

NEC 304

STLD

Lecture 19

Sequential Circuits: Latches

Rajeev Pandey

Department Of ECE

rajeevvce2007@gmail.com

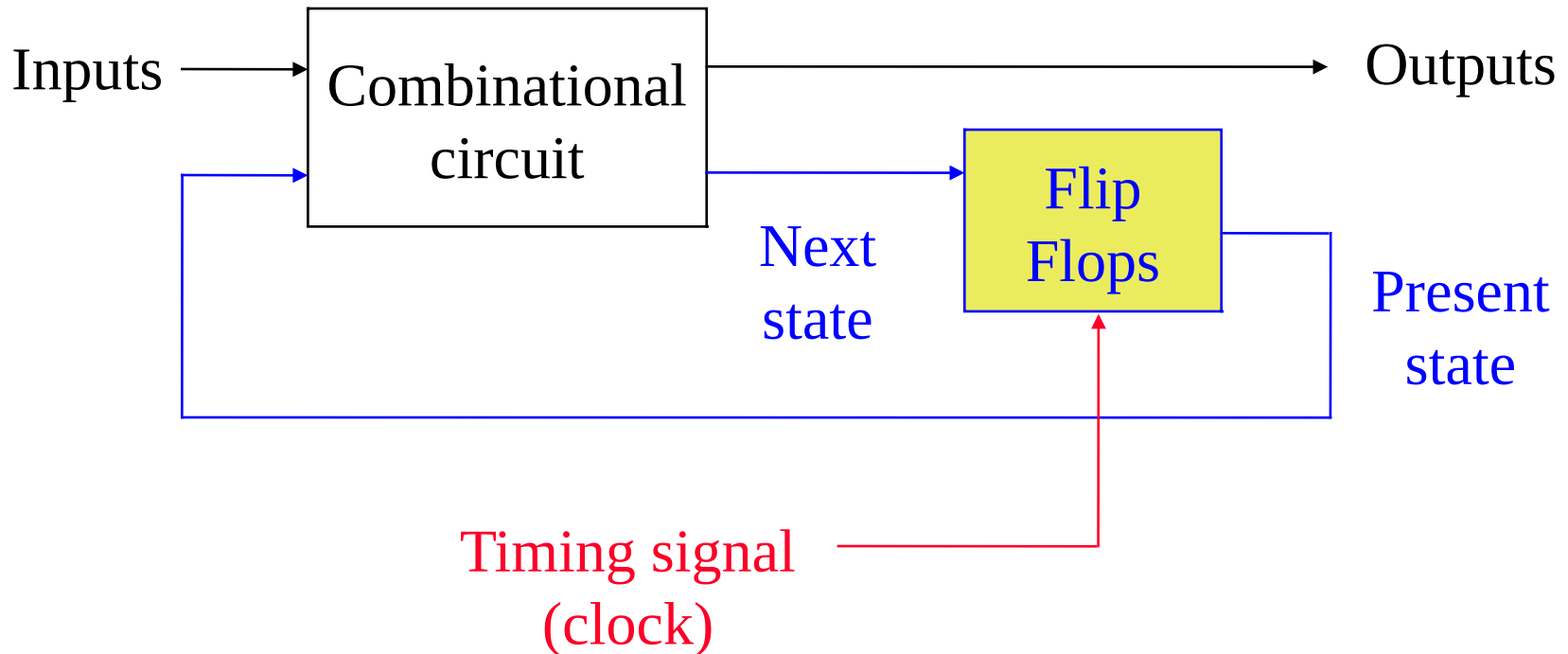
Overview

- Circuits require memory to store intermediate data
- Sequential circuits use a **periodic** signal to determine when to store values.
 - A **clock** signal can determine storage times
 - **Clock** signals are periodic
- Single bit storage element is a **flip flop**
- A basic type of flip flop is a **latch**
- Latches are made from logic gates
 - NAND, NOR, AND, OR, Inverter

The story so far ...

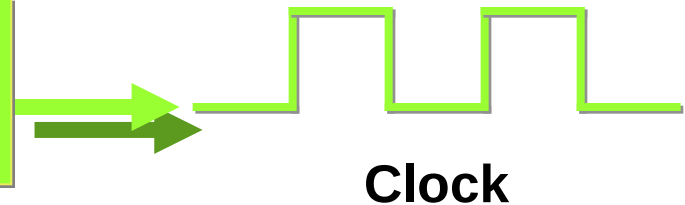
- **Logical operations which respond to combinations of inputs to produce an output.**
 - **Call these** combinational logic **circuits.**
- **For example, can add two numbers. But:**
 - **No way of adding two numbers, then adding a third (a sequential operation);**
 - **No way of remembering or storing information after inputs have been removed.**
- **To handle this, we need sequential logic capable of storing intermediate (and final) results.**

Sequential Circuits



Clock

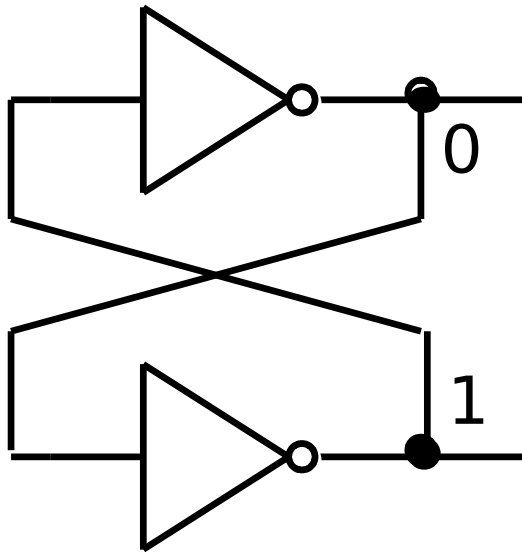
a periodic external event (input)



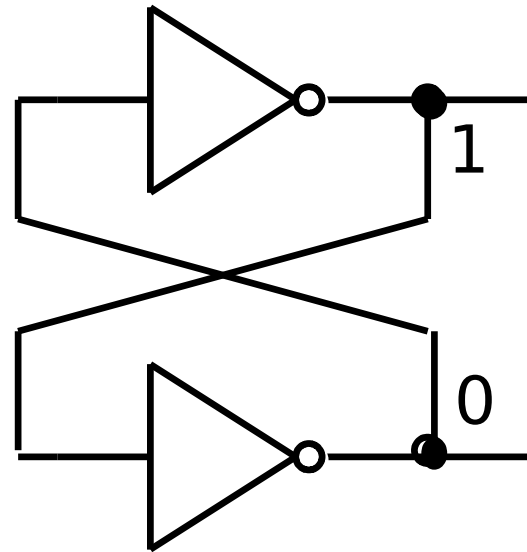
synchronizes when current state changes happen
keeps system well-behaved
makes it easier to design and build large systems

Cross-coupled Inverters

- A stable value can be stored at inverter outputs

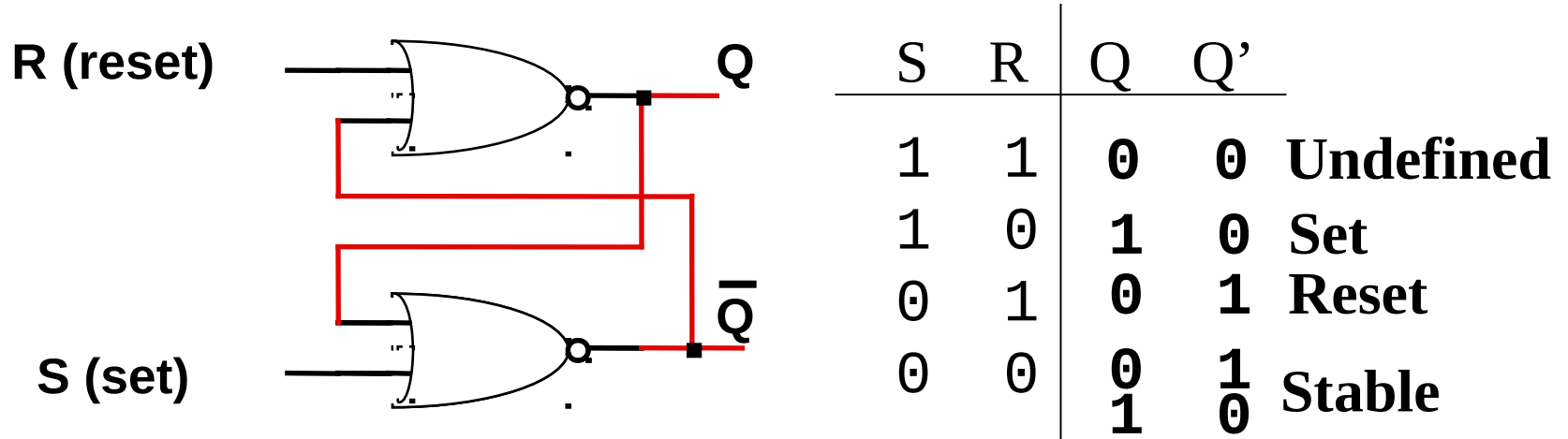


State 1



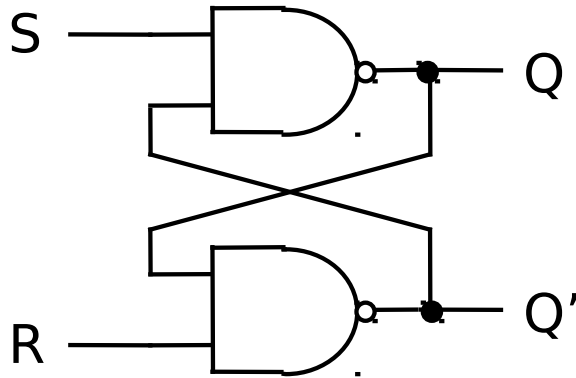
State 2

S-R Latch with NORs



- S-R latch made from **cross-coupled** NORs
- If $Q = 1$, set state
- If $Q = 0$, reset state
- Usually $S=0$ and $R=0$
- $S=1$ and $R=1$ generates unpredictable results

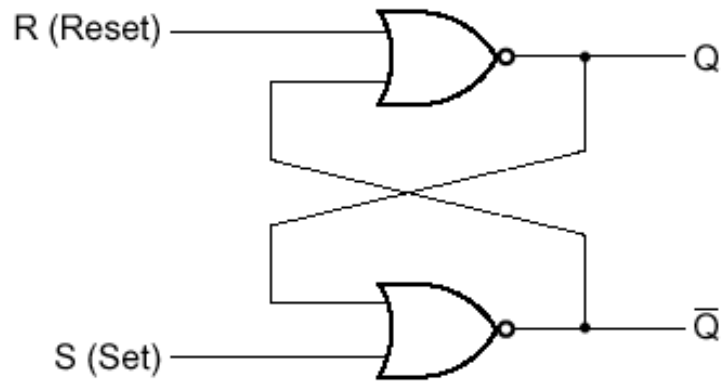
S-R Latch with NANDs



S	R	Q	Q'	
0	0	1	1	Disallowed
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	1	Store
		1	0	

- Latch made from **cross-coupled** NANDs
- Sometimes called S'-R' latch
- Usually S=1 and R=1
- S=0 and R=0 generates unpredictable results

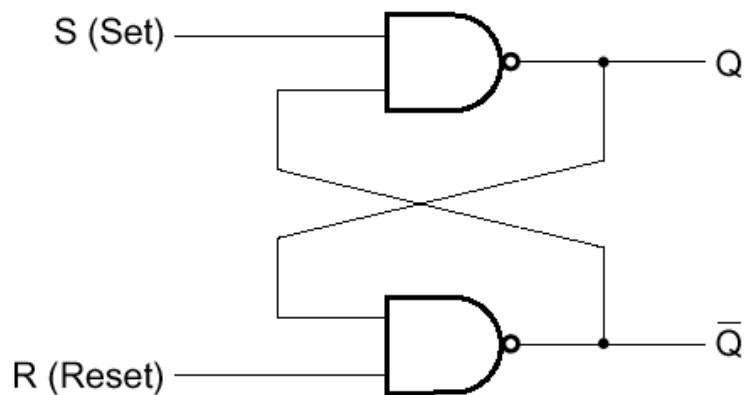
S-R Latches



(a) Logic diagram

S	R	Q	\bar{Q}	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table

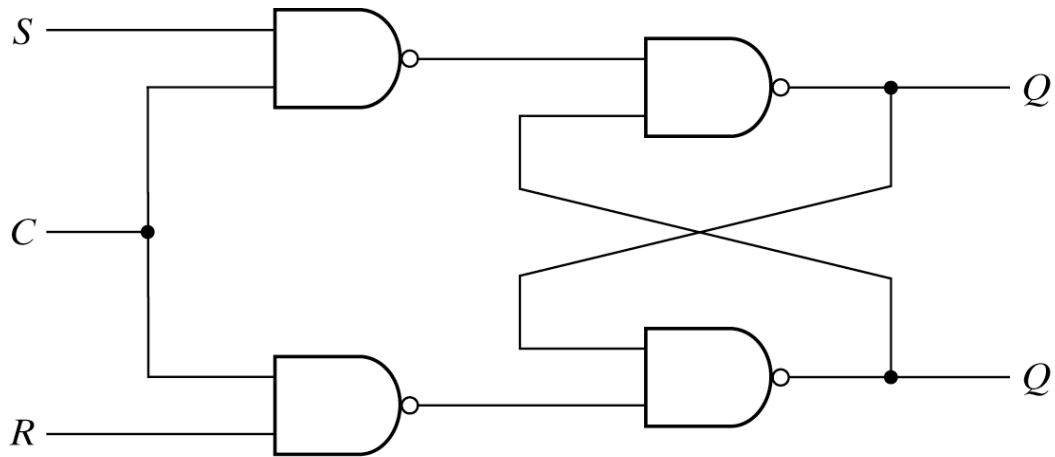


(a) Logic diagram

S	R	Q	\bar{Q}	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

S-R Latch with control input



(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; Reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(b) Function table

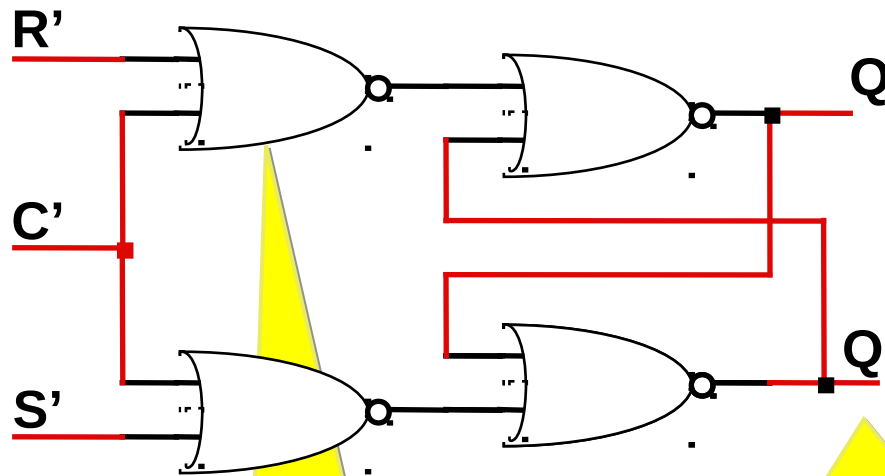
Fig. 5-5 SR Latch with Control Input

- Occasionally, desirable to avoid latch changes
- $C = 0$ disables all latch state changes
- Control signal **enables** data change when $C = 1$
- Right side of circuit same as ordinary S-R latch.

NOR S-R Latch with Control Input

Latch is **level-sensitive**, in regards to C

Only stores data if $C' = 0$



Latch operation
enabled by

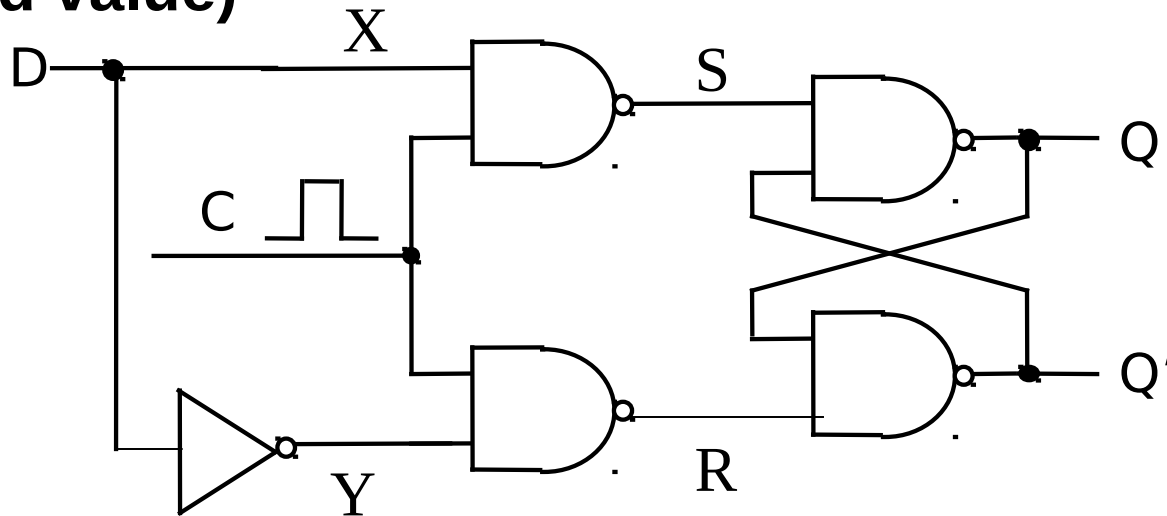
C

Input sampling
enabled by gates

Outputs change
when C is low:
RESET and SET
Otherwise: HOLD

D Latch

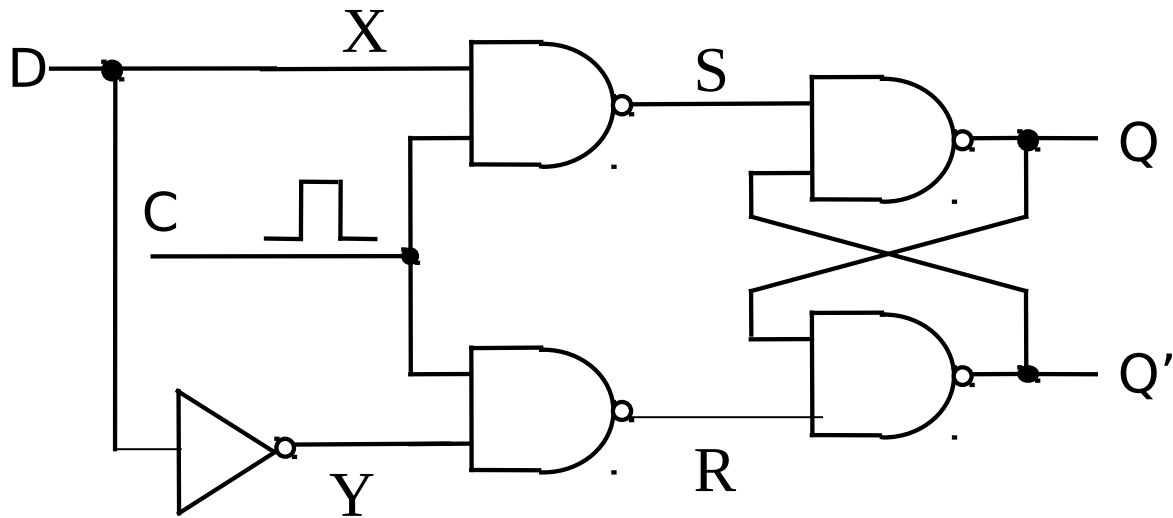
- Q₀ indicates the **previous state** (the previously stored value)



D	C	Q	Q'
0	1	0	1
1	1	1	0
X	0	Q ₀	Q ₀ '

X	Y	C	Q	Q'	
0	0	1	Q ₀	Q ₀ '	Store
0	1	1	0	1	Reset
1	0	1	1	0	Set
1	1	1	1	1	Disallowed
X	X	0	Q ₀	Q ₀ '	Store

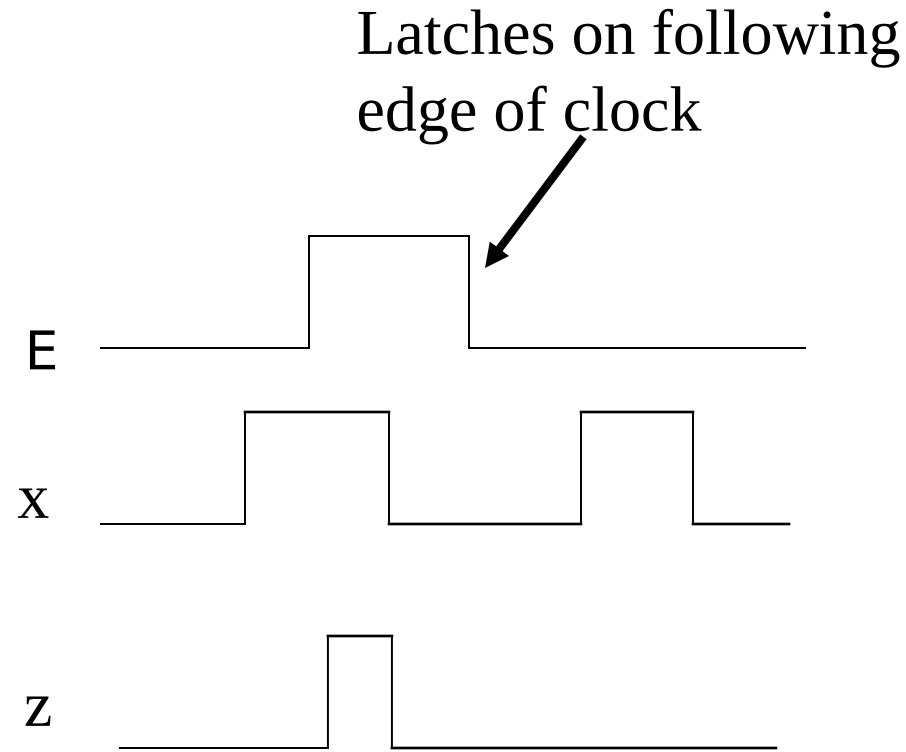
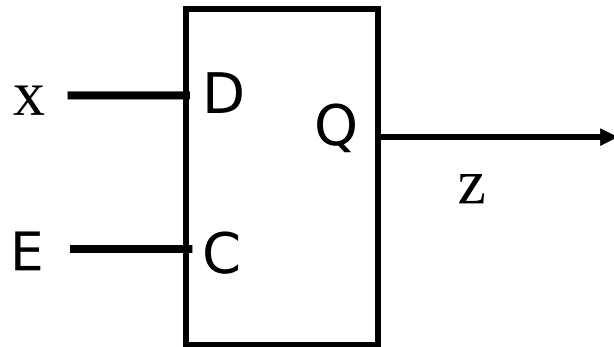
D Latch



D	C	Q	Q'
0	1	0	1
1	1	1	0
X	0	Q_0	Q_0'

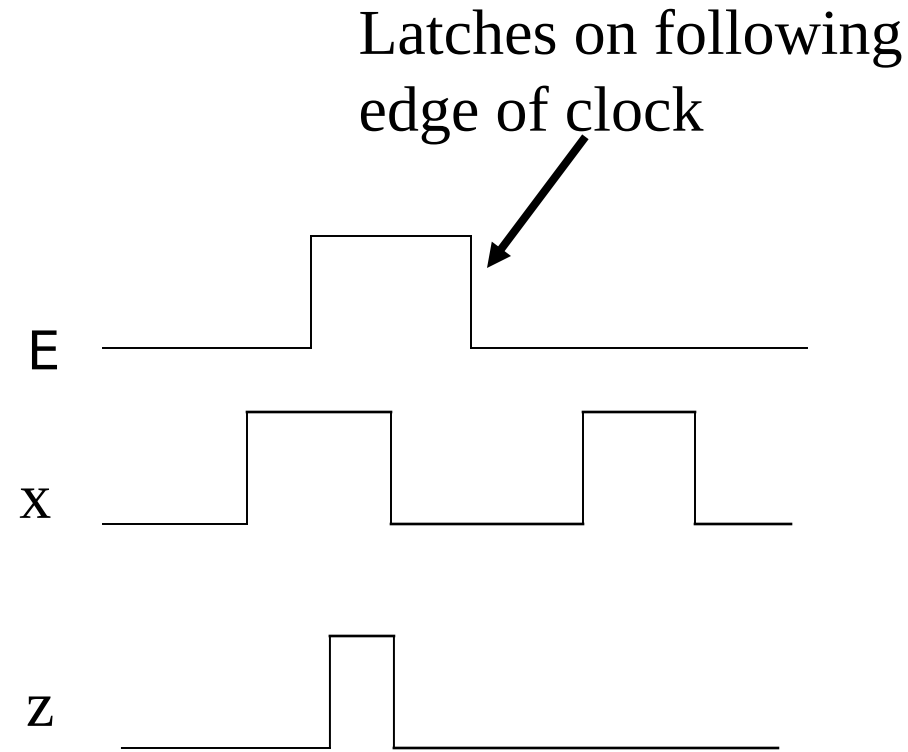
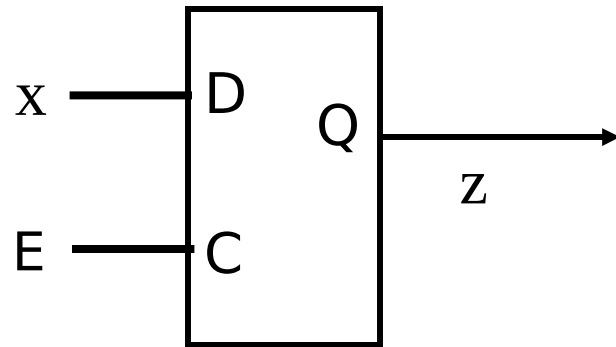
- Input value **D** is passed to output **Q** when **C** is high
- Input value **D** is ignored when **C** is low

D Latch



- **Z** only changes when **E** is high
- If **E** is high, **Z** will follow **X**

D Latch



- The **D latch** stores data indefinitely, regardless of input D values, if $C = 0$
- Forms basic storage element in computers

Symbols for Latches

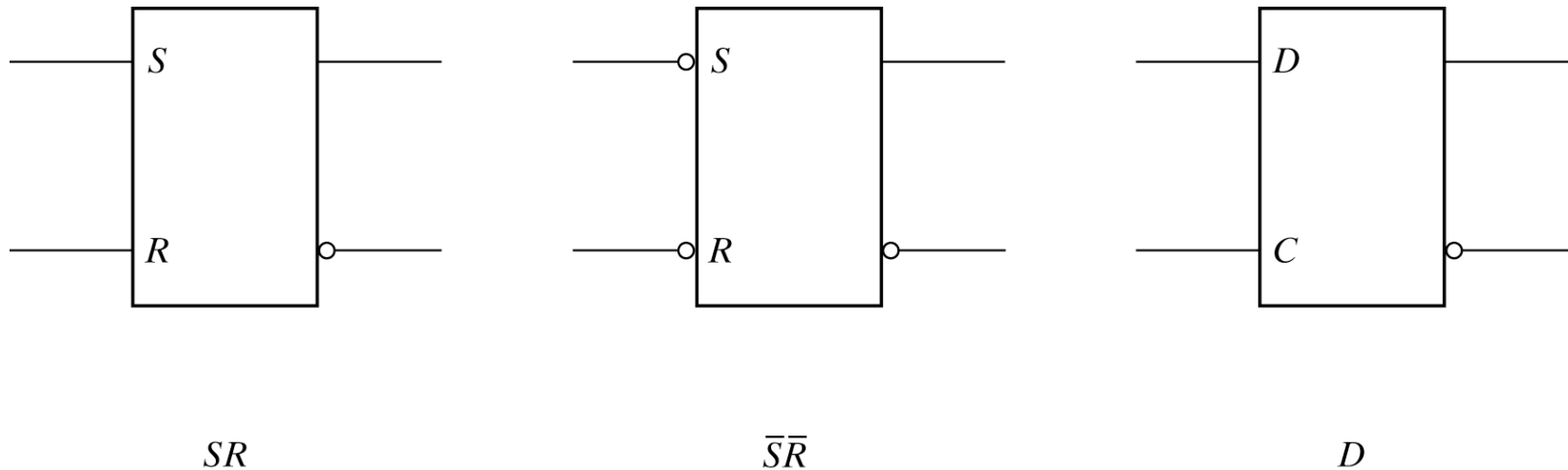


Fig. 5-7 Graphic Symbols for Latches

- **SR latch is based on NOR gates**
- **S'R' latch based on NAND gates**
- **D latch can be based on either.**
- **D latch sometimes called transparent latch**

Summary

- Latches are based on combinational gates (e.g. NAND, NOR)
- Latches store data even after data input has been removed
- S-R latches operate like cross-coupled inverters with control inputs (S = set, R = reset)
- With additional gates, an S-R latch can be converted to a D latch (**D** stands for **data**)
- D latch is simple to understand conceptually
 - When **C = 1**, data input **D** stored in latch and output as **Q**
 - When **C = 0**, data input **D** ignored and previous latch value output at **Q**
- Next time: **more storage elements!**