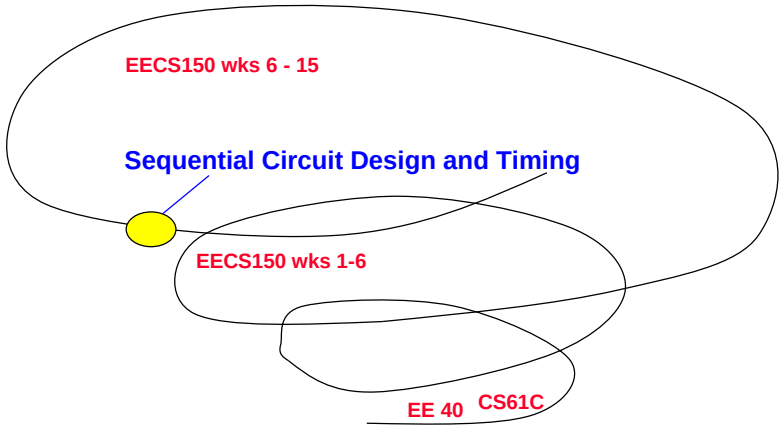
EECS 150 - Components and Design Techniques for Digital Systems

Lec 22 – Sequential Logic - Advanced

David Culler
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~culler
http://inst.eecs.berkeley.edu/~cs150

Traversing Digital Design



RTL & ISA

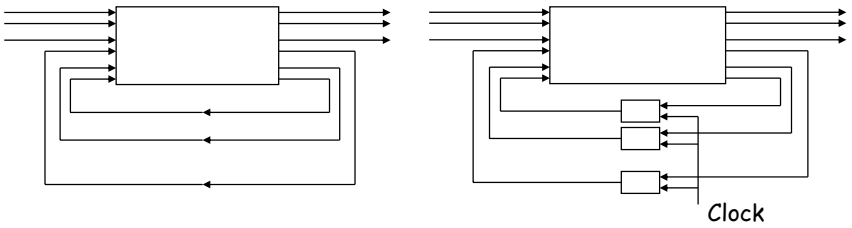
Types of Latches

- We have focused on D-flips
  - D latch => D FlipFlop => Registers (ld, clr)
  - Most commonly used today (CMOS, FPGA)
- Many other types of latches
  - RS, JK, T
  - Should be familiar with these too
- Opportunity to look much more closely at timing behavior

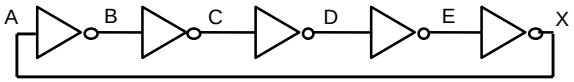
- Latch vs Flip Flops
- Timing Methodology

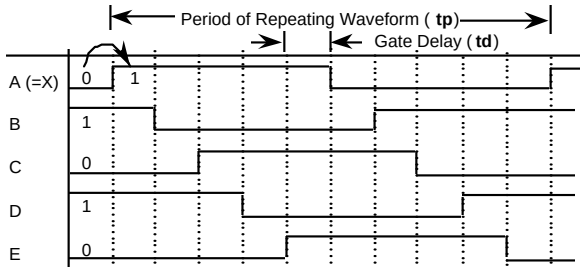
Recall: Forms of Sequential Logic

- **Asynchronous sequential logic – "state" changes occur whenever state inputs change (elements may be simple wires or delay elements)**
- **Synchronous sequential logic – state changes occur in lock step across all storage elements (using a periodic waveform - the clock)**



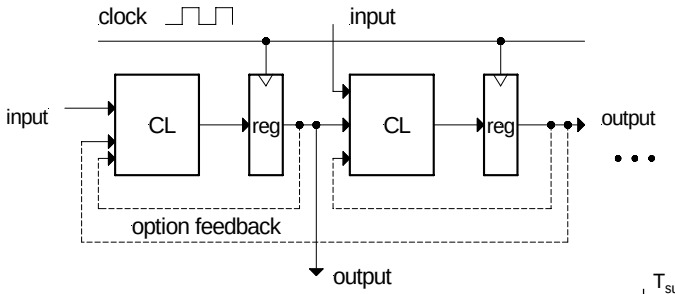
Example – ring oscillator

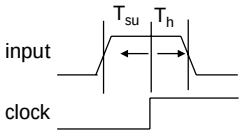




(b) Timing waveform

Recall: General Model of Synchronous Circuit

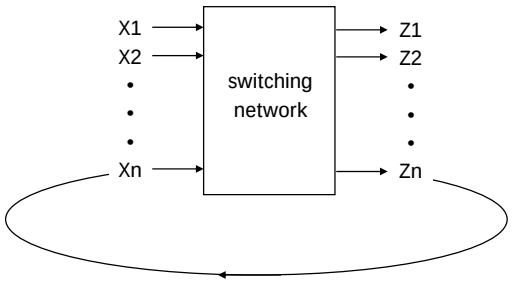




- **Our methodology so far:**
  - **registers as D flipflops with common control**
  - **Single-phase clock, edge triggered design**
- **Assumptions underlying the clean abstraction**
  - **Input to FF valid a setup time before clock edge**
  - **Outputs don't change too quickly after clock edge (hold time)**
    - » **Clk-to-Q => hold time**

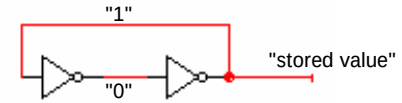
Circuits with Feedback

- **How to control feedback?**
  - **What stops values from cycling around endlessly**

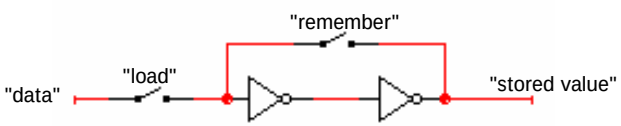


Simplest Circuits with Feedback

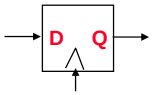
- Two inverters form a static memory cell
  - Will hold value as long as it has power applied

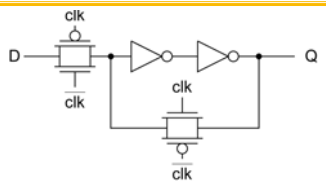


- How to get a new value into the memory cell?
  - Selectively break feedback path
  - Load new value into cell

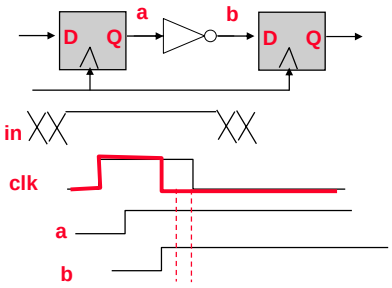


Latches



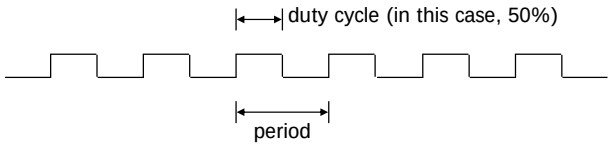


- Level-sensitive latch
  - holds value when clock is low
  - Transparent when clock is high
- What does it take to build a consistent timing methodology with only latches?
  - Very hard! All stages transparent at same time.
  - Require that *minimum* propagation delay is greater than high phase of the clock (duty period)



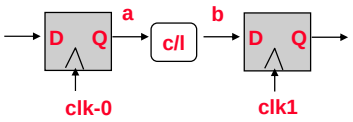
Clocks

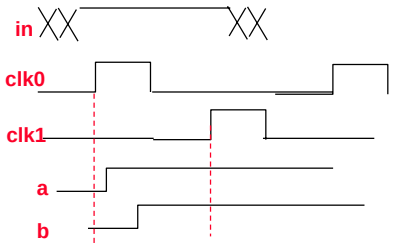
- Used to keep time
  - Wait long enough for inputs (R' and S') to settle
  - Then allow to have effect on value stored
- Clocks are regular periodic signals
  - Period (time between ticks)
  - Duty-cycle (time clock is high between ticks - expressed as % of period)

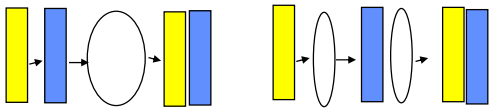


Two-phase non-overlapping clocks

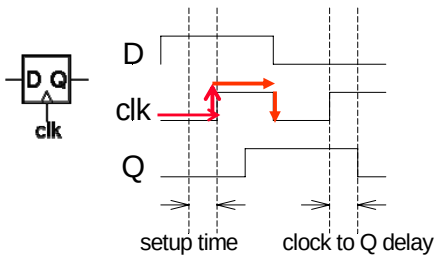
- Sequential elements partition into two classes
- phase0 ele'ts feed phase1
- phase1 ele'ts feed phase0
- Approximate single phase: each register replaced by a pair of latches on two phases
- Can push logic across (retiming)
- Can always slow down the clocks to meet all timing constraints

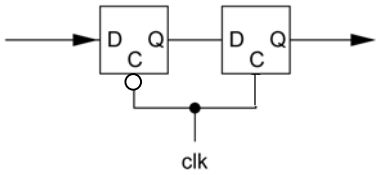




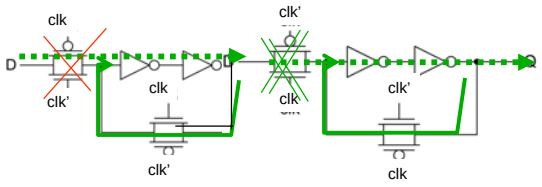


Master-Slave Structure



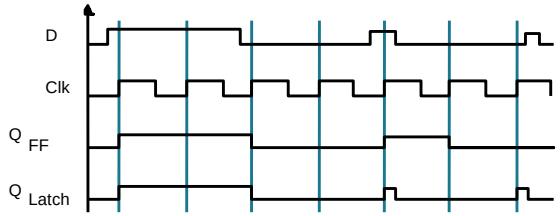


- **Construct D flipflop from two D latches**



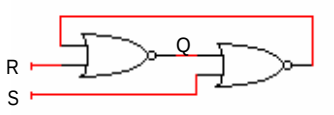
Latches vs FlipFlips

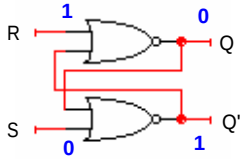
- **Level sensitive vs edge triggered**
- **Very different design methodologies for correct use**
- **Both are clocked, but latch is asynchronous**
  - **Output can change while clock is high**



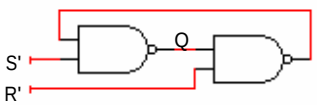
Asynchronous R-S Latch

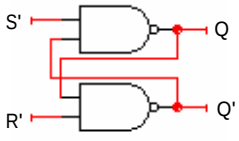
- **Cross-coupled NOR gates**
  - **Similar to inverter pair, with capability to force output to 0 (reset=1) or 1 (set=1)**





- **Cross-coupled NAND gates**
  - **Similar to inverter pair, with capability to force output to 0 (reset=0) or 1 (set=0)**



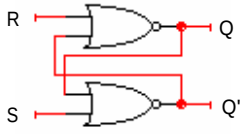


State Behavior of R-S latch

- **Transition Table**

S(t)	R(t)	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	

characteristic equation
$Q(t+\Delta) = S + R' Q(t)$

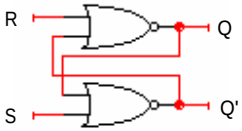


Q(t) \ S,R	0,0	0,1	1,1	1,0
0	0	X	1	1
1	0	X	1	0

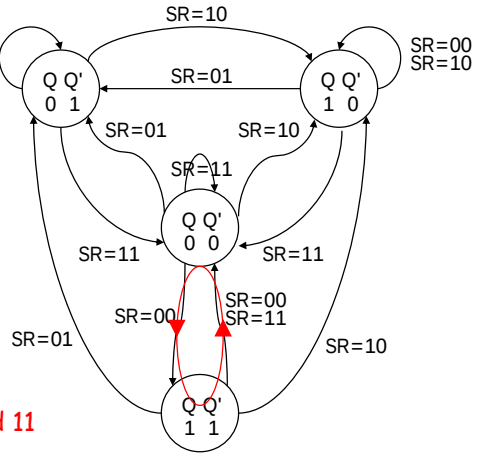
S
R

- **Sequential (output depends on history when inputs R=0, S=0) but asynchronous**

Theoretical R-S Latch Behavior

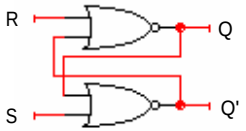


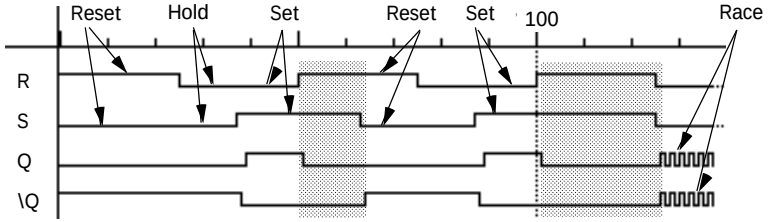
- **State Diagram**
  - **States: possible values**
  - **Transitions: changes based on inputs**



possible oscillation
between states 00 and 11

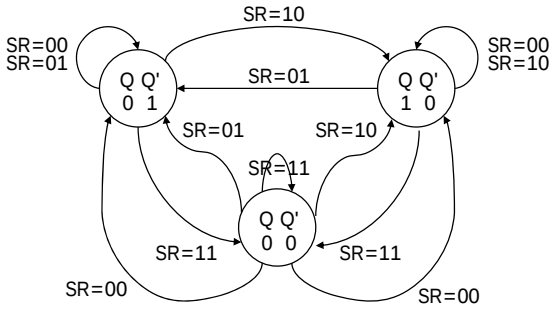
Timing Behavior





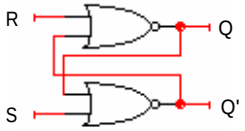
Observed R-S Latch Behavior

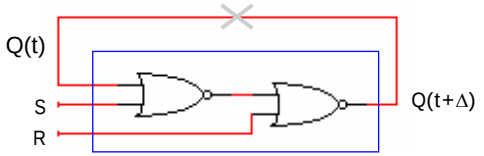
- **Very difficult to observe R-S latch in the 1-1 state**
  - **One of R or S usually changes first**
- **Ambiguously returns to state 0-1 or 1-0**
  - **A so-called "race condition"**
  - **Or non-deterministic transition**



R-S Latch Analysis

- **Break feedback path**





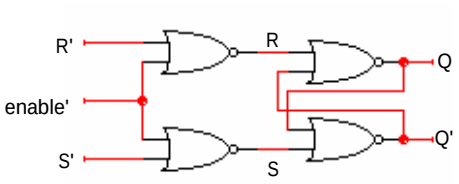
S	R	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	

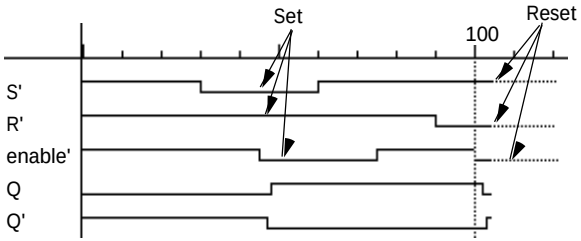
Q(t) \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

characteristic equation
$Q(t+\Delta) = S + R' Q(t)$

Gated R-S Latch

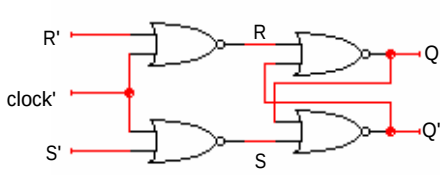
- **Control when R and S inputs matter**
  - **Otherwise, the slightest glitch on R or S while enable is low could cause change in value stored**
  - **Ensure R & S stable before utilized (to avoid transient R=1, S=1)**

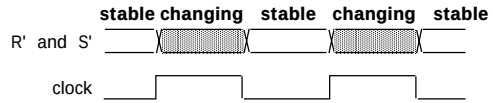




Towards a Synchronous Design

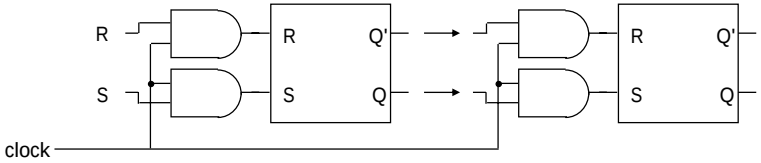
- **Controlling an R-S latch with a clock**
  - **Can't let R and S change while clock is active (allowing R and S to pass)**
  - **Only have half of clock period for signal changes to propagate**
  - **Signals must be stable for the other half of clock period**





Cascading Latches

- **Connect output of one latch to input of another**
- **How to stop changes from racing through chain?**
  - **Need to control flow of data from one latch to the next**
  - **Advance from one latch per clock period**
  - **Worry about logic between latches (arrows) that is *too fast***
    - » **Shortest paths, not critical paths**

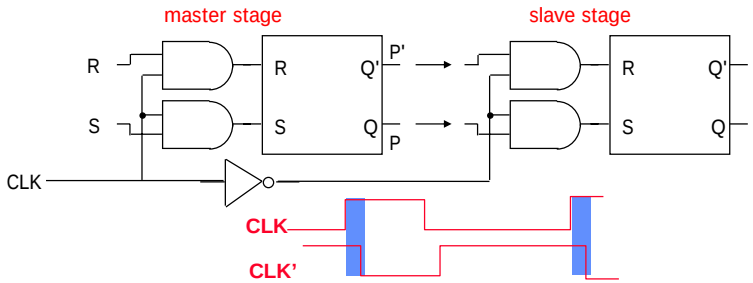


Announcements

- **Guest Lecture, Nov 29, Dr. Robert Iannucci, CTO Nokia**
- **Sarah Lecture on Testing Methodology Thurs.**
- **HW out tonight, due before Break**
- **Lab lecture covers "final point"**
- **Wireless CP this week**
- **Next week TAs will do extended office hours M-W rather than formal lab.**
- **Final Check off week 14**
- **No Class Dec 6.**
- **Final report Friday Dec. 7. Sign up for 10 min slots**
  - **5 min presentation, 5 min Q&A**
  - **Arrive 20 mins before scheduled slot to set up**

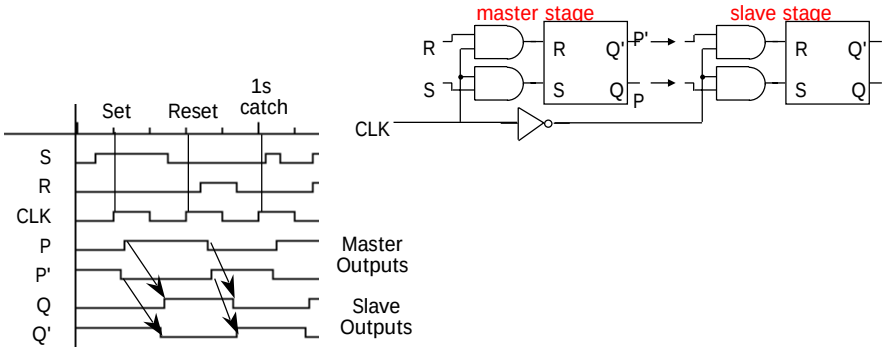
Master-Slave Structure

- **Break flow by alternating clocks (like an air-lock)**
  - **Use positive clock to latch inputs into one R-S latch**
  - **Use negative clock to change outputs with another R-S latch**
- **View pair as one basic unit**
  - **master-slave flip-flop**
  - **twice as much logic**
  - **output changes a few gate delays after the falling edge of clock but does not affect any cascaded flip-flops**



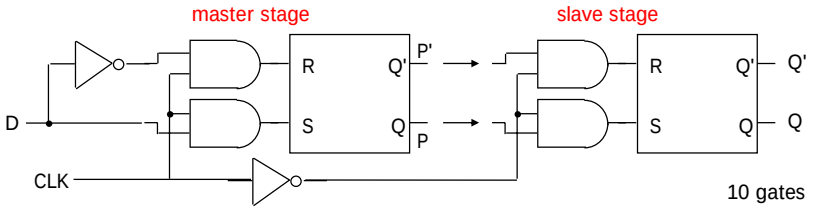
The 1s Catching Problem

- **In first R-S stage of master-slave FF**
  - **0-1-0 glitch on R or S while clock is high "caught" by master stage**
  - **Leads to constraints on logic (feeding the latch) to be hazard-free**



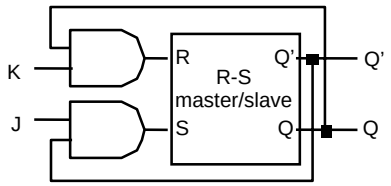
D Flip-Flop

- **Make S and R complements of each other in Master stage**
  - **Eliminates 1s catching problem**
    - » **Input only needs to settle by clock edge**
  - **Can't just hold previous value (must have new value ready every clock period)**
  - **Value of D just before clock goes low is what is stored in flip-flop**
  - **Can make R-S flip-flop by adding logic to make D = S + R' Q**



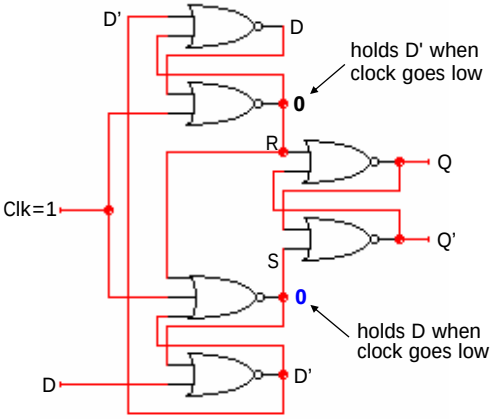
JK Flip Flops

J(t)	K(t)	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	



(neg) Edge-Triggered Flip-Flops

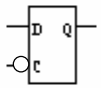
- **More efficient solution: only 6 gates**
  - **sensitive to inputs only near edge of clock signal (not while high)**



negative edge-triggered D flip-flop (D-FF)

4-5 gate delays

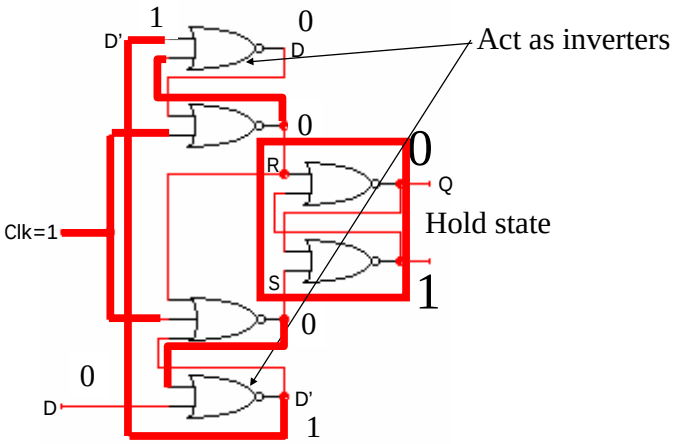
must respect setup and hold time constraints to successfully capture input



characteristic equation
Q(t+1) = D

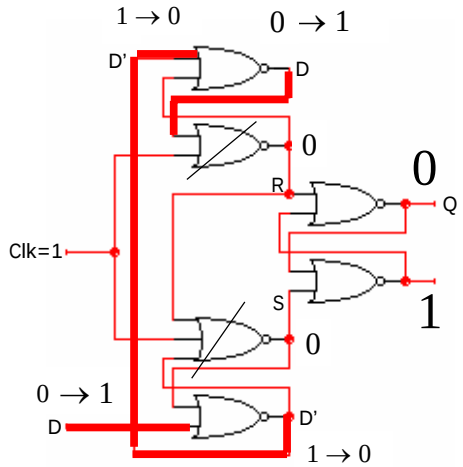
Edge-Triggered Flip-Flops (cont'd)

- **D = 0, Clk High**

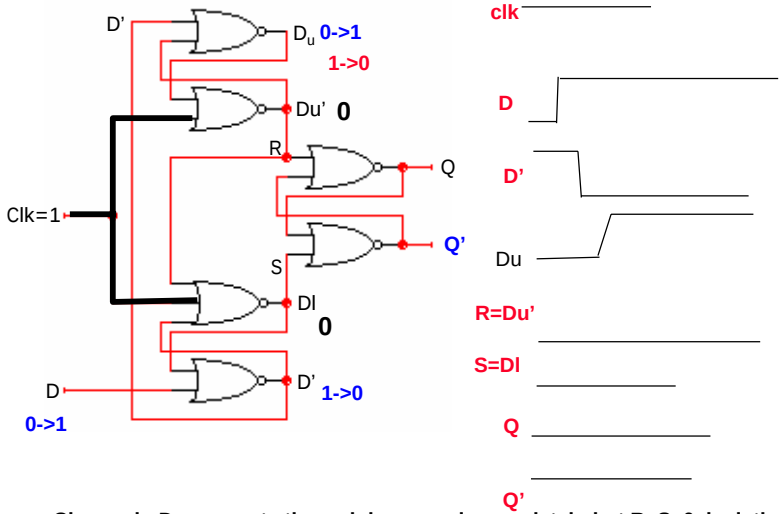


Edge-Triggered Flip-Flops (cont'd)

- **D = 1, Clk High**

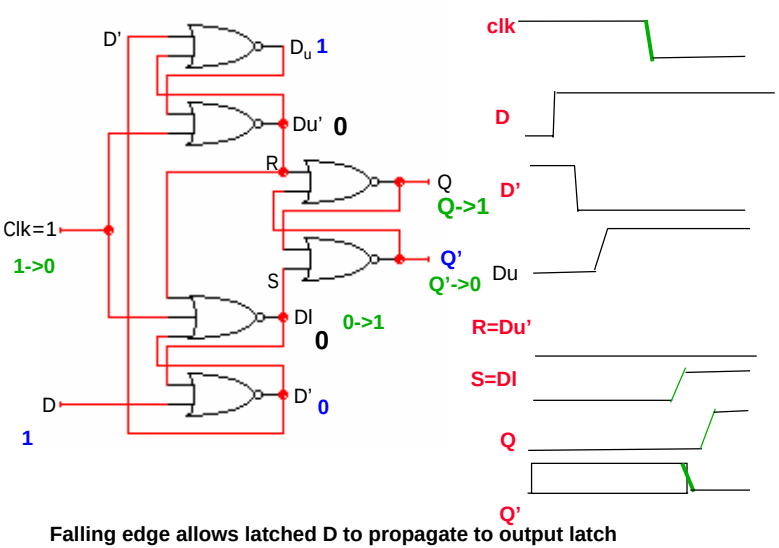


D-FF Behavior when CLK=1



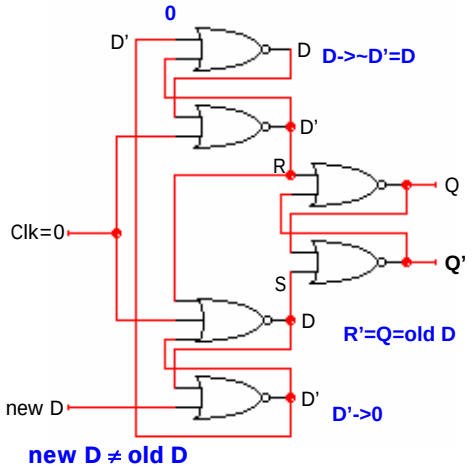
Change in D propagate through lower and upper latch, but R=S=0, isolating slave

Behavior when CLK 1->0



Falling edge allows latched D to propagate to output latch

D-FF; behavior when CLK==0

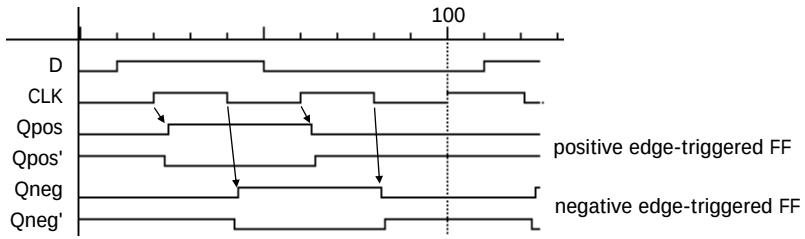


- **Lower output 0**
- **Upper latch retains old D**
- **RS unchanged**

when clock is low
data is held

Edge-Triggered Flip-Flops (cont'd)

- **Positive edge-triggered**
  - **Inputs sampled on rising edge; outputs change after rising edge**
- **Negative edge-triggered flip-flops**
  - **Inputs sampled on falling edge; outputs change after falling edge**

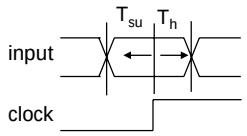


Timing Methodologies

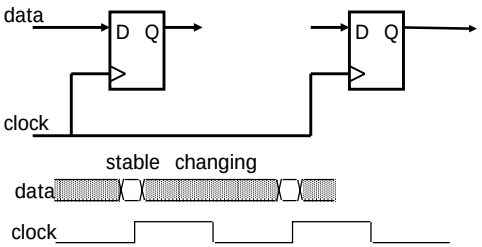
- **Rules for interconnecting components and clocks**
  - **Guarantee proper operation of system when strictly followed**
- **Approach depends on building blocks used for memory elements**
  - **Focus on systems with edge-triggered flip-flops**
    - » **Found in programmable logic devices**
  - **Many custom integrated circuits focus on level-sensitive latches**
- **Basic rules for correct timing:**
  - **(1) Correct inputs, with respect to time, are provided to the flip-flops**
  - **(2) No flip-flop changes state more than once per clocking event**

Timing Methodologies (cont'd)

- **Definition of terms**
  - **clock: periodic event, causes state of memory element to change; can be rising or falling edge, or high or low level**
  - **setup time: minimum time before the clocking event by which the input must be stable (Tsu)**
  - **hold time: minimum time after the clocking event until which the input must remain stable (Th)**



there is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized

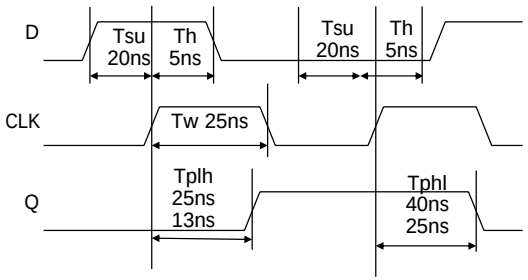


Comparison of Latches and Flip-Flops (cont'd)

Type	When inputs are sampled	When output is valid
unclocked latch	always	propagation delay from input change
level-sensitive latch	clock high (Tsu/Th around falling edge of clock)	propagation delay from input change or clock edge (whichever is later)
master-slave flip-flop	clock high (Tsu/Th around falling edge of clock)	propagation delay from falling edge of clock
negative edge-triggered flip-flop	clock hi-to-lo transition (Tsu/Th around falling edge of clock)	propagation delay from falling edge of clock

Typical Timing Specifications

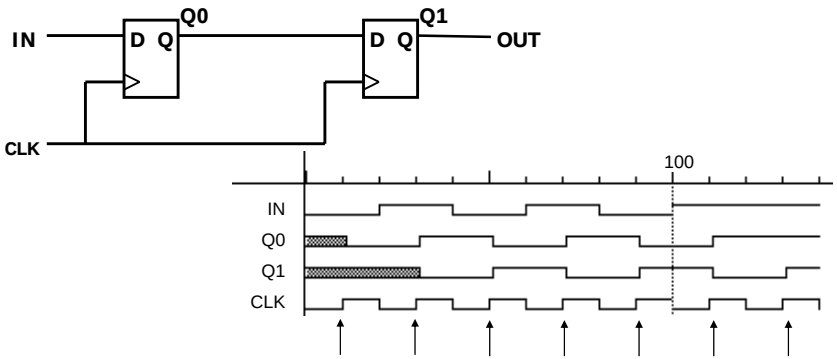
- **Positive edge-triggered D flip-flop**
  - **Setup and hold times**
  - **Minimum clock width**
  - **Propagation delays (low to high, high to low, max and typical)**



all measurements are made from the clocking event that is, the rising edge of the clock

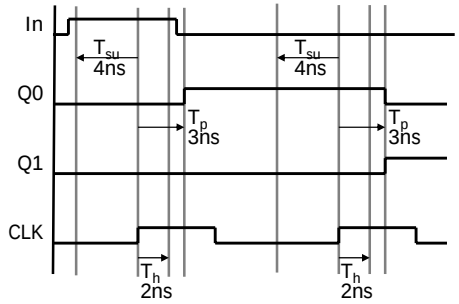
Cascading Edge-triggered Flip-Flops

- **Shift register**
  - **New value goes into first stage**
  - **While previous value of first stage goes into second stage**
  - **Consider setup/hold/propagation delays (prop must be > hold)**



Cascading Edge-triggered Flip-Flops (cont'd)

- **Why this works**
  - **Propagation delays exceed hold times**
  - **Clock width constraint exceeds setup time**
  - **This guarantees following stage will latch current value before it changes to new value**

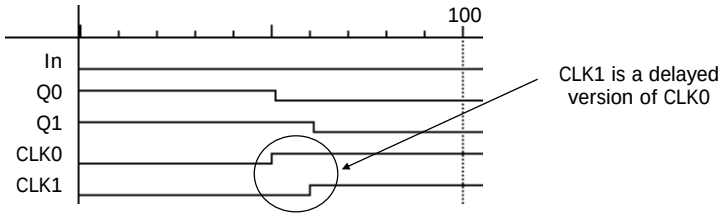


timing constraints guarantee proper operation of cascaded components

assumes infinitely fast distribution of the clock

Clock Skew

- **The problem**
  - **Correct behavior assumes next state of all storage elements determined by all storage elements at the same time**
  - **This is difficult in high-performance systems because time for clock to arrive at flip-flop is comparable to delays through logic**
  - **Effect of skew on cascaded flip-flops:**



original state: IN = 0, Q0 = 1, Q1 = 1
due to skew, next state becomes: Q0 = 0, Q1 = 0, and not Q0 = 0, Q1 = 1

Need Propagation – Skew > Hold Time

Summary of Latches and Flip-Flops

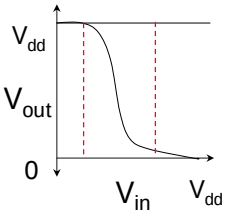
- **Development of D-FF**
  - **Level-sensitive used in custom integrated circuits**
    - » **can be made with 4 pairs of gates**
    - » **Usually follows multiphase non-overlapping clock discipline**
  - **Edge-triggered used in programmable logic devices**
  - **Good choice for data storage register**
- **Historically J-K FF was popular but now never used**
  - **Similar to R-S but with 1-1 being used to toggle output (complement state)**
  - **Good in days of TTL/SSI (more complex input function: D = JQ' + K'Q**
  - **Not a good choice for PALs/PLAs as it requires 2 inputs**
  - **Can always be implemented using D-FF**
- **Preset and clear inputs are highly desirable on flip-flops**
  - **Used at start-up or to reset system to a known state**

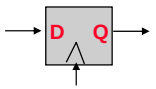
Flip-Flop Features

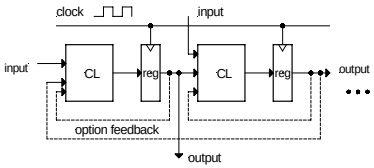
- **Reset (set state to 0) – R**
  - **Synchronous: Dnew = R' • Dold (when next clock edge arrives)**
  - **Asynchronous: doesn't wait for clock, quick but dangerous**
- **Preset or set (set state to 1) – S (or sometimes P)**
  - **Synchronous: Dnew = Dold + S (when next clock edge arrives)**
  - **Asynchronous: doesn't wait for clock, quick but dangerous**
- **Both reset and preset**
  - **Dnew = R' • Dold + S (set-dominant)**
  - **Dnew = R' • Dold + R'S (reset-dominant)**
- **Selective input capability (input enable/load) – LD or EN**
  - **Multiplexer at input: Dnew = LD' • Q + LD • Dold**
  - **Load may/may not override reset/set (usually R/S have priority)**
- **Complementary outputs – Q and Q'**

Maintaining the Digital Abstraction (in an analog world)

- Circuit design with very sharp transitions
- Noise margin for logical values
- Carefully Design Storage Elements (SE)
  - Internal feedback
- Structured System Design
  - SE + CL, cycles must cross SE
- Timing Methodology
  - All SE advance state together
  - All inputs stable across state change

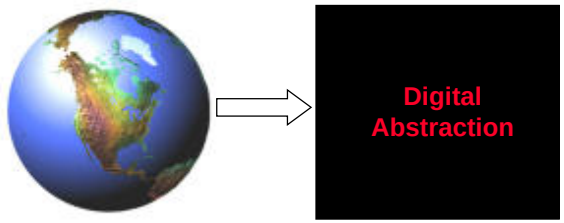


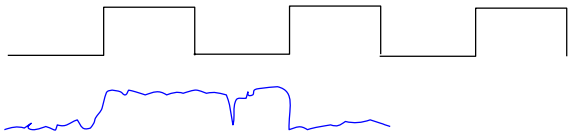




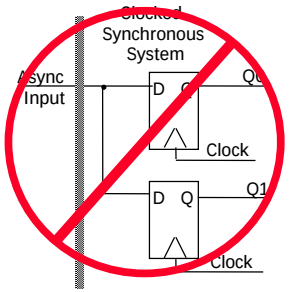
Where does this breakdown?

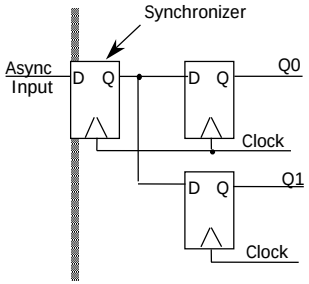
- Interfacing to the physical world
- Can't tell it "not to change near the clock edge"

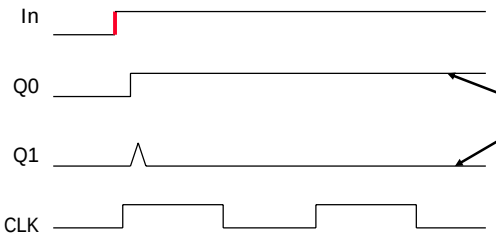




Example Problems







In is asynchronous and fans out to D0 and D1

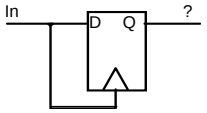
one FF catches the signal, one does not

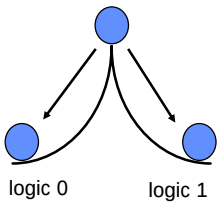
inconsistent state may be reached!

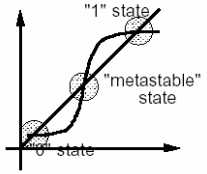
Metastability

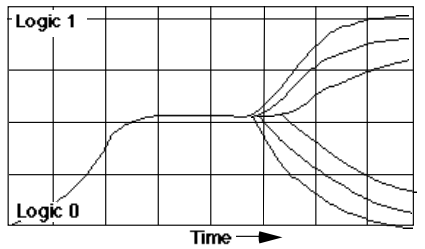
- In worst cast, cannot bound time for FF to decide if inputs can change right on the edge
  - Circuit has a metastable balance point

horrible example

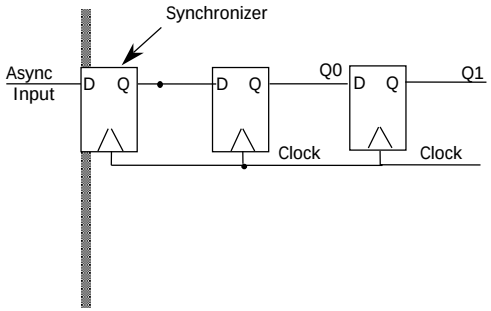








Practical Solution



- Series of synchronizers
  - each reduces the chance of getting stuck (exponentially)
- Make P(metastability) < P(device failure)
- Oversample and then low pass

Metastability throughout the ages

Buridan, Jean (1300-58), French Scholastic philosopher, who held a theory of determinism, contending that the will must choose the greater good. Born in Bethune, he was educated at the University of Paris, where he studied with the English Scholastic philosopher William of Ocham. After his studies were completed, he was appointed professor of philosophy, and later rector, at the same university. Buridan is traditionally but probably incorrectly associated with a philosophical dilemma of moral choice called "Buridan's ass." **In the problem an ass starves to death between two alluring bundles of hay because it does not have the will to decide which one to eat.**