

---

# Feedback and Flip-Flops

Philipp Koehn

X September 2019  
6

CA Office hours start  
Next week  
check Piazza



HW1 due Friday 9/13  
ASK QUESTIONS!  
(again, Piazza)

# The Story So Far



- We can encode numbers

# The Story So Far



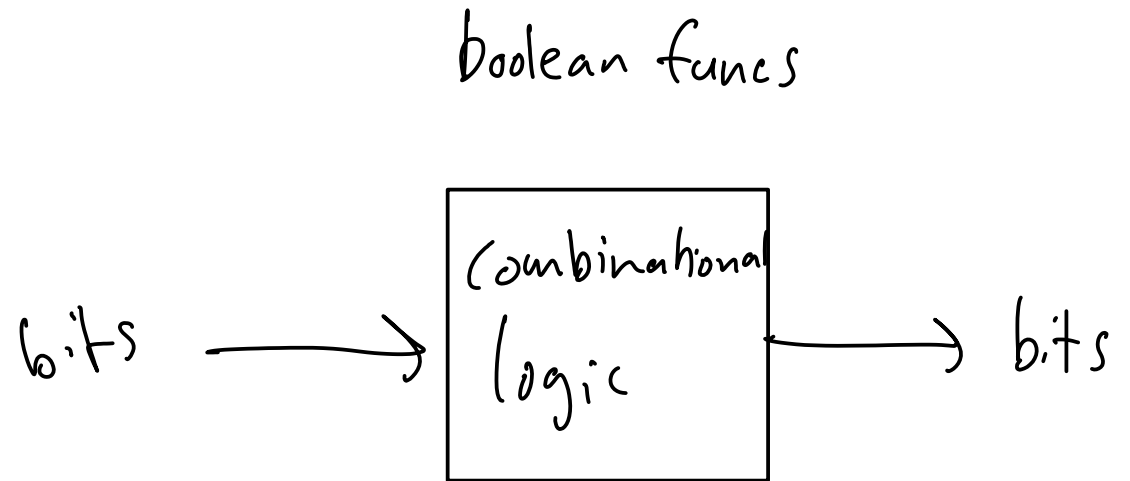
- We can encode numbers
- We can do calculation

# The Story So Far



1

- We can encode numbers
- We can do calculation
- ... but it's all a bit static



no memory

# The Story So Far



- We can encode numbers
- We can do calculation
- ... but it's all a bit static
  
- How about a counter?

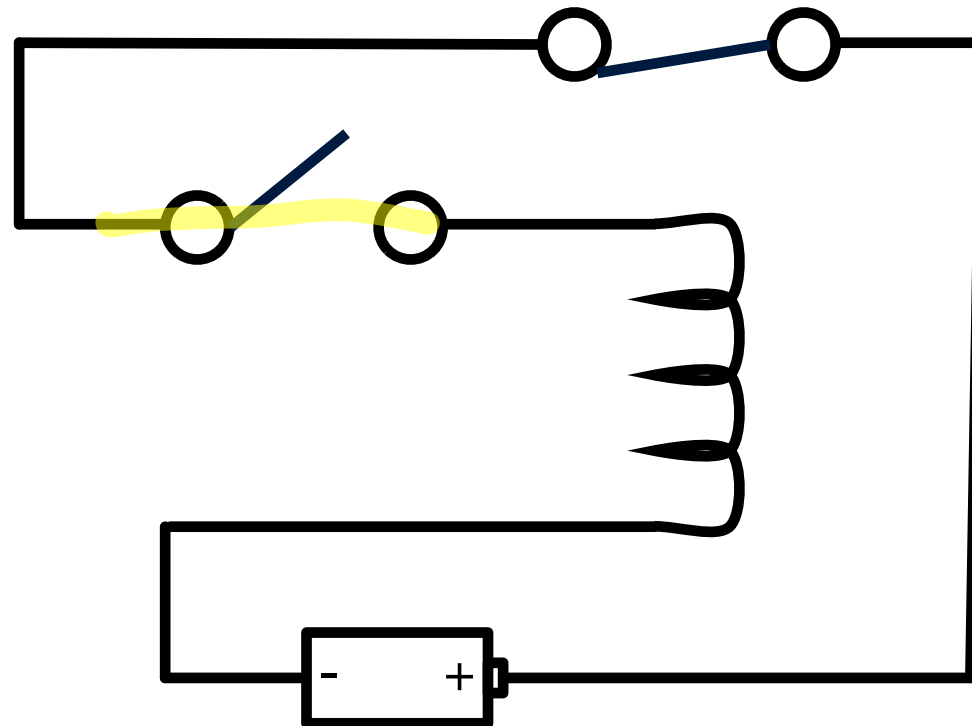
# The Story So Far



- We can encode numbers
- We can do calculation
- ... but it's all a bit static
  
- How about a counter?  
→ this requires "memory"

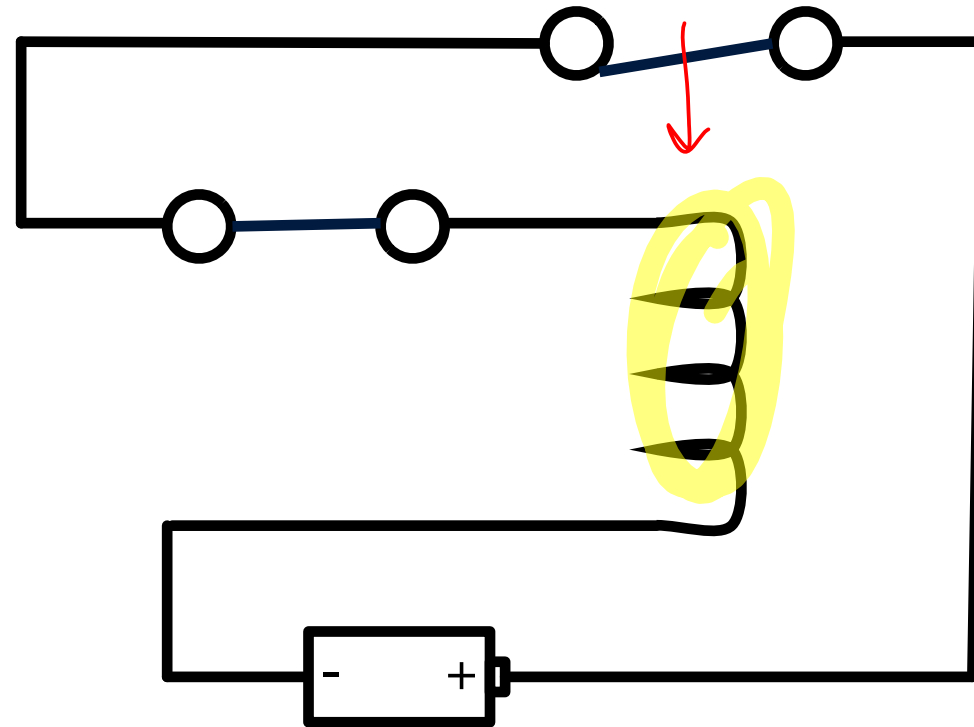
# feedback

# A Strange Contraption



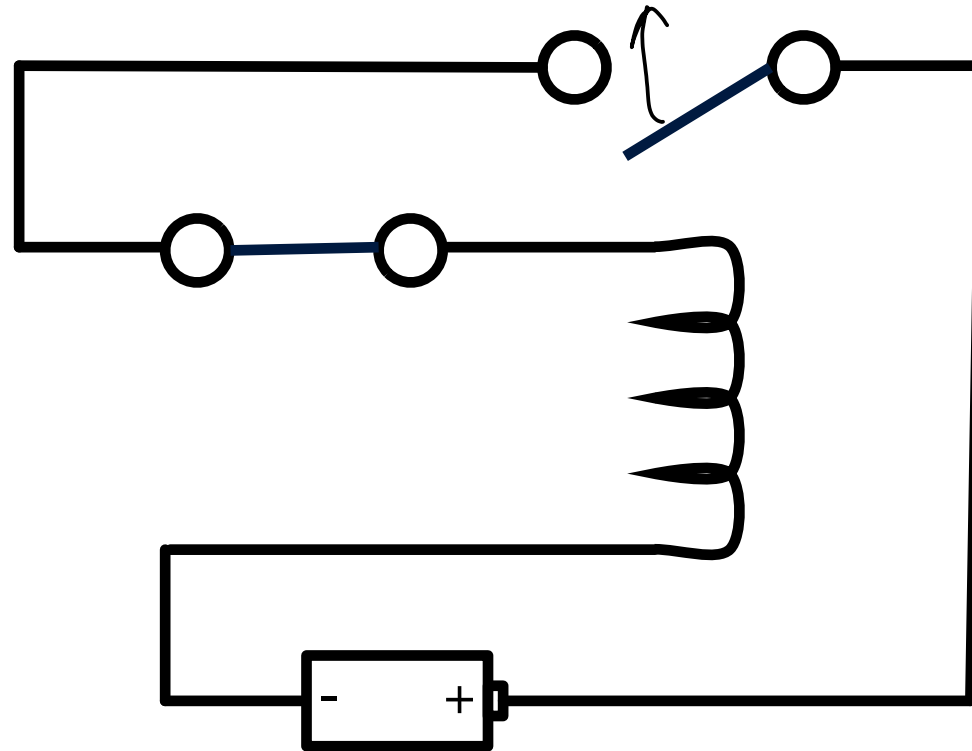


# Let's Turn It On



Electricity is on → this opens the normally closed key

# Let's Turn It On



Electricity is off → this closes the normally closed key

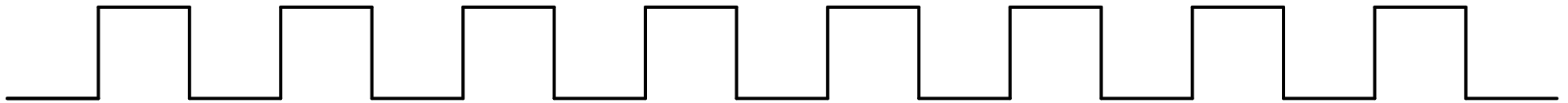
# What Do We Have?



- A Buzzer

# What Do We Have?

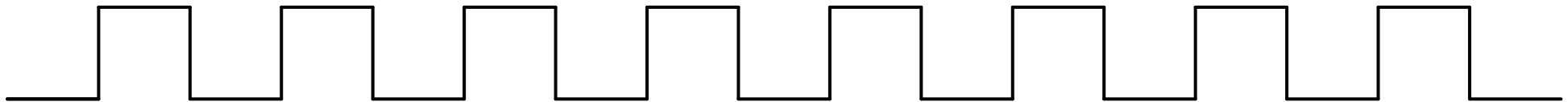
- A Buzzer
- A Clock



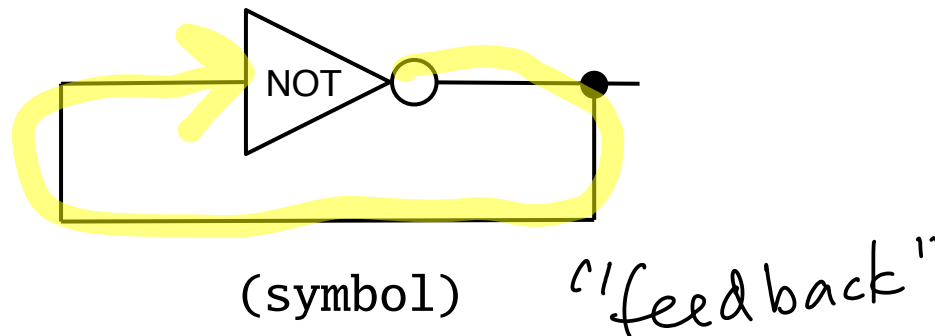
# What Do We Have?

- A Buzzer

- A Clock



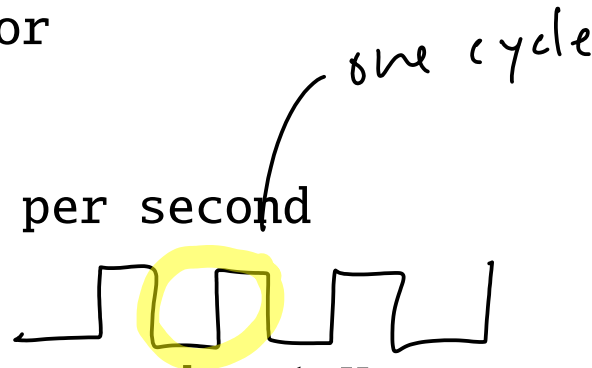
- An Oscillator



# Oscillator



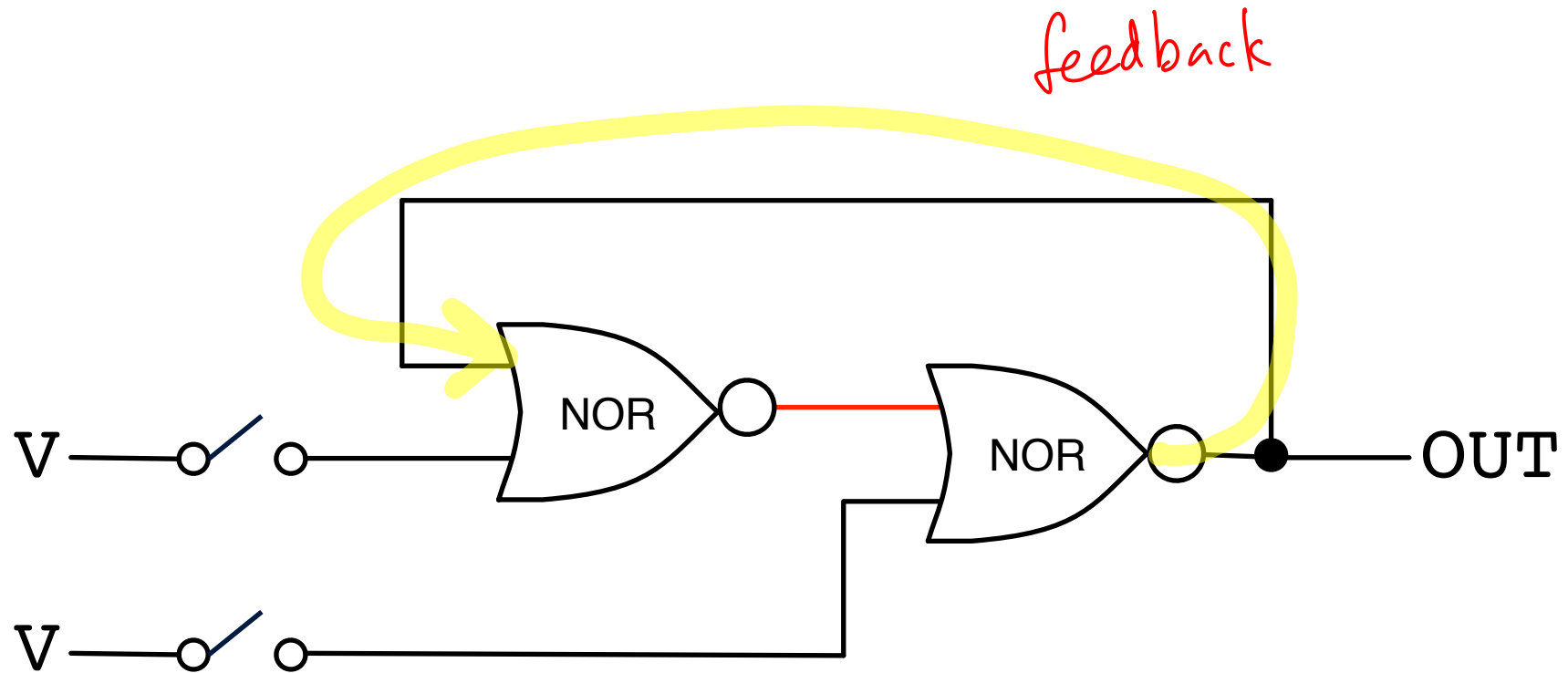
- **Period** of oscillator
- Frequency: cycles per second
- Unit: 1 cycle per second: 1 Hertz
- Modern computes:  
Billions of Hertz = Gigahertz (GHz)



Heinrich Hertz  
1857--1894

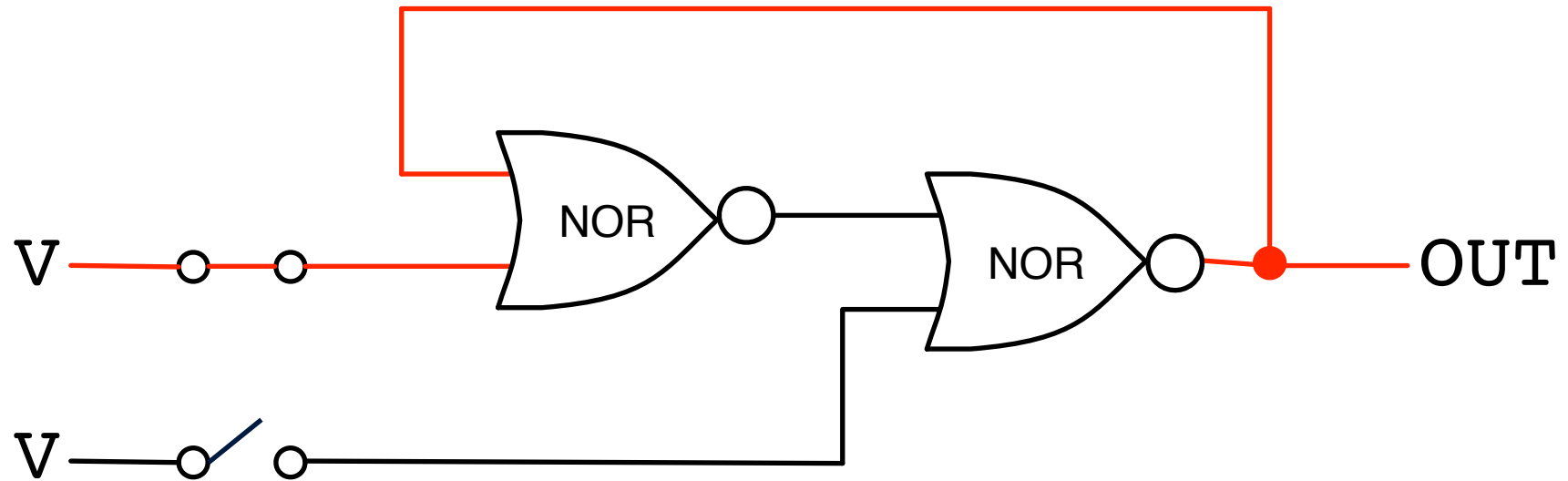
flip flop

# Another Contraption

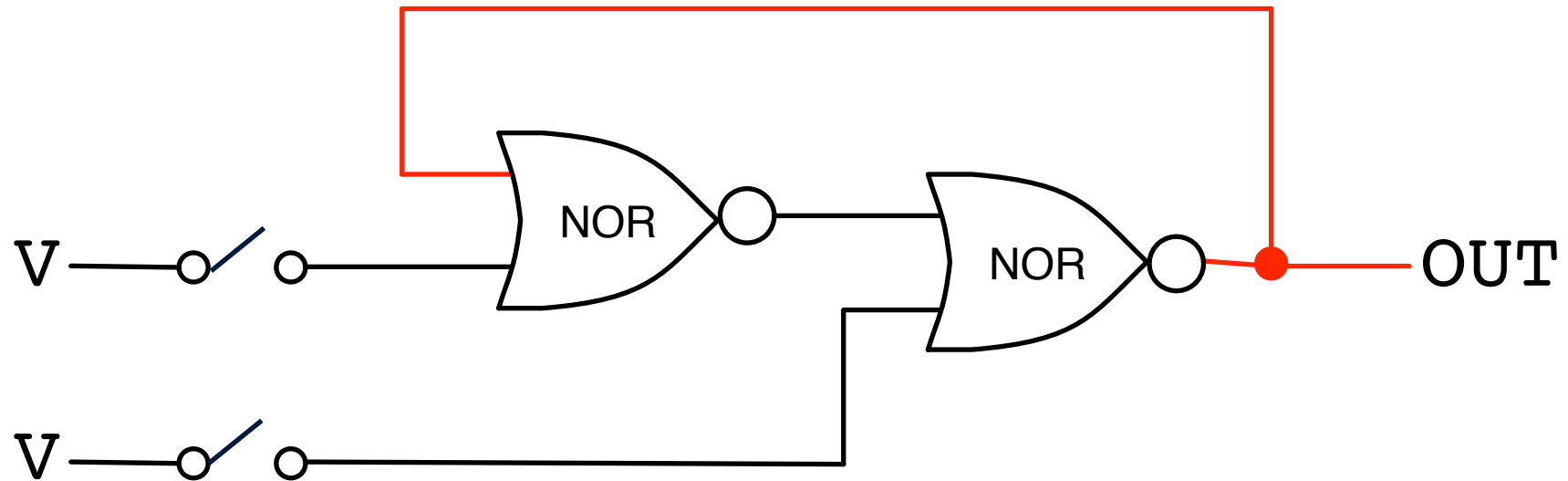




# Closing Upper Key



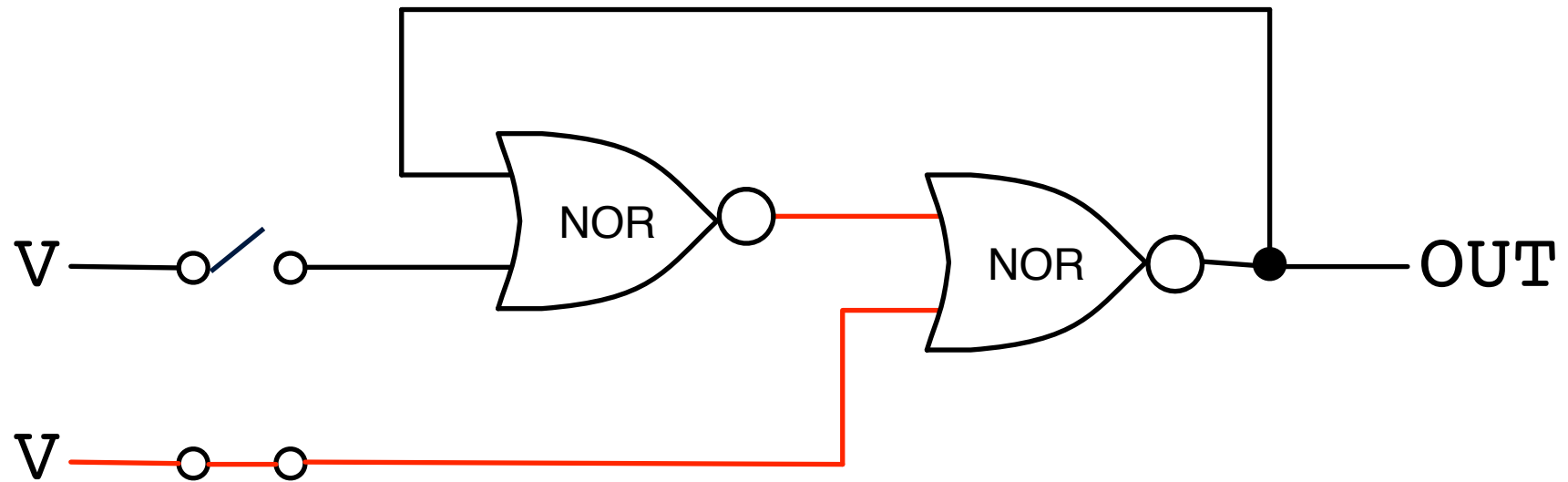
# Opening Upper Key



Same key configuration as initially

**But:** Now OUT is on --- we **remembered** the key turn

# Closing Lower Key



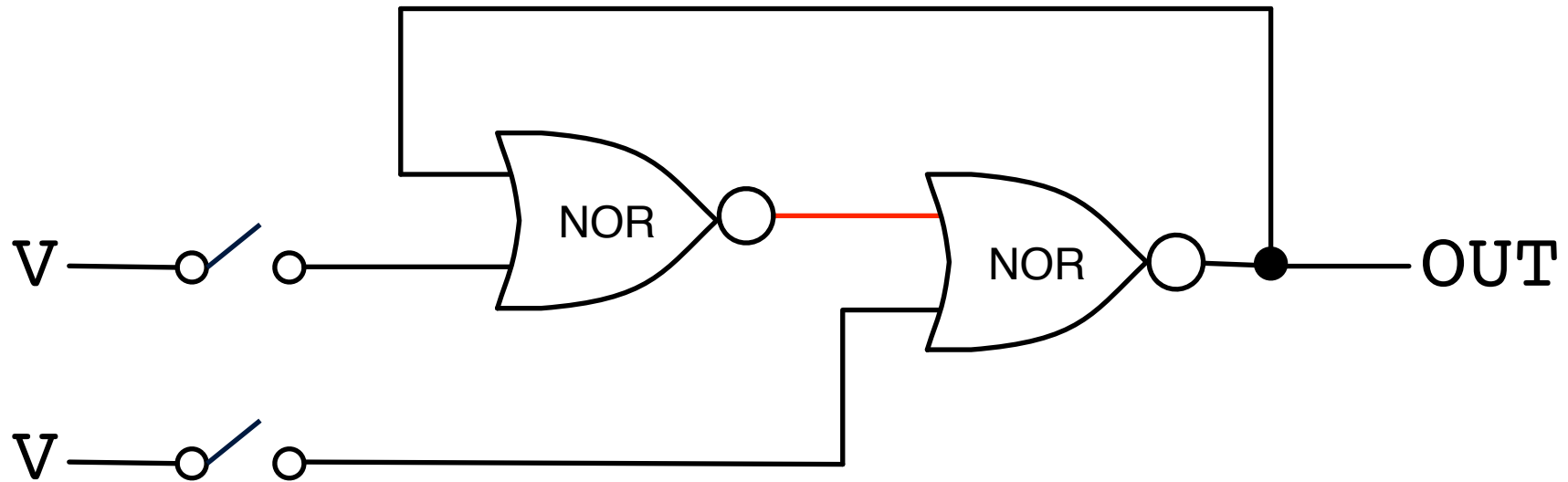


# Opening Lower Key

Sequential  
logic  
(state)

2 states  
1 bit!

out = 1  
out = 0



Back to initial state

# Memory

- We have memory -- called **Reset-Set Flip-Flop**

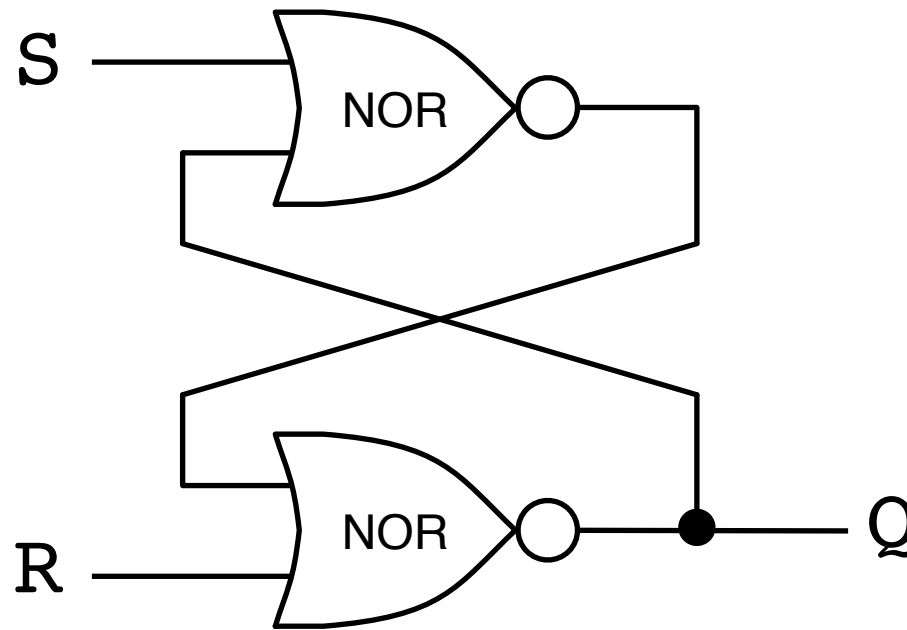
- Truth table

UPPER	LOWER	OUT
0	0	OUT
0	1	0
1	0	1
1	1	Illegal

- UPPER = SET

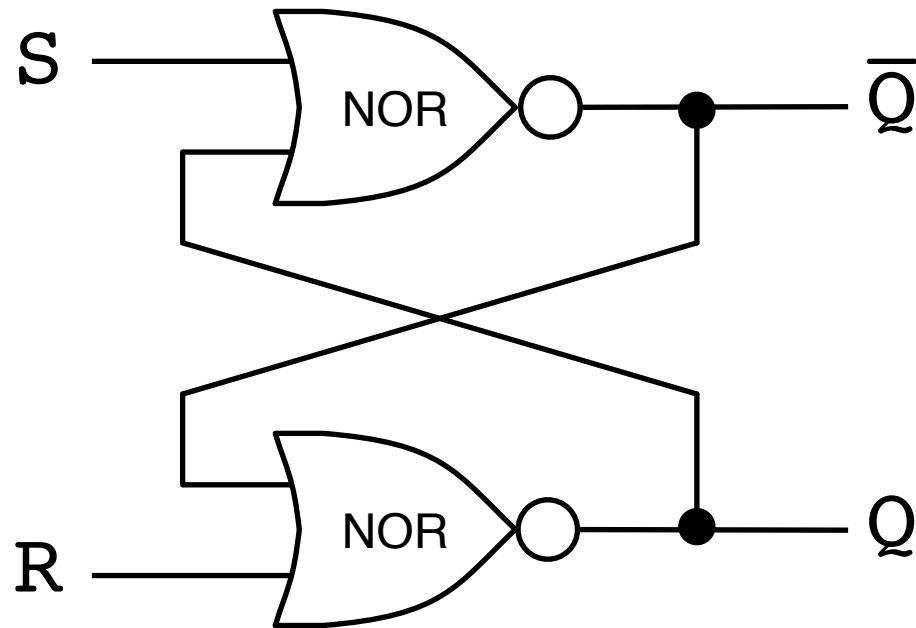
- LOWER = RESET

# Re-Arranged



# Symmetric

SR latch



# Truth Table

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

— stay in current state

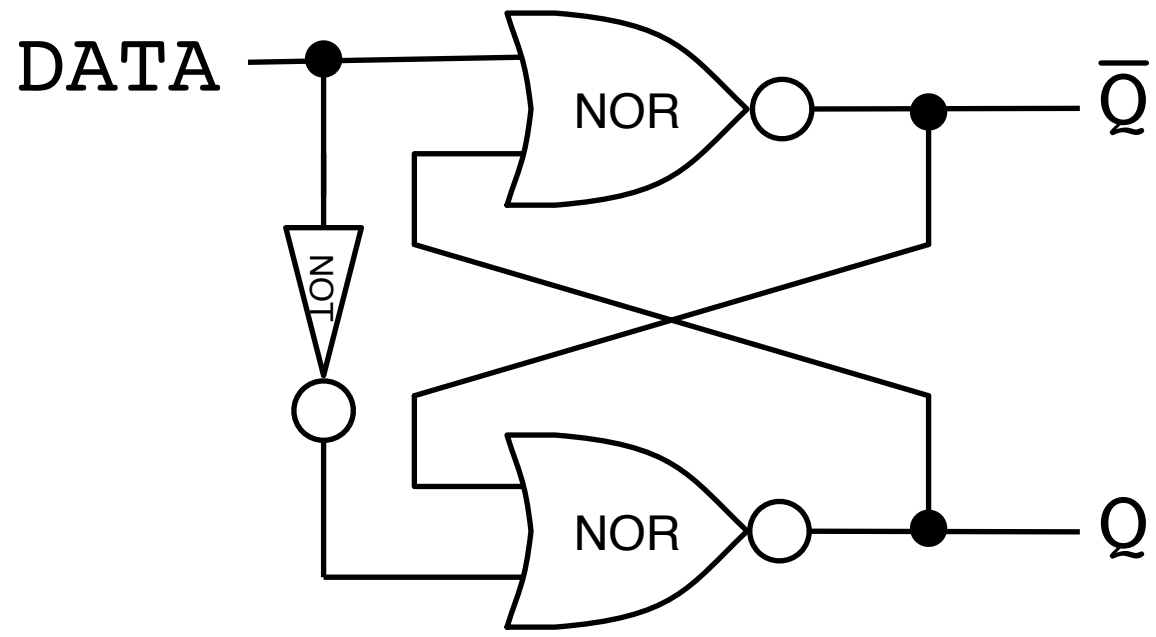




# d-type flip flop

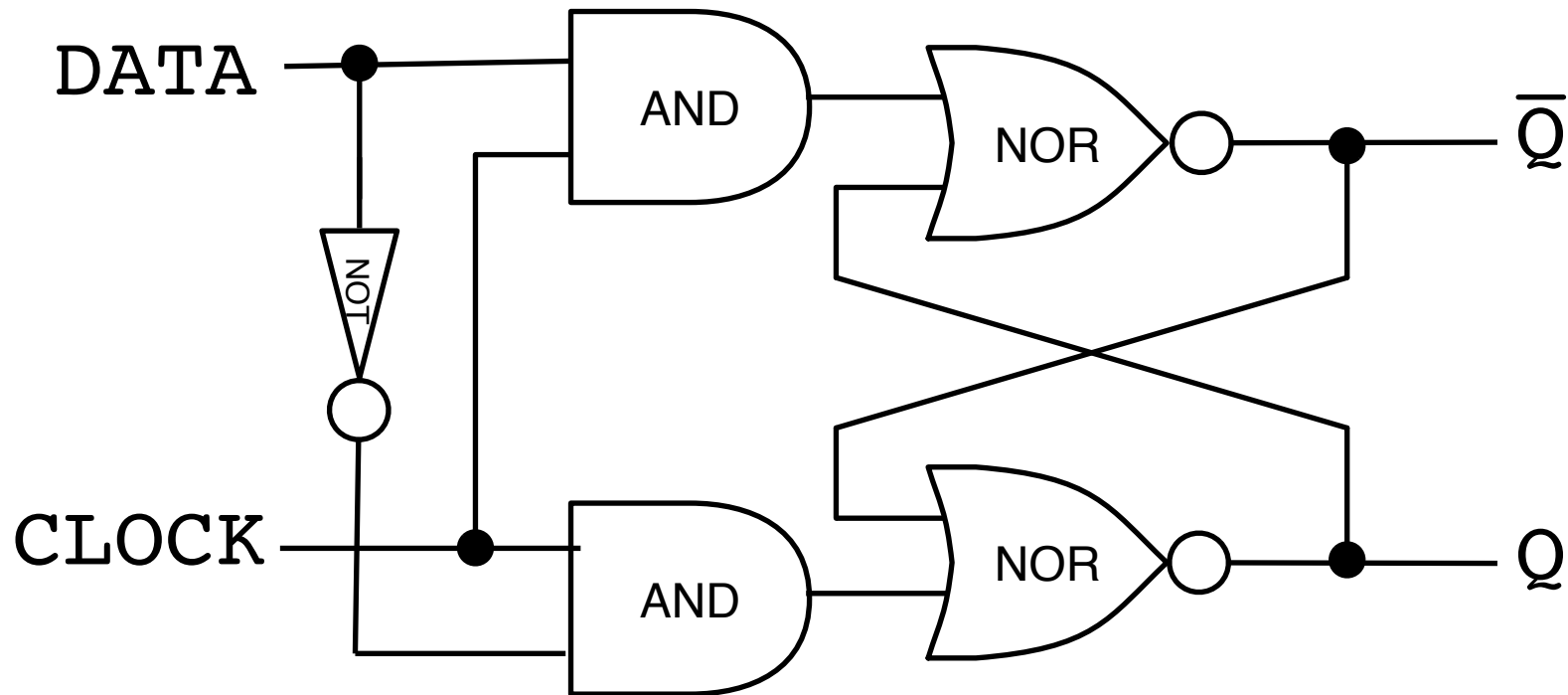
- Control bit ("clock")
  - on = write to memory
  - off = read from memory
- Data bit
  - data item to be written
- Output
  - current state of the memory

# Replace Set/Reset with Data



# Add Control Bit ("Clock")

D-type transparent latch



# D-Type Flip-Flop

- Also called D-type latch
- Circuit latches on one bit of memory and keeps it around
- Truth table

Data	Clock	Q	$\bar{Q}$
0	1	0	1
1	1	1	0
<i>"don't care"</i> X	0	Q	$\bar{Q}$

- Can also build these for multiple data bits



# accumulative adder

# Design Goal



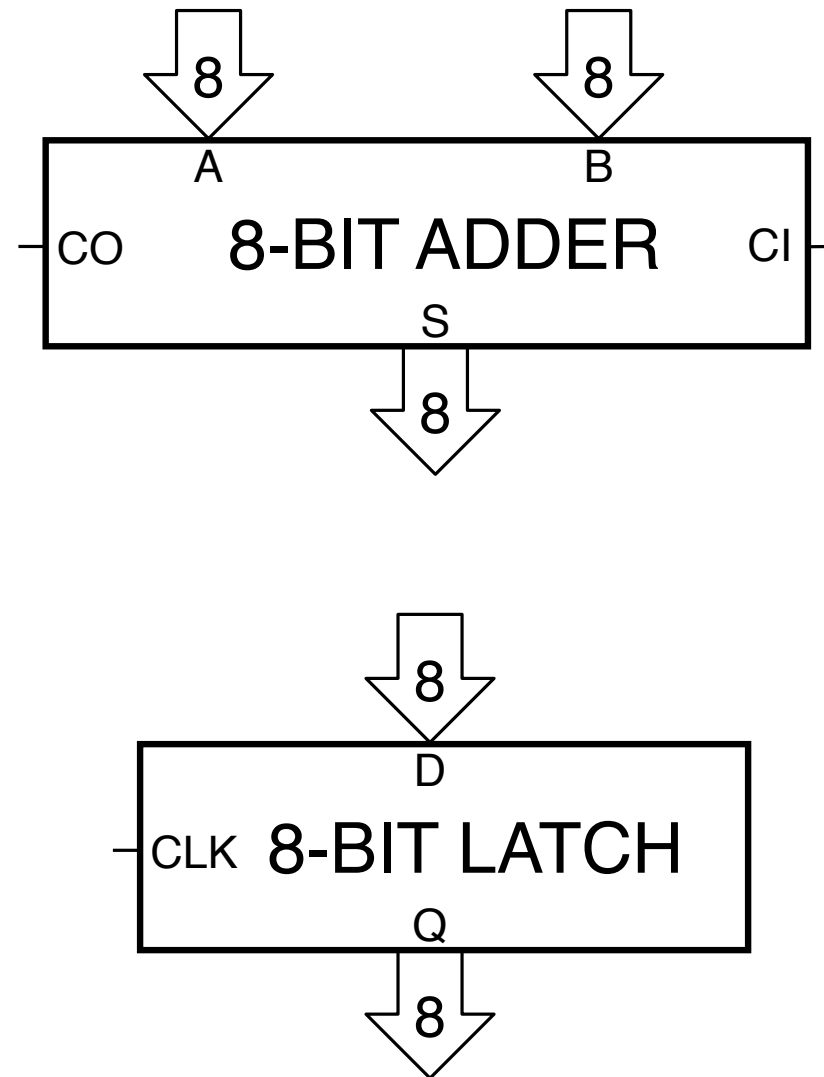
- Adder has initially value 0
- Adding a number  
→ value increases
- Resetting  
→ value goes back to 0

# Ingredients



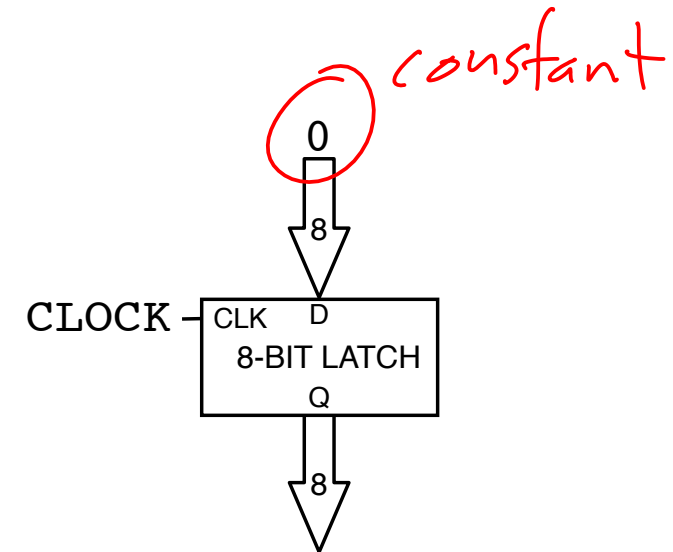


# Ingredients



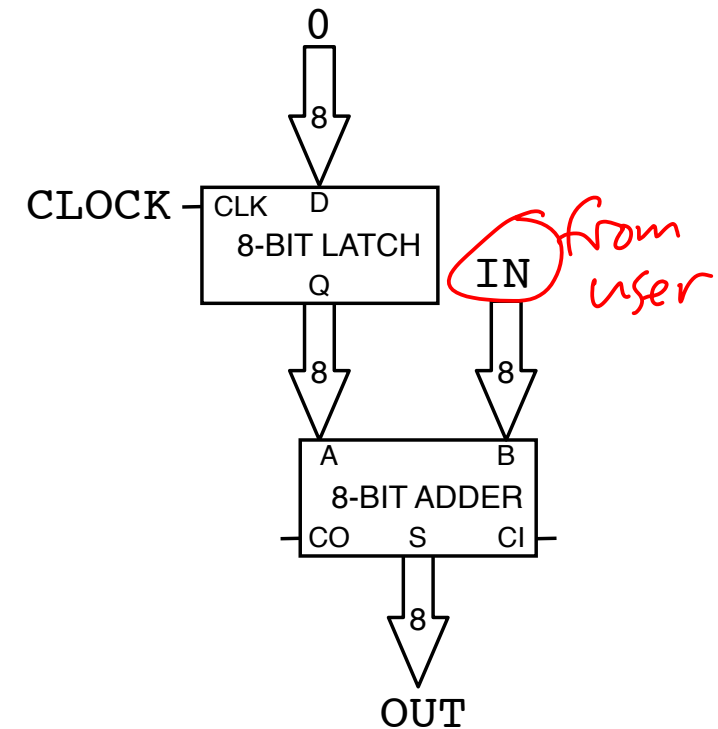
# Building an Accumulative Adder

- Latch: current sum
- Clock on  $\rightarrow$  set it to 0



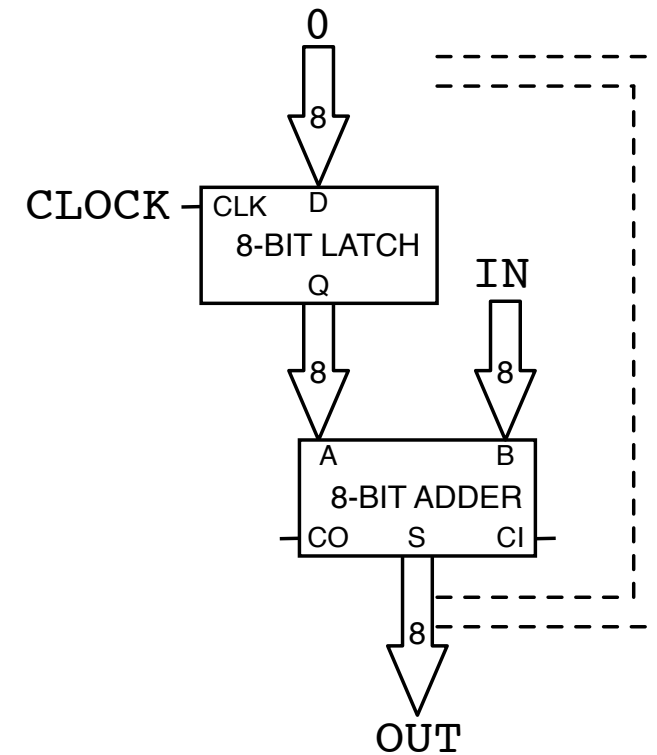
# Building an Accumulative Adder

- Adder
- Combines
  - current value
  - selected input



# Building an Accumulative Adder

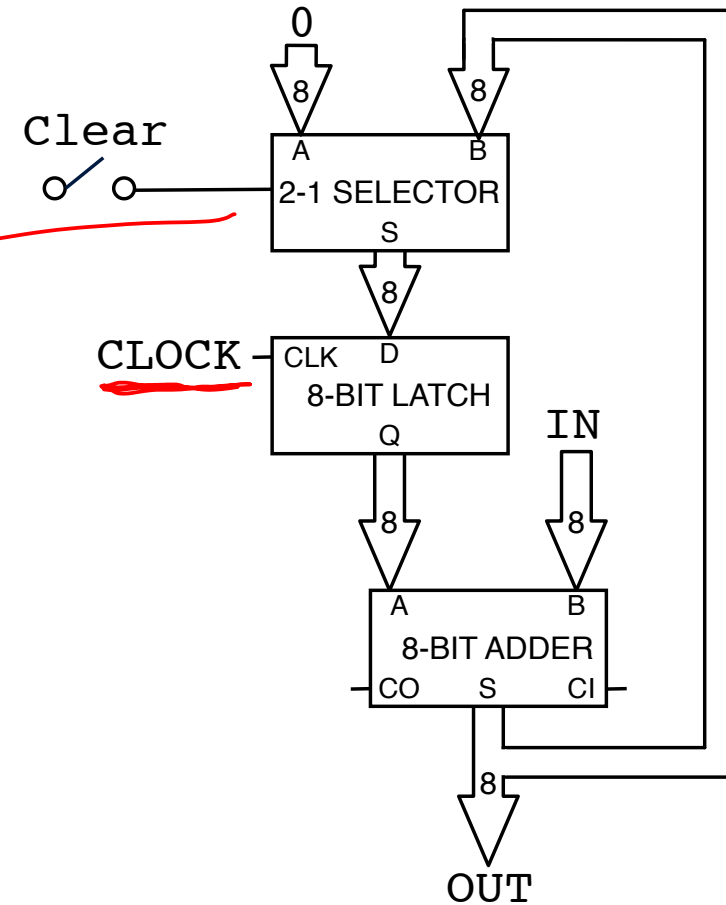
- Can we pass output directly to latch?
- Concerns
  - select between 0 and sum
  - only stored when clock on



# Building an Accumulative Adder

- 2-1 selector
- Either uses 0 or sum
- Built with AND gates
  
- Still have runaway feedback loop...

*aka "multiplexor"  
or  
"mux"*

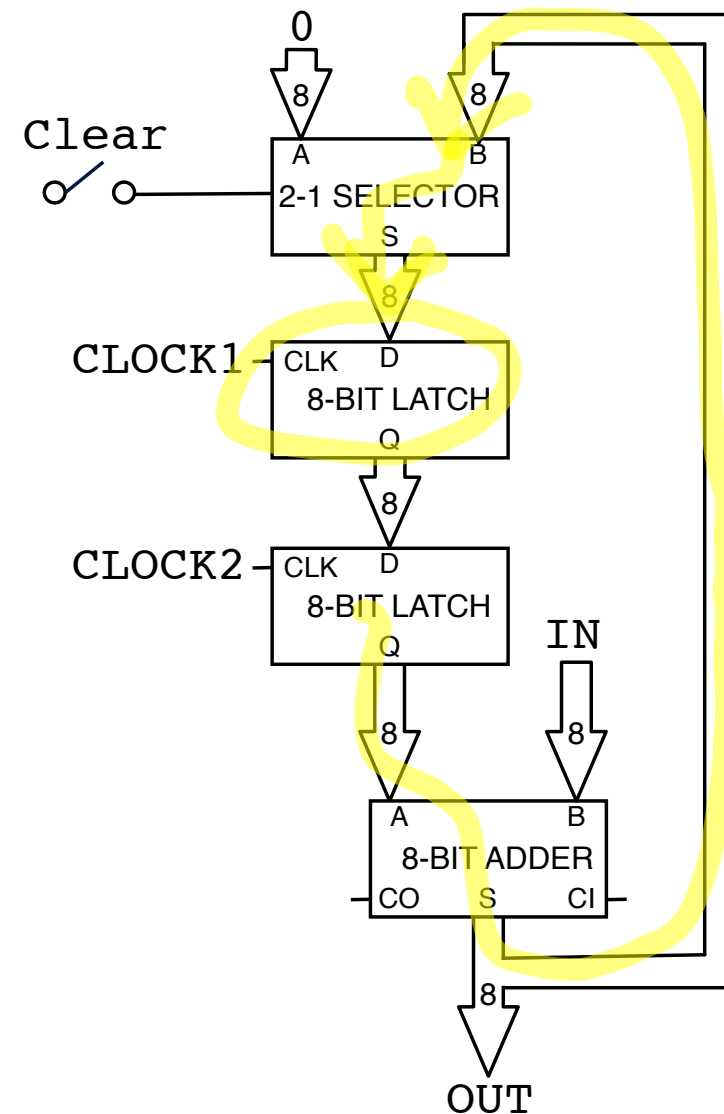


# Building an Accumulative Adder

31

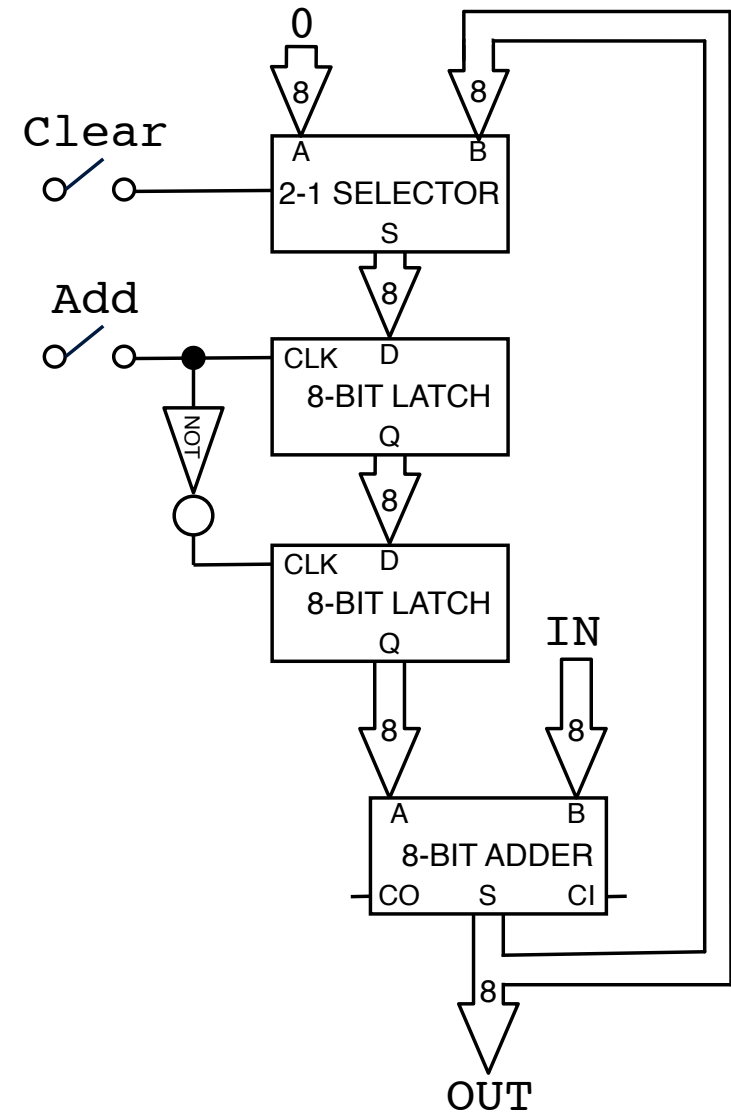


- Two Latches
  - one to store the sum
  - one to store input to adder
- Clock 1
  - carry out addition
  - store result
- Clock 2
  - transfer to set up next addition



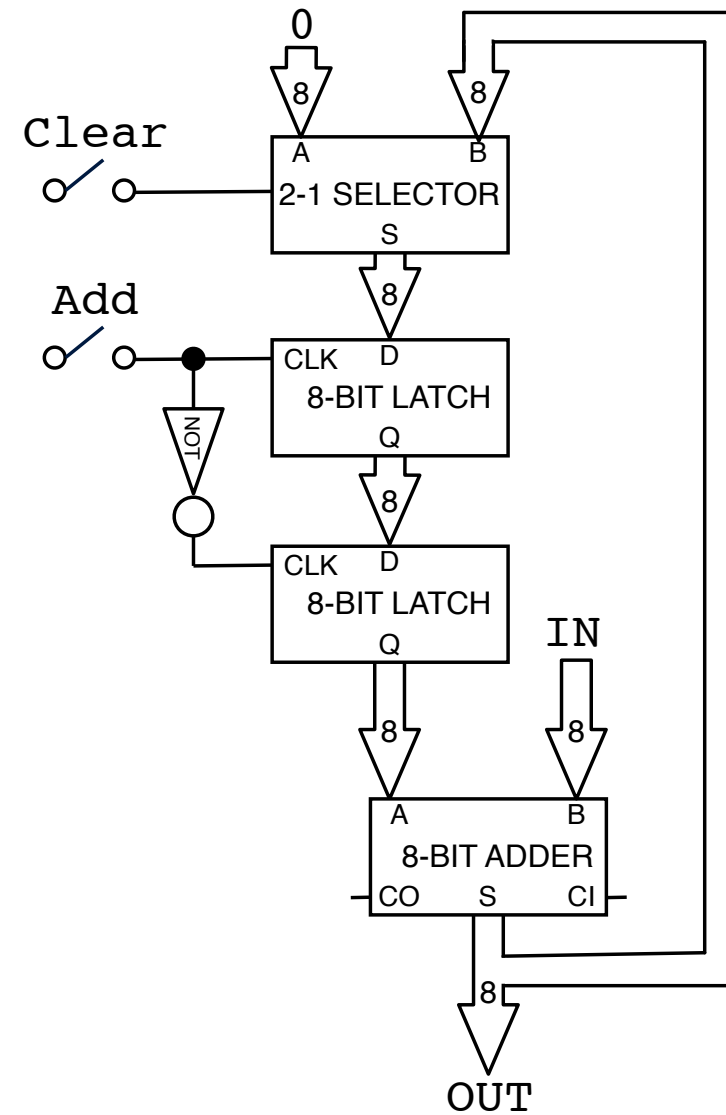
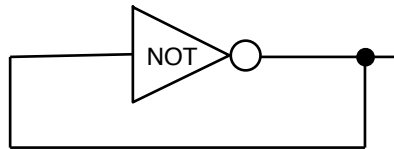
# Building an Accumulative Adder

- Combine the clocks
- Pressing the add key
  - carry out addition
  - store result in upper latch
- Release the add key
  - transfer to lower latch
  - set up next addition



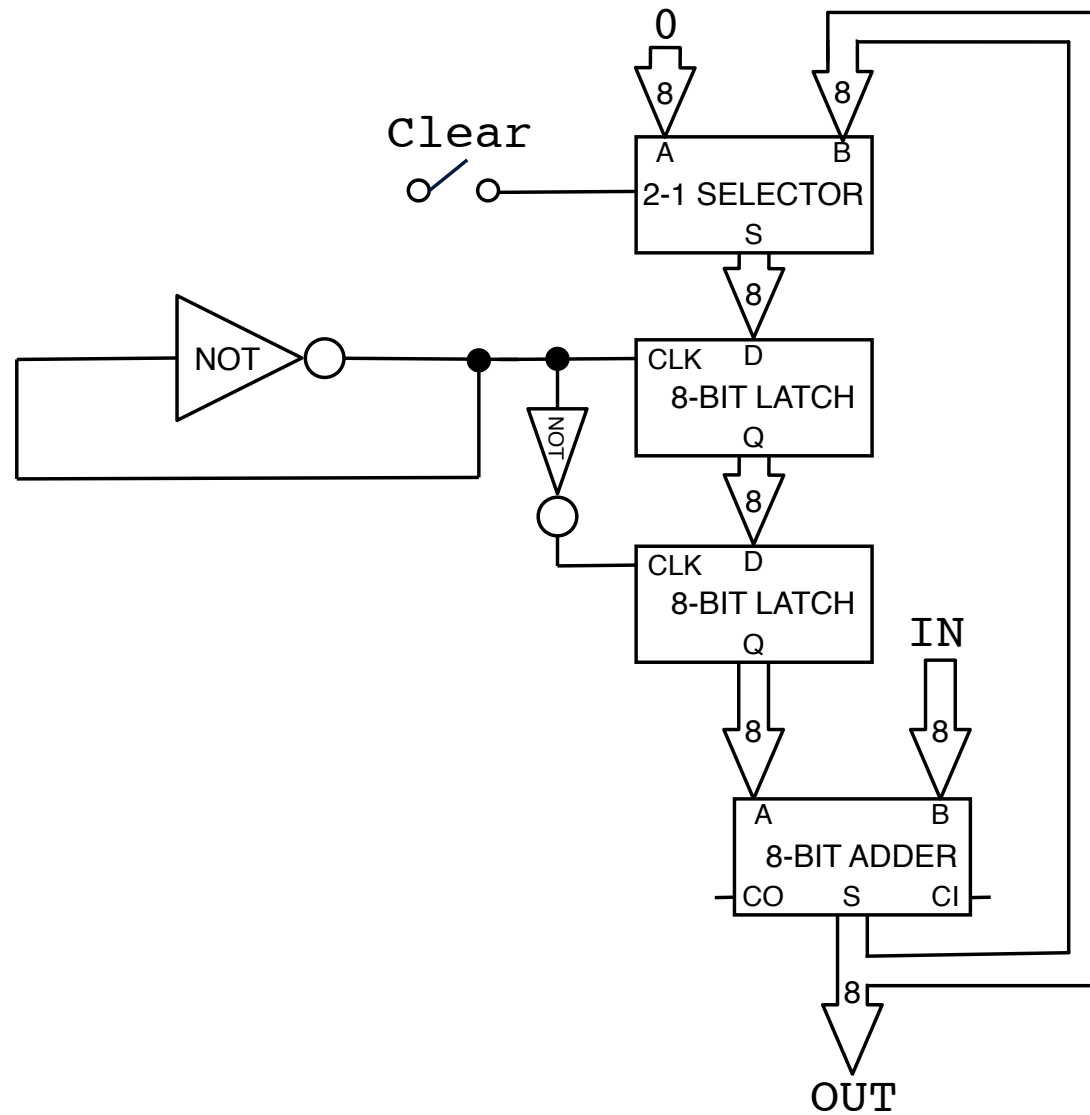
# What Else?

- Remember the oscillator?



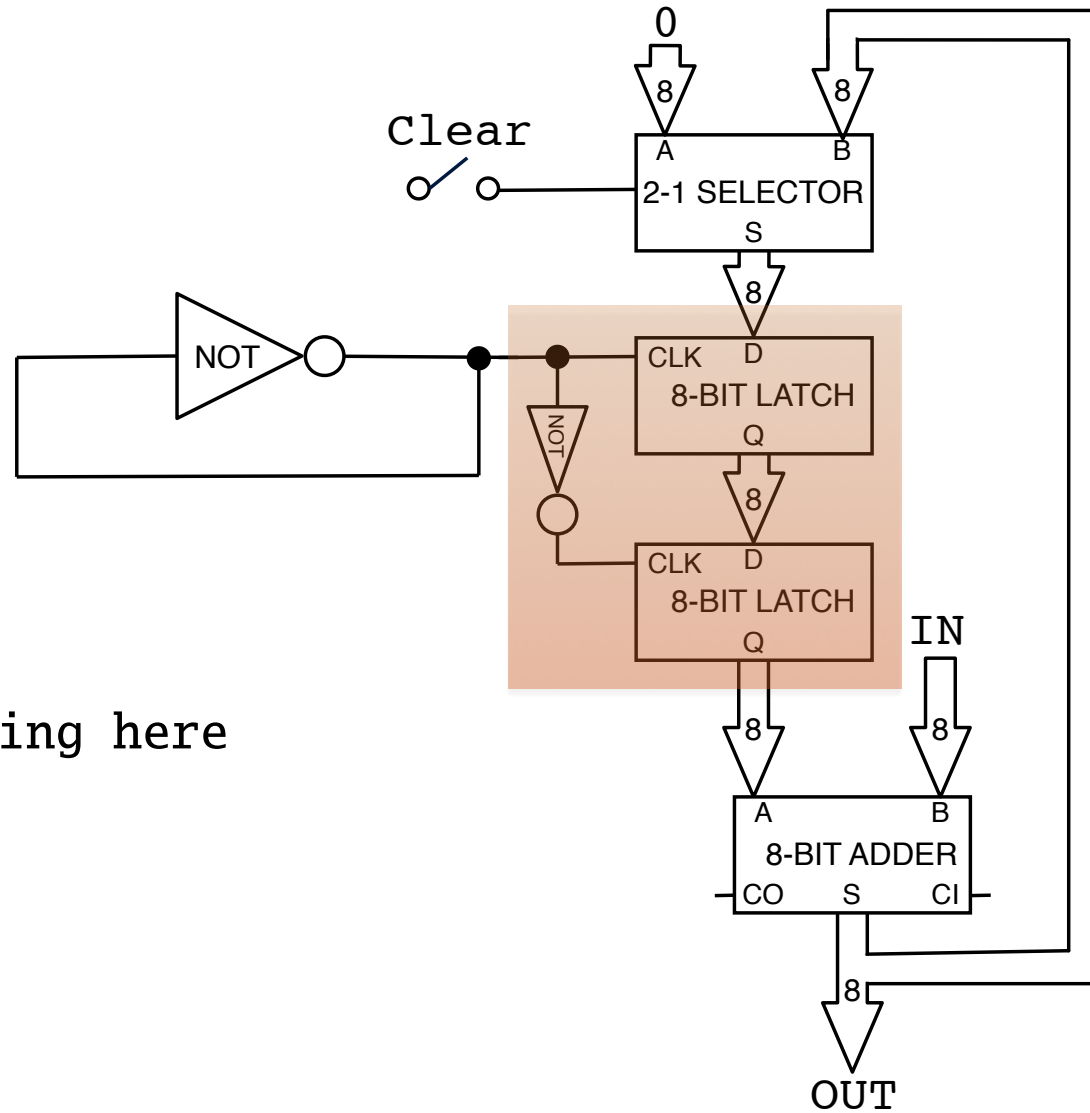


# What Else?



- Each cycle of oscillator:  
keeps adding

# What Else?



- We have something interesting here



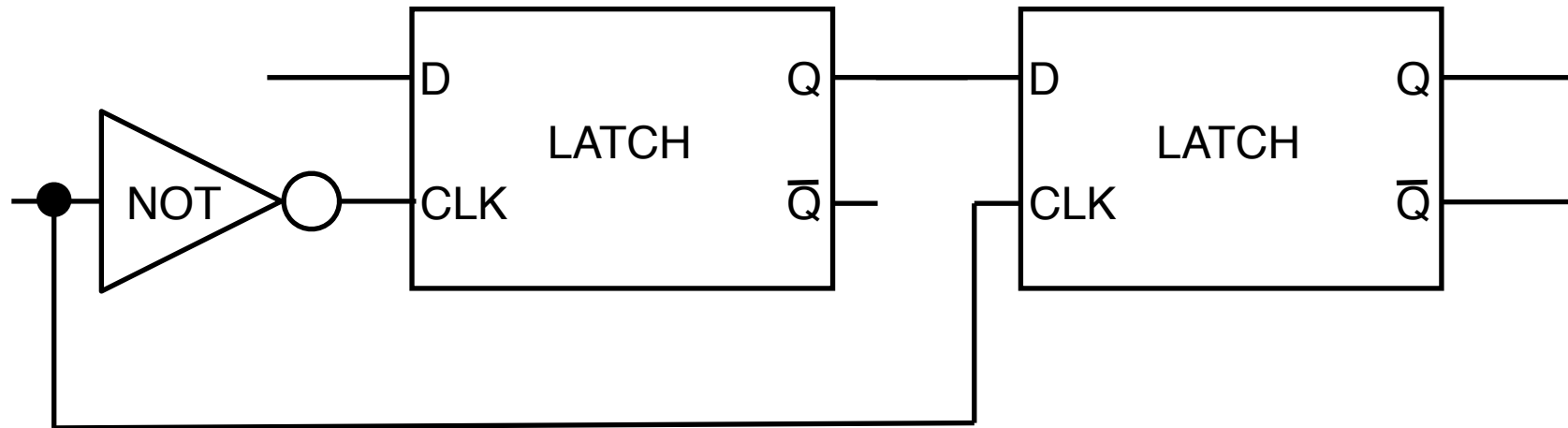
# edge triggered flip-flop

# D-Type Latch



- When clock is on, save data
- "Level-triggered"

# D-Type Latch



- "Edge-triggered": changes value, when *switched from 0 to 1*

# Edge Triggered D-Type Latch



Symbol

# Truth Table

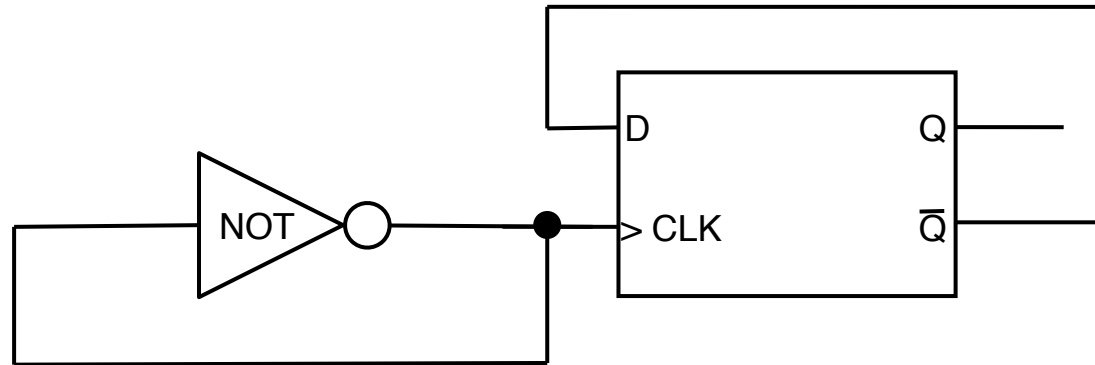
Data	Clock	Q	$\bar{Q}$
0	↑	0	1
1	↑	1	0
X	0	Q	$\bar{Q}$



# ripple counter

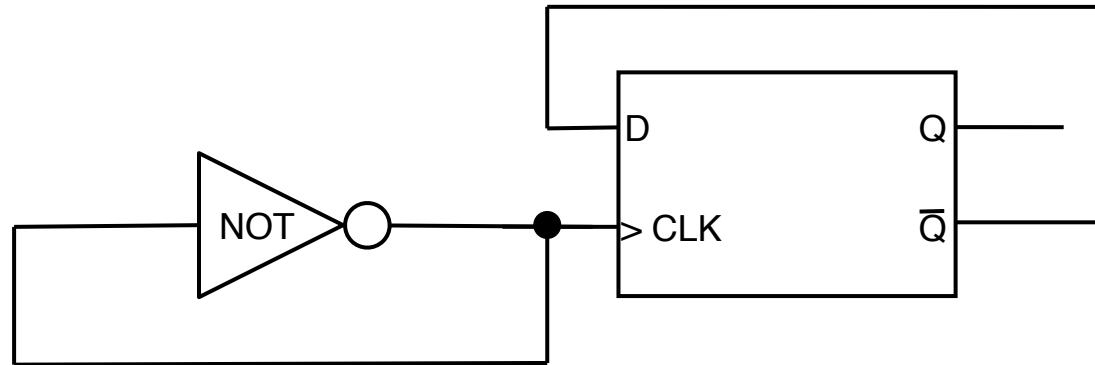


# Oscillator and Latch



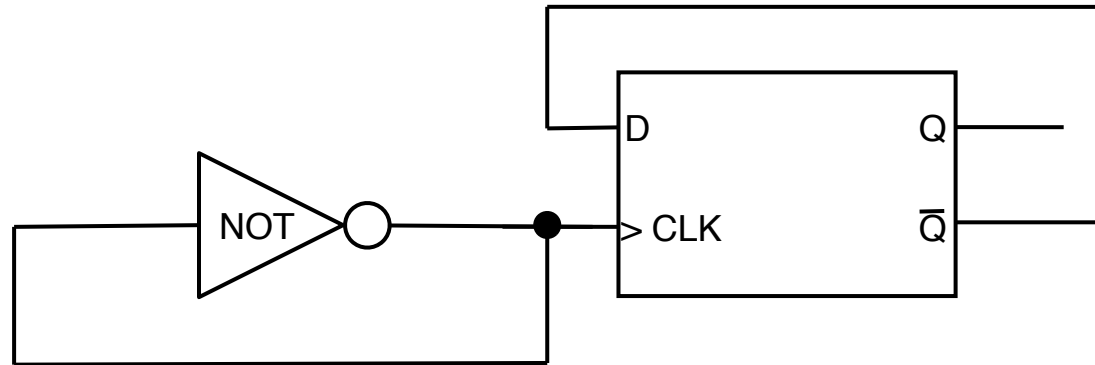
Data	Clock	Q	$\bar{Q}$
1	0	0	1

# Oscillator and Latch



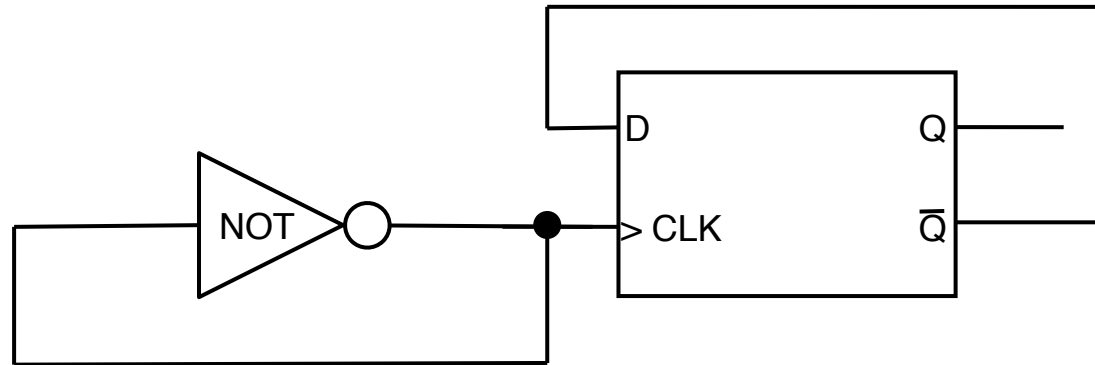
Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0

# Oscillator and Latch



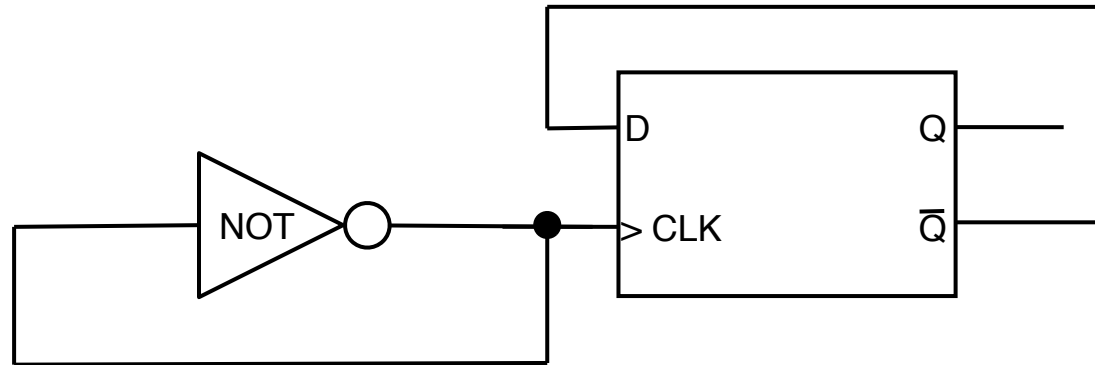
Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0

# Oscillator and Latch



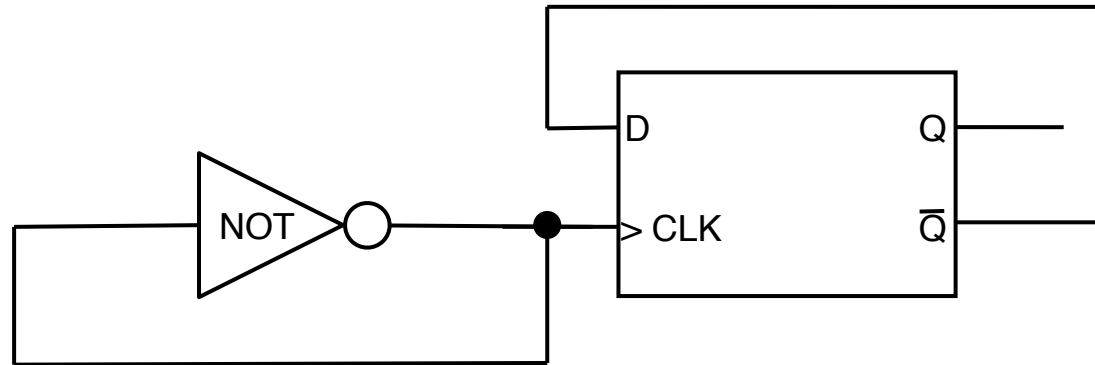
Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0

# Oscillator and Latch



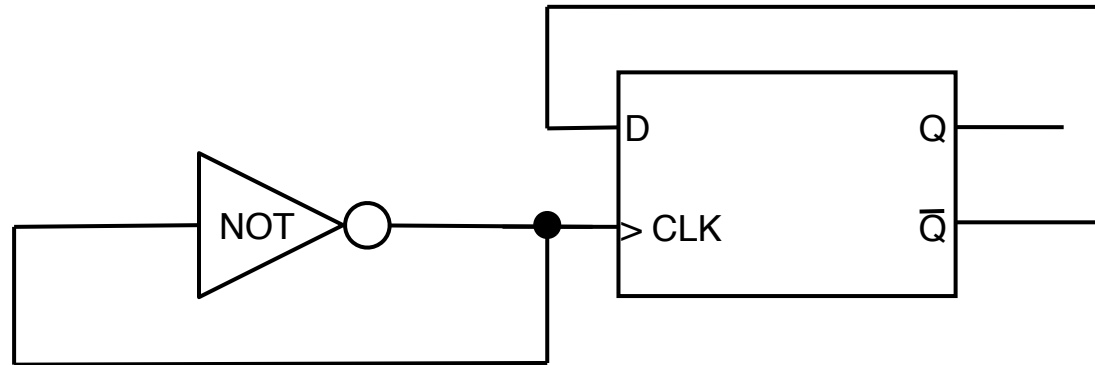
Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0
0	↑	0	1

# Oscillator and Latch



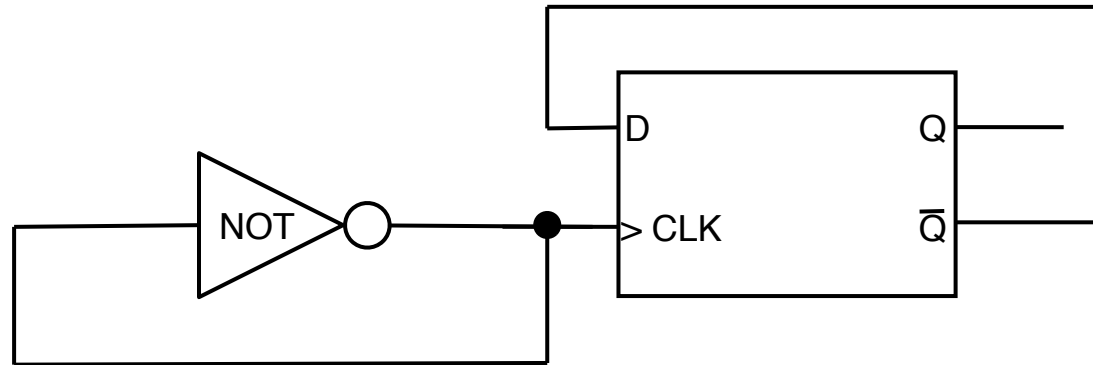
Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0
0	↑	0	1
1	1	0	1

# Oscillator and Latch



Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0
0	↑	0	1
1	1	0	1
1	0	0	1

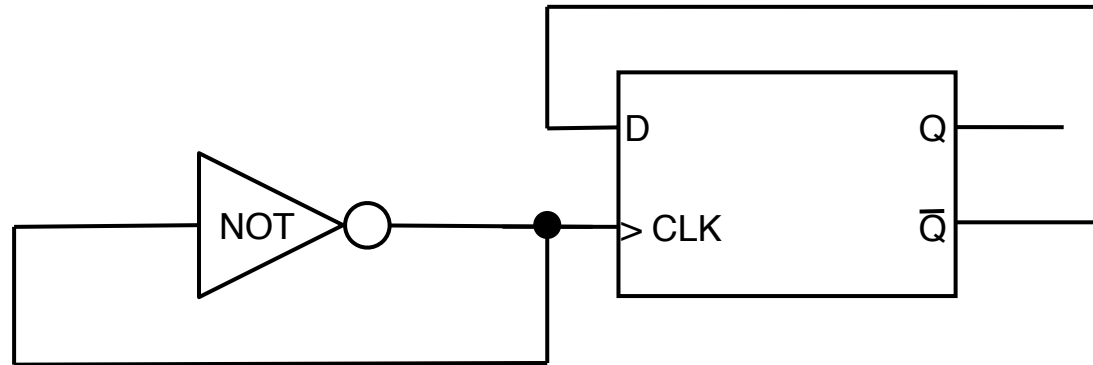
# Oscillator and Latch



Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0
0	↑	0	1
1	1	0	1
1	0	0	1



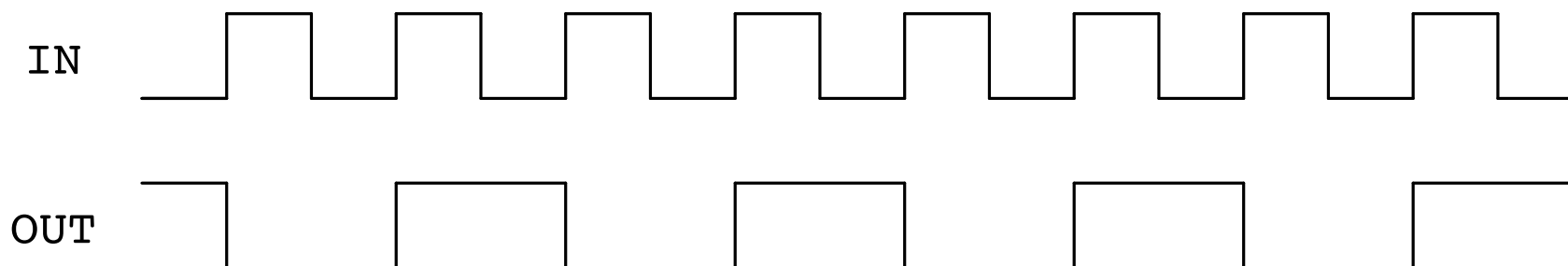
# Oscillator and Latch



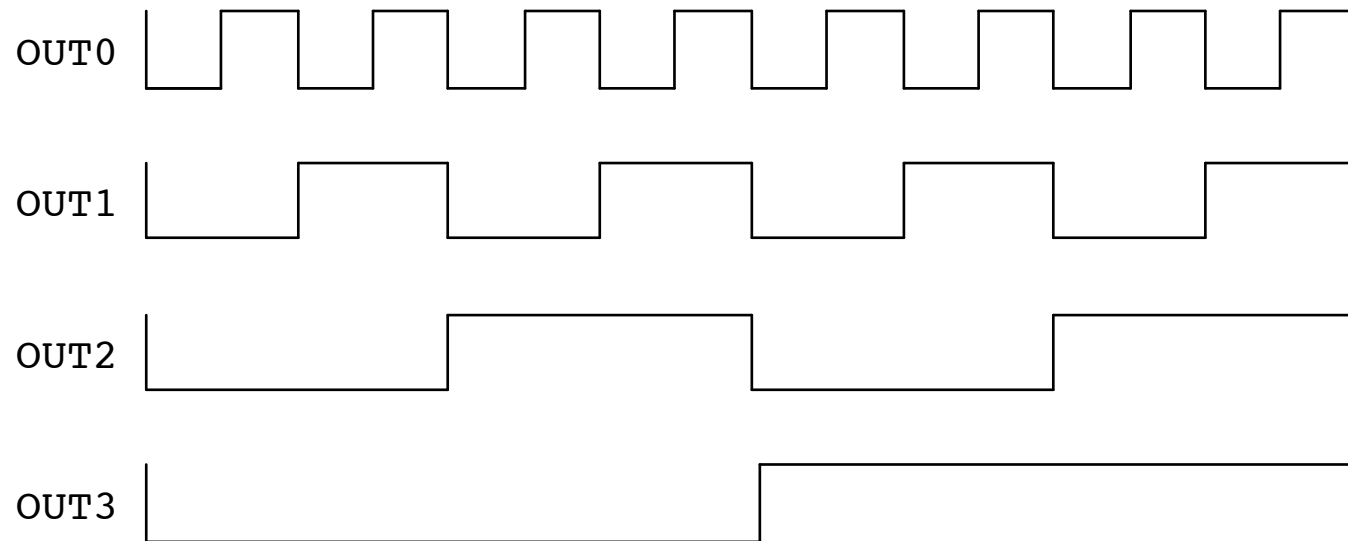
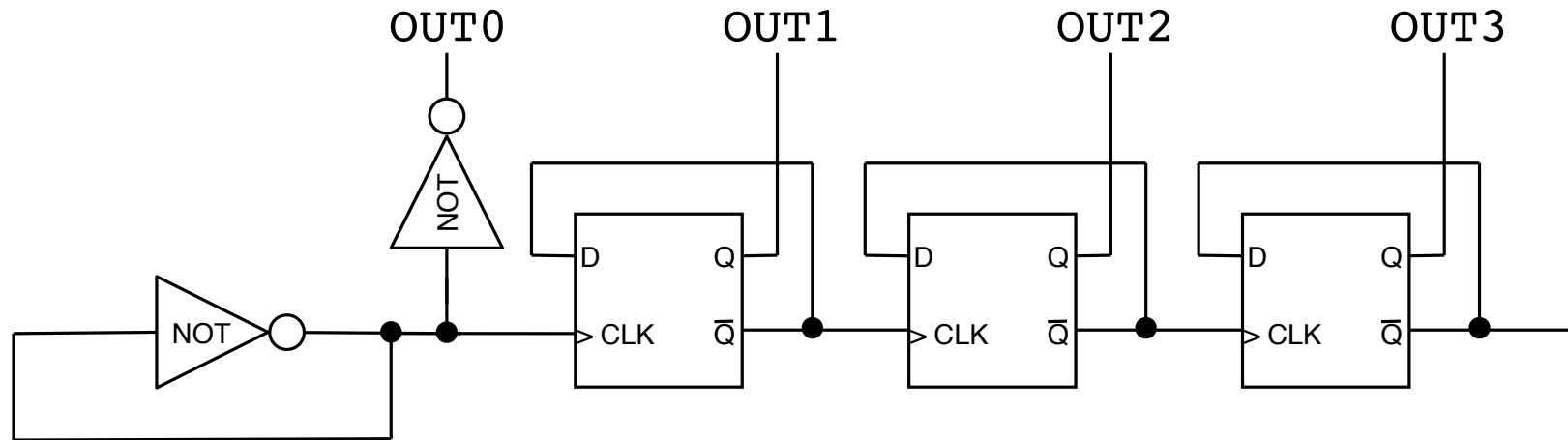
Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0
0	↑	0	1
1	1	0	1
1	0	0	1

# Halving of Frequency

Data	Clock	Q	$\bar{Q}$
1	0	0	1
1	↑	1	0
0	1	1	0
0	0	1	0
0	↑	0	1
1	1	0	1
1	0	0	1



# Multiple Bits



# Ripple Counter

