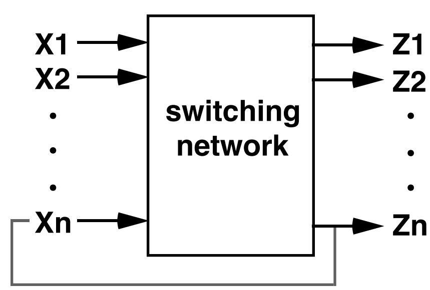
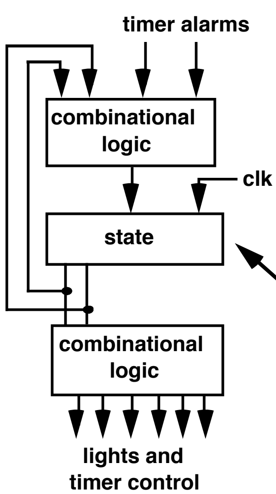
Chapter overview

- Sequential circuits
  - simple circuits with feedback
  - R-S latch
  - J-K master-slave
  - edge-triggered flip-flops
- Timing methodologies
  - cascading flip-flops for proper operation
  - clock skew
- Realizing circuits with flip-flops
  - choosing a FF type
  - characteristic equations
  - conversion among types
- Metastability and asynchronous inputs

Sequential circuits

- Circuits with feedback
  - some outputs are also inputs





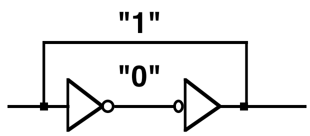
traffic light controller is an example of a sequential logic network

sequential logic forms basis for building "memory" into circuits

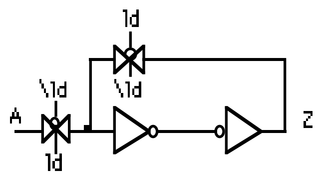
these memory elements are primitive sequential circuits

Sequential circuits

- Simple circuits with feedback
  - primitive memory elements created from cascaded gates
  - simplest gate component: inverter
  - basis for commercial static RAM designs
  - cross-coupled NOR gates and NAND gates also possible



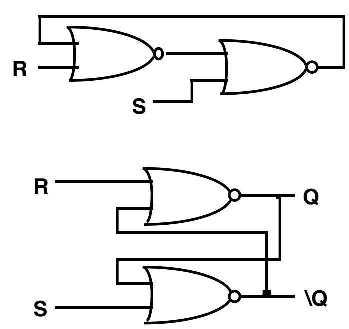
cascaded inverters
static memory cell



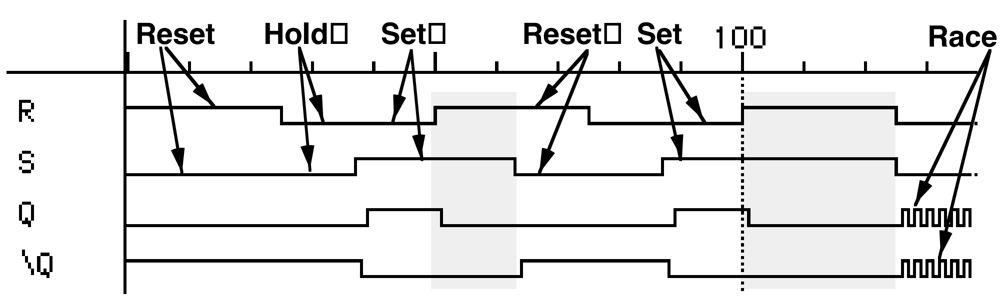
selectively break feedback path
to load new value into cell

Sequential circuits

- Cross-coupled NOR gates
  - just like cascaded inverters,
  - with capability to force output
  - to 0 (reset) or 1 (set)



- Timing waveform

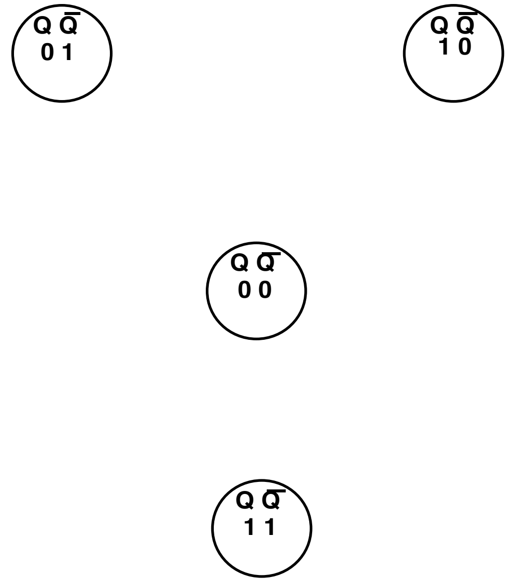


Sequential circuits

- State behavior of R-S latch

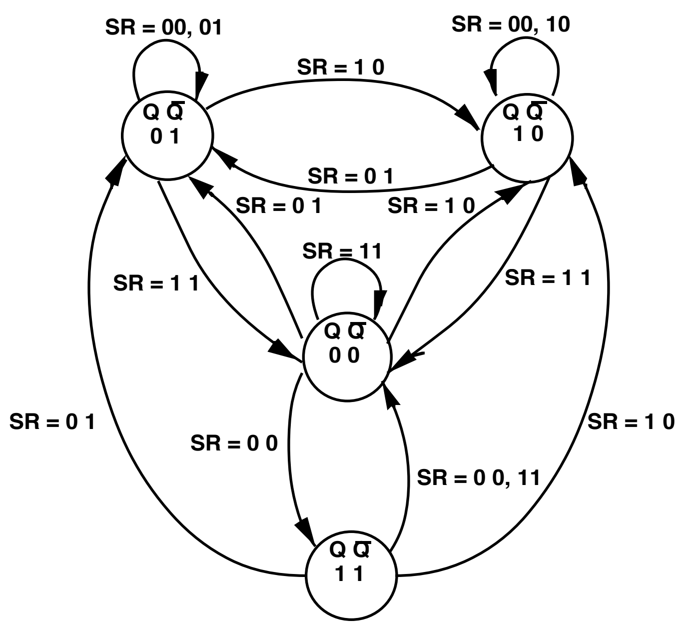
S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable

truth table summary
of R-S latch behavior



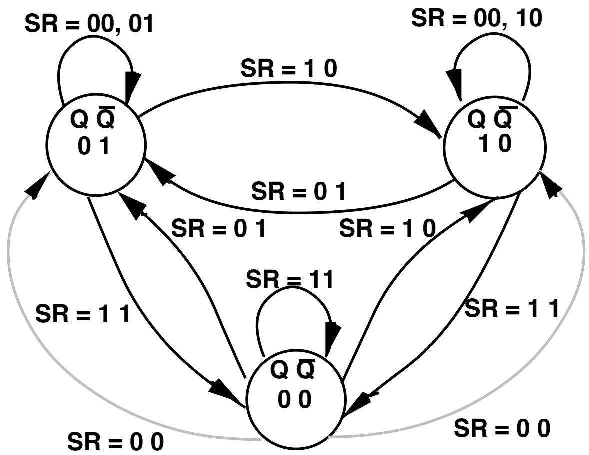
Sequential circuits

- Theoretical R-S latch state diagram



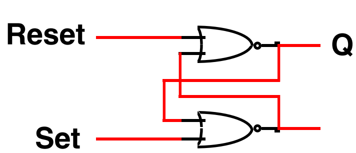
Sequential circuits

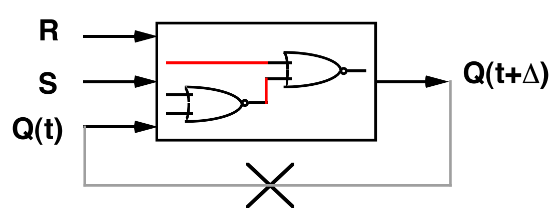
- Observed R-S latch behavior
  - very difficult to observe R-S latch in the 1-1 state
  - ambiguously returns to state 0-1 or 1-0 — a so-called "race condition"



Sequential circuits

- R-S latch analysis





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	

S

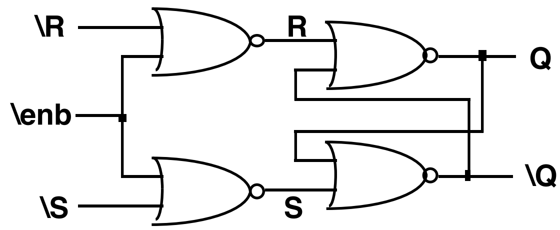
				S	S
		0	0	X	1
Q(t)		1	0	X	1

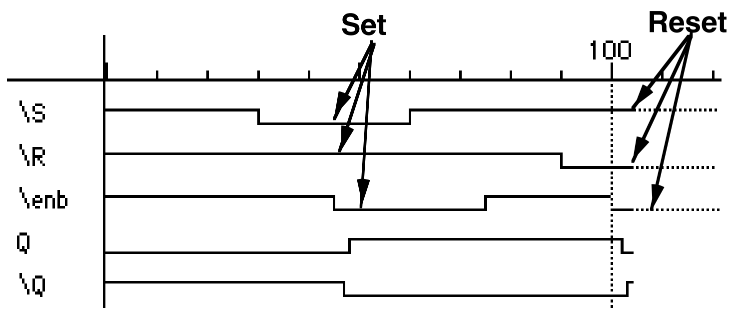
R

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

Sequential circuits

- Gated R-S latch



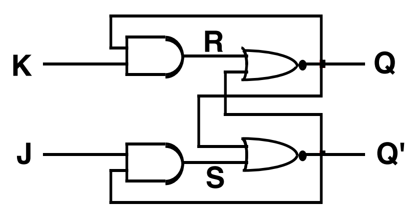


Sequential circuits

- J-K latch – eliminating the forbidden state

  - use output feedback to
  - guarantee that R and S
  - are never both one

- J, K both one yields toggle

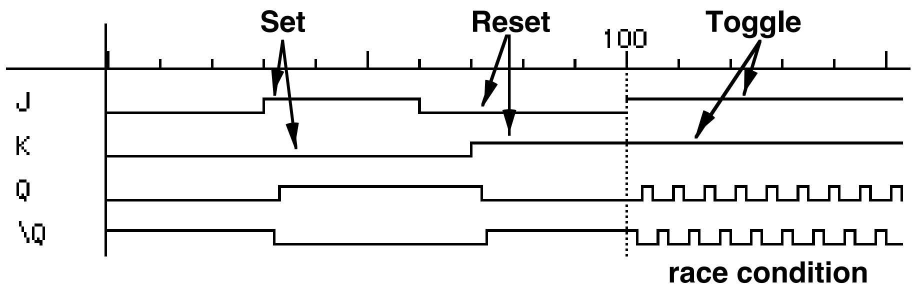


J	K	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

characteristic equation
$Q(t+\Delta) = K'Q(t) + JQ'(t)$

Sequential circuits

- J-K latch race condition

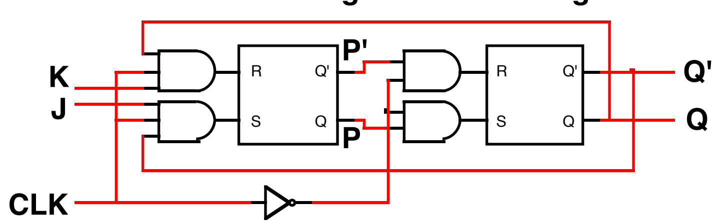


toggle correctness: single state change per clocking event

solution: master/slave flip-flop

Sequential circuits

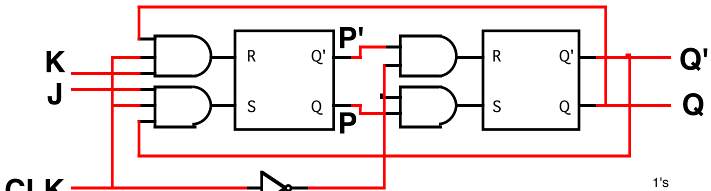
- Master-slave J-K flip-flop
  - use time to break feedback from outputs to inputs
  - sample inputs while clock is high
  - change outputs while clock is low

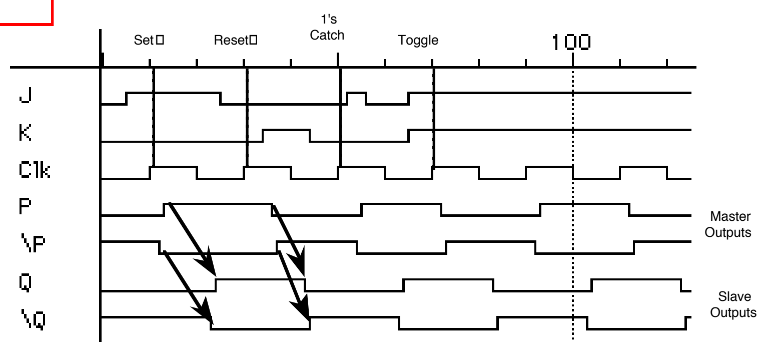


Q(t+1) = K'Q(t) + JQ'(t)

Sequential circuits

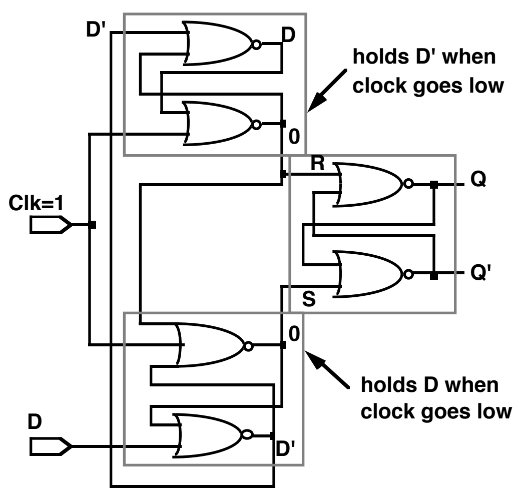
- Master-slave J-K flip-flop — 1s catching problem
  - a 0-1-0 glitch on J or K while clock is high is "caught" by master stage
  - leads to constraints on logic to be hazard-free





Sequential circuits

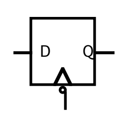
- Edge-triggered flip-flops
  - eliminates 1s catching problem
  - sensitive to inputs only near edge of enable signal



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

4-5 gate delays

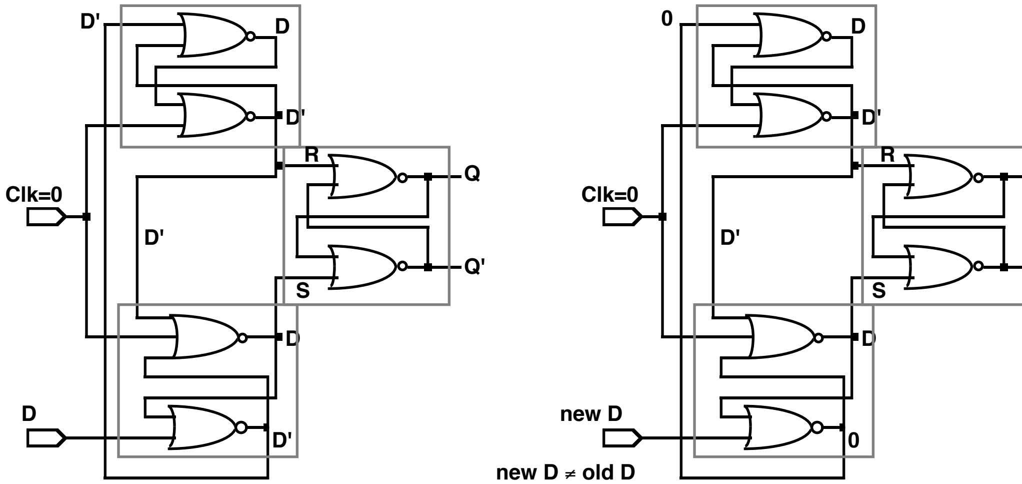
setup and hold times necessary to successfully latch the input



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

Sequential circuits

- Edge-triggered flip-flop
  - negative edge-triggered FF step-by-step analysis

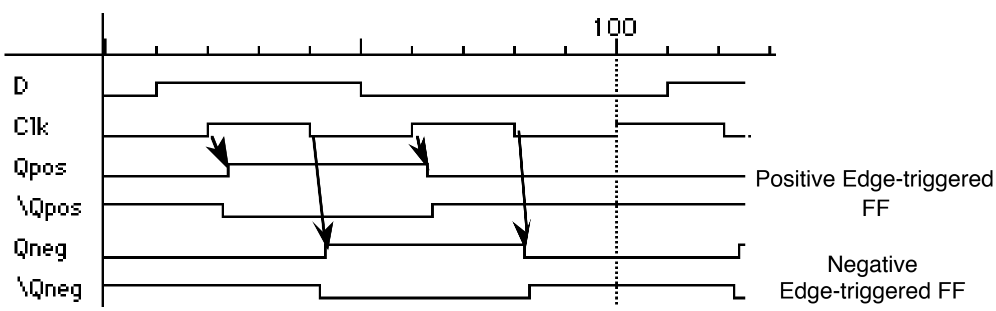


when clock goes high-to-low data is latched

when clock is low data is held

Sequential circuits

- 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

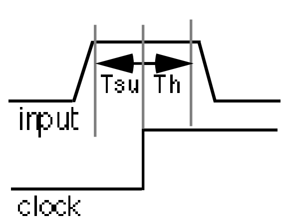


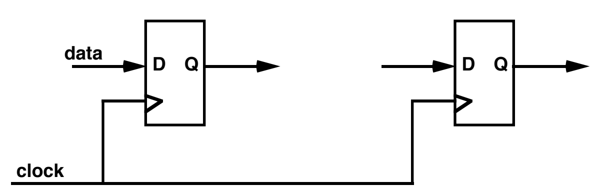
Timing methodologies

- Set of rules for interconnecting components and clocks
  - that when followed, guarantee proper operation of system
- Approach depends on building blocks used for memory elements
  - we'll focus on systems with edge-triggered flip-flops
  - also briefly consider level-senstive latches
- Basic rules for correct timing:
  - (1) correct inputs, with respect to time, are provided to the FFs
  - (2) no FF changes state more than once per clocking event

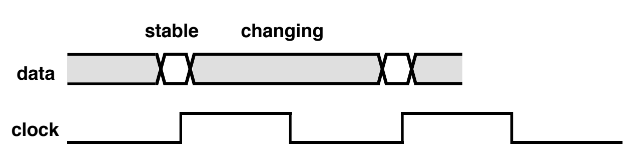
Timing methodologies

- 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 (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



Timing methodologies

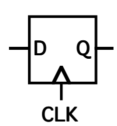
- **Comparison of latches and flip-flops**

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

Timing methodologies

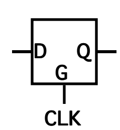
- **Comparison of latches and flip-flops**

7474

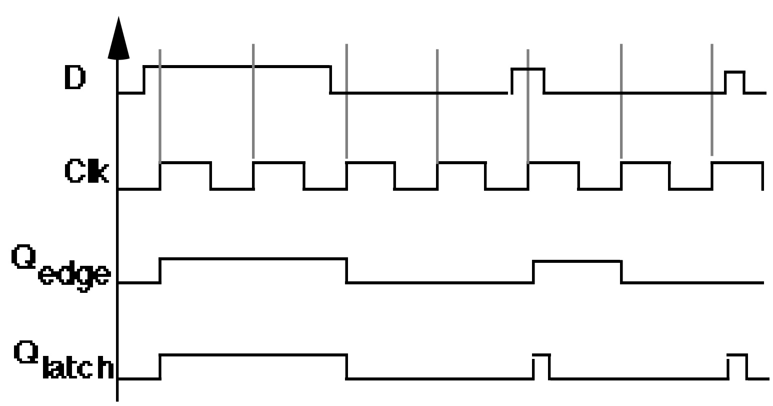


positive
edge-triggered
flip-flop

7475



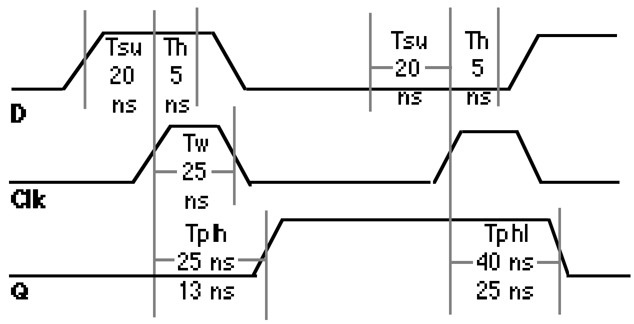
transparent
(level-sensitive)
latch



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

Timing methodologies

- Typical timing specifications: 74LS74 positive edge-triggered D flip-flop
  - setup and hold times
  - minimum clock width
  - propagation delays (low to high, high to low, max and typical)

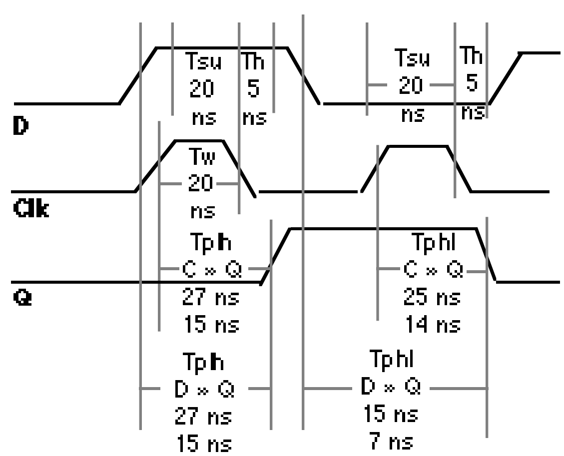


all measurements are made from the clocking event
that is, the rising edge of the clock

Timing methodologies

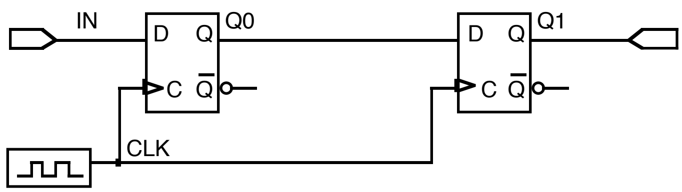
- Typical timing specifications: 74LS75 transparent latch
  - setup and hold times
  - minimum clock width
  - propagation delays: high to low, low to high, maximum, typical
    - must consider both data-to-output and clock-to-output
      -

measurements from
falling clock edge or
rising or falling data edge

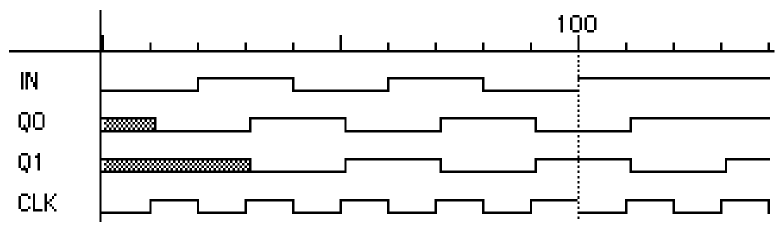


Timing methodologies

- Cascaded flip-flops and setup/hold/propagation delays



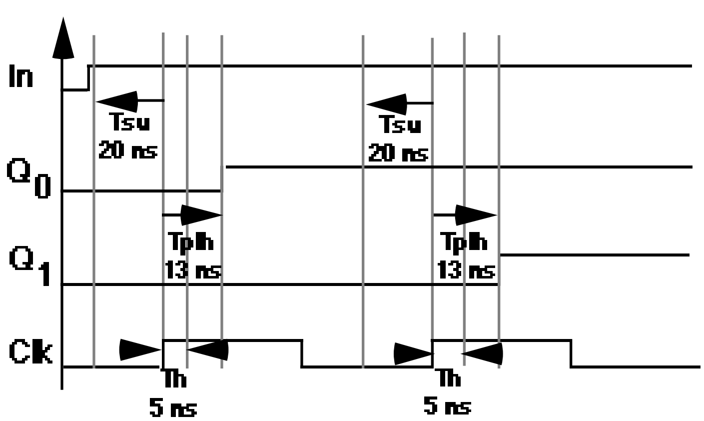
shift register:
new value to first stage while second stage obtains current value of first stage



Timing methodologies

- Cascaded flip-flops and setup/hold/propagation delays

  - why this works: propagation delays far exceed hold times
    clock width constraint exceeds setup time
  - this guarantees following stage will latch current value
    before it is replaced by new value



timing constraints guarantee proper operation of cascaded components

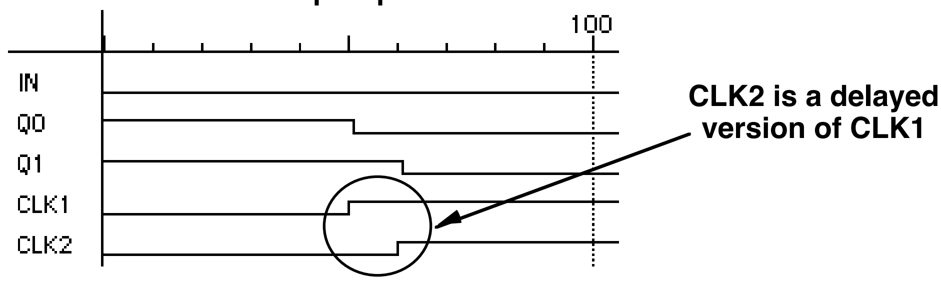
assumes infinitely fast distribution of the clock

Timing methodologies

- The problem of clock skew
  - correct behavior assumes next state of all storage elements
  - determined by all storage elements at <u>the same time</u>

  - this is not possible in real systems:
    - logical clock driven from more than one physical circuit
    - different wire delay to different points in the circuit

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

Timing methodologies

- Design strategies for minimizing clock skew

typical propagation delays for LS FFs: 13 ns (slow technology)
need skew of this order for problems to arise (usually not a problem)
nevertheless, the following are good design practices:

(1) distribute clock signals in general direction of data flow

(2) wire carrying the clock between two communicating components
- should be as short as possible

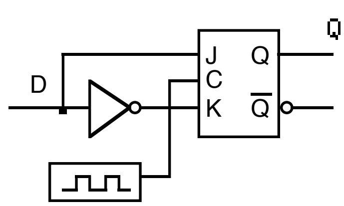
(3) try to make all wires from the clock generator be the same length

Summary of latches and flip-flops

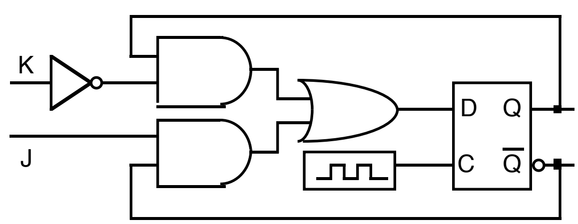
- Best choice is D-FF
  - simplest design technique, minimizes number of wires
  - preferred in PLDs, good choice for data storage register
- Historically J-K FF was popular
  - versatile building block, usually requires least amount of logic to
  - implement f(In,Q,Q+), but two inputs need more wiring
  - 1s catching problem – always use edge-triggered varieties
  - good in days of TTL/SSI, not a good choice for PLDs
  - can always be implemented using D-FF
- Level-sensitive latches in special circumstances
  - popular in VLSI because they can be made very small (4 transistors)
  - fundamental building block of all other flip-flop types
- Preset and clear inputs are highly desirable

Realizing circuits with different kinds of FFs

- Characteristic equations
  - R-S: $Q(t+1) = S + R' Q$
  - D: $Q(t+1) = D$
  - J-K: $Q(t+1) = J Q' + K' Q$
  - T: $Q(t+1) = T Q' + T' Q = T \text{ xor } Q$
- Implementing one FF in terms of another



D implemented with J-K



J-K implemented with D

Realizing circuits with different kinds of FFs

- **Excitation tables: what are the necessary inputs to cause a particular kind of change in state?**

Q(t)	Q(t+1)	R	S	J	K	T	D
0	0	X	0	0	X	0	0
0	1	0	1	1	X	1	1
1	0	1	0	X	1	1	0
1	1	0	X	X	0	0	1

Realizing circuits with different kinds of FFs

- **Implementing D FF with a J-K FF**

1) start with K-map of Q(t+1) = F(D, Q)
2) create K-maps for J and K with same inputs (D, Q)
3) fill in K-maps with appropriate values for J and K to cause the same state changes as in the original K-map

		D
	0	1
Q	0	1

Q(t+1) = D

		D
	0	1
Q	X	X

J = D

		D
	X	X
Q	1	0

K = D'

e.g., D = Q = 0, Q(t+1) = 0
then J = 0, K = X

Realizing circuits with different kinds of FFs

• Implementing J-K FF with a D FF:

□ 1) K-Map of Q(t+1) = F(J, K, Q)
□ 2) create K-map for D input as a function of J, K, and Q
□ 3) revised K-map using D's excitation table
□ □ its the same! that is why design procedure with D FF is simple!

			J	J
	0□	0□	1□	1
Q	1□	0□	0□	1
		K	K	

Q(t+1) = D = J Q' + K' Q

Of course, it is identical to the characteristic equation for a J-K FF

Metastability and asynchronous inputs

• Terms and definitions

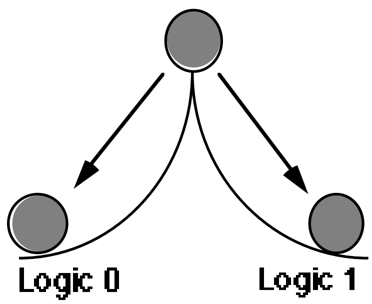
clocked synchronous circuits
□ inputs, state, and outputs sampled or changed in relation to a
□ □ common reference signal (called the clock)
□ (e.g., J-K master/slave)

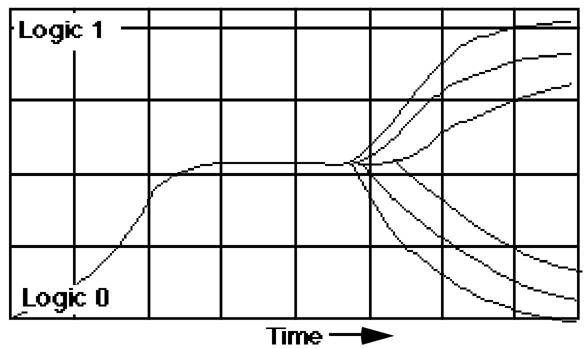
asynchronous circuits
□ inputs, state, and outputs sampled or changed independently of a
□ □ common reference signal (glitches/hazards a major concern)
□ (e.g., R-S latch)

• Asynchronous inputs to synchronous circuits
□ inputs can change at any time, will not meet setup/hold times
□ dangerous, synchronous inputs are greatly preferred
□ cannot be avoided (e.g., reset signal, memory wait, user input)

Metastability and asynchronous inputs

- Synchronizer failure: when FF input changes close to clock edge, the FF may enter a metastable state – neither a logic 0 nor 1 – it may stay in this state an indefinite amount of time, although this is not likely in real circuits



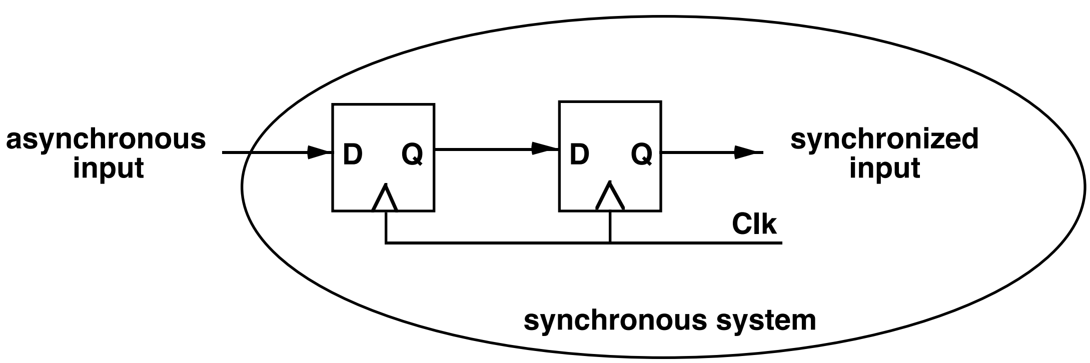


small, but non-zero probability that the FF output will get stuck in an in-between state

oscilloscope traces demonstrating synchronizer failure and eventual decay to steady state

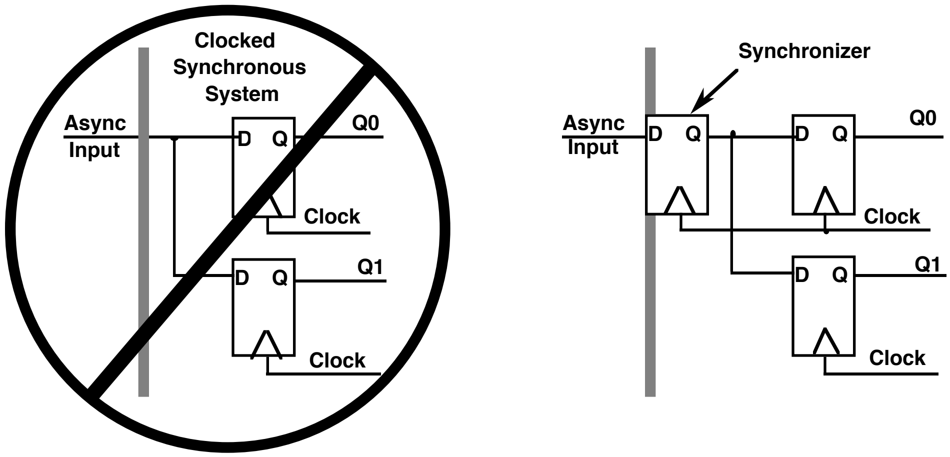
Metastability and asynchronous inputs

- Solutions to synchronizer failure
  - probability of failure can never be reduced to 0, but it can be reduced
  - (1) slow down the system clock this gives the synchronizer more time to decay into a steady state; synchronizer failure becomes a big problem for very high speed systems
  - (2) use fastest possible logic in the synchronizer this makes for a very sharp "peak" upon which to balance (S or AS TTL D-FFs are best)
  - (3) cascade two synchronizers



Metastability and asynchronous inputs

• Handling asynchronous inputs

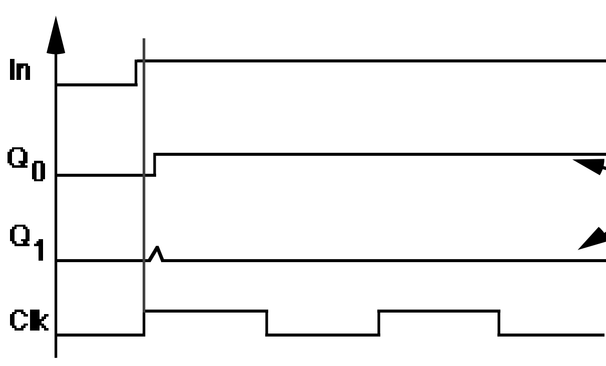


never allow asynchronous inputs to be fanned out to more than one FF within the synchronous system

Metastability and asynchronous inputs

• Handling asynchronous inputs (cont'd)

what can go wrong?



In is asynchronous and fans out to D0 and D1

one FF catches the signal, one does not

<u>impossible state might be reached!</u>

Chapter review

- Fundamental building block of circuits with state: latch and flipflop
  - R-S latch, J-K master/slave, edge-triggered D flip-flop
  - FF types, characteristic equations, conversion among types

- Timing methodologies:
  - cascaded FFs work because propagation delays exceed hold times
  - beware of clock skew

- Asynchronous inputs and their dangers
  - synchronizer failure: what it is and how to minimize its impact