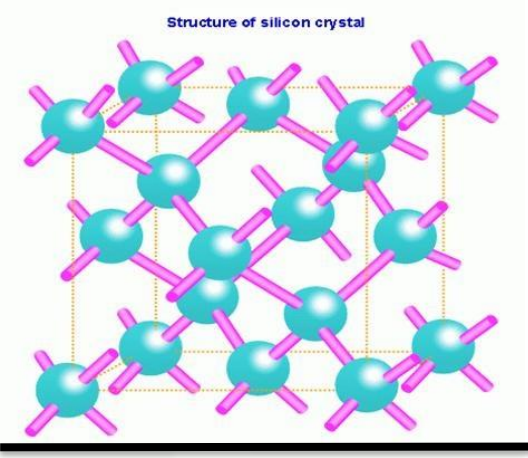
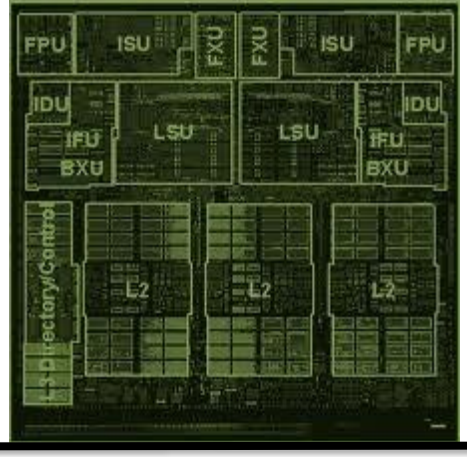
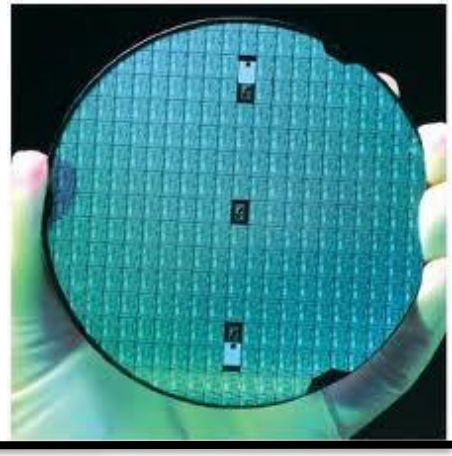




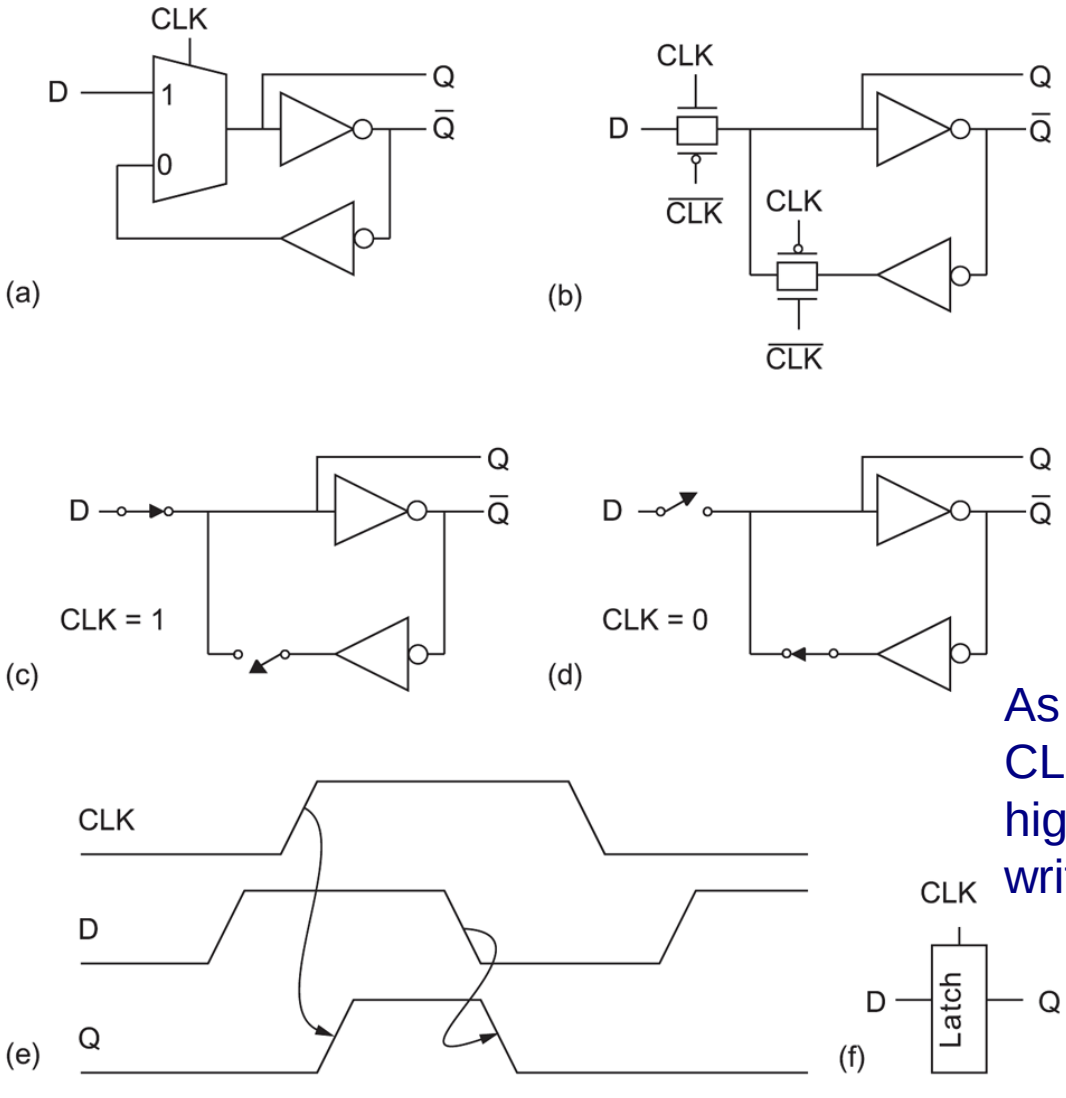


ECE 225
Lecture 16
Clocked Circuits, Timing and Clocking

Prof. Kaustav Banerjee
Electrical and Computer Engineering
University of California, Santa Barbara

Latches

Multiplexer based



CLK=1: D to Q

CLK=0:Holds state of Q

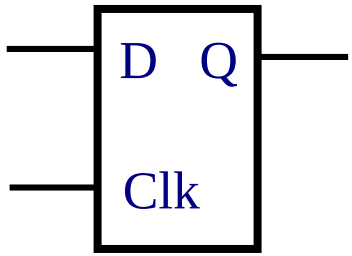
As long as CLK remains high, D will be written on Q

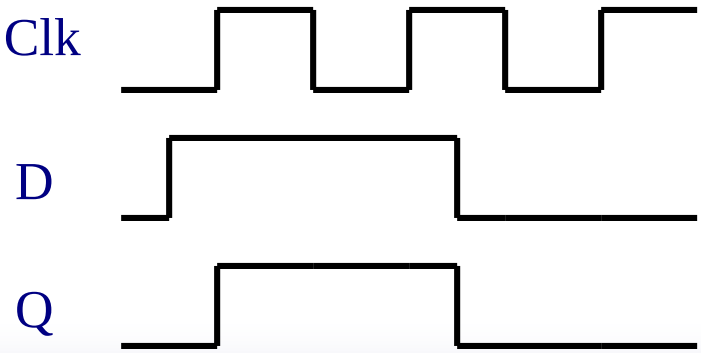
FIG 1.30 CMOS positive-level-sensitive D latch

Latch versus Register

- Latch

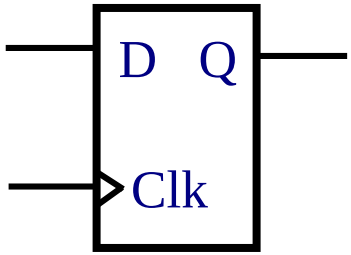
  stores data when clock is low (or high)

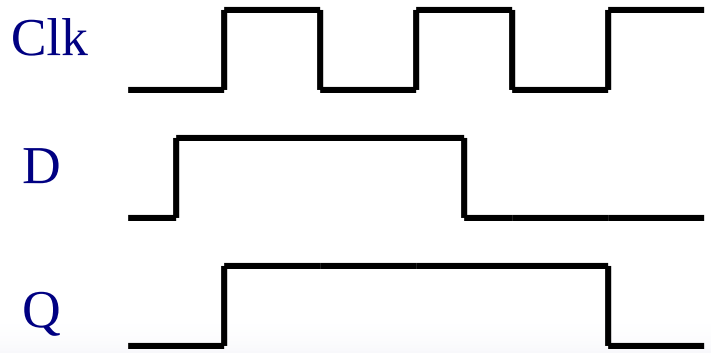




- Register

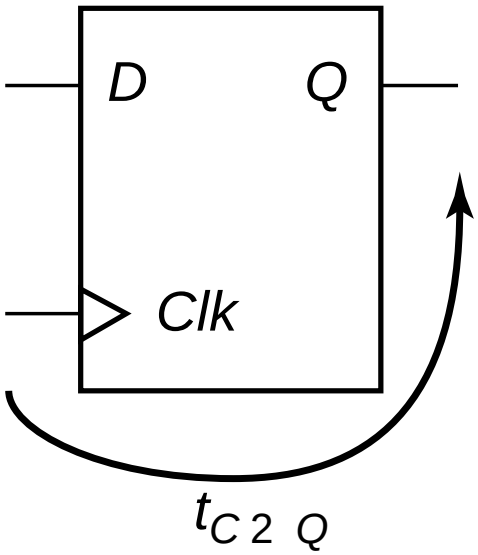
  stores data when clock rises (or falls)





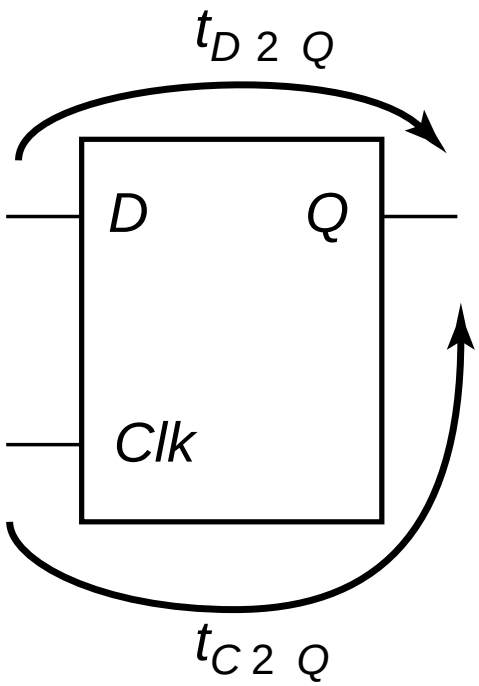
Characterizing Timing

Register



Data is ready when Clk arrives….

Latch



Data may arrive after Clk edge….

Requires an extra timing parameter…

Registers or Flip-Flops

Combines two latches:

One +ve sensitive (slave) and one –ve sensitive latch (master)

Edge Triggered FF or Master-Slave FF

CLK=0: D to QM

$\overline{QM} = \overline{D}$

Slave holds previous value of Q

CLK=1: master can't sample input and holds value of D

Slave opens and QM=(D) =Q

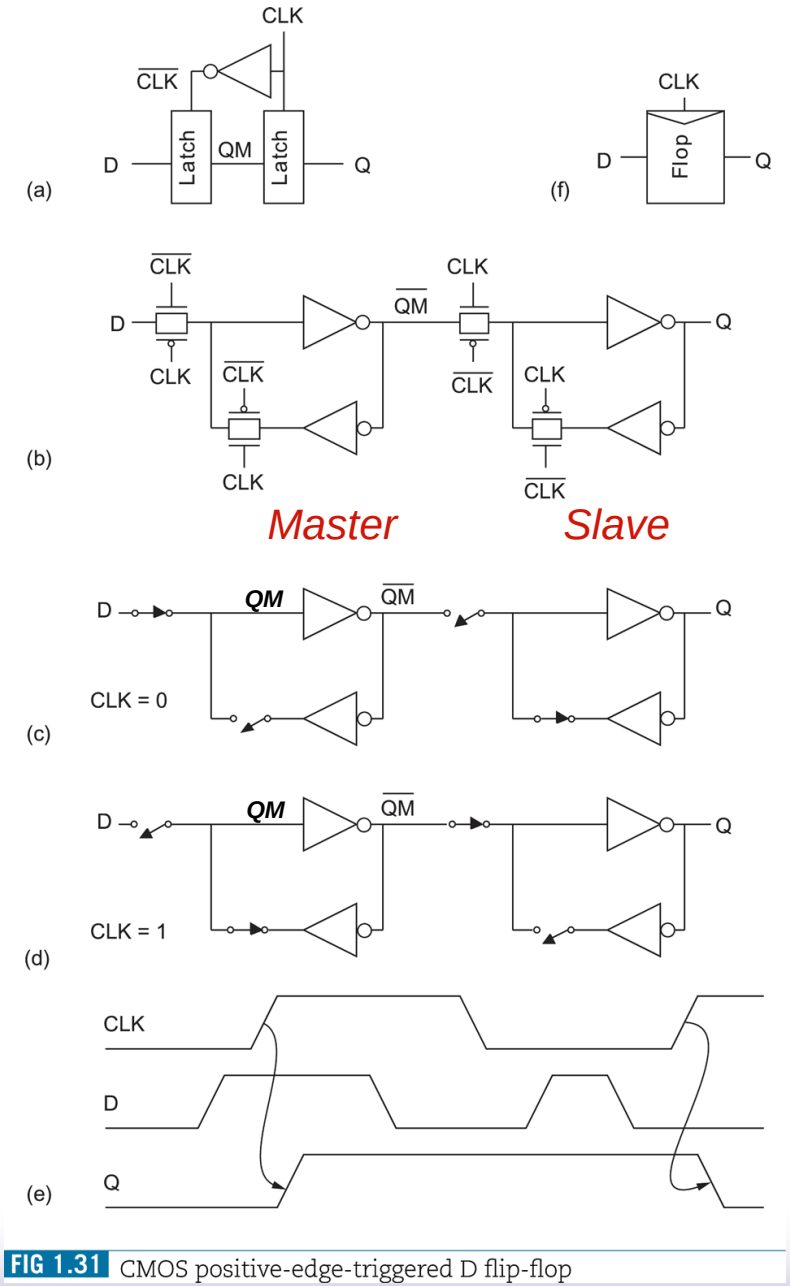
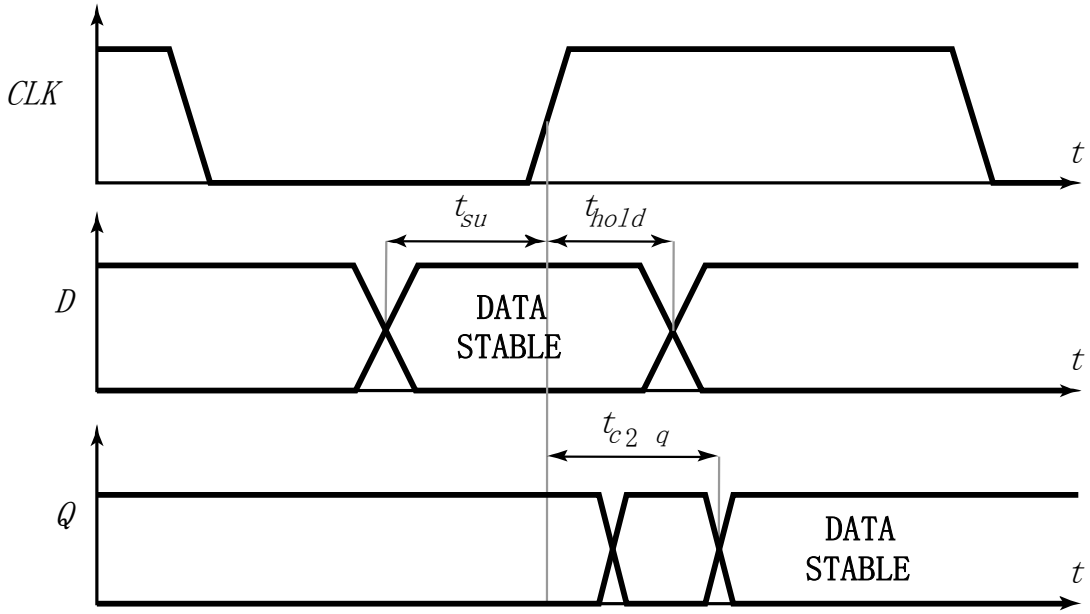
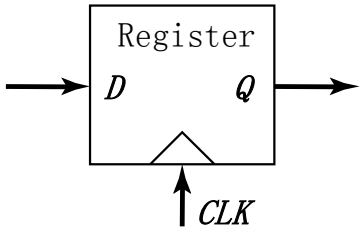


FIG 1.31 CMOS positive-edge-triggered D flip-flop

Timing Definitions



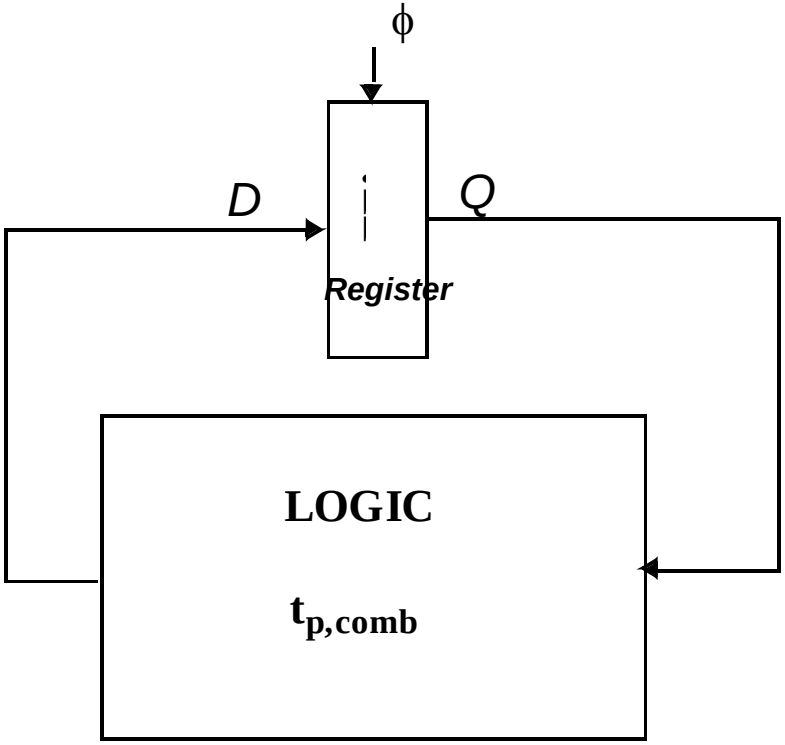


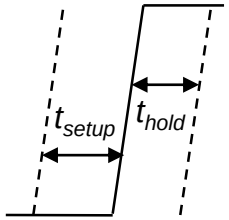
t_{su}= setup time =time for which the data inputs (D) must be valid before the CLK edge

t_{hold}= hold time =time for which data input must remain valid after the CLK edge

t_{c2q}= worst case propagation time through the Register (w.r.t the CLK edge)

Maximum Clock Frequency





To ensure that the input data of the sequential elements is held long enough after the CLK edge and is not modified too soon by the new wave of data coming in:

$$1)\ T_{min} = t_{clk\text{-}Q} + t_{p,comb} + t_{setup}$$

Clk period must accommodate the longest delay of any stage in the network

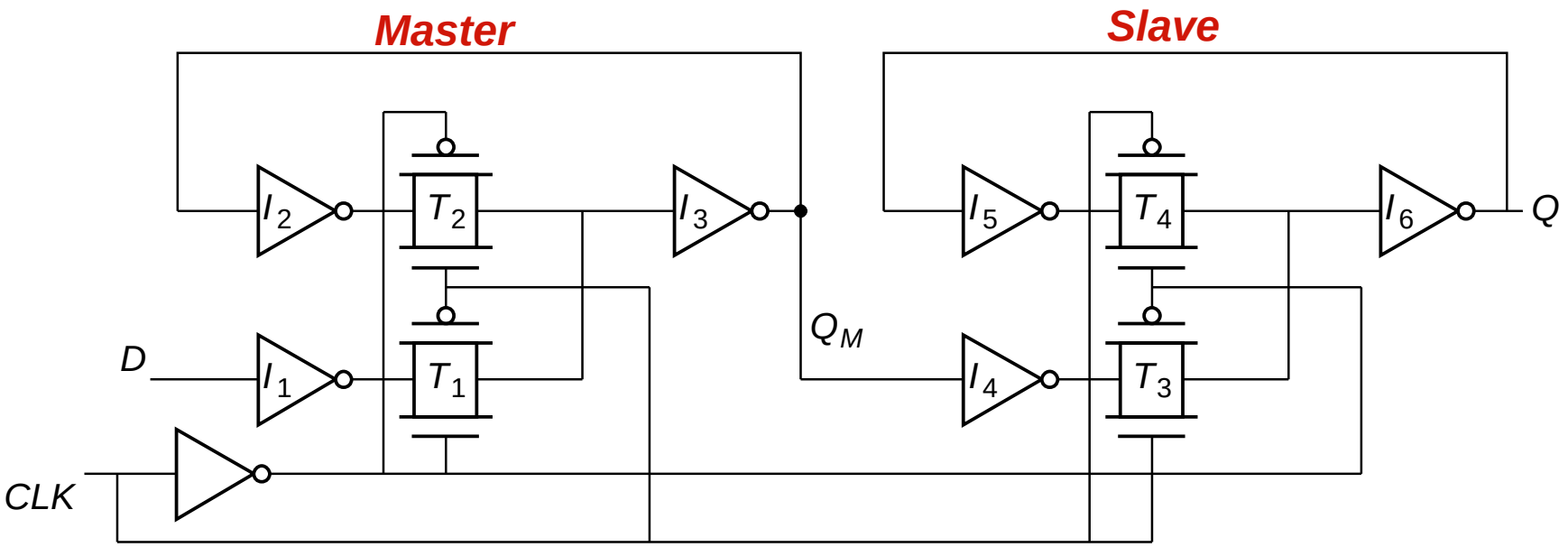
$$2)\ t_{cdreg} + t_{cdlogic} > t_{hold}$$

t_{cd}: contamination delay = minimum delay

Master-Slave +ve Edge Triggered Register

Transistor Level Implementation

X-gate Multiplexer-based latch pair



CLK=0: *T1 is on, T2 is off, D input sampled onto QM*

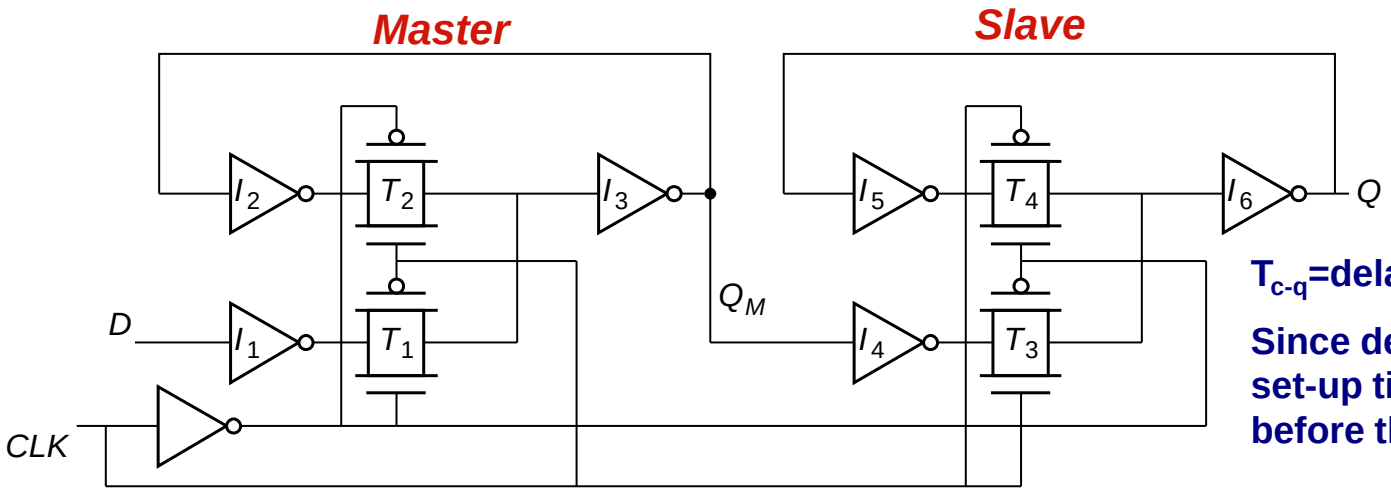
T3 off and T4 on: I5 and I6 hold the state of the Slave

CLK=1: *T3 is on, T4 is off, QM sampled onto Q*

T2 on and T1 off: I2 and I3 hold the state of QM

Master-Slave +ve Edge Triggered Register

Transistor Level Implementation



$T_{c\text{-}q}$=delay through T3 and I6

Since delay of I2 is included in set-up time output of I4 is valid before the rising edge of CLK

$T_{c\text{-}q} = t_{pd\_inv} + t_{pd\_tx}$

t_{su} = set-up time = time before the rising edge of the CLK during which the D input should remain stable so that QM samples the value reliably

Since D must propagate through I1, T1, I3, and I2 before the rising edge

$t_{su} = 3t_{pd\_inv} + t_{pd\_tx}$

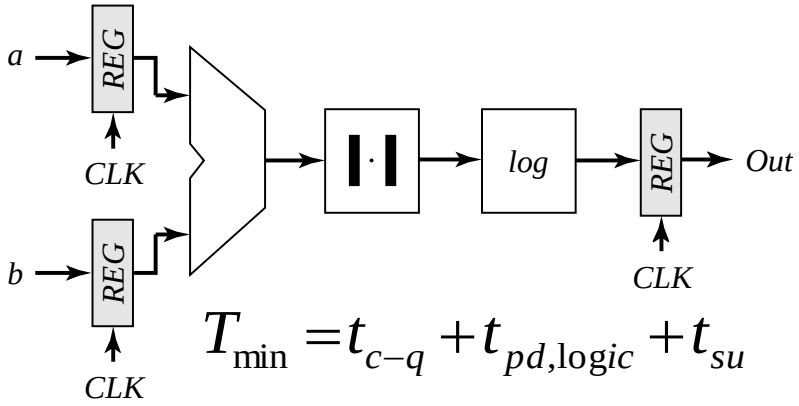
To ensure equal node voltages on both sides of the Xgate

$t_{hold} = 0$ (since T1 is cut off after CLK edge)

Pipelining: Optimizing Sequential Circuits

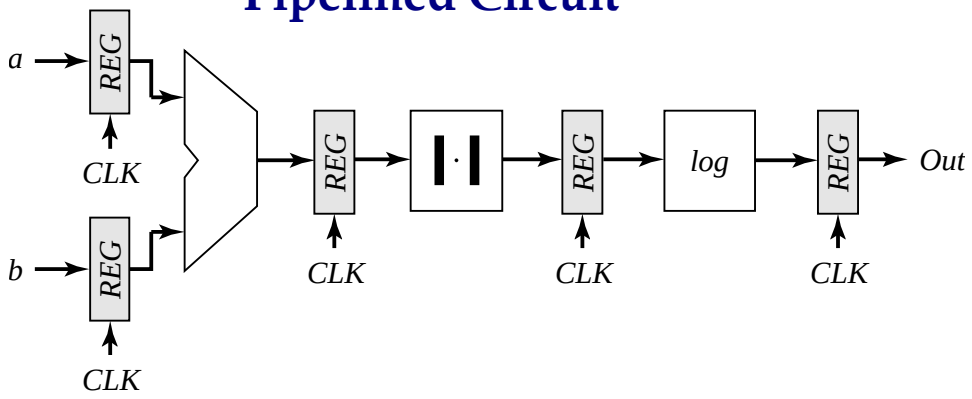
Widely used to accelerate the operation of datapaths in digital microprocessors…

Reference Circuit: computes log (|a+b|)



$$T_{\min} = t_{c-q} + t_{pd,\text{logic}} + t_{su}$$

Pipelined Circuit



$$T_{\min,pipe} = t_{c-q} + \max\left(t_{pd,adder} + t_{pd,abs} + t_{pd,\log}\right) + t_{su}$$

Clock Period	Adder	Absolute Value	Logarithm
1	$a_1 + b_1$		
2	$a_2 + b_2$	$\lvert a_1 + b_1 \rvert$	
3	$a_3 + b_3$	$\lvert a_2 + b_2 \rvert$	$\log(\lvert a_1 + b_1 \rvert)$
4	$a_4 + b_4$	$\lvert a_3 + b_3 \rvert$	$\log(\lvert a_2 + b_2 \rvert)$
5	$a_5 + b_5$	$\lvert a_4 + b_4 \rvert$	$\log(\lvert a_3 + b_3 \rvert)$

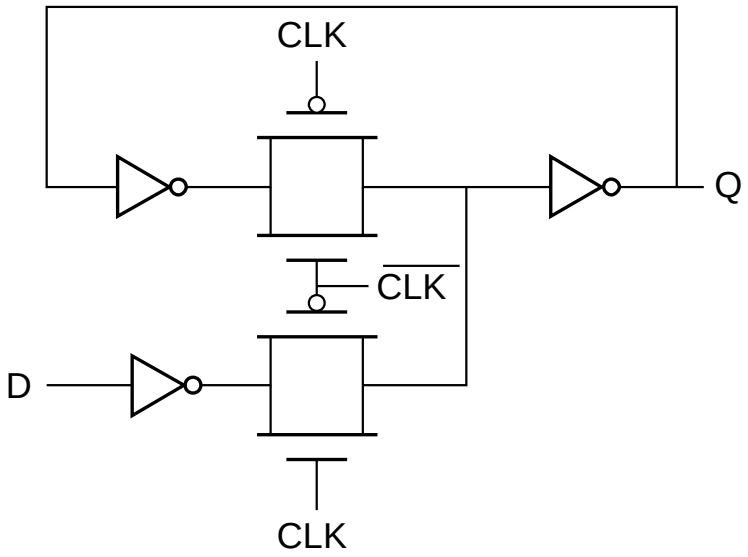
Computation of one set of input data spreads over several clock cycles.

Pipelining improves resource utilization and increases functional throughput.

Storage Mechanisms

A Static Latch

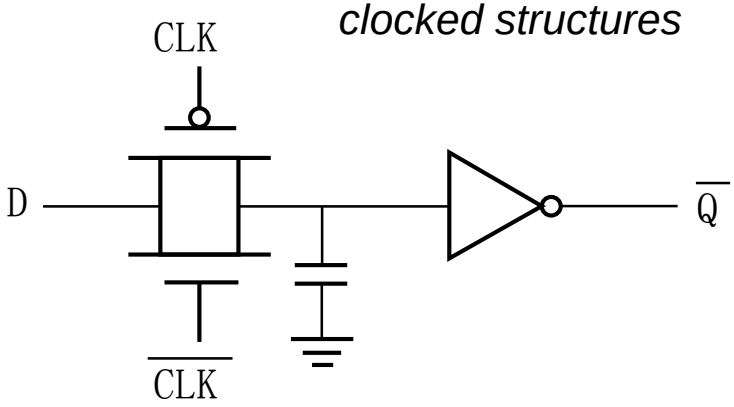
Uses a bistable memory device (FF), more complex



$$Q = \overline{Clk} \,.\, Q + Clk \,.\, D$$

Dynamic Latch (charge-based)

Useful for frequently clocked structures



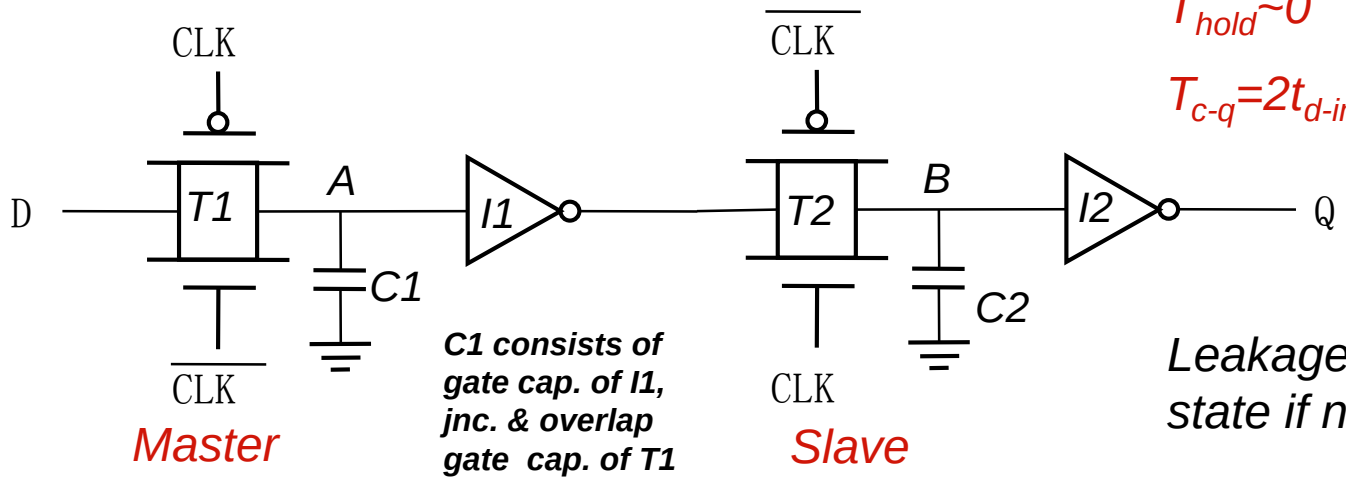
Stored value on a parasitic capacitor can remain for a limited time….few ms….need to refresh periodically

Dynamic X-gate Edge-Triggered Registers

$T_{su} = t_{dx\text{-}T1}$

$T_{hold} \sim 0$

$T_{c\text{-}q} = 2t_{d\text{-}inv} + t_{dx\_T2}$



Leakage will destroy the state if not refreshed

CLK=0: *D input is sampled at storage node 1, Slave stage in hold mode with node 2 in high-impedance state*

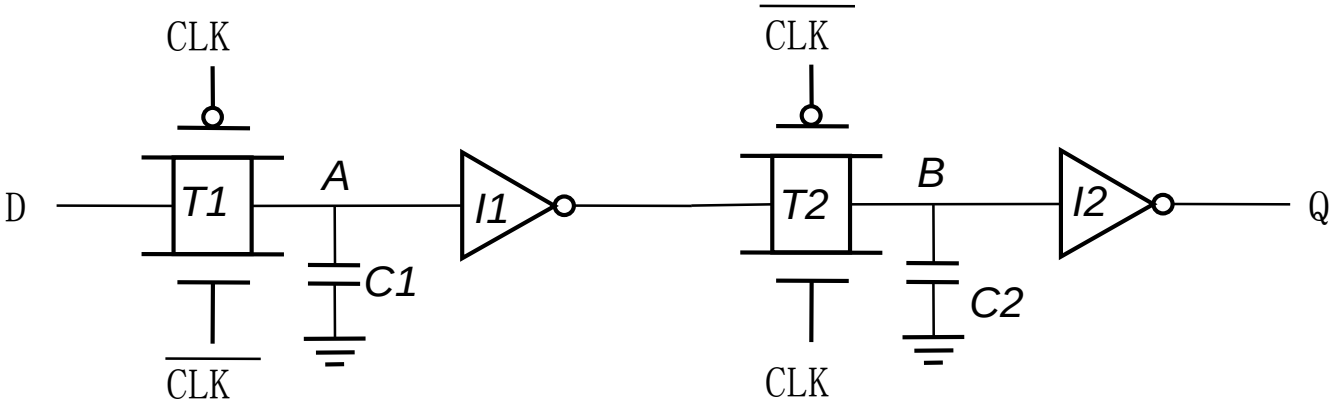
At the rising edge of CLK: *T2 is turned on and value sampled at node 1 right before the rising edge is copied to Q*

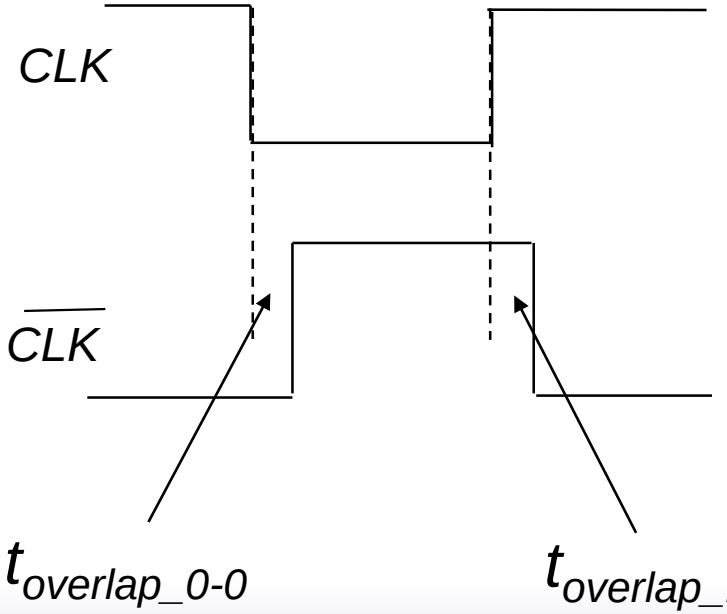
Very efficient: *requires only 8 transistors, can be made even simpler (6 transistors) using NMOS-only pass transistors*

Race condition *can occur due to CLK overlaps!*

Robustness *is also an issue….(state at the nodes can be distorted by injected noise)*

Impact of Non-overlapping Clocks…on +ve Edge Triggered Register





$$t_{overlap\_0\text{-}0} < t_{T1} + t_{I1} + t_{T2}$$

So that output Q doesn't change on the falling CLK edge….

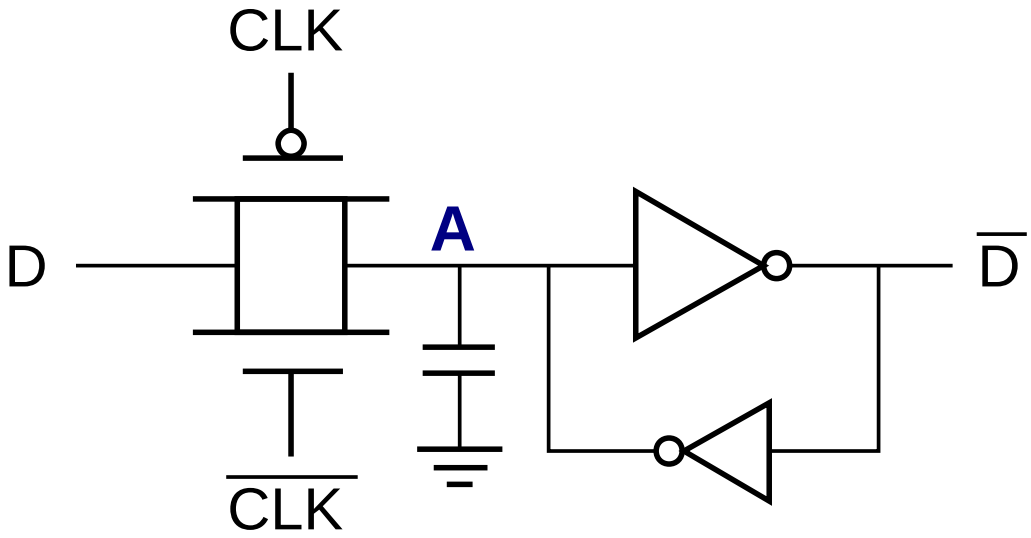
$$t_{hold} > t_{overlap\_1\text{-}1}$$

data must remain stable during the high overlap period

Making a Dynamic Latch Pseudo-Static

Problems with Dynamic Latches: *1) any signal net that is capacitively coupled to the internal storage node (such as A…) can inject significant noise and destroy the state 2) Leakage..during CLK gating or slowdown 3) internal dynamic node doesn't track variations in Vdd*

Problems of the dynamic register can be overcome by adding a weak feedback INV



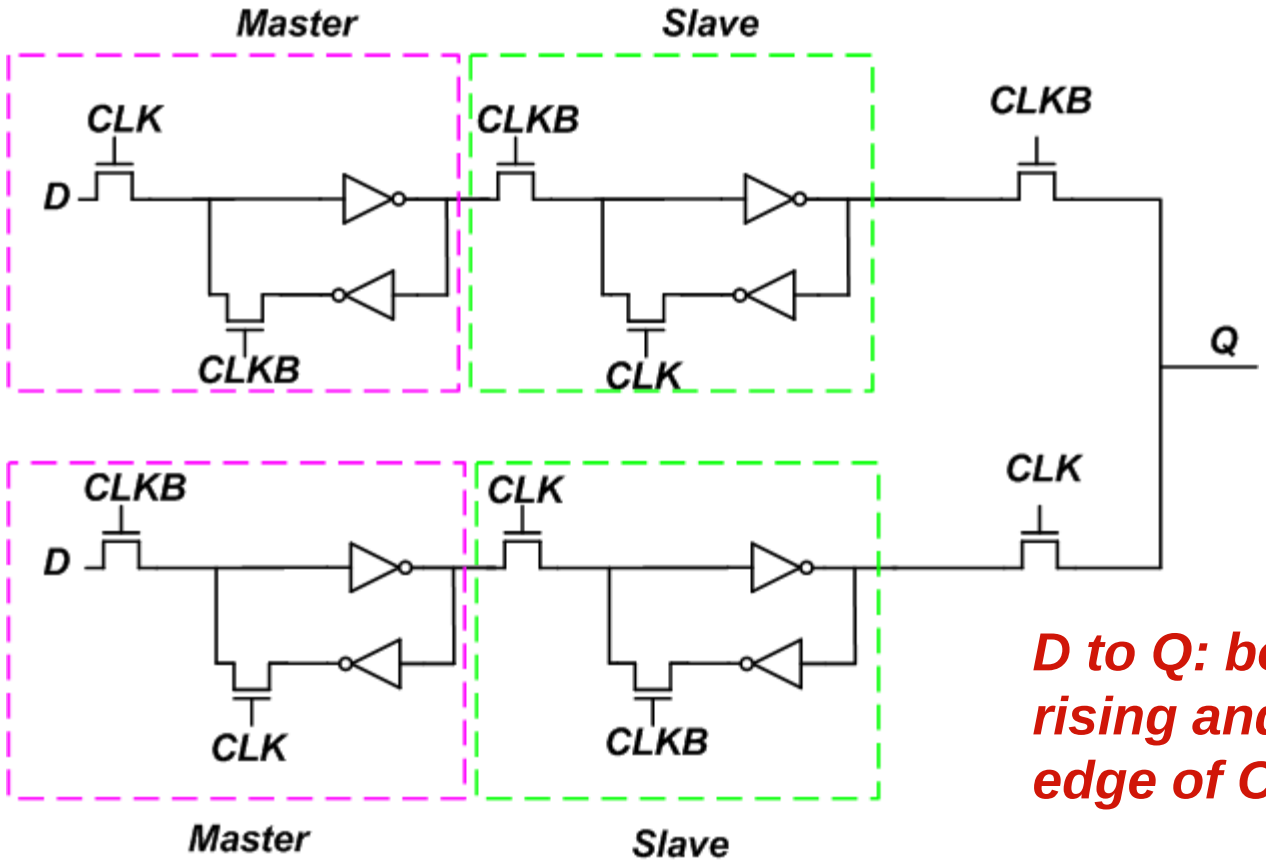
Increases delay slightly…but improves noise immunity

Most registers should be pseudo-static or static unless used in high-performance datapaths

Dual Edge-Triggered Master-Slave Flip-Flop

- Higher throughput
- Data is sampled both in positive-edge and negative edge of the clock
- Duty cycle of the clock should be 50%
- Setup time and hold time is important in both edges of the clock

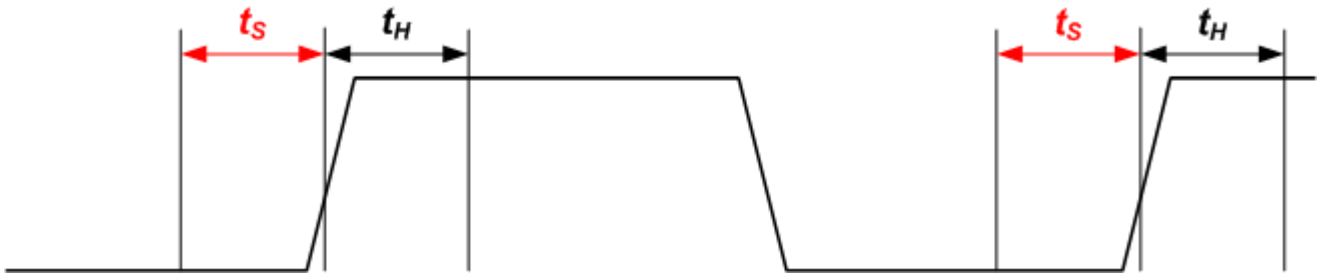
Dual Edge-Triggered Master-Slave Flip-Flop



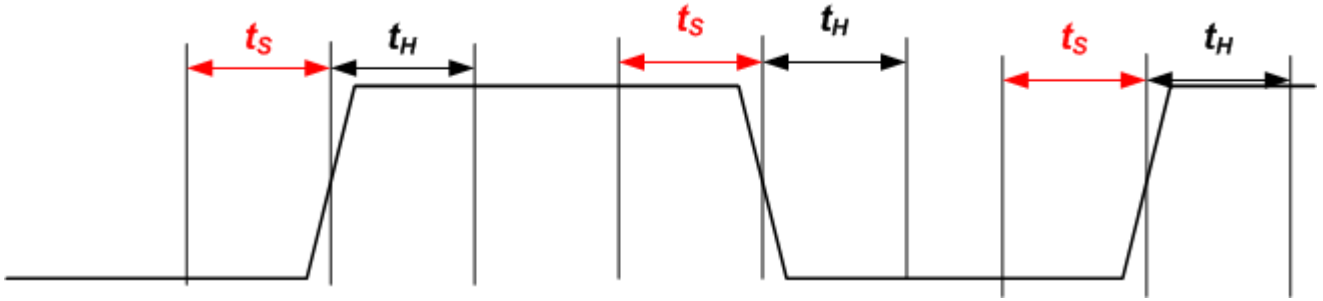
D to Q: both at the rising and falling edge of CLK

Single vs Dual Edge Triggered FF Timing

- Single Edge Triggered: $(T/2)_{CLK} > Max(t_S, t_H)$



- Double Edge Triggered: $(T/2)_{CLK} > t_S + t_H$



Note: lower frequency CLK giving same throughput, good for power savings

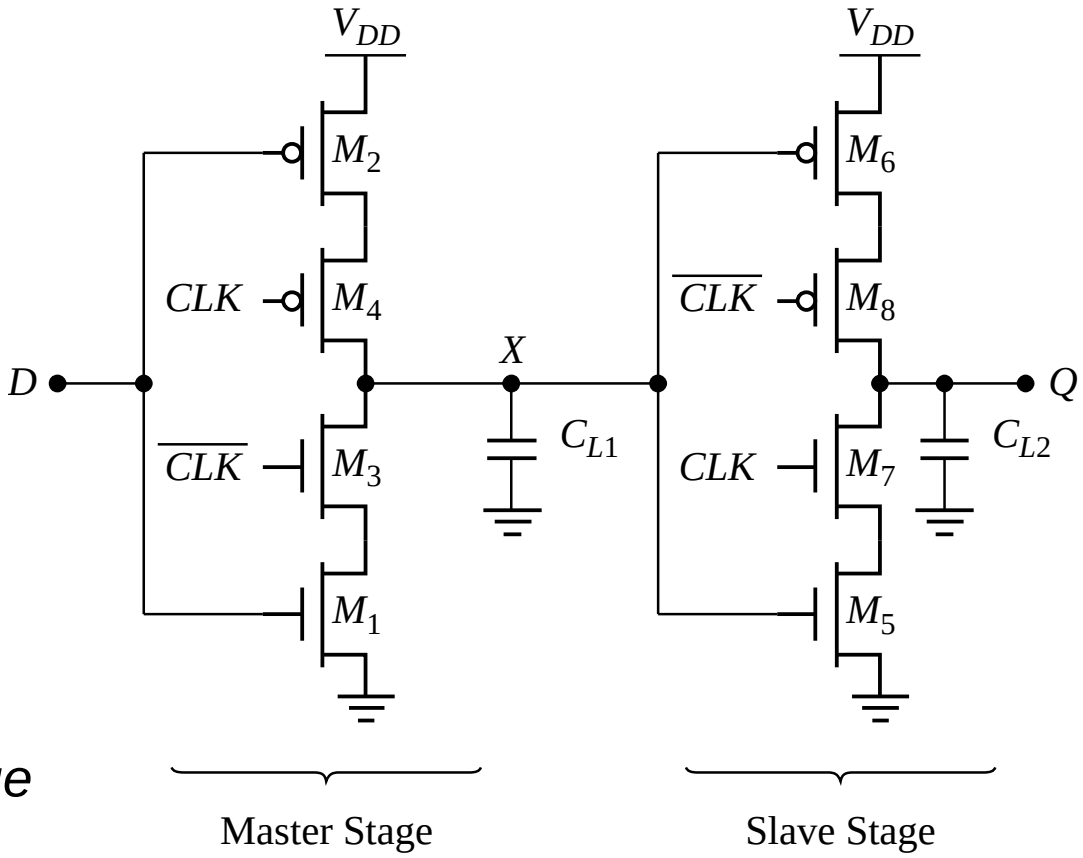
Other Latches/Registers: C²MOS

***C**locked **C**MOS* Register: +ve edge triggered Master-Slave FF

CLK=0:

first tri-state buffer turns on….and Master samples the inverted version of D

Q retains previous value on C_{L2}



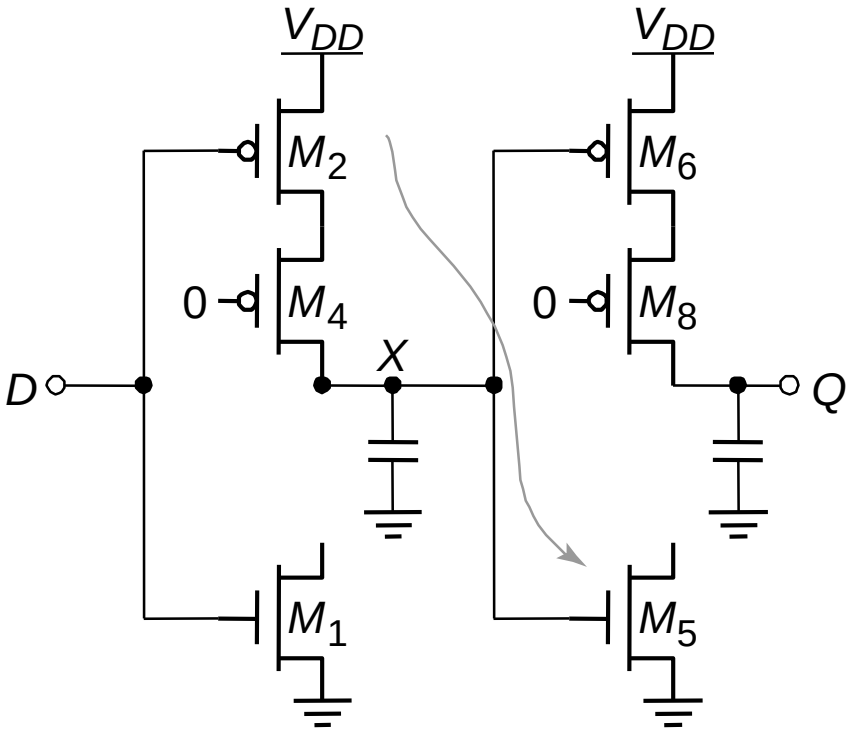
CLK=1:

Master is in hold mode…second stage turns on and C_{L1} propagates to the output Q

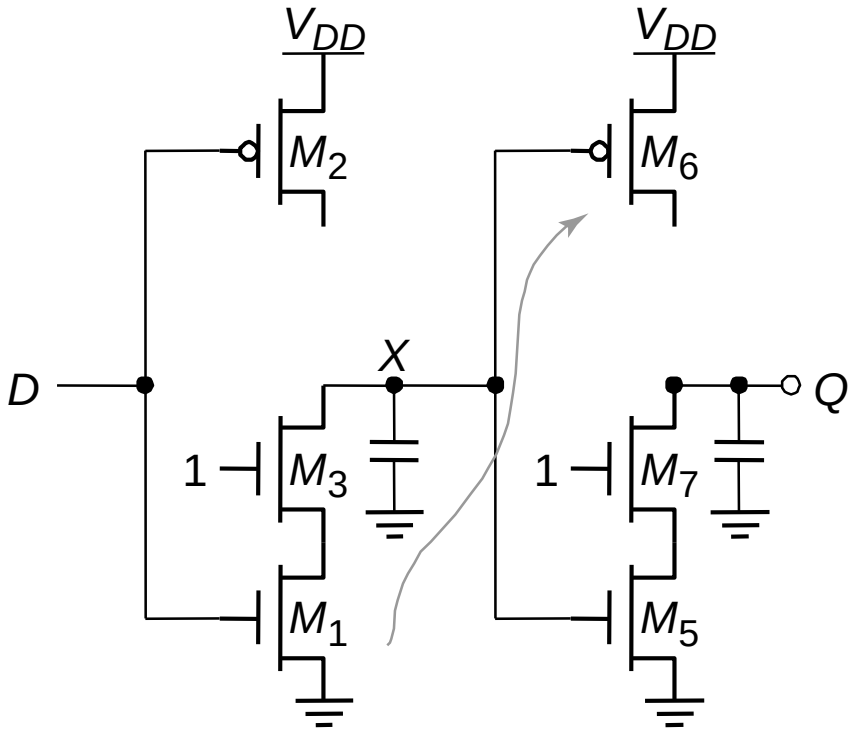
"Keepers" can be added to make circuit pseudo-static

C^2MOS: Insensitive to Clock-Overlap

Either PUN or PDN is activated by overlaps…but not both simultaneously….



(a) (0-0) overlap



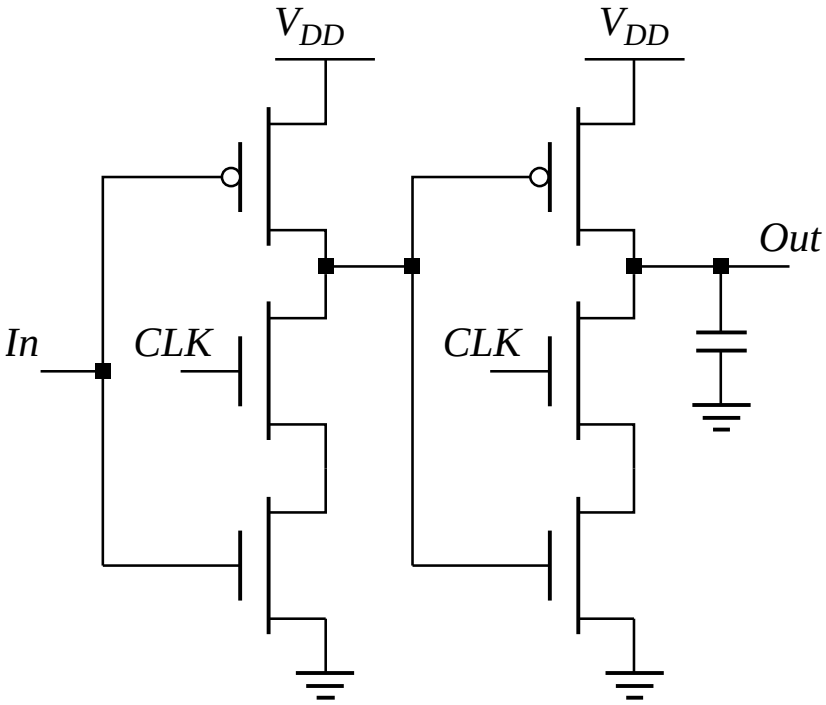
(b) (1-1) overlap

….hence any change in stored value at X cannot propagate to Q

Other Latches/Registers: TSPCR

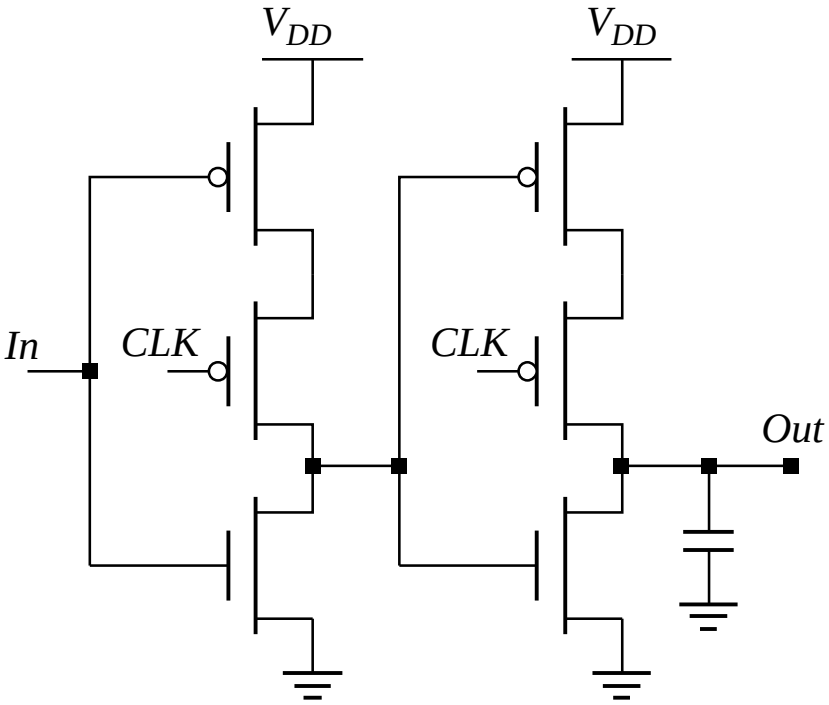
(True Single-Phase Clocked Register)

No signal can ever propagate from input to output…because of the dual stage approach….slightly increases number of transistors (=12)



Positive latch

(transparent when CLK= 1, when CLK=0, both inverters are disabled...latch is in hold mode)

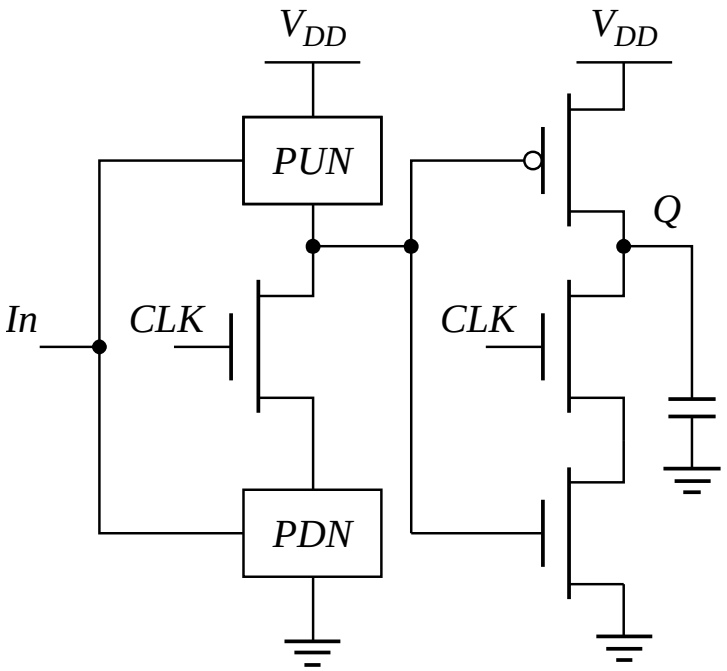


Negative latch

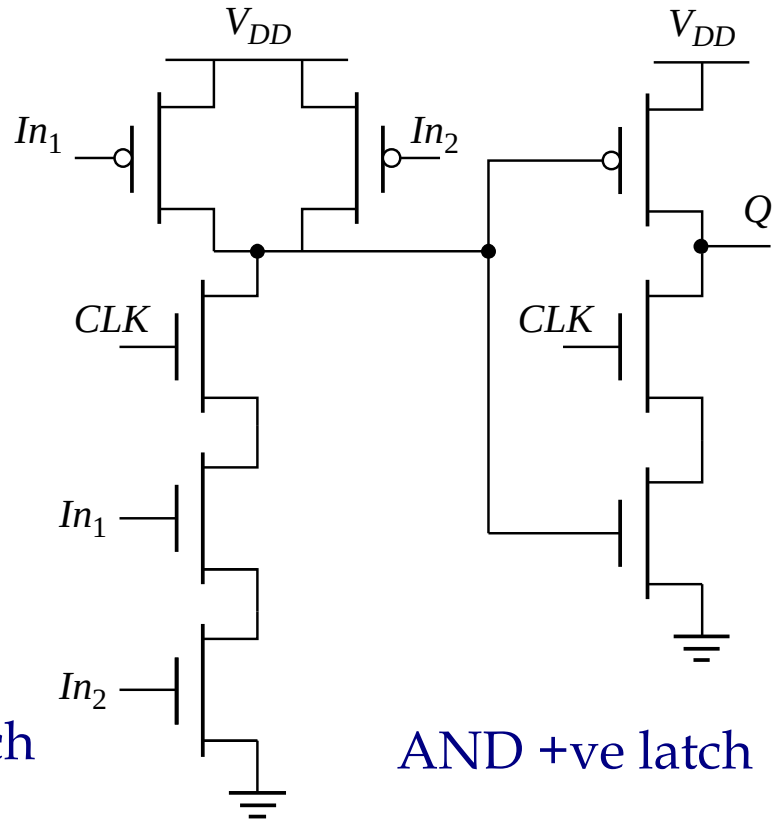
(transparent when CLK= 0)

Including Logic in TSPC

Reduces delay overhead associated with the latches…..



Example: logic inside the +ve latch



AND +ve latch

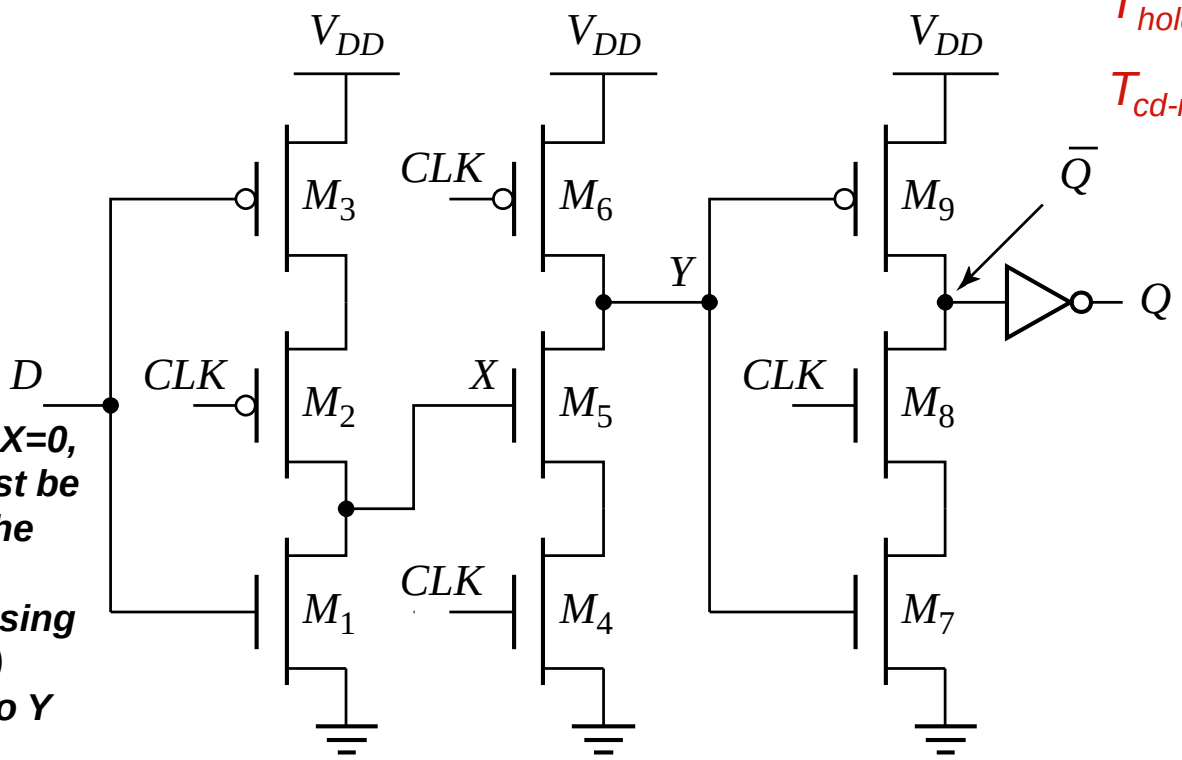
Note: set-up time increases a bit…but overall performance (T_{CLK}) is improved….used in the design of DEC Alpha Microprocessors + other HP processors

A Specialized TSPC Edge-Triggered Register

$T_{su} = t_{d\text{-}inv}$

$T_{hold} < 1x\ t_{d\text{-}inv}$

$T_{cd\text{-}reg} = 3t_{d\text{-}inv}$



Note: if D=1, X=0, hence, D must be stable until the value on X (before the rising edge of CLK) propagates to Y (hold time)

On the rising edge of CLK dynamic-inverter M4-M6 evaluates…inv M7-M9 is ON, and passes the value of Y to Q

First (static) inverter: samples inverted version of D at X …..for CLK=0

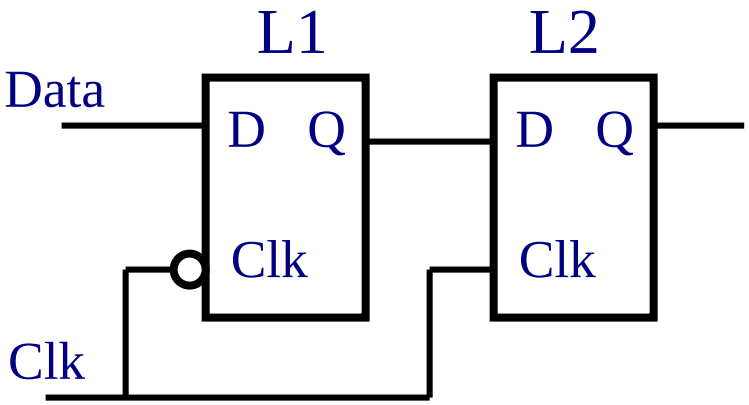
Second (dynamic) inverter: is in pre-charge mode…M6 pull up Y

Third (static) inverter: is in hold mode….M8 and M9 are off, Q is stable

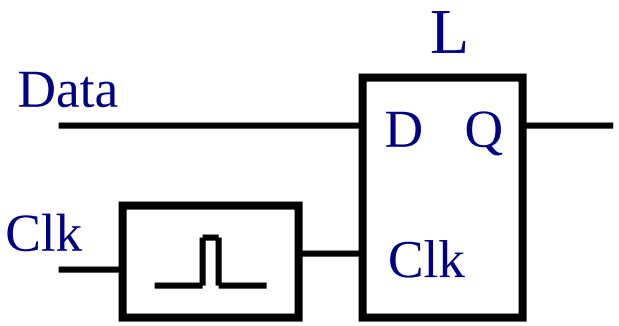
Pulse-Triggered Latches
An Alternative Approach

Ways to design an edge-triggered sequential cell:

Master-Slave Latches



Pulse-Triggered Latch



Pulsed Latches

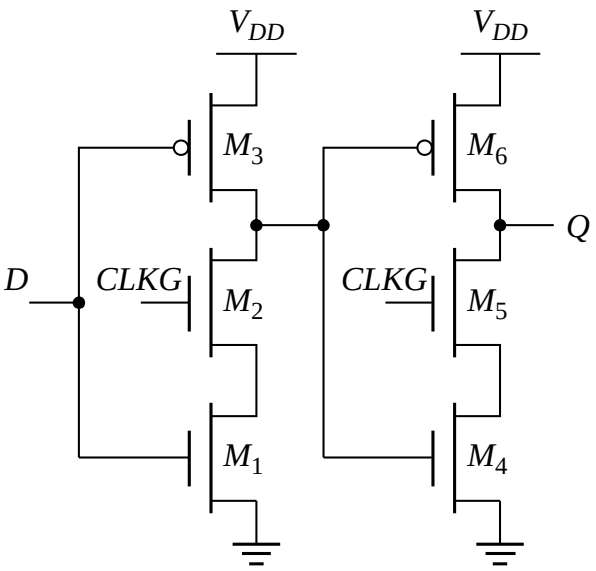
A short-pulse is constructed around the rising (or falling) edge of the CLK….

Pulse acts as the CLK input to a latch….sampling the input only in a small window…avoids RACE conditions

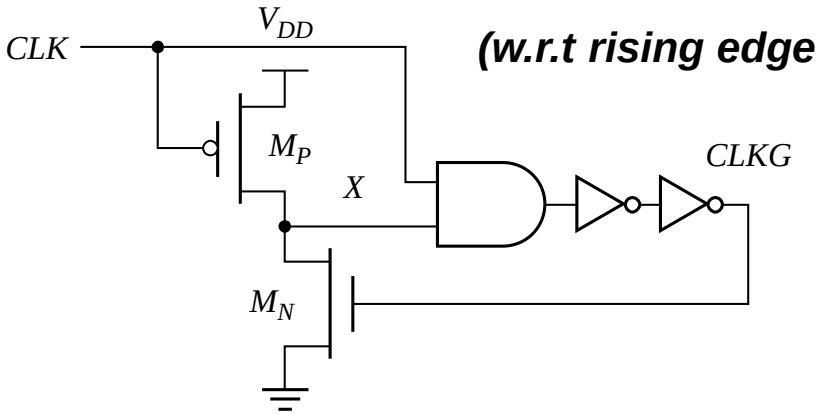
t_{su}=0;

t_h=pulse-width; t_{cq}=2 gate delays

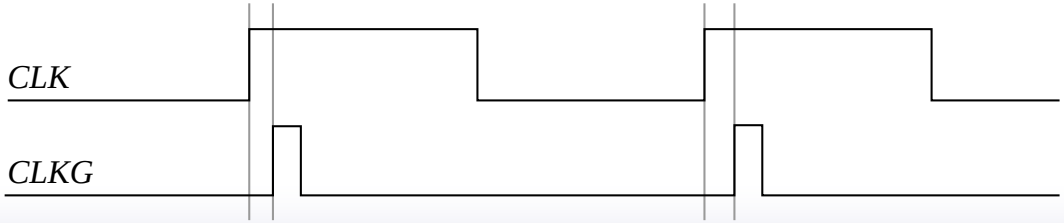
(w.r.t rising edge of CLKG)



(a) register



(b) glitch generation



(c) glitch clock

TIMING……is everything!!

Timing Classification of Digital Systems

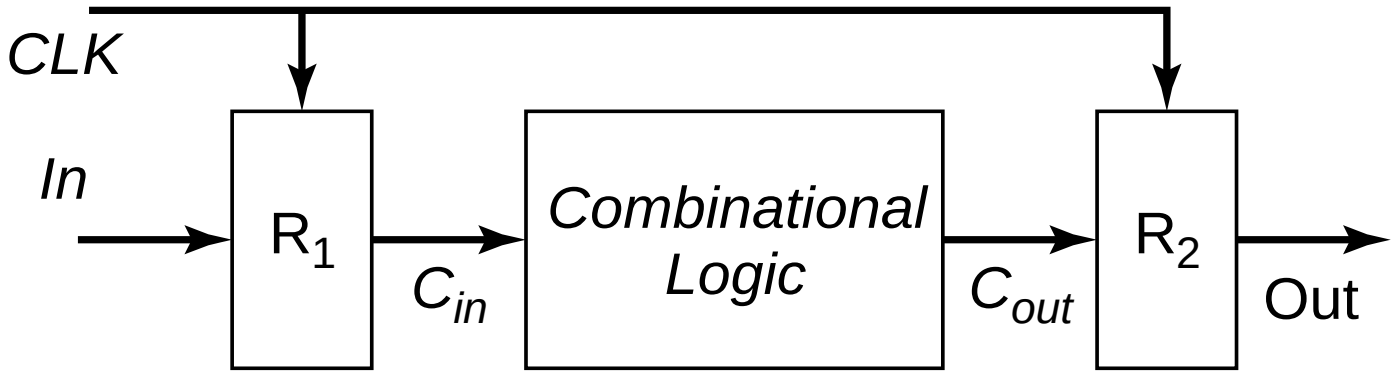
Synchronous: A signal that has exactly the same frequency as the local CLK and has a known fixed phase offset to that CLK

Mesochronous: A signal that has the same frequency as the local CLK but has an unknown phase offset w.r.t. that CLK.

Plesiochronous: A signal that has a “slightly different” frequency compared to that of the local CLK. This causes the phase difference to drift in time.

Asynchronous: A signal that has no fixed relationship w.r.t. that of the local CLK. Asynchronous signals can transition any time.

Synchronous Timing



The certainty period of the signal C_{out}---the period during which data are valid is synchronized with the system CLK. Hence, R2 can sample the data with confidence.

Mesochronous Timing

- If data are being passed between two different CLK domains, the data signal transmitted from the first module can have an unknown phase relationship to the CLK of the receiving module
- Not possible to directly sample the output data at the receiving module because of uncertainty in the phase offset
- A synchronizer is needed to synchronize the data signal with the receiving CLK

Plesiochronous Timing

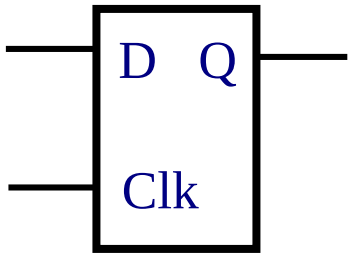
- Signal has slightly different frequency as compared to the local CLK---phase difference drifts over time
- Can arise due to the use of different CLK generators
- Practically, plesiochronous timing occurs in large distributed systems that involves long distance communications
- A timing recovery circuit is needed to ensure that all data are received

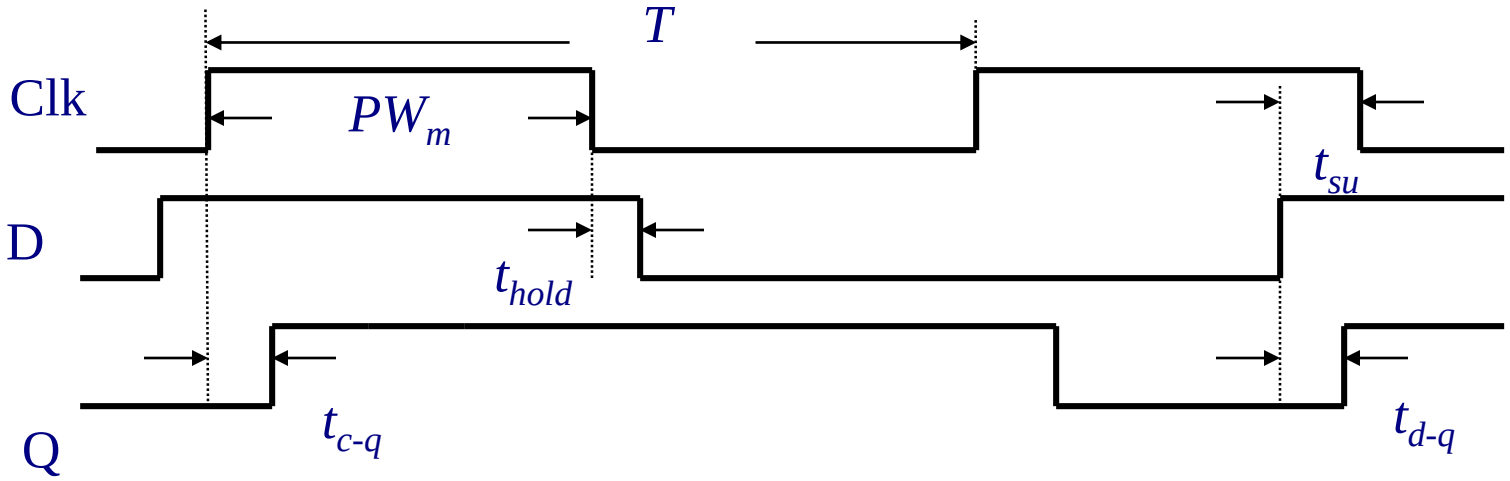
Asynchronous Timing

- Signals transition at arbitrary times
- Can synchronize asynchronous signals by detecting events and by introducing latencies into the data stream synchronized to a local CLK
- Alternatively, a self-timed asynchronous design approach may be adopted—communication between modules achieved not through a CLK but through a handshaking protocol that ensures proper ordering of operation
- Computations can be performed at speeds determined solely by the latency of the logic---no need to manage CLK skew!
- Increased complexity + overhead in communication, impacts performance. Also, CAD methodology is more complex.

Timing Definitions…

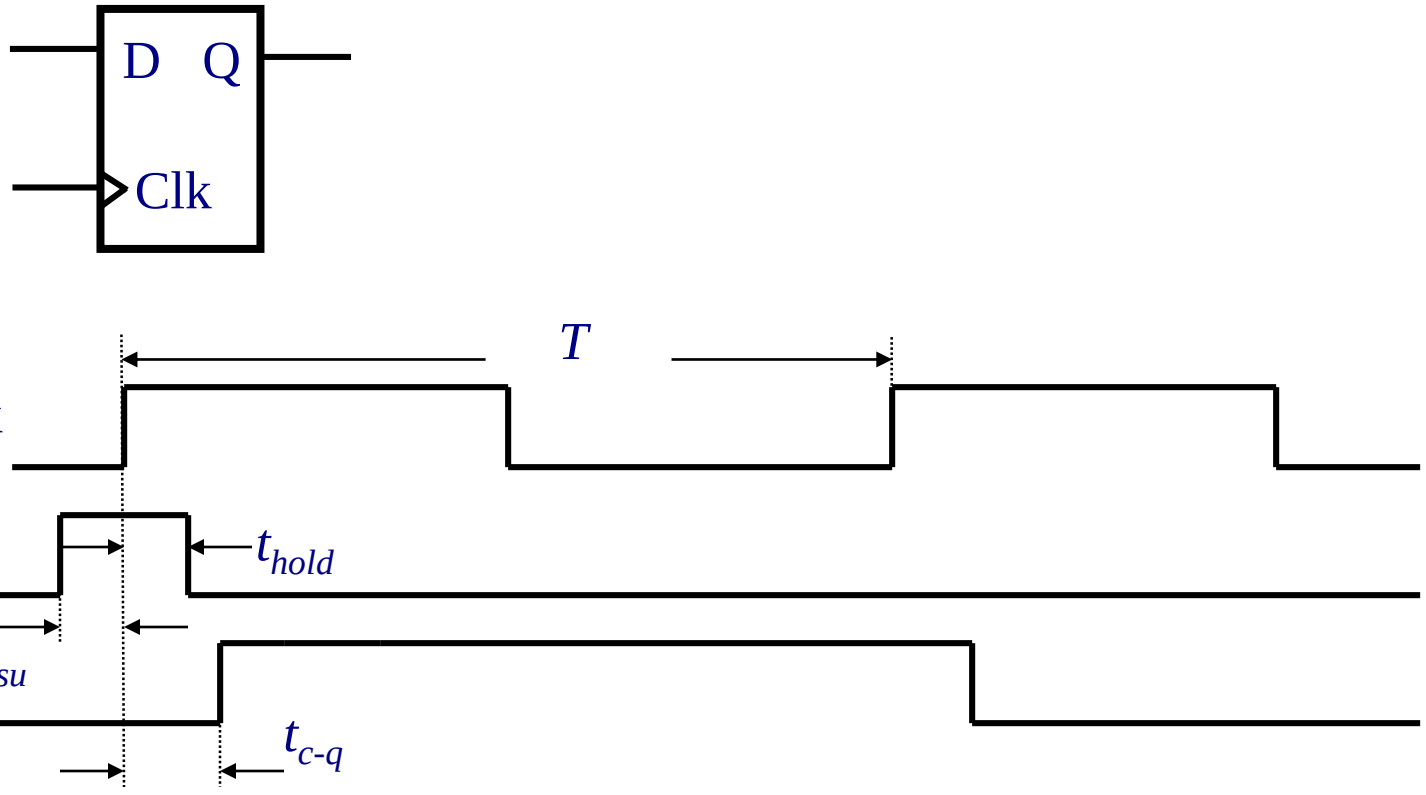
Latch Parameters





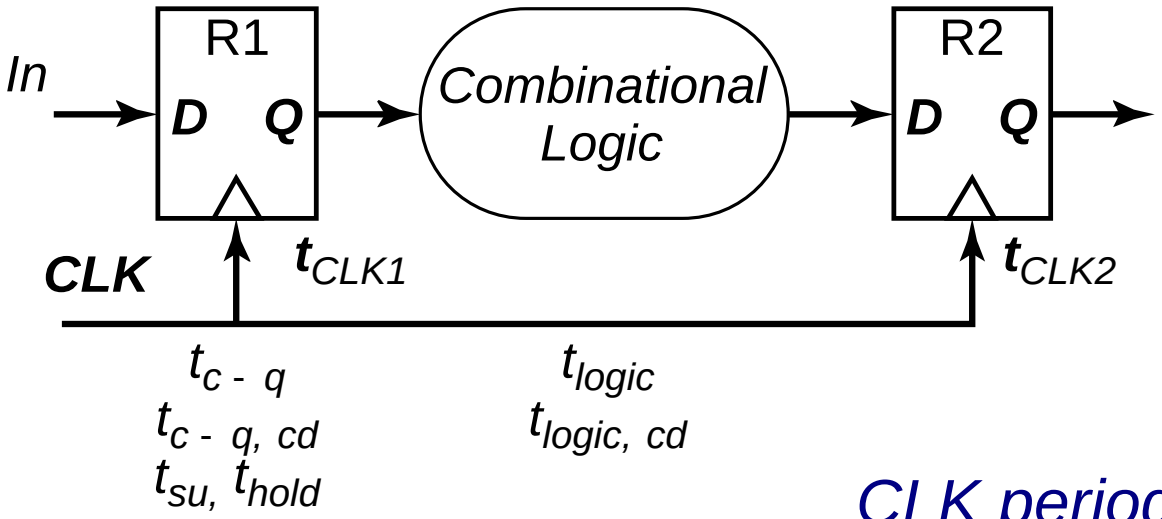
Delays can be different for rising and falling data transitions

Register Parameters



Delays can be different for rising and falling data transitions

Timing Constraints



CLK period constraint:

$$T_{CLK} > t_{c\text{-}q} + t_{logic} + t_{su}$$

Hold time constraint:

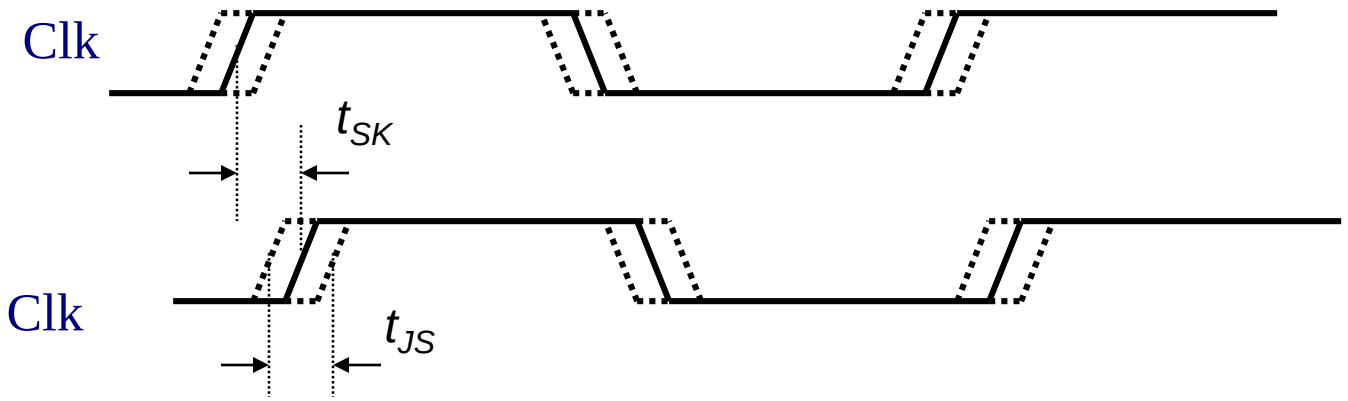
$$t_{(c\text{-}q,\,cd)} + t_{(logic,\,cd)} > t_{hold}$$

Worst case is when receiving edge arrives late
Race between data and clock

Clock Nonidealities

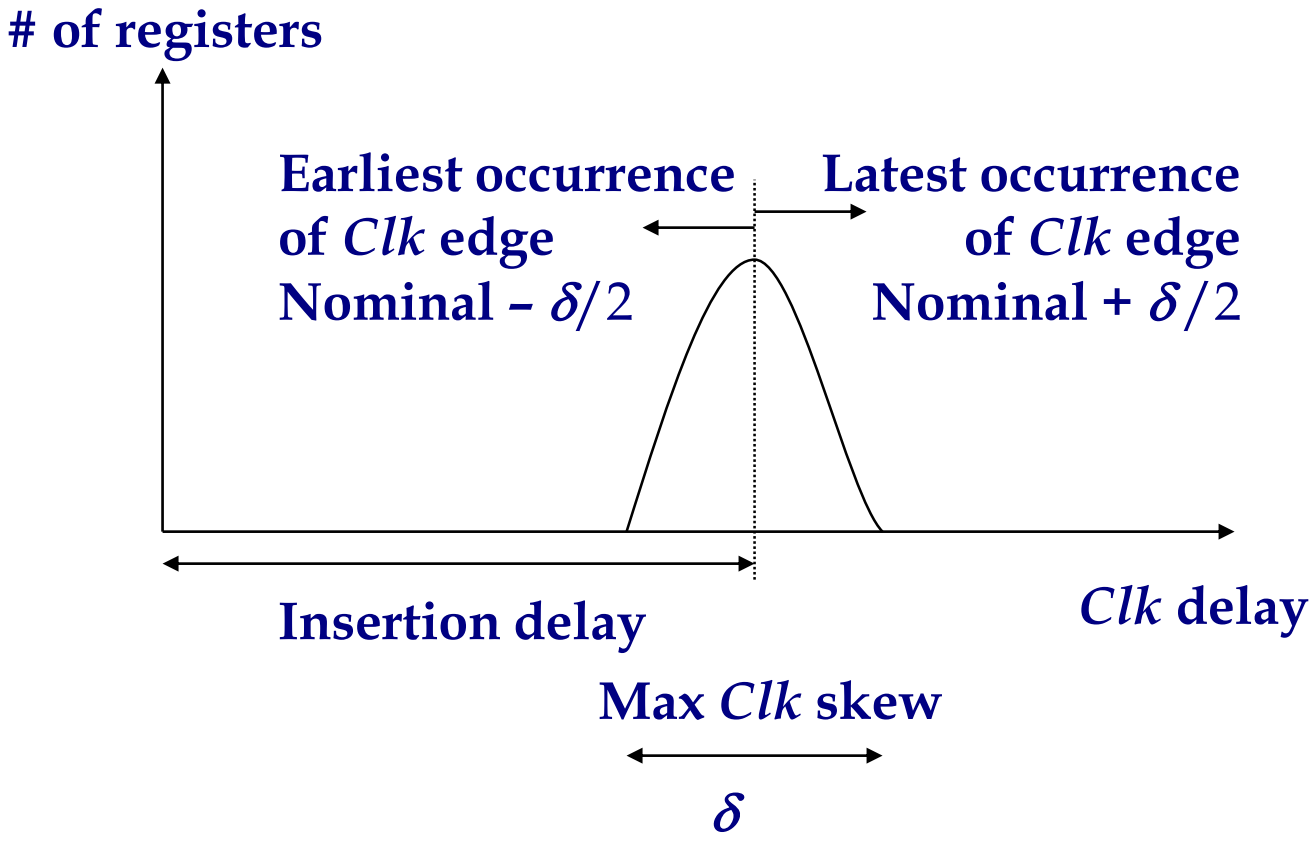
- **Clock skew**
  - Spatial variation in temporally equivalent clock edges; deterministic + random, t_{SK}
- **Clock jitter**
  - Temporal variations in consecutive edges of the clock signal; modulation + random noise
  - Cycle-to-cycle (short-term) t_{JS}
  - Long term t_{JL}
- **Variation of the pulse width**
  - Important for level sensitive clocking

Clock Skew and Jitter

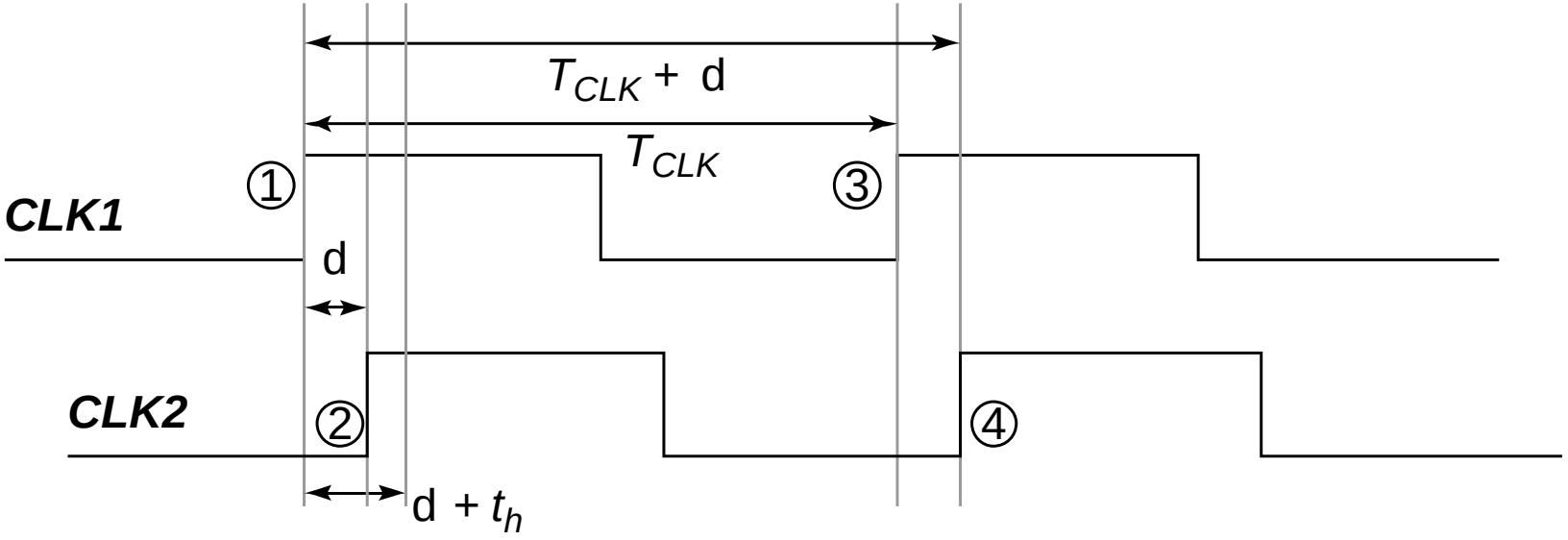


- Both skew and jitter affect the effective cycle time
- Only skew affects the race margin

Clock Skew

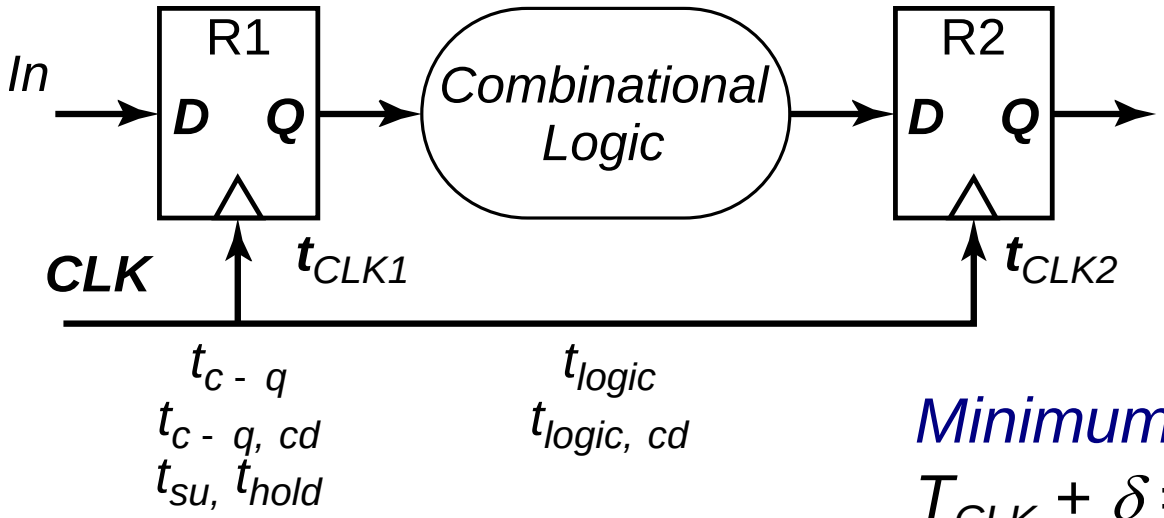

# of registers
Earliest occurrence of *Clk* edge Nominal – $\delta/2$
Latest occurrence of *Clk* edge Nominal + $\delta/2$
Insertion delay
Clk delay
Max *Clk* skew
δ

Positive Skew



Launching edge arrives before the receiving edge

Timing Constraints: +ve Skew



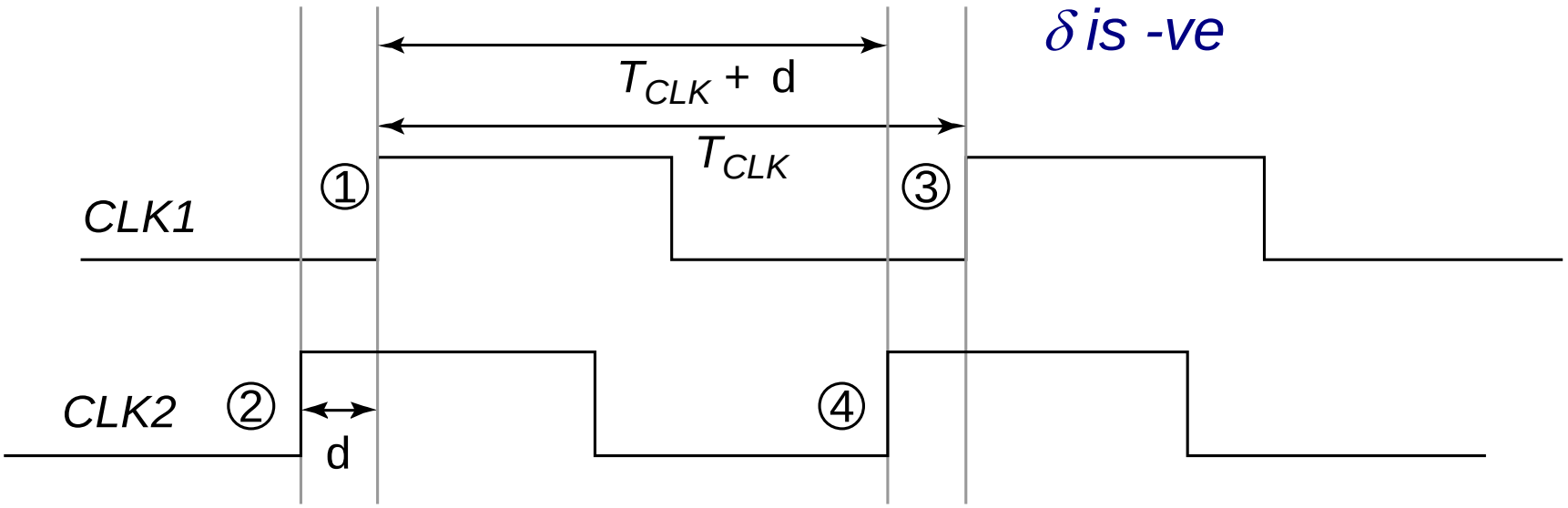
Minimum cycle time:

$$T_{CLK} + \delta = t_{c\text{-}q} + t_{su} + t_{logic}$$

CLK skew can improve performance?
Watch out for RACE problems….(if minimum logic delay is small, inputs to R2 can change before the CLK2 edge at②)

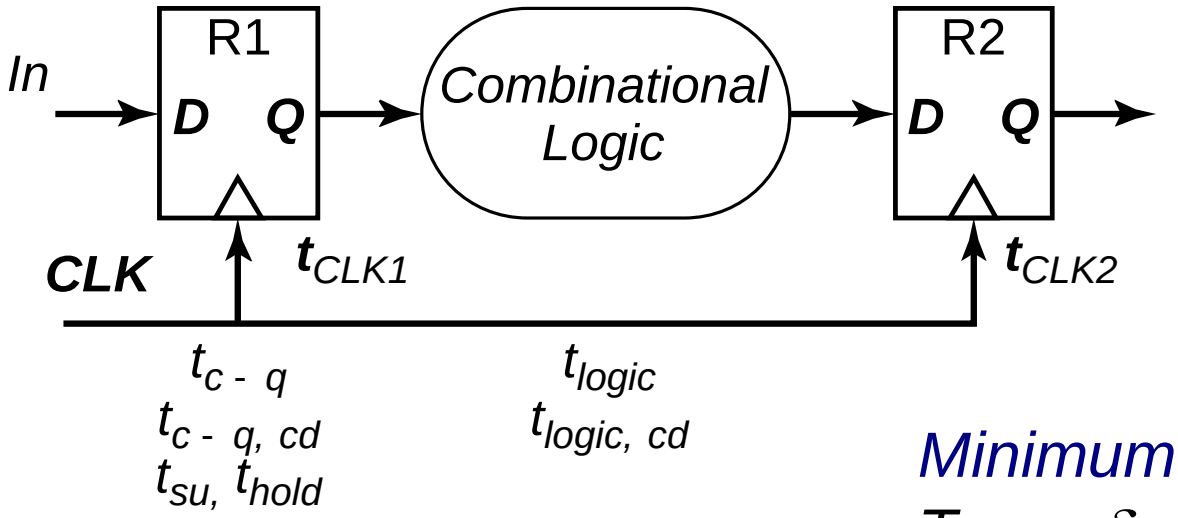
Minimum delay constraint: $\delta + t_{hold} < t_{c\text{-}q,cd} + t_{logic,cd}$

Negative Skew



Receiving edge arrives before the launching edge

Timing Constraints: -ve Skew



Minimum cycle time:

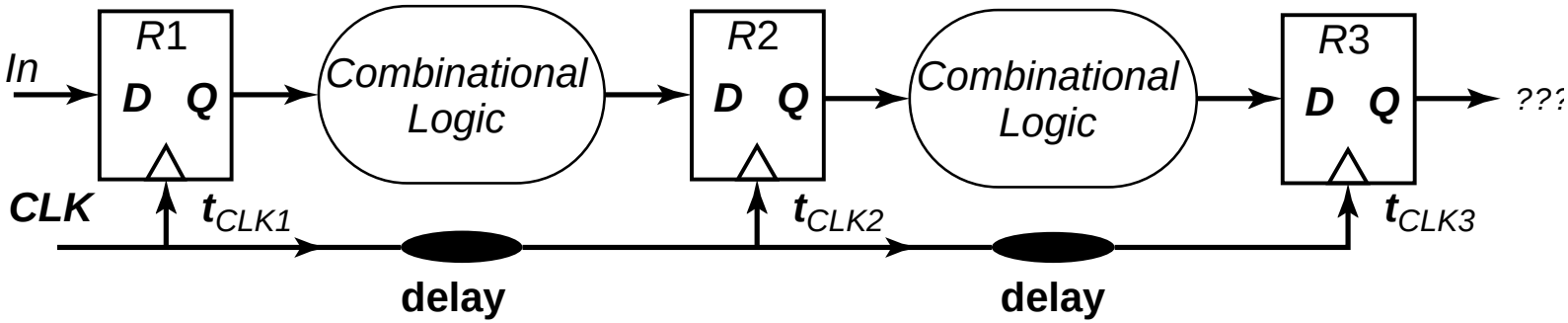
$$T_{CLK} - \delta = t_{c\text{-}q} + t_{su} + t_{logic}$$

-ve skew adversely affects the performance….

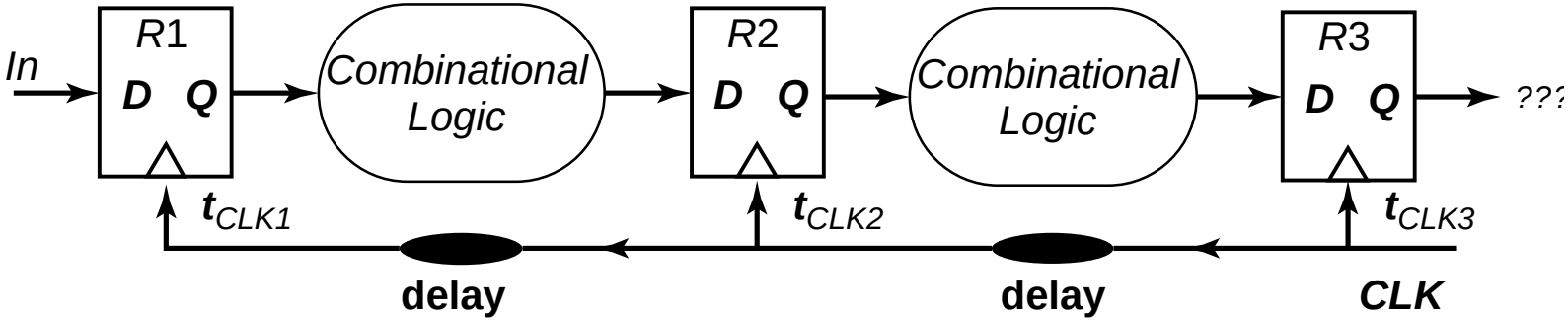
Minimum delay constraint: $\delta + t_{hold} < t_{c\text{-}q,cd} + t_{logic,cd}$*: eliminates RACE*

Positive and Negative Skew

CLK and data in the same direction



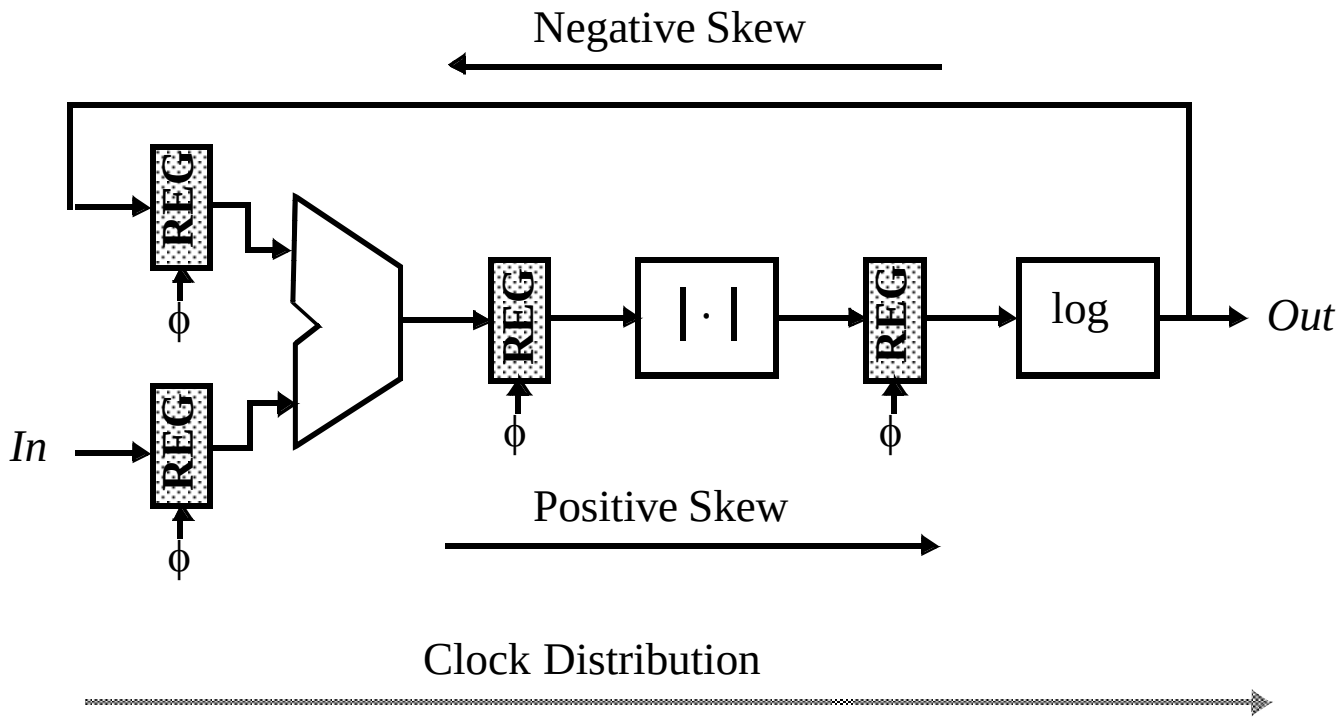
(a) Positive skew



(b) Negative skew

CLK and data in the opposite direction

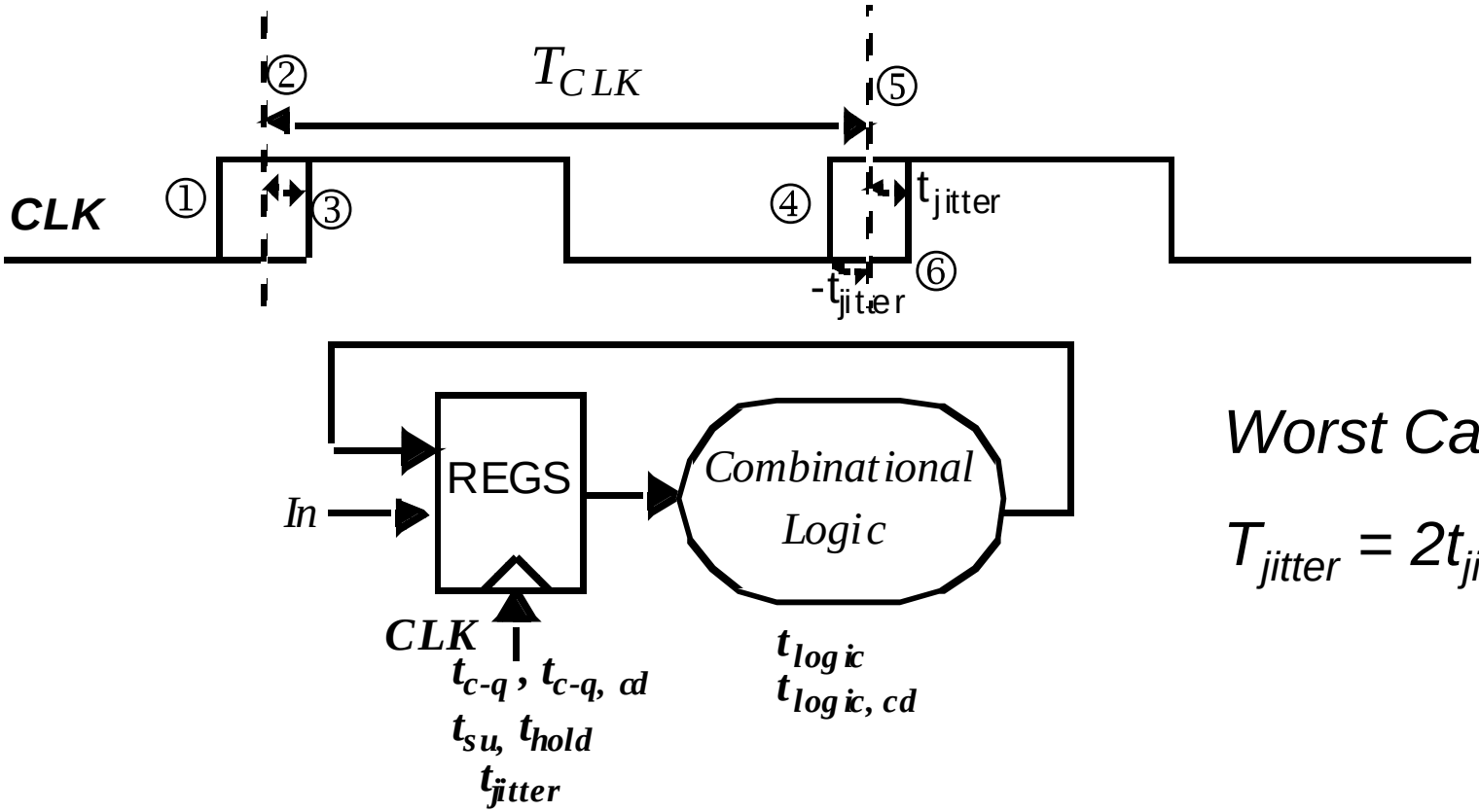
How to counter Clock Skew?



Data and Clock Routing

Must account for the worst case skew…

Impact of Jitter



Worst Case:

$T_{jitter} = 2t_{jitter}$

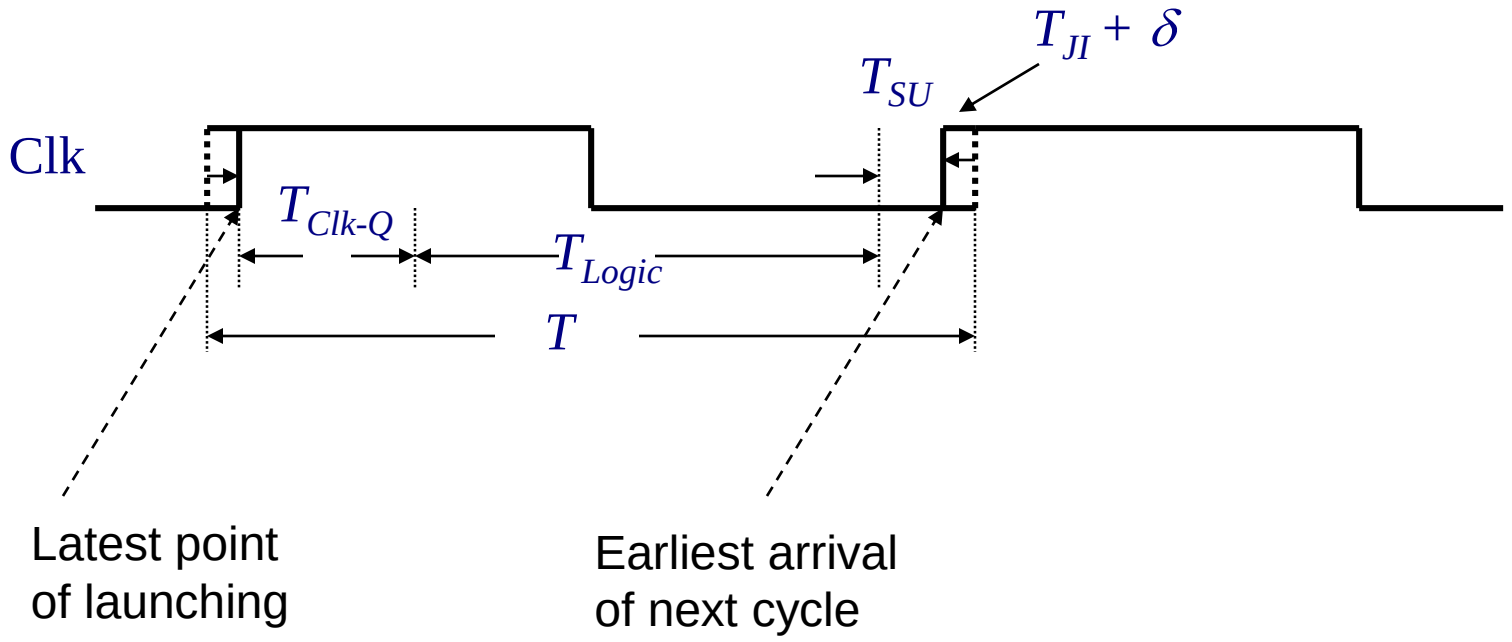
Absolute jitter = t_{jitter} = worst case variation of a CLK edge at a given location w.r.t. an ideally periodic reference CLK edge

Cycle-to-Cycle Jitter = T_{jitter} = time varying deviations of a single clock period relative to an ideal reference CLK:

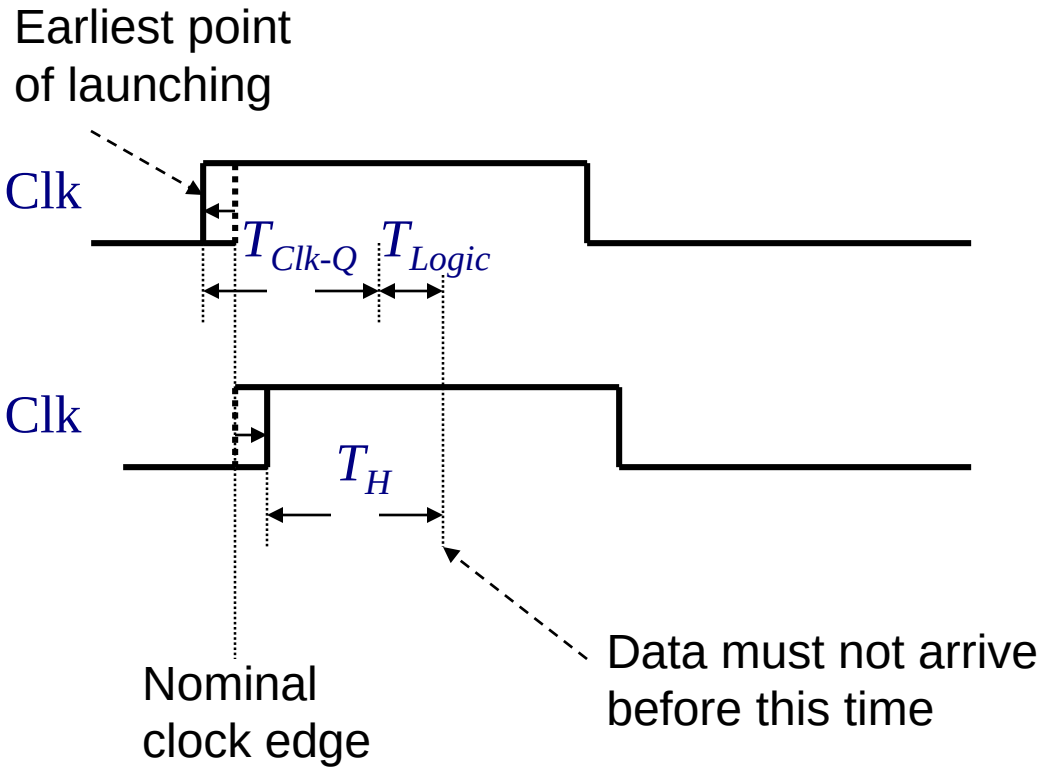
$$T^{i}_{jitter}(n) = t^{i}_{clk,n+1} + t^{i}_{clk,n} - T_{clk}$$

Arrival time of the nth CLK edge at i

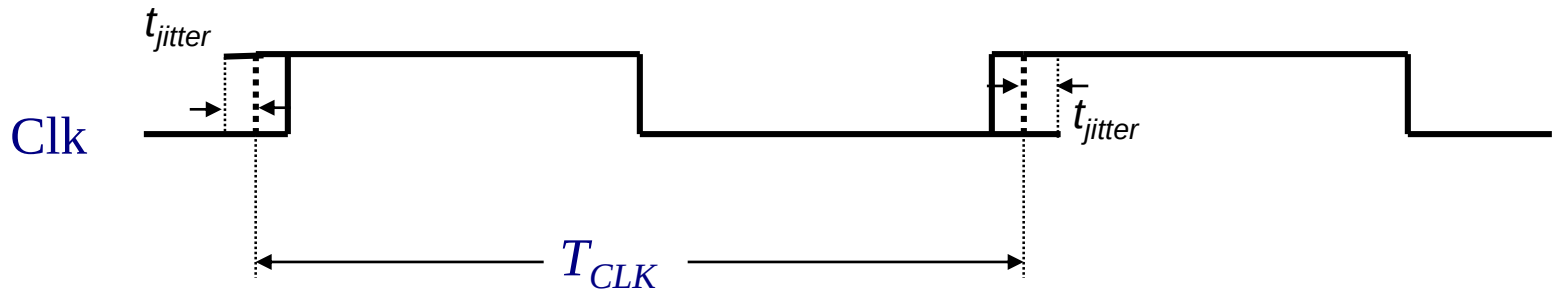
Longest Logic Path in Edge-Triggered Systems



Shortest Path



Impact of Jitter on Clock Constraints in Edge-Triggered Systems



Jitter directly impacts the performance of sequential systems:

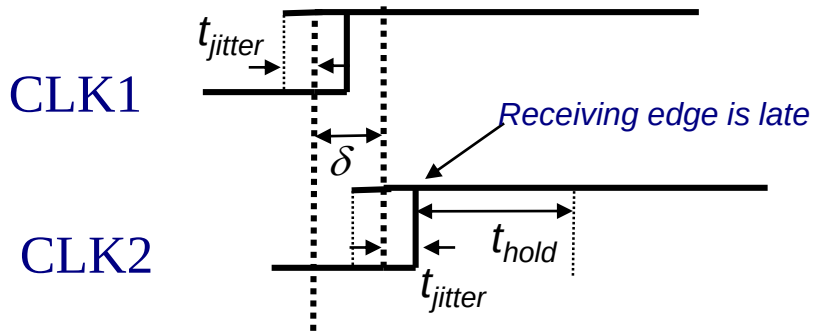
$$T_{CLK} - 2t_{jitter} > t_{c\text{-}q} + t_{logic} + t_{su}$$

Combined effects of Skew and Jitter:

$$T_{CLK} + \delta - 2\,t_{jitter} > t_{c\text{-}q} + t_{logic} + t_{su}$$

Skew can be either positive or negative. +ve skew improves performance, jitter always degrades performance

Impact of Jitter and Skew on Minimum Delay Constraints in Edge-Triggered Systems



If launching edge (CLK1) is early and receiving edge (CLK2) is late:

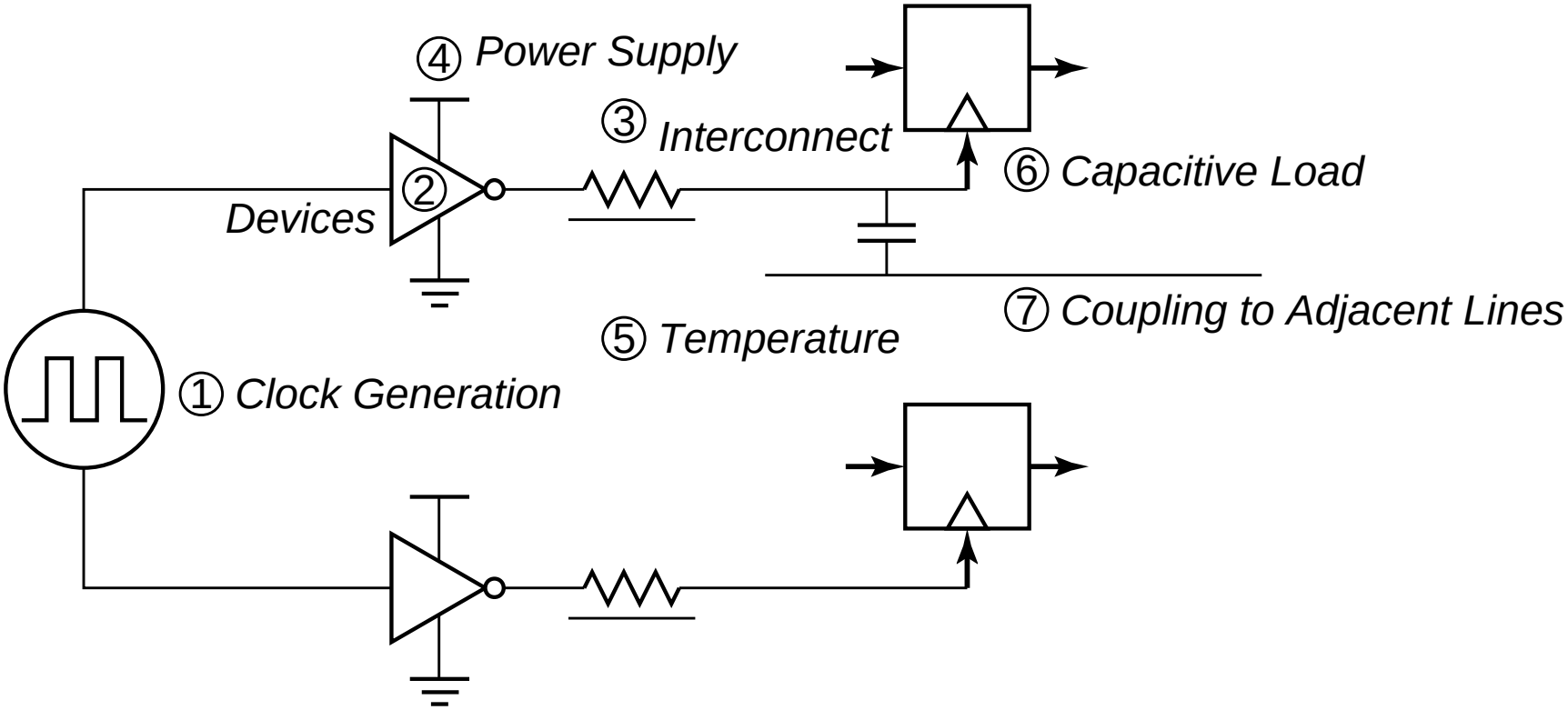
$$t_{hold} + \delta + 2t_{jitter} < t_{c\text{-}q,\,cd} + t_{logic,\,cd}$$

Minimum logic delay constraint:

$$\delta < t_{c\text{-}q,cd} + t_{logic,cd} - t_{hold} - 2t_{jitter}$$

Acceptable skew is reduced by the jitter of the two signals

Clock Uncertainties

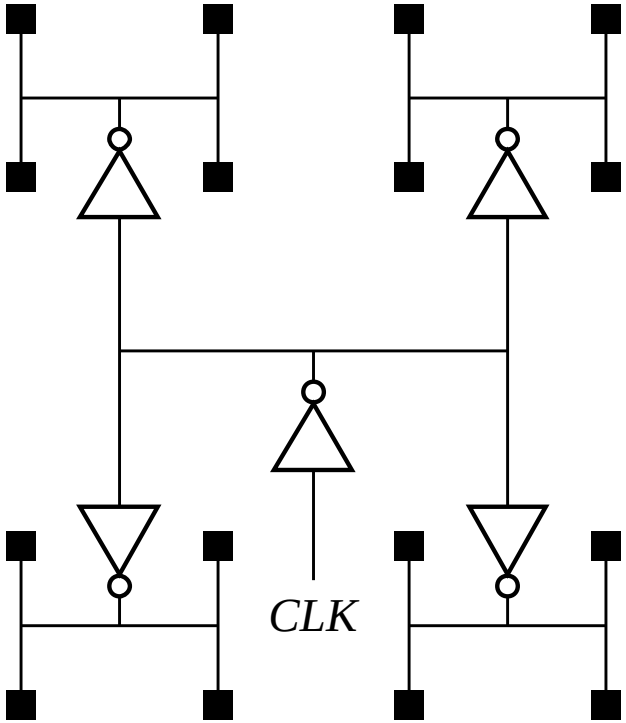


Sources of clock uncertainty

Clock Design Issues…

Clock Distribution

H-tree

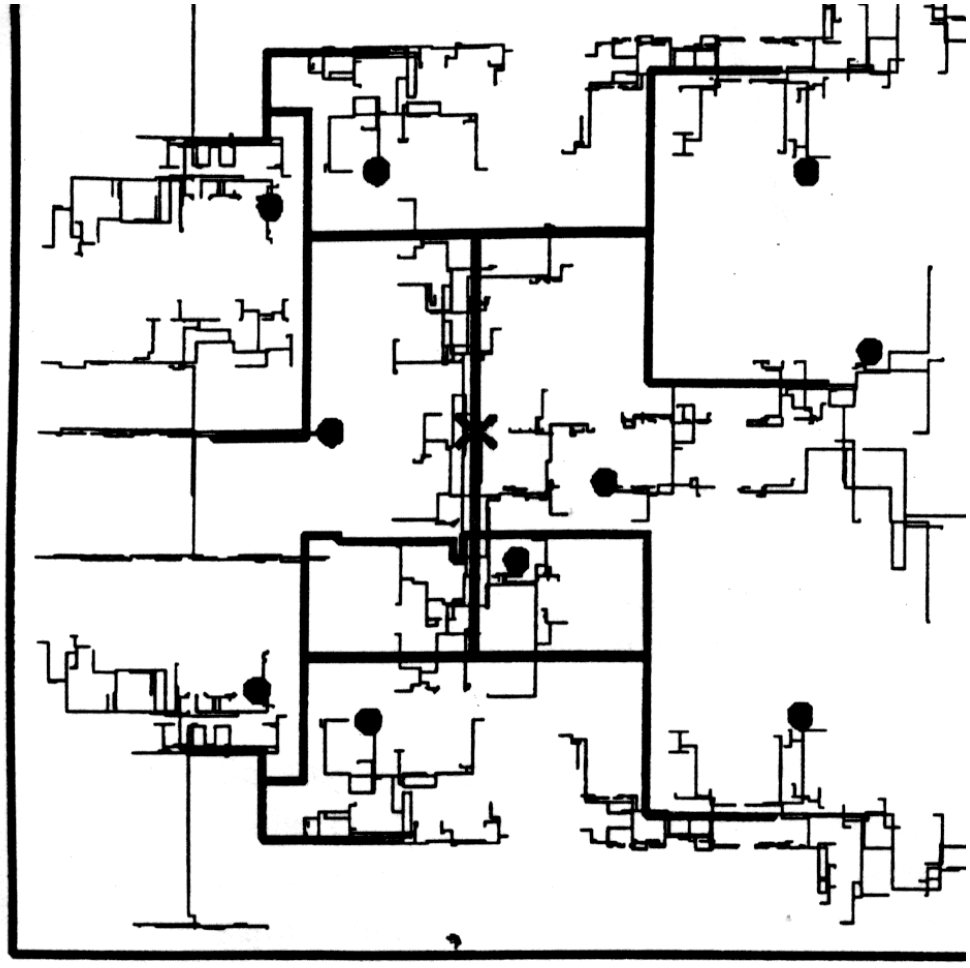


Each path must be balanced: equal interconnect and device load

Difficult to achieve due to parameter variations

Clock is distributed in a tree-like fashion

More realistic H-tree



Matched RC Trees

Doesn't not require a regular physical structure

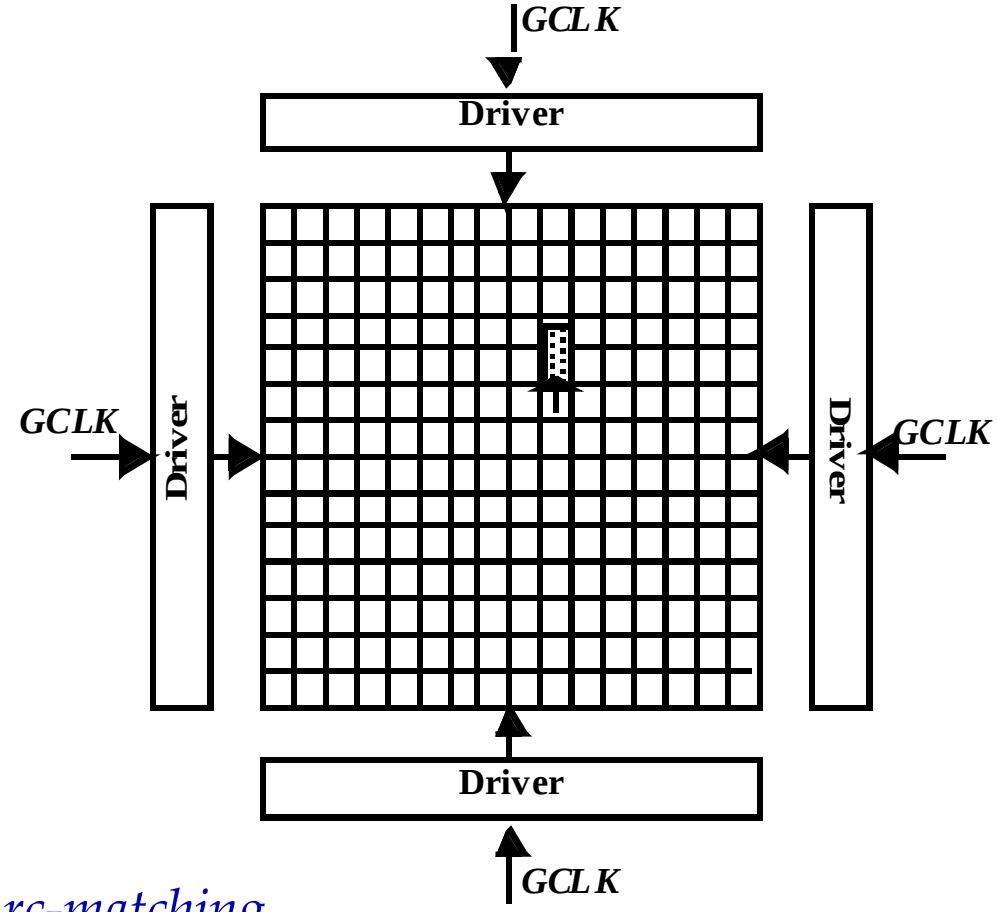
interconnections carrying CLK signals to different sub-blocks are of equal length

Chip is partitioned into balanced load segments (tiles)

Global CLK driver distributes the CLK to the tile drivers located at the dots in the figure

[Restle98]

The Grid System



- *No rc-matching*
- *Large power*

Allows for late design changes….CLK is more easily accessible

Delay from the final driver to each load is not matched

Absolute delay is minimized

Large power dissipation: due to excess interconnects

Example: DEC Alpha 21164

Clock Frequency: 300 MHz - 9.3 Million Transistors

0.5 um CMOS process

Total Clock Load: 3.75 nF

extensive use of dynamic logic….

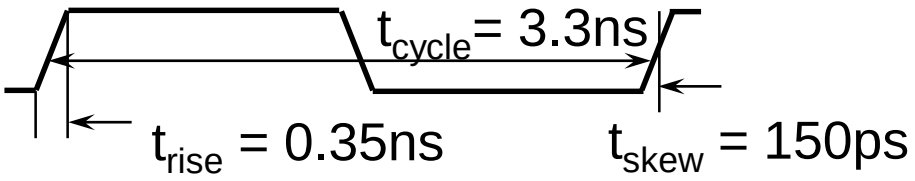
Power in Clock Distribution network : 20 W (out of 50)

Uses Two Level Clock Distribution:

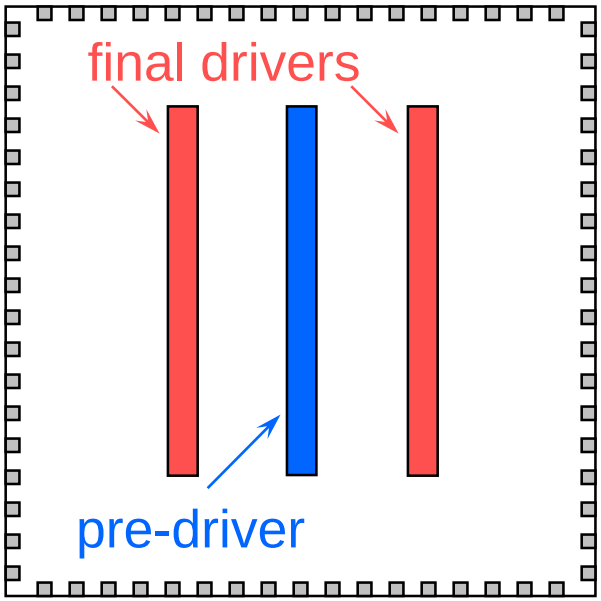
- **Single 6-stage driver at center of chip**
- **Secondary buffers drive left and right side clock grid in Metal3 and Metal4**

Total driver size: 58 cm!

21164 Clocking

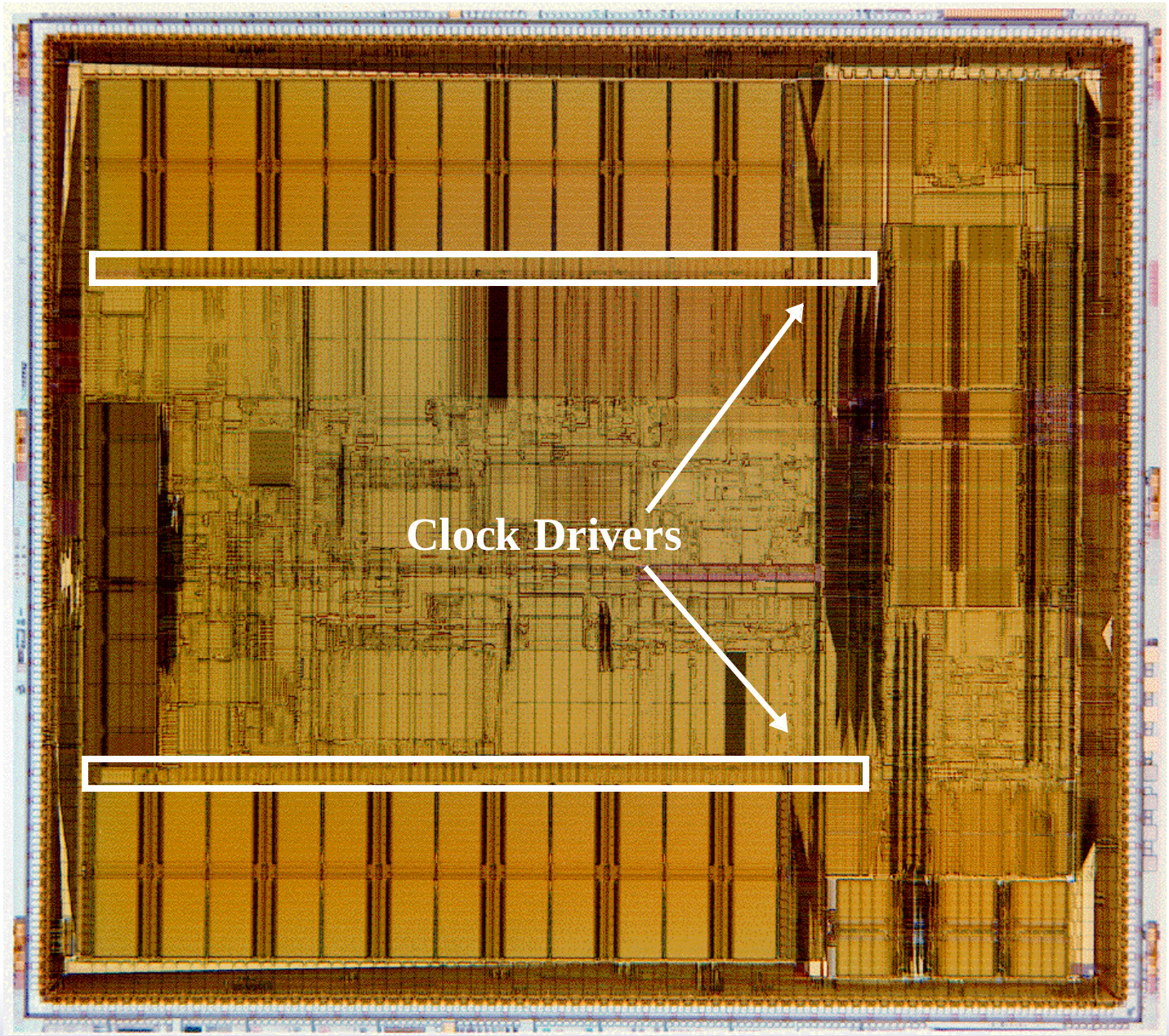


Clock waveform

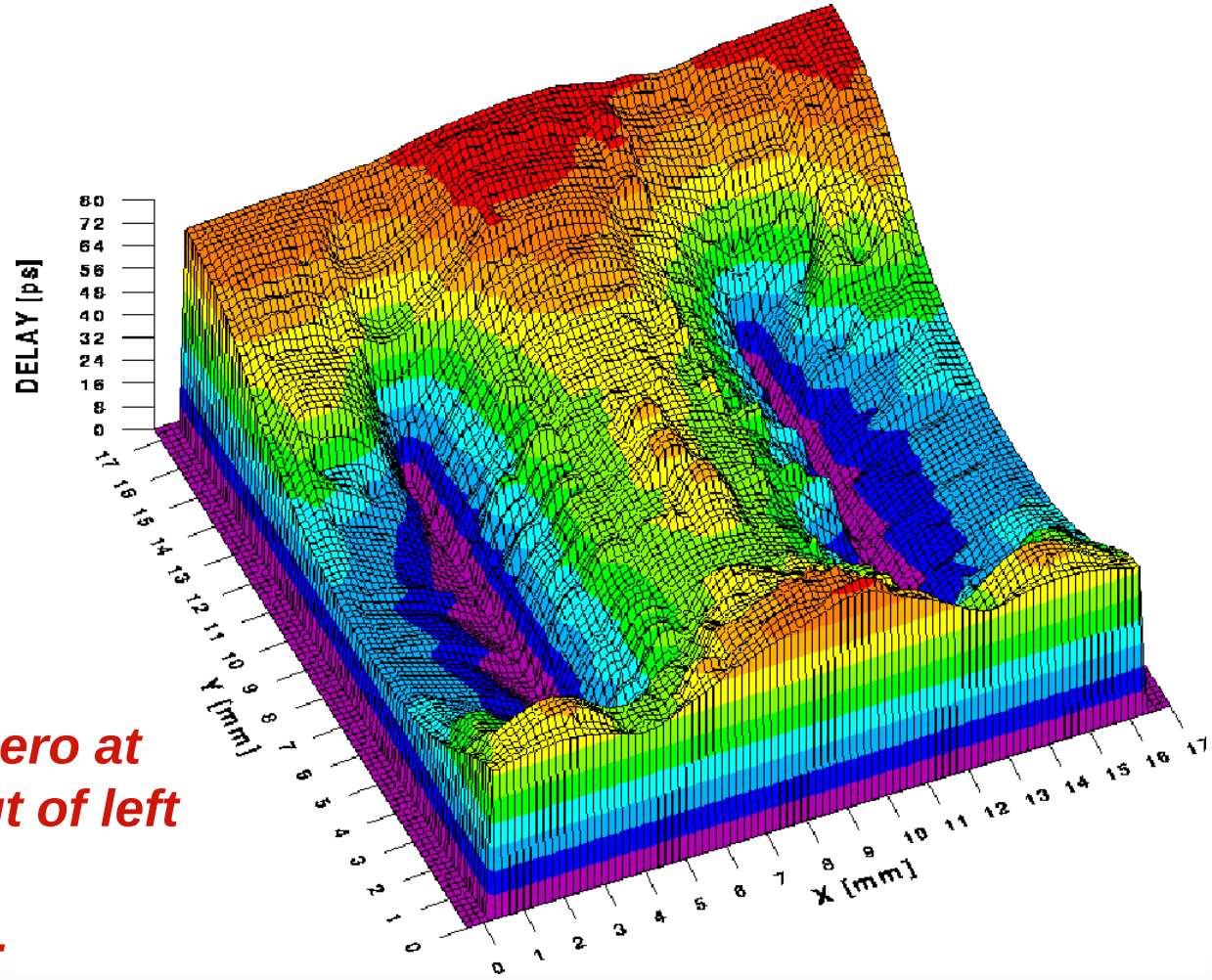


Location of clock driver on die

- 2 phase single wire clock, distributed globally
- 2 distributed driver channels
  - Reduced RC delay/skew
  - Improved thermal distribution
  - 3.75nF clock load
  - 58 cm final driver width
- Local inverters for latching
- Conditional clocks in caches to reduce power
- More complex race checking
- Device variation

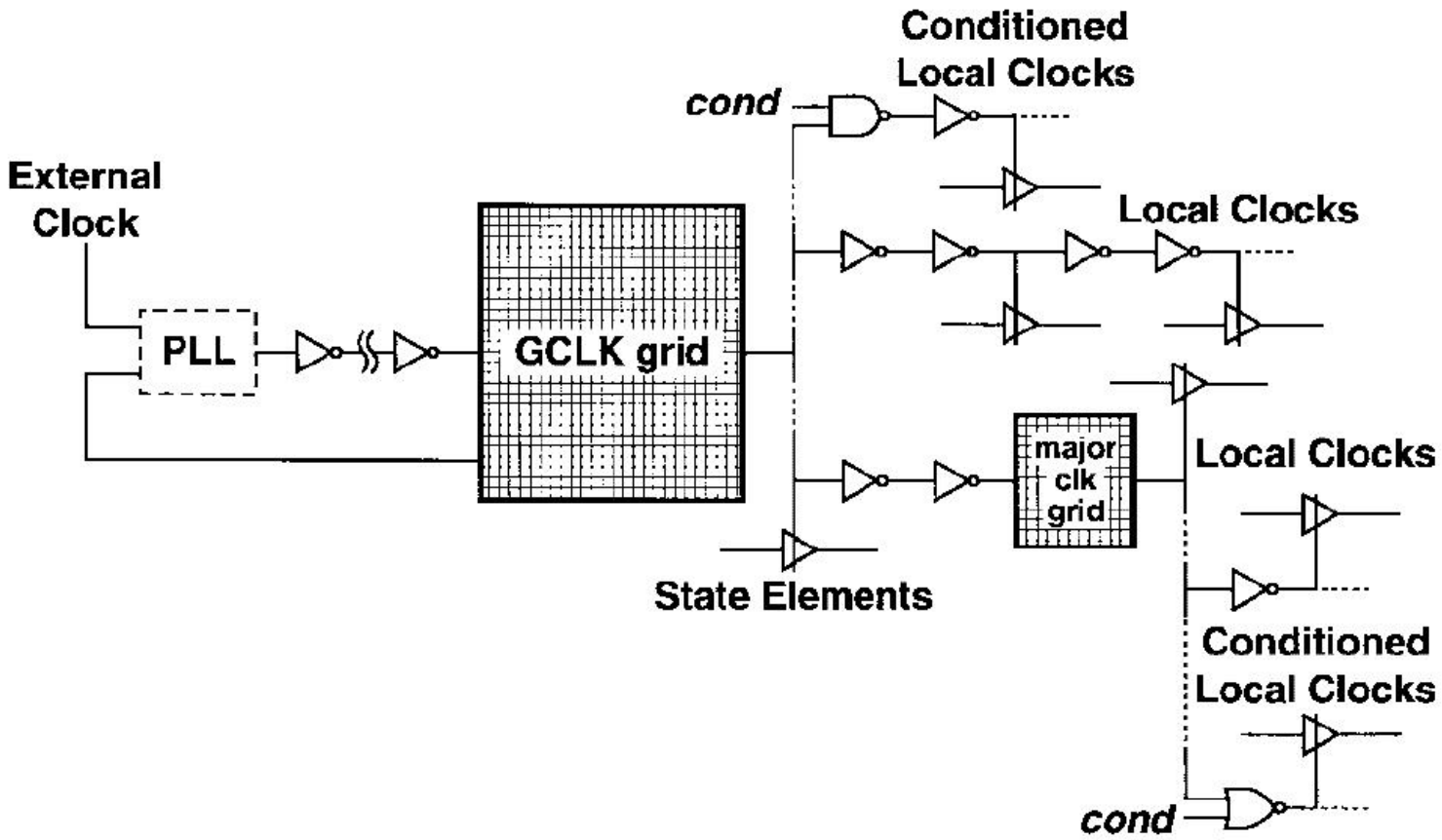

Clock Drivers

Clock Skew in Alpha Processor

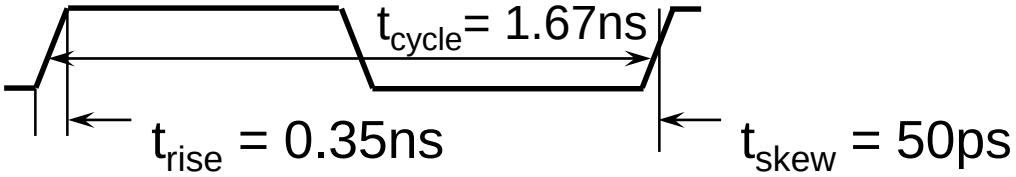


Skew is zero at the output of left and right drivers….

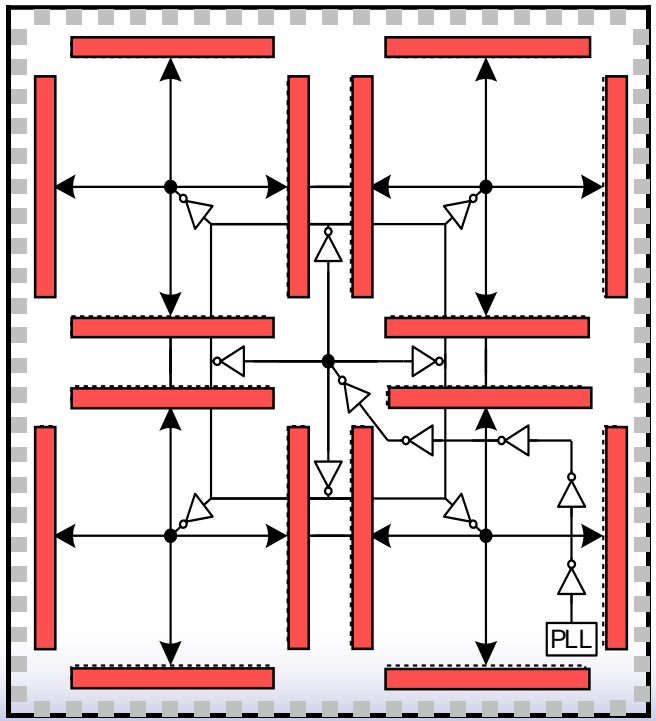
21264 Clocking



EV6 (Alpha 21264) Clocking
600 MHz – 0.35 micron CMOS

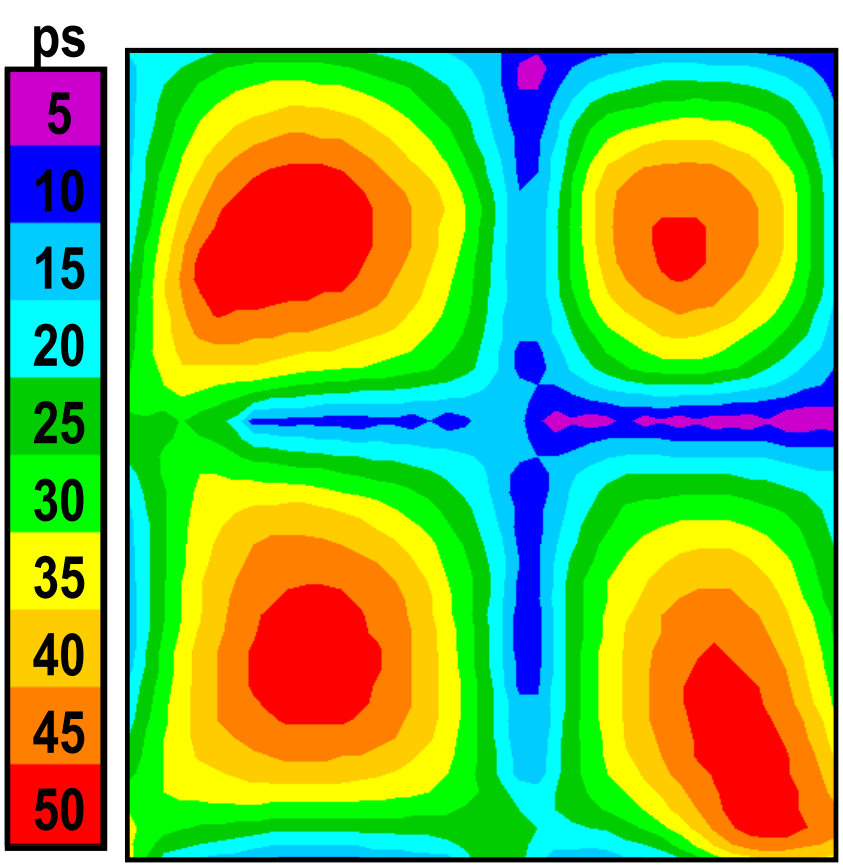


Global clock waveform

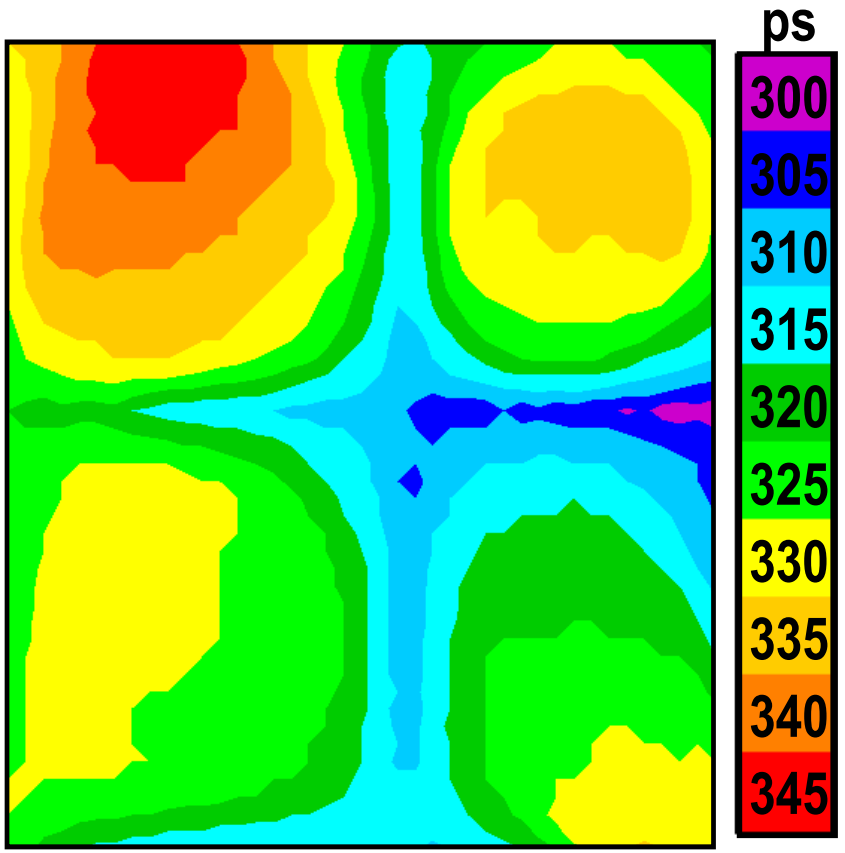


- 2 Phase, with multiple conditional buffered clocks
  - 2.8 nF clock load
  - 40 cm final driver width
- Local clocks can be gated "off" to save power
- Reduced load/skew
- Reduced thermal issues
- Multiple clocks complicate race checking

EV6 Clock Results



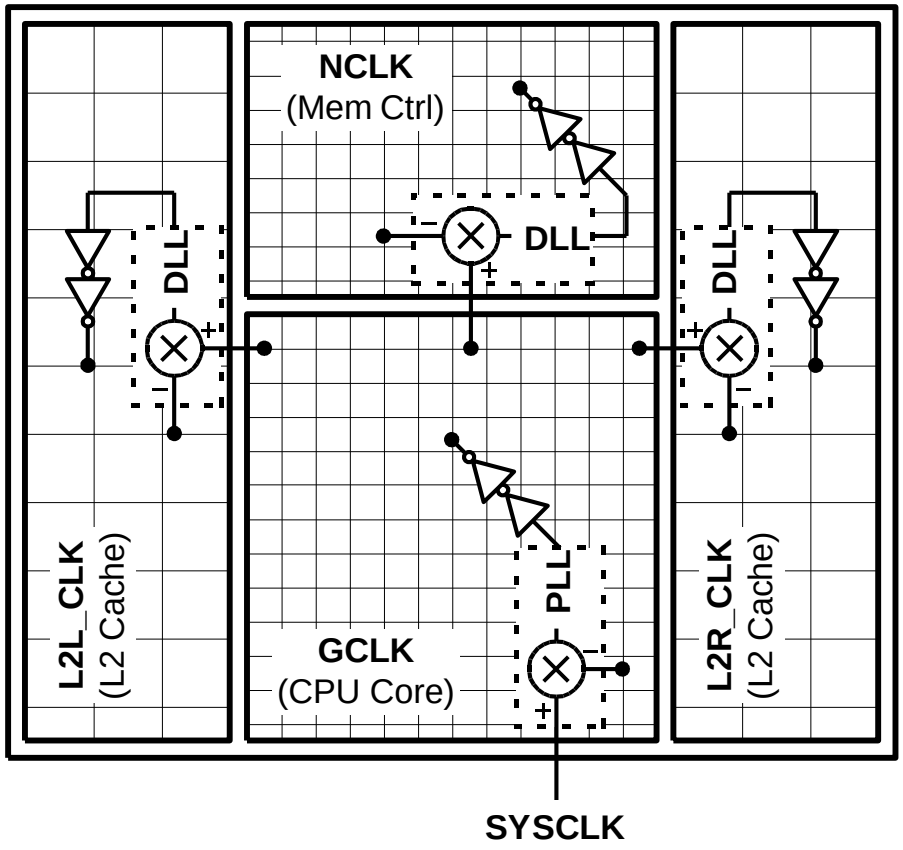
GCLK Skew
(at Vdd/2 Crossings)



GCLK Rise Times
(20% to 80% Extrapolated to 0% to 100%)

EV7 Clock Hierarchy

Active Skew Management and Multiple Clock Domains



+ widely dispersed drivers

+ DLLs compensate static and low-frequency variation

+ divides design and verification effort

– DLL design and verification is added work

+ tailored clocks