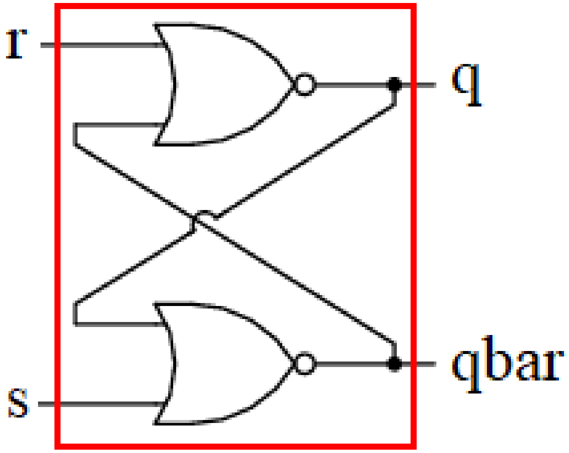
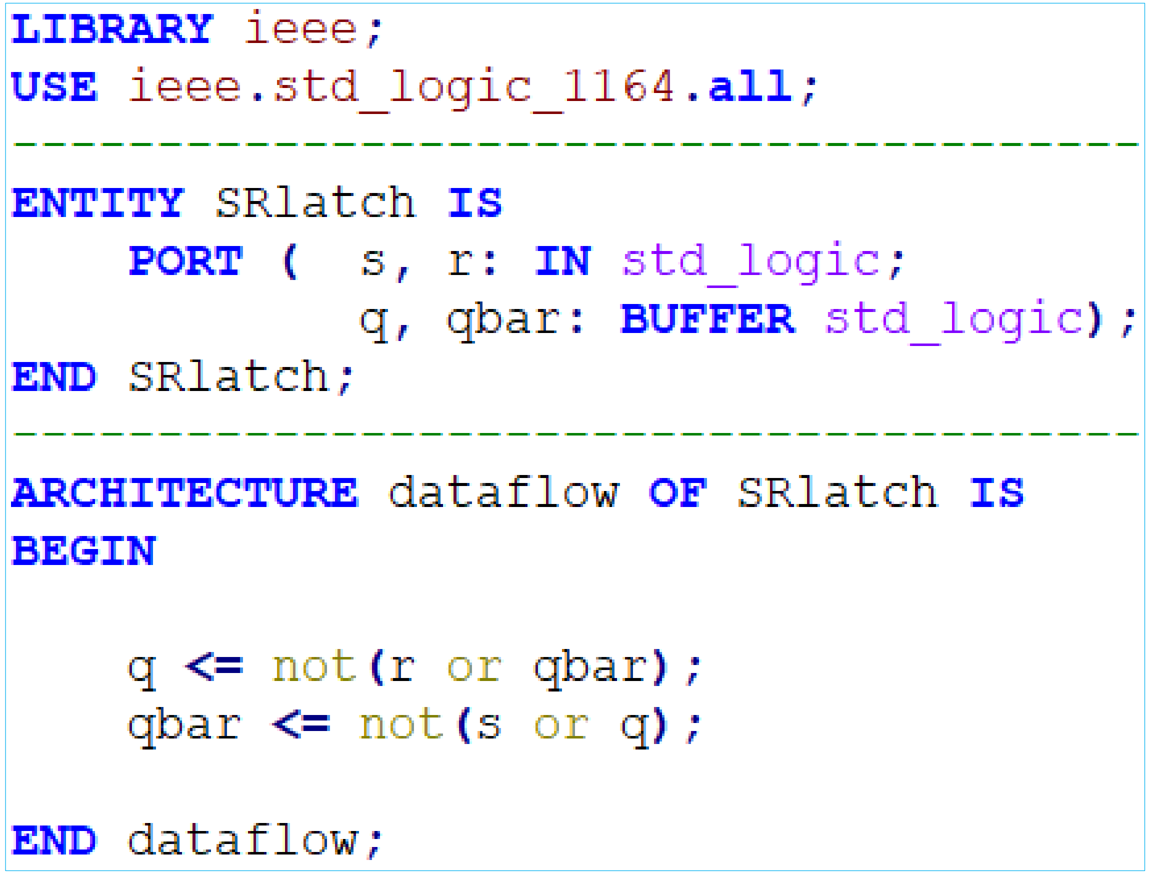
Synchronous Circuits Timing Analysis

©Hanan Ribo

NOR based SR Latch

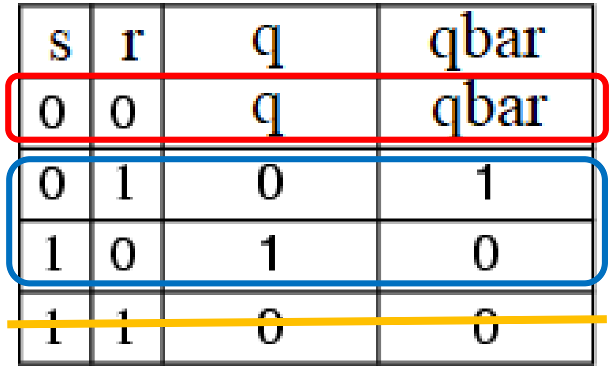




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------
ENTITY SRlatch IS
    PORT (  s, r: IN std_logic;
            q, qbar: BUFFER std_logic);
END SRlatch;
---------------------------------------
ARCHITECTURE dataflow OF SRlatch IS
BEGIN

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

END dataflow;
```



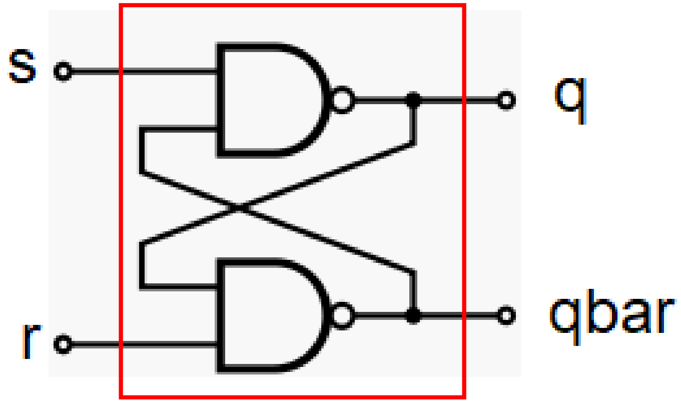
s	r	q	qbar
0	0	q	qbar
0	1	0	1
1	0	1	0
~~1~~	~~1~~	~~0~~	~~0~~

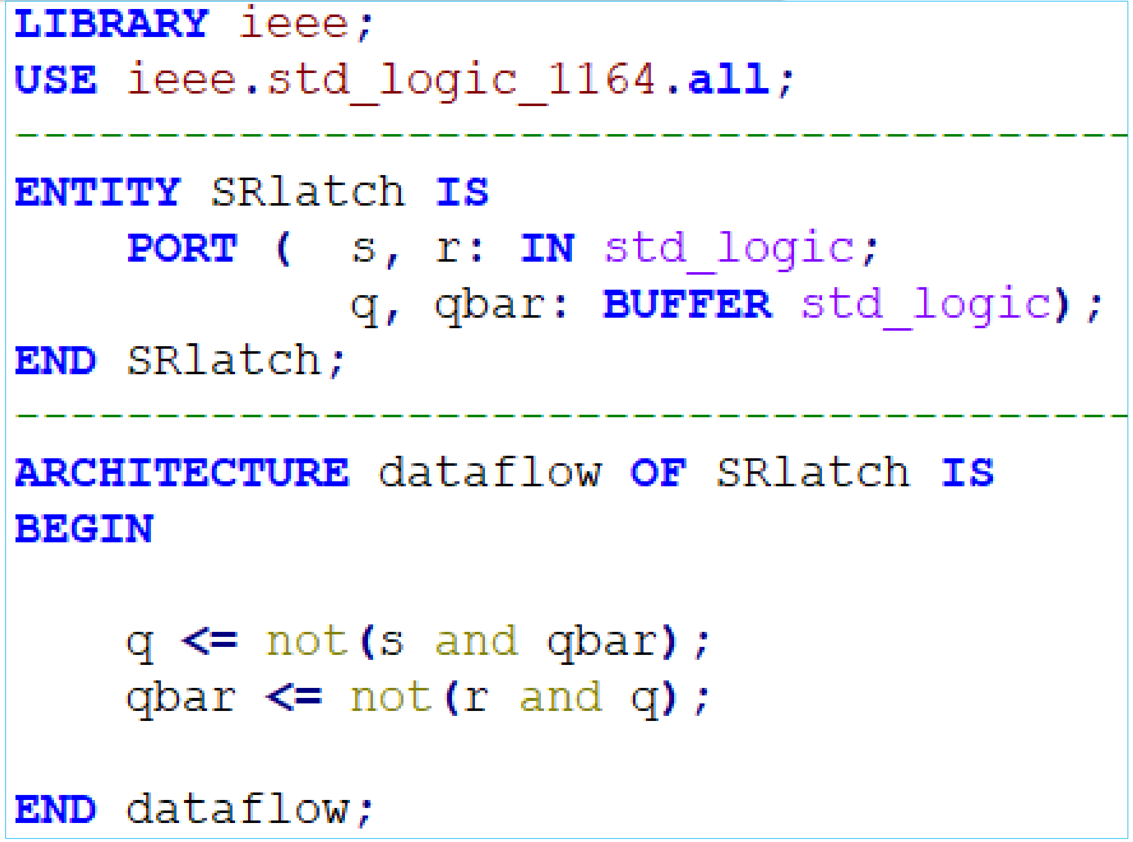
1-bit Memory=previous state (there are two possible different stable states)

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

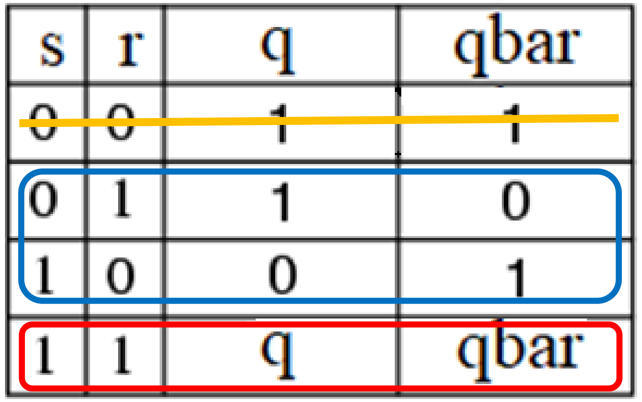




```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
----------------------------------------------
ENTITY SRlatch IS
    PORT (  s, r: IN std_logic;
            q, qbar: BUFFER std_logic);
END SRlatch;
----------------------------------------------
ARCHITECTURE dataflow OF SRlatch IS
BEGIN

    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)

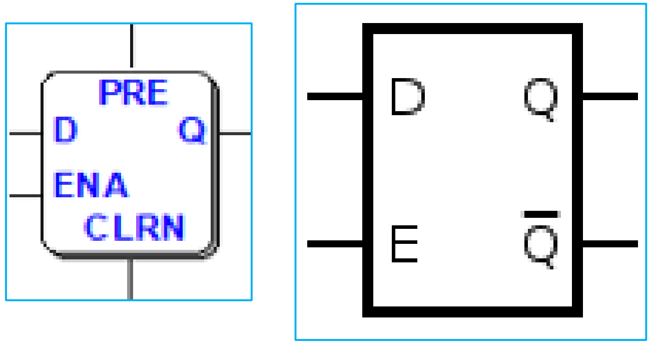
Only a single stable state

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

©Hanan Ribo

Gated D-Latch

In order to get rid from race condition (to get zero probability) in SR Latch we need to modify it to a structure called **Gated D-Latch** structure.



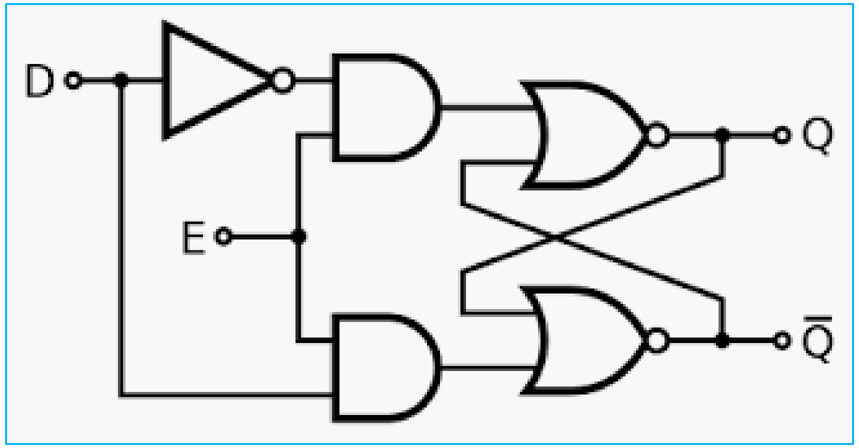
Symbol for a gated D latch

Output

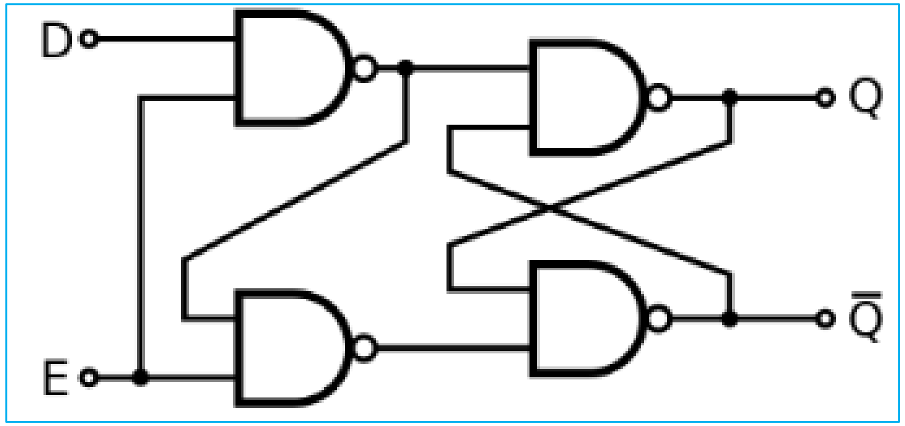
E=Enable	D=Data	Q	Qbar
0	x=don't care	Q	Qbar
1	0	0	1
1	1	1	0

1-bit Memory=previous state

Write operation



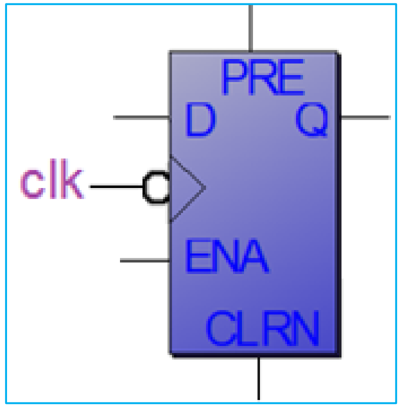
A gated D latch based on an SR NOR latch



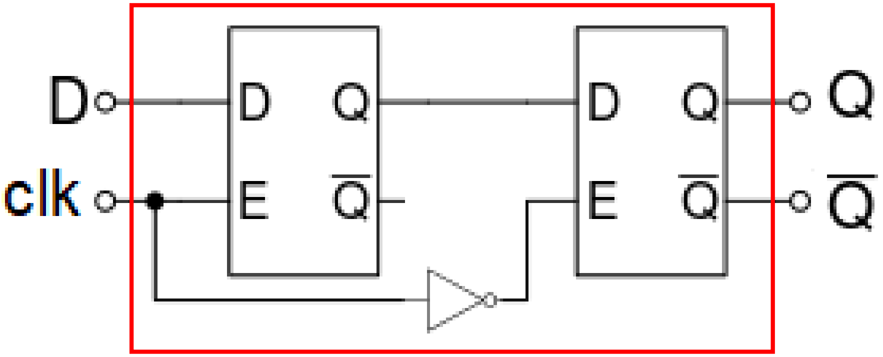
A gated D latch based on an SR NAND latch

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.

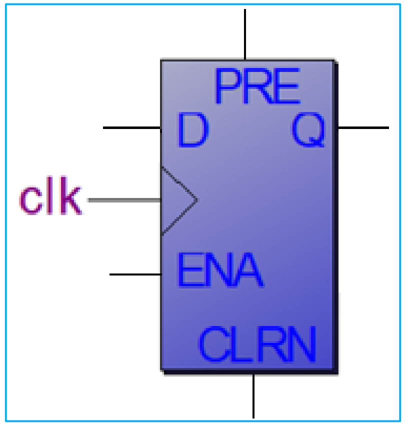


=

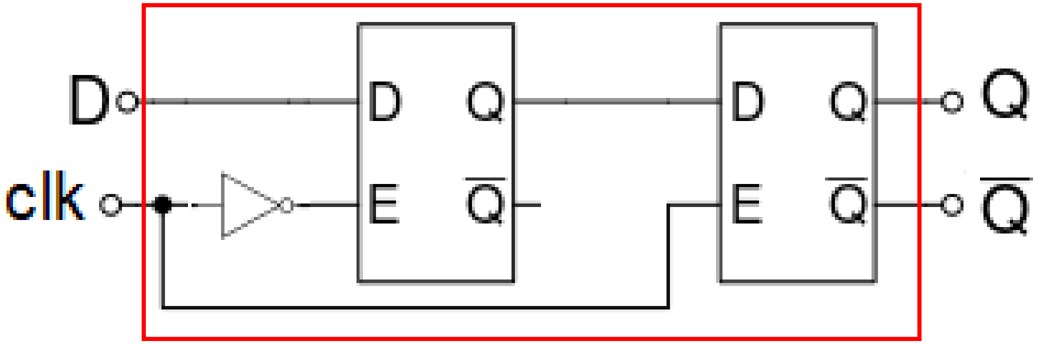


Negative edge DFF

Symbol for a DFF



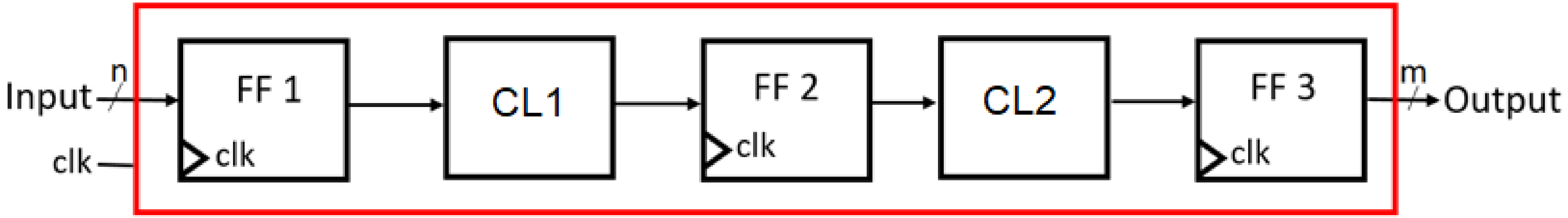
=

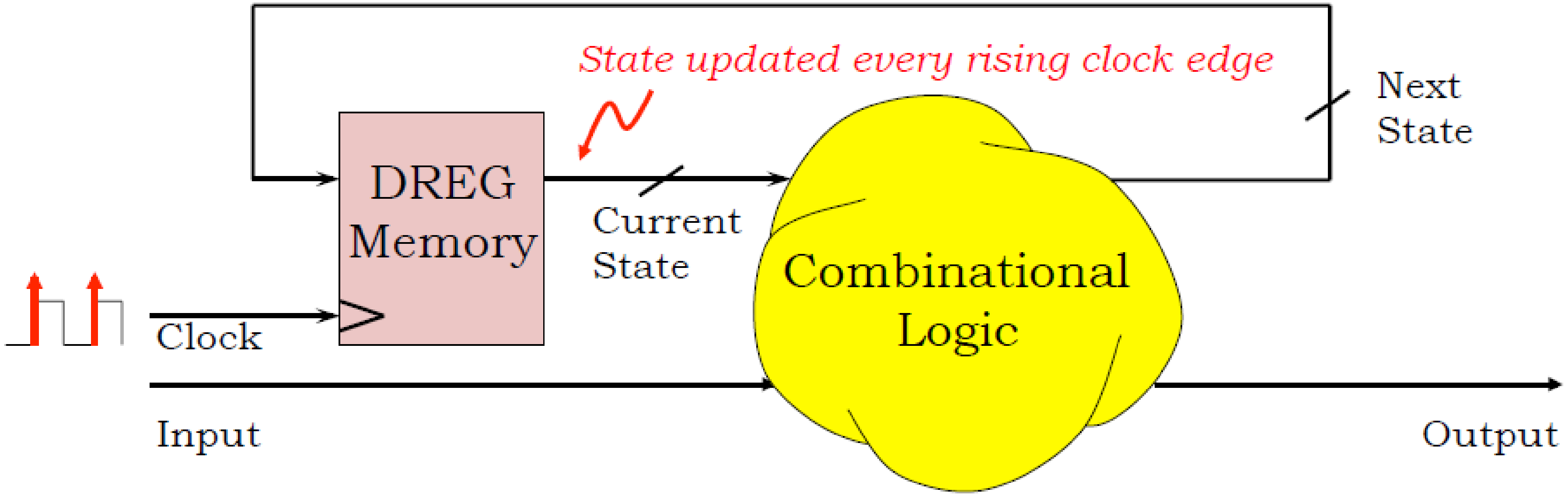


Positive edge DFF

Dynamic Discipline Obedience - Motivation

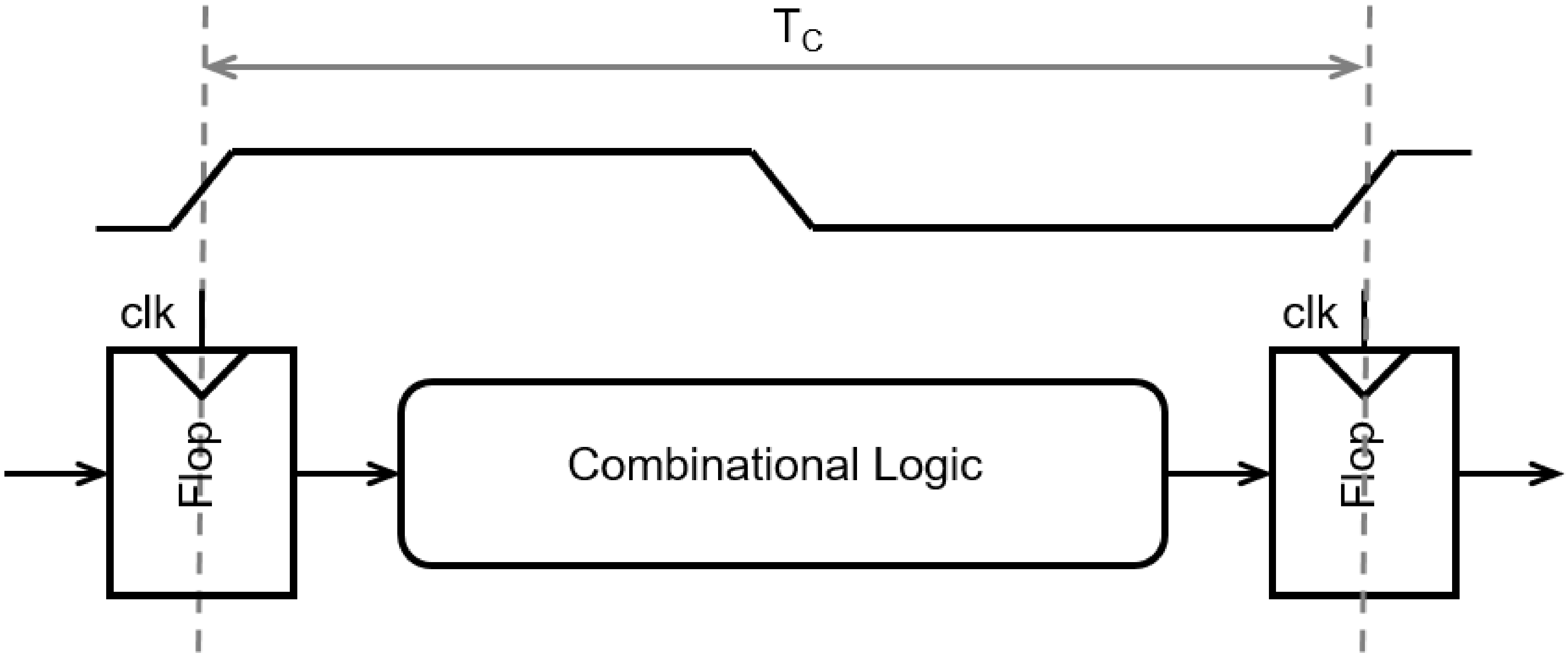
The output of each combinational logic path between two registers must be steady before each clock's rising edge.





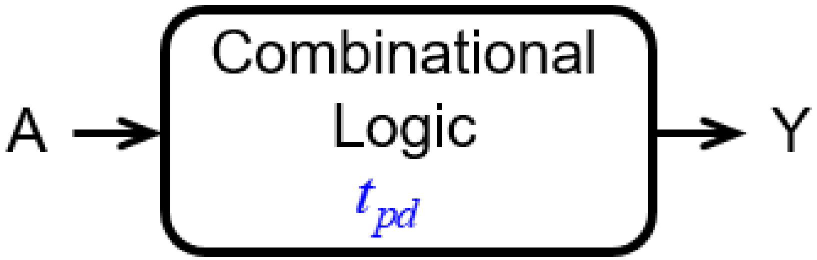
Synchronous Data Movement

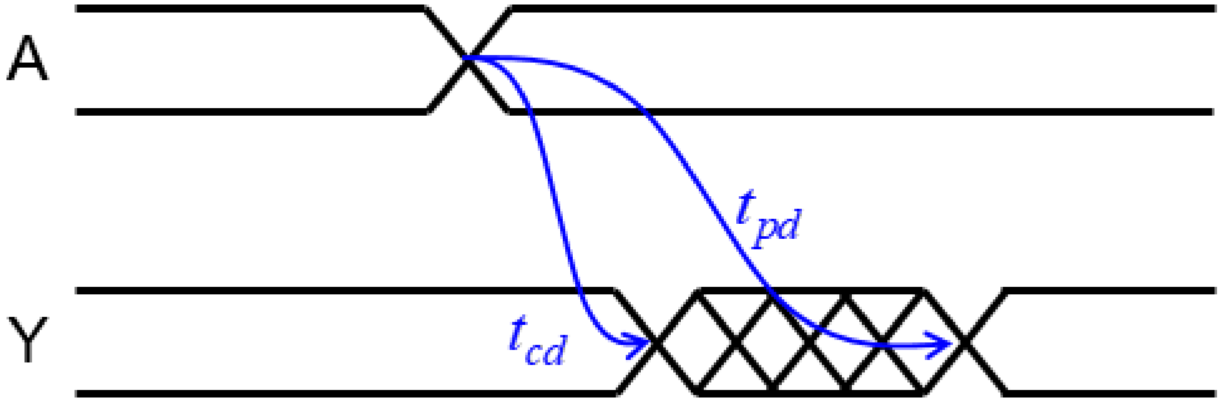
A single flip-flop is used on each cycle boundary. Data advance from one cycle to the next on each clock rising edge.



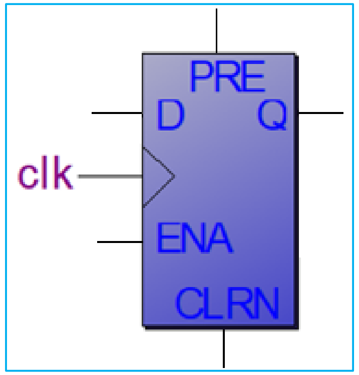
©Hanan Ribo

Combinational timing analysis

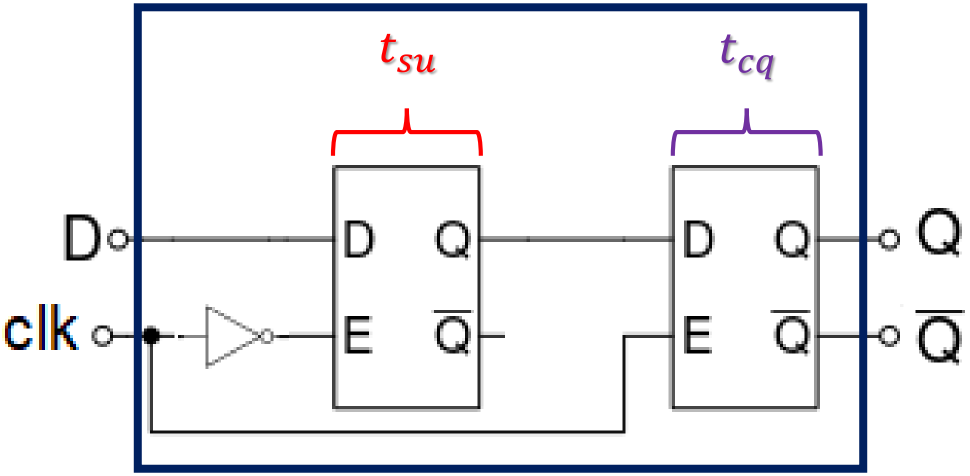




DFF timing analysis

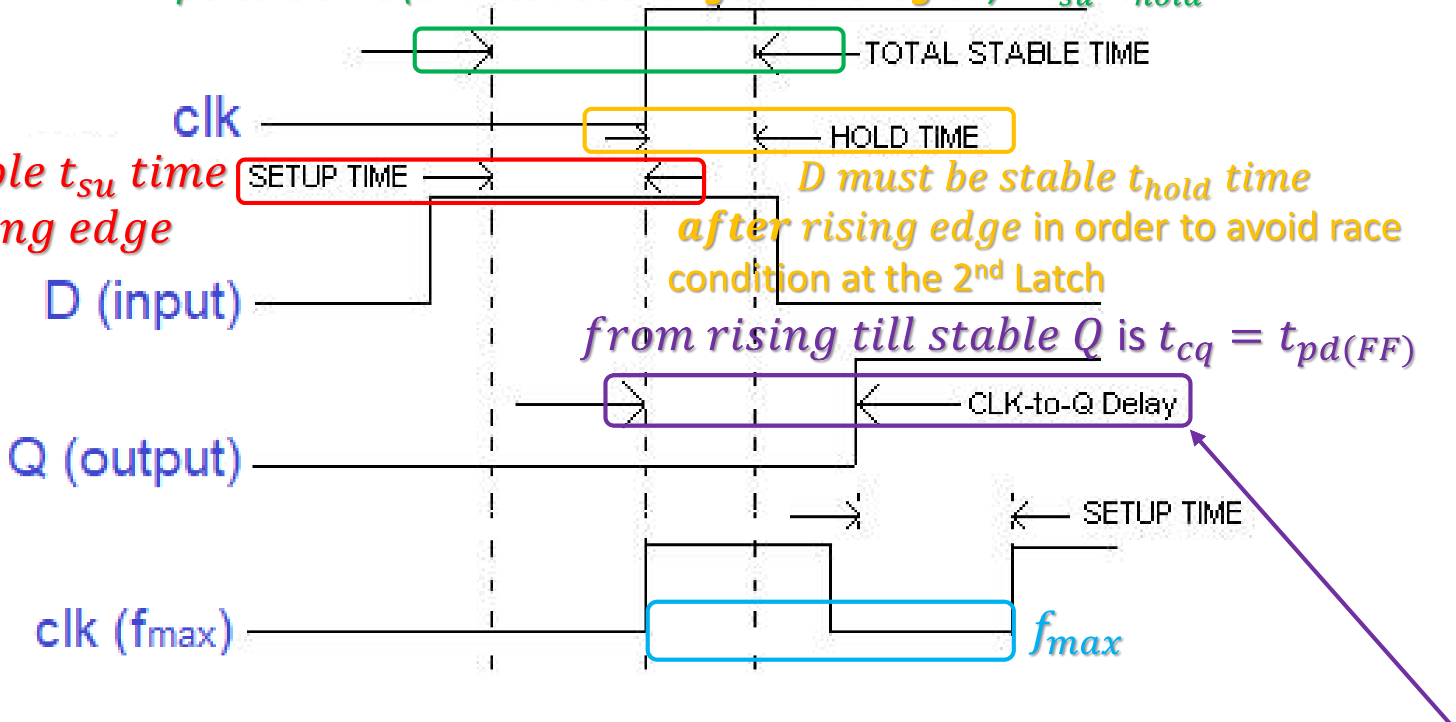


Symbol for a DFF



Positive edge DFF

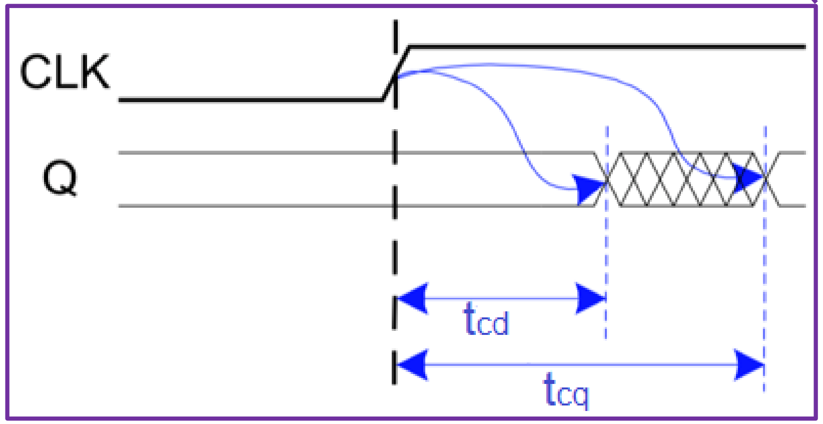
Aperture time (D must not change in this region) = $t_{su}+t_{hold}$



D must be stable t_{su} time ***before** rising edge*

D must be stable t_{hold} time ***after** rising edge* in order to avoid race condition at the 2nd Latch

from rising till stable Q is $t_{cq} = t_{pd(FF)}$



HW requirement:

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

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

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

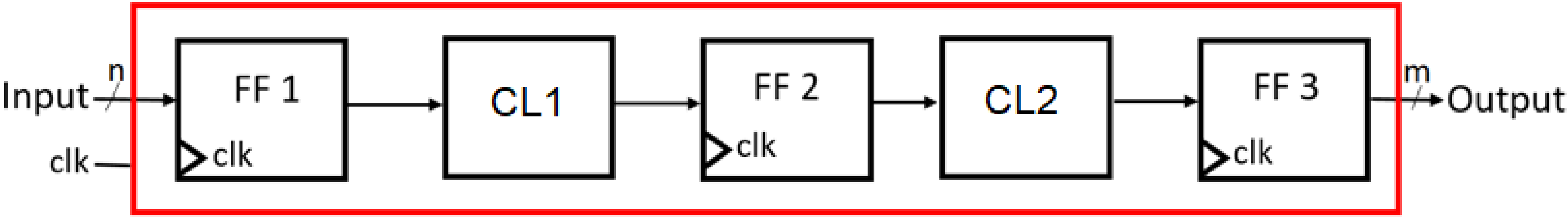
DFF Timing recap

- $t_{cq} = t_{pd(FF)}$: 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)

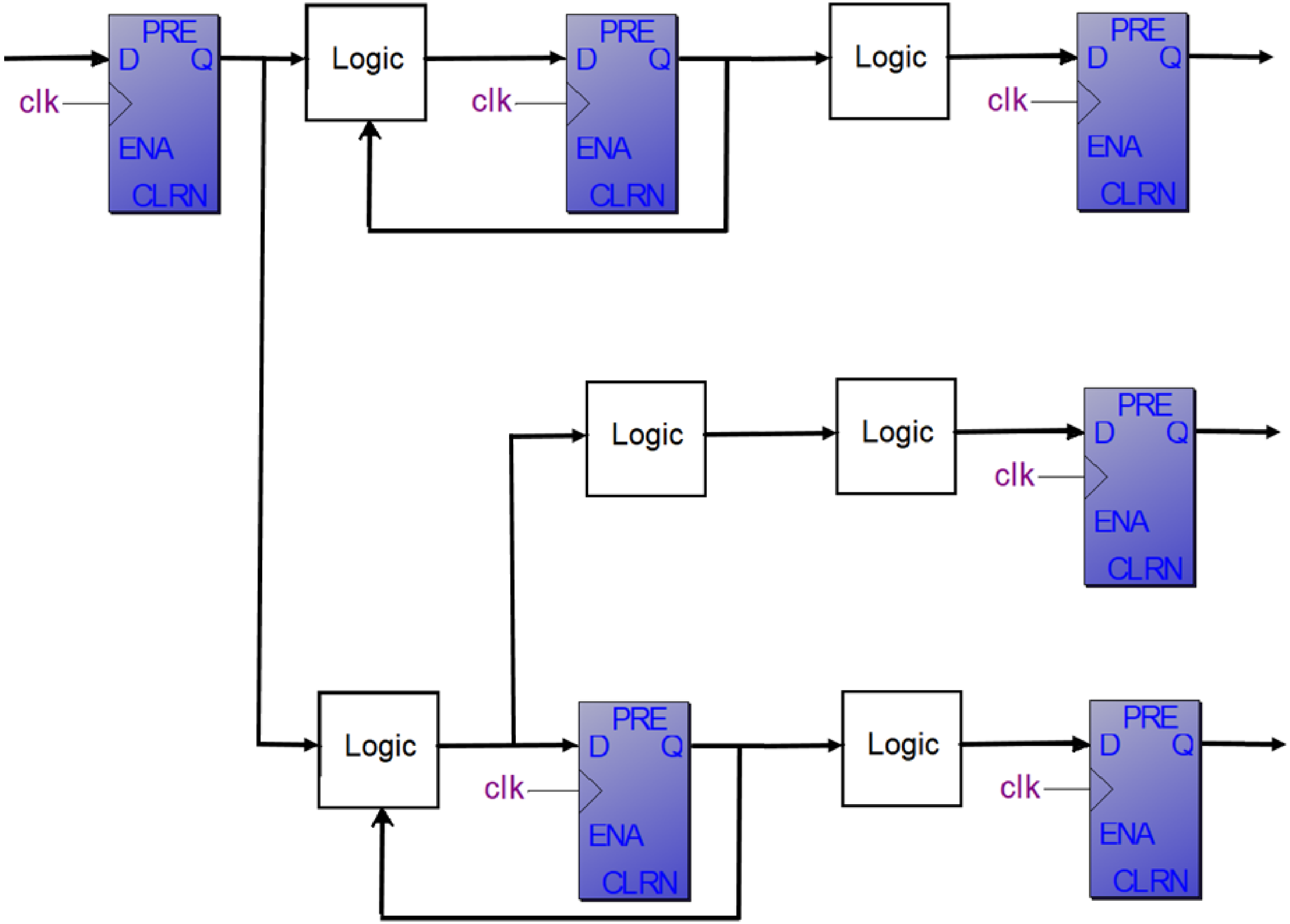
Performance Measurement f_{max}

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) .

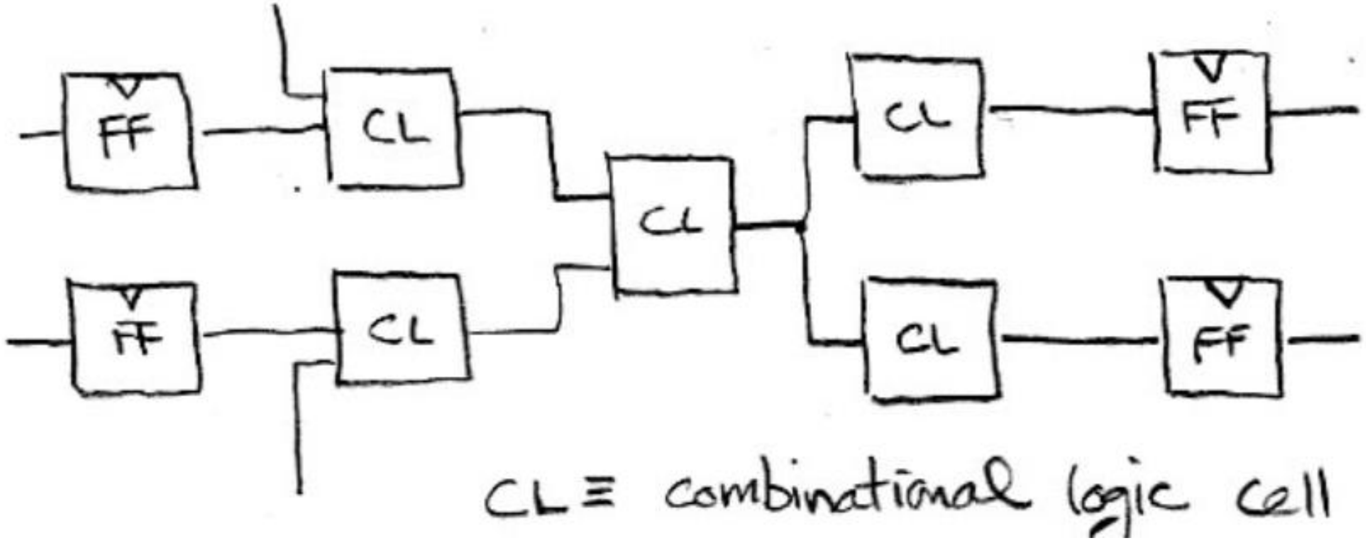


RTL design - example



©Hanan Ribo

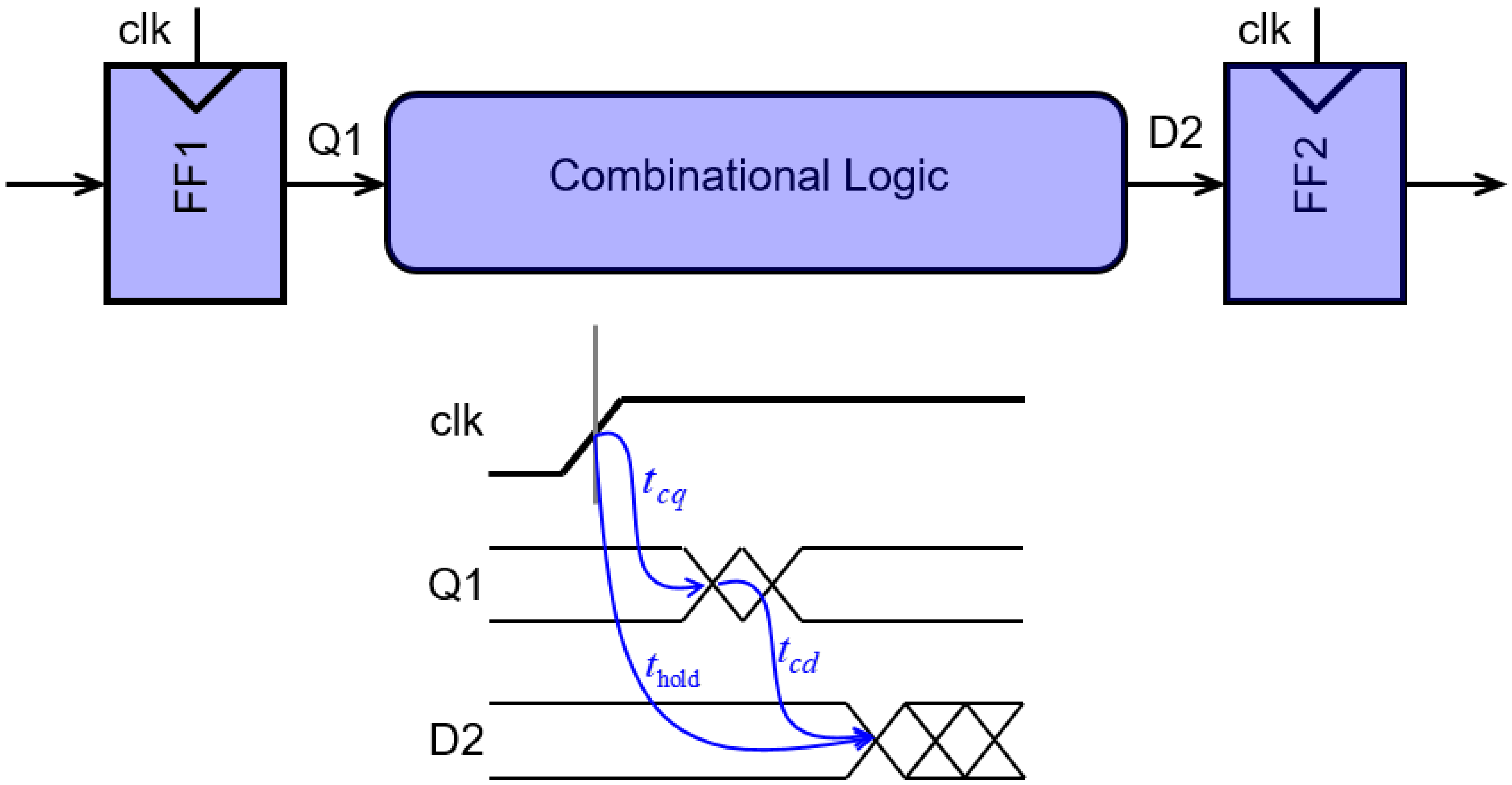
Components of Path Delay



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

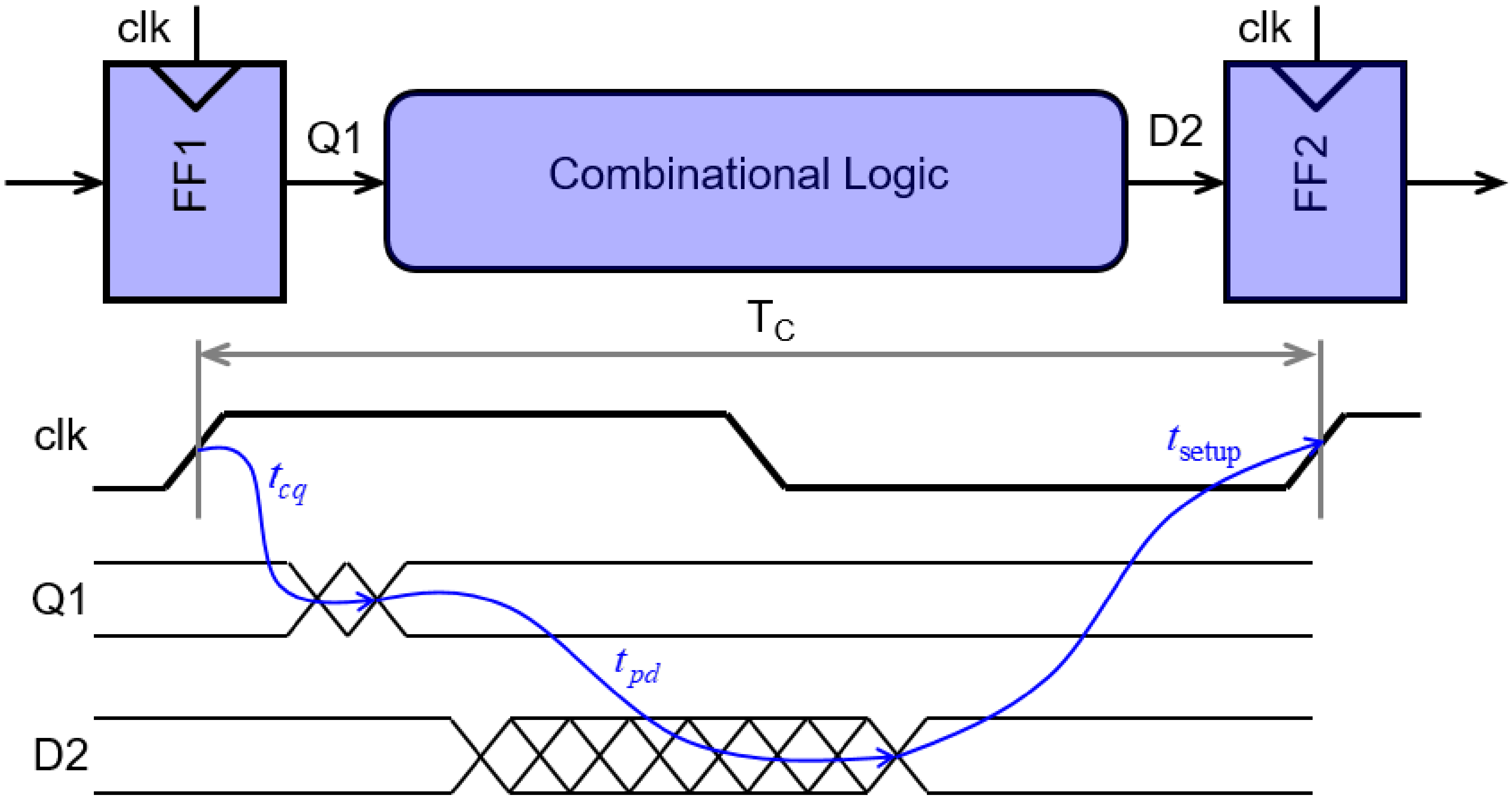
©Hanan Ribo

Timing Delay Constraints - t_{hold} condition



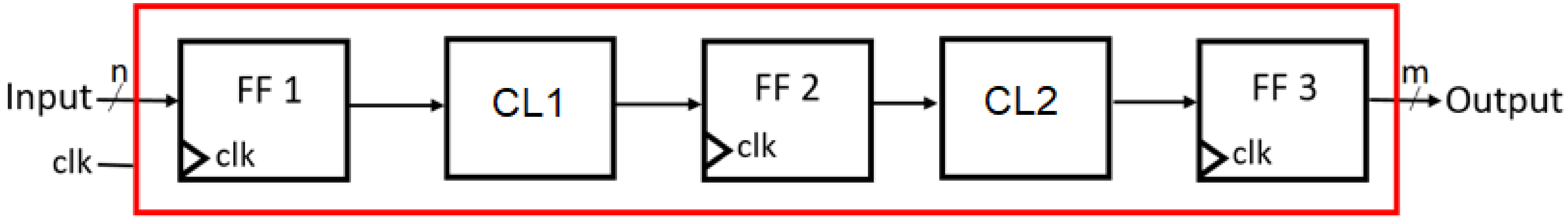
$$t_{cq} + t_{cd} \geq t_{\text{hold}} \quad \Rightarrow \quad t_{cd} \geq t_{\text{hold}} - t_{cq}$$

Timing Delay Constraints - t_{setup} condition



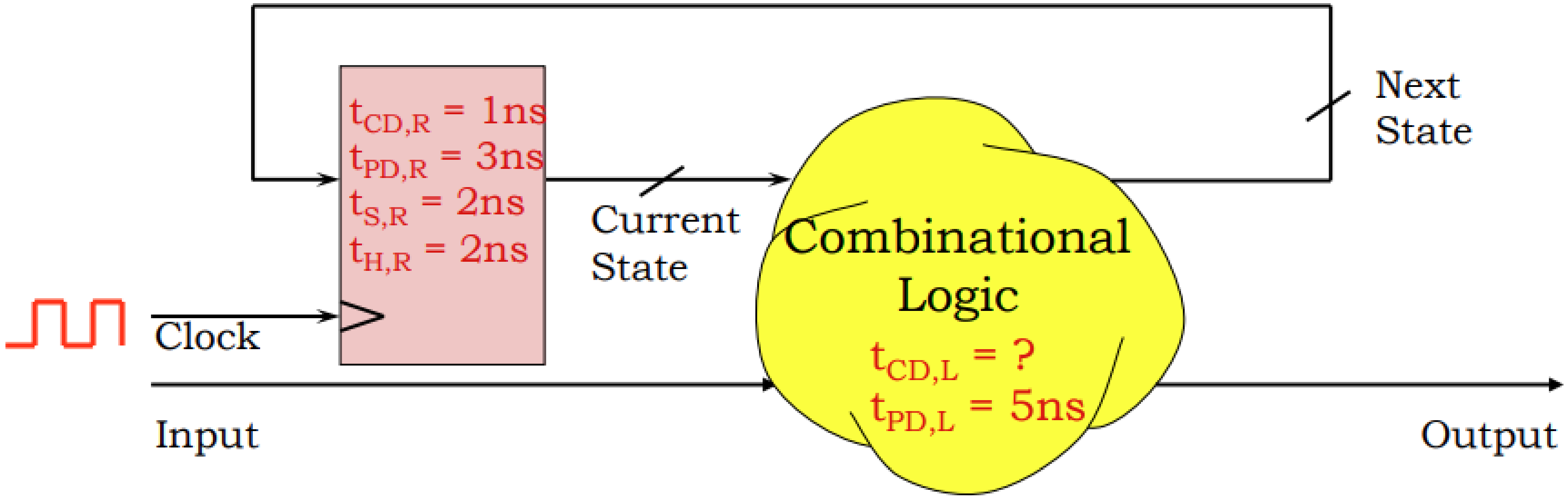
$$T_{clk} \geq t_{cq} + t_{pd(CL)} + t_{setup} \rightarrow t_{pd(CL)} \leq T_{clk} - \underbrace{(t_{setup} + t_{cq})}_{\text{sequencing overhead}}$$

f_{max} calculation – dynamic discipline Obedience



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: $t_{cq} + \boxed{\max_i\{t_{pd(CLi)}\}} + t_{su} = T_{min} \rightarrow \quad f_{max} = \frac{1}{T_{min}}$
   - Hold condition: $t_{cd} + \min_i\{t_{pd(CLi)}\} \geq t_{hold}$

Sequential Circuit Timing



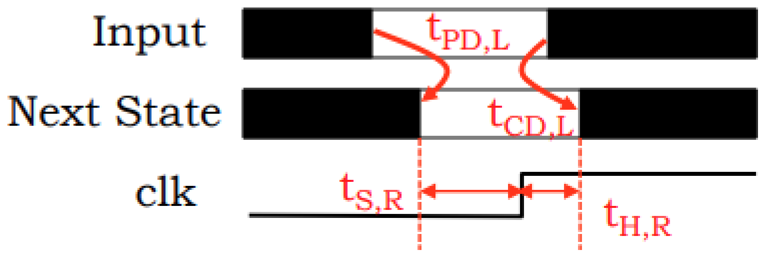
Questions:

- Constraints on t_{CD} for the logic?
- Minimum clock period?
- Setup, Hold times for Inputs?

$t_{CD,R}$ (1 ns) + $t_{CD,L}$(?) ≥ $t_{H,R}$(2 ns)
$t_{CD,L}$ ≥ 1 ns

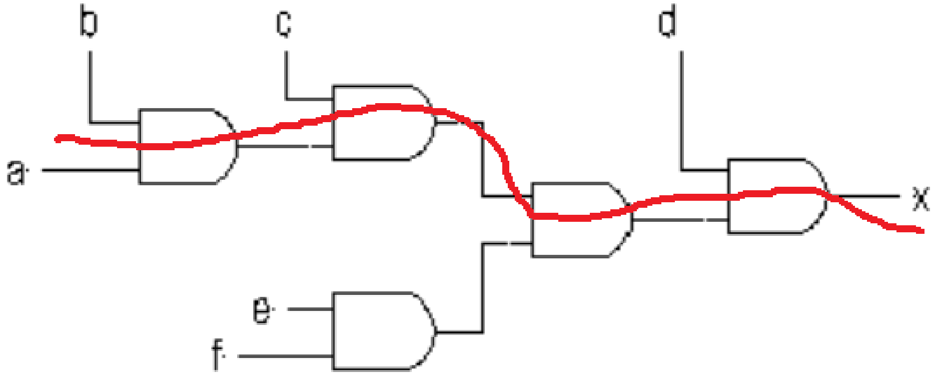
$t_{CLK} \geq t_{PD,R}+t_{PD,L}+ t_{S,R}$ = 10nS

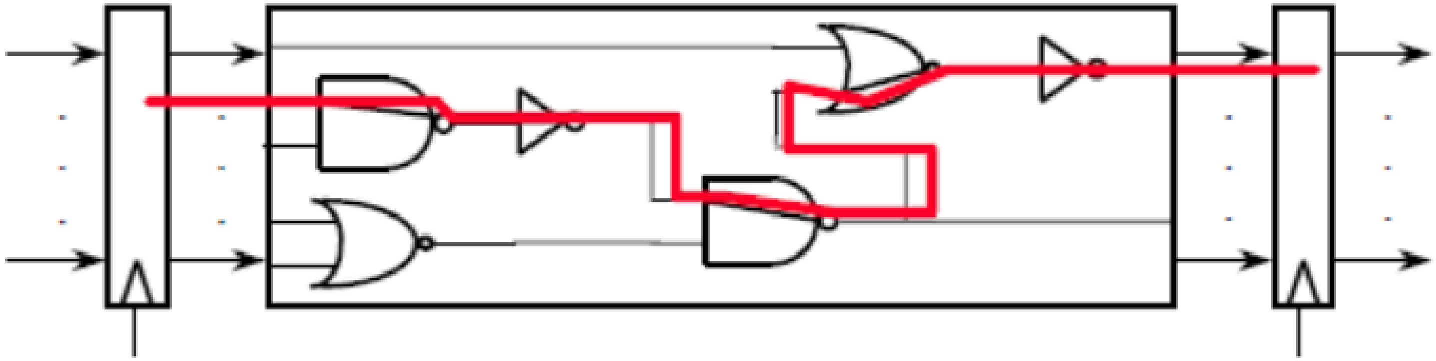
$t_{S,INPUT} = t_{PD,L} + t_{S,R}$ = 7 nS
$t_{H,INPUT} = t_{H,R} - t_{CD,L}$= 1 nS



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?



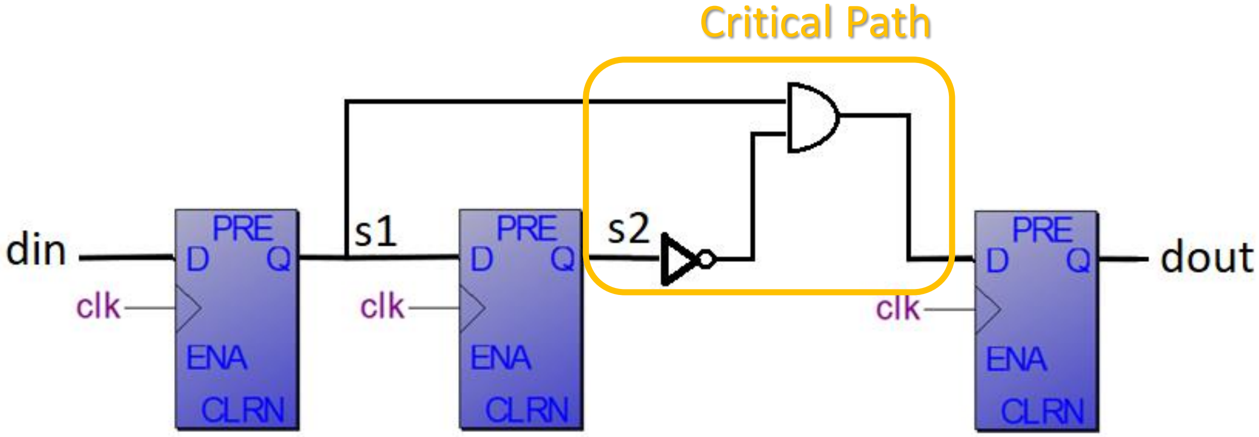


©Hanan Ribo

f_{max} *calculation example*

The Given data:

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



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