Digital Logic Design

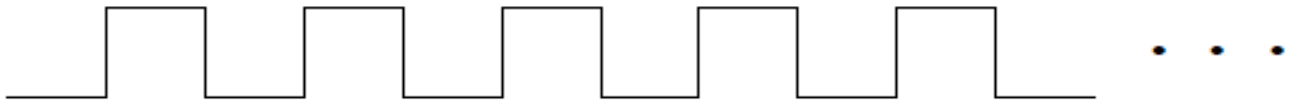
CS251

Clocks and Sequential Circuits

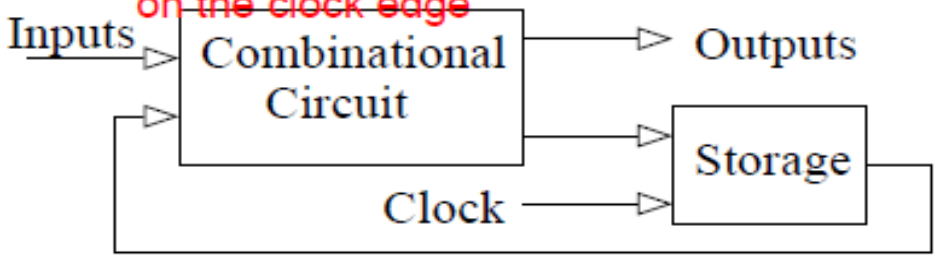
- Two types of sequential circuits:
- Synchronous: has a clock

  Memory changes only at discrete points in time

  Clock pulse: Free running signal with a fixed cycle time:



  Block diagram: We use edge triggered clocking : All changes of state occur on the clock edge



  Easier to analyze, tend to be more stable

- Asynchronous: no clock

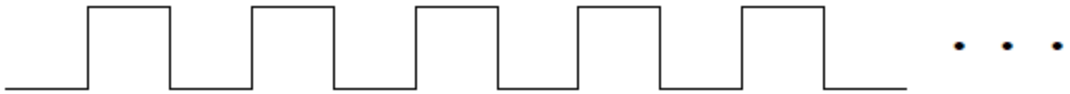
  Potentially faster and less power-hungry, but harder to design and analyze

Clocks and Sequential Circuits

- Two types of sequential circuits:
- Synchronous: has a clock

  Memory changes only at discrete points in time

  Clock pulse: Free running signal with a fixed cycle time:



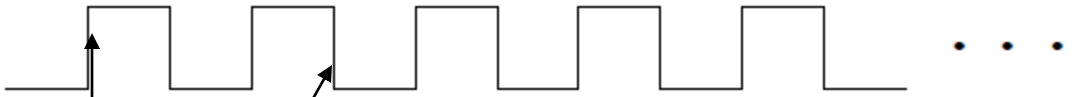
Clock Cycle: From one rising edge to another rising edge

Clocks and Sequential Circuits

- Two types of sequential circuits:
- Synchronous: has a clock

  Memory changes only at discrete points in time

  Clock pulse: Free running signal with a fixed cycle time:



Rising Edge

Falling Edge

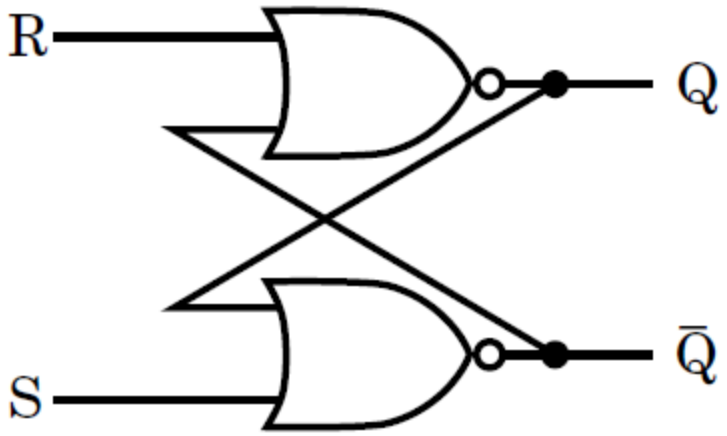
Set/Reset Latch

SR Latch with NOR gates

- SR Latch with NOR gates

Simplist Memory Element

UNCLOCKED



Values of Q represent stored state and its complement. This value is recycled through NOR gates and again we get the values of Q

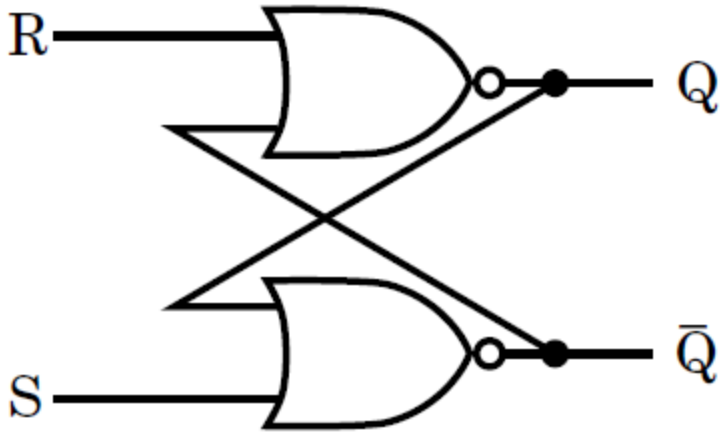
Set/Reset Latch

SR Latch with NOR gates

- SR Latch with NOR gates

Simplist Memory Element

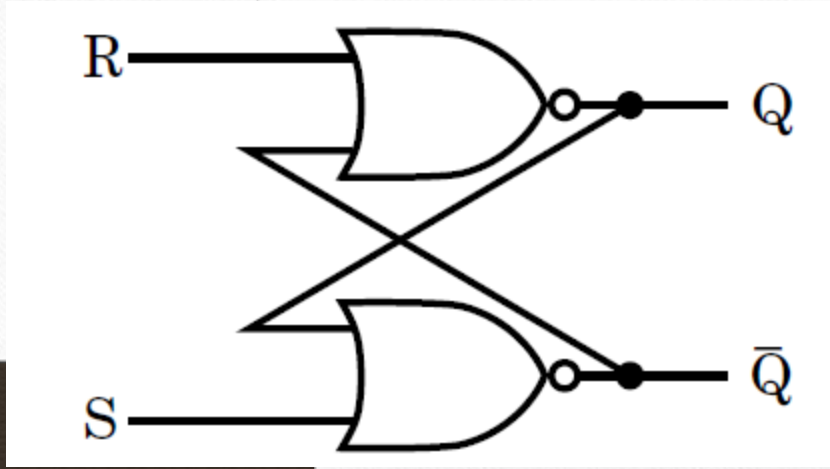
UNCLOCKED



Values of Q represent stored state and its complement. This value is recycled through NOR gates and again we get the values of Q

S, R transition	$Q, \overline{Q}$ transition
$1,0 \rightarrow 0,0$	$\rightarrow$
$0,1 \rightarrow 0,0$	$\rightarrow$
$1,1 \rightarrow 0,0$	$\rightarrow$

Maintaining the State : Or Value of Q as a memory Bit



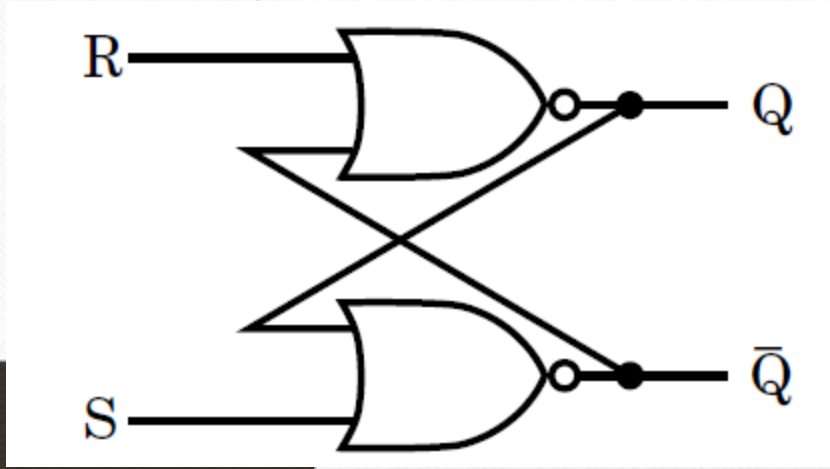
Q =1 $\bar{Q}$ =0

S =0 R=0 means, Keep the value of Q as it.

NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Maintaining the State : Or Value of Q as a memory Bit



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

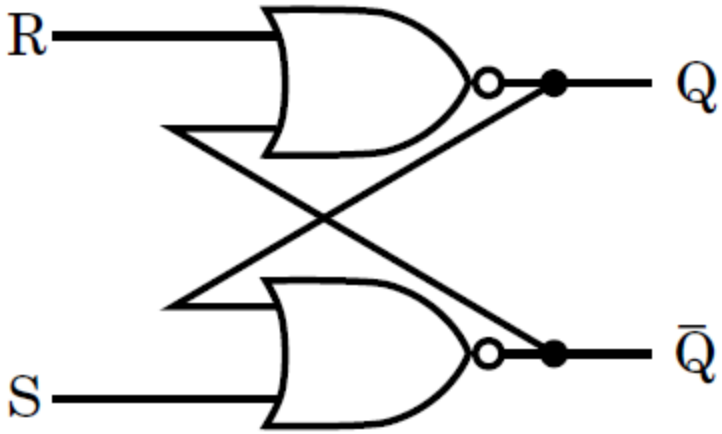
S =0 R=0 means, Keep the value of Q as it is.

Similarly when Q = 0 initially
S =0 R =0 means keep Q =0

NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Set, Reset : Change value of Q



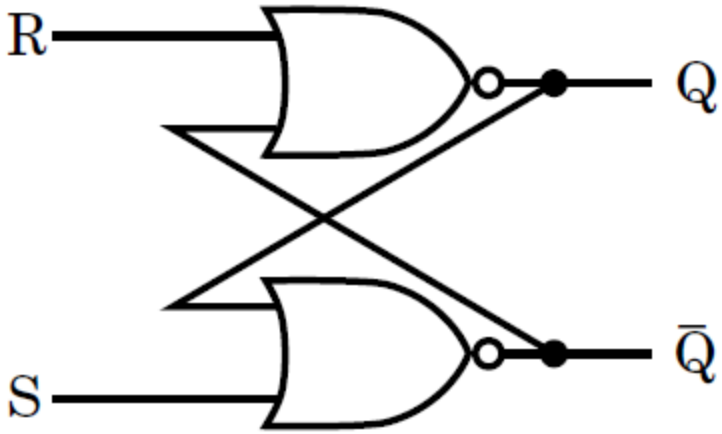
S =1 R=0 means, set Q :

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

NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Set, Reset : Change value of Q



S =1 R=0 means, set Q :

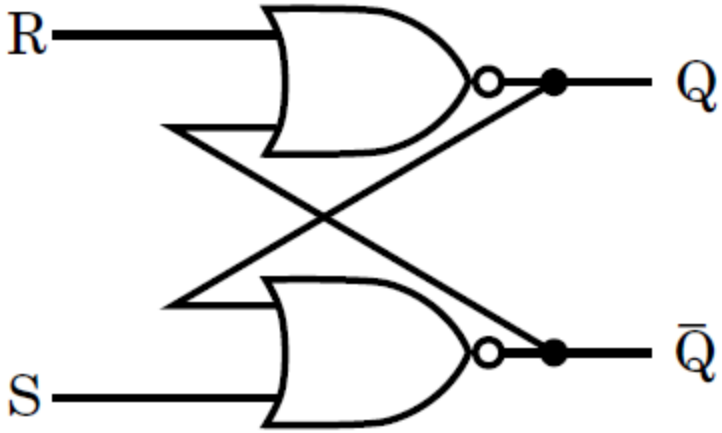
Q =1 $\bar{Q}$ =0

Even if Q was already =1. This will work

NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Set, Reset : Change value of Q



NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

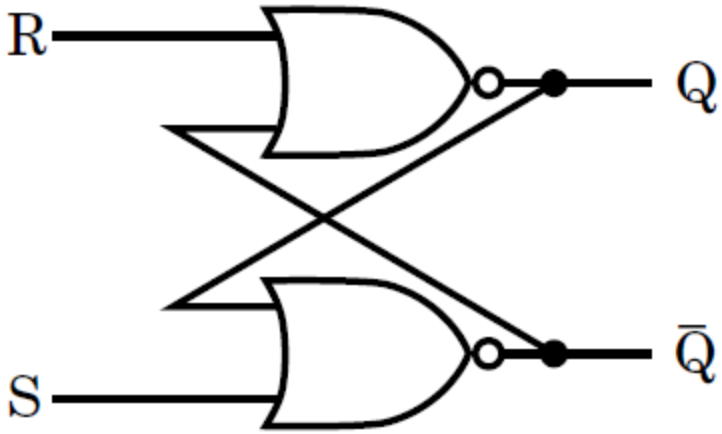
R =1 S=0 means, ***reset*** Q :

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

Even if Q was previously 0 or 1. This will work

Reset Q to 0

Set, Reset : **Change value of Q or Store Q as it is**



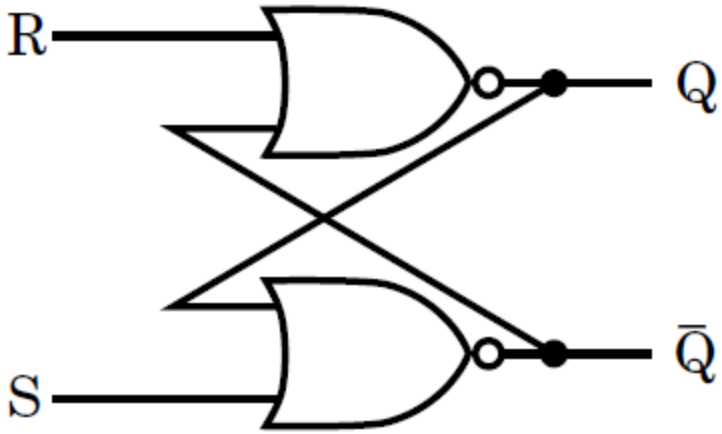
NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

S, R transition	$Q, \overline{Q}$ transition
$1,0 \rightarrow 0,0$	$1,0 \rightarrow 1,0$
$0,1 \rightarrow 0,0$	$0,1 \rightarrow 0,1$
$1,1 \rightarrow 0,0$	$\rightarrow$

What happens here?

Set, Reset : **Change value of Q or Store Q as it is**



NOR GATE:

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

S, R transition	$Q, \overline{Q}$ transition
$1,0 \rightarrow 0,0$	$1,0 \rightarrow 1,0$
$0,1 \rightarrow 0,0$	$0,1 \rightarrow 0,1$
$1,1 \rightarrow 0,0$	$\rightarrow$

What Happens when S=1 and R=1

A) Q=1 and $\bar{Q}$ = 1 then Q=0, $\bar{Q}$=1
B)Q=0 and $\bar{Q}$ = 0, then Q=1, $\bar{Q}$=1 oscillates
C)Q=0 and $\bar{Q}$ = 1
D)Q=1 and $\bar{Q}$ = 0
E)NOR Gates explode

Improvement on SR Latch : D Latch, D Flip Flop:

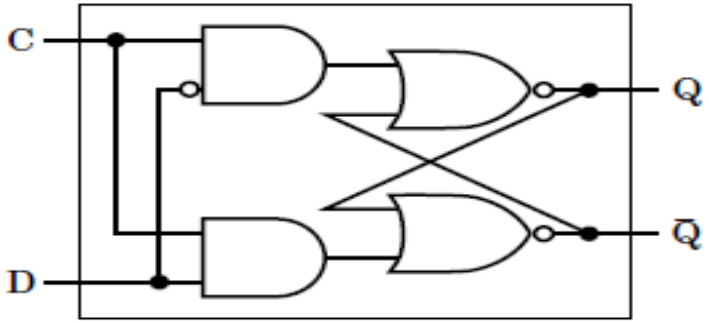
Using a CLOCK pulse to control when the State of a Memory Bit is changed.

When Clock pulse is asserted- data updates (Set,Reset) is allowed.

Otherwise Latch stays in a S=0, R=0 State

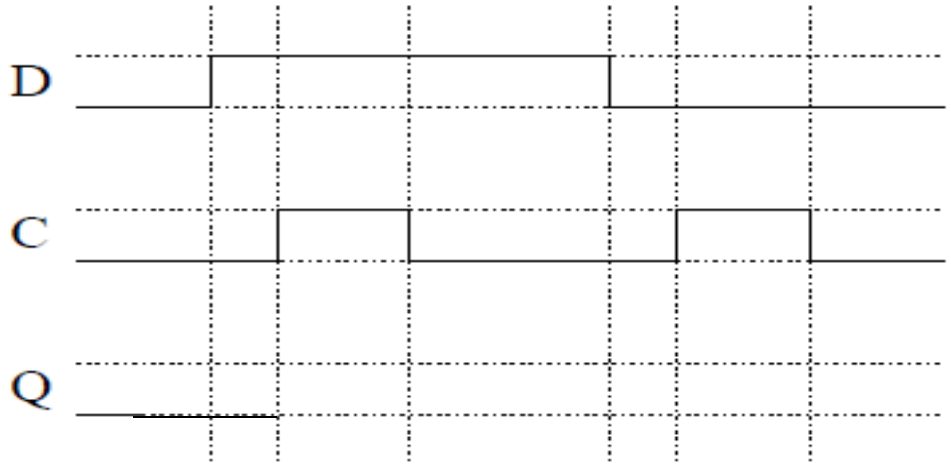
Course Notes

The D Latch

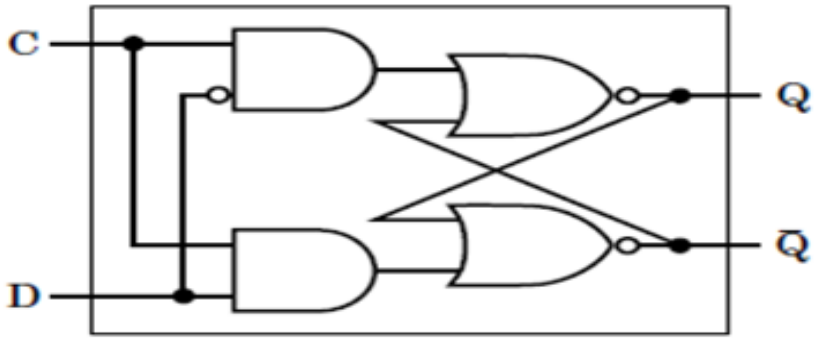


C	D	Next state of Q
0	X	No change
1	0	$Q = 0$ (Reset)
1	1	$Q = 1$ (Set)

Graphical example:

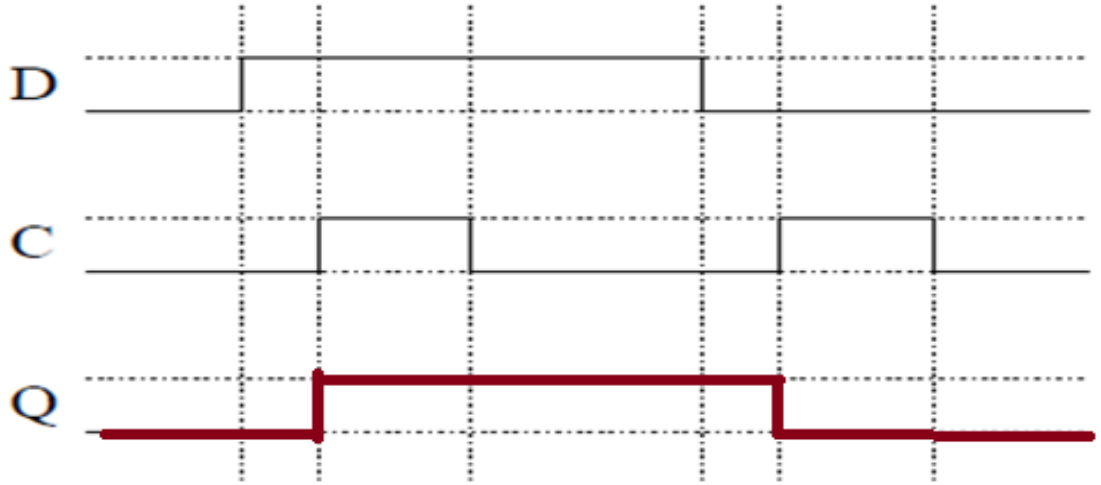


The D Latch



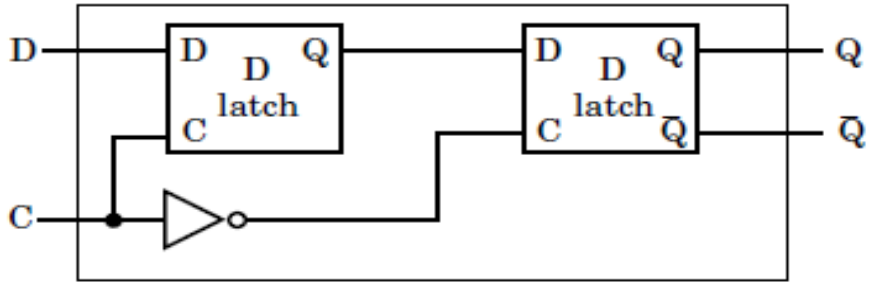
C	D	Next state of Q
0	X	No change
1	0	$Q = 0$ (Reset)
1	1	$Q = 1$ (Set)

Graphical example:

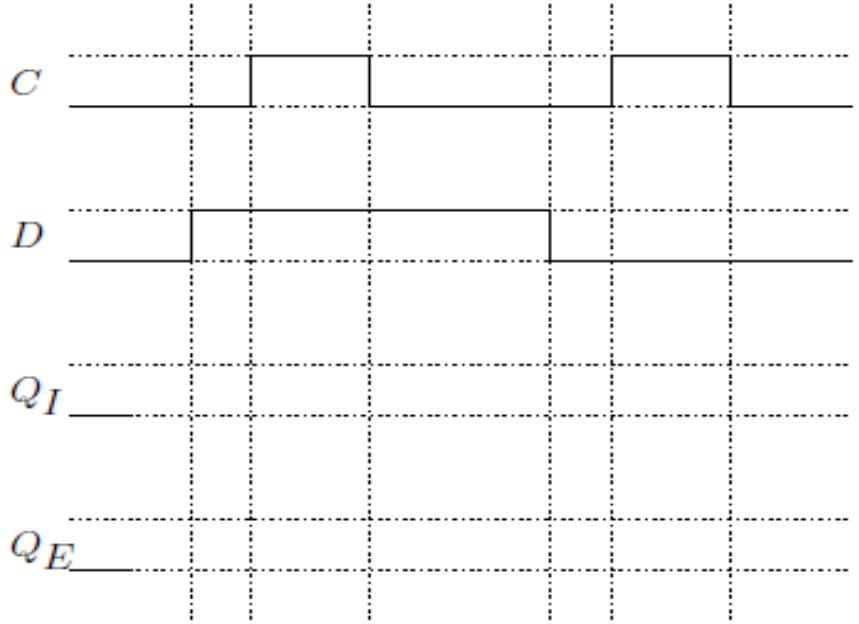


The D Flip-Flop

- We want state to be affected only at discrete points in time; a master-slave design achieves this.

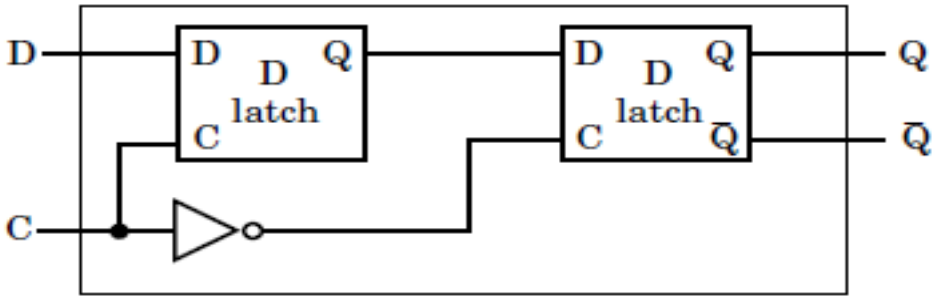


- Graphical example:

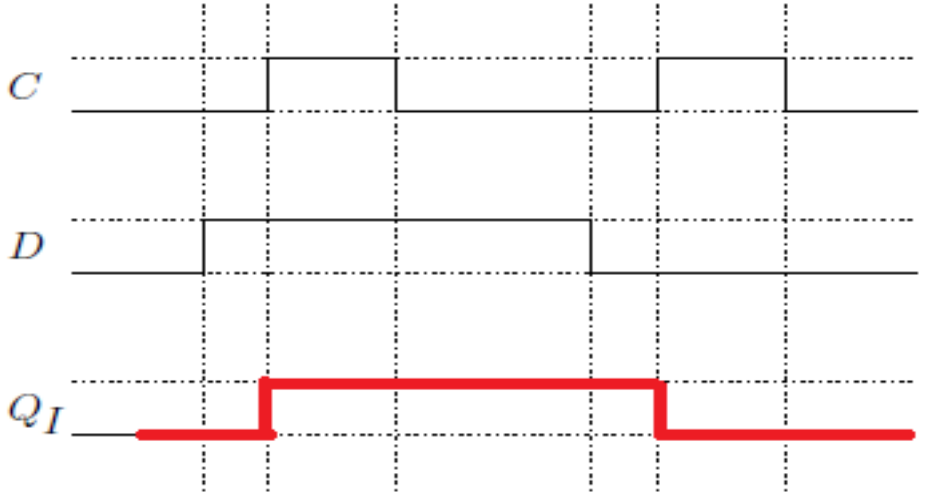


The D Flip-Flop

- We want state to be affected only at discrete points in time; a master-slave design achieves this.

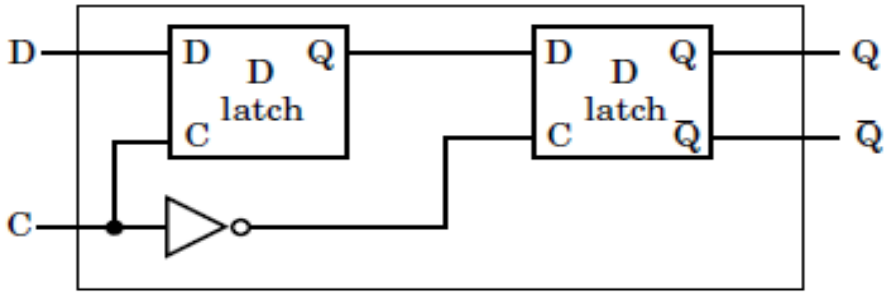


- Graphical example:

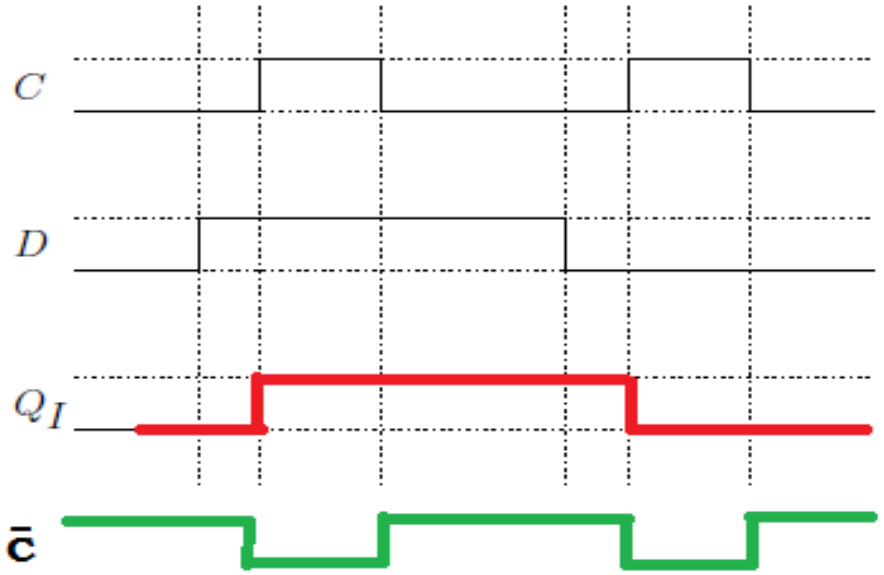


The D Flip-Flop

- We want state to be affected only at discrete points in time; a master-slave design achieves this.

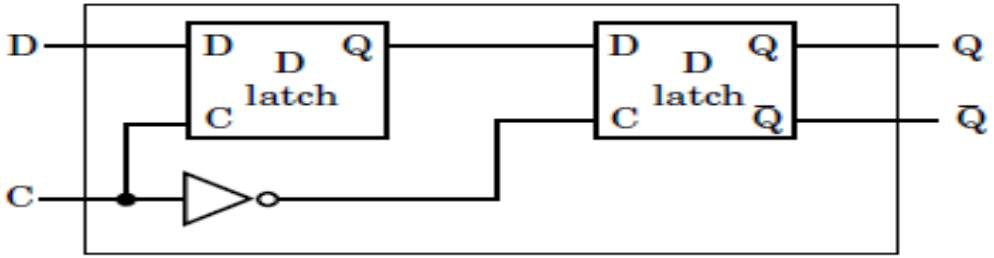


- Graphical example:

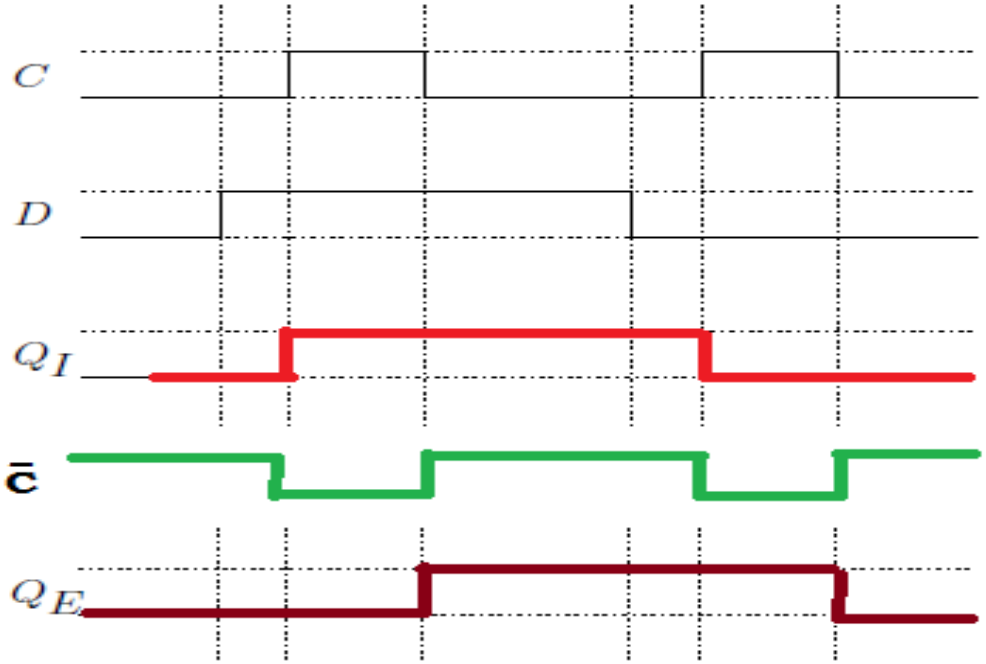


The D Flip-Flop

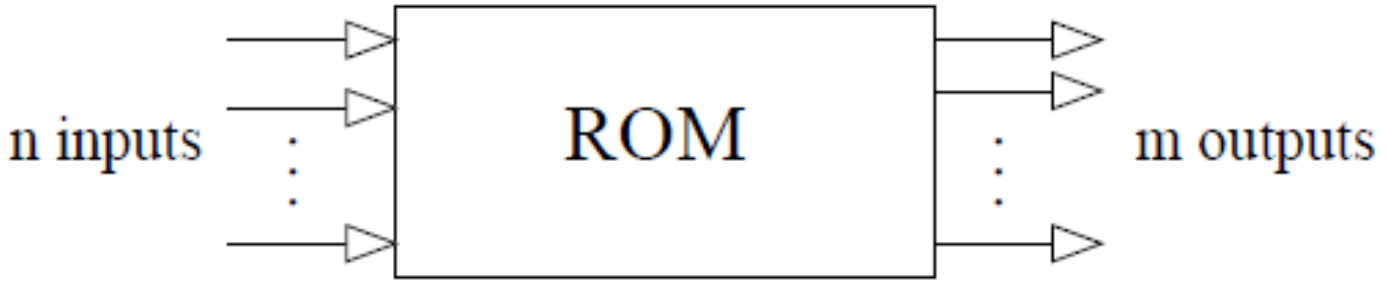
- We want state to be affected only at discrete points in time; a master-slave design achieves this.



- Graphical example:



Implementing Boolean Functions: ROMs



- Can think of ROM as table of 2^n m-bit words
- Can think of ROM as implementing m one-bit functions of n variables
- Internally, consists of a decoder plus an OR gate for each output
- Types of ROM: PROM, EPROM, EEPROM
- PLAs - simplified ROM

  Less hardware, but less flexible

Electrically Erasable Programmable Read Only Memory

RAM (Random Access Memory) this is memory that can be accessed very easily. The data is organized in such a way as to allow programs to access the data very efficiently. Main memory, easy access by CPU. Fast. Volatile storage. 'Save your work'

ROM (Read Only Memory) is memory that cannot be changed. It can only be read –Audio CD is a good example of a ROM. Cannot be changed. Computer ROM retains its contents even when system is off. Non Volatile

RAM usually can be written to and read from many times. **ROM** can only be written once, but it can be read many times.

Hard Drives: **Made up of several rigid metal, glass or ceramic disks**

PLA/ ROM

We saw Truth Tables implement logic functions:

Functions of the form

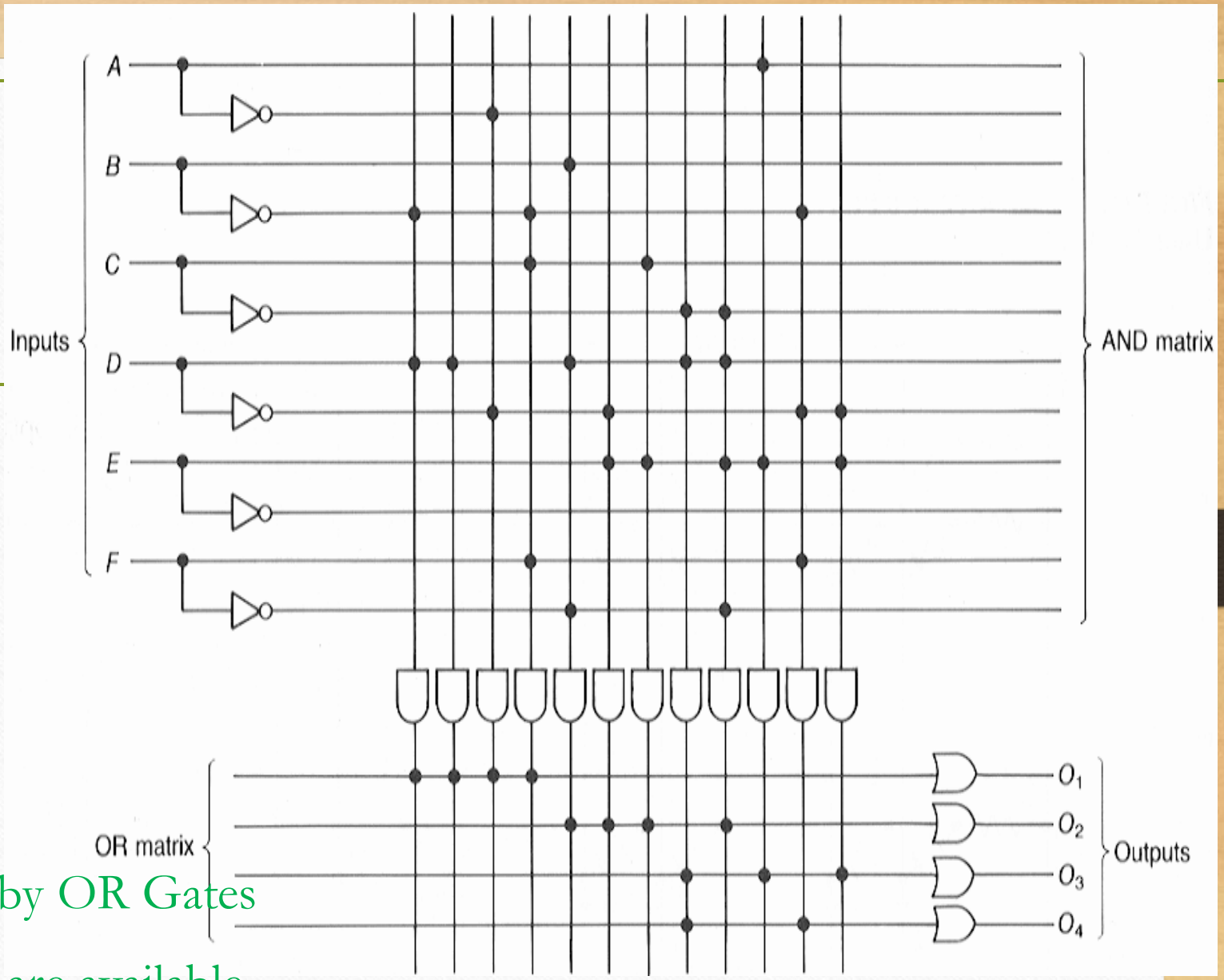
* Product of sums

* Layer of AND Gates followed by Layer of OR Gates

ROM/PLA:

Internally AND Gates followed by OR Gates

Every input and its complement are available



http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/cwl3/article2.html