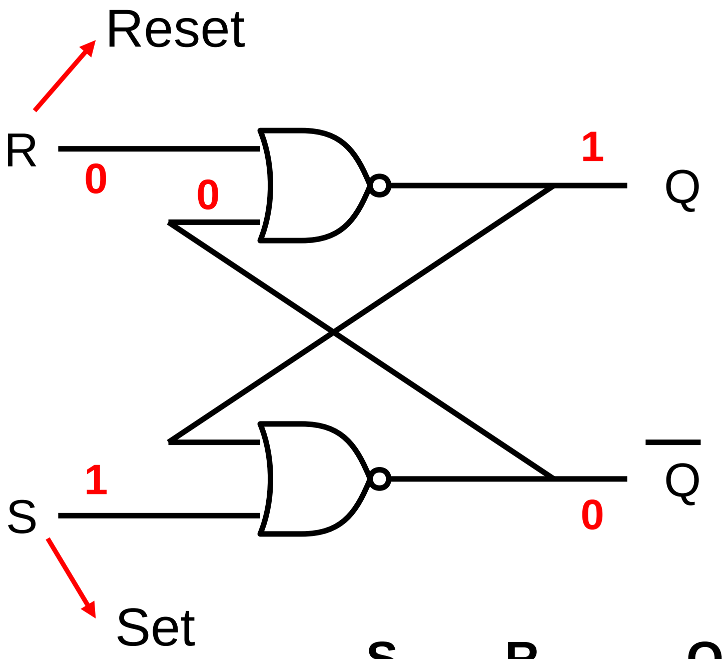


NOR-based SR Latch



NAND-based SR Latch

Set-Reset (SR) Latch: Set State



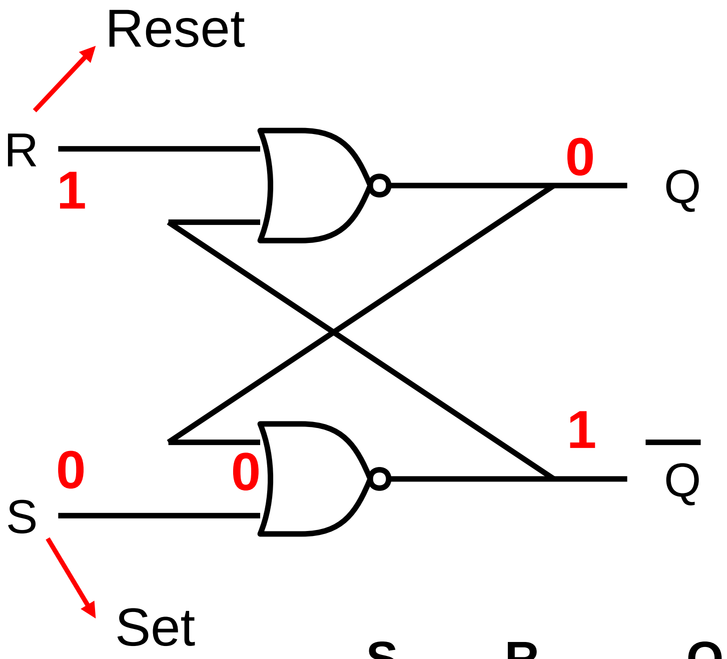
$Q = 1; \overline{Q} = 0$ Set State

$Q = 0; \overline{Q} = 1$ Re set State

$$\overline{1 + Q} = 0$$

S	R	Q	$\overline{Q}$	State
1	0	1	0	SET

Set-Reset (SR) Latch: Reset State



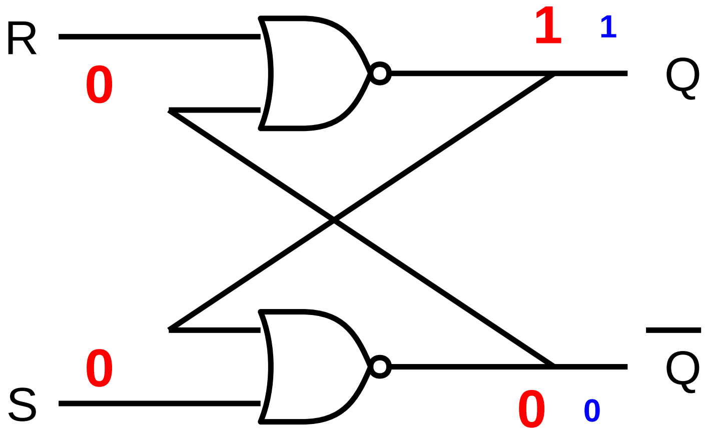
$$Q = 1; \overline{Q} = 0 \quad \text{Set State}$$

$$Q = 0; \overline{Q} = 1 \quad \text{Re set State}$$

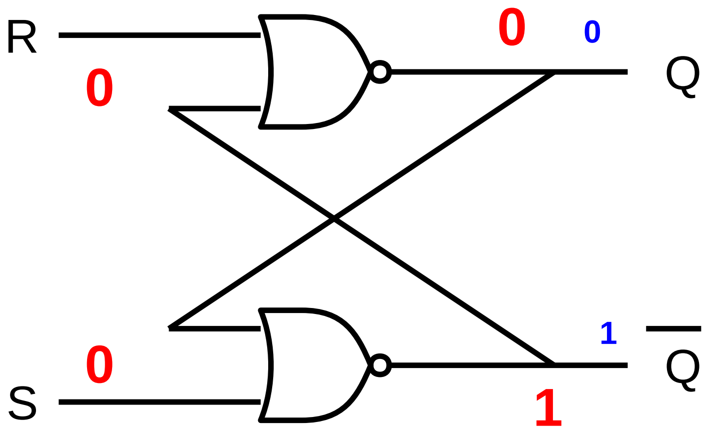
$$\overline{1 + \bar{Q}} = 0$$

S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	1	0	1	RESET

SR Latch: ‘Hold’ (memory)

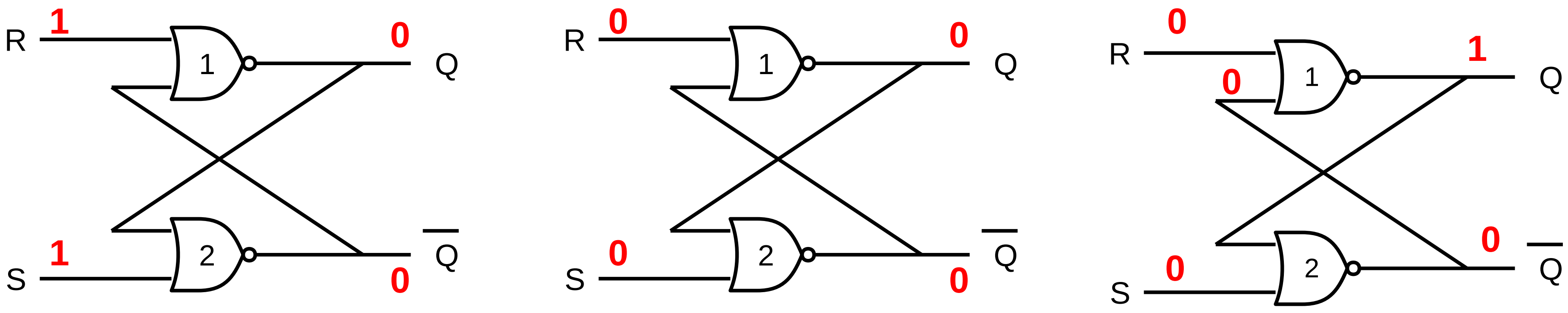


S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	0	1	0	HOLD



S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	0	1	0	HOLD
0	1	0	1	RESET
0	0	0	1	HOLD

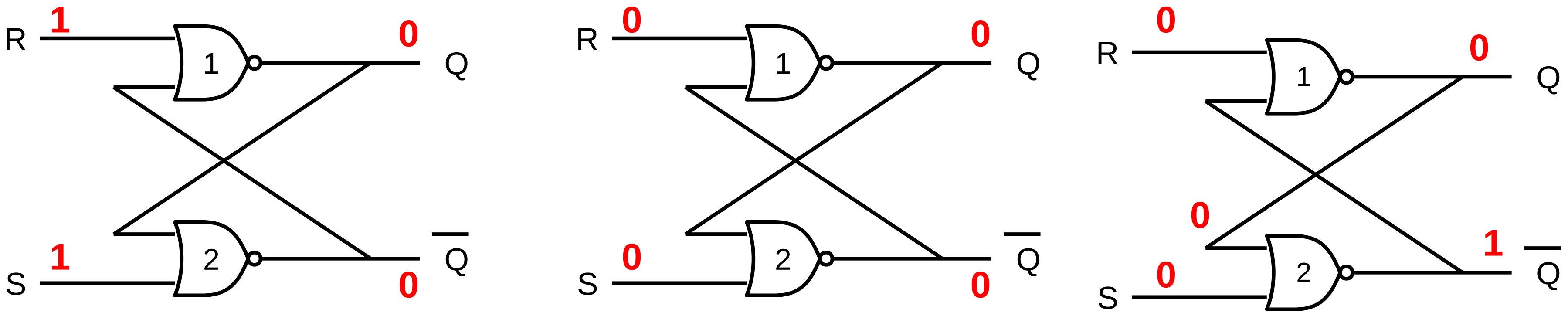
SR Latch: Invalid Input and Gate Delays



Suppose gate-1 is faster

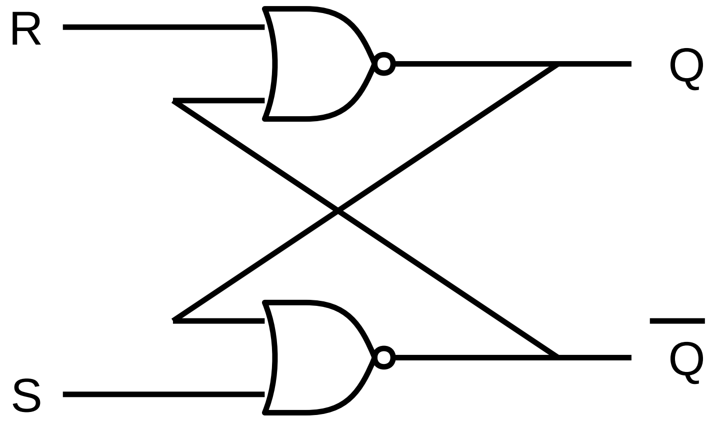
Q = 1

On the other hand suppose that gate-2 is faster.

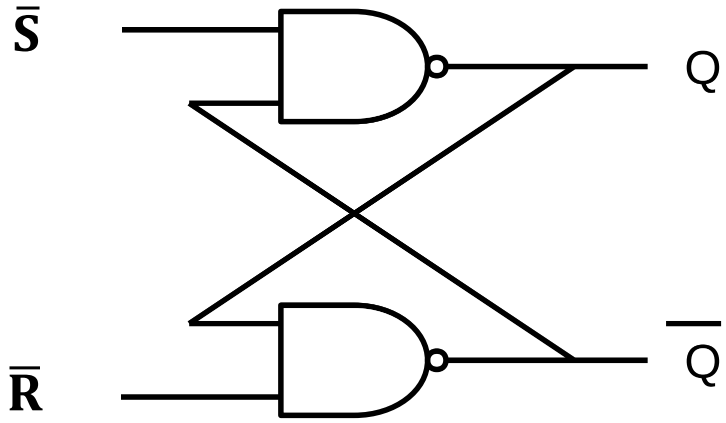


Again the output is unpredictable in general

NOR-based vs NAND-based SR Latch

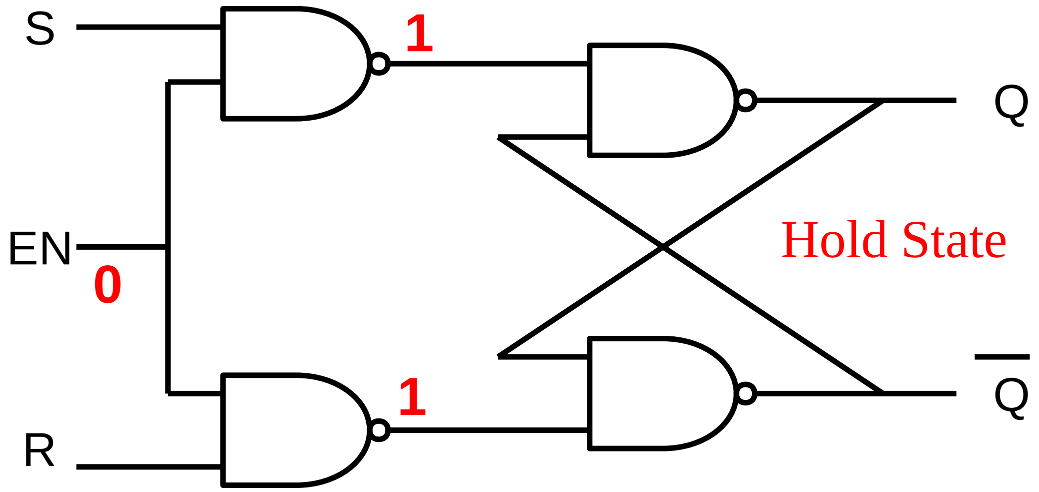


S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	1	0	1	RESET
0	0	Q	$\overline{Q}$	HOLD
1	1	0	0	INVALID

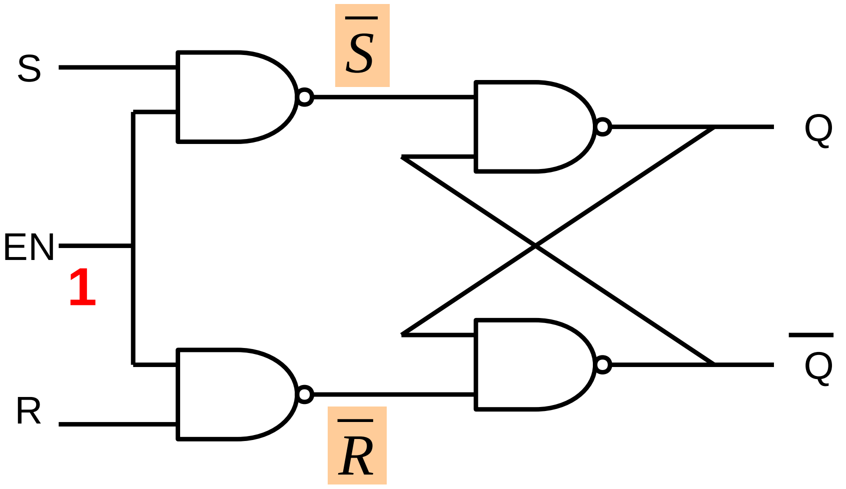


$\bar{S}$	$\bar{R}$	Q	$\overline{Q}$	State
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q	$\overline{Q}$	HOLD
0	0	1	1	INVALID

NAND-based SR Latch with Enable

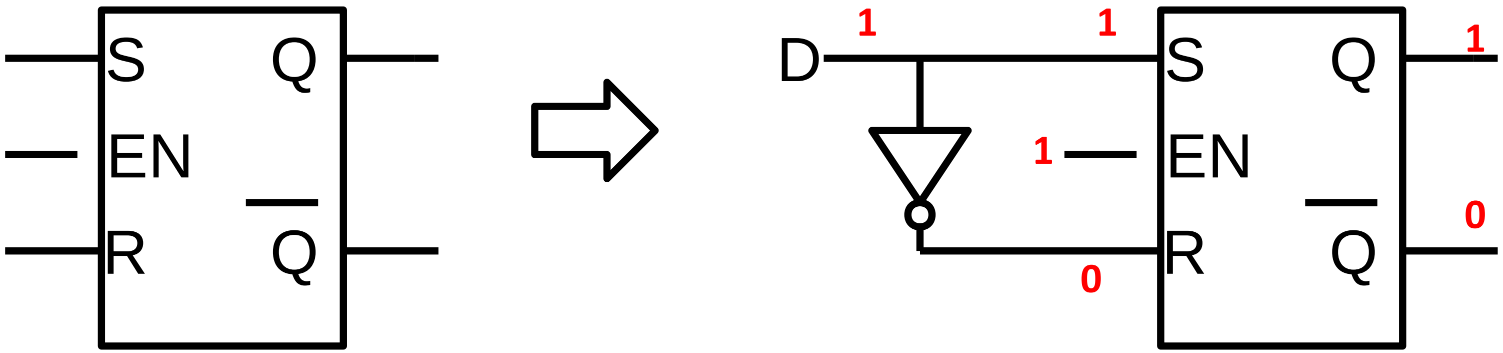


Hold State

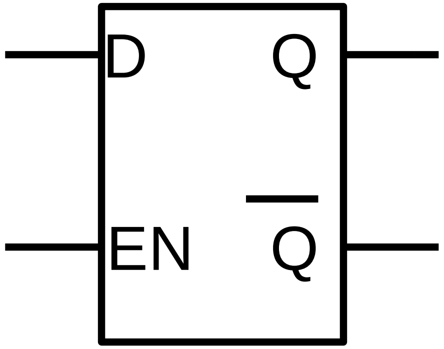


Enable	S R	Q $\overline{Q}$	State
0	x x	Q $\overline{Q}$	Hold
1	1 0	1 0	Set
1	0 1	0 1	Reset
1	0 0	Q $\overline{Q}$	Hold
1	1 1	0 0	Invalid

D latch

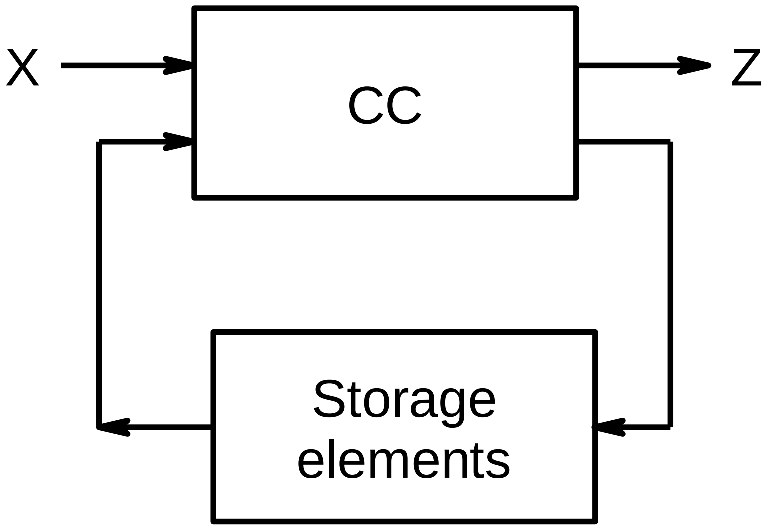


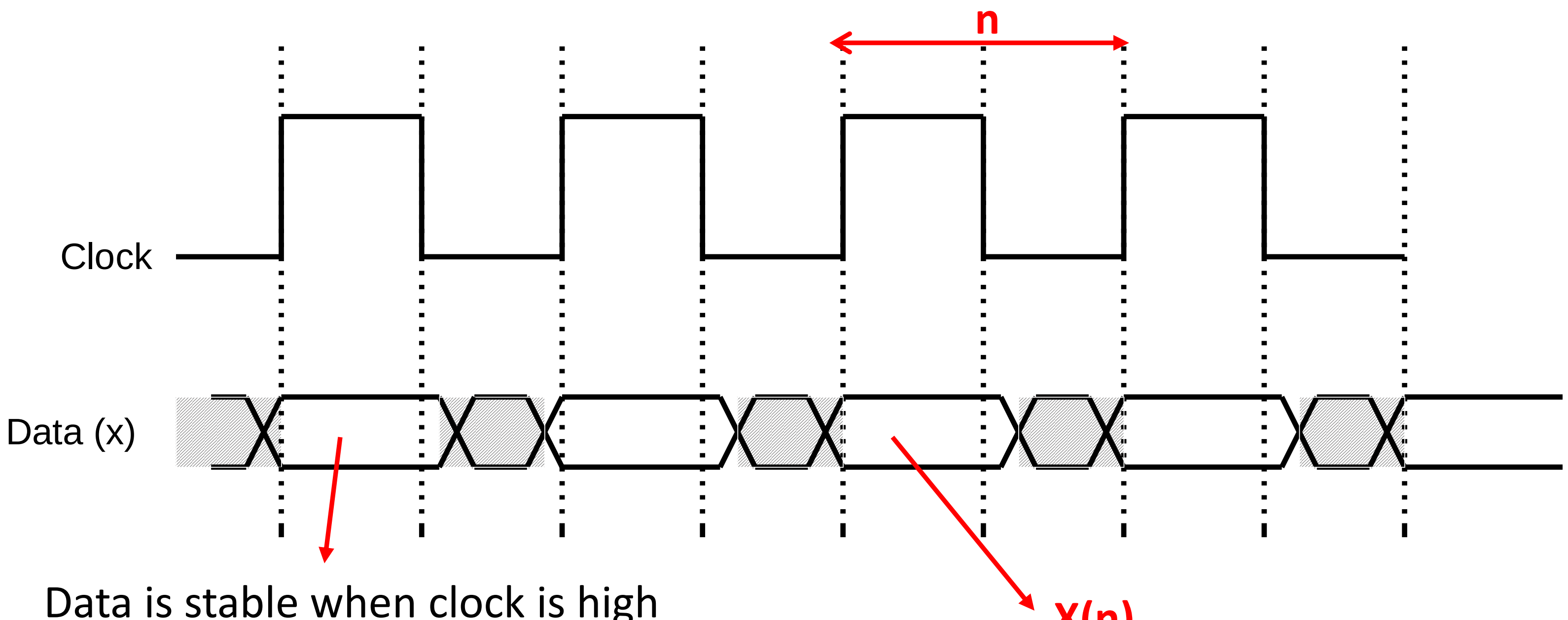
Enable	S R	Q $\overline{Q}$	State
0	x x	Q $\overline{Q}$	Hold
1	1 0	1 0	Set
1	0 1	0 1	Reset
1	0 0	Q $\overline{Q}$	Hold
1	1 1	0 0	Invalid



If EN = 1 then Q = D otherwise the latch is in Hold state

Synchronous Sequential Circuits

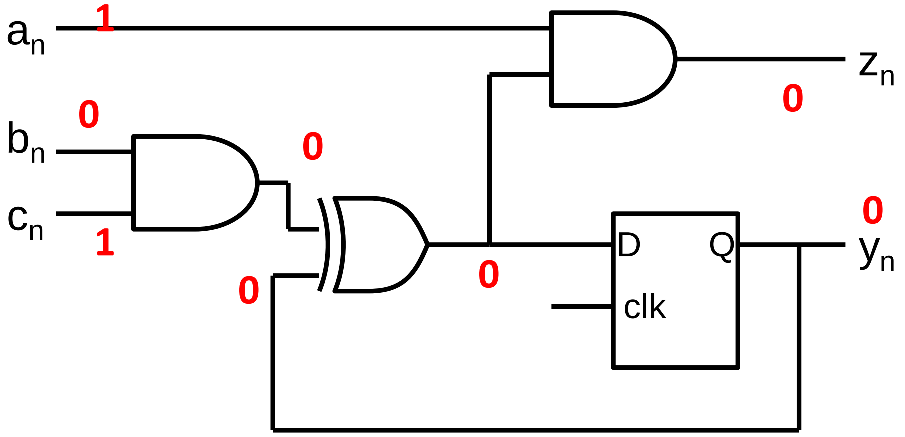


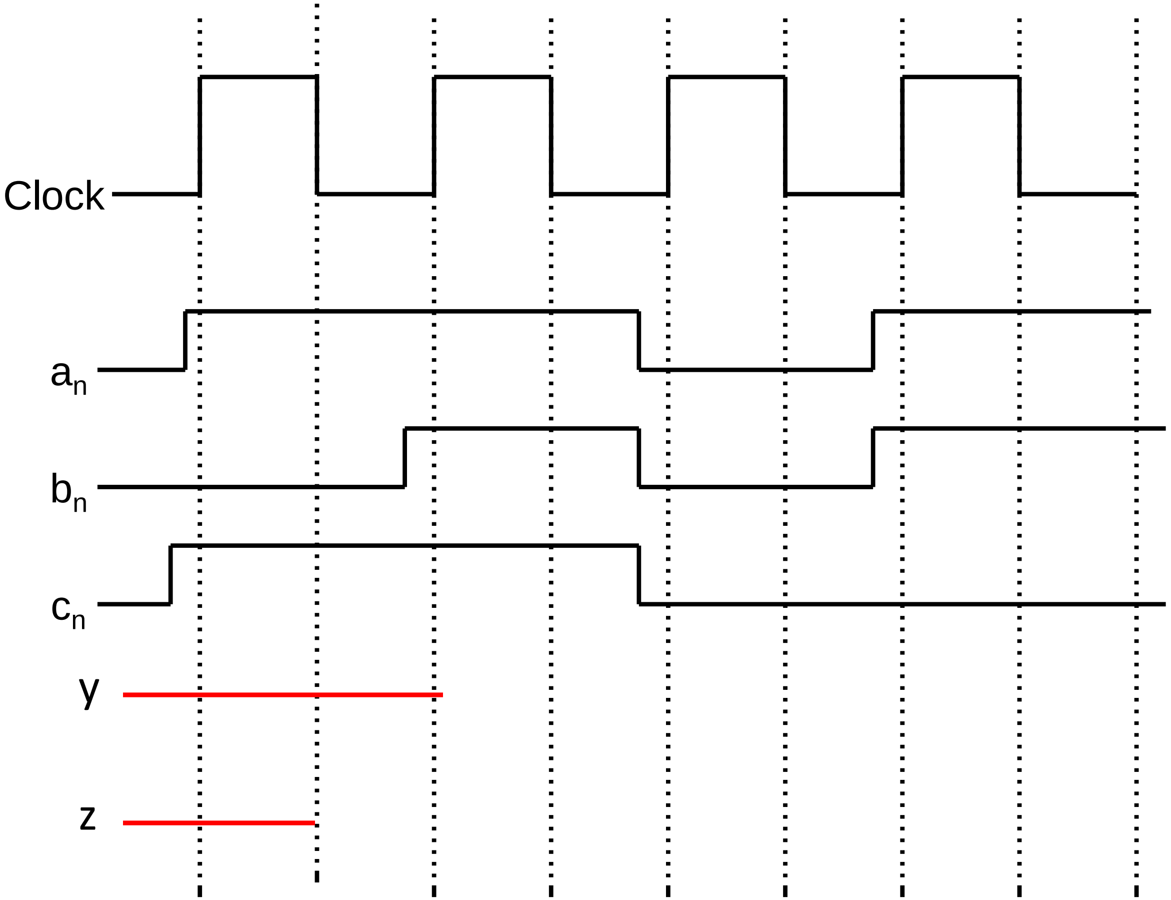


Data is stable when clock is high

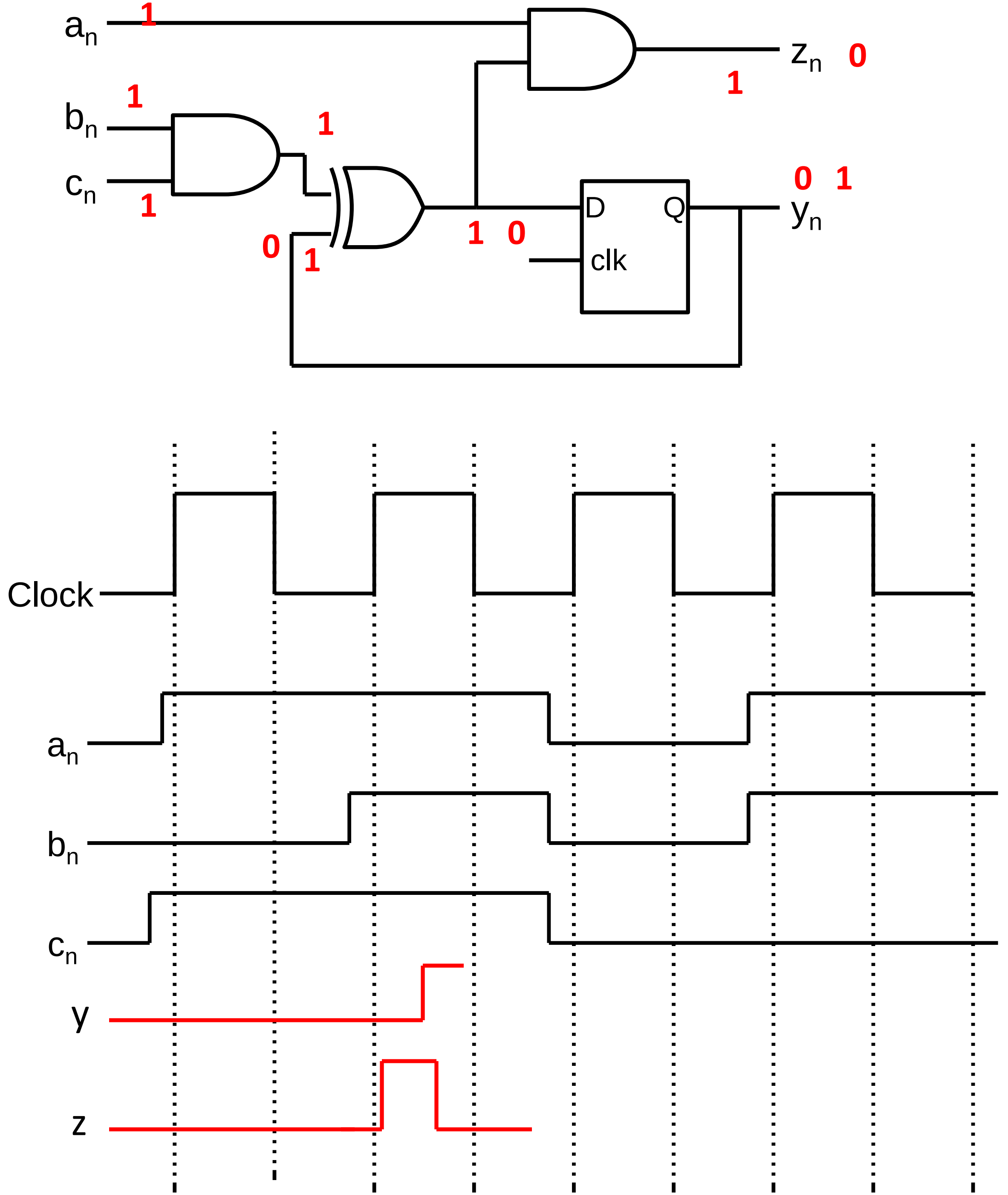
X(n)

Example

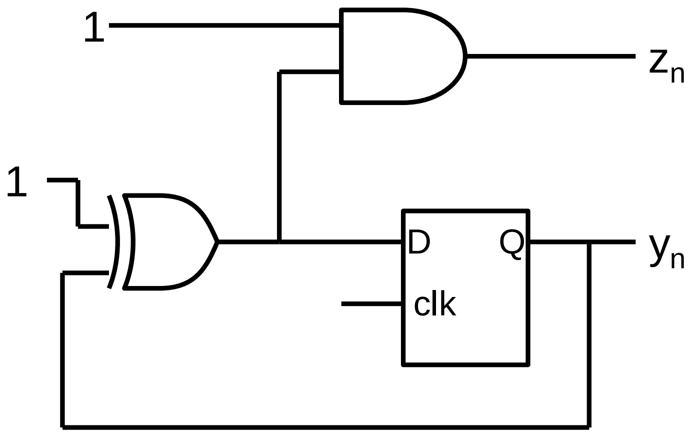




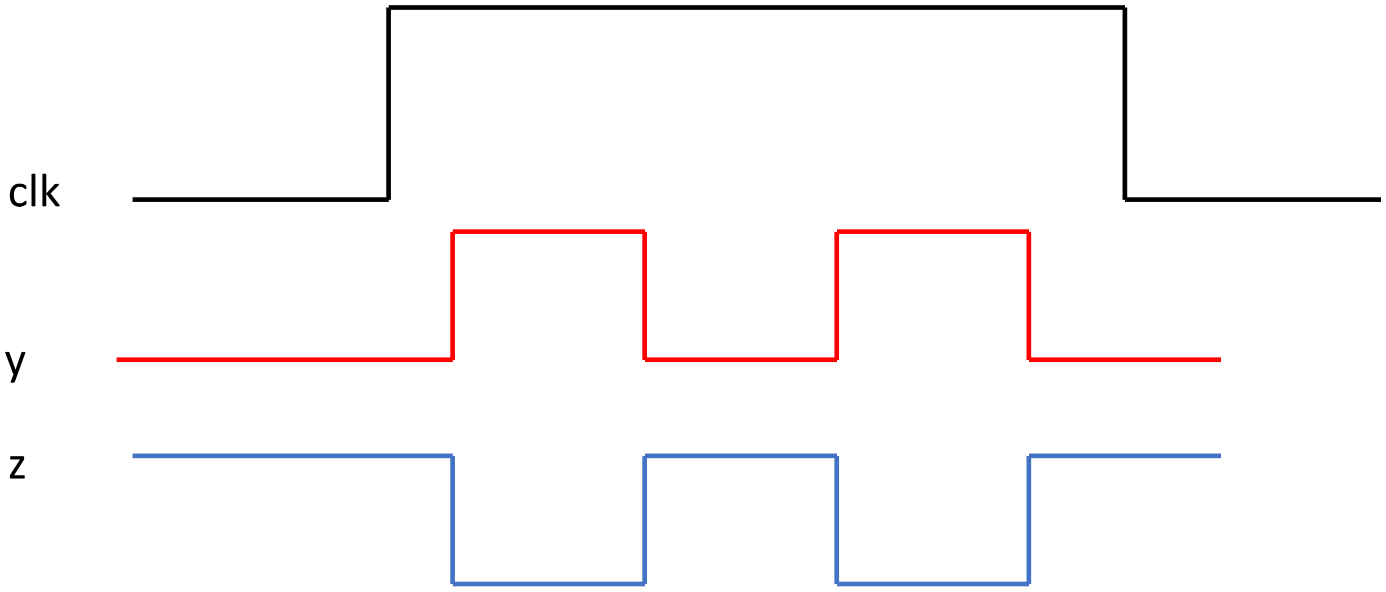
Example



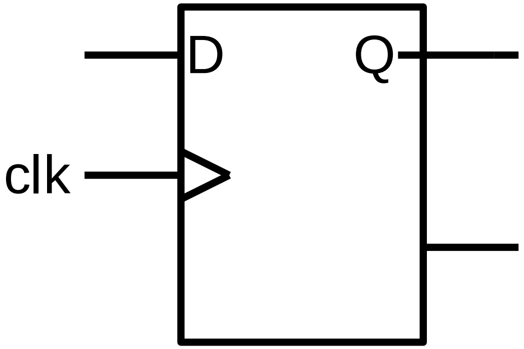
Problem with Latch

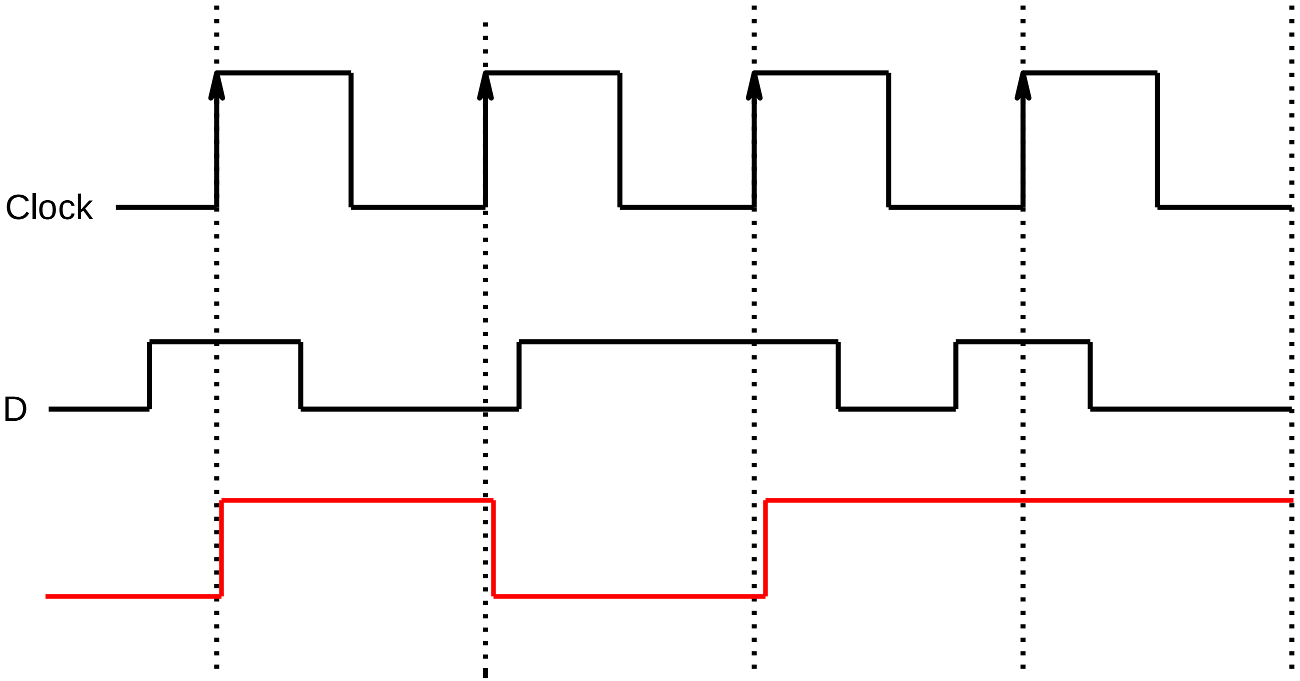


Circuits are designed with the idea there would be single change in output or memory state in single clock cycle.



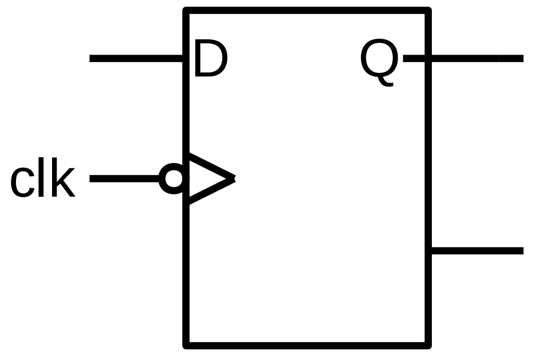
Edge Triggered Latch or Flip-flop

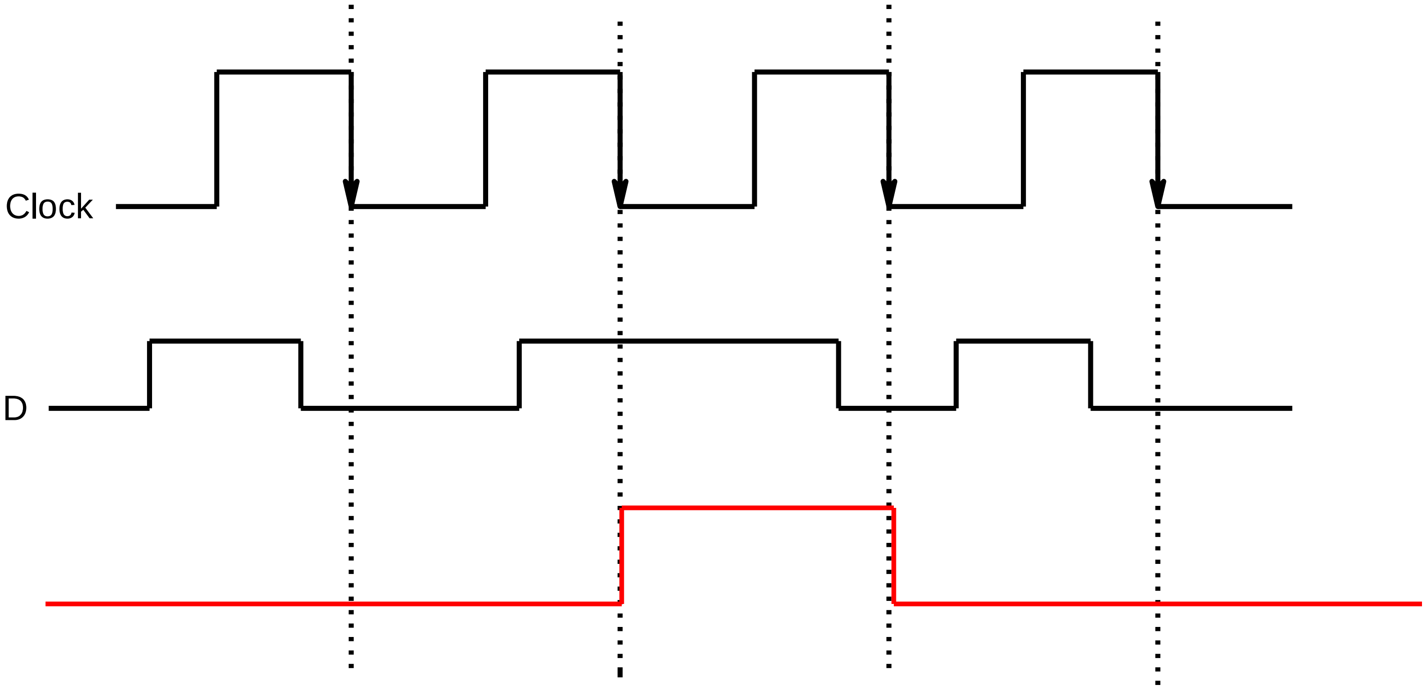




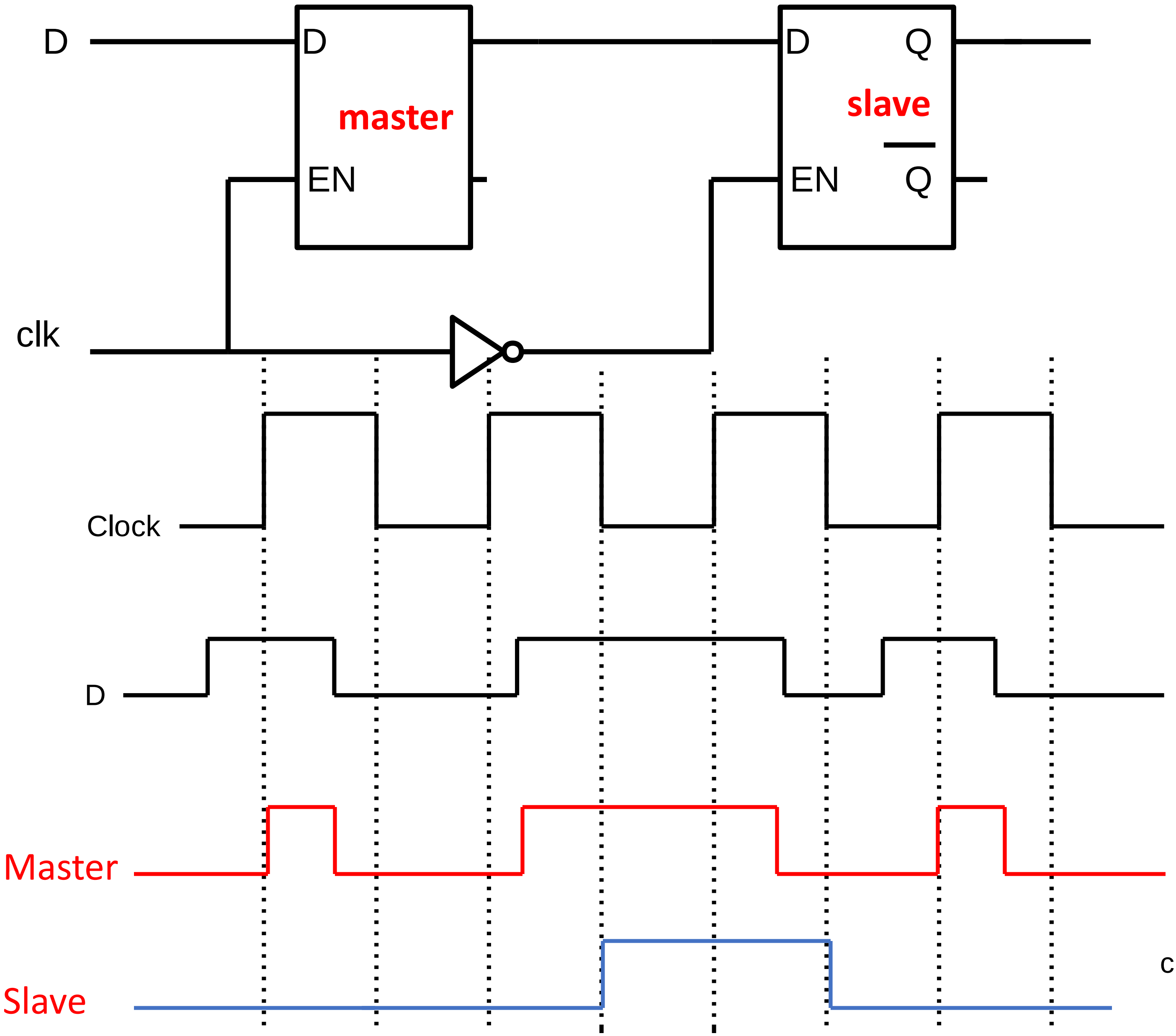
Positive edge triggered flipflop

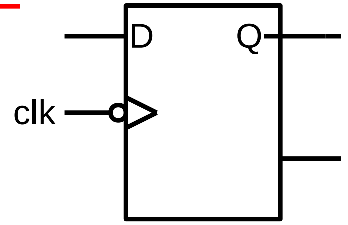
Negative Edge Triggered Latch or Flip-flop





Master-Slave D Flip-flop



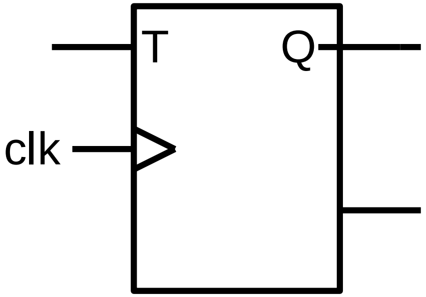


Characteristic Table vs Excitation Table

- What inputs are required for a particular state change?

Excitation table:

		Inputs
Q(t)	Q(t+1)	**T**
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

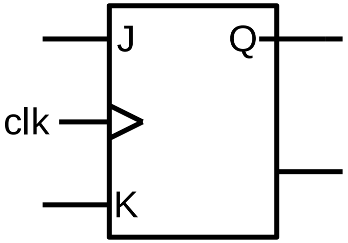


Characteristic table:

Inputs **(T)**	Q(t+1)
0	**Q(t)**
1	$\overline{\mathbf{Q(t)}}$

T	Q(t)	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table: Examples



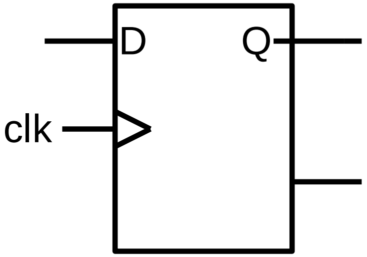
J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

Excitation table:

Inputs

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q	J	K	Q(t+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



D	Q(t+1)
0	0
1	1

Excitation table:

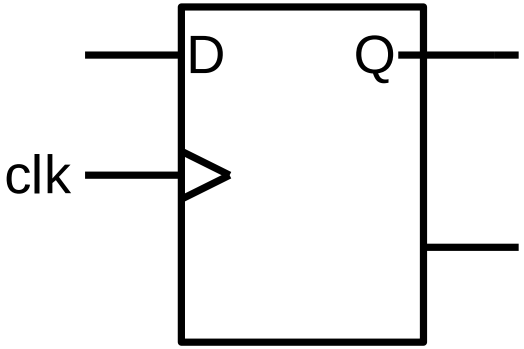
Inputs

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Implementing a T Flip Flop using D Flip Flop

- Suppose D flip flop is given, how do we design a T flip-flop?



At current instant, we have T and Q
How do we get D from signals we have

Q\T	0	1
0		1
1	1	

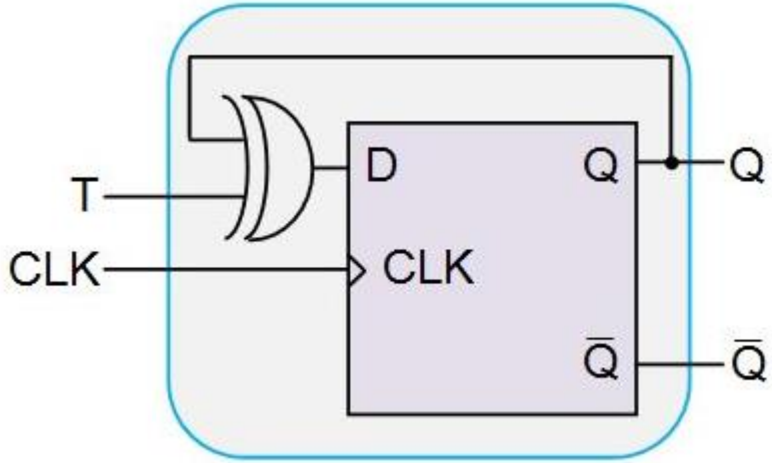
$$D = \bar{T}.Q(t) + T\overline{Q(t)}$$

$$D = T \oplus Q(t)$$

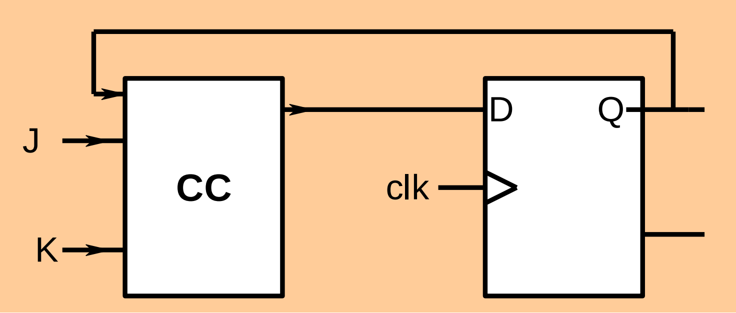
T	Q(t)	Q(t+1)	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Truth table of what we desire to make

Excitation table of what we have



D Flip Flop to JK Flip Flop



J	K	Q(t+1)	D
0	0	Q(t)	**Q(t)**
0	1	0	**0**
1	0	1	**1**
1	1	$\overline{Q(t)}$	$\overline{\mathbf{Q(t)}}$

Truth table of what we desire to make

Expanded Form

J	K	Q	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

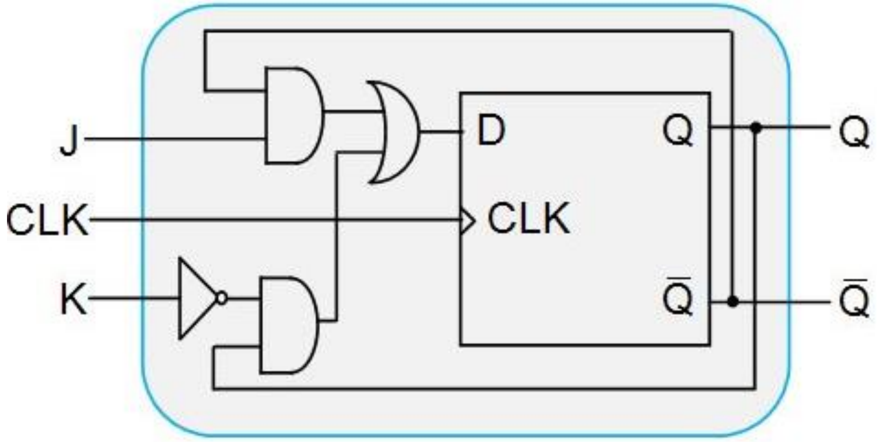
Excitation table of what we have

D
0
1
0
0
1
1
1
0

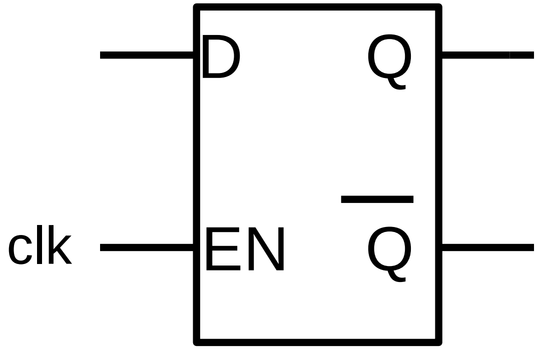
At current instant, we have J,K and Q
D from signals we have

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

$$D = \overline{Q}.J + Q.\overline{K}$$

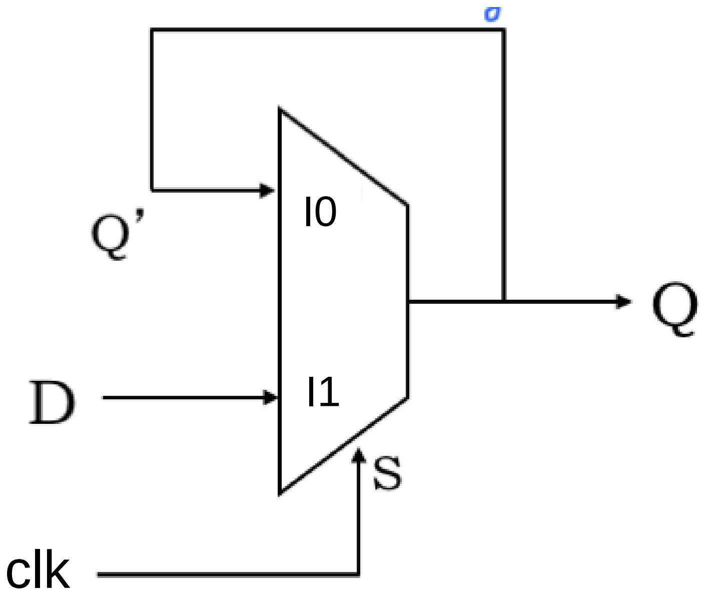


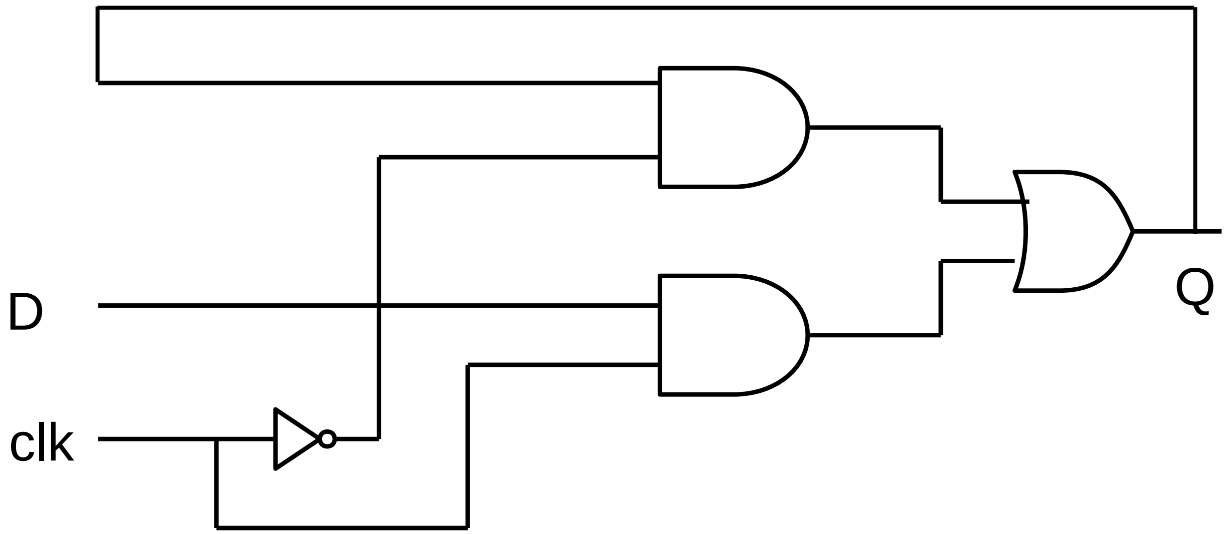
D Latch: MUX based Implementation



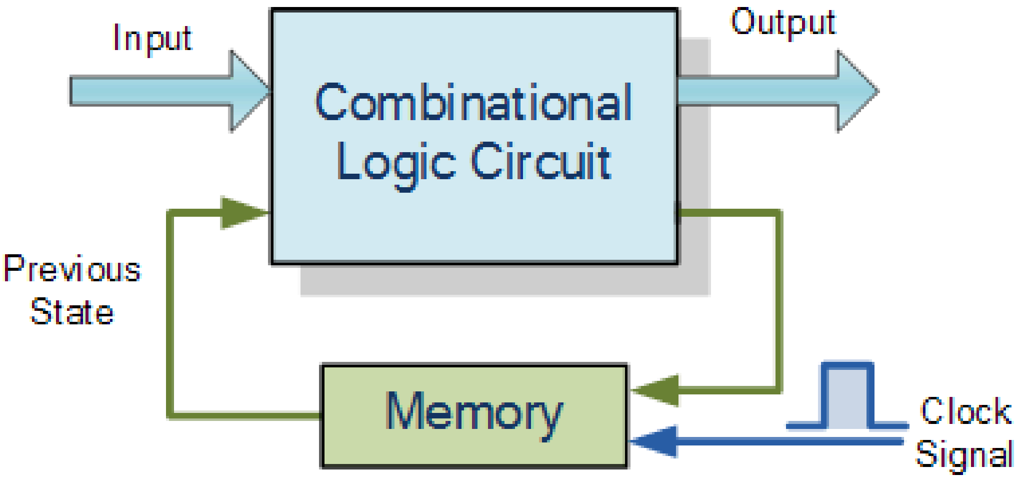
EN	D	Q	Description
0	X	Q	Hold
1	1	1	Set
1	0	0	Reset

Transparent when EN is high



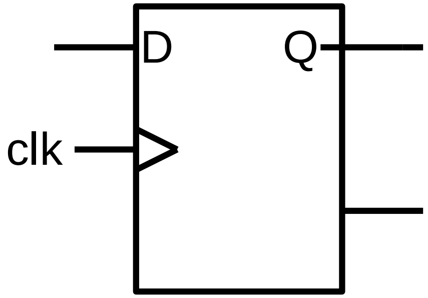


Sequential Circuits



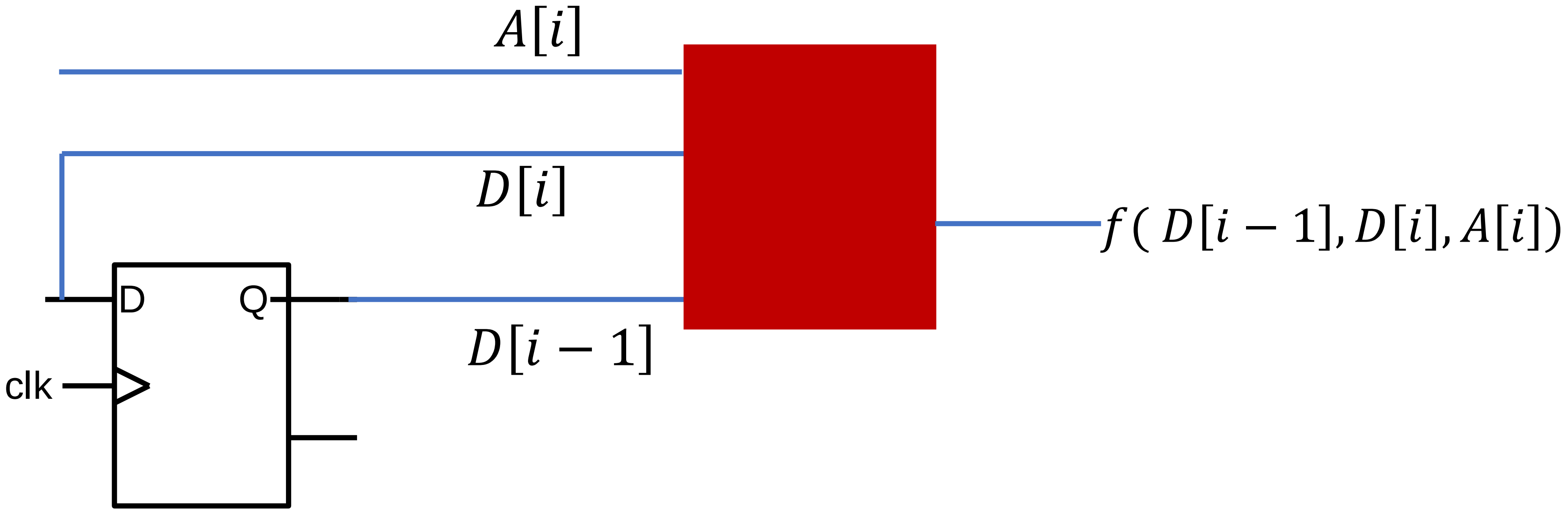
- Calculation divided into steps
- Each step is triggered by a clock
- At each step,
    - output is based on the current values of inputs and past values of inputs/outputs.
- Requires memory

D Flip Flop as 1 bit Storage/Memory

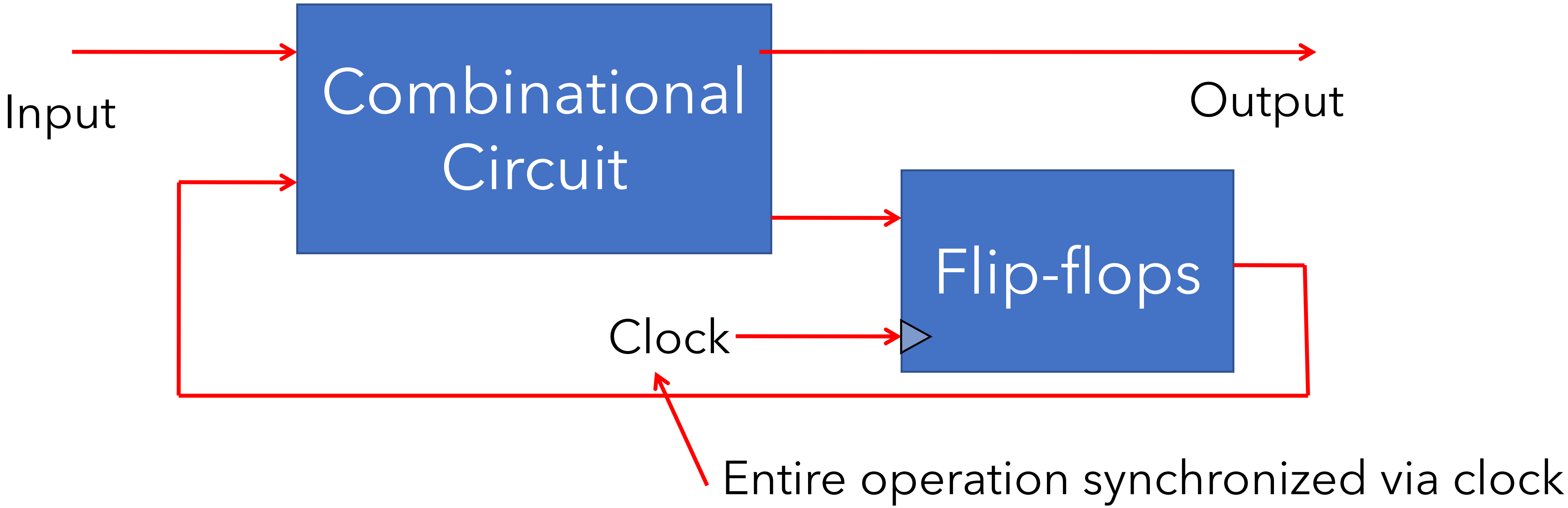


Inputs (D)	Q(t+1)
0	0
1	1

$$Q(t+1) = D$$



Synchronous Clocked Sequential Circuits

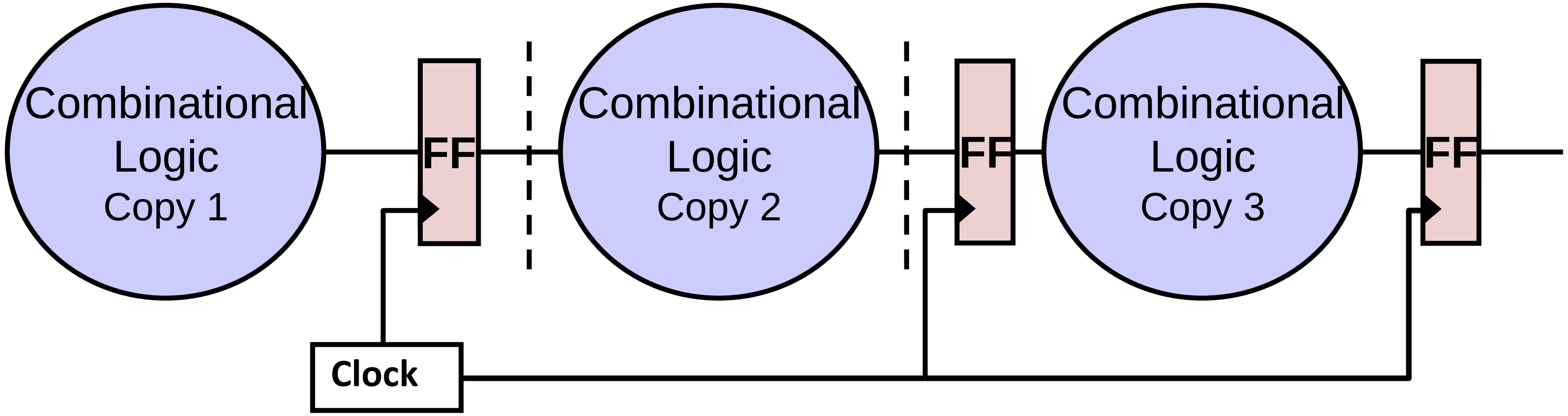


Entire operation synchronized via clock

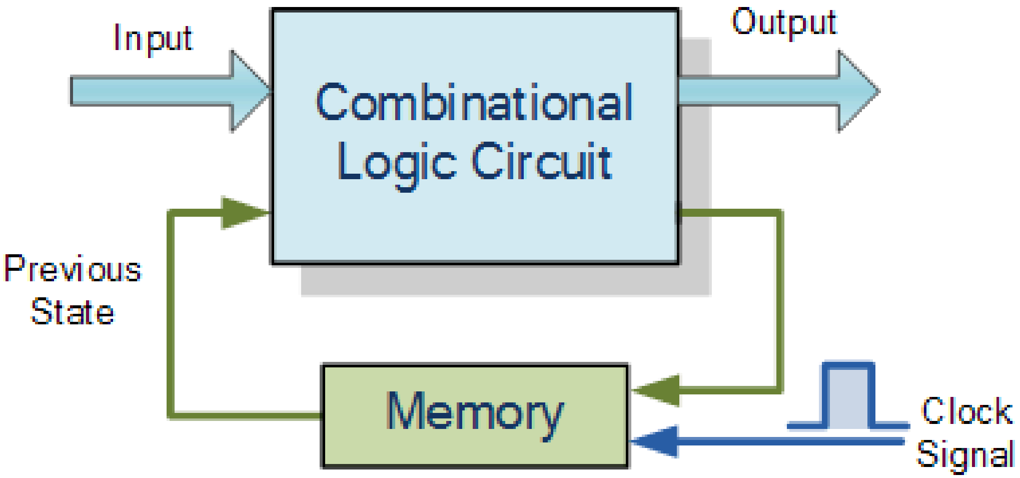
Employs signals that affect the stored value only at discrete instants of time.

Synchronization is achieved via the ***clock pulses***.

Sequential Circuit “unrolled” in Time



Sequential Circuits

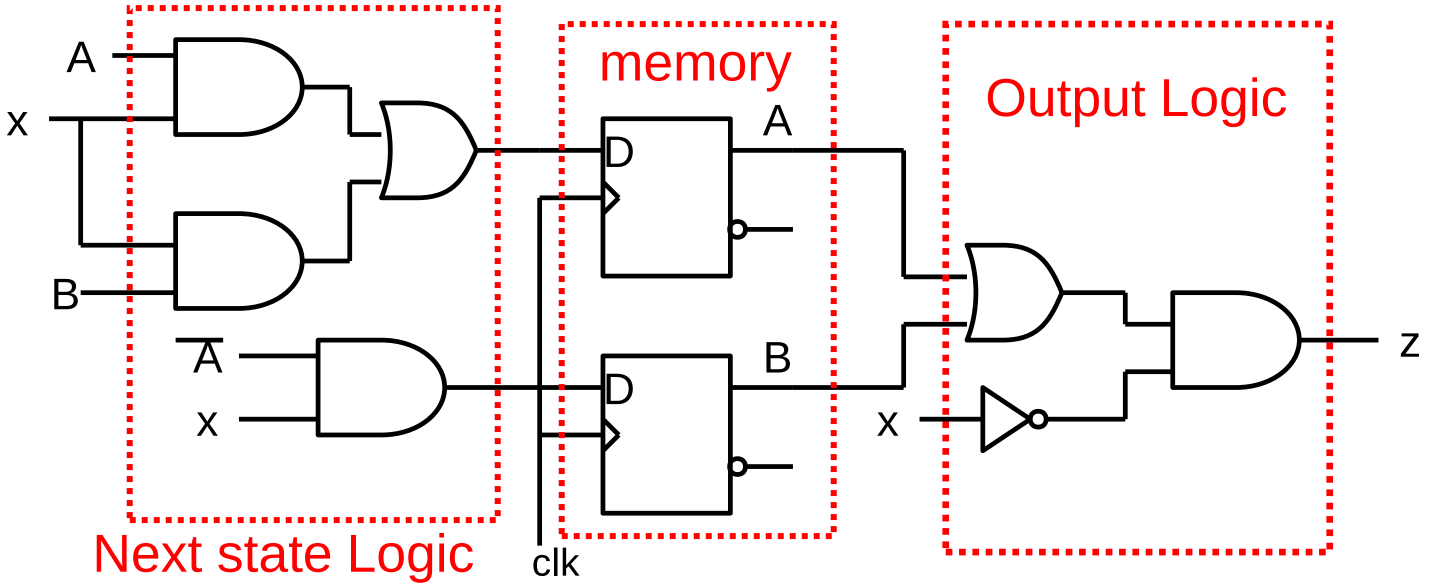


The binary information stored in the storage elements at any given time defines the ***state*** of the sequential circuit at that time

Output is a function of input as well as the present state (the stored value).

Next state is also a function of the present state and inputs.

Analyzing sequential circuits



- Output z depends on the input x and on the state of the memory (A,B)
- The memory has 2 FFs and each FF can be in state 0 or 1.
- Thus there are four possible states: AB: 00,01,10,11
- To describe the behavior of a sequential circuit, we need to show
  - How the system goes from one memory state to the next as the input changes
  - How the output responds to input in each state