Computer Organization and Design

Memories and Flip Flops

Henry Fuchs

Slides adapted from Montek Singh, who adapted them
from Leonard McMillan and from Gary Bishop
Back to McMillan & Chris Terman, MIT 6.004 1999

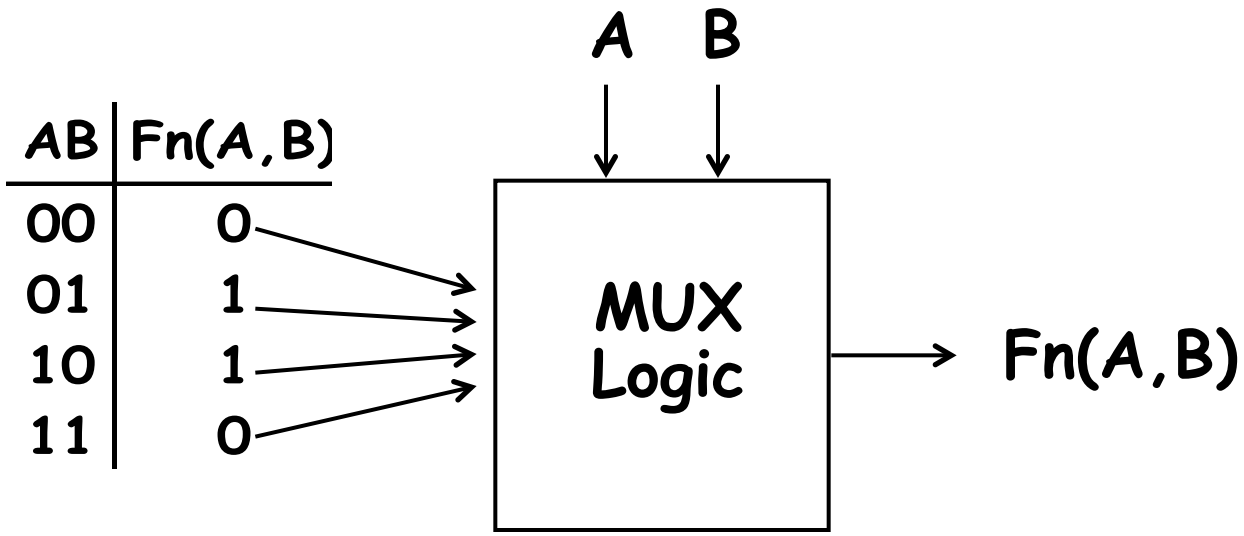
Tuesday, March 31, 2015

Lecture 13

Topics

- Memory Arrays
- Transparent Latches
- Edge-triggered Registers

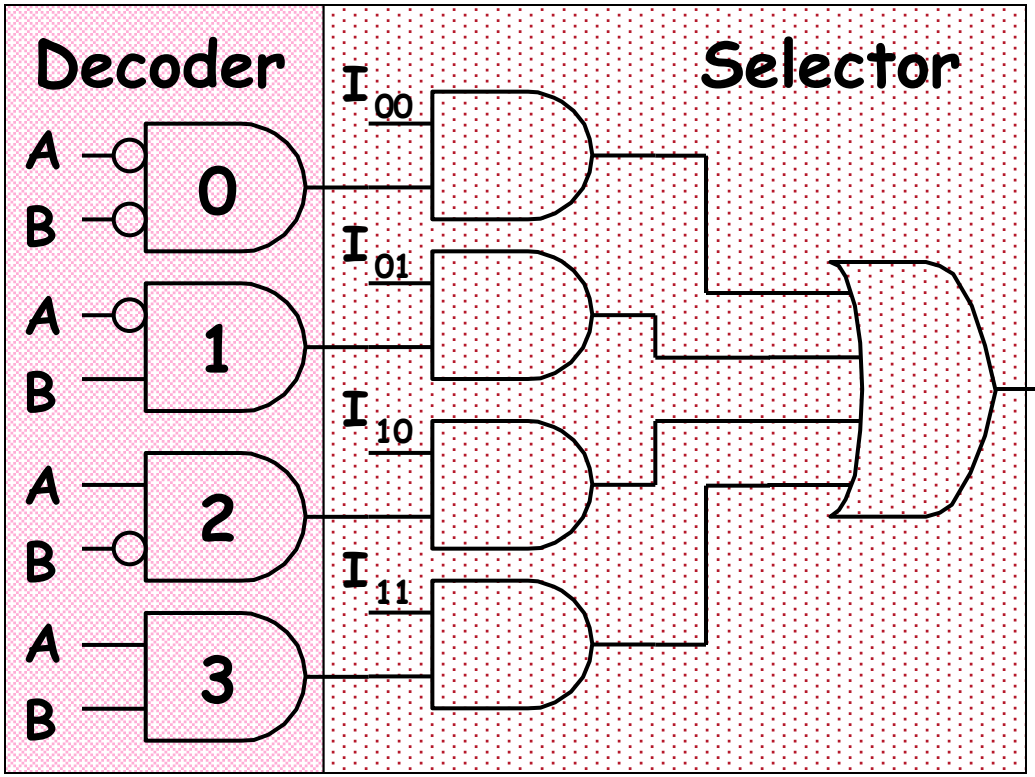
Memory as a Lookup Table



- A multiplexer to implement a lookup table:
  - Remember that, in theory, we can build any 1-output combinational logic block with multiplexers
  - For an N-input function we need a 2^N input multiplexer
- BIG Multiplexers?
  - How about 10-input function? 20-input?

A Mux's Guts

A decoder generates all possible product terms for a set of inputs



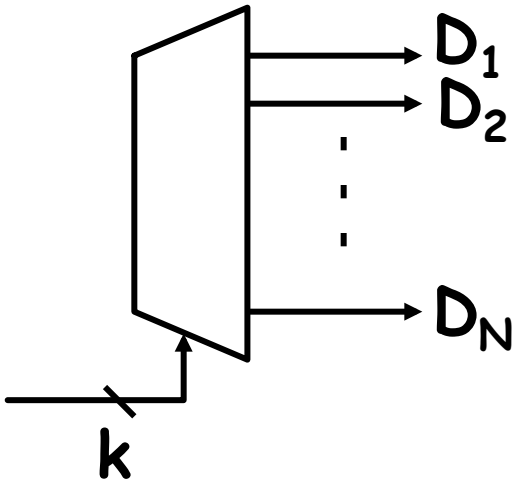
Multiplexers can be partitioned into two sections.

A DECODER that identifies the desired input,and

a SELECTOR that enables that input onto the output.

Hmmm, by sharing the decoder part of the logic, MUXs could be adapted to make lookup tables with any number of outputs

A New Combinational Device



DECODER:

k SELECT inputs,

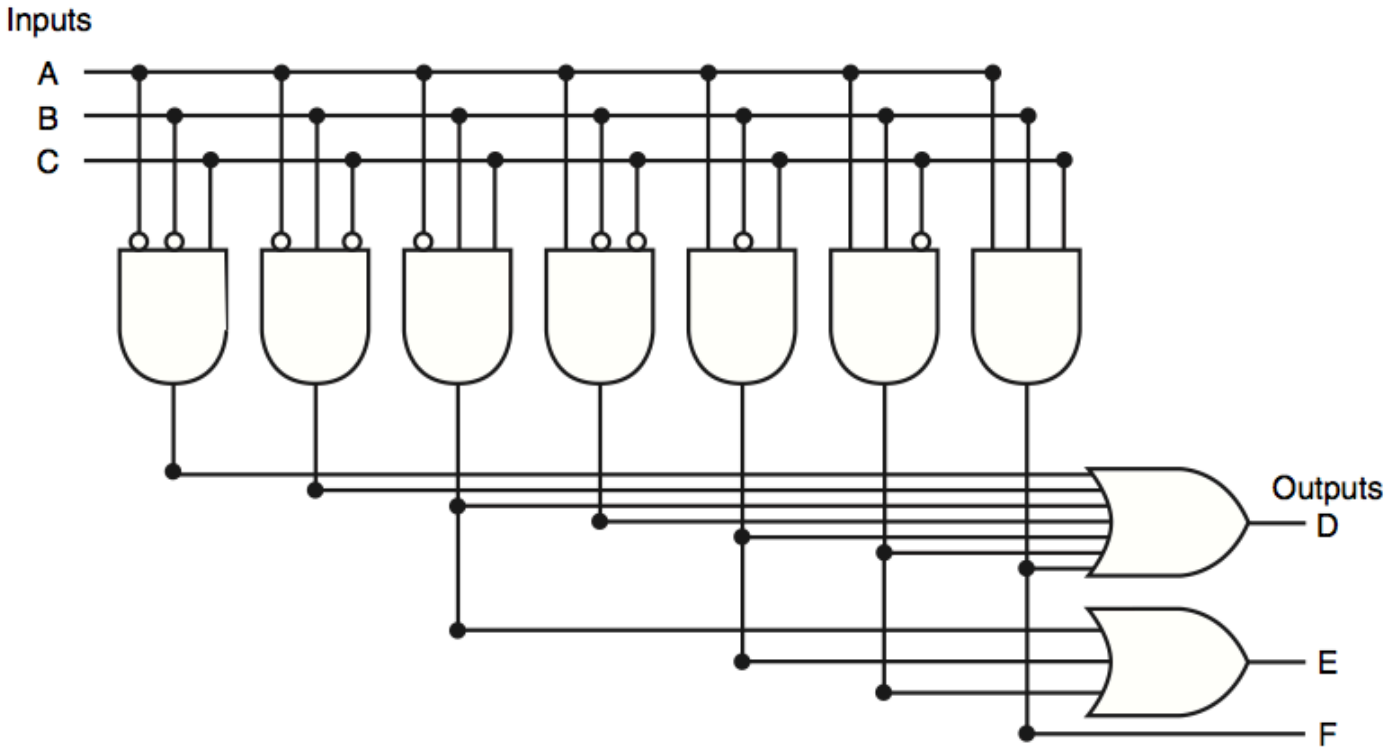
N = 2^k DATA OUTPUTs.

Selected D_j HIGH;
all others LOW.

NOW, we are well on our way to building a general purpose table-lookup device.

We can build a 2-dimensional ARRAY of decoders and selectors as follows ...

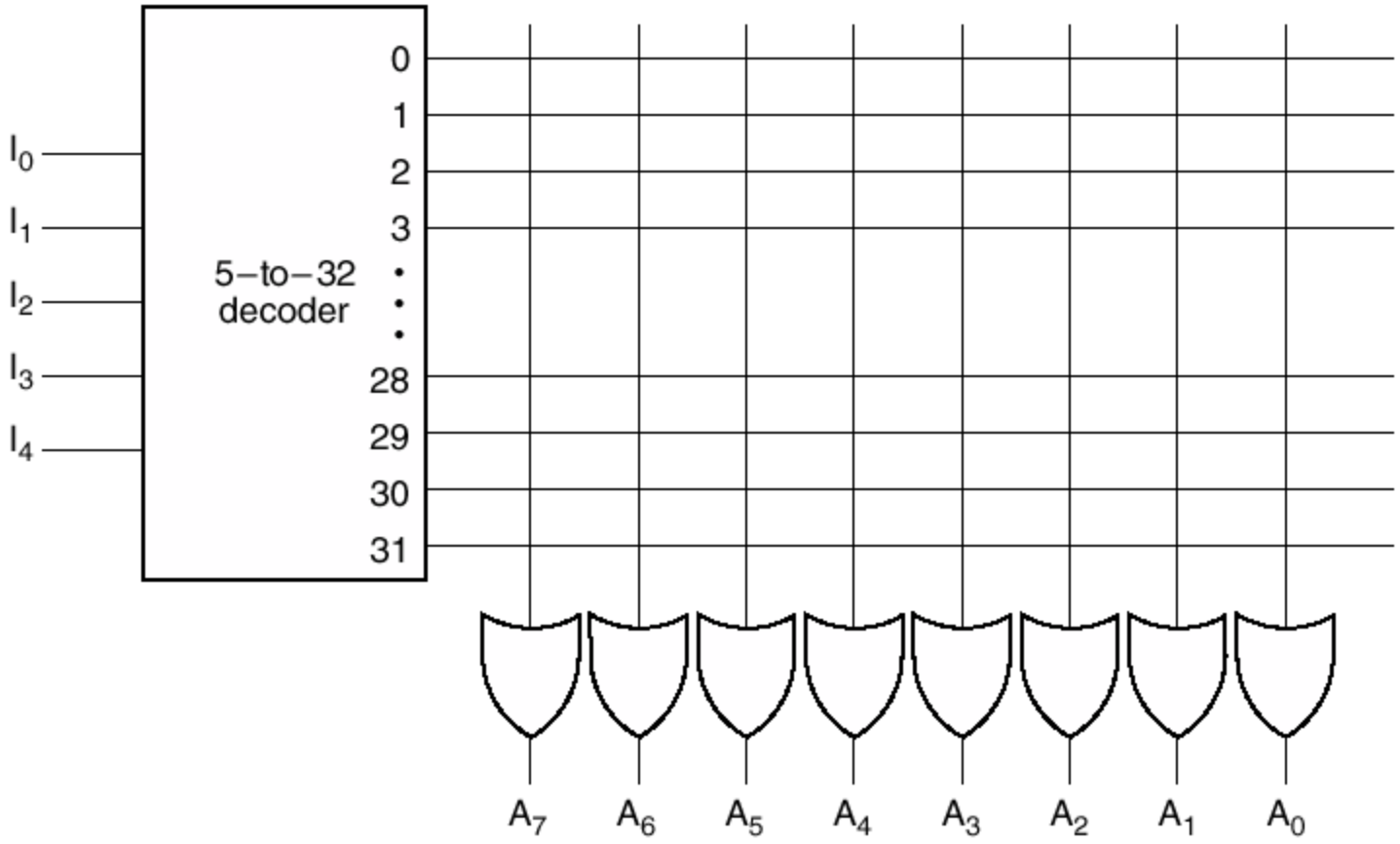
Shared Decoding Logic



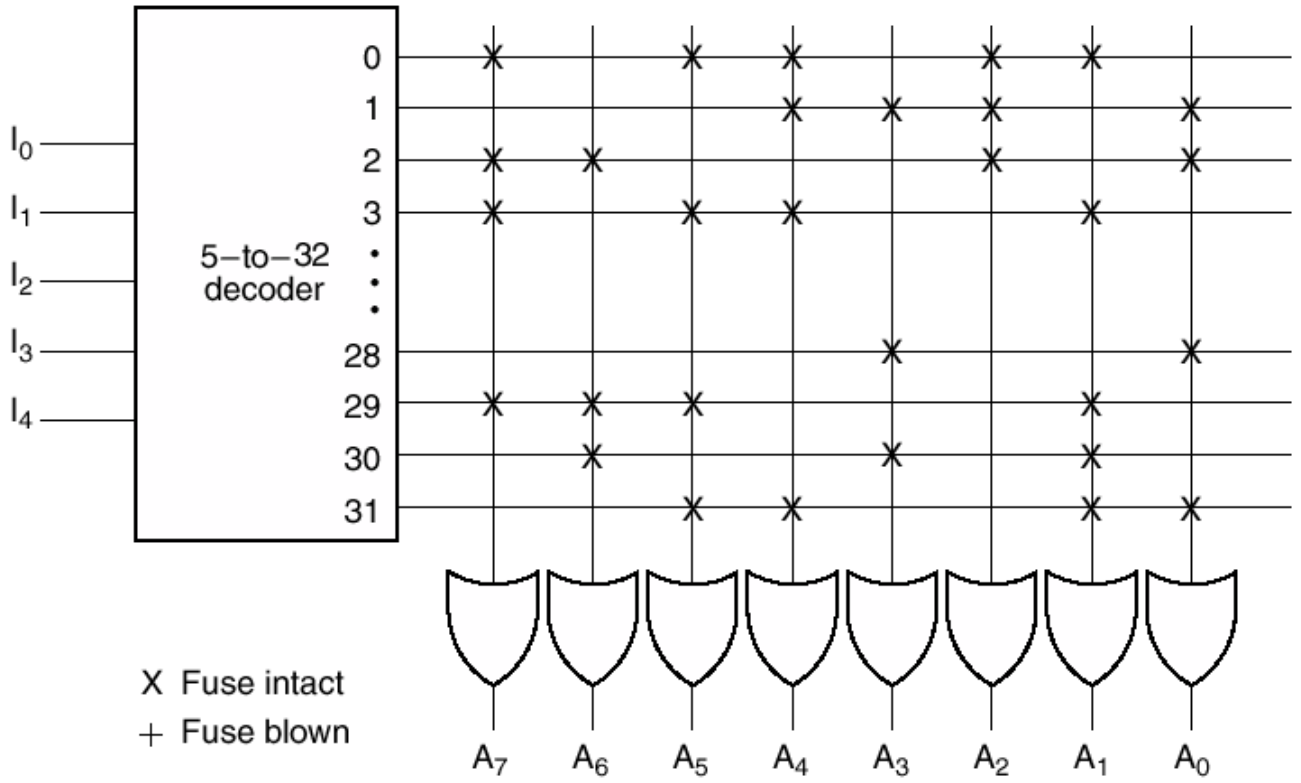
We can build a general purpose “table-lookup” device called a Read-Only Memory (ROM), from which we can implement any truth table and, thus, any combinational device

A Full ROM, Unprogrammed

Where wires cross, there is a fuse.
OR gate represent OR of all column.



A ROM, Programmed



ROM Truth Table (Partial)

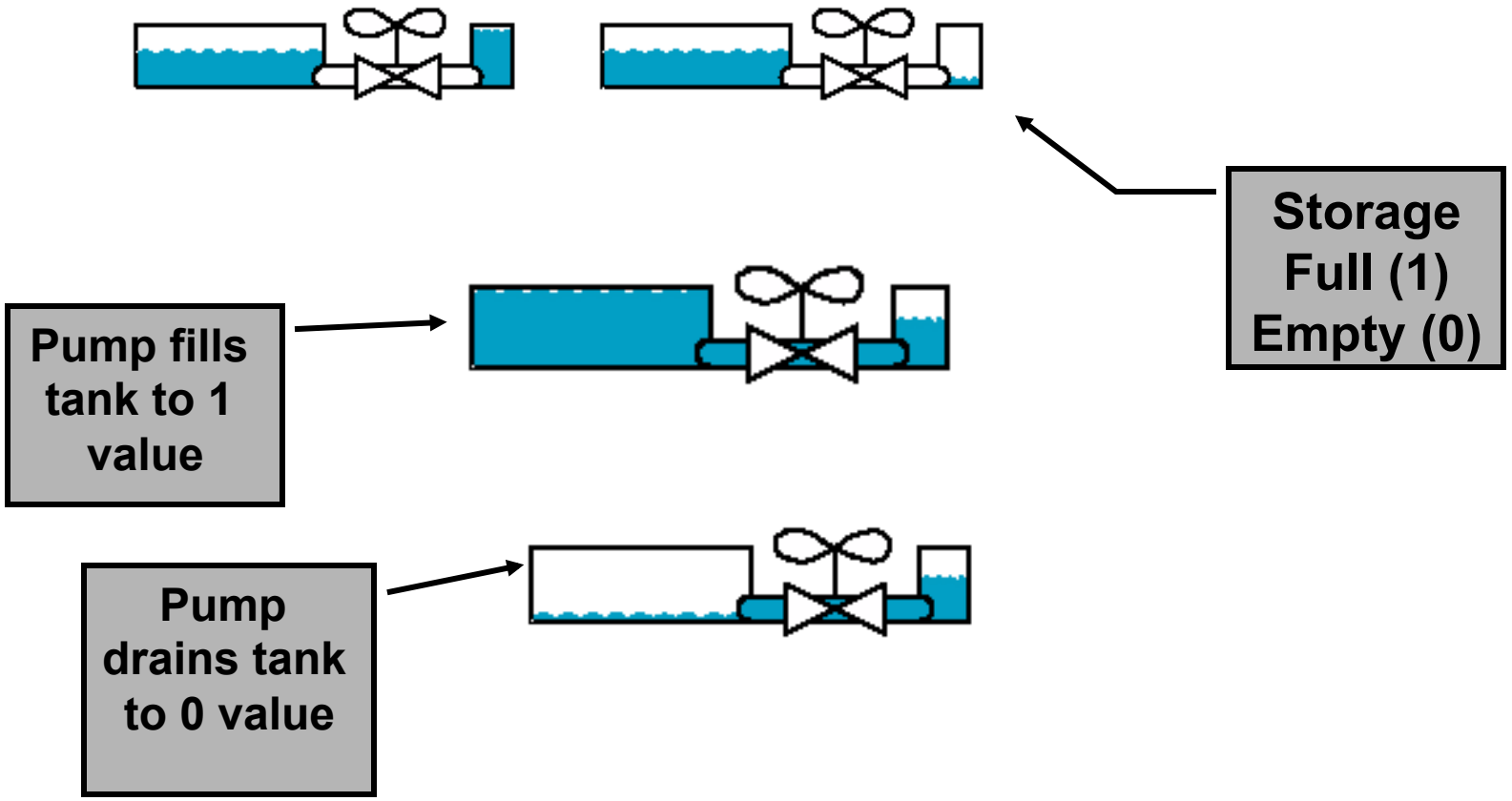
Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		.							.			
		.							.			
		.							.			
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Logic According to ROMs

- ROMs ignore the structure of combinational functions ...
  - Size, layout, and design are independent of function
  - Any Truth table can be "programmed" by minor reconfiguration:
    - Metal layer (masked ROMs)
    - Fuses (Field-programmable PROMs)
    - Charge on floating gates (EPROMs)
    - ... etc.

- Model: LOOK UP value of function in truth table...
  - Inputs: "ADDRESS" of a Truth Table entry
  - ROM SIZE = # TT entries...
    - ... for an N-input boolean function, size = 2^N x #outputs

Hydraulic Analogy of DRAM



Reading

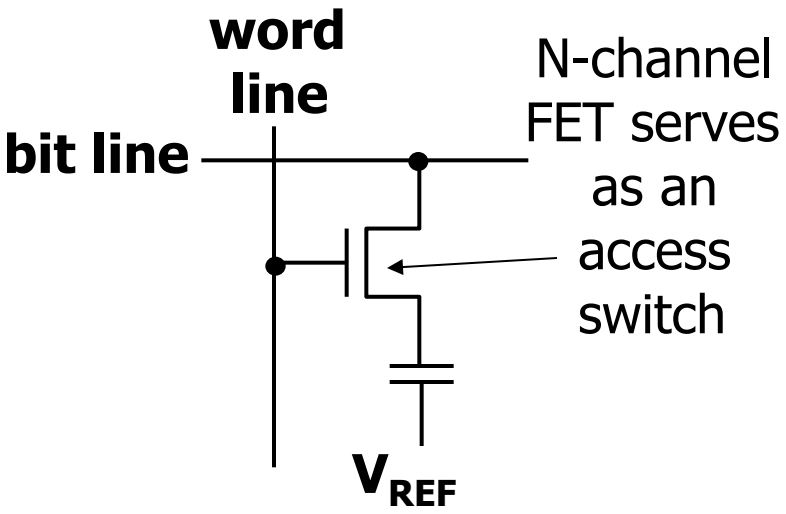
Outside water begins at intermediate level (black wavy line)

Tank had a 1 value – raises water level

Tank had a 0 value – lowers water level

Analog Storage: Using Capacitors

- We've chosen to encode information using voltages and we know from physics that we can "store" a voltage as "charge" on a capacitor



To write:
Drive bit line, turn on access fet, force storage cap to new voltage

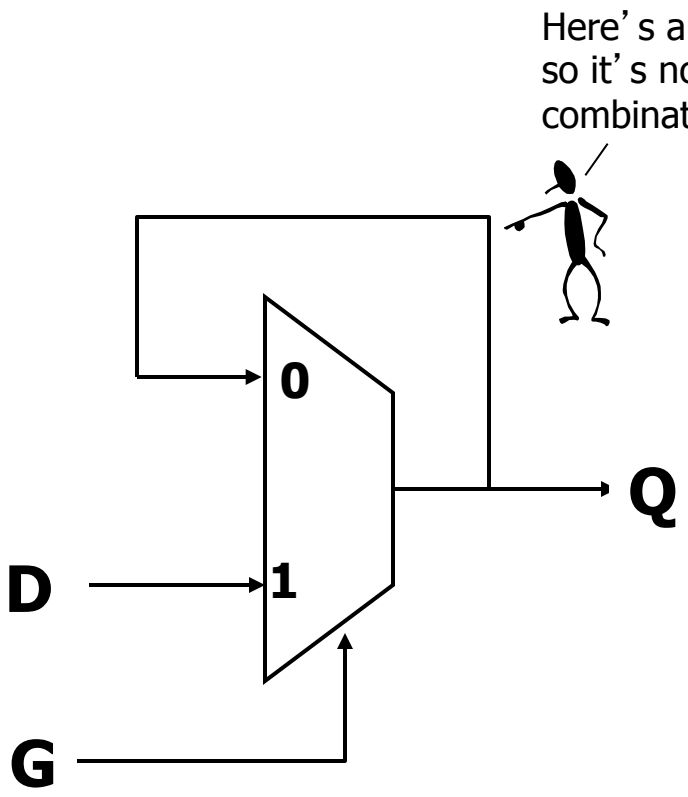
To read:
precharge bit line, turn on access fet, detect (small) change in bit line voltage

- Pros:
  - compact!
- Cons:
  - it leaks! ⇒ refresh
  - complex interface
  - reading a bit, destroys it
    - (you have to rewrite the value after each read)
  - it's NOT a digital circuit

This storage circuit is the basis for commodity DRAMs

A "Digital" Storage Element

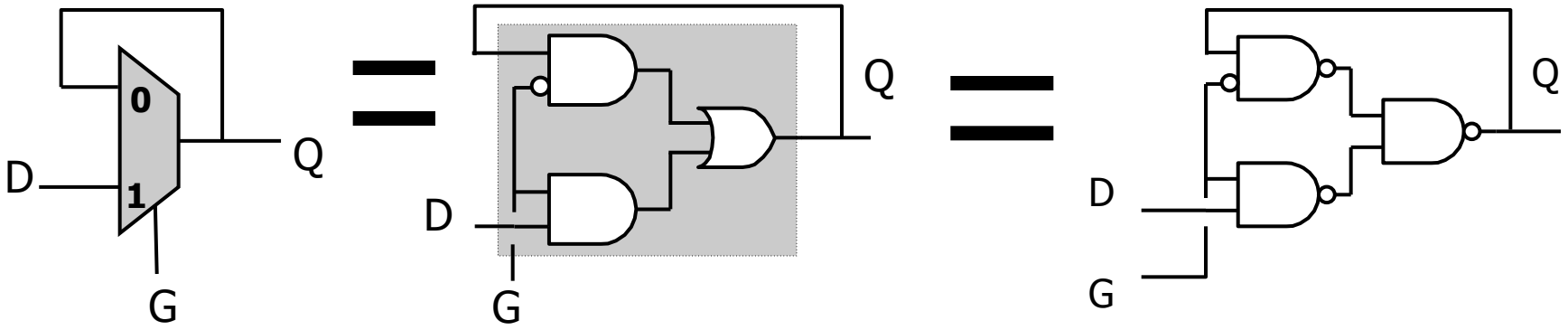
- It's also easy to build a settable DIGITAL storage element (called a latch) using a MUX and FEEDBACK:



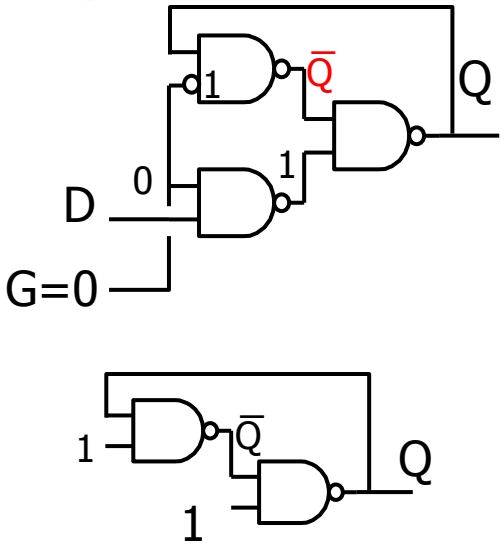
"state" signal appears as both input and output

G	D	Q_{IN}	Q_{OUT}	
0	--	0	0	Q stable
0	--	1	1	
1	0	--	0	Q follows D
1	1	--	1	

Looking Into the Mux Circuit



What is Q if G=0 ?

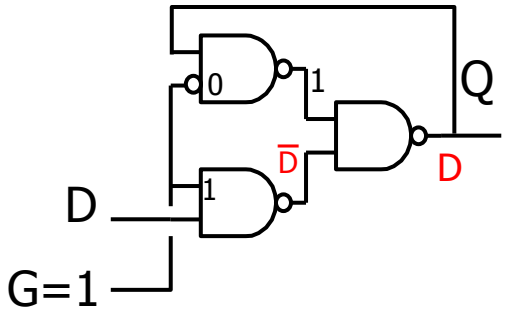


Q = was Q was before (since when?)

What if D changes while G is 0?

- Q doesn't change

What is Q if G=1 ?

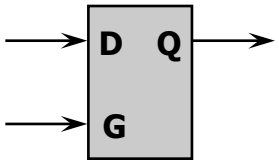


Q = D !

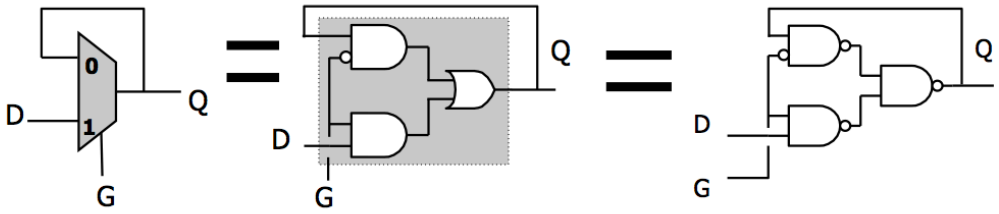
What if D changes while G =1?

- Q keeps changing to D

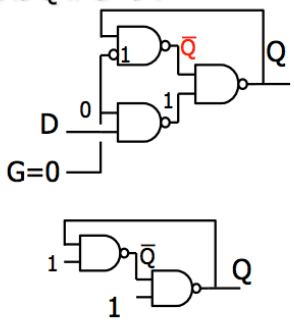
Static D Latch



Looking Into the Mux Circuit



What is Q if G=0 ?

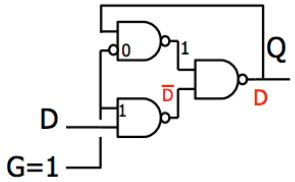


Q = was Q was before (since when?)

What if D changes while G is 0?

- Q doesn't change

What is Q if G=1 ?

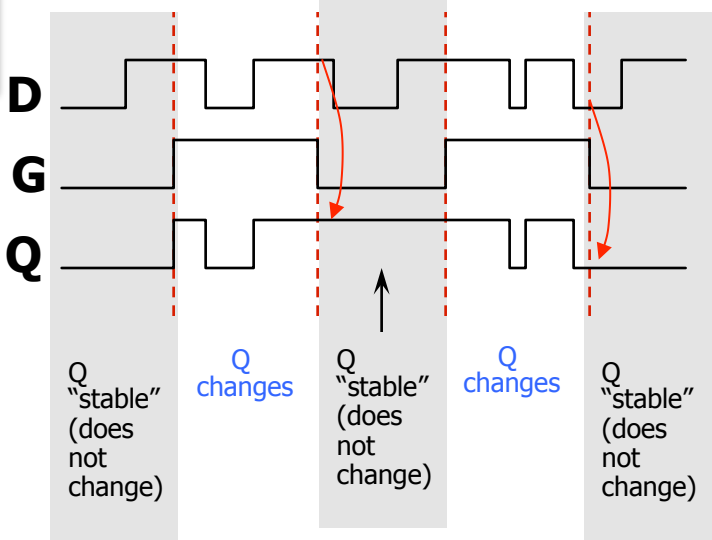


Q = D !

What if D changes while G =1?

- Q keeps changing to D

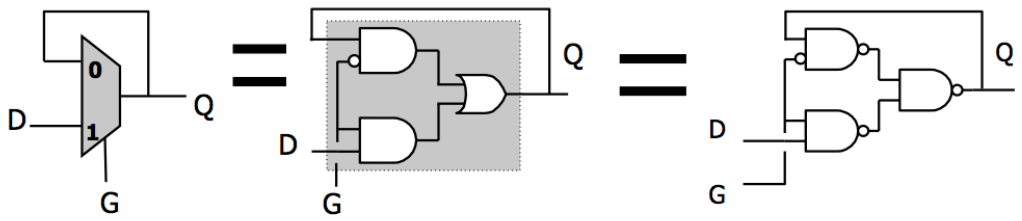
- Q will take on value of D the last instant that G is 1, just before G goes to 0.
- "Static" means latch will hold data (i.e., value of Q) while G is 0, however long that may be.



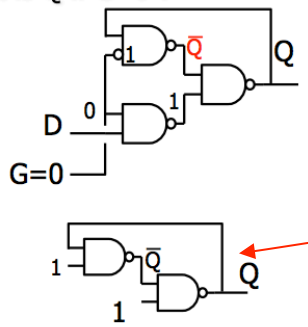
"Bi-Stable Storage Element" = One bit of storage

Looking Into the Mux Circuit

15 / 40



What is Q if G=0 ?

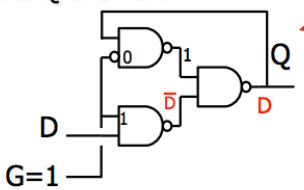


Q = was Q was before (since when?)

What if D changes while G is 0?

- Q doesn't change

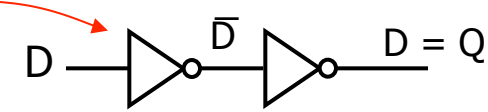
What is Q if G=1 ?



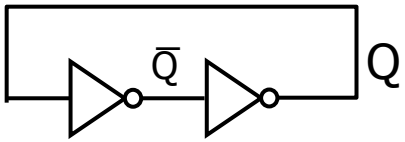
Q = D !

What if D changes while G =1?

- Q keeps changing to D



No feedback



Feedback

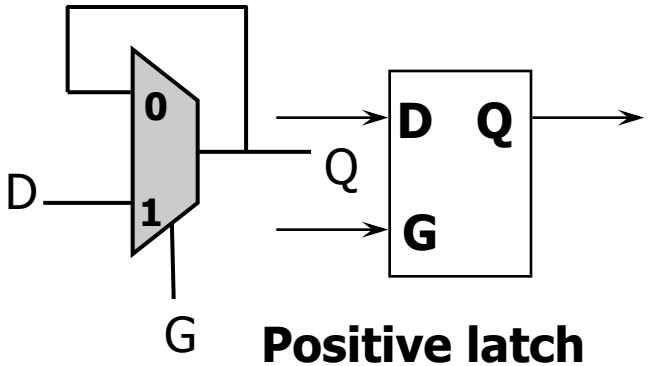
Advantages:

1) Maintains remembered state for as long as power is applied.
2) State is DIGITAL
3) logic gates are built to restore marginal signal levels, so noise shouldn't be a problem

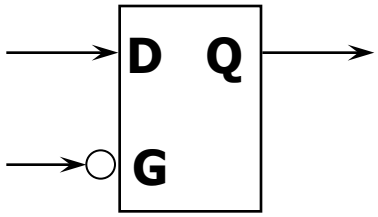
Disadvantage: Requires more transistors than DRAM, more space, more power, more expensive

This storage circuit is the basis for memories in CPU and nearby memories (caches)-Static Random Access Memories (SRAMs)

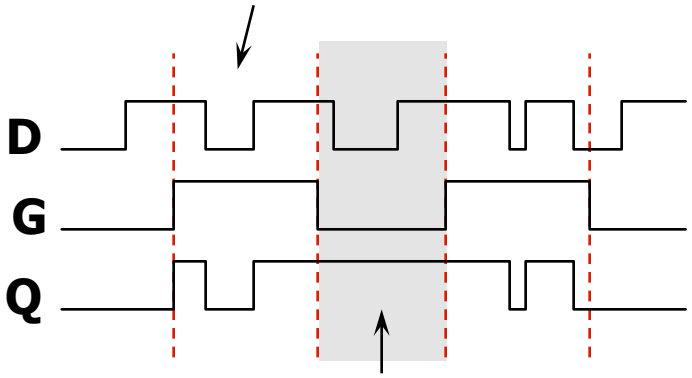
Static D Latch: Positive & Negative Versions



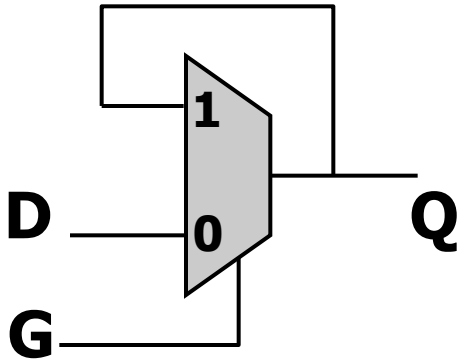
Positive latch



Negative latch



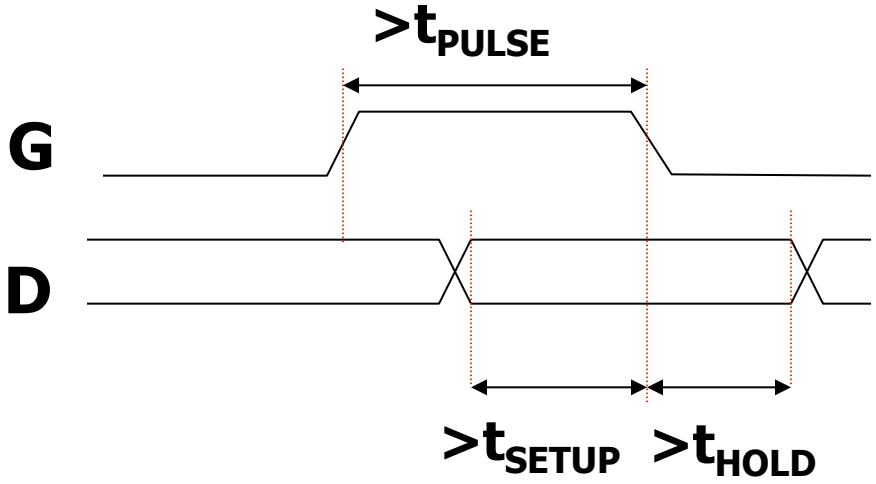
Q stable



[Show mux & gates for Negative latch, side by side with circuit for a Positive latch]

Latch Timing

- Circuits with memory must follow some rules
  - Guarantee that inputs to sequential devices are valid and stable during periods when they may influence state changes
    - This is assured with additional timing specifications



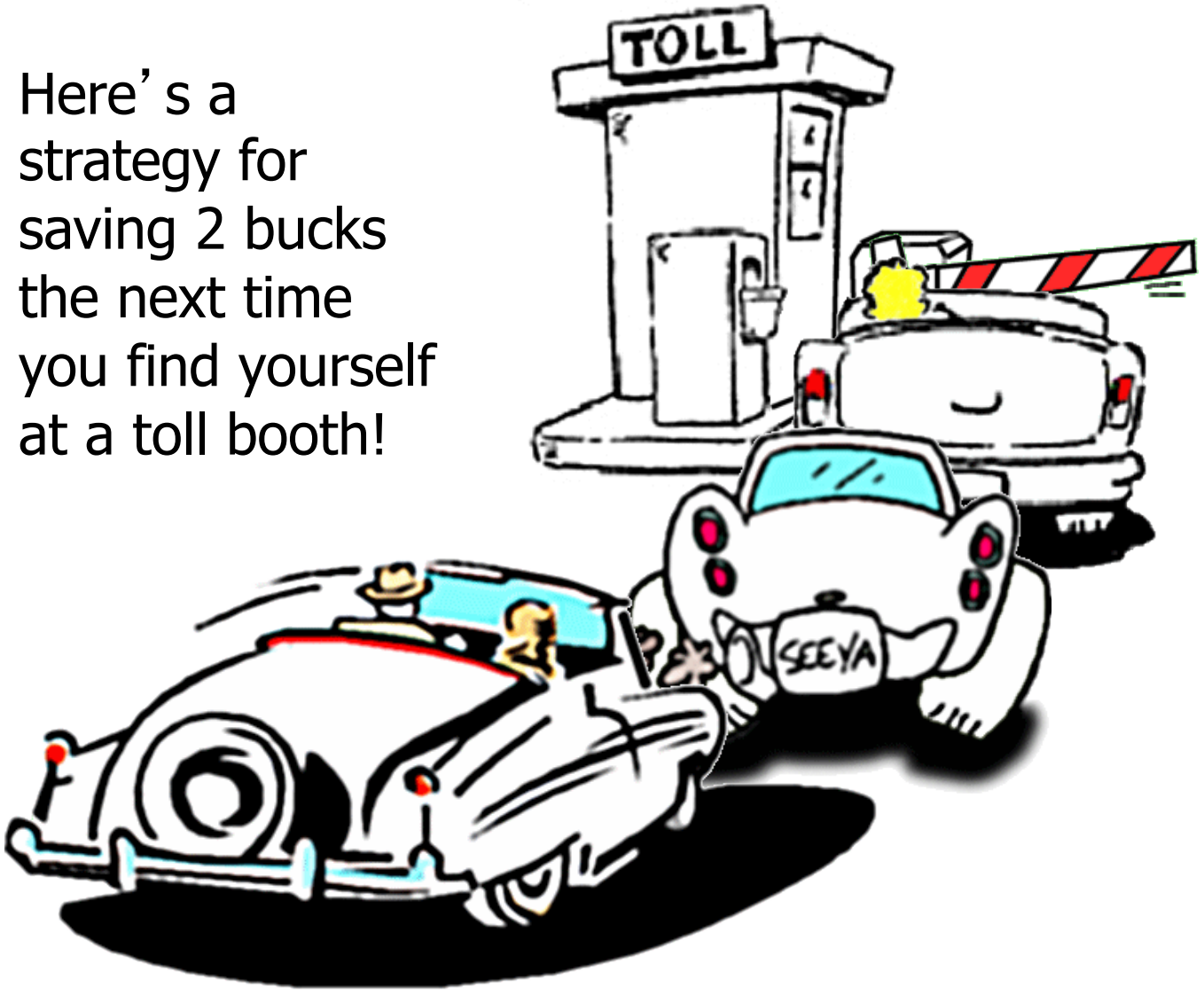
t_{PULSE} (minimum pulse width): *guarantee G is active for long enough for latch to capture data*

t_{SETUP} (setup time): *guarantee that D value has propagated through feedback path before latch becomes opaque*

t_{HOLD} (hold time): *guarantee latch is opaque and Q is stable before allowing D to change again*

Flakey Control Systems

Here's a strategy for saving 2 bucks the next time you find yourself at a toll booth!



Flakey Control Systems

Here's a strategy for saving 2 bucks the next time you find yourself at a toll booth!

Flakey Control Systems

Here's a strategy for saving 2 bucks the next time you find yourself at a toll booth!

WARNING:

Professional Drivers Used!

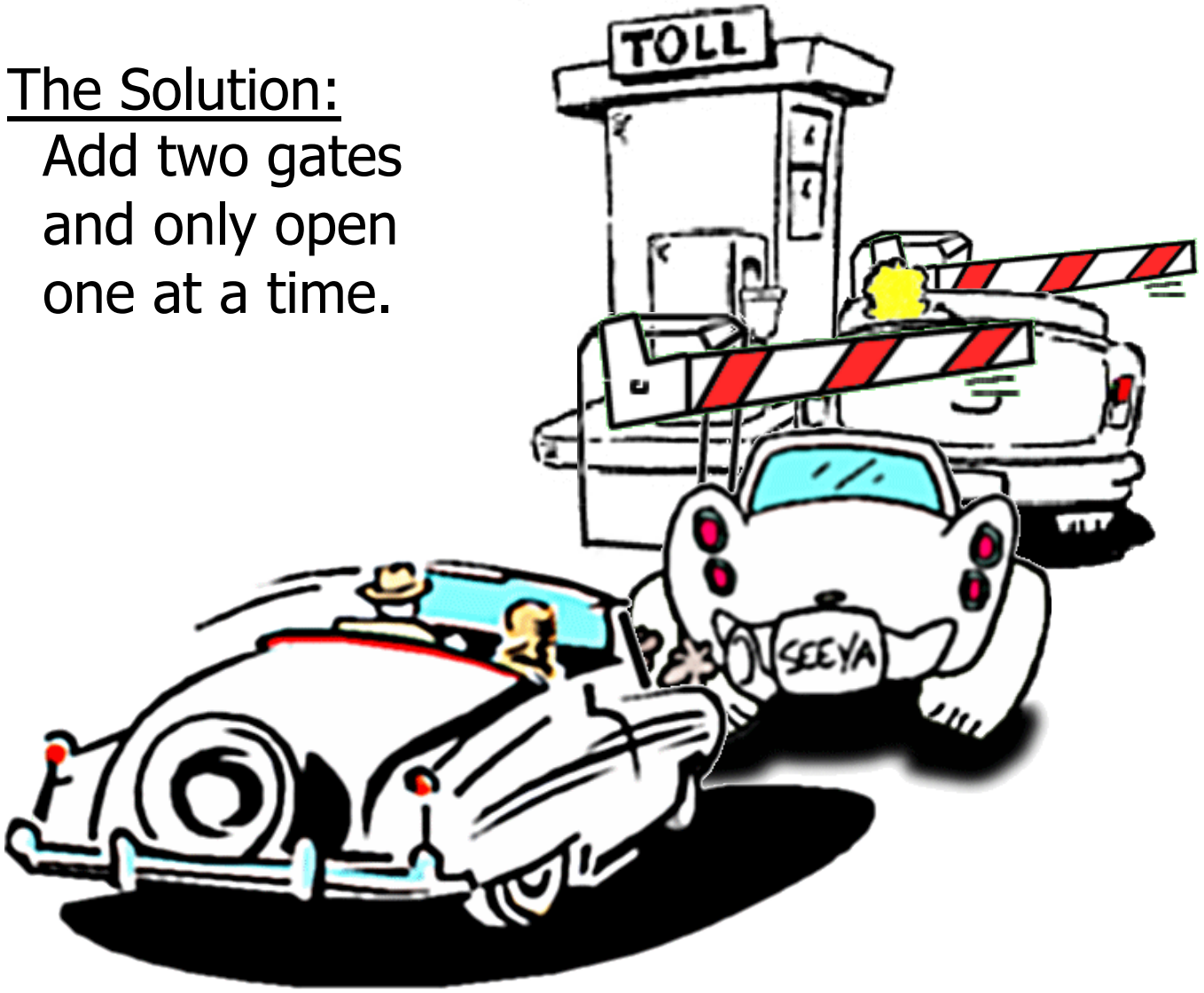
DON'T try this

At home!

Escapement Strategy

The Solution:

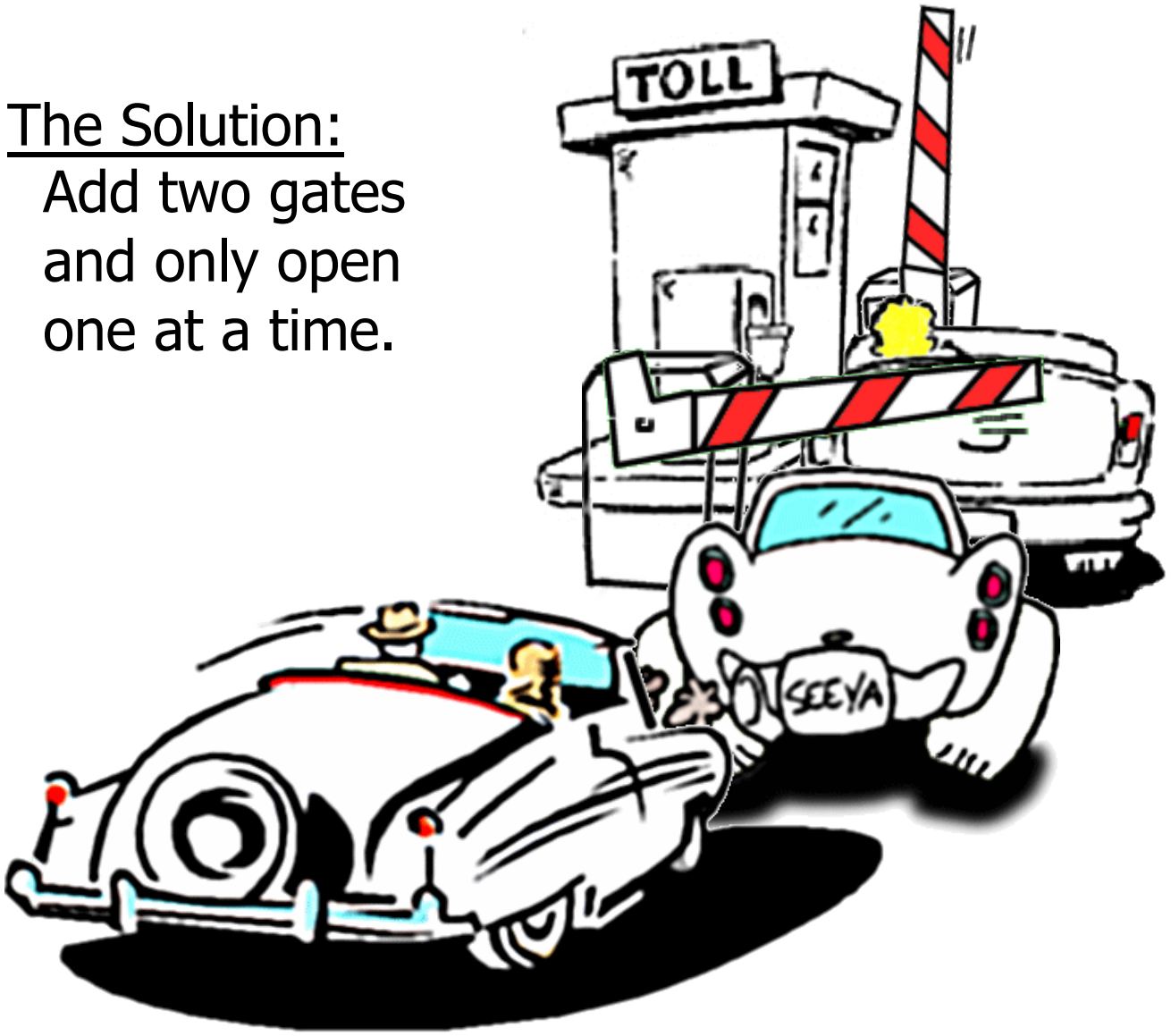
Add two gates and only open one at a time.



Escapement Strategy

The Solution:

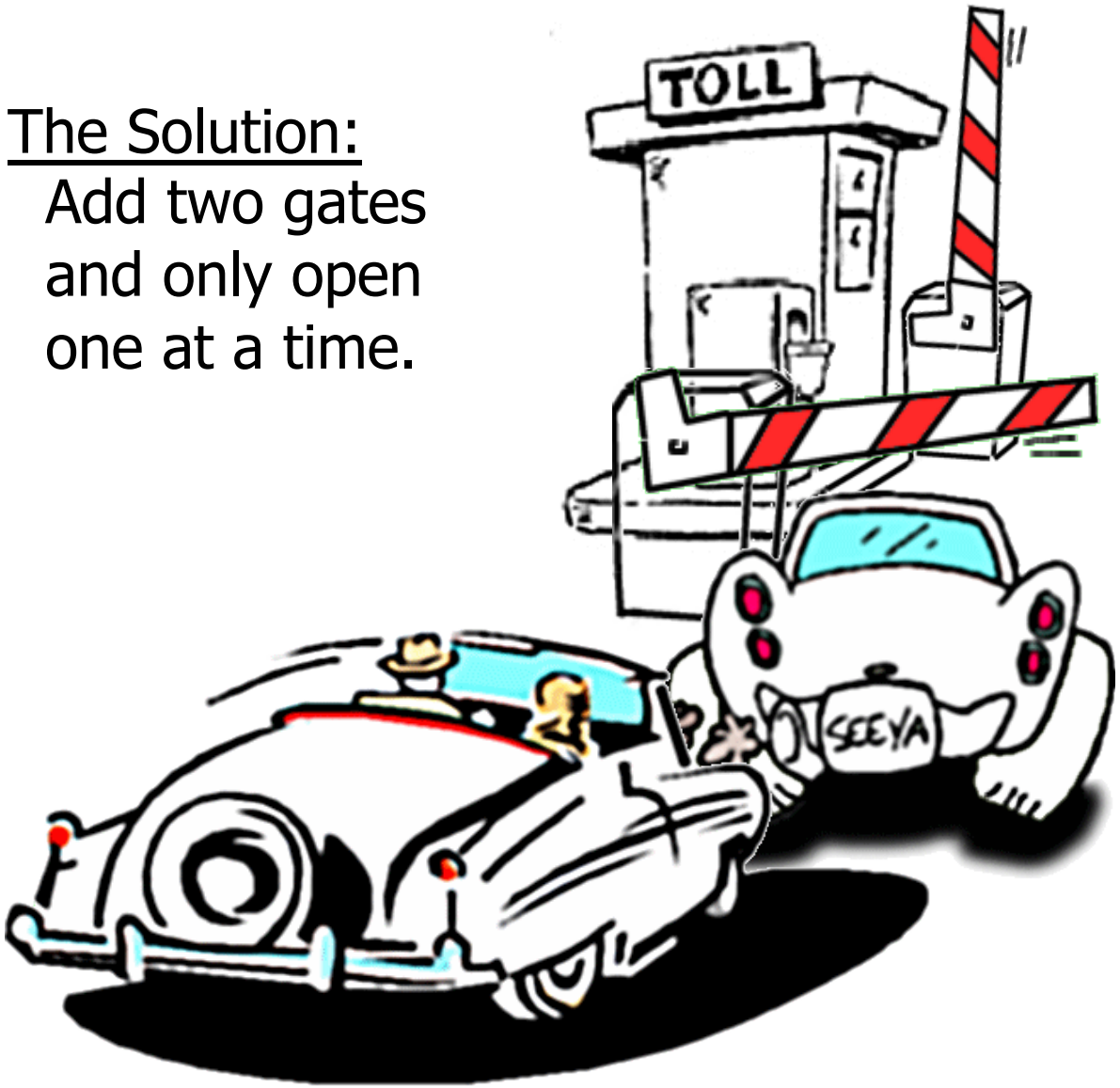
Add two gates and only open one at a time.



Escapement Strategy

The Solution:

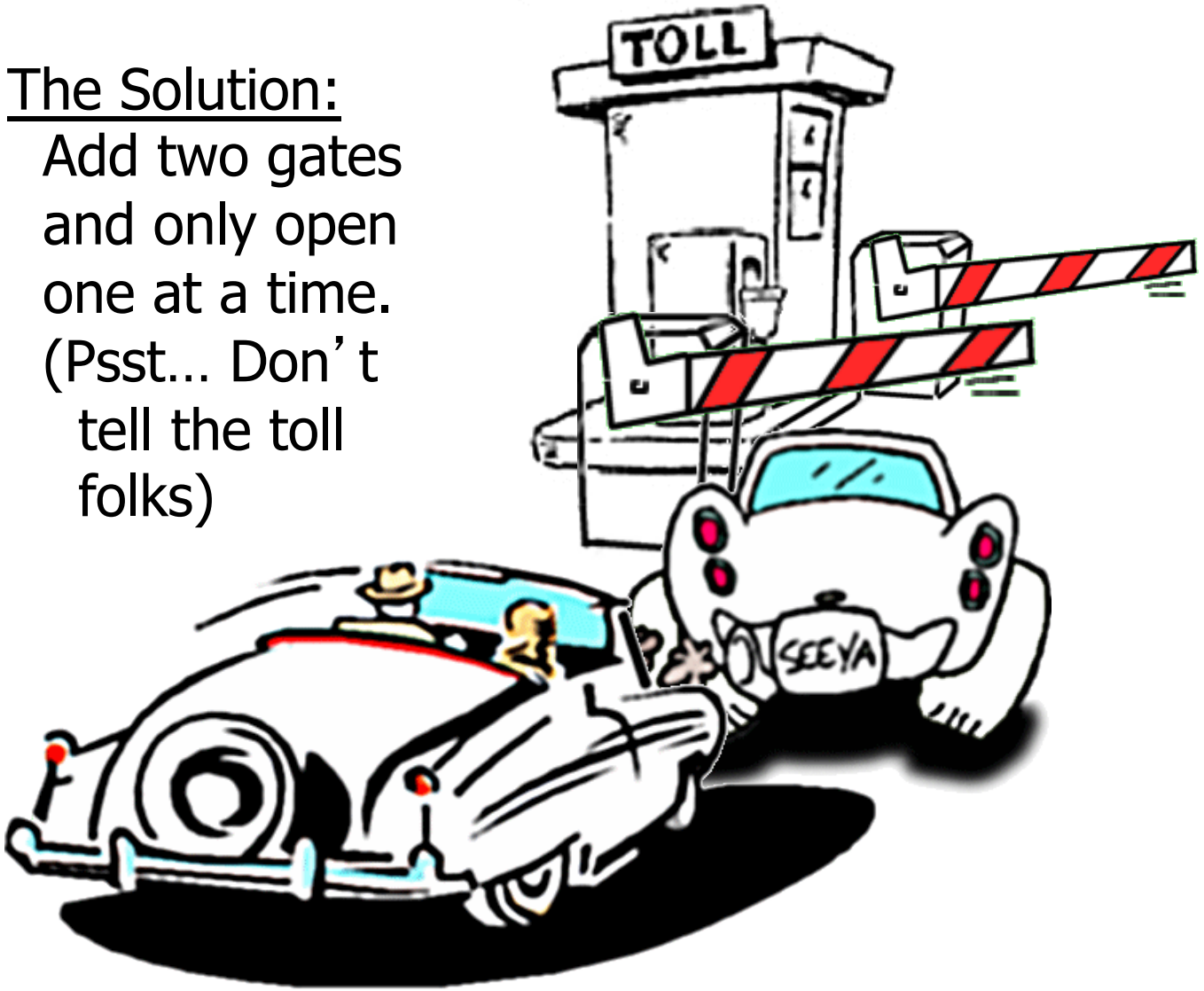
Add two gates and only open one at a time.



Escapement Strategy

The Solution:

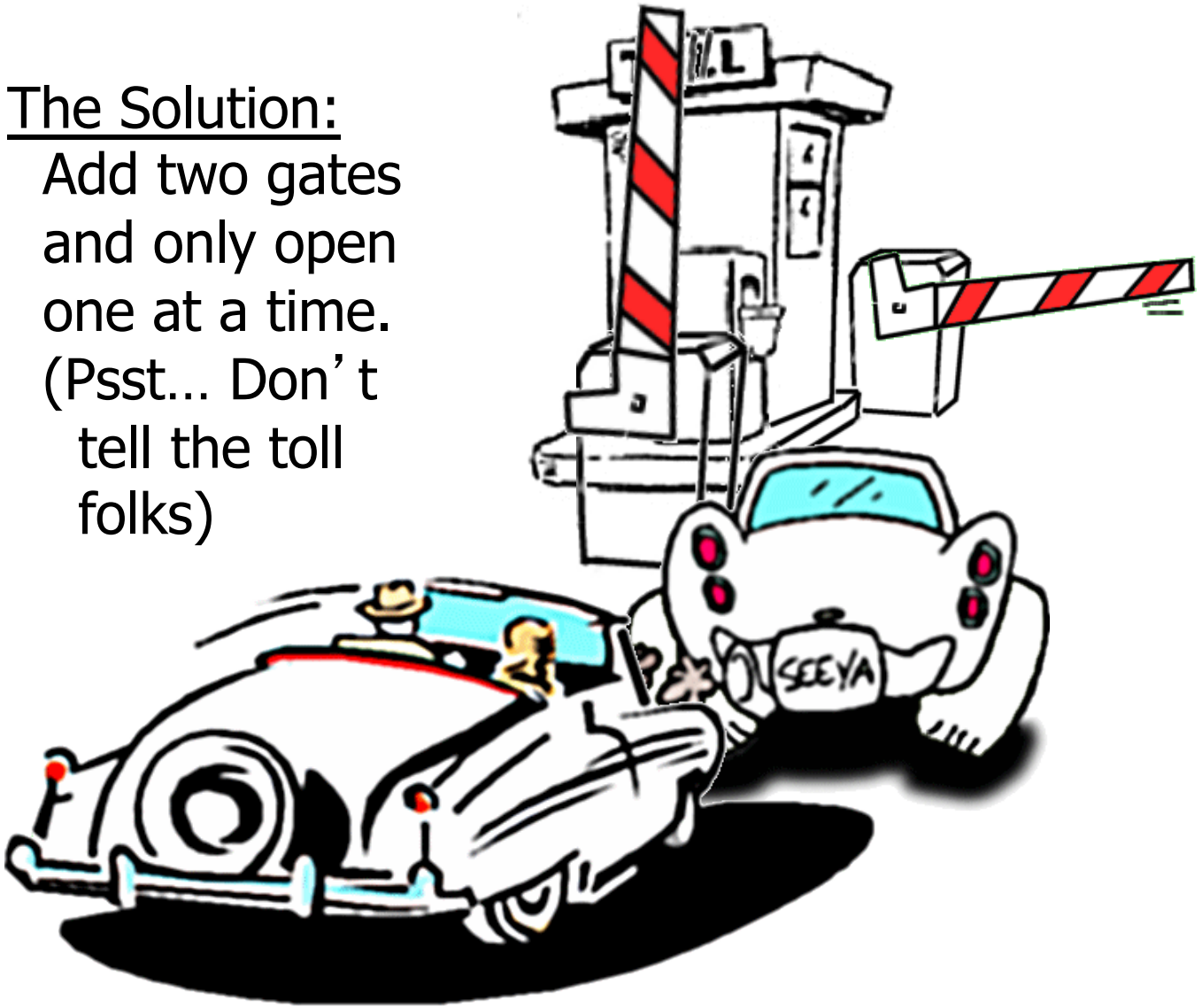
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

The Solution:

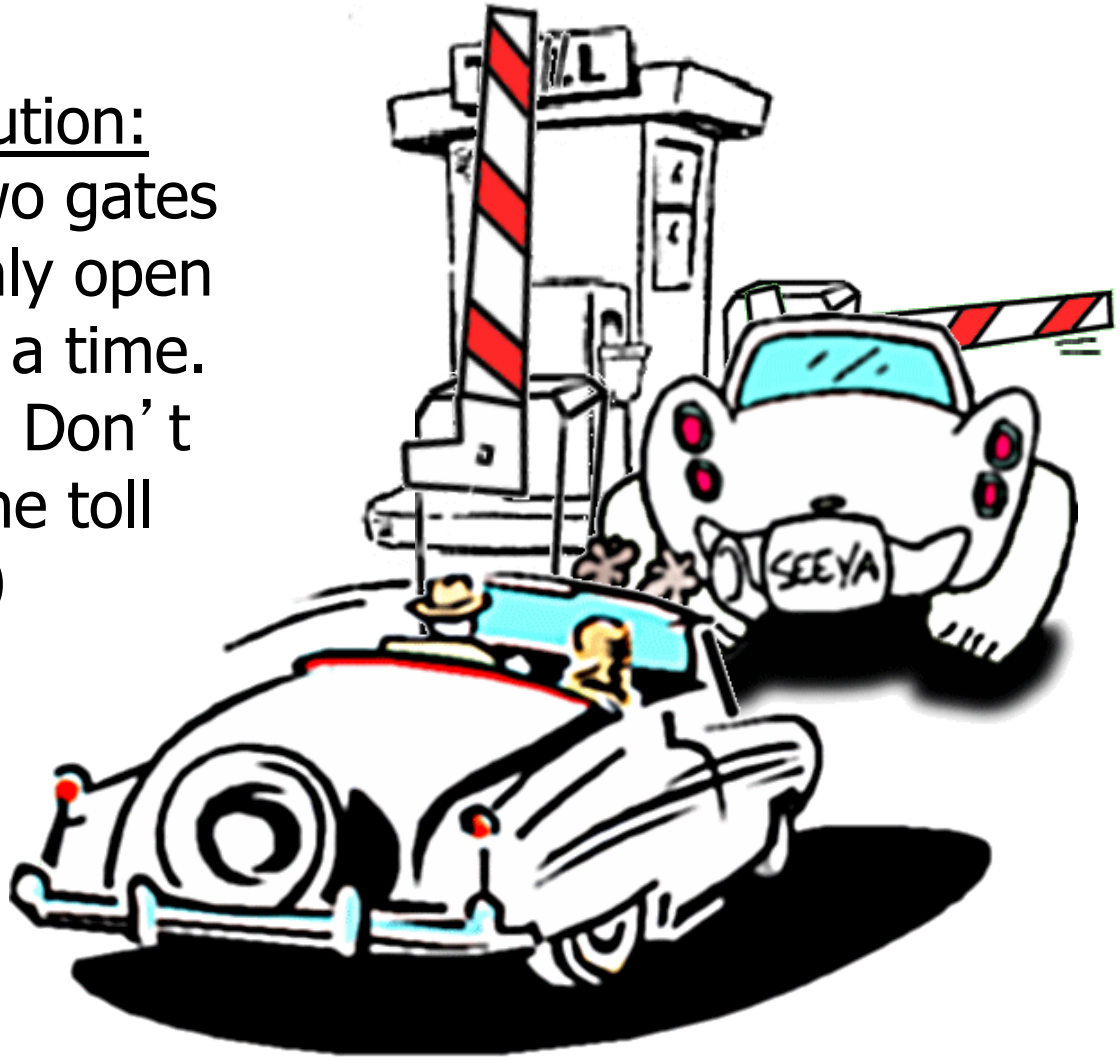
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

<u>The Solution:</u>

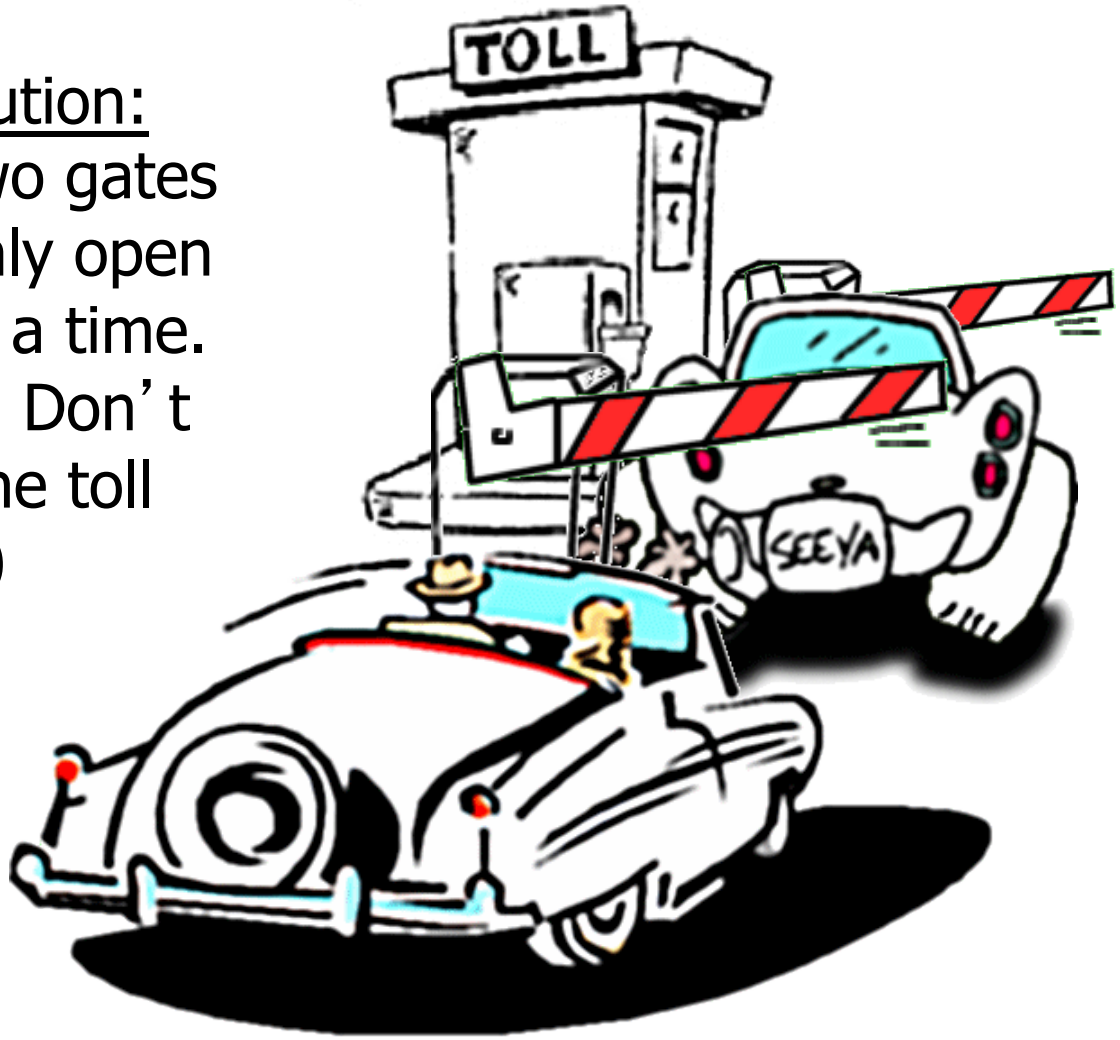
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

The Solution:

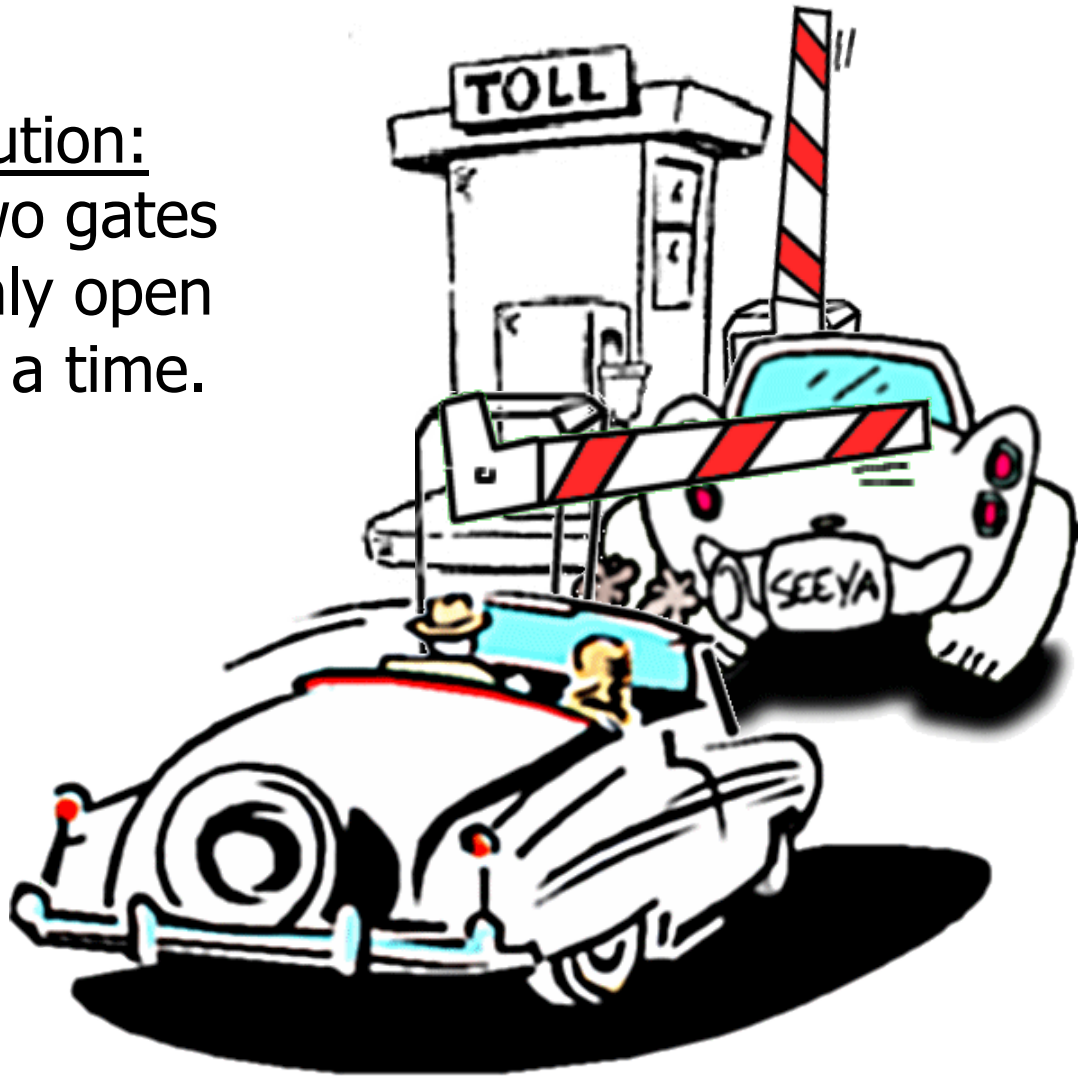
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

The Solution:

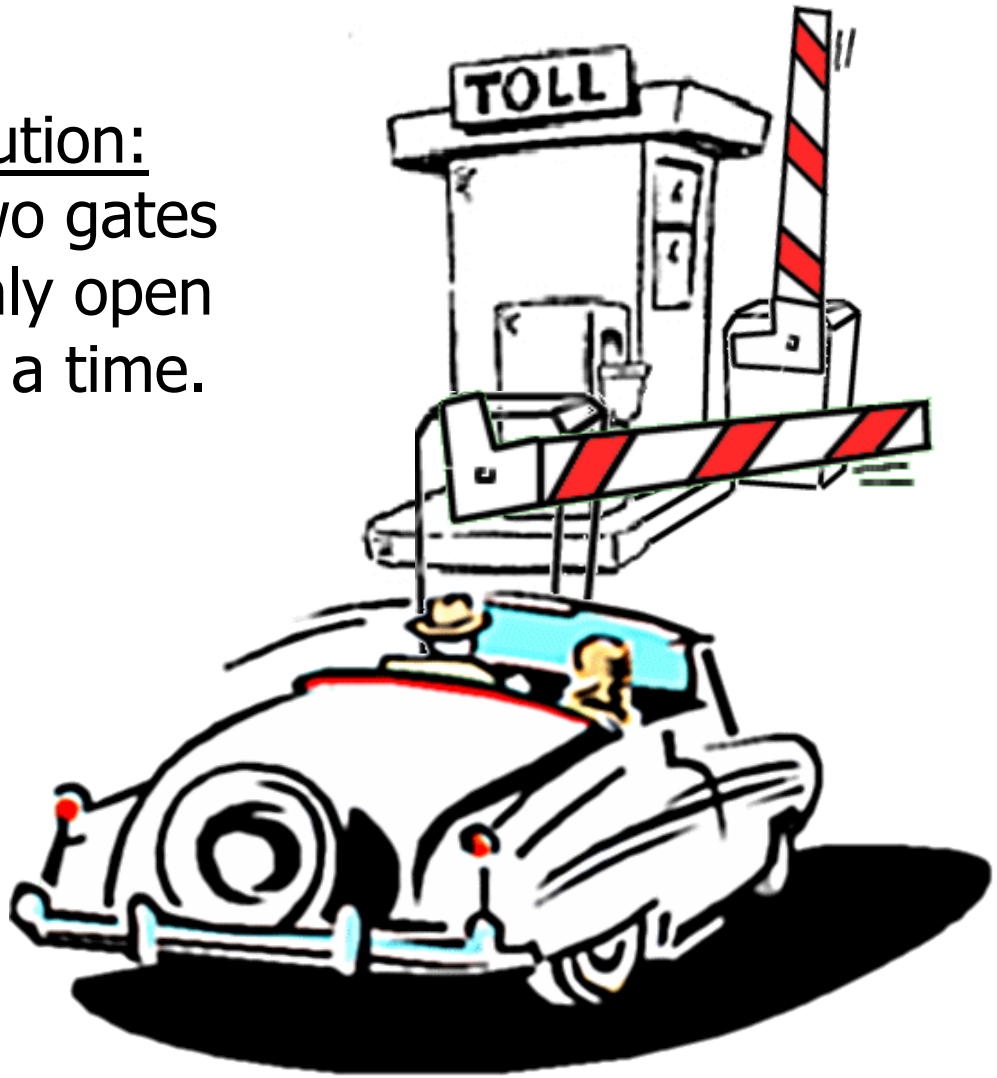
Add two gates and only open one at a time.



Escapement Strategy

The Solution:

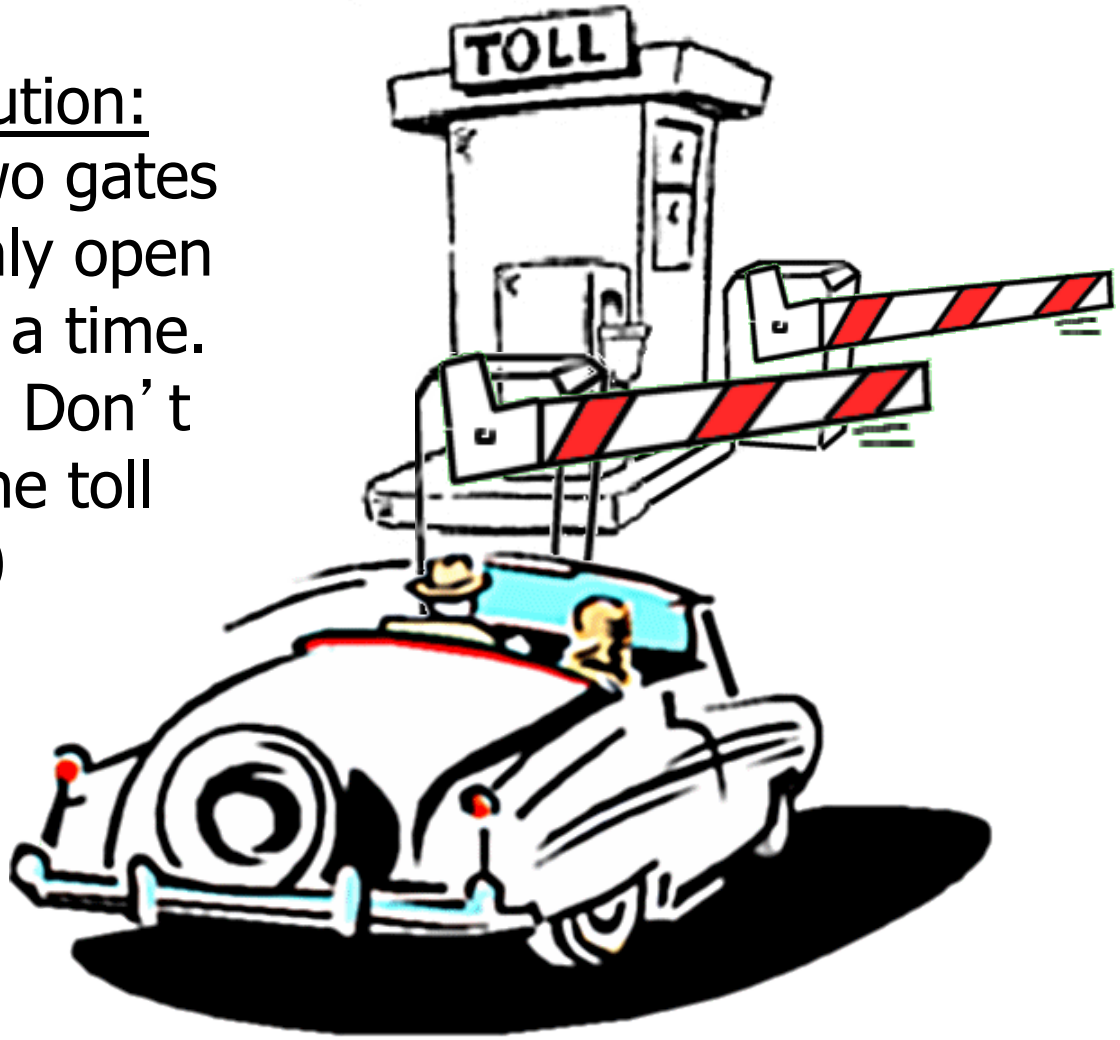
Add two gates and only open one at a time.



Escapement Strategy

<u>The Solution:</u>

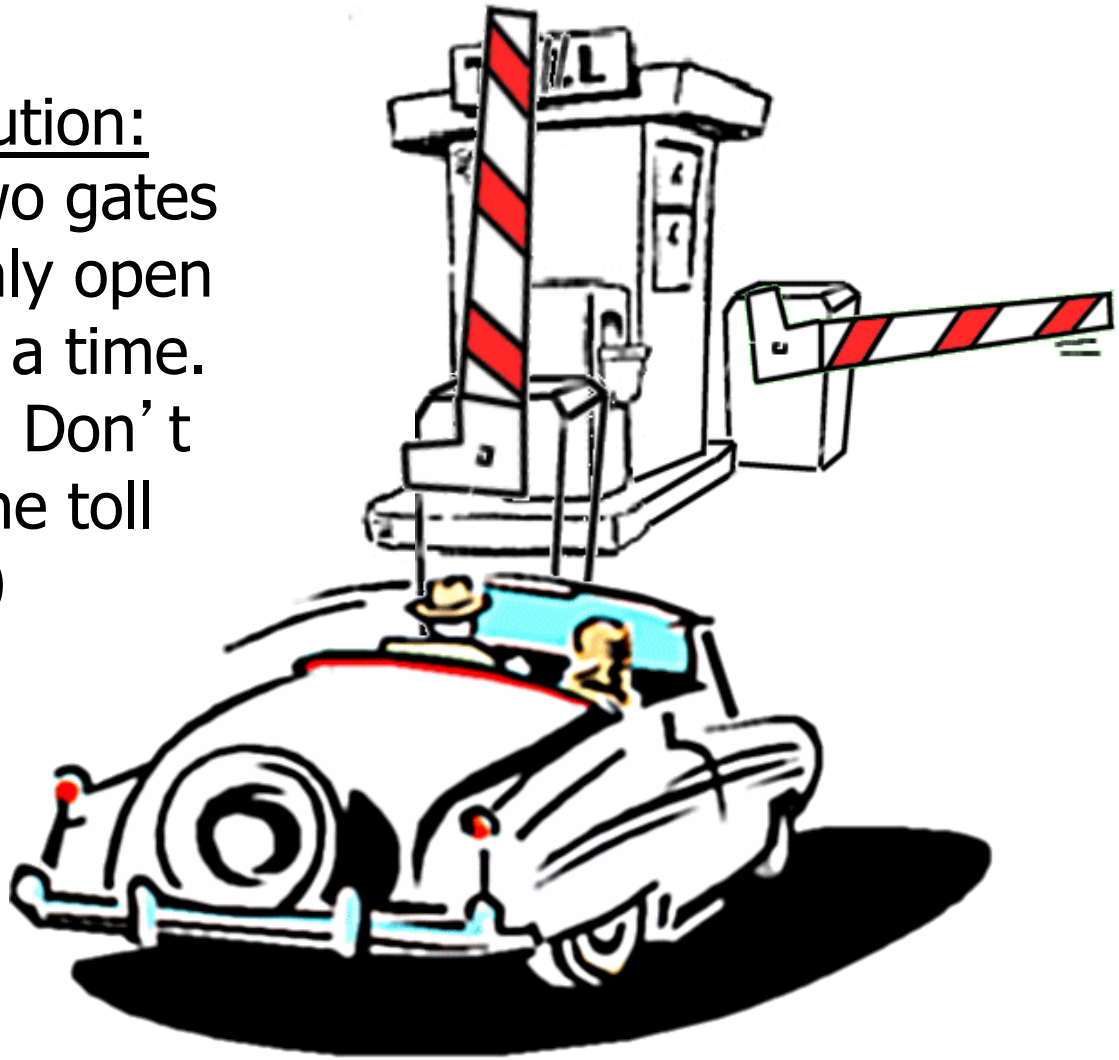
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

<u>The Solution:</u>

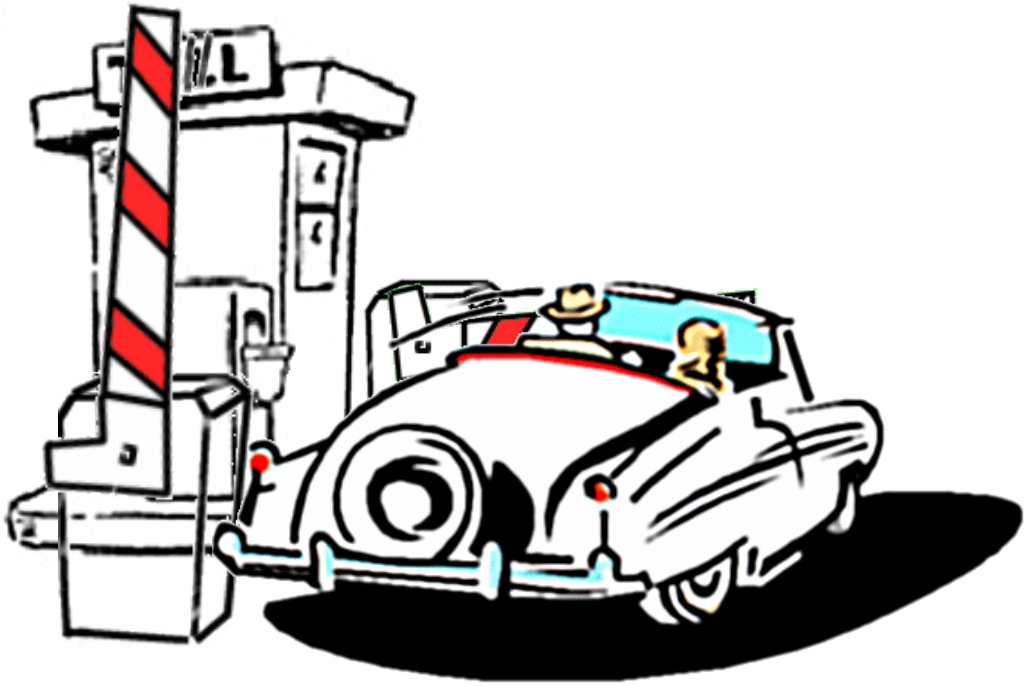
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

<u>The Solution:</u>

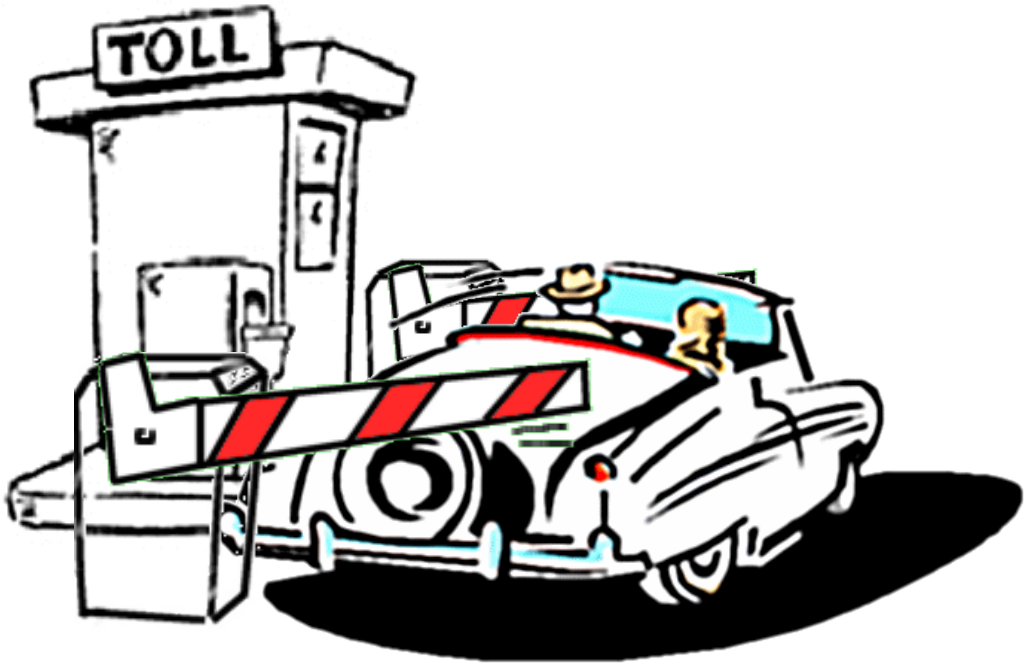
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

The Solution:

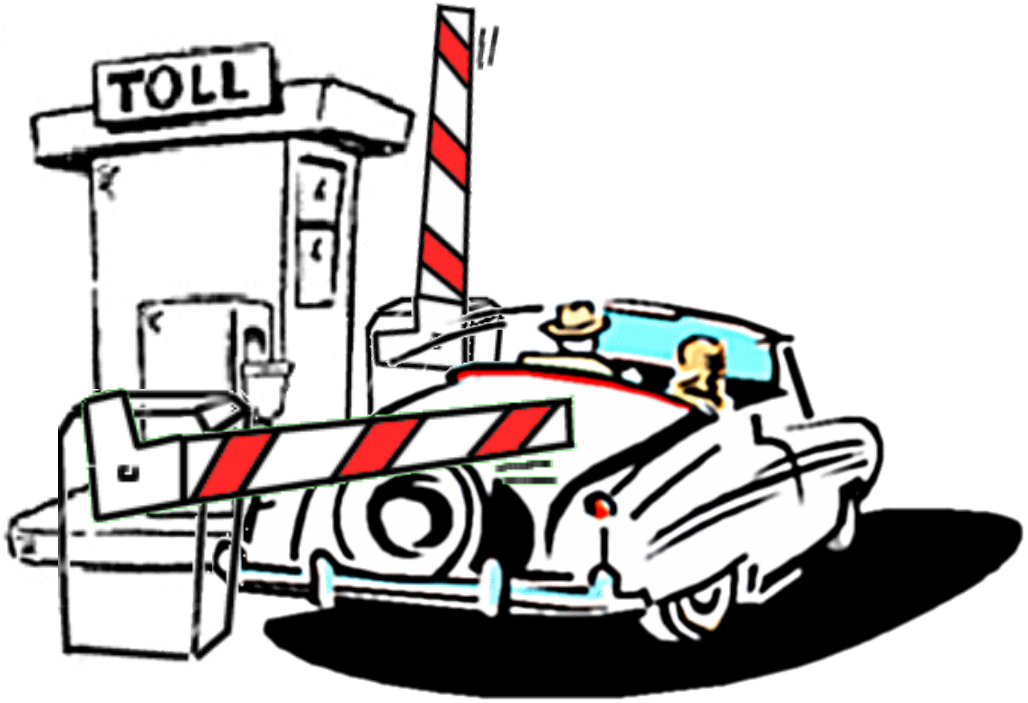
Add two gates and only open one at a time. (Psst… Don't tell the toll folks)



Escapement Strategy

The Solution:

Add two gates and only open one at a time.



Escapement Strategy

The Solution:

Add two gates and only open one at a time.

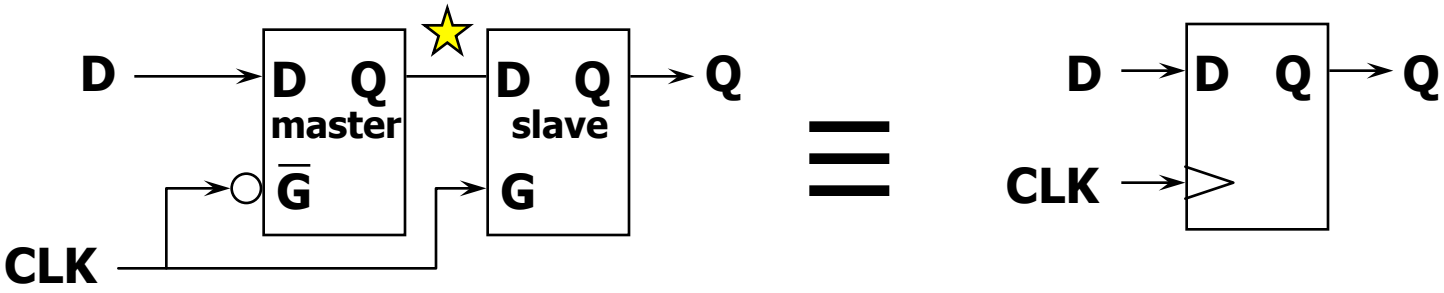
Escapement Strategy

The Solution:

Add two gates and only open one at a time. (Psst… Don't tell the toll folks)

Key Idea: At no time is there an open path through both gates…

Edge-triggered Flip Flop

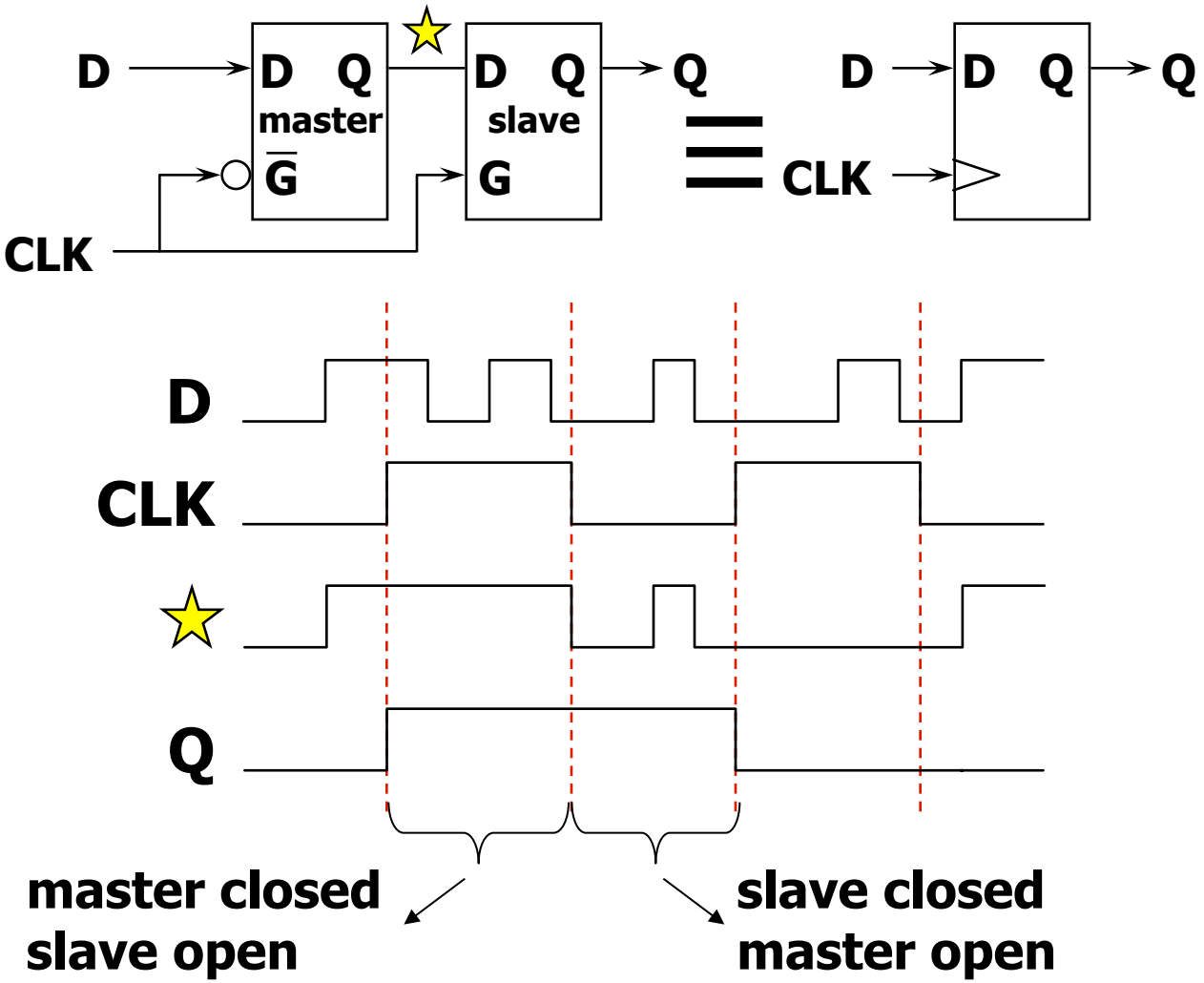


- Logical "escapement":
  - Double-gated toll booth built using logic gates
- Observations:
  - only one latch "transparent" at any time:
    - master closed when slave is open (CLK is high)
    - slave closed when master is open (CLK is low)
    - no combinational path all the way through flip flop
  - Q only changes shortly after 0 →1 transition of CLK, so flip flop appears to be "triggered" by rising edge of CLK

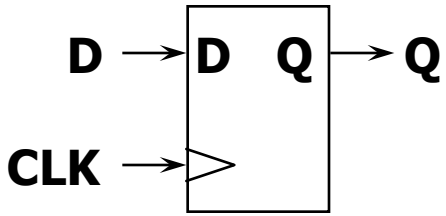
Transitions mark *instants*, not intervals

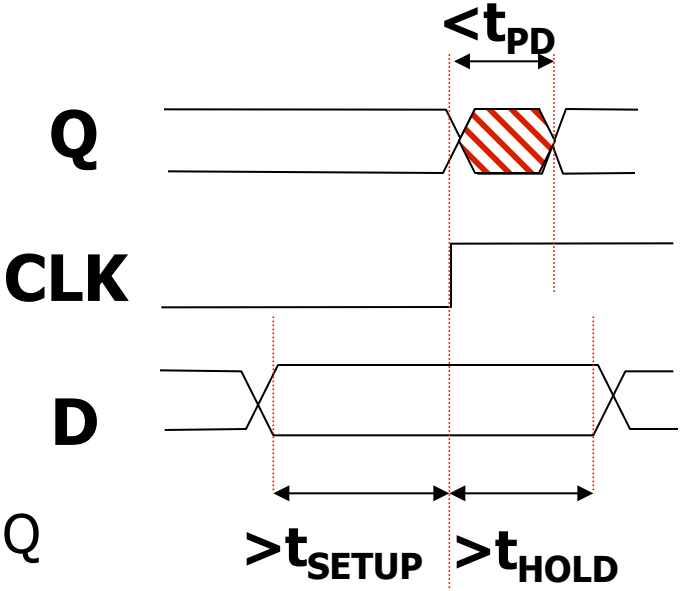
Each latch "latches" when G goes down

Flip Flop Waveforms



Flip Flop Timing





t_{PD}: maximum propagation delay, CLK →Q

t_{SETUP}: setup time

guarantee that D has propagated through feedback path before master closes

t_{HOLD}: hold time

guarantee master is closed and data is stable before allowing D to change

Summary

- Regular Arrays can be used to implement arbitrary logic functions
- Memories
  - ROMs are HARDWIRED memories
  - RAMs include storage elements that are read-write
    - dynamic memory: compact, only reliable short-term
    - static memory: controlled use of positive feedback
- For static storage:
  - Level-sensitive D-latches; edge-triggered flipflops
  - Timing issues: setup and hold times

More on RAM at a basic level:
http://archive.arstechnica.com/paedia/r/ram_guide/ram_guide.part1-1.html