EE/CE 6301: Advanced Digital Logic

Bill Swartz

Dept. of EE
Univ. of Texas at Dallas

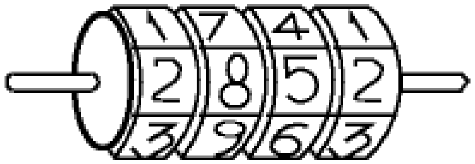
Session 09

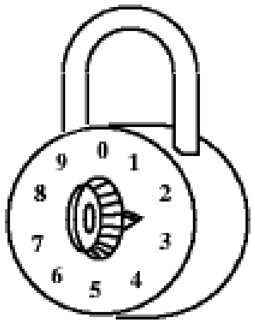
Synchronous / Asynchronous Design

Review of Sequential Components
(Dr. Balsara & Dr. Bhatia's lectures for EE3320)

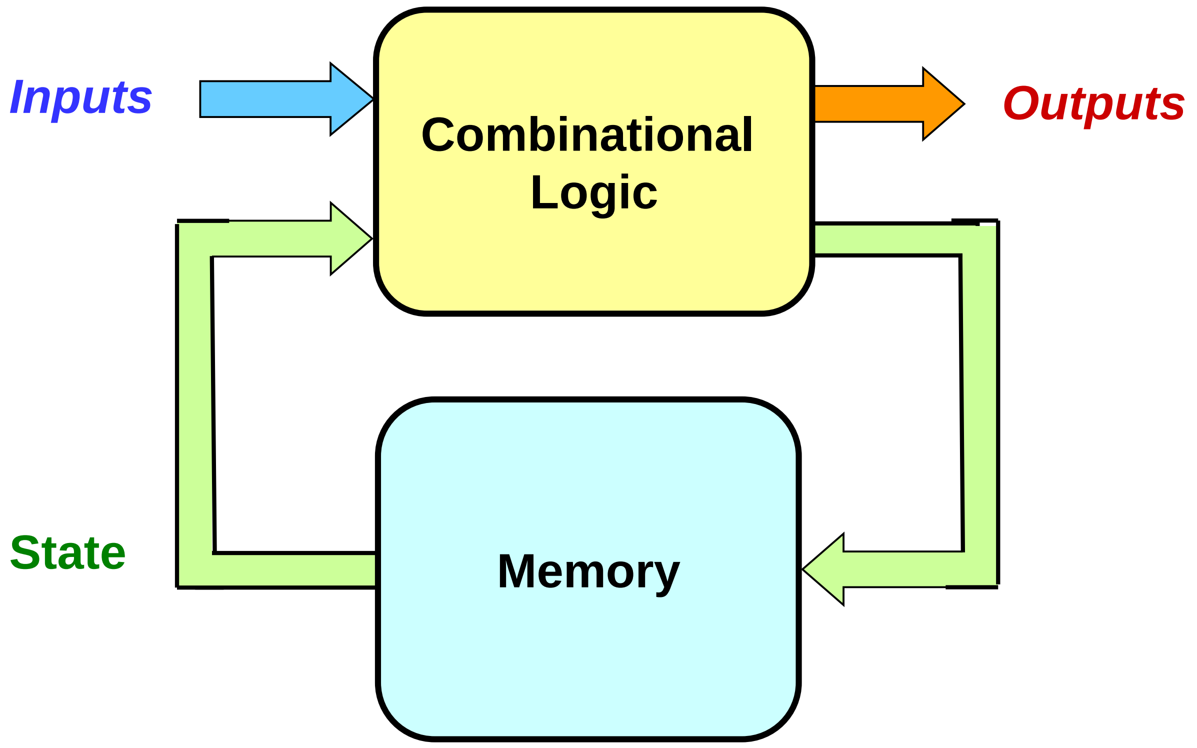
Sequential Logic Circuits

- In a *sequential circuit* steady state outputs are a function of the current inputs and past inputs; i.e., the circuit has memory
- Feedback in the logic paths.
- The state of the circuit is the state of the memory; the state is vector valued, each element of the state vector corresponds to one bit.
- Combinational vs. Sequential:





Sequential Network Model

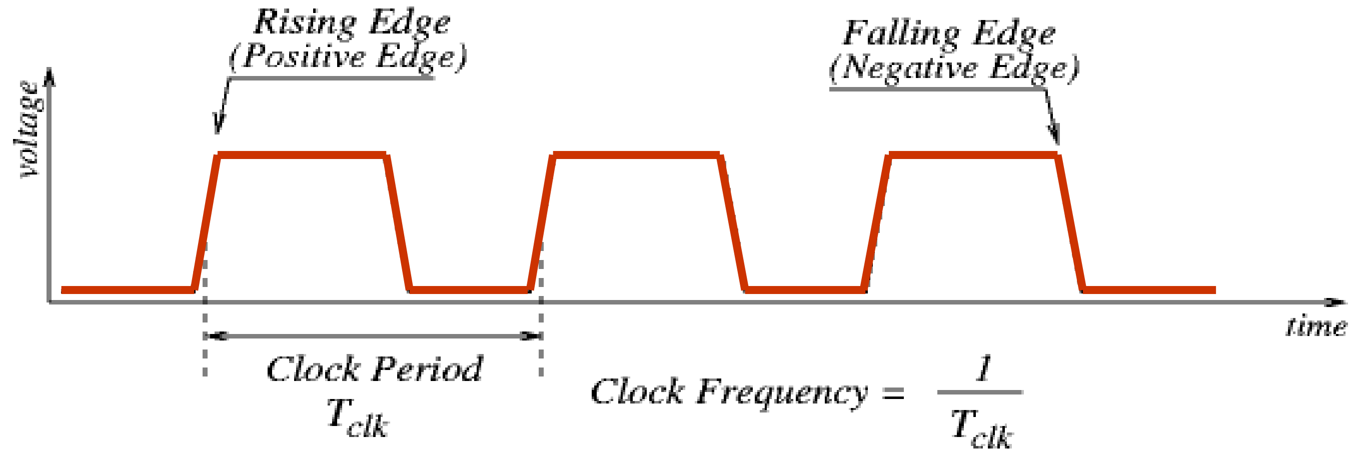


Synchronous vs. Asynchronous

- Sequential circuits can be
  - Asynchronous (feedback mode – level mode)
  - Synchronous (clocked mode)
- **Asynchronous Sequential Circuits**: State changes can occur at any time.
- **Synchronous Sequential Circuits**: State changes can occur only in conjunction with a reference timing signal. This signal is generally known as the *clock* signal of the system.

Clocking of Synchronous Circuits

- Clock signals are usually periodic although they often do not have a 50% duty cycle.



- Generally, state transitions occur at clock edges and usually at either rising edges or falling edges but not both.
- The edge that causes state transitions is called the *active* edge of the clock.

Latches and Flip-Flops

- Latches and flip-flops are the basic building blocks of sequential circuits

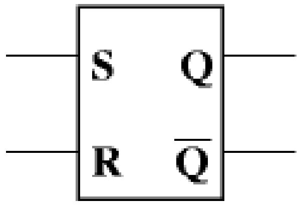
- Bistable devices – 0 state and 1 state.

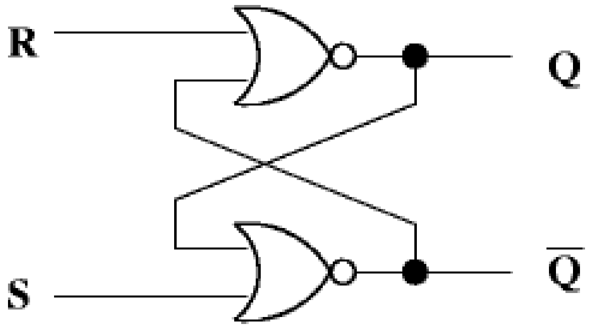
- Two modes of operation:
  - *Direct mode*: the state of the device responds directly to applied inputs (asynchronous circuit).
  - *Clocked mode*: the state of the inputs only matters when the clock signal is active.

SR Latch

- The most basic sequential circuit is an SR latch (the term latch may be used to indicate the device operates in a direct mode – but the term is not always used this way)
- *S* = set (input); *R* = reset (input); *Q* = current state of the latch (output); $\overline{Q}$ = complement of current latch state (output);

  note – no clock signal ⇨ *SR latch is asynchronous*.
- ***NOR implementation of an SR latch:***





NOR SR Latch – *Characteristic Table*

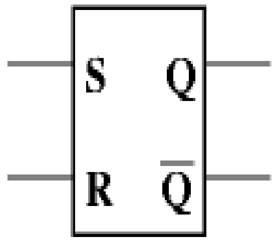
- Given the current state and inputs to a latch, what is the next state. Typically, symbols Q, Q_{n-1}, Q^t , etc. are used to denote the current state, and correspondingly , Q^*, Q_n, Q^{t+1}, etc. denote the next state.

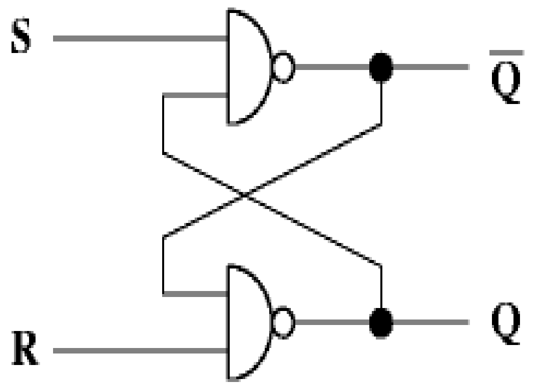
Q	S	R	Q*	$\overline{Q^*}$
0	0	0	0	1
0	0	1	0	1
0	1	0	1	0
0	1	1	*Undefined* (0 0)	
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	*Undefined* (0 0)	

S	R	Q*
0	0	Q
0	1	0
1	0	1
1	1	*Undefined* (0 0)

Condensed form

NAND SR Latch



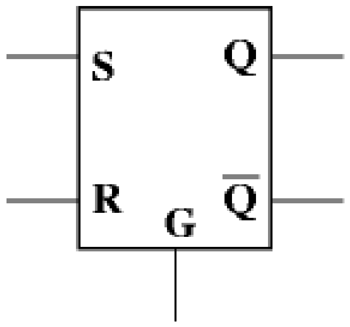


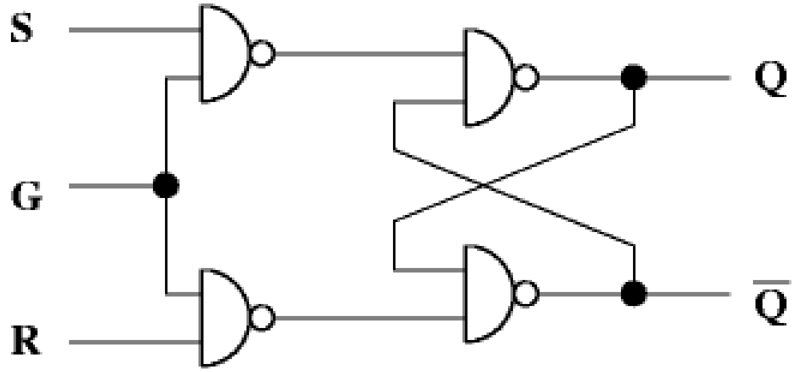
S	*R*	*Q**
0	0	*Undefined* (1 1)
0	1	0
1	0	1
1	1	*Q*

Characteristic Table

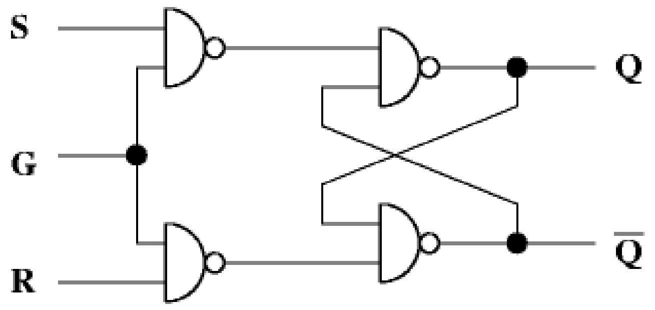
Gated SR Latch

- Motivated by the need to control when state transitions occur – don't want latch to respond to spurious glitches on input lines
- Add a gate input, G; when G is inactive, the latch does not change state regardless of the other inputs – synchronous circuit.





Gated SR Latch – *Characteristic Table*

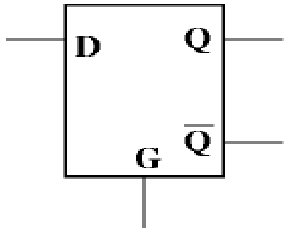


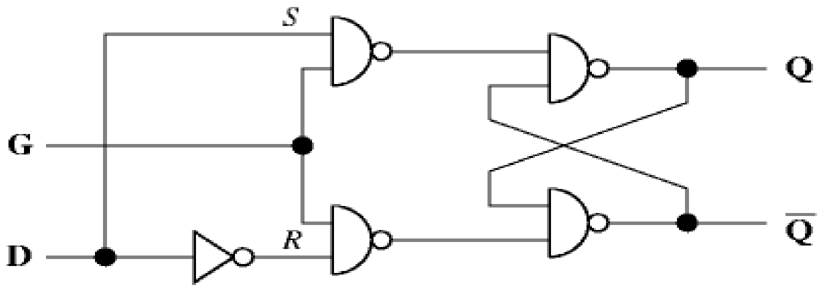
G	*S*	*R*	*Q**
1	0	0	*Q*
1	0	1	0
1	1	0	1
1	1	1	*Undefined* (0 0)
0	X	X	*Q*

- Still have problems with 11 inputs to SR latch – one solution is to define the condition away: never allow both inputs to the SR to have the same value.

Gated D latch

- D (data) latch or D flip-flop does not allow both inputs to an SR latch to have the same value.





G	*D*	*Q**	*$\overline{Q}$**
1	0	0	1
1	1	1	0
0	X	Q	$\overline{Q}$

Characteristic Table

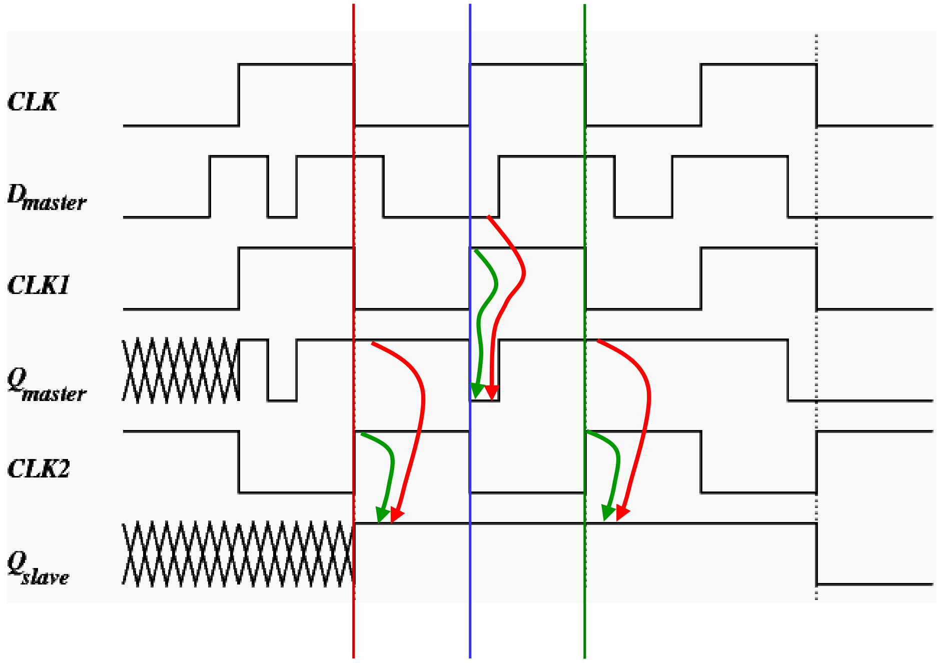
D Flip-flop

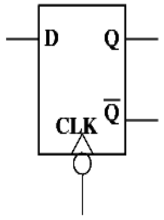
- The D latch has a shortcoming – the inputs should not change while the gate signal is asserted (otherwise there are multiple asynchronous state changes which can lead to problems in a circuit).

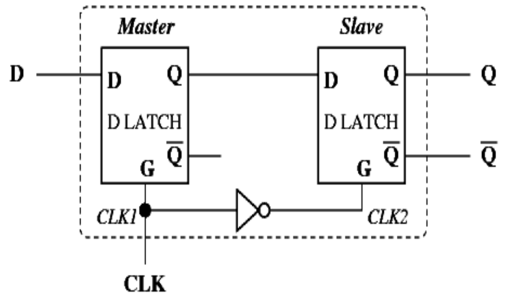
- One solution is to design the circuit so that state changes occur during clock edges rather than during clock levels –this type of device is called edge-triggered (i.e., flip-flop).

Master-Slave D FF

- When CLK is high, output of master is allowed to change with D; when CLK is low (falling edge), the output of the master is fixed and propagated through to the output of the slave ⇨ this flip-flop triggers on *falling* or *negative edge*.



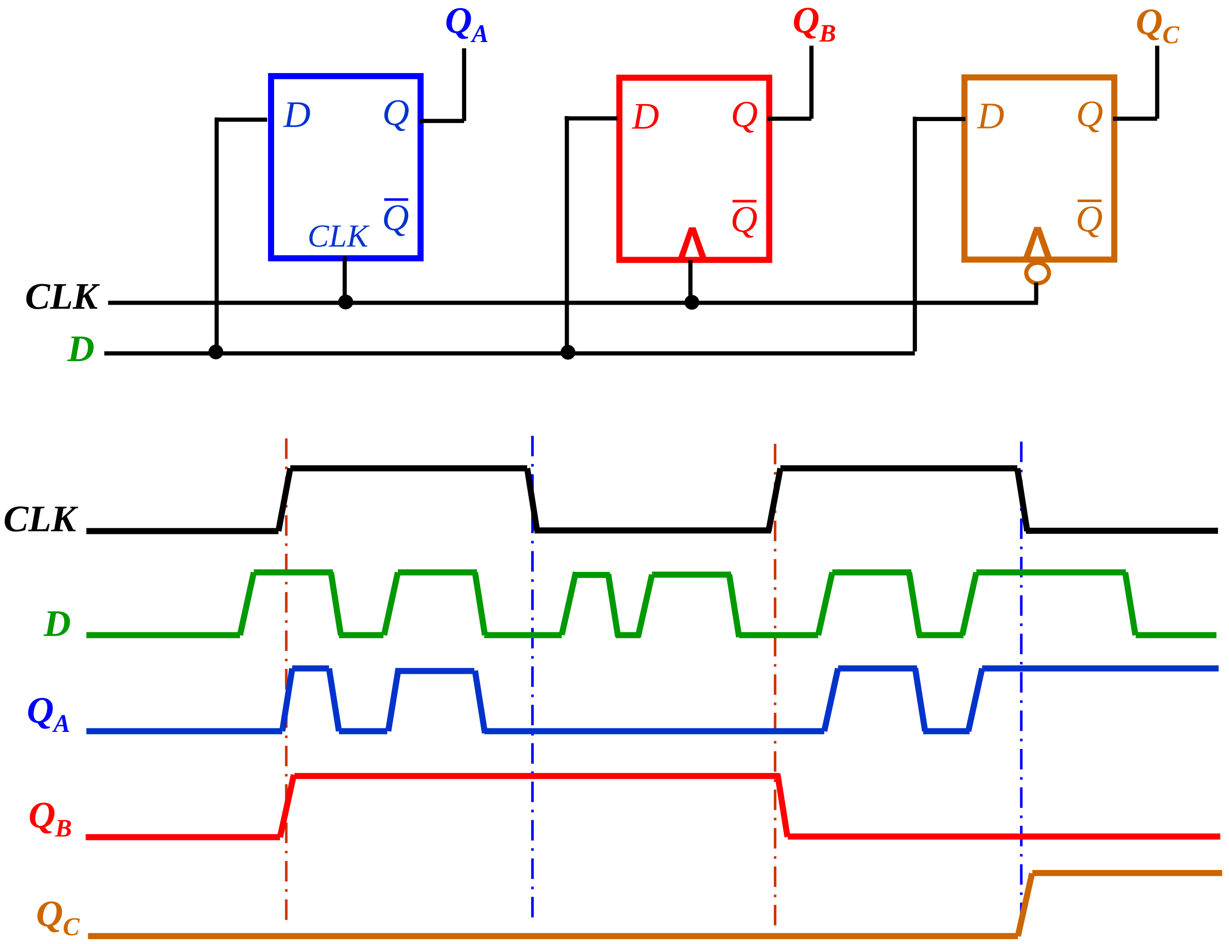




CLK	D	Q*	$\overline{Q}$*
↴	0	0	1
↴	1	1	0

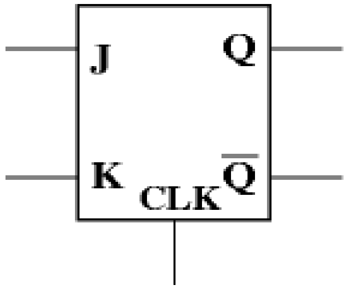
Characteristic Table

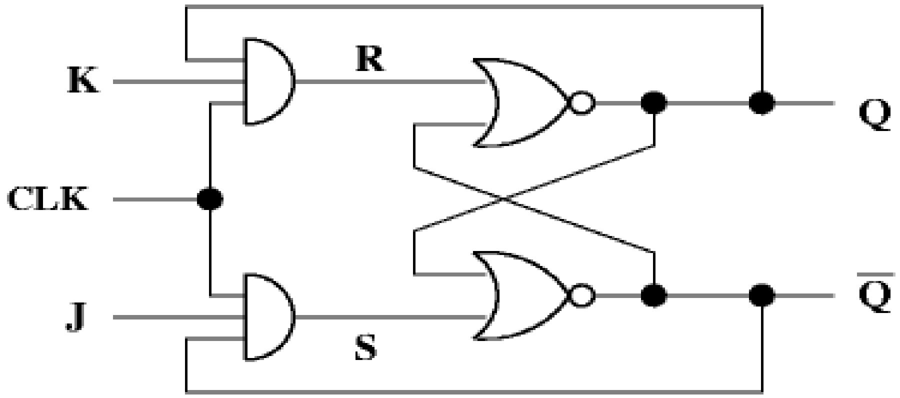
Level vs. Edge Triggering



JK Flip-flop

- A JK flip-flop can be considered an extension of SR latch that does not allow both inputs to an SR latch to have the same value.
- Letters J and K have no particular meaning, but were selected to avoid conflict with other commonly used symbols*.

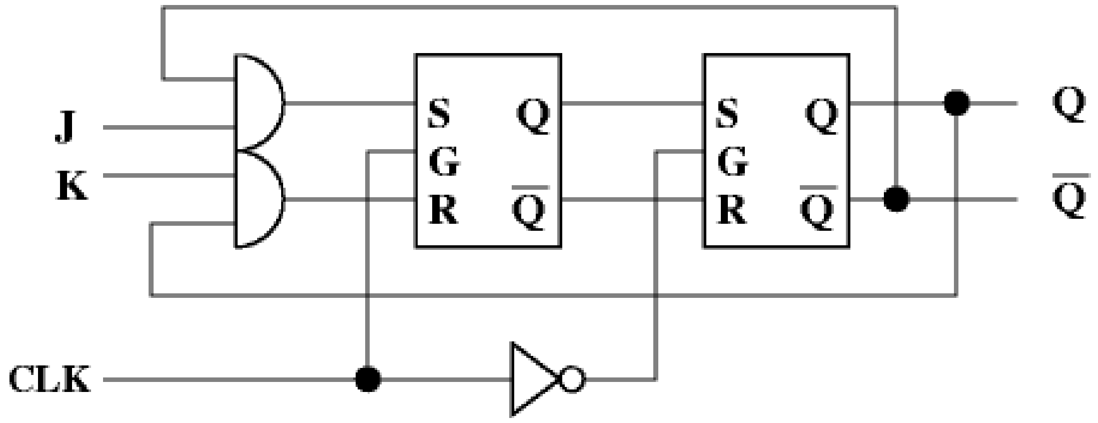




*Sam's Modern Dictionary of Electronics, 1984

JK Flip-flop (cont.)

- The correct design requires the use of a master/slave pair.

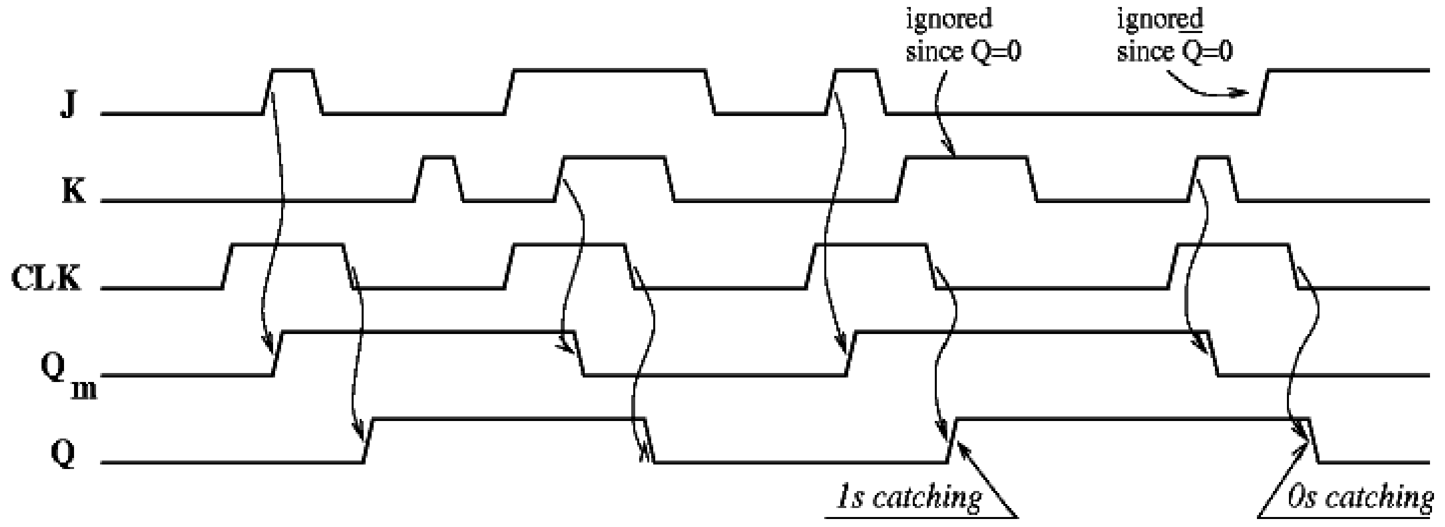


- Characteristic table for a negative edge-triggered JK flip-flop.

CLK	*J*	*K*	*Q**
↓	0	0	Q
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q}$

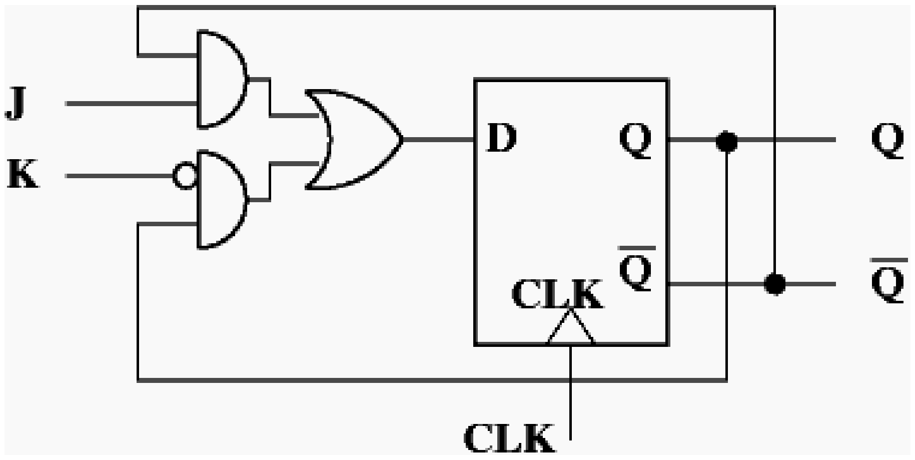
1/0 Catching in JK Flip-flops

- If inputs *J* and *K* are not held valid during the entire period when CLK is active for the master, the above flip-flop exhibits *1s and 0s catching* behavior.
- *1s Catching*: Output changes to 1 even though *K* and not *J* is asserted at the end of the triggering pulse.
- *0s Catching*: Output changes to 0 even though *J* and not *K* is asserted at the end of the triggering pulse.



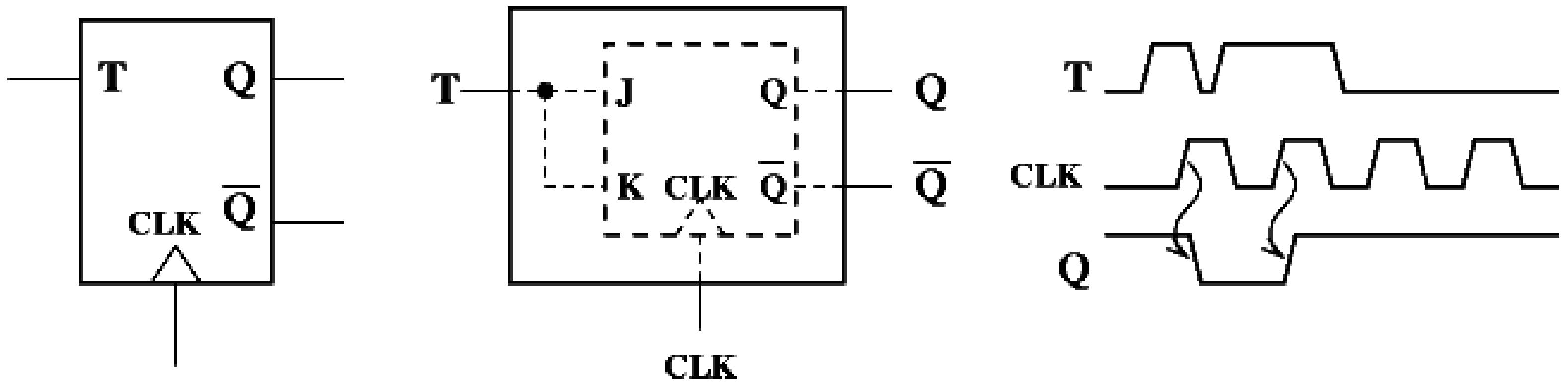
Better JK Flip-flop

- The problem of 1s and 0s catching can be solved by using the edge-triggered JK flip-flop which uses an edge-triggered D flip-flop internally:



T Flip-flop

- A T flip-flop changes state on every clock if it is enabled (T="1"). It can be implemented by connecting together the J and K inputs of a JK flip-flop.

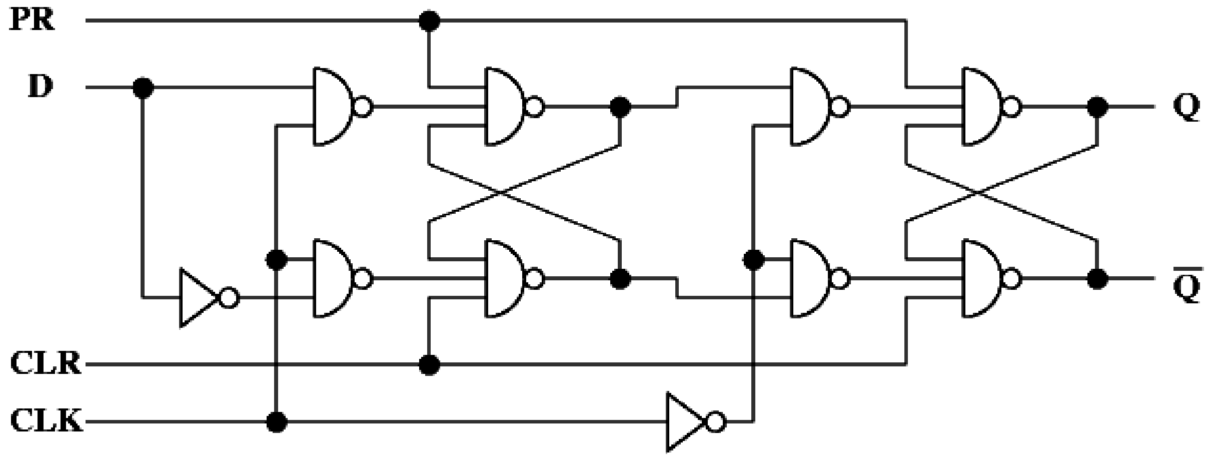


- Characteristic table for a positive edge-triggered T flip-flop with enable.

CLK	*T*	*Q**
↑	0	Q
↑	1	$\overline{Q}$

Flip-flop with Asynchronous Inputs

- Some flip-flops have *asynchronous inputs* that may be used to force the flip-flop to a particular state independent of CLK and other inputs.
- *Example*: A negative edge-triggered D flip-flop with active low asynchronous preset and clear inputs.



Setup and Hold Times

- **Setup time,** *t_{su}*, is the time period prior to the clock becoming active (edge or level) during which the flip-flop inputs must remain stable.
- **Hold time,** *t_h*, is the time after the clock becomes inactive during which the flip-flop inputs must remain stable.
- Setup time and hold time define a *window of time during which the flip-flop inputs cannot change* – quiescent interval.

Propagation Delay

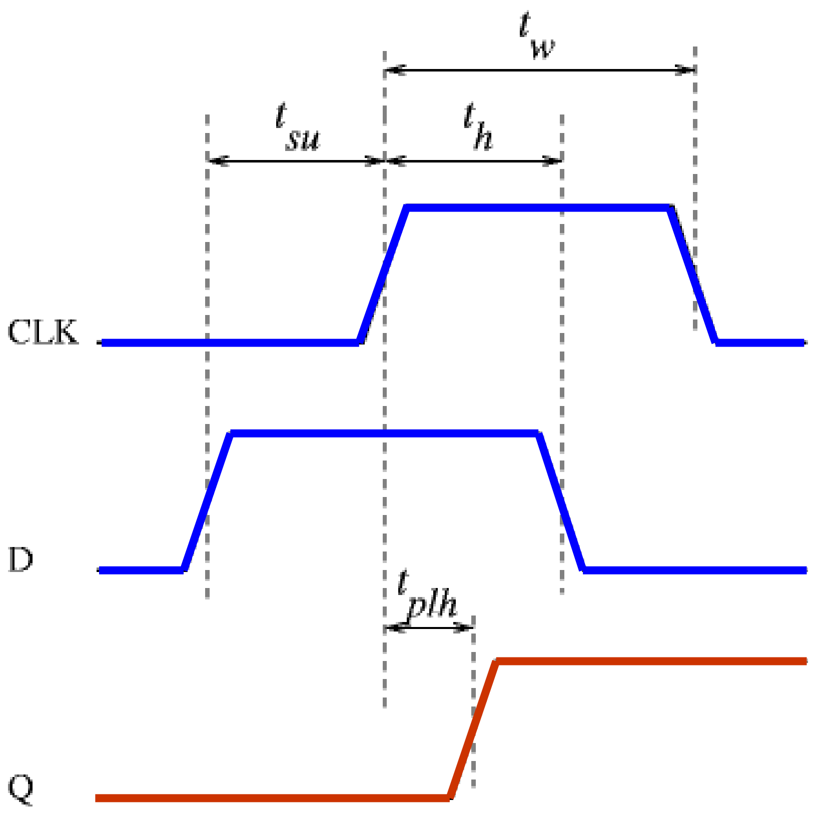
- **Propagation delay,** t_{pHL} and t_{pLH} , has the same meaning as in combinational circuit – beware propagation delays usually will not be equal for all input to output pairs. There can be two propagation delays:

  $t_{C\text{-}Q}$ (*clock→Q* delay) and $t_{D\text{-}Q}$ (*data→Q* delay).

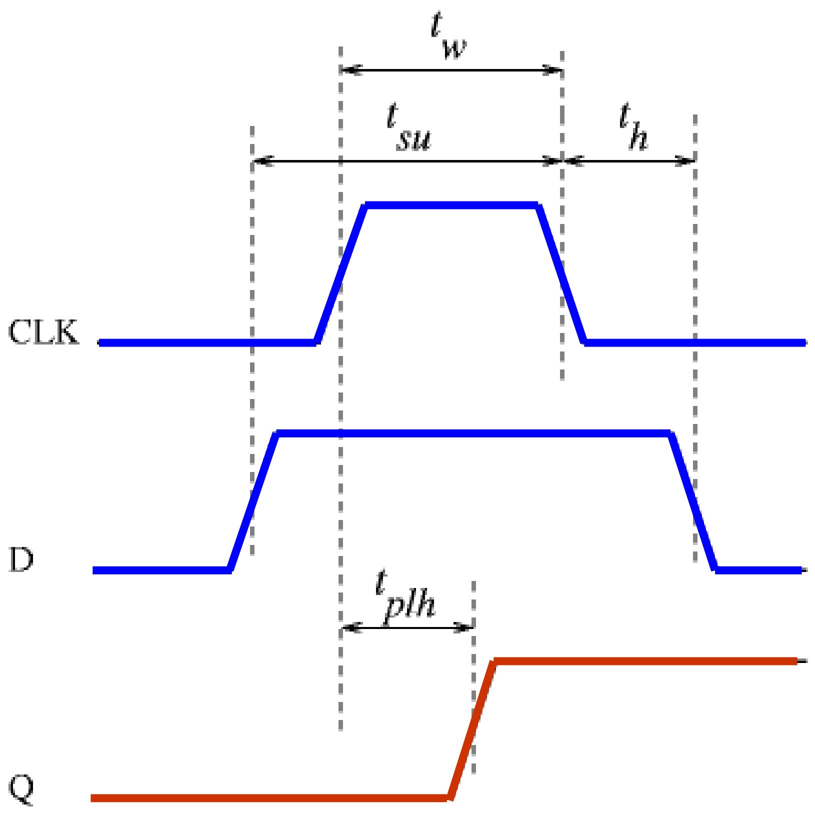
- For a level or pulse triggered latch:
  - Data input should remain stable till the clock becomes inactive.
  - Clock should remain active till the input change is propagated to Q output. That is, active period of the clock,

$$t_w > \max\{t_{pLH}, t_{pHL}\}$$

Latch & Flip-flop Timing Parameters

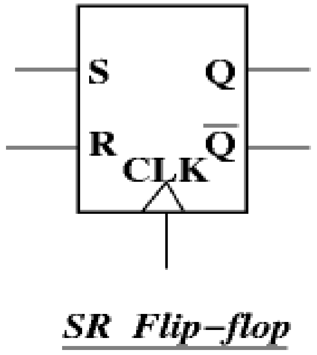
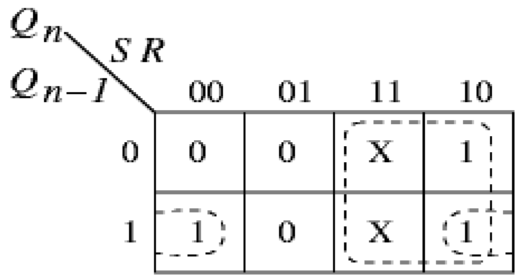
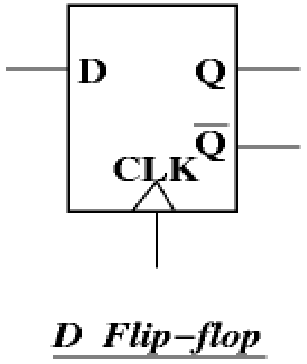
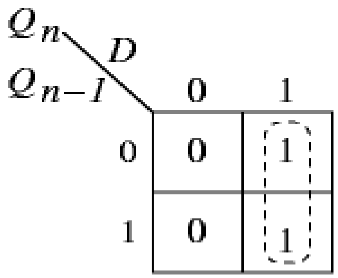


D Flip–flop (edge–triggered)
(positive edge triggering)



D Latch (level or pulse triggered)
(positive level triggering)

Summary of Flip-flops

Symbol	Characteristic Table	Characteristic Equation
		
		$Q_n = S + \overline{R}\,Q_{n-1}$
		
		$Q_n = D$

SR characteristic table:

S	R	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	?

D characteristic table:

D	Q_n
0	0
1	1

SR K-map:

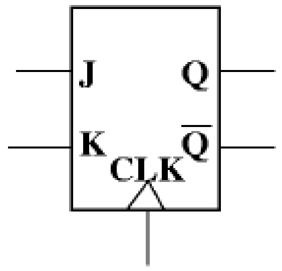
$$\begin{array}{c|cccc} Q_{n-1} \backslash SR & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 0 & X & 1 \\ 1 & 1 & 0 & X & 1 \end{array}$$

D K-map:

$$\begin{array}{c|cc} Q_{n-1} \backslash D & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 0 & 1 \end{array}$$

Summary of Flip-flops (cont.)

Symbol	Characteristic Table	Characteristic Equation

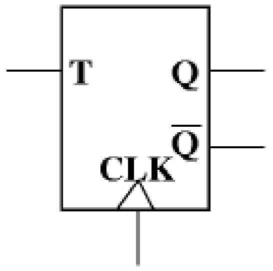


JK Flip-flop

J	K	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	$\overline{Q_{n-1}}$

$$\begin{array}{c|cccc} Q_{n-1} \backslash JK & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{array}$$

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



T Flip-flop

T	Q_n
0	Q_{n-1}
1	$\overline{Q_{n-1}}$

$$\begin{array}{c|cc} Q_{n-1} \backslash T & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

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

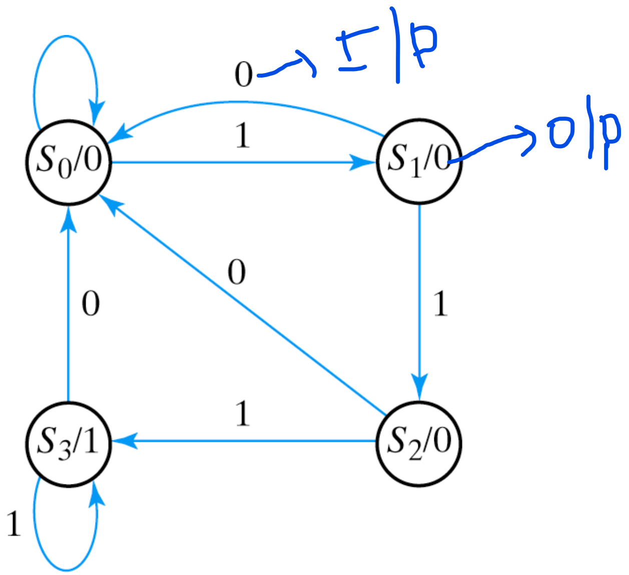
Review of Sequential Circuit Design

Basic FSM Design Procedure

1. Understand the problem and the different information classes (minimal number) required to solve it.
2. Convert these information classes into distinct states, and determine the state transition diagram of the FSM.
3. Encode states in binary, and obtain state transition table and FF excitation for desired FF type.
4. Minimize the FF input functions (using K-Maps, for example) and implement the FSM using these FFs and logic gates (or MUXes) that implement the FF's input functions.

Example #1

- Detect 3 consecutive 1's in a bit stream.
- Definition of states:
  - **State S0 : zero 1s detected**
  - **State S1 : one 1 detected**
  - **State S2 : two 1s detected**
  - **State S3 : three 1s detected**



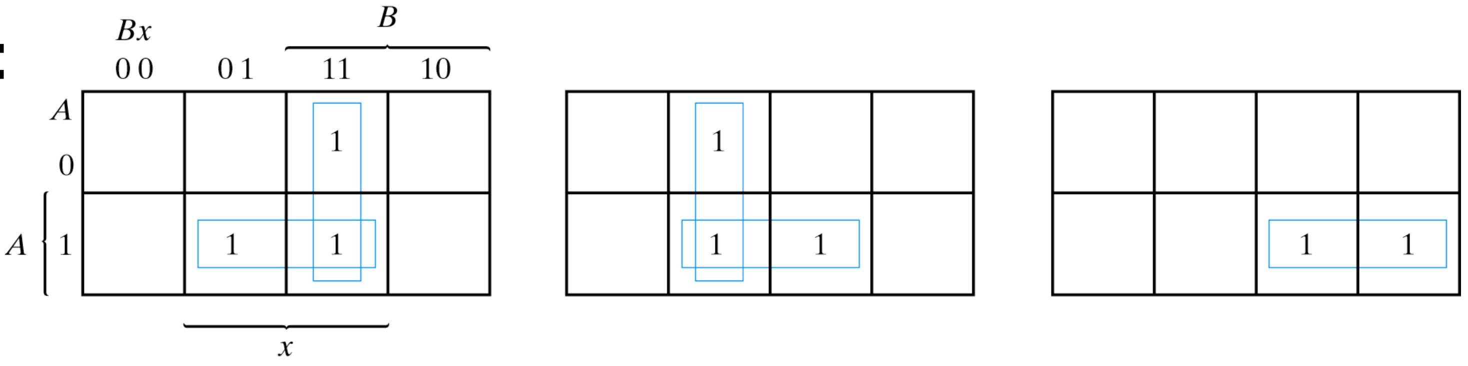
Example #1 (cont.)

- Sequence of outputs, inputs, and flip flop states enumerated in state table
- Present state indicates current value of flip flops
- Next state indicates state after next rising clock edge
- Output is output value on current clock edge
- State codes:
  - **State S0 : 00**
  - **State S1 : 01**
  - **State S2 : 10**
  - **State S3 : 11**

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

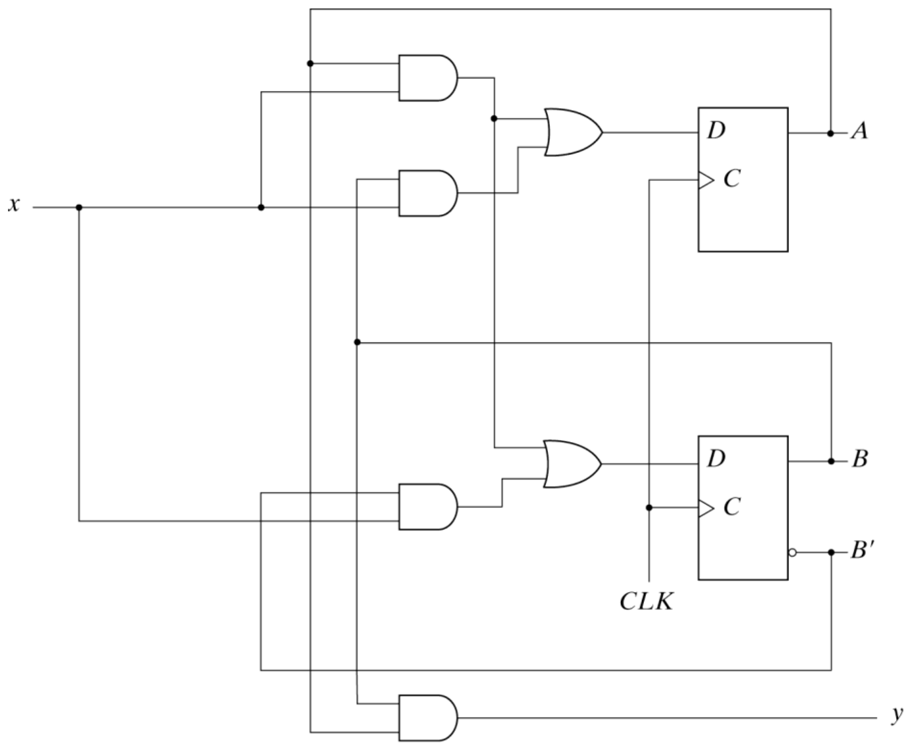
Example #1 (cont.)

- K-maps:



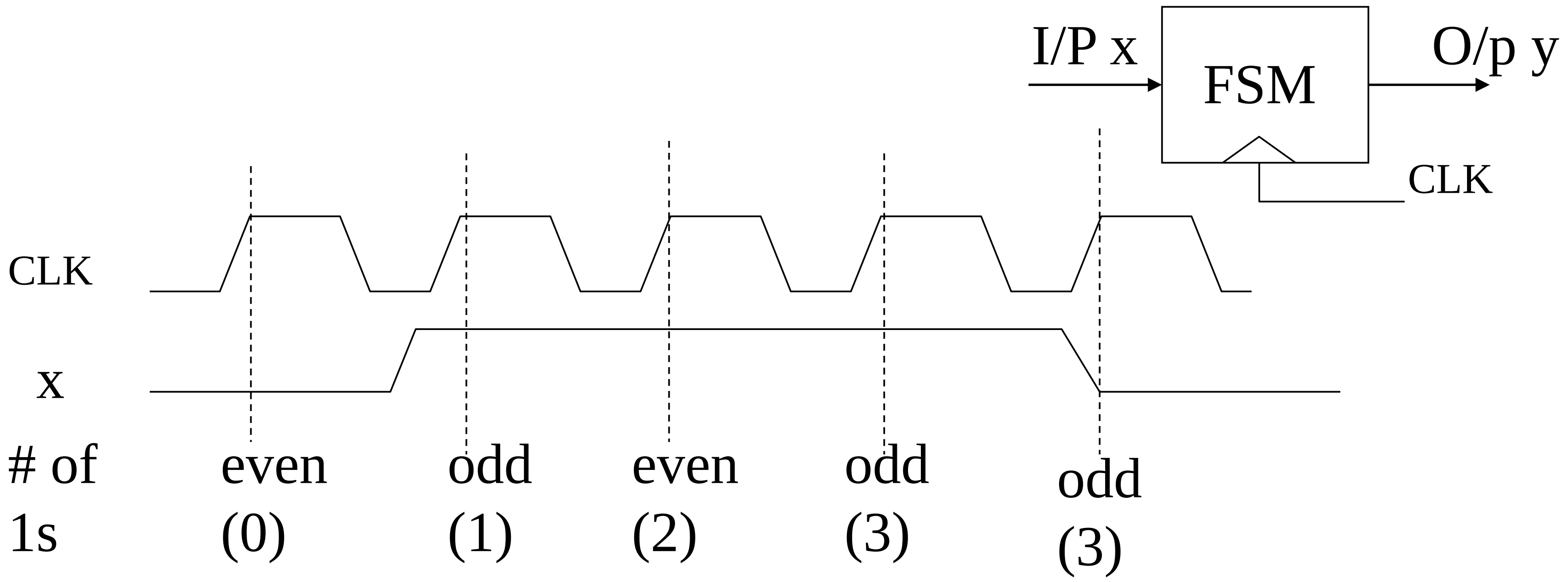
$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$

- Circuit:



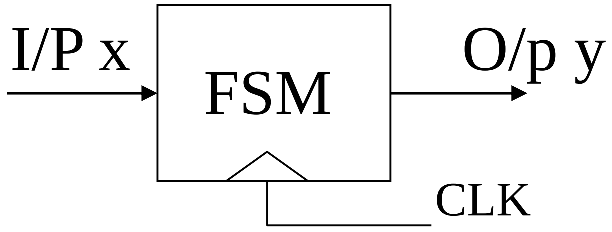
Example #2 (Even-Parity Checker)

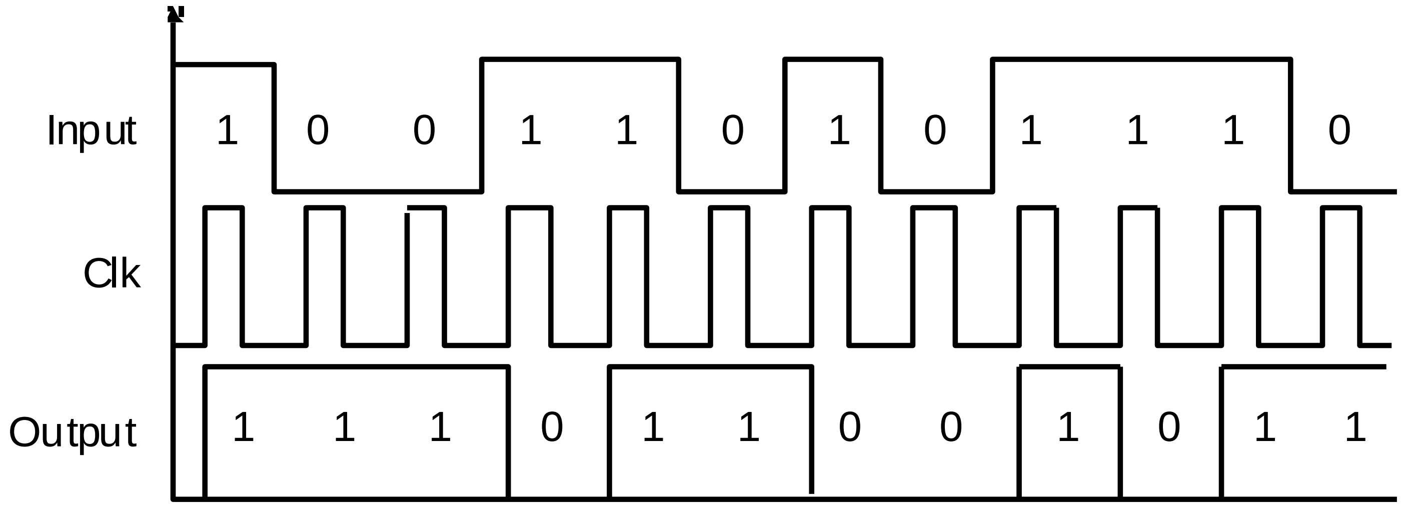
- There is a bit-serial I/P line. Design an FSM that outputs a '0' if an even # of 1's have been received on the I/P line and the outputs a '1' otherwise.
  - If a synchronous sequential circuit is being designed, the counting of the # of 1s occur every clock cycle.



Example #2 (Cont.)

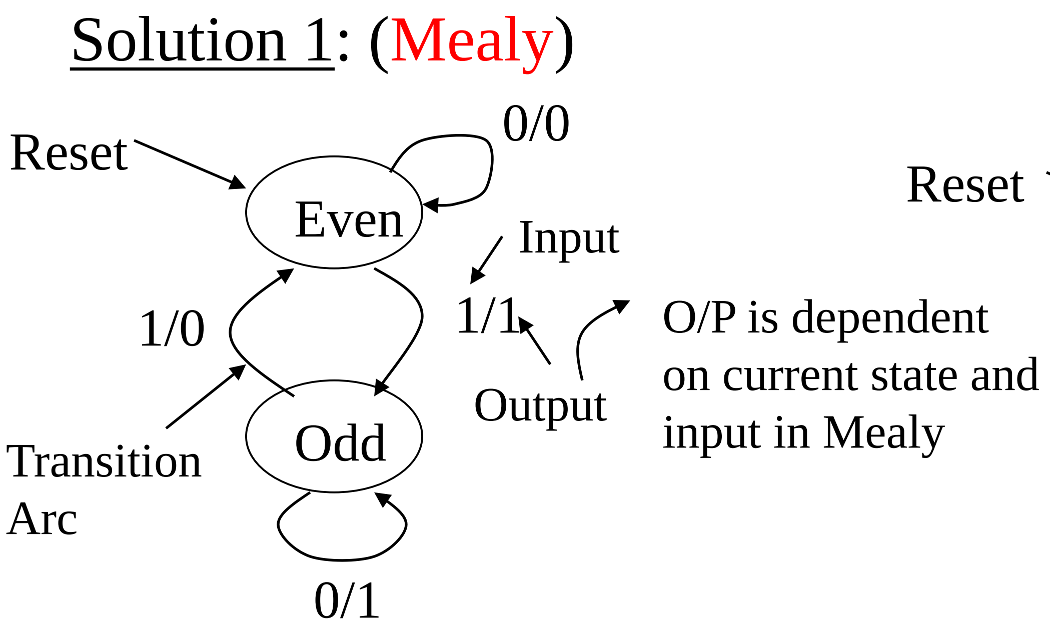
- Typical Behavior:

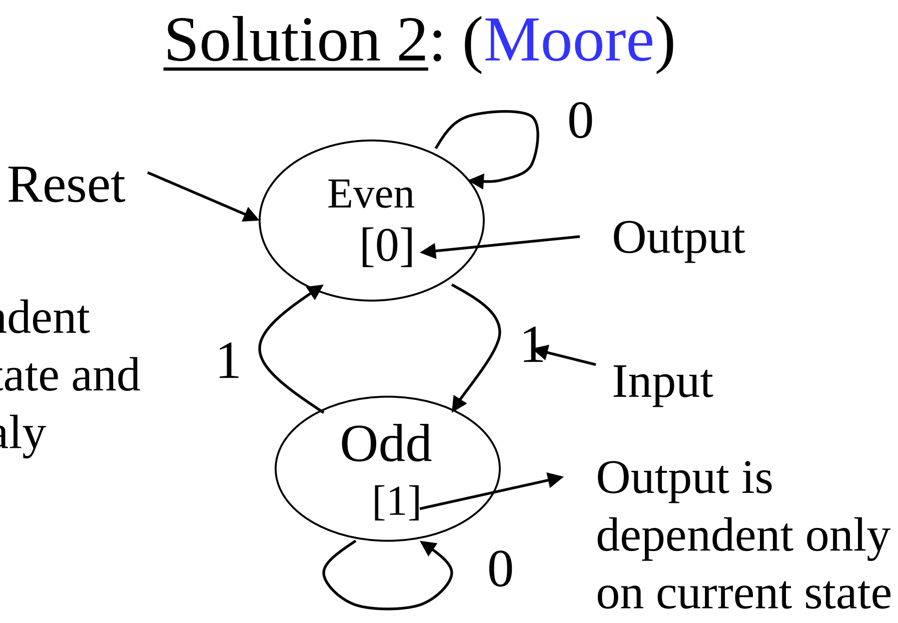




Timing Behavior: Input 1 0 0 1 1 0 1 0 1 1 1 0

Example #2 – Mealy vs. Moore Models

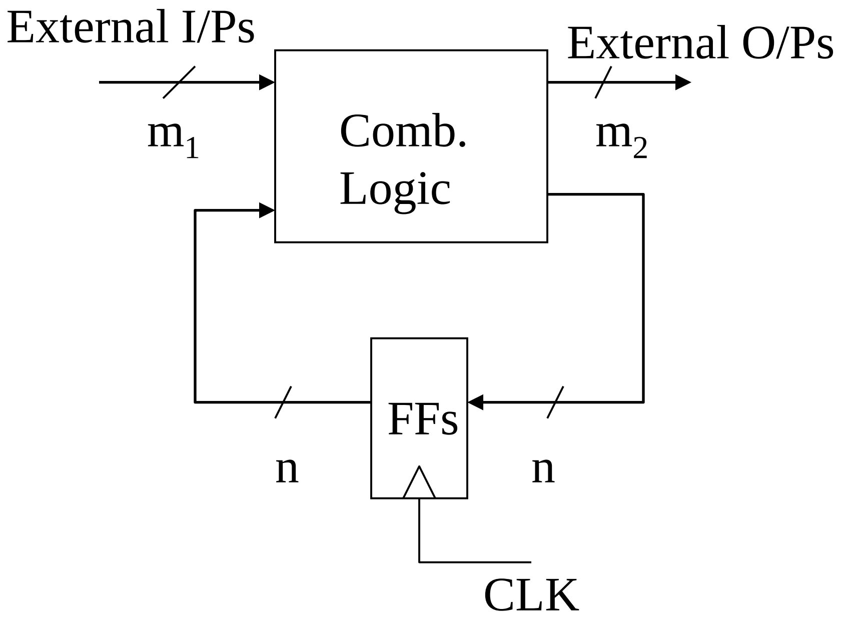




Mealy Machine: Output is associated with the state transition, and appears **before** the state transition is completed (by the next clock pulse).

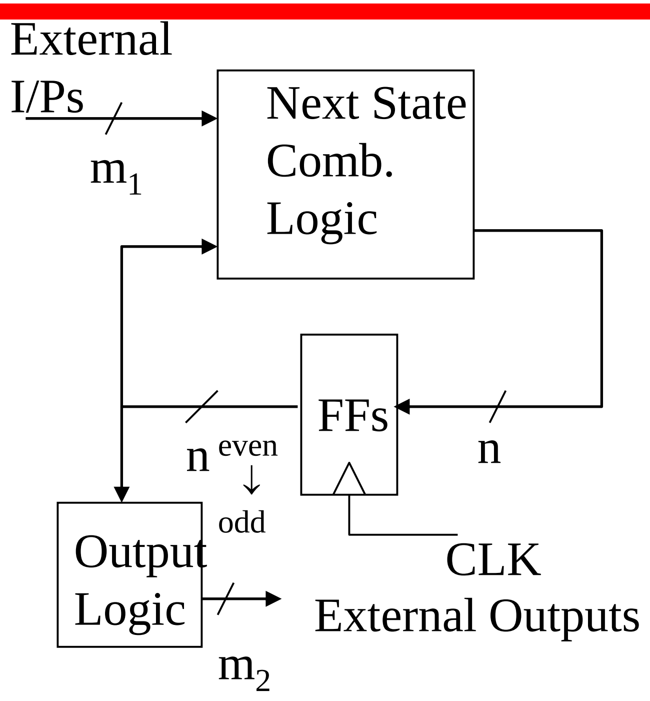
Moore Machine: Output is associated with the state and hence appears **after** the state transition takes place.

Example #2 – Mealy vs. Moore Models



Mealy Machine Model

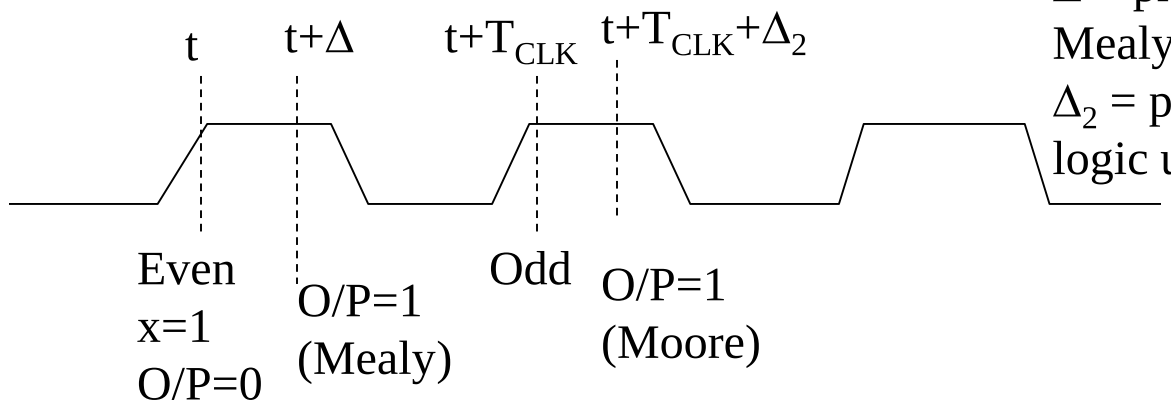
Time t : Even I/P



Moore Machine Model

Δ = propagation delay of logic of Mealy M/C

Δ_2 = propagation delay of O/P logic unit of Moore M/C



Example #2 – Transition Table

Even State: 0 ; Odd State: 1; State Variable A

Present State A	Input x	Next State A^+	Moore O/P y_1	Mealy O/P y_2	D-FF Excit. D_A	T-FF Excit. T_A
0	0	0	0	0	0	0
0	1	1	0	1	1	1
1	0	1	1	1	1	0
1	1	0	1	0	0	1

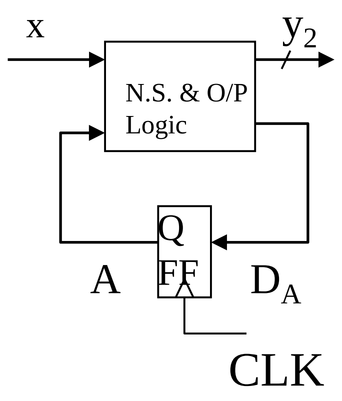
Input variables to comb. logic

Output functions

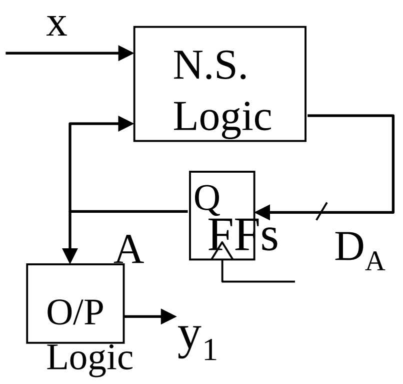
$D_A = A \oplus x$; $T_A = x$

$y_1 = A$ for Moore

$y_2 = A \oplus x$ for Mealy

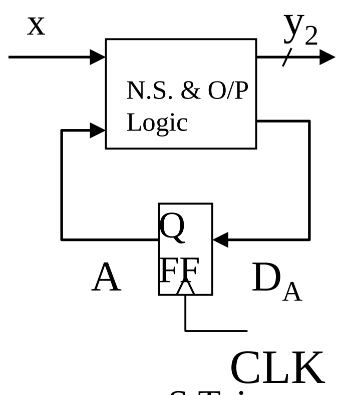


Or

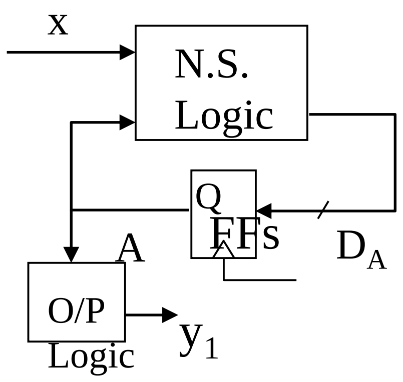


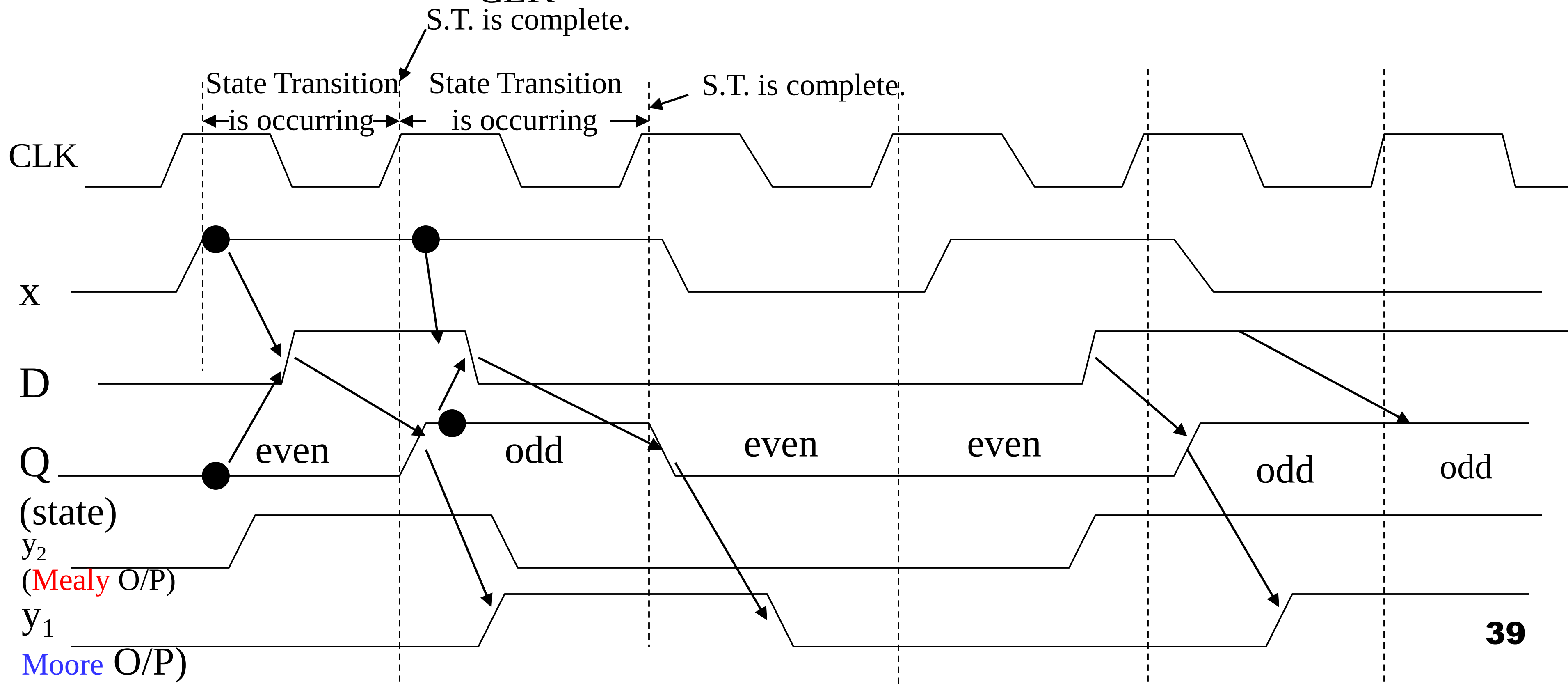
Example #2 – Behavior Using D FF

Mealy



Moore

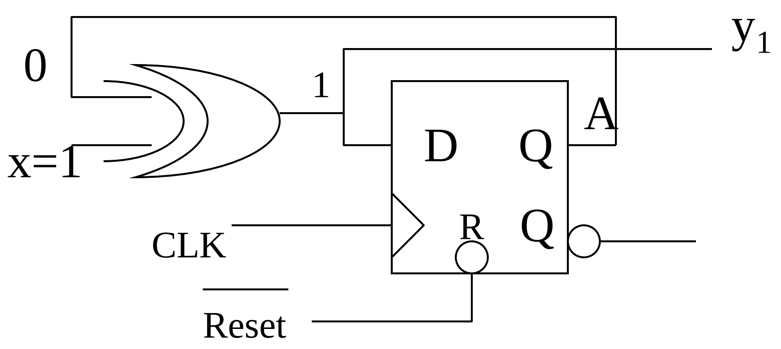




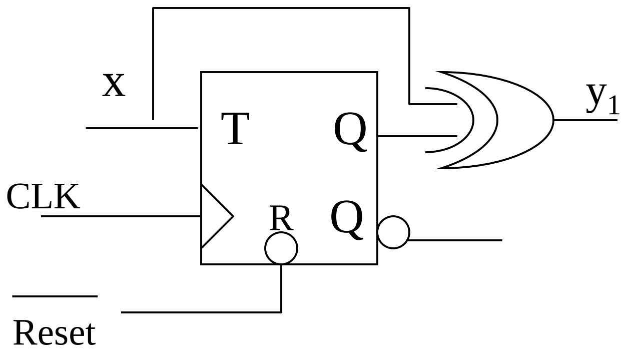
Example #2 – Implementations

Mealy M/C Implementation

a) D-FF



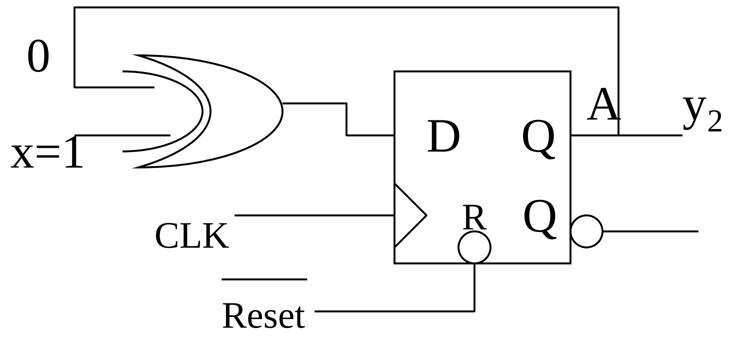
b) T-FF



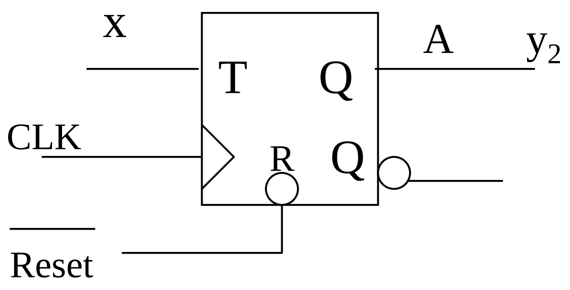
Mealy O/P is not synchronized with clock.

Moore M/C Implementation

a) D-FF



b) T-FF



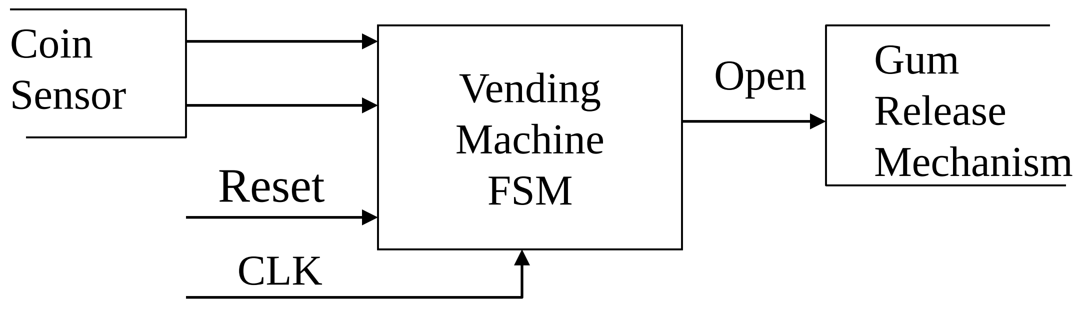
Moore O/P is synchronized with clock.

Mealy vs. Moore Machines

	Mealy	Moore
(1)	O/Ps depend on the present state and present I/Ps	O/Ps depend only on the present state
(2)	The O/P change asynchronously with the enabling clock edge	Since the O/Ps change when the state changes, and the state change is synchronous with the enabling clock edge, O/Ps change synchronously with this clock edge
(3)	A counter is not a Mealy machine	A counter is a Moore machine
(4)	A Mealy machine will have the same # or fewer states than a Moore machine	

Example #3 (from Katz/Borriello Book)

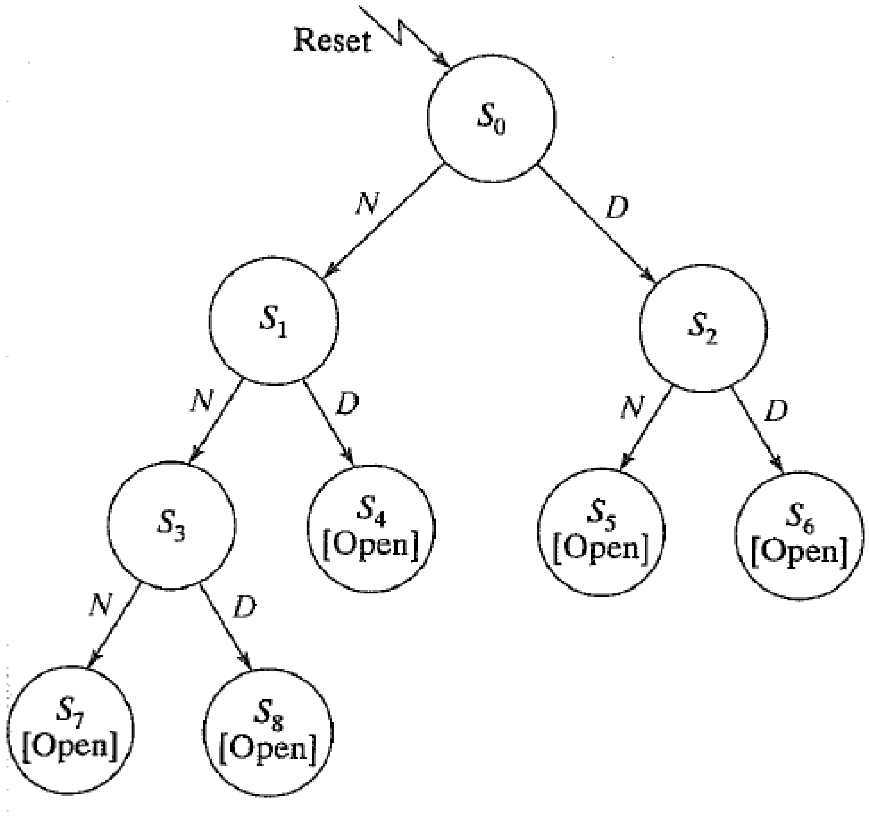
Design the controller for a simple vending machine. This machine delivers an item after it receives 15 cents in coins. The machine has a single coin slot that receives only nickels and dimes, one at a time. The machine does not provide a change.



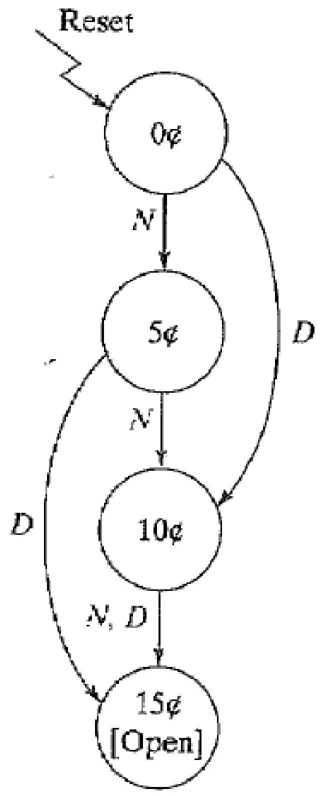
Vending Machine block diagram

Example #3 – Definition of States

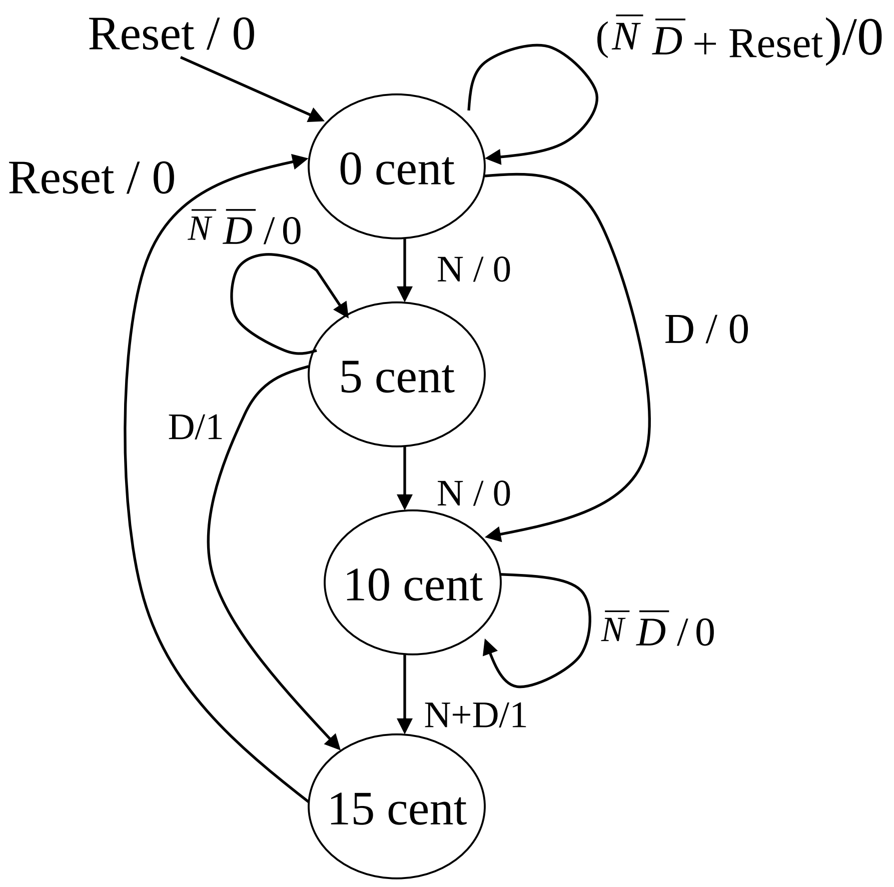
Initial State Diagram



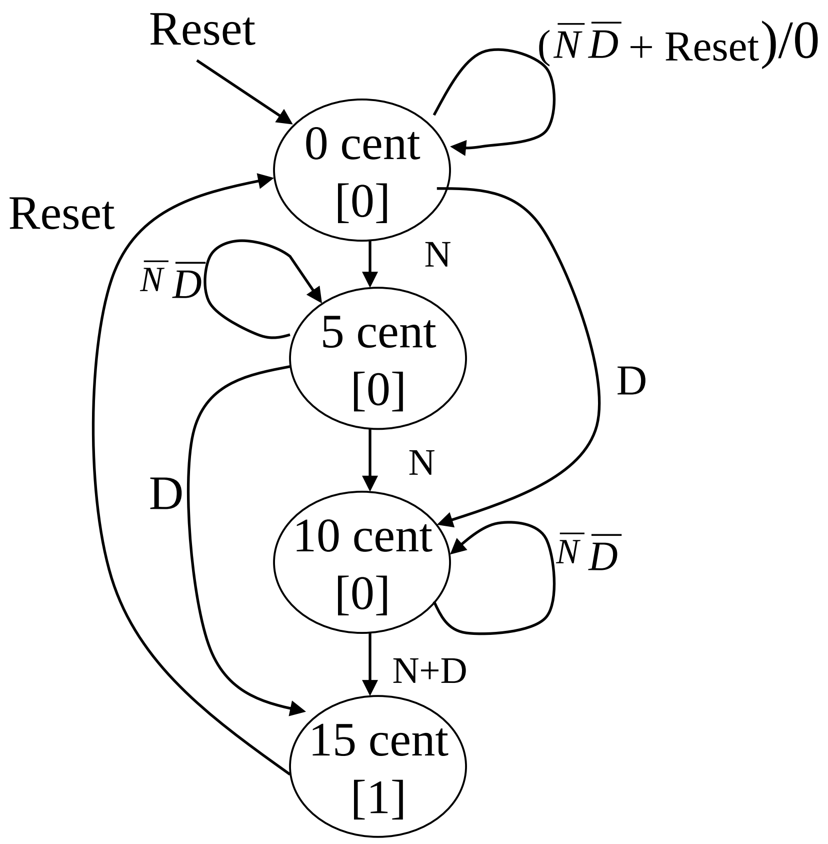
Reduced State Diagram



Example #3 – Mealy vs. Moore


Reset / 0
$(\overline{N}\,\overline{D} + \text{Reset})/0$
0 cent
Reset / 0
$\overline{N}\,\overline{D}/0$
N / 0
D / 0
5 cent
D/1
N / 0
10 cent
$\overline{N}\,\overline{D}/0$
N+D/1
15 cent

Mealy machine


Reset
$(\overline{N}\,\overline{D} + \text{Reset})/0$
0 cent [0]
Reset
$\overline{N}\,\overline{D}$
N
5 cent [0]
D
N
D
10 cent [0]
$\overline{N}\,\overline{D}$
N+D
15 cent [1]

Moore machine

Example #3 – Transition Table

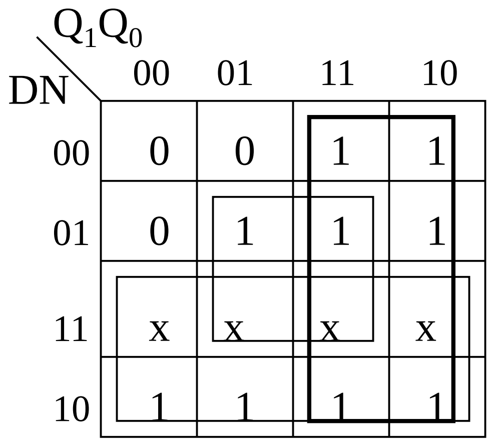
Present State		Inputs		Next State		Moore Output	Mealy Output
Q_1	Q_2	D	N	Q_1^+	Q_2^+	Open	Open
0	0	0	0	0	0	0	0
		0	1	0	1	0	0
		1	0	1	0	0	0
		1	1	x	x	x	x
0	1	0	0	0	1	0	0
		0	1	1	0	0	0
		1	0	1	1	0	1
		1	1	x	x	x	x
1	0	0	0	1	0	0	0
		0	1	1	1	0	1
		1	0	1	1	0	1
		1	1	x	x	x	x
1	1	0	0	1	1	1	1
		0	1	1	1	1	1
		1	0	1	1	1	1
		1	1	x	x	x	x

$Q^+ = D$

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

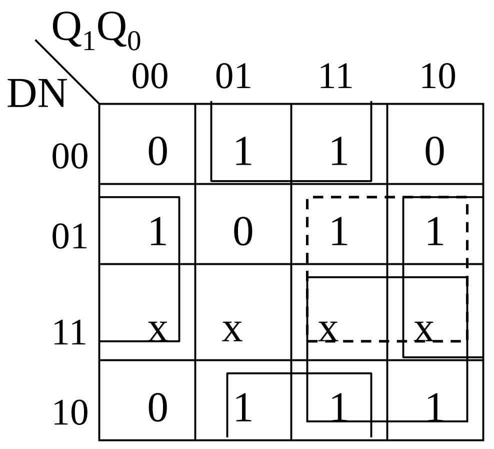
Encoded vending machine state transition table.

Example #3 – Implementation Using D FF



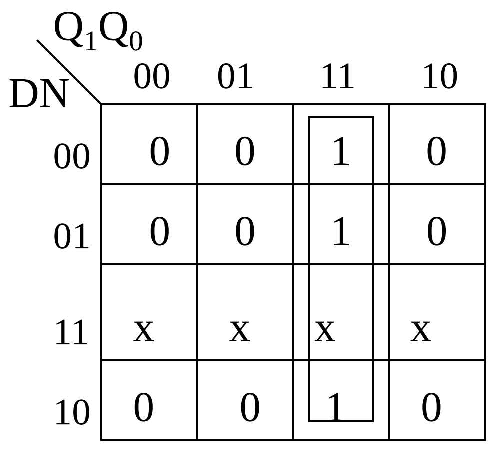
DN \ Q_1Q_0	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	x	x	x	x
10	1	1	1	1

K-map for D_1



DN \ Q_1Q_0	00	01	11	10
00	0	1	1	0
01	1	0	1	1
11	x	x	x	x
10	0	1	1	1

K-map for D_0



DN \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	x	x	x	x
10	0	0	1	0

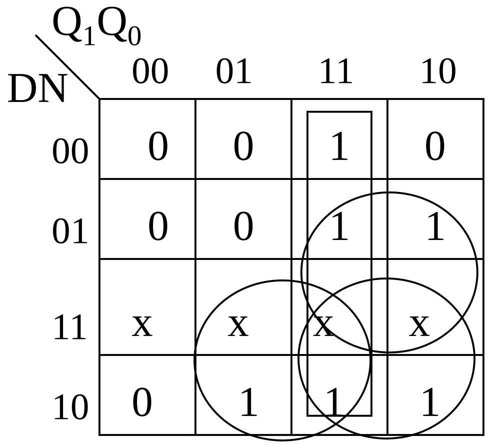
K-map for Open (Moore)

$D_1 = Q_1 + D + Q_0 \cdot N$

$D_0 = N \cdot \overline{Q_0} + Q_0 \cdot \overline{N} + Q_1 \cdot N + Q_1 \cdot D$

Moore → $\text{OPEN} = Q_1 \cdot Q_0$

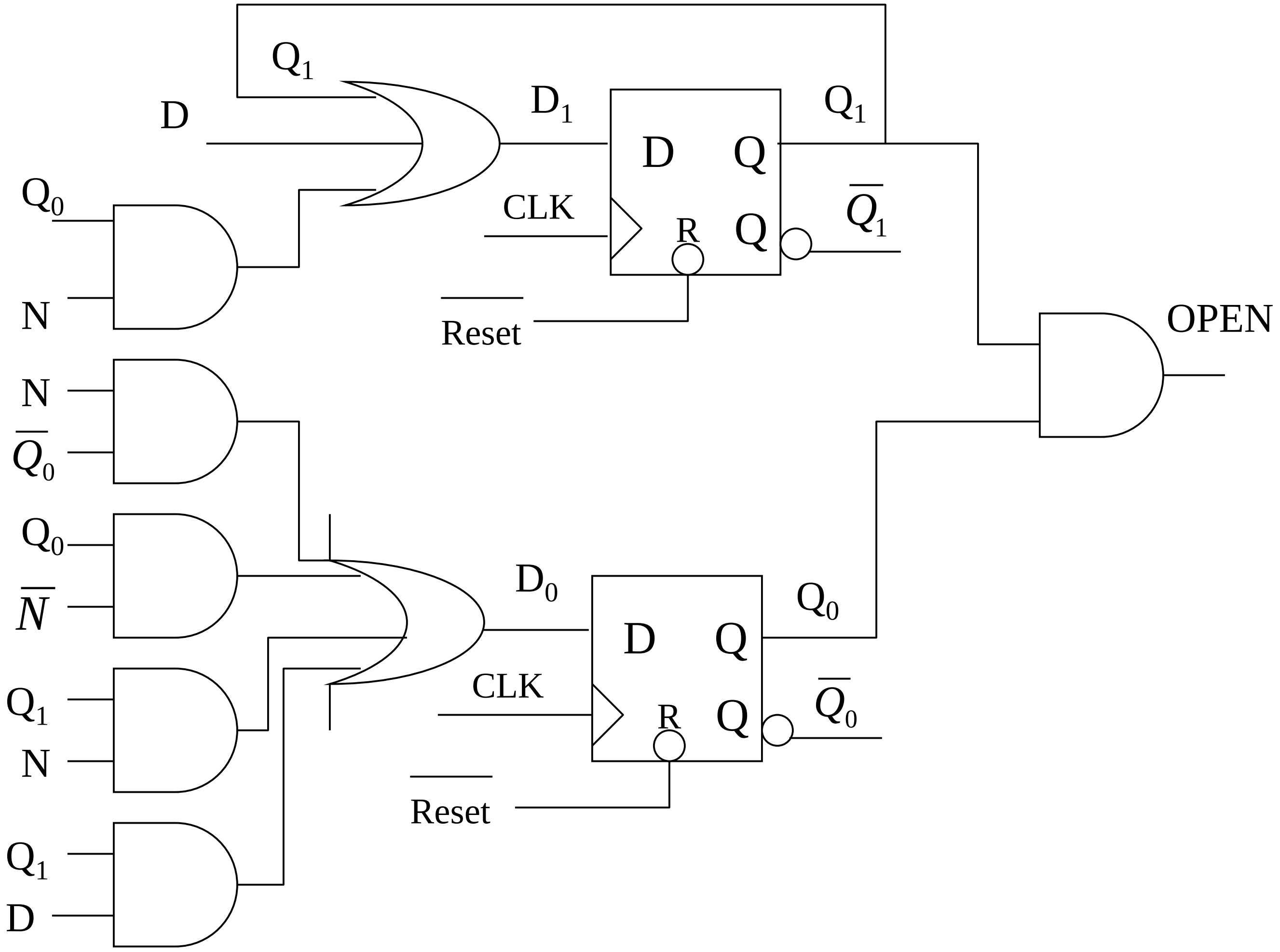
$\text{OPEN} = Q1 \cdot Q_0 + D \cdot Q_0 + D \cdot Q_1 + N \cdot Q_1$ ← Mealy



DN \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	x	x	x	x
10	0	1	1	1

K-map for Open (Mealy)

Example #3 – Circuit Implementation



Vending machine FSM implementation based on D flip-flops (Moore).
For Mealy implementation, only the OPEN function changes.

Example #3 – Table Using J-K FF

Q_1	Q_2	D	N	Q_1^+	Q_2^+	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	x	0	x
		0	1	0	1	0	x	1	x
		1	0	1	0	1	x	0	x
		1	1	x	x	x	x	x	x
0	1	0	0	0	1	0	x	x	0
		0	1	1	0	1	x	x	1
		1	0	1	1	1	x	x	0
		1	1	x	x	x	x	x	x
1	0	0	0	1	0	x	0	0	x
		0	1	1	1	x	0	1	x
		1	0	1	1	x	0	1	x
		1	1	x	x	x	x	x	x
1	1	0	0	1	1	x	0	x	0
		0	1	1	1	x	0	x	0
		1	0	1	1	x	0	x	0
		1	1	x	x	x	x	x	x

J-K Excitation

Q	Q^+	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Remapped next-state functions for the vending machine example.

Example #3 – K-Maps Using J-K FF

DN \ Q_1Q_0	00	01	11	10
00	0	0	x	x
01	0	1	x	x
11	x	x	x	x
10	1	1	x	x

K-map for J_1

DN \ Q_1Q_0	00	01	11	10
00	x	x	0	0
01	x	x	0	0
11	x	x	x	x
10	x	x	0	0

K-map for K_1

DN \ Q_1Q_0	00	01	11	10
00	0	x	x	0
01	1	x	x	1
11	x	x	x	x
10	0	x	x	1

K-map for J_0

DN \ Q_1Q_0	00	01	11	10
00	x	0	0	x
01	x	1	0	x
11	x	x	x	x
10	x	0	0	x

K-map for K_0

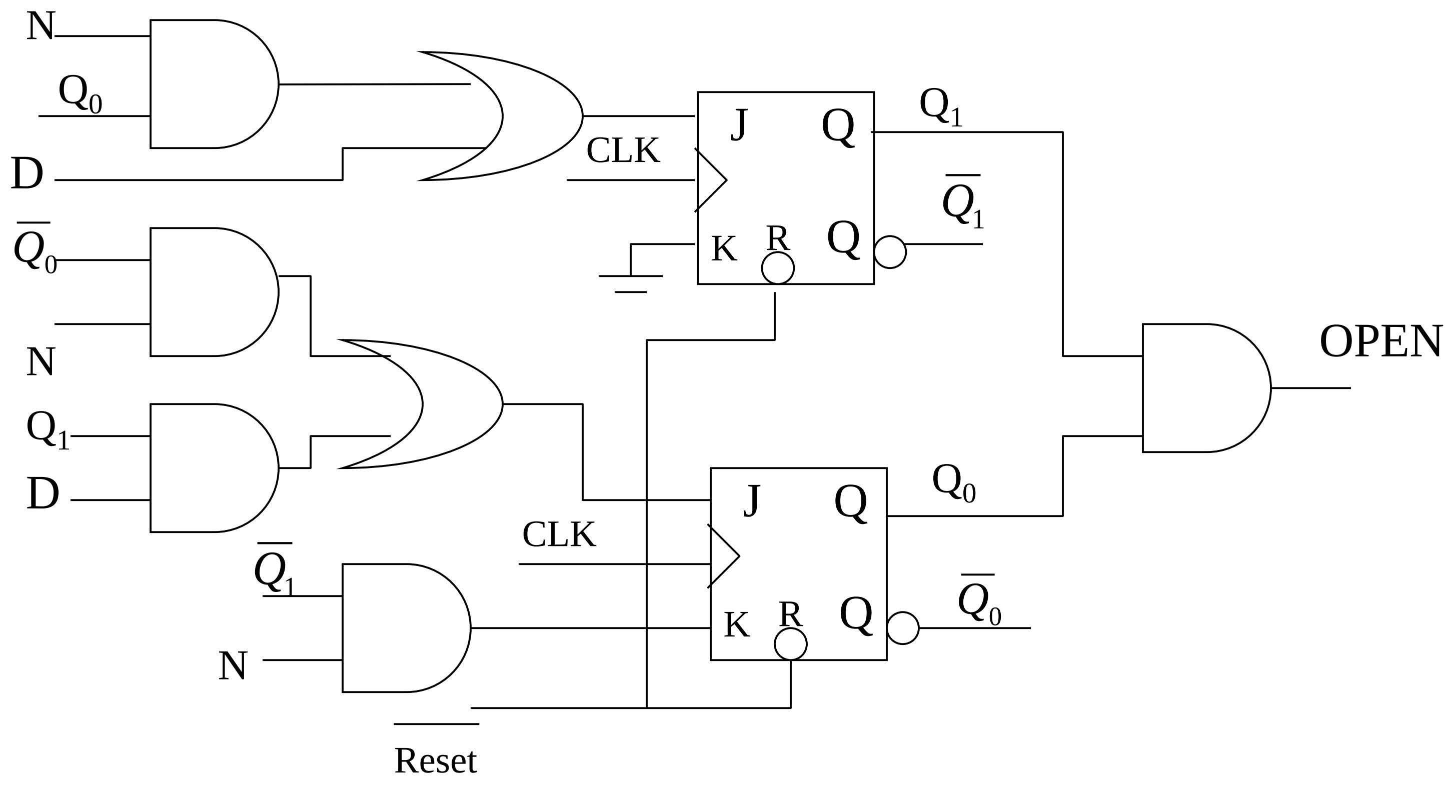
$$J_1 = D + Q_0 \cdot N$$

$$K_1 = 0$$

$$J_0 = \overline{Q_0} \cdot N + Q_1 \cdot D$$

$$K_0 = \overline{Q_1} \cdot N$$

Example #3 – Circuit Implementation



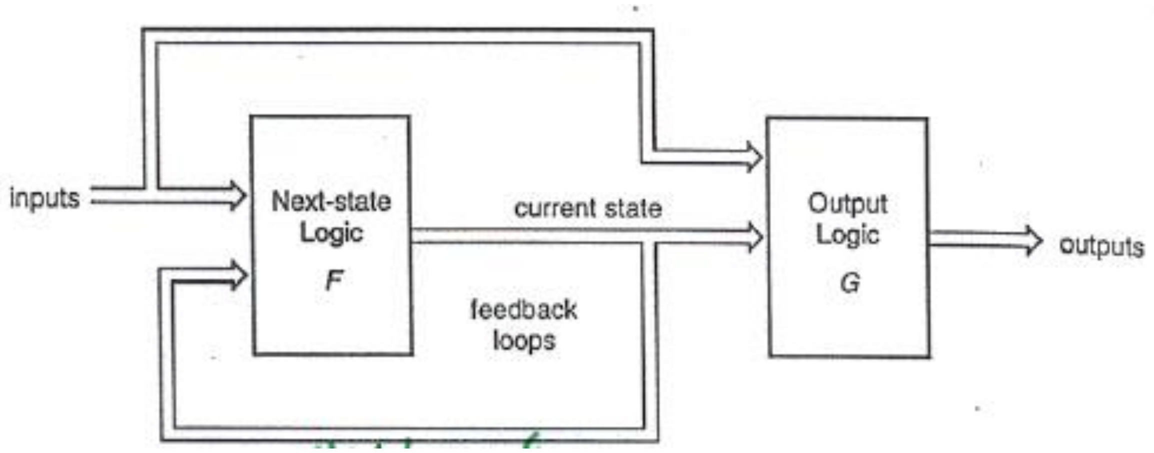
J-K flip-flop implementation for the vending machine example (Moore).

Similarly, a Mealy implementation; only the OPEN function changes.

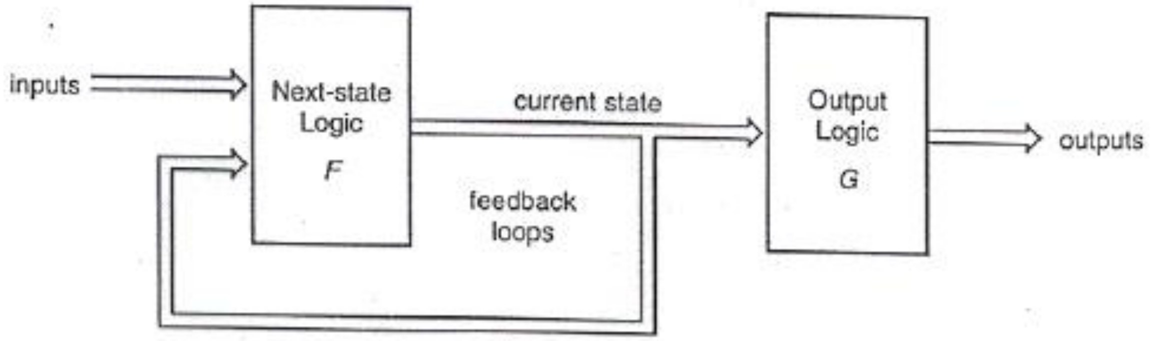
Basis of Asynchronous Circuits

Mealy vs. Moore Models

- The feedbacks provide the concept of states.
  - n feedback lines provide up to 2^n states
- Mealy

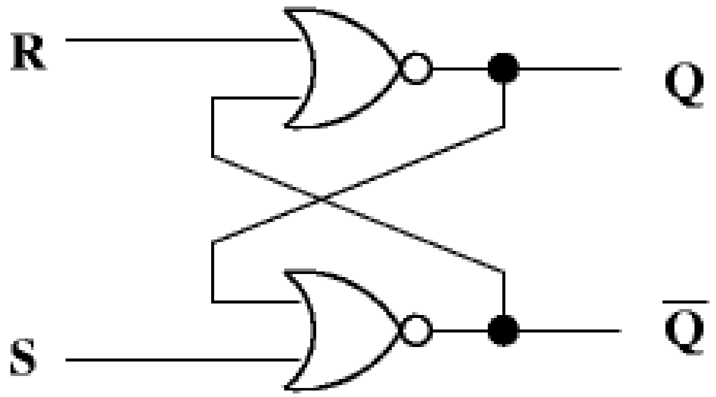


- Moore



Design Analysis

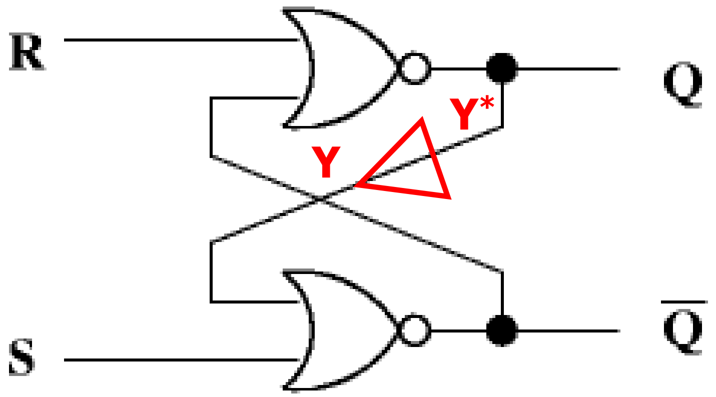
- Analyze this asynchronous circuit



- Analysis Steps
  1. Find and break the (minimum number of) loops
  2. Write the functions for outputs and broken feedbacks
  3. Construct/optimize the transition (flow) table
  4. Identify the stable states, races, etc.

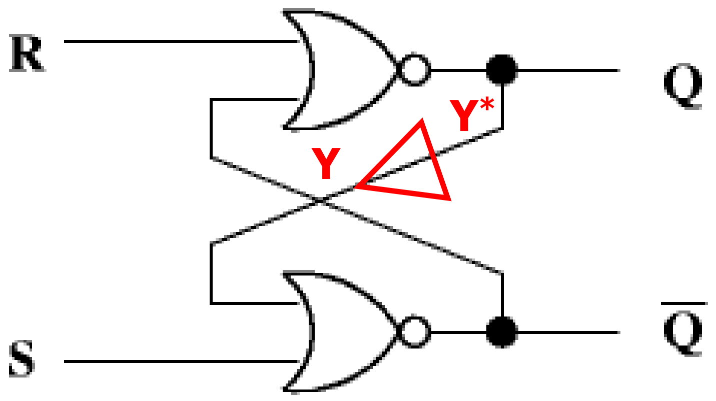
Design Analysis – Step 1

- Find and break the (minimum number of) loops
  - There is only one feedback loop here



Design Analysis – Step 2

- Write the functions for outputs and broken feedbacks



$$Y^* = \overline{R + \overline{Y} \cdot \overline{S}} = \overline{R} \cdot (Y + S) = \overline{R} \cdot Y + \overline{R} \cdot S$$

$$Q = Y^* = \overline{R} \cdot Y + \overline{R} \cdot S$$

$$\overline{Q} = \overline{Y + S} = \overline{Y} \cdot \overline{S}$$

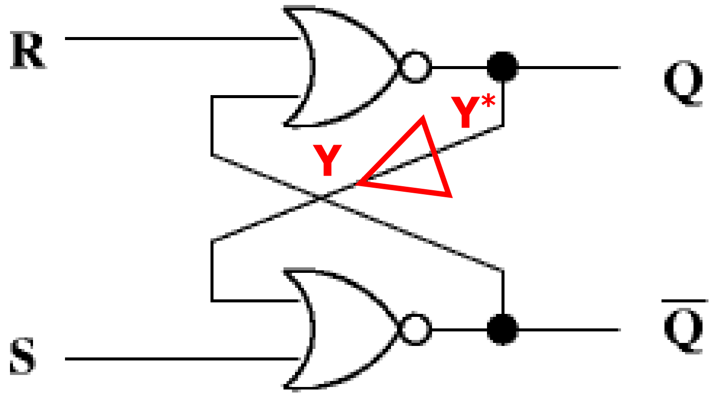
Design Analysis – Step 3

- Construct/optimize the transition (flow) table

$$Y^* = \overline{R + \overline{Y} \cdot \overline{S}} = \overline{R} \cdot (Y + S) = \overline{R} \cdot Y + \overline{R} \cdot S$$

$$Q = Y^* = \overline{R} \cdot Y + \overline{R} \cdot S$$

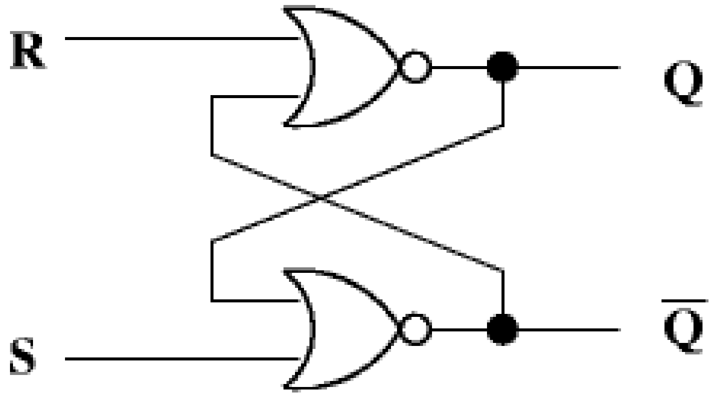
$$\overline{Q} = \overline{Y + S} = \overline{Y} \cdot \overline{S}$$



	SR			
y	00	01	11	10
0	**0,01**	**0,01**	**0,00**	**1,10**
1	**1,10**	**0,00**	**0,00**	**1,10**
	y*,QQ’			

Design Analysis – Step 4

- Identify the stable states, transitions, races, etc.



Hold 0

Reset

Set

Hold 1

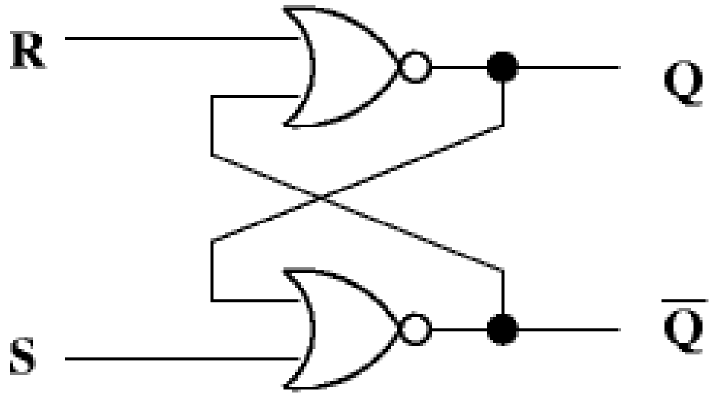
	SR			
y	00	01	11	10
0	0,01	0,01	0,00	1,10
1	1,10	0,00	0,00	1,10
	y*,QQ'			

Fundamental Mode Assumption

- The inputs change one at a time, allowing enough time between successive changes for the circuit to settle down into a stable state.
- Example: SR=11 changing to SR=00 happens one of these ways:
  - 11 → 10 → 00
  - 11 → 01 → 00
- Critical race exists if the circuit may end up in different states unpredictably (e.g. depending on the actual delay of transistors/gates) for the same inputs change.

Design Analysis – Step 4 (cont.)

- Identify the stable states, transitions, races, etc.
  - 11 → 10 → 00
  - 11 → 01 → 00



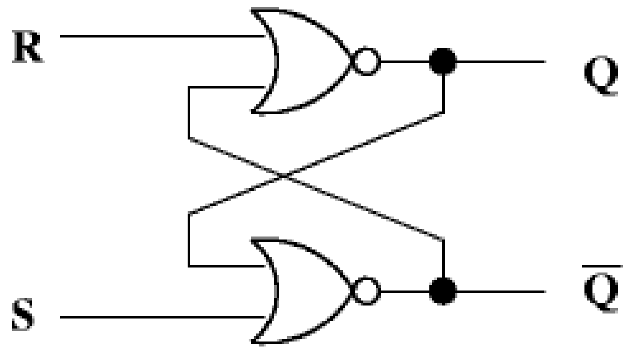
Critical race (due to input change)
Unpredictable Final State

Unstable (Forbidden State)

	SR			
y	00	01	11	10
0	**0,01**	**0,01**	**0,00**	**1,10**
1	**1,10**	**0,00**	**0,00**	**1,10**
	y*,QQ'			

Design Analysis – Overall Functionality

- Familiar Behavior of SR Latch



Hold 0

Hold 1

Reset

Undefined

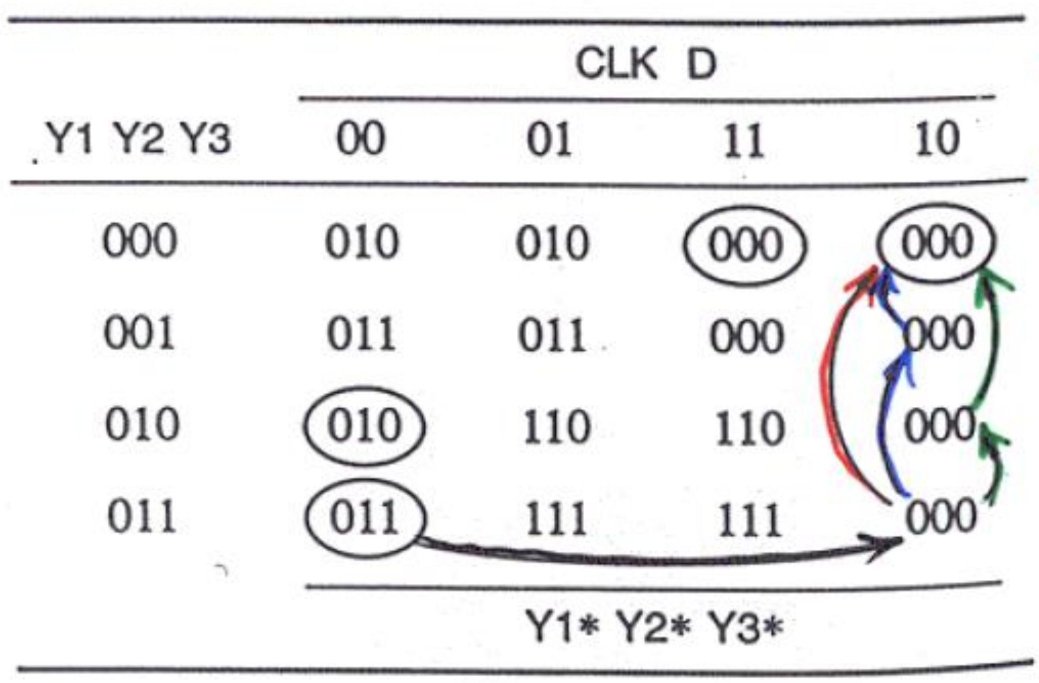
Set

	SR			
y	00	01	11	10
0	**0,01**	**0,01**	**0,00**	**1,10**
1	**1,10**	**0,00**	**0,00**	**1,10**
	y*,QQ′			

S	***R***	***Q****
0	0	*Q*
0	1	0
1	0	1
1	1	*Undefined* (0 0)

Race Due to State-Code Change

- Portion of a table showing **non-critical race** (the final state does not depend on the order in which the state variables (code) change



	CLK D			
Y1 Y2 Y3	00	01	11	10
000	010	010	000	000
001	011	011	000	000
010	010	110	110	000
011	011	111	111	000
	Y1* Y2* Y3*			

Theory:

→ : 011 → 000

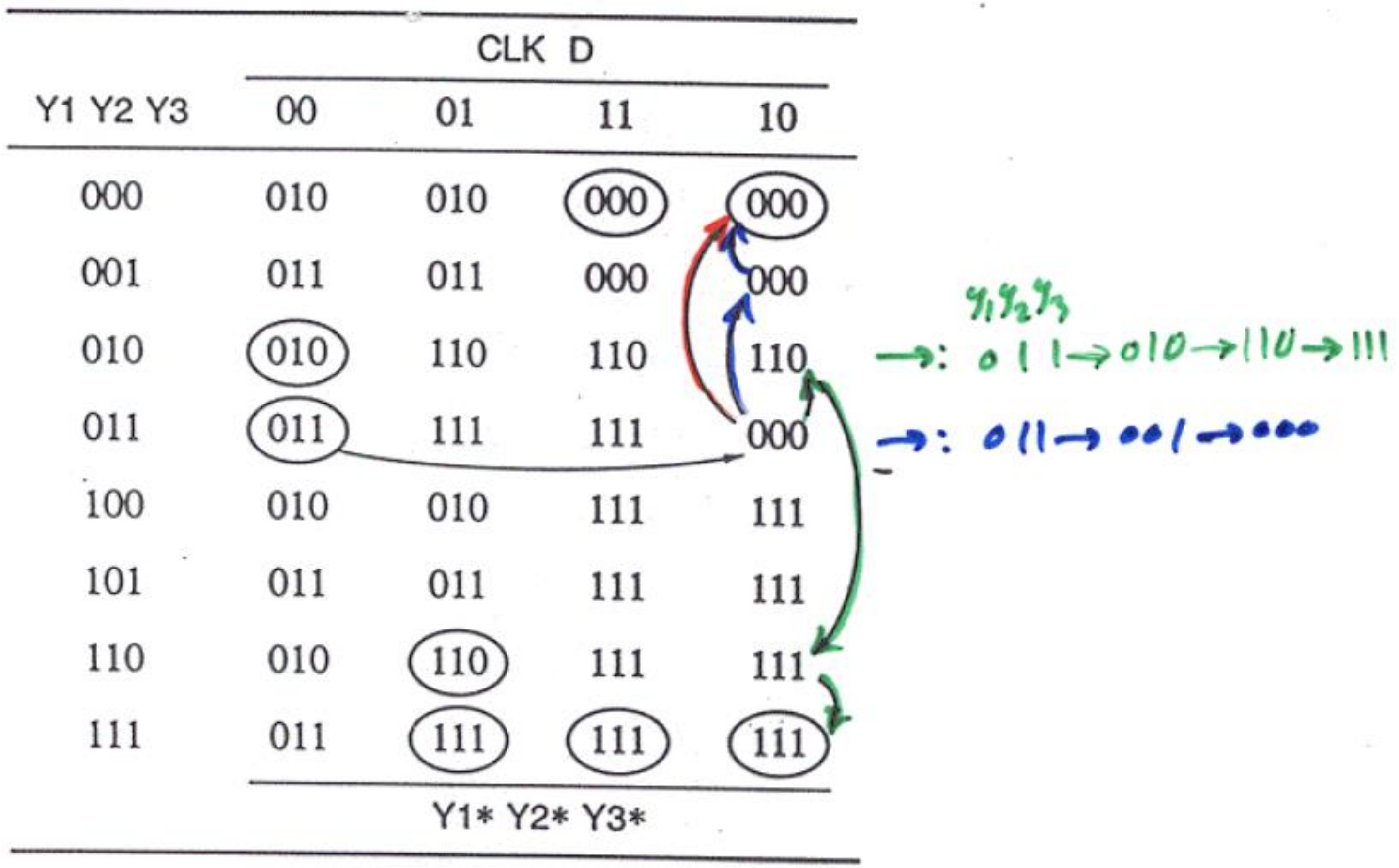
Reality:
$y_1 y_2 y_3$

→ : 011 → 010 → 000

→ : 011 → 001 → 000

Race Due to State-Code Change (cont.)

- Portion of a table showing **critical race** (the final state depends on the delay of components



	CLK D			
Y1 Y2 Y3	00	01	11	10
000	010	010	$\boxed{000}$	$\boxed{000}$
001	011	011	000	000
010	$\boxed{010}$	110	110	110
011	$\boxed{011}$	111	111	000
100	010	010	111	111
101	011	011	111	111
110	010	$\boxed{110}$	111	111
111	011	$\boxed{111}$	$\boxed{111}$	$\boxed{111}$
	Y1* Y2* Y3*			

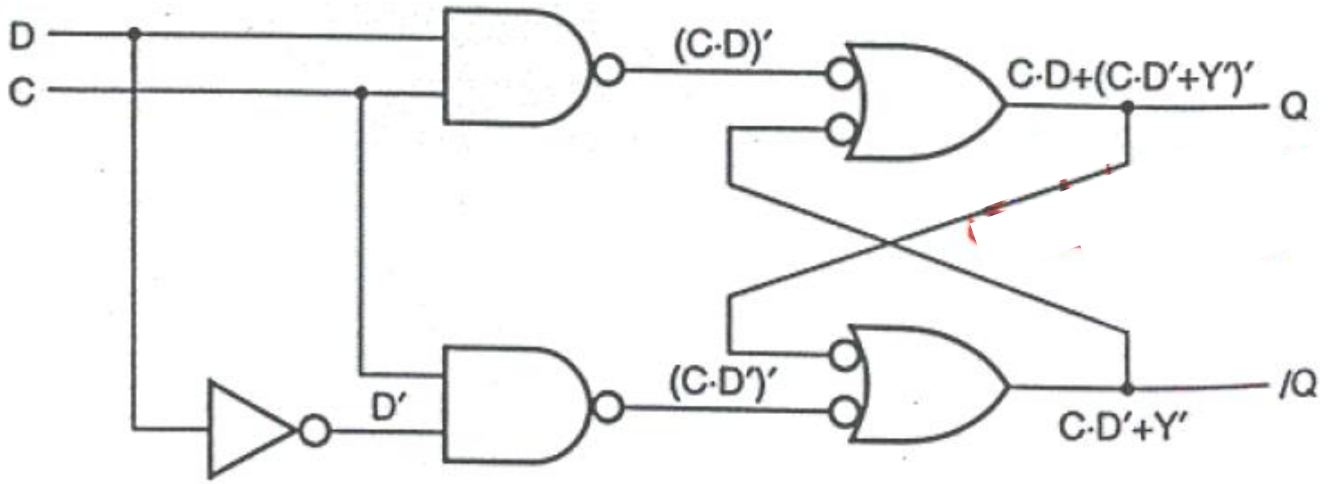
$y_1y_2y_3$

$\rightarrow$: $011 \rightarrow 010 \rightarrow 110 \rightarrow 111$

$\rightarrow$: $011 \rightarrow 001 \rightarrow 000$

Design Analysis – D Latch

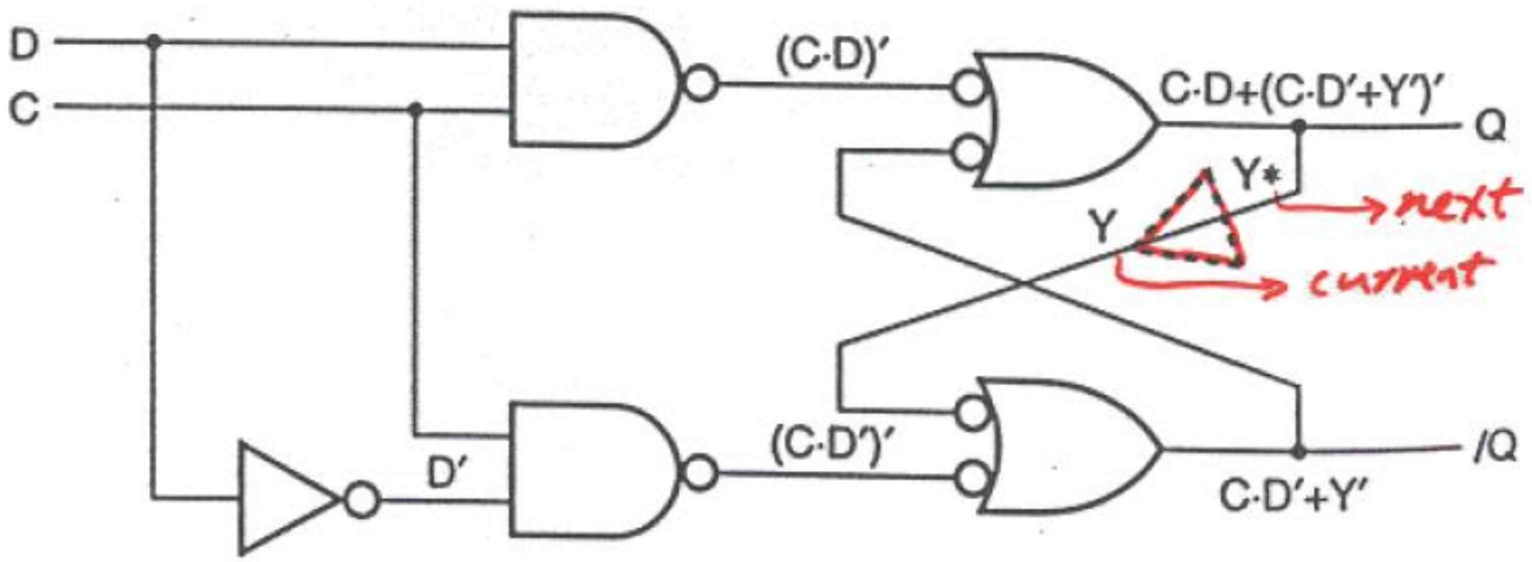
- Analyze this asynchronous circuit



- Analysis Steps
  1. Find and break the (minimum number of) loops
  2. Write the functions for outputs and broken feedbacks
  3. Construct/optimize the transition (flow) table
  4. Identify the stable states, races, etc.

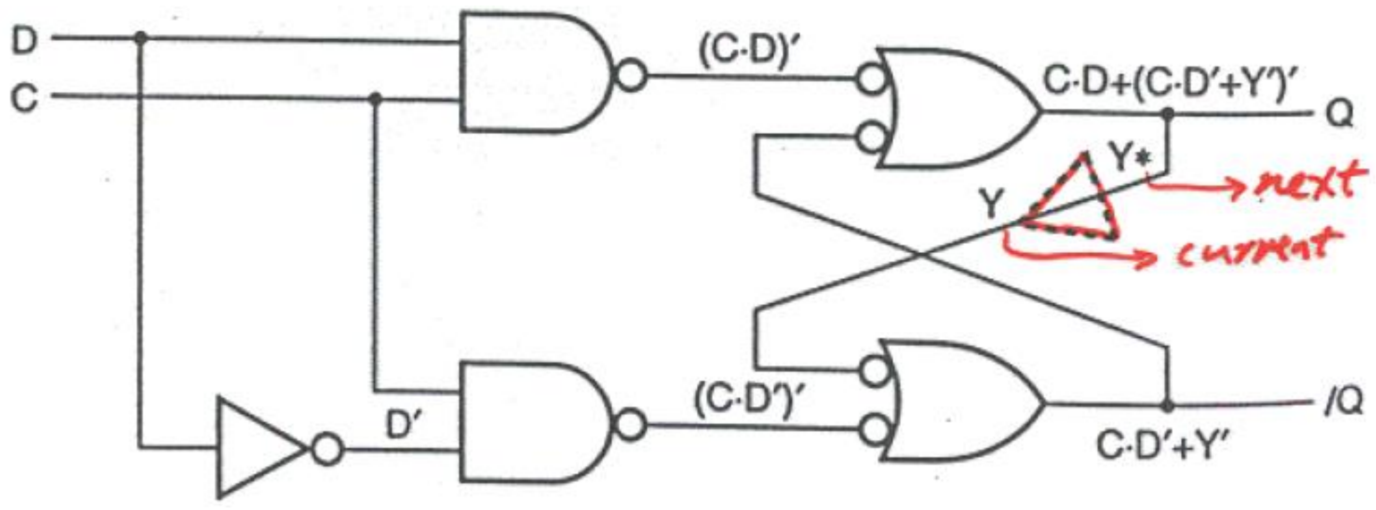
Design Analysis – Step 1

- Find and break the (minimum number of) loops
  - There is only one feedback loop here



Design Analysis – Step 2

- Write the functions for outputs and broken feedbacks



$$Y^* = (C\cdot D) + (C\cdot D' + Y')' = C\cdot D + C'\cdot Y + D\cdot Y$$

$$Q = Y^* = C\cdot D + C'\cdot Y + D\cdot Y$$

$$\overline{Q} = C\cdot D' + Y'$$

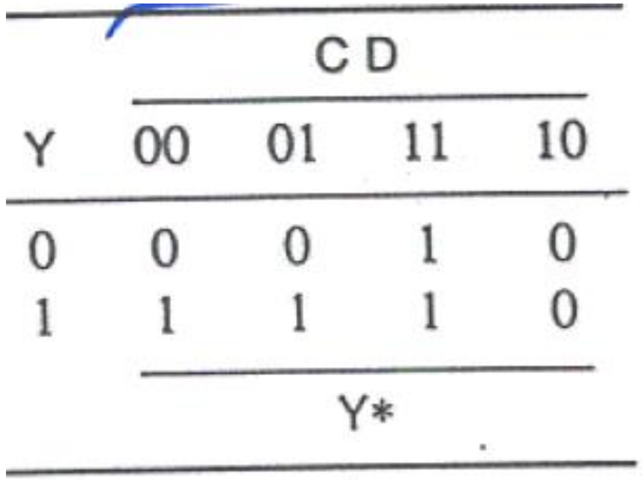
Design Analysis – Step 3

- Construct/optimize the transition (flow) table

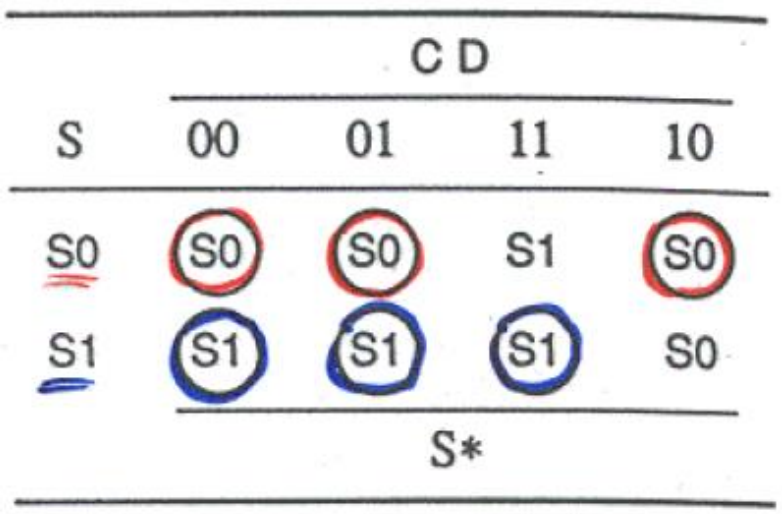
$$Y^{*} = (C \cdot D) + (C \cdot D' + Y')' = C \cdot D + C' \cdot Y + D \cdot Y$$

$$Q = Y^{*} = C \cdot D + C' \cdot Y + D \cdot Y$$

$$\overline{Q} = C \cdot D' + Y'$$



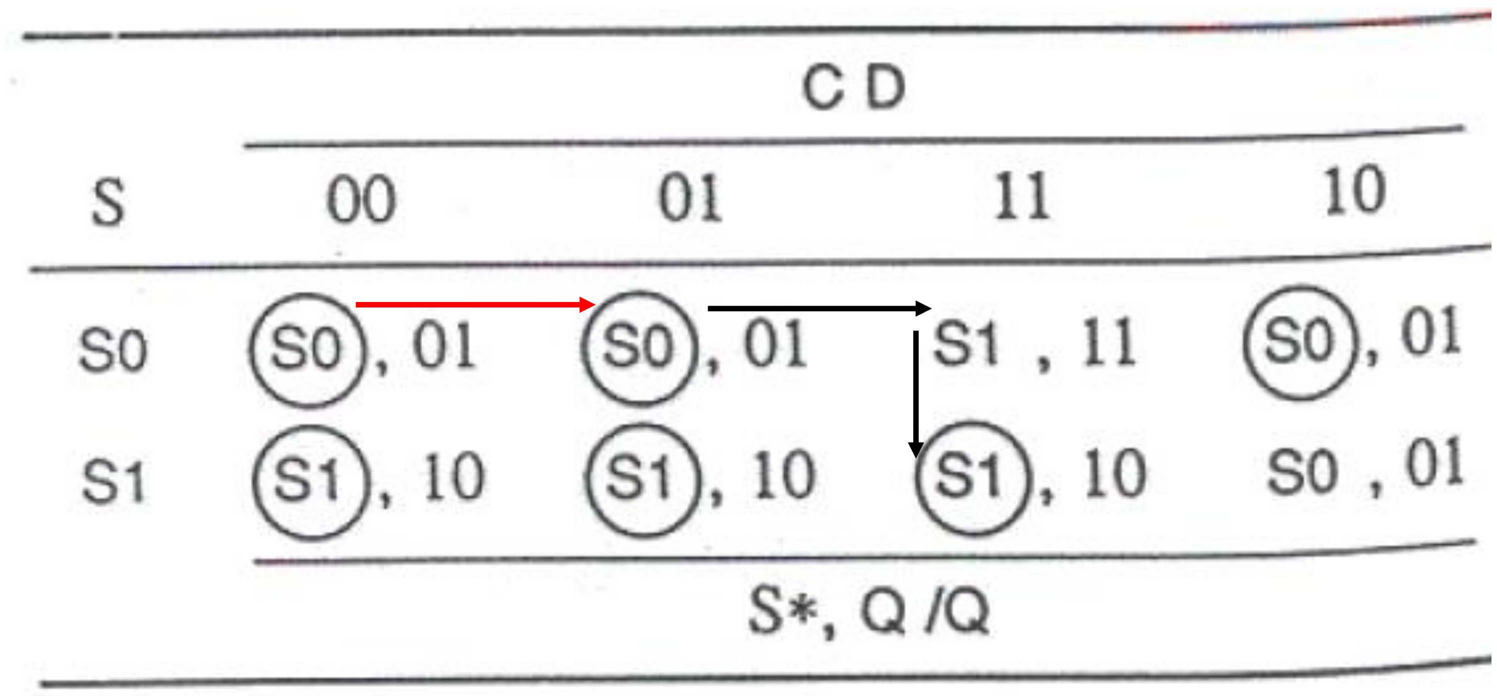
	CD			
Y	00	01	11	10
0	0	0	1	0
1	1	1	1	0
	Y*			



	CD			
S	00	01	11	10
S0	S0	S0	S1	S0
S1	S1	S1	S1	S0
	S*			

Design Analysis – Step 4

- Identify the stable states, transitions, races, etc.
  - CD: 00 → 01
  - CD: 01 → 11

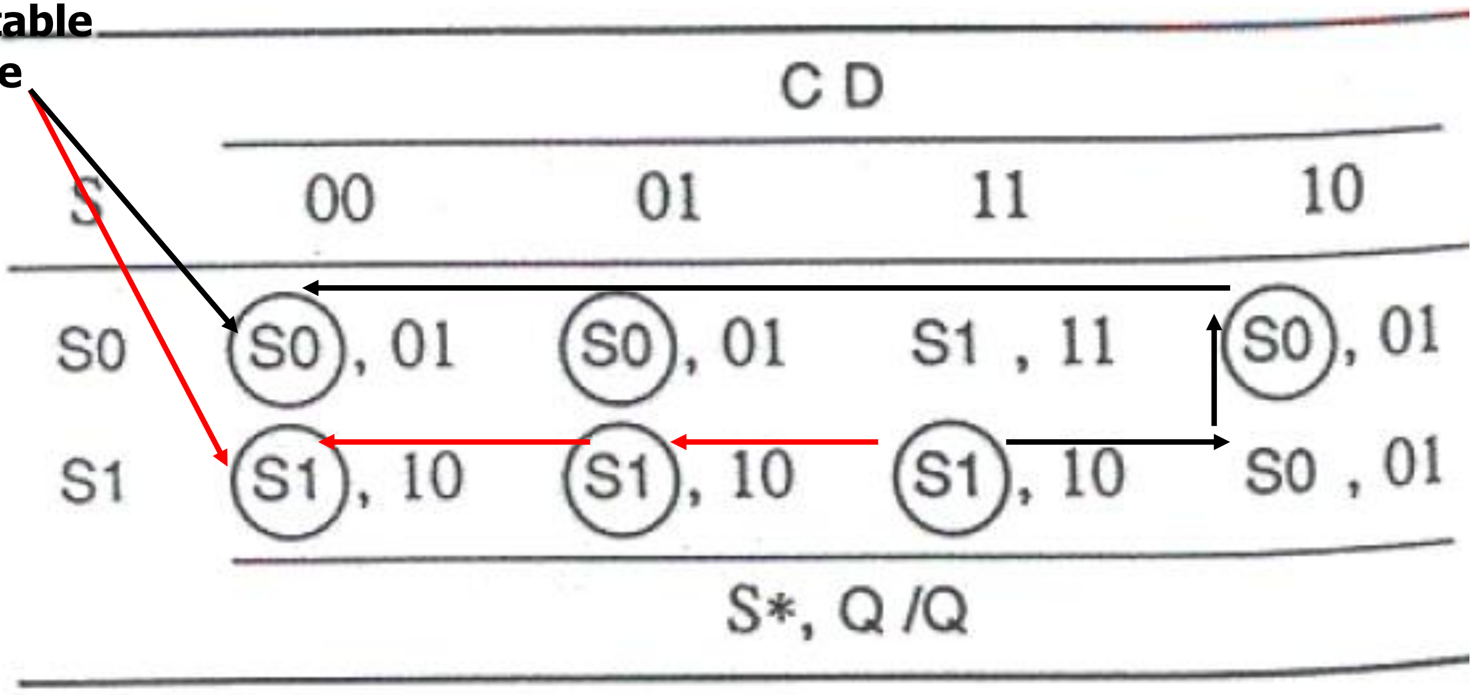


	CD			
S	00	01	11	10
S0	S0, 01	S0, 01	S1 , 11	S0, 01
S1	S1, 10	S1, 10	S1, 10	S0 , 01
	S*, Q /Q			

Design Analysis – Step 4 (cont.)

- Possibility of critical race (due to multiple bit input change) exists
  - CD: 11 → 01 → 00
  - CD: 11 → 10 → 00

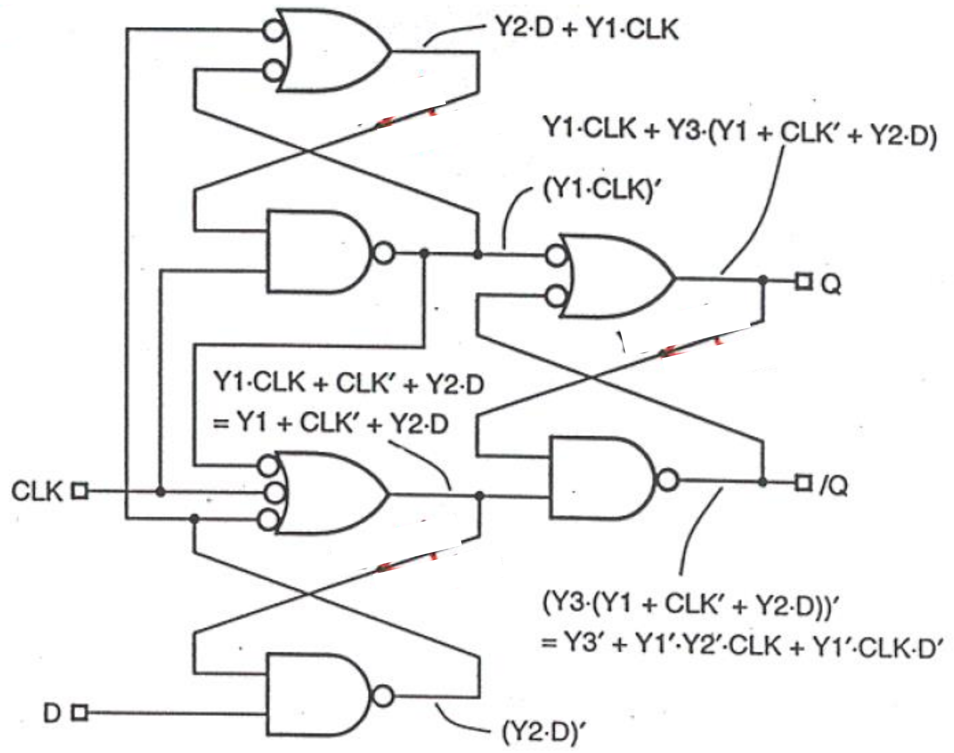
Critical race (due to input change)
Unpredictable Final State



S	CD 00	01	11	10
S0	S0, 01	S0, 01	S1 , 11	S0, 01
S1	S1, 10	S1, 10	S1, 10	S0 , 01
	S*, Q /Q			

Design Analysis – Positive Edge-Triggered D FF

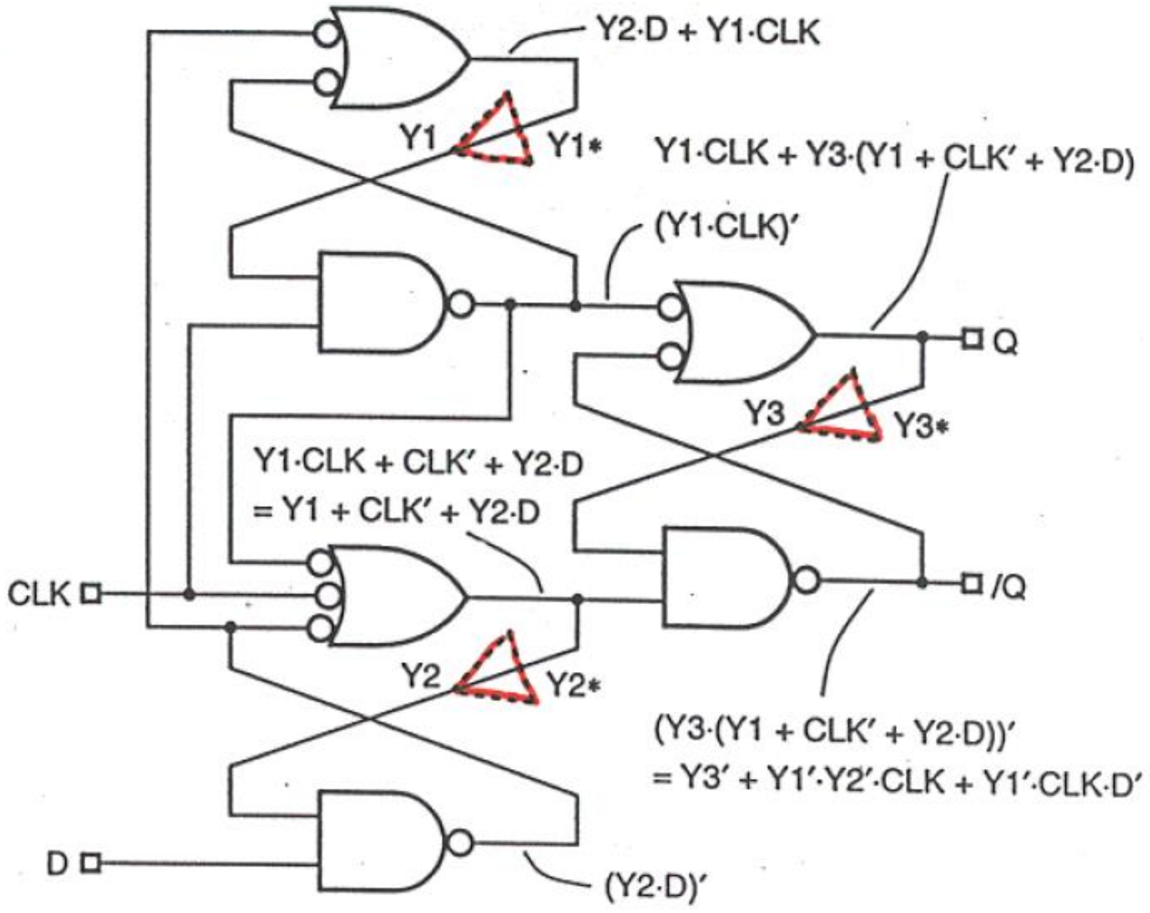
- Analyze this asynchronous circuit



- Analysis Steps
  1. Find and break the (minimum number of) loops
  2. Write the functions for outputs and broken feedbacks
  3. Construct/optimize the transition (flow) table
  4. Identify the stable states, races, etc.

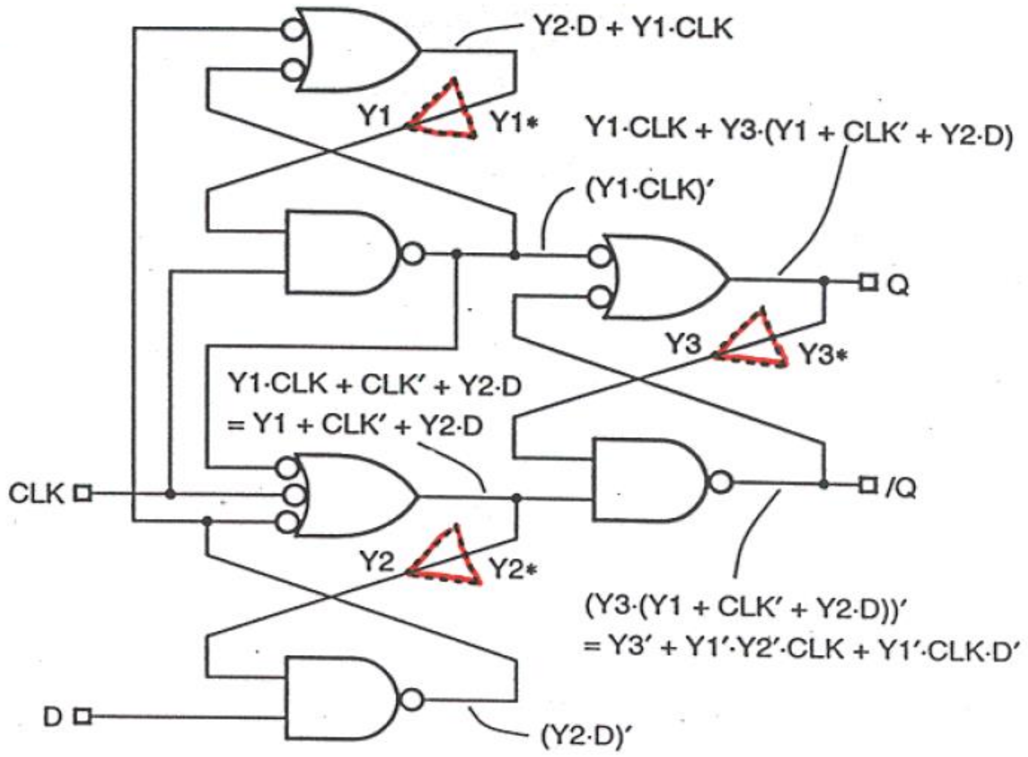
Design Analysis – Step 1

- Find and break the (minimum number of) loops
  - There are three feedback loops here



Design Analysis – Step 2

- Write the functions for outputs and broken feedbacks



$$Y1^* = Y2 \cdot D + Y1 \cdot CLK$$

$$Y2^* = Y1 + CLK' + Y2 \cdot D$$

$$Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$Q = Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$\overline{Q} = Y3' + Y1' \cdot Y2' \cdot CLK + Y1' \cdot CLK \cdot D'$$

Design Analysis – Step 3

- Construct/optimize the transition (flow) table

$$Y1^* = Y2 \cdot D + Y1 \cdot CLK$$

$$Y2^* = Y1 + CLK' + Y2 \cdot D$$

$$Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$Q = Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$\overline{Q} = Y3' + Y1' \cdot Y2' \cdot CLK + Y1' \cdot CLK \cdot D'$$

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0				
0 0 1				
0 1 0				
0 1 1				
1 0 0				
1 0 1				
1 1 0				
1 1 1				

Design Analysis – Step 3

- Construct/optimize the transition (flow) table

$$Y1^* = Y2 \cdot D + Y1 \cdot CLK$$

$$Y2^* = Y1 + CLK' + Y2 \cdot D$$

$$Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$Q = Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$\overline{Q} = Y3' + Y1' \cdot Y2' \cdot CLK + Y1' \cdot CLK \cdot D'$$

Use logic equations to fill in the table

For example:
Let CLK = 0 and D = 0
and state = 0 0 0
Y1* = Y2&D + Y1&CLK
= 0 & 0 + 0 & 0 = 0
Y2* = Y1 + CLK’ + Y2&D
= 0 + 0’ + 0&0 = 1
Y3* = Y1&CLK + Y1&Y3 +
Y3&CLK’ + Y2&Y3&D
= 0&0 + 0&0 + 0&1 + 0&0&0 = 0
Q = Y3* = 0
Q’ = Y3’ + Y1’&Y2’&CLK + Y1’&CLK&D’
= 1 + 1&1&0 + 1&0&1 = 1

So the entry in the table is 010,01

Design Analysis – Step 3

- Construct/optimize the transition (flow) table

$$Y1^* = Y2 \cdot D + Y1 \cdot CLK$$

$$Y2^* = Y1 + CLK' + Y2 \cdot D$$

$$Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$Q = Y3^* = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$$

$$\overline{Q} = Y3' + Y1' \cdot Y2' \cdot CLK + Y1' \cdot CLK \cdot D'$$

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0	010,01			
0 0 1				
0 1 0				
0 1 1				
1 0 0				
1 0 1				
1 1 0				
1 1 1				

Design Analysis – Step 3

- Complete the transition (flow) table

$Y1^{*} = Y2 \cdot D + Y1 \cdot CLK$

$Y2^{*} = Y1 + CLK' + Y2 \cdot D$

$Y3^{*} = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$

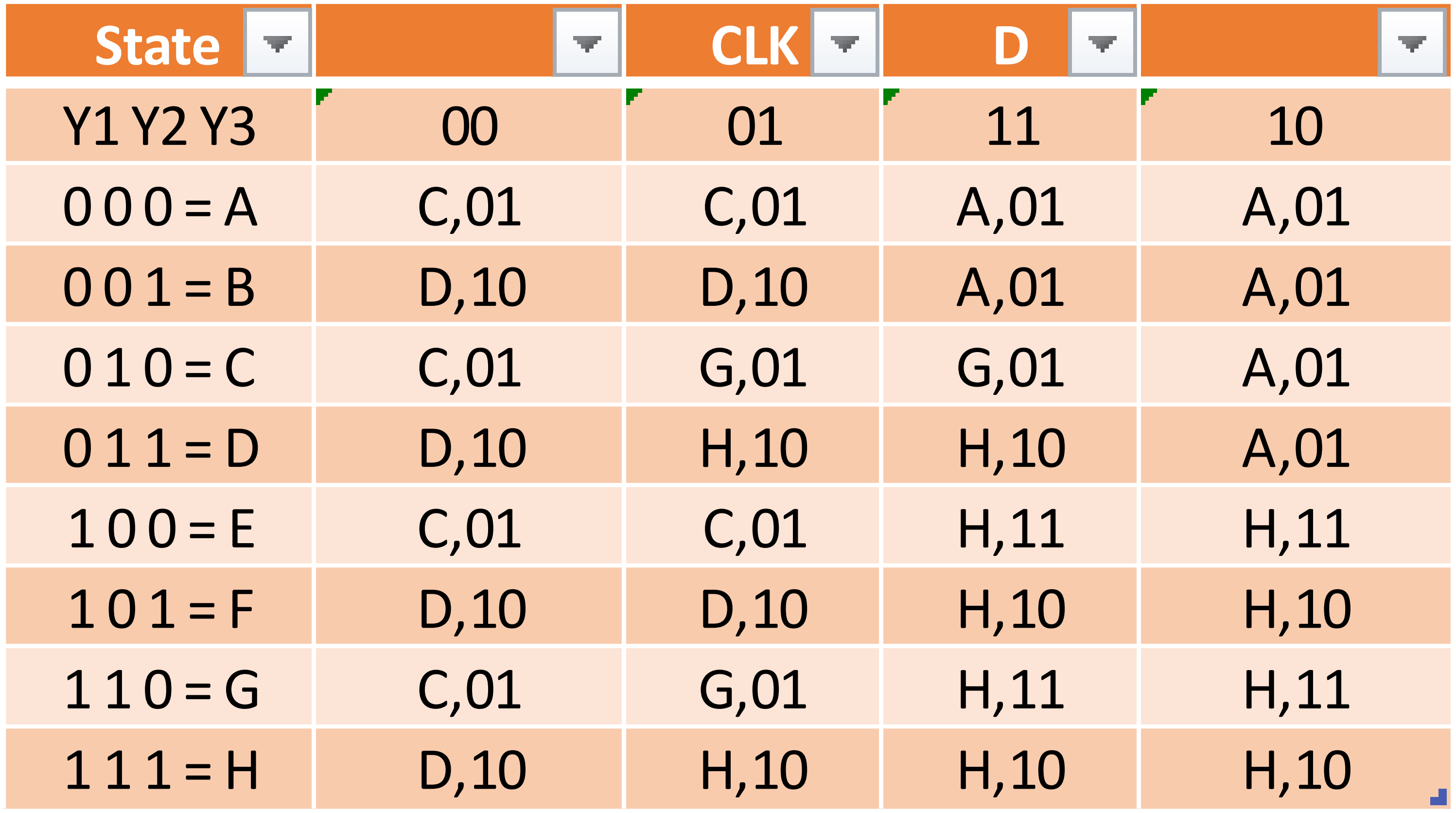
$Q = Y3^{*} = Y1 \cdot CLK + Y1 \cdot Y3 + Y3 \cdot CLK' + Y2 \cdot Y3 \cdot D$

$\overline{Q} = Y3' + Y1' \cdot Y2' \cdot CLK + Y1' \cdot CLK \cdot D'$

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0	010,01	010,01	000,01	000,01
0 0 1	011,10	011,10	000,01	000,01
0 1 0	010,01	110,01	110,01	000,01
0 1 1	011,10	111,10	111,10	000,01
1 0 0	010,01	010,01	111,11	111,11
1 0 1	011,10	011,10	111,10	111,10
1 1 0	010,01	110,01	111,11	111,11
1 1 1	011,10	111,10	111,10	111,10

Design Analysis – Step 3

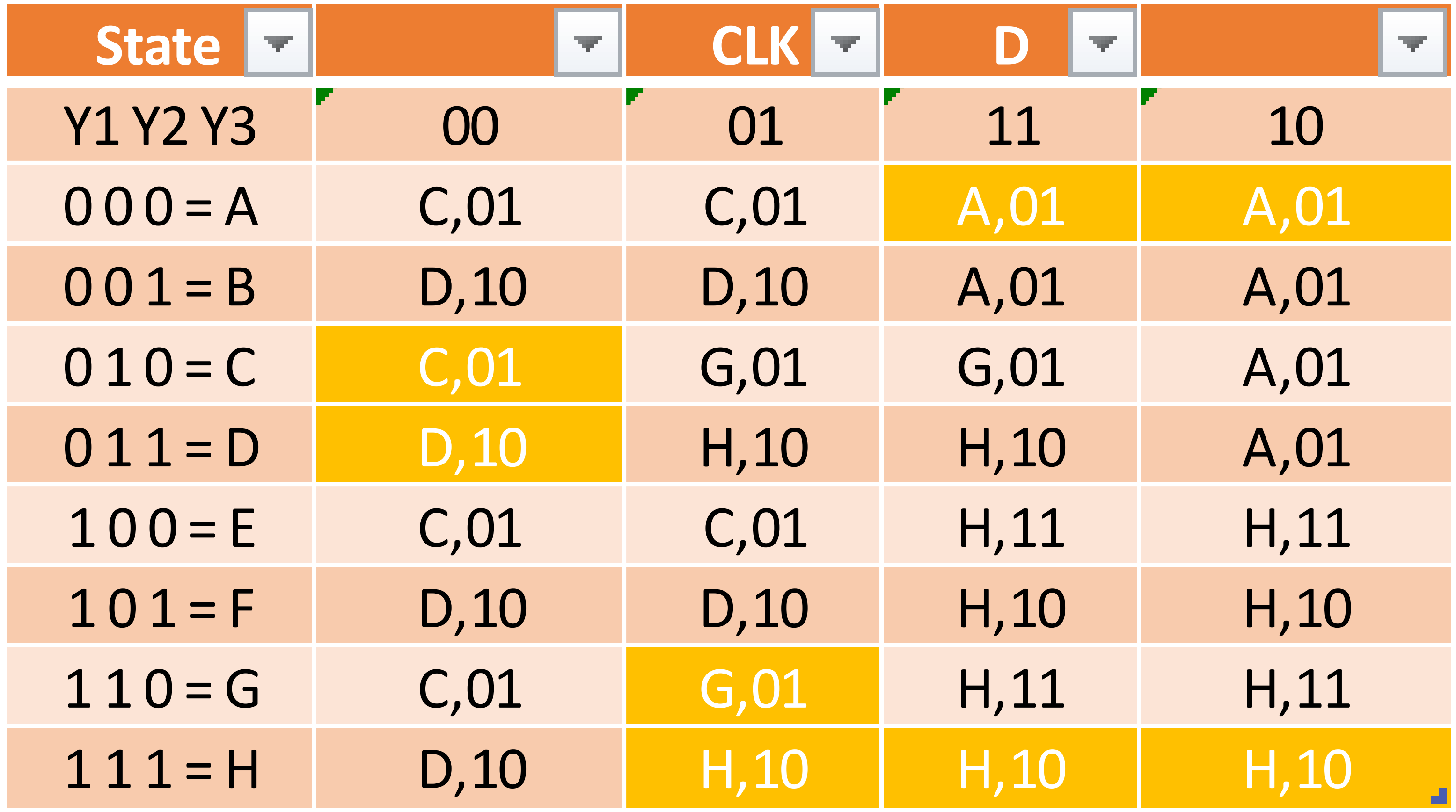
- Rename the states



State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	C,01	A,01	A,01
0 0 1 = B	D,10	D,10	A,01	A,01
0 1 0 = C	C,01	G,01	G,01	A,01
0 1 1 = D	D,10	H,10	H,10	A,01
1 0 0 = E	C,01	C,01	H,11	H,11
1 0 1 = F	D,10	D,10	H,10	H,10
1 1 0 = G	C,01	G,01	H,11	H,11
1 1 1 = H	D,10	H,10	H,10	H,10

Design Analysis – Step 3

- Mark the stable states



State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	C,01	A,01	A,01
0 0 1 = B	D,10	D,10	A,01	A,01
0 1 0 = C	C,01	G,01	G,01	A,01
0 1 1 = D	D,10	H,10	H,10	A,01
1 0 0 = E	C,01	C,01	H,11	H,11
1 0 1 = F	D,10	D,10	H,10	H,10
1 1 0 = G	C,01	G,01	H,11	H,11
1 1 1 = H	D,10	H,10	H,10	H,10

Design Analysis – Step 3

- Remove rows with no stable states

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	C,01	A,01	A,01
0 0 1 = B	D,10	D,10	A,01	A,01
0 1 0 = C	C,01	G,01	G,01	A,01
0 1 1 = D	D,10	H,10	H,10	A,01
1 0 0 = E	C,01	C,01	H,11	H,11
1 0 1 = F	D,10	D,10	H,10	H,10
1 1 0 = G	C,01	G,01	H,11	H,11
1 1 1 = H	D,10	H,10	H,10	H,10

Rows with no stable state

Design Analysis – Step 3

- Perform transitive closure

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	~~C,01~~ G,01	A,01	A,01
0 1 0 = C	C,01	G,01	~~G,01~~ H,10	A,01
0 1 1 = D	D,10	H,10	H,10	A,01
1 1 0 = G	C,01	G,01	H,11	H,11
1 1 1 = H	D,10	H,10	H,10	H,10

Design Analysis – Step 3

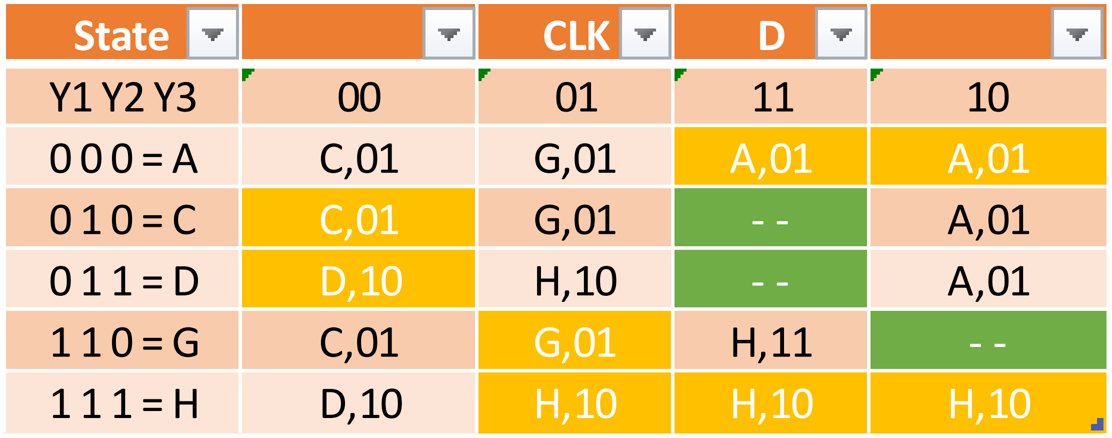
- Remove states with no adjacent stable states
- Every unstable state must be adjacent to at least one stable state due to fundamental mode assumption

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	G,01	A,01	A,01
0 1 0 = C	C,01	G,01	- -	A,01
0 1 1 = D	D,10	H,10	- -	A,01
1 1 0 = G	C,01	G,01	H,11	- -
1 1 1 = H	D,10	H,10	H,10	H,10

States with no adjacent stable state

Design Analysis – Step 4

- Identify the stable states, transitions, races, etc.

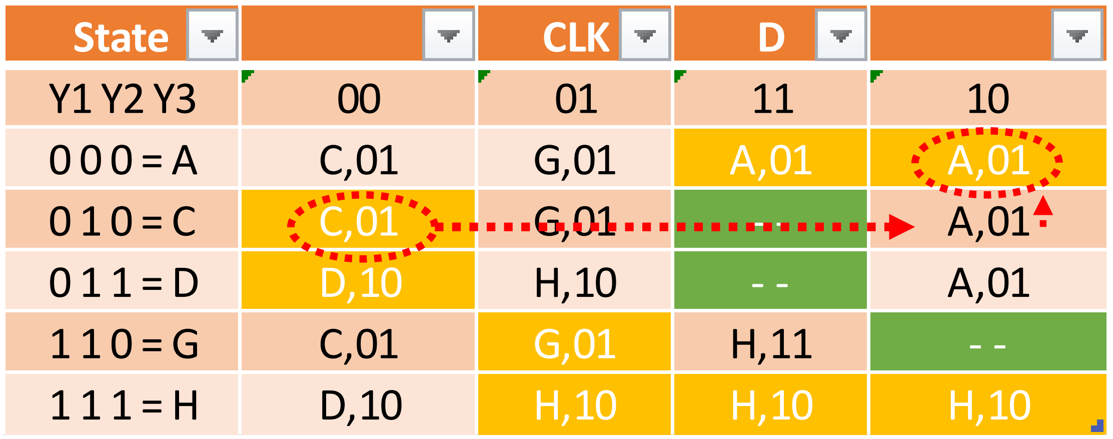


State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	G,01	A,01	A,01
0 1 0 = C	C,01	G,01	- -	A,01
0 1 1 = D	D,10	H,10	- -	A,01
1 1 0 = G	C,01	G,01	H,11	- -
1 1 1 = H	D,10	H,10	H,10	H,10

Let D = 0, what happens when CLK= 0->1?

Design Analysis – Step 4

- Identify the stable states, transitions, races, etc.



State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	G,01	A,01	A,01
0 1 0 = C	C,01	G,01	- -	A,01
0 1 1 = D	D,10	H,10	- -	A,01
1 1 0 = G	C,01	G,01	H,11	- -
1 1 1 = H	D,10	H,10	H,10	H,10

Let D = 0, what happens when CLK= 0->1?
Starting at inputs CLK=0 D=0, there are two stable states

From C: this transition changes state but not the output.

Design Analysis – Step 4

- Identify the stable states, transitions, races, etc.

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	G,01	A,01	A,01
0 1 0 = C	C,01	G,01	- -	A,01
0 1 1 = D	D,10	H,10	- -	A,01
1 1 0 = G	C,01	G,01	H,11	- -
1 1 1 = H	D,10	H,10	H,10	H,10

Let D = 0, what happens when CLK= 0->1?
Starting at inputs CLK=0 D=0, there are two stable states

From D: this transition changes state to A like C but the output changes from Q=1 to Q=0

Design Analysis – Step 4

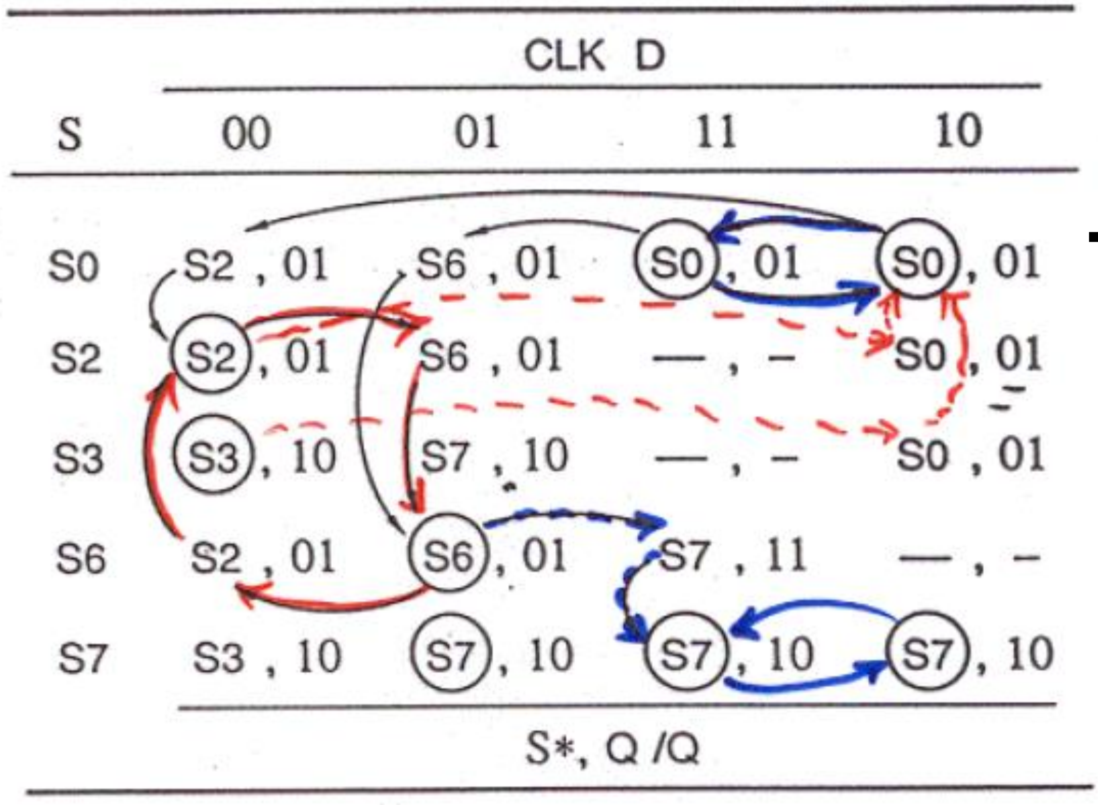
- Identify the stable states, transitions, races, etc.

State		CLK	D	
Y1 Y2 Y3	00	01	11	10
0 0 0 = A	C,01	G,01	A,01	A,01
0 1 0 = C	C,01	G,01	- -	A,01
0 1 1 = D	D,10	H,10	- -	A,01
1 1 0 = G	C,01	G,01	H,11	- -
1 1 1 = H	D,10	H,10	H,10	H,10

D	*clk*	*Q**	*Q*′*
0	↑	**0**	***1***

Design Analysis – Step 4

- Identify the stable states, transitions, races, etc.
- Behavior of a positive edge-triggered D FF

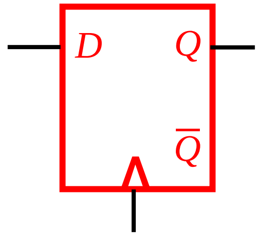


	CLK D			
S	00	01	11	10
S0	S2 , 01	S6 , 01	$\boxed{S0}$, 01	$\boxed{S0}$, 01
S2	$\boxed{S2}$, 01	S6 , 01	— , –	S0 , 01
S3	$\boxed{S3}$, 10	S7 , 10	— , –	S0 , 01
S6	S2 , 01	$\boxed{S6}$, 01	S7 , 11	— , –
S7	S3 , 10	$\boxed{S7}$, 10	$\boxed{S7}$, 10	$\boxed{S7}$, 10
	S*, Q /Q			

$\longrightarrow$

D	*clk*	*Q**	*Q*′*
0	↑	0	*1*
1	↑	1	0
x	0	Q	Q′
x	1	Q	Q′

$\downarrow$


D
Q
$\overline{Q}$