Topic 8: Sequential Circuits

Readings :

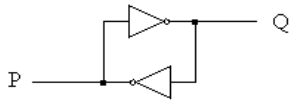
Patterson & Hennesy, Appendix B.4 - B.6

Goals

- Basic Principles behind ***Memory Elements***
- ***Clocks***
- ***Applications*** *of sequential circuits*
- Introduction to the concept of the ***State Machine***

Bistable Devices

Consider the following element



This device exhibits two stable states (bistable)

If P=0(Reset), then Q=1(Set).

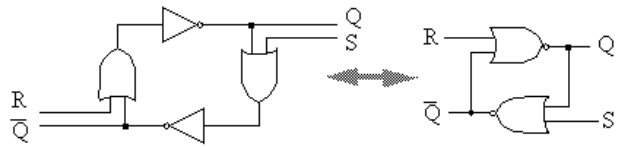
If P=1(Set), then Q=0(Reset).
So ,

$$P = \overline{Q}$$

It is able to retain the two states, Set & Reset indefinitely.

This is a memory cell, but there is no way to set it!

Solution : Add OR gates to make it set-able.



- The circuit on the right is called an S-R Latch (Set-Reset Latch)
- The value of the output Q and its complement represent the stored state.
- R, S are the inputs of the Latch

S-R Latches

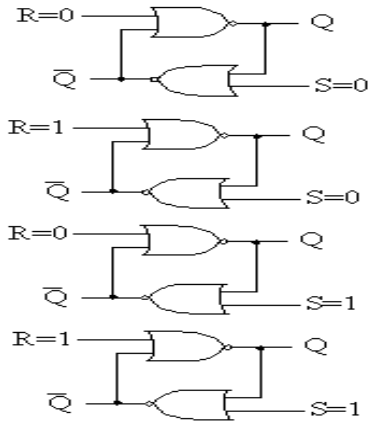
Four possible cases (inputs) :

RS = 00, 01, 10, 11

Reminder : For a NOR gate, if any input is 1 then the output is 0.

Truth table

R	S	Q	Q'
0	0	1/0	0/1
0	1	1	0
1	0	0	1
1	1	0	0



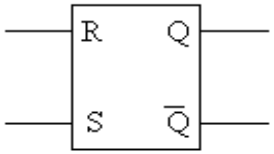
S-R Latches

RS Latch summary

- Q is value stored in memory latch
- RS = 00 is "Hold" condition
- RS = 01 is " Set" condition (force Q=1)
- RS = 10 is " Reset" condition (force Q=0)
- RS = 11 is " Forbidden"

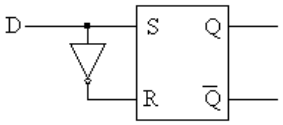
  It can lead to incorrect operation : oscillation or "metastability".
  Q is not the complement of Q' .

Symbol



Problems and Solutions

Problem : What to do about R=S=1
Solution : Don't let it happen. Tie together inputs with an inverter.

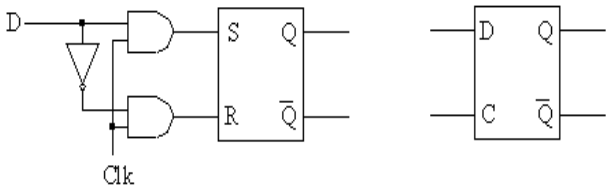


Now we cannot get R=S=1, but :

Problem : We cannot get R=S=0 ("Hold" condition)
Solution : Use "Clock" input.

Result : Clocked D-latch (No clock = hold)

D-Latch



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

Necessary topic before we proceed :
"Clocks"

Definition :

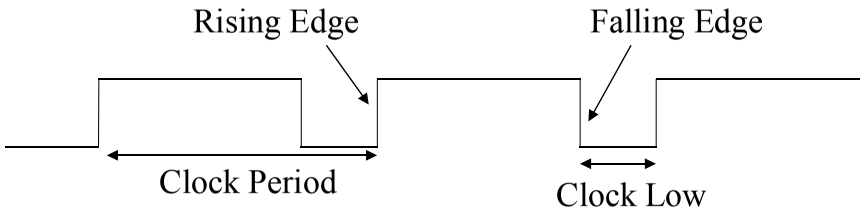
Clock is a free-running signal with a fixed ***Cycle Time*** or ***Clock Period.***

Clock Frequency = 1/Cycle Time

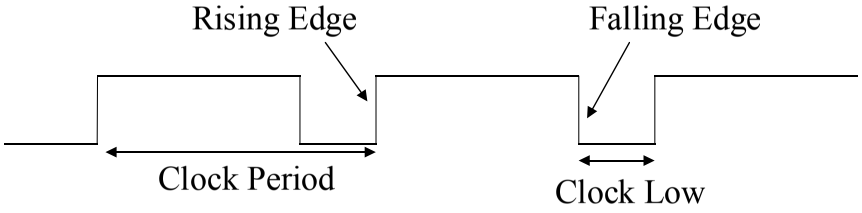
The clock period is divided into two portions:

- Clock is "High"
- Clock is "Low"

- Rising Edge : Transition from Low to High
- Falling Edge: Transition from High to Low



•The duration of "High" and "Low" parts of the clock cycle need not be the same.
•Duty cycle : Fraction of the Period, when the signal is considered active.
•Edge Triggered clocking : All state changes occur at a clock edge.



Edge-Triggered Methodology :

Either the Rising or the Falling Edge of the Clock is considered ***Active = Causes state changes to occur.***

Constraint of Clocked or ***Synchronous Systems*** **:**

Signals must be ***Valid*** when the active clock edge occurs.

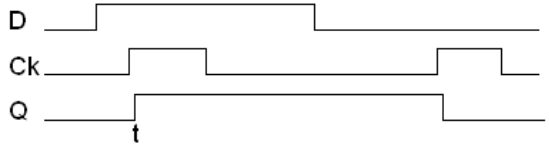
Valid = Stable (not changing) for a short period before and after the edge rise or fall.

Return to the *D-latch*

Clk=0 ; Nothing happens ("Hold" condition)
•Ck = 0 equivalent to (S=0 and R=0)

Clk=1 ; read D into Latch. Update value until C changes to 0.
•Ck = 1 equivalent to S=D and R=D'

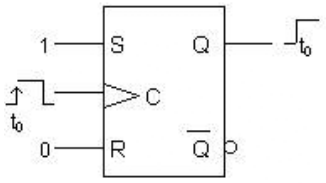
Operation (assuming output is initially deasserted, ie Low)



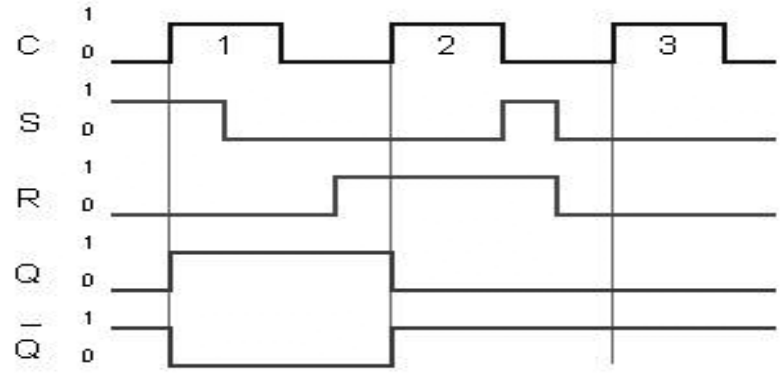
Note : NOT an Edge-Triggered D-Flip-Flop

Edge Triggered S-R Latch

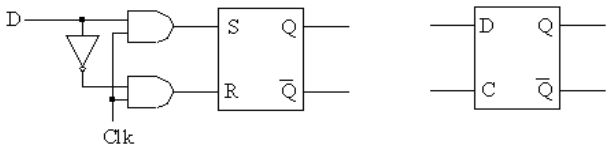
Different Approach

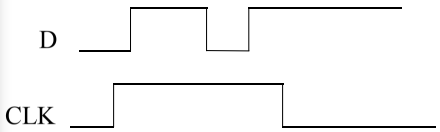


Inputs			Outputs		
S	R	C	Q	Q'	Comments
0	0	↑	Q	Q'	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	Invalid



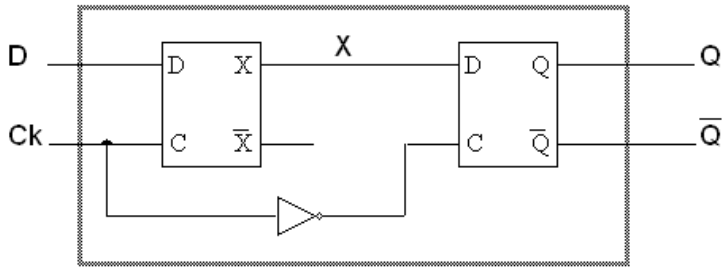
Race Condition : For a D-Latch, if the input changes state, while the clock is still high, then there will be an additional change to the output, although the data to be written was the former value.





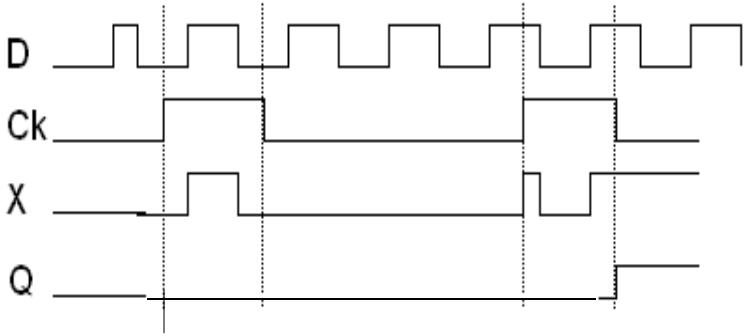
Solution to race condition: ***Master/Slave Flip-Flop***

Also : *Remember the previous reference to the edge triggered S-R*

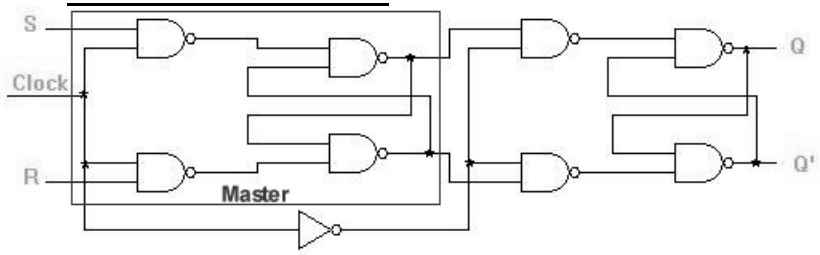


This is a ***Falling-Edge Triggered Master/Slave D Flip-Flop.***

The first Latch, called the Master, follows the input D when the Clock input is asserted. When Ck falls , X is closed and the second latch, called the slave, is open and gets the input from the output of the master latch.



S-R Latch Master/Slave

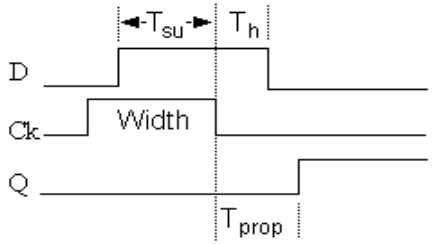


Edge Triggered devices = Flip Flops

•The Actual circuit is more complex than shown above!

•Characteristics:

- •Setup time (Tsu)
- •Hold time (Th)
- •Propagation Delay (Tprop)
- •Minimum clock width
- •... all are relative to clock edge

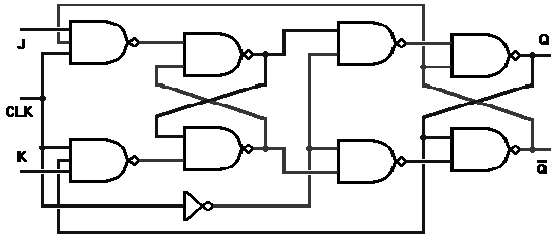


For a 74LS74:
T_{su} = 20 ns
T_h = 5ns
Min Clock Width = 25 ns
T_{prop} (max/typical)
low to high: 25/13
high to low: 40/25

Applications :

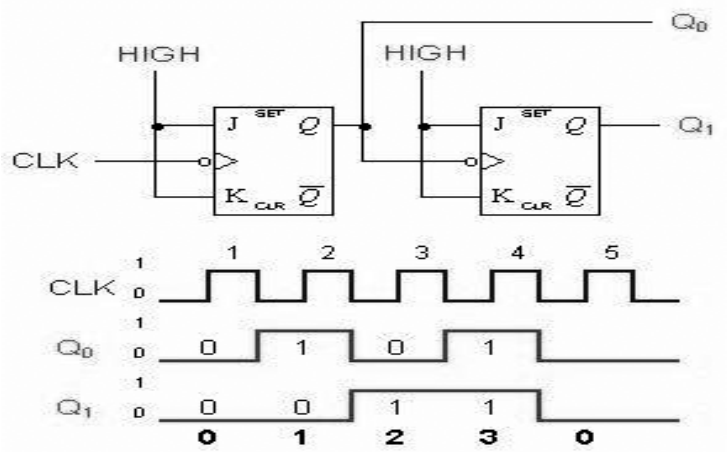
First, introduce another Flip-Flop Type :

J-K Flip-Flop

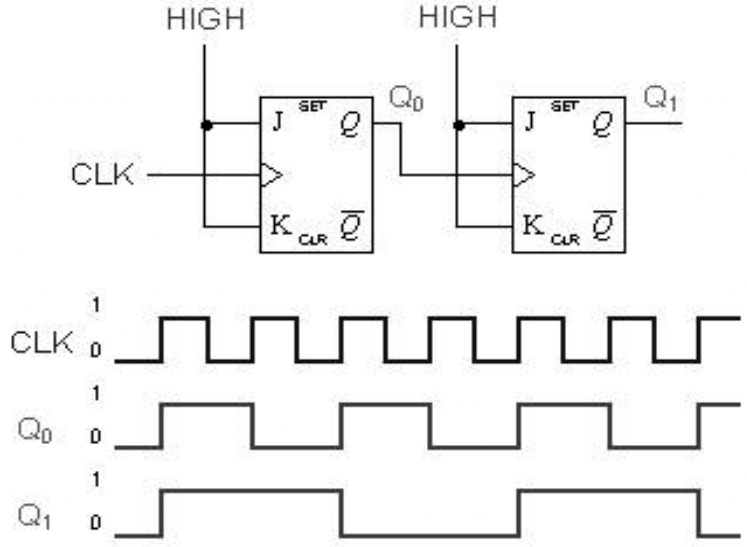


Inputs			Outputs		
J	K	C	Q	Q'	Comments
0	0	↑	Q	Q'	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	Q'	Q	Toggle

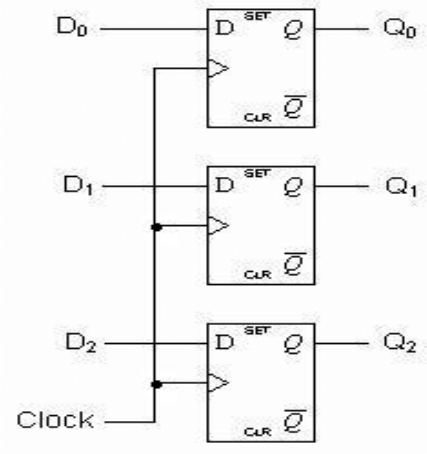
•*Counter*



•*Frequency Divider*



•*Parallel Data Storage*



The Basic Idea Behind Registers

Summary

Flip flops v. latches
- •Flip flops are edge triggered
- •Latches are level sensitive

Many variants
- •S-R, J-K, D, T latch/flip-flop
- •D-type most useful in processor design

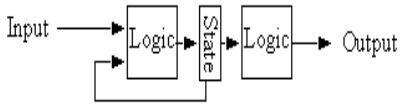
Beware of race conditions
- •Need edge triggered device in any feed-back path

State machines design

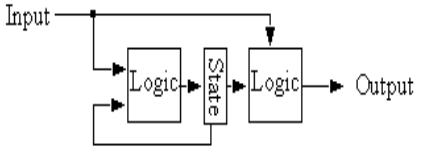
Interesting state machines have inputs and outputs.

State will change depending on input.

Two classes are Mealy and Moore:
- •Moore machine: output = f(state)



Mealy machine: output = f(state,input)



Generally: Moore machines have more states (mnemonic), but are easier to understand.

Design Process

General Procedure
- •Step 1: Create a State Diagram
- •Step 2: Write down a State Transition Table
- •Step 3: Figure out the inputs to the flip flops using the excitation table.
- •Step 4: Figure out functions for input to flip flops
- •Step 5: Implement the machine

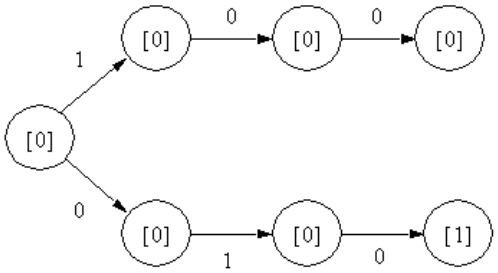
Example
- •"Write a string recognizer that recognizes the pattern 010 in its input.
- •When an input bit pattern is recognized, output a 1.
- •When the sequence 100 is seen, output zeros thereafter until a reset is asserted"
- •Examples:

X: 001010100010
Z: 000101010000

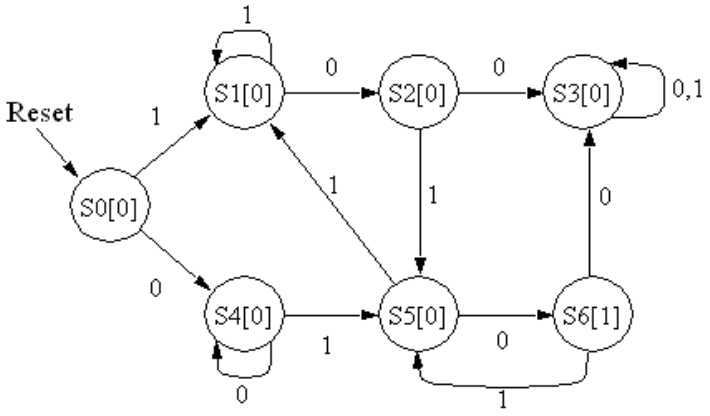
X: 0110110100100
Z: 0000000010000

1. State diagram creation

Start with easy paths, fill in the rest, reuse as much as possible!
This is a ***Moore*** machine:



2. Final state diagram



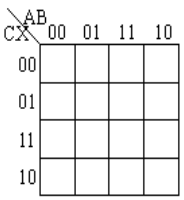
3. Symbolic State Transition Table:

State Number	Current State	Next (X=0)	Next (X=1)	Output
	S0			
	S1			
	S2			
	S3			
	S4			
	S5			
	S6			

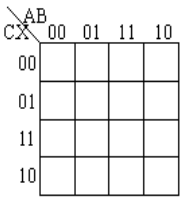
4. Figure out flip flop inputs

A	B	C	X	A'	B'	C'
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

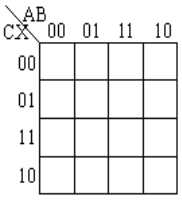
5. Reduce using K-maps



A' =



B' =

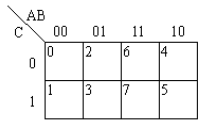


C' =

6. Output Logic

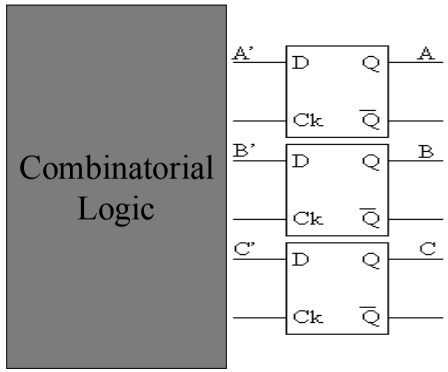
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	X

Karnaugh map



O =

6. Final Circuit



State minimization

Goal: to minimize the number of states in a state diagram.

- Basic idea: Identify and combine states with equivalent behavior.
- 2 states are equivalent if the output is the same and, for each input combinations, the next state is the same state or an equivalent state.

Approach:

- Start with state transition table.
- Identify states with the same behavior
- If such states go to the same next state, combine them and rename each occurrence of the old state in the state table.
- Repeat with new state table until no new combinations are possible.

Next Time : Minimization

State Transition Table

Prefix	Name	X=0	X=1	Output (X=0)	Output (X=1)
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011	S10	S0	S0	1	0
100	S11	S0	S0	0	0
101	S12	S0	S0	1	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0

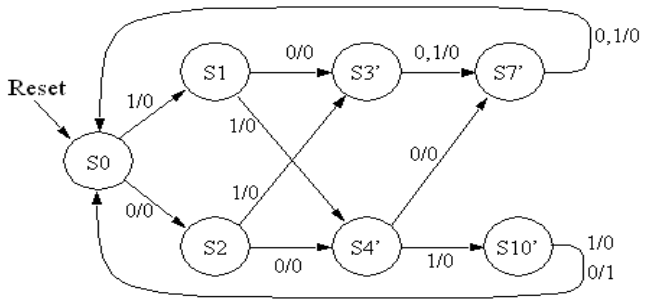
Modified State Transition Table

Prefix	**Name**	**X=0**	**X=1**	**Output (X=0)**	**Output (X=1)**
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S7’	S7’	0	0
01	S4	S7’	S10’	0	0
10	S5	S7’	S10’	0	0
11	S6	S7’	S7’	0	0
00x, 010 100 11x	S7’	S0	S0	0	0
011 101	S10’	S0	S0	1	0

Modified State Transition Table (2nd iteration)

Prefix	Name	X=0	X-1	Output (X=0)	Output (X=1)
Reset	S0	S1	S2	0	0
0	S1	S3’	S4’	0	0
1	S2	S4’	S3’	0	0
00, 11	S3’	S7’	S7’	0	0
01, 10	S4’	S7’	S10’	0	0
00x, 010 100 11x	S7’	S0	S0	0	0
011 101	S10’	S0	S0	1	0

State Diagram:



Summary

Components

- •Flip flops (edge triggered)
- •Latches (usually level triggered).

Designing Sequential Circuits

- •Step 1: Create a State Diagram
- •Step 2: Write down a State Transition Table
- •Step 3: Do state minimization
- •Step 4: Do state assignment
- •Step 5: Figure out the inputs to the flip flops using the excitation table.
- •Step 6: Figure out functions for input to flip flops