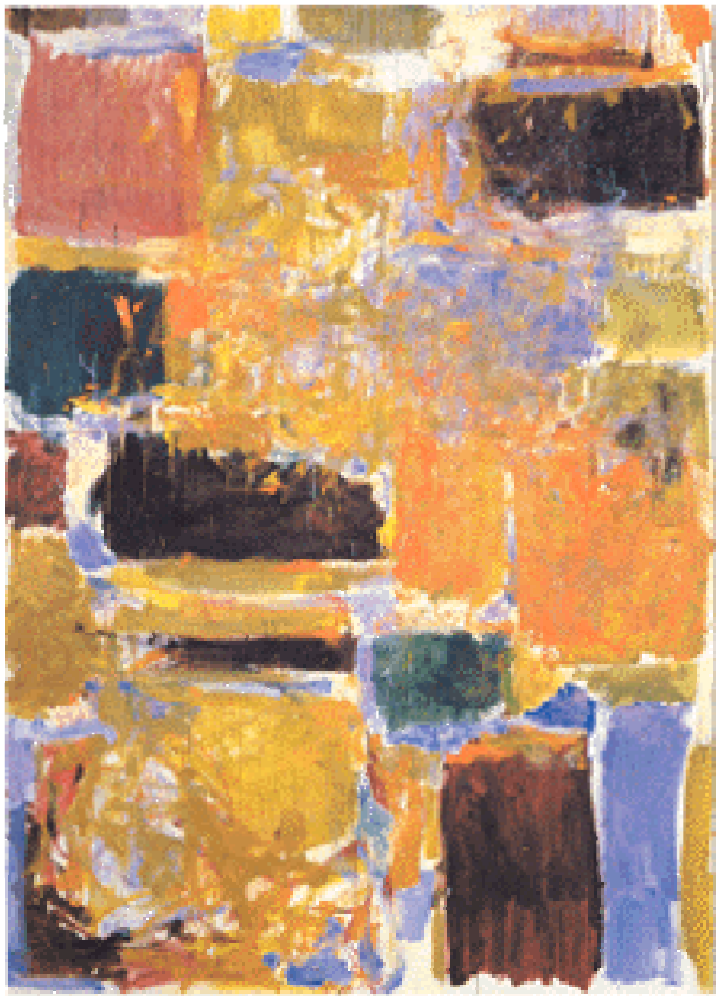


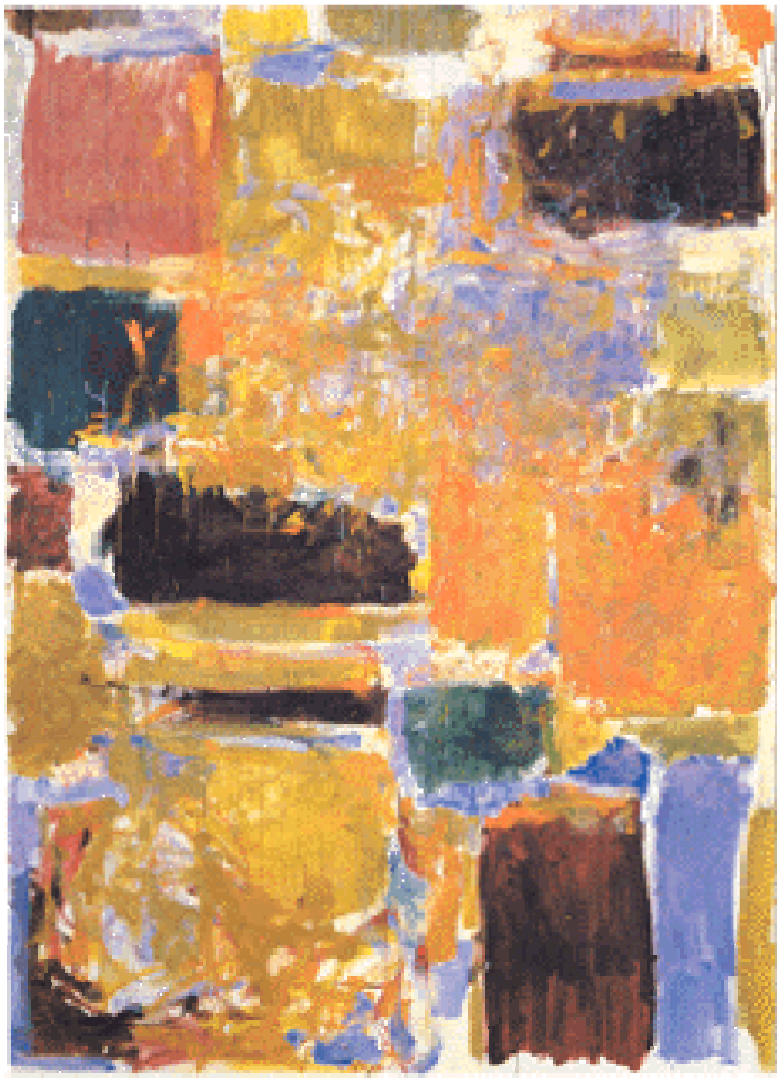
CS150 - EE141/241A
Fall 2014
Digital Design and Integrated Circuits

Instructors:
John Wawrzynek and Vladimir Stojanovic

Lecture 22

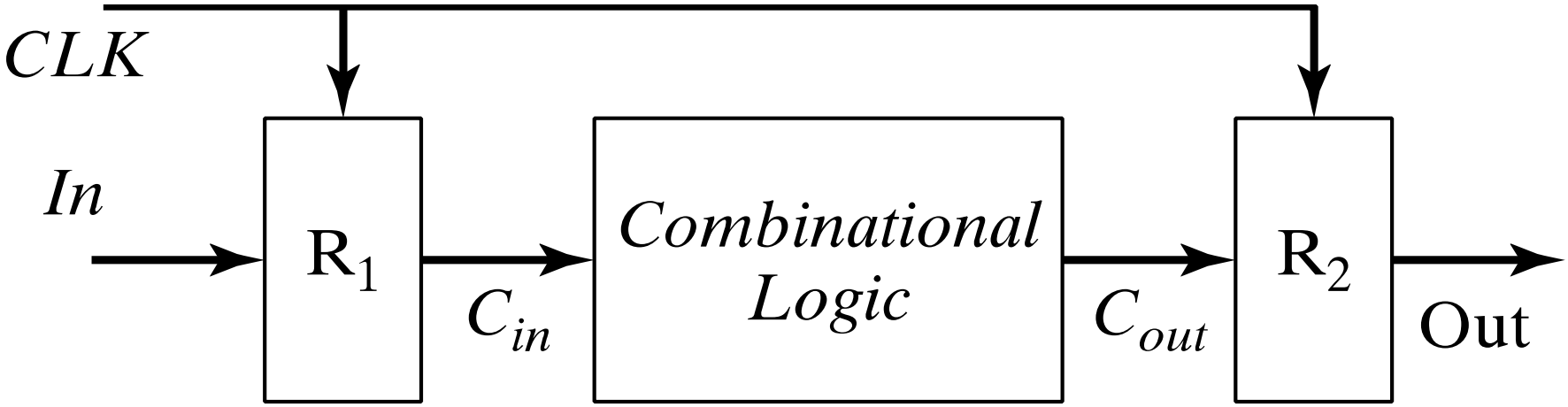
Outline

- Clock nonidealities
- Timing
- Clock Distribution

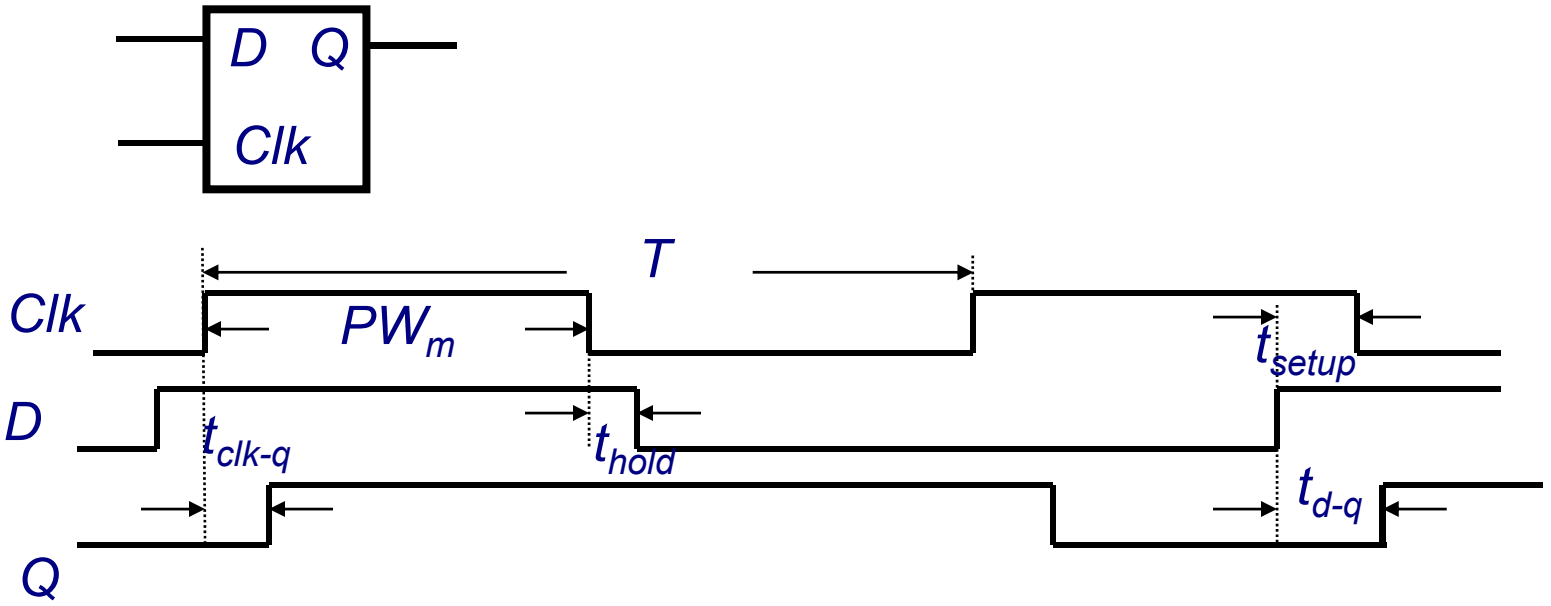


Synchronous Timing - Review

Synchronous Timing

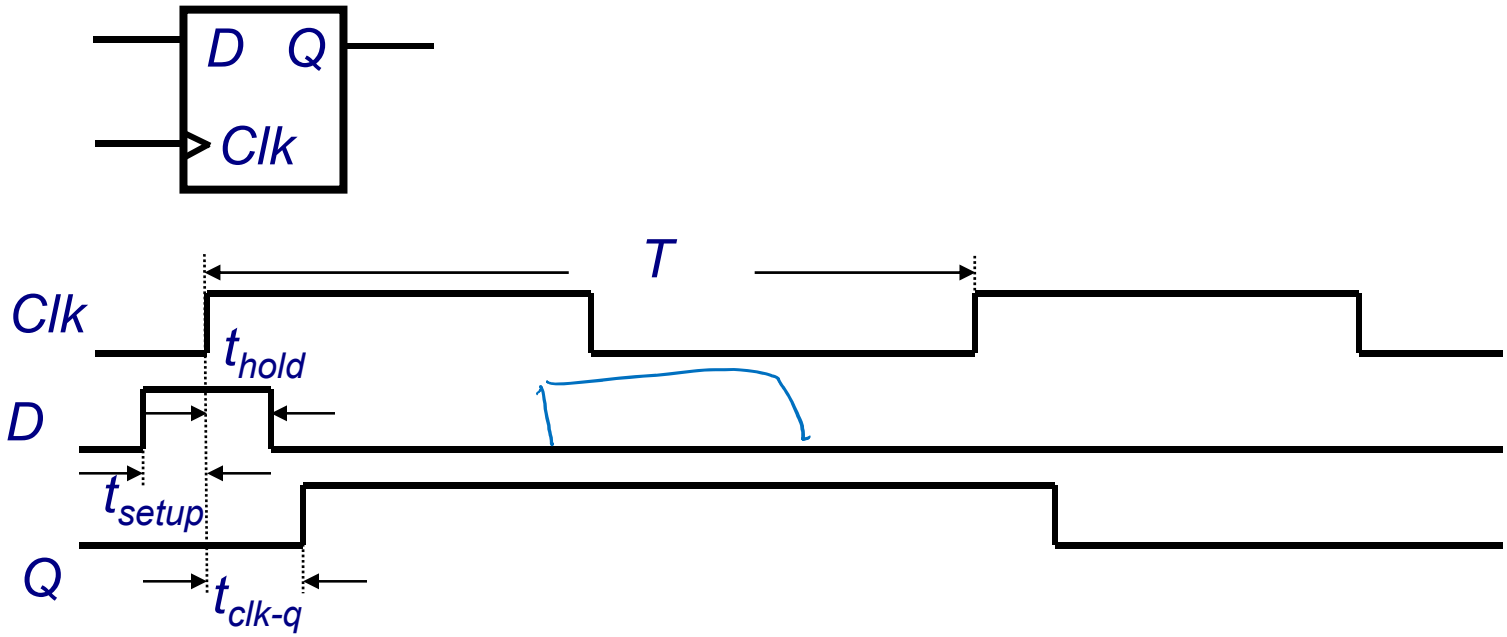


Latch Parameters



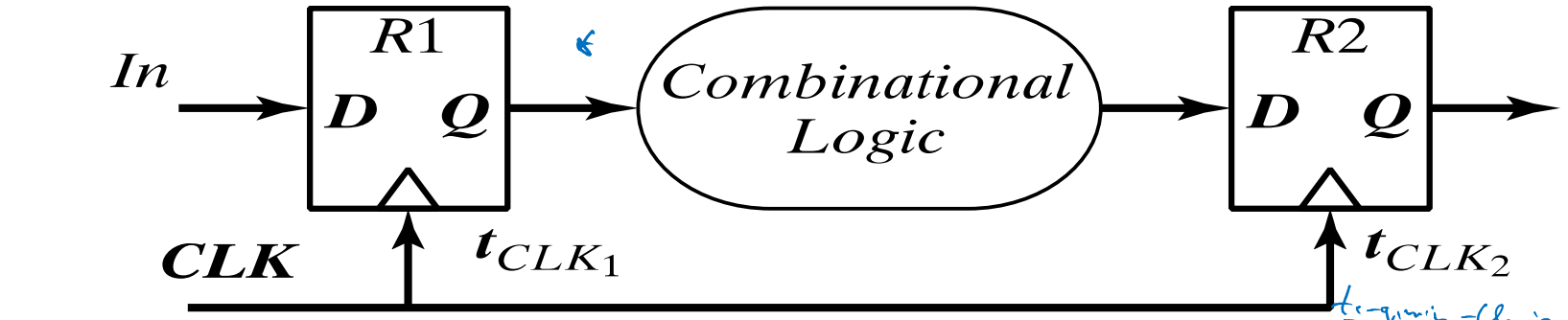
Delays can be different for rising and falling data transitions

Register Parameters



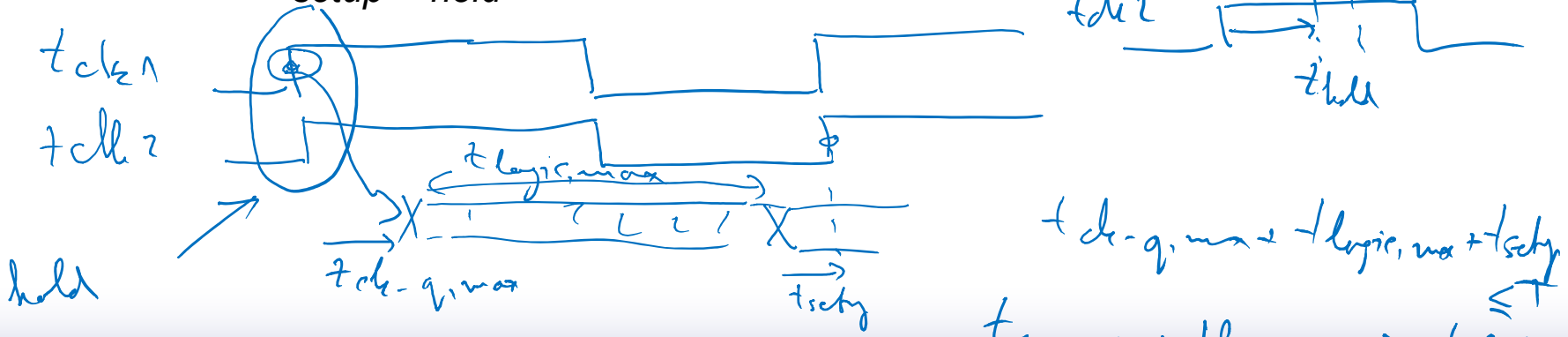
Delays can be different for rising and falling data transitions

Timing Constraints



$t_{clk\text{-}q,max}$
$t_{clk\text{-}q,min}$
t_{setup}, t_{hold}

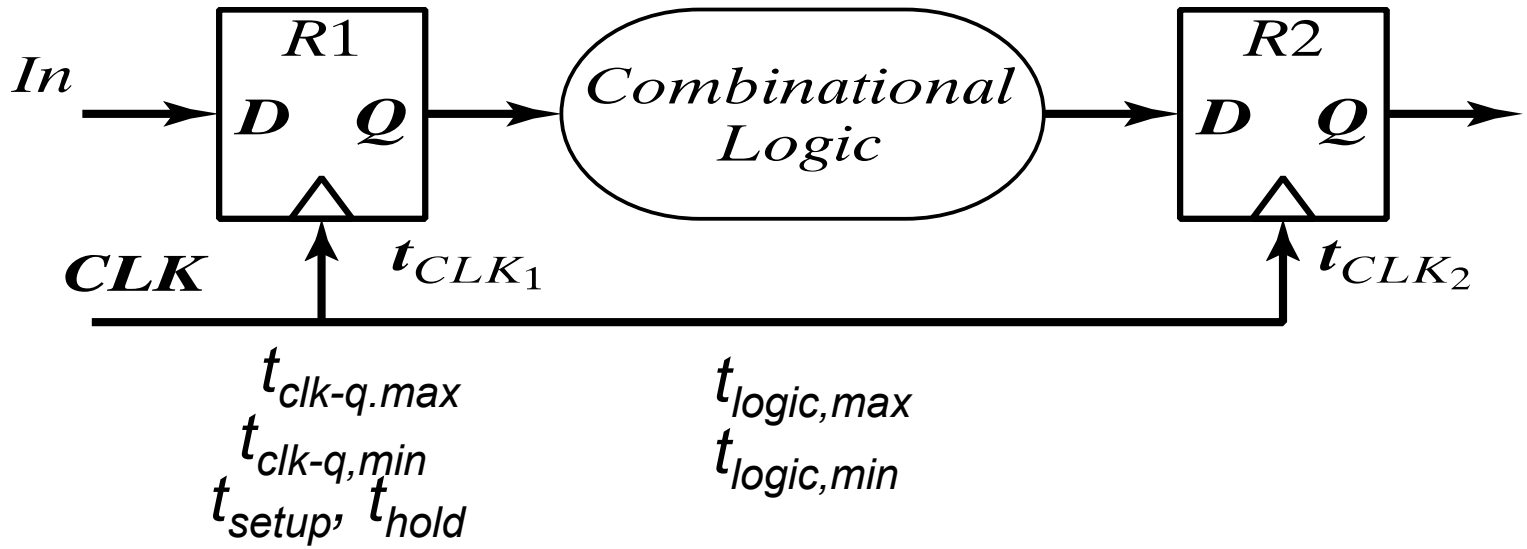
$t_{logic,max}$
$t_{logic,min}$



$t_{clk-q,max} + t_{logic,max} + t_{setup} \leq T$

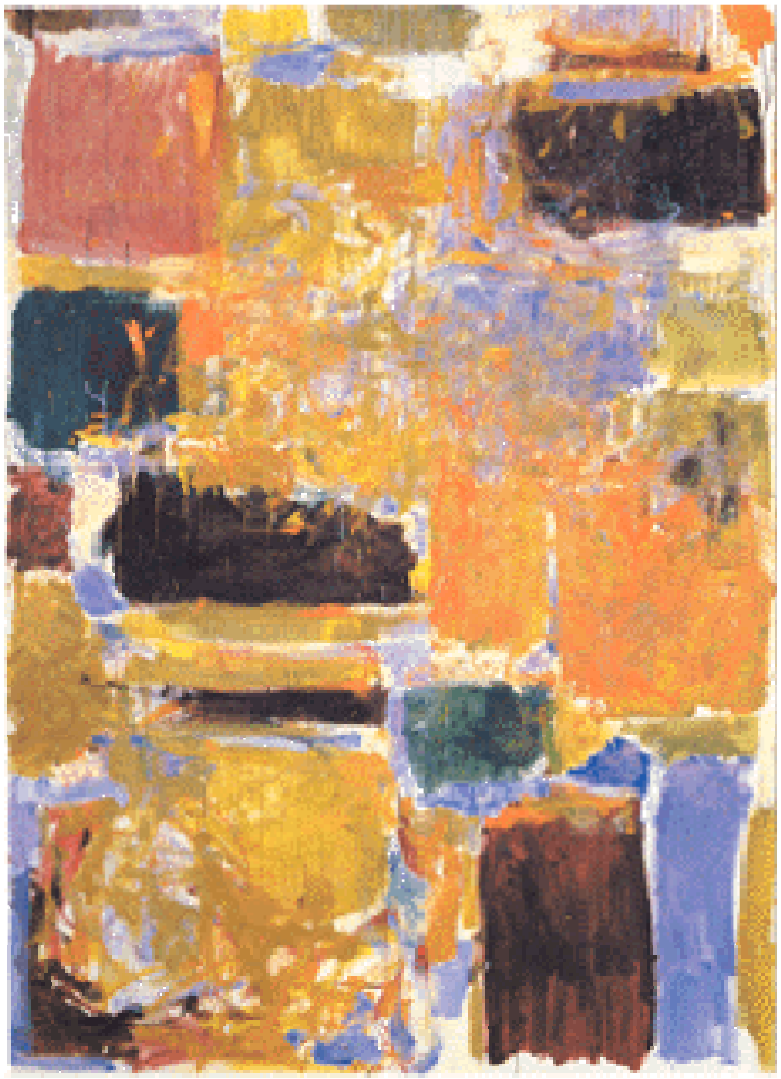
$t_{clk-q,min} + t_{logic,min} > t_{hold}$

Timing Constraints



Cycle time (max): $T_{Clk} > t_{clk\text{-}q,max} + t_{logic,max} + t_{setup}$

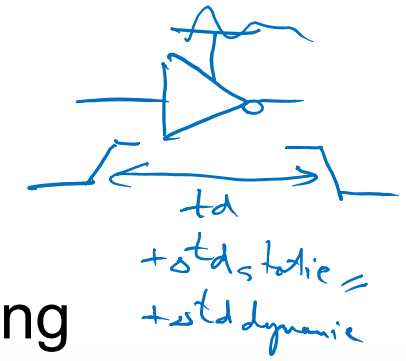
Race margin (min): $t_{hold} < t_{clk\text{-}q,min} + t_{logic,min}$



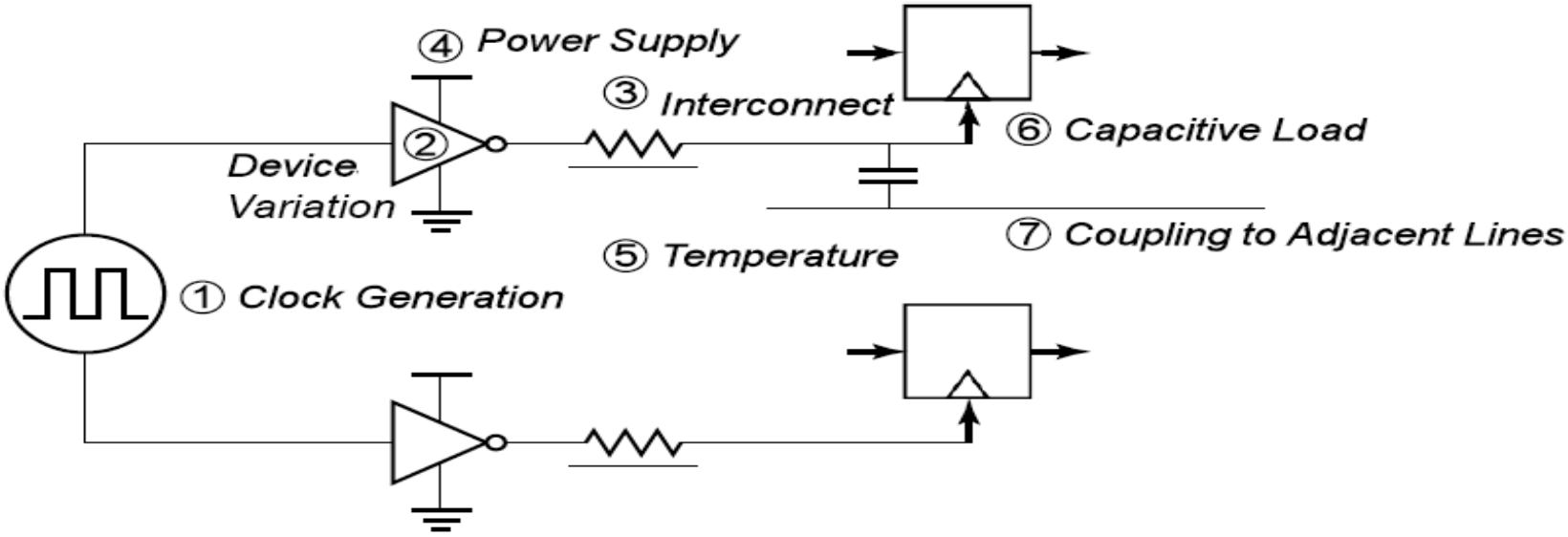
Clock Nonidealities

Clock Nonidealities

- **Clock skew**
  - Time difference between the sink (receiving) and source (launching) edge
  - 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

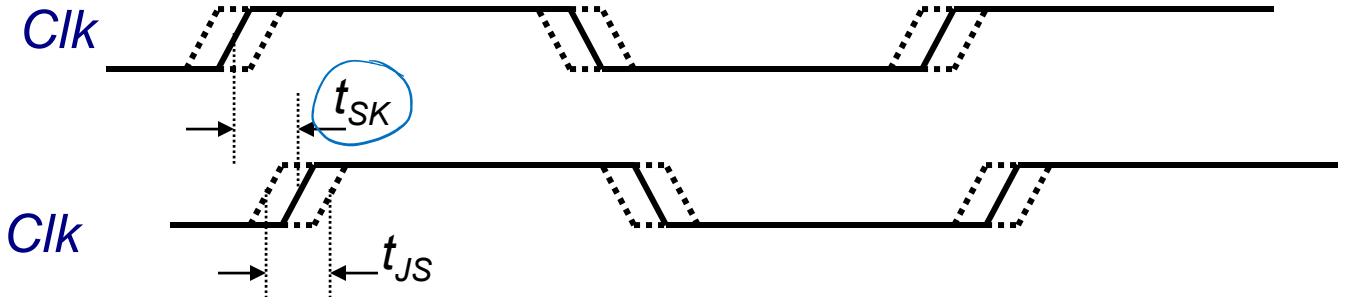


Clock Uncertainties

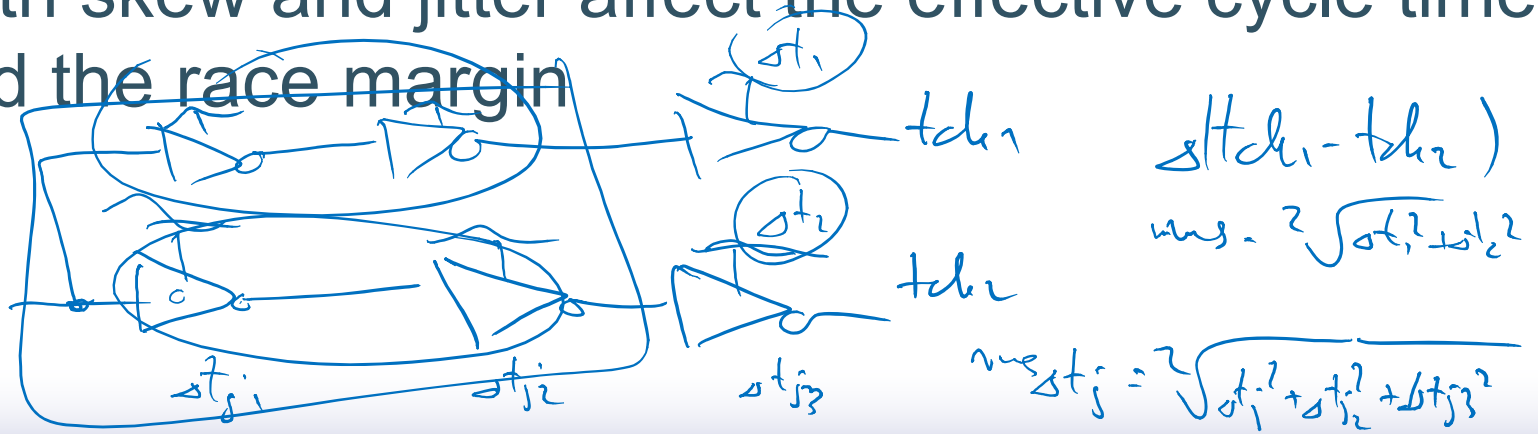


Sources of clock uncertainty

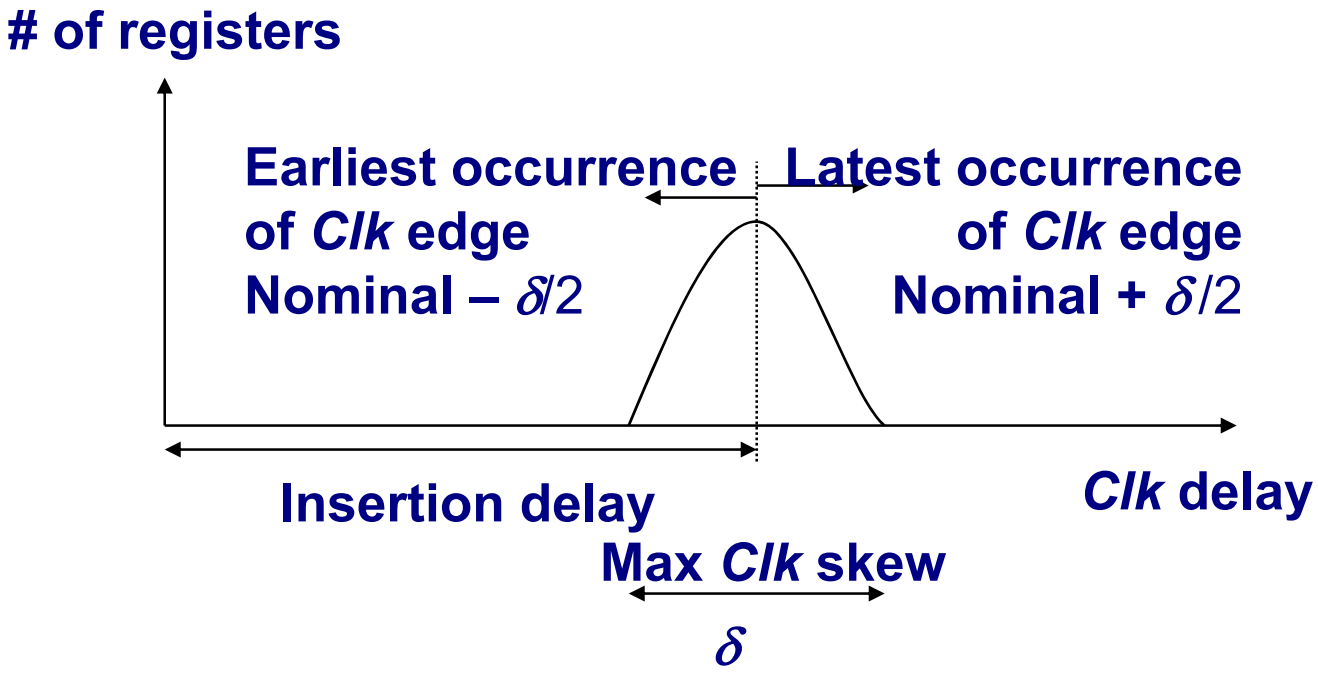
Clock Skew and Jitter



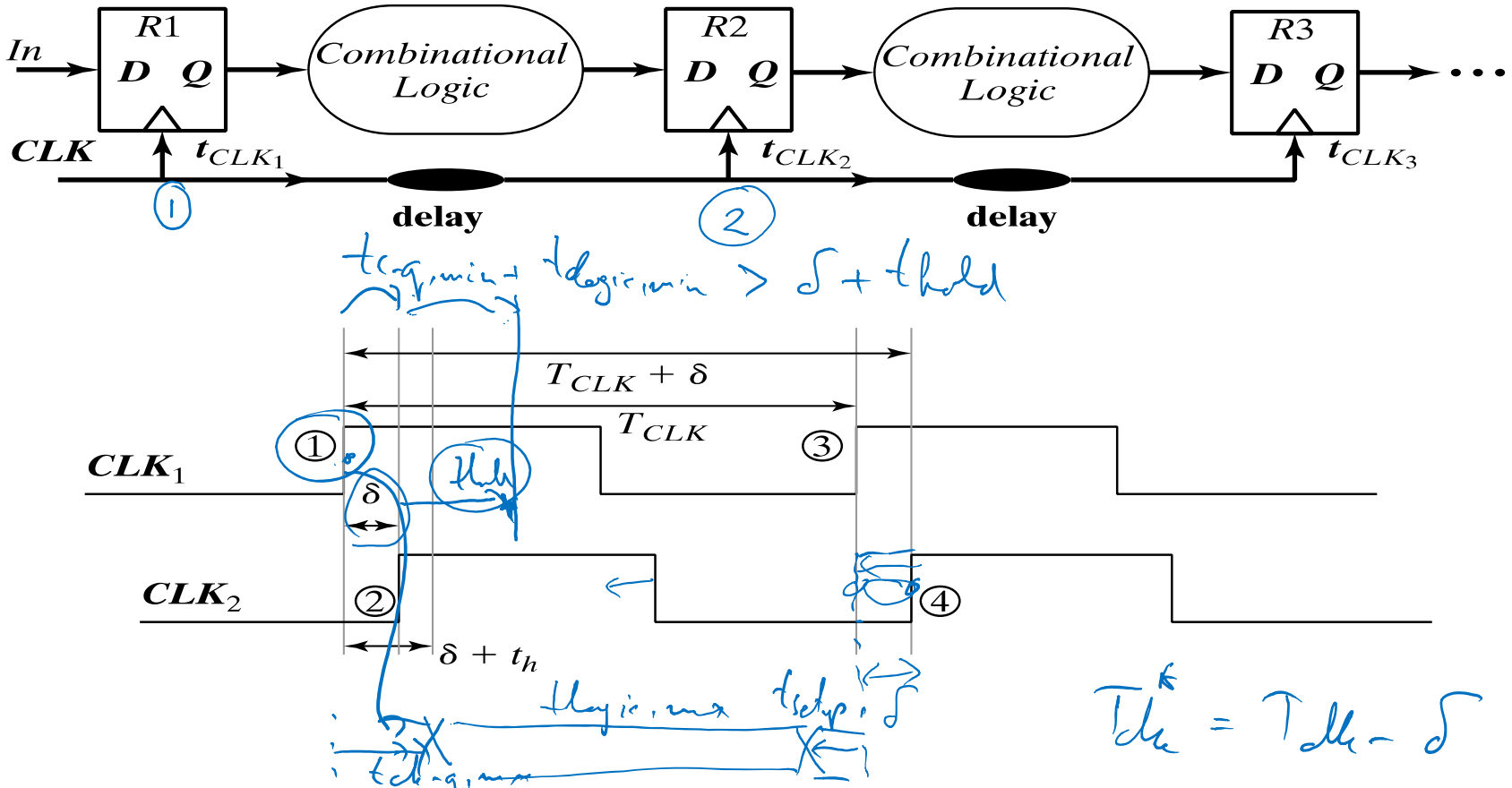
- Both skew and jitter affect the effective cycle time and the race margin



Clock Skew

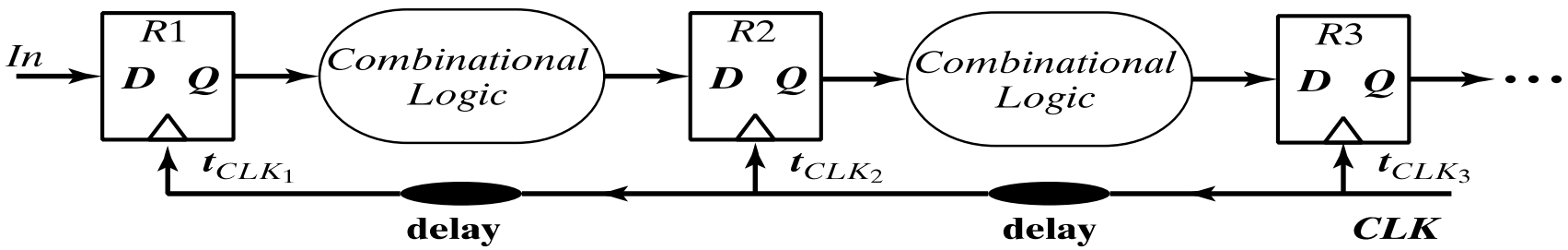

# of registers
Earliest occurrence of *Clk* edge Nominal – $\delta/2$
Latest occurrence of *Clk* edge Nominal + $\delta/2$
Insertion delay
Clk delay
Max *Clk* skew
δ

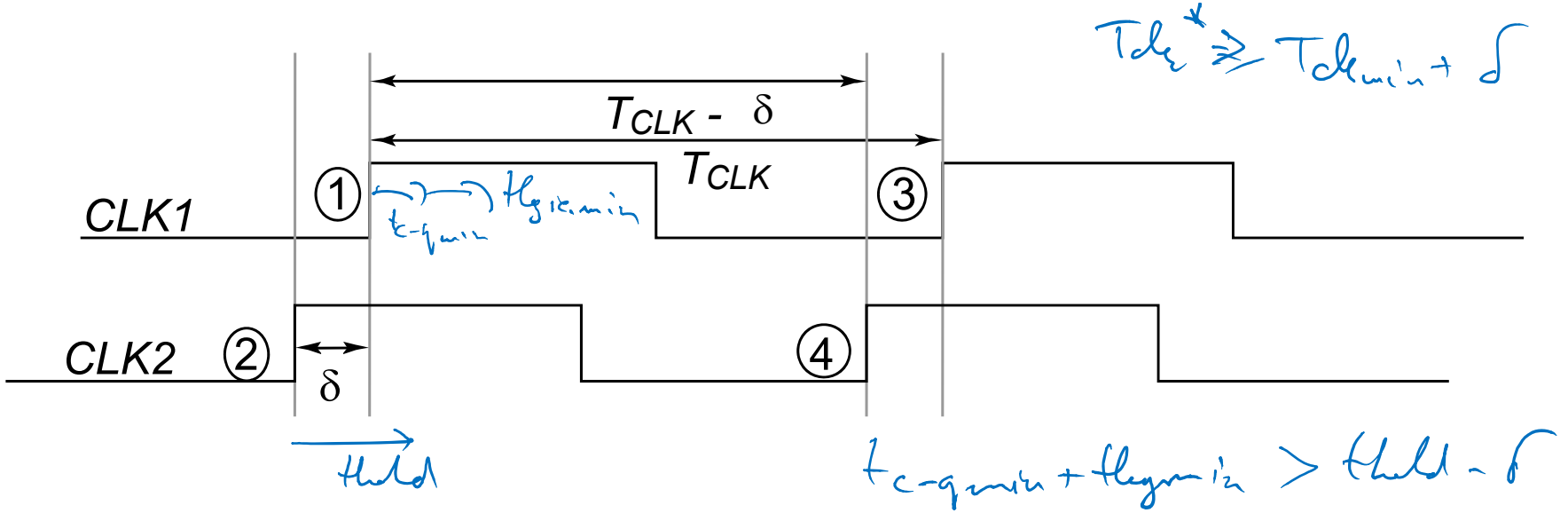
Positive Skew



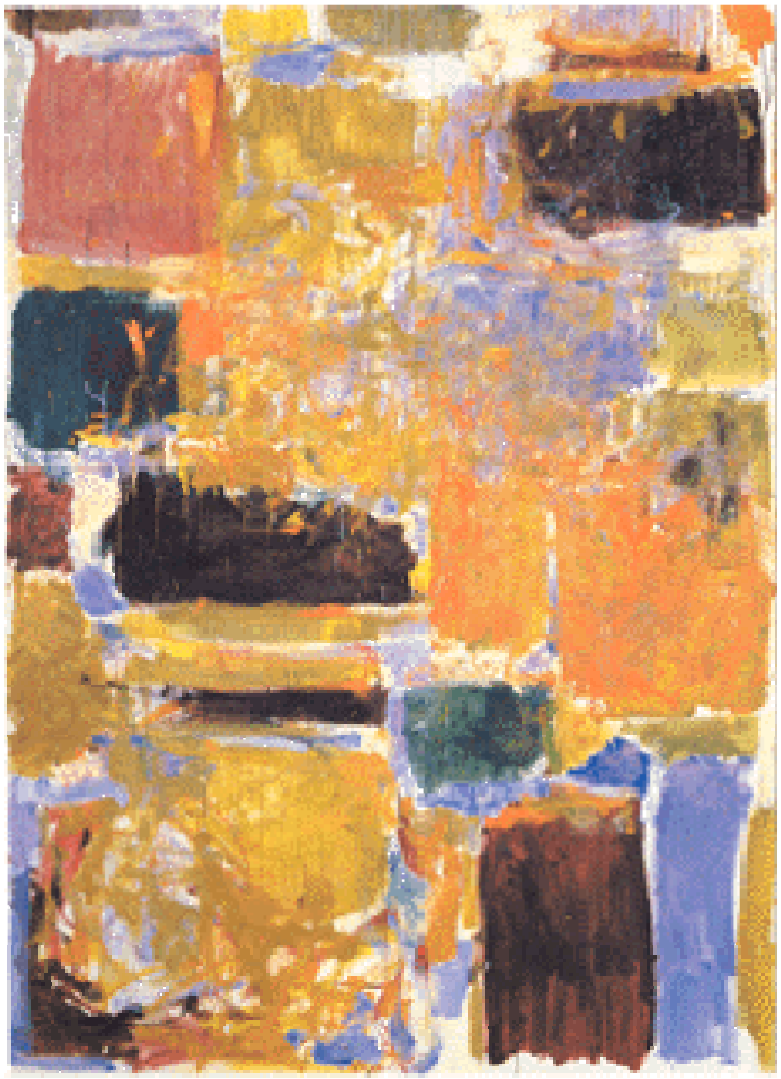
Launching edge arrives before the receiving edge

Negative Skew



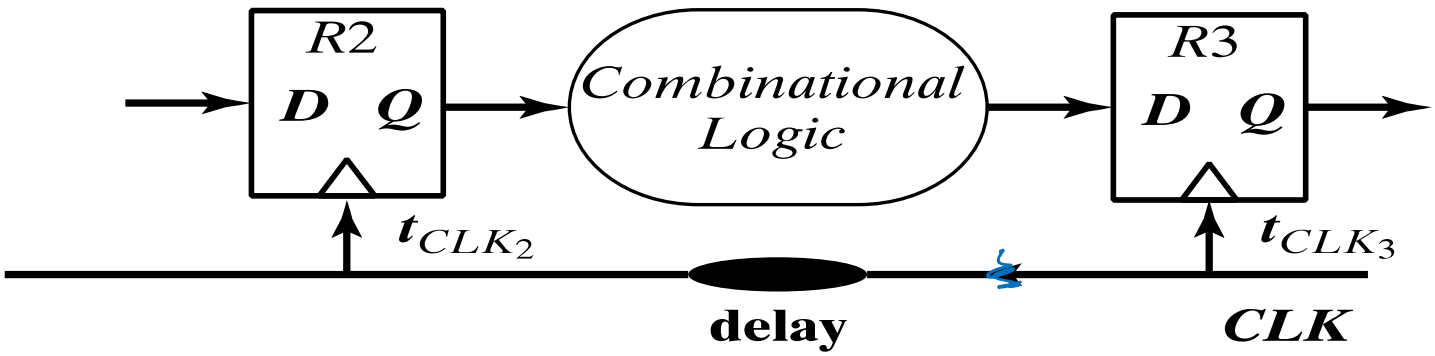


Receiving edge arrives before the launching edge



Timing with Clock Nonidealities

Timing Constraints



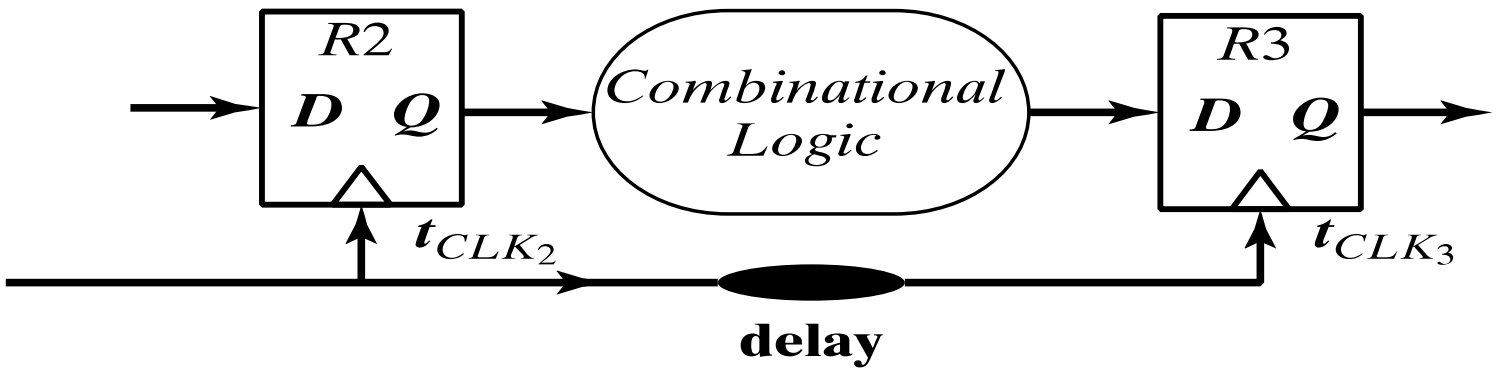
$t_{clk\text{-}q,max}$
$t_{clk\text{-}q,min}$
t_{setup}, t_{hold}

$t_{logic,max}$
$t_{logic,min}$

Minimum cycle time:

$$T_{clk} + \delta = t_{clk\text{-}q,max} + t_{setup} + t_{logic,max}$$

Timing Constraints



$t_{clk\text{-}q,max}$ $t_{logic,max}$

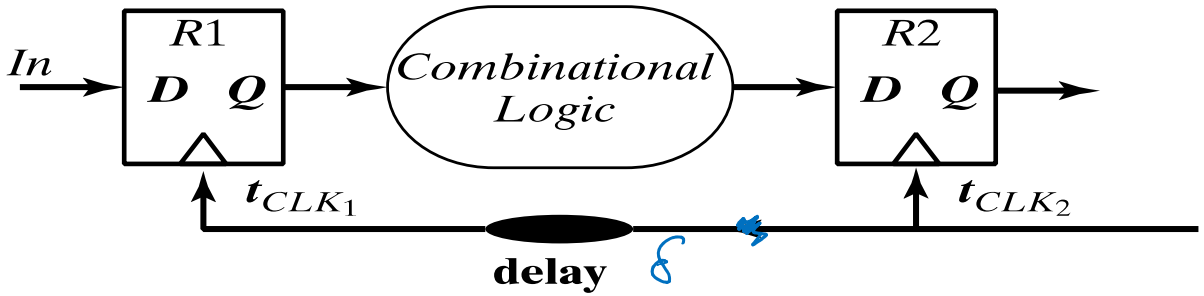
$t_{clk\text{-}q,min}$ $t_{logic,min}$

t_{setup}, t_{hold}

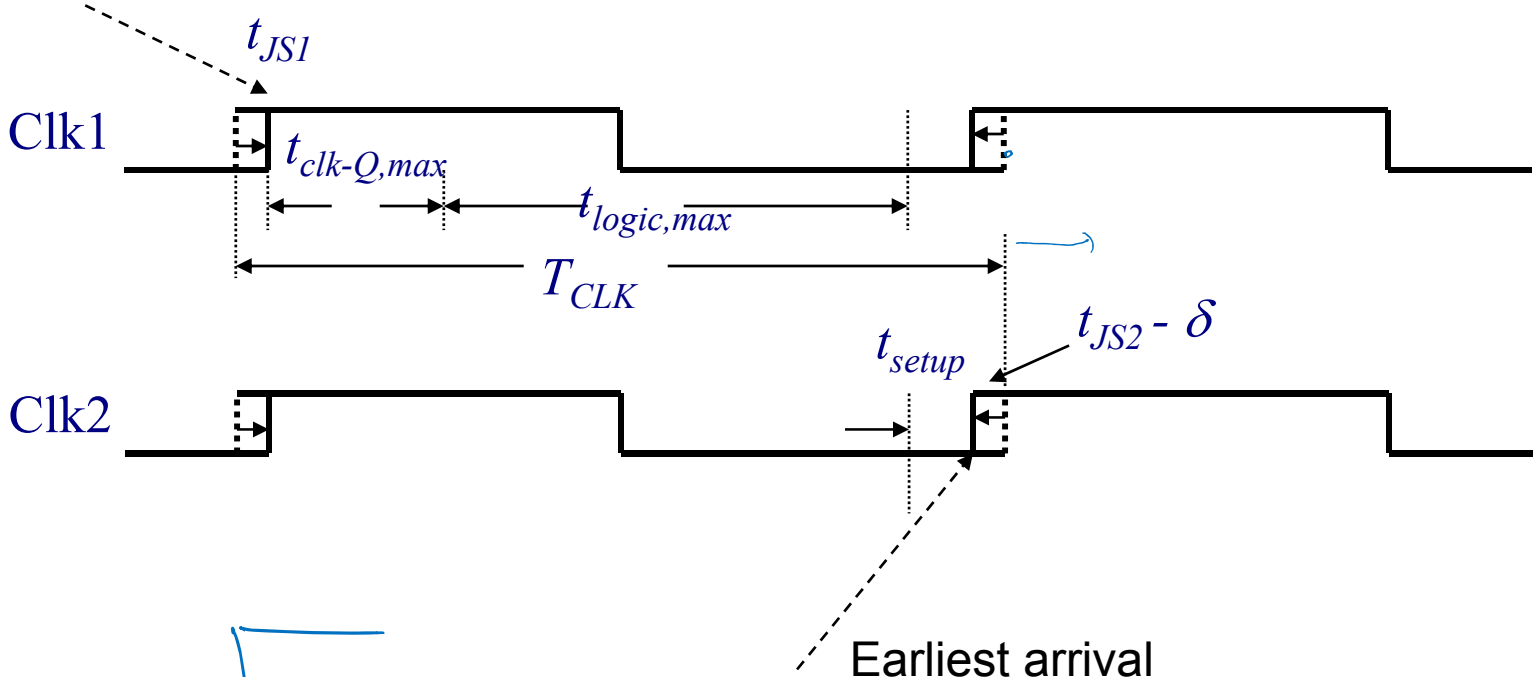
Hold time constraint:

$t_{(clk\text{-}q,min)} + t_{(logic,min)} > t_{hold} + \delta$

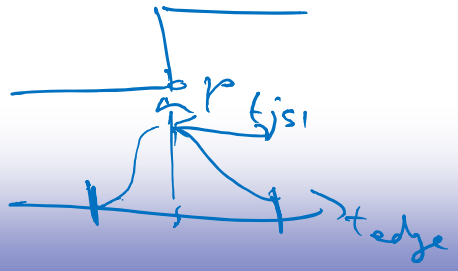
Longest Logic Path in Edge-Triggered Systems



Latest point of launching



Earliest arrival of next cycle



Clock Constraints in Edge-Triggered Systems

If launching edge is late and receiving edge is early, the data will not be too late if:

$$t_{clk\text{-}q,max} + t_{logic,max} + t_{setup} < T_{CLK} - t_{JS,1} - t_{JS,2} + \delta$$

Minimum cycle time is determined by the maximum delays through the logic

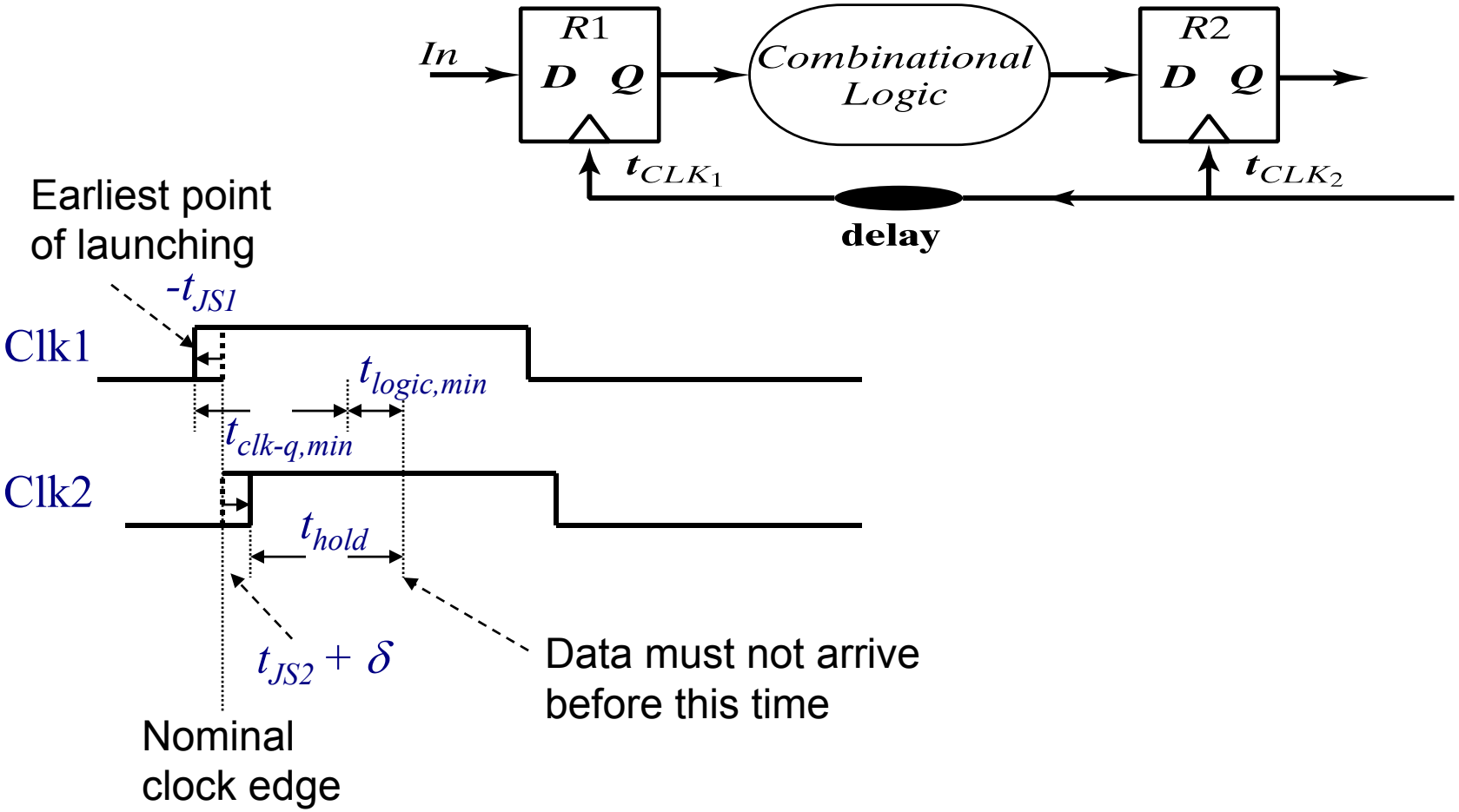
$$t_{clk\text{-}q,max} + t_{logic,max} + t_{setup} - \delta + 2t_{JS} < T_{CLK}$$

peak-to-peak jitter

Skew can be either positive or negative

Jitter t_{JS} usually expressed as peak-to-peak or N x rms value

Shortest Path



Clock Constraints in Edge-Triggered Systems

If launching edge is early and receiving edge is late:

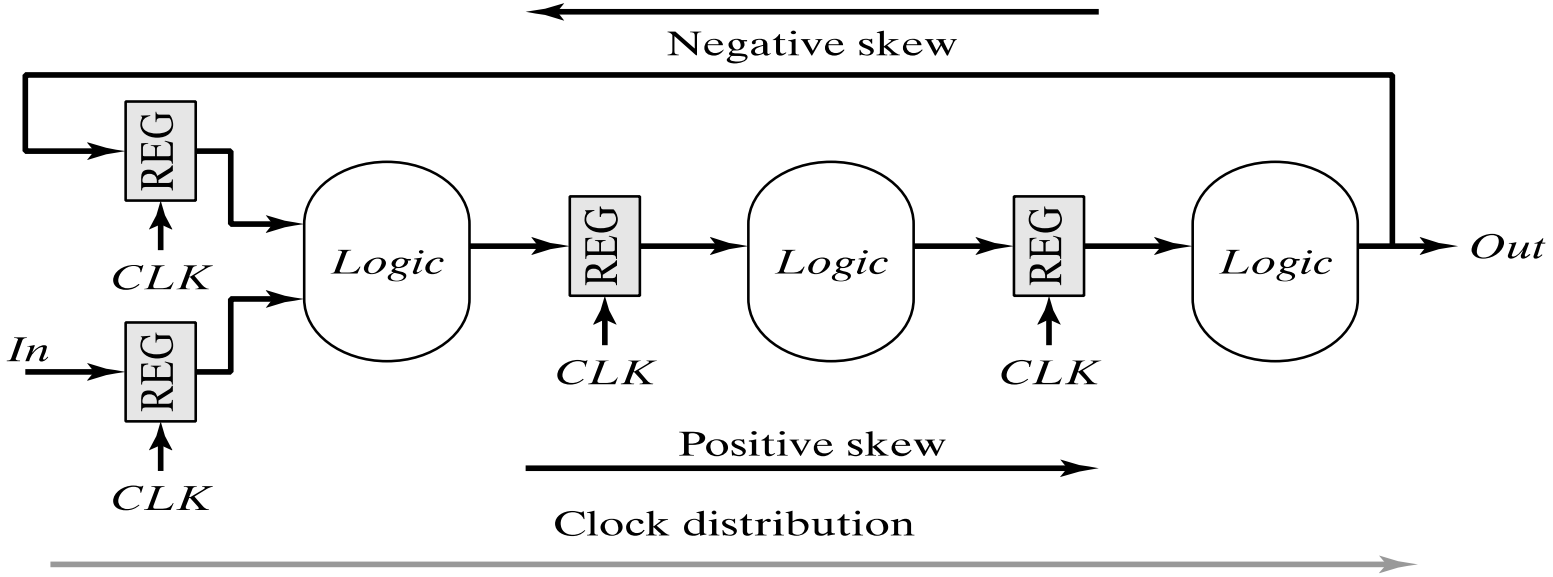
$$t_{clk\text{-}q,min} + t_{logic,min} - t_{JS,1} > t_{hold} + t_{JS,2} + \delta$$

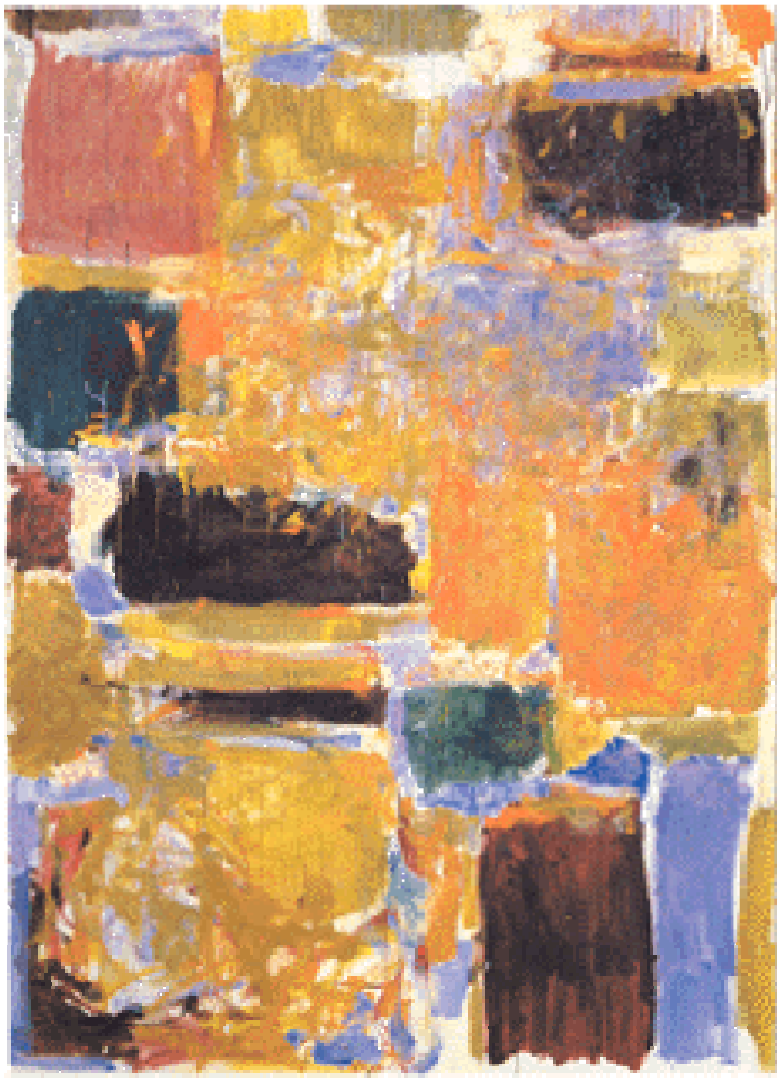
Minimum logic delay

$$t_{clk\text{-}q,min} + t_{logic,min} > t_{hold} + 2t_{JS} + \delta$$

(This assumes jitter at launching and receiving clocks are independent – which usually is not true)

Datapath with Feedback





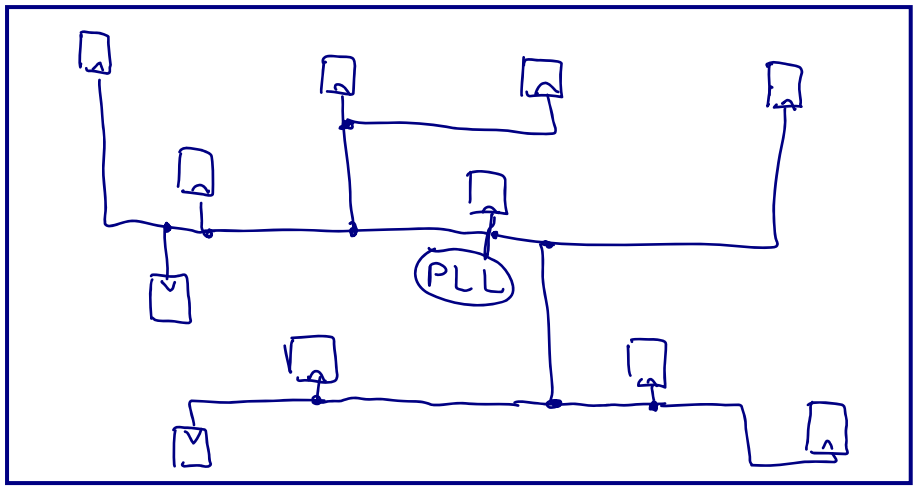
Clock Distribution

Clock Distribution

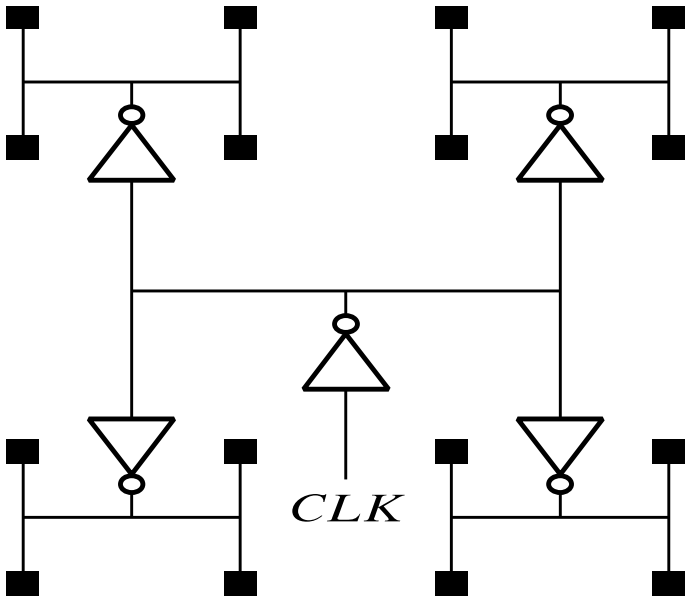
- Single clock generally used to synchronize all logic on the same chip
  - Need to distribute clock over the entire die
  - While maintaining low skew/jitter
  - (And without burning too much power)

Clock Distribution

- What's wrong with just routing wires to every point that needs a clock?

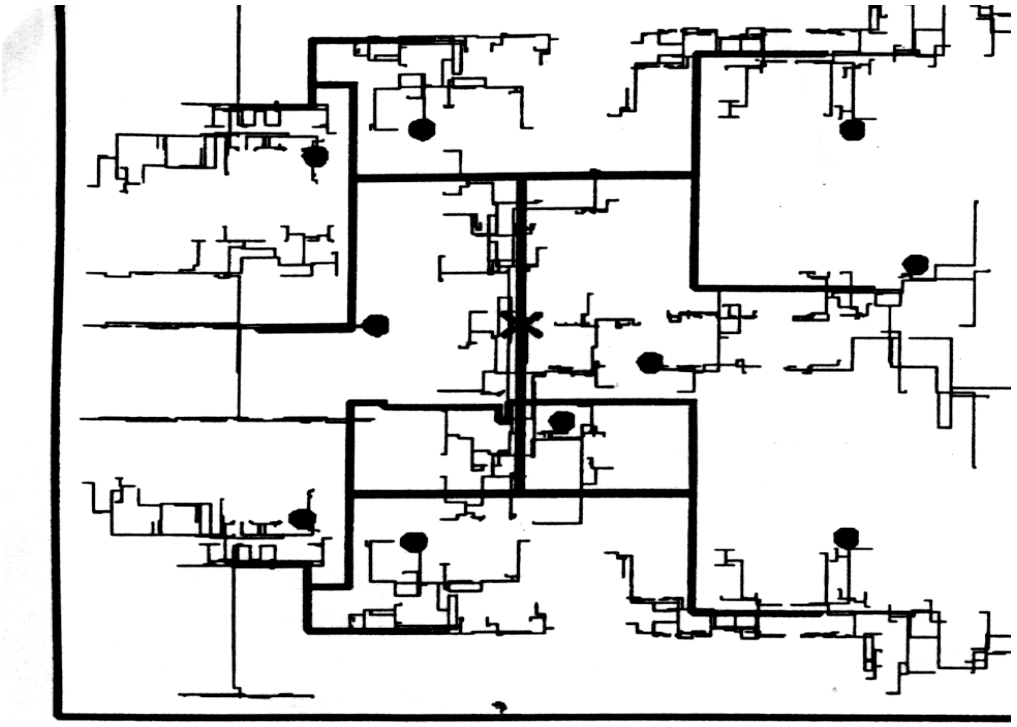


H-Tree



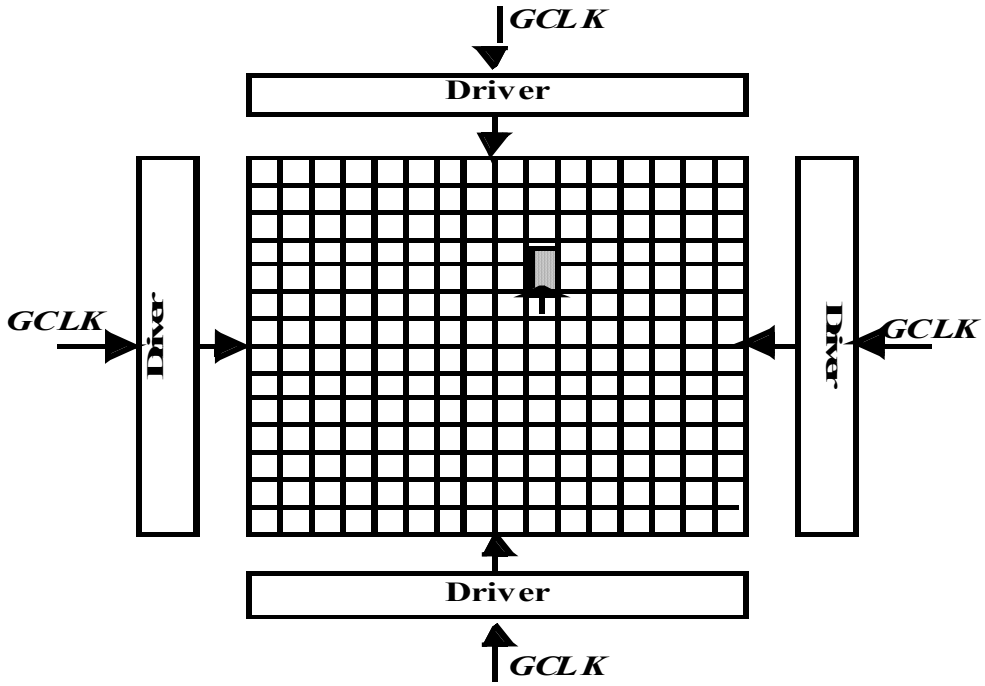
Equal wire length/number of buffers to get to every location

More realistic H-tree



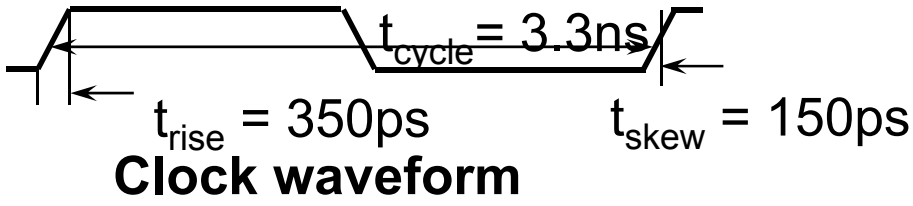
[Restle98]

Clock Grid

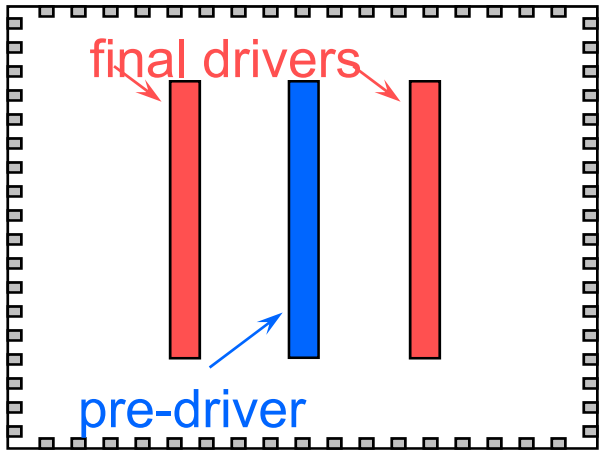


- No RC matching
- But huge power

Example: DEC Alpha 21164 (1995)

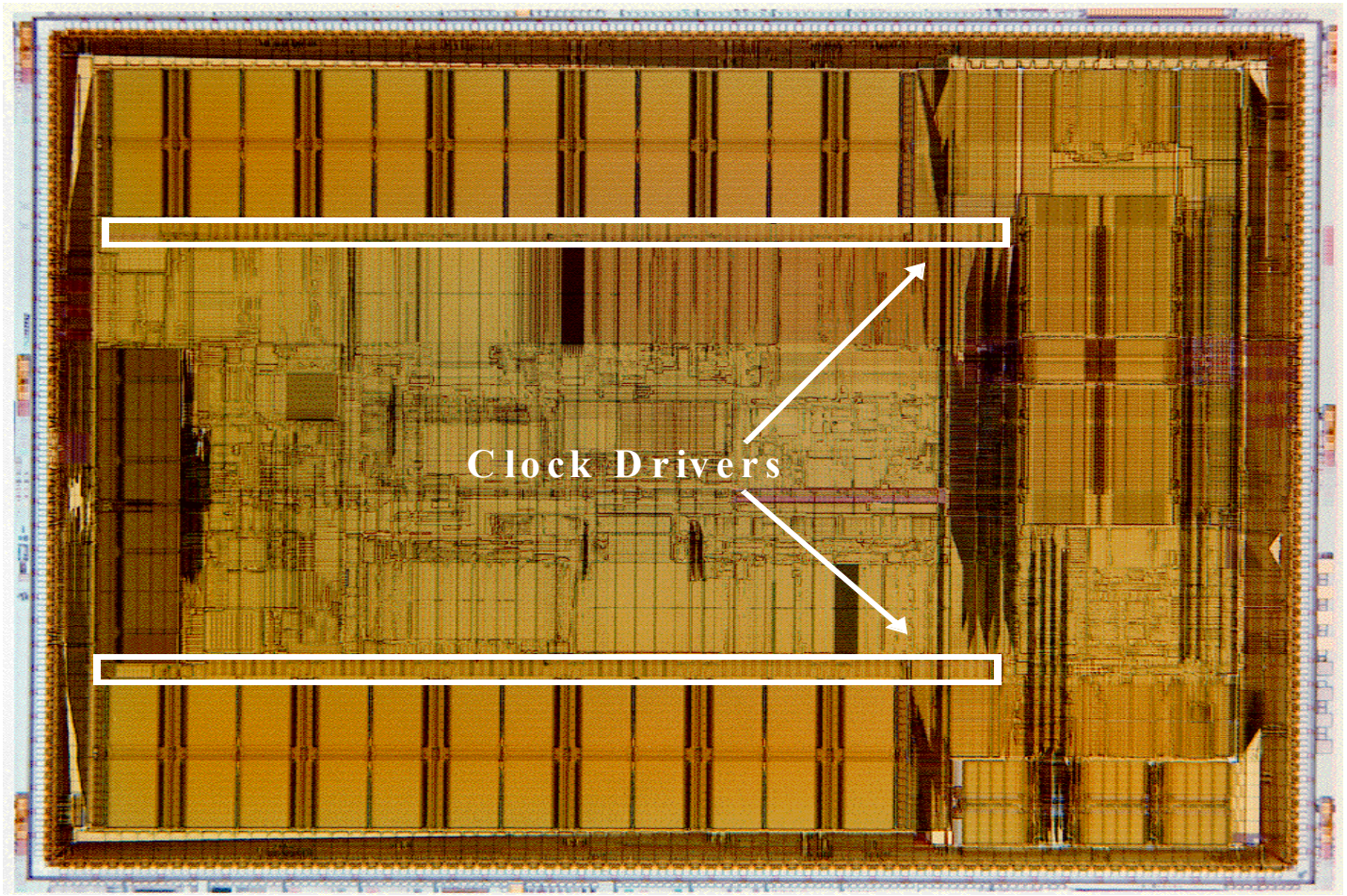


Clock waveform

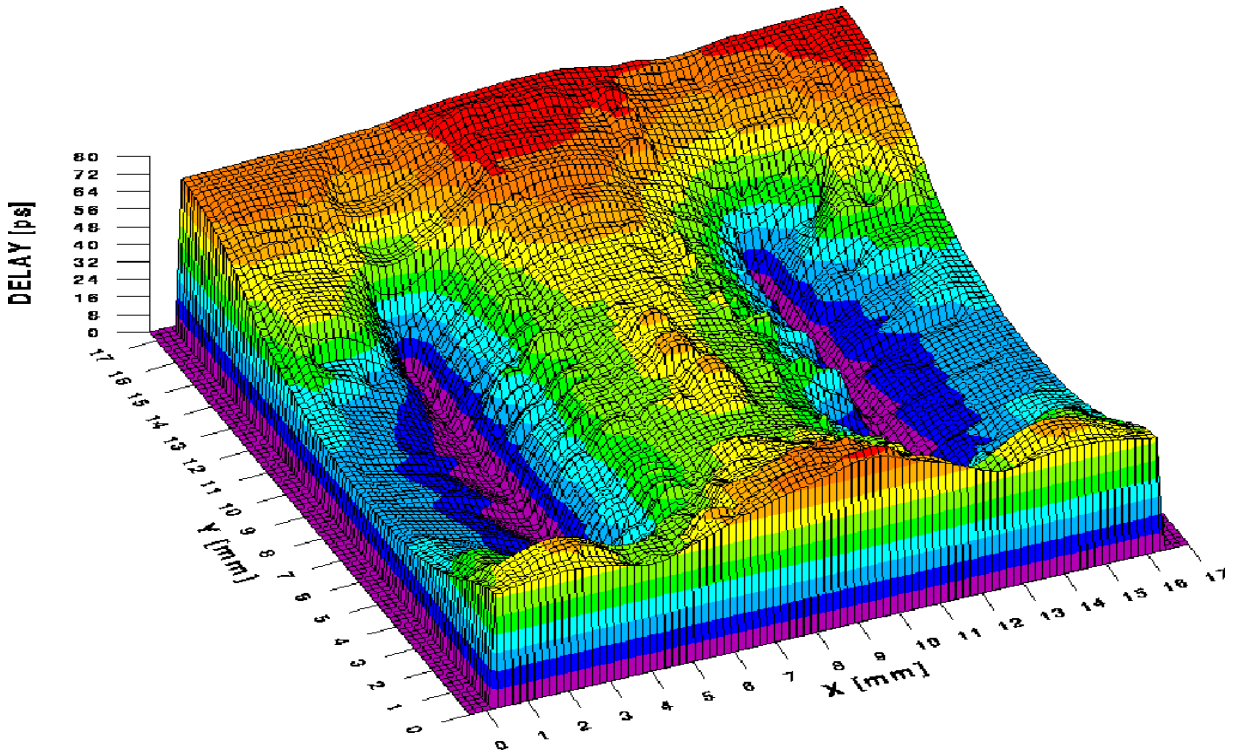


Location of clock driver on die

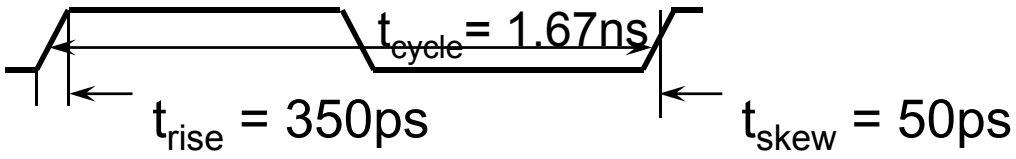
- 2 phase single wire clock, distributed globally
- 2 distributed driver channels
  - Reduced RC delay/skew
  - Improved thermal distribution
  - 3.75nF clock load, 20W power
  - 58 cm final driver width
- Local inverters for latching
- Conditional clocks in caches to reduce power
- More complex race checking
- Device variation


Clock Drivers

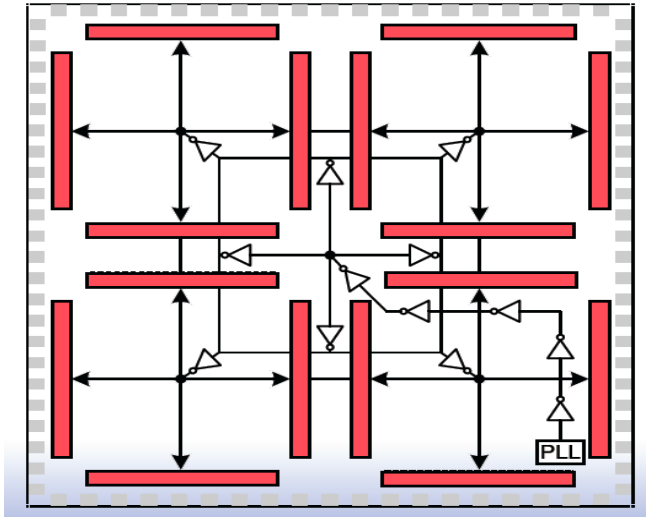
Clock Skew in Alpha Processor



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

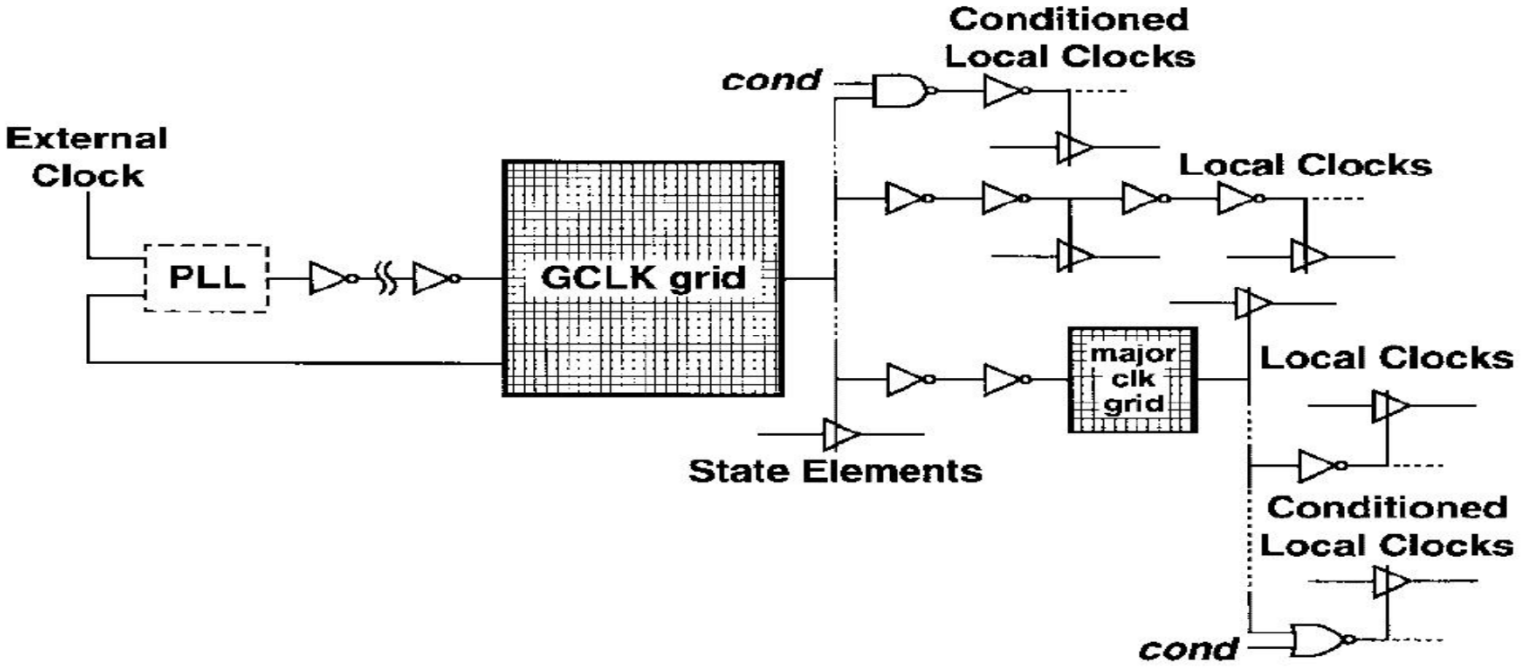


Global clock waveform

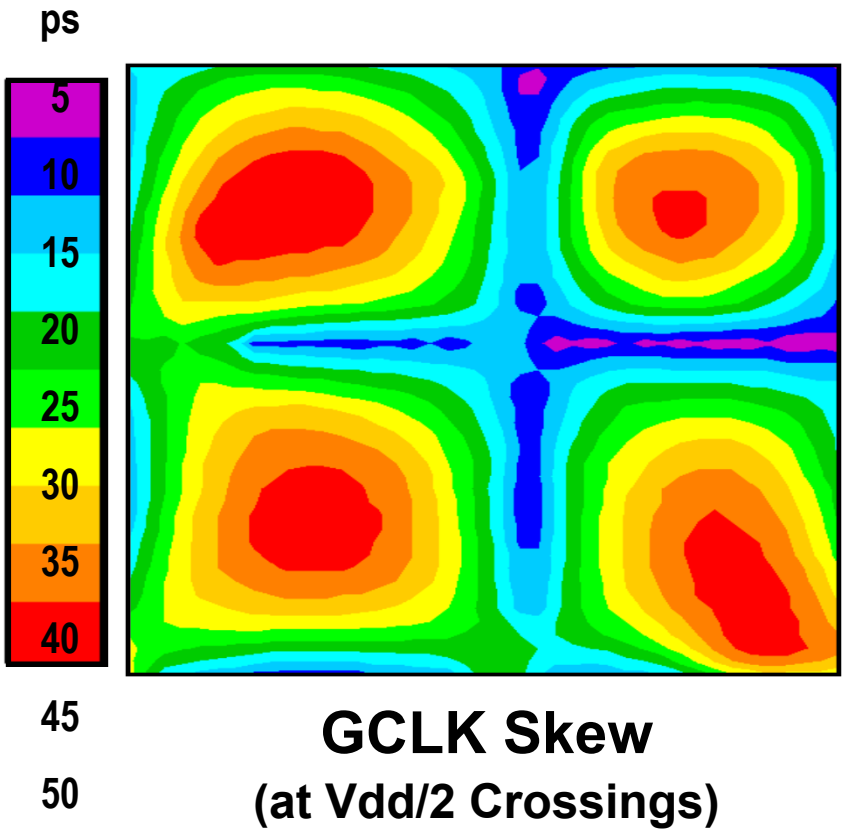


- 2 Phase, with multiple conditional buffered clocks
  - 2.8 nF clock load
  - 40 cm final driver width
- Local clocks can be gated "off" to save power
- Reduced load/skew
- Reduced thermal issues
- Multiple clocks complicate race checking

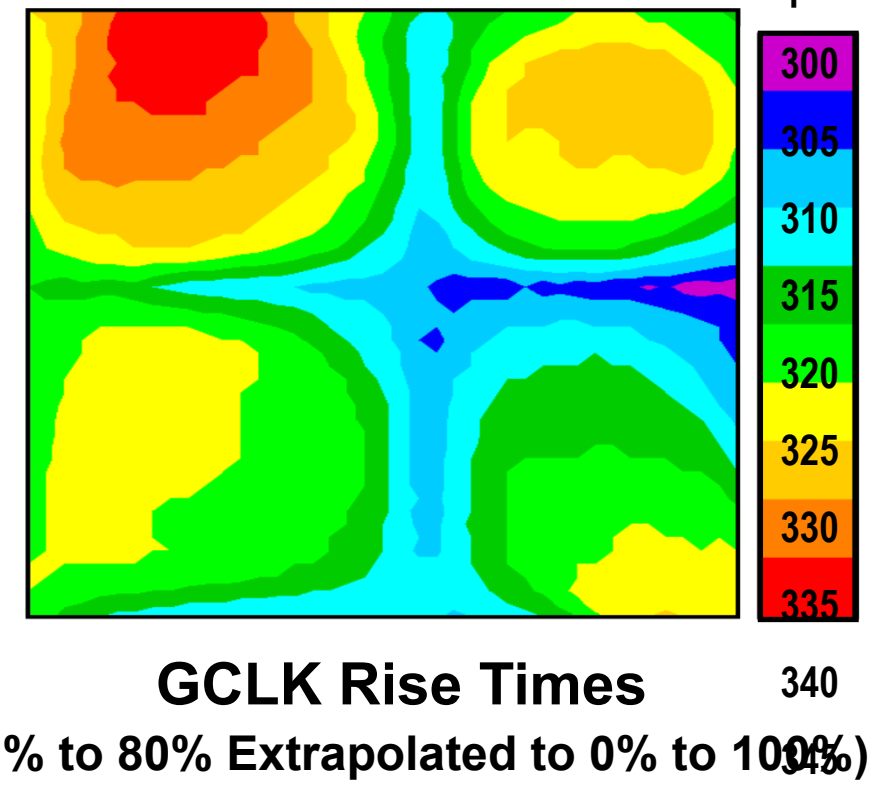
21264 Clocking



EV6 Clock Results



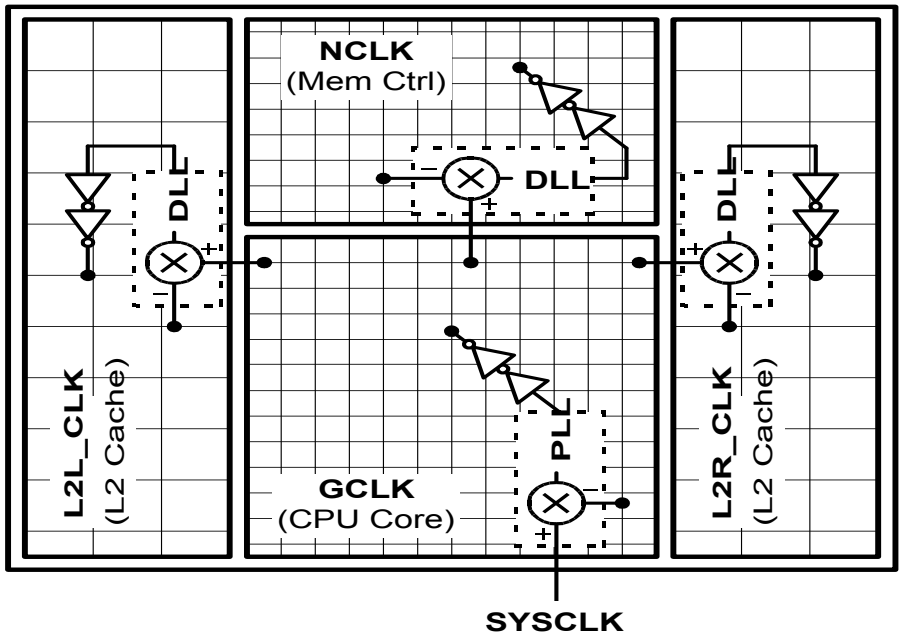
GCLK Skew
(at Vdd/2 Crossings)



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

EV7 Clock Hierarchy (2002)

Active Skew Management and Multiple Clock Domains



+ widely dispersed drivers

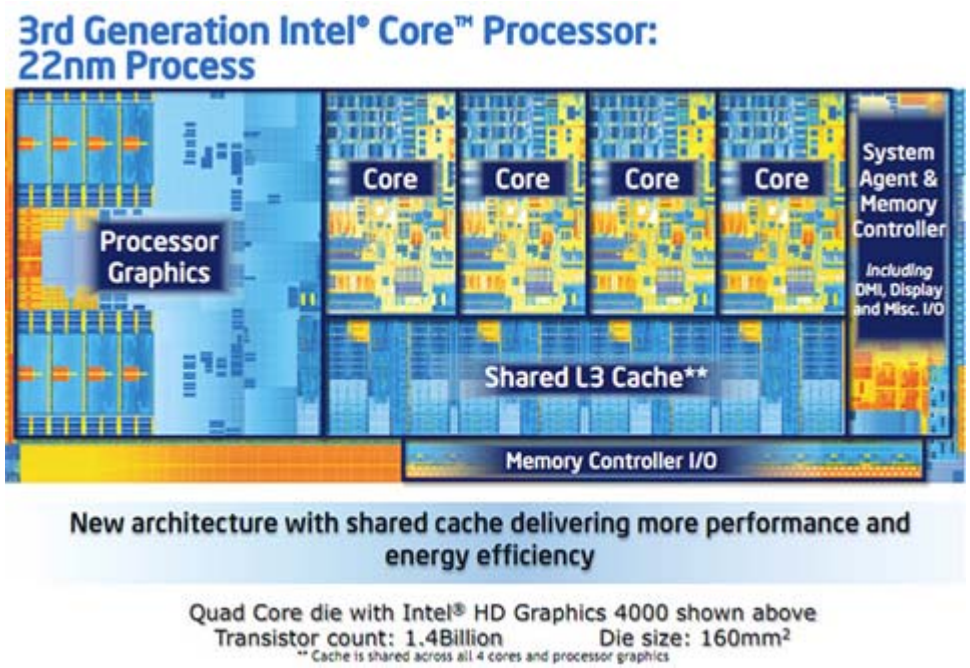
+ DLLs compensate static and low-frequency variation

+ divides design and verification effort

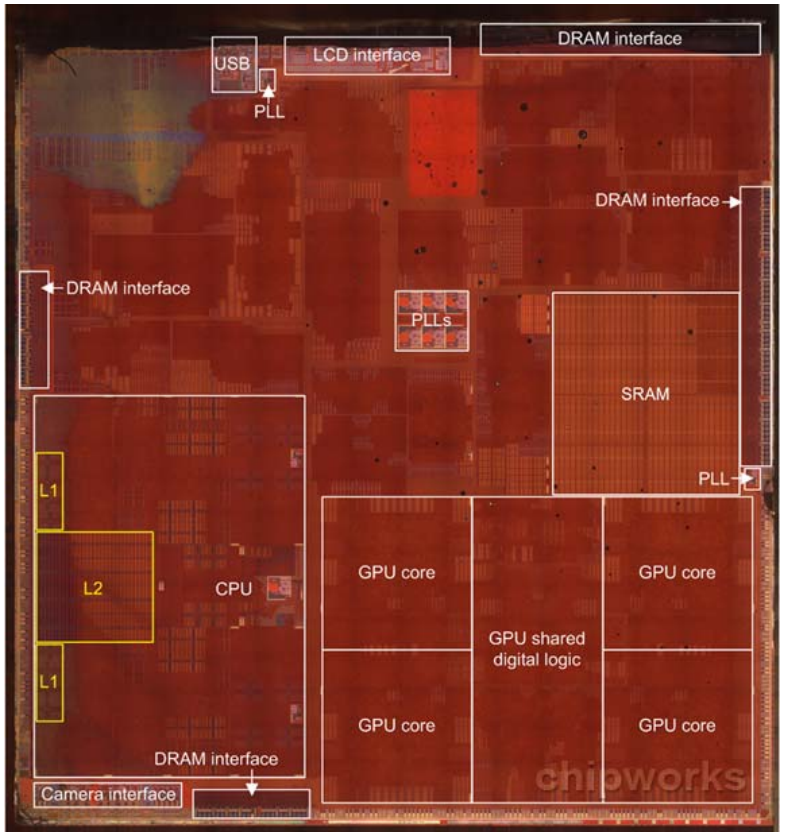
– DLL design and verification is added work

+ tailored clocks

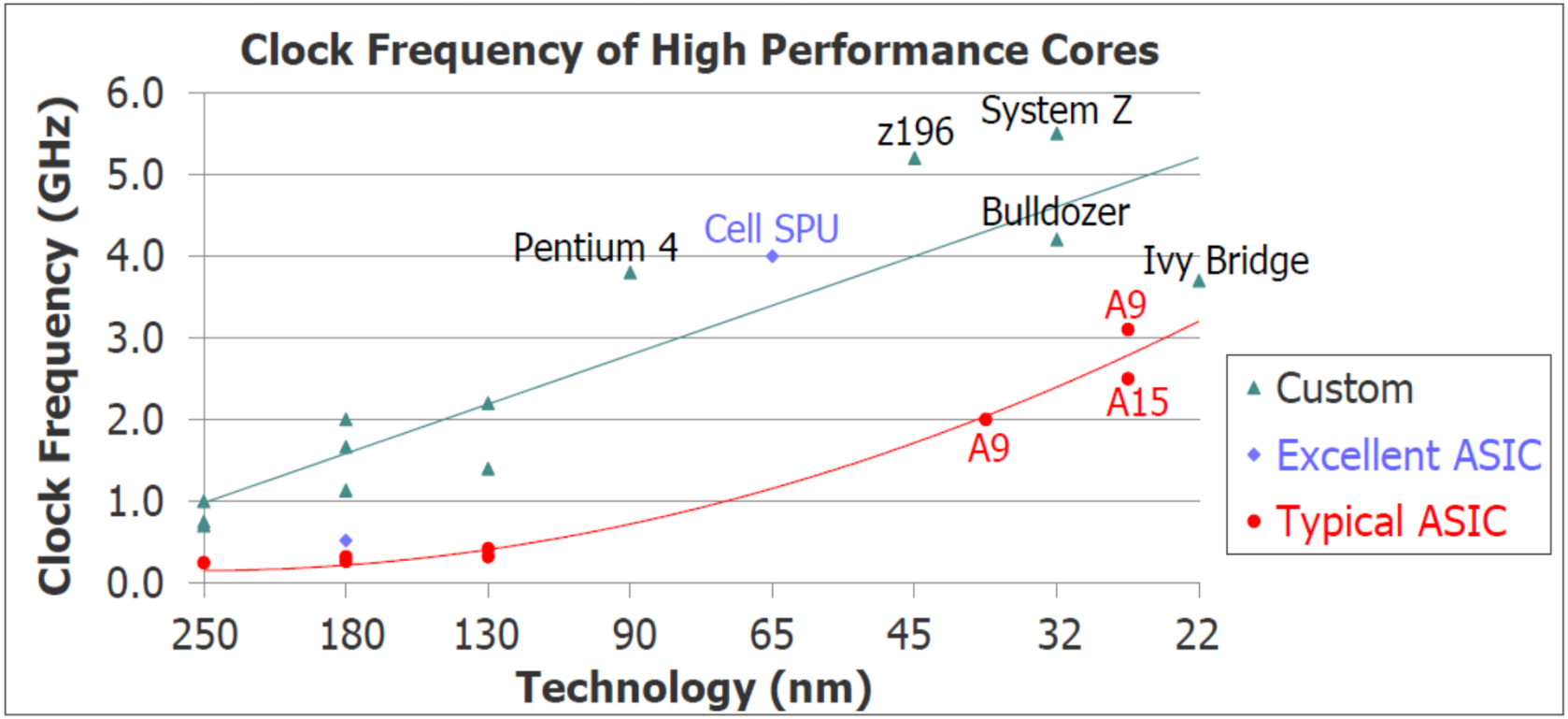
Modern Processors



Apple A7



Clocking in Modern Designs



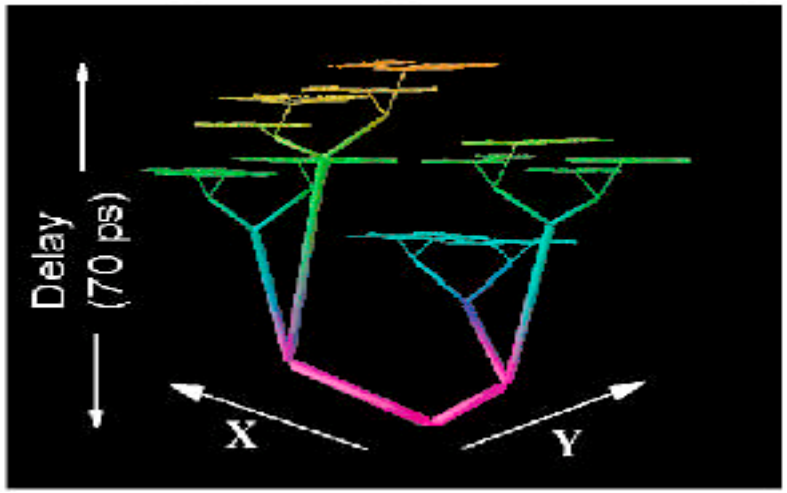
David Chinnery Mentor Graphics

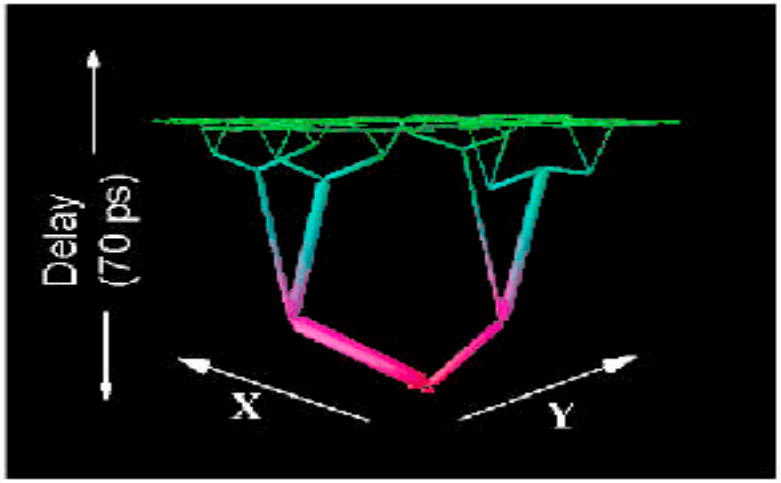
- Custom designs were 3 to 8× faster than ASICs
- Performance gap is below 2× today, custom limited by long design time

Clock Animations

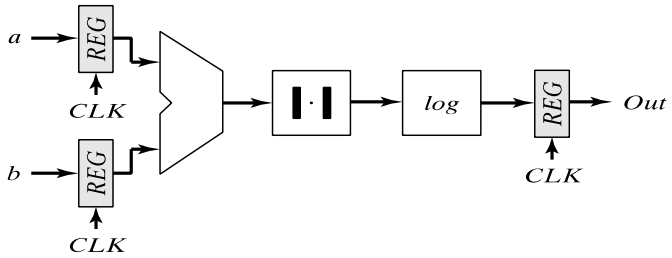
- By Phillip Restle (IBM)

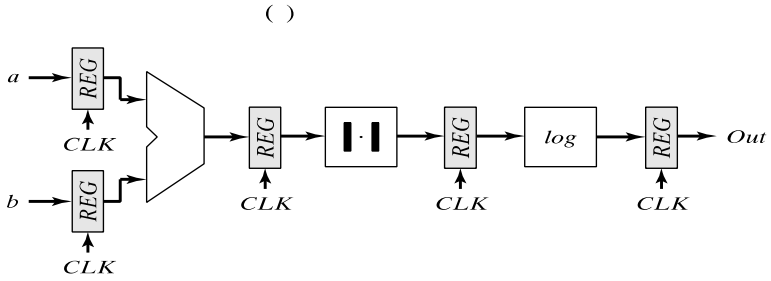
http://www.research.ibm.com/people/r/restle/resonate.html





Pipelining



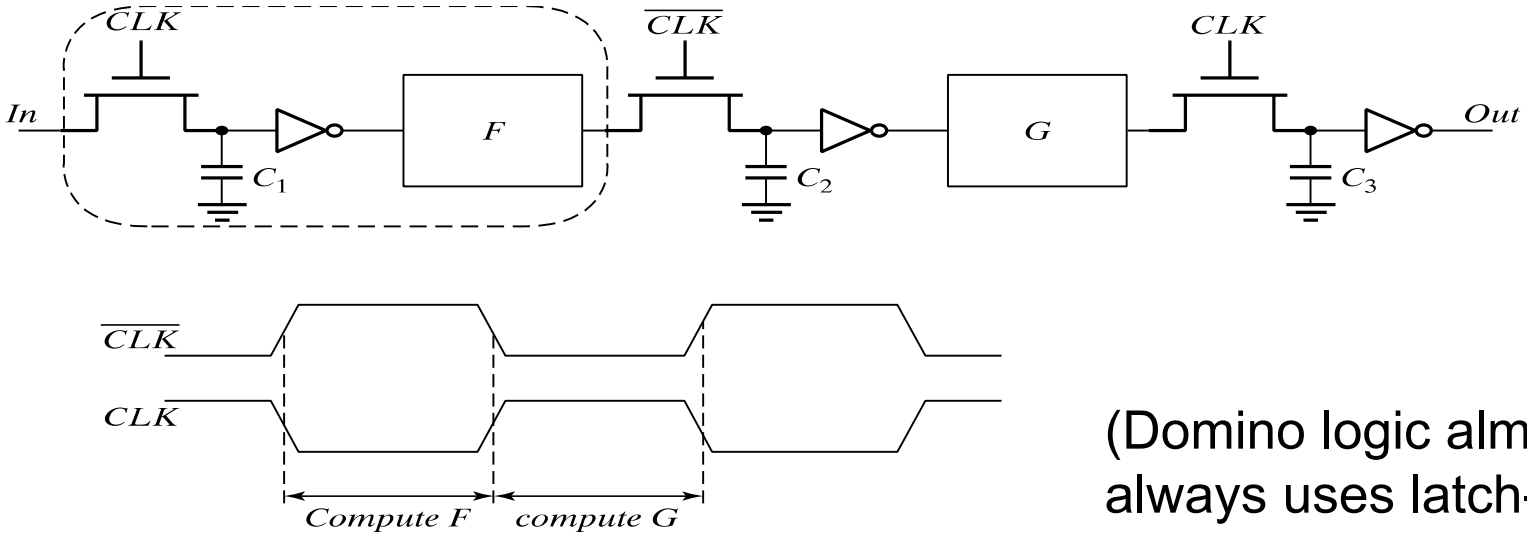


Reference

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	$a_5 + b_5$	$\lvert a_4 + b_4 \rvert$	$\log(\lvert a_3 + b_3 \rvert)$

Pipelined

Latch-Based Clocking

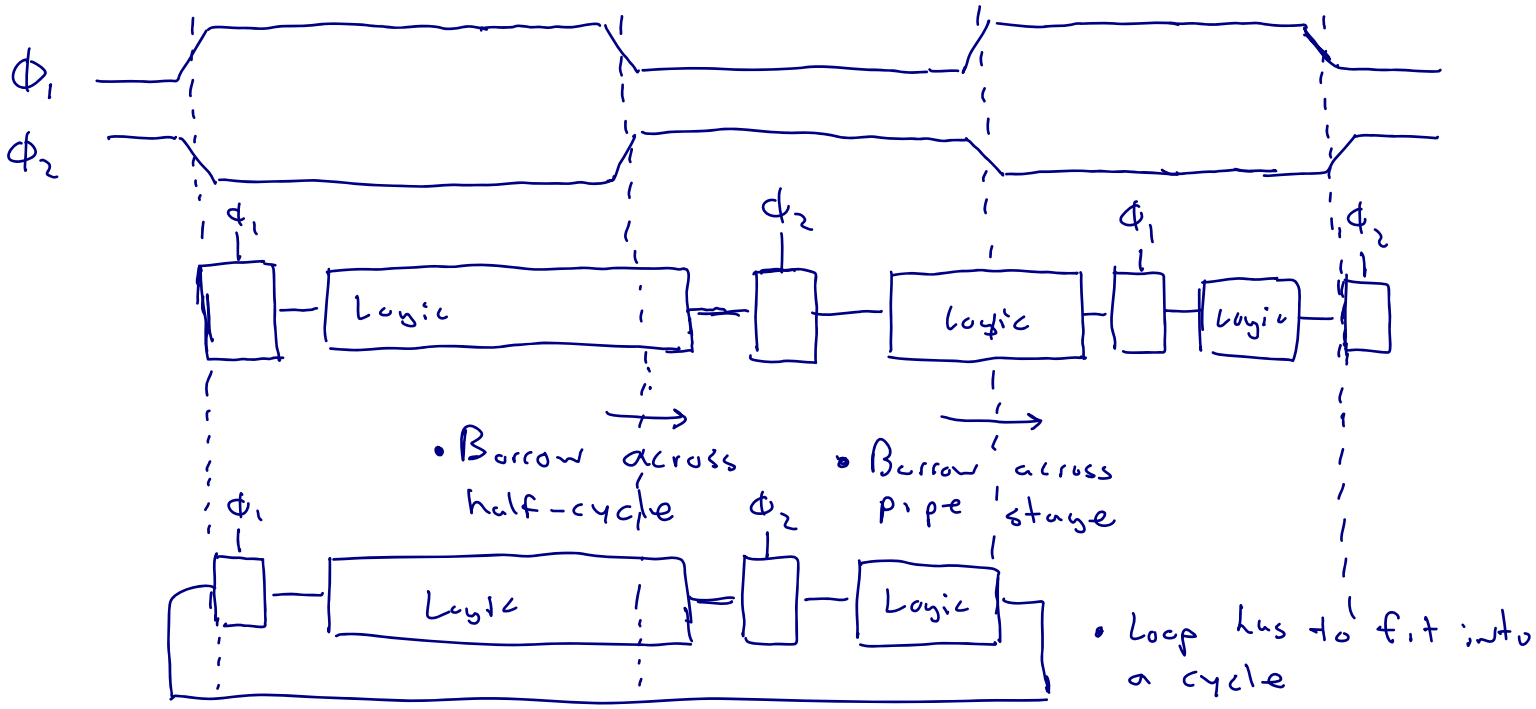


(Domino logic almost always uses latch-based clocking)

Latch vs. Flip-flop

- In a flip-flop based system:
  - Data launches on one rising edge
    - And must arrive before next rising edge
  - If data arrives late, system fails
    - If it arrives early, wasting time
  - Flip-flops have hard edges
- In a latch-based system:
  - Data can pass through latch while it is transparent
  - Long cycle of logic can borrow time into next cycle
    - As long as each loop finished in one cycle

Time Borrowing Example



Latch vs. Flip-flop Summary

- Flip-flops generally easier to use
  - Most digital ASICs designed with register-based timing
- But, latches (both pulsed and level-sensitive) allow more flexibility
  - And hence can potentially achieve higher performance
  - Latches can also be made more tolerant of clock un-certainty
  - More in “Advanced Digital Logic” Class

Pulse-Triggered Latches

Ways to design an edge-triggered sequential cell: