# EDA322
# Digital Design

Lecture 6:
**Sequential Circuits**

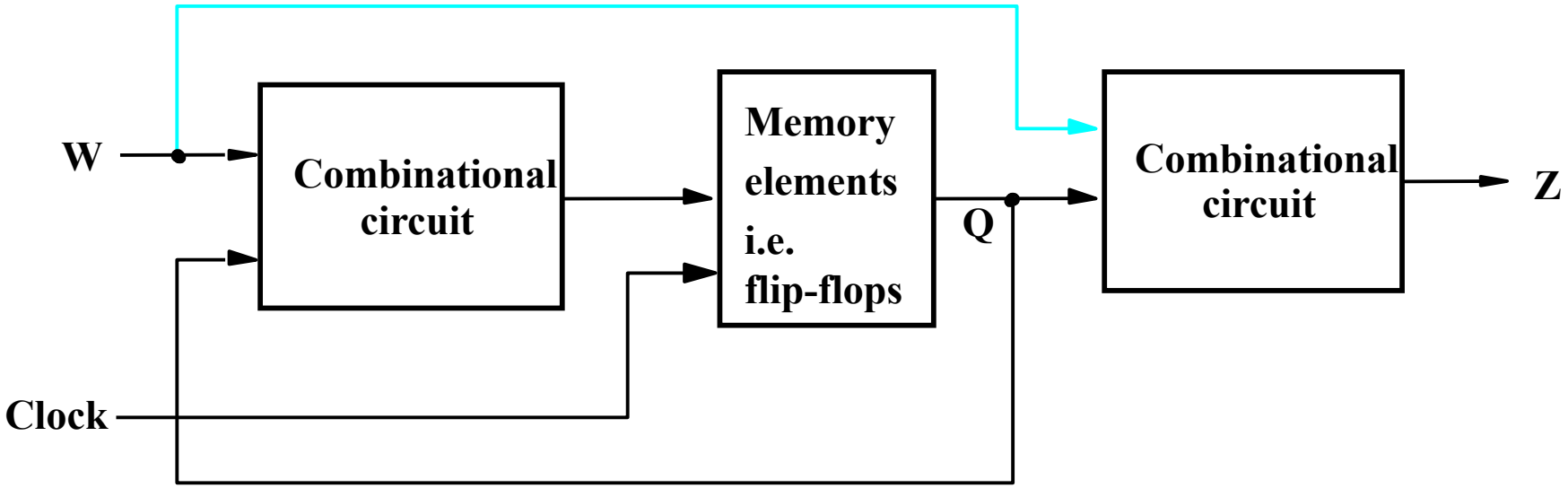
Ioannis Sourdis

# Outline of Lecture 6

- Introduction to sequential circuits
- Latches, Flip-flops
- Registers, Shift-registers, Counters
- Manual design steps
- Two design examples

# Sequential circuits

- The outputs depend not only on the **present inputs** but also on the **past behavior** of the circuit.
- Include memory elements
  - in addition to combination logic
- The contents of the memory elements represent the *state* of the circuit
- The circuit state transitions through a sequence of states



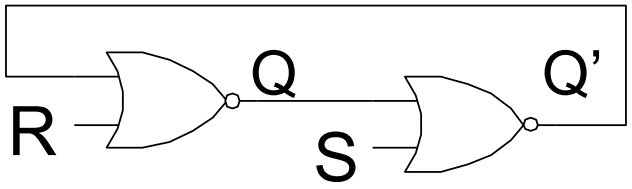

**The general form of a sequential circuit.**

**Q1 → Q2 → Q3 …**

# State-Holding Memory Elements

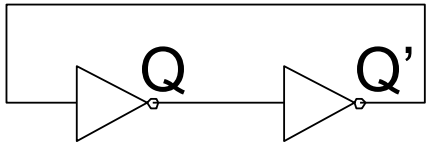
- Latch versus Flip-Flop
  - Latches are level-sensitive:
    - whenever clock is high, latch is transparent (output <= input)
    - whenever clock is low, latch keeps its previous value
  - Flip-flops are edge-sensitive: data passes through (i.e. data is sampled) only on a rising (or falling) edge of the clock
  - Latches cheaper to implement than flip-flops
  - Flip-flops are easier to design with than latches
- In this course, primarily use D flip-flops

# Cross-coupled NOR gates

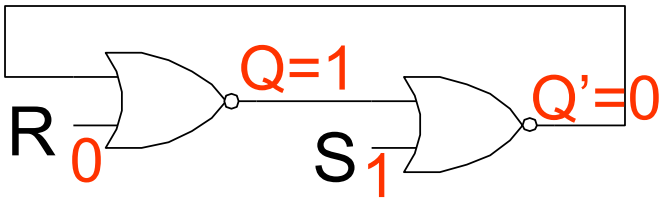



| NOR | |
|---|---|
| 00 | 1 |
| 01 | 0 |
| 10 | 0 |
| 11 | 0 |

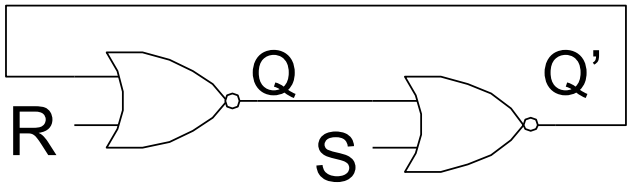
- If both R=0 & S=0, then cross-coupled NORs equivalent to a stable latch:




- What happens if R or S or both become = 1?

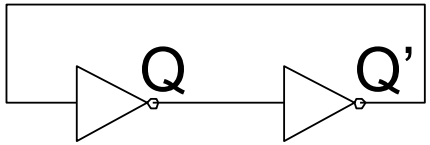


# Cross-coupled NOR gates

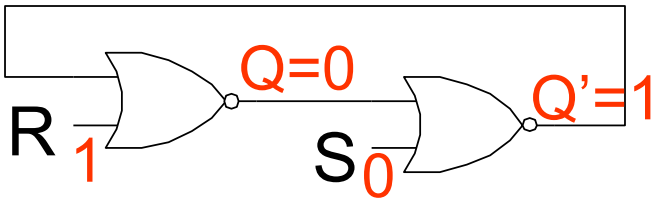



| NOR | |
|---|---|
| 00 | 1 |
| 01 | 0 |
| 10 | 0 |
| 11 | 0 |

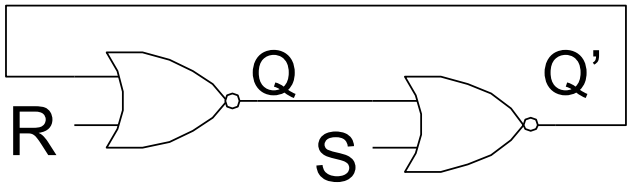
- If both R=0 & S=0, then cross-coupled NORs equivalent to a stable latch:




- What happens if R or S or both become = 1?

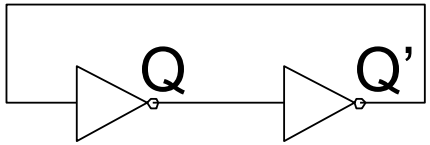


# Cross-coupled NOR gates

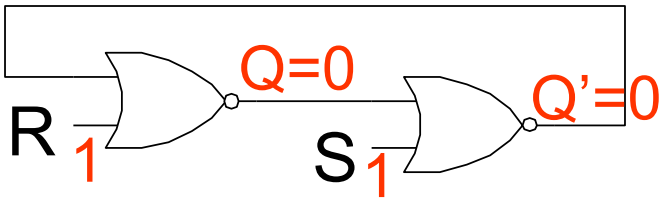



| NOR | |
|---|---|
| 00 | 1 |
| 01 | 0 |
| 10 | 0 |
| 11 | 0 |

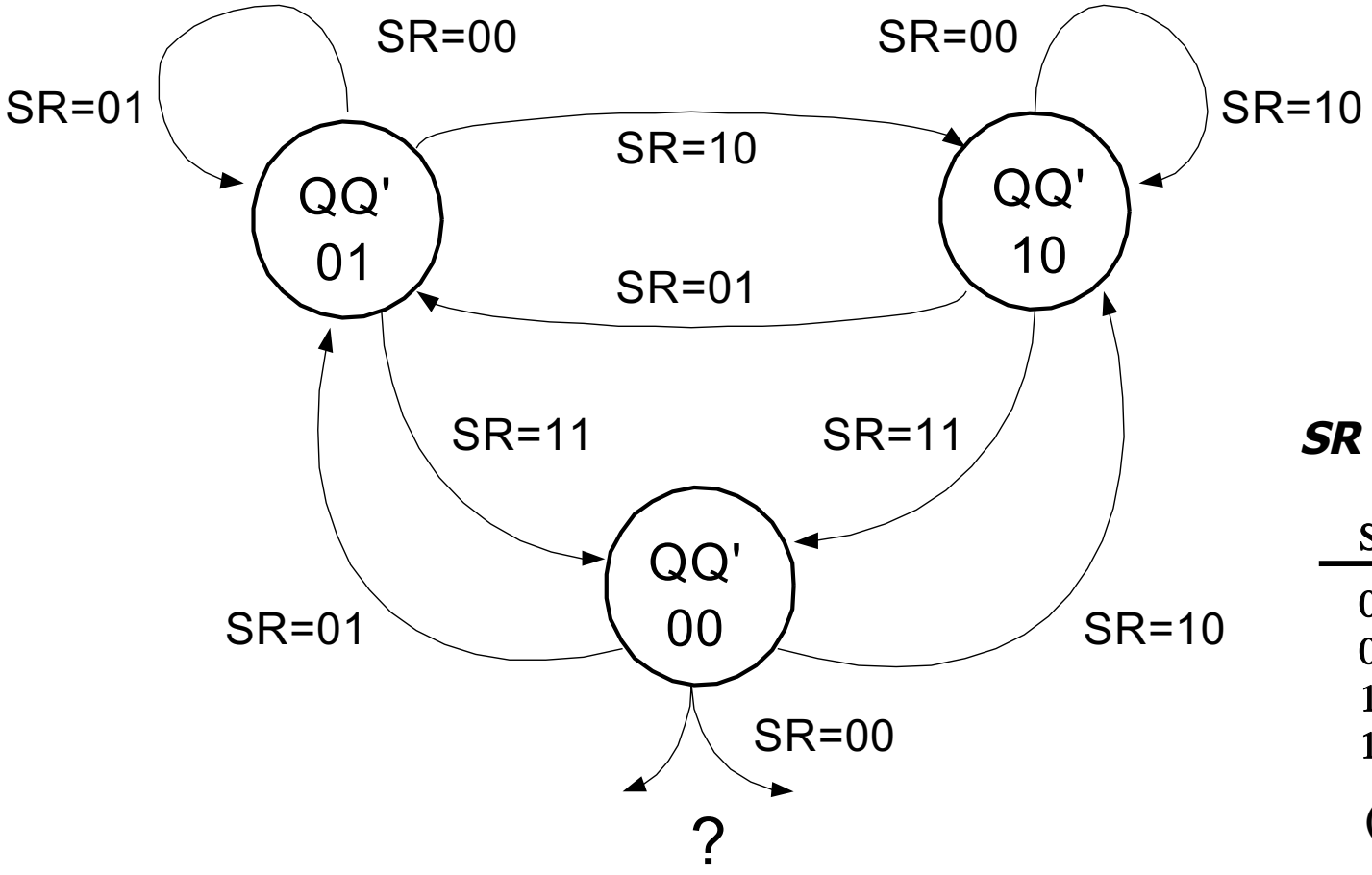
- If both R=0 & S=0, then cross-coupled NORs equivalent to a stable latch:




- What happens if R or S or both become = 1?



# Asynchronous State Transition Diagram

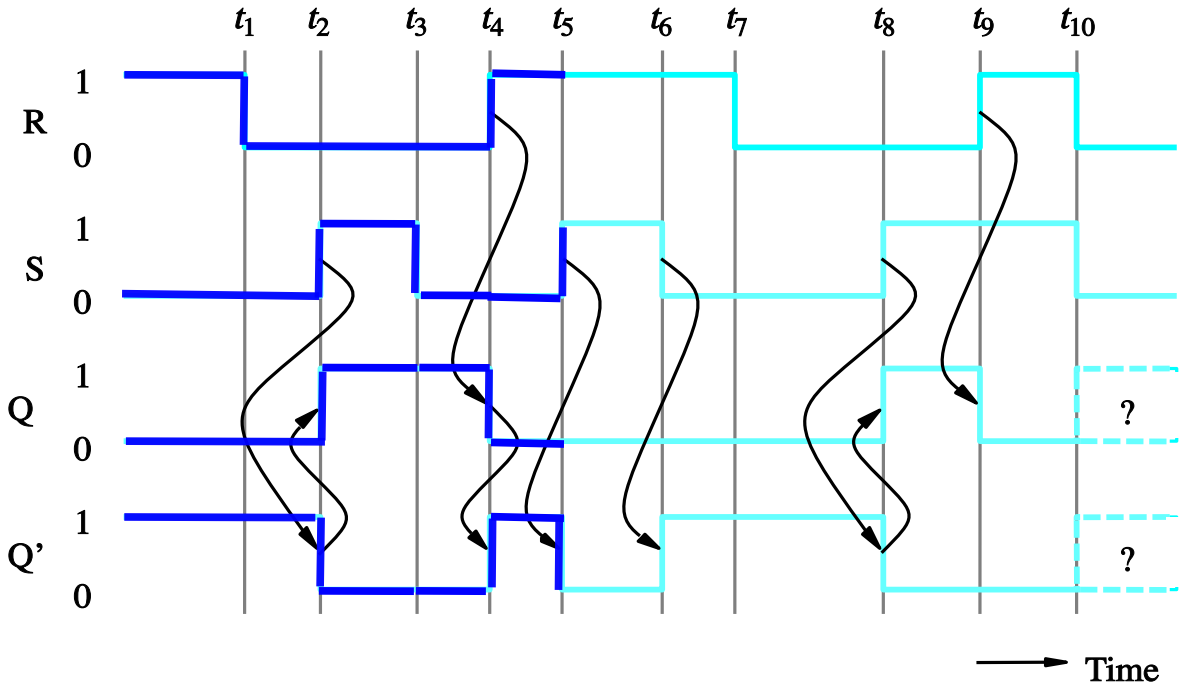



***SR Latch:***

| S | R | Q | Q' | |
|---|---|---|---|---|
| 0 | 0 | 0/1 | 1/0 | (no change) |
| 0 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |

**(b) Truth table**

# SR-latch timing diagram

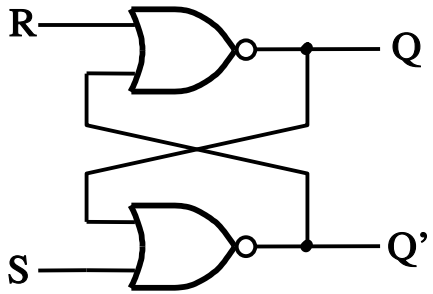



(c) Timing diagram

***SR Latch:***

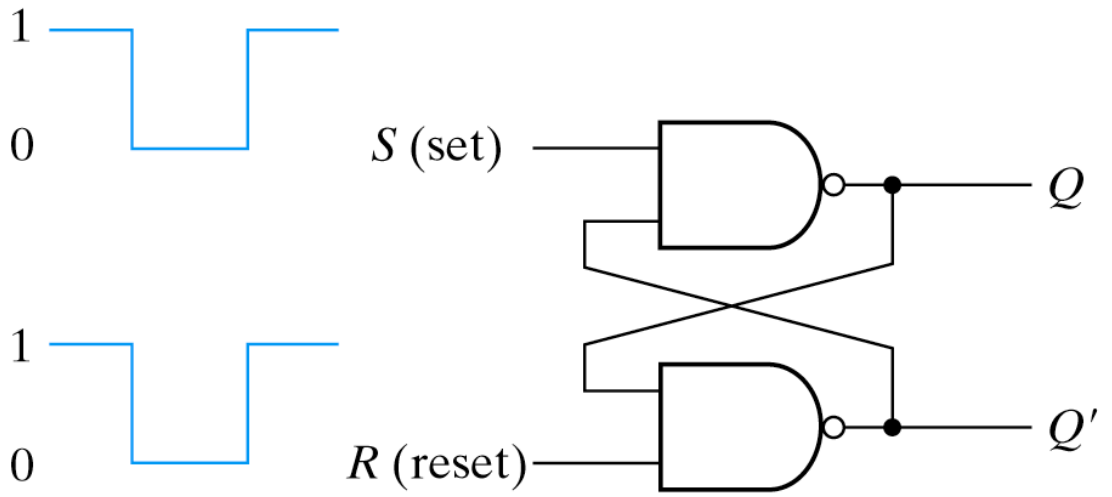
| S | R | Q | Q' | |
|---|---|---|---|---|
| 0 | 0 | 0/1 | 1/0 | (no change) |
| 0 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |

(b) Truth table




(a) Circuit

# NAND-gate based SR latch

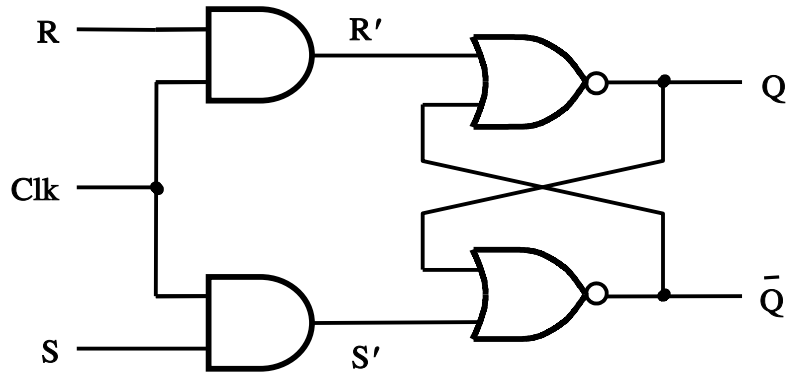



(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | |
| 1 | 1 | 0 | 1 | (after $S = 1, R = 0$) |
| 0 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 0 | (after $S = 0, R = 1$) |
| 0 | 0 | 1 | 1 | |

(b) Function table

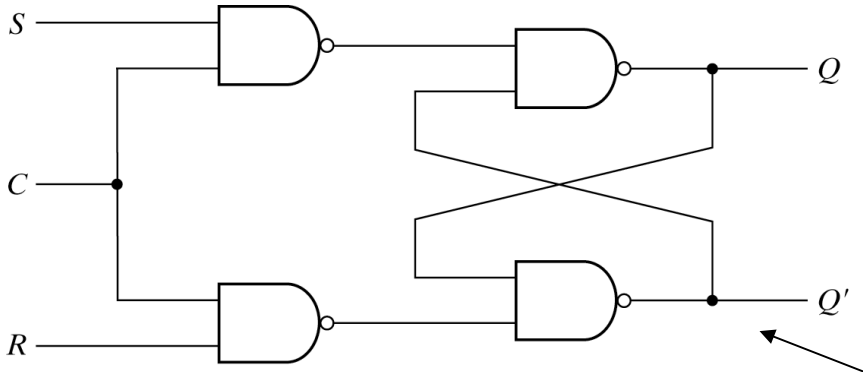
# Level-sensitive SR latch




(a) Circuit

| Clk | S | R | Q(t + 1) |
|---|---|---|---|
| 0 | x | x | Q(t) (no change) |
| 1 | 0 | 0 | Q(t) (no change) |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | x |

(b) Truth table



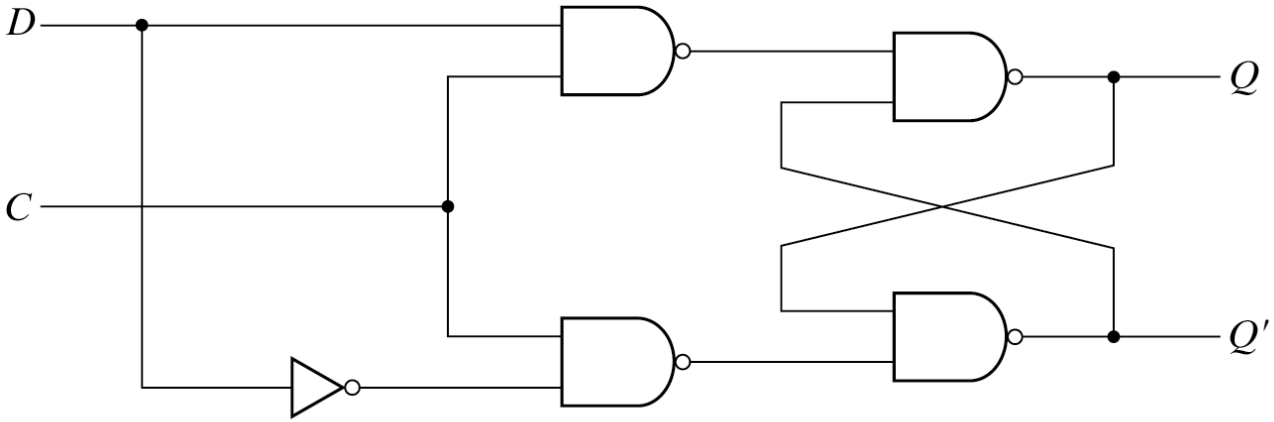

(a) Logic diagram

| C | S | R | Next state of Q |
|---|---|---|---|
| 0 | X | X | No change |
| 1 | 0 | 0 | No change |
| 1 | 0 | 1 | Q = 0; Reset state |
| 1 | 1 | 0 | Q = 1; set state |
| 1 | 1 | 1 | Indeterminate |

(b) Function table

**Requires fewer transistors**

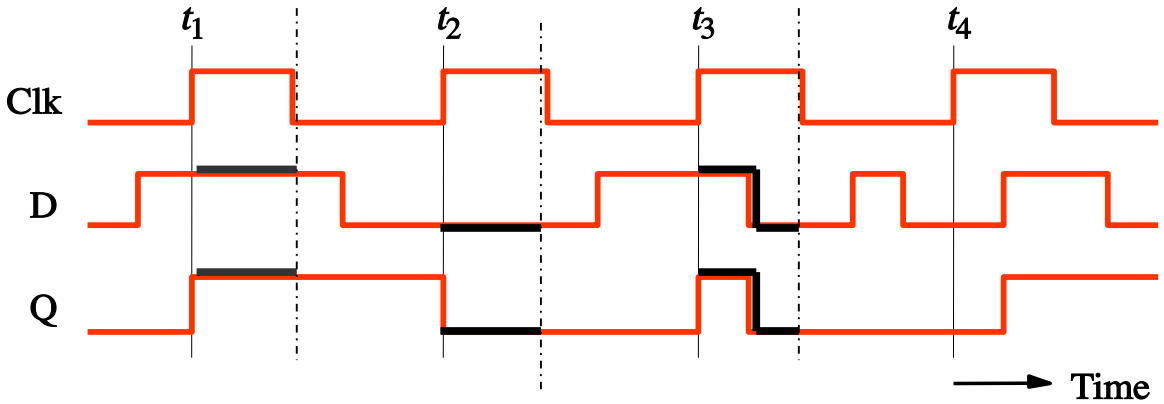
# D-latch




(a) Logic diagram

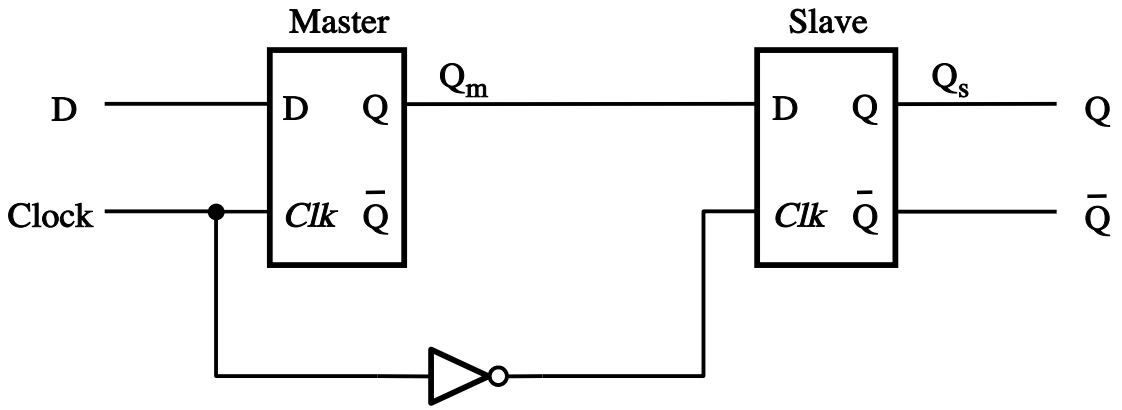
| C | D | Next state of $Q$ |
|---|---|---|
| 0 | X | No change |
| 1 | 0 | $Q = 0$; Reset state |
| 1 | 1 | $Q = 1$; Set state |

(b) Function table

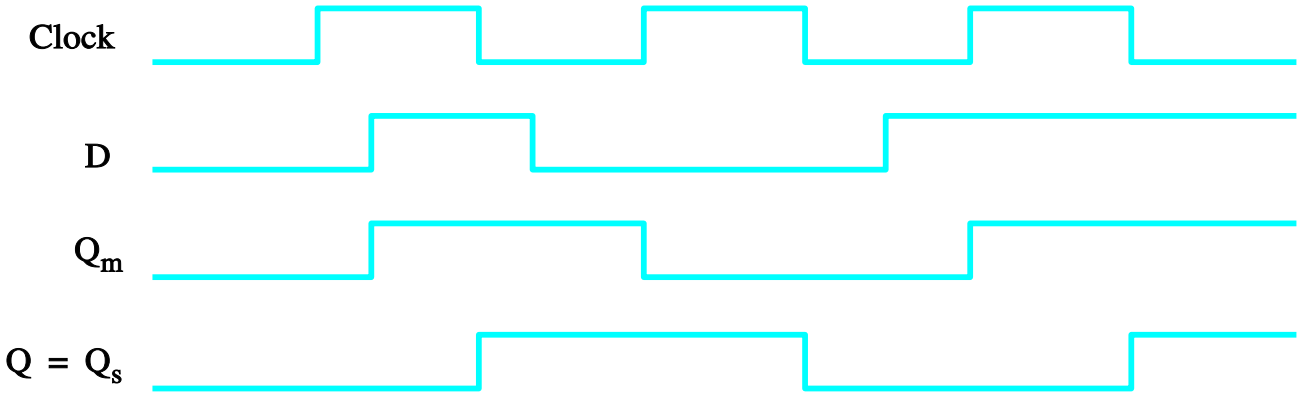



**(d) Timing diagram**

# Master-Slave D Flip-flop

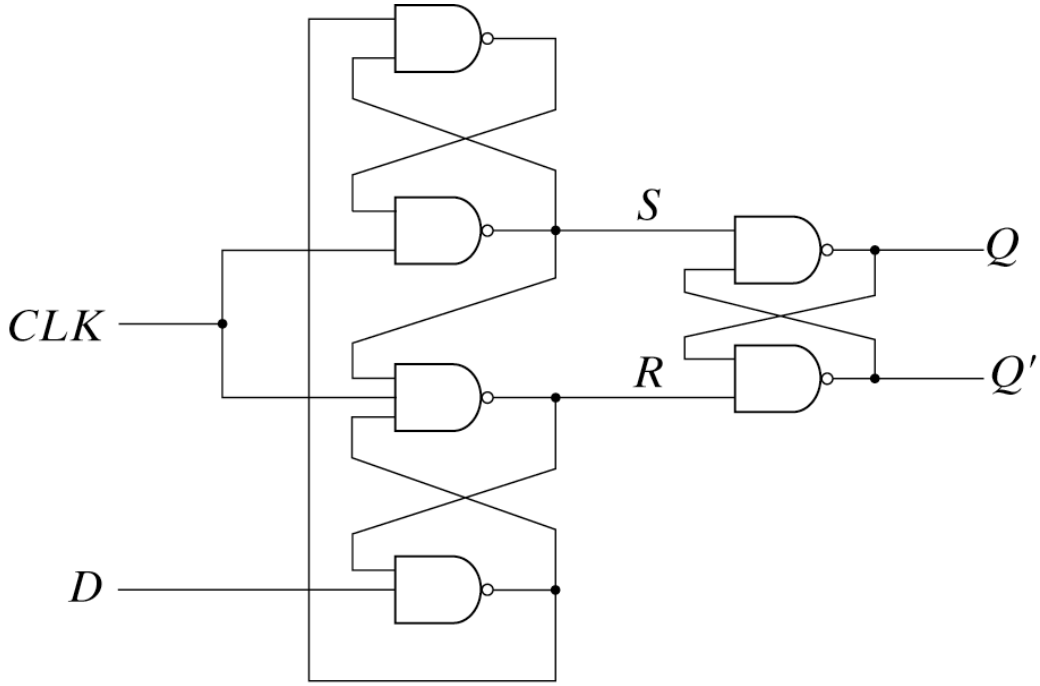



**(a) Circuit**



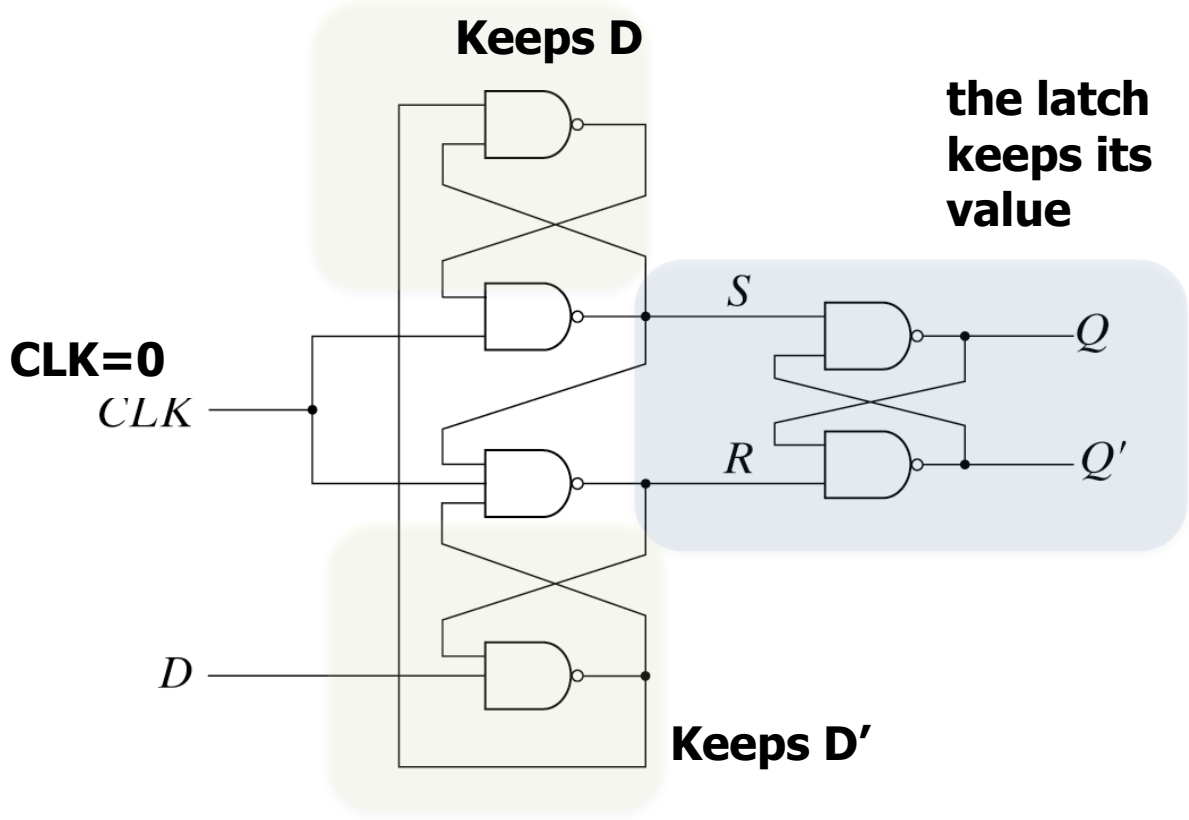

**(b) Timing diagram**

# Positive edge triggered D-Flip-flops



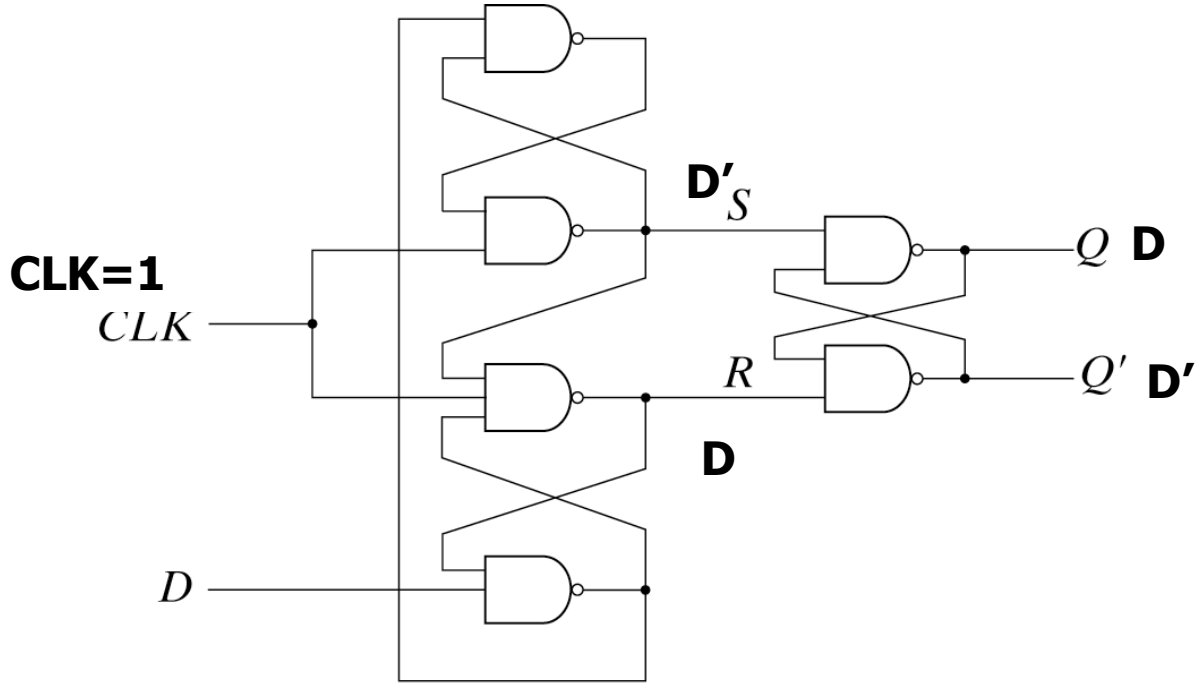

*D*-Type Positive-Edge-Triggered Flip-Flop

# Positive edge triggered D-Flip-flops




*D*-Type Positive-Edge-Triggered Flip-Flop

# Positive edge triggered D-Flip-flops

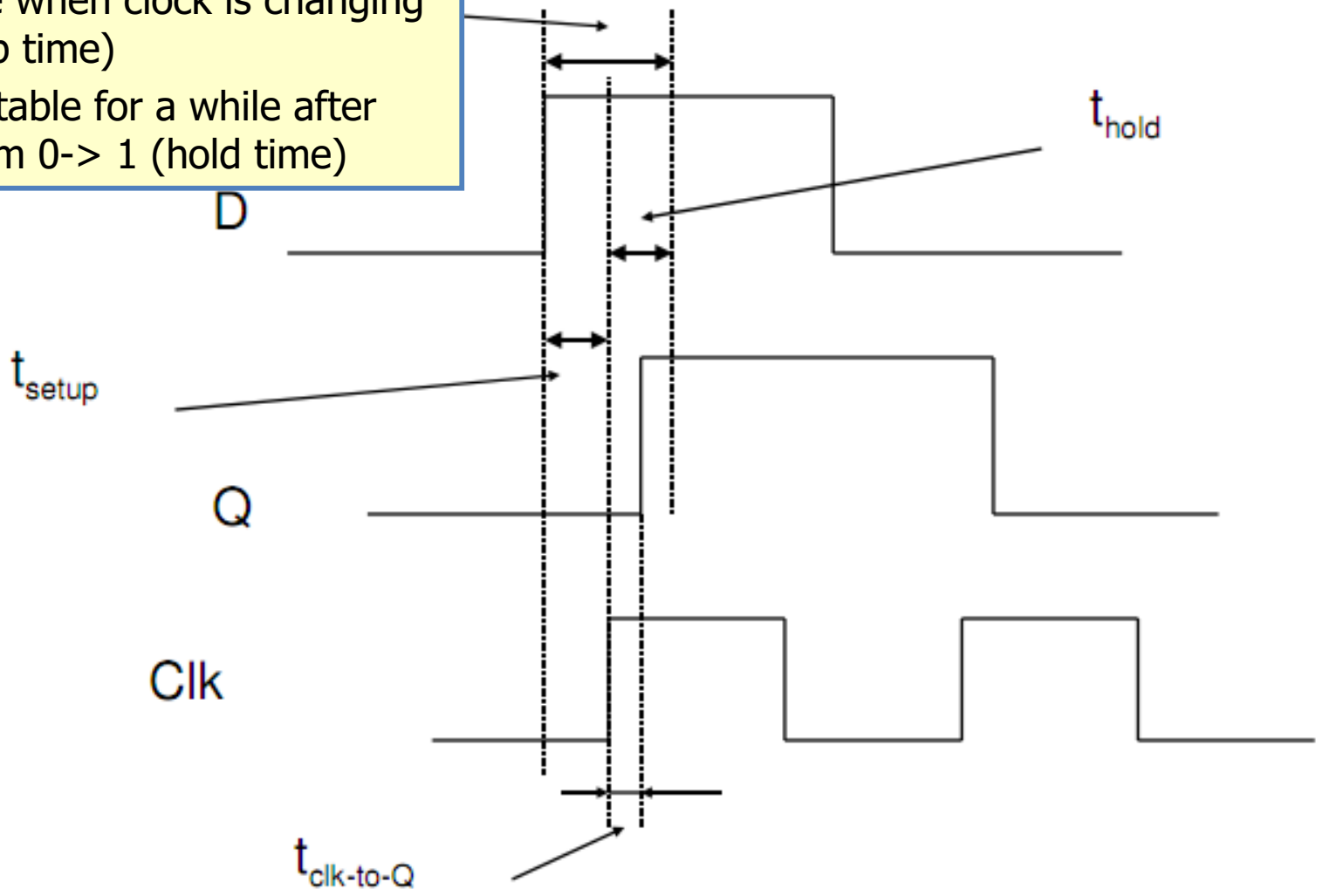



*D*-Type Positive-Edge-Triggered Flip-Flop

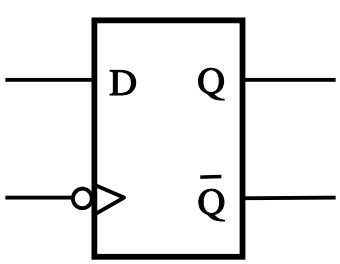
# Setup & Holdtime

D must be stable when clock is changing from 0->1 (setup time)

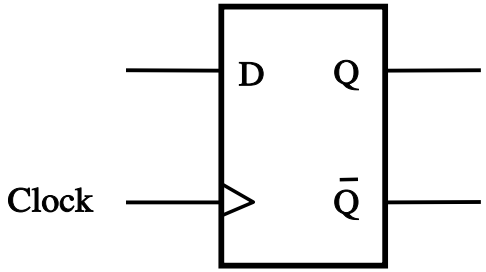
D must remain stable for a while after clock change from 0-> 1 (hold time)



# D Flip-flop

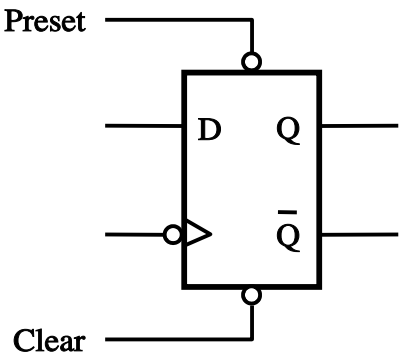


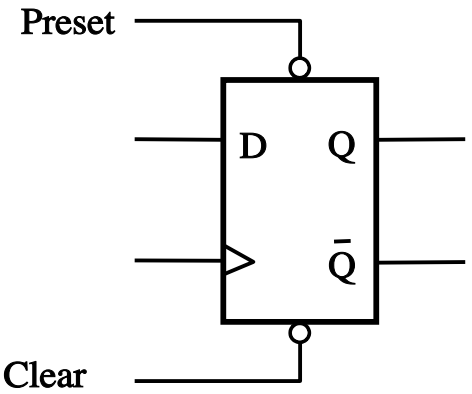

**Negative-edge DFF**




**Positive-edge**

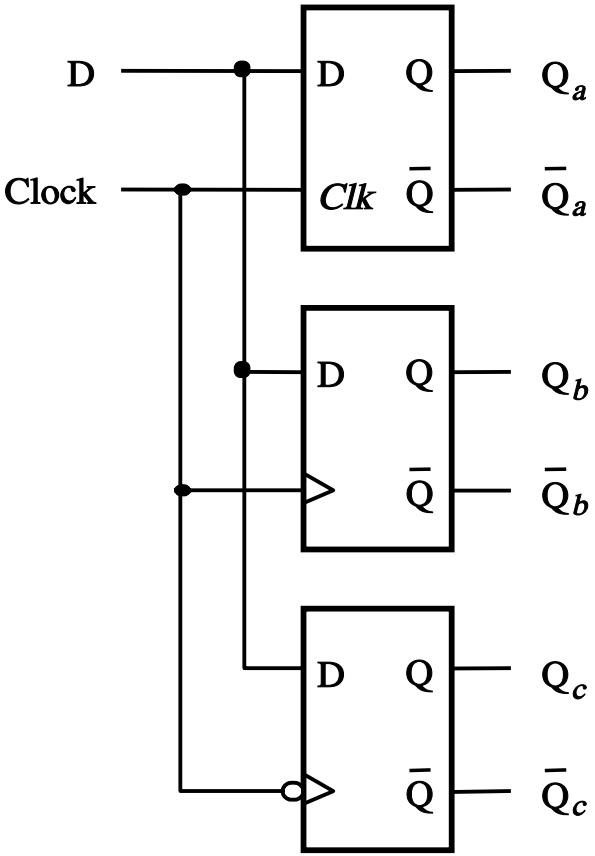
**Preset and clear:**
- **active low;**
- **not 0 at same time**

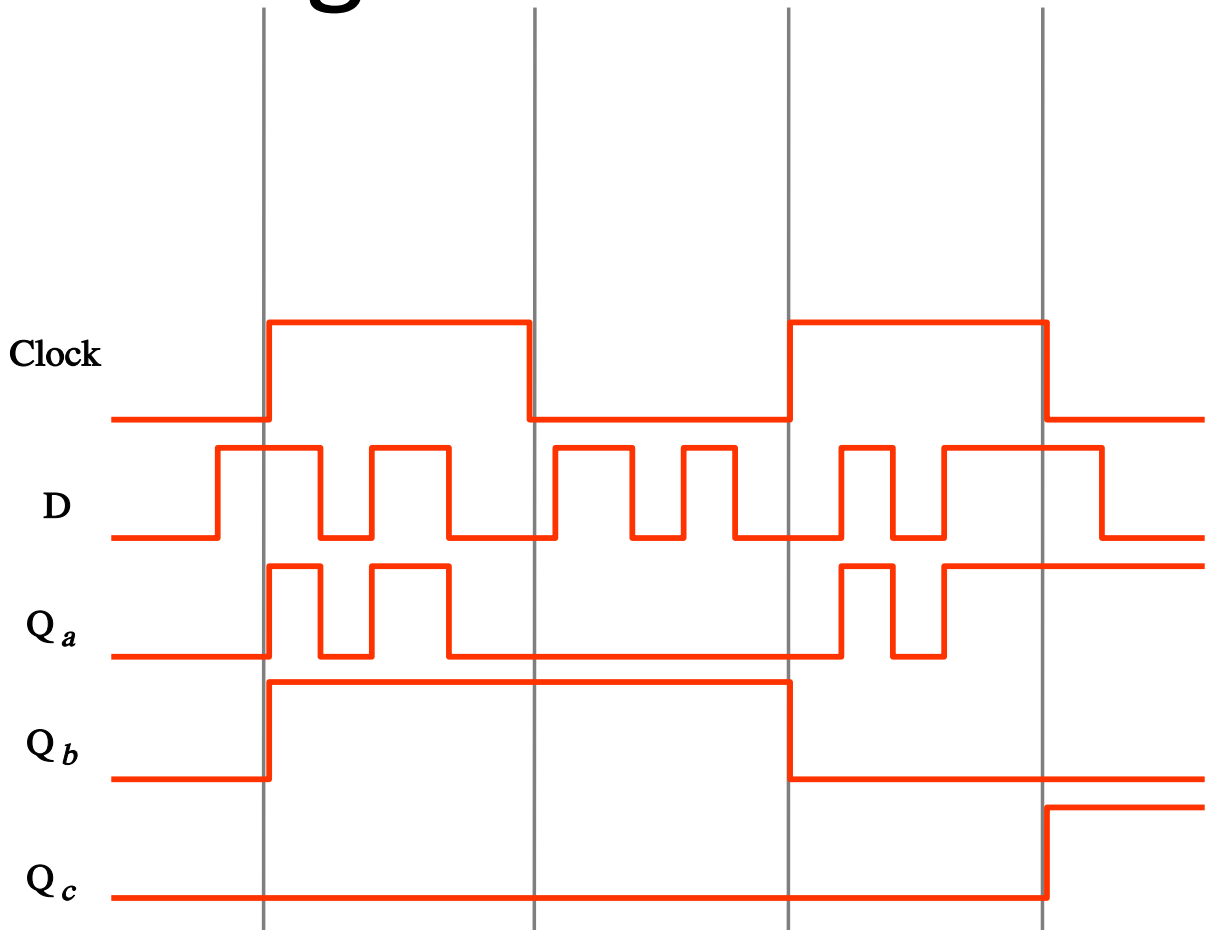


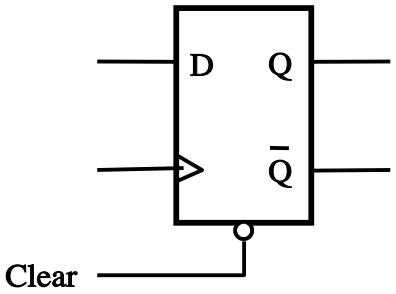

**with preset and Clear**

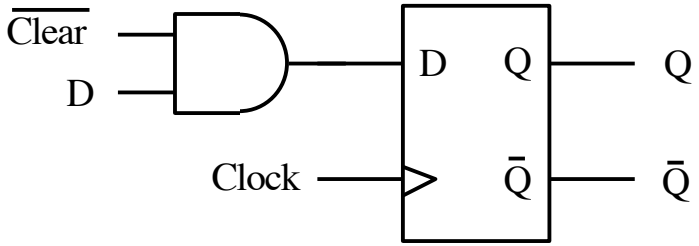
# Latch vs. Positive triggered FF vs. Negative trigered FF

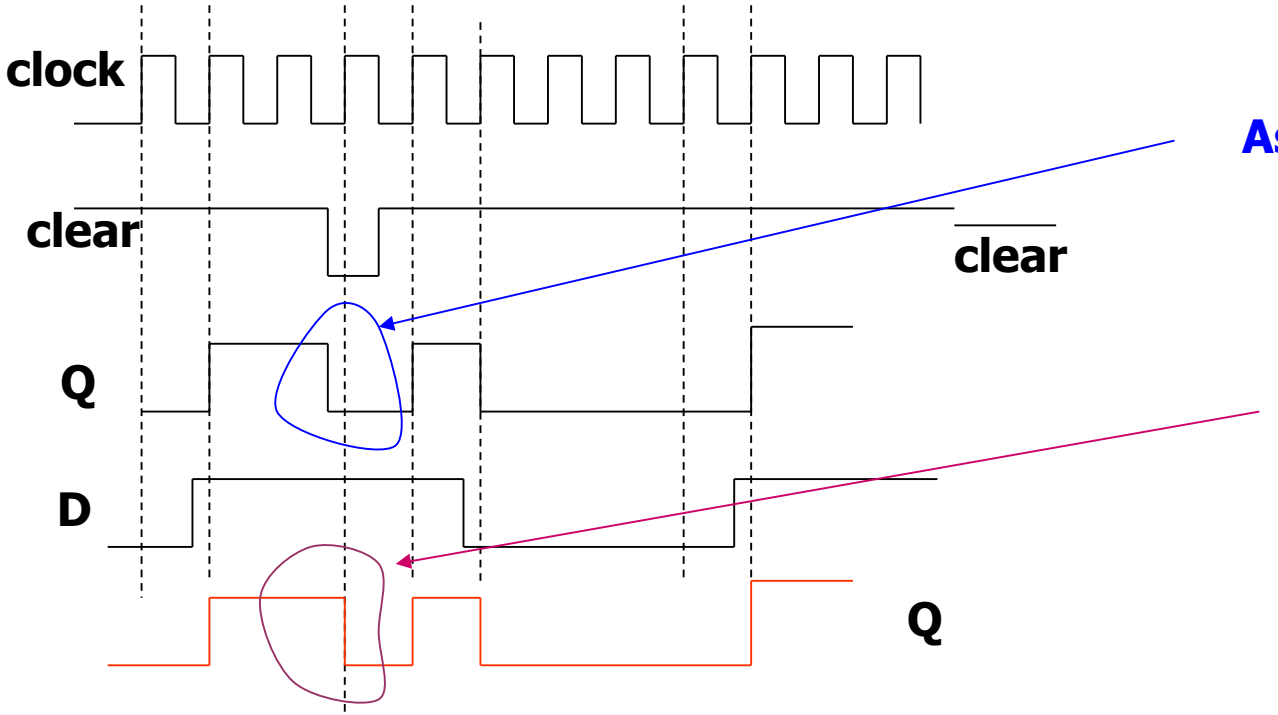


# Reset of D flip-flop



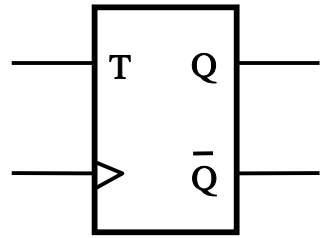

**Asynchronous reset**

**Synchronous reset**

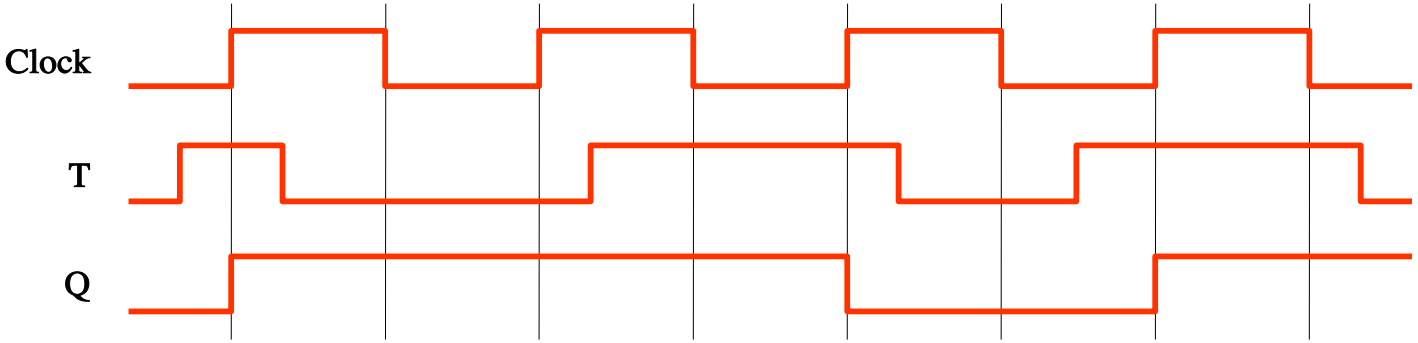
# T Flip-flop

| T | Q(t + 1) |
|---|---|
| 0 | Q(t) |
| 1 | $\overline{Q}(t)$ |

(b) Truth table




(c) Graphical symbol

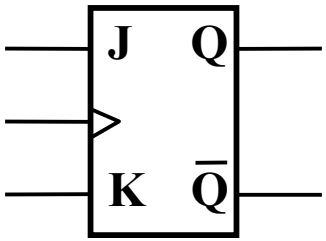



(d) Timing diagram

# JK Flip-flop

| J | K | Q(t+1) |
|---|---|---|
| 0 | 0 | Q(t) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}(t)$ |

**(b) Truth table**




**(c) Graphical symbol**

# Quiz 4-1

http://m.socrative.com/student/#joinRoom

**room number: 713113**

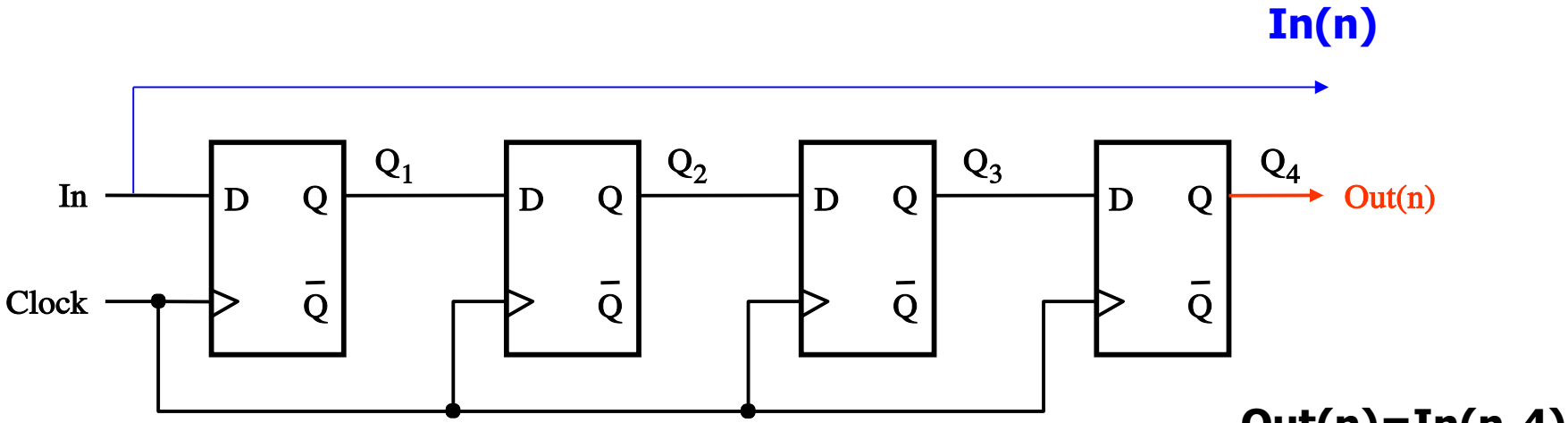
- Q2: What happens to the Q and Q' outputs of an SR-latch when S-R are:
  - 00
  - 01
  - 10
  - 11

# Registers,
# Shift Registers,
# Counters

# Register

- A register contains **N** flip-flops.
- A flip-flop stores 1-bit information; a register stores N-bit information.
- Usually a common clock is used for each flip-flop in the register.

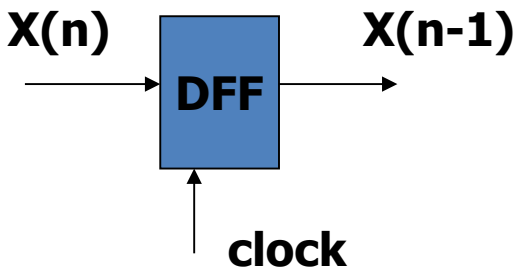
# Shift register

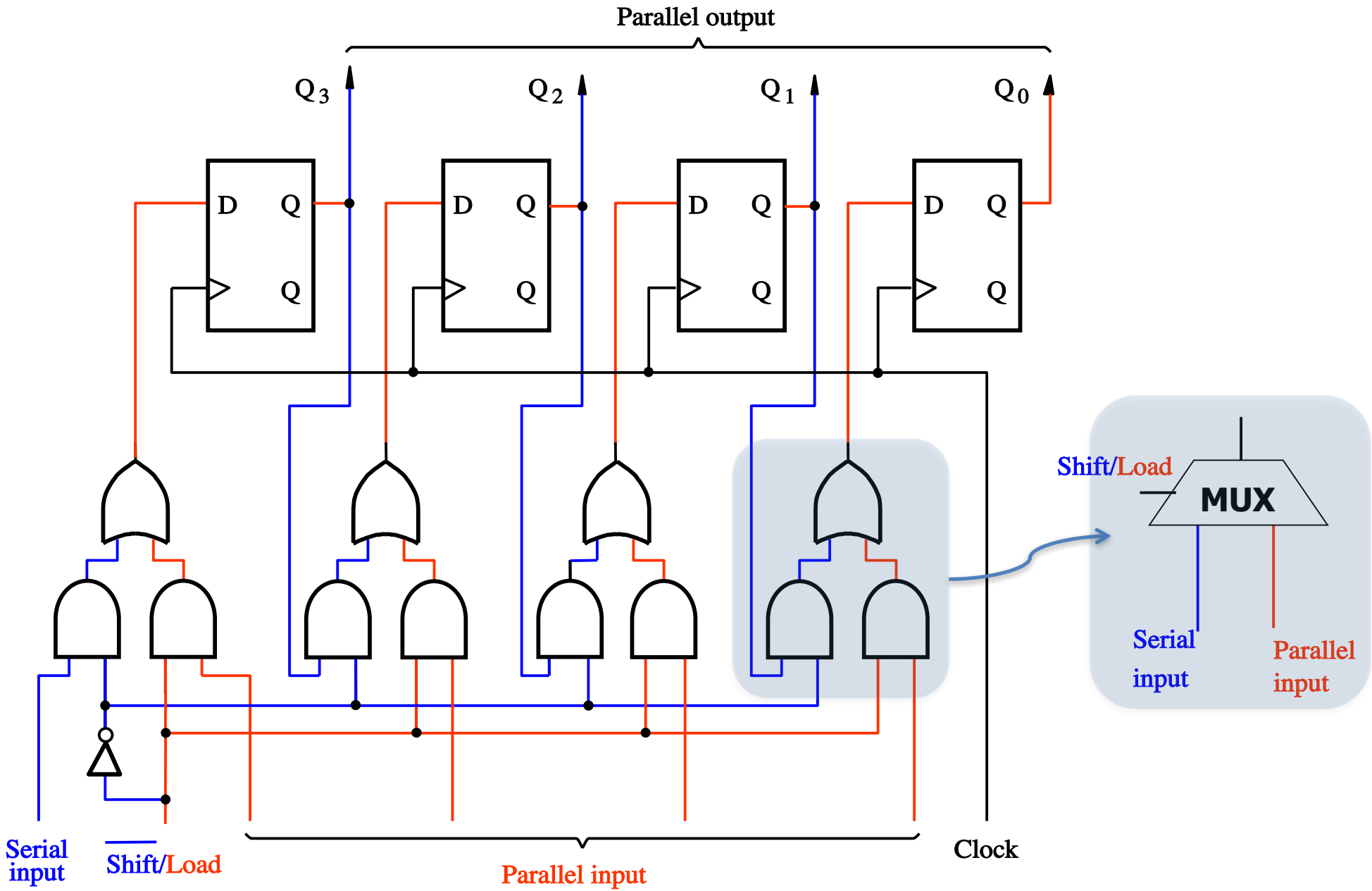



| | In | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ = Out |
|---|---|---|---|---|---|
| $t_0$ | 1 | 0 | 0 | 0 | 0 |
| $t_1$ | 0 | 1 | 0 | 0 | 0 |
| $t_2$ | 1 | 0 | 1 | 0 | 0 |
| $t_3$ | 1 | 1 | 0 | 1 | 0 |
| $t_4$ | 1 | 1 | 1 | 0 | 1 |
| $t_5$ | 0 | 1 | 1 | 1 | 0 |
| $t_6$ | 0 | 0 | 1 | 1 | 1 |
| $t_7$ | 0 | 0 | 0 | 1 | 1 |

Applications:

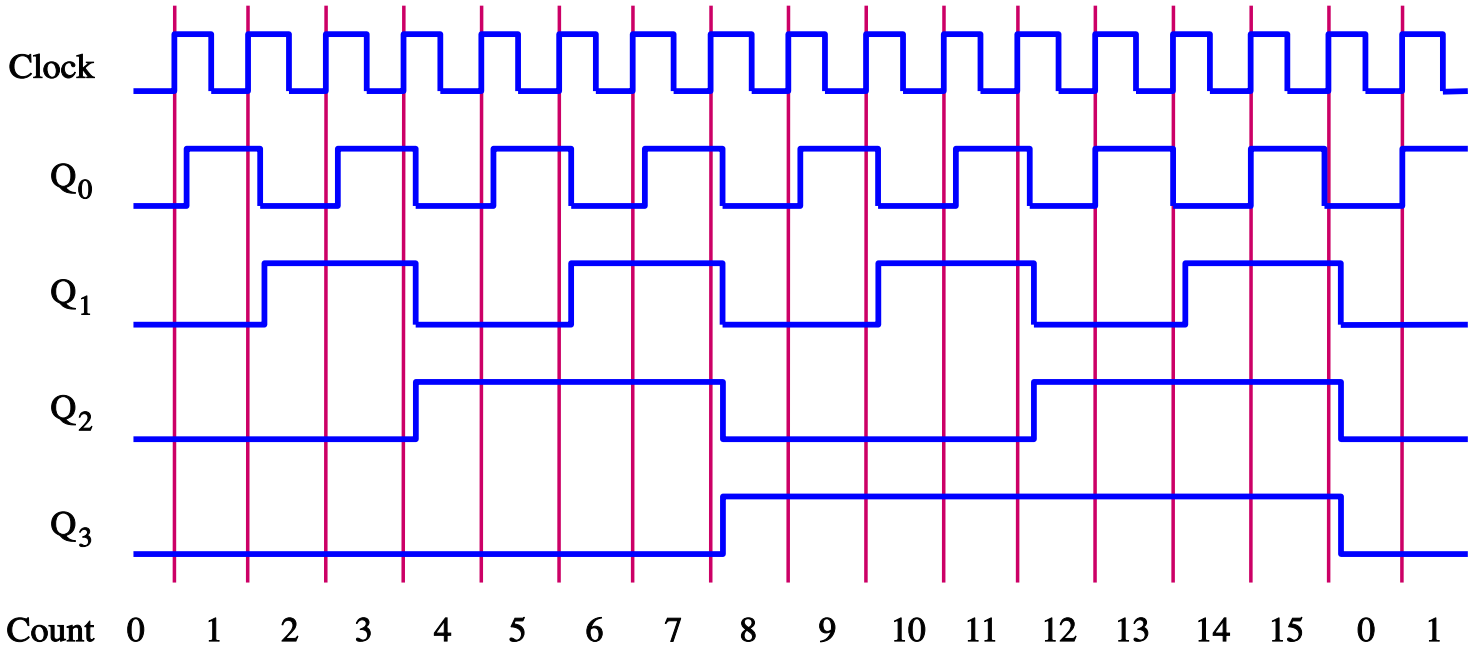
1) division by 2 ←→ shift right by 1bit
2) implement delay elements in DSP

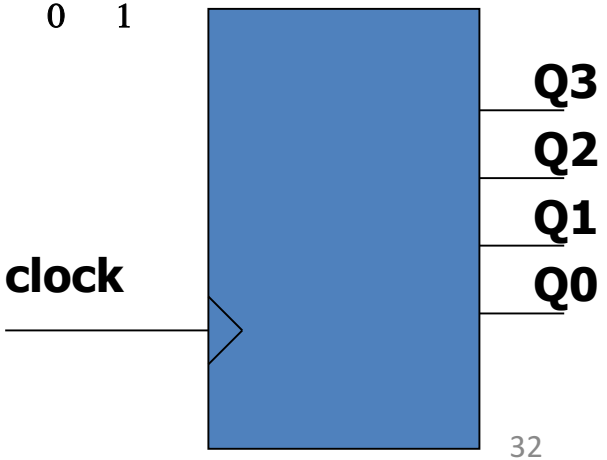




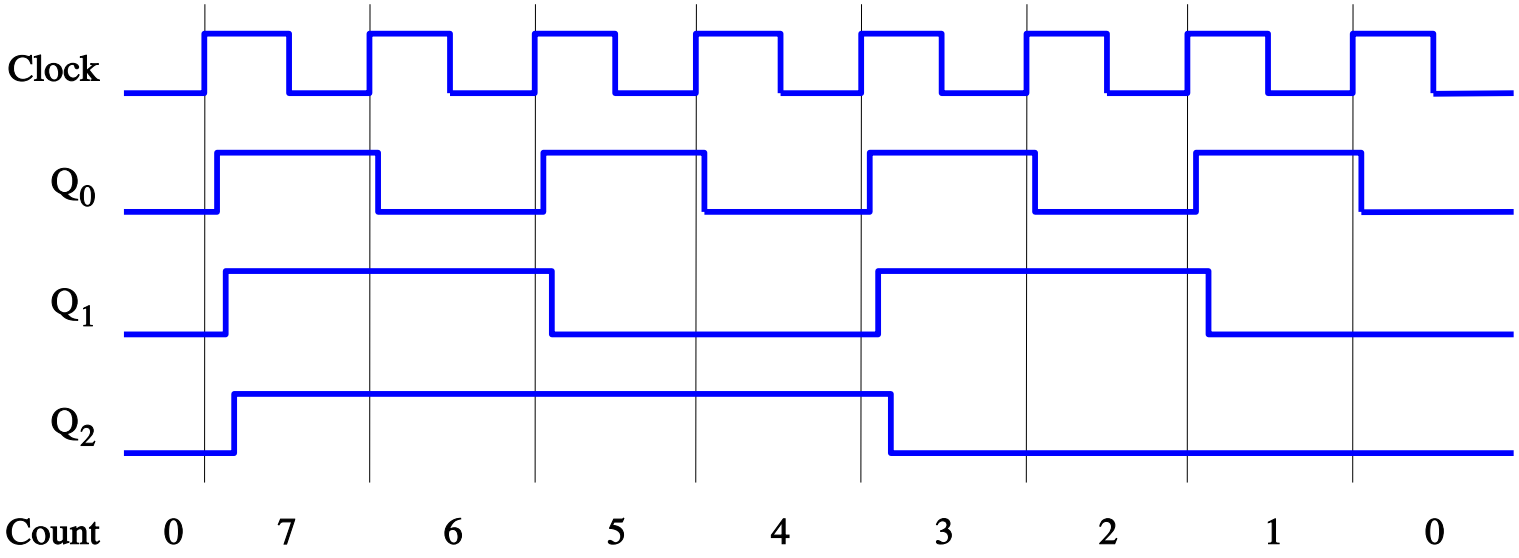

# Parallel-shift register

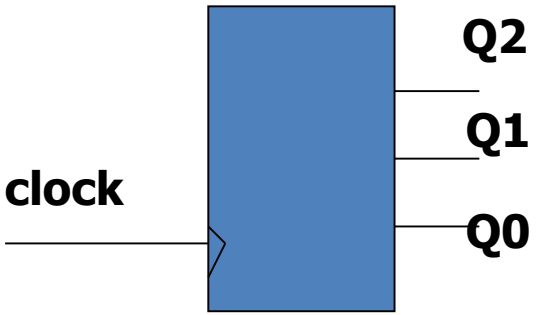
# Up-Counter


Clock
$Q_0$
$Q_1$
$Q_2$
$Q_3$
Count 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1

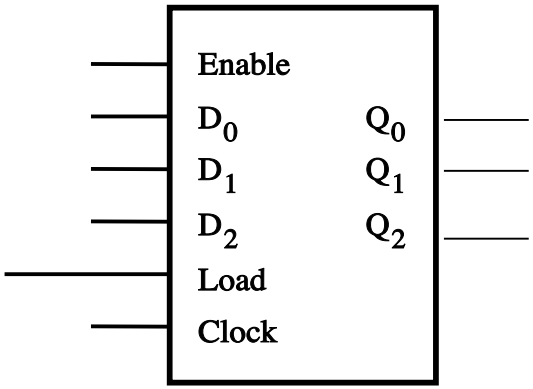

clock
Q3
Q2
Q1
Q0

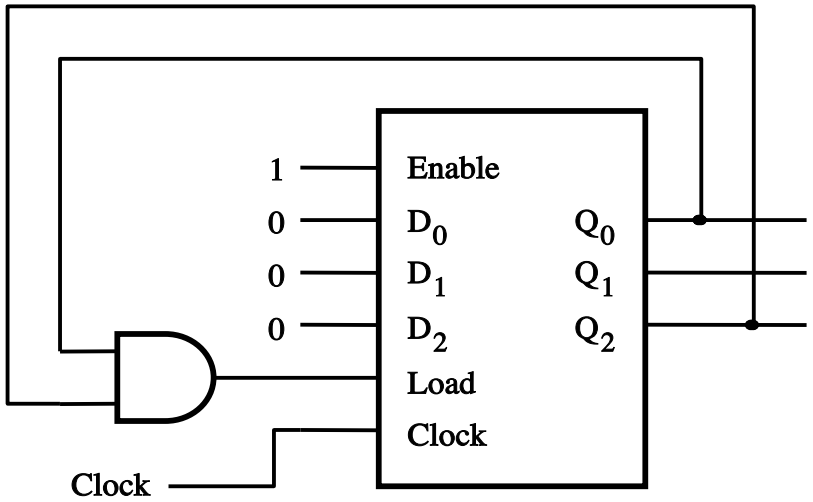
# Down-counter

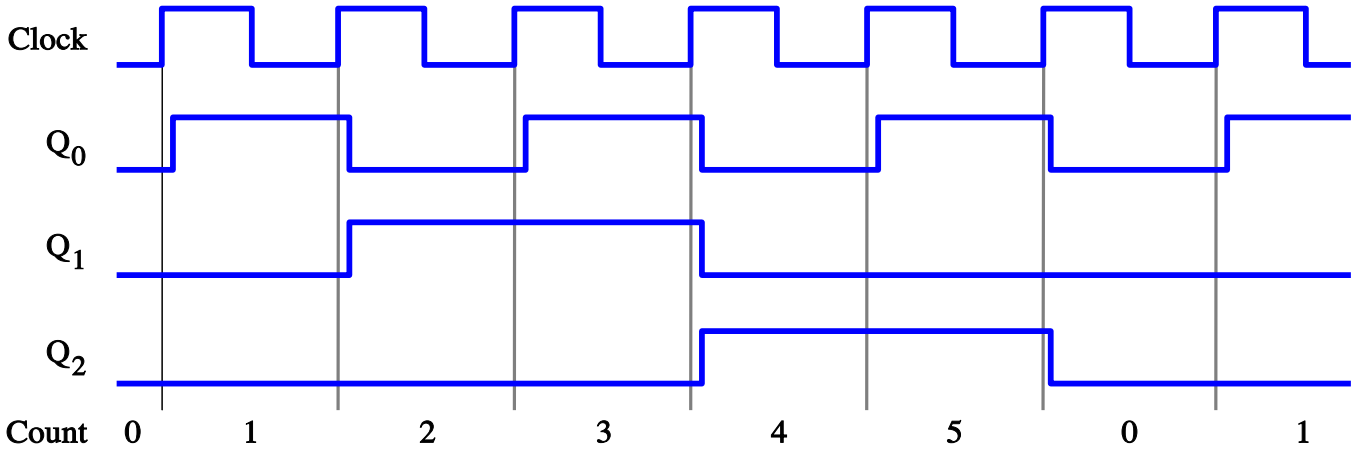


# Counter with parallel load



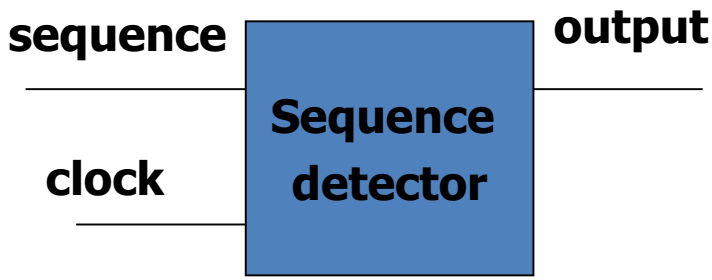



# Design of Sequential circuits

# Example of sequential circuit

- Sequence detector
  - To detect two consecutive 1s in an input sequence.
  - Output will indicate if there are two immediate 1s.




| Clockcycle: | $t_0$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w$: | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| $z$: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

# General form of a sequential circuit

Realized using combinational logic and flip-flops

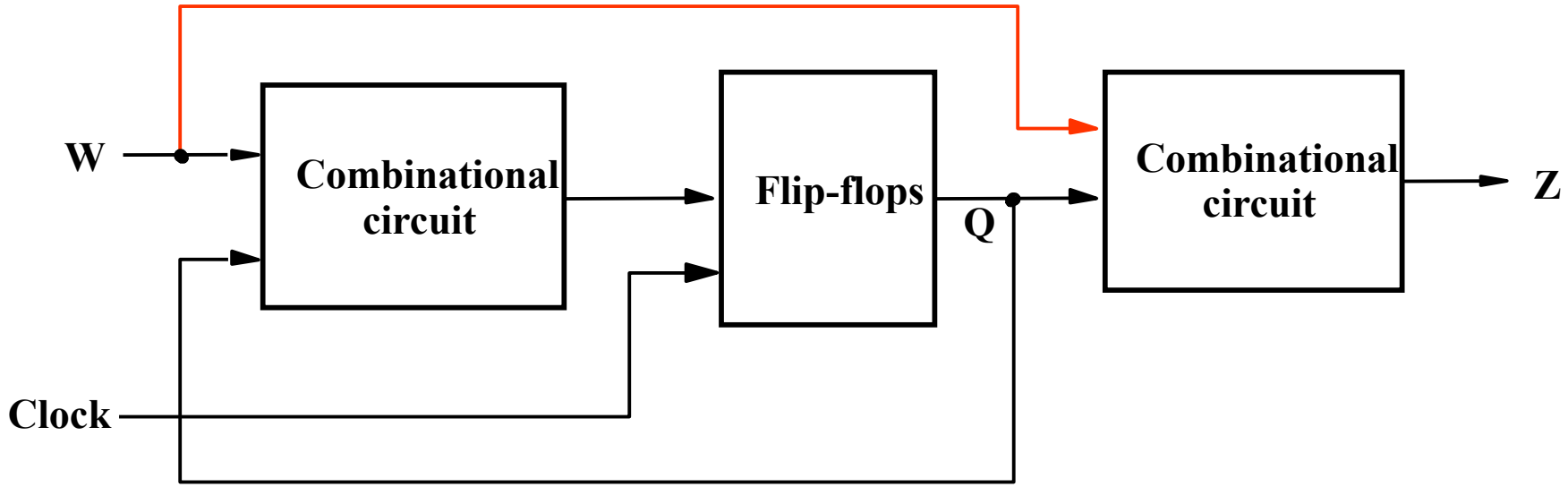
**Primary inputs: w**

**Outputs: z**

**State: Q**

**Moore FSMs:** outputs depend only on the state

**Mealy FSMs:** outputs depend on both state and primary inputs



# Manual design steps (1): specifications

Specifications:

1. the circuit has one input, w, and one output, z.
2. all changes in the circuit occur on the positive edge of a clock signal.
3. the output z is equal to 1 if during two immediately preceding clock cycles the input w was equal to 1. otherwise, the value of z is equal to 0.

| Clockcycle: | $t_0$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w$: | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| $z$: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

# Manual design steps (2): state diagram

- *starting* state A: when power is on or *reset* signal is applied.
- As long as *w* is 0, it remains in A.
- After *w* → *1*, it moves to state B.
- Then,
  - If *w* →*0*, it moves back to state A.
  - If *w* →*1*, it moves to state C, and z=1.
- When in state C,
  - If *w* → *0*, back to state A, and z=0;
  - If *w* →*1*, remain in state C.

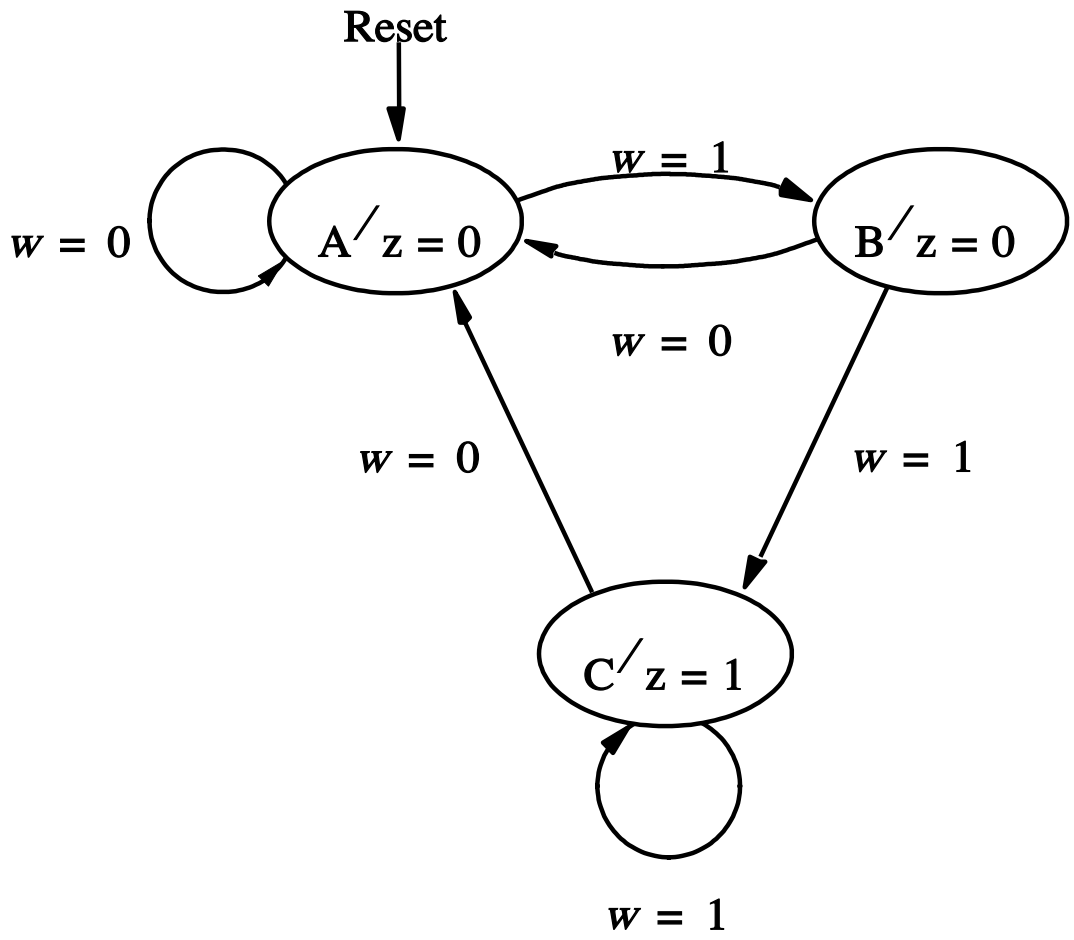
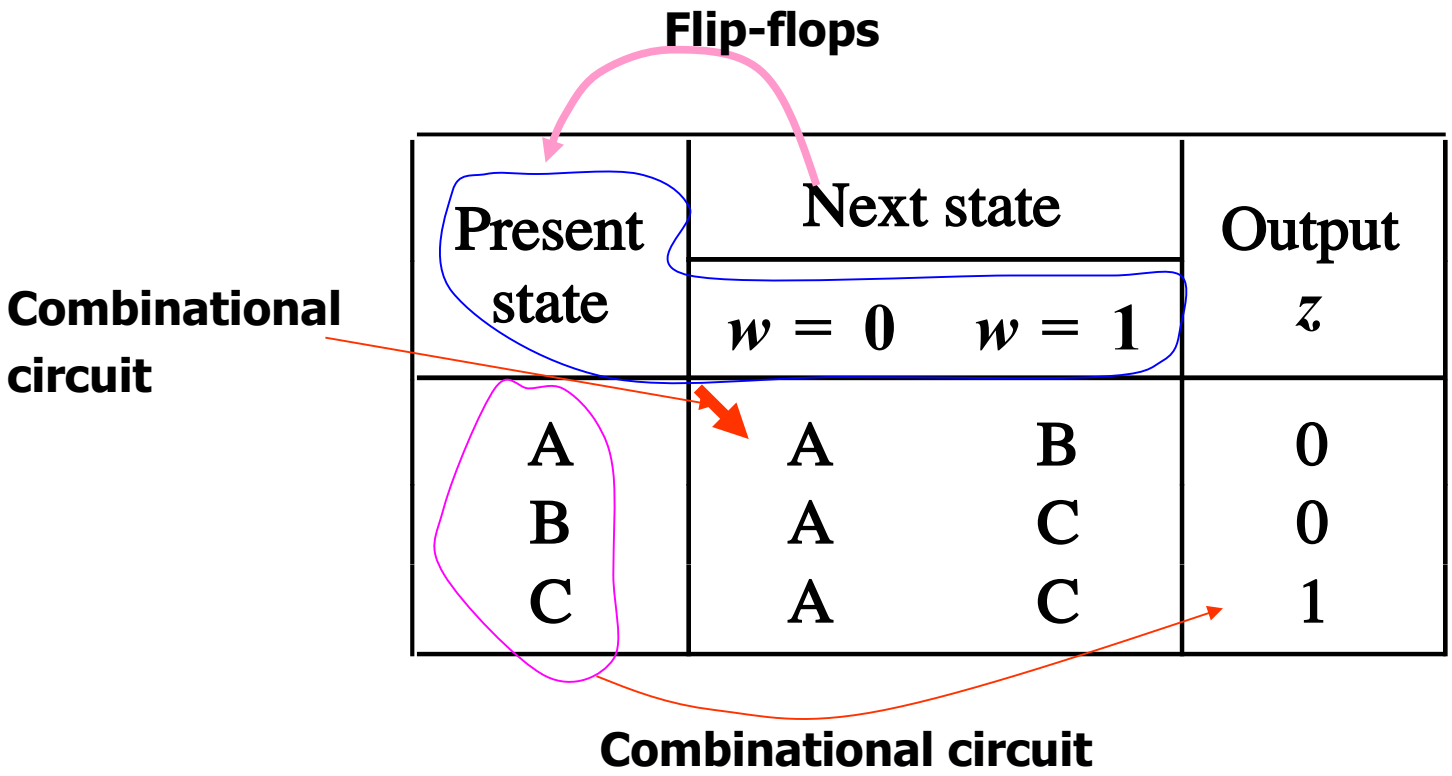



**Figure 8.3. State diagram of a simple sequential circuit.**

# Manual design steps (3): state table

**From the state diagram, we have the state table**

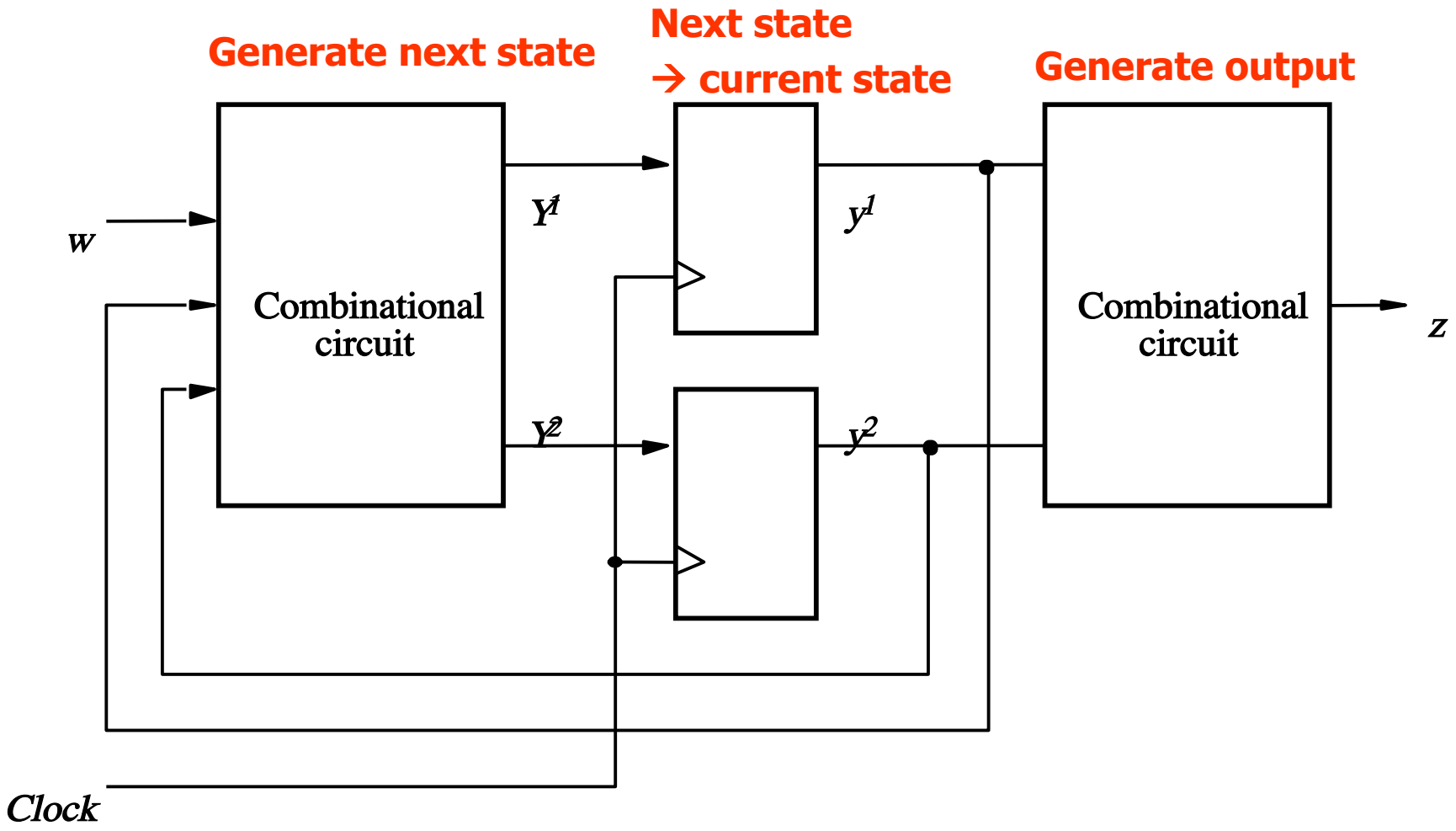


# Manual design steps (4): state assignment

**A: 00**
**B: 01**
**C: 10**

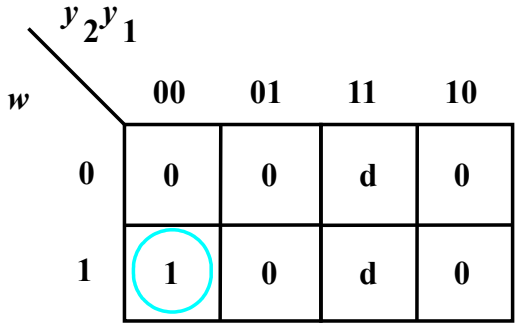
| | Present state $y_2 y_1$ | Next state $w = 0$ $Y_2 Y_1$ | Next state $w = 1$ $Y_2 Y_1$ | Output $z$ |
|---|---|---|---|---|
| A | 00 | 00 | 01 | 0 |
| B | 01 | 00 | 10 | 0 |
| C | 10 | 00 | 10 | 1 |
| | 11 | *dd* | *dd* | *d* |

# Circuit diagram

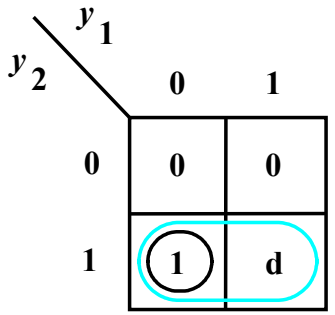


# Manual design steps (5): implementation

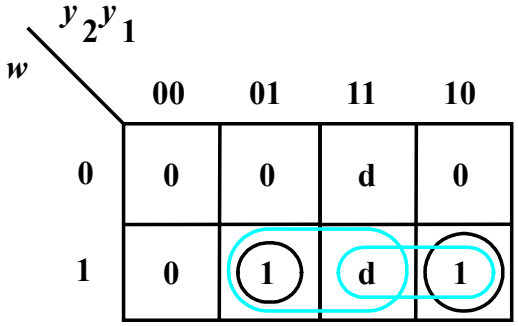
**From state-assigned table: we have the following Karnaugh maps**




$$Y_1 = w\bar{y}_1\bar{y}_2$$

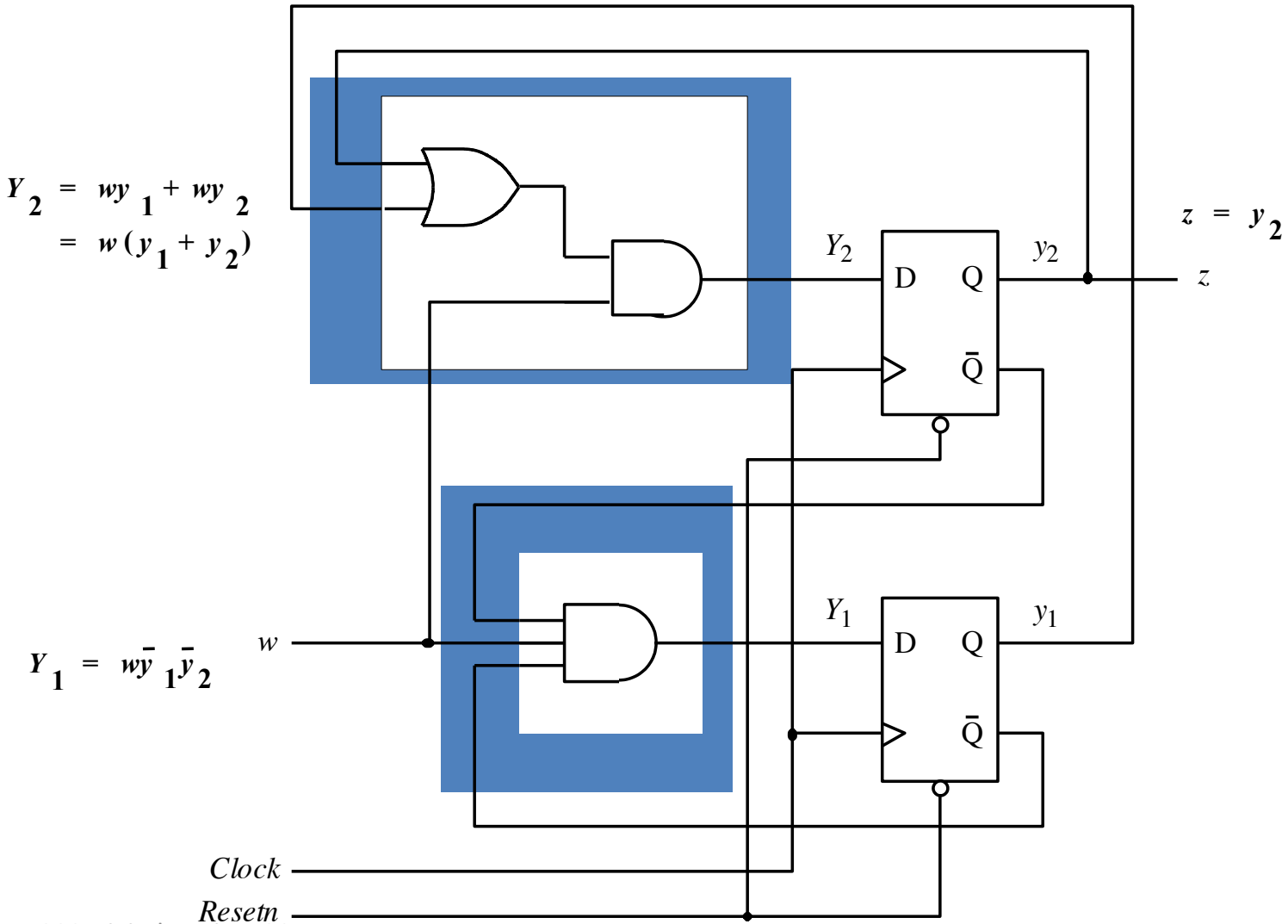



$$z = y_2$$

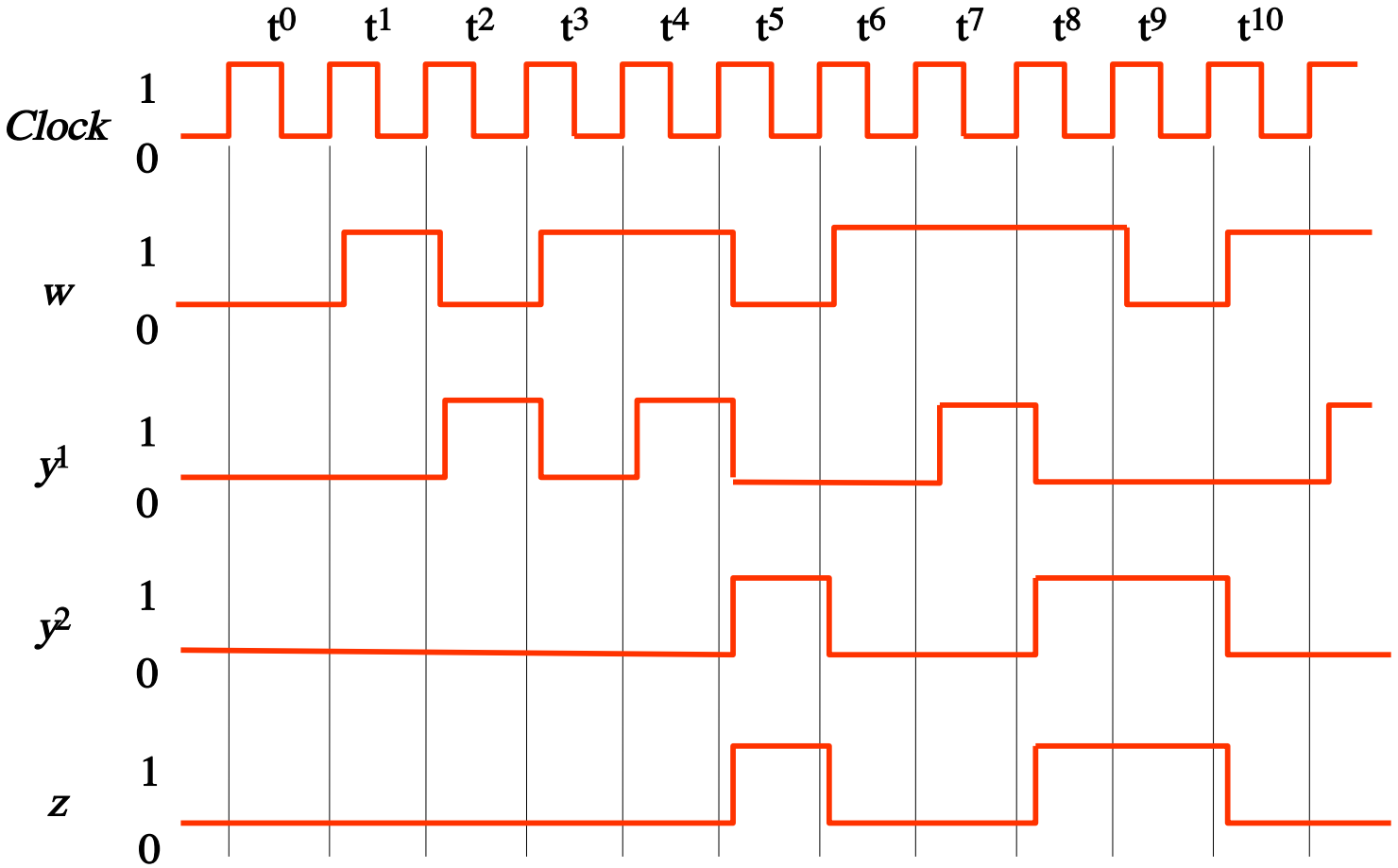



$$Y_2 = wy_1 + wy_2$$
$$= w(y_1 + y_2)$$

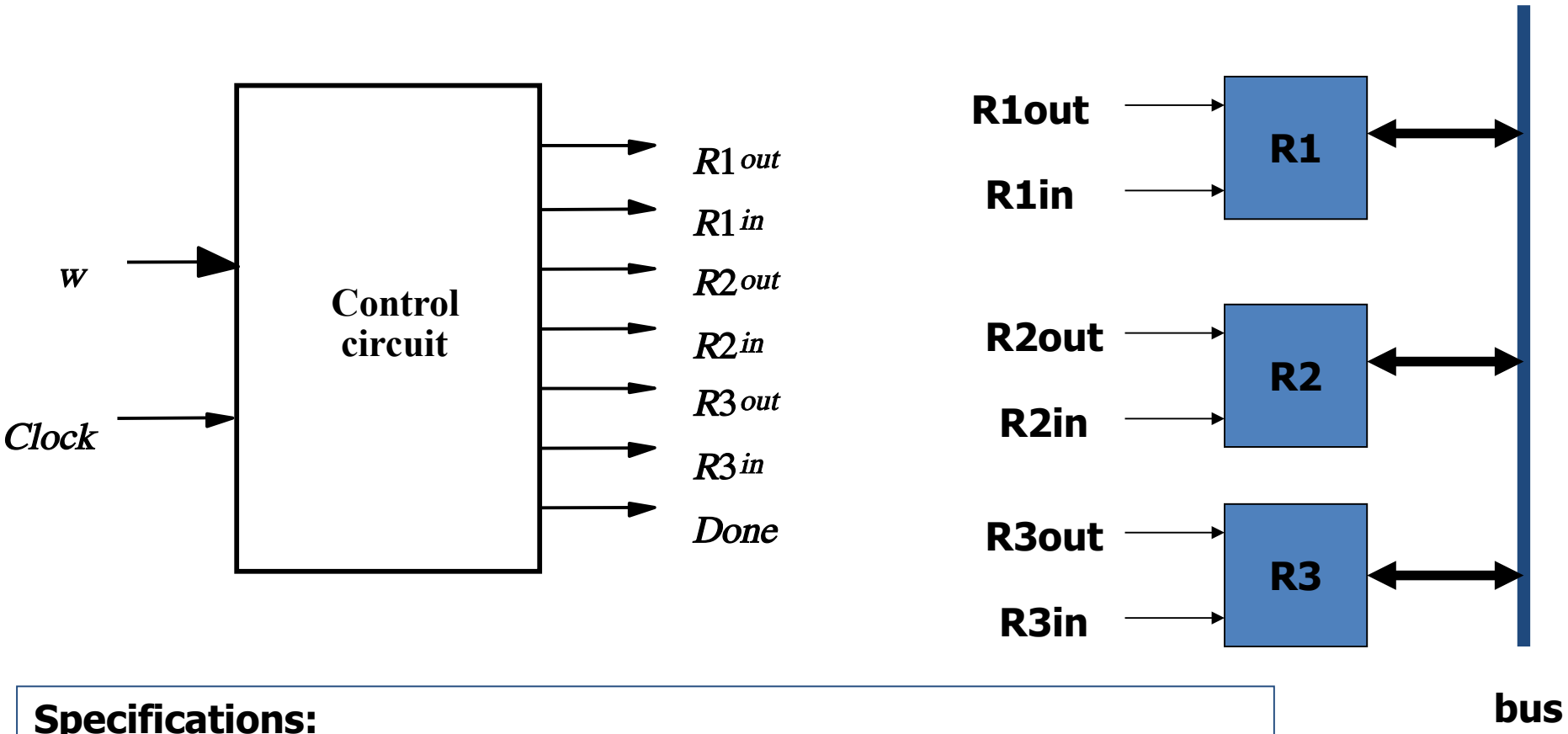
# Final implementation



# Timing diagram



# One more example



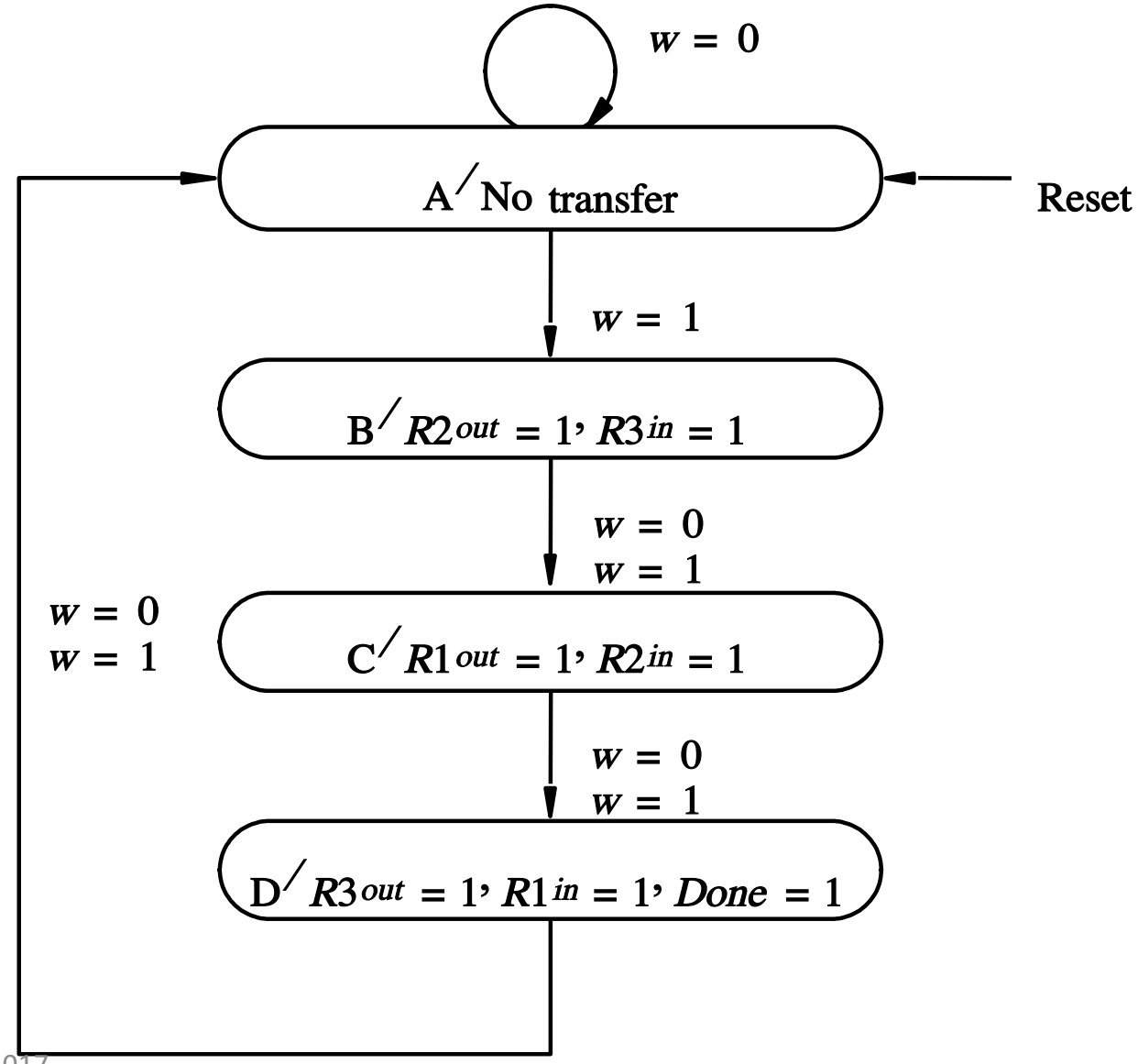

**Specifications:**

**1) After w: 0 →1, R2→R3 (R2out=1, R3in=1).**

**2) then, R1→R2 (R1out=1, R2in=1), regardless of w**

**3) then, R3→R1 (R3out=1, R1in=1), regardless of w, done=1.**

# State diagram

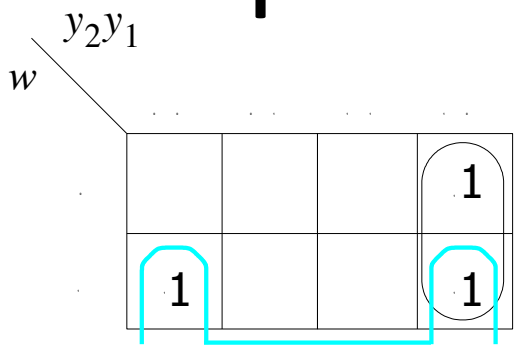


# State table

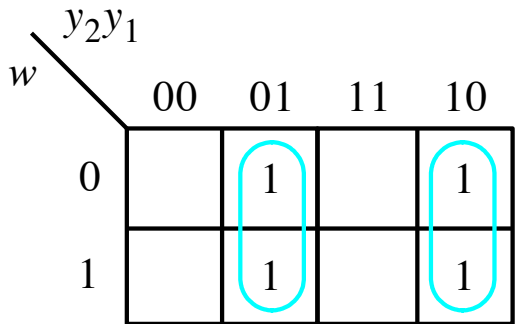
| Present state | Next state | | Outputs | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $w=0$ | $w=1$ | $R1_{out}$ | $R1_{in}$ | $R2_{out}$ | $R2_{in}$ | $R3_{out}$ | $R3_{in}$ | $Done$ |
| A | A | B | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | C | C | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| C | D | D | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| D | A | A | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

| | Present state | Nextstate | | Outputs | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $w=0$ | $w=1$ | | | | | | | |
| | $y_2y_1$ | $Y_2Y_1$ | $Y_2Y_1$ | $R1_{out}$ | $R1_{in}$ | $R2_{out}$ | $R2_{in}$ | $R3_{out}$ | $R3_{in}$ | $Done$ |
| A | 00 | 00 | 01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 01 | 10 | 10 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| C | 10 | 11 | 11 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| D | 11 | 00 | 00 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

# Expressions of next state and outputs




$$Y_1 = w\bar{y}_1 + \bar{y}_1 y_2$$



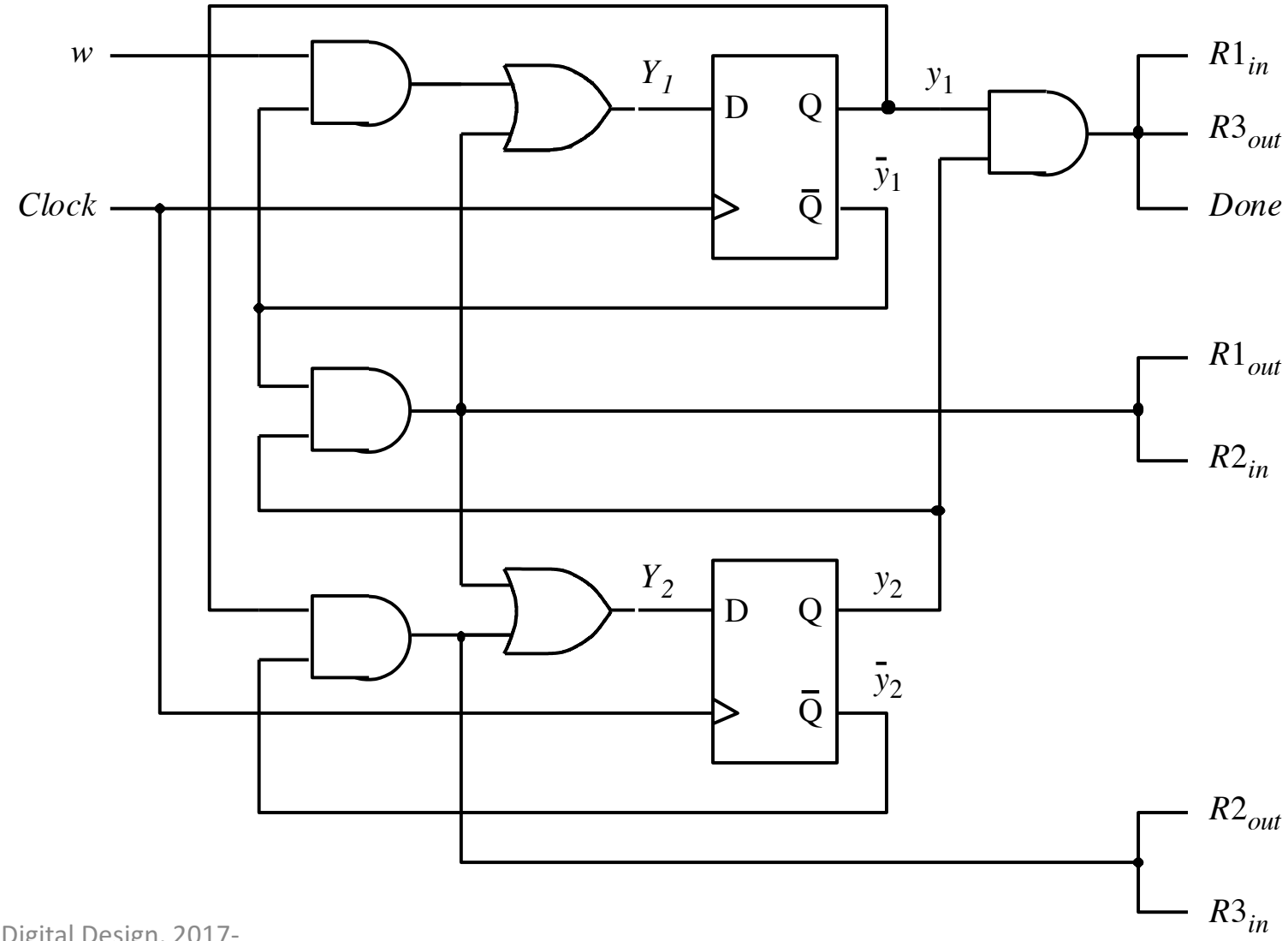

$$Y_2 = y_1\bar{y}_2 + \bar{y}_1 y_2$$

$$R1_{out} = R2_{in} = \bar{y}_1 y_2$$

$$R1_{in} = R3_{out} = Done = y1y2$$

$$R2_{out} = R3_{in} = y_1\bar{y}_2$$

# Final implementation



# Summary of design steps

- Specifications
- Derive the states, and create a state diagram
- Create a state table
- (State minimization)
- State assignment
- Choose flip-flops
- Derive logic expressions for state and outputs
- Implementation

# Quiz 4-2

http://m.socrative.com/student/#joinRoom

**room number: 713113**

- Q1: A D-flipflop is edge triggered when input is stored when
  - Clock =1
  - Clock is changing from 0 to 1
- Q2: An (active when '1') asynchronous reset resets the value of a D-flip-flop to zero when:
  - Reset=1 and clock =1
  - Reset=1
  - Reset=1, and D=0
  - Reset=1, clock=1, and D=0
- Q3: A sequential circuit that the output depends both at the current state and its inputs is
  - A Mealy
  - A Moore

# Summary of Lecture 6

- Introduction to sequential circuits
- Latches
- flip-flops
- Registers, Shift-registers, Counters
- Manual design steps
- Two design examples

- Book (complimentary to the slides):
  - Chapters 14, 27.1, 27.2

- Next Lecture 7:
  - Sequential circuits **VHDL**