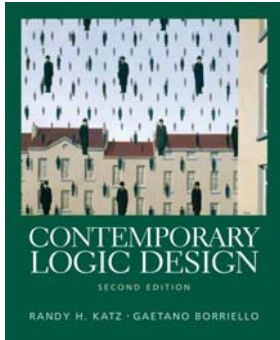


CSE 370 Spring 2006

Introduction to Digital Design

Lecture 14: Latches and Flip-Flops



Last Lecture

- Intro to Sequential Logic

Today

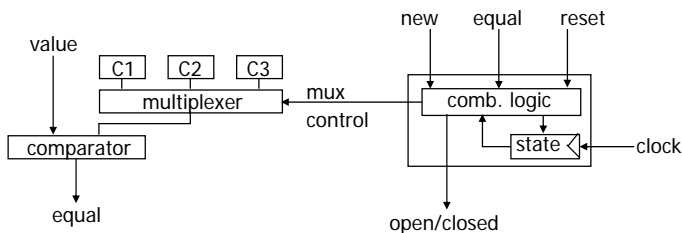
- Latches
- Flip-flops
- Timing Methodology

Administrivia

HW due Friday.

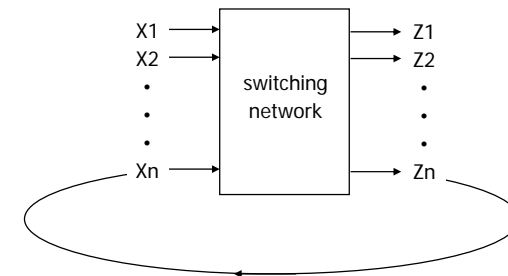
Sequential circuits

- Circuits with feedback
 - outputs = $f(\text{inputs, past inputs, past outputs})$
 - basis for building "memory" into logic circuits
 - door combination lock is an example of a sequential circuit
 - state is memory
 - state is an "output" and an "input" to combinational logic
 - combination storage elements are also memory



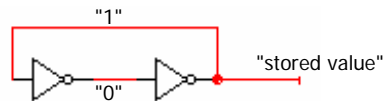
Circuits with feedback

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

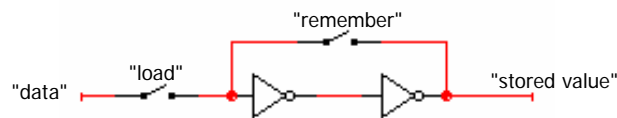


Simplest circuits with feedback

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

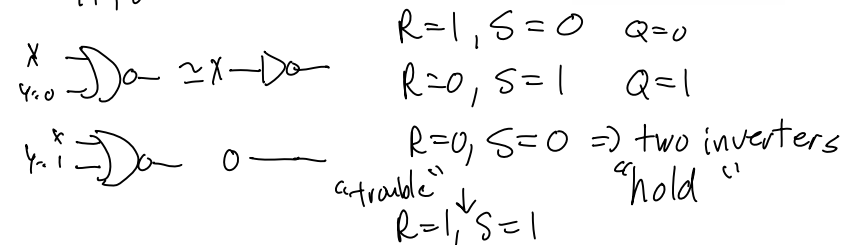
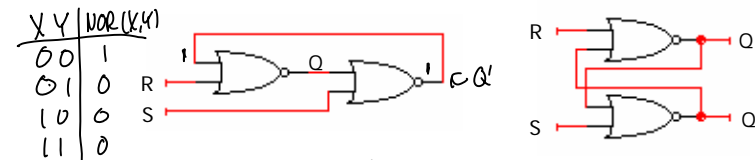


- How to get a new value into the memory cell?
 - selectively break feedback path
 - load new value into cell



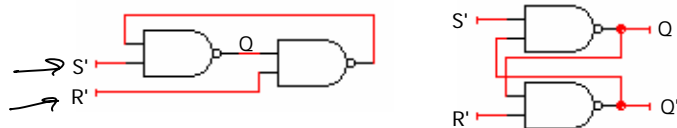
Memory with cross-coupled gates

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

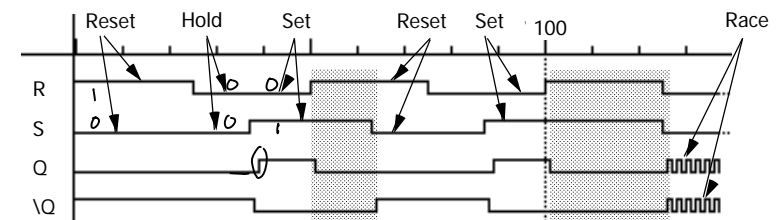
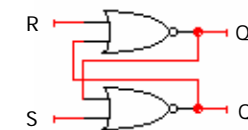


Memory with cross-coupled gates

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



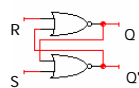
Timing behavior



State behavior or R-S latch

Truth table of R-S latch behavior

S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable

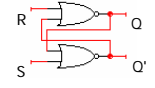
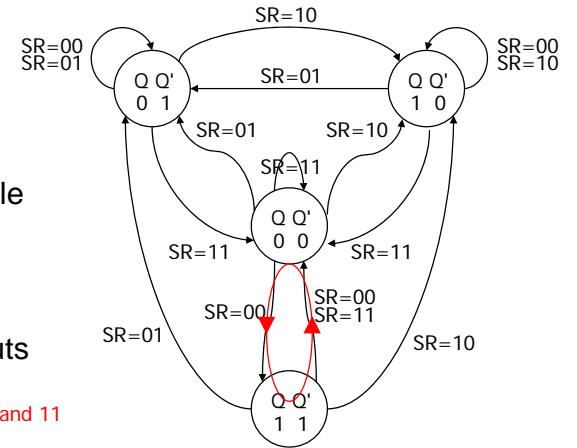


Theoretical R-S latch behavior

State diagram

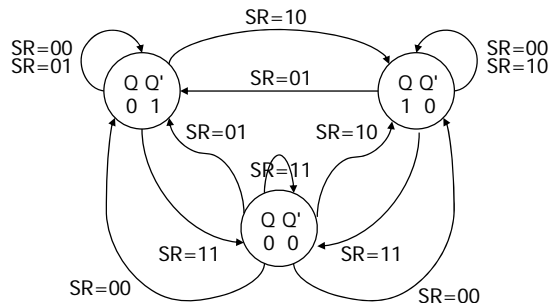
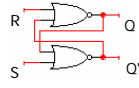
- states: possible values
- transitions: changes based on inputs

possible oscillation between states 00 and 11



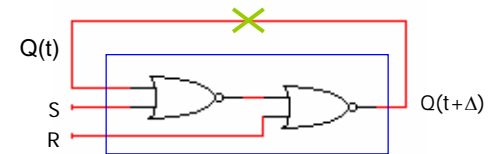
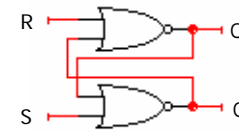
Observed R-S latch behavior

- 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

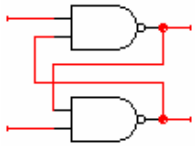


S	R	Q(t)	Q(t+Δ)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

		S	
		0	1
Q(t)	0	0	X
	1	0	X

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

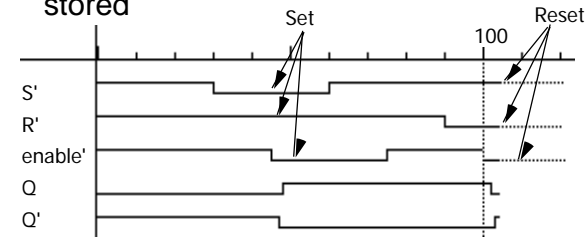
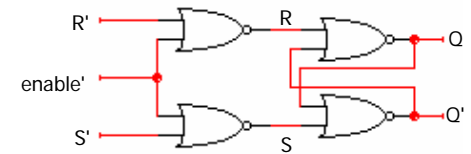
Activity: R-S latch using NAND gates



Gated R-S latch

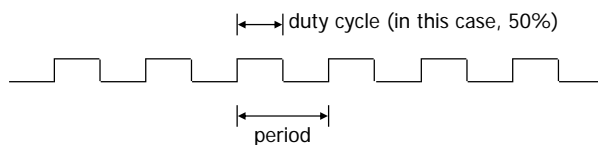
- Control when R and S inputs matter

- otherwise, the slightest glitch on R or S while enable is low could cause change in value stored



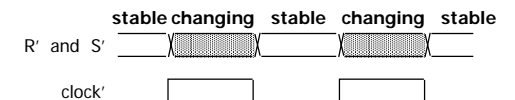
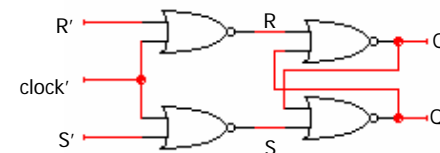
Clocks

- 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)



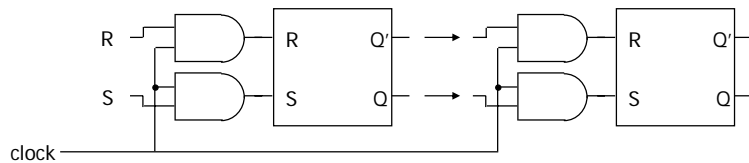
Clocks (cont'd)

- 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



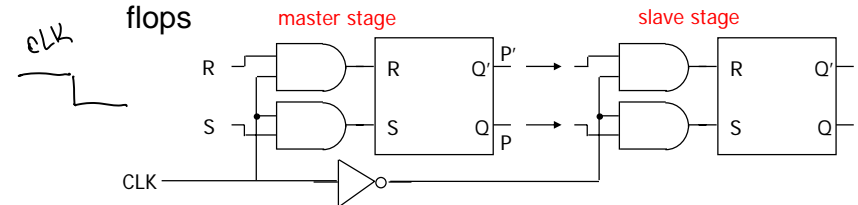
Cascading latches

- Connect output of one latch to input of another
- How to stop changes from racing through chain?
 - need to be able to control flow of data from one latch to the next
 - move one latch per clock period
 - have to worry about logic between latches (arrows) that is too fast



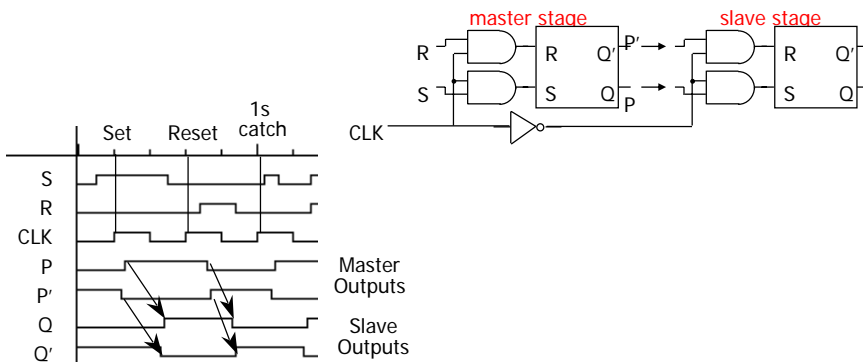
Master-slave structure

- 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



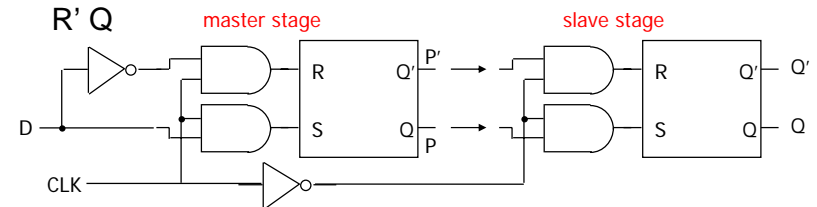
The 1s catching problem

- In first R-S stage of master-slave FF
 - 0-1-0 glitch on R or S while clock is high is "caught" by master stage
 - leads to constraints on logic to be hazard-free



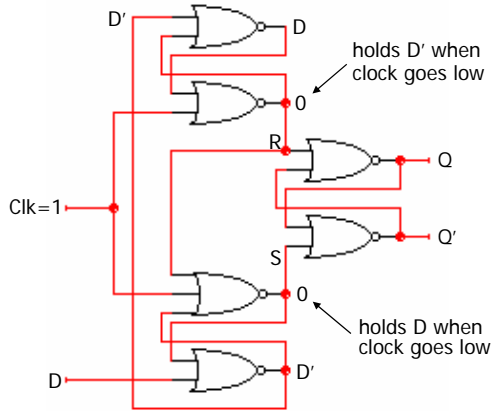
D flip-flop

- Make S and R complements of each other
 - eliminates 1s catching problem
 - 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 $D = S + R'Q$



Edge-triggered flip-flops

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

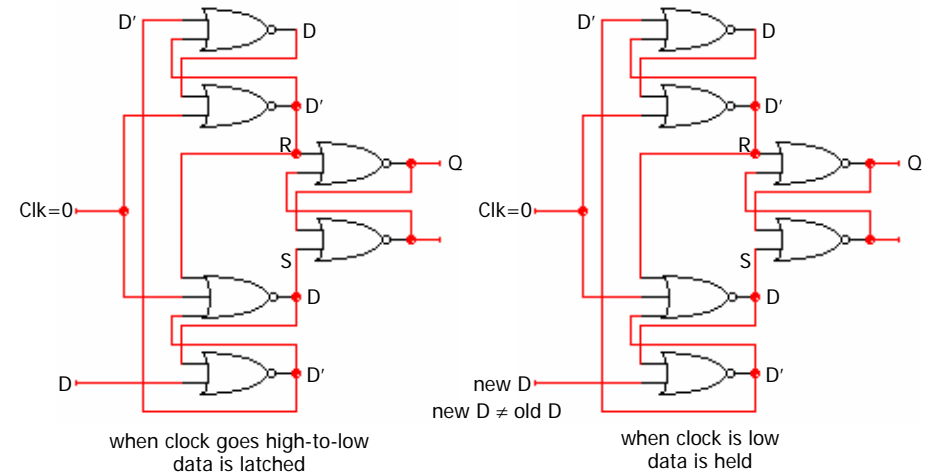


negative edge-triggered D flip-flop (D-FF)
 4-5 gate delays
 must respect setup and hold time constraints to successfully capture input

characteristic equation
 $Q(t+1) = D$

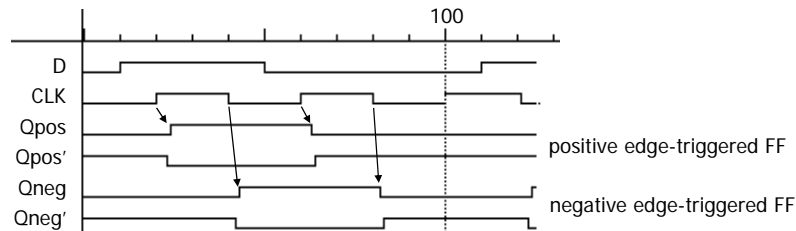
Edge-triggered flip-flops (cont'd)

- Step-by-step analysis



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



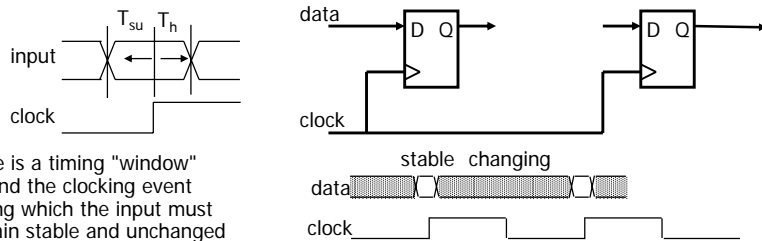
Timing methodologies

- Rules for interconnecting components and clocks
 - guarantee proper operation of system when strictly followed
- Approach depends on building blocks used for memory elements
 - we'll 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)

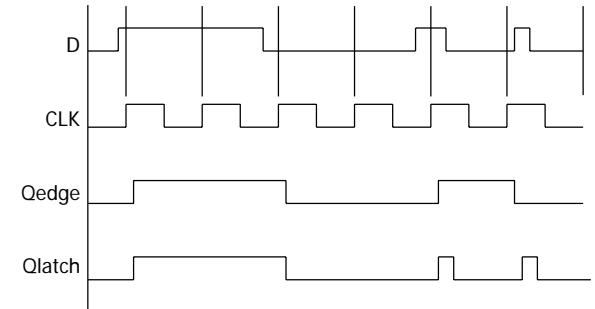
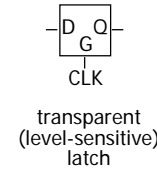
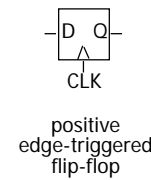
Definition of terms

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



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



behavior is the same unless input changes while the clock is high

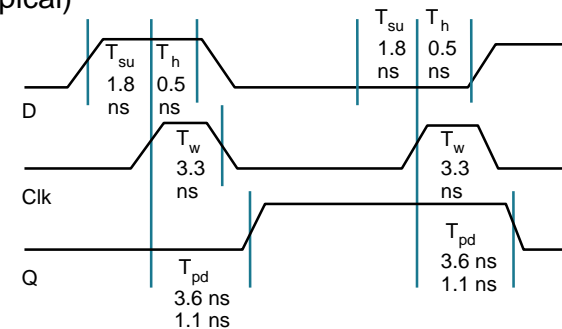
Comparison of latches and flip-flops (cont'd)

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

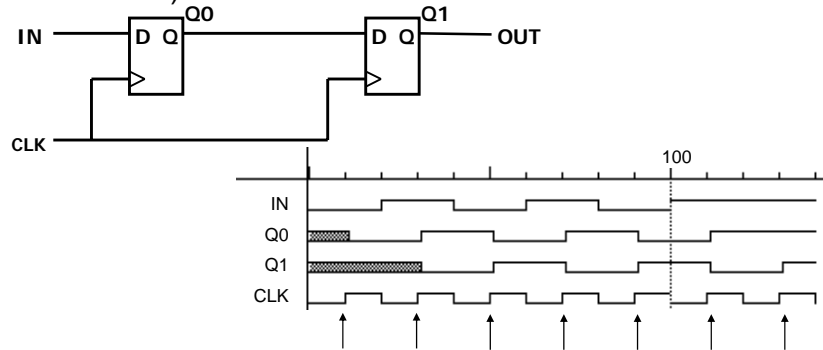


all measurements are made from the clocking event (the rising edge of the clock)

Cascading edge-triggered flip-flops

■ Shift register

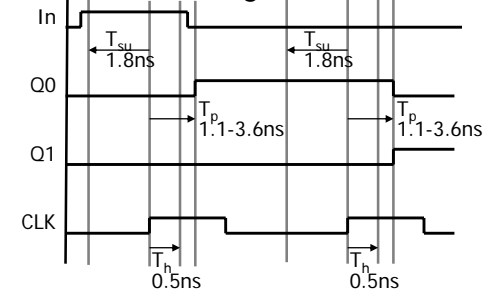
- new value goes into first stage
- while previous value of first stage goes into second stage
- consider setup/hold/propagation delays (prop must be > hold)



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