# State

**Prof. Hakim Weatherspoon**
**CS 3410**
Computer Science
Cornell University
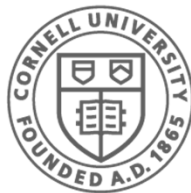
[Weatherspoon, Bala, Bracy, and Sirer]

# Goals for Today

## State

- How do we store **one** bit?
- Attempts at storing (and changing) one bit
  - Set-Reset Latch
  - D Latch
  - D Flip-Flops
  - Master-Slave Flip-Flops
- Register: storing more than one bit, N-bits

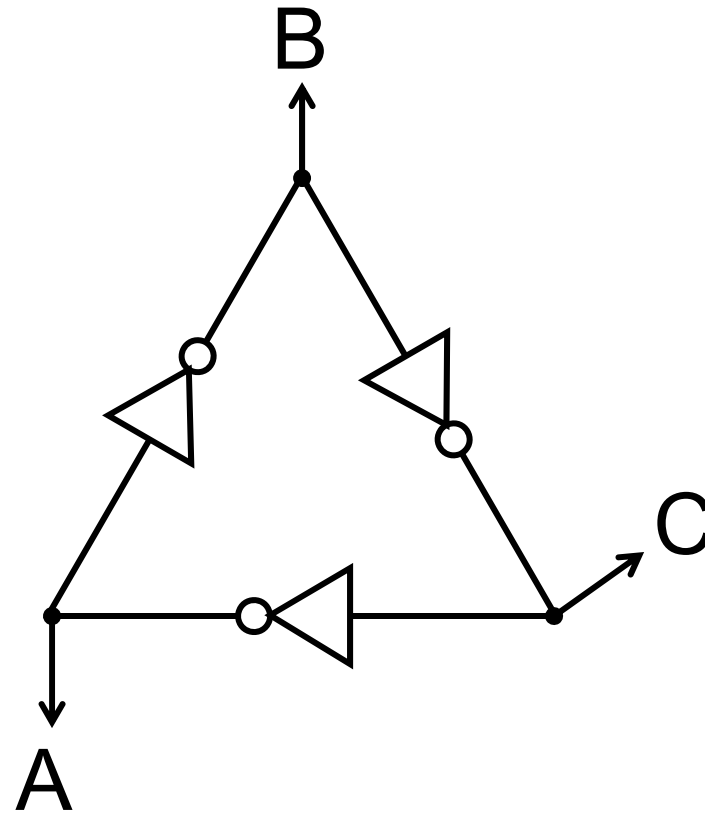## Basic Building Blocks

- Decoders and Encoders

# Goal

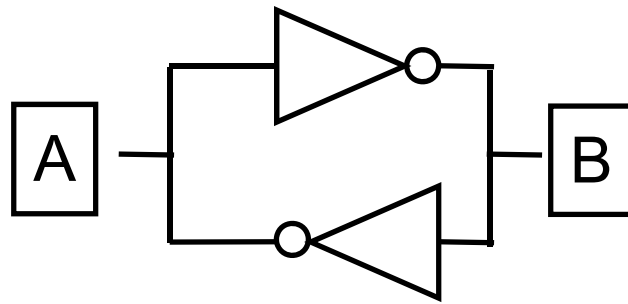How do we store store **one** bit?
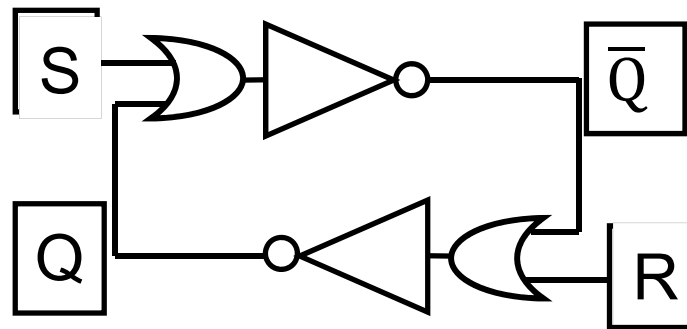
# First Attempt: Unstable Devices

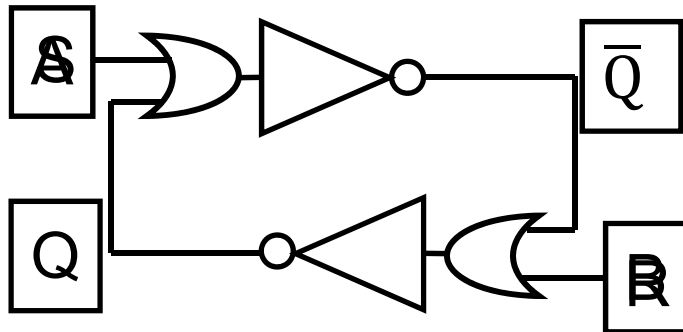# Second Attempt: Bistable Devices

- Stable and unstable equilibria?

A Simple Device

# Third Attempt: Set-Reset Latch

# Third Attempt: Set-Reset Latch



| A | B | OR | NOR |
|---|---|----|-----|
| 0 | 0 | 0  | 1   |
| 0 | 1 | 1  | 0   |
| 1 | 0 | 1  | 0   |
| 1 | 1 | 1  | 0   |

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

Set-Reset (S-R) Latch
Stores a value Q and its complement

# Third Attempt: Set-Reset Latch
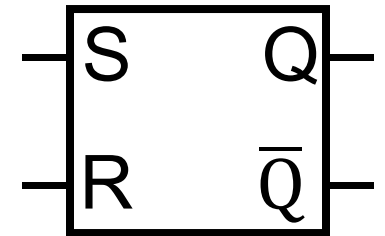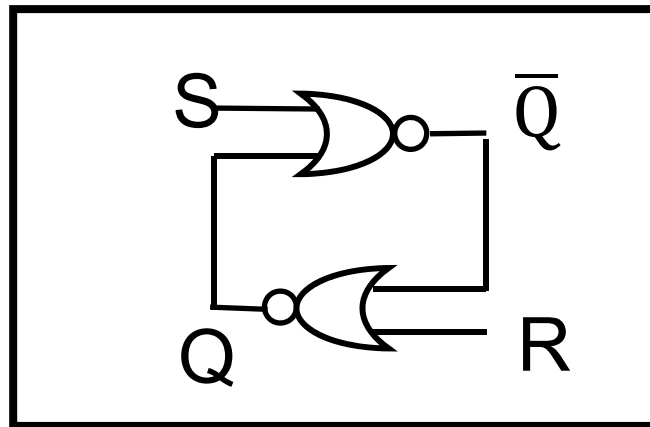


| S | R | Q | $\bar{Q}$ |
|---|---|---|---|
| 0 | 0 | Q | $\bar{Q}$ |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 |   |   |

Set-Reset (S-R) Latch
Stores a value Q and its complement

# Takeaway

Set-Reset (SR) Latch can store one bit and we can change the value of the stored bit. But, SR Latch has a forbidden state.
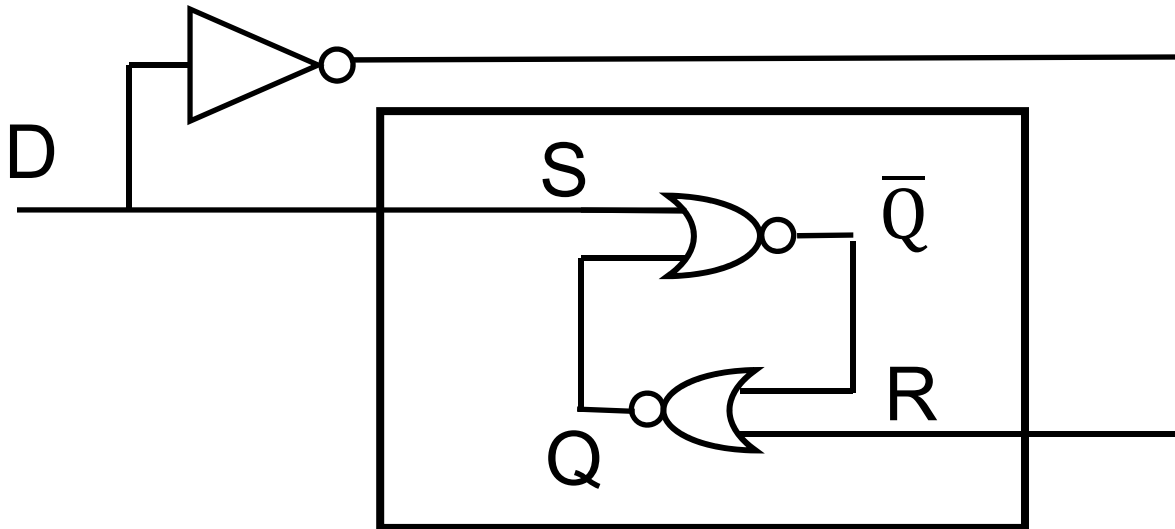
# Next Goal

How do we avoid the forbidden state of S-R Latch?

# Fourth Attempt: (Unclocked) D Latch



Fill in the truth table?

| D | Q | $\overline{Q}$ |
|---|---|---|
| 0 |   |   |
| 1 |   |   |

| A | B | OR | NOR |
|---|---|----|-----|
| 0 | 0 | 0  | 1   |
| 0 | 1 | 1  | 0   |
| 1 | 0 | 1  | 0   |
| 1 | 1 | 1  | 0   |

# Takeaway

Set-Reset (SR) Latch can store one bit and we can change the value of the stored bit.  But, SR Latch has a forbidden state.
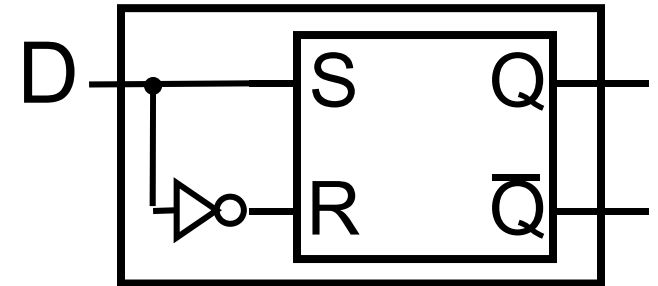
(Unclocked) D Latch can store and change a bit like an SR Latch while avoiding the forbidden state.
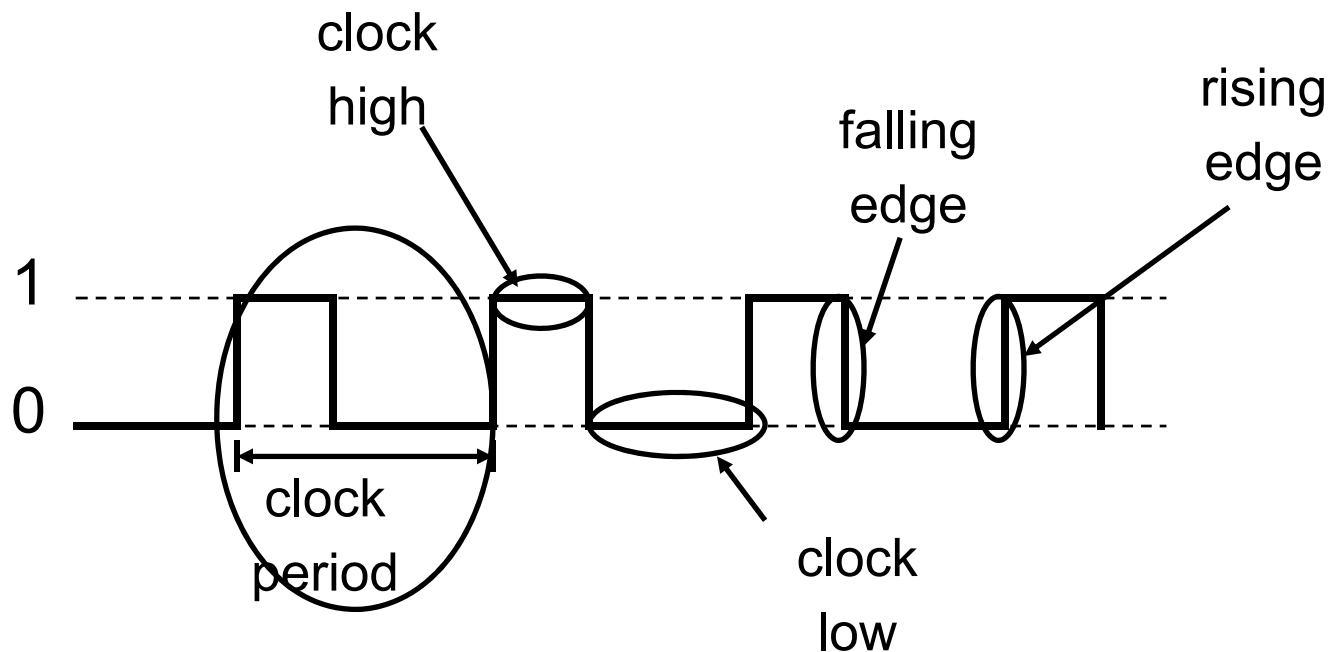
# Next Goal

How do we coordinate state changes to a D Latch?
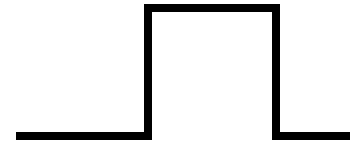
# Aside: Clocks

Clock helps coordinate state changes

- Usually generated by an oscillating crystal
- Fixed period
- Frequency = 1/period
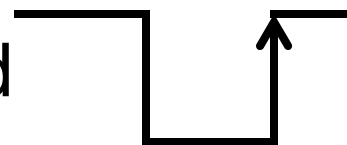
# Clock Disciplines

## Level sensitive

- State changes when clock is high (or low)

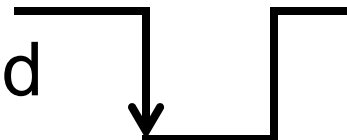## Edge triggered

- State changes at clock edge

positive edge-triggered

negative edge-triggered

# Clock Methodology

## Clock Methodology

- ## Negative edge, synchronous



$t_{setup}$    $t_{hold}$

clk    $t_{combinational}$

| compute | save | compute | save | compute |
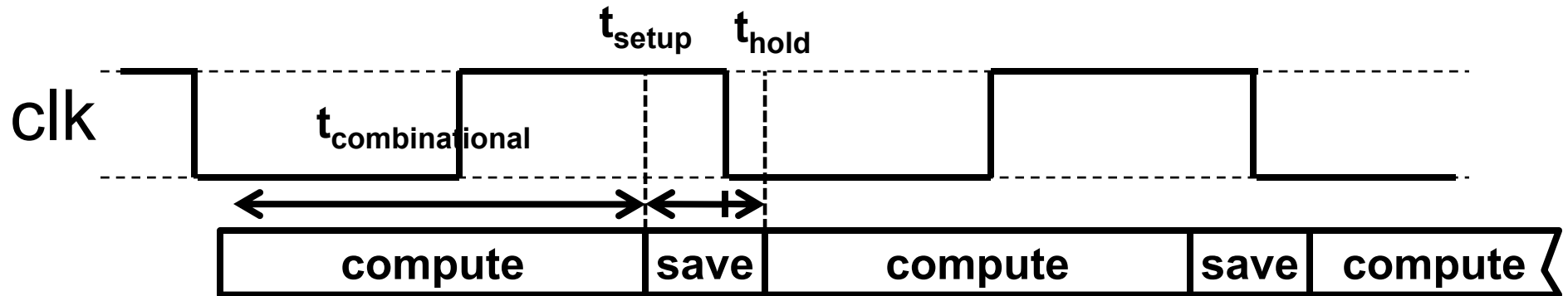
Edge-Triggered → signals must be stable near falling edge

"near" = before and after

    $t_{setup}$        $t_{hold}$

# Round 2: D Latch (1)



- Inverter prevents SR Latch from entering 1,1 state

|  | D | Q | $\overline{Q}$ |  |
|---|---|---|---|---|
|  | 0 |  |  | *Reset* |
|  | 1 |  |  | *Set* |
|  |  |  |  |  |
|  |  |  |  |  |

# Round 2: D Latch (1)
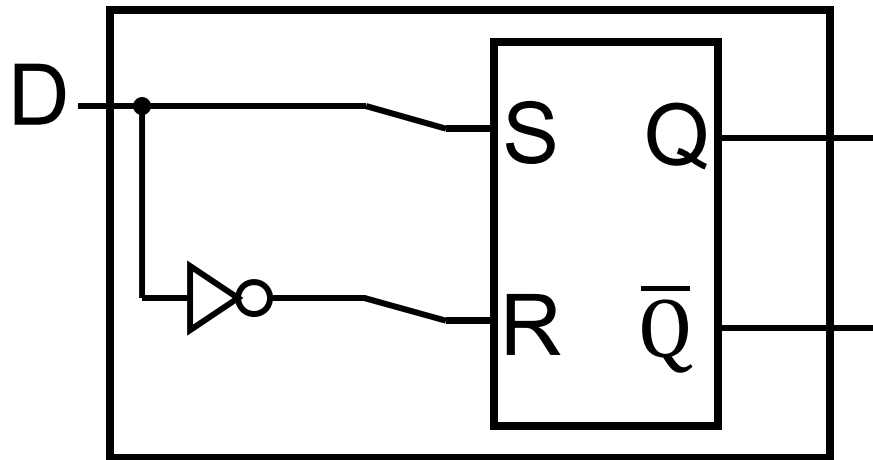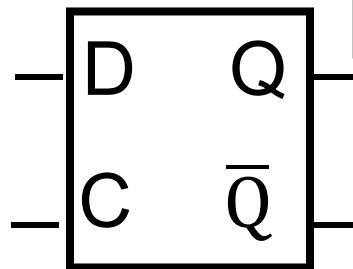
- Level sensitive
- Inverter prevents SR Latch from entering 1,1 state
- C enables changes

C = 1, D Latch *transparent*:
   set/reset (according to D)
C = 0, D Latch *opaque*:
   keep state (ignore D)

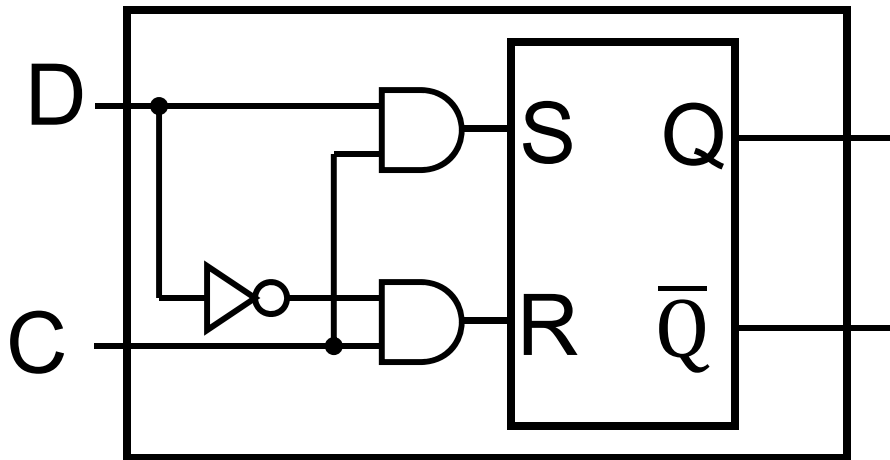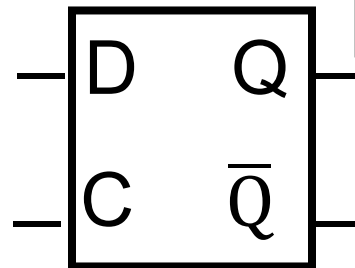| C | D | Q | $\overline{Q}$ | |
|---|---|---|---|---|
| 0 | 0 | | | *No Change* |
| 0 | 1 | | | |
| 1 | 0 | | | *Reset* |
| 1 | 1 | | | *Set* |

18

# Round 2: D Latch (1)



- Level sensitive
- Inverter prevents SR Latch from entering 1,1 state
- C enables changes

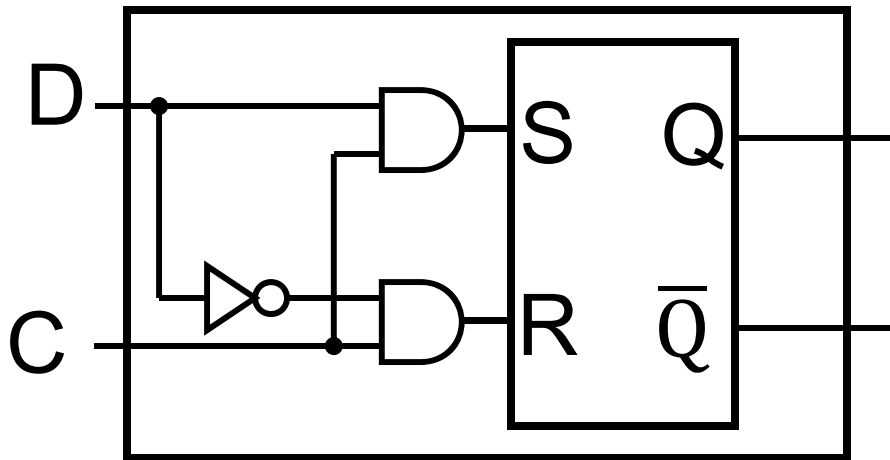C = 1, D Latch *transparent*: set/reset (according to D)

C = 0, D Latch *opaque*: keep state (ignore D)

| S | R | Q | $\overline{Q}$ | |
|---|---|---|---|---|
| 0 | 0 | Q | $\overline{Q}$ | hold |
| 0 | 1 | 0 | 1 | reset |
| 1 | 0 | 1 | 0 | set |
| 1 | 1 | forbidden | | |

| C | D | Q | $\overline{Q}$ | |
|---|---|---|---|---|
| 0 | 0 | | | *No Change* |
| 0 | 1 | | | |
| 1 | 0 | | | *Reset* |
| 1 | 1 | | | *Set* |
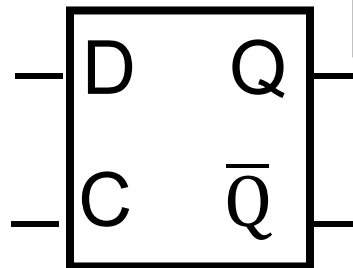
# Round 2: D Latch(1)
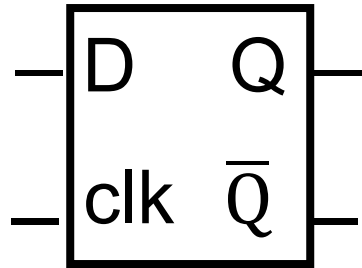
Level Sensitive D Latch

Clock high:

    set/reset (according to D)

Clock low:

    keep state (ignore D)
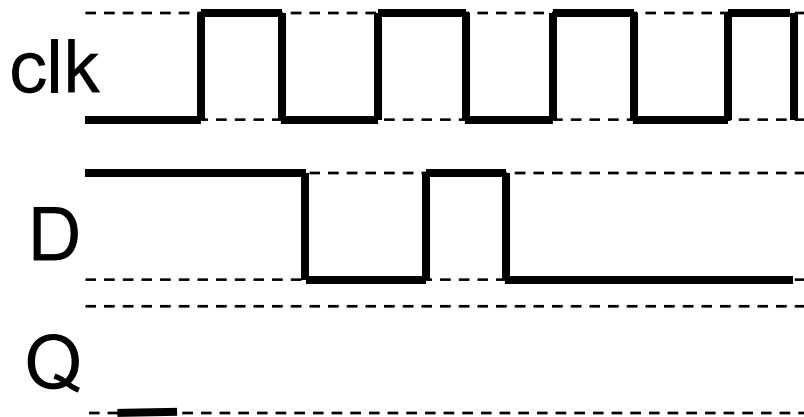


| clk | D | Q | $\overline{Q}$ |
|-----|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

# Round 3: D Flip-Flop
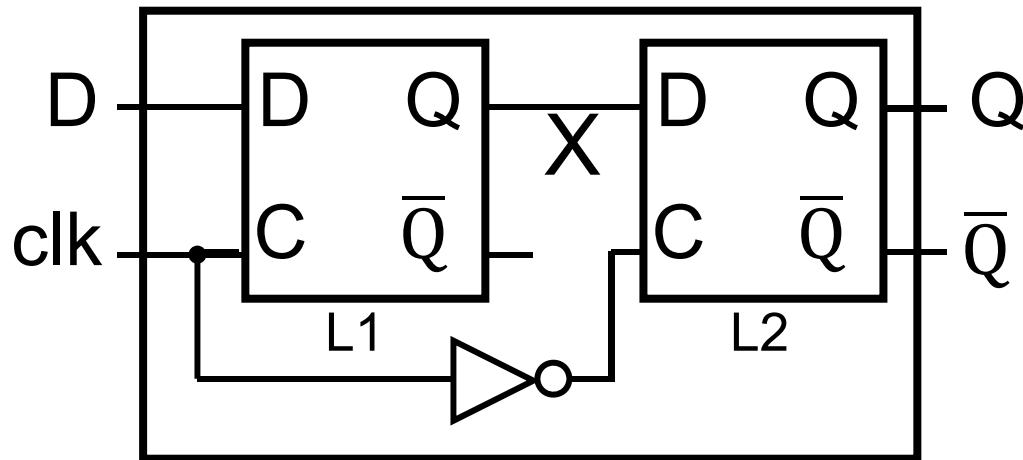


- Edge-Triggered
- Data captured when clock high
- Output changes only on falling edges

# Round 3: D Flip-Flop

**Clock = 1:** L1 *transparent*
L2 *opaque*

*When CLK rises (0→1),
now X can change,
Q does not change*
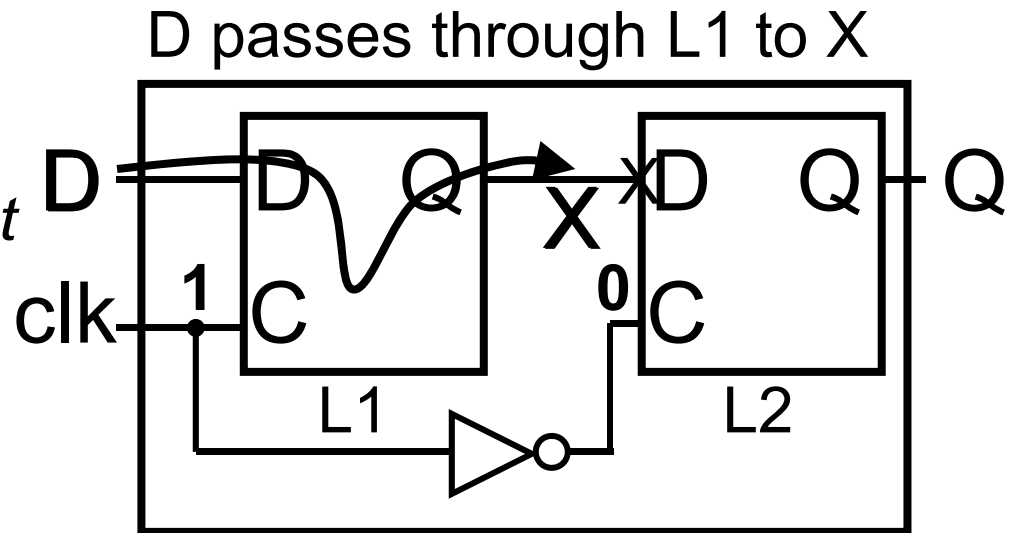
**Clock = 0:** L1 *opaque*
L2 *transparent*

*When **CLK falls** (1→0),
Q gets X, X cannot change*

D passes through L1 to X

D → D Q → X → D Q ⊢ Q
clk → **1** C    **0** C
L1    L2

X passes through L2 to Q

D → D Q → X → D Q → Q
clk → **0** C    **1** C
L1    L2

22
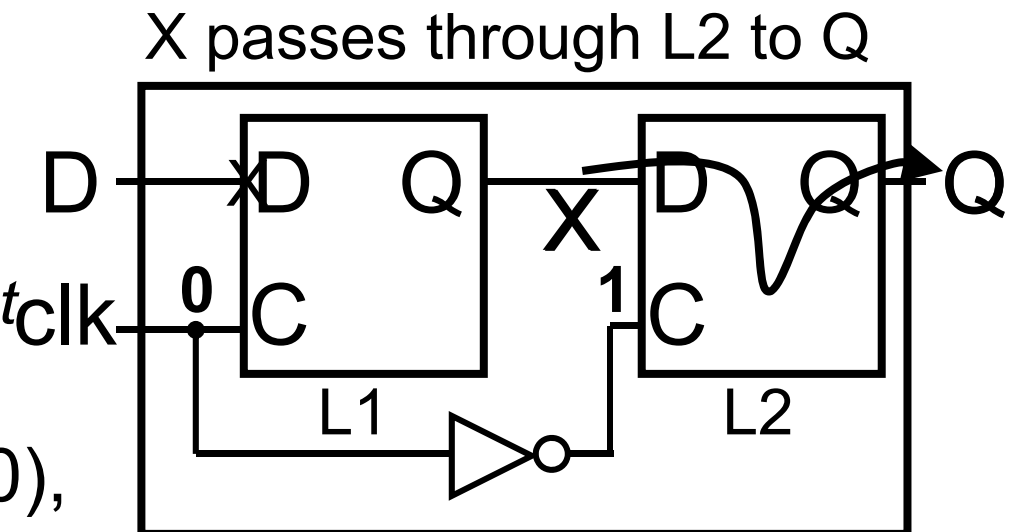
# Edge-Triggered D Flip-Flop
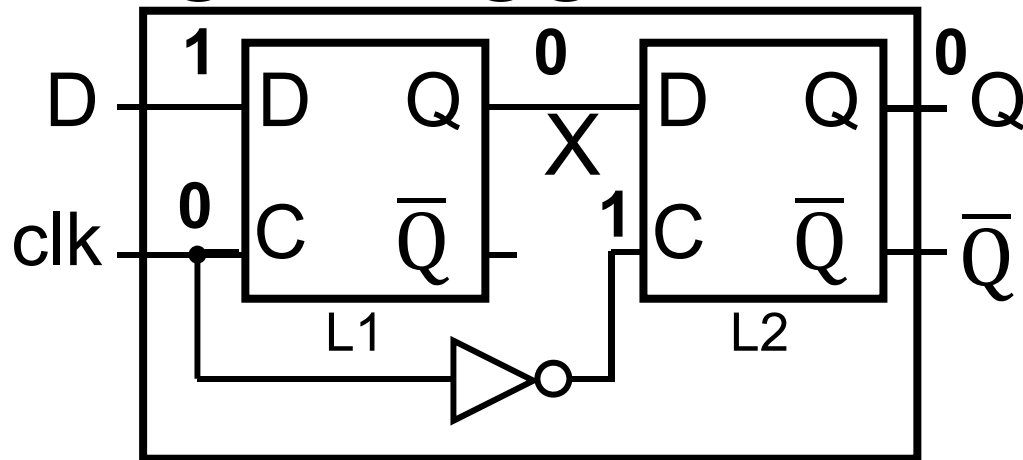
## D Flip-Flop

- Edge-Triggered
- Data captured when clock is high
- Output changes only on falling edges
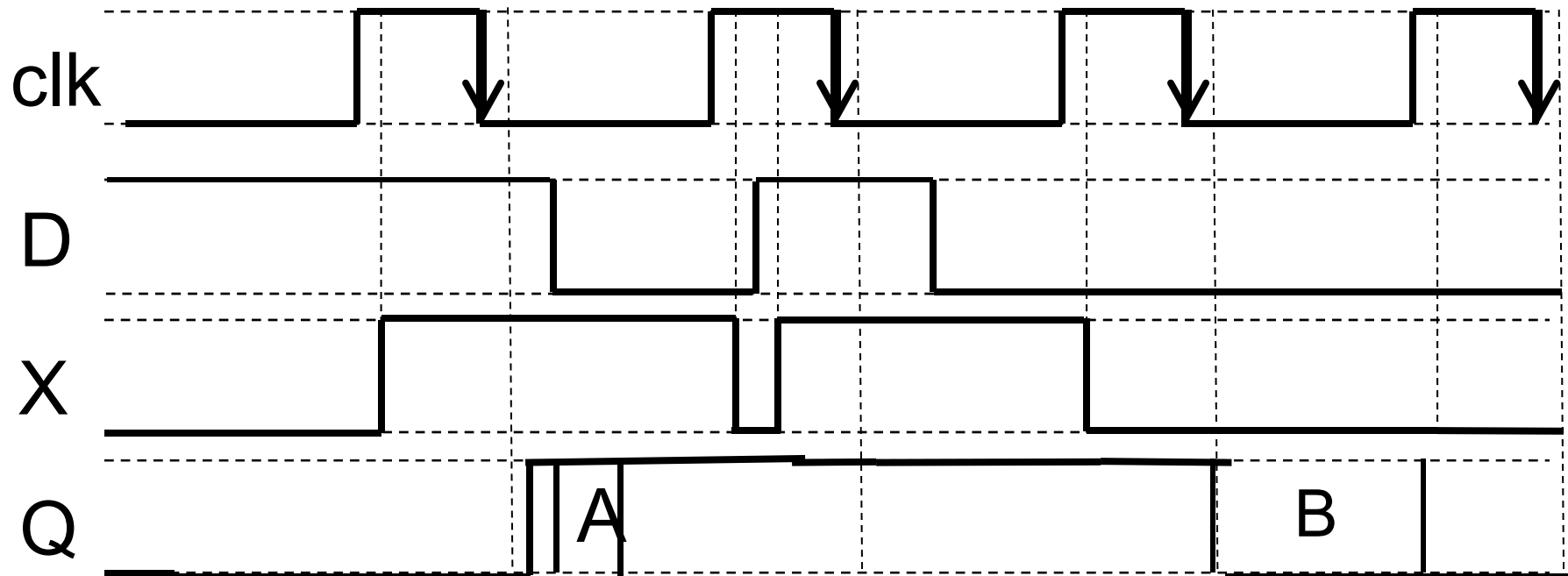
# Takeaway

Set-Reset (SR) Latch can store one bit and we can change the value of the stored bit. But, SR Latch has a forbidden state.

(Unclocked) D Latch can store and change a bit like an SR Latch while avoiding a forbidden state.

An Edge-Triggered D Flip-Flip (aka Master-Slave D Flip-Flip) stores one bit. The bit can be changed in a synchronized fashion on the edge of a clock signal.

# Next Goal

How do we store more than one bit, N bits?

# Registers

D0

D1

D2

D3

clk

Register
- D flip-flops in parallel
- shared clock
- extra clocked inputs:
  write_enable, reset, …
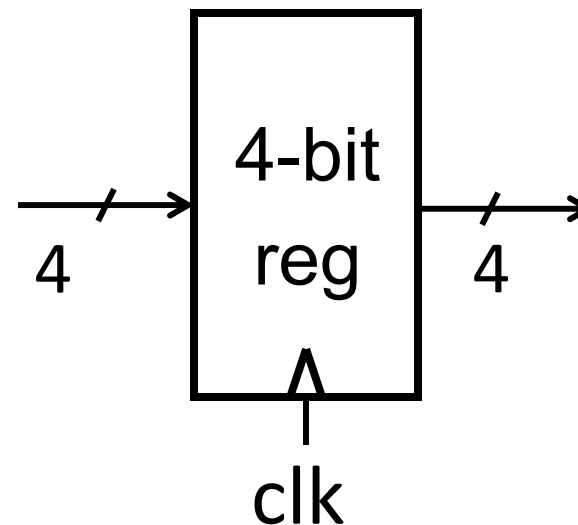
4-bit
reg

4

4

clk

# Takeaway

Set-Reset (SR) Latch can store one bit and we can change the value of the stored bit.  But, SR Latch has a forbidden state.

(Unclocked) D Latch can store and change a bit like an SR Latch while avoiding a forbidden state.

An Edge-Triggered D Flip-Flip (aka Master-Slave D Flip-Flip) stores one bit.  The bit can be changed in a synchronized fashion on the edge of a clock signal.
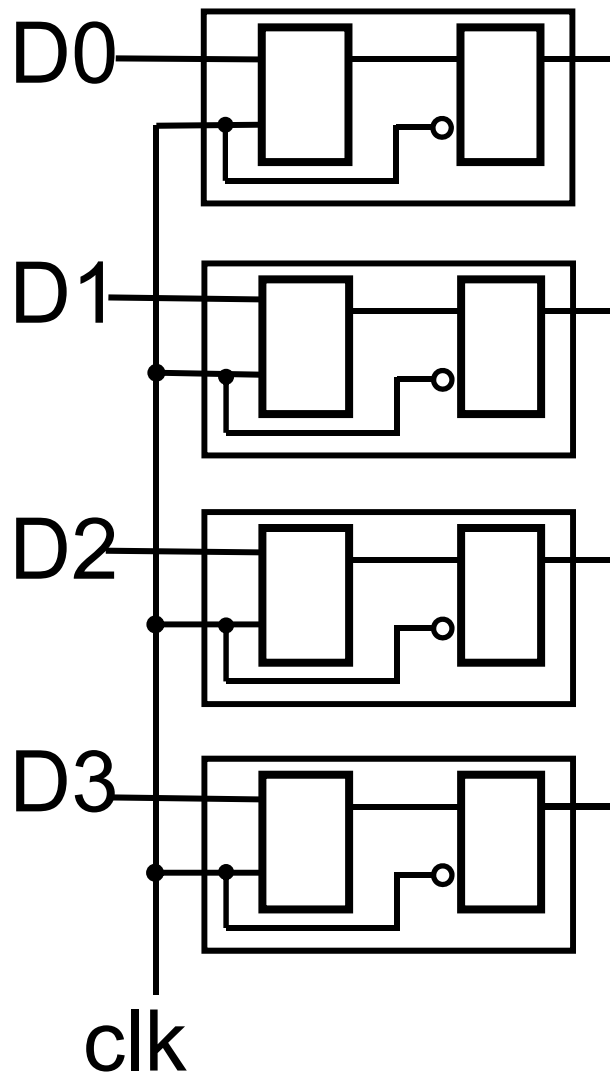
An $N$-bit **register** stores $N$-bits.  It is created with $N$ D-Flip-Flops in parallel along with a shared clock.

# An Example: What will this circuit do?
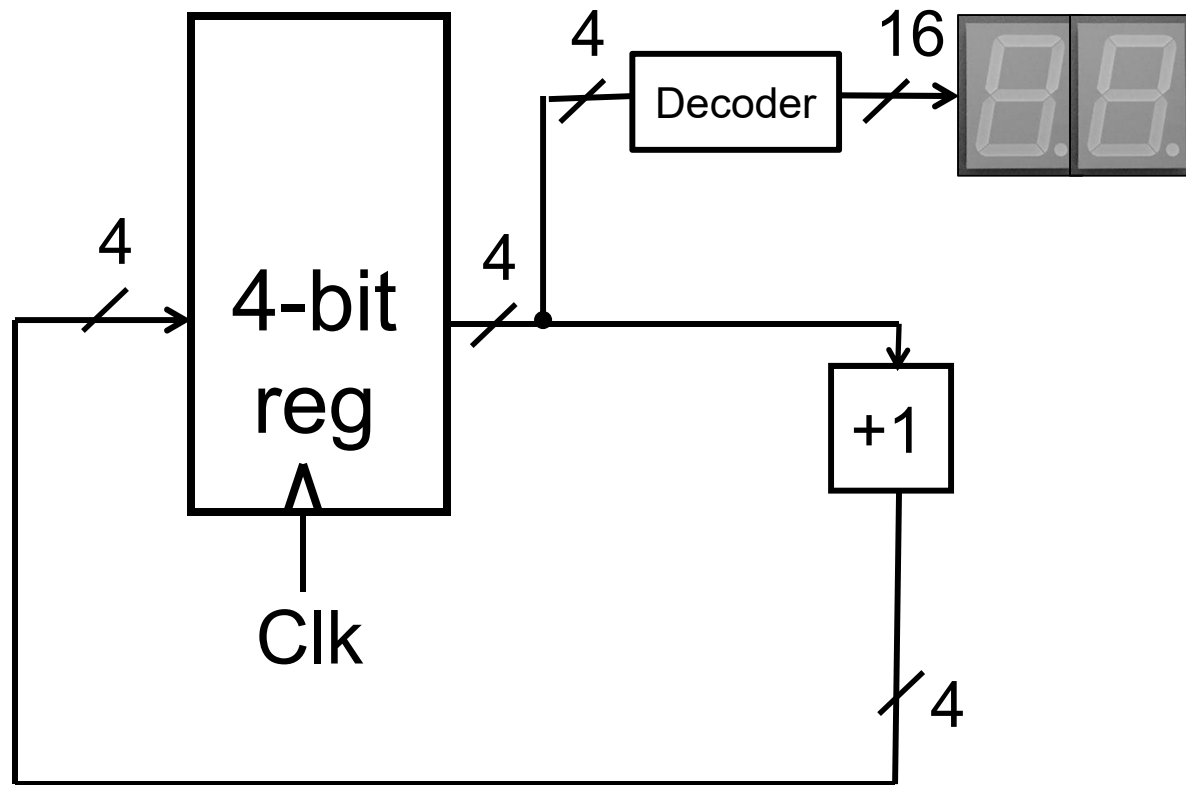
# Decoder Example: 7-Segment LED

## 7-Segment LED

- photons emitted when electrons fall into holes

d7 d6 ⏚ d5 d4



d3 d2 ⏚ d1 d0

# Decoder Example: 7-Segment LED Decoder
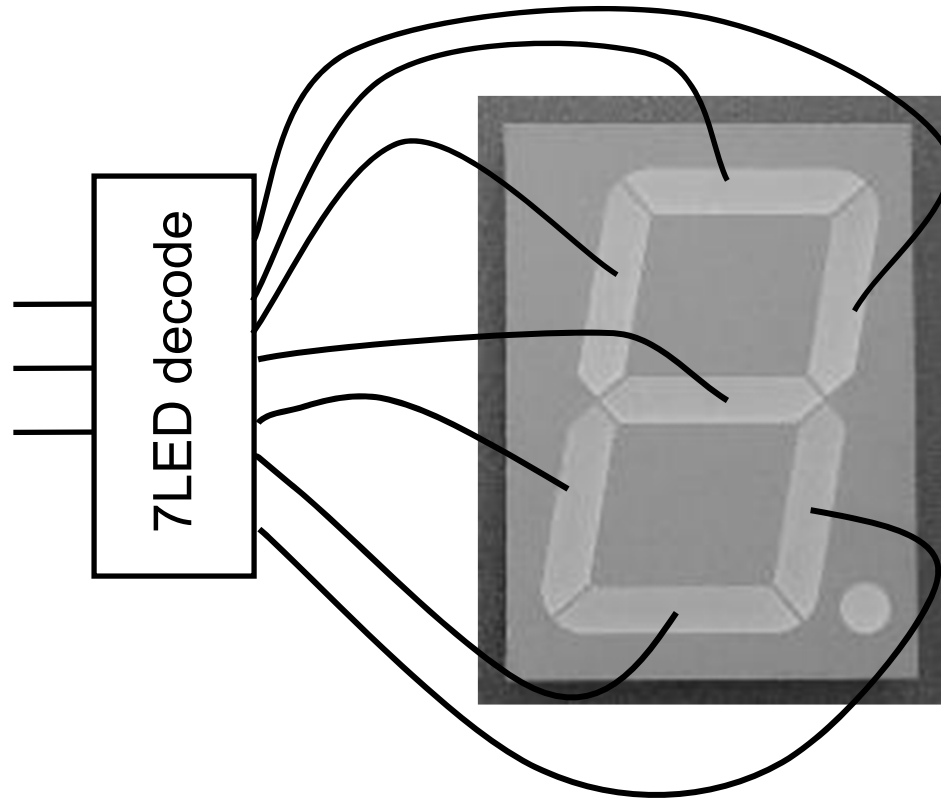
7LED decode

3 inputs
- encode 0 – 7 in binary

7 outputs
- one for each LED

# 7 Segment LED Decoder Implementation

| b2 | b1 | b0 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
|----|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | | | | | | | |
| 0 | 0 | 1 | | | | | | | |
| 0 | 1 | 0 | | | | | | | |
| 0 | 1 | 1 | | | | | | | |
| 1 | 0 | 0 | | | | | | | |
| 1 | 0 | 1 | | | | | | | |
| 1 | 1 | 0 | | | | | | | |
| 1 | 1 | 1 | | | | | | | |

# Basic Building Blocks We have Seen

binary encoder

$2^N$

$N$

binary decoder

$N$

$2^N$

Multiplexor

$N$ → 0

$N$ → 1

$N$ → 2

...

$N$ → $2^M$-1

$N$

$M$

# Encoders

```
        ┌──────────────┐
    ───┤ 0            │
        │              │
    ───┤ 1            │
        │              │
    ───┤ 2            ├───
        │              │
    ───┤ 3            ├───
        │              │
    ───┤ 4   encoder  ├───
        │              │   .
    ───┤ 5            │   .
        │              │   .
    ───┤ 6            ├───
        │              │
    ───┤ 7            │
     .  │              │
     .  │              │
     .  │              │
    ───┤ N            │
        └──────────────┘
```

N Input wires
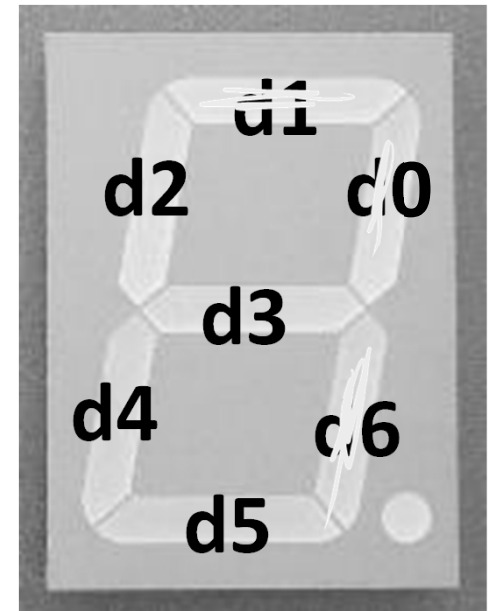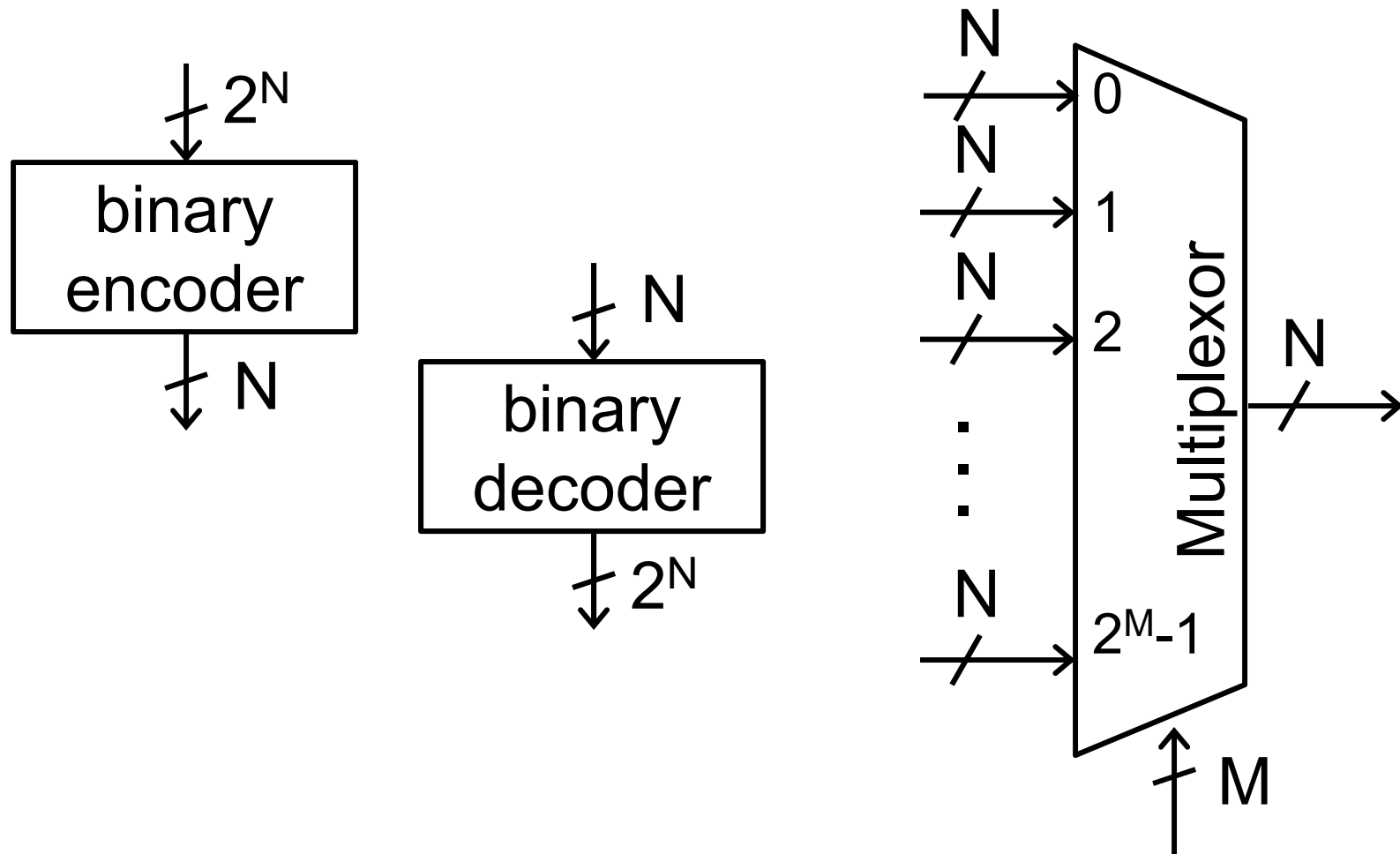
$Log_2(N)$ outputs wires
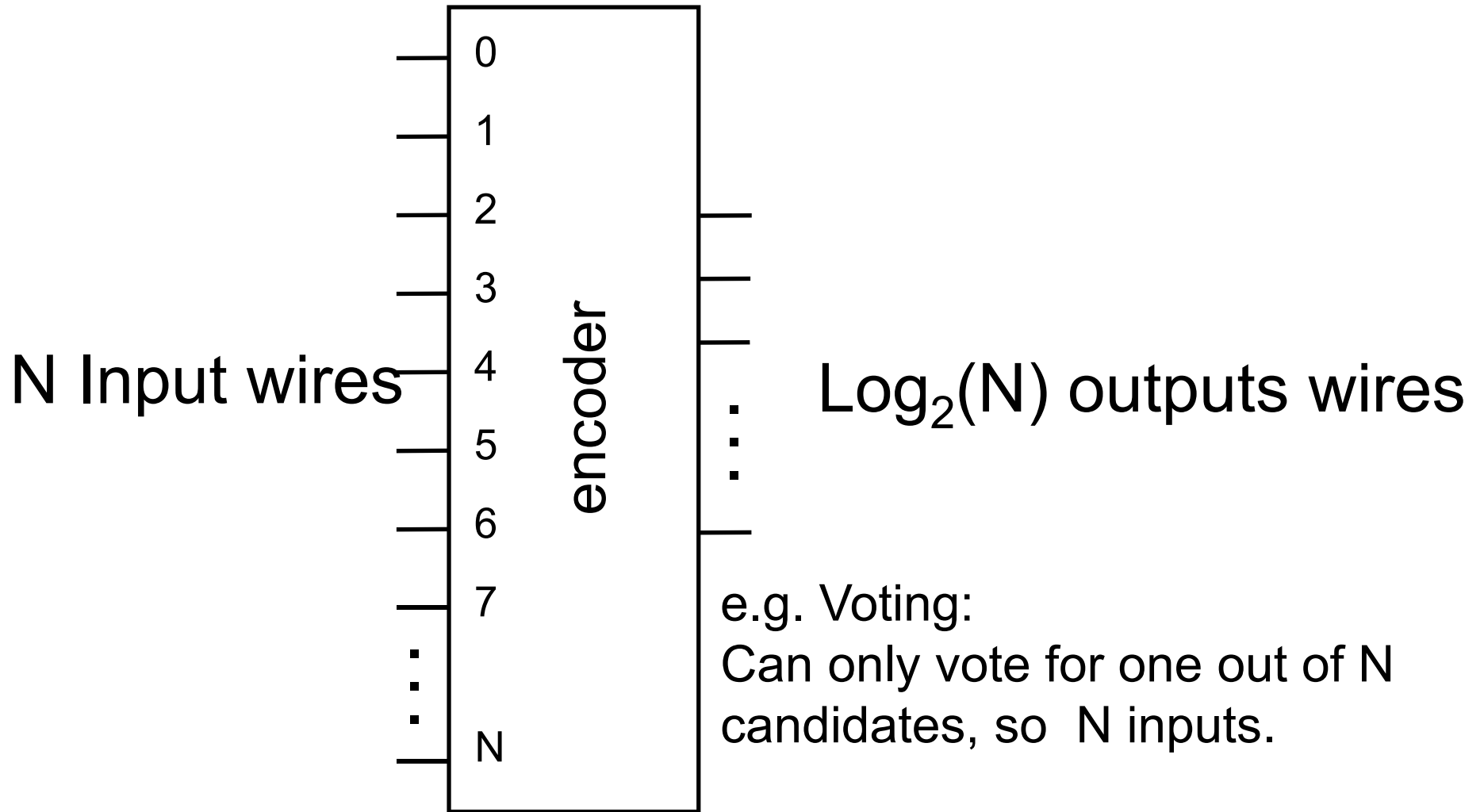
e.g. Voting:
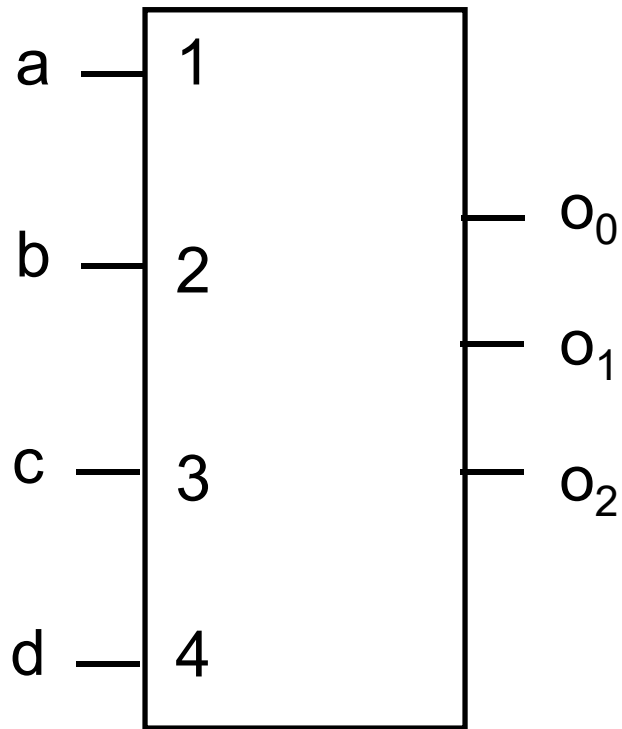Can only vote for one out of N
candidates, so  N inputs.

But can encode vote efficiently
with binary encoding.

33

# Example Encoder Truth Table

| a | b | c | d | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | |
| 1 | 0 | 0 | 0 | | | | |
| 0 | 1 | 0 | 0 | | | | |
| 0 | 0 | 1 | 0 | | | | |
| 0 | 0 | 0 | 1 | | | | |

a — 1

b — 2

c — 3

d — 4

$o_0$

$o_1$

$o_2$

A 3-bit
encoder
with 4 inputs
for simplicity

# Basic Building Blocks Example: Voting



Ballots

The 3410 optical scan
vote reader
machine

7LED

enc

7

# Basic Building Blocks We have Seen



binary
encoder

$2^N$

$N$

binary
decoder

$N$

$2^N$

Multiplexor

$N$ → 0
$N$ → 1
$N$ → 2
...
$N$ → $2^M$-1

$N$ →

$M$
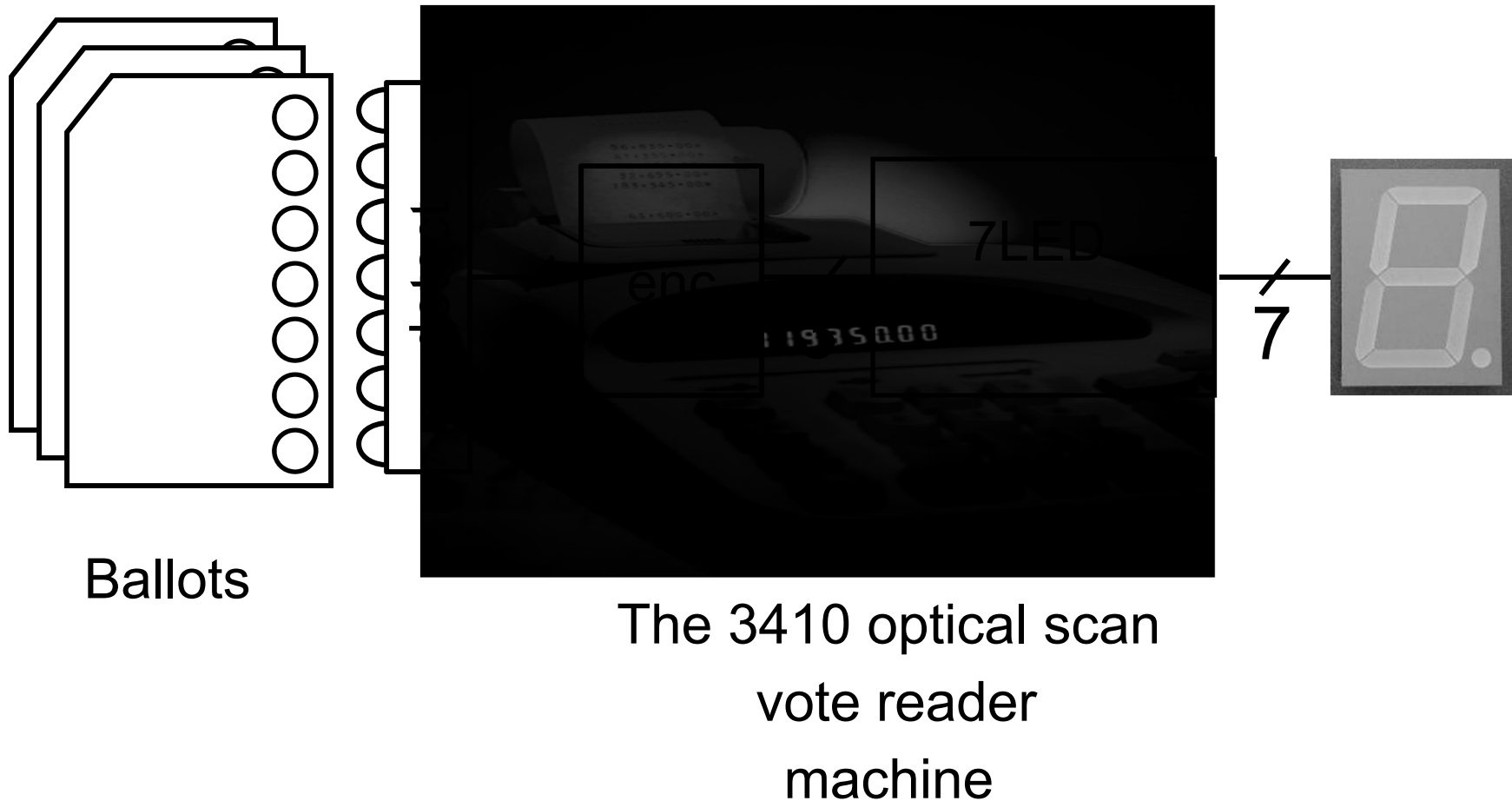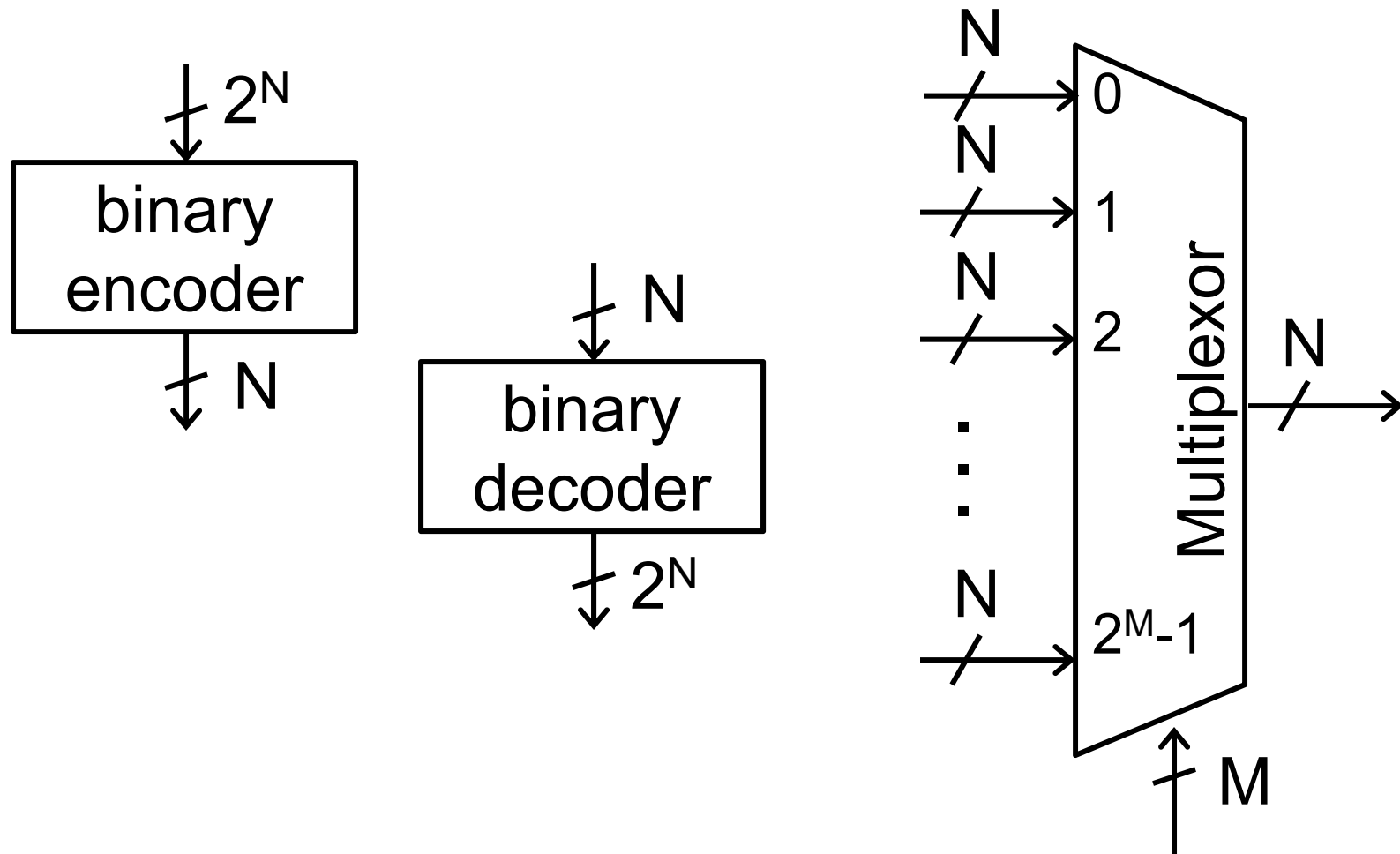
# Recap

We can now build interesting devices with sensor

- Using combinational logic

We can also store data values (aka Sequential Logic)

- In state-holding elements
- Coupled with clocks

# Summary

We can now build interesting devices with sensor
- Using combinational logic

We can also store data values
- Stateful circuit elements (D Flip Flops, Registers, …)
- Clock to synchronize state changes