Lecture 6

Clocked Elements

Computer Systems Laboratory
Stanford University
horowitz@stanford.edu

Some material taken from lecture notes by Vladimir Stojanovic and Ken Mai

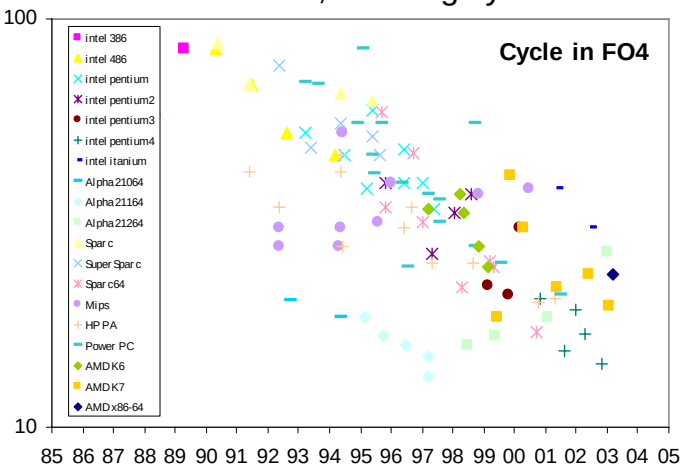
Overview

- Readings (For next lecture on clocking)
  - Gronowski Alpha clocking paper
  - Restle Clock grid paper
  - Harris Variations paper

- Today's topics
  - Latches and flops overview
  - Power and timing metrics
  - High-performance design and low-energy design
  - Examples

Why Are Clocked Elements Important?

- A graph from the first lecture, showing cycle time in FO4



- A clock cycle contains less and less time each generation

We Need Faster Clocked Elements

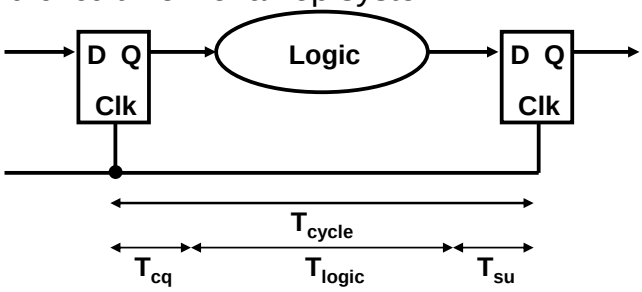
- Note that the previous graph invites us to extrapolate blindly
  - "In just a few years, clock frequencies will go to infinity!!"
  - Reality says clock cycle times will level out
- There is some disagreement on the actual limits
  - "6-8 FO4 per cycle is optimum" (Hrishikesh, ISCA '02)
  - "Yes, but including overhead, more like 18" (Srinivasan, ISM '02)
- Today: 16 FO4/cycle in Pentium4; 11 FO4/cycle in Sony Cell
  - If a flop has an overhead of 3FO4, this is 15%-30% overhead
  - We can fill less of the cycle time with "work"
- Making a faster flop helps performance significantly

We Need Lower Power Clocked Elements

- Recall that chip power density is climbing up
  - "Rocket nozzle!" "Surface of the sun!" "Oh, my!"
  - Because power is $C*V^2f$ and V is not scaling anymore (V != $10L_{gate}$)
- Part of this power is for clock distribution
  - Clocking requires 70% of core power in Power4 (2001)
  - Clocking requires 25% of core power in dual-core Itanium (2005)
- Almost all of this power is in driving the latches/flops at the ends
  - Only around 20-25% of clock power is in clock transmission
  - Saving power at the ends is better than saving power in the wires
- Making a more efficient flop helps power significantly

Timing Overhead, Illustrated

- A standard circuit view of a flop system



  - Flop timing overhead is the "data-to-output(Q)" delay
    - $T_{data\text{-}to\text{-}Q} = T_{setup} + T_{clk\text{-}to\text{-}Q} = T_{su} + T_{cq}$
- Next look at some basic latch and flop designs and metrics
  - But first, an analogy…

An Analogy Of Timing

- Clocked datapaths are like streets with traffic lights
  - Cars moving down the street are data
    - Some cars speed, some cars drive normally, some crawl
    - Principal rule: cannot have two cars collide into each other
  - As a car (data) comes to a light (flop or latch)
    - It passes through if the light is green and waits if it's red
- A clocked system has a master controller for the lights
  - All the lights turn red/green at the same time (as in Manhattan)
    - In some systems, the lights are green very briefly (flops)
    - In other systems, the lights stay green half the time (latches)
  - Satisfy principal rule: ensure all cars only pass one block per light
- The point of traffic lights is to slow down fast cars

The Analogy's Timing Failures

- A traffic light (flop or latch) can fail the principal rule in two ways
  - A car can be too slow to make it to the next block: Max path
    - The car hadn't reached the intersection when the light turned red
  - A car can race through more than one block: Min path
    - The intersection didn't stay clear just when the light went red
- Failures often arise because traffic light timing is not perfect
  - Differences between traffic lights (skew), or local variations (jitter)
- Fixing these failures
  - Max paths are fixable: Slow down the master controller
  - Min paths are not: Rebuild the street (or the light control wires)!
    - Cost of 3 months fab time and $1M in mask costs

By The Way

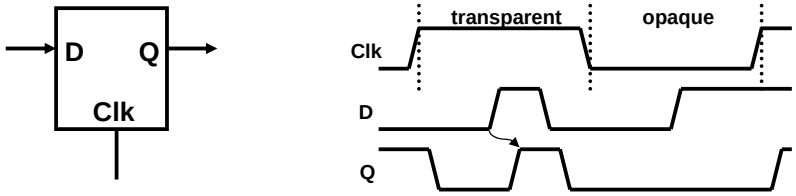
- Do you really need traffic lights?
  - No, if you can guarantee that all cars travel at the same rate
  - Wave-pipelining ensures all datapaths have the same delay
- Do all the lights need to switch at the same time?
  - No, and long blocks might benefit from different light timing
  - Intentional clock skew on chips helps for timing problems
- Do you really need a master controller for all the lights?
  - No, if you have cars (drivers) negotiate between themselves
  - Asynchronous circuits handshake between data items

A Note On Terminology

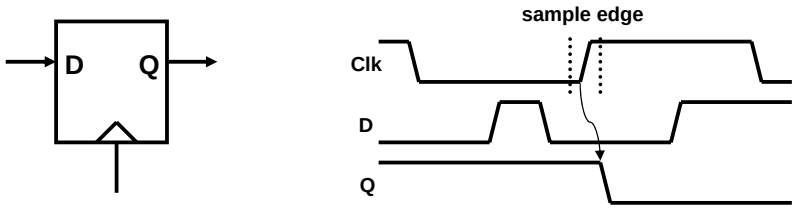
- Avoid saying a latch or flop is “open” or “closed”
  - There is ambiguity in these terms
  - Water flows if you “open a valve”
  - Current stops if you “open a switch”
- The common wording today is a little awkward, but clear
  - A timing element transfers data when it is **transparent**
  - It does not transfer data when it is **opaque**
- Other choices include “blocking” and “non-blocking”
  - The oddest I have ever seen was “Permissive” and “Prohibitive”
  - But don’t use that…

Latches and Flops

- Latch has soft timing: transparent when clock=1



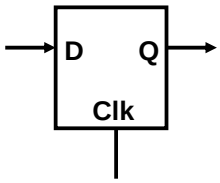
- Flop has hard timing: "transparent" when clock 0→1

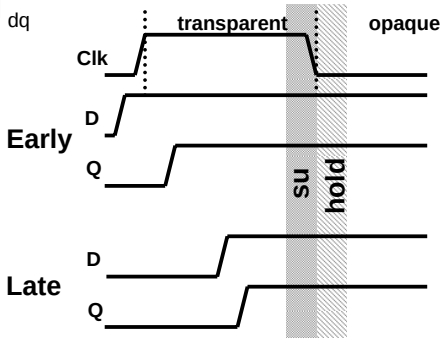


Latch Timing

- Setup and hold times are defined relative to the clock **fall**
  - Setup time: how long before the clock fall must the data arrive
  - Hold time: how long after the clock fall must the data not change

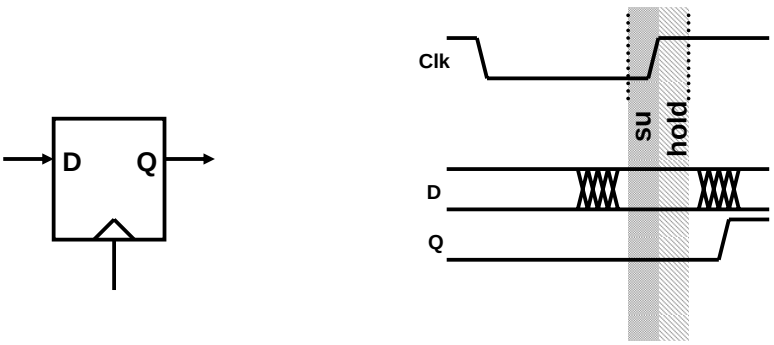
- Delay depends on arrival time of data relative to clock **rise**
  - On early data arrival, delay = T_{cq}
  - On late data arrival, delay = T_{dq}





Flop Timing

- Setup and hold times are defined relative to the clock **rise**
  - Setup time: how long before the clock rise must the data arrive
  - Hold time: how long after the clock rise must the data not change
- Delay is always T_{cq}, as long as data hits the setup constraint

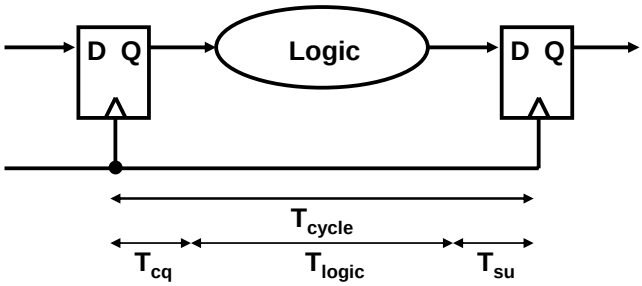


Latch and Flop Timing

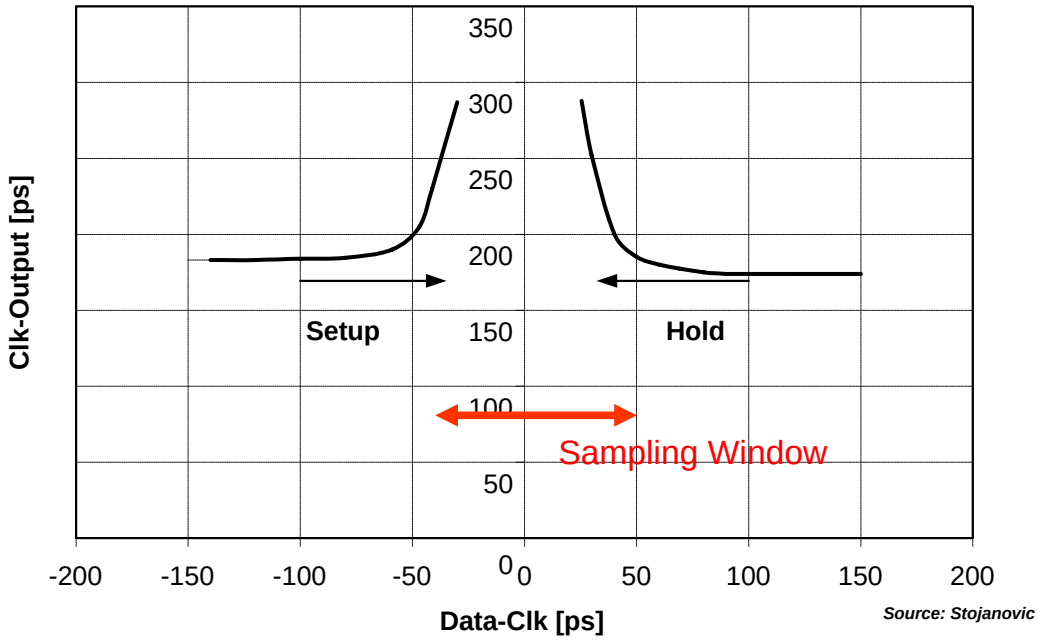
- Softness of latch timing edges allows time borrowing
  - Nominally a latch expects its data when the latch goes transparent
  - But the latch will accommodate late data
    - Until the data runs into the falling edge of the clock (going opaque)
  - Time-borrowing works backwards (“slack forwarding”) and forwards
- Flops generally do not allow time borrowing
- For some latches and flops, setup time is negative
  - The data can change just after the latch goes opaque
  - The latch can still see that data change

Flop Delay

- What's the delay through a flop?
  - Data must get to the flop by a setup time T_{su}
  - Data must traverse the flop and take up T_{cq}
- So the time available to do logic is what's left
  - $T_{logic} = T_{cycle} - T_{su} - T_{cq} - T_{skew}$

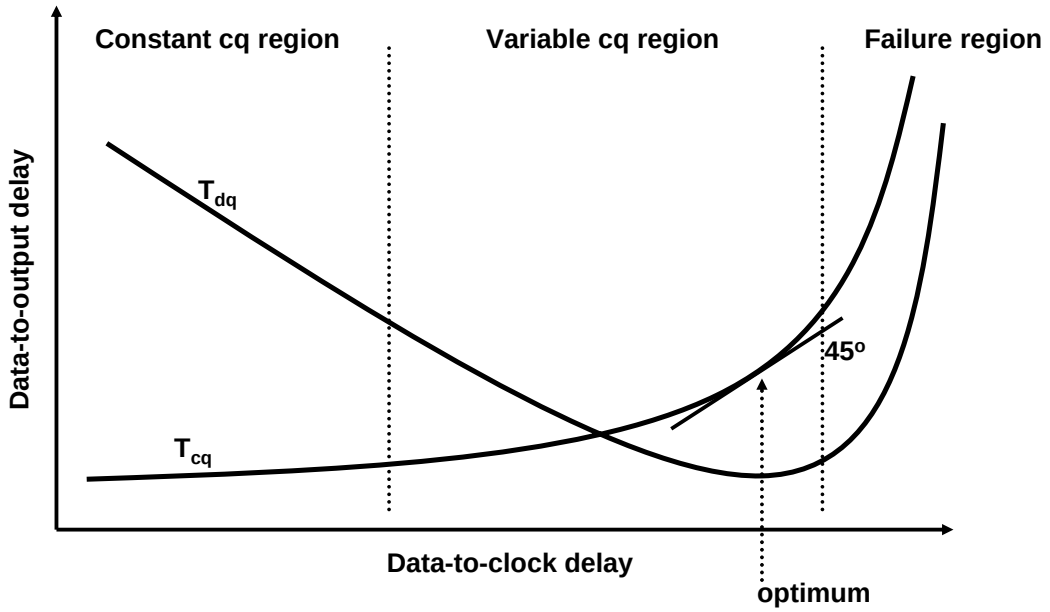


Flop Clk-to-Q Delay Depends On Data-to-Clk



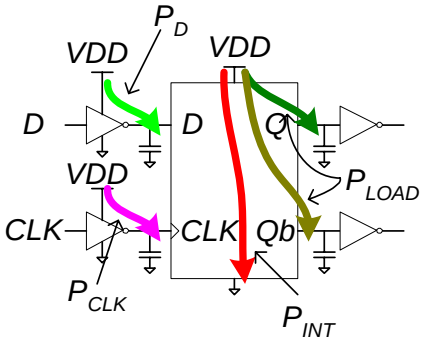
Source: Stojanovic

Examine The Setup Half Of That Graph



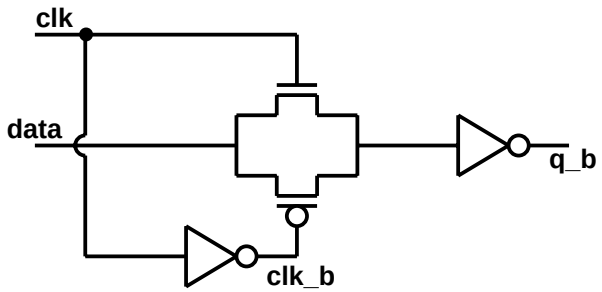
What About Power?

- To fairly compare power of various flops or latches, include
  - External power to drive the clock input
  - External power to drive the data input
  - Internal power required to switch interior nodes
  - Internal power required to drive output load



Simplest CMOS Latch

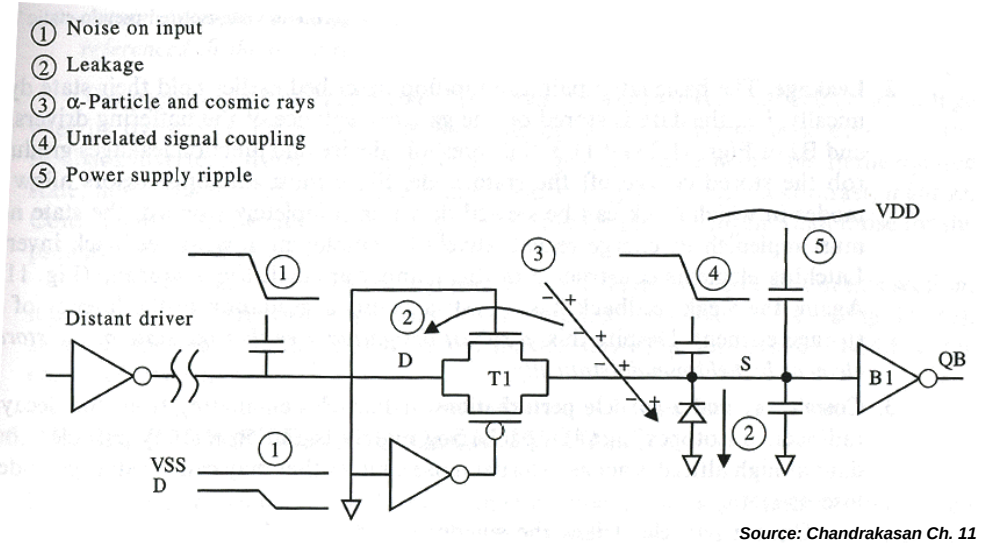
- Basic transparent high latch (Figure 11.2) is simply a passgate



  - Very simple and compact
  - Stores data dynamically – subject to leakage and noise problems

Bad Things Can Happen To This Latch

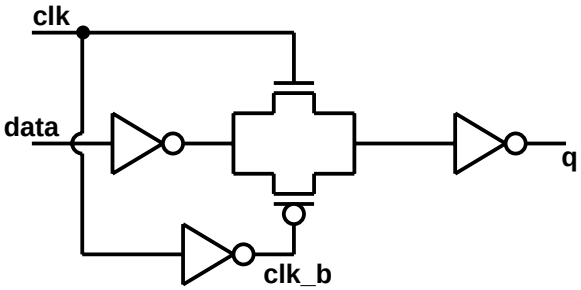
- Various modes of failure, including



Source: Chandrakasan Ch. 11

Buffered Transparent High Latch

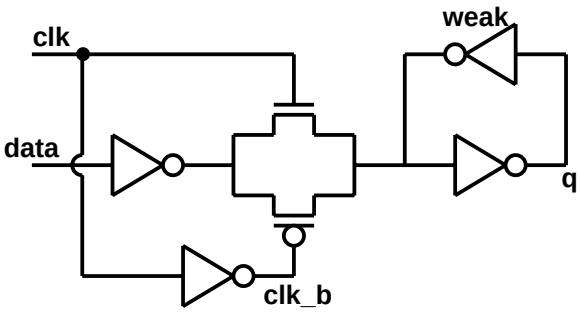
- Avoid input noise with a local input inverter



  - Still have problems with the storage node

Jam Latch

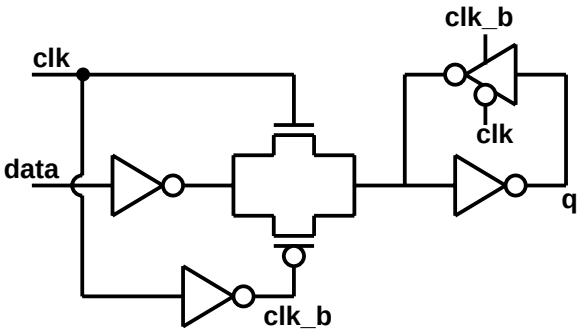
- Make the storage node static



  - Feedback inverter is very weak and loses in a fight
  - Burns power during the fight until the latch flips
  - Beware mixed process corners (SF, FS) for overpowering latch!

Tristate Latch

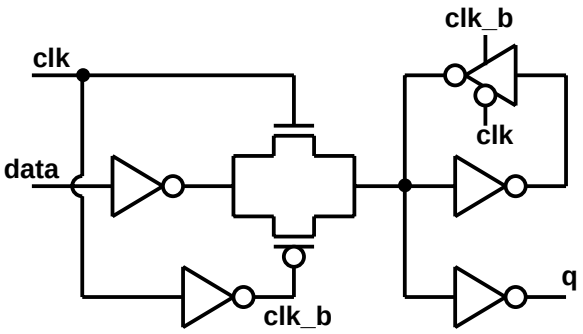
- Prevent the fight by shutting off the feedback device



  - Count on constant input drive during latch transparency
  - Note that input inverter+passgate can be a tristate as well

Robust CMOS Latch

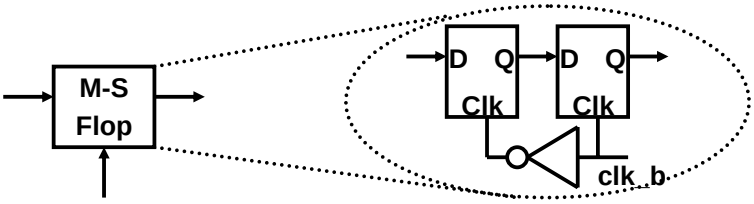
- Don't take the output from the storage node directly



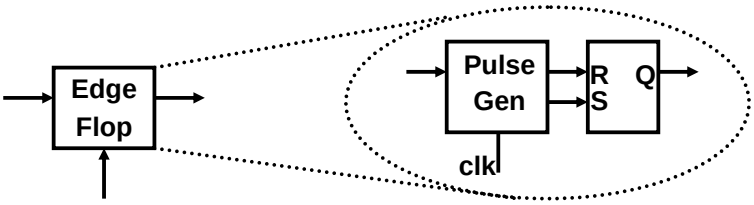
  - This is a good, safe latch design
  - Not the fastest in the world (can trade-off speed for DANGER)
  - Setup and hold times are close to 0

Flip-Flops Come In Several Flavors

- You can make a flip-flop out of two back-to-back latches

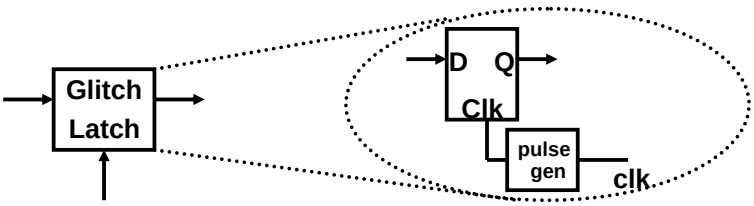


- You can make it out of a edge-triggered element, plus SR latch



Flip-Flops Come In Several Flavors, con't

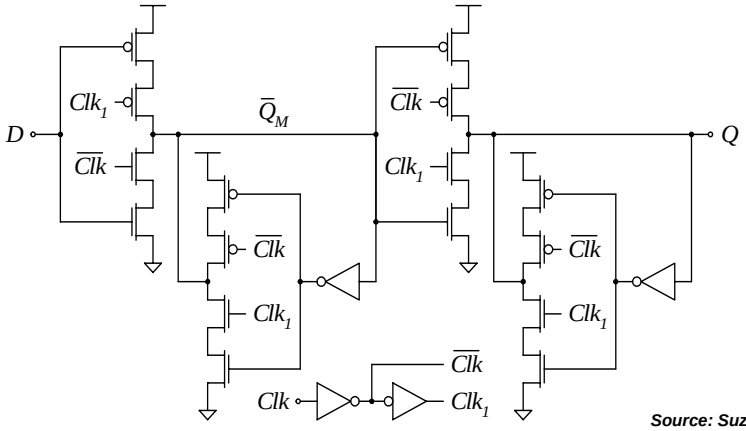
- You can also make a flip-flop out of a pulsed (glitch) latch



- All three flavors are (almost) functionally indistinguishable
  - Although they may react differently to clock skew
  - Will look at this in a later lecture

Flavor 1: Simplest Flop Is Master-Slave

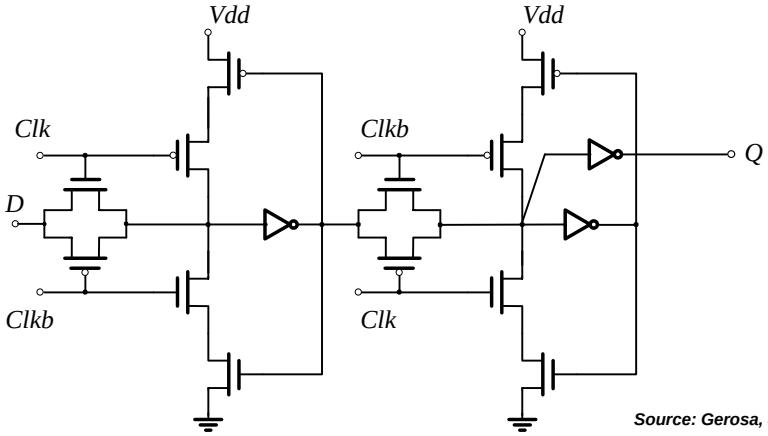
- World's first LSI calculator chip (Tokyo Shibaura Electric)
  - Real "old-school" but it's a master-slave
  - All tristates instead of passgates



Source: Suzuki, JSSC 1973

Flavor 1: Another Master-Slave

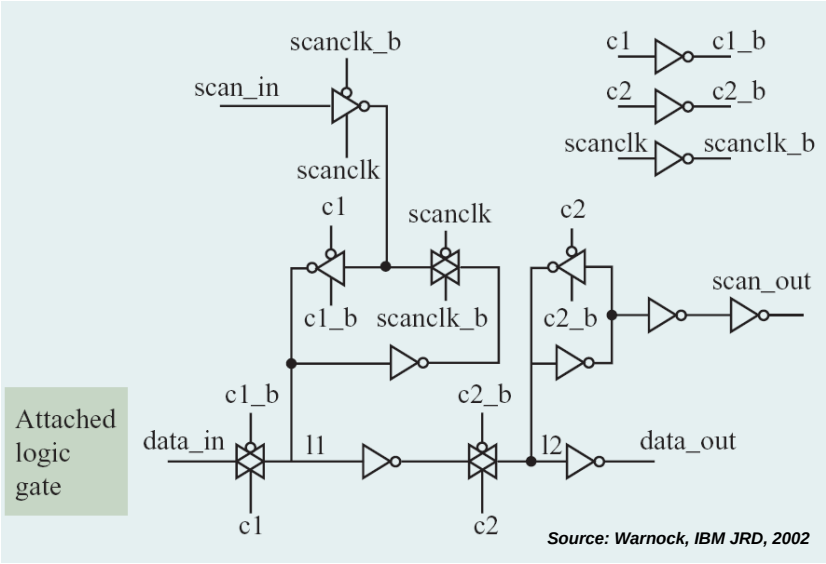
- PowerPC 603 flop
  - Note this is a negative edge-triggered flop (clks are reversed)
  - Faster than C2MOS, but at worse input noise (no tristate)
  - Master clk usually generated from slave (why not vice versa?)



Source: Gerosa, JSSC 1994

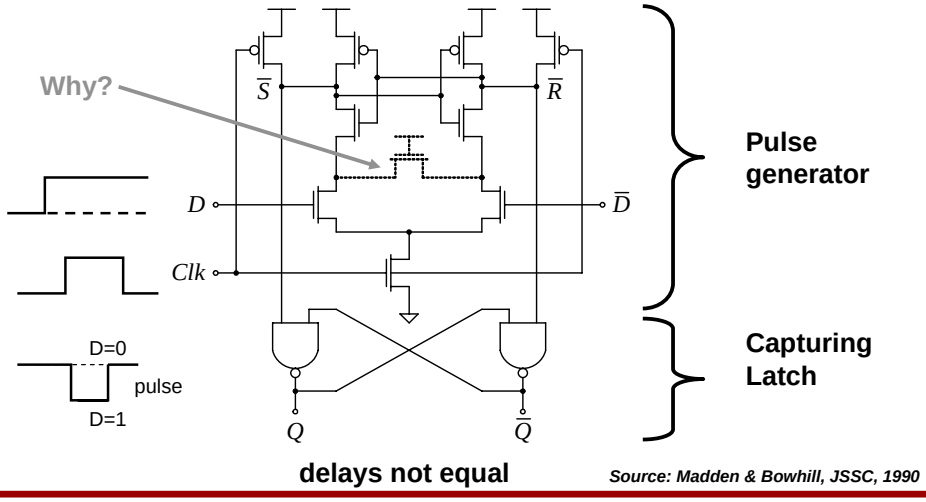
Flavor 1: Yet Another Master-Slave (With Scan)

- IBM Power4 latch with scan-ability (very common today)



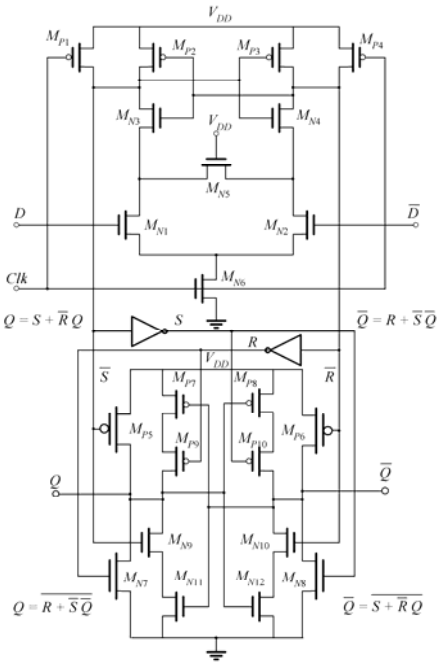
Flavor 2: True Edge-Triggered Flops

- Dec 21264 Alpha flop (Madden & Bowhill, '90, Matsui '94)
  - A sense-amp (pulse generator) followed by a capturing RS latch
  - Negative setup time available



(Flavor 2) Improved "Strong-Arm Flop"

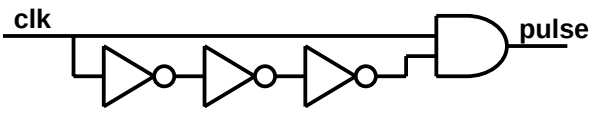
- Faster S-R stage
  - Sized for improved switching
  - Bigger drivers and smaller keepers
- Symmetric Q and Q_b delays
- Reduce the hysterisis of the latch
  - Slam the node high/low strongly
  - Then in precharge, hold it weakly
  - Makes it faster, but at what cost?
    - Coupling immunity
    - Yet another application of NFL
      - "No free lunch"



Source: Nikolic & Stojanovic, ISSCC '99

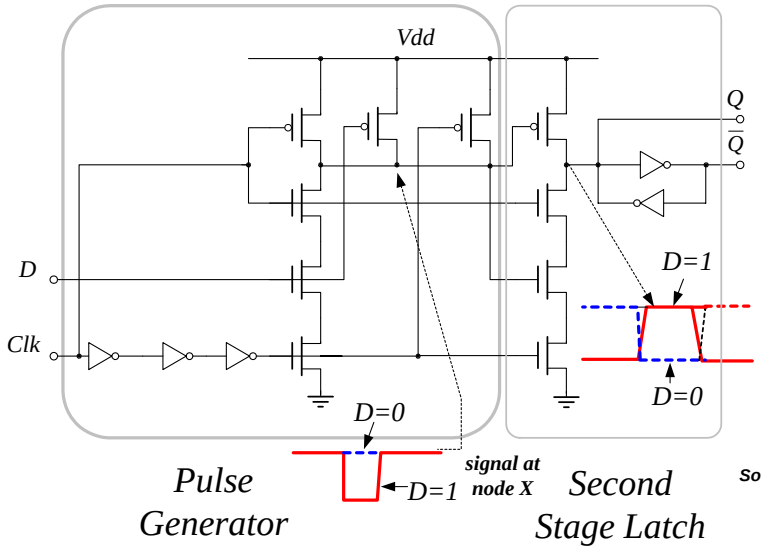
Flavor 3: Glitch Latch or Pulse Latch

- Just like a standard latch, only with a pulsed (glitched) clock
  - Looks and smells like a flip-flop
  - Pulse gives you negative setup time and a soft edge (latch-like)
- Generating the pulsed clock can be tricky over process corners
  - A "clock chopper" or "single-shot"
  - Do this locally at the pulse latch for each latch or small group
    - Distributing this pulse is dangerous: pulses disappear
  - So often combine this functionality into the latch itself



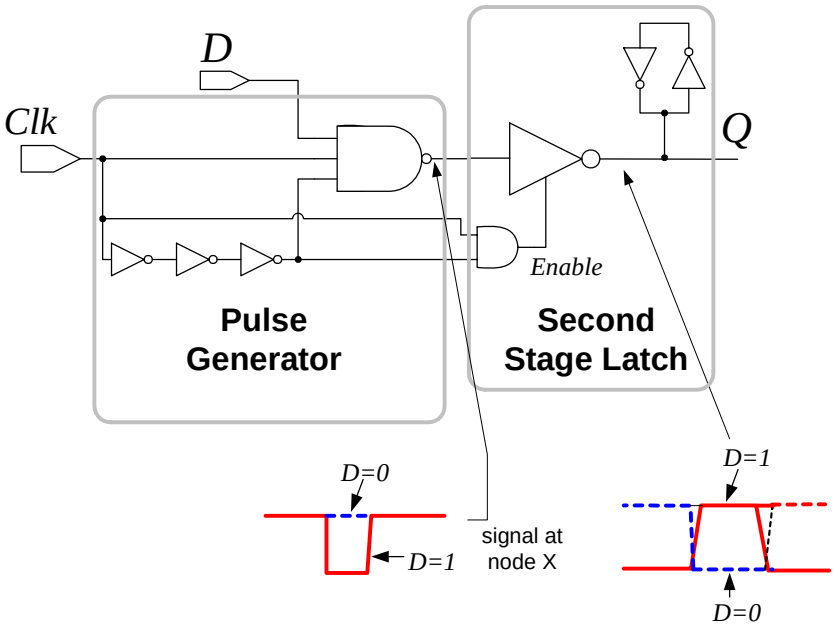
(Flavor 3) Glitch Latch

- AMD K6 latch, called a “Hybrid Latch Flip-Flop” (i.e., glitch latch)



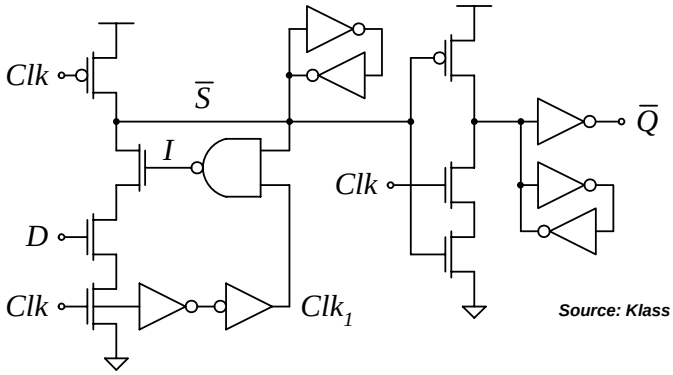
Source: Partovi ISSCC '96

(Flavor 3) Abstraction of the HLFF



(Flavor 3) A “Semi-Dynamic” Flip-Flop

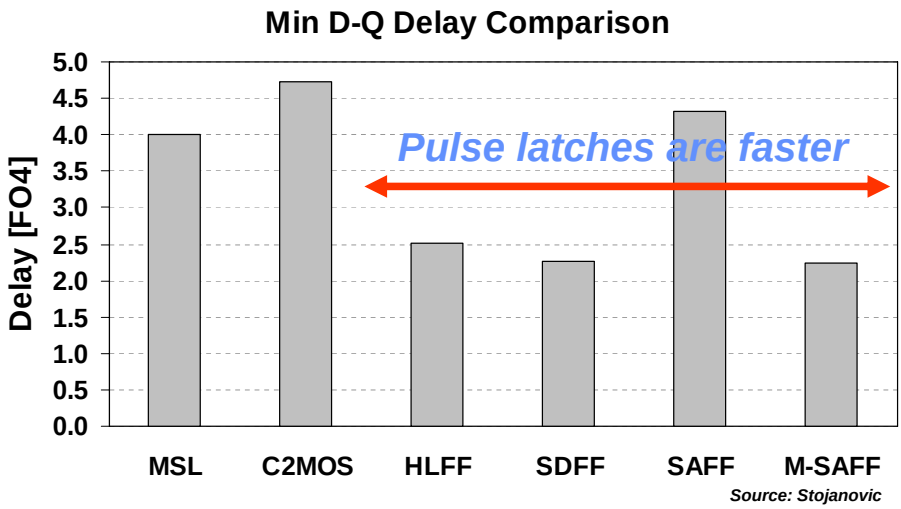
- Sun UltrasparcIII pulsed latch



  - Dynamic pulse generator, unlike the NAND3 in the HLFF
  - Beware the 1-1 glitch

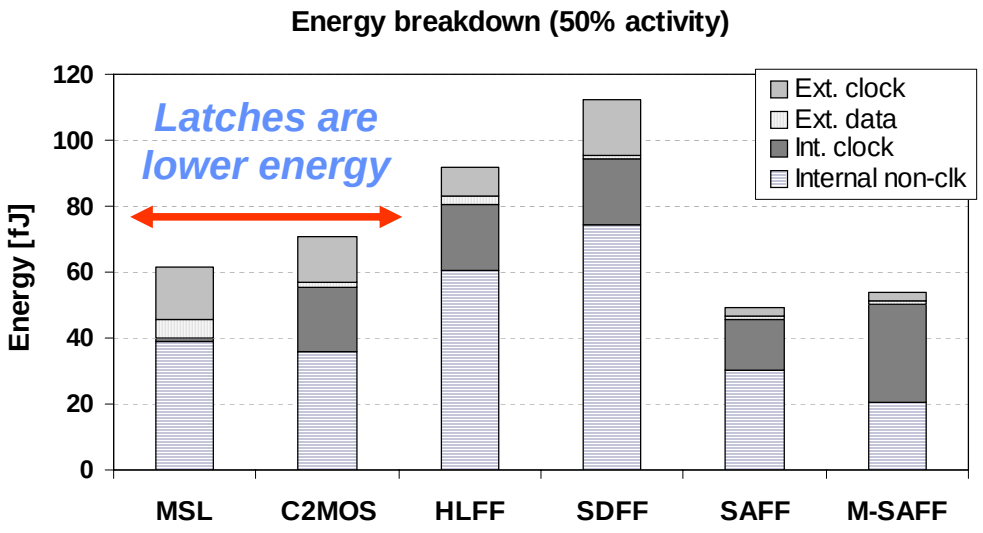
Delay Comparisons

- Driving a load of 14 inverters, in a 0.18μm technology



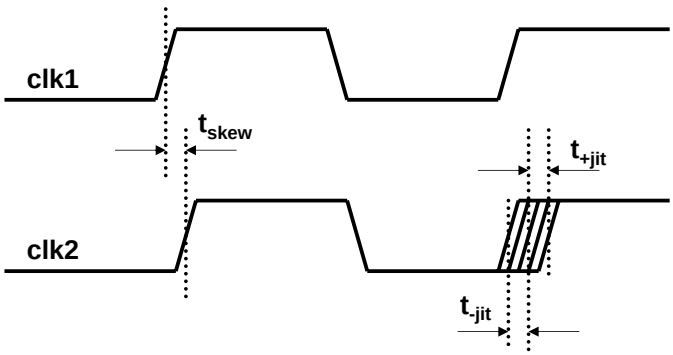
Energy Comparisons

- Driving a load of 14 inverters, in a 0.18μm technology



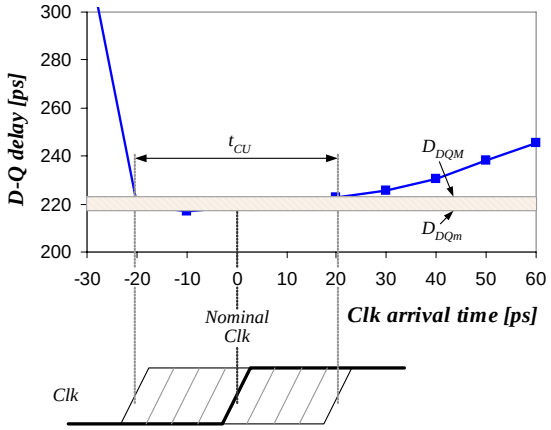
What About Imperfect Clocks?

- Clocks do not always arrive “on time”
  - Clocks to different clocked elements arrive at different times: SKEW
  - One clock will arrive at different times from cycle to cycle: JITTER
- Clock performance is a function of the distribution grid (later)



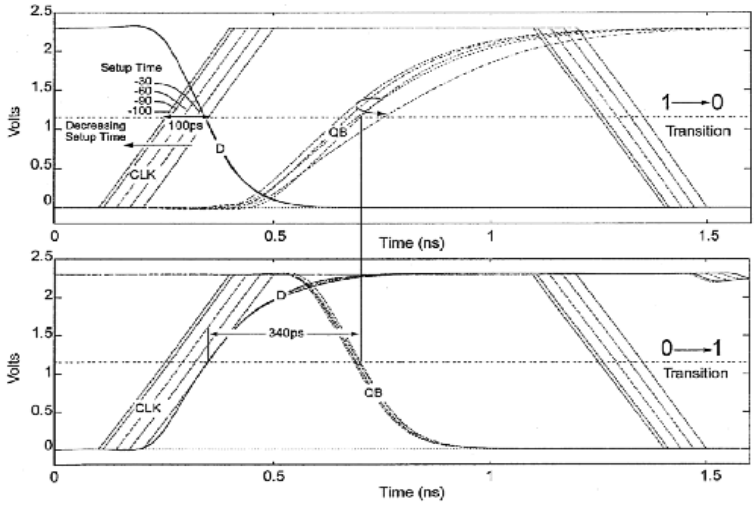
Do Clocked Elements Have Transparency?

- Change in D-Q delay < clock uncertainty
  - The clocked element absorbs some of the clock uncertainty
- There is a range of clk arrivals for which D-Q is constant
  - For that range the clocked element looks like a combinational gate



"Clock Uncertainty Absorption"

- Helps to mitigate the effects of clock skew (HLFF shown below)



- More skew-tolerant circuits will be discussed in a later lecture