It is all about Timing

Note: Some slides having pictures in this lecture have been taken from various websites

Timing Analysis

A mandatory step in digital design process is the analysis, determination and elimination of possible *timing violations* within any design's tolerance range. This fact is becoming more important for nano scale technology and the high speed requirements where greater density of today's MCM, SOC, NoC and ASIC designs that are pushing the limits of speed.

Any design must consider the fact that *component timing* varies from component to component chip to chip and board to board. Each chip/board that is manufactured contains a slightly different combination of fast and slow components that can cause some *timing margin*. The ASIC designer must always account for these variations and check timing violation if any within the ASIC chip and in the board that contains the ASIC using worse case corner analysis.

Dynamic Timing Analysis

Dynamic timing analysis verifies circuit timing by applying test vectors to a circuit under test to *detect timing errors*. It insures that the circuit timing is tested in its functional behavior. Dynamic timing tests *report* the timing errors that the circuit encountered by exercising all possible paths of the circuit. This is similar to simulation.

Dynamic timing analysis uses *worst-case scenario* conditions. Dynamic timing analysis tools (CAD tools) are used to evaluate the circuit using the minimum and maximum *propagation delays* of every component in the design taking into account all kinds of possible variations.

For worst case analysis, the algorithm used, determines when to apply a minimum or maximum delay. This is based on circuit connectivity and checking the setup and hold time constraints.

The disadvantages being:

1) It requires a complete logic simulation, thus computational time issues.

2) The functionality of the circuit has to be known apriori.

Timing....

A circuit board, MCM, SOC, ASIC … contain a variety of devices that cover the range of acceptable *performance specifications*. Naturally some devices will be fast and the others slow, but overall the circuit is expected to be within the specifications.

Numerous factors can affect the *performance* of an ASIC/MCM/SOC/board such as the :

Environment Factors:

1)Devices may be at different temperatures. Thus some devices will run faster than the others.

3)The layout affects the timing due to time constant of the routing area.

3) Process variation can cause the timing of each device to *vary* from typical specification.

Many effects must be taken into account to ensure that the overall design is *reliable* and can be with *high quality and meets its original specification.*

Designed for the worst case

LOAD Condition

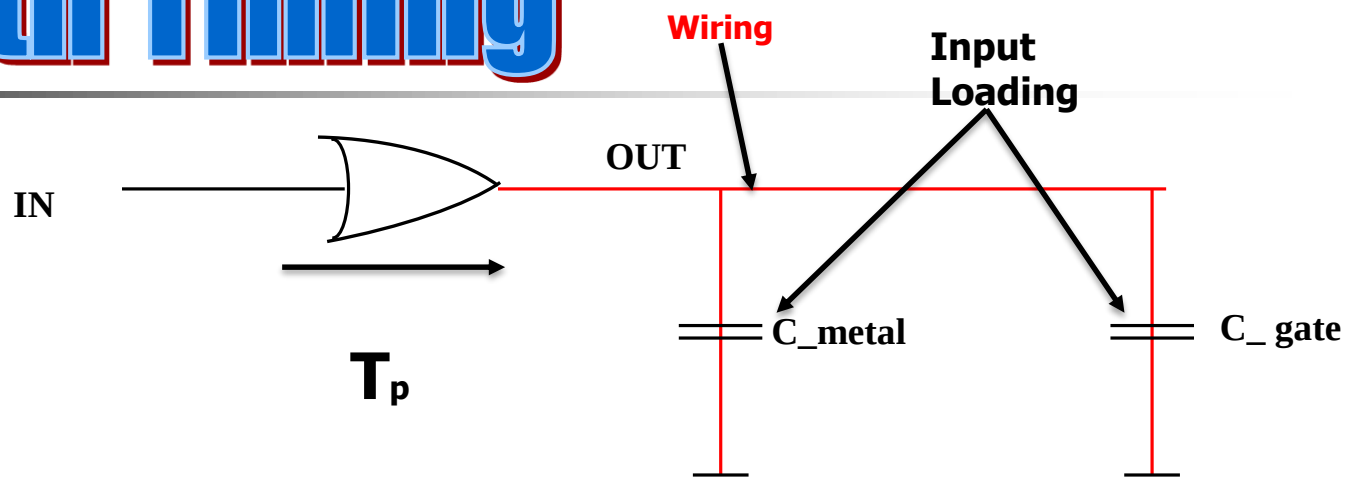
Timing....

Delay **in a circuit is dependent upon:**

- **The intrinsinic delay through each device**
- **The number of loads connected to each net**
- **Temperature**
- **Voltage**
- **Layout (Track length and impedance)**

In submicron processes the loading and environmental effects are *first order* in nature and the intrinsic delay through the device is increasingly becoming a *second-order effect* i.e. the delay due to temperature, voltage and process variation predominate. This is a significant development as the geometries continue to decrease.

Path Timing



Delay and Timing Parameters

Capacitances represent CMOS gate inputs

Delay Equation

Delay = $[T_P + K_1 \Sigma Ni + K_2 M_L] K^*$

T_P = Intrinsic delay in (ns)

K_1 = fan-out derating factor (ns / fan-out)

ΣNi = Sum of input load being driven by the gate (Equivalent unit loads)

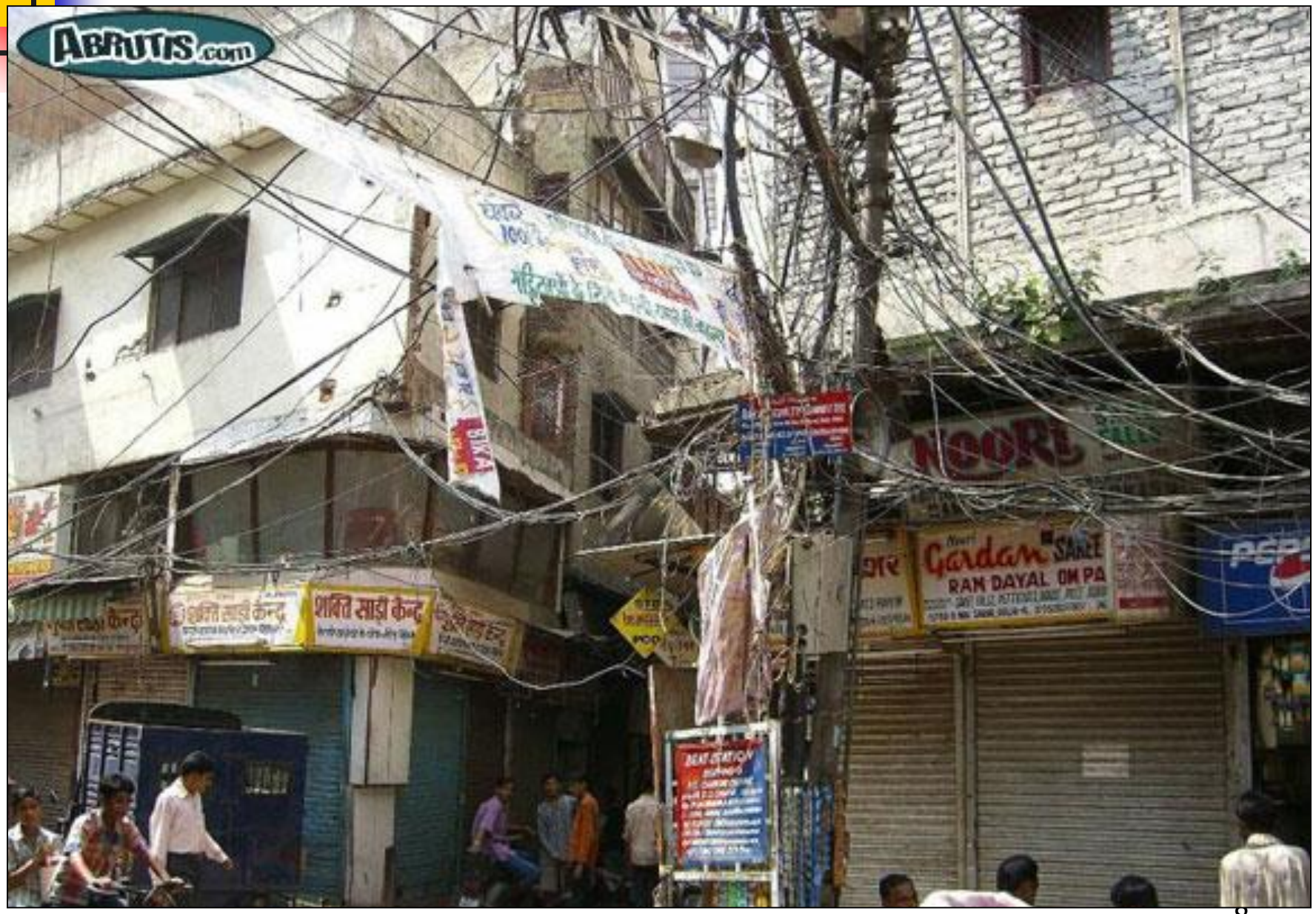
K_2 = Metal load derating factor (ns / µm)

M_L = Metal length being driven by the output (µm)

K* = Composite derating factor due to process, temperature and voltage variation.

- Delay equation applies to both tplh and tphl.

Design checked before implementation



Derating Factor.... (PVT)

K* - Composite Derating Factor

Temperature Effect

td = td/ft = td . Kt

$f_t = (T_J/T_a)^{-M}$ **Tj is the junction Temperature, Ta is the ambient Temperature**

M = 1.5 to 2

Voltage Effect

td = td/(1+ fs) = td . Kv

fs = supply variation factor (%Voltage variation/100)

Process Effect

td = td (1 + o.o1 fp) = td . Kp

fp = process variation factor

K* = Kt . Kv . Kp .

Timing.. an example

Example

In this example we will see how the fan-in and fan-out contribute to delay and later we will show how environmental, process, and voltage variation are incorporated in the determination of the worst case delay estimation.

Determine the overall derating factor, K* for the following condition:
±10% voltage variation
40% process variation. (which are the factors contributing to the regions of ambiguity).
Assume an ambient temperature of 25 °C.
Power dissipation = 1W and
chip and ceramic packaging has thermal resistance θ_{Ja}=35 °C/W

Temperature Variation

Spatial and temporal temperature variation can have drastic effects on signal propagation delay. In CMOS, the variation in propagation delay due to temperature is primarily because of the ***channel current*** of the conducting device. Propagation along a given path for CMOS network can be as much as ***50% slower*** at 125 °C then at room temperature.

Timing....(temperature)

1) Temperature:

Propagation delay increases with increase in temperature.

Lets assume f_t is the temperature variation factor:

$$ft = \left(\frac{Tj}{Ta}\right)^{-M}$$

M = 1.5 to 2 range, T = temp. in ° K

T_J : Junction Temperature, Ta: Ambient Temperature

then, $t'_d = t_d\, kt$ …. $kt = 1/ft > 1$

For ambient temp. T_a = 25 °C, Power dissipation = 1W and chip and ceramic packaging of thermal resistance θ_{Ja}=35 °C/W

The derating factor due to temperature. may be obtained from:

Power removed = $(T_J - T_a) / \theta_{Ja}$ $(T_J - 25)/35 = 1$ T_J = 60 °C

$$ft = \left(\frac{273+60}{273+25}\right)^{-1.5}$$

= (333/298) $^{-1.5}$ = 0.84

K_t = $1/f_t$ = 1/0.84 = 1.18 or increase of 18%

Timing....[voltage]

2) Voltage

The delay increases inversely with voltage:

$t_d \ \alpha \ 1/\mathbf{V}_{dd}$

let fv be defined as the voltage variation factor then:

$\mathbf{t_{dv} = k_v\ t_d}$ where t_{dv}, is the delay with voltage variation

for ± 10% variation on a 1volt supply **fv= ± 10%**

kv = (1 ± 0.01 fv) $^{-1}$

= (1 ± 0.1) $^{-1}$

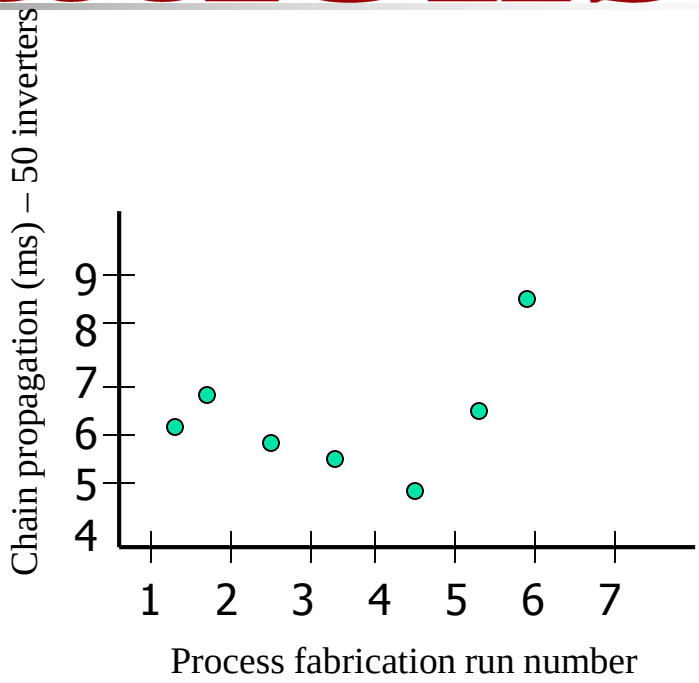
= 1 / 0.9

= 1.11 or 11% increase

Process Variations

Process	Propagation Delay
Fast	14.
Slow	21
Typical	18.

Usually from +20 % to –20 % fluctuation i.e. 40% change



Process parameters are characterized as nominal and corner:

- **The nominal process is the expected process.**
- **Corner processes are the limits of acceptable process parameters.**

Timing....[process]

3) Process Effect

td = td (1 + 0.01fp)

= t_d . kp

fp = process variation factor

= ± 40% i.e. kp =1.4 (or 0.6)

Neglecting other variations then the total maximum effect of the temprature, voltage and process variation is:

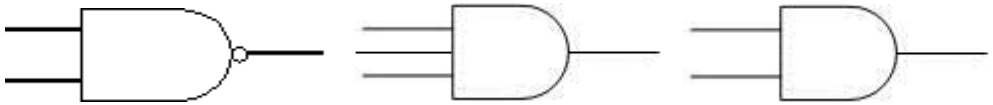
K_{max}= 1.18 * 1.11 * 1.4

= 1.833 or increase of 83.33% in delay for the considered example.

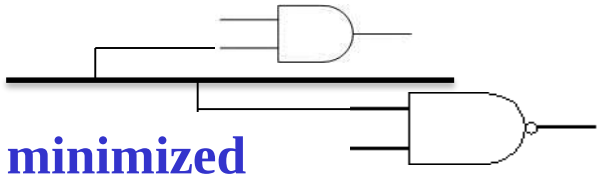
Timing....[different gates]

Apart from external variables discussed, the data dependent delay results from two effects:

1) In CMOS circuits type of the gate used and the number of the inputs is a factor i.e. 2 or 3-input NAND or NOR will have different delay sensitivities. This is why when designs are done with one type like 2-input NAND/INV. The delay variation is more predictable.



2) Signal propagation does not generally occur from a single input to a given output along one path. There are transitions occurring along multiple paths interacting with each other from any number of inputs to the given output.

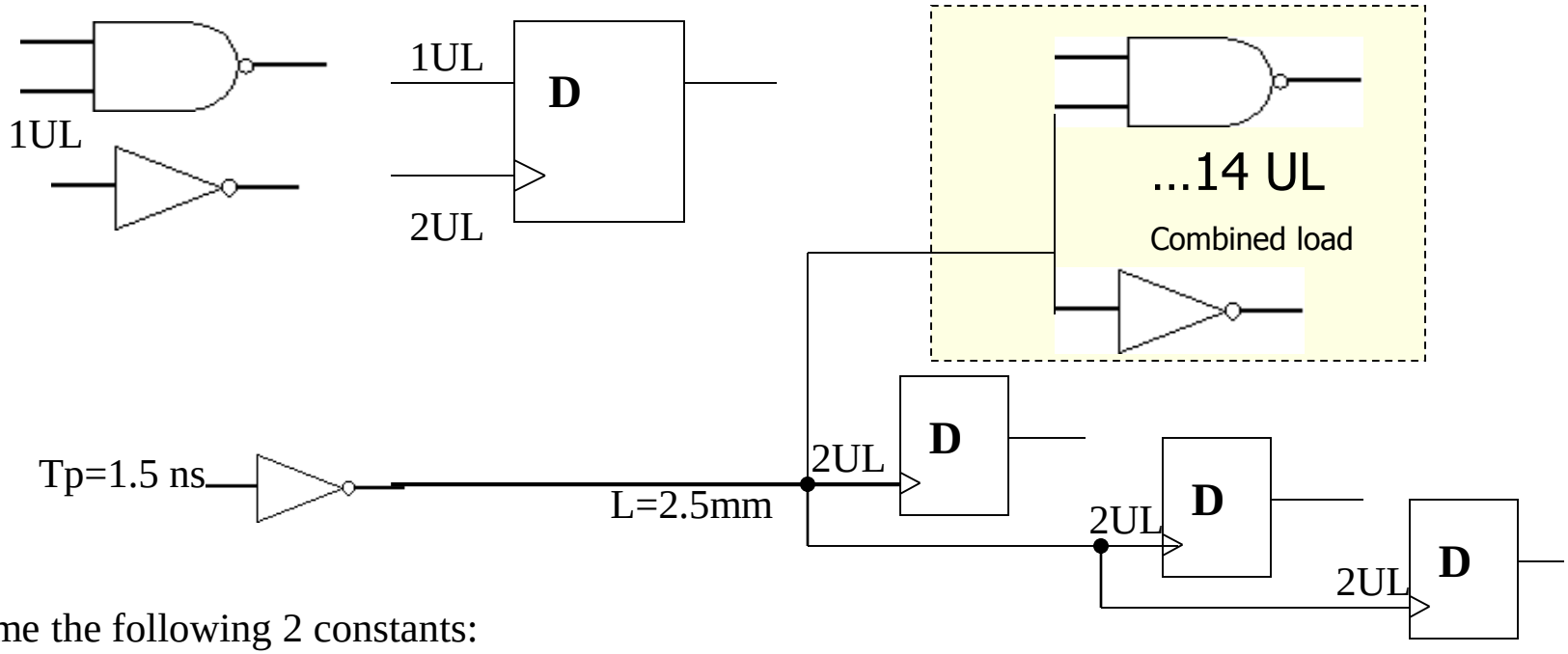


Both of these data dependent delays can be minimized by design. Path length variations and data dependencies are in turn dependent upon logic function implementation.

Example

Determine the delay experienced by the driver circuit given below with a nominal delay of Tp=1.5 ps**. This is a very much simplified delay calculation to show the effects of the routing and fanout**

Assume 1UL(unit load) for basic inverter or 2-input NAND value e.g. 5fF



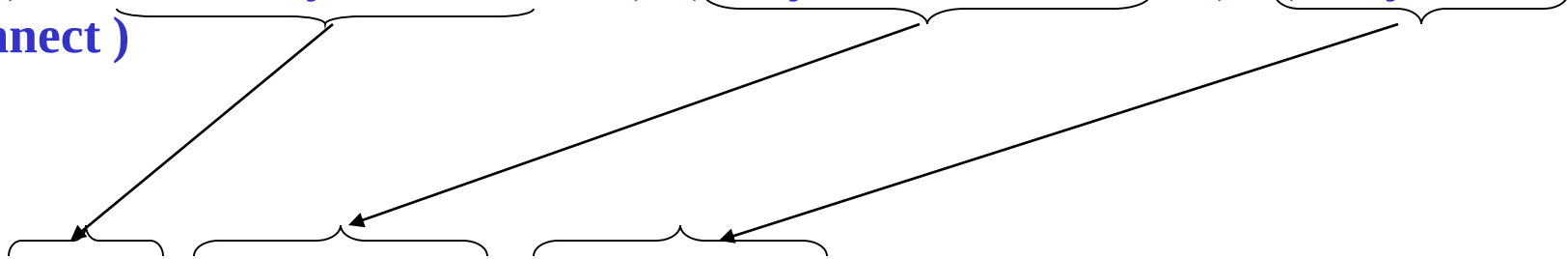
Lets assume the following 2 constants:

K1= 0.06 ns/UL delay factor per unit load connected

K2= 0.08 ns/mm delay factor per unit length wire

Example

Delay =(Nominal delay of the driver)+(delay due to the fan out) +(delay of the interconnect)



Delay = 1.5 ns + $0.06 \Sigma N_i$ + $0.08(l)$

= 1.5 ns + 0.06 x 20 + 0.08 x 2.5 = 1.5 + 1.5 + 0.2 = **2.9 ns**

From the previous example: $V_{DD} = 5 + 10\%$,

$\Theta. = 35°C / W$

Process variation of 40%,

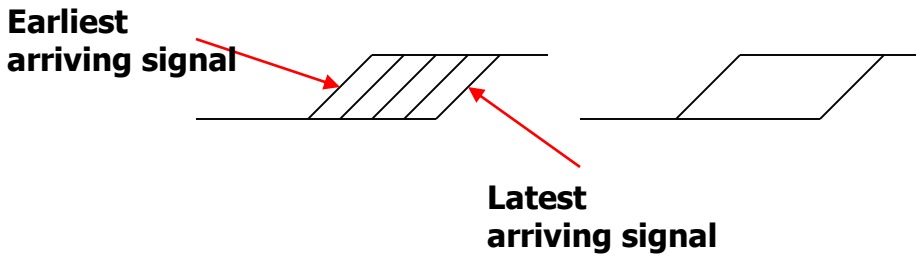
Max. delay = 2.9 x 1.833 = **5.32 ns**

Timing Analysis..... ambiguity regions

Worst Case Timing Analysis

Worst case timing analysis involves determining the regions of *ambiguity*, or the range of time in which a state strength may change.

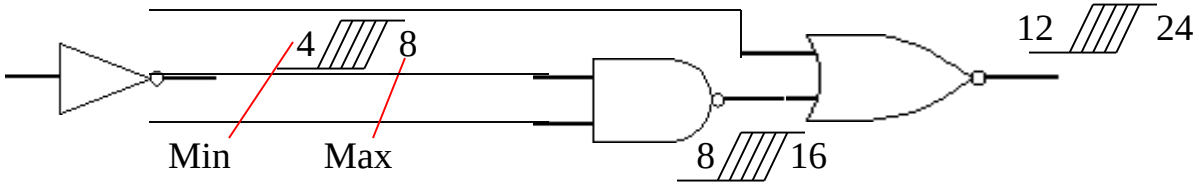
Standard representation of the ambiguity region is by a series of signal edges close together or with a *timing envelope*. The beginning and end of the ambiguity region are measured with respect to a *referenced event such as the clock*. The diagram below shows the earliest and the latest time at which the specified signal makes the *transition* for '0' to '1'.



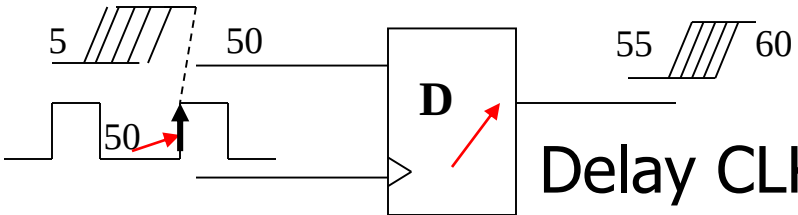
Rising signal ambiguity Envelope

Timing Analysis....

If several combinational logic elements are connected serially, the ambiguity region can grow quite large, and in fact this accurately represents the results of the several worst cases combined. The case when all the components have minimum delay, and the case when all have maximum delay.

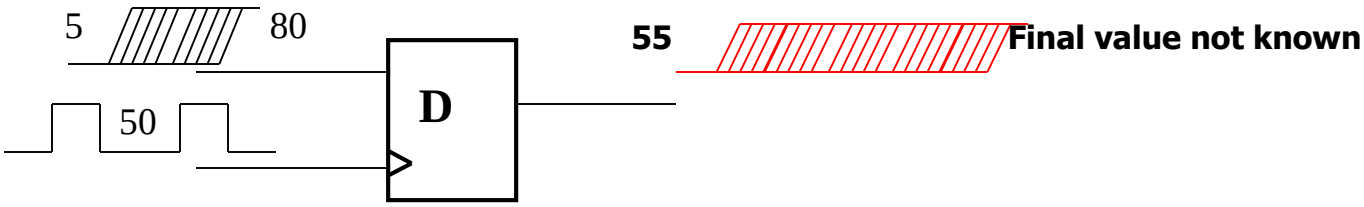


A D-type flip flop can reduce the ambiguity of a signal on its input and this is a major reason for using synchronous elements in a design. Any accumulated ambiguity on the D input is eliminated and is replaced with the combined ambiguity of the CLK input and the output.



Overcoming Ambiguity

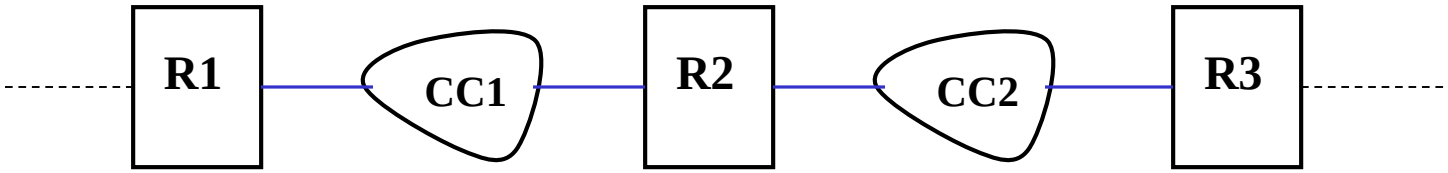
The analysis below shows set up time violation when the D input ambiguity envelope is changing at the time a clock event occurs.

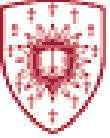


Dynamic analysis has some disadvantages:

It does not find all the errors, because it is pattern dependent. It can only check the timing paths sensitized by the input pattern. If the patterns do not cause an error to occur, the error is not detected.

So, in order to reduce ambiguity and increase speed, the circuit' combinational logic is broken up and placed in between registers.



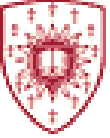


Timing ANALYSIS..notes..

- It is the process of checking the delays of signals within a circuit to see if it operates properly within some timing constraints.These constraints include:

    1. Desired cycle times and duty cycles for all external clocks and internally derived clocks.
    2. Arrival times of all input ports.
    3. Required times of all output ports.
    4. Pin-to-pin propagation delay.

- Clocks referenced by constraints can be real clocks or clocks used only for timing constraints.
- Timing analysis calculates:
    1. Arrival times and required times of all critical nodes
    2. Slack time of critical nodes of the circuit



Timing….Violations, Slacks

Arrival Time and Required Time

- Timing analysis uses the constraints to calculate the time at which each signal becomes valid, based on its sources (arrival time) and the time at which a signal is needed by its destination (required time).
- The arrival time for each signal is the worst case (longest) time, considering all signals that affect the delay.

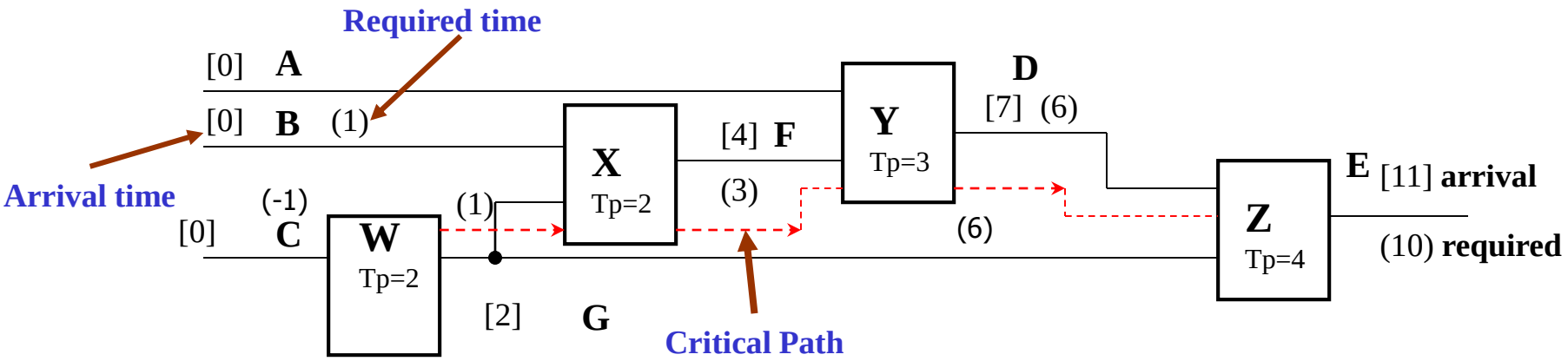
  The required time is the worst case time considering the requirements of all destinations of the signal.

Slack Time

- It is the difference between the required time and the arrival time.
- A positive slack indicates that the circuit will meet its requirements.

Static Timing Analysis....

[ ] Left to right (arrival time)
( ) Right to left (required time)
Use the critical path to evaluate



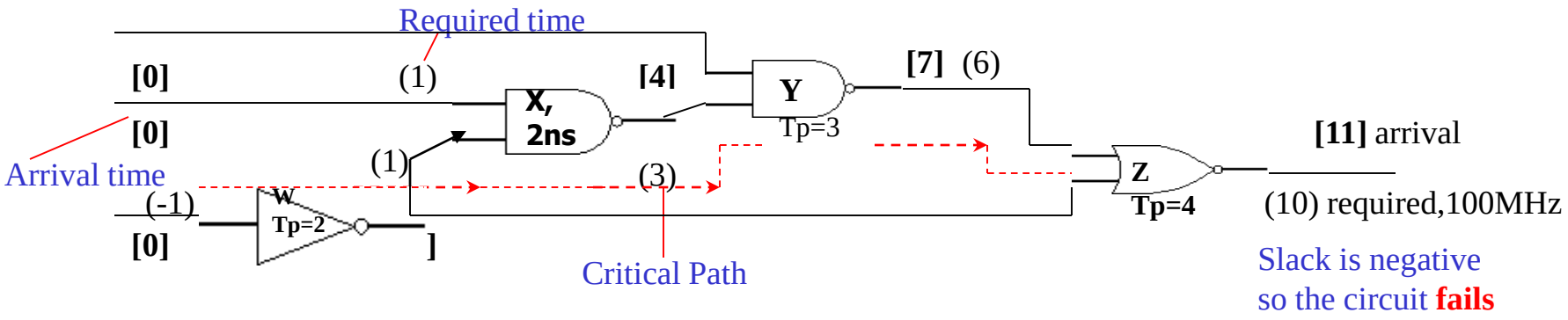
Signal	Arrival Time	Required Time	Slack
A	0	3	3
B	0	1	1
C	0	-1	-1
D	7	6	-1
E	11	10	-1
F	4	3	-1
G	2	1	-1

Using the longest path, [11] is the arrival time with (10) of required time. Circuit fails, so the signal 'C' should arrive earlier to reduce delay in the critical path

Static Timing Analysis

Arrival Times: **Start with arrival times from primary inputs, then move forward to the direction of the o/p using all the paths.**

Required Times: **Starts with o/p with constraints and move back to the primary input, dictated by the max. frequency.**



Solution: **Signal C should arrive 1 ns earlier or reduce delays in the critical path (i.e. re-design building block or circuit).**

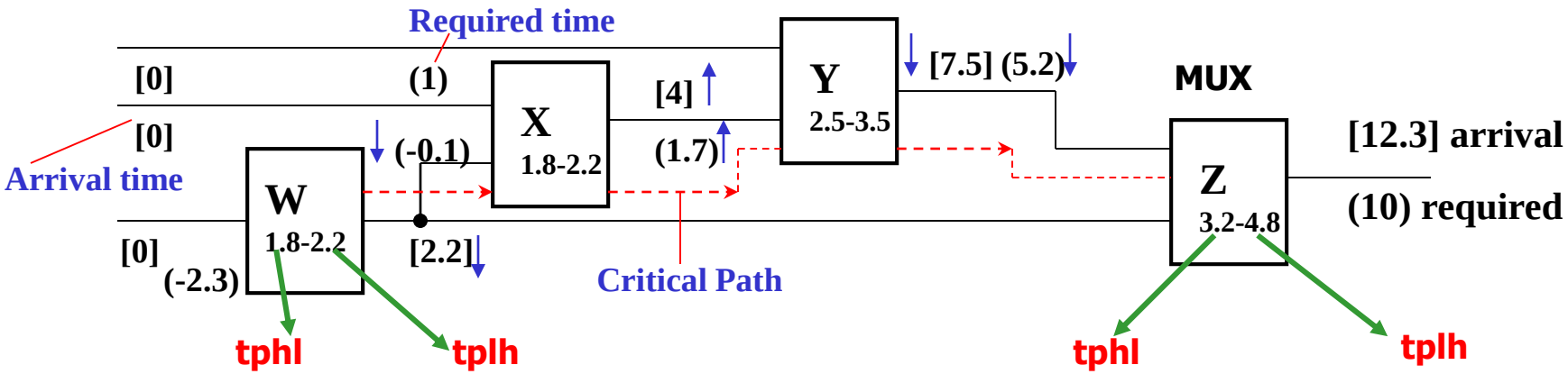
At this stage we have not taken the propagation delay of the path into account.

Static Timing Analysis....

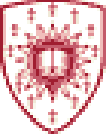
The previous slide propagation delays are averages for propagation delay, t_{pd}. However, t_{phl} and t_{plh} values can be different. To be more accurate, analysis has to be based on inversions and the real numbers of 2 input conditions.

a) Giving o/p of t_{phl}

b) Giving o/p of t_{plh}



This is only for up and down switching not taking the tplh and tplh difference into account.



TIMING PARAMETERS....

- Combinational circuits have two timing parameters, t_{pHL} and t_{pLH}.
  They are calculated according to the delay equation of each component.
- Sequential circuits have a number of timing parameters and timing constants:

  CLK to Q(Q) delay t_{CQ}
  Setup and hold times t_h t_{SU}
  Set to Q(Q) delay t_{SQ}
  Reset to Q(Q) delay t_{RQ}
  Minimum clock pulse (high and low). T
  Minimum set (reset) pulse width.

Set up time:

It is the minimum amount of time that the Data should be held steady ***before*** the clock event that captures the data.

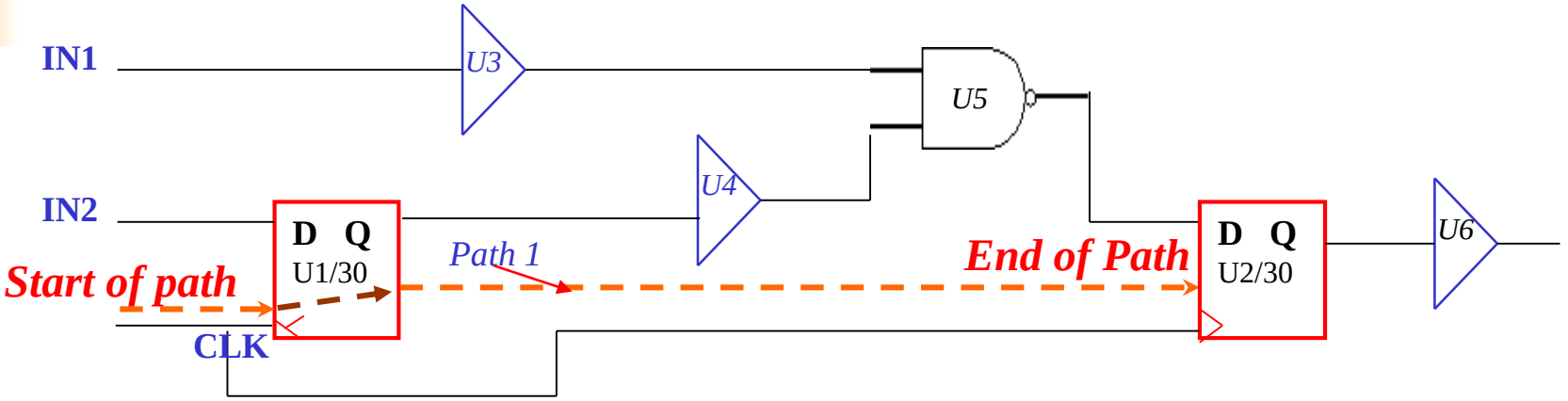
Hold time:

Is the minimum amount of time that the data should be held steady ***after*** the clock event that captures the data. HOLD time is measured with respect to the active CLK edge only. It is to ensure the correct data transmission during switching of the clock. There is a finite delay for transmission gates to switch off and on. In the meantime it is necessary to maintain a stable value at the input

Static Analysis....path

- **A path *starts* at a source flip flop(or at a primary input) and terminates at a destination flip flop.**
- **A path does not go through a flip flop unless it is transparent latch. Thus if a signal goes from Register A to Register B and then to Register C, the signal contains two paths.**
- **A path *terminates* when it encounters a clocked device.**

Static Analysis: introducing the registers



Path 1 goes from CLK to Q on Register U1 through U4 buffer and U5 NAND gate to the D input of U2. Path analysis adds up the delay along this path and compares the total with the clock arrival time at U2.

Example: Assuming:

U1 has tpd from ***CLK → Q of 30 ps***

U4 has tpd ***of 15 ps***

U5 has tpd ***of 15 ps***

Since the signal must arrive in time to satisfy set up time, on the destination register U2, we have to add up the set up time to the total delay (say 20ps). This yields a total delay along path 1 of 80 ps. If the clock speed is faster than 80 ps, the setup time constraint is violated.

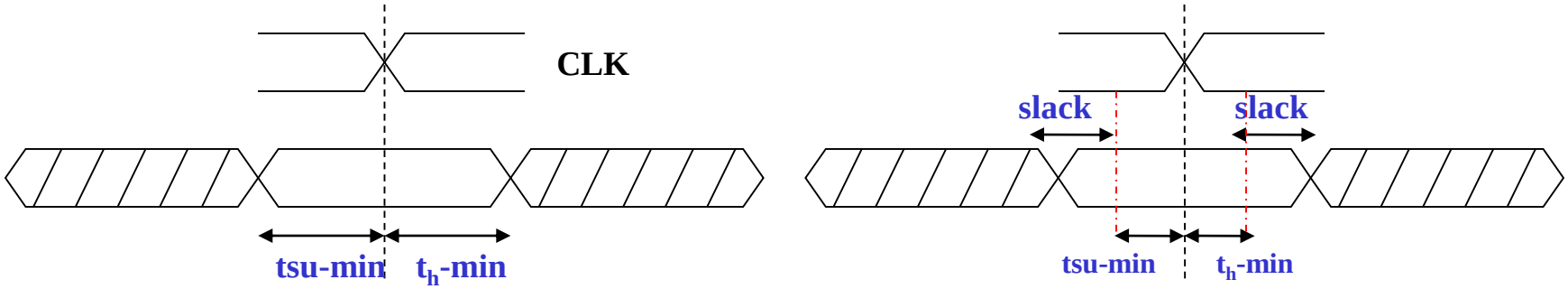
Setup clocks checks are made using the minimum delay on the destination register's clock line, and the maximum delay on the data line. "***Pulling***" the clock edge "***in***" as early as possible and "***pushing***" the data arrival time "***out***" as late as possible creates the worst case scenario for checking ***set up time*** and vice versa for ***hold time.***

Set up and Hold Time

To ensure proper flip-flop operation, certain timing restrictions must be placed on when flip-flop input signals can change value. The restrictions are known as

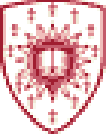
1) Setup time 2) Hold time

- **For the edge triggered FF(flip-flop) to function properly, input signals that determine the next state must be stable for a period t_h after the clock transition. Setup and hold times are not equal and are usually positive values but can be negative as well.**



Slack

Slack is the amount of time to spare before a signal violates a set up or hold constraints. Path that have a **negative** slack time have already violated a set up or hold constraint and require attention. Paths that have a small slack are the speed **limiting paths** in the design.

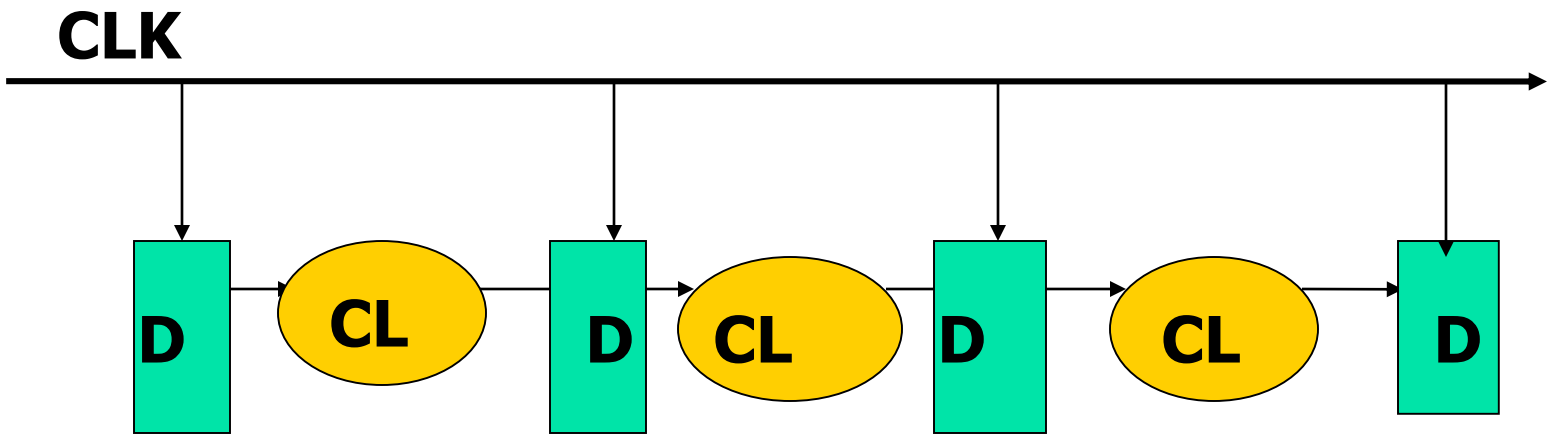


The clock

- For any sequential circuit to work properly, a switching event must synchronize memory devices. For Synchronous systems this is the clock edge.
- The clock signal is distributed globally through all parts of the circuit reaching all clocked devices.
- Although clock distribution is usually done in such a way as there is no difference between the clock signal arrival at different part of the circuit, timing delay is inevitable.
- Clock skew is the difference in time between two clock signal arrivals (usually at two adjacent locations, registers)

Synchronous Systems

- All events are controlled by the clock
- The clock insures that physical timing constraints are met.
- The clock provides a time base for ordering of events
- Advantages: Design is simple
- Disadvantages: constrained by slowest CL



Timing Uncertainty & Clock

- The advantage of a synchronous method is to regulate the flow of data throughout the system. However, this synchronizing approach depends on the ability to accurately relay a clock signal to millions of individual clocked loads.

- Any timing error introduced by the clock distribution has the potential of causing a functional error leading to system malfunctioning. Therefore, the timing uncertainty of the clock signal must be estimated and taken into account in the first design stages. The two categories of timing uncertainties in a clock distribution are skew and jitter.

A Clock Distribution Network (CDN)

A set of interconnections (usually with buffers) that delivers reliably a time reference, clock signal, to every register element in a synchronous digital system.

The challenge:

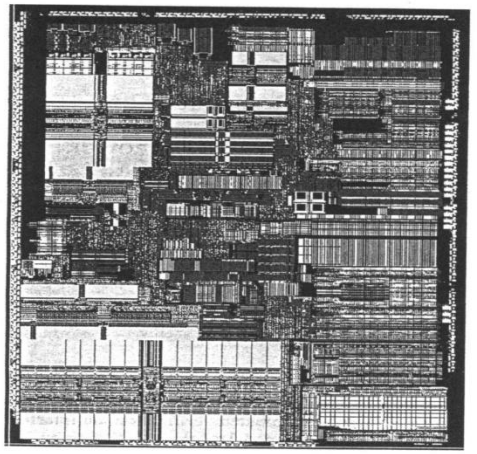
How to connect to all register elements?

How to satisfy the time constraints?

How it will affect the system performance?

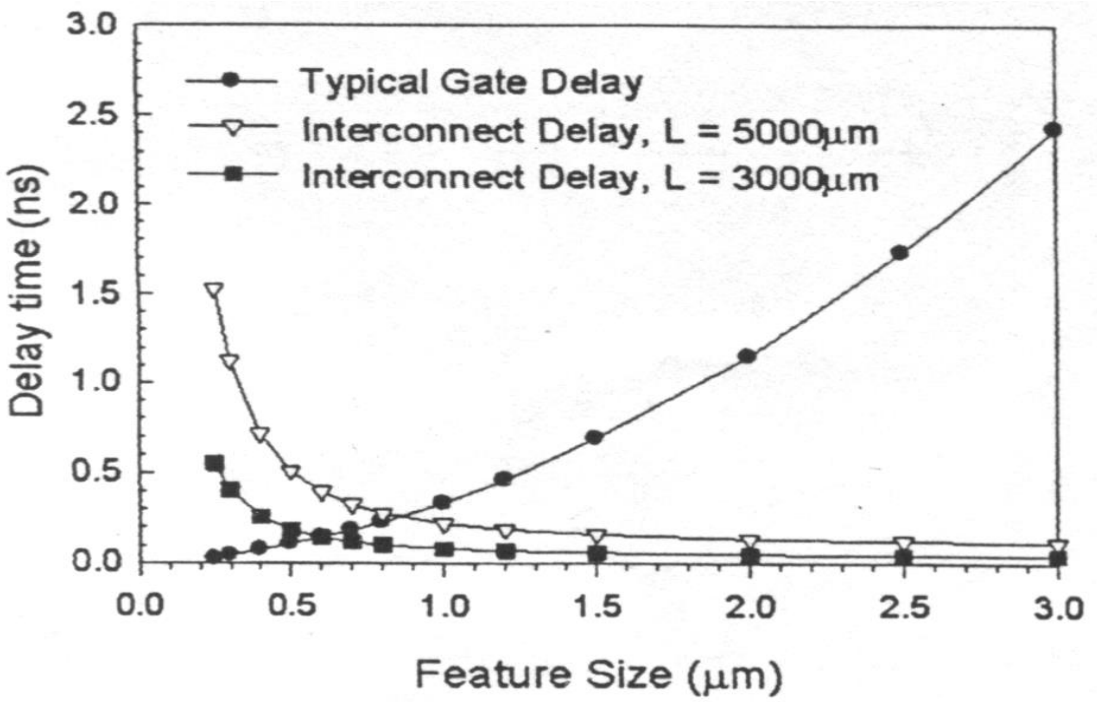
How to connect the set of registers R with minimum wire length and get an equal delay between the registers?

PowerPC microprocessor
32,000 master/slave latch



Reduction of the feature size

Increase in the influence of the interconnect delay on system performance



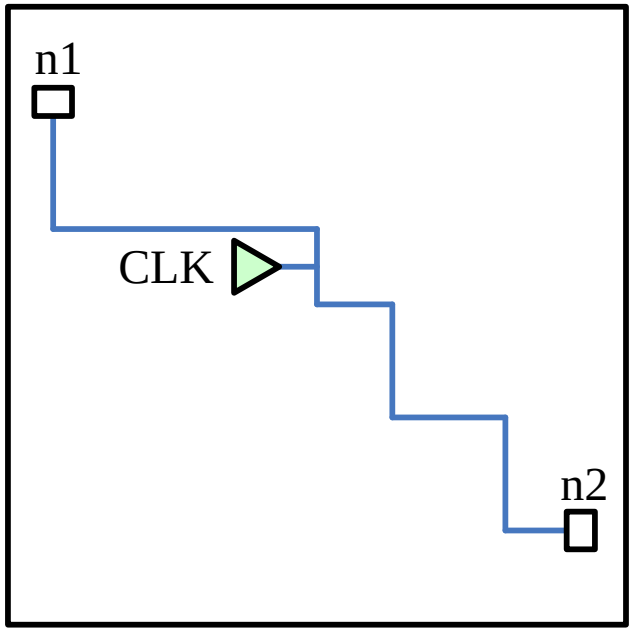
Skew

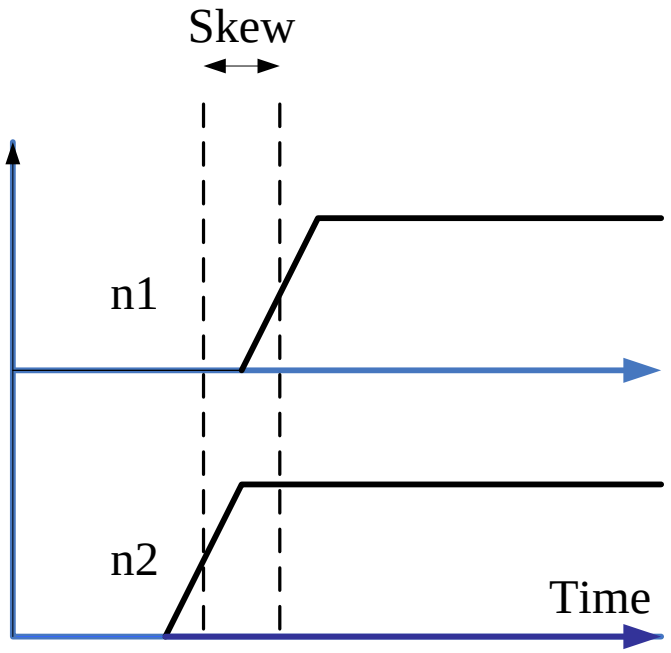
The difference in the arrival times of the clock signal to all registers in a synchronous digital system

Skew

- Clock skew refers to the absolute time difference in clock signal arrival between two points in the clock network. An example of clock skew is shown in the next slide where the clock signal at n1 precedes the clock signal at n2 by an amount= “skew”.
- Clock skew generally is caused by mismatches of either device or interconnect within the clock distribution or by temperature or voltage variations around the die or sometimes by load variations.
- Although skew generally refers to the timing difference between any two points on a die, a more important metric for current and future synchronous system is the components of clock skew.

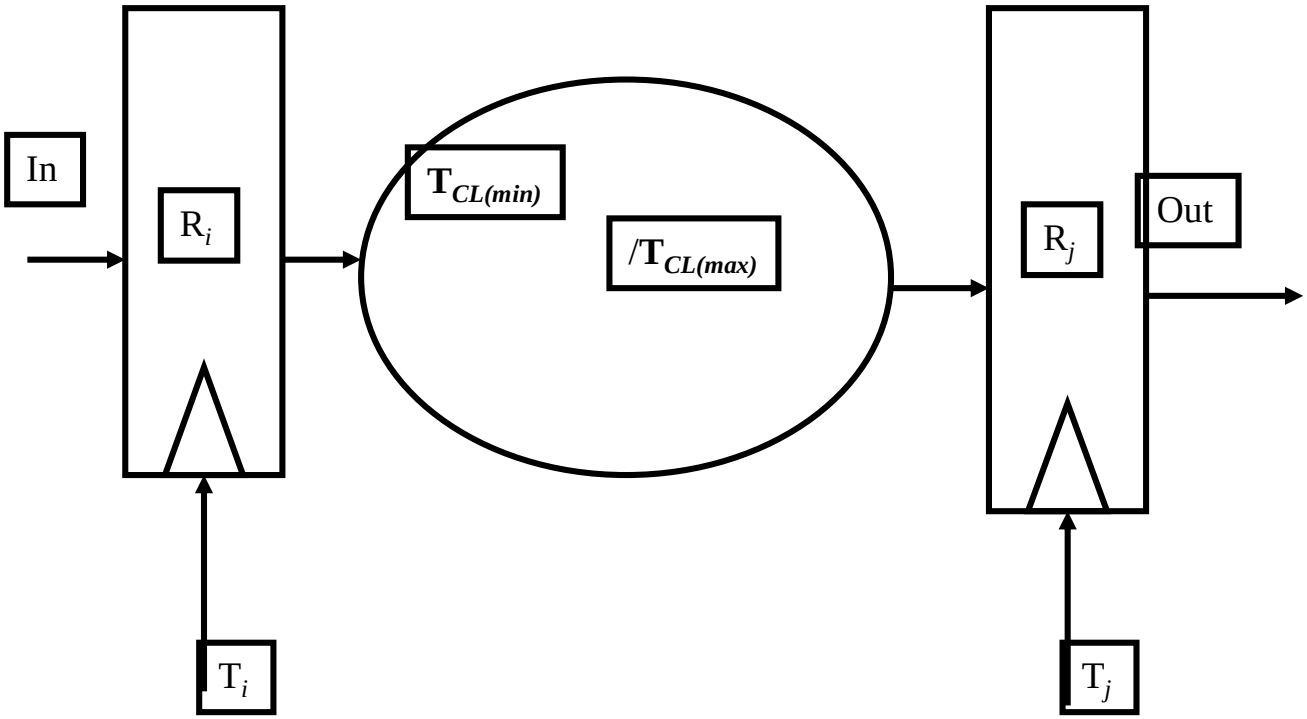
A portion of a clock tree including a clock source and two nodes





SKEW (Min, Max)

Every component in the system will have Max and Min delay time, due to variation of physical components and PVT variation. This applies to SKEW as well.



Skew variation

- There are two components to the clock skew: the skew caused due to the static noise (such as imbalanced routing) that is *deterministic* and the one caused by the system device and environmental variations that is *random*. An ideal clock distribution would have zero skew, although in practice it is sometimes beneficial to intentionally skew the clock to speed-up specific paths in the design.

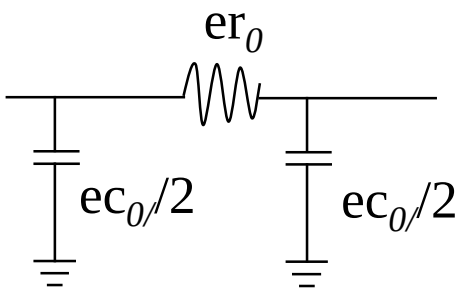
- The designer can control skew influence by 3 methods:

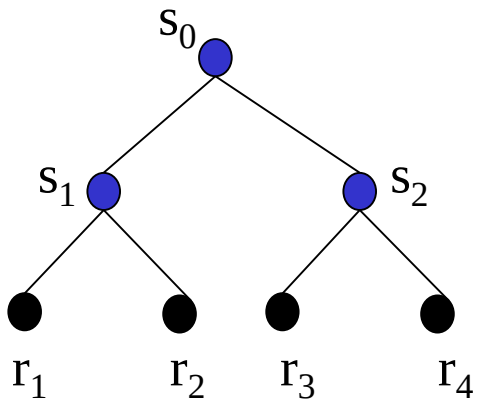
1. Route the data and the clock opposite to each other,
2. Build enough contingency in the clock period,
3. Insert some delays (say, inverter, buffer) between adjacent latches if there is no logic, so that the signal will not overtake the clock.

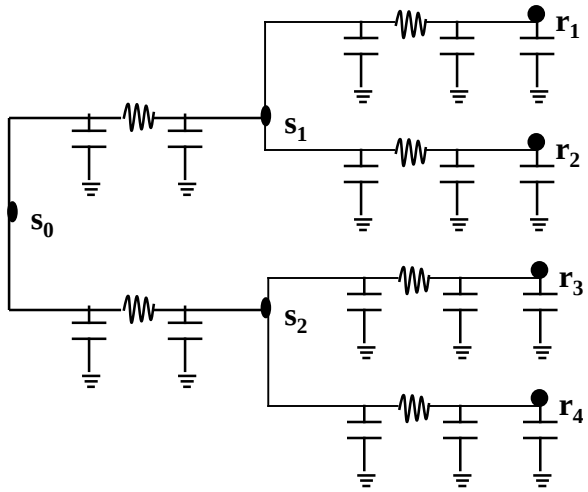
Delay model of the CDN

Elmore Delay model

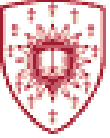
It takes into account the interconnect resistance and capacitance and the capacitance of the registers







$$\text{Delay}(w, v) = \Sigma\, e_i\, r_0 \left(\frac{e_i\, c_0}{2} + C_i \right) \quad , \text{ for every } e_i \in \text{Path}(w, v)$$



Parameters influencing Clock Distribution Networks

- The Clock Network designer can influence the skew and the jitter by:
- Shape of the Clock Network Distribution (CND) and the algorithm used to design it. (Such as minimum Skew algorithms).
- CDN Architecture, Levels of distribution, Levels of Buffering and the type of buffers used.
- Load distribution.
- Single ended CDN, Differential CDN, single plane, Double plane, interconnect material selection.

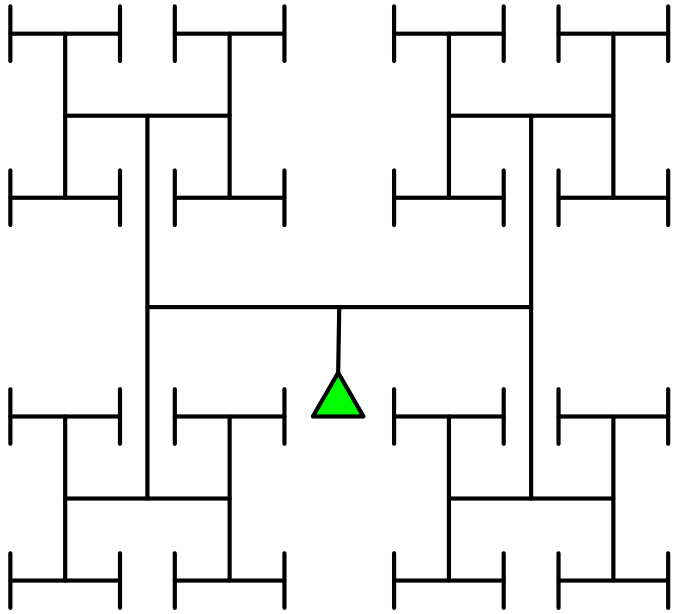
Example of Clock Network– The H-tree

- There are a variety of Clock Distribution Networks that target zero skew (low skew) deterministic clock network.

The H-Tree is such a Network.

This network has a symmetrical distribution of the clock tree.

The set of clock loads is divided into two symmetric sets recursively by vertical-horizontal lines consecutively until all sets become one



Jitter

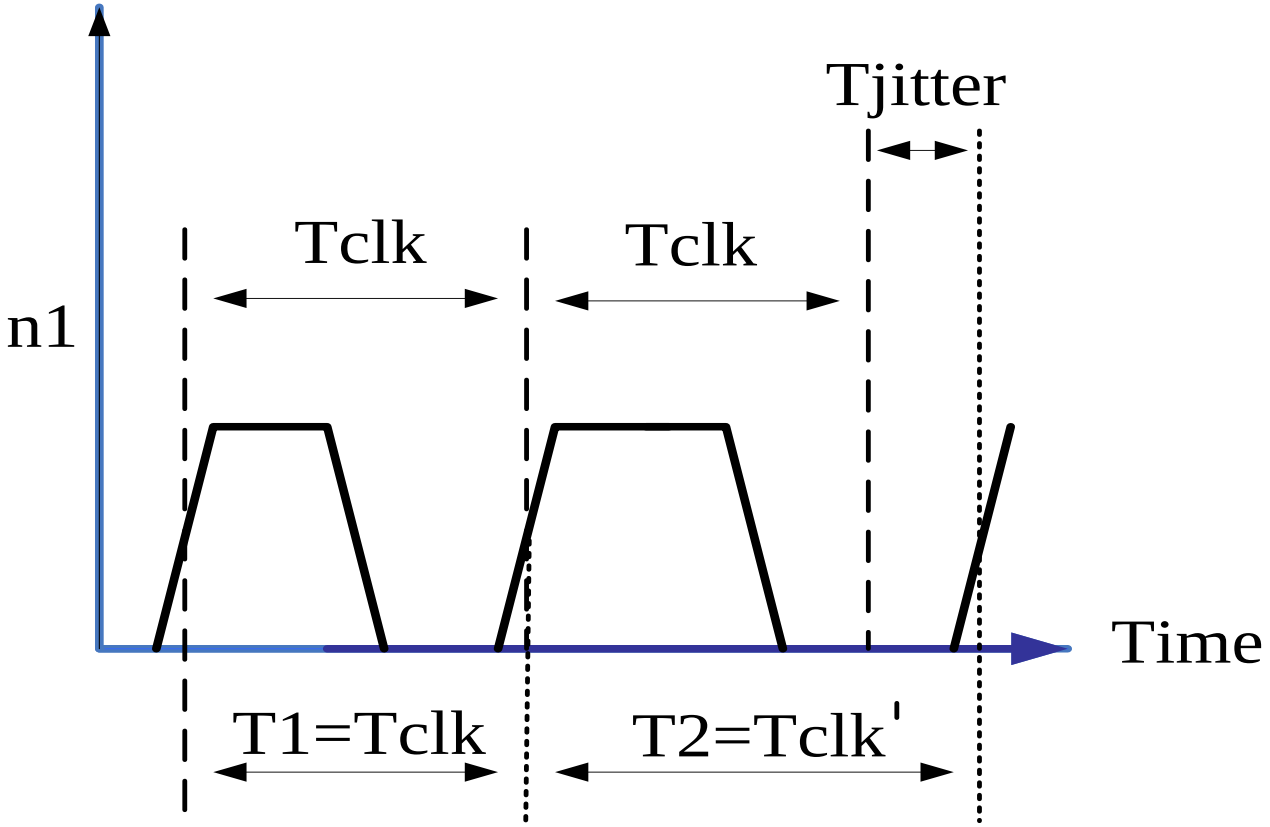
- Jitter is clock signal source of dynamic timing uncertainties at a single clock load.
- The key measure of jitter for a synchronous system is the period or cycle-to-cycle jitter, which is the difference between the nominal cycle time Tclk-n, and the actual cycle time Tclk.

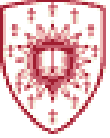
Absolute Jitter: is the worst case variation at a location wrt to an ideal reference clock.

Cycle to Cycle Jitter : is the time difference variation of a single clock period in respect to the ideal clock. Under worst case condition the cycle to cycle jitter is twice the absolute jitter.

Jitter

The first cycle, is the same as clock signal period and the second cycle, the clock period is longer by Tjitter. The total clock jitter is the sum of the jitter from the clock source and from the clock distribution. Power supply noise causes jitter in both the clock source and the distribution.





Sources of Jitter

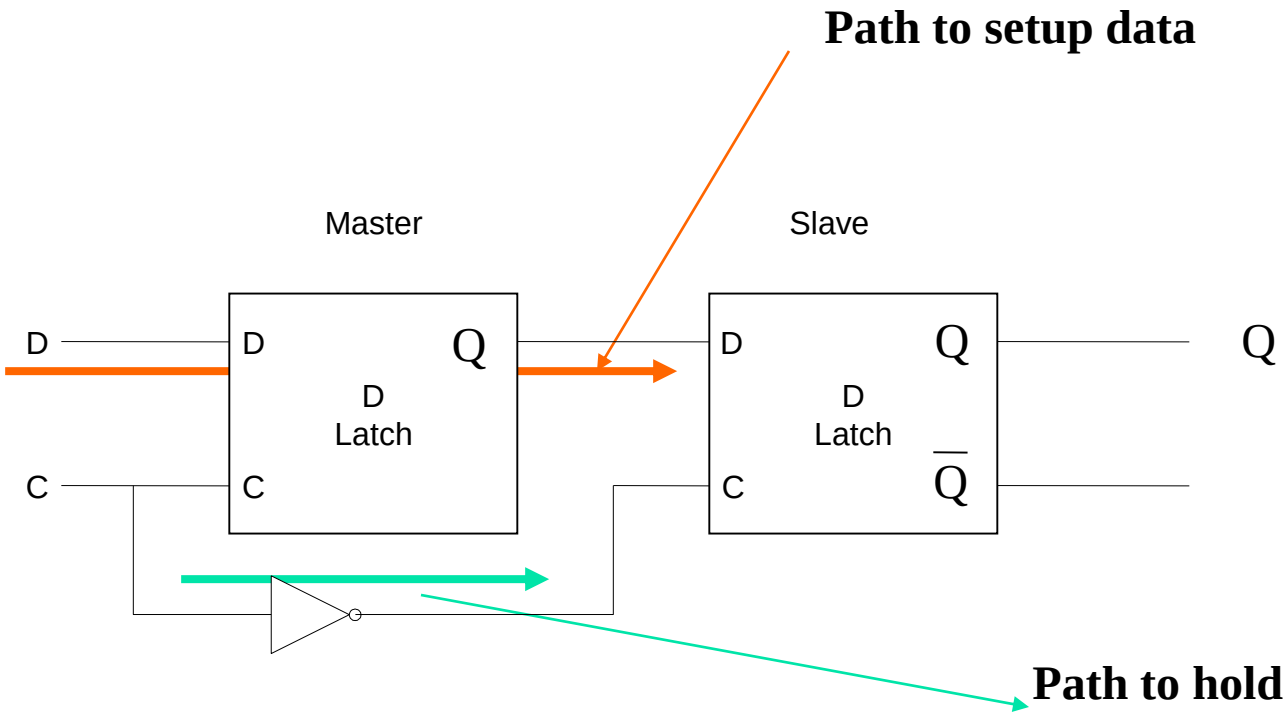
- In the past, jitter was mainly due to the clock source which generally is an on-chip phase-locked loop (PLL) that multiplies up an off-chip clock reference to the core clock frequency.
- The PLL jitter has scaled well with technology while the jitter in the clock distribution has not. As a result, the dominant source of clock jitter for today high performing systems is the clock distribution.

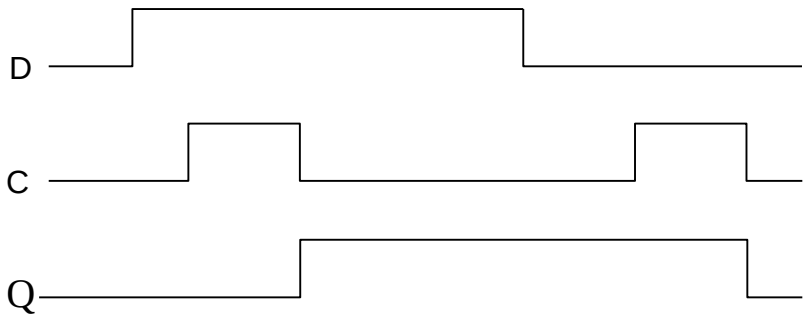
Safe Delivery of Data

Master Slave Flip Flop

Edge sensitive,-Falling Edge

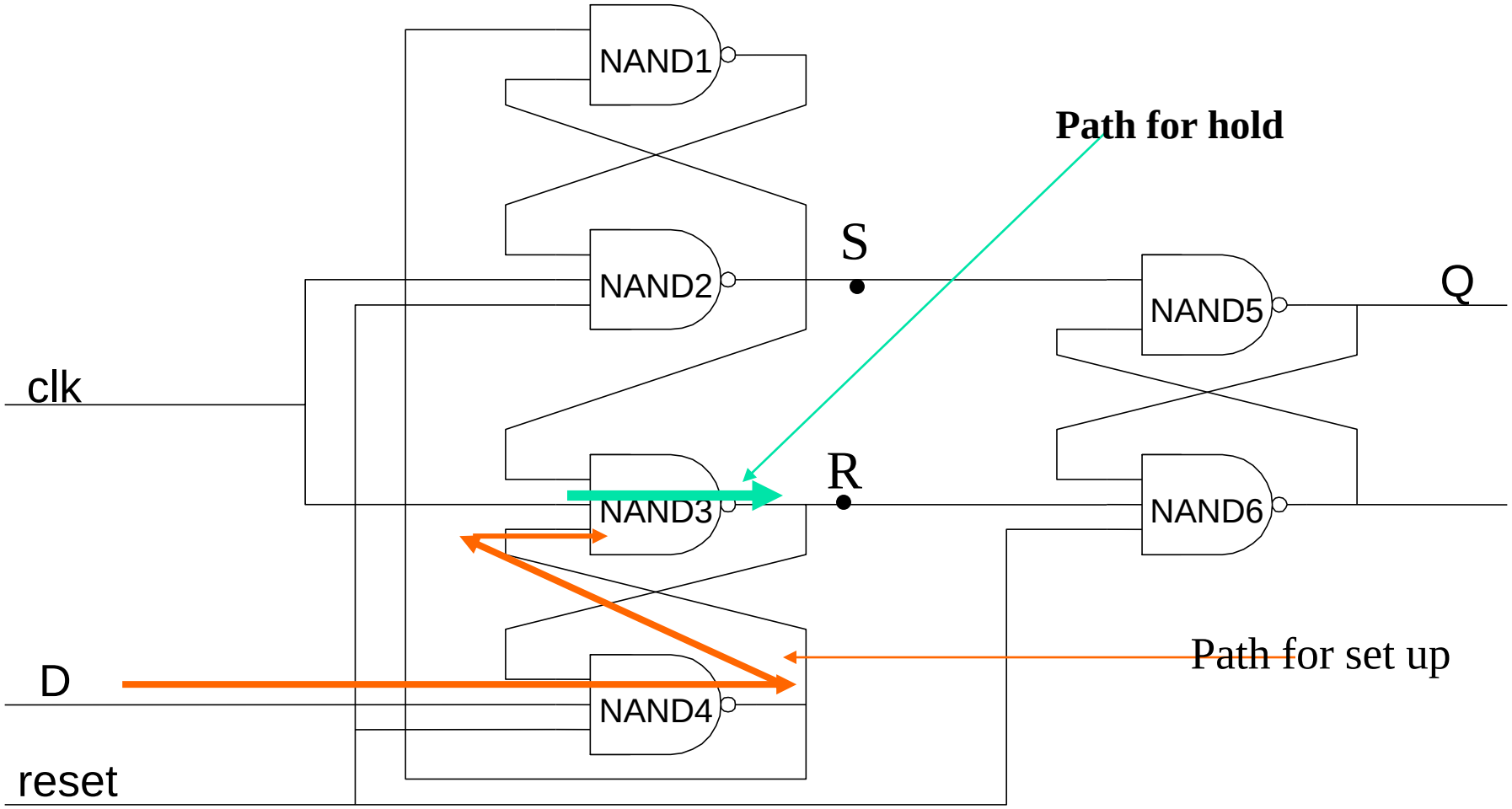
Set Up and Hold Time constraints





When CLK changes from 0 to 1

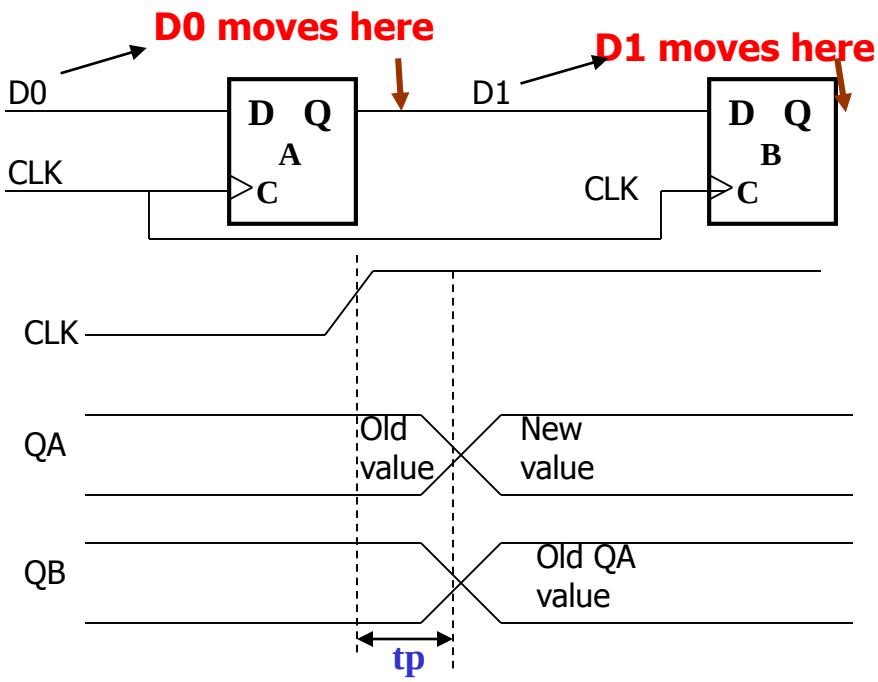
Case1, D=0: t_{setup}= t4, t_{hold}=t3



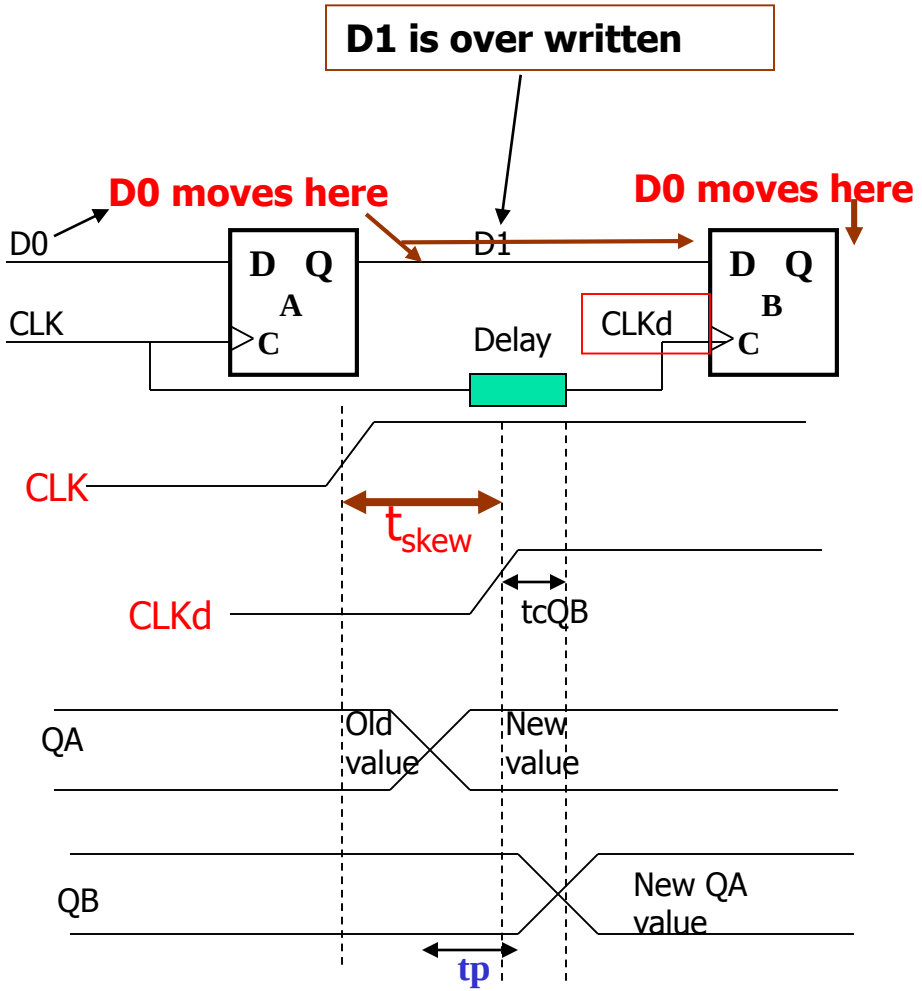
THE SKEW PROBLEM...

Tp propagation delay

tcQB clock to output delay



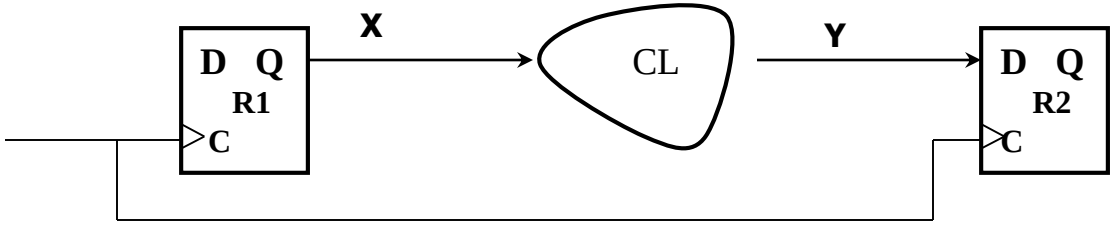
Assuming NO clock skew

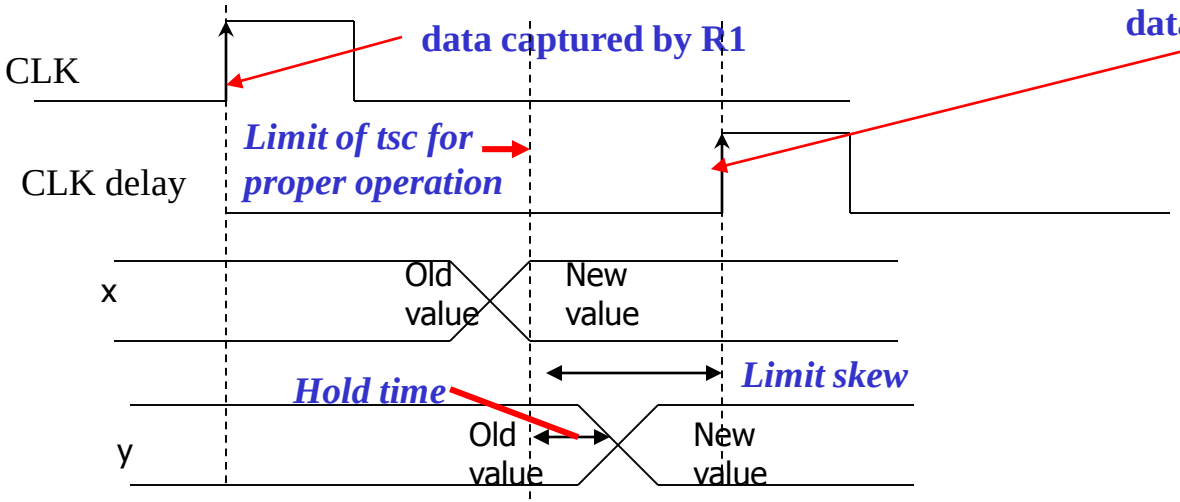


Assuming clock skew tcs

LOOK AT CLK AND CLKd TIMING

More on skew....





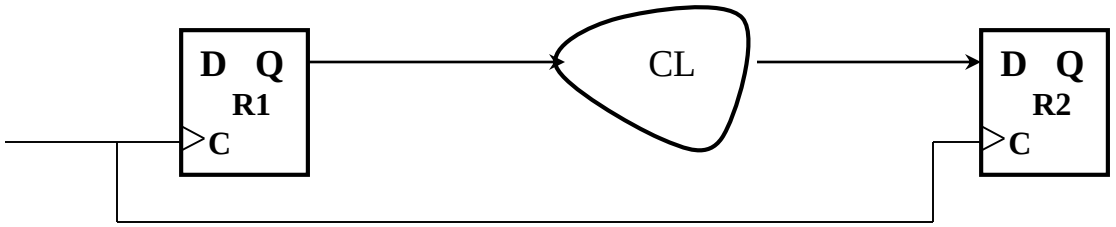
Tsc : clock skew delay
TCQ: FF clock to output delay
Tcl : logic delay
ti :interconnect delay
tsu :set up time

If the data changes fast at y and skew was large as shown, y can change and the edge of clock will capture the new data. In this case there is a race situation and there is a limit on the clock skew to capture the old data.

$$Tsc_{max} \leq t_{CQ}(min) + t_{CL}(min) + t_i(min) + t_{su}(min)_{R2}$$

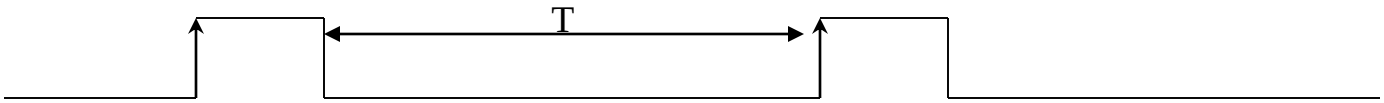
Hold time constraint **has to be taken care of as well, as it will be shown later**

clock and the skew....



Tsc : clock skew delay
TCQ: FF clock to output delay
Tcl : logic delay
ti :interconnect delay
tsu :set up time

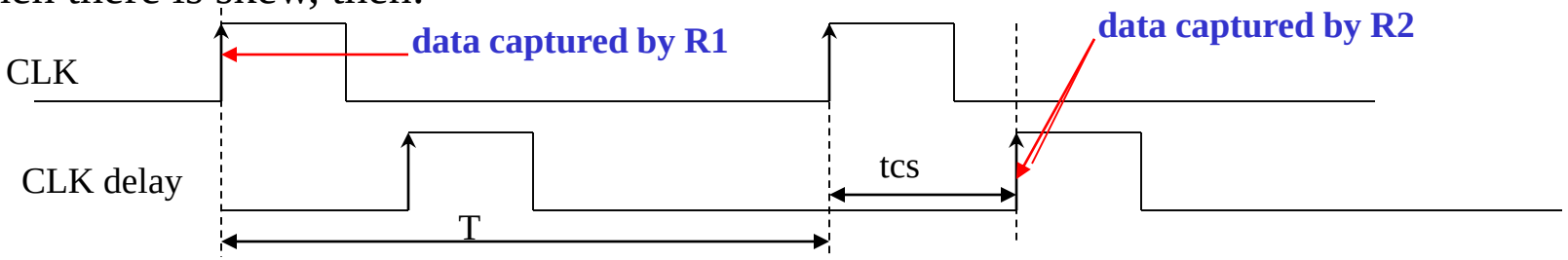
For the purpose of clock cycle determination, data is captured by R1 at time t: Transmitted through R1 and it is captured by R2 in the next clock cycle.



T should be enough for the data to travel along so that can be captured by R2. Therefore,

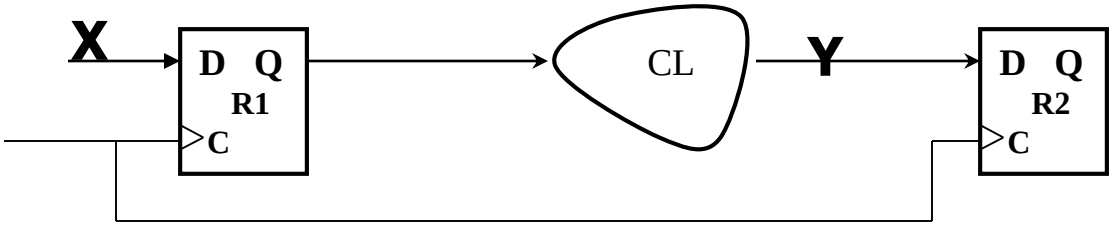
$$T = t_{CQ}(max) + t_{CL}(max) + t_i(max) + t_{su}(max)$$

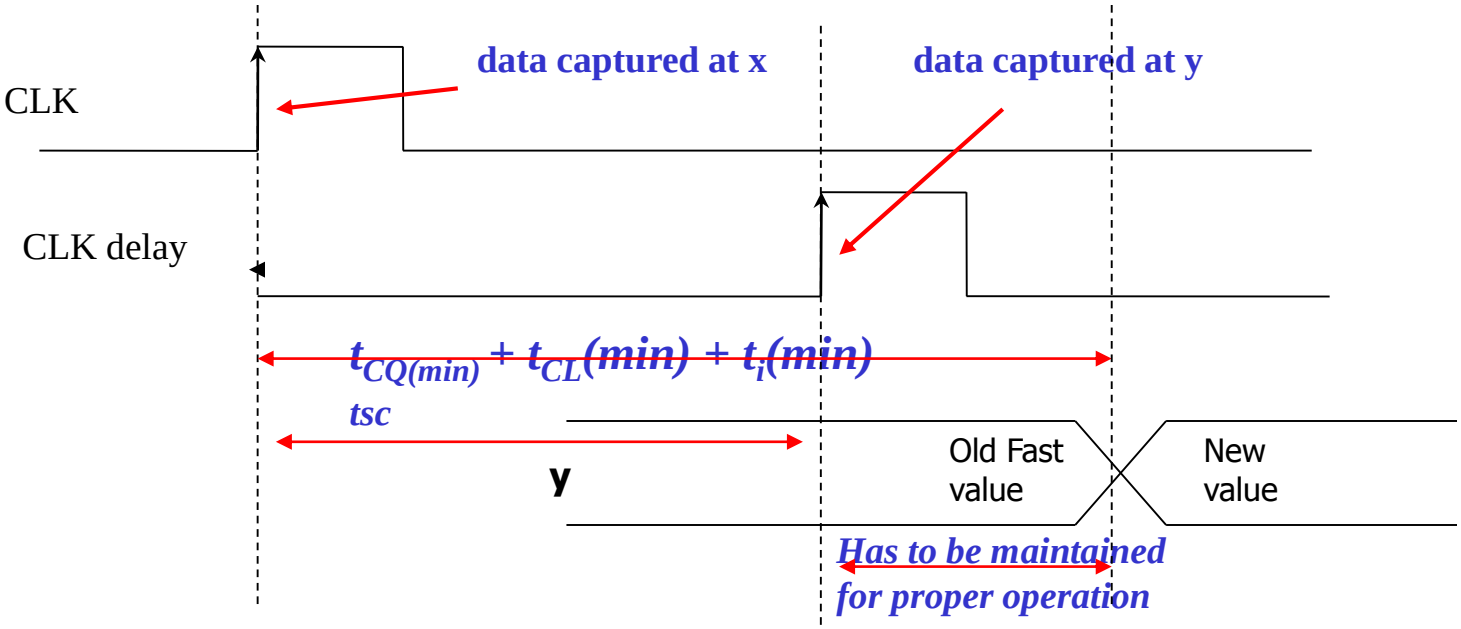
When there is skew, then:



Skew extends the period, so the actual clock period can be "the reduced original Clock period". To take care of this: $t_{actual} = t_{CQ(max)} + t_{CL}(max) + t_i(max) + t_{su}(max) - t_{cs}(min)$

Set up and Hold Time....





$$T_{hold}(max)_{R2} \leq t_{CQ}(min) + t_{CL}(min) - tcs\ (max)$$

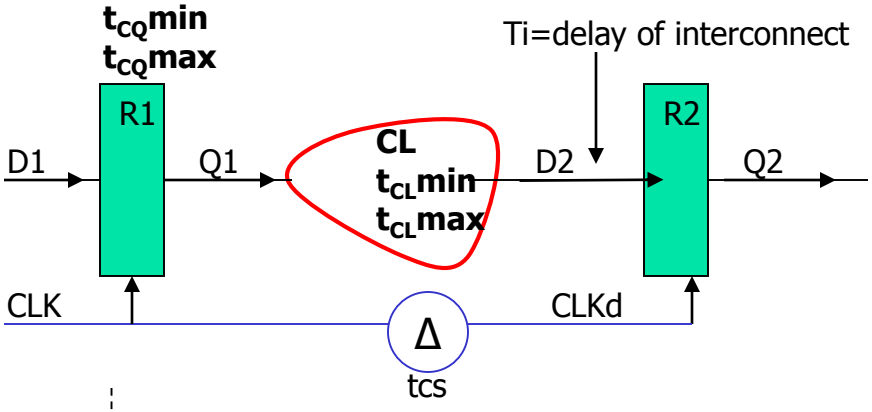
$$t_{CL} = t_{logic} + t_{interconnect}$$

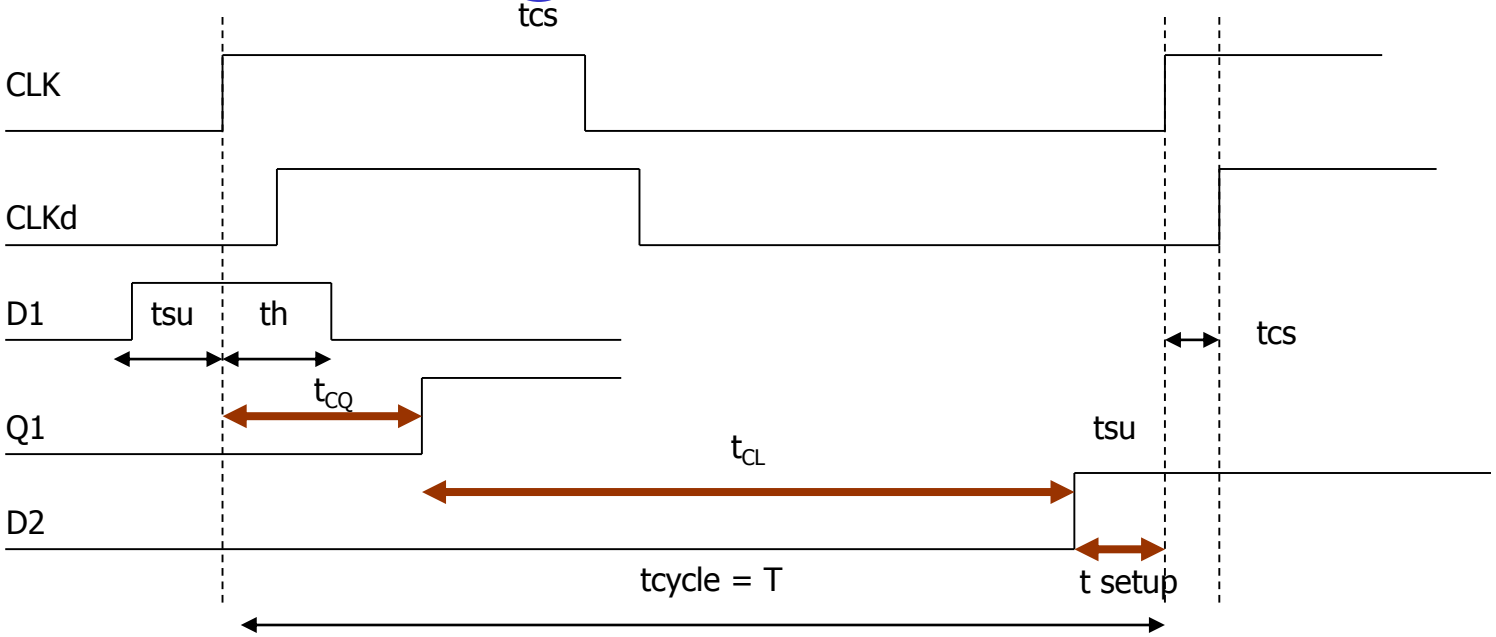
CONSTRAINTS ON CYCLE TIME....

For proper operation:

1. $t_{cs,max} \leq t_{CQmin} + t_{CLmin} + t_{su.min}$
2. $T_{min} \geq t_{CQmax} + t_{CLmax} + t_{su^{max}R2} - t_{csmin}$
3. $t_{hmaxR2} < tCQ_{min} + t_{CLmin} - t_{csmax}$

$t_{CL} = t_{Logic} + t_{interconnect}$





CONSTRAINTS ON CYCLE TIME....

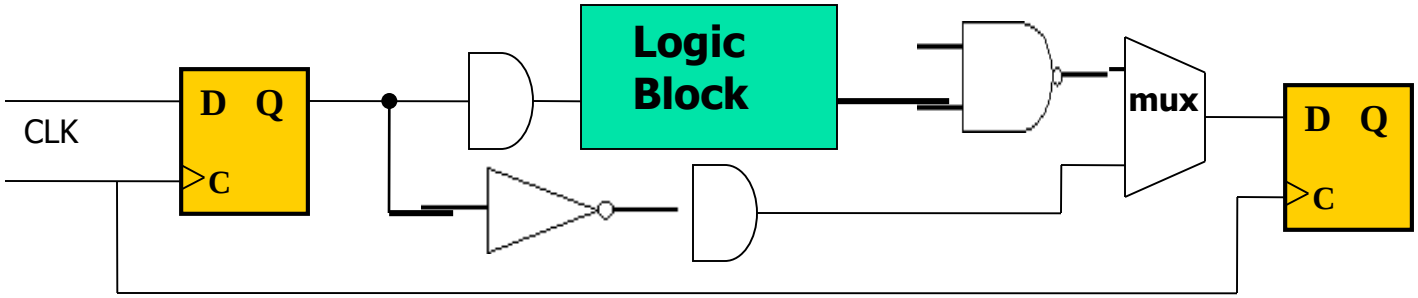
1) **Clock skew can harm edge triggered FF as follows: The latest time that R2 can make a transition is at $t_{CQ}min + t_i min + t_{CL}min + t_{su}max$. If the local clock of R2 is delayed with respect to the clock of R1, it might happen that the inputs of R2 change before the previous data is latched. A race occurs, and the circuit yields erroneous results.**
2) **The correct input data is stable at R2 after the worst case propagation delay or at time $t_{CQ}max + t_{CL}max + t_i max + t_{su} max - t_{cs}min$. Therefore, this is a limit on the clock period. (Clock skew is subtracted because it is assumed that the logic is routed the same direction of the clock) .**
3) **In addition, to prevent a race condition in which the data may go through two FFs at the same cycle, minimum FF to FF delay must be greater then the maximum hold time. $Thold\text{-}max_{R2} < t_{CQ}min + t_{CL}min + t_i min - t_{cs}max$**
If this inequality does not hold, the o/p of a FF may reach the next FF while 2nd FF is still trying to capture data from the previous FF. Even with 0 ns hold time there can be problems because of CLK skew. If necessary, buffers may be inserted to increase the delay as a remedy.

Example CLOCK Skew

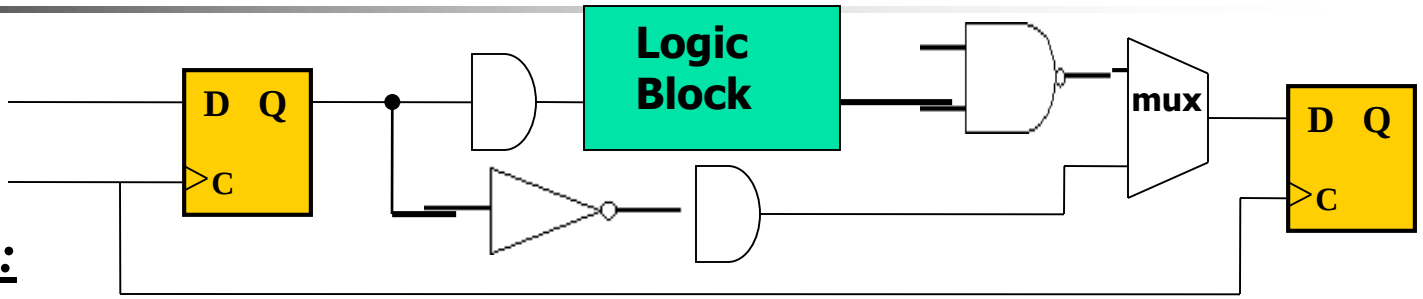
Example

Consider the logic shown below. Determine the minimum and maximum allowable clock skew so that the circuit operates correctly. Assume the following parameters:

Gate delays = **tg for all gates**
Logic Block delay =3tg
Mux delay: **tmux** **=2**tg
Latch setup: **tsu** **=**tg
Interconnect delay: **ti** **=**tg
FlipFlop CLK→ Q: t_{CQ} =2tg (t_{CQ})
Clock skew : **tcs**



Clock Skew Example



Longest Path:

1. $T \geq t_{CQ} + 2\,tg + t_{block} + t_{mux} + t_i + t_{SU} - t_{CS}$
   $\geq 11\,tg - t_{CS}$

Bringing t skew to the left and T to the right of the inequality gives

$t_{CS} > t_{CQ} + 2\,tg + t_{block} + t_{mux} + t_{SU} + t_i - T$
$> 11\,tg - T$

BUT

Shortest Path:

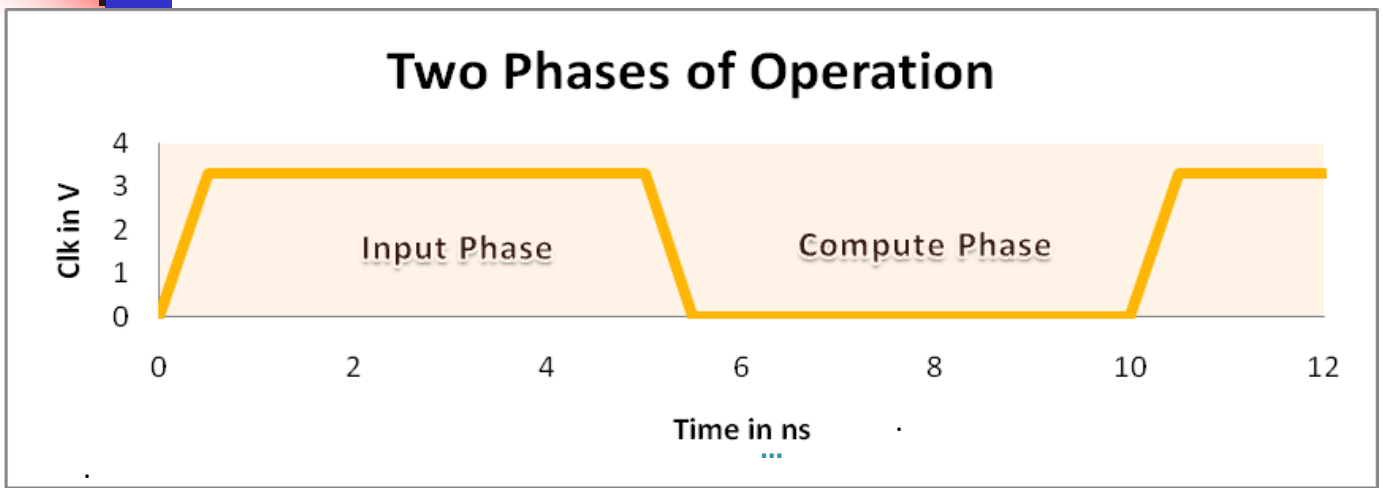
2. $t_{CS} < t_{CQ} + 2\,tg + t_{mux} + t_{SU} + t_i$
   $< 8tg$

Combining 1 & 2

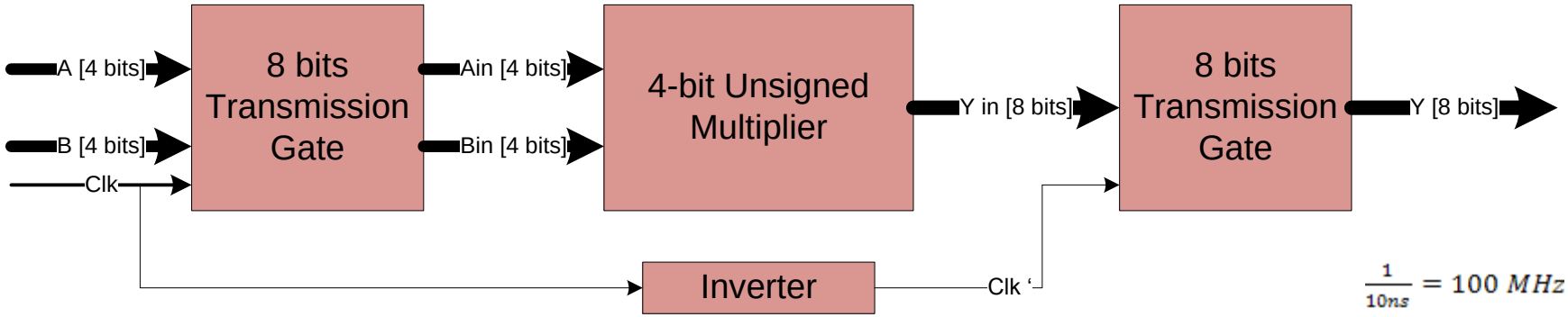
$t_{CS} < t_{CQ} + tg + t_{mux} + t_{SU} + t_i$

$8tg > t_{CS} > 11tg - T$

Two Phase Operation

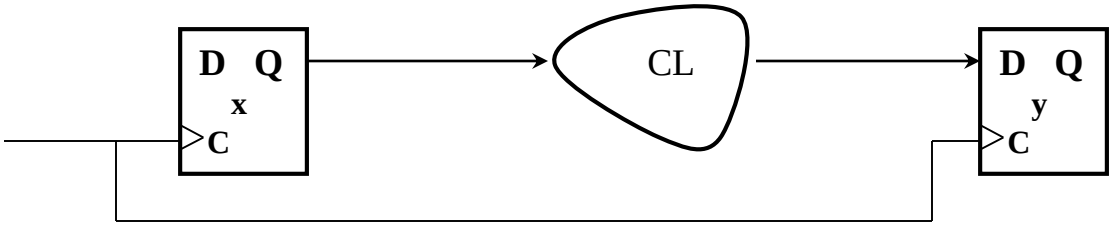


$$\frac{1}{8ns} = 125\ MHz$$

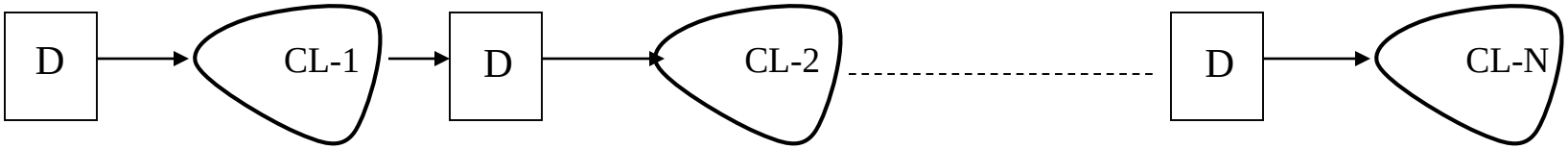


$$\frac{1}{10ns} = 100\ MHz$$

Traditional Pipelining

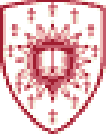


If the combinational logic, CL, is divided into **N stages**, then



The system throughput can be increased in theory by N times. However, there is a limit, depending on the cost. Problems:

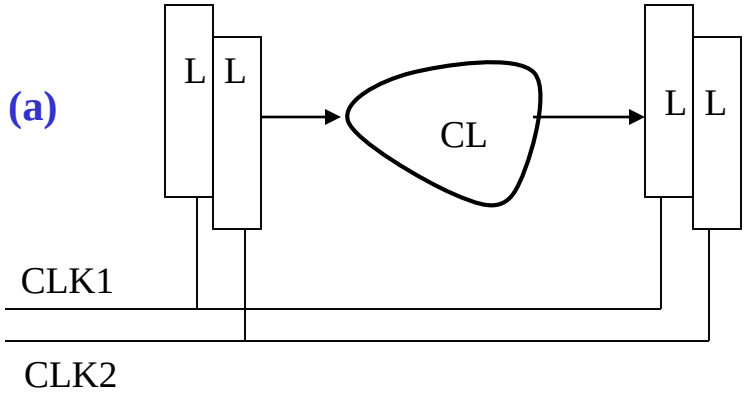
a) Latency is increased by **N-1**.
b) Combinational circuit cannot be broken into **N equal parts**, therefore it has to be broken into the biggest breakable combinational part possible.
c) Area overhead of flip flops and larger clock drivers.
d) Power overhead of flip flops and larger clock drivers.
e) Clock cycle overhead due to set up time, hold time, skew time and clock drivers.
f) Extra overhead in extra wiring of clock.

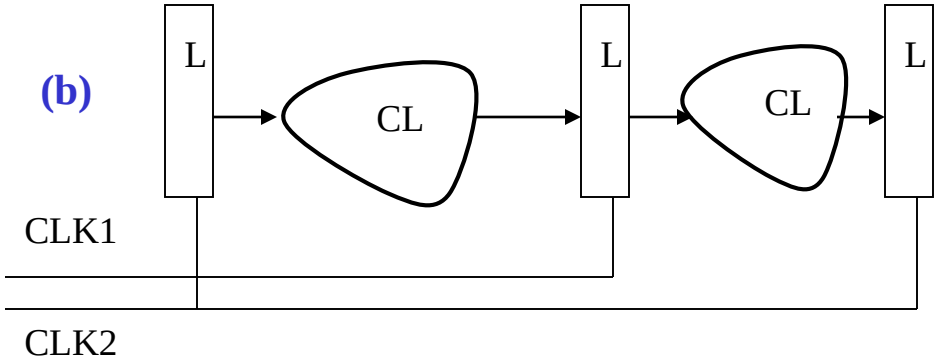


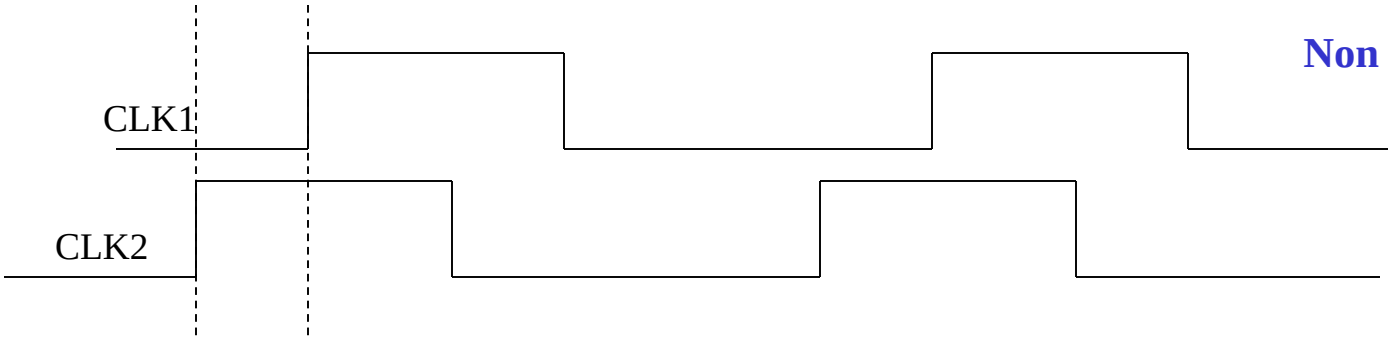
Wave Pipelining

Wave pipelining relies on the ***finite propagation delay*** of signals through a combinational digital circuit to store data. Rather than allowing data to propagate from synchronizing element, through the combinational network to another synchronizing element prior to initiating the subsequent data to the network as soon as it can be guaranteed that it will not interface with the current data wave. In this manner, multiple waves of data are simultaneously propagated through distinct regions of the logic network.

Set up and Hold Time....







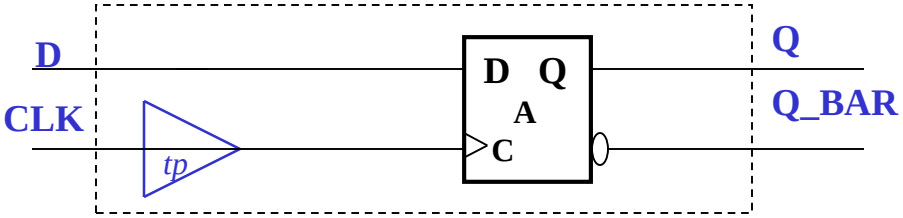
Non overlapping clocks

Two phase non-overlapping clocking

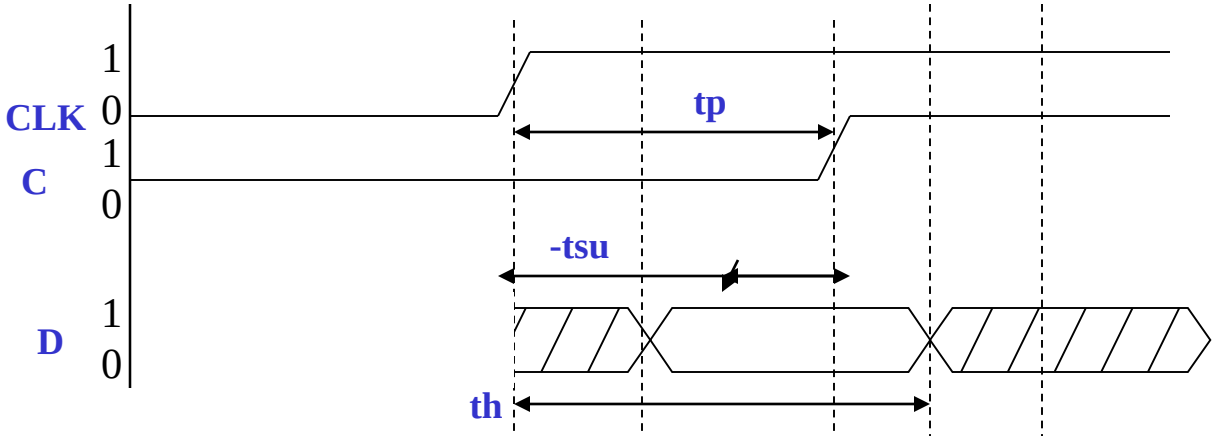
a) Double Latch Design

b) Single Latch Design

Set up and Hold Time....



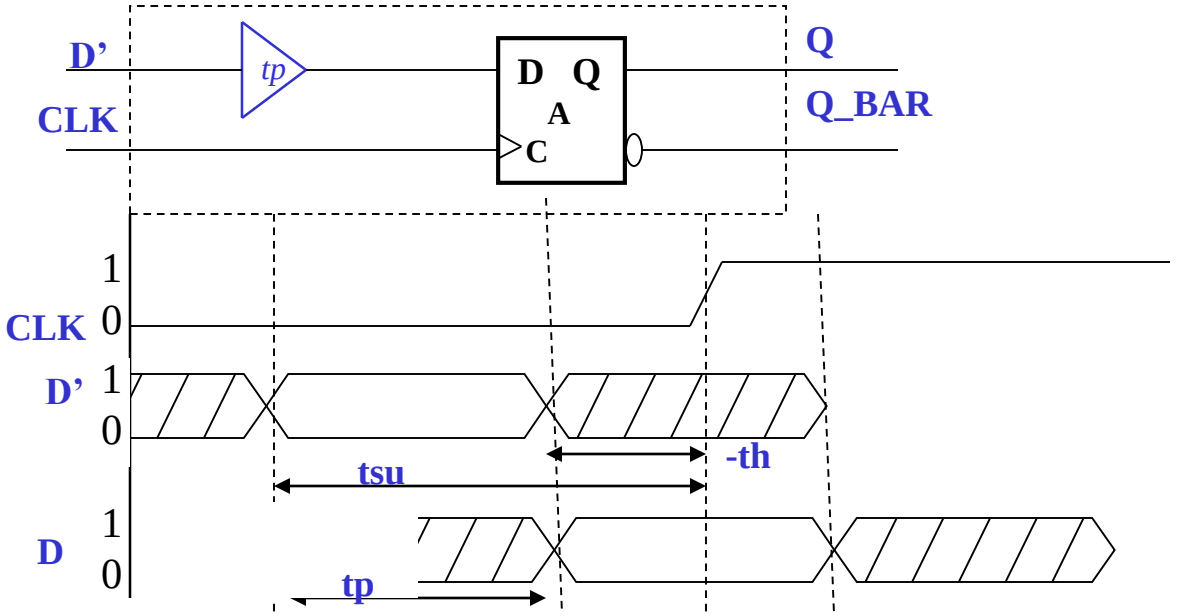
PART (a)



PART (b)

The same scenario as the previous slide is shown here. The buffer is packaged with the FF.

Set up and Hold Time....



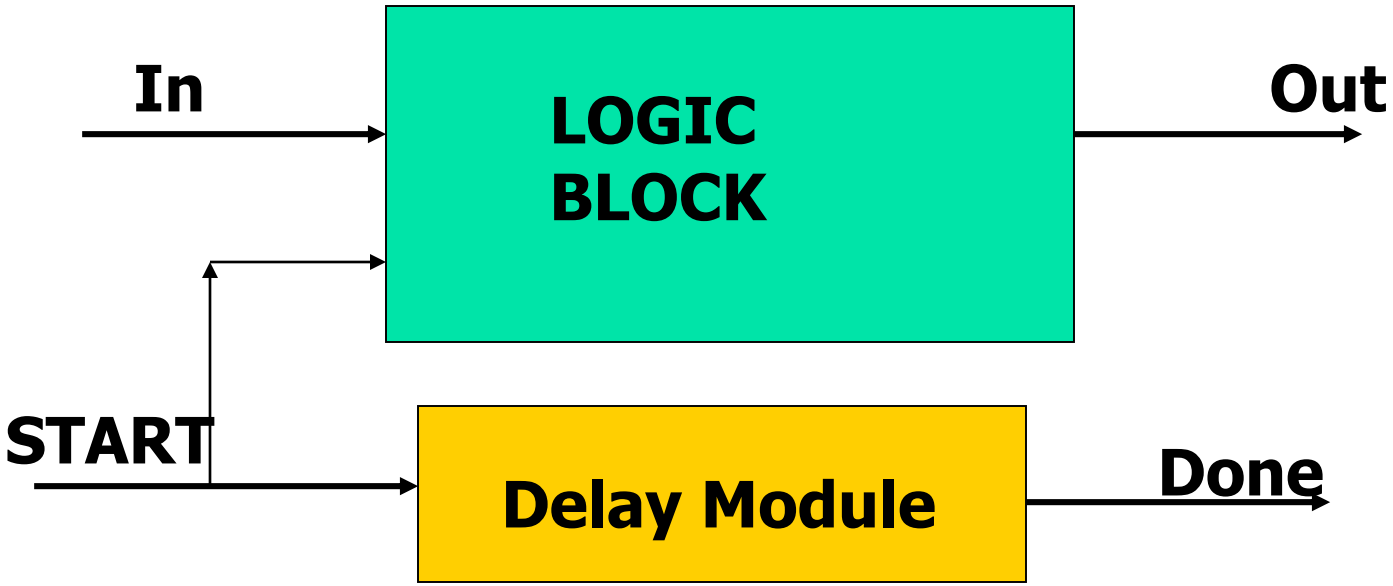
D flip-flop (DFF) memory that includes a buffer on the data (D) input for signal refresh. The buffer is internal to the cell as the buffer and the FF are packaged together as a single component. The bottom signal waveform D drives DFF and is shifted in time by the propagation delay of the buffer. The length of time the data D' must remain stable is **tsu+th** but the hold time is negative. This convention implies the data/clock timing shown for D' and CLK. Adhering to the negative time constraints guarantees that the data D delayed by the buffer and tp is correctly aligned.

Asynchronous approach

- Local solution to timing problem
  - Each Combinational function has
    "Start" signal to initiate the computation
    - Means of indicating that it has "completed" the computation

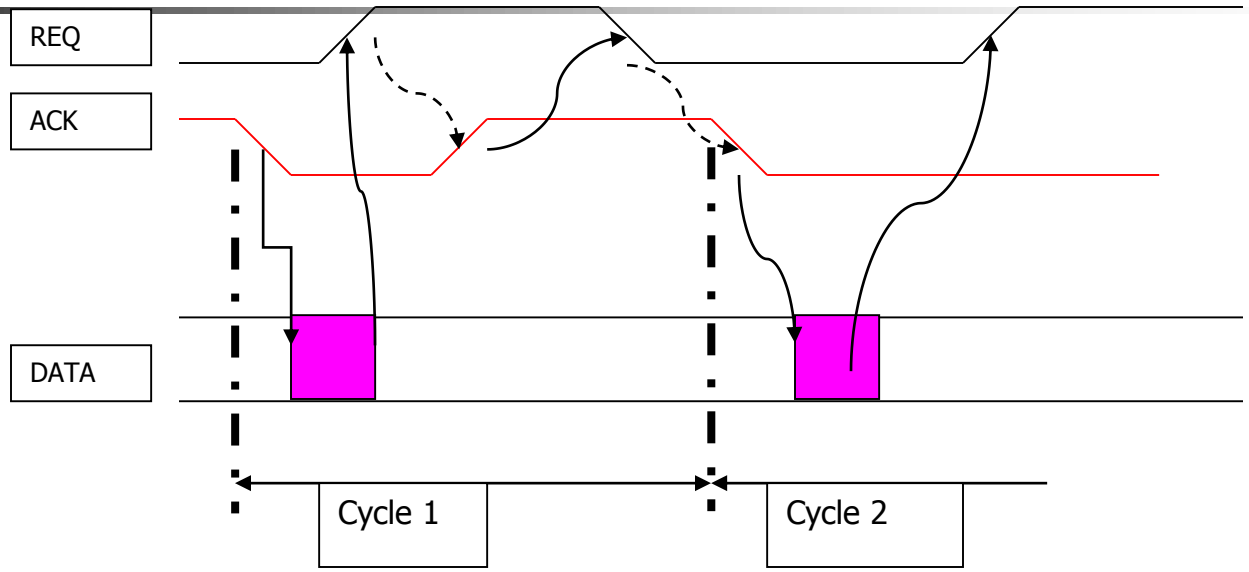
      This does come at an expense of implementing **Delay Modules**
  - 4-phase asynchronous handshake protocol between Multiplier and adjacent modules

Completion-signal generation using delay module



•This mechanism helps to establish the precise duration of a computation in a data-path fashion(taking care of physical constraints).

4-Phase Handshake protocol



When data is put on the bus,

Phase1-The "REQ" is raised .

Phase 2-The receiver accepts the data and raises" ACK".

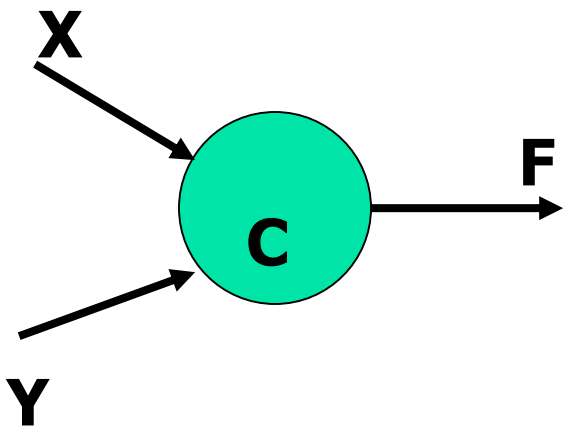
Data is now with the receiver, then:

Phase 3- "REQ" goes back to the initial state.

Phase 4 -"ACK" goes back to its initial state.

Now new data can be put on the data bus.

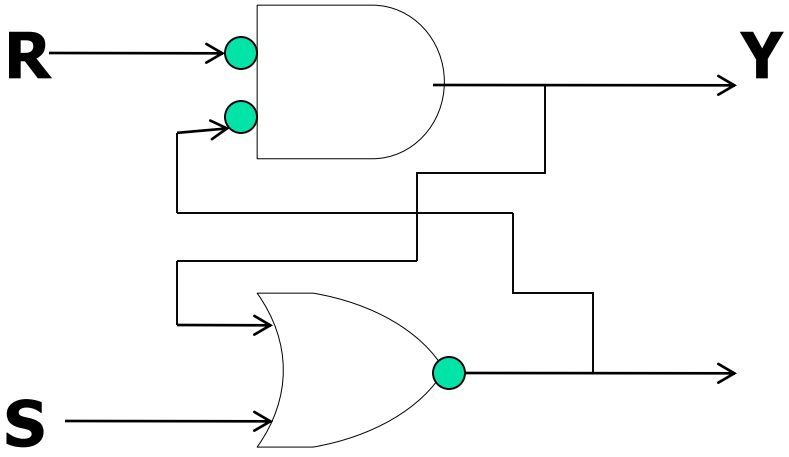
The Muller C-Module



X	Y	F_{n+1}
0	0	0
0	1	F_n
1	0	F_n
1	1	1

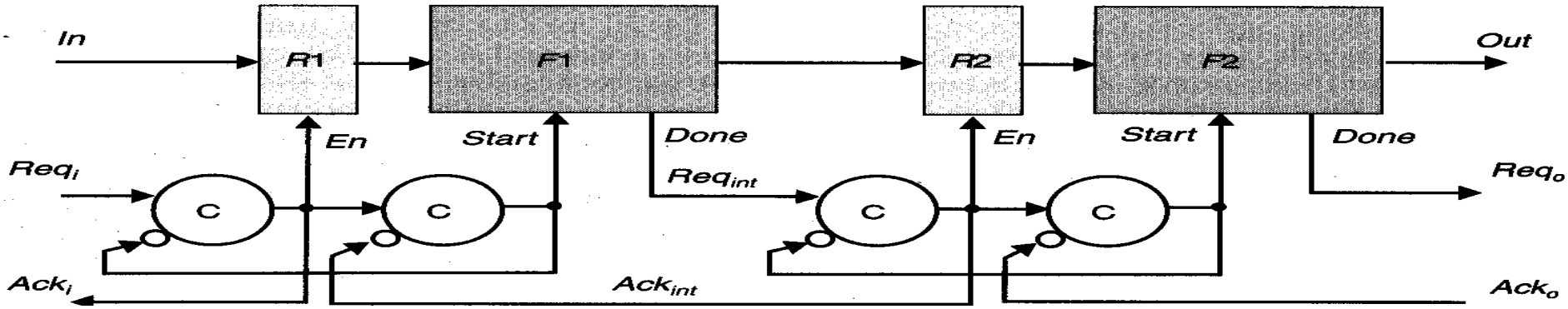
Essential component for implementation of 4-phase handshaking protocol

Muller C- Module



Yt	Yt+1	Yt
0 ⟶	0	0
0 ⟶	1	1
1 ⟶	0	0
1 ⟶	1	1

Asynchronous pipelined datapath-complete composition

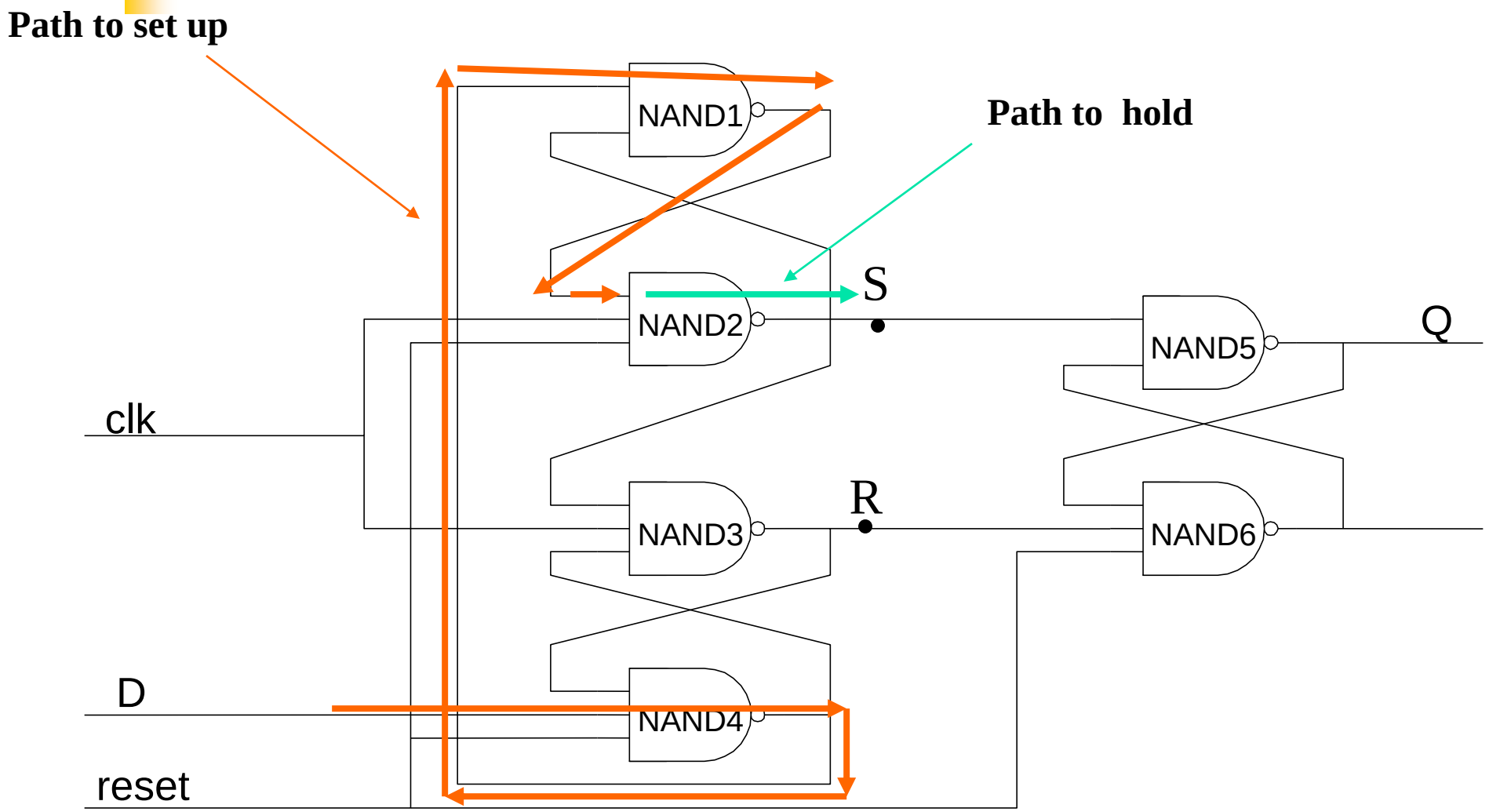


- **"Req" and "Ack" are initially in zero state and "Start" signals are low**
- **An input "Req_i" triggers the first C-element**
- **The enable signal "En" of R1 is raised and input data is latched**
- **"Acki" acknowledges the acceptance of the data and input buffer can respond to it by resetting the "Req_i"**
- **Second C-element is triggered, since Ackint is low**
- **"Start" signal is raised and starts the evaluation of F1**
- **At its completion, the output data is placed on bus and "Reqint" is raised**
- **Second stage acknowledges its acceptance by raising "Ackint", while stage one is blocked for further computations**
- **This lowers "En" and "Acki", "Start" goes low and F1 is ready for new data**


ABRUTIS.com
SAFETY
Safe!!

When CLK changes from 0 to 1

Case2, D=1 t_{setup}=t4 + t1 t_{hold}= t2

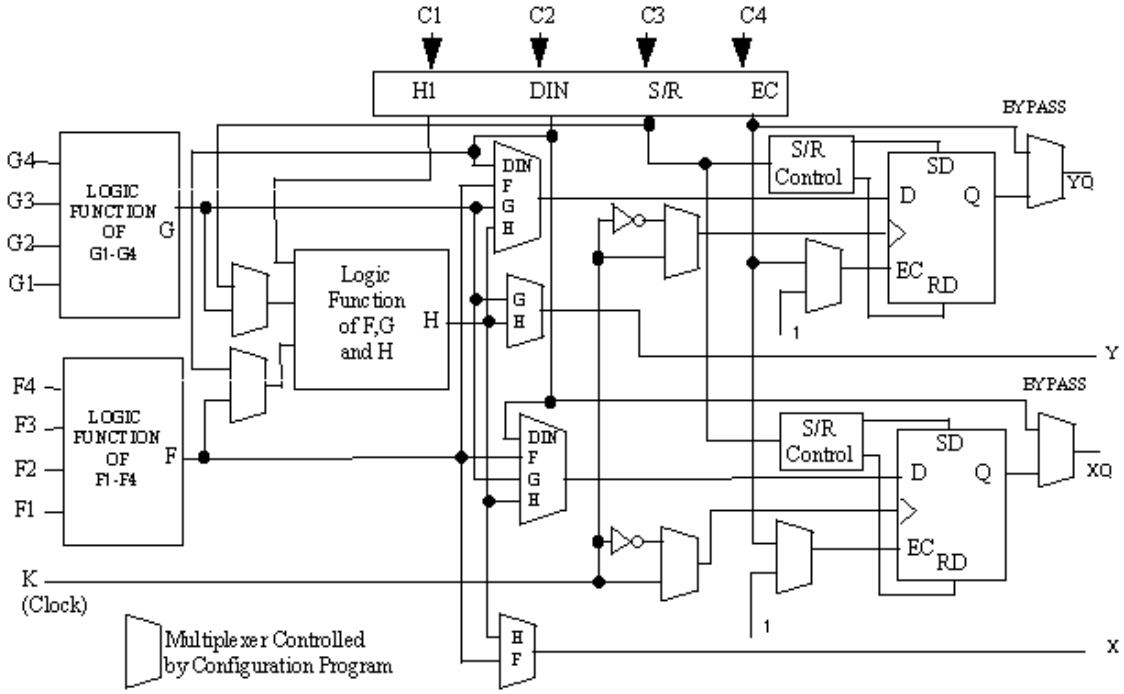


Xilinx 4000e series CLB's

Each CLB contains:

- A pair of flip-flops
- Two independent 4-input function generators
- Thirteen inputs and four outputs

Hint: CLBs implement most of the logic in an FPGA



Block Diagram of XC4000 Families
Configuration Logic Block (CLB)