DIGITAL VLSI DESIGN

Rekha S S
Department of
Electronics and
Communication Engg.

Sequential MOS Logic Circuits

Reference Books:

R1: CMOS Digital Integrated Circuits Analysis and Design, Sung-Mo (Steve) Kang,3rd Edition.
Introduction, Behavior of Bi-stable Elements, The SR Latch Circuit (8.1,8.2, 8.3)
Clocked Latch and Flip Flop Circuits (8.4)
CMOS D-Latch and Edge-Triggered Flip-Flop (8.5)

Sequential MOS Logic Circuits

Unit 4: Sequential MOS Logic Circuits				
35 - 36	R1: Chap 8 (8.1 to 8.5) R2: Chap 7 (7.1 and 7.2)	Introduction, Behavior of Bi-stable Elements, The SR Latch Circuit (8.1,8.2, 8.3)	21%	81%
37		Clocked Latch and Flip Flop Circuits (8.4)		
38-39		CMOS D-Latch and Edge-Triggered Flip-Flop (8.5)		
40-41		Sequencing static circuits, Sequencing Methods (7.1)		
42		Max-Delay Constraints (7.2)		
43		Min-Delay Constraints (7.2)		
44		Time Borrowing (7.2)		
45		Clock Skew (7.2), Problems on Max and Min Delay Constraints at design level.		

Reference Books:

R1: CMOS Digital Integrated Circuits Analysis And Design, **Sung-Mo (Steve) Kang**

R2: CMOS VLSI design A Circuits and Systems Perspective, Neil Weste and David Harris, 3rd Edition.

Sequential MOS Logic Circuits

- **Sequential logic circuits contain one or more combinational logic blocks along with memory in a feedback loop with the logic**

Sequential MOS Logic Circuits

- **Sequential logic circuits contain one or more combinational logic blocks along with memory in a feedback loop with the logic**
  - The next state of the machine depends on the present state and the inputs
  - The output depends on the present state of the machine and perhaps also on the inputs

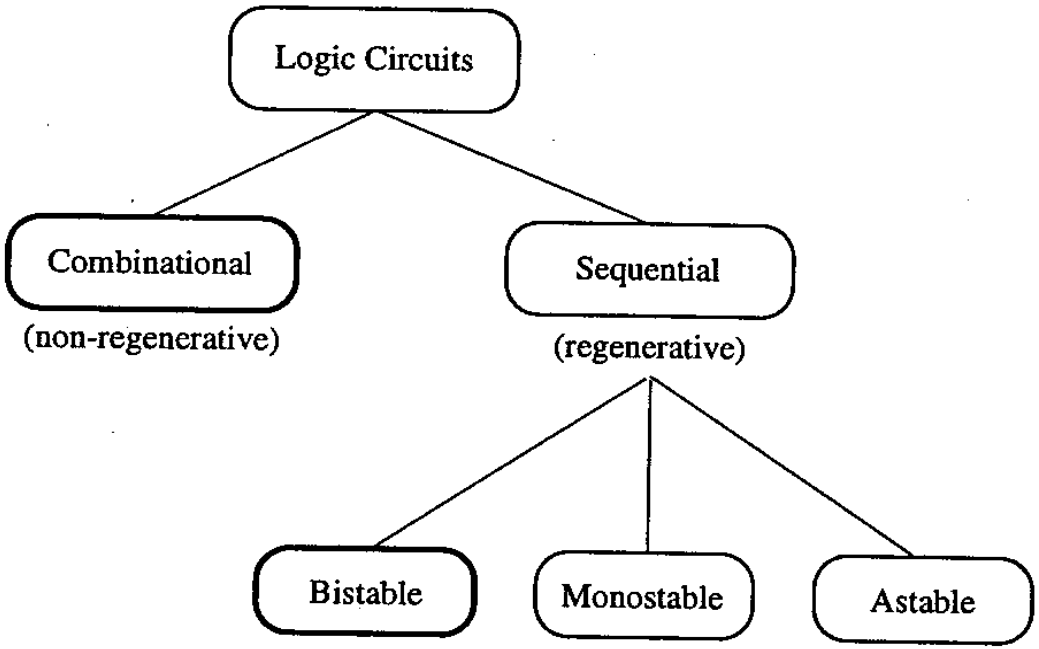
Sequential MOS Logic Circuits

- **Sequential logic circuits contain one or more combinational logic blocks along with memory in a feedback loop with the logic**
  - The next state of the machine depends on the present state and the inputs
  - The output depends on the present state of the machine and perhaps also on the inputs
    - **Mealy machine:** the output depends on both the present state and the inputs
    - **Moore machine**: output depends only on the state of the machine

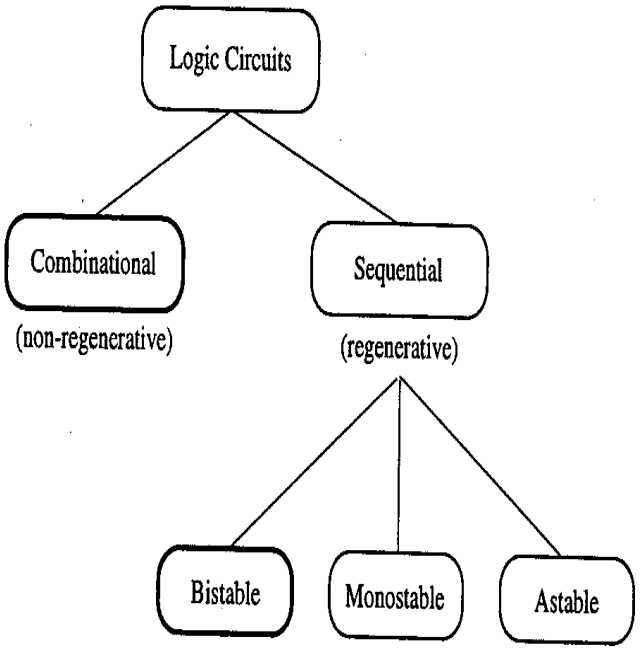
Sequential MOS Logic Circuits

- **Sequential logic circuits contain one or more combinational logic blocks along with memory in a feedback loop with the logic**
  - The next state of the machine depends on the present state and the inputs
  - The output depends on the present state of the machine and perhaps also on the inputs
    - **Mealy machine:** the output depends on both the present state and the inputs

Logic Circuit Classification: Sequential Circuit Types

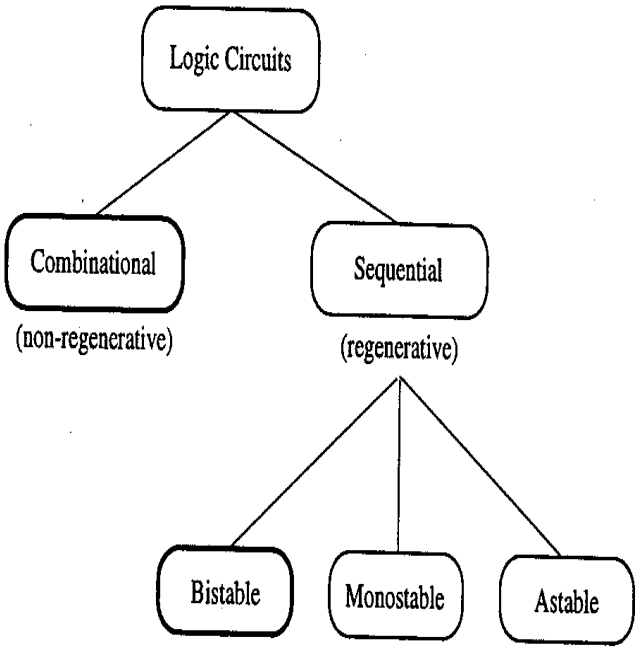


Logic Circuit Classification: Sequential Circuit Types



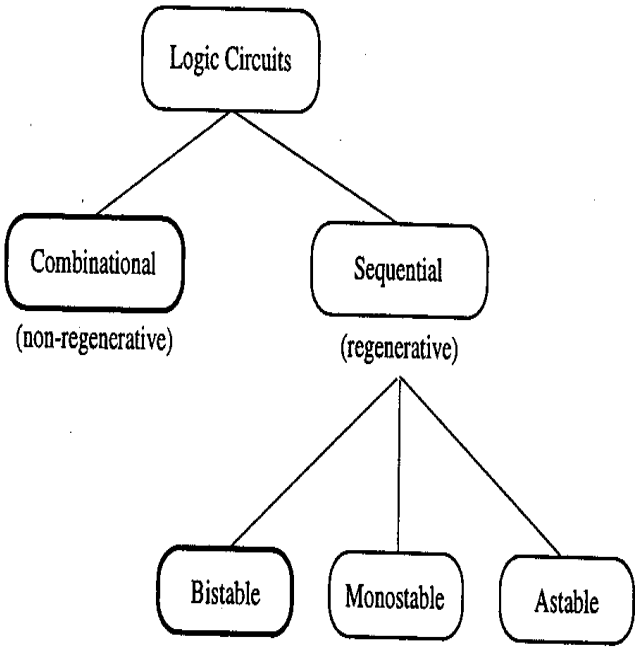
- Sequential circuits (also called regenerative circuits) fall into three types:
  - **Bistable**
  - **Monostable**
  - **Astable**

Logic Circuit Classification: Sequential Circuit Types



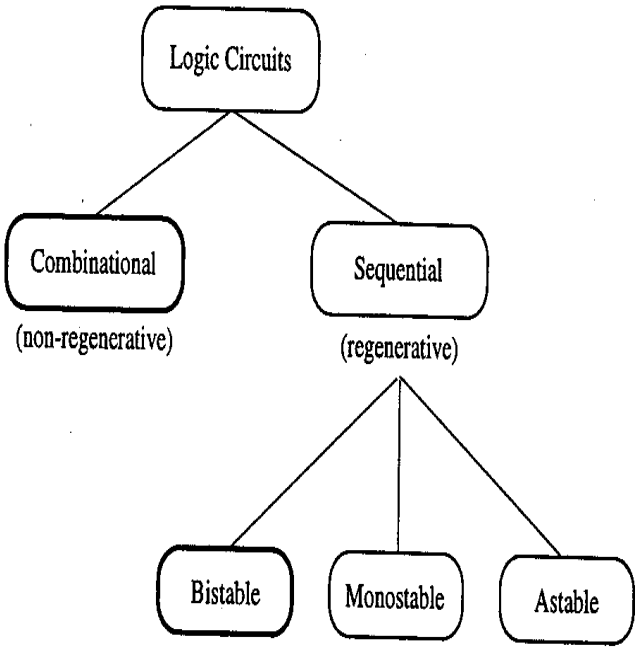
- Sequential circuits (also called regenerative circuits) fall into three types:
  - **Bistable**
  - **Monostable**
  - **Astable**
- Bistable circuits have two stable operating points and will remain in either state unless perturbed to the opposite state
  - Memory cells, latches, flip-flops, and registers

Logic Circuit Classification: Sequential Circuit Types



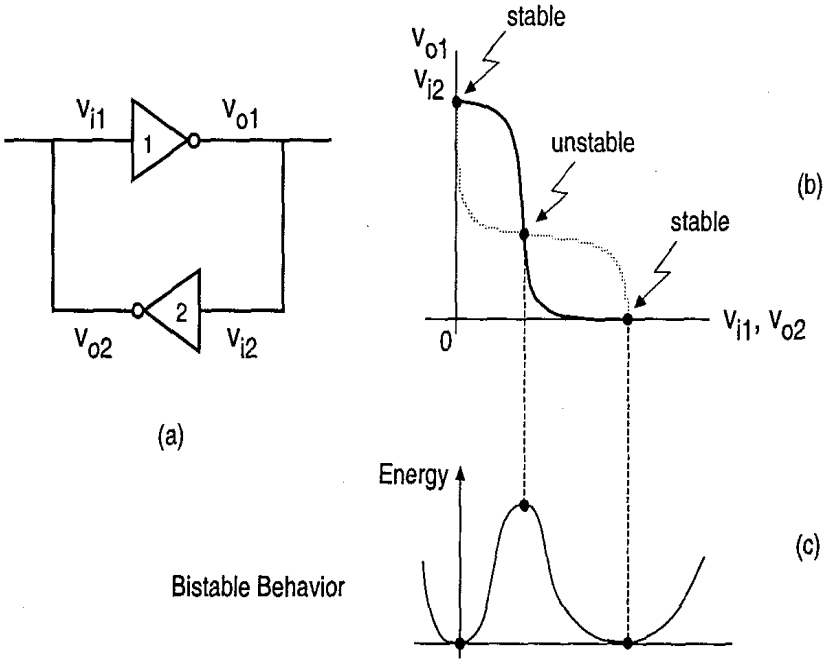
- Sequential circuits (also called regenerative circuits) fall into three types:
  - **Bistable**
  - **Monostable**
  - **Astable**
- Bistable circuits have two stable operating points and will remain in either state unless perturbed to the opposite state
  - Memory cells, latches, flip-flops, and registers
- Monostable circuits have only one stable operating point, and even if they are temporarily perturbed to the opposite state, they will return in time to their stable operating point

Logic Circuit Classification: Sequential Circuit Types



- Sequential circuits (also called regenerative circuits) fall into three types:
  - **Bistable**
  - **Monostable**
  - **Astable**
- Bistable circuits have two stable operating points and will remain in either state unless perturbed to the opposite state
  - Memory cells, latches, flip-flops, and registers
- Monostable circuits have only one stable operating point, and even if they are temporarily perturbed to the opposite state, they will return in time to their stable operating point
- Astable circuits have no stable operating point and oscillate between several states
  - Ring oscillator

Behavior of Bistable Elements



A memory cell is comprised of two inverters connected back-to-back, i.e. output of one to input of the other and vice-versa.

The VTC of inverter(1) with to the v01 – vi1 axis pair.

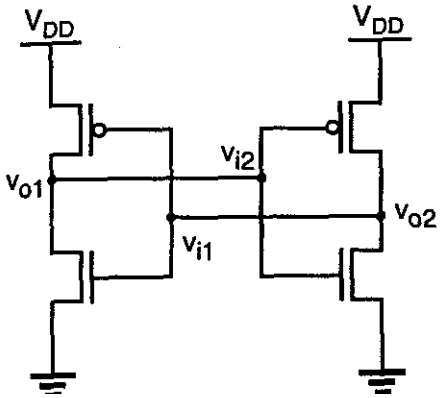
The i/p and o/p voltages of inv (2) corresponds to the o/p and i/p voltages of inv(1) using same axis pair.

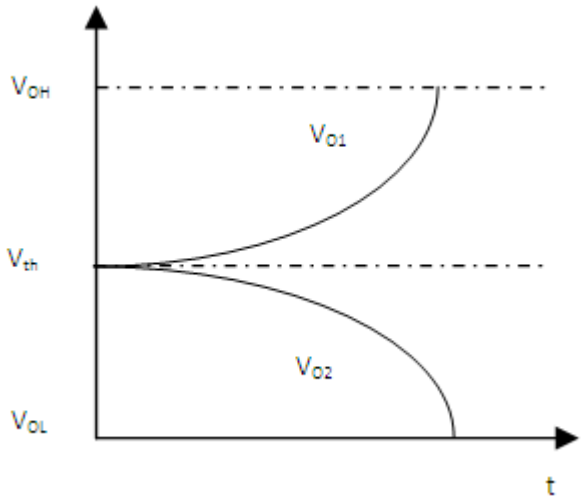
VTC intersect at 3 points.

If the circuit is initially operating at one of these two stable points, it will preserve this state unless it is forced externally to change its operating point

- In actual physical circuits the memory cell will never stay at the unstable point, since any small electrical noise in the circuit will trigger it to one side or the other

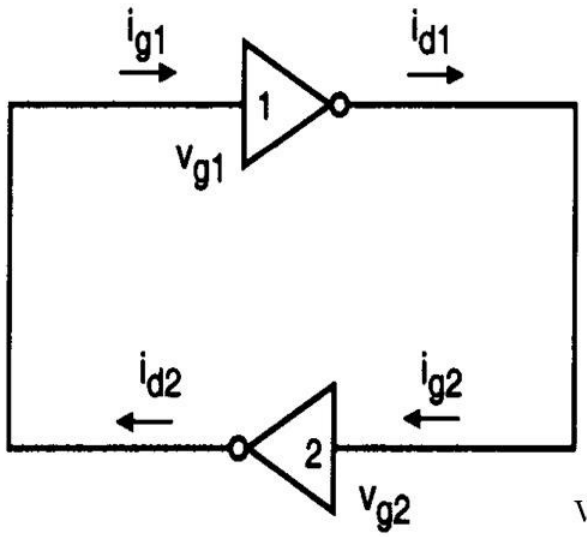
Behavior of Bistable Elements





- The CMOS SRAM cell at the left will either be in state "0" with V01 at GND and V02 at V_{DD} or in state "1" with V01 at V_{DD} and V02 at GND.
- At unstable operating point of this circuit, all four transistors are in saturation
- If initial operating condition is set at this point, any small voltage perturbation will cause significant changes in the operating modes of the transistors
- Thus outputs of the two inverters to diverge and eventually settle at V_{OH} and V_{OL}

Behavior of Bistable Elements



At $V_{01} = V_{02} = V_{th}$ ⟶ Unstable condition

$$i_{g1} = i_{d2} = g_m v_{g2}$$
$$i_{g2} = i_{d1} = g_m v_{g1}$$

Eqn. 1

where $g_m = (g_{mN} + g_{mP})$: *All four Transistor are in saturation* . The drain current of each inverter is also equal to the gate current of the other inverter.

Gate Capacitance >>> Drain Capacitance

Gate Charges

$$v_{g1} = \frac{q_1}{C_g} \qquad v_{g2} = \frac{q_2}{C_g}$$

Eqn. 2

Behavior of Bistable Elements

Derivative of small-signal gate voltages

$$i_{g1} = C_g \frac{dv_{g1}}{dt}$$

$$i_{g2} = C_g \frac{dv_{g2}}{dt}$$

Eqn. 3

Combining Eqn. 1 and Eqn.3

$$g_m v_{g2} = C_g \frac{dv_{g1}}{dt}$$

$$g_m v_{g1} = C_g \frac{dv_{g2}}{dt}$$

Eqn. 4

Behavior of Bistable Elements

$$\frac{dq_1}{dt} = ig_1$$

$$i_{g1} = g_m v_{g2}$$

$$v_{g2} = \frac{q_2}{C_g}$$

$$\frac{g_m}{C_g} q_2 = \frac{dq_1}{dt}$$

$$\frac{g_m}{C_g} q_1 = \frac{dq_2}{dt}$$

Combination above three equations

Eqn. 5

Behavior of Bistable Elements

$$\frac{g_m}{C_g} q_2 = \frac{dq_1}{dt} \quad -5a \qquad \frac{g_m}{C_g} q_1 = \frac{dq_2}{dt} \quad -5b$$

(5a) can be rewritten as,

$$q_2 = \frac{C_g}{g_m} \frac{dq_1}{dt} \quad -(6)$$

Substitute (6) in (5b)

Combination of two equation yields 2nd order differential equation

$$\frac{g_m}{C_g} q_1 = \frac{C_g}{g_m} \frac{d^2 q_1}{dt^2} \Rightarrow \text{rearranged to} \Rightarrow \frac{d^2 q_1}{dt^2} = \left(\frac{g_m}{C_g} \right)^2 q_1$$

Time behaviour of gate charge q1

Behavior of Bistable Elements

$$\frac{d^2 q_1}{dt^2} = \frac{1}{\tau_0^{\,2}} q_1 \quad with \ \tau_0 = \frac{C_g}{g_m}$$

τ_0 ↓ transit time Constant

$$\frac{d^2 q_1}{dt^2} - \frac{q_1}{\tau_0^2} = 0$$

$$\frac{d^2 q_1}{dt^2} - \frac{q_1}{a} = 0 \qquad \left(a = \tau_0^2\right)$$

Solution:-

$$q_1 = e^{rt}$$

$$\frac{dq_1}{dt} = re^{rt} \quad ; \frac{d^2 q_1}{dt^2} = r^2 e^{rt}$$

Behavior of Bistable Elements

$$s^2 e^{st} - \frac{e^{st}}{a} = 0$$

$$e^{st}\left(s^2 - \frac{1}{a}\right) = 0$$

$$s^2 = \frac{1}{a}$$

$$s = \pm \frac{1}{\sqrt{a}} \qquad \text{where } \sqrt{a} = \tau_0$$

Now $q_1(t) = e^{st}$

$$q_1(t) = A e^{+\frac{1}{\tau_0} t} + B e^{-\frac{1}{\tau_0} t}$$

At $t = 0$

$$q_1(0) = A + B$$

Behavior of Bistable Elements

$$q_1(t) = A e^{+\frac{1}{\tau_0}t} + B e^{-\frac{1}{\tau_0}t}$$

$$q_1'(t) = A \frac{1}{\tau_0} e^{+\frac{1}{\tau_0}t} - B \frac{1}{\tau_0} e^{-\frac{1}{\tau_0}t}$$

$$q_1'(t)\,\tau_0 = A e^{+\frac{1}{\tau_0}t} - B e^{-\frac{1}{\tau_0}t}$$

$$q_1'(0)\,\tau_0 = A - B$$

$$A = \frac{q_1(0) + \tau_0\, q_1'(0)}{2}$$

$$B = \frac{q_1(0) - \tau_0\, q_1'(0)}{2}$$

Behavior of Bistable Elements

$$q_1(t) = A e^{+\frac{1}{\tau_0}t} + B e^{-\frac{1}{\tau_0}t}$$

$$q_1(t) = \frac{q_1(0) + \tau_0 q_1'(0)}{2} e^{+\frac{1}{\tau_0}t} + \frac{q_1(0) - \tau_0 q_1'(0)}{2} e^{-\frac{1}{\tau_0}t}$$

Time-domain expression for q_1 yields:

$$q_1(t) = \underbrace{\frac{q_1(0) - \tau_0\, q_1'(0)}{2}}_{B}\, e^{-\frac{t}{\tau_0}} + \underbrace{\frac{q_1(0) + \tau_0\, q_1'(0)}{2}}_{A}\, e^{+\frac{t}{\tau_0}}$$

initial condition is given as:

$$q_1(0) = C_g \cdot v_{g1}(0)$$

Behavior of Bistable Elements

Note that $v_{g1} = v_{o2}$ and $v_{g2} = v_{o1}$. Replacing the gate charge of both inverters with the corresponding output-voltage variables, we obtain,

$$v_{o2}(t) = \frac{1}{2}\left(v_{o2}(0) - \tau_0\, v_{o2}'(0)\right)e^{-\frac{t}{\tau_0}} + \frac{1}{2}\left(v_{o2}(0) + \tau_0\, v_{o2}'(0)\right)e^{+\frac{t}{\tau_0}}$$

$$v_{o1}(t) = \frac{1}{2}\left(v_{o1}(0) - \tau_0\, v_{o1}'(0)\right)e^{-\frac{t}{\tau_0}} + \frac{1}{2}\left(v_{o1}(0) + \tau_0\, v_{o1}'(0)\right)e^{+\frac{t}{\tau_0}}$$

Assume ~ Large t

$$v_{o1}(t) \approx \frac{1}{2}\left(v_{o1}(0) + \tau_0\, v_{o1}'(0)\right)e^{+\frac{t}{\tau_0}}$$

$$v_{o2}(t) \approx \frac{1}{2}\left(v_{o2}(0) + \tau_0\, v_{o2}'(0)\right)e^{+\frac{t}{\tau_0}}$$

The magnitude of both output voltages increases exponentially with time. Depending on the polarity of the initial small perturbations dvo0 (O) and dvo2(O), the output voltages of both inverters will diverge from their initial value of V_{th} to either VOL or VOH.

v_{o1}: $V_{th} \rightarrow V_{OH}$ *or* V_{OL}

v_{o2}: $V_{th} \rightarrow V_{OL}$ *or* V_{OH}

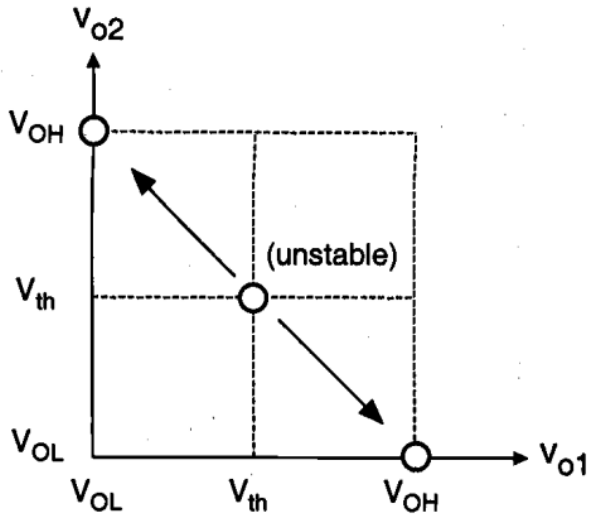
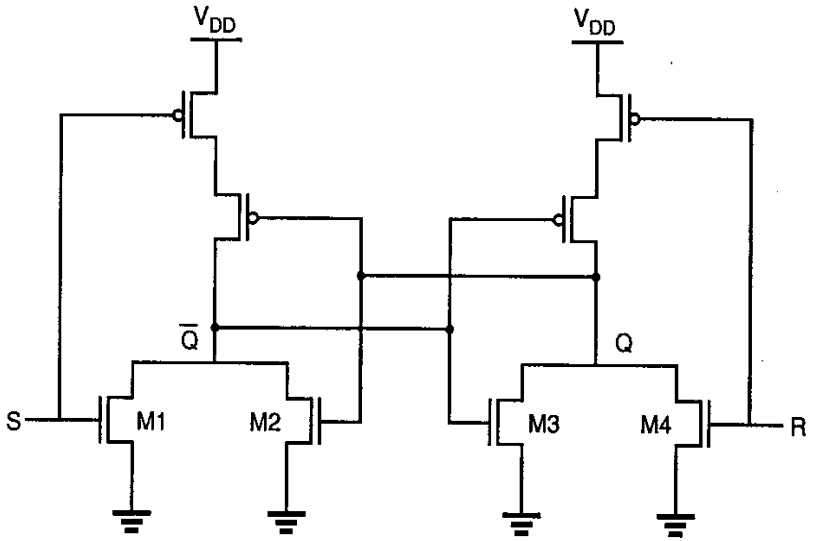
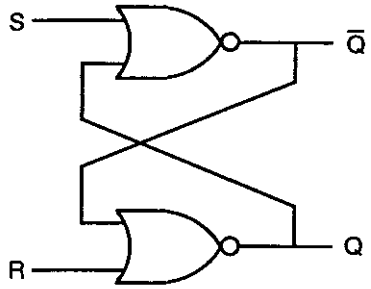


Figure 8.5. Phase-plane representation of the bistable circuit behavior.

CMOS SR Latch: NOR Gate Version



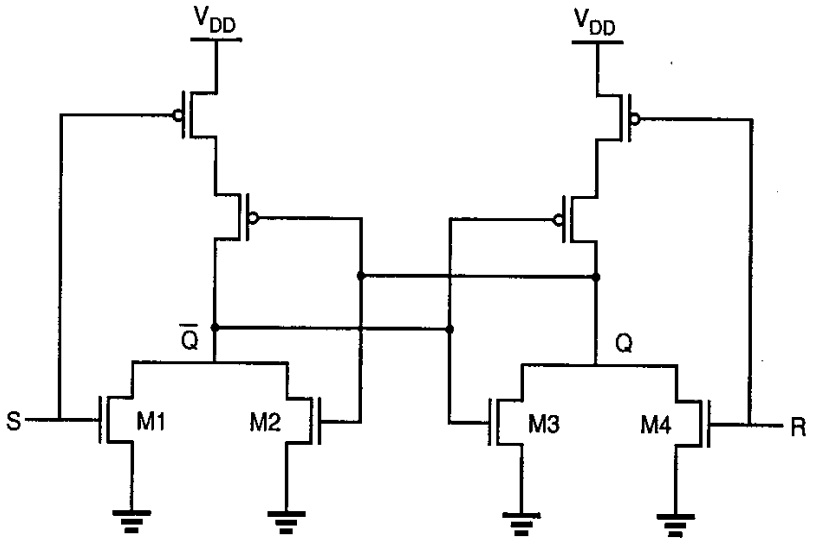
- Can perform a simple memory function of holding its state.
- Simple CMOS SR latch, has two triggering inputs S and R



S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	Q_n	$\overline{Q_n}$	hold
1	0	1	0	set
0	1	0	1	reset
1	1	0	0	not allowed

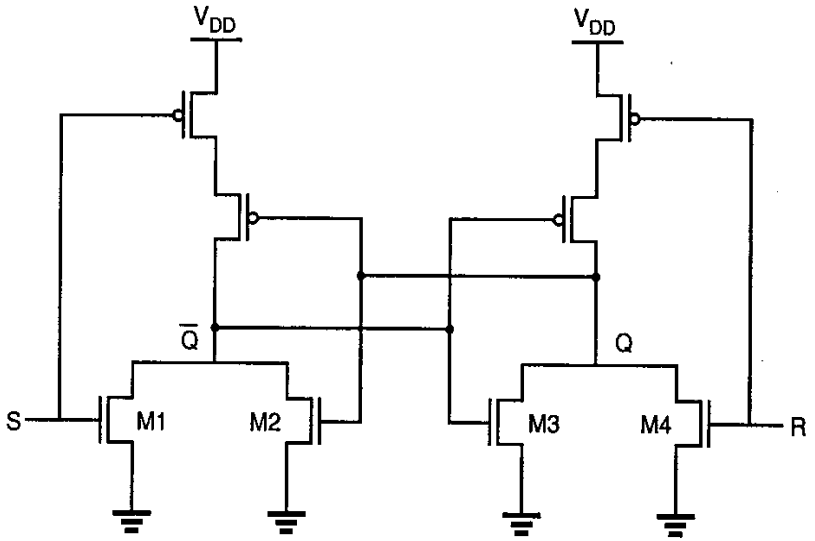
The gate-level symbol and truth table for the NOR-based SR latch

CMOS SR Latch: NOR Gate Version



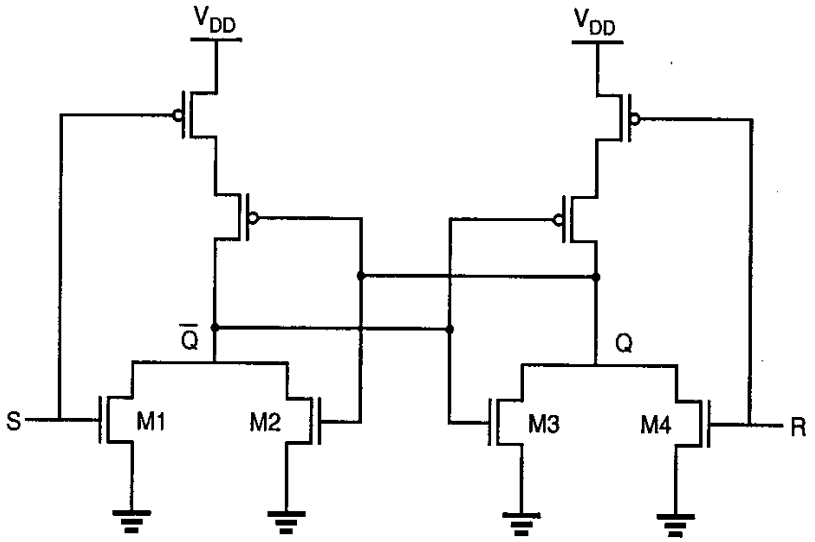
- If Set goes high, M1 is turned on forcing Q' low which, in turn, pulls Q high
  - S=1 ➔ Q = 1

CMOS SR Latch: NOR Gate Version



- If Set goes high, M1 is turned on forcing Q' low which, in turn, pulls Q high
  - S=1 ➔ Q = 1
- If Reset goes high, M4 is turned on, Q is pulled low, and Q' is then pulled high
  - R=1 ➔ Q' = 1

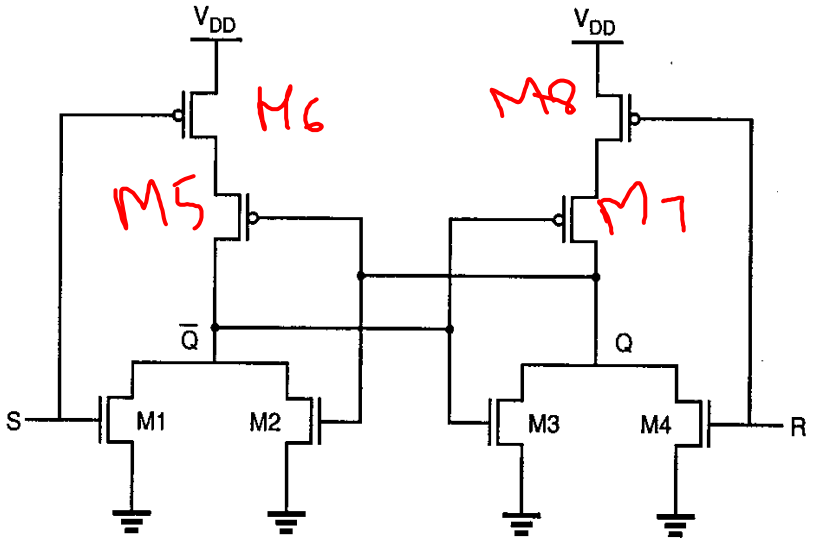
CMOS SR Latch: NOR Gate Version



- If Set goes high, M1 is turned on forcing Q' low which, in turn, pulls Q high
  - S=1 ➔ Q = 1
- If Reset goes high, M4 is turned on, Q is pulled low, and Q' is then pulled high
  - R=1 ➔ Q' = 1

- If both Set and Reset are low, both M1 and M4 are off, and the latch holds its existing state indefinitely

CMOS SR Latch: NOR Gate Version



If Set goes high, M1 is turned on forcing Q' low which, in turn, pulls Q high

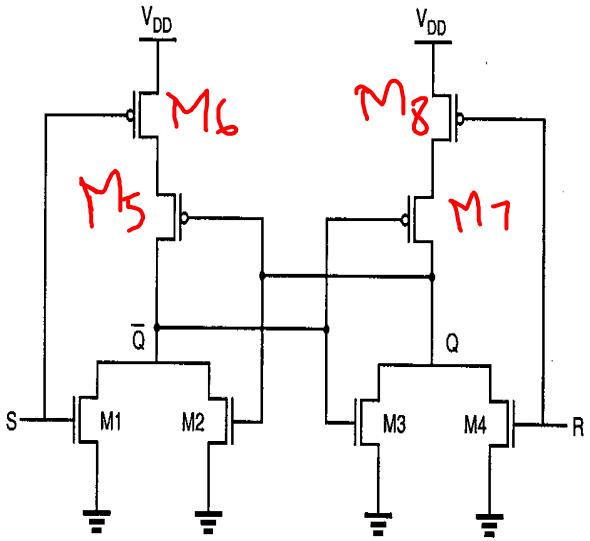
- S=1 ➔ Q = 1

If Reset goes high, M4 is turned on, Q is pulled low, and Q' is then pulled high

- R=1 ➔ Q' = 1

- If both Set and Reset are low, both M1 and M4 are off, and the latch holds its existing state indefinitely

- If both Set and Reset go high, both Q and Q' are pulled low, giving an indefinite state. Therefore, R=S=1 is not allowed



S	R	Q_{N+1}	bQ_{n+1}	operations
V_{OH}	V_{OL}	V_{OH}	V_{OL}	M1 and M2 on, M3 and M4 off
V_{OL}	V_{OH}	V_{OL}	V_{OH}	M1 and M2 off, M3 and M4 on
V_{OL}	V_{OL}	V_{OH}	V_{OL}	M1 and M4 off, M2 on or
V_{OL}	V_{OL}	V_{OL}	V_{OH}	M1 and M4 off, M3 on

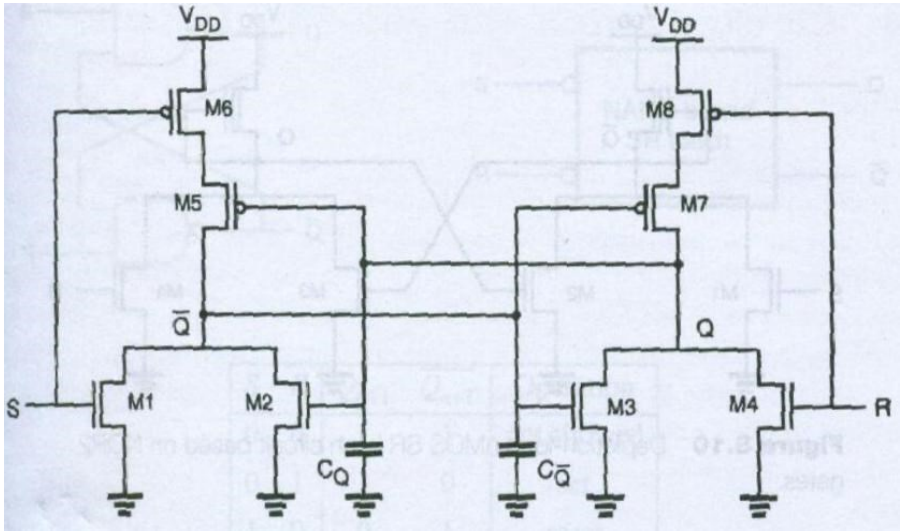
Calculating of Rise time of SR Latch

- Thus, an interesting problem is to estimate the amount of time required for the simultaneous switching of the two output nodes.
- Solution – requires the simultaneous solution of two coupled differential equations, one each for each output node.
- Assume that two events take place in sequence rather than simultaneously.--- causes an overestimation of the switching time
- To calculate the switching times for the both output nodes
- find the total parasitic capacitance

$$C_Q = C_{gb,2} + C_{gb,5} + C_{db,3} + C_{db,4} + C_{db,7} + C_{sb,7} + C_{db,8}$$

$$C_{\overline{Q}} = C_{gb,3} + C_{gb,7} + C_{db,1} + C_{db,2} + C_{db,5} + C_{sb,5} + C_{db,6}$$

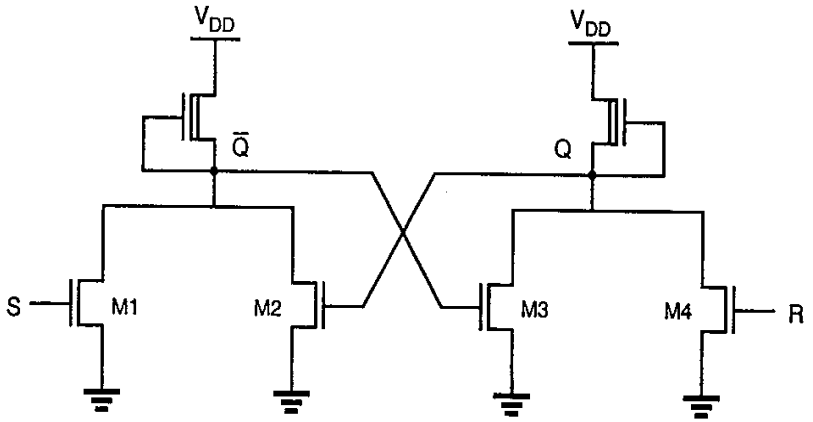
Calculating of Rise time of SR Latch



$$\tau_{rise,Q}(SR-latch) = \tau_{rise,Q}(\text{NOR2}) + \tau_{fall,\bar{Q}}(\text{NOR2})$$

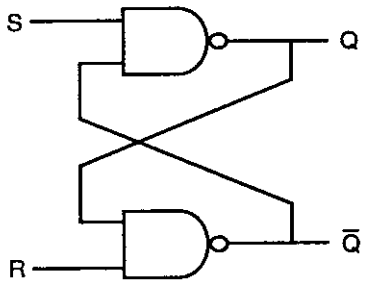
Depletion Load NMOS SR Latch: NOR Version

- From the logic point of view, the operation principle of the depletion-load nMOS NOR-based SR latch is identical to that of the CMOS SR latch.
- The latch is a ratio circuit
  - Low side conducts dc current, causing higher standby power than CMOS version

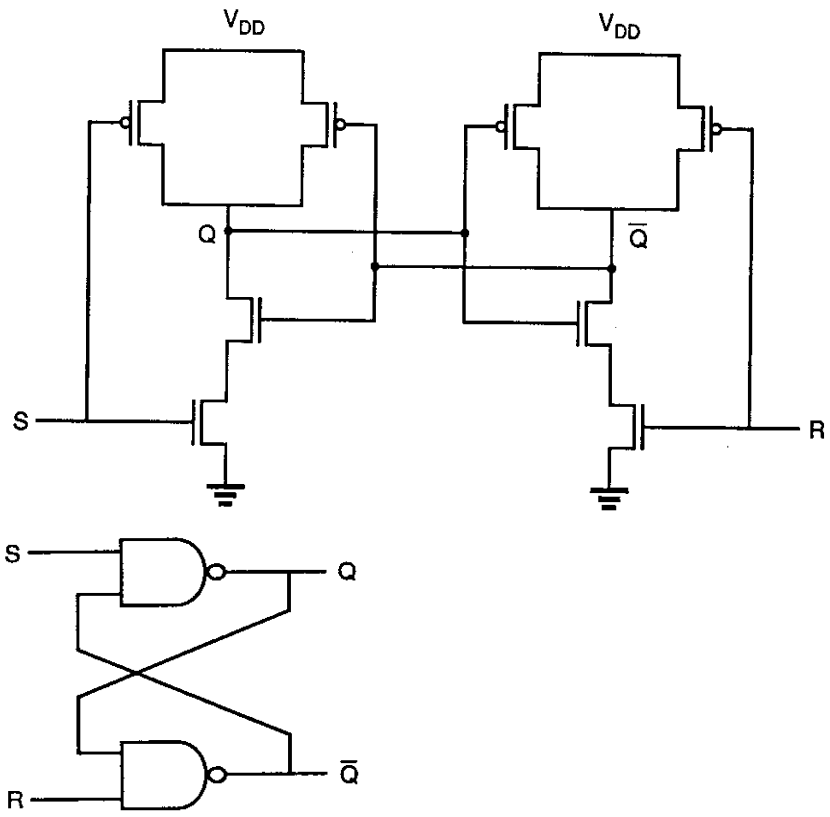


CMOS SR Latch: NAND Gate Version

- A CMOS SR latch built with two 2-input NAND gates is shown at left
  - The basic memory cell comprised of two back-to-back CMOS inverters is seen



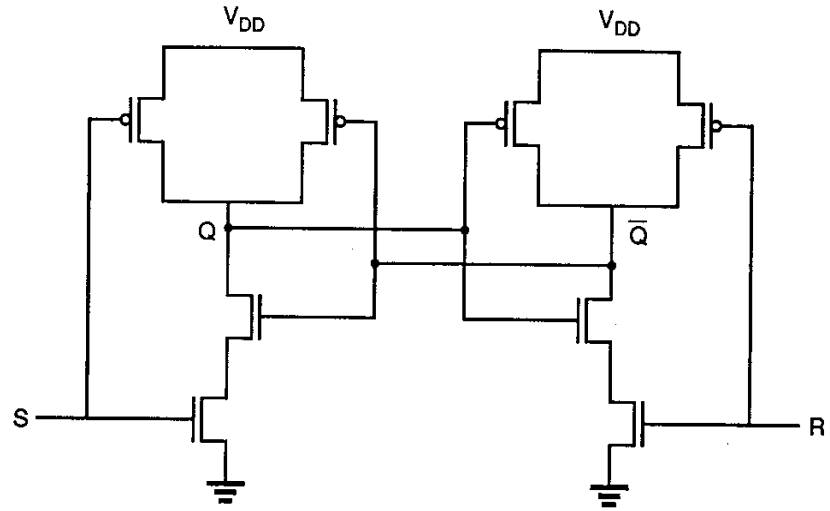
CMOS SR Latch: NAND Gate Version

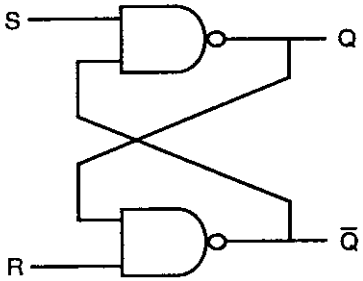


A CMOS SR latch built with two 2-input NAND gates is shown at left

- The basic memory cell comprised of two back-to-back CMOS inverters is seen

CMOS SR Latch: NAND Gate Version





S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	1	1	not allowed
0	1	1	0	set
1	0	0	1	reset
1	1	Q_n	$\overline{Q_n}$	hold

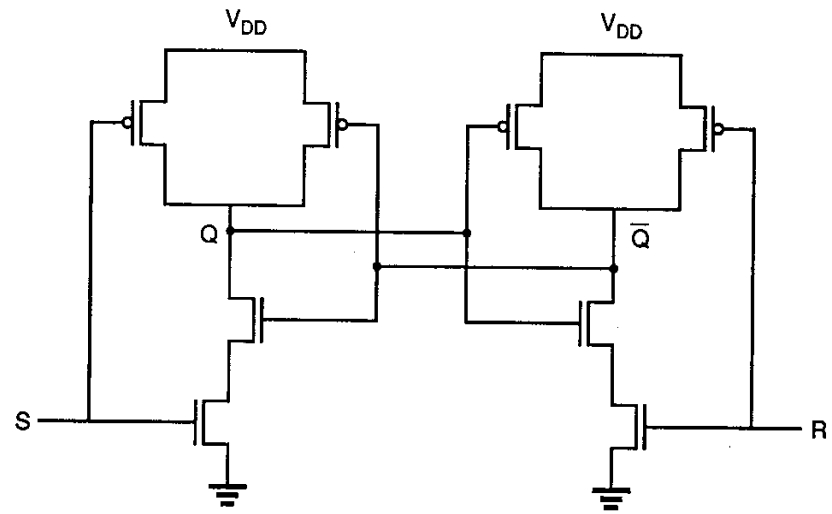
A CMOS SR latch built with two 2-input NAND gates is shown at left

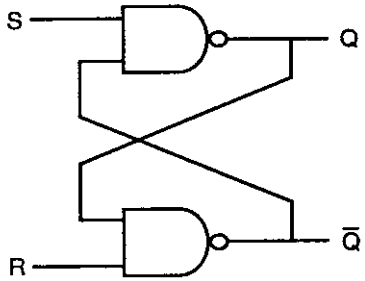
- The basic memory cell comprised of two back-to-back CMOS inverters is seen

The circuit responds to active low S and R inputs

- If S goes to 0 (while R = 1), Q goes high, pulling Q' low and the latch enters Set state
  - S=0 ➔ Q = 1 (if R = 1)

CMOS SR Latch: NAND Gate Version





S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	1	1	not allowed
0	1	1	0	set
1	0	0	1	reset
1	1	Q_n	$\overline{Q_n}$	hold

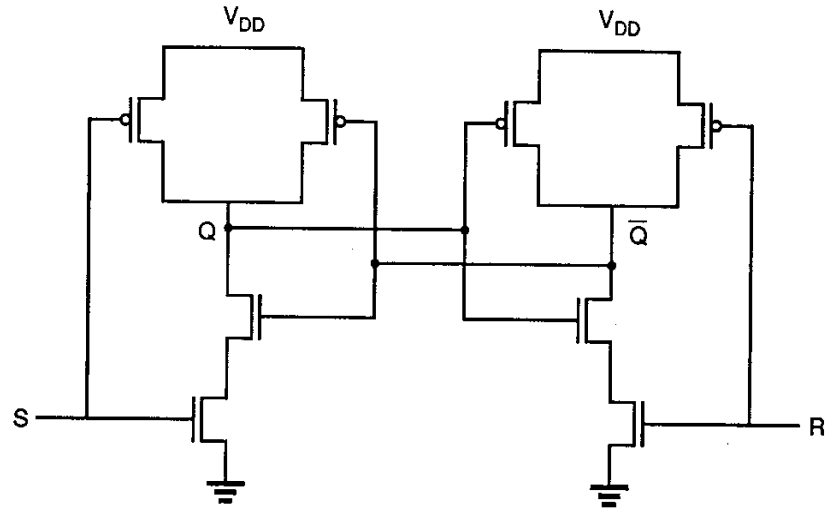
A CMOS SR latch built with two 2-input NAND gates is shown at left

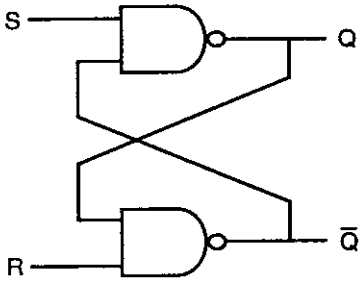
- The basic memory cell comprised of two back-to-back CMOS inverters is seen

The circuit responds to active low S and R inputs

- If S goes to 0 (while R = 1), Q goes high, pulling Q' low and the latch enters Set state
  - S=0 ➔ Q = 1 (if R = 1)
- If R goes to 0 (while S = 1), Q' goes high, pulling Q low and the latch is Reset
  - R=0 ➔ Q' = 1 (if S = 1)

CMOS SR Latch: NAND Gate Version





S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	1	1	not allowed
0	1	1	0	set
1	0	0	1	reset
1	1	Q_n	$\overline{Q_n}$	hold

A CMOS SR latch built with two 2-input NAND gates is shown at left

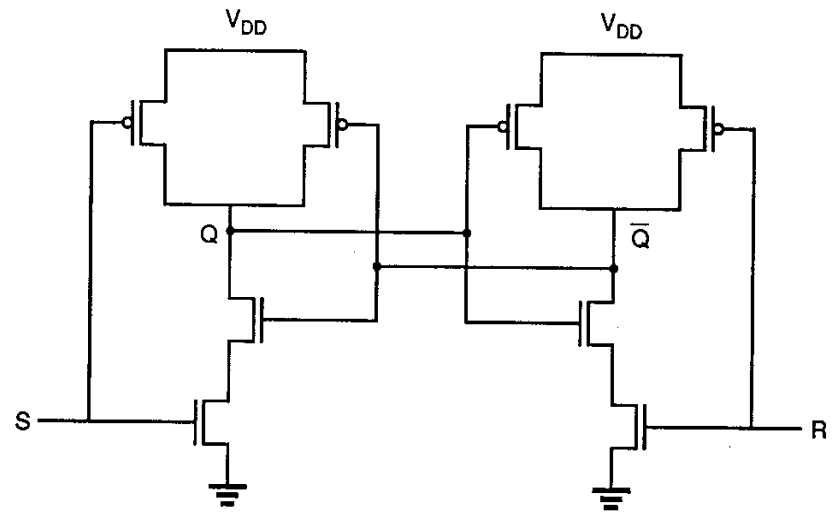
- The basic memory cell comprised of two back-to-back CMOS inverters is seen

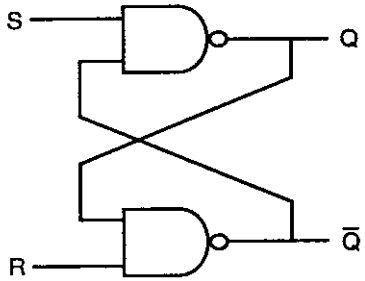
The circuit responds to active low S and R inputs

- If S goes to 0 (while R = 1), Q goes high, pulling Q' low and the latch enters Set state
  - S=0 ➔ Q = 1 (if R = 1)
- If R goes to 0 (while S = 1), Q' goes high, pulling Q low and the latch is Reset
  - R=0 ➔ Q' = 1 (if S = 1)

- Hold state requires both S and R to be high

CMOS SR Latch: NAND Gate Version





S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	1	1	not allowed
0	1	1	0	set
1	0	0	1	reset
1	1	Q_n	$\overline{Q_n}$	hold

A CMOS SR latch built with two 2-input NAND gates is shown at left

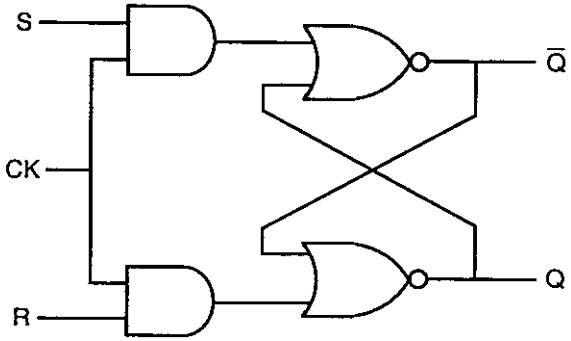
- The basic memory cell comprised of two back-to-back CMOS inverters is seen

The circuit responds to active low S and R inputs

- If S goes to 0 (while R = 1), Q goes high, pulling Q' low and the latch enters Set state
  - S=0 ➔ Q = 1 (if R = 1)
- If R goes to 0 (while S = 1), Q' goes high, pulling Q low and the latch is Reset
  - R=0 ➔ Q' = 1 (if S = 1)

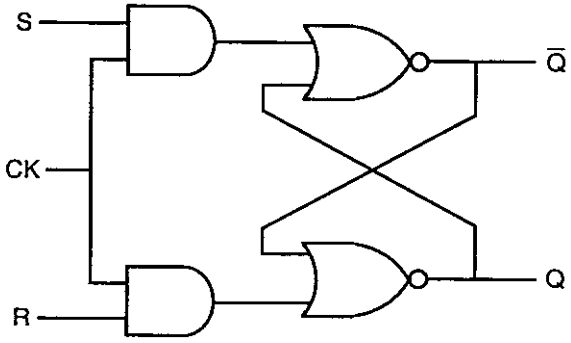
- Hold state requires both S and R to be high
- S = R = 0 if not allowed, as it would result in an indeterminate state

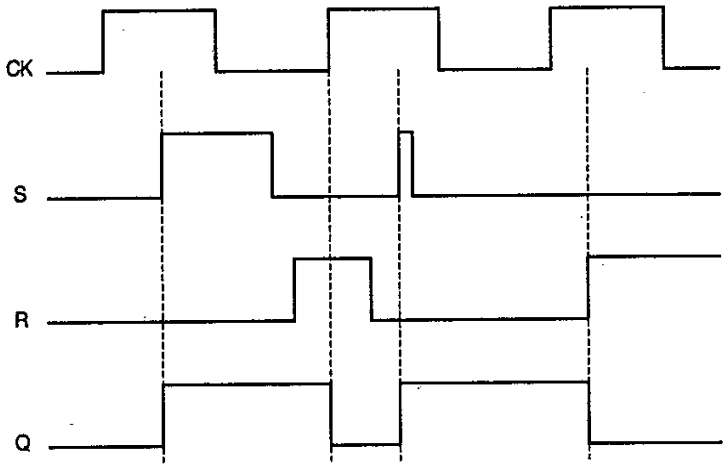
Clocked SR Latch: NOR Version



- Shown at left is the NOR-based SR latch with a clock added.
  - The latch is responsive to inputs S and R only when CLK is high
  - When CLK is low, the latch retains its current state

Clocked SR Latch: NOR Version



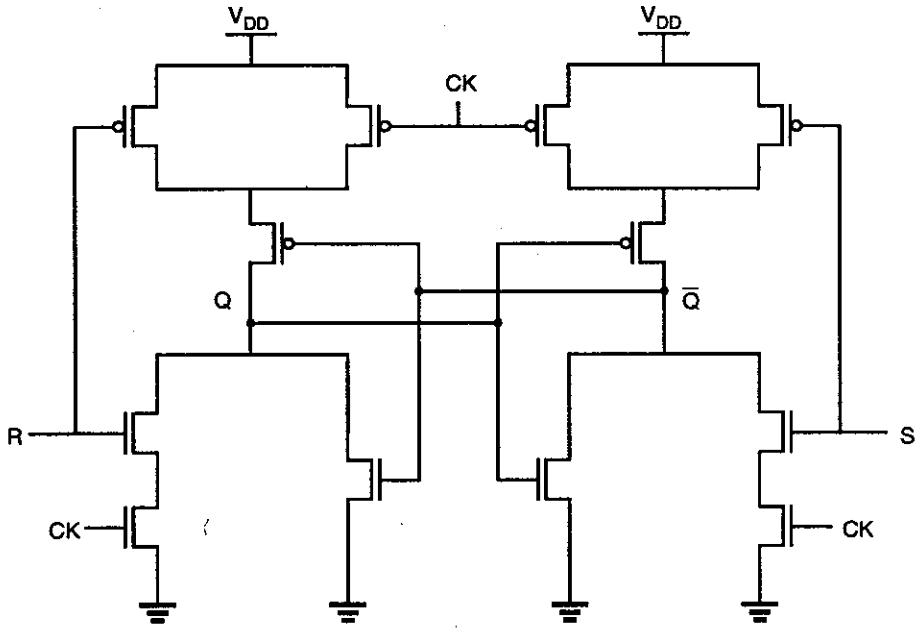


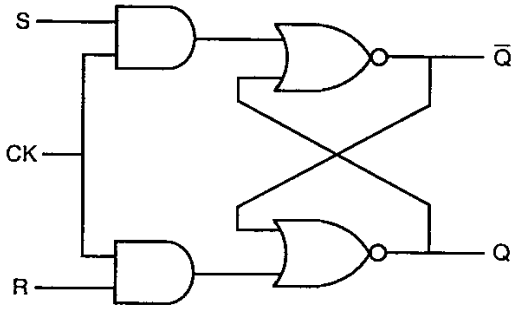
- Shown at left is the NOR-based SR latch with a clock added.
  - The latch is responsive to inputs S and R only when CLK is high
  - When CLK is low, the latch retains its current state
- Timing diagram shows the level-sensitive nature of the clocked SR latch.
  - Note four times where Q changes state:
    - When S goes high during positive CLK
    - On leading CLK edge after changes in S & R during CLK low time
    - A positive glitch in S while CLK is high
    - When R goes high during positive CLK

Clocked SR Latch: NOR Version

- When S=R=1,
  - clk =1 both outputs go momentarily to zero.
  - When clk =0 the state of the latch is indeterminate

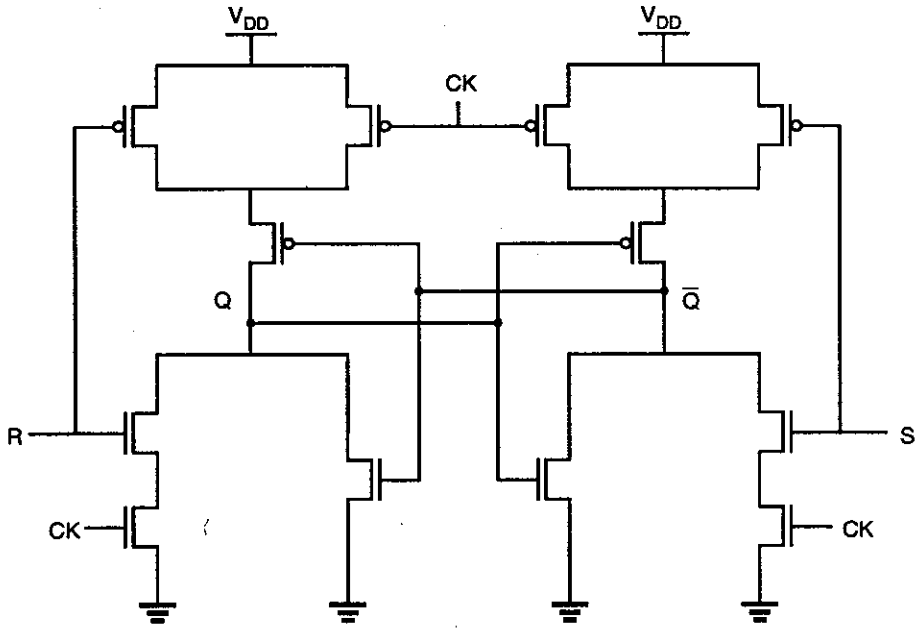
Clocked SR Latch: NOR Version

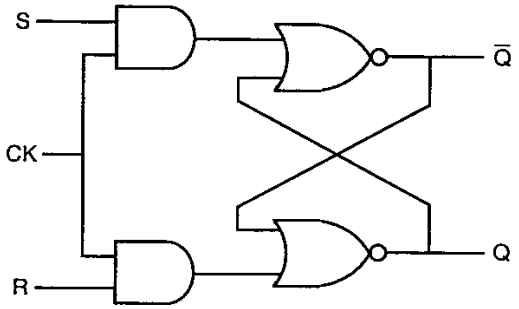




CMOS AOI implementation of clocked NOR-based SR latch shown at left with logic symbol circuit below

Clocked SR Latch: NOR Version

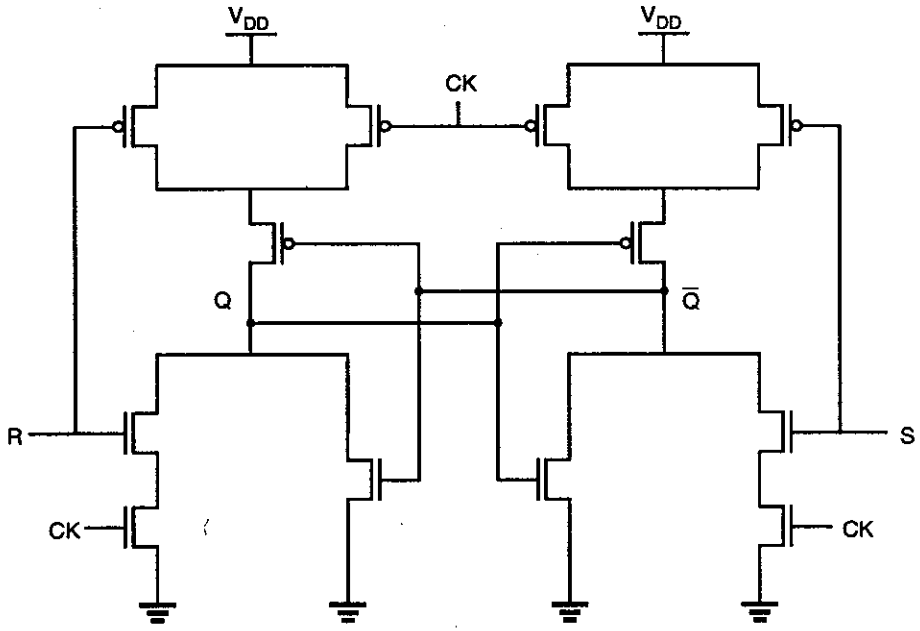


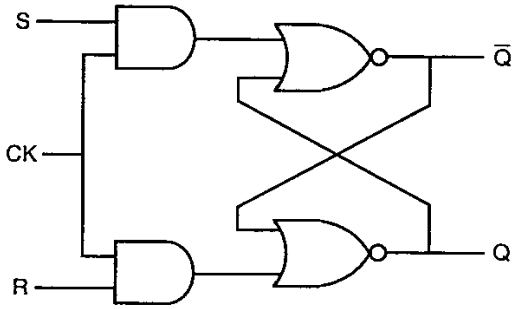


CMOS AOI implementation of clocked NOR-based SR latch shown at left with logic symbol circuit below

- Only 12 transistors required

Clocked SR Latch: NOR Version

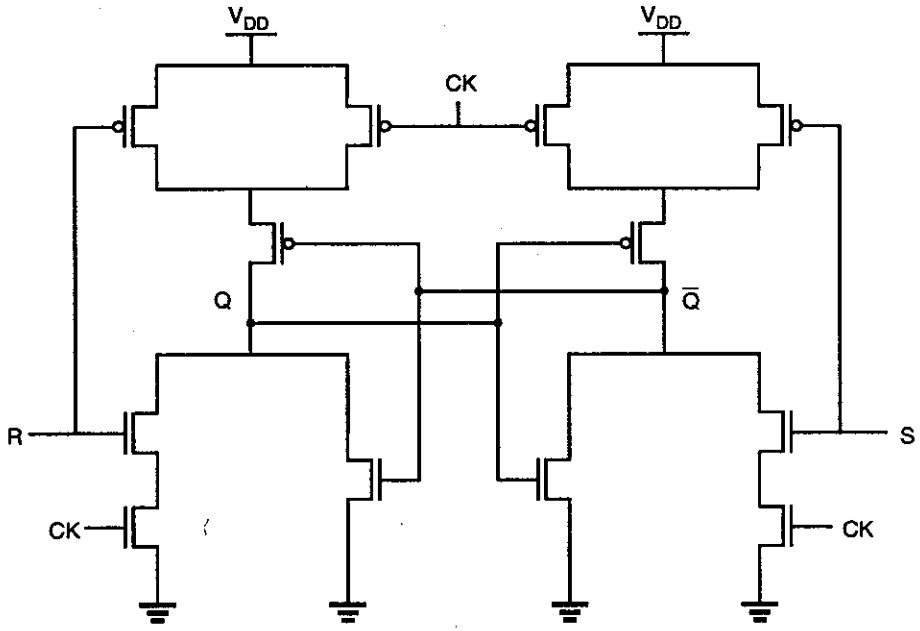


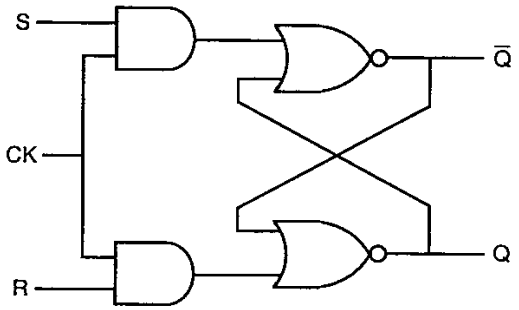


CMOS AOI implementation of clocked NOR-based SR latch shown at left with logic symbol circuit below

- Only 12 transistors required
- When CLK is low, two series legs in N tree are open and two parallel transistors in P tree are ON, thus retaining state in the memory cell

Clocked SR Latch: NOR Version

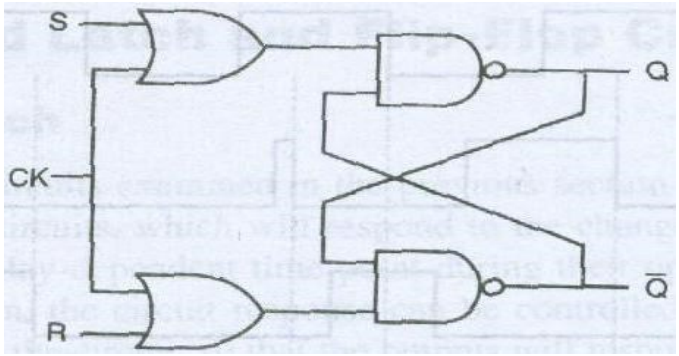




CMOS AOI implementation of clocked NOR-based SR latch shown at left with logic symbol circuit below

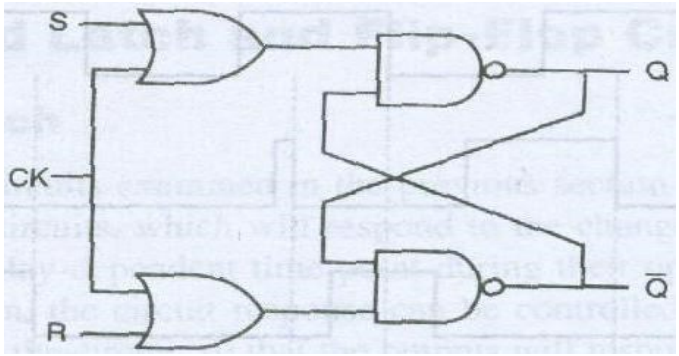
- Only 12 transistors required
- When CLK is low, two series legs in N tree are open and two parallel transistors in P tree are ON, thus retaining state in the memory cell
- When CLK is high, the circuit becomes simply a NOR-based CMOS latch which will respond to inputs S and R

Clocked SR Latch

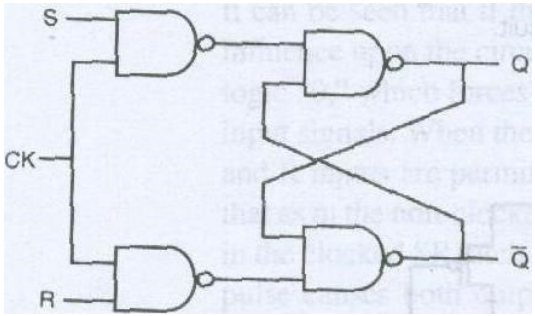


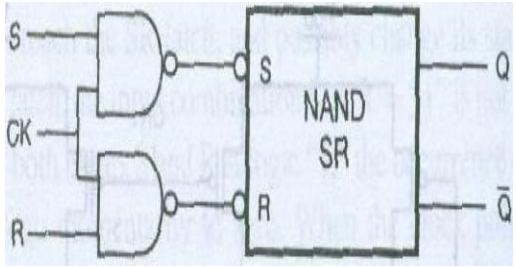
Gate- level schematic of the clocked NAND-based SR latch circuit, with active low inputs

Clocked SR Latch



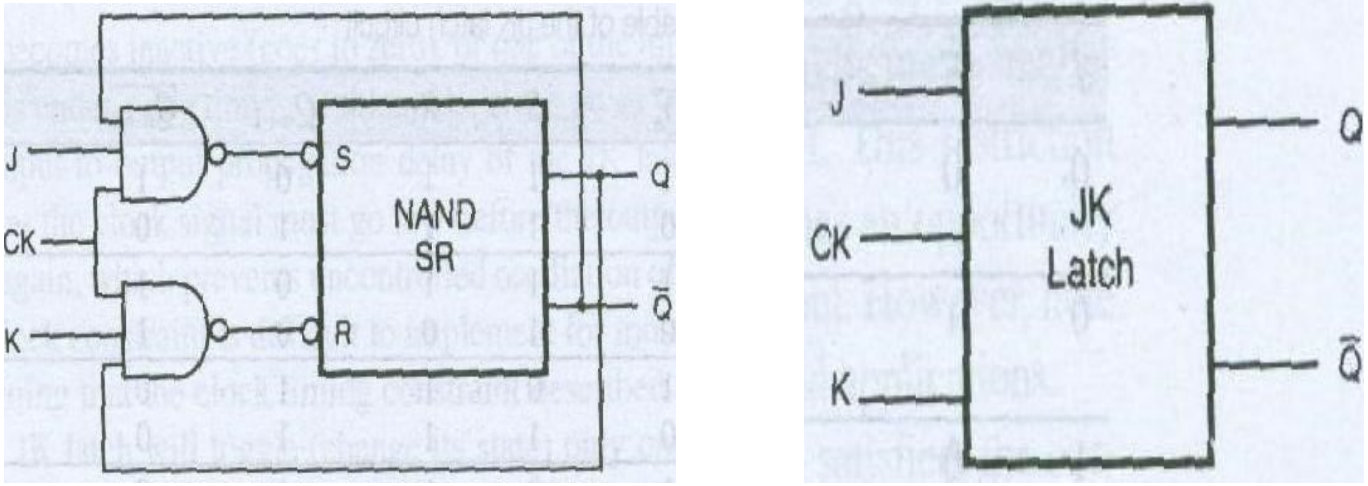
Gate- level schematic of the clocked NAND-based SR latch circuit, with active low inputs





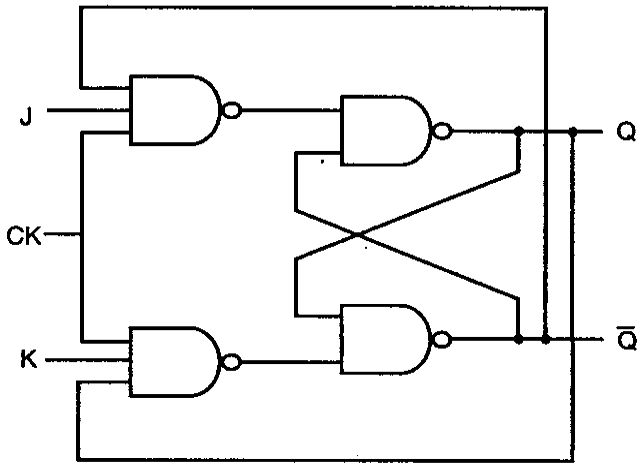
Gate- level schematic of the clocked NAND-based SR latch circuit, with active high inputs and block diagram representation of the same

Clocked JK Latch



Gate- level schematic of the clocked NAND-based JK latch circuit and block diagram representation

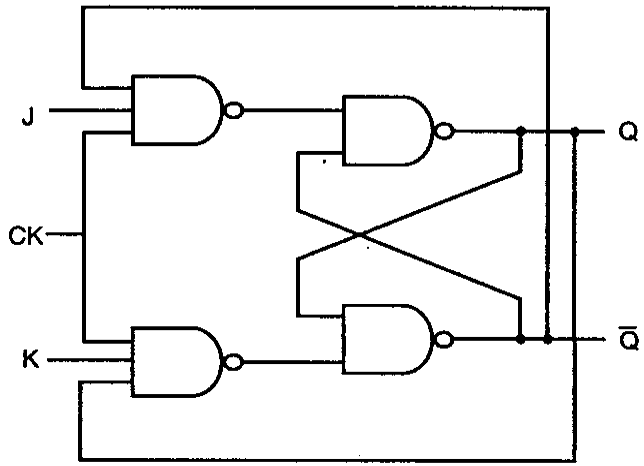
Clocked CMOS JK Latch: NAND Version



J	K	Q_n	$\overline{Q_n}$	S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	0	1	1	1	0	1	*hold*
		1	0	1	1	1	0	

- The SR latch has a problem in that when both S and R are high, its state becomes indeterminate
- The JK latch shown at left eliminates this problem by using feedback from output to input, such all states in the truth table are allowable
  - If J = K = 0, the latch will hold its present state

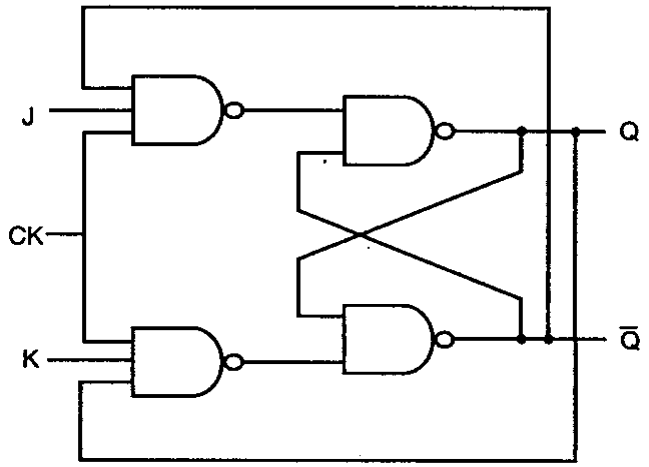
Clocked CMOS JK Latch: NAND Version



J	K	Q_n	$\overline{Q_n}$	S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	0	1	1	1	0	1	*hold*
		1	0	1	1	1	0	
0	1	0	1	1	1	0	1	*reset*
		1	0	1	0	0	1	

- The SR latch has a problem in that when both S and R are high, its state becomes indeterminate
- The JK latch shown at left eliminates this problem by using feedback from output to input, such all states in the truth table are allowable
  - If J = K = 0, the latch will hold its present state
  - If J = 0 and K = 1, the latch will reset on the next positive-going clock edge, i.e. Q' = 1 and Q = 0

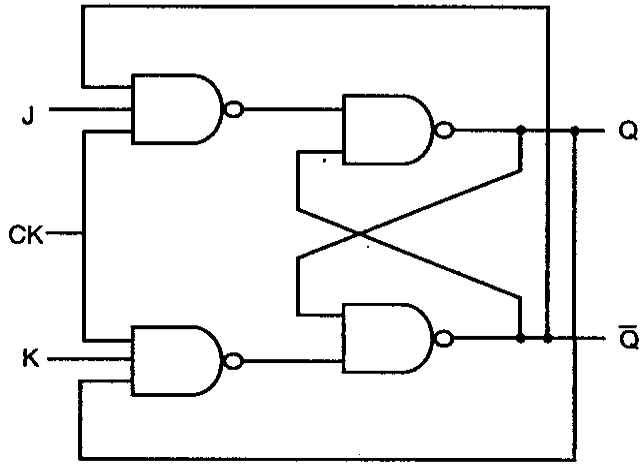
Clocked CMOS JK Latch: NAND Version



J	K	Q_n	$\overline{Q_n}$	S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Operation
0	0	0	1	1	1	0	1	*hold*
		1	0	1	1	1	0	
0	1	0	1	1	1	0	1	*reset*
		1	0	1	0	0	1	
1	0	0	1	0	1	1	0	*set*
		1	0	1	1	1	0	

- The SR latch has a problem in that when both S and R are high, its state becomes indeterminate
- The JK latch shown at left eliminates this problem by using feedback from output to input, such all states in the truth table are allowable
  - If J = K = 0, the latch will hold its present state
  - If J = 0 and K = 1, the latch will reset on the next positive-going clock edge, i.e. Q' = 1 and Q = 0
  - If J = 1 and K = 0, the latch will set on the next positive-going clock edge, i.e. Q = 1, Q' = 0

Clocked CMOS JK Latch: NAND Version



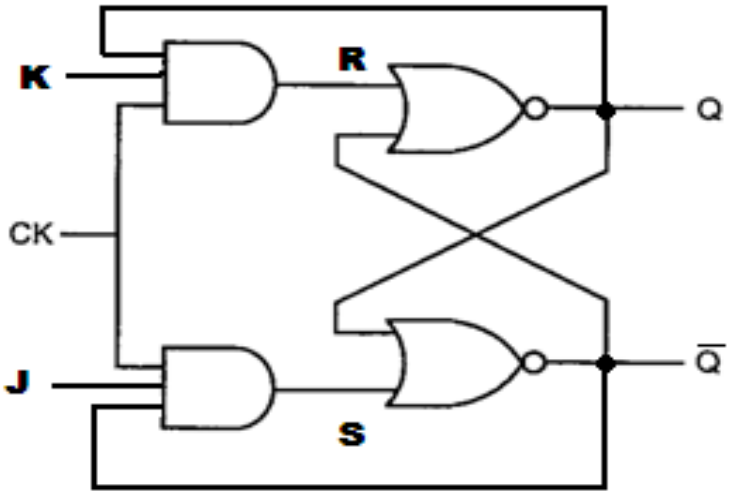
J	K	Q_n	$\overline{Q_n}$	S	R	Q_{n+1}	$\overline{Q_{n+1}}$	*Operation*
0	0	0	1	1	1	0	1	*hold*
		1	0	1	1	1	0	
0	1	0	1	1	1	0	1	*reset*
		1	0	1	0	0	1	
1	0	0	1	0	1	1	0	*set*
		1	0	1	1	1	0	
1	1	0	1	0	1	1	0	*toggle*
		1	0	1	0	0	1	

- The SR latch has a problem in that when both S and R are high, its state becomes indeterminate
- The JK latch shown at left eliminates this problem by using feedback from output to input, such all states in the truth table are allowable
  - If J = K = 0, the latch will hold its present state
  - If J = 1 and K = 0, the latch will set on the next positive-going clock edge, i.e. Q = 1, Q' = 0
  - If J = 0 and K = 1, the latch will reset on the next positive-going clock edge, i.e. Q' = 1 and Q = 0
  - If J = K = 1, the latch will toggle on the next positive-going clock edge
    -

Clocked CMOS JK Latch: NAND Version

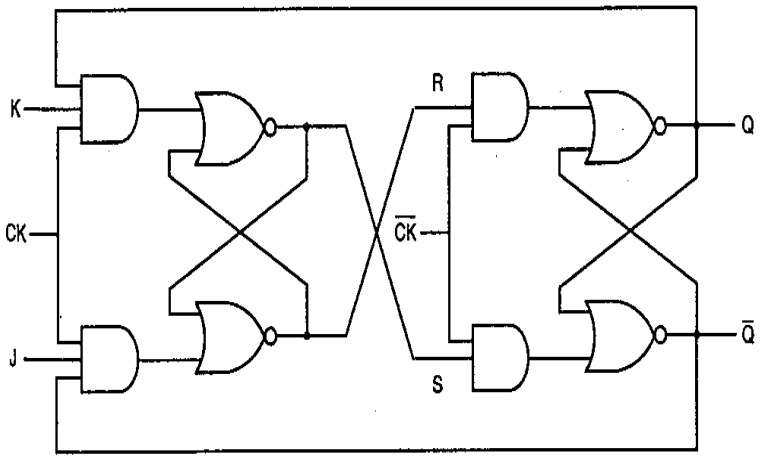
- Note that in order to prevent the JK Latch above from oscillating continuously during the clock active time, the clock width must be kept smaller than the switching delay (input-to-output propagation delay) time of the latch. Otherwise, several oscillations may occur before the clock goes low again. In practice this may be difficult to achieve.

Clocked JK latch



Gate- level schematic of the clocked NOR-based JK latch circuit

JK Master-Slave Flip-Flop



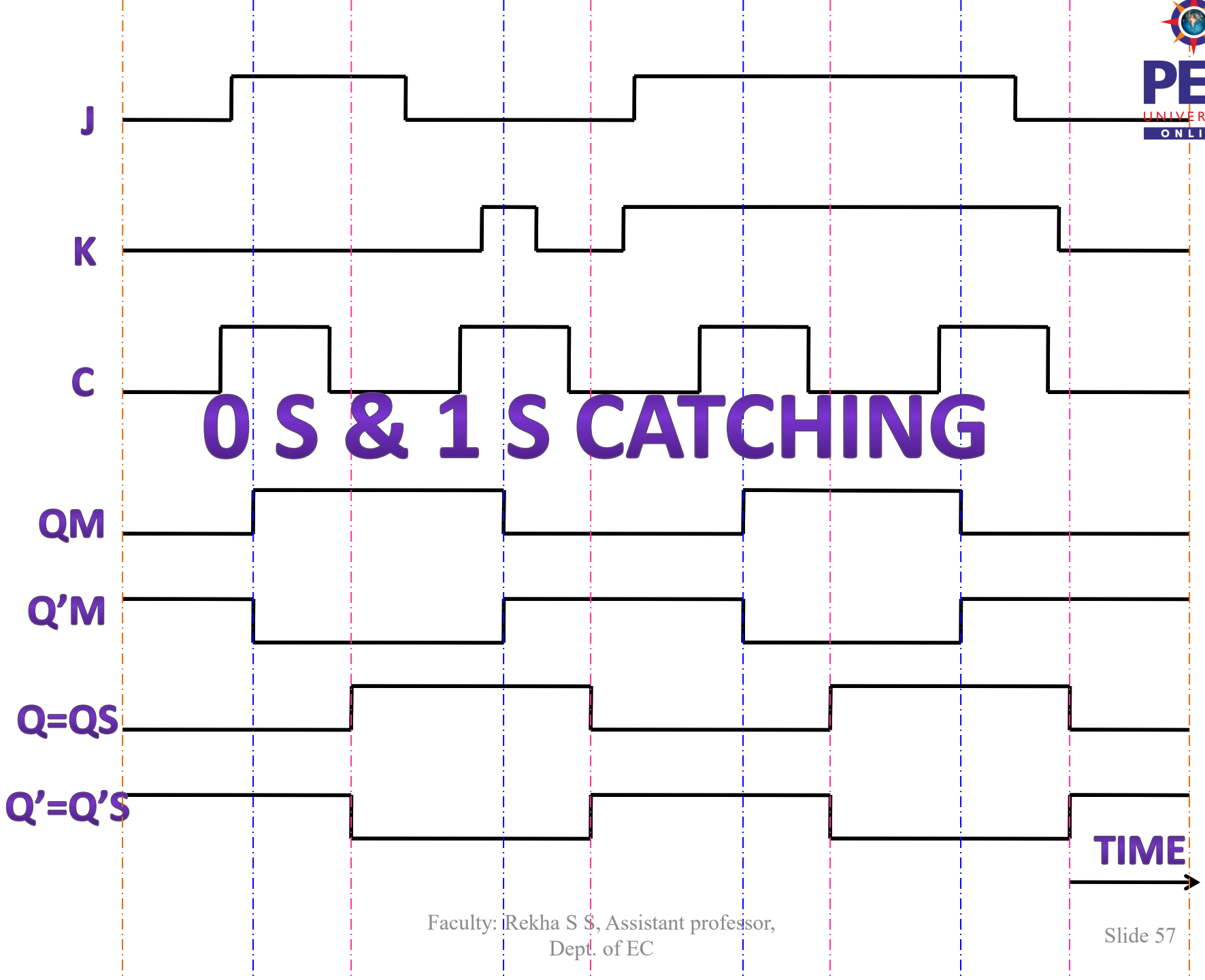
Can be susceptible to "ones catching", i.e. a positive glitch in either the J or K input while the CLK is high, which can change the state of the master latch (and the slave latch on next edge)

A Flip-Flop is defined as two latches connected serially and activated with opposite phase clocks

- First latch is the Master; Second latch is the Slave
- Eliminates transparency, i.e. a change occurring in the primary inputs is never reflected directly to the outputs, since opposite phase clocks are used to activate the M and S latches.

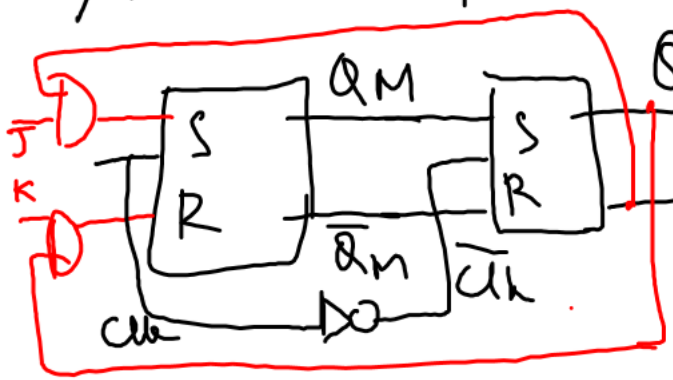
A JK master-slave flip-flop (NOR-based version) is shown below:

- The feedback paths occur from Q and Q' slave outputs to the master inputs AOI gates
- does not exhibit any tendency to oscillate when J = K = 1 no matter how long the clock period, since opposite clock phases activate the master and slave latches separately.


J
K
C
0 S & 1 S CATCHING
QM
Q'M
Q=QS
Q'=Q'S
TIME

0's Catching

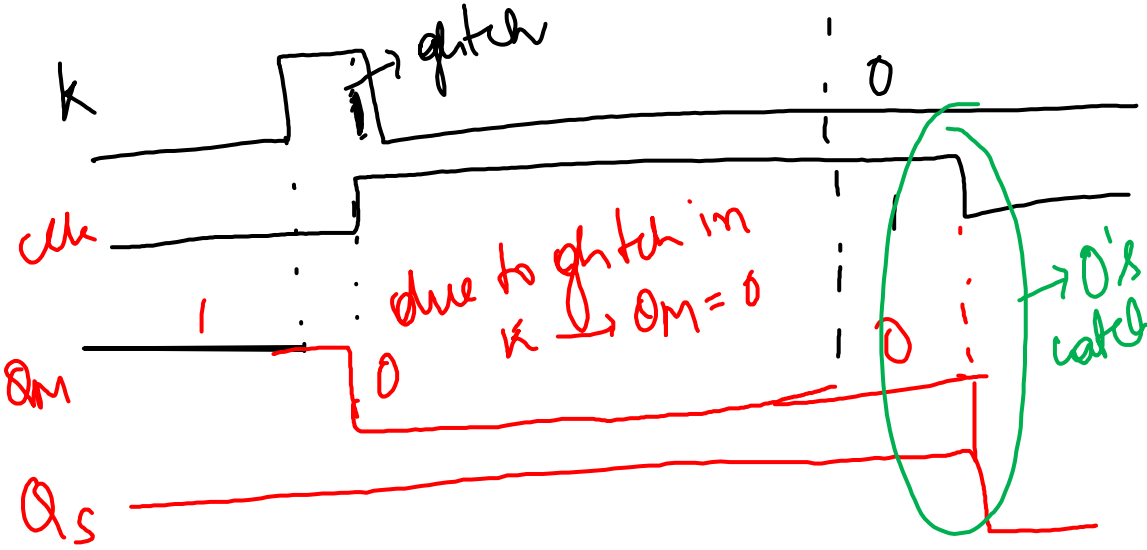
Assume $Q = 1$, $\bar{Q} = 0$



And gate
$J = 1, \bar{Q} = 0$
so O/P will be 0

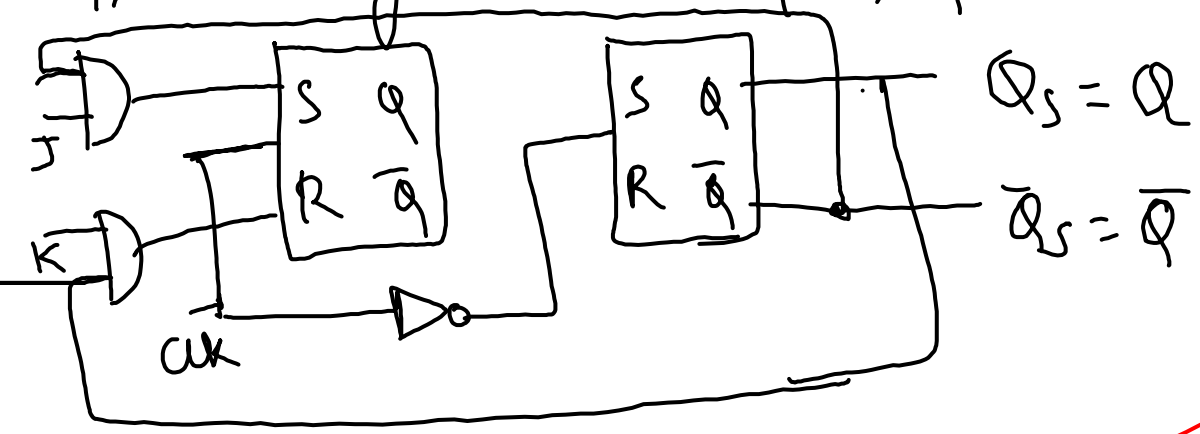
$K = 0$ & $Q = 1$, and gate O/P will 0
so $S = R = 0$, ⇒ previous value

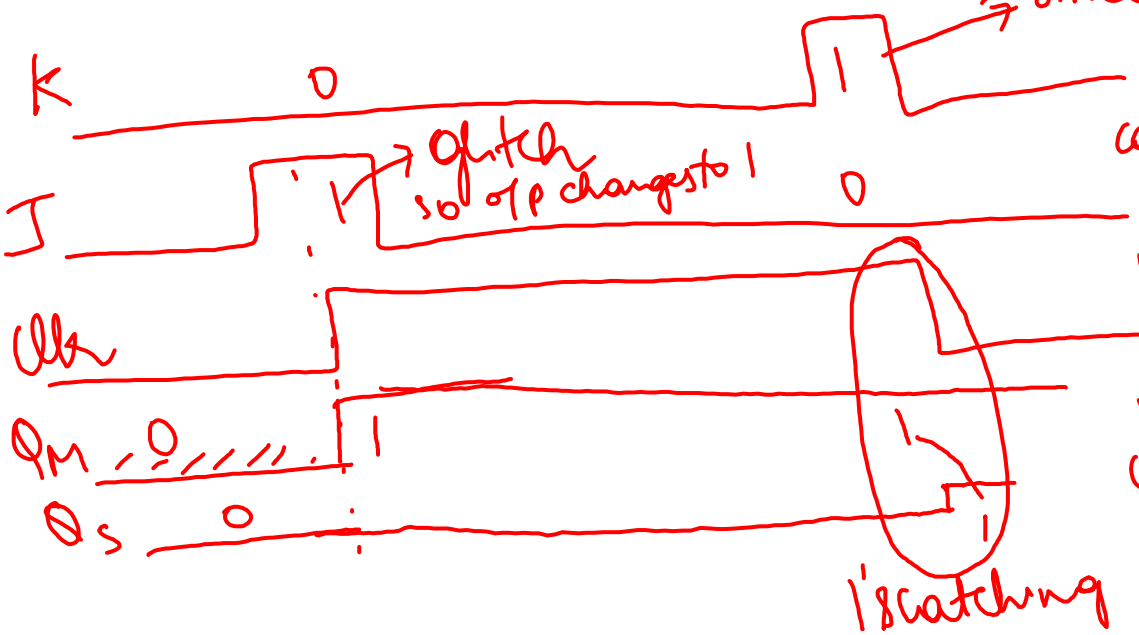
Change in J from 0 to 1



Q_M will not change due 0's Catching because of glitch and same 0 is reflected at Q_S

1's Catching. Assume $Q = 0$, $\bar{Q} = 1$





→ here AND gate input $J = 0$ & $\bar{Q} = 1$

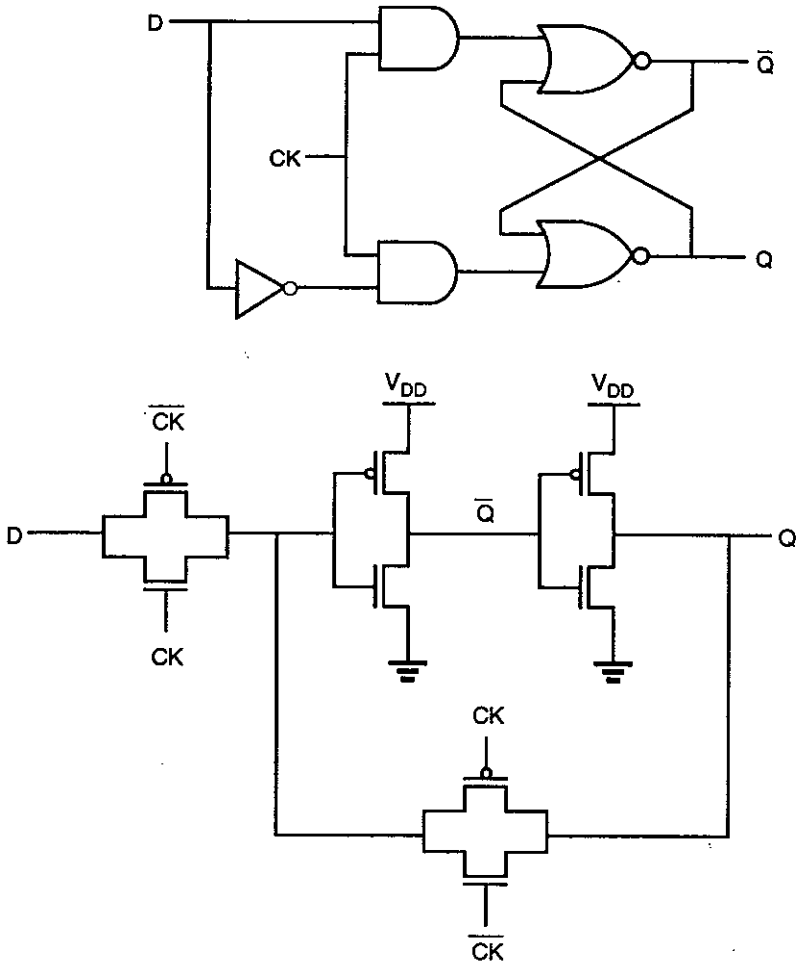
→ Since $J = 0$, $K = 1$ o/p will cannot change to 0 will be the same 1 which is observed due to the glitch in J

So AND gate o/p will be zero fed to AND gate

$K = 1$ & $Q = 0$ So o/p the respective and gate is zero

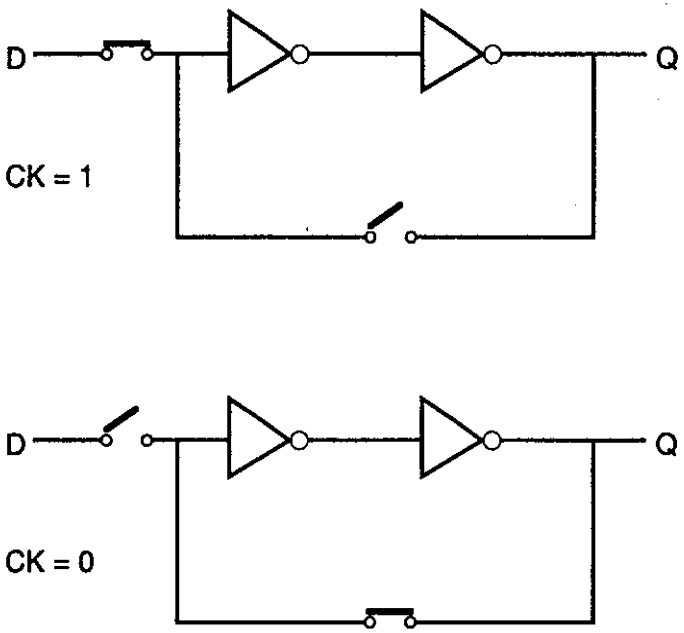
$S = R = 0 \Rightarrow$ retains previous value logic, which is due to glitch

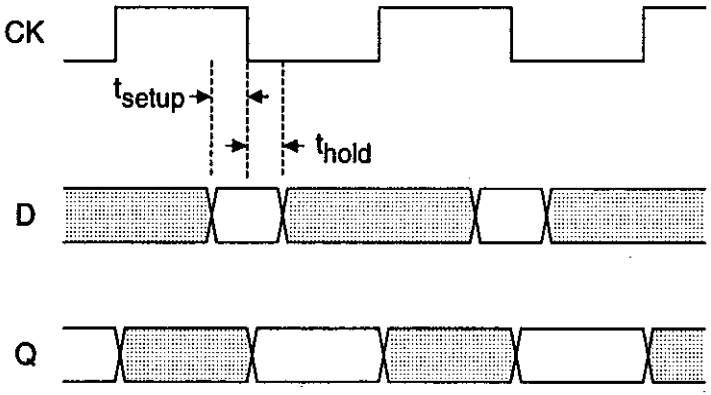
CMOS D-Latch Implementation



- A D-latch is implemented, at the gate level, by simply utilizing a NOR-based S-R latch, connecting D to input S, and connecting D' to input R with an inverter.
  - When CLK goes high, D is transmitted to output Q (and D' to Q')
  - When CLK goes low, the latch retains its previous state
- The D latch is normally implemented with transmission gate (TG) switches, as shown at the left
  - The input TG is activated with CLK while the latch feedback loop TG is activated with CLK'
  - Input D is accepted when CLK is high
  - When CLK goes low, the input is open-circuited and the latch is set with the prior data D

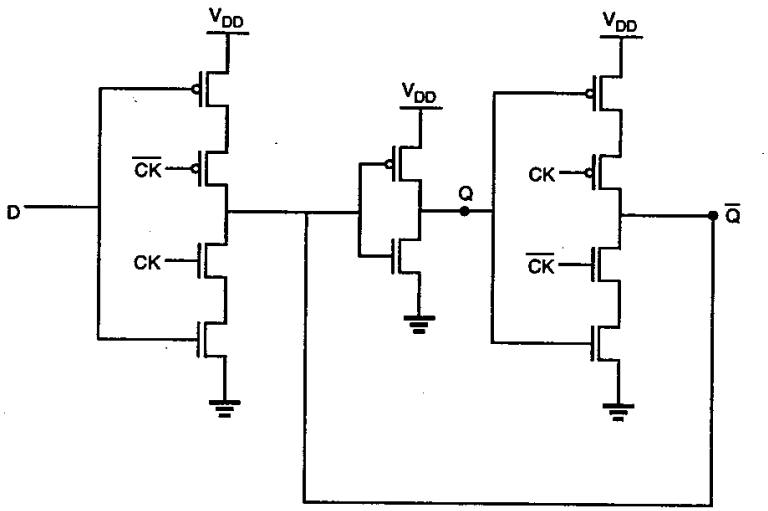
CMOS D-Latch Schematic View and Timing





- A schematic view of the D-Latch can be obtained using simple switches in place of the TG's
  - When CLK = 1, the input switch is closed allowing new input data into the latch
  - When CLK = 0, the input switch is opened and the feedback loop switch is closed, setting the latch
- Timing diagram:
  - In order to guarantee adequate time to get correct data at the first inverter input before the input switch opens, the data must be valid for a given time (T_{setup}) prior to the CLK going low.
  - In order to guarantee adequate time to set the latch with correct data, the data must remain valid for a time (T_{hold}) after the CLK goes low.
  - Violations of T_{setup} and T_{hold} can cause metastability problems and chaotic transient behavior.

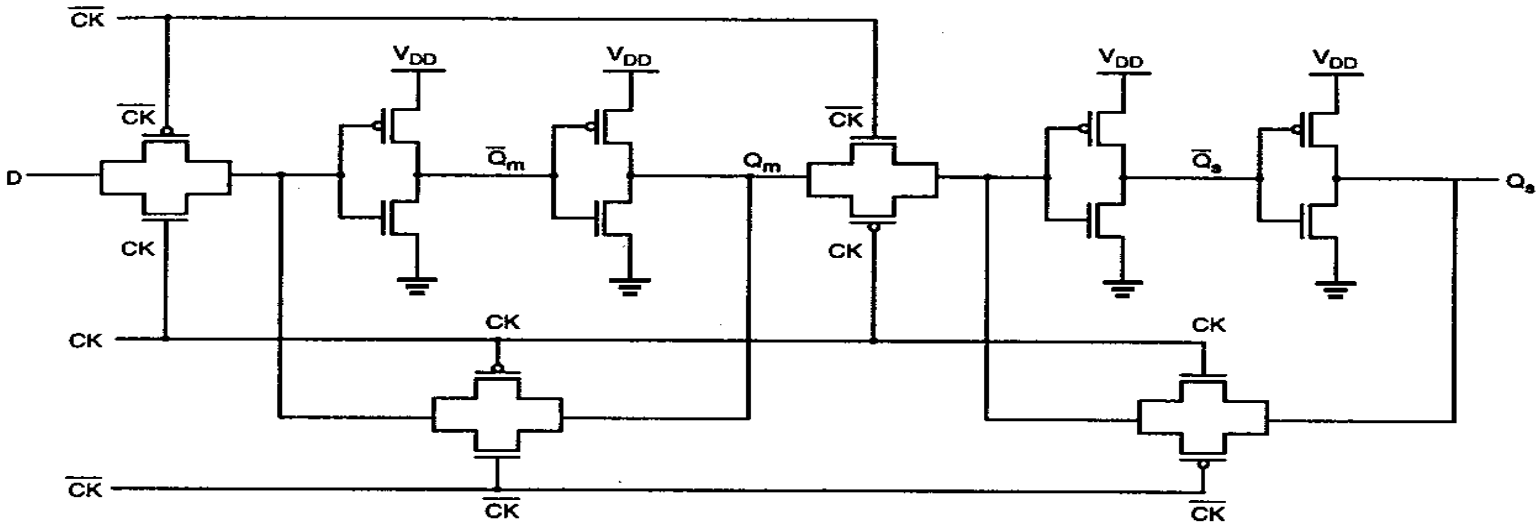
Alternate CMOS D-Latch Implementation

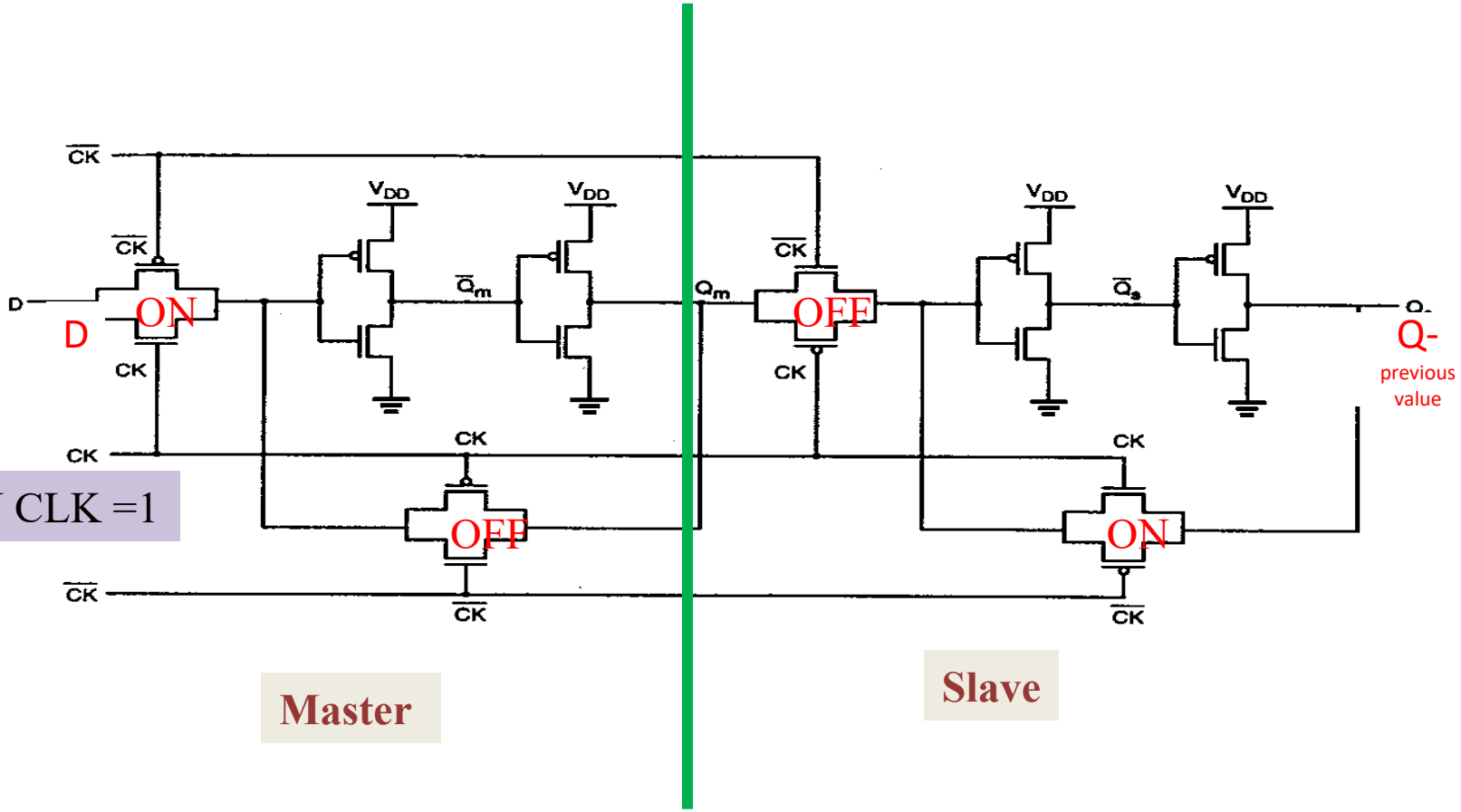


- An alternate (preferred) version of the CMOS D-Latch (shown at left) is implemented with two tri-state inverters and a normal CMOS inverter.
- Functionally it is similar to the previous chart D-Latch
  - When CLK is high, the first tri-state inverter sends the inverted input through to the second inverter, while the second tri-state is in its high Z state.
    - Output Q is following input D
  - When CLK is low, the first tri-state goes into its high Z state, while the second tri-state inverter closes the feedback loop, holding the data Q and Q' in the latch.

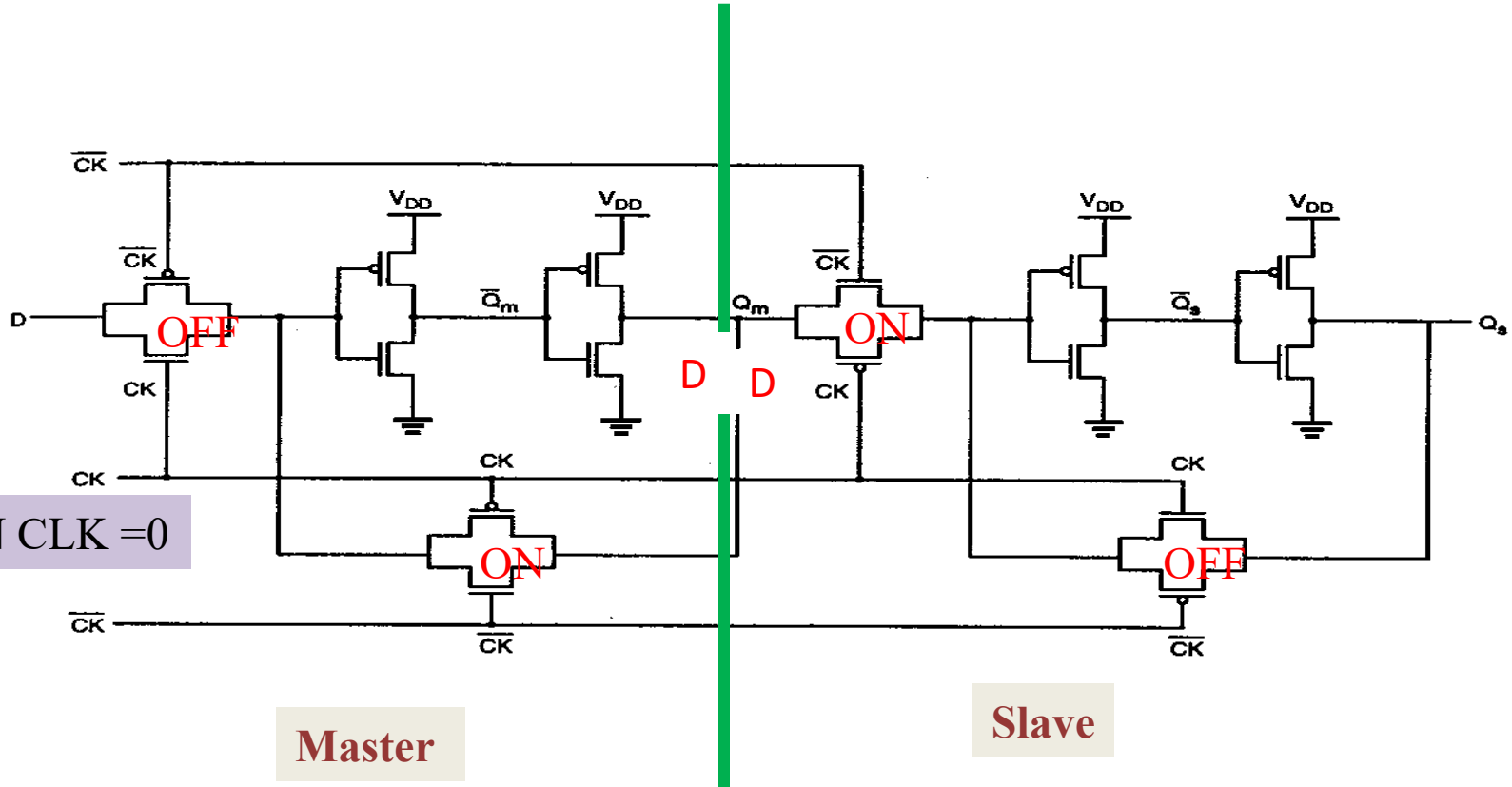
CMOS D Flip-Flop: Falling Edge-Triggered

- Shown below is a D Flip-Flop, constructed by cascading two D-Latch circuits from the previous chart
  - Master latch is positive level sensitive (receives data when CLK is high)
  - Slave latch is negative level sensitive (receives data Qm when CLK is low)
- The circuit is **negative-edge triggered**



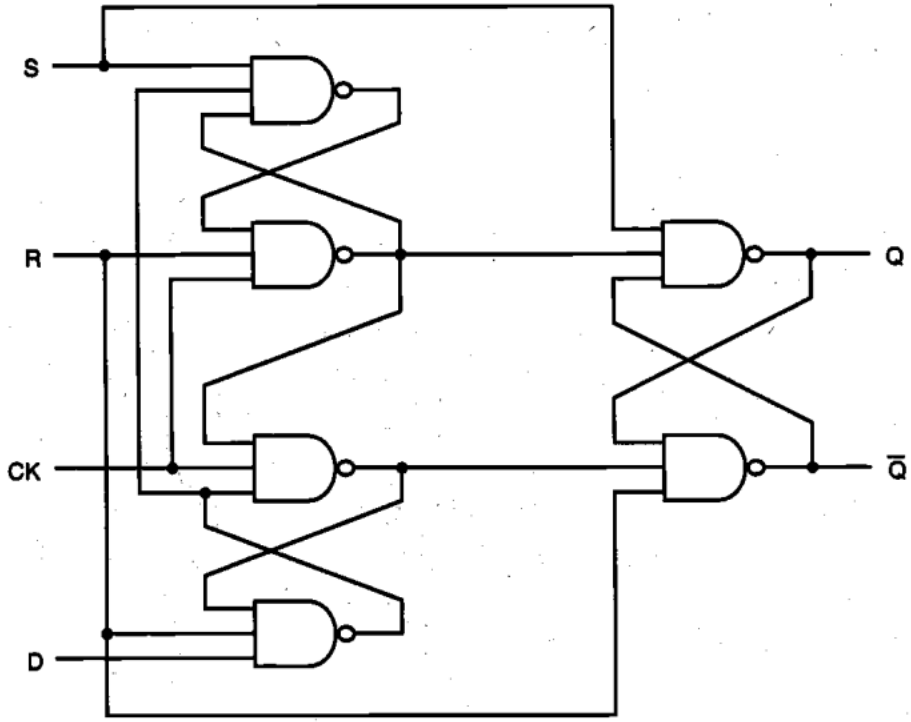


When Clock = 1, Present date is reflected at output of master. But, the output of the slave will be previous value



When Clock = 0, master is disabled and output of master will have the value which is latched during clk =1.
Slave is enabled and the input which latched when clk=1 is now reflected at the output of the slave

Positive edge-triggered D Flip-flop



S	R	Clk	D	Q	$\bar{Q}$	operation
0	1	X	X	1	0	→ Set
1	0	X	X	0	1	→ Reset
1	1	↑	0	0	1	Copies D
1	1	↑	1	1	0	Copies D
1	1	0	X	previous value		
1	1	1	X			

Sequential Circuit Design

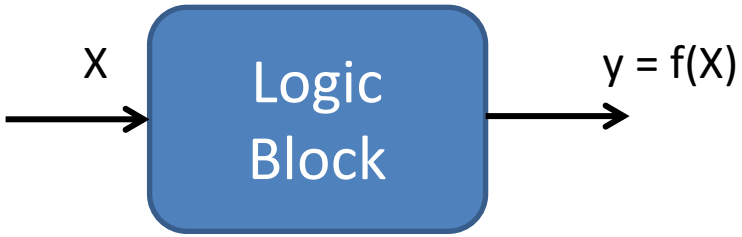
Reference Books:
R2: CMOS VLSI design A Circuits and Systems Perspective, Neil Weste and David Harris, 3rd Edition.
Chapter 10 (10.1 to 10.2) – as per Soft copy
Sequencing static circuits, Sequencing Methods, Max-Delay Constraints, Min-Delay Constraints, Time Borrowing, Clock Skew, Problems on Max and Min Delay Constraints at design level

INTRODUCTION

Digital Circuits

Combinational Circuits

- Output is a function of the current inputs

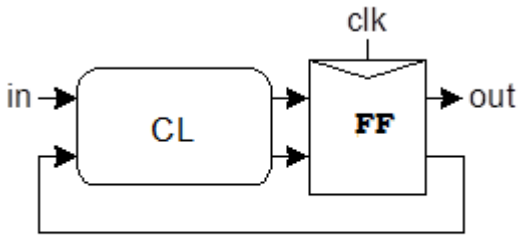


Sequential Circuits

- Output depends on previous as well as current inputs
- Such circuit are said to have state (like present state, next state, previous state)
- FSM (finite state machine) and pipeline are two examples of sequential circuits.

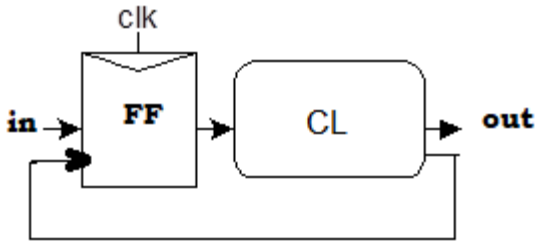
INTRODUCTION

- **FSM**

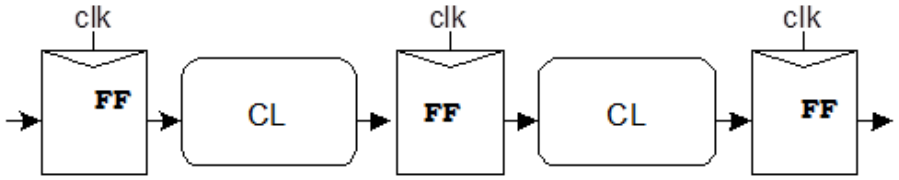


Finite State Machine

Or



- **Pipeline**



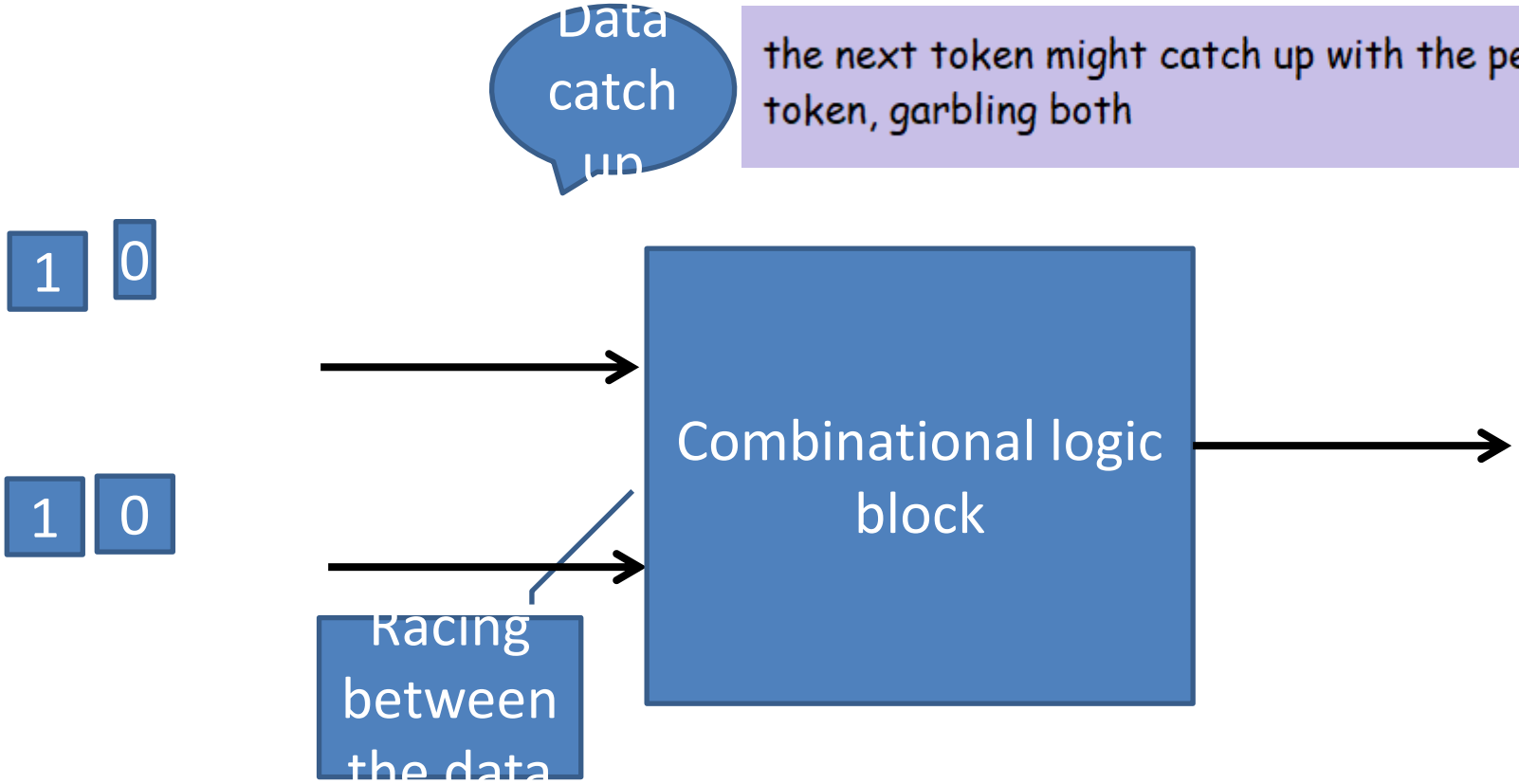
Pipeline

CL --- Combinational Logic Block

FF --- Flip Flop

Sequential Circuits Design

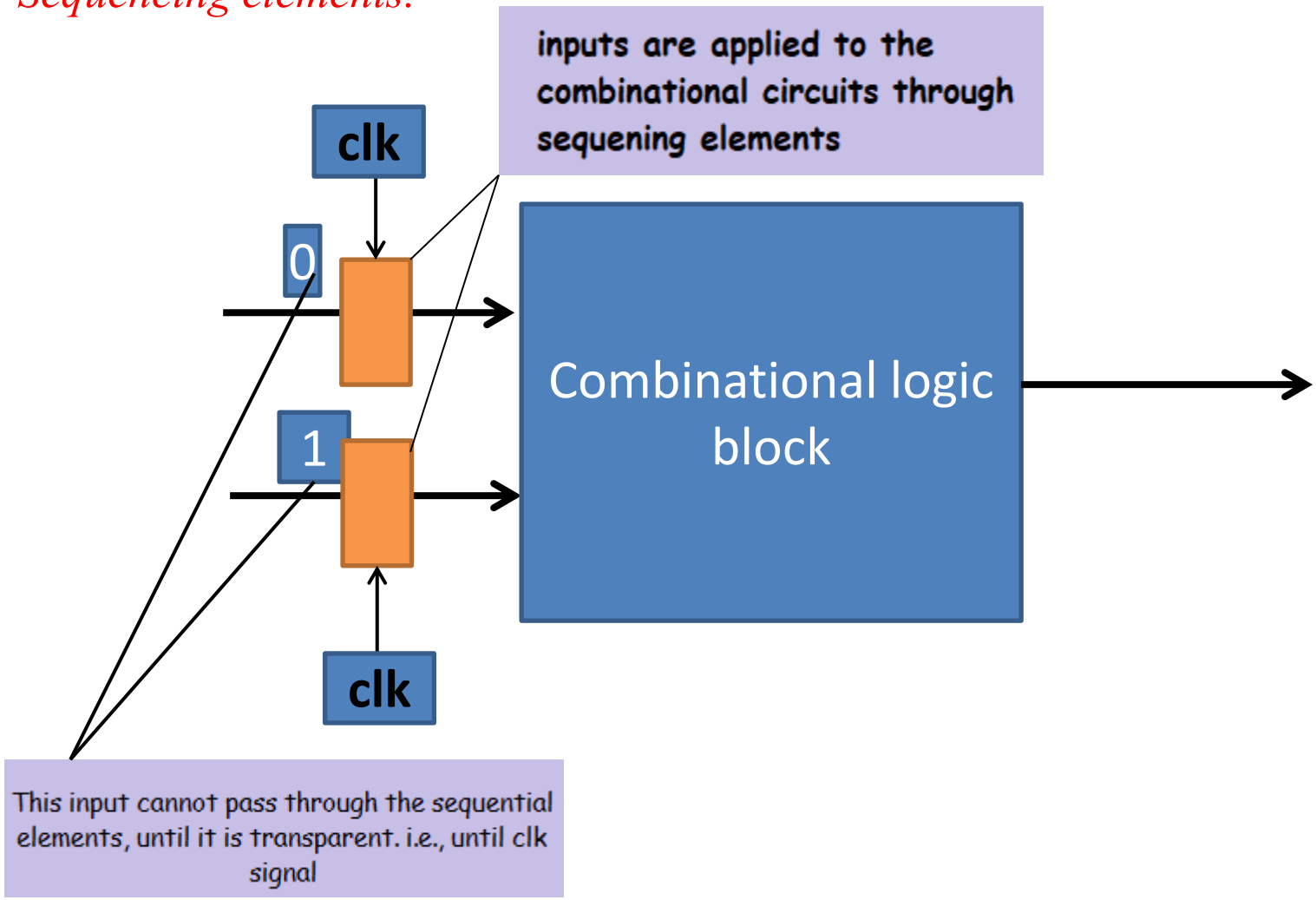
- Flip flops or latches, are sometimes called memory elements, that hold data called tokens.



- The purpose of these (Flip flop or latch)elements is not really memory; instead, it is to enforce sequence, to distinguish the ***current token*** from the ***previous or next token*** -- *Sequencing elements.*

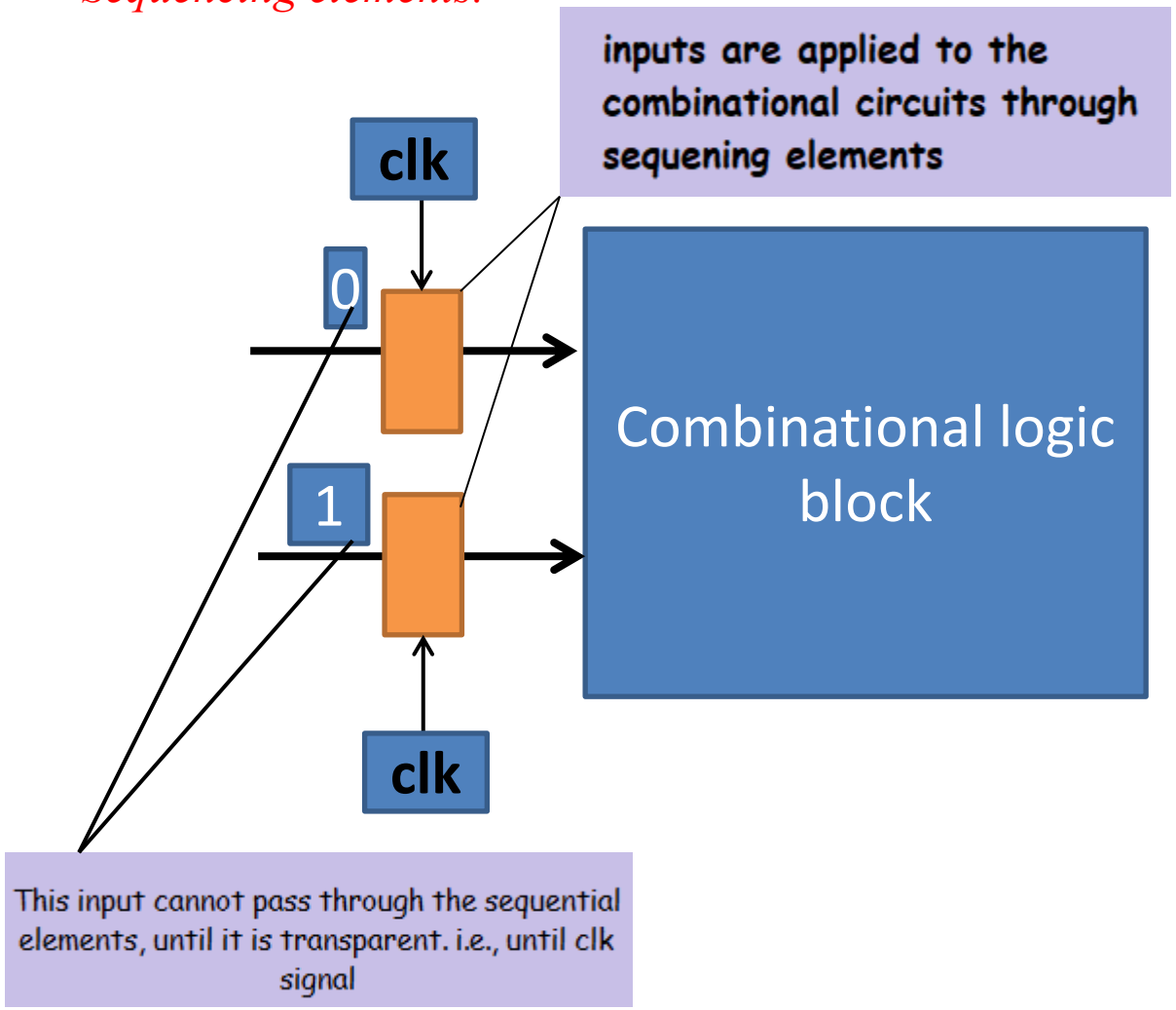
Sequential Circuits Design

- The purpose of these elements is not really memory; instead, it is to enforce sequence, to distinguish the ***current token*** from the ***previous or next token*** -- *Sequencing elements.*



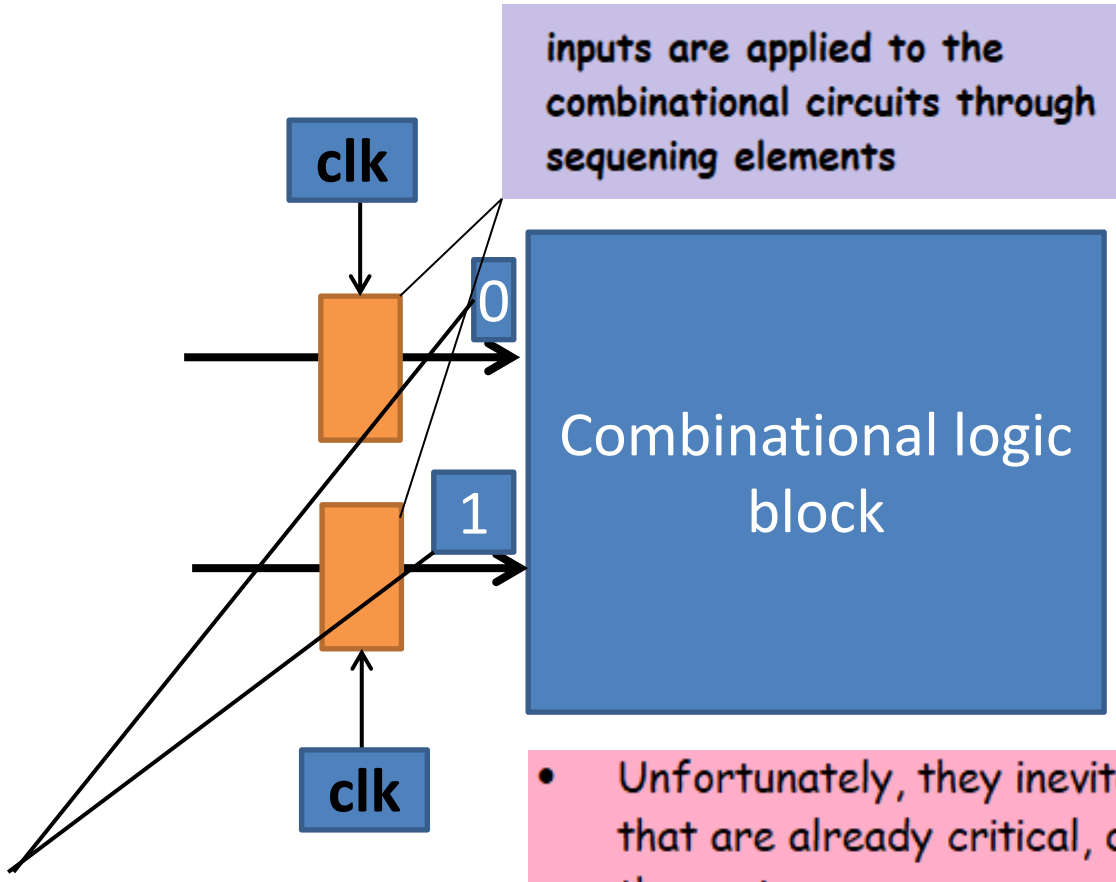
Sequential Circuits Design

- The purpose of these elements in not really memory; instead, it is to enforce sequence, to distinguish the ***current token*** from the ***previous or next token*** -- *Sequencing elements.*



Sequential Circuits Design

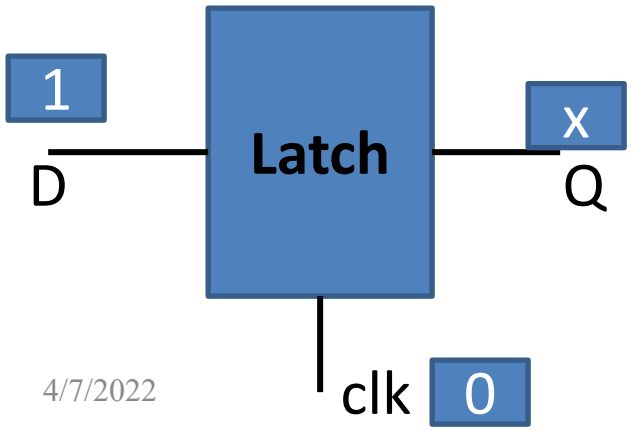
- without sequencing elements, the next token might catch up with the pervious token, garbling both.
  - Sequencing elements delay tokens that arrive too early, preventing them from catch up with the pervious tokens



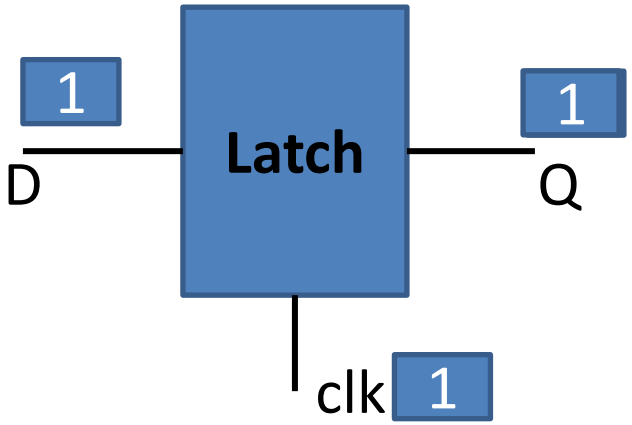
- Unfortunately, they inevitably add some delay to tokens that are already critical, decreasing the performance of the system.
- This extra delay is called ***Sequencing Overhead***

Sequencing static circuits

- Flip Flop and Latches are most commonly used sequencing elements.
- Both have three terminals
  - Data input (D)
  - Data output (Q)
  - Clock Input (clk)
- Latch is transparent when the clock is high and opaque when the clock is low



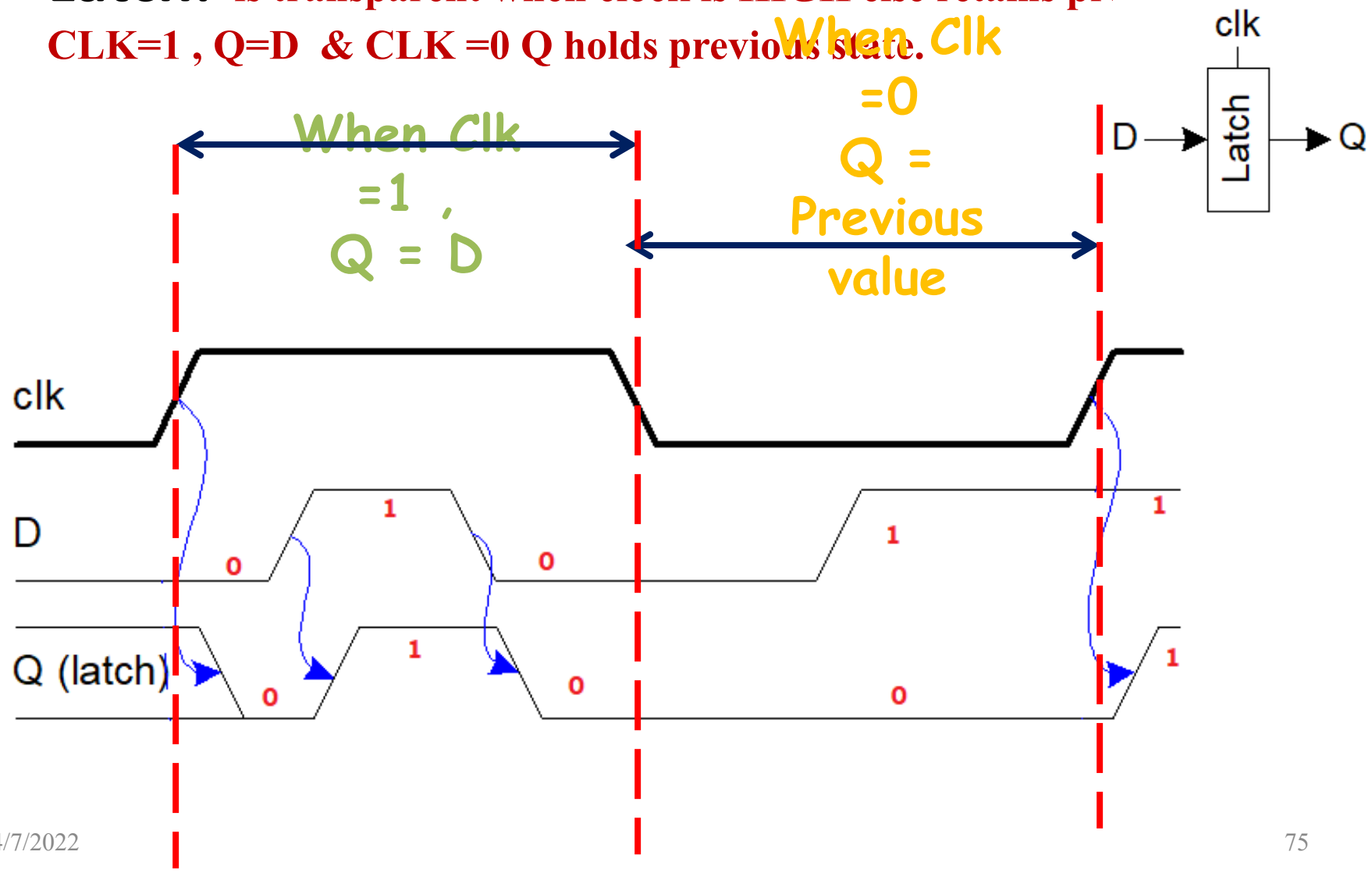
Clk	Q
1	D
0	Previous state



Sequencing static circuits

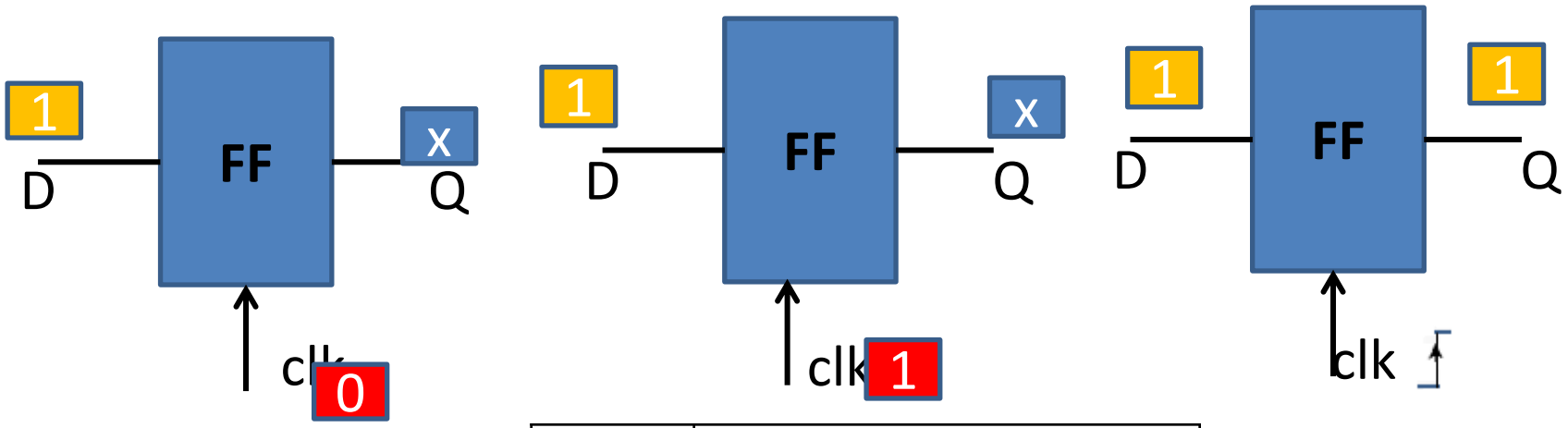
- **Latch:** is transparent when clock is HIGH else retains previous state

CLK=1 , Q=D & CLK =0 Q holds previous state.



Sequencing static circuits

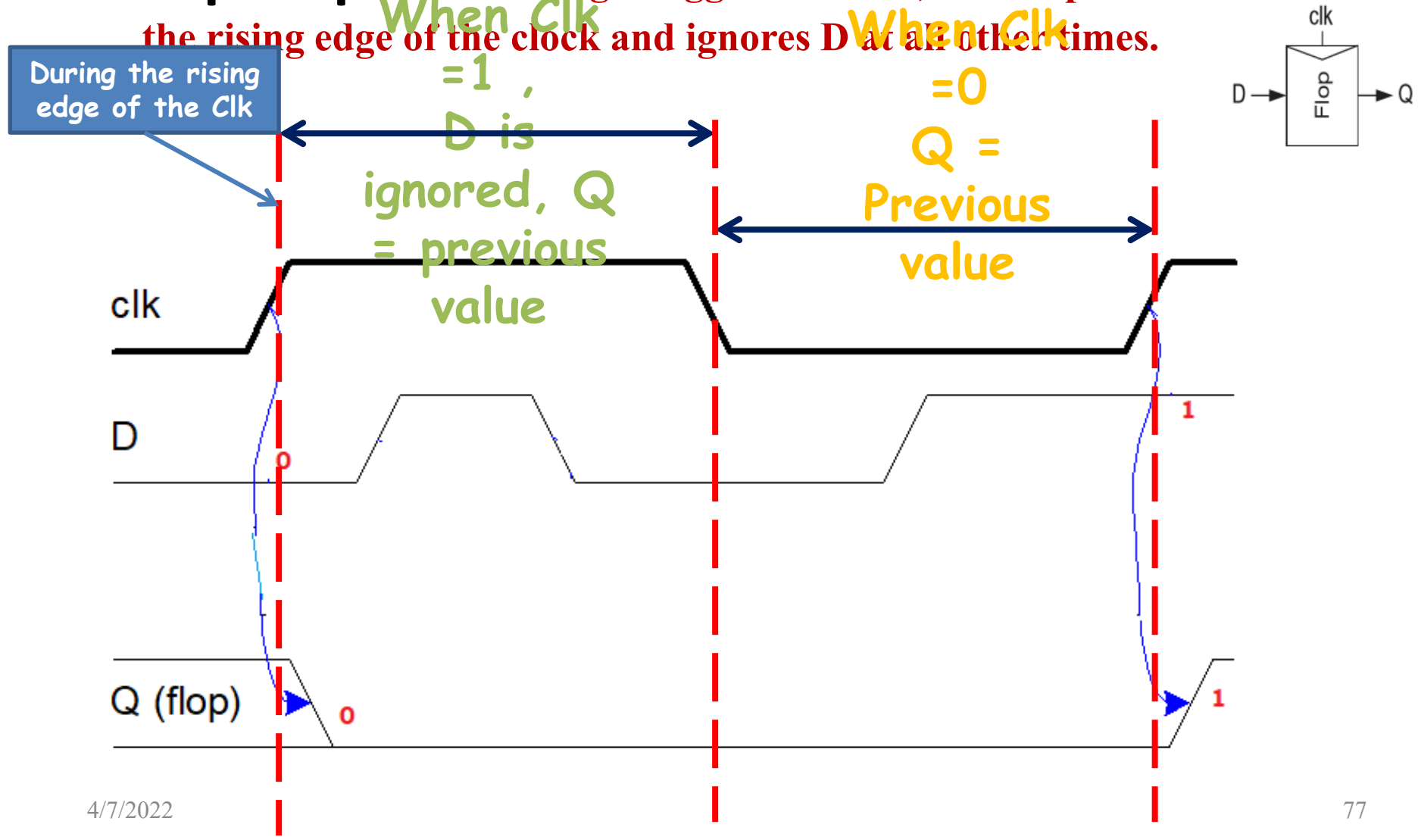
- **Flip Flop**: **is a +ve edge-triggered device, that copies D to Q on the rising edge of the clock and ignores D at all other times.**



Clk	Q
1	No change
0	No change
↑	D

Sequencing static circuits

- **Flip Flop:** is a +ve edge-triggered device, that copies D to Q on the rising edge of the clock and ignores D at all other times.



Observations

- The purpose of FF and latch is not really memory; instead, it is to enforce sequence, to distinguish the ***current token*** from the ***previous or next token*** -- *Sequencing elements.*
- without sequencing elements, the next token might catch up with the pervious token, garbling both.
- Sequencing elements delay tokens that arrive too early, preventing them from catch up with the pervious tokens
- Unfortunately, they inevitably add some delay to tokens that are already critical, decreasing the performance of the system.
- This extra delay is called ***Sequencing Overhead***

Sequencing Methods of Combinational Logic

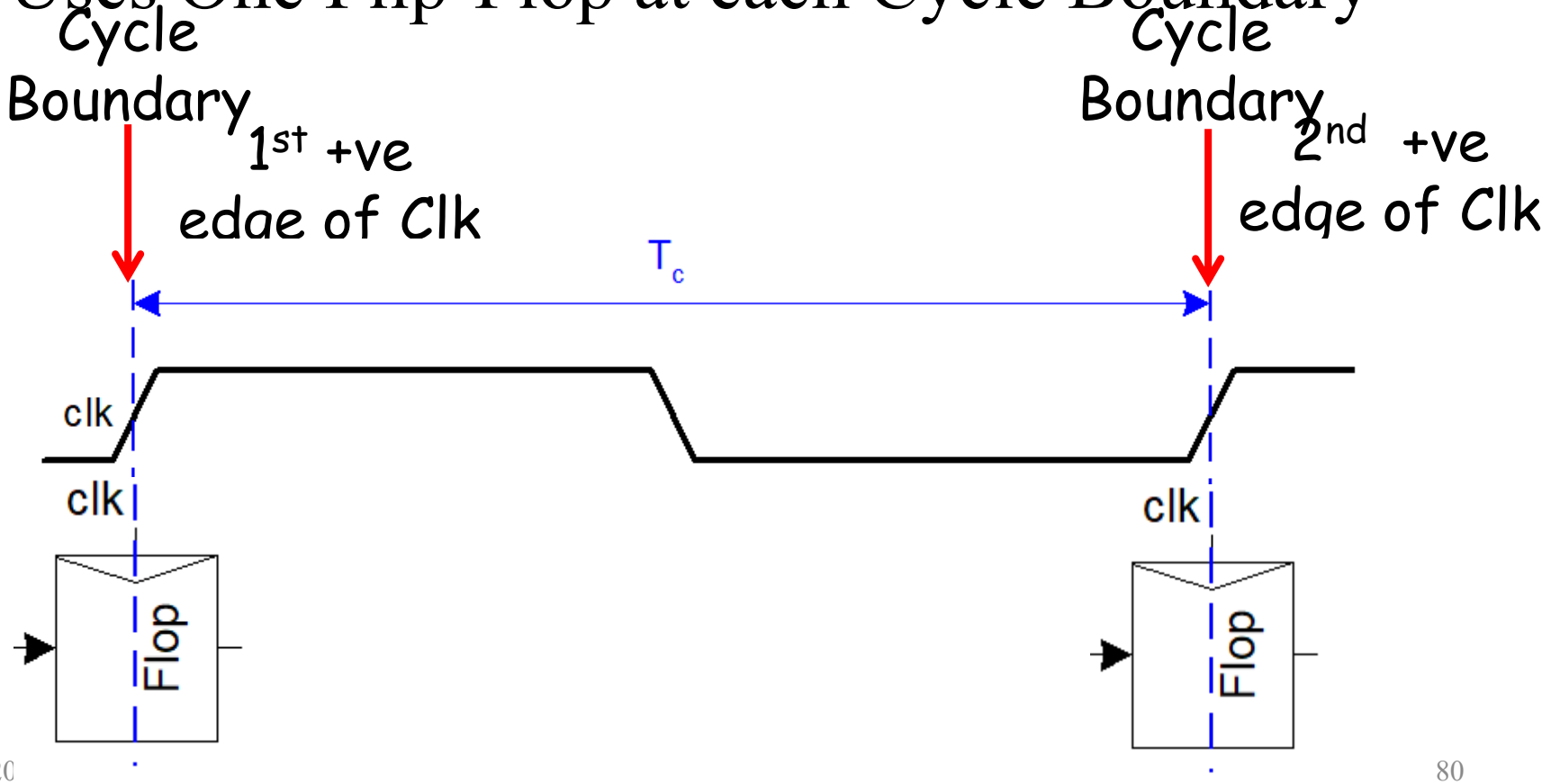
- There are three methods of sequencing blocks of combinational logic.

1. **Using Flip- Flop as Sequencing Element**
2. **Using Two phase Latches as Sequencing Element**
3. **Using Pulsed Latches as Sequencing Element**

Sequencing Methods of Combinational Logic

1. Using Flip- Flop as Sequencing Element

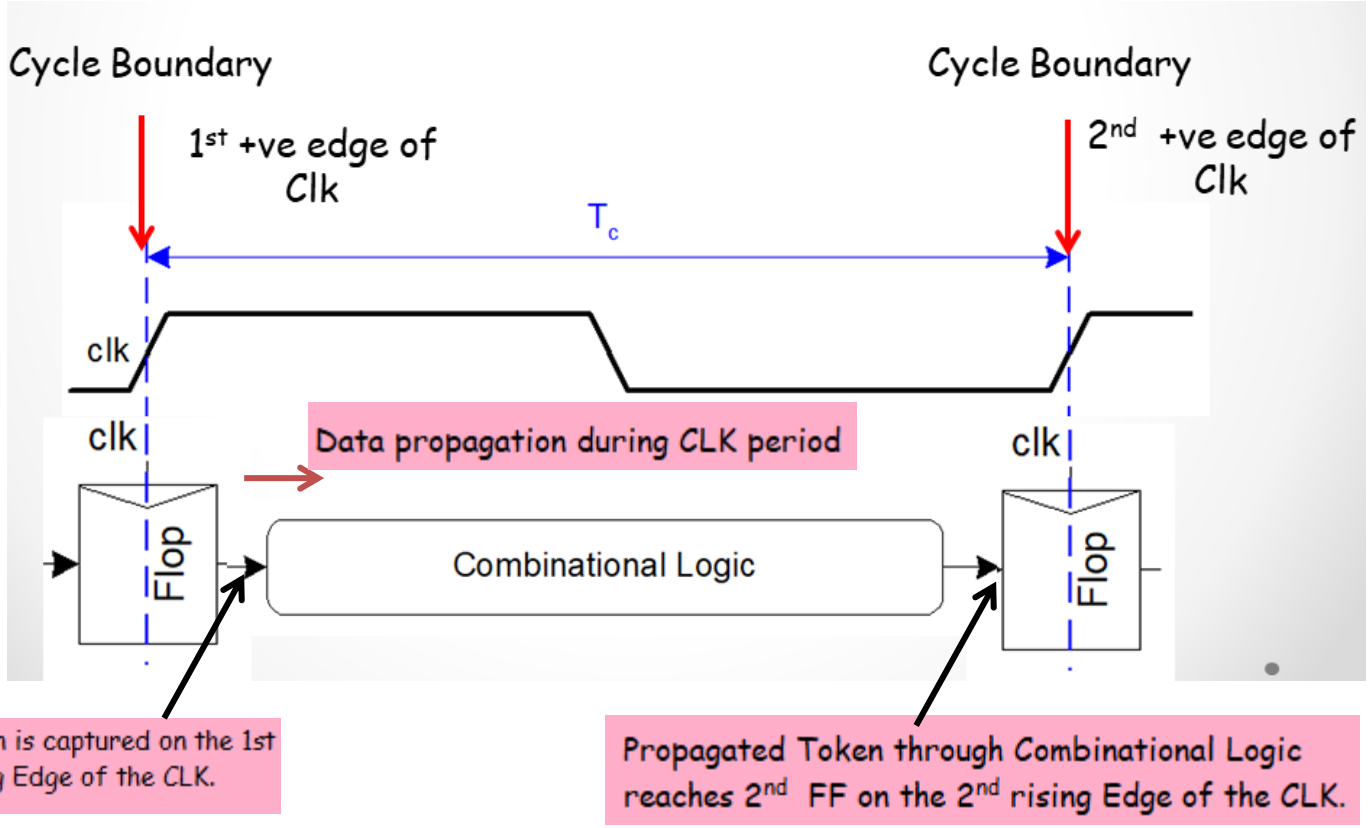
- Uses One Flip-Flop at each Cycle Boundary



Sequencing Methods of Combinational Logic

1. Using Flip- Flop as Sequencing Element

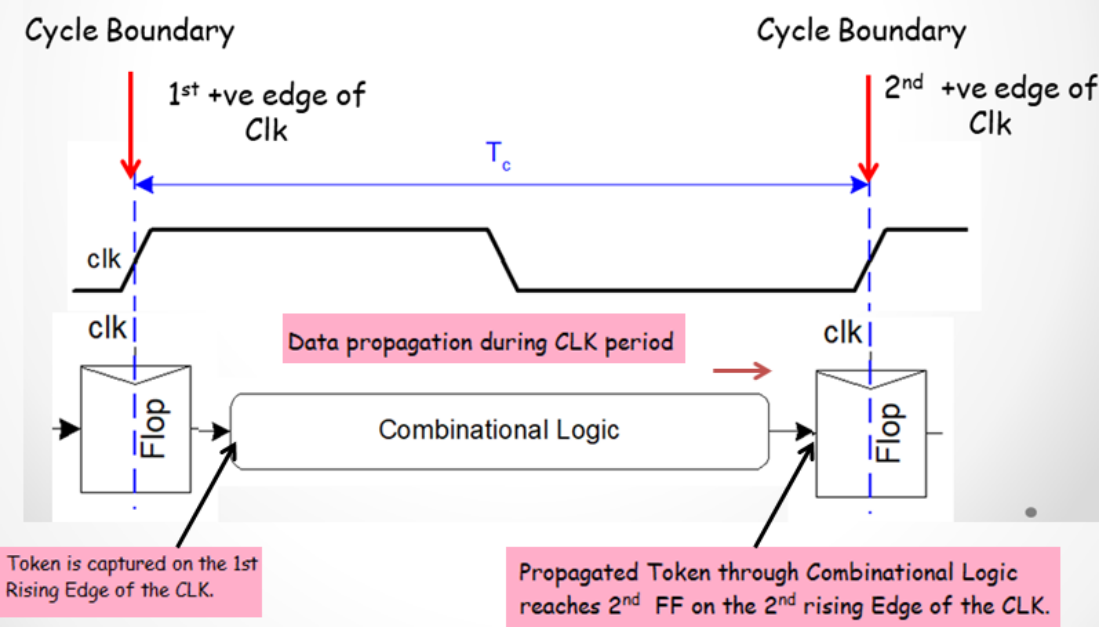
- Uses One Flip-Flop at each Cycle Boundary
- Tokens advances from one cycle to next on rising edge. If the token arrives early , it waits at the flip-flop unit the next cycle (rising edge)



Sequencing Methods of Combinational Logic

1. Using Flip- Flop as Sequencing Element

- Uses One Flip-Flop at each Cycle Boundary
- Tokens advances from one cycle to next on rising edge. If the token arrives early , it waits at the flip-flop unit the next cycle (rising edge)

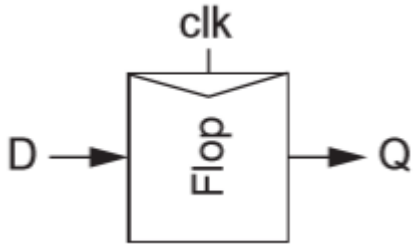


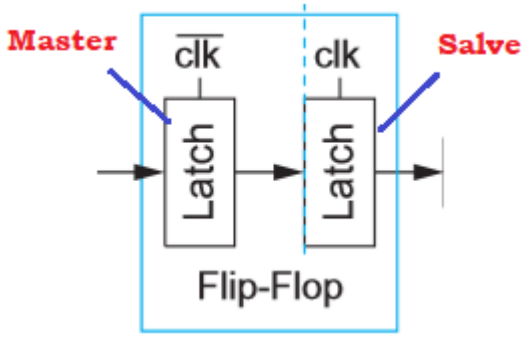
- So, data is captured in the first flip-flop on the first rising edge of the clock. Data propagates through combinational logic and reaches second flip-flop on the second rising edge of the clock.

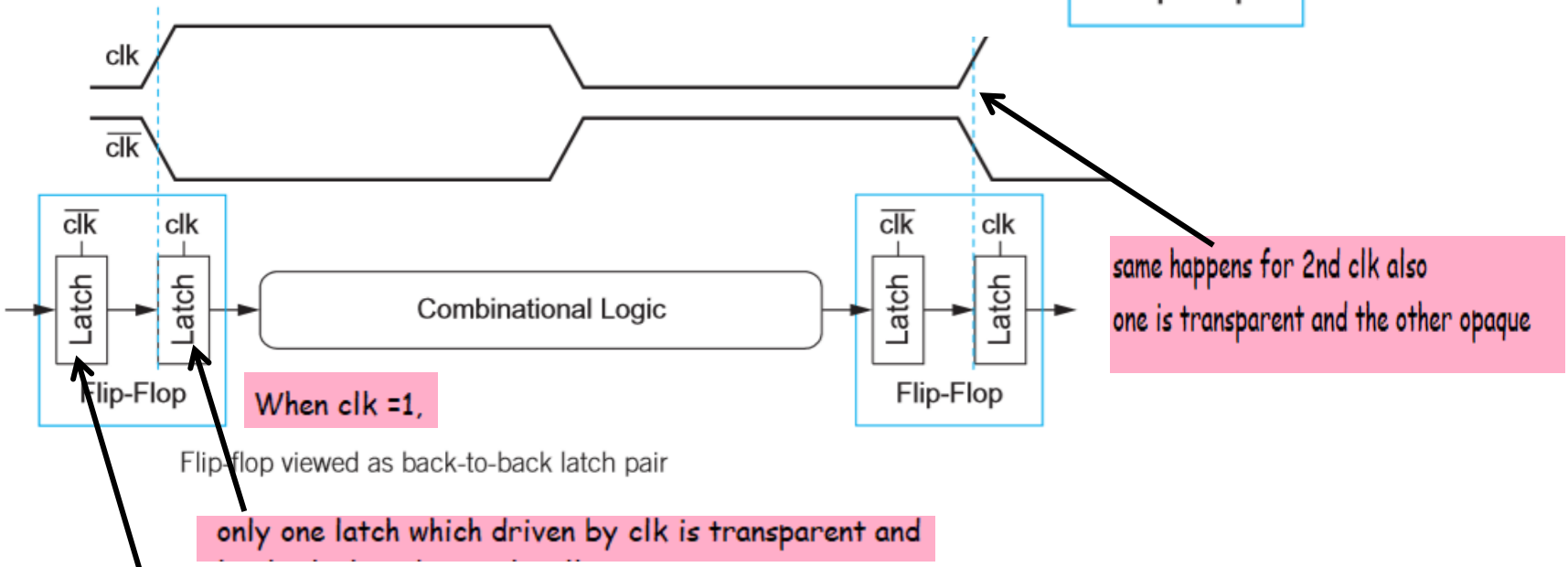
Sequencing Methods of Combinational Logic

2. Using Two phase Latches as Sequencing Element

Si[illegible]o can be construct[illegible]s fl[illegible]



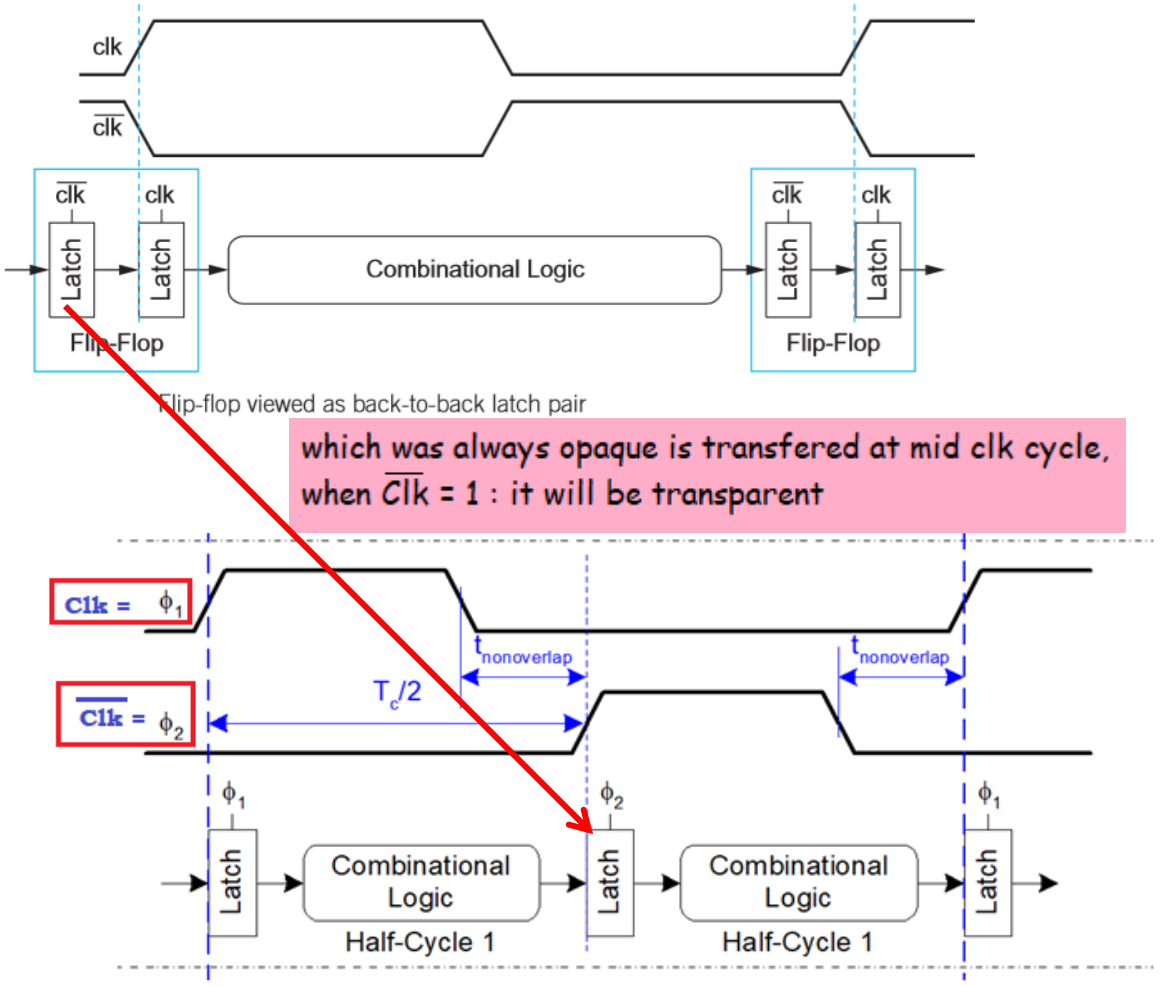




Sequencing Methods of Combinational Logic

2. Using Two phase Latches as Sequencing Element

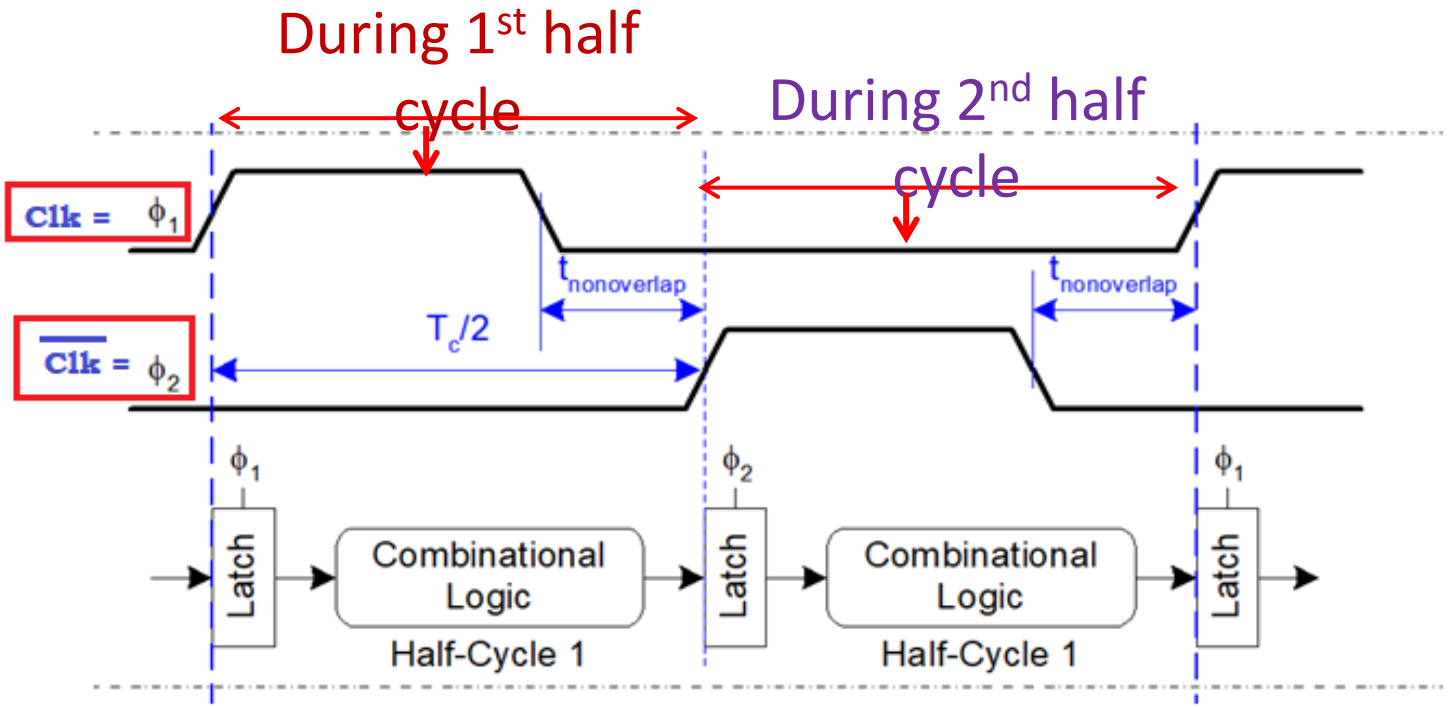
Since Flip Flop can be constructed using latch as flows



Sequencing Methods of Combinational Logic

2. Using Two phase Latches as Sequencing Element

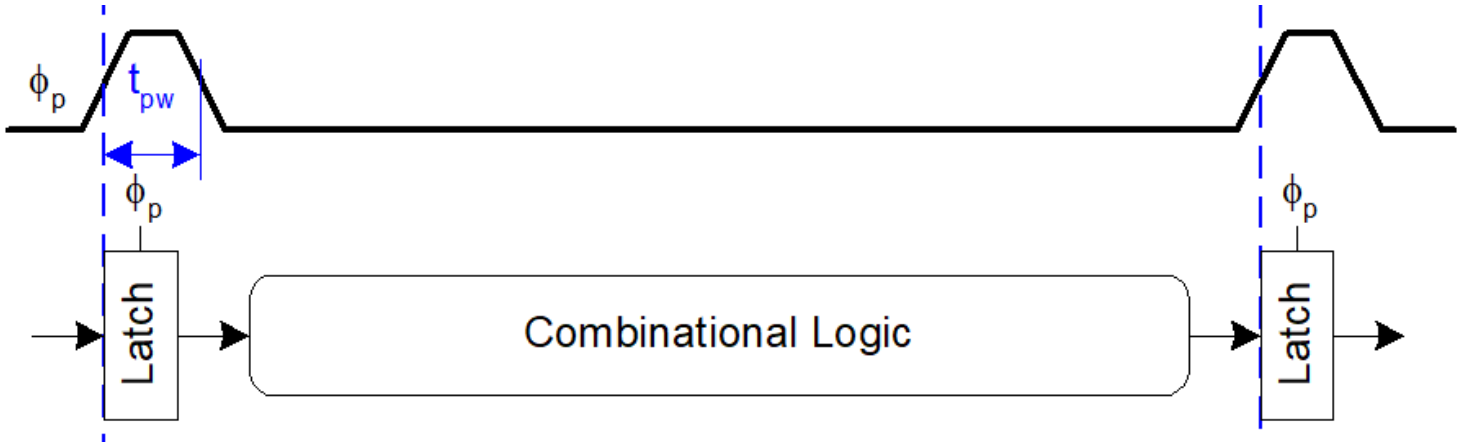
- Divide the full cycle of combinational logic into two phases, sometimes called *half-cycles*.
- Non Overlapping Two phase Clocks are used to Control two Latches and are referred as Φ1 and Φ2.
- At any given time, at least one clock is LOW and the corresponding latch is opaque, preventing one token from catching up with another.



Sequencing Methods of Combinational Logic

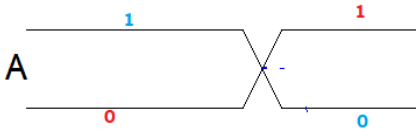
3. Using Pulsed Latches as Sequencing Element

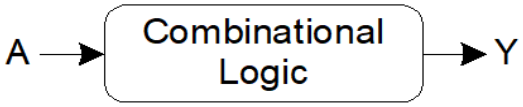
- Pulsed latch systems eliminate one of the latches from each cycle and apply a brief pulse to the remaining latch.
- If the pulse is shorter than the delay through the combinational logic, we can still expect that a token will only advance through one clock cycle on each pulse.



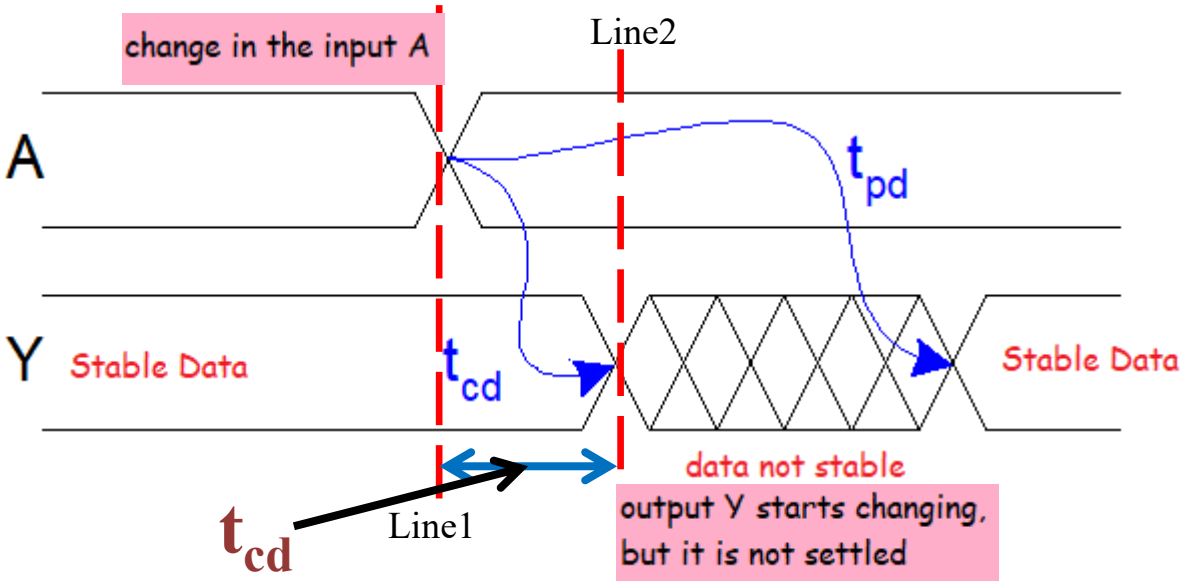
Delays and Timing Constraints of the combinational Logic

- Let the input *A* changing from one arbitrary value to another.



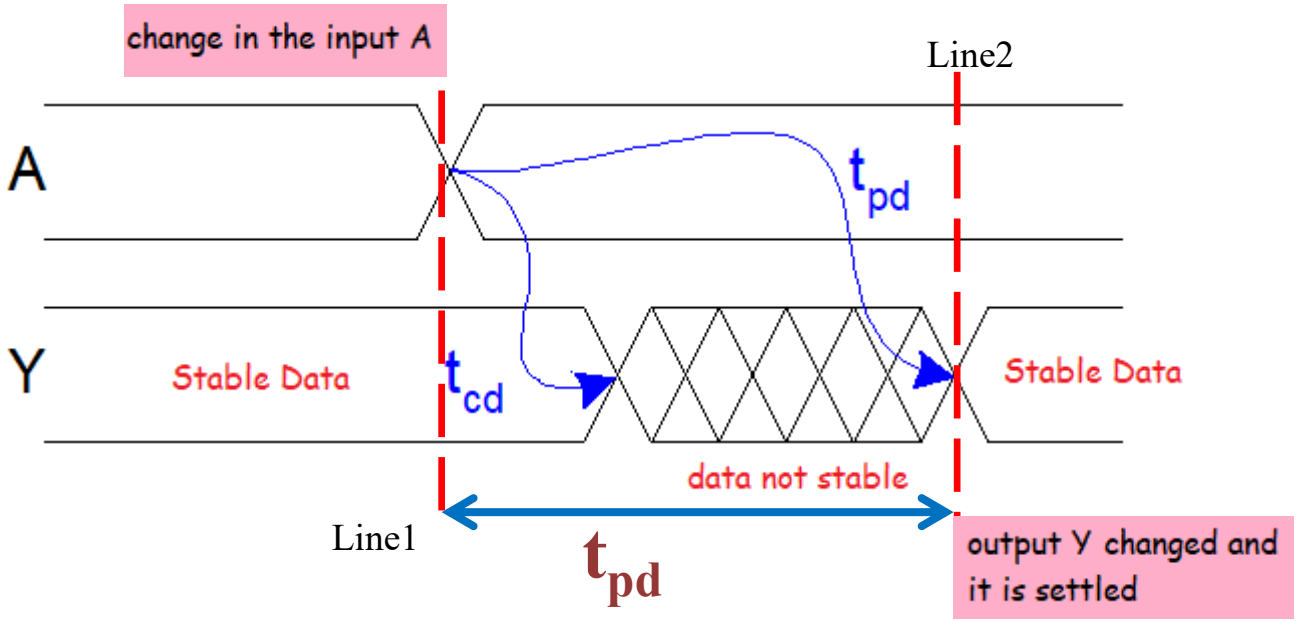


- But the output Y will not respond immediately. There are two different delays associated with comb. Logic
- **Contamination Delay (t_{cd}):** minimum amount of time from when an input changes until any output start to change its value, but the value has not reached the stable condition.
- **Propagation Delay (t_{pd})**: max. amount of time from when an input changes until output change to its final value.



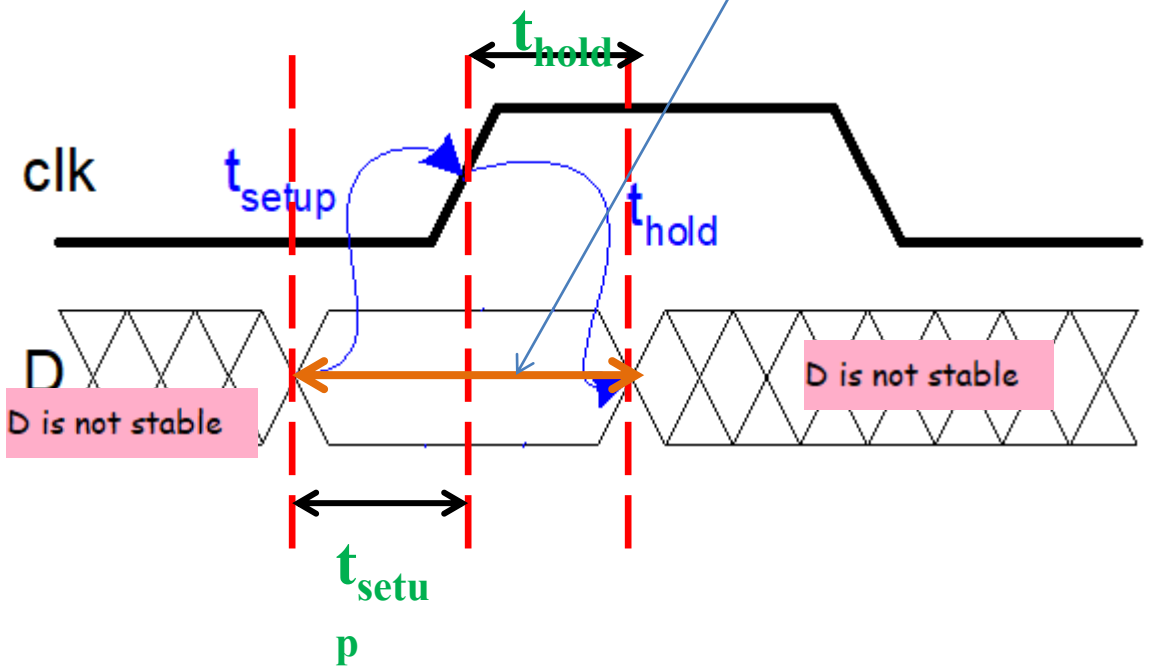
Delays and Timing Constraints of the combinational Logic

- **Propagation Delay (t_{pd})**: max. amount of time from when an input changes until output change to its final value.



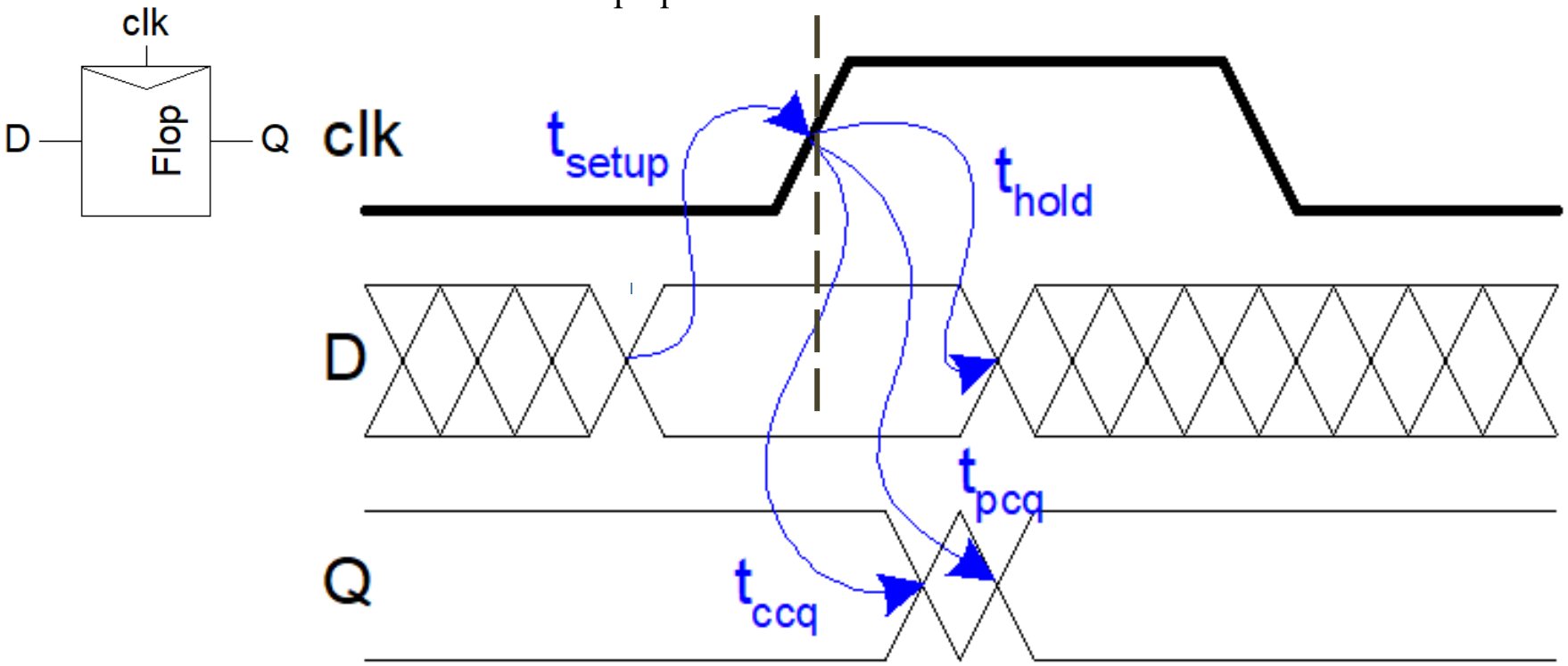
Delays and Timing Constraints of the Flip Flop

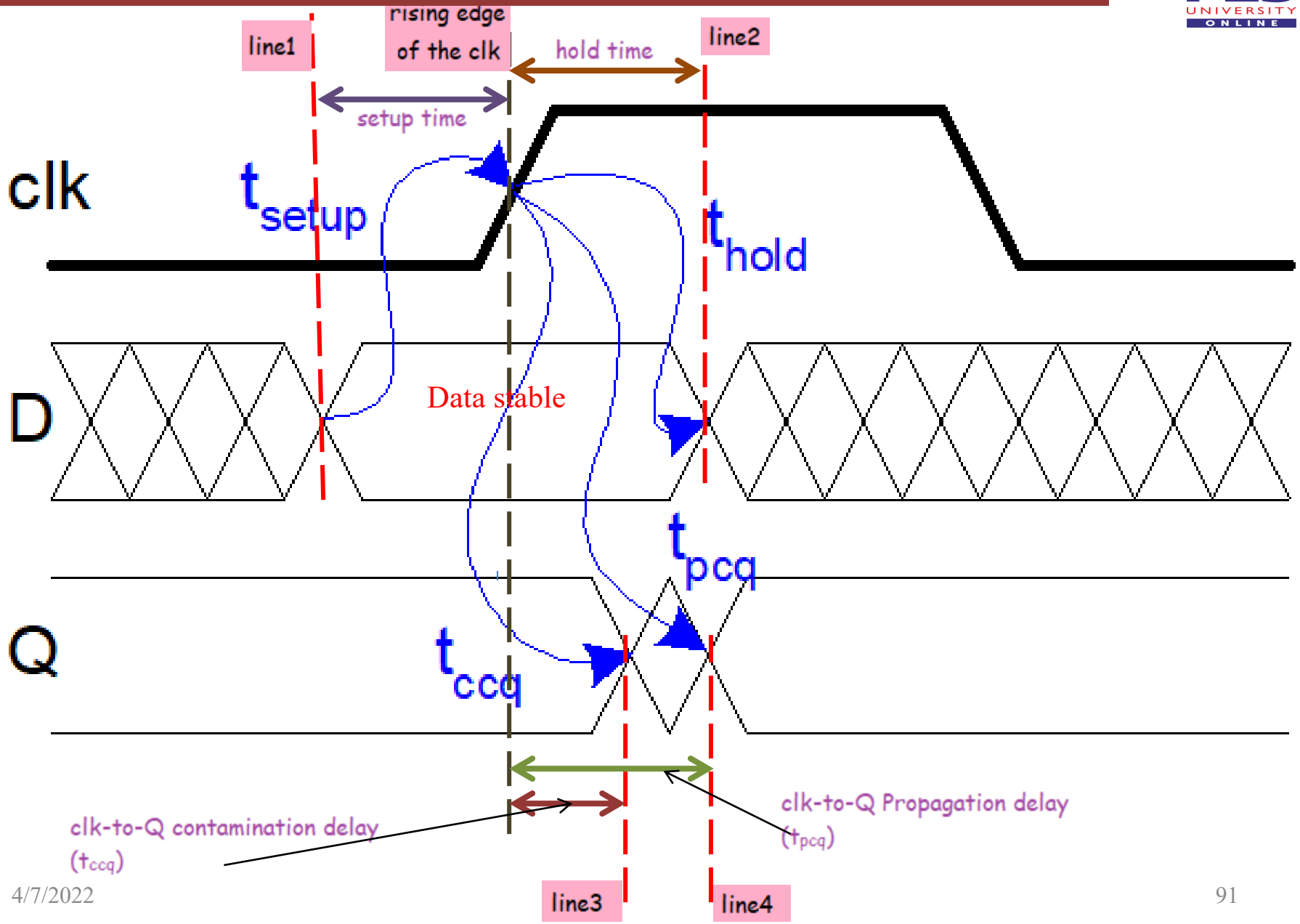
- The data input must be stable for some window around the rising edge of the flip flop if it is to be accurately sampled.
- Thant means, a minimum amount of time where the data should be stable (should not change) before clock goes high is known as **Setup time (t_{setup})** of FF.
- A minimum amount of time where the data should be stable after clock goes high is called as **Hold time (t_{hold})** of FF.



Delays and Timing Constraints of the Flip Flop

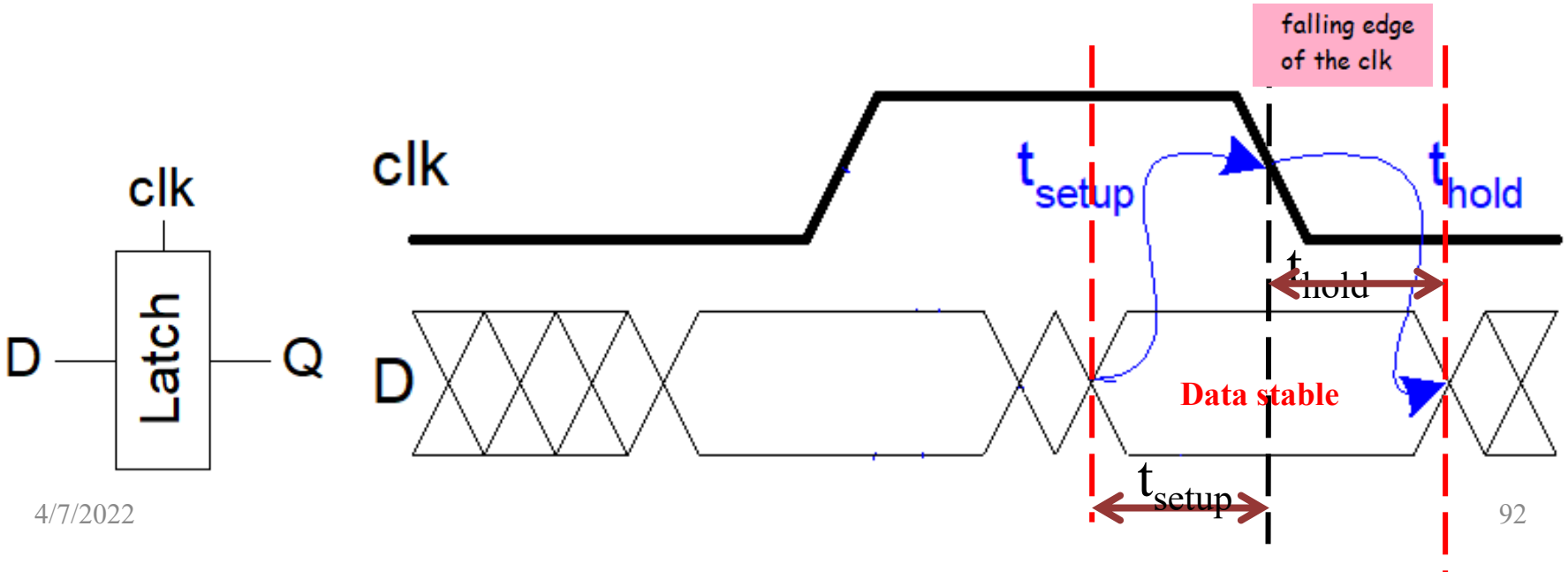
- Since in FF change in output occurs with respect to the rising edge of the clk.
- The output of FF begins to change after clk-to-Q contamination delay (t_{ccq}) and completely settles after a clk-to-Q propagation dealy (t_{pcq}).



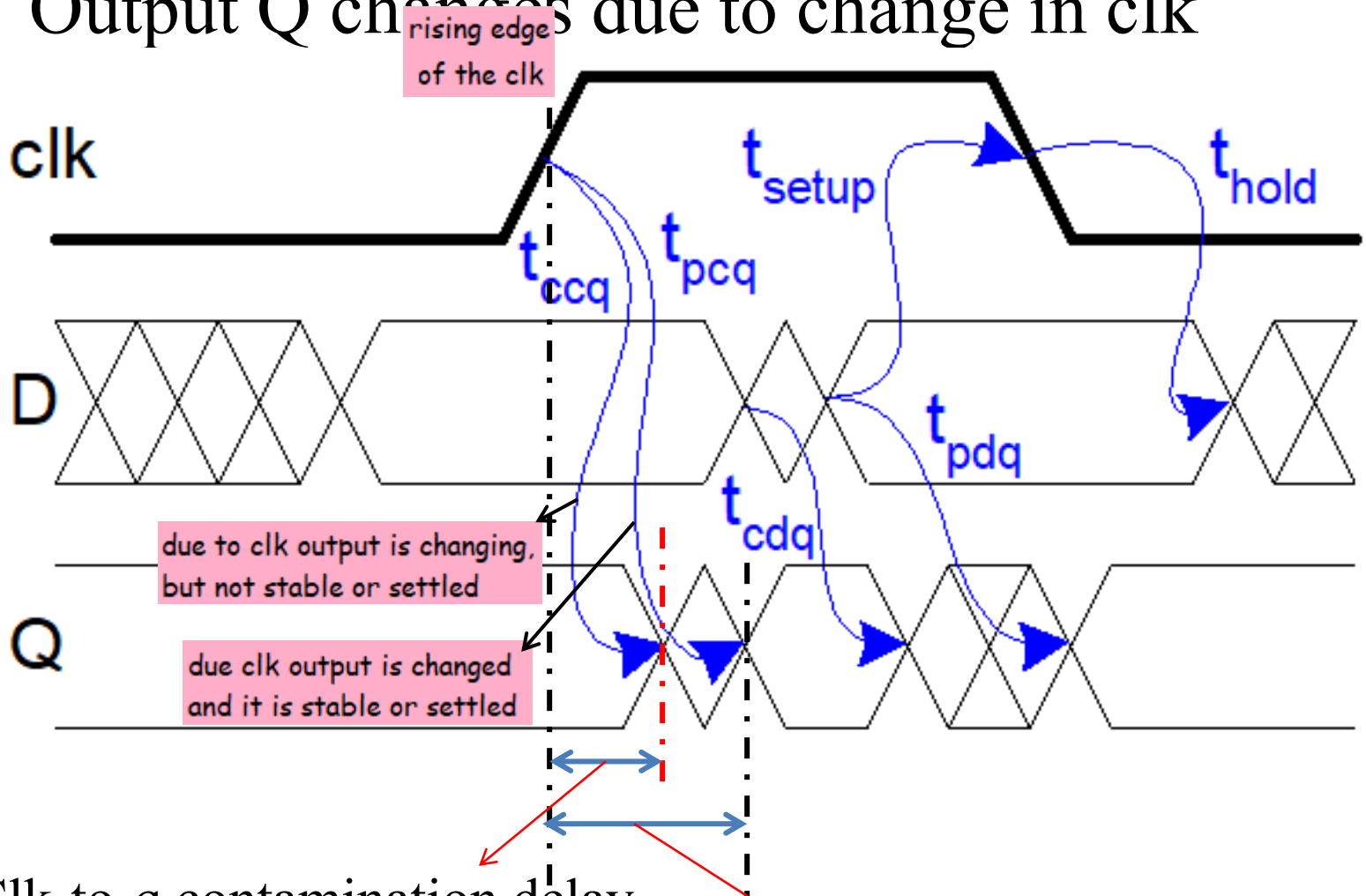

line1
rising edge of the clk
hold time
line2
setup time
clk
t_{setup}
t_{hold}
D
Data stable
t_{pcq}
Q
t_{ccq}
clk-to-Q contamination delay (t_{ccq})
clk-to-Q Propagation delay (t_{pcq})
line3
line4

Delays and Timing Constraints of the Latch

- Since D latch is level sensitive, that is, it is changes in D is reflected at Q when clk =1 and opaque when clk=0.
- Hence input D must setup and hold around the falling edge that defines the end of the sampling period.
- Since output Q changes due to the clk, so it is associated with clk-to-Q contamination and propagation delay.
- And also output Q changes due to change in D when Clk =1, so it is associated with D-to-Q contamination and propagation delay.



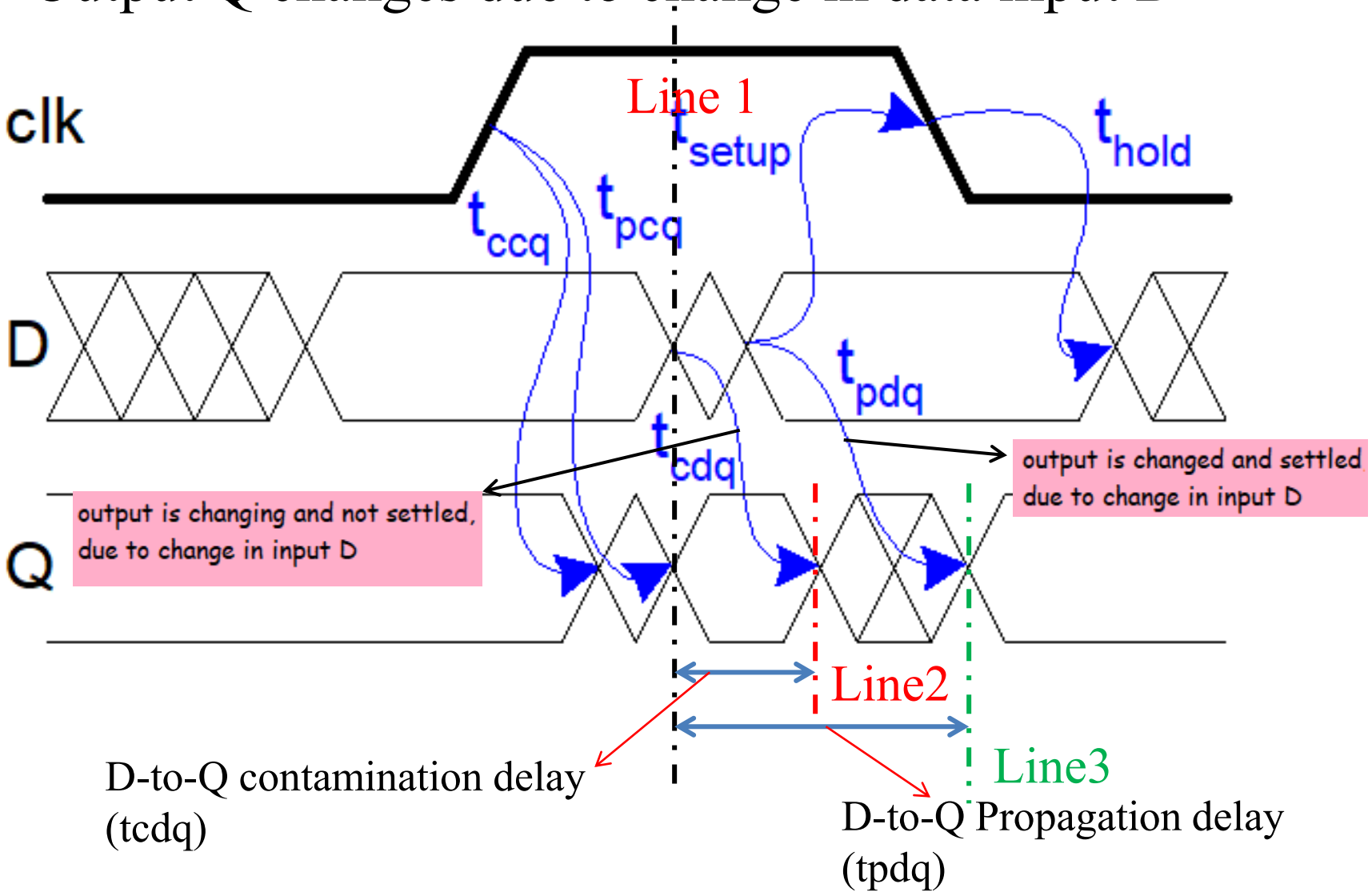
- Output Q changes due to change in clk



Clk-to-q contamination delay (tccq)

Clk-to-q Propagation delay (tpcq)

- Output Q changes due to change in data input D



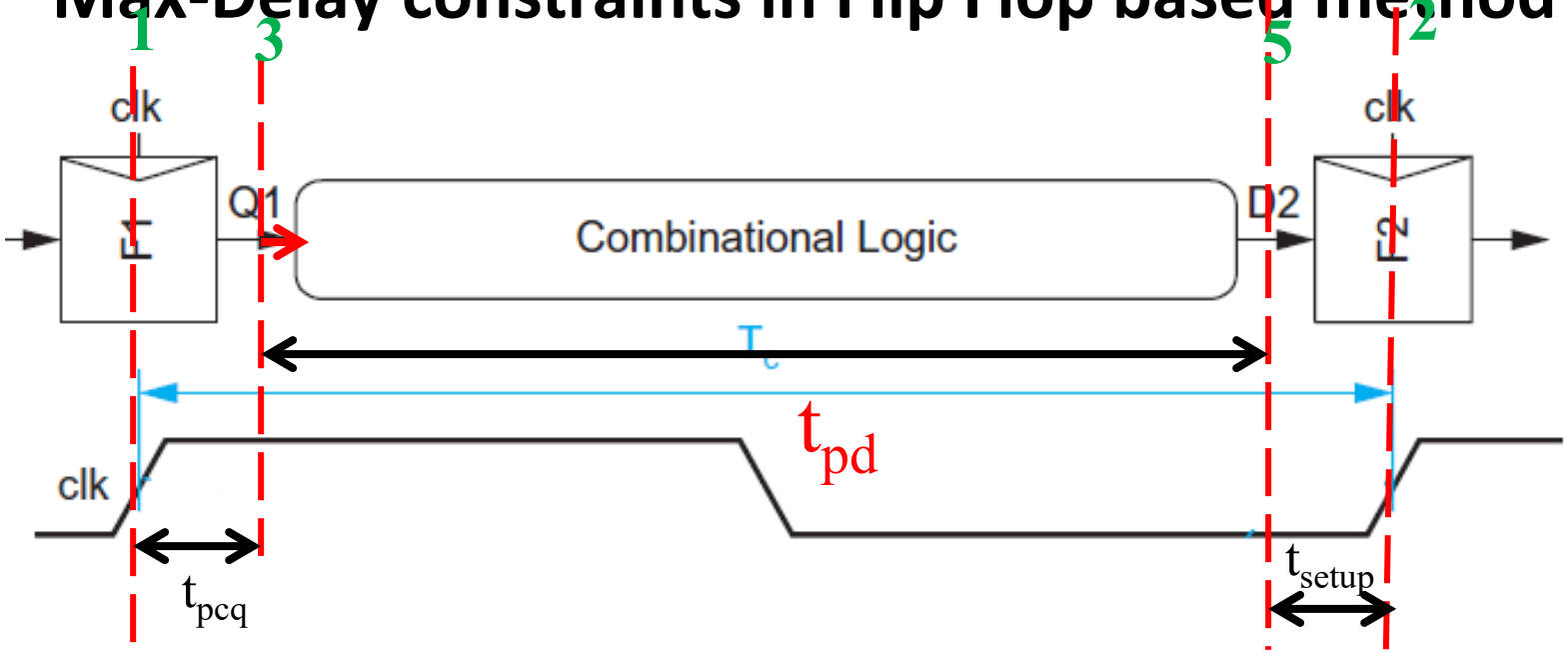
Sequencing element timing notation

Term	Name
t_{pd}	Logic Propagation Delay
t_{cd}	Logic Contamination Delay
t_{pcq}	Latch/Flop Clock-to-Q Propagation Delay
t_{ccq}	Latch/Flop Clock-to-Q Contamination Delay
t_{pdq}	Latch D-to-Q Propagation Delay
t_{cdq}	Latch D-to-Q Contamination Delay
t_{setup}	Latch/Flop Setup Time
t_{hold}	Latch/Flop Hold Time

Max-Delay constraints

- Ideally, the entire clock cycle would be available for computations in the combinational logic
- However, in the real logic, the entire clock cycle is not available for the computations in the combinational logic.
- This is due to the sequencing overheads.
- **Setup time failure or max-delay failure**: if the combinational logic delay is too great, the receiving element will miss its setup time and sample the wrong value. This is called a setup time failure or max-delay failure.

Max-Delay constraints in Flip Flop based method



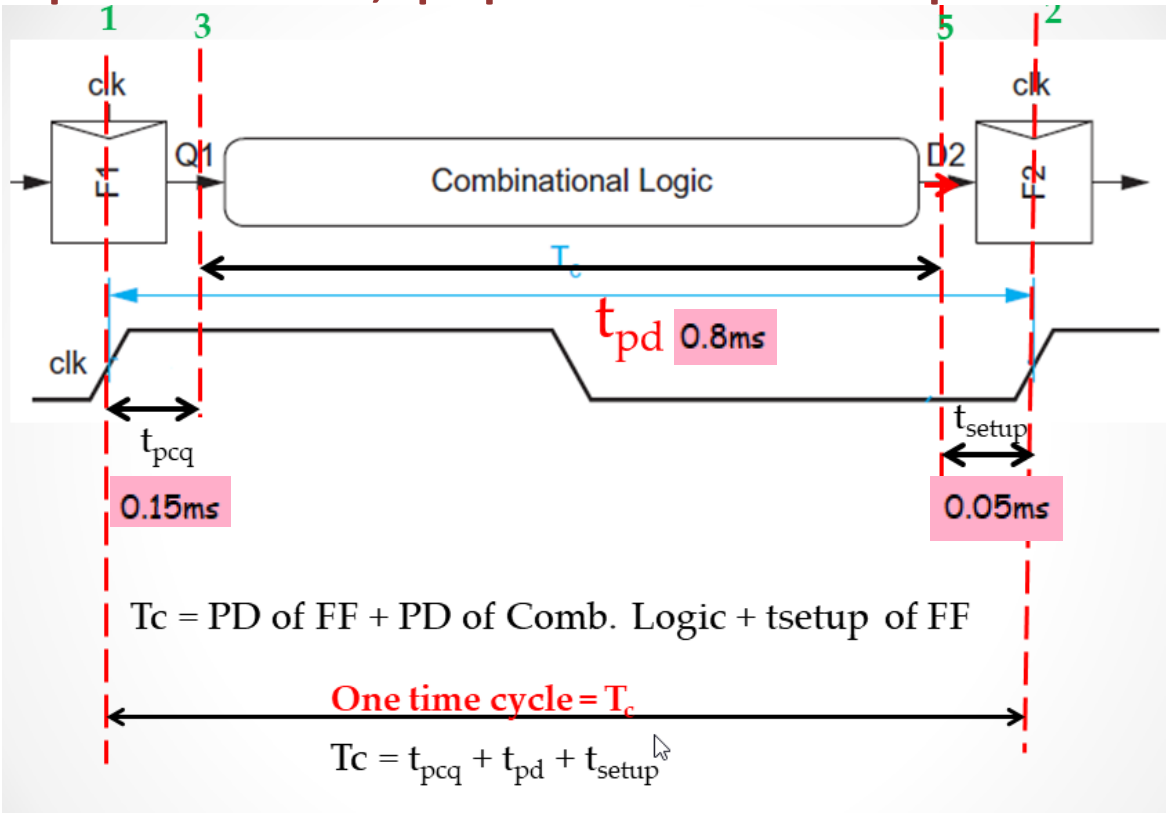
Tc = PD of FF + PD of Comb. Logic + tsetup of FF

One time cycle

~~Tc~~ $T_c = t_{pcq} + t_{pd} + t_{setup}$

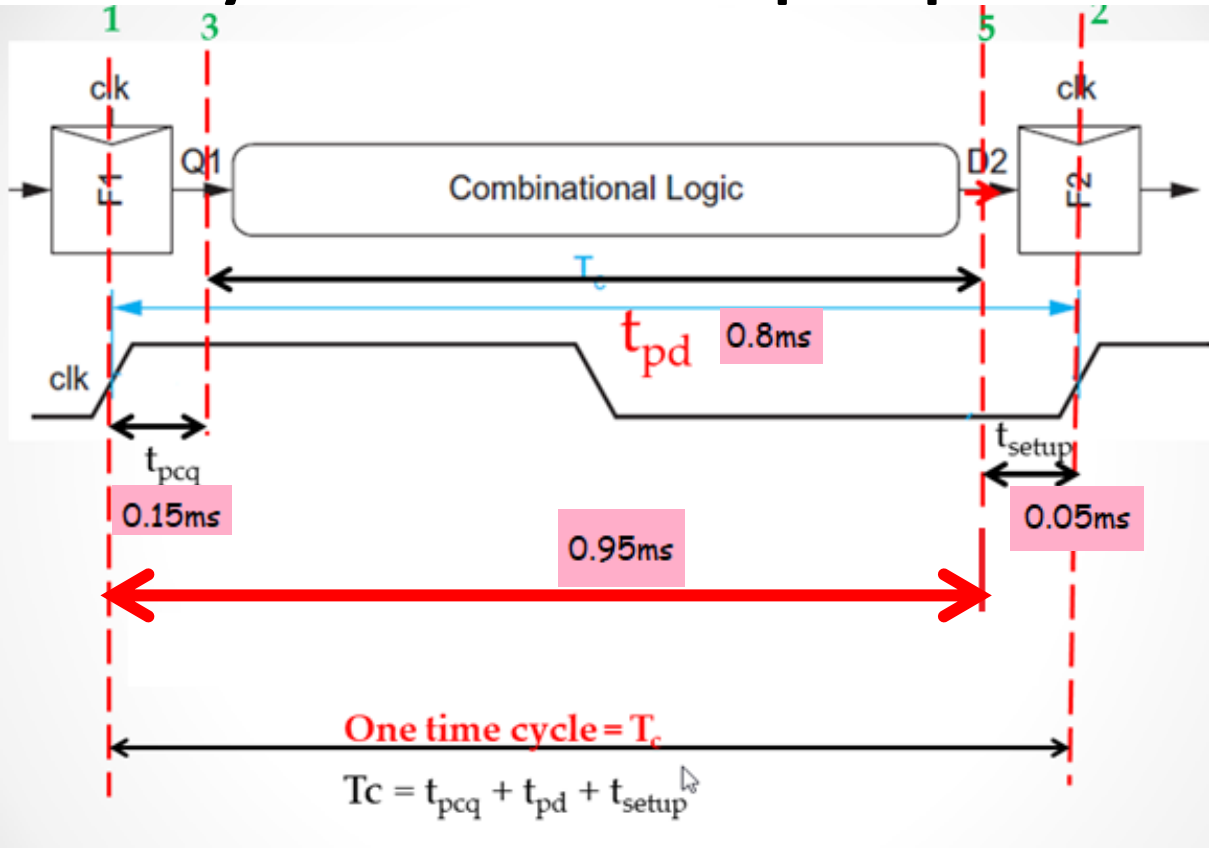
Max-Delay constraints in Flip Flop based method

For example: if Tc = 1ms, tpcq = 0.15ms and tsetup =0.05ms



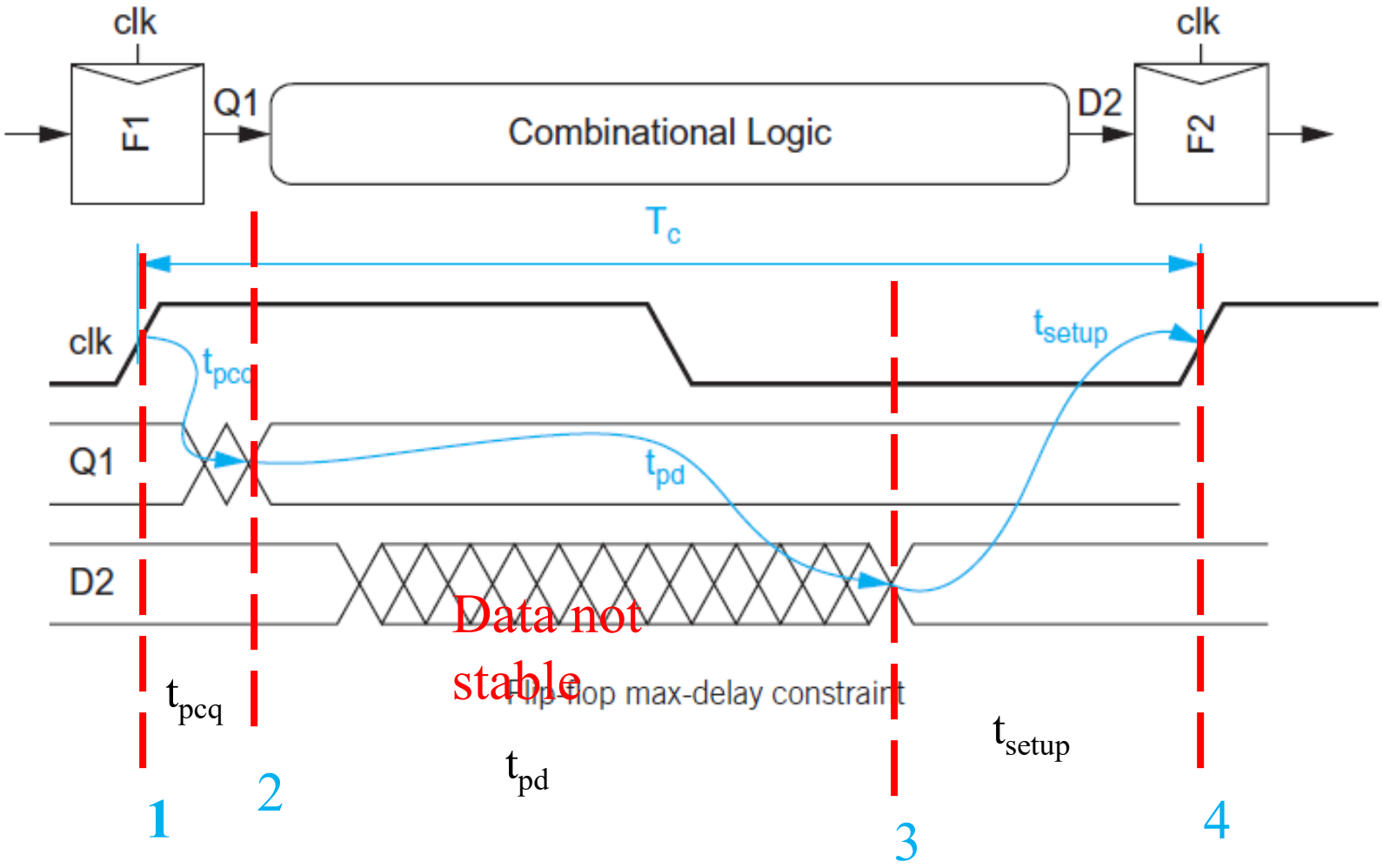
Time available for combinational logic block to compute the entire function is 0.8ms i., tpd = 0.8ms

Max-Delay constraints in Flip Flop based method



Now, output of the comb. Logic is not ready by 0.95ms, the receiving element (F2) will miss its setup time and sample the wrong value. This called setup time failure or max-delay failure.

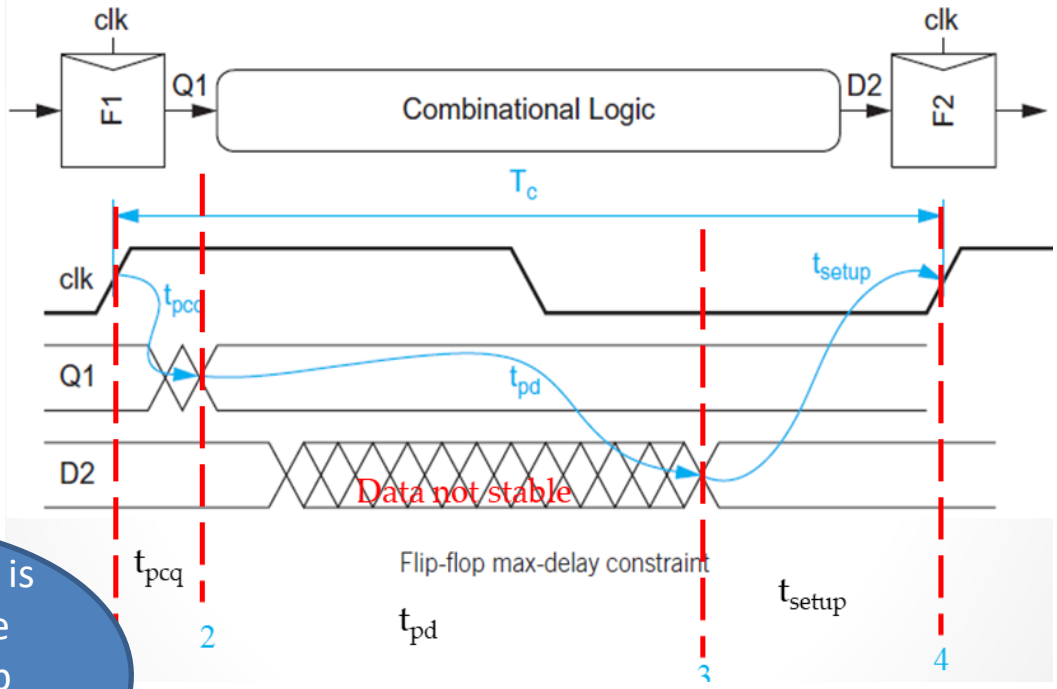
Max-Delay constraints in Flip Flop based method



Max-Delay constraints in Flip Flop based method

- Thus, the clock period must be at least

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$



- Thus, the maximum allowable logic delay is

If this condition is satisfied, there won't be setup time failure or max-delay failure

$$t_{pd} \leq T_c - \underbrace{\left(t_{setup} + t_{pcq}\right)}_{\text{sequencing overhead}}$$

- **Example 10.1**

For the fig 10.6, the propagation delays and contamination delays of the path are given in Table 10.2. Suppose the registers are built from flip-flops with a setup time of 62ps, hold time of –10 ps, propagation delay of 90ps, and contamination delay of 75ps. Calculate the minimum cycle time *Tc* at which the ALU self-bypass path will operate correctly.

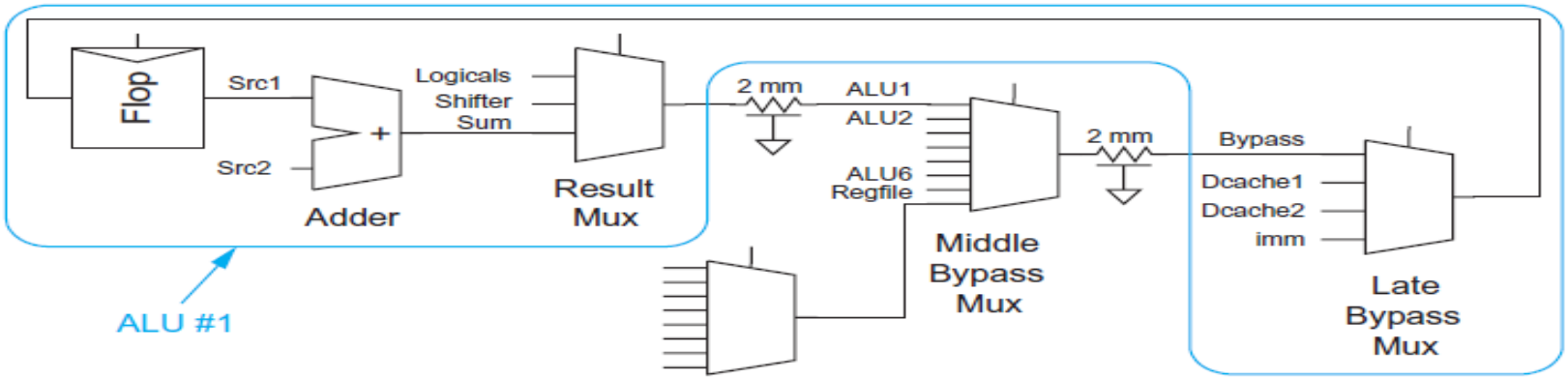
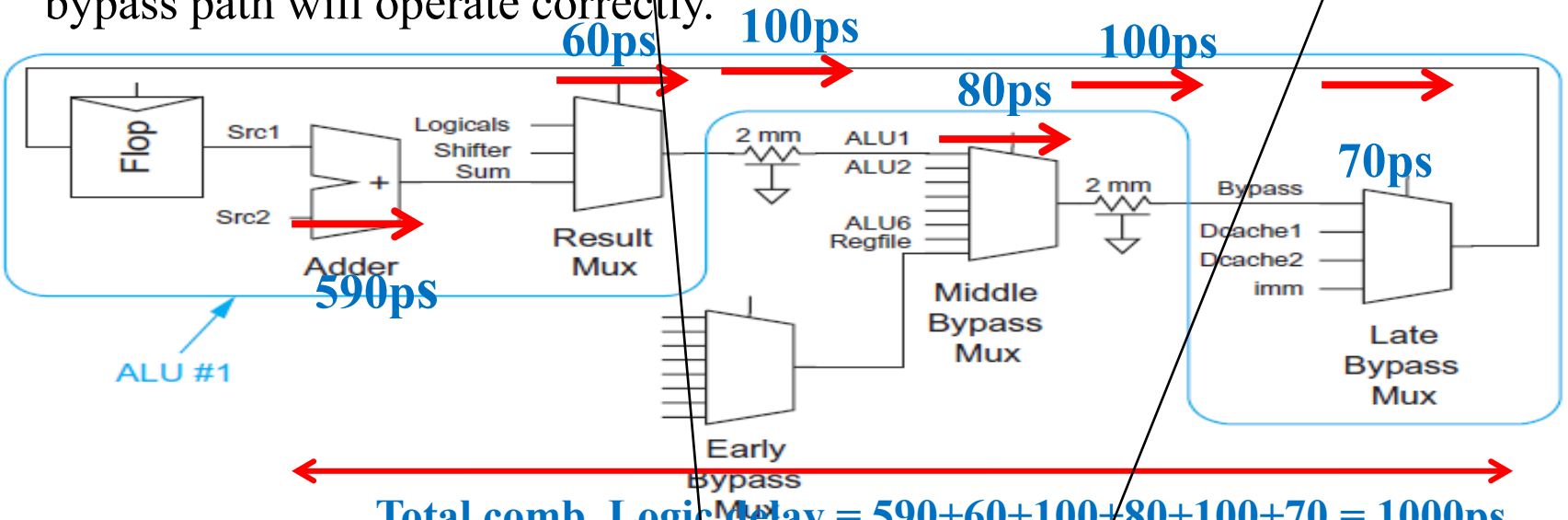


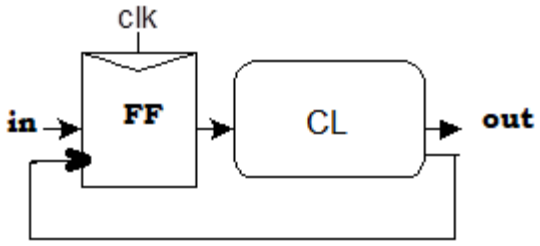
TABLE 10.2 Combinational logic delays

Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps

- **Example 10.1**

For the fig 10.6, the propagation delays and contamination delays of the path are given in Table 10.2. Suppose the registers are built from flip-flops with a setup time of 62ps, hold time of –10 ps, propagation delay of 90ps, and contamination delay of 75ps. Calculate the minimum cycle time *Tc* at which the ALU self-bypass path will operate correctly.



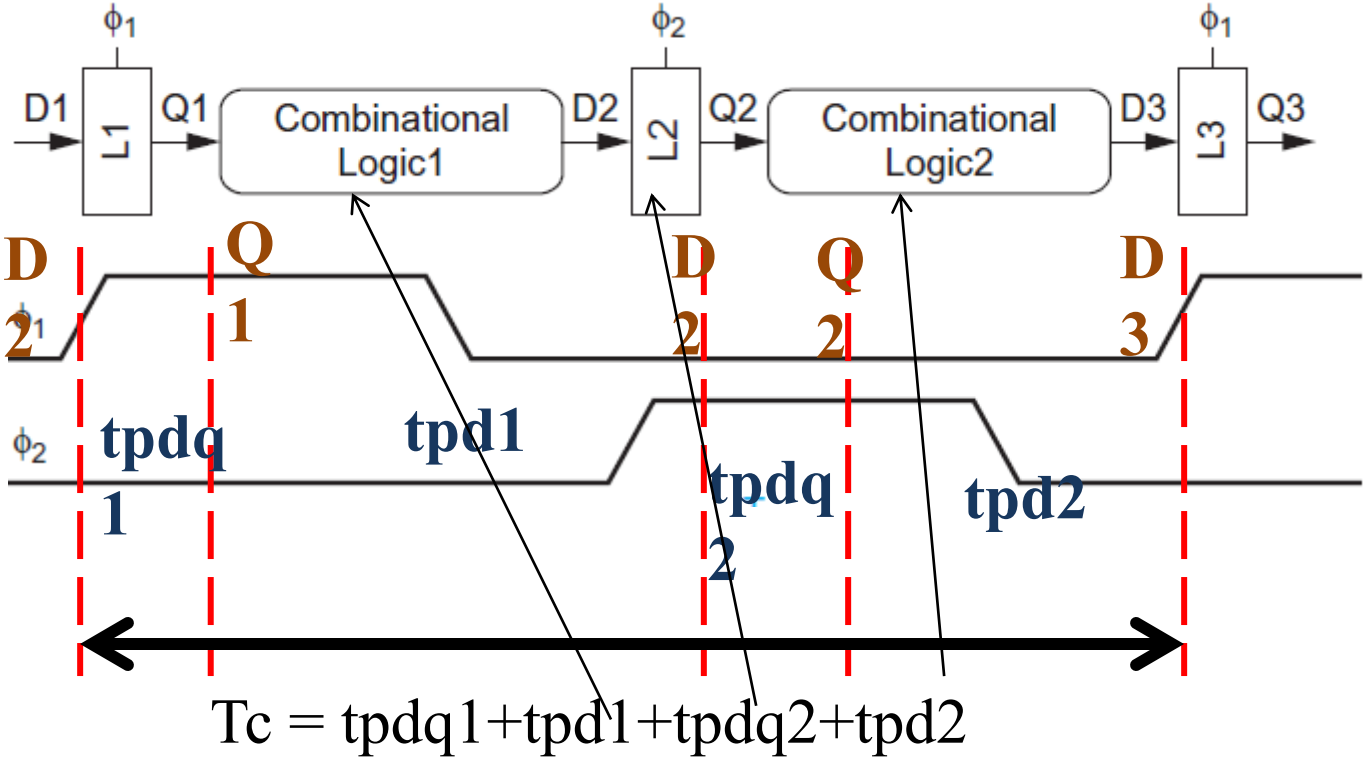


$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

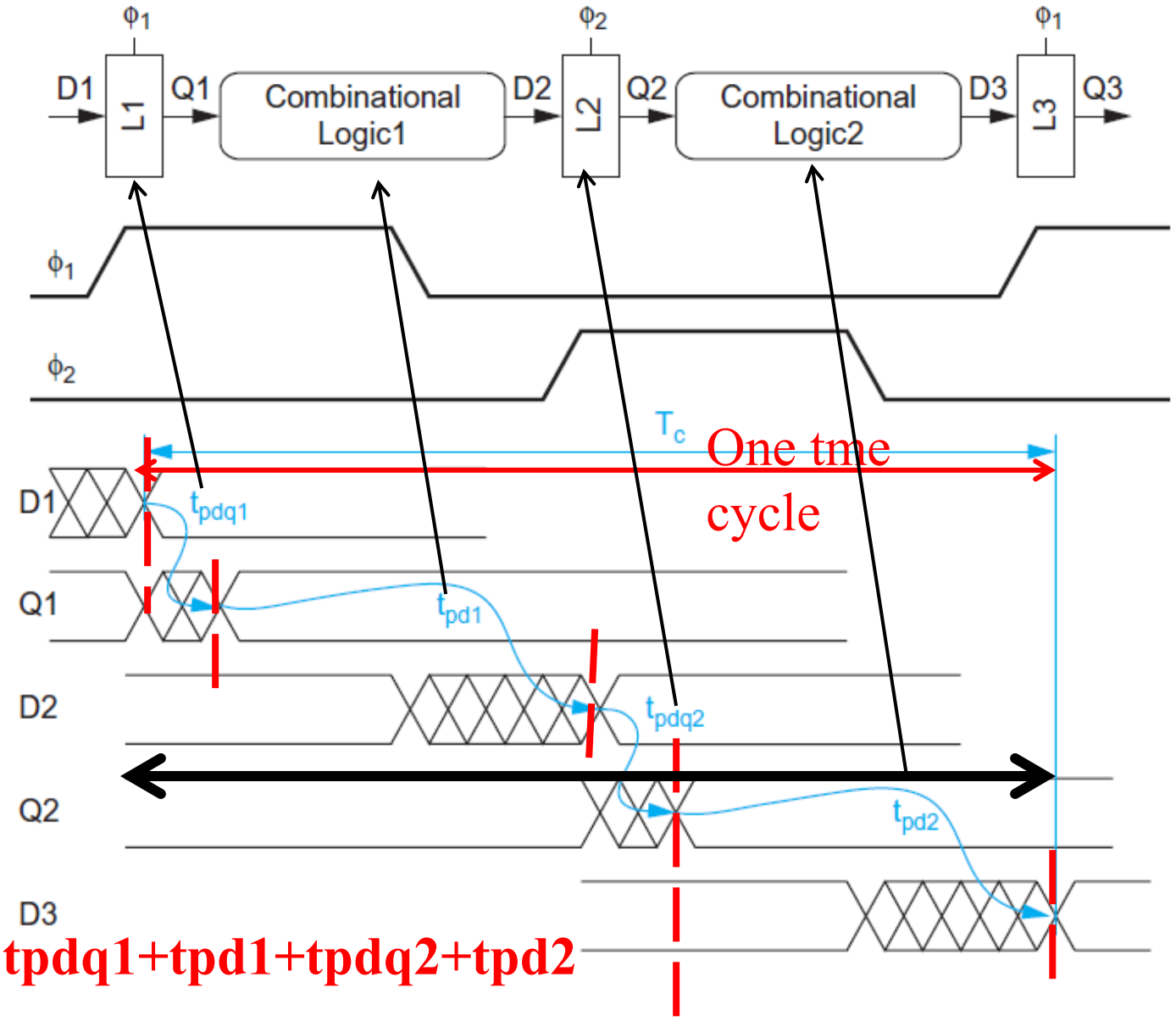
Tc = 90ps+1000ps+62ps = 1152ps

Max-Delay constraints in Two Phase Latch

- Let us assume that data $D1$ arrives at $L1$ while the latch is transparent ($\Phi 1$ high).
- The data propagates through $L1$, the first block of combinational logic, $L2$, and the second block of combinational logic.
- Technically, $D3$ could arrive as late as a setup time before the falling edge of $\Phi 1$ and still be captured correctly by $L3$.



Sequential Circuits Design

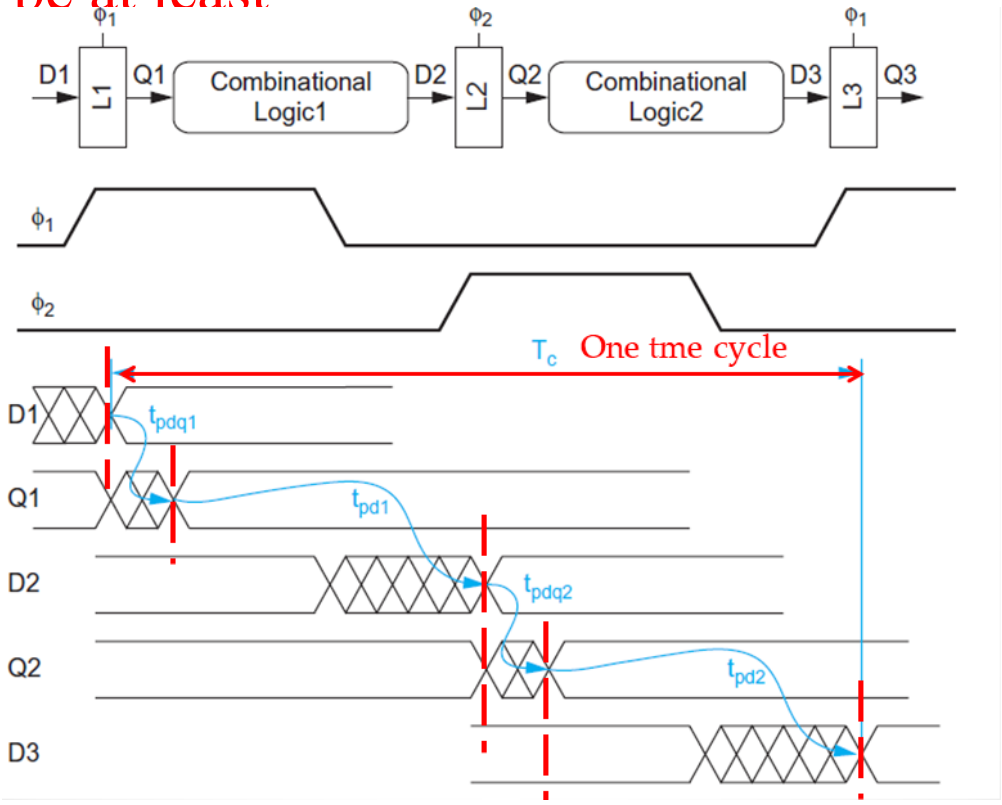


Tc = tpdq1+tpd1+tpdq2+tpd2

Max-Delay constraints in Two Phase Latch

Thus, the clock period must be at least

$$T_c \geq t_{pdq1} + t_{pd1} + t_{pdq2} + t_{pd2}$$

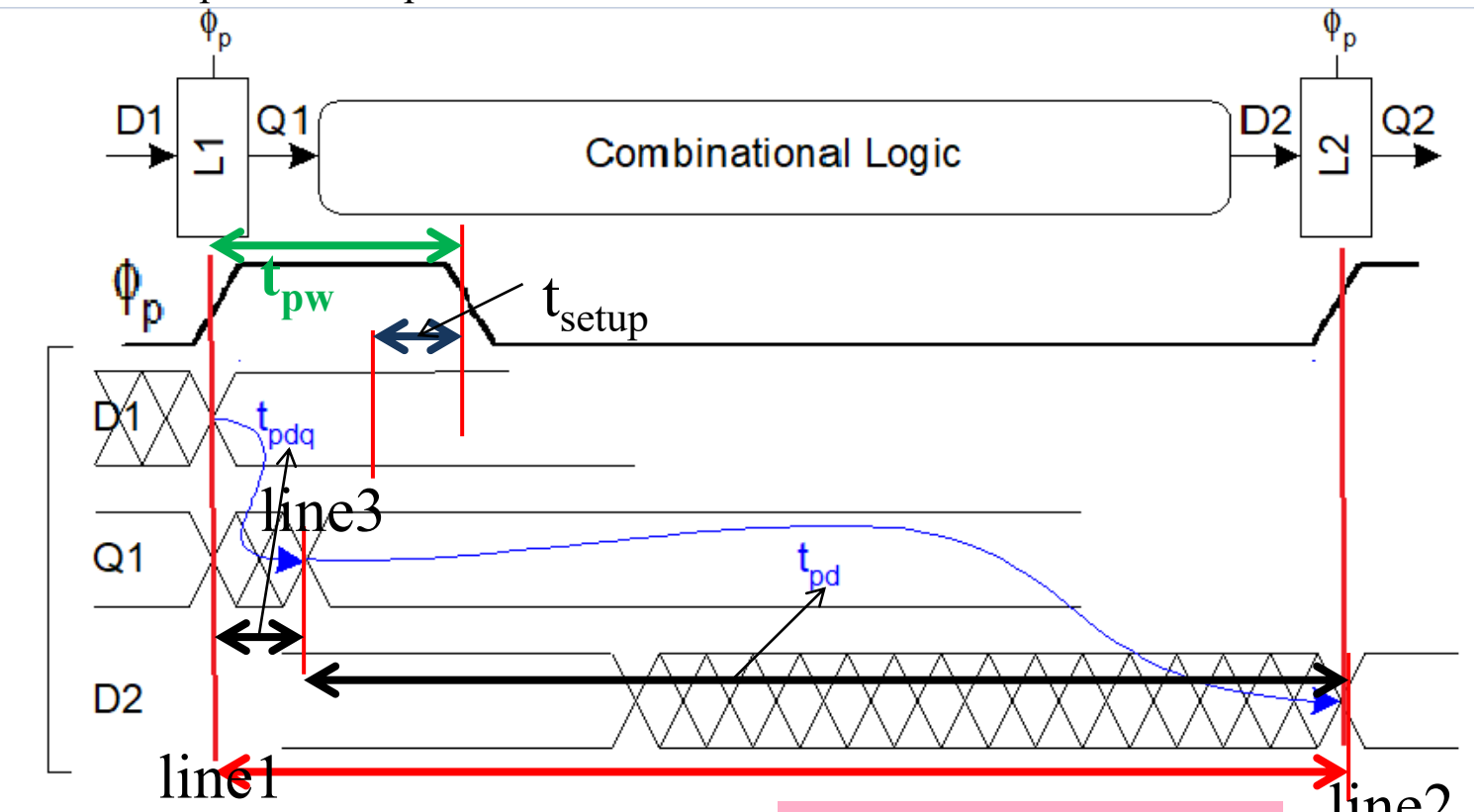


Thus, the maximum allowable logic delay

$$t_{pd} = t_{pd1} + t_{pd2} \leq T_c - \underbrace{\left(2t_{pdq}\right)}_{\text{sequencing overhead}}$$

Max-Delay constraints in Phased Latch

Case 1: $T_{pw} > t_{setup}$



the clock period must be at least

Tc ≥ tpdq + tpd

the maximum allowable logic delay

Tpd ≤ Tc - tpdq

Max-Delay constraints in Phased Latch

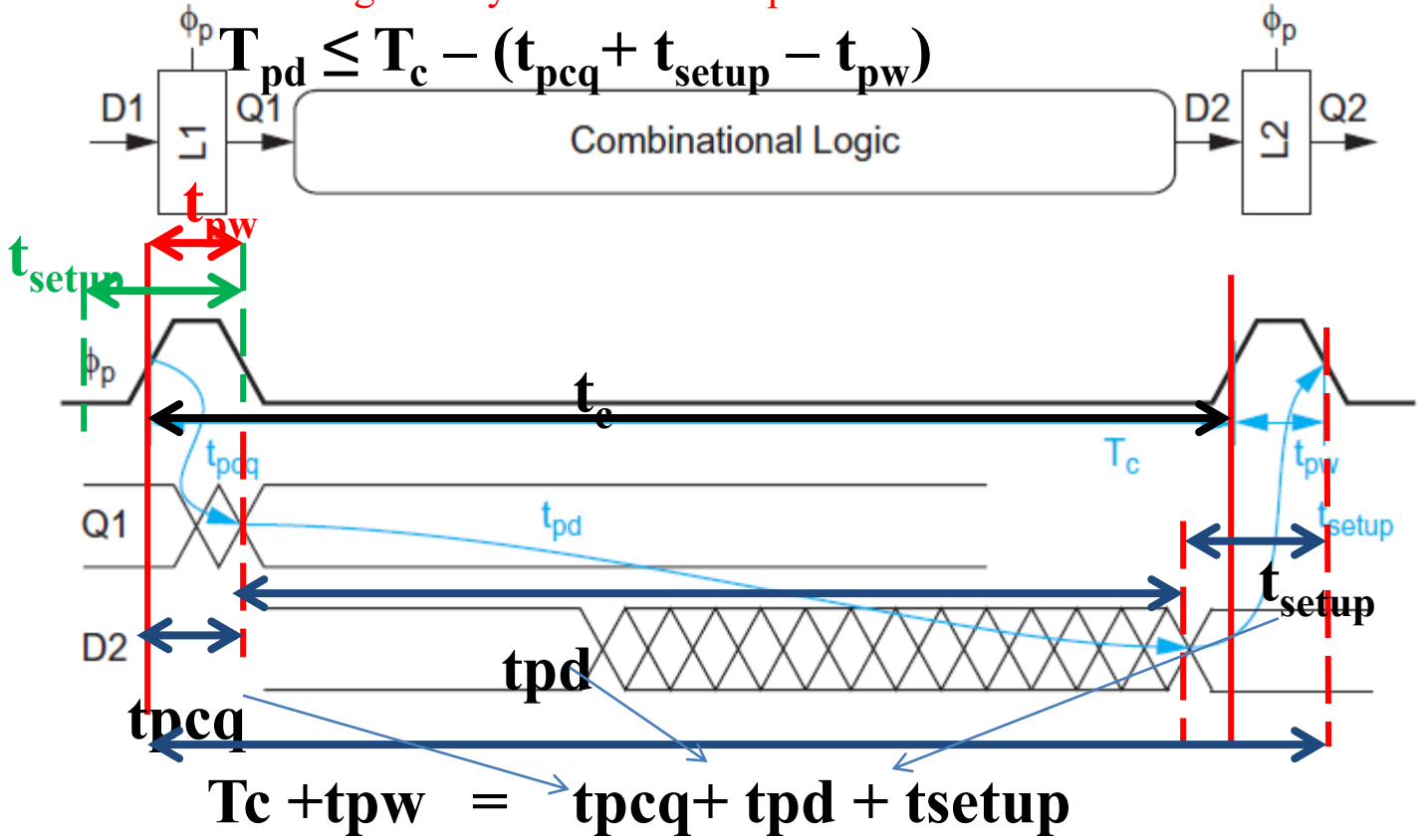
Case 2: $T_{pw} < t_{setup}$

Thus, the clock period must be at least

$$Tc \geq t_{pcq} + t_{pd} + t_{setup} - t_{pw}$$

Thus, the maximum allowable logic delay is without setup time failure

$$T_{pd} \leq T_c - (t_{pcq} + t_{setup} - t_{pw})$$



$$Tc + tpw = tpcq + tpd + tsetup$$

Problems on Max-Delay

- **Example 10.2**: Recompute the ALU self-bypass path cycle time if the flip-flop is replaced with a pulsed latch. The pulsed latch has a pulse width of 150 ps, a setup time of 40 ps, a hold time of 5 ps, a *clk*-to-*Q* propagation delay of 82 ps and contamination delay of 52 ps, and a *D*-to-*Q* propagation delay of 92 ps.

Solution: The propagation delay of the combinational block is still 1000ps (same as example 10.1)

Since pulsed width of the latch is greater than setup time,

so According to equation

$$T_c \geq t_{pdq} + t_{pd}$$

The cycle time must be at least 92 + 1000 = 1092 ps.

Max-Delay constraints

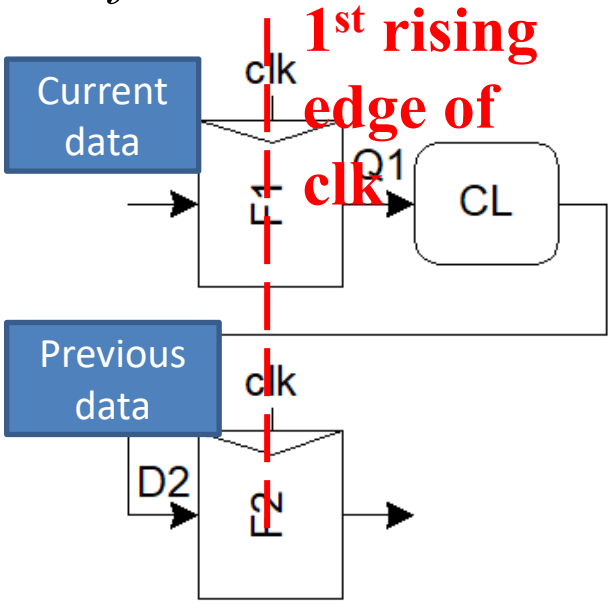
- To avoid Max delay constraints or setup time failure in the design

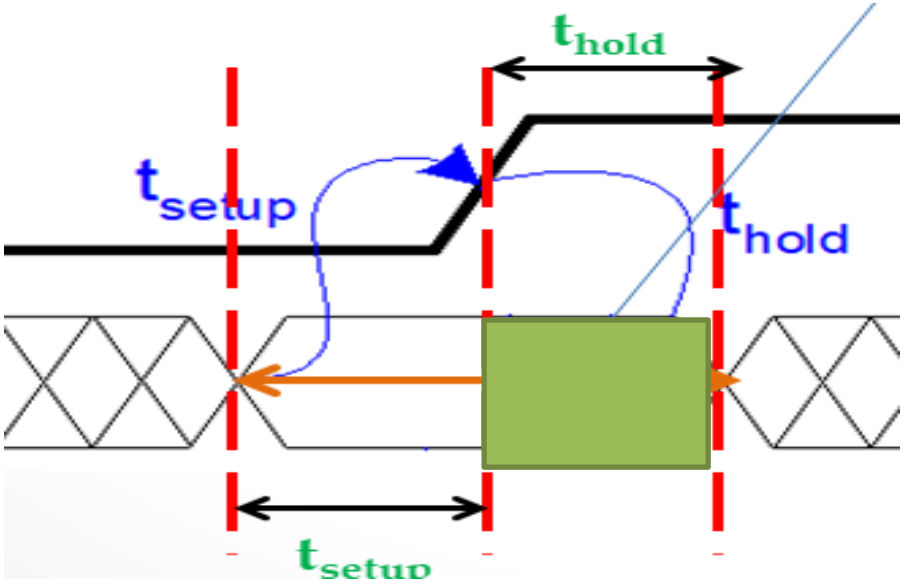
➢ **Increase the time cycle**

➢ **Decrease the propagation delay of the combinational logic.**

Min-Delay Constraints

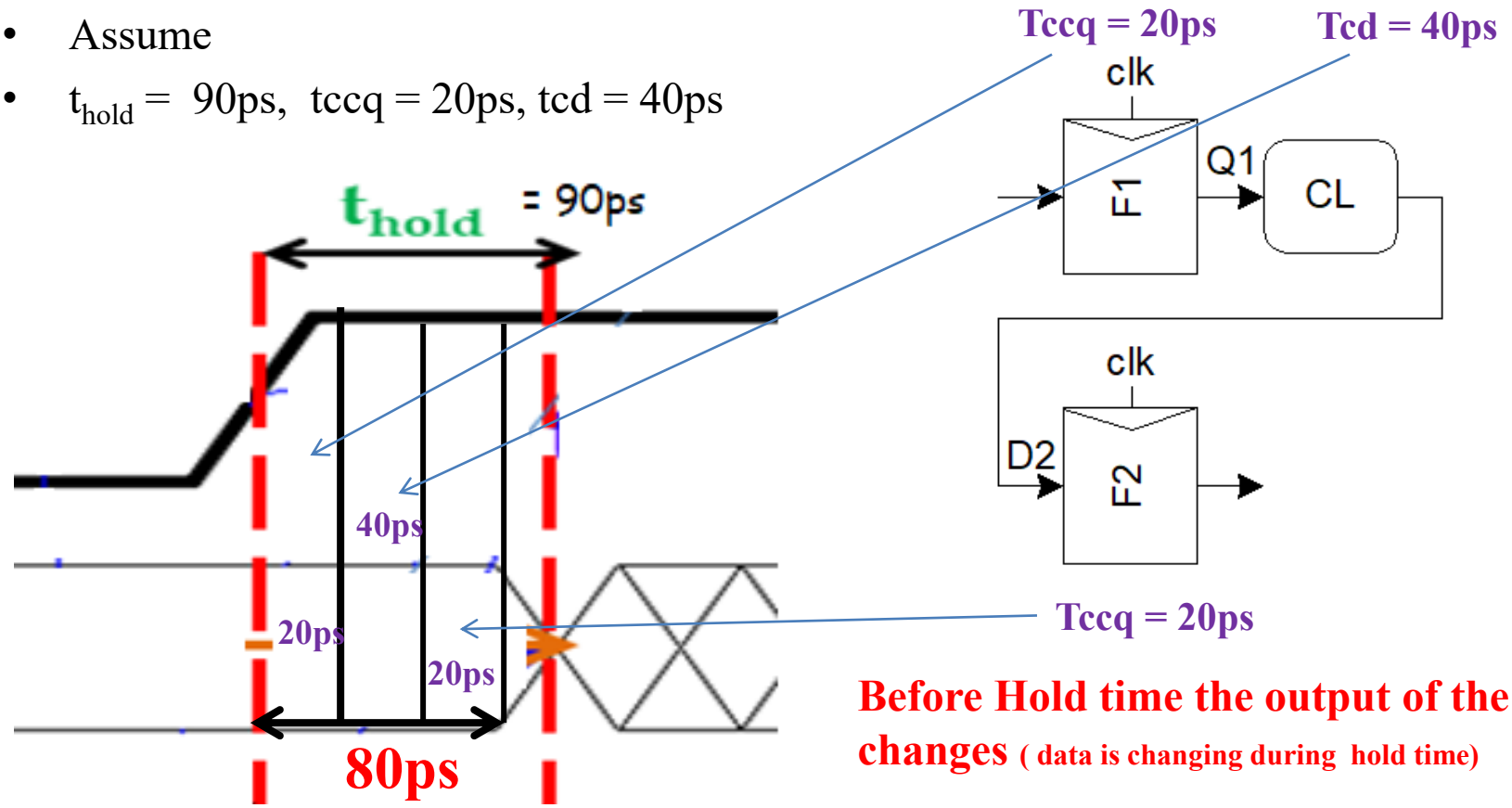
- If the hold time is large and the contamination delay is small, data can incorrectly propagate through two successive elements on one clock edge, corrupting the state of the system. This is called a *race condition*, *hold-time failure*, or *min-delay failure*.





Min-Delay Constraint in Flip Flop

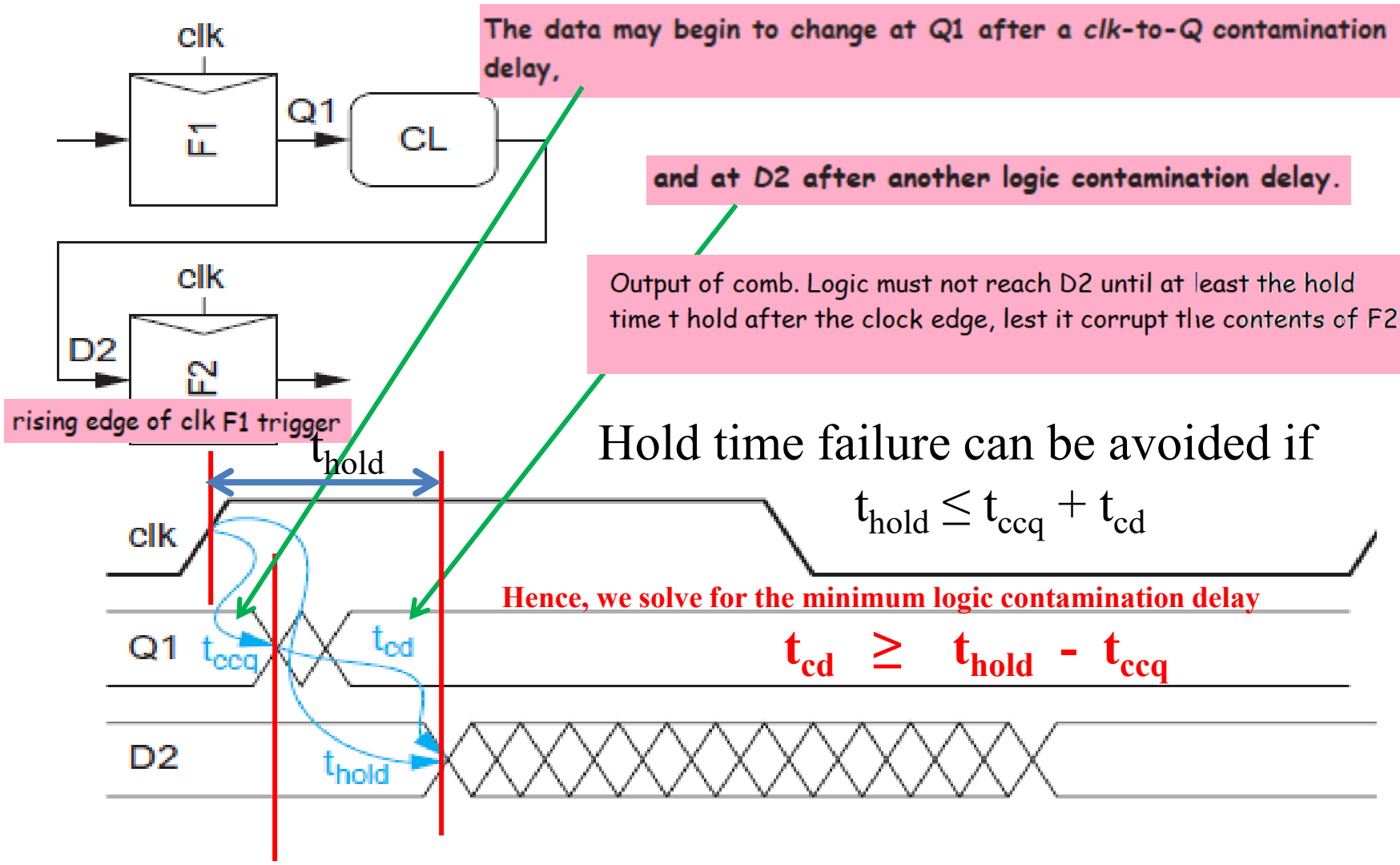
- If the hold time is large and the contamination delay is small, data can incorrectly propagate through two successive elements on one clock edge, corrupting the state of the system. This is called a *race condition*, *hold-time failure*, or *min-delay failure*.
- Assume
- t_{hold} = 90ps, tccq = 20ps, tcd = 40ps



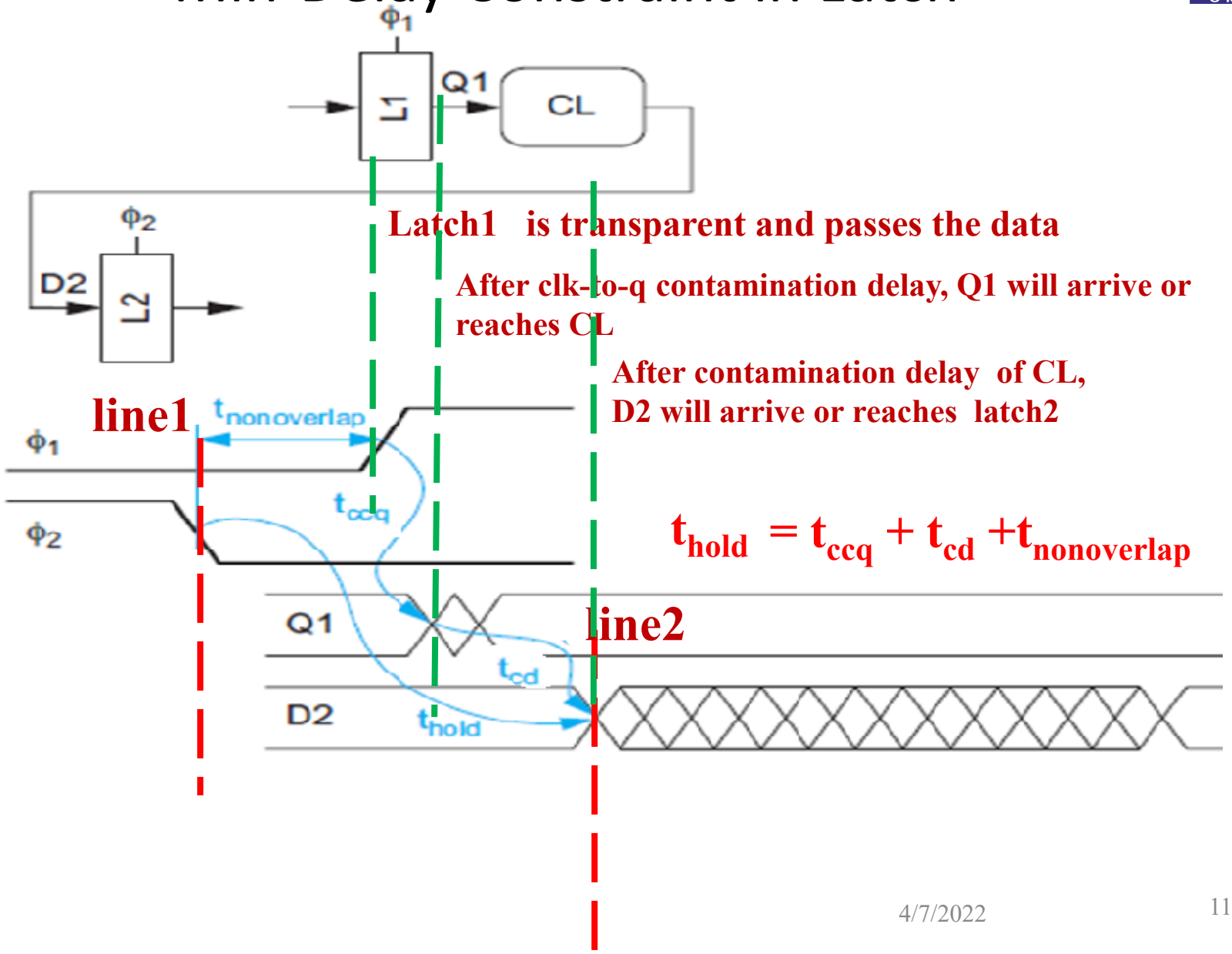
Before Hold time the output of the FF changes (data is changing during hold time)

Min-Delay Constraint in Flip flop

- The path begins with the rising edge of the clock triggering *F*1.



Min-Delay Constraint in Latch

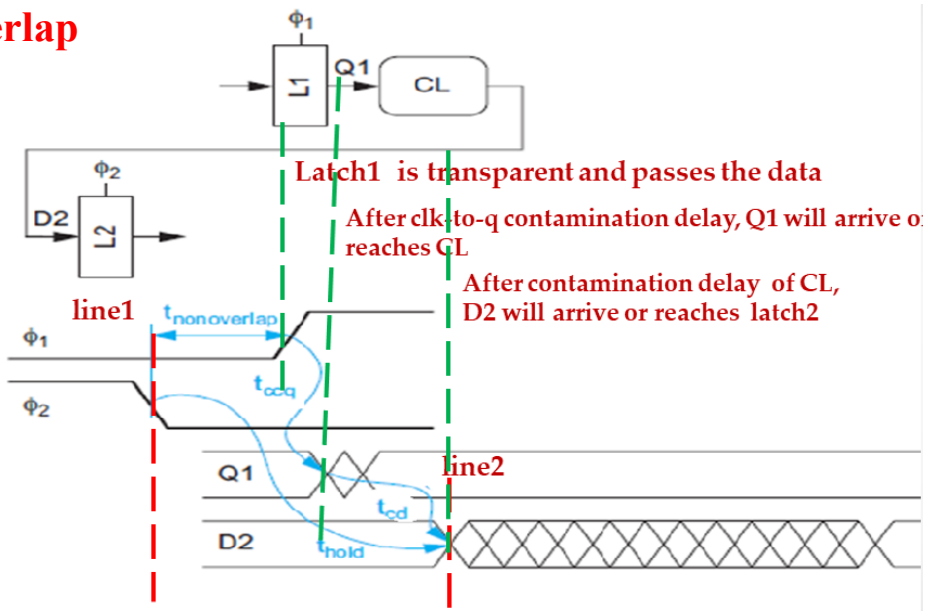


Min-Delay Constraint in Latch

- The path begins with data passing through L1 on the rising edge of Φ1.
- It (Q1) must not reach L2 until a hold time after the previous falling edge of Φ2 because L2 should have become safely opaque before L1 becomes transparent .
- As the edges (falling edge of Φ2 and rising edge of Φ1) separated by $t_{nonoverlap}$, the minimum logic contamination delay through each phase of logic is

$$t_{cd1}, t_{cd2} \geq t_{hold} - t_{ccq} - t_{nonoverlap}$$

$$t_{hold} \leq t_{ccq} + t_{cd} + t_{nonoverlap}$$



Min-Delay Constraint in Pulsed Latch

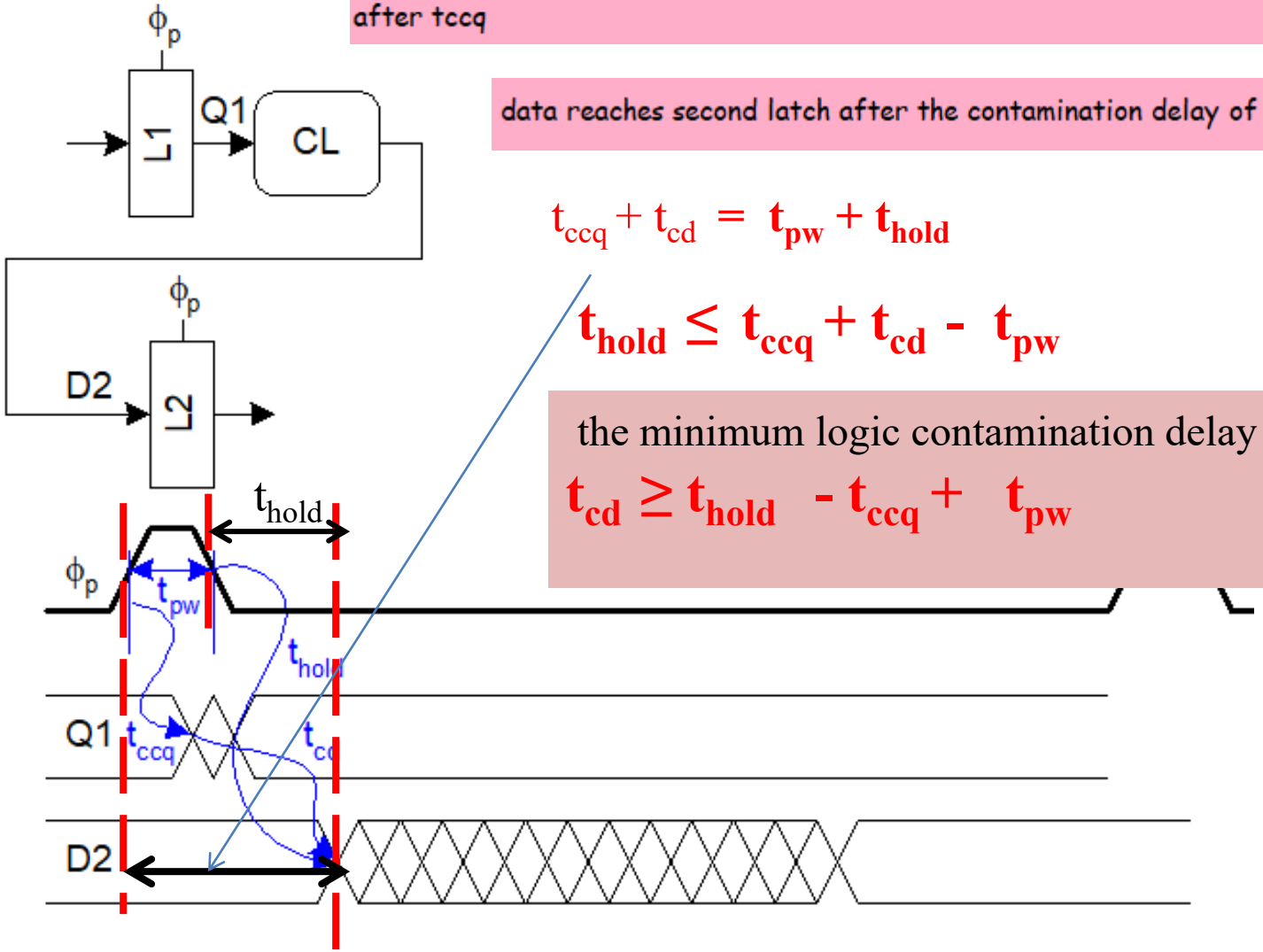
- Case 1: $t_{pw} > t_{setup}$ is similar to latch operation
- Case 2: $t_{pw} < t_{setup}$

Min-Delay Constraint in

- Case 2: $t_{pw} < t_{setup}$

during the rising edge of the pulse latch becomes transparent and data reach CL after tccq

data reaches second latch after the contamination delay of CL (tcd)



$$t_{ccq} + t_{cd} = t_{pw} + t_{hold}$$

$$t_{hold} \leq t_{ccq} + t_{cd} - t_{pw}$$

the minimum logic contamination delay is

$$t_{cd} \geq t_{hold} - t_{ccq} + t_{pw}$$

Min-Delay constraints

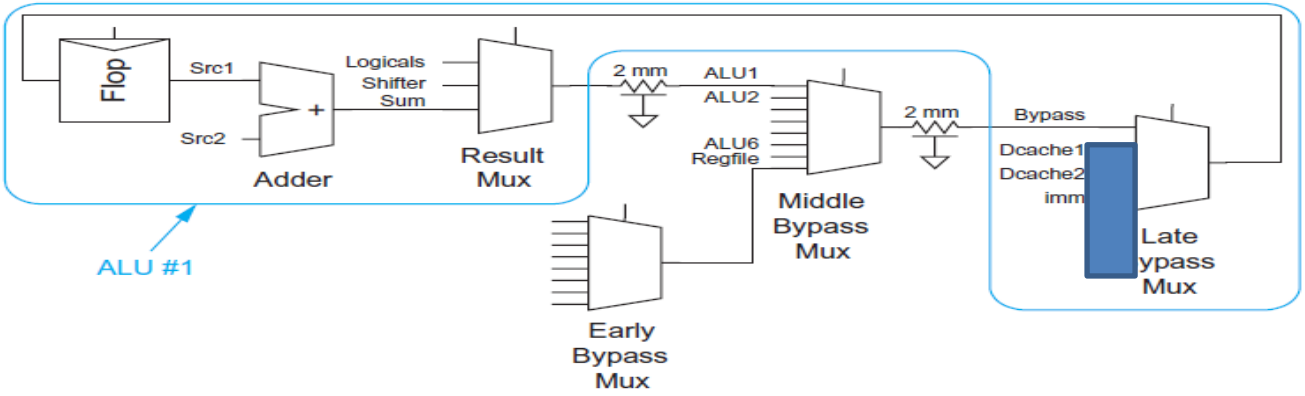
- To avoid Min delay constraints or hold time failure in the design

➢ **Increase the contamination delay of the combinational logic.**

➢ **That is hold time must be less than the contamination delay**

Problems on Min-Delay

- **Example 10.3**: In the ALU self-bypass example with flip-flops from Figure 10.6, the earliest input to the late bypass multiplexer is the *imm* value coming from another flip-flop. Will this path experience any hold-time failures?



flip-flops with a setup time of 62ps, hold time of –10 ps, delay of 90ps, and contamination delay of 75ps

TABLE 10.2 Combinational logic delays

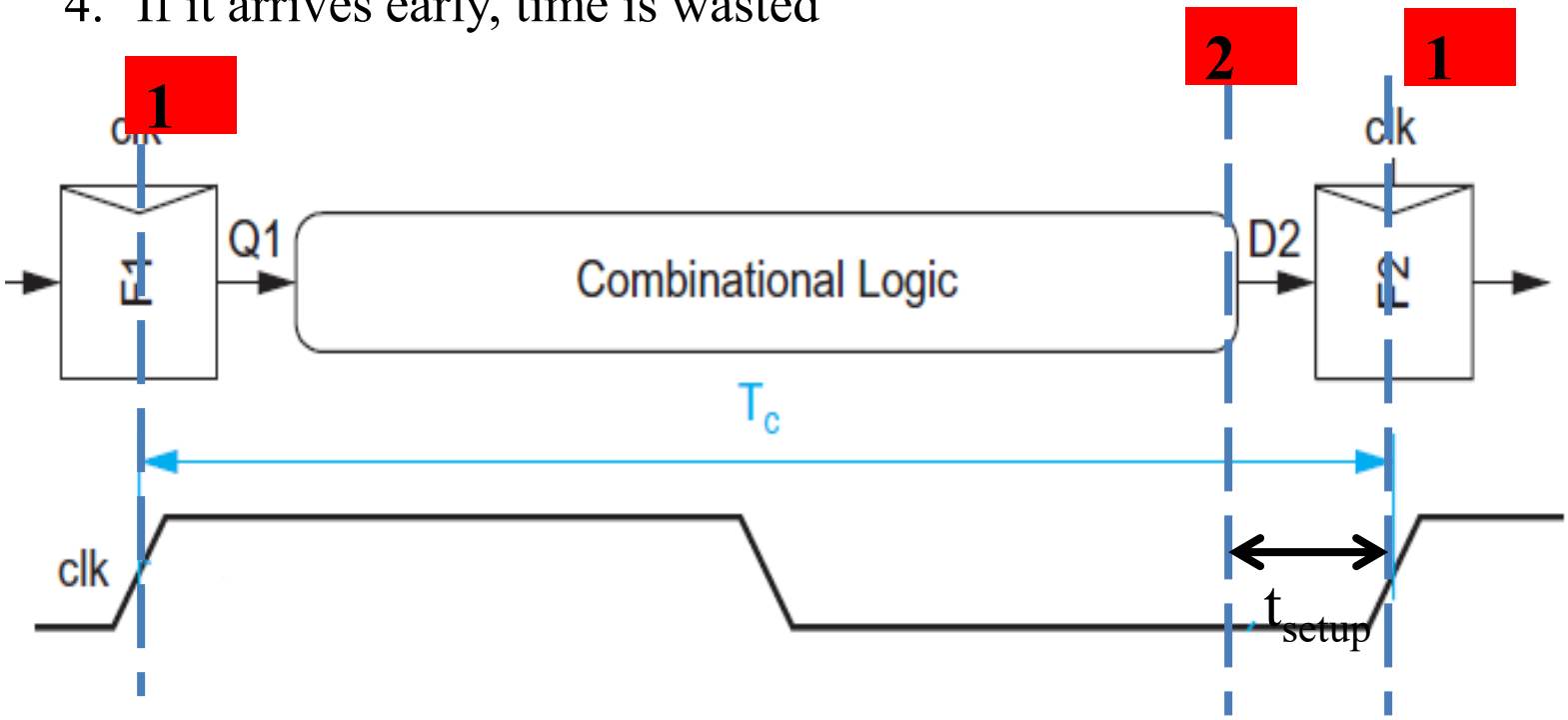
Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps

The late bypass mux has the contamination delay, *tcd* = 45 ps. The flip-flops have *t*hold = –10 ps and *tccq* = 75 ps.
Hence, equation $t_{cd} \geq t_{\text{hold}} - t_{ccq}$

is easily satisfied

Time Borrowing

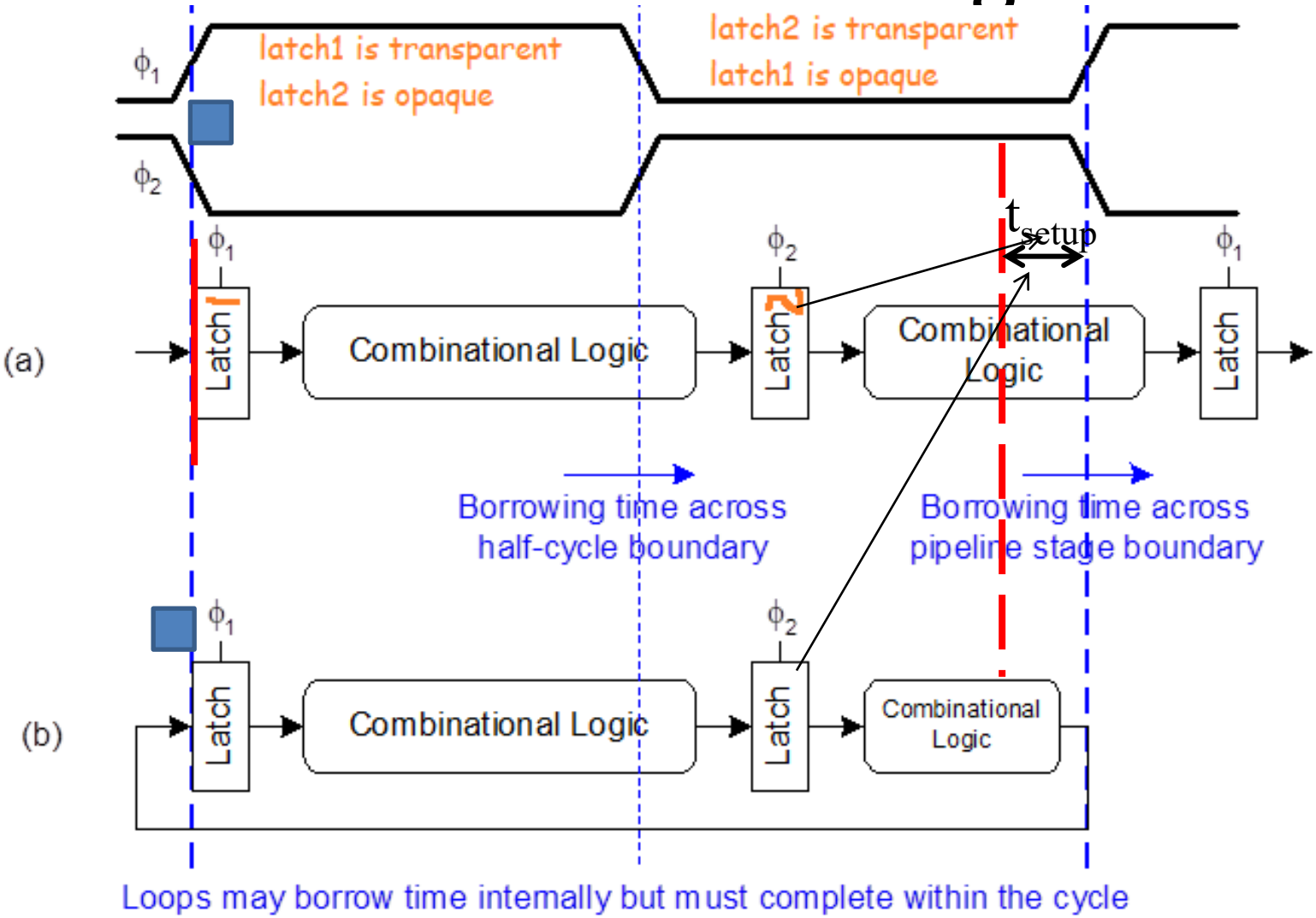
- In a flip flop-based system:
  1. Data launches on one rising edge
  2. Must setup before next rising edge
  3. If it arrives late, system fails
  4. If it arrives early, time is wasted



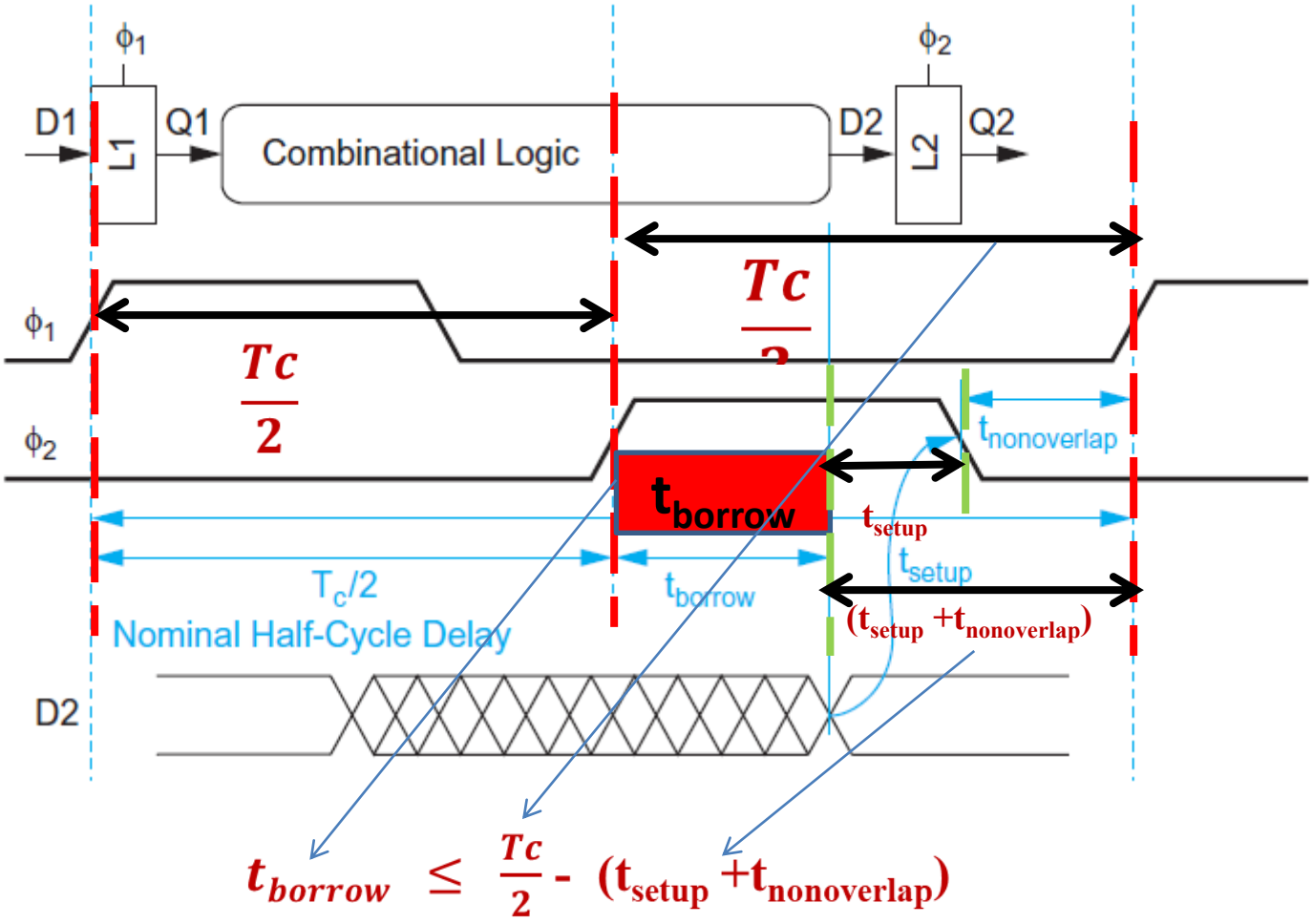
Time Borrowing

- In a latch-based system
  - Data can pass through latch while transparent
  - Long cycle of logic can borrow time into next
  - As long as each loop completes in one cycle

Time Borrowing



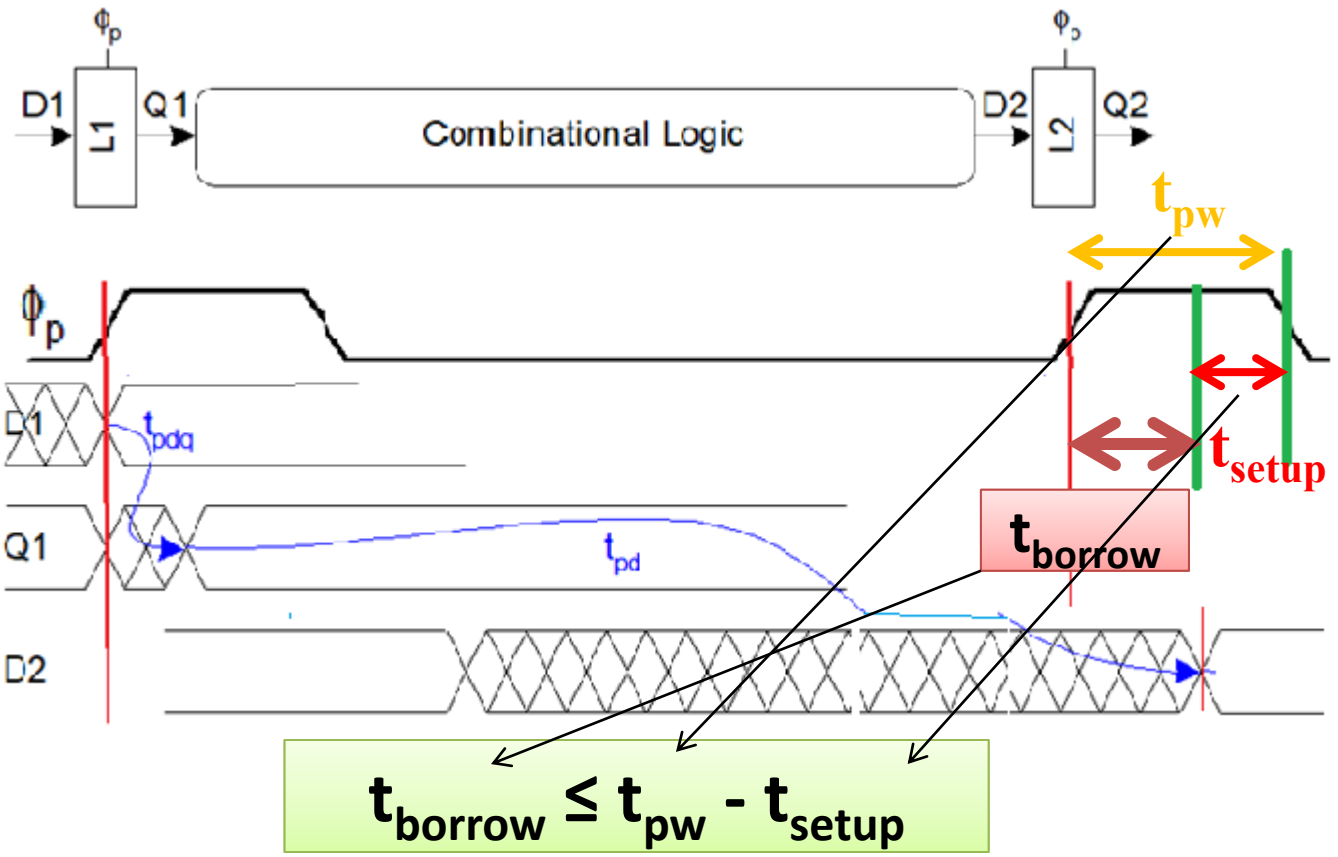
How Much time can be borrowed in two phase latch method ?



How Much time can be borrowed in Pulsed Latch method ?

- **Time borrowing is applicable only for the case where**

$t_{pw} > t_{setup}$



Problems on Time Borrowing

- **Example 10.5**: Suppose the ALU self-bypass path is modified to use two-phase transparent latches. A mid-cycle Φ2 latch is placed after the adder, as shown in Figure 10.14. The latches have a setup time of 40 ps, a hold time of 5 ps, a *clk*-to-*Q* propagation delay of 82 ps and contamination delay of 52 ps, and a *D*-to-*Q* propagation delay of 82 ps.
- Compute the minimum cycle time for the path. How much time is borrowed through the mid-cycle latch at this cycle time? If the cycle time is increased to 2000 ps, how much time is borrowed?

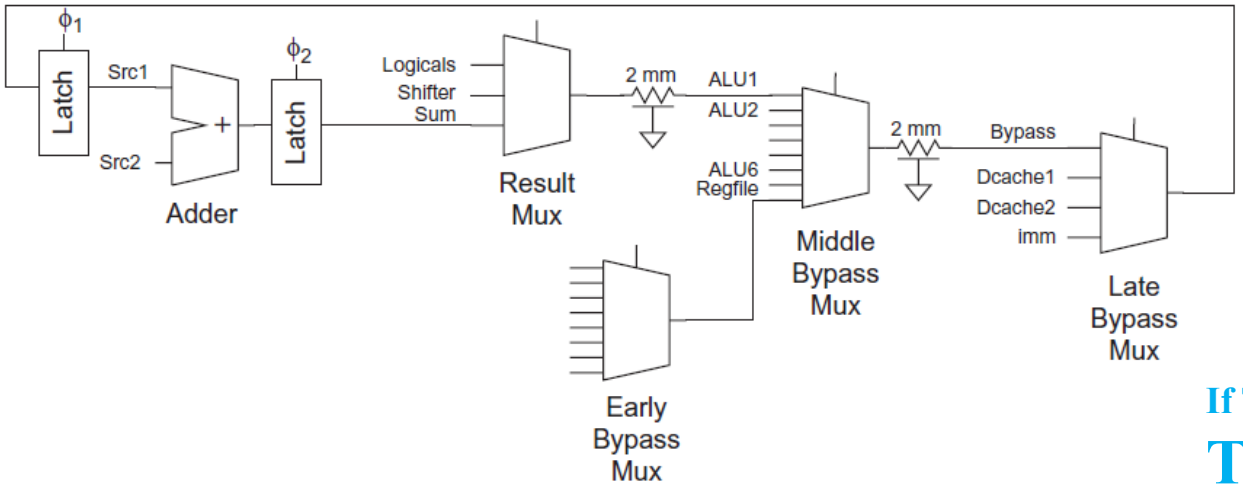


FIGURE 10.14 ALU self-bypass path with two-phase latches

$T_c = ?$

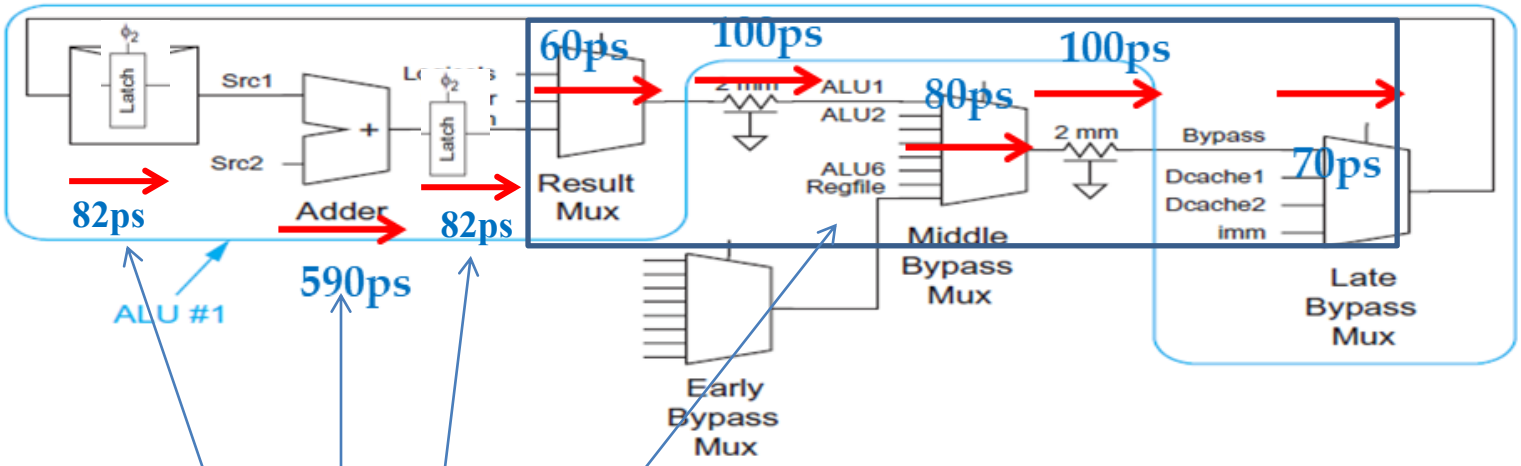
$T_{borrow} = ?$

If Tc is increased to 2000ps

$T_{borrow} = ?$

Problems on Time Borrowing

- **Solution:** According to $T_c \geq t_{pdq1} + t_{pd1} + t_{pdq2} + t_{pd2}$



The latches have a setup time of 40 ps, a hold time of 5 ps, a *clk*-to-*Q* propagation delay of 82 ps and contamination delay of 52 ps, and a *D*-to-*Q* propagation delay of 82 ps.

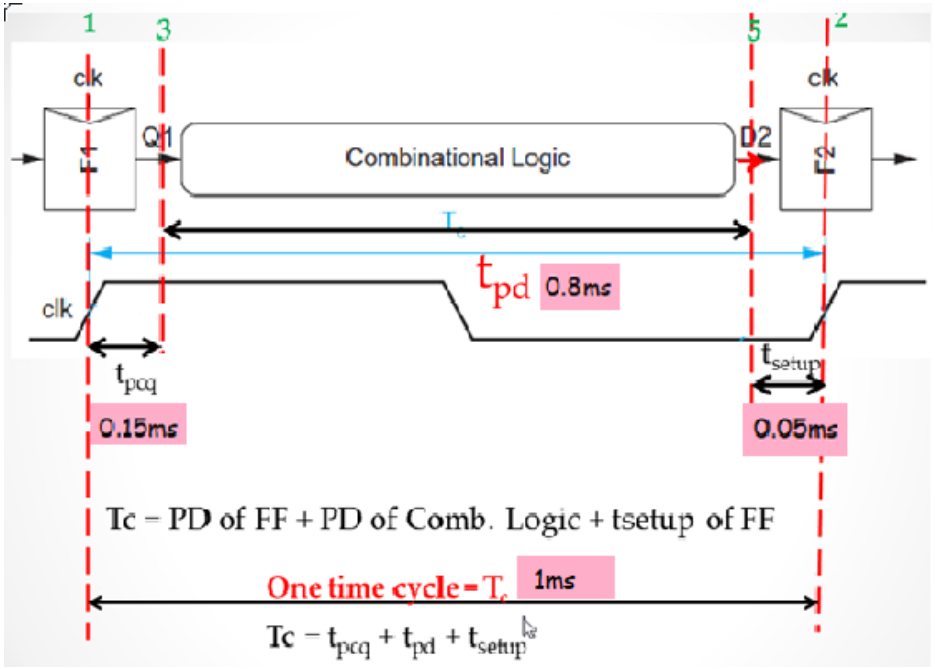
- The cycle time is *Tc* = 82 + 590 + 82 + 410 = 1164 ps.
- The first half of the cycle involves the latch and adder delays and consumes 82 + 590 = 672ps.
- The nominal half-cycle time is *Tc* /2 = 1164 / 2 = 582 ps.
  Hence, the path borrows 672 – 582 = 90 ps from the second half-cycle.
- If the cycle time increases to 2000 ps and the nominal halfcycle time becomes 1000 ps, time borrowing no longer occurs.

Clock Skew

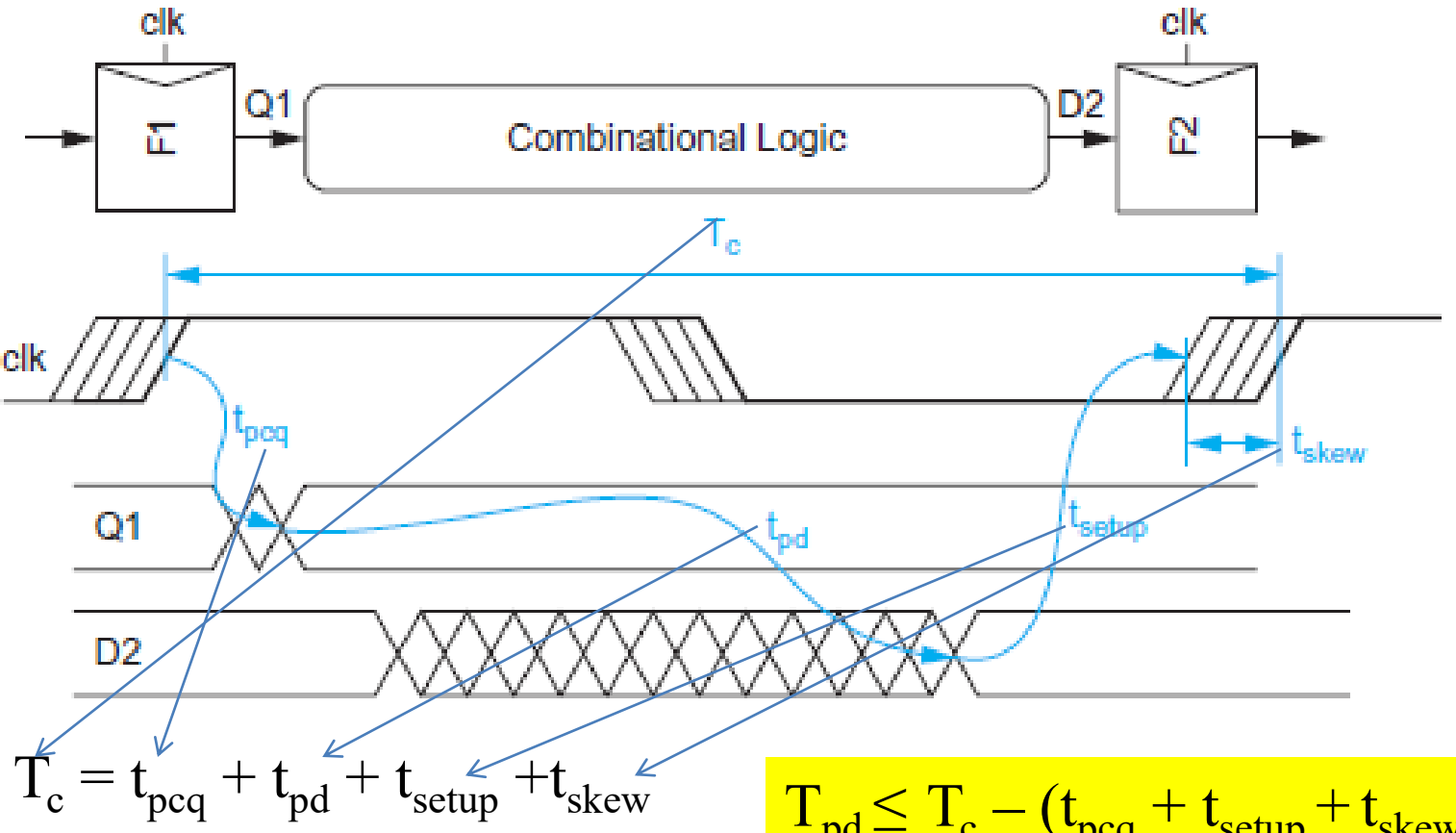
- We have assumed zero clock skew till now
- But Clocks really have uncertainty in arrival time, which will
  - Decreases maximum propagation delay (means decreases the propagation delay of the combinational logic)
  - Increases minimum contamination delay (means increases the contamination delay of the combinational logic)
  - Decreases time borrowing

Clock skew in FF based system

- Assume clk arrives at 0.9ms instead of 1ms
- But the sequencing overheads (i.e., tpcq and tsetup) will not change
- So if clk arrives at 0.9ms, then data from the combinational circuit must be ready at 0.85ms ($T_c - t_{setup}$)
- But according to the example data from CL arrives at 0.95ms.
- The only solution is to decrease the propagation delay of the CL to 0.7ms instead of 0.8ms



Clock skew in FF based system

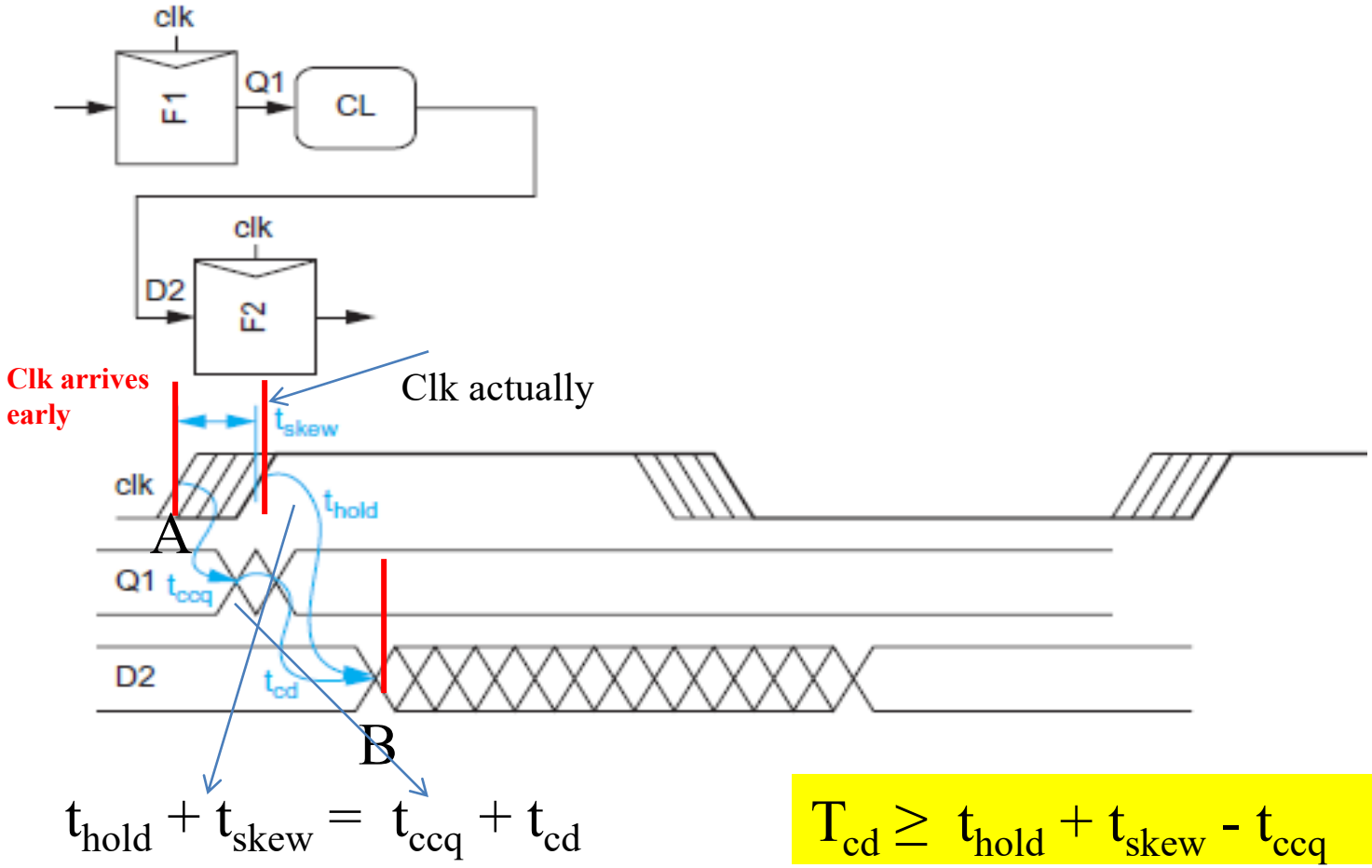


$$T_c = t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$T_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew})$$

- **Sequencing overheads(FF delay components) cannot be changed hence in order to avoid setup time failure the propagation delay of the logic block has to be reduced**

Clock skew in FF based svstem



$t_{hold} + t_{skew} = t_{ccq} + t_{cd}$

$T_{cd} \geq t_{hold} + t_{skew} - t_{ccq}$

- **Sequencing overheads(FF delay components) cannot be changed hence in order to avoid hold time failure the contamination delay of the logic block has to be increased**

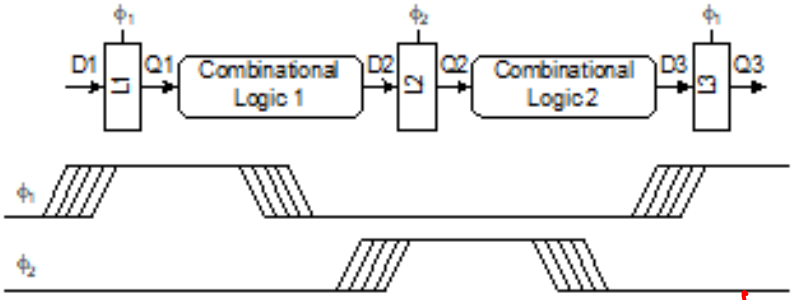
Clock skew in latches

2-Phase Latches

$$t_{pd} \le T_c - \underbrace{(2t_{pdq})}_{\text{sequencing overhead}}$$

$$t_{cd1}, t_{cd2} \ge t_{\text{hold}} - t_{ccq} - t_{\text{nonoverlap}} + t_{\text{skew}}$$

$$t_{\text{borrow}} \le \frac{T_c}{2} - (t_{\text{setup}} + t_{\text{nonoverlap}} + t_{\text{skew}})$$



Pulsed Latches

$$t_{pd} \le T_c - \underbrace{\max(t_{pdq}, t_{pcq} + t_{\text{setup}} - t_{pw} + t_{\text{skew}})}_{\text{sequencing overhead}}$$

$$t_{cd} \ge t_{\text{hold}} + t_{pw} - t_{ccq} + t_{\text{skew}}$$

$$t_{\text{borrow}} \le t_{pw} - (t_{\text{setup}} + t_{\text{skew}})$$

$t_{pw} < t_{setup}$

$t_{pd} \le T_c - (t_{pcq} + t_{setup} - t_{pw} + t_{skew})$

$t_{pw} > t_{setup}$

$t_{pd} \le T_c - t_{pdq}$

THANK YOU

Rekha S S

Department of Electronics and Communication

rekha.ss@pes.edu