

Digital Logic II



So Where Are We Going With This?

- Sequential Logic Circuits
 - State
 - Memory
 - Address space
 - Addressability
 - $2^2 \times 3$ bit memory
 - Clocks
 - Edge-triggered and level-triggered flip-flops
 - Example State Machine
 - Design
 - Simplification
 - Implementation
 - One-hot and Encoded State Machines

Warning

➤ Another big concept approaching!

Combinational vs. Sequential Logic

➤ Combinational

- Combination of AND, OR, NOT (plus NAND & NOR)
- Same inputs always produce same output
- Analogous to cheap bicycle lock

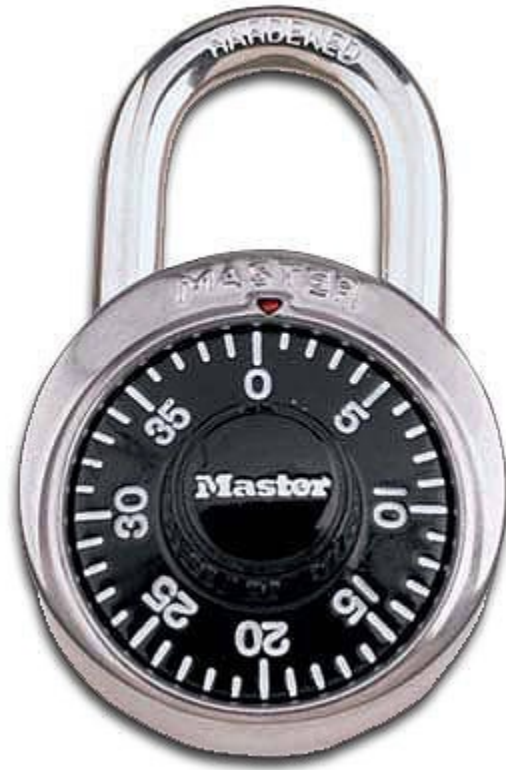
➤ Sequential

- Requires **storage elements**
- Output depends on inputs plus ***state***
- Analogous to a RLR combination lock
- Used to build memory & ***state machines***

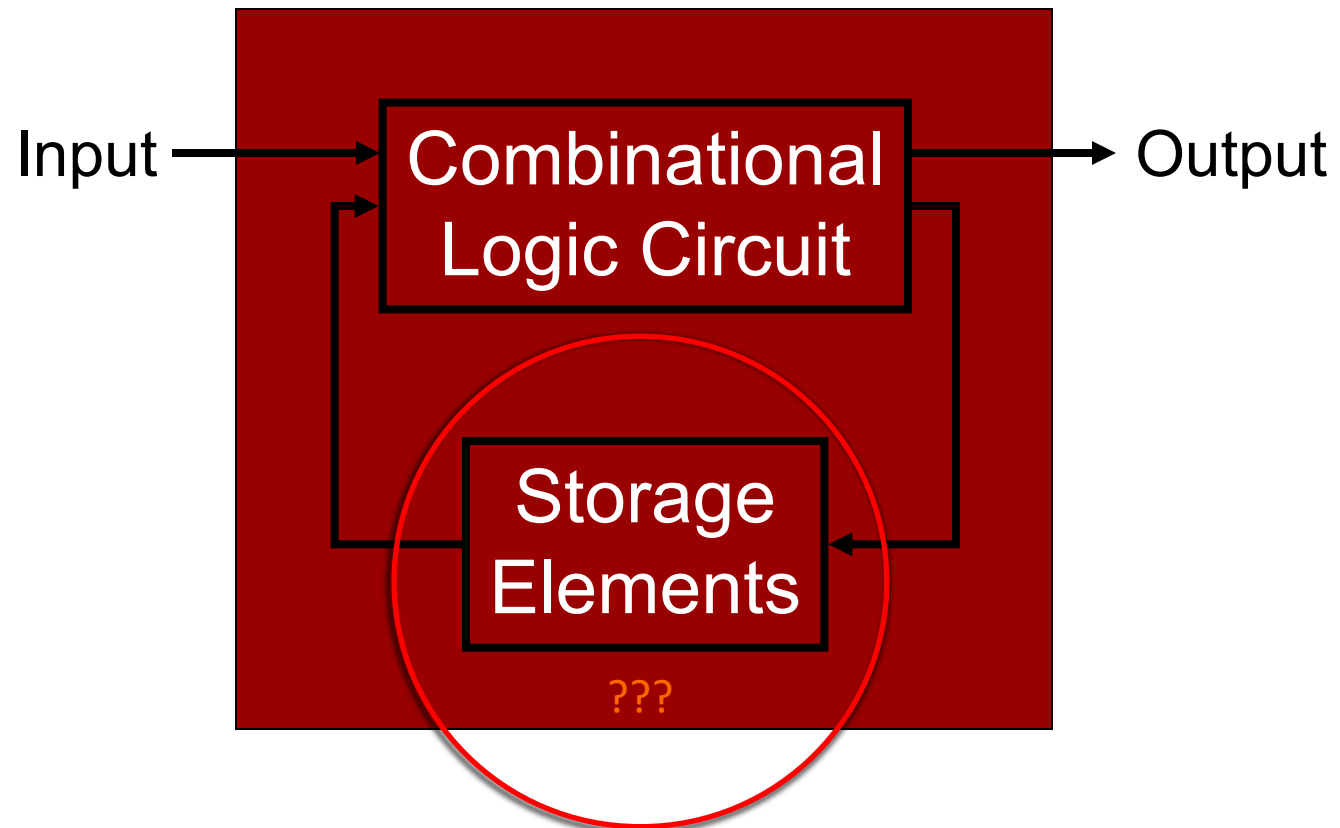
So What is *State*?



State or No State?

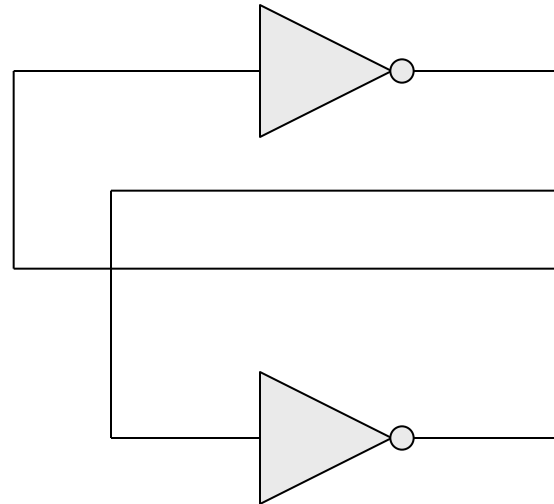


Sequential Logic Circuit

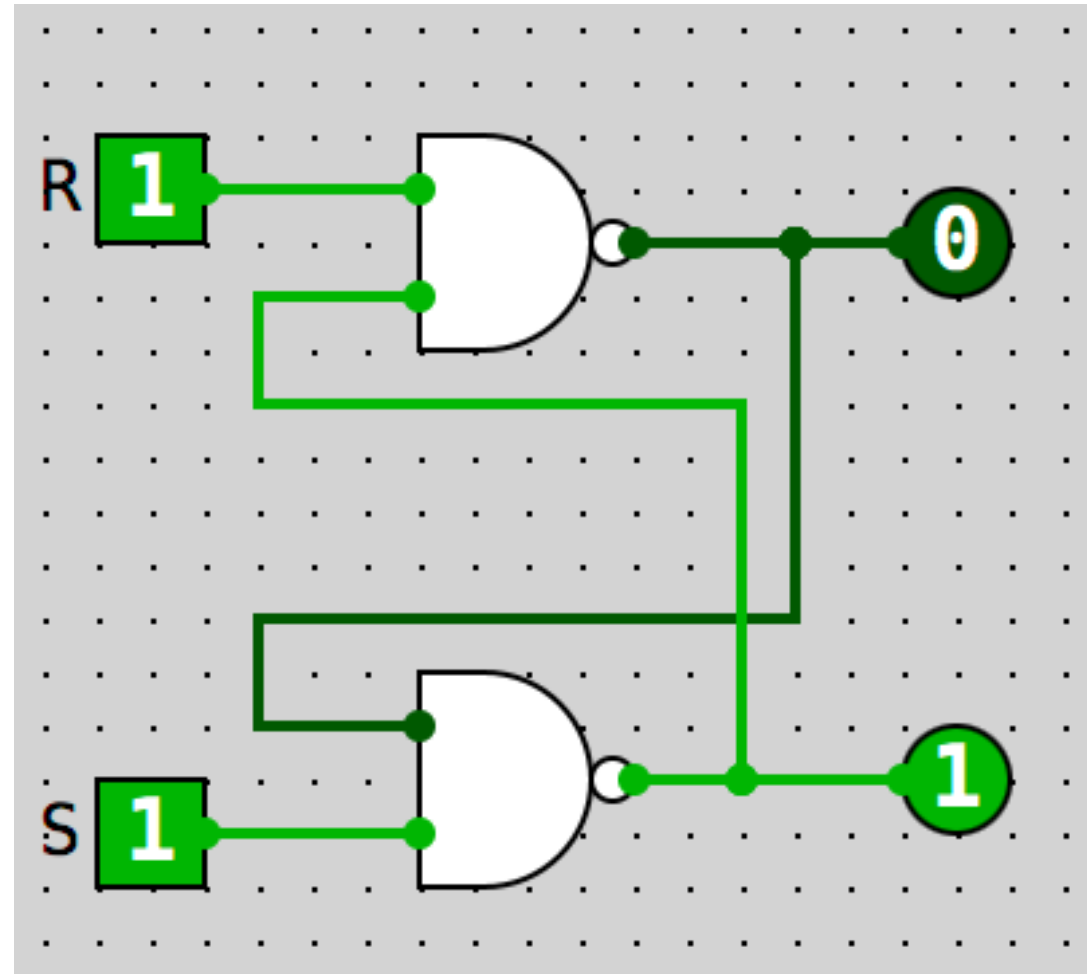


Basic Storage

Consider:



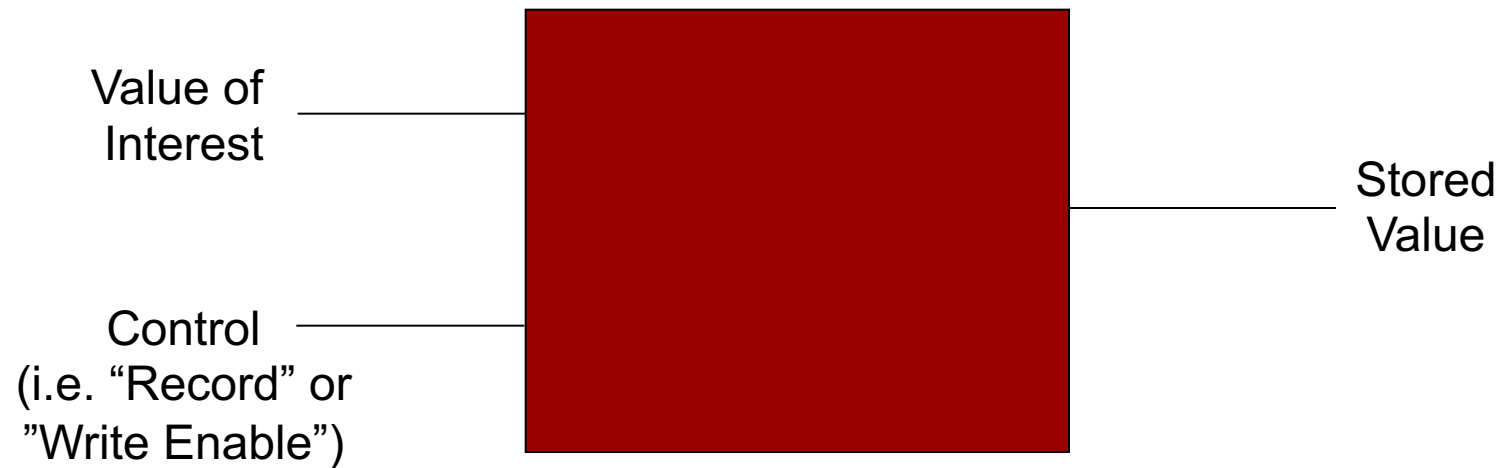
R/S Latch



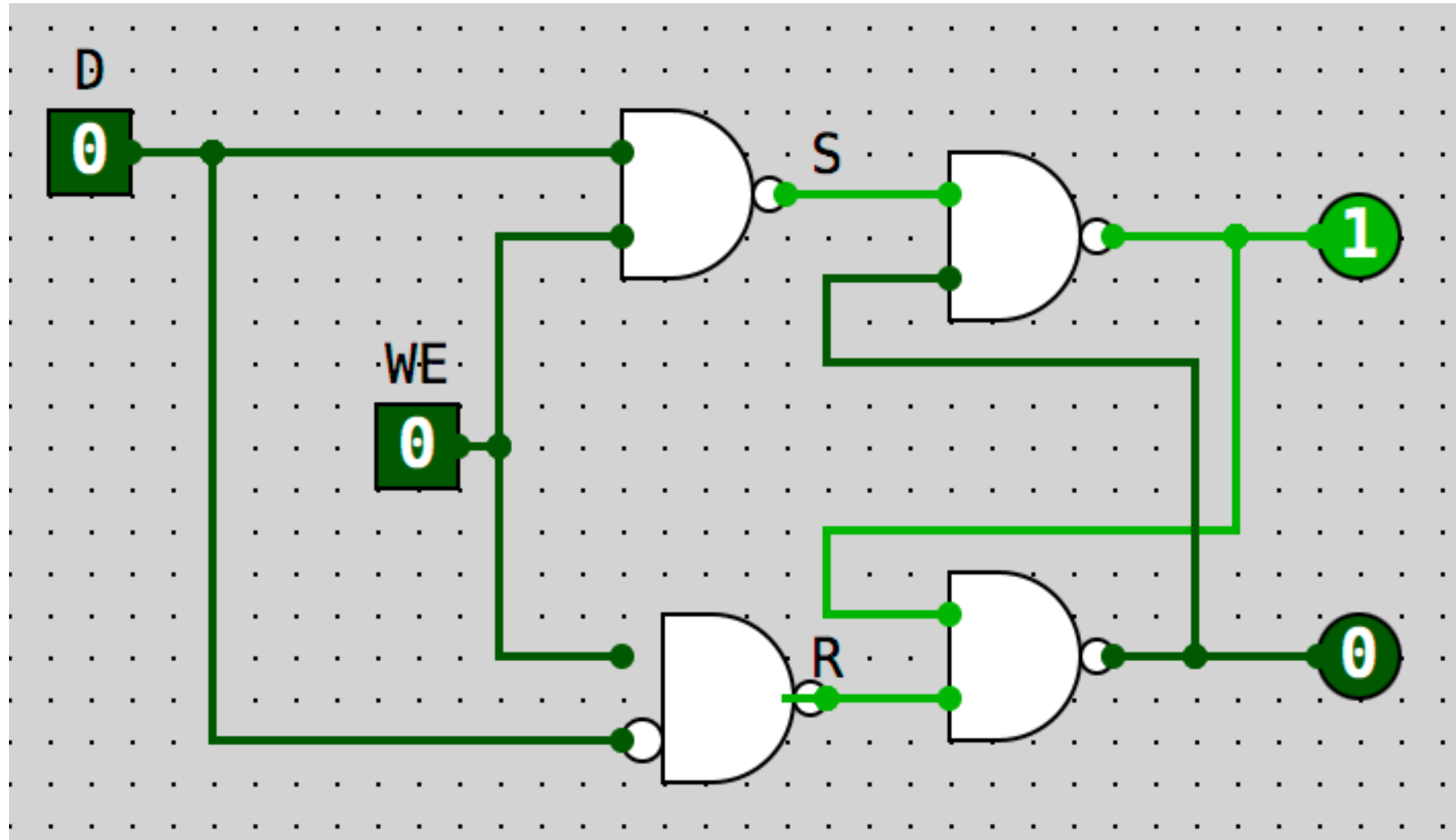
What happens if both inputs to the R/S latch are 0?

Gated D Latch

- The RS Latch can be set or cleared but what if we want to record the value presented on the input?

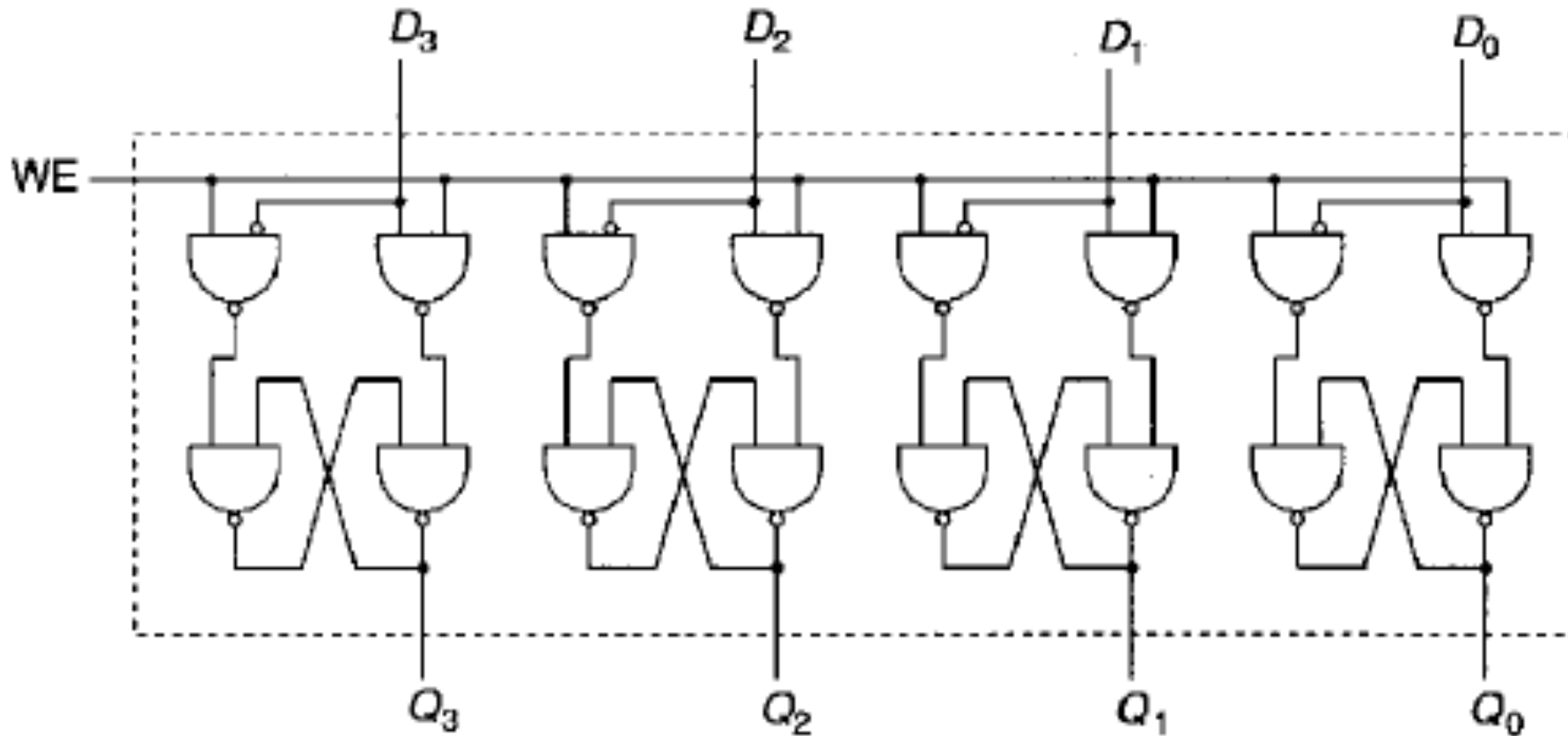


Gated D Latch



What is a Register?

➤ Basically an array of Gated D latches!



Memories (or How is This Different From Registers?)

- Do you think it would be useful to have more than one storage location accessible with only one set of wires?

Definitions

- Address Space -- How many addresses are possible?
- Addressability -- How big is each memory location?

If you have a memory with a 16-bit address in which each byte is given a distinct memory address, what is the addressability of this memory?

- A. 8 bits
- B. 16 bits
- C. 24 bits
- D. 32 bits



Today's number is 82,200

If you have a memory with a 16-bit address in which each byte is given a distinct memory address, what is this memory's address space?

- A. 256
- B. 65536
- C. 524288
- D. 4294967296



➤ Basic Building Blocks of a Memory

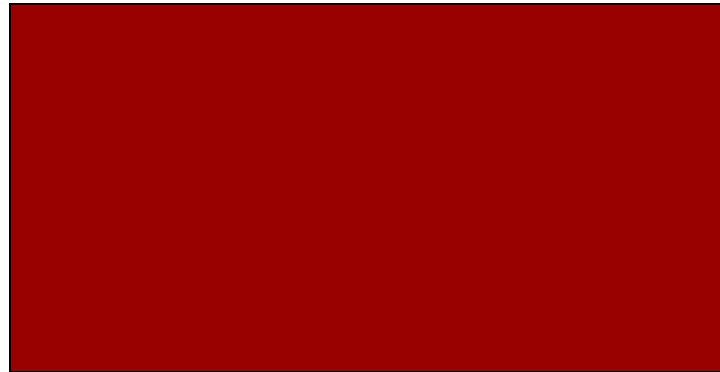
➤ Register

➤ Gated D Latch

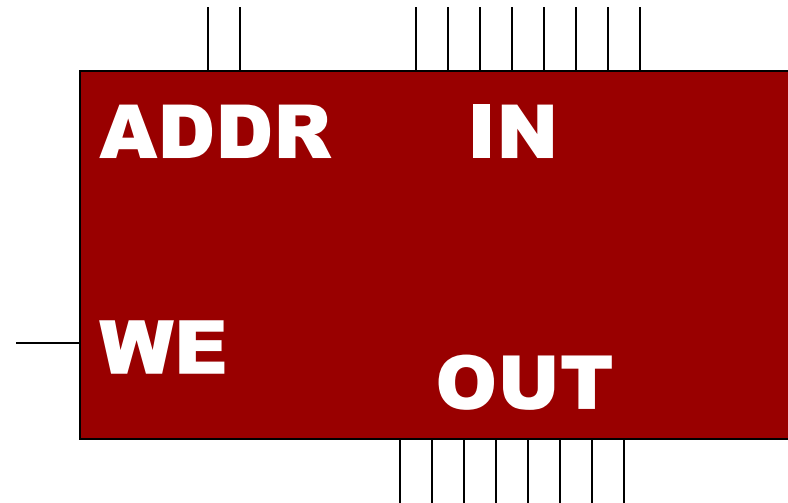
➤ Decoder

➤ Multiplexors

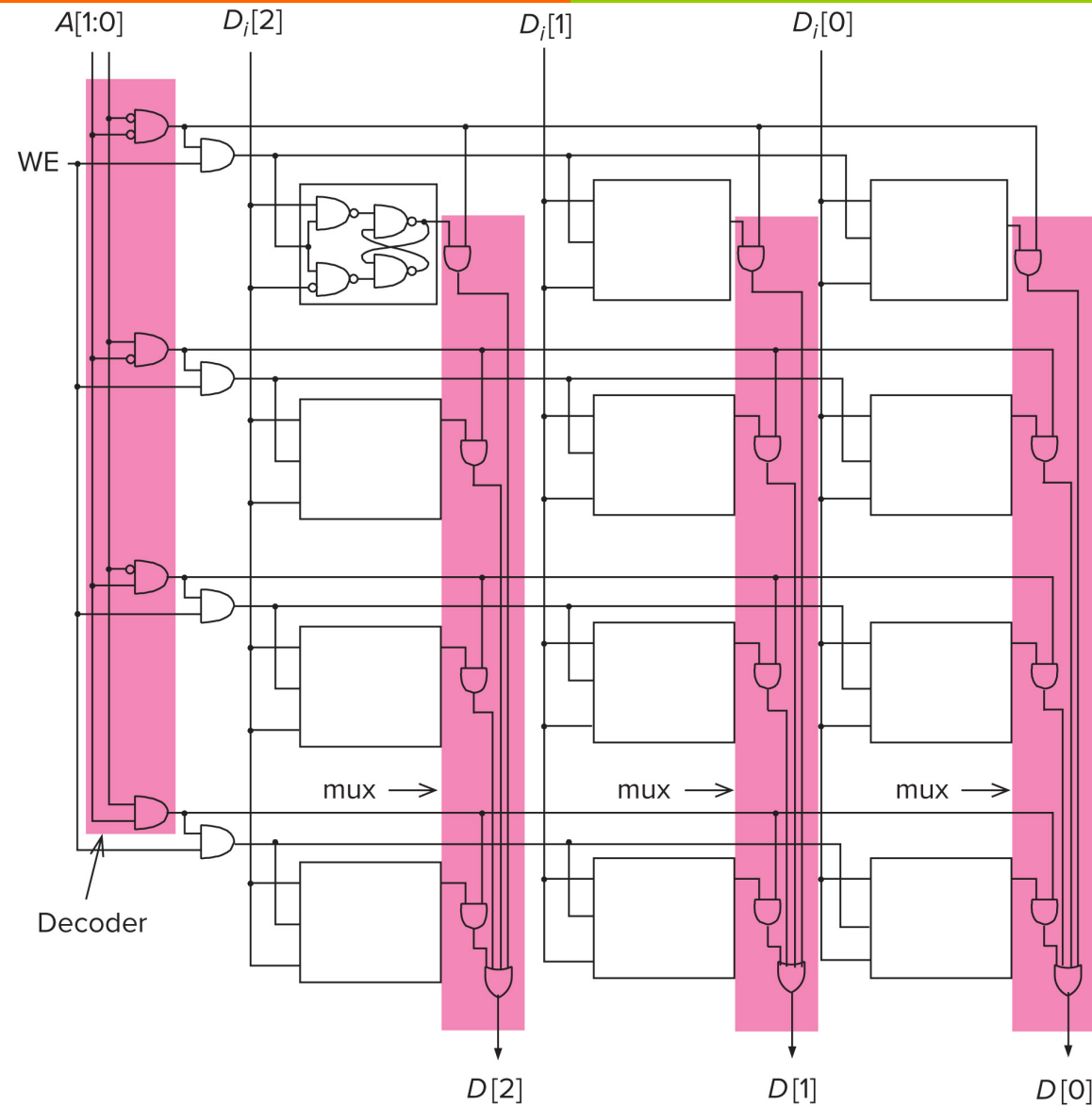
➤ Looking from the outside, what do we need?



➤ Looking from the outside, what do we need?



A $2^2 \times 3$ -bit Word Memory



Sequential Logic Circuits, Now That We Have Storage



Remember: Compare and Contrast

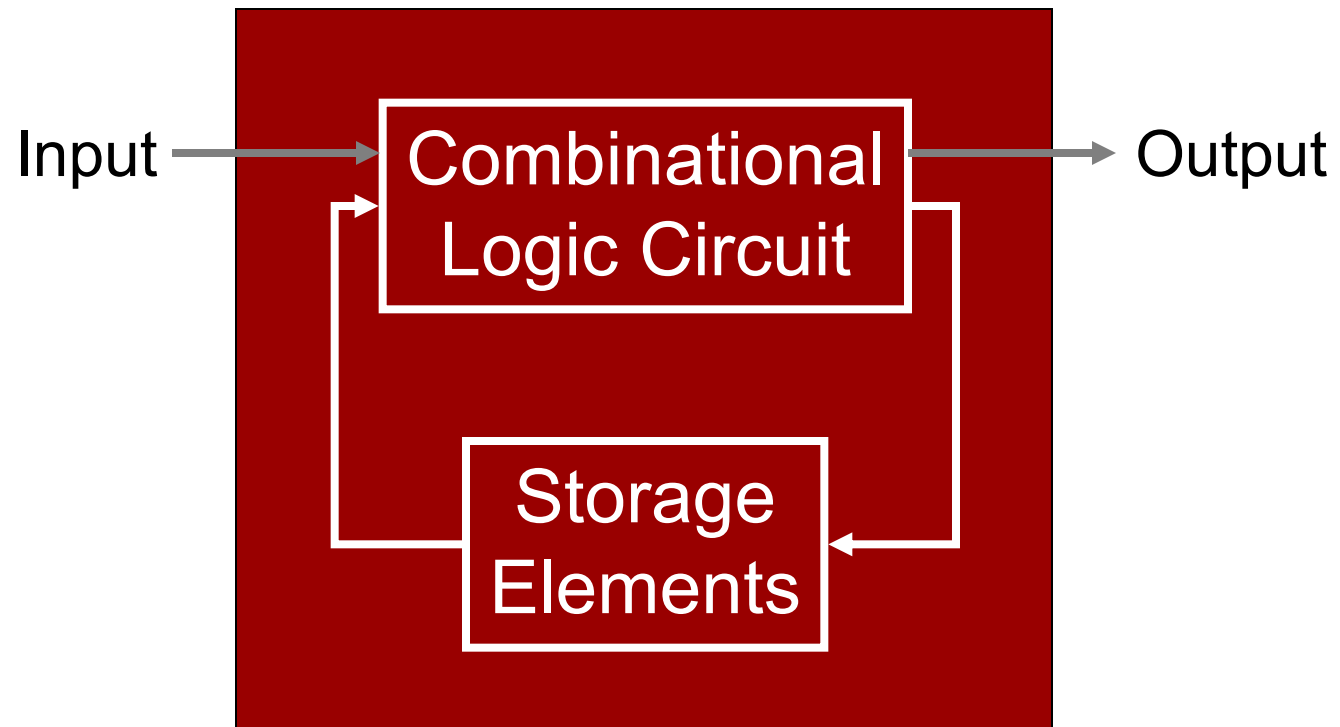
➤ Combinational Logic Circuits

- Make decisions
- Same inputs always produce same output
- Depends on what is **happening now**

➤ Sequential Logic Circuits

- Make decisions and store information
- Output depends on **inputs AND state**
- Depends on what has **happened in the past** as well as what is **happening now**

Sequential Logic Circuit

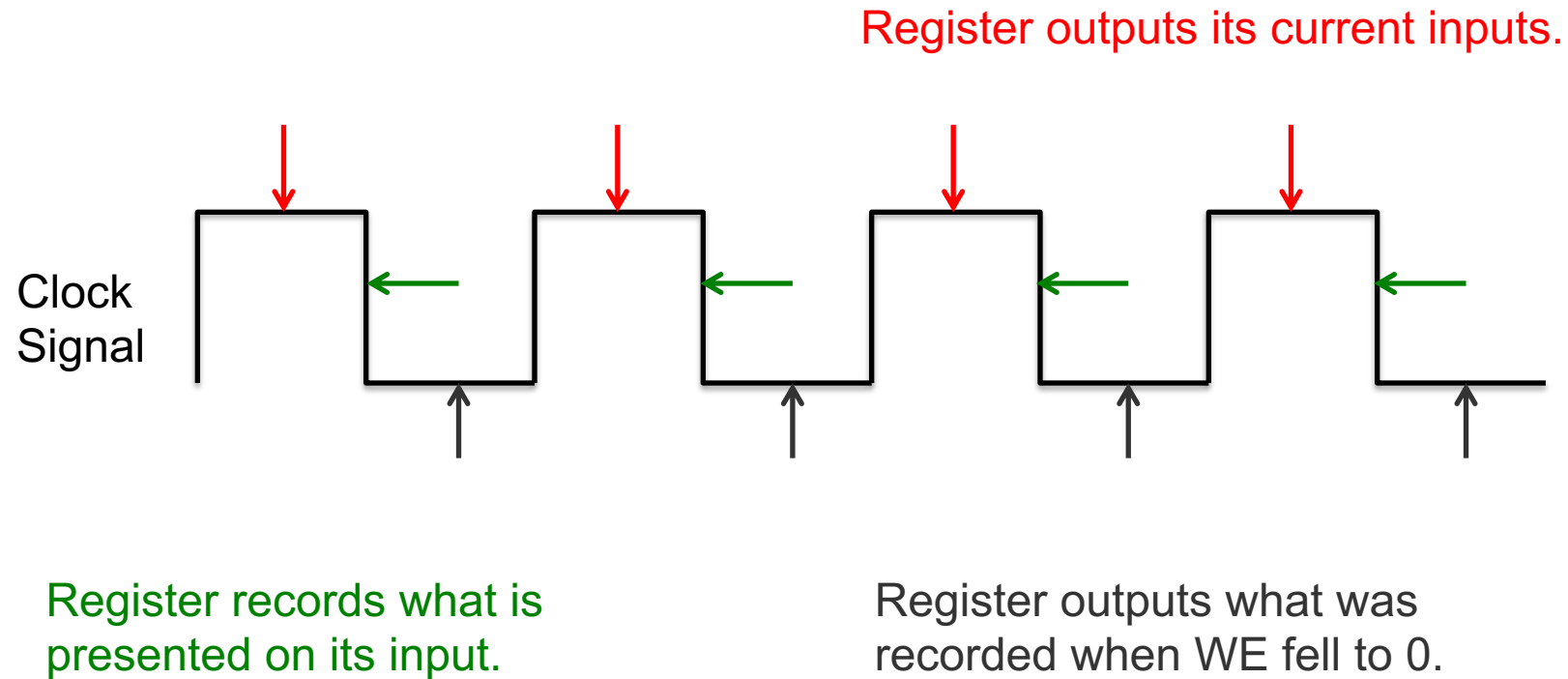


A Wrinkle With the D Latch

- Consider the Gated D latch...
- It is simple, and simple is what you want if you are going to implement a memory using 4 billion of them
- But there is a drawback...
- A Gated D Latch is a ***level-triggered*** device, that is, it will store whatever value is present on the input when the write enable goes from true to false

- To make the job of designing digital circuits easier it is useful to synchronize the operation using a clock
- But there's an issue when *outputs are used as inputs* to the same circuit element....
- How's that?

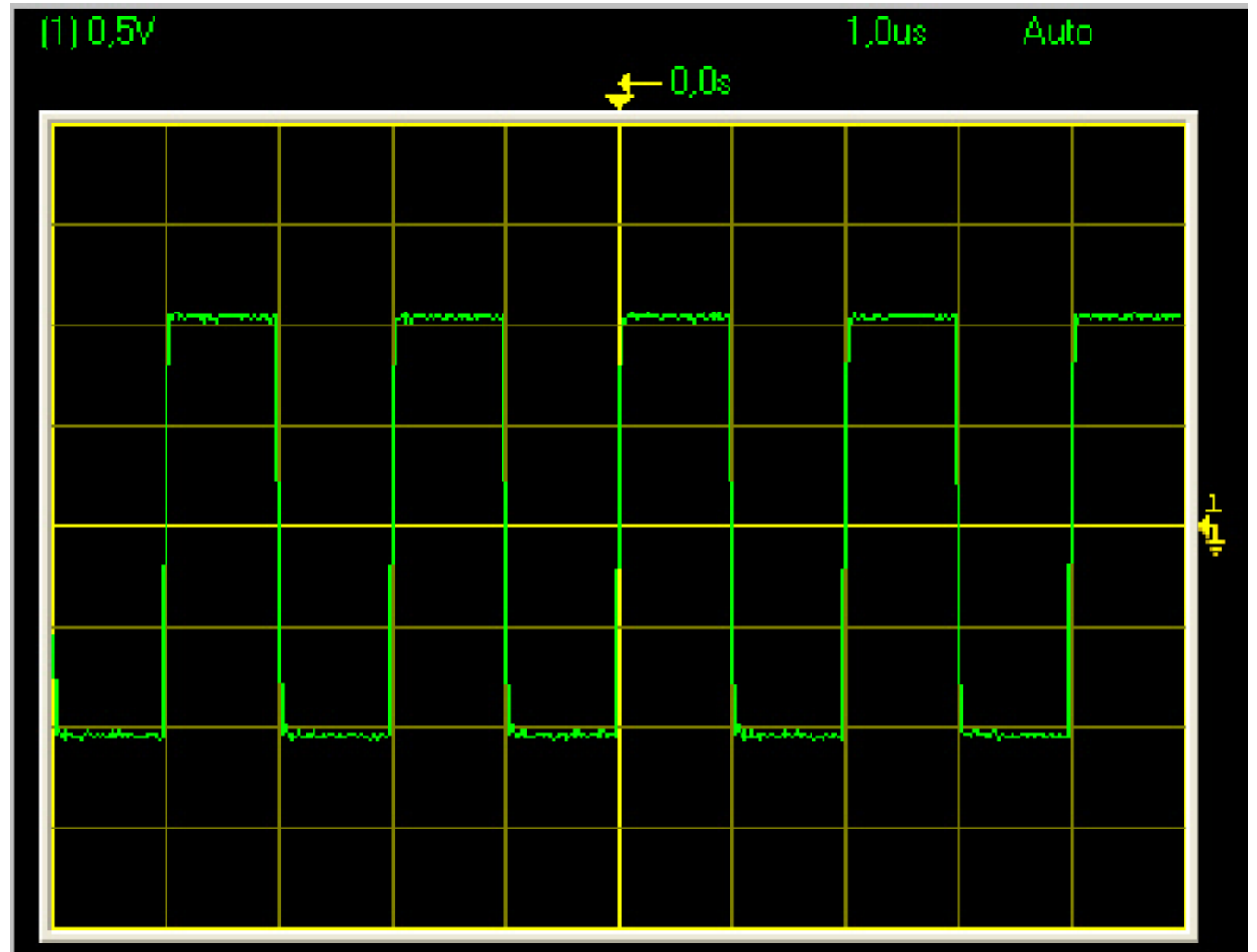
Clock Edges



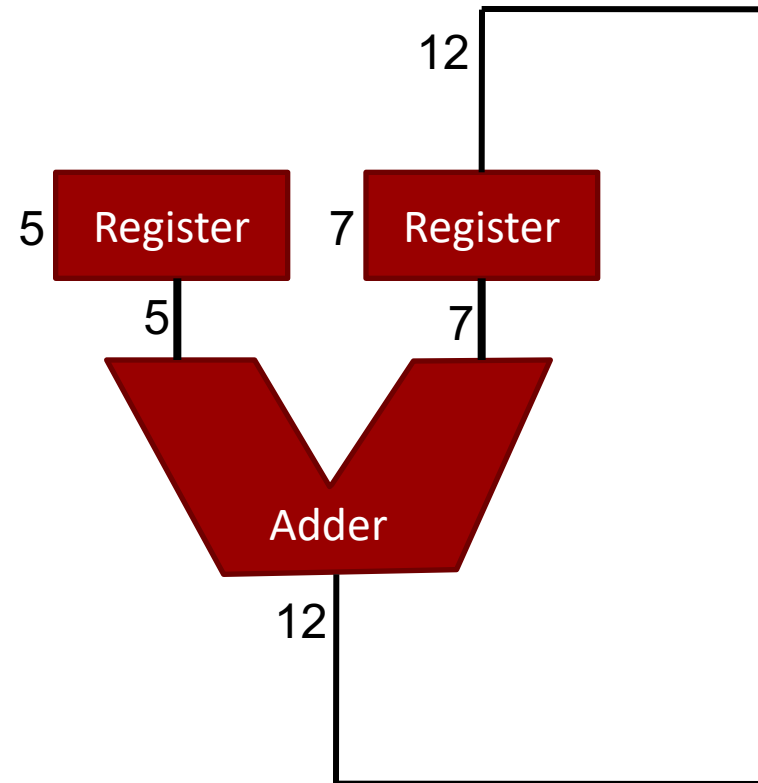
Now we just connect the clock to Write Enable....

Clock Signal (Square Wave)

- The vertical edges are not exactly vertical
- There is noise at the top and bottom of the waveforms
- Even the oscilloscope probe cables can induce noise into the circuit
- It's a luxury for computer scientists to be able to presume the signal is just "low" or "high"



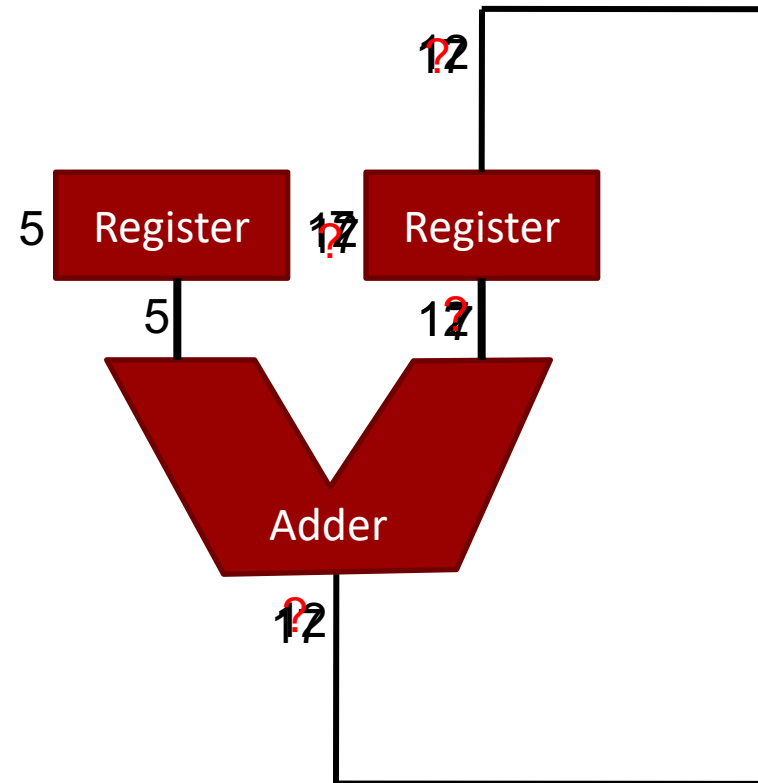
The Problem



And here we stay until the clock transitions

Clock signal **Low**
Registers output recorded value

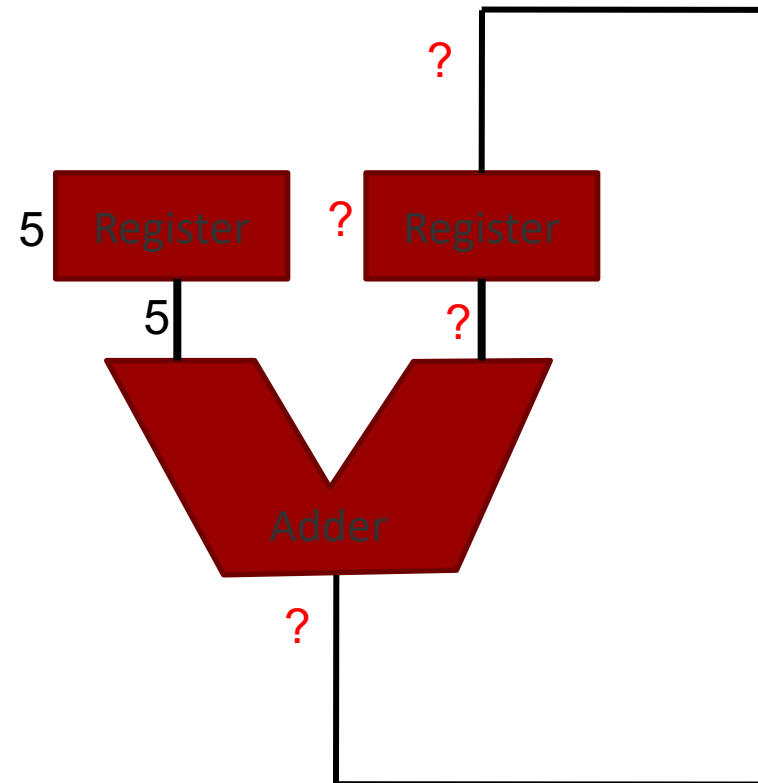
The Problem



And we stop when??

Clock signal **High**
Registers output recorded value

The Problem

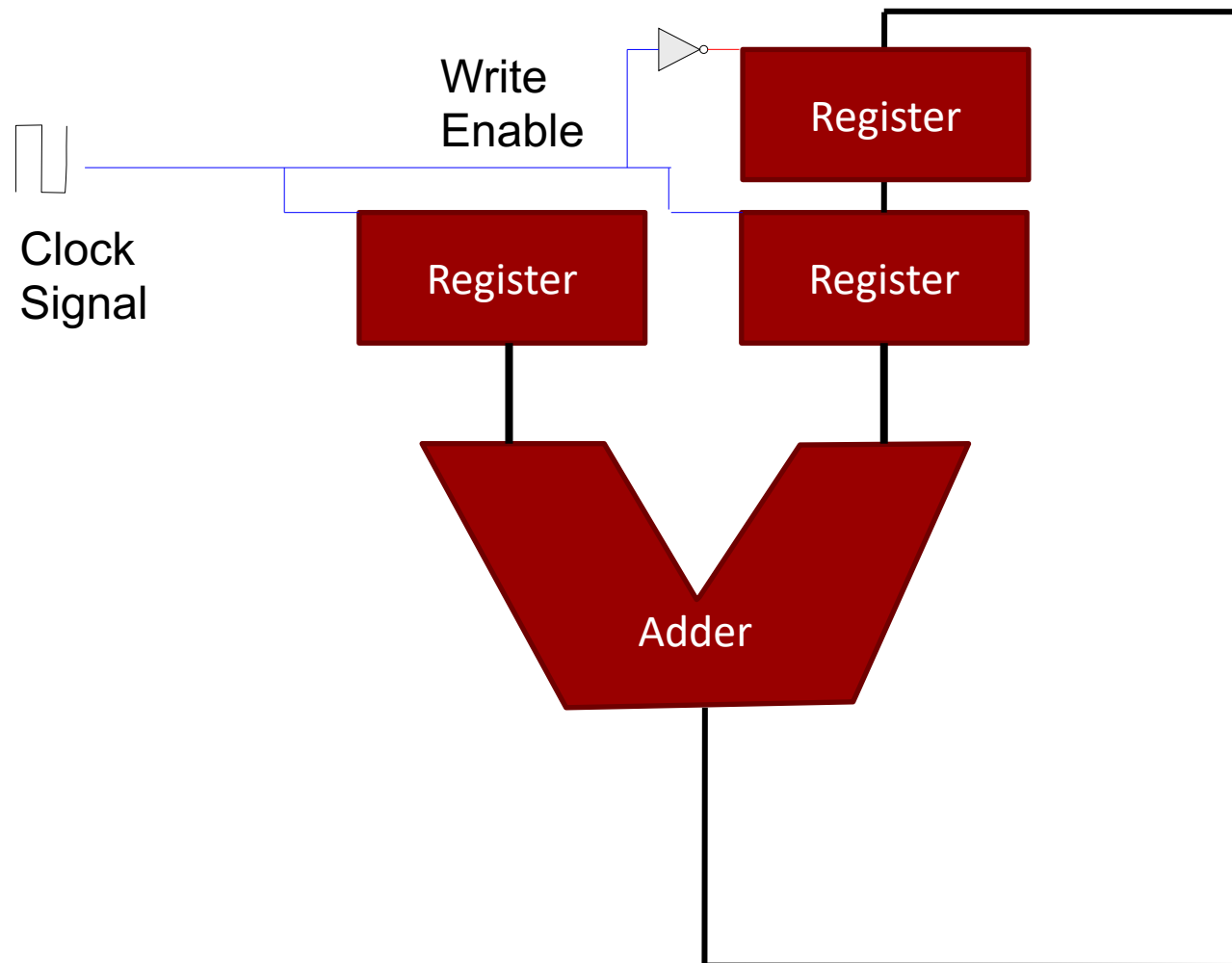


And what value is in the register?

Clock signal **LOW**

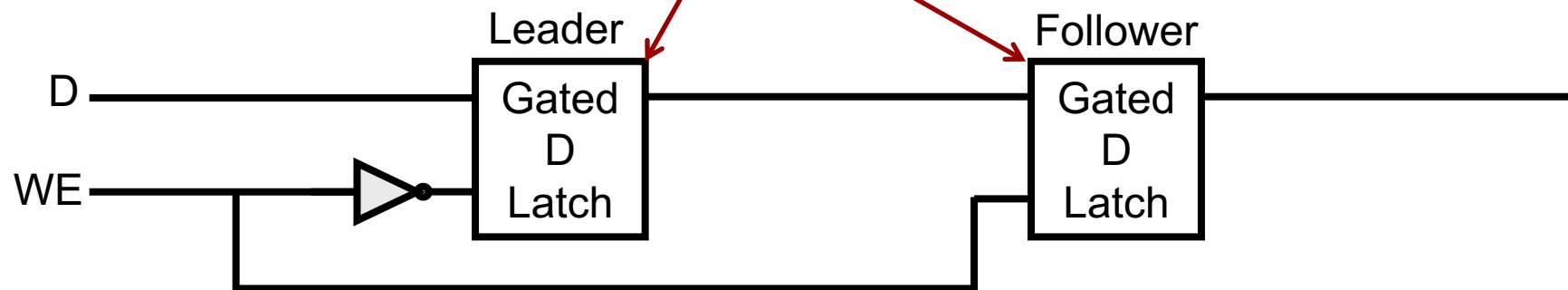
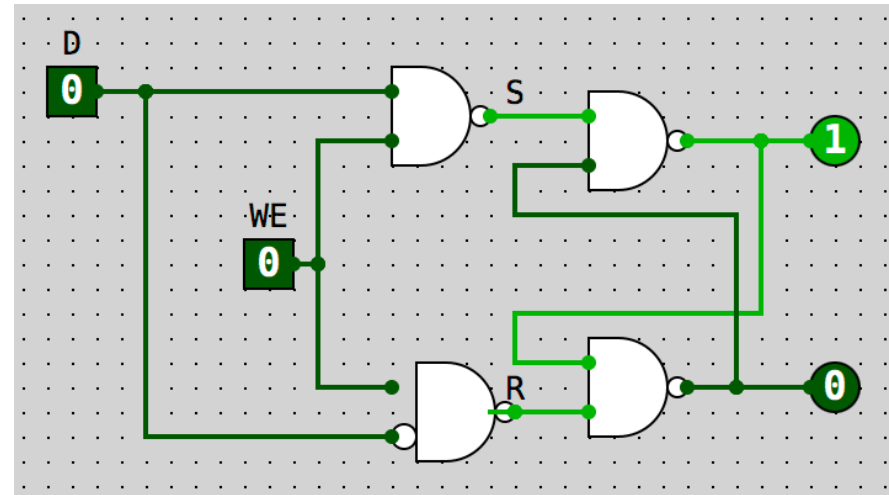
Registers output recorded value

Possible Solution?



Leader-Follower Flip Flop

Remember our
Gated D Latch?

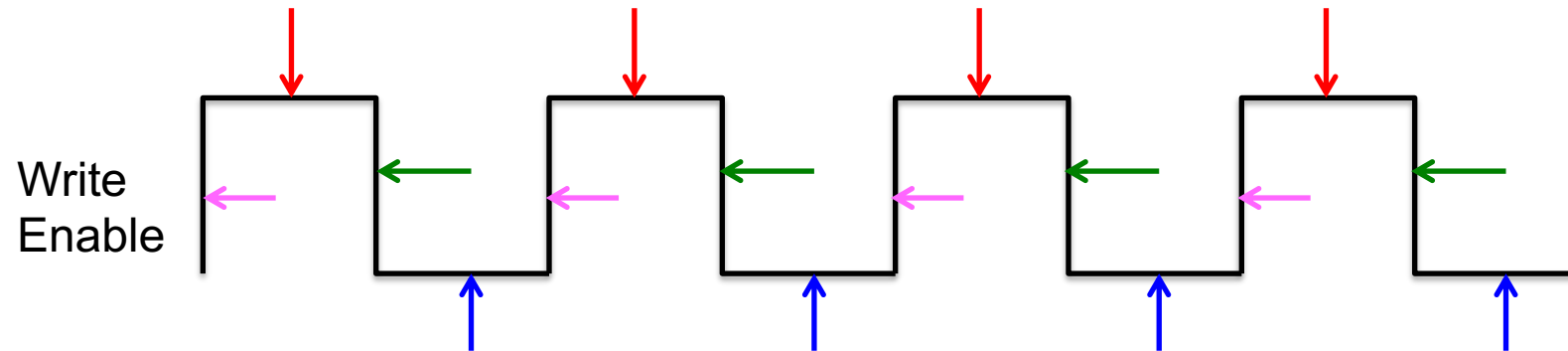


In the textbook this circuit is referred to as a Master/Slave flip flop

Clock Edges and Leader-Follower

Leader register records what is presented on its input.

Follower register outputs its inputs.
Leader register outputs what was recorded when WE rose to 1.

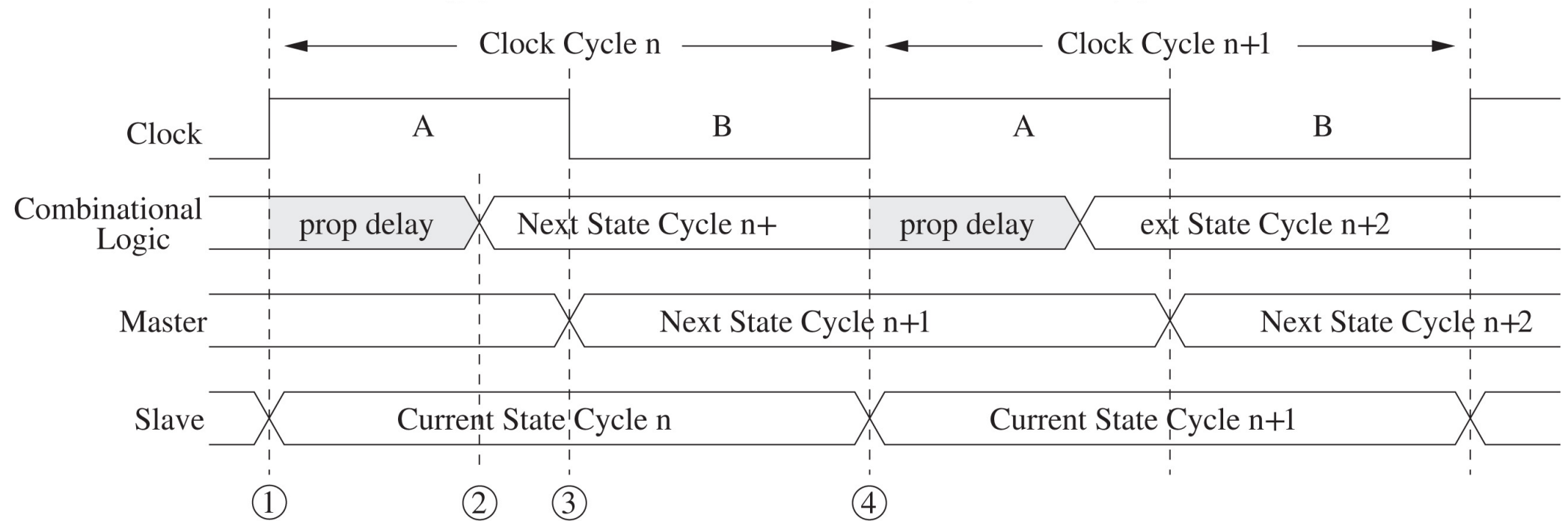


Follower register records what is presented on its input.

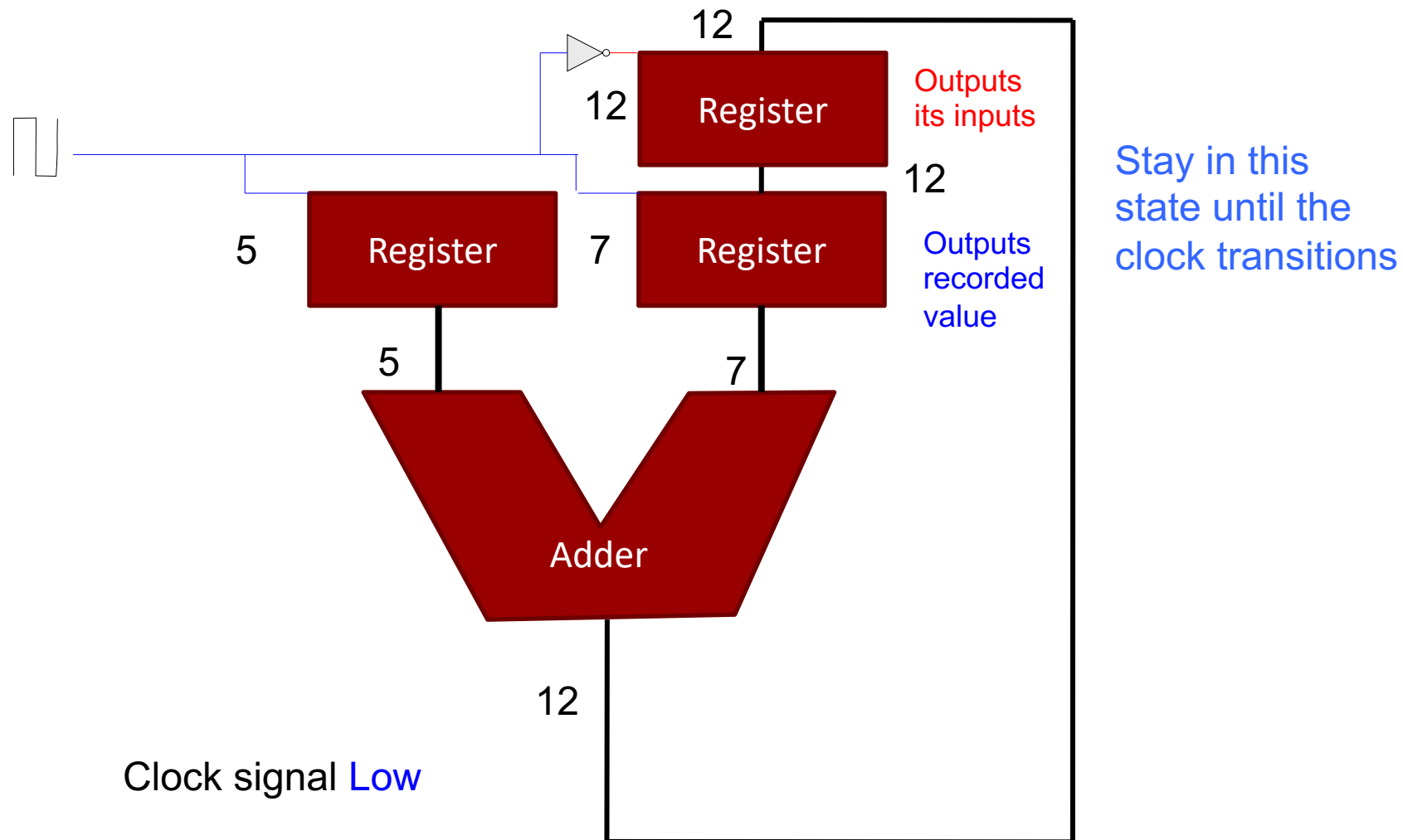
Follower register outputs what was recorded when WE fell to 0.
Leader register outputs its inputs.

Now we just connect Write Enable to the clock....

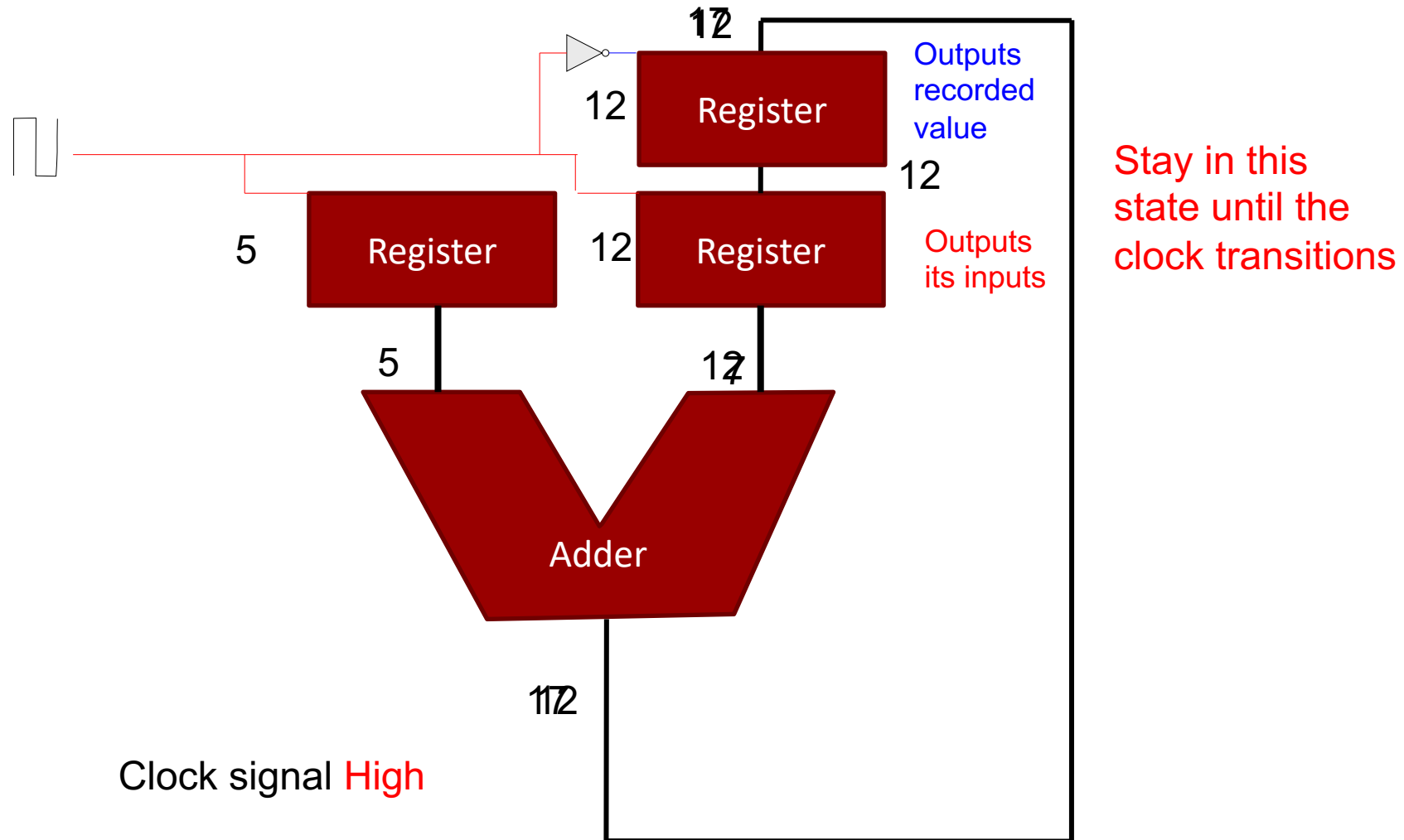
Another way to look at it



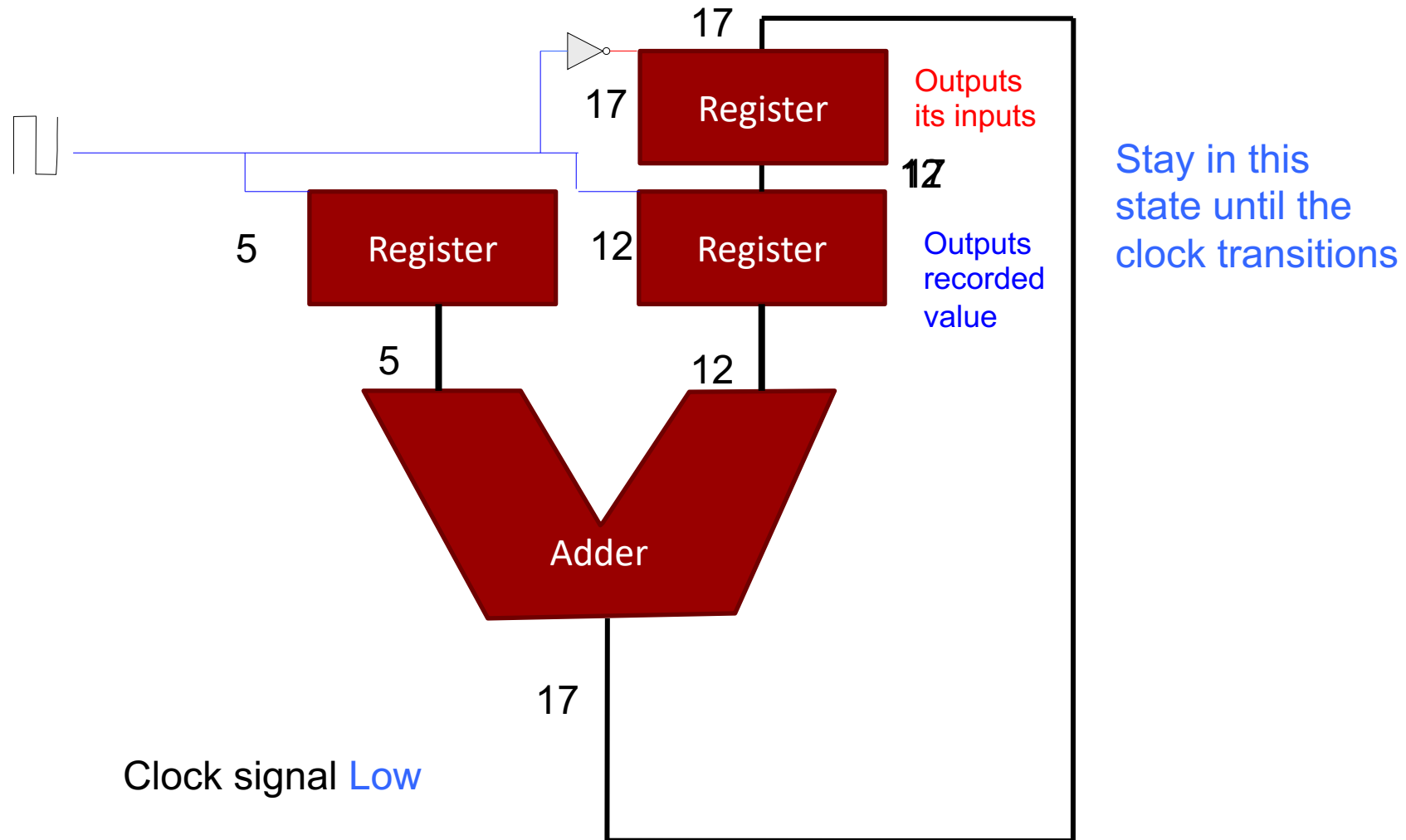
Possible Solution?



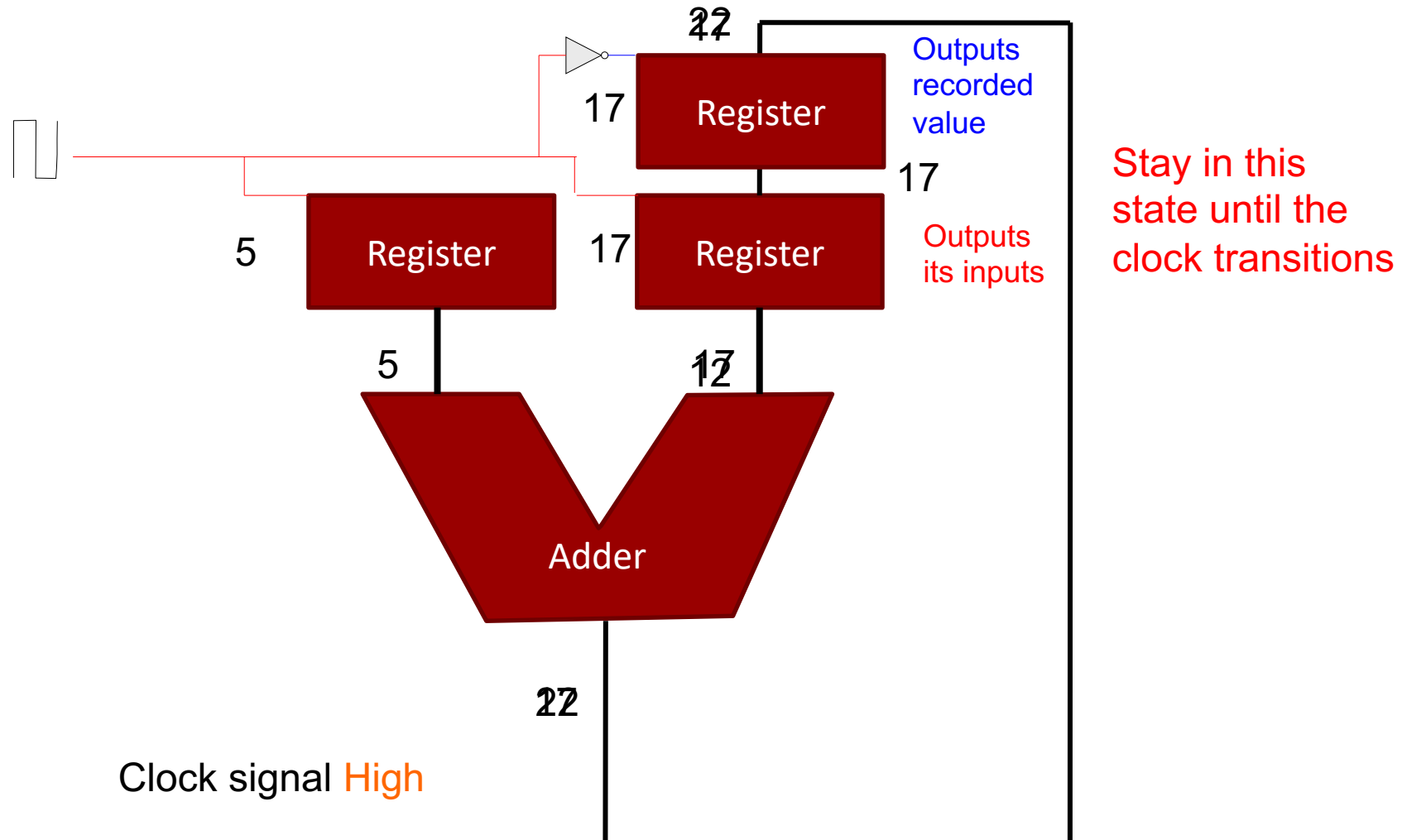
Possible Solution?



Possible Solution?



Possible Solution?



Terminology

Patt

Gated D Latch

Leader-Follower Flip-Flop

Master-Slave Flip-flop

Circuitsim

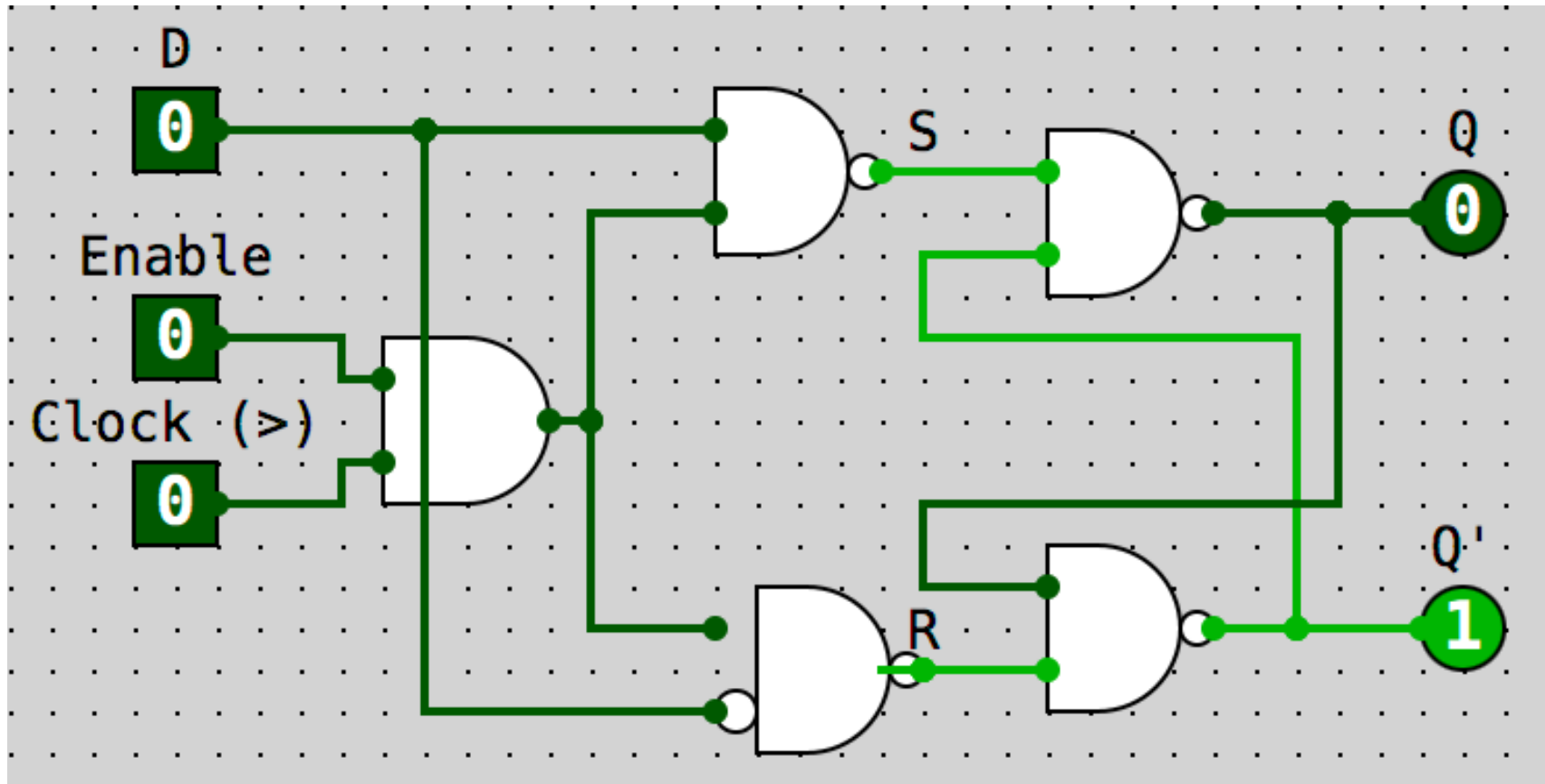
Build it yourself!

D Flip-Flop

-or-

Register

How Does Enable Work?



Level-triggered sequential circuits can yield unexpected results if

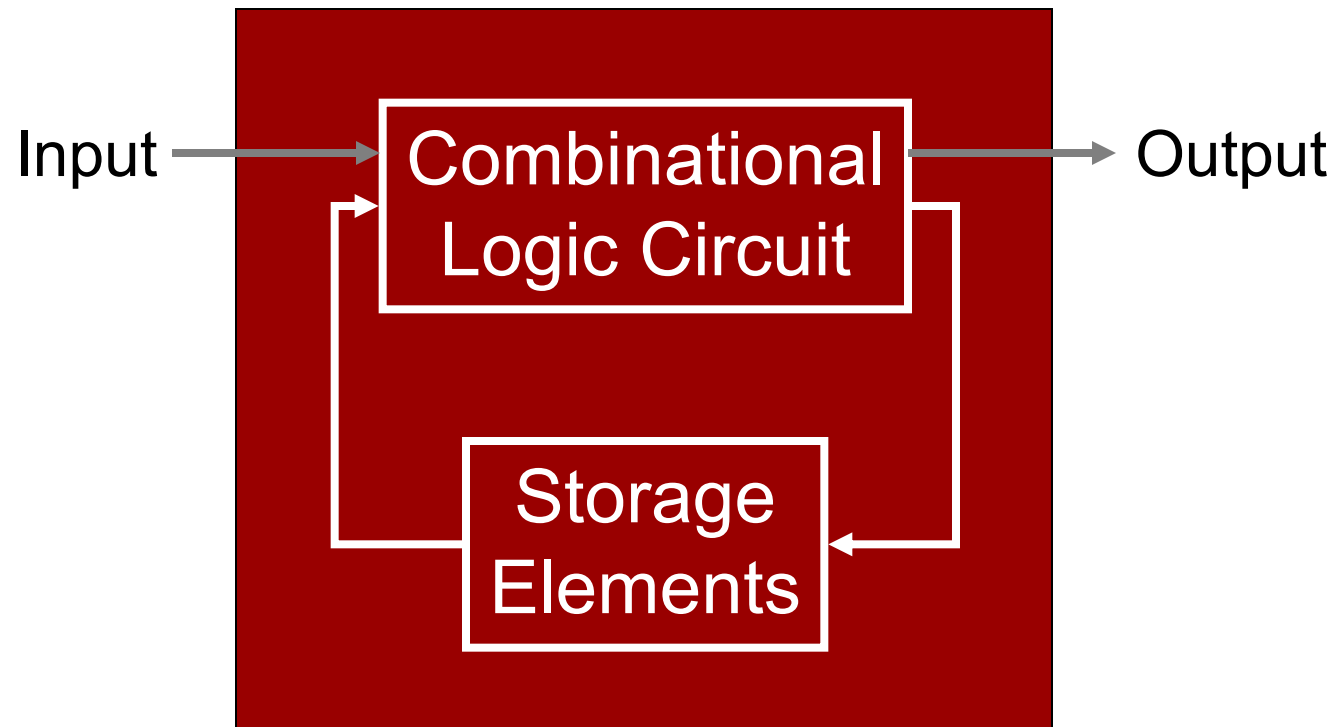
- A. The device is presented with different inputs in subsequent clock cycles
- B. The input of the device is derived from manually operated switches
- C. The clock signal to the device is stopped
- D. The output of the device is used in computing the input to the same device



Now For Finite State Machines

➤ Which is what we build with Sequential Logic...

Sequential Logic Circuit

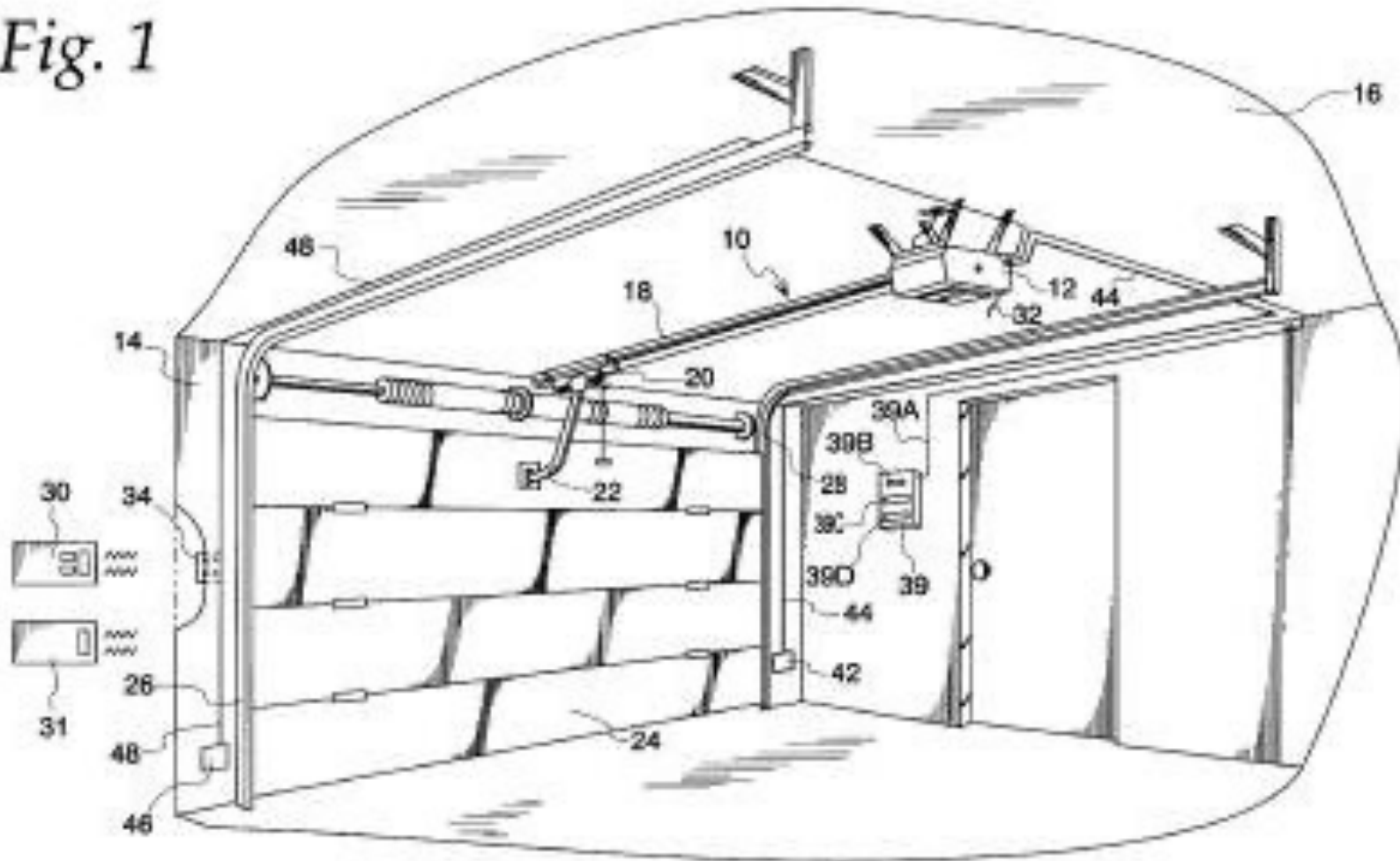


Concept of State

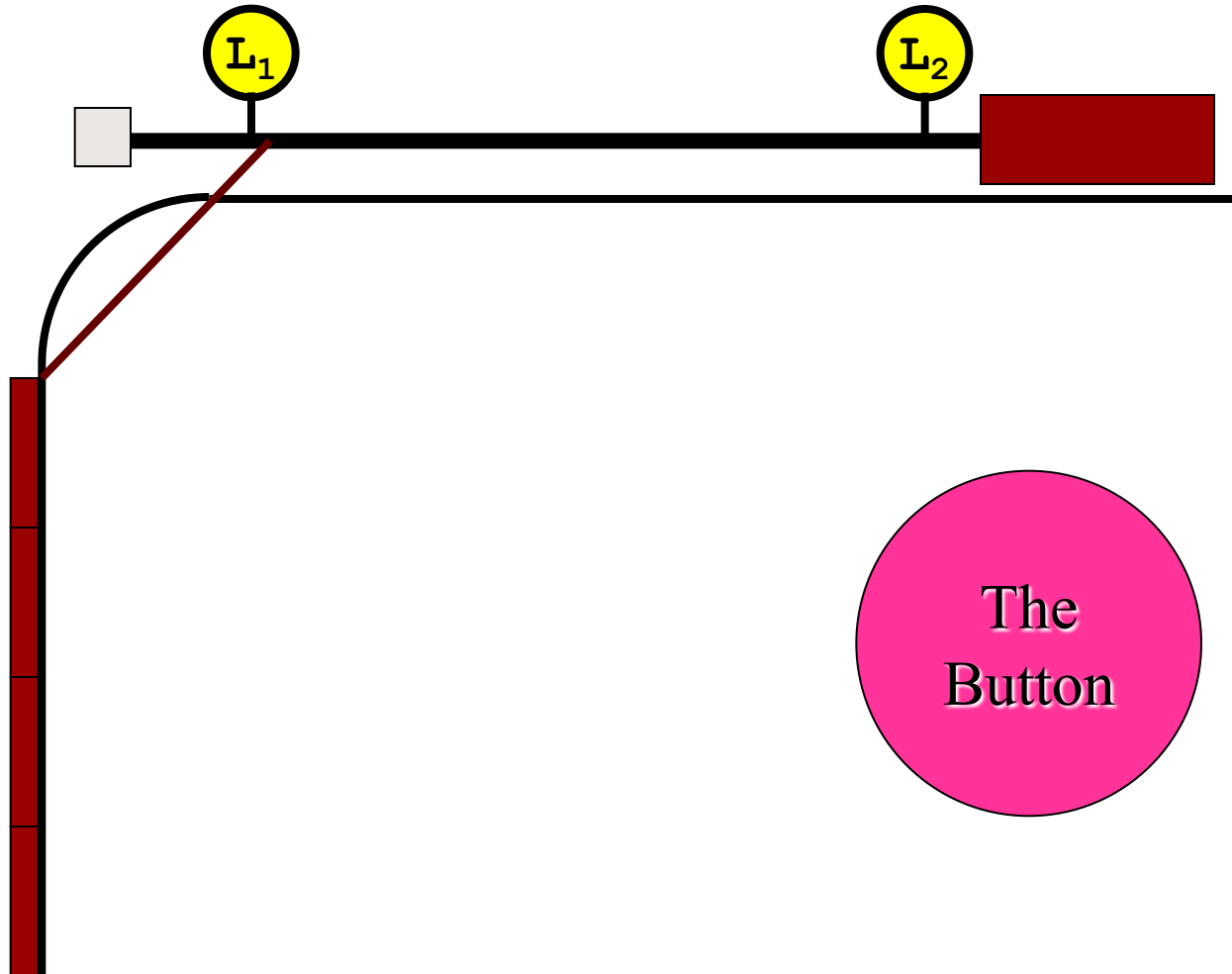
- Vending Machines
- Adding Machines
- Traffic Signal Controls
- Parking Lot Controls
- Combination Locks
- Computers
- Garage Door Openers

A Garage Door Opener

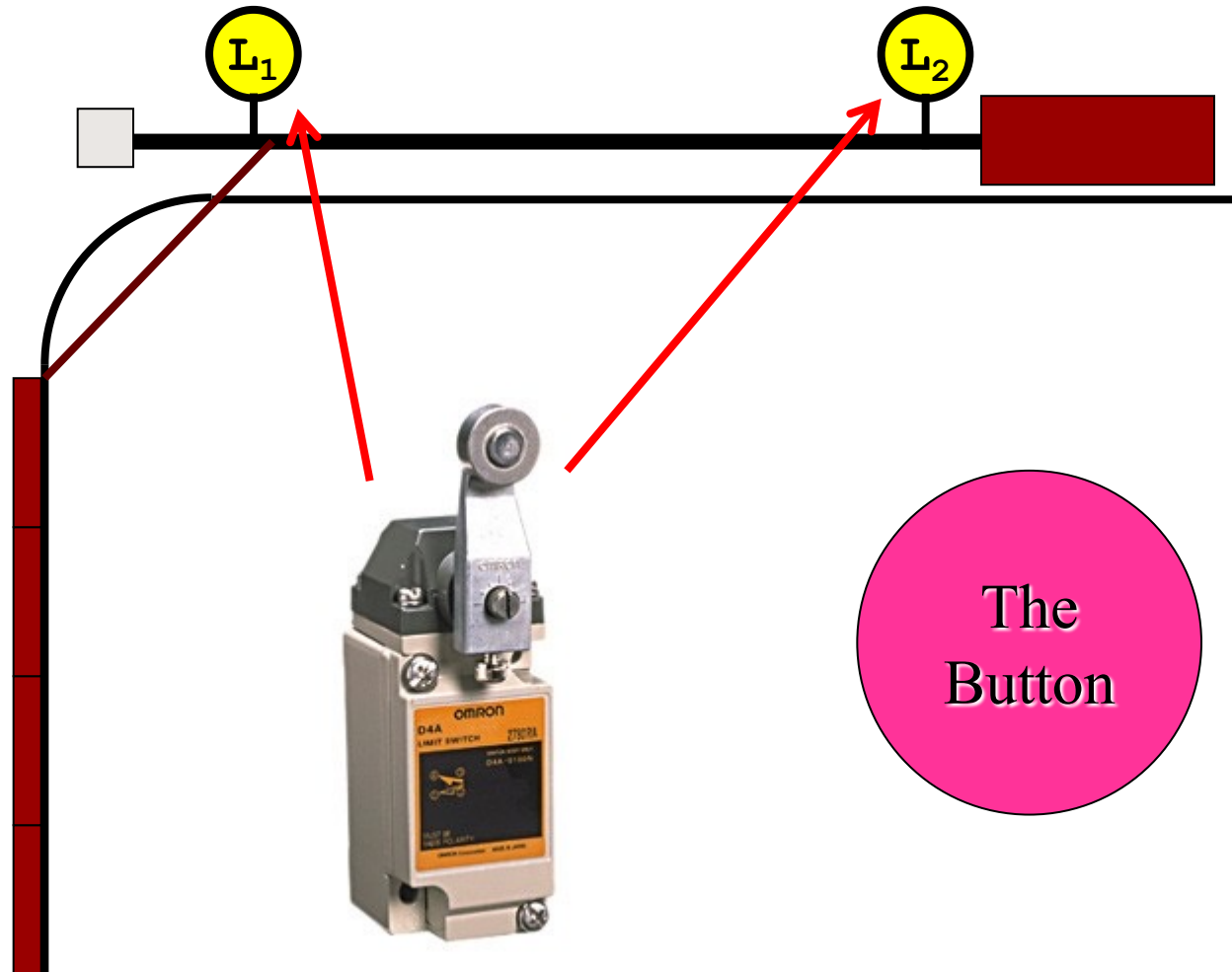
Fig. 1



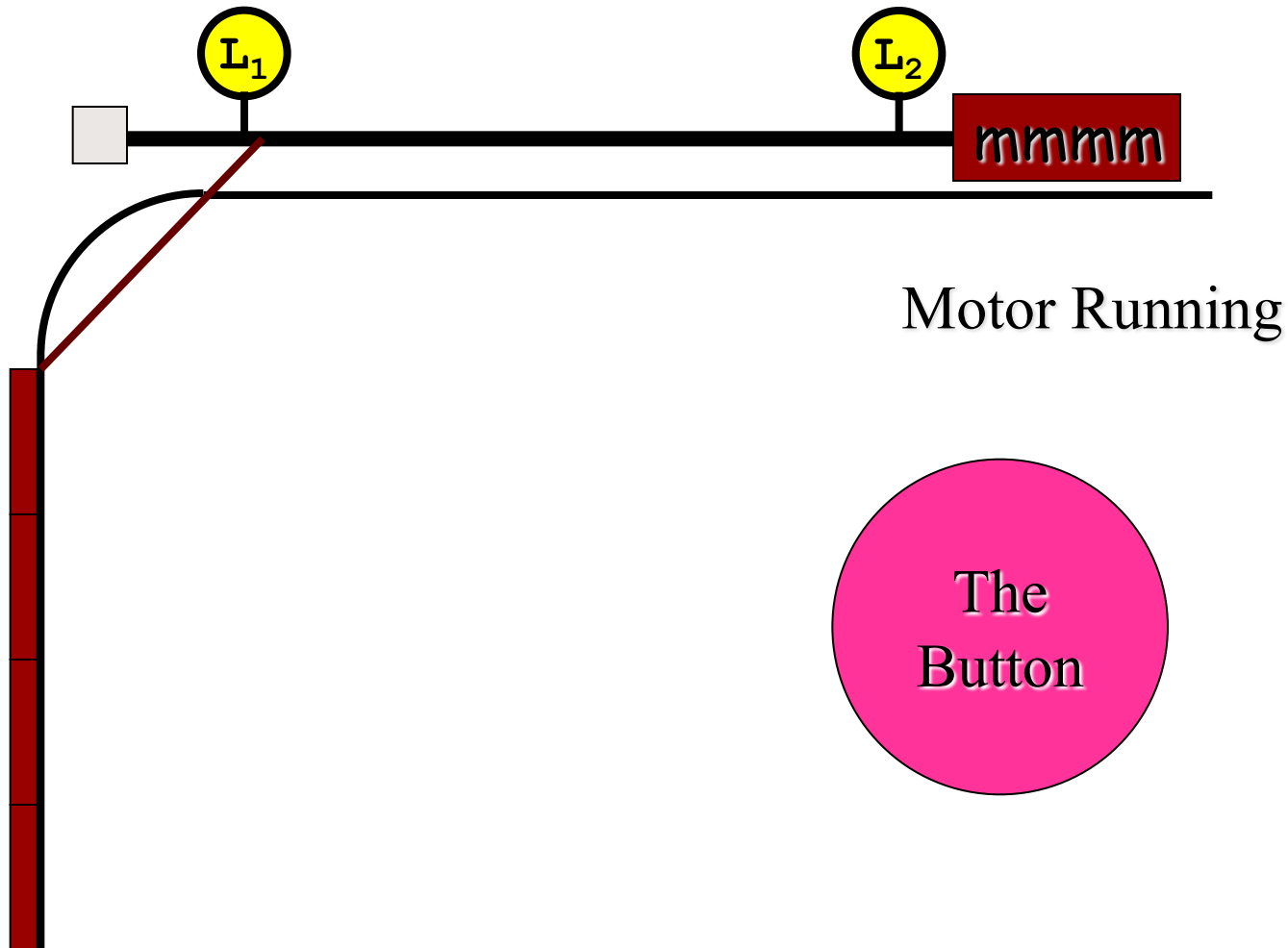
My Garage Door



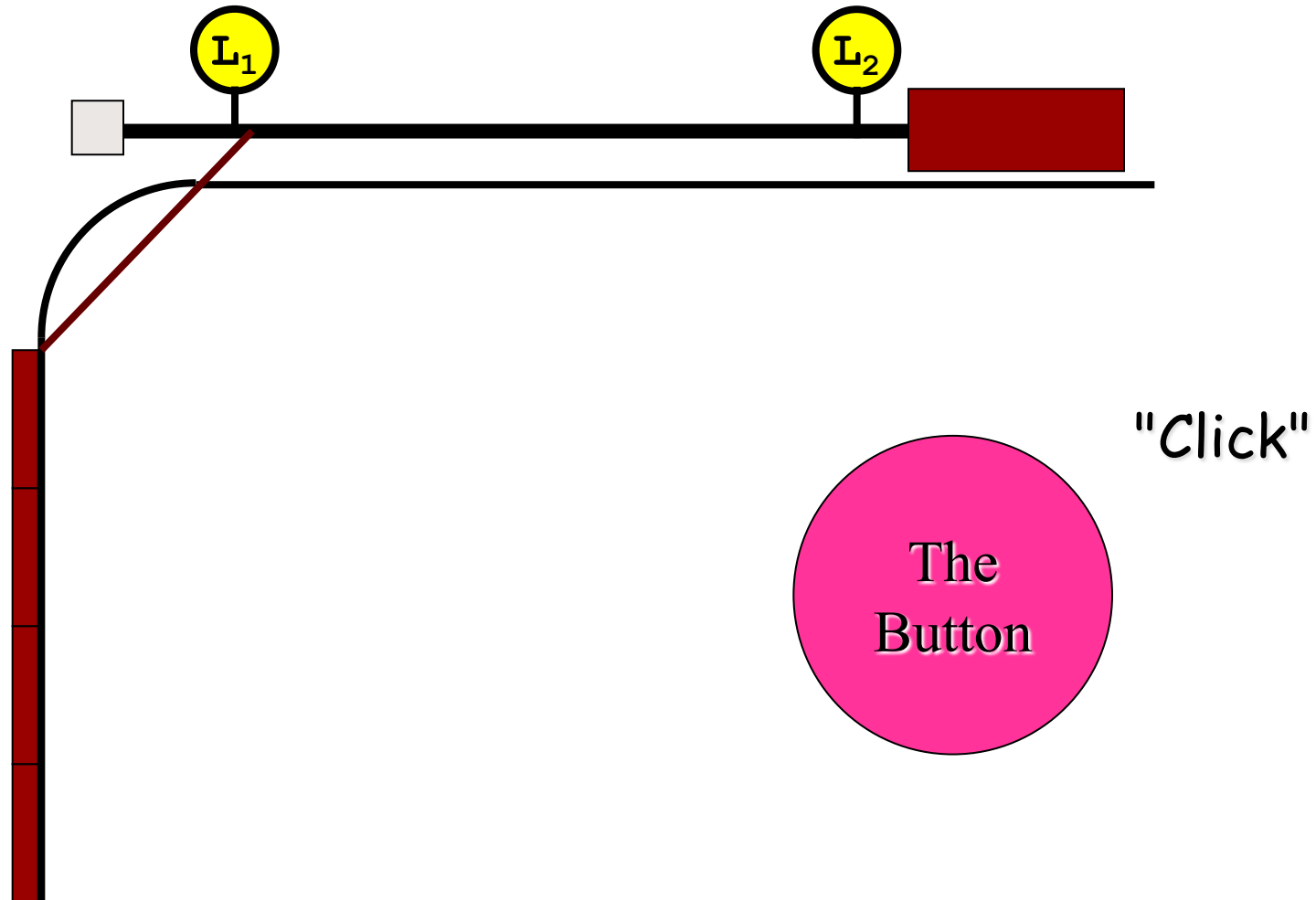
My Garage Door



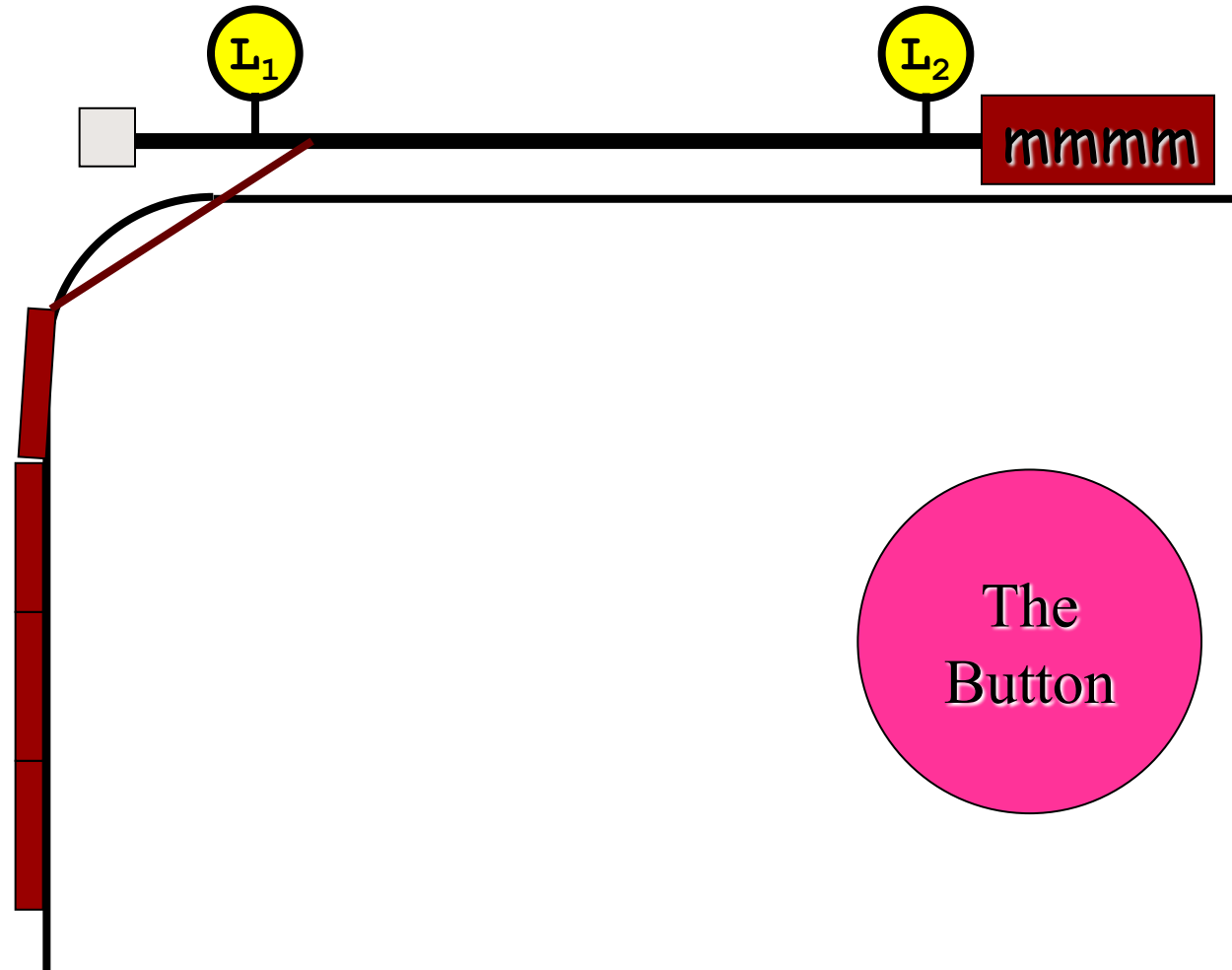
My Garage Door



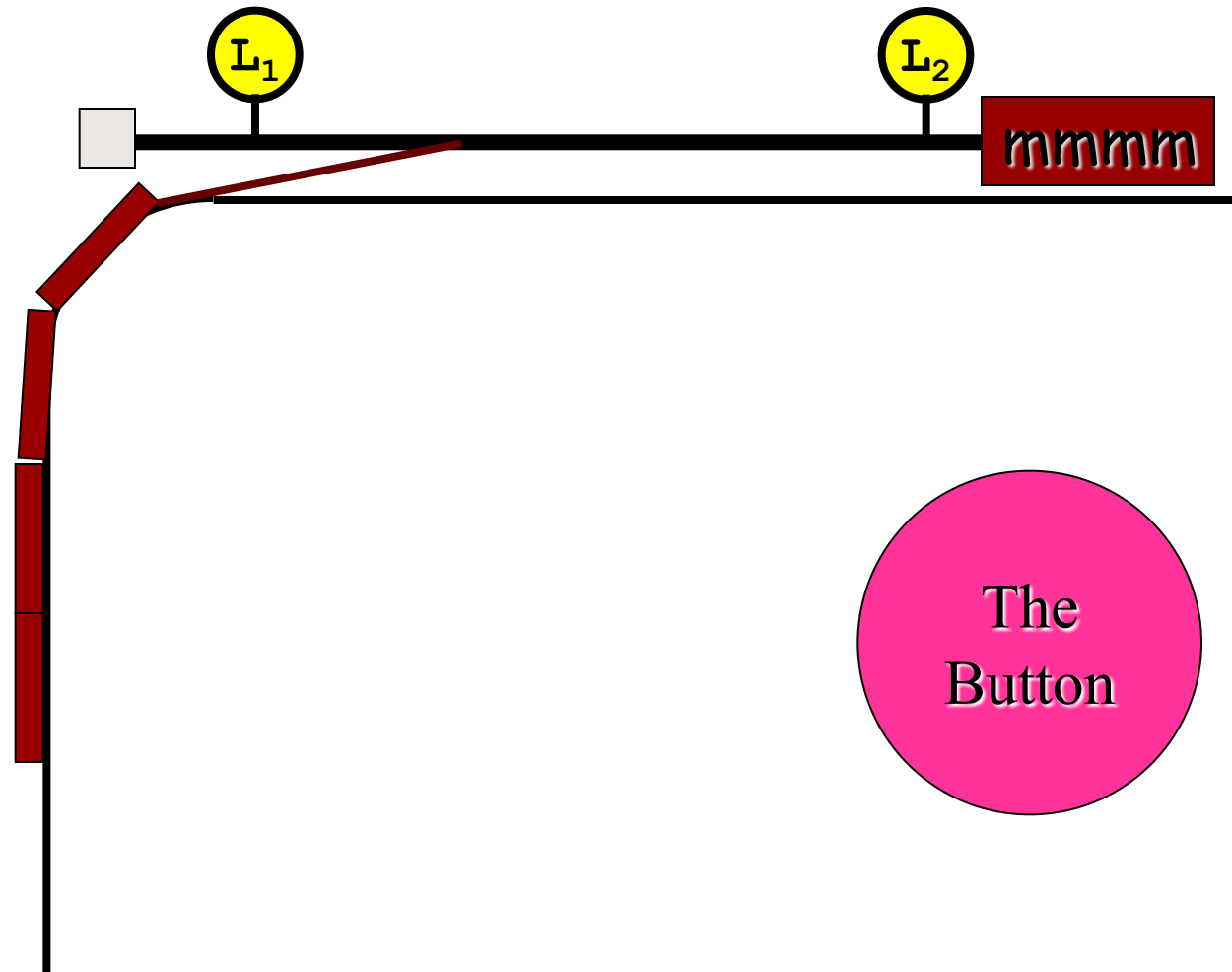
My Garage Door



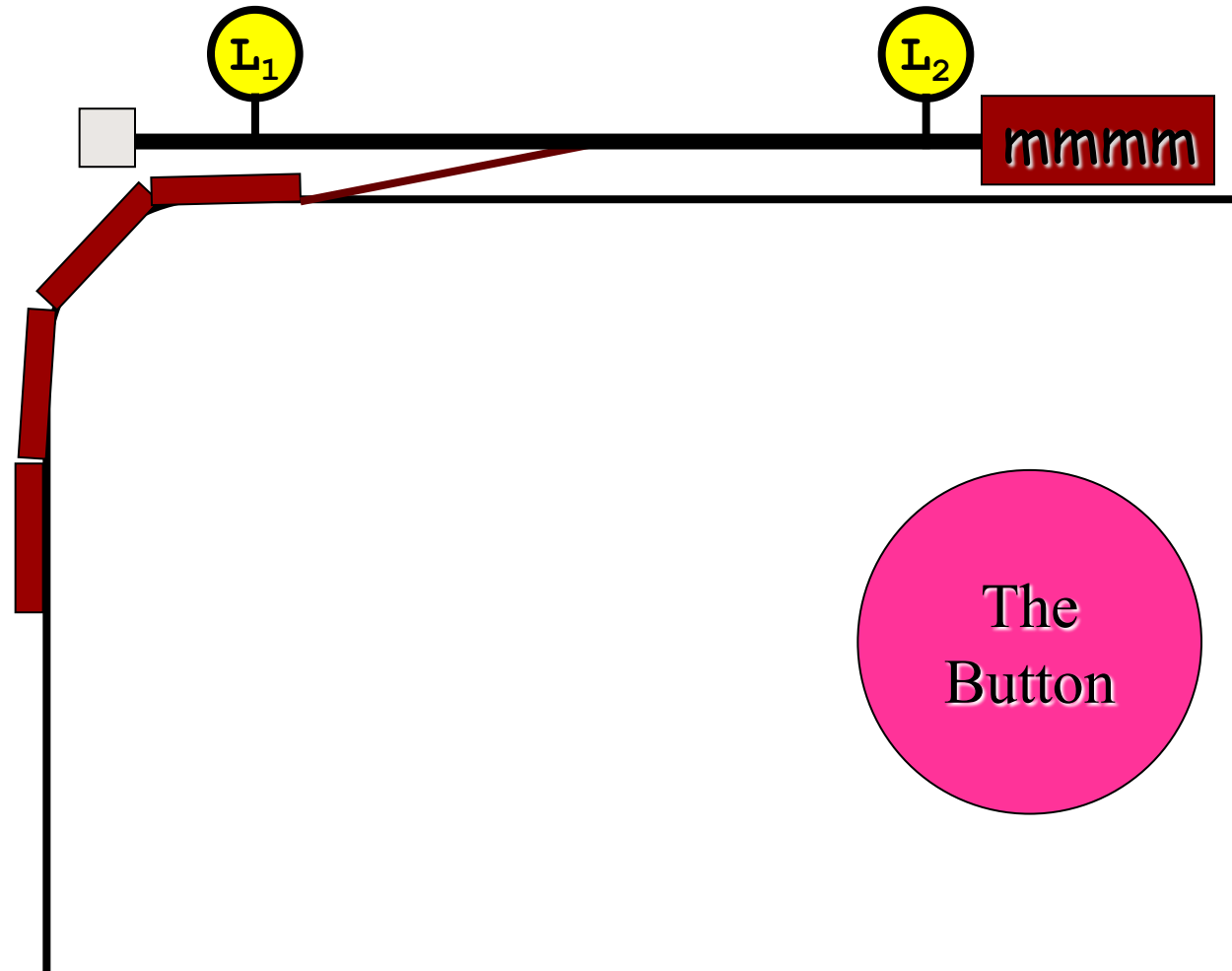
My Garage Door



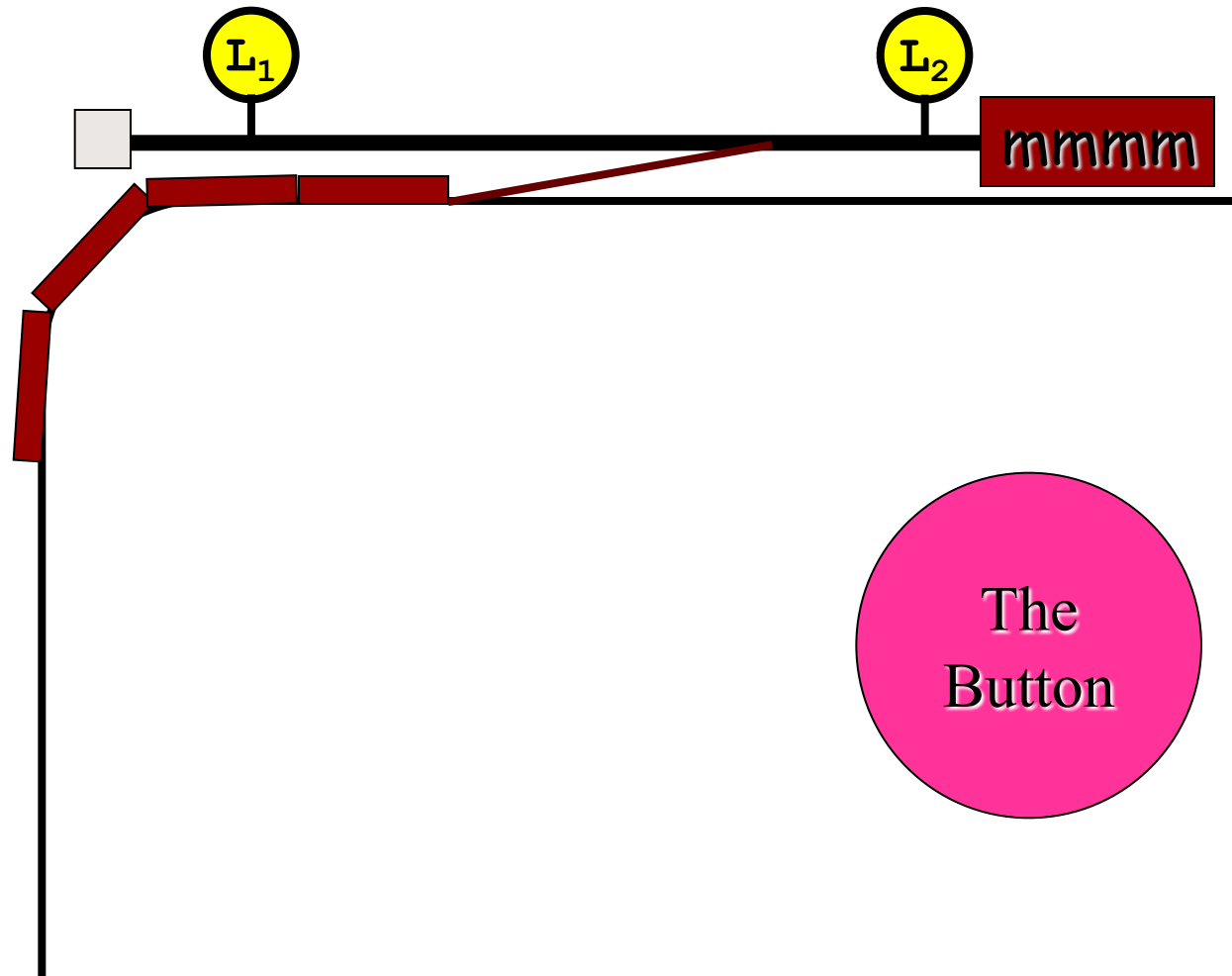
My Garage Door



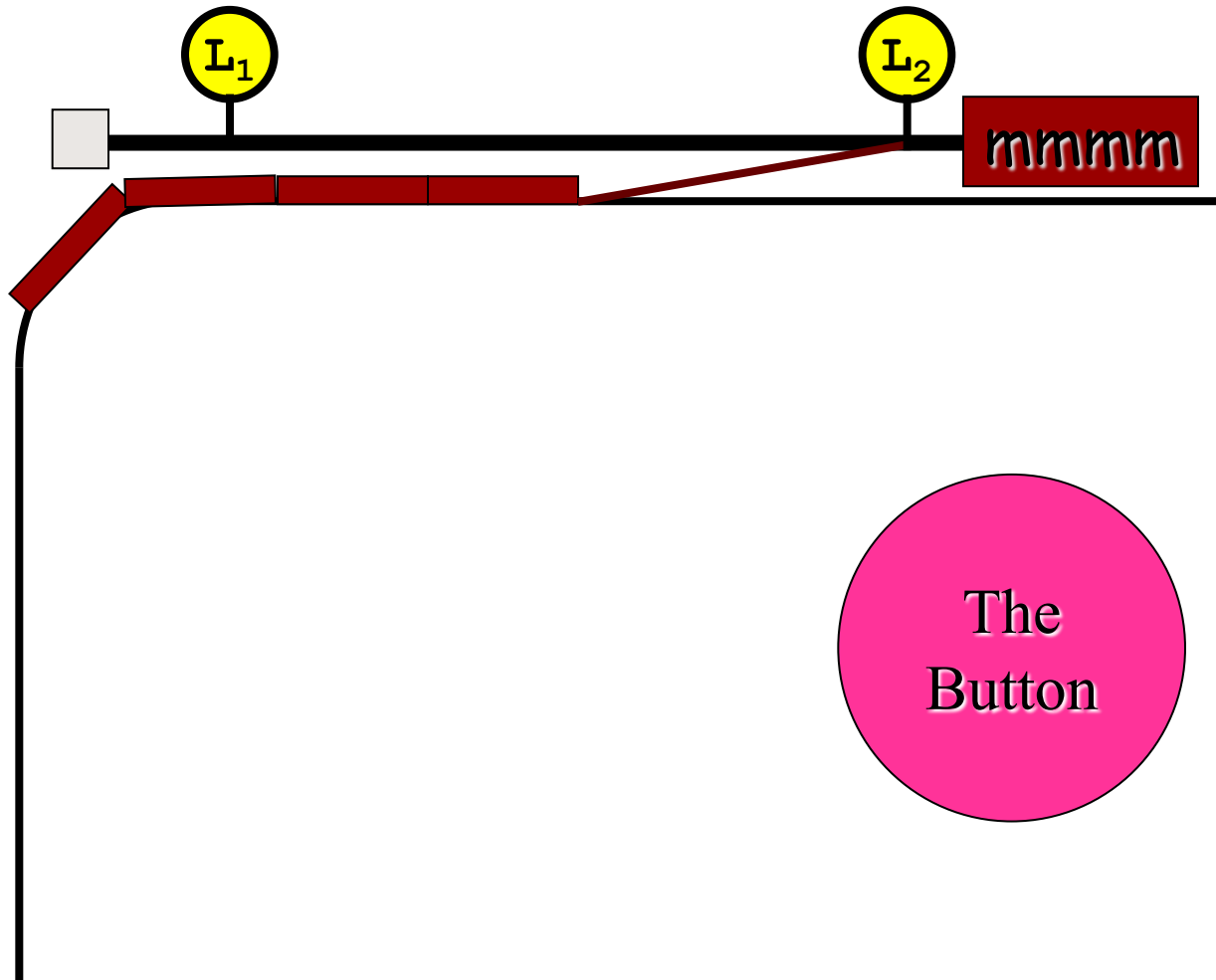
My Garage Door



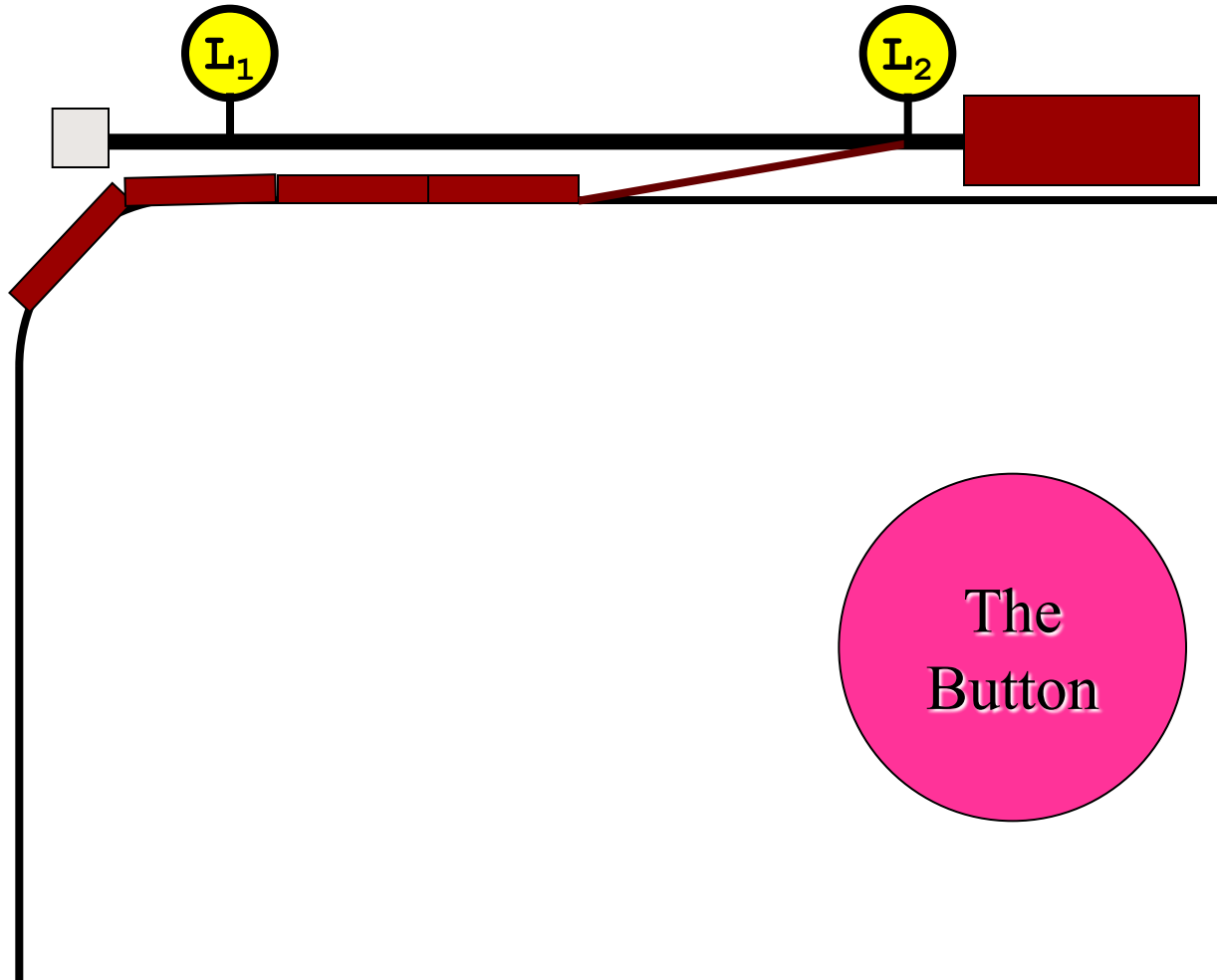
My Garage Door



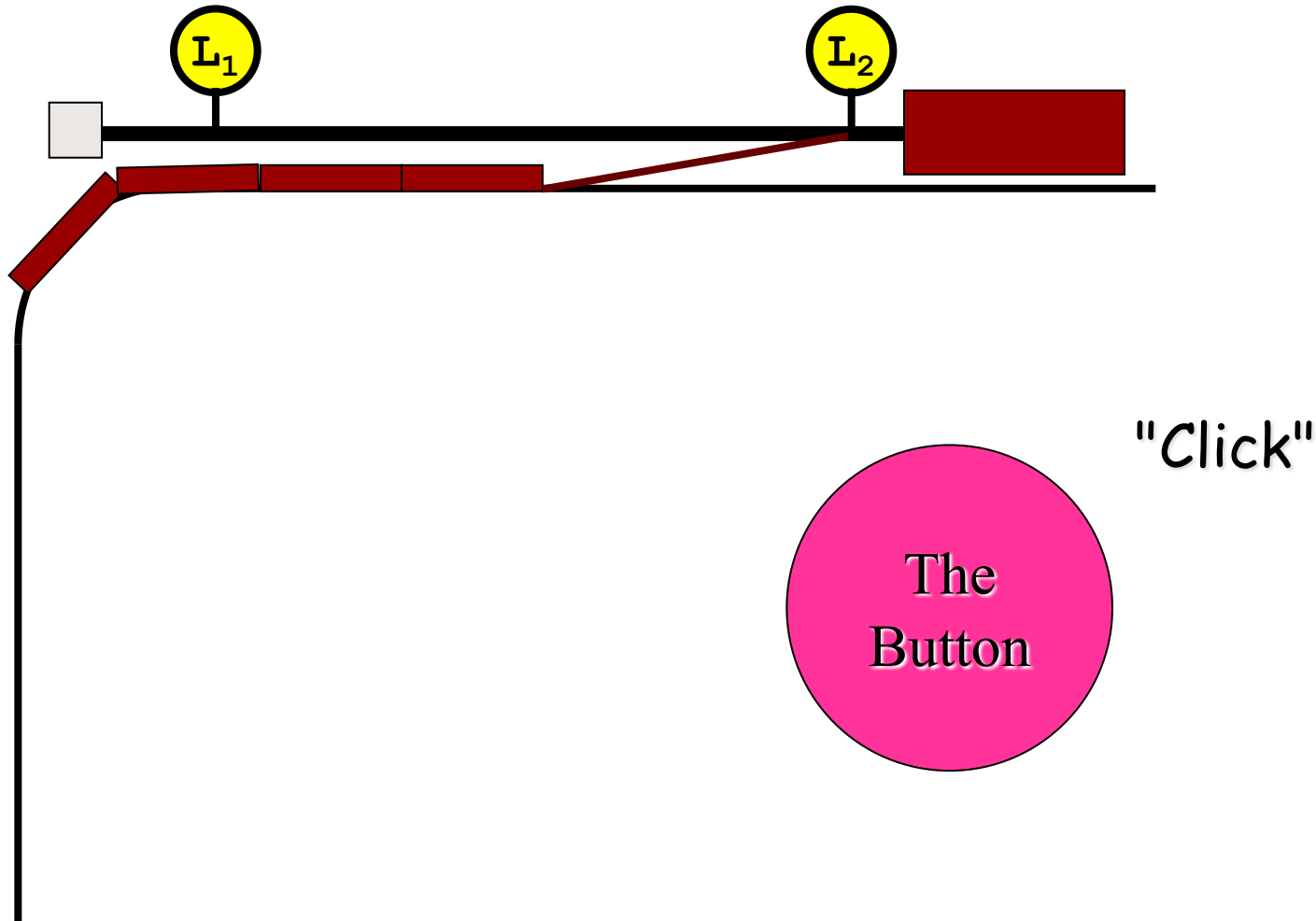
My Garage Door



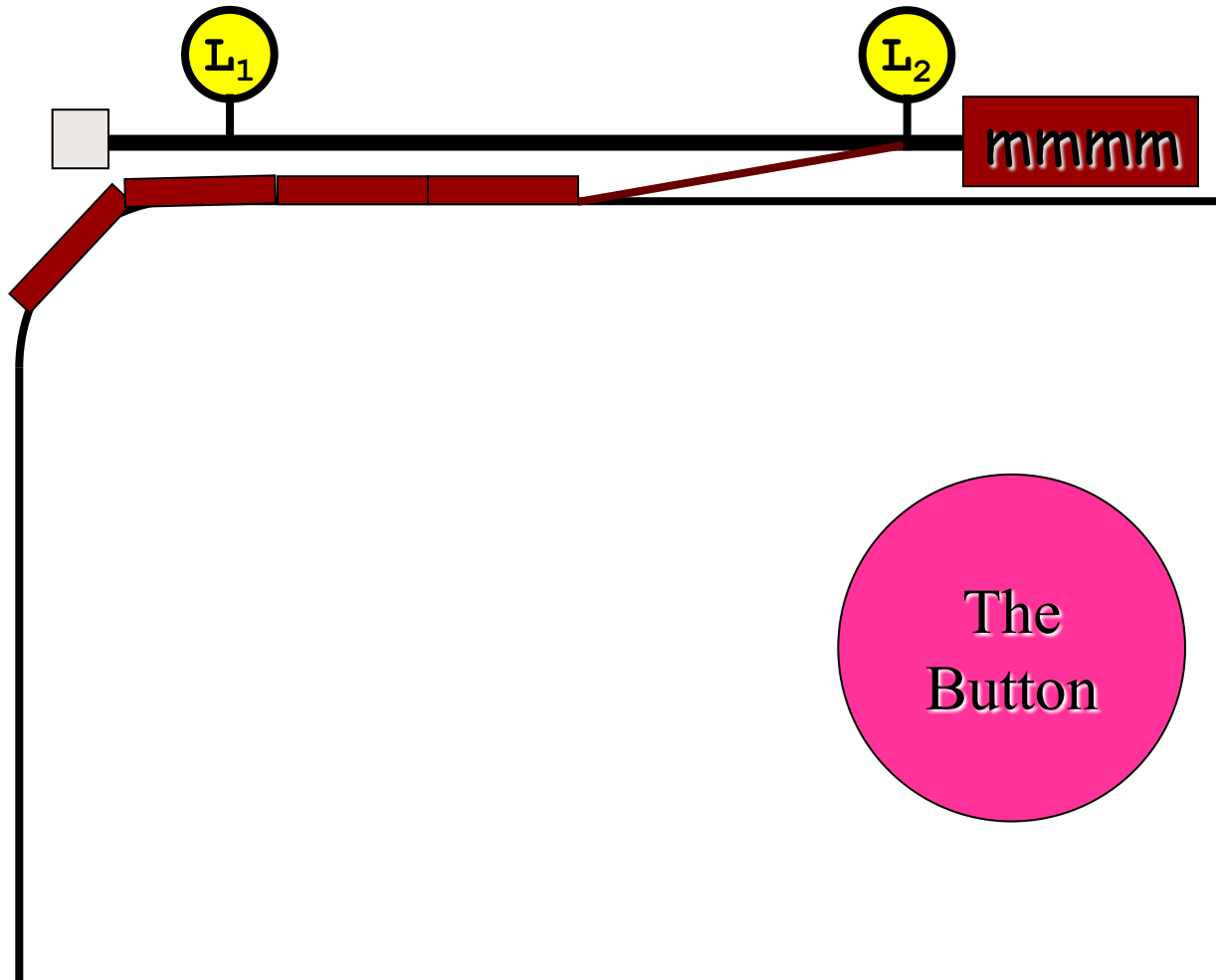
My Garage Door



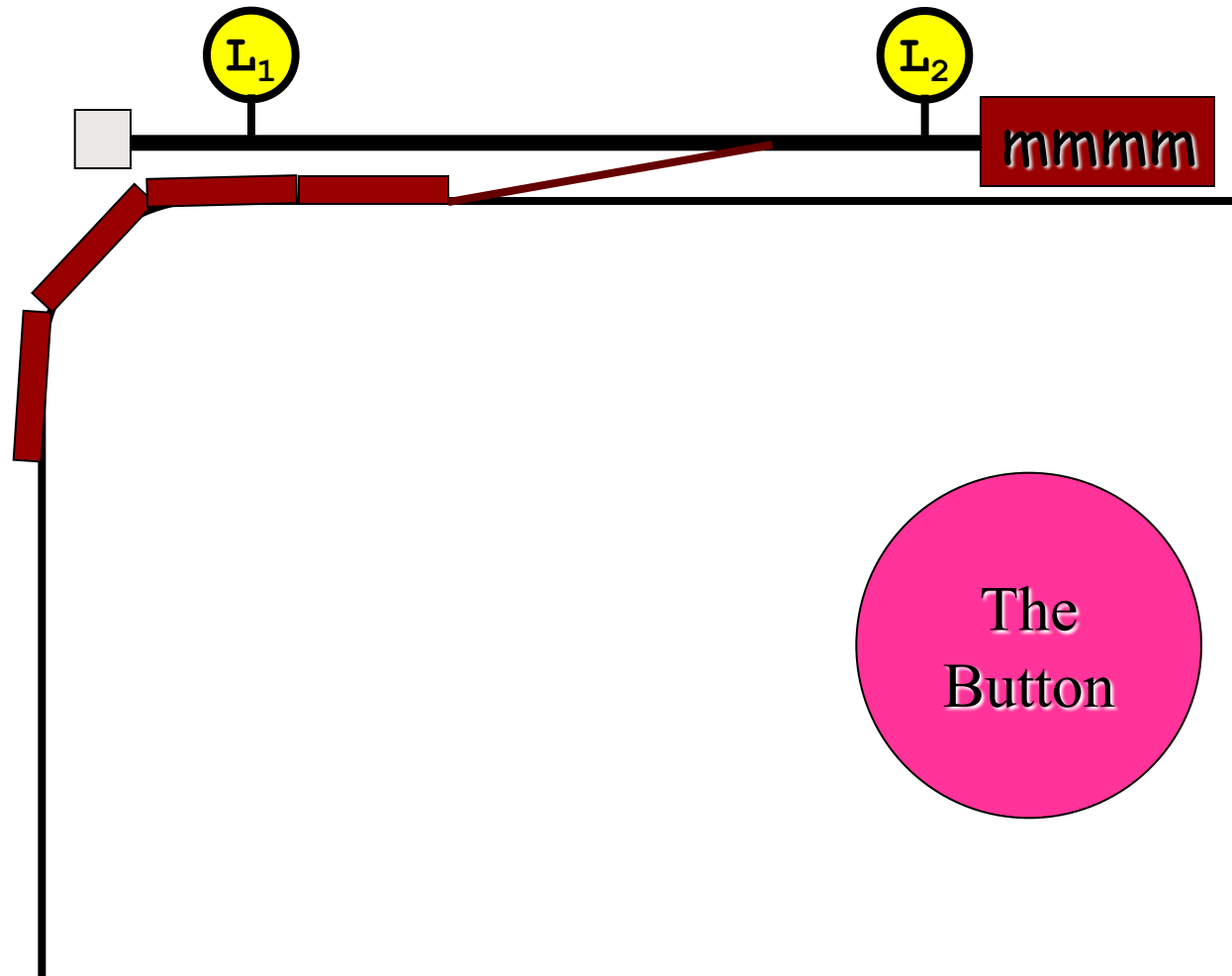
My Garage Door



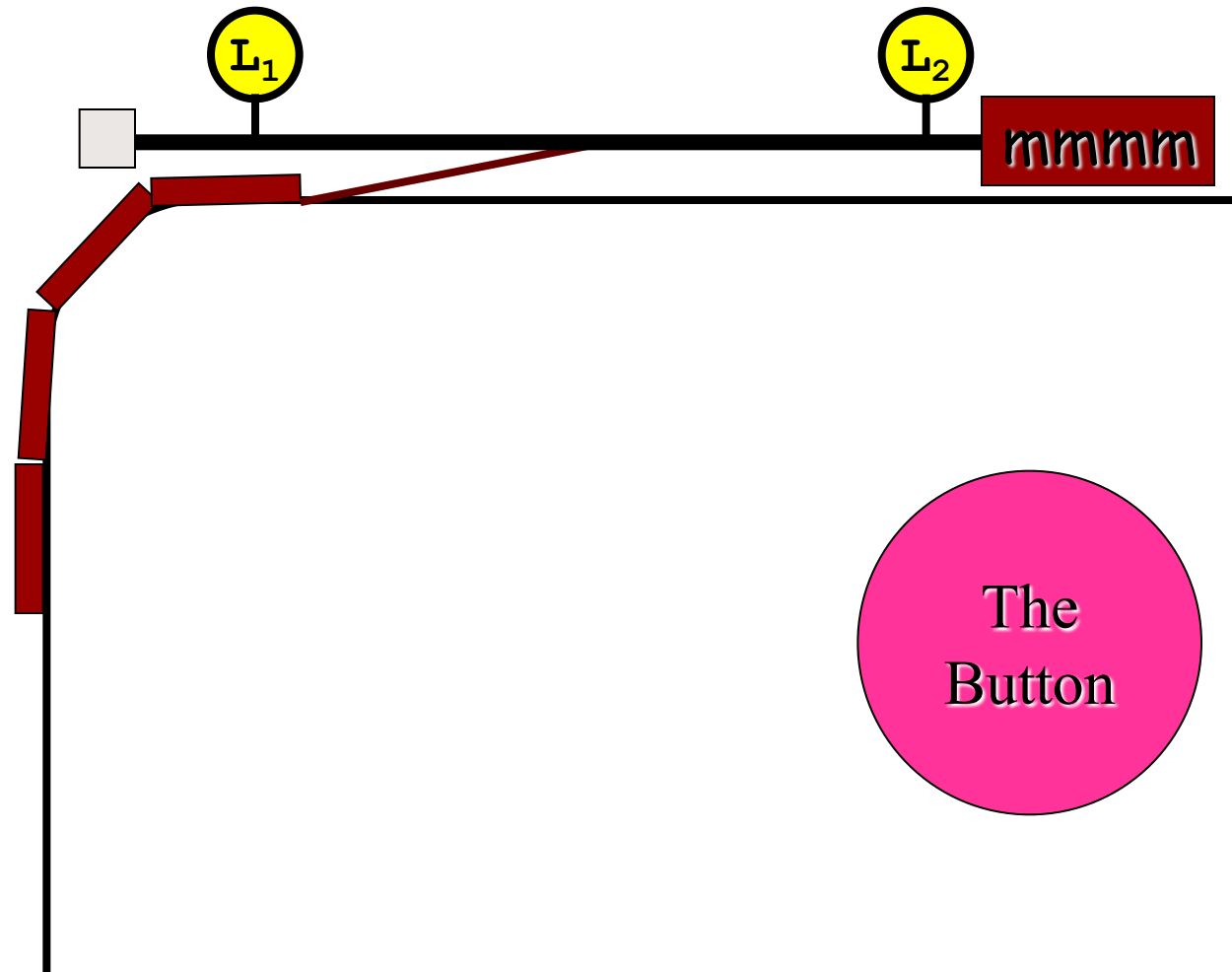
My Garage Door



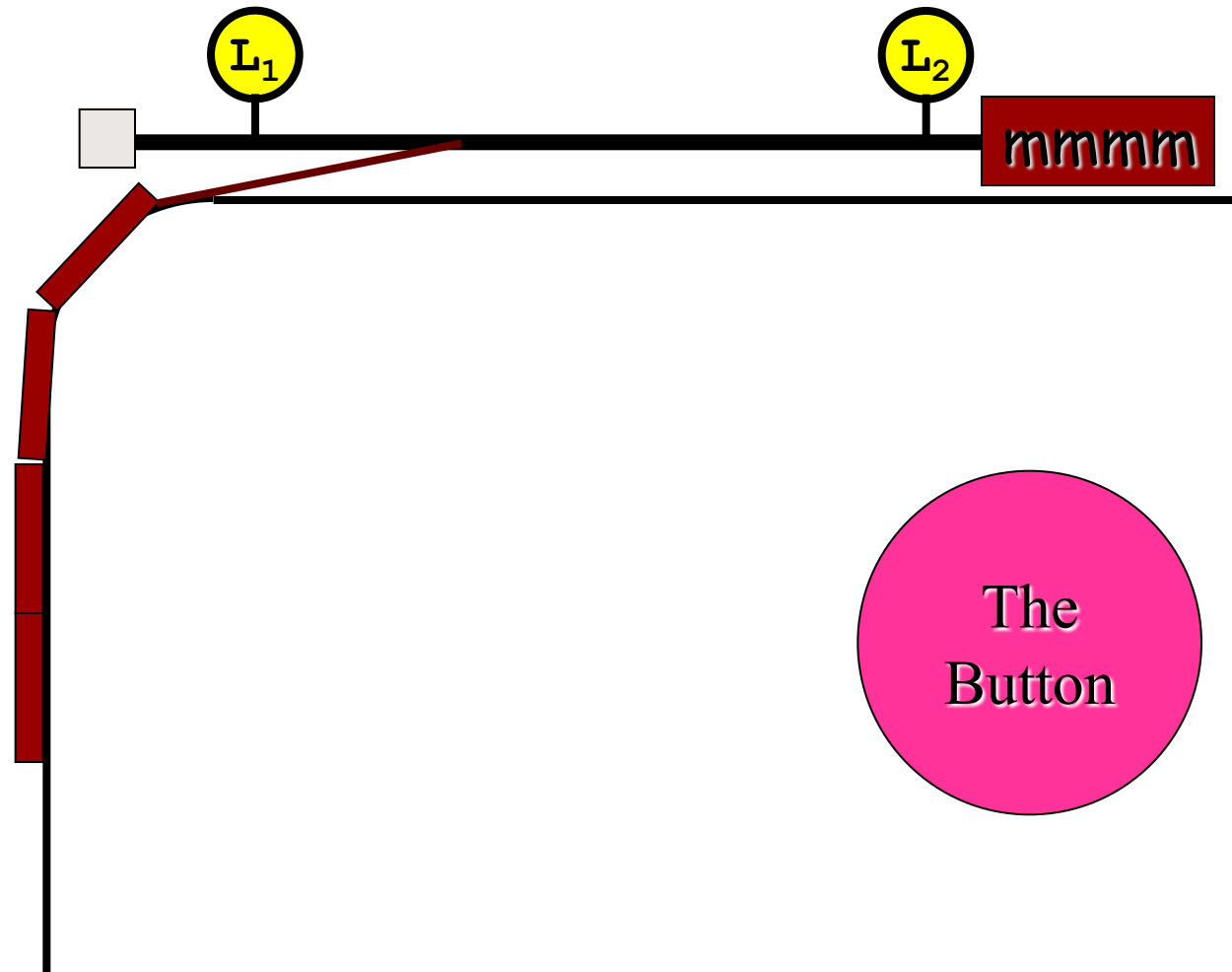
My Garage Door



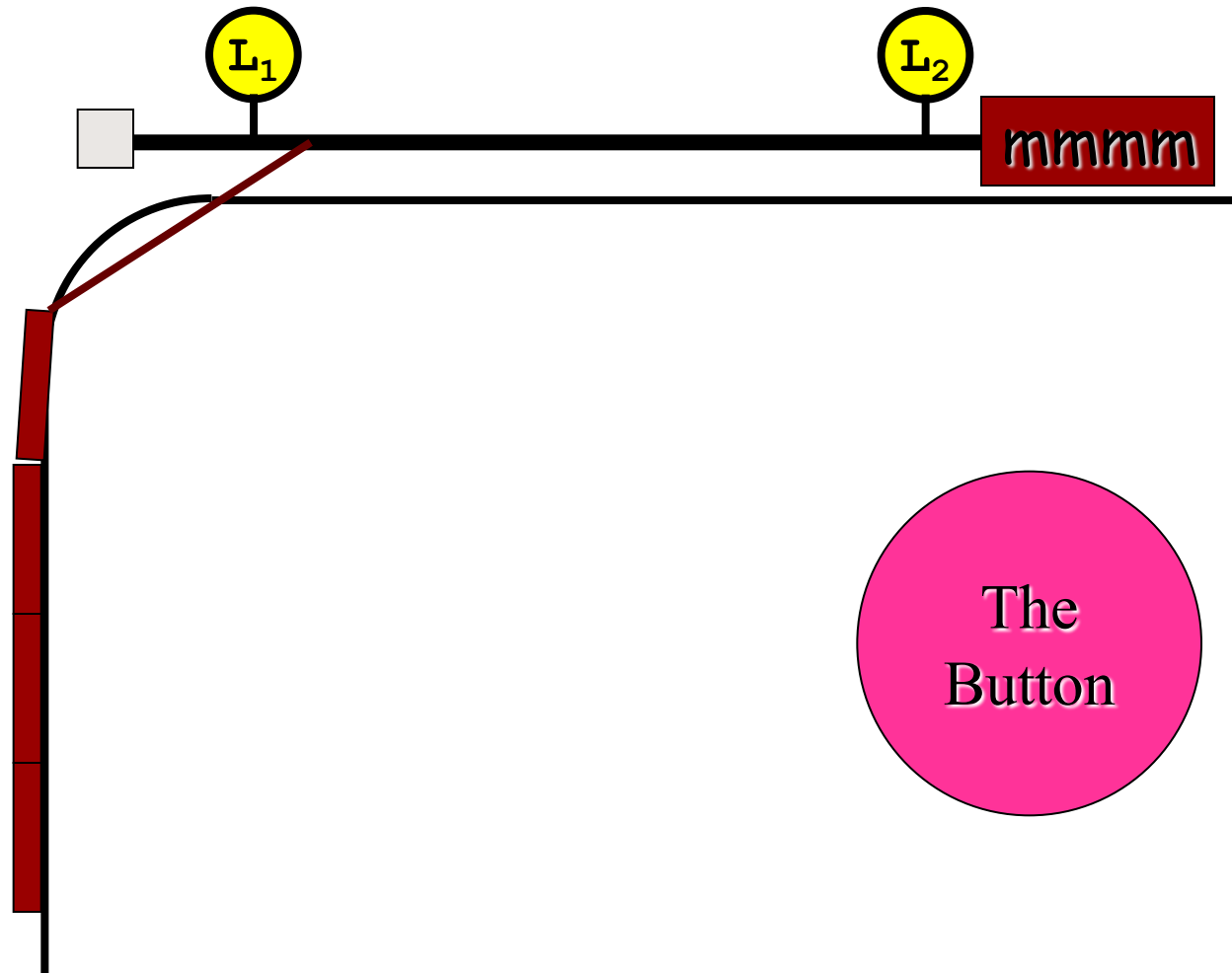
My Garage Door



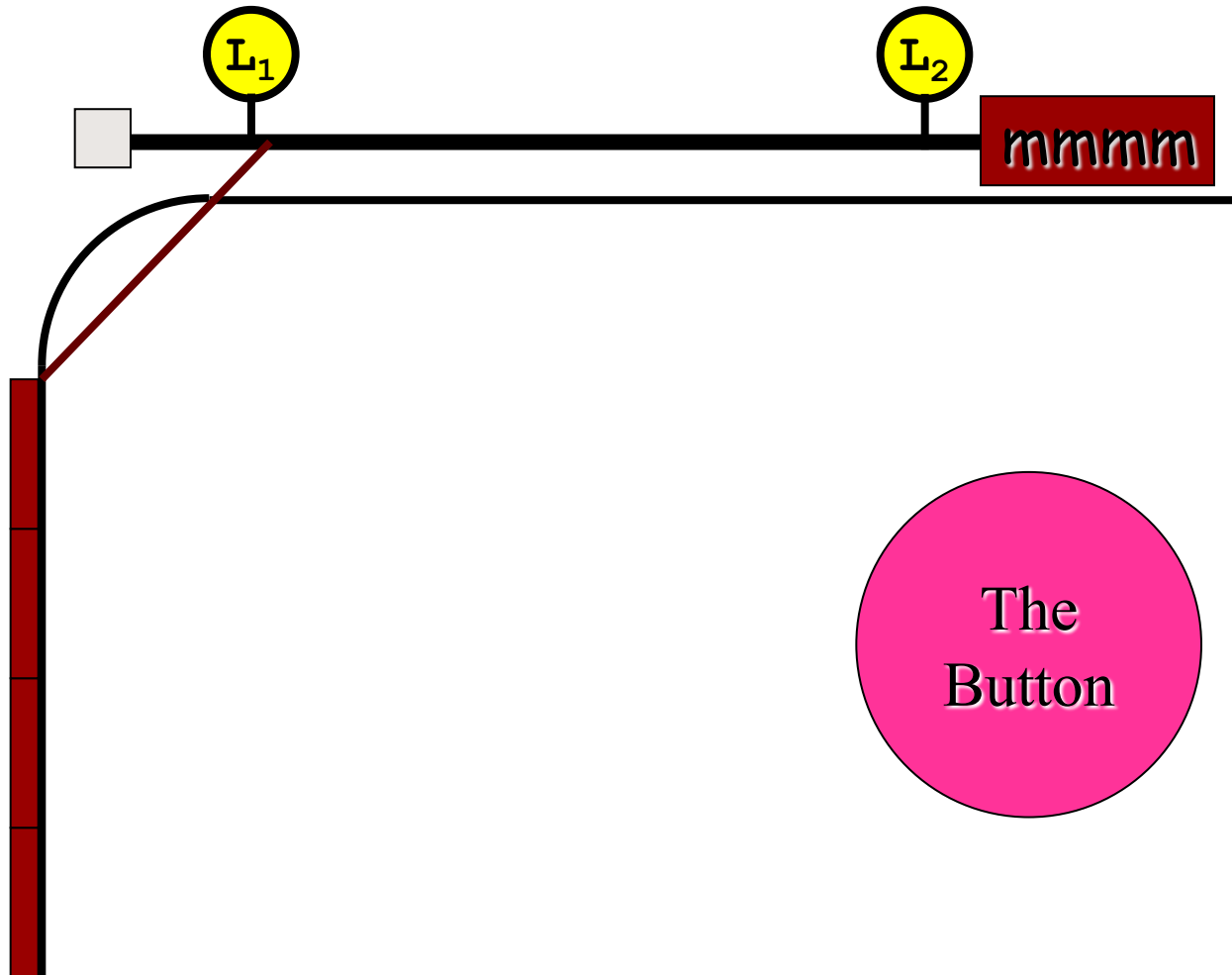
My Garage Door



My Garage Door

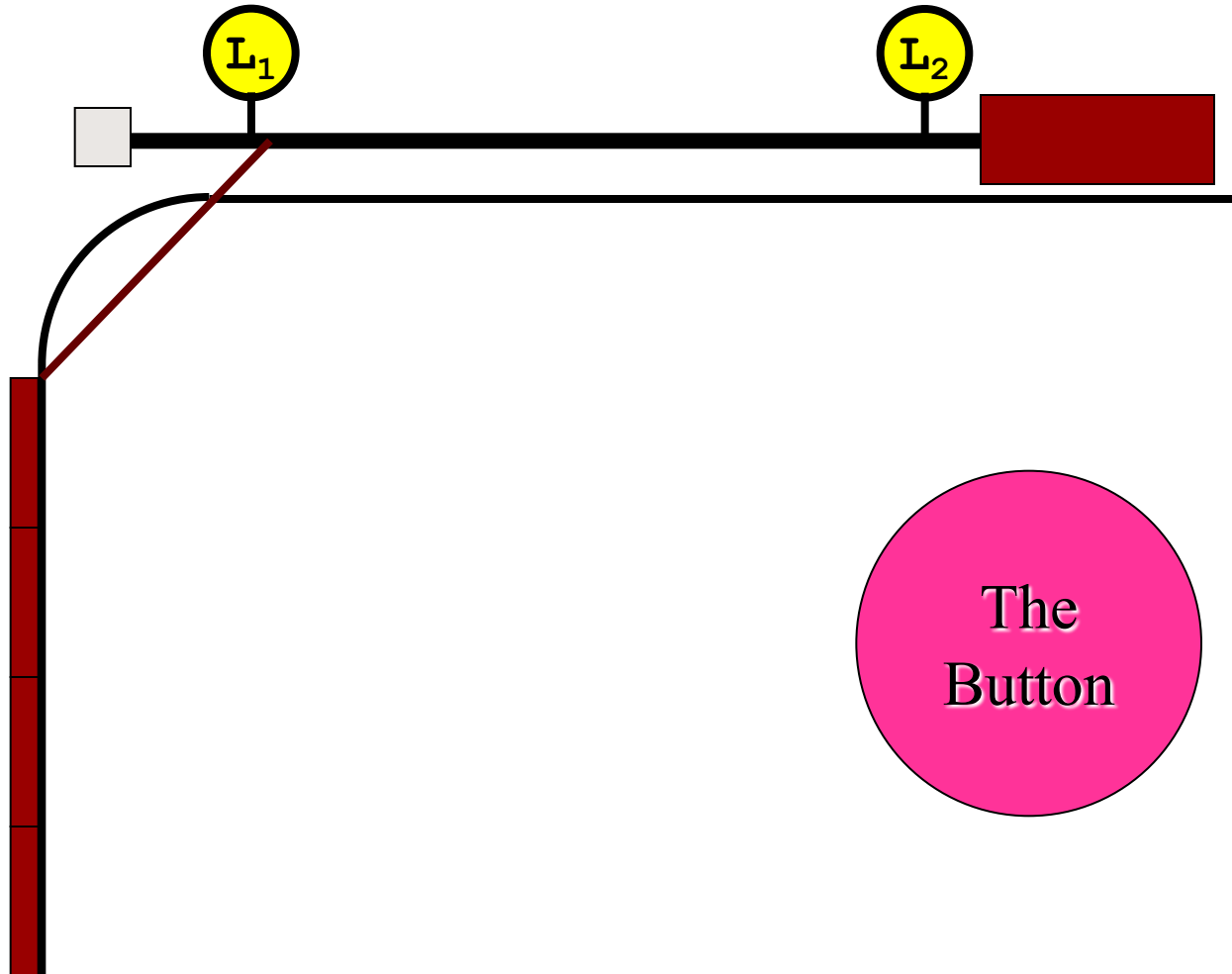


My Garage Door



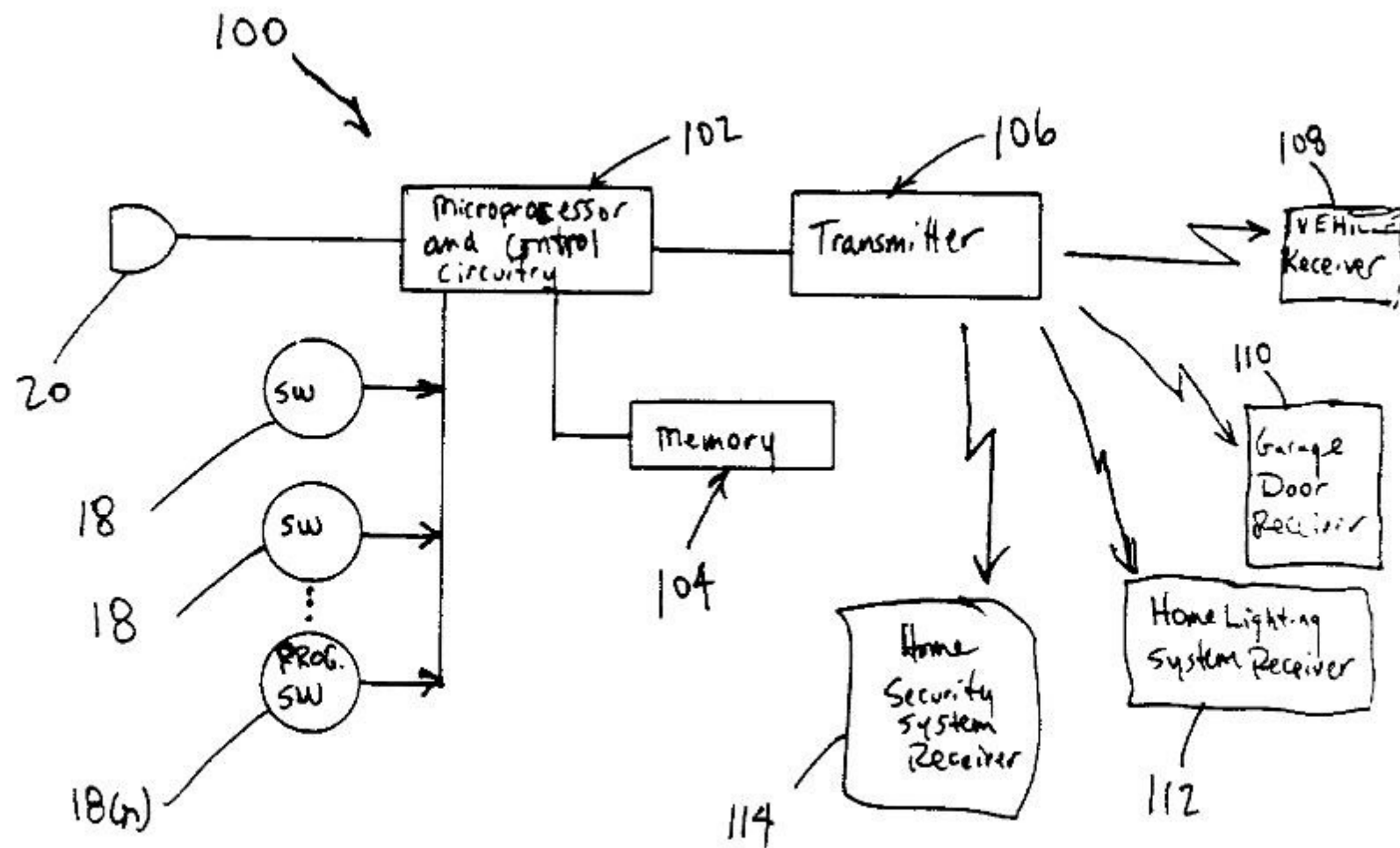
The
Button

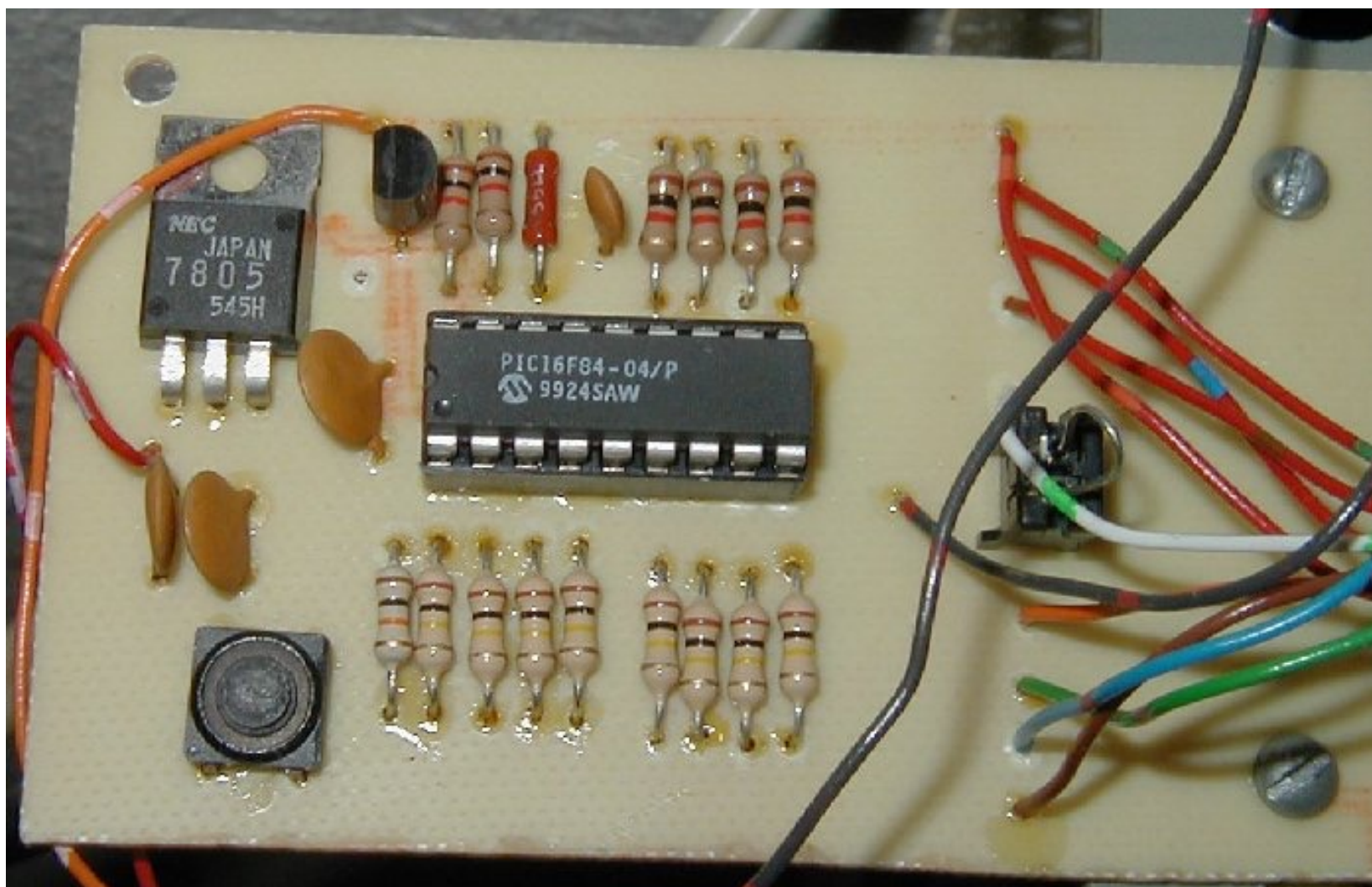
My Garage Door

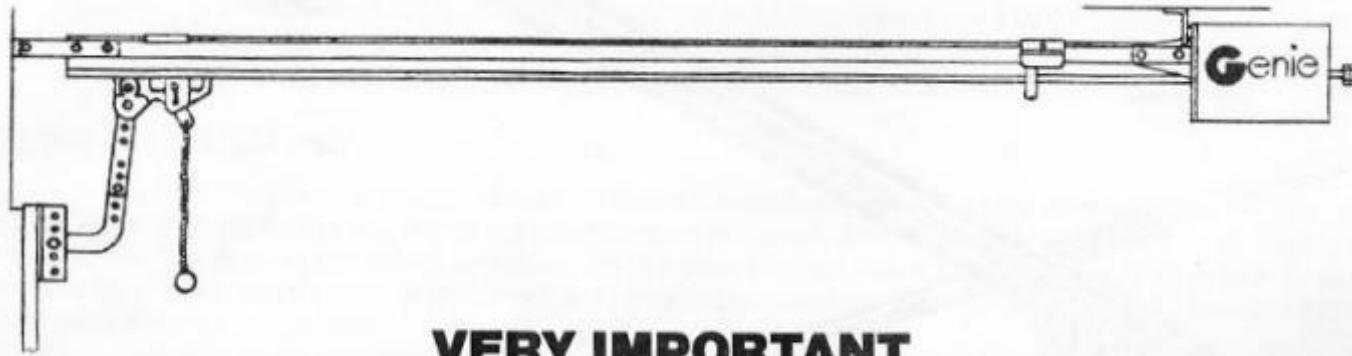


How can we describe the control logic?



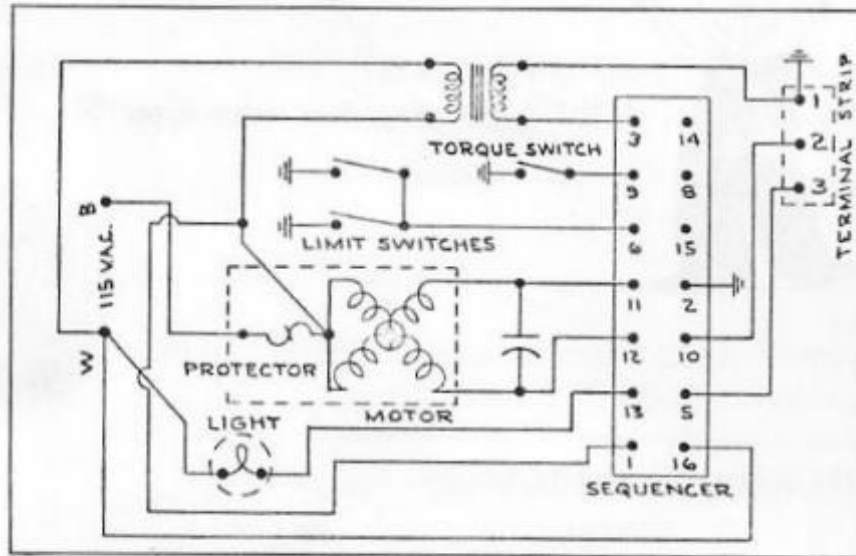




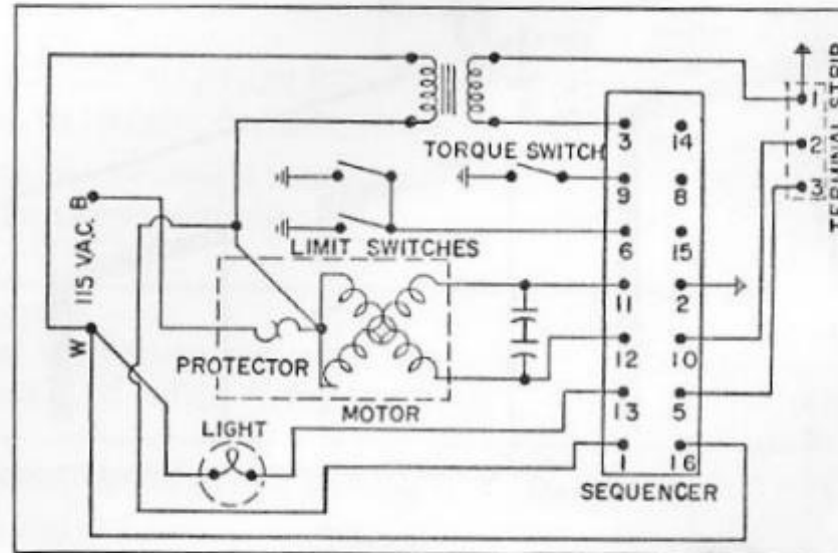


VERY IMPORTANT

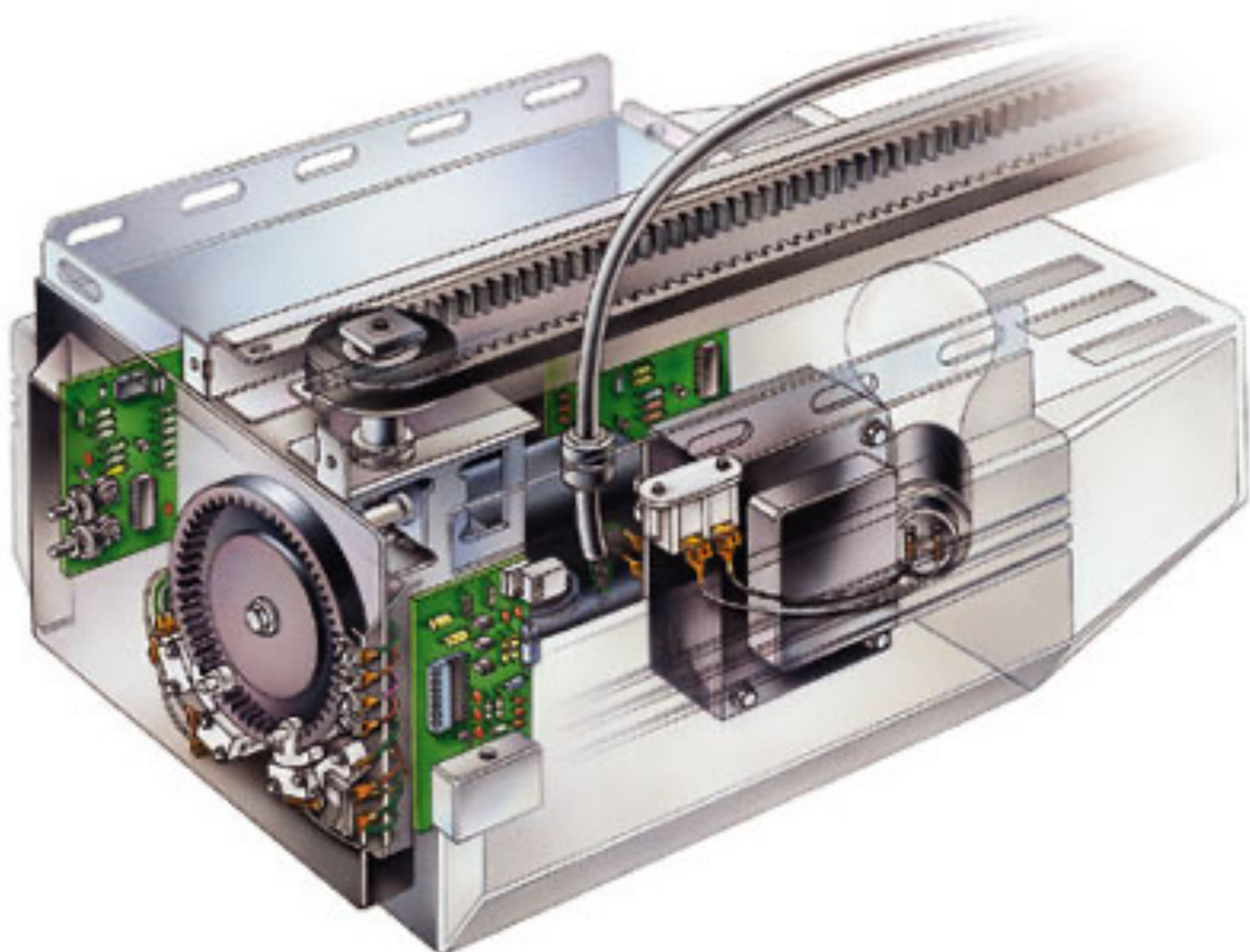
Handles, ropes and any projection on your door not necessary to its operation should be removed when installing a door opener. Failure to remove these items may cause accidental contact or engagement of these projections when door opener is operated. **This is unsafe!** Fixed mechanical stops should be used instead of rope stops on door hardware.



Model 404 wiring diagram



Model 414 wiring diagram



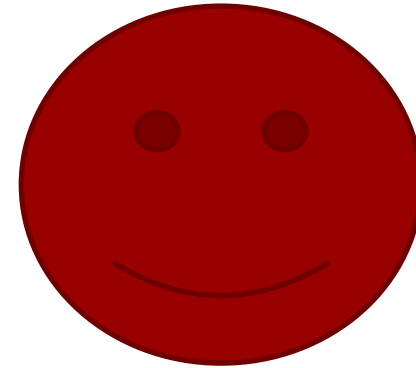
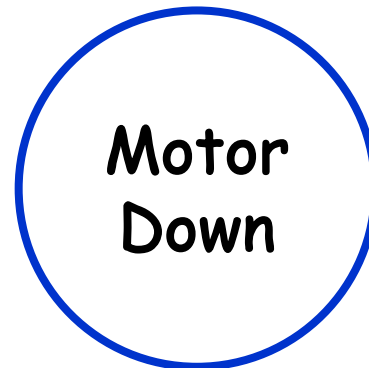
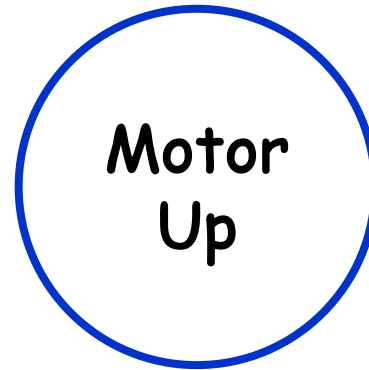
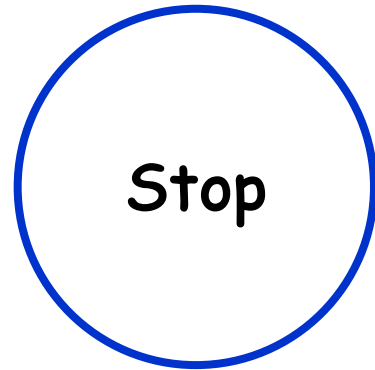
➤ A Finite State Machine
(FSM) Diagram!

So How Will We Do This?

- Decide how we will encode our states
 - A binary number will always work
- Draw a state diagram including
 - Labeled states (encoding)
 - Outputs during the state
 - Arcs with conditions for state changes
- Create truth tables with
 - Inputs: **Circuit inputs** and **Current state**
 - Output: **Circuit outputs** and **Next state**
 - Fill in the outputs for all combinations based on the state diagram
- Implement a combinational circuit for each output using the truth table

Start →

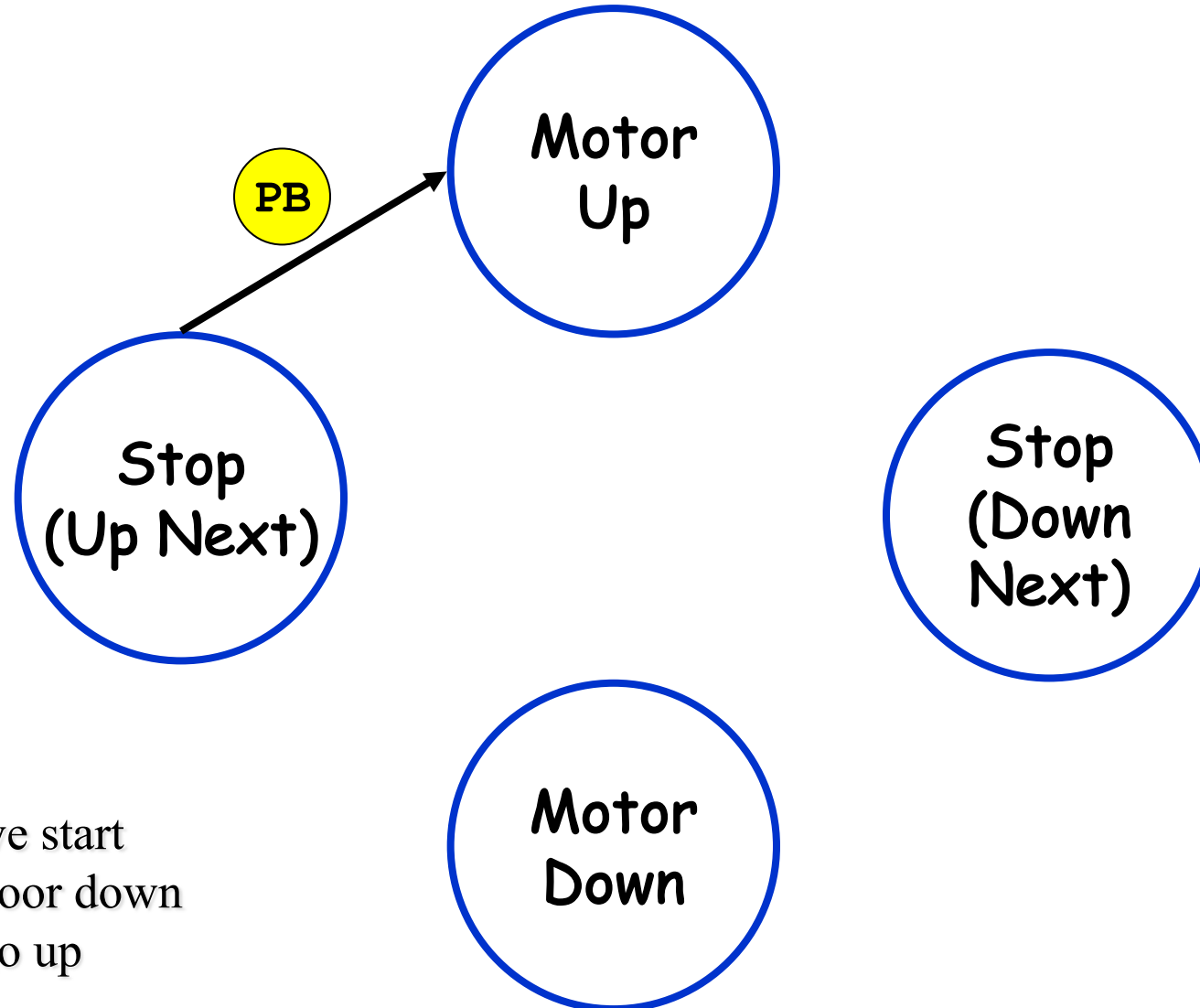
State Machine



Enough States?

Start

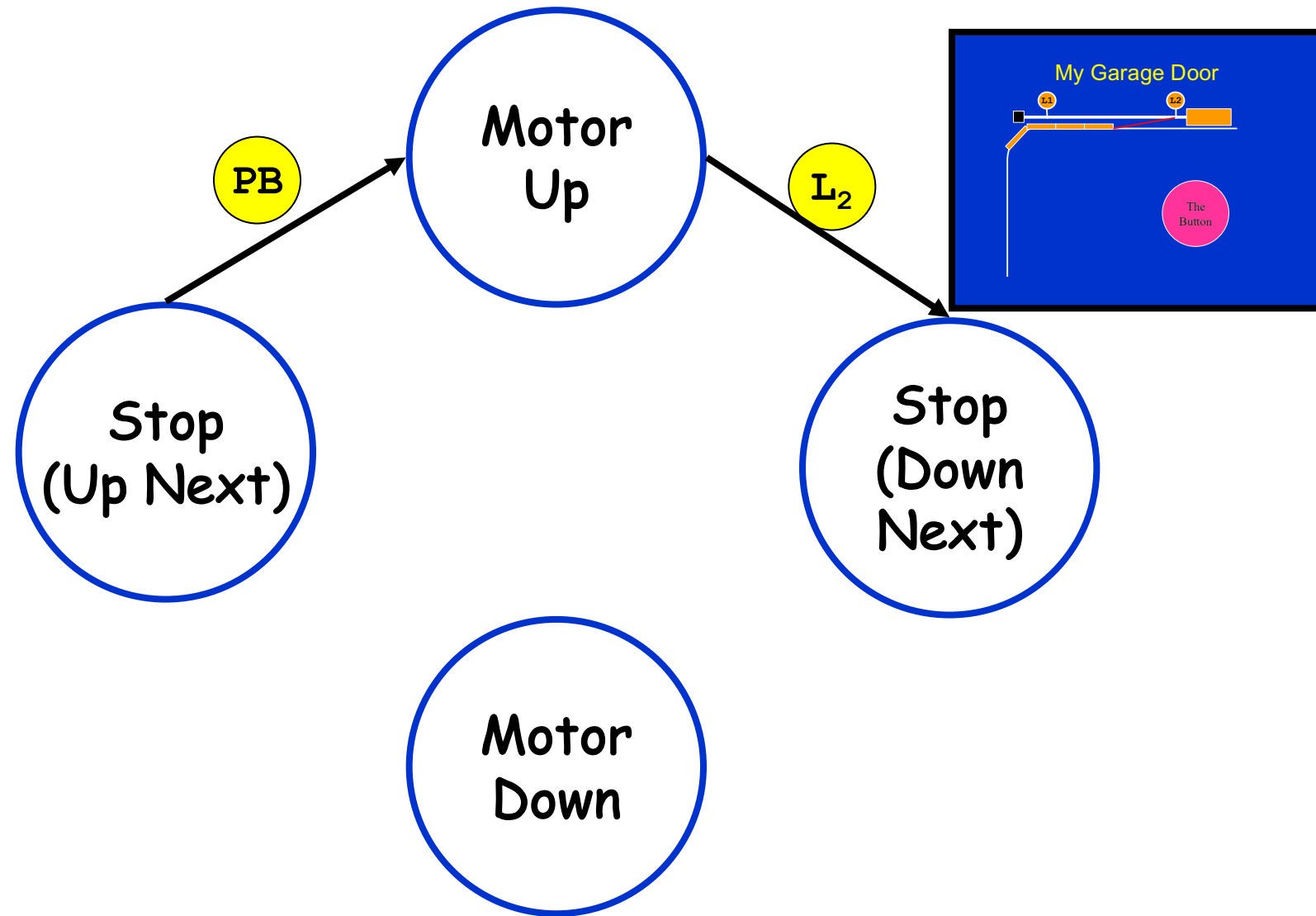
State Machine



Assume we start
with the door down
ready to go up

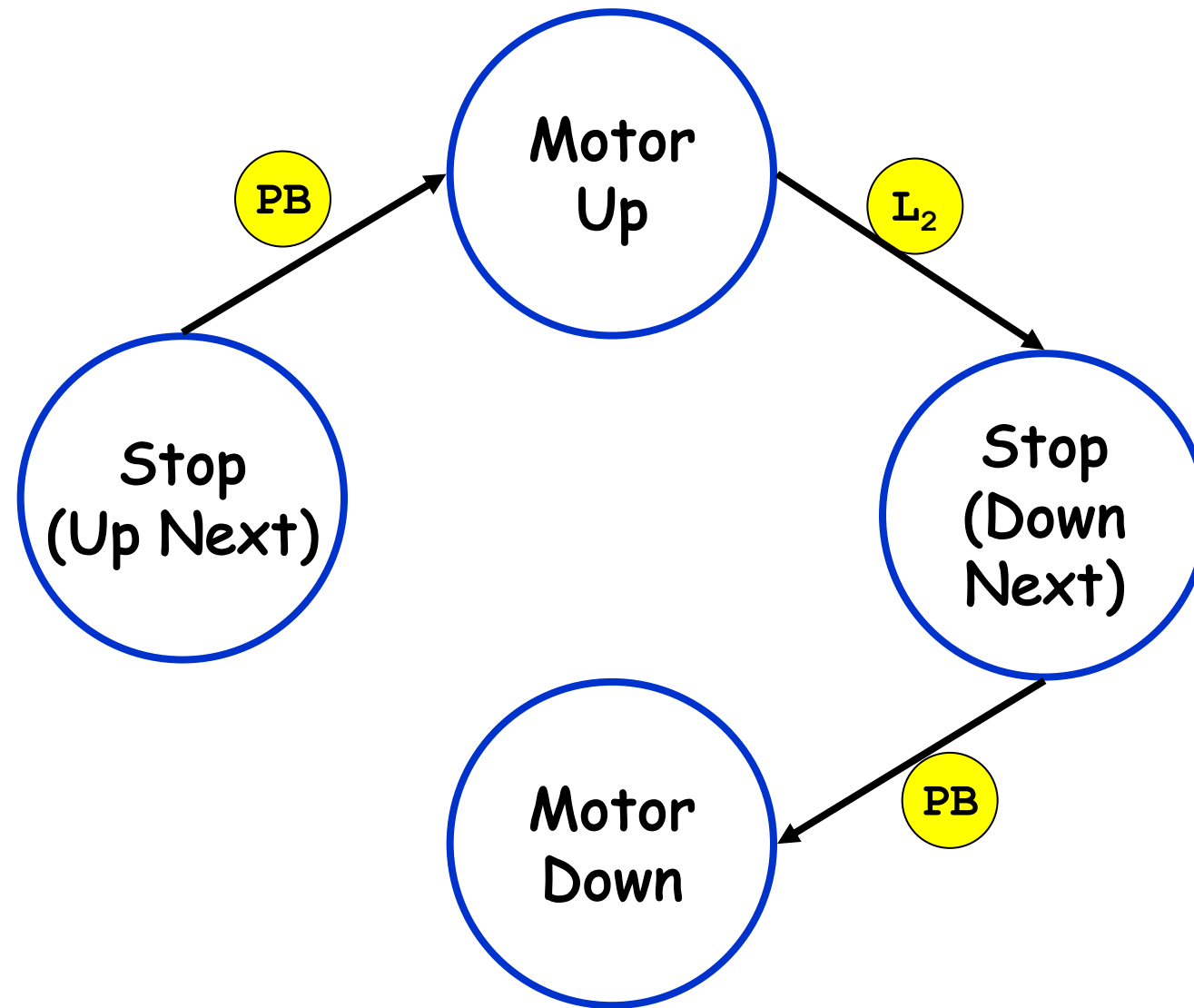
Start

State Machine



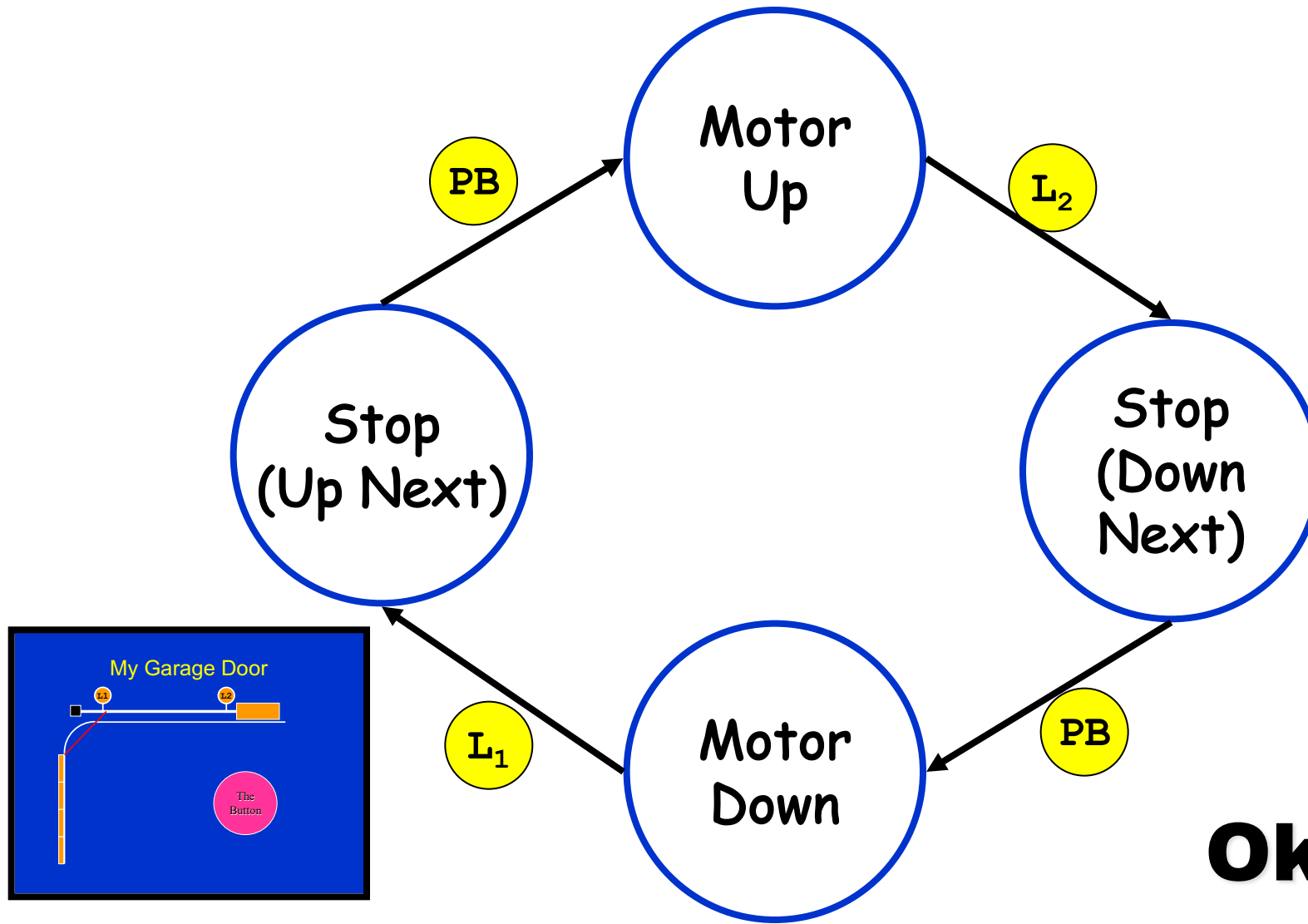
Start

State Machine



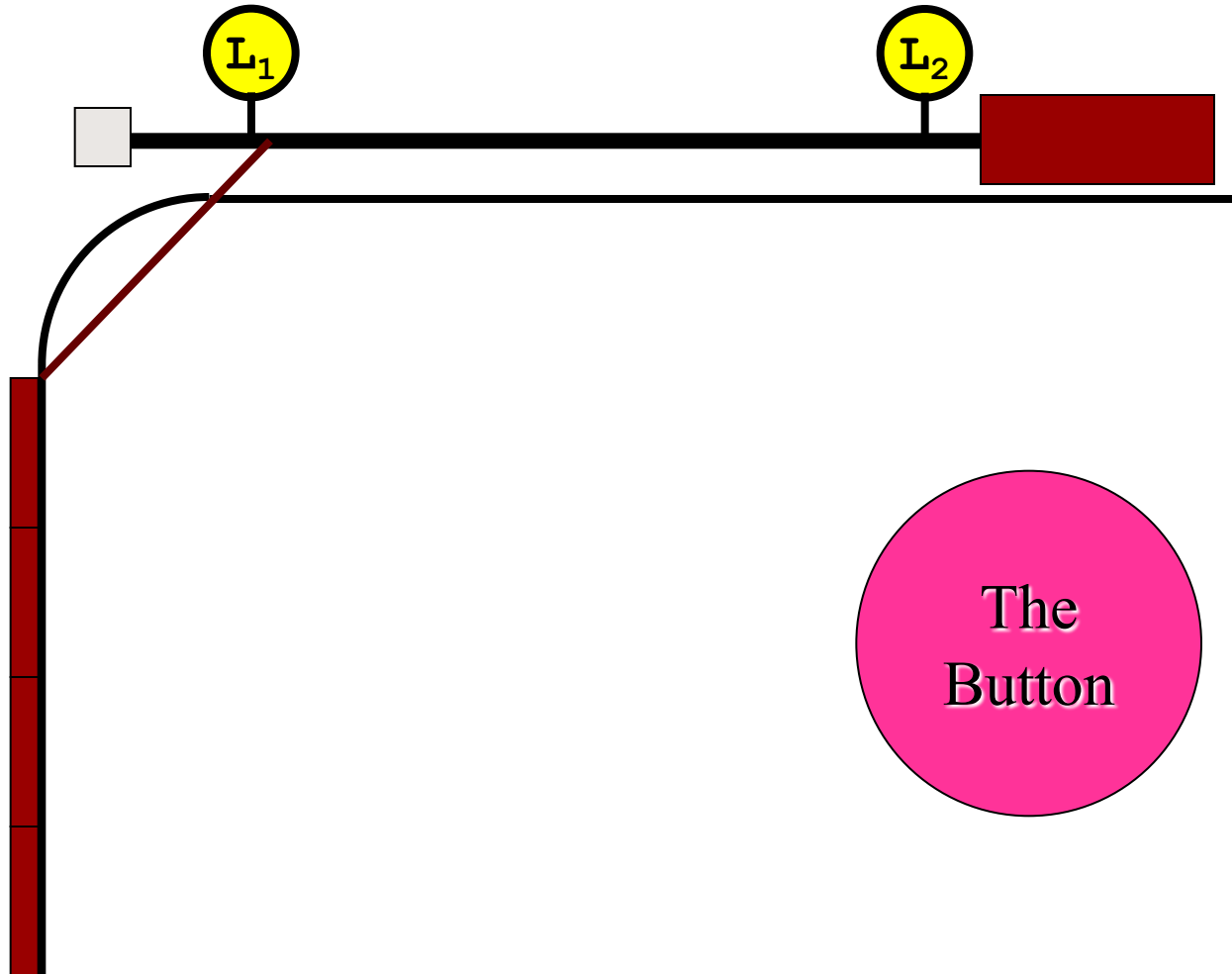
Start

State Machine

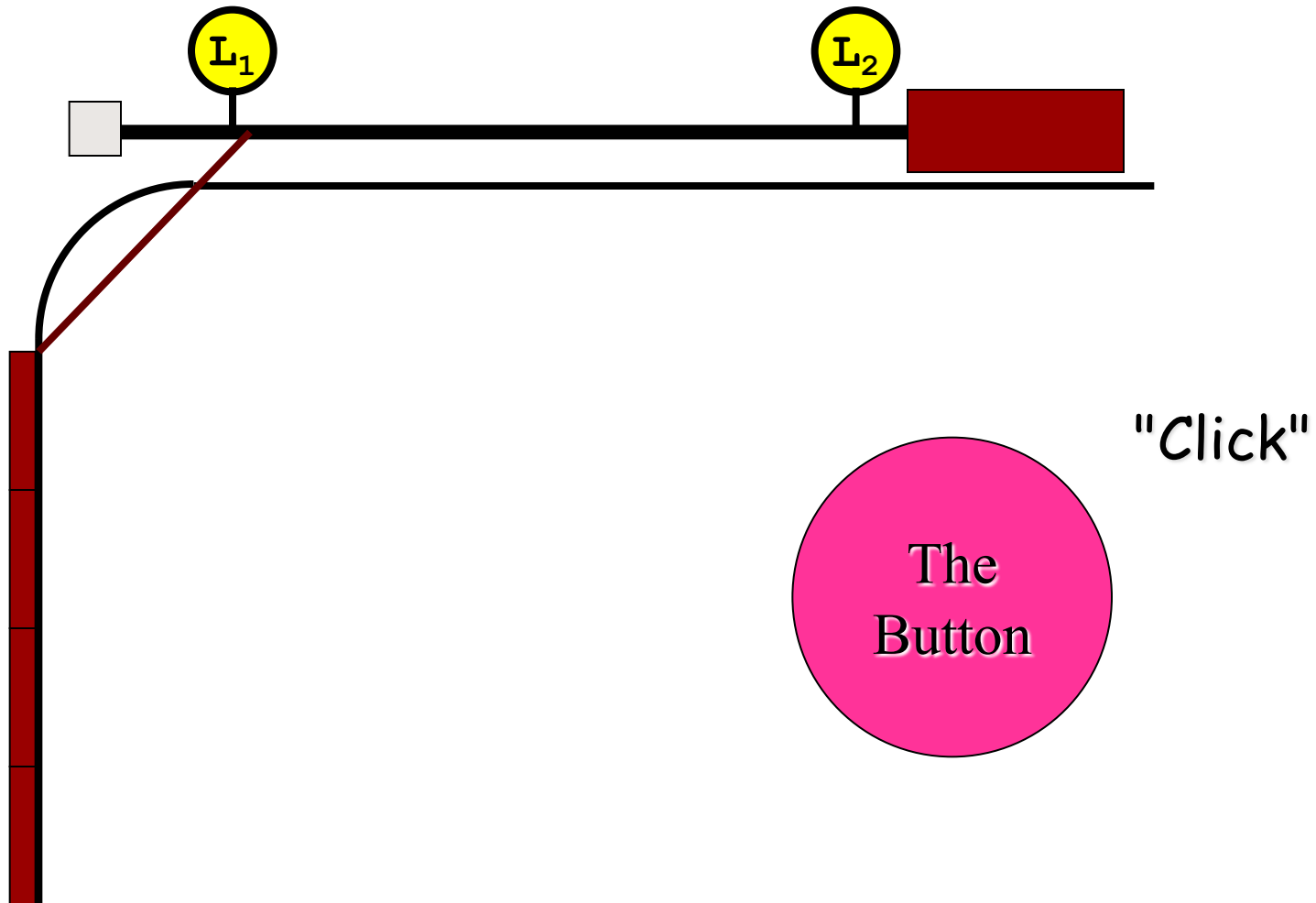


Okay?

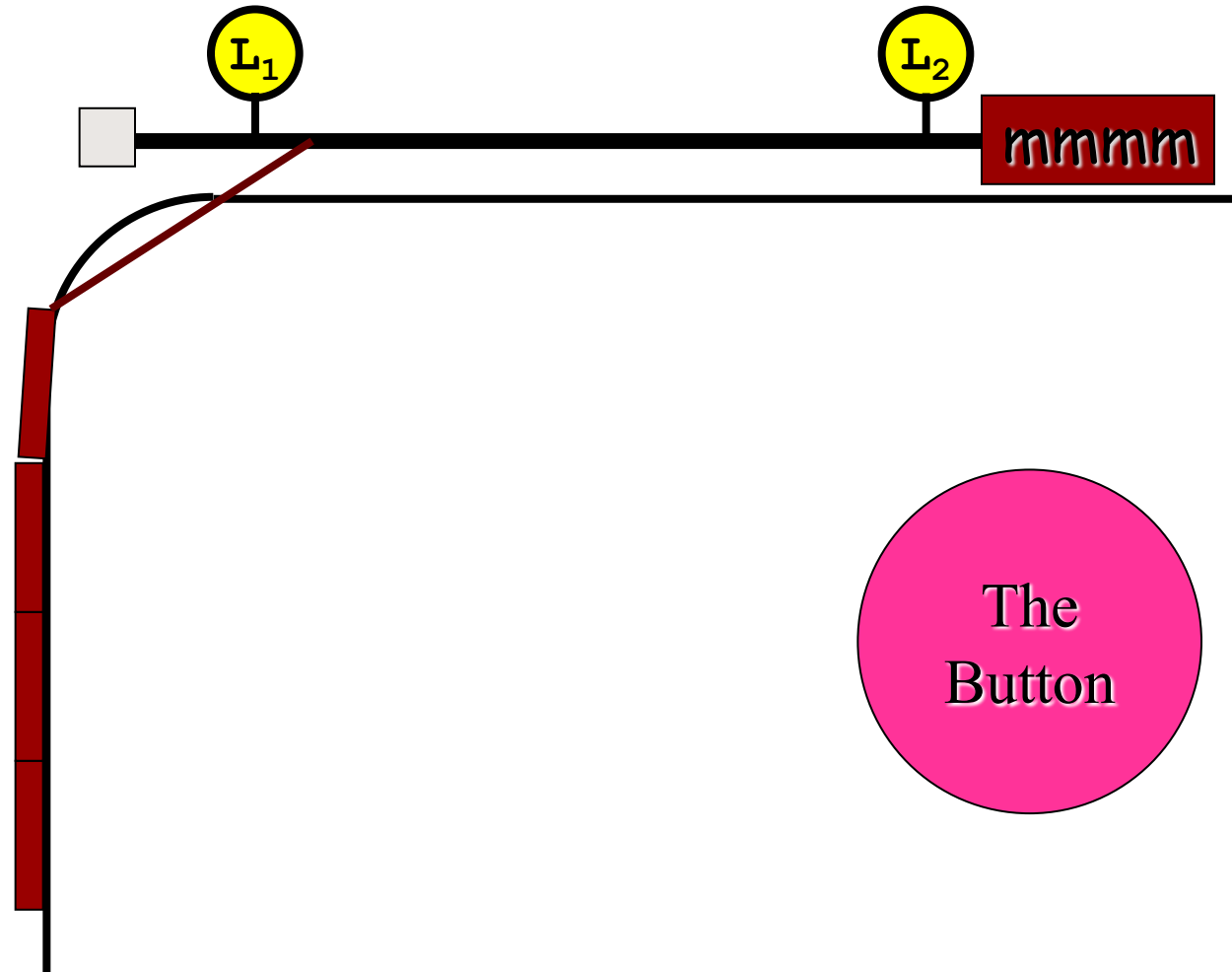
My Garage Door



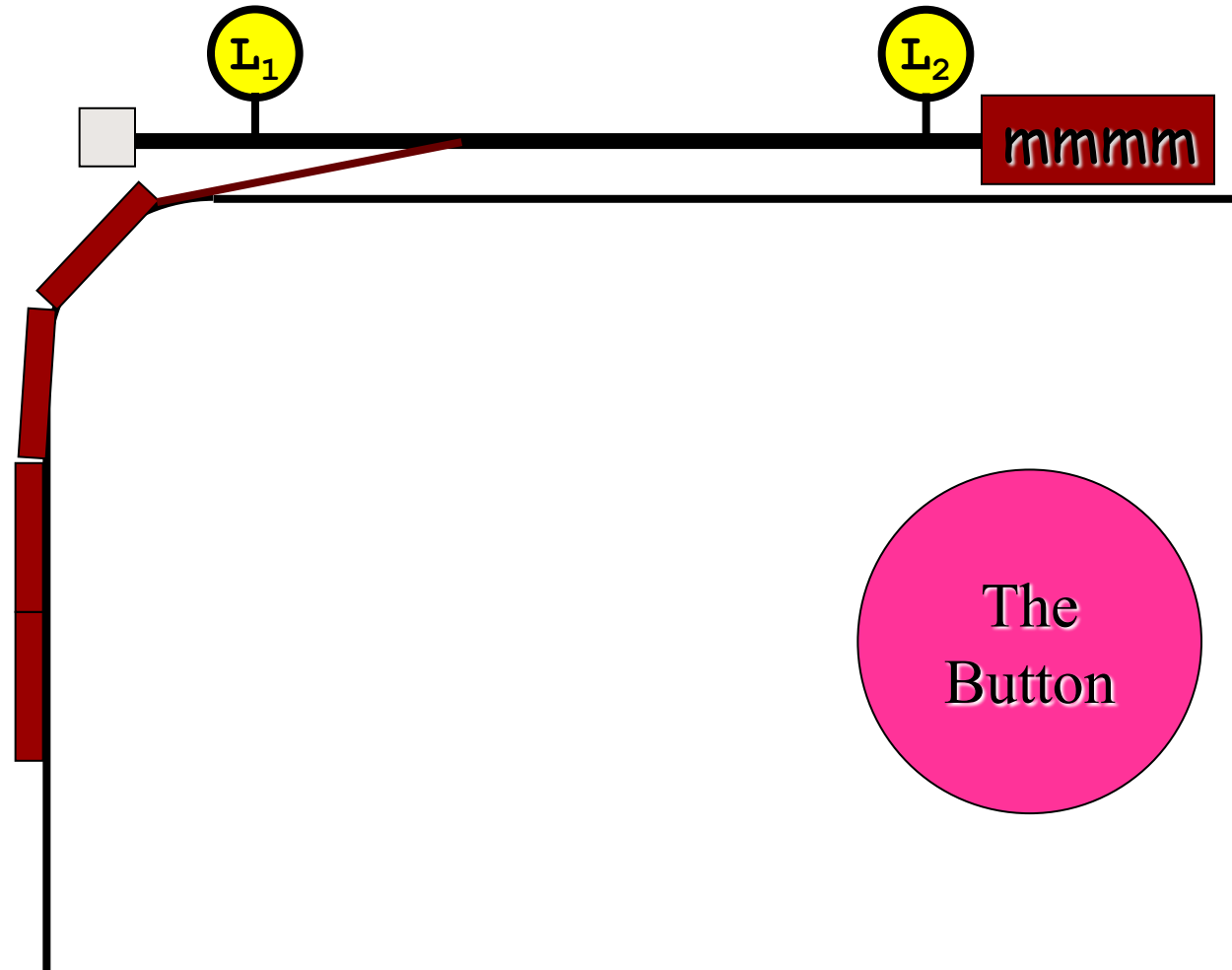
My Garage Door



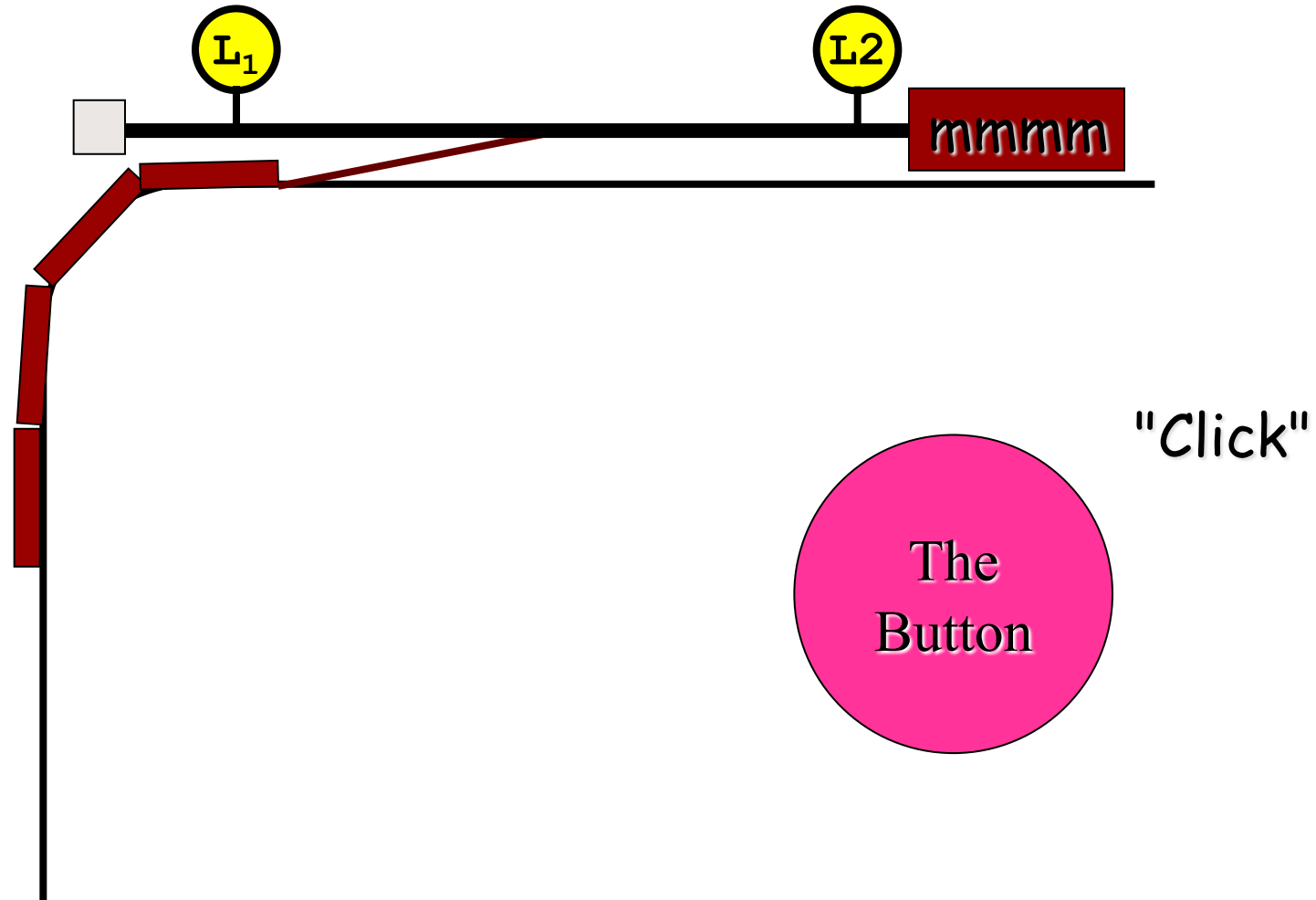
My Garage Door



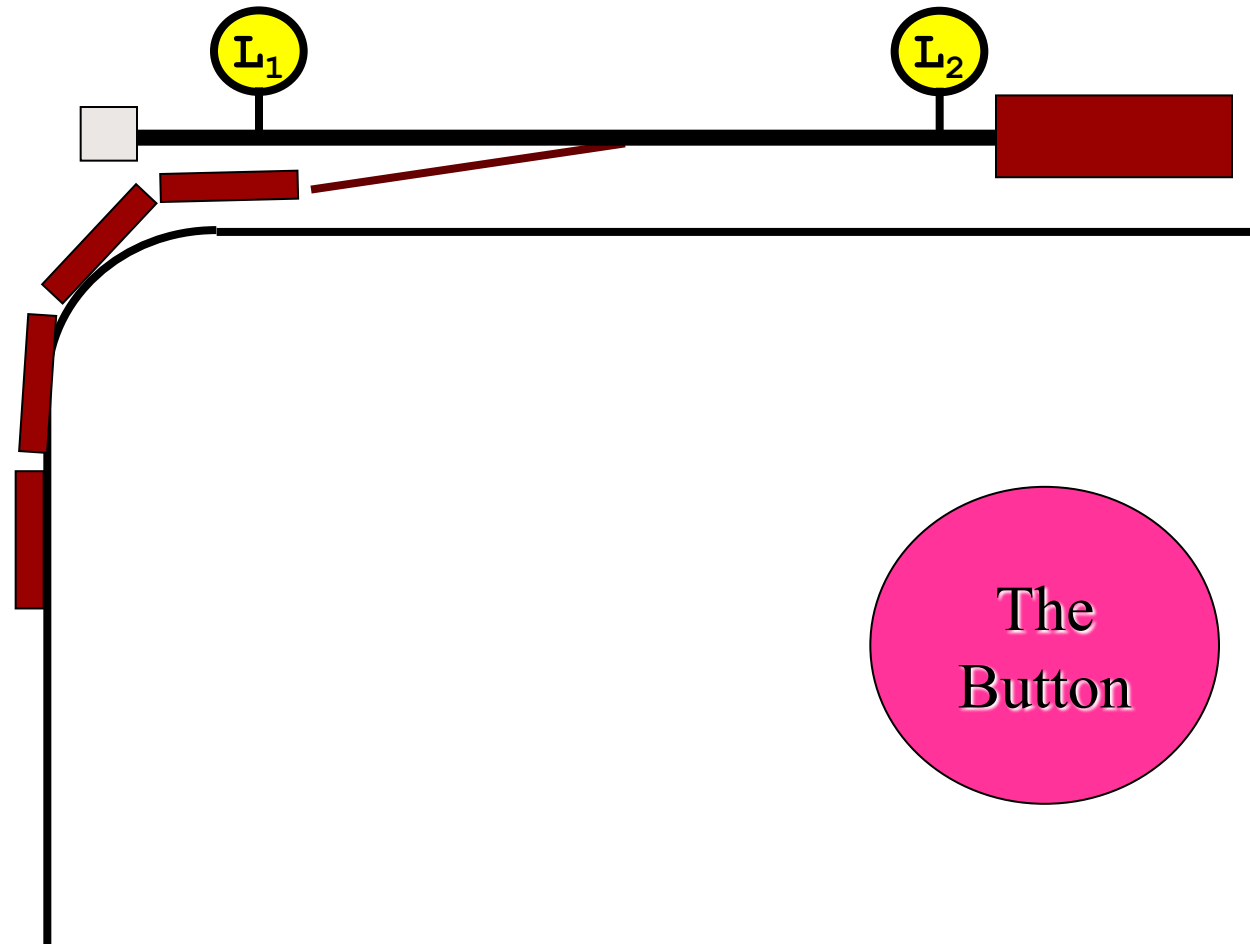
My Garage Door



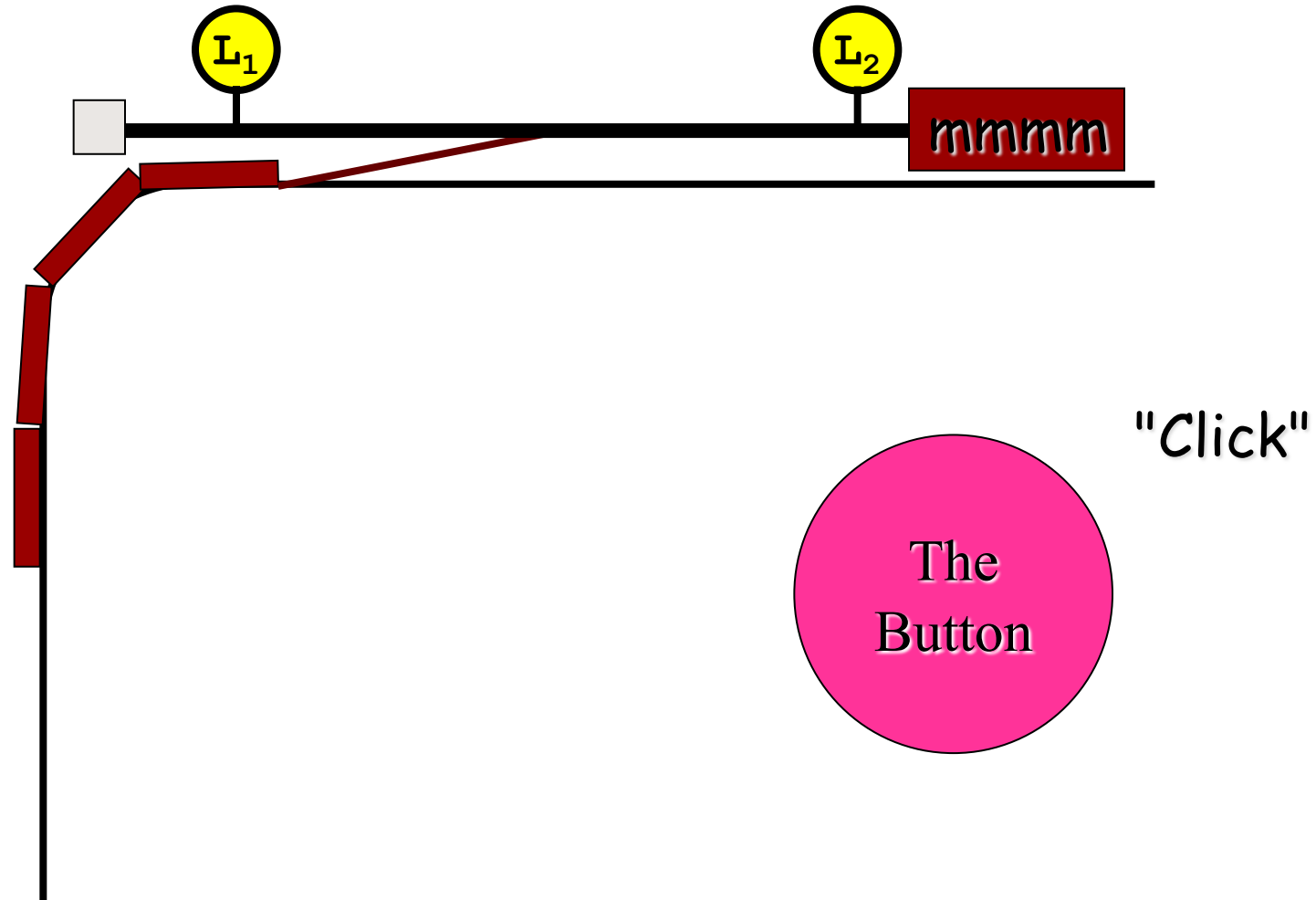
My Garage Door



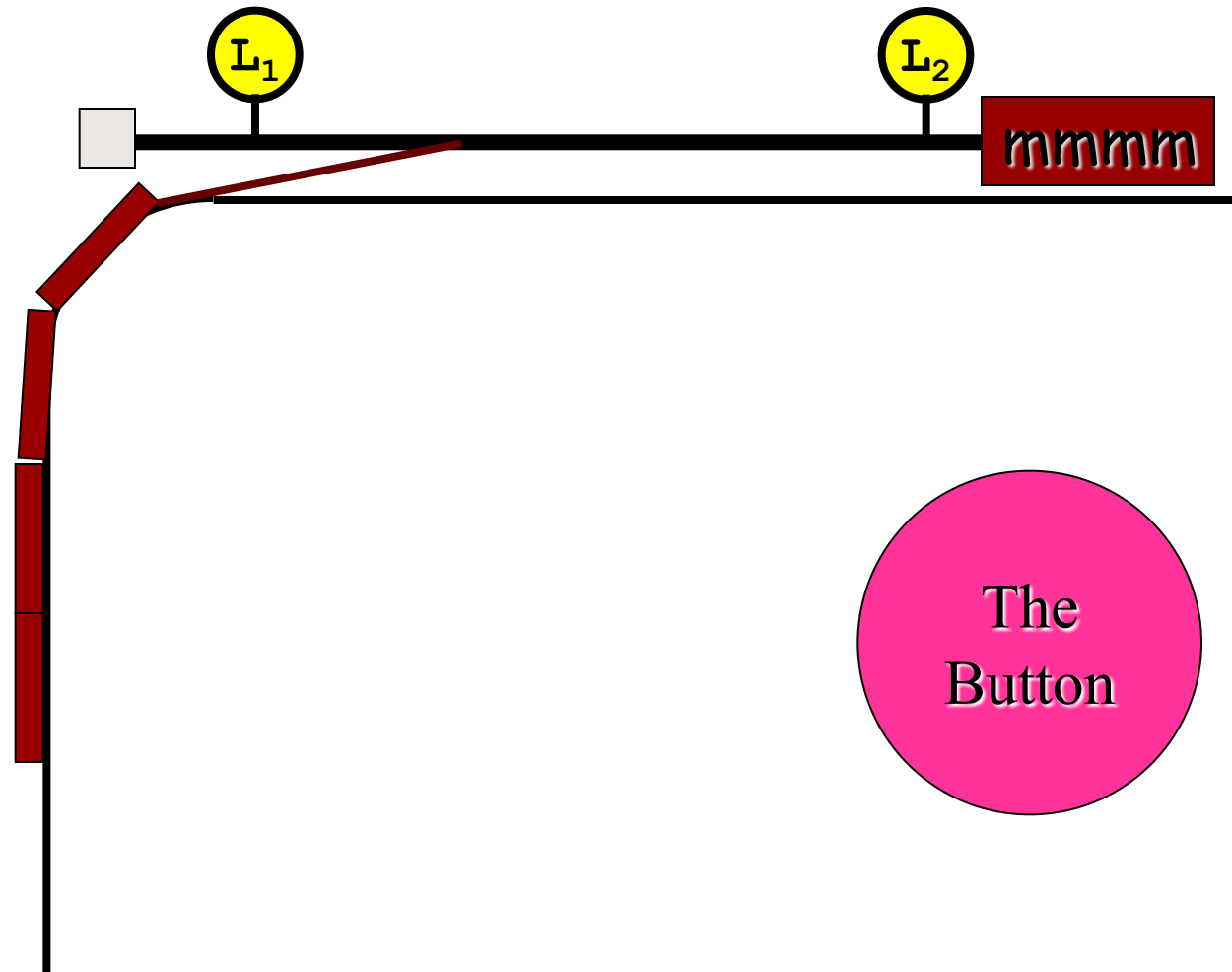
My Garage Door



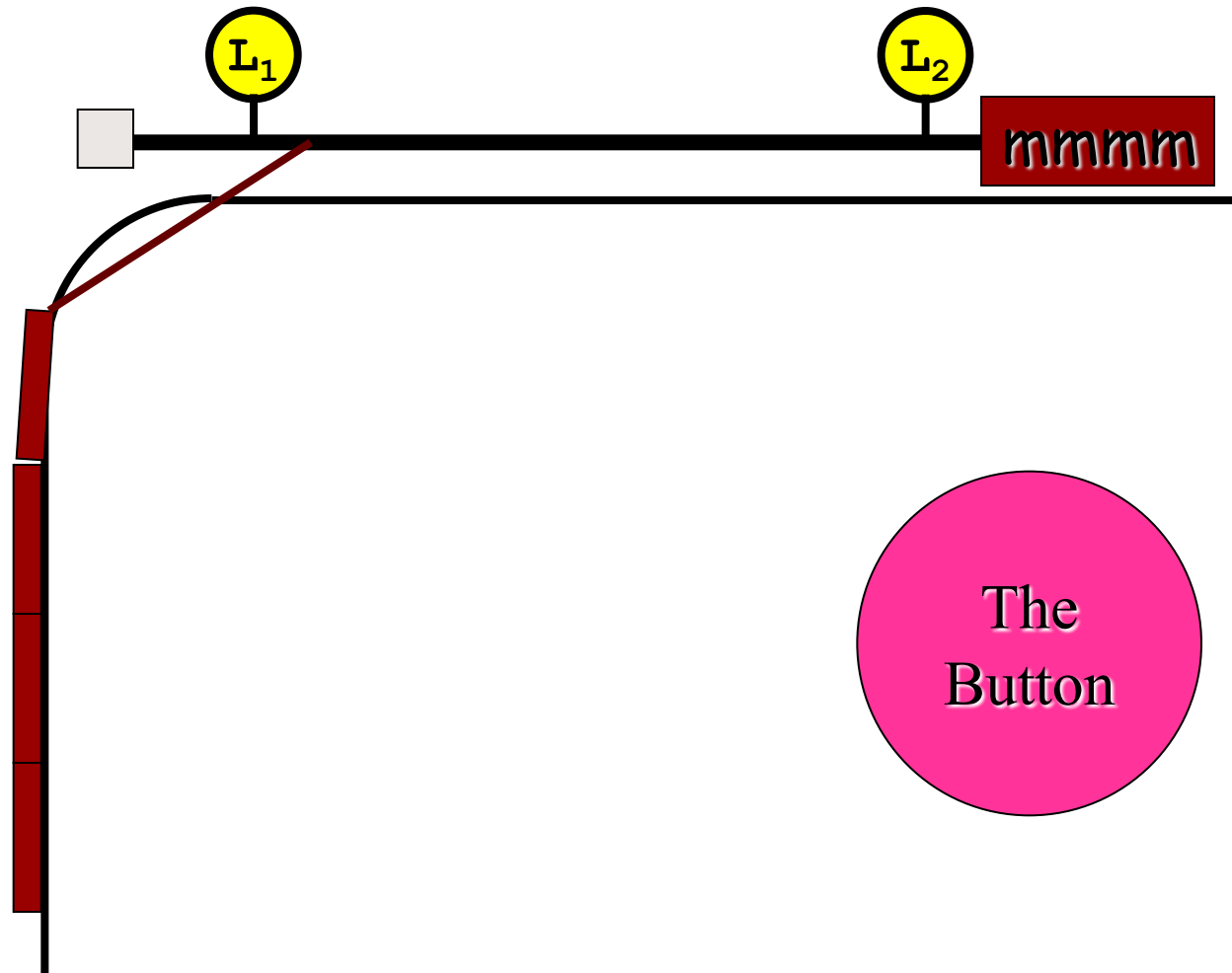
My Garage Door



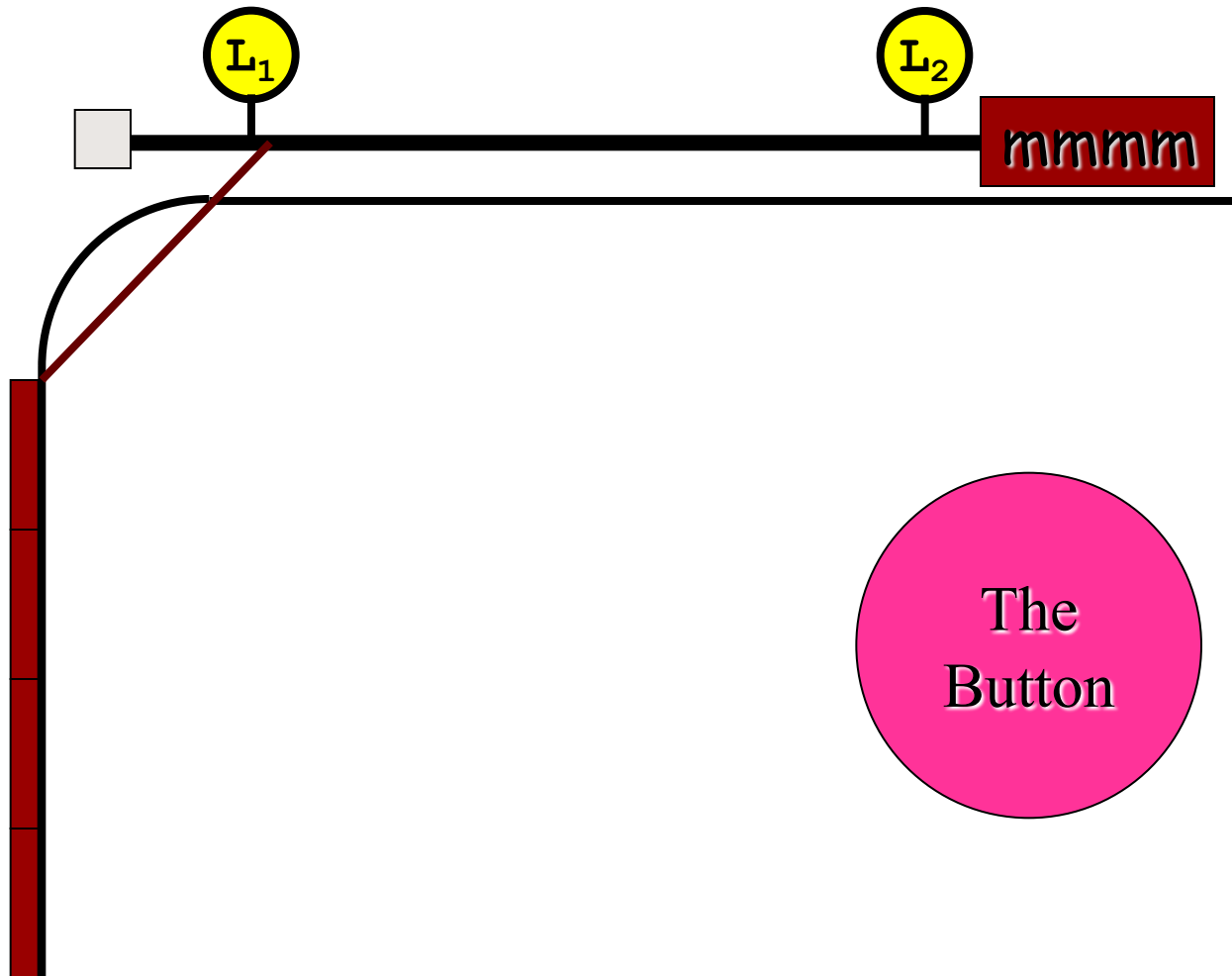
My Garage Door



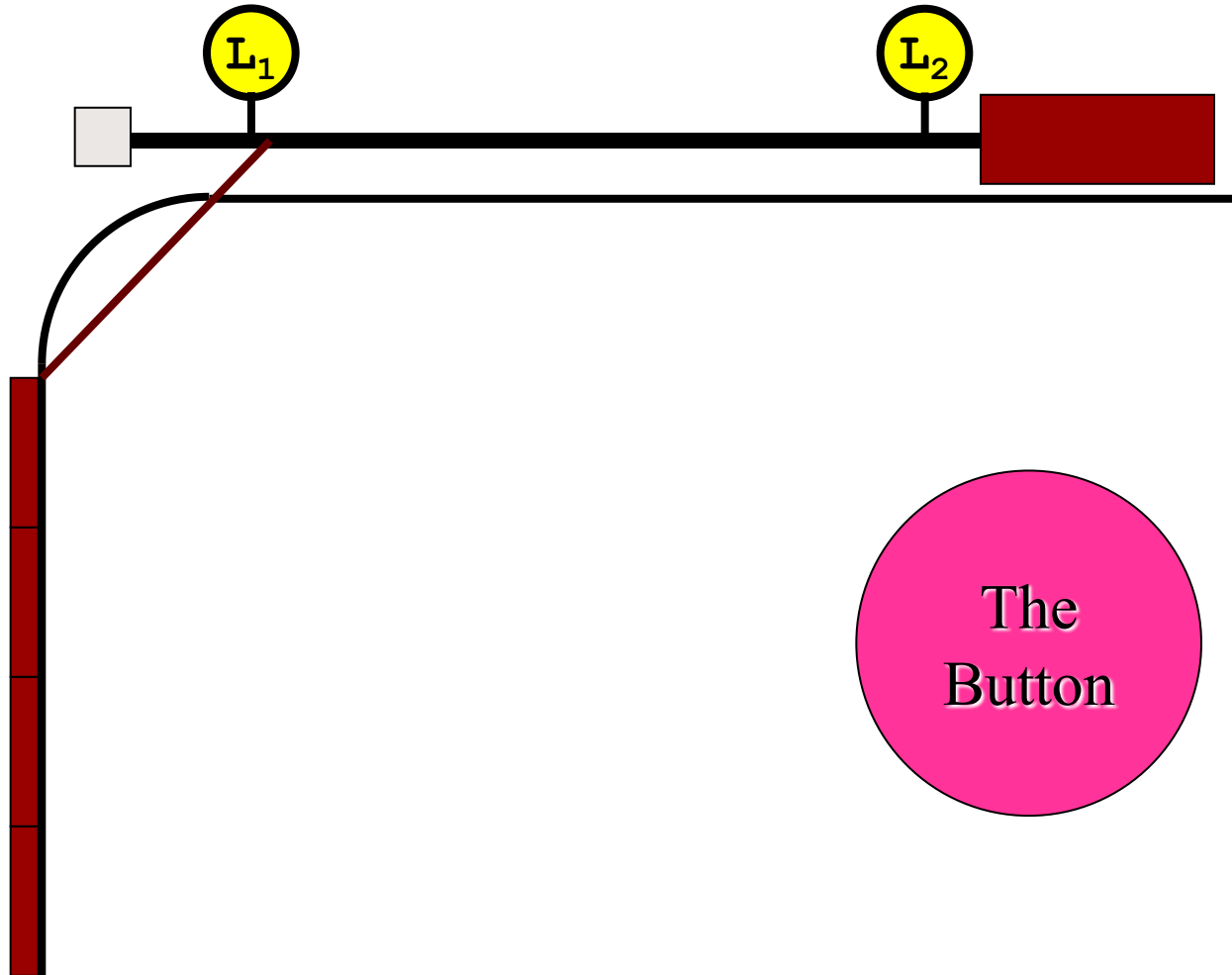
My Garage Door



My Garage Door

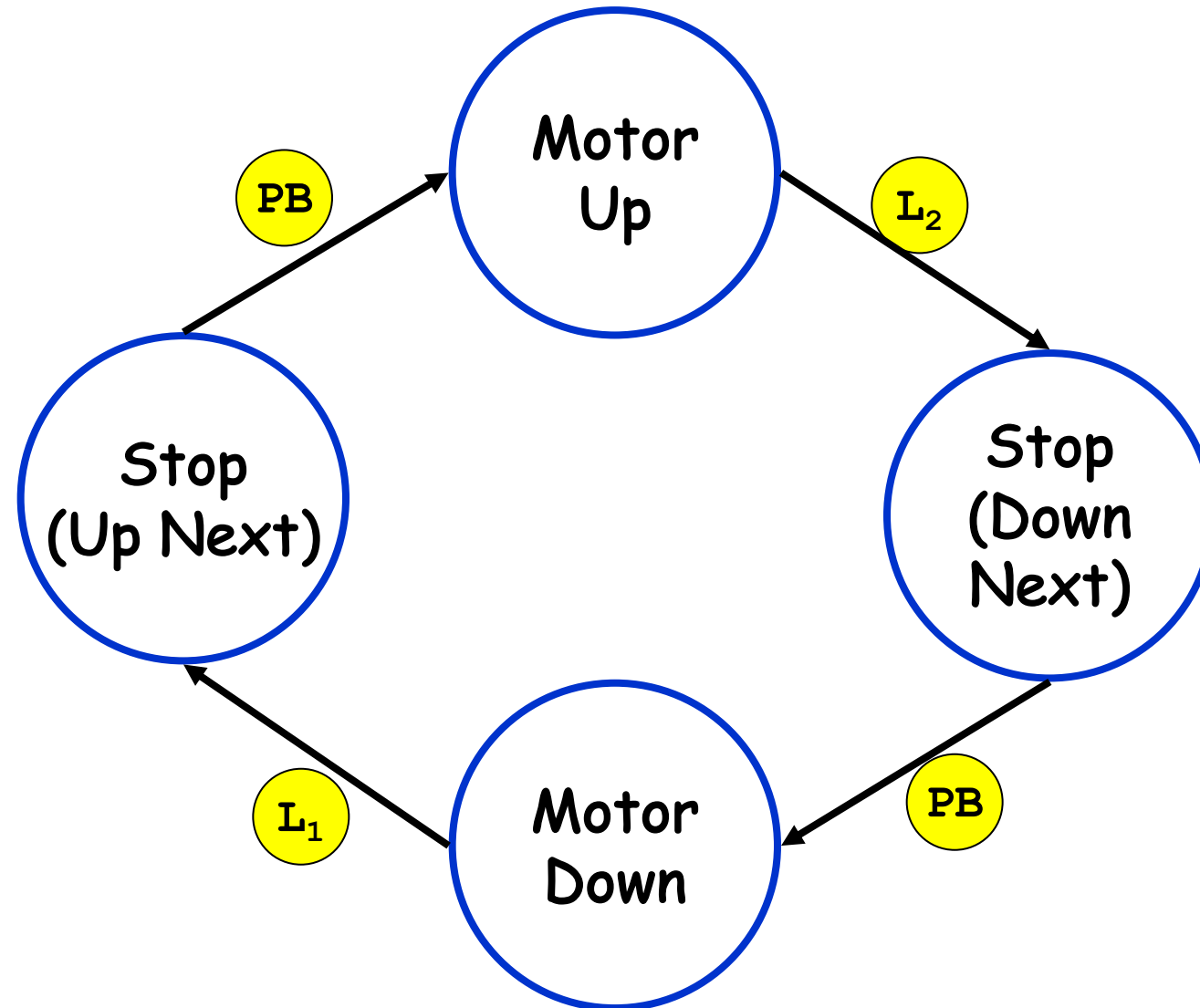


My Garage Door



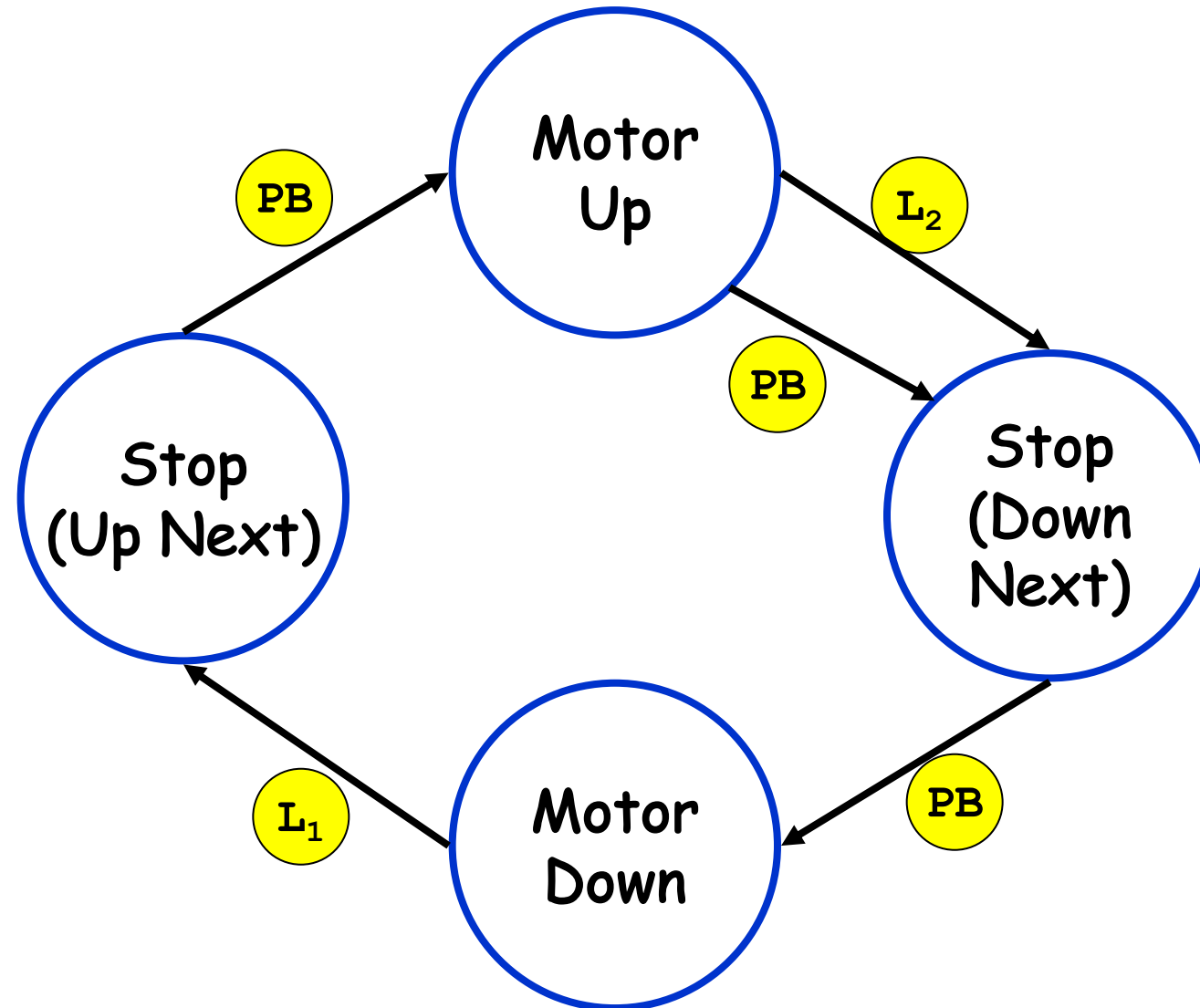
Start

State Machine



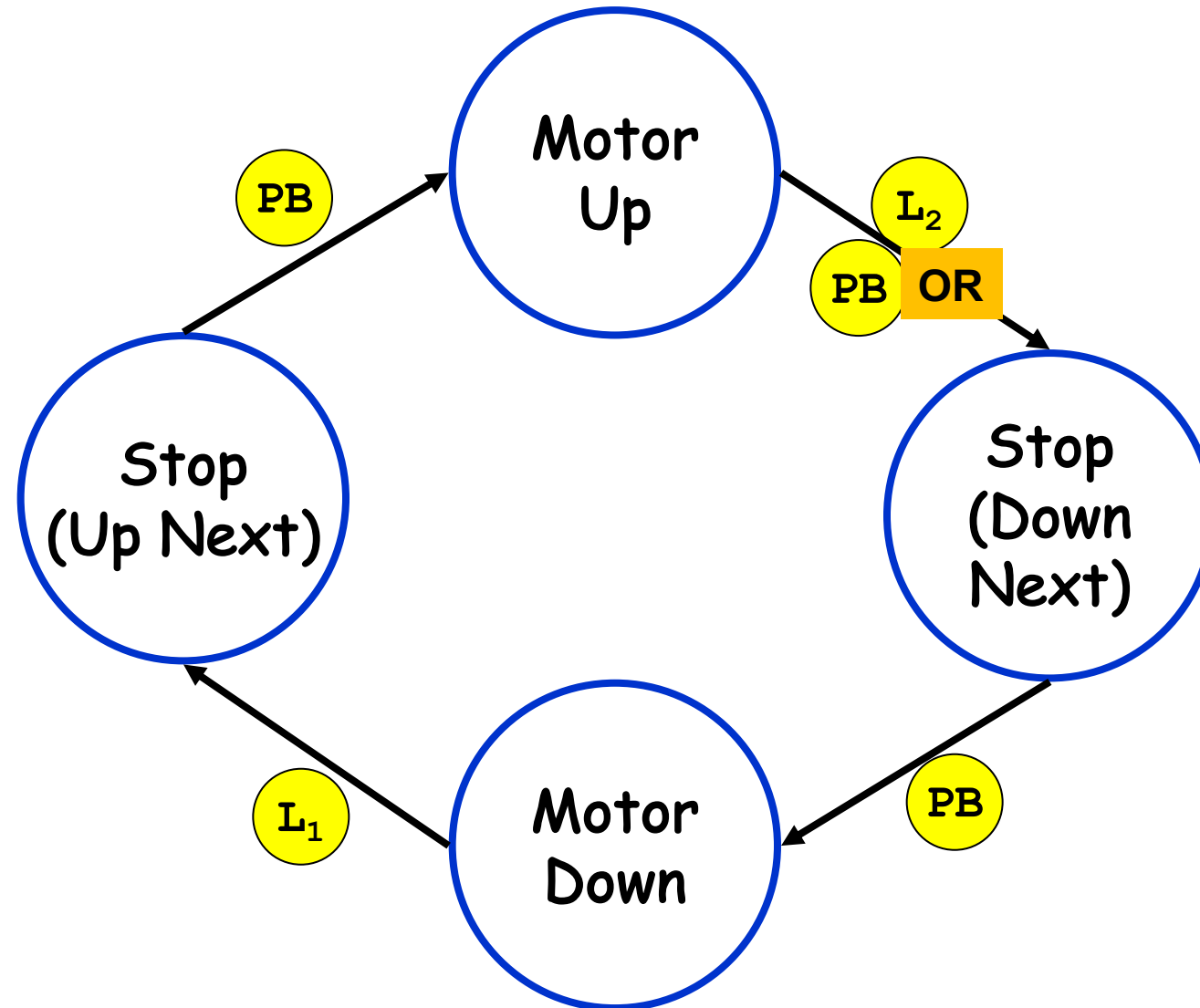
Start

State Machine



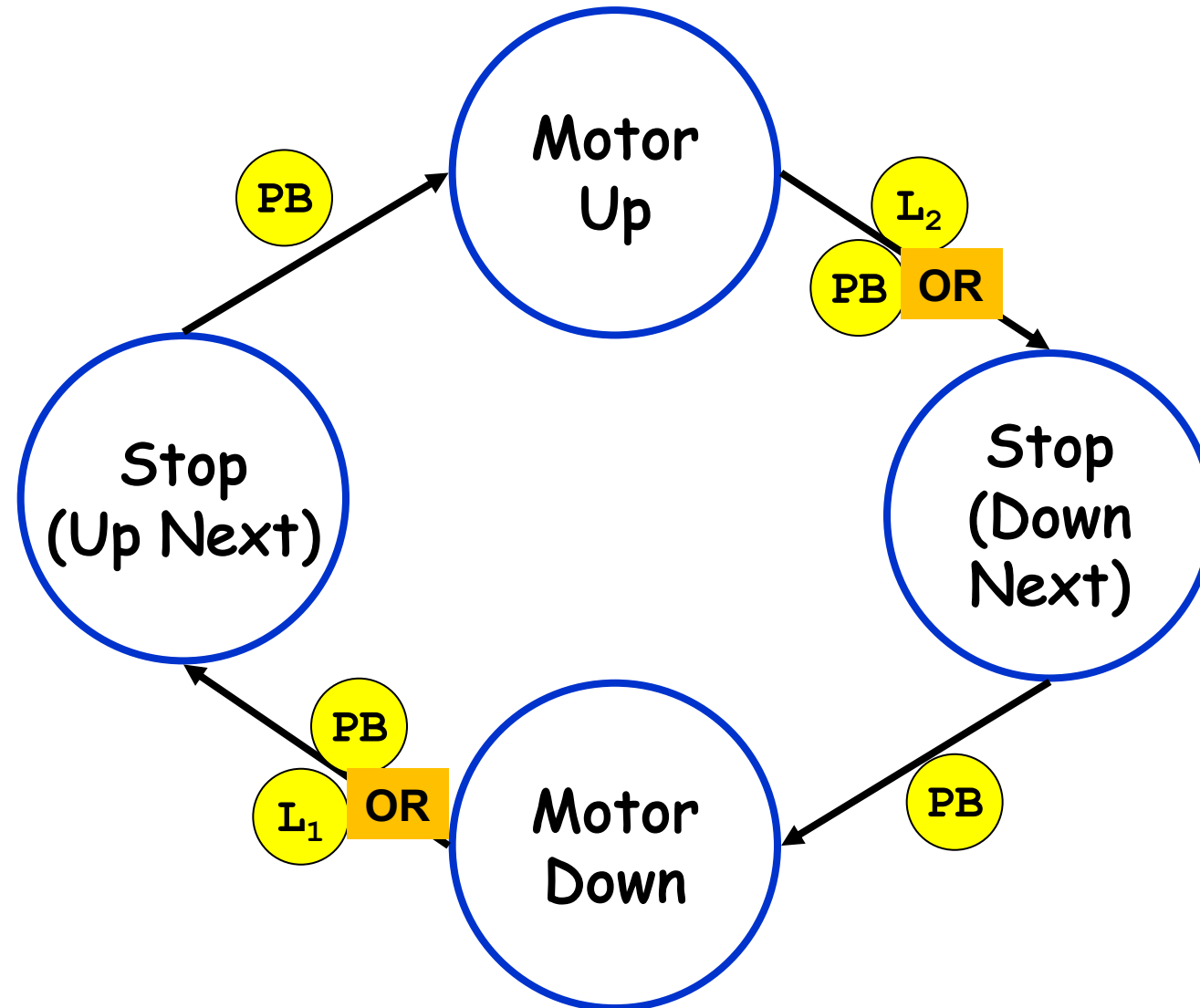
Start

State Machine



Start

State Machine



Suppose we wish to build a circuit to control our garage door?

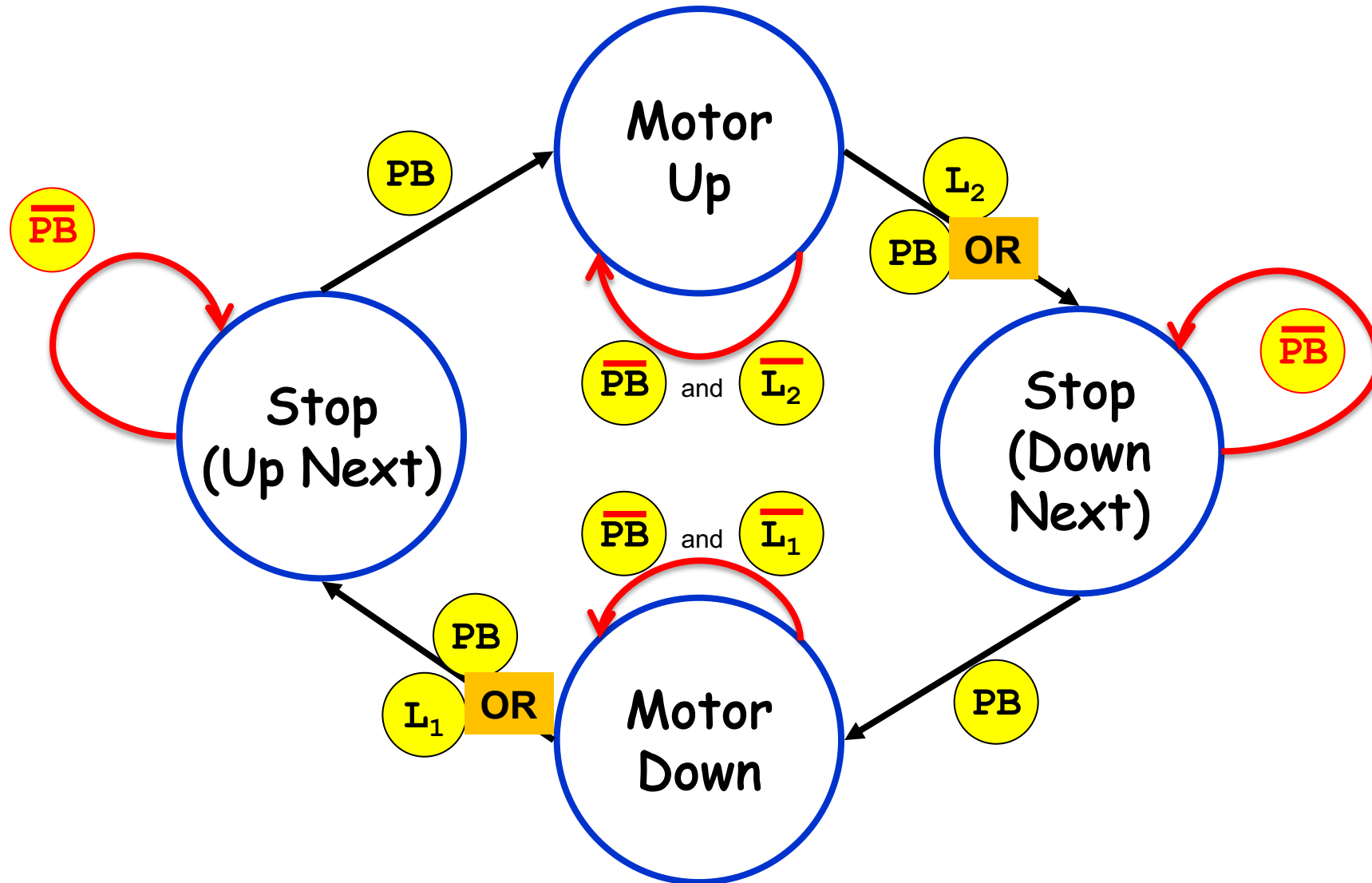
➤ What will we need?

What do we need?

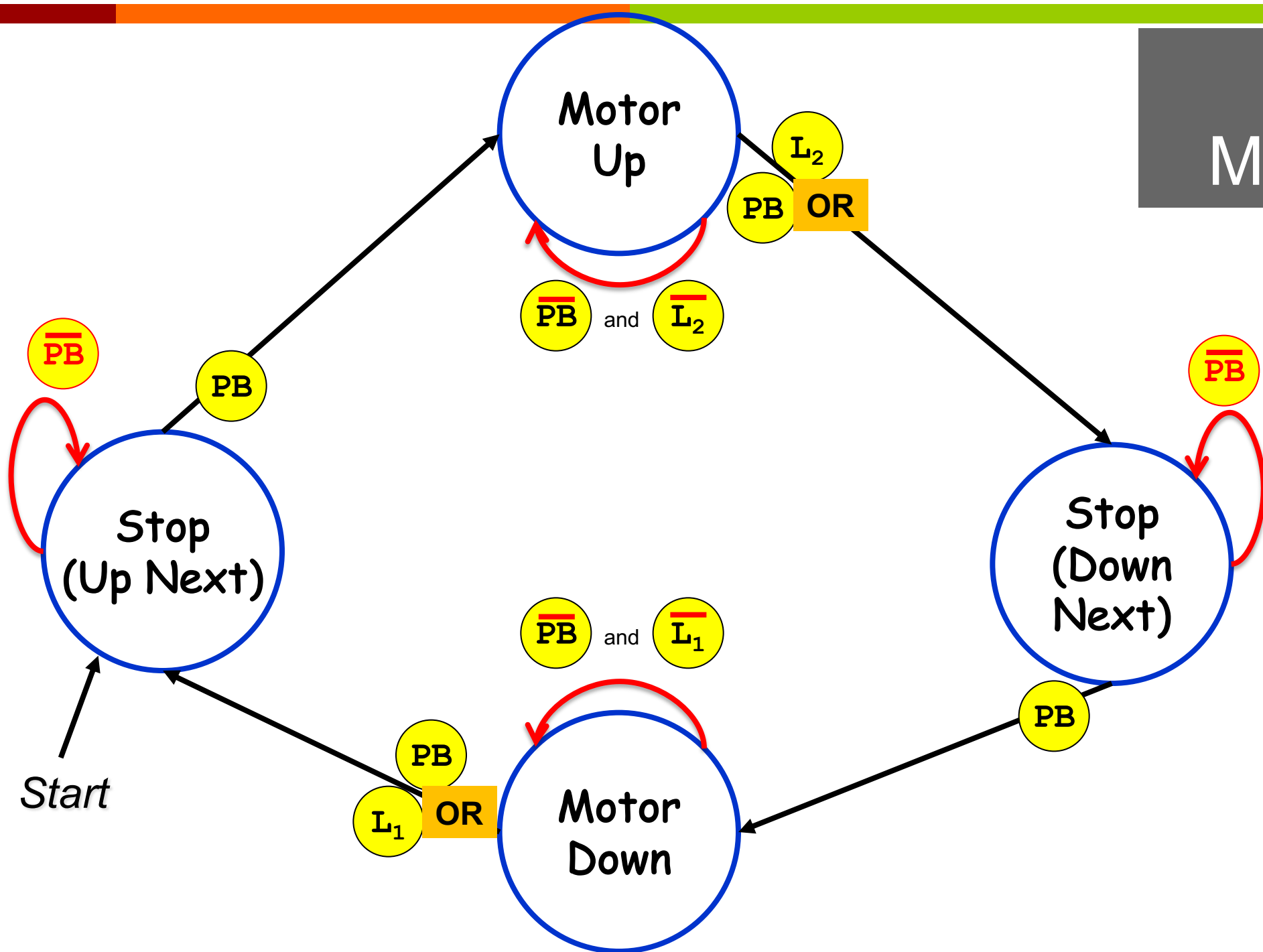
- Something to hold state
- Combinational logic
- A clock
- Reset button
- Limit switches
- The Pushbutton
- Handle being in no state! i.e. Startup
- What keeps you in a state?

Start

State Machine

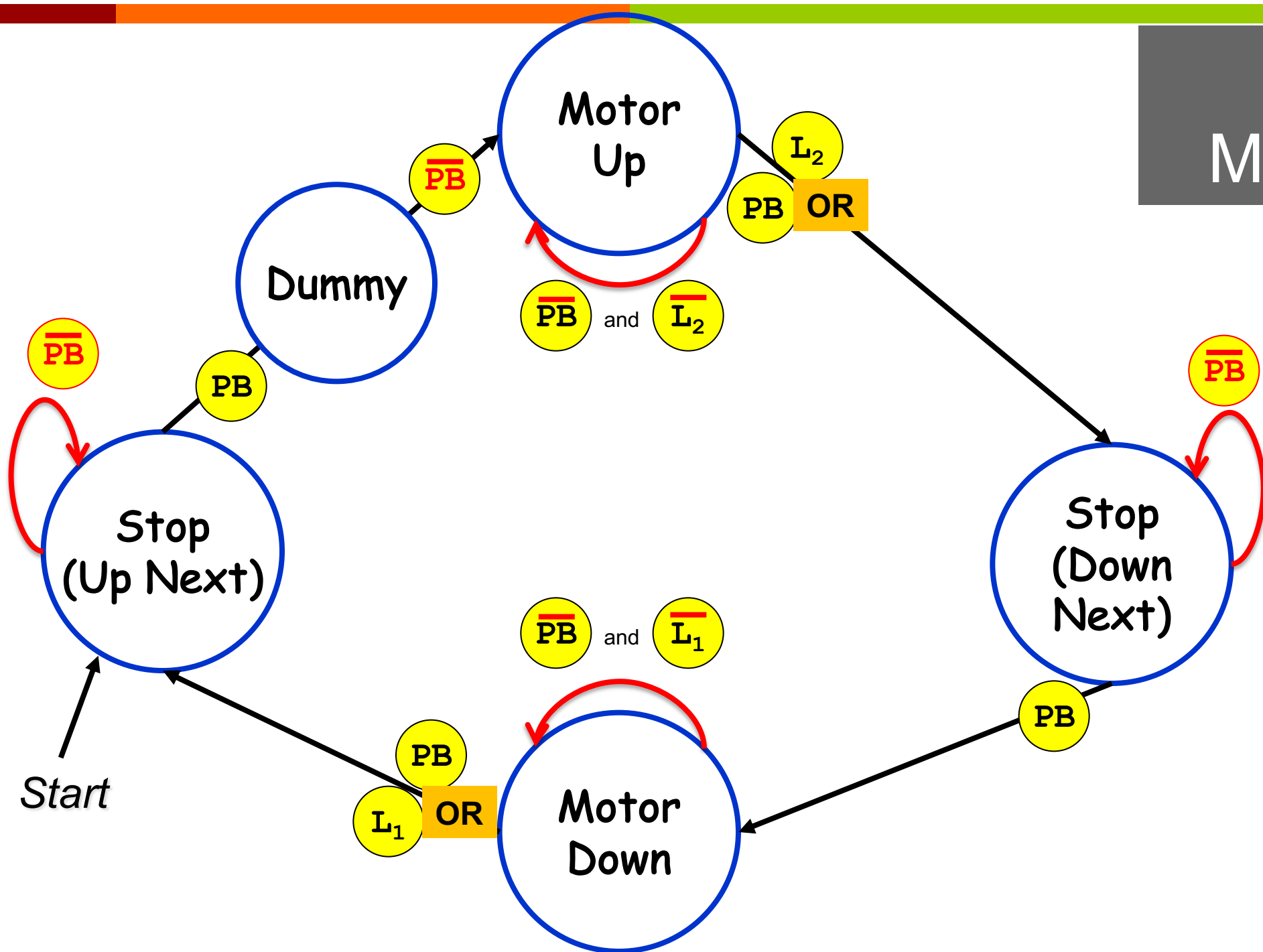


State Machine

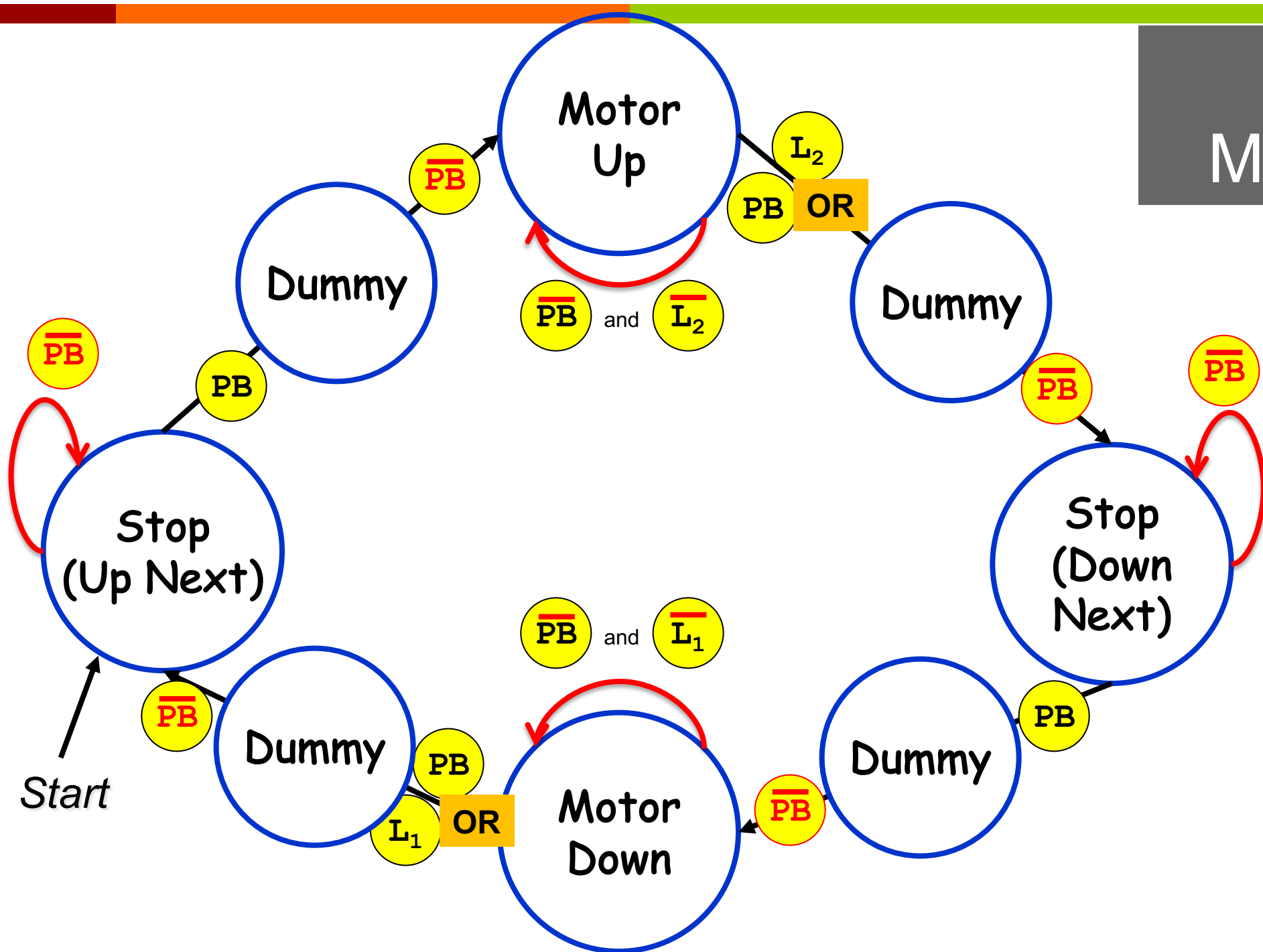


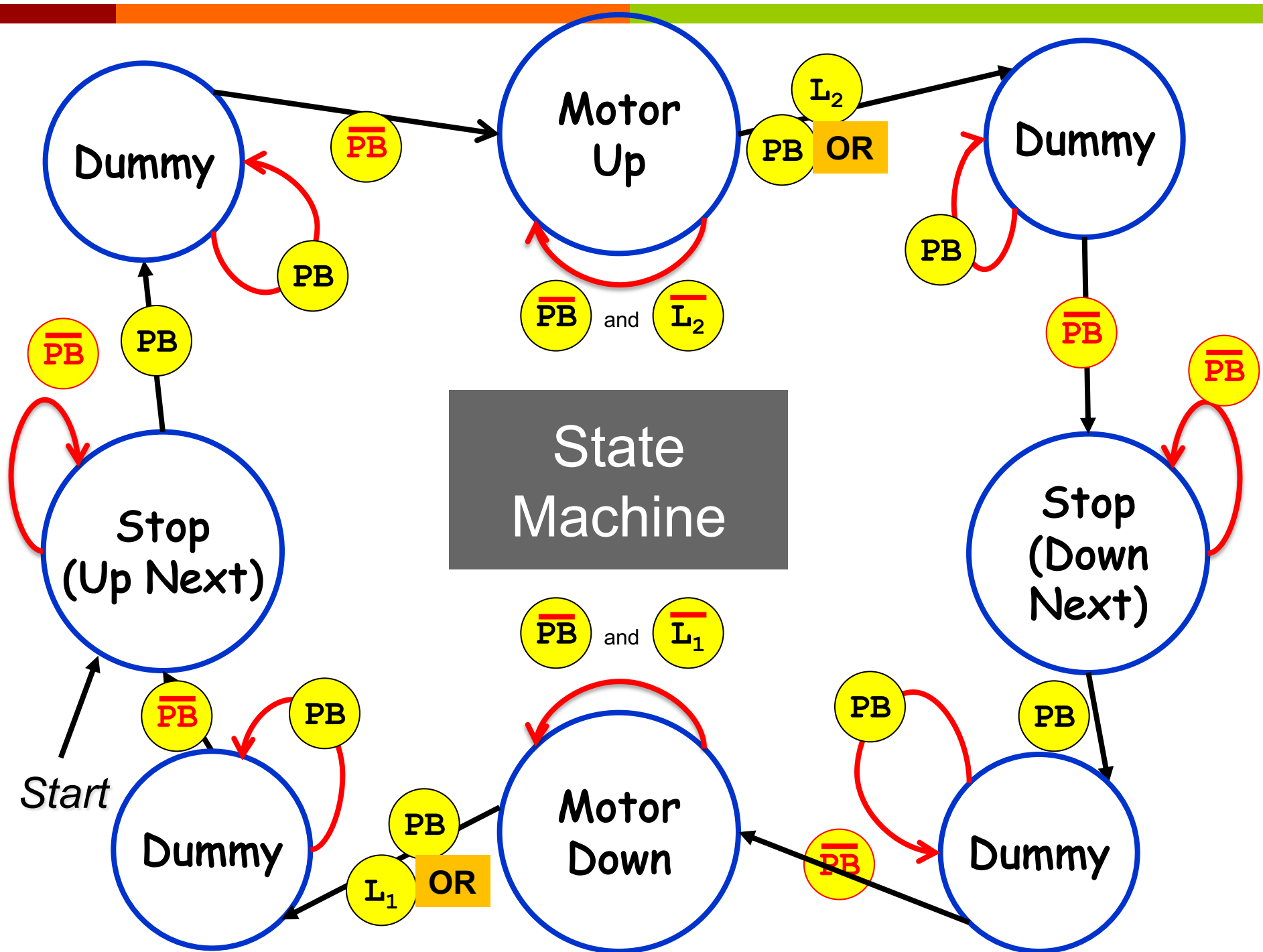
What happens if we just hold down the pushbutton?

State Machine

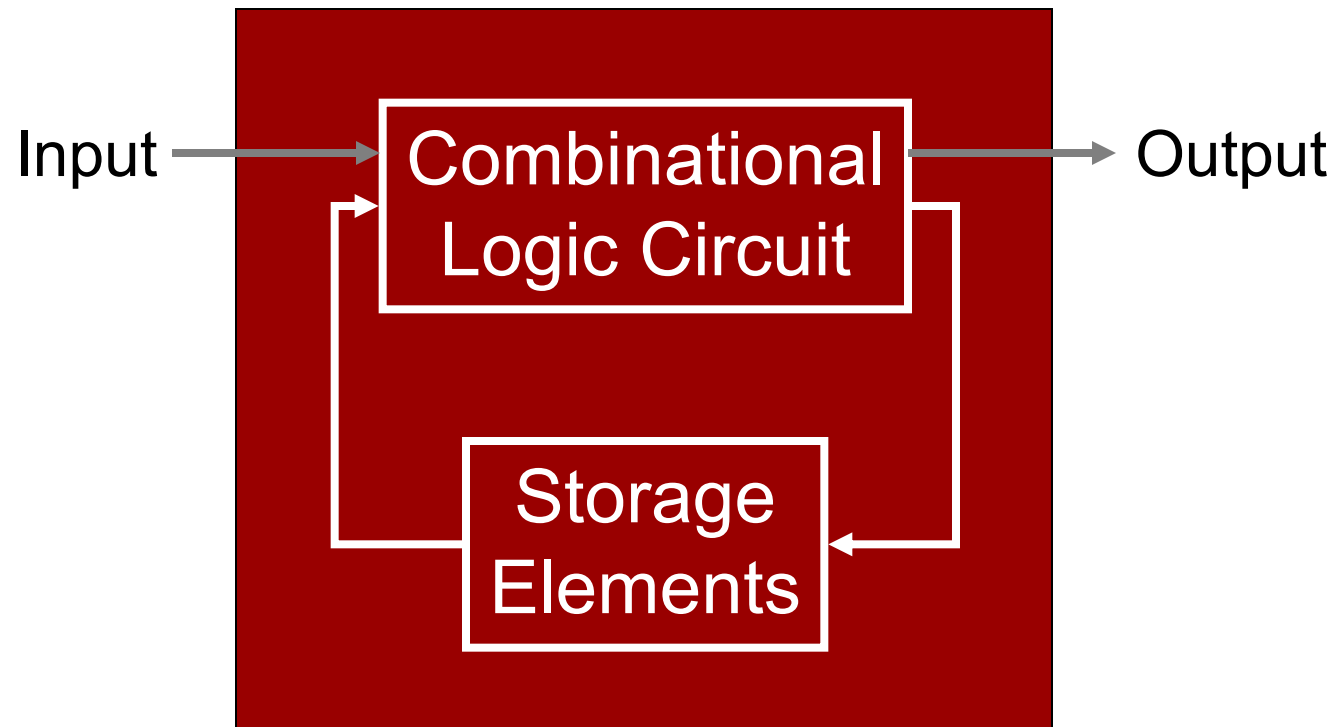


State Machine

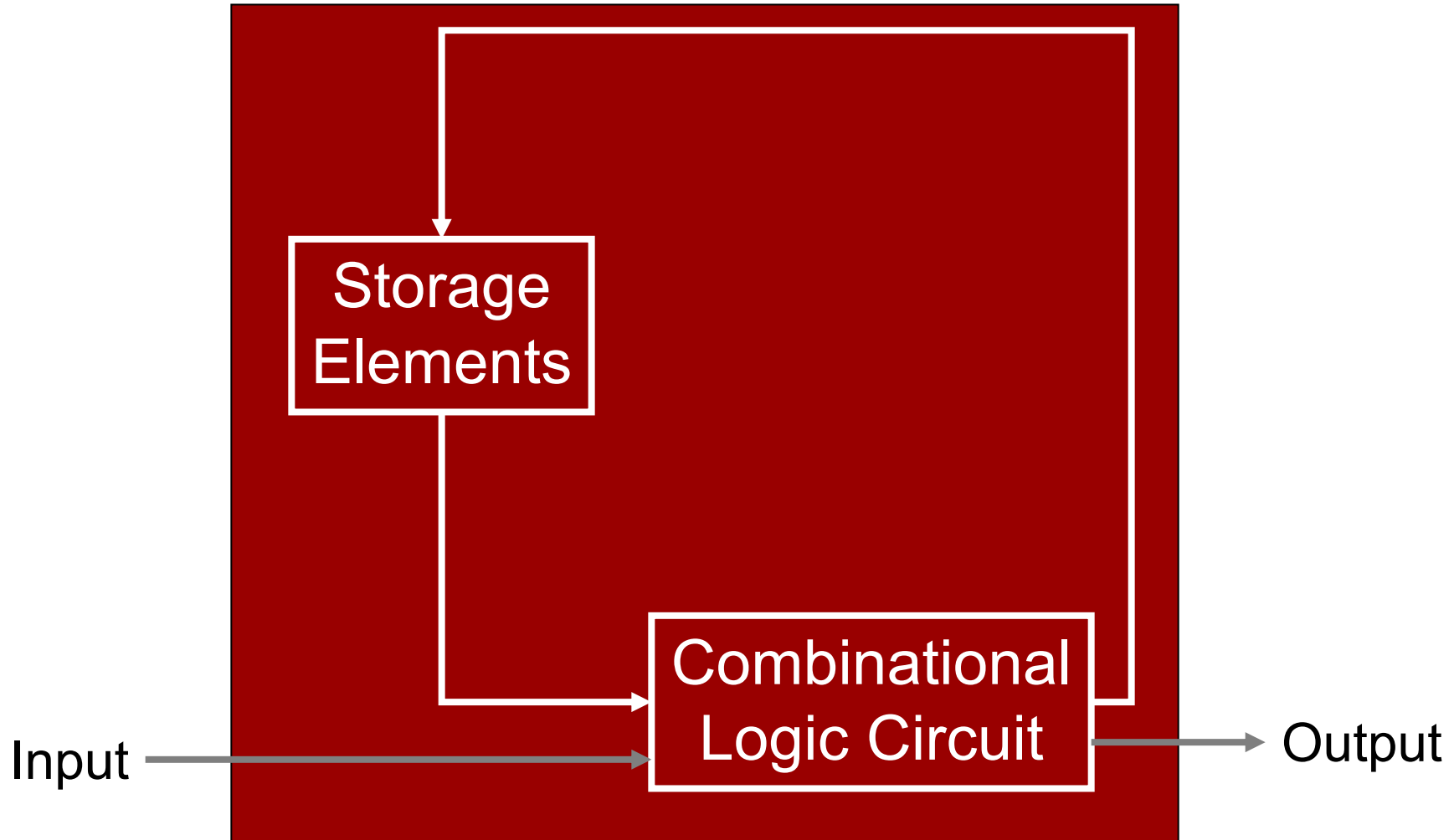




Sequential Logic Circuit



Sequential Logic Circuit



How many bits of storage are we going to need for the garage door opener state machine?

A. 1

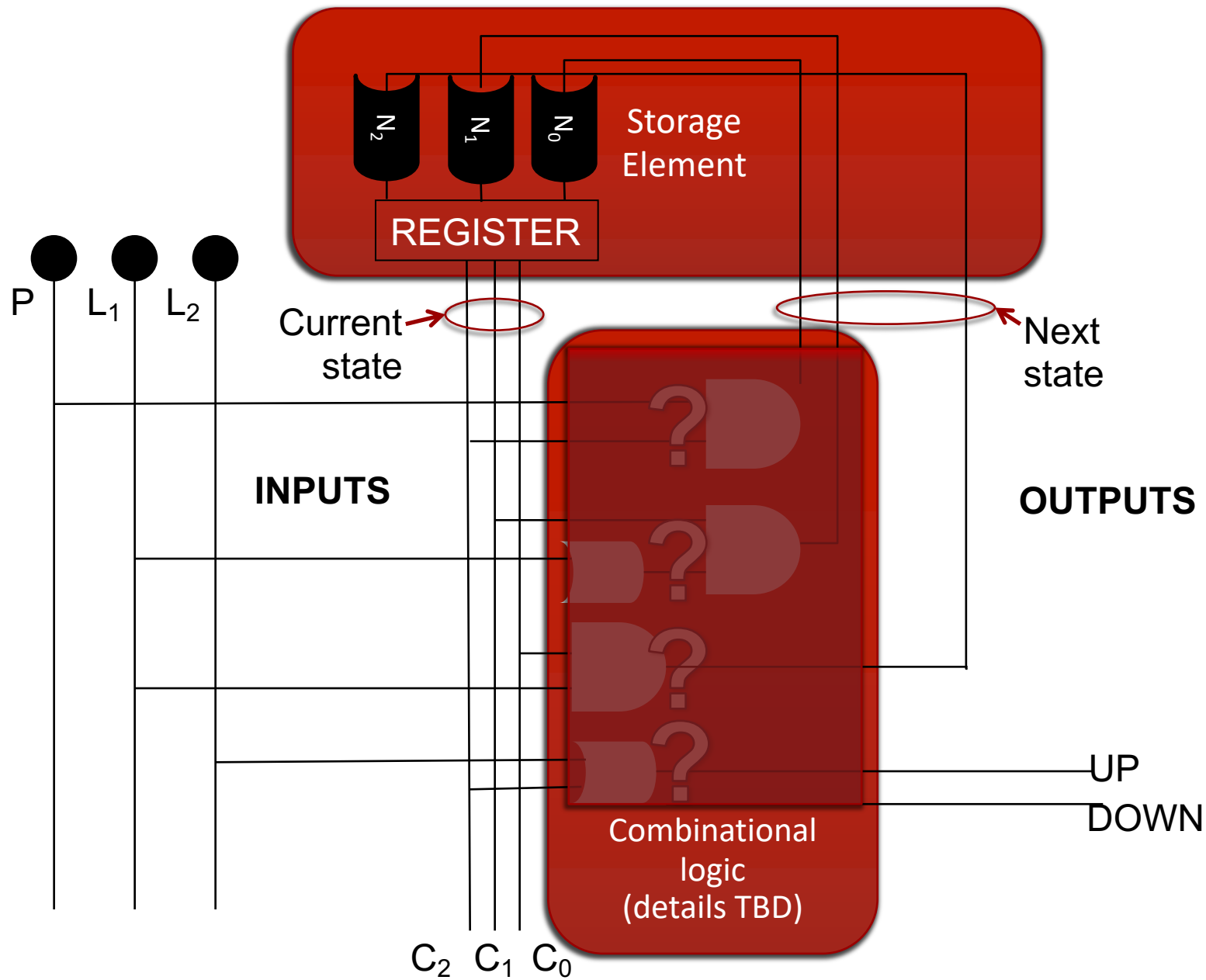
B. 2

C. 3

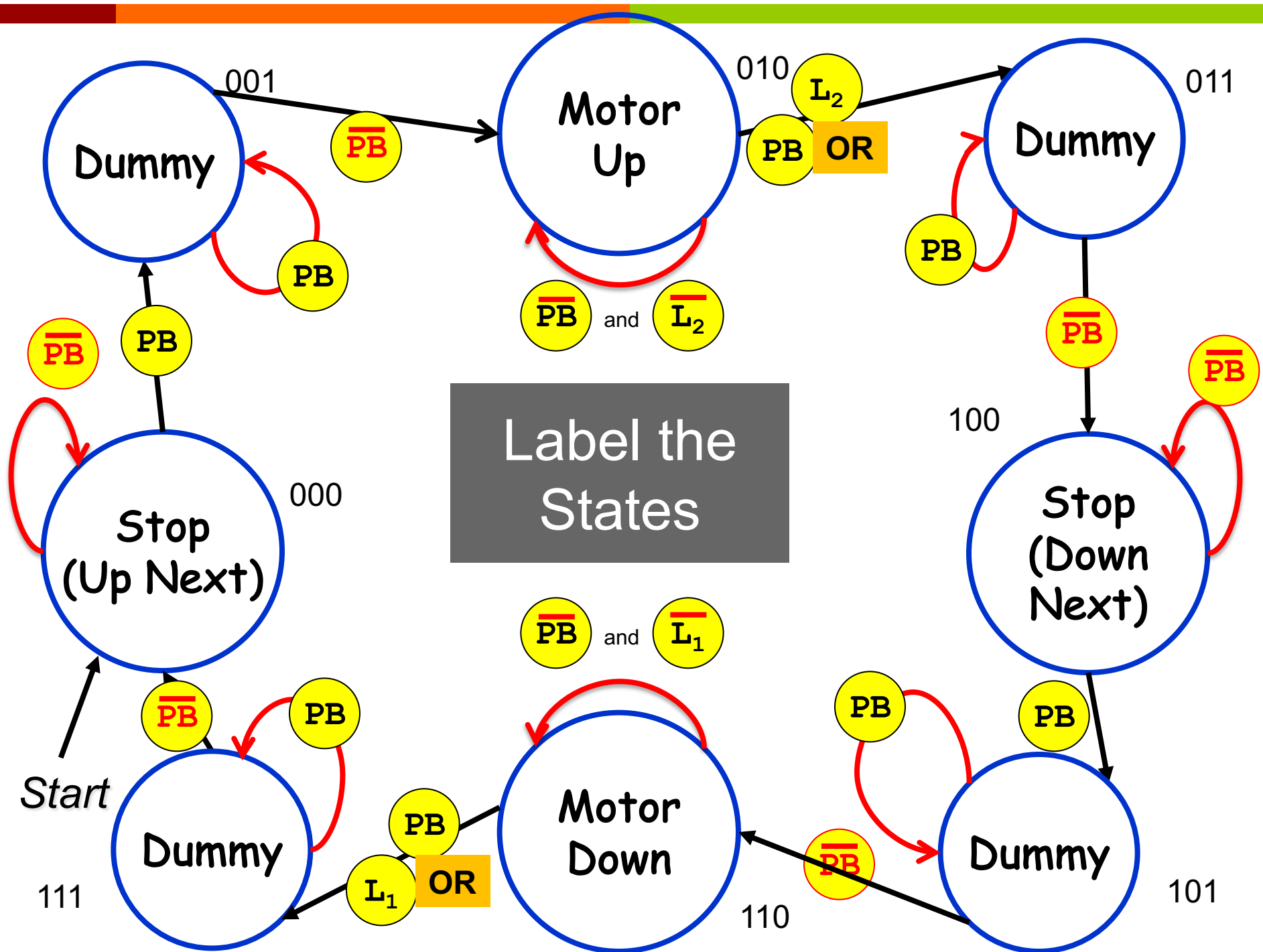


D. 4

E. 5



State Machine Circuit



P	L ₁	L ₂	C ₂	C ₁	C ₀	N ₂	N ₁	N ₀	U	D
0	x	x	0	0	0	0	0	0	0	0
1	x	x	0	0	0	0	0	1	0	0
0	x	x	0	0	1	0	1	0	0	0
1	x	x	0	0	1	0	0	1	0	0
0	x	0	0	1	0	0	1	0	1	0
x	x	1	0	1	0	0	1	1	1	0
1	x	x	0	1	0	0	1	1	1	0
0	x	x	0	1	1	1	0	0	0	0
1	x	x	0	1	1	0	1	1	0	0
0	x	x	1	0	0	1	0	0	0	0
1	x	x	1	0	0	1	0	1	0	0
0	x	x	1	0	1	1	1	0	0	0
1	x	x	1	0	1	1	0	1	0	0

What happens
in state 000

What happens
in state 001

What happens
in state 010

Circuit
Inputs

Current
state

Next state

Circuit
Outputs

P	L ₁	L ₂	C ₂	C ₁	C ₀	N ₂	N ₁	N ₀	U	D
0	0	x	1	1	0	1	1	0	0	1
1	x	x	1	1	0	1	1	1	0	1
x	1	x	1	1	0	1	1	1	0	1
0	x	x	1	1	1	0	0	0	0	0
1	x	x	1	1	1	1	1	1	0	0

Circuit Inputs

Current state

Next state

Circuit Outputs

With three inputs and three state bits, how many lines would the truth table require if it didn't contain "don't care" entries?

A. 16

B. 32

C. 64



Today's number: 84,210

D. 128

E. 256

Equations from the Truth Table...

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + \sim P C_2 \sim C_1 \sim C_0 \\ & + P C_2 \sim C_1 \sim C_0 \\ & + \sim P C_2 \sim C_1 C_0 \\ & + P C_2 \sim C_1 C_0 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + P C_2 C_1 \sim C_0 \\ & + L_1 C_2 C_1 \sim C_0 \\ & + P C_2 C_1 C_0 \end{aligned}$$

(Hint: Look down the N_2 column for the 1 bits – these are the input terms that make N_2 true!)

$$AB + \sim AB = B$$

$$\begin{aligned}
 N_2 = & \sim P \sim C_2 C_1 C_0 \\
 & + \sim P C_2 \sim C_1 \sim C_0 \\
 & + P C_2 \sim C_1 \sim C_0 \\
 & + \sim P C_2 \sim C_1 C_0 \\
 & + P C_2 \sim C_1 C_0 \\
 & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\
 & + P C_2 C_1 \sim C_0 \\
 & + L_1 C_2 C_1 \sim C_0 \\
 & + P C_2 C_1 C_0
 \end{aligned}$$

$$AB + \sim AB = B$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + C_2 \sim C_1 \sim C_0 \\ & + C_2 \sim C_1 C_0 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + P C_2 C_1 \sim C_0 \\ & + L_1 C_2 C_1 \sim C_0 \\ & + P C_2 C_1 C_0 \end{aligned}$$

$$A + B = B + A$$

$$\begin{aligned} N_2 = & \quad \sim P \sim C_2 C_1 C_0 \\ & + C_2 \sim C_1 \sim C_0 \\ & + C_2 \sim C_1 C_0 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + P C_2 C_1 \sim C_0 \\ & + L_1 C_2 C_1 \sim C_0 \\ & + P C_2 C_1 C_0 \end{aligned}$$

$$A + B = B + A$$

$$\begin{aligned} N_2 = & \quad \sim P \sim C_2 C_1 C_0 \\ & + \quad PC_2 C_1 C_0 \\ & + \quad C_2 \sim C_1 \sim C_0 \\ & + \quad C_2 \sim C_1 C_0 \\ & + \quad \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + \quad PC_2 C_1 \sim C_0 \\ & + \quad L_1 C_2 C_1 \sim C_0 \end{aligned}$$

$$AC + BC = (A + B)C$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \sim C_0 \\ & + C_2 \sim C_1 C_0 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + PC_2 C_1 \sim C_0 \\ & + L_1 C_2 C_1 \sim C_0 \end{aligned}$$

$$AB + BC = (A + B)C$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \sim C_0 \\ & + C_2 \sim C_1 C_0 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + (P + L_1) C_2 C_1 \sim C_0 \end{aligned}$$

$$AB + \sim AB = B$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \sim C_0 \\ & + C_2 \sim C_1 C_0 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + (P + L_1) C_2 C_1 \sim C_0 \end{aligned}$$

$$AB + \sim AB = B$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + (P + L_1) C_2 C_1 \sim C_0 \end{aligned}$$

$$AC + BC = (A + B)C$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \\ & + \sim P \sim L_1 C_2 C_1 \sim C_0 \\ & + (P + L_1) C_2 C_1 \sim C_0 \end{aligned}$$

$$AC + BC = (A + B)C$$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \\ & + ((\sim P \sim L_1) + (P + L_1)) C_2 C_1 \sim C_0 \end{aligned}$$

DeMorgan and $(A + \sim A)B = B$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + PC_2 C_1 C_0 \\ & + C_2 \sim C_1 \\ & + (\sim P \sim L_1 + P + L_1) C_2 C_1 \sim C_0 \end{aligned}$$

DeMorgan and $(A + \sim A)B = B$

$$\begin{aligned} N_2 = & \sim P \sim C_2 C_1 C_0 \\ & + P C_2 C_1 C_0 \\ & + C_2 \sim C_1 \\ & + C_2 C_1 \sim C_0 \end{aligned}$$

And We're Down to 4 Variables...

$$\begin{aligned} N_2 = & \quad \sim P \sim C_2 C_1 C_0 \\ & + \quad P C_2 C_1 C_0 \\ & + \quad C_2 \sim C_1 \\ & + \quad C_2 C_1 \sim C_0 \end{aligned}$$

$$N_2 = \sim P \sim C_2 C_1 C_0 + P C_2 C_1 C_0 + C_2 \sim C_1 + C_2 C_1 \sim C_0$$

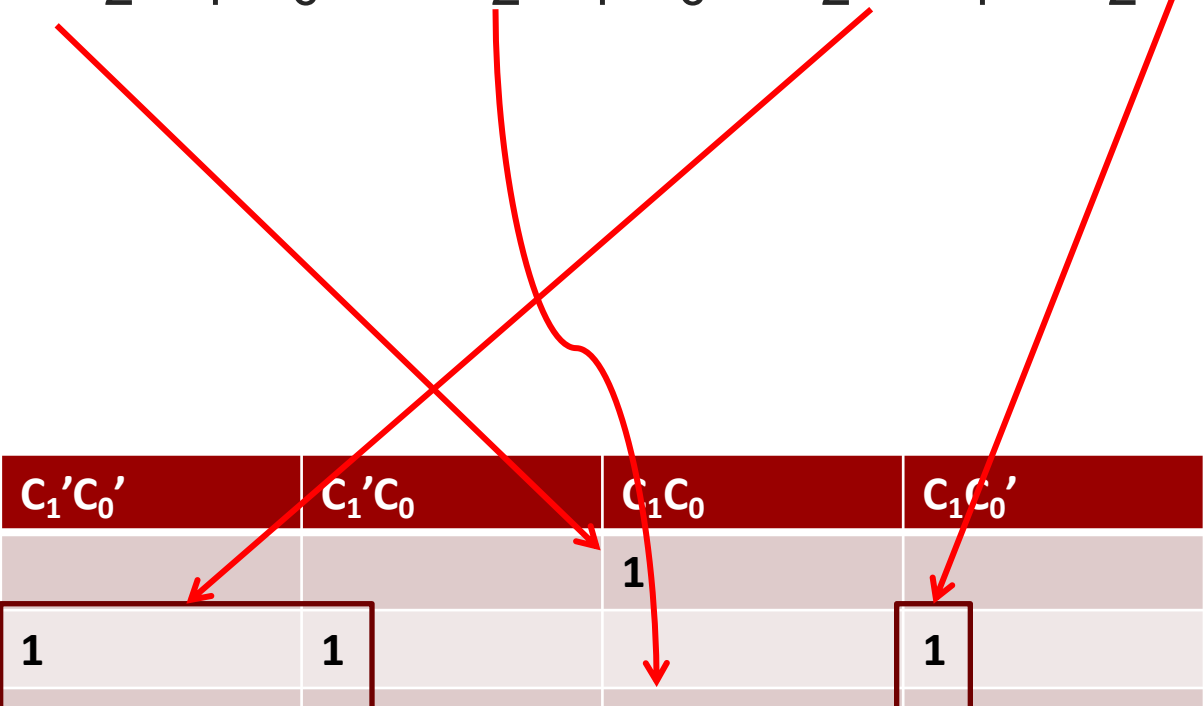
How About a K-Map To Go Farther?

$$N_2 = \sim P \sim C_2 C_1 C_0 + P C_2 C_1 C_0 + C_2 \sim C_1 + C_2 C_1 \sim C_0$$

	$C_1' C_0'$	$C_1' C_0$	$C_1 C_0$	$C_1 C_0'$
$P' C_2'$				
$P' C_2$				
$P C_2$				
$P C_2'$				

Fill Out the Map

$$N_2 = \sim P \sim C_2 C_1 C_0 + P C_2 C_1 C_0 + C_2 \sim C_1 + C_2 C_1 \sim C_0$$



	$C_1' C_0'$	$C_1' C_0$	$C_1 C_0$	$C_1 C_0'$
$P' C_2'$			1	
$P' C_2$	1	1		1
$P C_2$	1	1	1	1
$P C_2'$				

Look for Power-of-2 Rectangles

$$N_2 = \sim P \sim C_2 C_1 C_0 + P C_2 C_1 C_0 + C_2 \sim C_1 + C_2 C_1 \sim C_0$$

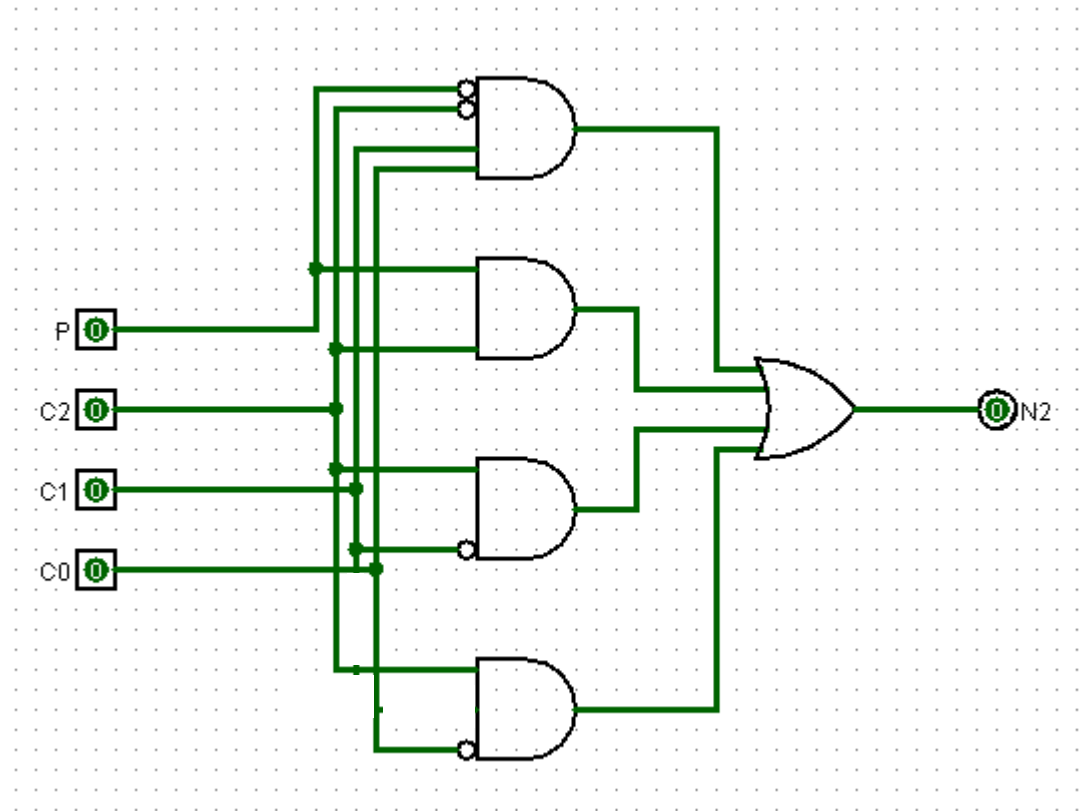
	$C_1' C_0'$	$C_1' C_0$	$C_1 C_0$	$C_1 C_0'$
$P' C_2'$			1	
$P' C_2$	1	1		1
$P C_2$	1	1	1	1
$P C_2'$				

$$N_2 = \sim P \sim C_2 C_1 C_0 + P C_2 C_1 C_0 + C_2 \sim C_1 + C_2 C_1 \sim C_0$$

	$C_1' C_0'$	$C_1' C_0$	$C_1 C_0$	$C_1 C_0'$
$P' C_2'$			1	
$P' C_2$	1	1		1
$P C_2$	1	1	1	1
$P C_2'$				

$$N_2 = \sim P \sim C_2 C_1 C_0 + P C_2 + C_2 \sim C_1 + C_2 \sim C_0$$

Draw the Circuit from the Boolean...



How Many Circuits?

We've designed the circuit for the N_2 output. How many more circuits do we need to design to complete the state machine?

A. 1

Why?

B. 2

Go back and look at the truth table

C. 3

We have 5 output columns, each of which needs a circuit to compute it

D. 4



E. 5

Let's look at the circuit for U

P	L ₁	L ₂	C ₂	C ₁	C ₀	N ₂	N ₁	N ₀	U	D
0	x	x	0	0	0	0	0	0	0	0
1	x	x	0	0	0	0	0	1	0	0
0	x	x	0	0	1	0	1	0	0	0
1	x	x	0	0	1	0	0	1	0	0
0	x	0	0	1	0	0	1	0	1	0
x	x	1	0	1	0	0	1	1	1	0
1	x	x	0	1	0	0	1	1	1	0
0	x	x	0	1	1	1	0	0	0	0
1	x	x	0	1	1	0	1	1	0	0
0	x	x	1	0	0	1	0	0	0	0
1	x	x	1	0	0	1	0	1	0	0
0	x	x	1	0	1	1	1	0	0	0
1	x	x	1	0	1	1	0	1	0	0

The only place
U=1 is here

Circuit
Inputs

Current
state

Next state

Circuit
Outputs

P	L ₁	L ₂	C ₂	C ₁	C ₀	N ₂	N ₁	N ₀	U	D
0	0	x	1	1	0	1	1	0	0	1
1	x	x	1	1	0	1	1	1	0	1
x	1	x	1	1	0	1	1	1	0	1
0	x	x	1	1	1	0	0	0	0	0
1	x	x	1	1	1	1	1	1	0	0

Circuit Inputs

Current state

Next state

Circuit Outputs

One More Example

$$\begin{aligned} \nearrow U = & P'L_2'C_2'C_1C_0' \\ & + L_2C_2'C_1C_0' \\ & + PC_2'C_1C_0' \end{aligned}$$

$$A + A'B = A + B$$

$$\begin{aligned} \Rightarrow U &= P'L_2'C_2'C_1C_0' \\ &\quad + L_2C_2'C_1C_0' \\ &\quad + PC_2'C_1C_0' \end{aligned}$$

$$A + A' = 1$$

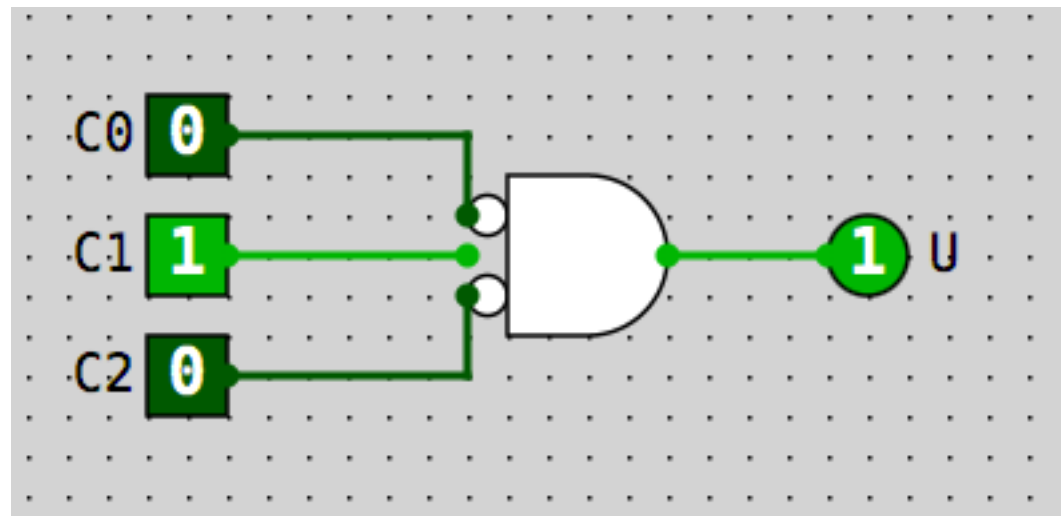
$$\begin{aligned} \Rightarrow U = & \textcolor{red}{P}'C_2'C_1C_0' \\ & + L_2C_2'C_1C_0' \\ & + \textcolor{red}{P}C_2'C_1C_0' \end{aligned}$$

$$A + AB = A$$

$$\nearrow U = C_2' C_1 C_0' + L_2 C_2' C_1 C_0'$$

At last!

➤ $U = C_2' C_1 C_0'$



When is the only time U is 1?

And we must still make circuits for N_1 , N_0 , and D

The Other Three Circuits

➤ $N_1 = C_1 \sim C_0 + PC_1 + \sim P \sim C_1 C_0$

➤ $N_0 = P$

➤ $D = C_2 C_1 \sim C_0$

Two Kinds of State Machines!

One Hot

- One bit per state
- Only one bit is on at a time
- Faster
- Requires more flip flops
- States progress
00001»00010»00100»
01000»10000

Binary Encoded

- Encode state as a binary number
- Use a decoder to generate a line for each state
- Slower
- More complicated
- States progress
000»001»010»011»100

Which one did we just use to implement the garage door opener?

- Sequential Logic Circuits
 - State
 - Memory
 - Address space
 - Addressability
 - $2^2 \times 3$ bit memory
 - Clocks
 - Edge-triggered and level-triggered flip-flops
 - Example State Machine
 - Design
 - Simplification
 - Implementation
 - One-hot and Encoded State Machines