Chapter #6: Sequential Logic Design

Contemporary Logic Design

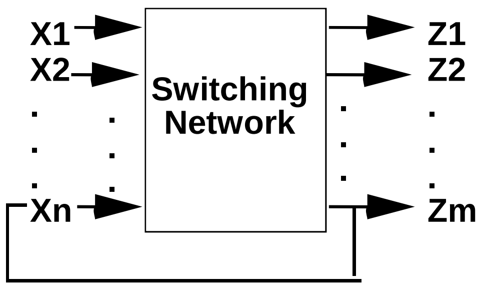
Randy H. Katz
University of California, Berkeley

June 1993

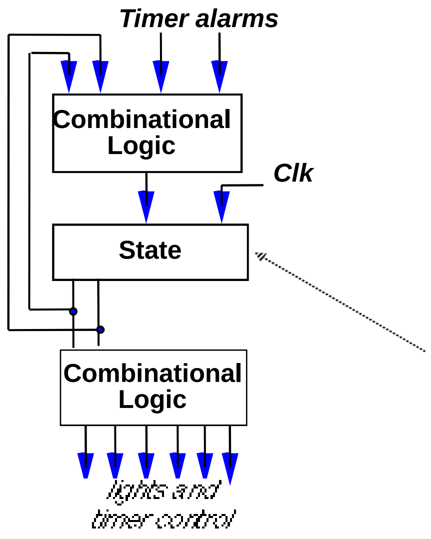
Chapter Overview

- *Sequential Networks*
  Simple Circuits with Feedback
  R-S Latch
  J-K Flipflop
  Edge-Triggered Flipflops
- *Timing Methodologies*
  Cascading Flipflops for Proper Operation
  Narrow Width Clocking vs. Multiphase Clocking
  Clock Skew
- *Realizing Circuits with Flipflops*
  Choosing a FF Type
  Characteristic Equations
  Conversion Among Types
- *Metastability and Asynchronous Inputs*
- *Self-Timed Circuits*

Sequential Switching Networks



Circuits with Feedback:
Some outputs are also inputs



Traffic Light Controller is a complex sequential logic network

Sequential logic forms basis for building "memory" into circuits

These memory elements are primitive sequential circuits

Sequential Switching Networks

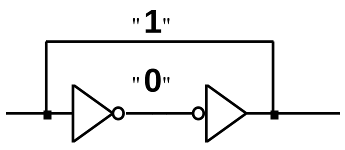
Simple Circuits with Feedback

Primitive memory elements created from cascaded gates

Simplest gate component: inverter

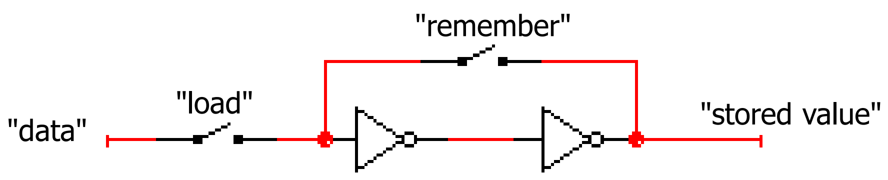
Basis for commercial static RAM designs

Cross-coupled NOR gates and NAND gates also possible



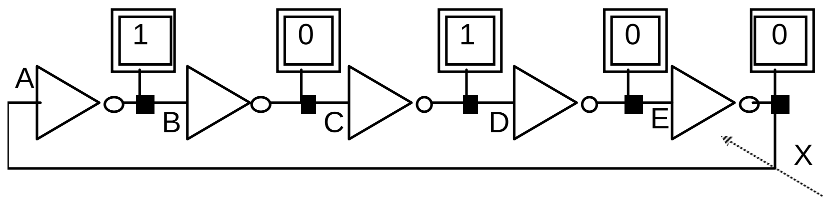
Cascaded Inverters: Static Memory Cell

Selectively break feedback path to load new value into cell



Sequential Switching Networks

Inverter Chains

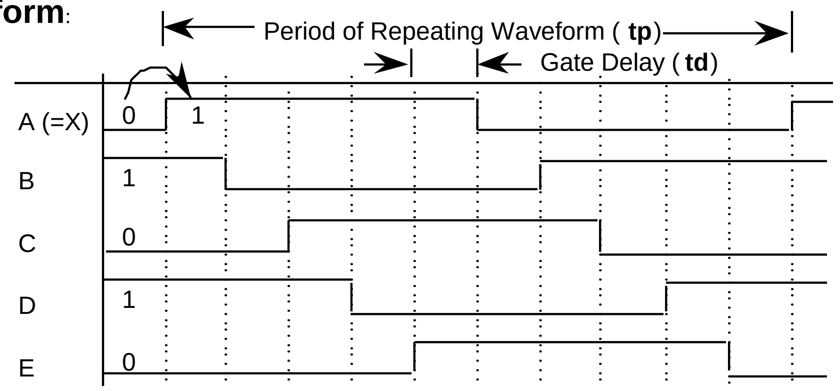


Odd # of stages leads to ring oscillator
Snapshot taken just before last inverter changes

Output high propagating thru this stage

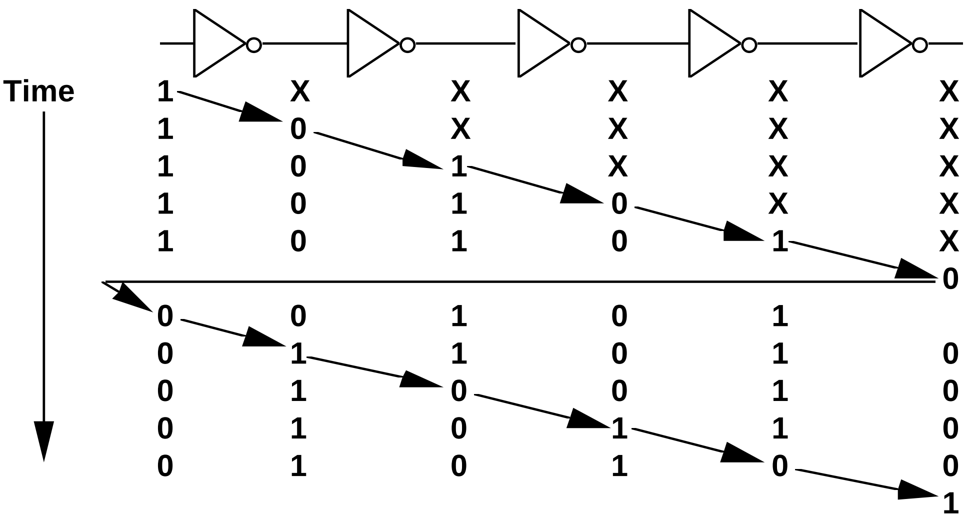
Timing Waveform:

tp = n * td
n = # inverters



Sequential Switching Networks

Inverter Chains

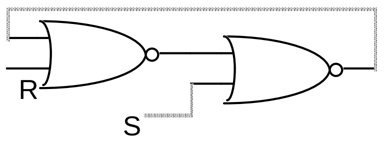


1	X	X	X	X	X
1	0	X	X	X	X
1	0	1	X	X	X
1	0	1	0	X	X
1	0	1	0	1	X
					0
0	0	1	0	1	
0	1	1	0	1	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	0	1	0	0
					1

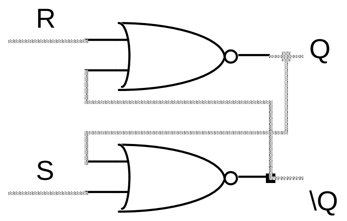
Propagation of Signals through the Inverter Chain

Sequential Switching Networks

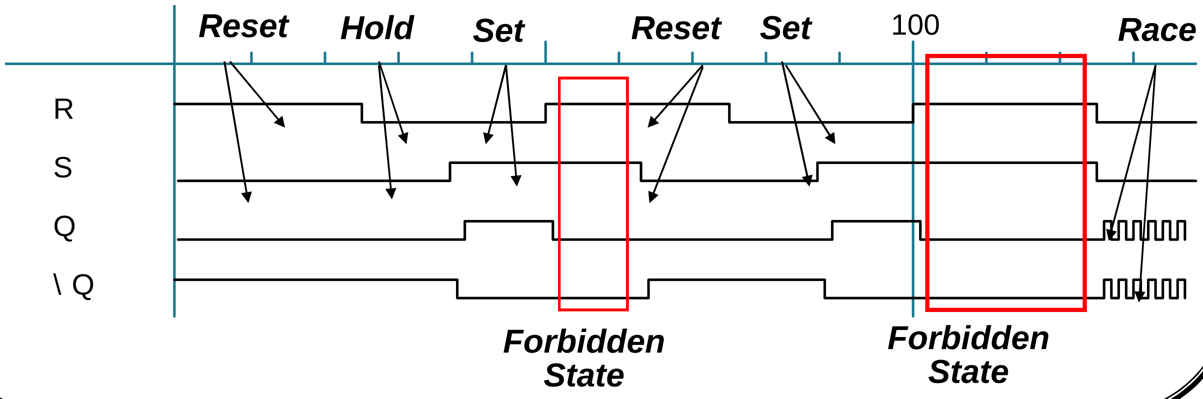
Cross-Coupled NOR Gates



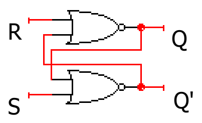
Just like cascaded inverters, with capability to force output to 0 (reset) or 1 (set)



Timing Waveform

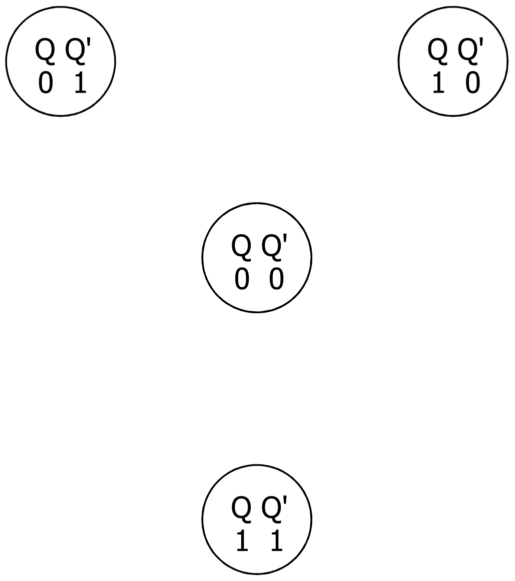


State behavior or R-S latch



- **Truth table of R-S latch behavior**

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

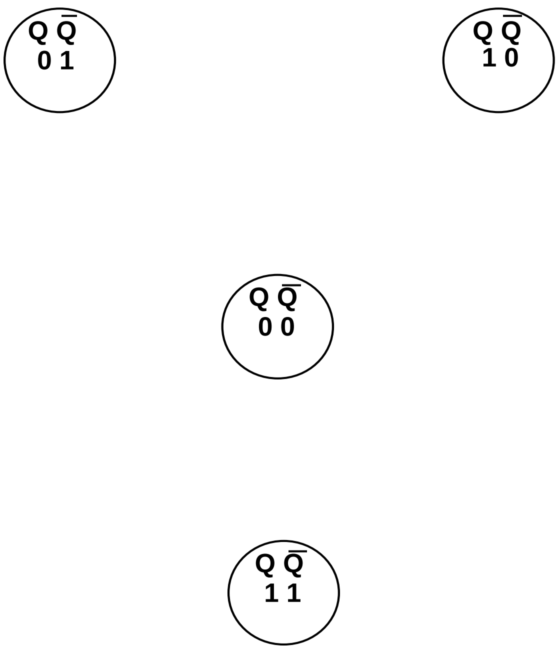


Sequential Switching Networks

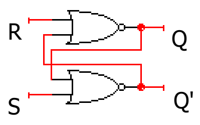
State Behavior of R-S Latch

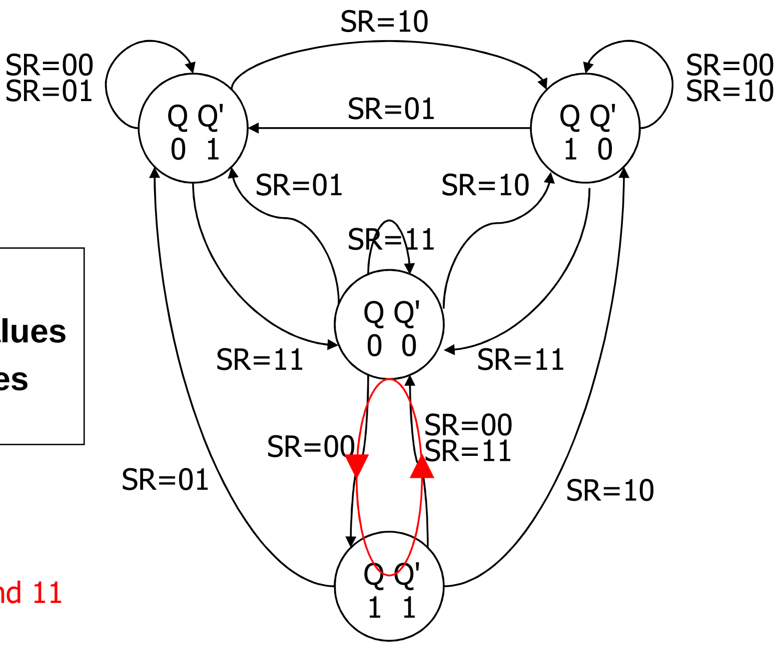
S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable

Truth Table Summary of R-S Latch Behavior



Theoretical R-S latch behavior



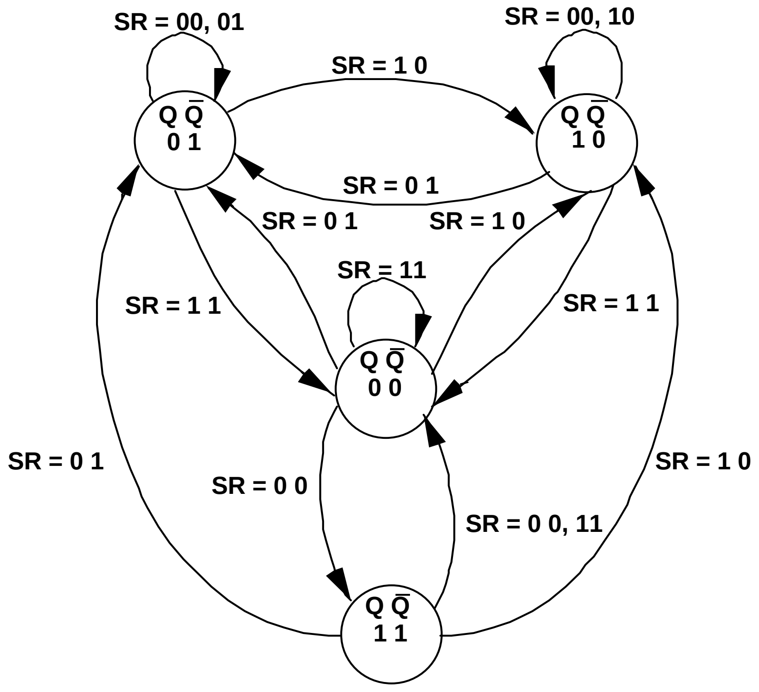


- **State diagram**
  - **states: possible values**
  - **transitions: changes based on inputs**

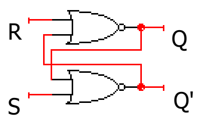
possible oscillation between states 00 and 11

Sequential Logic Networks

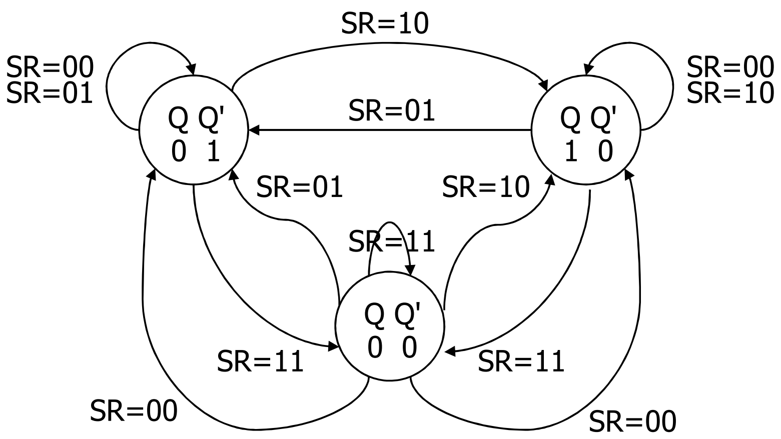
Theoretical R-S Latch State Diagram



Observed R-S latch behavior

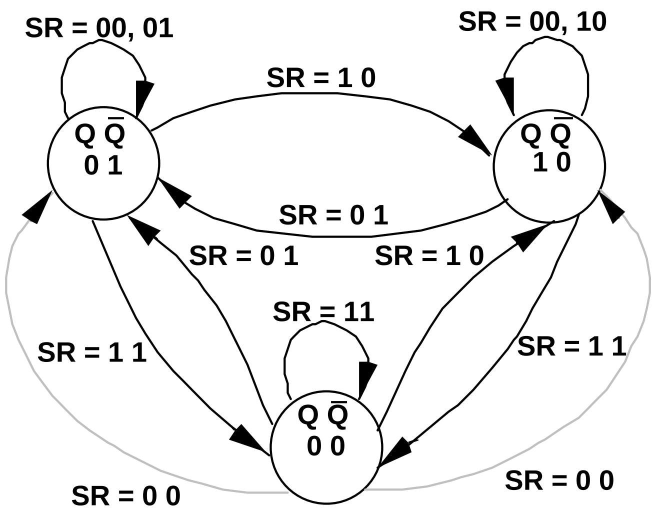


- **Very difficult to observe R-S latch in the 1-1 state**
  - **one of R or S usually changes first**
- **Ambiguously returns to state 0-1 or 1-0**
  - **a so-called "race condition"**
  - **or non-deterministic transition**



Sequential Logic Networks

Observed R-S Latch Behavior



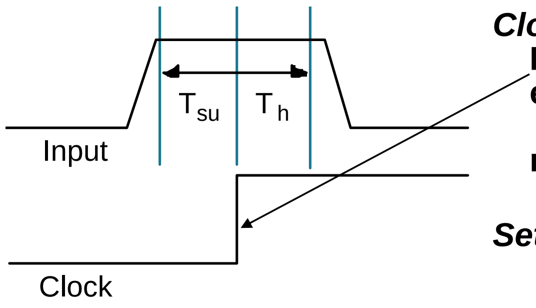
Very difficult to observe R-S Latch in the 1-1 state

Ambiguously returns to state 0-1 or 1-0

A so-called "race condition"

Sequential Switching Networks

Definition of Terms



Clock:
Periodic Event, causes state of memory element to change

rising *edge*, falling *edge*, high *level*, low *level*

Setup Time (Tsu)
Minimum time before the clocking event by which the input must be stable

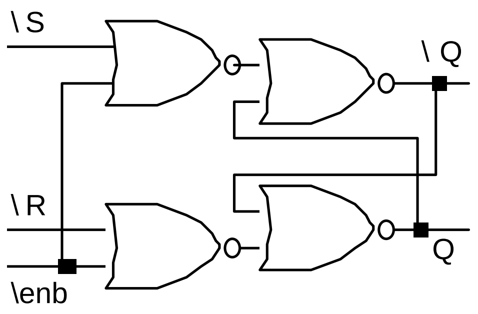
Hold Time (Th)
Minimum time after the clocking event during which the input must remain stable

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

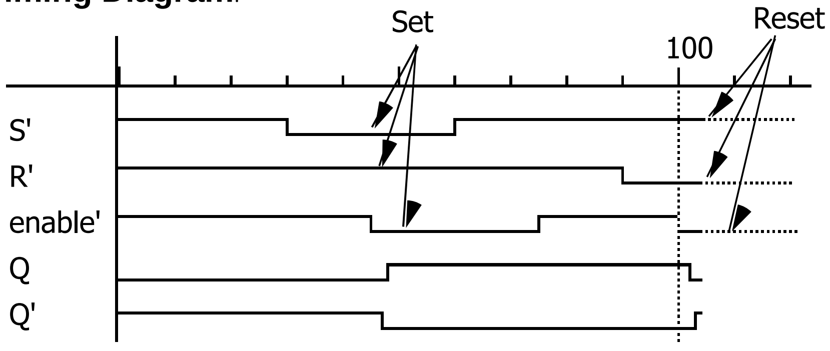
Sequential Switching Networks

Level-Sensitive Latch **aka Gated R-S Latch**

Schematic:

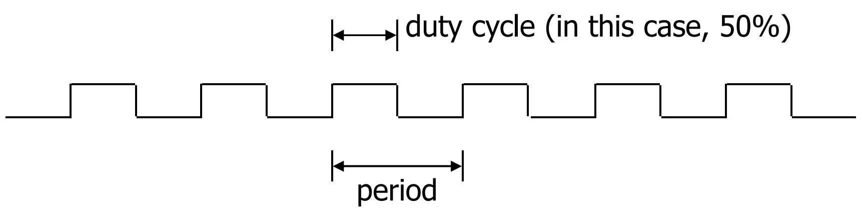


Timing Diagram:



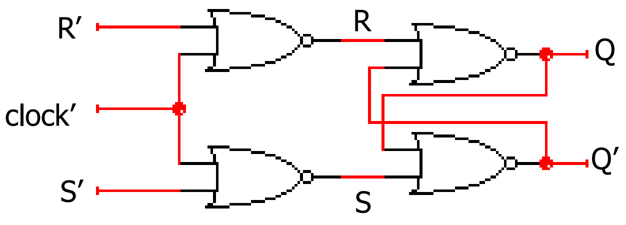
Clocks

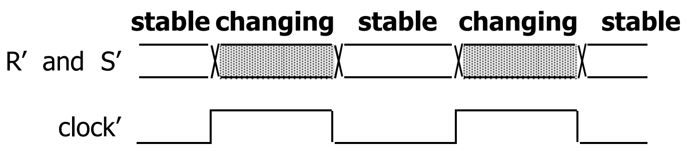
- **Used to keep time**
  - **wait long enough for inputs (R' and S') to settle**
  - **then allow to have effect on value stored**
- **Clocks are regular periodic signals**
  - **period (time between ticks)**
  - **duty-cycle (time clock is high between ticks - expressed as % of period)**



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





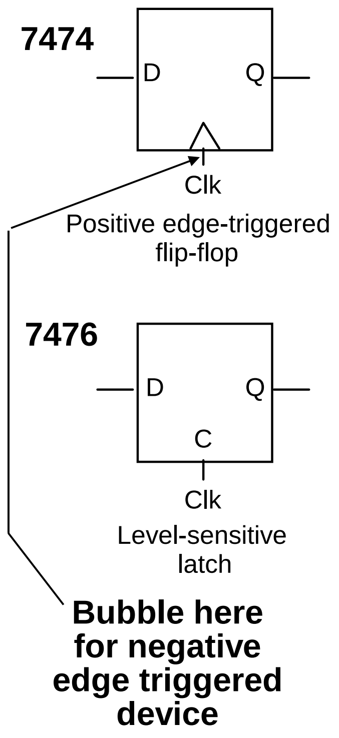
Sequential Switching Networks

Latches vs. Flipflops

Input/Output Behavior of Latches and Flipflops

Type	When Inputs are Sampled	When Outputs are Valid
unclocked latch	always	propagation delay from input change
level sensitive latch	clock high (Tsu, Th around falling clock edge)	propagation delay from input change
positive edge flipflop	clock lo-to-hi transition (Tsu, Th around rising clock edge)	propagation delay from rising edge of clock
negative edge flipflop	clock hi-to-lo transition (Tsu, Th around falling clock edge)	propagation delay from falling edge of clock
master/slave flipflop	clock hi-to-lo transition (Tsu, Th around falling clock edge)	propagation delay from falling edge of clock

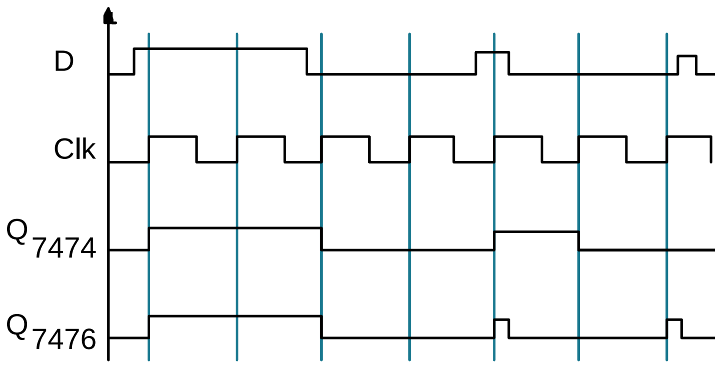
Sequential Switching Networks


7474
D
Q
Clk
Positive edge-triggered flip-flop
7476
D
Q
C
Clk
Level-sensitive latch
Bubble here for negative edge triggered device

Edge triggered **device sample inputs on the event edge**

Transparent latches **sample inputs as long as the clock is asserted**

Timing Diagram:



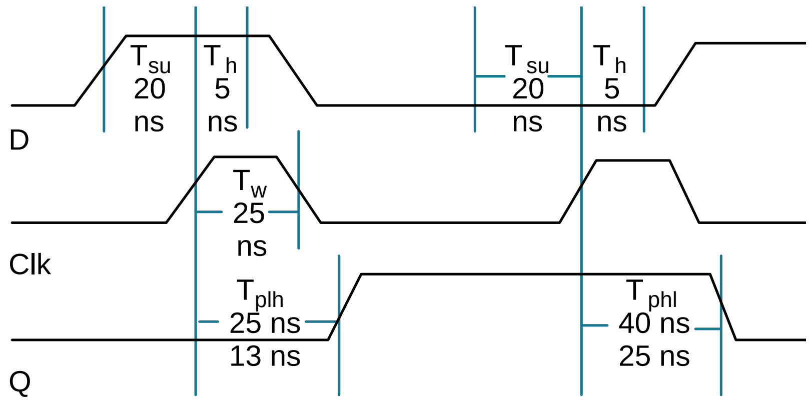
Behavior the same unless input changes while the clock is high

Sequential Switching Elements

Typical Timing Specifications: Flipflops vs. Latches

74LS74 Positive Edge Triggered D Flipflop

- **Setup time**
- **Hold time**
- **Minimum clock width**
- **Propagation delays (low to high, high to low, max and typical)**



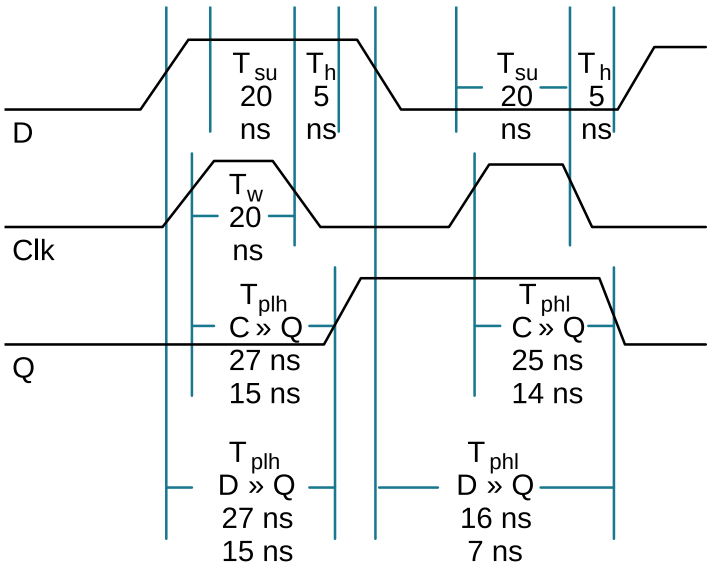
All measurements are made from the clocking event that is, the *rising edge* of the clock

Sequential Switching Networks

Typical Timing Specifications: Flipflops vs. Latches

74LS76 Transparent Latch

- **Setup time**
- **Hold time**
- **Minimum Clock Width**
- **Propagation Delays:**
  high to low, low to high,
  maximum, typical
  data to output
  clock to output


D
Clk
Q
T_{su} 20 ns
T_h 5 ns
T_{su} 20 ns
T_h 5 ns
T_w 20 ns
T_{plh} C » Q 27 ns 15 ns
T_{phl} C » Q 25 ns 14 ns
T_{plh} D » Q 27 ns 15 ns
T_{phl} D » Q 16 ns 7 ns

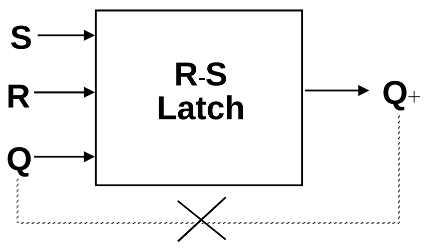
Measurements from falling clock edge or rising or falling data edge

Sequential Switching Elements

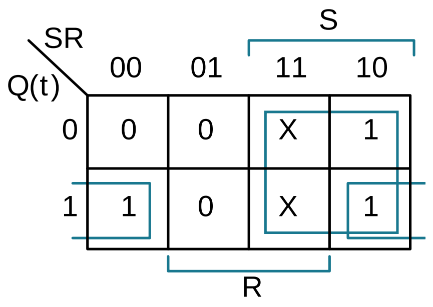
R-S Latch Revisited

Truth Table:
Next State = F(S, R, Current State)

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	



Derived K-Map:



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

Characteristic Equation:

$$Q^+ = S + \overline{R}\, Q_t$$

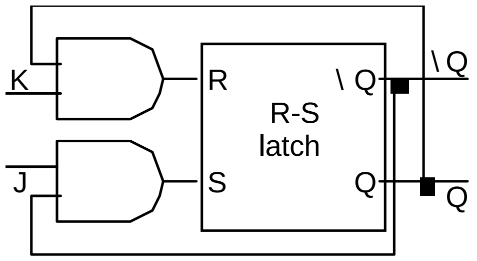
Sequential Switching Networks

J-K Flipflop

How to eliminate the forbidden state?

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

J, K both one yields toggle



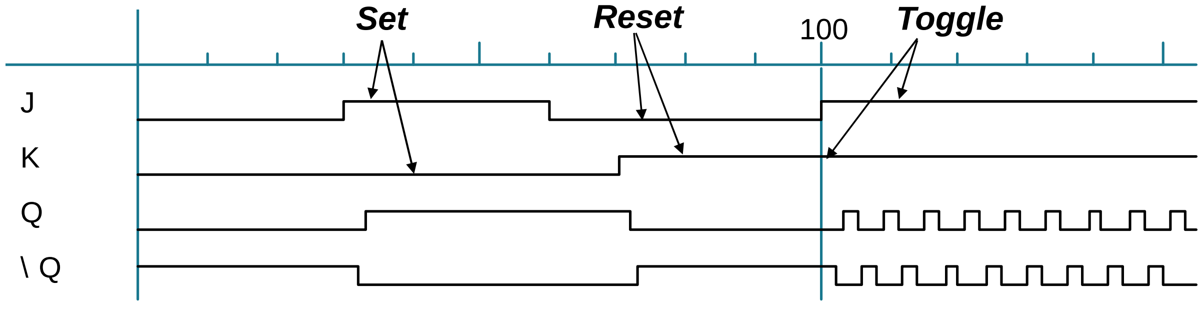
J(t)	K(t)	Q(t)	Q(t+Δ)	
0	0	0	0	Hold
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

Characteristic Equation:

$$Q^{+} = Q\,\overline{K} + Q\,\overline{J}$$

Sequential Switching Networks

J-K Latch: Race Condition



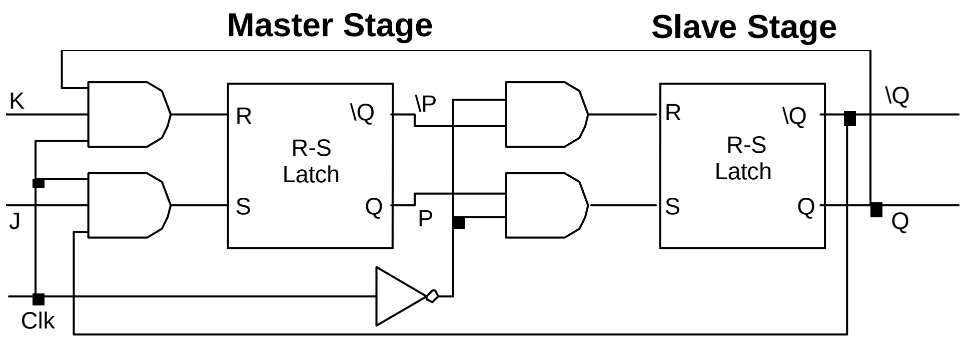
Race Condition

Toggle Correctness: Single State change per clocking event

Solution: Master/Slave Flipflop

Sequential Switching Network

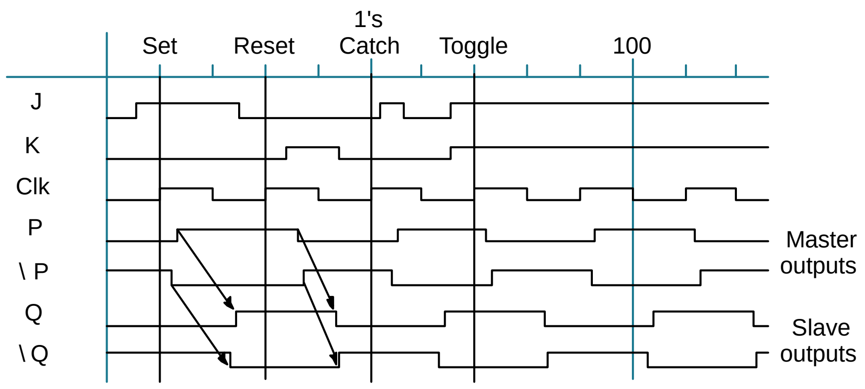
Master/Slave J-K Flipflop



Sample inputs while clock high

Sample inputs while clock low

Uses time to break feedback path from outputs to inputs!



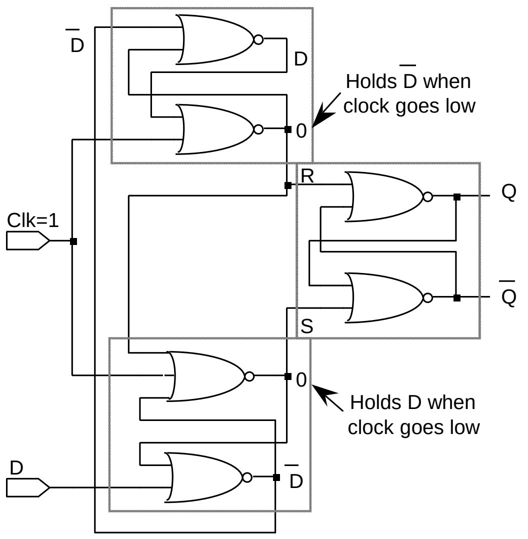
Correct Toggle Operation

Sequential Switching Networks

Edge-Triggered Flipflops

***1's Catching*: a 0-1-0 glitch on the J or K inputs leads to a state change!**
forces designer to use hazard-free logic

***Solution*: edge-triggered logic**



Negative edge-triggered FF
when clock is high

Negative Edge-Triggered D flipflop

4-5 gate delays

setup, hold times necessary to successfully latch the input

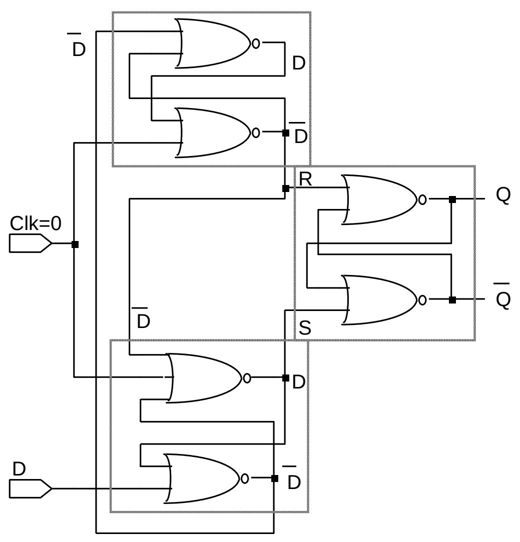
Characteristic Equation:

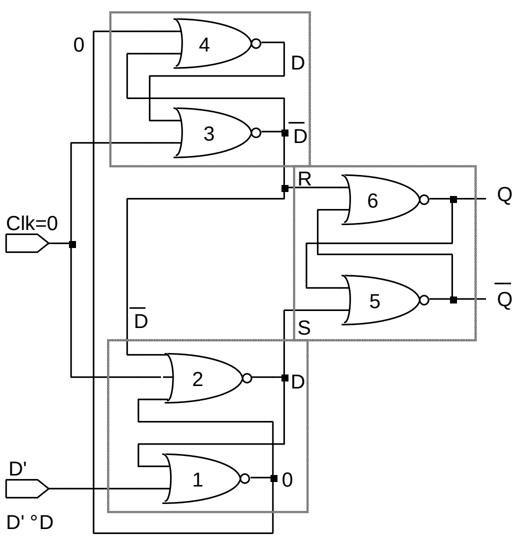
$Q+ = D$

Sequential Switching Network

Edge-triggered Flipflops

Step-by-step analysis



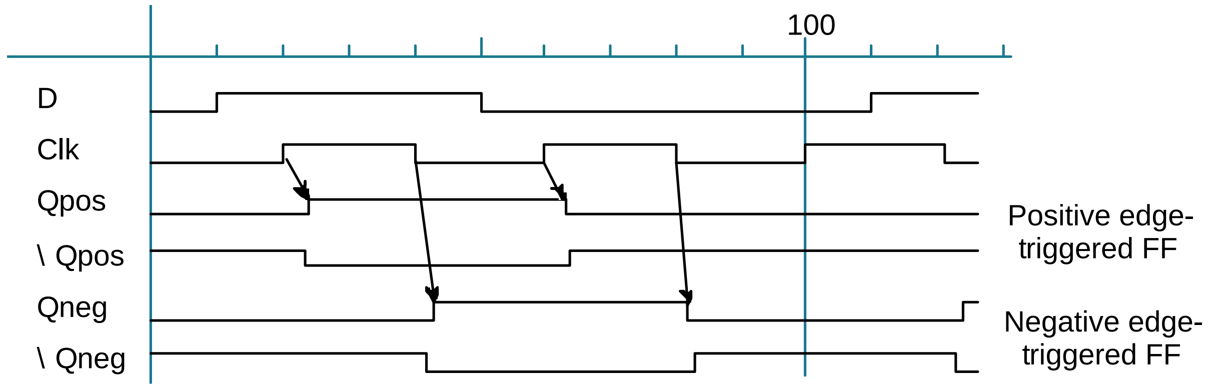


Negative edge-triggered FF when clock goes high-to-low data is latched

Negative edge-triggered FF when clock is low data is held

Sequential Switching Networks

Positive vs. Negative Edge Triggered Devices



Positive Edge Triggered

Inputs sampled on rising edge
Outputs change after rising edge

Negative Edge Triggered

Inputs sampled on falling edge
Outputs change after falling edge

Toggle Flipflop

Formed from J-K with both inputs wired together

Timing Methodology

Overview

- Set of rules for interconnecting components and clocks
- When followed, guarantee proper operation of system

- Approach depends on building blocks used for memory elements

  For systems with latches:

  Narrow Width Clocking

  Multiphase Clocking (e.g., Two Phase Non-Overlapping)

  For systems with edge-triggered flipflops:

  Single Phase Clocking

- Correct Timing:

  (1) correct inputs, with respect to time, are provided to the FFs

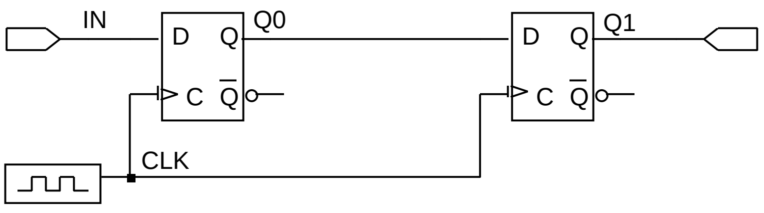
  (2) no FF changes more than once per clocking event

Timing Methodologies

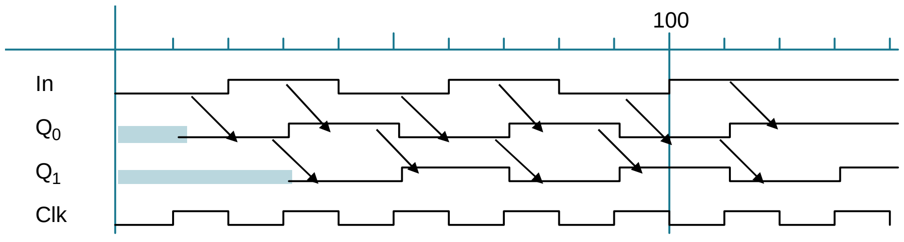
Cascaded Flipflops and Setup/Hold/Propagation Delays

Shift Register
S,R are preset, preclear

New value to first stage
while second stage
obtains current value
of first stage



Correct Operation,
assuming positive
edge triggered FF

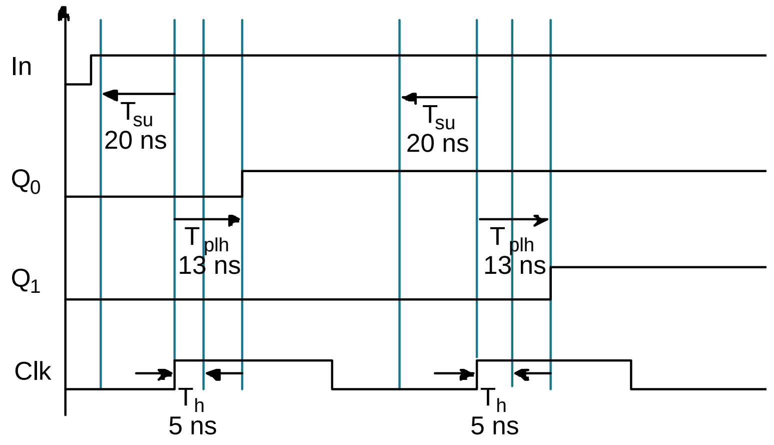


Timing Methodologies

Cascaded Flipflops 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
- Assumes infinitely fast distribution of the clock



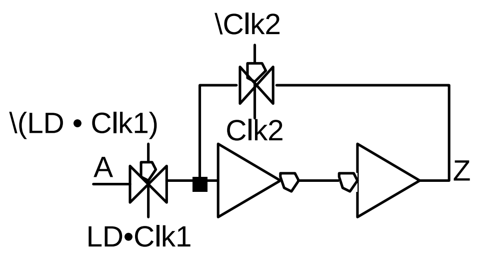
Timing constraints guarantee proper operation of cascaded components

Timing Methodologies

Narrow Width Clocking versus Multiphase Clocking

Level Sensitive Latches vs. Edge Triggered Flipflops

- Latches use fewer gates to implement a memory function
- Less complex clocking with edge triggered devices



CMOS Dynamic Storage Element

Feedback path broken by two phases of the clock (just like master/slave idea!)

8 transistors to implement memory function

but requires two clock signals constrained to be non-overlapping

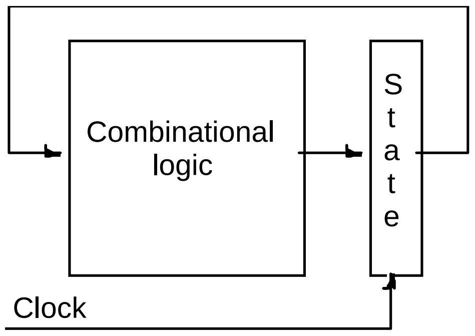
Edge-triggered D-FF: 6 gates (5 x 2-input, 1 x 3-input) = 26 transistors!

Timing Methodologies

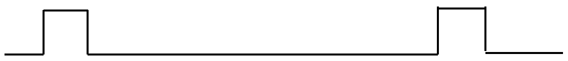
Narrow Width Clocking for Systems with Latches for State

Generic Block Diagram for Clocked Sequential System

state implemented by latches or edge-triggered FFs



Two-sided Constraints:
must be careful of very fast signals as well as very slow signals!



Clock Width < fastest propagation through comb. logic
plus latch prop delay

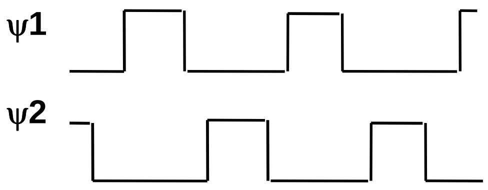
Clock Period > slowest propagation through comb. logic
(rising edge to rising edge)

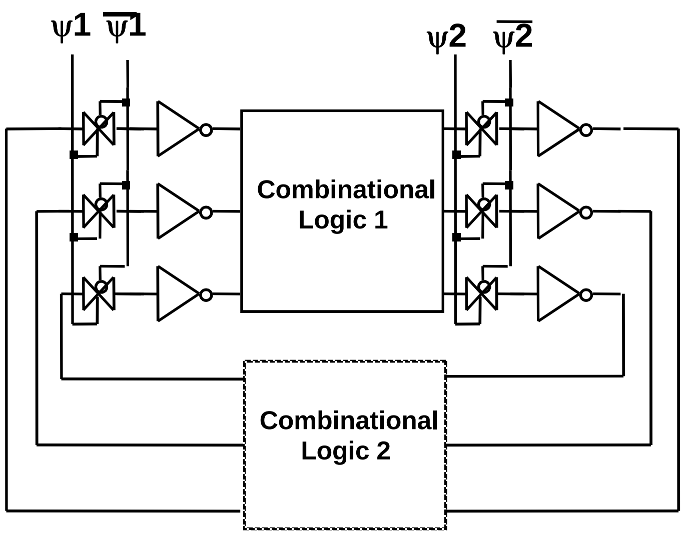
Timing Methodologies

Two Phase Non-Overlapped Clocking

Clock Waveforms:
must never overlap!

only worry about slow signals





Embedding CMOS storage element into Clocked Sequential Logic

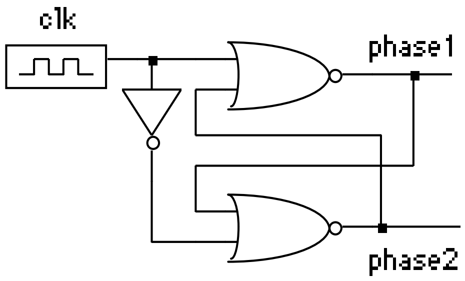
Note that Combinational Logic can be partitioned into two pieces

C/L1: inputs latched and stable by end of phase 1; compute between phases, latch outputs by end of phase 2

C/L2: just the reverse

Timing Methodologies

Generating Two-Phase Non-Overlapping Clocks

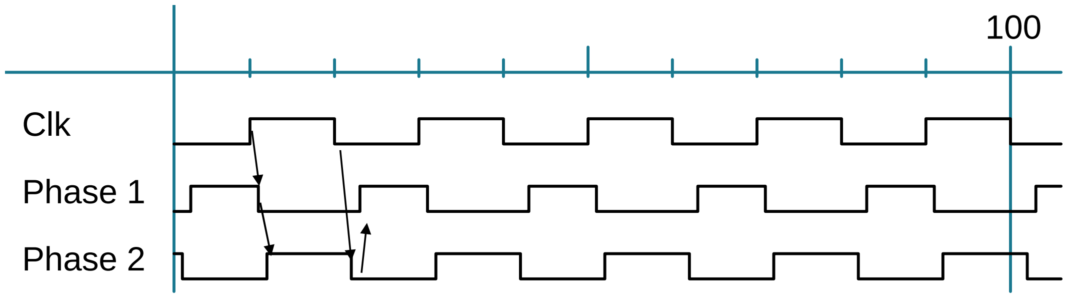


Single reference clock (or crystal)

Phase 1 high while clock is low

Phase 2 high while clock is high

Phase X cannot go high until phase Y goes low!



Non-overlap time can be increased by increasing the delay on the feedback path

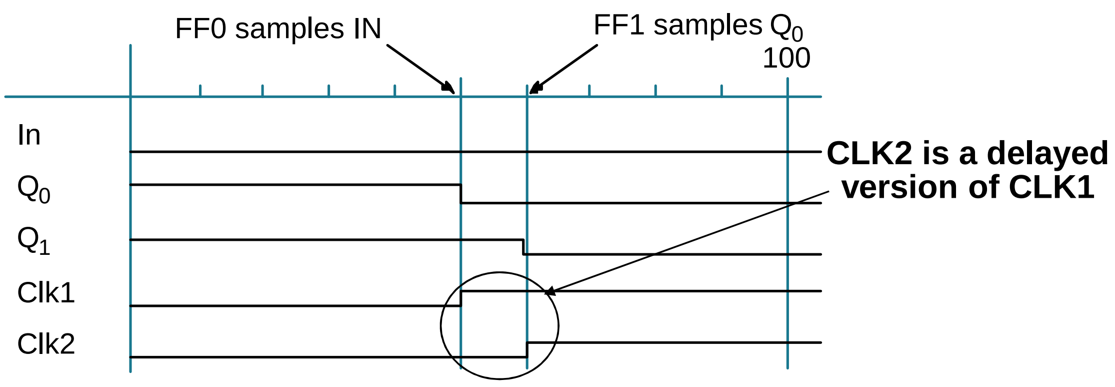
Timing Methodologies

The Problem of Clock Skew

Correct behavior assumes next state of all storage elements determined by all storage elements *at the same time*

Not possible in real systems!
- logical clock driven from more than one physical circuit with timing behavior
- different wire delay to different points in the circuit

Effect of Skew on Cascaded Flipflops:



Original State: Q0 = 1, Q1 = 1, In = 0
Because of skew, next state becomes: Q0 = 0, Q1 = 0, not Q0 = 0, Q1 = 1

Timing Methodologies

Design Strategies for Minimizing Clock Skew

Typical propagation delays for LS FFs: 13 ns

Need substantial clock delay (on the order of 13 ns) for skew to be a problem in this relatively slow technology

Nevertheless, the following are good design practices:

- distribute clock signals in general direction of data flows
- wire carrying the clock between two communicating components should be as short as possible
- for multiphase clocked systems, distribute all clocks in similar wire paths; this minimizes the possibility of overlap
- for the non-overlap clock generate, use the phase feedback signals from the furthest point in the circuit to which the clock is distributed; this guarantees that the phase is seen as low everywhere before it allows the next phase to go high

Realing Circuits with Different Kinds of FFs

Choosing a Flipflop

R-S Clocked Latch:
used as storage element in narrow width clocked systems
its use is not recommended!
however, fundamental building block of other flipflop types

J-K Flipflop:
versatile building block
can be used to implement D and T FFs
usually requires least amount of logic to implement (In,Q,Q+)
but has two inputs with increased wiring complexity

because of 1's catching, never use master/slave J-K FFs
edge-triggered varieties exist

D Flipflop:
minimizes wires, much preferred in VLSI technologies
simplest design technique
best choice for storage registers

T Flipflops:
don't really exist, constructed from J-K FFs
usually best choice for implementing counters

Preset and Clear inputs highly desirable!!

Realizing Circuits with Different Kinds of Flipflops

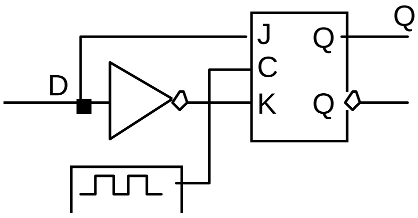
Characteristic Equations

R-S:	$Q^+ = S + R\overline{Q}$
D:	$Q^+ = D$
J-K:	$Q^+ = J\overline{Q} + K\overline{Q}$
T:	$Q^+ = T\overline{Q} + T\overline{Q}$

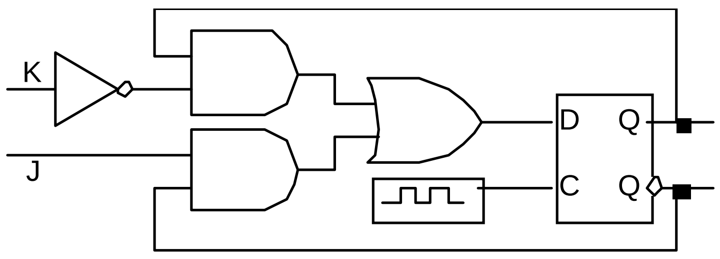
Derived from the K-maps for $Q^+ = (\text{Inputs}, Q)$

E.g., J=K=0, then $Q^+ = Q$
J=1, K=0, then $Q^+ = 1$
J=0, K=1, then $Q^+ = 0$
J=1, K=1, then $Q^+ = \overline{Q}$

Implementing One FF in Terms of Another



D implemented with J-K



J-K implemented with D

Realizing Circuits with Different Kinds of Flipflops

Design Procedure

Excitation Tables: What are the necessary inputs to cause a particular kind of change in state?

Q	Q^+	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

Implementing D FF with a J-K FF:

1) Start with K-map of $Q^+ = (D, Q)$

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

Q \ D	0	1
0	0	1
1	0	1

$Q^+ = D$

E.g., D = Q= 0, $Q^+ = 0$
then J = 0, K = X

Q \ D	0	1
0	0	1
1	X	X

J = D

Q \ D	0	1
0	X	X
1	1	0

$K = \overline{D}$

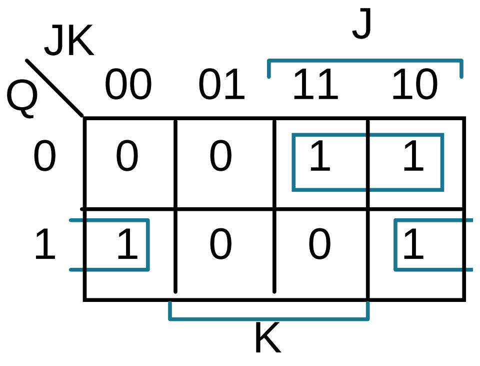
Realizing Circuits with Different Kinds of Flipflops

Design Procedure (Continued)

Implementing J-K FF with a D FF:

1) K-Map of $Q^+ = F(J, K, Q)$

2,3) Revised K-map using D's excitation table
its the same! that is why design procedure with D FF is simple!



$$Q^+ = D = J\overline{Q} + \overline{K}Q$$

Resulting equation is the combinational logic input to D to cause same behavior as J-K FF. Of course it is identical to the characteristic equation for a J-K FF.

Metastability and Asynchronous Inputs

Terms and Definitions

Clocked synchronous circuits
- **common reference signal called the clock**
- **state of the circuit changes in relation to this clock signal**

Asynchronous circuits
- **inputs, state, and outputs sampled or changed independent of a common reference signal**

R-S latch is asynchronous, J-K master/slave FF is synchronous

Synchronous inputs
- **active only when the clock edge or level is active**

Asynchronous inputs
- **take effect immediately, without consideration of the clock**

Compare R, S inputs of clocked transparent latch vs. plain latch

Metastability and Asynchronous Inputs

Asynchronous Inputs Are Dangerous!

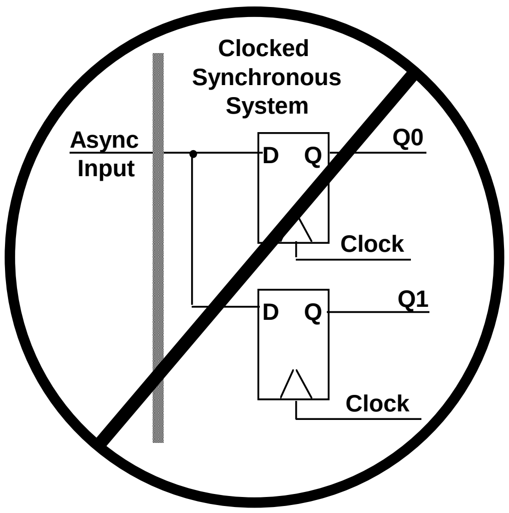
Since they take effect immediately, glitches can be disastrous

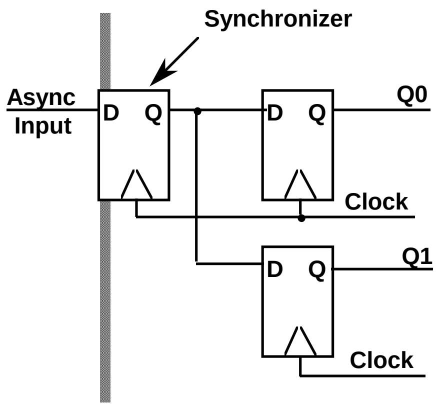
Synchronous inputs are greatly preferred!

But sometimes, asynchronous inputs cannot be avoided
e.g., reset signal, memory wait signal

Metastability and Asynchronous Outputs

Handling Asynchronous Inputs

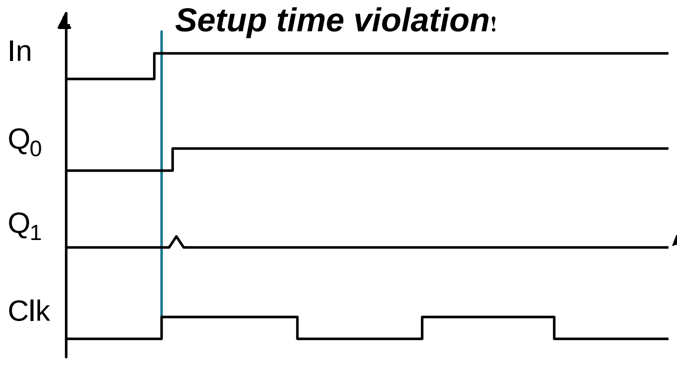




Never allow asynchronous inputs to be fanned out to more than one FF within the synchronous system

Metastability and Asynchronous Inputs

What Can Go Wrong



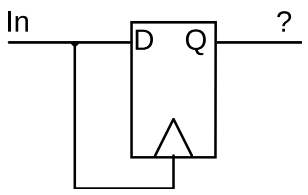
In is asynchronous
Fans out to D0 and D1
One FF catches the signal, one does not

impossible state might be reached!

Single FF that receives the asynchronous signal is *a synchronizer*

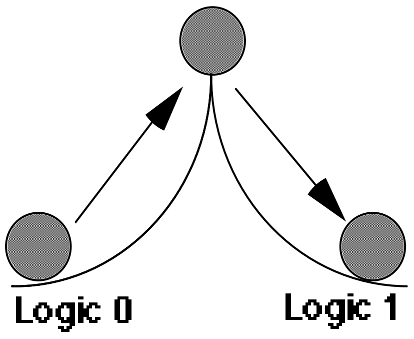
Metastability and Asynchronous Inputs

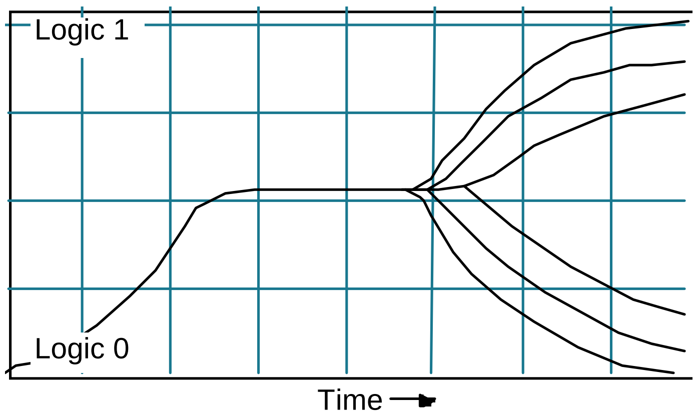
Synchronizer Failure



When FF input changes close to clock edge, the FF may enter the *metastable* state: neither a logic 0 nor a logic 1

It may stay in this state an indefinate amount of time, although this is not likely in real circuits





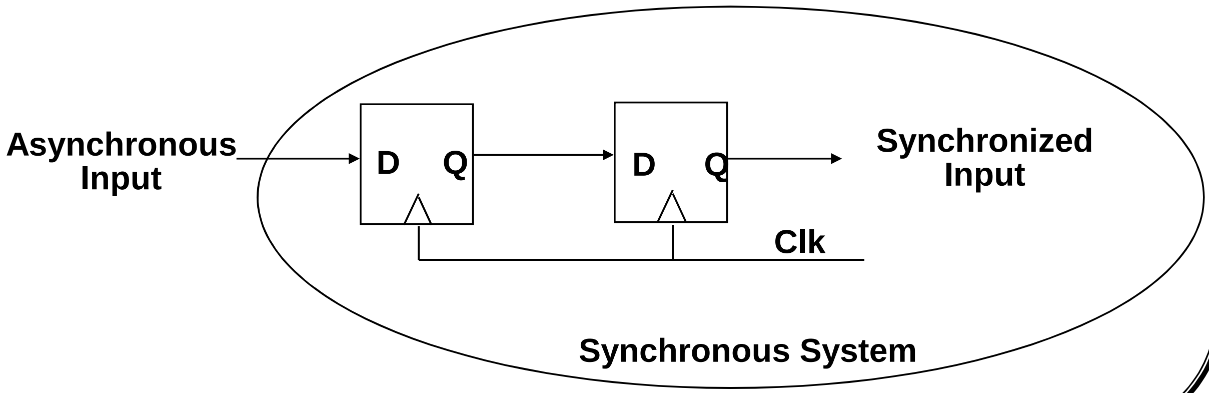
Small, but non-zero probability that the FF output will get stuck in an in-between state

Oscilloscope Traces Demonstrating Synchronizer Failure and Eventual Decay to Steady State

Metastability and Asynchronous Inputs

Solutions to Synchronizer Failure

- the probability of failure can never be reduced to 0, but it can be reduced
- 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
- 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 recommended
- cascade two synchronizers



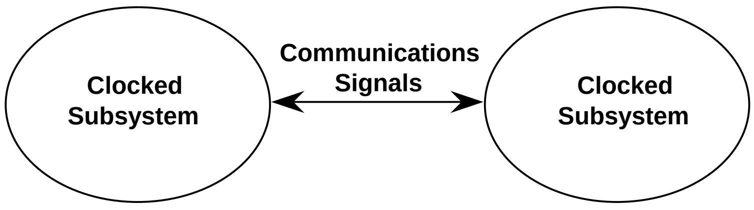
Self-Timed and Speed Independent Circuits

Limits of Synchronous Systems

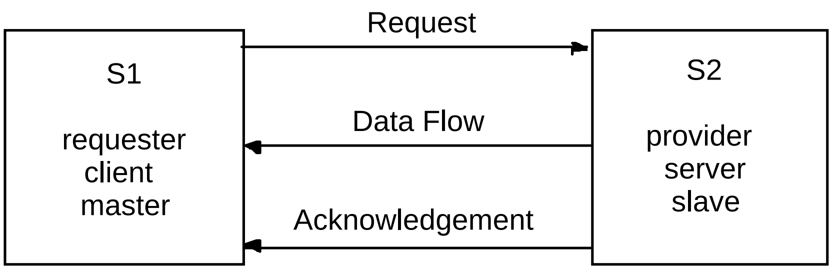
Fully synchronous not possible for very large systems
because of problems of clock skew

Partition system into components that are locally clocked

These communicate using "speed independent" protocols

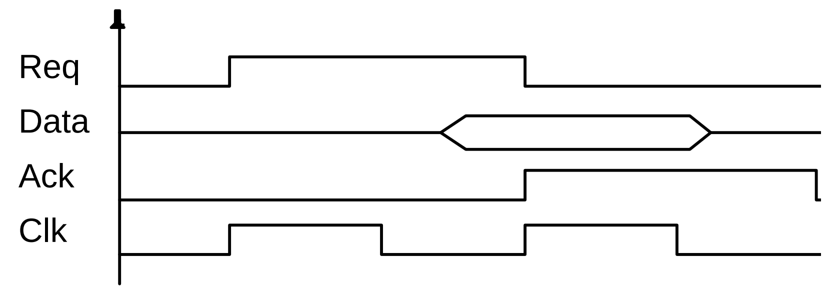


Request/Acknowledgement Signaling

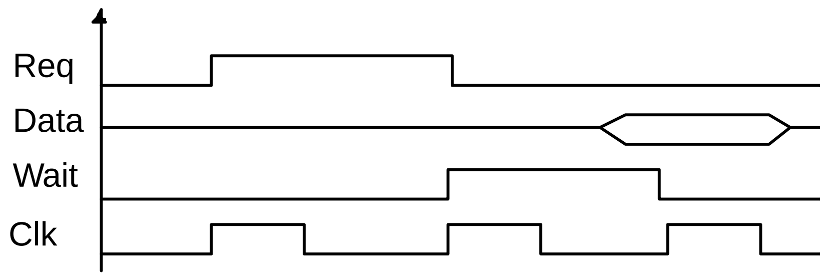


Self-Timed and Speed Independent Circuits

Synchronous Signaling



Master issues read request; Slave produces data and acks back



Alternative Synchronous Scheme:

Slave issues WAIT signal if it cannot satisfy request in one clock cycle

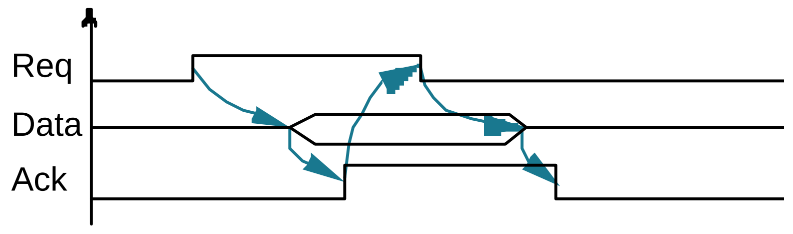
Self-Timed and Speed Independent Circuits

Asynchronous/Speed Independent Signaling

Communicate information by signal levels rather than edges!

No clock signal

4 Cycle Signaling/Return to Zero Signaling



(1) master raises request slave performs request

(2) slave "done" by raising acknowledge

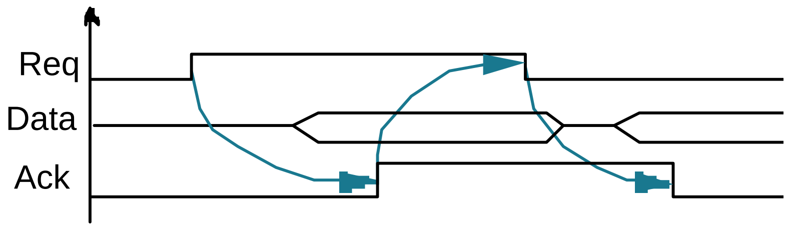
(3) master latches data acks by lowering request

(4) slave resets self by lowing acknowledge signal

Self-Timed and Speed Independent Circuits

Alternative: 2 cycle signaling

Non-Return-to-Zero



**(1) master raises request
slave services request**

**(2) slave indicates that it is
done by raising acknowledge**

Next request indicated by low level of request

**Requires additional state in master and slave
to remember previous setting or request/acknowledge**

4 Cycle Signaling is more foolproof

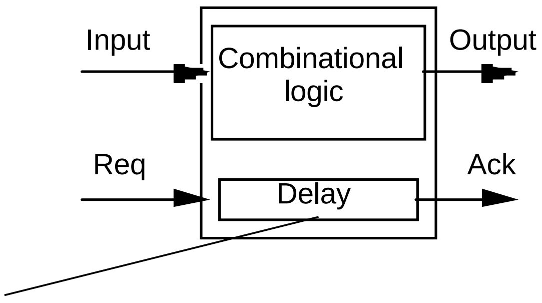
Self-Timed and Speed Independent Circuits

Self-Timed Circuits

Determine on their own when a given request has been serviced

No internal clocks

**Usually accomplished by modeling worse case delay within
self-timed component**



**Models worst case delay
e.g., if combinational logic is 5 gate levels deep,
delay line between request in and ack out is
also 5 levels deep**

Chapter Summary

- Fundamental Building Block of Circuits with State: latch and flipflop
- R-S Latch, J-K master/slave Flipflop, Edge-triggered D Flipflop
- Clocking Methodologies:

  For latches: Narrow width clocking vs. Multiphase Non-overlapped
  - Narrow width clocking and two sided timing constraints
  - Two phase clocking and single sided timing constraints

  For FFs: Single phase clocking with edge triggered flipflops

  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
- Speed Independent Circuits

  Asynchronous Signaling Conventions: 4 and 2 Cycle Handshakes

  Self-Timed Circuits