

Fundamentals of Computer Systems

Sequential Logic

Stephen A. Edwards

Columbia University

Summer 2021

State-Holding Elements

- Bistable Elements

- RS Latch

- D Latch

- Positive-Edge-Triggered D Flip-Flop

- D Flip-Flop with Enable

Synchronous Digital Logic

- The Synchronous Paradigm

- Shift Registers

- Counters

Timing in Synchronous Circuits

- Flip-Flop Timing

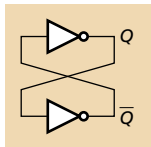
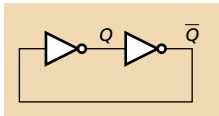
- Timing in Synchronous Circuits

- Clock Skew



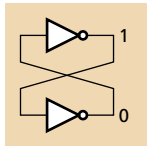
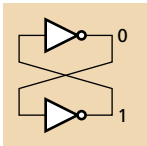
State-Holding Elements

Bistable Elements

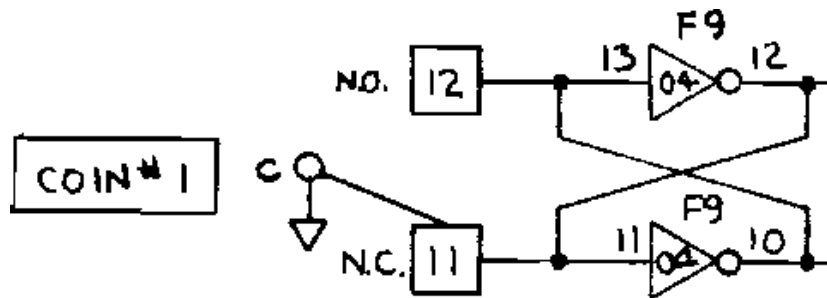


Equivalent circuits; right is more traditional.

Two stable states:



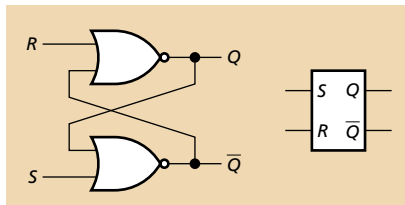
A Bistable in the Wild



This "debounces" the coin switch.

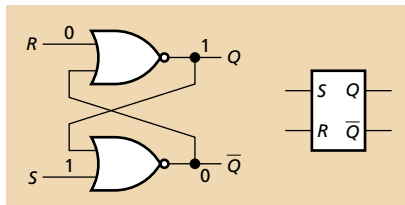
Breakout, Atari 1976.

RS Latch



R	S	Q	\bar{Q}
0	0		
0	1		
1	0		
1	1		

RS Latch

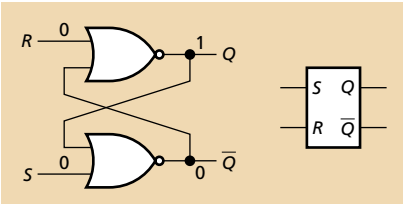


R	S	Q	\bar{Q}
0	0		
0	1	1	0
1	0		
1	1		

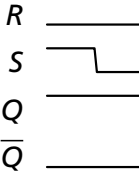
Set

R —
 S —
 Q — Set
 \bar{Q} —

RS Latch



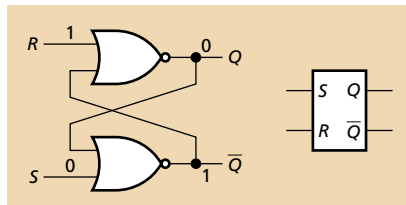
R	S	Q	Q̄	
0	0	Q	Q̄	Hold
0	1	1	0	Set
1	0			
1	1			



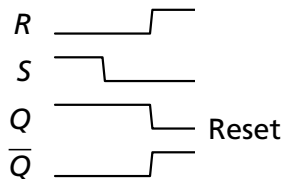
Hold, State 1



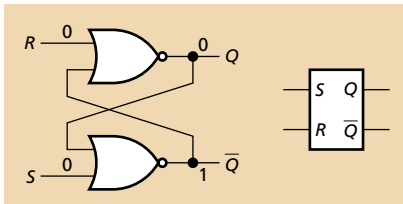
RS Latch



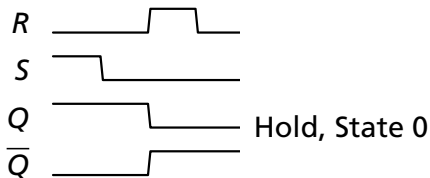
R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1			



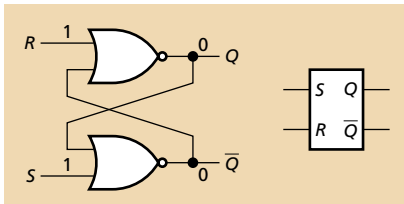
RS Latch



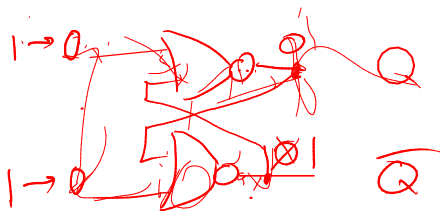
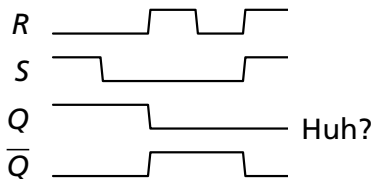
R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1			



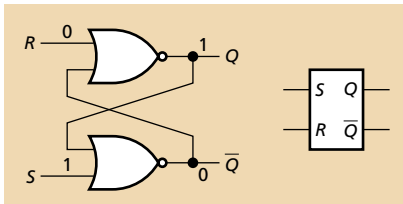
RS Latch



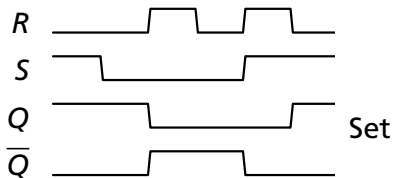
R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Bad



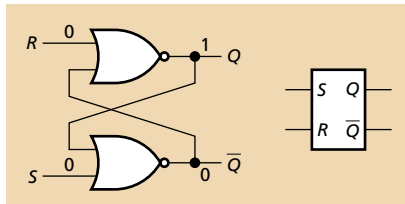
RS Latch



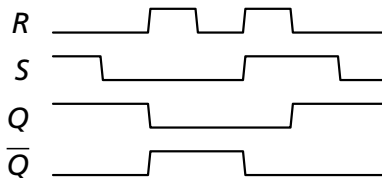
R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Bad



RS Latch

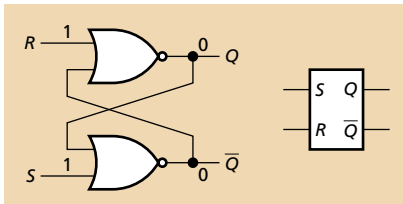


R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Bad

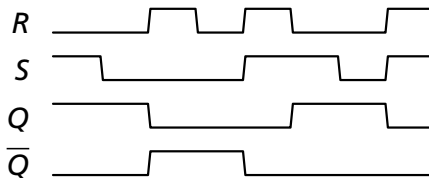


Hold, State 1

RS Latch

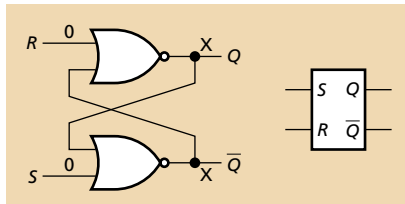


R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Bad

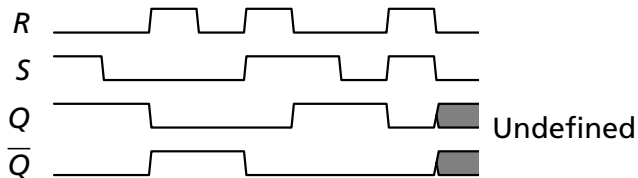


Huh?

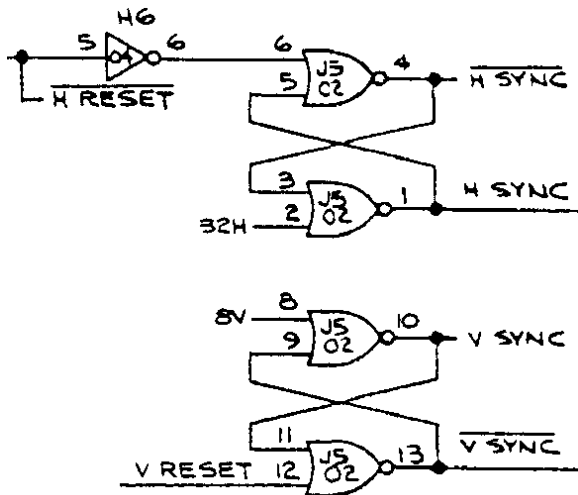
RS Latch



R	S	Q	\bar{Q}	
0	0	Q	\bar{Q}	Hold
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Bad



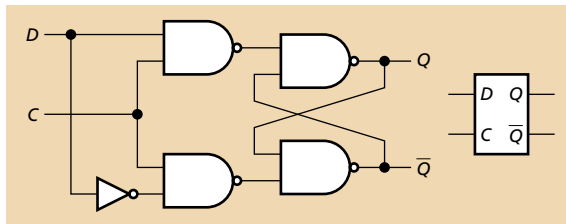
SR Latches in the Wild



Generates horizontal and vertical synchronization waveforms from counter bits.

Stunt Cycle, Atari 1976.

D Latch

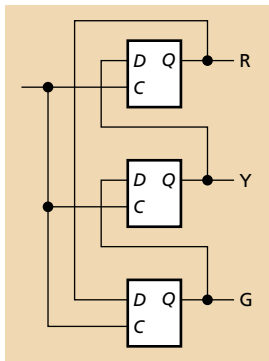


inputs		outputs	
C	D	Q	\bar{Q}
0	X	Q	\bar{Q}
1	0	0	1
1	1	1	0

A Challenge

A simple traffic light controller.

Want the lights to cycle green-yellow-red.



Does this work?





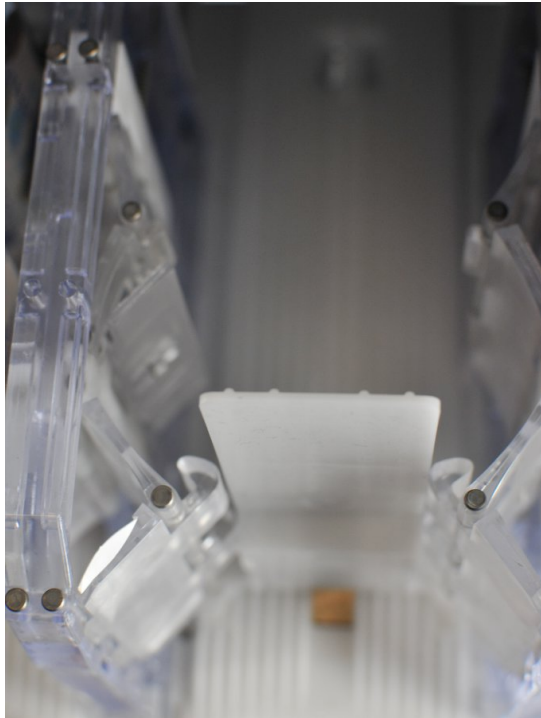


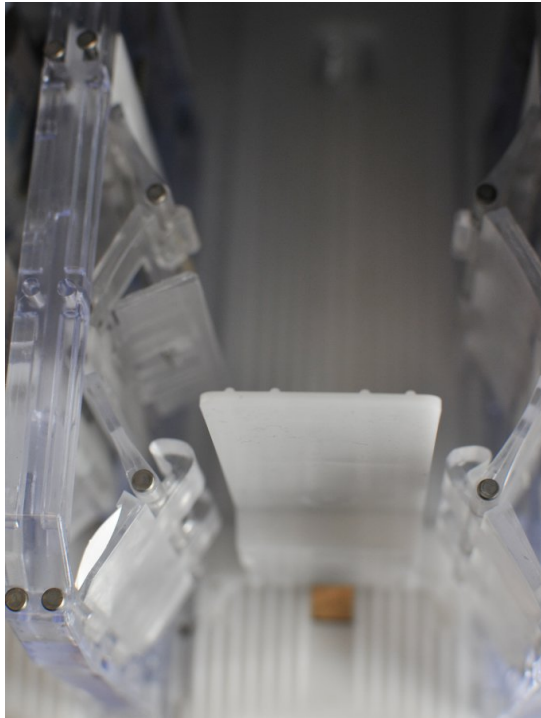


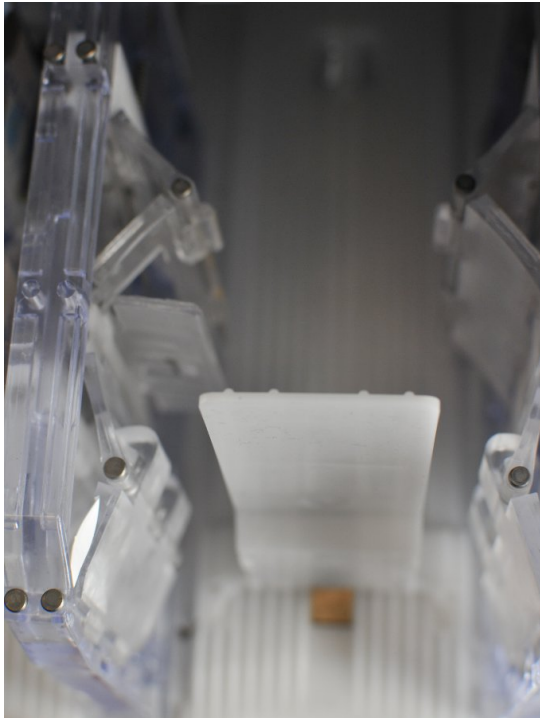


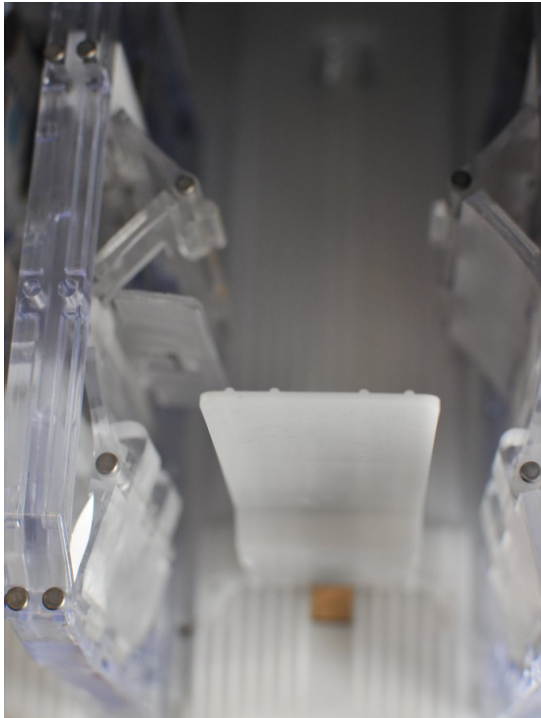


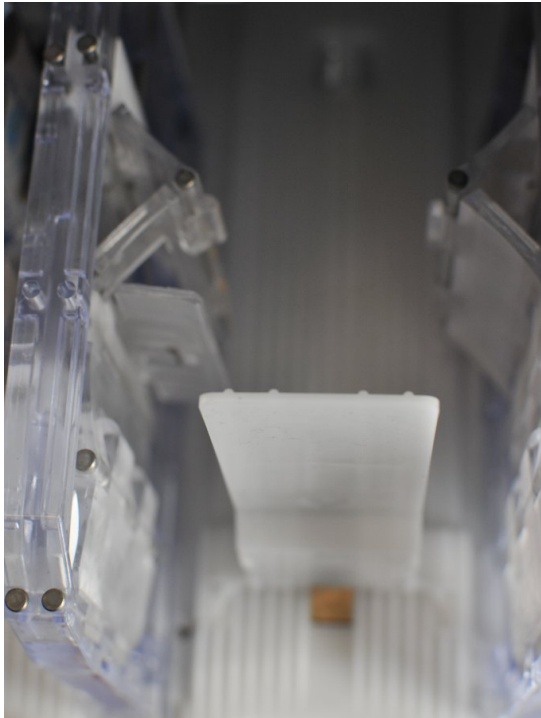






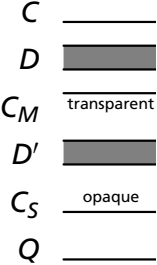
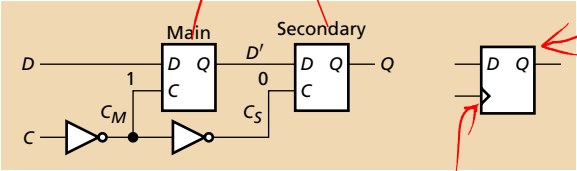






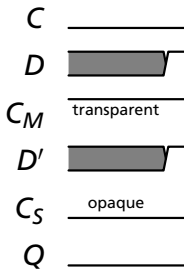
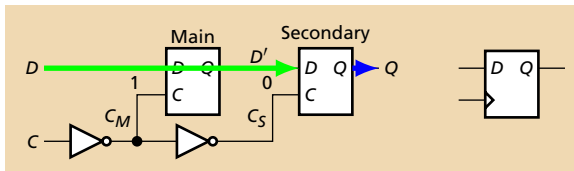
Positive-Edge-Triggered D Flip-Flop

Transparent D latches

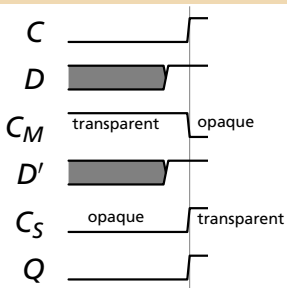
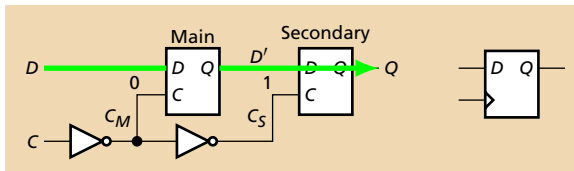


clock input → pos. edge

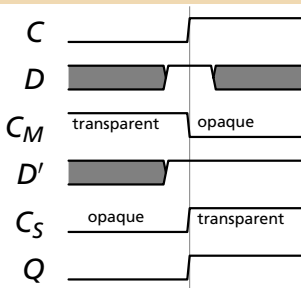
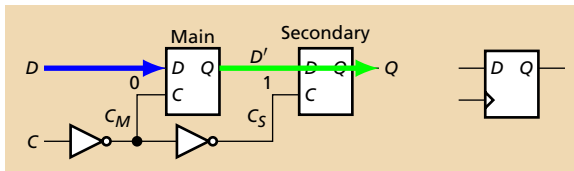
Positive-Edge-Triggered D Flip-Flop



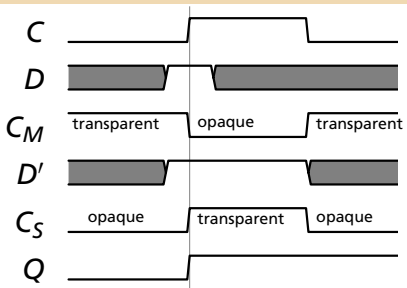
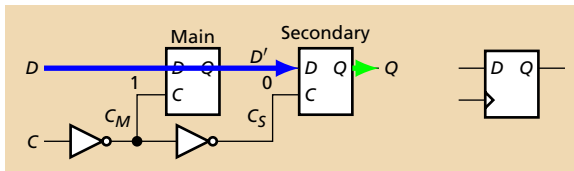
Positive-Edge-Triggered D Flip-Flop



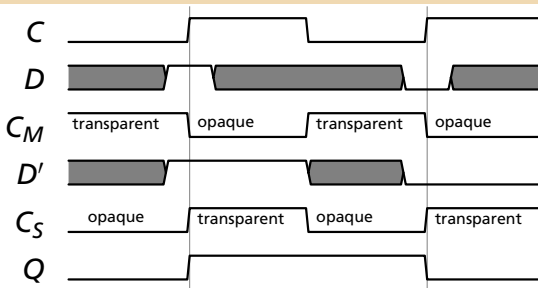
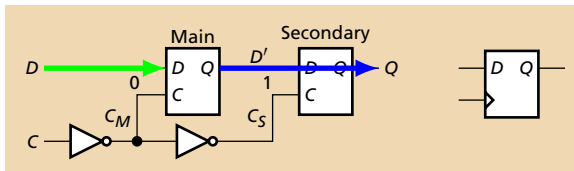
Positive-Edge-Triggered D Flip-Flop



Positive-Edge-Triggered D Flip-Flop

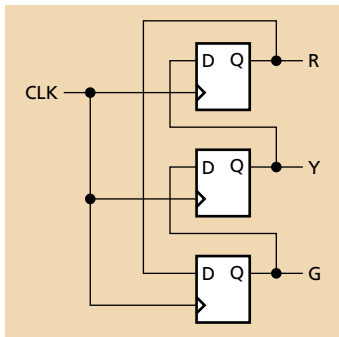


Positive-Edge-Triggered D Flip-Flop



The Traffic Light Controller: A second try

Let's try this again with D flip-flops.



CLK ___

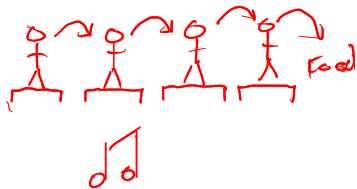
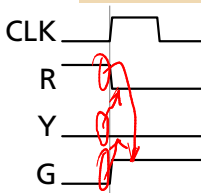
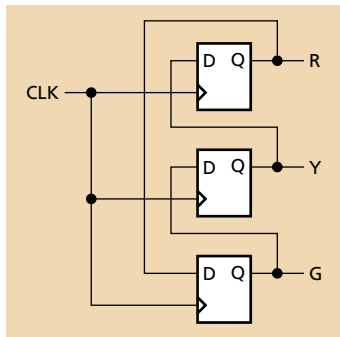
R ___

Y ___

G ___

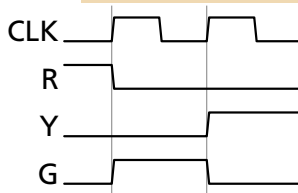
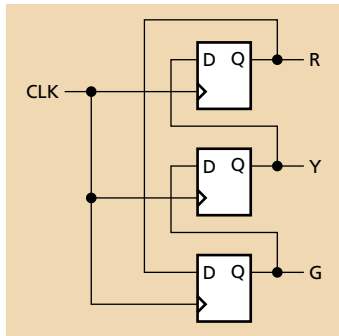
The Traffic Light Controller: A second try

Let's try this again with D flip-flops.



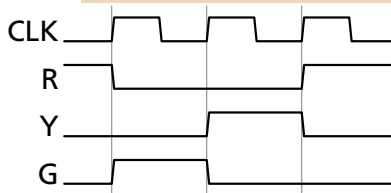
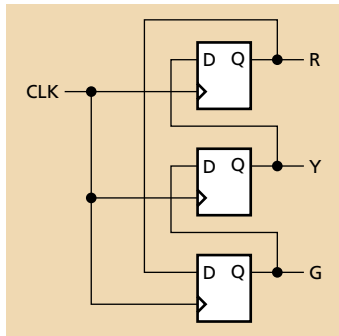
The Traffic Light Controller: A second try

Let's try this again with D flip-flops.



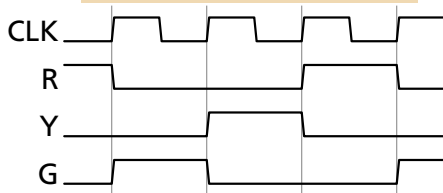
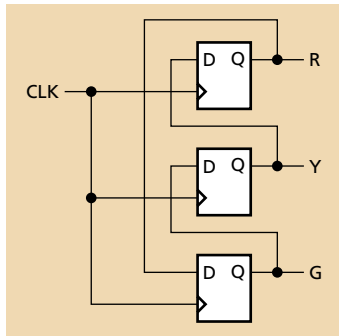
The Traffic Light Controller: A second try

Let's try this again with D flip-flops.

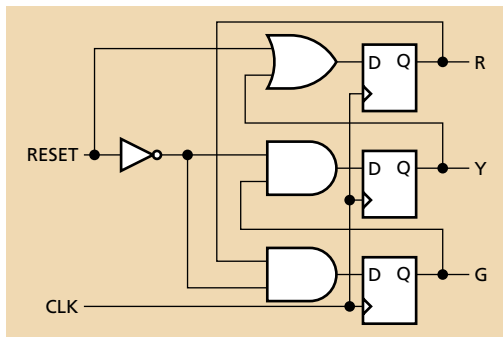


The Traffic Light Controller: A second try

Let's try this again with D flip-flops.

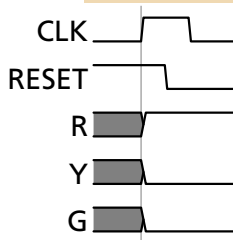
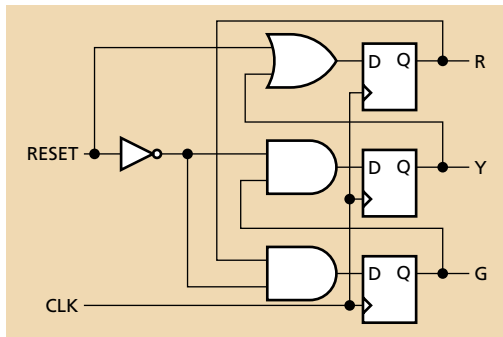


The Traffic Light Controller with Reset

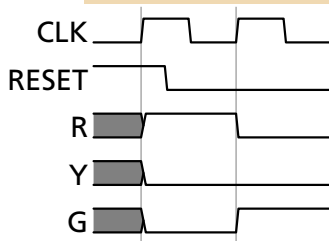
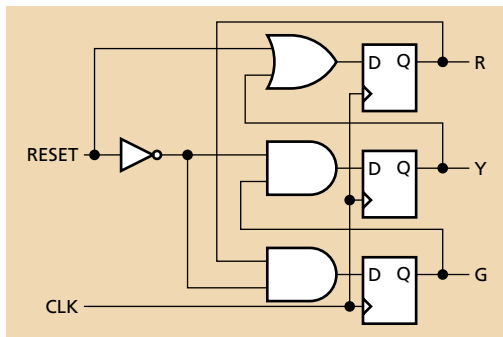


CLK _____
RESET _____
R
Y
G

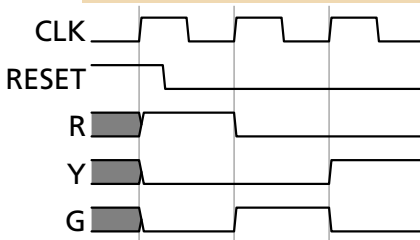
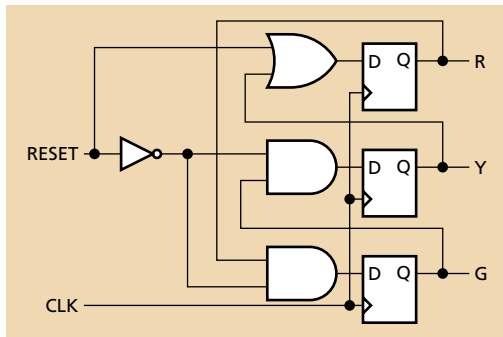
The Traffic Light Controller with Reset



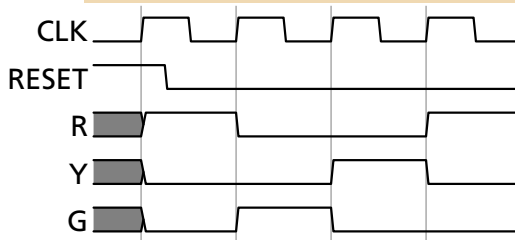
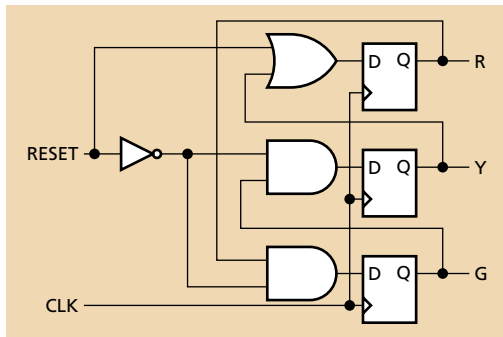
The Traffic Light Controller with Reset



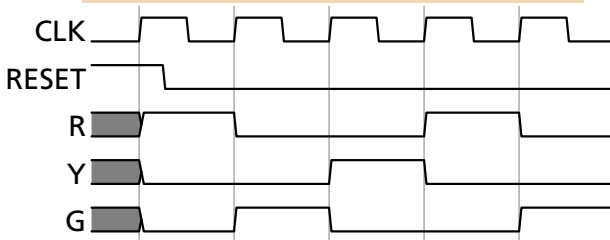
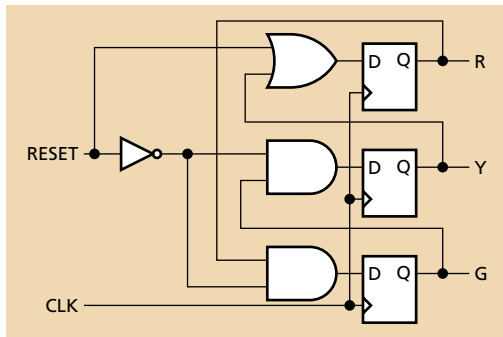
The Traffic Light Controller with Reset



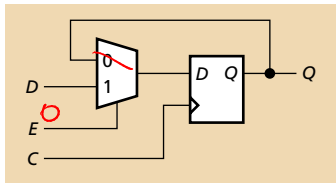
The Traffic Light Controller with Reset



The Traffic Light Controller with Reset

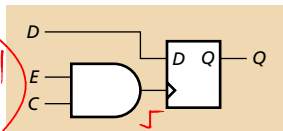


D Flip-Flop with Enable



1 "high"
0 "low"
rising/positive edge falling/negative edge

C	E	D	Q
↑	0	X	Q
↑	1	0	0
↑	1	1	1
0	X	X	Q
1	X	X	Q



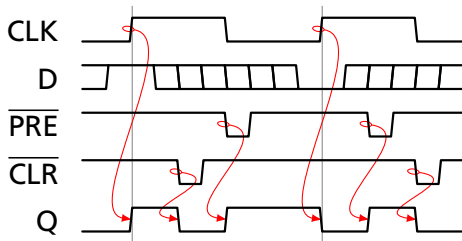
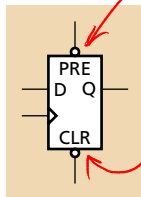
$E=0 \rightarrow C=1$

What's wrong with this solution?

C _____
E

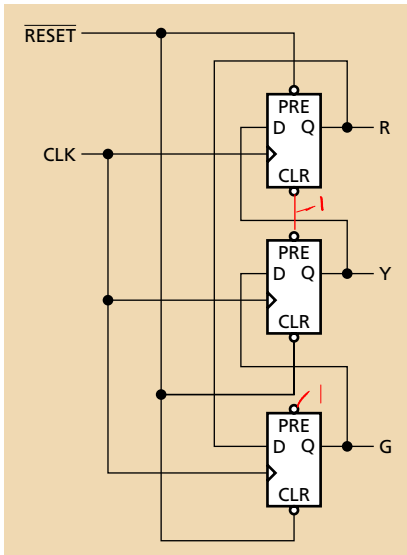
the problem: If clock is high, a glitch on E if E goes high could change Q

Asynchronous Preset/Clear



} Not how
you
use it

The Traffic Light Controller w/ Async. Reset

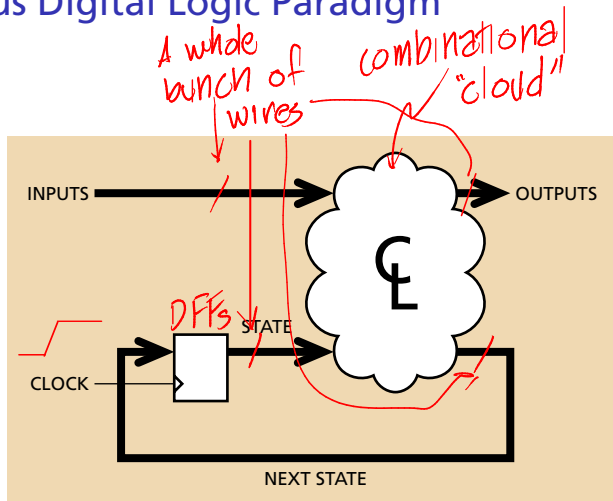


The Synchronous Digital Logic Paradigm

Gates and D flip-flops only

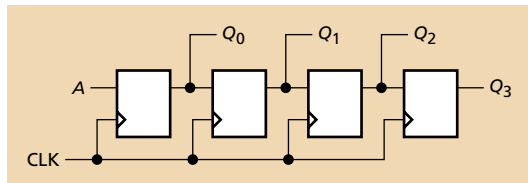
Each flip-flop driven by the same clock

Every cyclic path contains at least one flip-flop



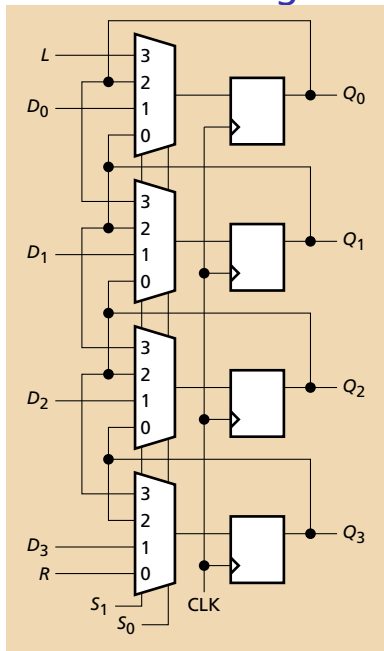
Discretizing Time

Cool Sequential Circuits: Shift Registers



A	Q_0	Q_1	Q_2	Q_3
0	X	X	X	X
1	0	X	X	X
1	1	0	X	X
0	1	1	0	X
1	0	1	1	0
0	1	0	1	1
0	0	1	0	1
0	0	0	1	0
1	0	0	0	1
0	1	0	0	0

Universal Shift Register

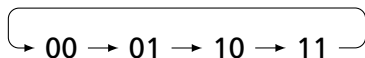


S_1	S_0	Q_3	Q_2	Q_1	Q_0
0	0	R	Q_3	Q_2	Q_1
0	1	D_3	D_2	D_1	D_0
1	0	Q_3	Q_2	Q_1	Q_0
1	1	Q_2	Q_1	Q_0	L

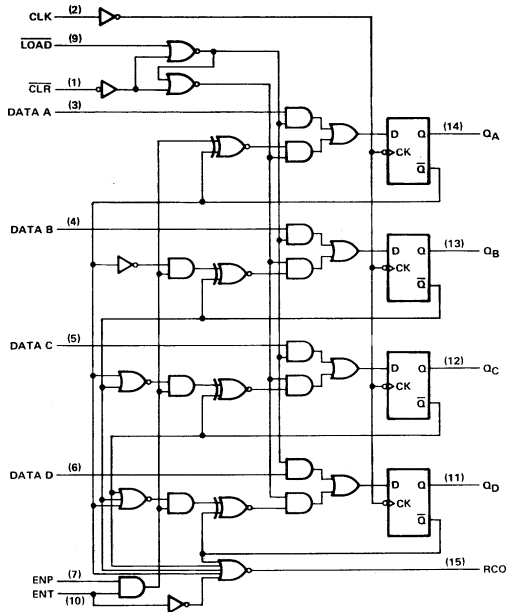
S_1	S_0	Operation
0	0	Shift right
0	1	Load
1	0	Hold
1	1	Shift left

Cool Sequential Circuits: Counters

Cycle through sequences of numbers, e.g.,

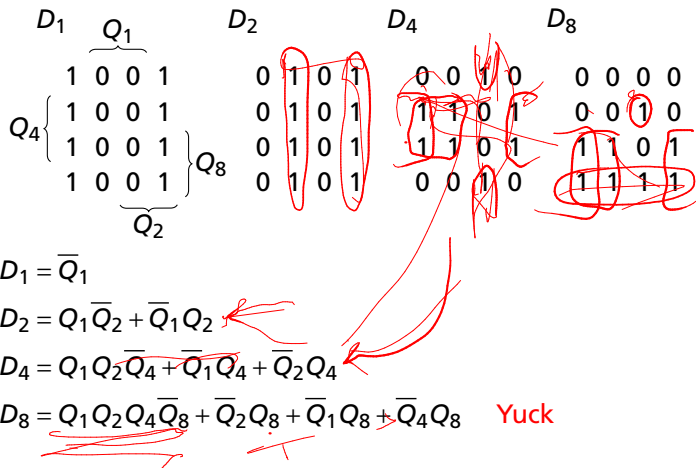


The 74LS163 Synchronous Binary Counter



Implementing a 4-bit Binary Counter

Q_{8421}	D_{8421}
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	1010
1010	1011
1011	1100
1100	1101
1101	1110
1110	1111
1111	0000



Implementing a 4-bit Binary Counter

Q_{8421}	D_{8421}
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	1010
1010	1011
1011	1100
1100	1101
1101	1110
1110	1111
1111	0000

D_1	Q_1	D_2	D_4	D_8
1	0 0 1	0 1 0 1	0 0 1 0	0 0 0 0
Q_4	1 0 0 1	0 1 0 1	1 1 0 1	0 0 1 0
	1 0 0 1	0 1 0 1	1 1 0 1	1 1 0 1
	1 0 0 1	0 1 0 1	0 0 1 0	1 1 1 1
Q_2				

Trick: XOR each D with its Q

$D_1 \oplus Q_1$	$D_2 \oplus Q_2$	$D_4 \oplus Q_4$	$D_8 \oplus Q_8$
Q_1			
1 1 1 1	0 1 1 0	0 0 1 0	0 0 0 0
Q_4	0 1 1 0	0 0 1 0	0 0 1 0
	0 1 1 0	0 0 1 0	0 0 1 0
	0 1 1 0	0 0 1 0	0 0 0 0
Q_2			

$$D_1 \oplus Q_1 = 1$$

$$D_4 \oplus Q_4 = Q_1 Q_2$$

$$D_2 \oplus Q_2 = Q_1$$

$$D_8 \oplus Q_8 = Q_1 Q_2 Q_4$$

"count up"
to 5111

Implementing a 4-bit Binary Counter

Theorem:

$$X \oplus 1 \oplus 1 = X$$

If $X \oplus Y = Z$ then $X = Y \oplus Z$

Proof:

$X \oplus Y = Z$ (assumed)

$Y \oplus X \oplus Y = Y \oplus Z$ (apply $Y \oplus$)

$Y \oplus Y \oplus X = Y \oplus Z$ (\oplus commutes)

$X = Y \oplus Z$ ($y \oplus y \oplus x = x$)

So if

$$D_1 \oplus Q_1 = 1$$

$$D_2 \oplus Q_2 = Q_1$$

$$D_4 \oplus Q_4 = Q_1 Q_2$$

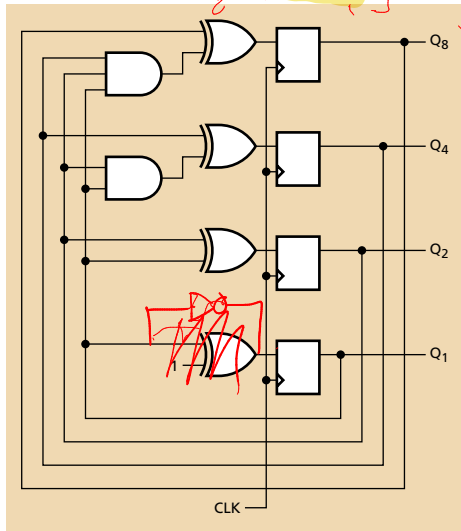
$$D_8 \oplus Q_8 = Q_1 Q_2 Q_4$$

$$D_1 = Q_1 \oplus 1$$

$$D_2 = Q_2 \oplus Q_1$$

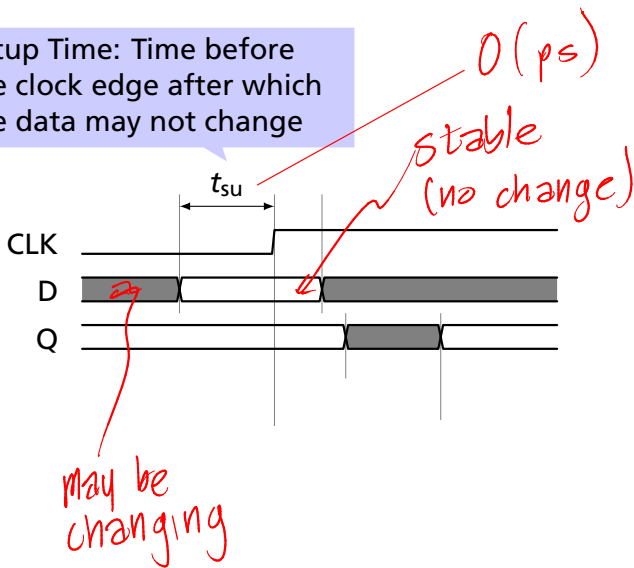
$$D_4 = Q_4 \oplus Q_1 Q_2$$

$$D_8 = Q_8 \oplus Q_1 Q_2 Q_4$$



Flip-Flop Timing

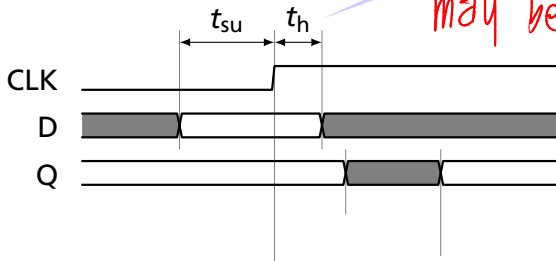
Setup Time: Time before the clock edge after which the data may not change



Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

Hold Time: Time after the clock edge after which the data may change

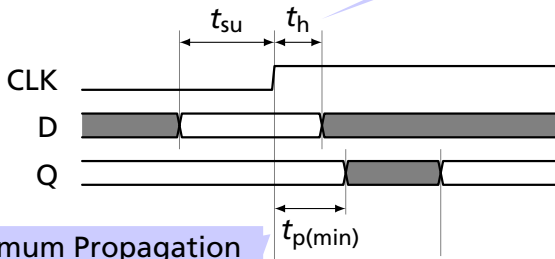


may be negative!

Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

Hold Time: Time after the clock edge after which the data may change

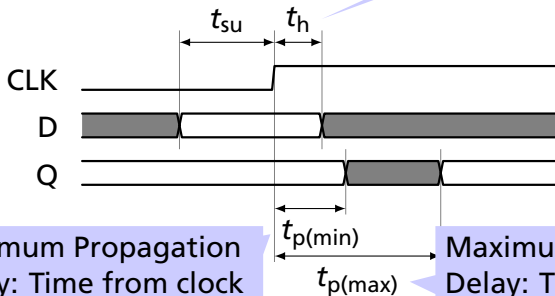


Minimum Propagation Delay: Time from clock edge to when Q might start changing

Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

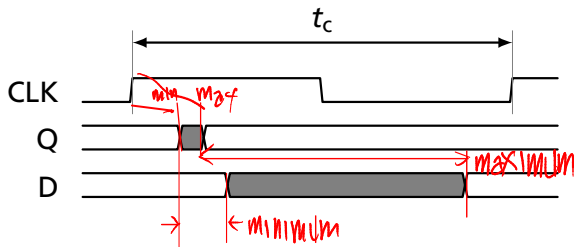
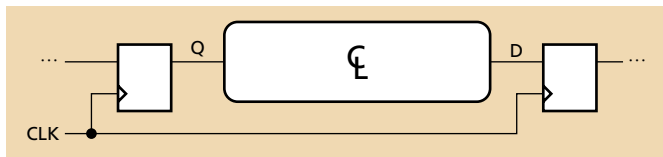
Hold Time: Time after the clock edge after which the data may change



Minimum Propagation Delay: Time from clock edge to when Q might start changing

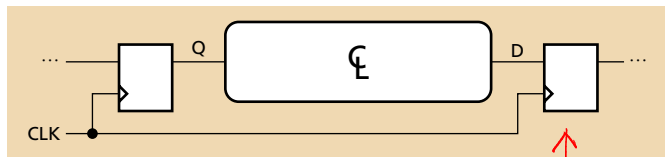
Maximum Propagation Delay: Time from clock edge to when Q guaranteed stable

Timing in Synchronous Circuits

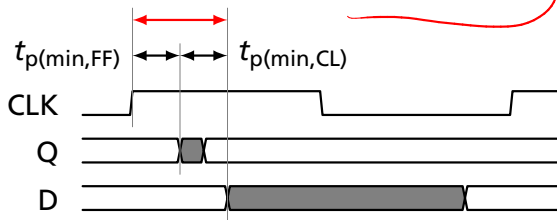


t_c : Clock period. E.g., 10 ns for a 100 MHz clock

Timing in Synchronous Circuits

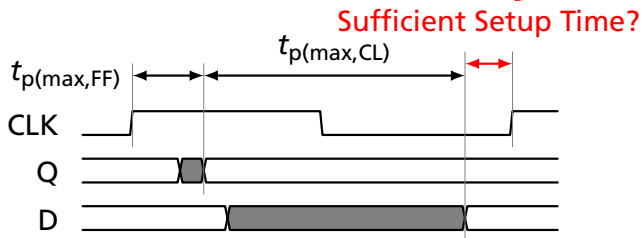
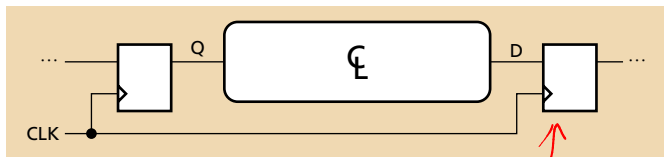


Sufficient Hold Time?



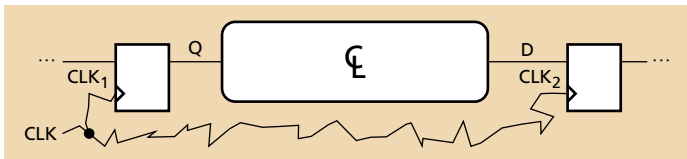
Hold time constraint: how soon after the clock edge can D start changing? Min. FF delay + min. logic delay

Timing in Synchronous Circuits

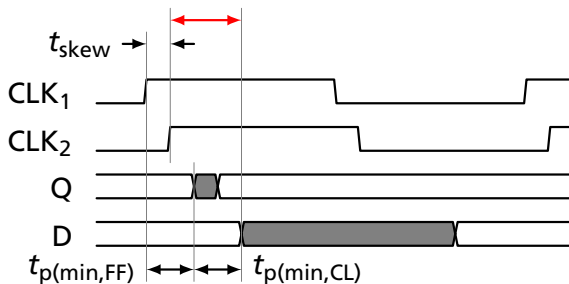


Setup time constraint: when before the clock edge is D guaranteed stable? Max. FF delay + max. logic delay

Clock Skew: What Really Happens

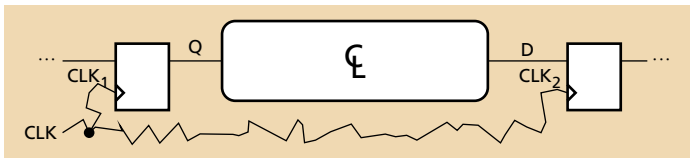


Sufficient Hold Time?

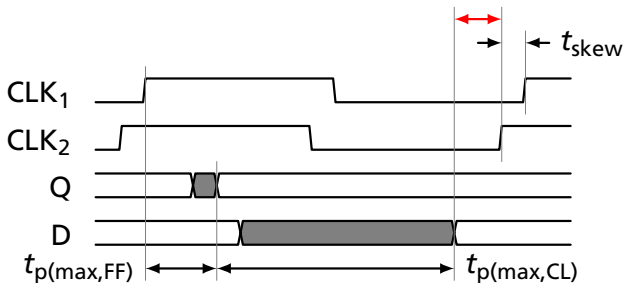


CLK_2 arrives late: clock skew reduces hold time

Clock Skew: What Really Happens



Sufficient Setup Time?



CLK₂ arrives early: clock skew reduces setup time