# Combinational Logic

Multiple levels of representation:

- Logic equations
- Truth tables
- Gate diagrams
- Switching circuits

Boolean algebra: tool to manipulate logic equations

An algebra on a set of two elements: $\{0, 1\}$

Operations: AND, OR, complement

# Boolean Algebra

**Identities:**

$$0a = 0 \qquad 1a = a \qquad aa = a \qquad a\overline{a} = 0$$
$$0 + a = a \qquad 1 + a = 1 \qquad a + a = a \qquad a + \overline{a} = 1$$

$$ab = ba \qquad a(bc) = (ab)c$$
$$a + b = b + a \qquad a + (b + c) = (a + b) + c$$

$$a(b + c) = ab + ac \qquad a + (bc) = (a + b)(a + c)$$

$$\overline{(a + b)} = \overline{a}\,\overline{b} \qquad \overline{(ab)} = \overline{a} + \overline{b}$$

**Precedence: AND takes precedence over OR.**

# Proving Logic Equations

**Example:** $(a+b)(a+c) = a + bc$

**Algebraic proof?**

**Proof with Truth Tables:**

| $a$ | $b$ | $c$ | $a+b$ | $a+c$ | $LHS$ | $bc$ | $RHS$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Truth Tables To Logic Equations

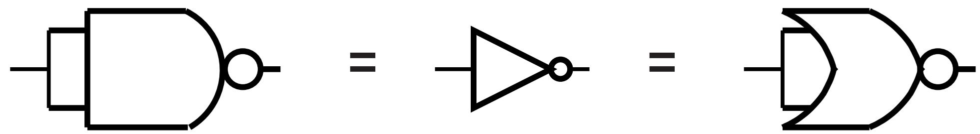
| $a$ | $b$ | $c$ | $out$ | $Minterms$ | $Maxterms$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $\overline{a}\overline{b}\overline{c}$ | $a+b+c$ |
| 0 | 0 | 1 | 1 | $\overline{a}\overline{b}c$ | $a+b+\overline{c}$ |
| 0 | 1 | 0 | 1 | $\overline{a}b\overline{c}$ | $a+\overline{b}+c$ |
| 0 | 1 | 1 | 0 | $\overline{a}bc$ | $a+\overline{b}+\overline{c}$ |
| 1 | 0 | 0 | 1 | $a\overline{b}\overline{c}$ | $\overline{a}+b+c$ |
| 1 | 0 | 1 | 1 | $a\overline{b}c$ | $\overline{a}+b+\overline{c}$ |
| 1 | 1 | 0 | 0 | $ab\overline{c}$ | $\overline{a}+\overline{b}+c$ |
| 1 | 1 | 1 | 0 | $abc$ | $\overline{a}+\overline{b}+\overline{c}$ |

Sum of Products: $\overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\overline{c} + a\overline{b}c$

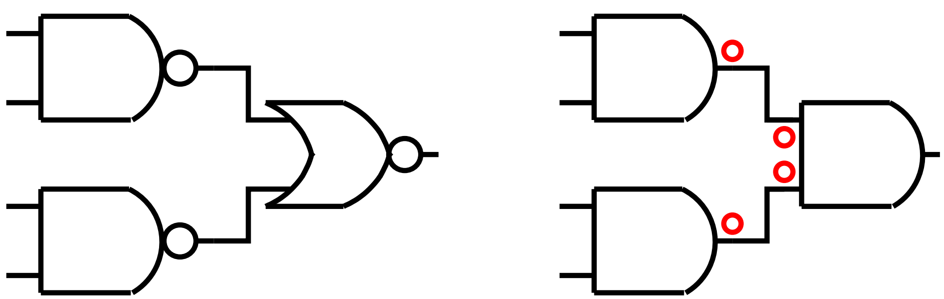
Product of Sums:

$$(a+b+c)(a+\overline{b}+\overline{c})(\overline{a}+\overline{b}+c)(\overline{a}+\overline{b}+\overline{c})$$

# Universality: NAND and NOR



Universal: can implement any combinational function using just NAND or just NOR gates.



# Minimizing Logic Equations

Earlier example:

$$\overline{a}\overline{b}c + \overline{a}b\overline{c} + \underbrace{a\overline{b}\overline{c} + a\overline{b}c}_{a\overline{b}(c+\overline{c}) = a\overline{b}}$$

One can use Boolean algebra to simplify equations.

Systematic techniques:

- Karnaugh maps
- Quine-McCluskey

(details in section next week)

## Word Problems

"Increment input by 1, compute result mod 5"

Representation: 3-bit binary input

| | $I_2$ | $I_1$ | $I_0$ | $O_2$ | $O_1$ | $O_0$ |
|---|---|---|---|---|---|---|
| $Input = 0$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $Input = 1$ | 0 | 0 | 1 | 0 | 1 | 0 |
| $Input = 2$ | 0 | 1 | 0 | 0 | 1 | 1 |
| $Input = 3$ | 0 | 1 | 1 | 1 | 0 | 0 |
| $Input = 4$ | 1 | 0 | 0 | 0 | 0 | 0 |
| $Input = 5$ | 1 | 0 | 1 | 0 | 0 | 1 |
| $Input = 6$ | 1 | 1 | 0 | 0 | 1 | 0 |
| $Input = 7$ | 1 | 1 | 1 | 0 | 1 | 1 |

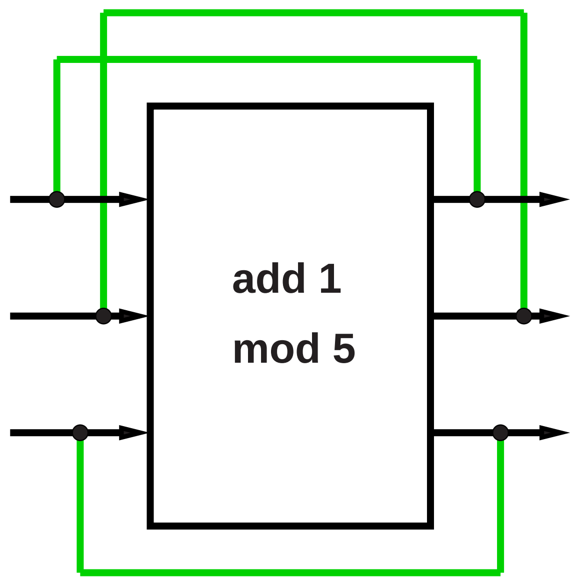
# Don't Cares

Given: the input is always between 0 and 4:

| | $I_2$ | $I_1$ | $I_0$ | $O_2$ | $O_1$ | $O_0$ |
|---|---|---|---|---|---|---|
| $Input = 0$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $Input = 1$ | 0 | 0 | 1 | 0 | 1 | 0 |
| $Input = 2$ | 0 | 1 | 0 | 0 | 1 | 1 |
| $Input = 3$ | 0 | 1 | 1 | 1 | 0 | 0 |
| $Input = 4$ | 1 | 0 | 0 | 0 | 0 | 0 |
| $Input = 5$ | 1 | 0 | 1 | $X$ | $X$ | $X$ |
| $Input = 6$ | 1 | 1 | 0 | $X$ | $X$ | $X$ |
| $Input = 7$ | 1 | 1 | 1 | $X$ | $X$ | $X$ |

Can be used to simplify logic equations.

# What If I Want to Keep Counting?




What happens?

# Sequential Circuits

Need a way to *sequence* operations.

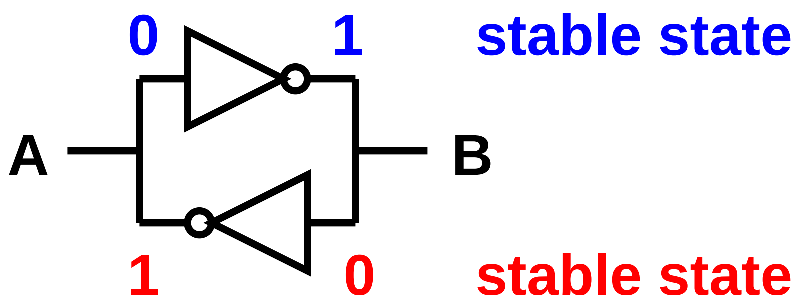
Idea:

- Introduce devices that can hold state
  called state-holding elements
- Read stable inputs from state-holding elements
- Write stable outputs to state-holding elements
- Generate outputs from inputs using combinational logic

# Bi-Stable Devices

## Part I: state-holding devices

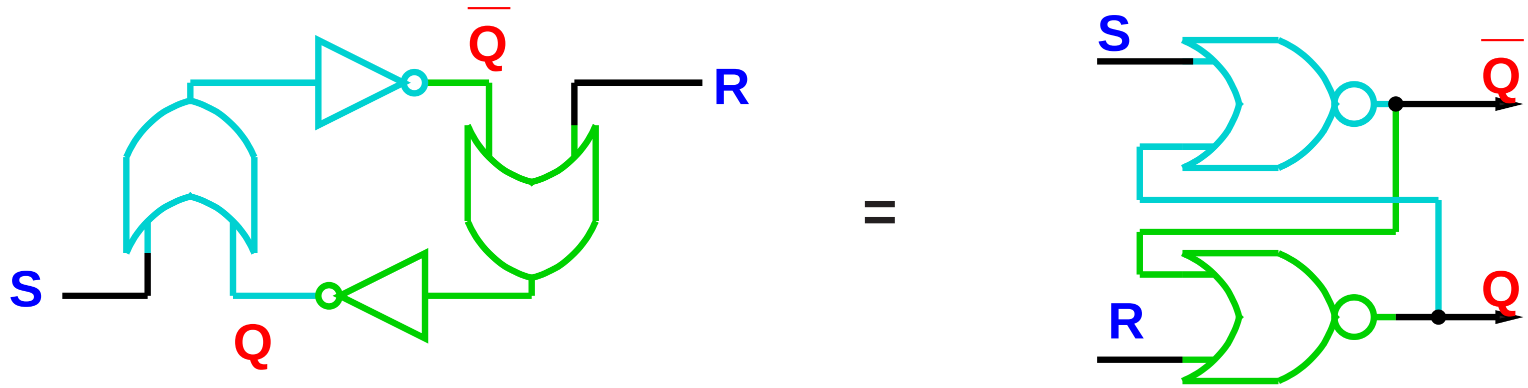
A simple device:




In a stable state, $A = \overline{B}$

- How do we change the state?

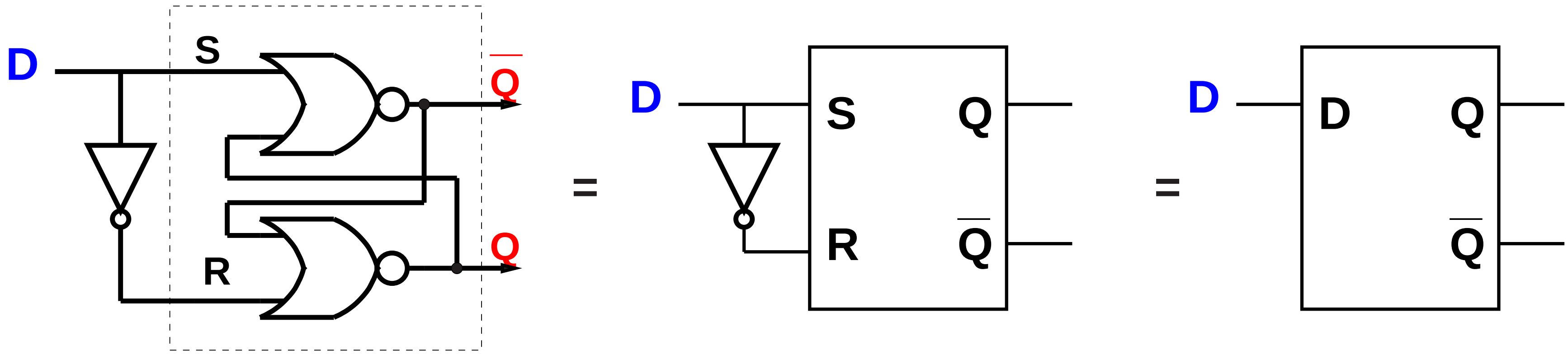
# SR Latch




| $S$ | $R$ | $Q$ | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | $Q$ | $\overline{Q}$ |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | ? | ? |

- SR Latch (set-reset)
- $Q$: stored value
- $\overline{Q}$: complement
- $S = 1$ and $R = 1$?

# D Latch




- When $D$ changes, $Q$ changes...

  ... immediately.

Need to control when the output changes.

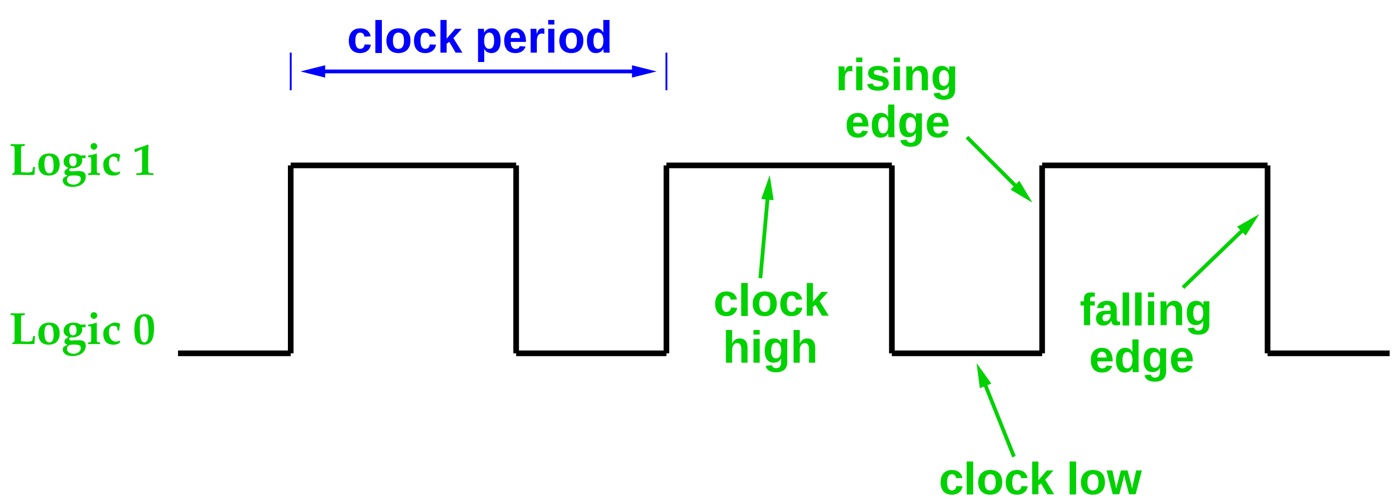
# Clocks

## Part II: modifying state-holding elements

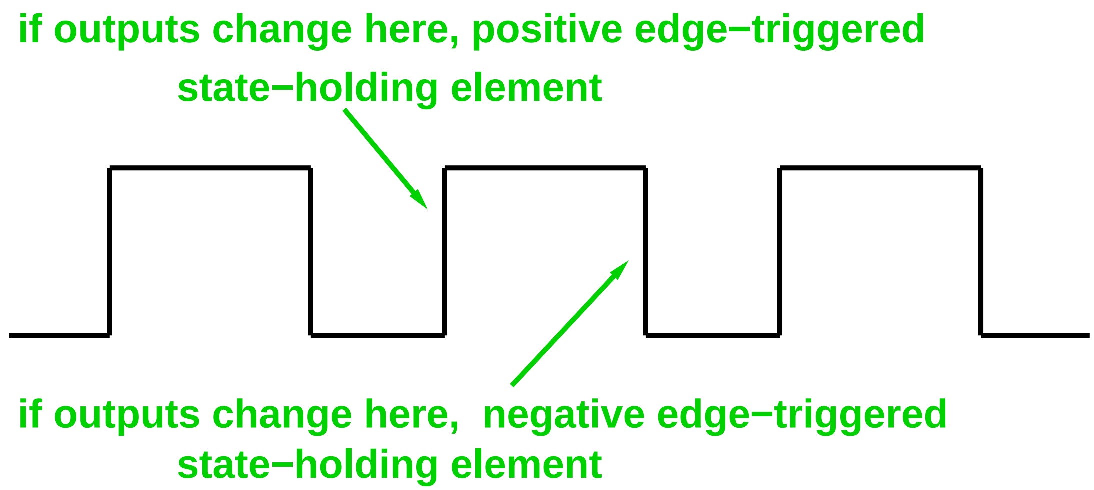
Introduce a free-running signal: the clock

Clock signal has a fixed cycle time (a.k.a. cycle period).

Clock frequency = 1/cycle time



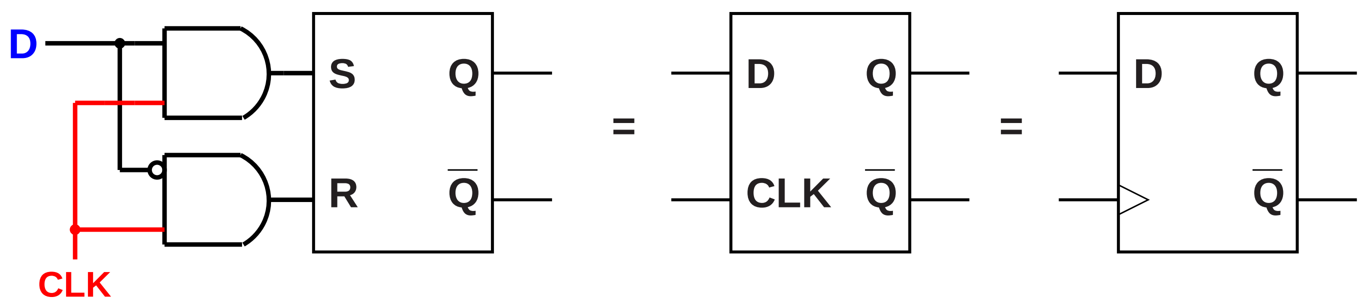
# Edge Triggered Clocking



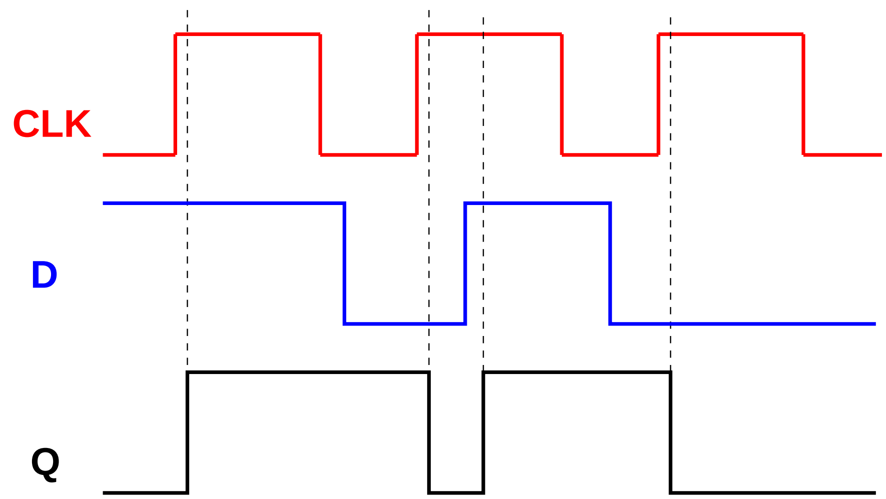

- Inputs must be *stable* just before the clock edge where the outputs change.

Lots of other choices... (EE 438)

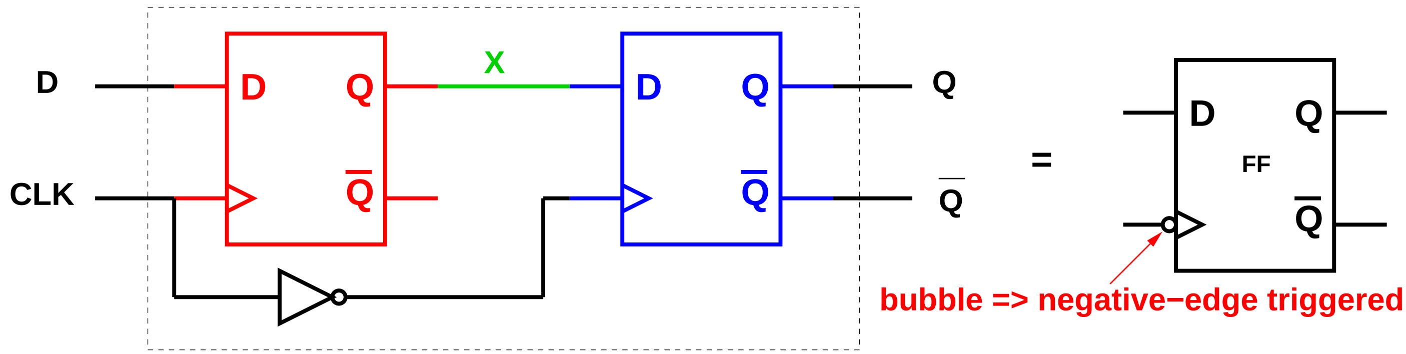
# First Attempt

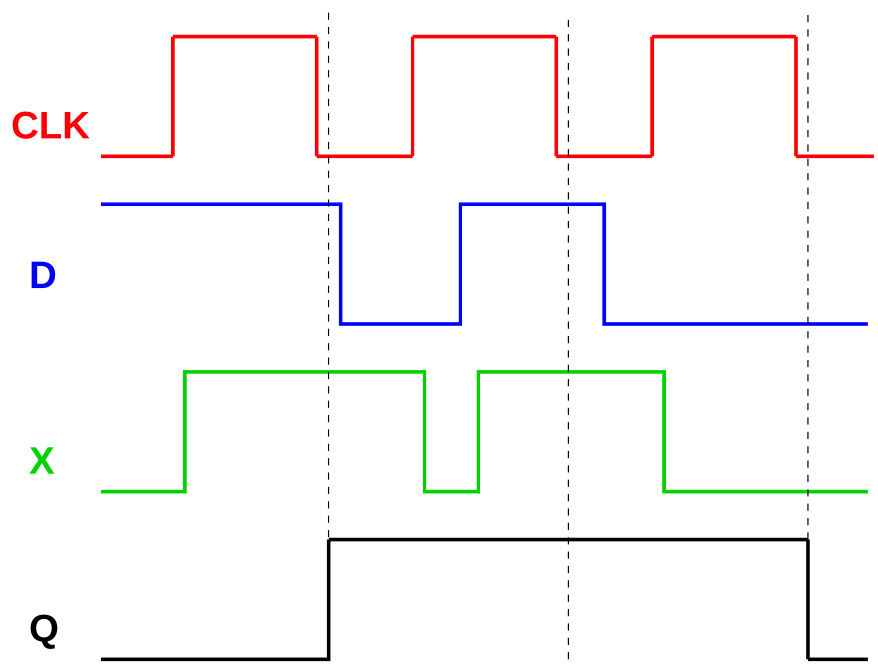



- How does the output behave?



# Master-Slave Flip-Flop



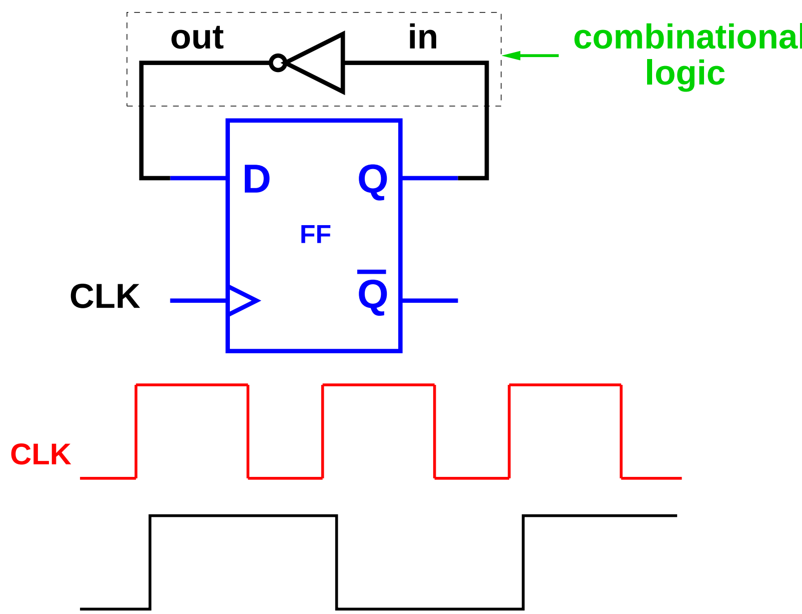



# Example: 1-Bit Counter

## Truth-table:

| $in$ | $out$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Circuit:



# Finite State Machines

Basic Idea: A circuit has

- External inputs
- Externally visible outputs
- Internal state

Output and next state depend on:

- Inputs
- Current State

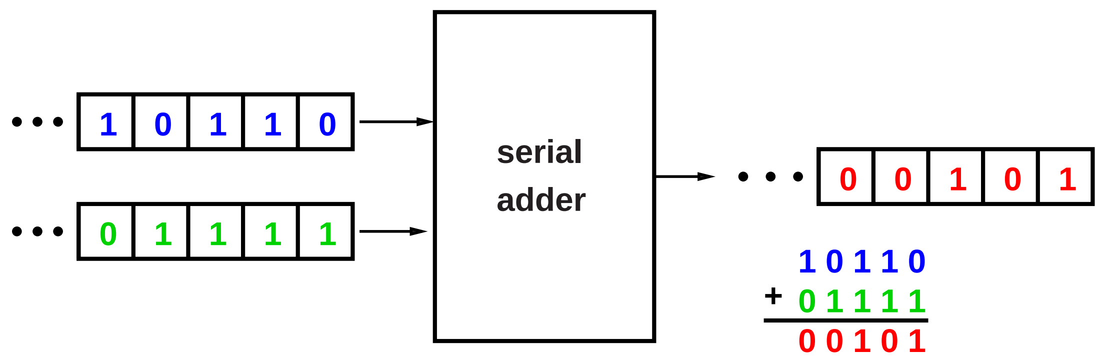
Two types:

- Mealy: output is a function of state and input
- Moore: output is a function of state only

# Designing a Finite-State Machine

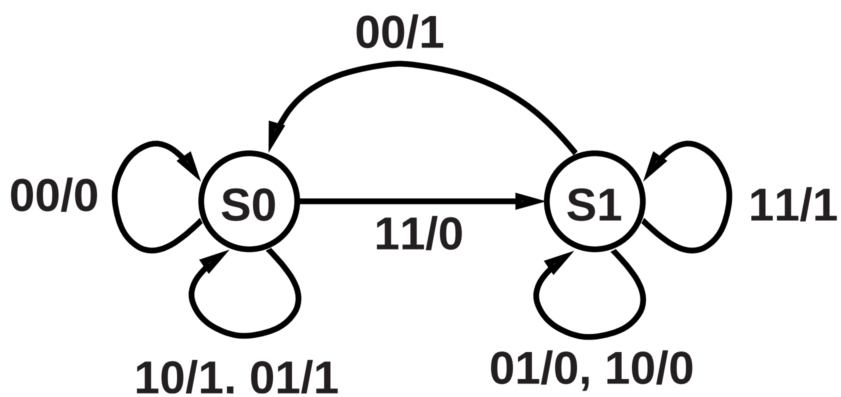
- Draw a state diagram
- Write down state transition table
- State assignment
- Determine logic equations for all flip-flops and outputs

Example: add two input bit-streams (least-significant-bit first).



# The Serial Adder

- Two states: S0 (carry is zero), S1 (carry is 1)
- Inputs: $a$ and $b$
- Output: $z$




Arcs labelled
with input vector
and output
ab/z

# State Table

| $a$ | $b$ | $state$ | $z$ | $next\ state$ |
|---|---|---|---|---|
| 0 | 0 | $S0$ | 0 | $S0$ |
| 0 | 1 | $S0$ | 1 | $S0$ |
| 1 | 0 | $S0$ | 1 | $S0$ |
| 1 | 1 | $S0$ | 0 | $S1$ |
| 0 | 0 | $S1$ | 1 | $S0$ |
| 0 | 1 | $S1$ | 0 | $S1$ |
| 1 | 0 | $S1$ | 0 | $S1$ |
| 1 | 1 | $S1$ | 1 | $S1$ |

For each input combination and state combination, write down output and next state.

# State Assignment

Pick encoding of states. We have two states, so use one bit $s$.

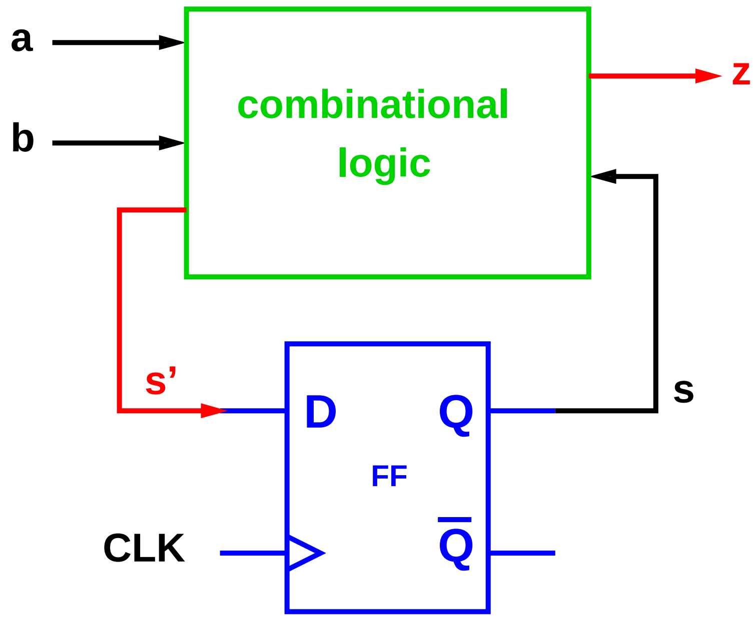
- S0: $s = 0$, S1: $s = 1$

| $a$ | $b$ | $s$ | $z$ | $s'$ | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | $\overline{a}\overline{b}\overline{s}$ |
| 0 | 1 | 0 | 1 | 0 | $\overline{a}b\overline{s}$ |
| 1 | 0 | 0 | 1 | 0 | $a\overline{b}\overline{s}$ |
| 1 | 1 | 0 | 0 | 1 | $ab\overline{s}$ |
| 0 | 0 | 1 | 1 | 0 | $\overline{a}\overline{b}s$ |
| 0 | 1 | 1 | 0 | 1 | $\overline{a}bs$ |
| 1 | 0 | 1 | 0 | 1 | $a\overline{b}s$ |
| 1 | 1 | 1 | 1 | 1 | $abs$ |

# Logic Equations and Circuit

$$z = \overline{a}b\overline{s} + a\overline{b}\overline{s} + \overline{a}\overline{b}s + abs$$

$$s' = ab\overline{s} + \overline{a}bs + a\overline{b}s + abs = ab + bs + as$$




What's the clock period?