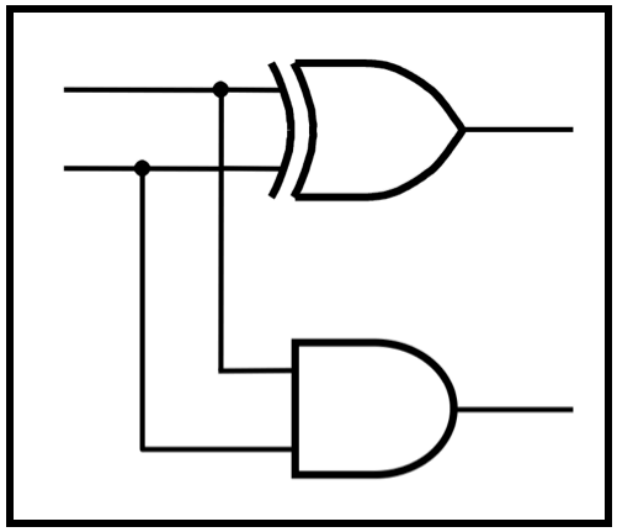


CprE 281: Digital Logic

Instructor: Alexander Stoytchev

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

Registers

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

Administrative Stuff

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

Administrative Stuff

- **Midterm Exam #2**
- **When: Friday October 26 @ 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 for this exam**
- **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

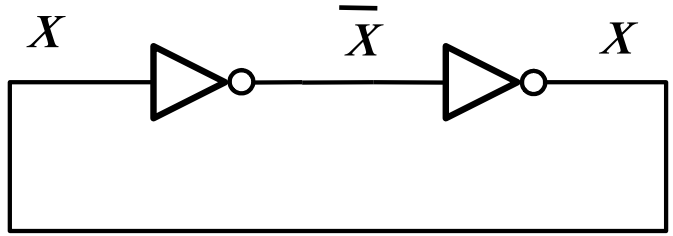
- Decoders (circuits and function)
- Demultiplexers
- Encoders (binary and priority)
- Code Converters
- K-maps for 2, 3, and 4 variables

- 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

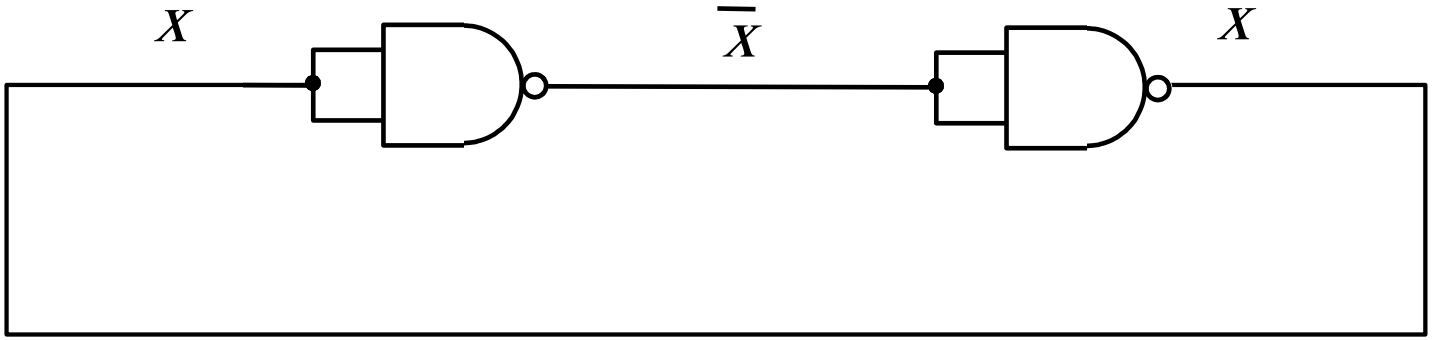
- Latches (circuits, behavior, timing diagrams)
- Flip-Flops (circuits, behavior, timing diagrams)
- Registers and Register Files

Review of Flip-Flops

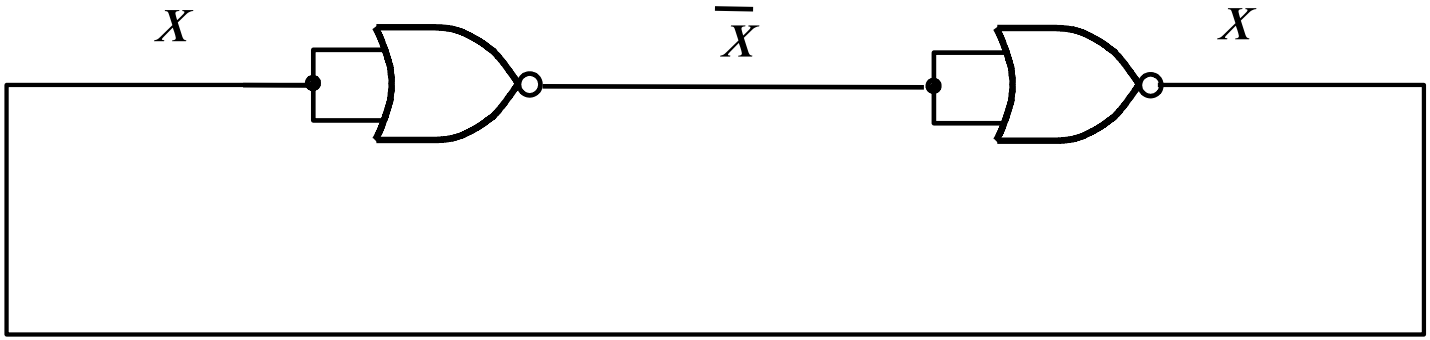
A simple memory element with NOT Gates



A simple memory element with NAND Gates

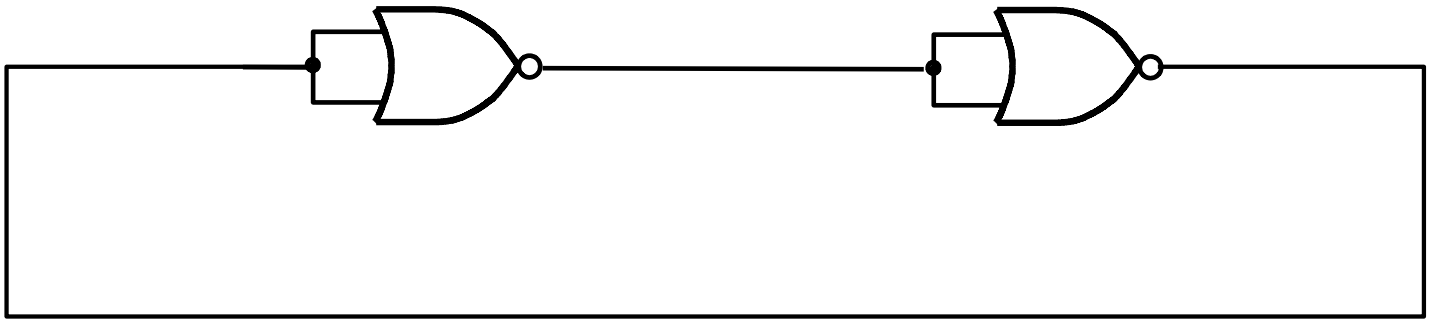


A simple memory element with NOR Gates

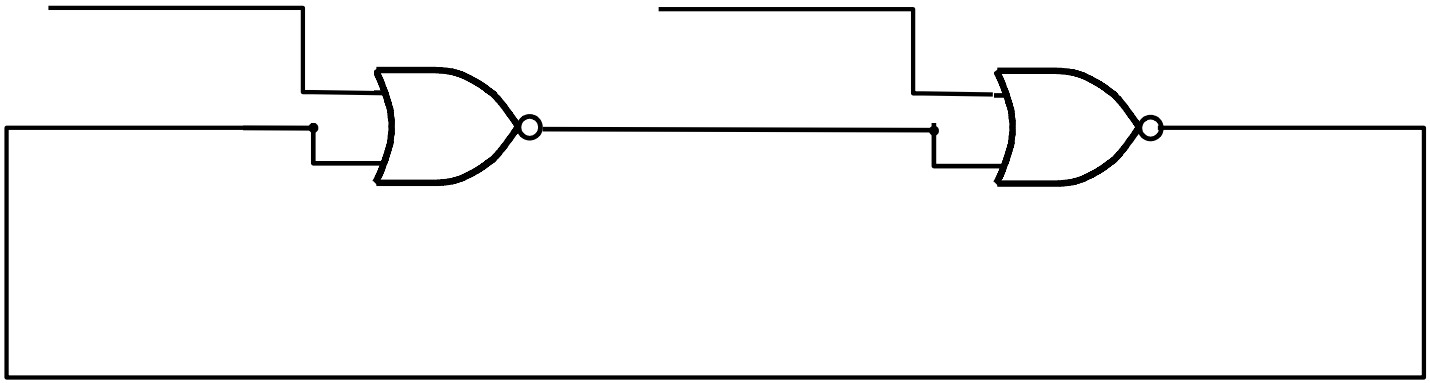


Basic Latch

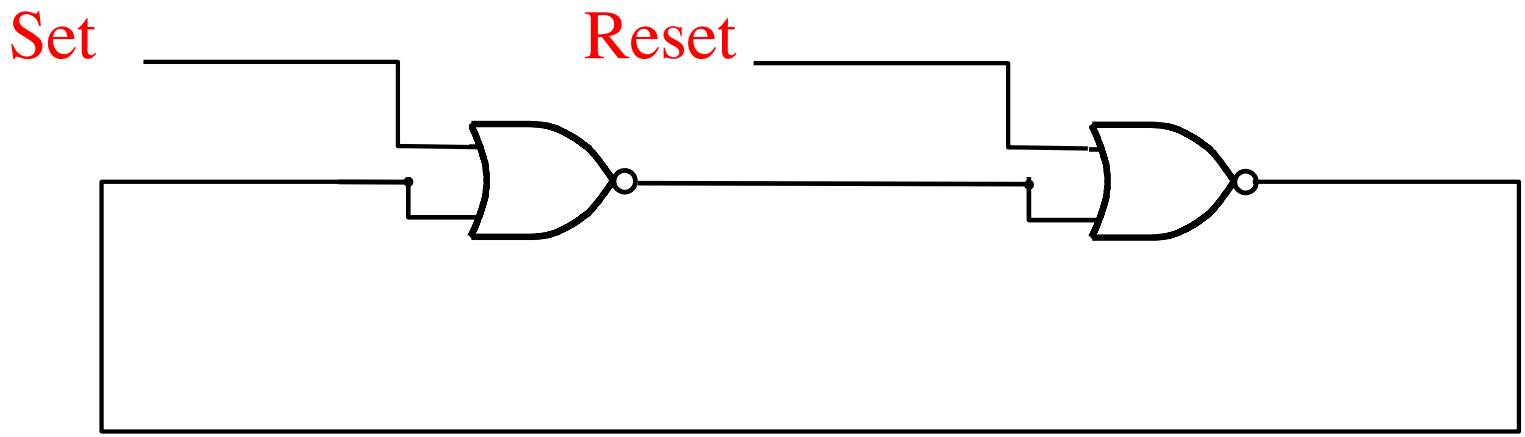
A simple memory element with NOR Gates



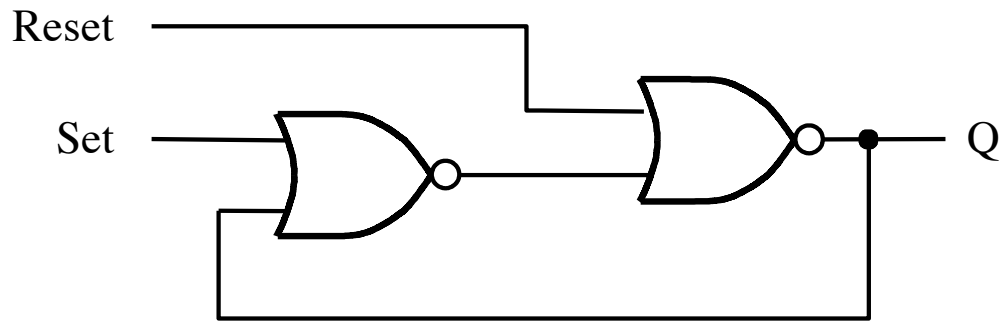
A simple memory element with NOR Gates



A simple memory element with NOR Gates

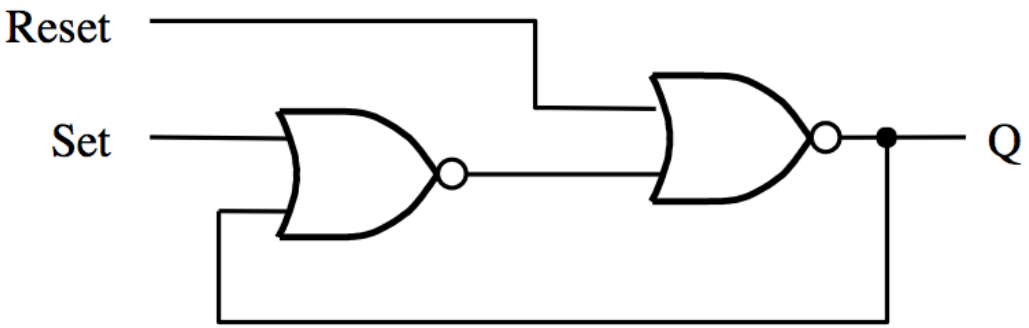


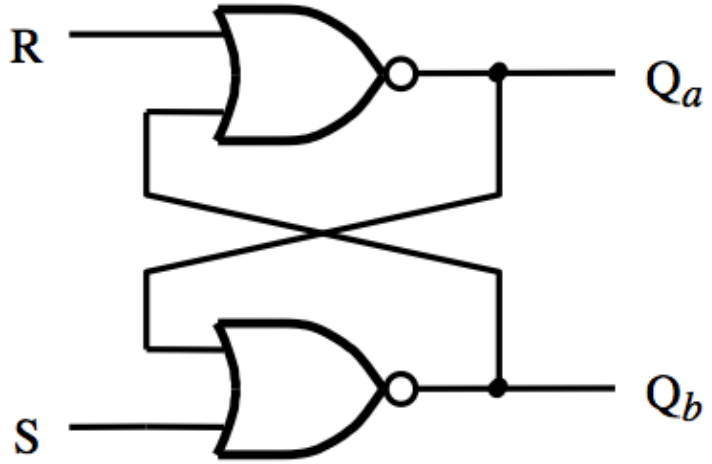
A memory element with NOR gates



[Figure 5.3 from the textbook]

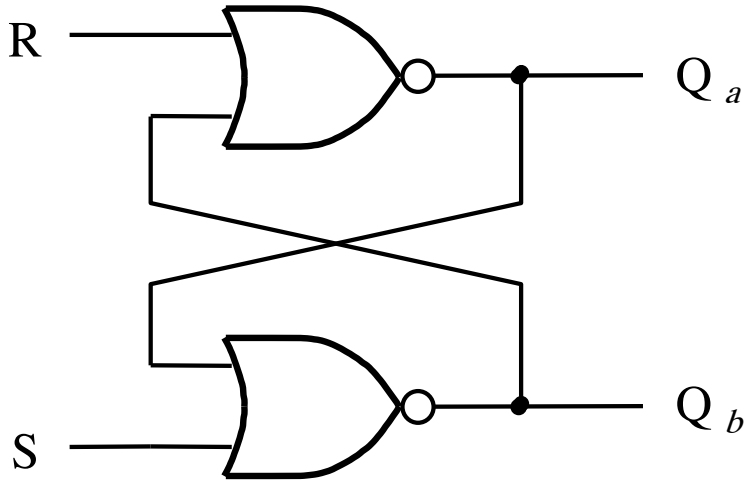
Two Different Ways to Draw the Same Circuit





[Figure 5.3 & 5.4 from the textbook]

SR Latch: Circuit and Truth Table



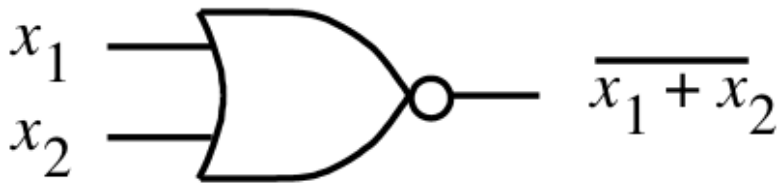
(a) Circuit

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

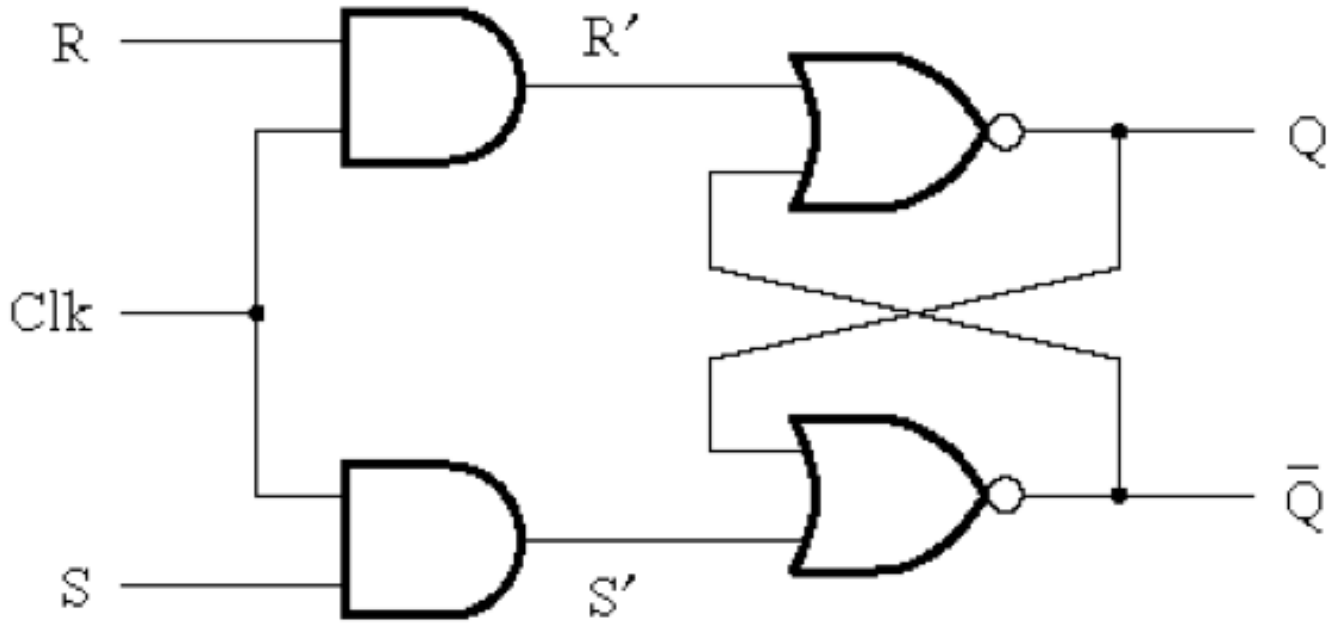


NOR Gate Truth table

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

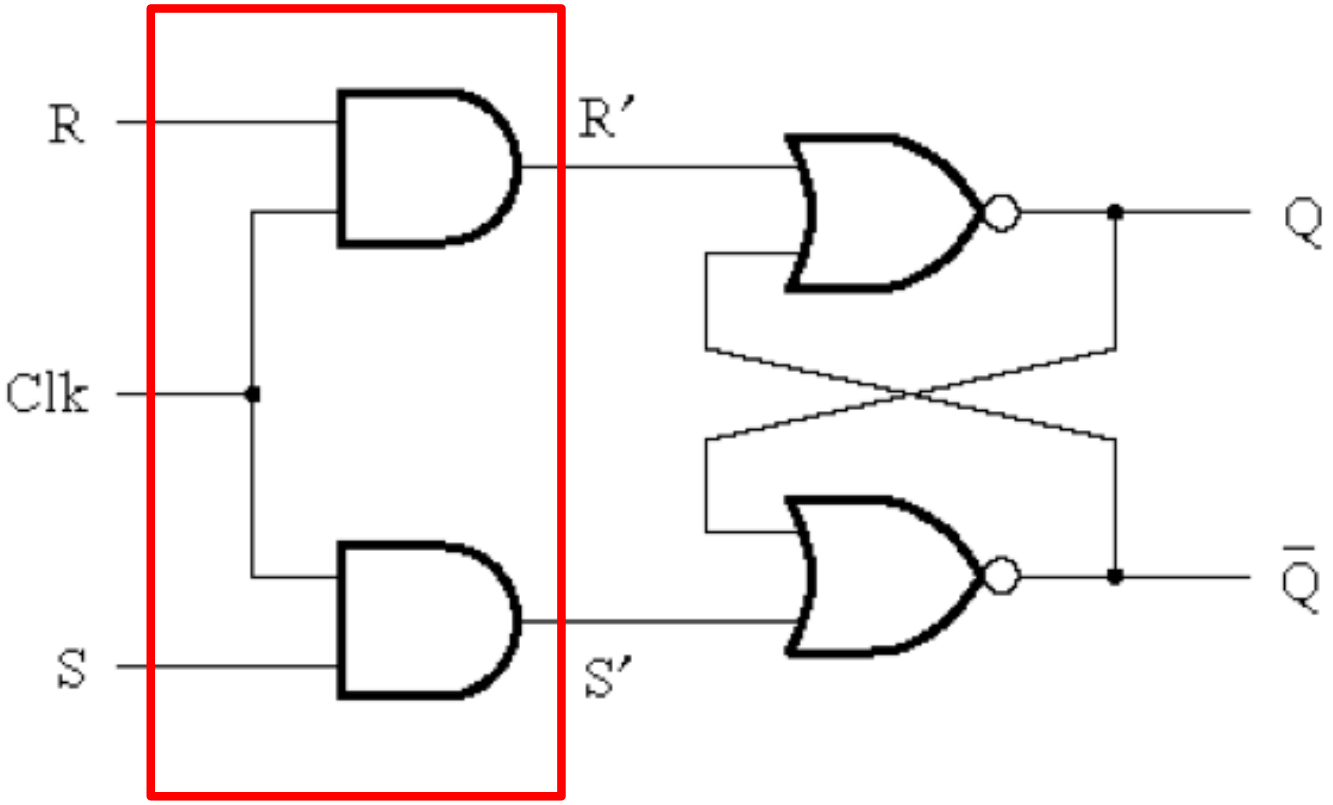
Gated SR Latch

Circuit Diagram for the Gated SR Latch



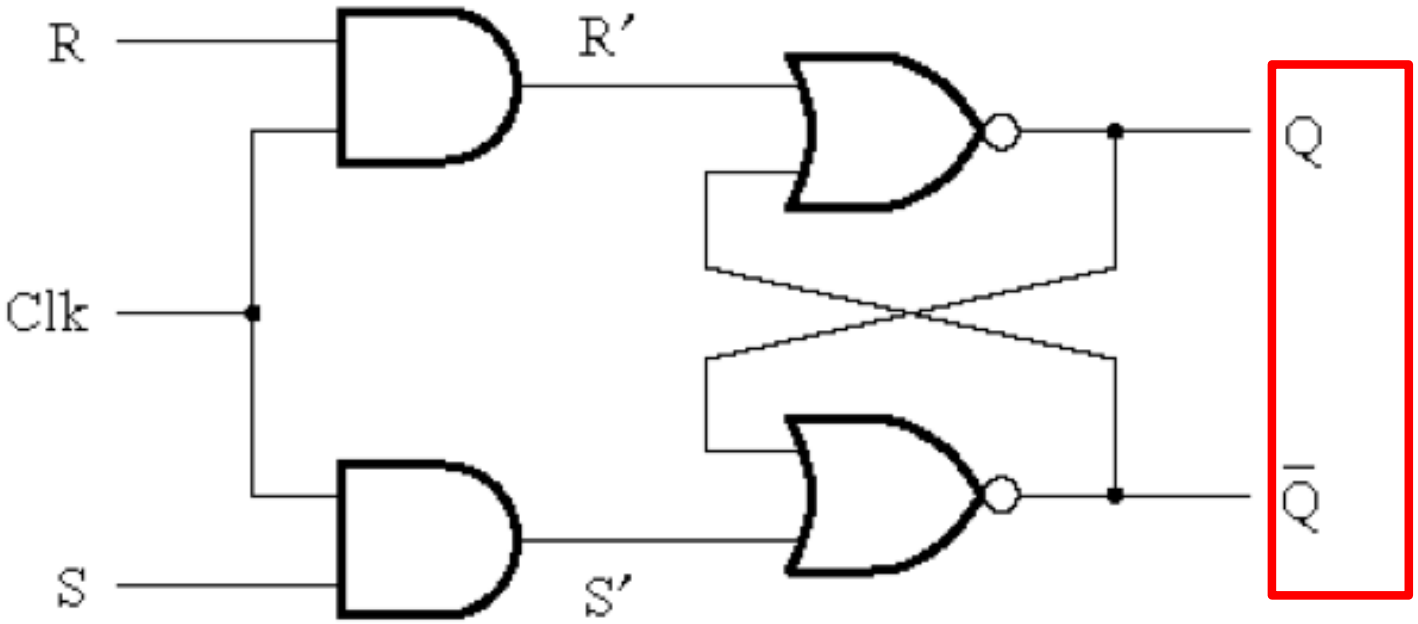
[Figure 5.5a from the textbook]

Circuit Diagram for the Gated SR Latch



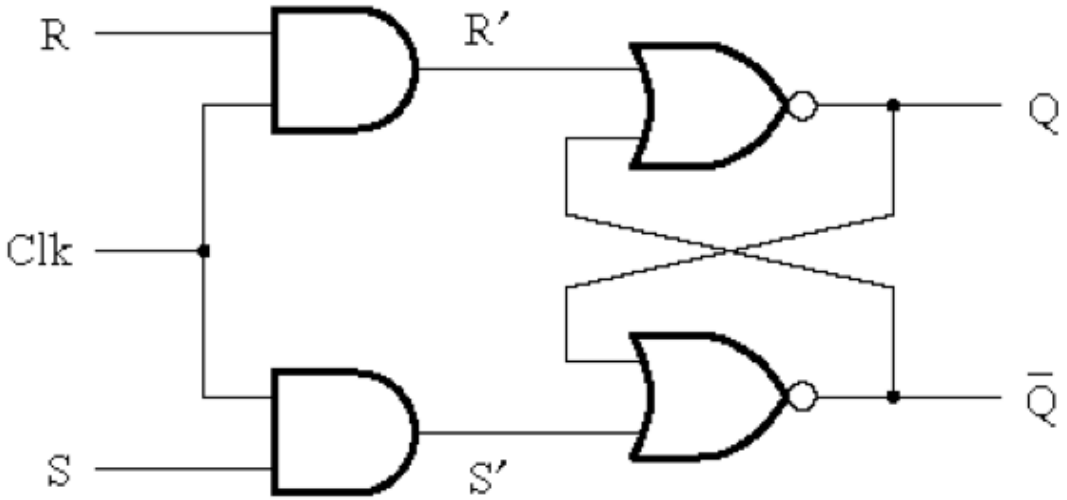
This is the “gate”
of the gated latch

Circuit Diagram for the Gated SR Latch

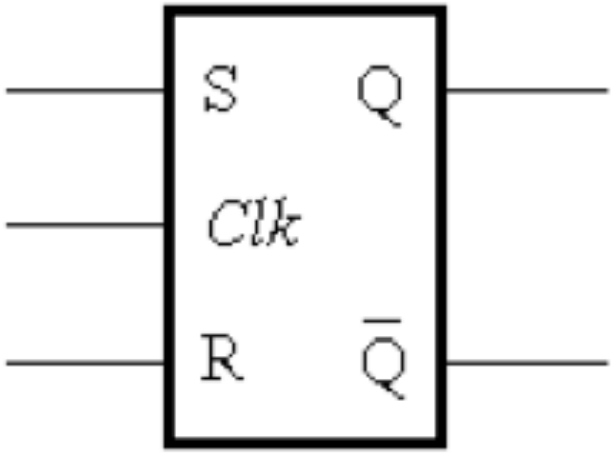


Notice that these are complements of each other

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

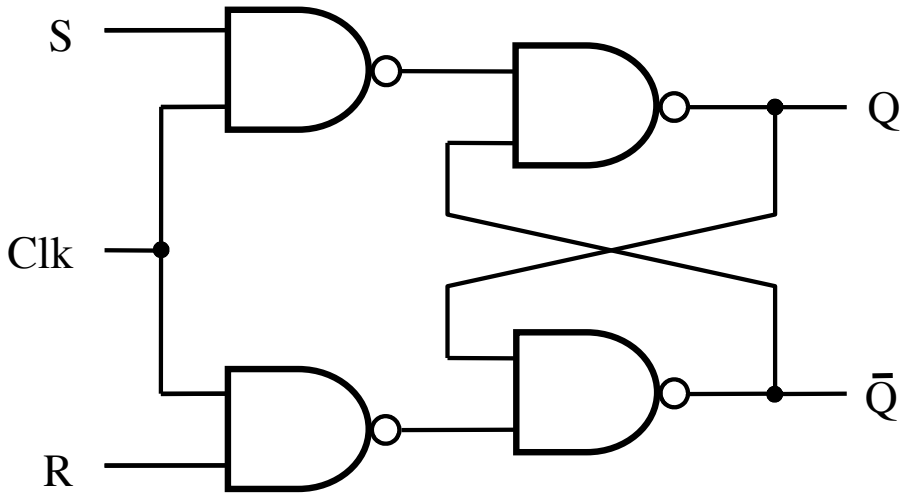


Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x (Undesirable)



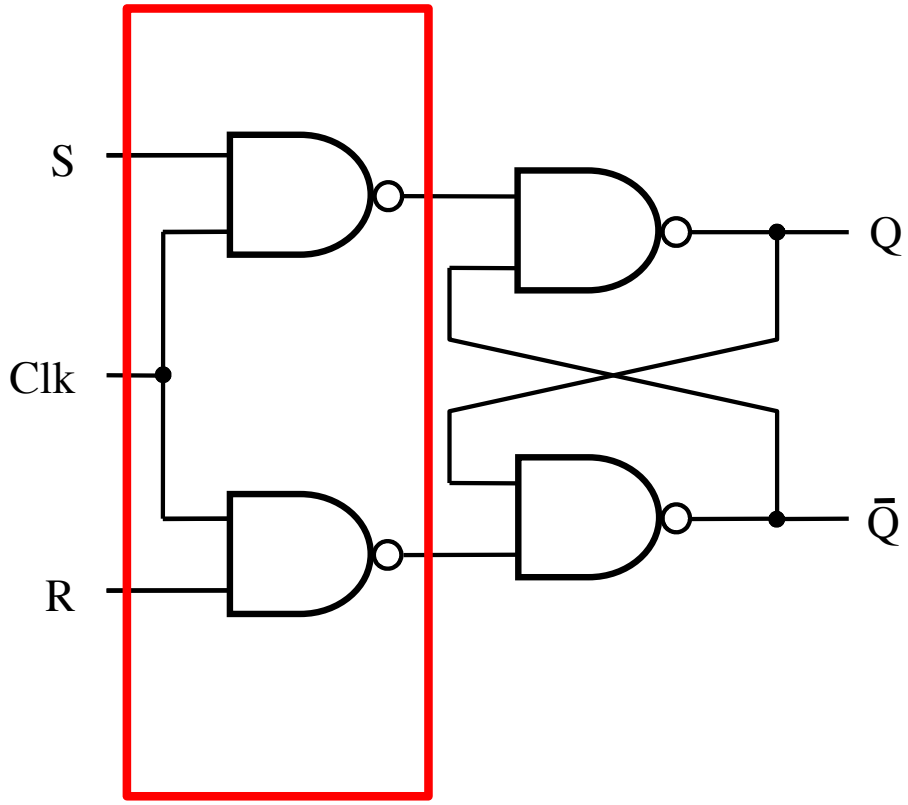
[Figure 5.5 from the textbook]

Gated SR latch with NAND gates



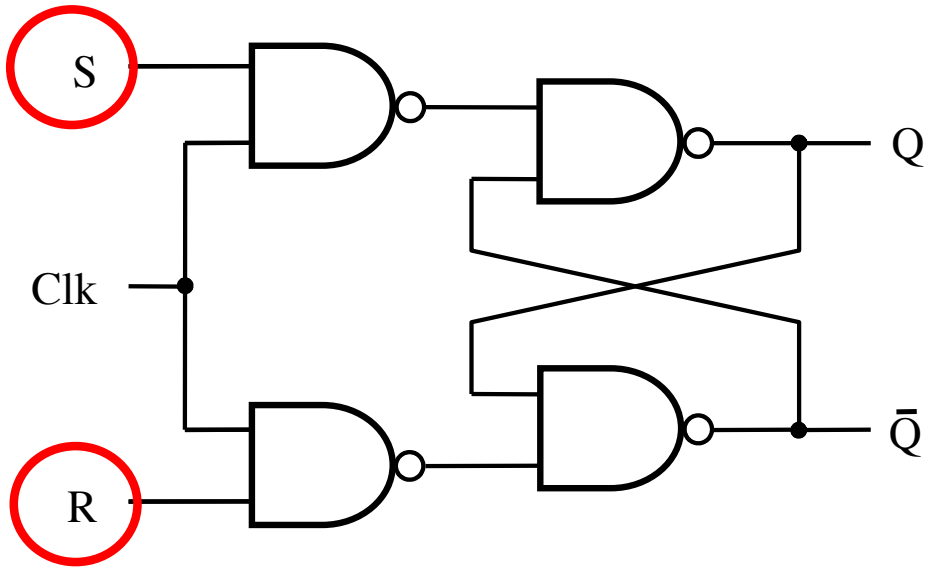
[Figure 5.6 from the textbook]

Gated SR latch with NAND gates



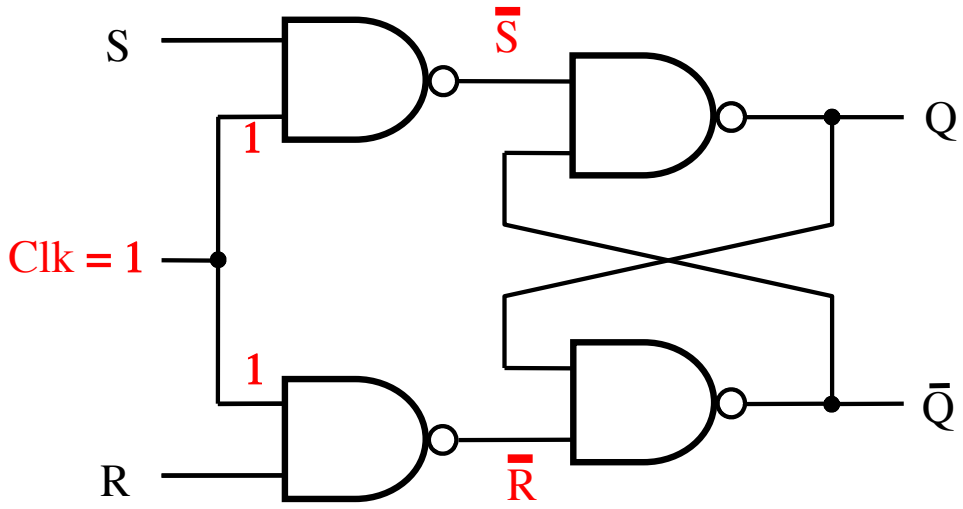
In this case the “gate” is constructed using NAND gates! Not AND gates.

Gated SR latch with NAND gates



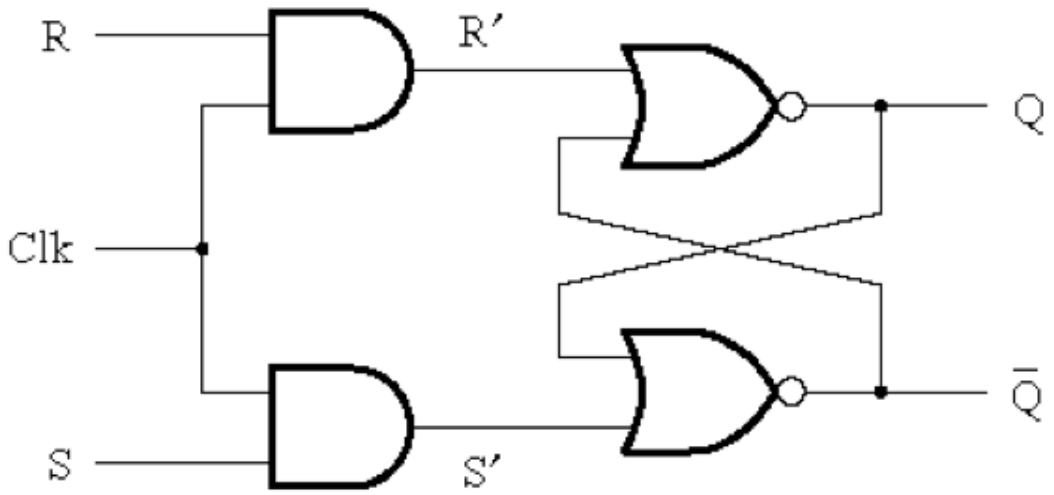
Also, notice that the positions of S and R are now swapped.

Gated SR latch with NAND gates

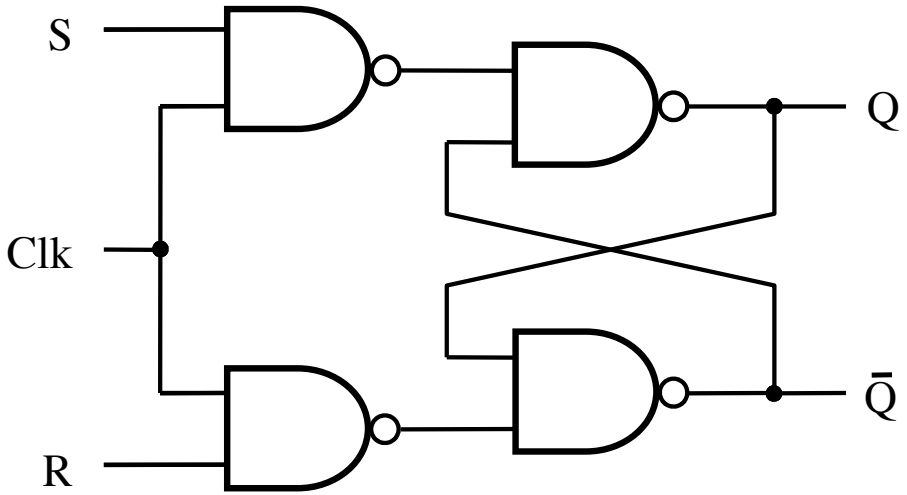


Finally, notice that when Clk=1 this turns into the basic latch with NAND gates, i.e., the $\overline{S}\overline{R}$ Latch.

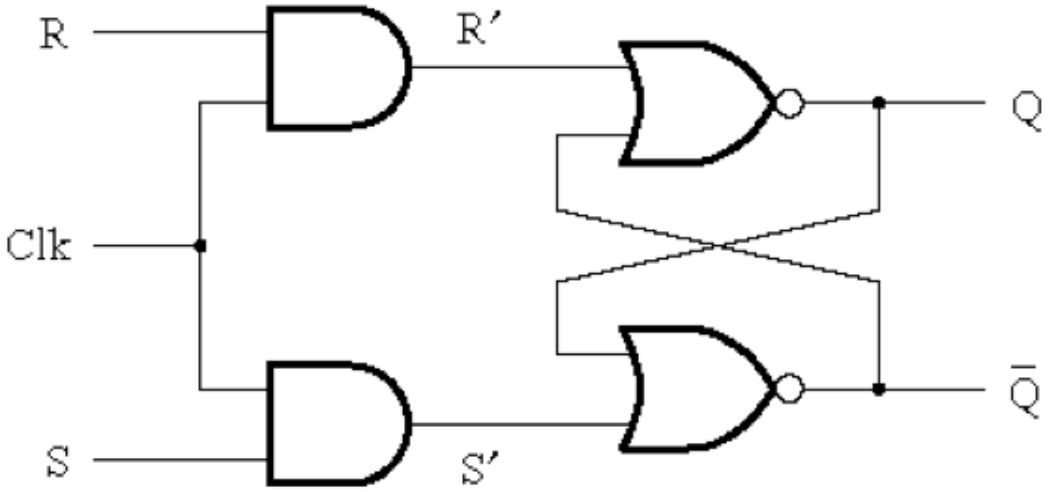
Gated SR latch with NOR gates

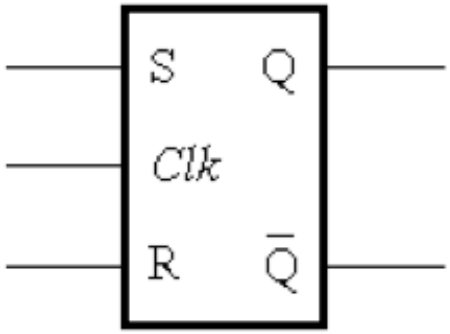


Gated SR latch with NAND gates

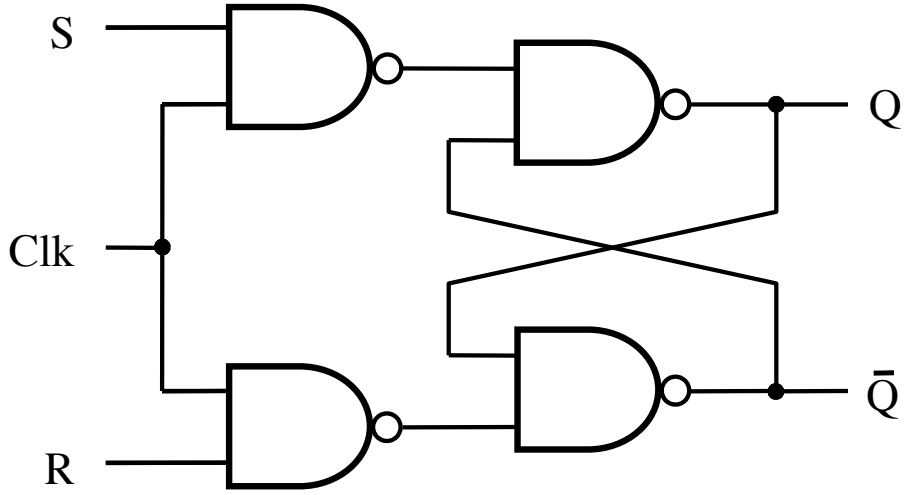


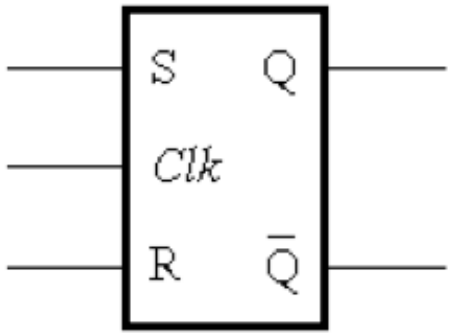
Gated SR latch with NOR gates





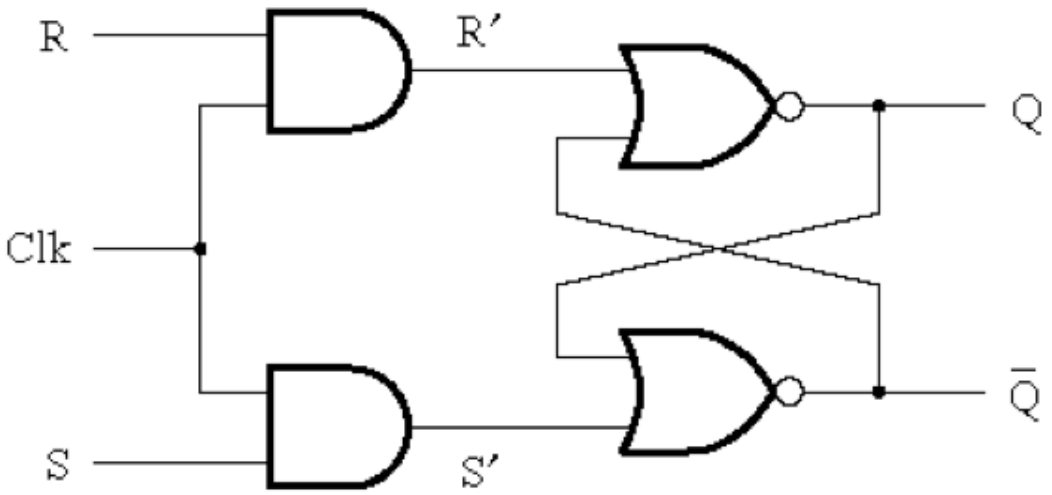
Gated SR latch with NAND gates





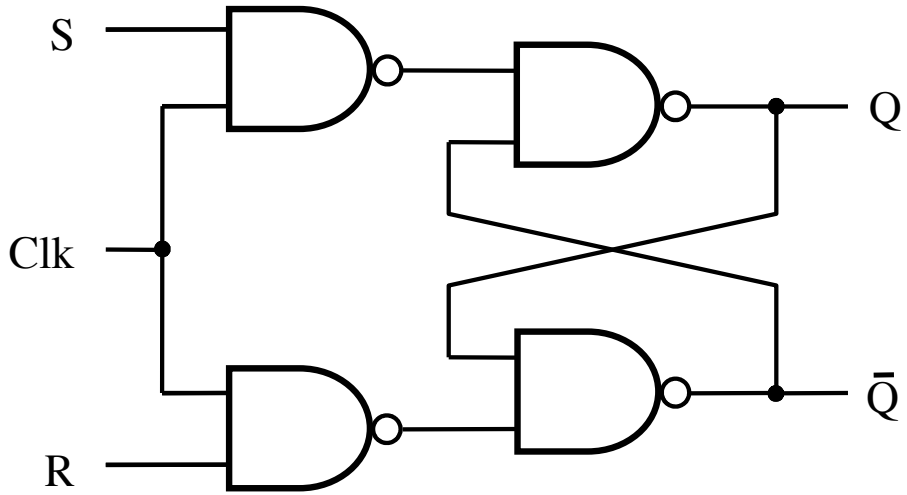
Graphical symbols are the same

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

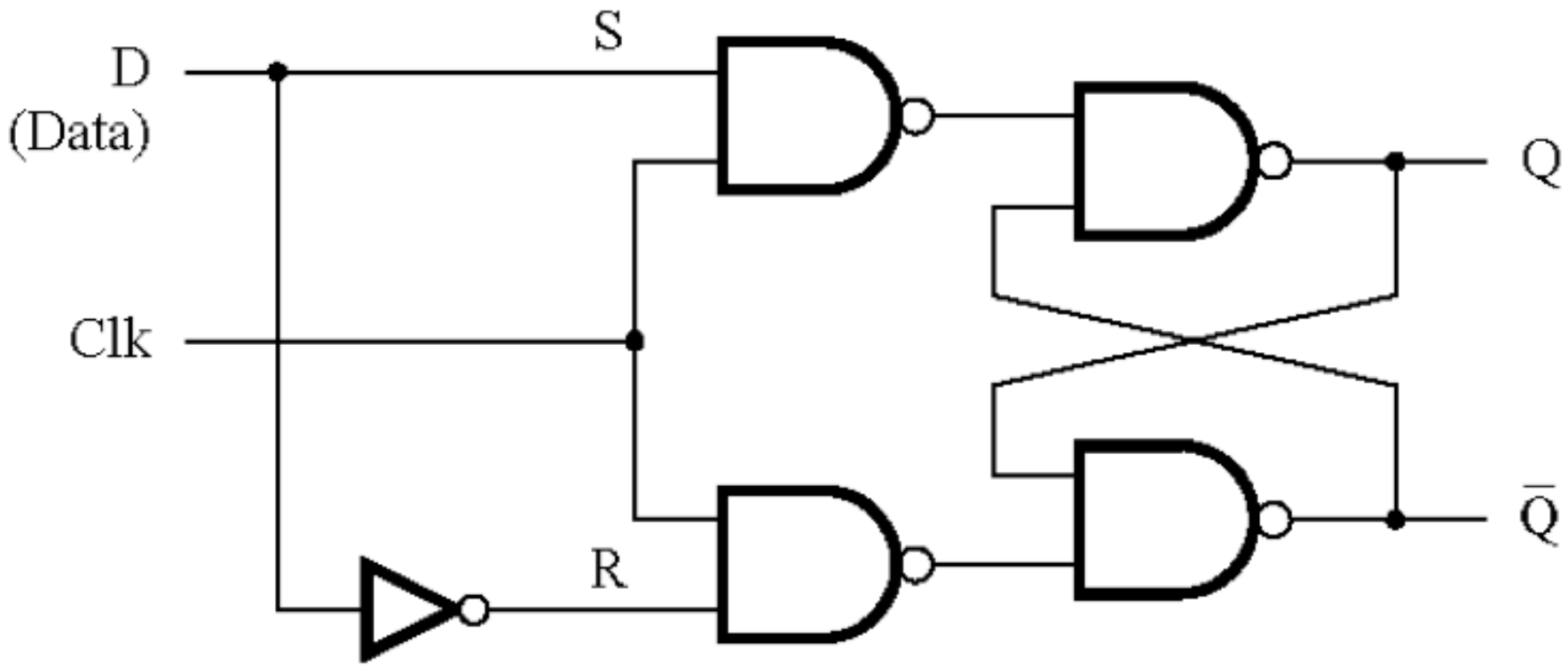


Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x (undesirable)

Characteristic tables are the same

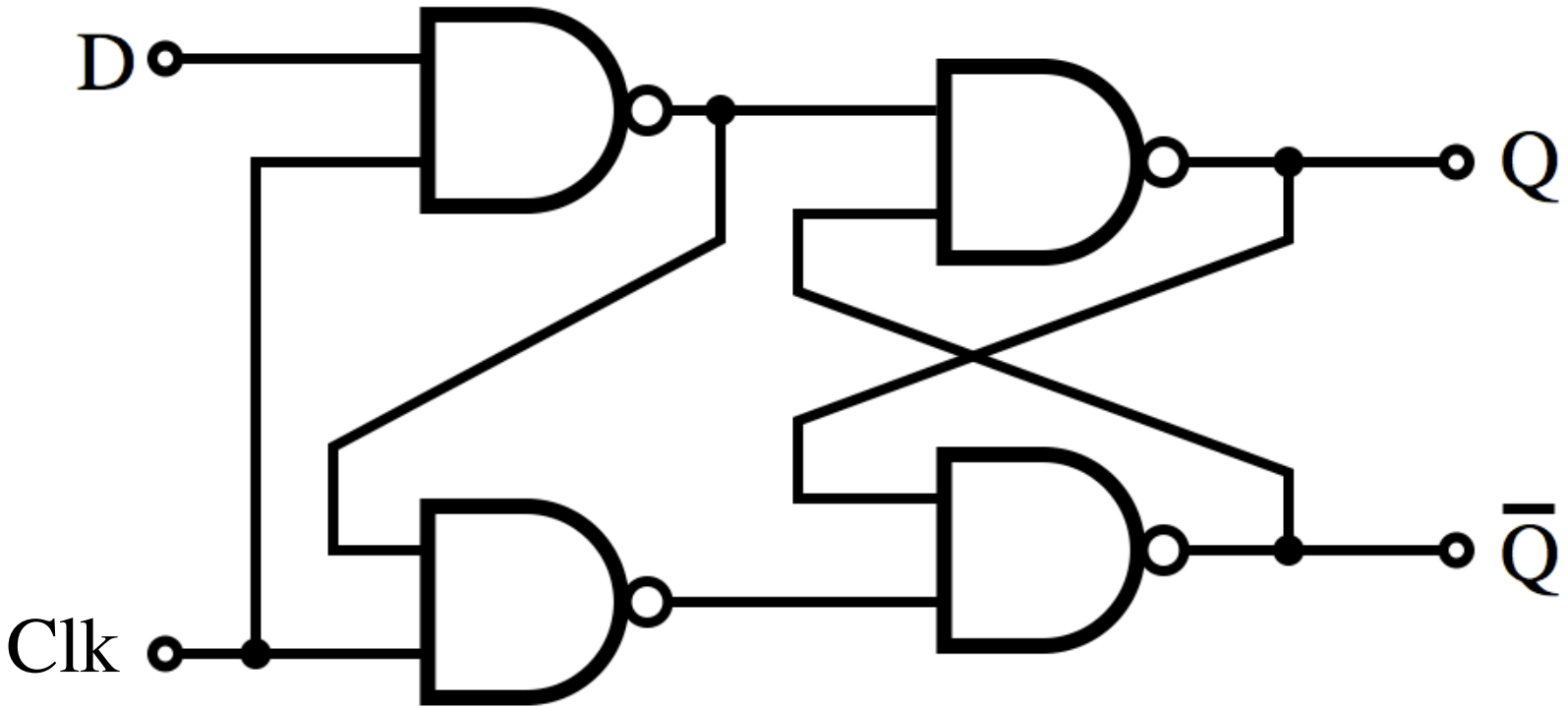
Gated D Latch

Circuit Diagram for the Gated D Latch



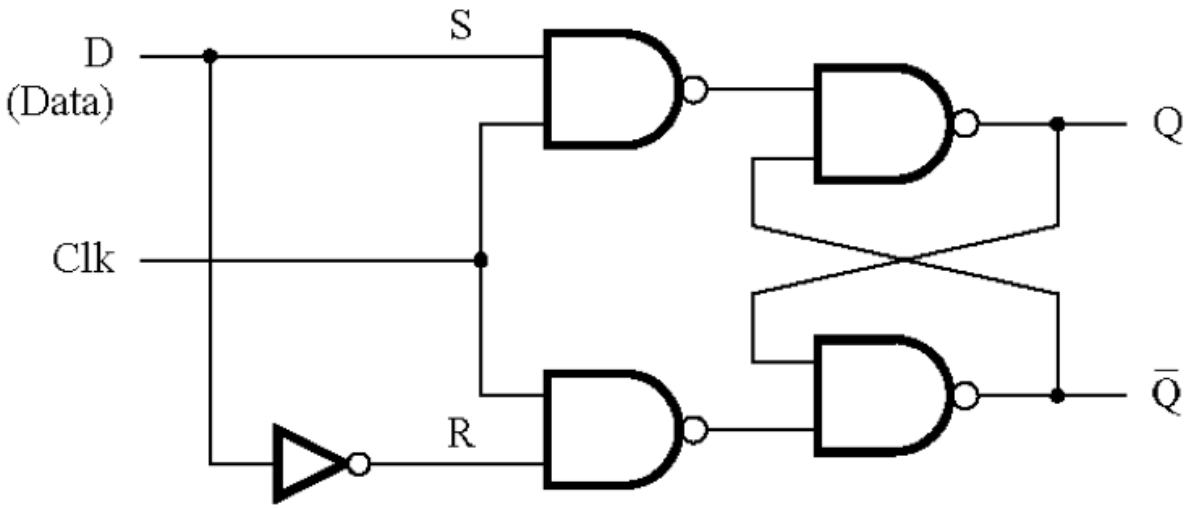
[Figure 5.7a from the textbook]

Gated D Latch: Alternative Design



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

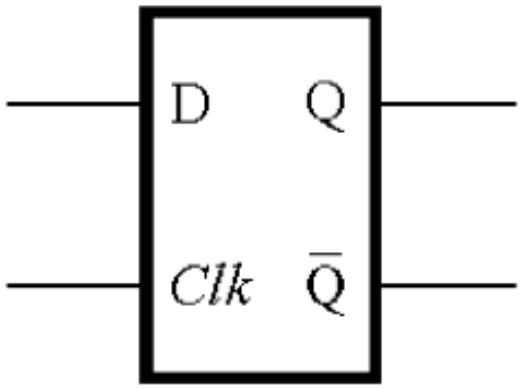
Gated D Latch: Circuit Diagram, Characteristic Table, and Graphical Symbol



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

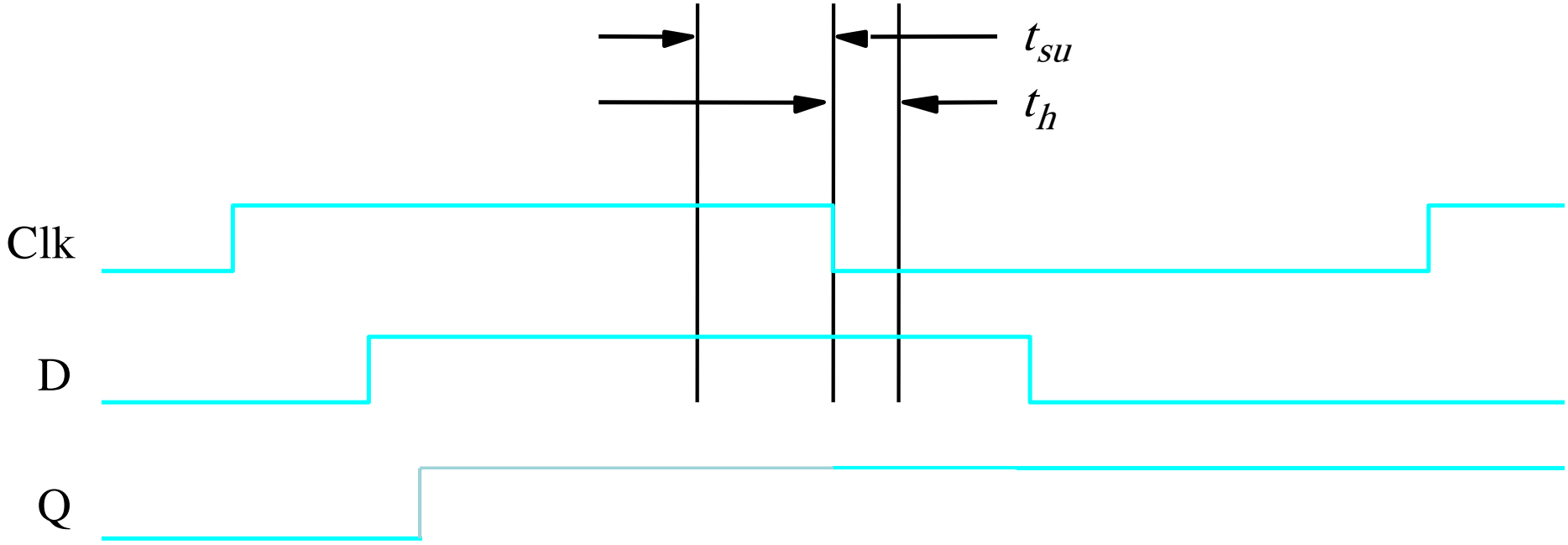
Note that it is now impossible to have S=R=1.

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

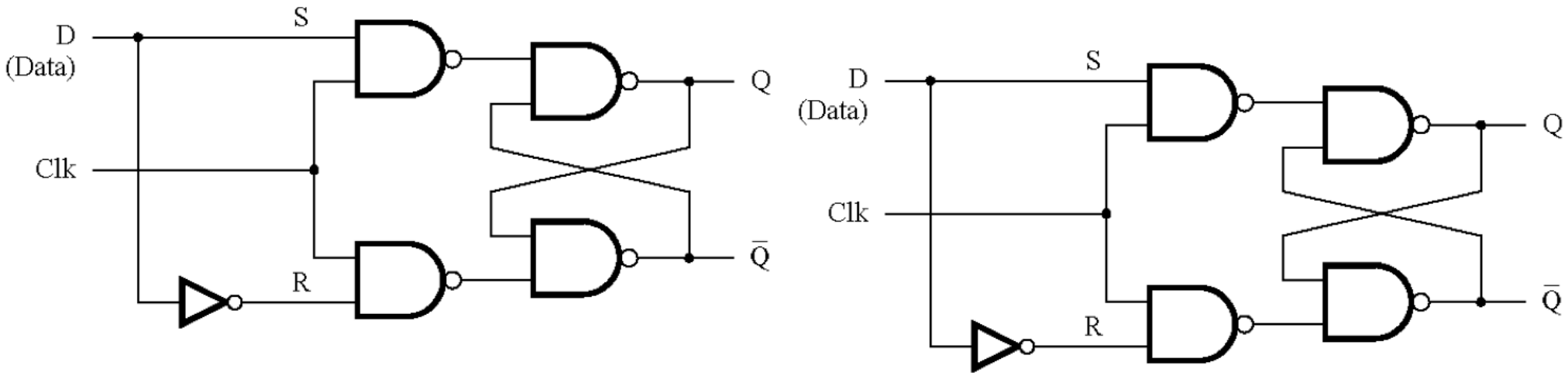
[Figure 5.8 from the textbook]

Master-Slave D Flip-Flop

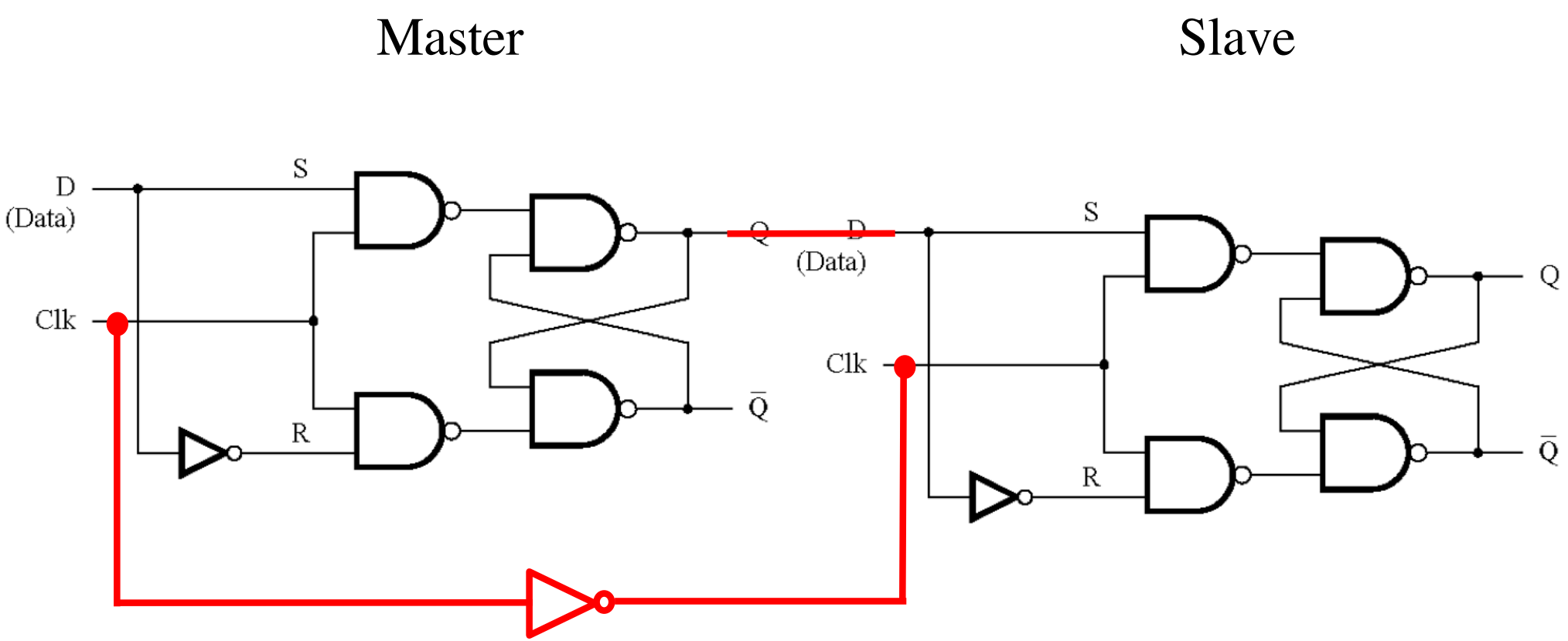
Constructing a Master-Slave D Flip-Flop From Two D Latches

Master

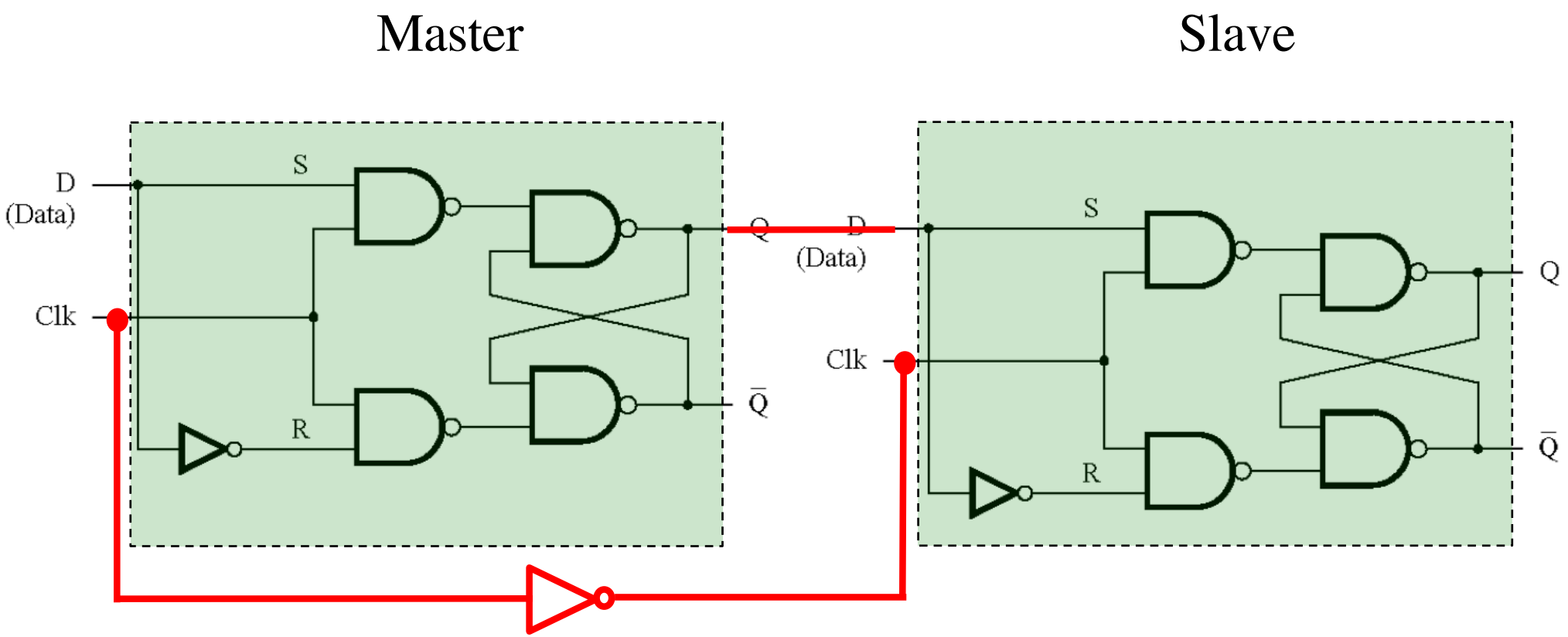
Slave



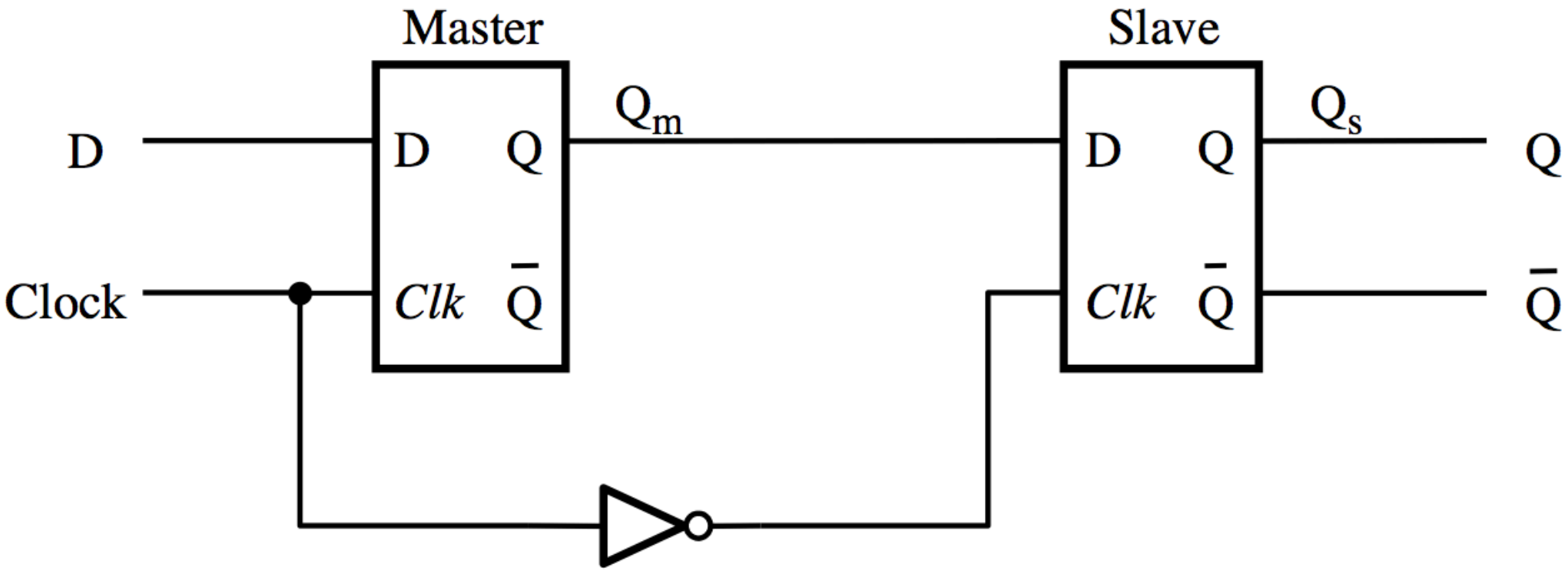
Constructing a Master-Slave D Flip-Flop From Two D Latches



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



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



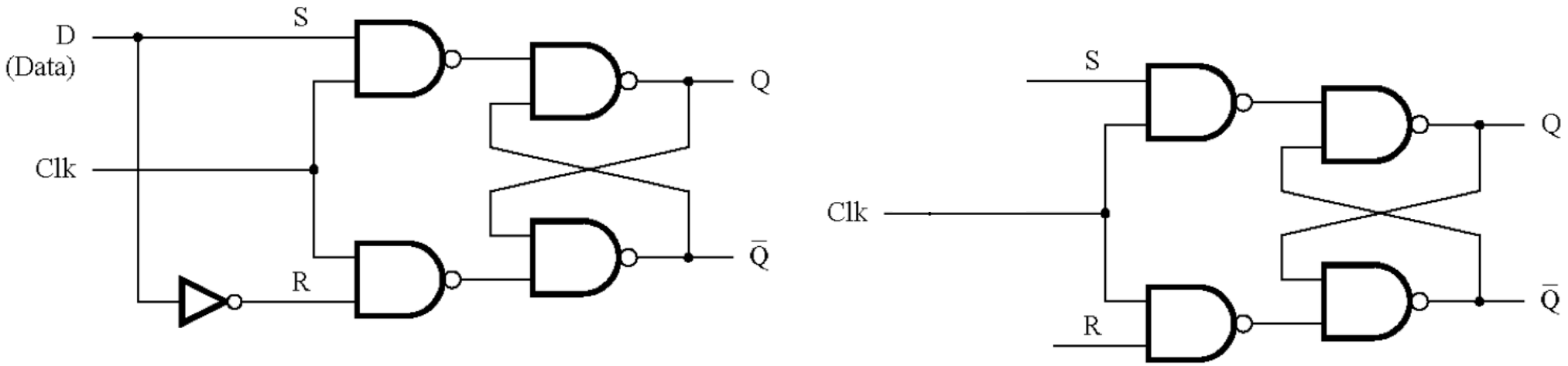
[Figure 5.9a from the textbook]

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

(This version uses one less NOT gate)

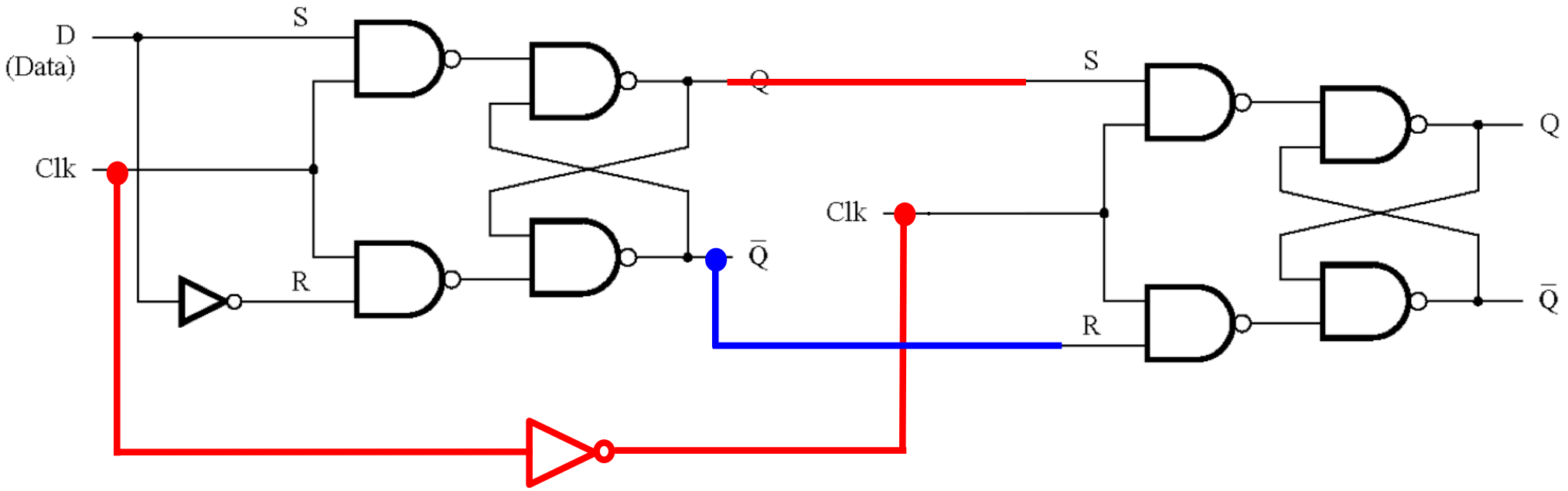
Master

Slave



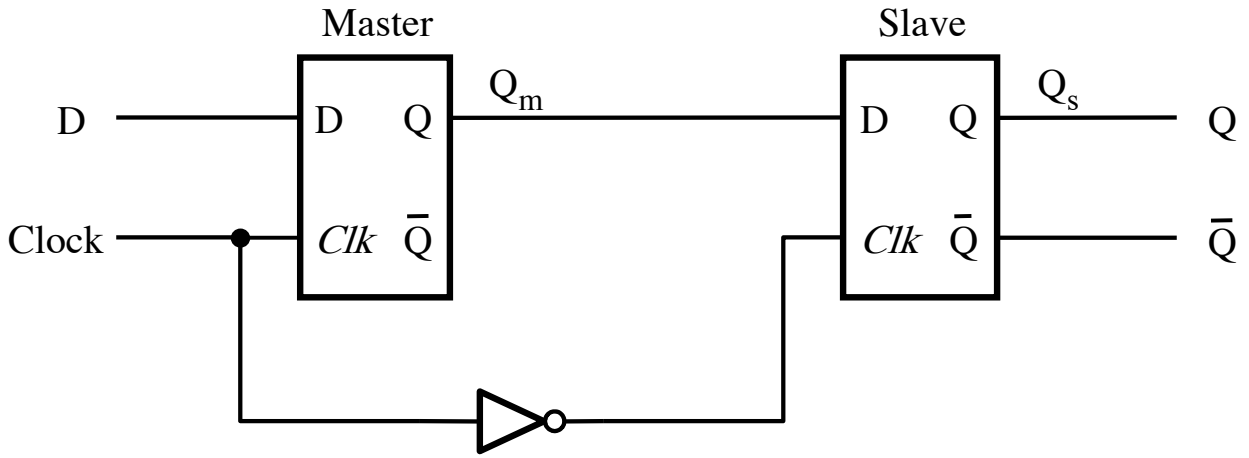
Constructing a Master-Slave D Flip-Flop From one D Latch and one Gated SR Latch

(This version uses one less NOT gate)



Edge-Triggered D Flip-Flops

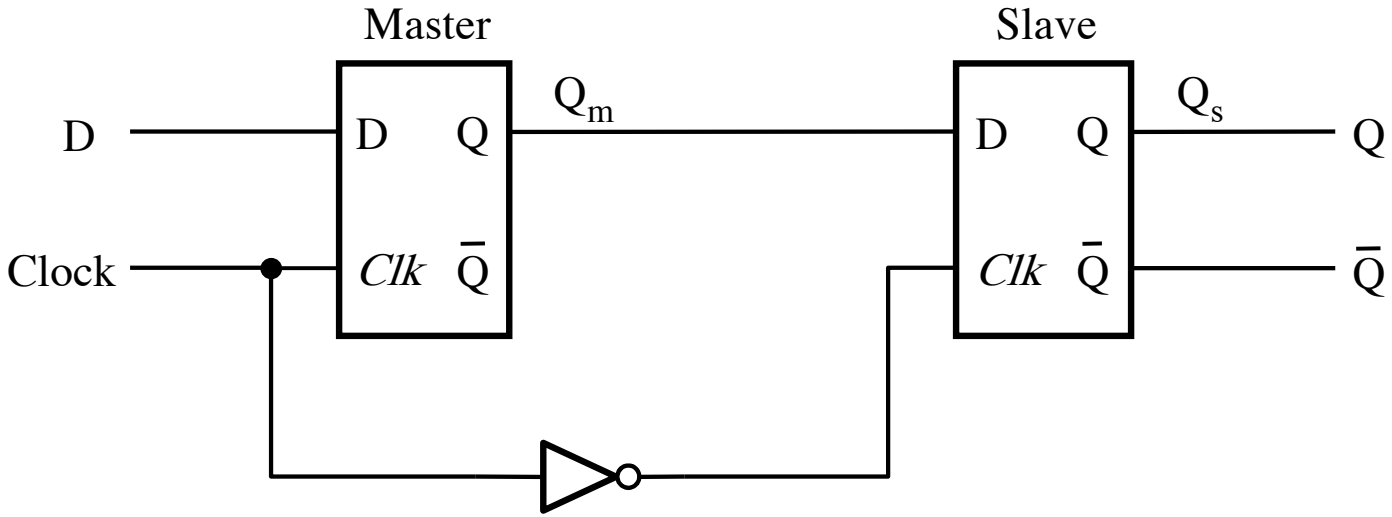
Master-Slave D Flip-Flop



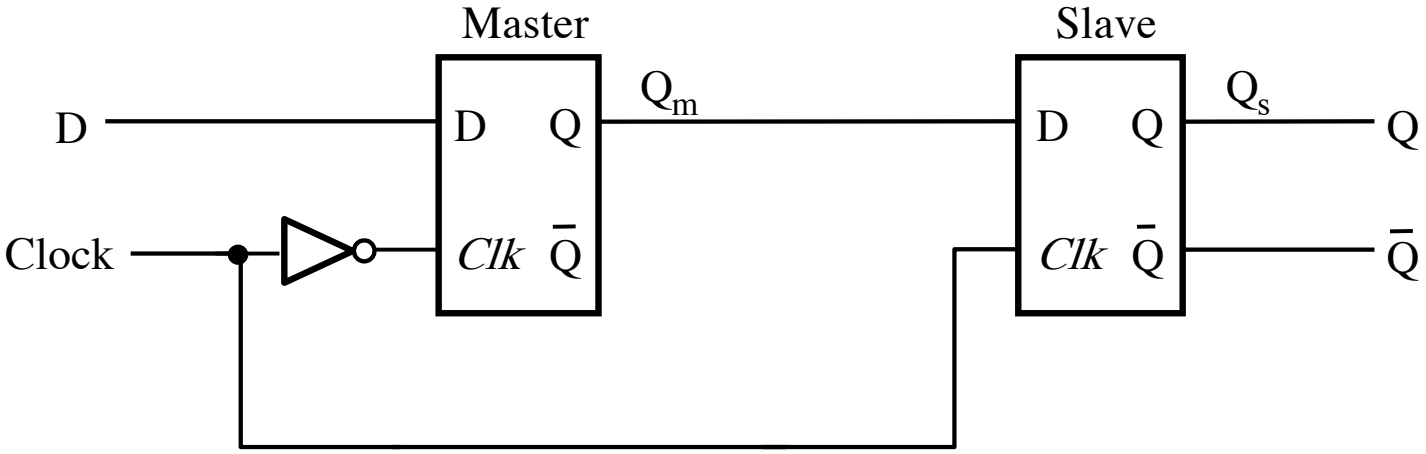
(a) Circuit

[Figure 5.9a from the textbook]

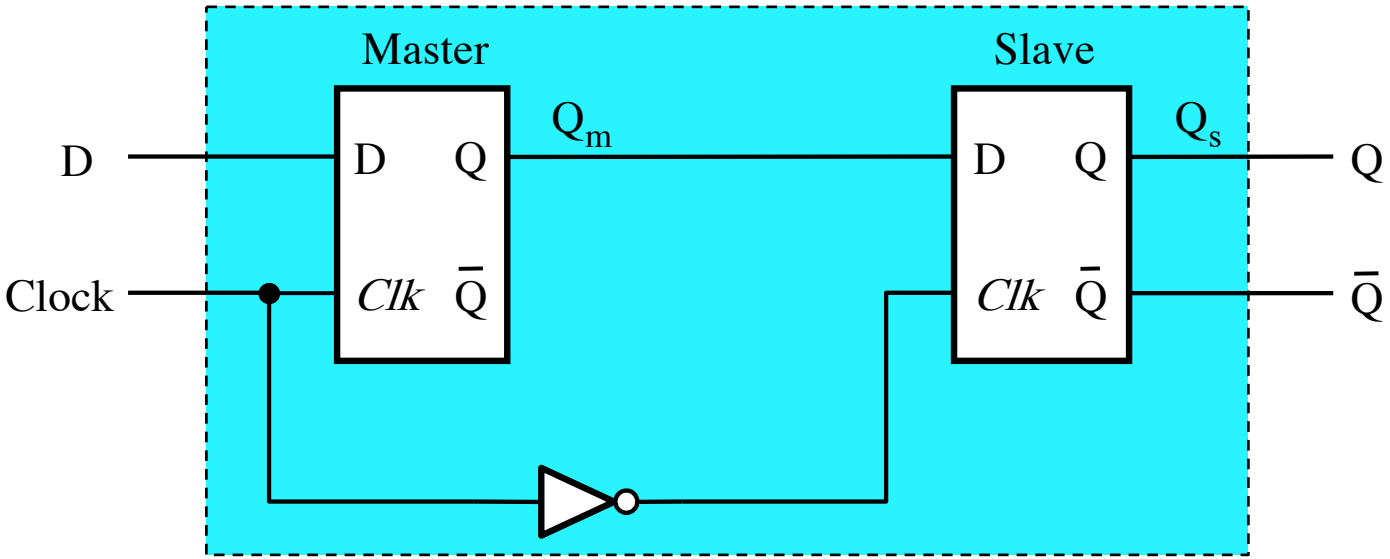
Negative-Edge-Triggered Master-Slave D Flip-Flop



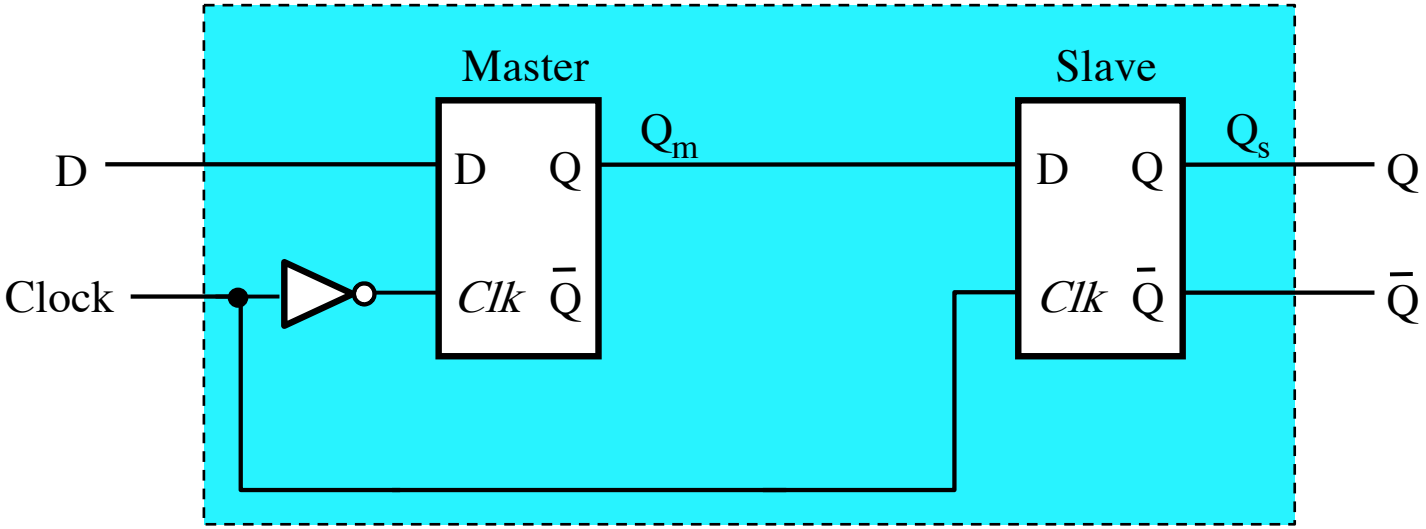
Positive-Edge-Triggered Master-Slave D Flip-Flop



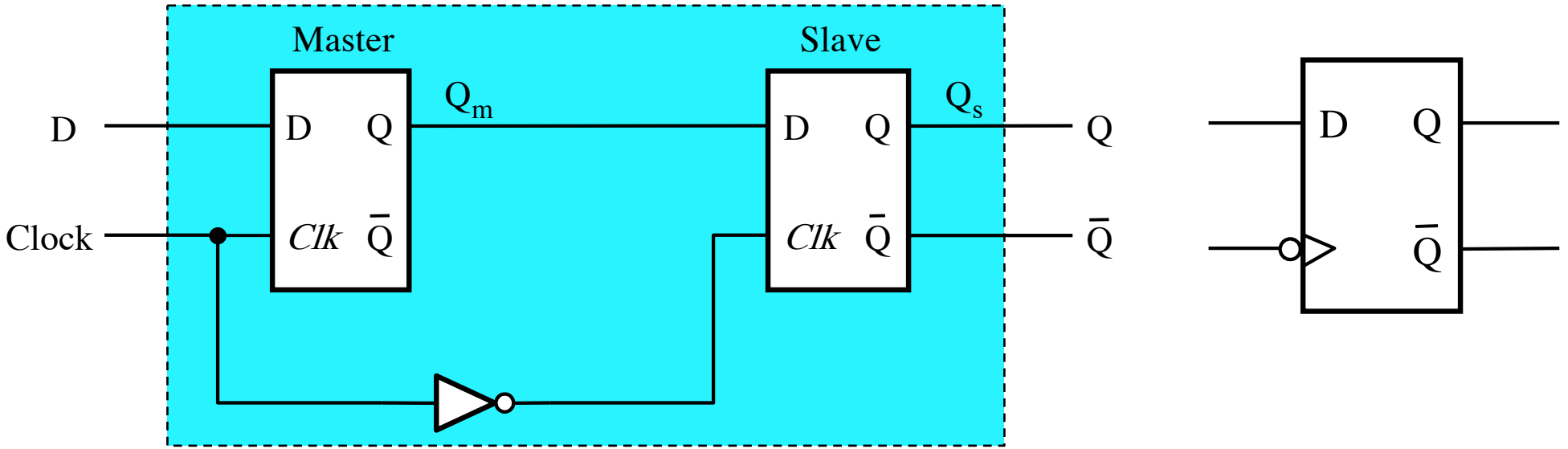
Negative-Edge-Triggered Master-Slave D Flip-Flop



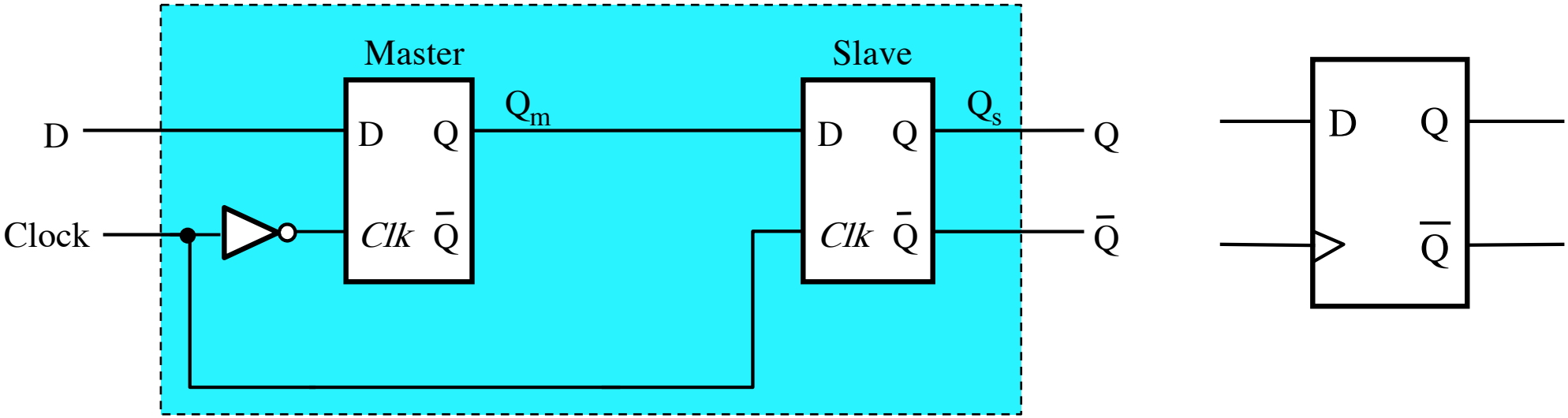
Positive-Edge-Triggered Master-Slave D Flip-Flop



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

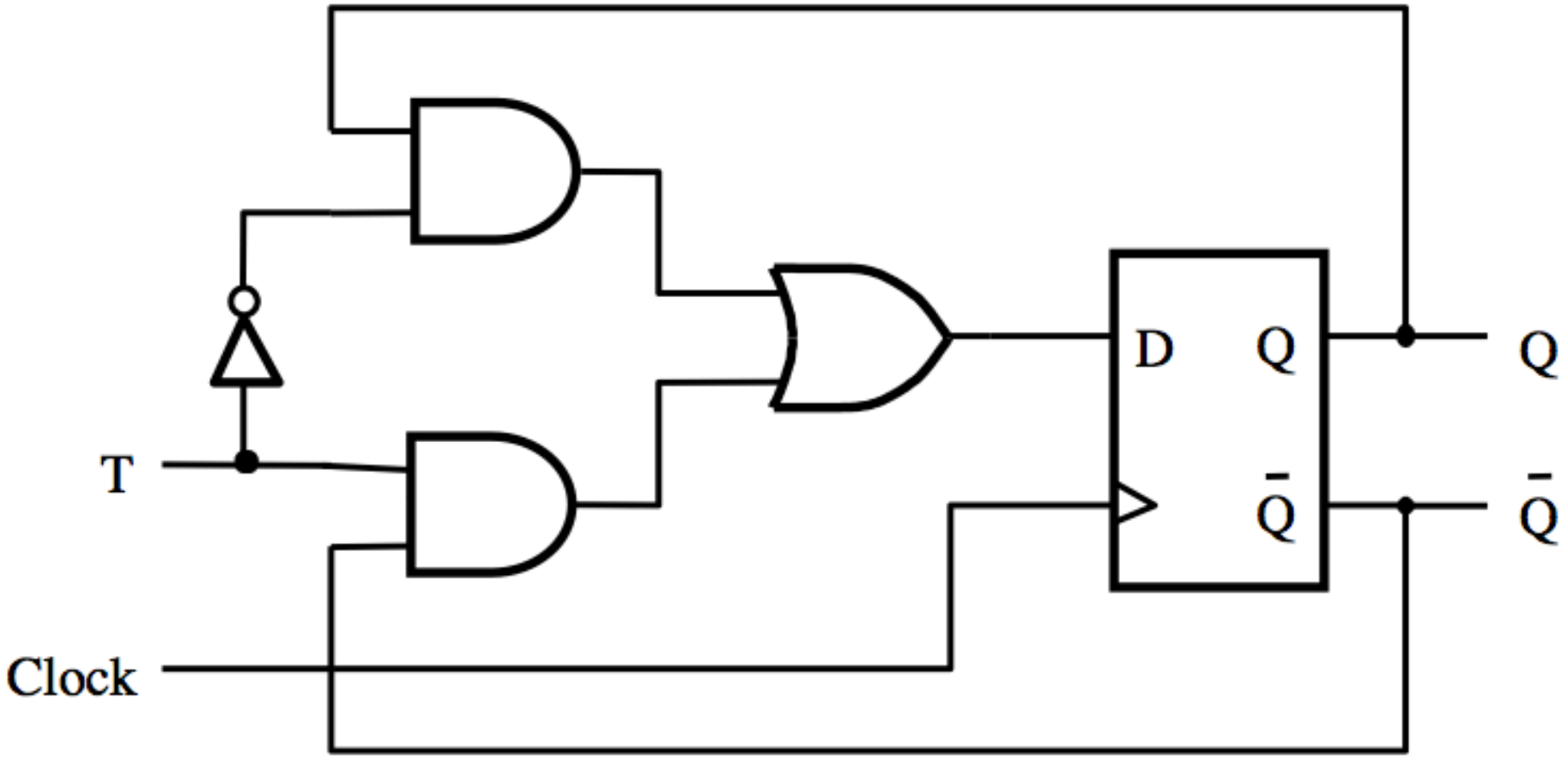


Positive-Edge-Triggered Master-Slave D Flip-Flop



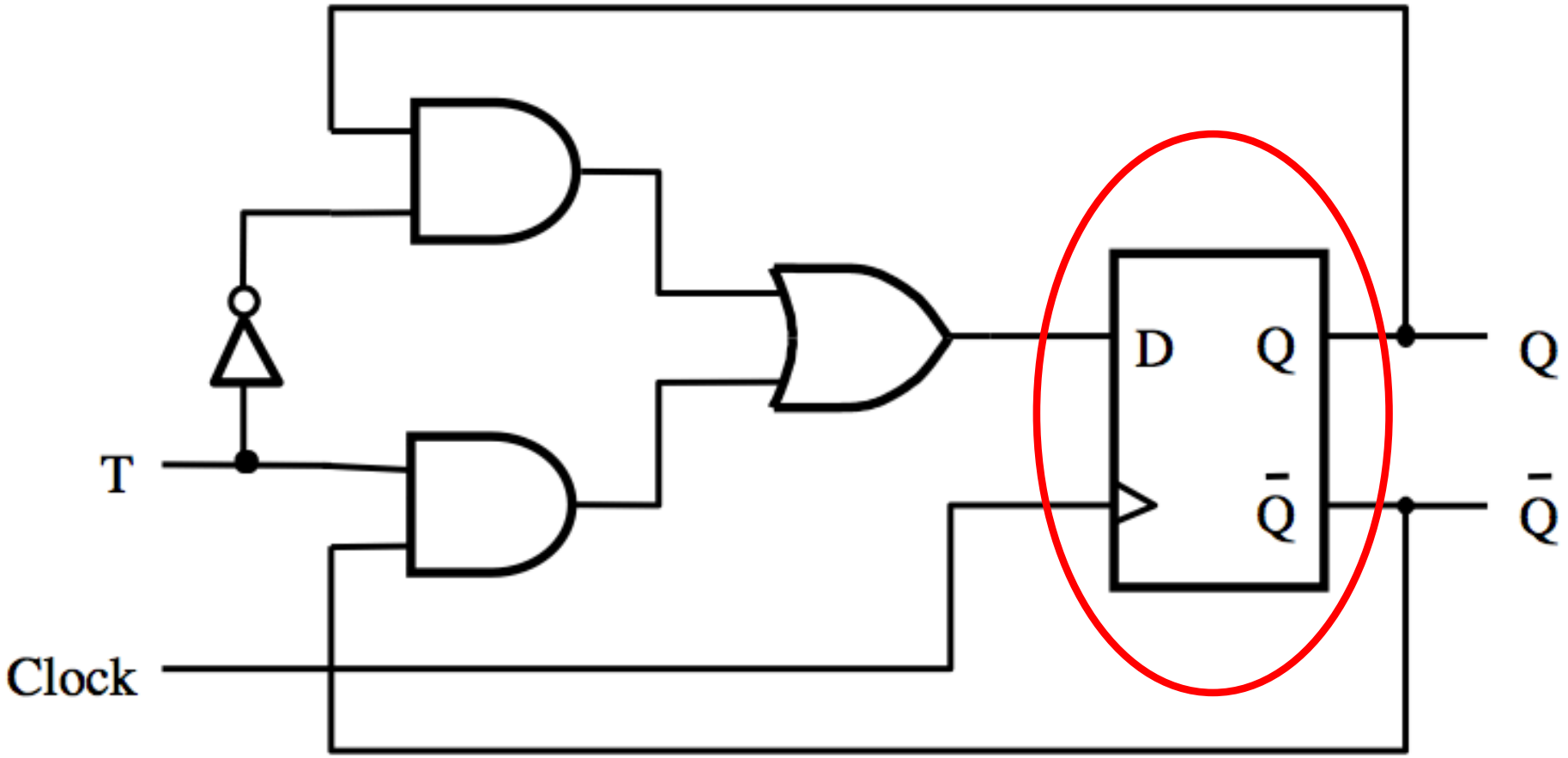
T Flip-Flop

T Flip-Flop



[Figure 5.15a from the textbook]

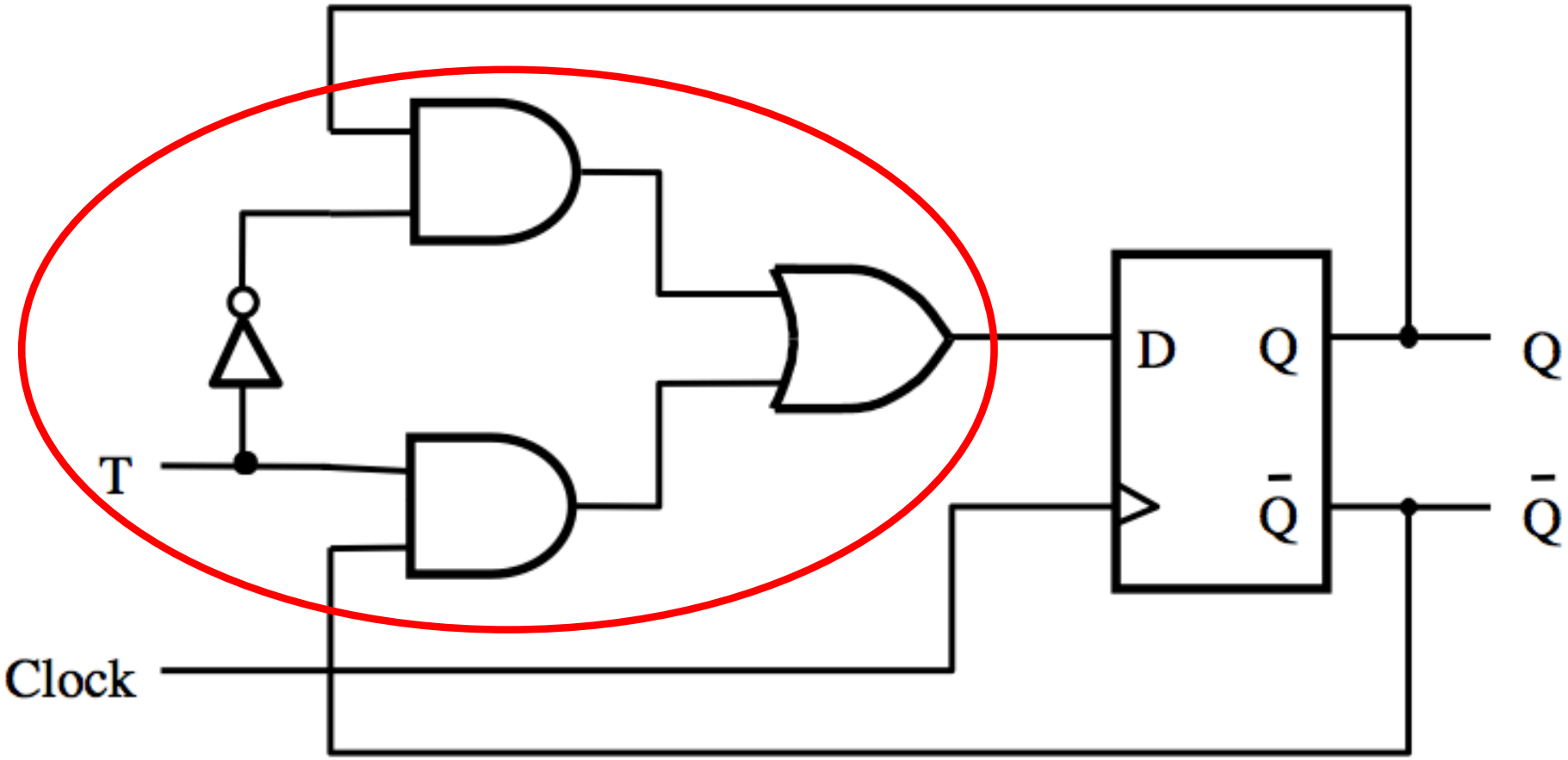
T Flip-Flop



Positive-edge-triggered
D Flip-Flop

[Figure 5.15a from the textbook]

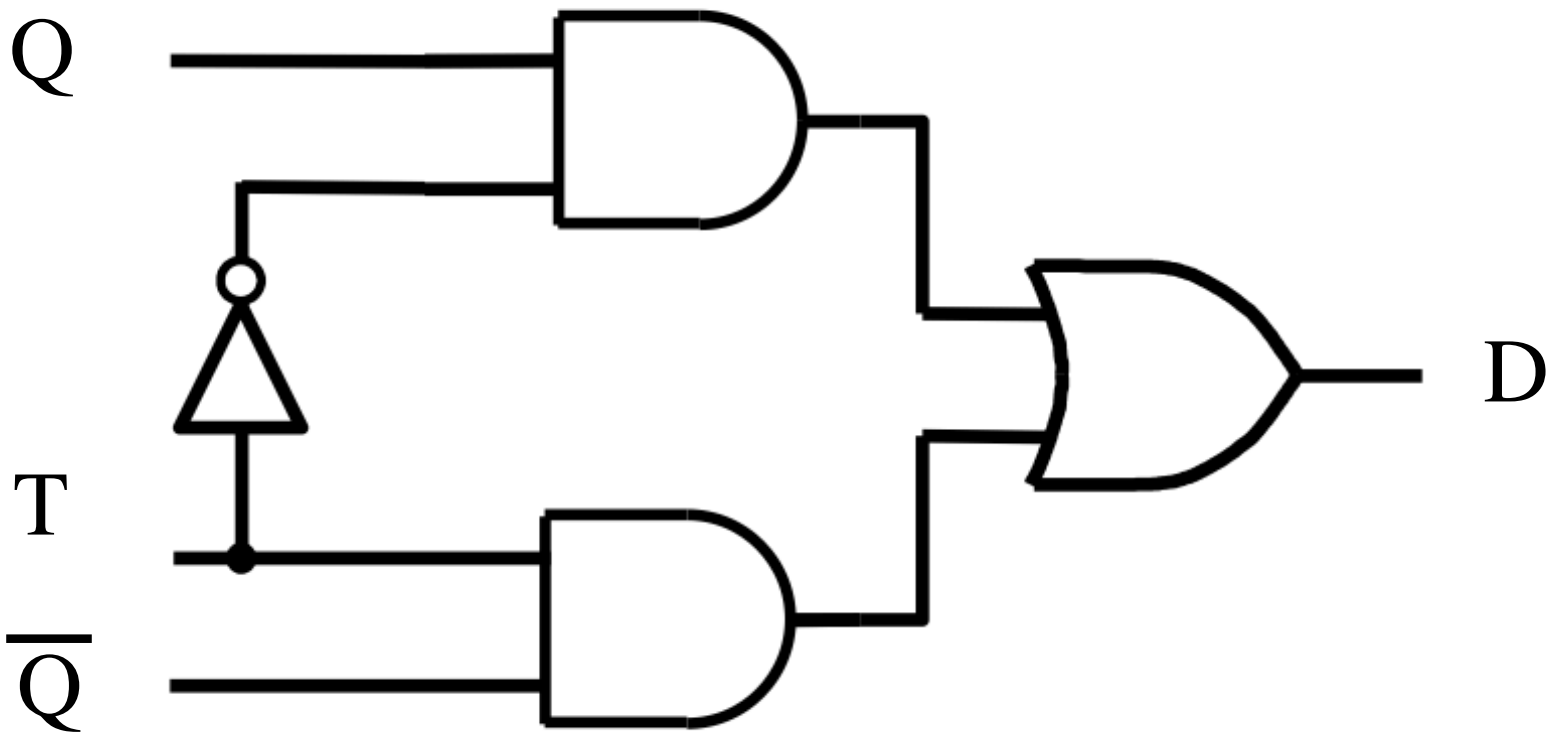
T Flip-Flop



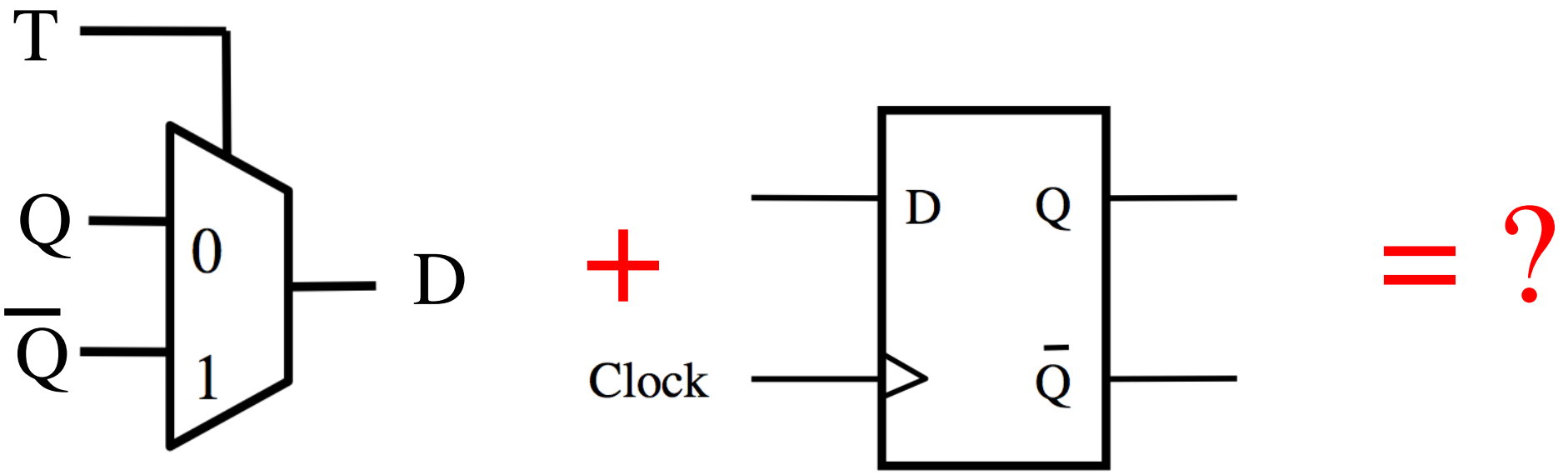
What is this?

[Figure 5.15a from the textbook]

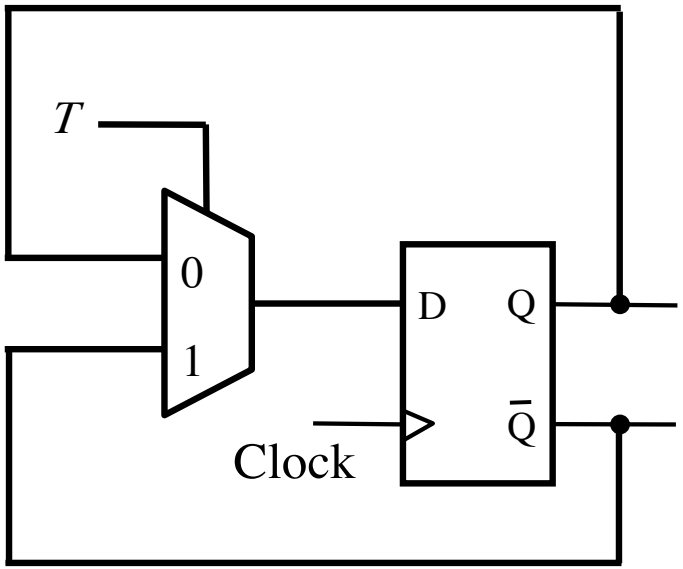
What is this?



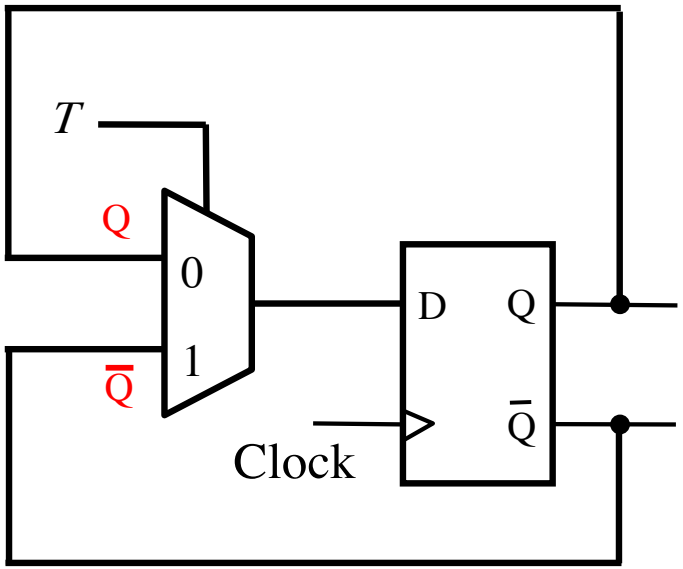
What is this?



T Flip-Flop

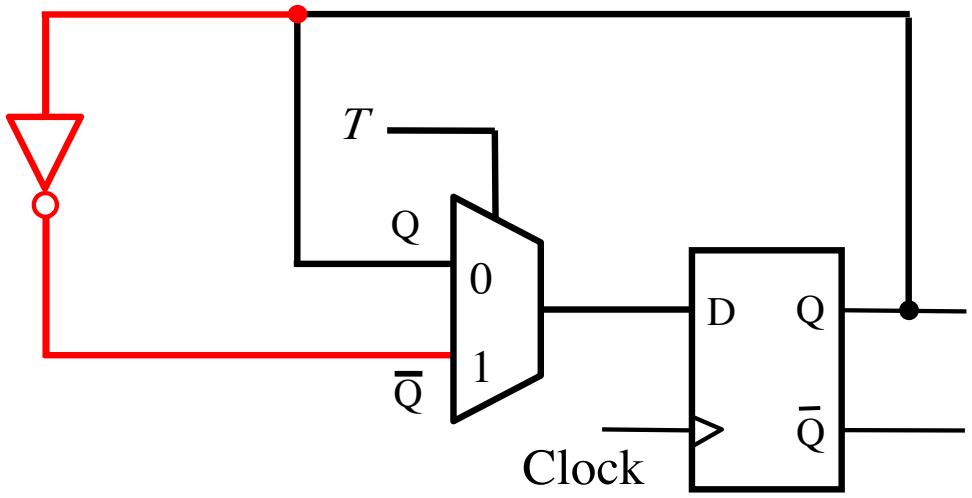


T Flip-Flop

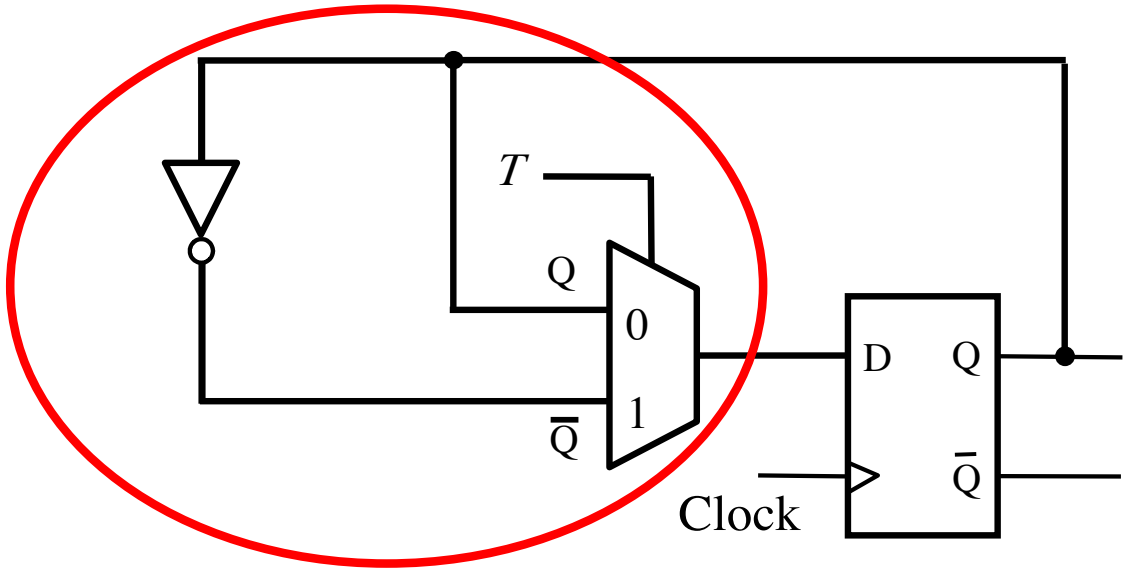


Note that the two inputs to the multiplexer are inverses of each other.

Another Way to Draw This

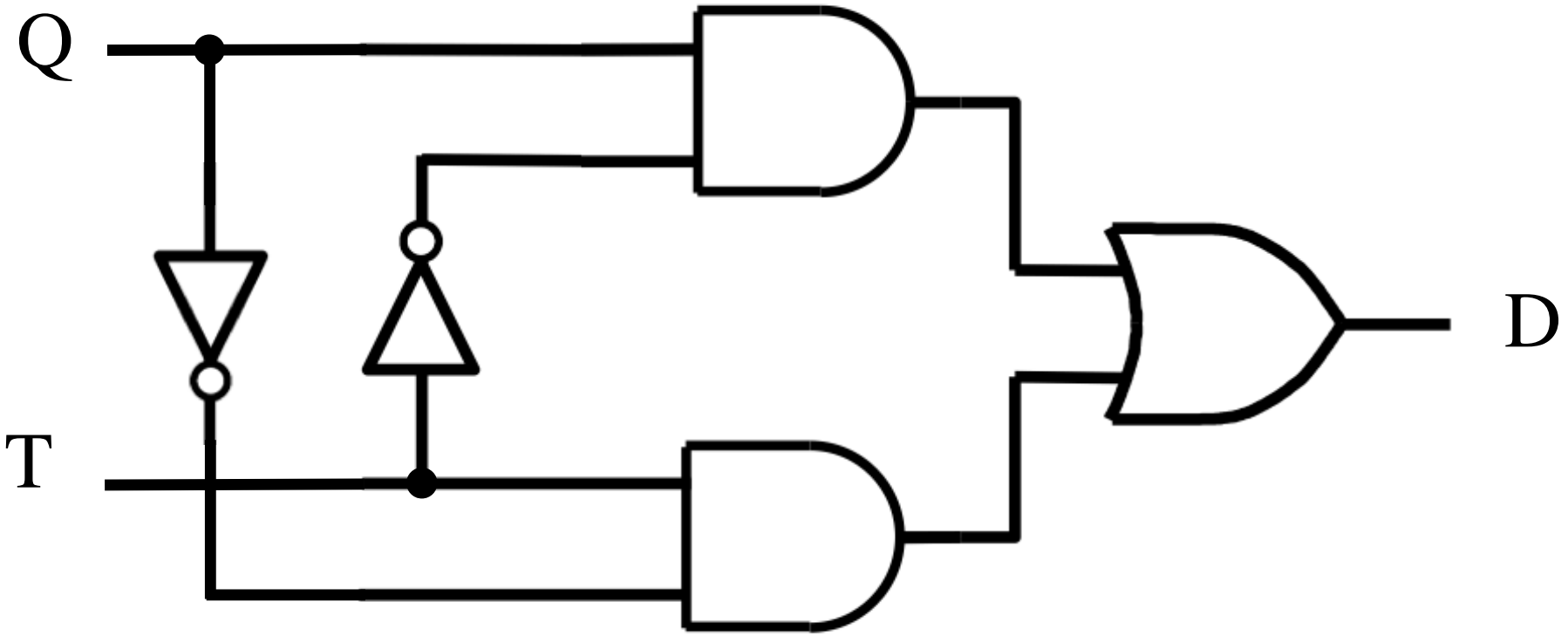


Another Way to Draw This

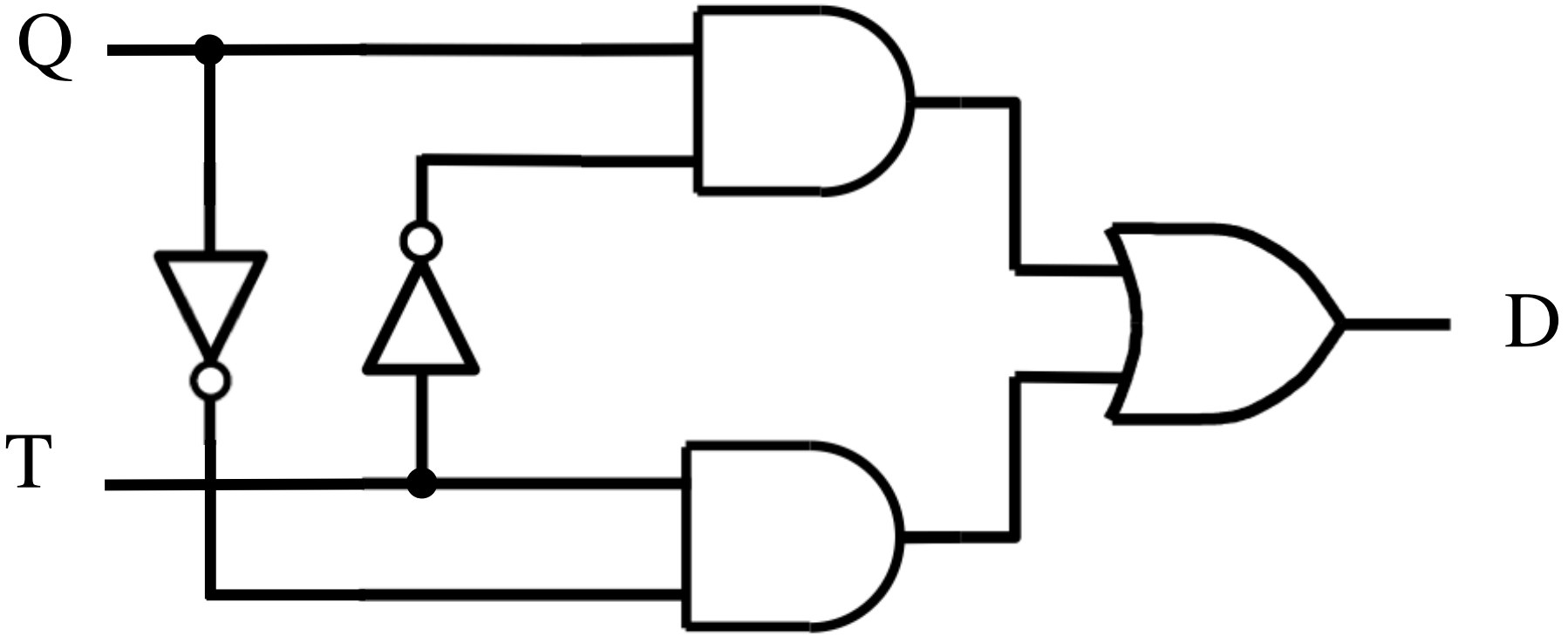


What is this?

What is this?

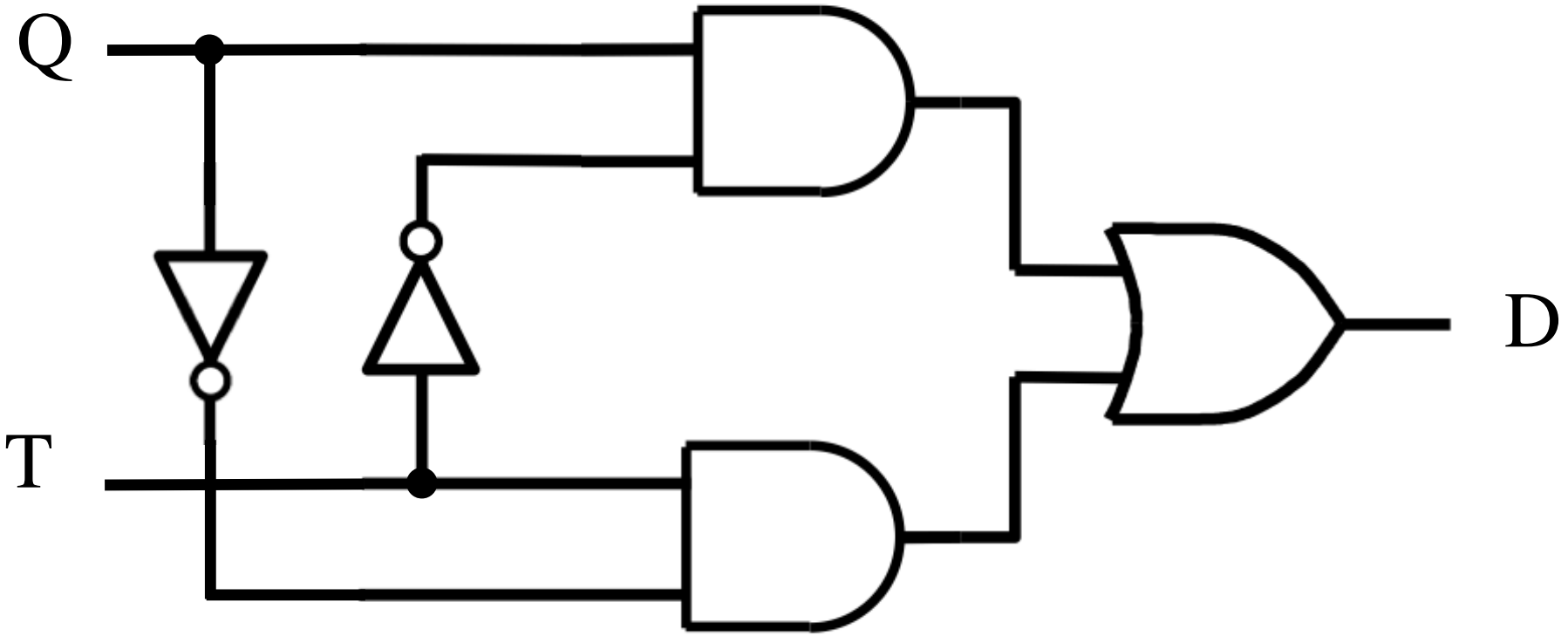


What is this?



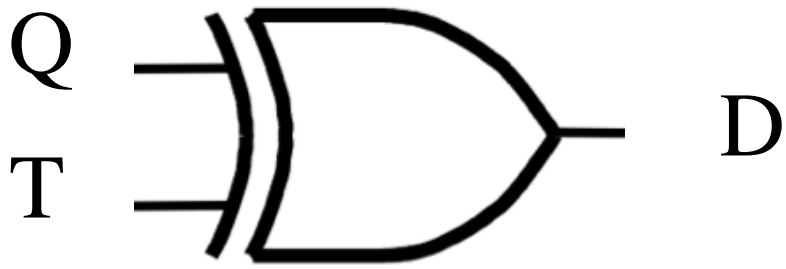
$$D = Q\overline{T} + \overline{Q}T$$

It is an XOR



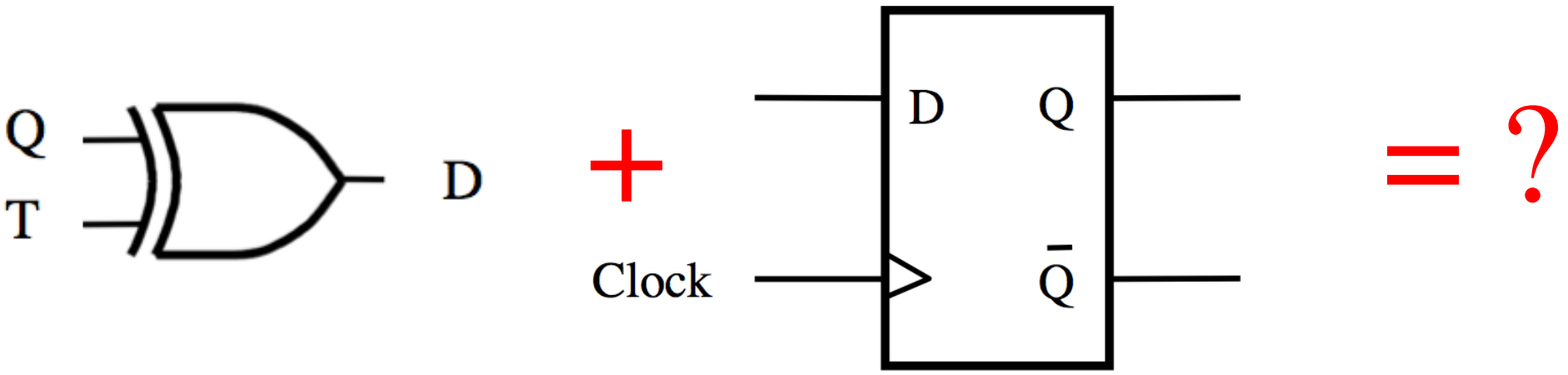
$$D = Q \oplus T$$

It is an XOR

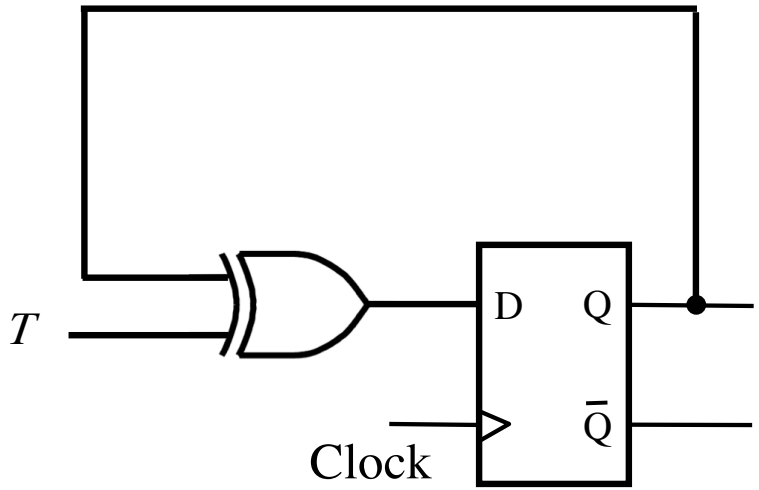


$$D = Q \oplus T$$

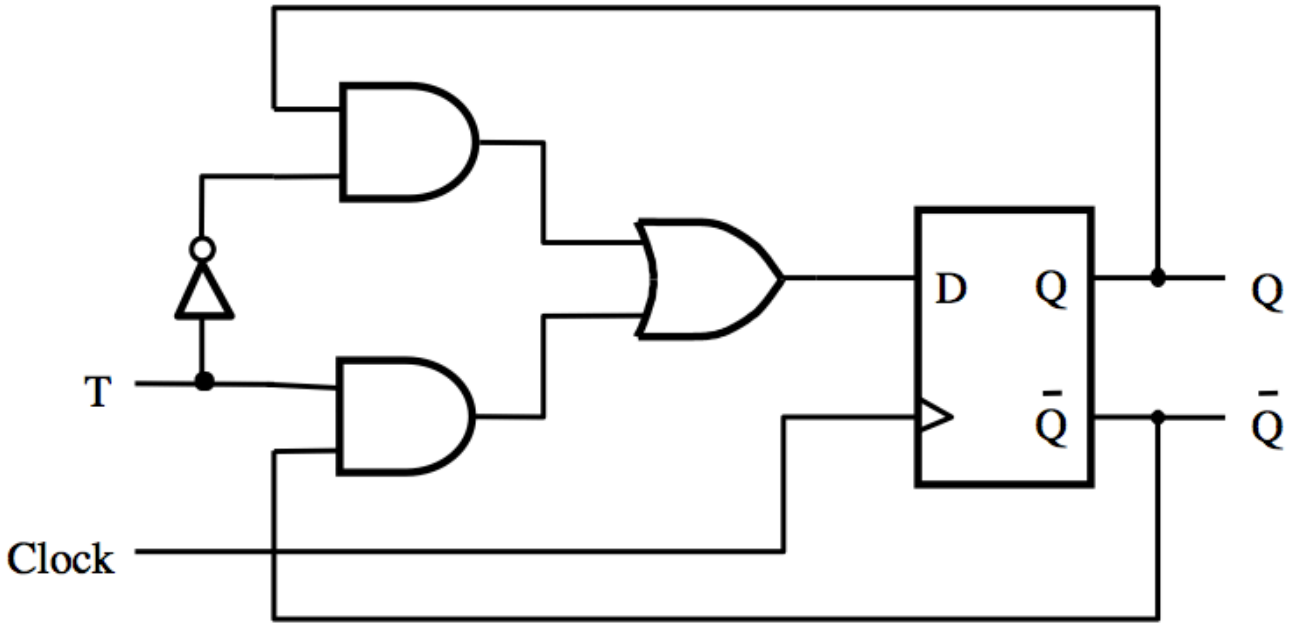
What is this?



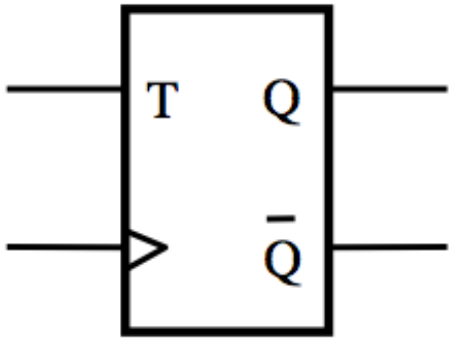
T Flip-Flop



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)

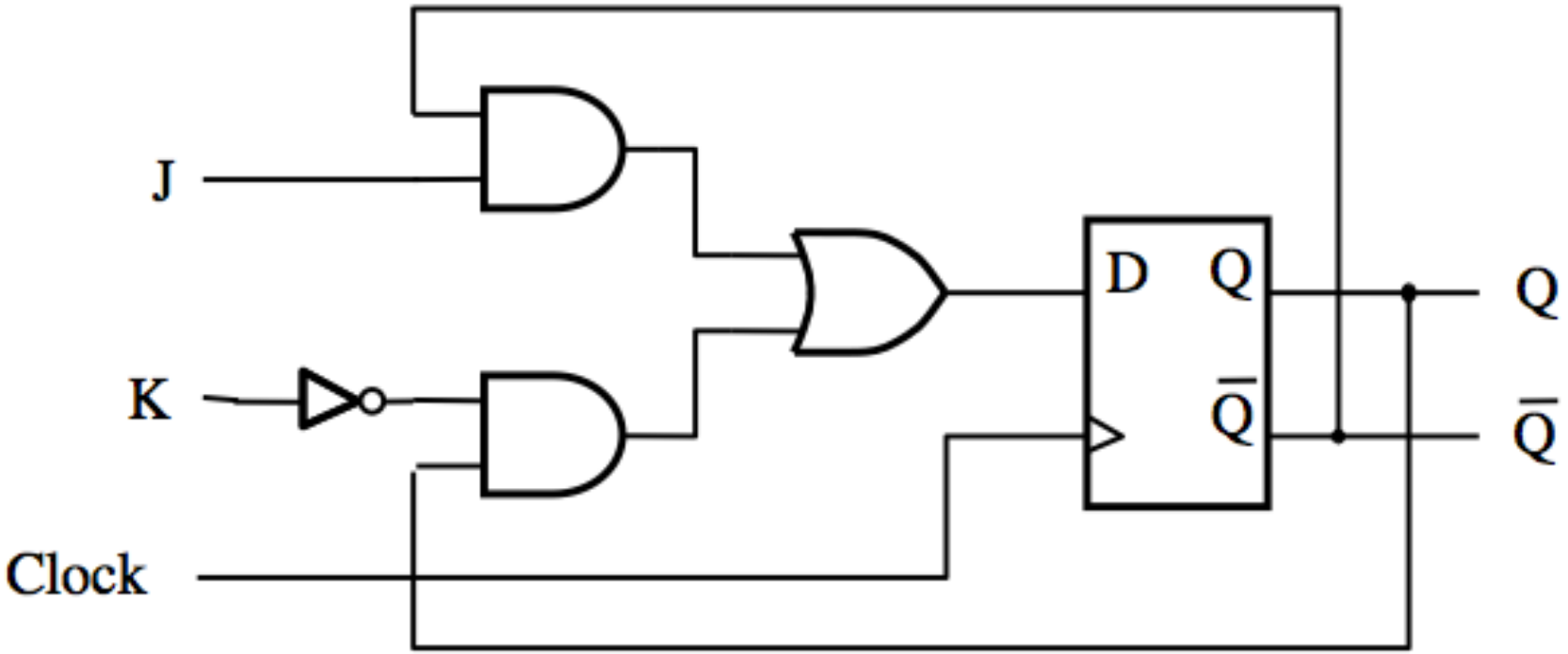
If T=0 then it stays in its current state

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

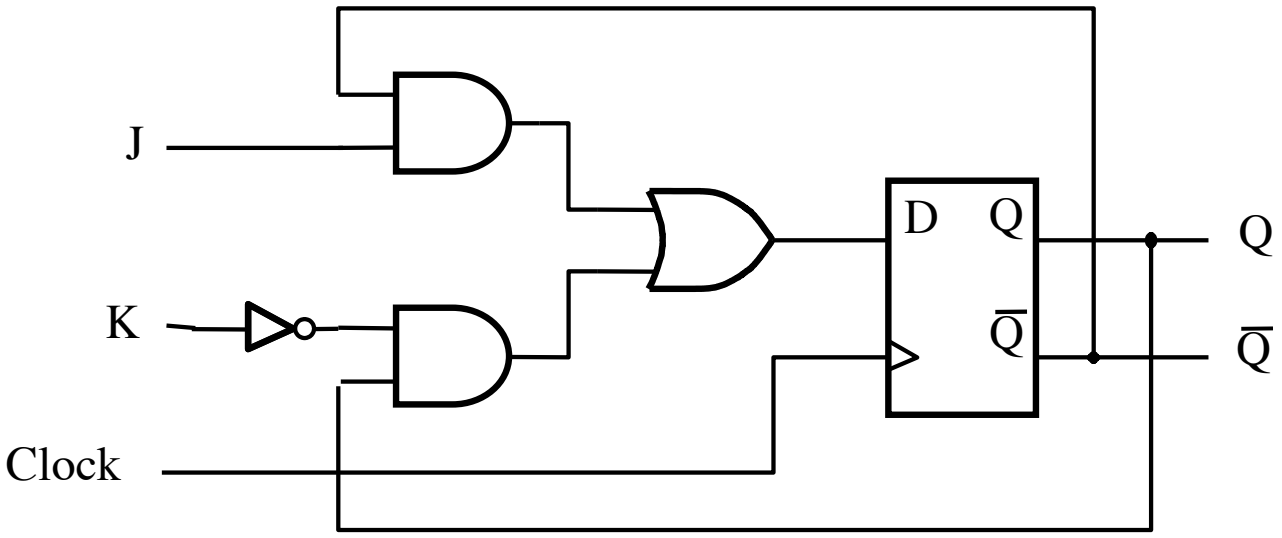
JK Flip-Flop



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

[Figure 5.16a from the textbook]

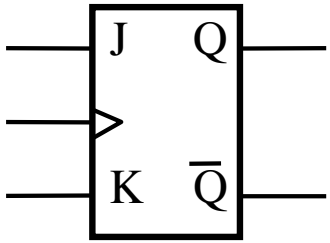
JK Flip-Flop



(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

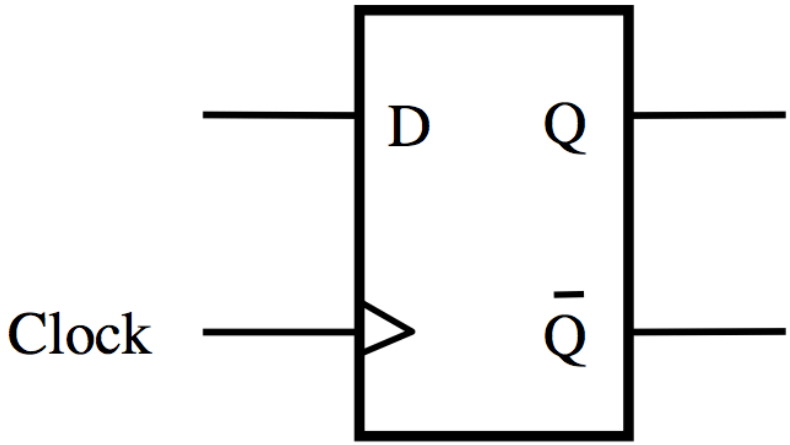
Just like SR It can be set and reset

J=S and K=R

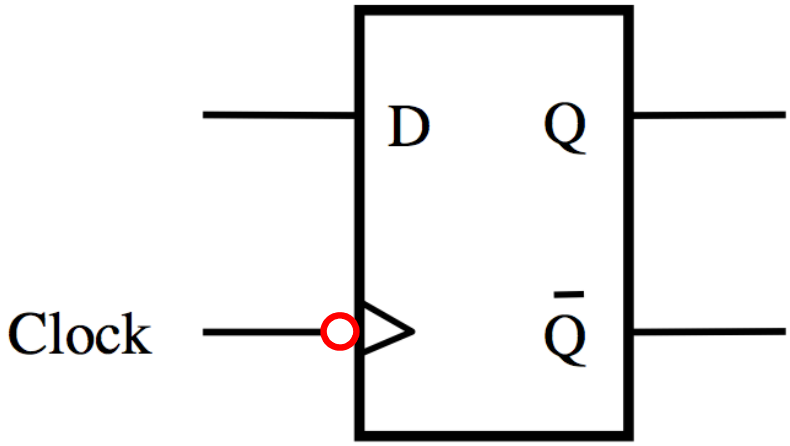
If J=K=1 then it behaves as a T flip-flop

Complete Wiring Diagrams

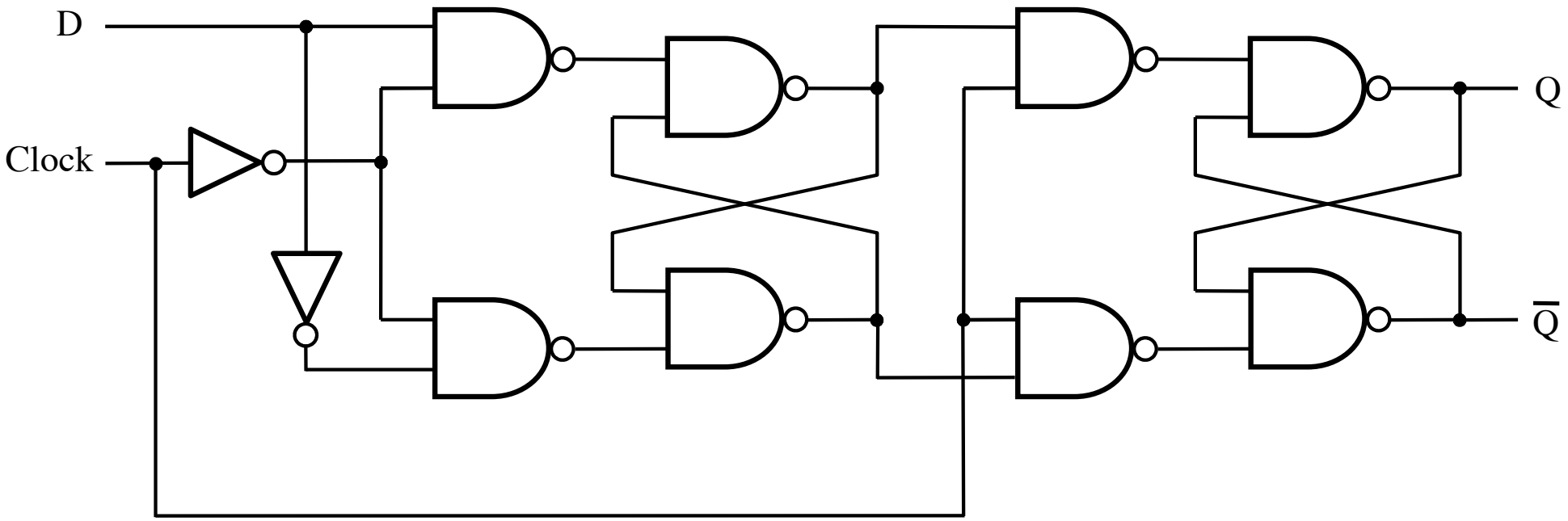
Positive-Edge-Triggered D Flip-Flop



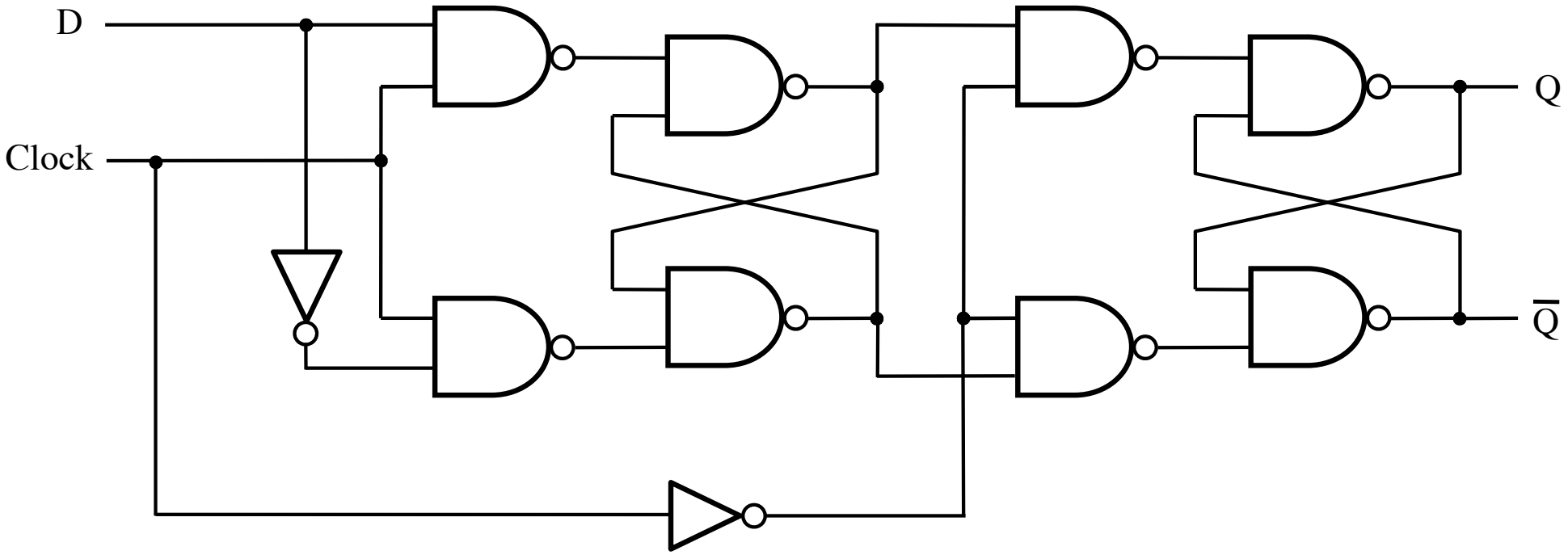
Negative-Edge-Triggered D Flip-Flop



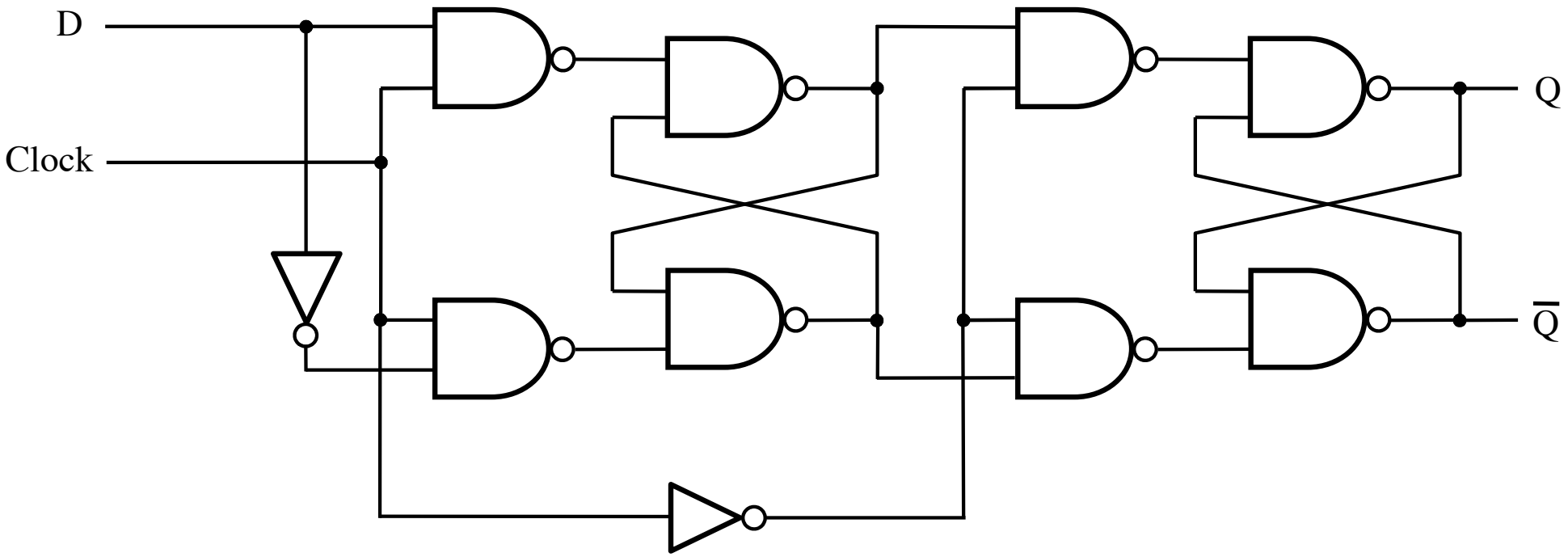
The Complete Wiring Diagram for a Positive-Edge-Triggered D Flip-Flop



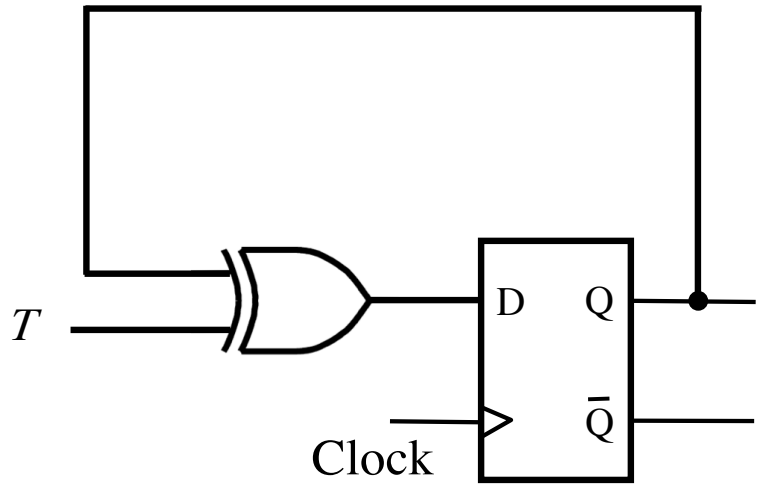
The Complete Wiring Diagram for a Negative-Edge-Triggered D Flip-Flop



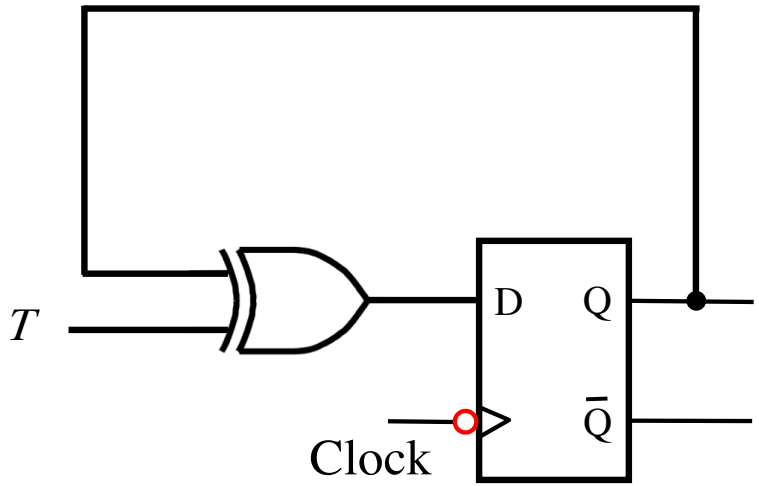
The Complete Wiring Diagram for a Negative-Edge-Triggered D Flip-Flop



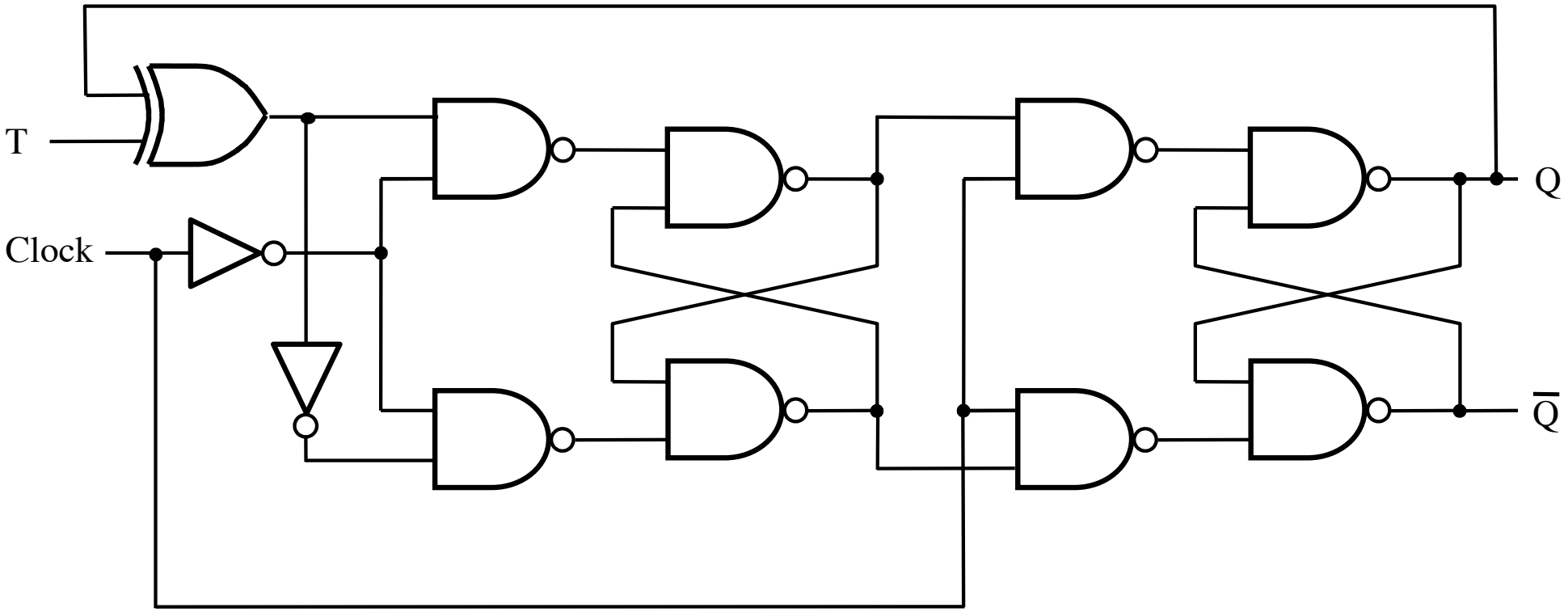
Positive-Edge-Triggered T Flip-Flop



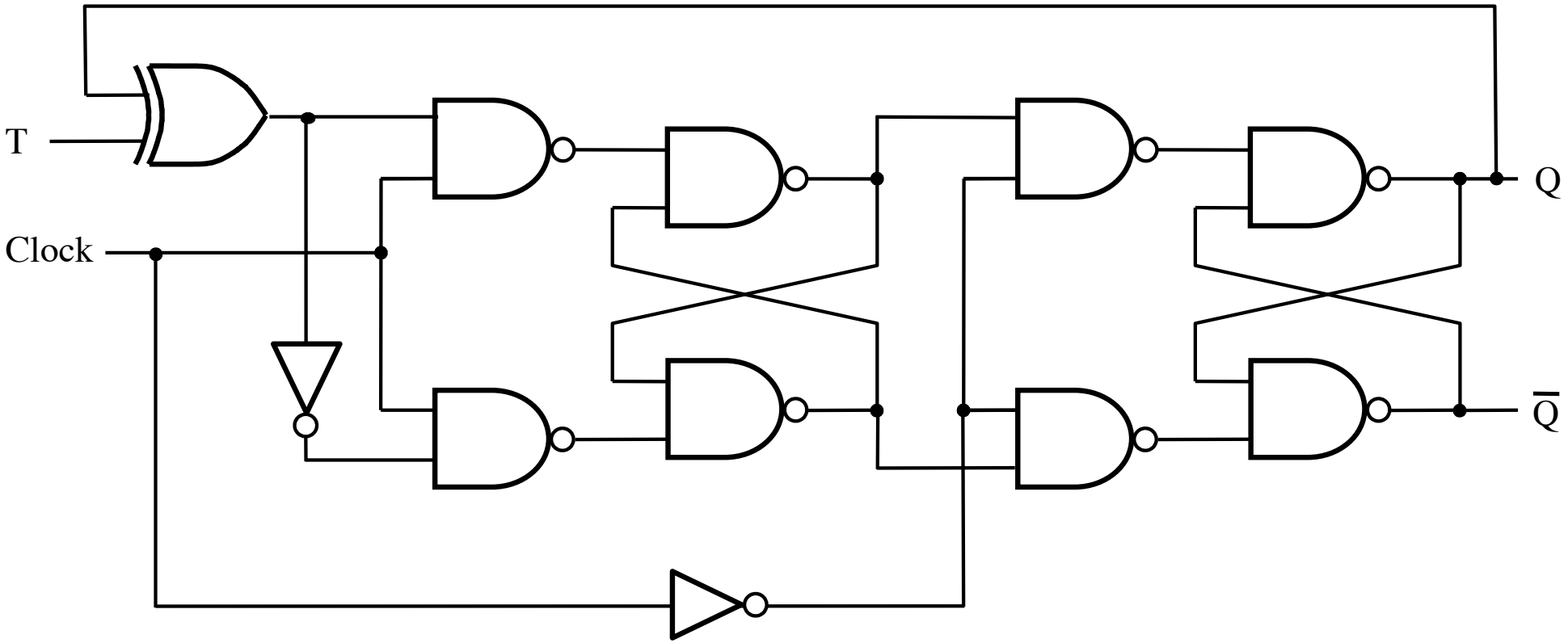
Negative-Edge-Triggered T Flip-Flop



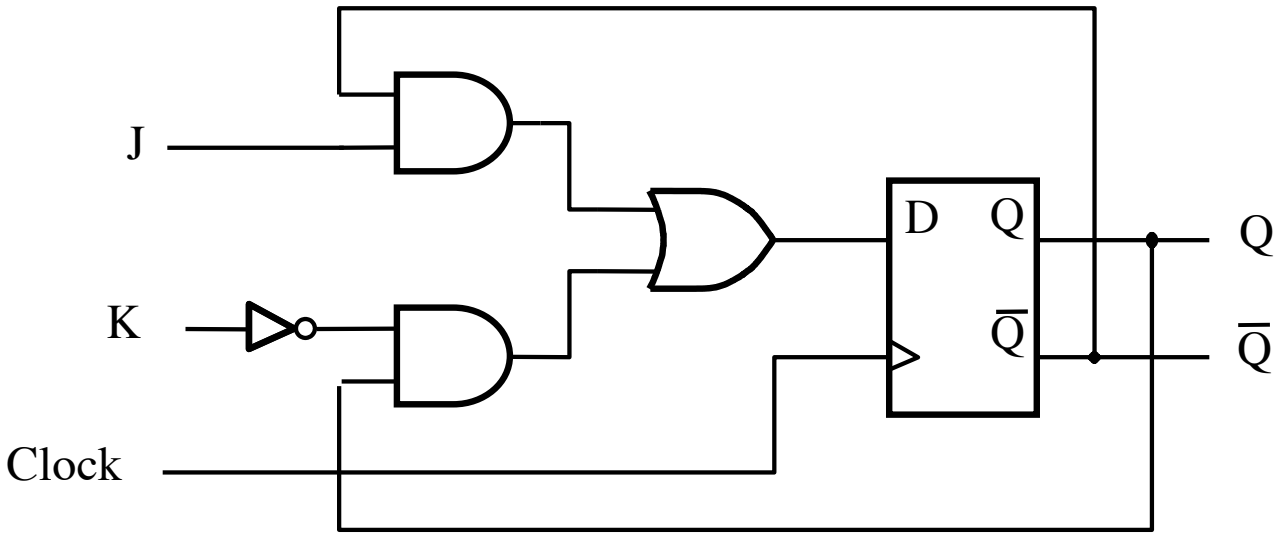
The Complete Wiring Diagram for a Positive-Edge-Triggered D Flip-Flop



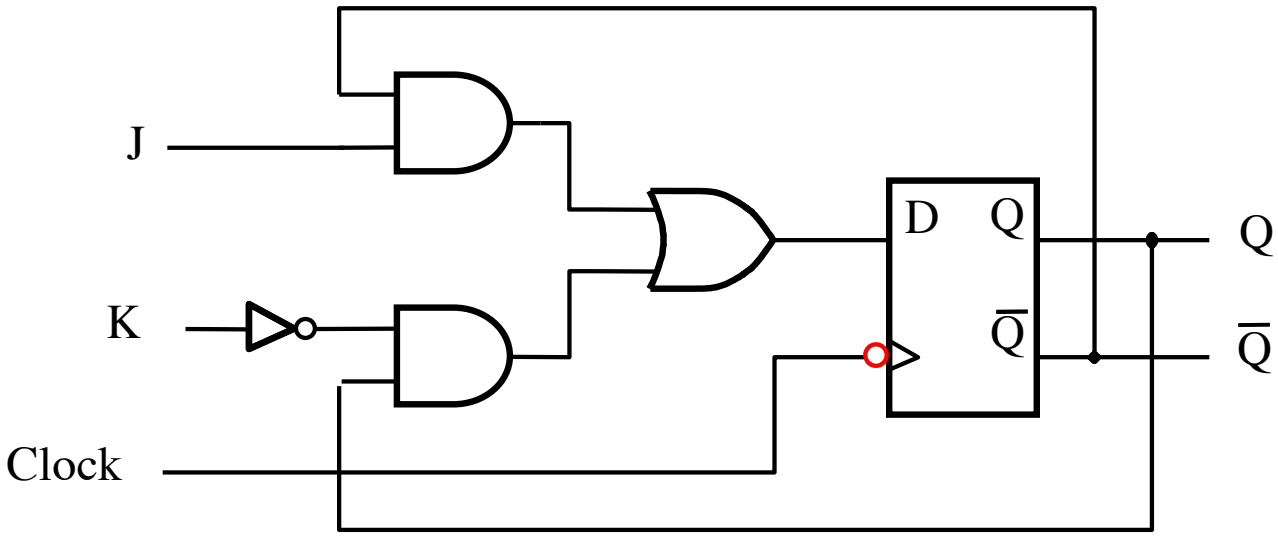
The Complete Wiring Diagram for a Negative-Edge-Triggered D Flip-Flop



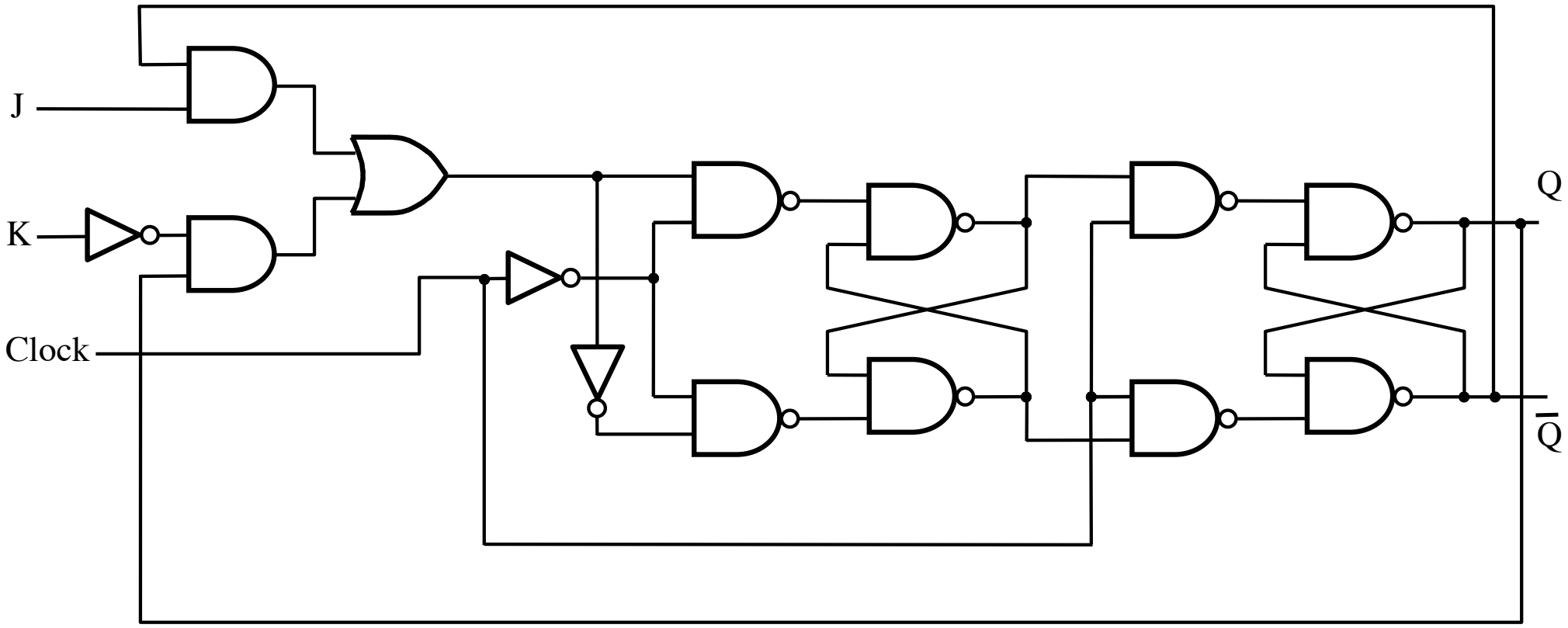
Positive-Edge-Triggered JK Flip-Flop



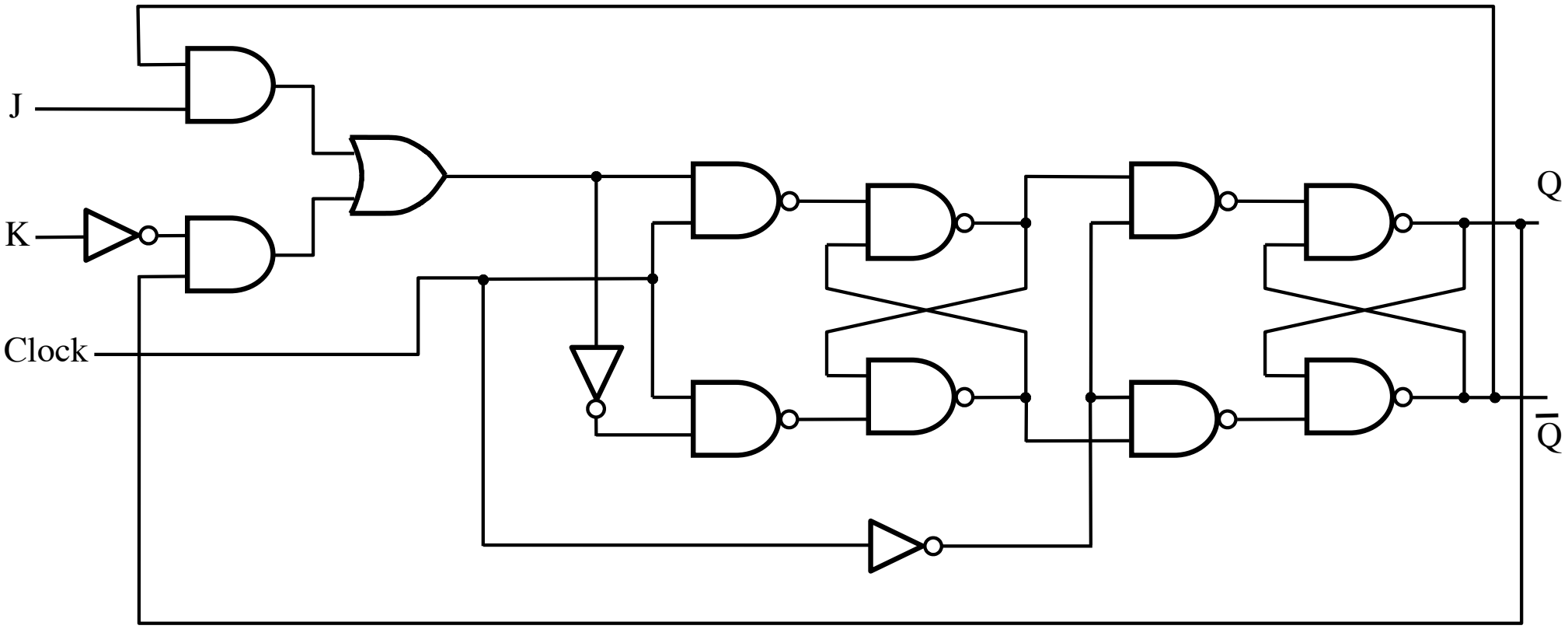
Negative-Edge-Triggered JK Flip-Flop



The Complete Wiring Diagram for a Positive-Edge-Triggered JK Flip-Flop



The Complete Wiring Diagram for a Negative-Edge-Triggered JK Flip-Flop



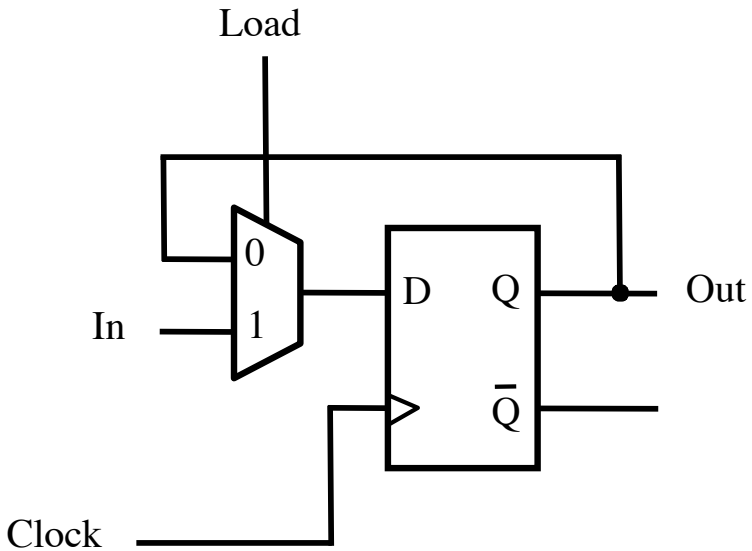
Registers

Register
(Definition)

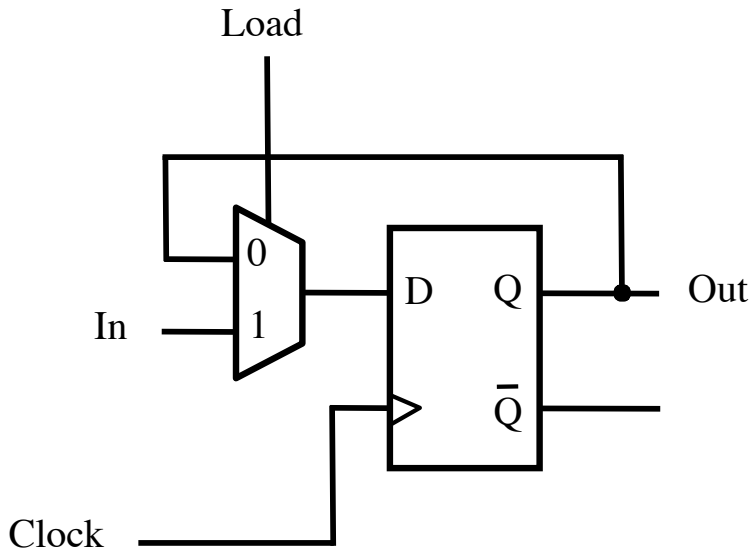
An n-bit structure consisting of flip-flops.

Parallel-Access Register

1-Bit Parallel-Access Register



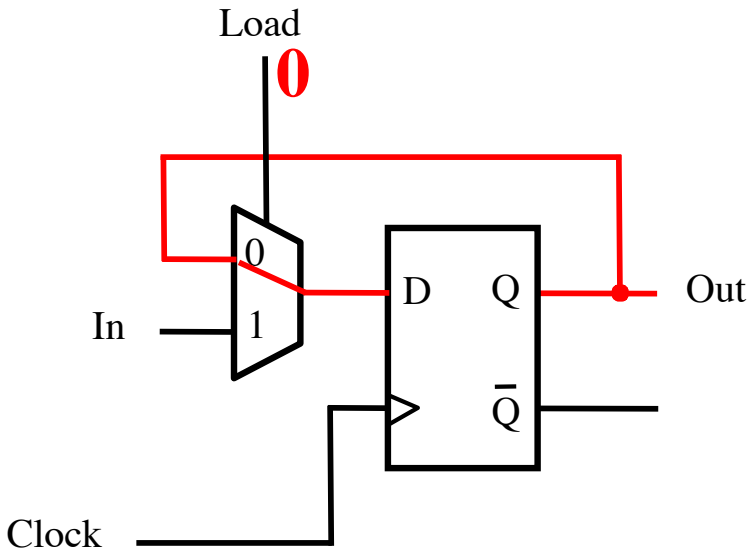
1-Bit Parallel-Access Register



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.

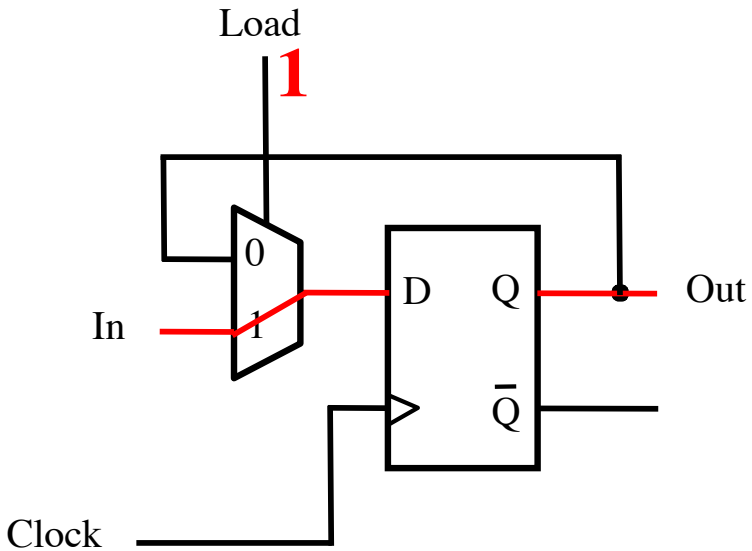
The output of this circuit is the Q output of the flip-flop.

1-Bit Parallel-Access Register



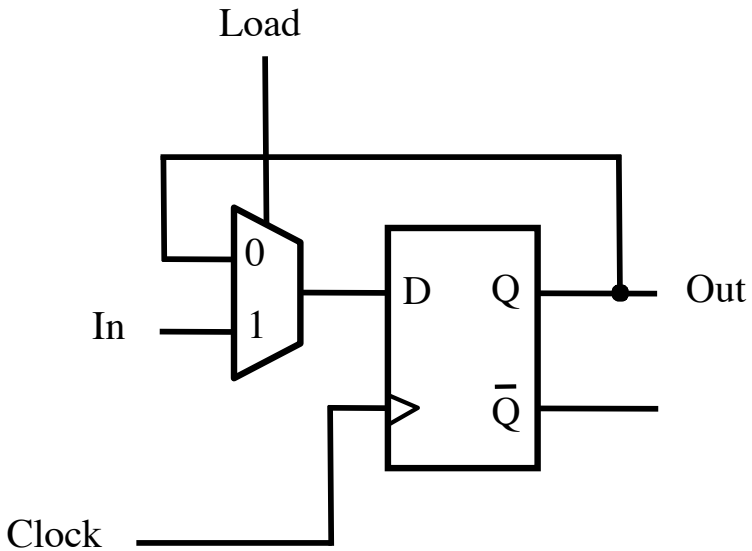
If Load = 0, then retain the old value.

1-Bit Parallel-Access Register



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

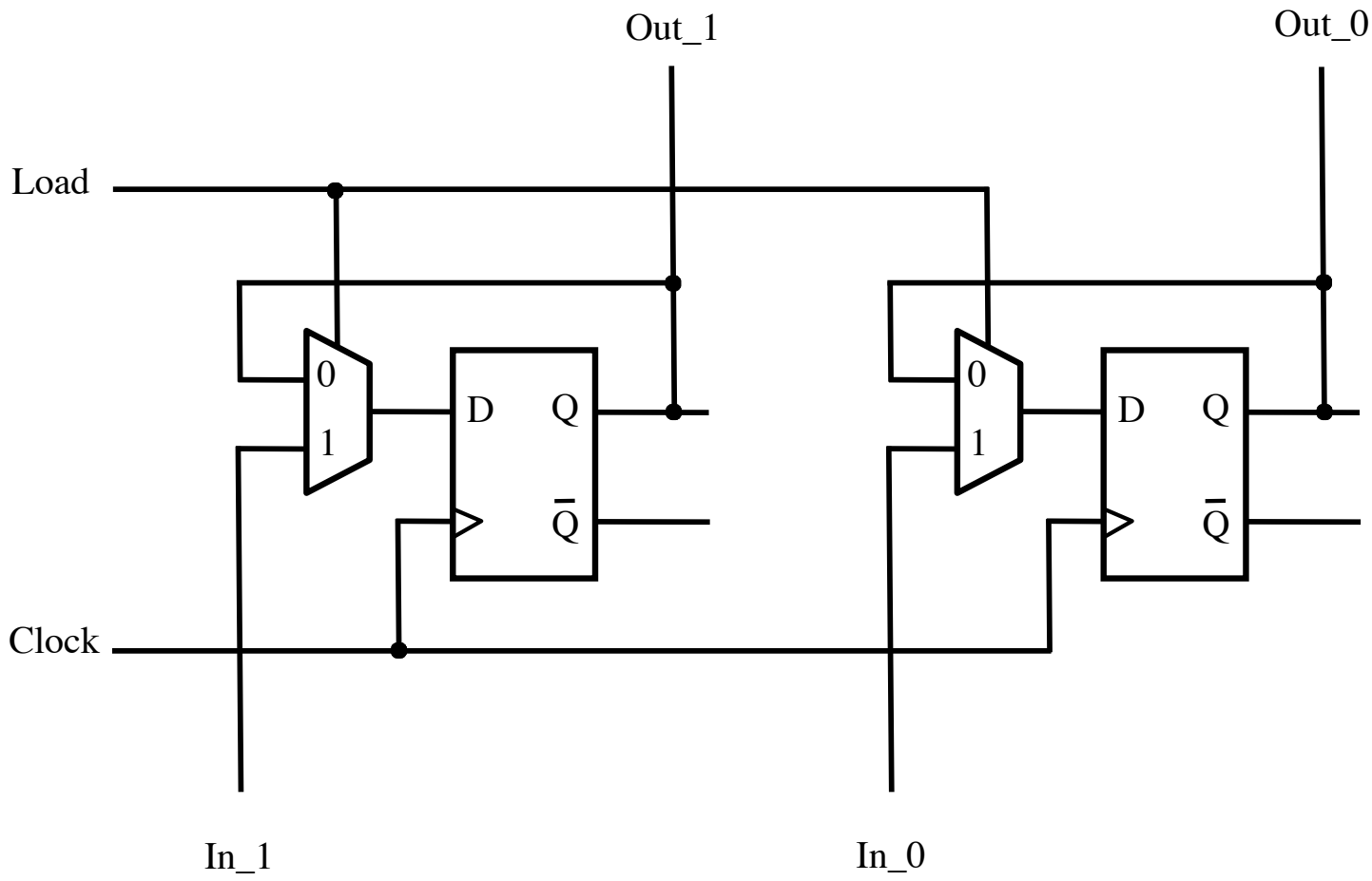
1-Bit Parallel-Access Register



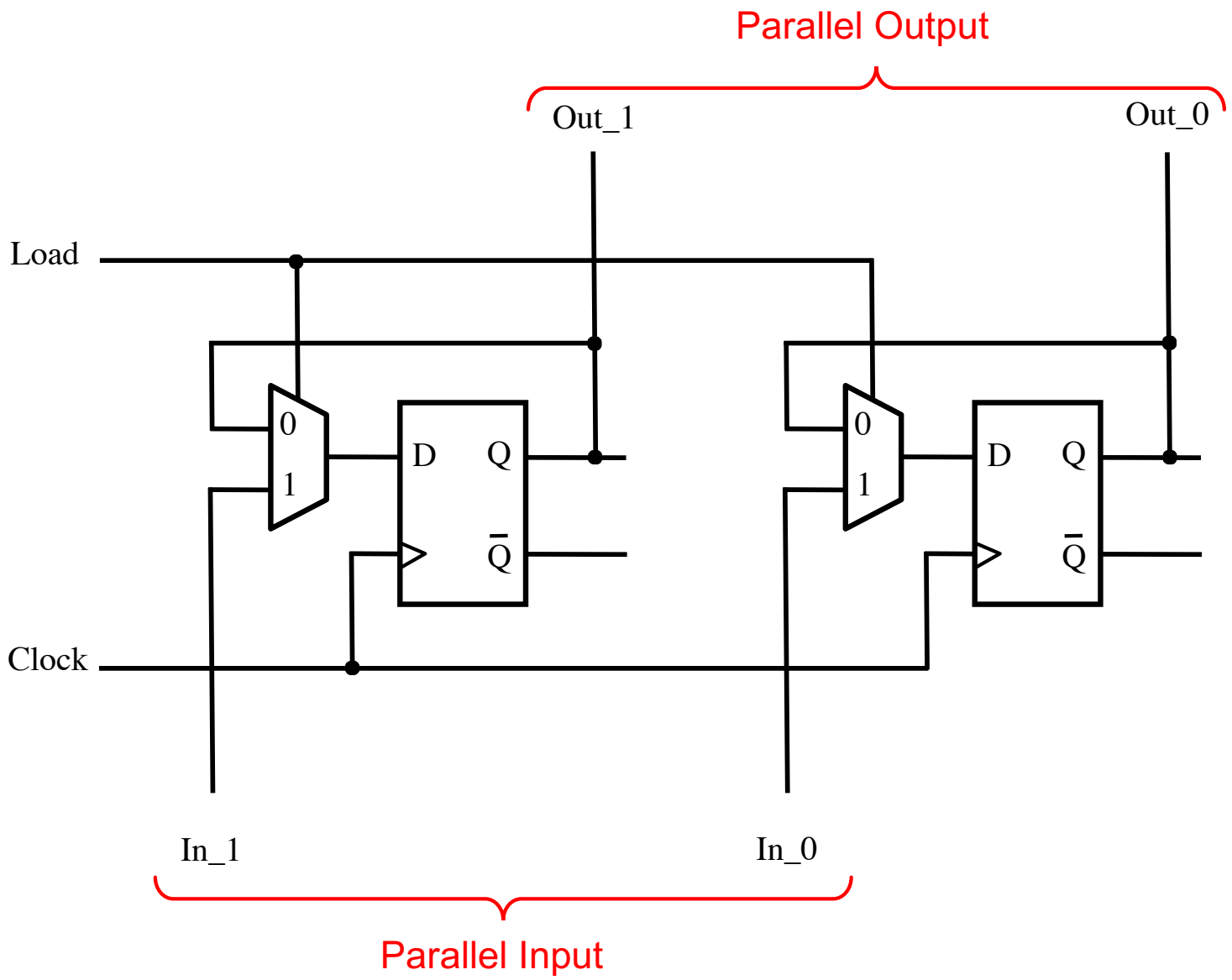
If Load = 0, then retain the old value.

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

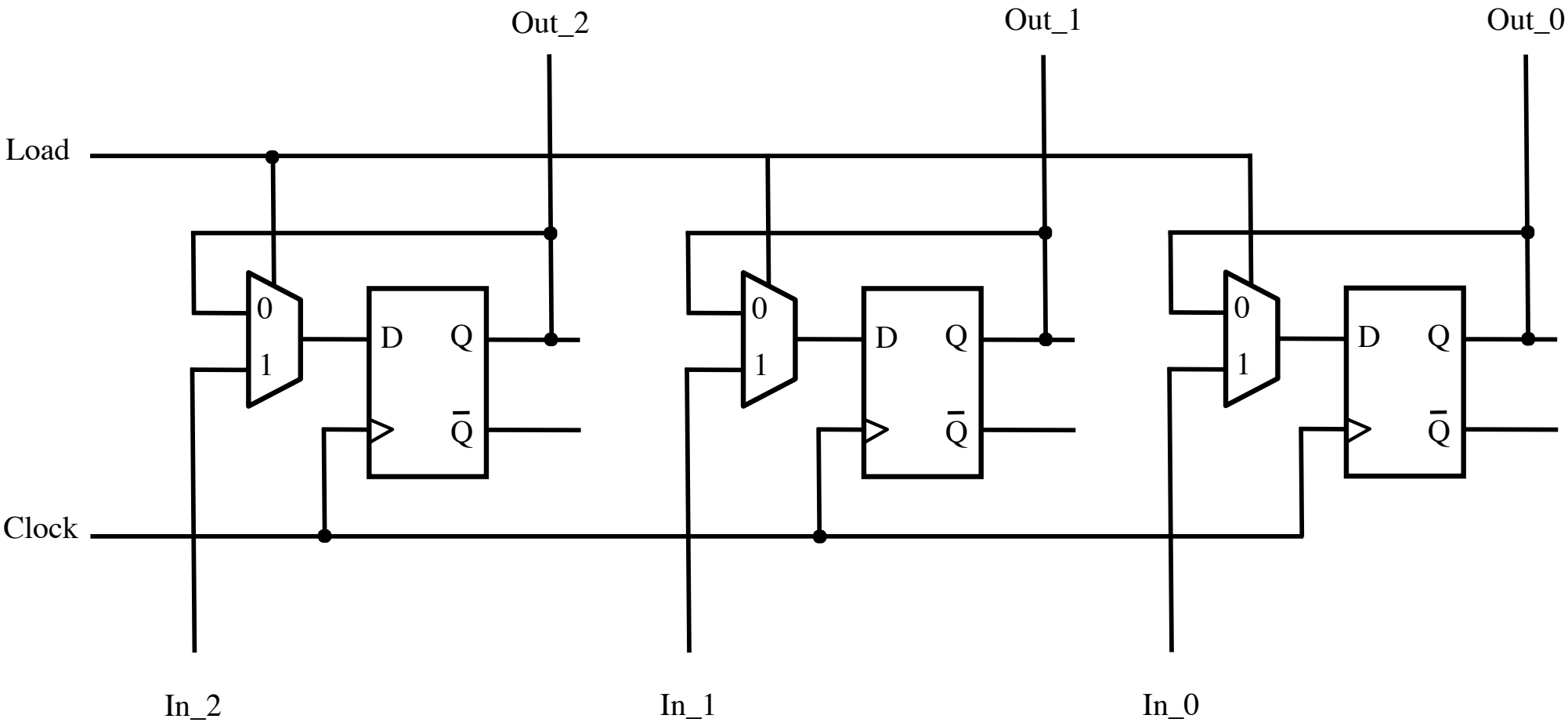
2-Bit Parallel-Access Register



2-Bit Parallel-Access Register

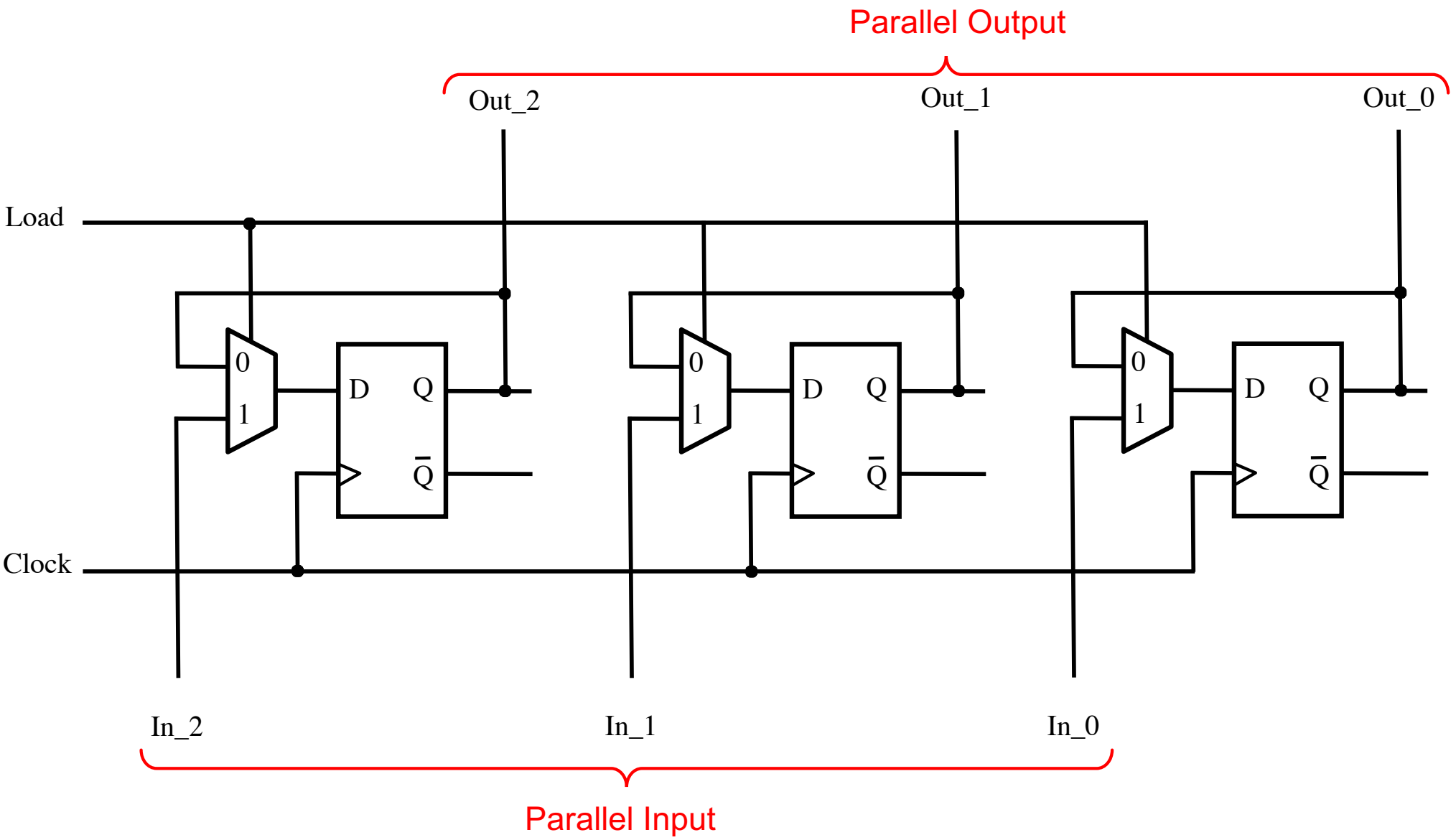


3-Bit Parallel-Access Register

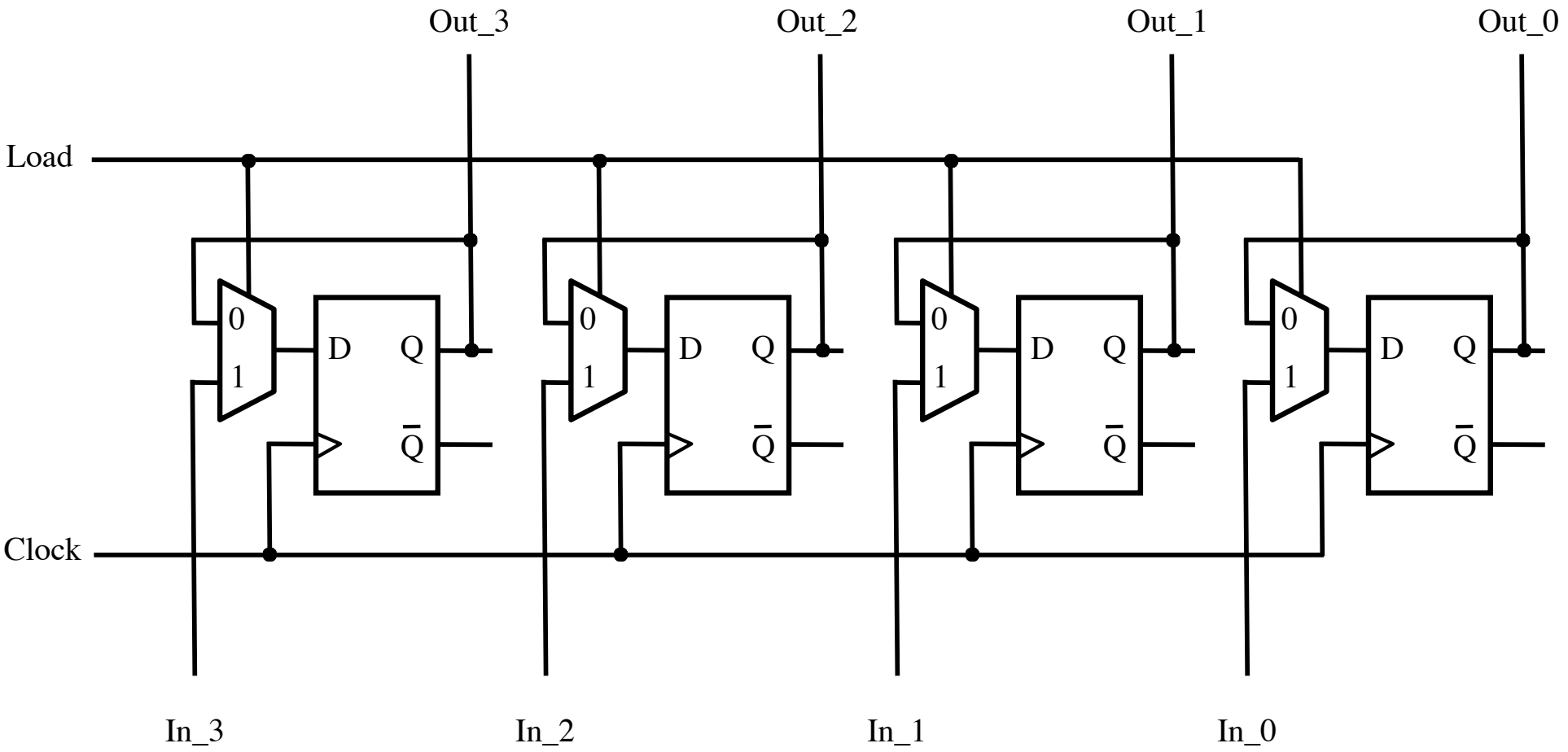


Notice that all flip-flops are on the same clock cycle.

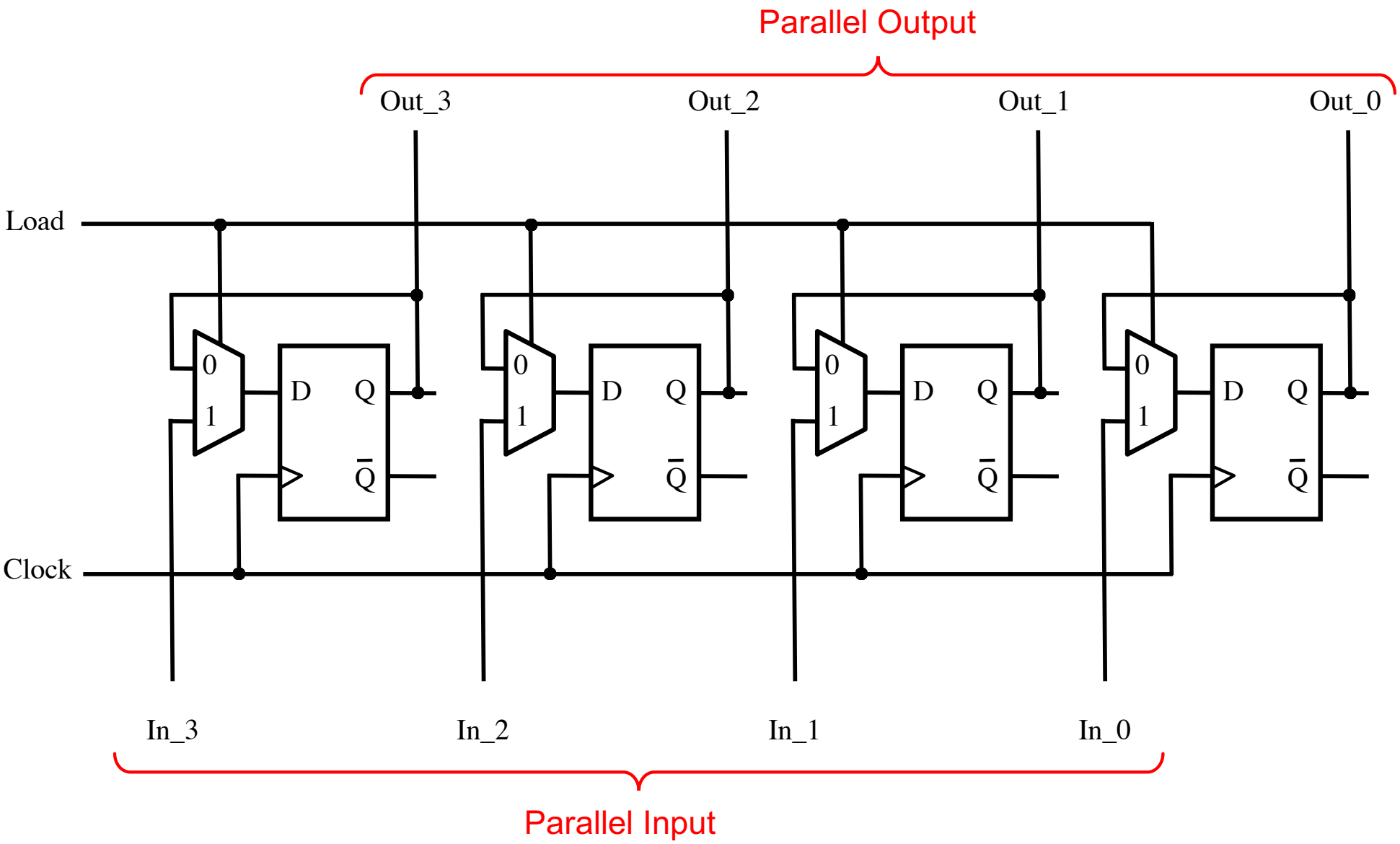
3-Bit Parallel-Access Register



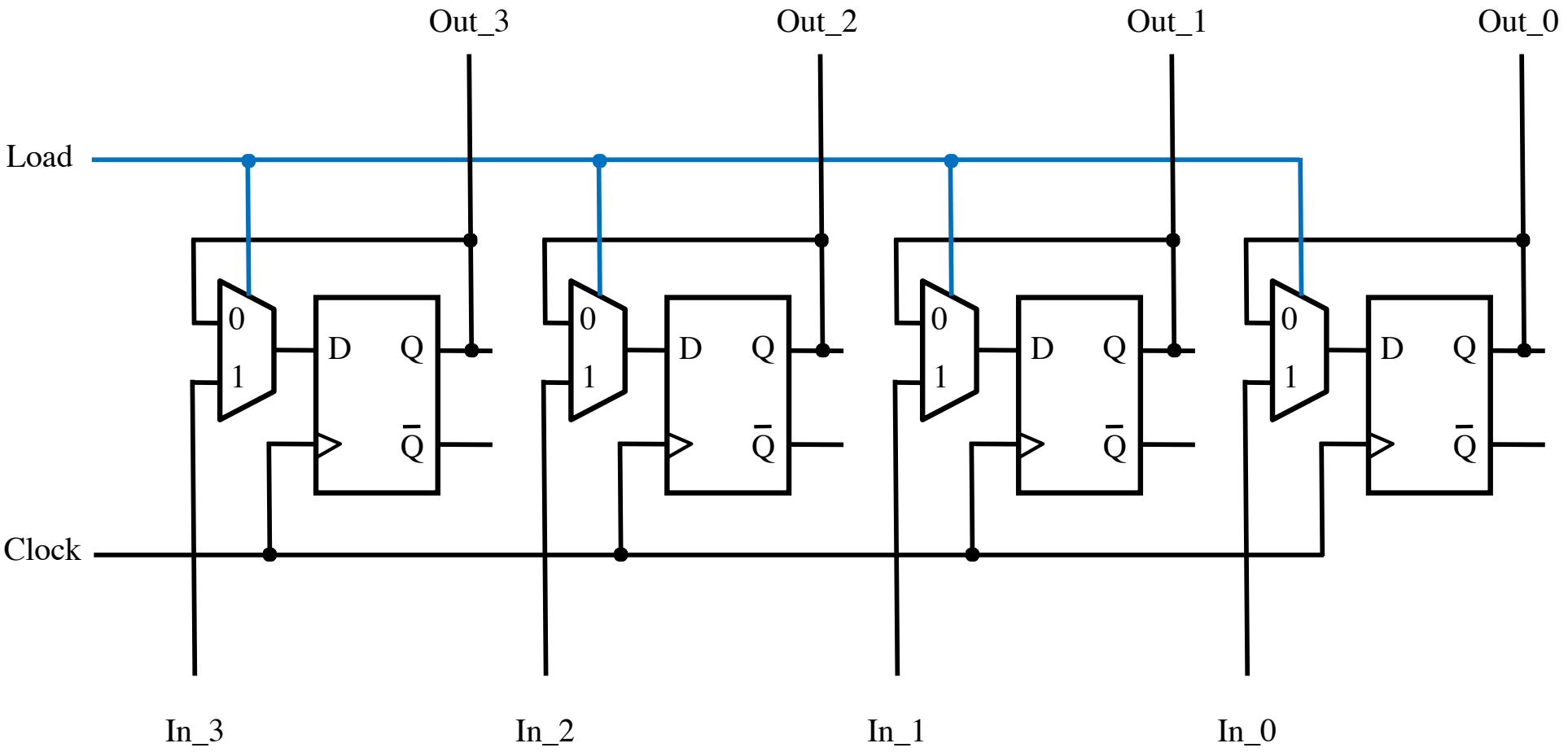
4-Bit Parallel-Access Register



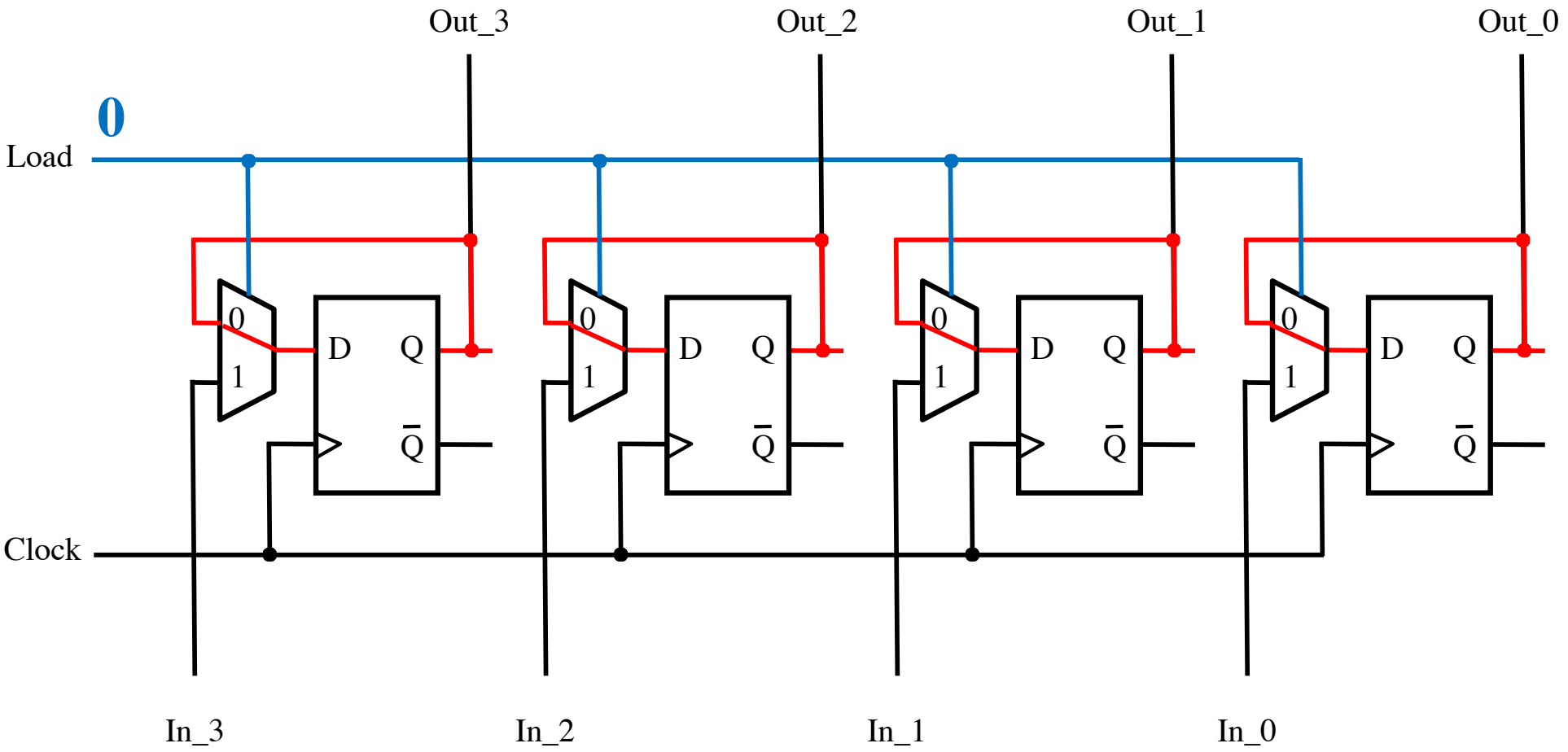
4-Bit Parallel-Access Register



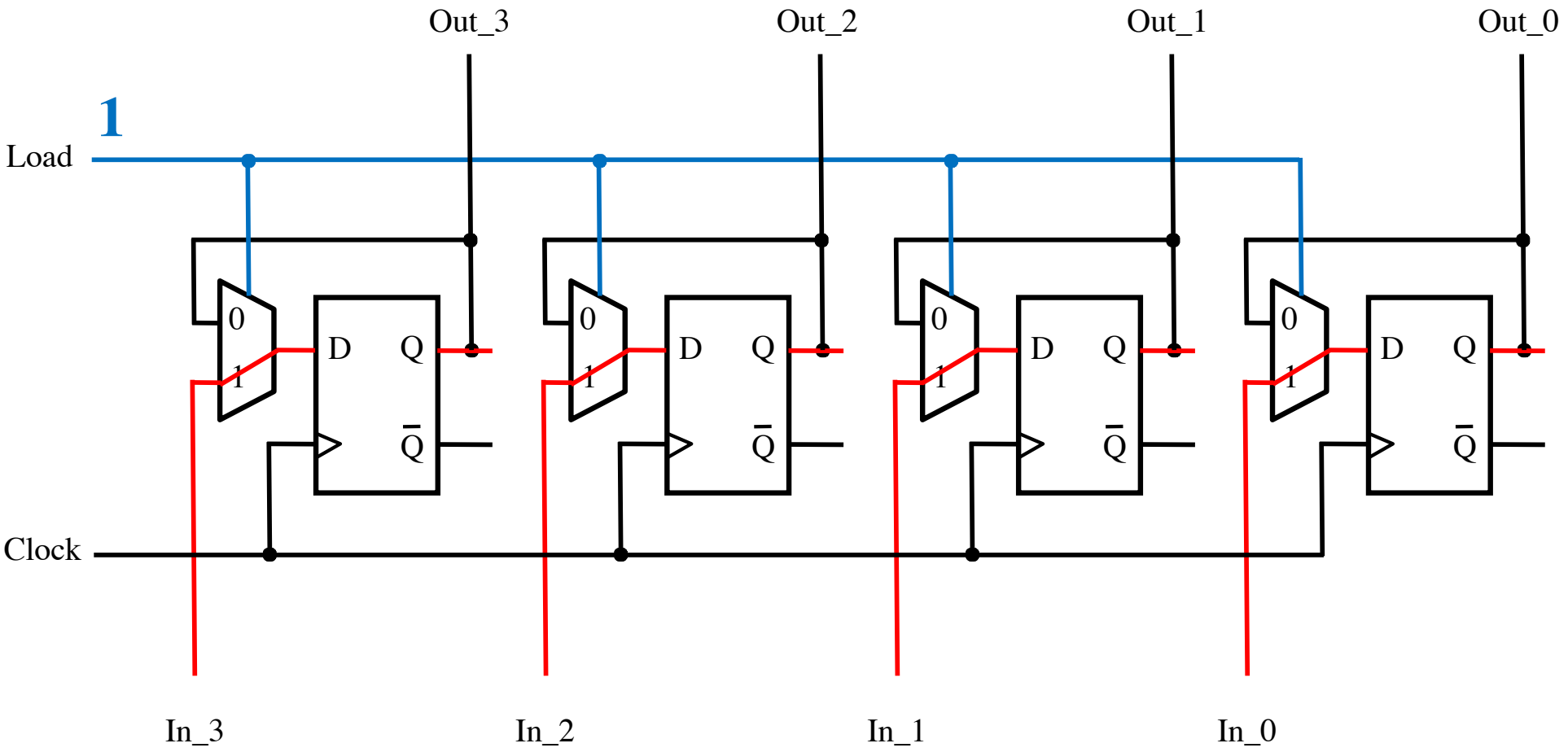
4-Bit Parallel-Access Register



4-Bit Parallel-Access Register

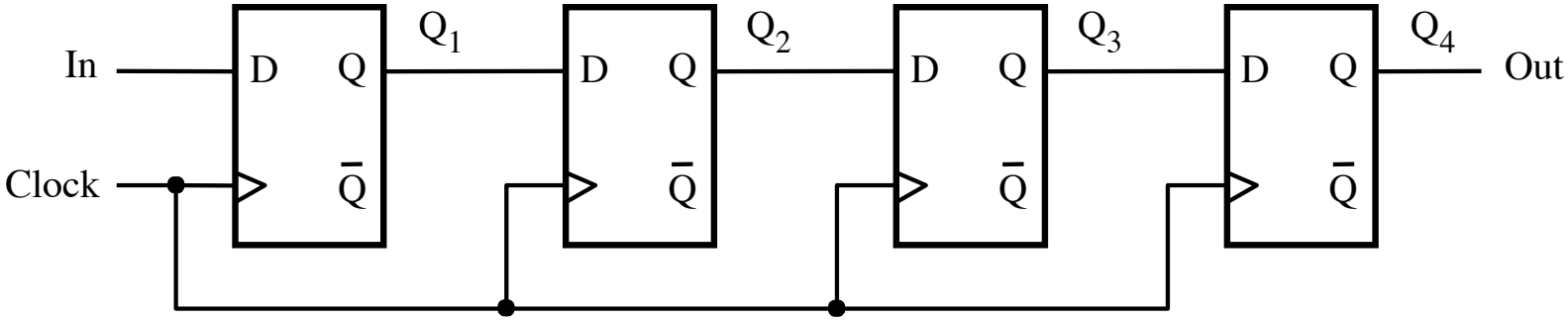


4-Bit Parallel-Access Register



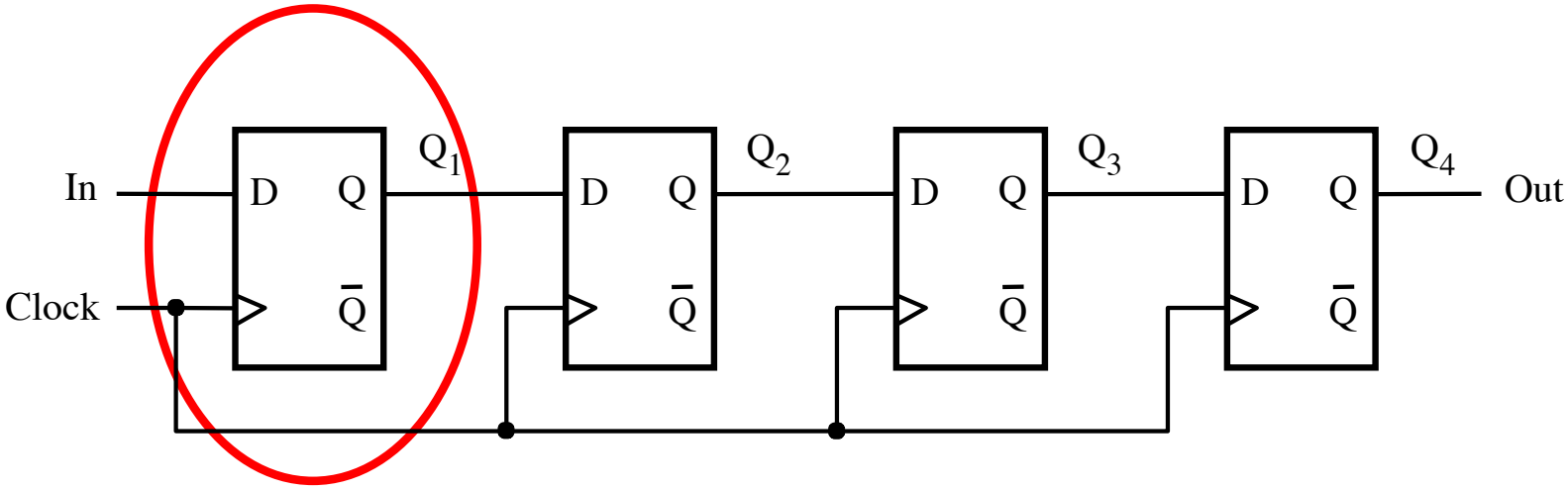
Shift Register

A simple shift register



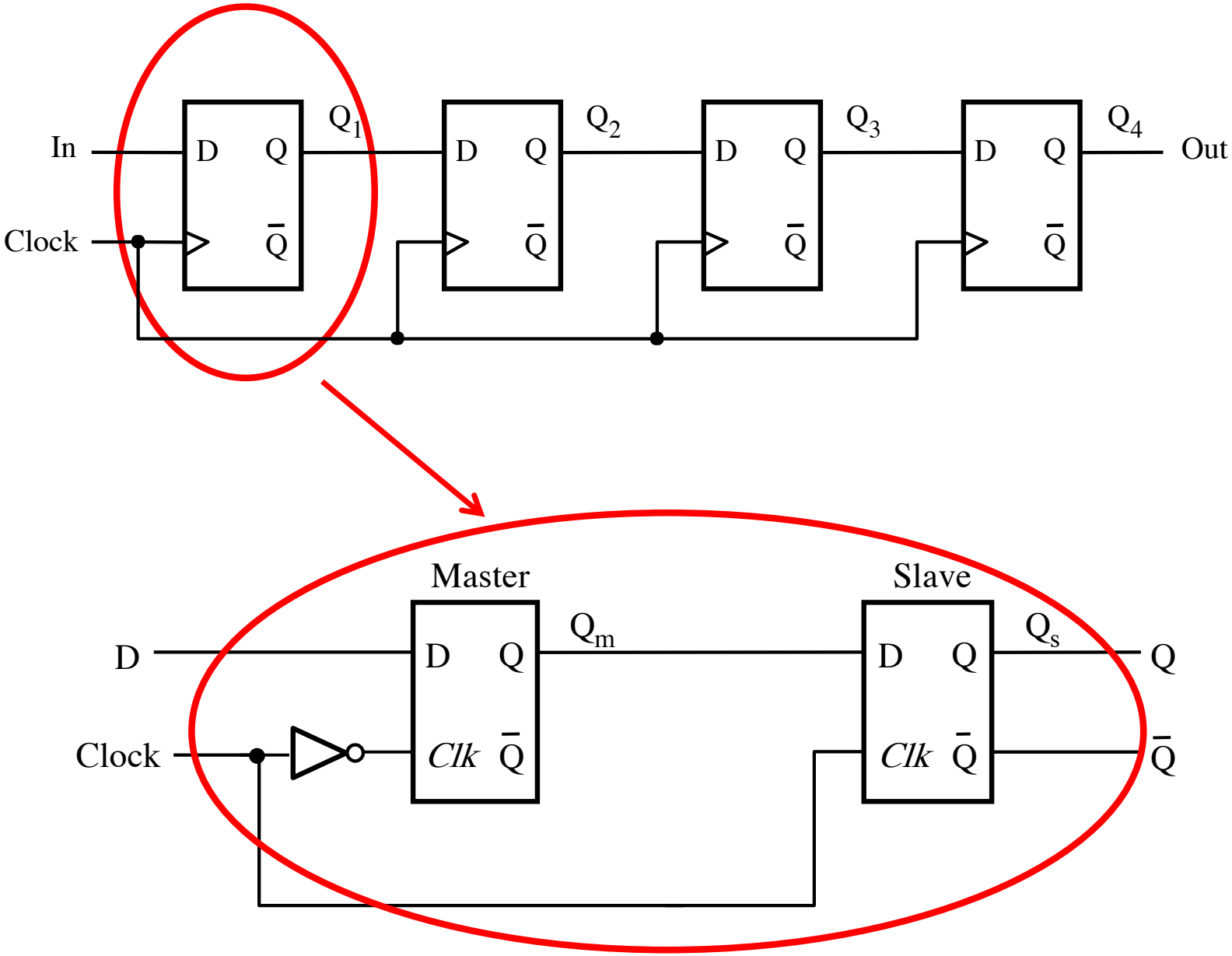
[Figure 5.17a from the textbook]

A simple shift register

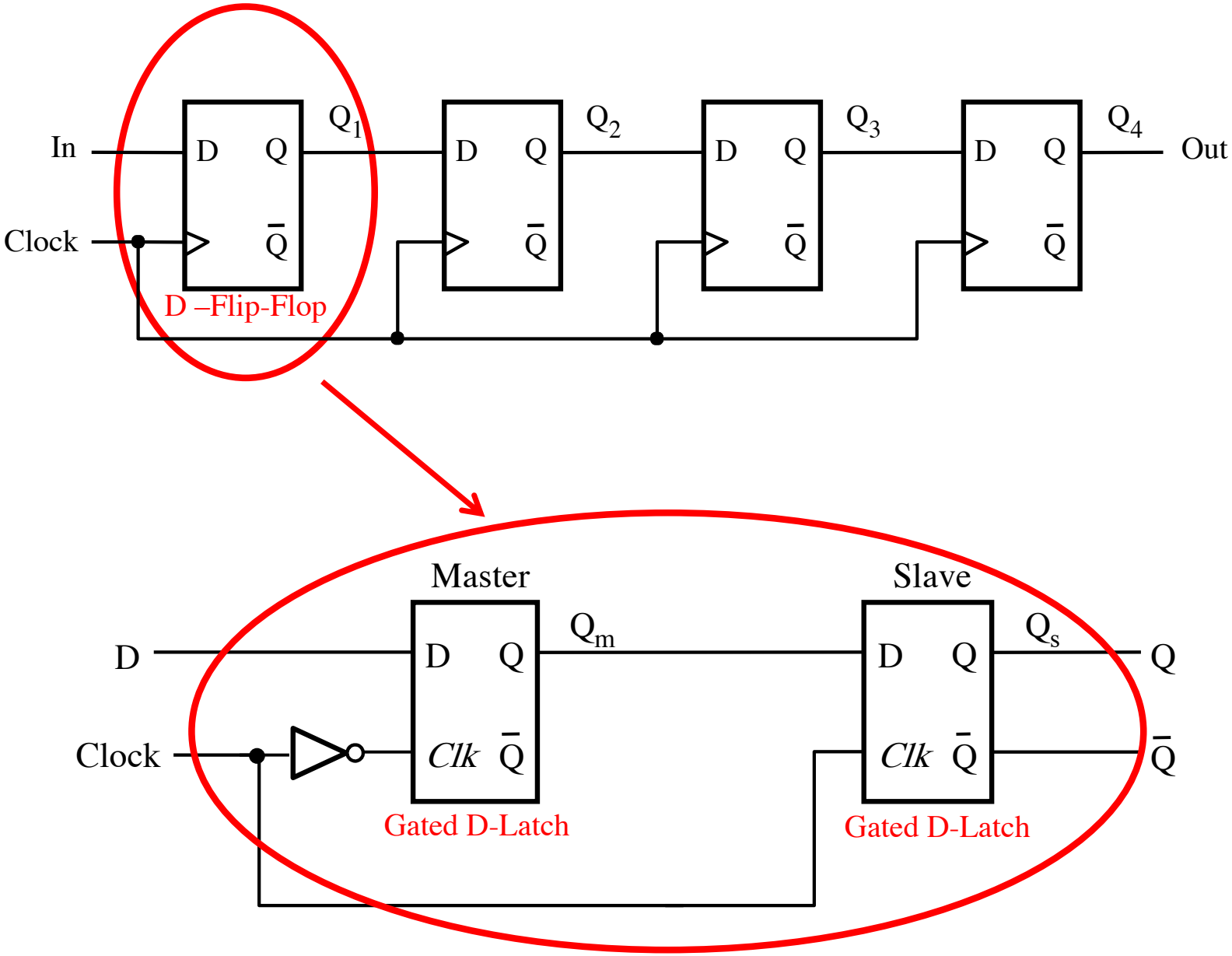


Positive-edge-triggered
D Flip-Flop

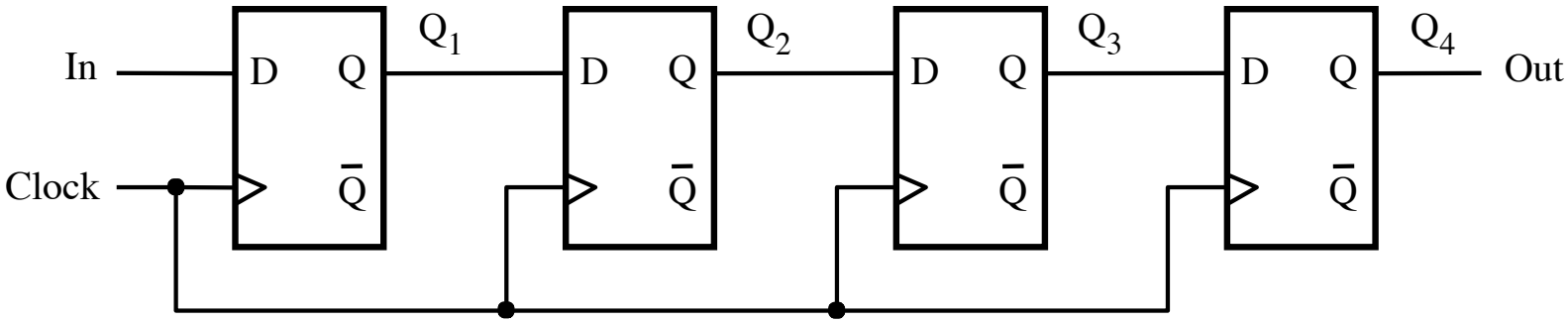
A simple shift register



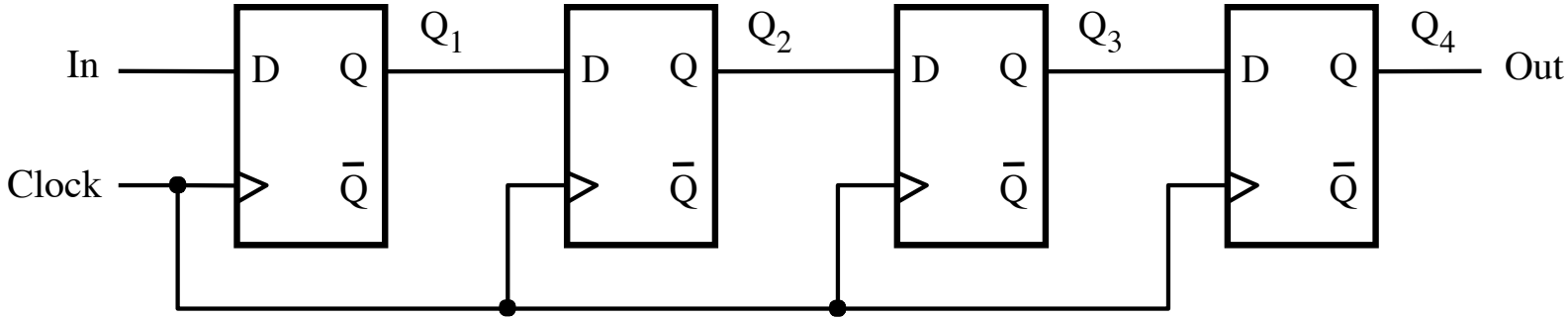
A simple shift register

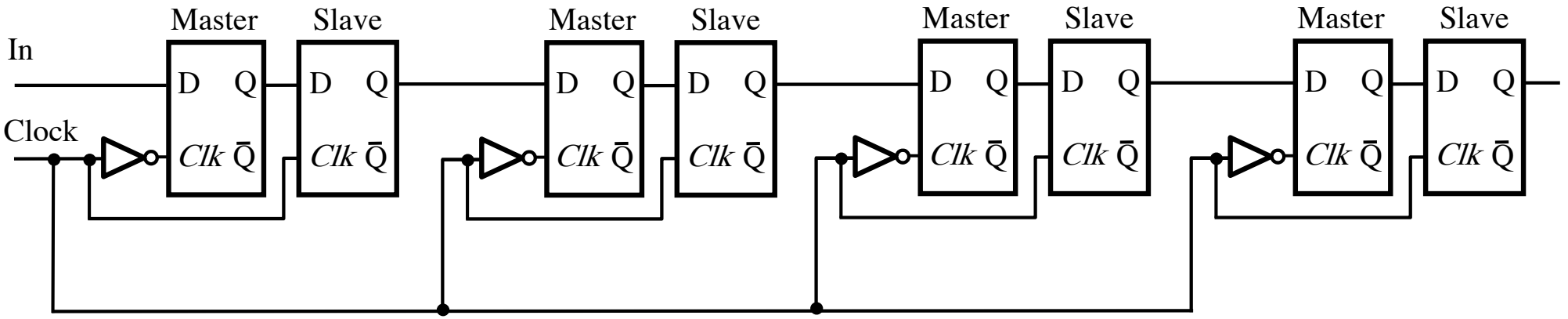


A simple shift register

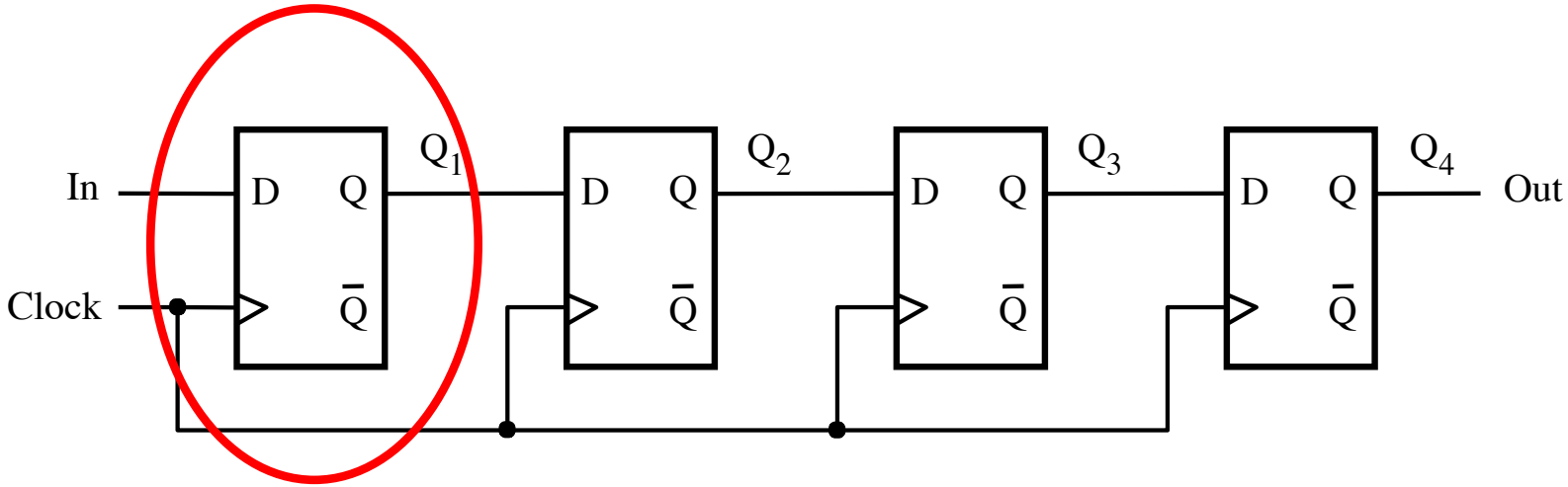


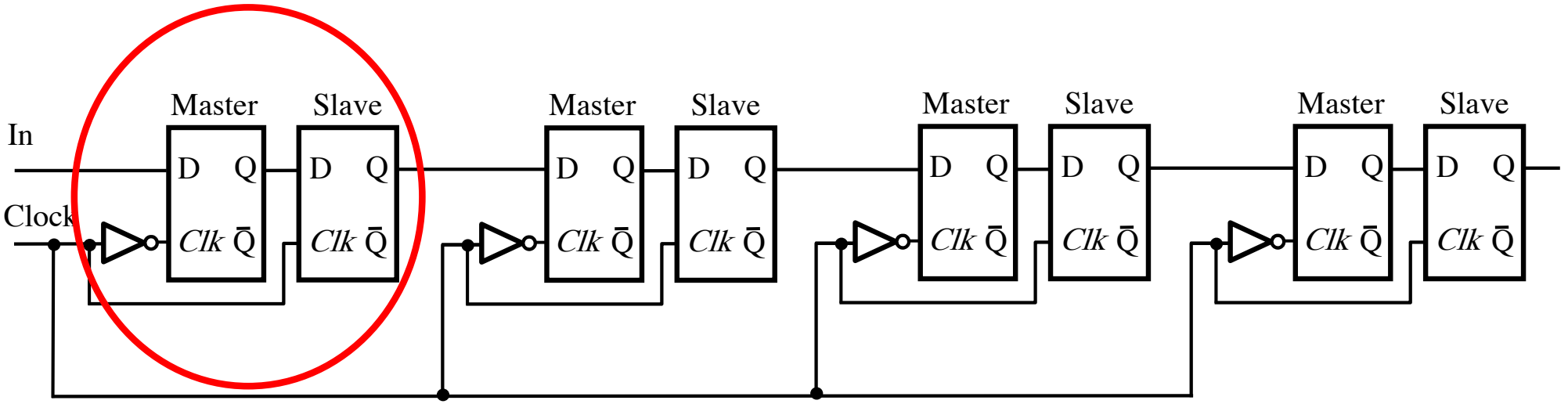
A simple shift register



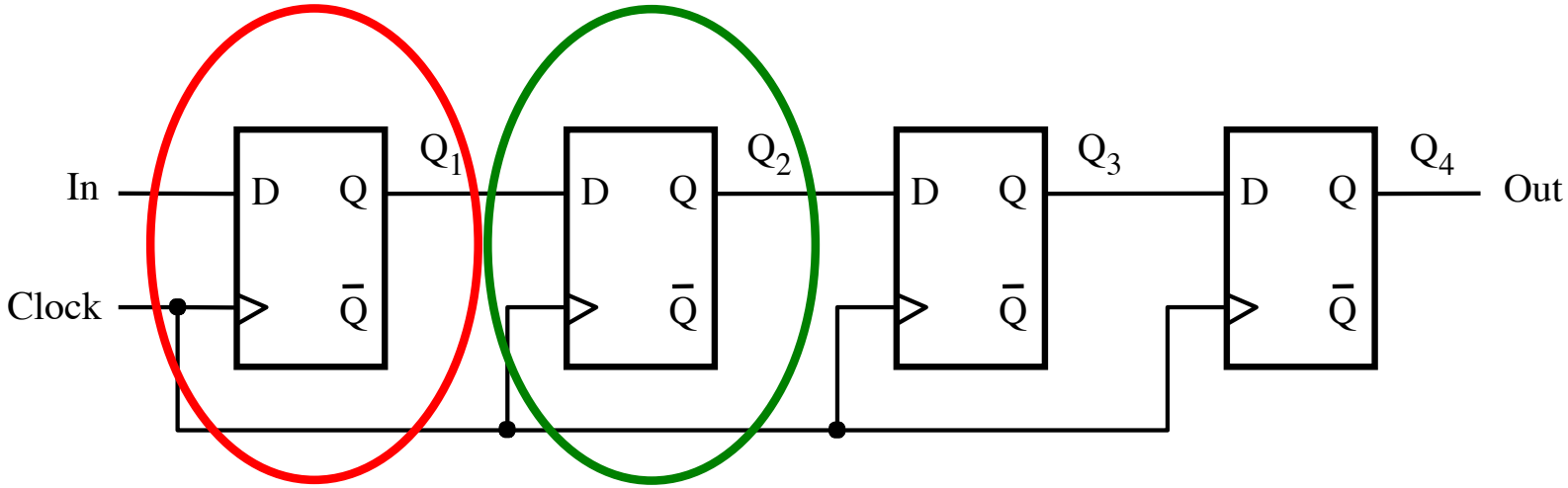


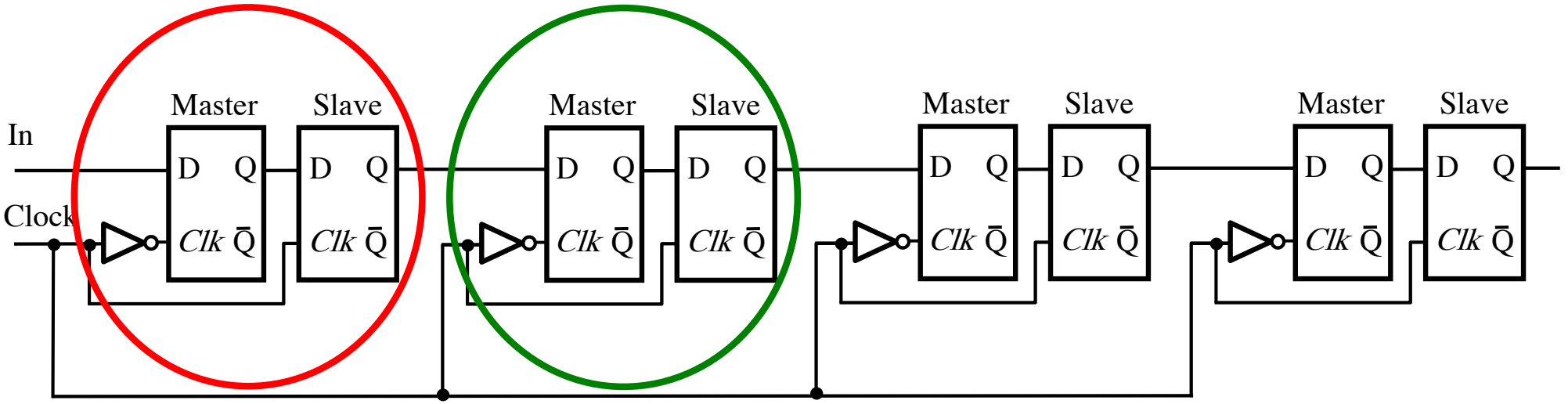
A simple shift register



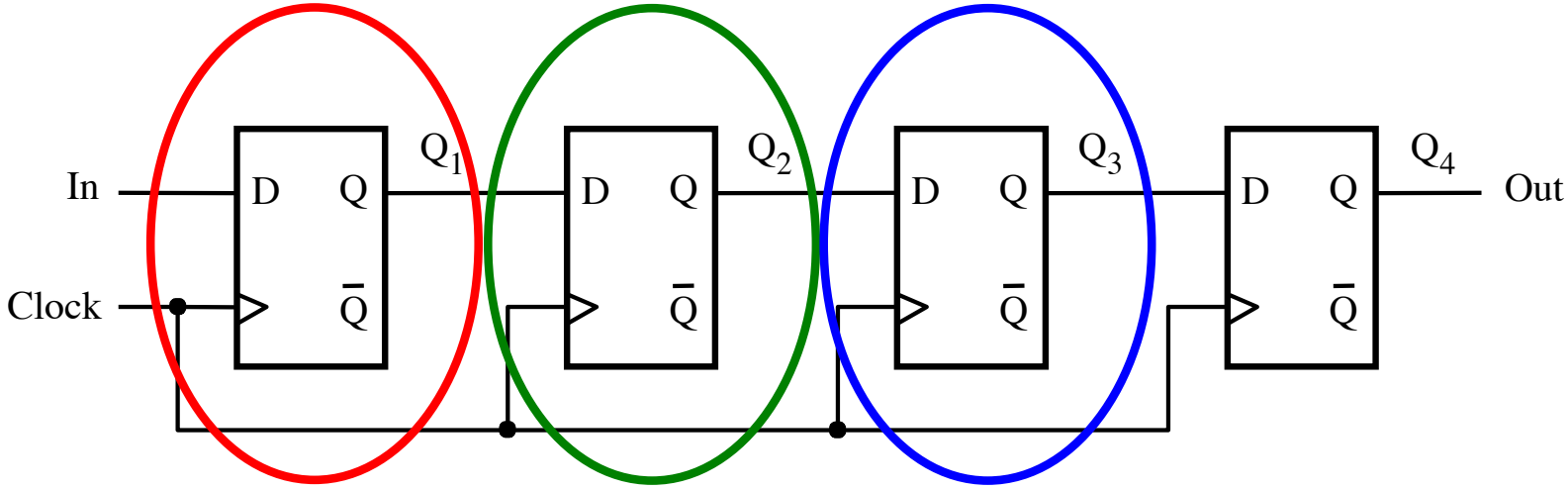


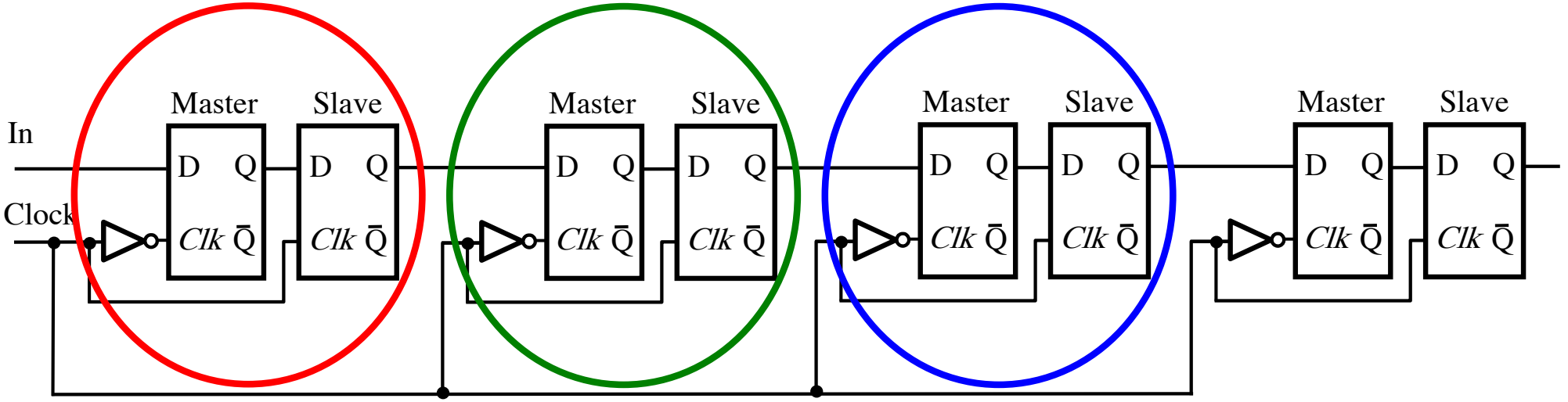
A simple shift register



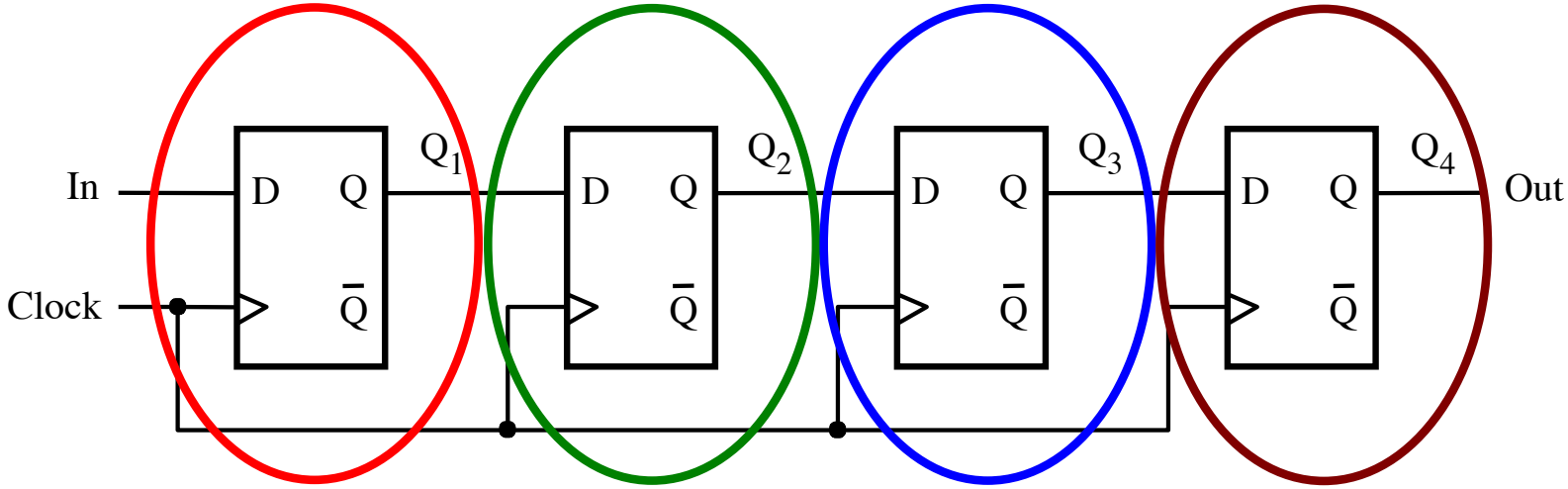


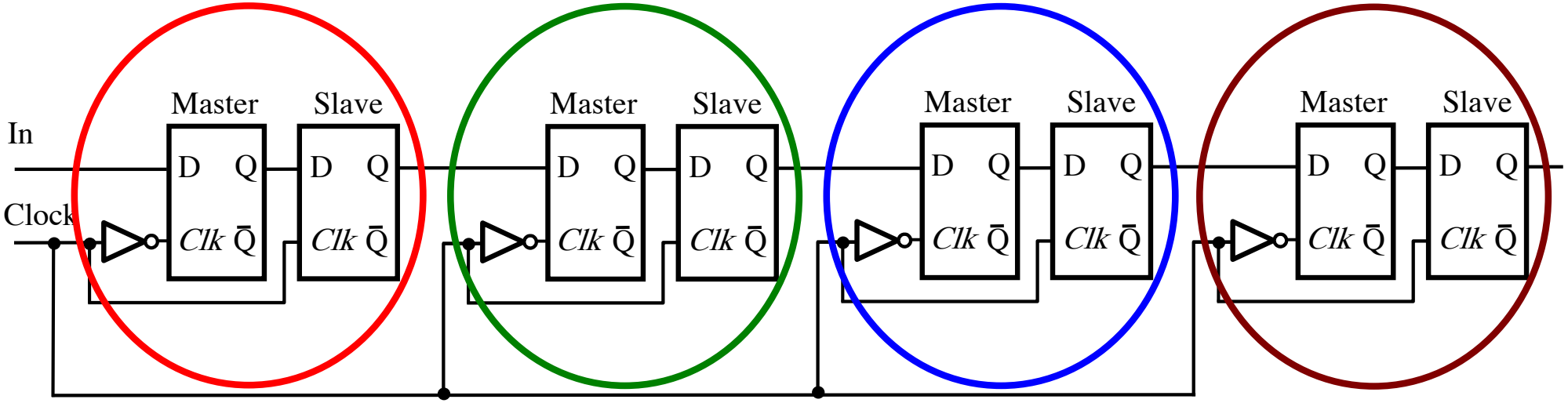
A simple shift register



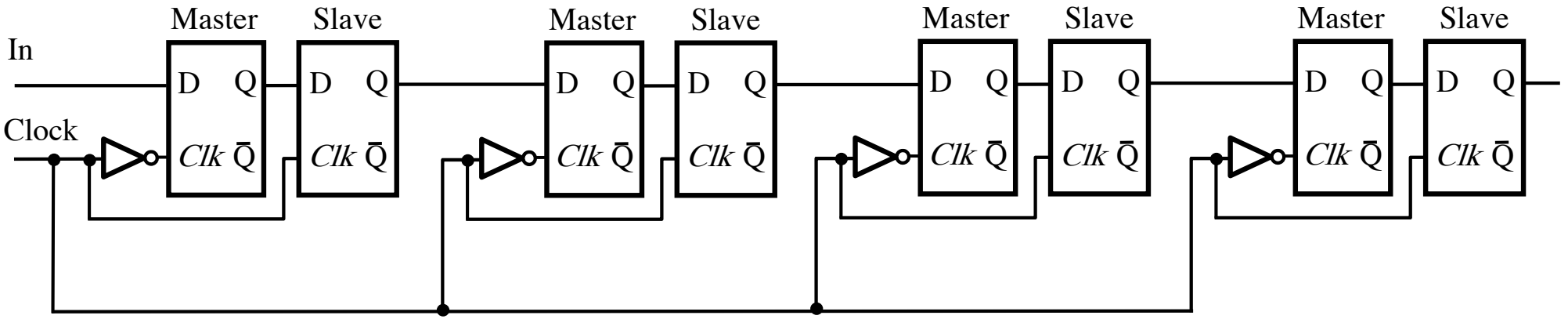


A simple shift register

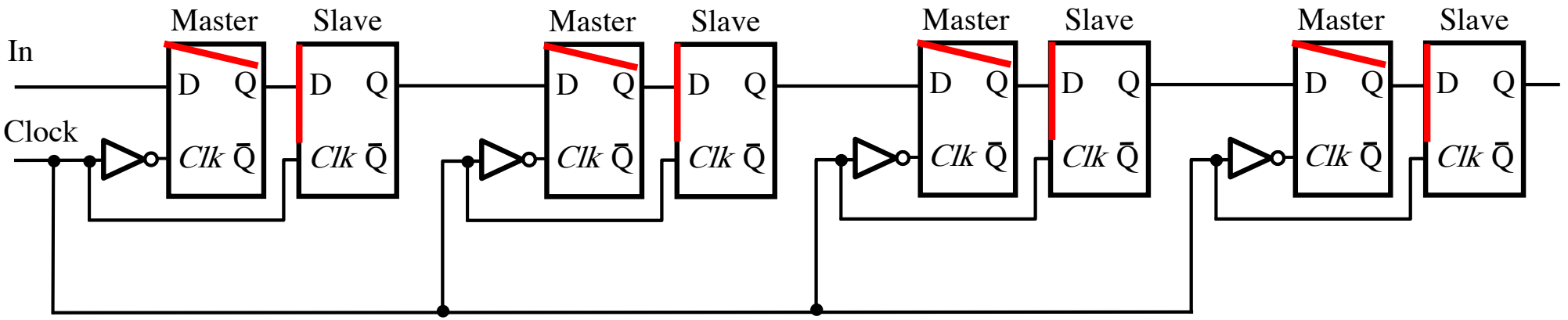




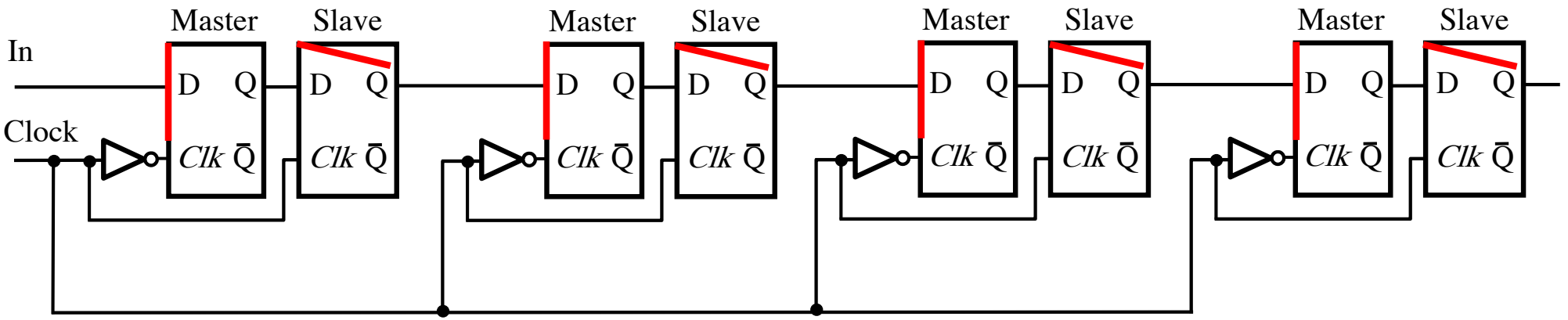
A simple shift register



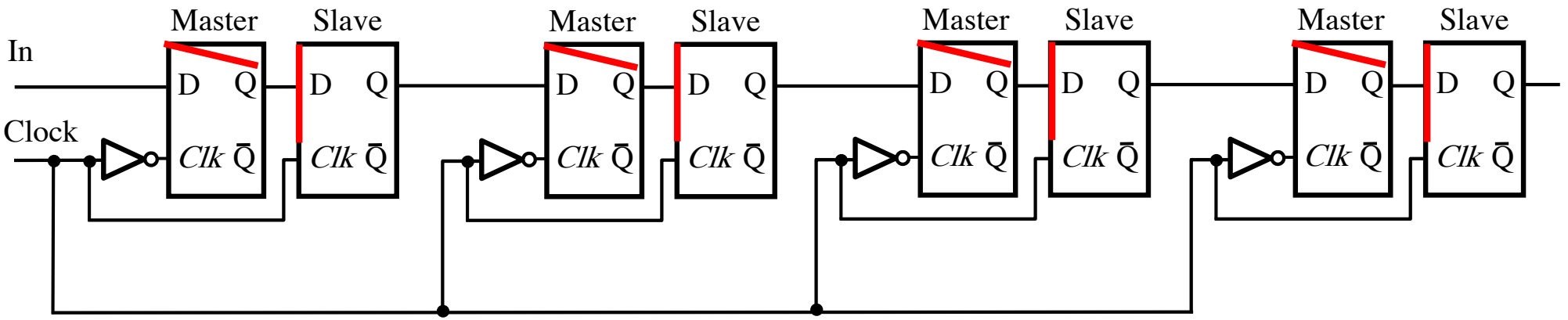
A simple shift register



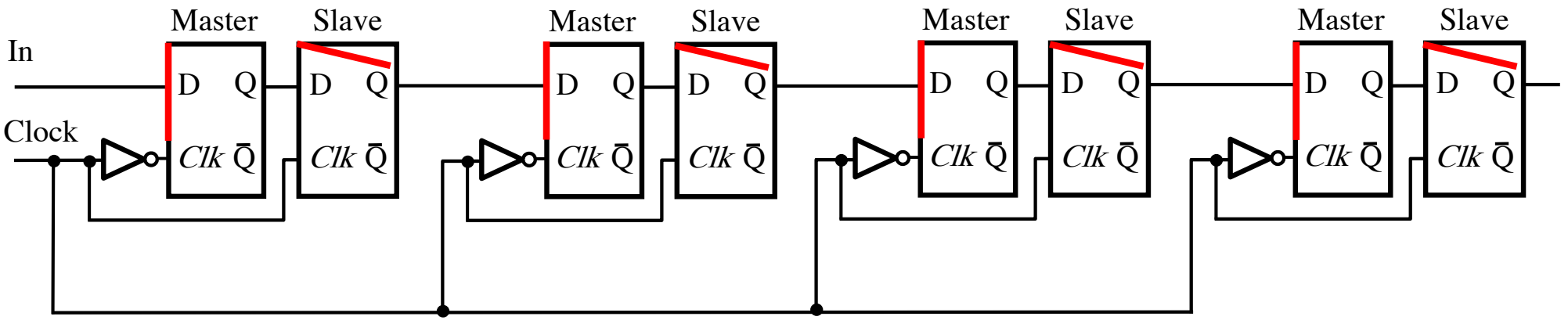
A simple shift register



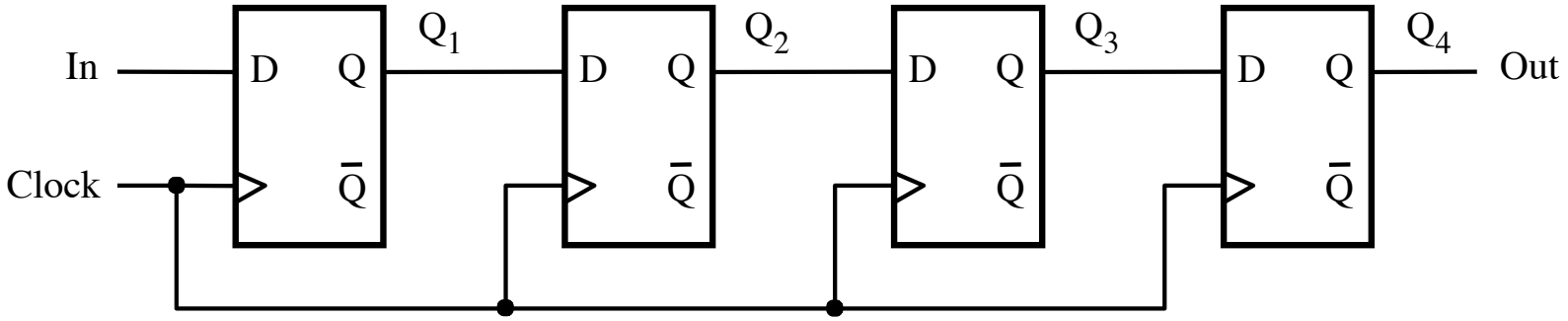
A simple shift register



A simple shift register



A simple shift register



(a) Circuit

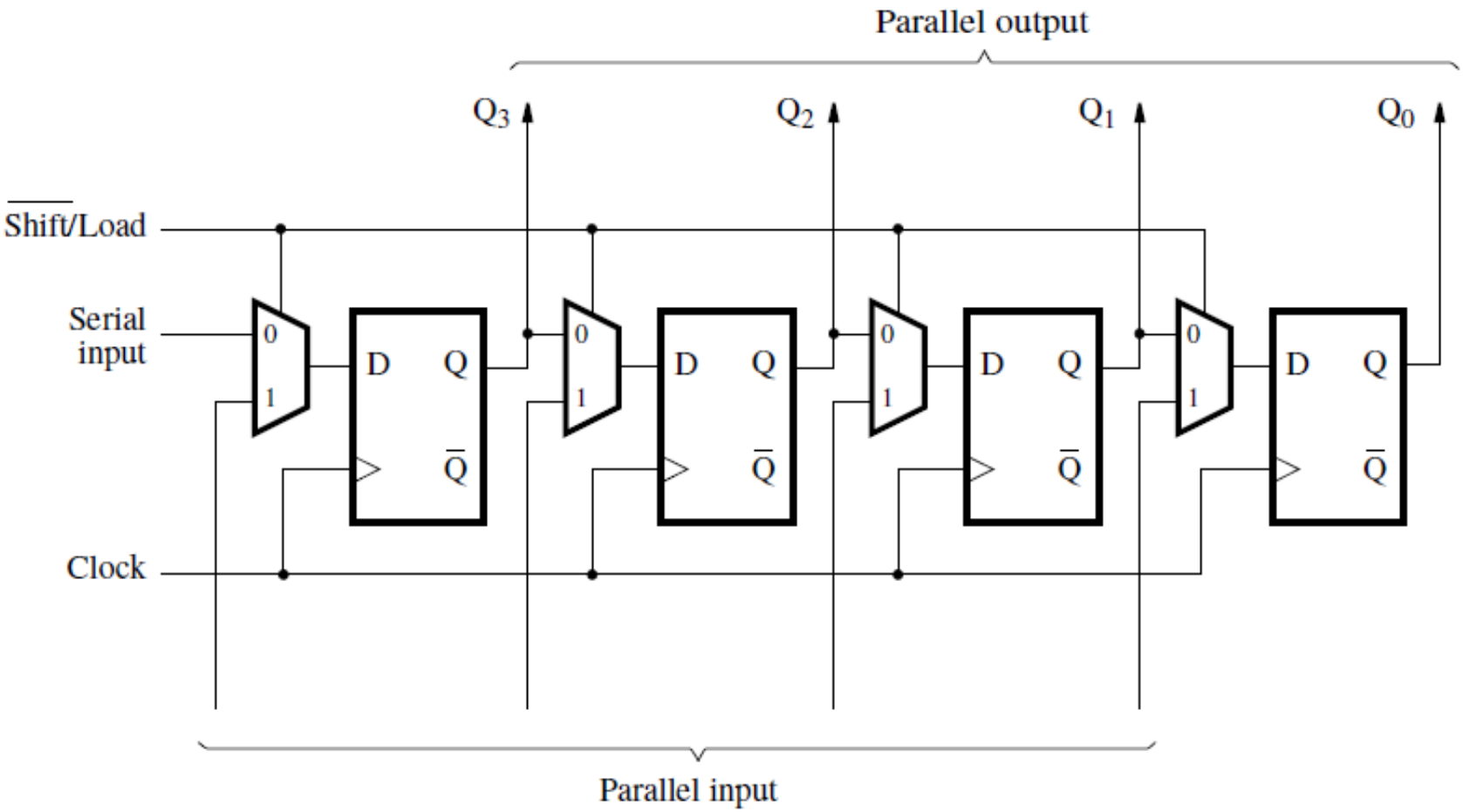
	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

(b) A sample sequence

[Figure 5.17 from the textbook]

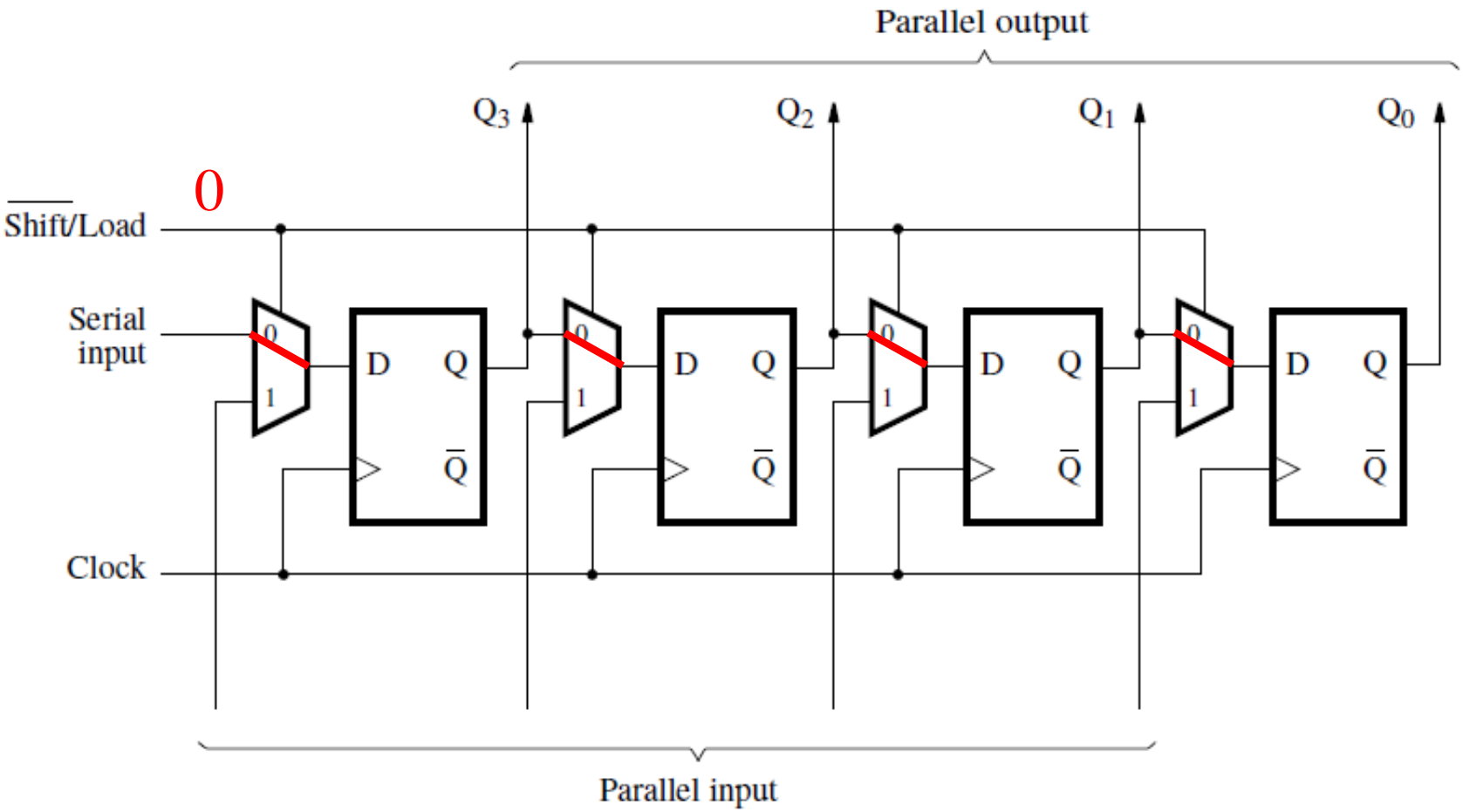
Parallel-Access Shift Register

Parallel-access shift register



[Figure 5.18 from the textbook]

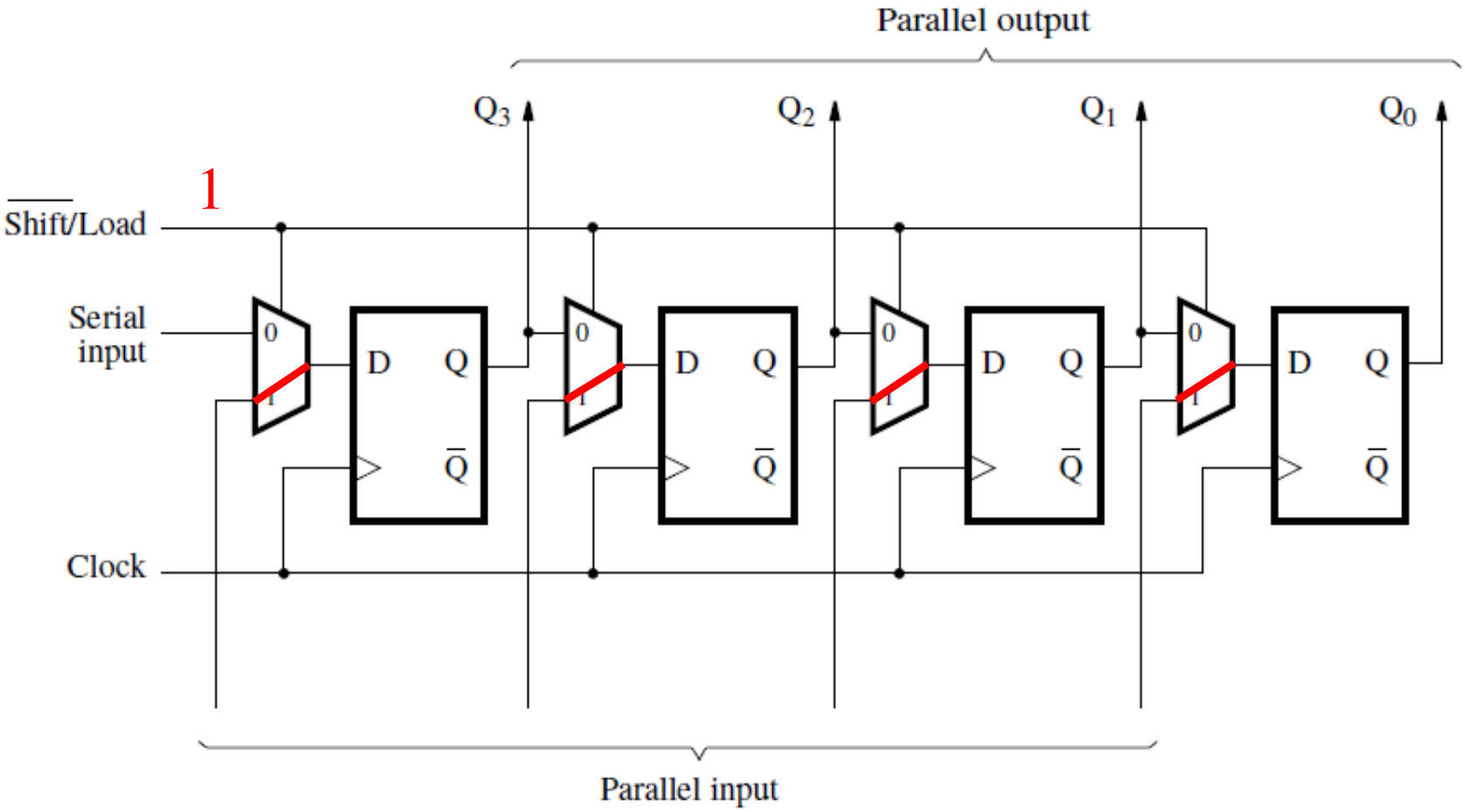
Parallel-access shift register



When Load=0, this behaves like a shift register.

[Figure 5.18 from the textbook]

Parallel-access shift register

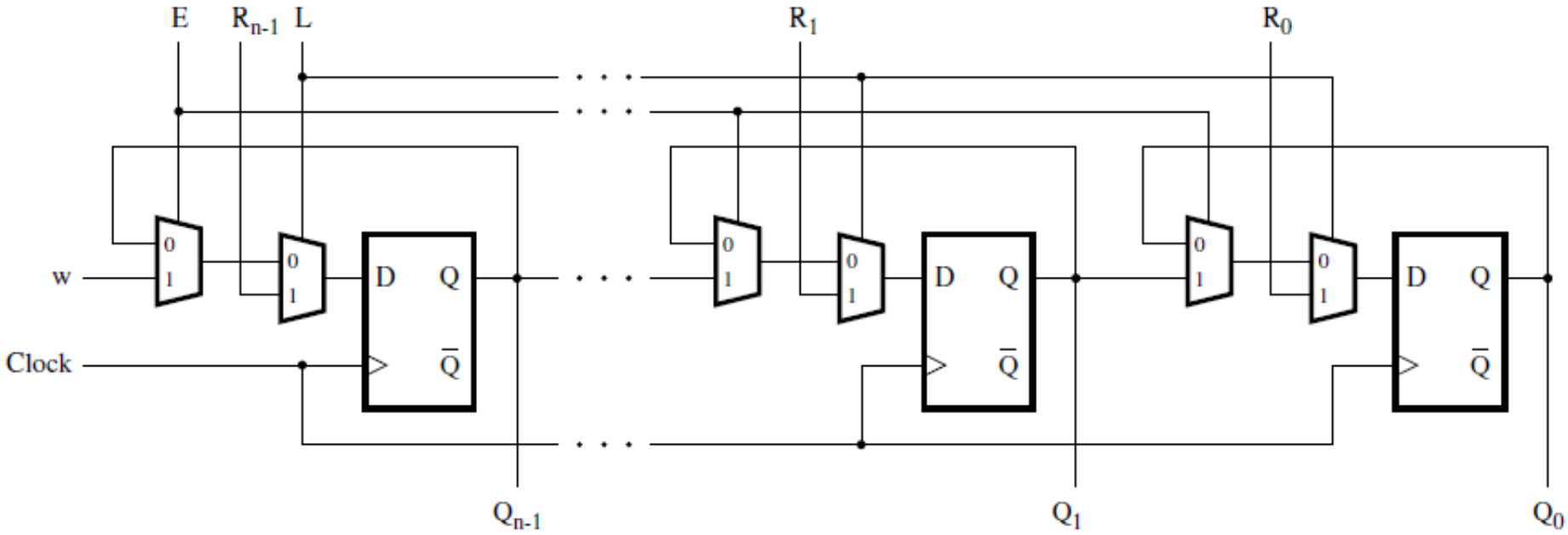


When Load=1, this behaves like a parallel-access register.

[Figure 5.18 from the textbook]

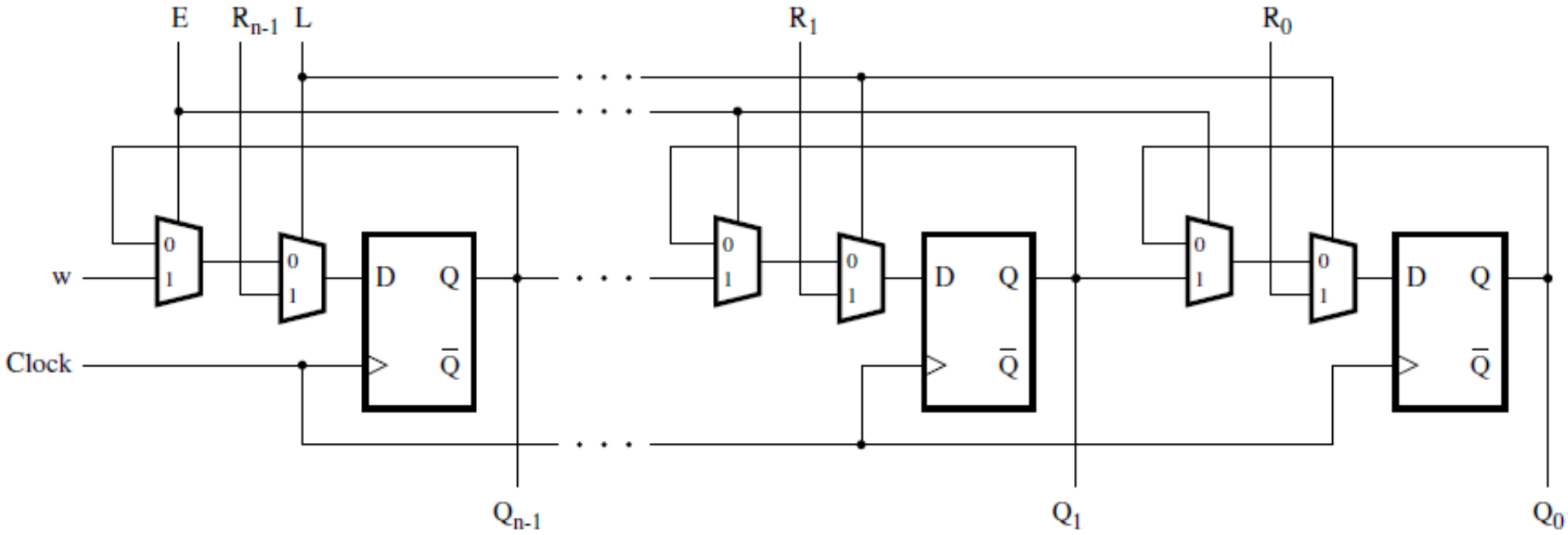
Shift Register With Parallel Load and Enable

A shift register with parallel load and enable control inputs



[Figure 5.59 from the textbook]

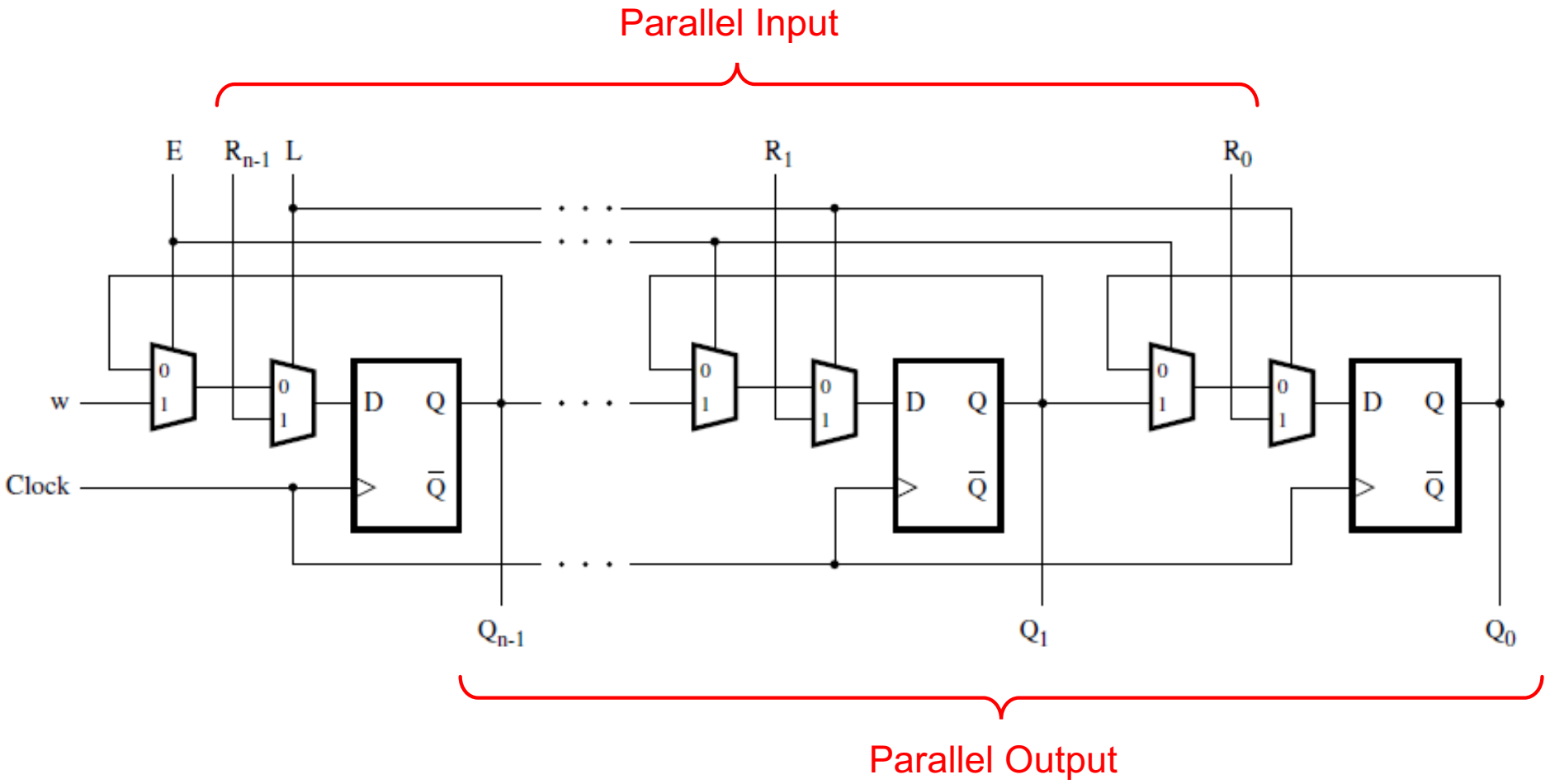
A shift register with parallel load and enable control inputs



The directions of the input and output lines are switched relative to the previous slides.

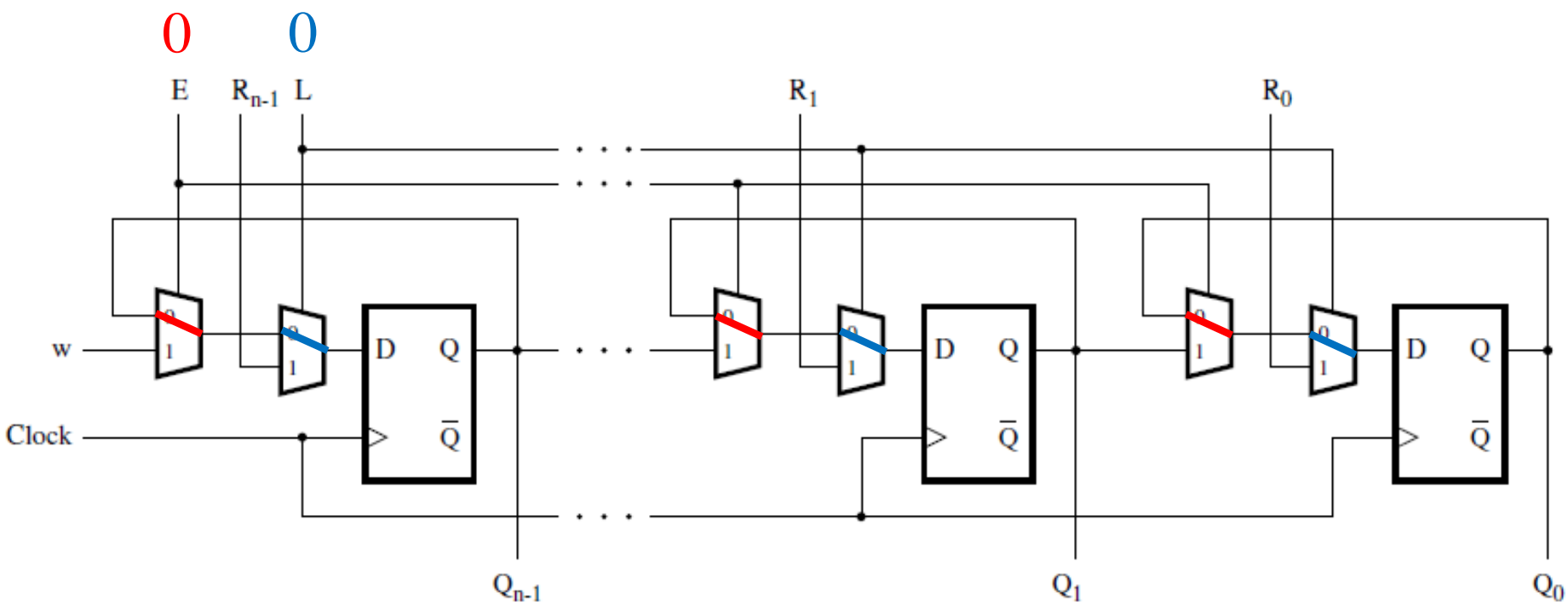
[Figure 5.59 from the textbook]

A shift register with parallel load and enable control inputs



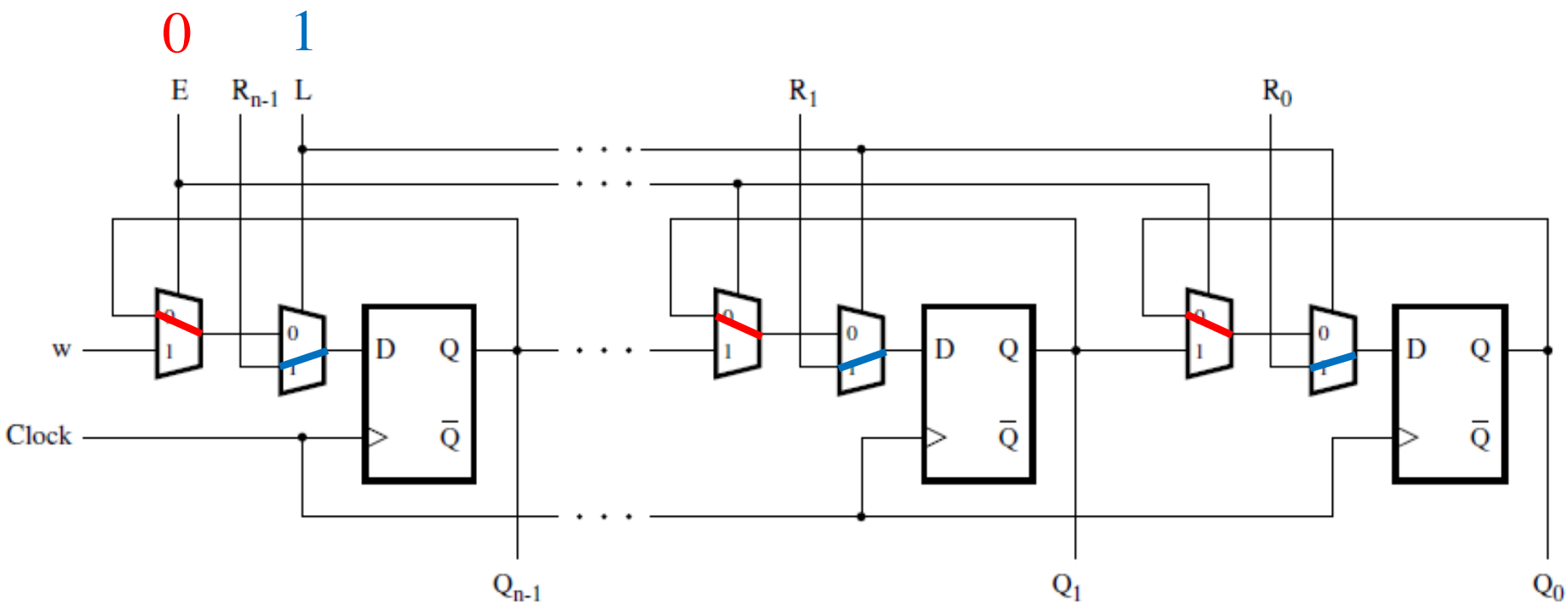
[Figure 5.59 from the textbook]

A shift register with parallel load and enable control inputs



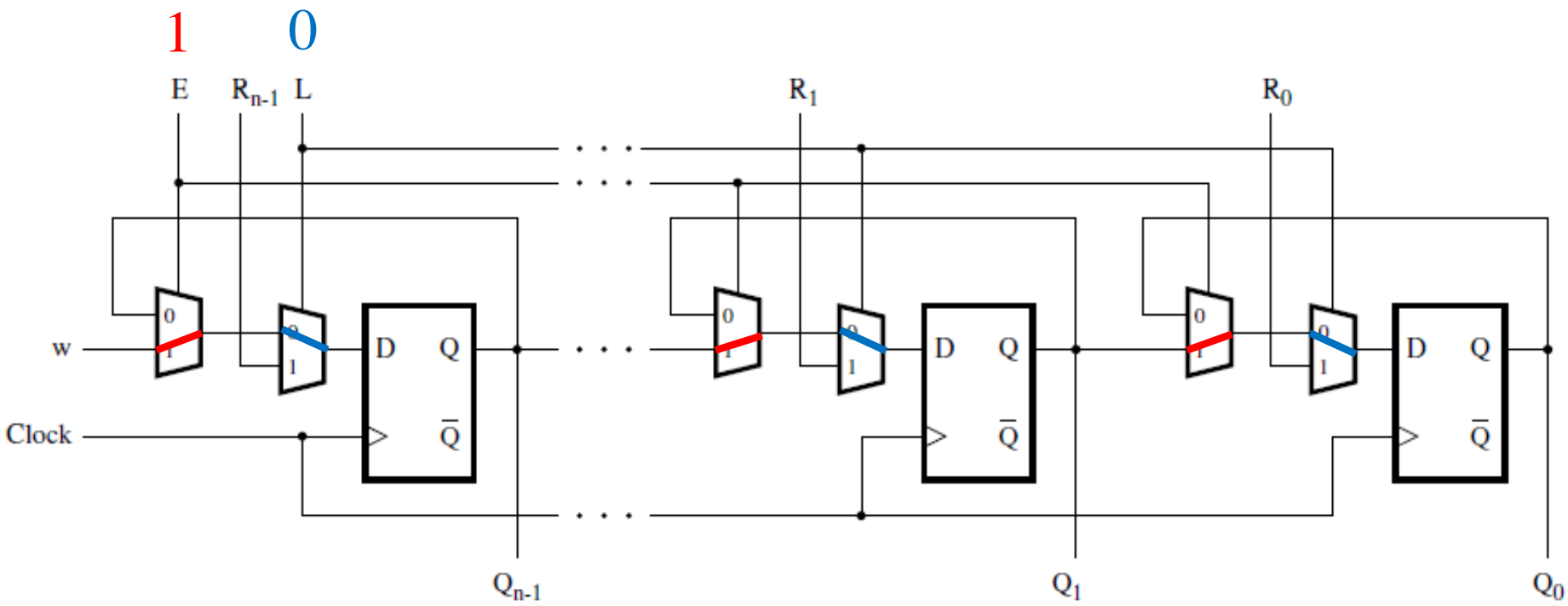
[Figure 5.59 from the textbook]

A shift register with parallel load and enable control inputs



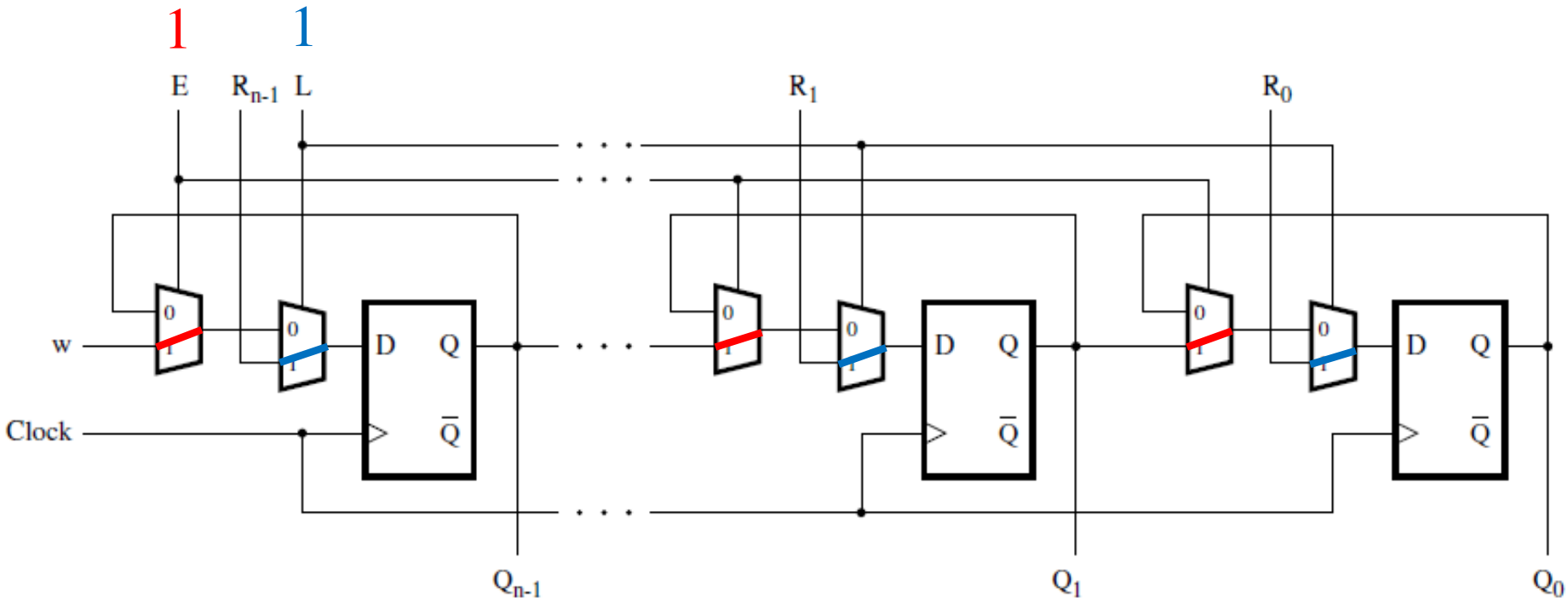
[Figure 5.59 from the textbook]

A shift register with parallel load and enable control inputs



[Figure 5.59 from the textbook]

A shift register with parallel load and enable control inputs



[Figure 5.59 from the textbook]

Parallel-access shift left / right register

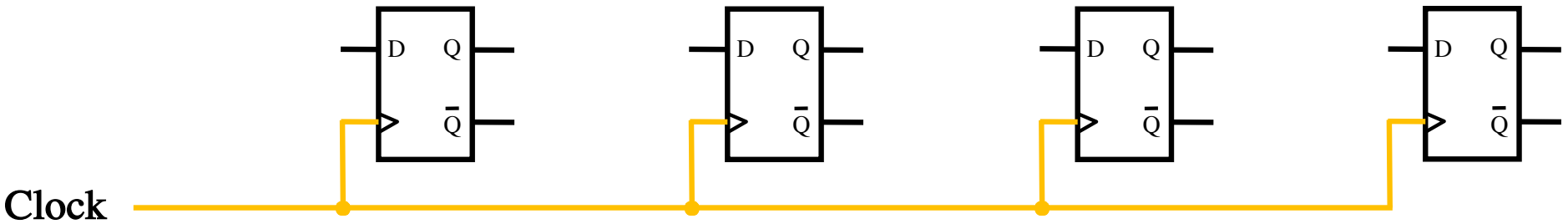
Parallel-access shift left/right register

Complete the following circuit diagram to implement a 4-bit register that has both parallel load and shift left/right functionality. The register has two control inputs (C1 and C0), four parallel input lines (I3, I2, I1, and I0), and four output lines (Q3, Q2, Q1, and Q0). Depending on the values of C1 and C0, the register performs one of the following four operations:

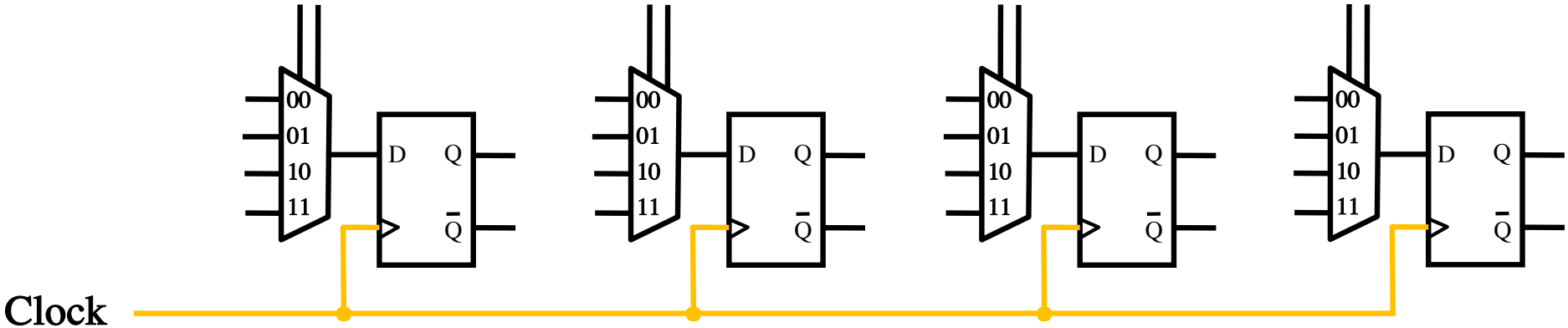
C_1	C_0	Operation
0	0	Hold the current value (i.e., $Q_3\ Q_2\ Q_1\ Q_0$ are not changed)
0	1	Shift left (i.e., new Q_3=Q_2, new Q_2=Q_1, new Q_1=Q_0, new Q_0=I_0)
1	0	Shift right (i.e., new Q_3=I_3, new Q_2=Q_3, new Q_1=Q_2, new Q_0=Q_1)
1	1	Load new data (i.e., new Q_3=I_3, new Q_2=I_2, new Q_1=I_1, new Q_0=I_0)

Parallel-access shift left/right register

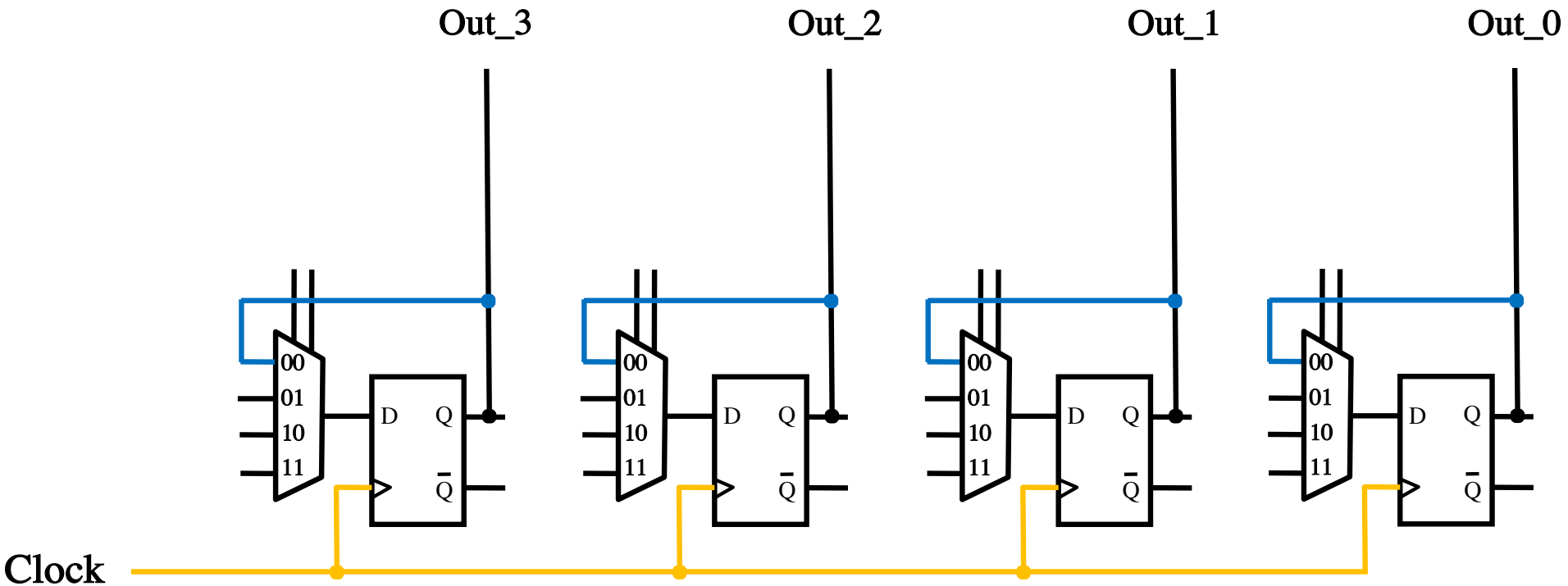
Parallel-access shift left/right register



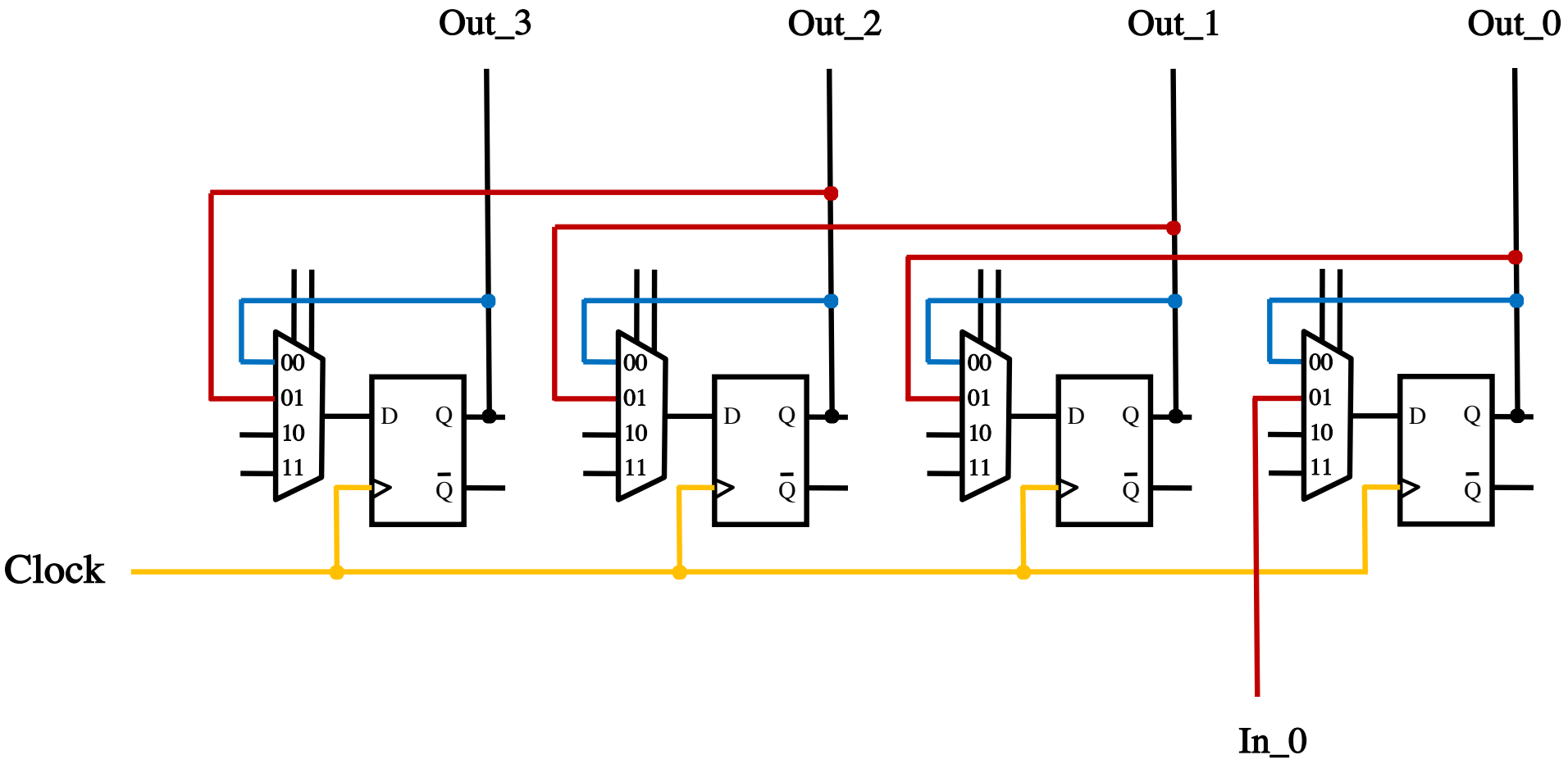
Parallel-access shift left/right register



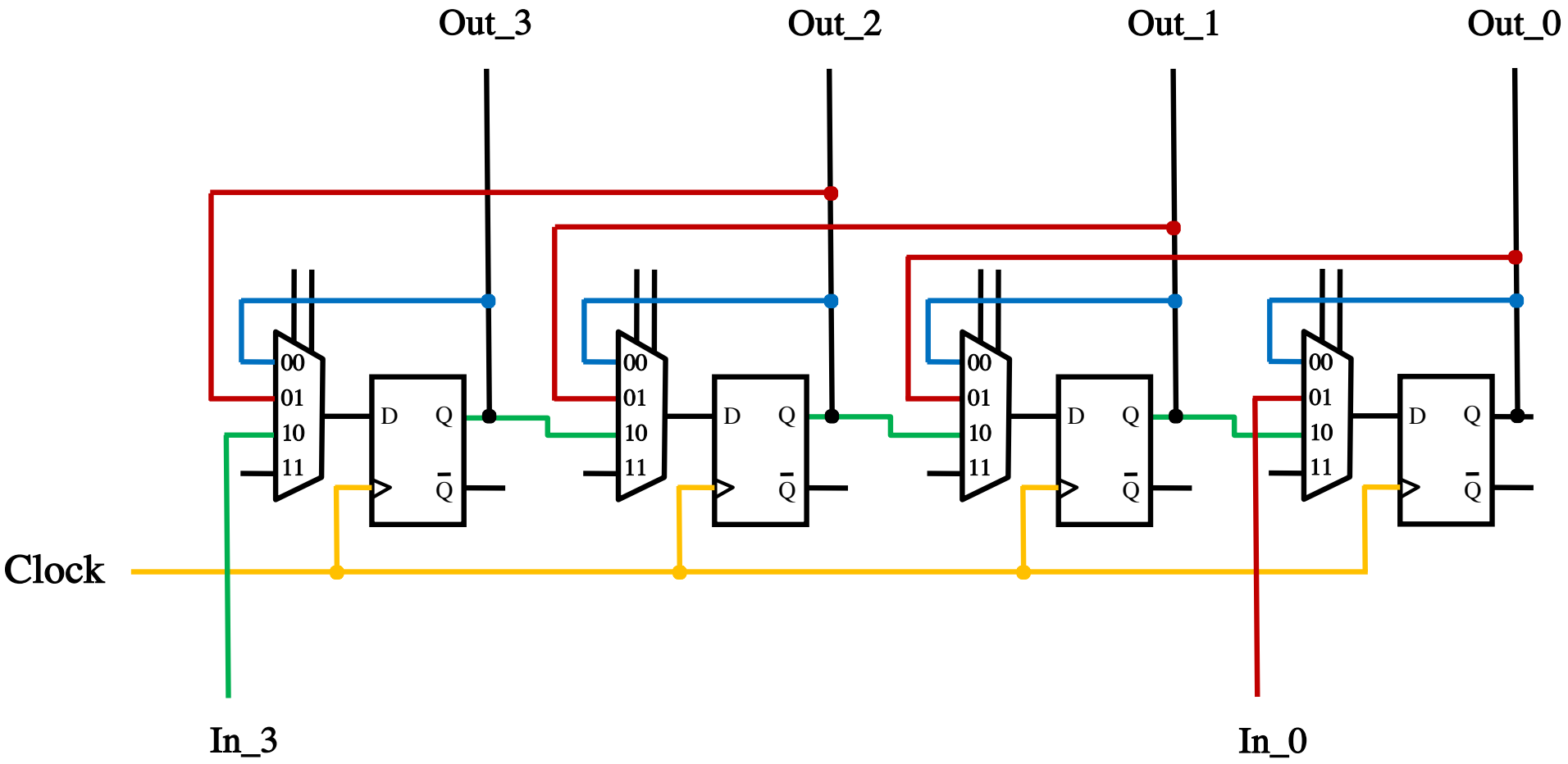
Parallel-access shift left/right register



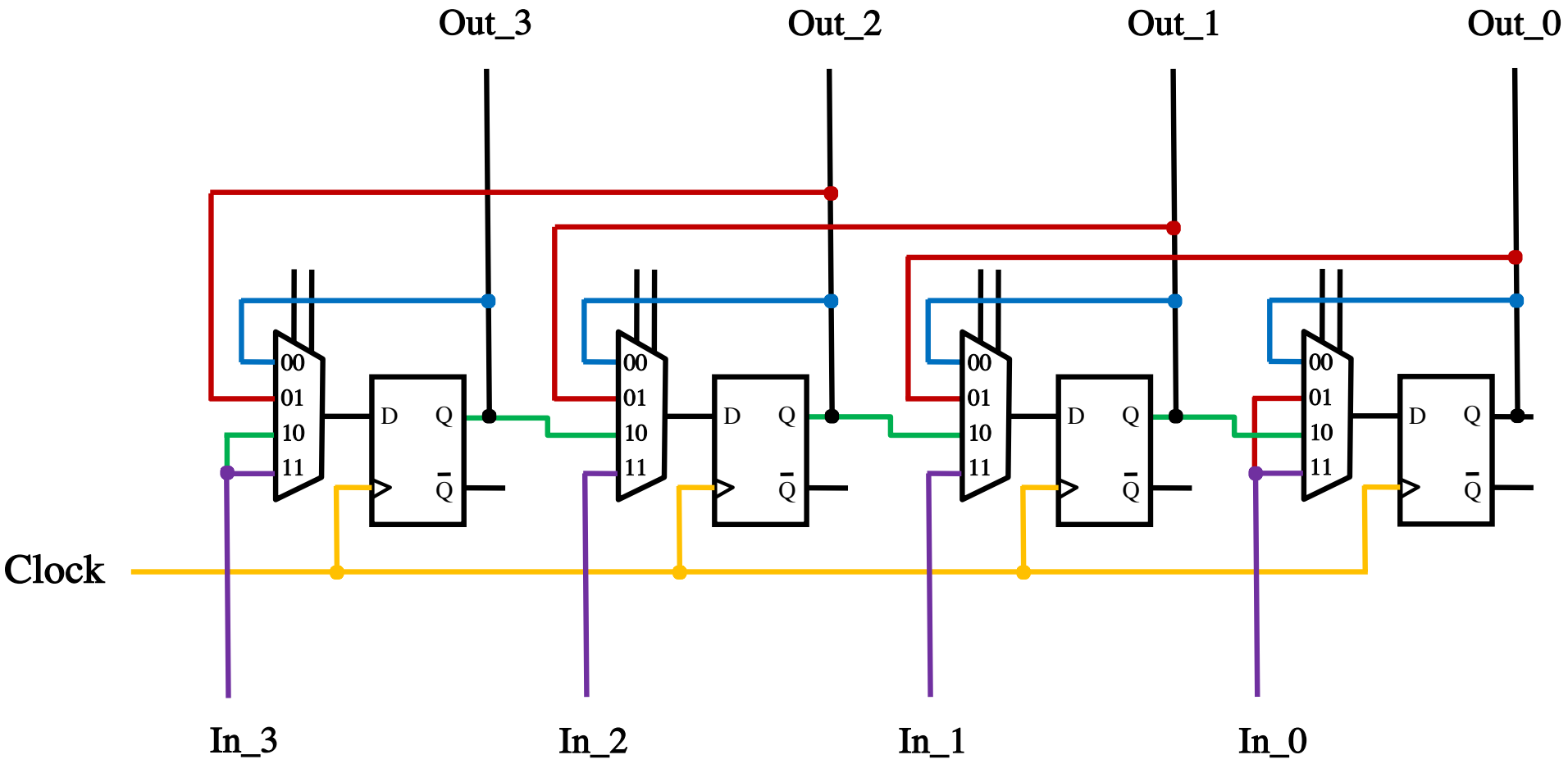
Parallel-access shift left/right register



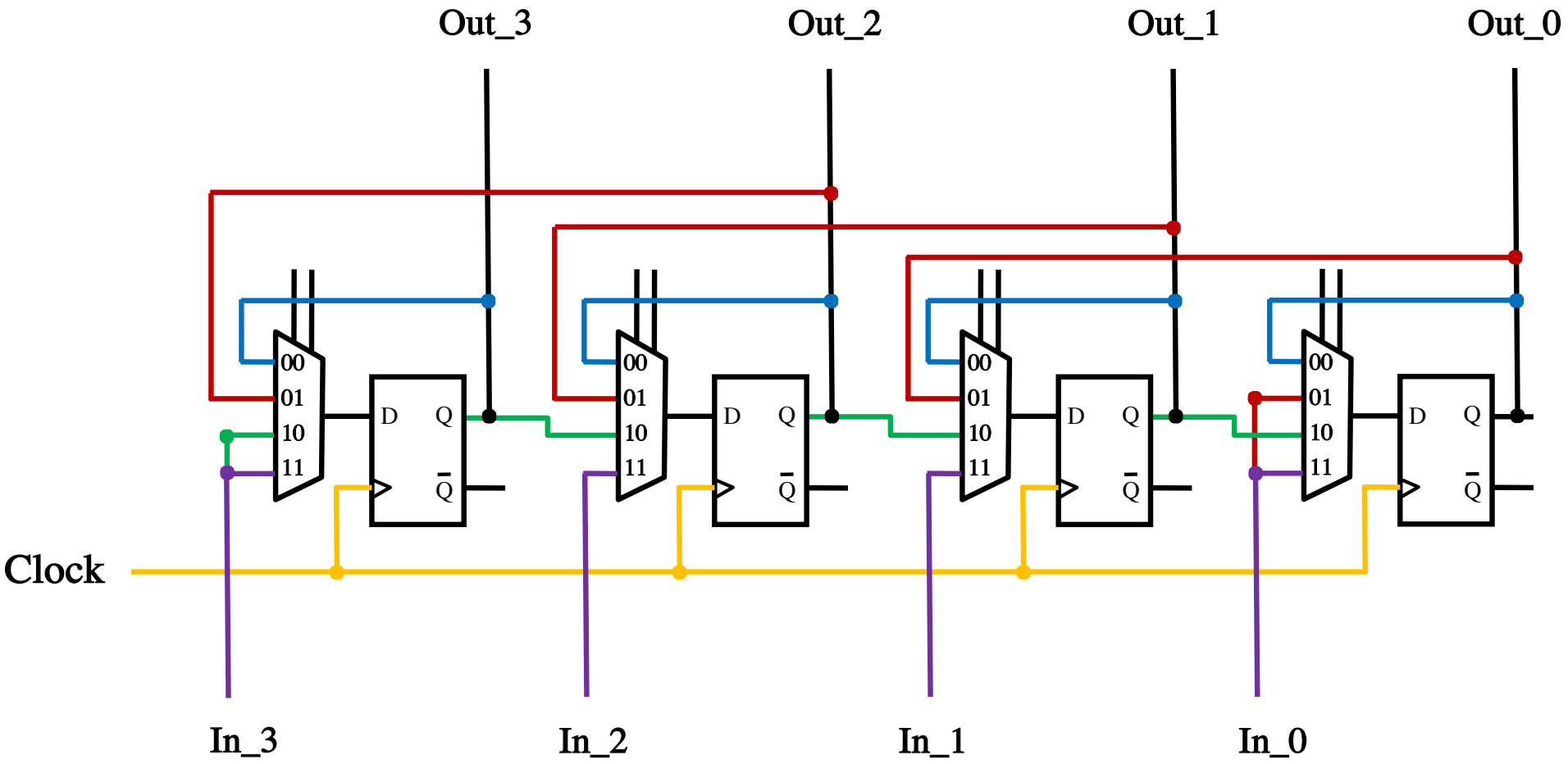
Parallel-access shift left/right register



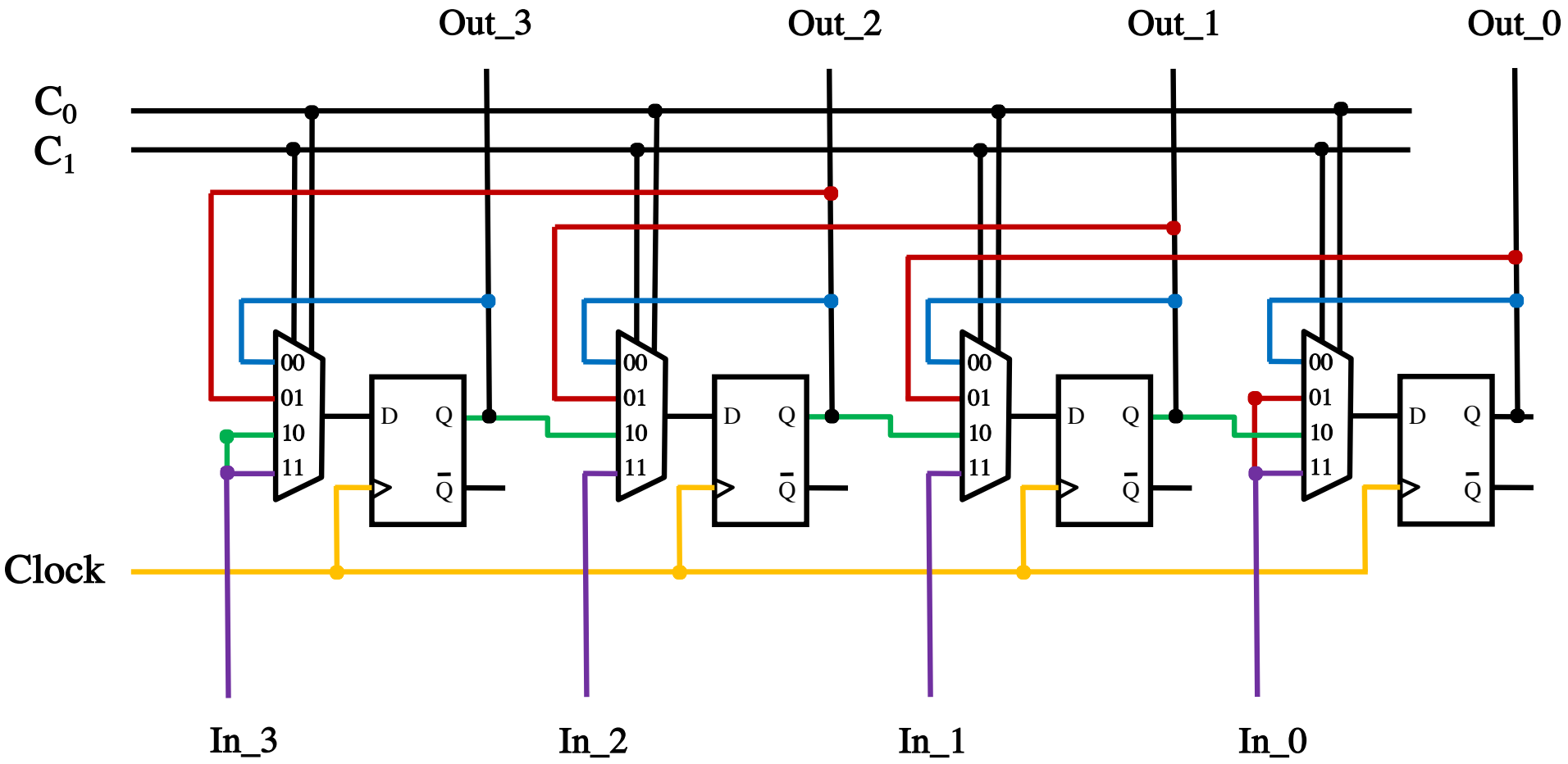
Parallel-access shift left/right register



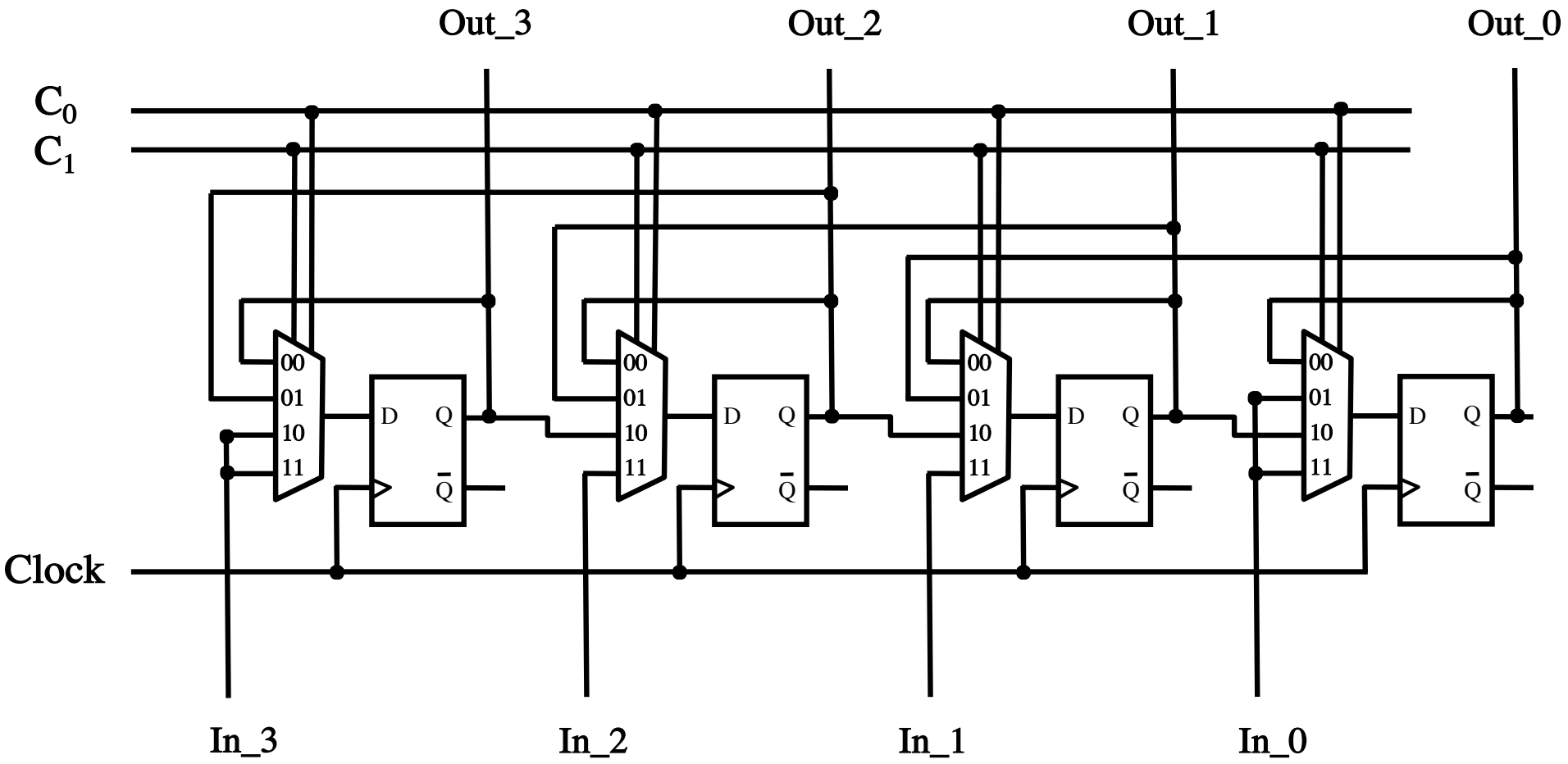
Parallel-access shift left/right register



Parallel-access shift left/right register

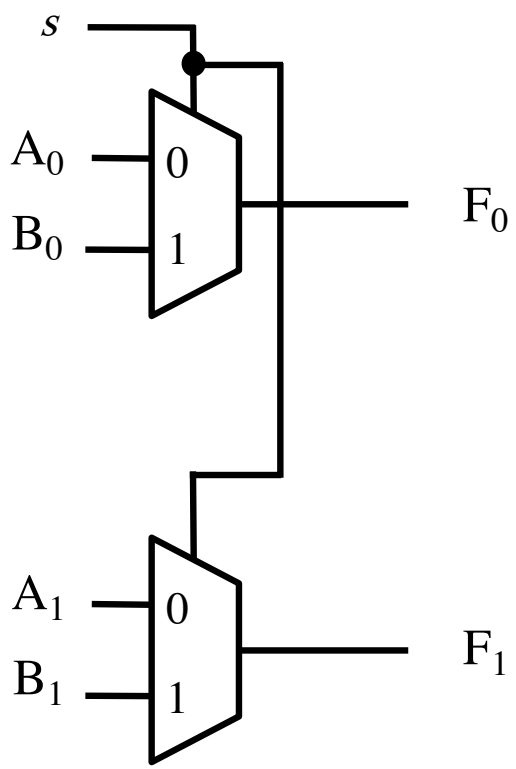


Parallel-access shift left/right register

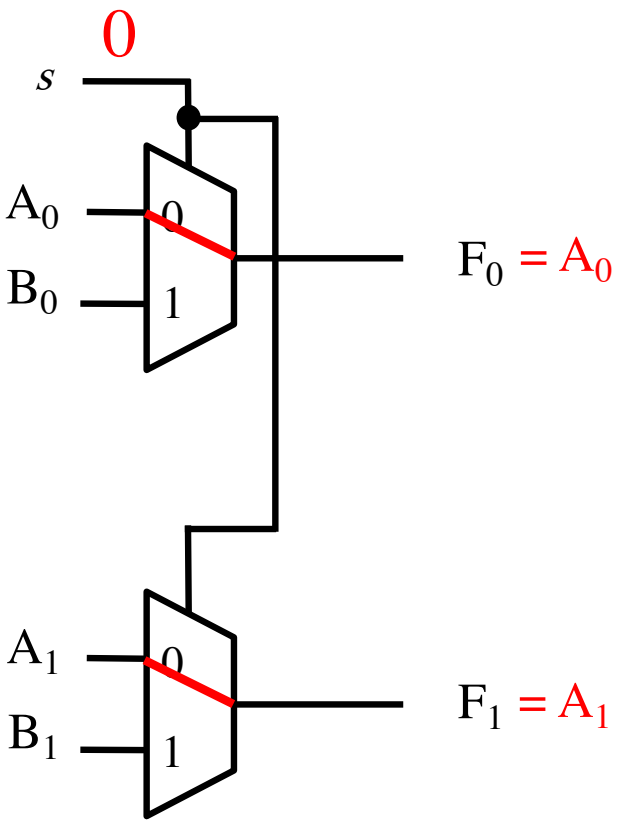


Multiplexer Tricks
(select one of two 2-bit numbers)

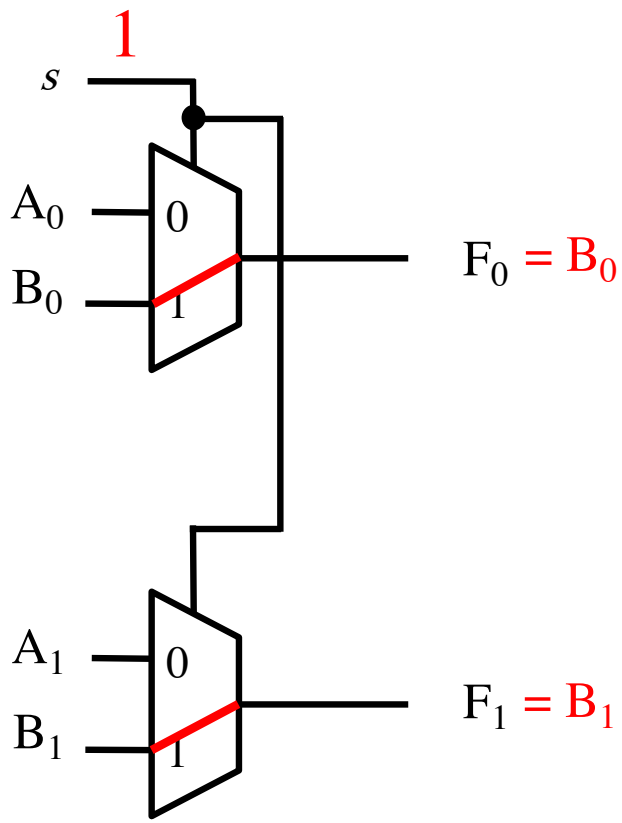
Select Either $A=A_1A_0$ or $B=B_1B_0$



Select Either $A=A_1A_0$ or $B=B_1B_0$

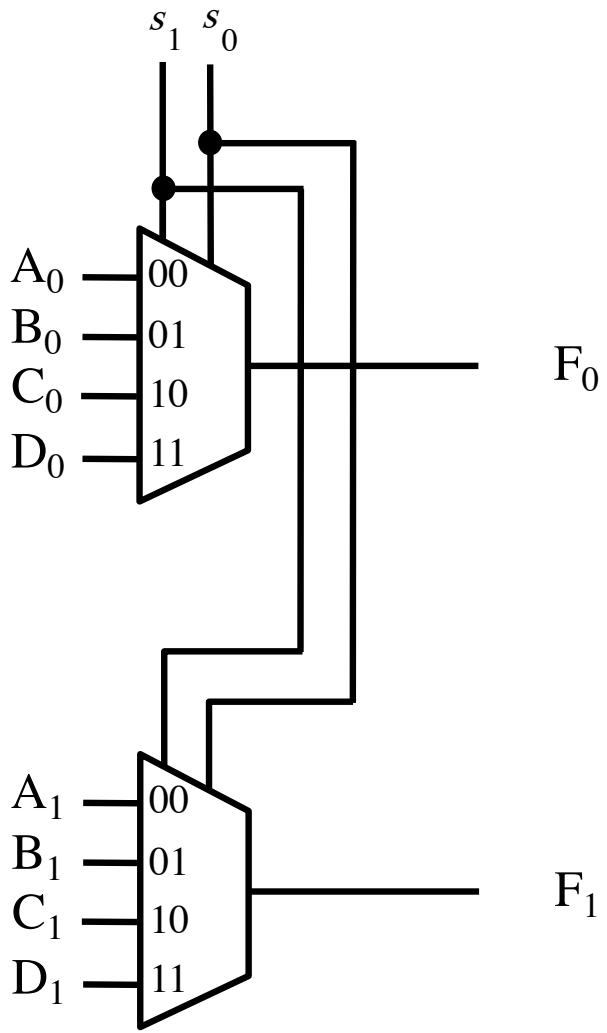


Select Either $A=A_1A_0$ or $B=B_1B_0$

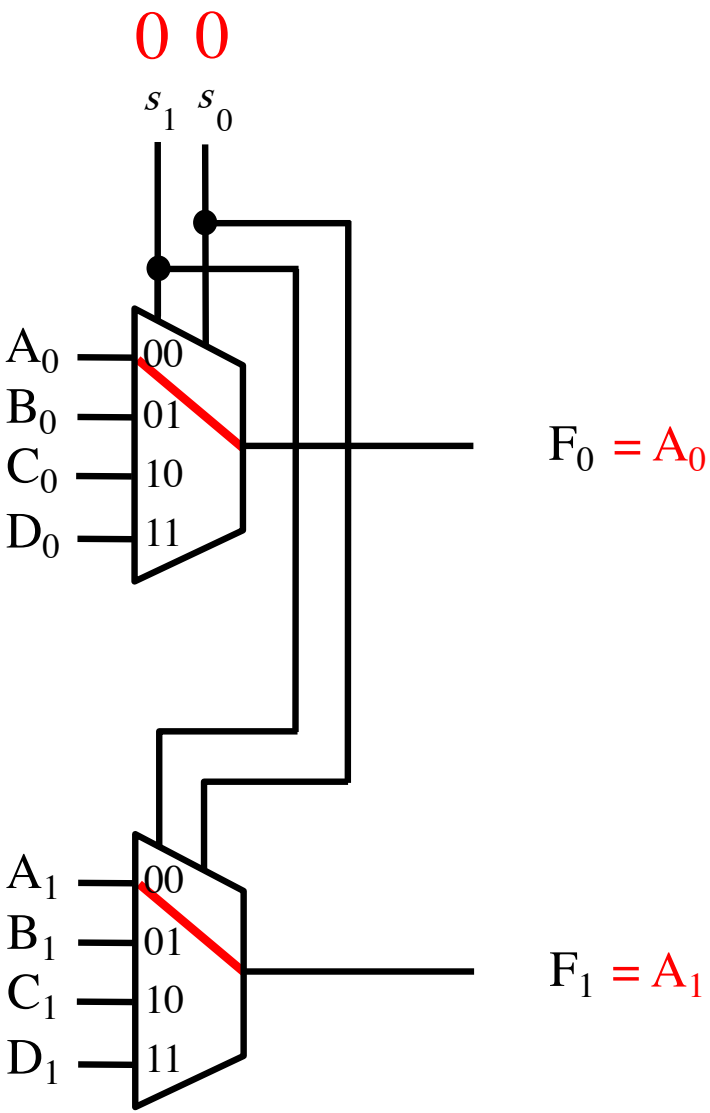


Multiplexer Tricks
(select one of four 2-bit numbers)

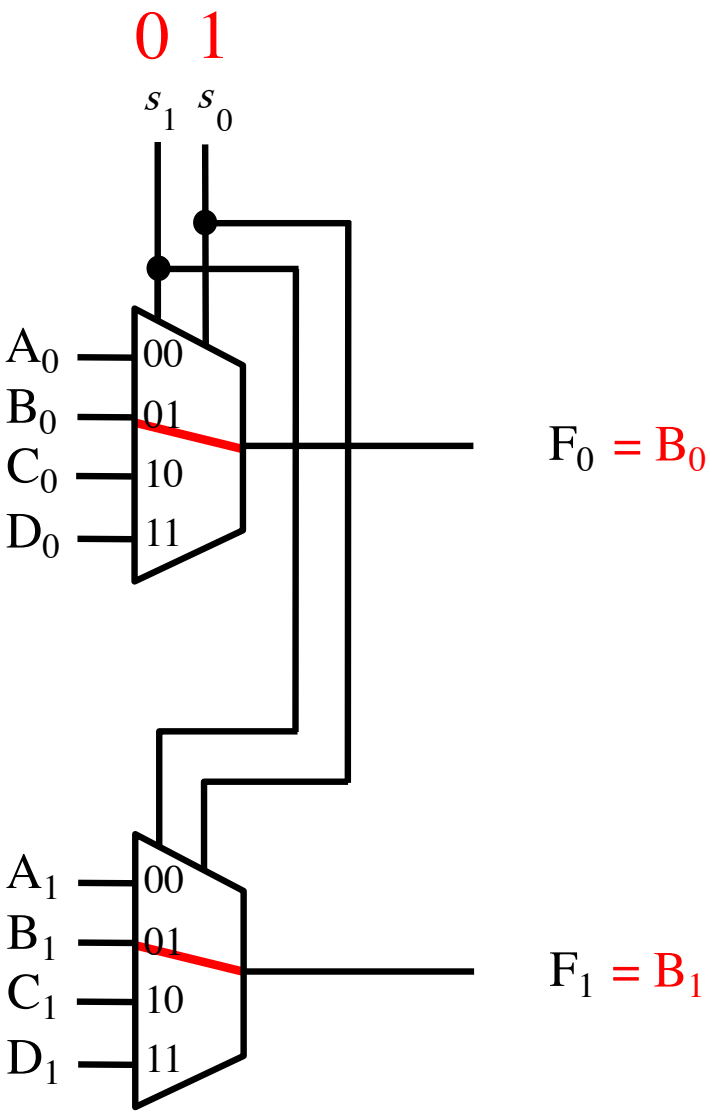
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



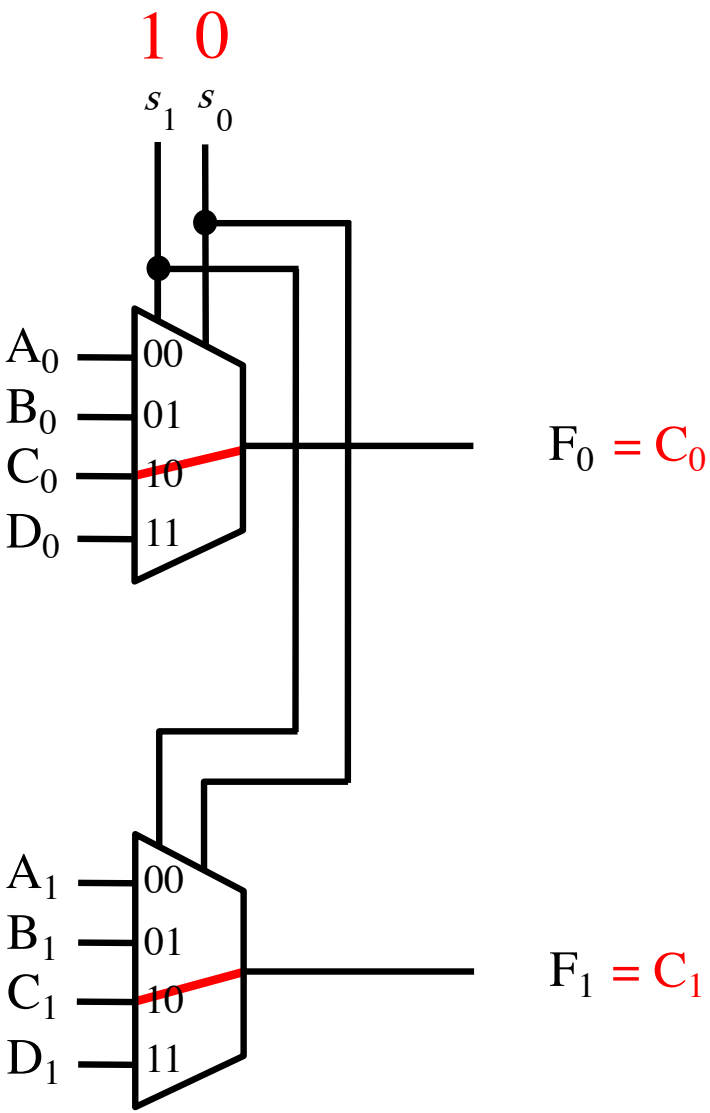
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



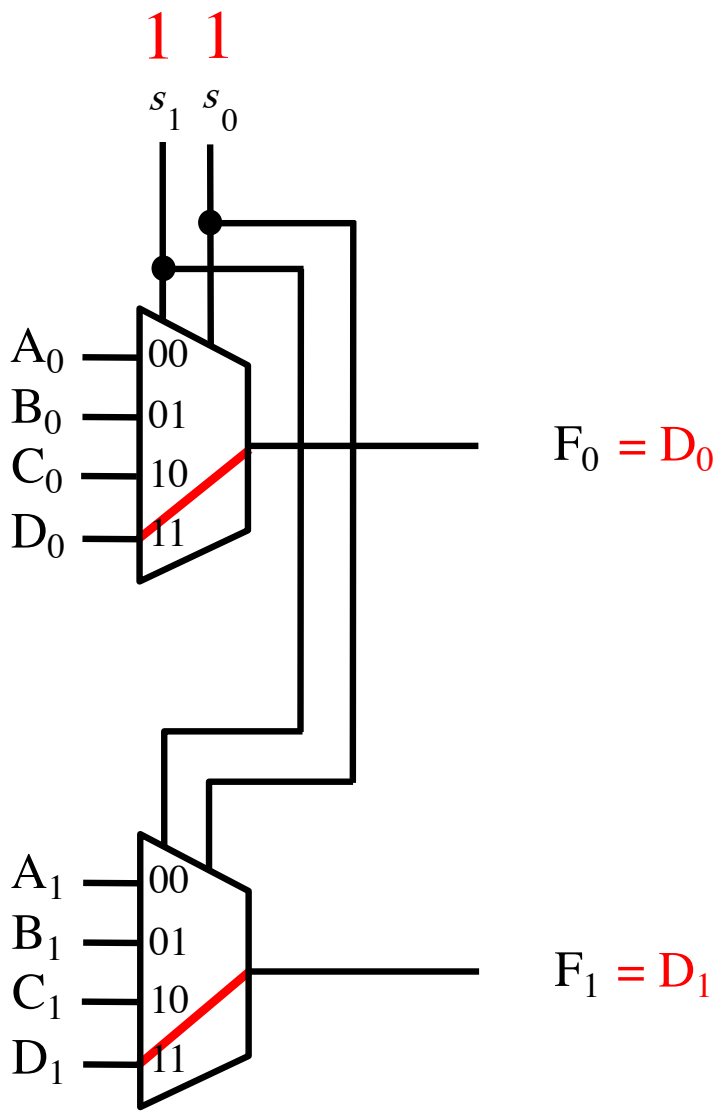
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$

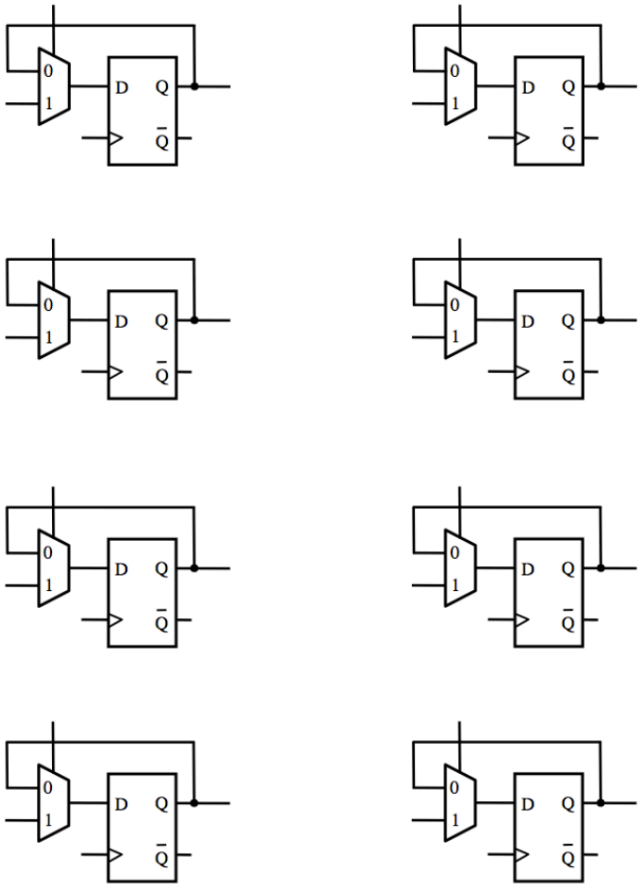


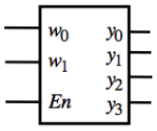
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$

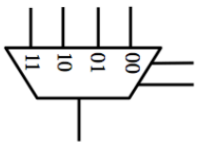


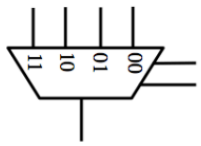
Register File

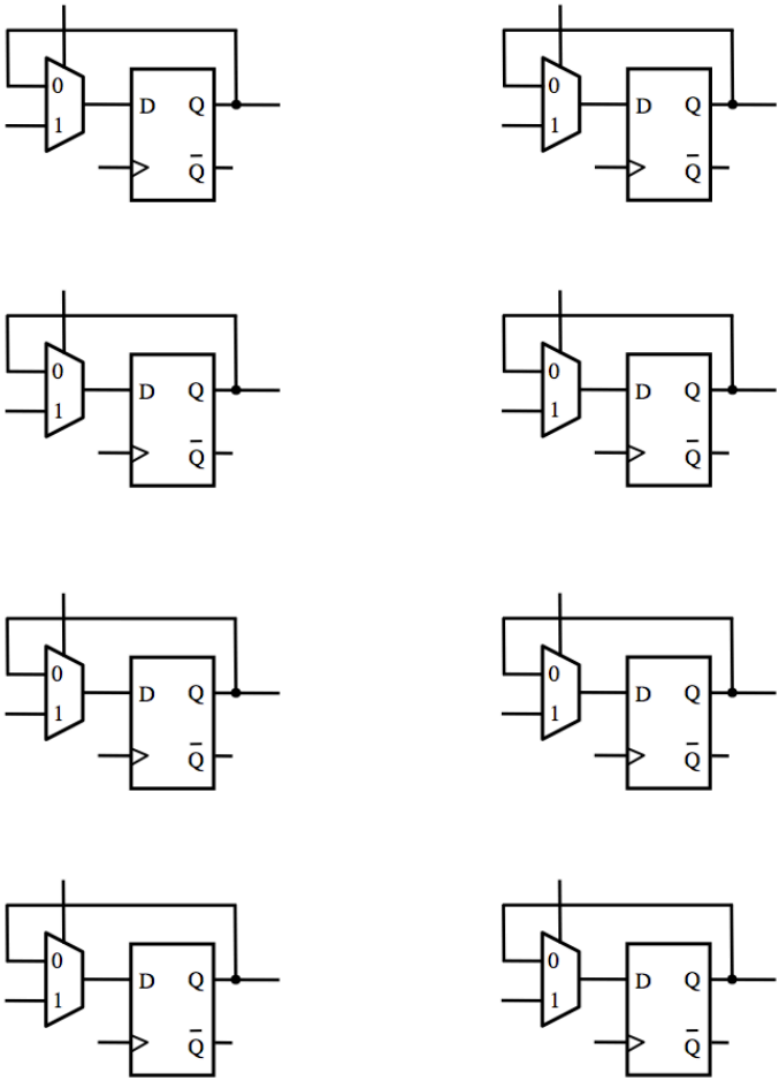
Complete the following circuit diagram to implement a register file with four 2-bit registers, one write port, one read port, and one write enable line.

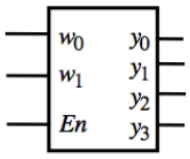


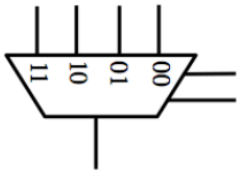


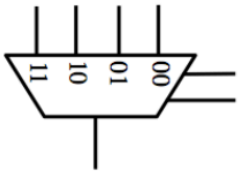


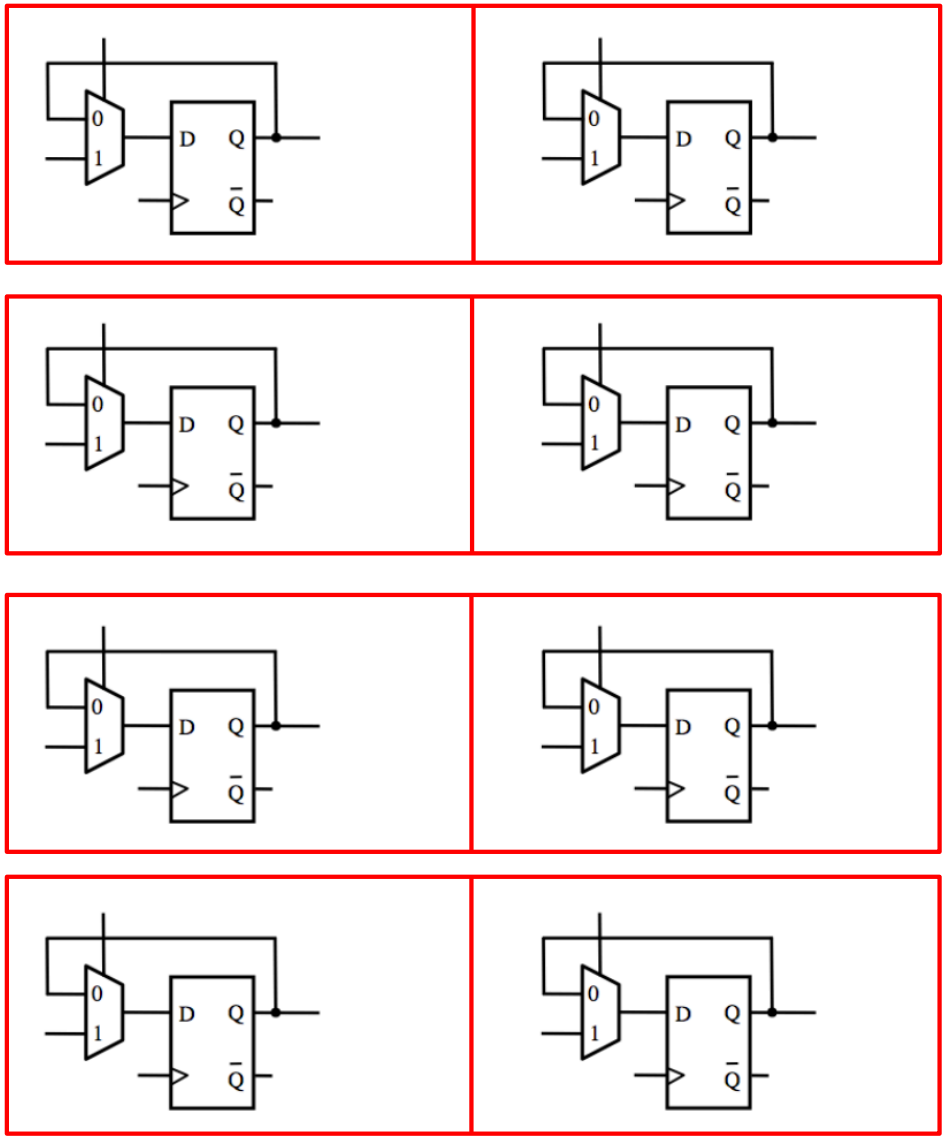


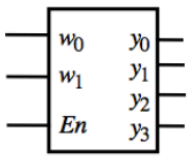

0
1
D
Q
Q

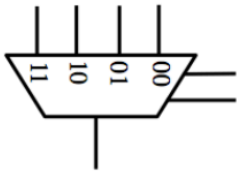

w_0
w_1
En
y_0
y_1
y_2
y_3

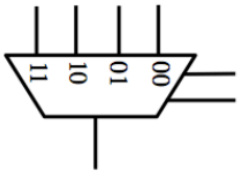

11
10
01
00

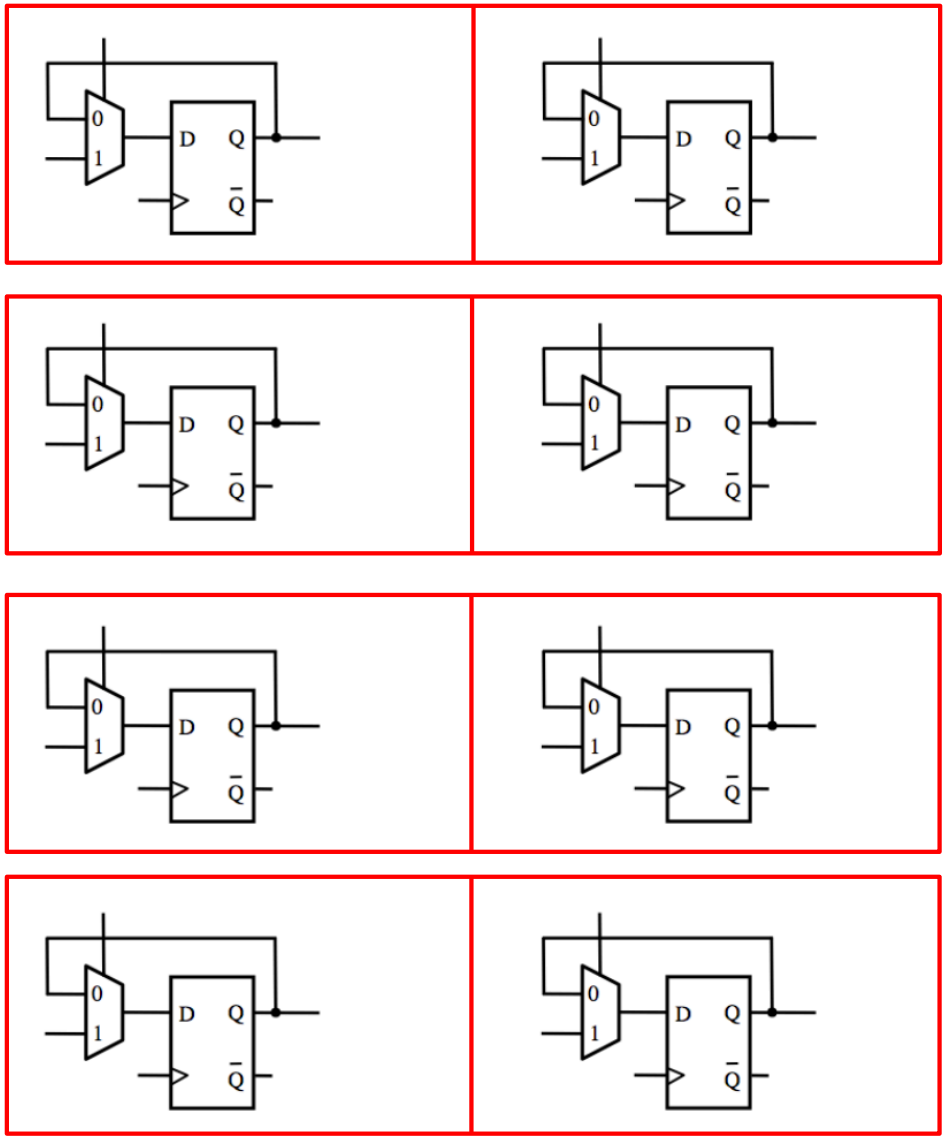

11
10
01
00

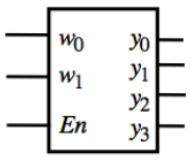

Register 0
Register 1
Register 2
Register 3
D
Q
$\bar{Q}$
0
1

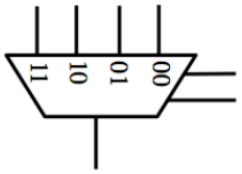

w_0
w_1
En
y_0
y_1
y_2
y_3

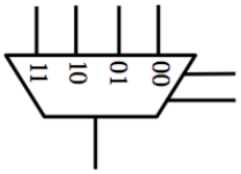

11
10
01
00

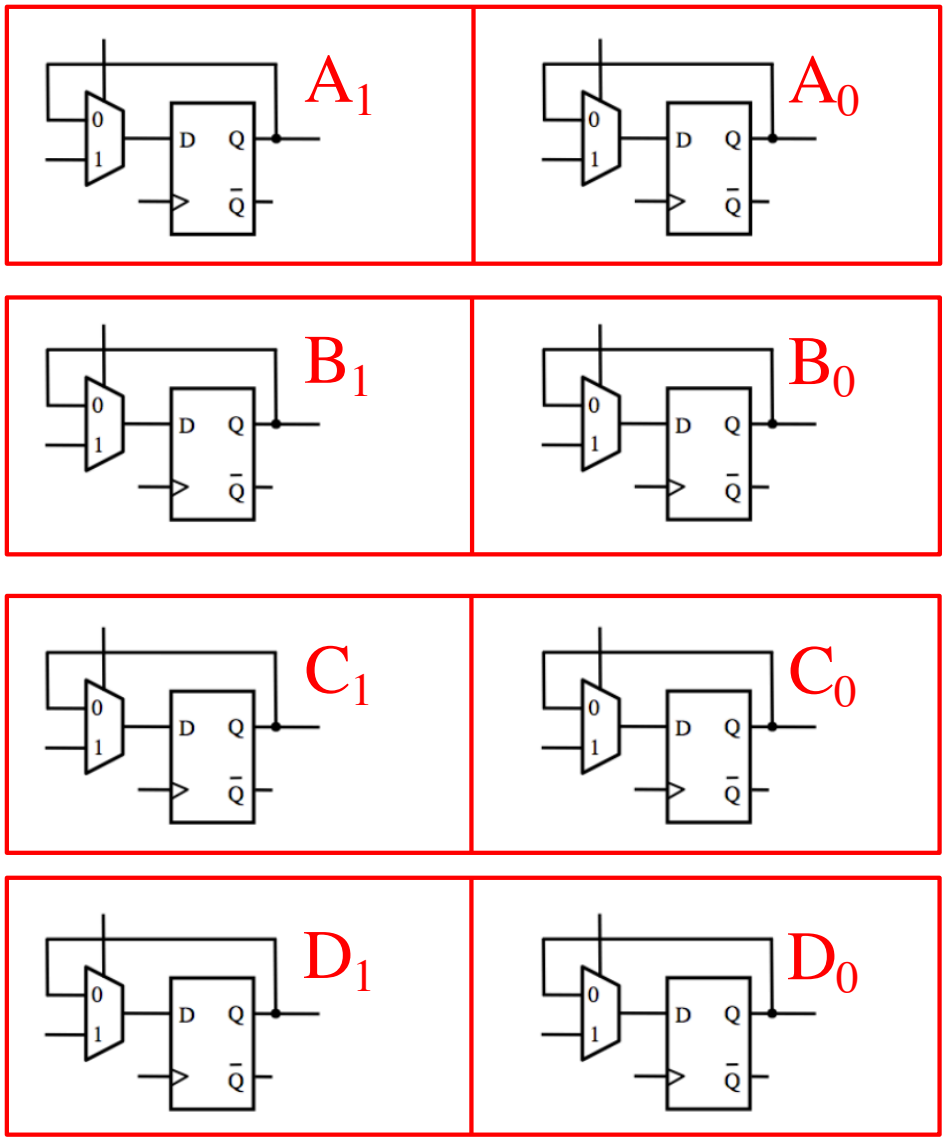

11
10
01
00

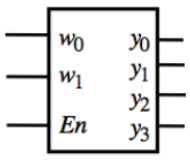

Register A
Register B
Register C
Register D

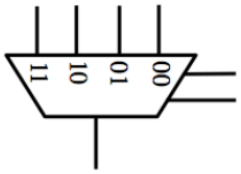

w_0
w_1
En
y_0
y_1
y_2
y_3

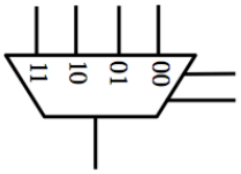

11
10
01
00

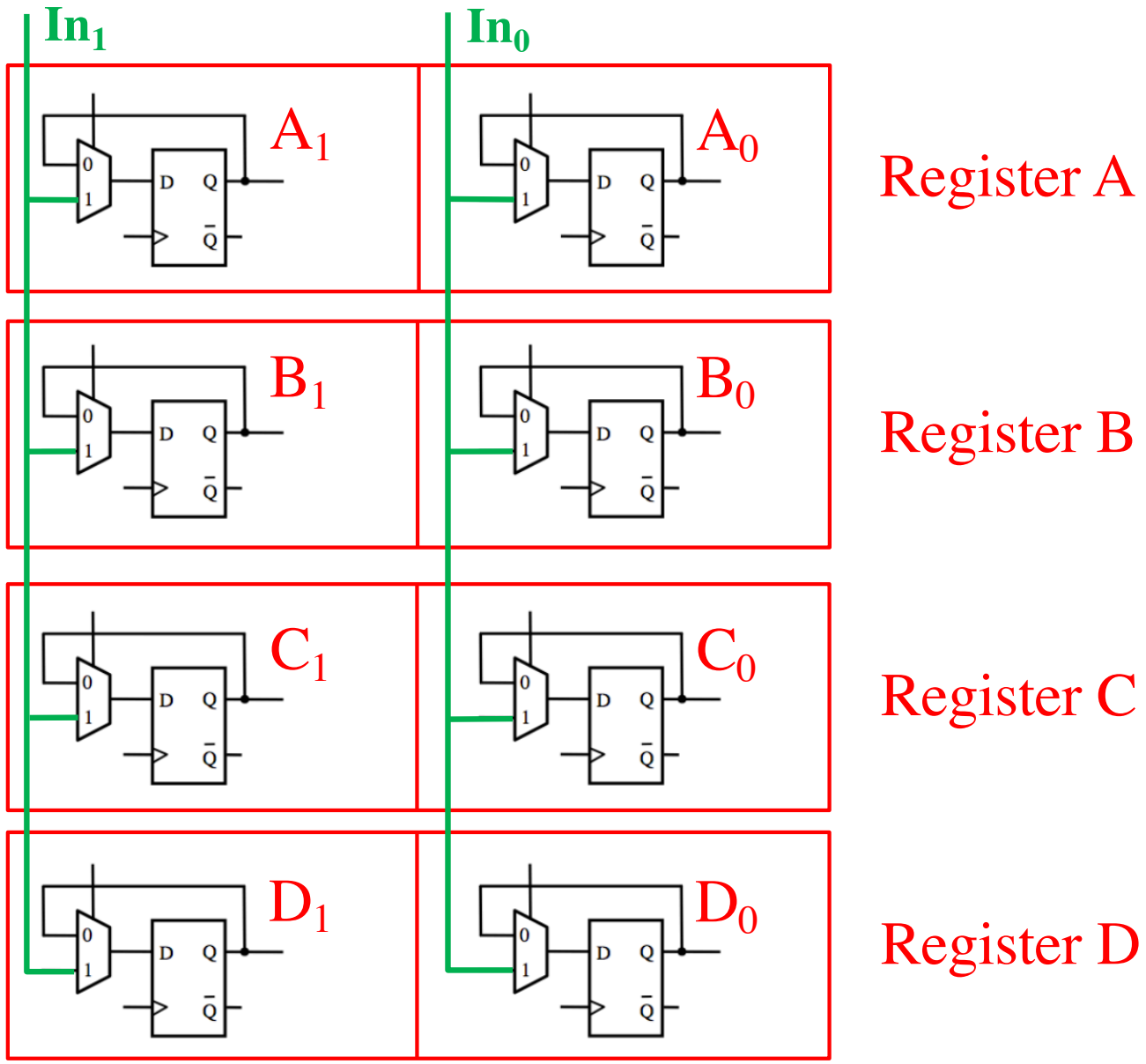

11
10
01
00

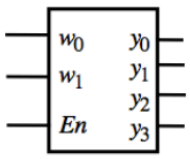

A1
A0
Register A
B1
B0
Register B
C1
C0
Register C
D1
D0
Register D
D
Q
0
1

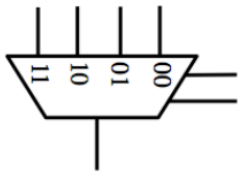

w0
w1
En
y0
y1
y2
y3

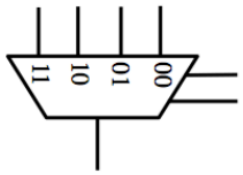

11
10
01
00

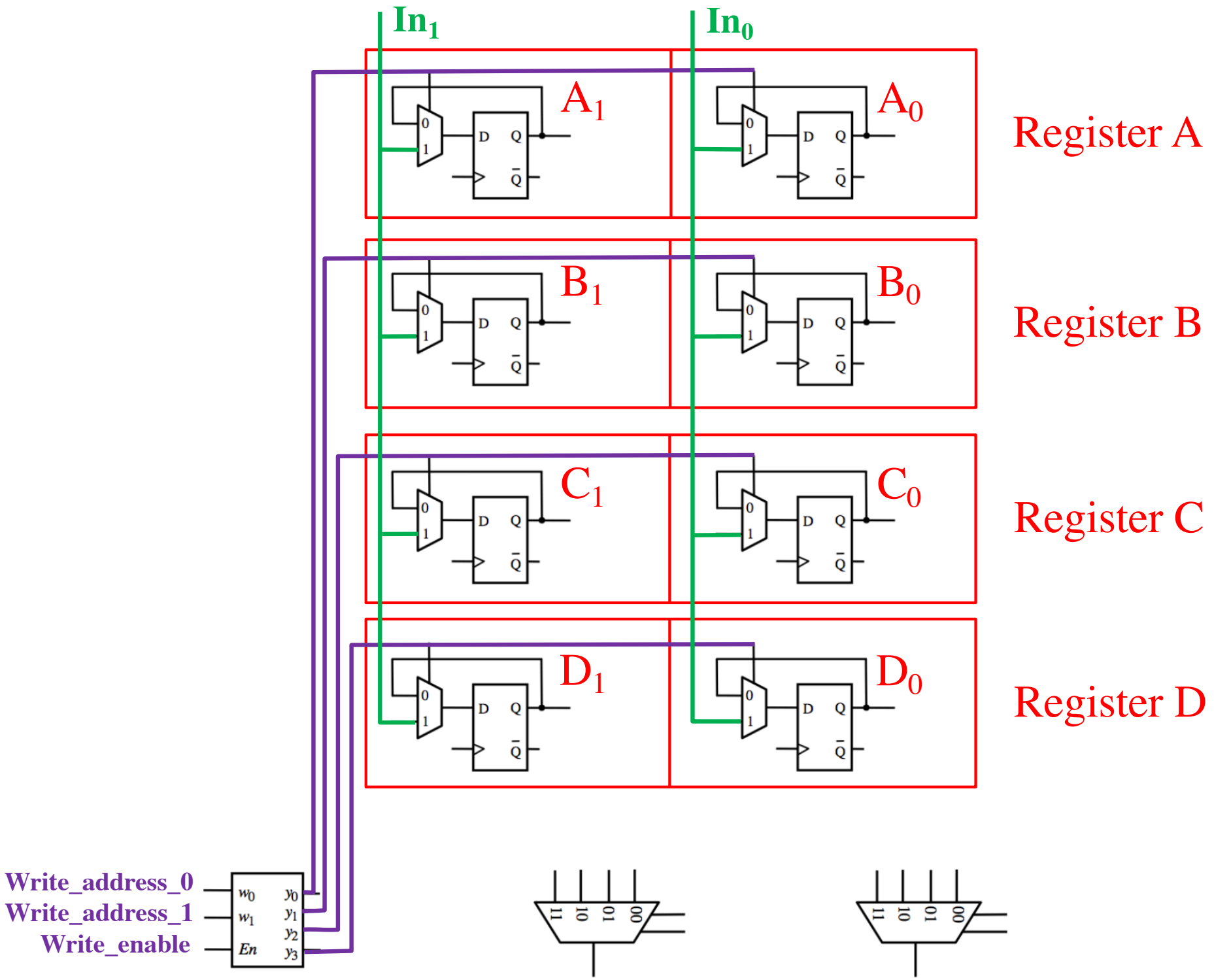

11
10
01
00

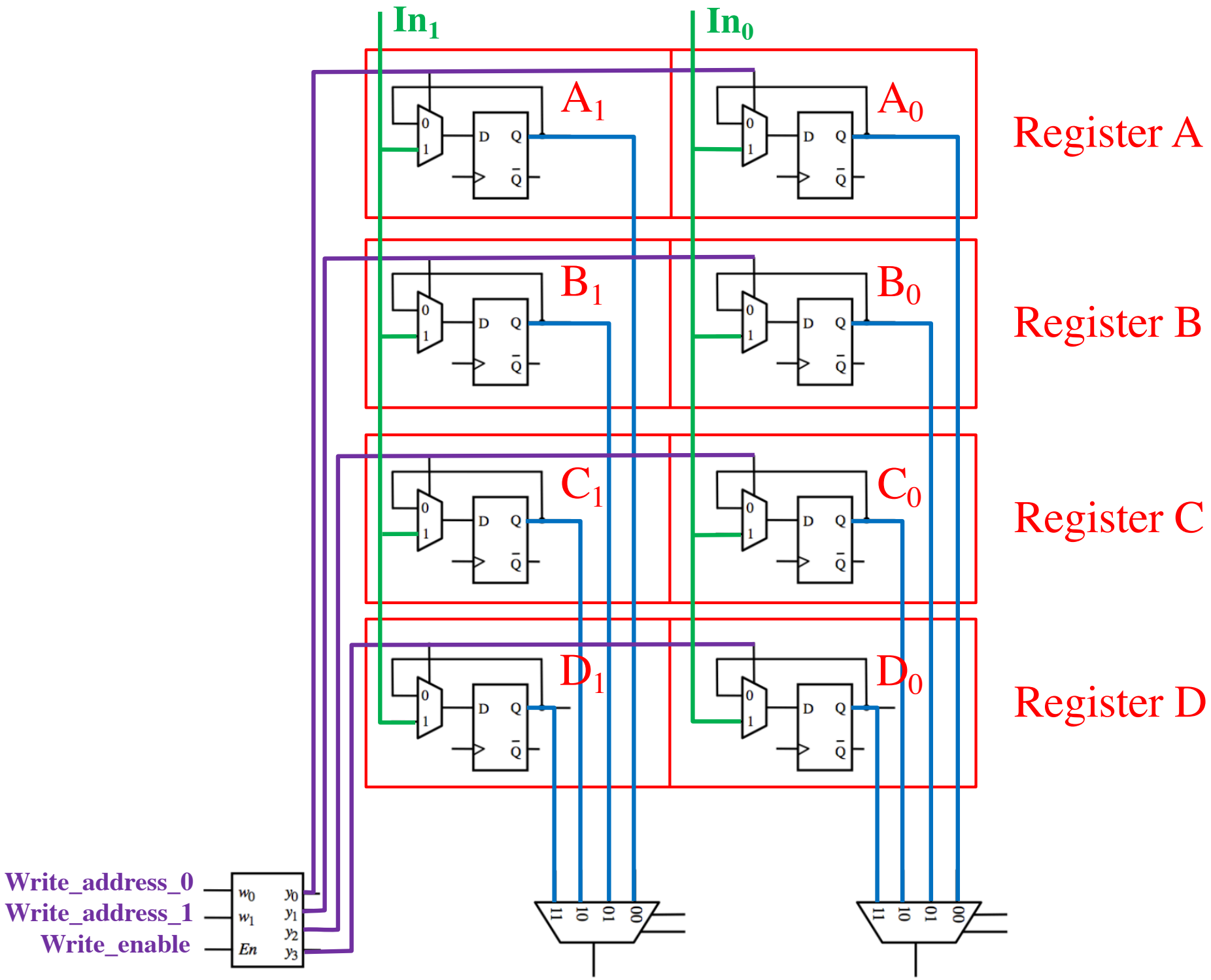

In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
D
Q
0
1

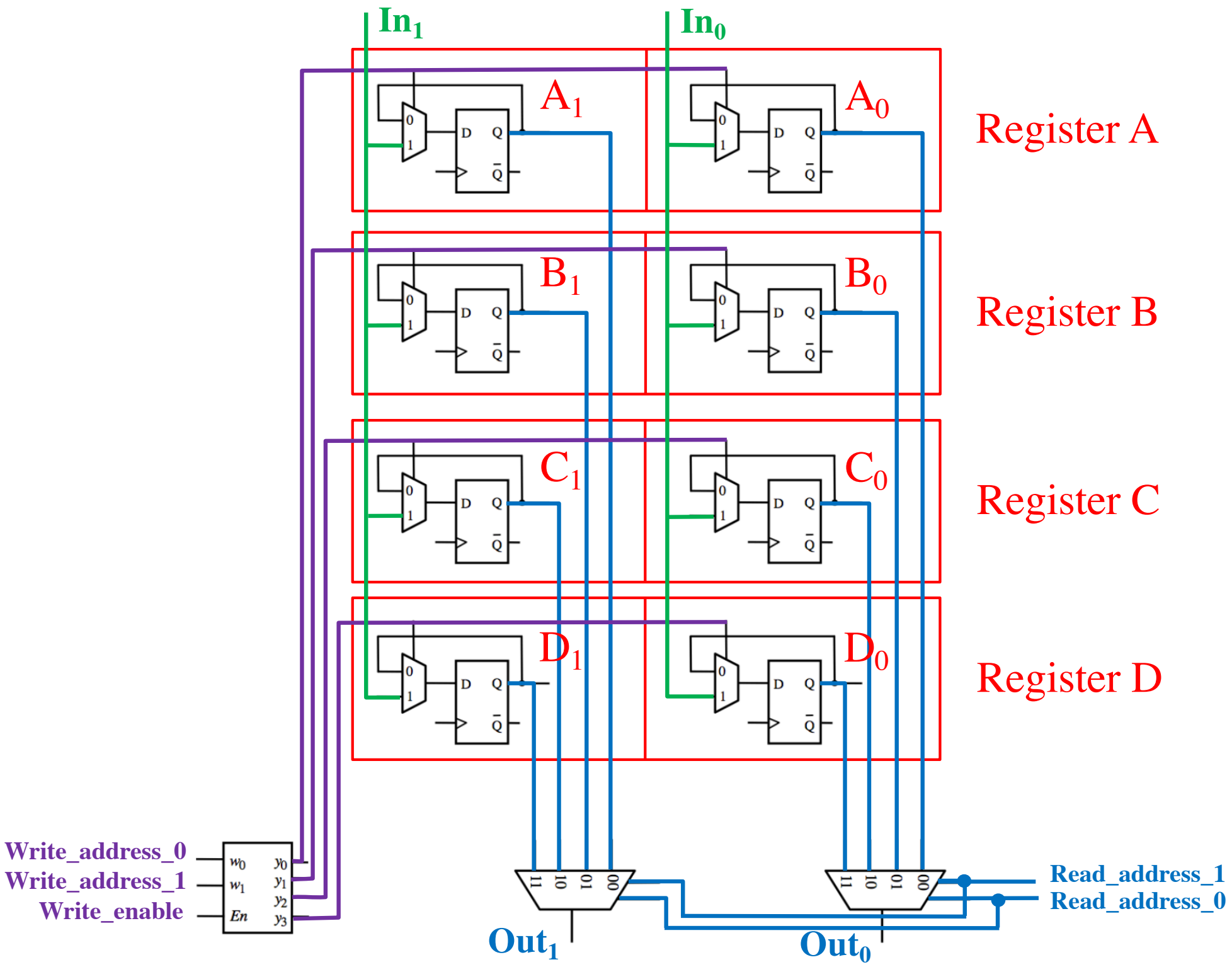

w0
w1
En
y0
y1
y2
y3

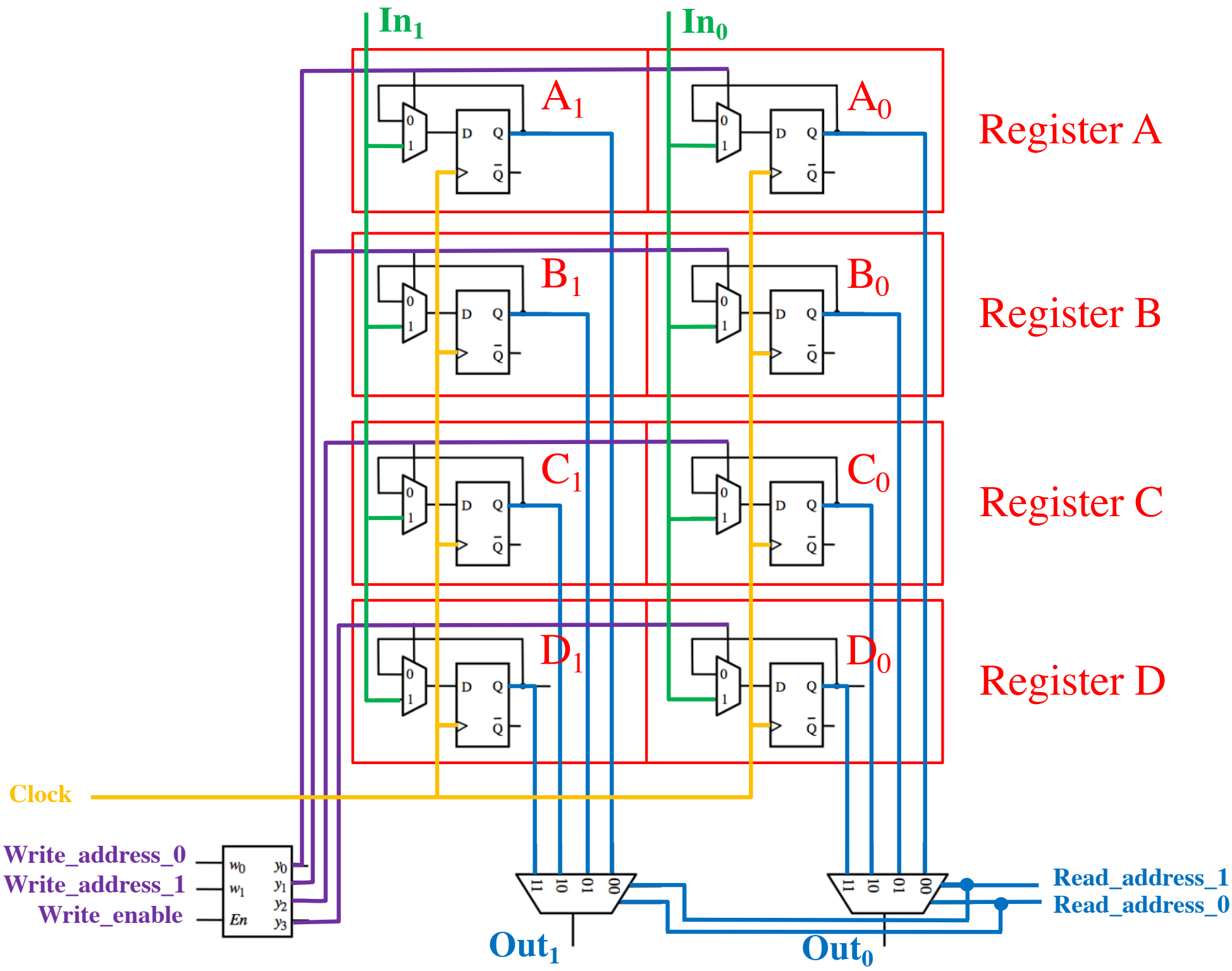

11
10
01
00

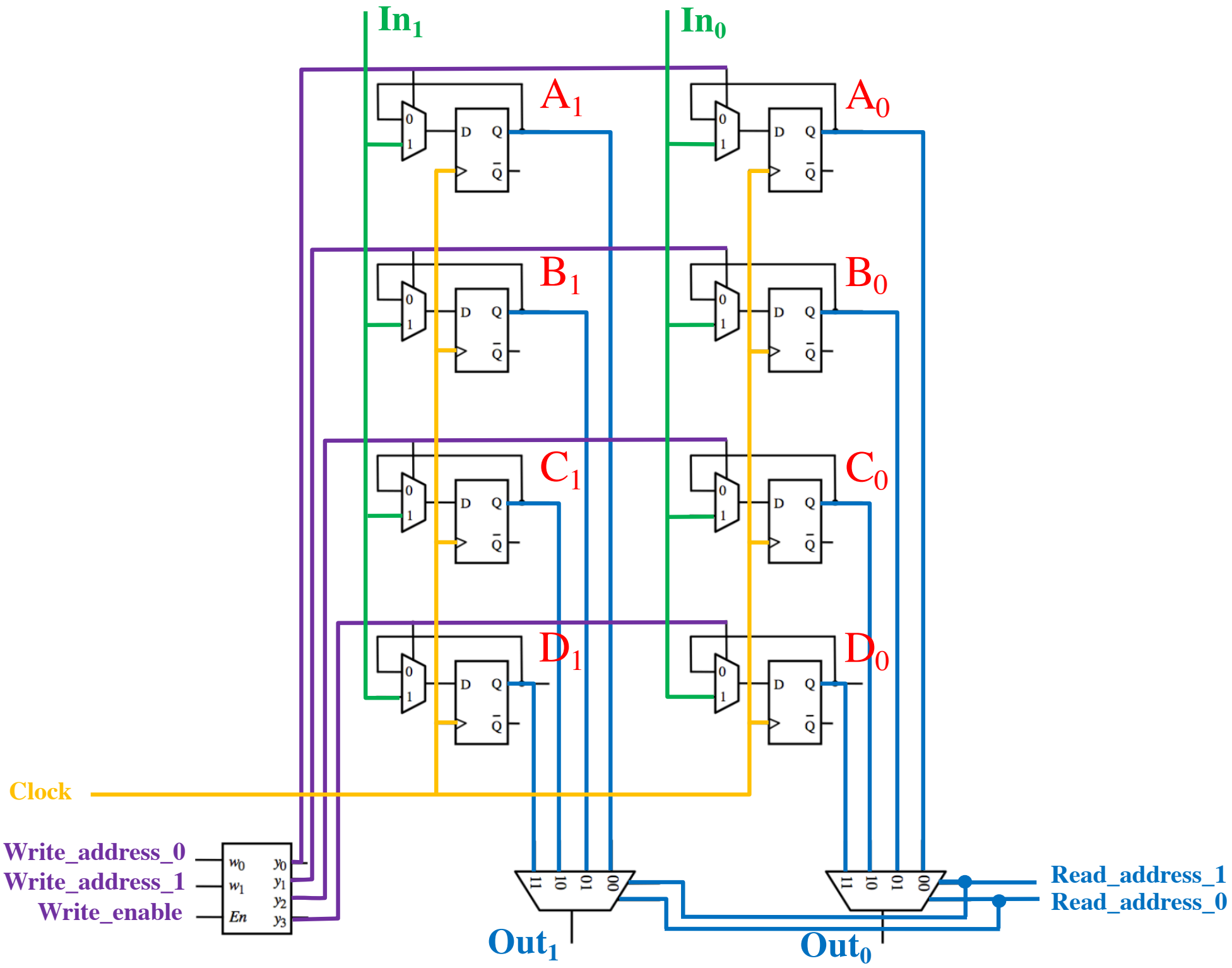

11
10
01
00


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
11
10
01
00


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
0
1
D
Q
Q̄
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
Register A
Register B
Register C
Register D
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0


In1
In0
A1
A0
B1
B0
C1
C0
D1
D0
D
Q
Clock
Write_address_0
Write_address_1
Write_enable
w0
w1
En
y0
y1
y2
y3
11
10
01
00
Out1
Out0
Read_address_1
Read_address_0

Questions?

THE END