

Digital Logic Design

ECEN 3233

Module 7 – Timing Methodologies

Weihoa Sheng
School of Electrical and Computer Engineering
Oklahoma State University

Spring 2007

Flipflop Review

- Notation, Characteristic Equations
 - Q^* (or Q_+) means “the next value of Q .”
 - “Excitation” is the input applied to a flipflop that determines the next state.
 - “Characteristic equation” specifies the next state of a device as a function of its excitation.
 - Edge-triggered D flip-flop: $Q^* = D$
 - Very simple relationship
 - “The D flip flop is your friend!”

Characteristic Equations

$$\text{R-S: } Q_+ = S + \bar{R}Q$$

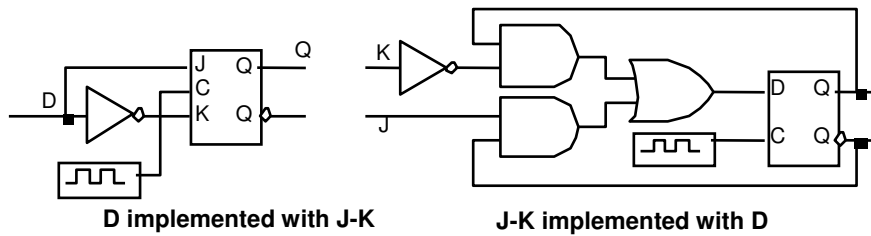
$$\text{D: } Q_+ = D$$

$$\text{J-K: } Q_+ = J\bar{Q} + \bar{K}Q$$

$$\text{T: } Q_+ = T\bar{Q} + \bar{T}Q$$

Realizing Circuits with Different Kinds of Flipflops

Implementing One FF in Terms of Another



Realizing Circuits with Different Kinds of Flipflops

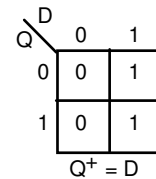
Design Procedure

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

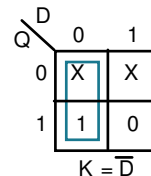
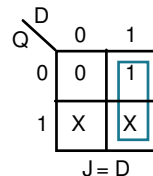
Q	Q ⁺	R	S	J	K	T	D
0	0	X	0	0	X	0	0
0	1	0	1	1	X	1	1
1	0	1	0	X	1	1	0
1	1	0	X	X	0	0	1

Implementing D FF with a J-K FF:

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



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



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!

		J			
	JK	00	01	11	10
Q	0	0	0	1	1
	1	1	0	0	1
		K			

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

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.

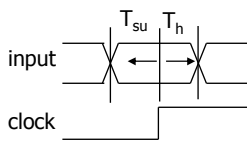
Timing methodologies

- 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

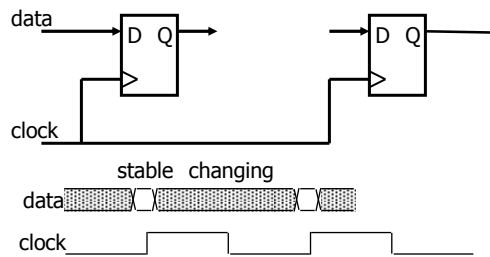
Timing methodologies (cont'd)

■ Definition of terms

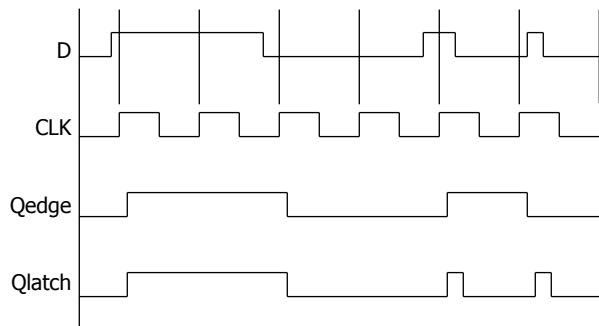
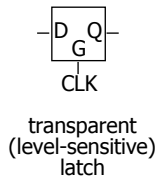
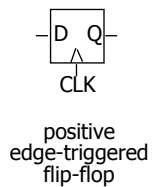
- clock: periodic event, causes state of memory element to change
can be rising edge or falling edge or high level or low level
- setup time: minimum time before the clocking event by which the input must be stable (T_{su})
- hold time: minimum time after the clocking event until which the input must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



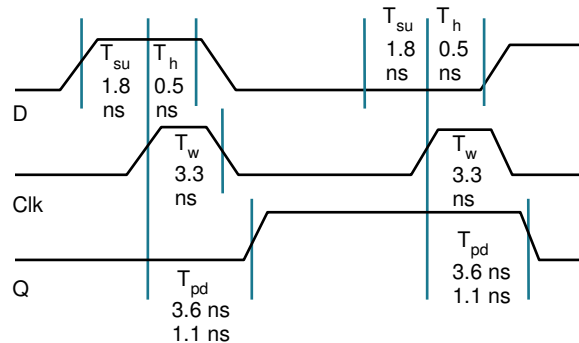
behavior is the same unless input changes while the clock is high

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
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

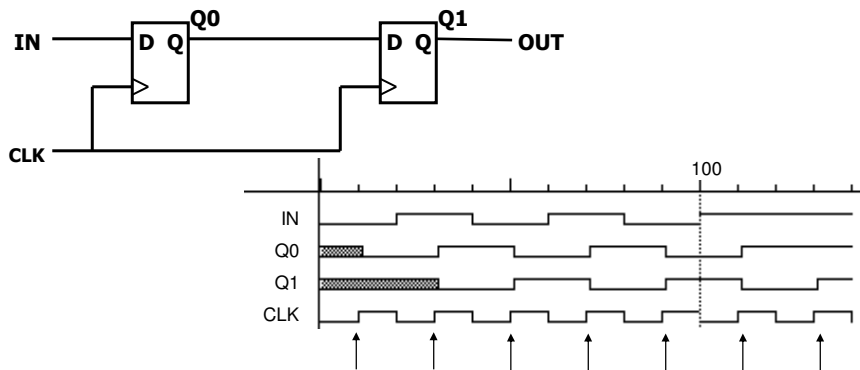
- Positive edge-triggered D flip-flop
 - setup and hold times
 - minimum clock width
 - propagation delays (low to high, high to low, max and typical)



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

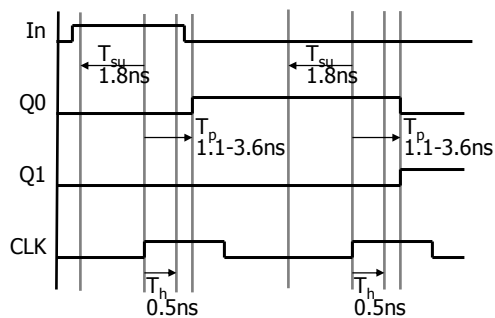
Cascading edge-triggered flip-flops

- 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

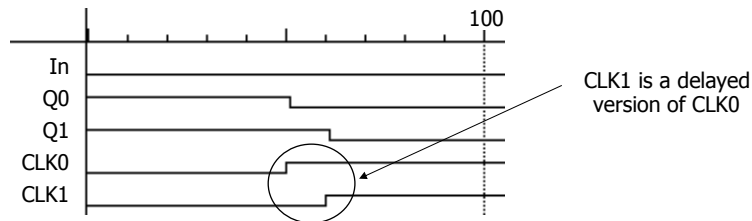


timing constraints
guarantee proper
operation of
cascaded components

assumes infinitely fast
distribution of the clock

Clock skew

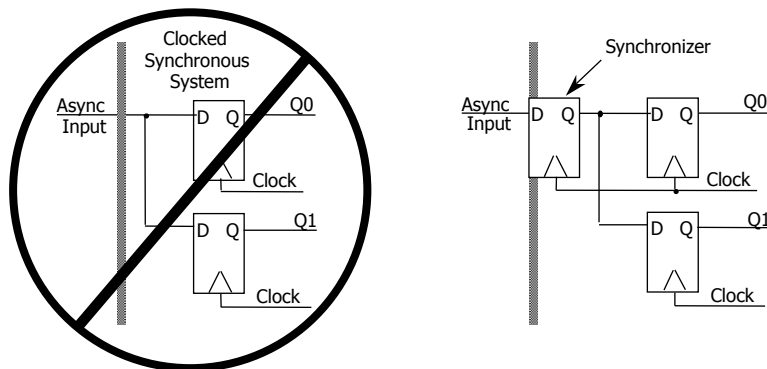
- The problem
 - correct behavior assumes next state of all storage elements determined by all storage elements at the same time
 - this is difficult in high-performance systems because time for clock to arrive at flip-flop is comparable to delays through logic
 - effect of skew on cascaded flip-flops:



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

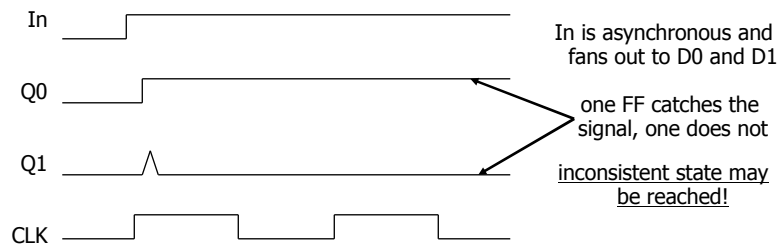
Handling asynchronous inputs

- Never allow asynchronous inputs to fan-out to more than one flip-flop
 - synchronize as soon as possible and then treat as synchronous signal



Handling asynchronous inputs (cont'd)

- What can go wrong?
 - input changes too close to clock edge (violating setup time constraint)

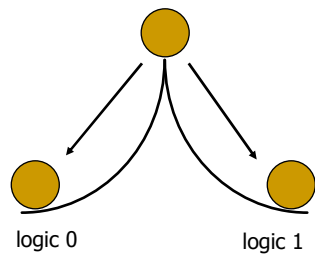


Metastability and asynchronous inputs

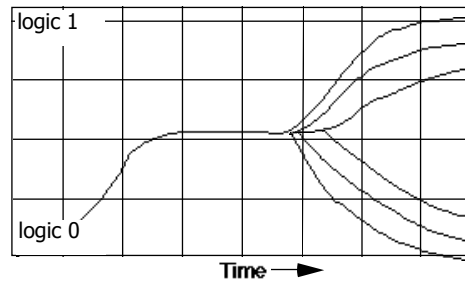
- Clocked synchronous circuits
 - inputs, state, and outputs sampled or changed in relation to a common reference signal (called the clock)
 - e.g., master/slave, edge-triggered
- Asynchronous circuits
 - inputs, state, and outputs sampled or changed independently of a common reference signal (glitches/hazards 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)

Synchronization failure

- Occurs 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
 - this is not likely in practice but has some probability



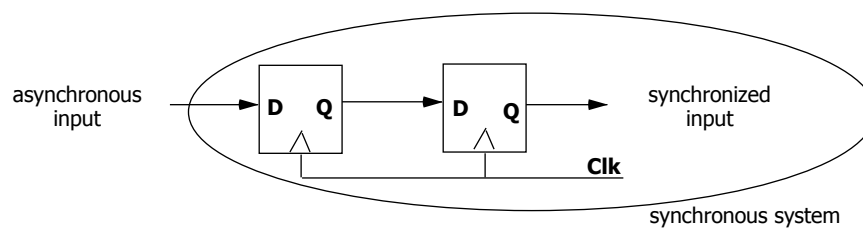
small, but non-zero probability
that the FF output will get stuck
in an in-between state



oscilloscope traces demonstrating
synchronizer failure and eventual
decay to steady state

Dealing with synchronization 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 technology in the synchronizer
this makes for a very sharp "peak" upon which to balance
 - (3) cascade two synchronizers
this effectively synchronizes twice (both would have to fail)



Summary

- Flipflop review
- Timing issues
 - use of clocks
 - cascaded FFs work because propagation delays exceed hold times
 - beware of clock skew
 - metastability
- Next: Finite State Machines