



CprE 281: Digital Logic

Instructor: Alexander Stoytchev

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

Registers and Register Files

CprE 281: Digital Logic
Iowa State University, Ames, IA
Copyright © Alexander Stoytchev

Administrative Stuff

- **Homework 8 is due on Monday Oct 25 @ 4pm.**
- **The second midterm exam is next week (Friday Oct 29).**

Administrative Stuff

- **Midterm Exam #2**
- **When: Friday October 29 @ 4:20pm.**
- **Where: This room**
- **What: Chapters 1, 2, 3, 4 and 5**
- **The exam will be closed book but 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

- **K-maps for 2, 3, and 4 variables**
- **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, delay calculation**

- **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**

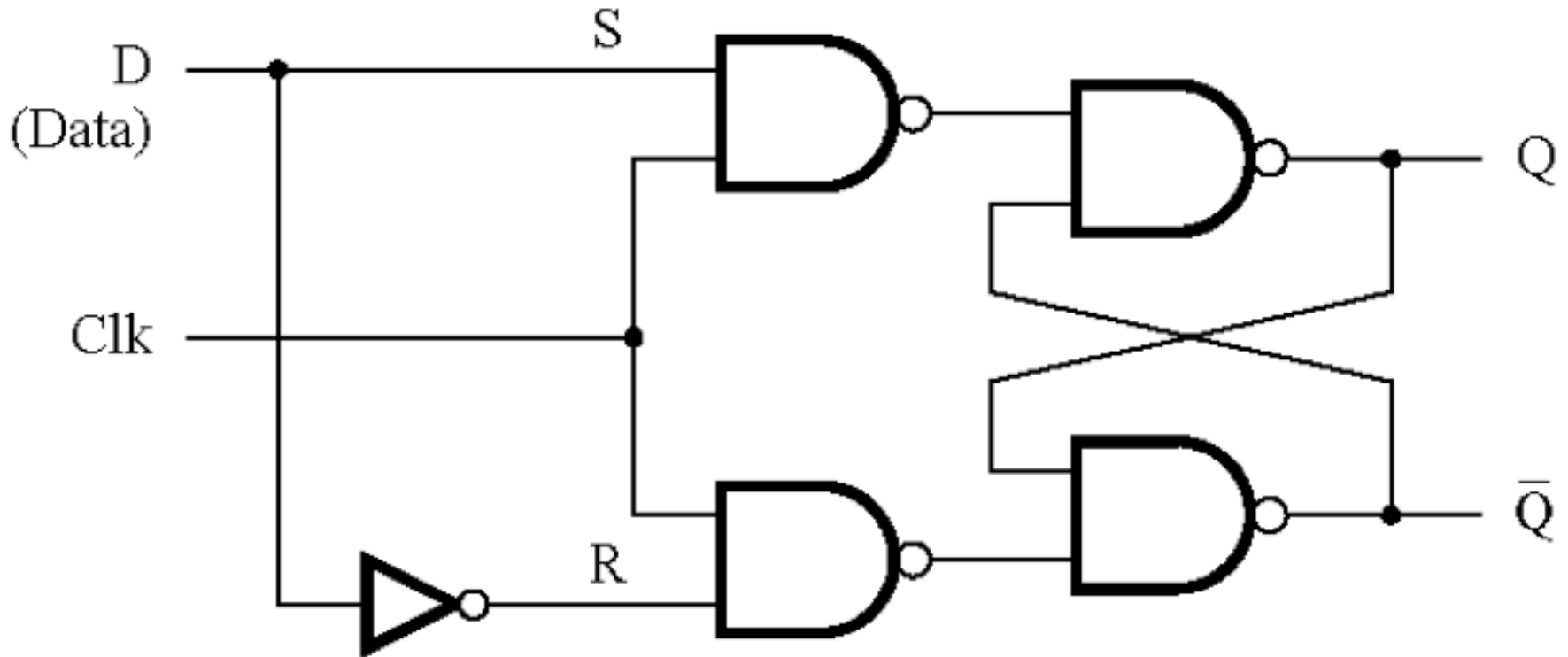
Midterm 2: Topics

- **Decoders (circuits and function)**
- **Demultiplexers**
- **Encoders (binary and priority)**
- **Code Converters and Comparison Circuits**
- **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**
- **Counters**

Quick Review

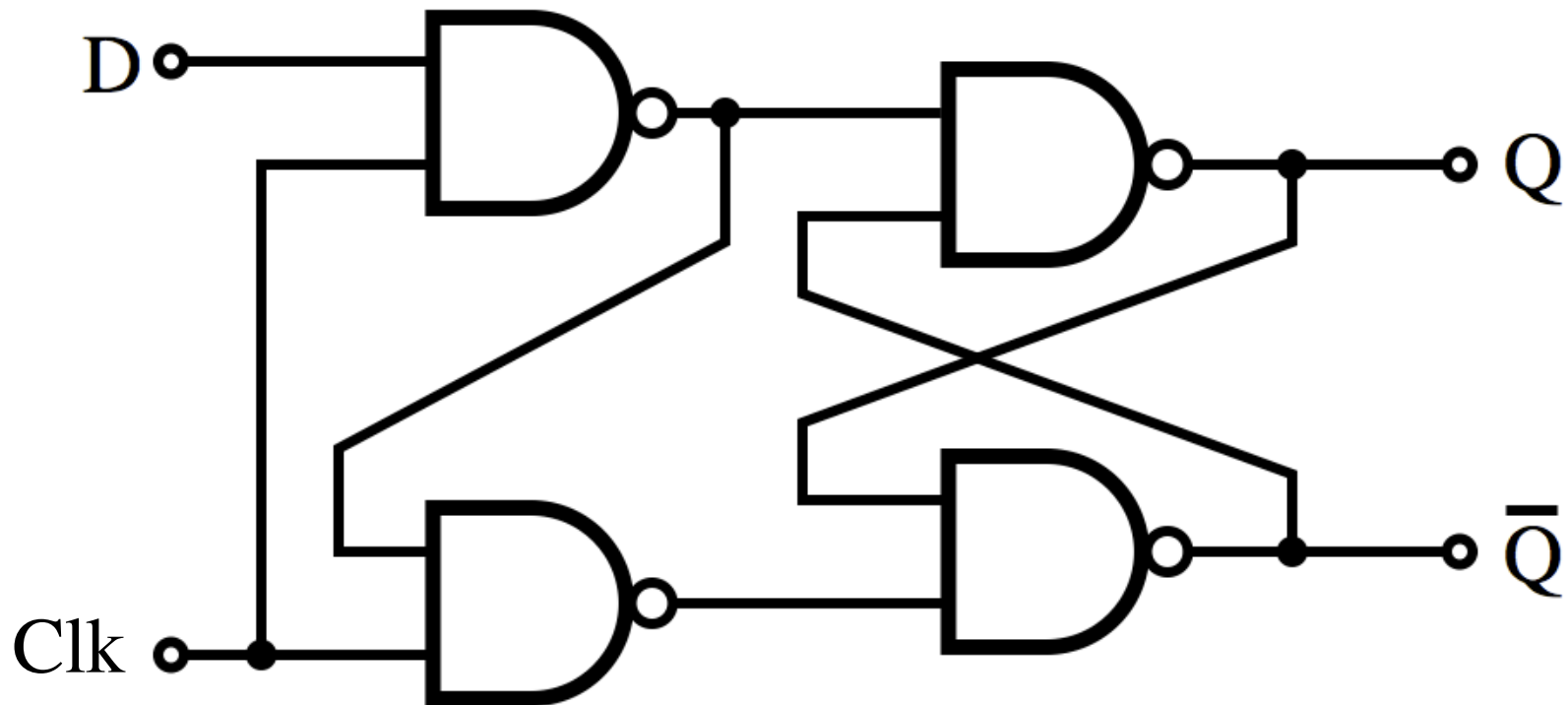
Gated D Latch

Circuit Diagram for the Gated D Latch

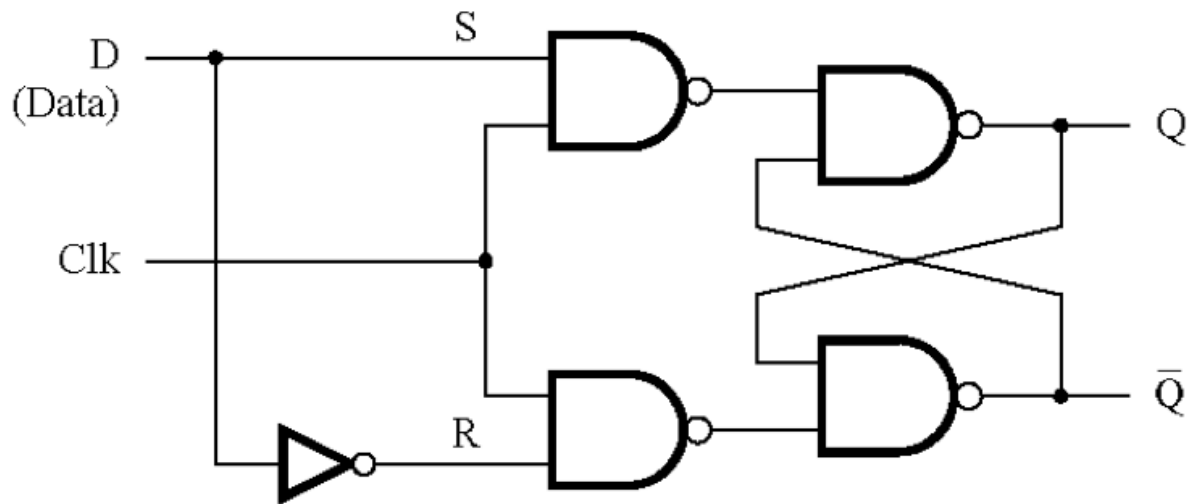


[Figure 5.7a from the textbook]

Gated D Latch: Alternative Design



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

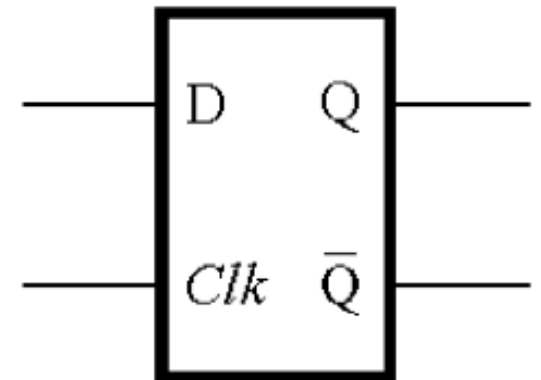


Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

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

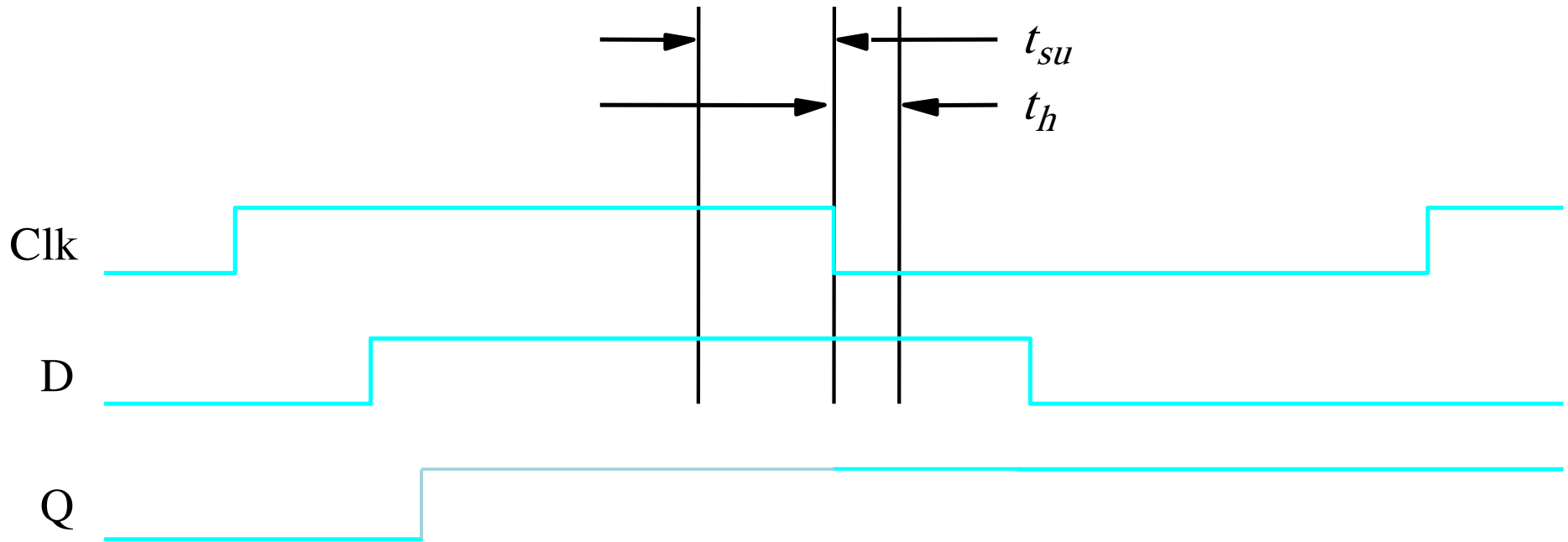
When $Clk=1$ the output follows the D input.

When $Clk=0$ the output cannot be changed.



[Figure 5.7a,b from the textbook]

Setup and hold times for Gated D latch



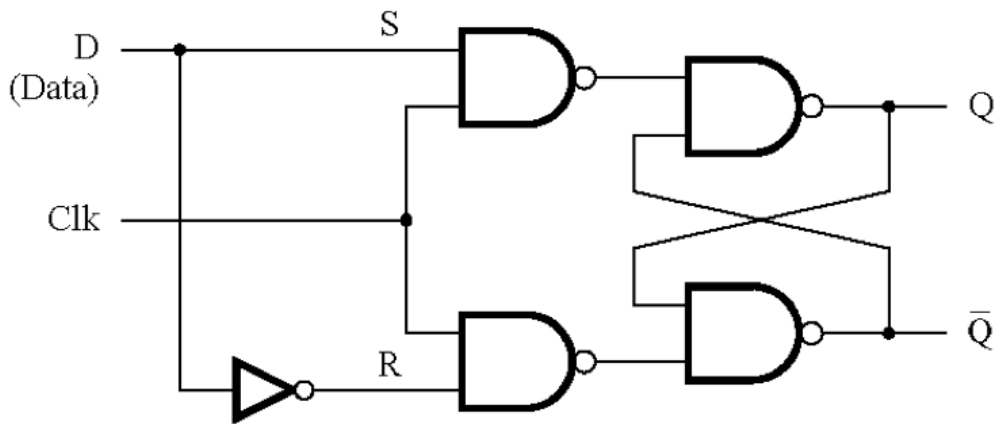
Setup time (t_{su}) – the minimum time that the D signal must be stable prior to the the negative edge of the Clock signal

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

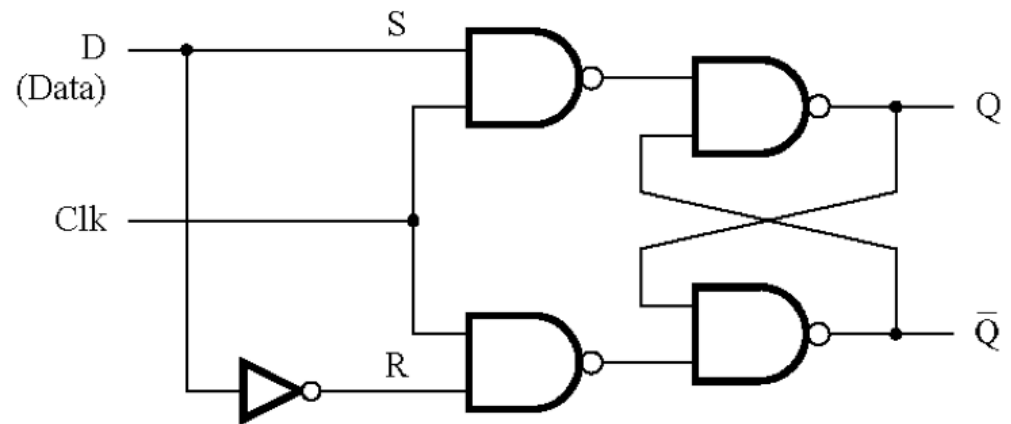
D Flip-Flop

It takes two D latches to construct one D flip-flop

Master Latch



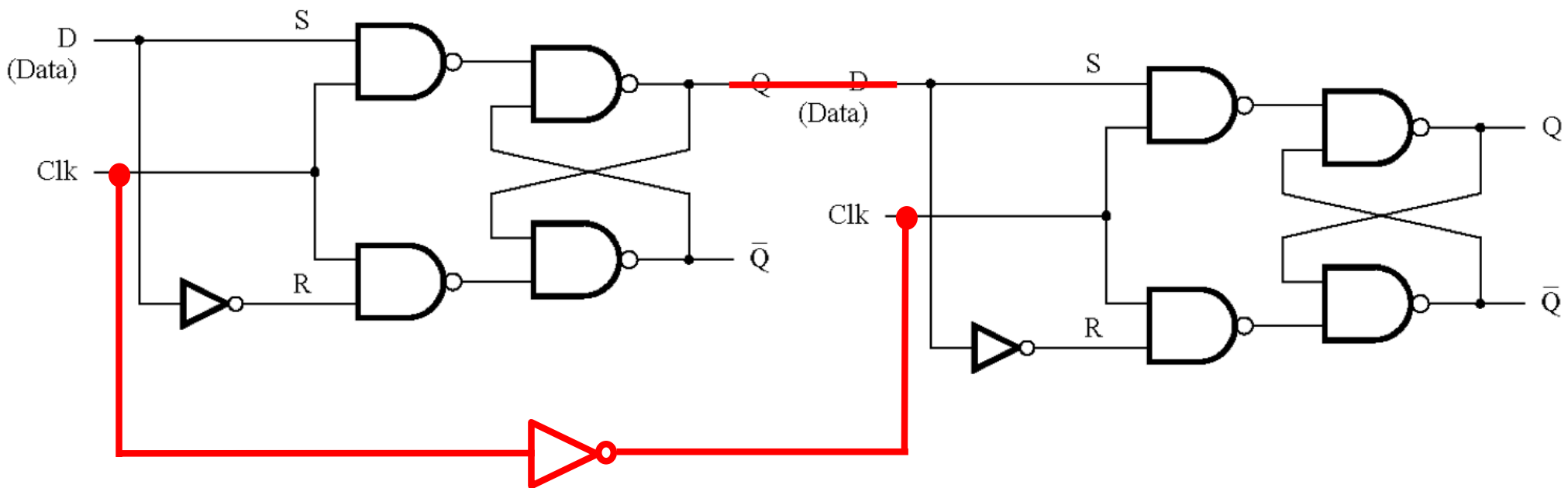
Slave Latch



It takes two D latches to construct one D flip-flop

Master Latch

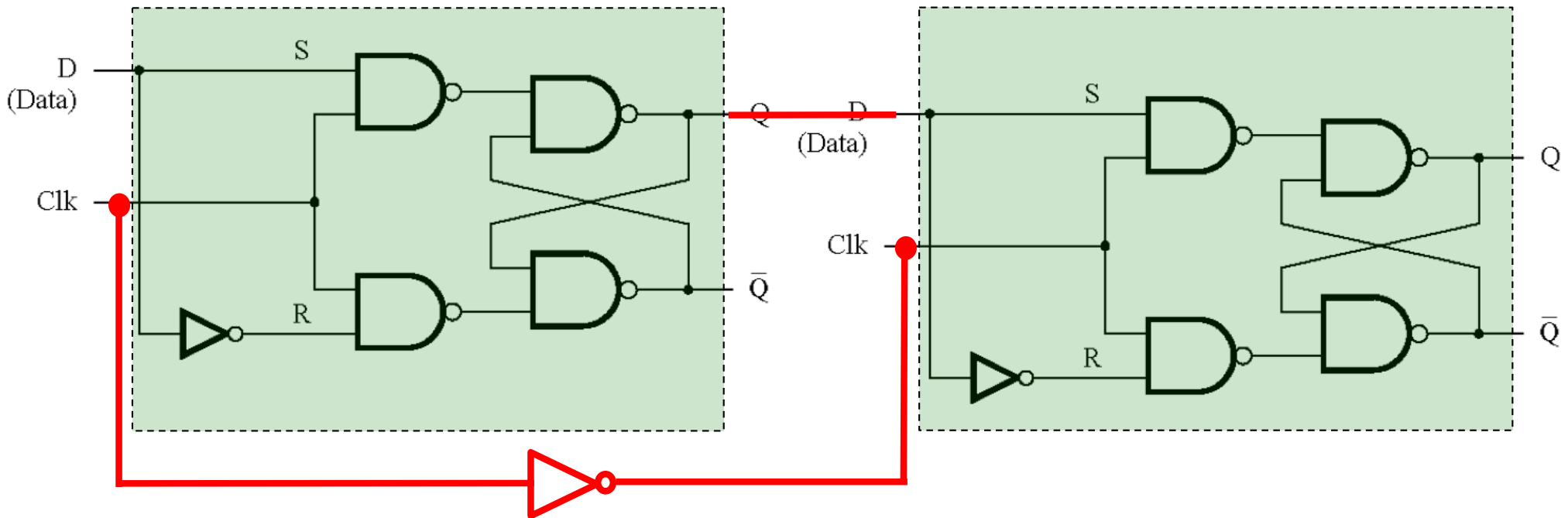
Slave Latch



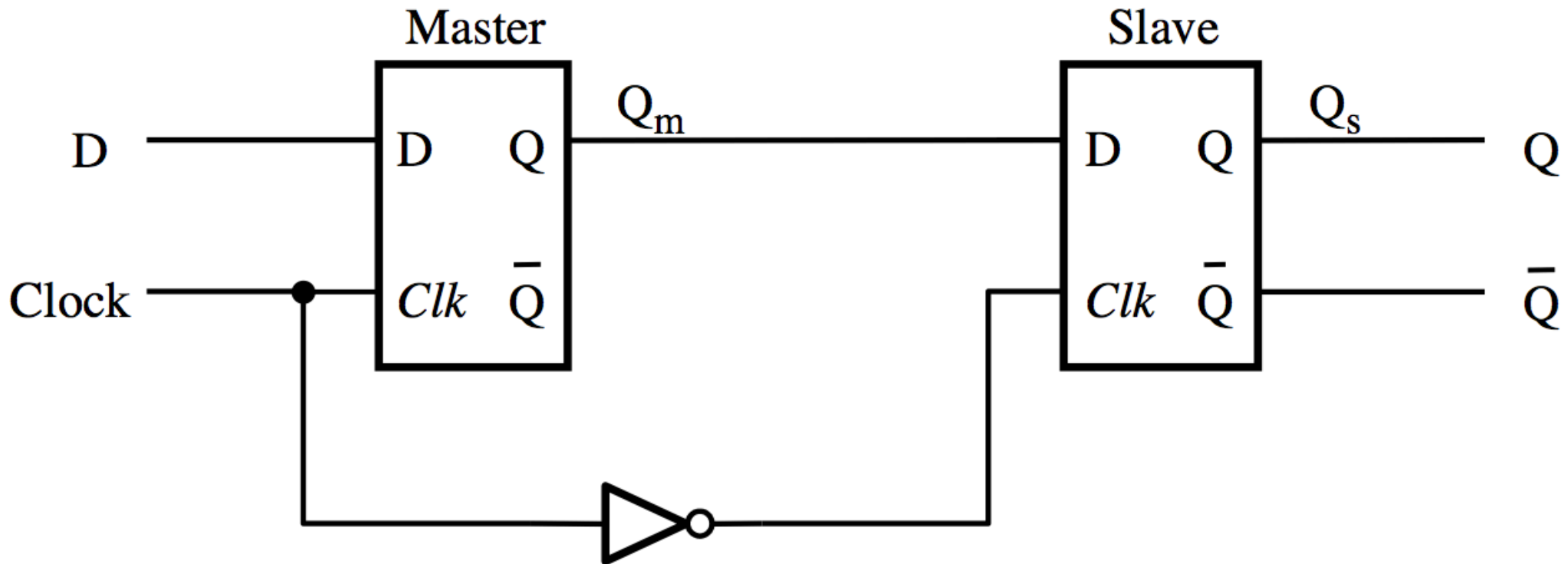
It takes two D latches to construct one D flip-flop

Master Latch

Slave Latch



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

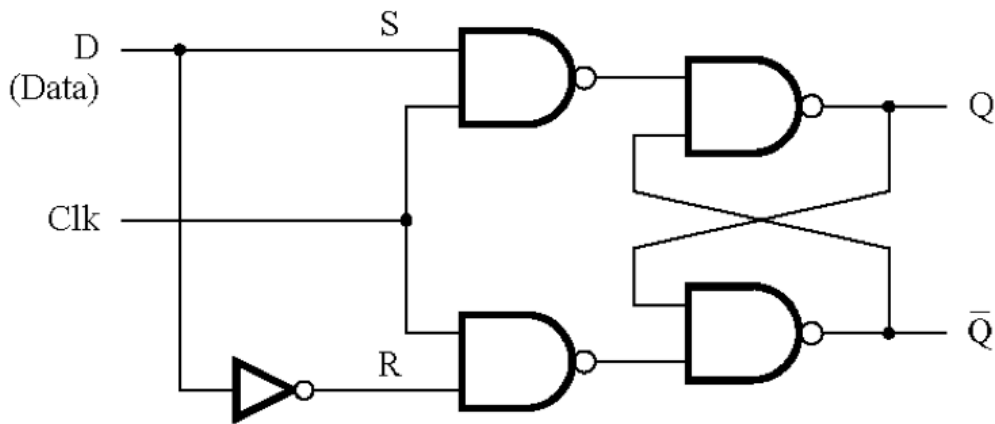


[Figure 5.9a from the textbook]

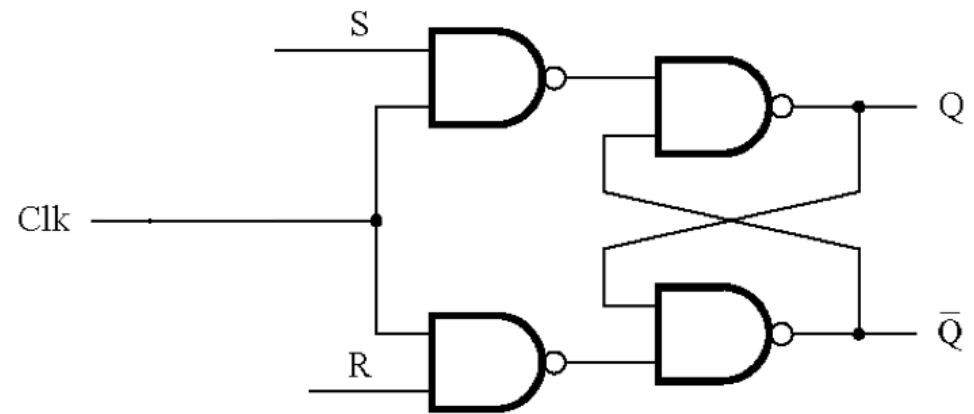
Constructing a Master-Slave D Flip-Flop From one D Latch and one Gated SR Latch

(This version uses one less NOT gate)

Master



Slave

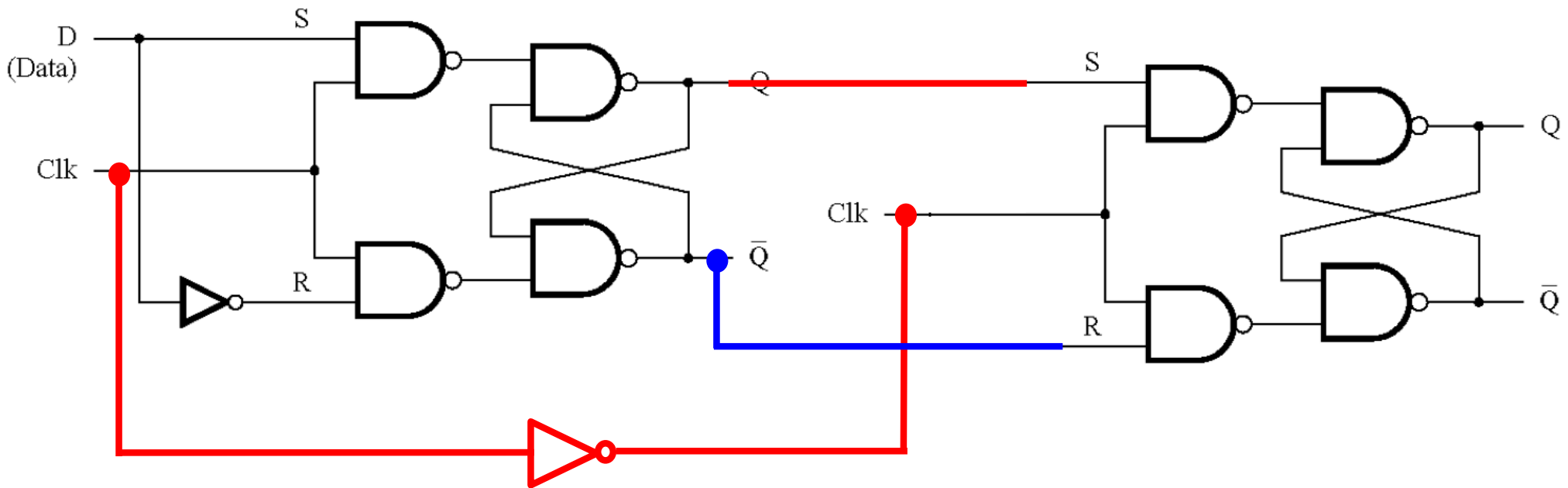


Constructing a Master-Slave D Flip-Flop From one D Latch and one Gated SR Latch

(This version uses one less NOT gate)

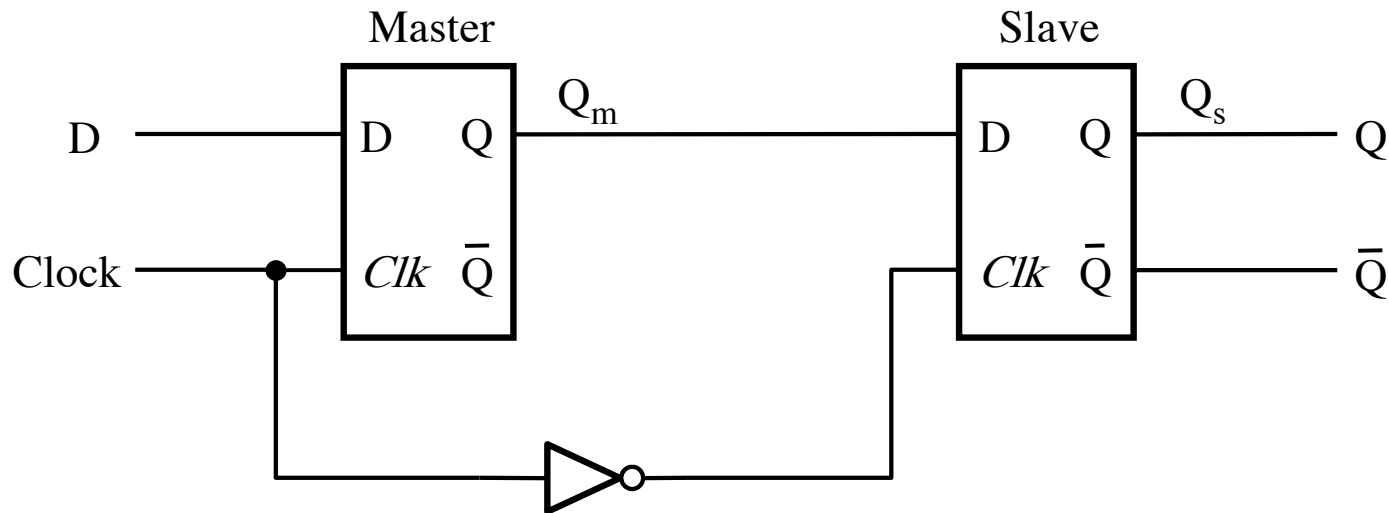
Master

Slave

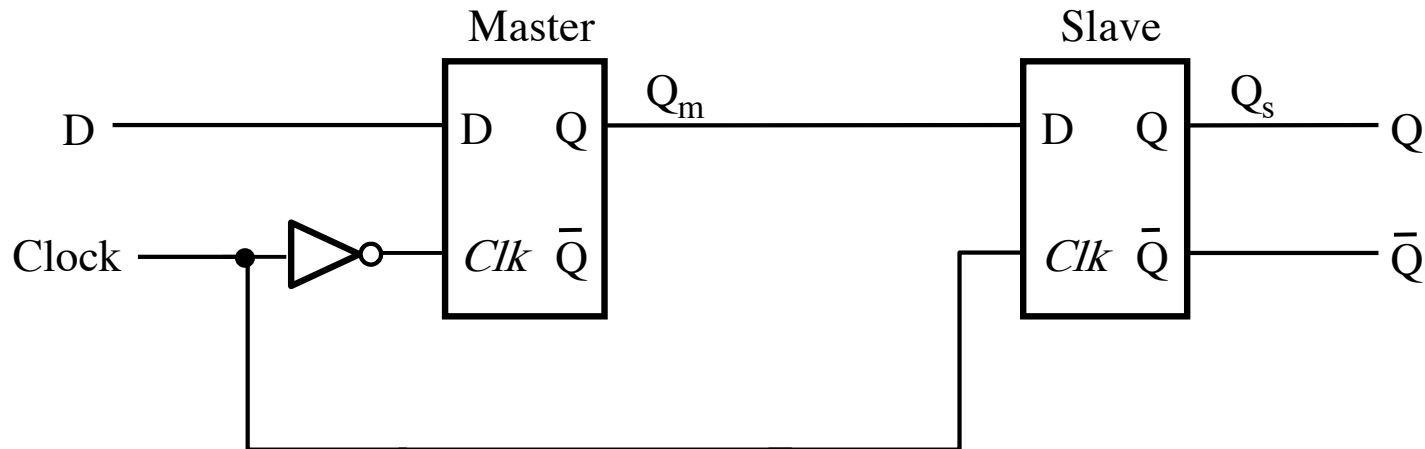


Edge-Triggered D Flip-Flops

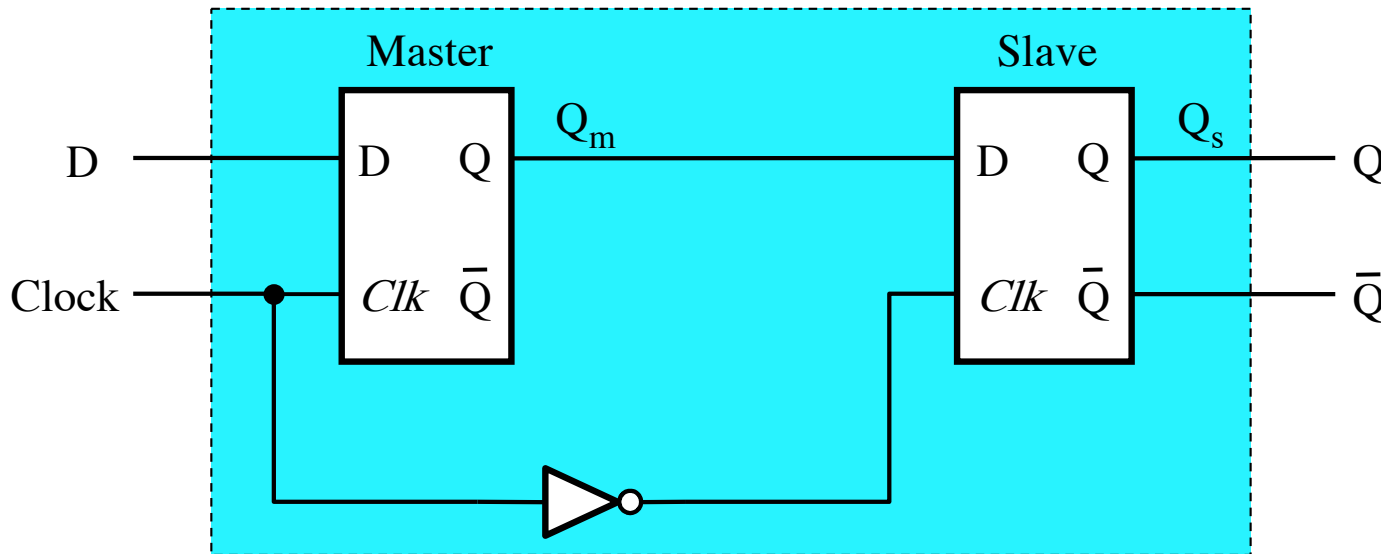
Negative-Edge-Triggered Master-Slave D Flip-Flop



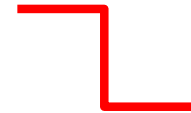
Positive-Edge-Triggered Master-Slave D Flip-Flop



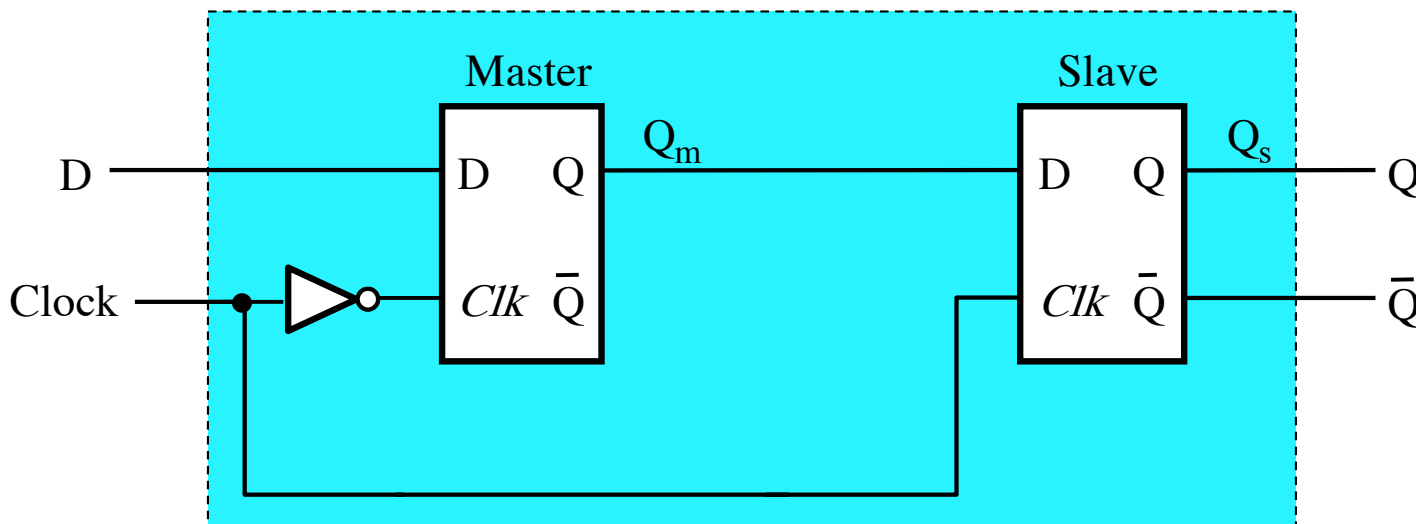
Negative-Edge-Triggered Master-Slave D Flip-Flop



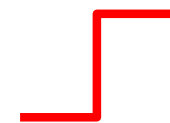
negative edge
(of the Clock)



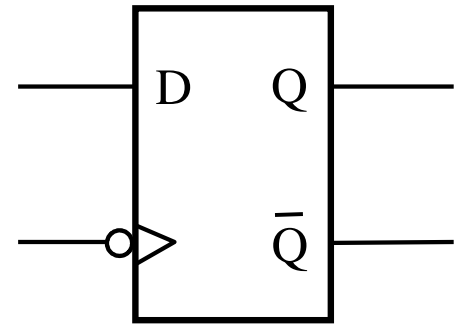
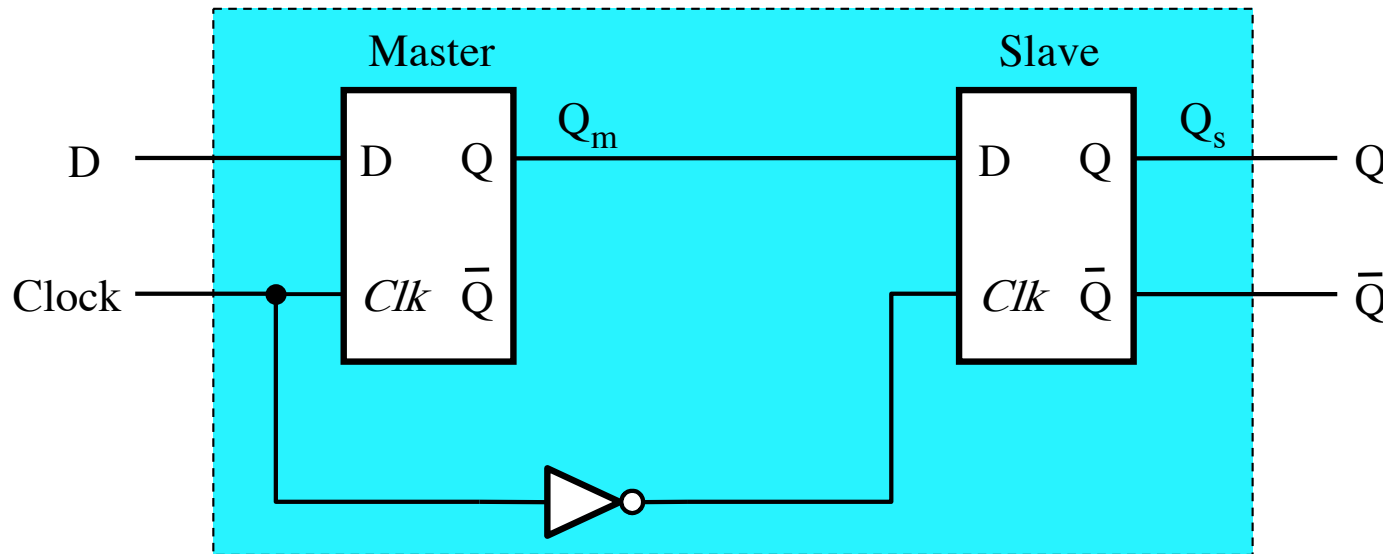
Positive-Edge-Triggered Master-Slave D Flip-Flop



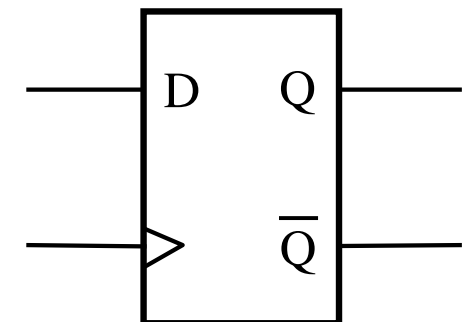
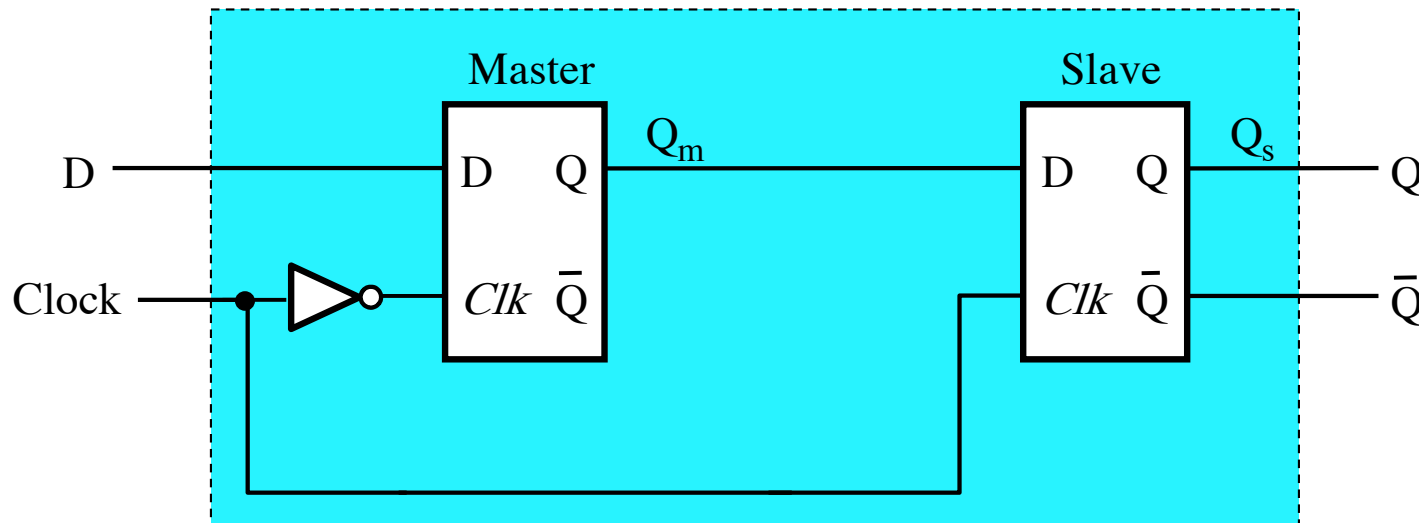
positive edge
(of the Clock)



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

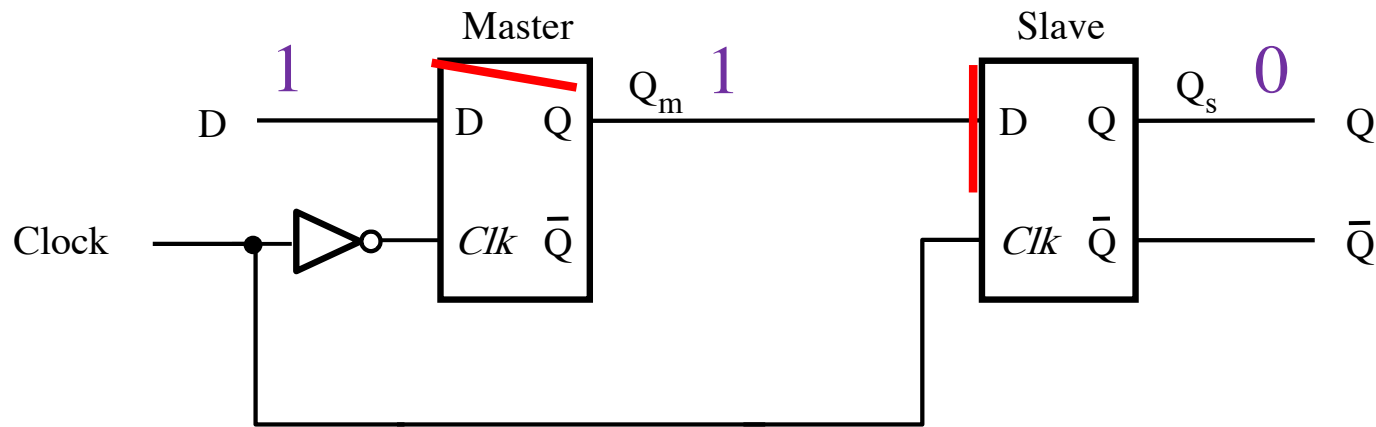


Positive-Edge-Triggered Master-Slave D Flip-Flop

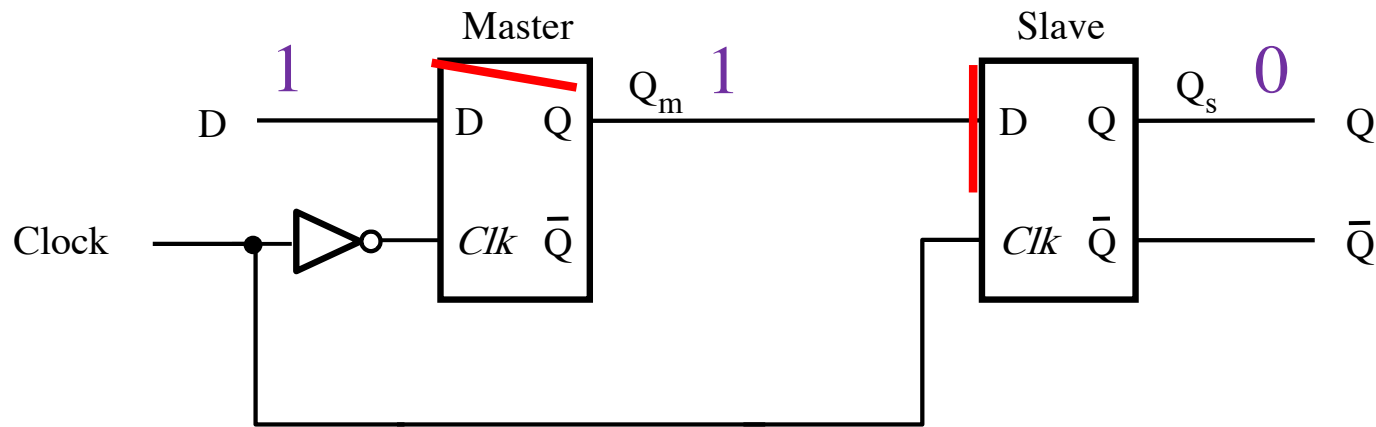


D Flip-Flop: A Double Door Analogy

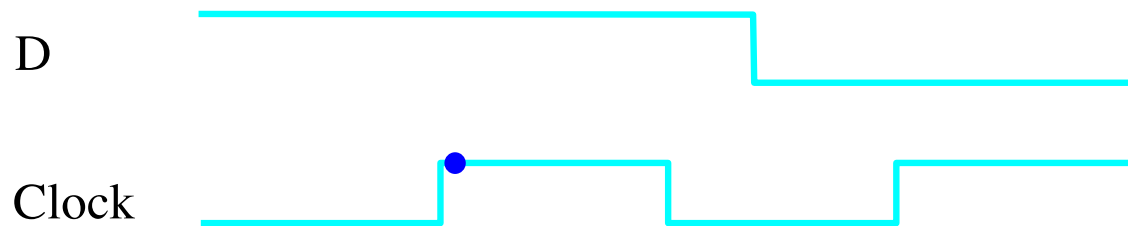
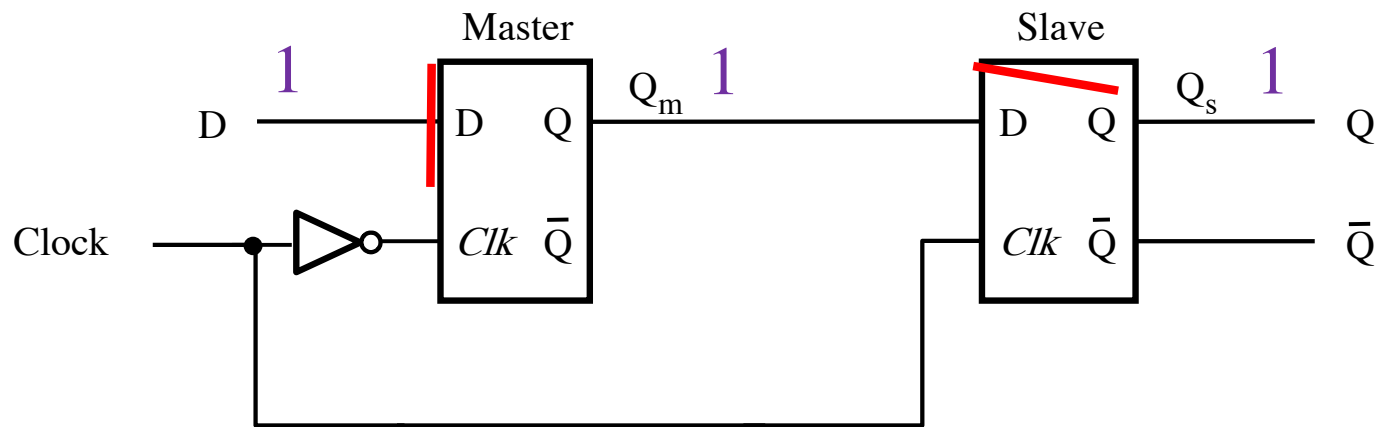
Positive-Edge-Triggered Master-Slave D Flip-Flop



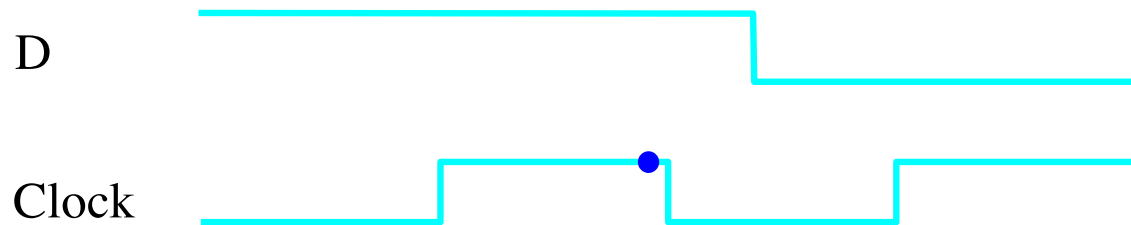
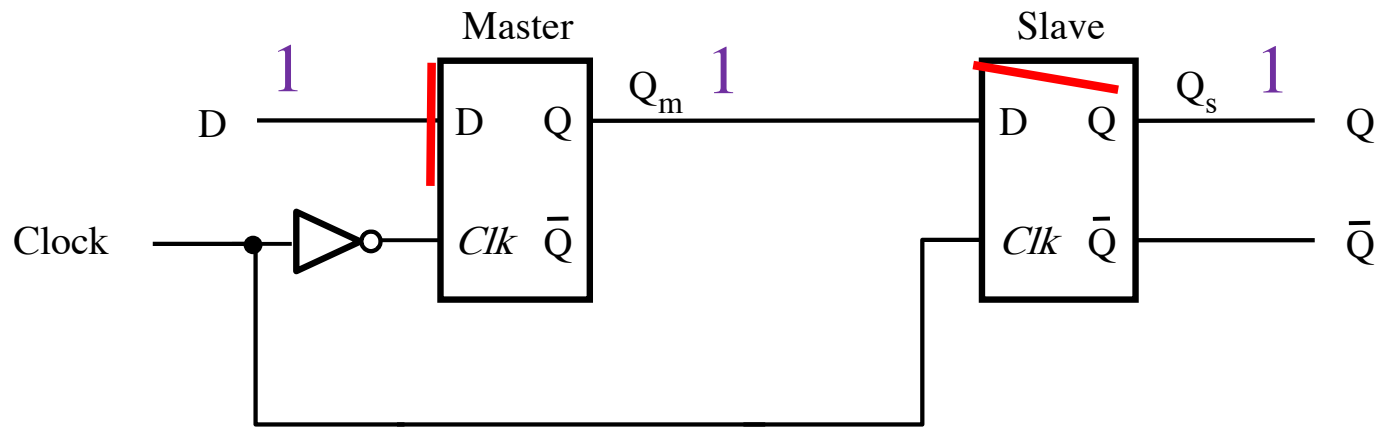
Positive-Edge-Triggered Master-Slave D Flip-Flop



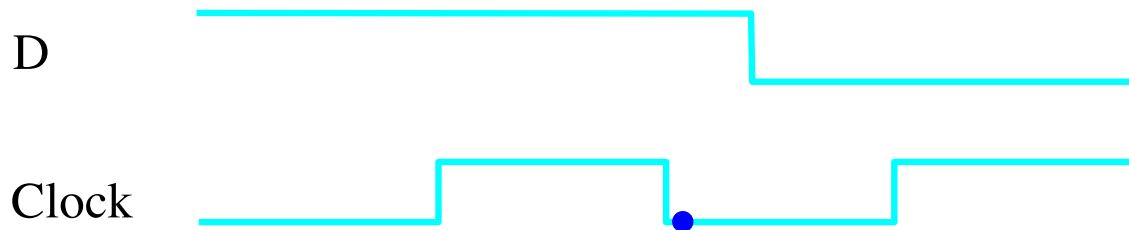
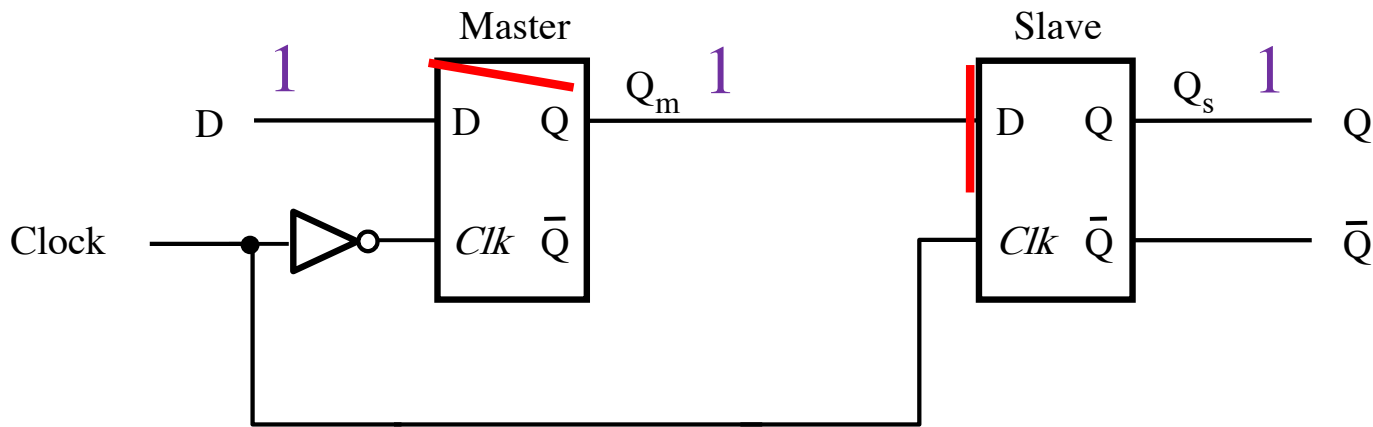
Positive-Edge-Triggered Master-Slave D Flip-Flop



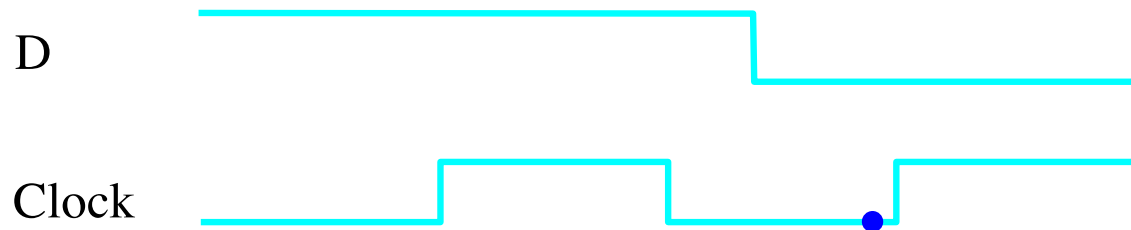
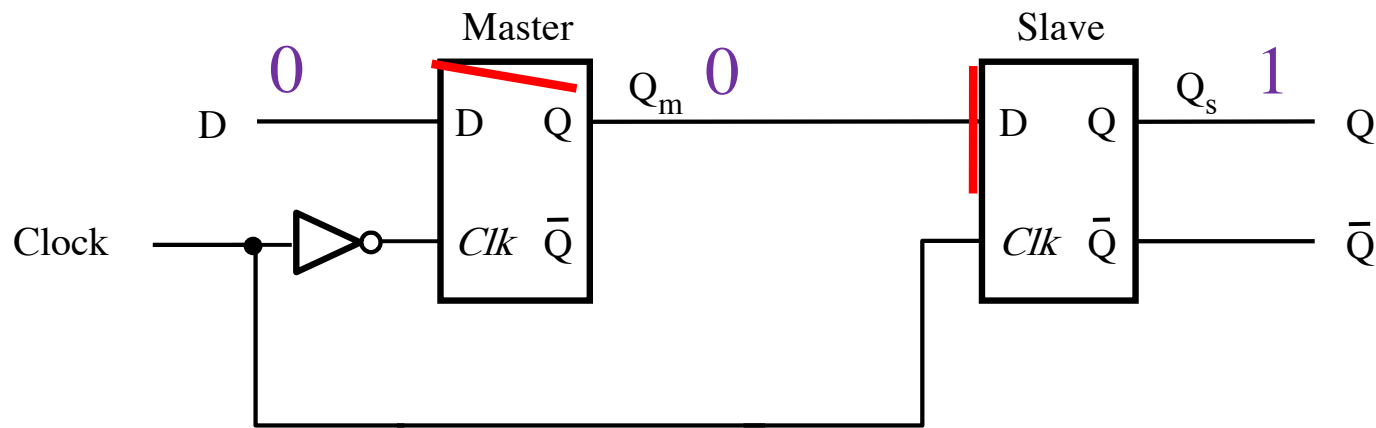
Positive-Edge-Triggered Master-Slave D Flip-Flop



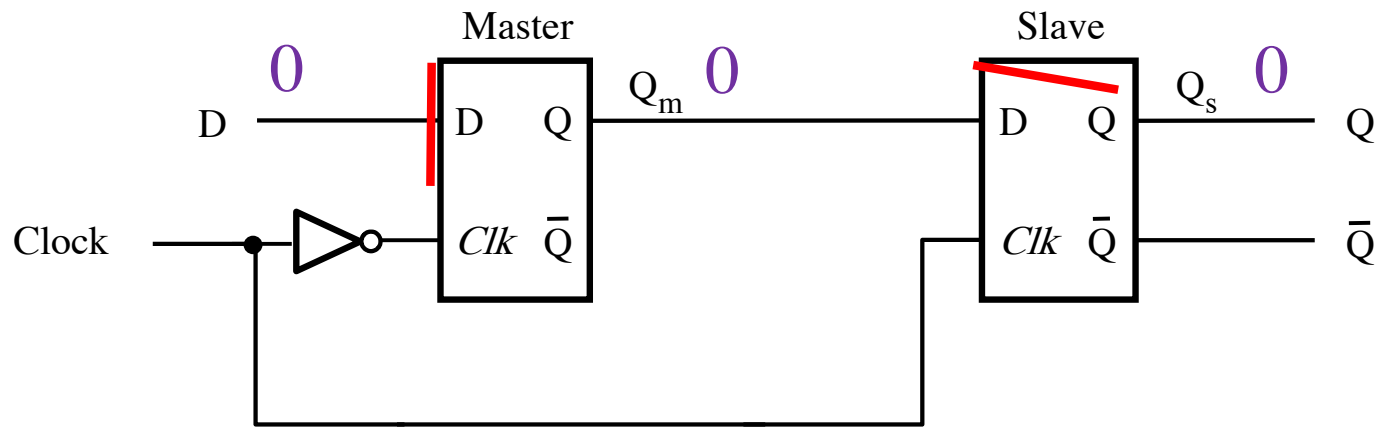
Positive-Edge-Triggered Master-Slave D Flip-Flop



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

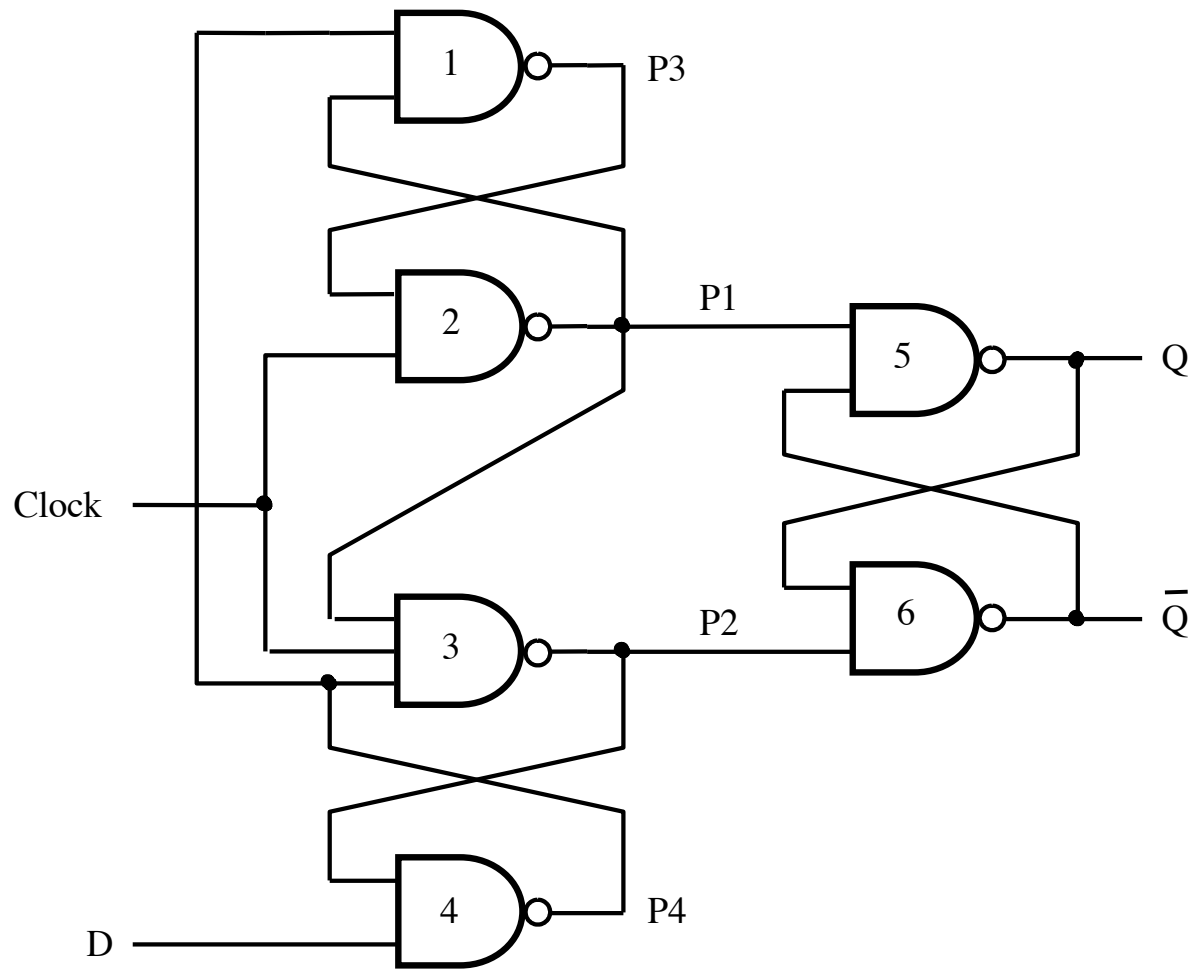


Positive-Edge-Triggered Master-Slave D Flip-Flop

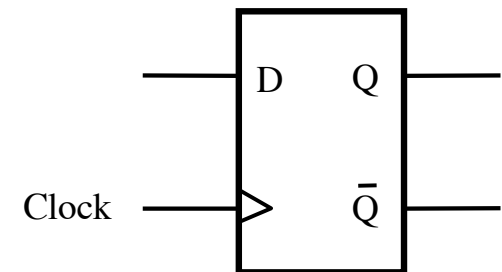


An alternative D Flip-Flop Design

A positive-edge-triggered D flip-flop



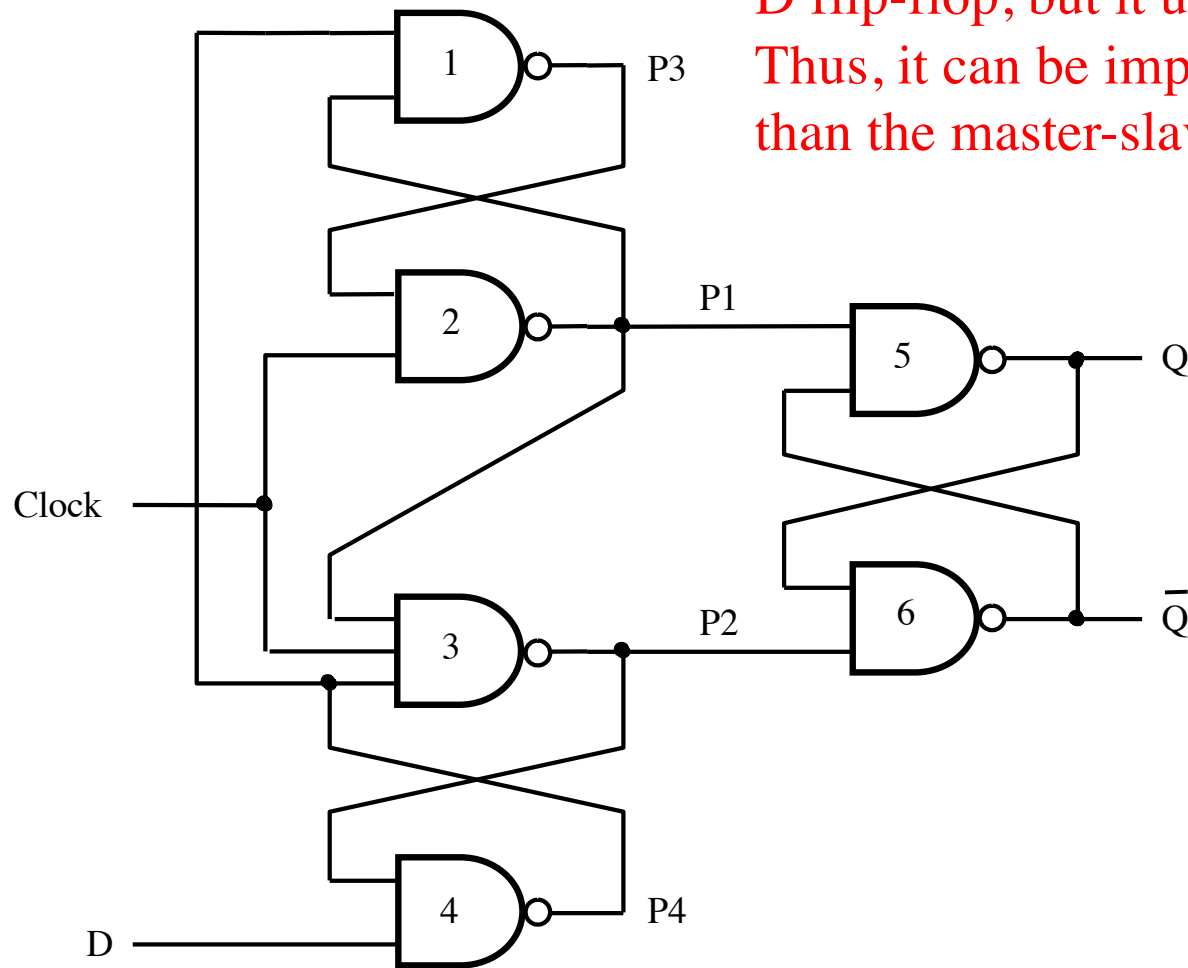
(a) Circuit



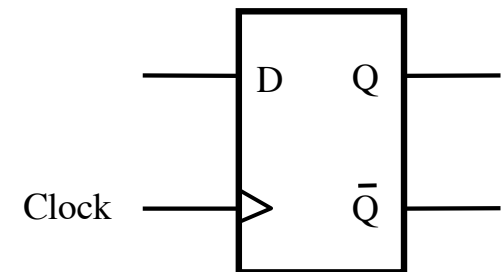
(b) Graphical symbol

A positive-edge-triggered D flip-flop

This circuit behaves like a positive-edge-triggered D flip-flop, but it uses only 6 NAND gates. Thus, it can be implemented with fewer transistors than the master-slave D flip-flop.



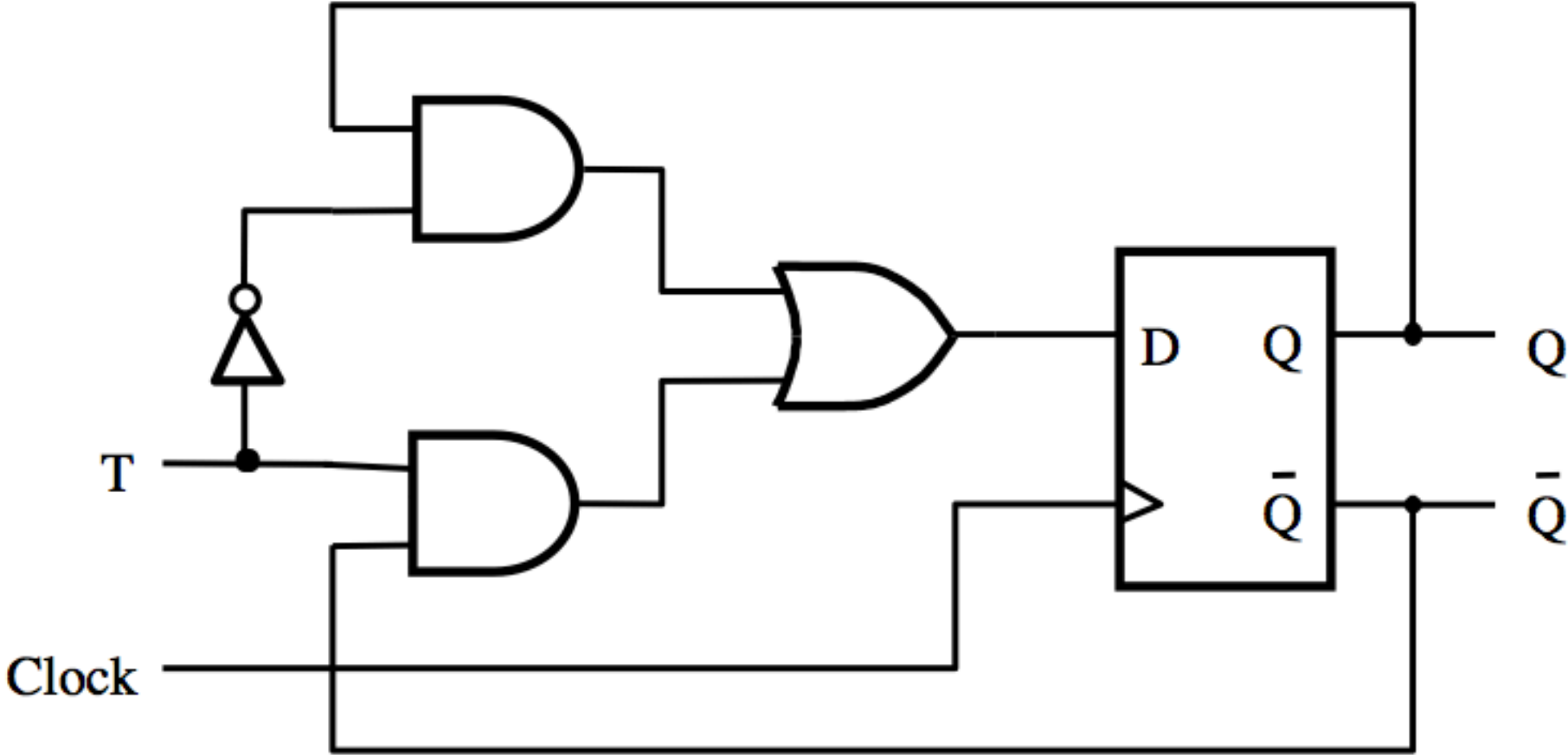
(a) Circuit



(b) Graphical symbol

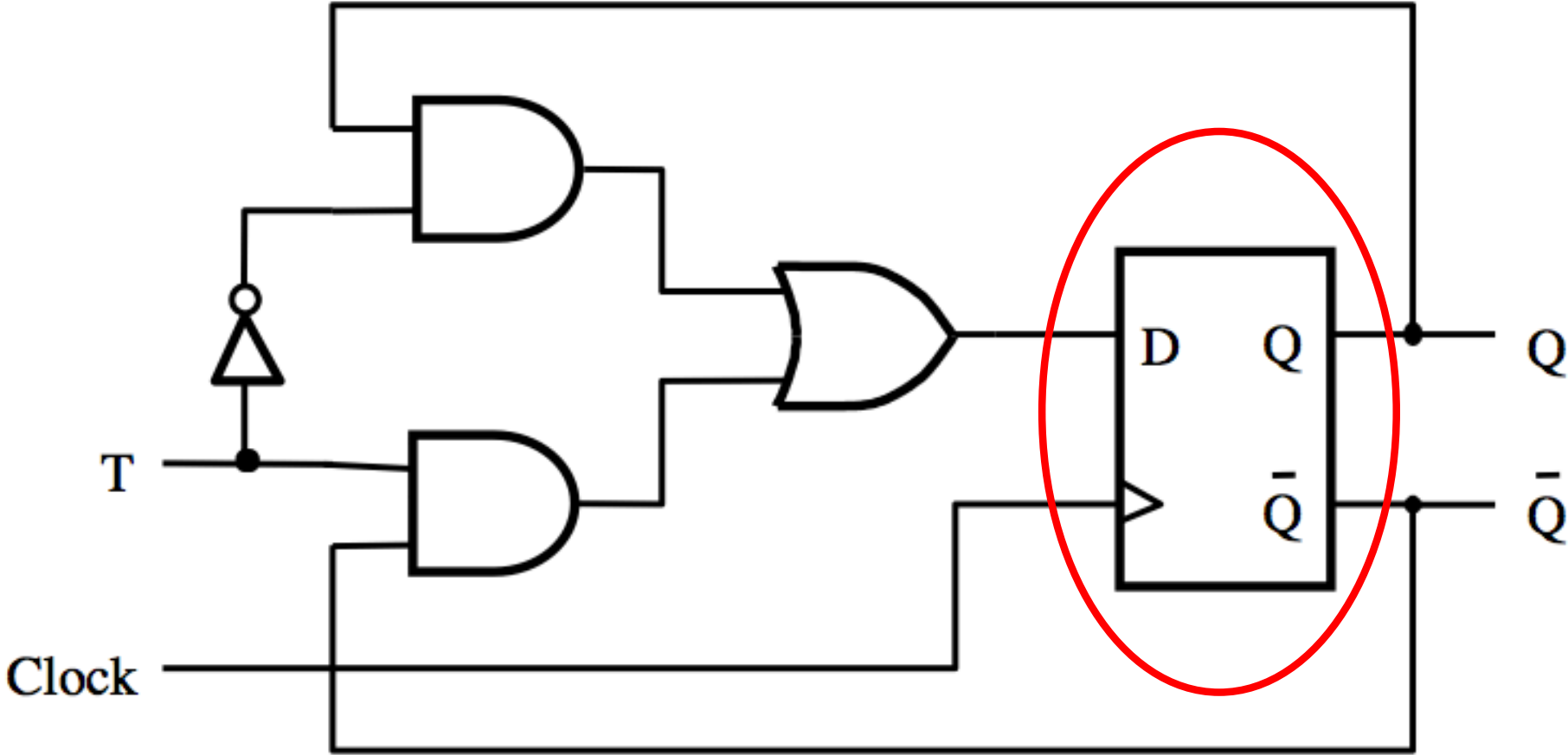
T Flip-Flop

T Flip-Flop



[Figure 5.15a from the textbook]

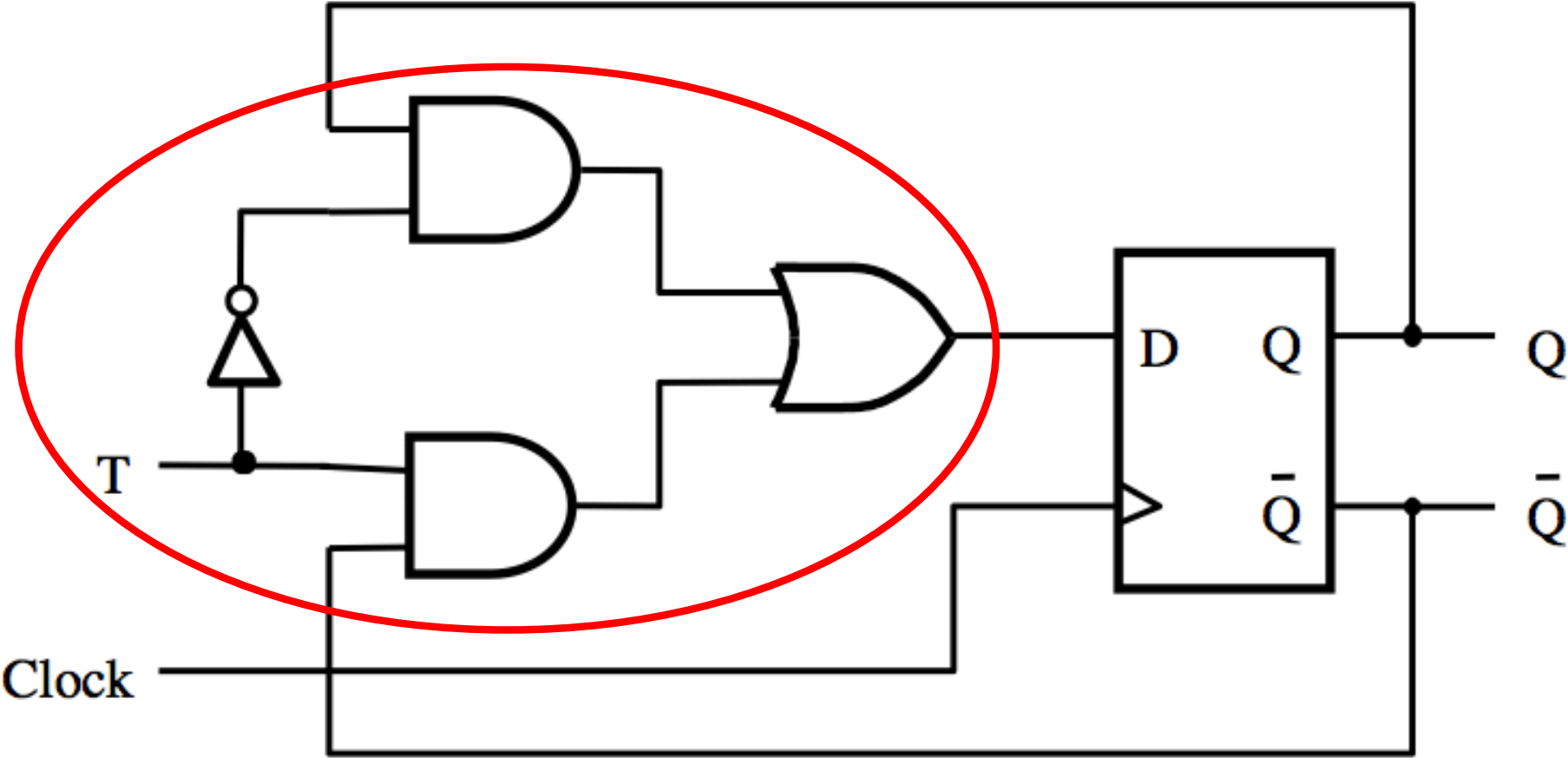
T Flip-Flop



Positive-edge-triggered
D Flip-Flop

[Figure 5.15a from the textbook]

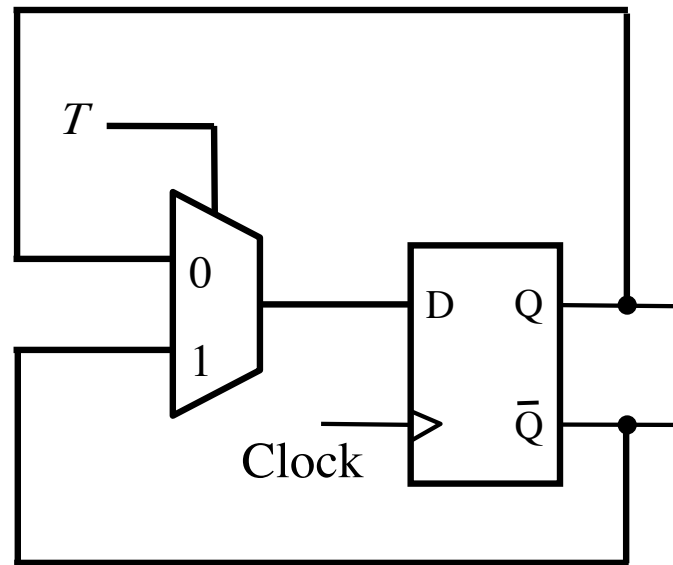
T Flip-Flop



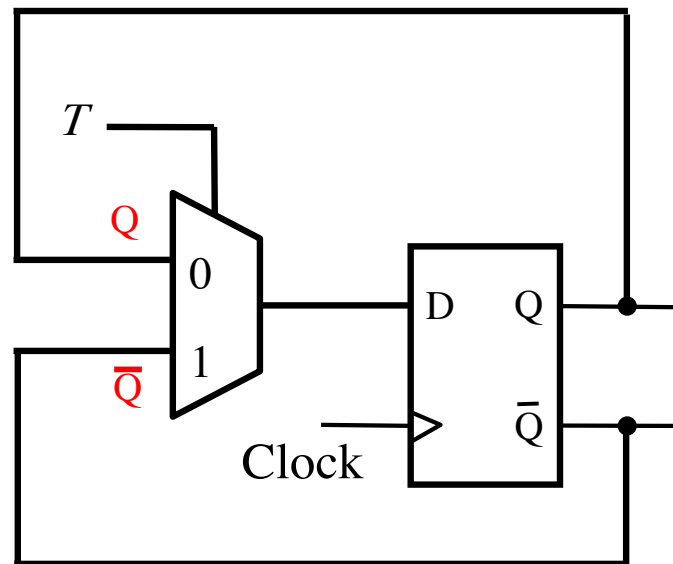
2-to-1 Mux

[Figure 5.15a from the textbook]

Another Way to Draw a T Flip-Flop

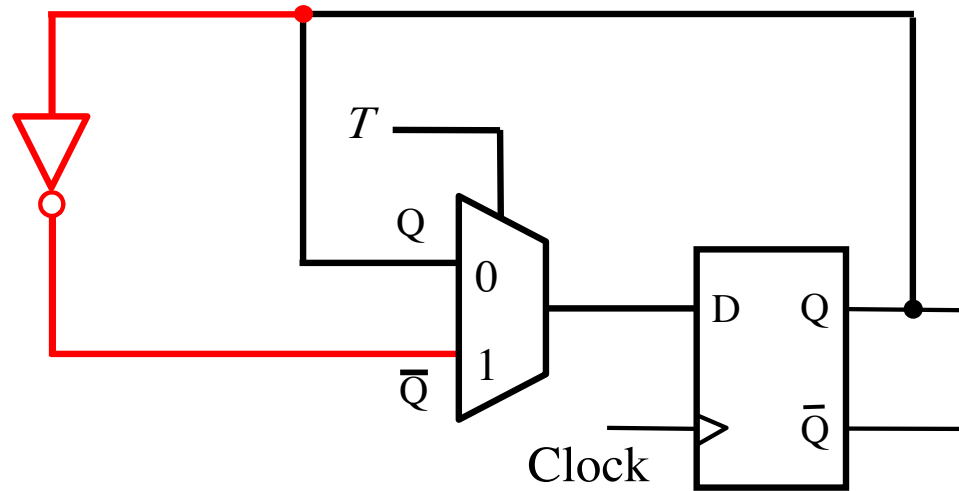


Another Way to Draw a T Flip-Flop

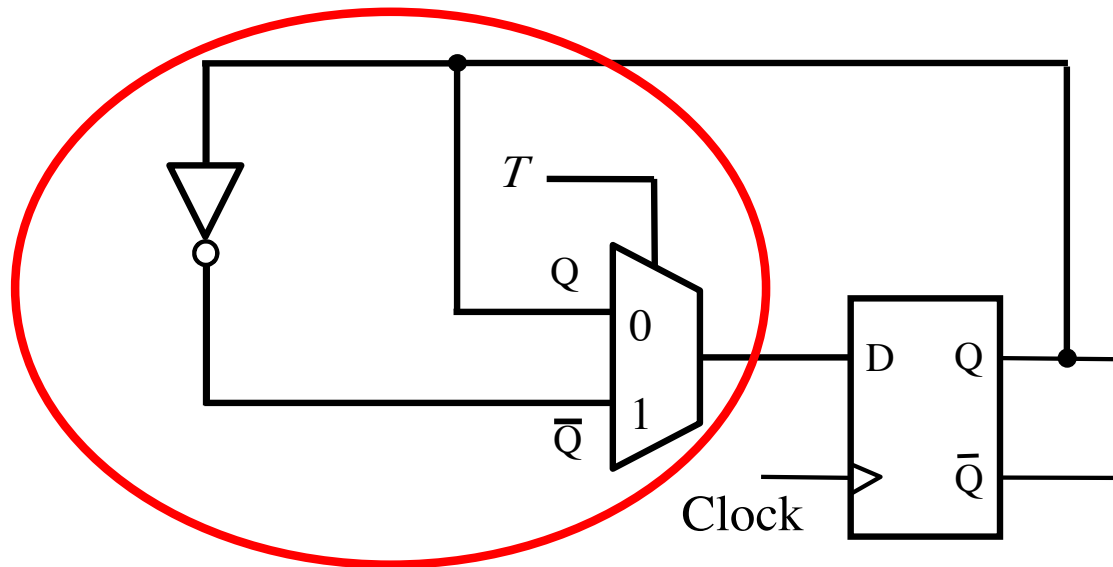


Note that the two inputs to the multiplexer are inverses of each other.

Another Way to Draw This

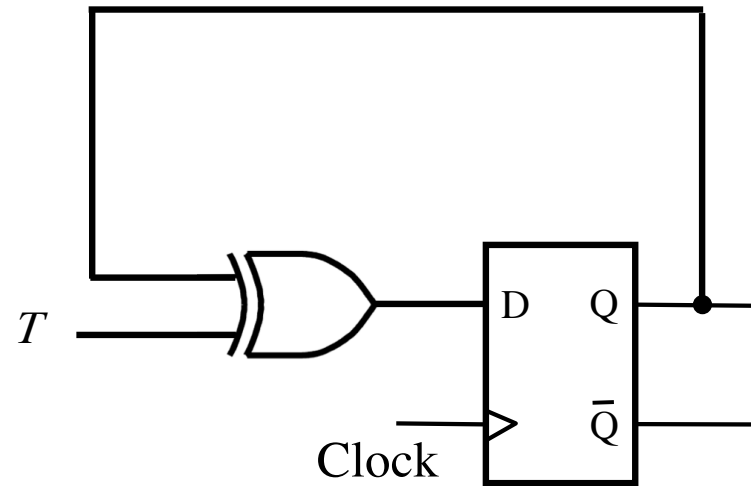


Another Way to Draw This



XOR

Yet Another Way to Draw a T Flip-Flop



T Flip-Flop

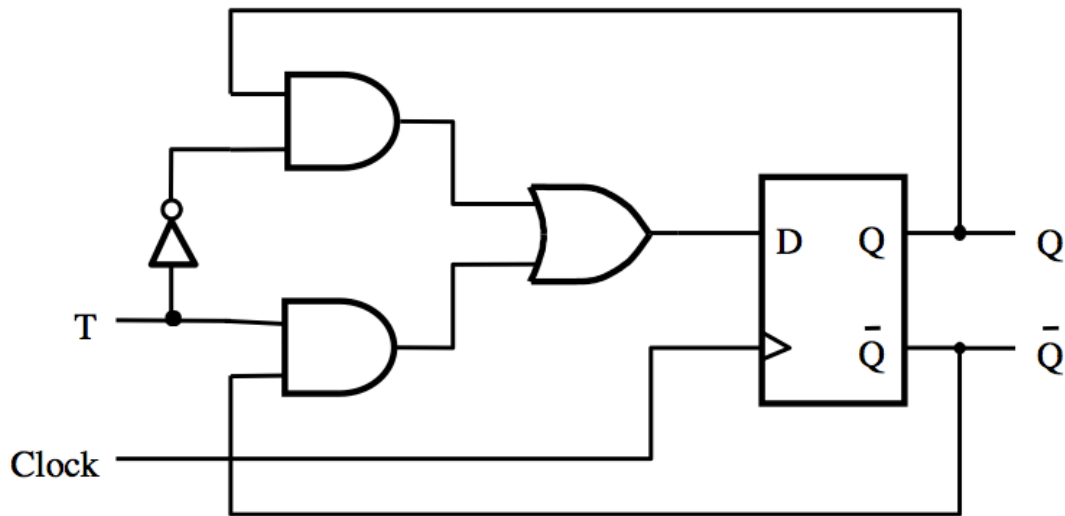
(how it works)

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.

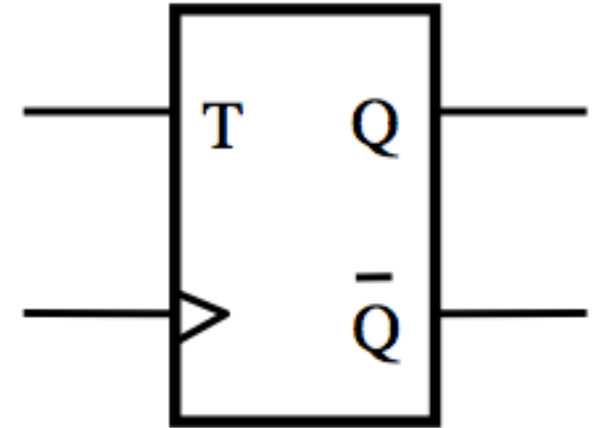
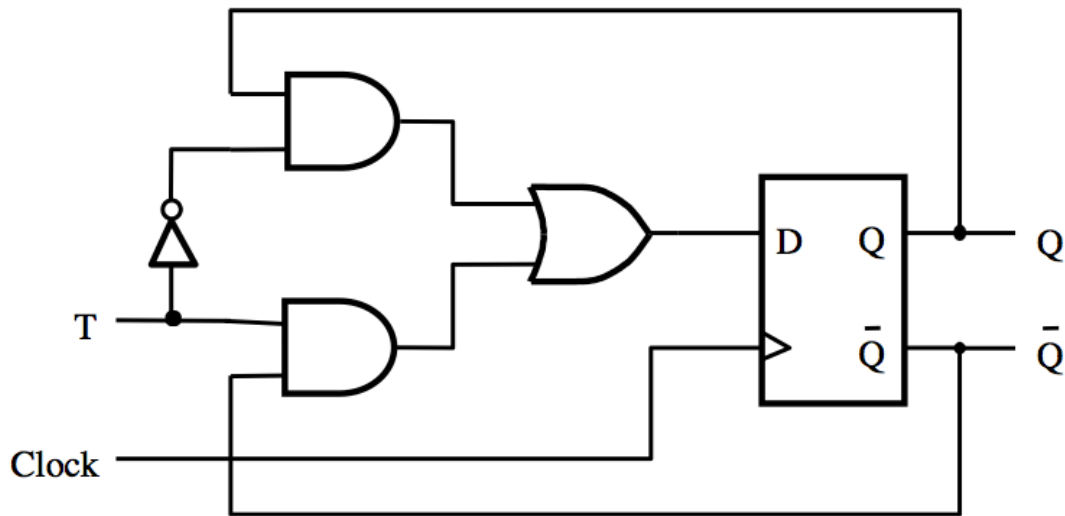
T Flip-Flop (circuit and truth table)



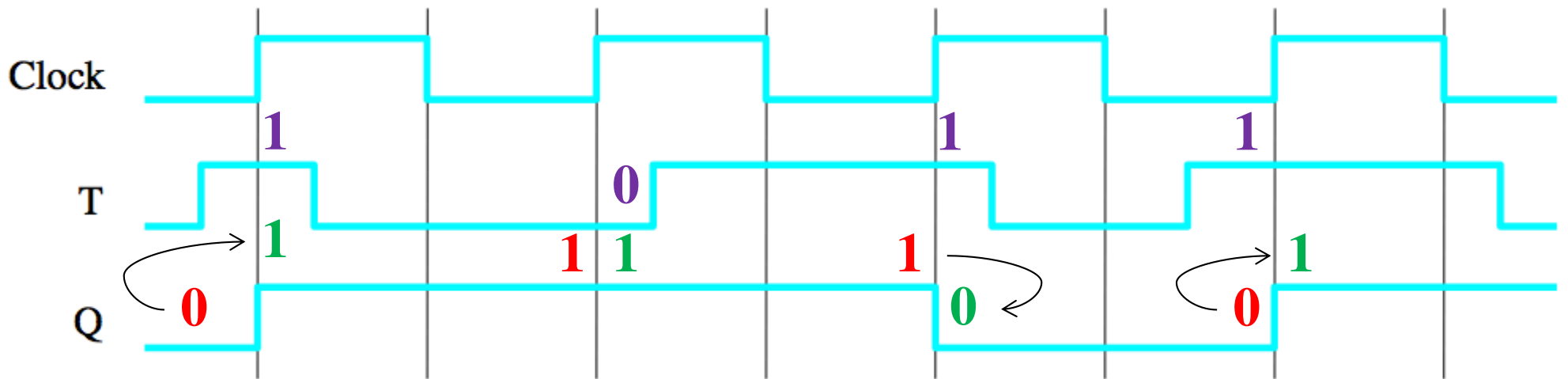
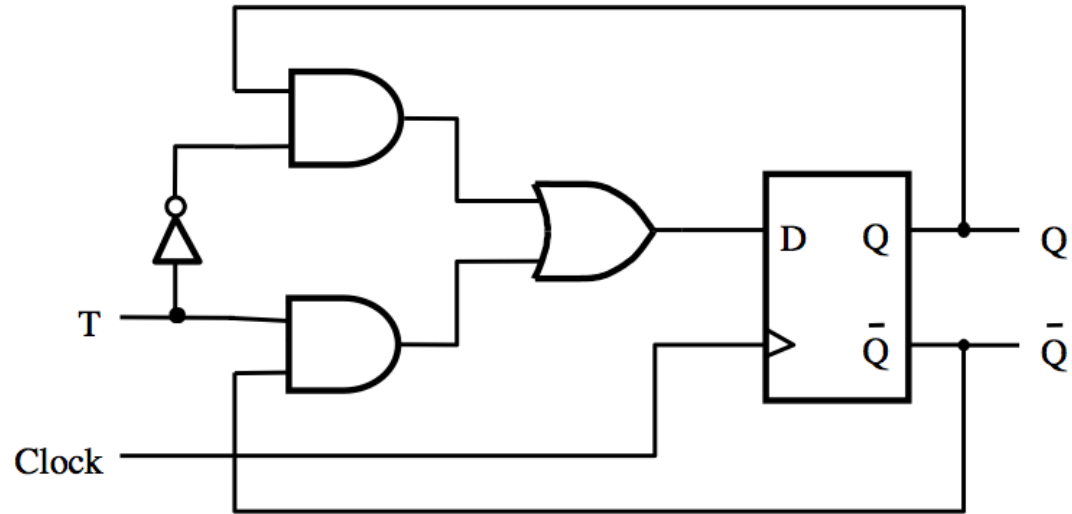
T	$Q(t+1)$
0	$Q(t)$
1	$\overline{Q(t)}$

T Flip-Flop

(circuit and graphical symbol)

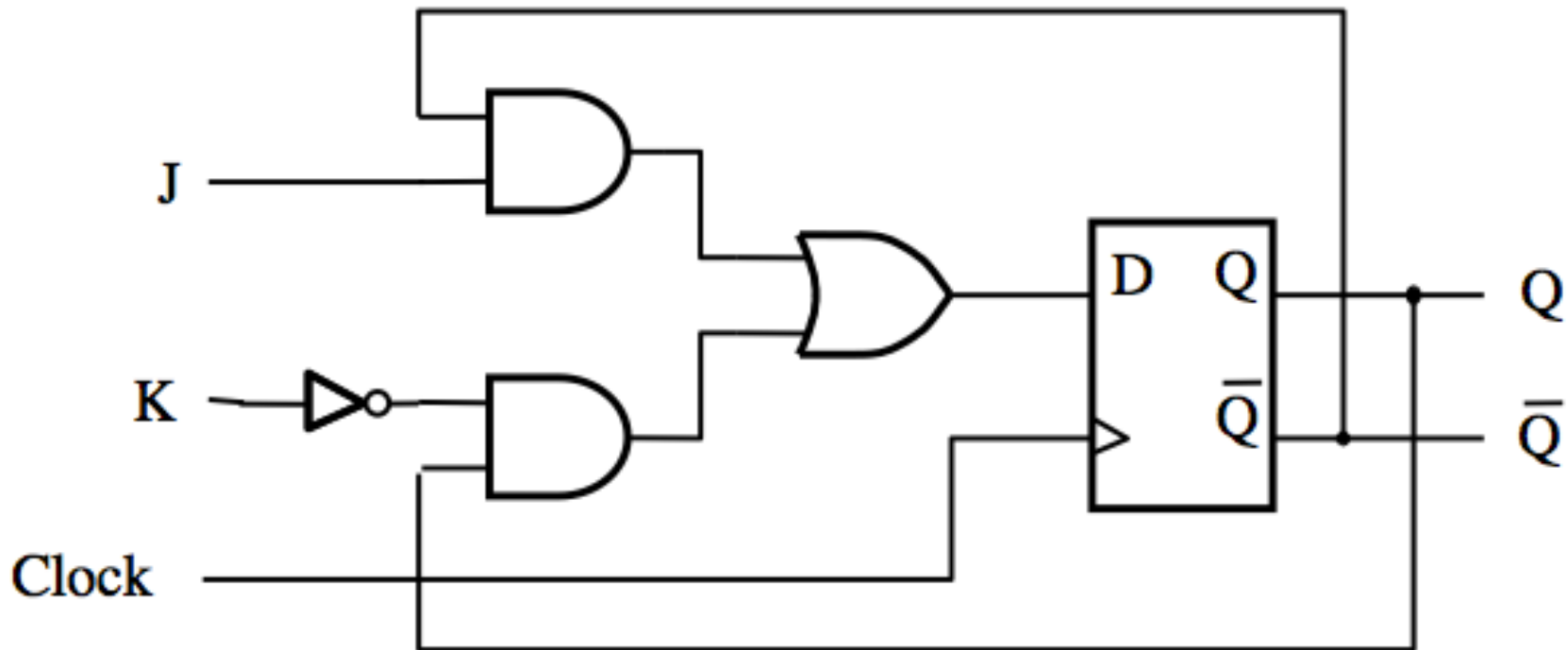


T Flip-Flop (Timing Diagram)



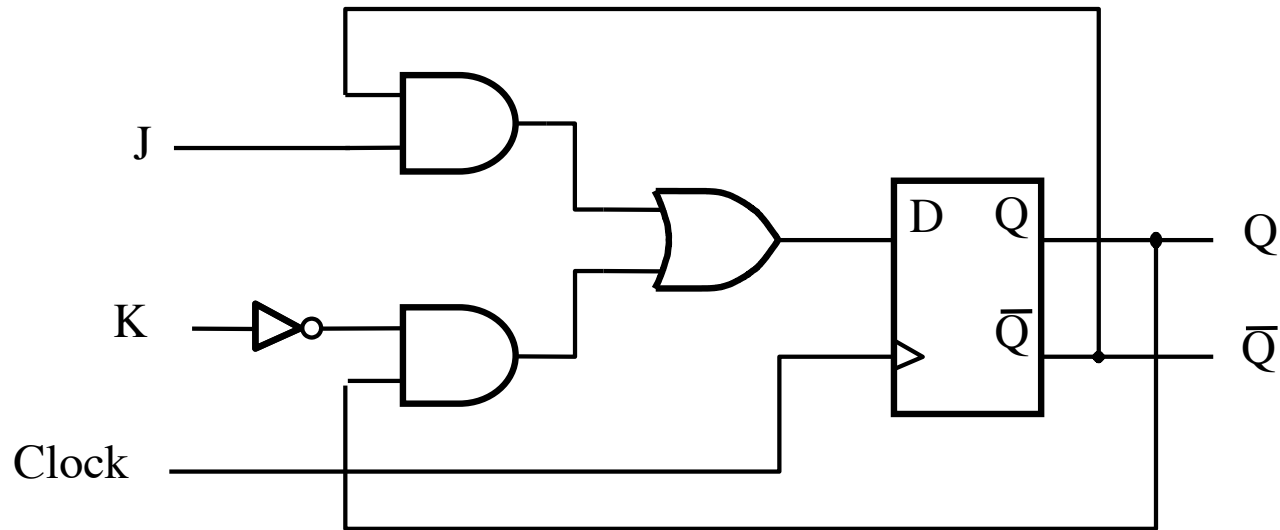
JK Flip-Flop

JK Flip-Flop



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

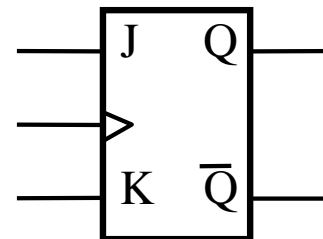
JK Flip-Flop



(a) Circuit

J	K	$Q(t+1)$	
0	0	$Q(t)$	Hold
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}(t)$	Toggle

(b) Truth table



(c) Graphical symbol

JK Flip-Flop (how it works)

A more versatile flip-flop

If $J=0$ and $K=0$ it stays in the same state

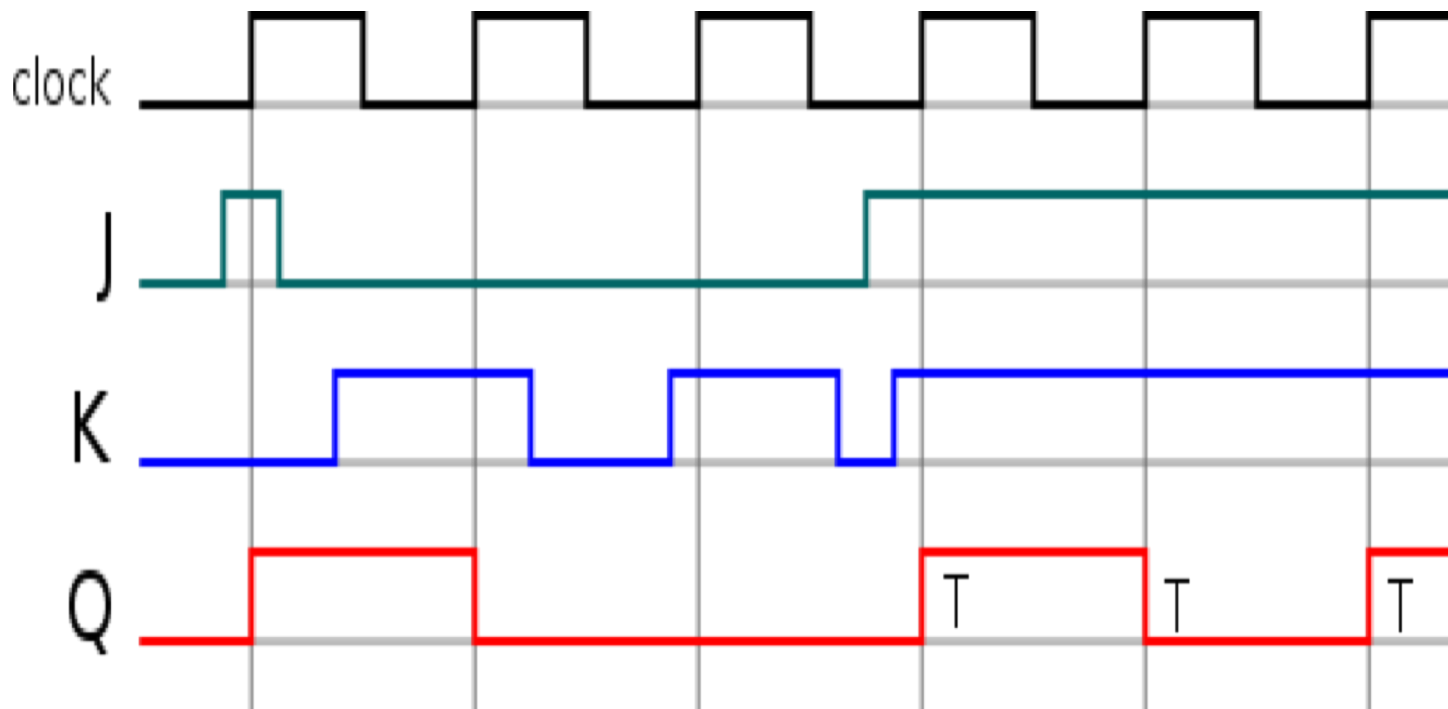
If $J=1$ and $K=0$ it sets the output Q to 1

If $J=0$ and $K=1$ it resets the output Q to 0

If $J=1$ and $K=1$ it toggles the output Q

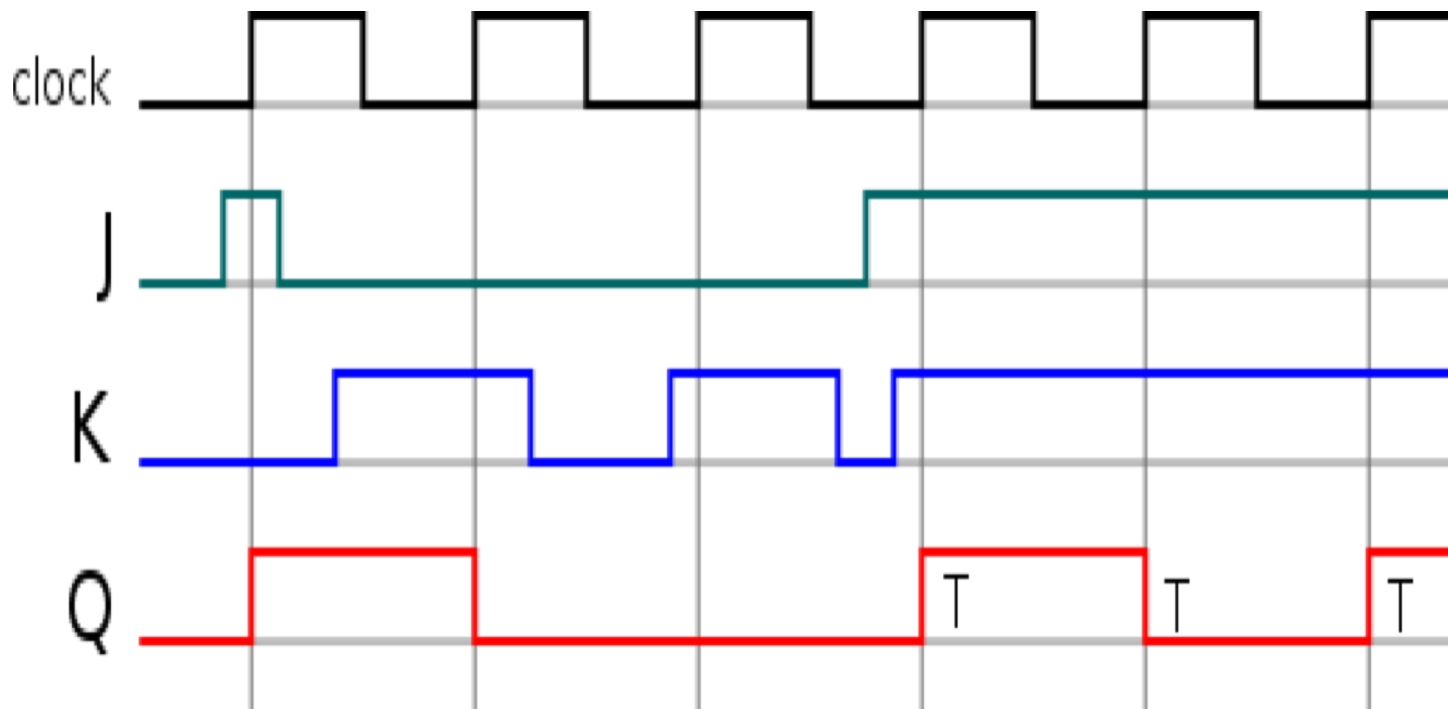
If $J=K$ then it behaves like a T flip-flop

JK Flip-Flop (timing diagram)



J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

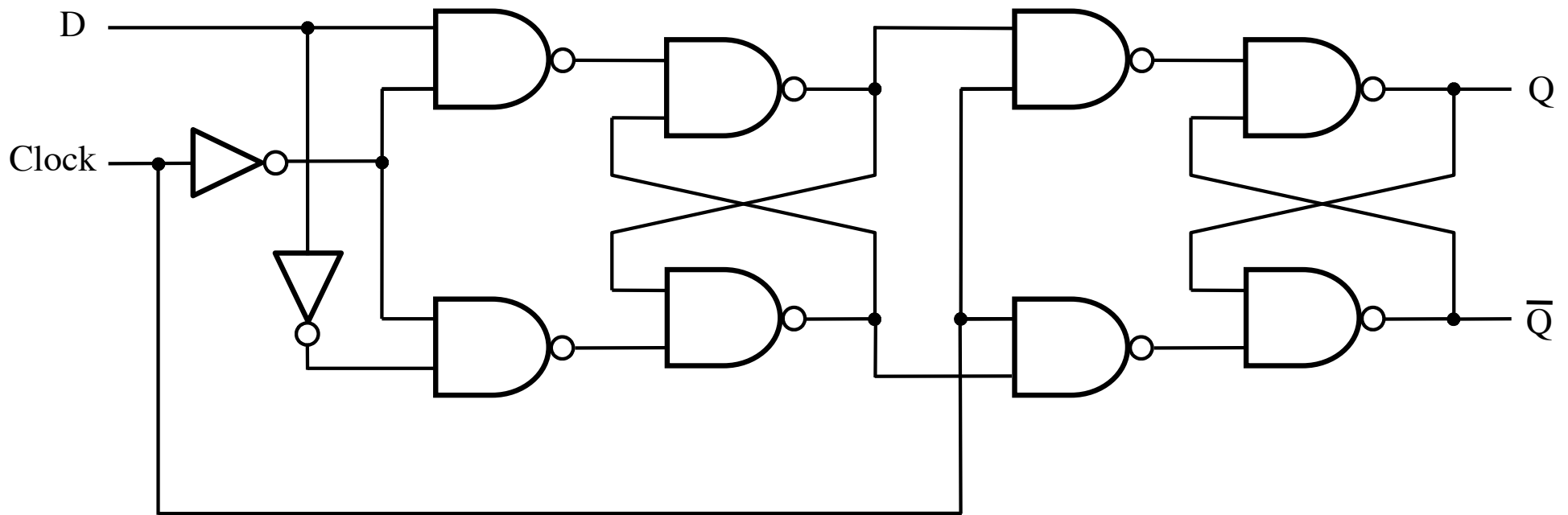
JK Flip-Flop (timing diagram)



		set	reset	
J	K			$Q(t+1)$
0	0			$Q(t)$
0	1			0
1	0			1
1	1			$\bar{Q}(t)$

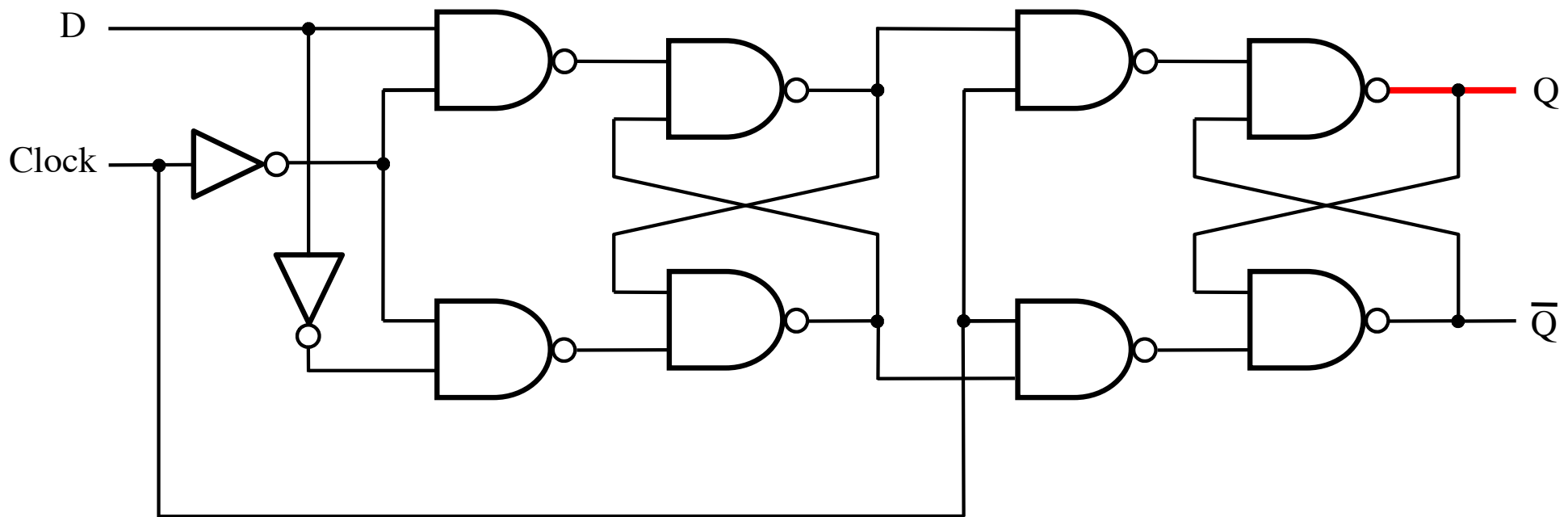
Complete Wiring Diagrams

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

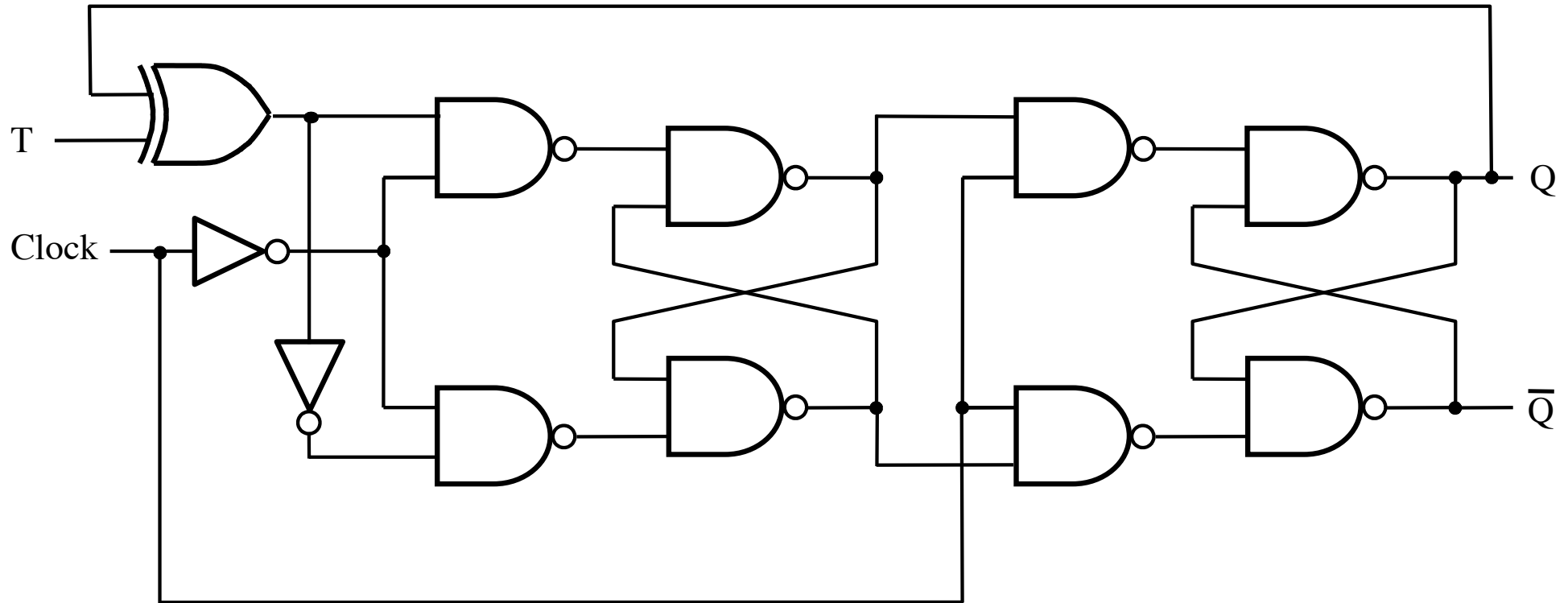


The Complete Wiring Diagram for a Positive-Edge-Triggered D Flip-Flop

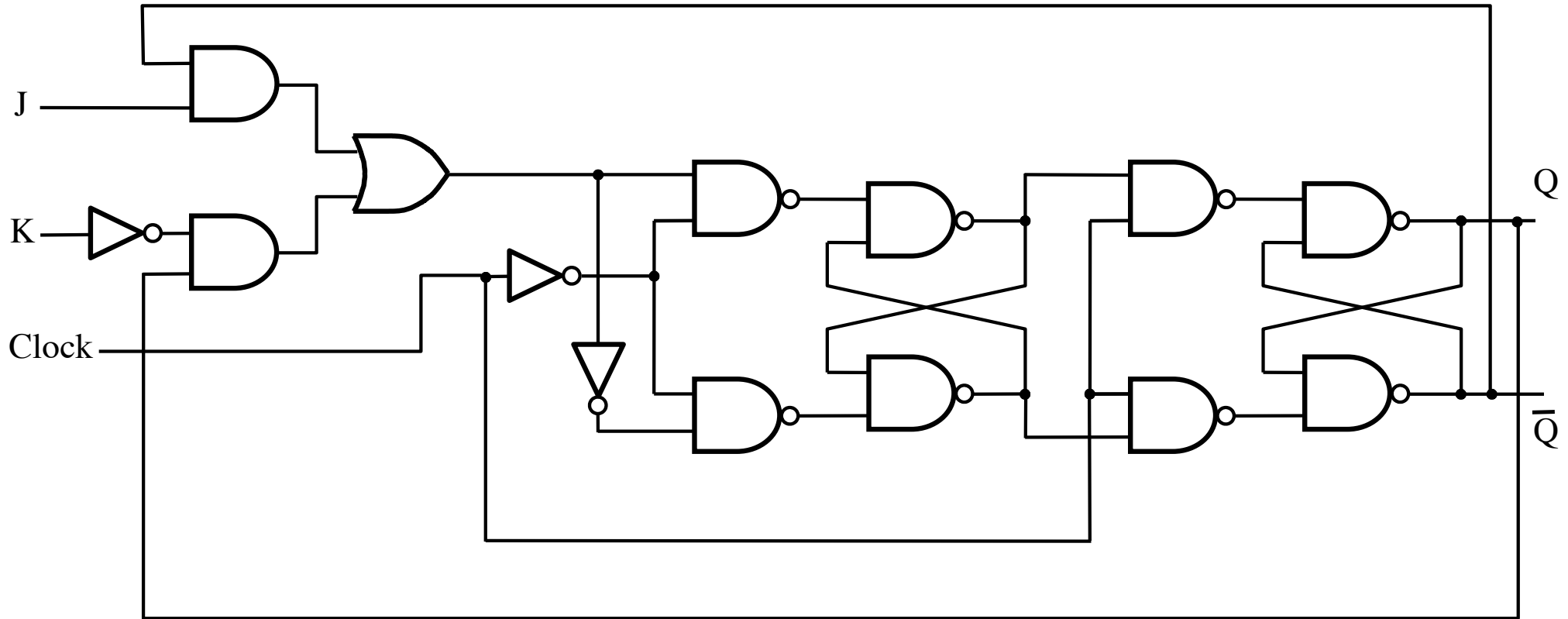
We need all of this just to store 1 bit!



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



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



**Complete the Timing diagrams
(for positive-edge-triggered F-F)**

D

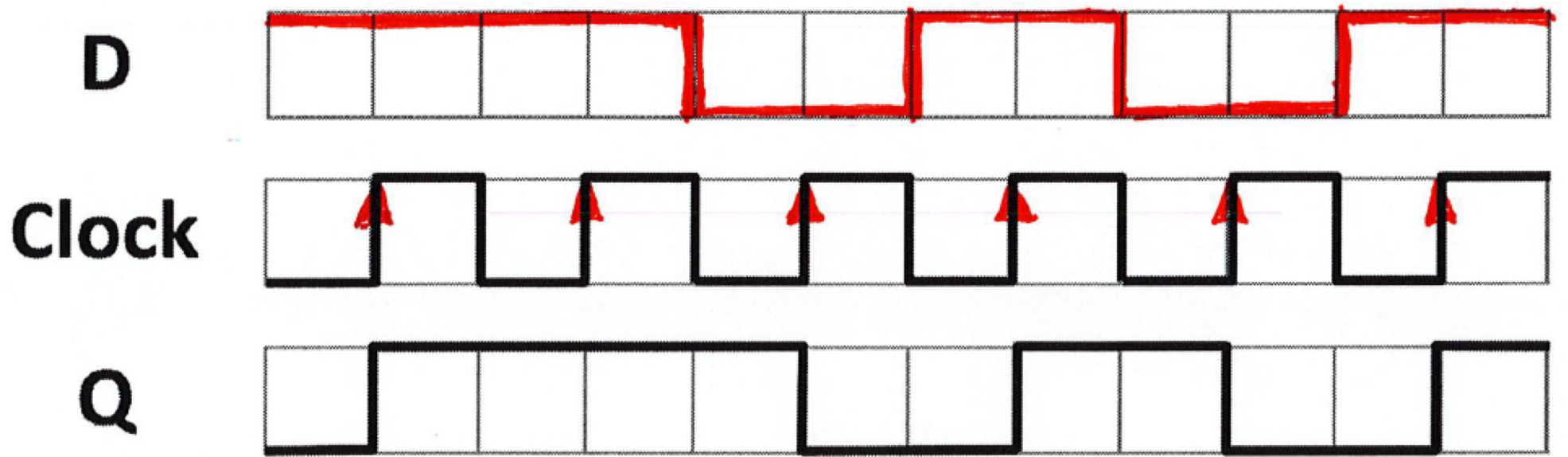


Clock



Q





T

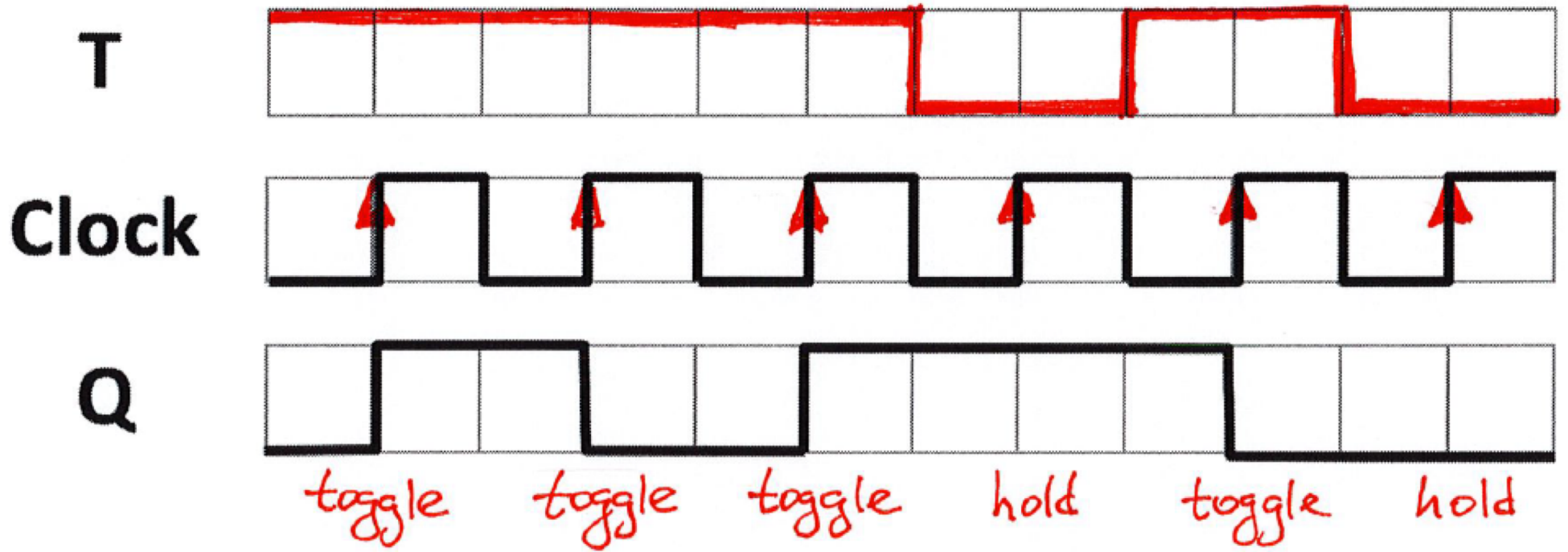


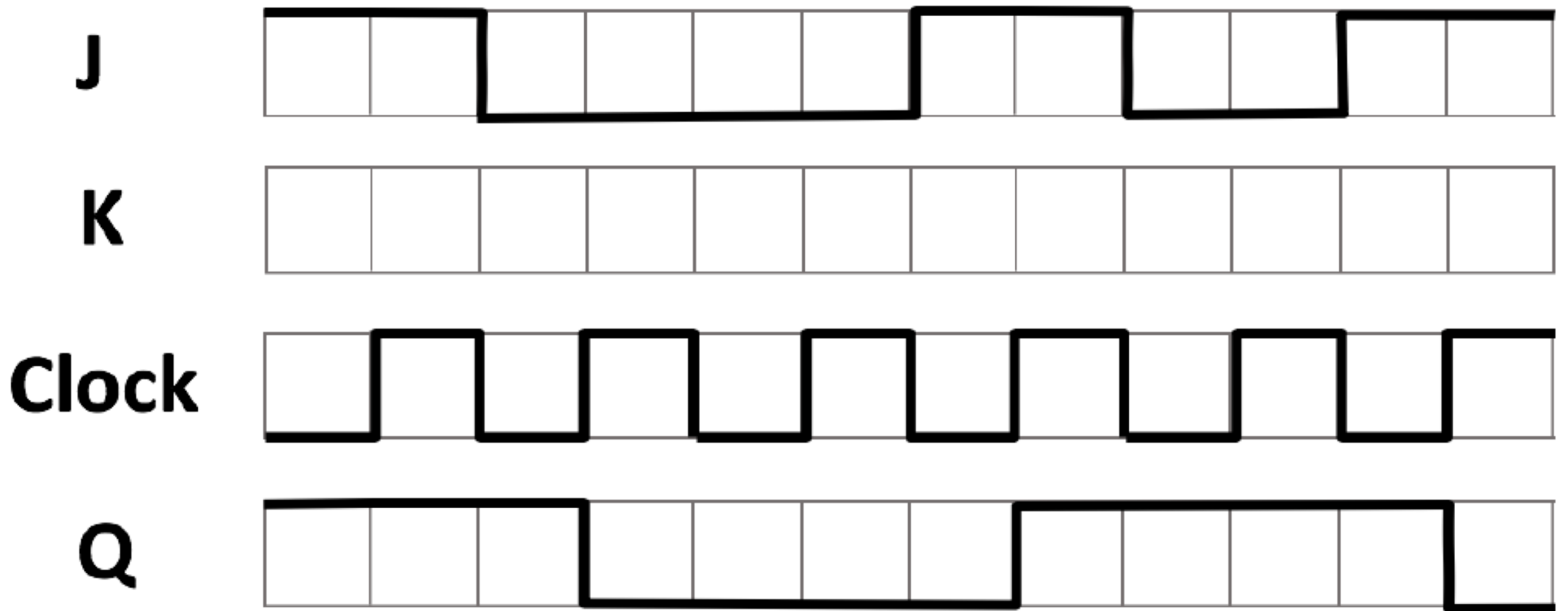
Clock

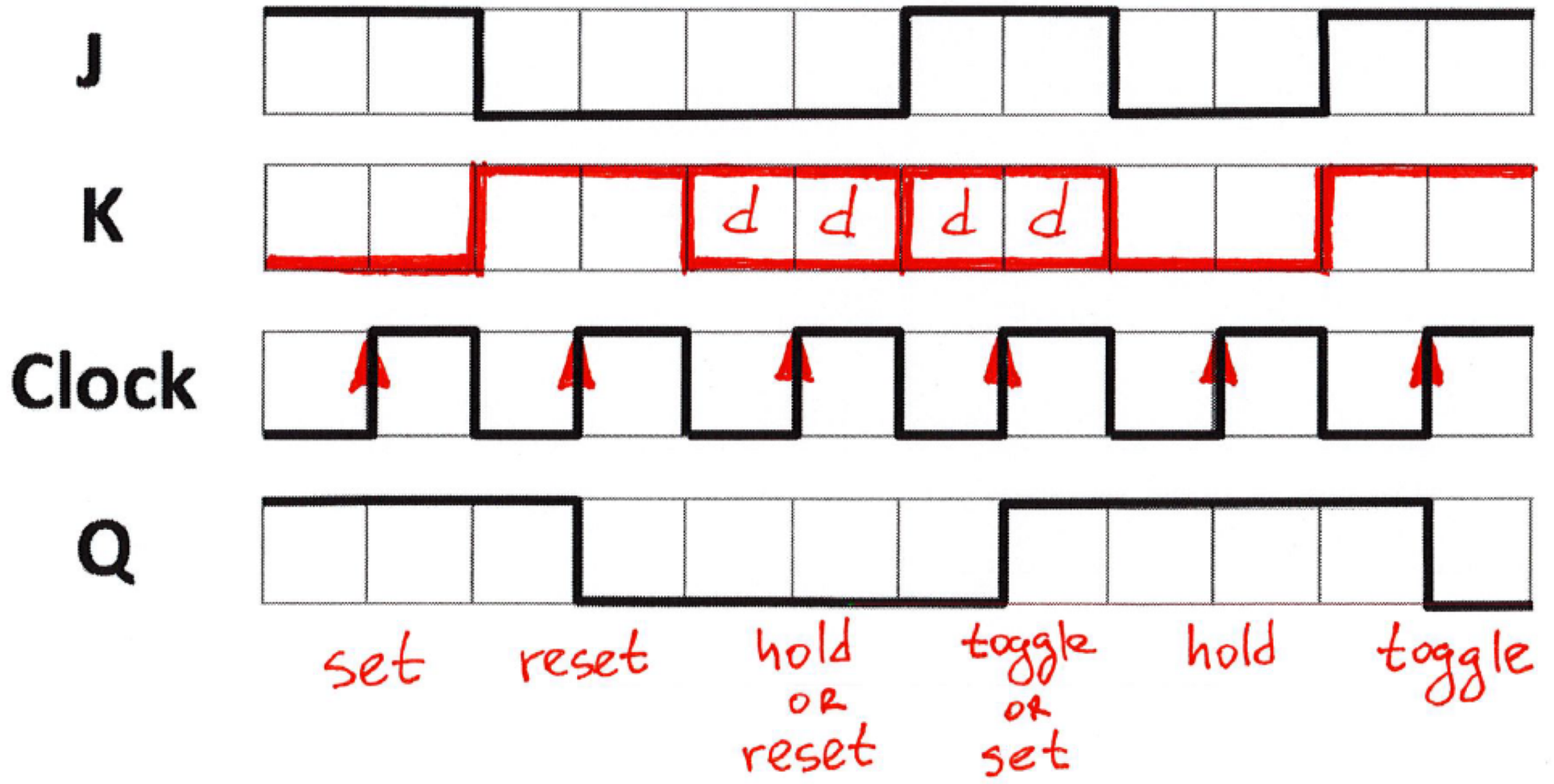


Q









Registers

Register (Definition)

An n-bit structure consisting of flip-flops.

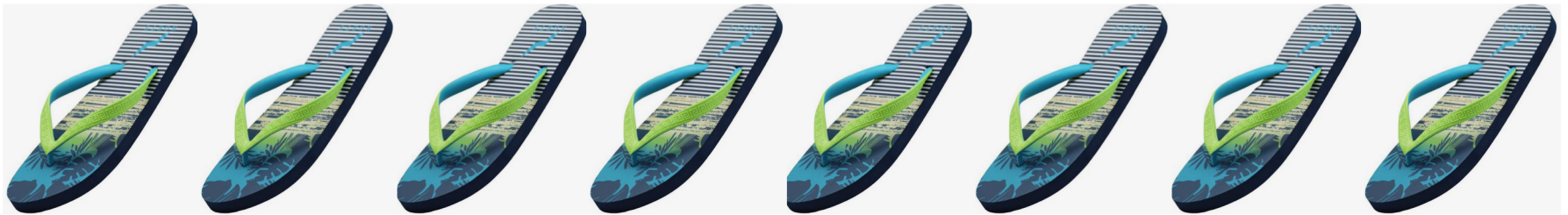
Flip-Flop



4-bit Register

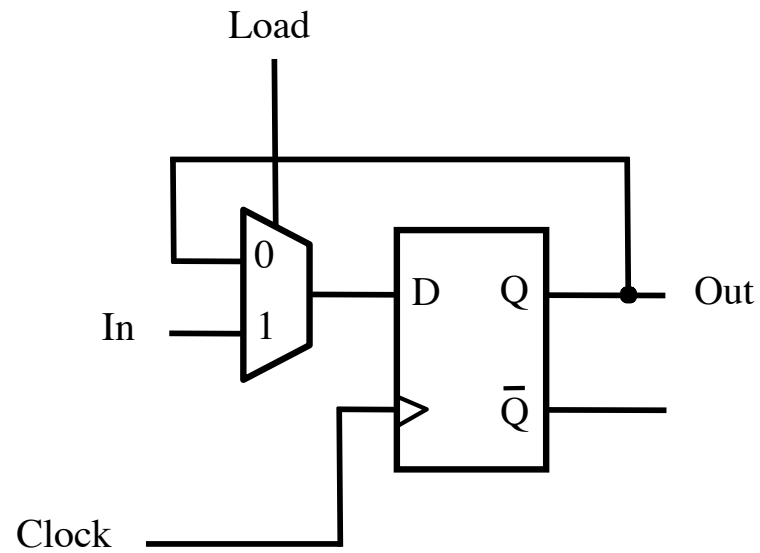


8-bit Register

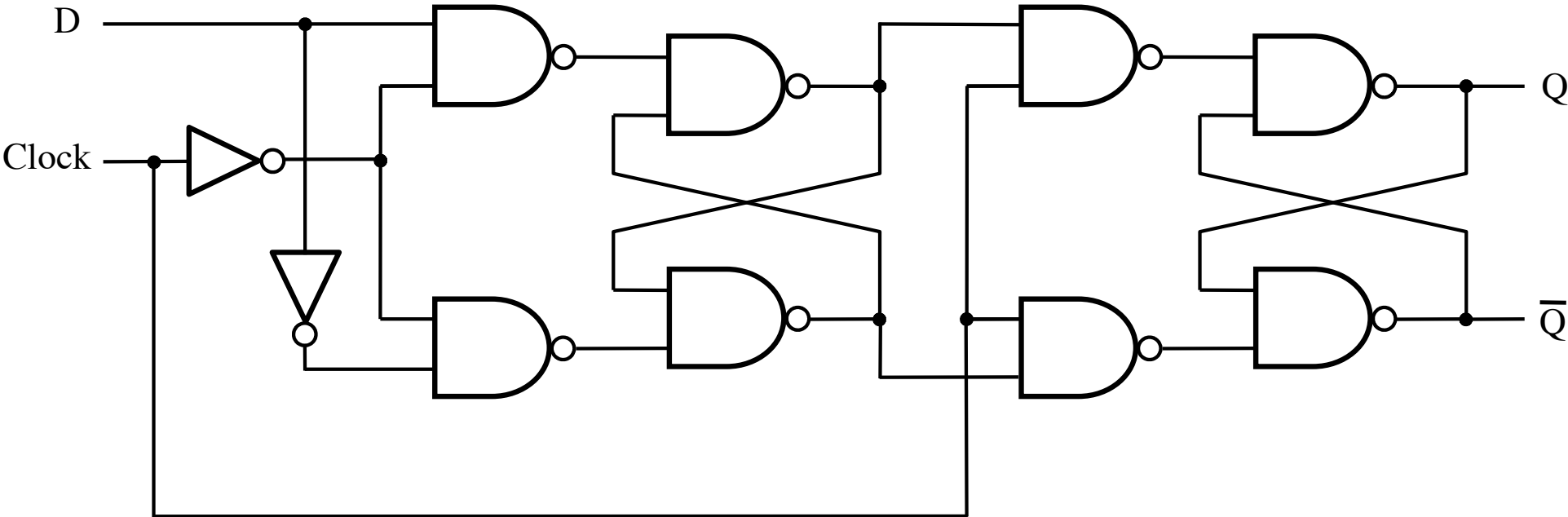


Parallel-Access Register

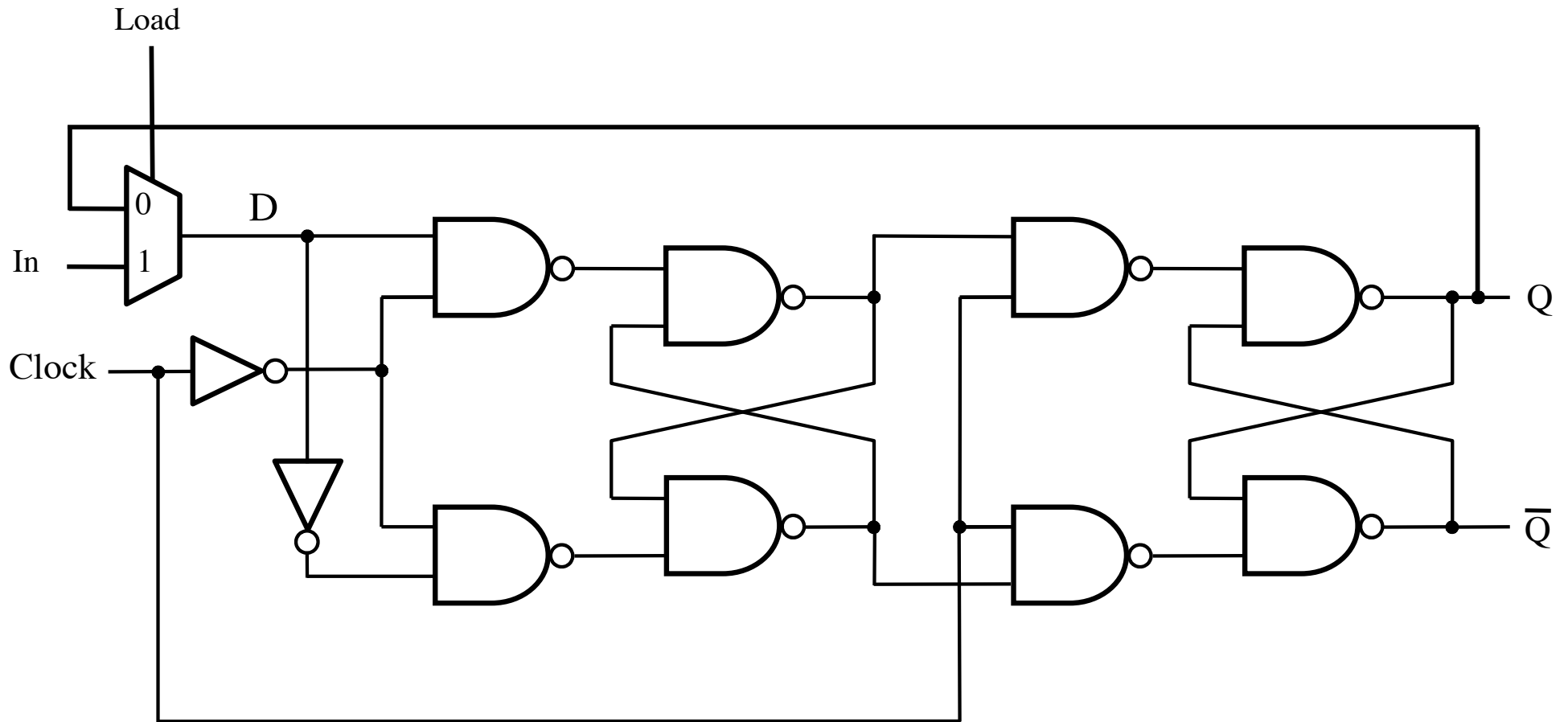
1-Bit Parallel-Access Register



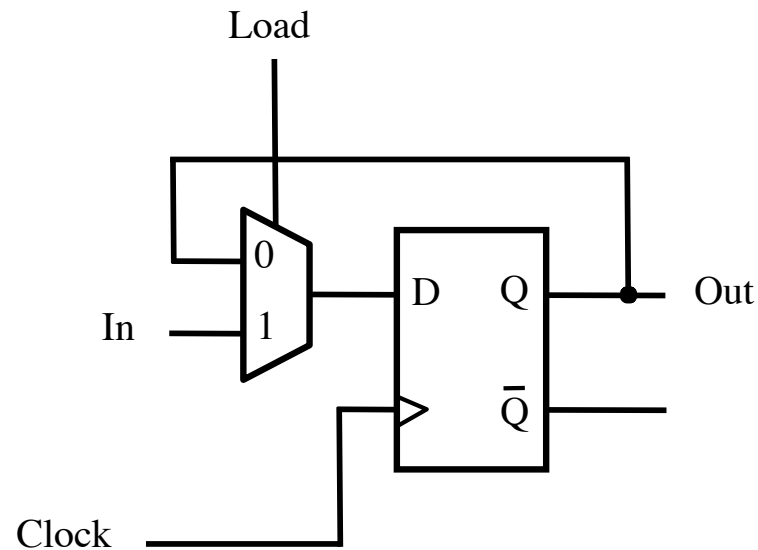
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register



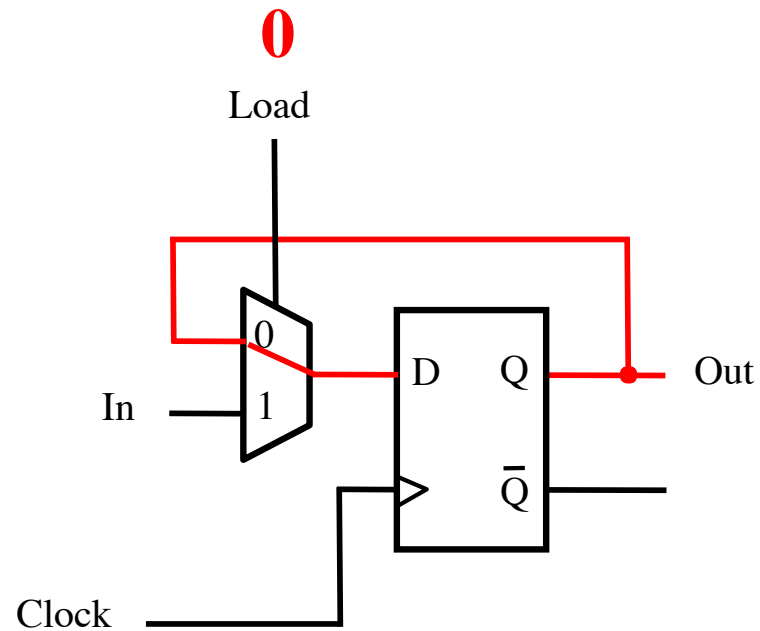
1-Bit Parallel-Access Register



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.

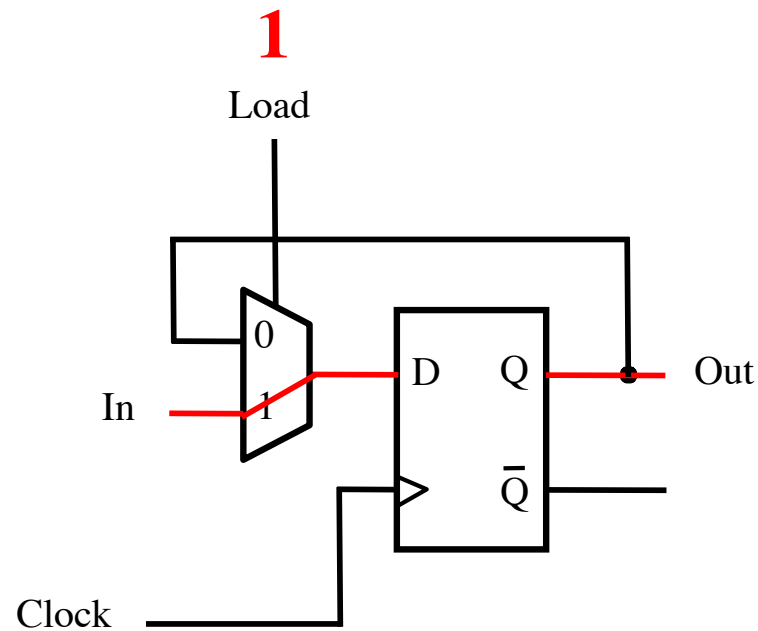
The output of this circuit is the Q output of the flip-flop.

1-Bit Parallel-Access Register



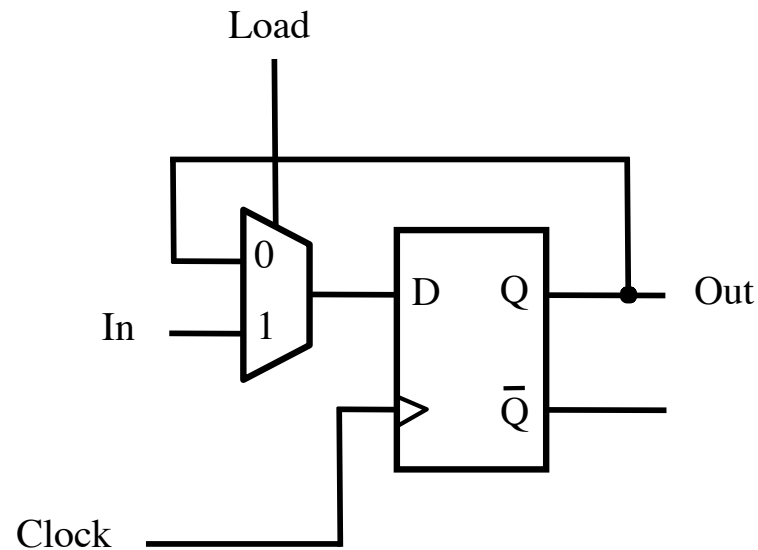
If Load = 0, then retain the old value.

1-Bit Parallel-Access Register



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

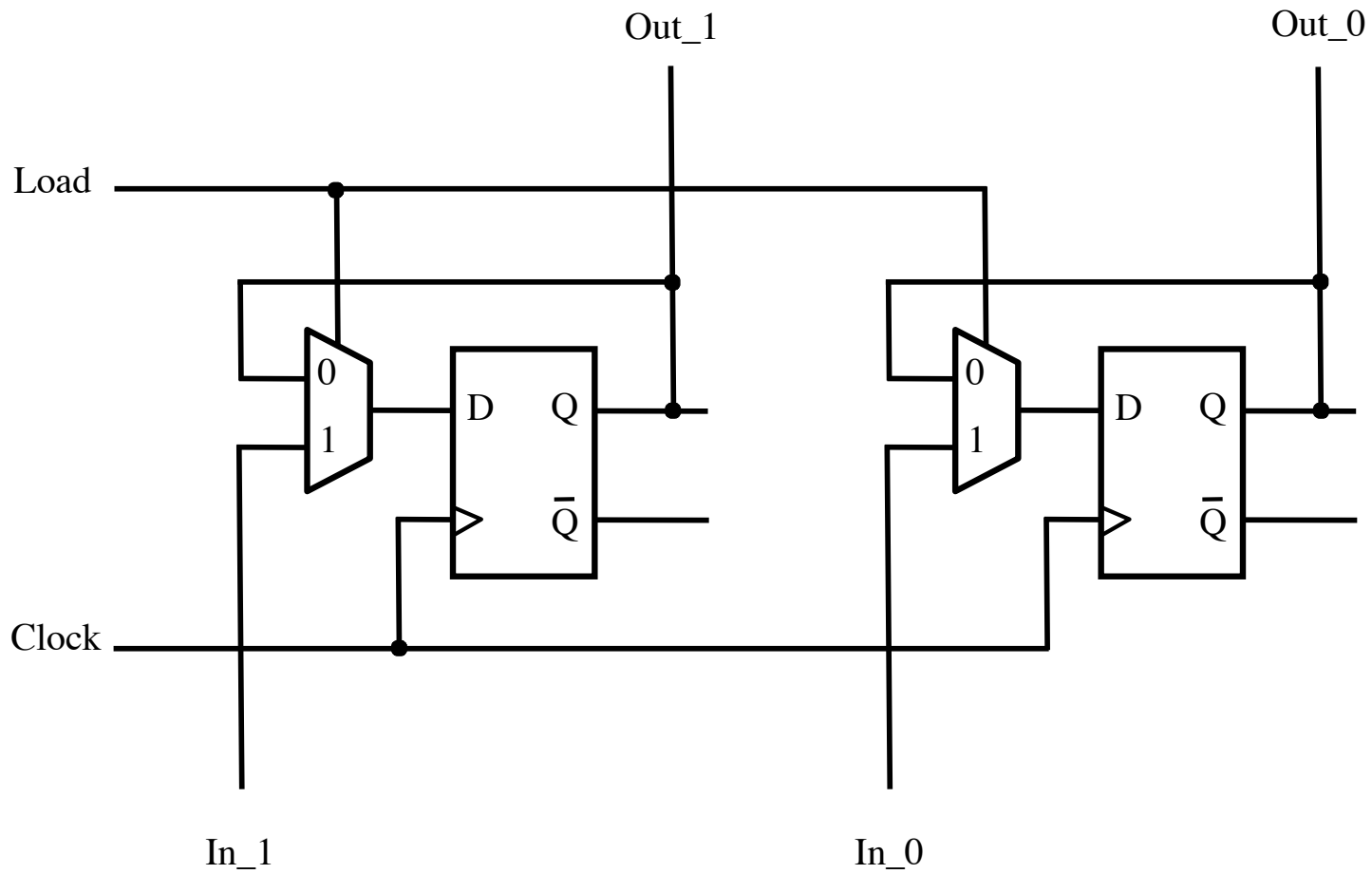
1-Bit Parallel-Access Register



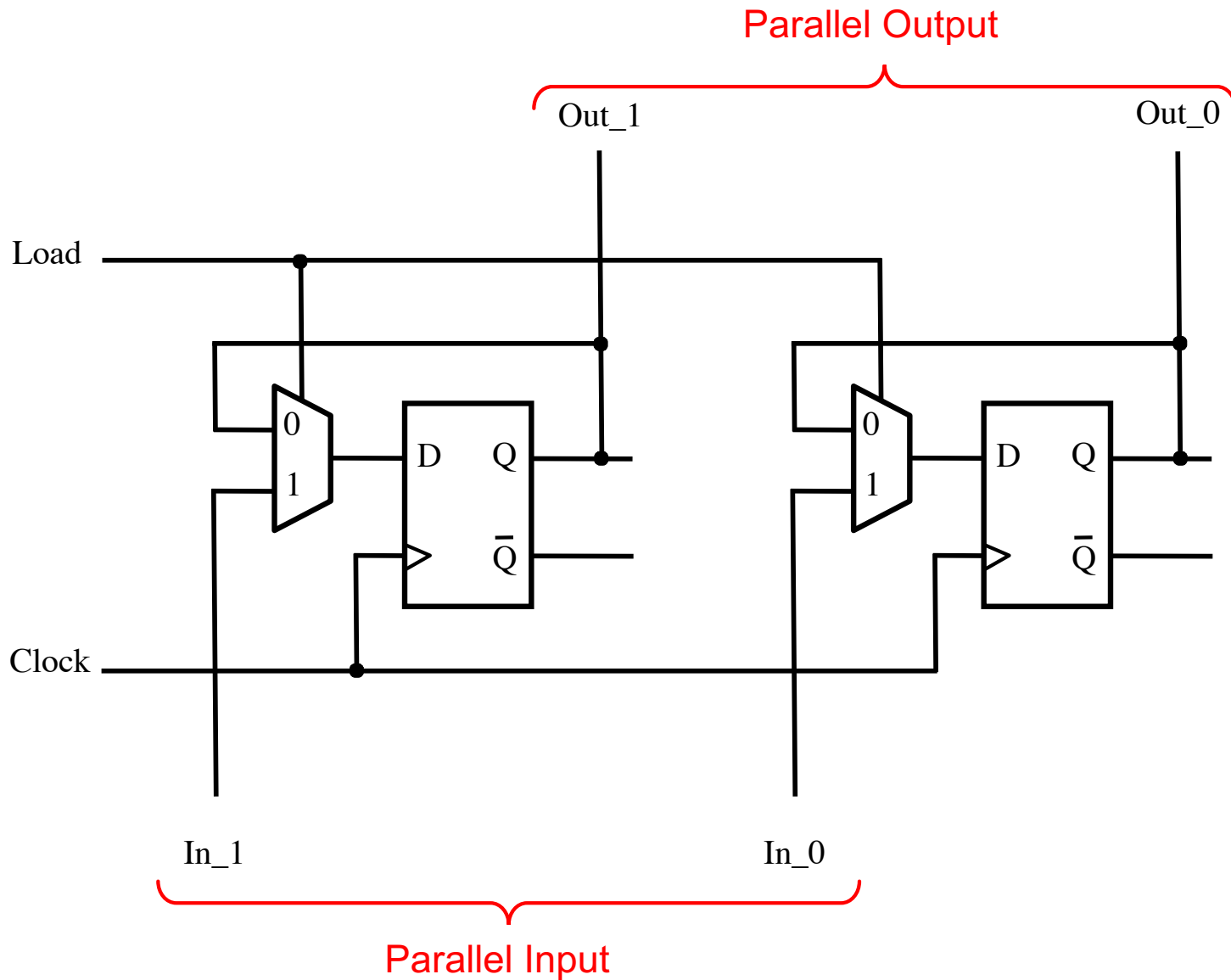
If Load = 0, then retain the old value.

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

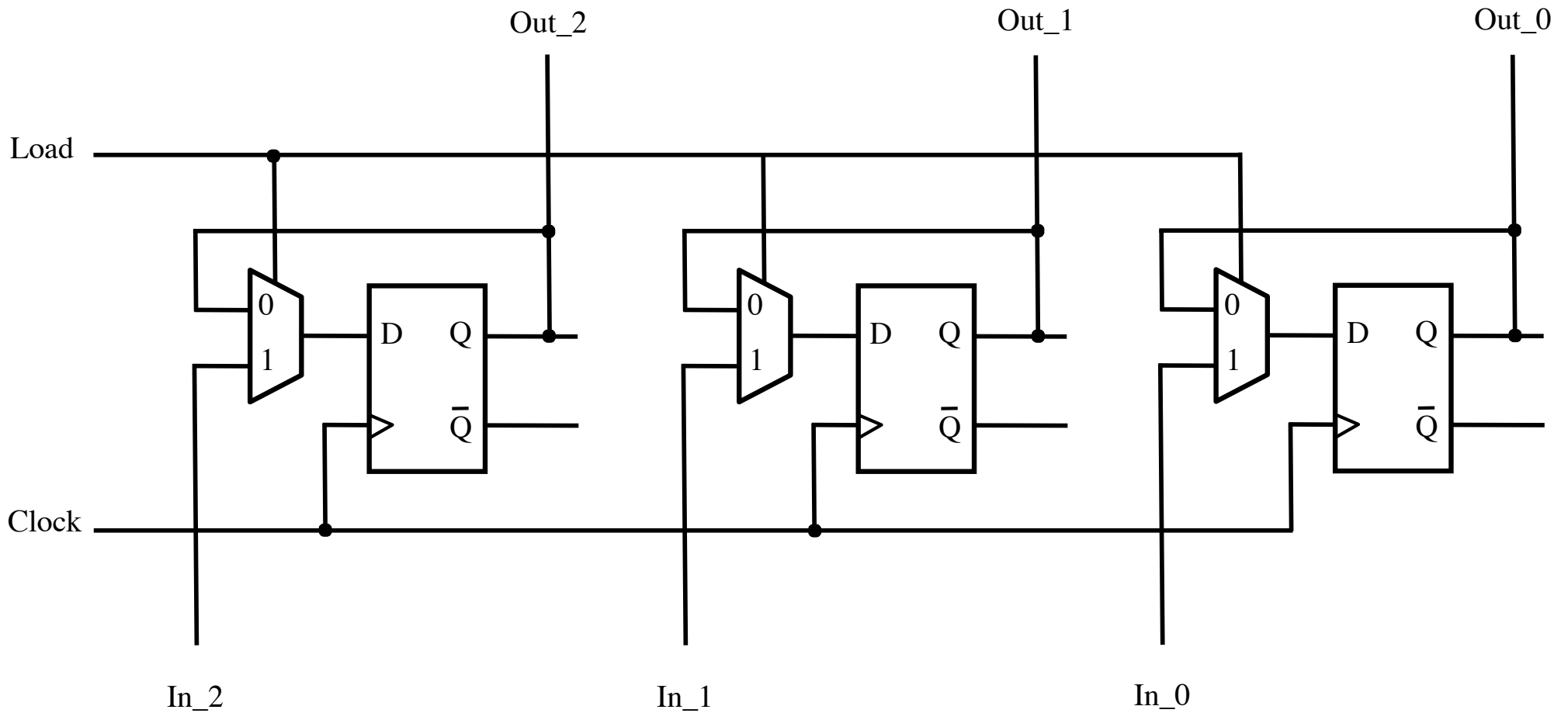
2-Bit Parallel-Access Register



2-Bit Parallel-Access Register

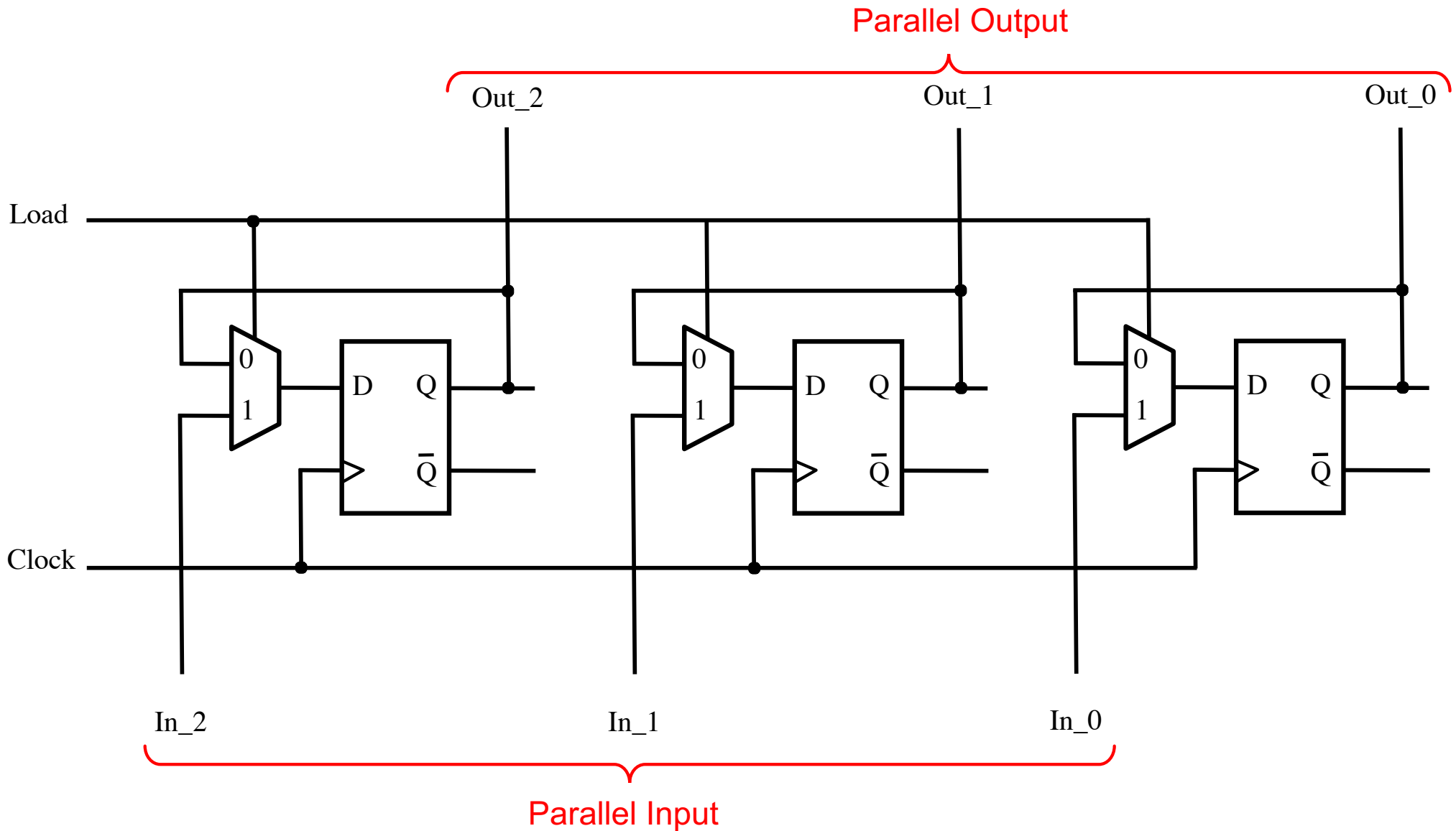


3-Bit Parallel-Access Register

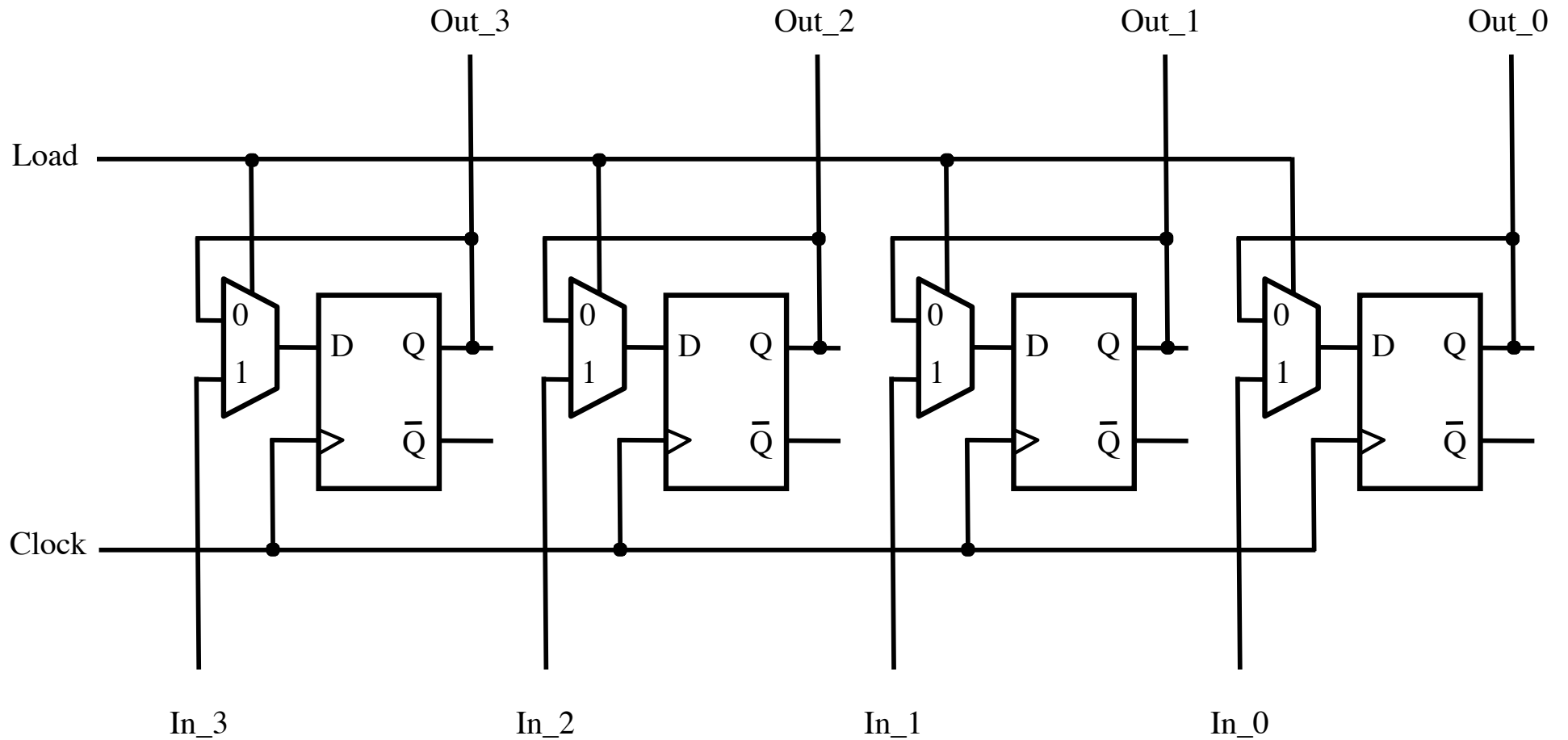


Notice that all flip-flops are on the same clock cycle.

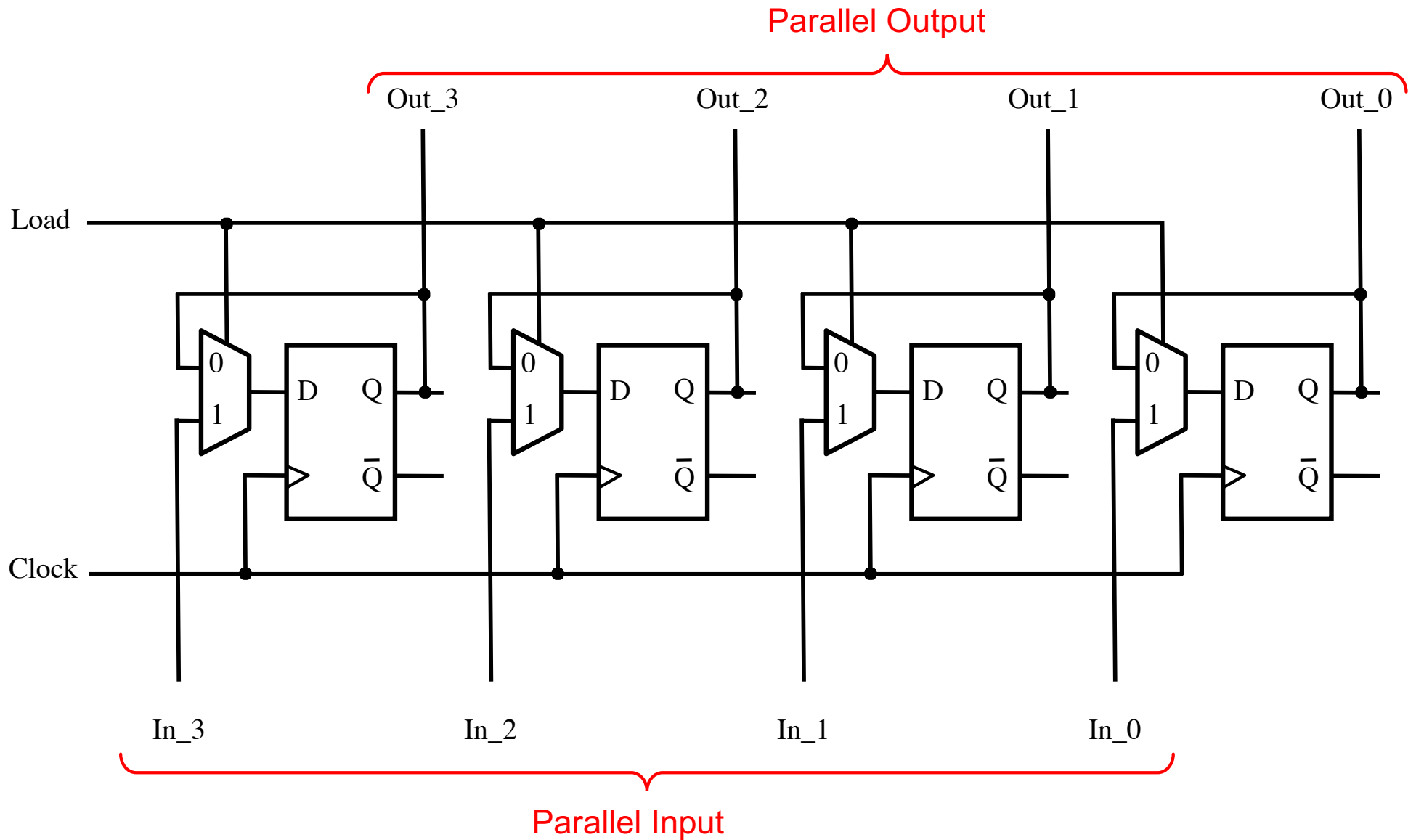
3-Bit Parallel-Access Register



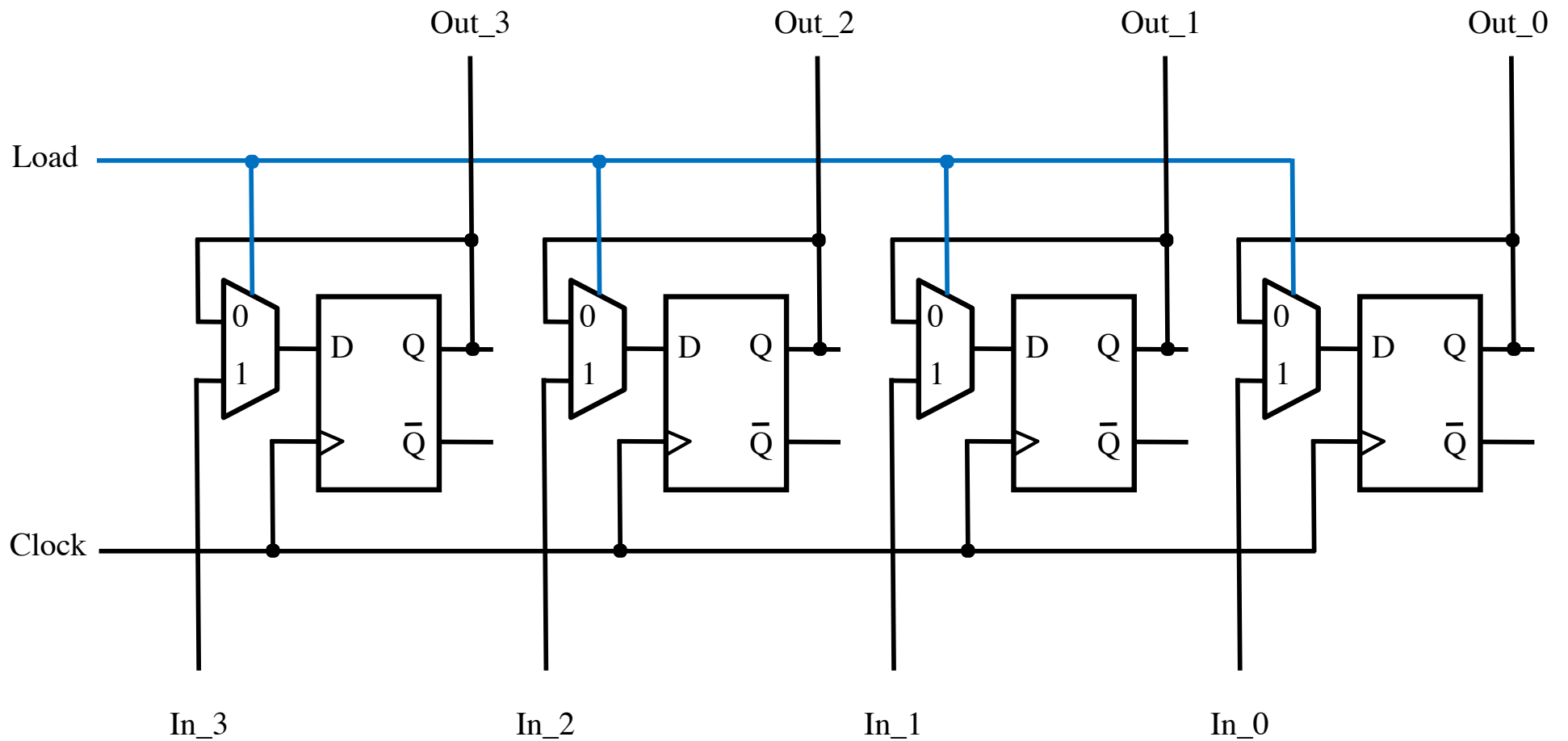
4-Bit Parallel-Access Register



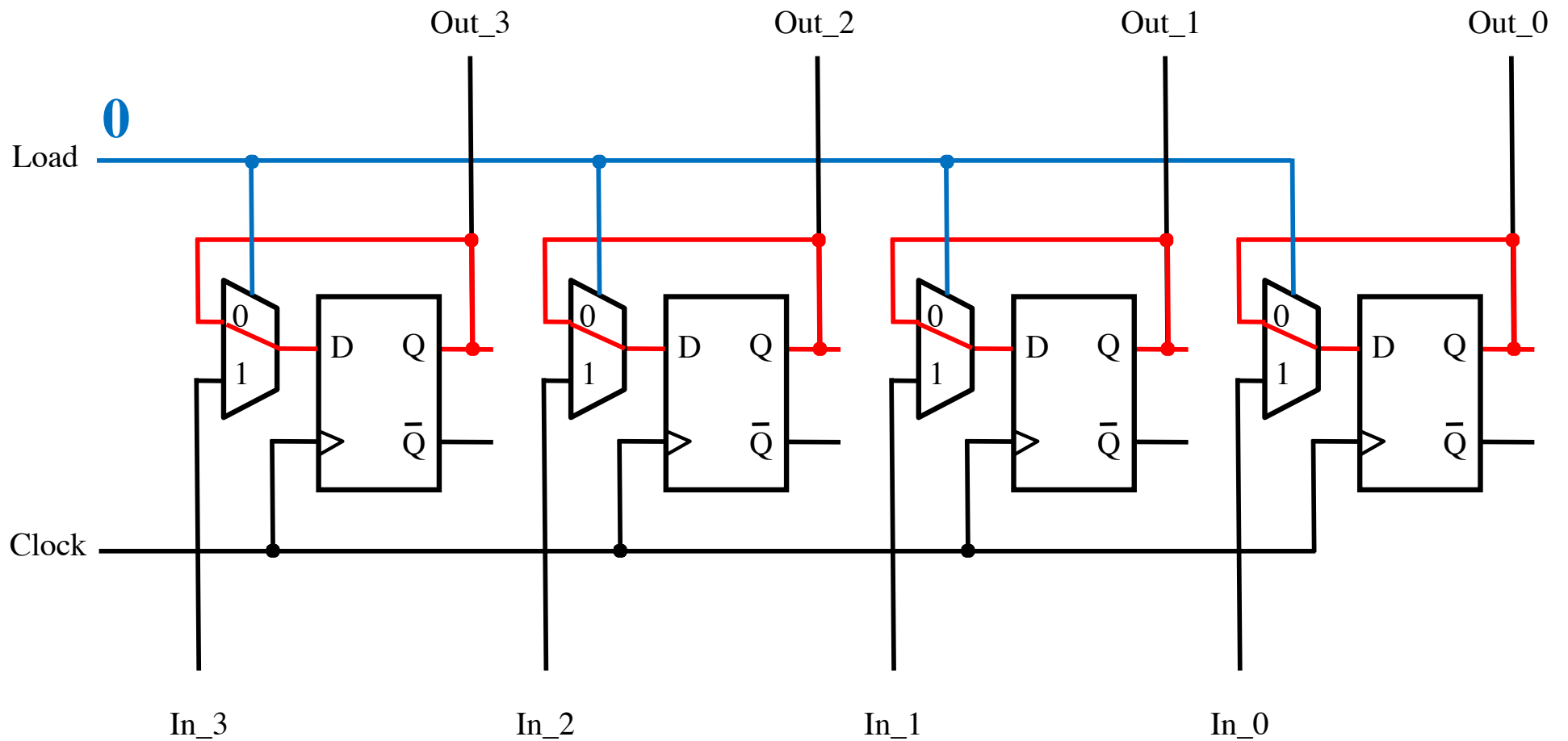
4-Bit Parallel-Access Register



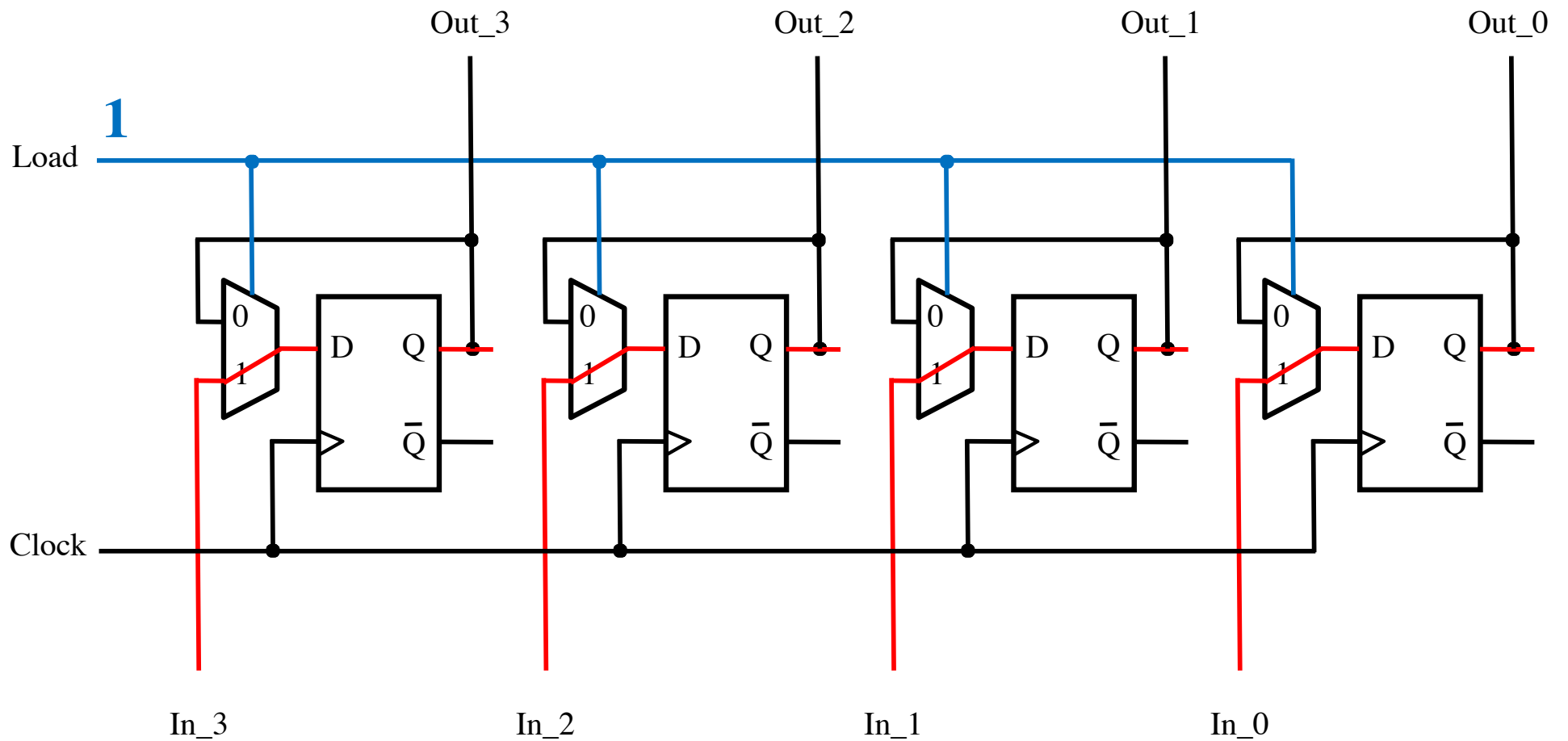
4-Bit Parallel-Access Register



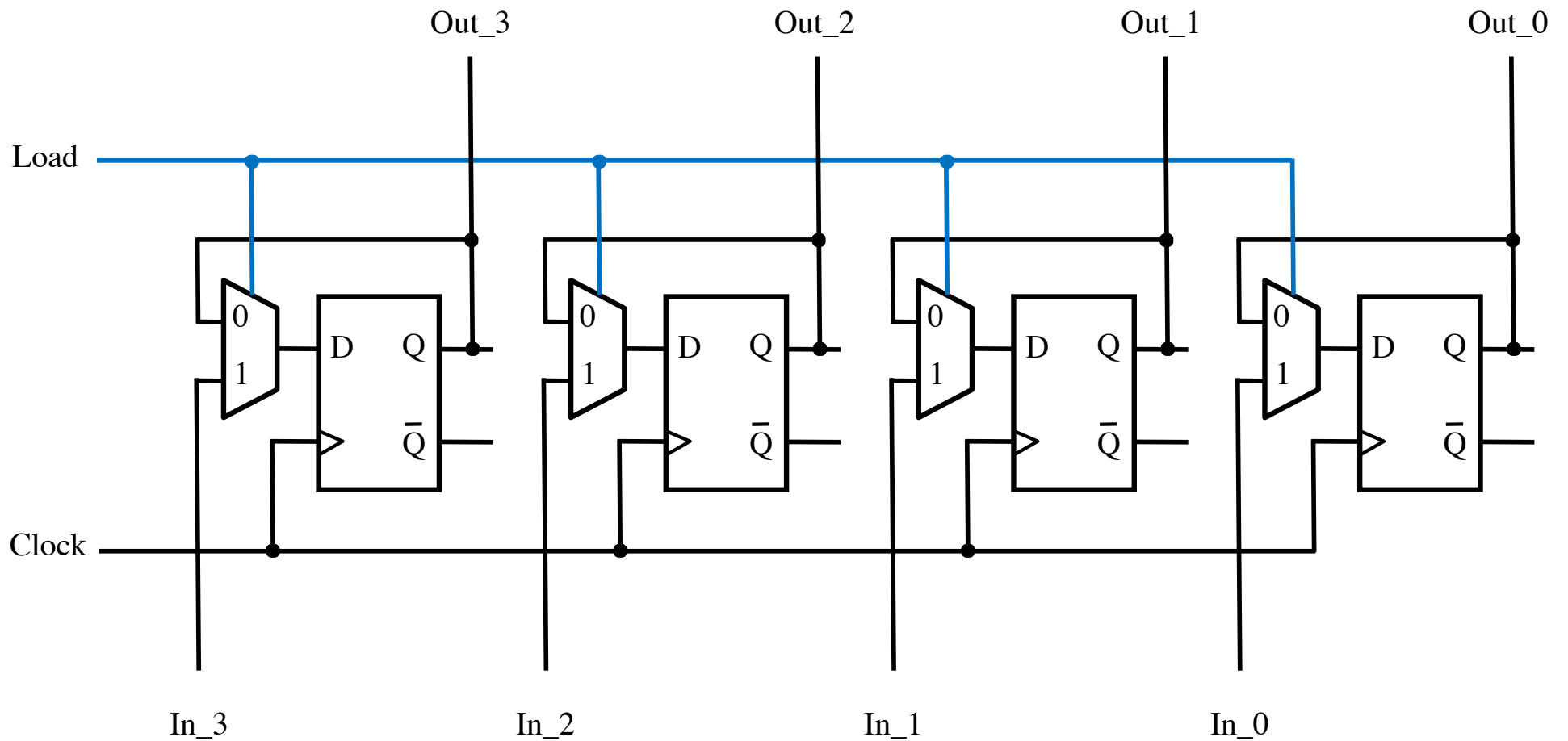
4-Bit Parallel-Access Register



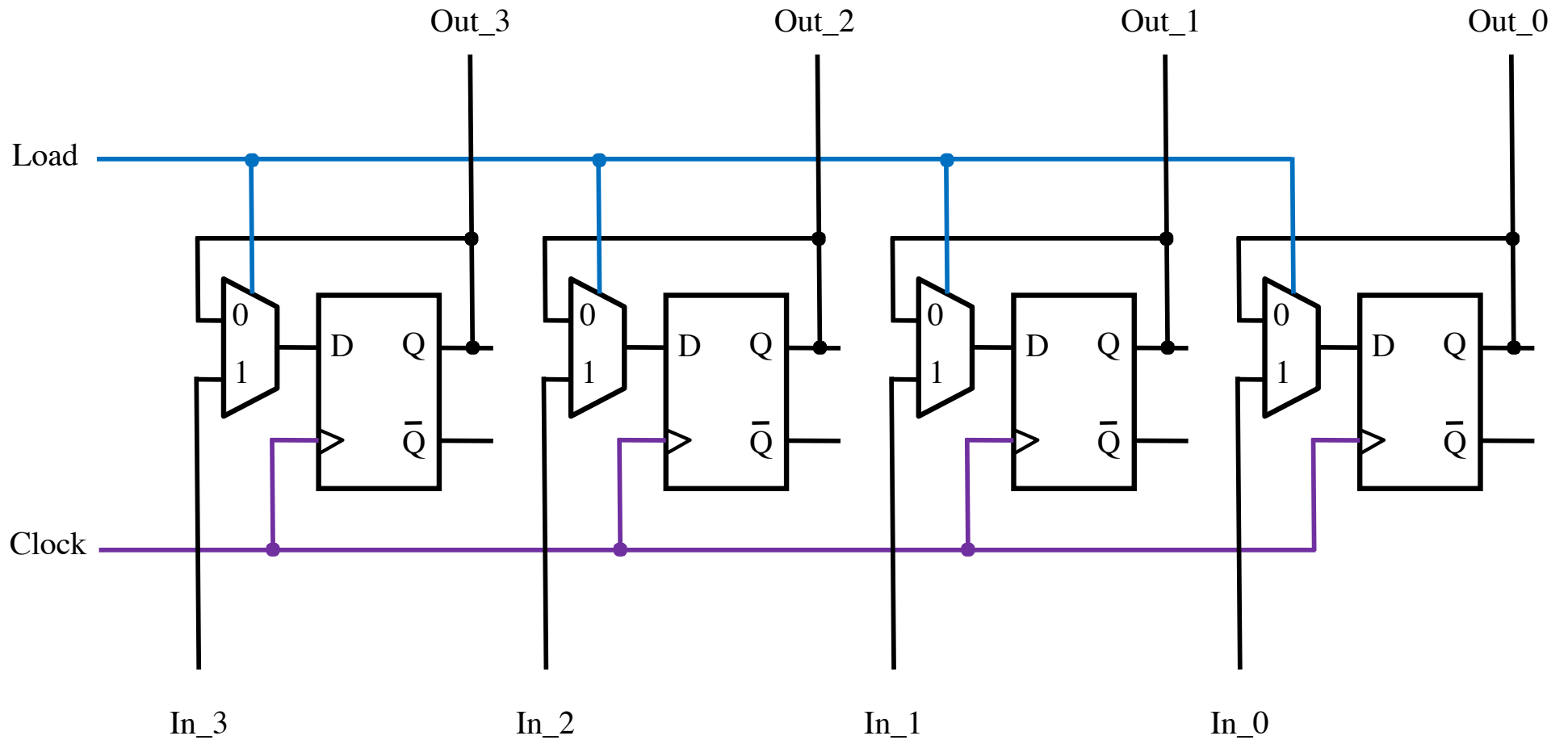
4-Bit Parallel-Access Register



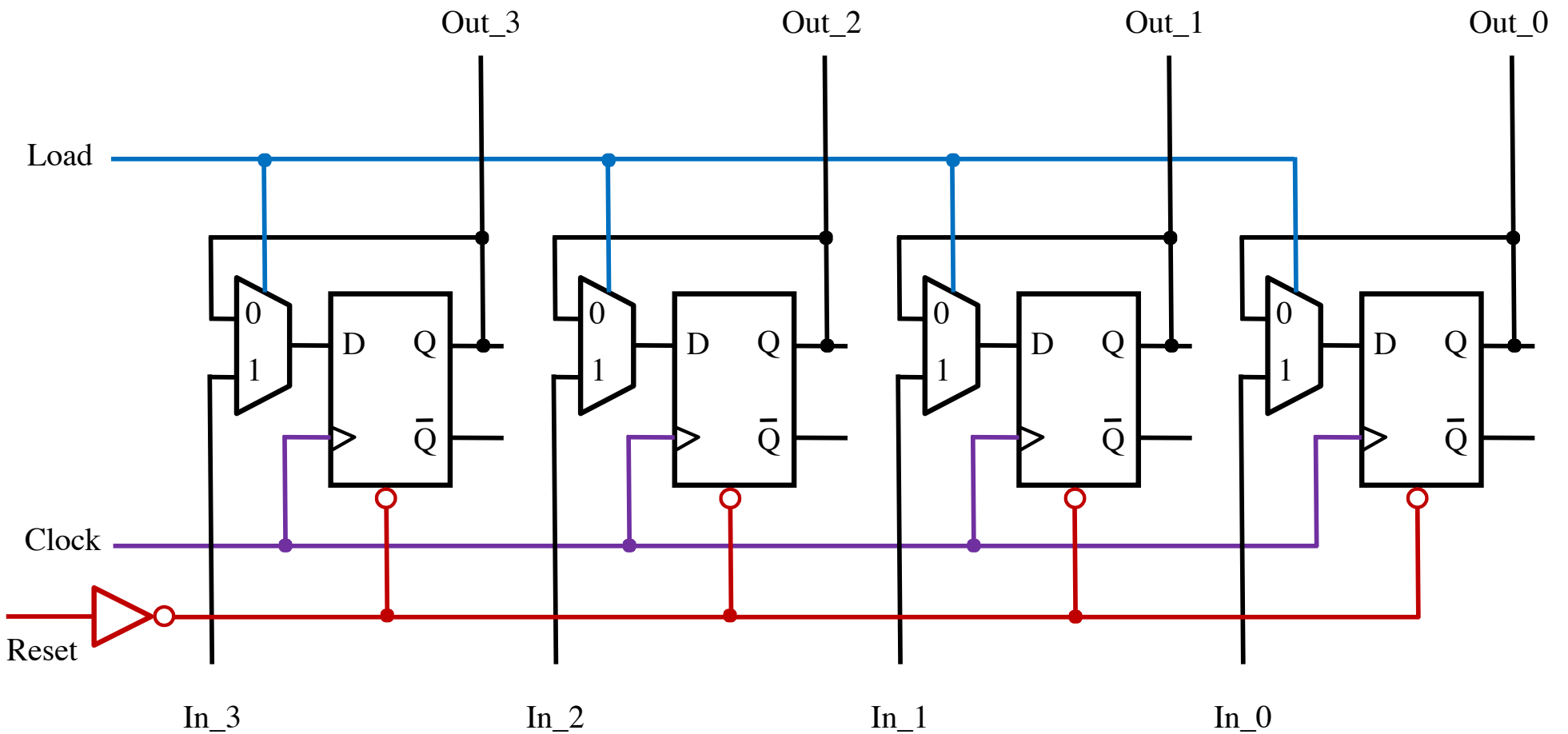
4-Bit Parallel-Access Register



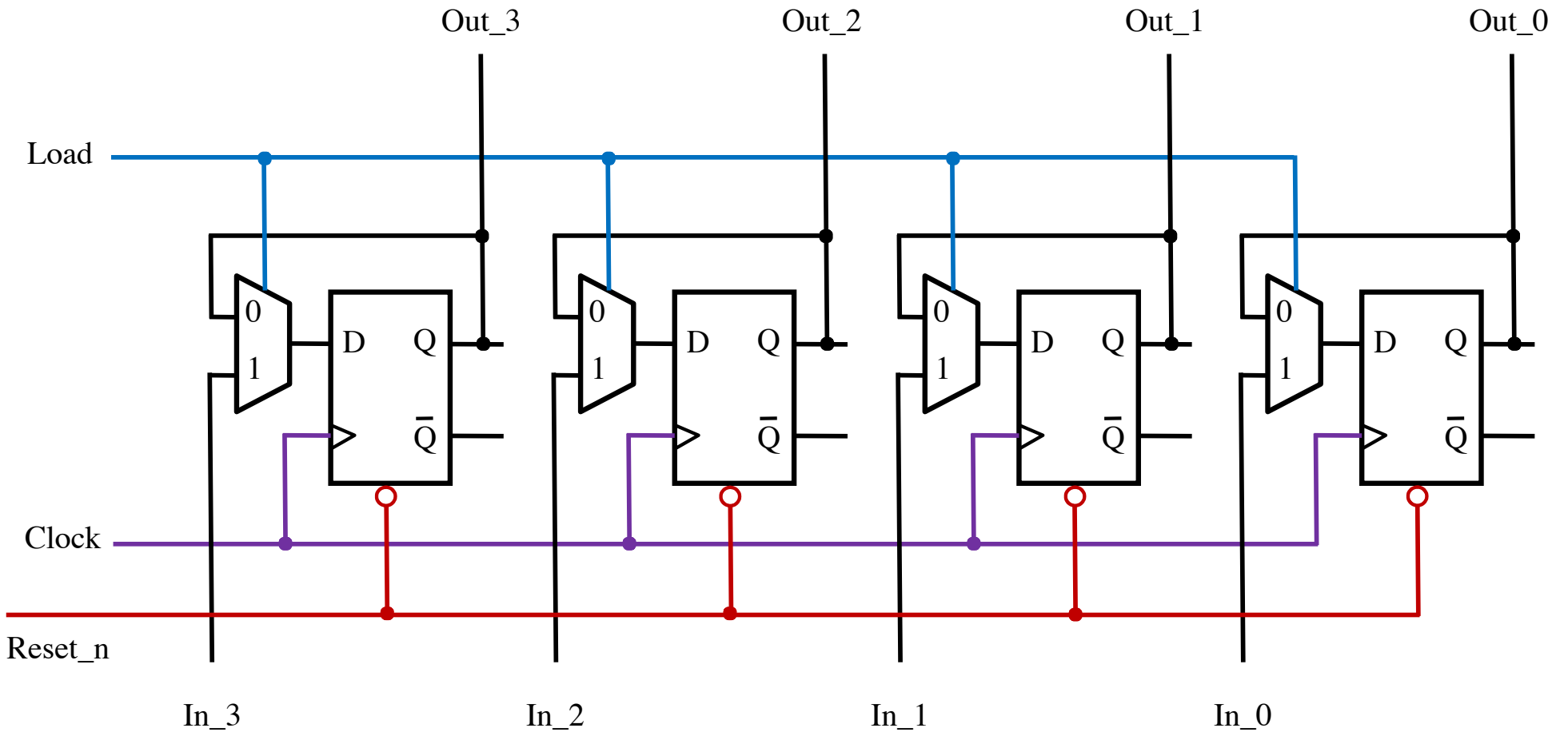
4-Bit Parallel-Access Register



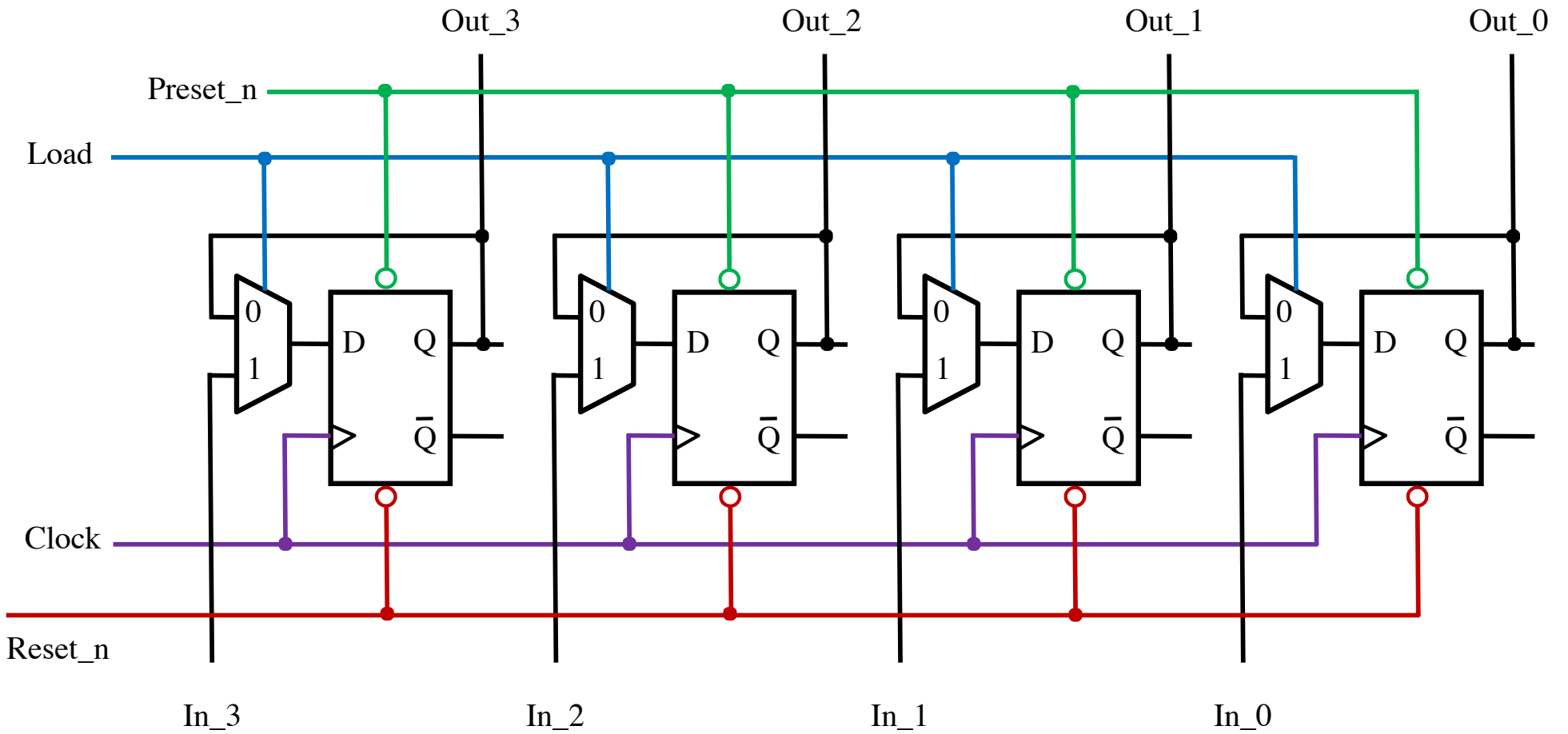
4-Bit Parallel-Access Register



4-Bit Parallel-Access Register

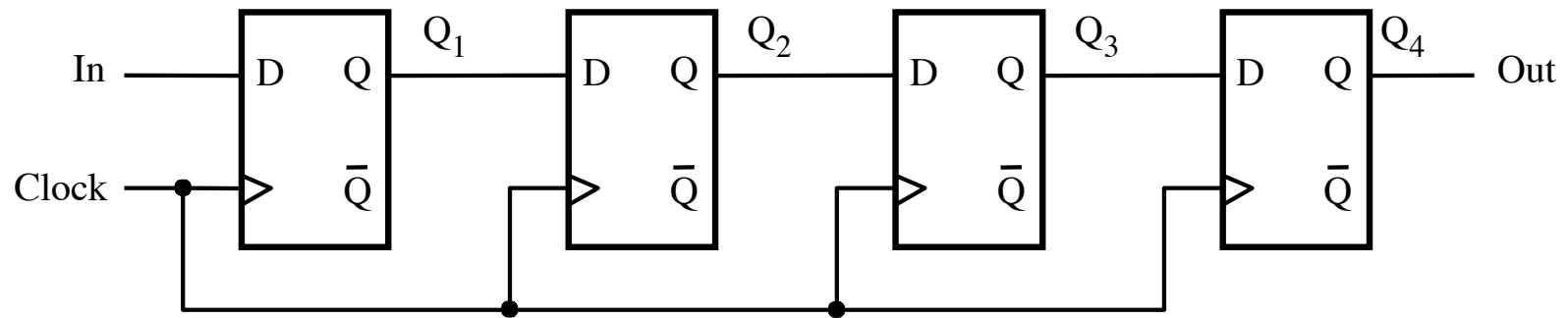


4-Bit Parallel-Access Register

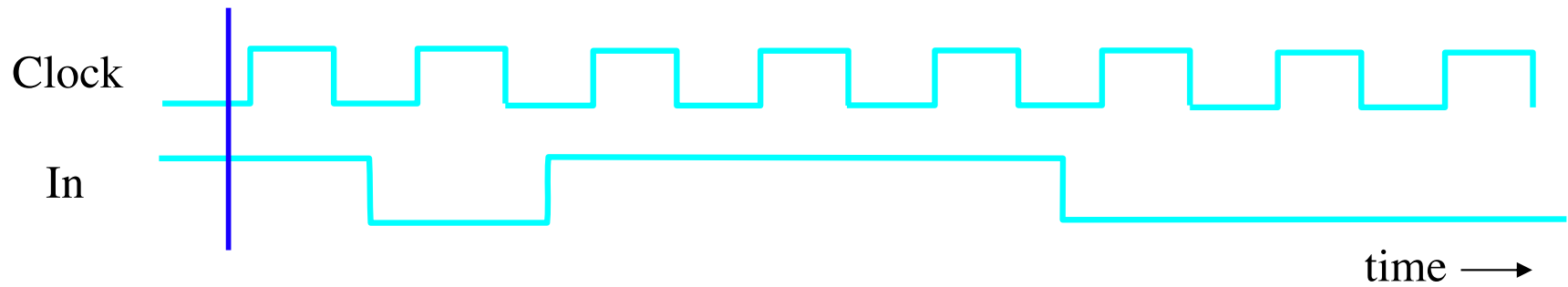
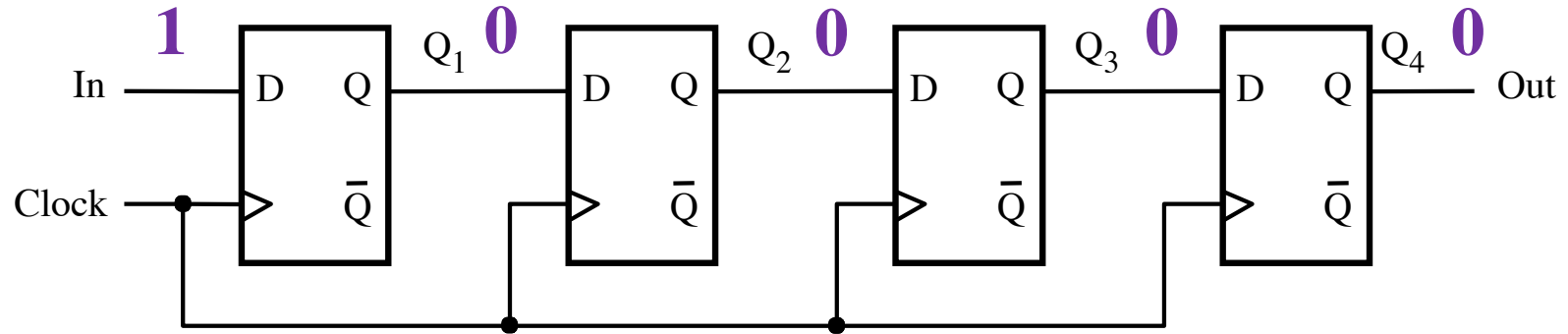


Shift Register

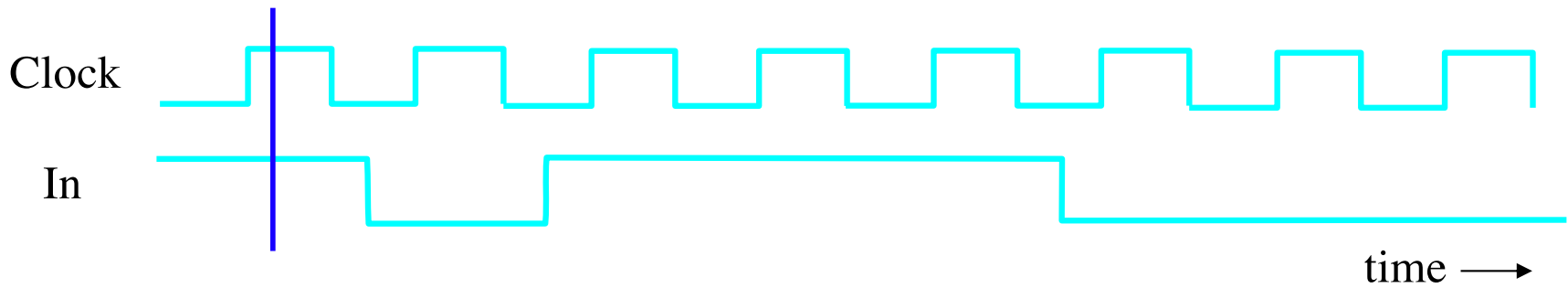
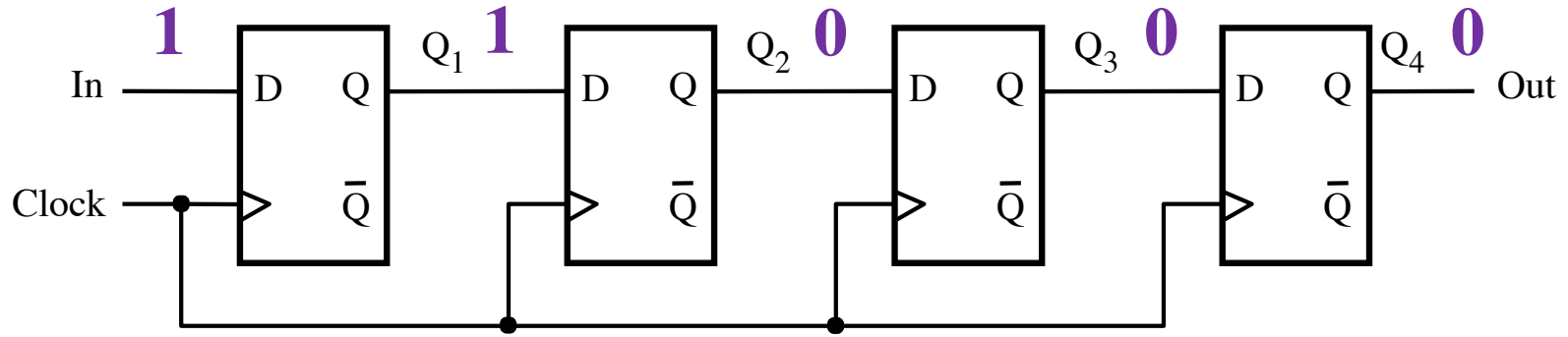
A simple shift register



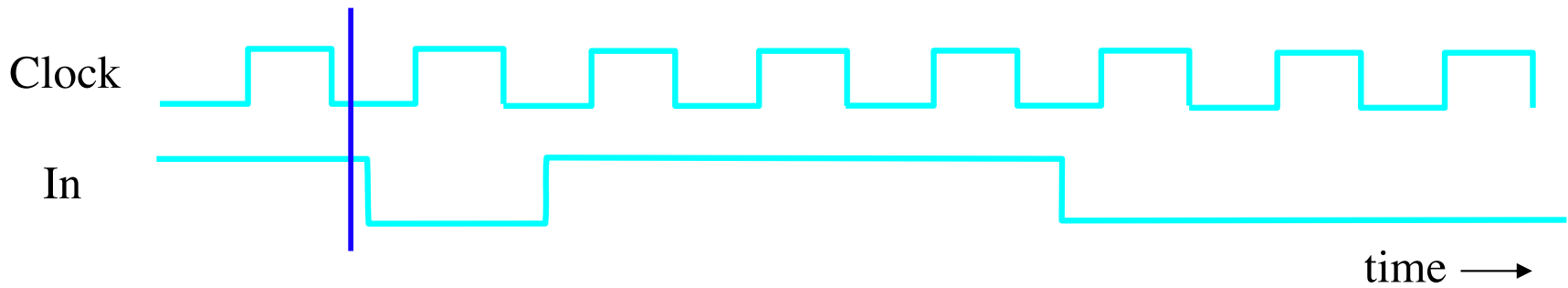
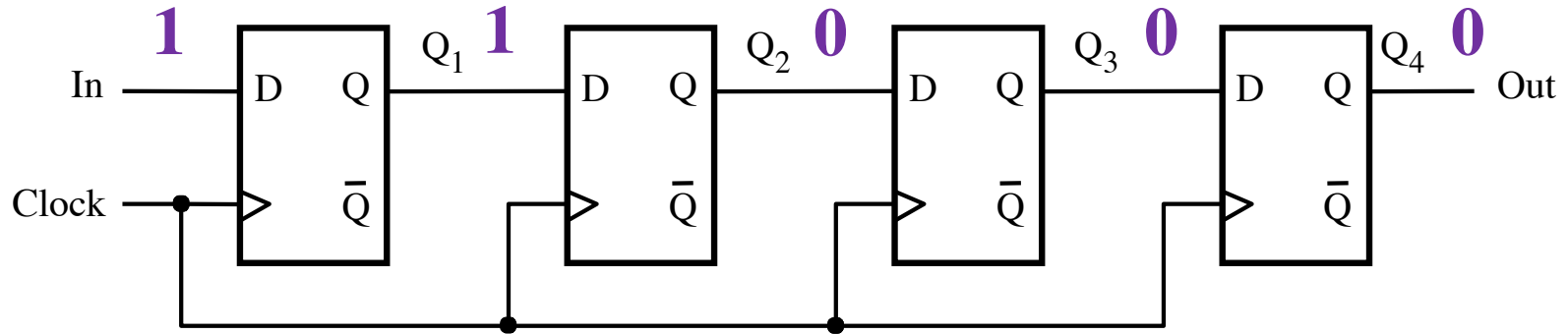
Shift Register Simulation



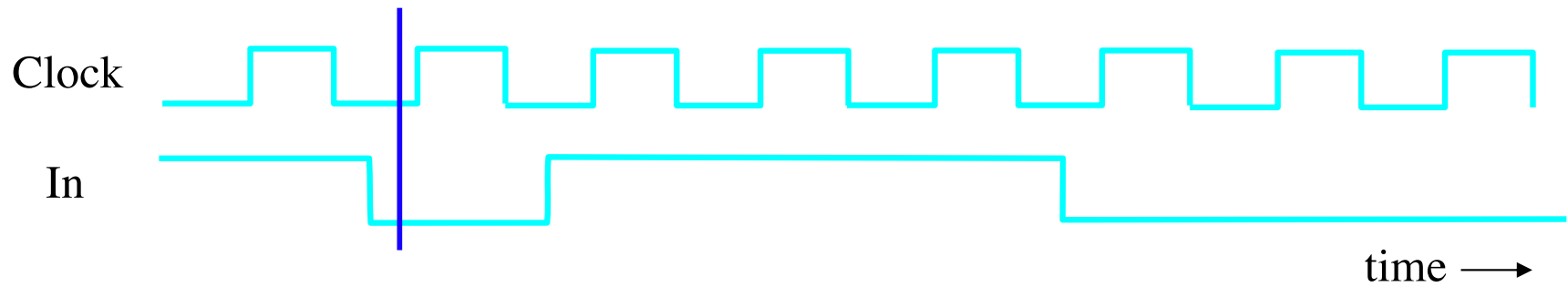
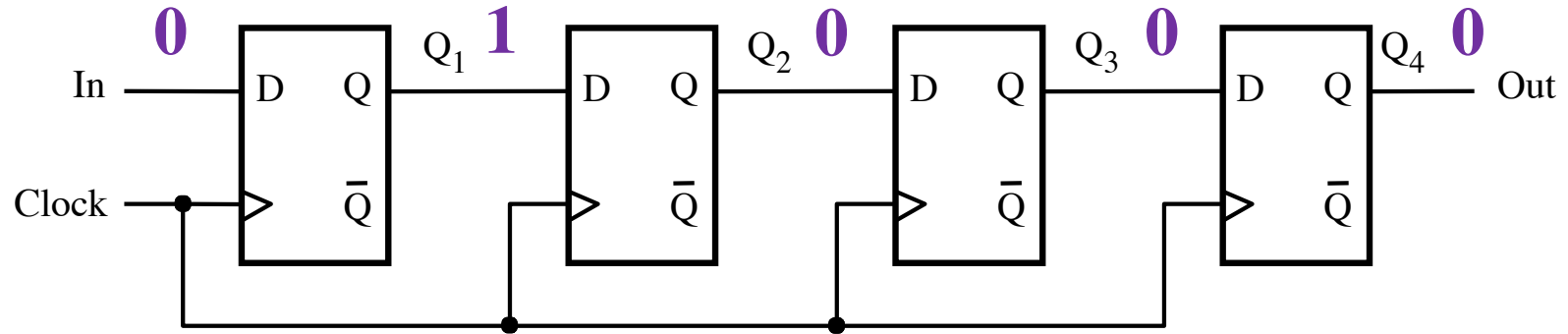
Shift Register Simulation



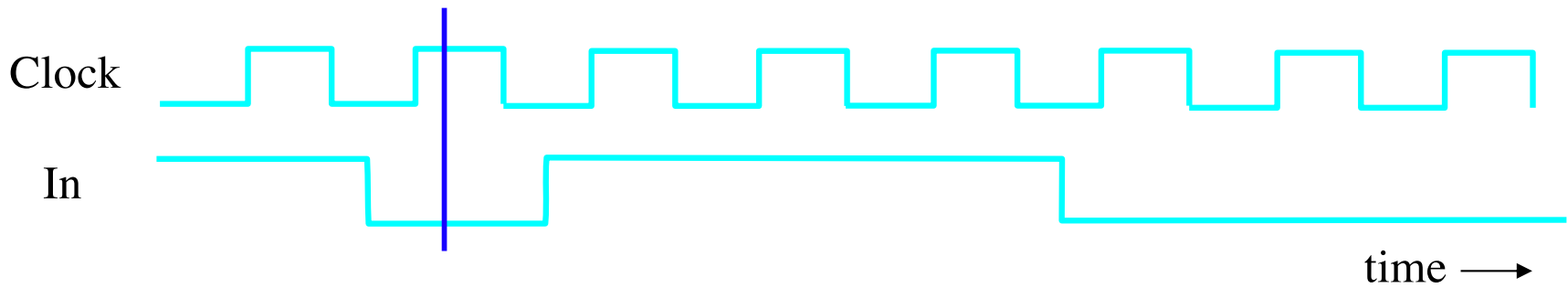
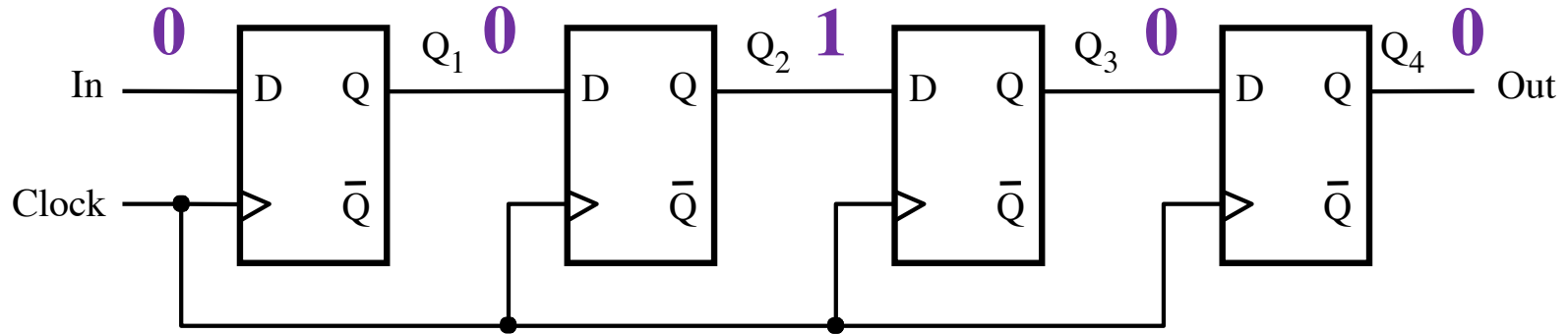
Shift Register Simulation



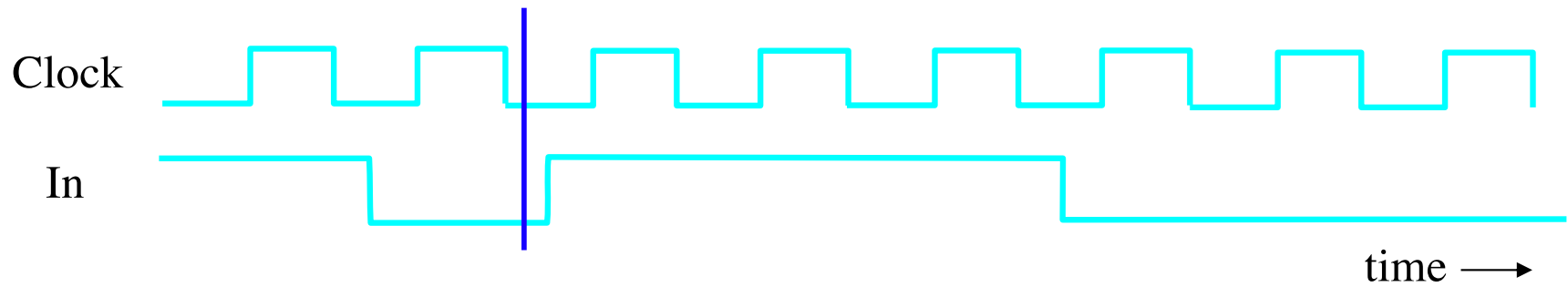
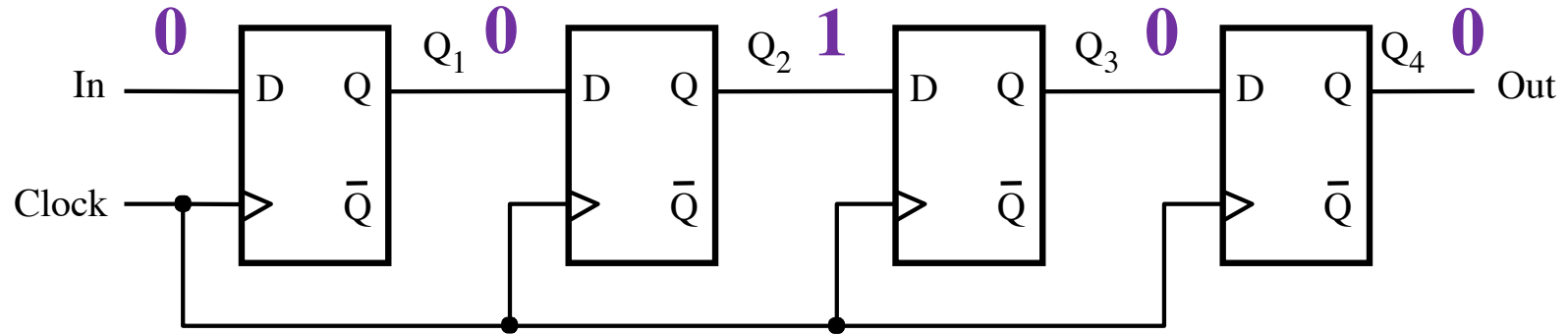
Shift Register Simulation



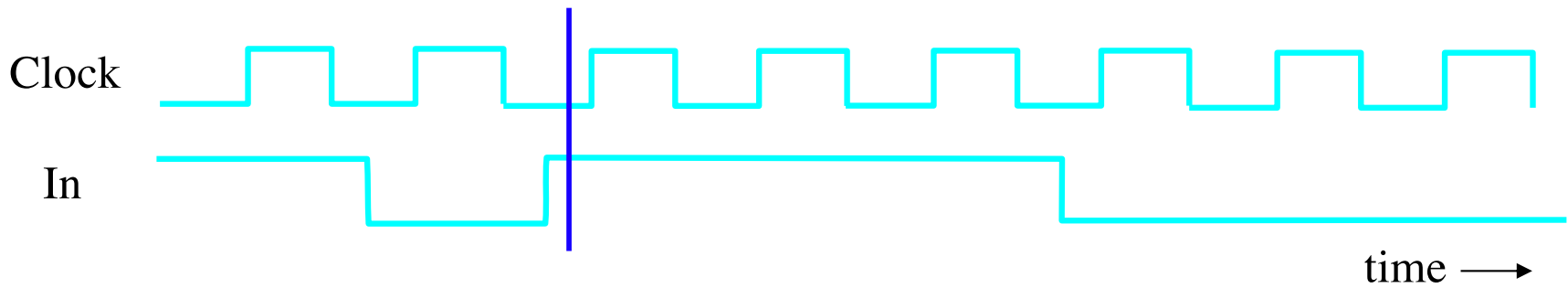
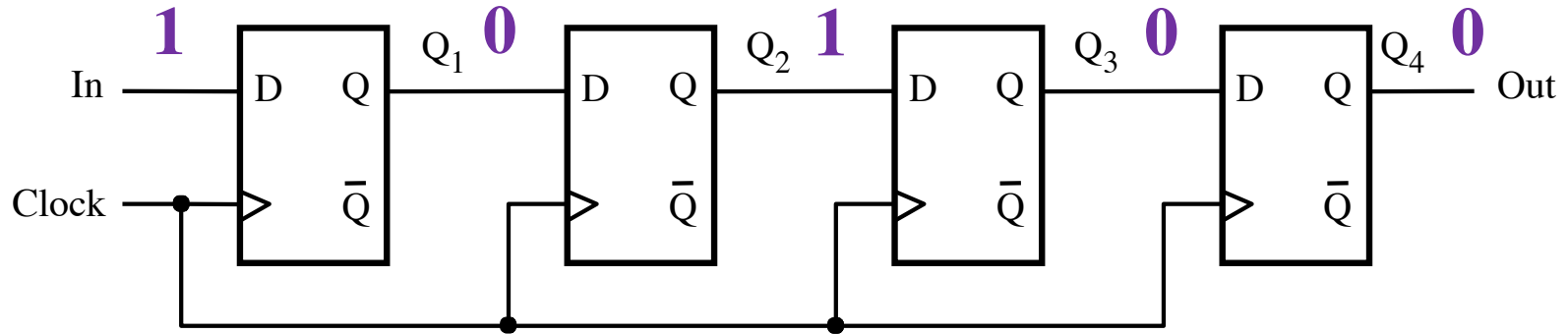
Shift Register Simulation



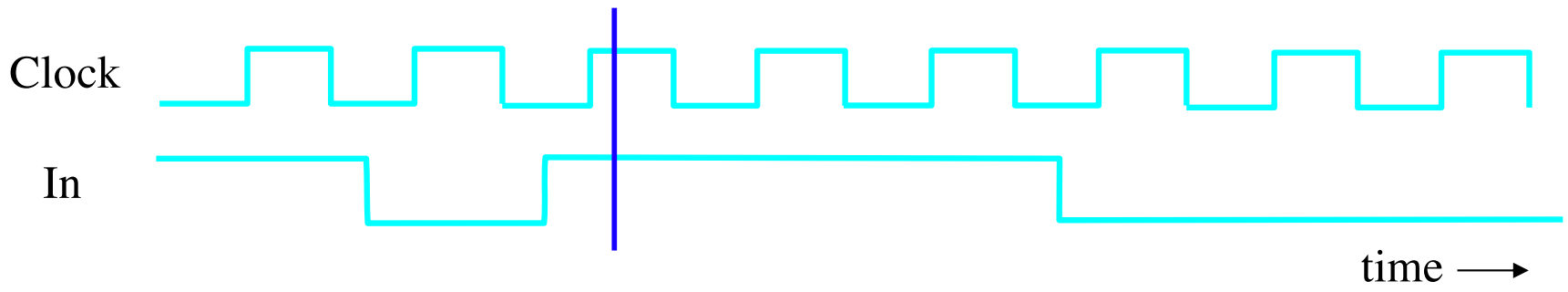
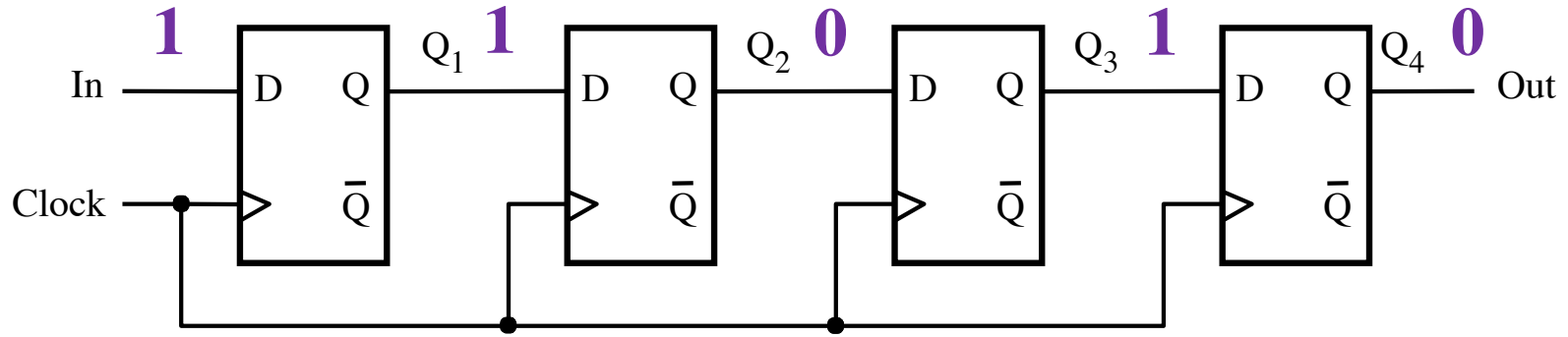
Shift Register Simulation



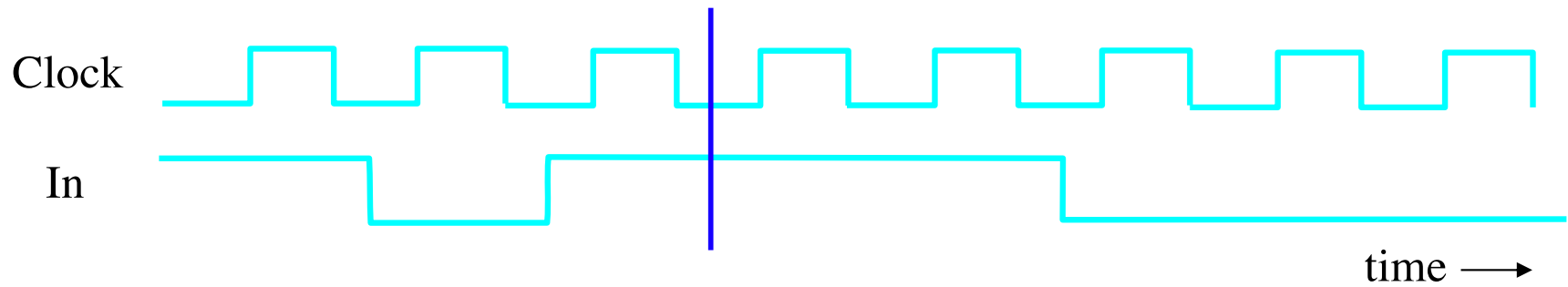
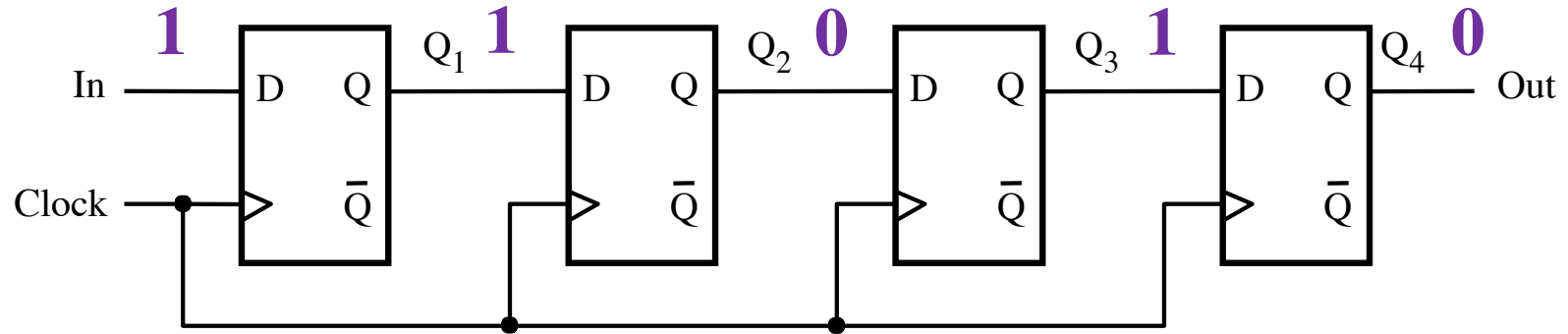
Shift Register Simulation



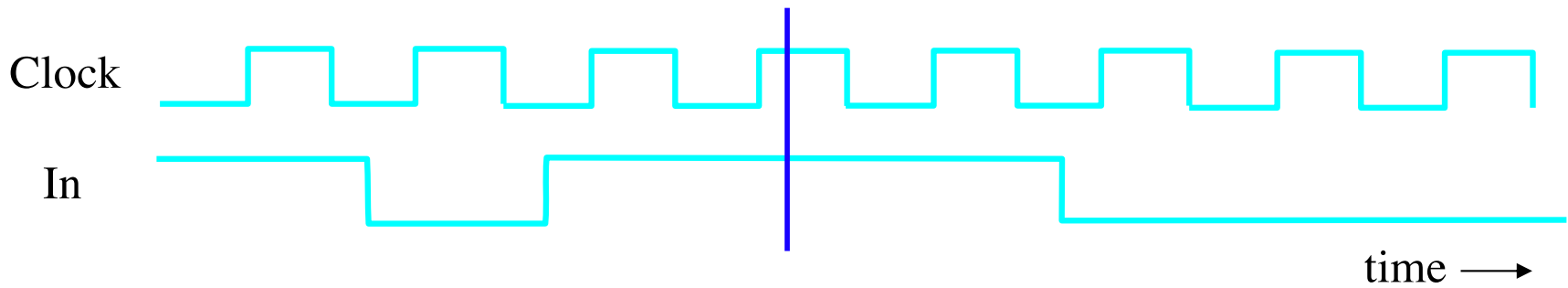
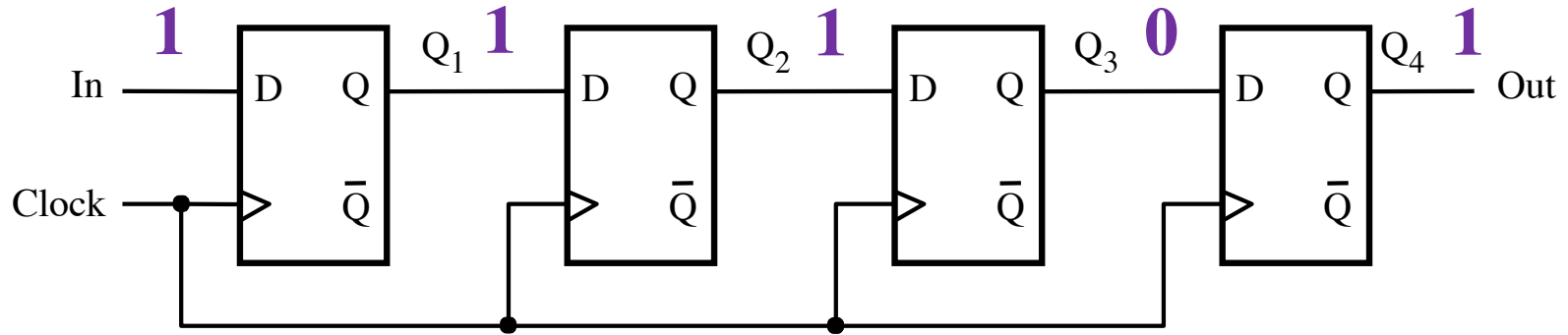
Shift Register Simulation



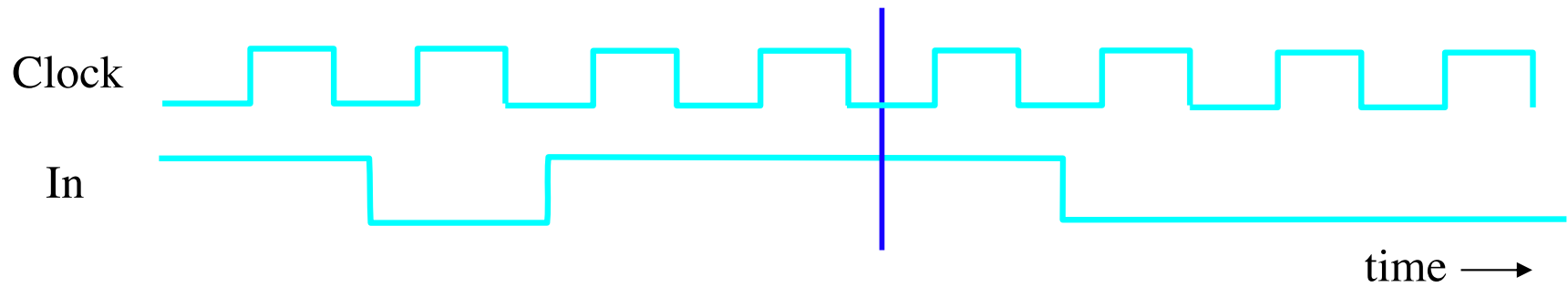
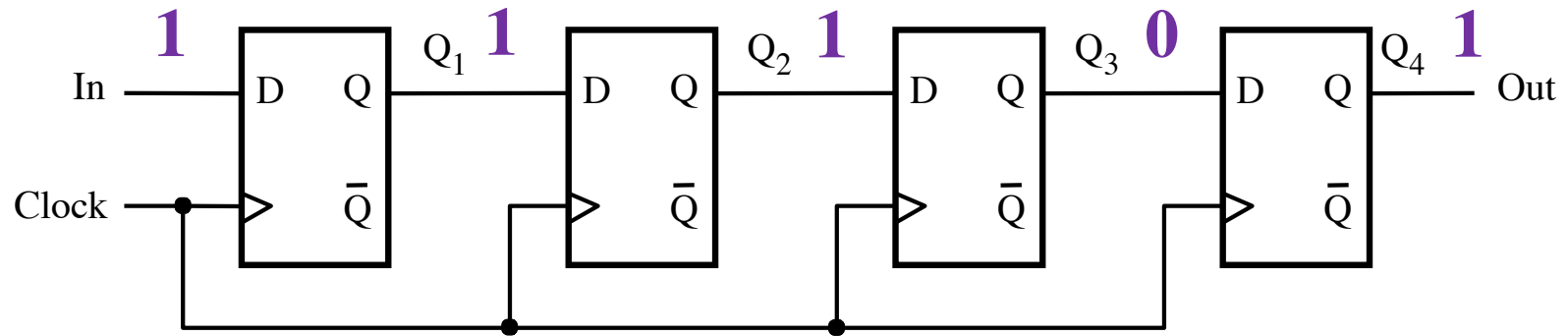
Shift Register Simulation



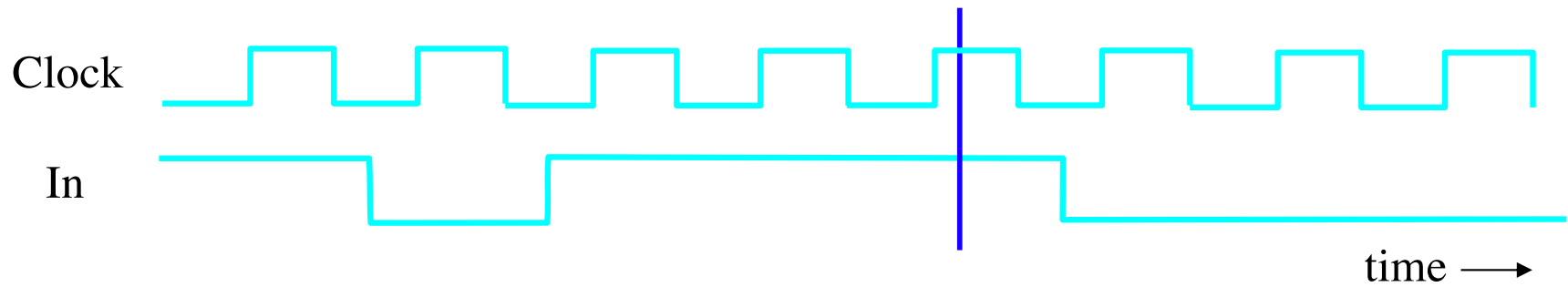
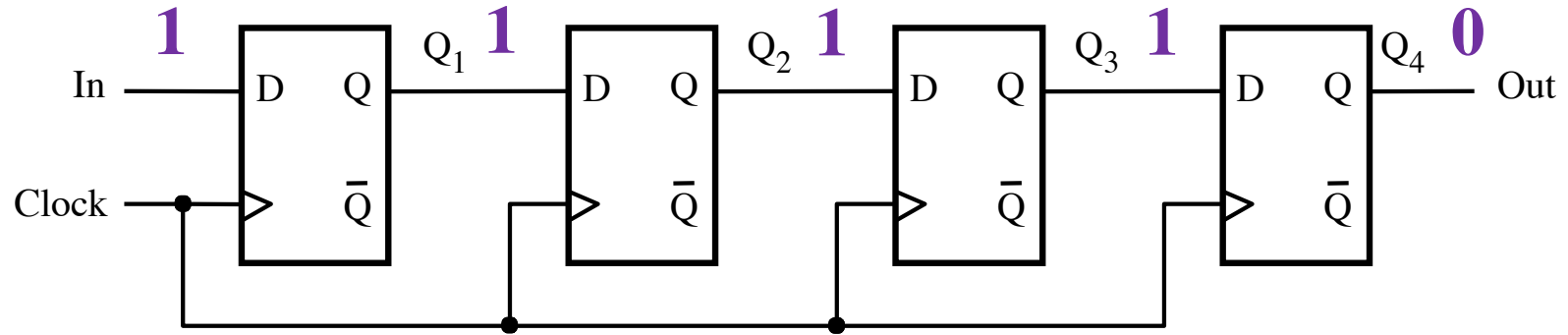
Shift Register Simulation



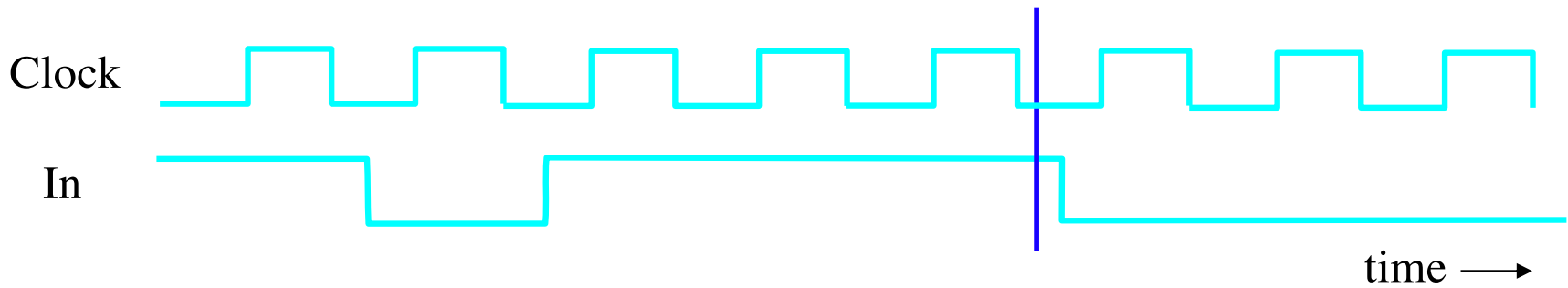
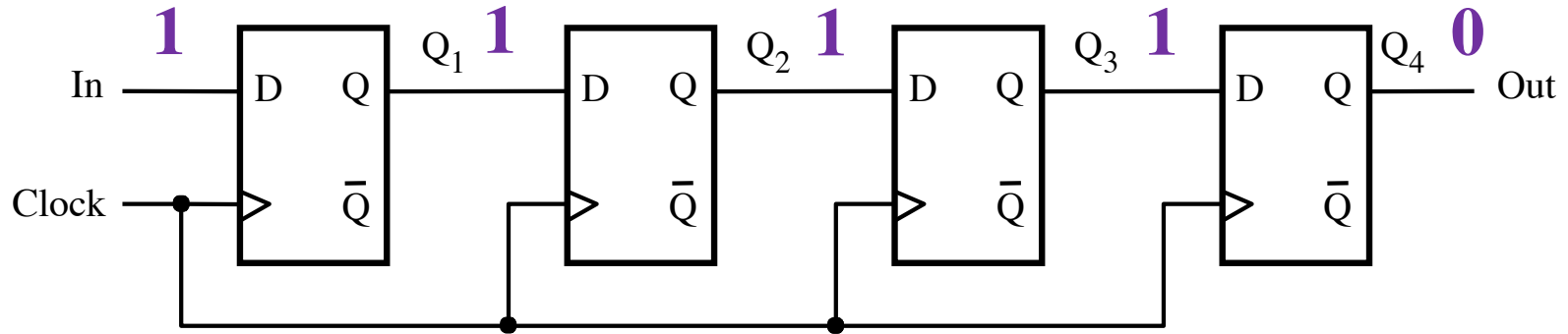
Shift Register Simulation



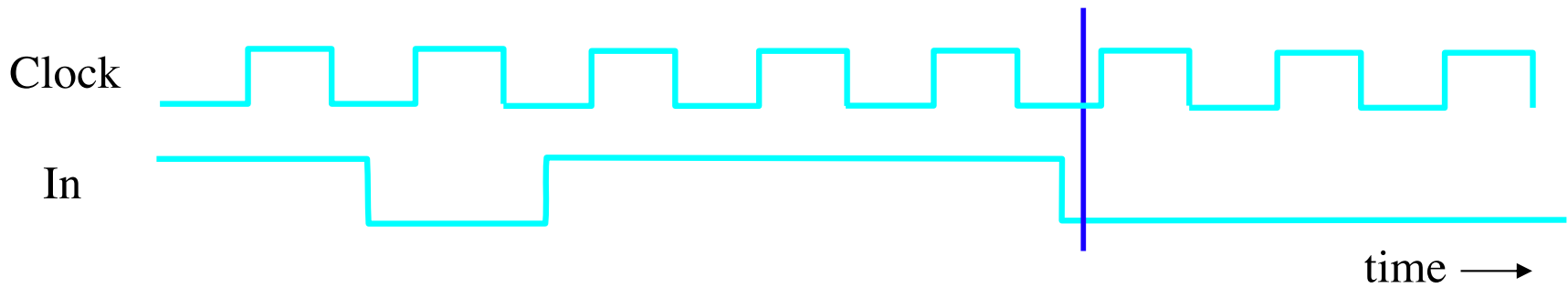
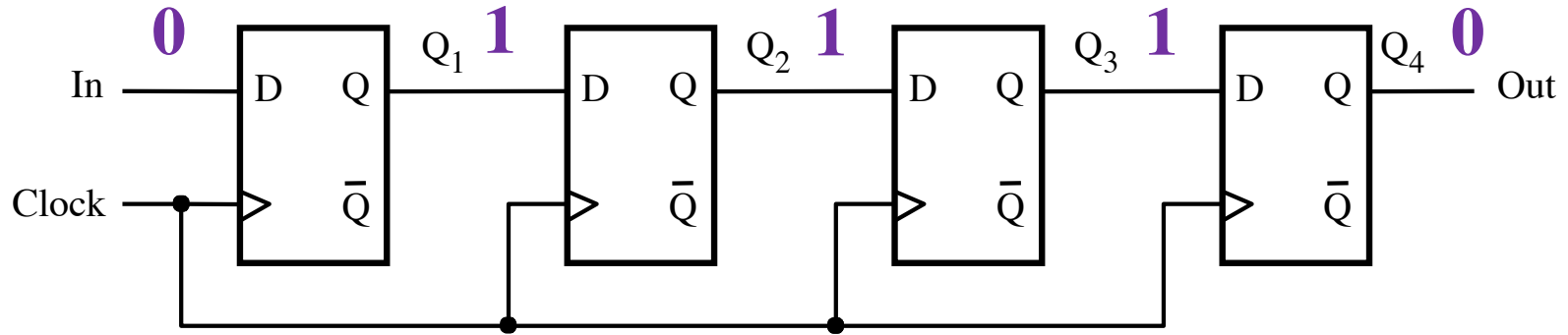
Shift Register Simulation



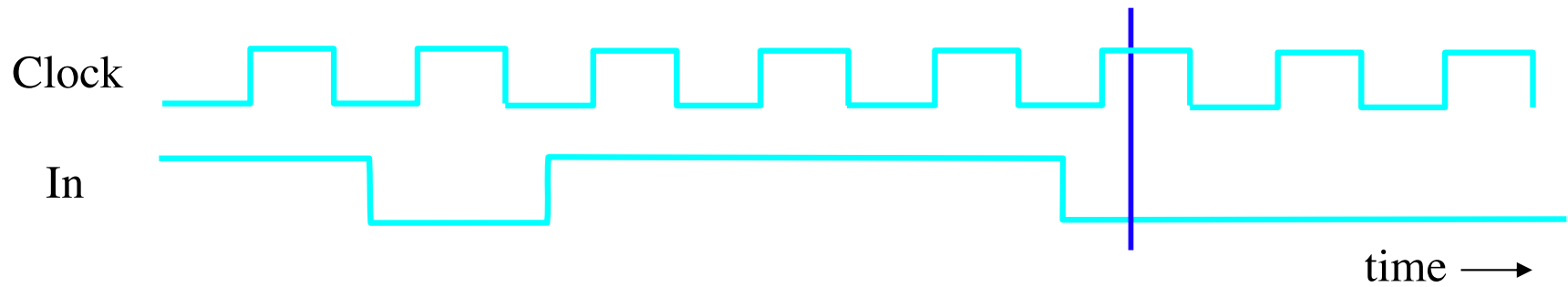
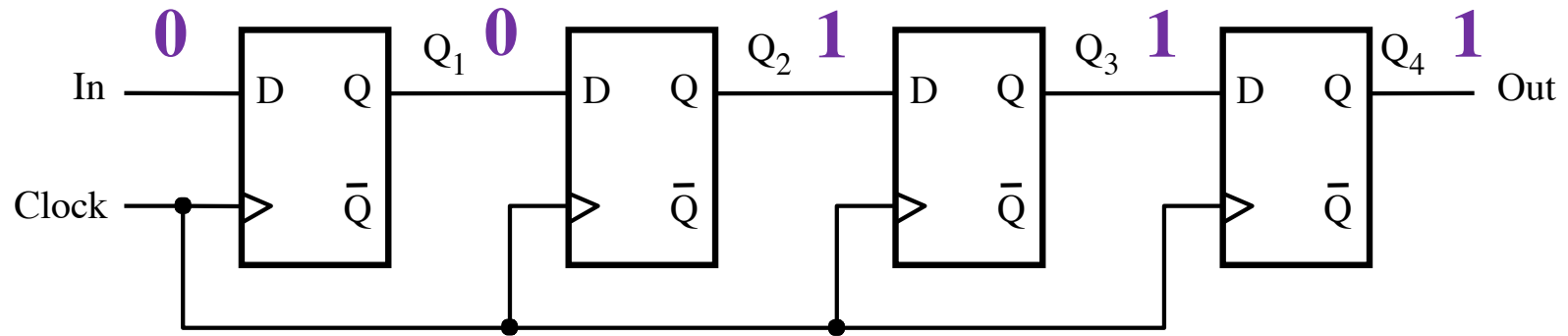
Shift Register Simulation



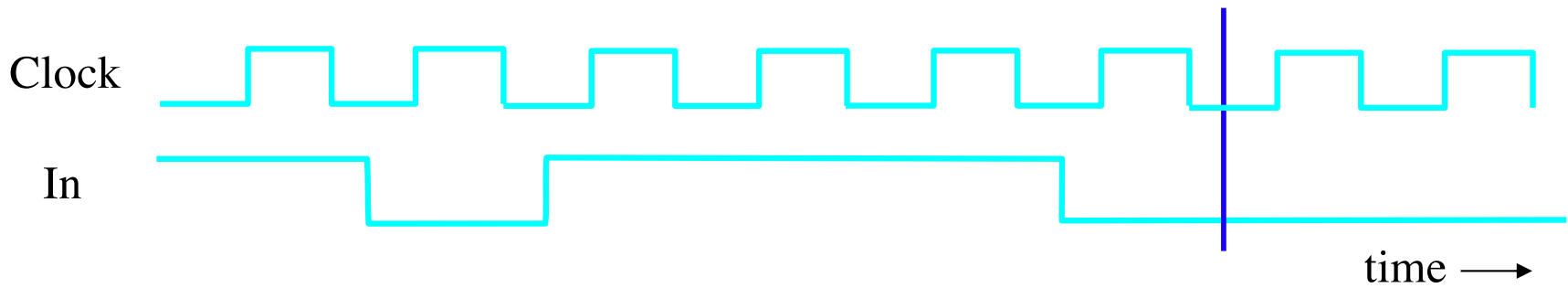
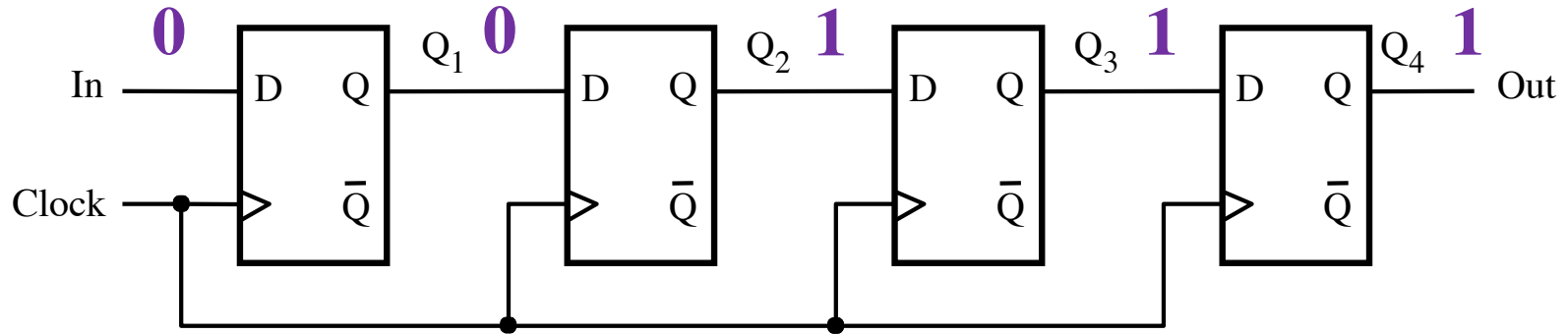
Shift Register Simulation



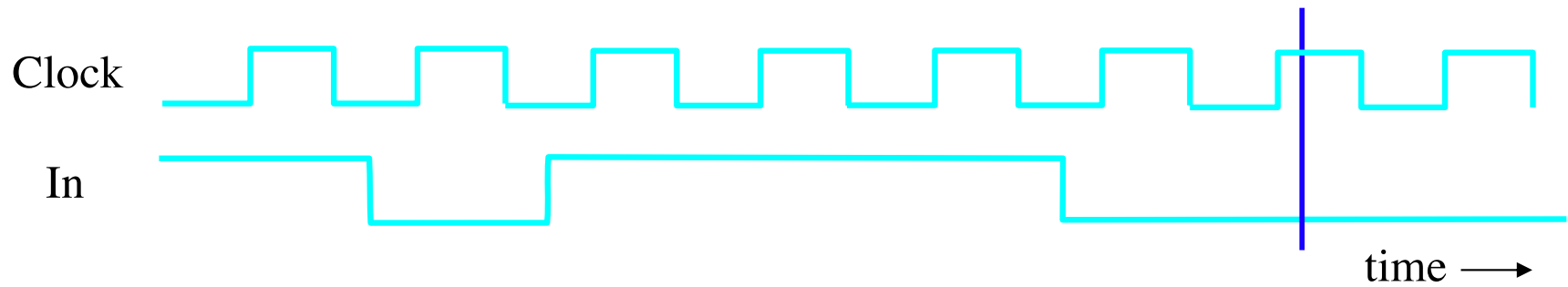
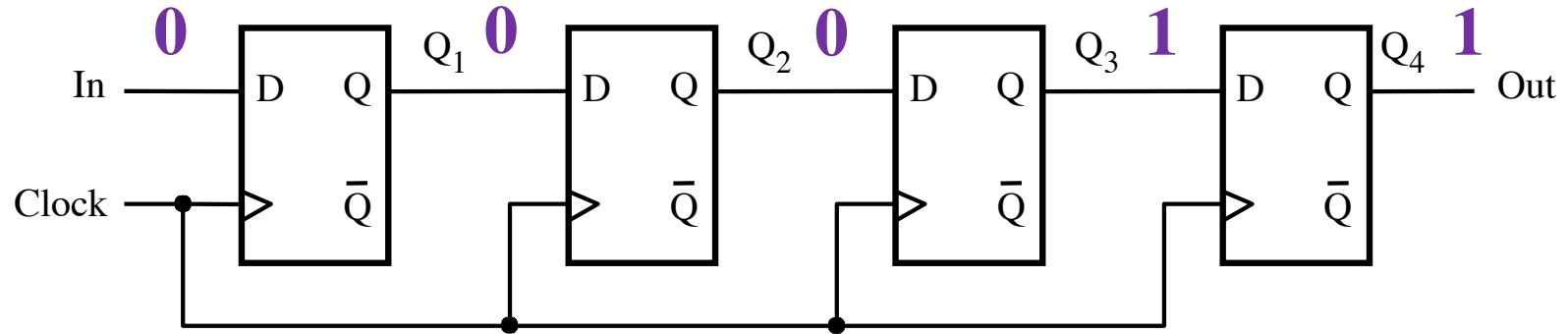
Shift Register Simulation



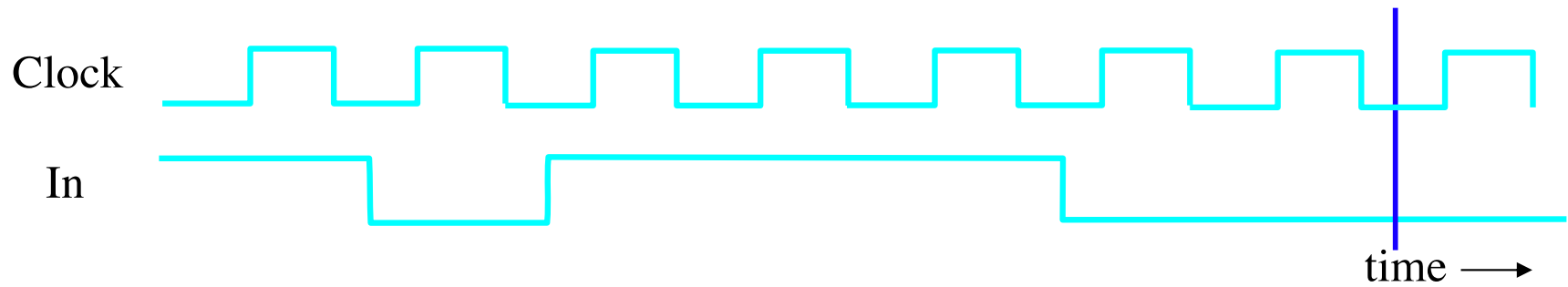
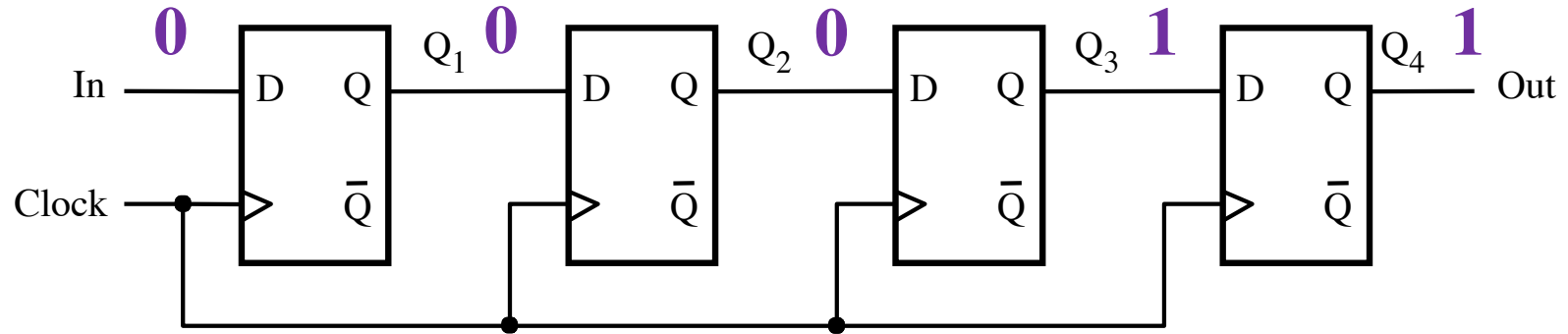
Shift Register Simulation



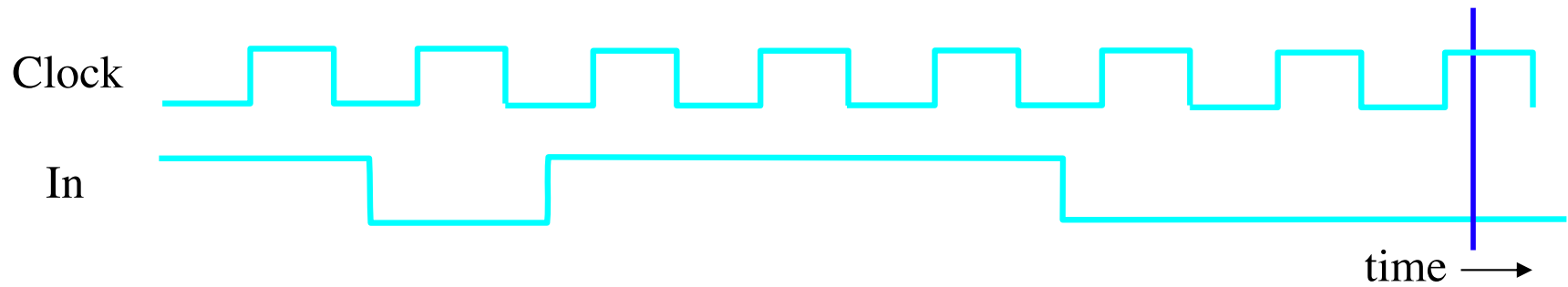
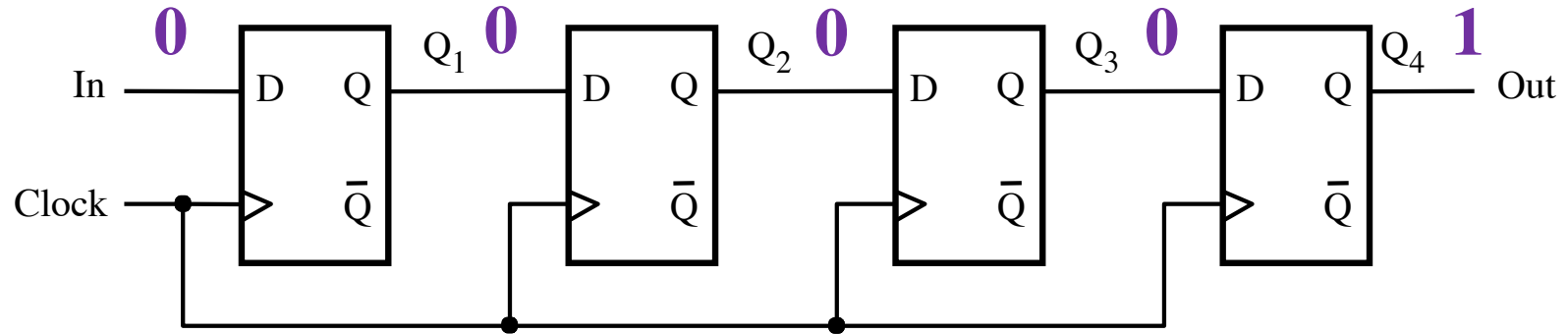
Shift Register Simulation



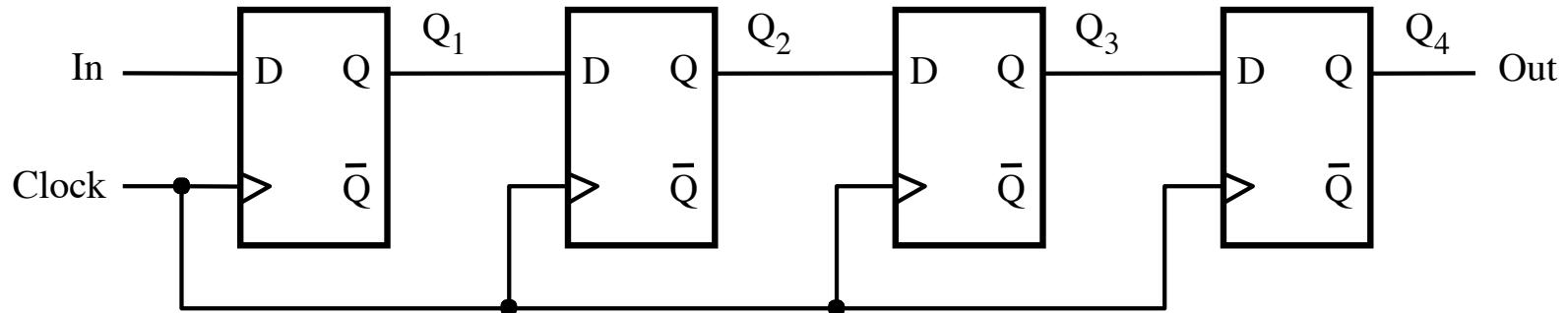
Shift Register Simulation



Shift Register Simulation



A simple shift register

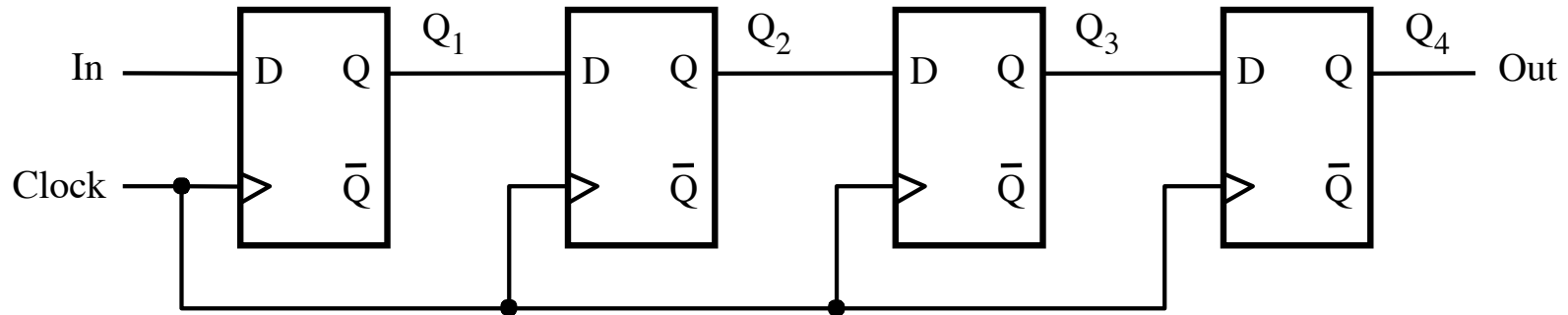


(a) Circuit

time	In	Q_1	Q_2	Q_3	$Q_4 = \text{Out}$
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

A simple shift register

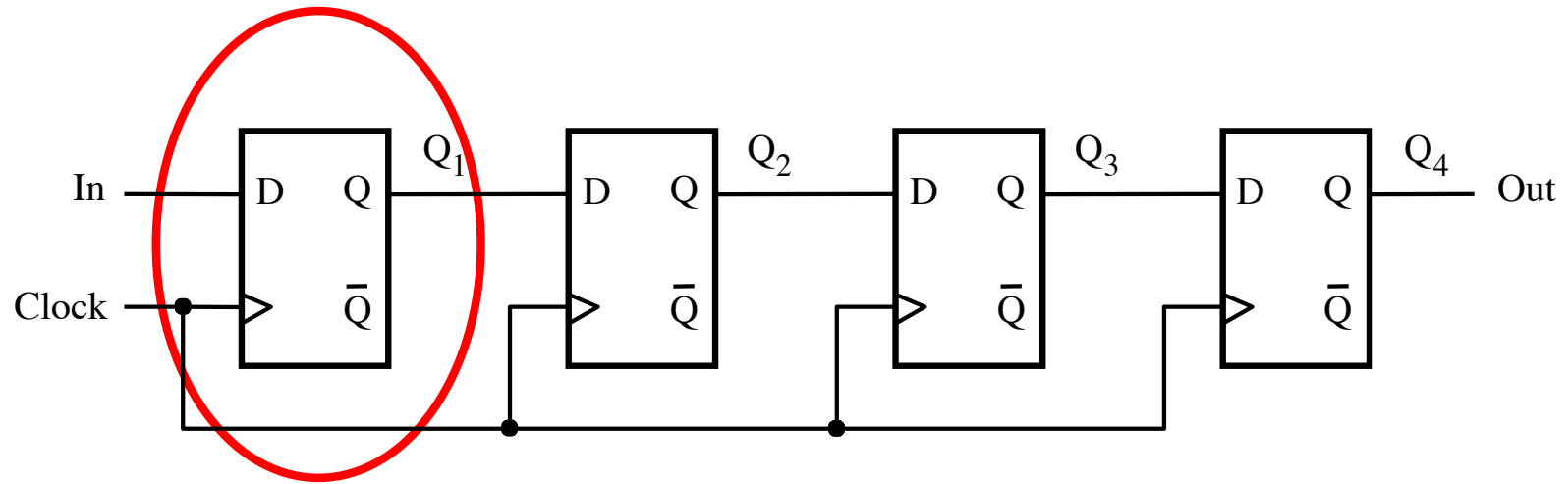


(a) Circuit

time	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
	0	0	0	0	1

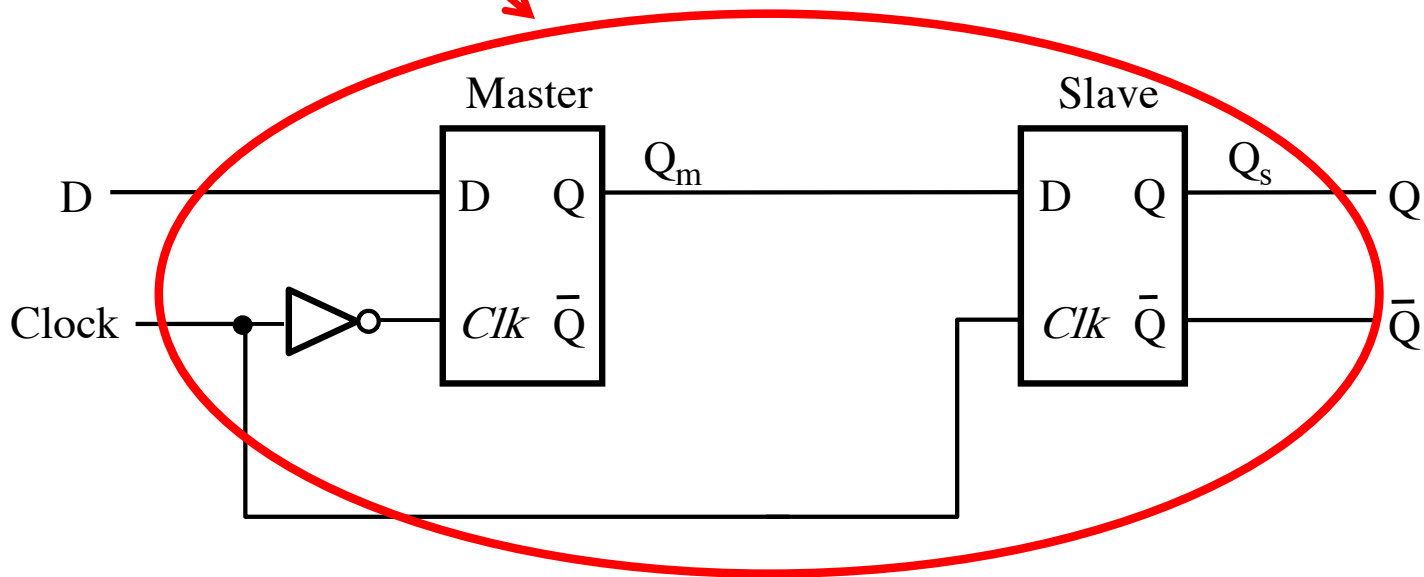
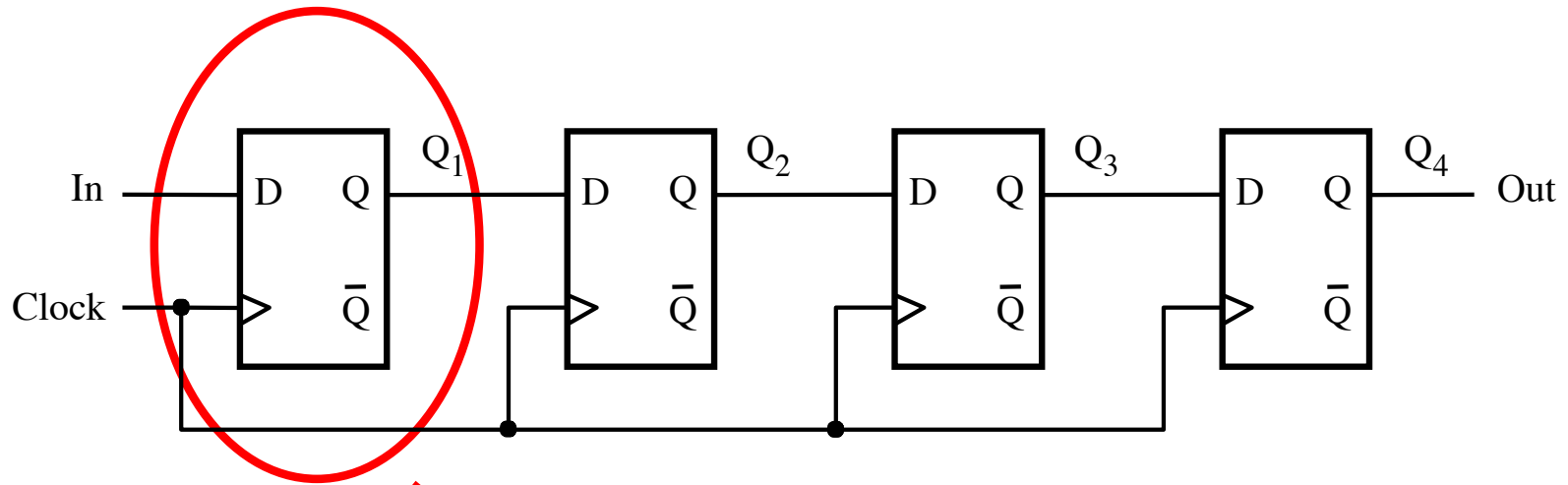
The simulation goes
one step further

A simple shift register

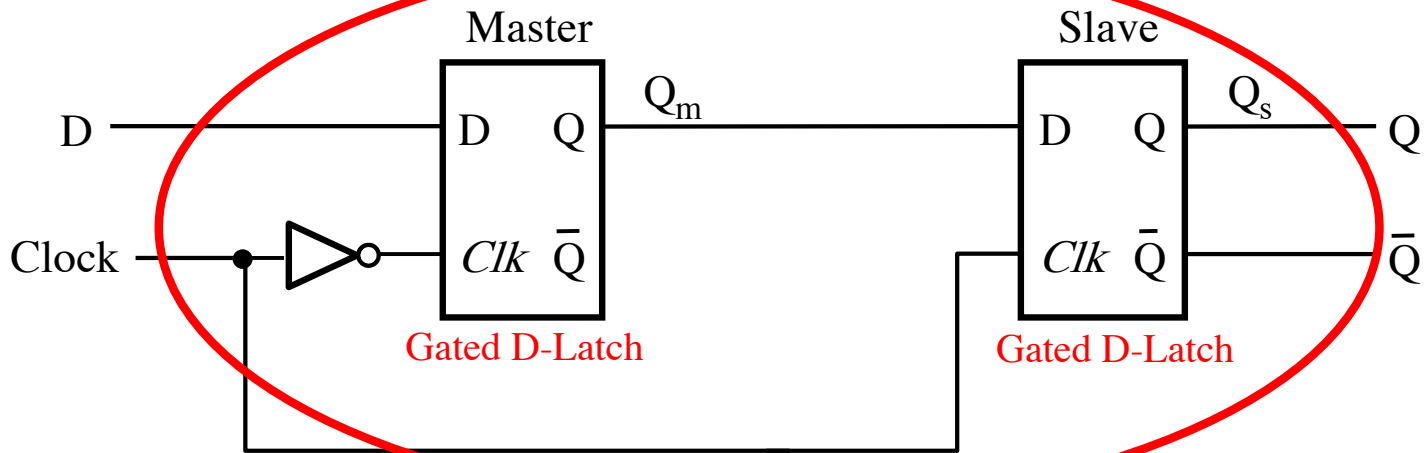
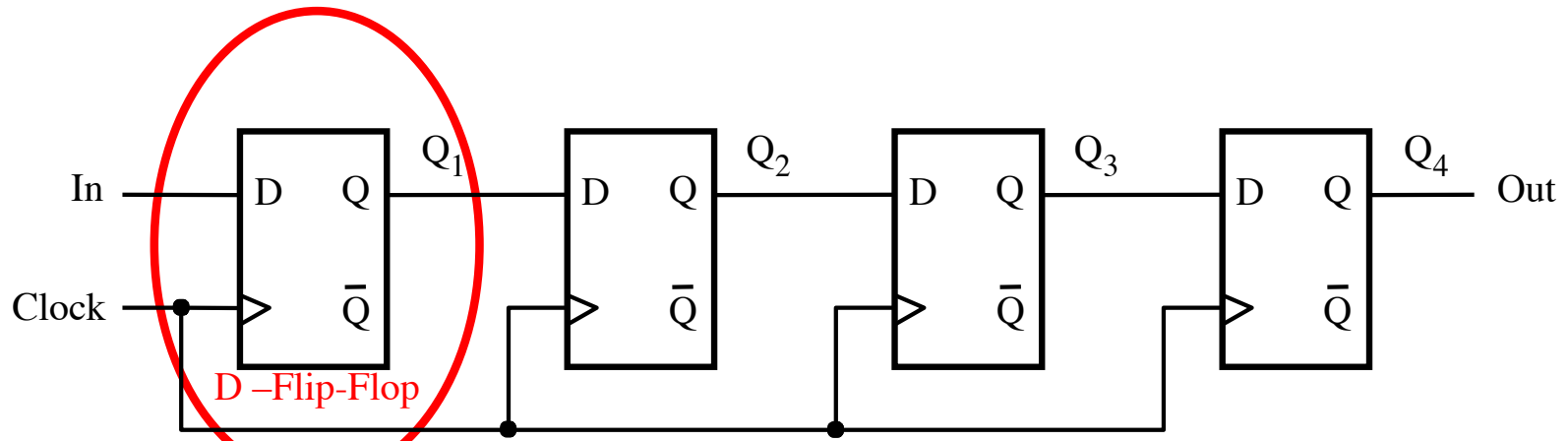


Positive-edge-triggered
D Flip-Flop

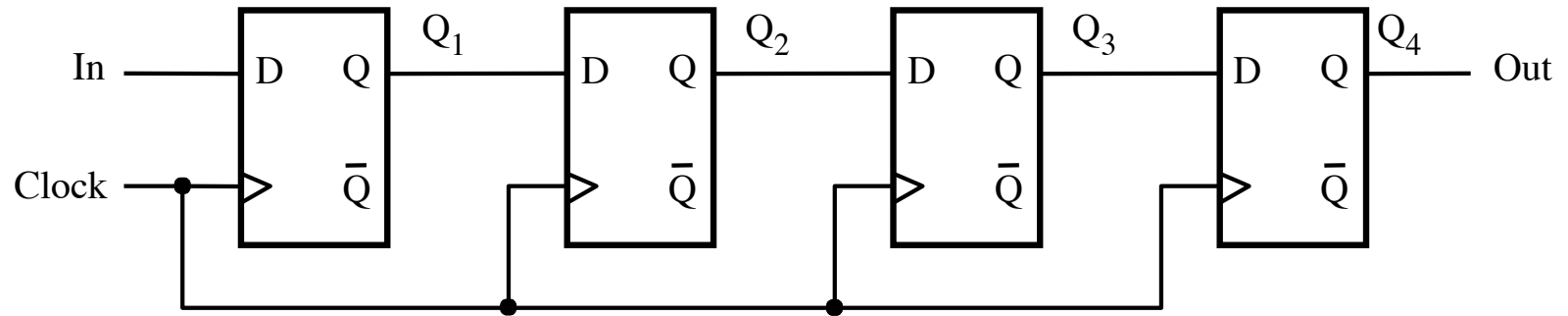
A simple shift register



A simple shift register



A simple shift register



You need 2 latches to make a flip-flop



=



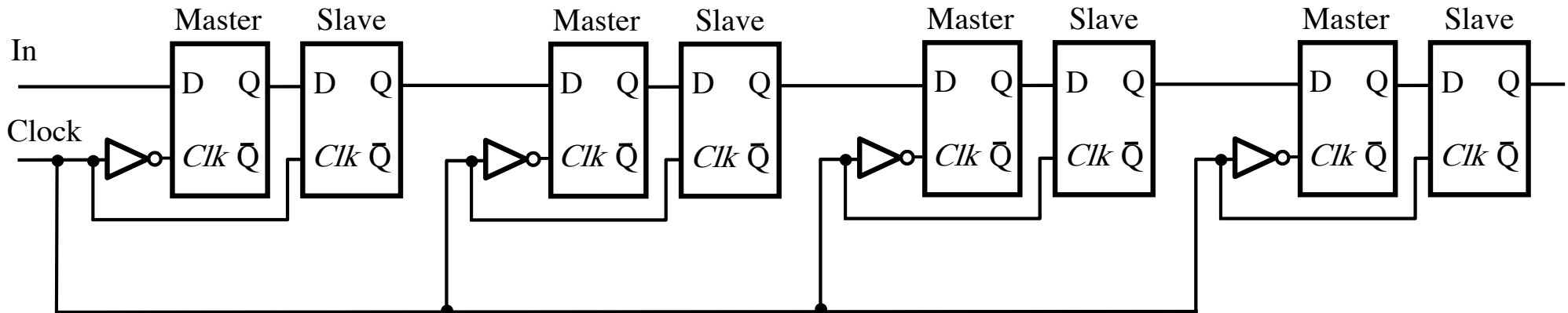
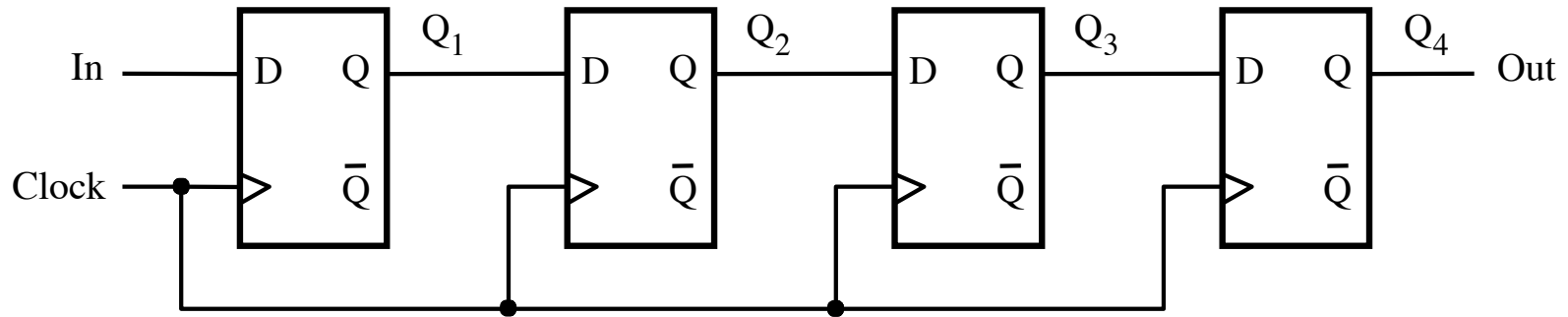
4-bit Register



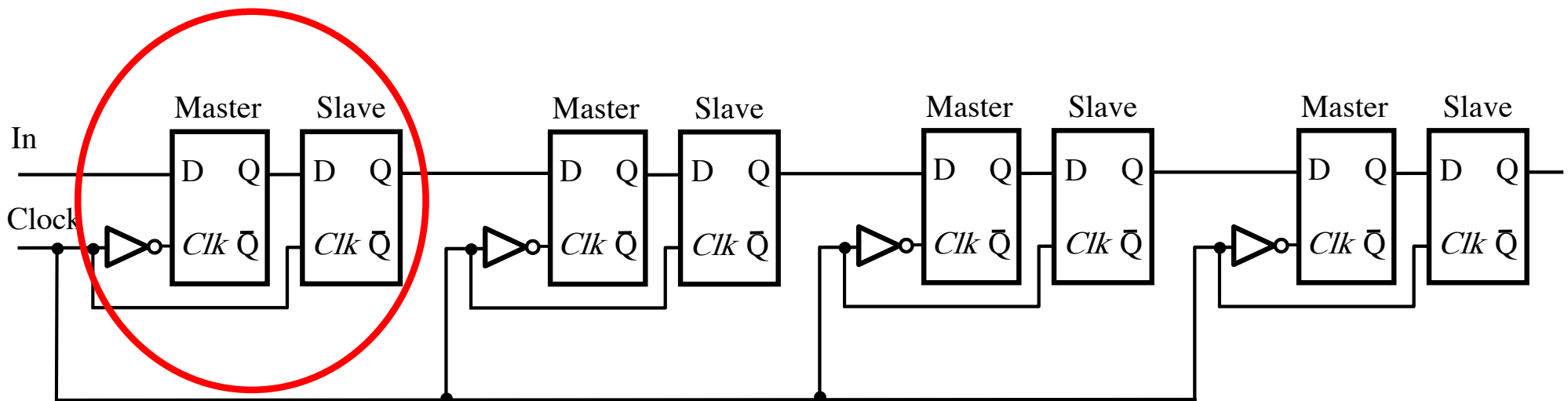
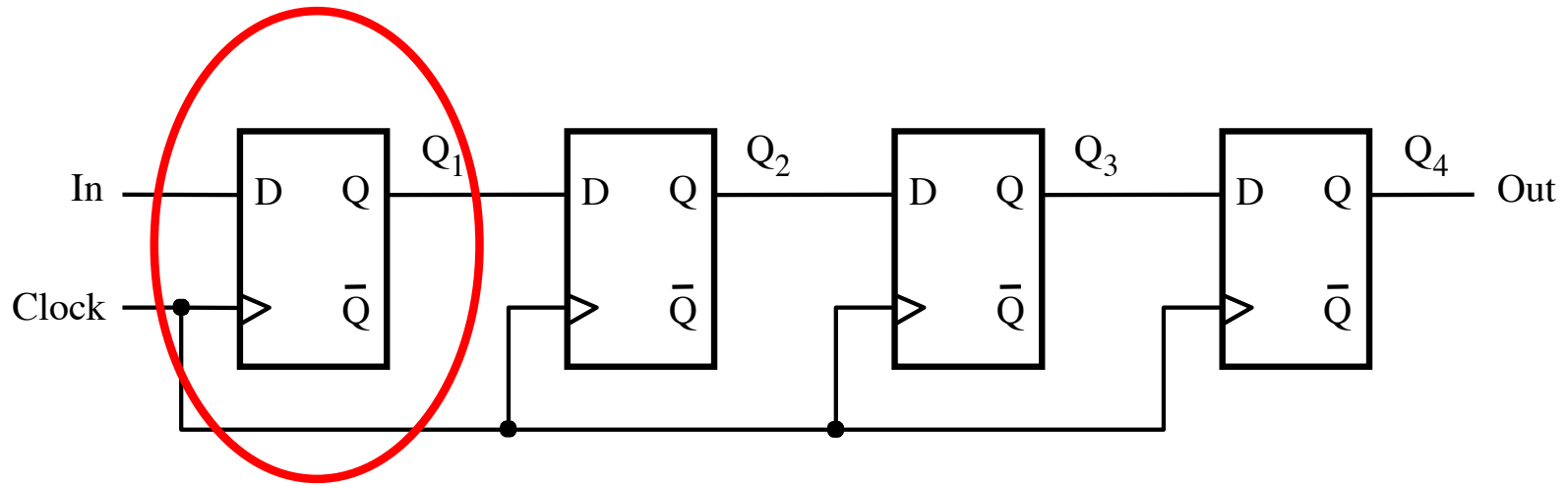
4-bit Register



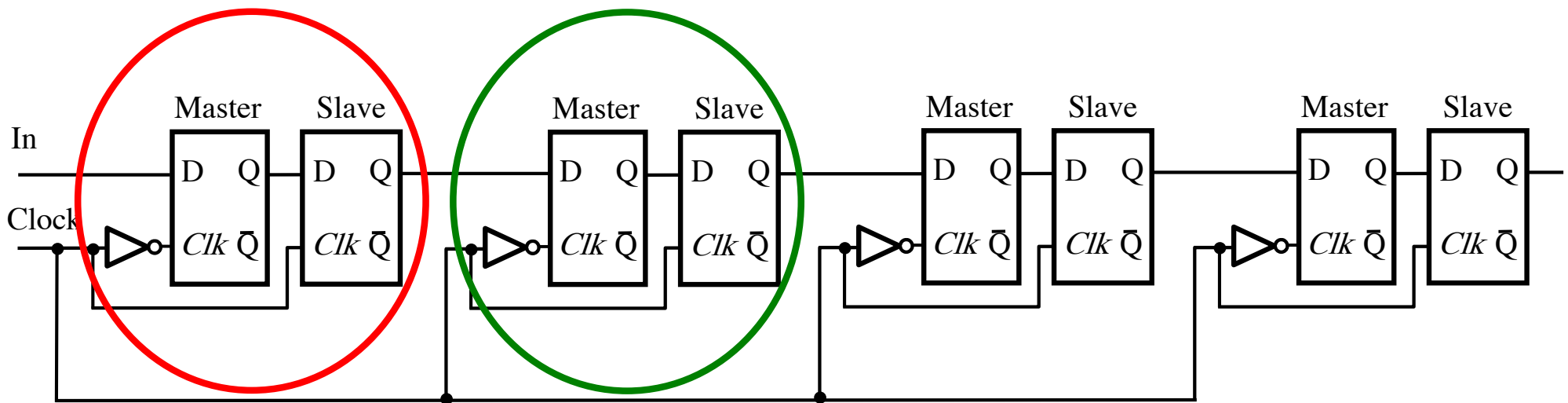
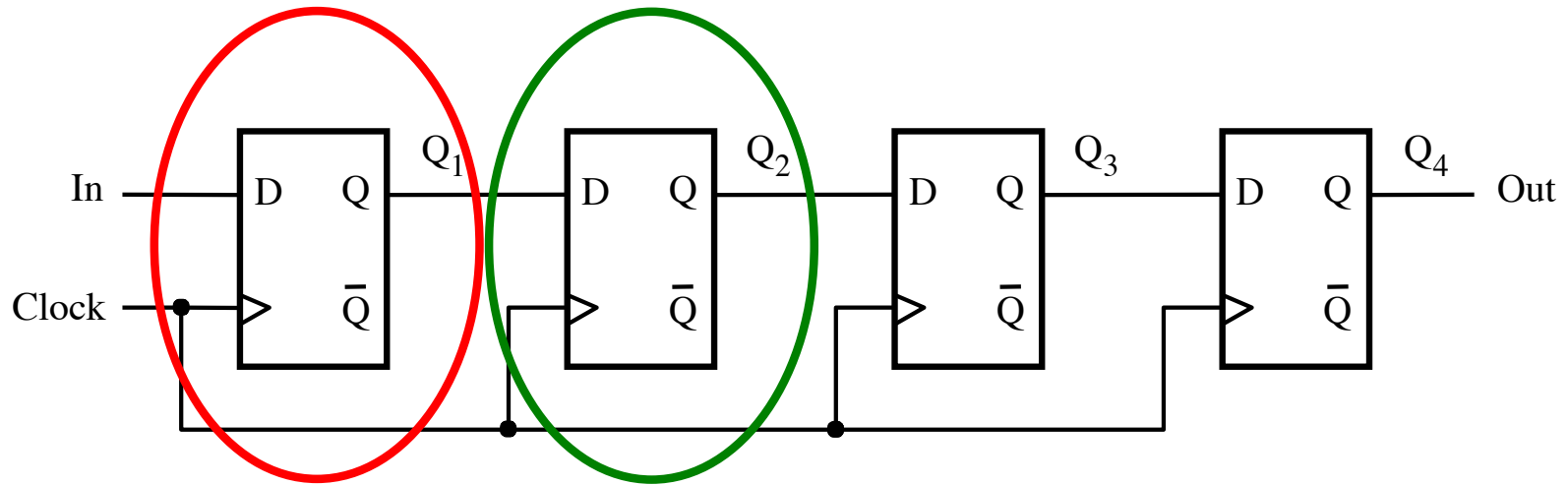
A simple shift register



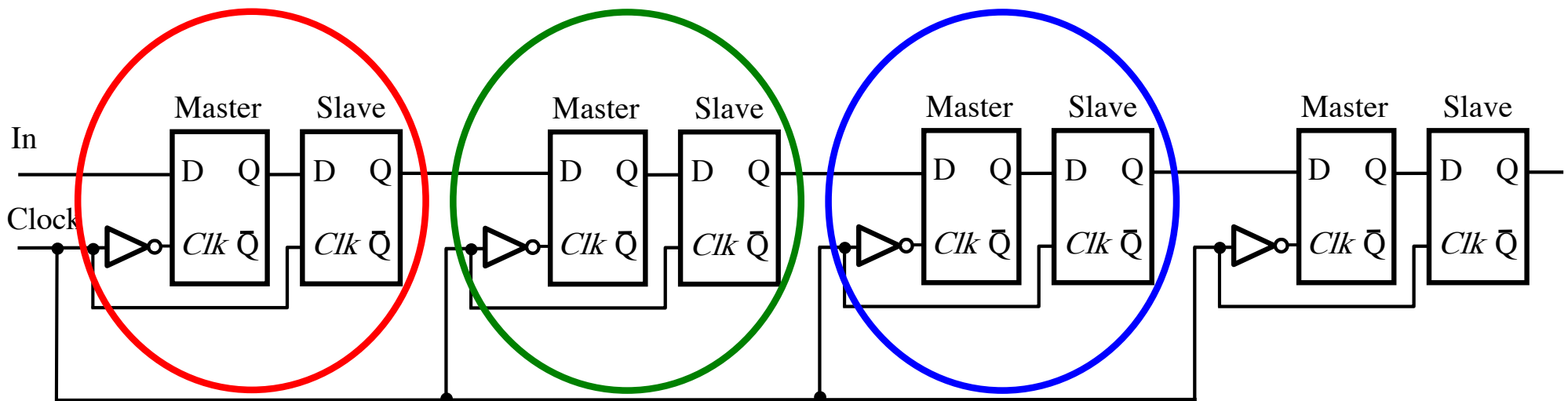
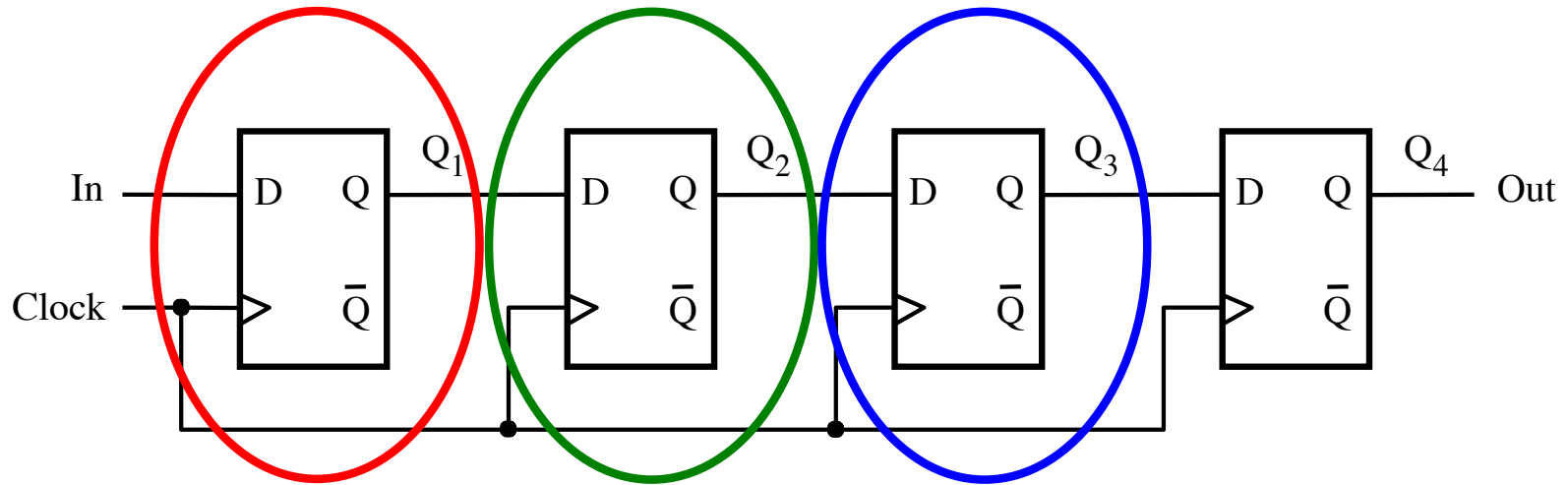
A simple shift register



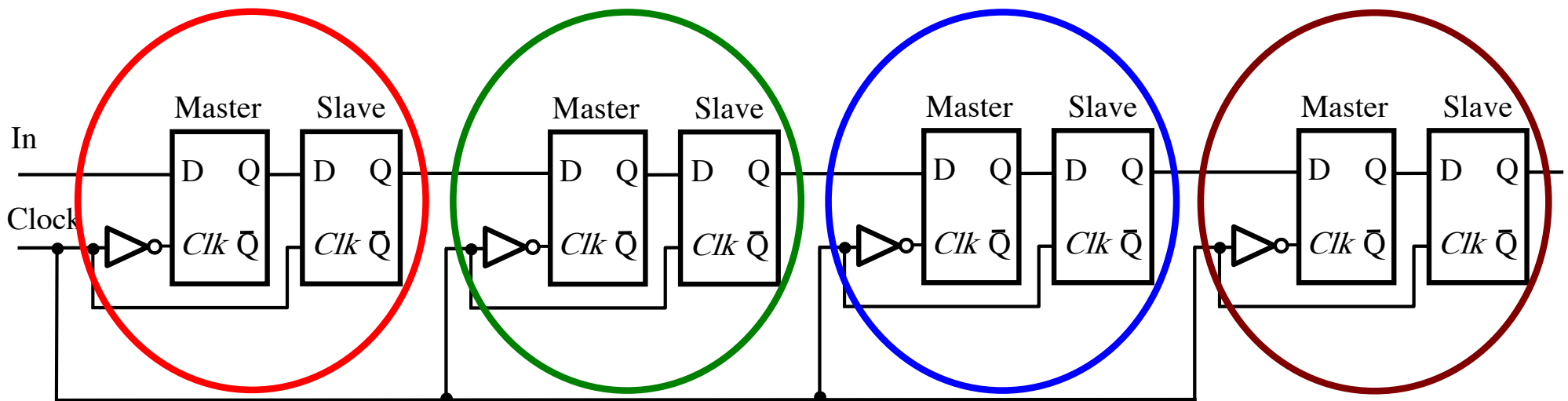
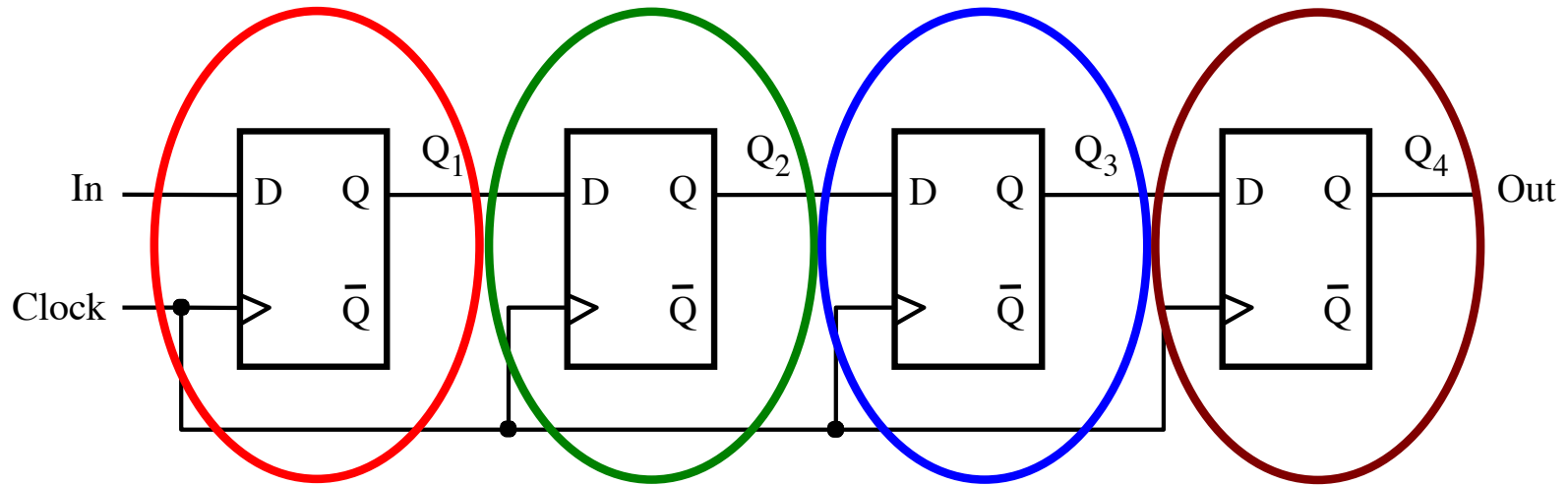
A simple shift register



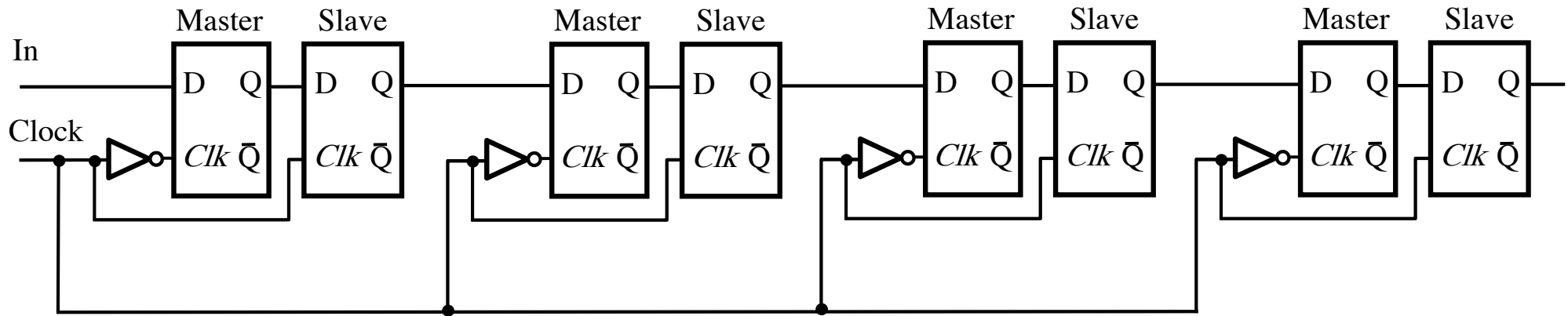
A simple shift register



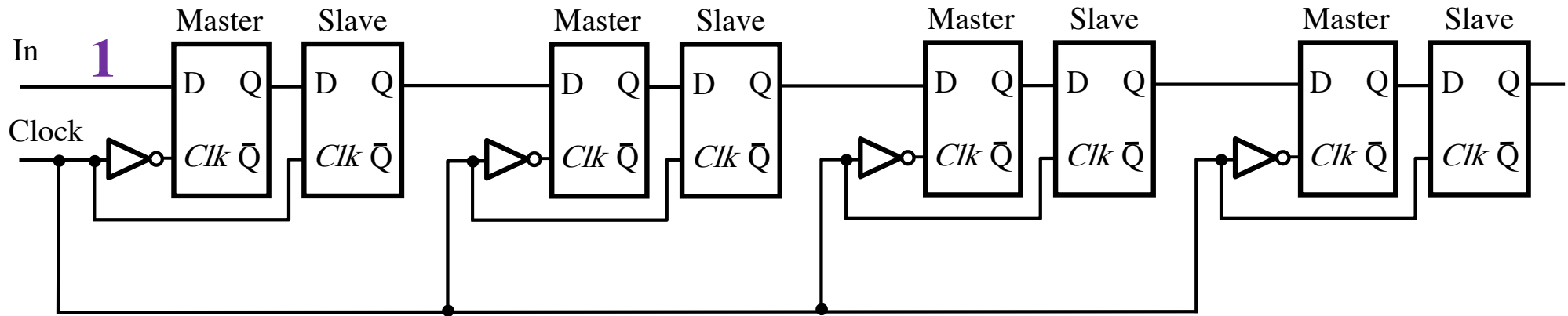
A simple shift register



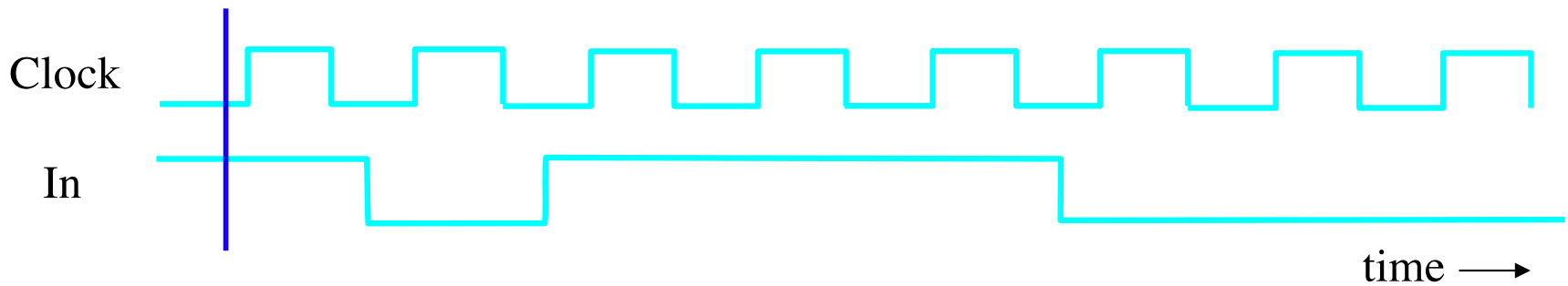
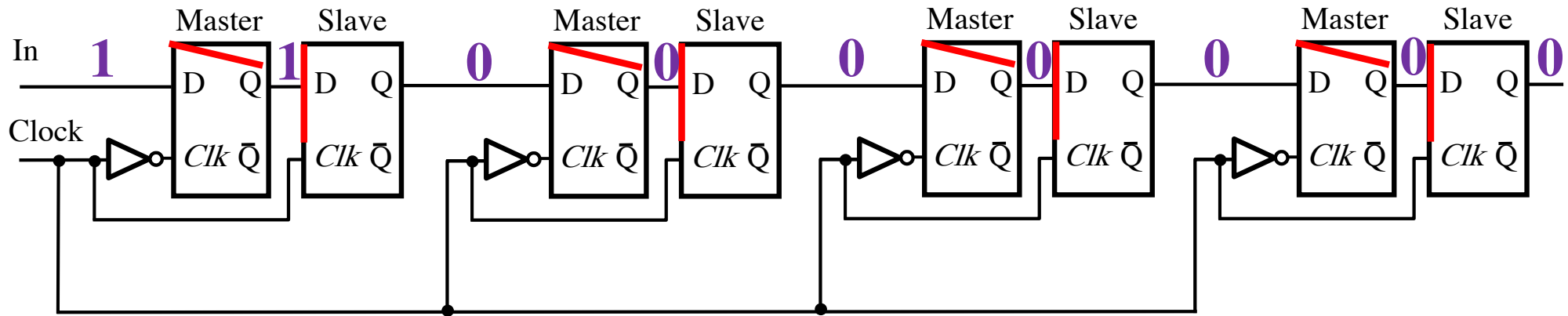
Simulating a shift register



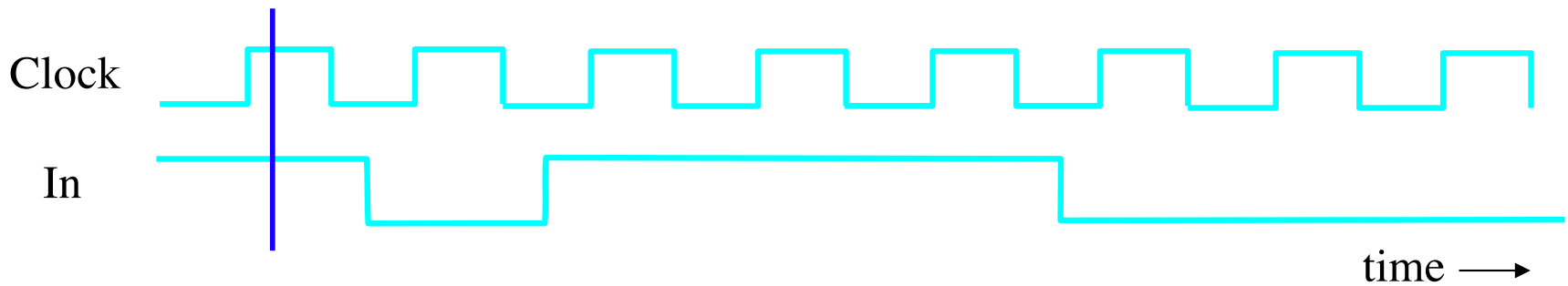
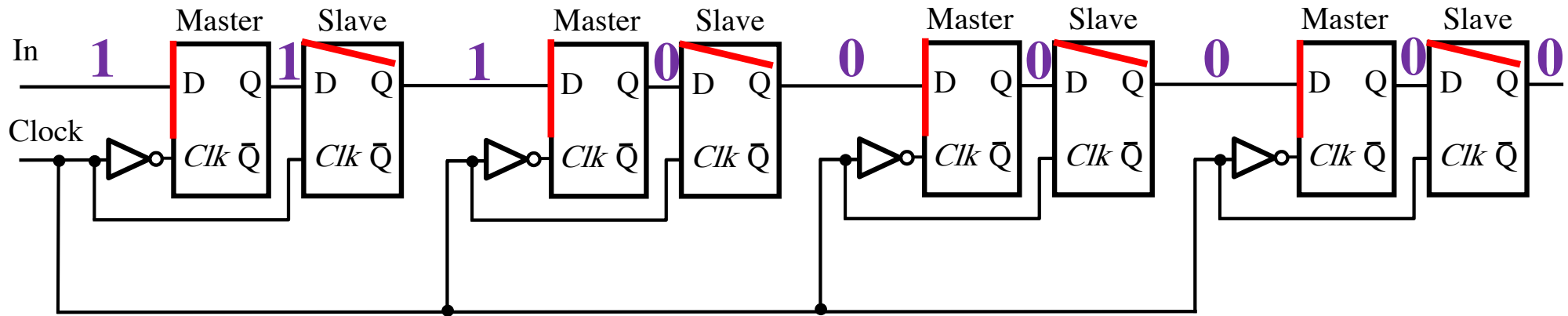
Simulating a shift register



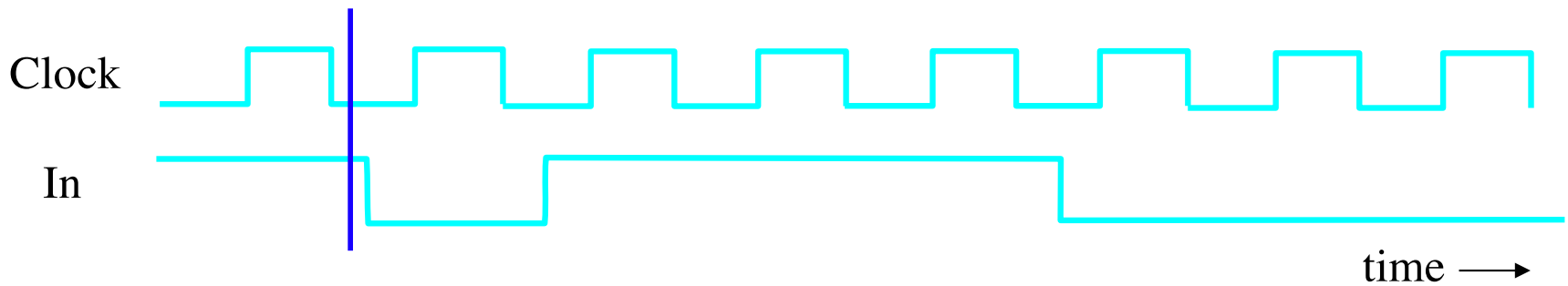
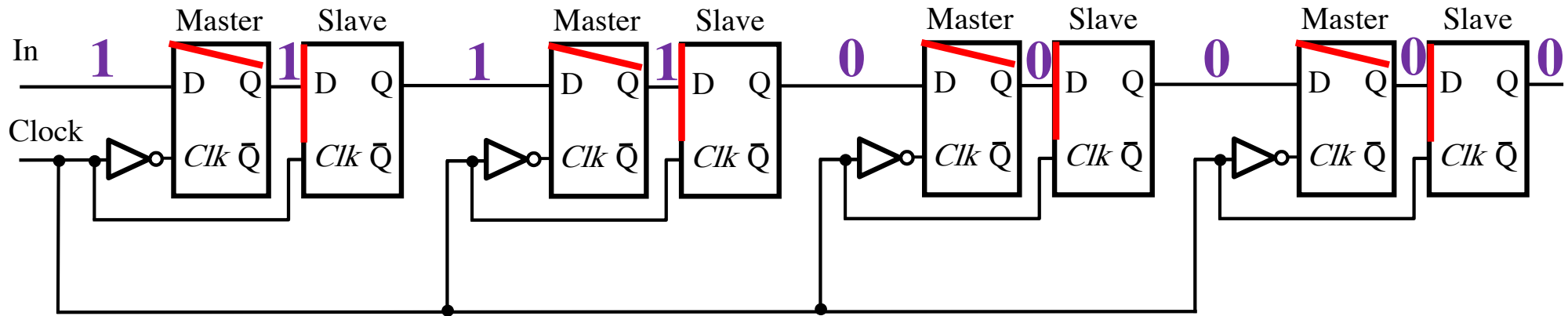
Simulating a shift register



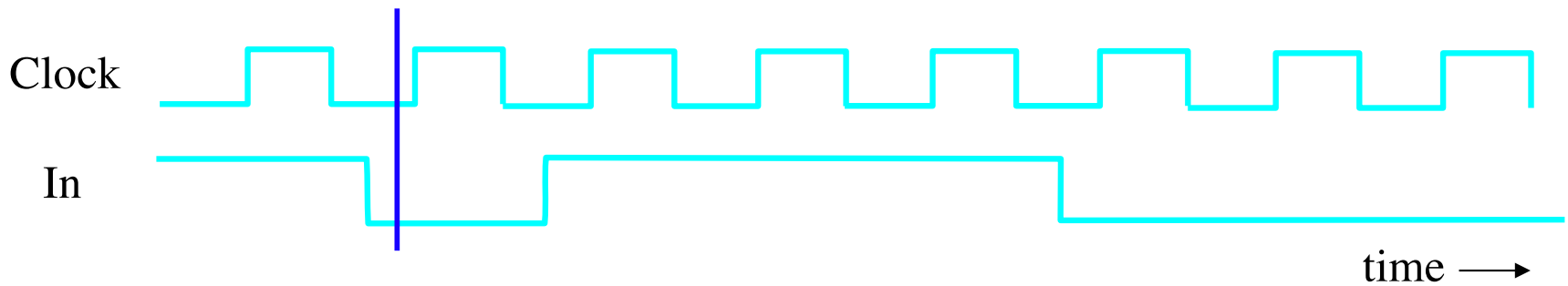
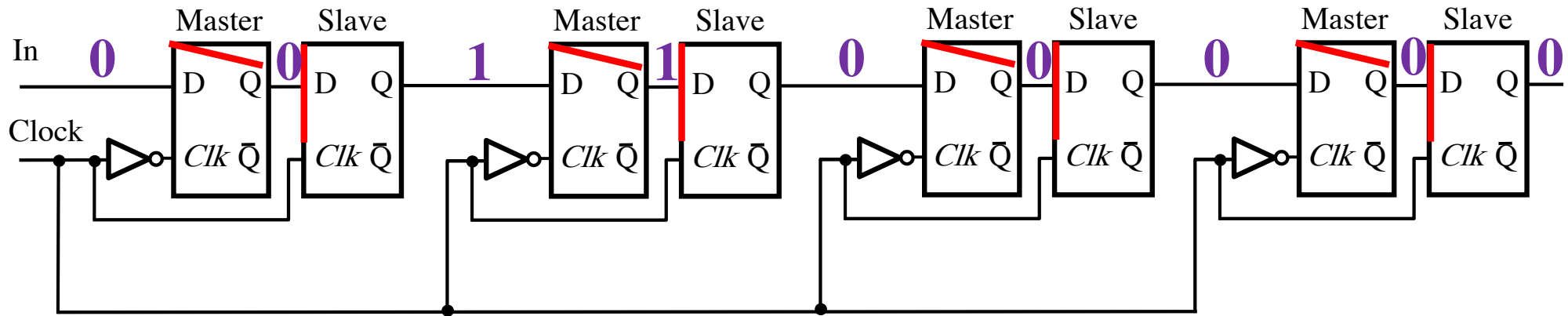
Simulating a shift register



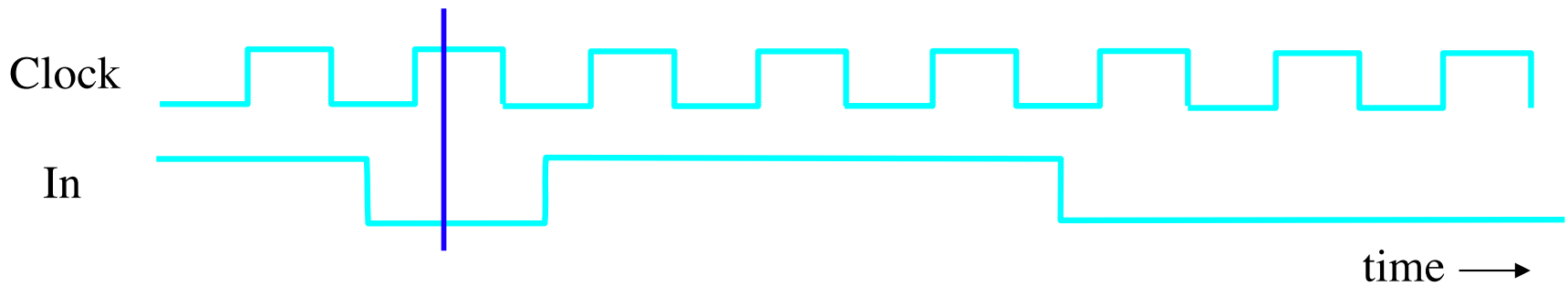
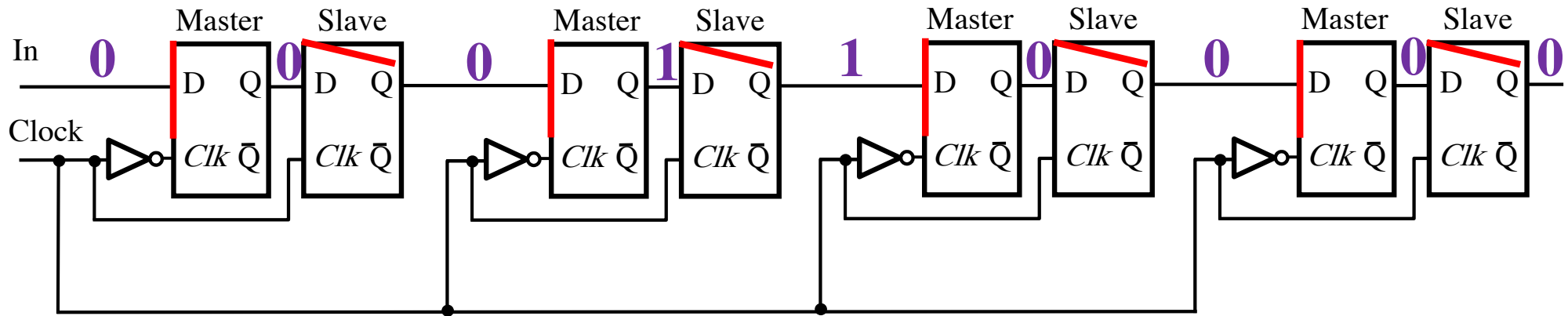
Simulating a shift register



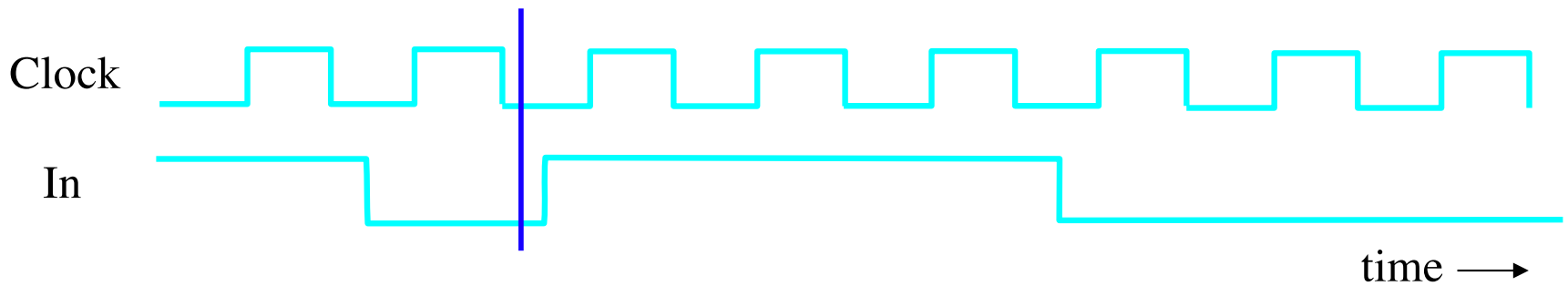
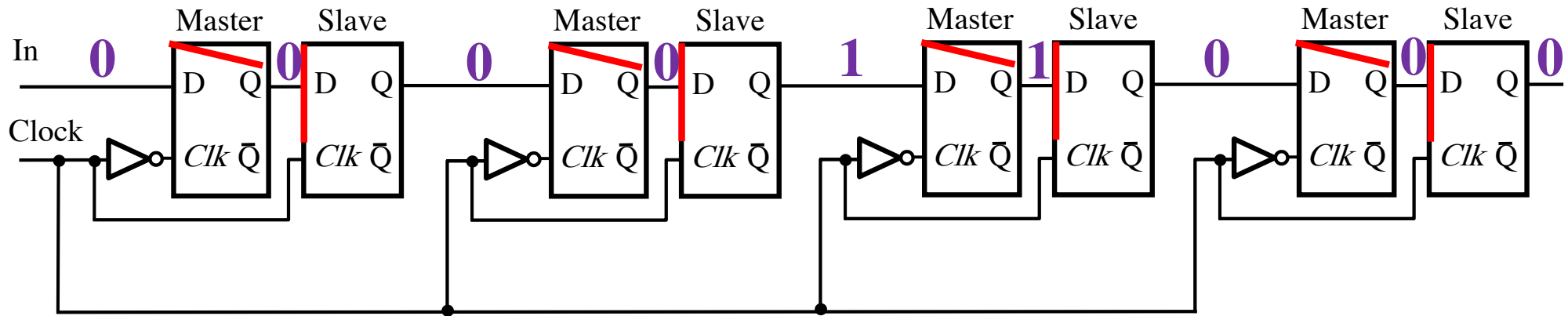
Simulating a shift register



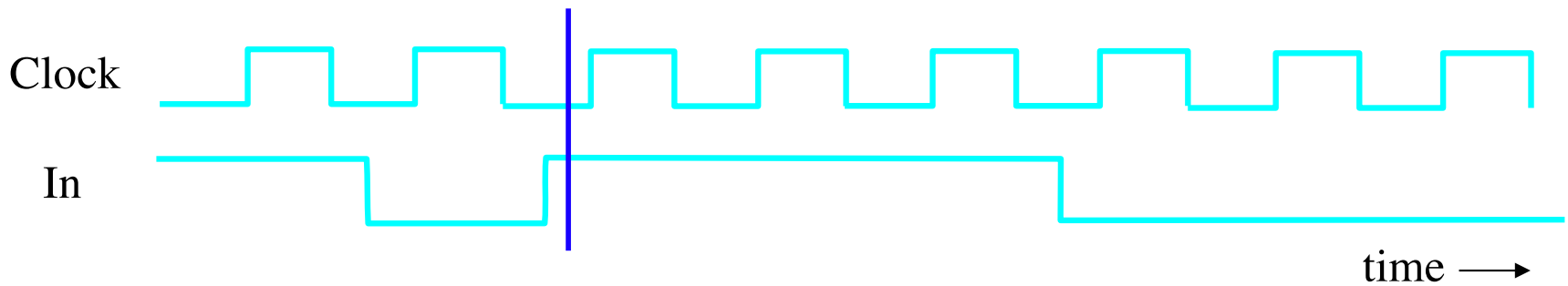
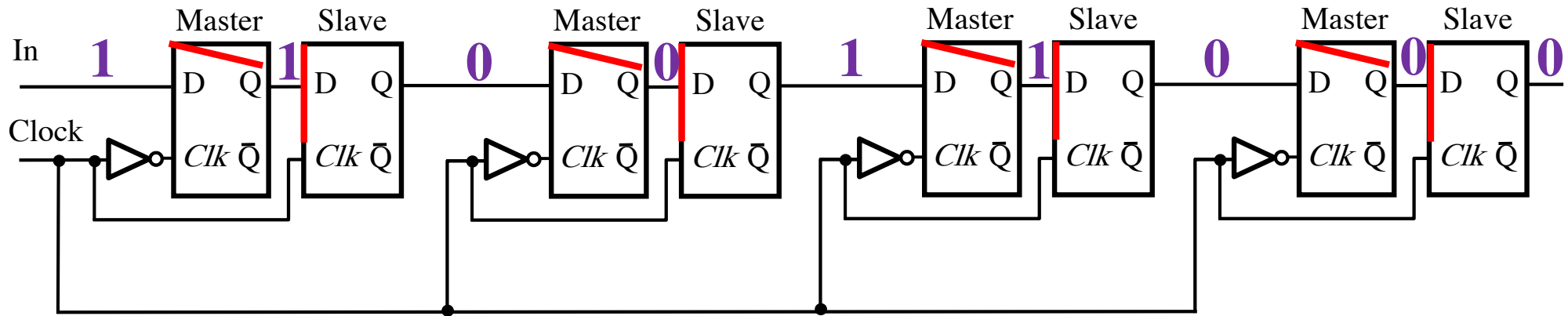
Simulating a shift register



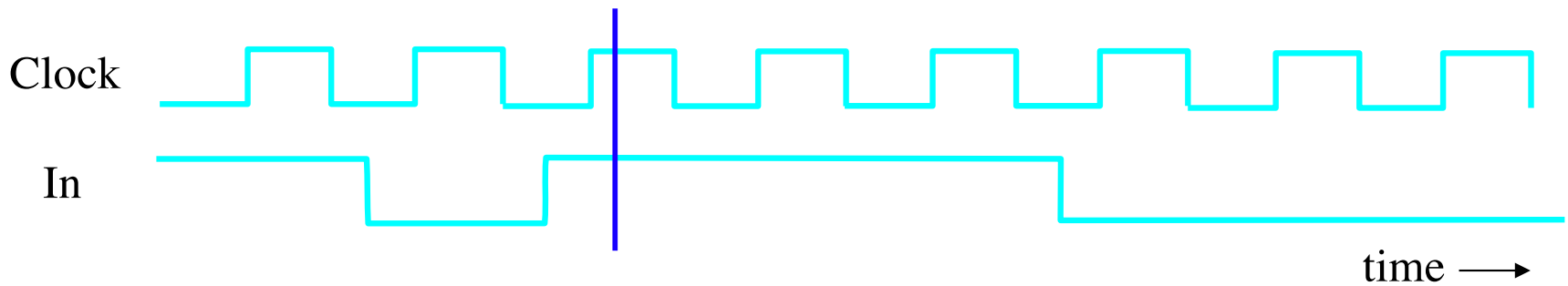
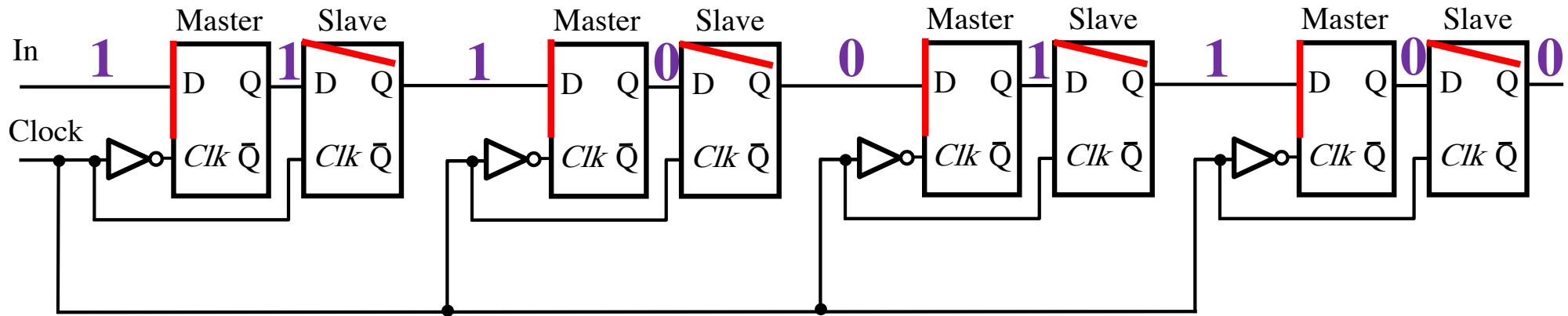
Simulating a shift register



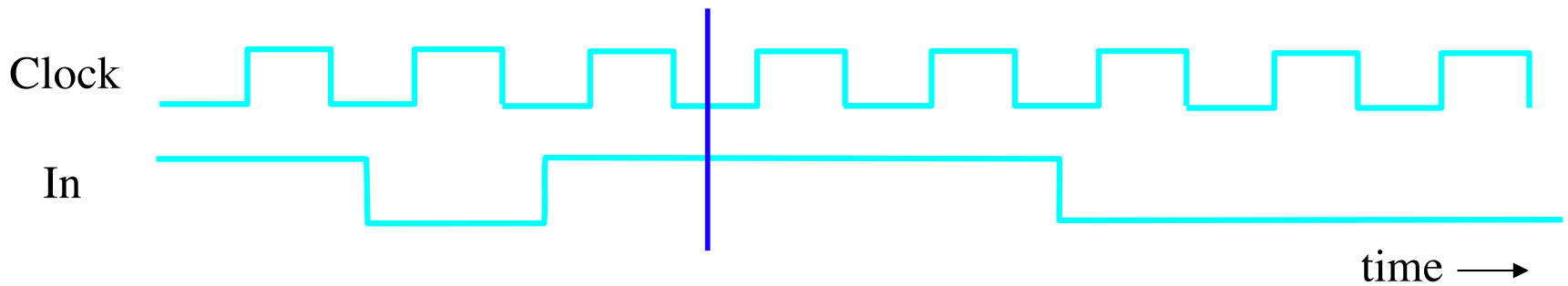
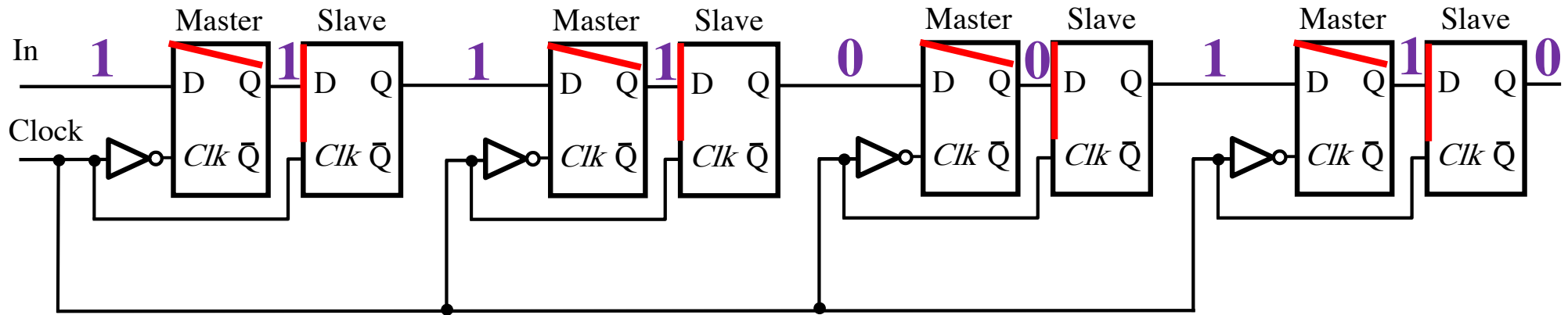
Simulating a shift register



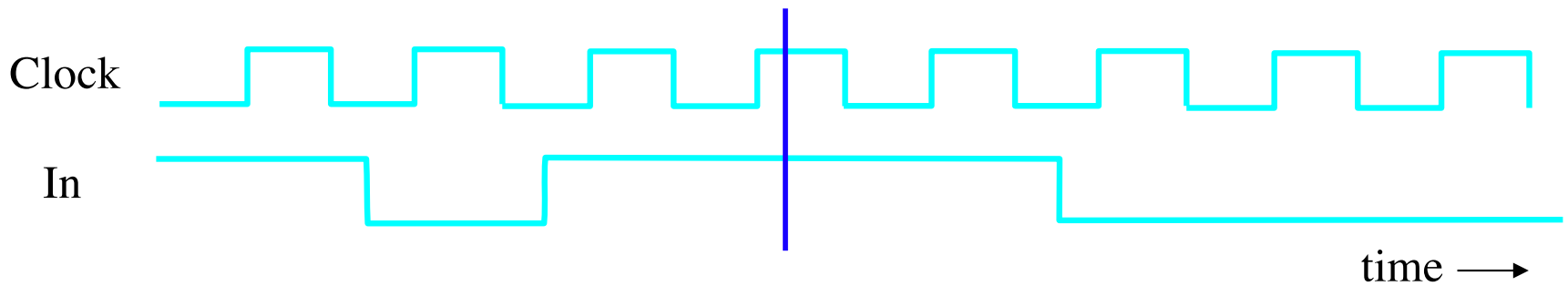
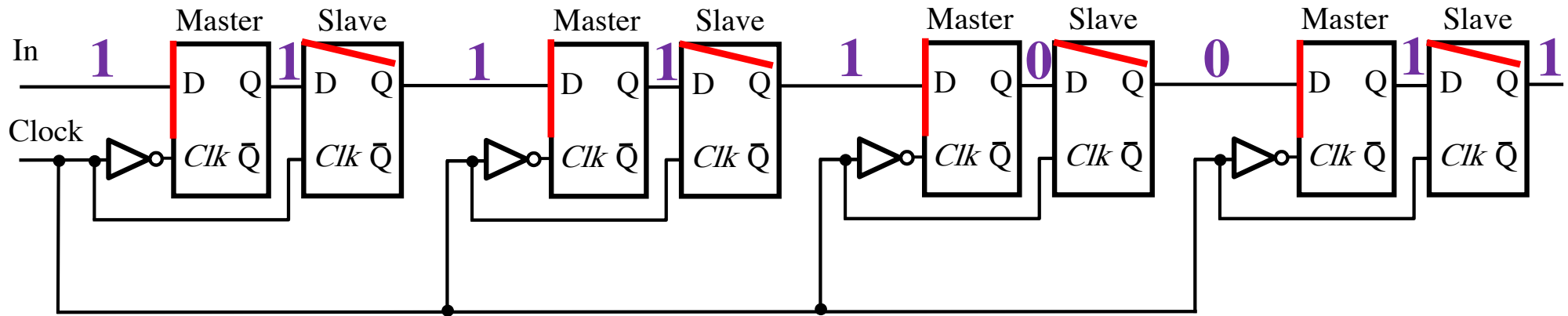
Simulating a shift register



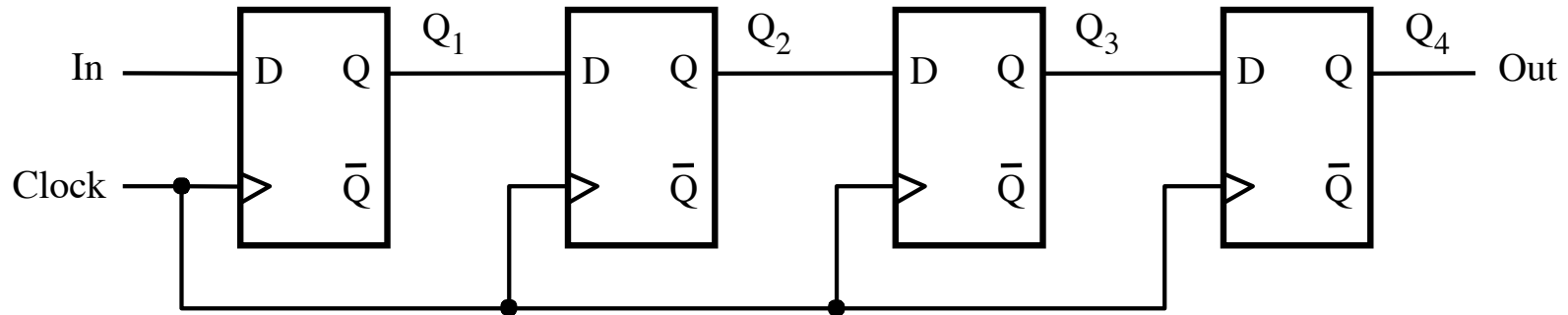
Simulating a shift register



Simulating a shift register



A simple shift register



(a) Circuit

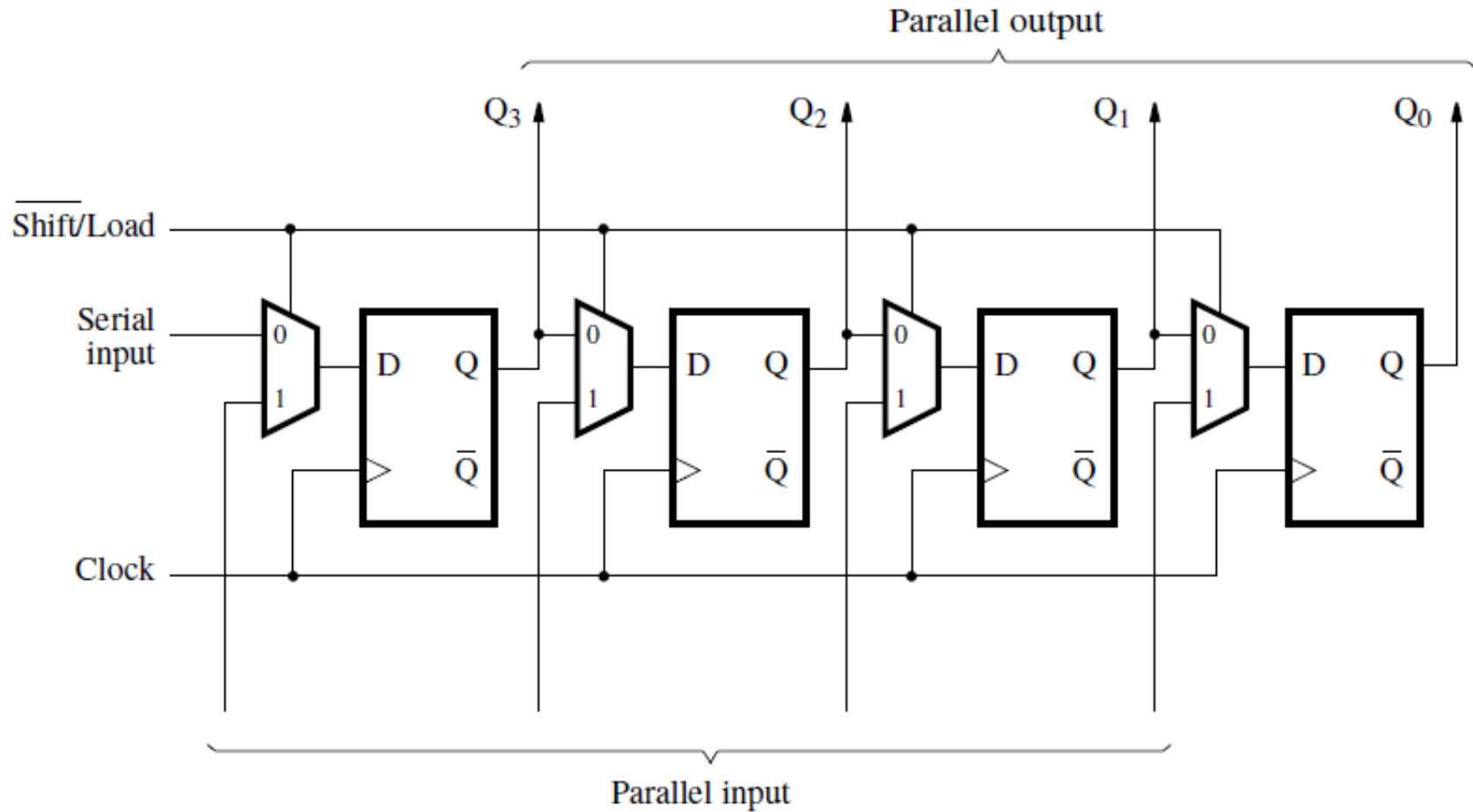
	In	Q ₁	Q ₂	Q ₃	Q ₄ = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

This simulation goes only up to here

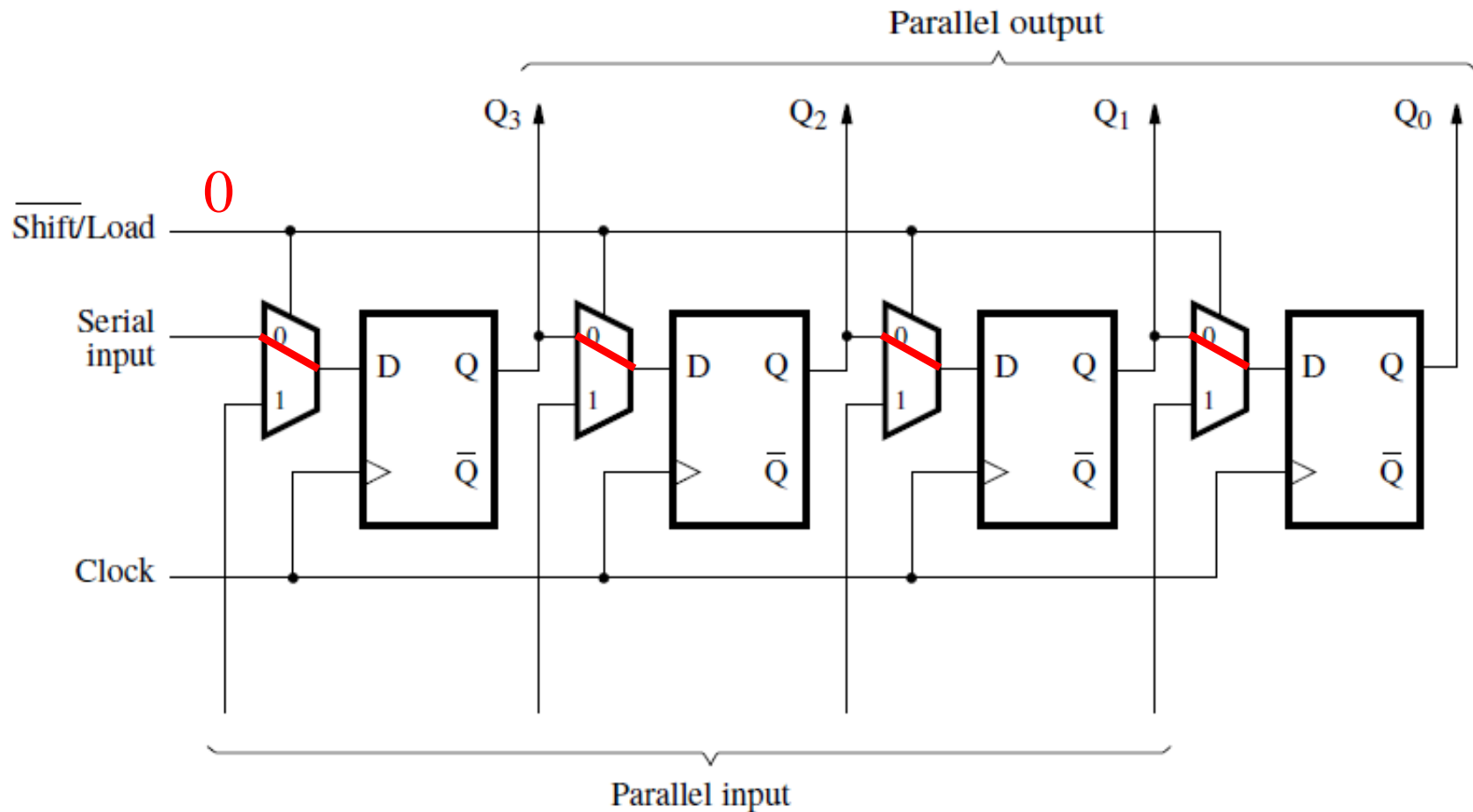
(b) A sample sequence

Parallel-Access Shift Register

Parallel-access shift register

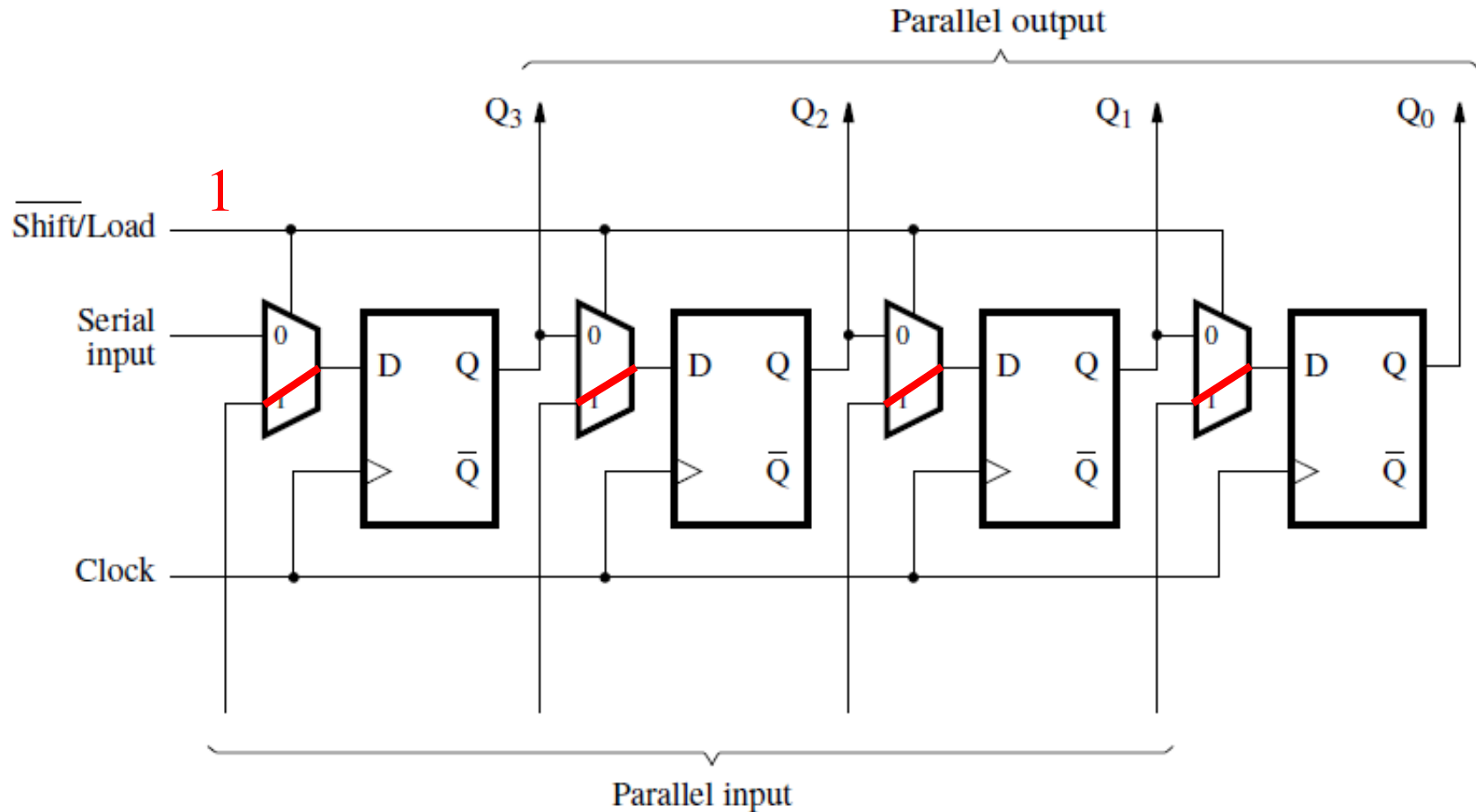


Parallel-access shift register



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

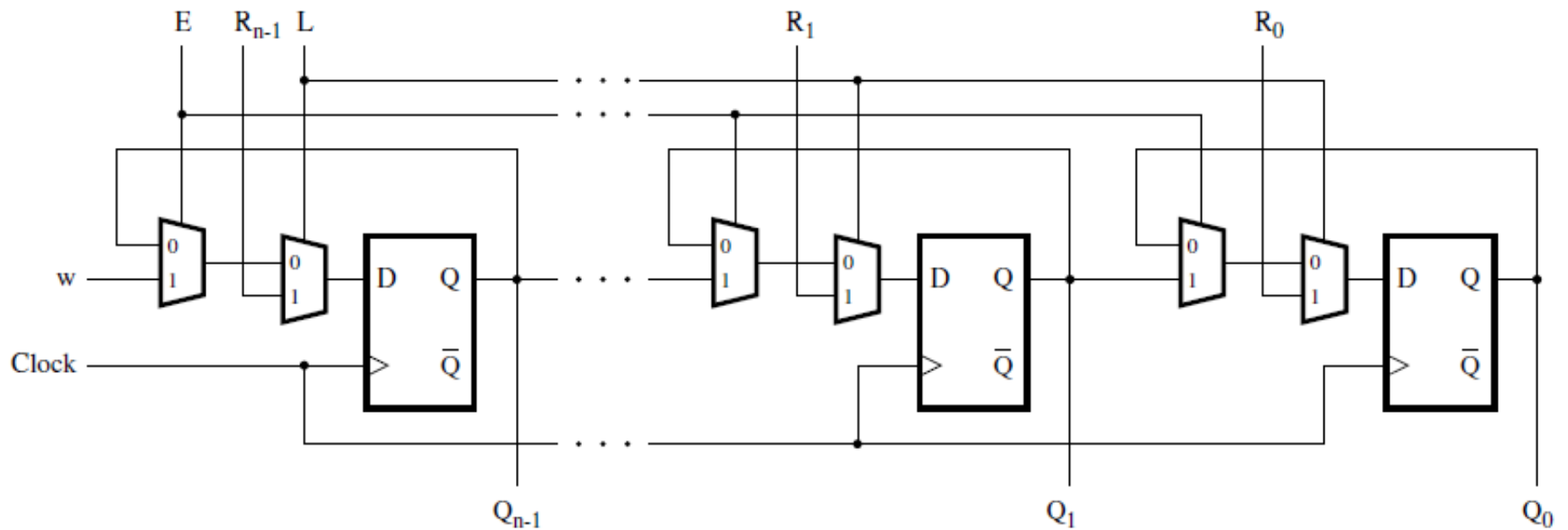
Parallel-access shift register



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

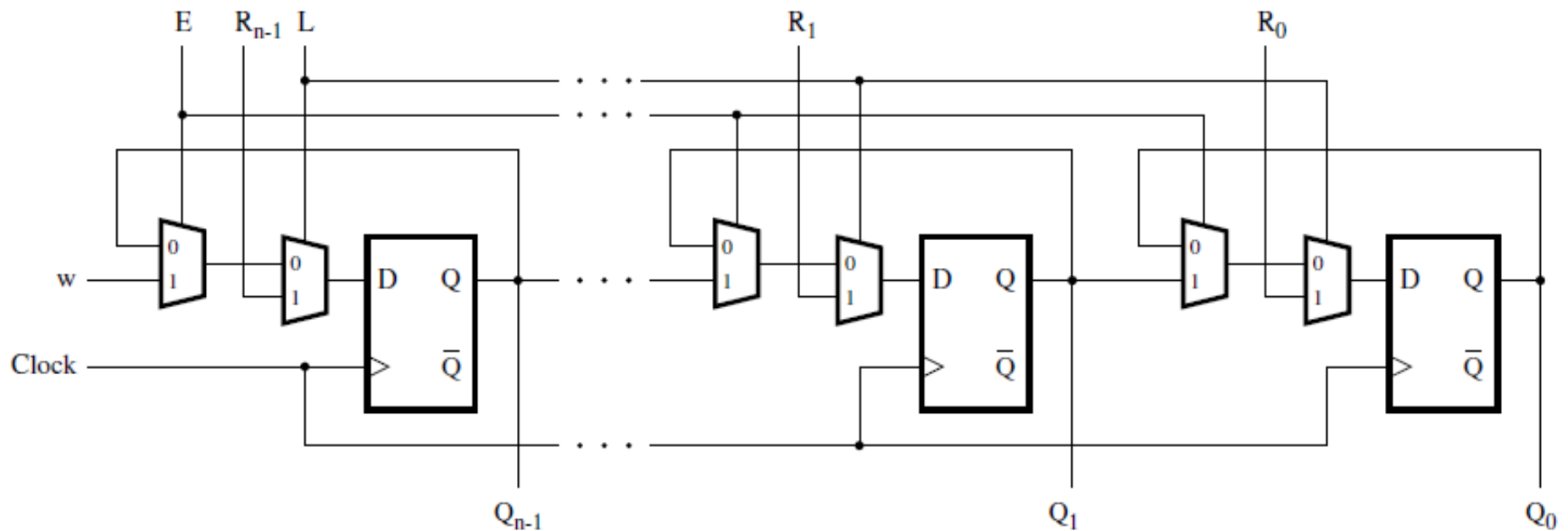
Shift Register With Parallel Load and Enable

A shift register with parallel load and enable control inputs



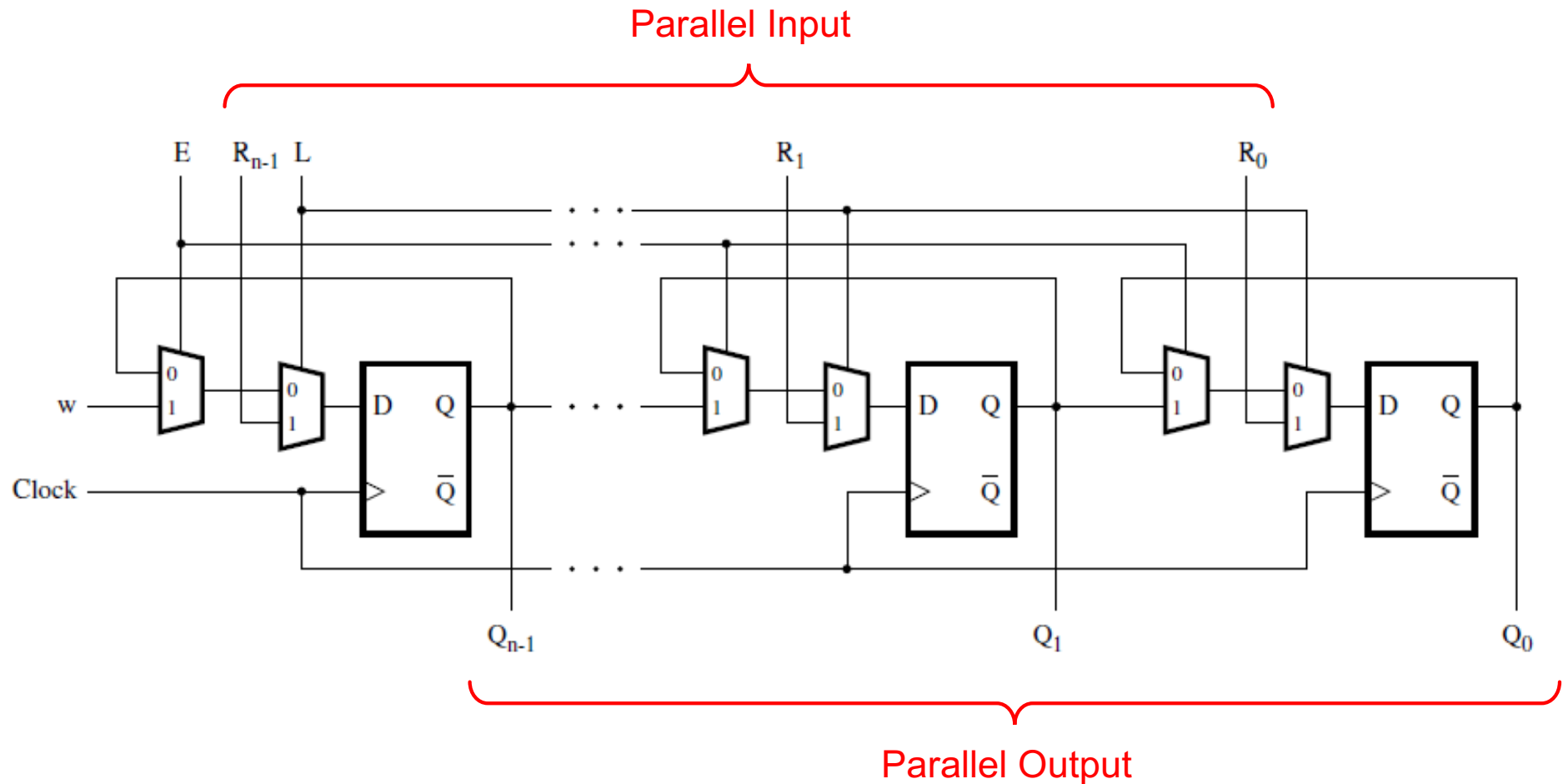
[Figure 5.59 from the textbook]

A shift register with parallel load and enable control inputs

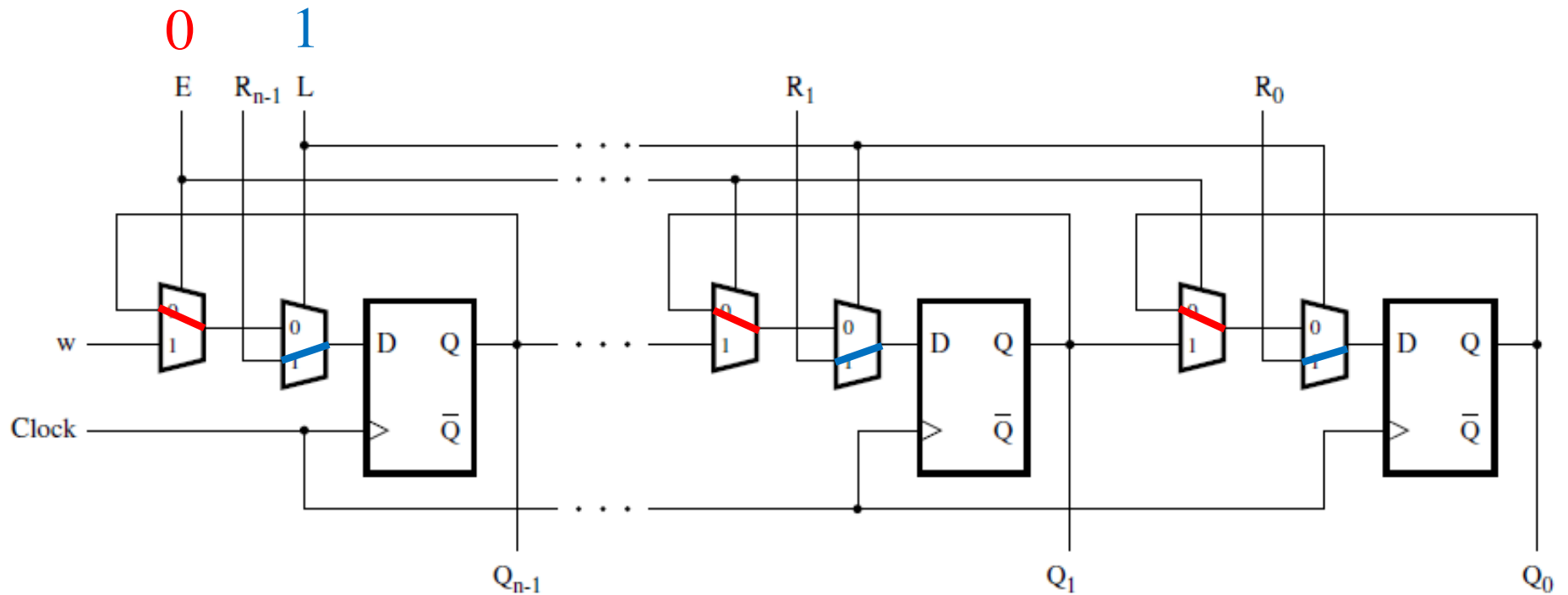


The directions of the input and output lines are switched relative to the previous slides.

A shift register with parallel load and enable control inputs

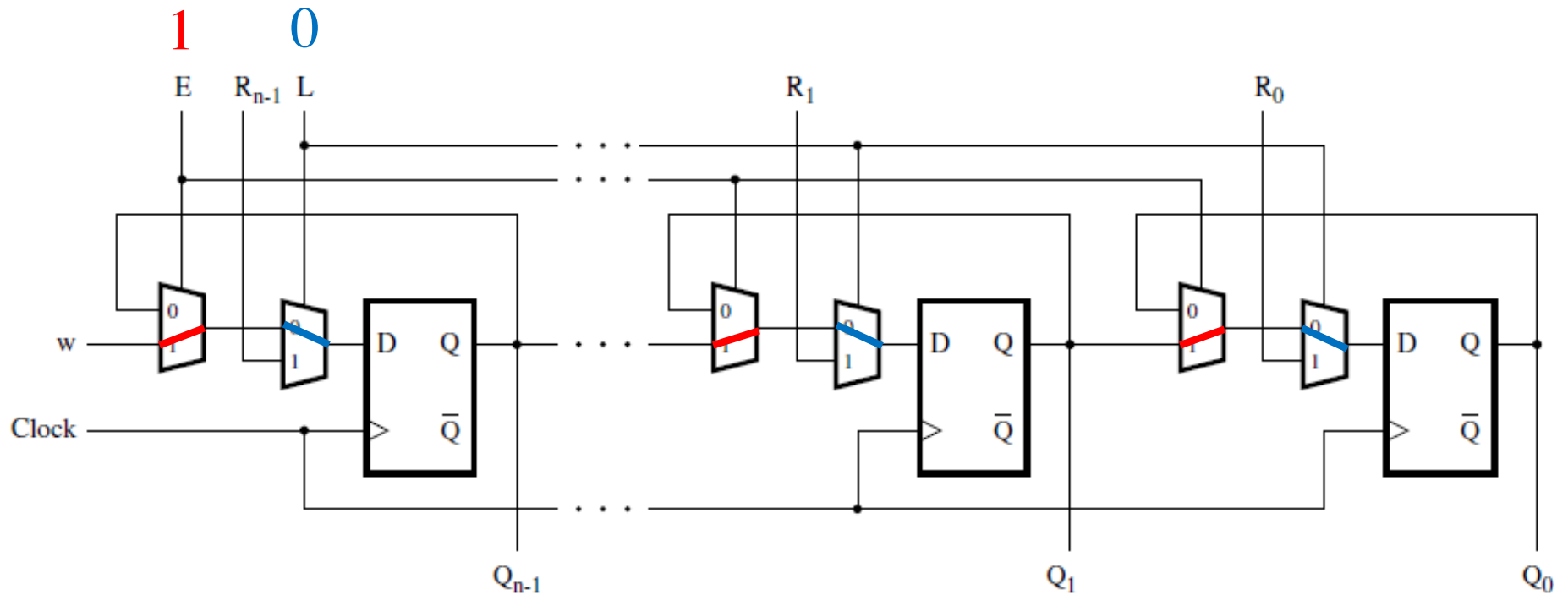


A shift register with parallel load and enable control inputs



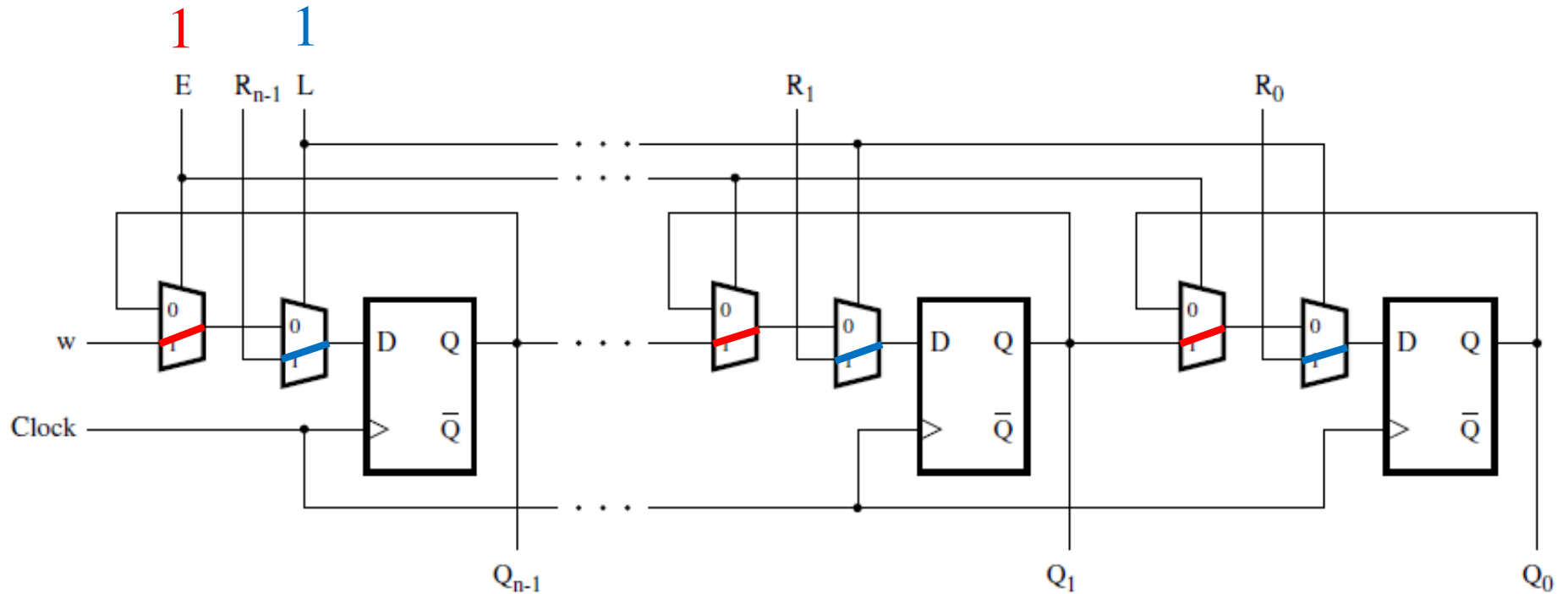
[Figure 5.59 from the textbook]

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]

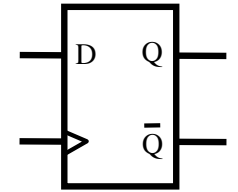
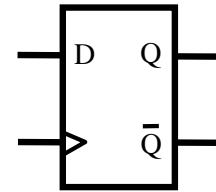
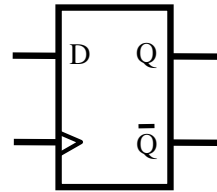
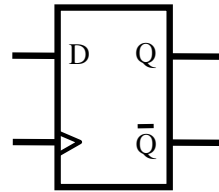
Parallel-access shift left / right register

Parallel-access shift left/right register

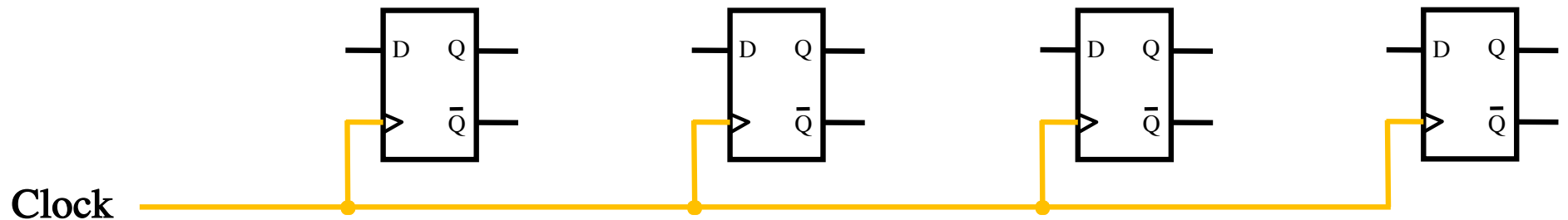
Complete the following circuit diagram to implement a 4-bit register that has both parallel load and shift left/right functionality. The register has two control inputs (C1 and C0), four parallel input lines (I3, I2, I1, and I0), and four output lines (Q3, Q2, Q1, and Q0). Depending on the values of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

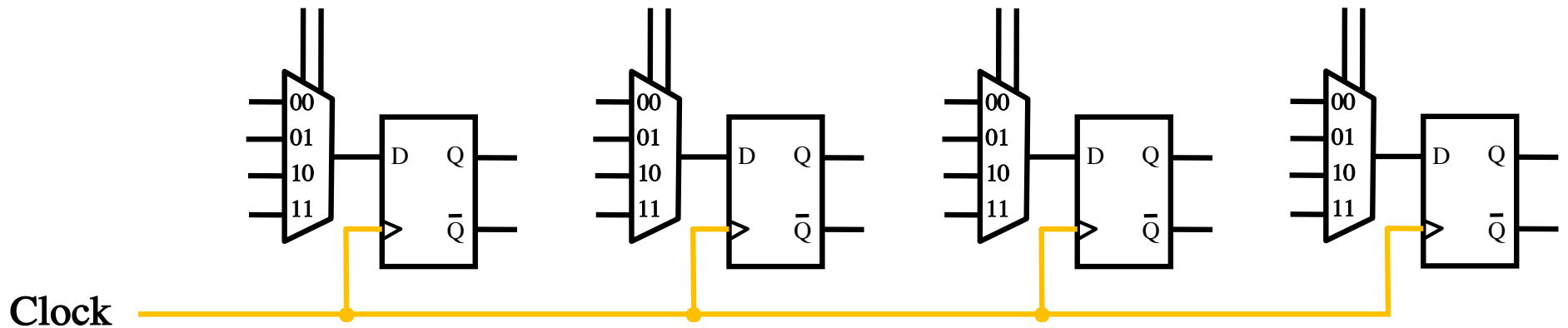
Parallel-access shift left/right register



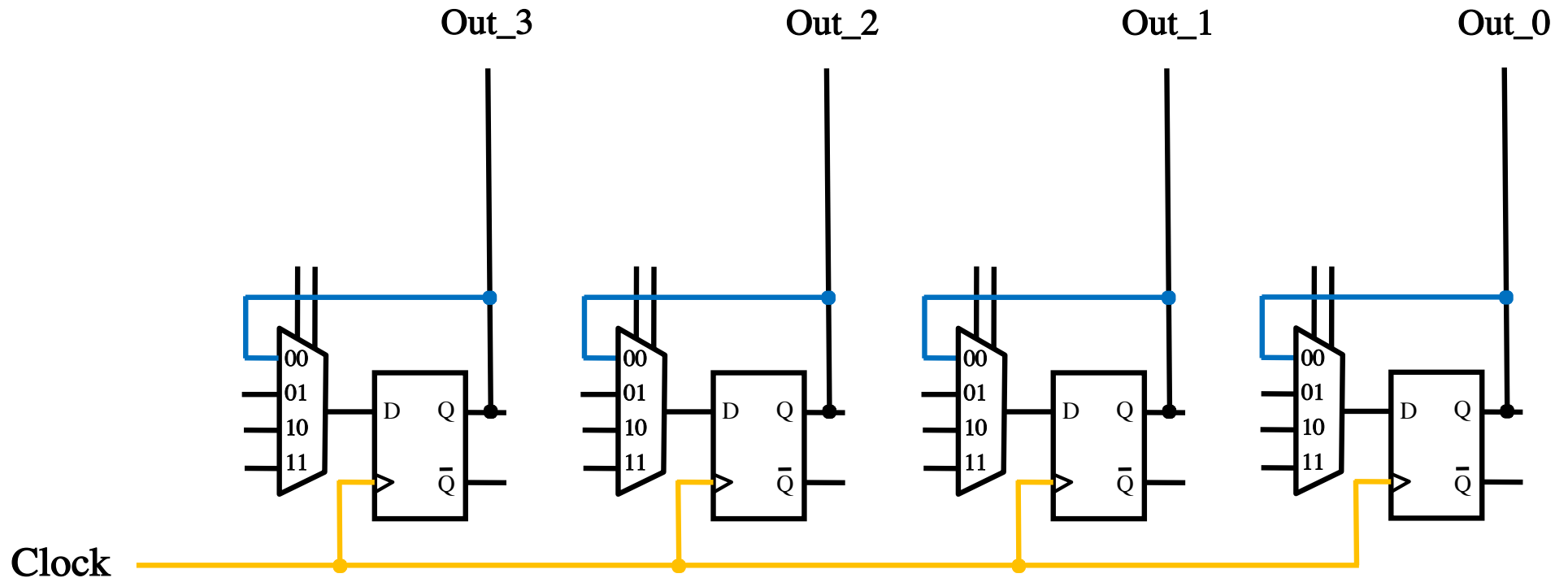
Parallel-access shift left/right register



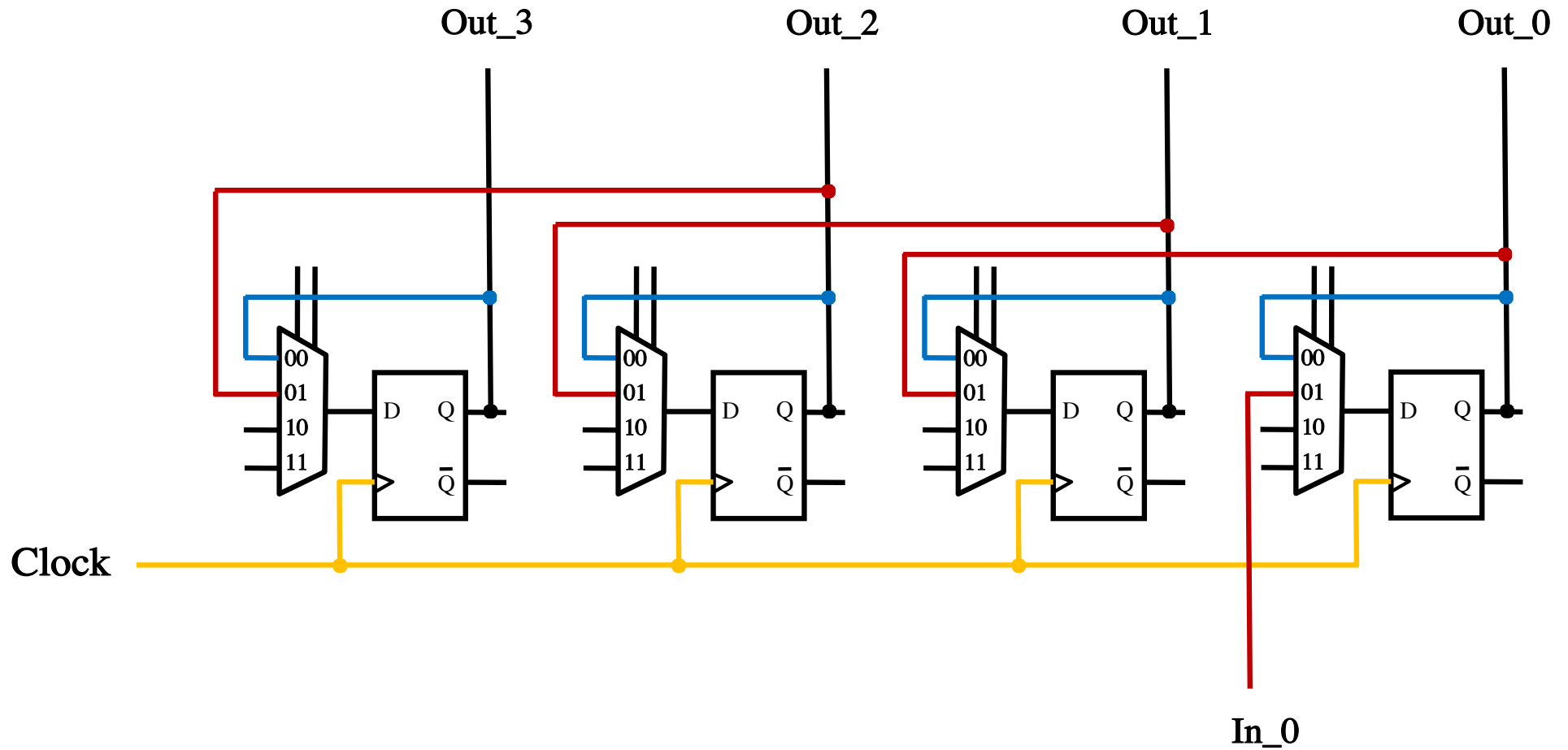
Parallel-access shift left/right register



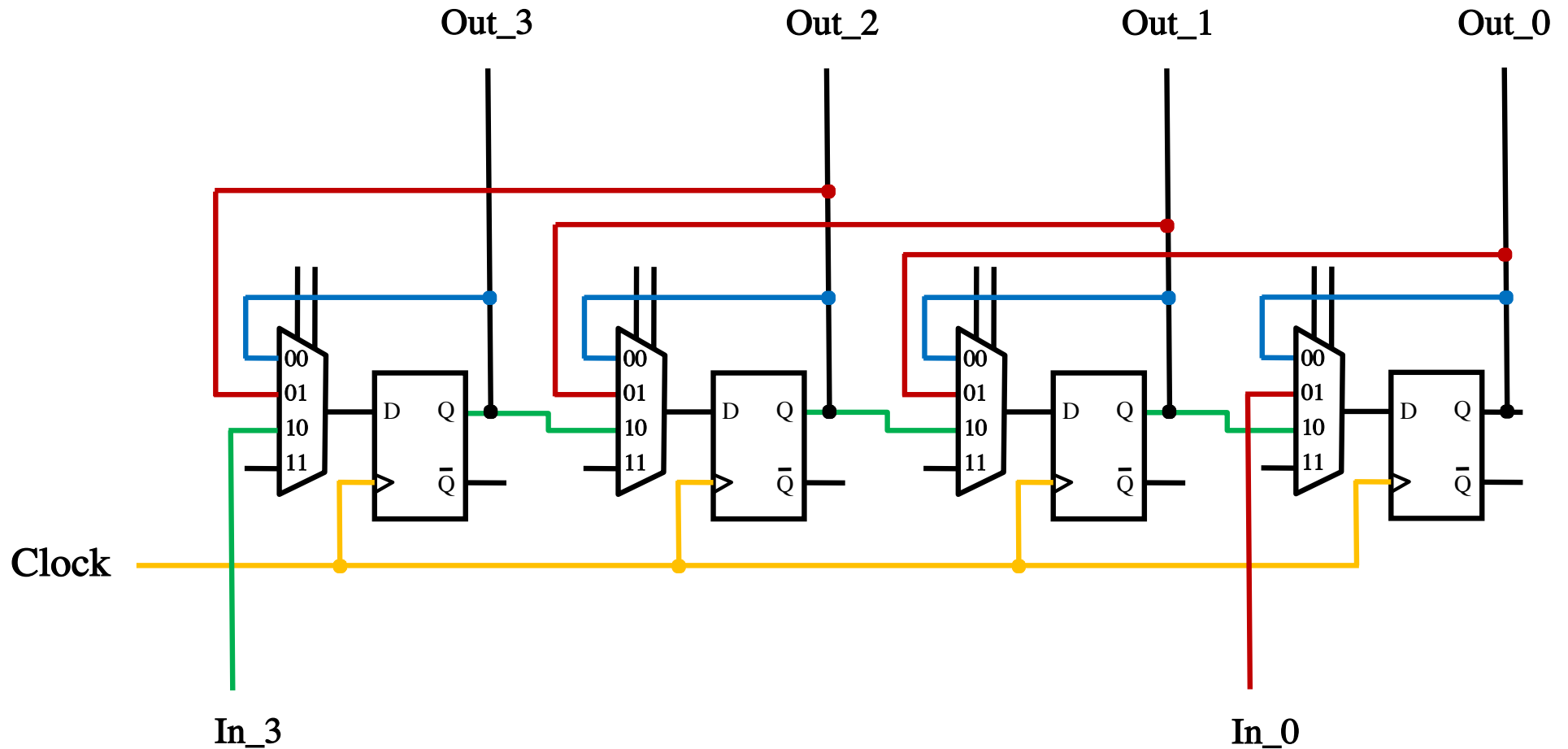
Parallel-access shift left/right register



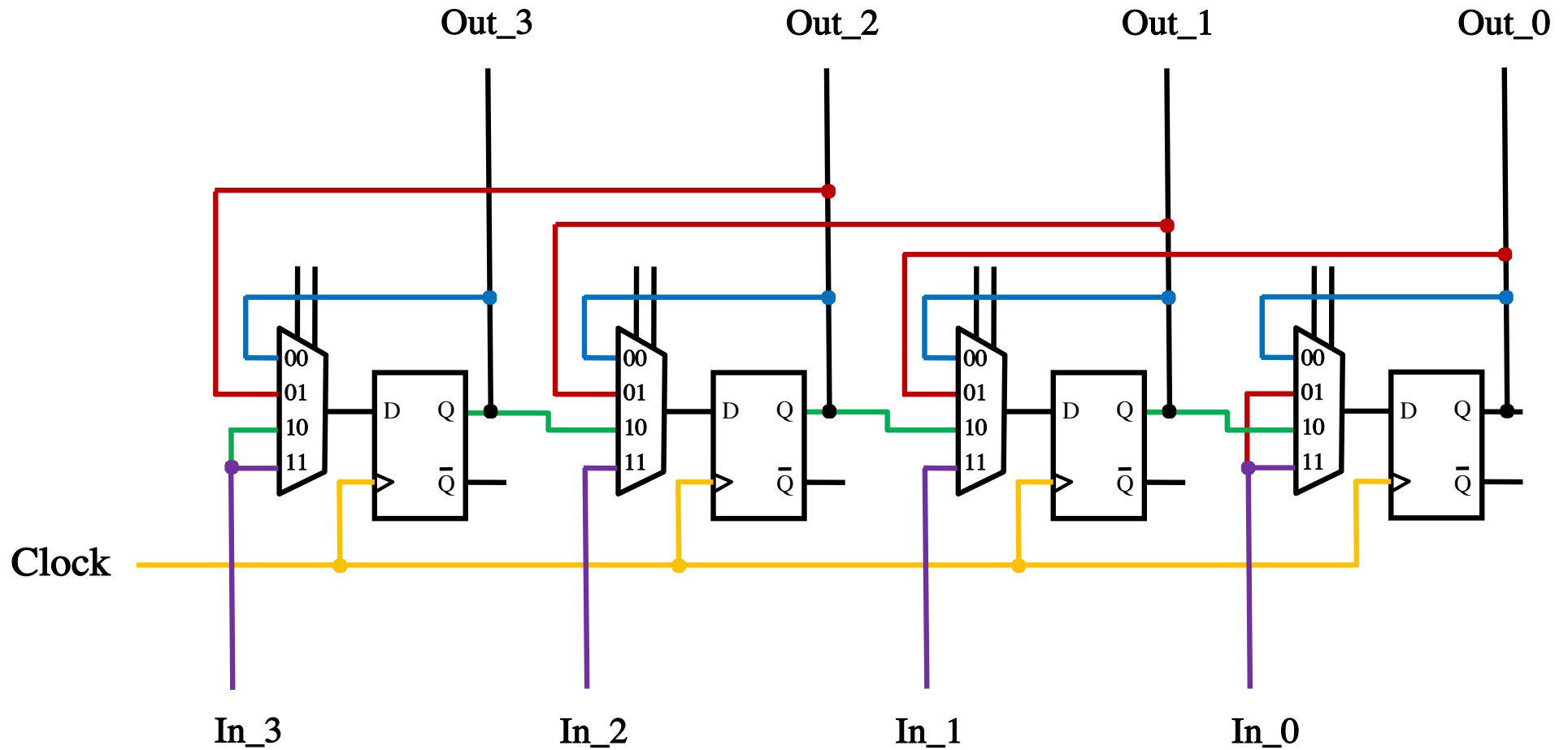
Parallel-access shift left/right register



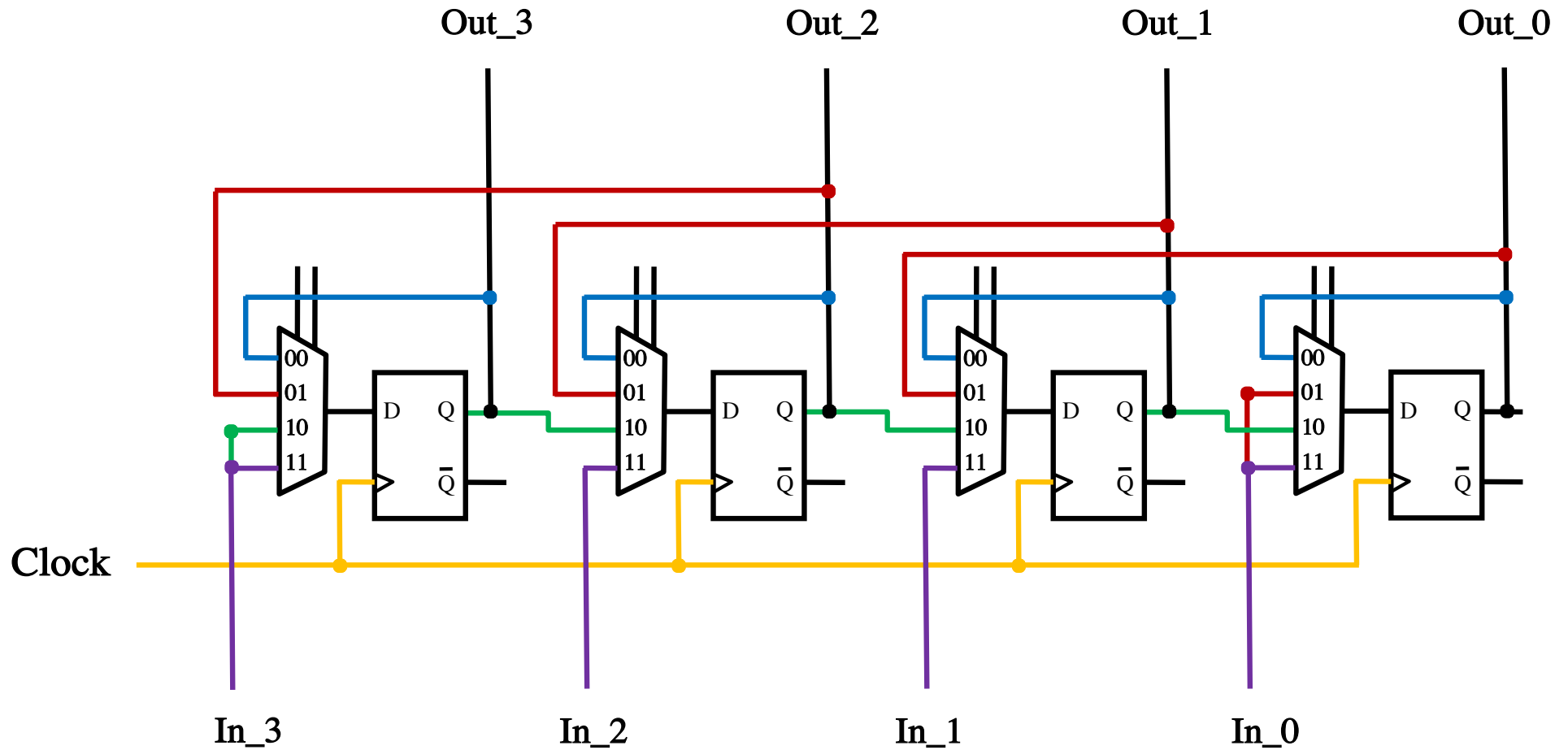
Parallel-access shift left/right register



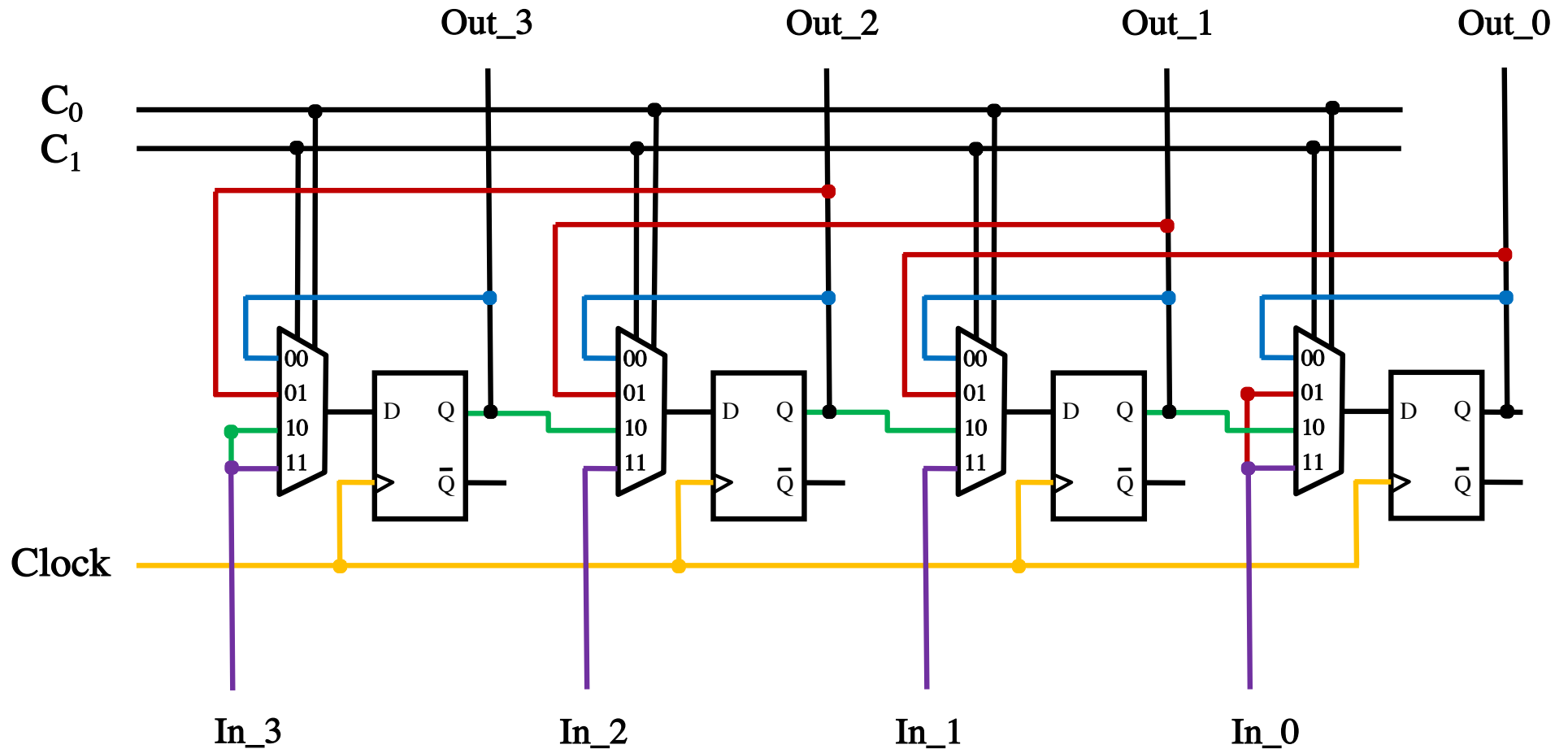
Parallel-access shift left/right register



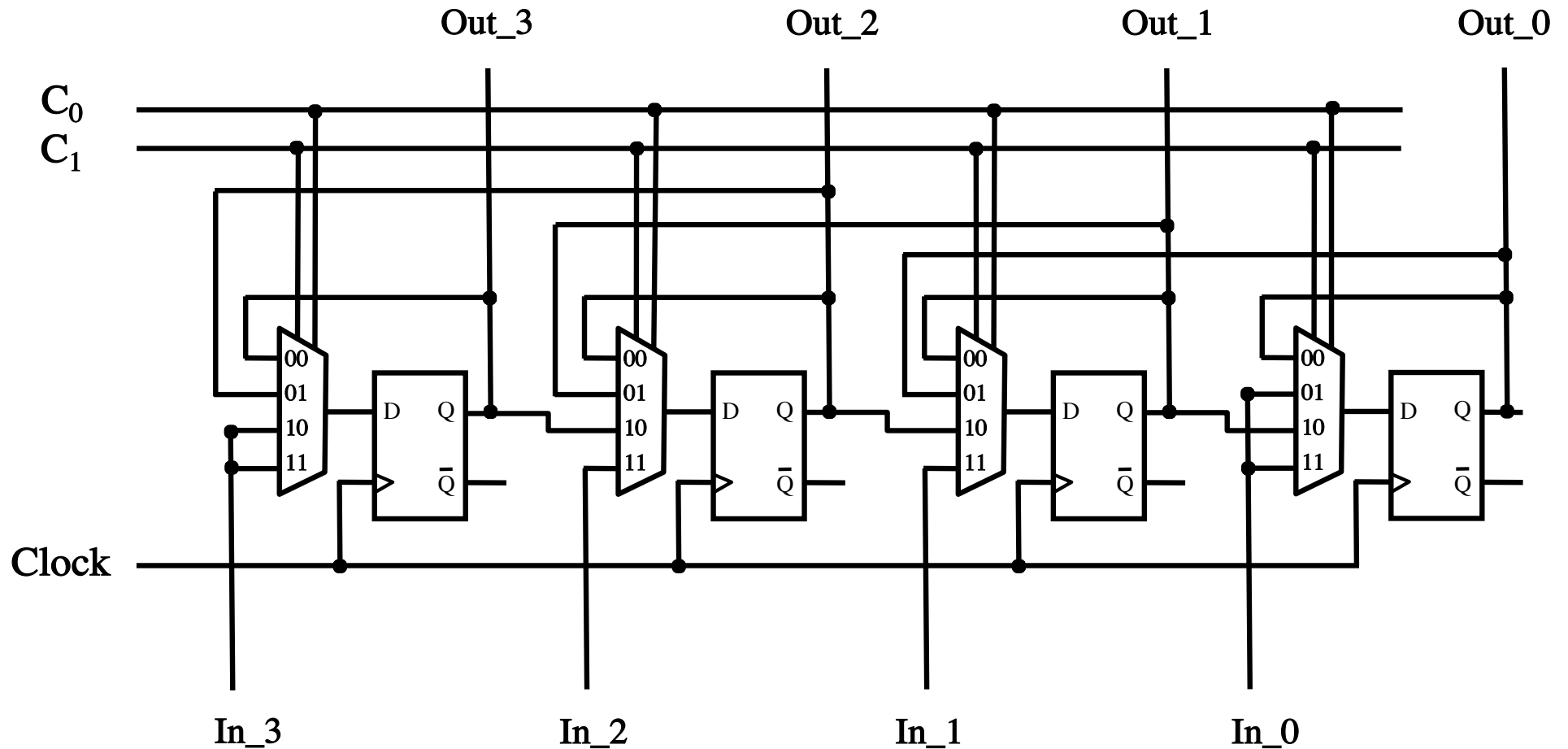
Parallel-access shift left/right register



Parallel-access shift left/right register



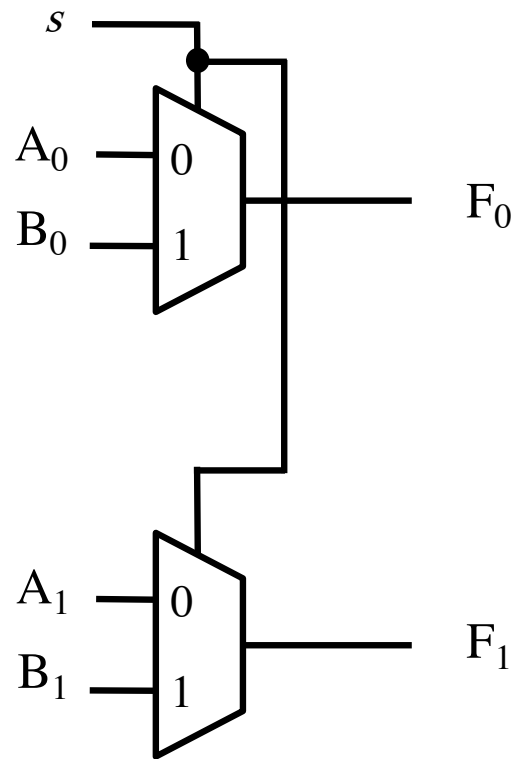
Parallel-access shift left/right register



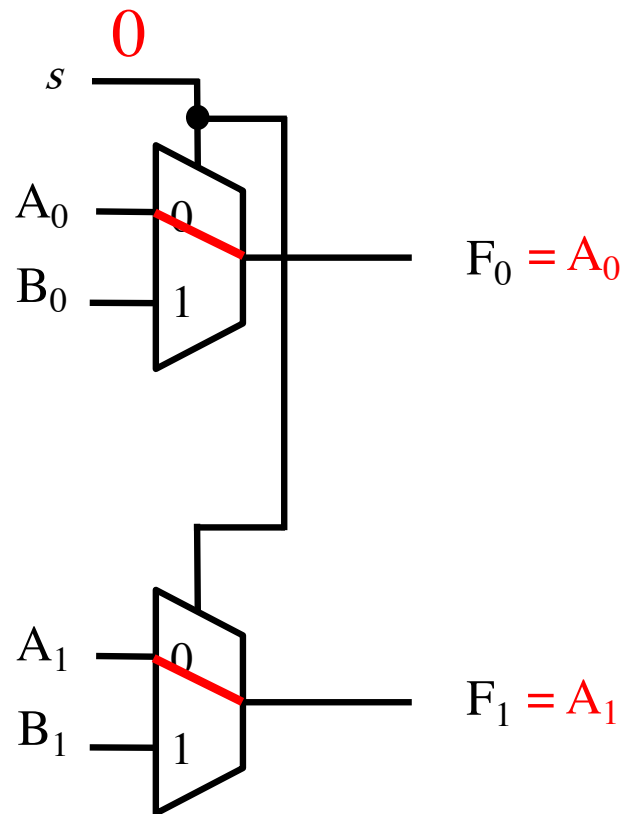
Multiplexer Tricks

(select one of two 2-bit numbers)

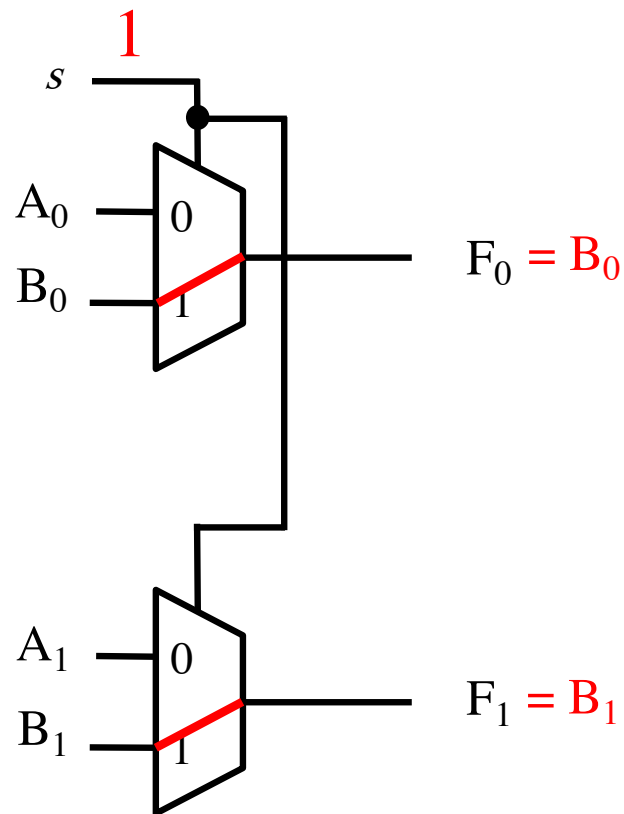
Select Either $A=A_1A_0$ or $B=B_1B_0$



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



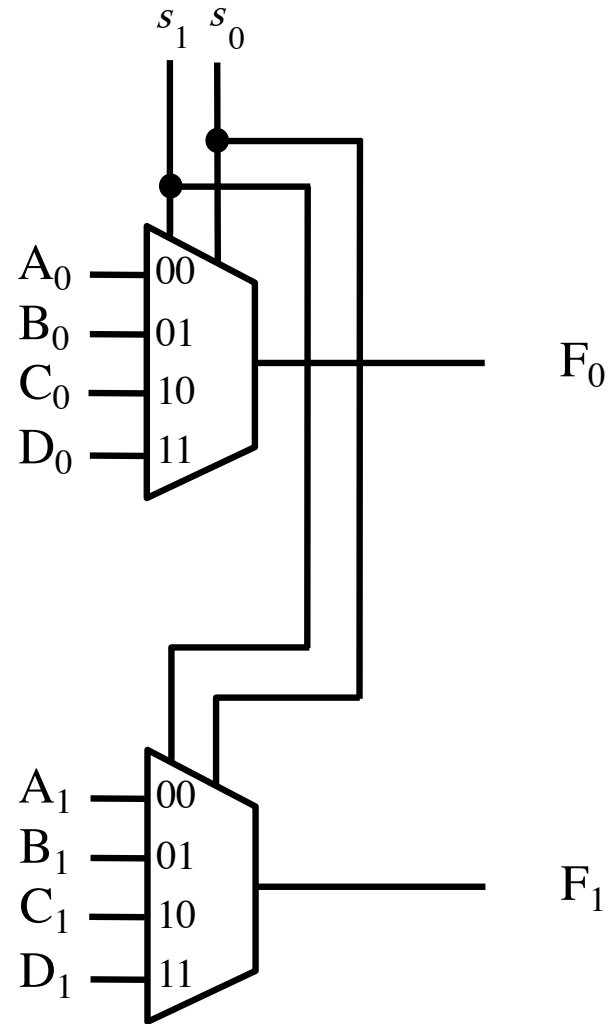
Select Either $A=A_1A_0$ or $B=B_1B_0$



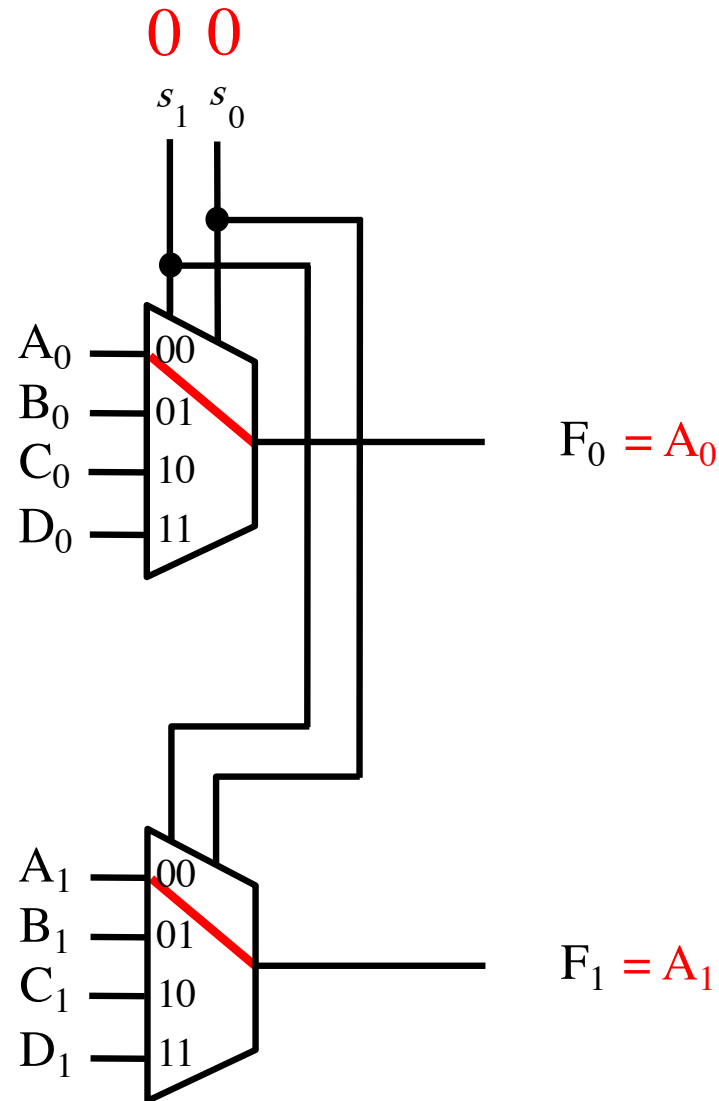
Multiplexer Tricks

(select one of four 2-bit numbers)

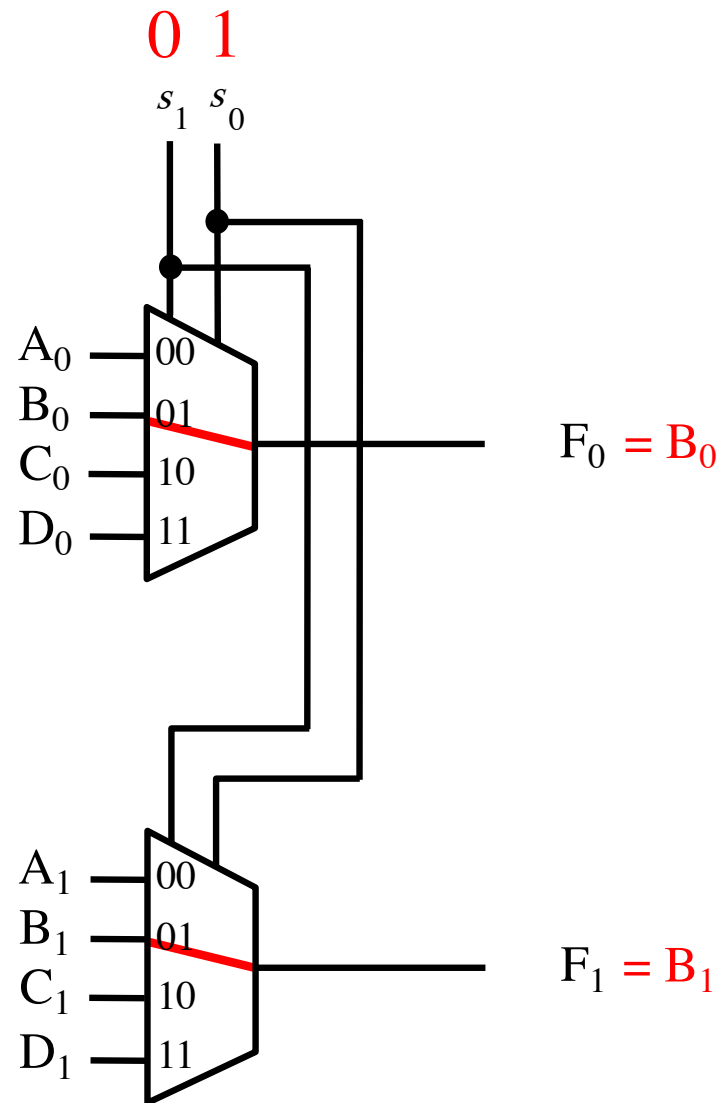
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



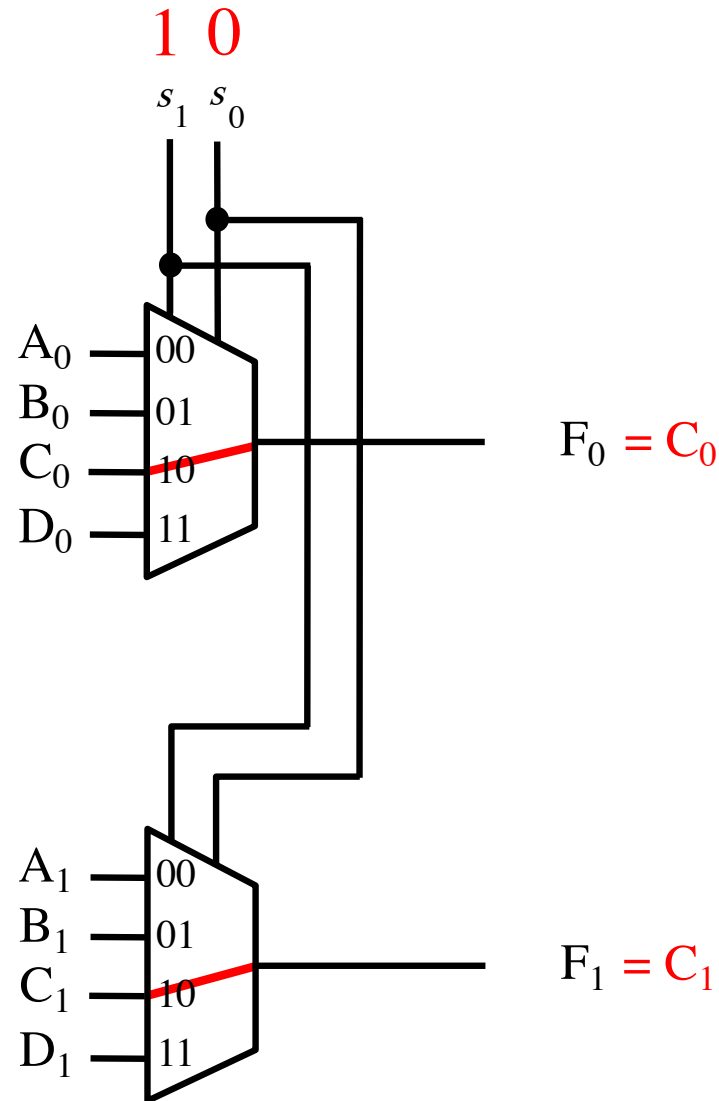
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



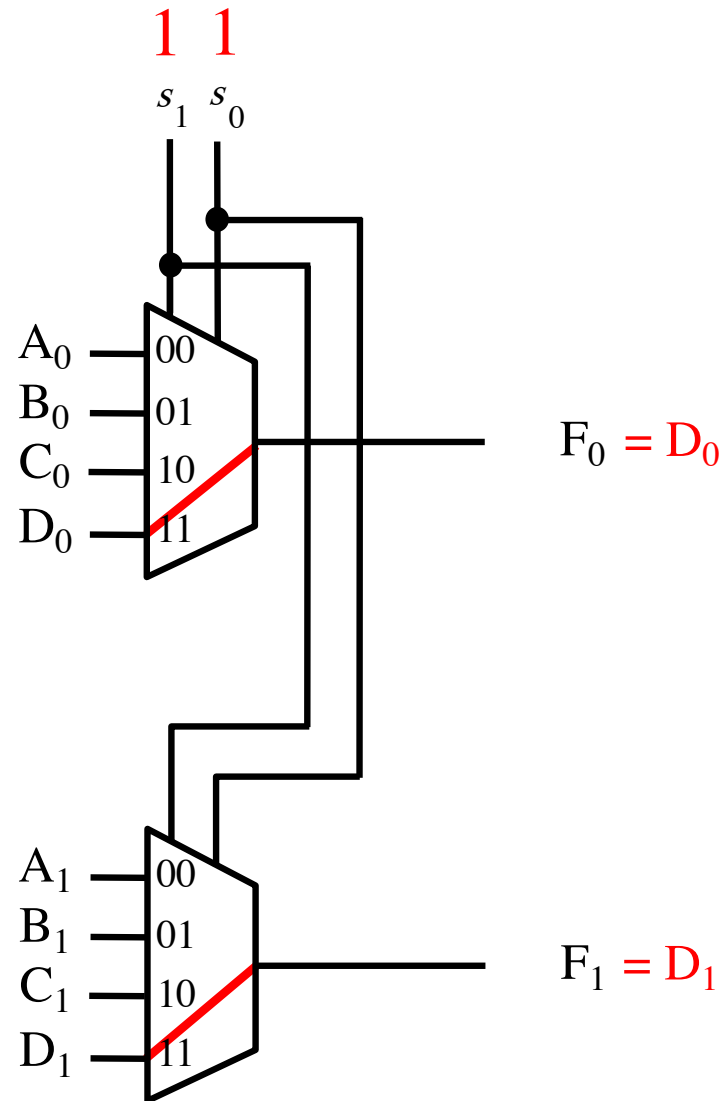
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



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 $D=D_1D_0$

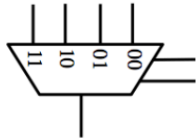
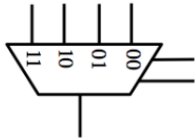
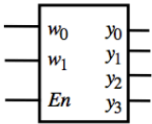
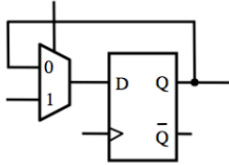
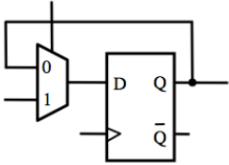
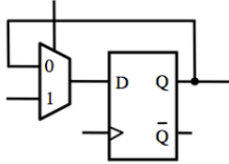
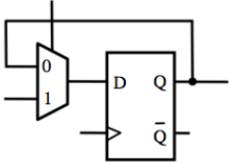
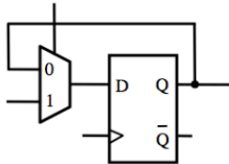
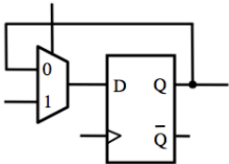
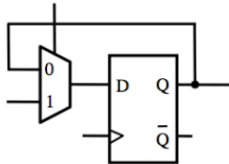
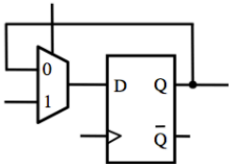


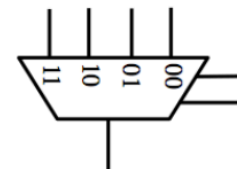
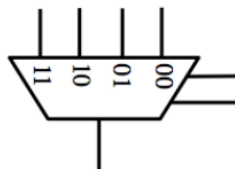
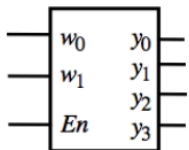
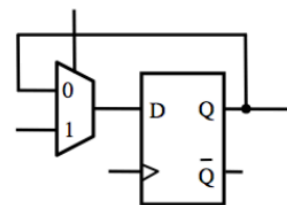
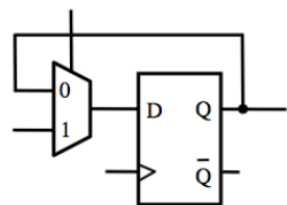
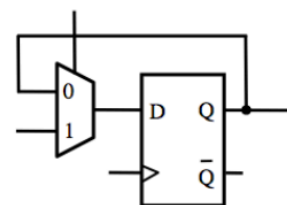
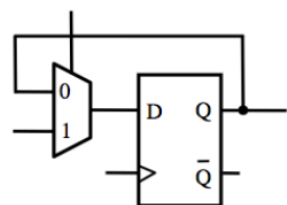
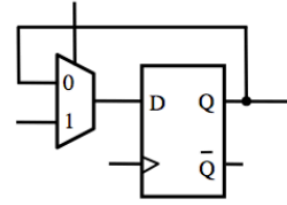
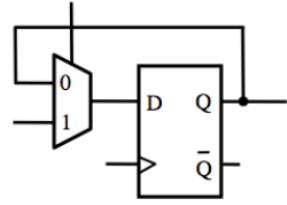
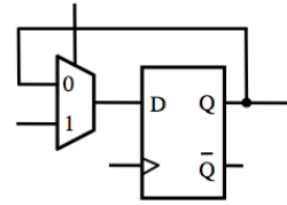
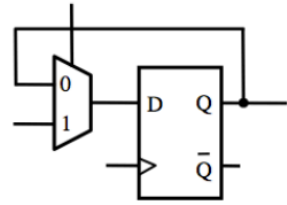
Register File

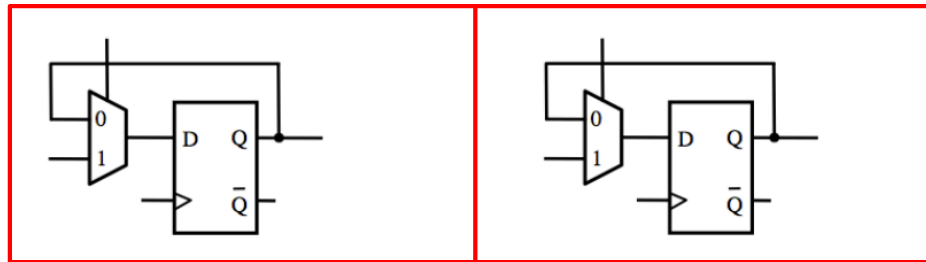
Motivation

- **We would like to build a circuit that can store several numbers which can be read or written independently.**
- **Each number is stored in a separate n-bit register.**
- **To write: a decoder selects which register is enabled for writing. An input bus provides the values.**
- **To read: a set of multiplexers select which register will be read and copied to the output bus.**
- **Some register files come with two read ports. In those designs the multiplexer circuitry is doubled.**

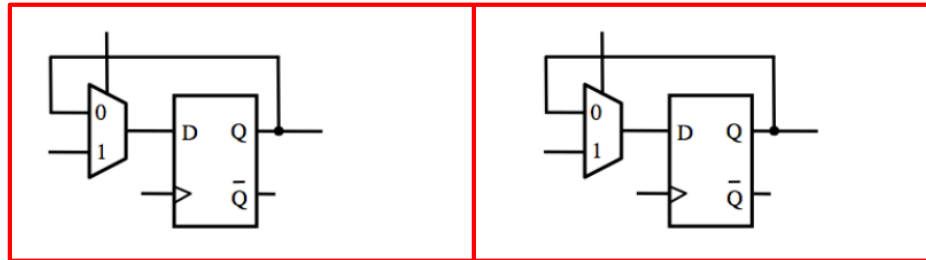
Complete the following circuit diagram to implement a register file with four 2-bit registers, one write port, one read port, and one write enable line.



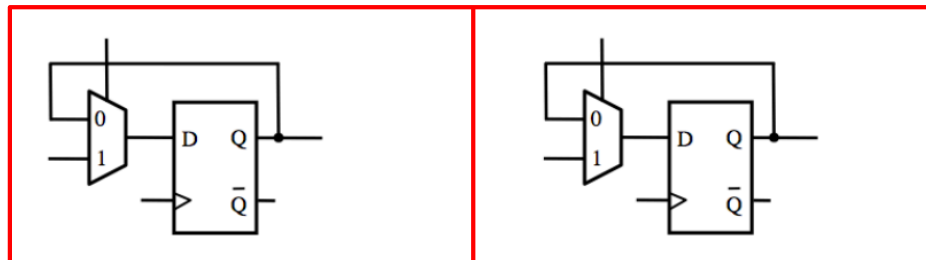




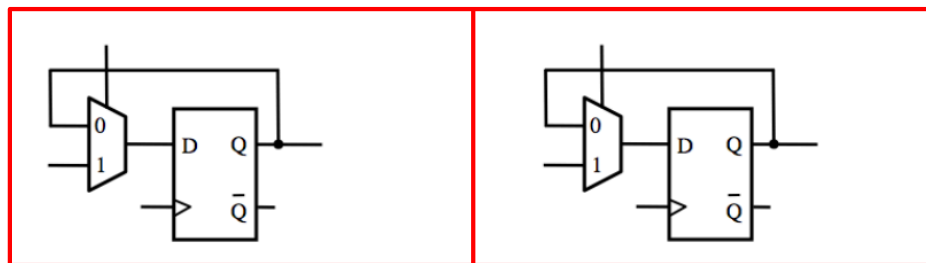
Register 0



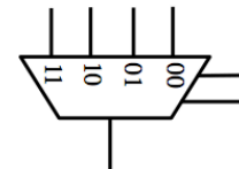
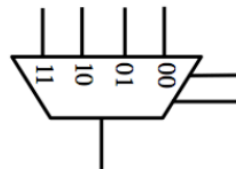
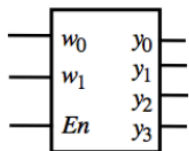
Register 1

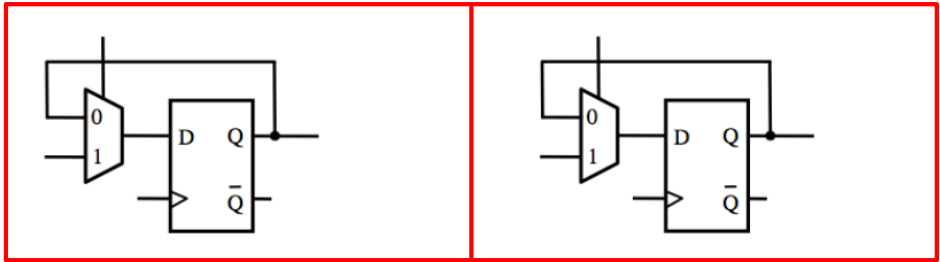


Register 2

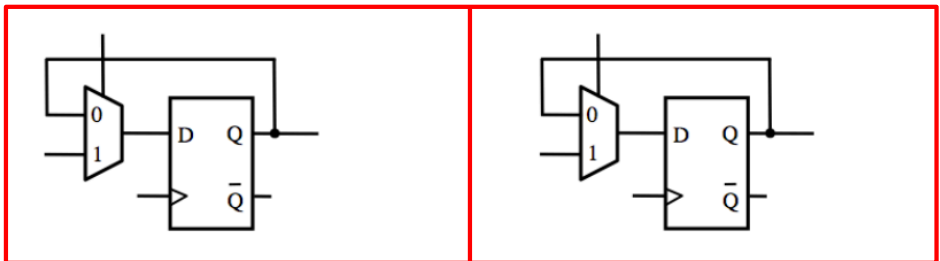


Register 3

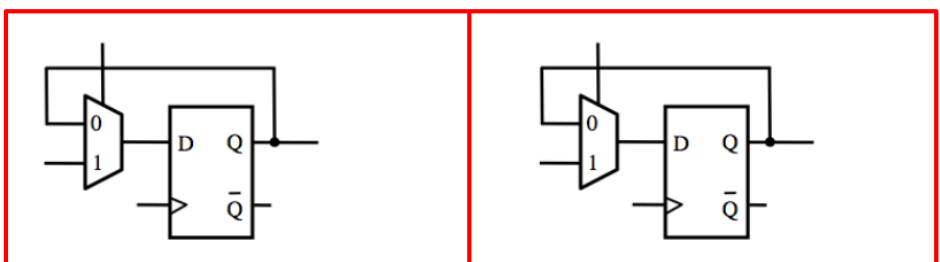




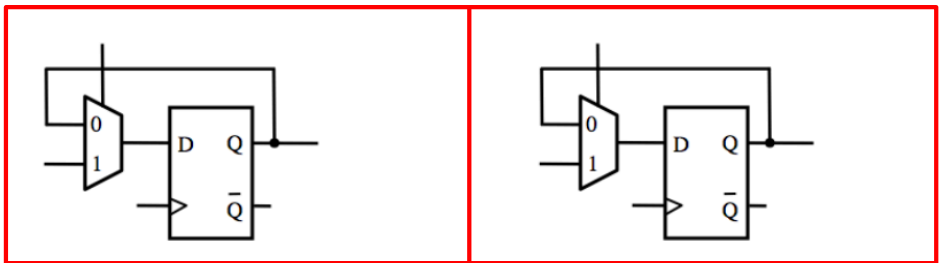
Register A



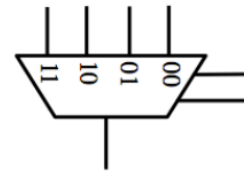
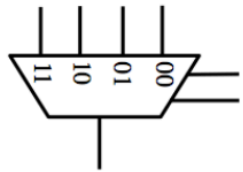
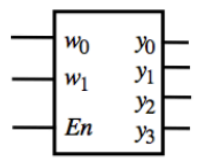
Register B

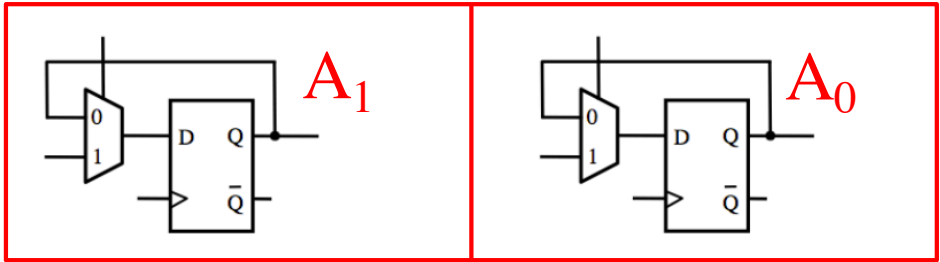


Register C

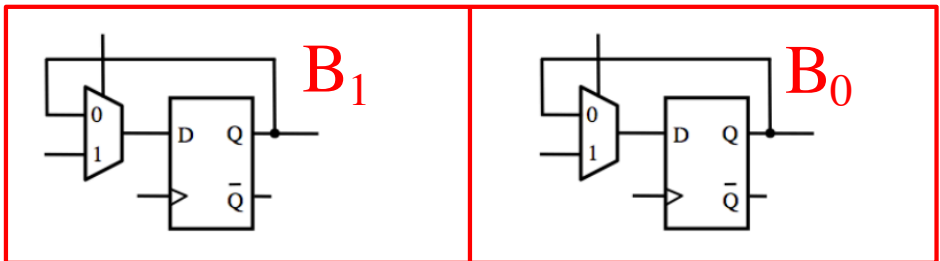


Register D

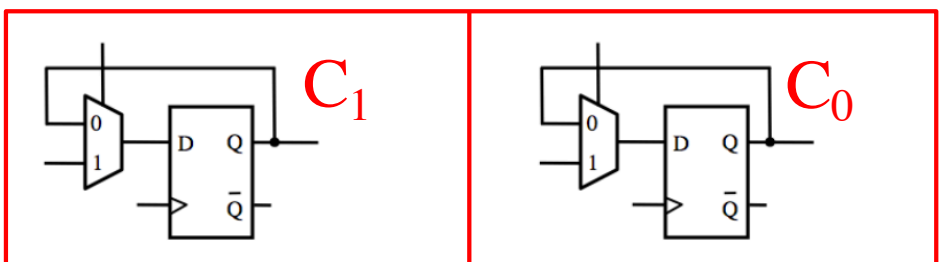




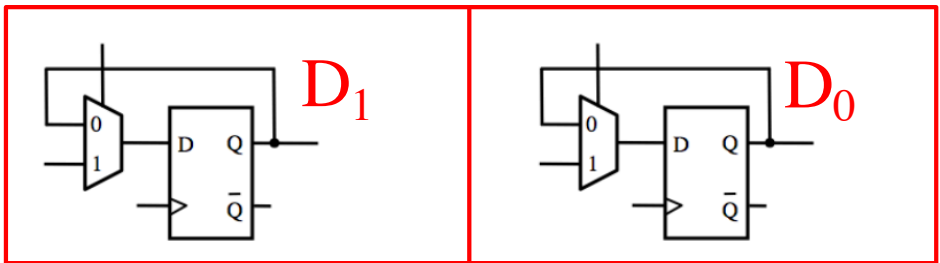
Register A



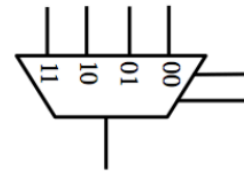
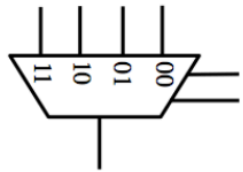
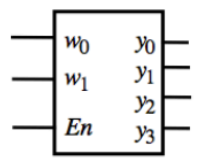
Register B

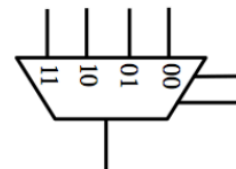
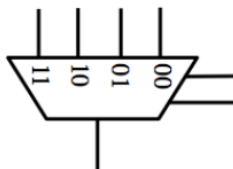
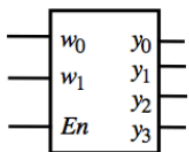
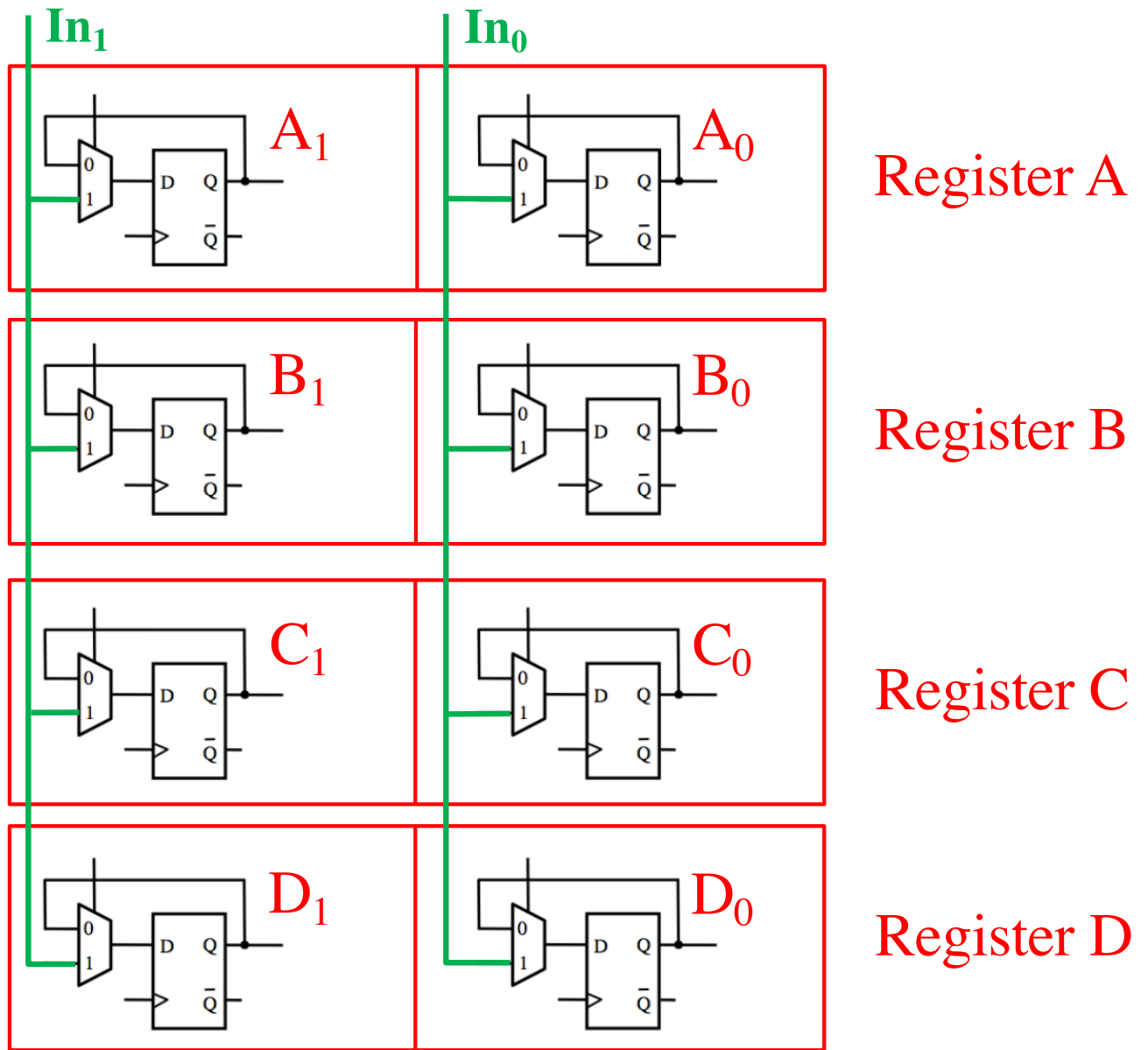


Register C



Register D





In_1

In_0

A_1

A_0

Register A

B_1

B_0

Register B

C_1

C_0

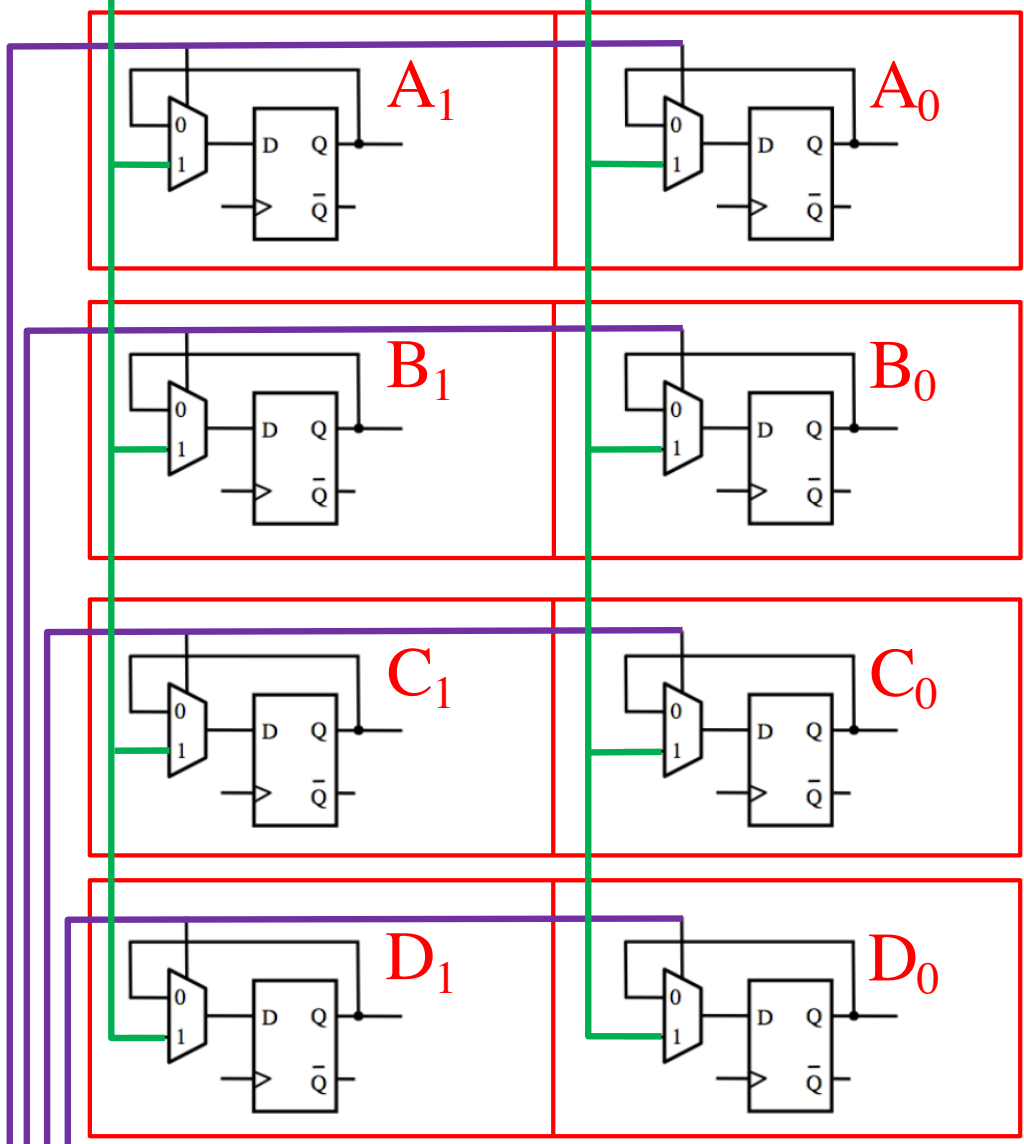
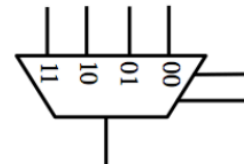
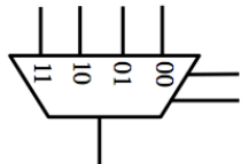
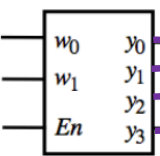
Register C

D_1

D_0

Register D

Write_address_0
Write_address_1
Write_enable



In₁

In₀

A₁

A₀

Register A

B₁

B₀

Register B

C₁

C₀

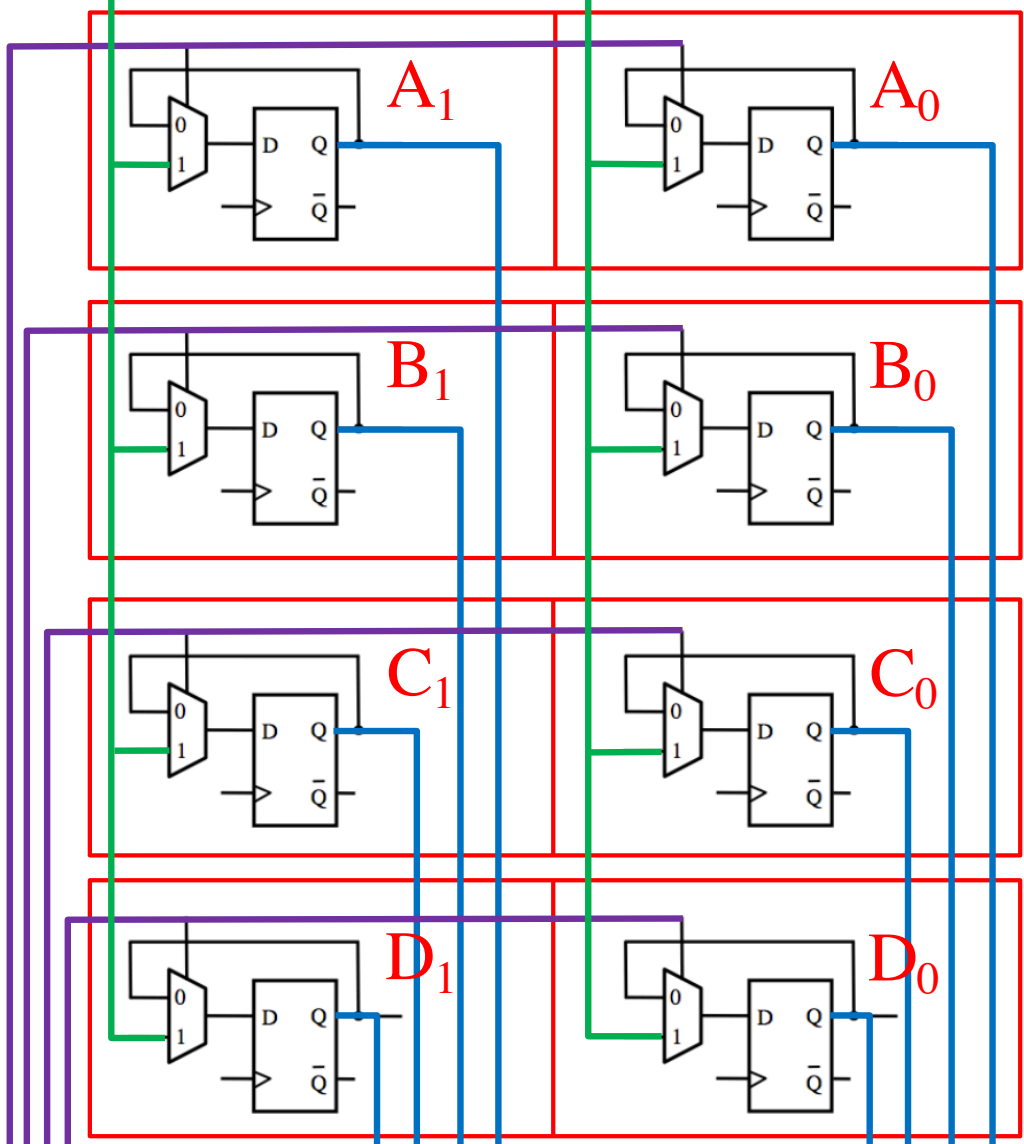
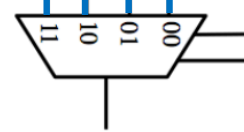
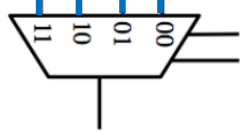
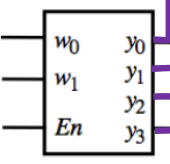
Register C

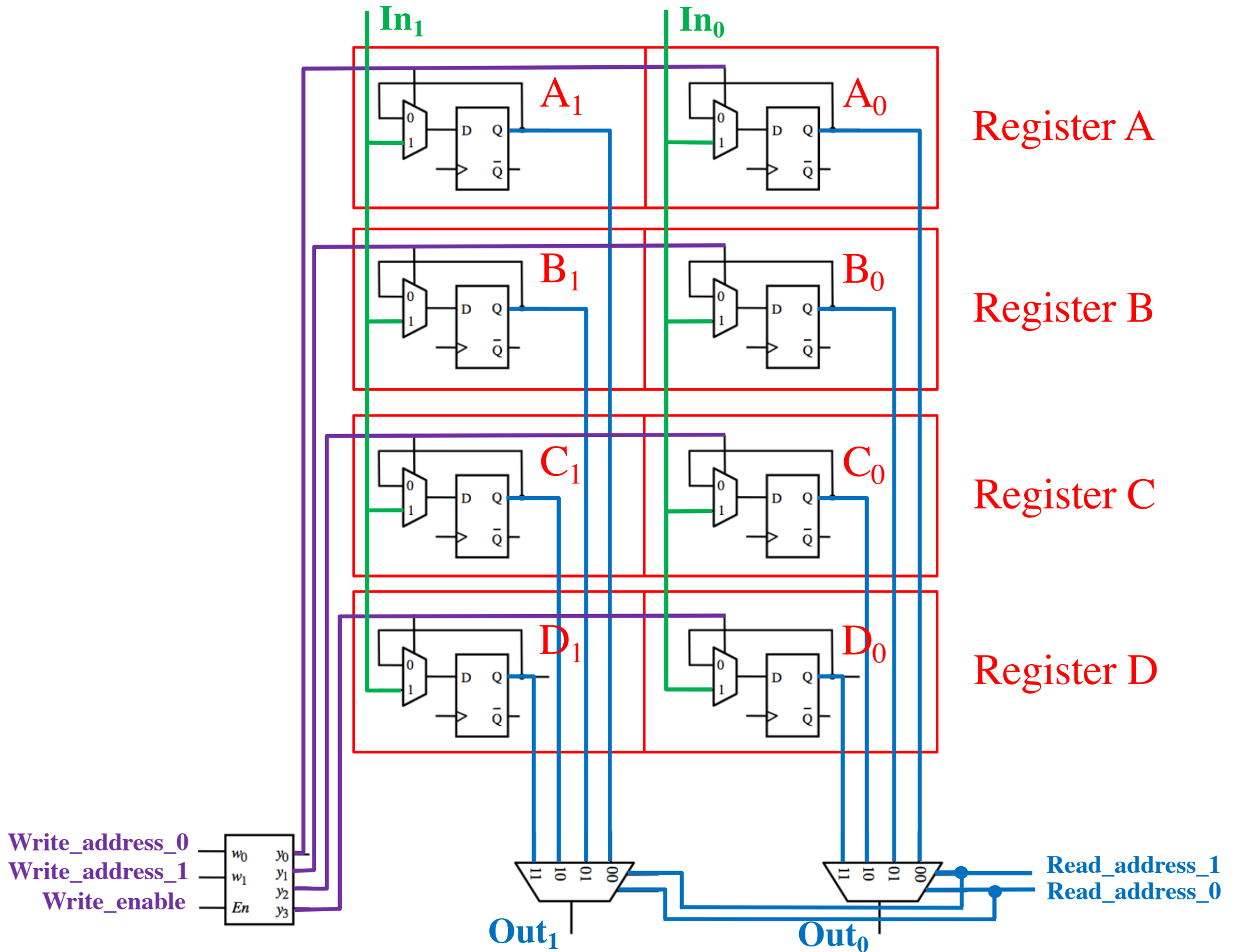
D₁

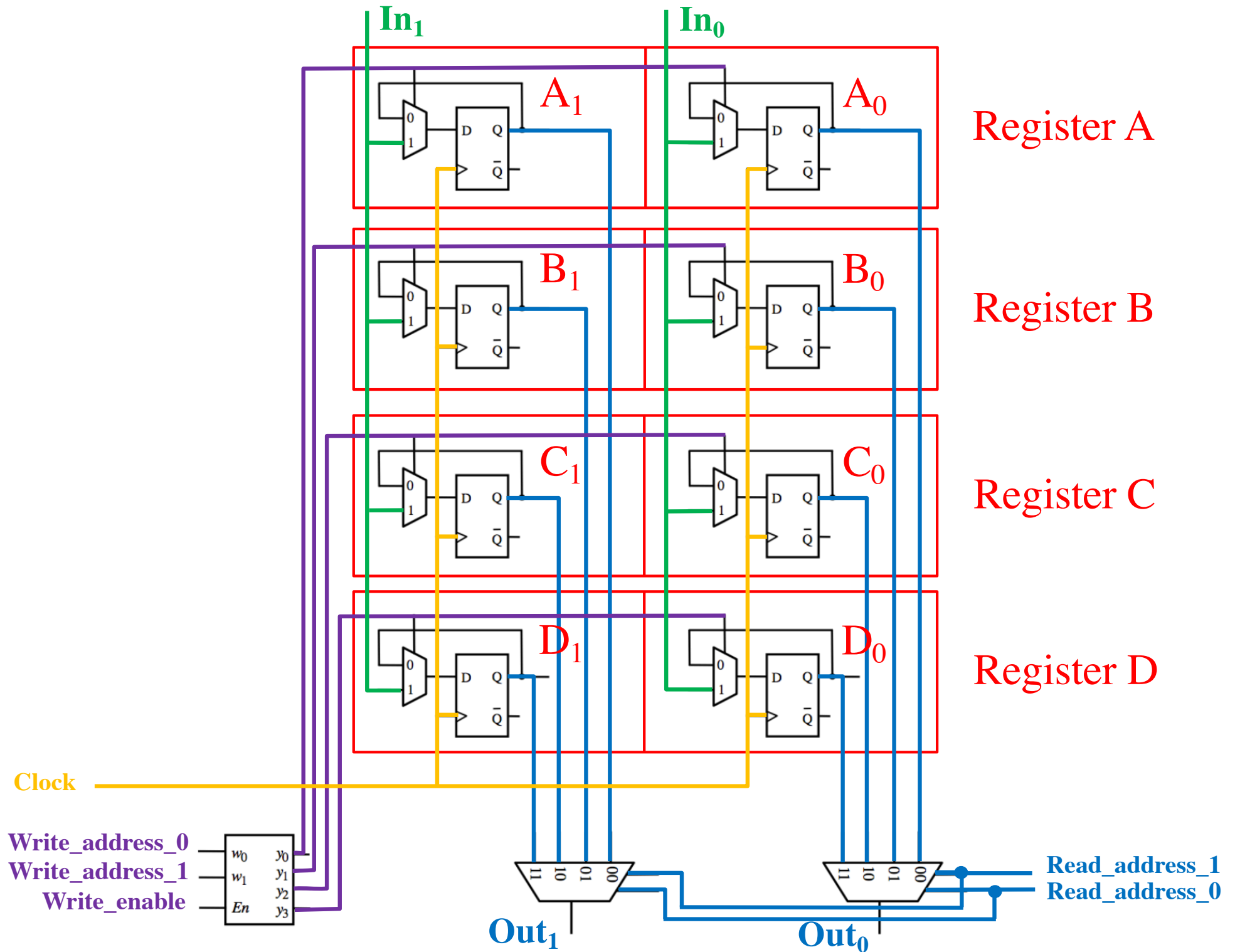
D₀

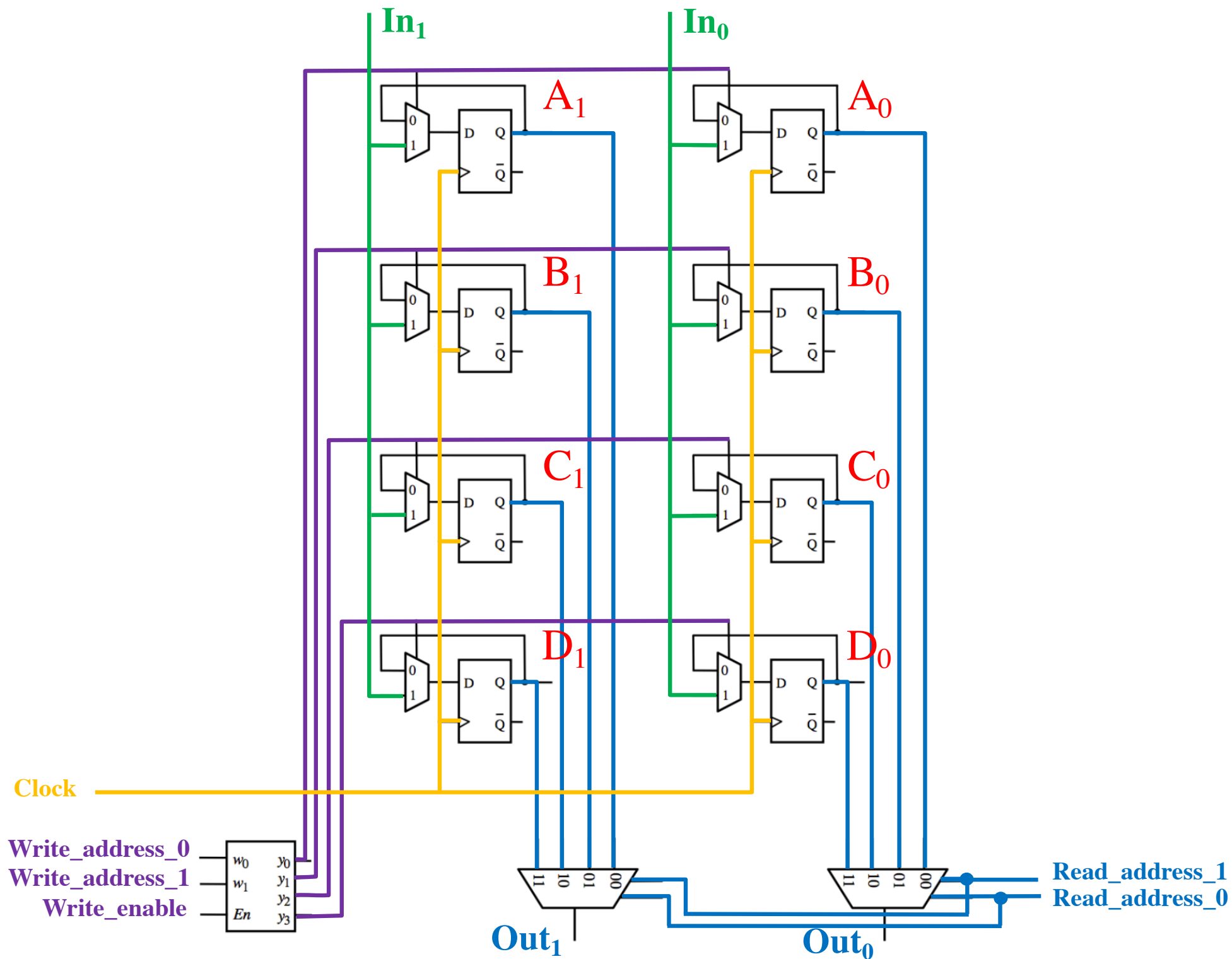
Register D

Write_address_0
Write_address_1
Write_enable



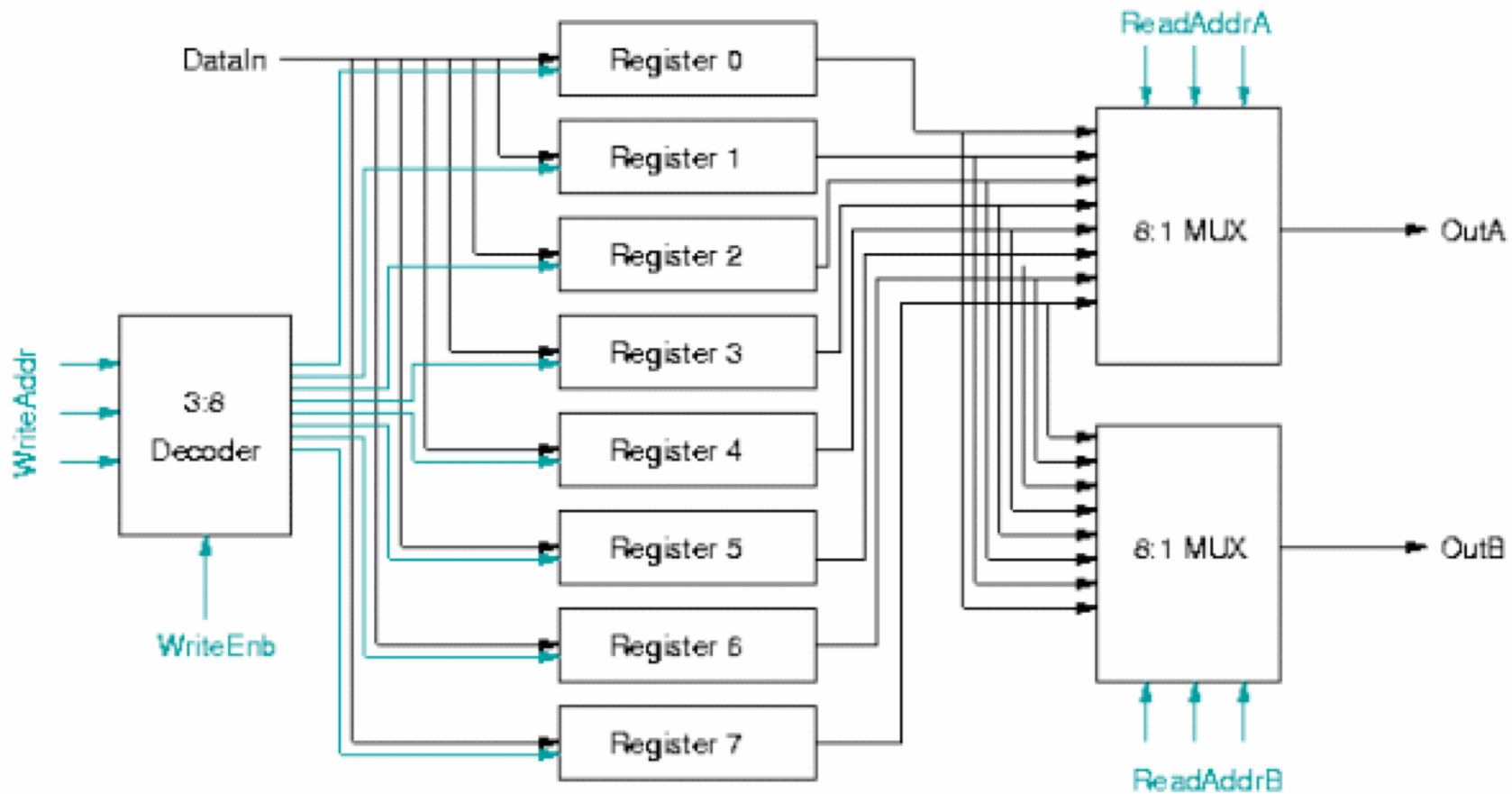






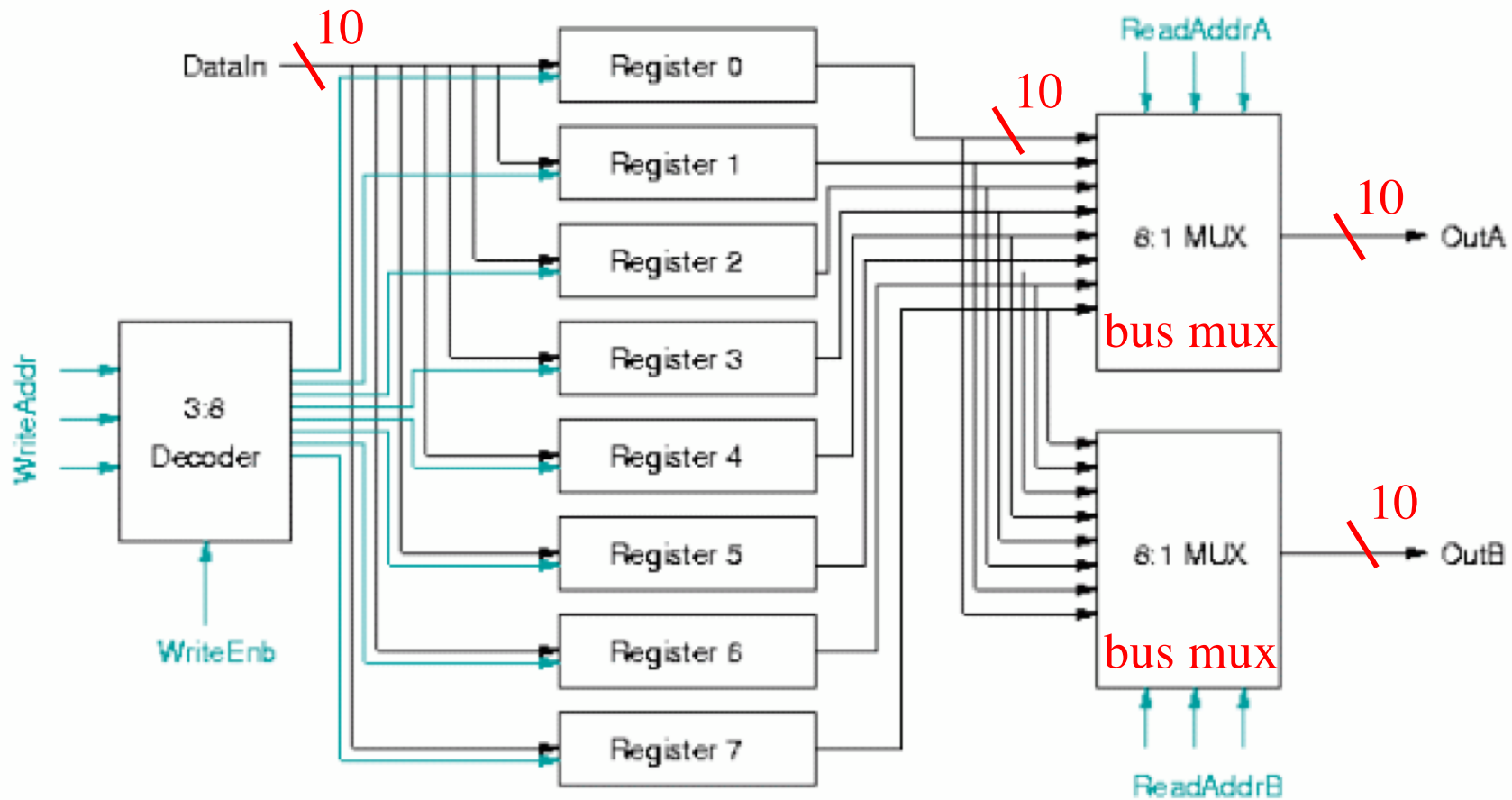
Register file with eight 10-bit registers and two read ports

Register file with 2 read ports



Gray lines are 1-bit signals
Black lines are 10-bit signals

Register file with 2 read ports



Gray lines are 1-bit signals
Black lines are 10-bit signals

Questions?

THE END