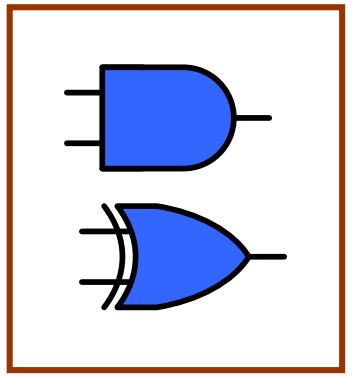
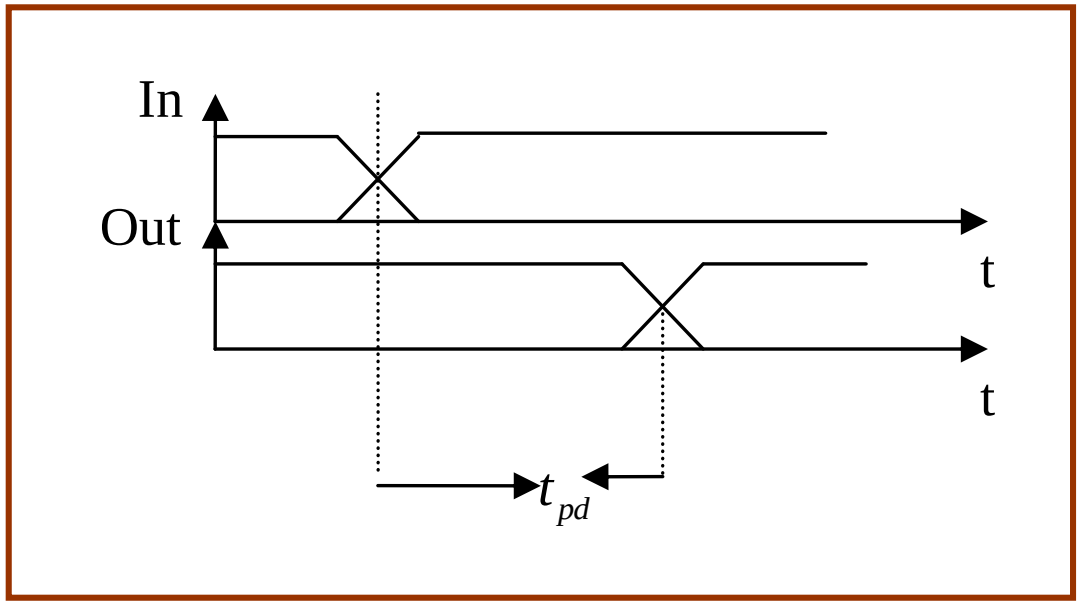
Digital Logic Analysis

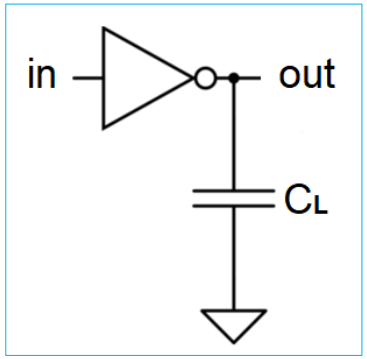
Logic Gates Timing

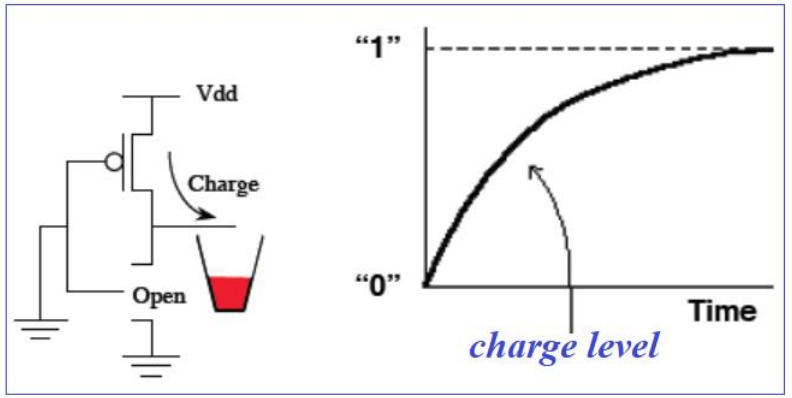
t_{pd}: Time from state change at input to state change at output

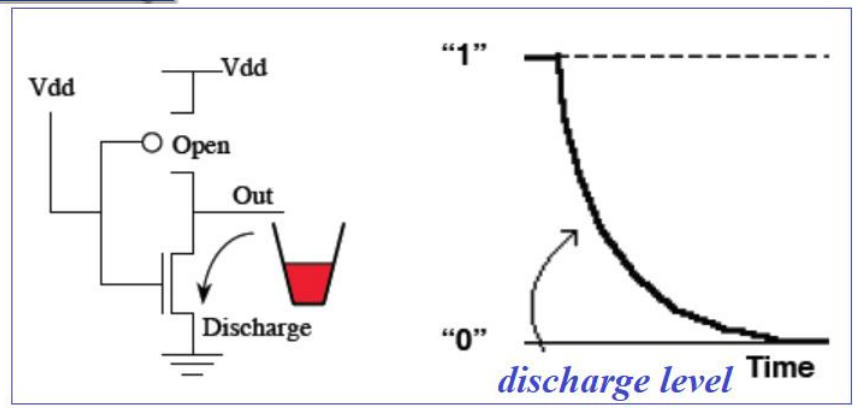




CMOS NOT Gate Delay







$C_L = C_{out} + C_{line} + N \cdot C_{in}$

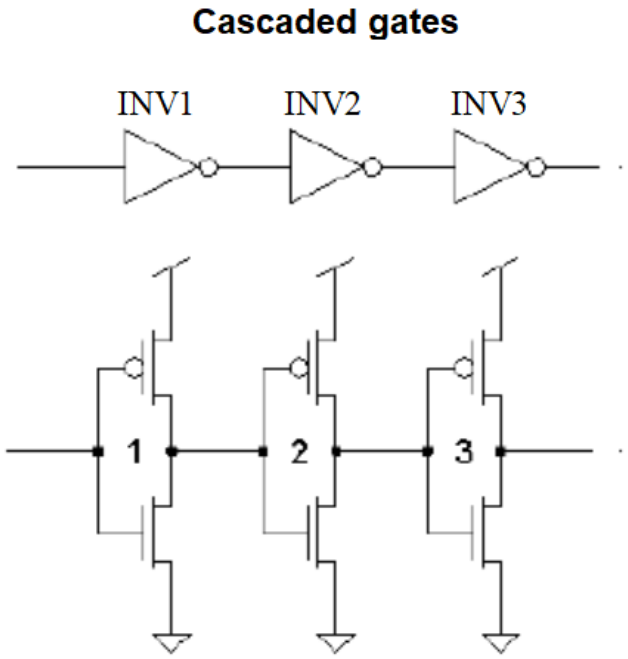
C_{out} - parasitic capacitance due to transistors configuration between output and GND

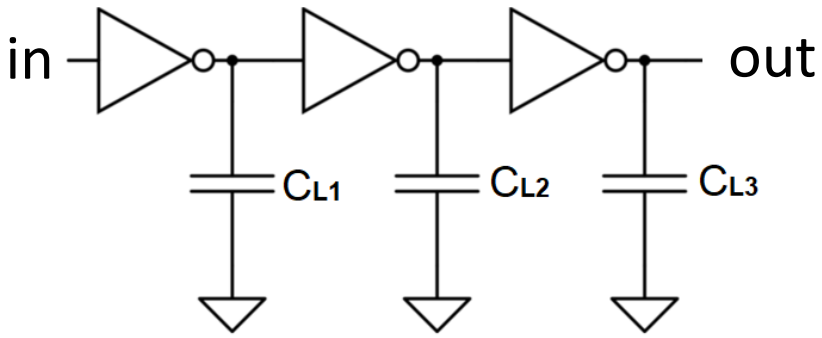
C_{line} - parasitic capacitance due to lines connection

$N \cdot C_{in}$- input capacitance of N gates at the next rank only (CMOS), connected to its output

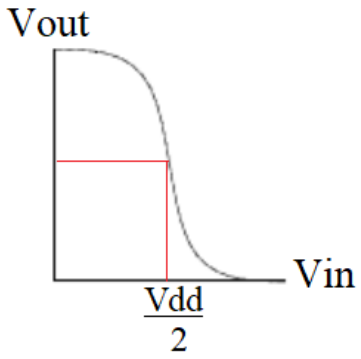
$t_{pd} = f(C_L, T[°C], V_{CC})$

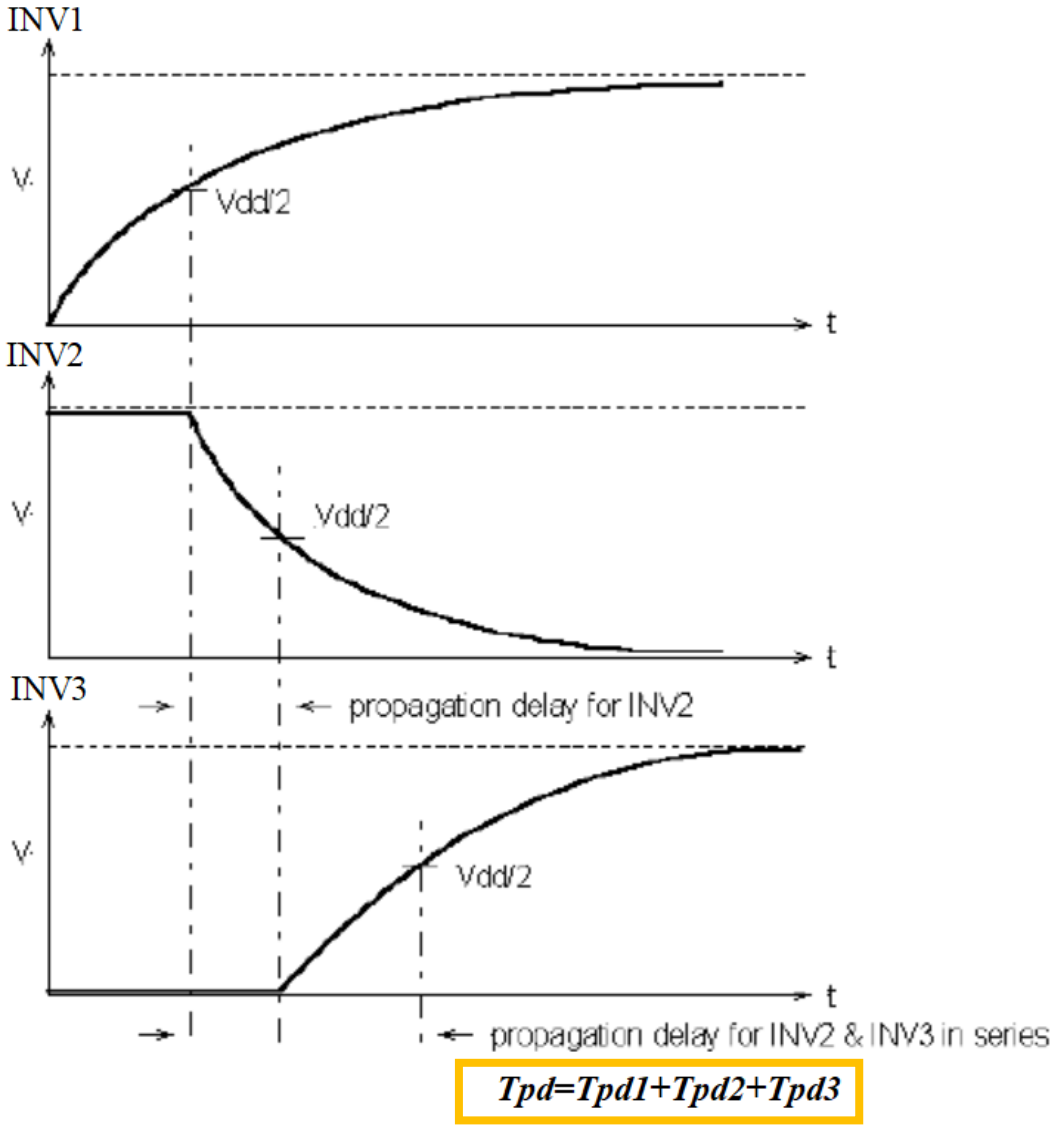
Cascaded gates delay





$$t_{pd} = t_{pd1} + t_{pd2} + t_{pd3}$$

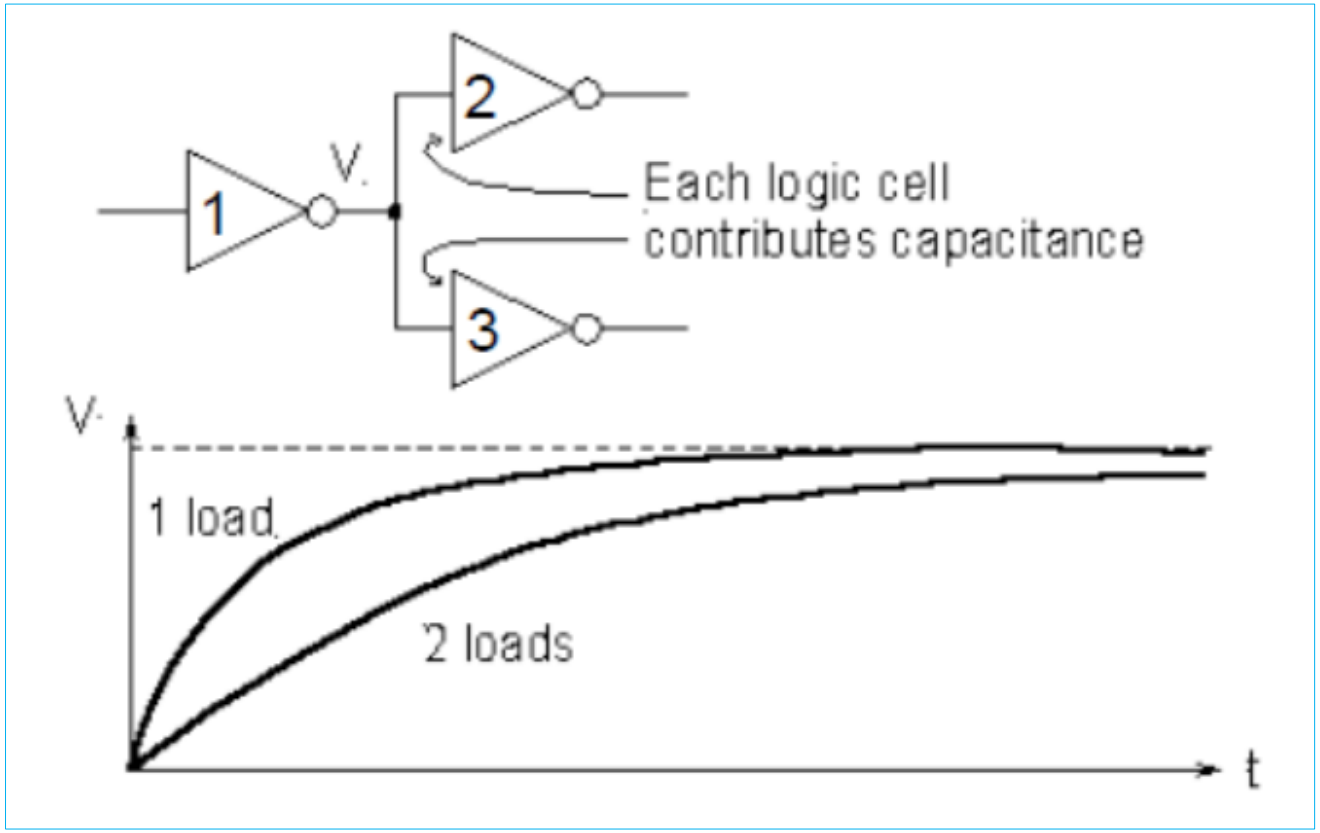




Tpd=Tpd1+Tpd2+Tpd3

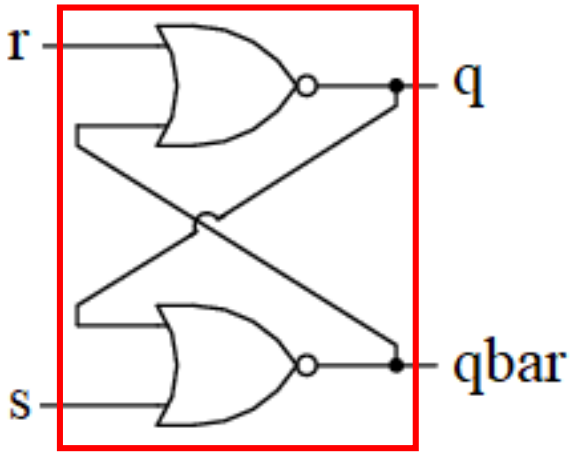
Fan-out delay

- The delay of a gate is proportional to its output capacitance. Connecting the output of gate to more than one other gate increases it's output capacitance.
- Driving wires also contributes to fan-out delay.



Synchronous Circuits Timing

NOR based SR Latch




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-----------------------------------------
ENTITY SRlatch IS
    PORT (  s, r: IN std_logic;
            q, qbar: BUFFER std_logic);
END SRlatch;
-----------------------------------------
ARCHITECTURE dataflow OF SRlatch IS
BEGIN

    q <= not(r or qbar);
    qbar <= not(s or q);

END dataflow;
```

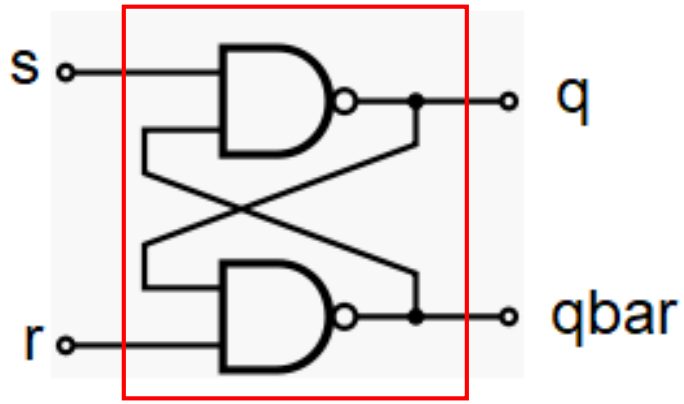
s	r	q	qbar
0	0	q	qbar
0	1	0	1
1	0	1	0
1	1	0	0

1-bit Memory=previous state (there are two possible different stable states)

Only a single stable state

Invalid input, causes a race condition when the input changes in the same time from s=r='1' to s=r='0' (there are two possible different stable states)

NAND based SR Latch




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-----------------------------------------
ENTITY SRlatch IS
    PORT (  s, r: IN std_logic;
            q, qbar: BUFFER std_logic);
END SRlatch;
-----------------------------------------
ARCHITECTURE dataflow OF SRlatch IS
BEGIN

    q <= not(s and qbar);
    qbar <= not(r and q);

END dataflow;
```

s	r	q	qbar
~~0~~	~~0~~	~~1~~	~~1~~
0	1	1	0
1	0	0	1
1	1	q	qbar

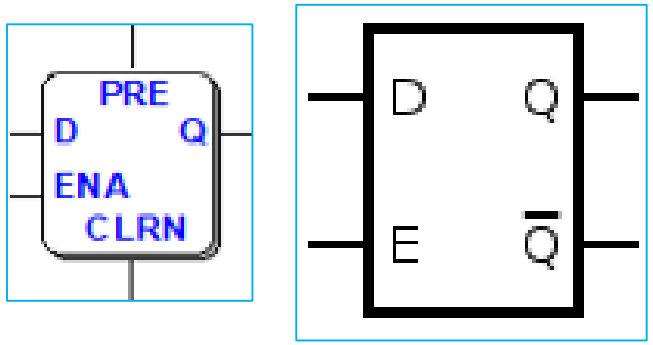
Invalid input, causes a race condition when the input changes in the same time from s=r='0' to s=r='1' (there are two possible different stable states)

Only a single stable state

1-bit Memory=previous state (there are two possible different stable states)

Gated D-Latch

In order to avoid from race condition (to get zero probability) in SR Latch it is modified to Gated D-Latch structure.



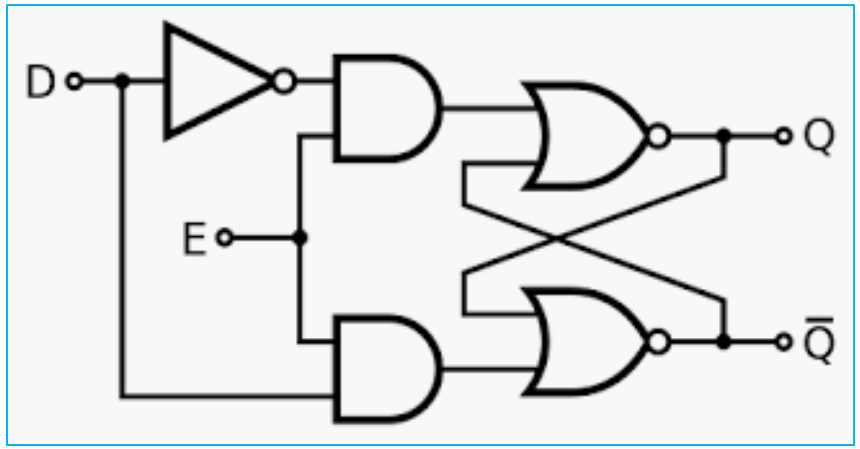
Symbol for a gated D latch

Output

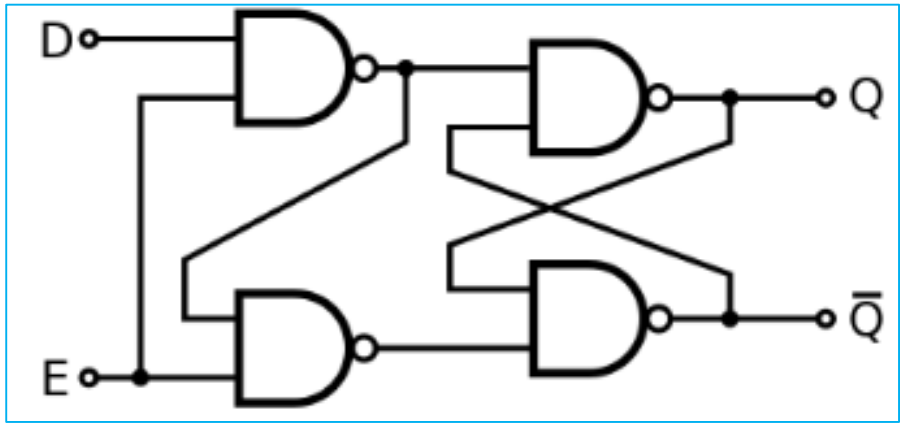
E=Enable	D=Data	Q	Qbar
0	x=don't care	Q	Qbar
1	0	0	1
1	1	1	0

1-bit Memory=previous state

Write operation



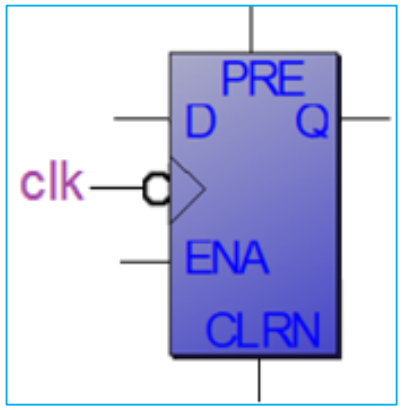
A gated D latch based on an SR NOR latch



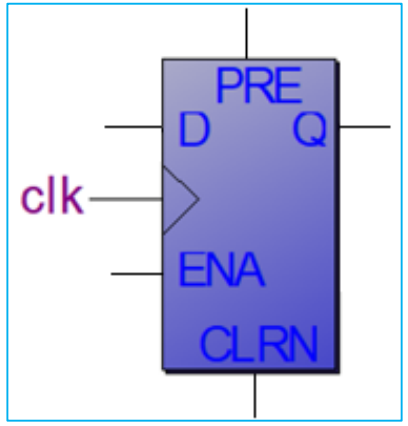
A gated D latch based on an SR NAND latch

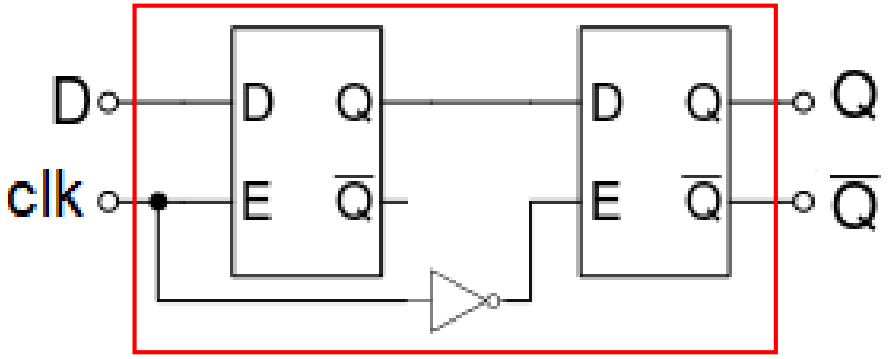
D Flip-Flop = DFF

In order to sample the input data at a single point in time (positive or negative edge trigger) we need to use rising/falling derivator structure.

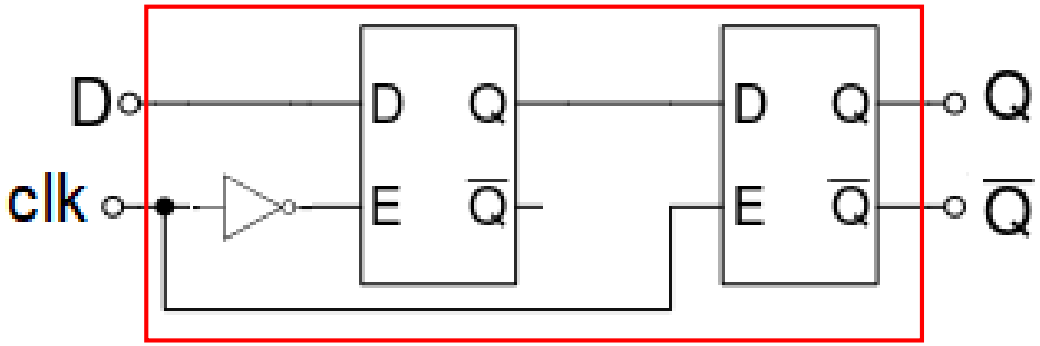


Symbol for a DFF



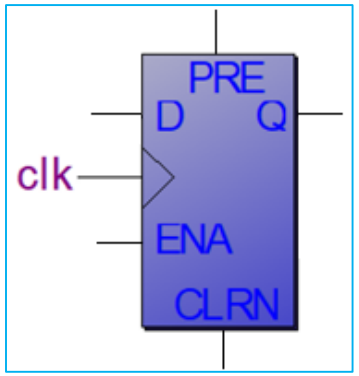


Negative edge DFF

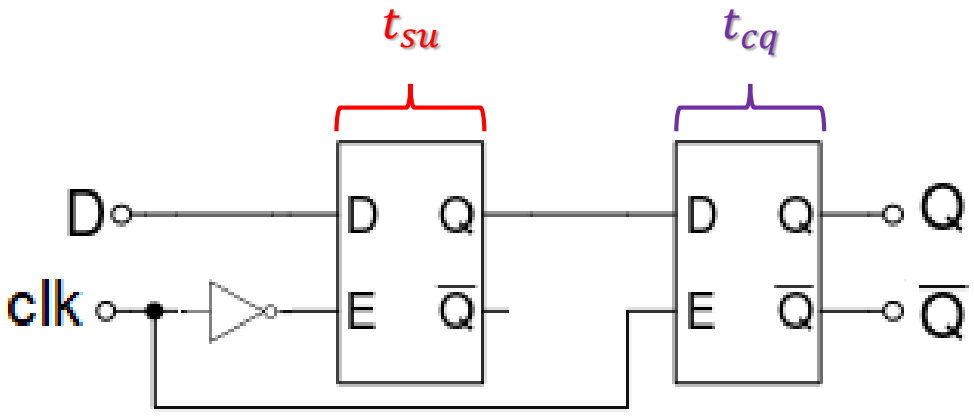


Positive edge DFF

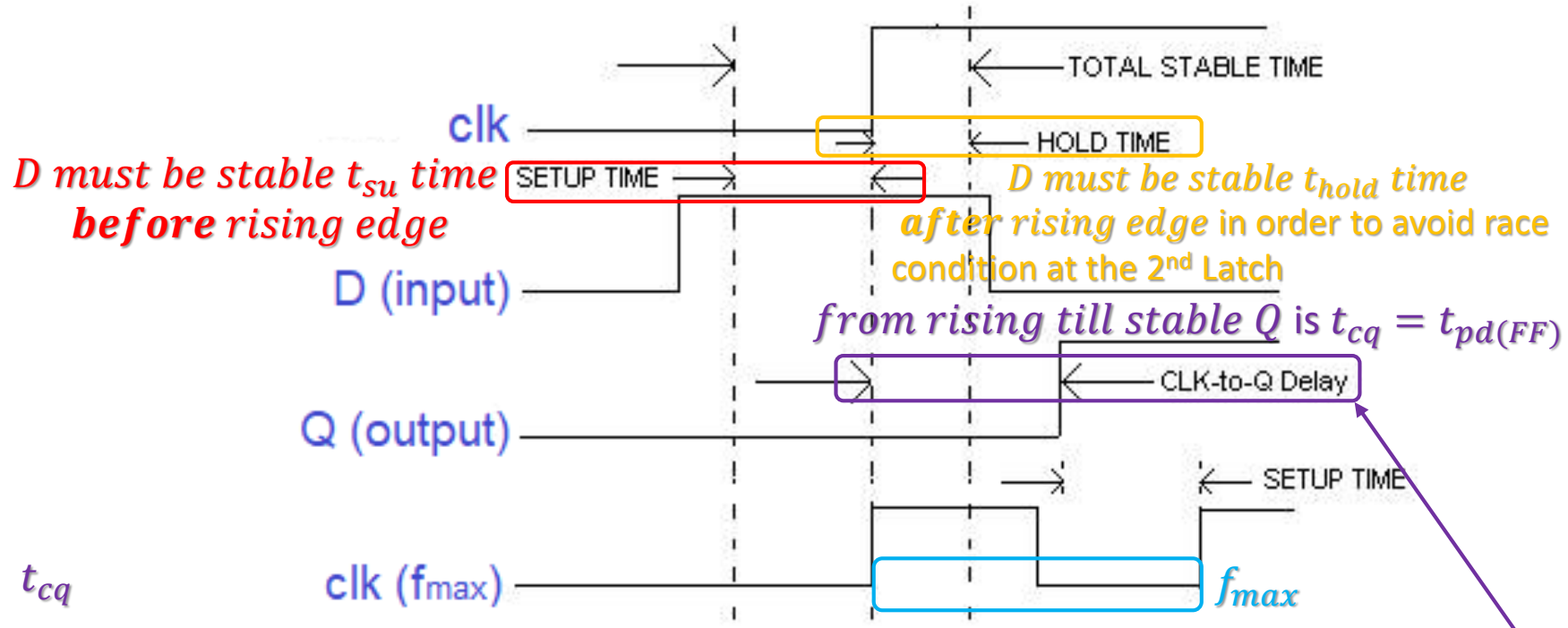
DFF timing analysis



Symbol for a DFF



Positive edge DFF

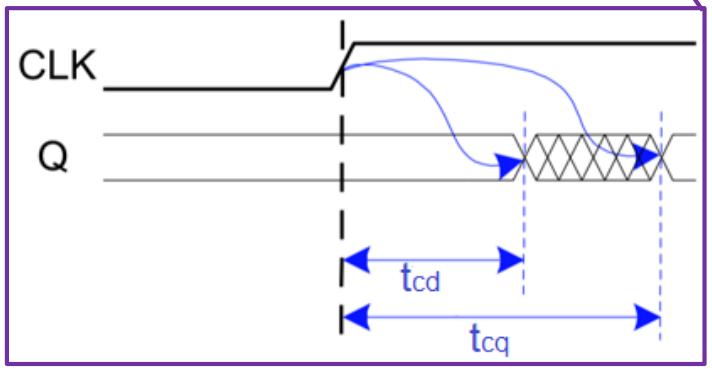


HW requirement:

$$t_{su} < t_{hold} < t_{cd} < t_{cq}$$

$$T_{min} = t_{cq} + t_{su}$$

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{t_{cq} + t_{su}}$$



DFF Timing recap

- $t_{cq} = t_{pd(FF)}$: time after clock change that the output is guaranteed to be stable (propagation delay).
- t_{su}: time before clock edge, data must be stable (Setup time)
- t_{hold} : time after clock edge data must be stable (Hold time)
- $t_a = t_{su}+t_{hold}$: time around clock edge data must be stable (Aperture time)
- t_{cd}: time after clock edge that Q is stable by the previous value (Continuation delay)
- $t_{cq} - t_{cd}$: time after clock edge that Q might be unstable (Contamination delay)

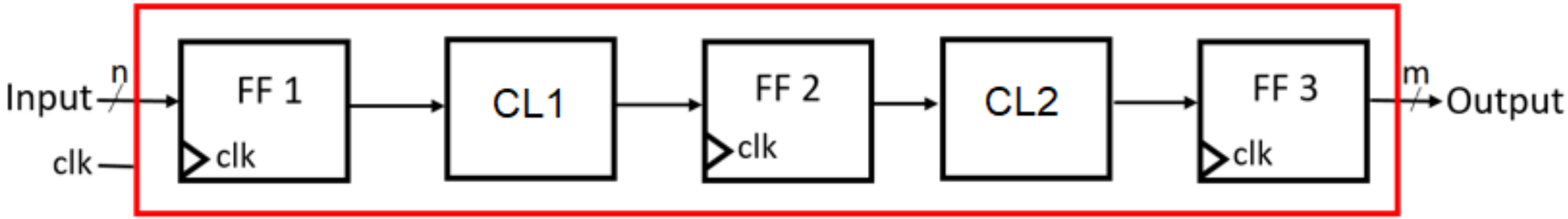
Performance Measurement

Continuation delay

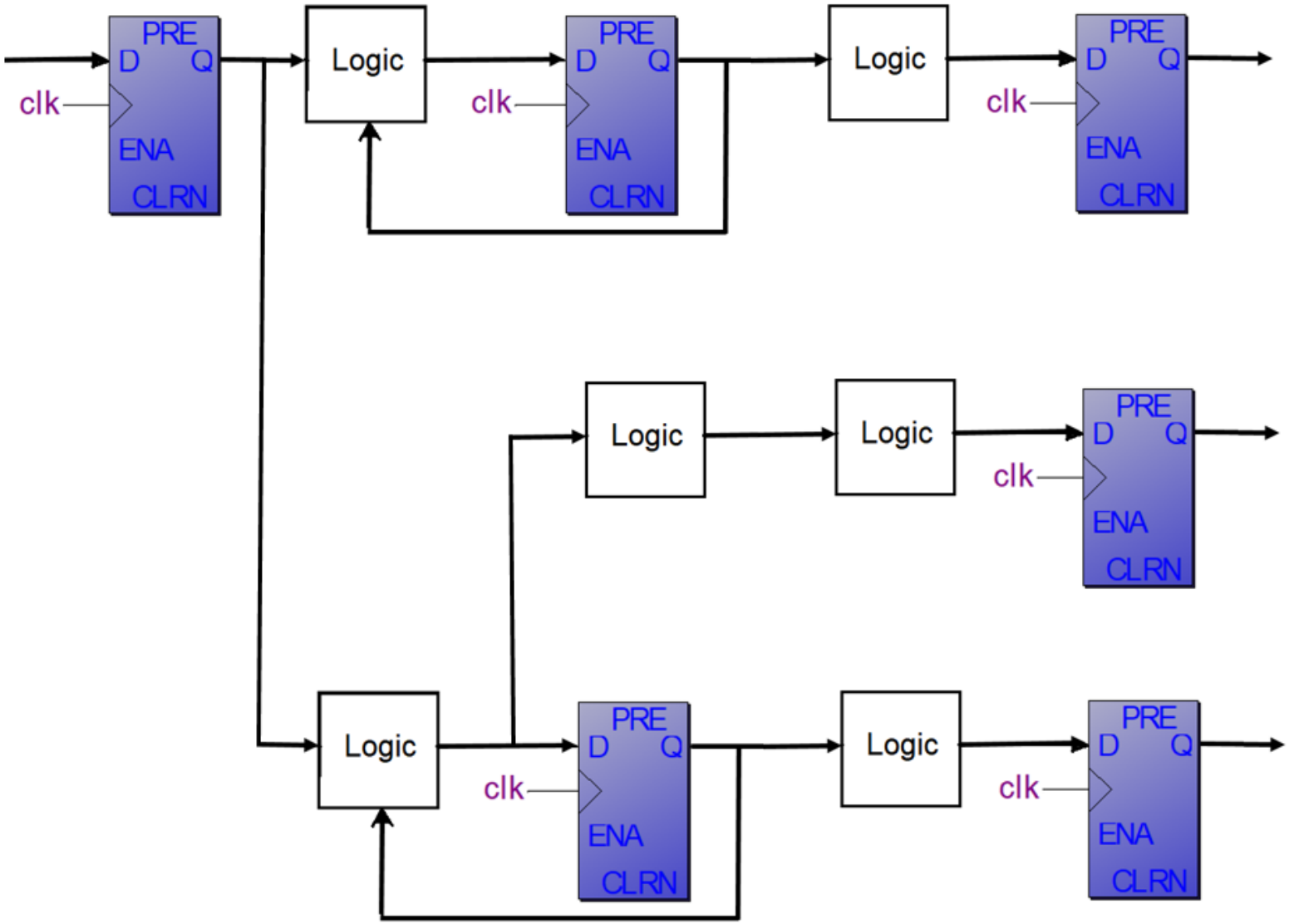
f_{max}

High Level System Description

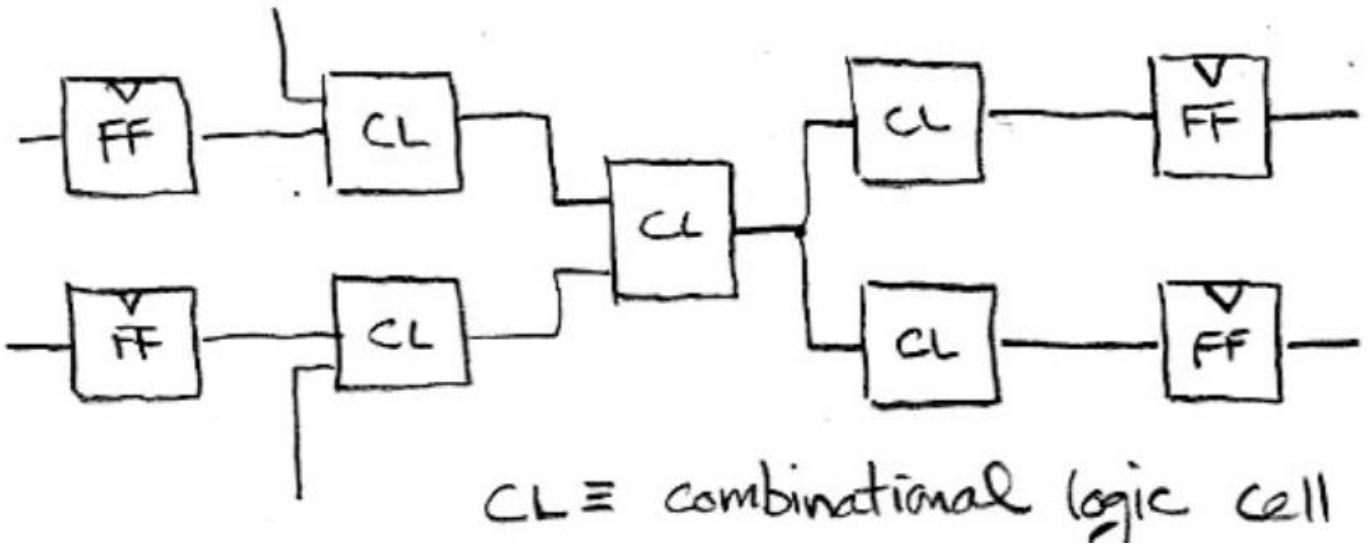
- The most common mistake is to start HDL coding before describing the required digital system in high level of RTL form.
- In RTL system description it becomes clear what are the system's combinational and synchronous subunits.
- Any digital logic system can be disassembled to Combinational Logic blocks chained to FFs (registers) .



RTL design - example

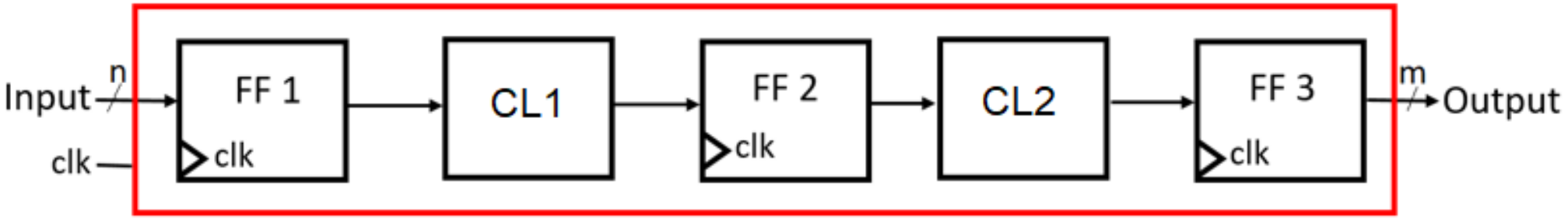


Components of Path Delay



1. # of levels of logic
2. Internal cell delay
3. wire delay
4. cell input capacitance
5. cell fan-out
6. cell output drive strength

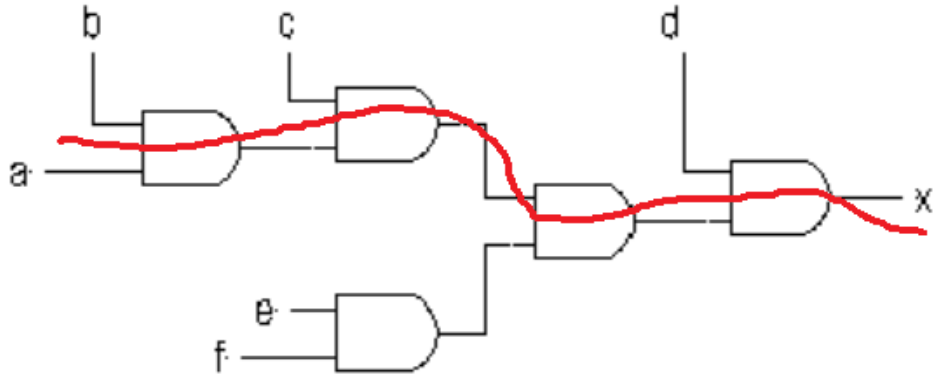
f_{max} calculation

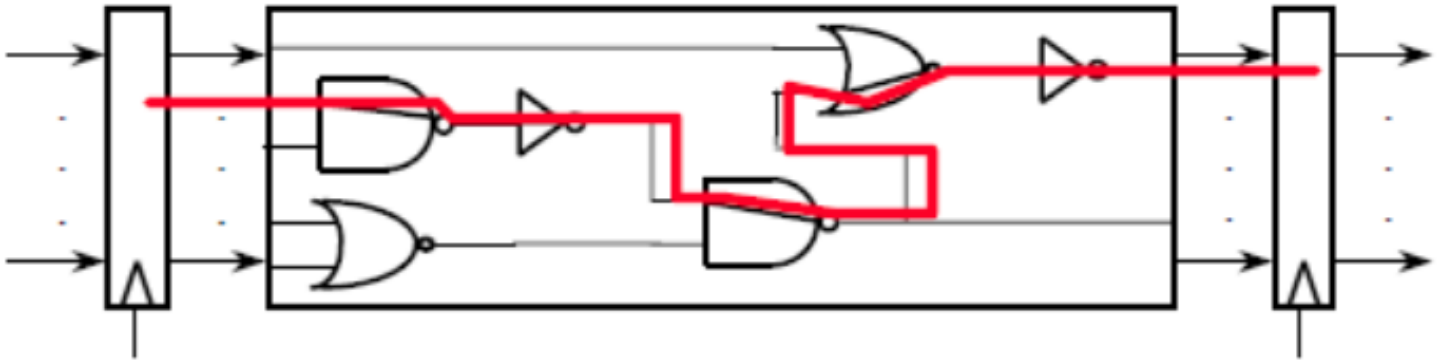


1. Notations:
   - Each CL_i has its own $t_{pd(CL_i)}$
   - We assume same technology for all FFs, means same $t_{cq}, t_{cd}, t_{su}, t_{hold}$
2. In order to get general expression for f_{max} , lets start with the timing condition between the path FF1 to FF2:
   - FF2 setup condition: $t_{cq(FF1)} + t_{pd(CL1)} + t_{su(FF2)} \leq T_{clk}$
   - FF2 hold condition: $t_{cd(FF1)} + t_{pd(CL1)} \geq t_{hold(FF2)}$
3. General expression for f_{max}: Critical Path - we strive in our design for balanced CL paths
   - Setup condition: $t_{cq} + \max_i\{t_{pd(CLi)}\} + t_{su} = T_{min} \rightarrow f_{max} = \frac{1}{T_{min}}$
   - Hold condition: $t_{cd} + \min_i\{t_{pd(CLi)}\} \geq t_{hold}$

Critical Path

- *Critical Path:* the path in the entire design with the maximum delay.
- This could be from state element to state element, from input to state element, state element to output, from input to output (unregistered paths).
- Example: what is the critical path in this circuit?

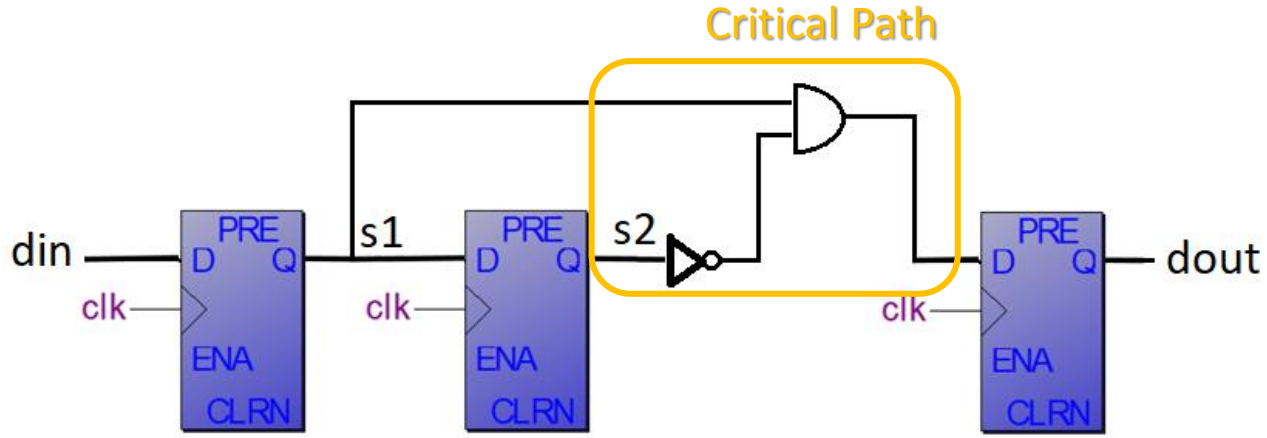




f_{max} calculation example

The Given data:

$$t_{su} = 200ps, t_{cq} = 300ps, t_{pd(gate)} = 100ps, hold\ condition\ exists$$

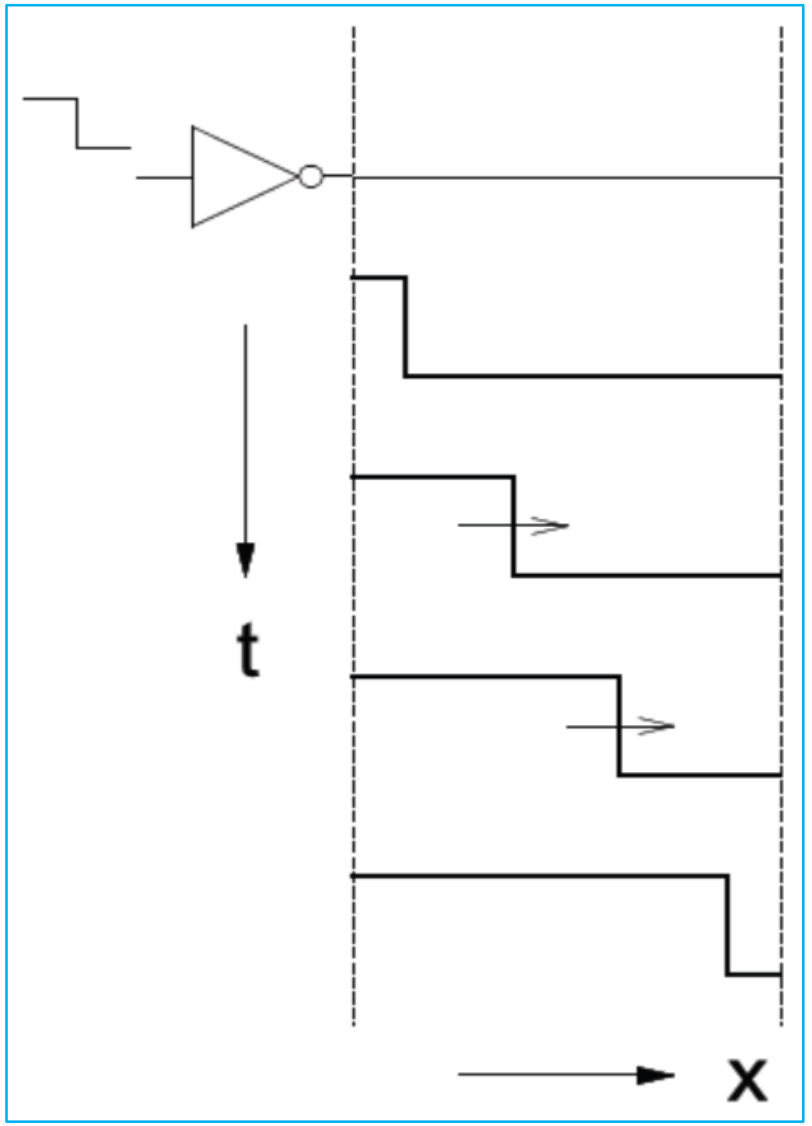


$$T_{min} = t_{cq} + 2 \cdot t_{pd(gate)} + t_{su} = 700ps \rightarrow f_{max} = \frac{1}{T_{min}} = 1.428\text{GHz}$$

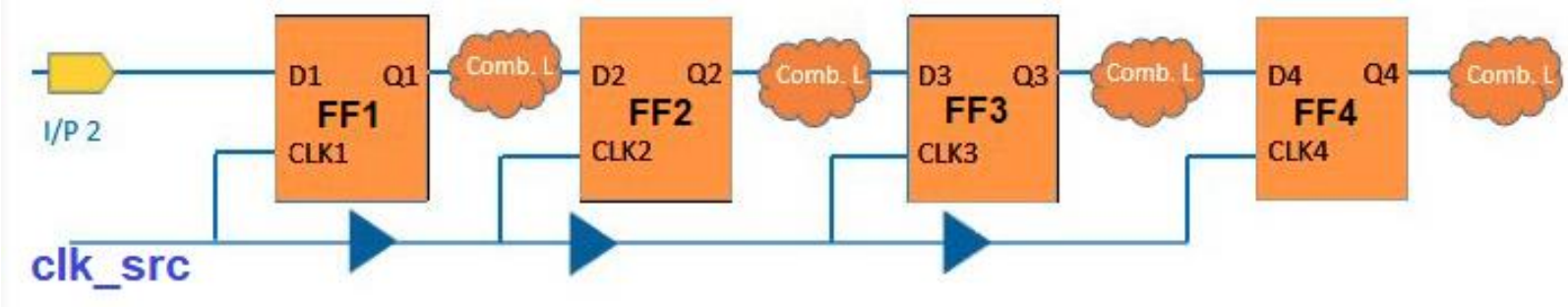
Clock Skew

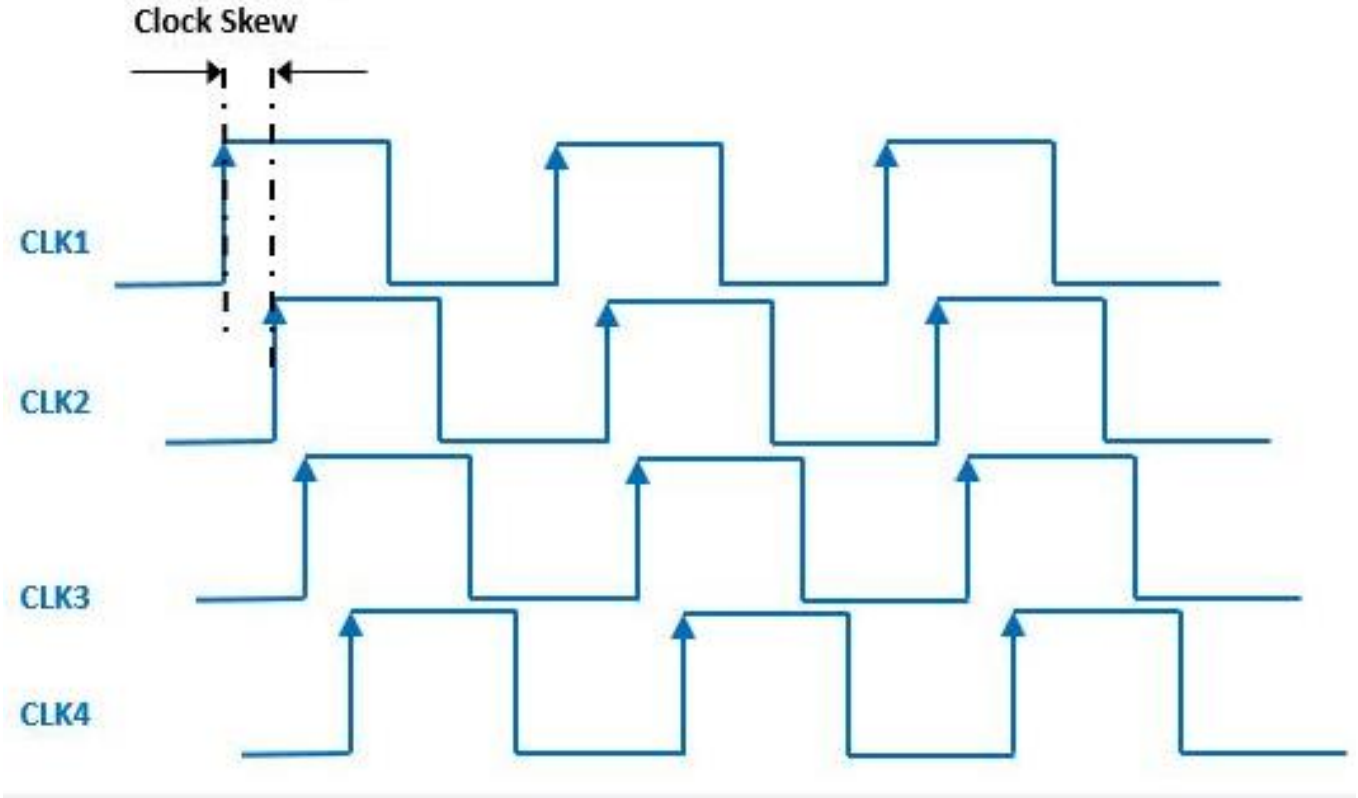
Wire Delay

- Signal wave-front moves close to the speed of light ~1ft/ns
- Time from source to destination is called the “transit time". In ICs most wires are short, and the transit times are relatively short compared to the clock period and can be ignored.
- The clk lines longer than ICs most wires and because physically the wires have resistance, capacitance and inductance (frequency dependent). There is clk skews (phase difference) between the systems FFs (Not all flip-flops see the clock at the same time).

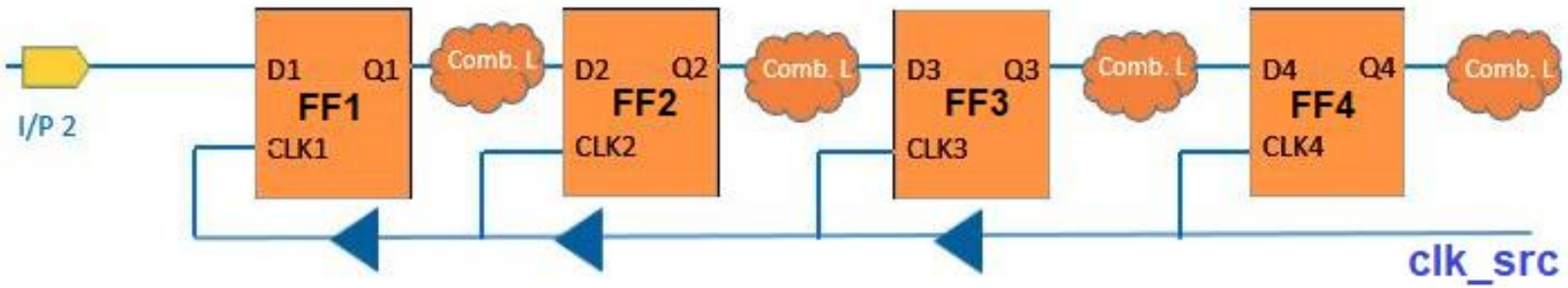


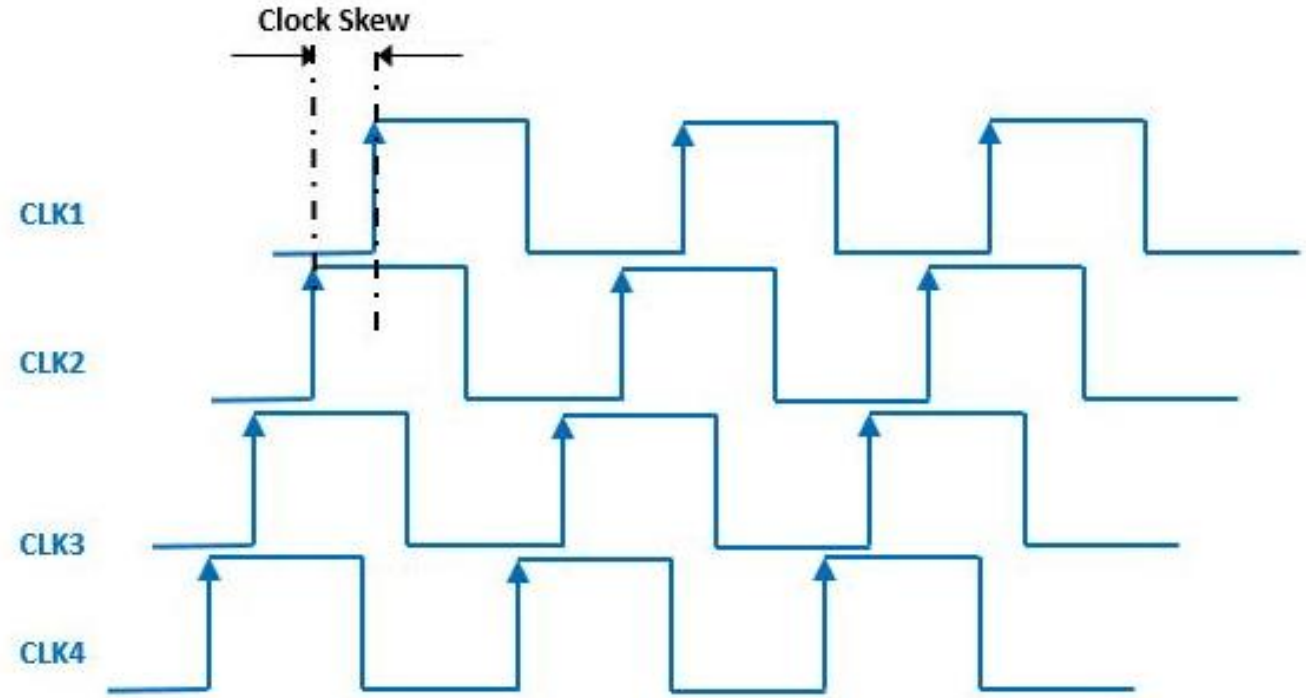
Clock Skew (positive skew)





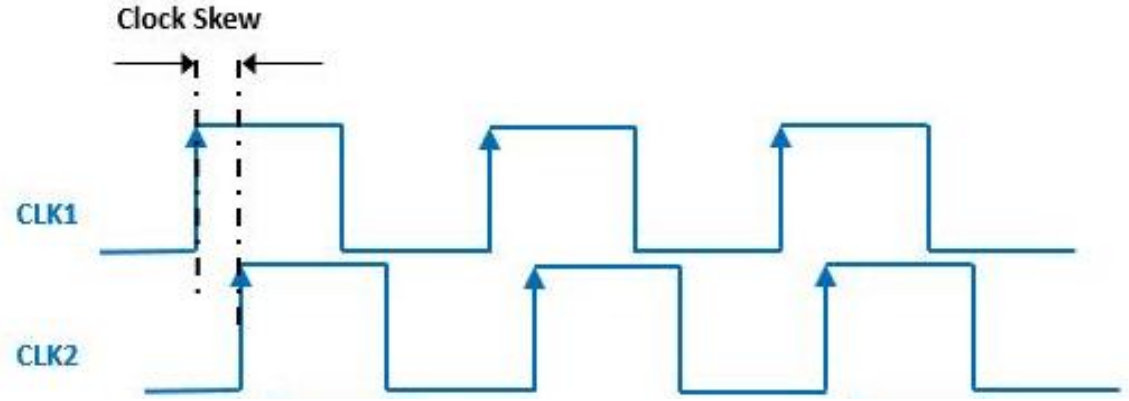
Clock Skew (negative skew)



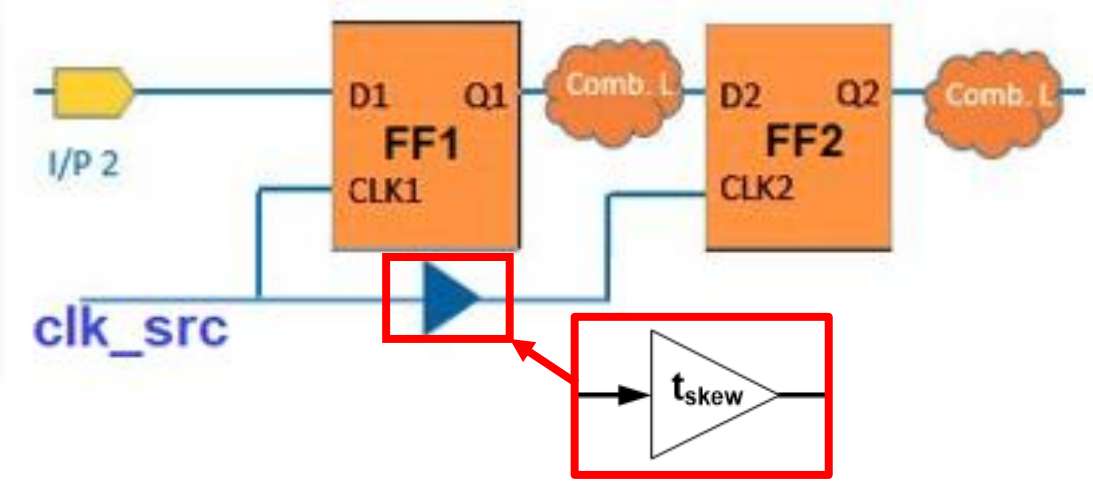


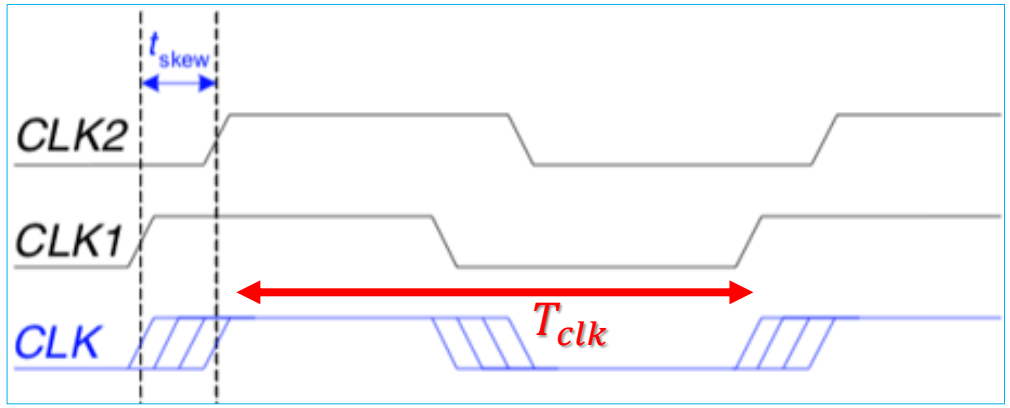
Clock Skew

In order to guarantee dynamic discipline is not violated for any register, we perform the worst case analysis.



$$clk2 = clk1 + t_{skew} \rightarrow clk1 = clk2 - t_{skew}$$





Private case:

- Setup condition (FF1 to FF2):
  $t_{cq(FF1)} + t_{pd(CL1)} + t_{su(FF2)} - t_{skew} \leq T_{clk}$
- Hold condition (FF1 to FF2):
  $t_{cd(FF1)} + t_{pd(CL1)} \geq t_{hold(FF2)} + t_{skew}$

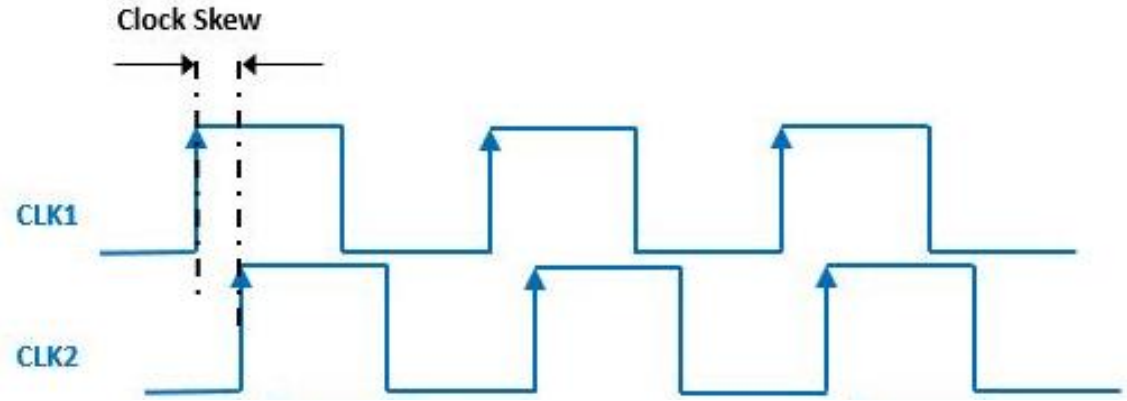
Timing Equations of Clock Skew - General case

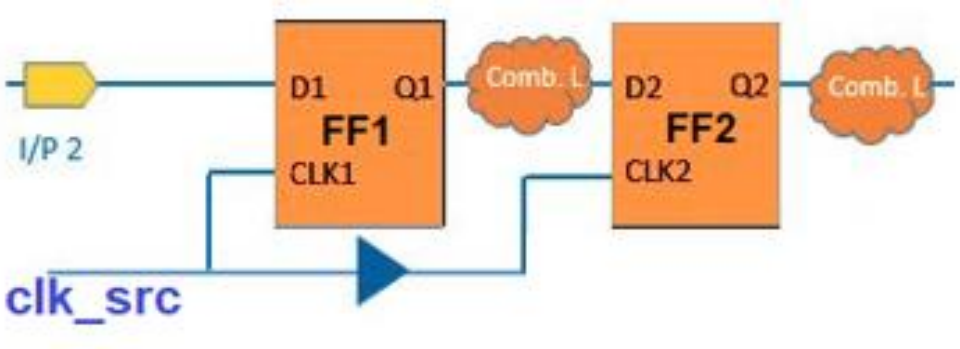
- $T_{min} = \frac{1}{f_{max}} = t_{cq} + max\{t_{pd(CLi)}\} + t_{su} - (s_d - s_s)$
- $t_{cd} + min\{t_{pd(CLi)}\} - t_{hold} \geq (s_d - s_s)$

When:

s_d is the clock skew to the destination register.

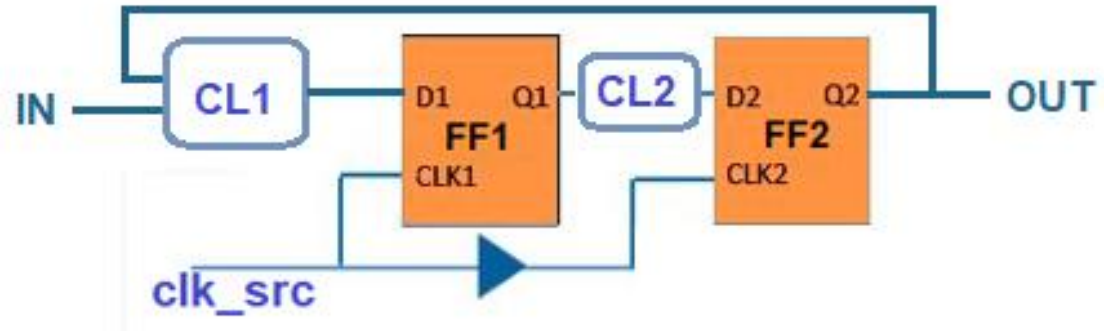
s_s is the clock skew to the source register.

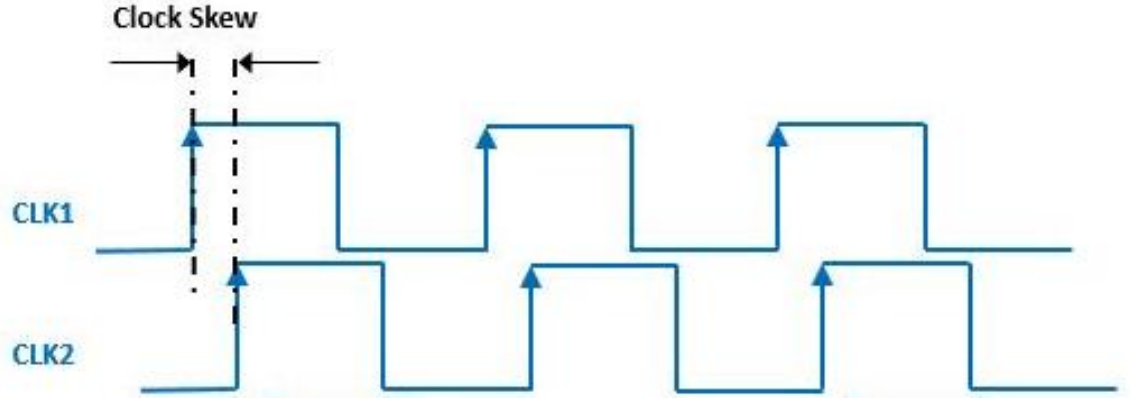




$clk2 = clk1 + t_{skew} \rightarrow s_s = clk1\ ,\ s_d = clk2 \rightarrow (s_d - s_s) = t_{skew}$

Clock Skew - example





$clk2 = clk1 + t_{skew}$

Given data:

$t_{su} = 2ns\ , t_{hold} = 3ns\ , t_{cd} = 4ns\ , t_{cq} = 5ns\ , t_{pd(CL1)} = 4ns\ , t_{pd(CL2)} = 2ns$

Q1: what is the t_{skew} *upper bound* = ?

$t_{skew} \leq 3ns$

- FF1 to FF2: $t_{cd} + t_{pd(CL2)} - t_{hold} \geq (s_d - s_s) \rightarrow (4 + 2 - 3)ns = 3ns \geq t_{skew}$
- FF2 to FF1: $t_{cd} + t_{pd(CL1)} - t_{hold} \geq (s_d - s_s) \rightarrow (4 + 4 - 3)ns = 5ns \geq -t_{skew}$

$t_{skew} \geq 5ns$

Q2: $f_{max} = \frac{1}{T_{min}} = ?$ $f_{max} = 71.4MHz$

$T_{min} = t_{cq} + max\{t_{pd(CLi)}\} + t_{su} - (s_d - s_s) = 5ns+4ns+2ns+3ns=14ns$