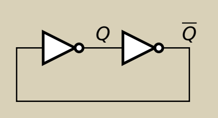
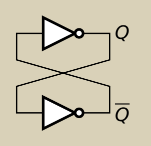
Fundamentals of Computer Systems

Sequential Logic

Harris and Harris
Chapter 3.1-3.3,3.5

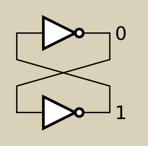
Bistable Elements

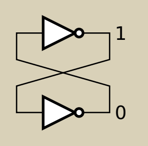




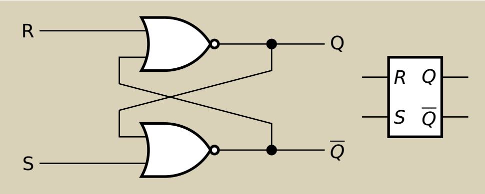
Equivalent circuits; right is more traditional.

Two stable states:



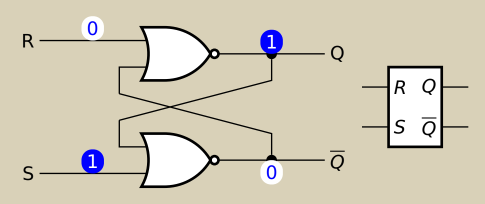


RS Latch



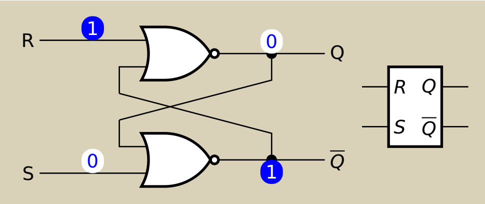
R	S	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		

RS Latch



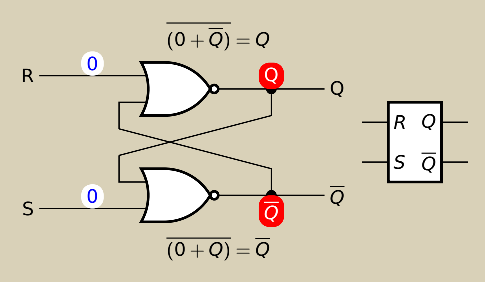
R	S	Q	$\overline{Q}$	
0	0			
0	1	1	0	Set ($Q = 1$)
1	0			
1	1			

RS Latch



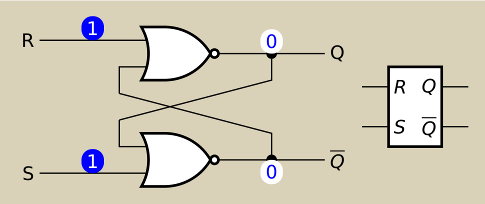
R	S	Q	$\overline{Q}$	
0	0			
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1			

RS Latch



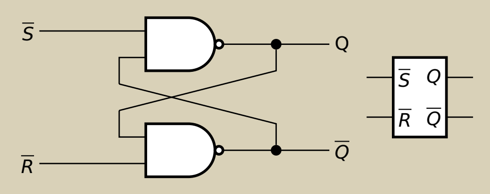
R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Hold previous value
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1			

RS Latch



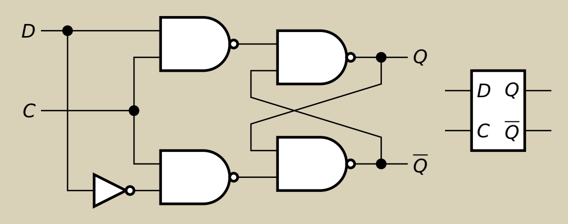
R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Hold previous value
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1	0	0	Bad. Do not use.

$\overline{R}\,\overline{S}$ Latch



$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$	
0	0	1	1	Bad. Do not use.
0	1	0	1	Reset ($Q = 0$)
1	0	1	0	Set ($Q = 1$)
1	1	Q	$\overline{Q}$	Hold previous value

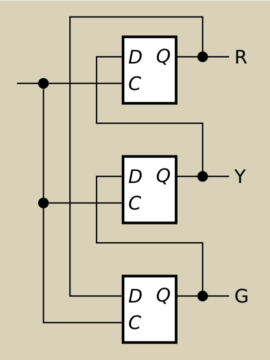
D Latch



C	D	Q	$\overline{Q}$
0	X	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

A Challenge: Build a traffic light controller

Want the lights to cycle green-yellow-red.

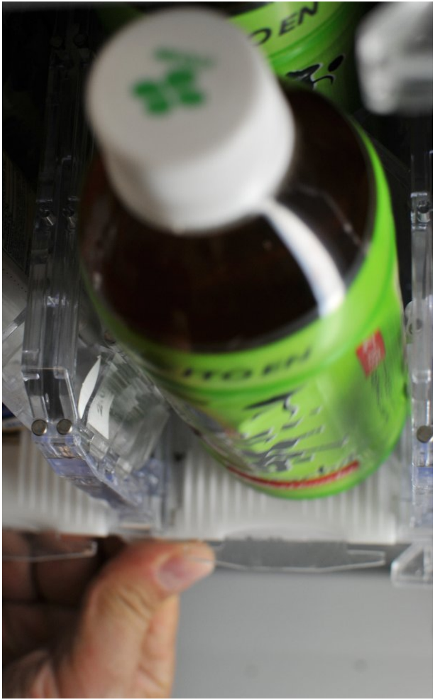


Does this work?


ITO EN


ITO EN

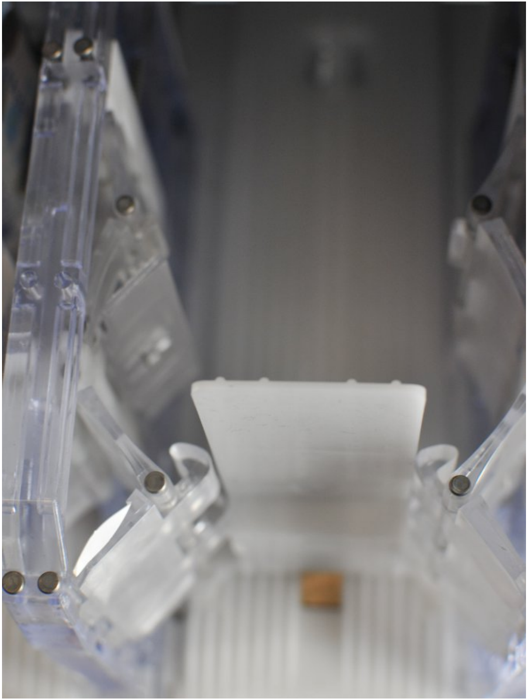

ITO EN

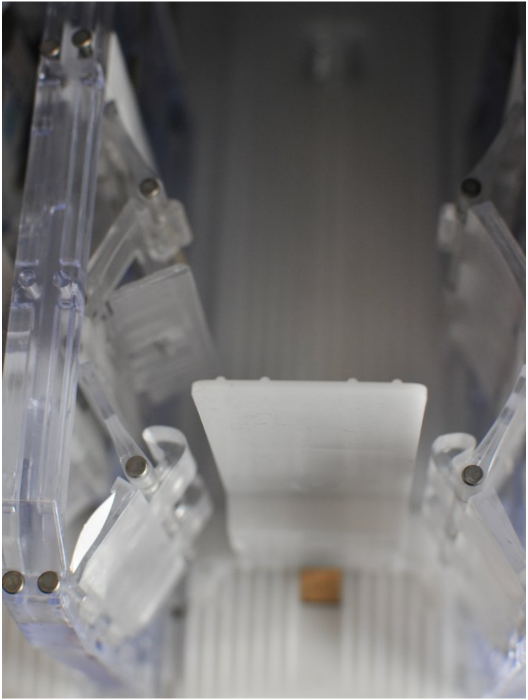


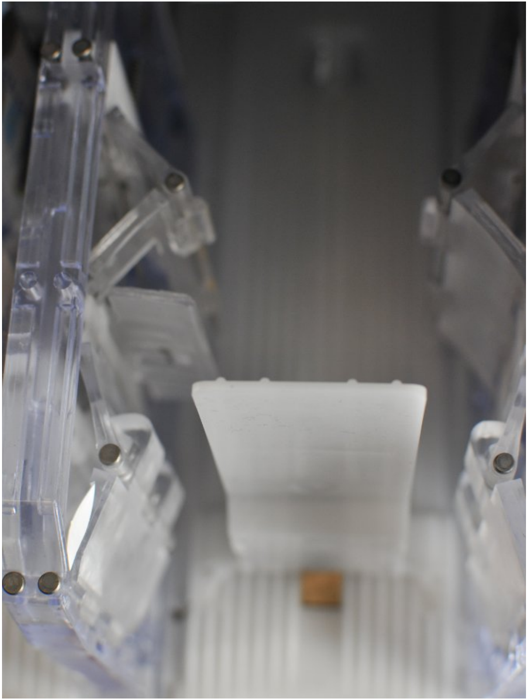

ITO EN
お〜いお茶
おいしさは香り

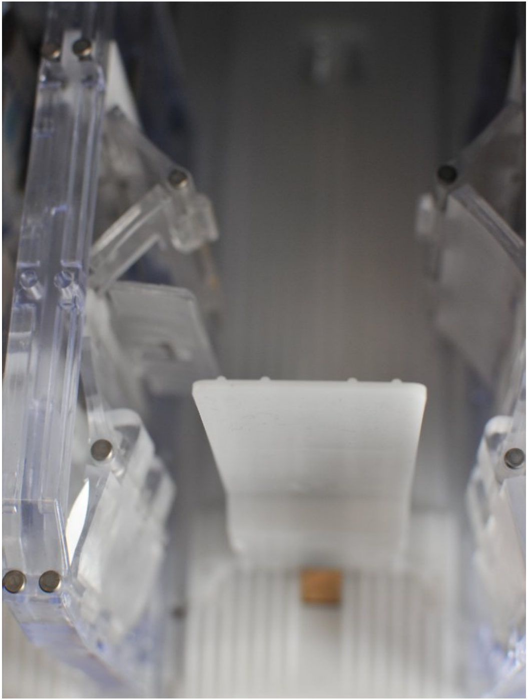

ITO EN
おいしさは香り

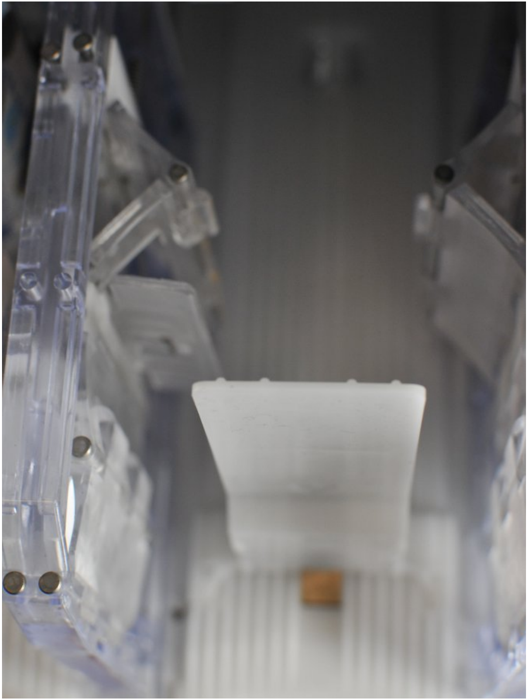

ITO EN
おいしさは香り



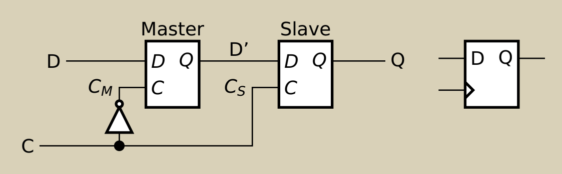


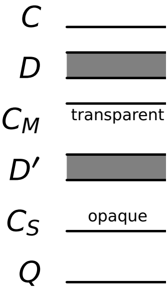




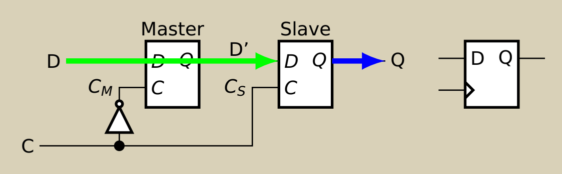


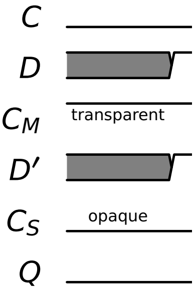
Positive-Edge-Triggered D Flip-Flop



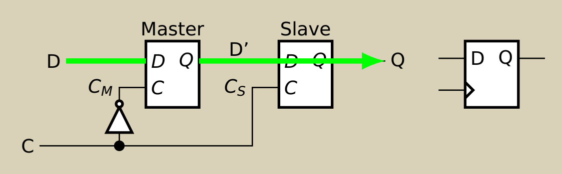


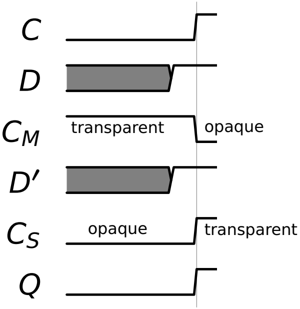
Positive-Edge-Triggered D Flip-Flop



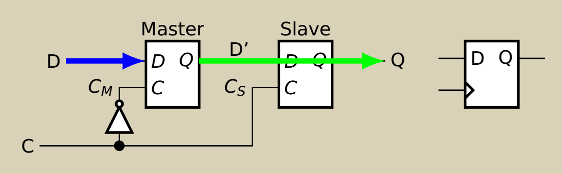


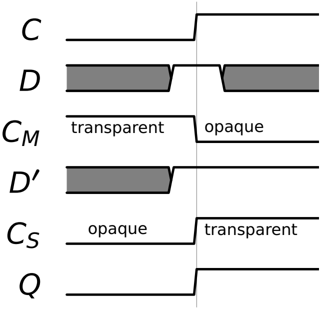
Positive-Edge-Triggered D Flip-Flop



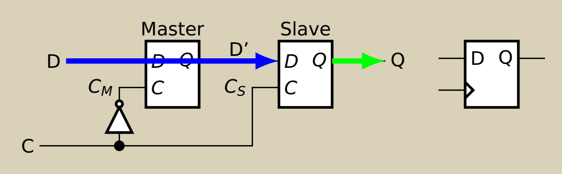


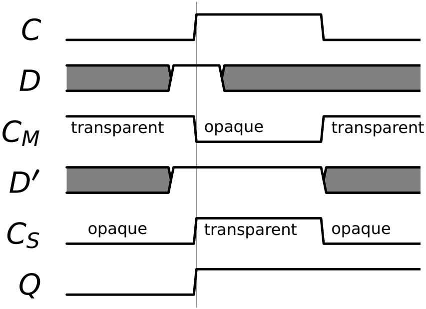
Positive-Edge-Triggered D Flip-Flop



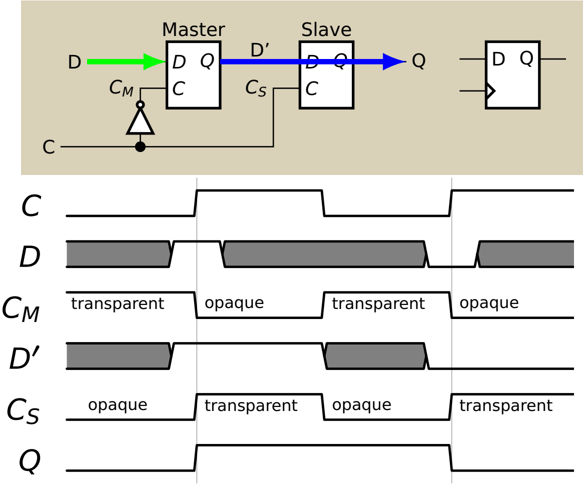


Positive-Edge-Triggered D Flip-Flop



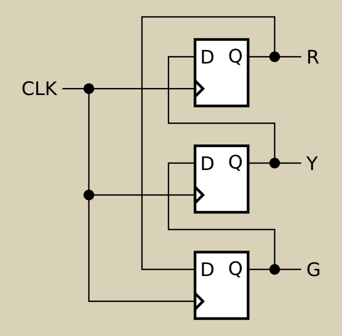


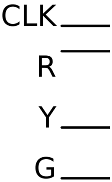
Positive-Edge-Triggered D Flip-Flop



The Traffic Light Controller: A second try

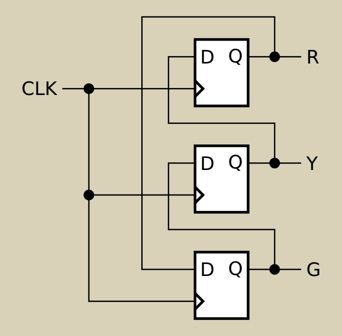
Let's try this again with D flip-flops.

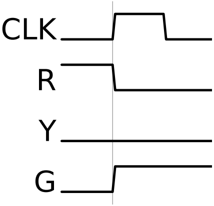




The Traffic Light Controller: A second try

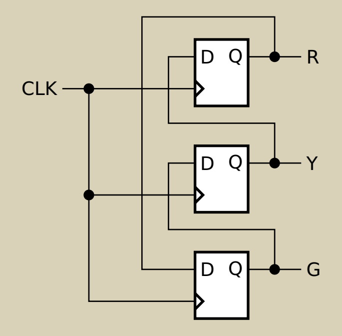
Let's try this again with D flip-flops.

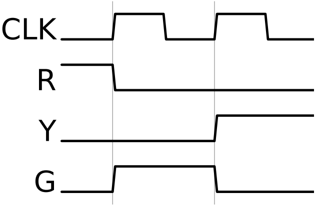




The Traffic Light Controller: A second try

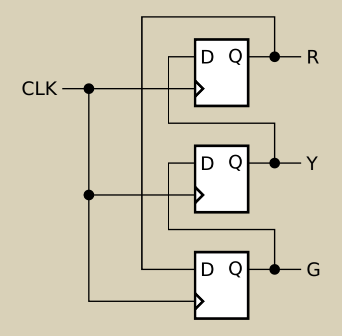
Let's try this again with D flip-flops.

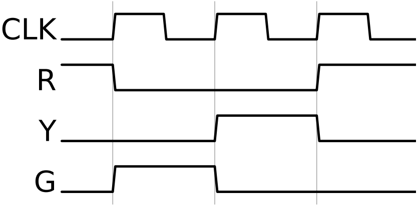




The Traffic Light Controller: A second try

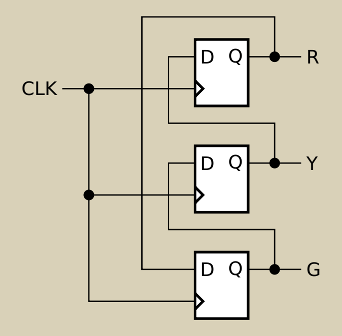
Let's try this again with D flip-flops.

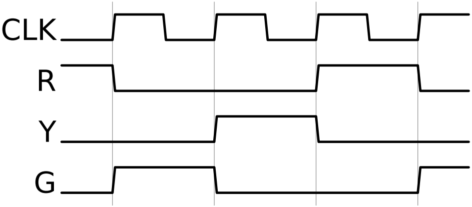




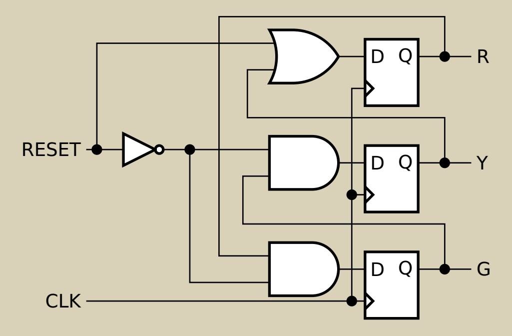
The Traffic Light Controller: A second try

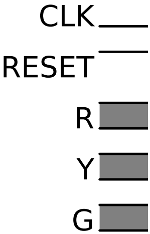
Let's try this again with D flip-flops.



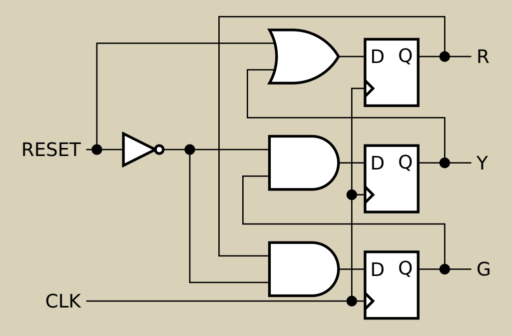


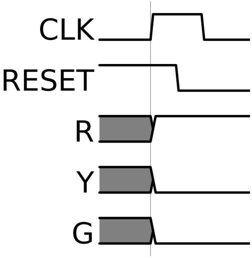
The Traffic Light Controller with Reset



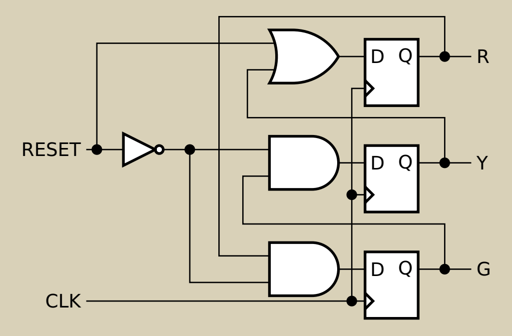


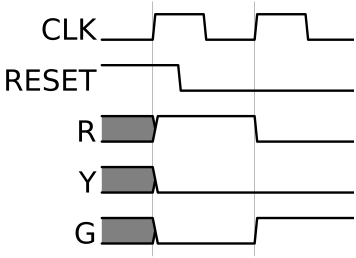
The Traffic Light Controller with Reset



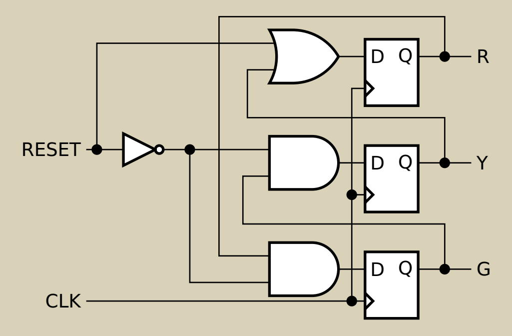


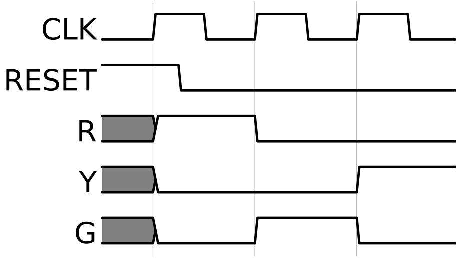
The Traffic Light Controller with Reset



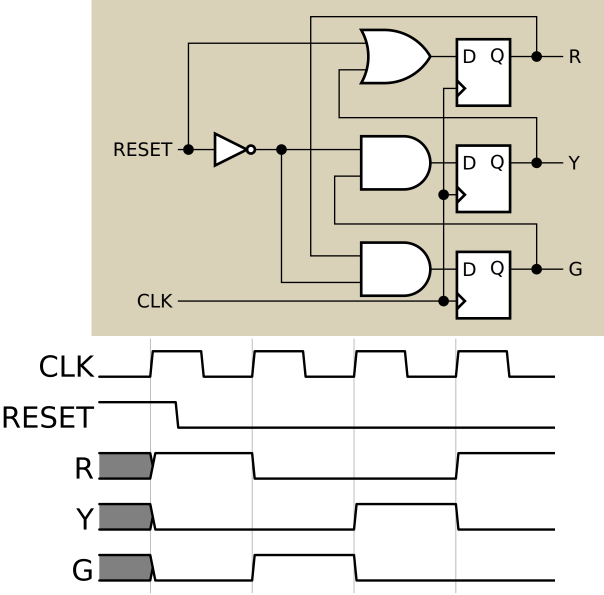


The Traffic Light Controller with Reset

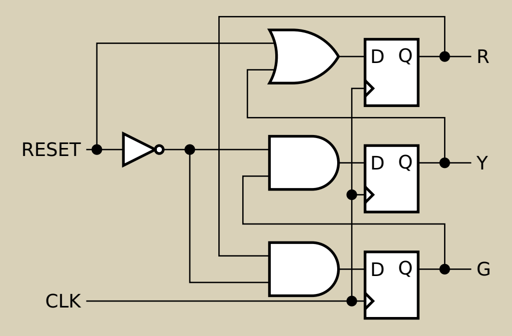


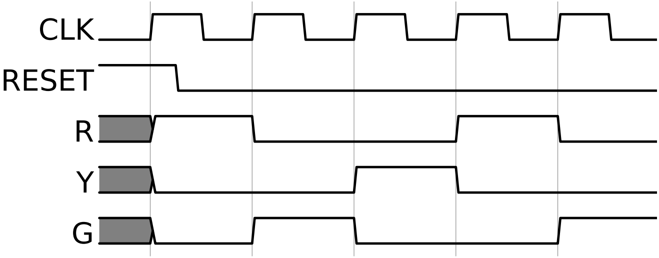


The Traffic Light Controller with Reset

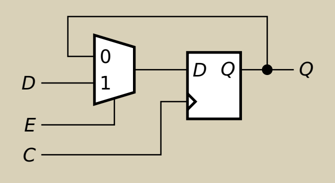


The Traffic Light Controller with Reset

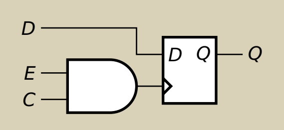




D Flip-Flop with Enable

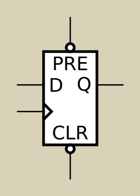


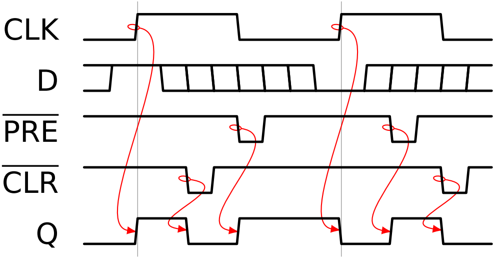
C	E	D	Q
$\uparrow$	0	X	Q
$\uparrow$	1	0	0
$\uparrow$	1	1	1
0	X	X	Q
1	X	X	Q



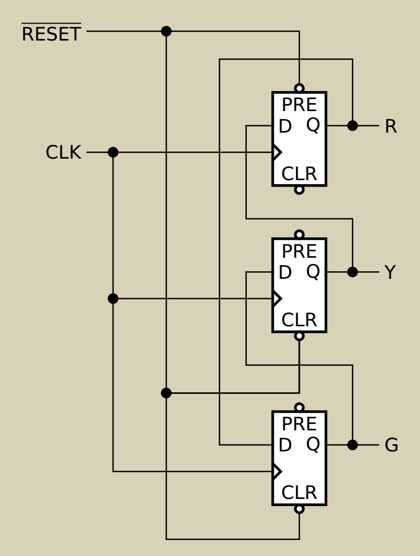
What's wrong with this solution?

Asynchronous Preset/Clear





The Traffic Light Controller w/ Async. Reset

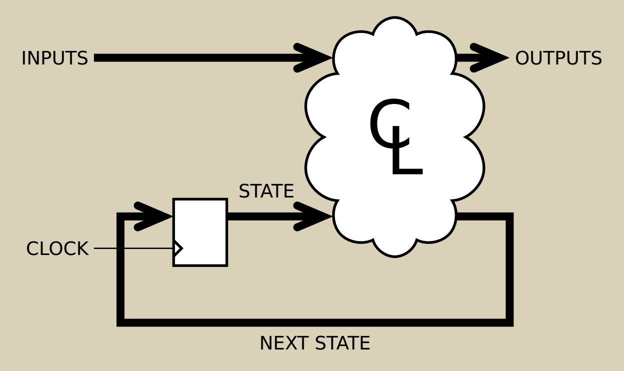


The Synchronous Digital Logic Paradigm

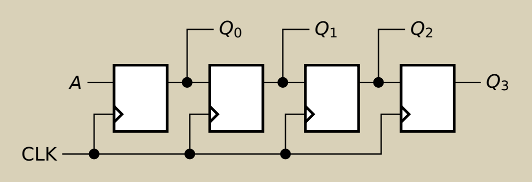
Gates and D flip-flops only

Each flip-flop driven by the same clock

Every cyclic path contains at least one flip-flop

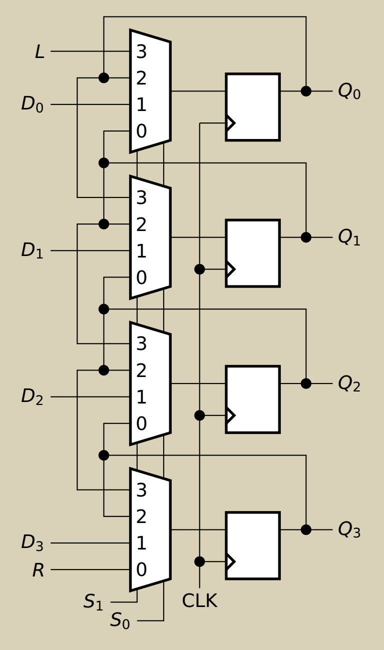


Cool Sequential Circuits: Shift Registers



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

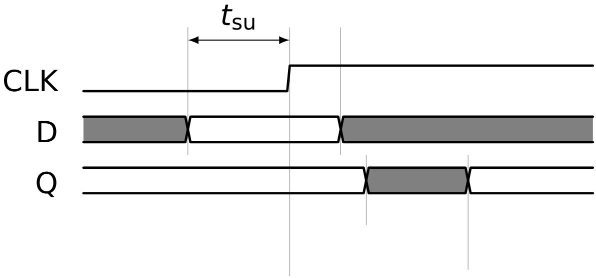
Cool Sequential Circuits: Counters

Cycle through sequences of numbers, e.g.,

$$00 \to 01 \to 10 \to 11$$

Flip-Flop Timing

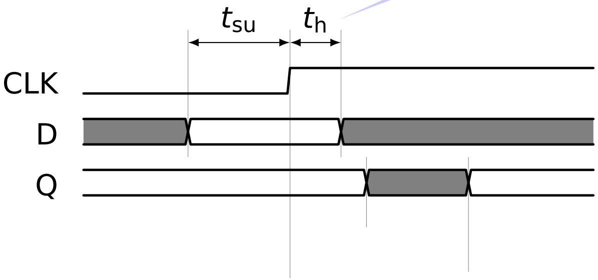
Setup Time: Time before the clock edge after which the data may not change



Flip-Flop Timing

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

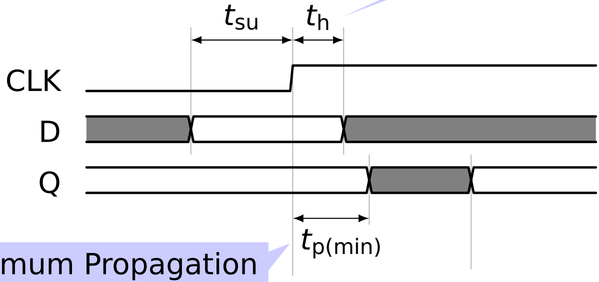
Hold Time: Time after the clock edge after which the data may change



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

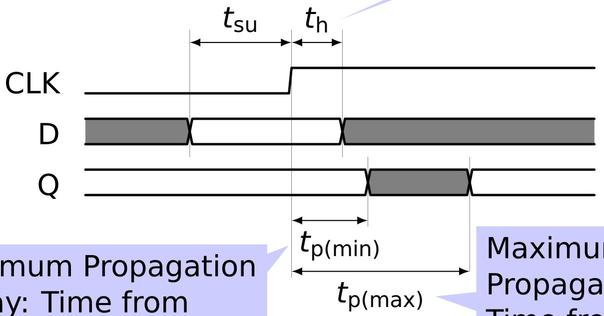


Minimum Propagation Delay: Time from clock edge to when Q might start changing

Flip-Flop Timing

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

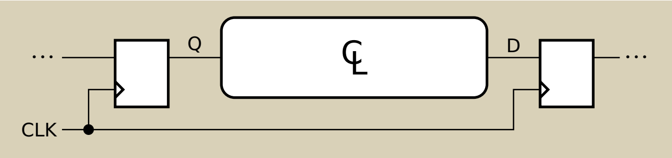
Hold Time: Time after the clock edge after which the data may change

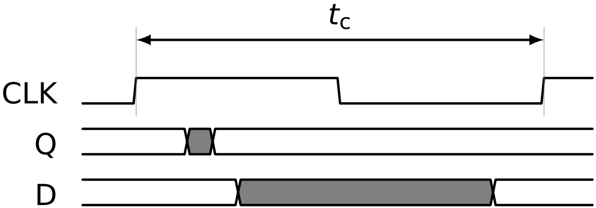


Minimum Propagation Delay: Time from clock edge to when Q might start changing

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

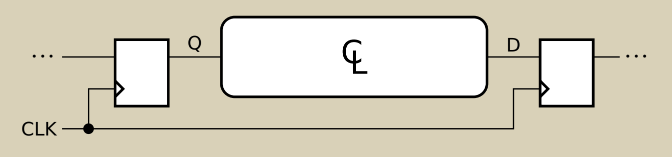
Timing in Synchronous Circuits

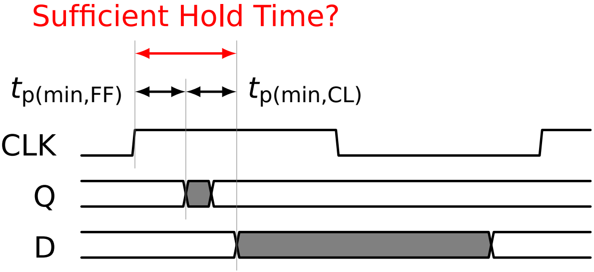




t_c: Clock period. E.g., 10 ns for a 100 MHz clock

Timing in Synchronous Circuits

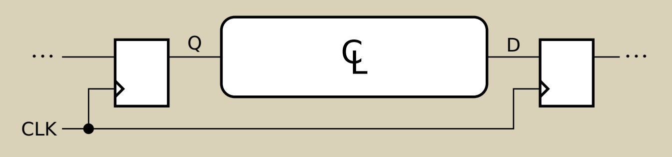


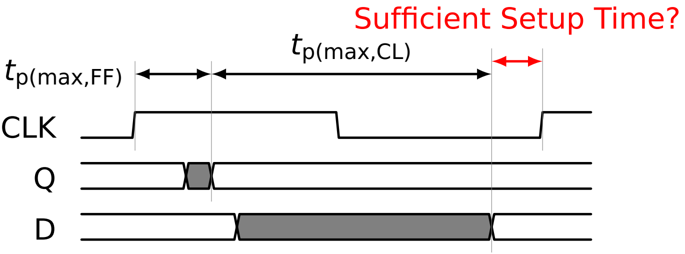


Hold time constraint: how soon after the clock edge is D liable to start changing?

Min. FF delay + min. logic delay

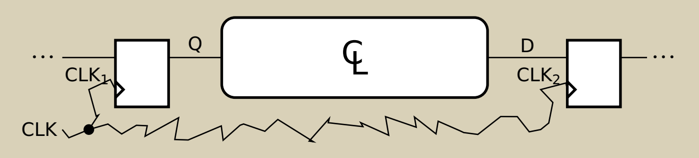
Timing in Synchronous Circuits

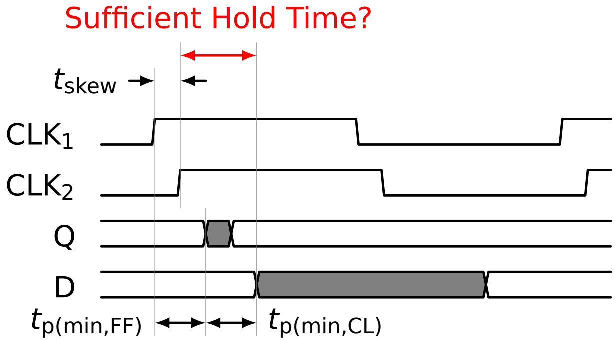




Setup time constraint: when before the clock edge is D guaranteed to have stabilized?
Max. FF delay + max. logic delay

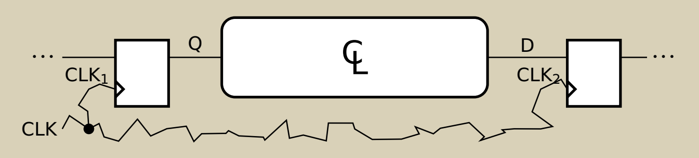
Clock Skew: What Really Happens

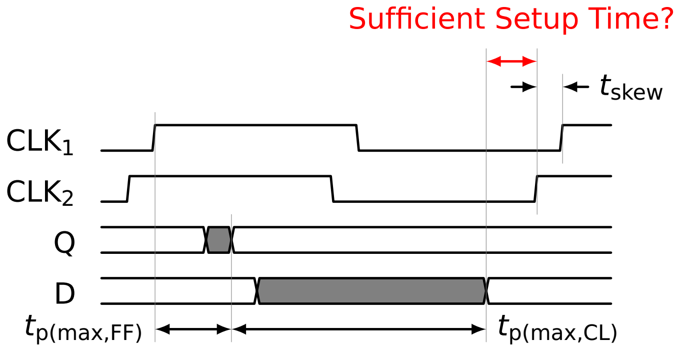




CLK$_2$ arrives late, creating potential hold time violation

Clock Skew: What Really Happens





CLK_2 arrives early, creating potential setup time violation