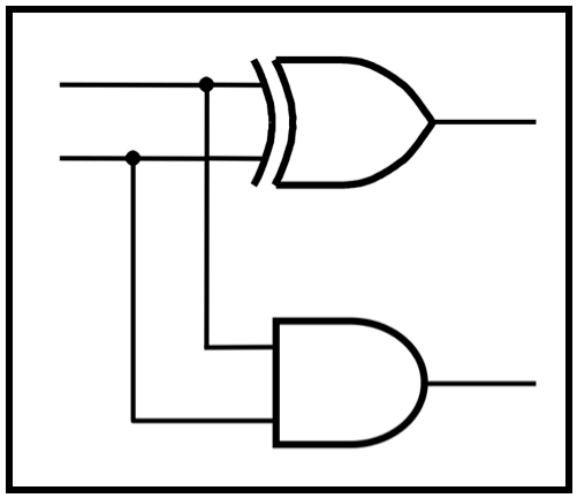


CprE 281: Digital Logic

Instructor: Alexander Stoytchev

http://www.ece.iastate.edu/~alexs/classes/

Registers

CprE 281: Digital Logic
Iowa State University, Ames, IA

Administrative Stuff

- **The second midterm is next Friday.**
- **Homework 8 is due next Monday.**

Administrative Stuff

- **Midterm Exam #2**
- **When: Friday October 28 @ 4pm.**
- **Where: This classroom**
- **What: Chapters 1, 2, 3, 4 and 5.1-5.8**
- **The exam will be open book and open notes (you can bring up to 3 pages of handwritten notes).**

Midterm 2: Format

- **The exam will be out of 130 points**
- **You need 95 points to get an A**
- **It will be great if you can score more than 100 points.**
  - **but you can't roll over your extra points ☹**

Midterm 2: Topics

- Binary Numbers and Hexadecimal Numbers
- 1's complement and 2's complement representation
- Addition and subtraction of binary numbers
- Circuits for adders and fast adders

- Single and Double precision IEEE floating point formats
- Converting a real number to the IEEE format
- Converting a floating point number to base 10

- Multiplexers (circuits and function)
- Synthesis of logic functions using multiplexers
- Shannon's Expansion Theorem

Midterm 2: Topics

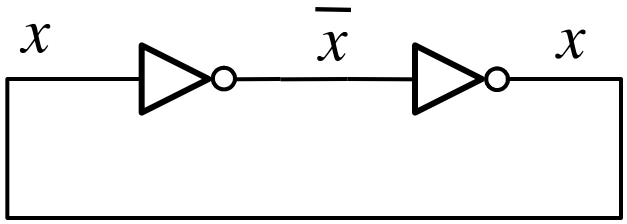
- Decoders (circuits and function)
- Demultiplexers
- Encoders (binary and priority)
- Code Converters
- K-maps for 2, 3, and 4 variables

- Synthesis of logic circuits using adders, multiplexers, encoders, decoders, and basic logic gates
- Synthesis of logic circuits given constraints on the available building blocks that you can use

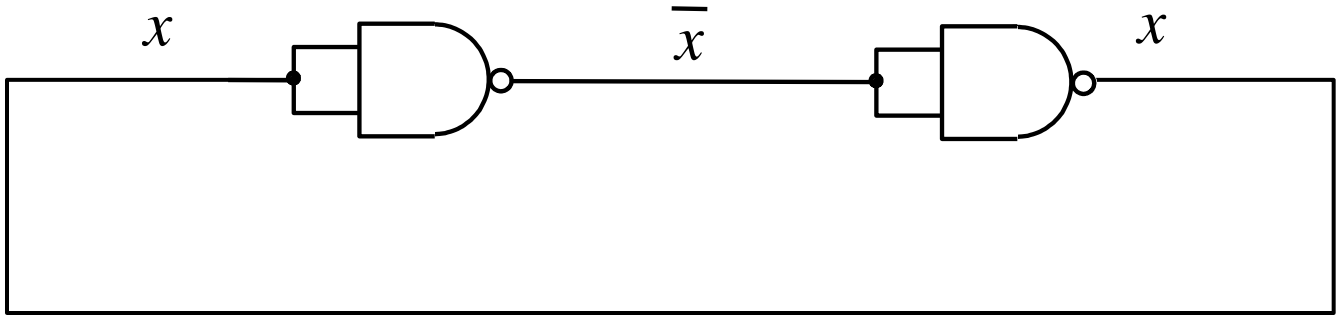
- Latches (circuits, behavior, timing diagrams)
- Flip-Flops (circuits, behavior, timing diagrams)
- Registers and Register Files

Review of Flip-Flops

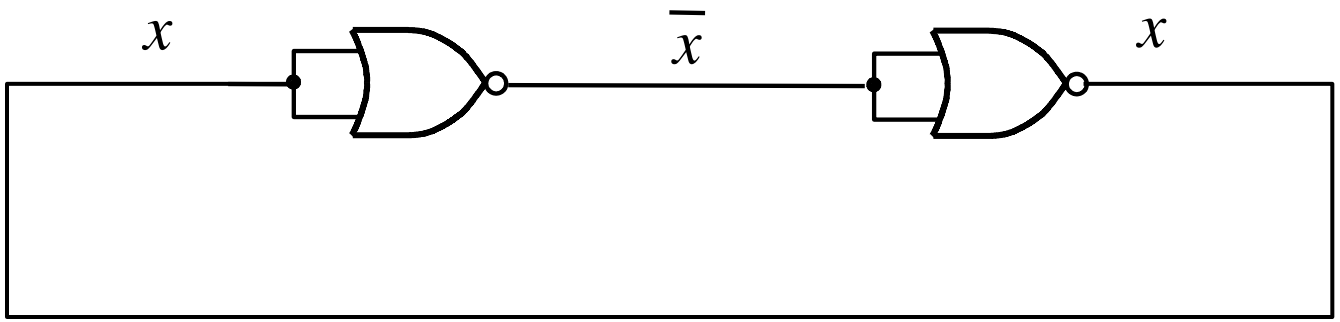
A simple memory element with NOT Gates



A simple memory element with NAND Gates

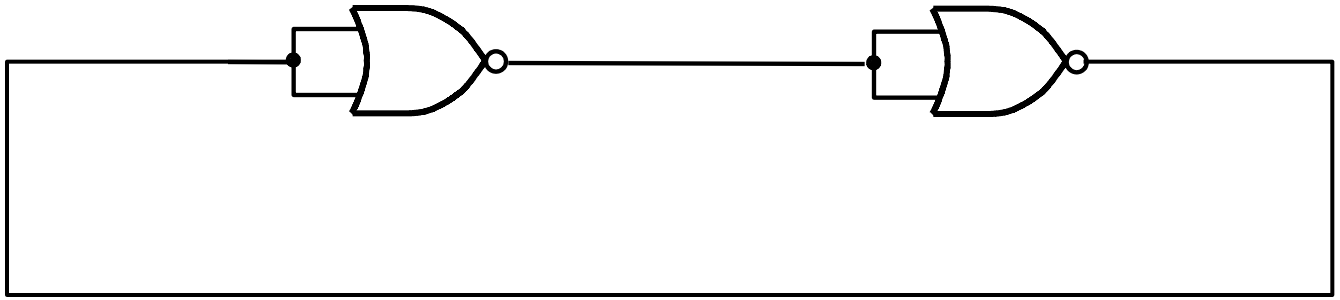


A simple memory element with NOR Gates

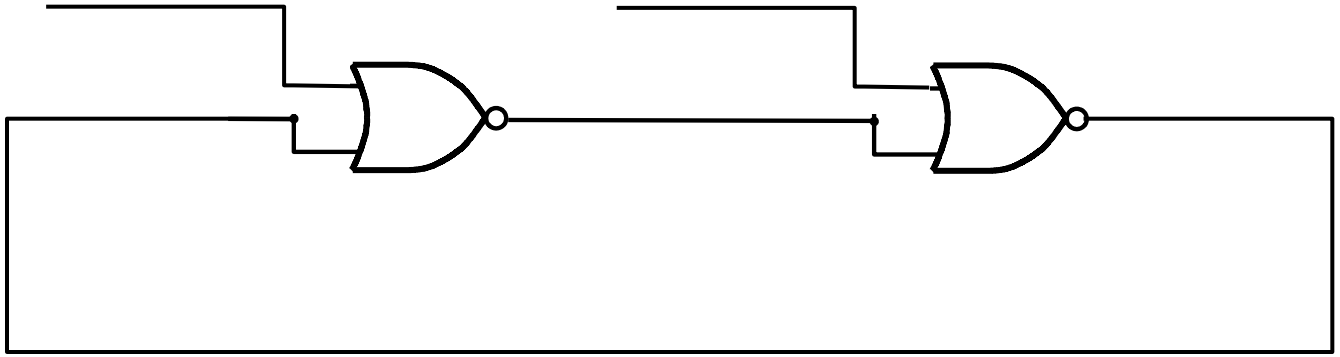


Basic Latch

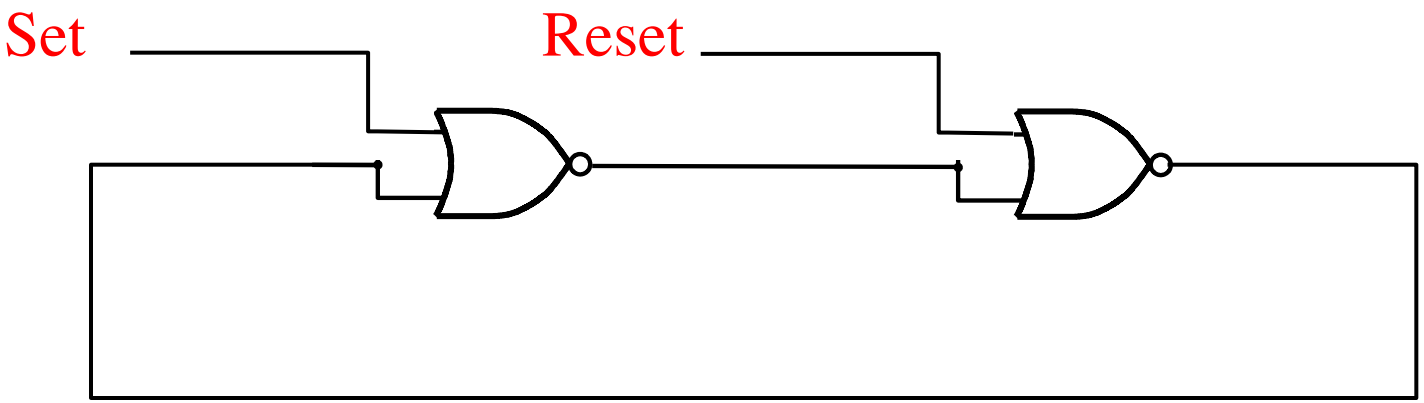
A simple memory element with NOR Gates



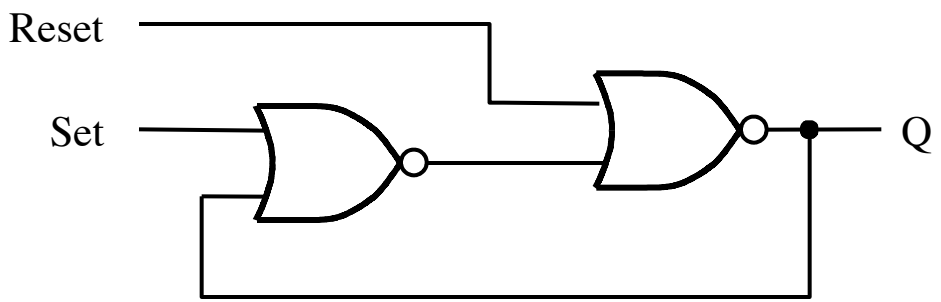
A simple memory element with NOR Gates



A simple memory element with NOR Gates

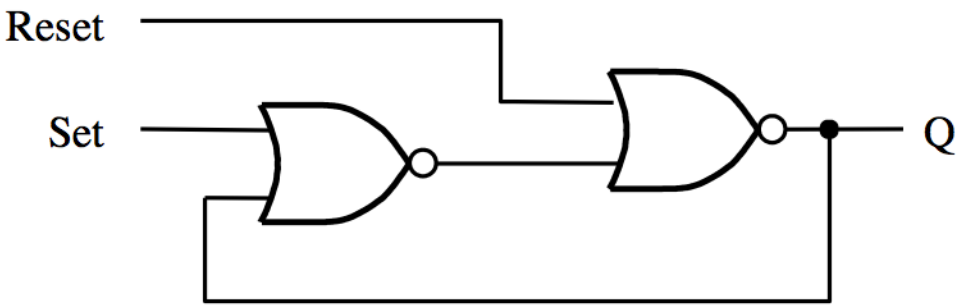


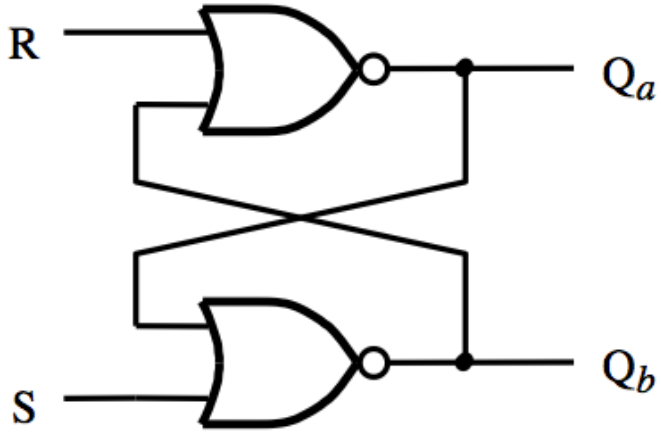
A memory element with NOR gates



[Figure 5.3 from the textbook]

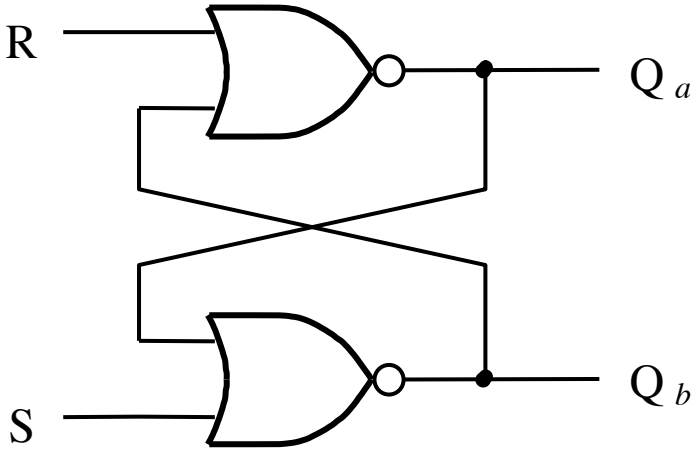
Two Different Ways to Draw the Same Circuit





[Figure 5.3 & 5.4 from the textbook]

SR Latch: Circuit and Truth Table



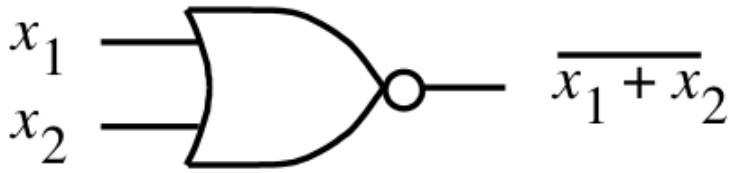
(a) Circuit

S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	(Undesirable)

(b) Truth table

[Figure 5.4a,b from the textbook]

NOR Gate

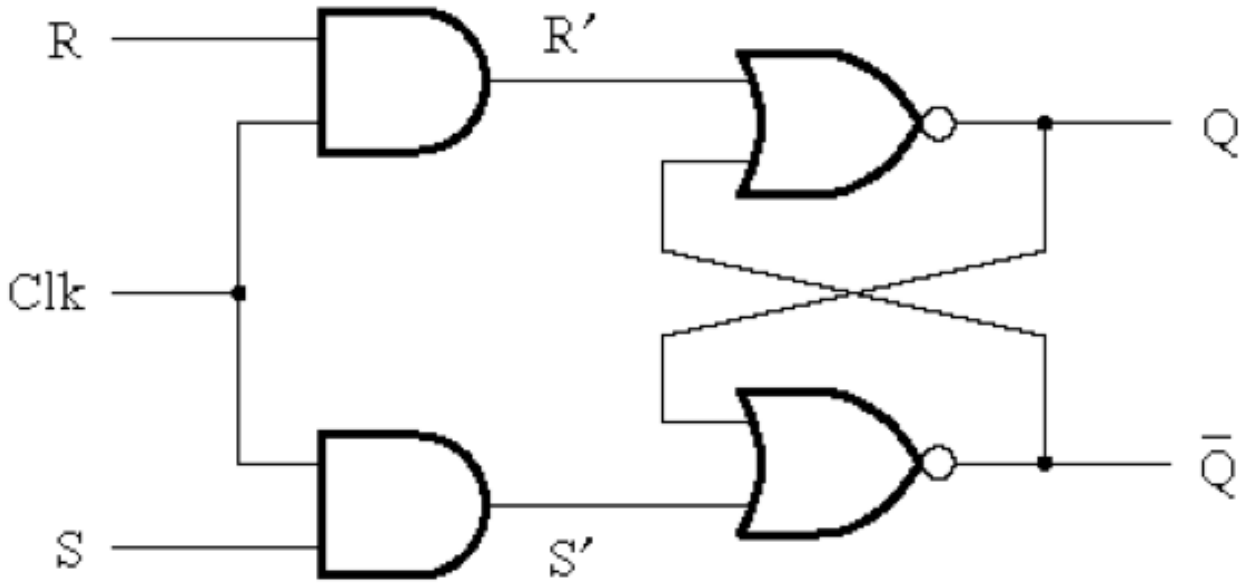


NOR Gate Truth table

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

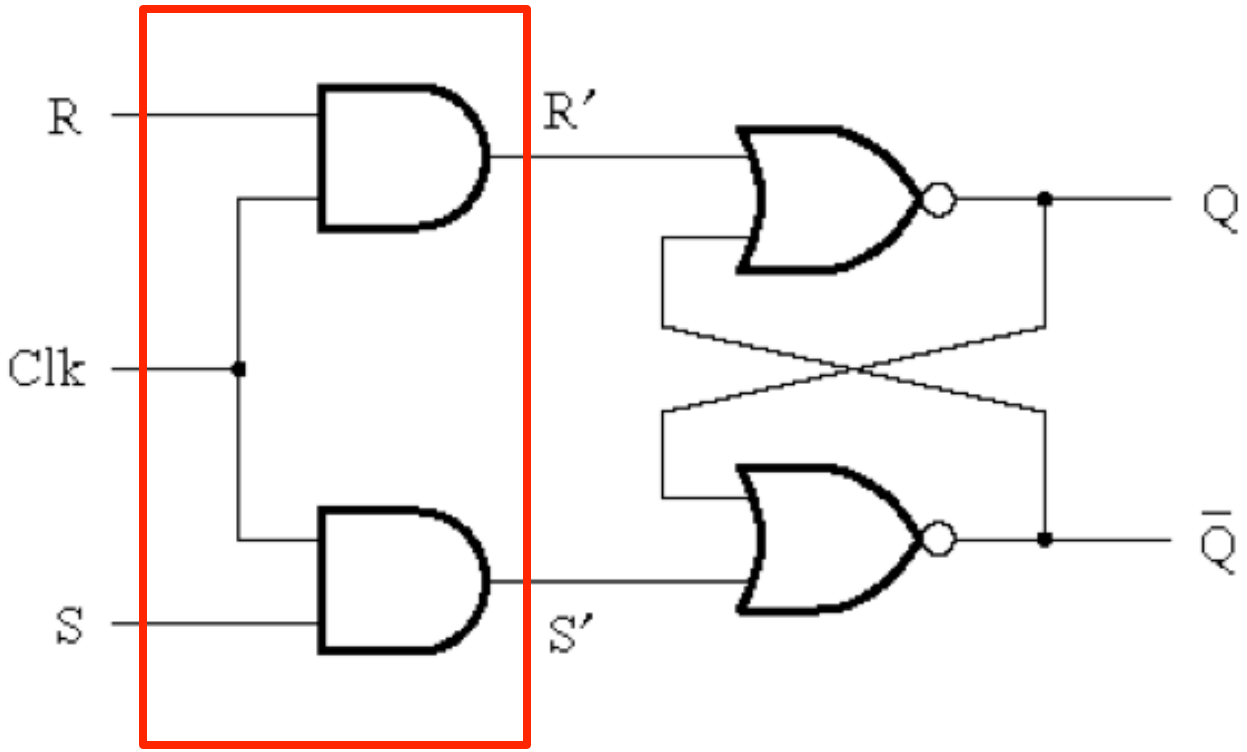
Gated SR Latch

Circuit Diagram for the Gated SR Latch



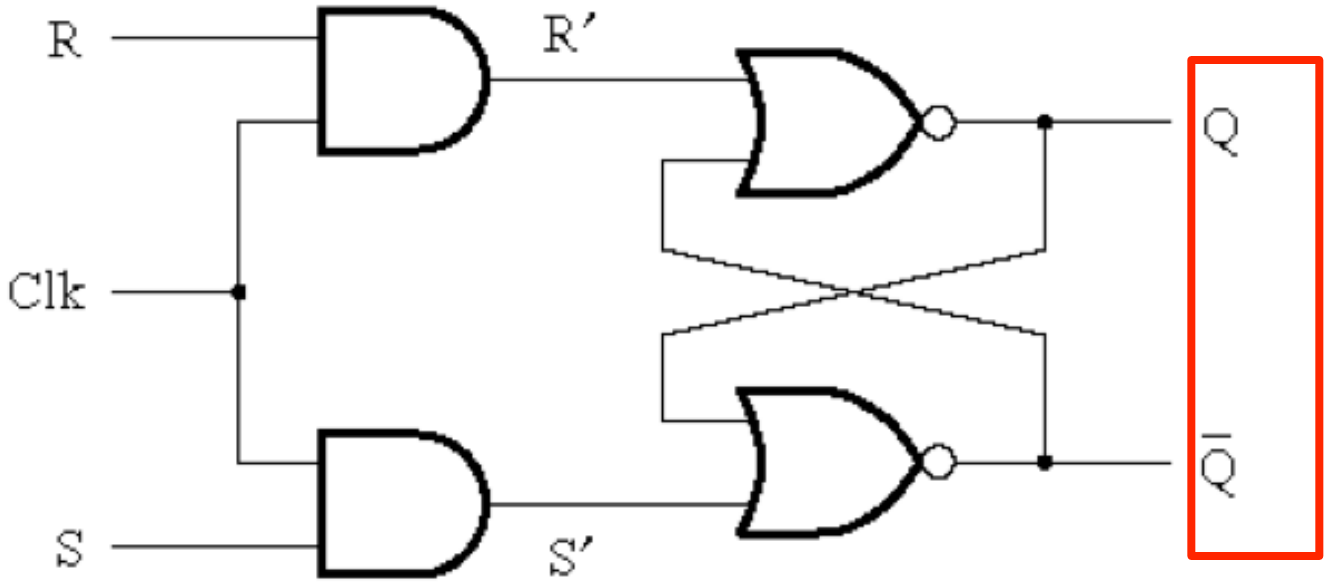
[Figure 5.5a from the textbook]

Circuit Diagram for the Gated SR Latch



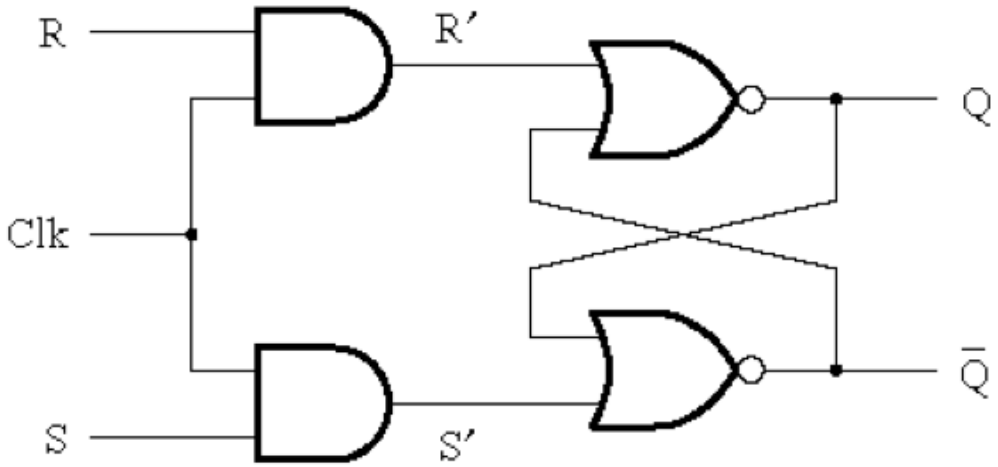
This is the “gate”
of the gated latch

Circuit Diagram for the Gated SR Latch

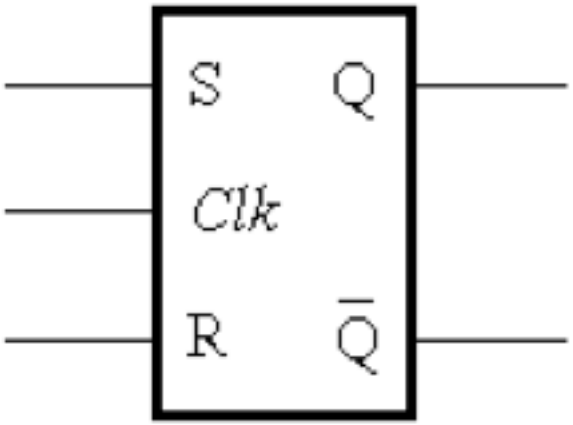


Notice that these are complements of each other

Gated SR Latch: Circuit Diagram, Characteristic Table, and Graphical Symbol

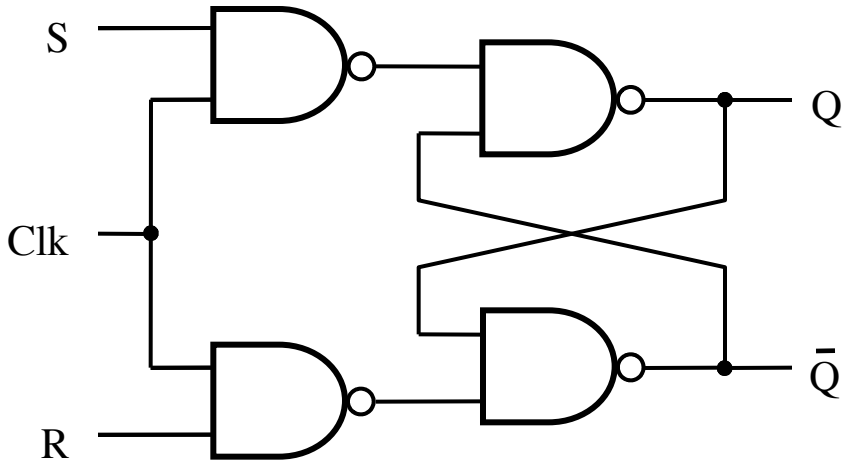


Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x (Undesirable)



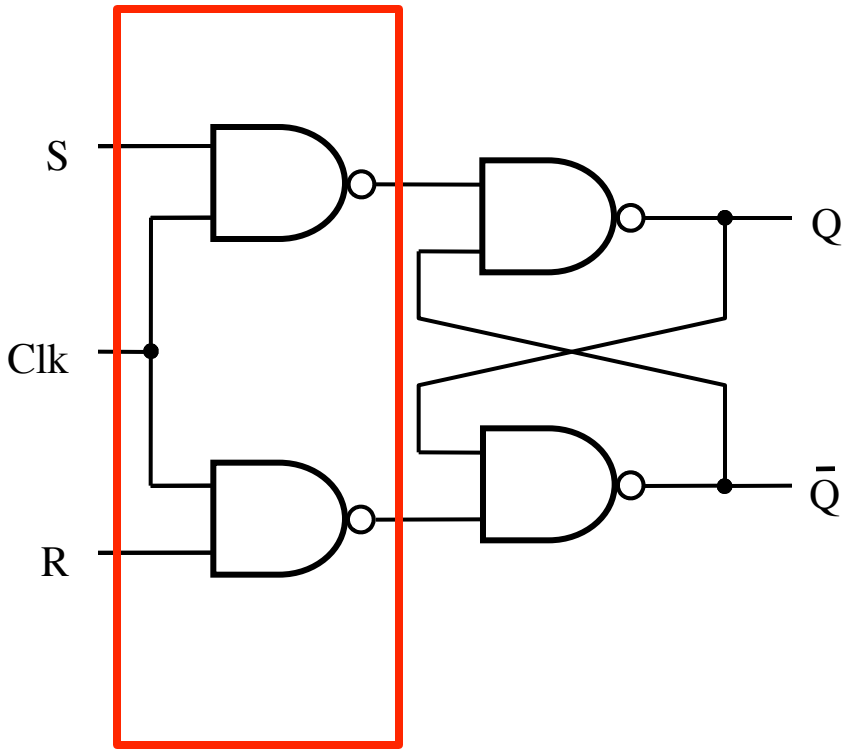
[Figure 5.5 from the textbook]

Gated SR latch with NAND gates



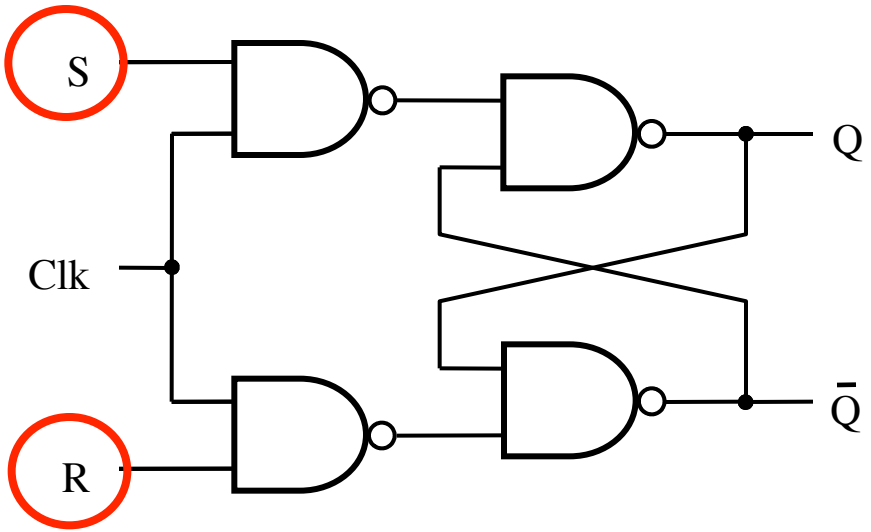
[Figure 5.6 from the textbook]

Gated SR latch with NAND gates



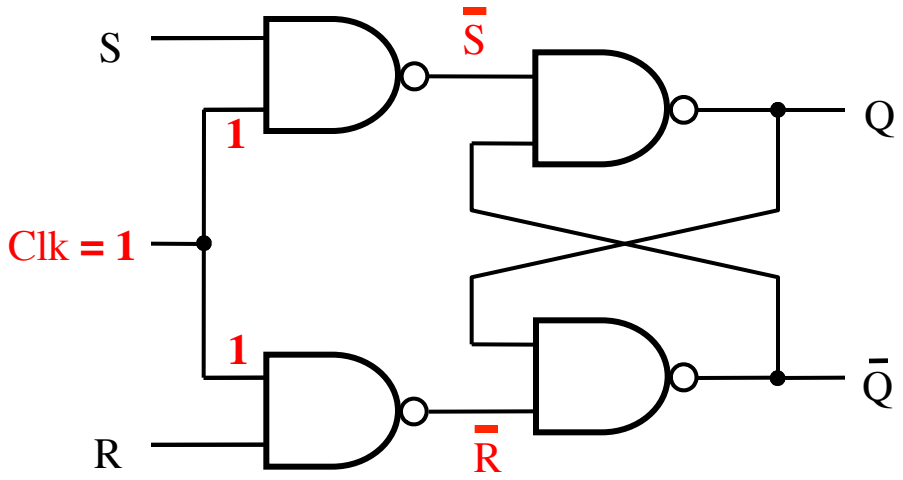
In this case the “gate” is constructed using NAND gates! Not AND gates.

Gated SR latch with NAND gates



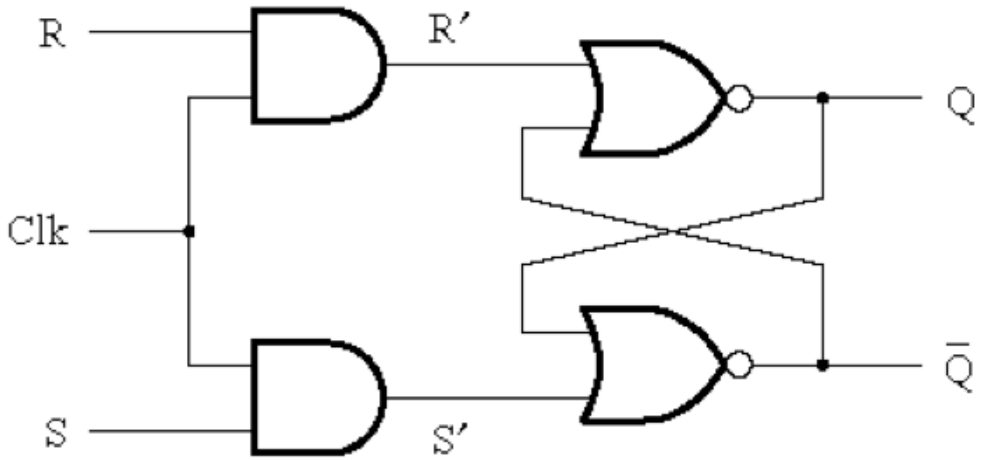
Also, notice that the positions of S and R are now swapped.

Gated SR latch with NAND gates

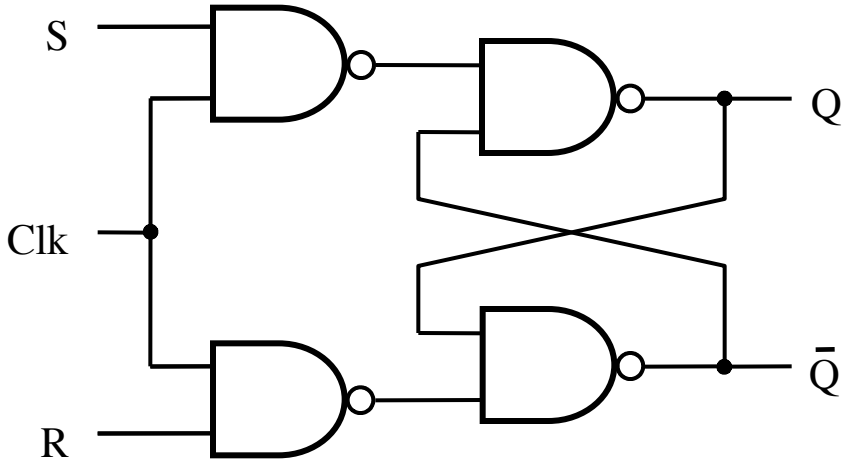


Finally, notice that when Clk=1 this turns into the basic latch with NAND gates, i.e., the $\overline{S}\overline{R}$ Latch.

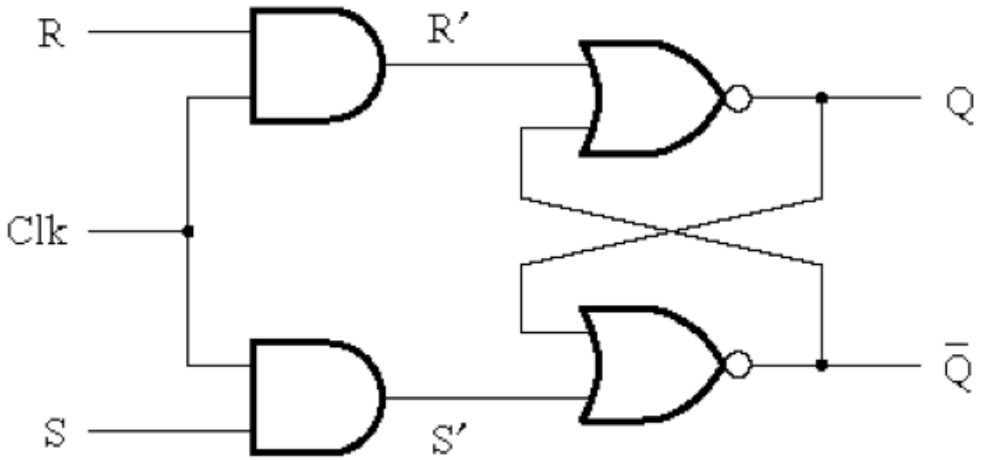
Gated SR latch with NOR gates

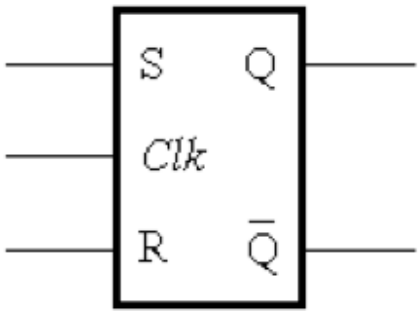


Gated SR latch with NAND gates

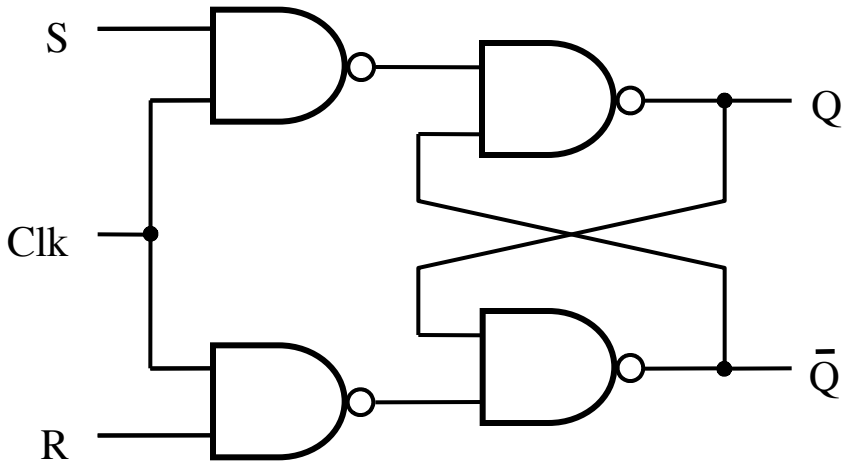


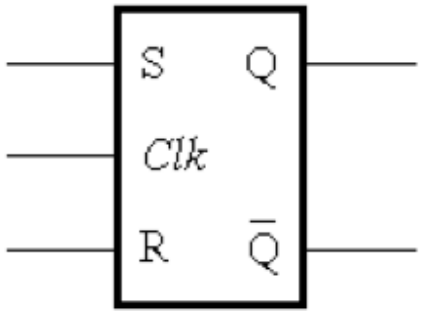
Gated SR latch with NOR gates





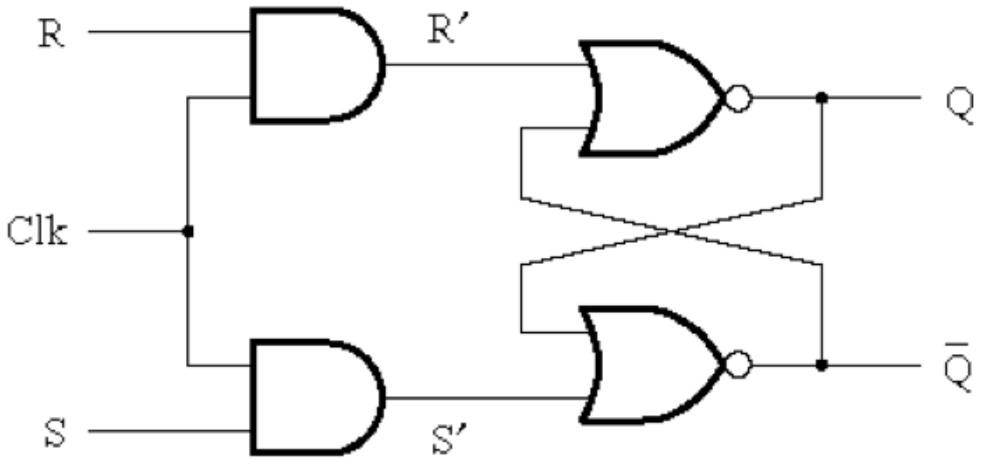
Gated SR latch with NAND gates





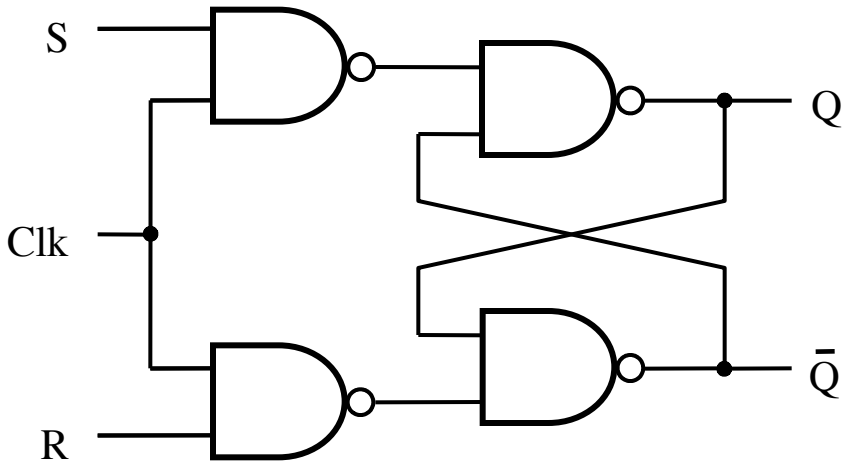
Graphical symbols are the same

Gated SR latch with NOR gates



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 (undesirable)

Gated SR latch with NAND gates

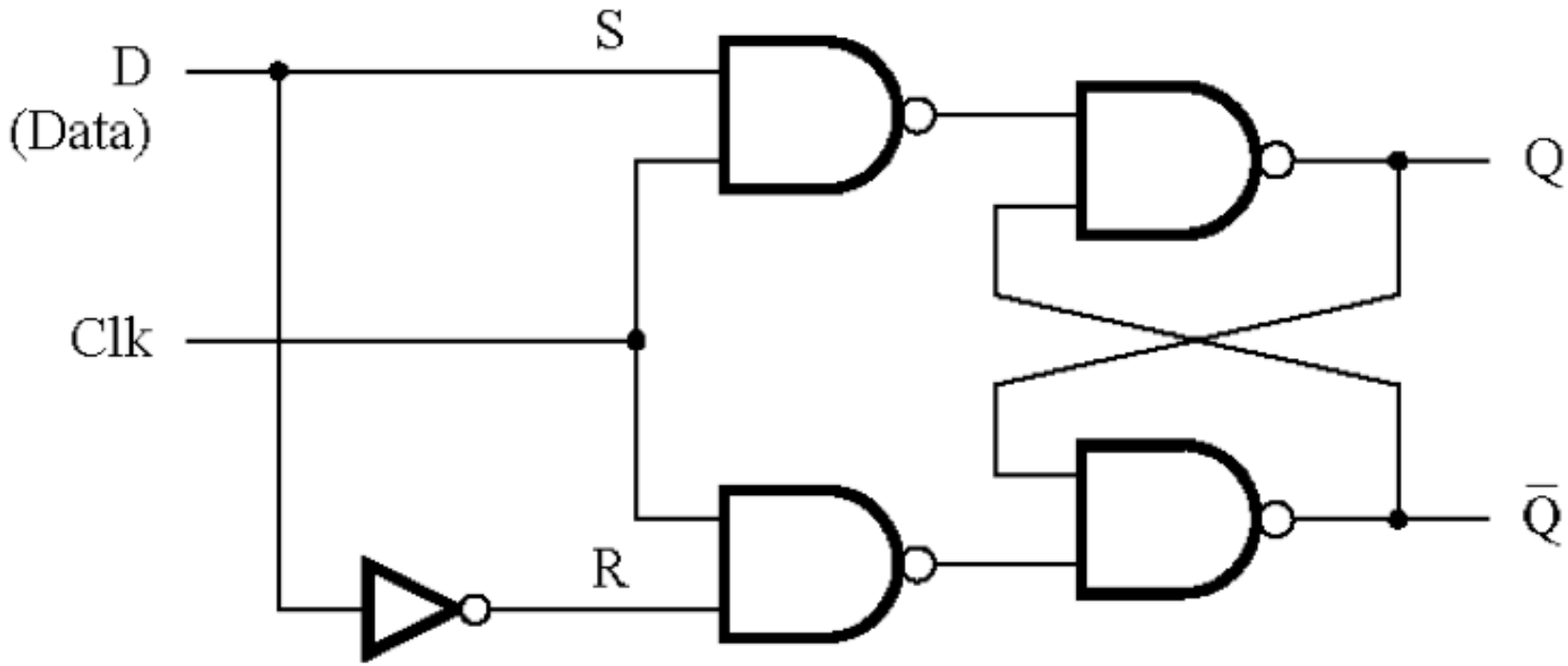


Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x (undesirable)

Characteristic tables are the same

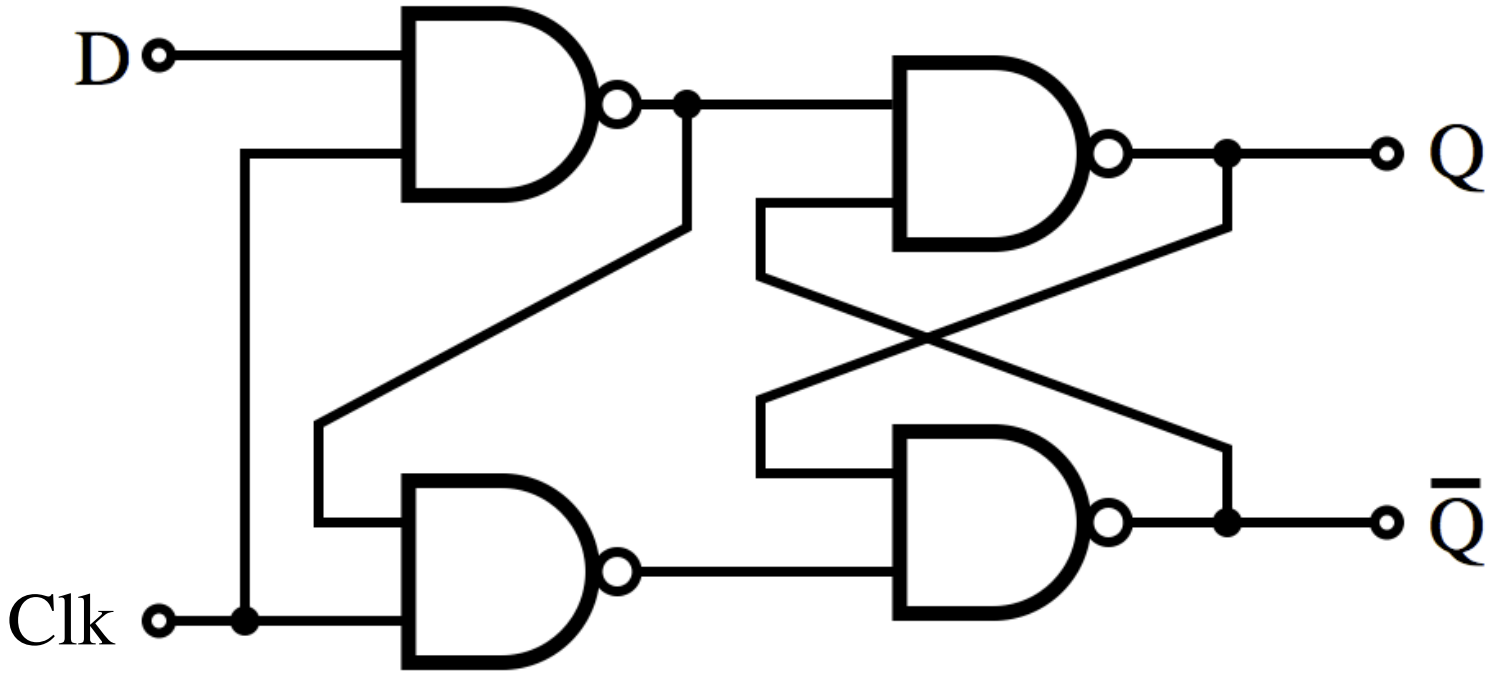
Gated D Latch

Circuit Diagram for the Gated D Latch



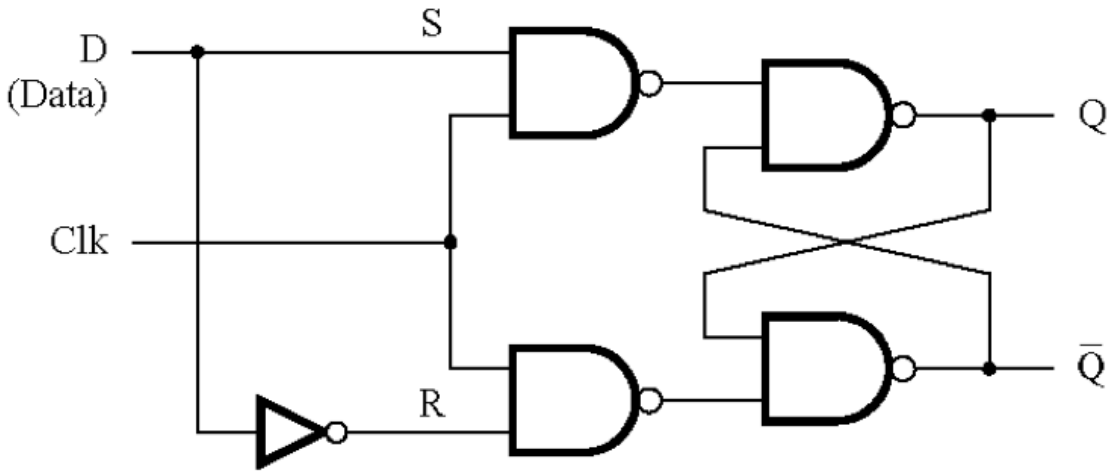
[Figure 5.7a from the textbook]

Gated D Latch: Alternative Design



[https://en.wikipedia.org/wiki/Flip-flop_(electronics)]

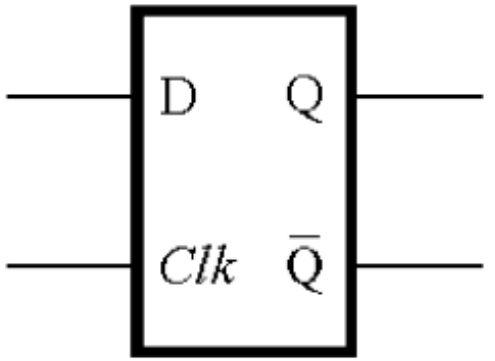
Gated D Latch: Circuit Diagram, Characteristic Table, and Graphical Symbol



Clk	D	Q(t + 1)
0	x	Q(t)
1	0	0
1	1	1

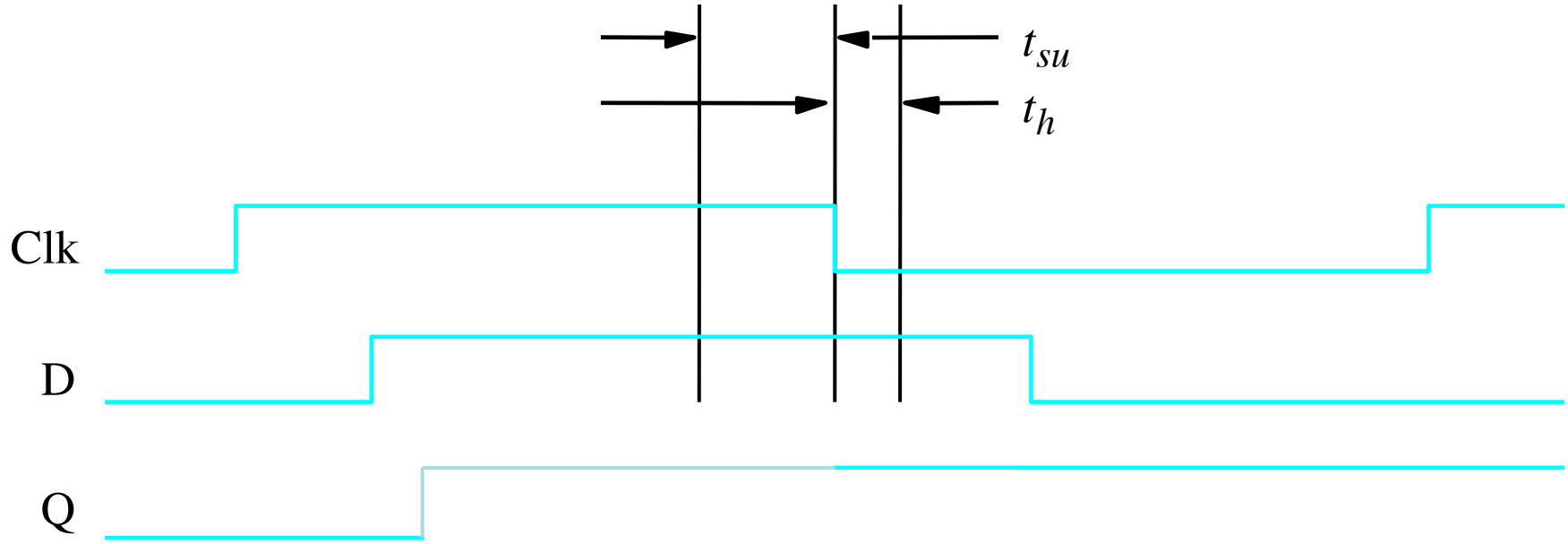
Note that it is now impossible to have S=R=1.

When Clk=1 the output follows the D input.
When Clk=0 the output cannot be changed.



[Figure 5.7a,b from the textbook]

Setup and hold times for Gated D latch



Setup time (t_{su}) – the minimum time that the D signal must be stable prior to the the negative edge of the Clock signal

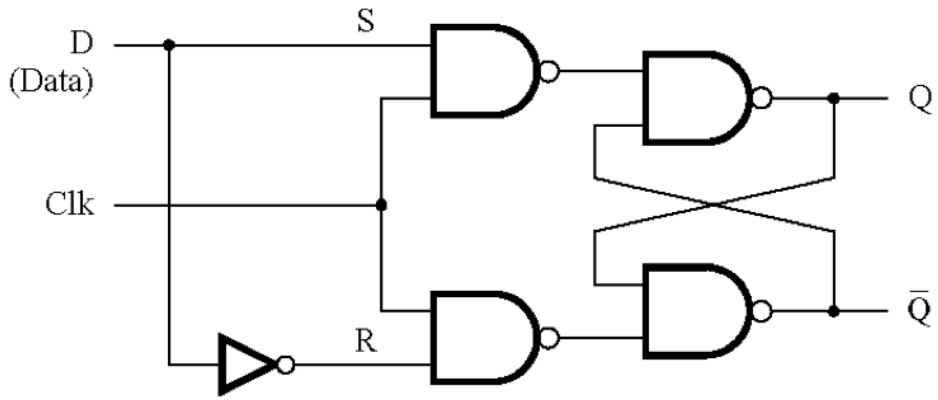
Hold time (t_h) – the minimum time that the D signal must remain stable after the the negative edge of the Clock signal

[Figure 5.8 from the textbook]

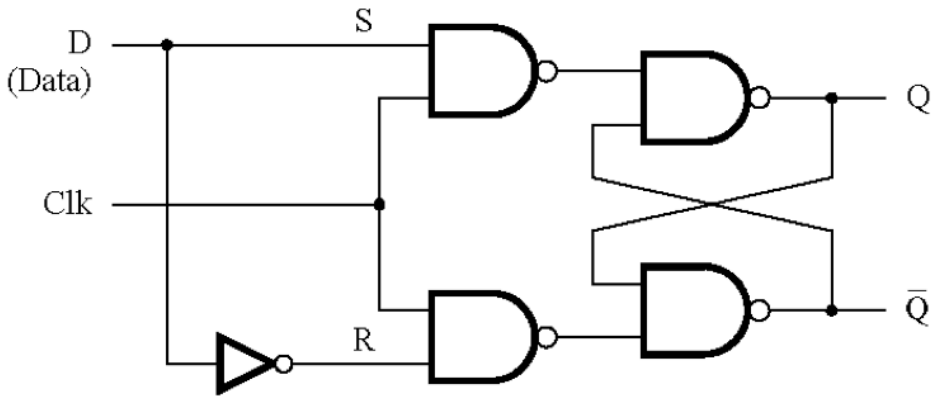
Master-Slave D Flip-Flop

Constructing a Master-Slave D Flip-Flop From Two D Latches

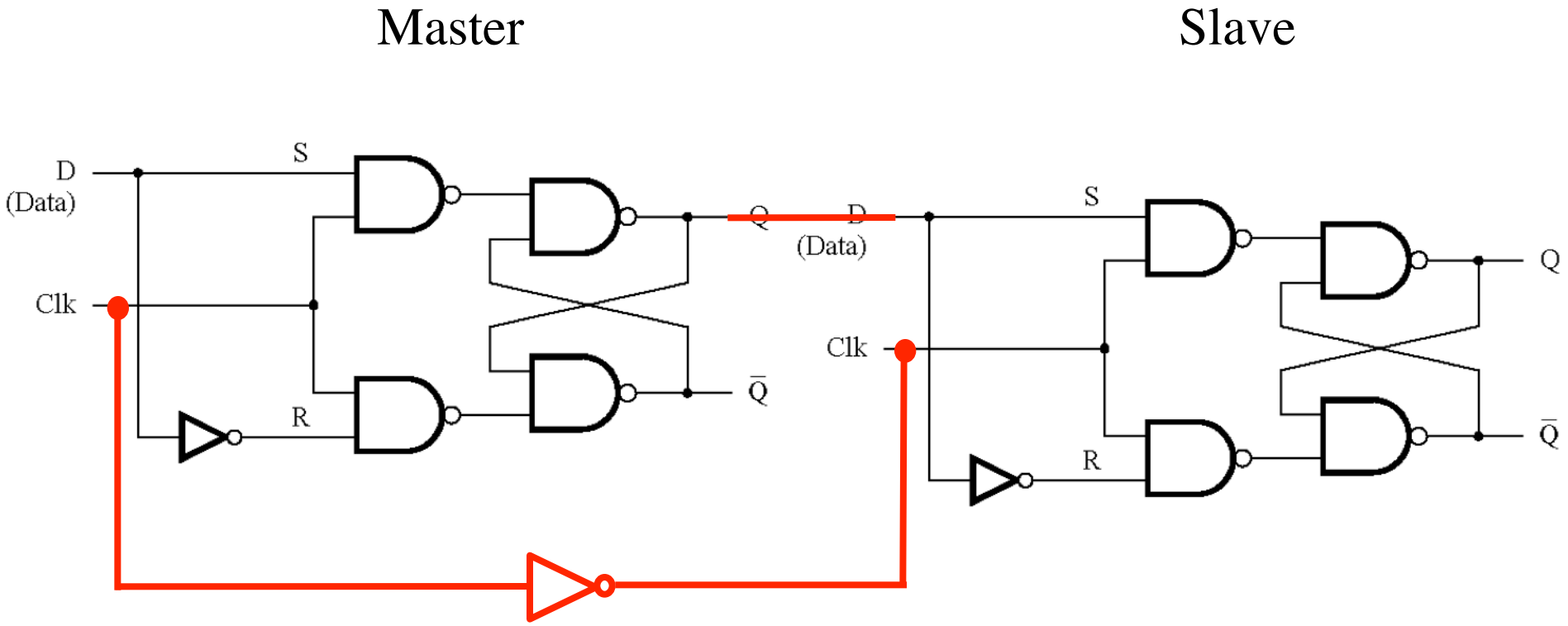
Master



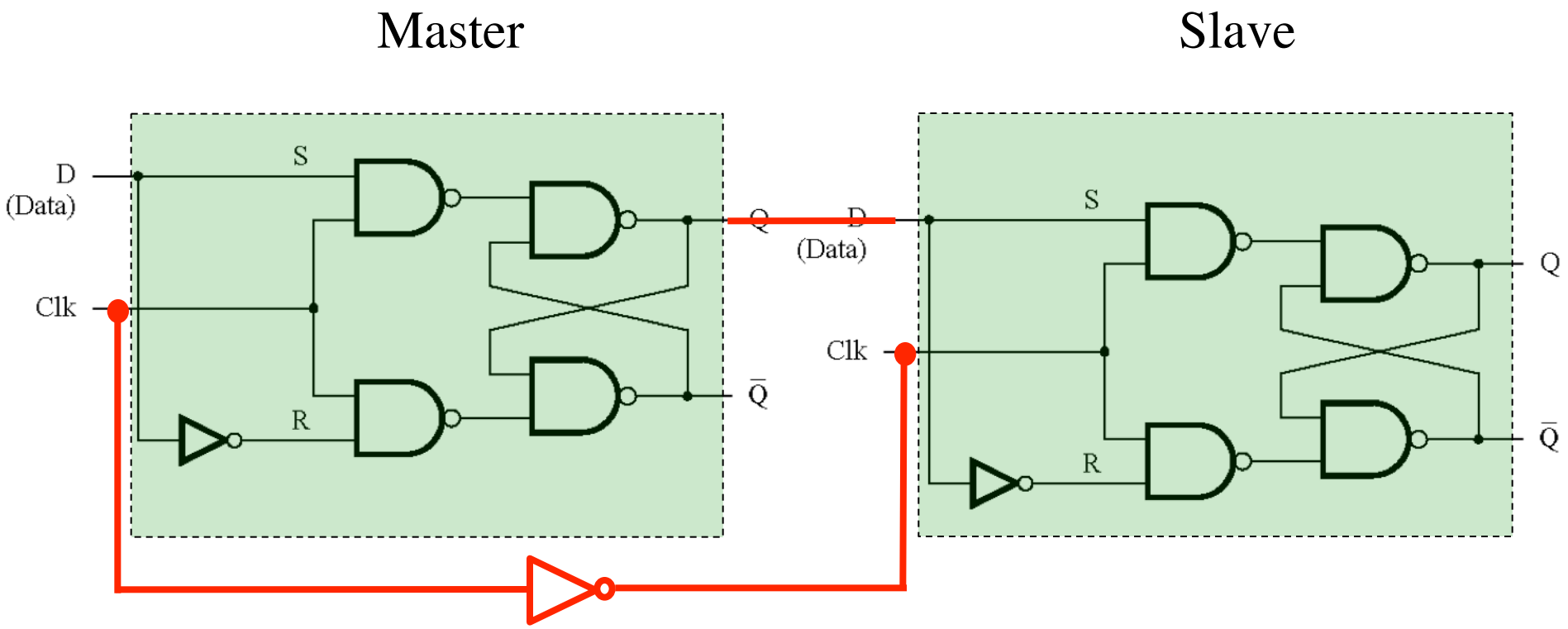
Slave



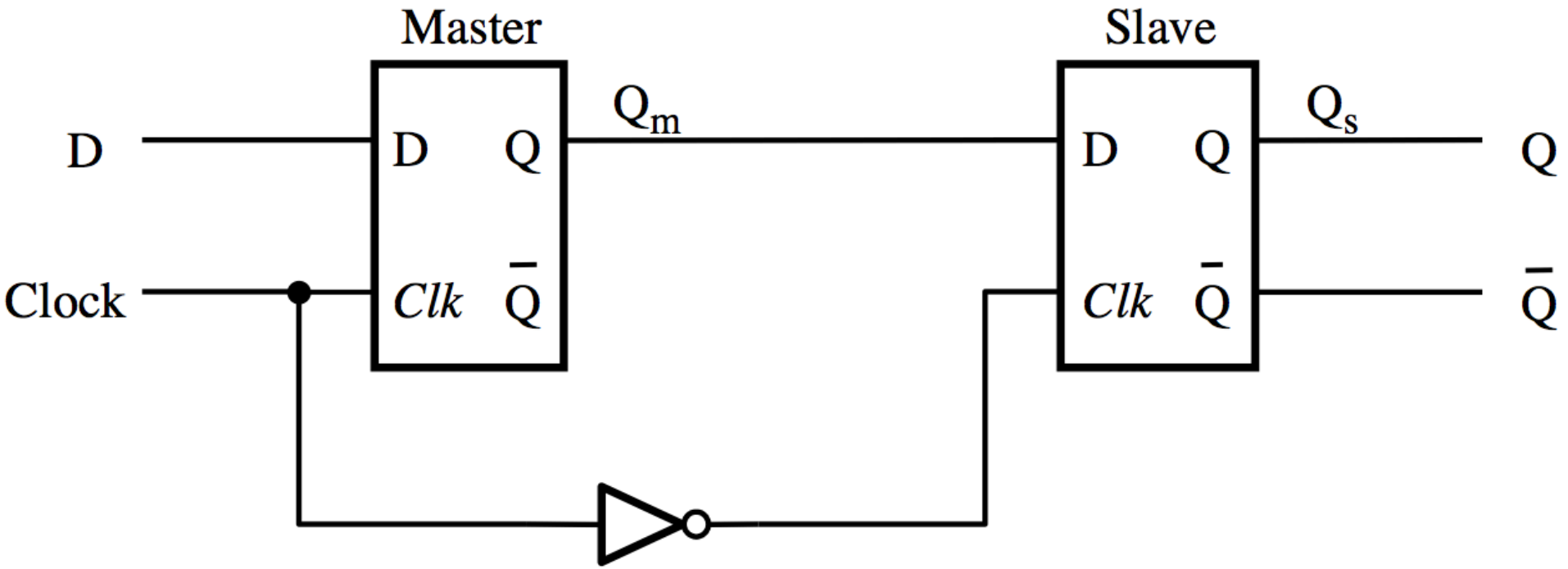
Constructing a Master-Slave D Flip-Flop From Two D Latches



Constructing a Master-Slave D Flip-Flop From Two D Latches



Constructing a Master-Slave D Flip-Flop From Two D Latches



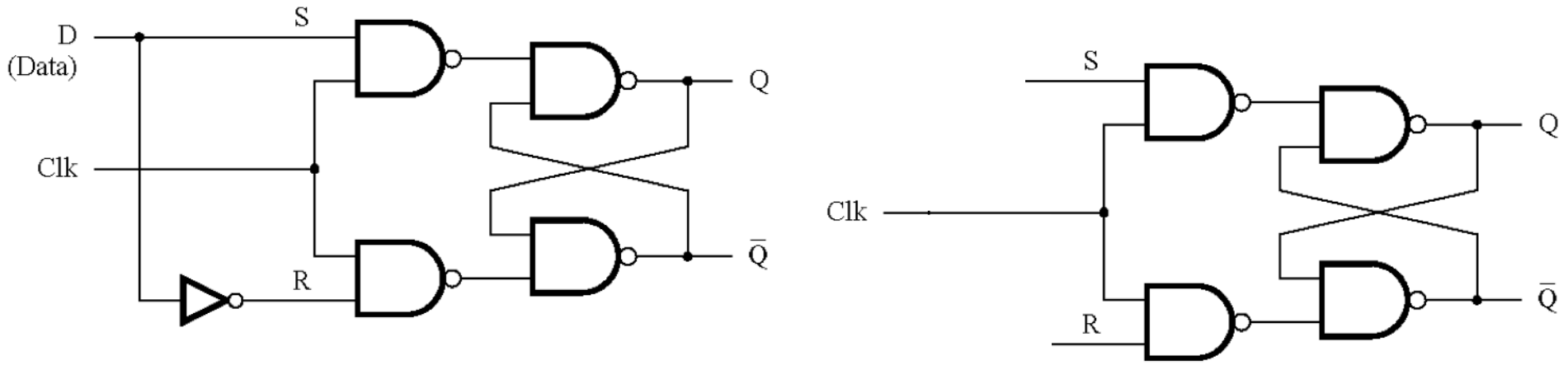
[Figure 5.9a from the textbook]

Constructing a Master-Slave D Flip-Flop From one D Latch and one Gated SR Latch

(This version uses one less NOT gate)

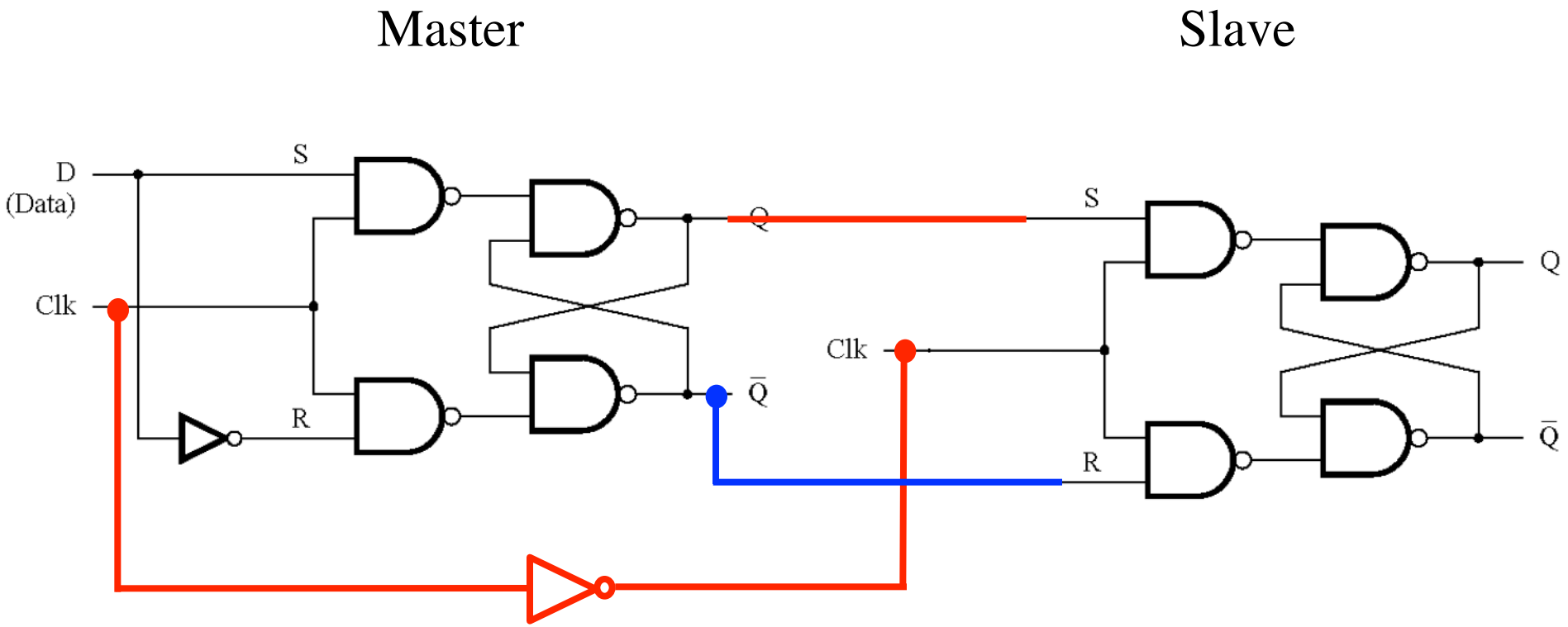
Master

Slave



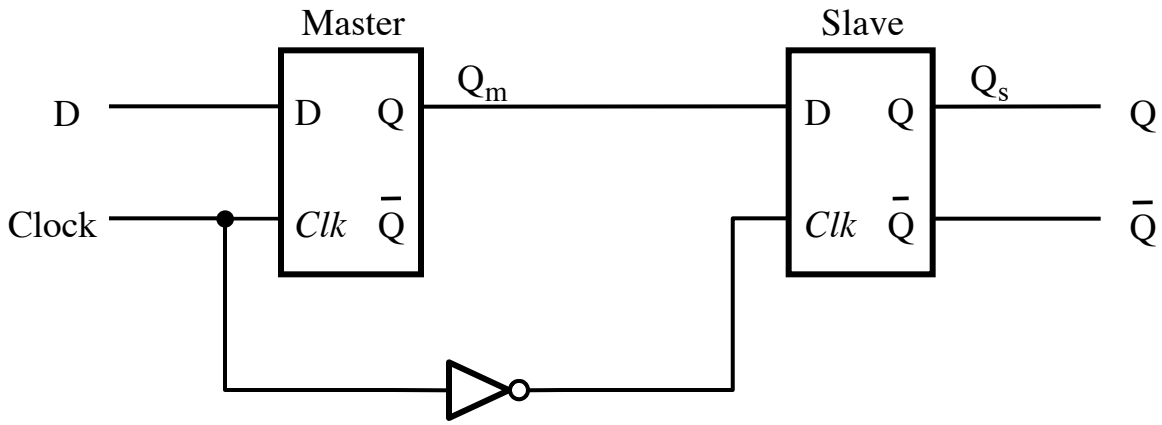
Constructing a Master-Slave D Flip-Flop From one D Latch and one Gated SR Latch

(This version uses one less NOT gate)



Edge-Triggered D Flip-Flops

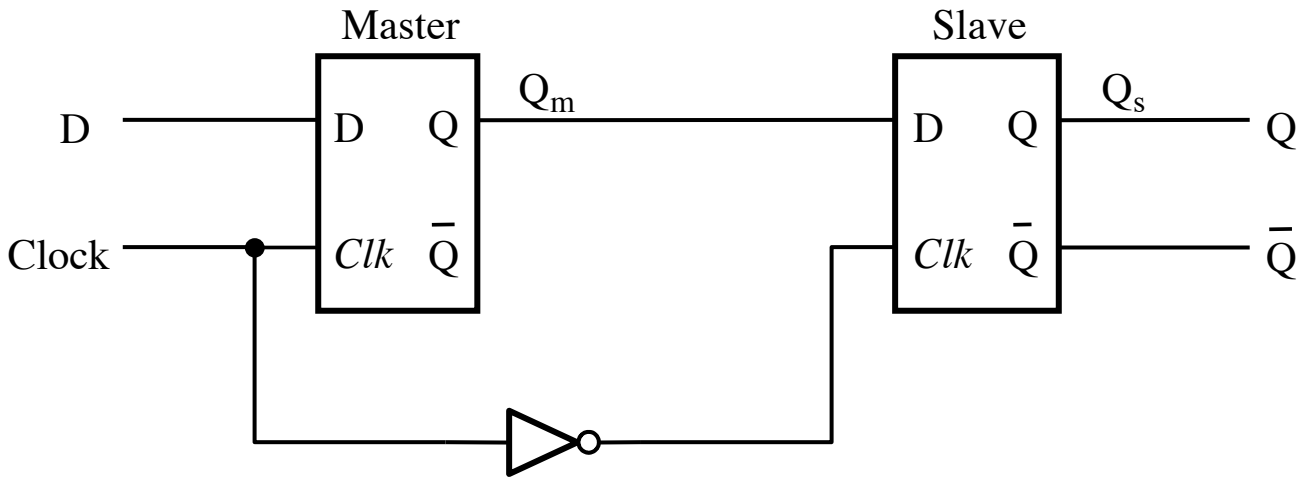
Master-Slave D Flip-Flop



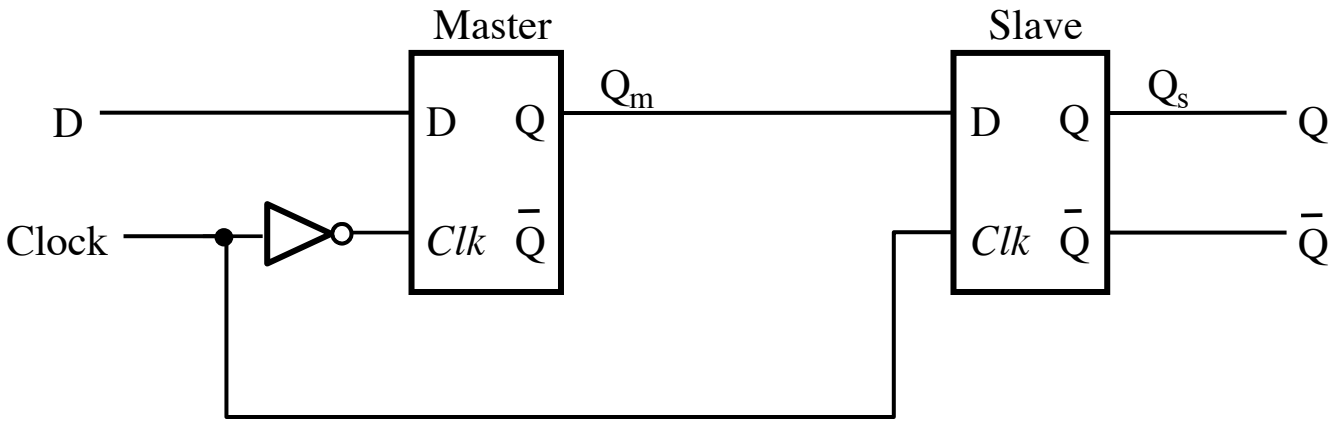
(a) Circuit

[Figure 5.9a from the textbook]

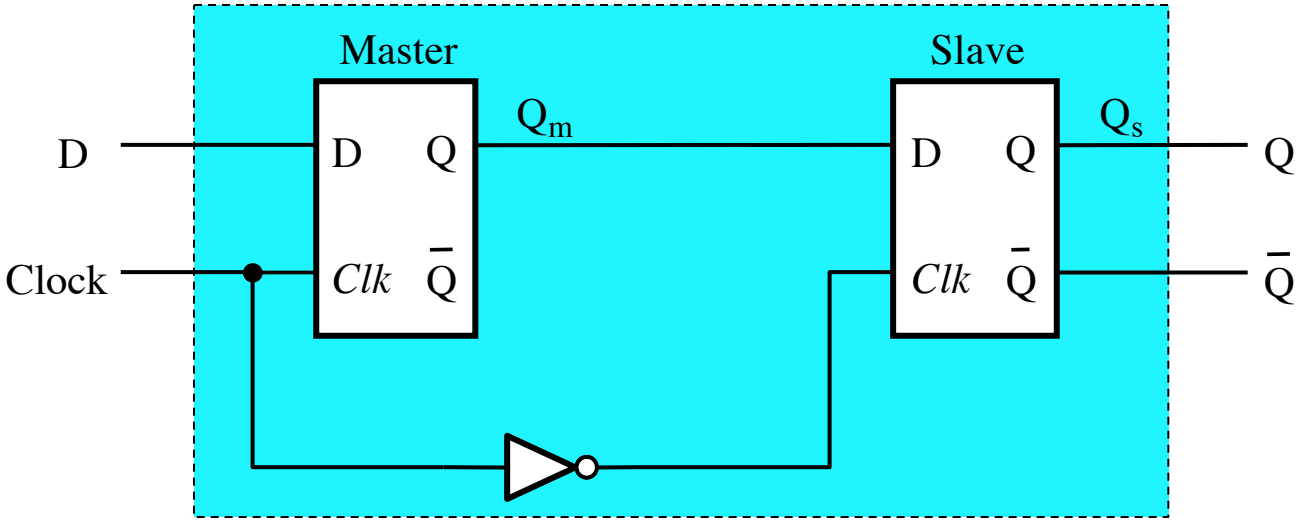
Negative-Edge-Triggered Master-Slave D Flip-Flop



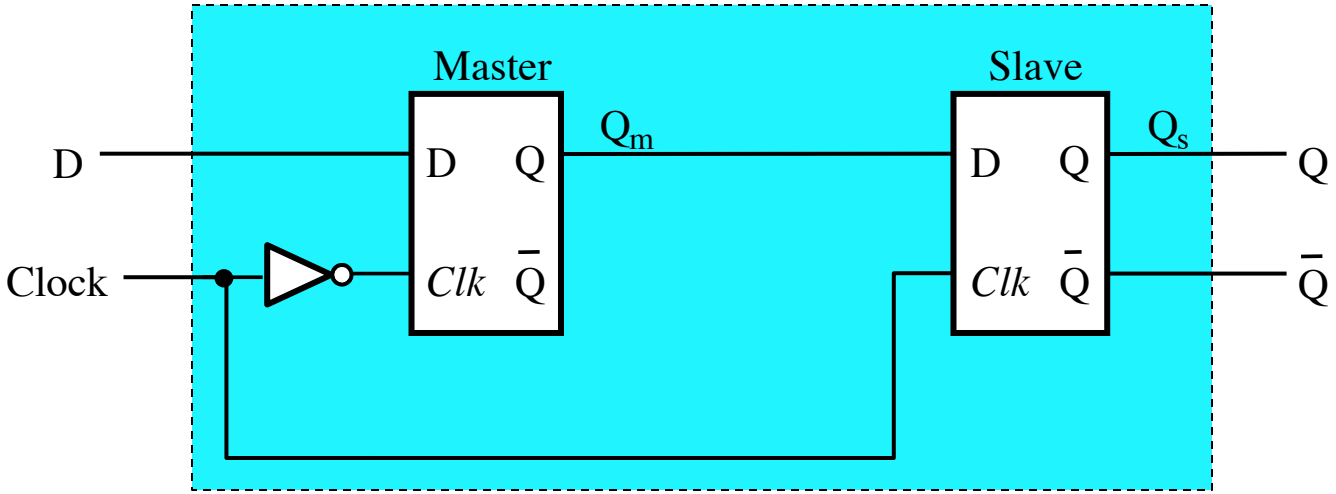
Positive-Edge-Triggered Master-Slave D Flip-Flop



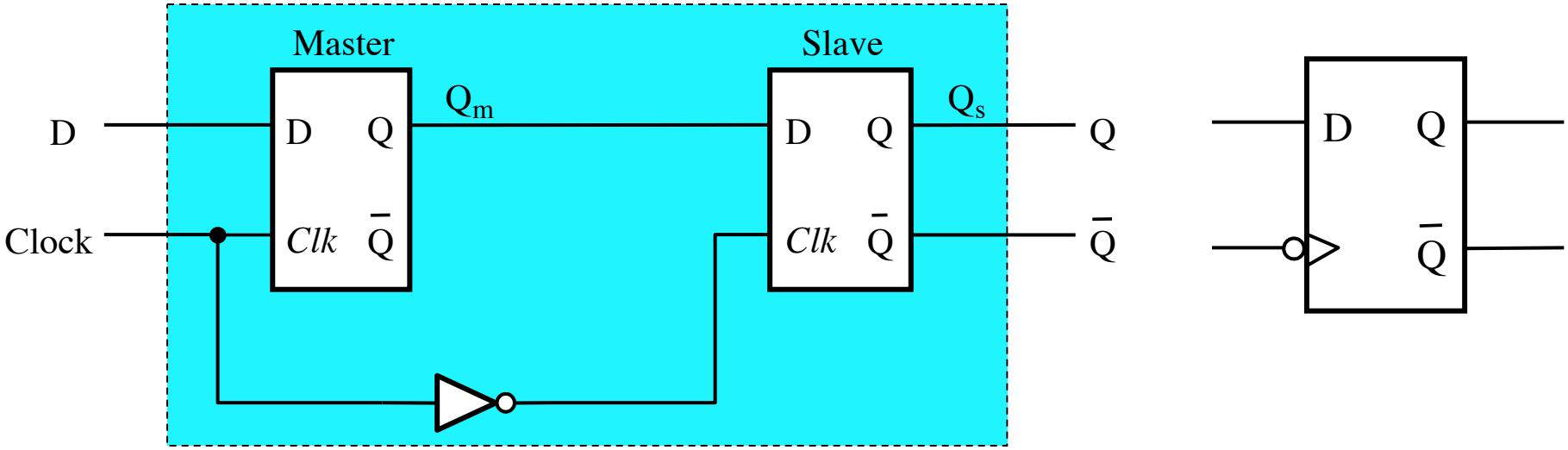
Negative-Edge-Triggered Master-Slave D Flip-Flop



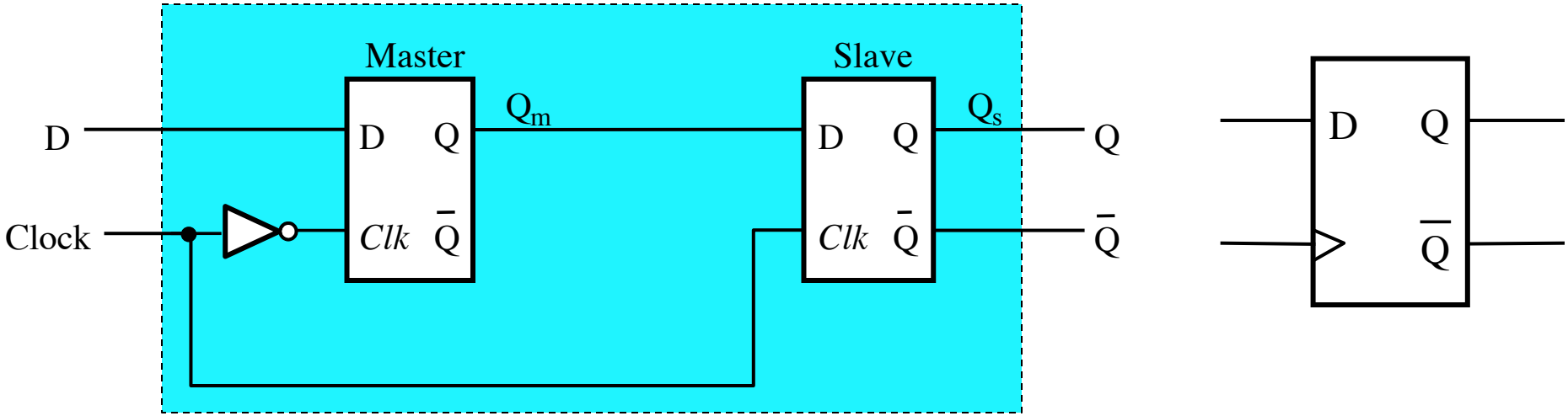
Positive-Edge-Triggered Master-Slave D Flip-Flop



Negative-Edge-Triggered Master-Slave D Flip-Flop

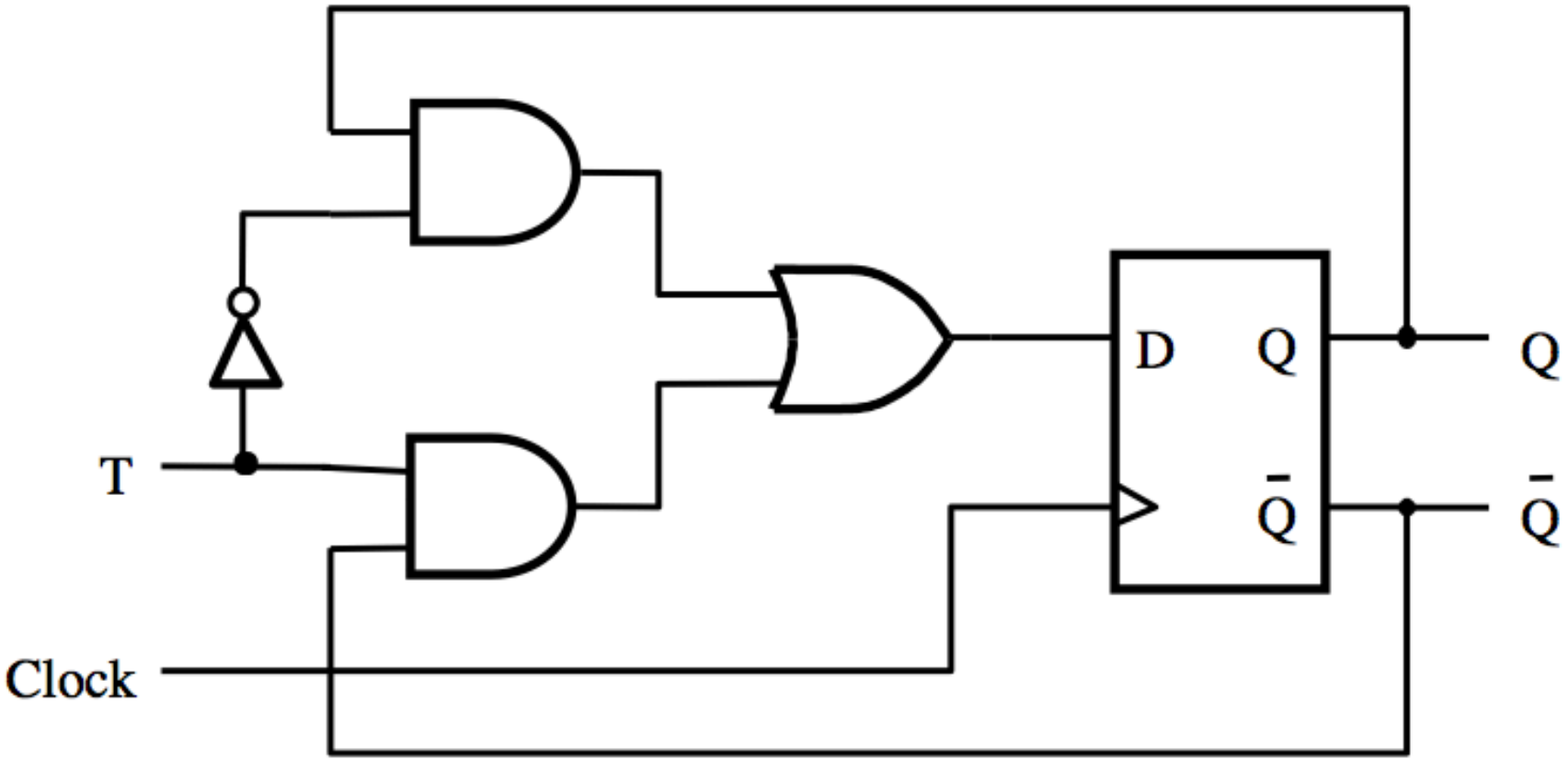


Positive-Edge-Triggered Master-Slave D Flip-Flop



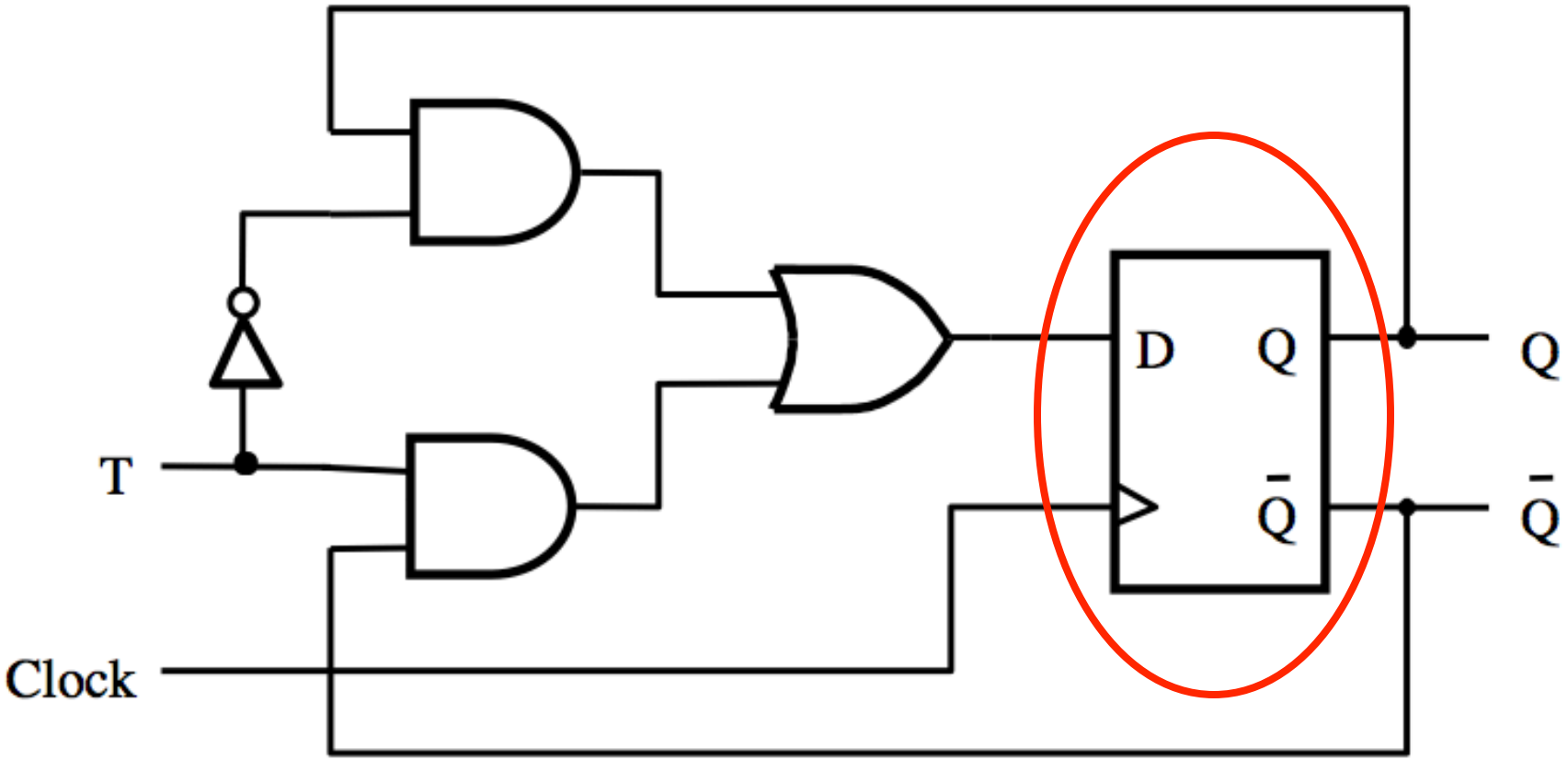
T Flip-Flop

T Flip-Flop



[Figure 5.15a from the textbook]

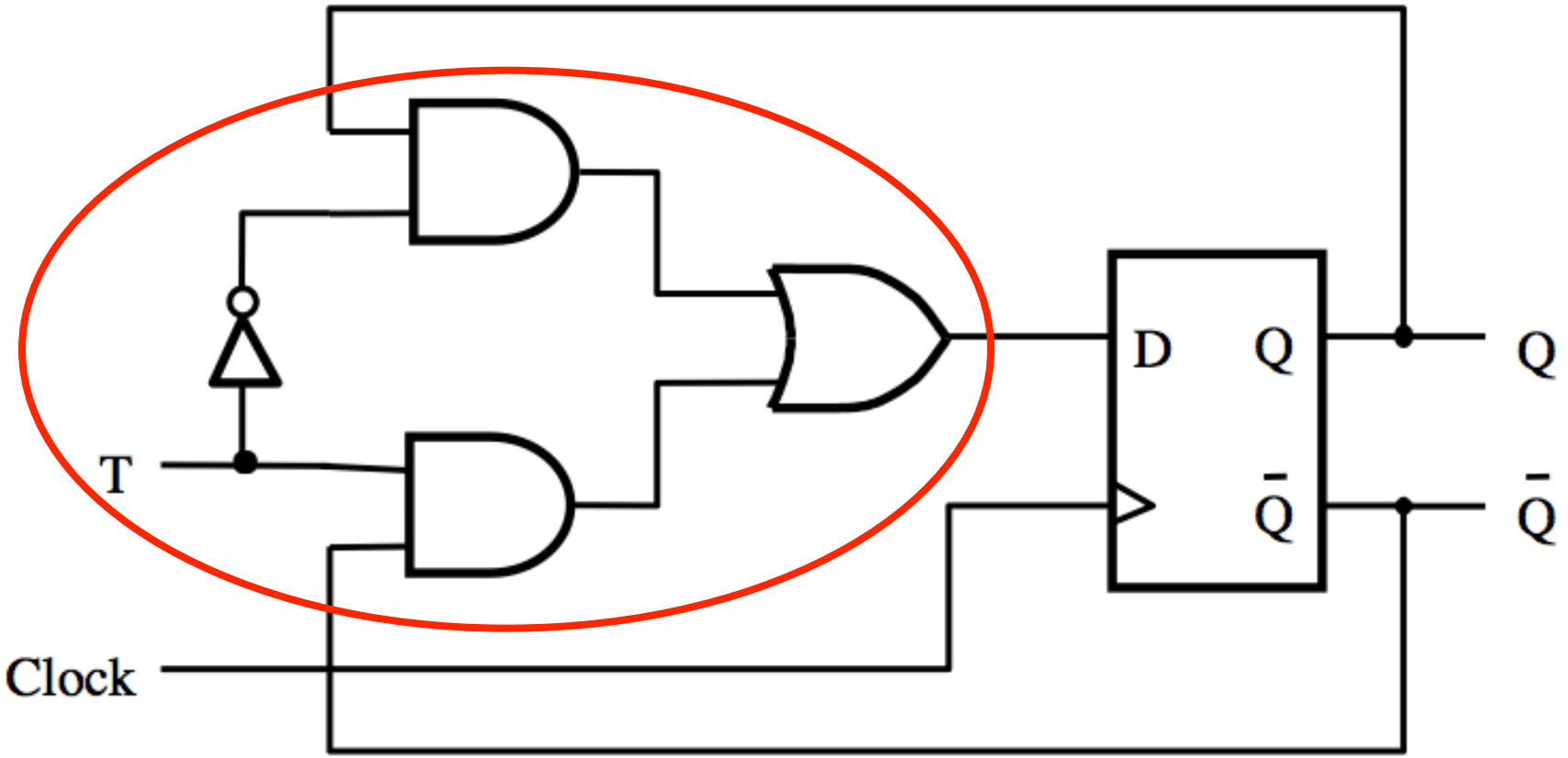
T Flip-Flop



Positive-edge-triggered
D Flip-Flop

[Figure 5.15a from the textbook]

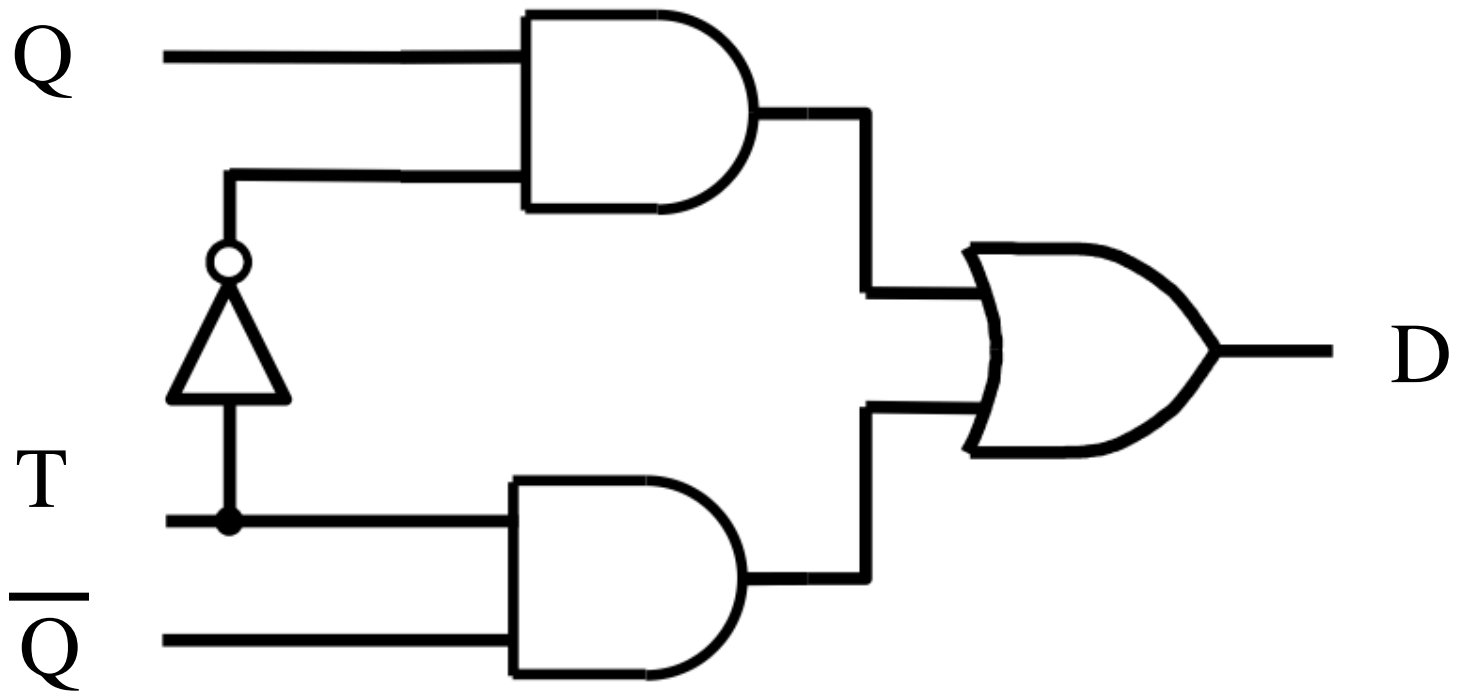
T Flip-Flop



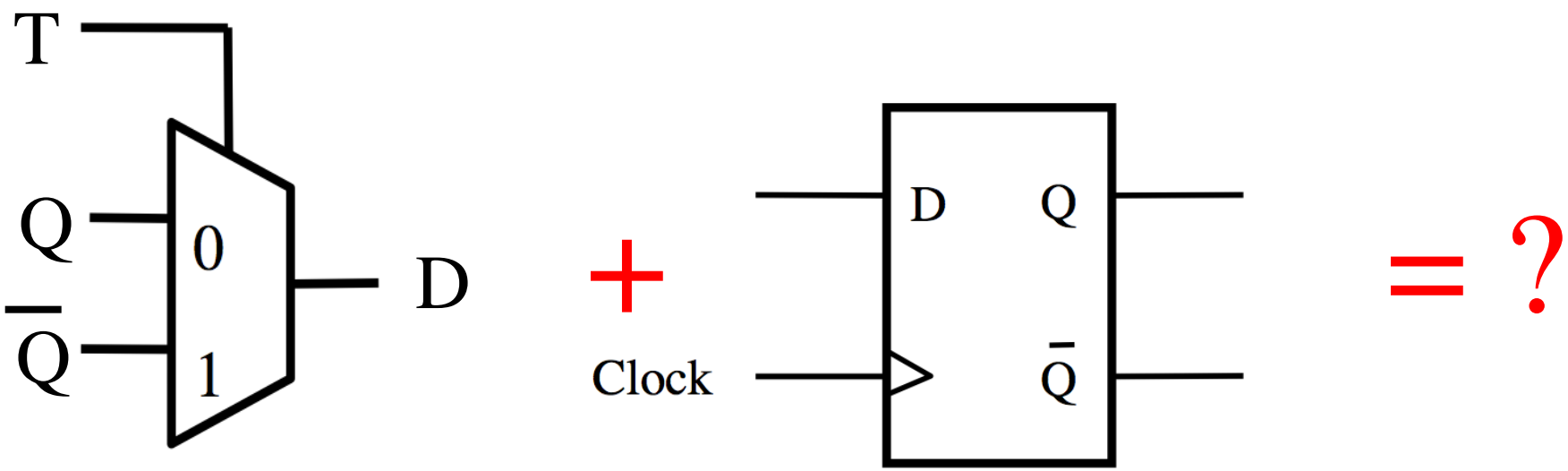
What is this?

[Figure 5.15a from the textbook]

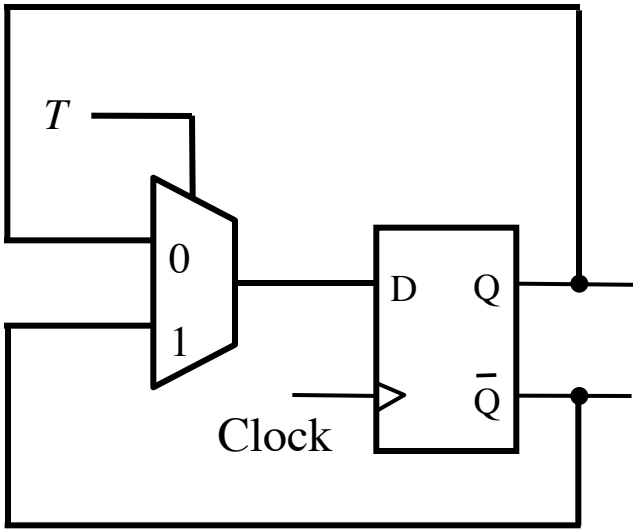
What is this?



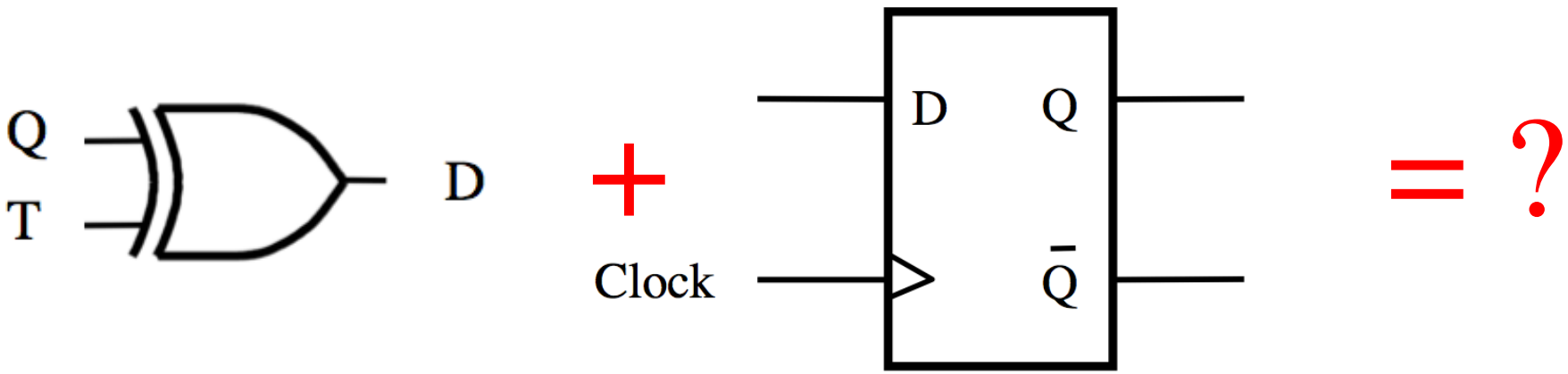
What is this?



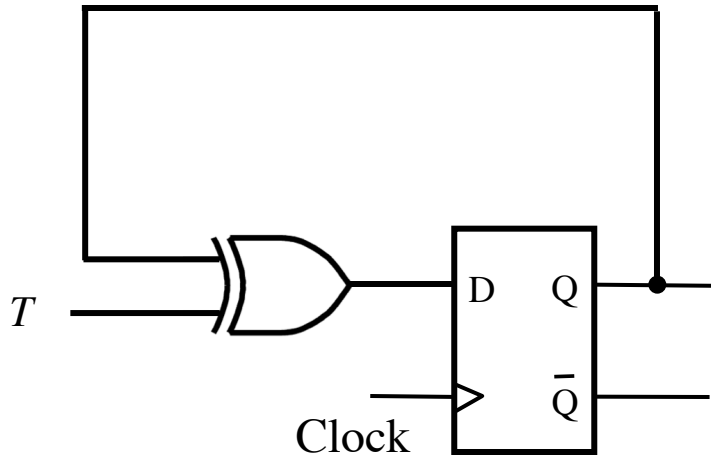
T Flip-Flop



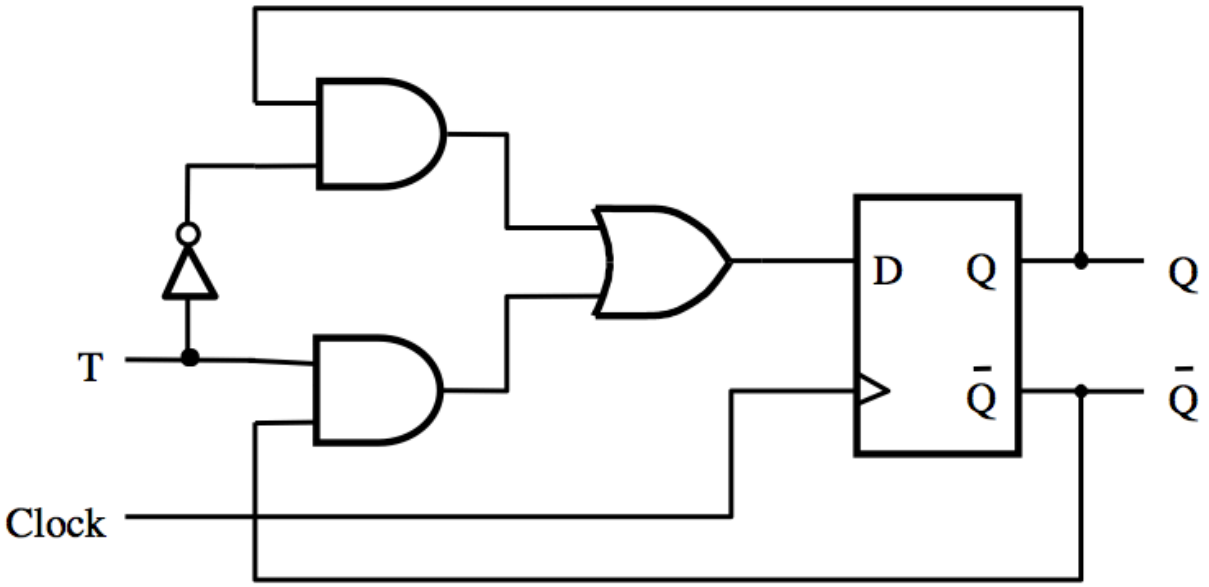
What is this?



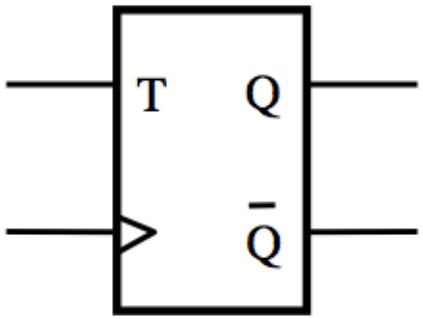
T Flip-Flop



T Flip-Flop
(circuit, truth table and graphical symbol)



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



[Figure 5.15a-c from the textbook]

T Flip-Flop
(How it Works)

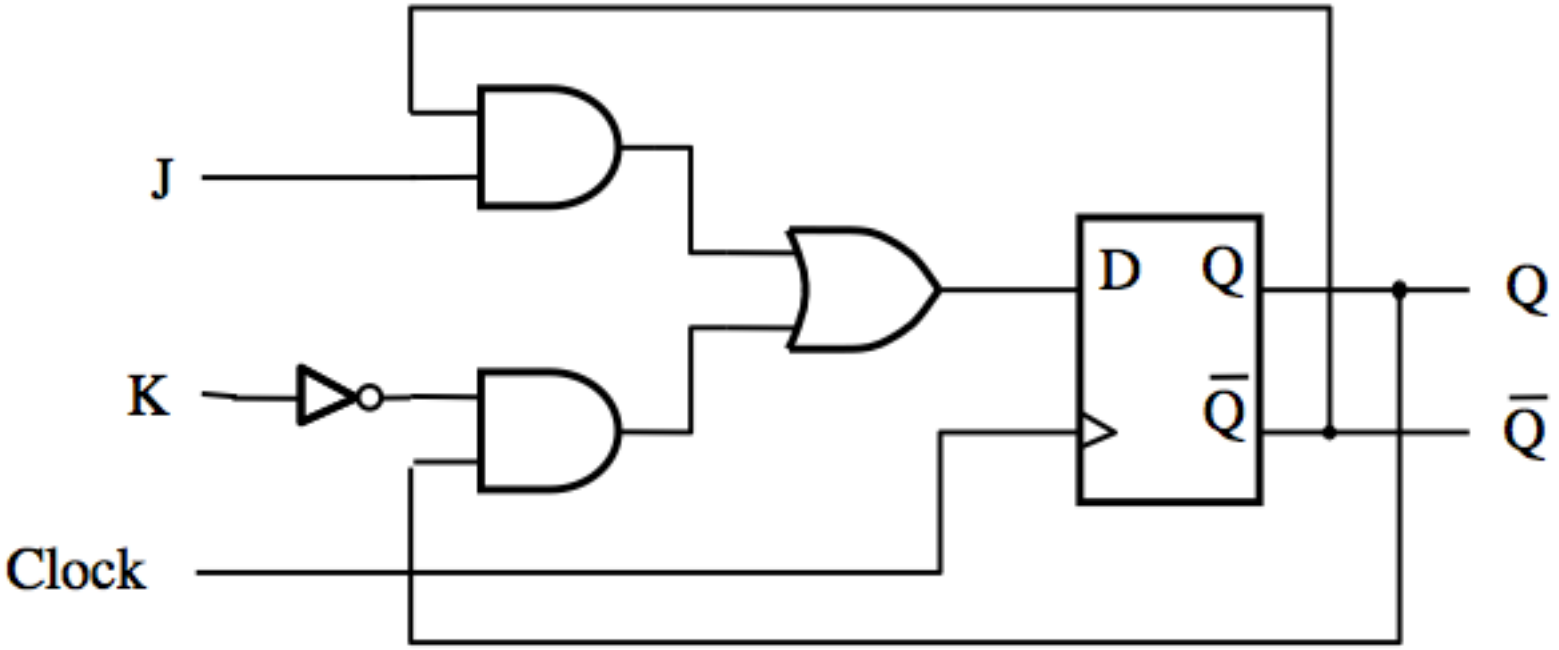
If T=0 then it stays in its current state

If T=1 then it reverses its current state

In other words the circuit “toggles” its state when T=1. This is why it is called T flip-flop.

JK Flip-Flop

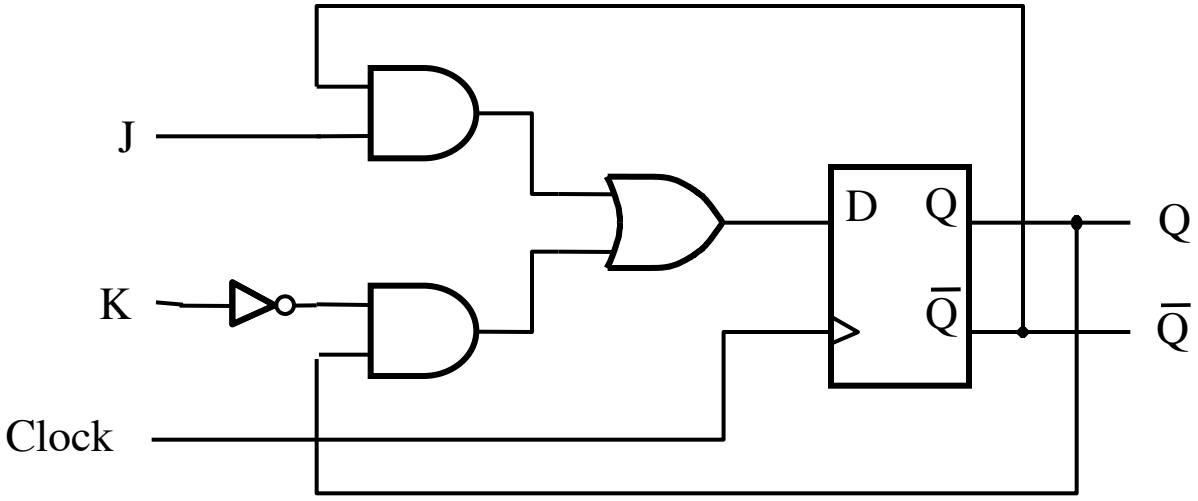
JK Flip-Flop



$$D = J\overline{Q} + \overline{K}Q$$

[Figure 5.16a from the textbook]

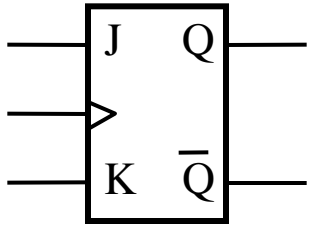
JK Flip-Flop



(a) Circuit

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

[Figure 5.16 from the textbook]

JK Flip-Flop
(How it Works)

A versatile circuit that can be used both as a SR flip-flop and as a T flip flop

If J=0 and S =0 it stays in the same state

Just like SR It can be set and reset

J=S and K=R

If J=K=1 then it behaves as a T flip-flop

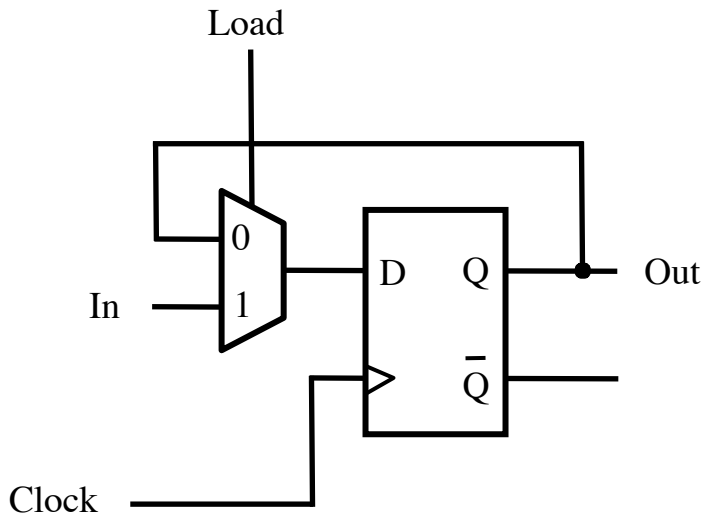
Registers

Register
(Definition)

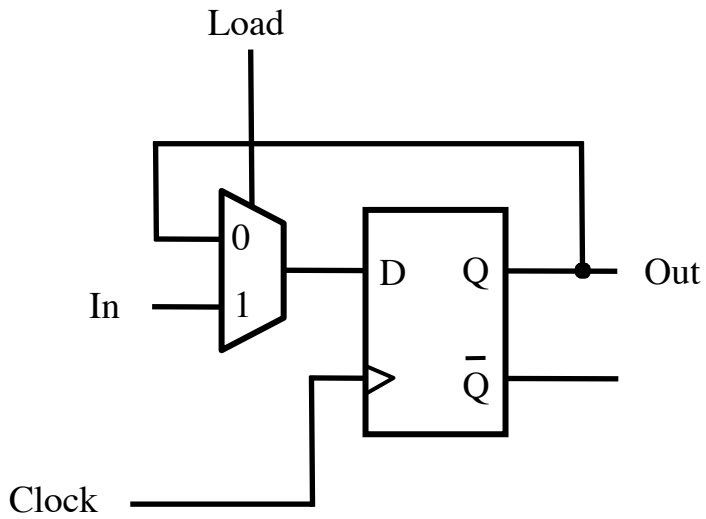
An n-bit structure consisting of flip-flops

Parallel-Access Register

1-Bit Parallel-Access Register



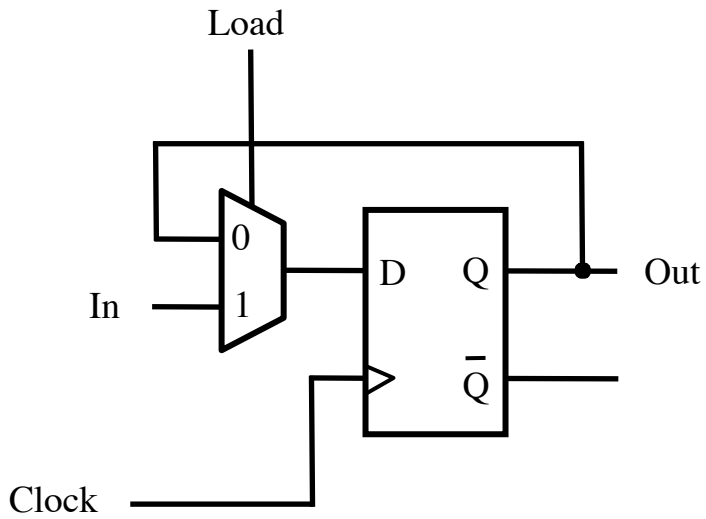
1-Bit Parallel-Access Register



The 2-to-1 multiplexer is used to select whether to load a new value into the D flip-flop or to retain the old value.

The output of this circuit is the Q output of the flip-flop.

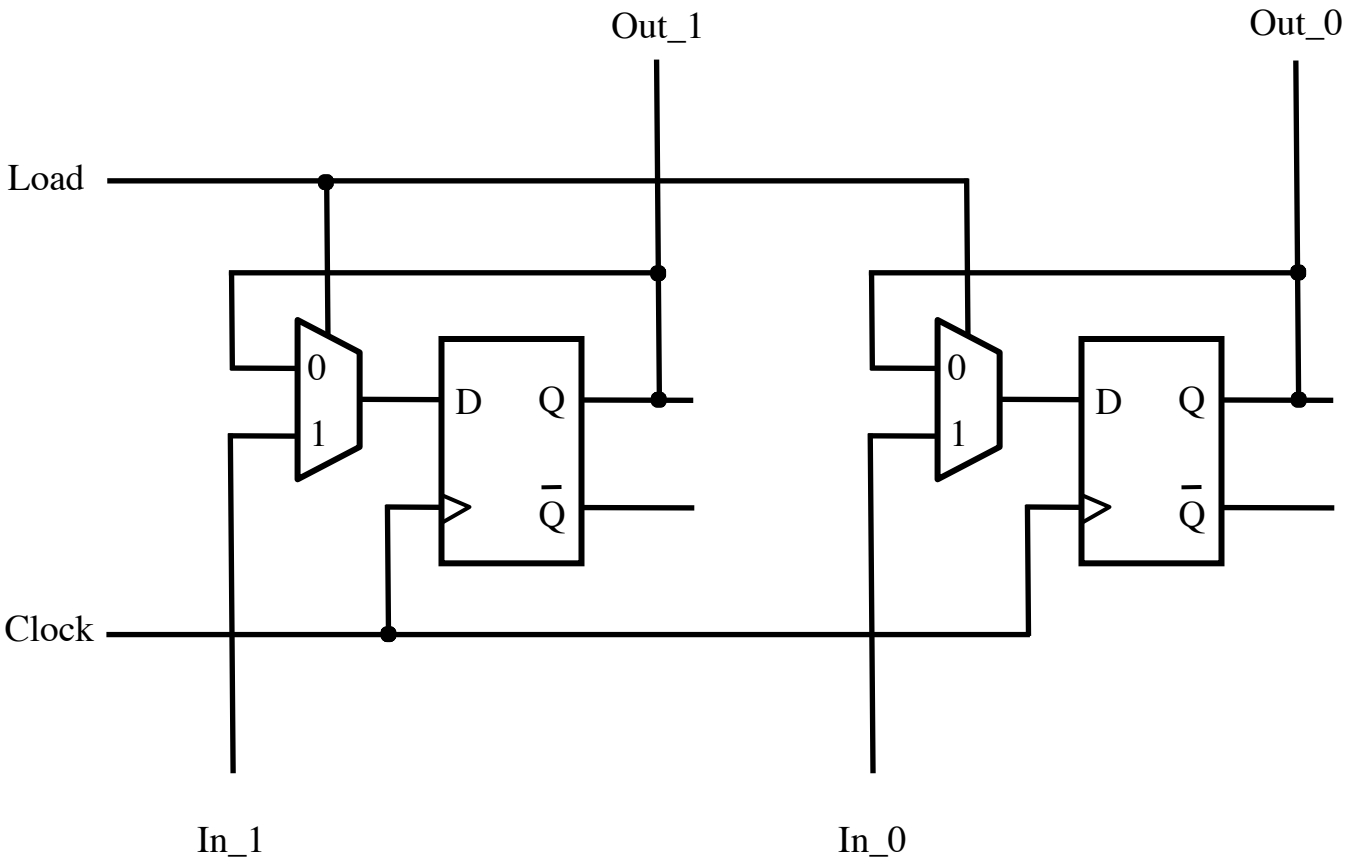
1-Bit Parallel-Access Register



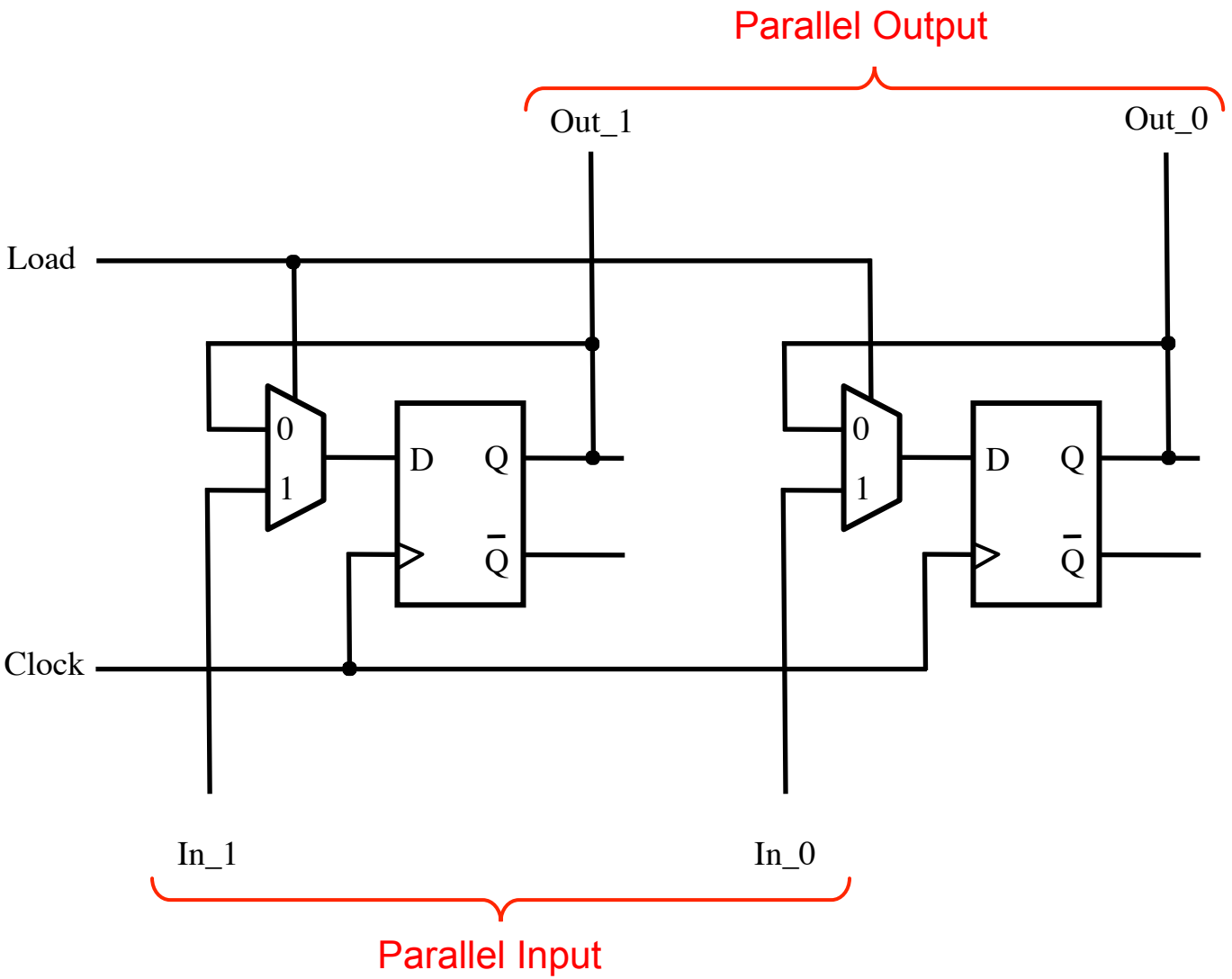
If Load = 0, then retain the old value.

If Load = 1, then load the new value from In.

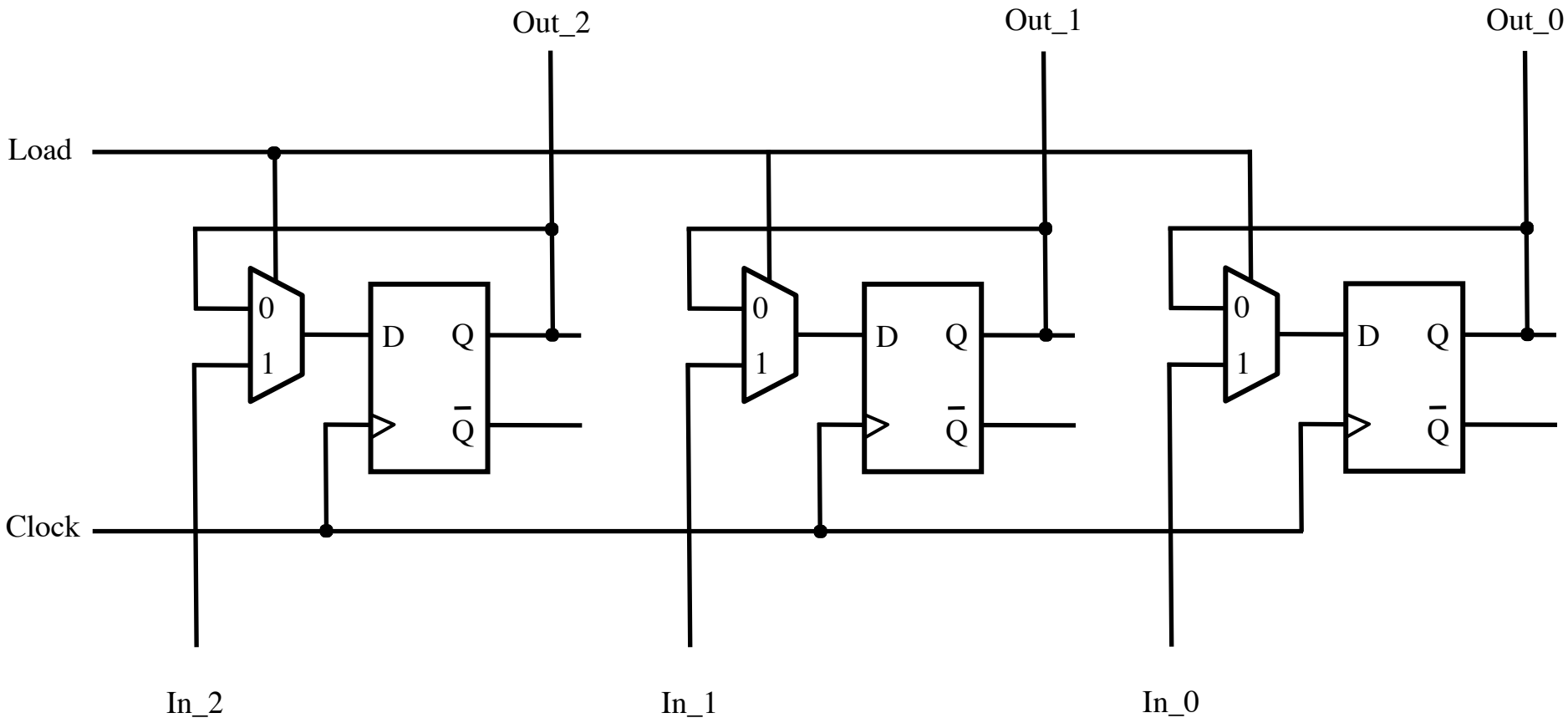
2-Bit Parallel-Access Register



2-Bit Parallel-Access Register

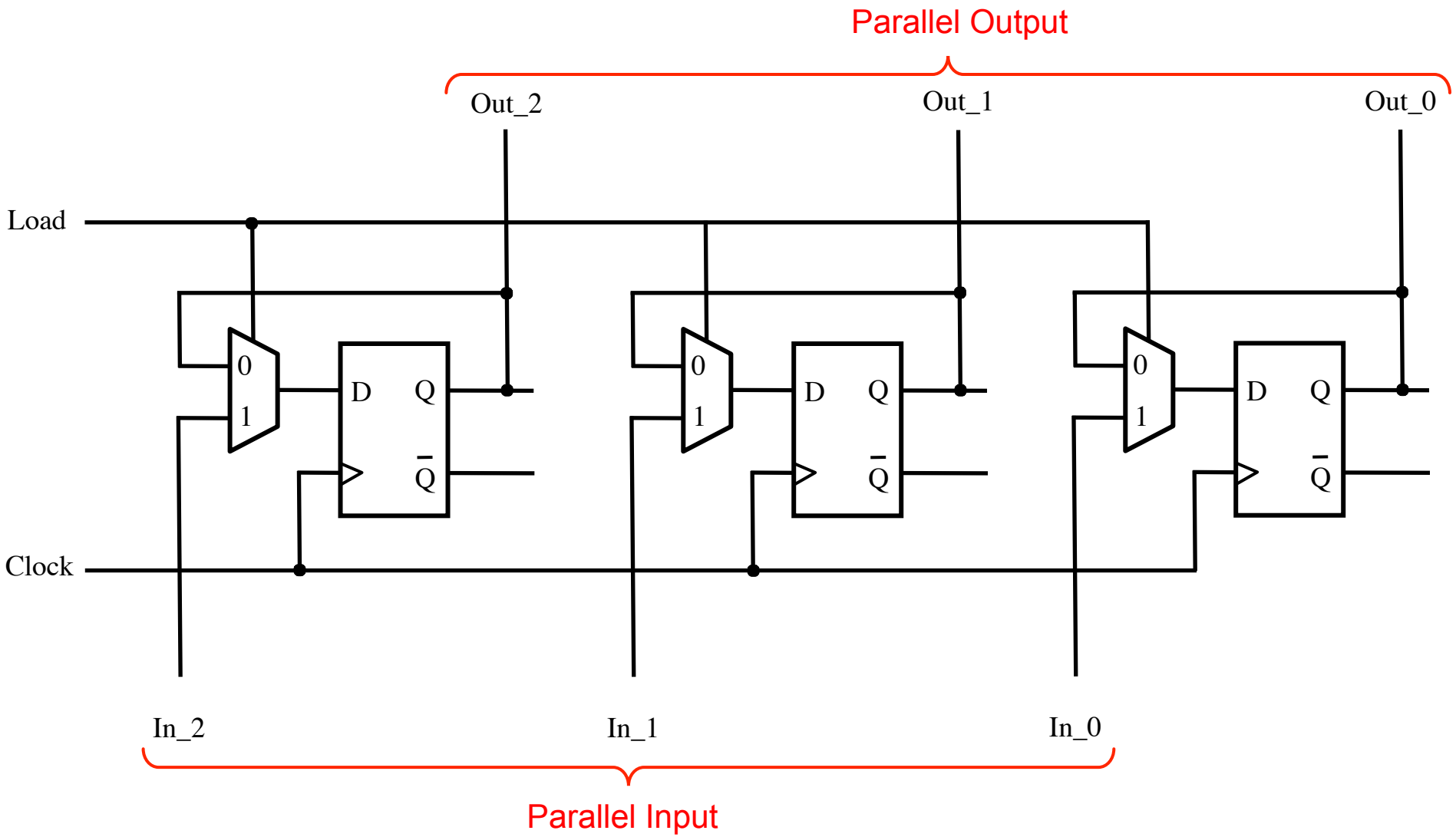


3-Bit Parallel-Access Register

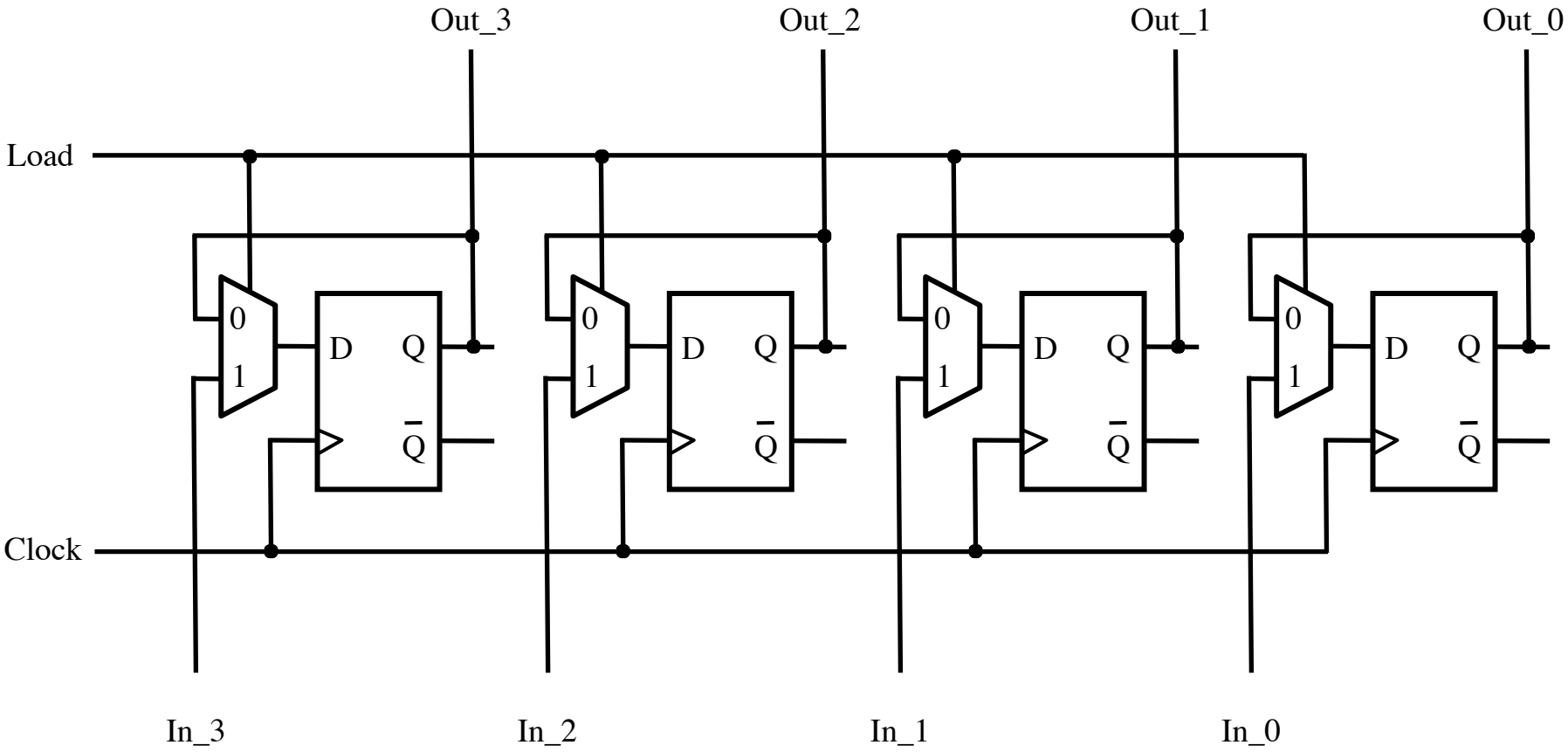


Notice that all flip-flops are on the same clock cycle.

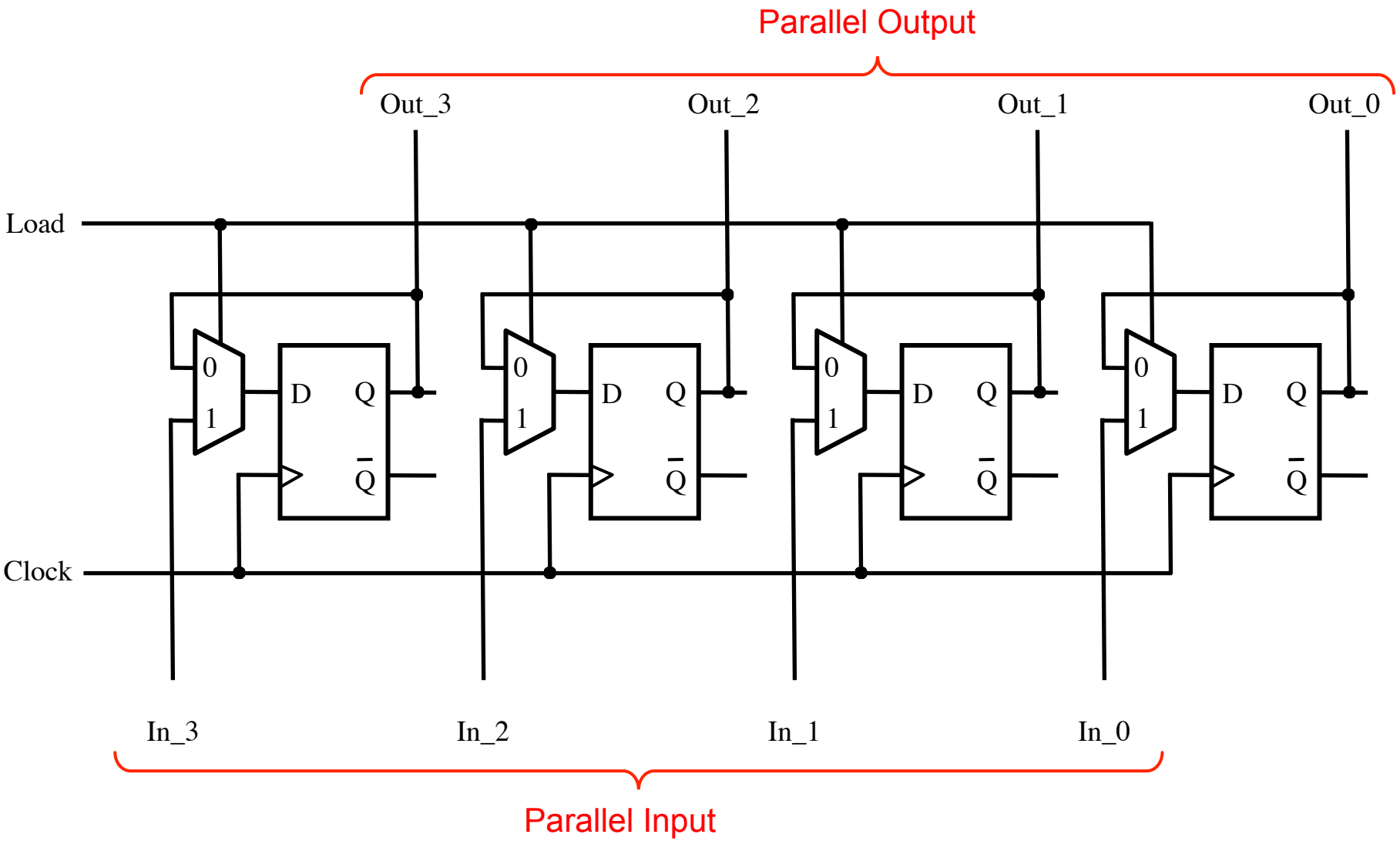
3-Bit Parallel-Access Register



4-Bit Parallel-Access Register

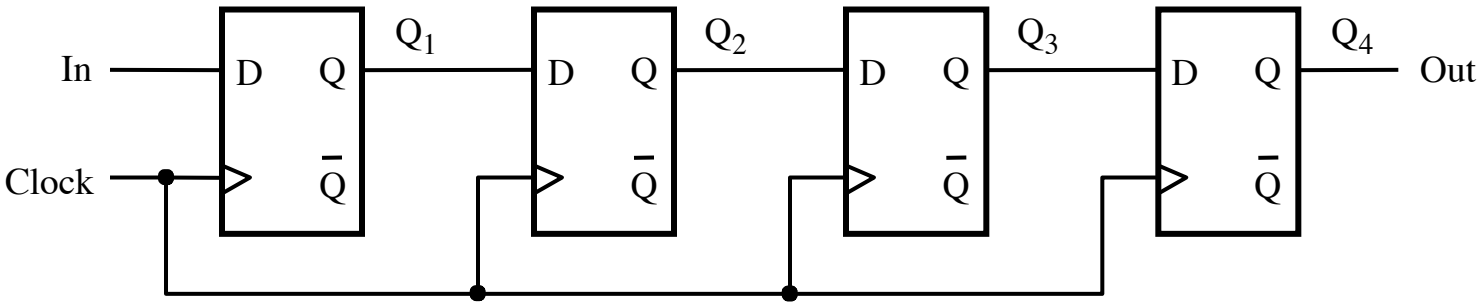


4-Bit Parallel-Access Register



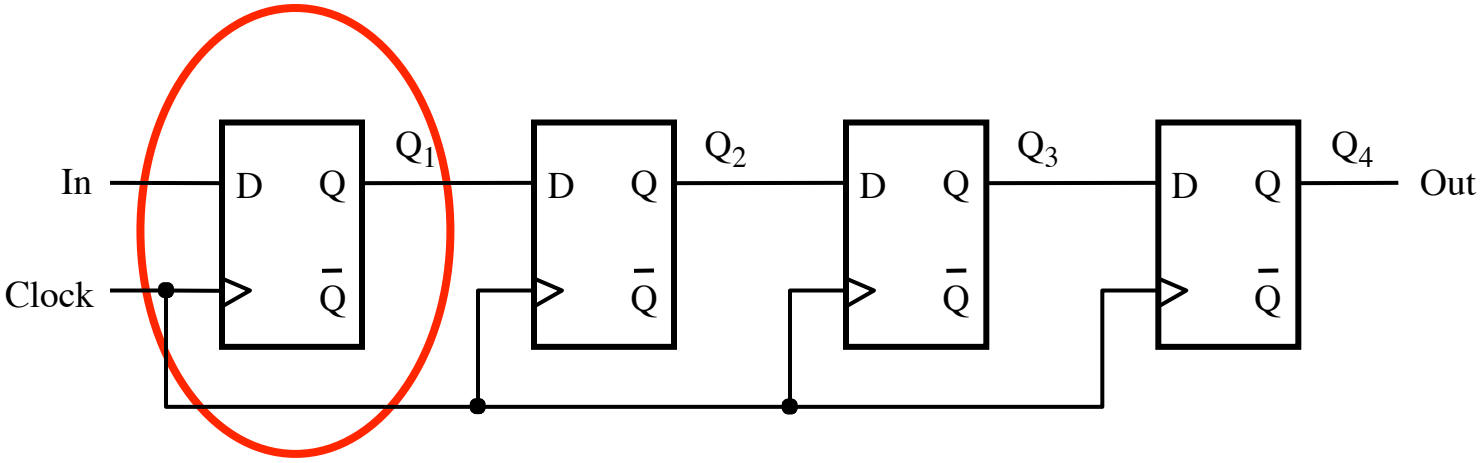
Shift Register

A simple shift register



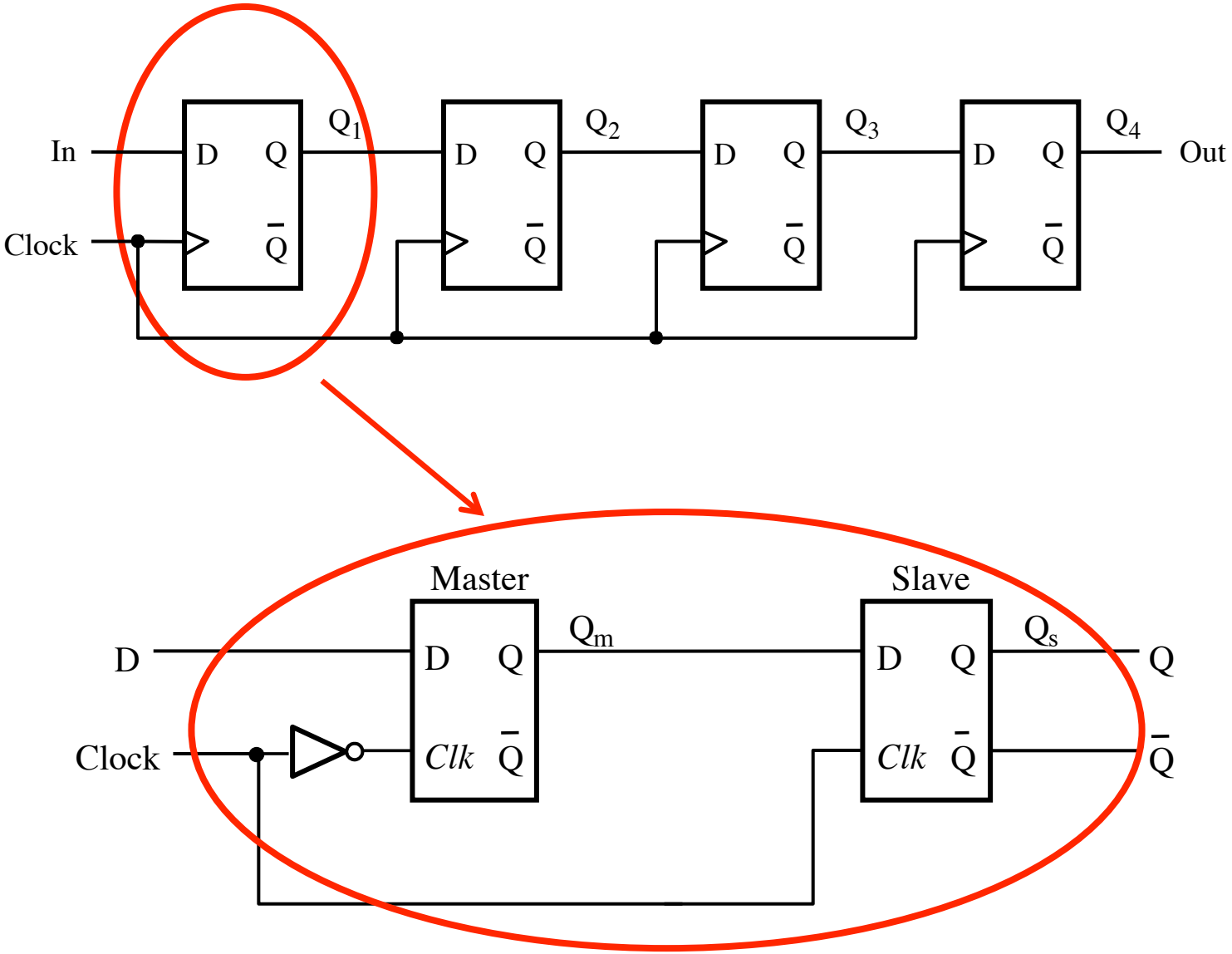
[Figure 5.17a from the textbook]

A simple shift register

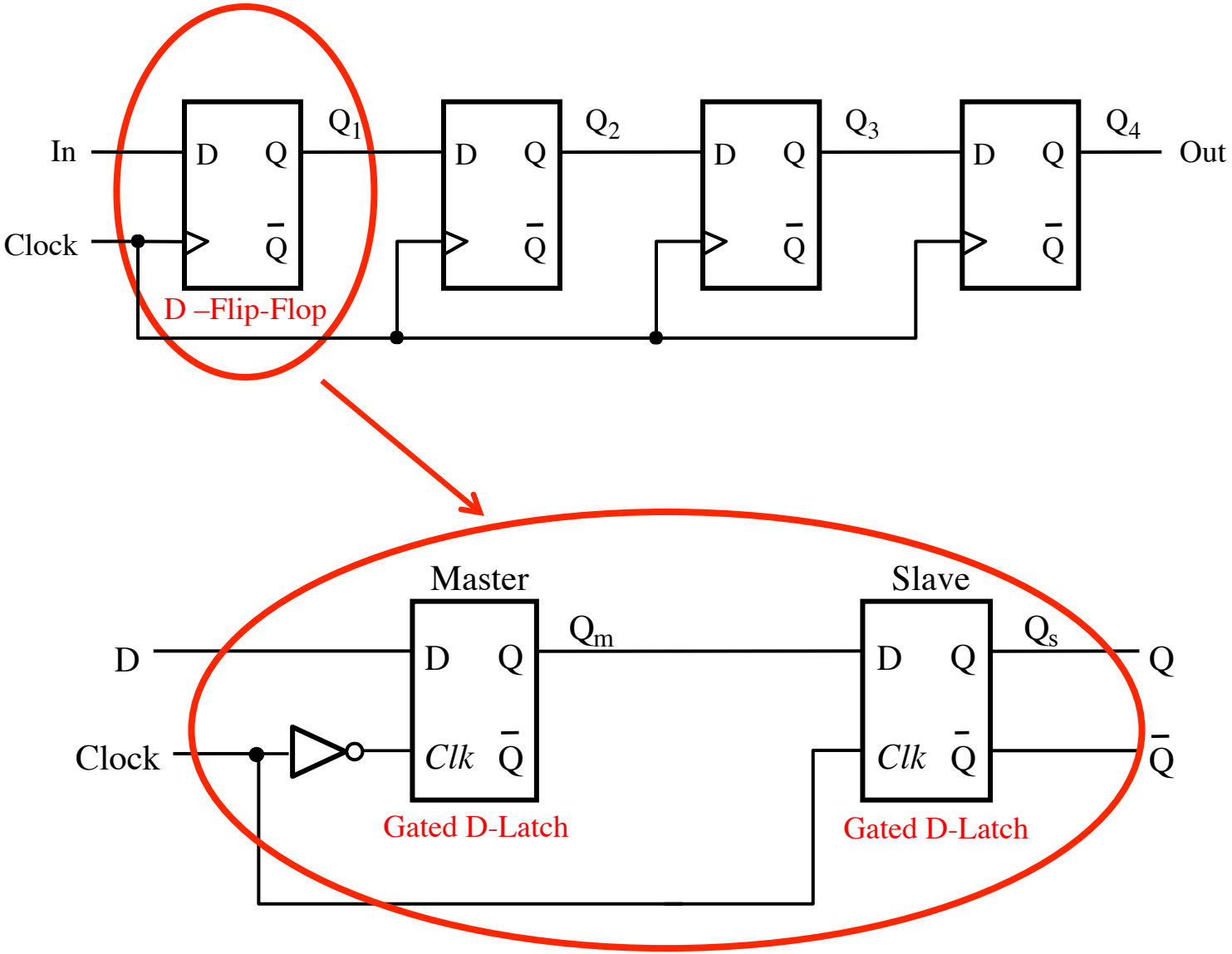


Positive-edge-triggered
D Flip-Flop

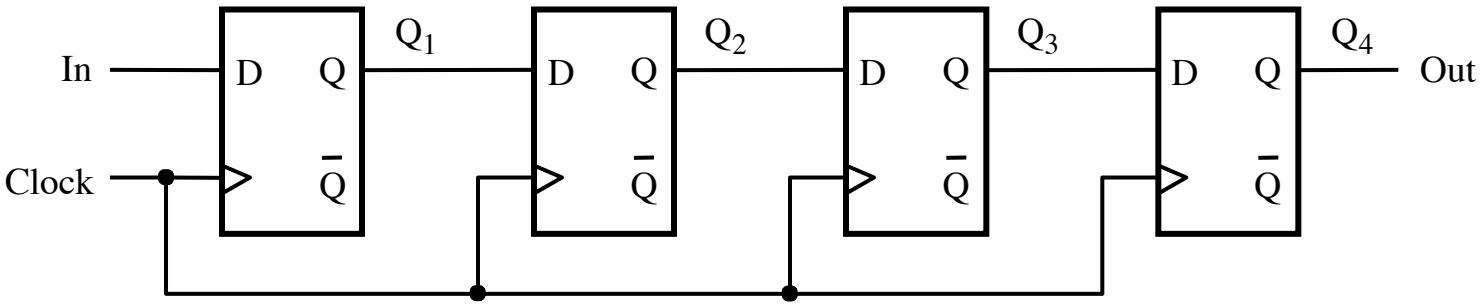
A simple shift register



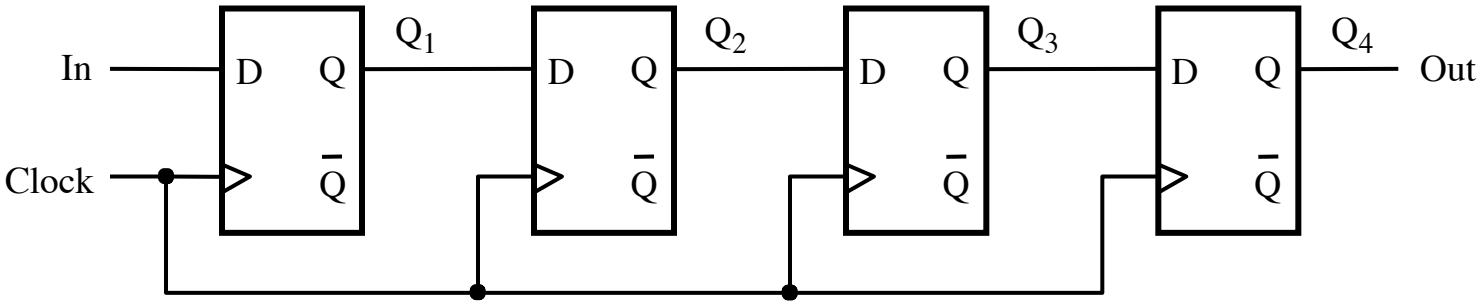
A simple shift register

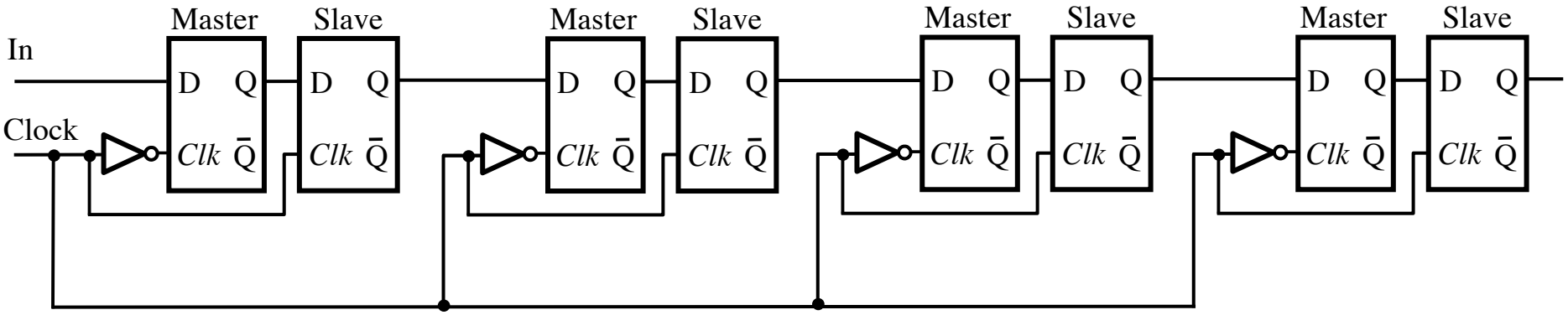


A simple shift register

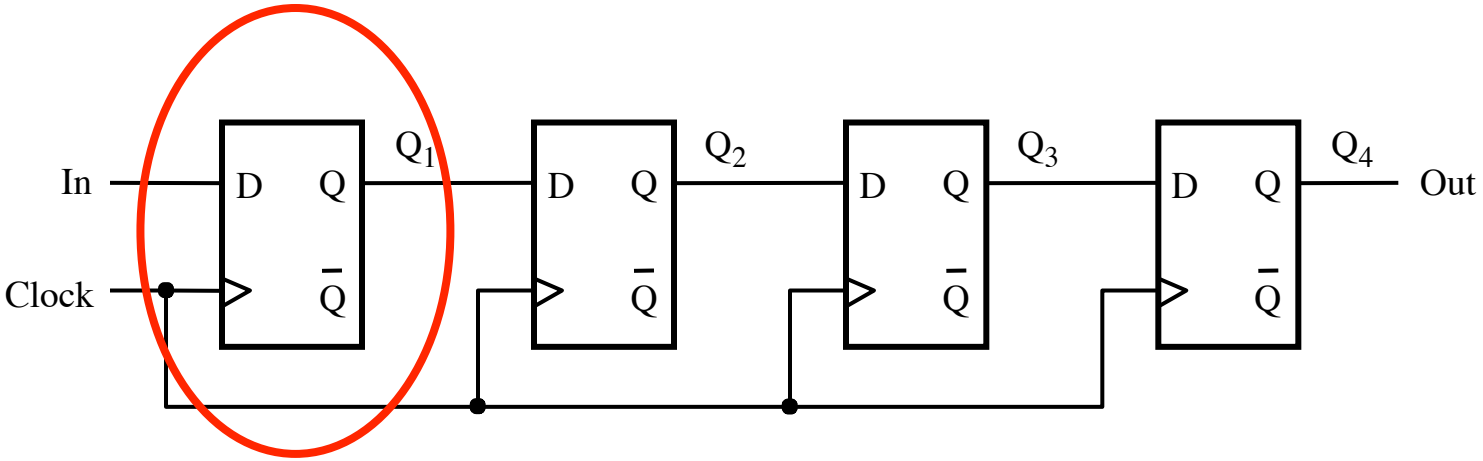


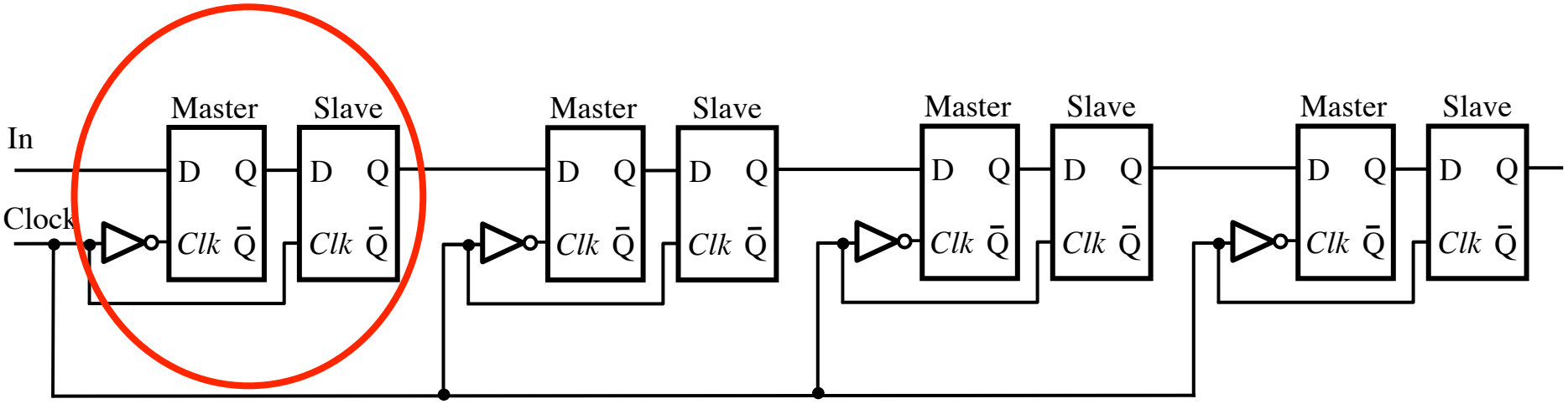
A simple shift register



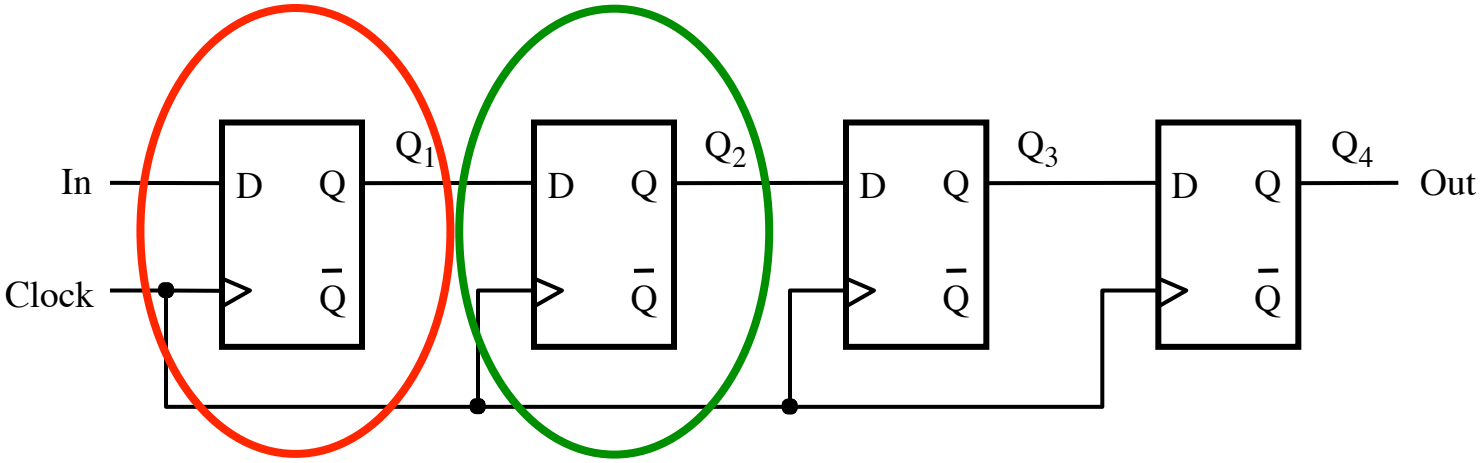


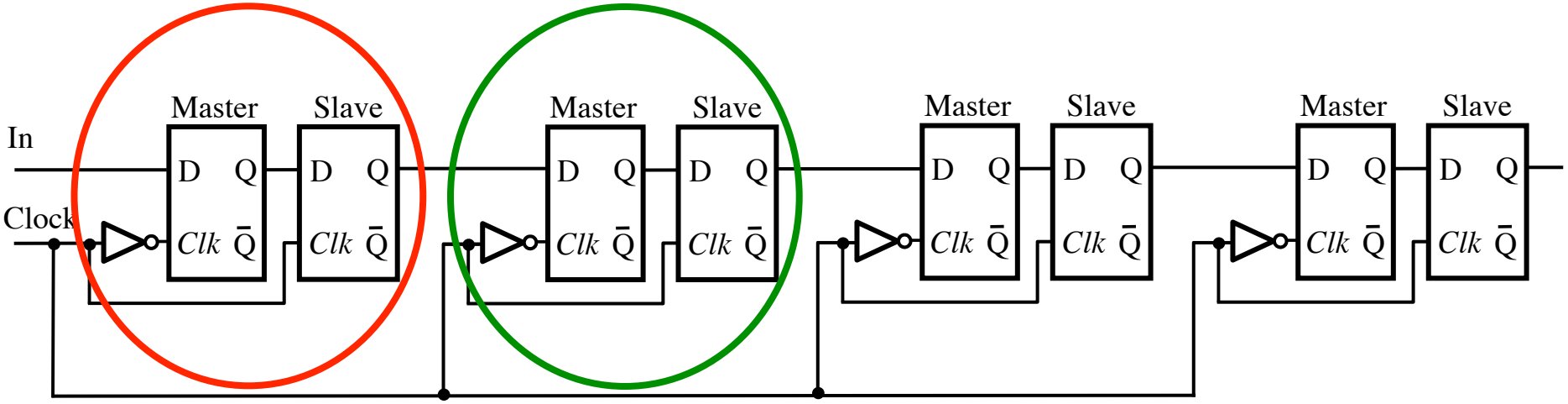
A simple shift register



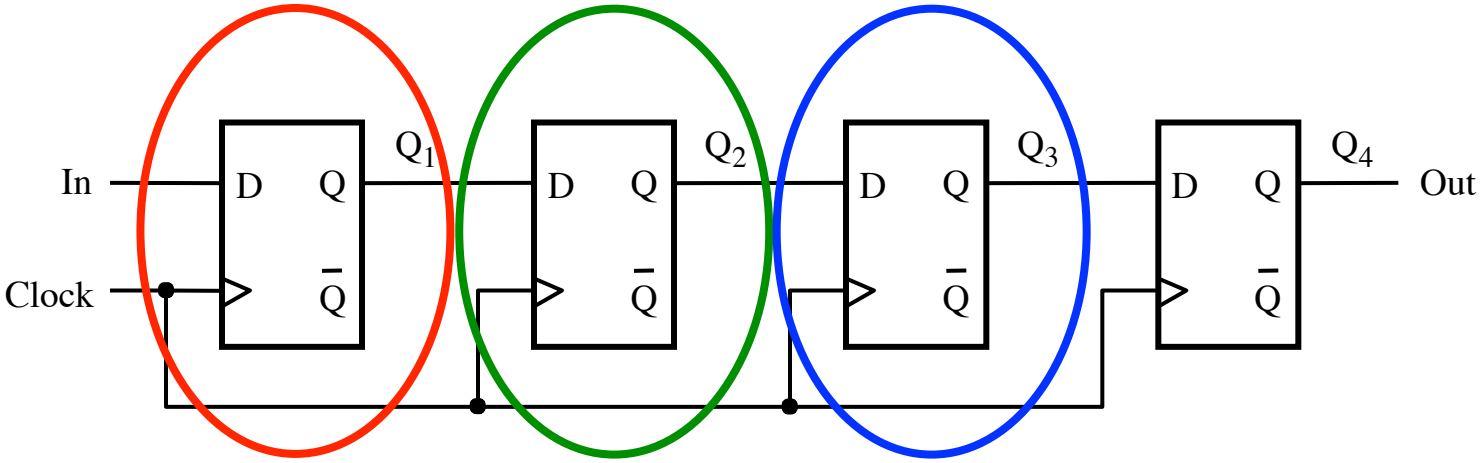


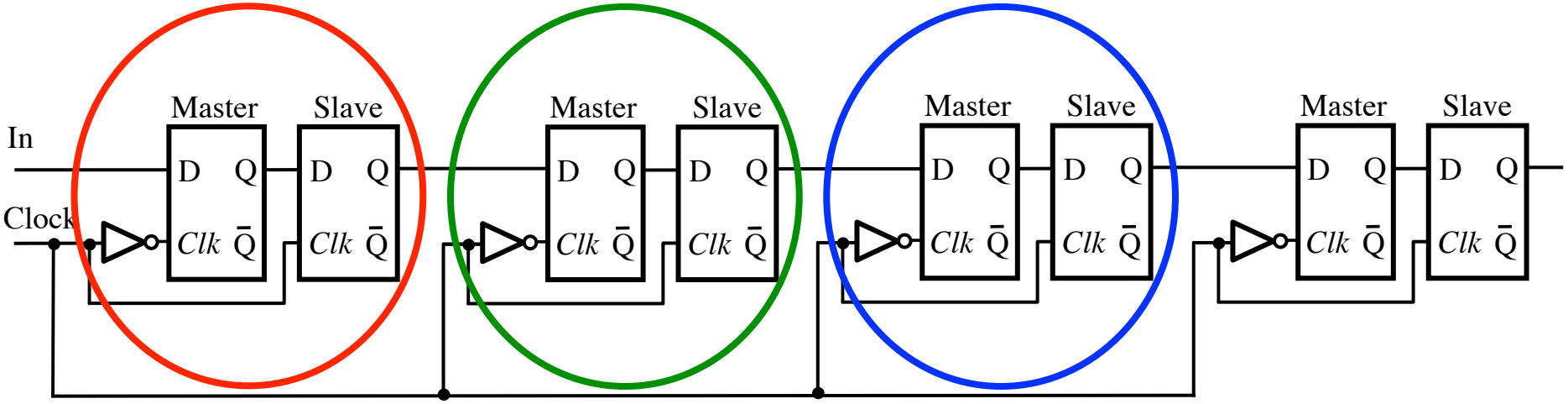
A simple shift register



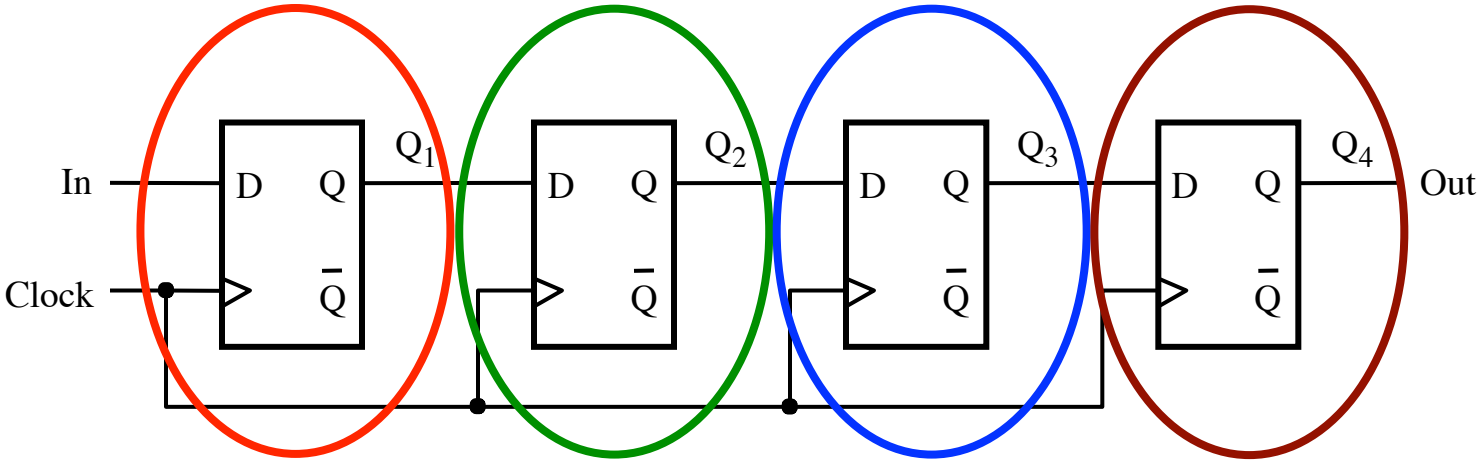


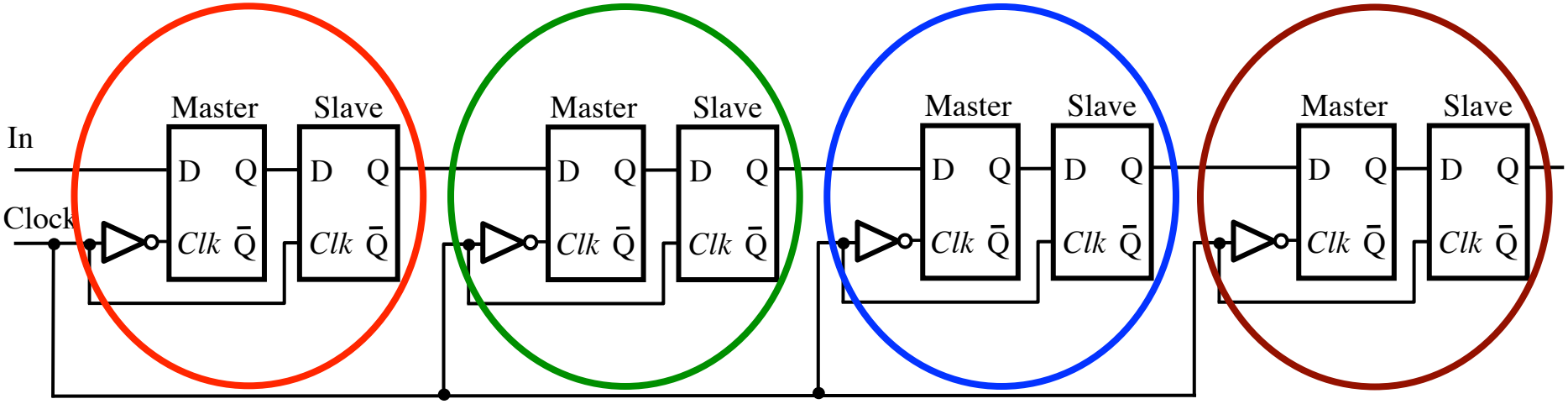
A simple shift register



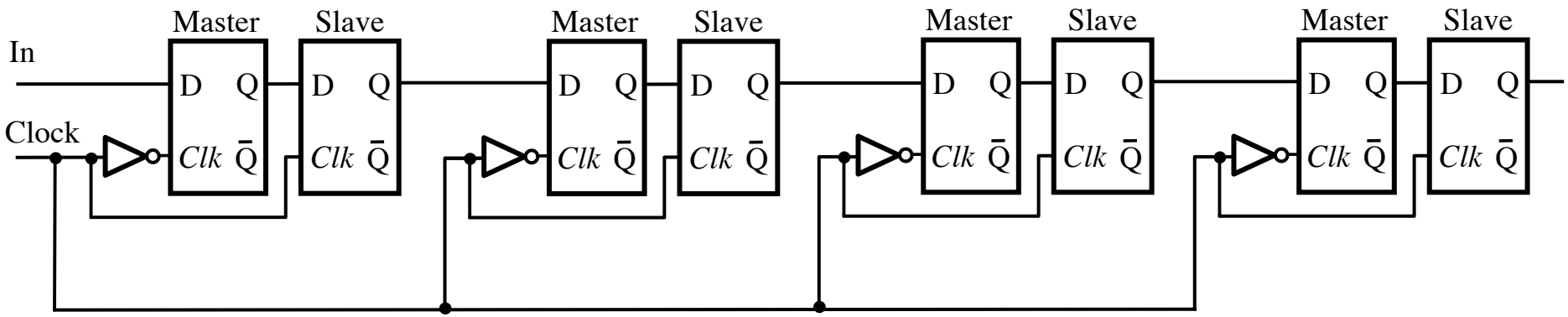


A simple shift register

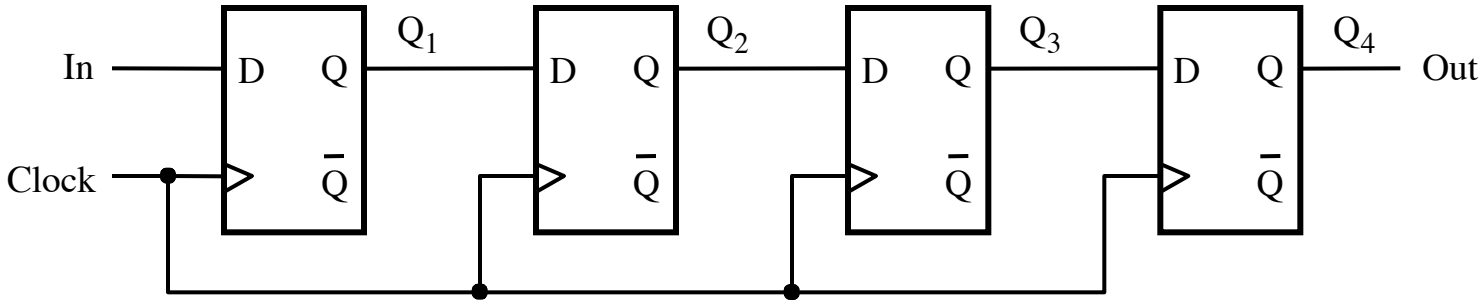




A simple shift register



A simple shift register



(a) Circuit

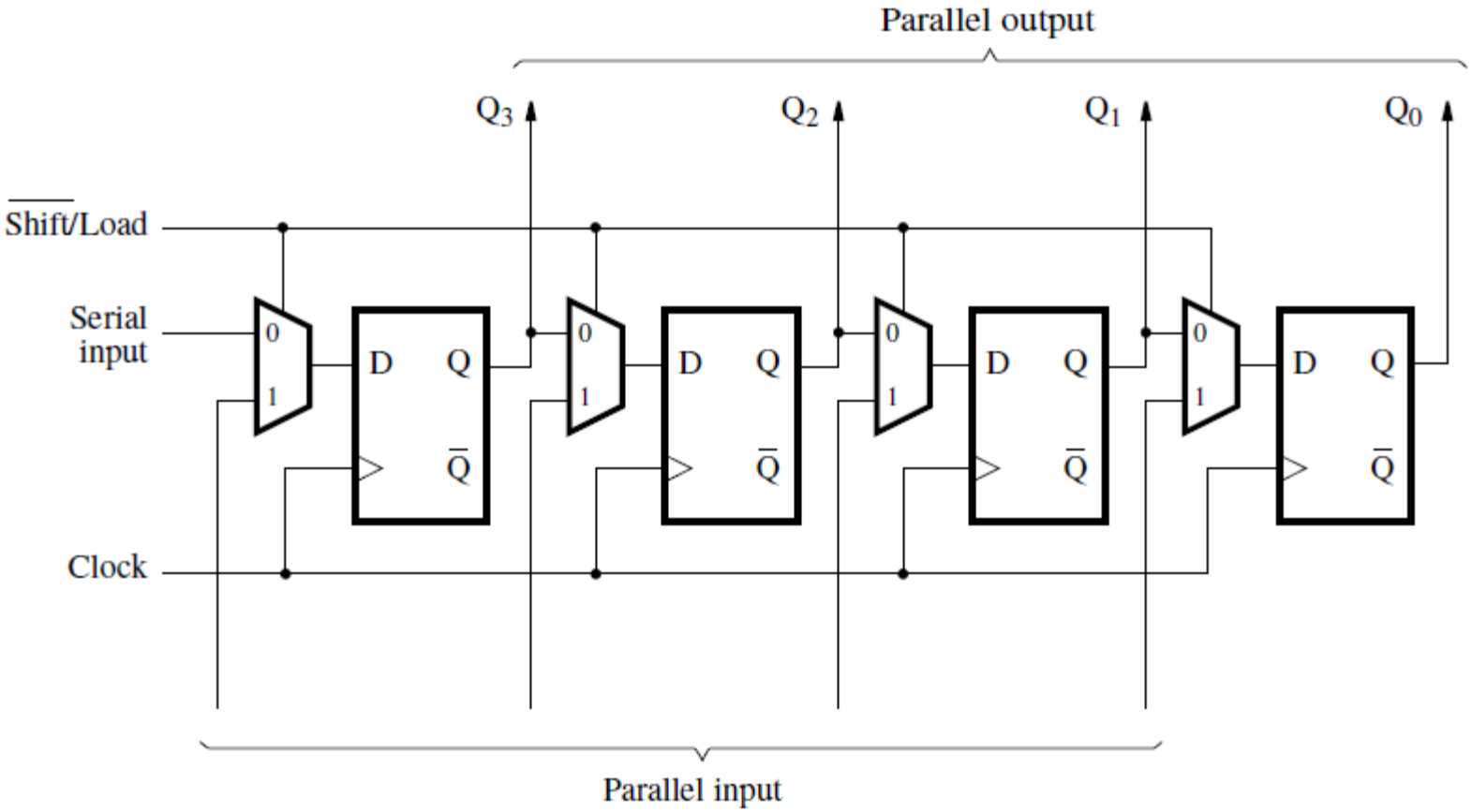
	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

(b) A sample sequence

[Figure 5.17 from the textbook]

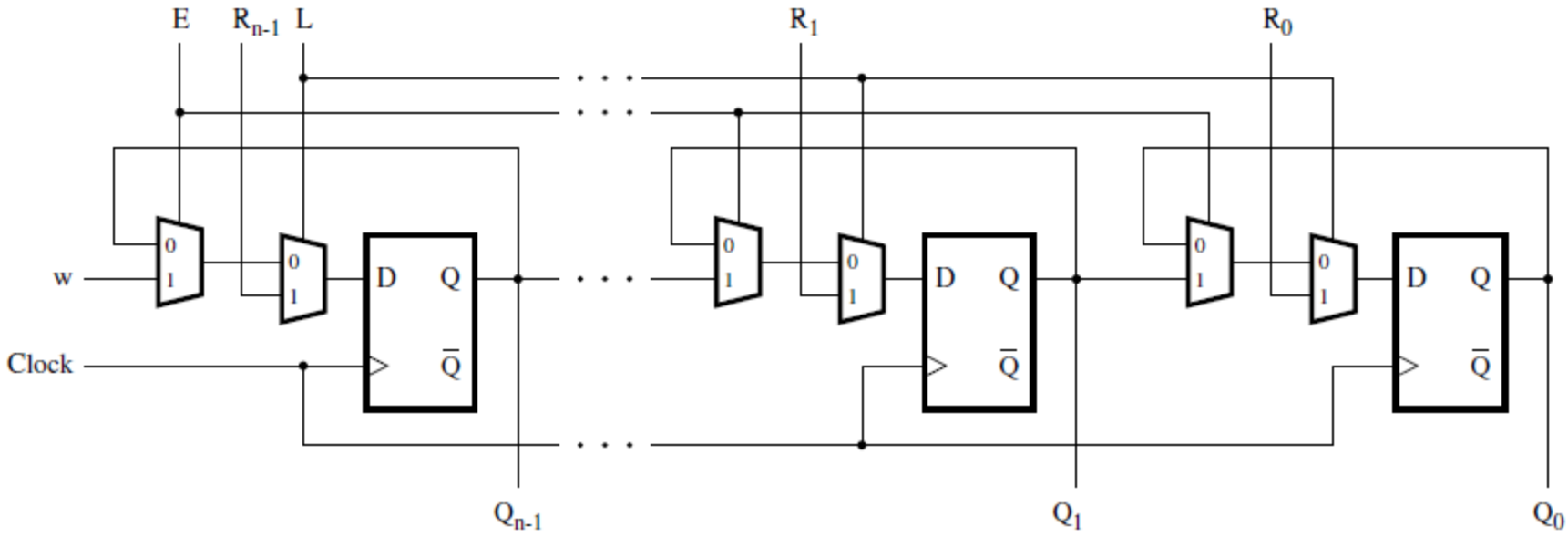
Parallel-Access Shift Register

Parallel-access shift register



[Figure 5.18 from the textbook]

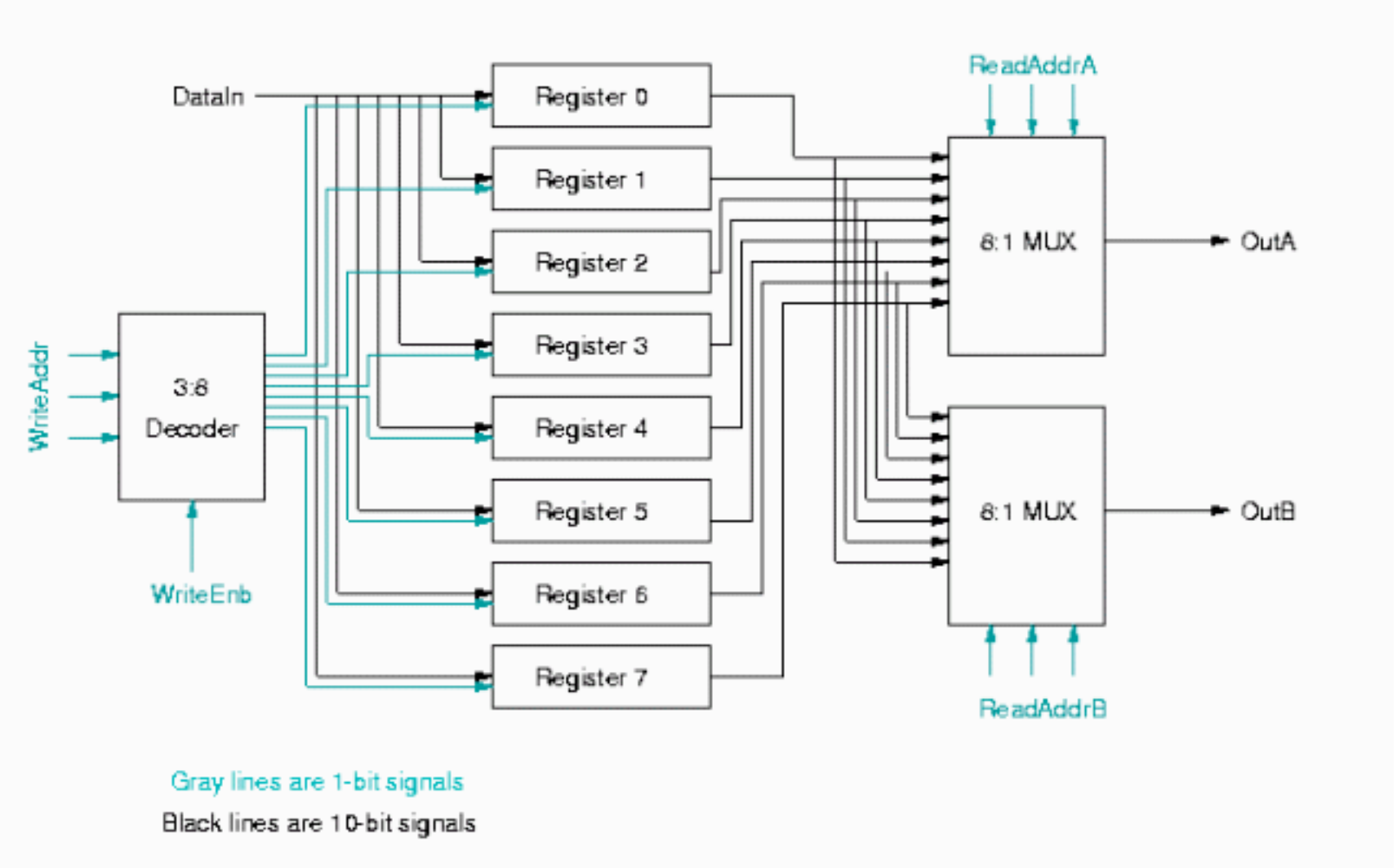
A shift register with parallel load and enable control inputs



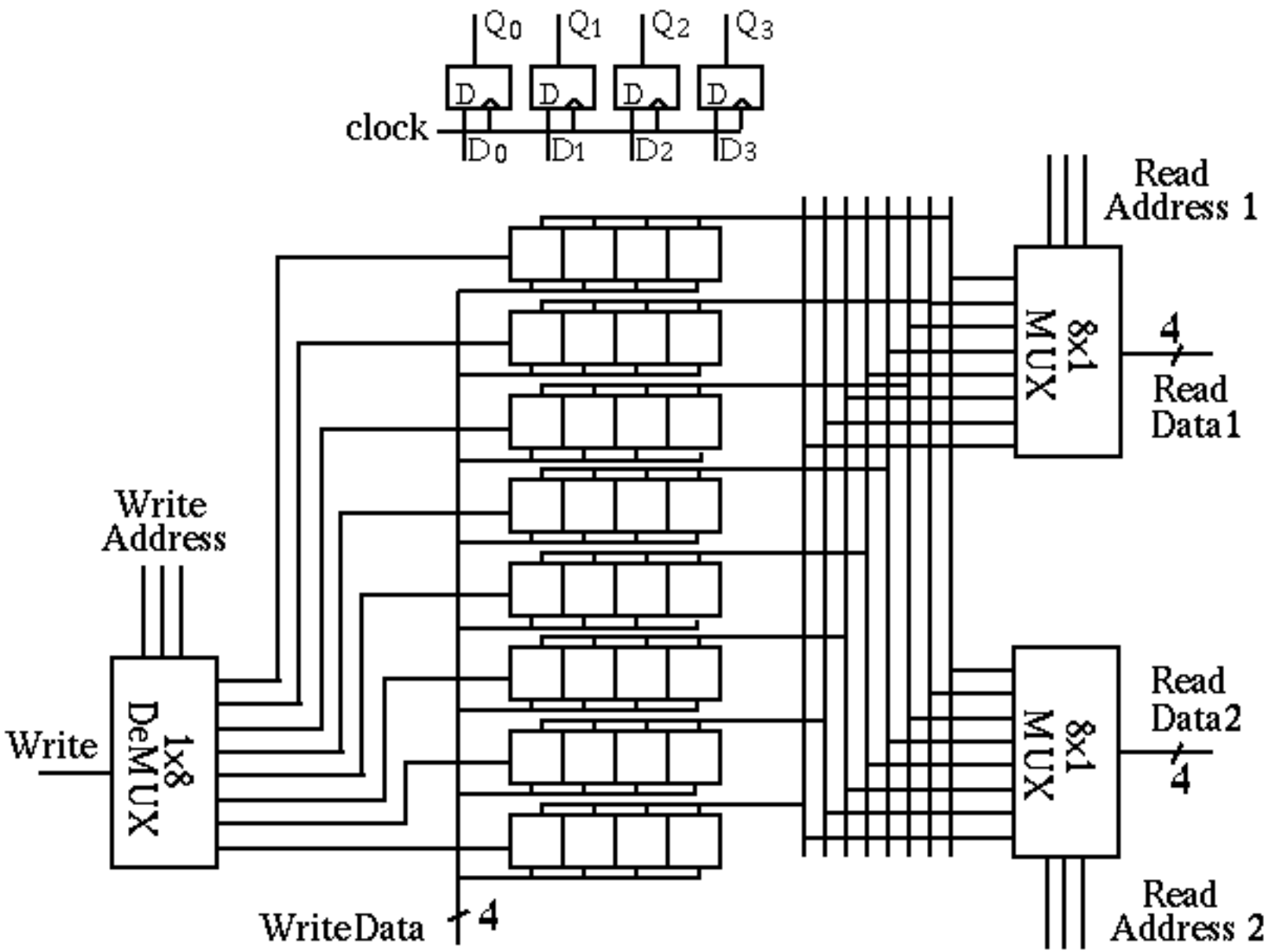
[Figure 5.59 from the textbook]

Register File

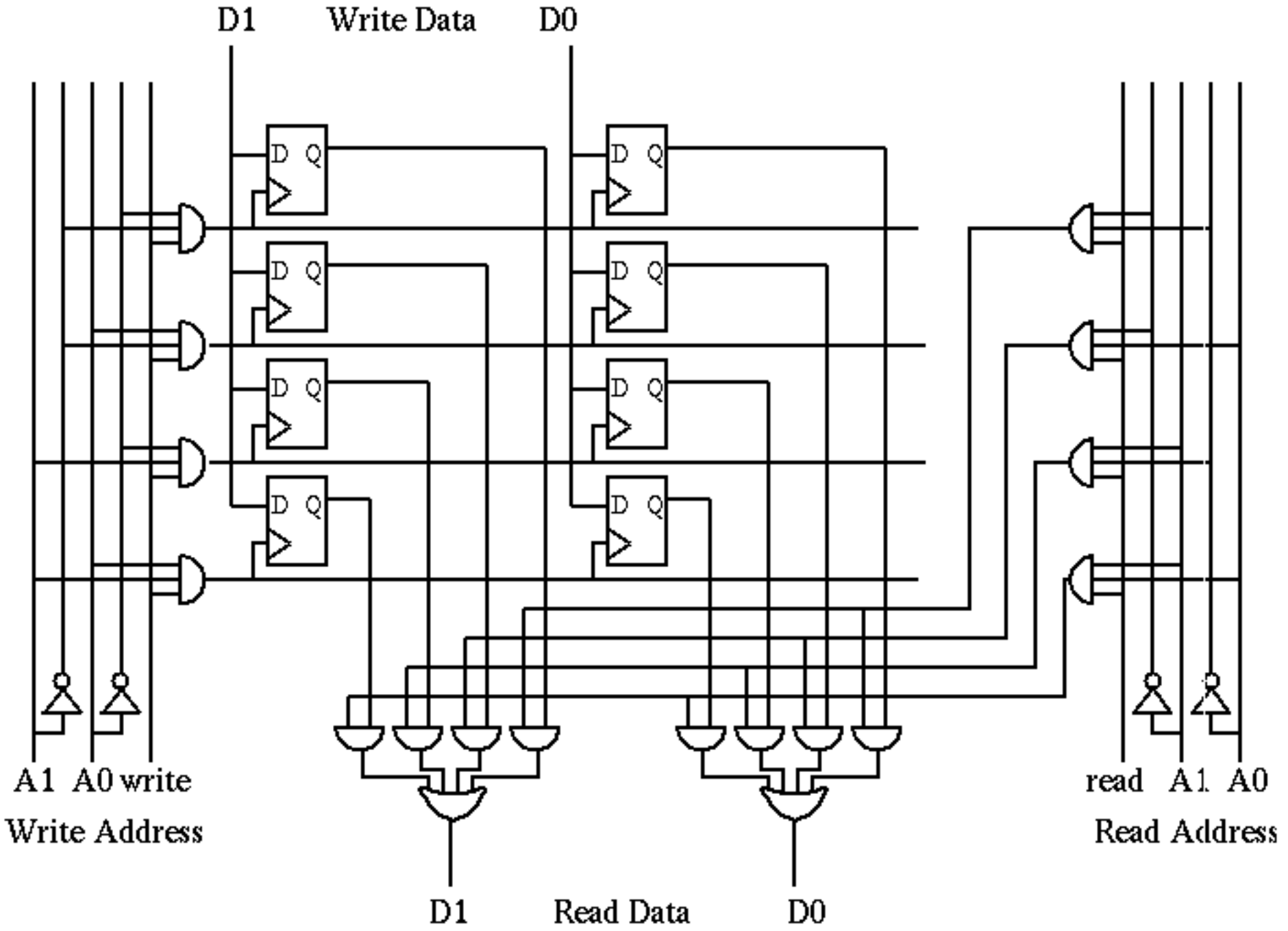
Register File



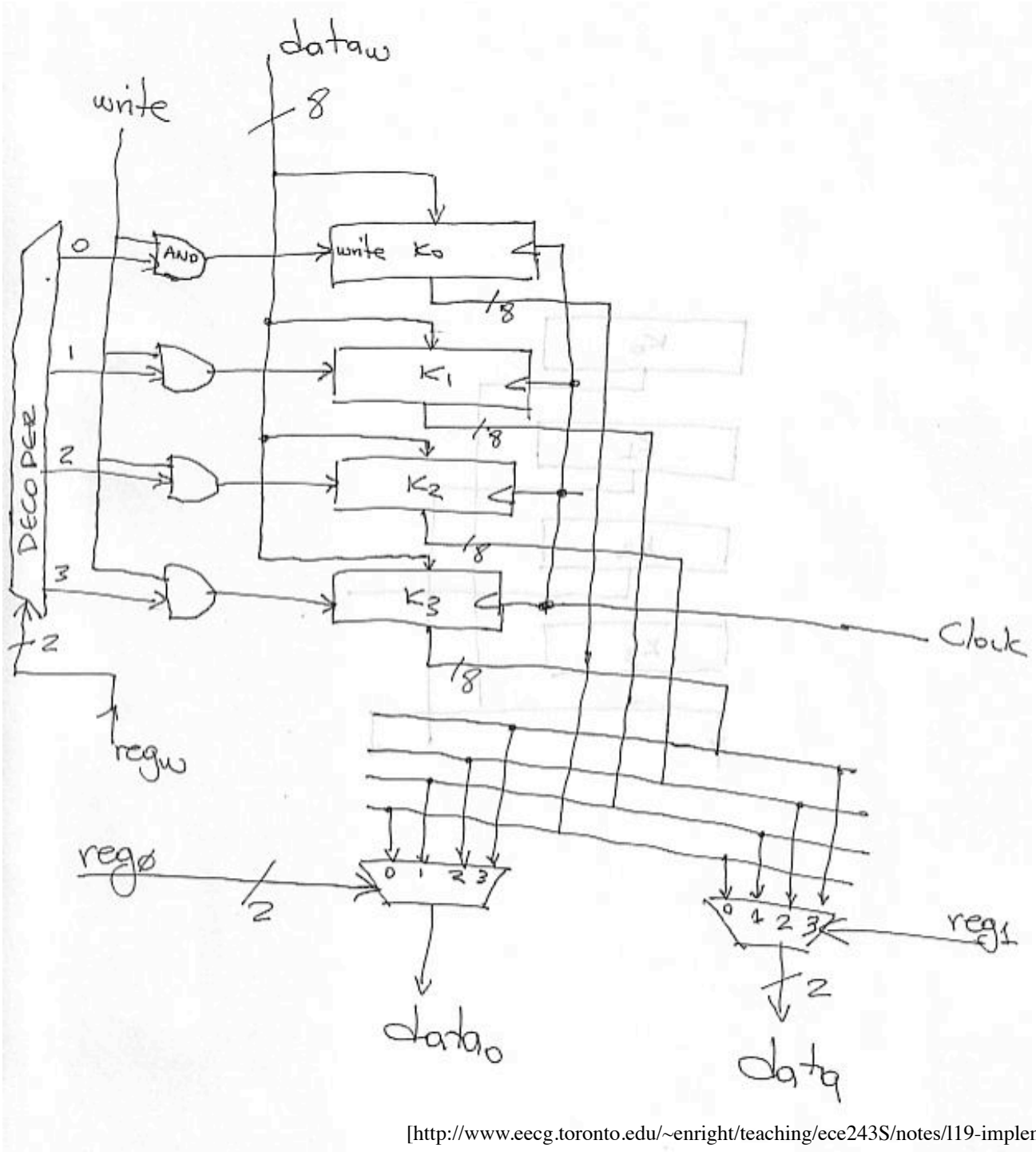
[https://jindongpu.wordpress.com/2012/03/07/register-file/]



[http://fourier.eng.hmc.edu/e85_old/lectures/digital_logic/node19.html]



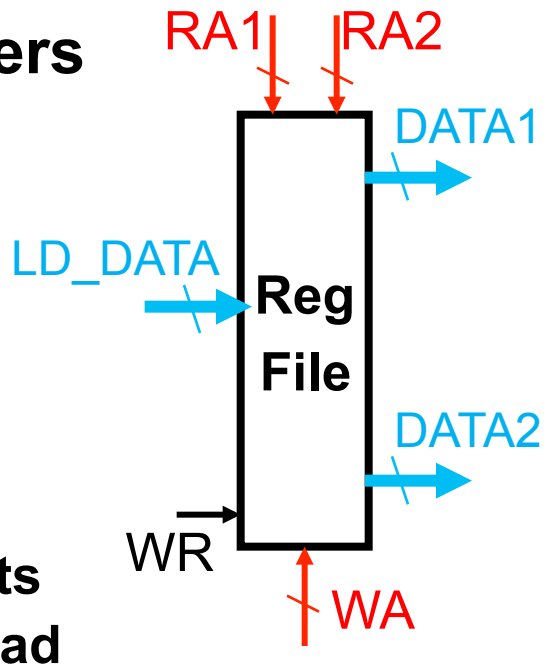
[http://fourier.eng.hmc.edu/e85_old/lectures/digital_logic/node19.html]



[http://www.eecg.toronto.edu/~enright/teaching/ece243S/notes/l19-implemenation-single-cycle.html]

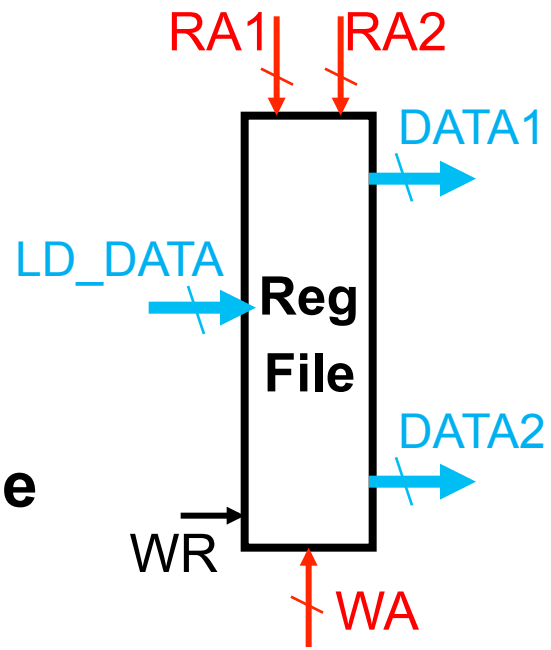
Register File

- **Register file is a unit containing r registers**
  - **r can be 4, 8, 16, 32, etc.**
- **Each register has n bits**
  - **n can be 4, 8, 16, 32, etc.**
  - **n defines the data path width**
- **Output ports (DATA1 and DATA2) are used for reading the register file**
  - **Any register can be read from any of the ports**
  - **Each port needs a $\log_2 r$ bits to specify the read address (RA1 and RA2)**
- **Input port (LD_DATA) is used for writing data to the register file**
  - **Write address is also specified by $\log_2 r$ bits (WA)**
  - **Writing is enabled by a 1-bit signal (WR)**



Register File: Exercise

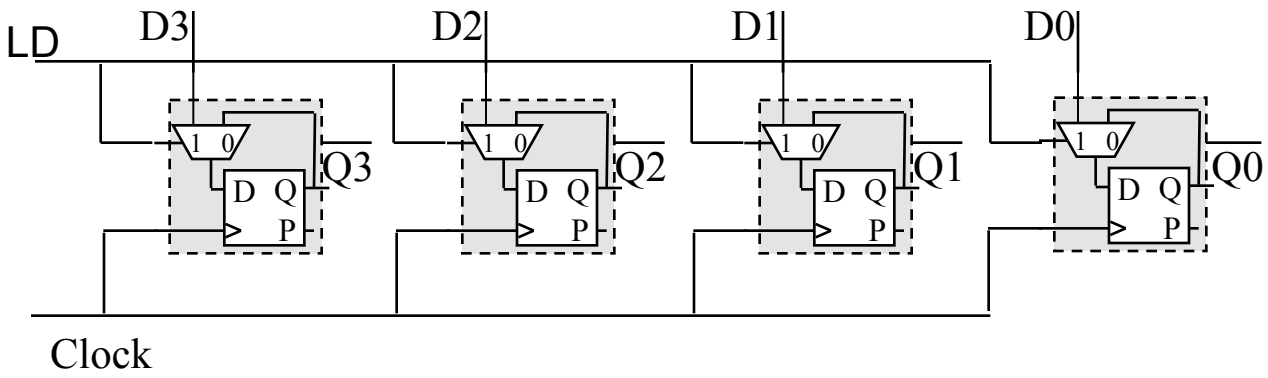
- **Suppose that a register file**
  - **contains 32 registers**
  - **width of data path is 16 bits**
    (i.e., each register has 16 bits)

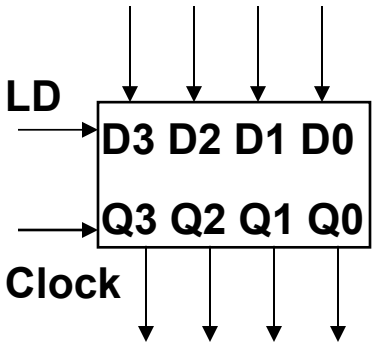


- **How many bits are there for each of the signals?**
  - **RA1** **5**
  - **RA2** **5**
  - **DATA1** **16**
  - **DATA2** **16**
  - **WA** **5**
  - **LD_DATA** **16**
  - **WR** **1**

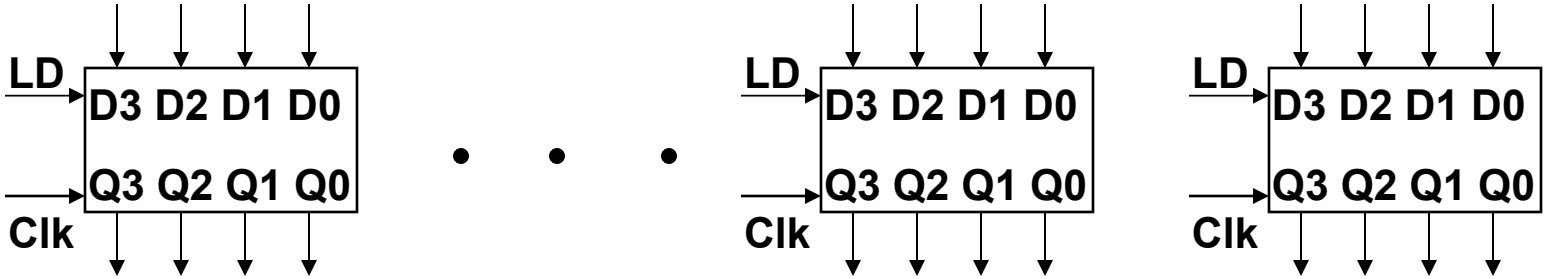
Register file design

- We will design an eight-register file with 4-bit wide registers
- A single 4-bit register and its abstraction are shown below





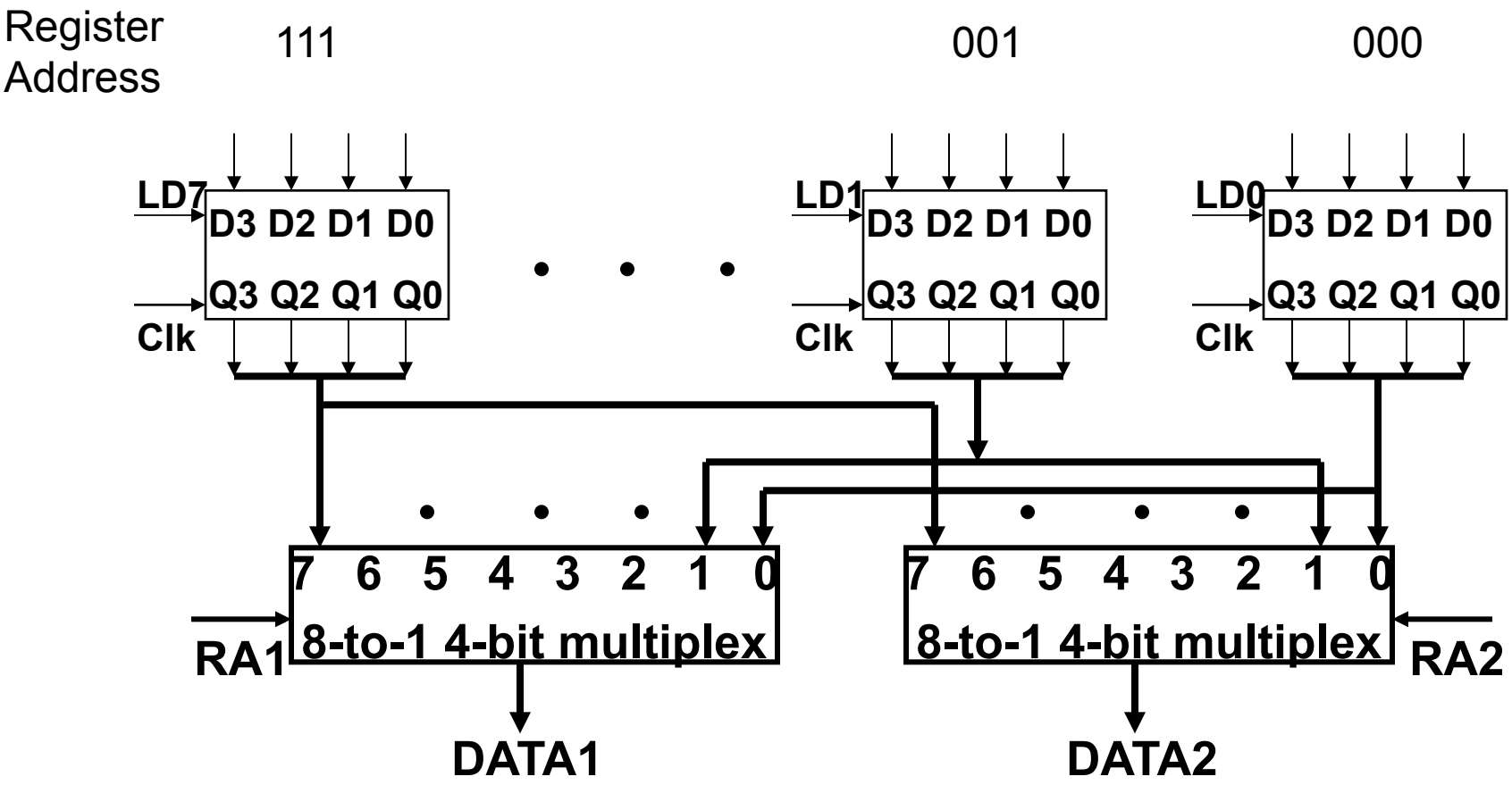
- We have to use eight such registers to make an eight register file



- How many bits are required to specify a register address?

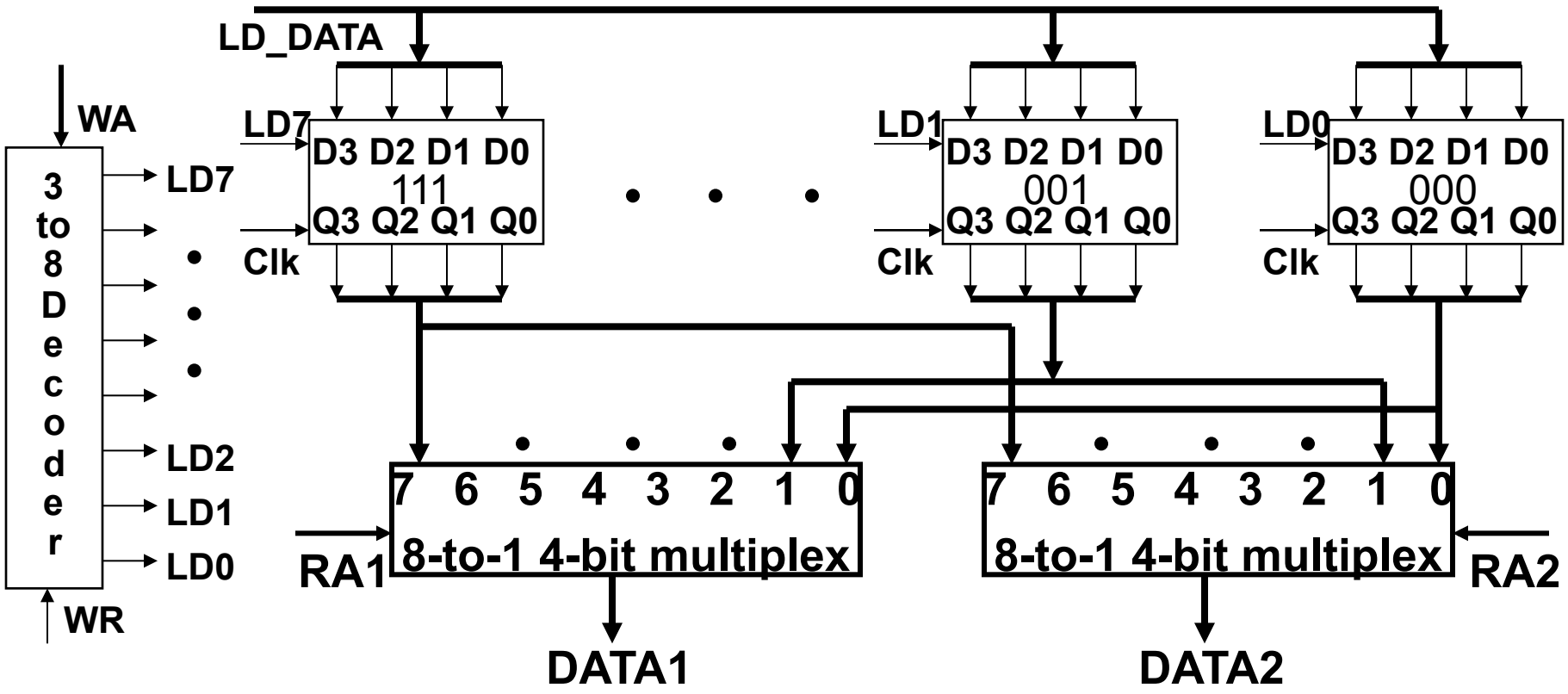
Reading Circuit

- **A 3-bit register address, RA, specifies which register is to be read**
- **For each output port, we need one 8-to-1 4-bit multiplier**



Adding write control to register file

- To write to any register, we need the register's address (WA) and a write register signal (WR)
- A 3-bit write address is decoded if write register signal is present
- One of the eight registers gets a LD signal from the decoder



Questions?

THE END