



CprE 281: Digital Logic

Instructor: Alexander Stoytchev

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

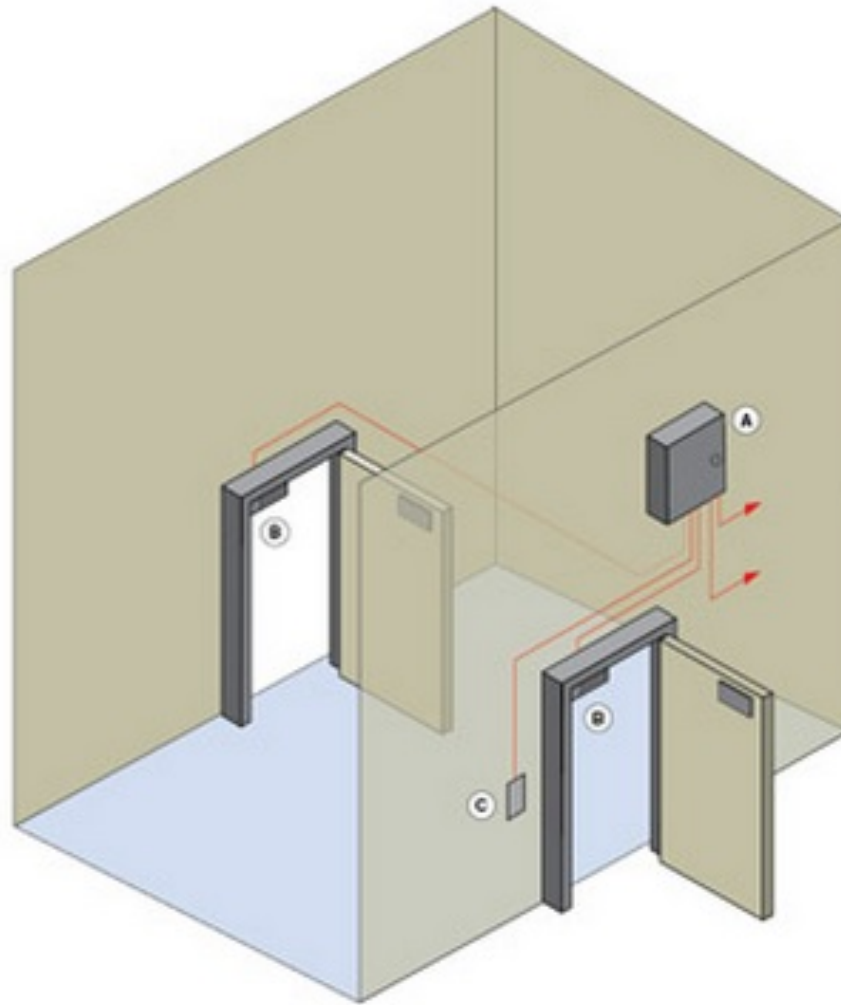
Registers and Counters

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

Flip-Flop Analogy

(double door)

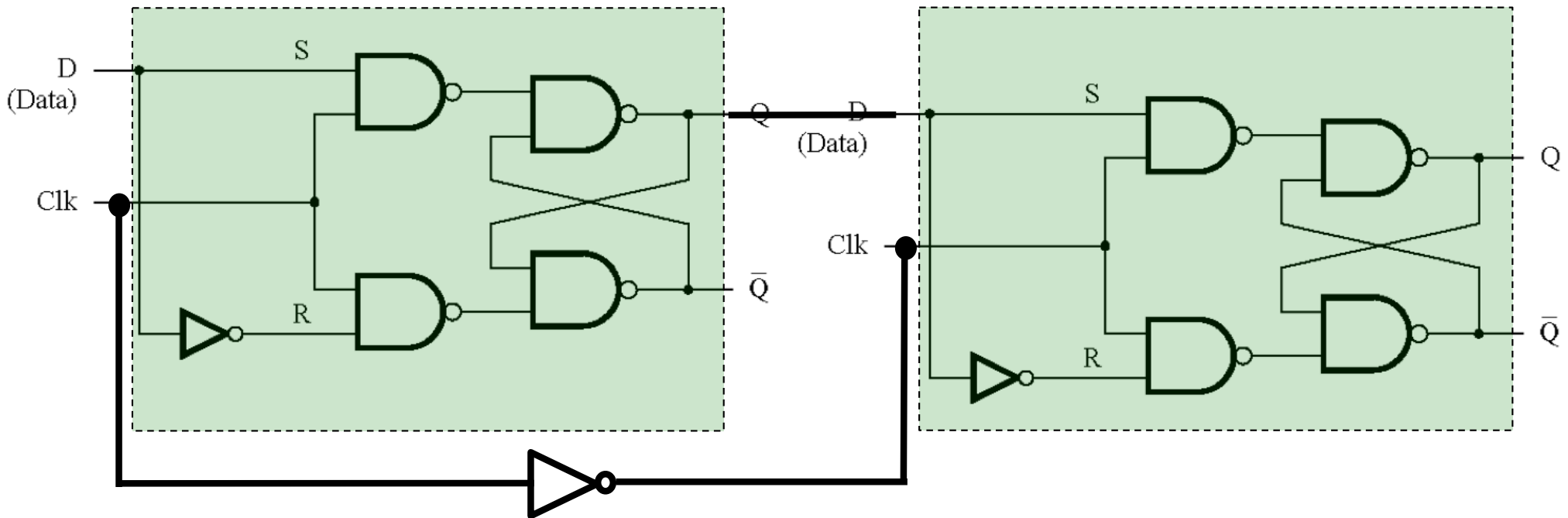
D Flip-Flop Analogy



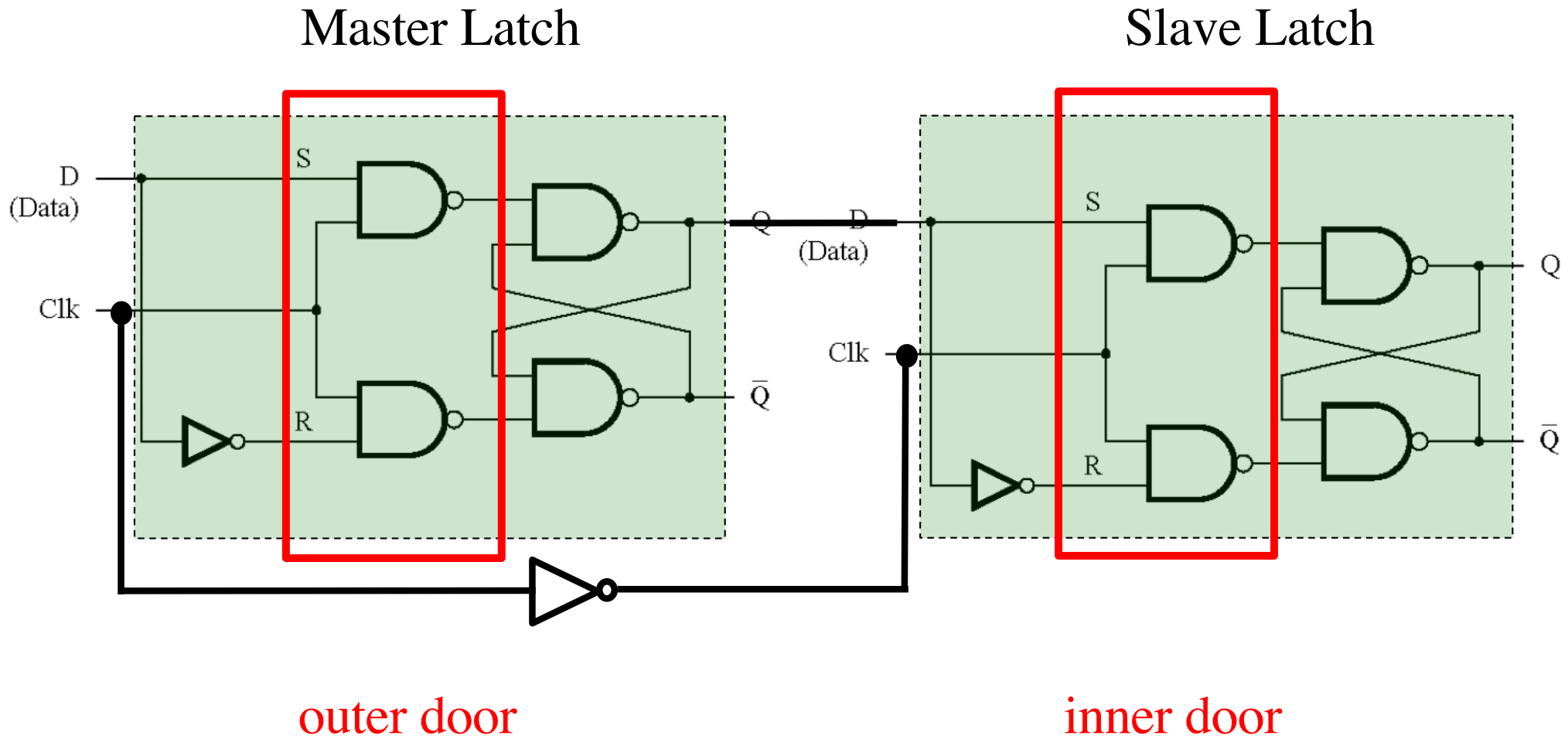
D Flip-Flop Analogy

Master Latch

Slave Latch



D Flip-Flop Analogy

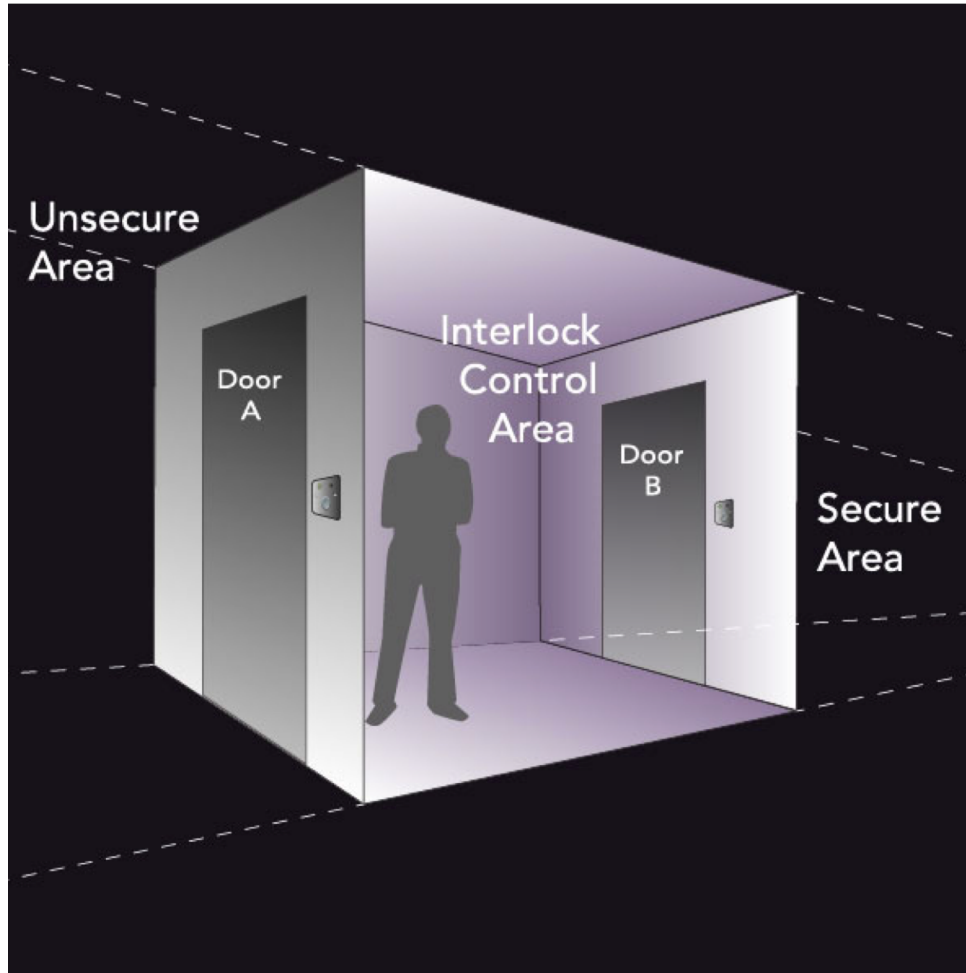


D Flip-Flop Analogy

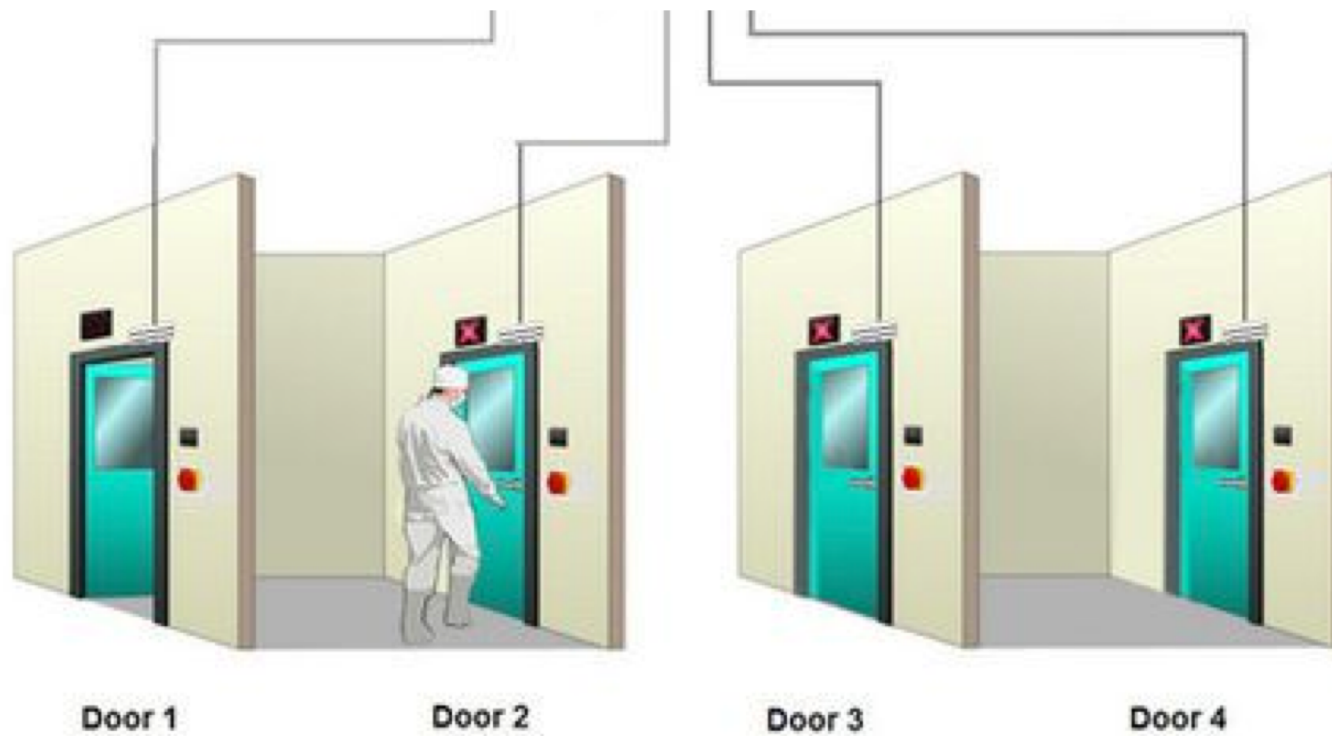


Outer Door
Will Not Unlock When
Inner Door is Open

Inner Door
Will Not Unlock When
Outer Door is Open



Shift-Register Analogy



Registers

Register (Definition)

An n-bit structure consisting of flip-flops

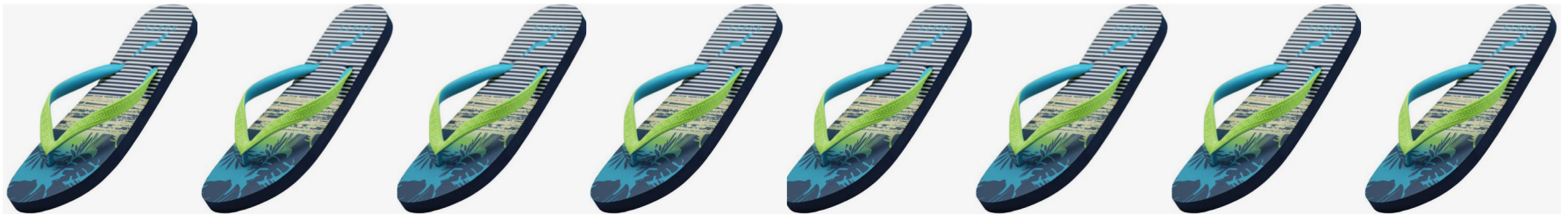
Flip-Flop



4-bit Register

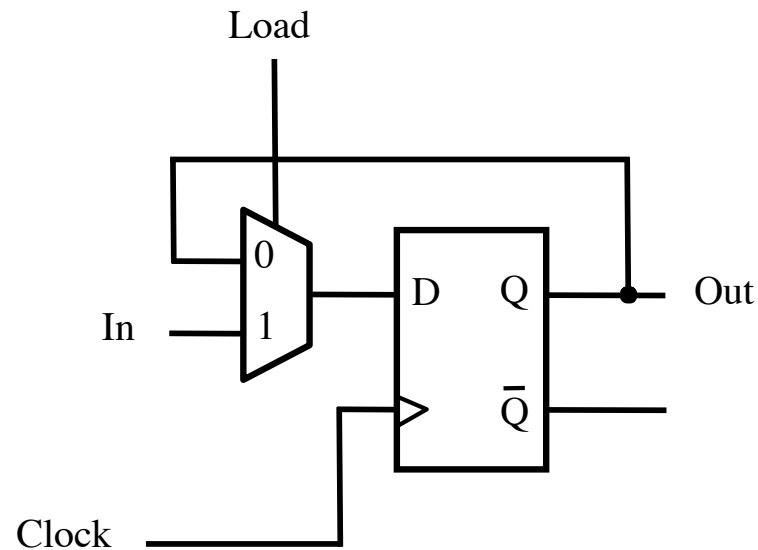


8-bit Register



Parallel-Access Register

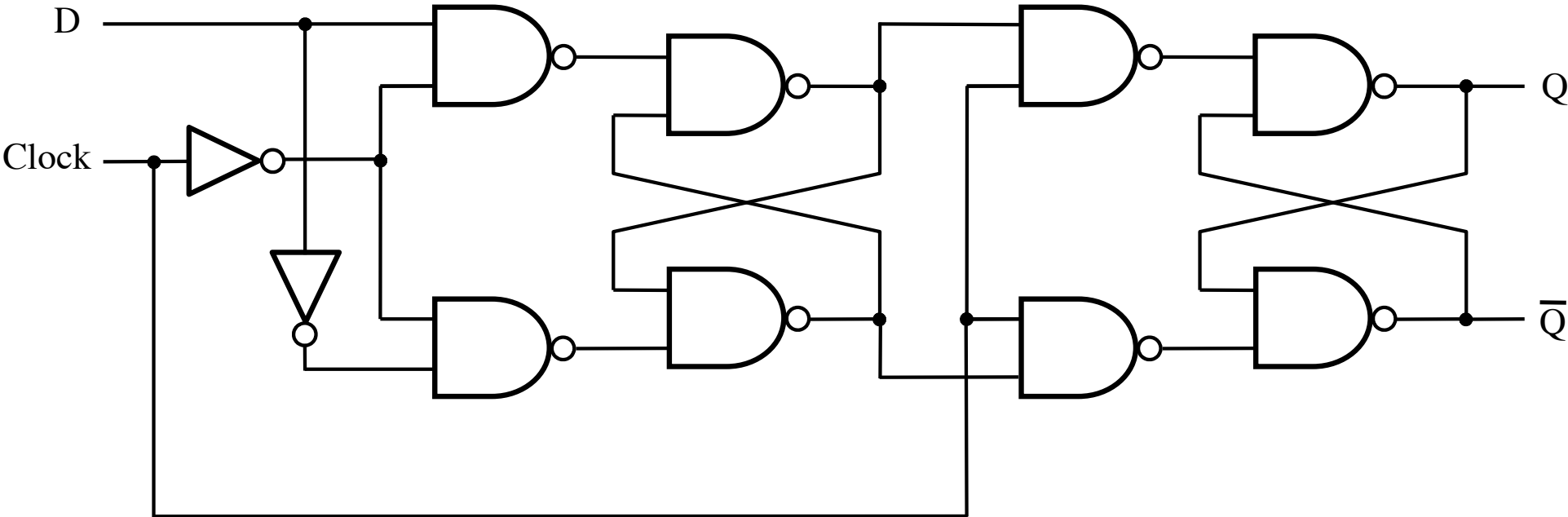
1-Bit Parallel-Access Register



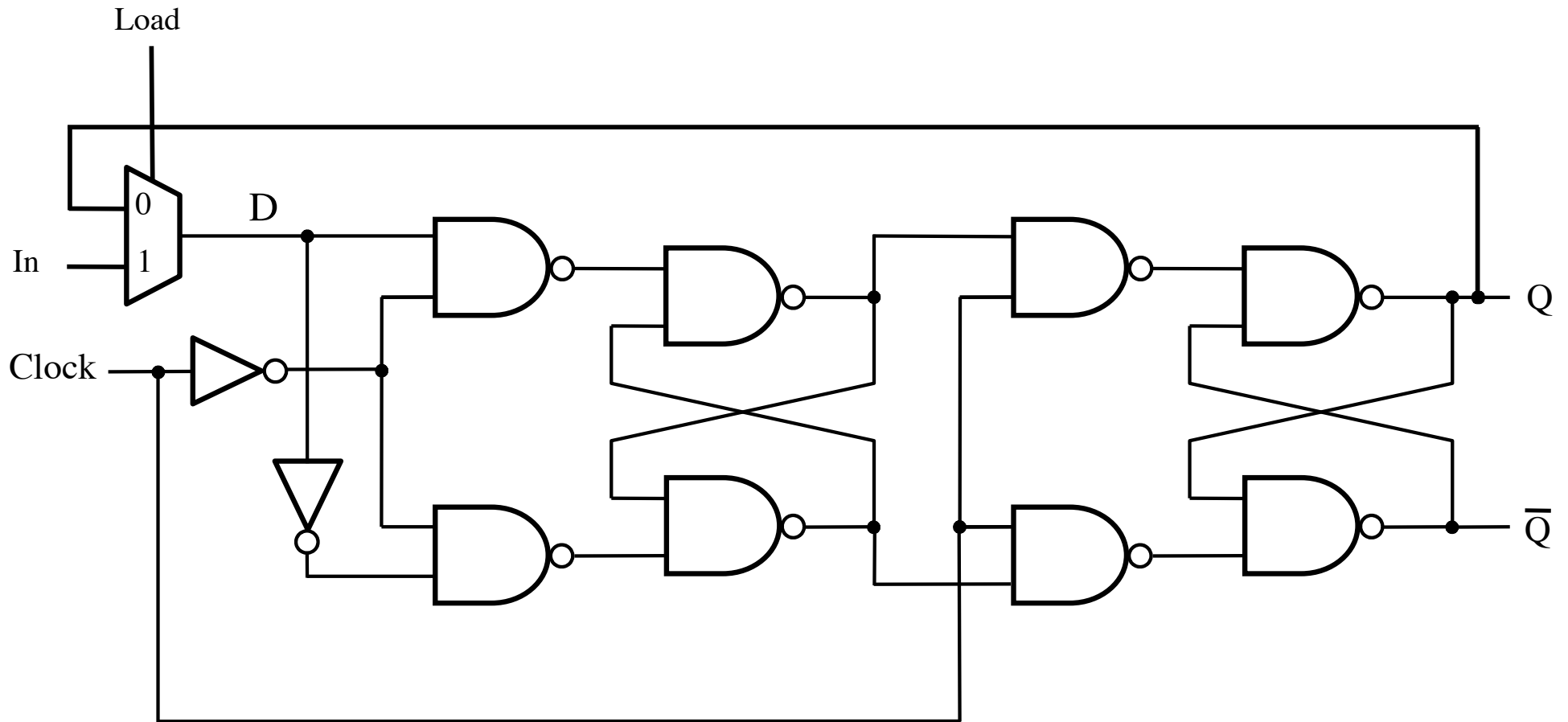
The 2-to-1 multiplexer is used to select whether to load a new value into the D flip-flop or to retain the old value.

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

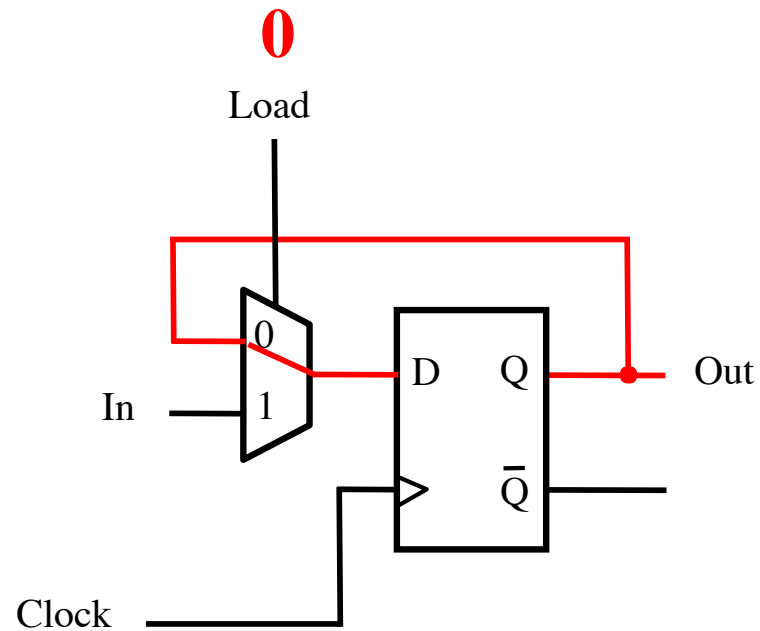
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register

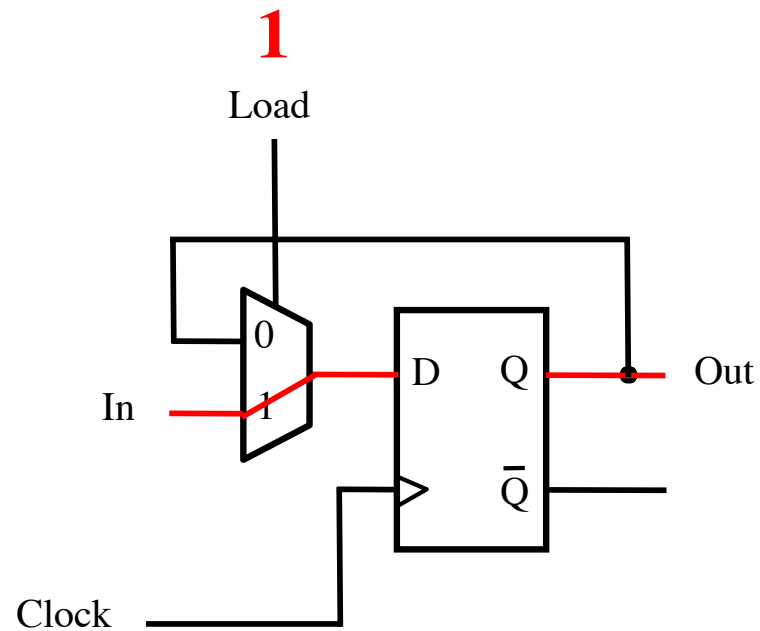


1-Bit Parallel-Access Register



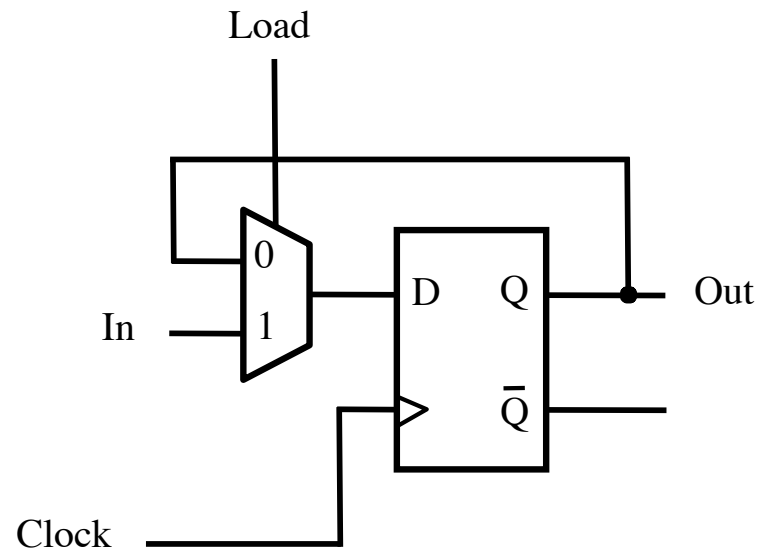
If Load = 0, then retain the old value.

1-Bit Parallel-Access Register



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

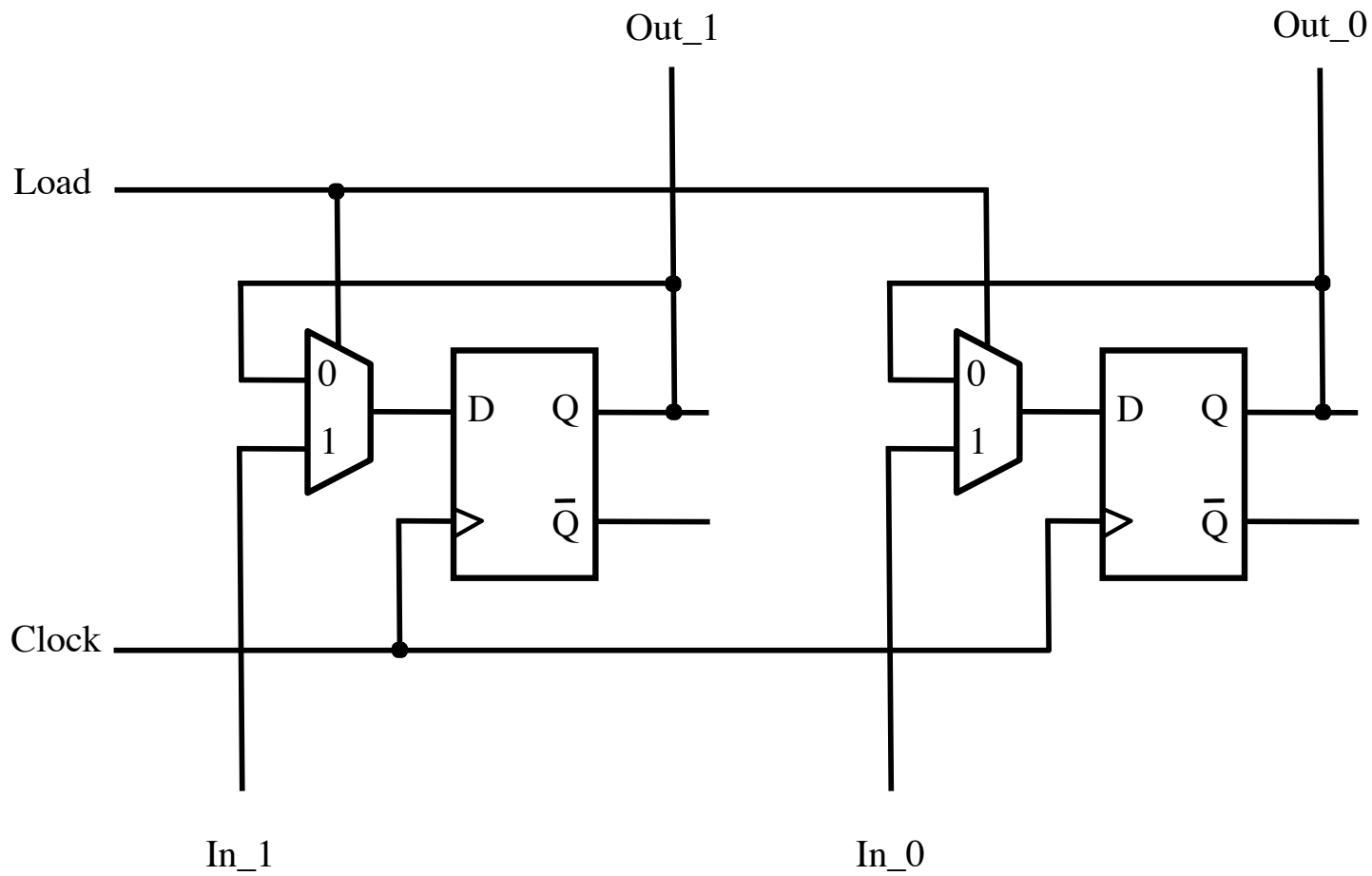
1-Bit Parallel-Access Register



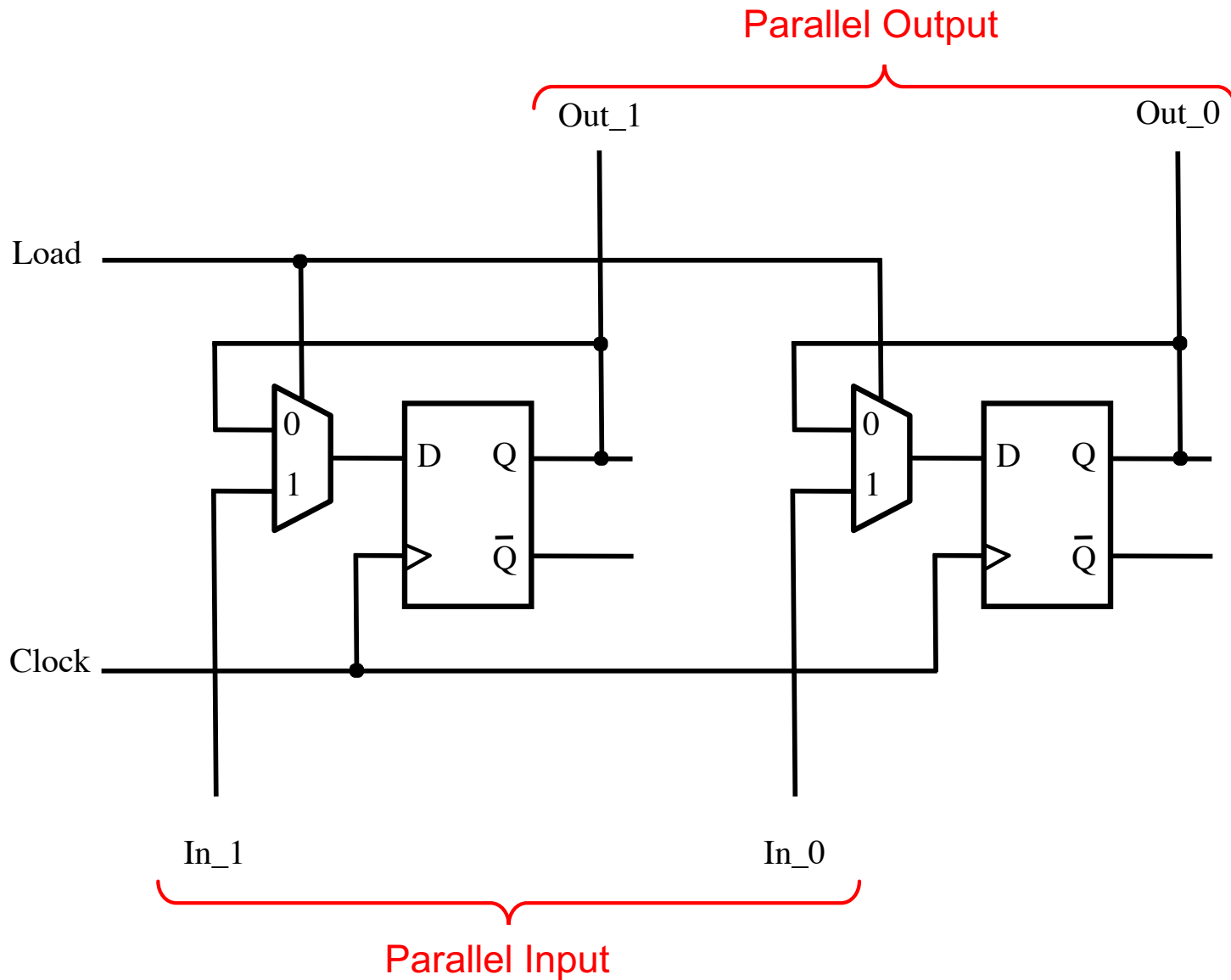
If Load = 0, then retain the old value.

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

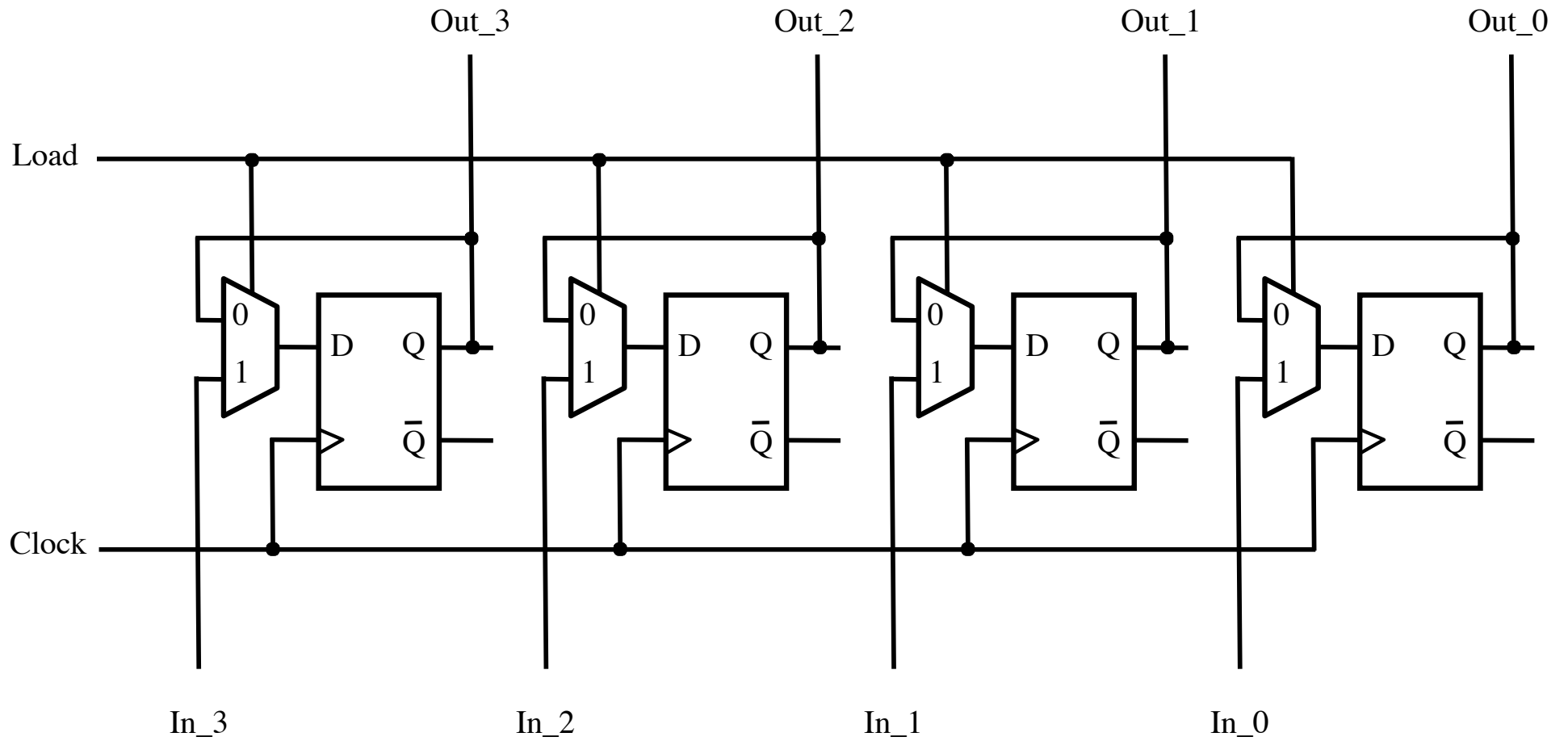
2-Bit Parallel-Access Register



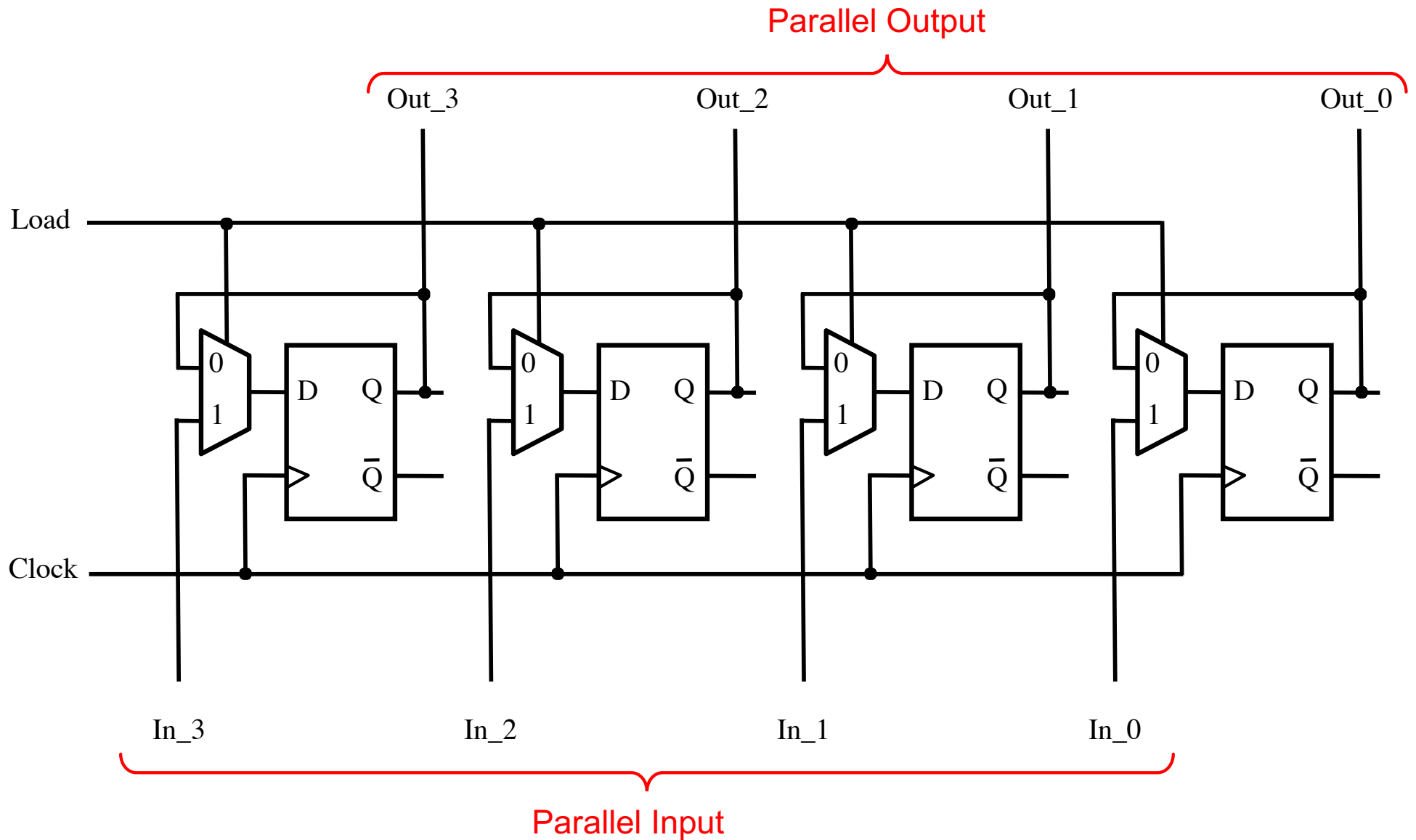
2-Bit Parallel-Access Register



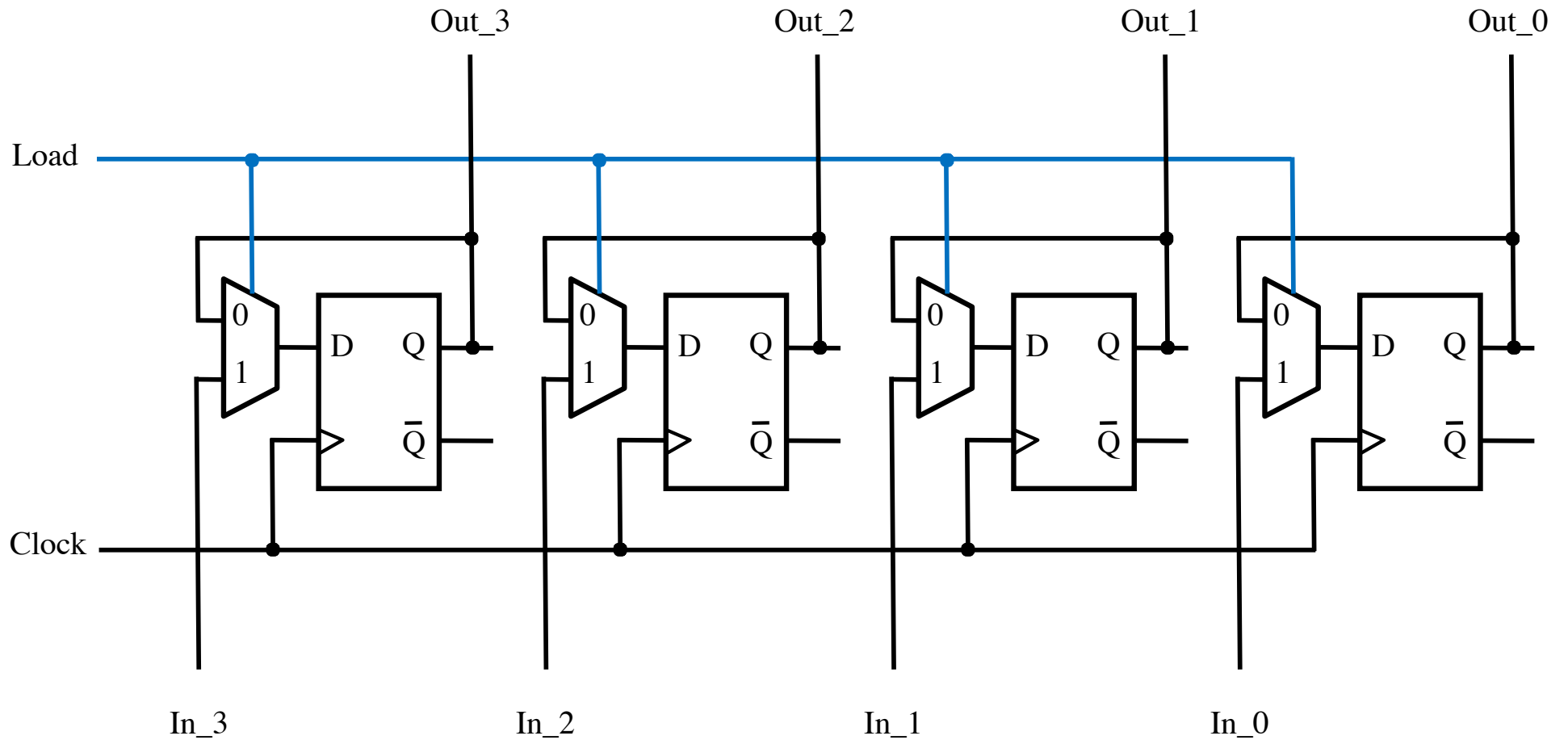
4-Bit Parallel-Access Register



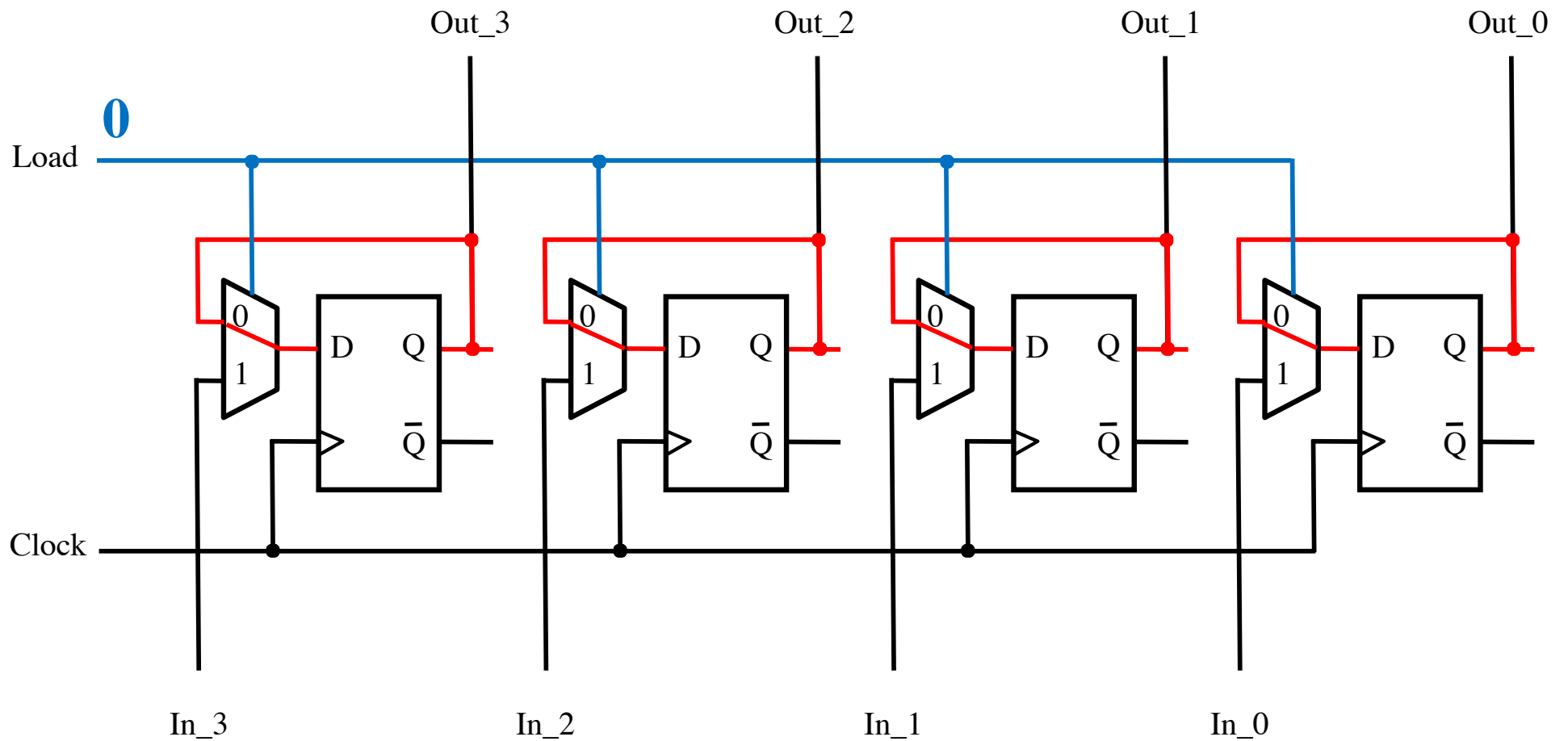
4-Bit Parallel-Access Register



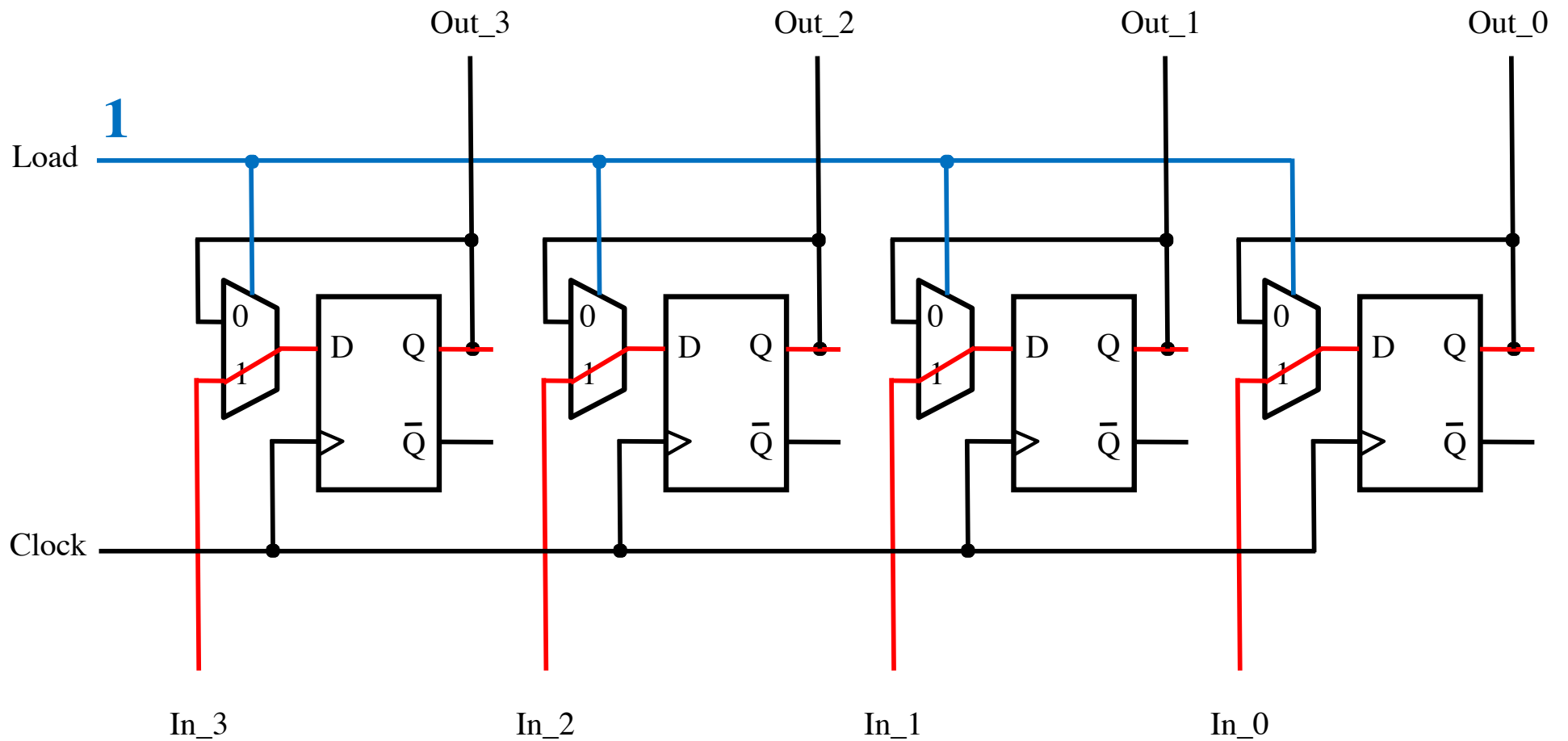
4-Bit Parallel-Access Register



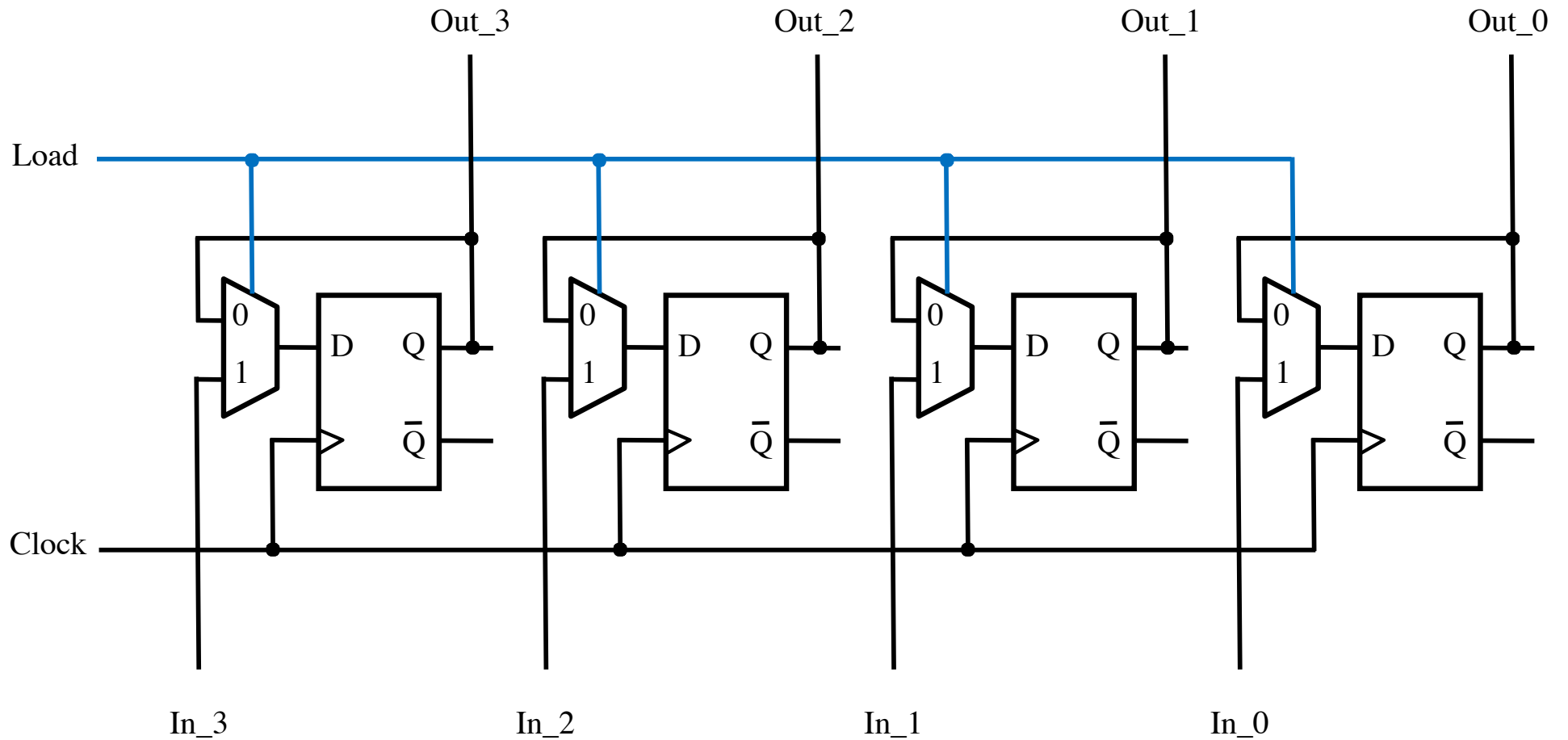
4-Bit Parallel-Access Register



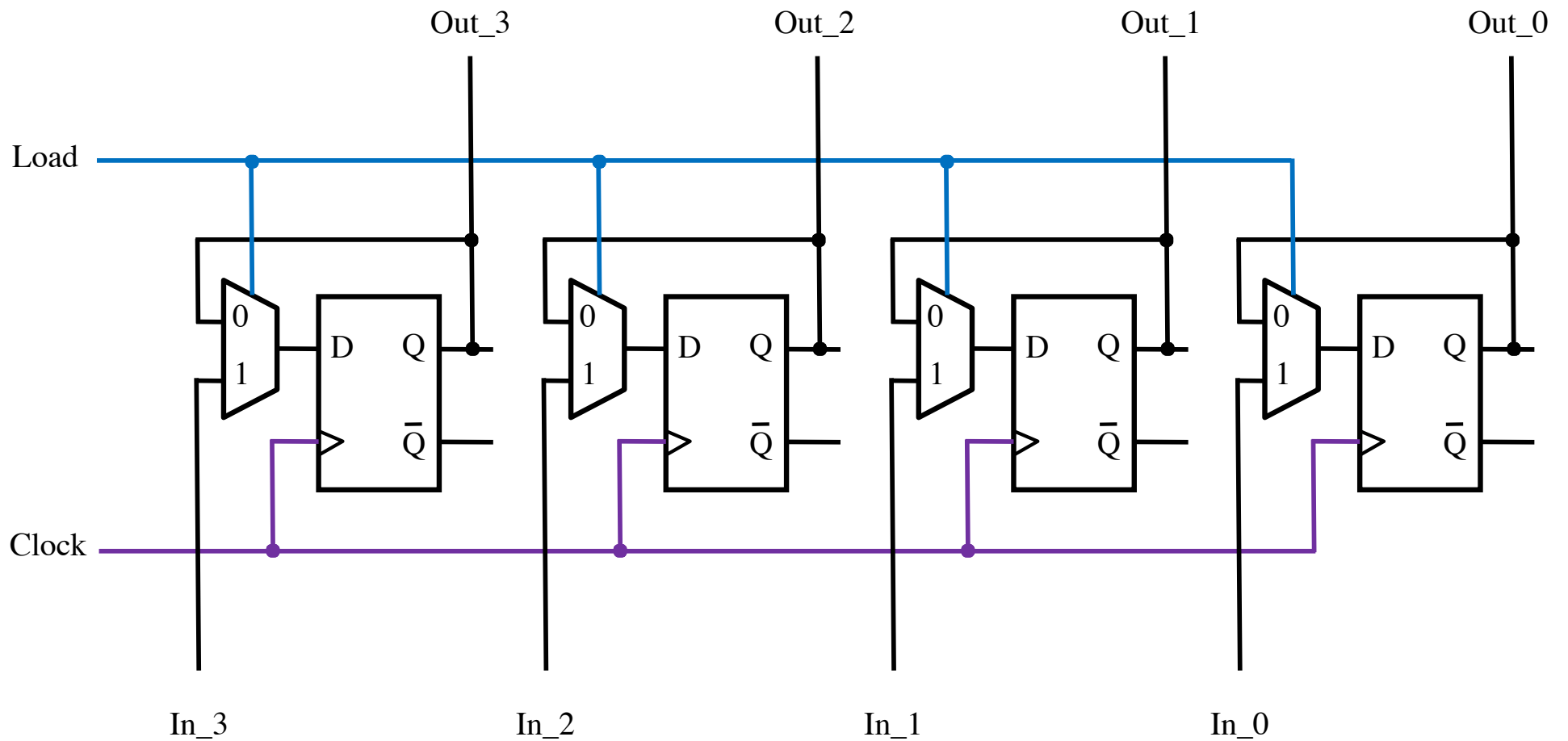
4-Bit Parallel-Access Register



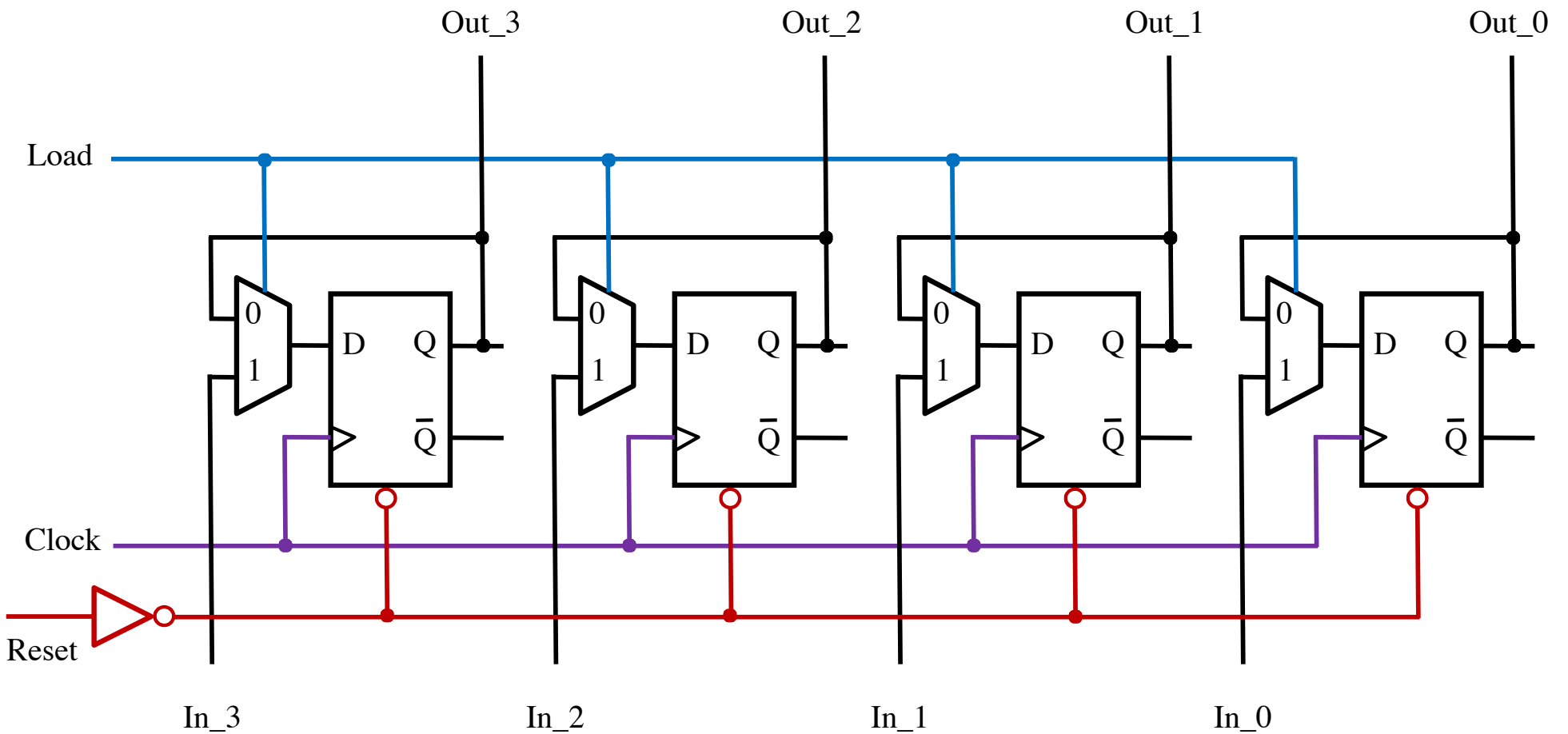
4-Bit Parallel-Access Register



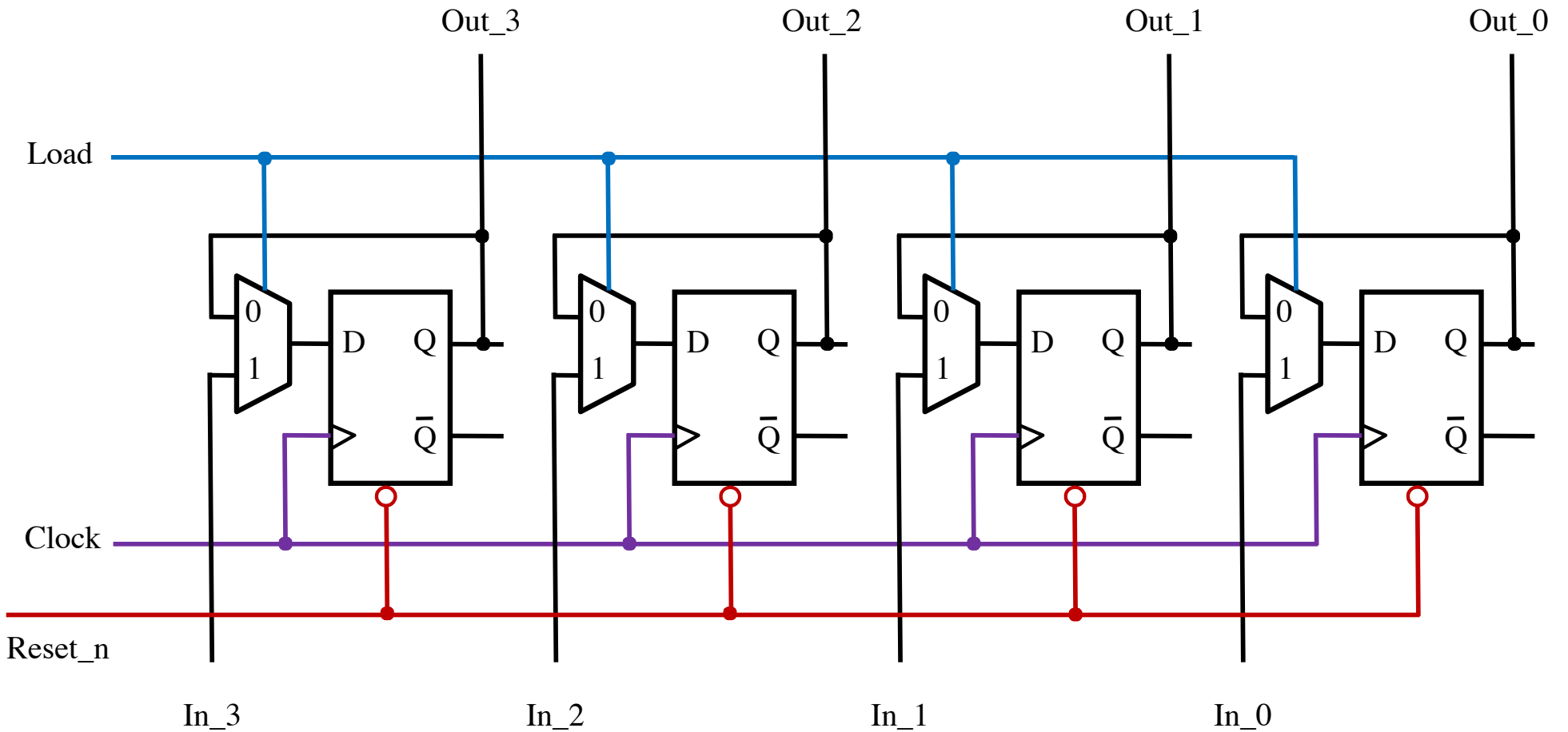
4-Bit Parallel-Access Register



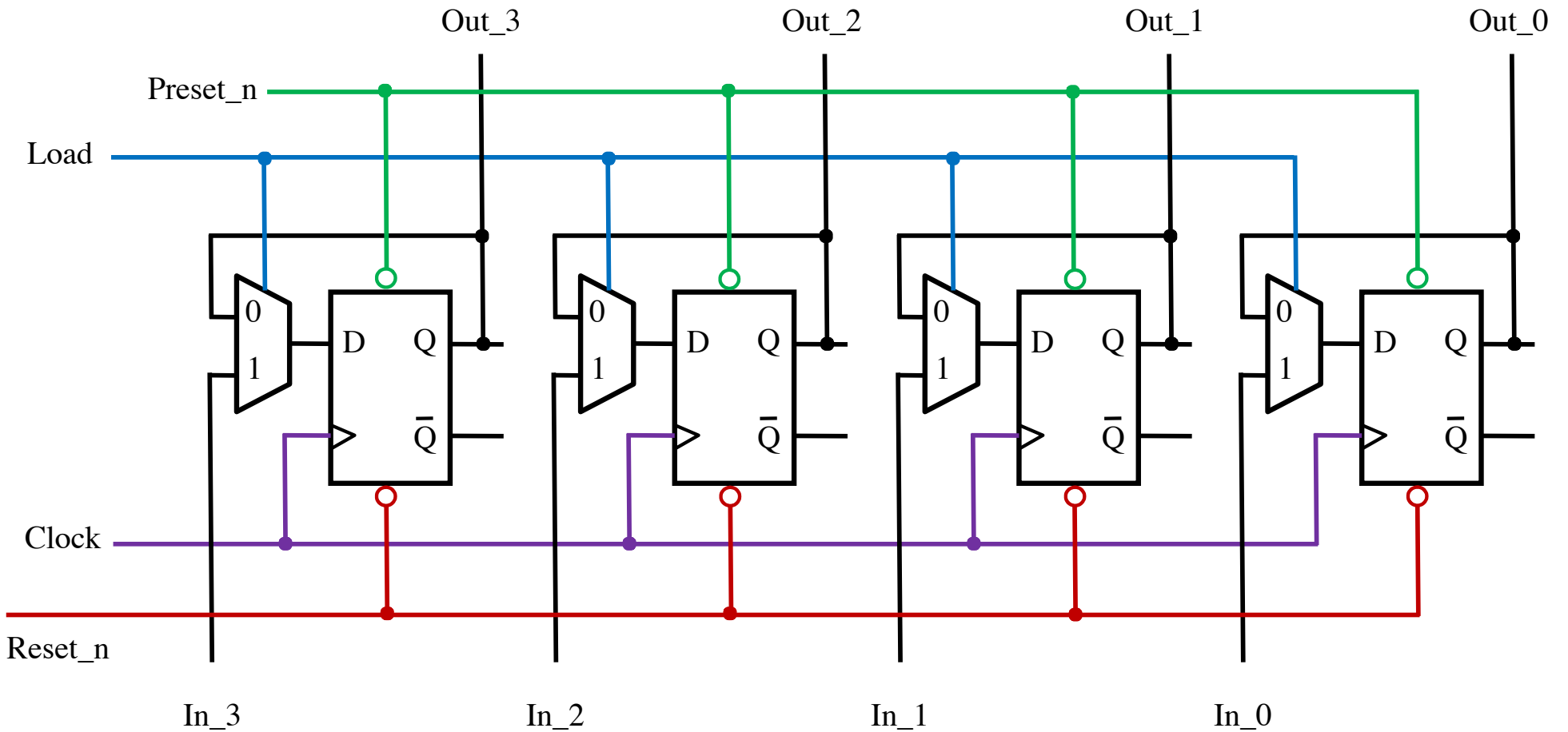
4-Bit Parallel-Access Register



4-Bit Parallel-Access Register

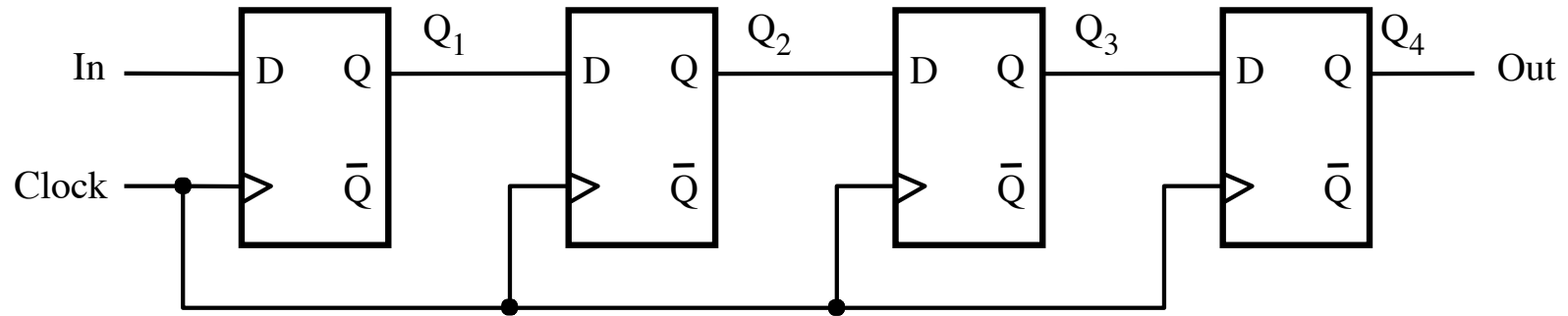


4-Bit Parallel-Access Register

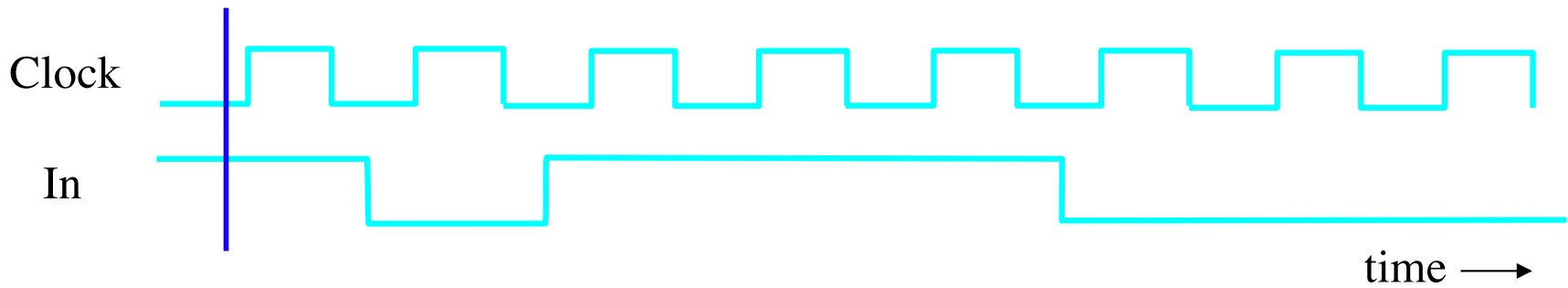
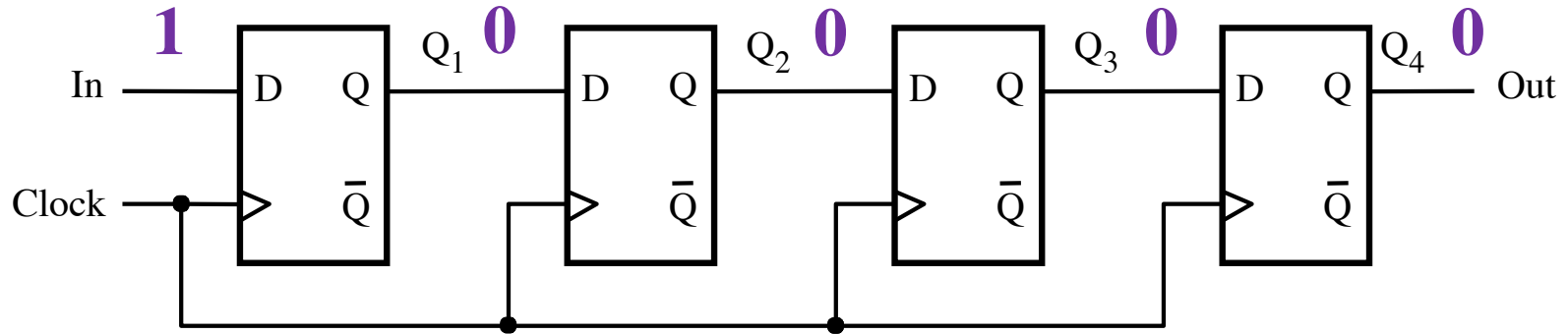


Shift Register

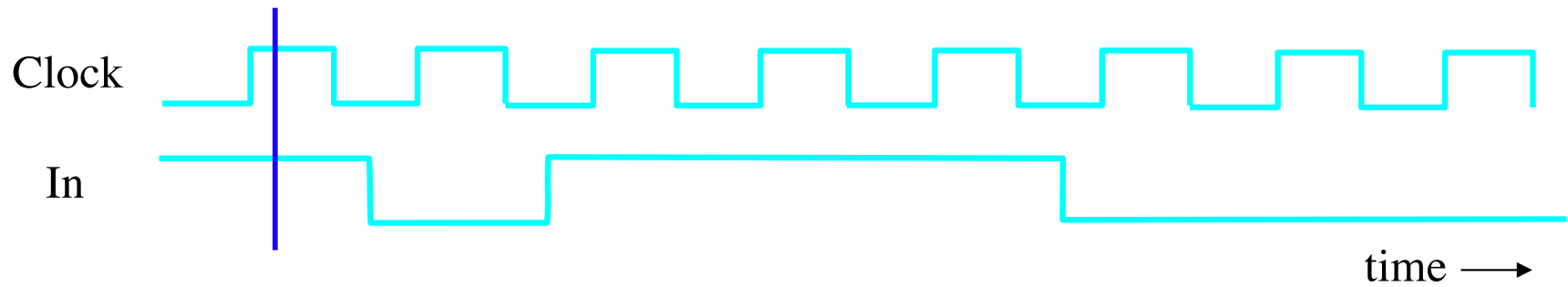
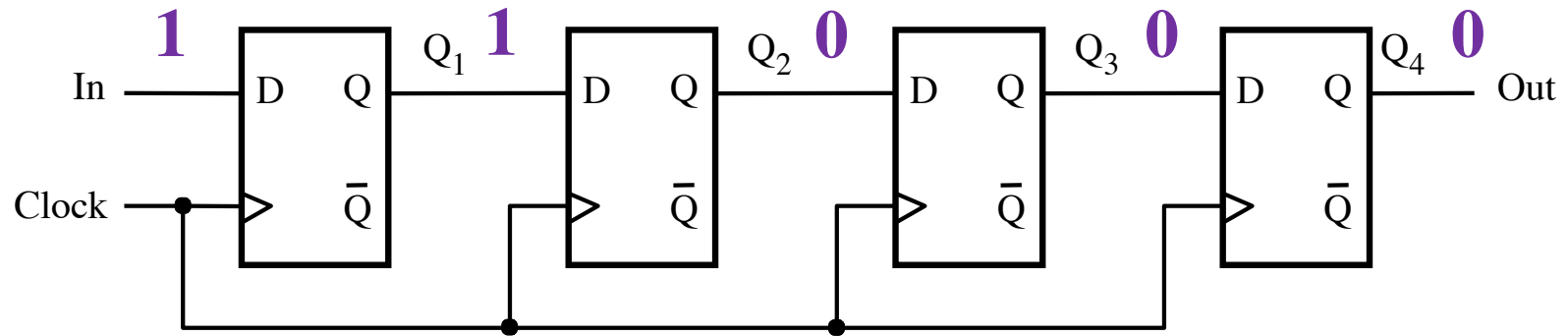
A simple shift register



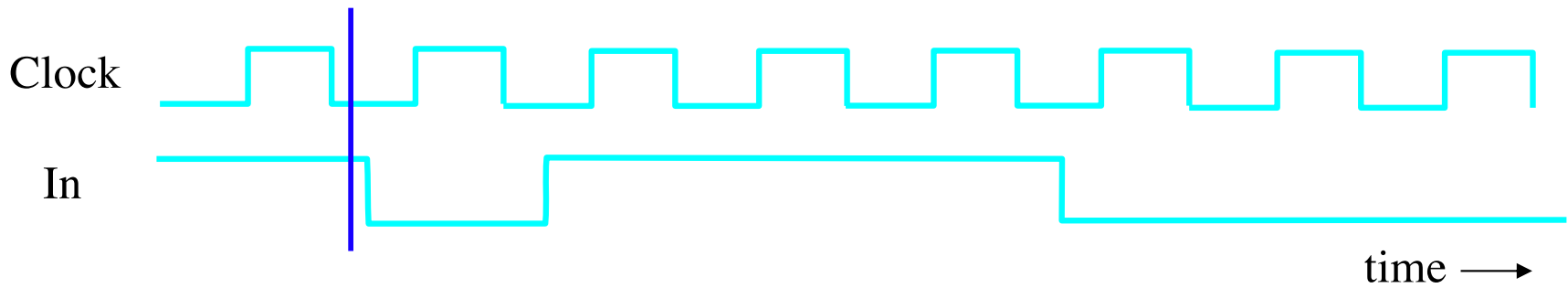
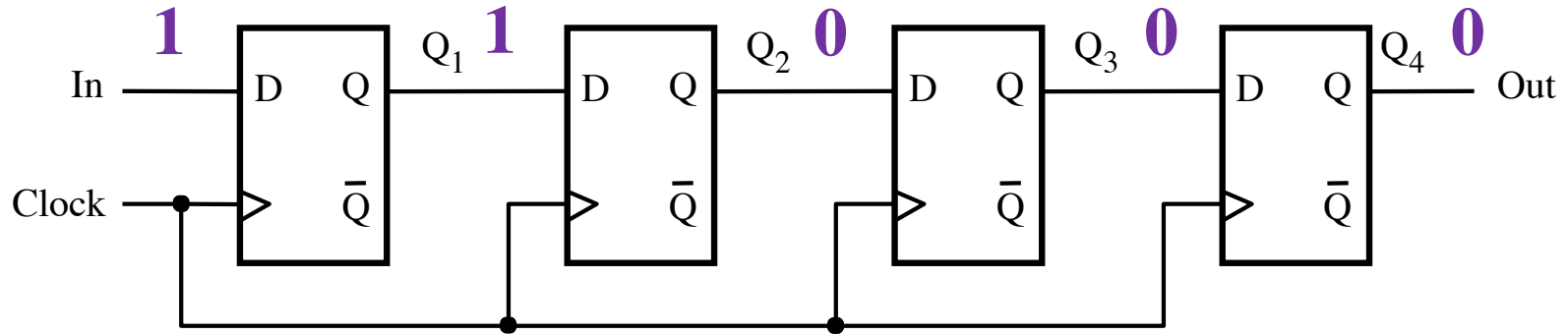
Shift Register Simulation



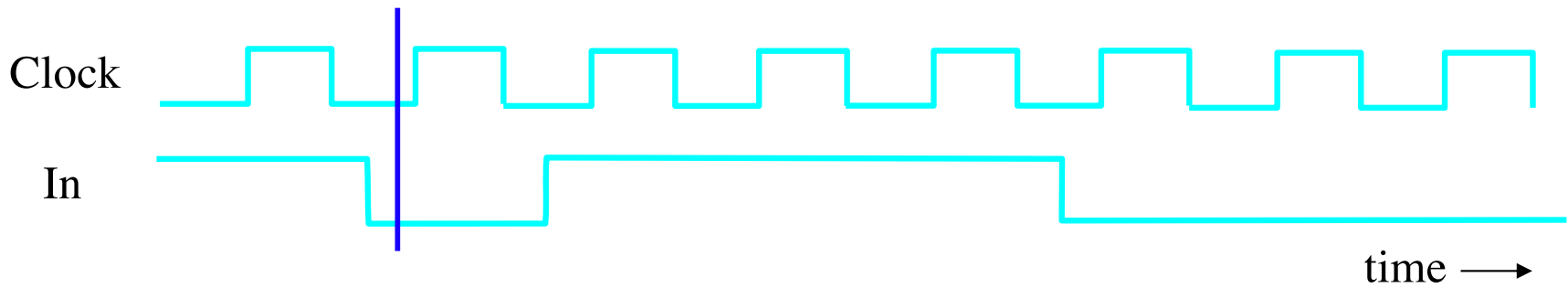
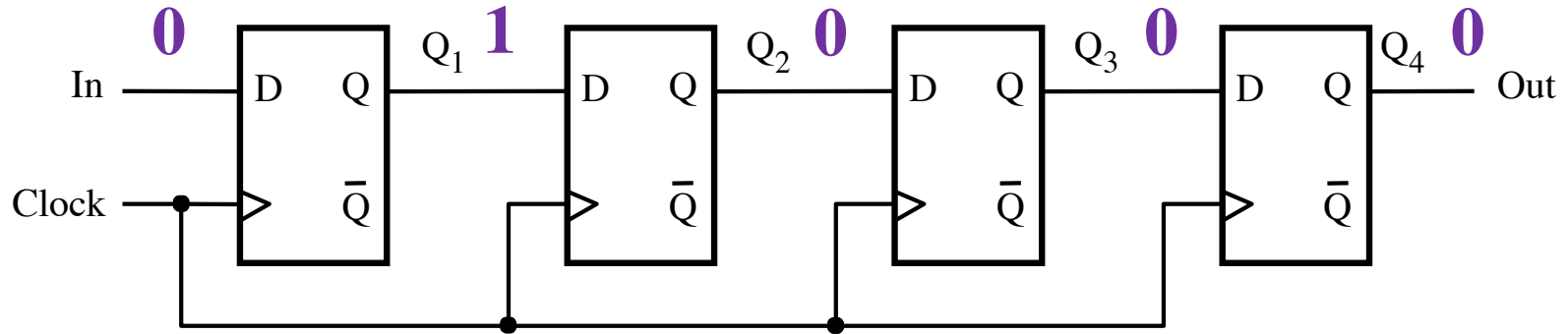
Shift Register Simulation



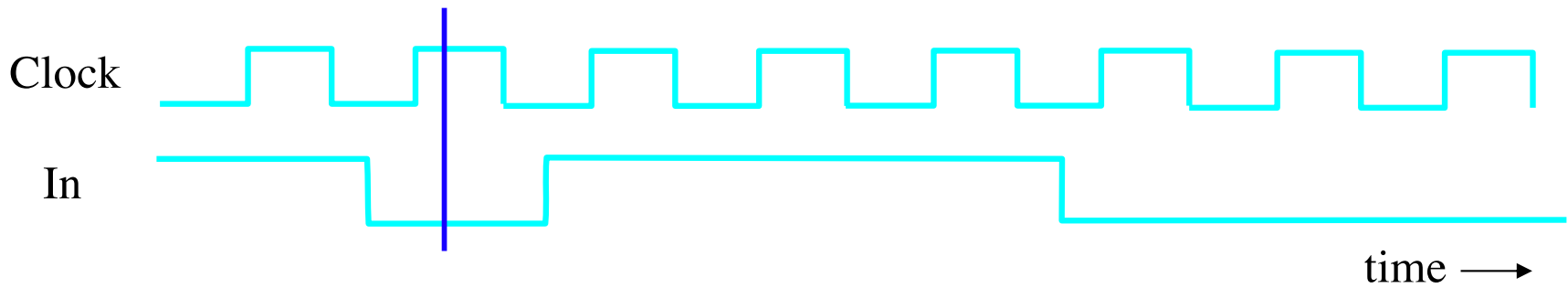
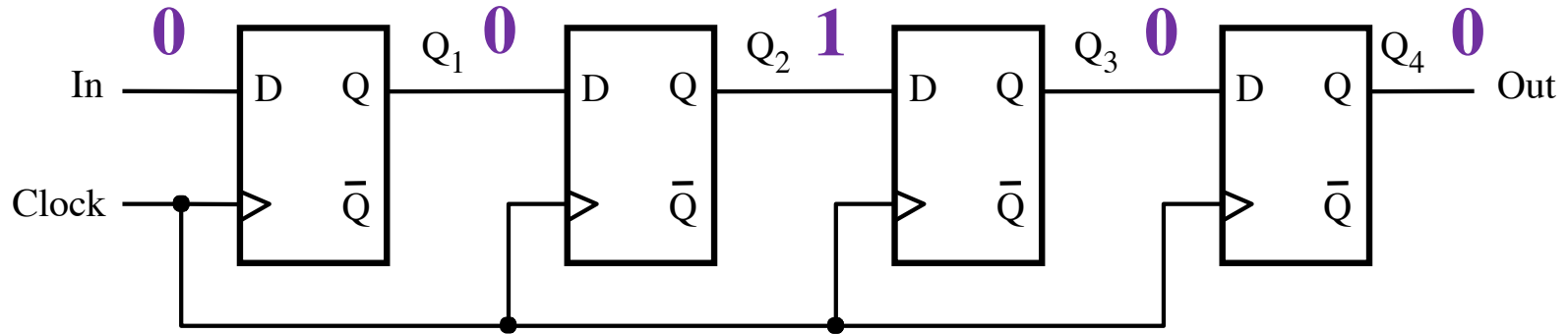
Shift Register Simulation



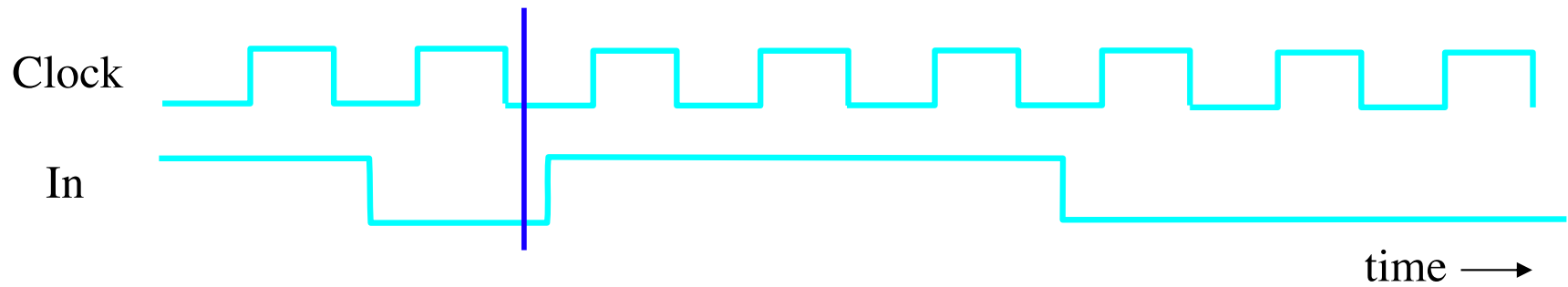
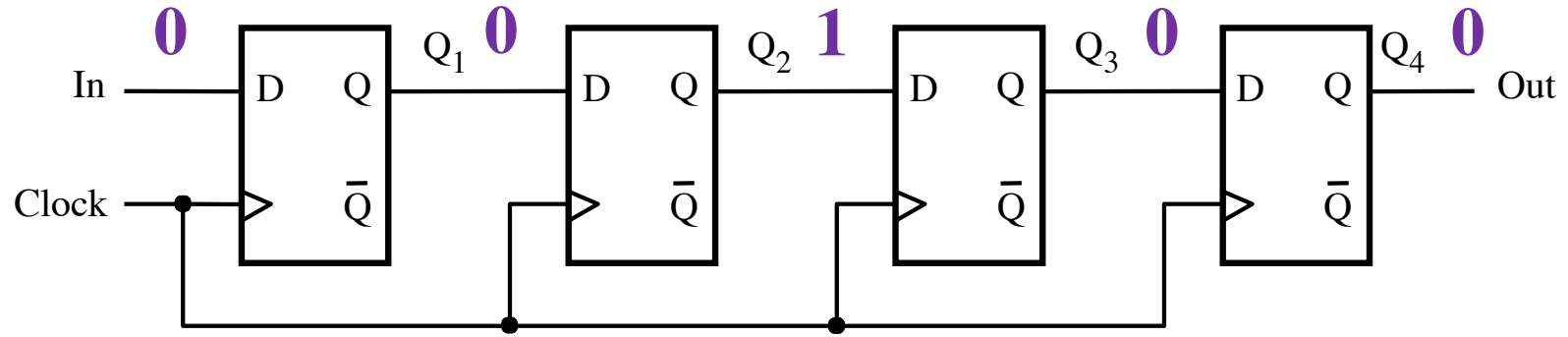
Shift Register Simulation



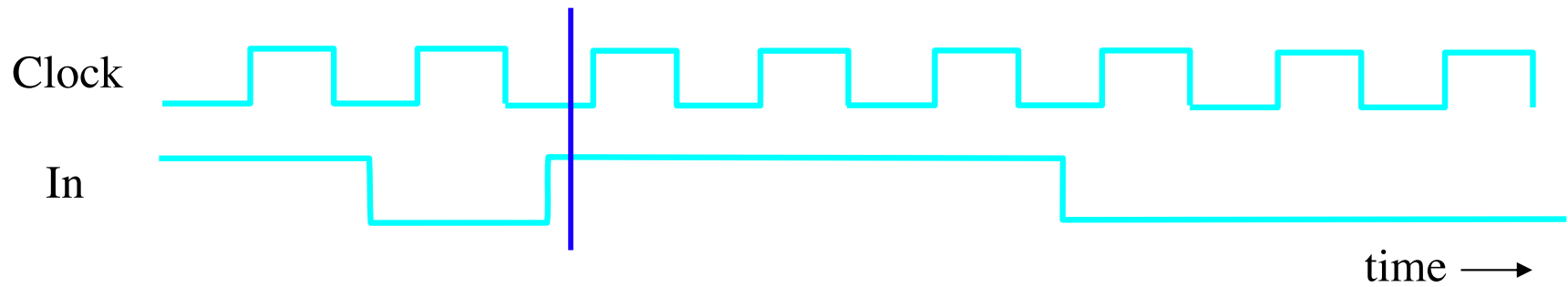
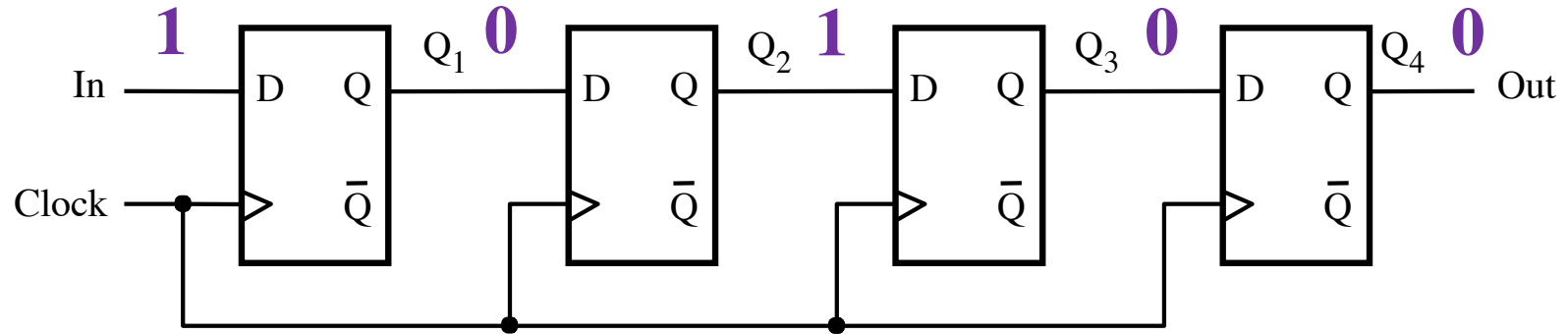
Shift Register Simulation



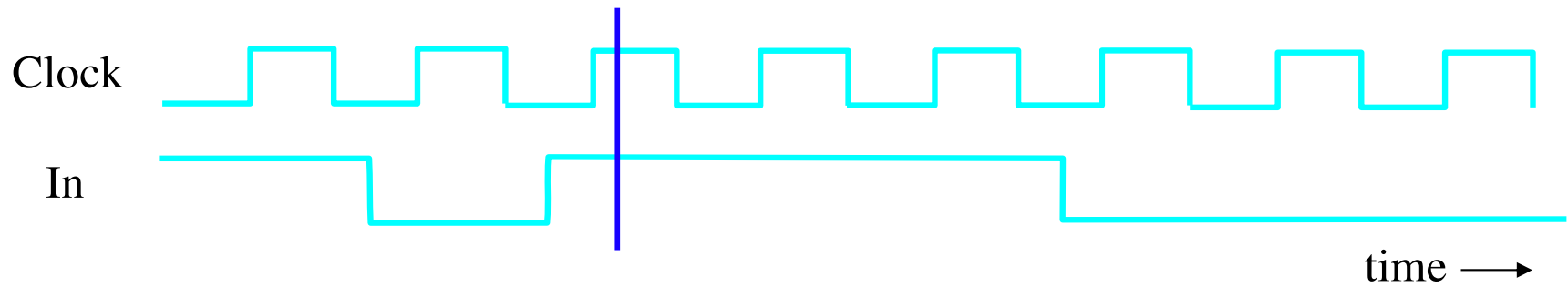
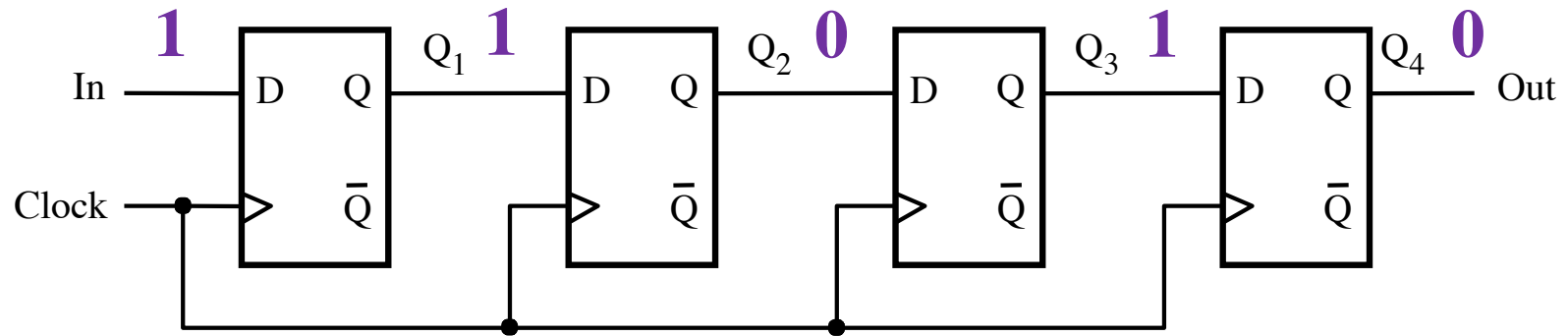
Shift Register Simulation



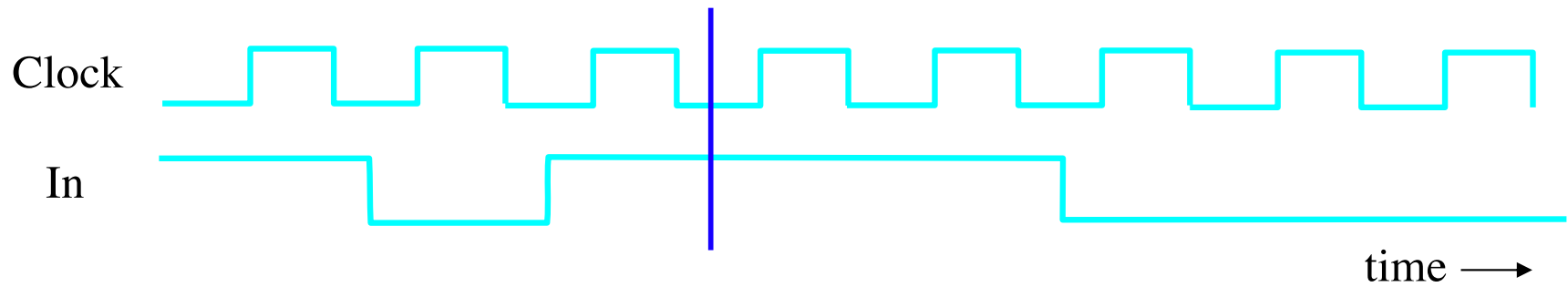
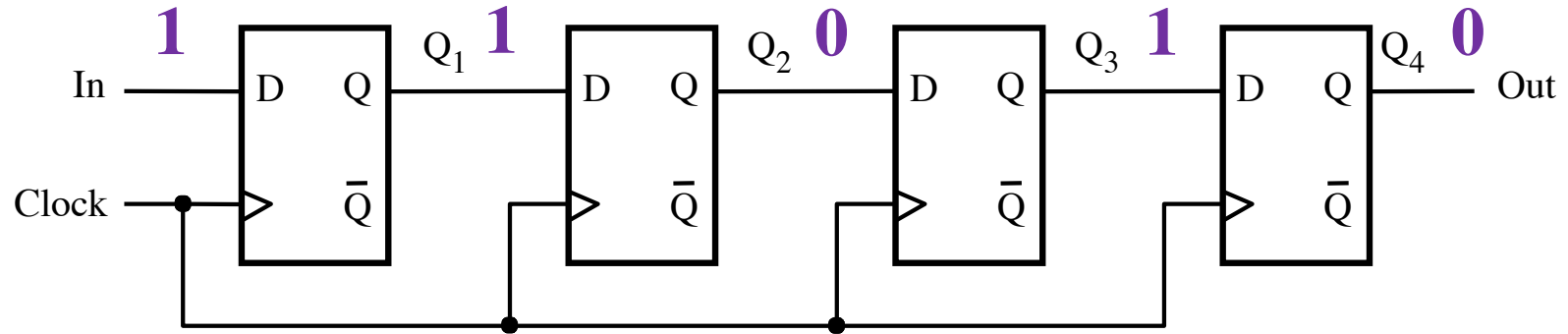
Shift Register Simulation



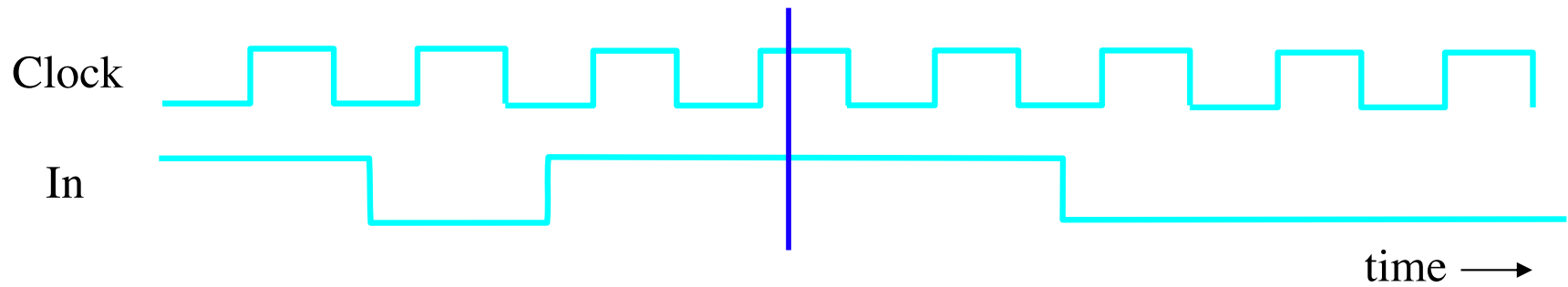
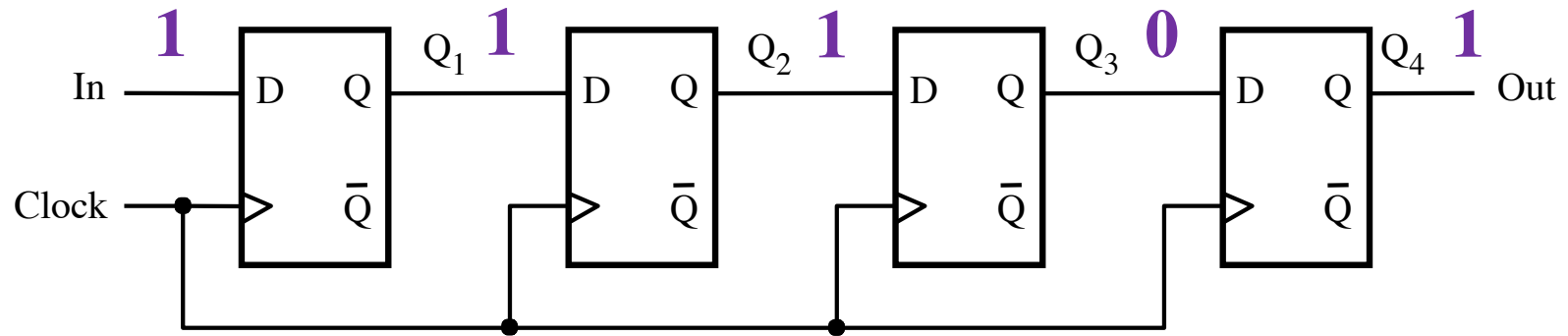
Shift Register Simulation



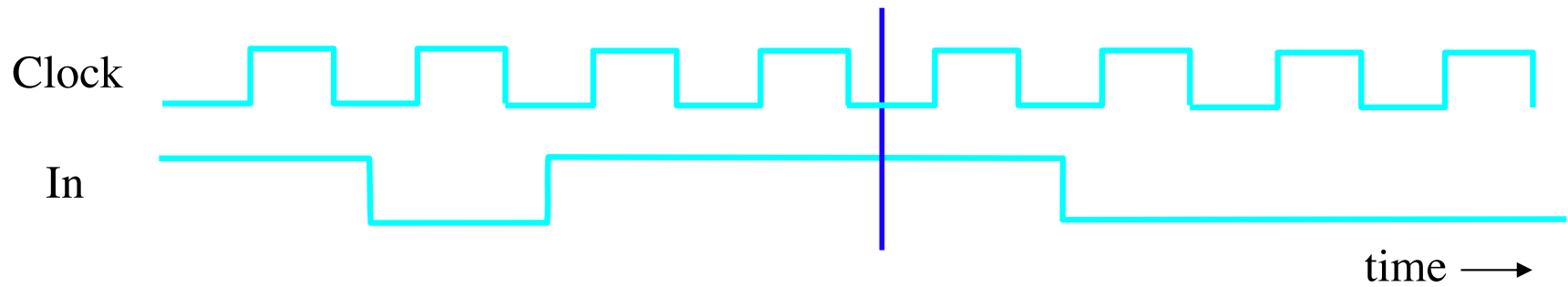
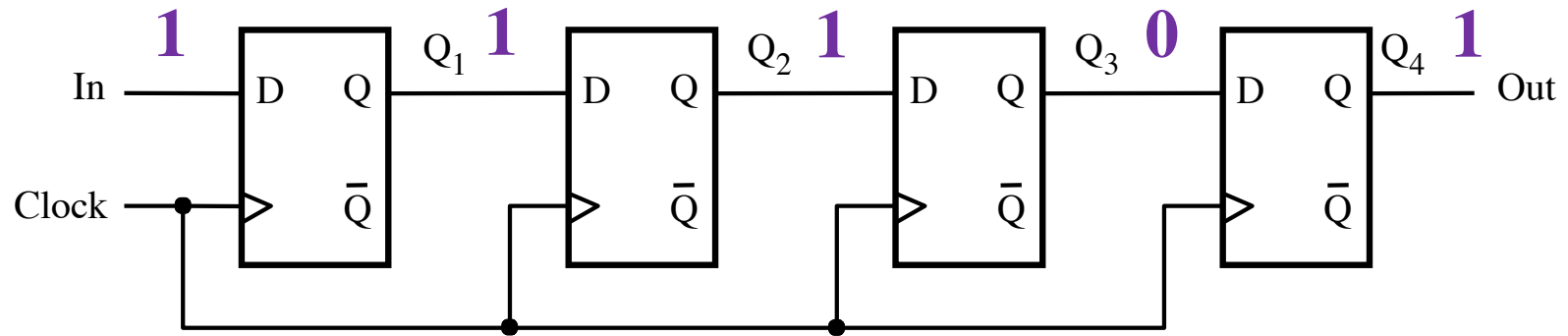
Shift Register Simulation



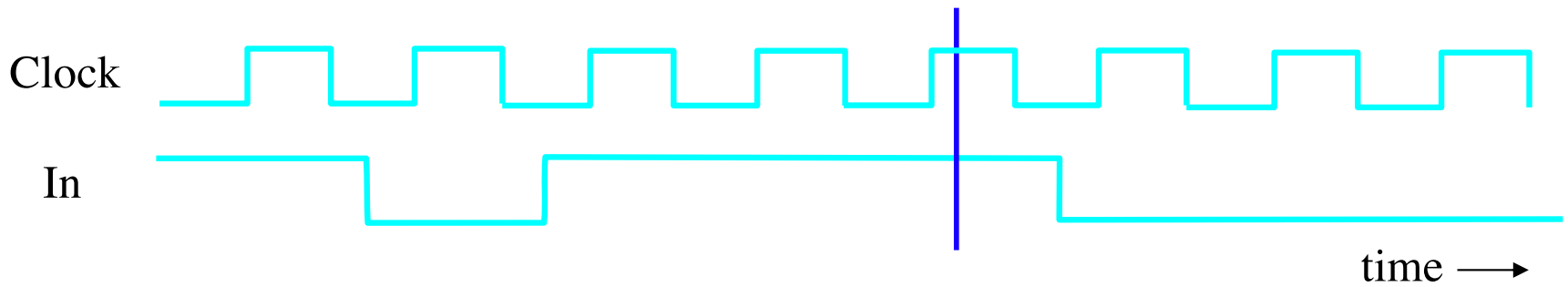
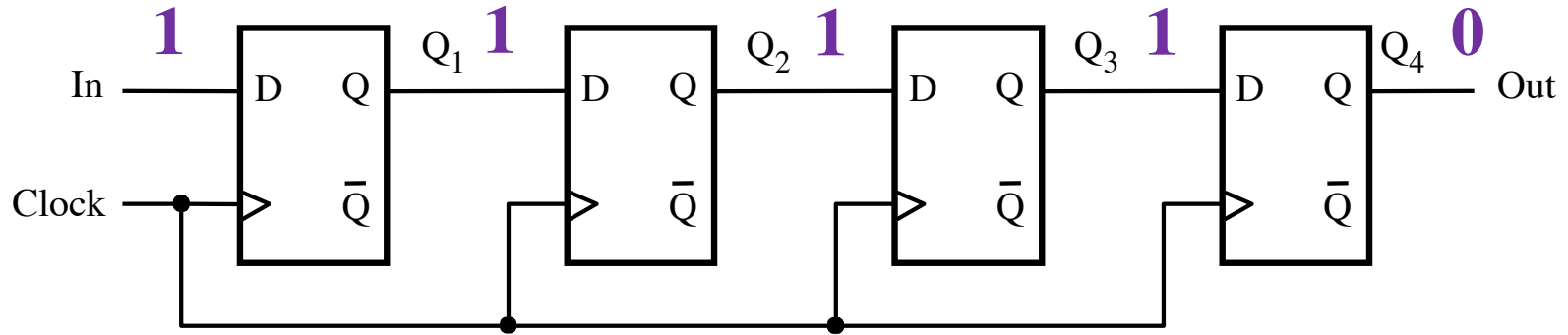
Shift Register Simulation



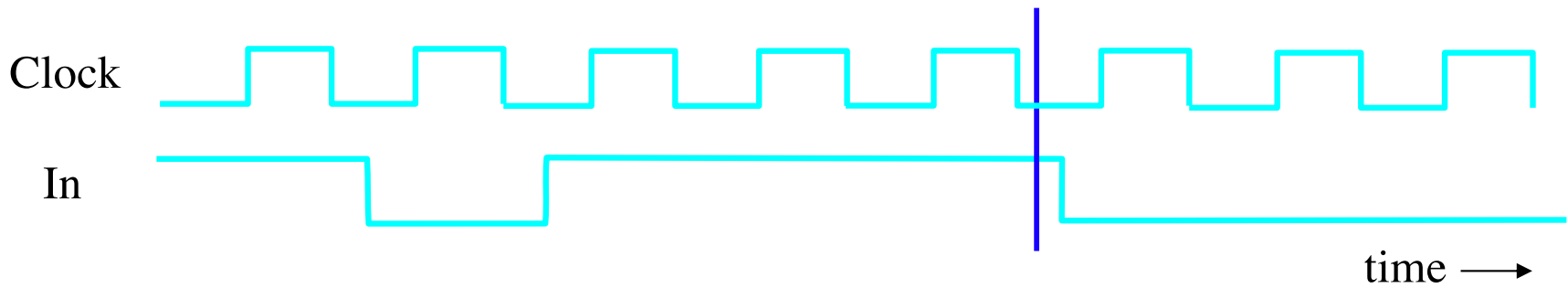
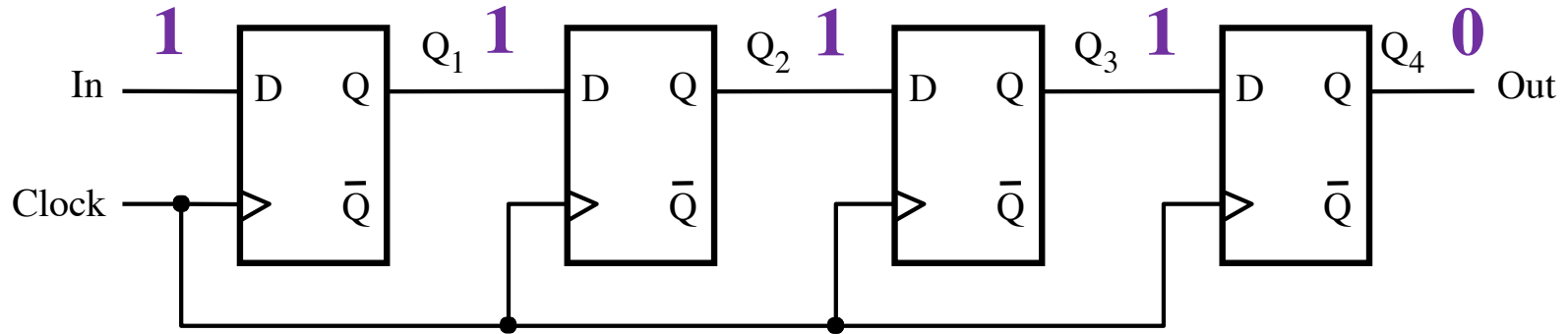
Shift Register Simulation



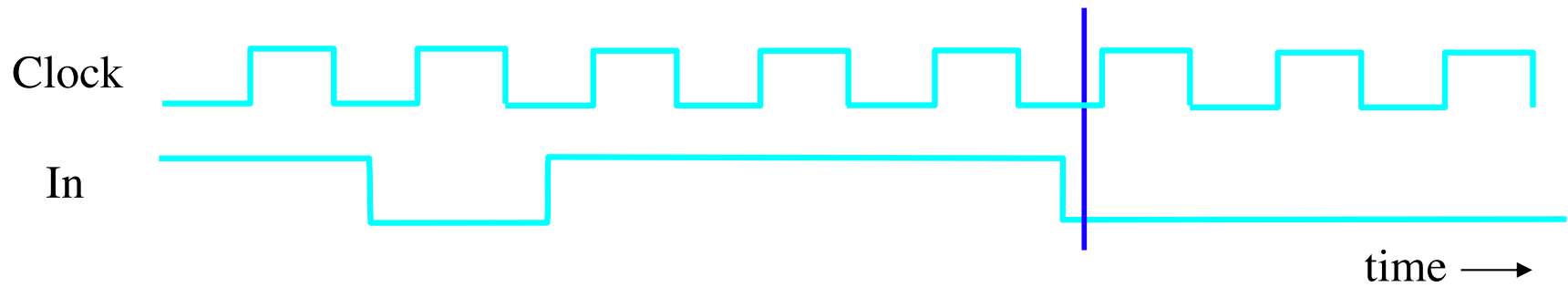
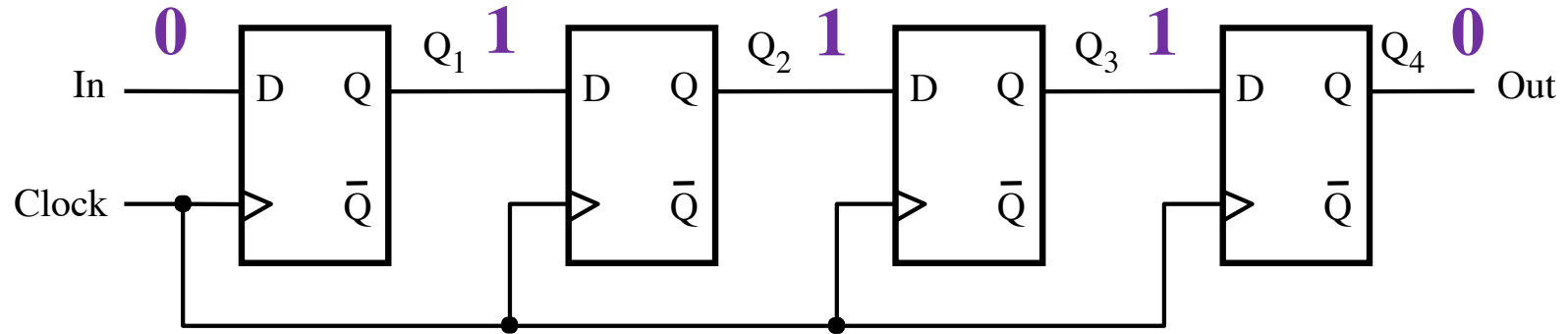
Shift Register Simulation



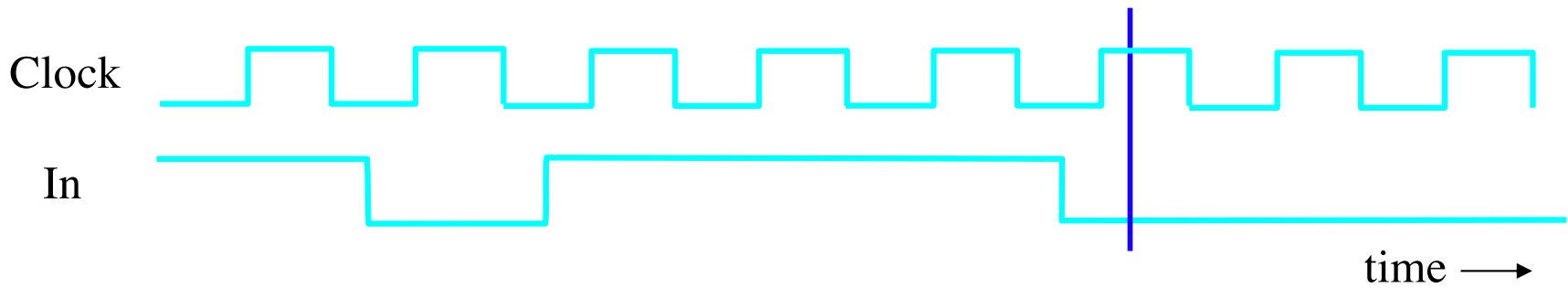
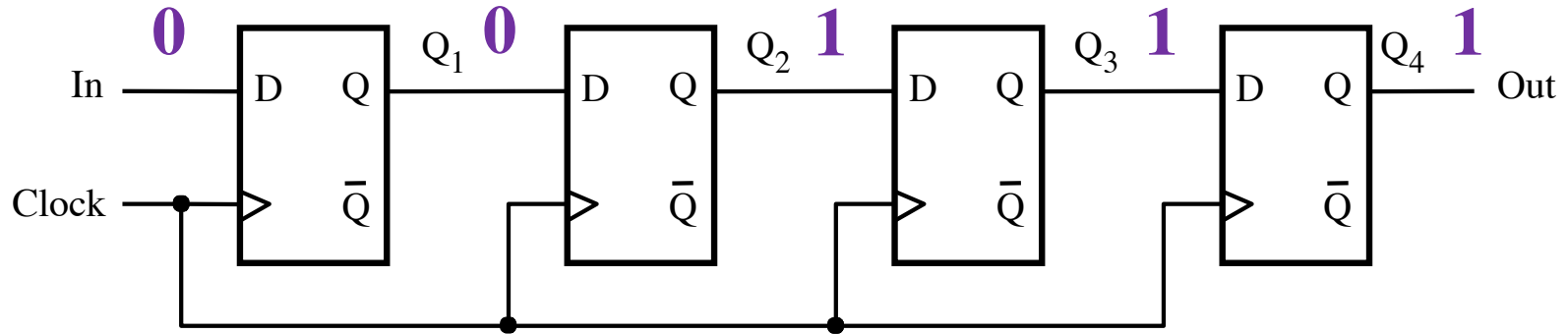
Shift Register Simulation



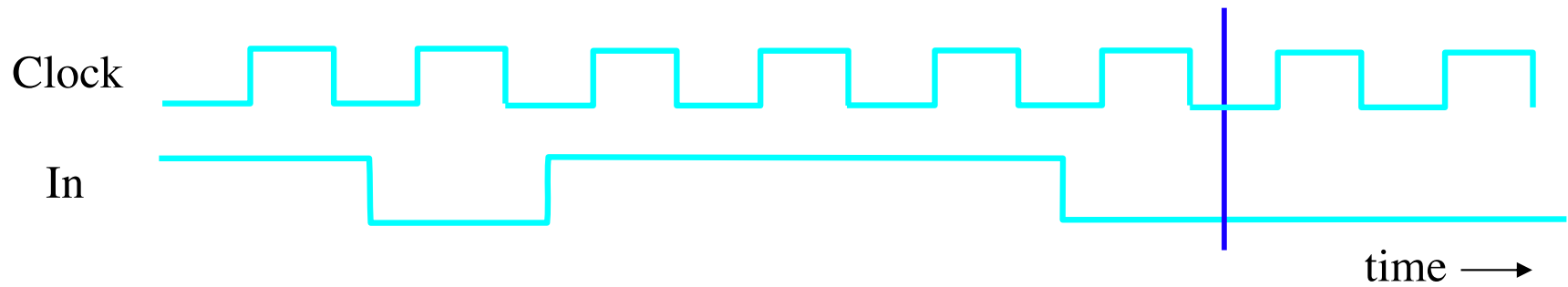
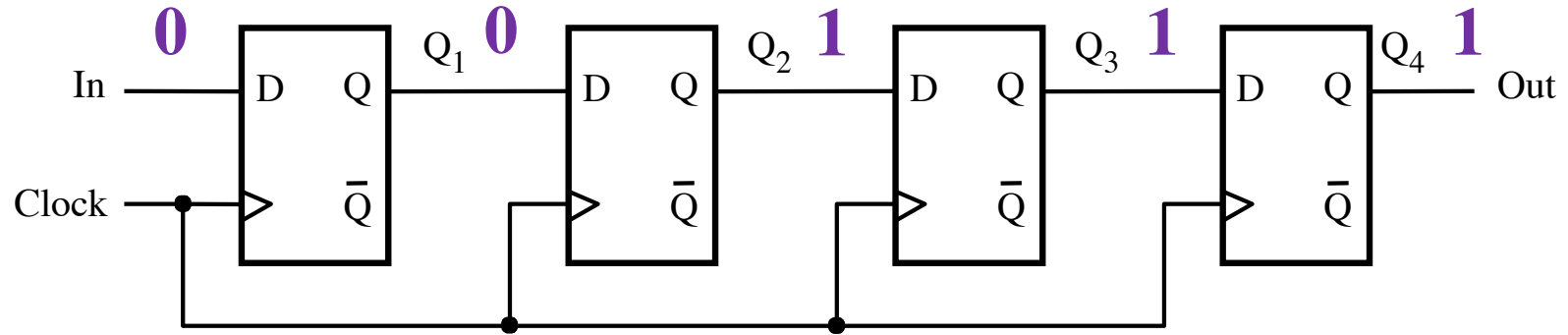
Shift Register Simulation



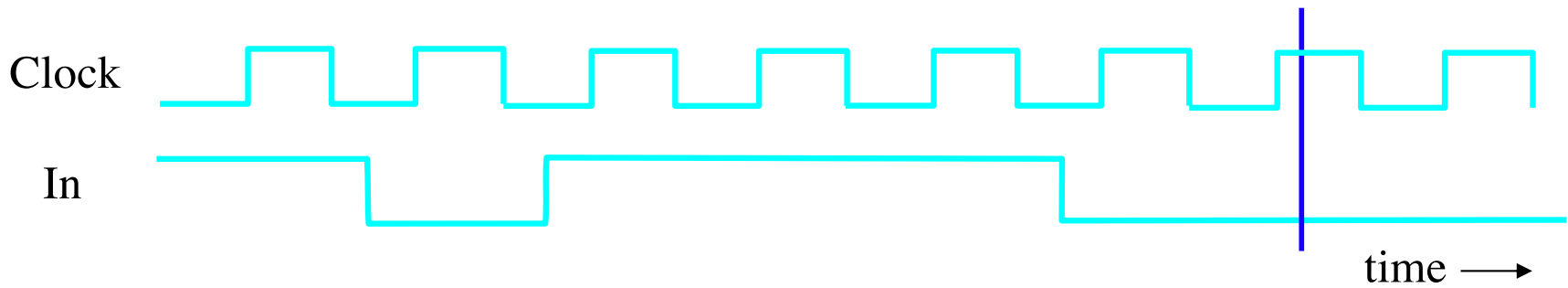
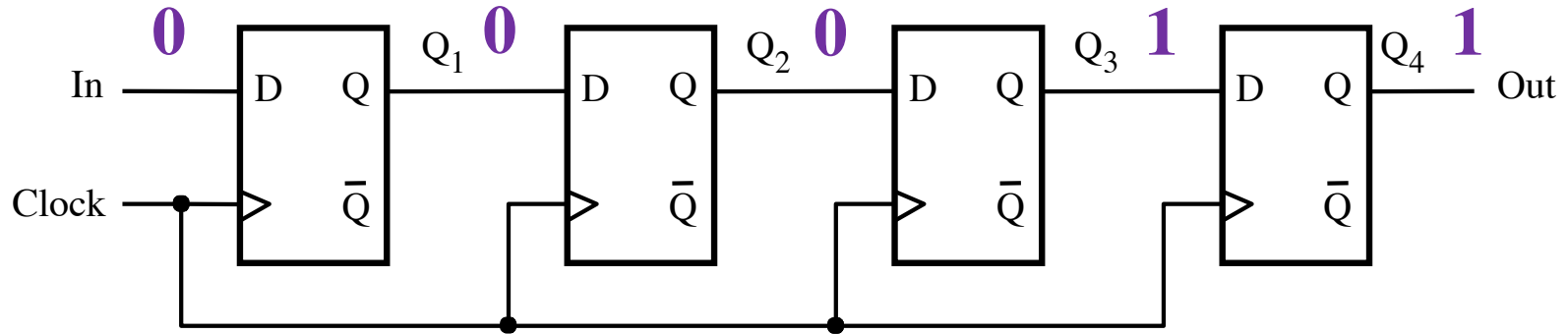
Shift Register Simulation



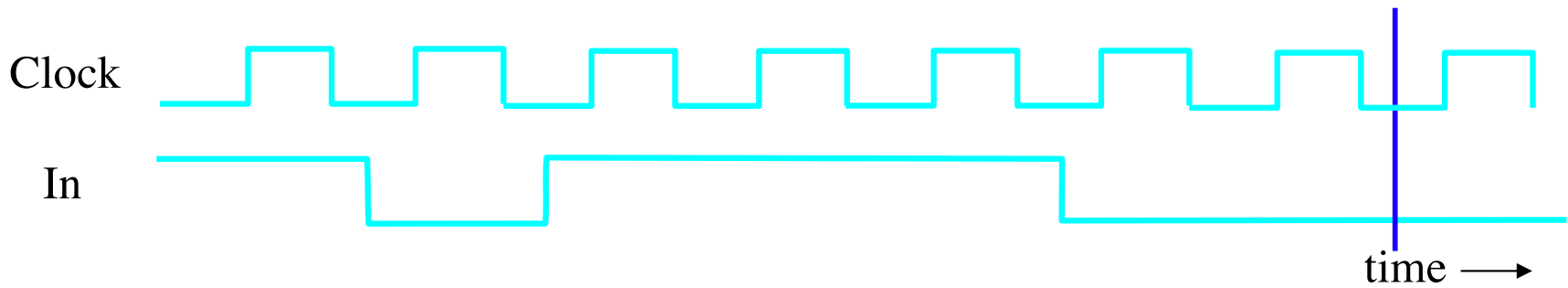
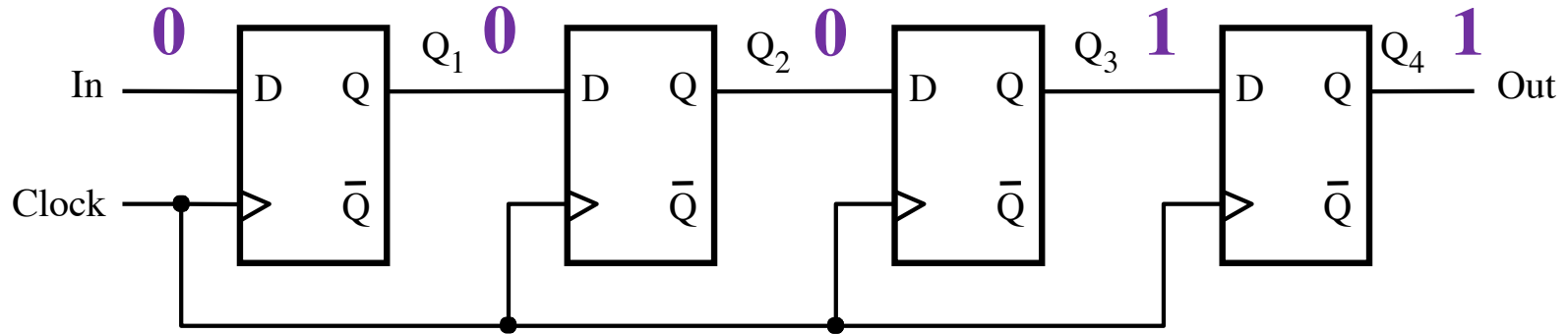
Shift Register Simulation



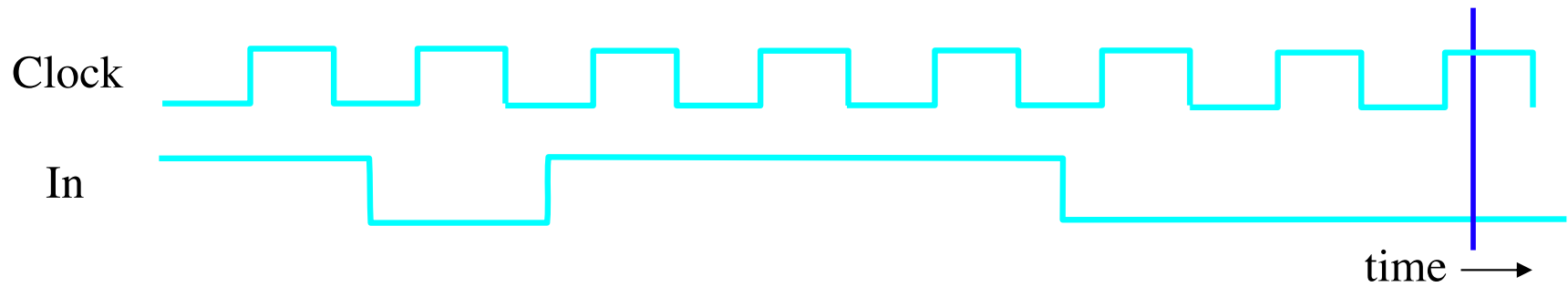
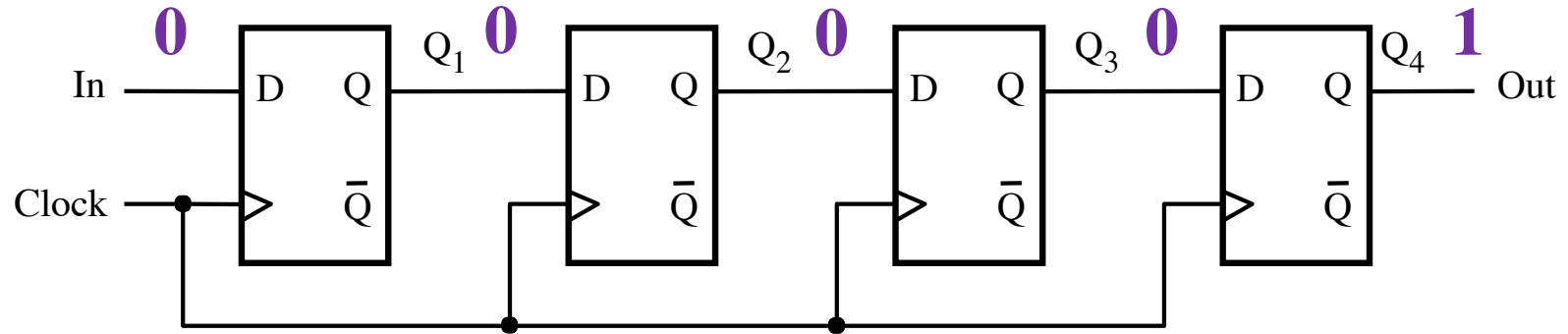
Shift Register Simulation



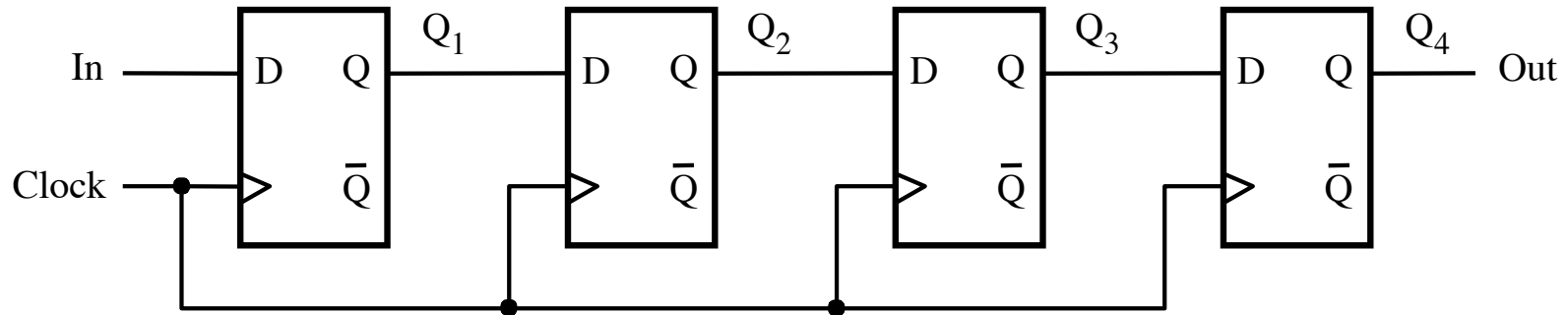
Shift Register Simulation



Shift Register Simulation



A simple shift register

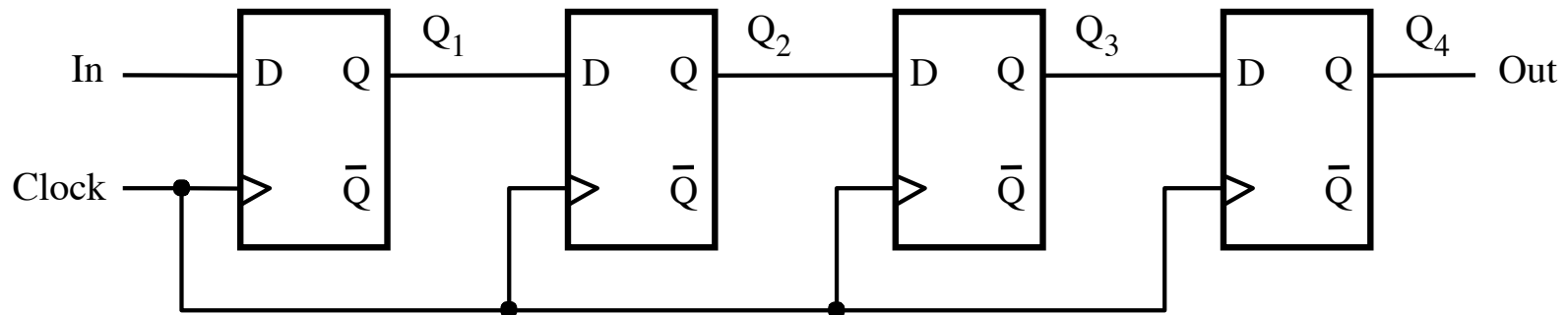


(a) Circuit

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

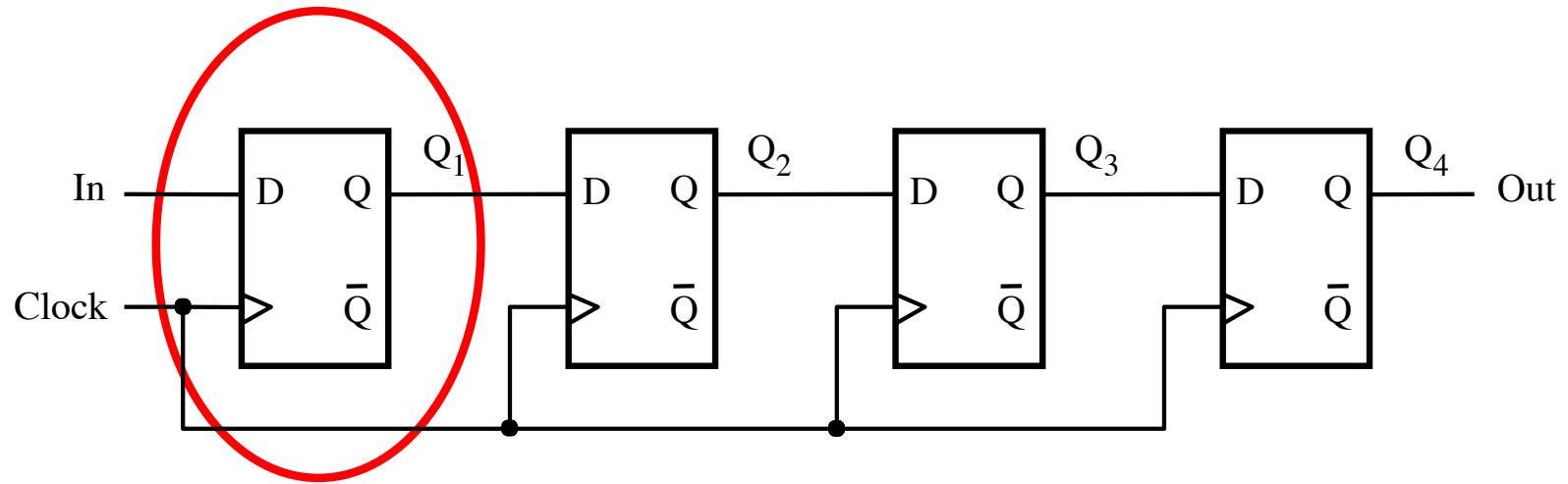


(a) Circuit

time	In	Q_1	Q_2	Q_3	$Q_4 = \text{Out}$
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
	0	0	0	0	1

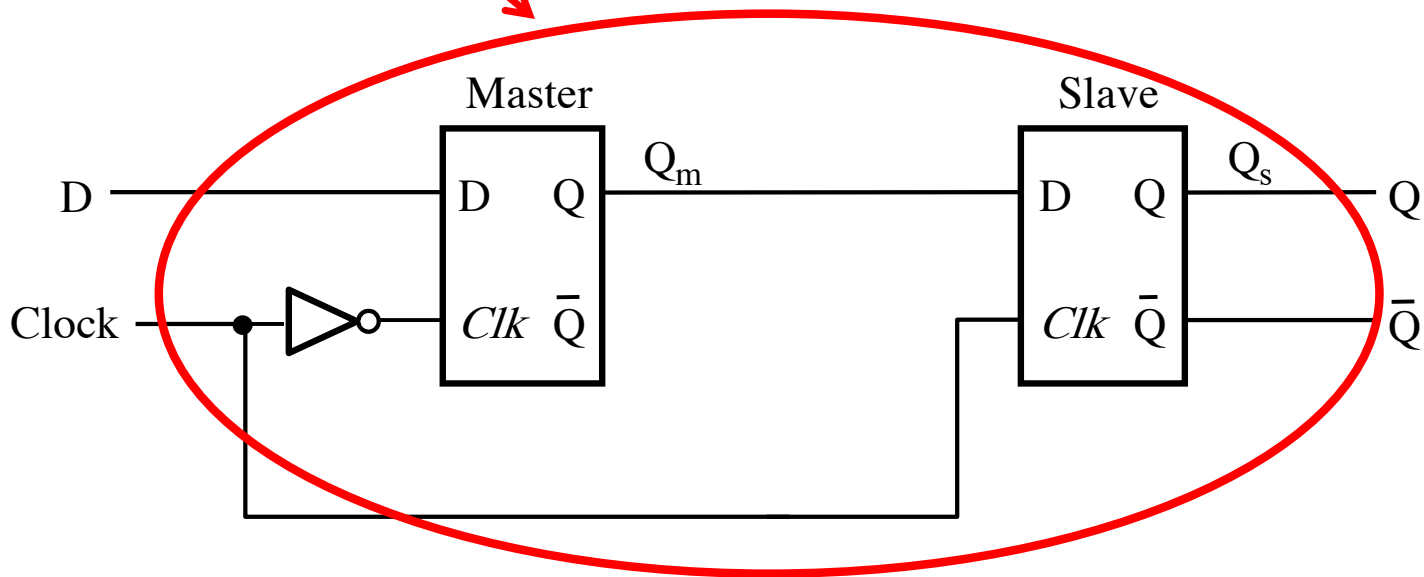
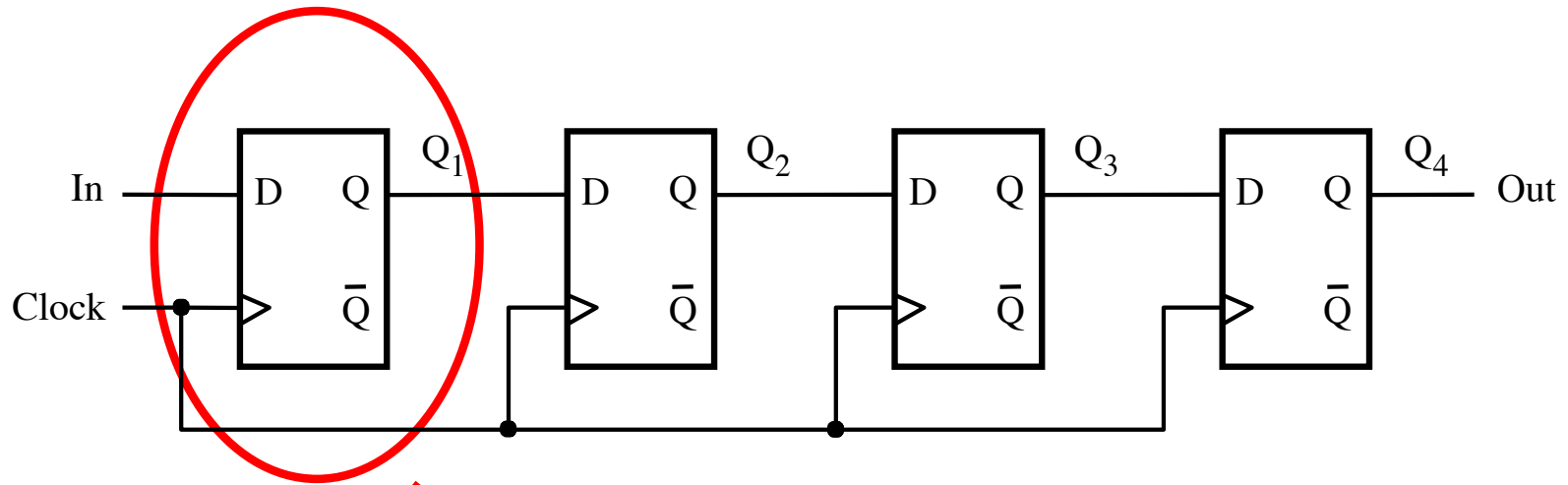
The simulation goes
one step further

A simple shift register

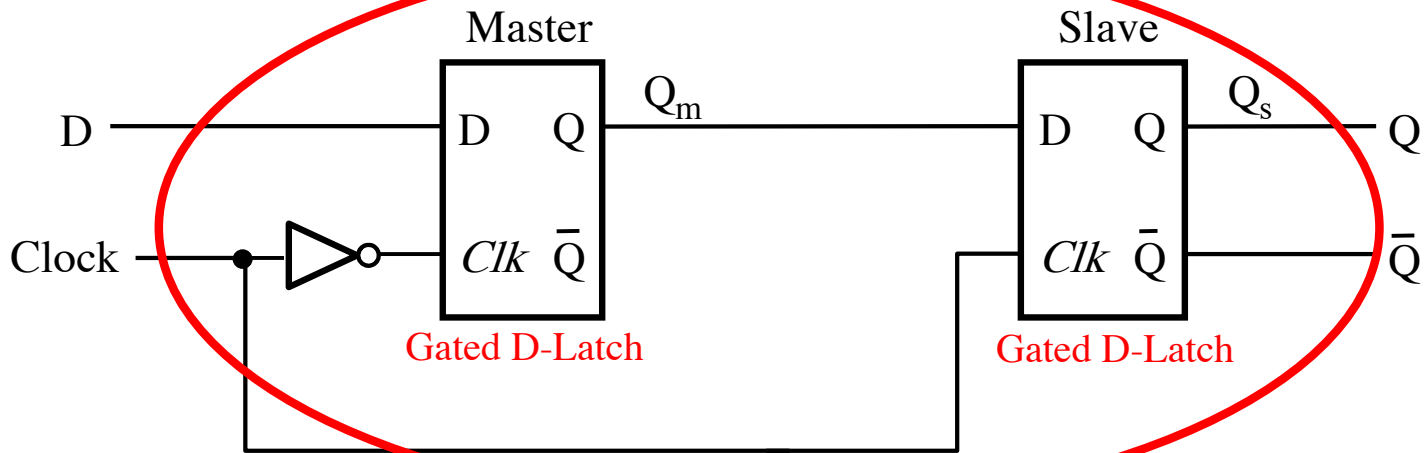
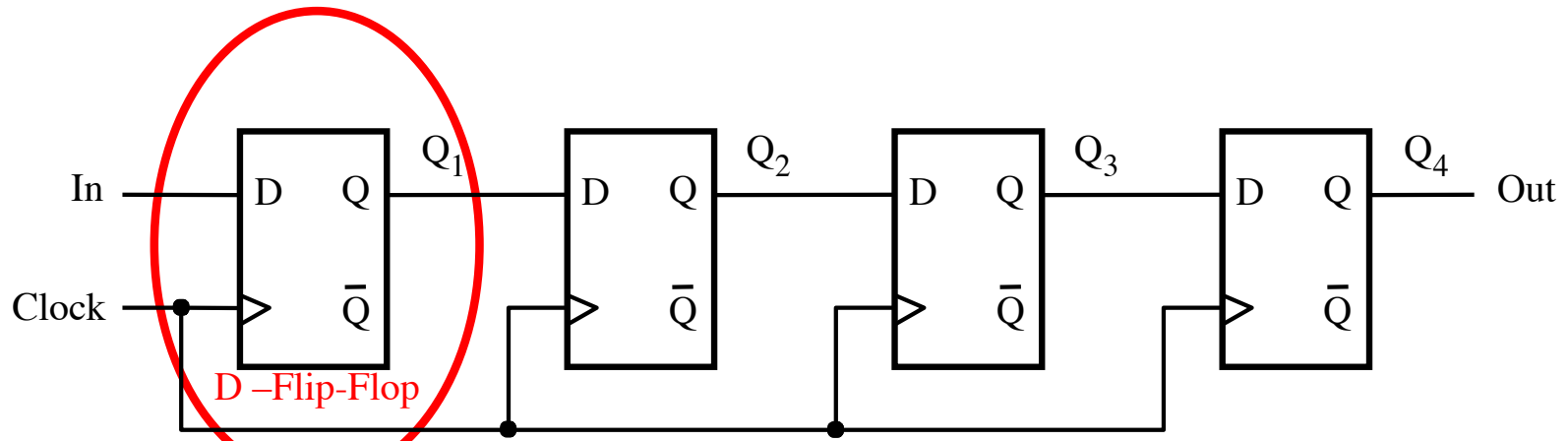


Positive-edge-triggered
D Flip-Flop

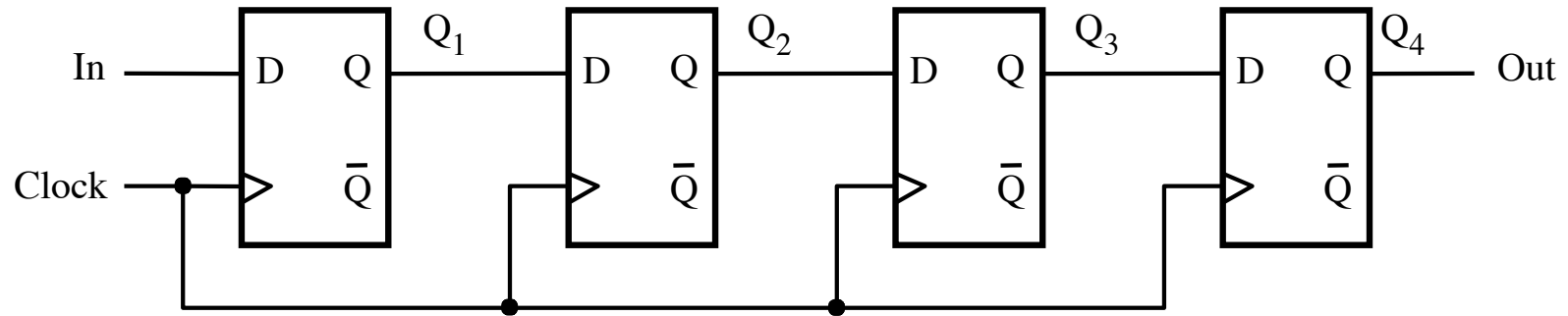
A simple shift register



A simple shift register



A simple shift register



You need 2 latches to make a flip-flop



=



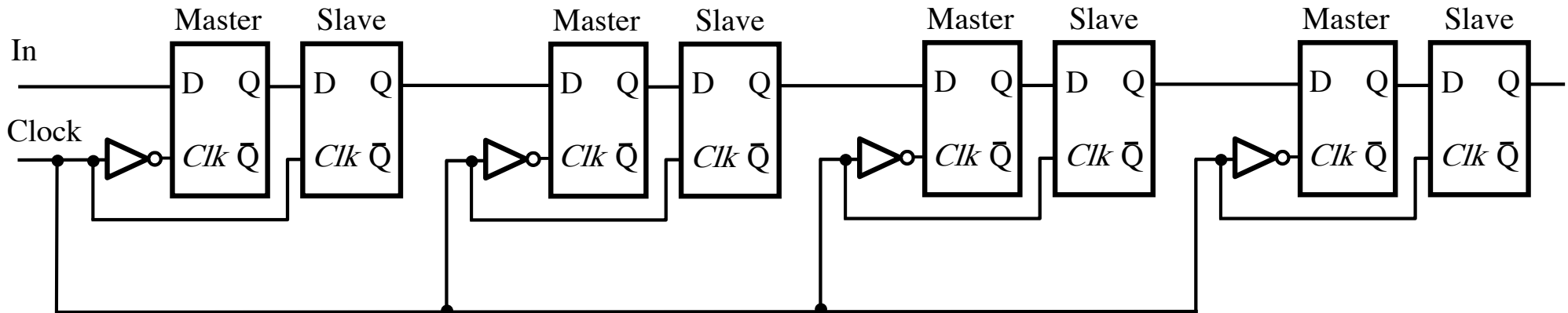
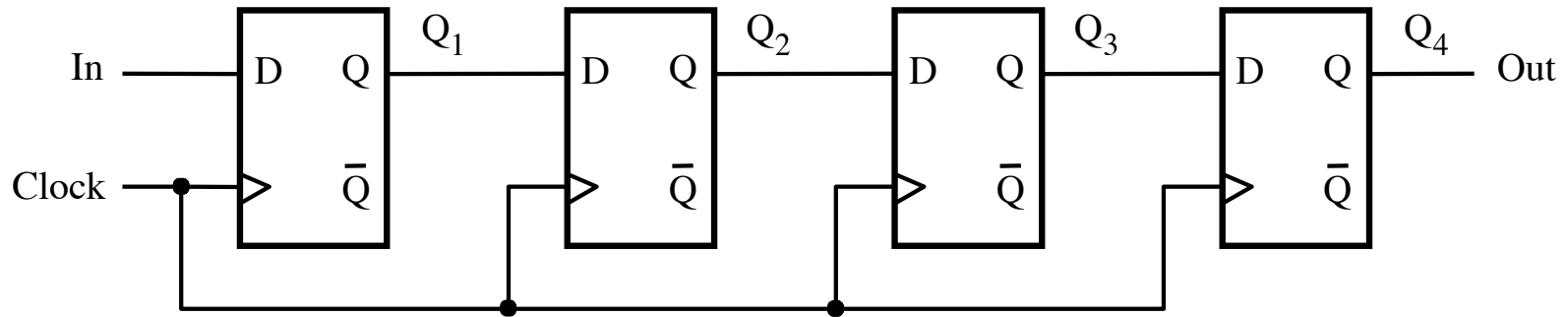
4-bit Register



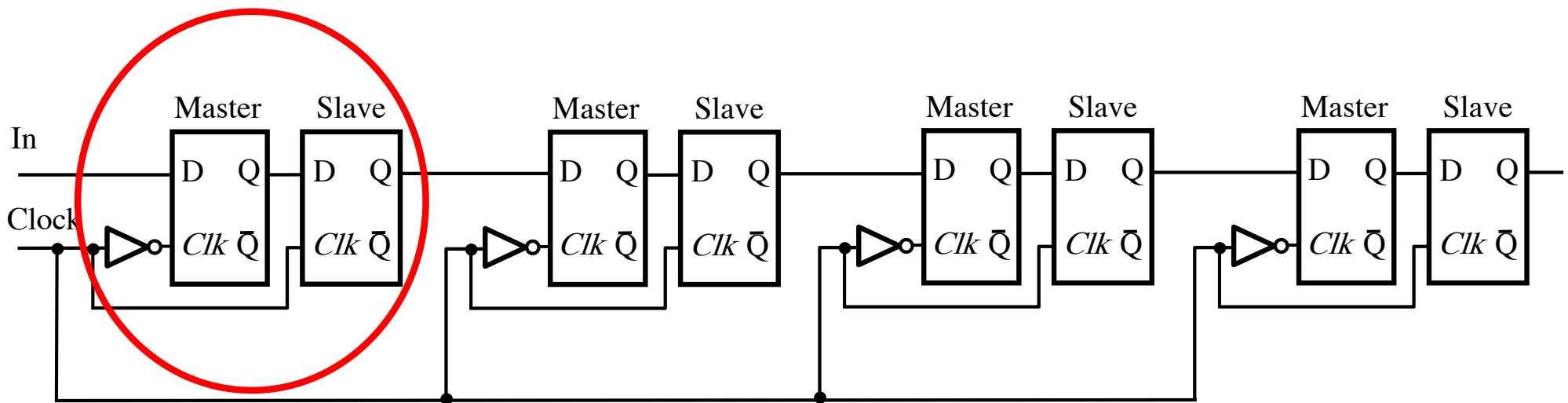
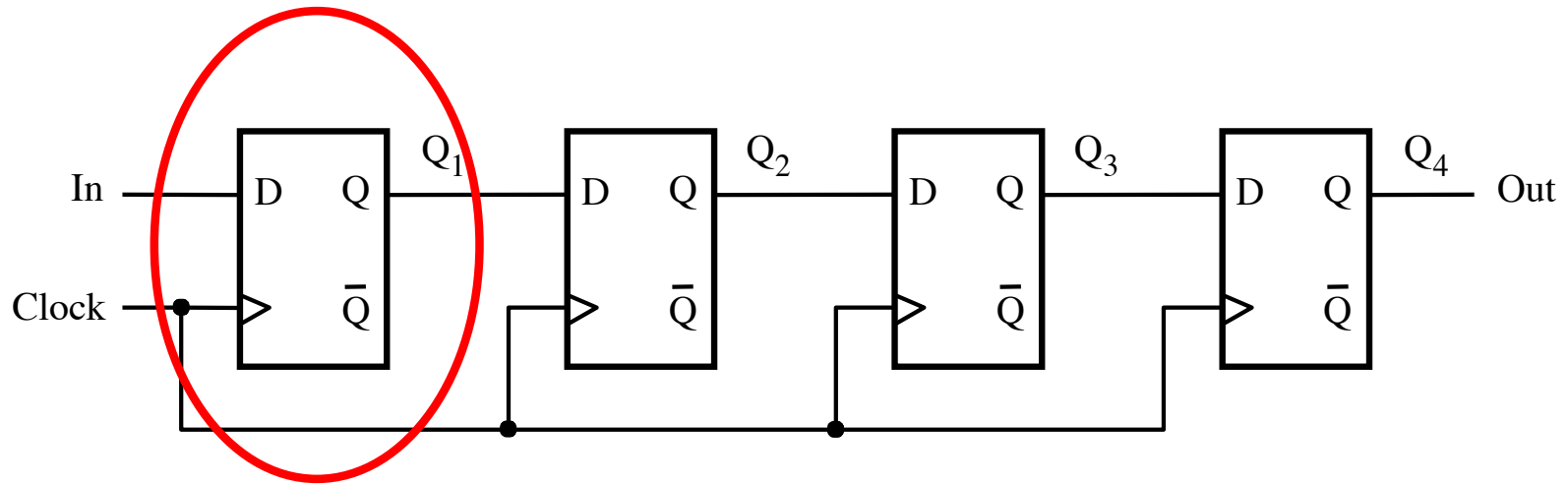
4-bit Register



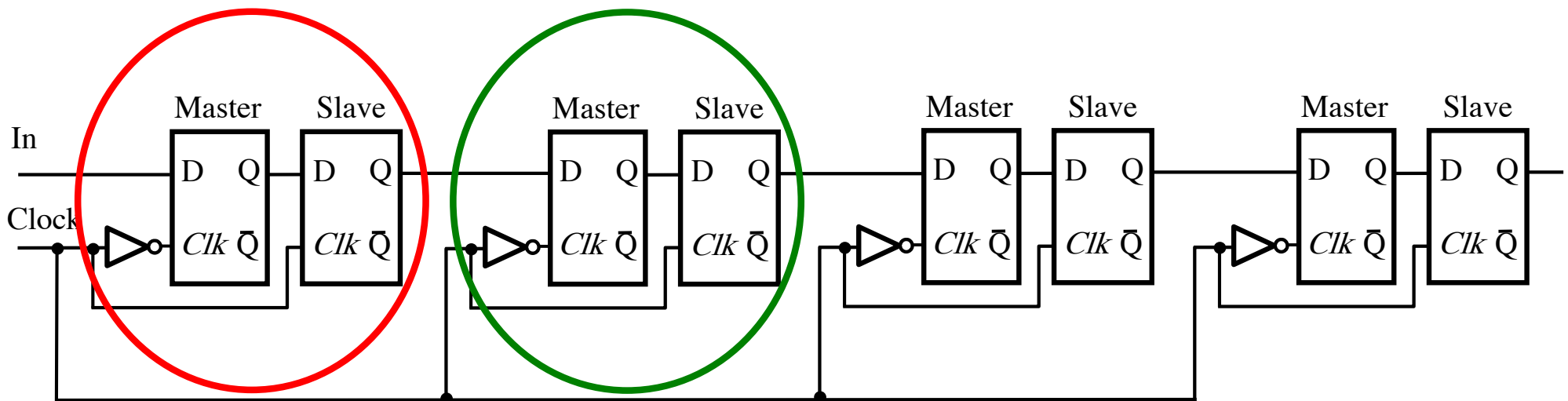
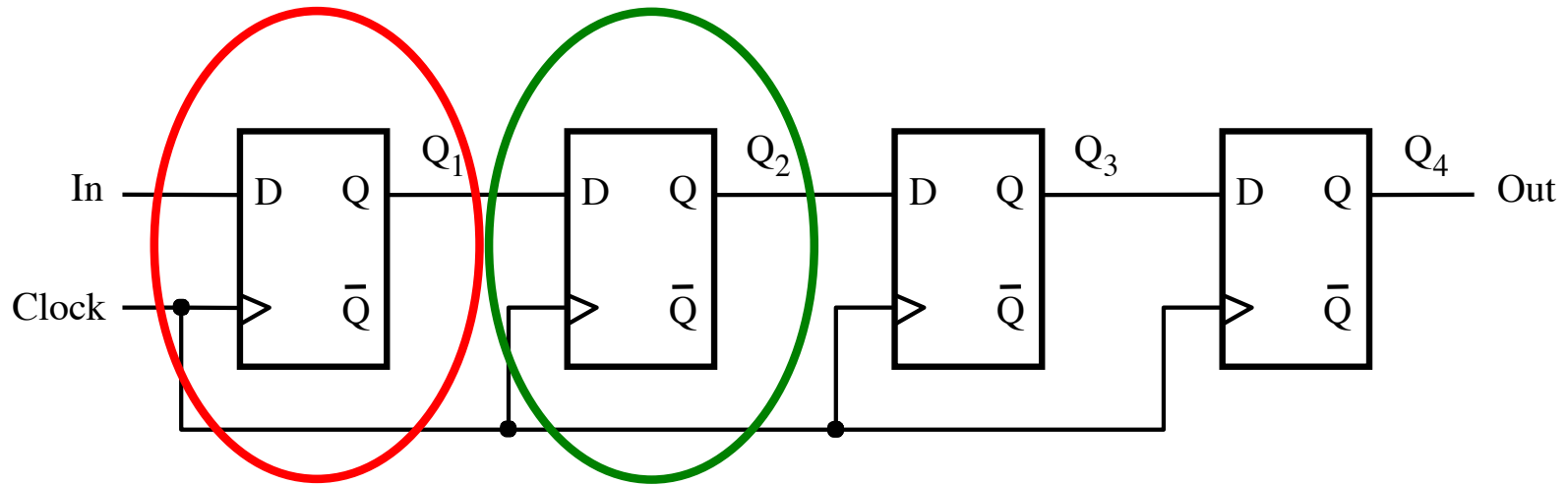
A simple shift register



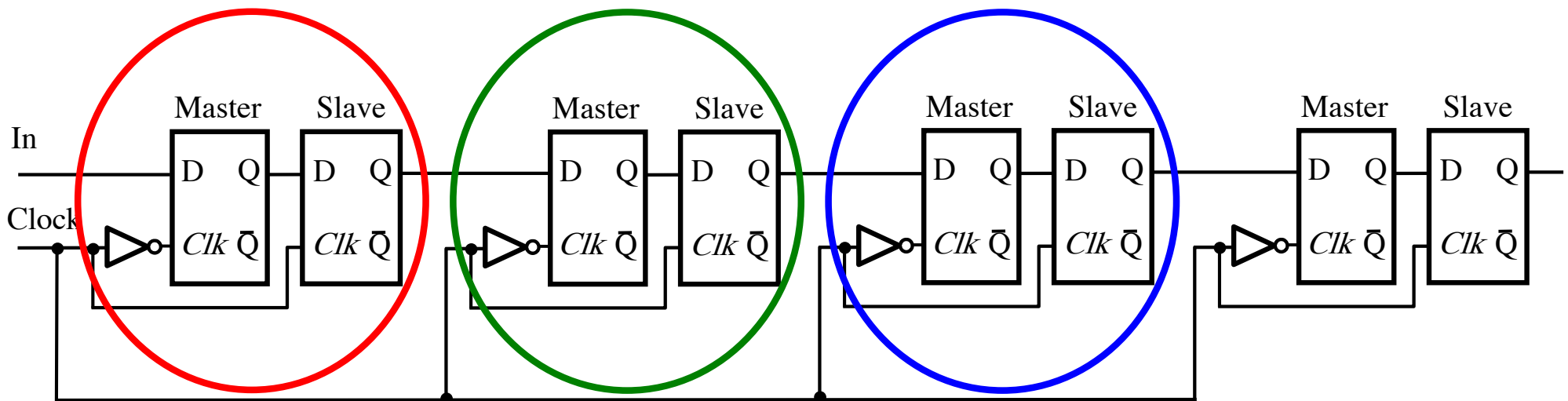
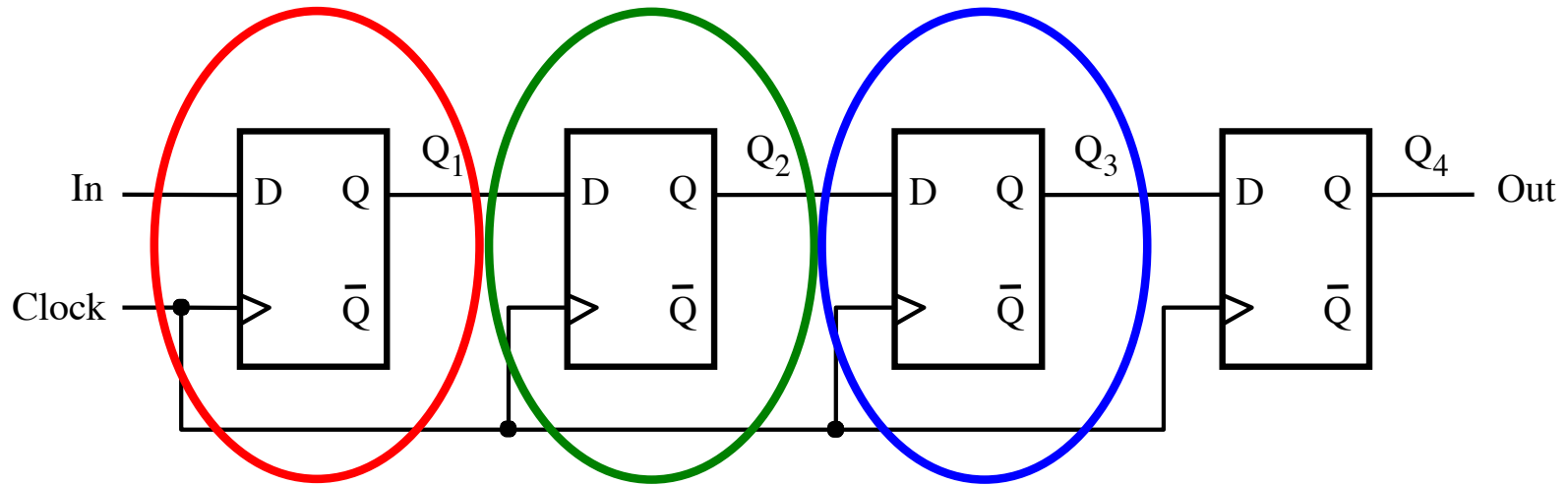
A simple shift register



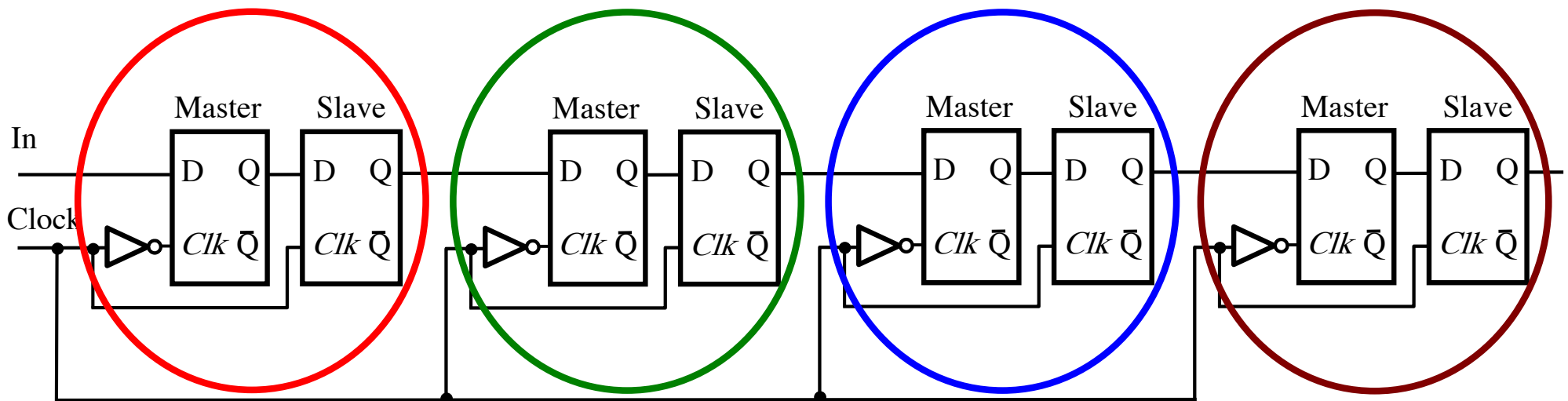
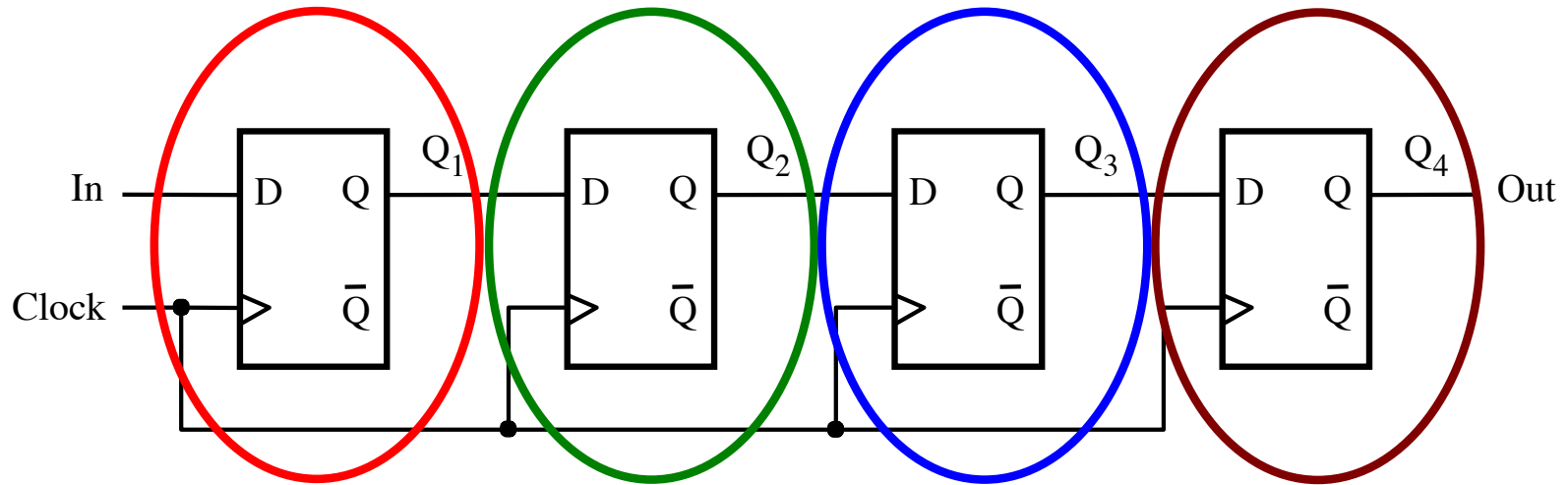
A simple shift register



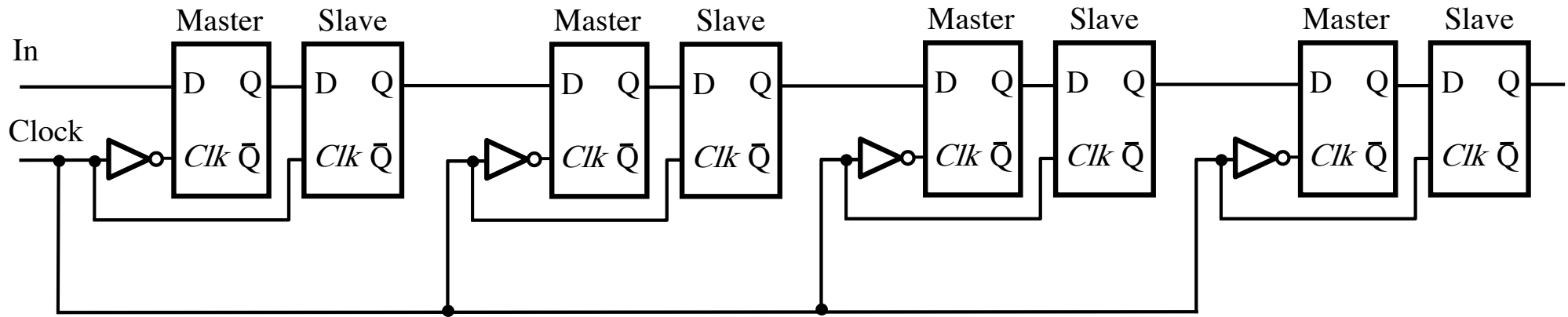
A simple shift register



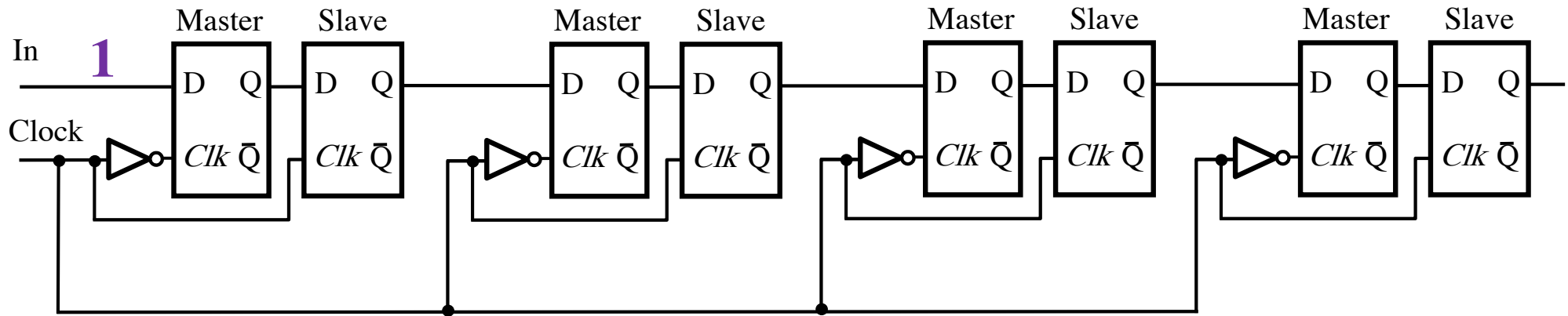
A simple shift register



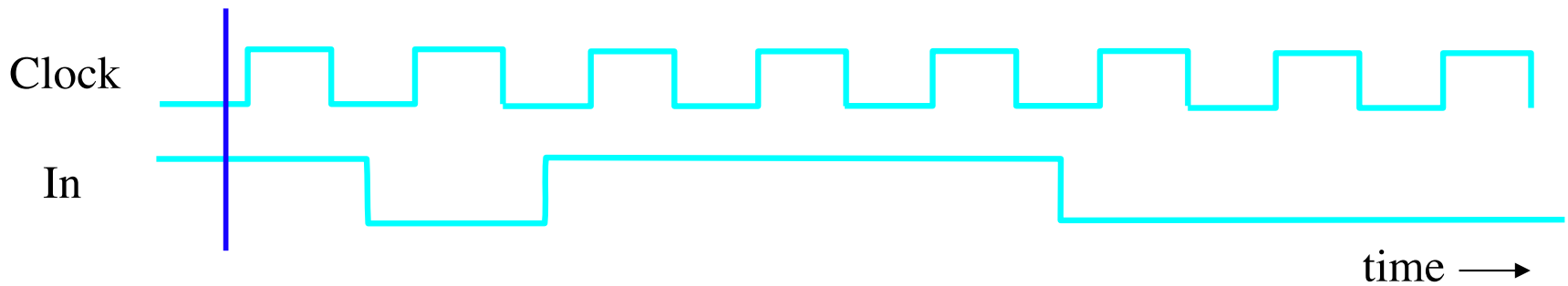
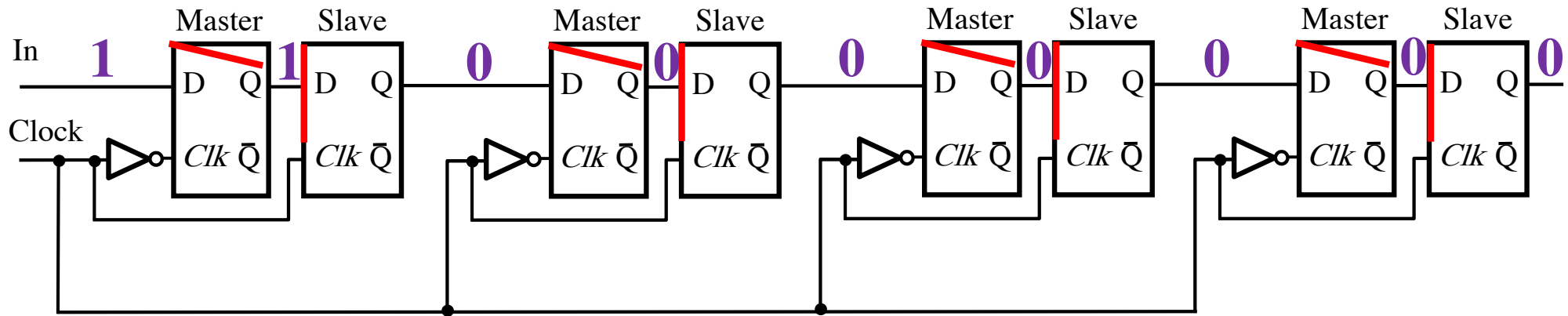
Simulating a shift register



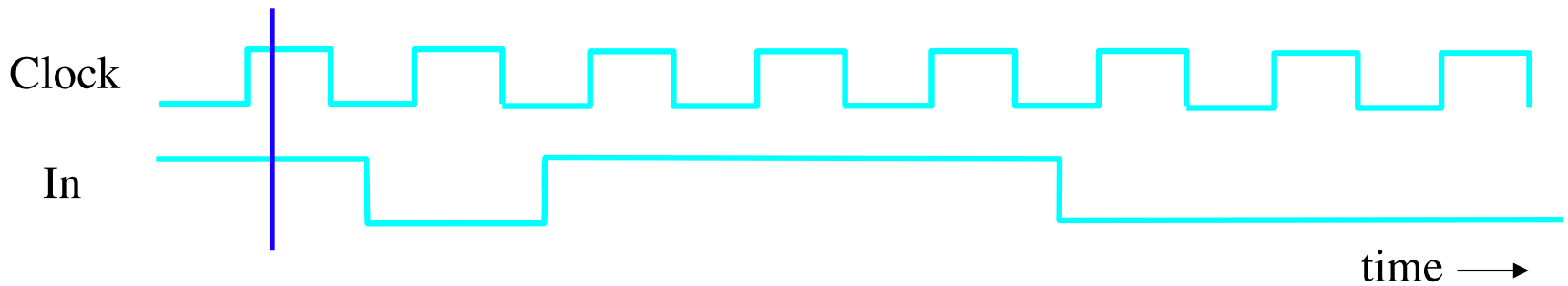
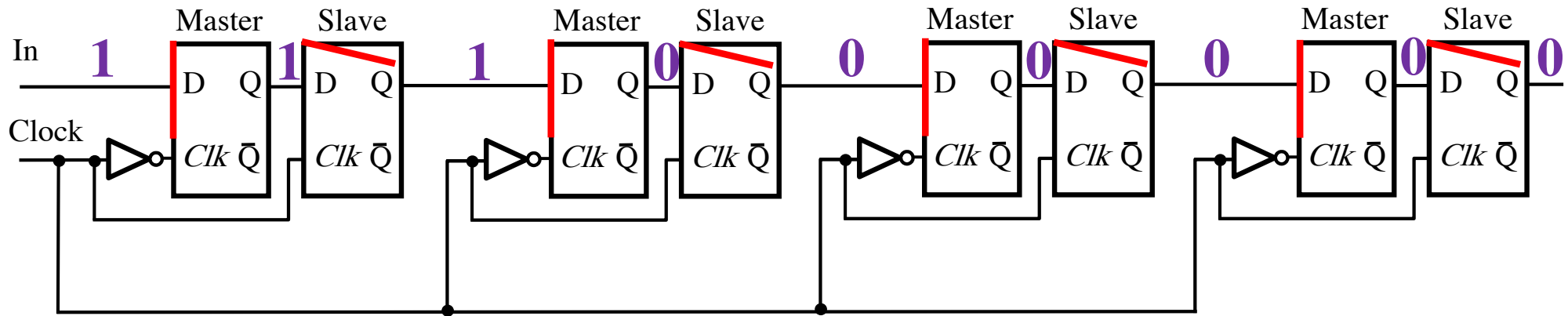
Simulating a shift register



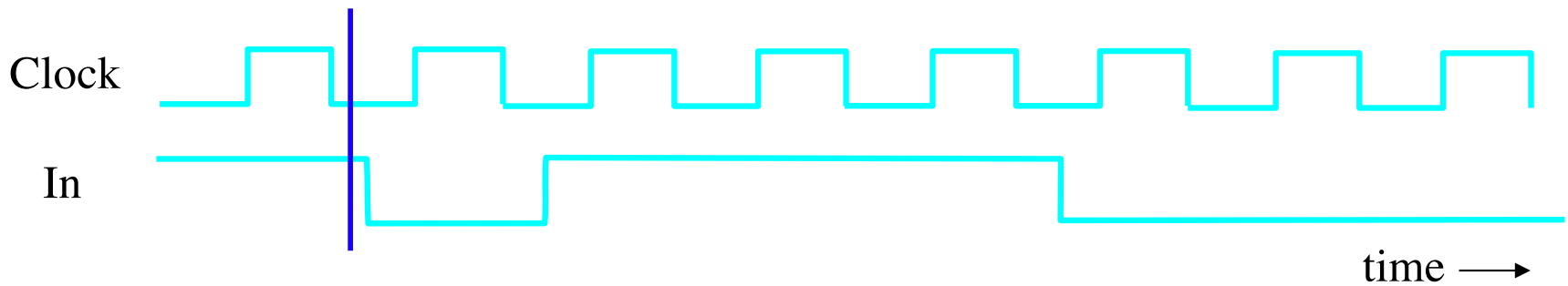
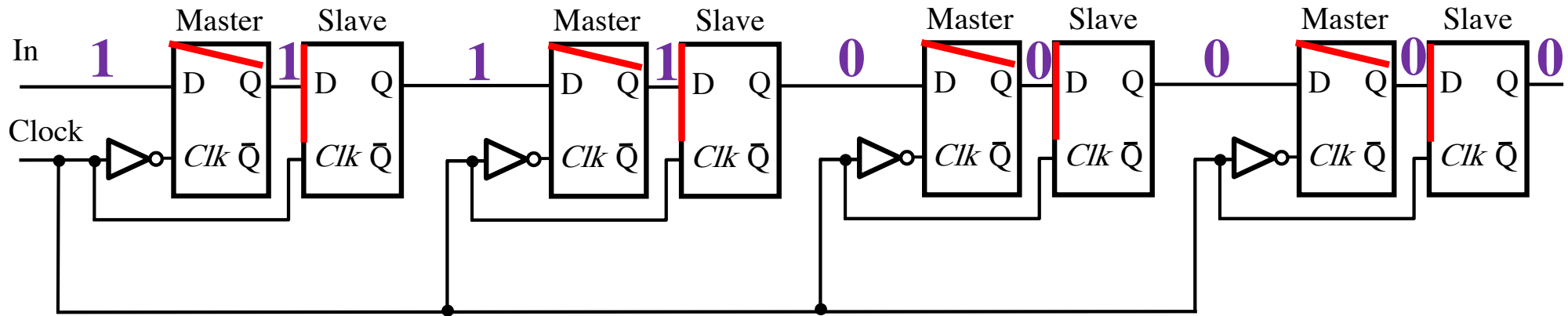
Simulating a shift register



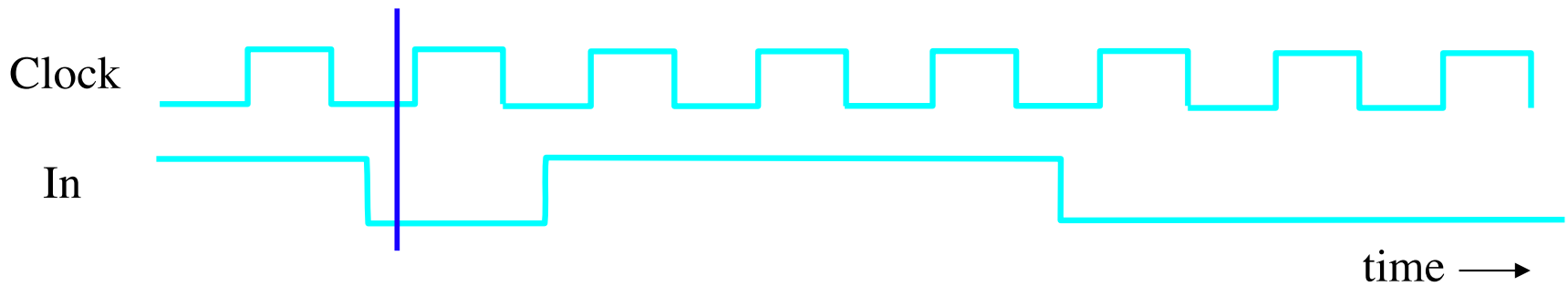
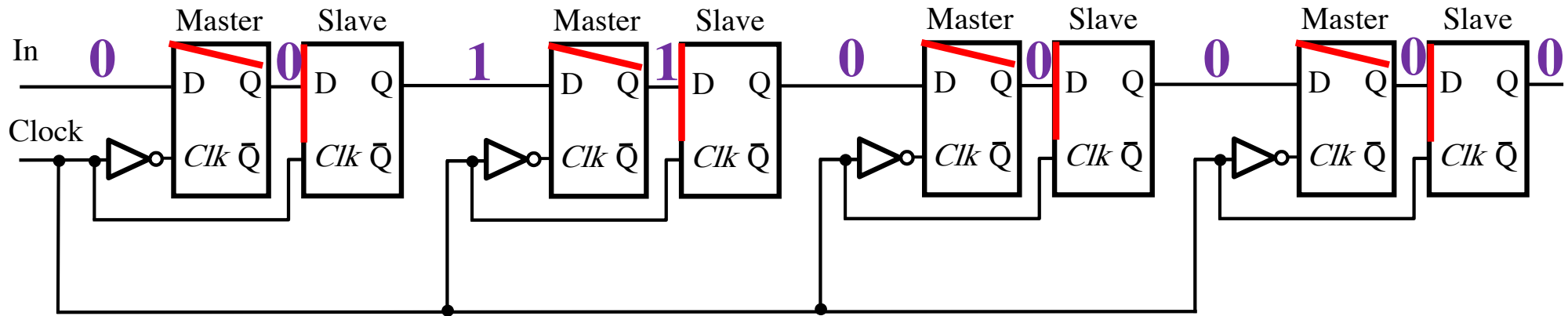
Simulating a shift register



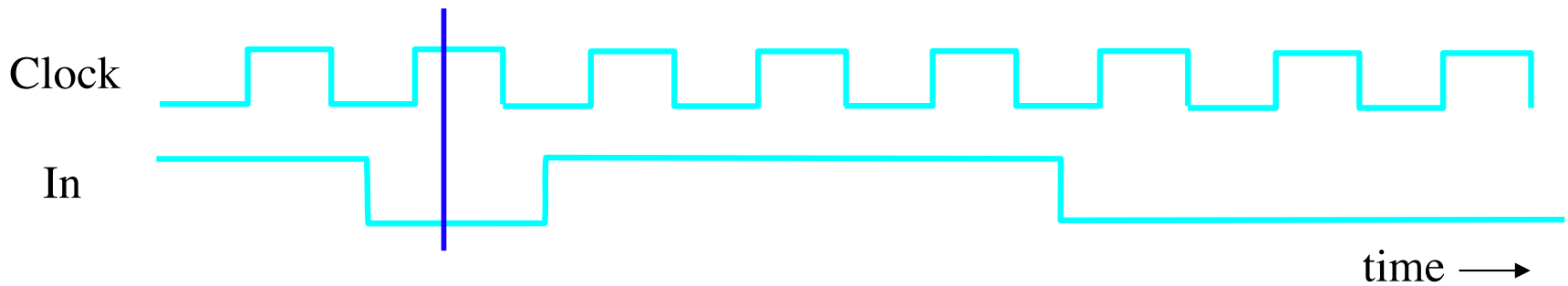
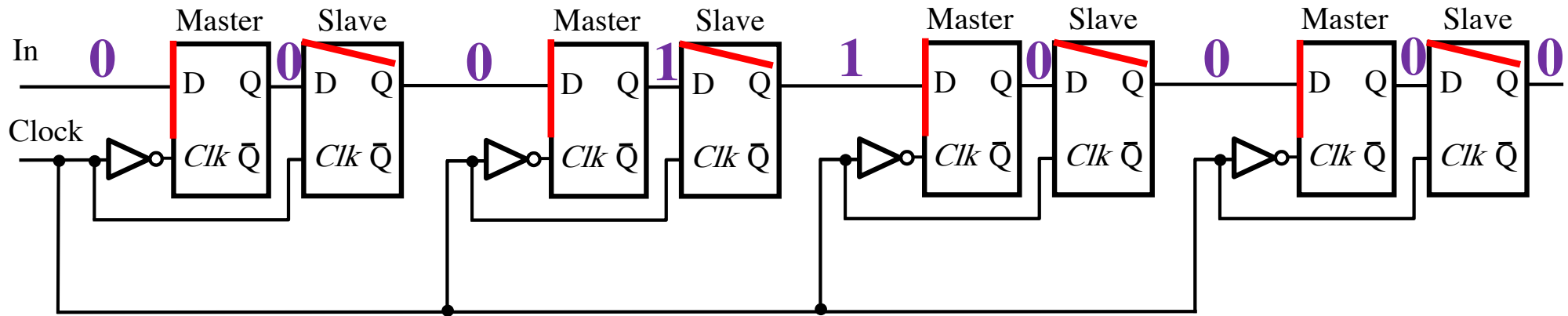
Simulating a shift register



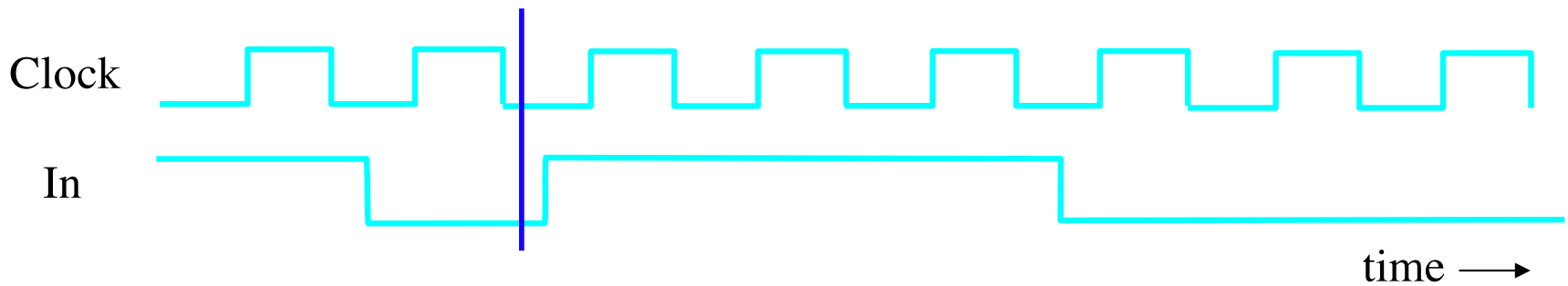
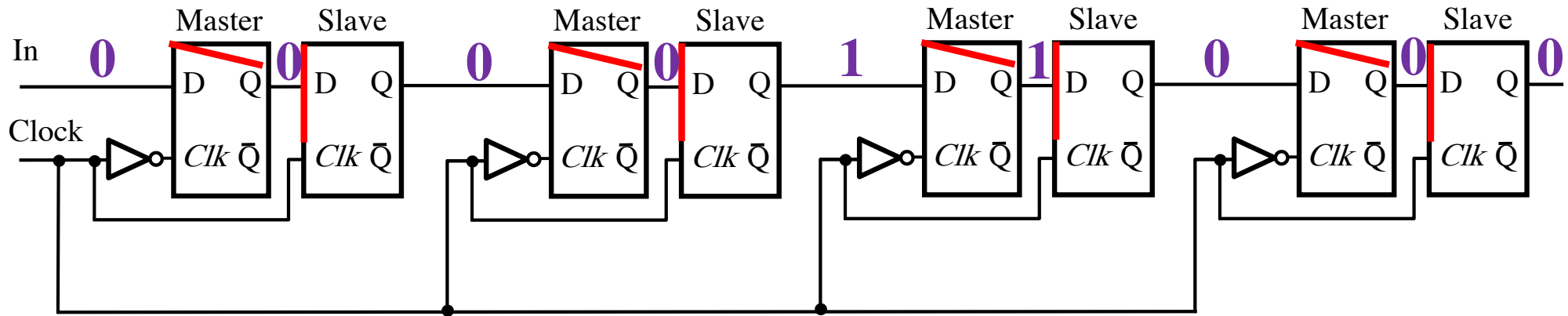
Simulating a shift register



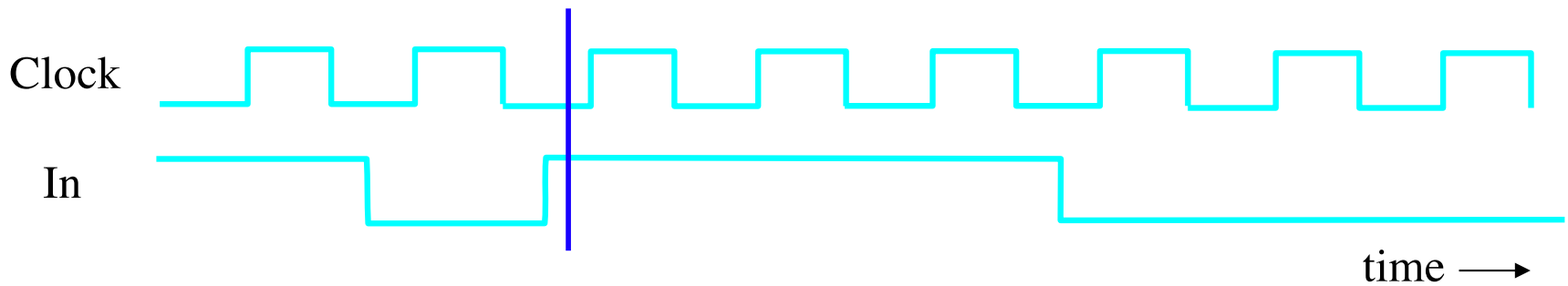
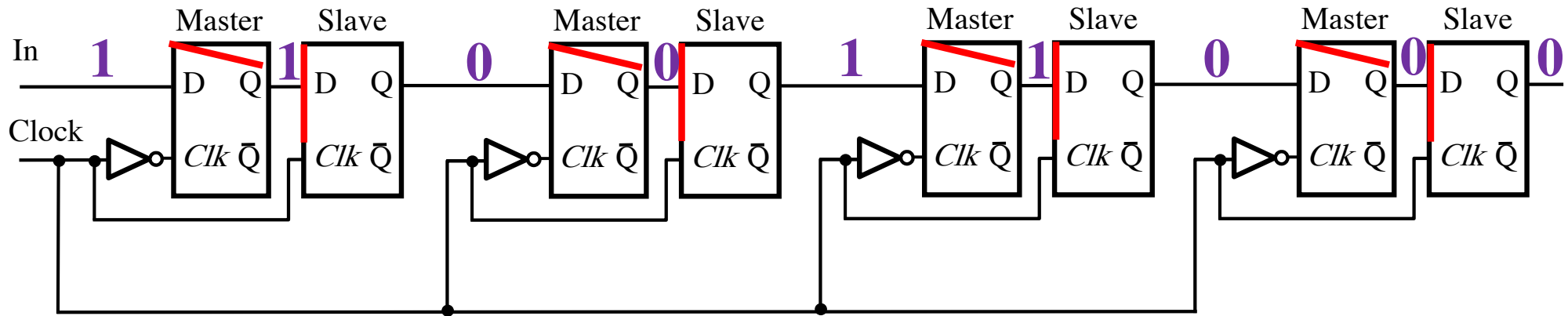
Simulating a shift register



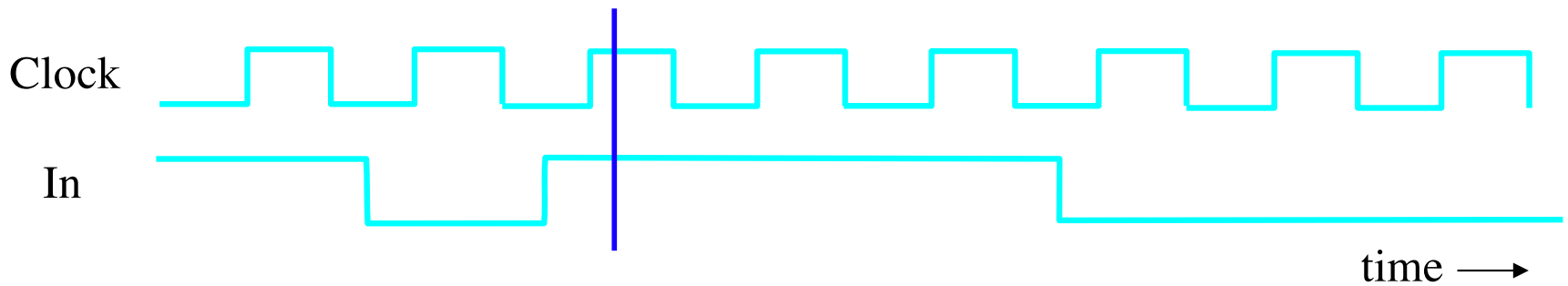
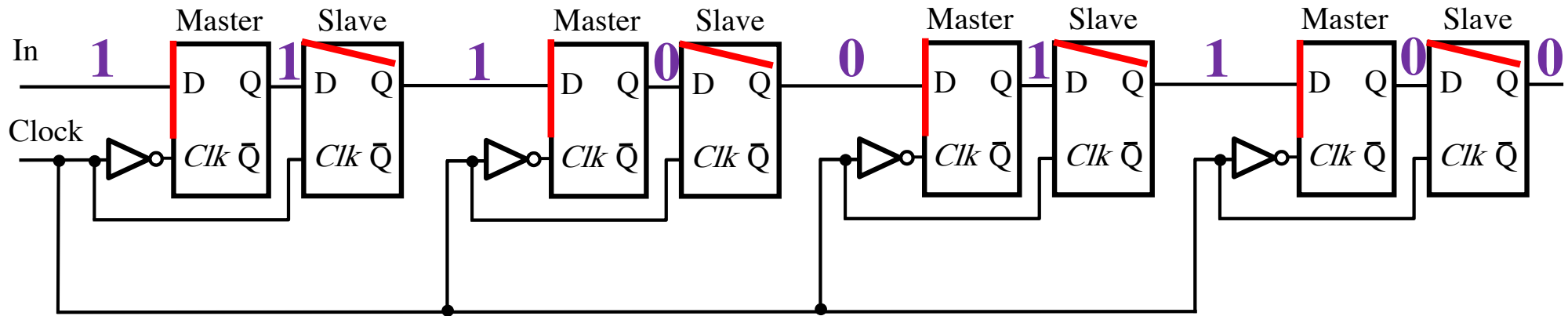
Simulating a shift register



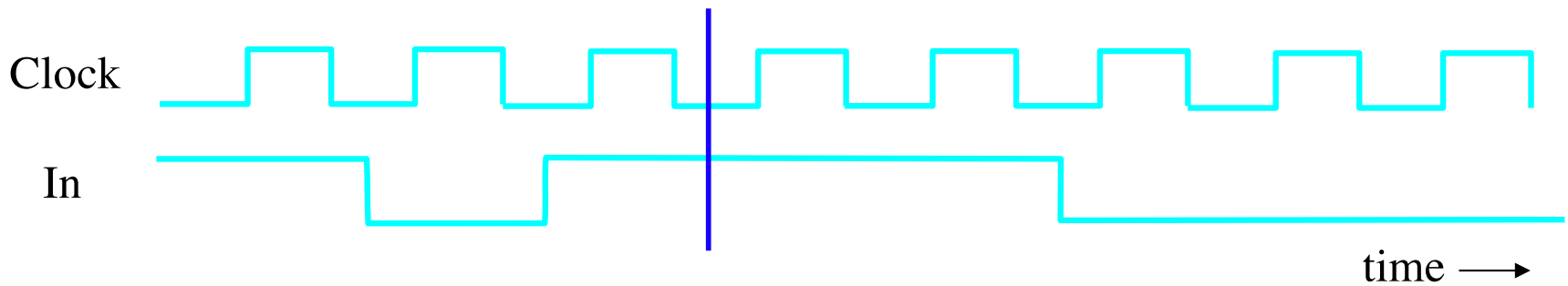
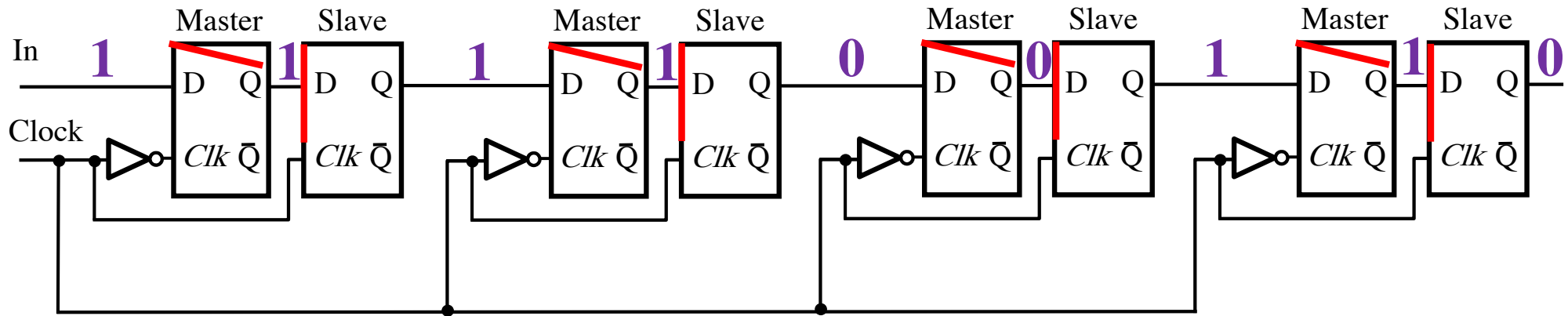
Simulating a shift register



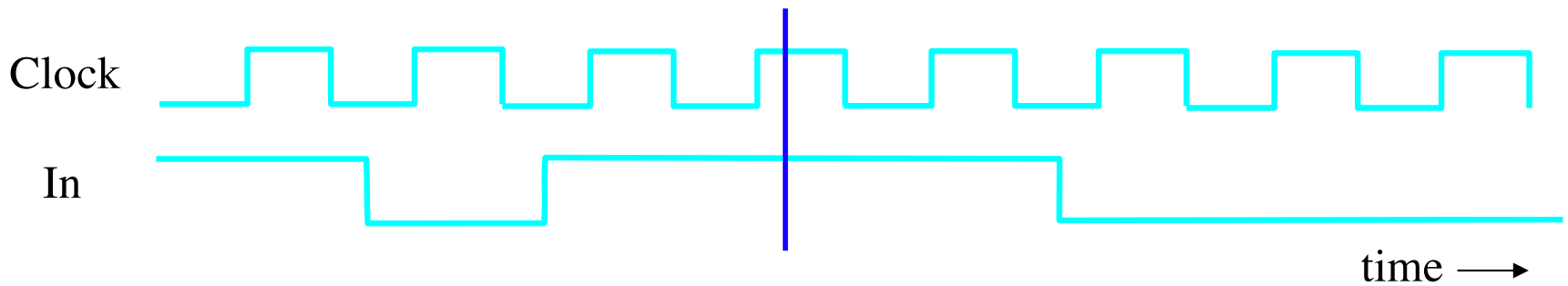
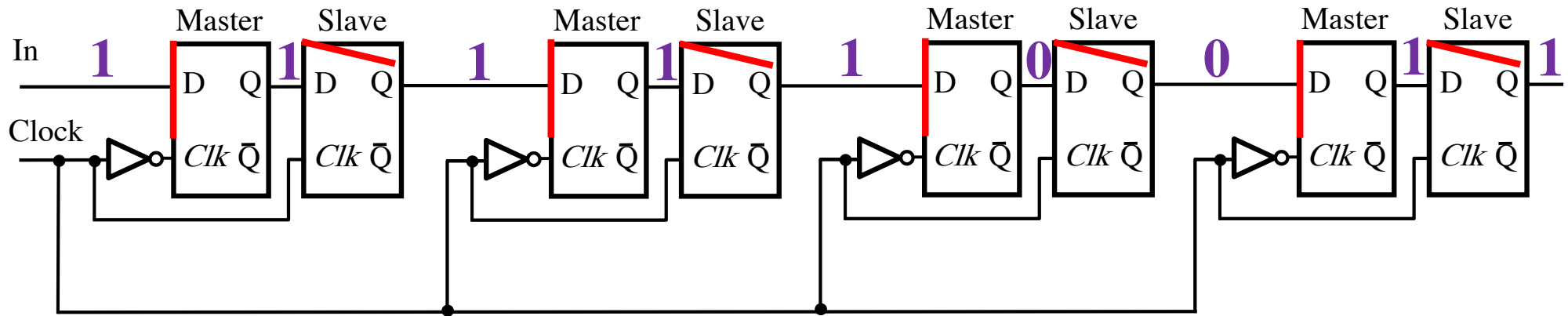
Simulating a shift register



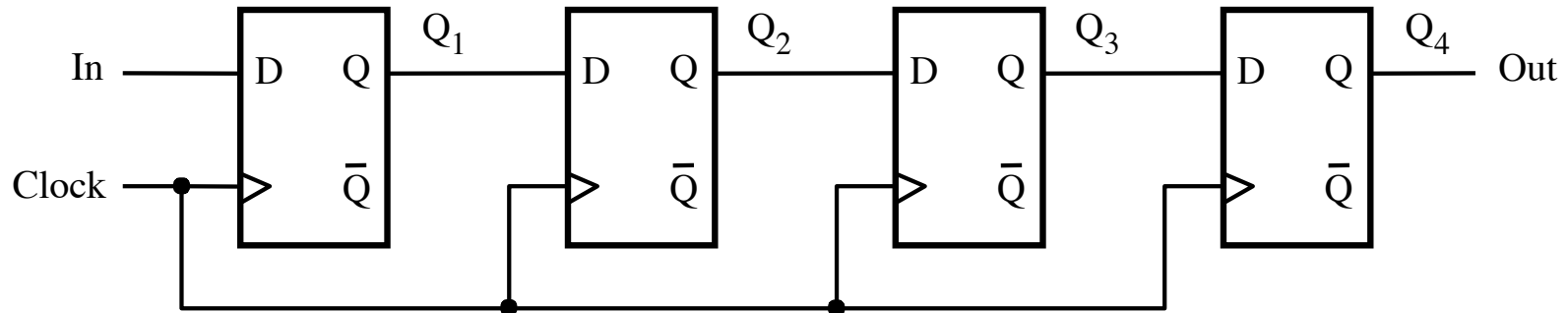
Simulating a shift register



Simulating a shift register



A simple shift register



(a) Circuit

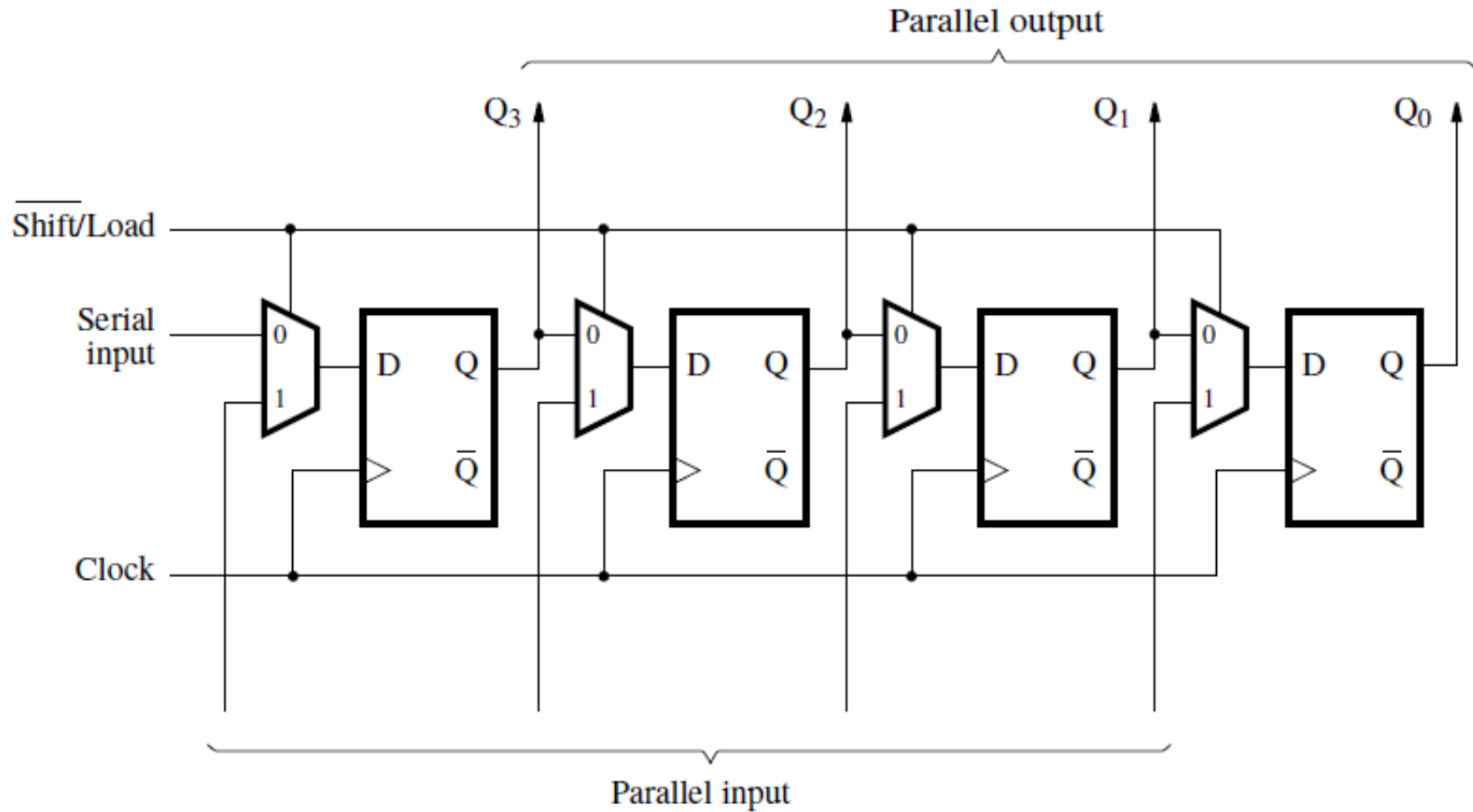
	In	Q ₁	Q ₂	Q ₃	Q ₄ = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

This simulation goes only up to here

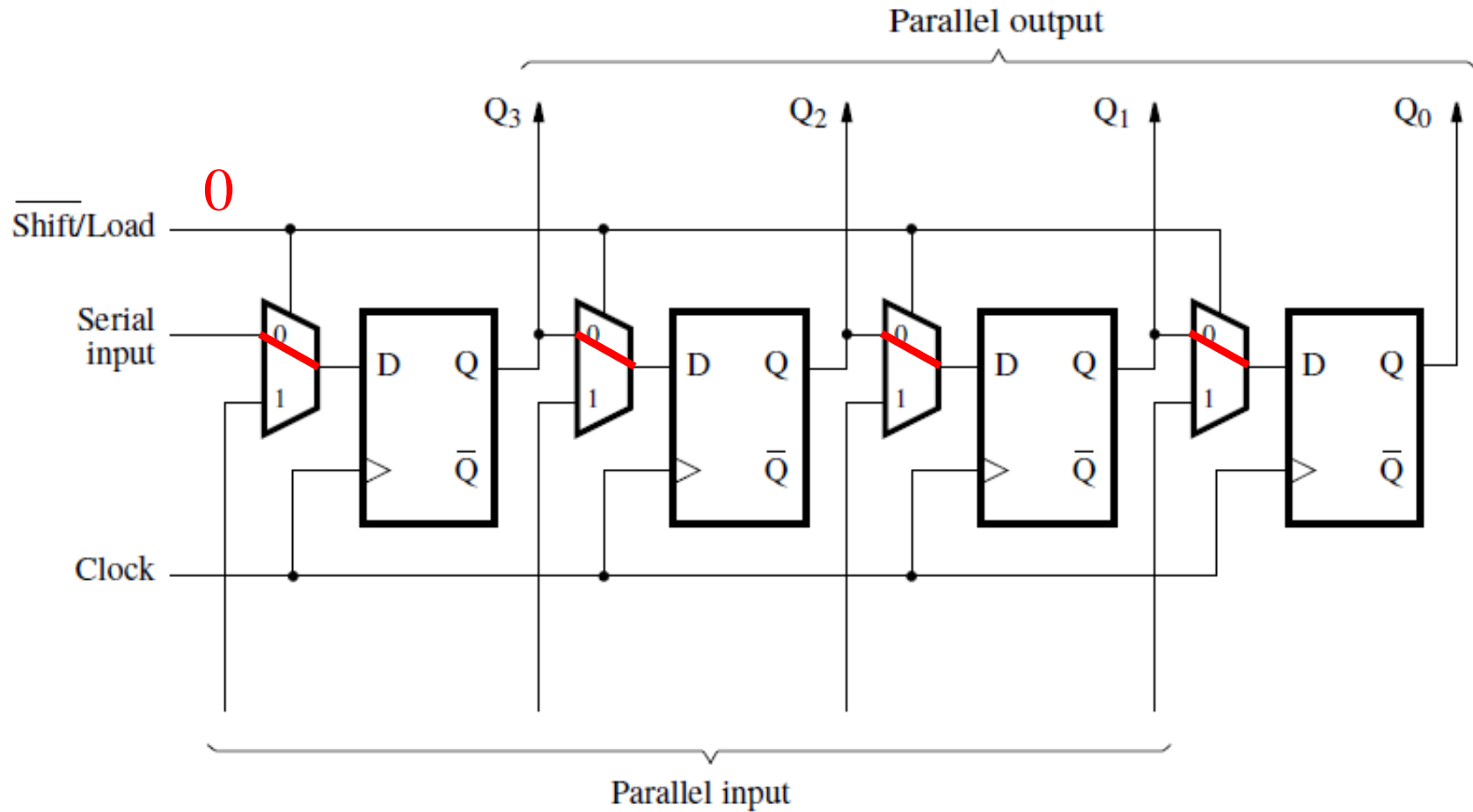
(b) A sample sequence

Parallel-Access Shift Register

Parallel-access shift register

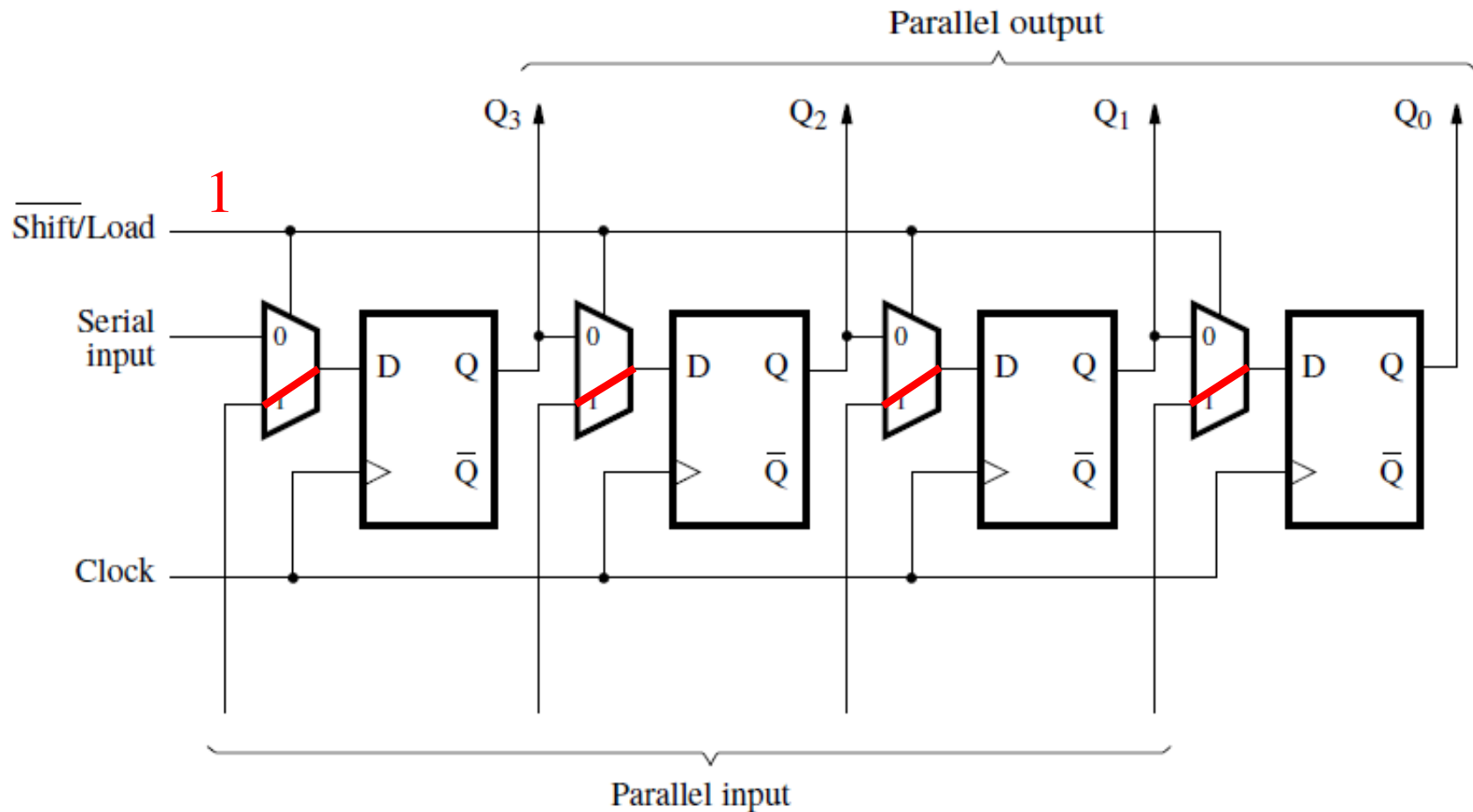


Parallel-access shift register



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

Parallel-access shift register



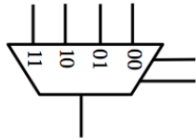
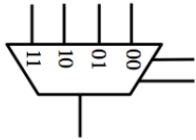
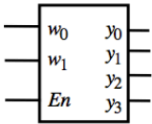
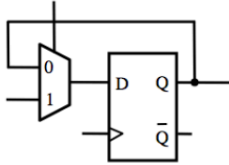
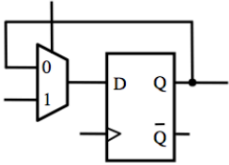
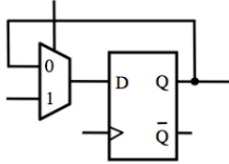
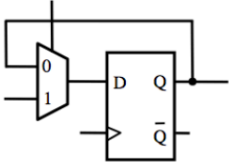
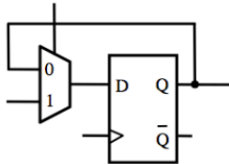
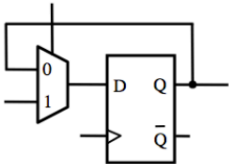
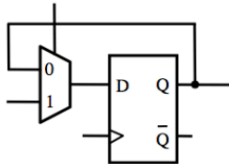
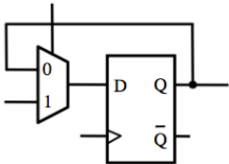
When Load=1, this behaves like a parallel-access register.

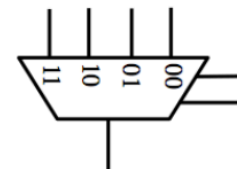
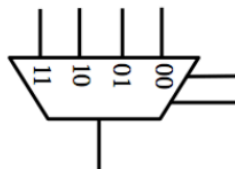
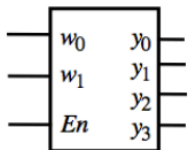
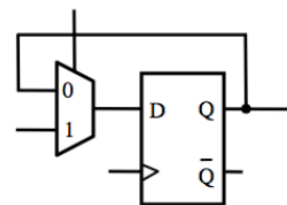
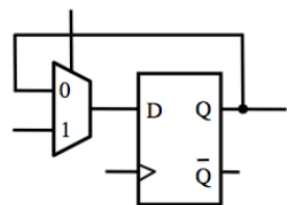
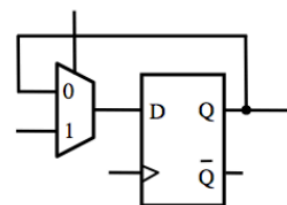
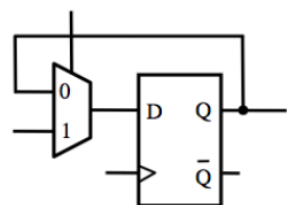
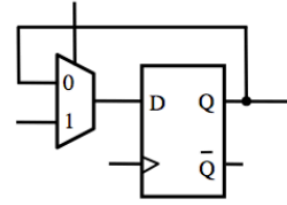
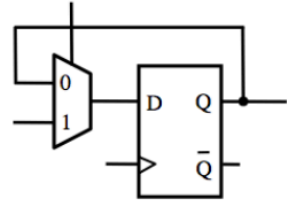
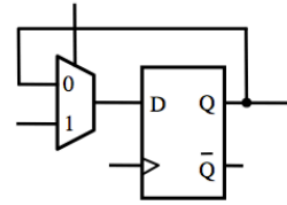
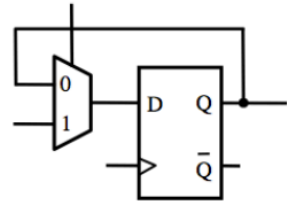
Register File

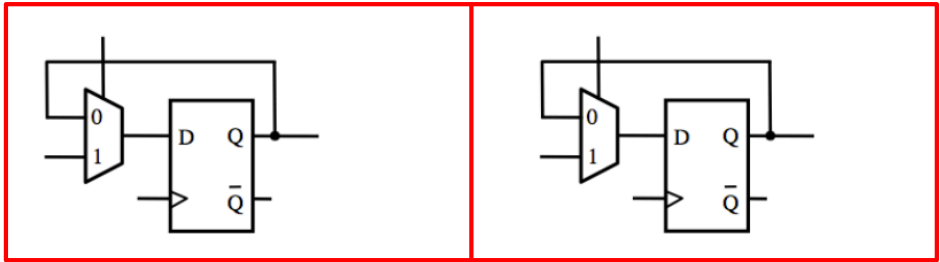
Motivation

- **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.**

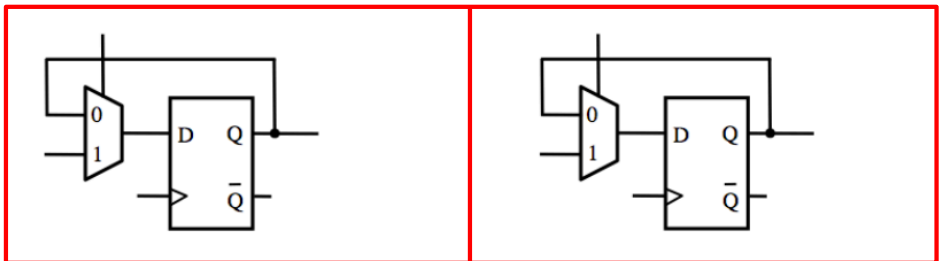
Complete the following circuit diagram to implement a register file with four 2-bit registers, one write port, one read port, and one write enable line.



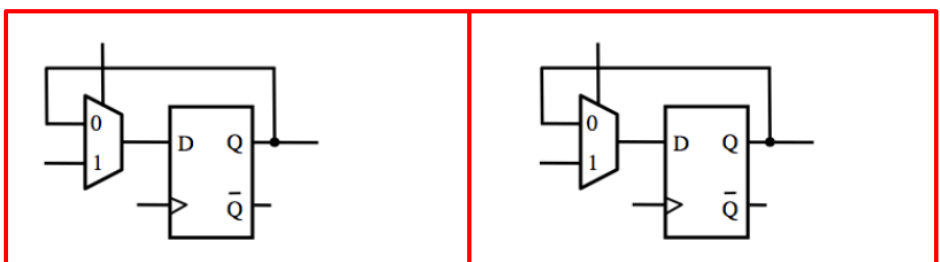




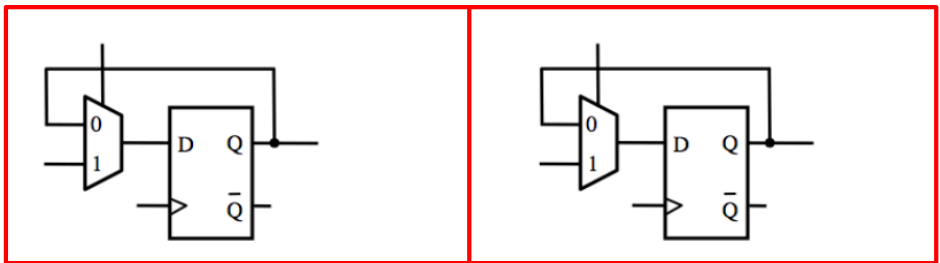
Register 0



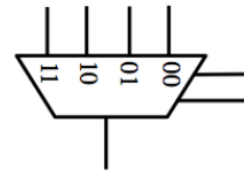
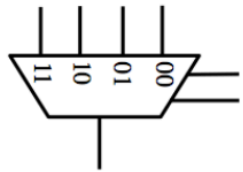
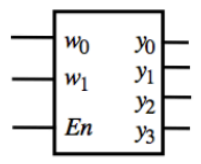
Register 1

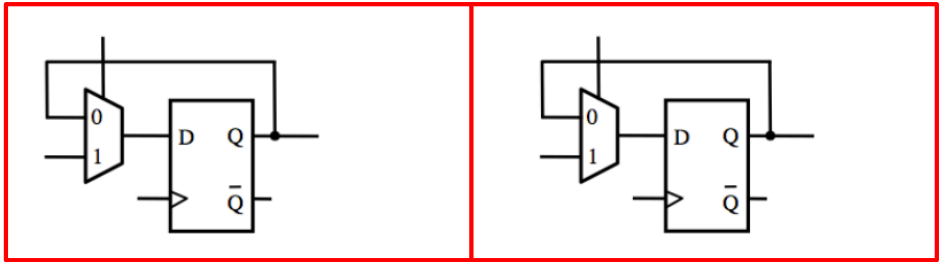


Register 2

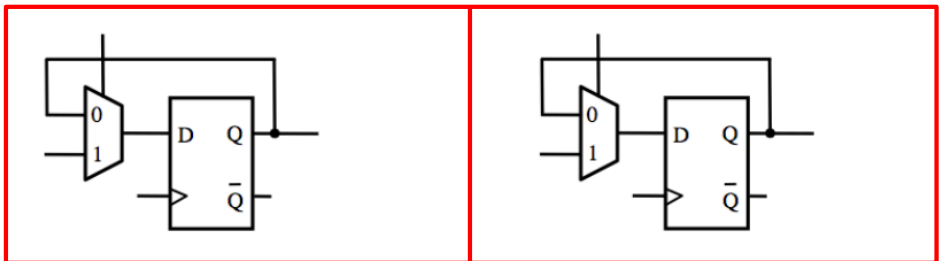


Register 3

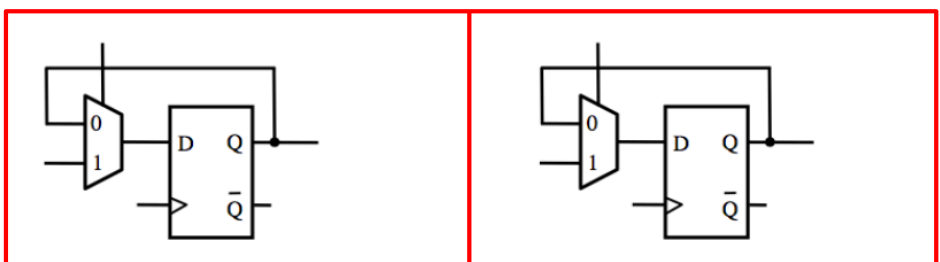




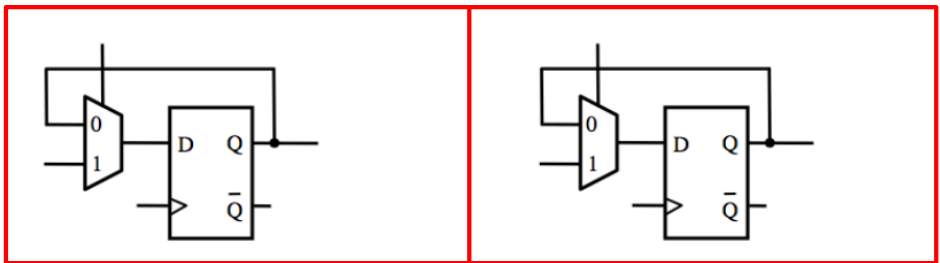
Register A



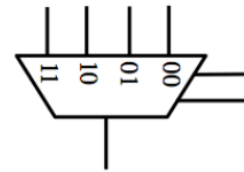
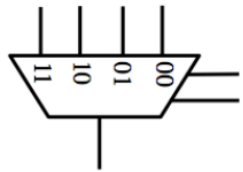
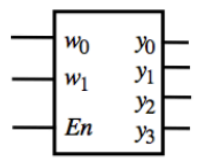
Register B

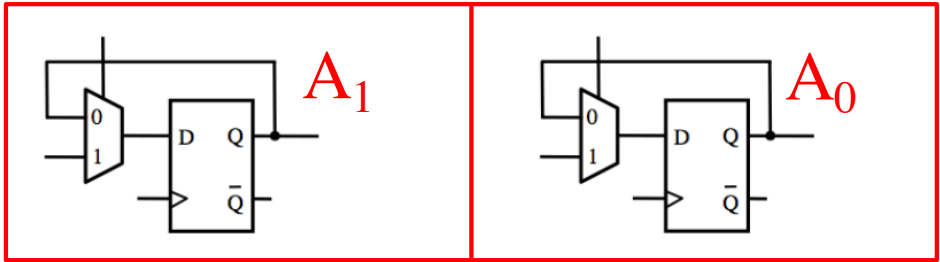


Register C

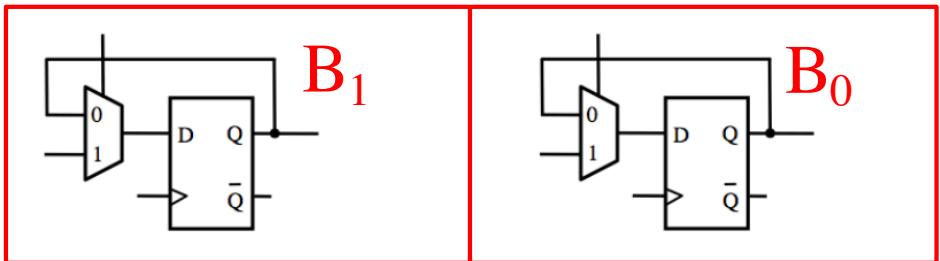


Register D

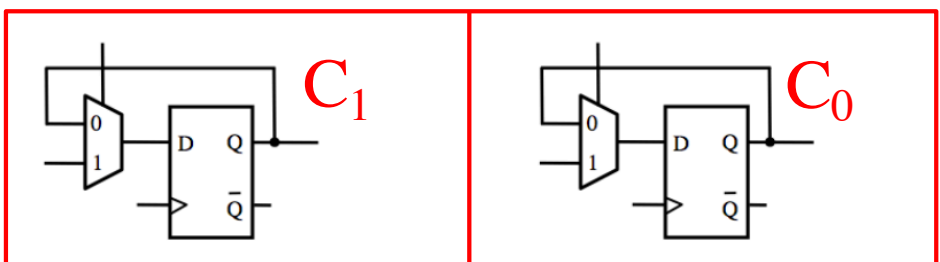




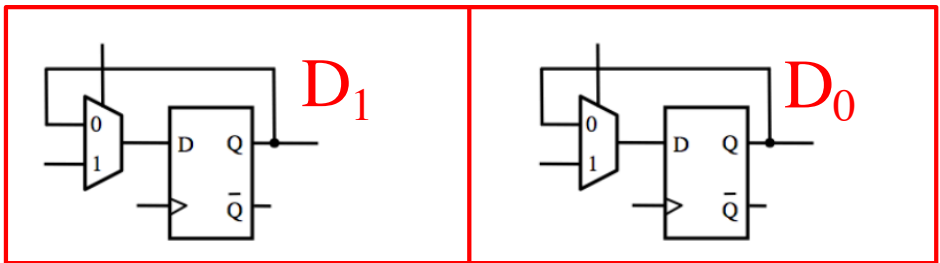
Register A



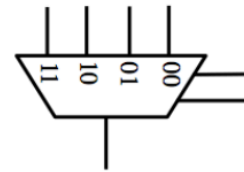
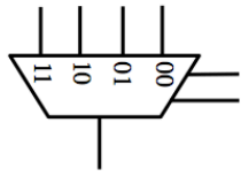
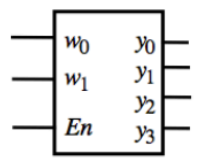
Register B

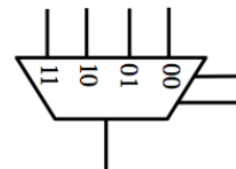
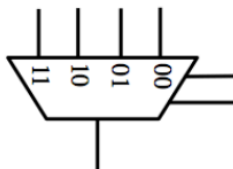
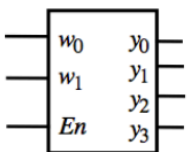
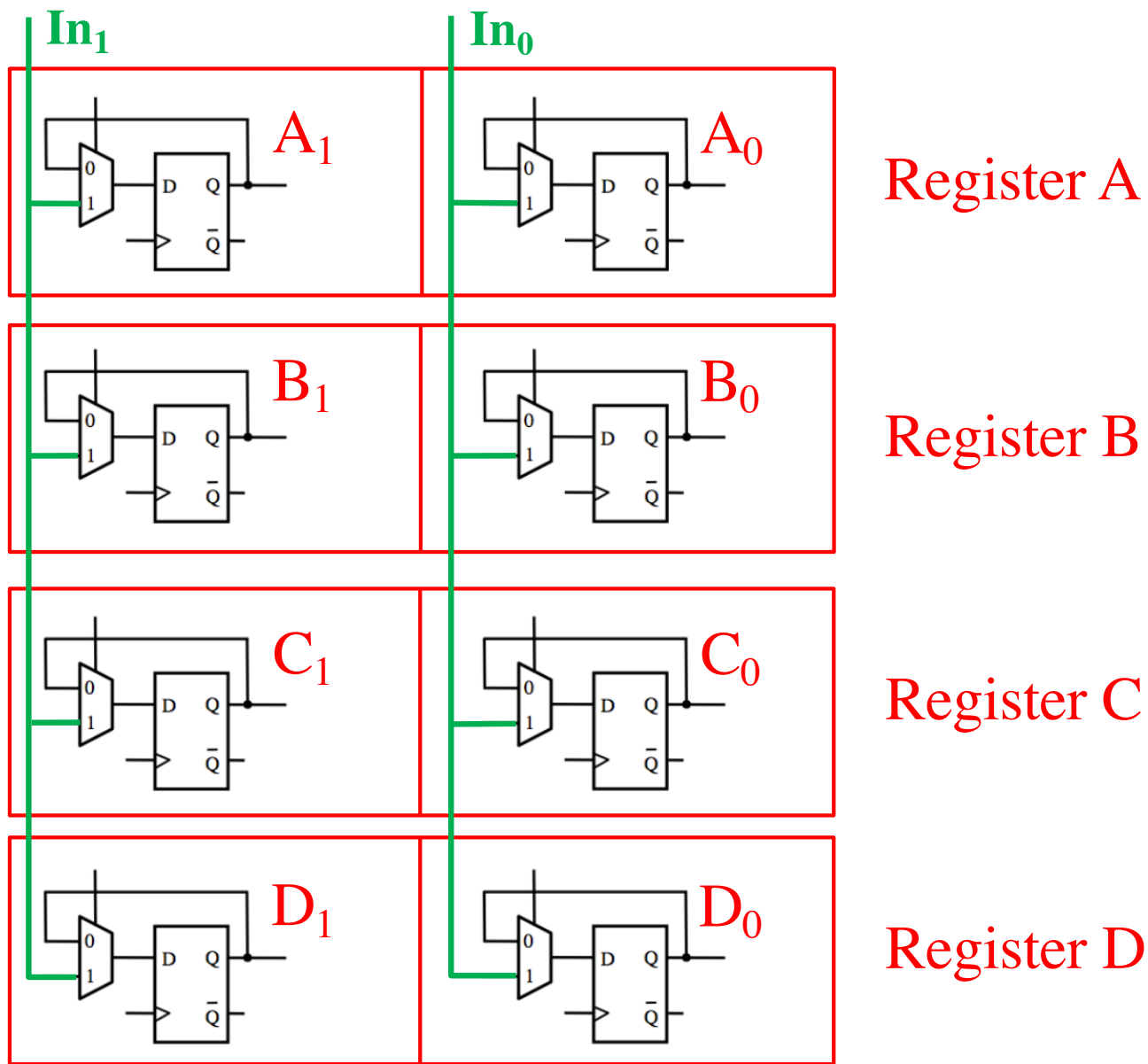


Register C



Register D





In_1

In_0

A_1

A_0

Register A

B_1

B_0

Register B

C_1

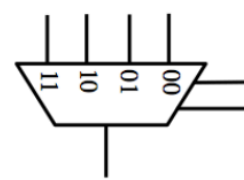
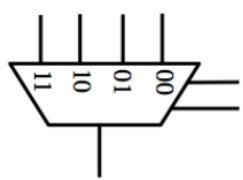
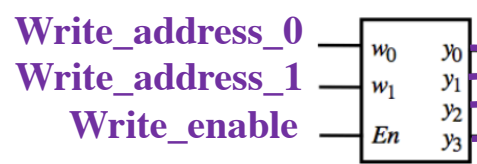
C_0

Register C

D_1

D_0

Register D



In₁

In₀

A₁

A₀

Register A

B₁

B₀

Register B

C₁

C₀

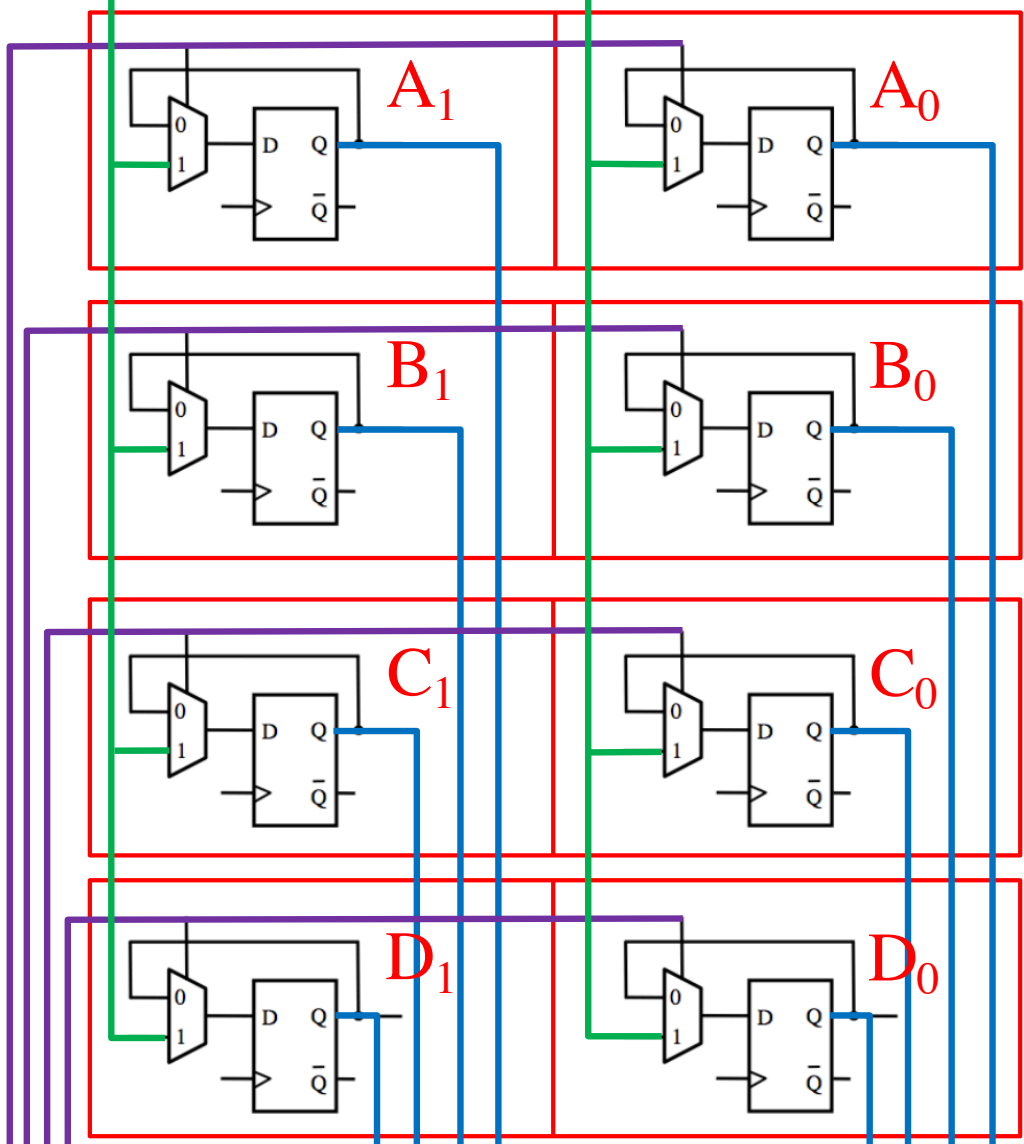
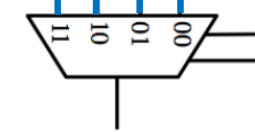
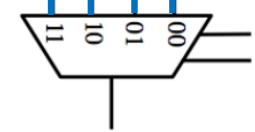
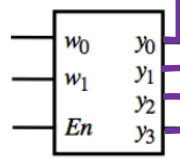
Register C

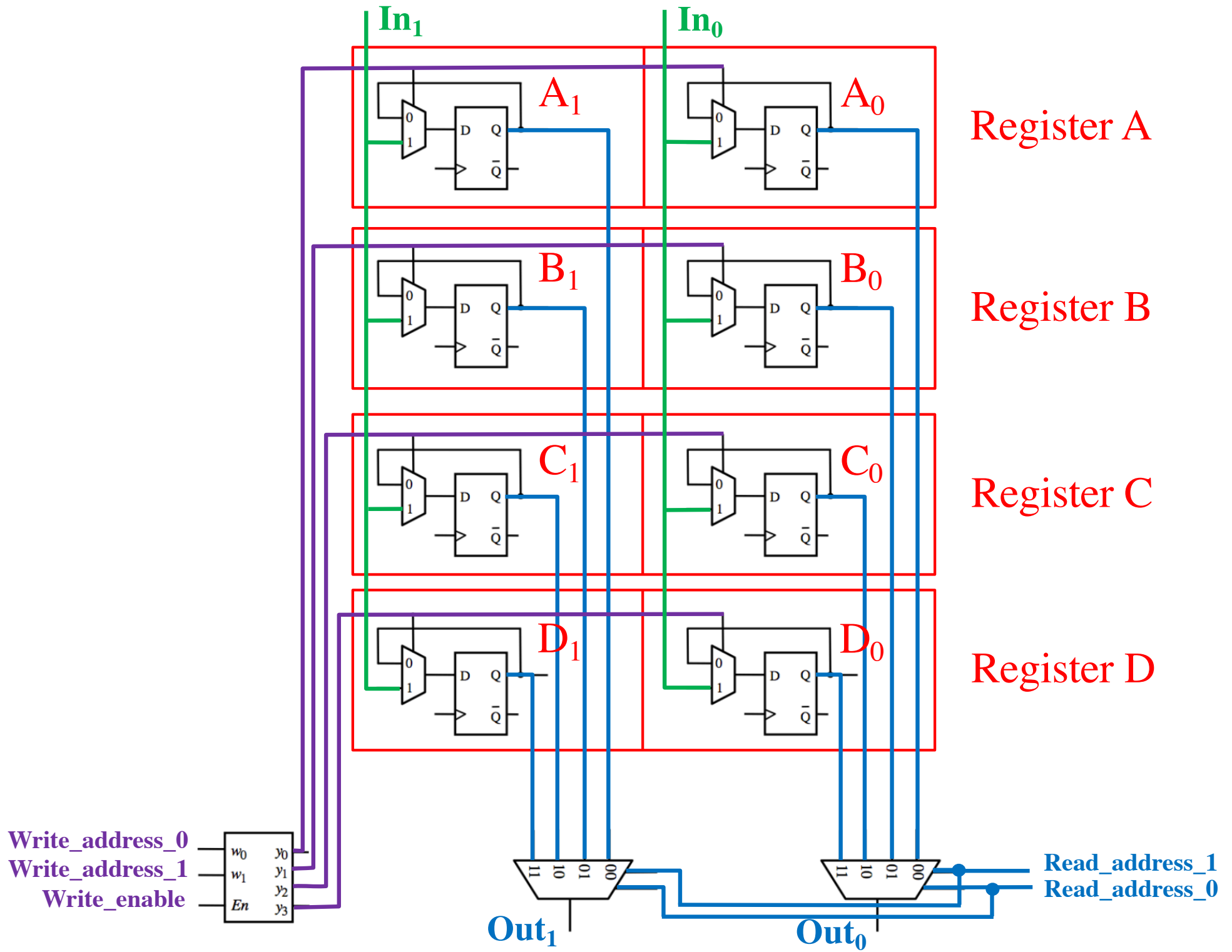
D₁

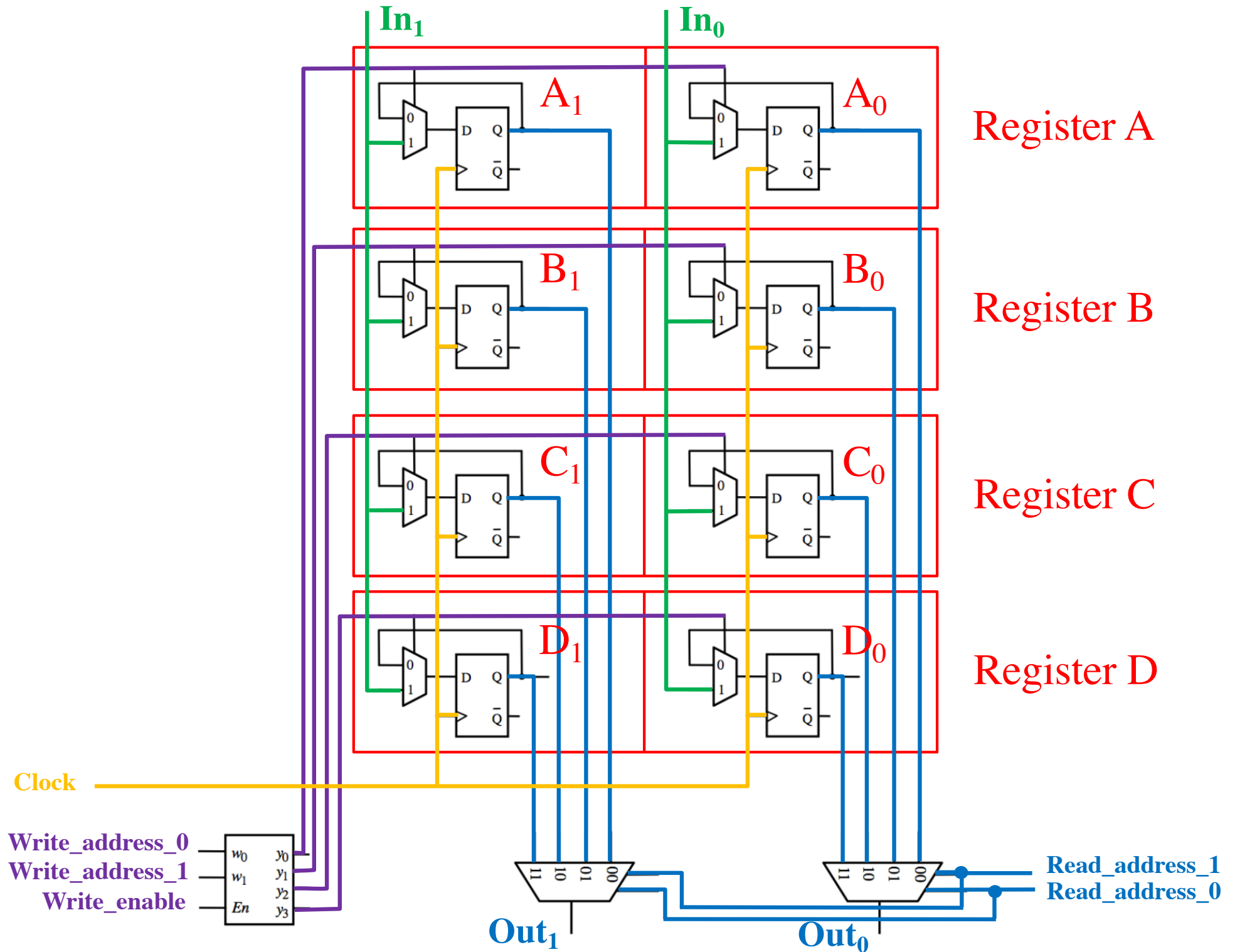
D₀

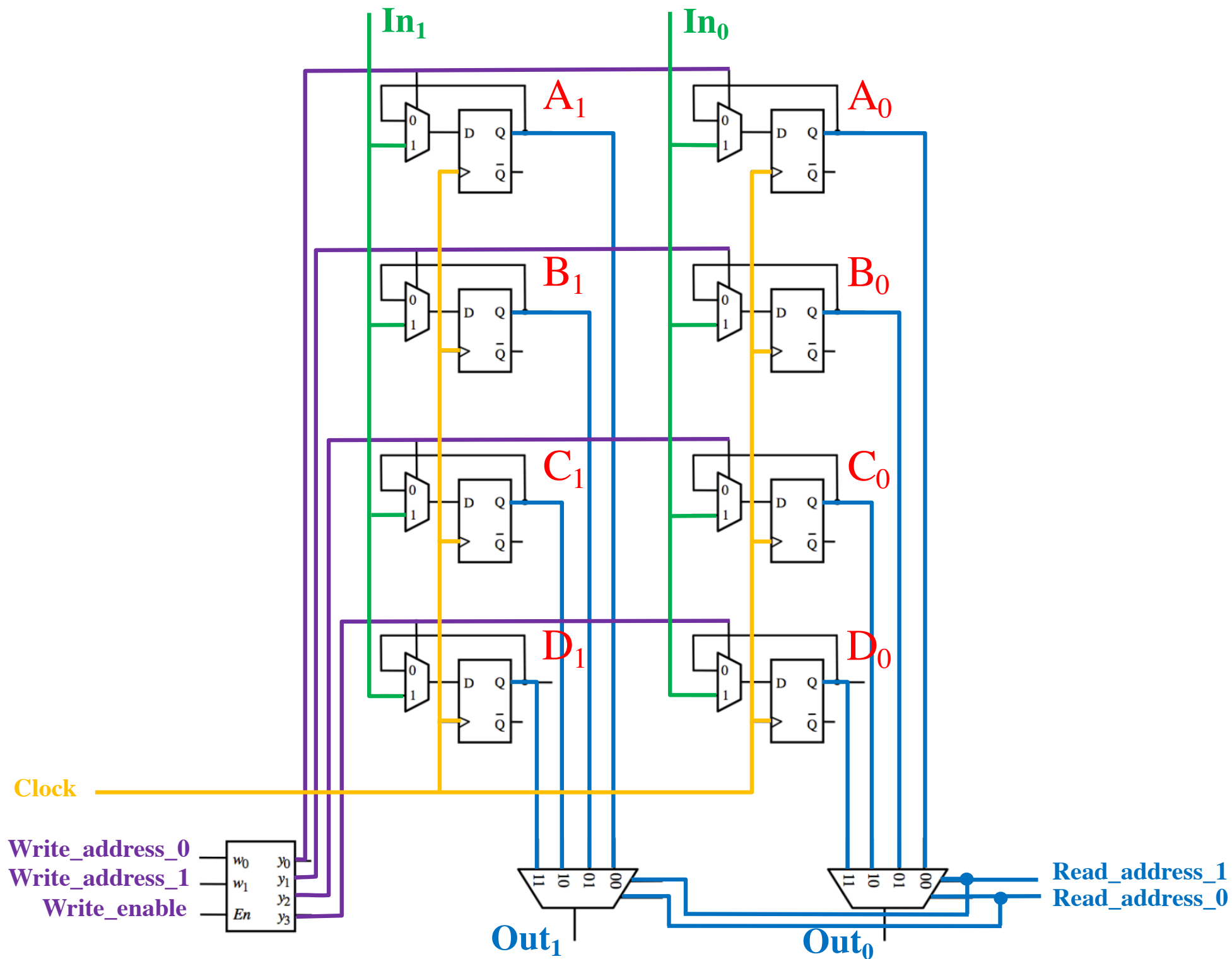
Register D

Write_address_0
Write_address_1
Write_enable









Components of the Register File

2-to-4 Decoder

In_1

In_0

A_1

A_0

B_1

B_0

C_1

C_0

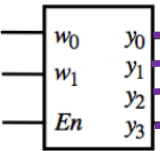
D_1

D_0

2-to-4 decoder
with enable

Clock

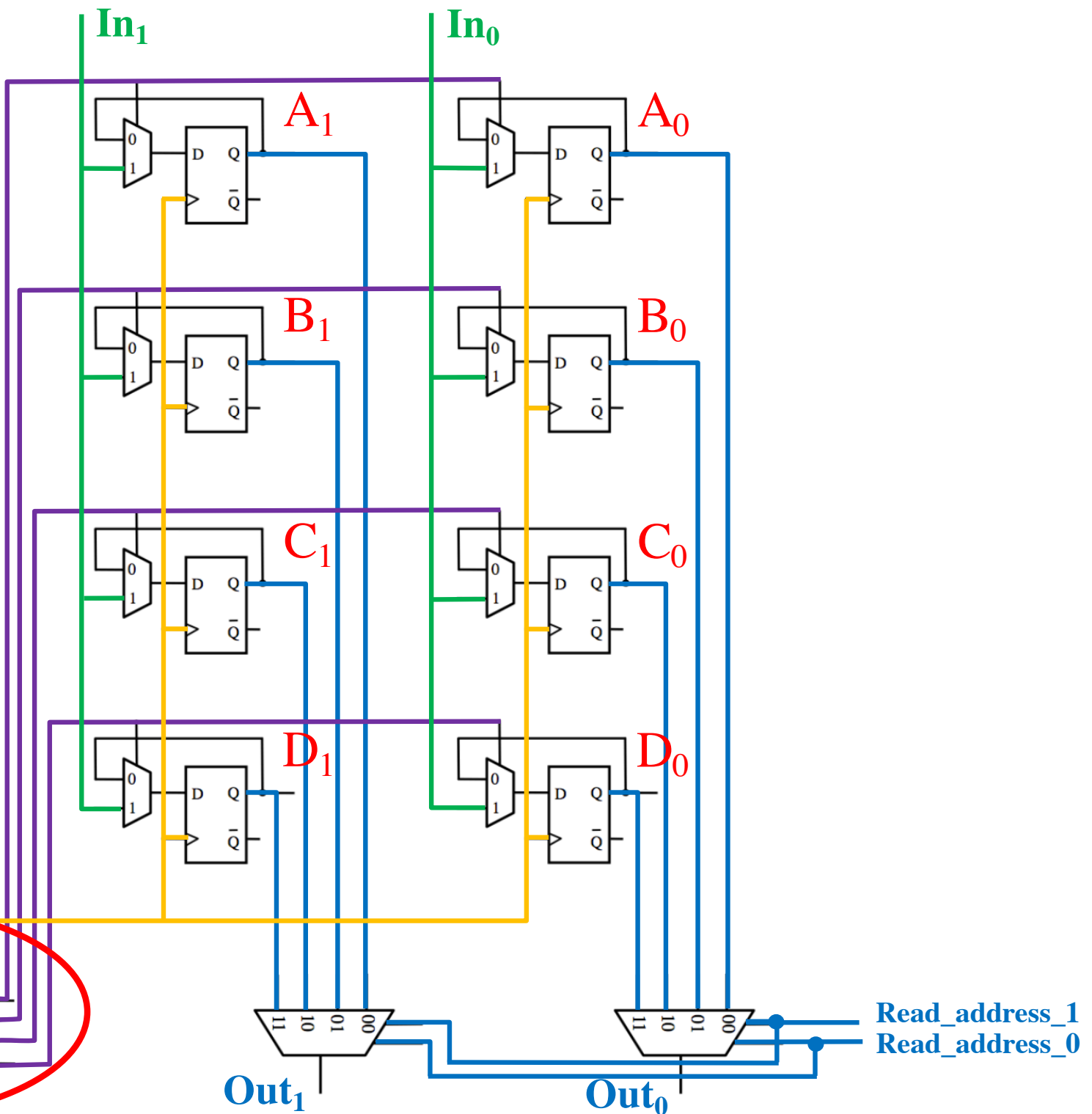
Write_address_0
Write_address_1
Write_enable



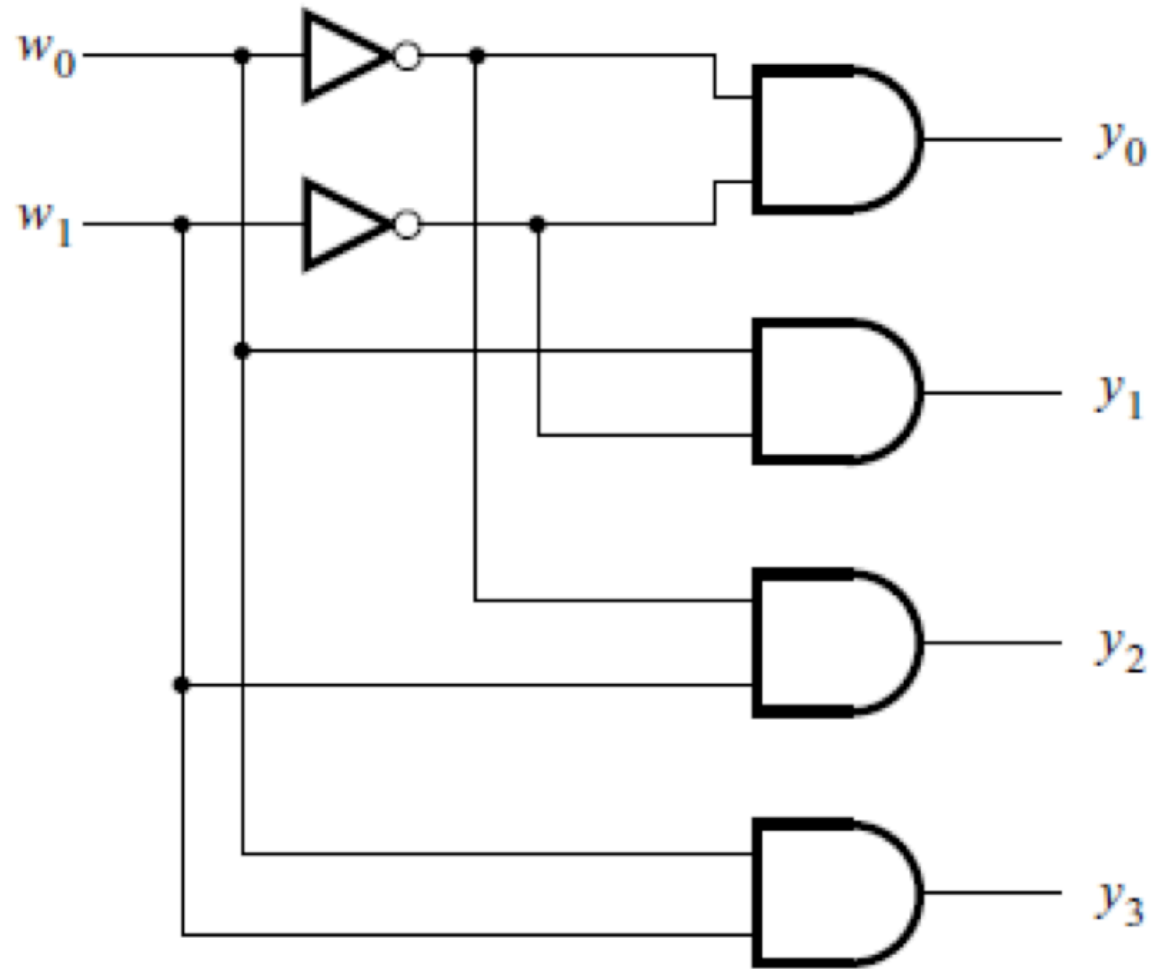
Out_1

Out_0

Read_address_1
Read_address_0

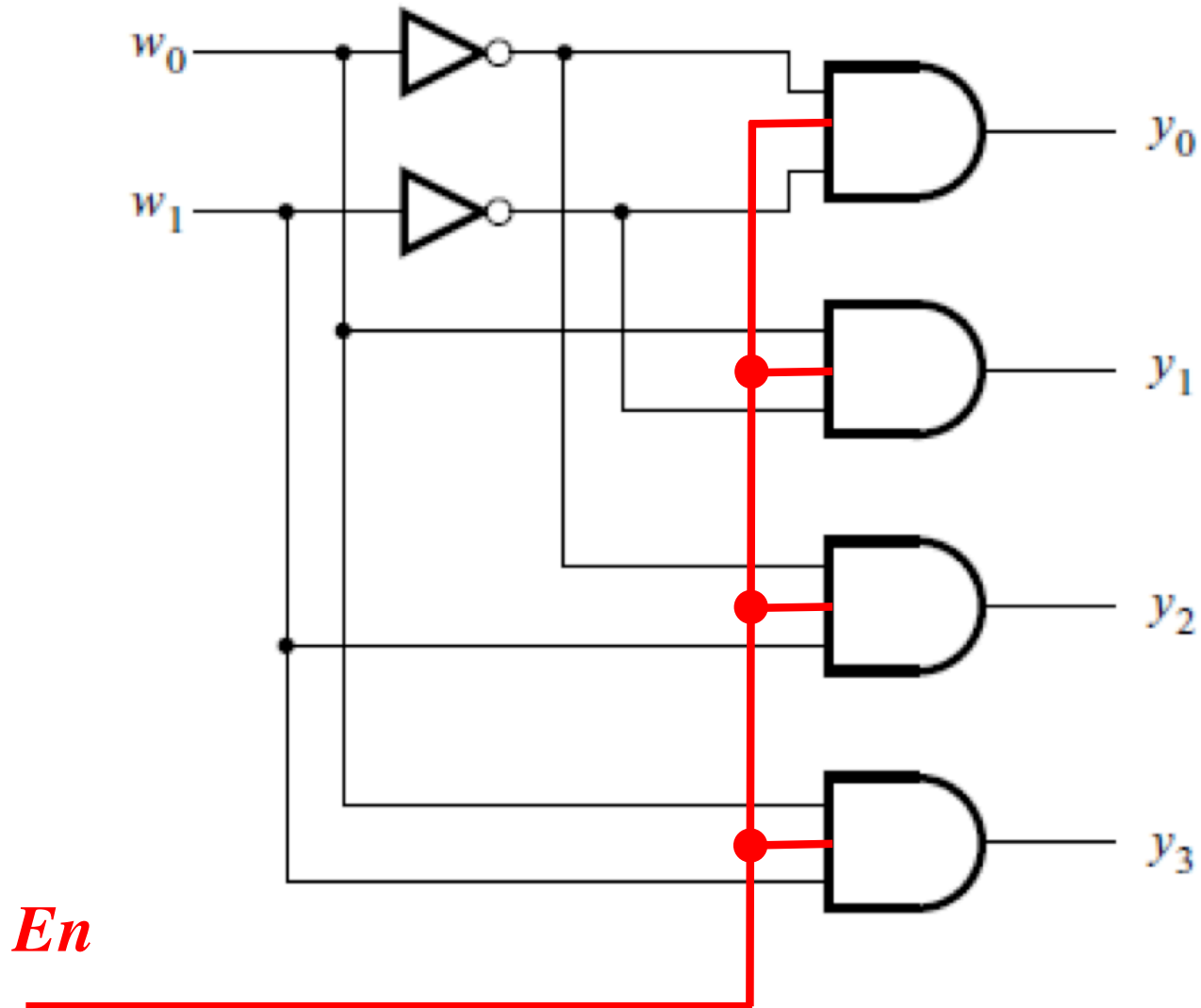


2-to-4 Decoder

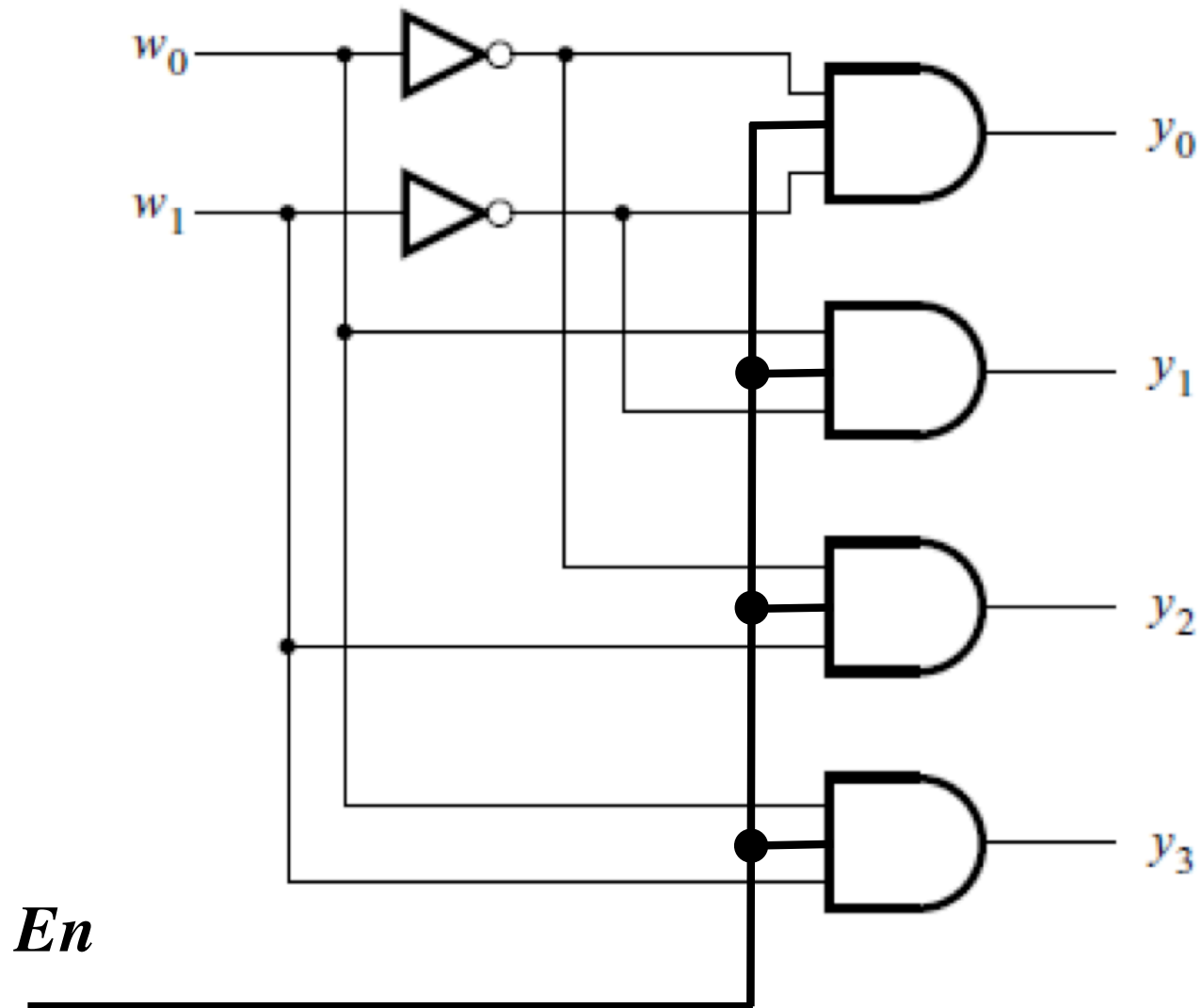


[Figure 4.14c from the textbook]

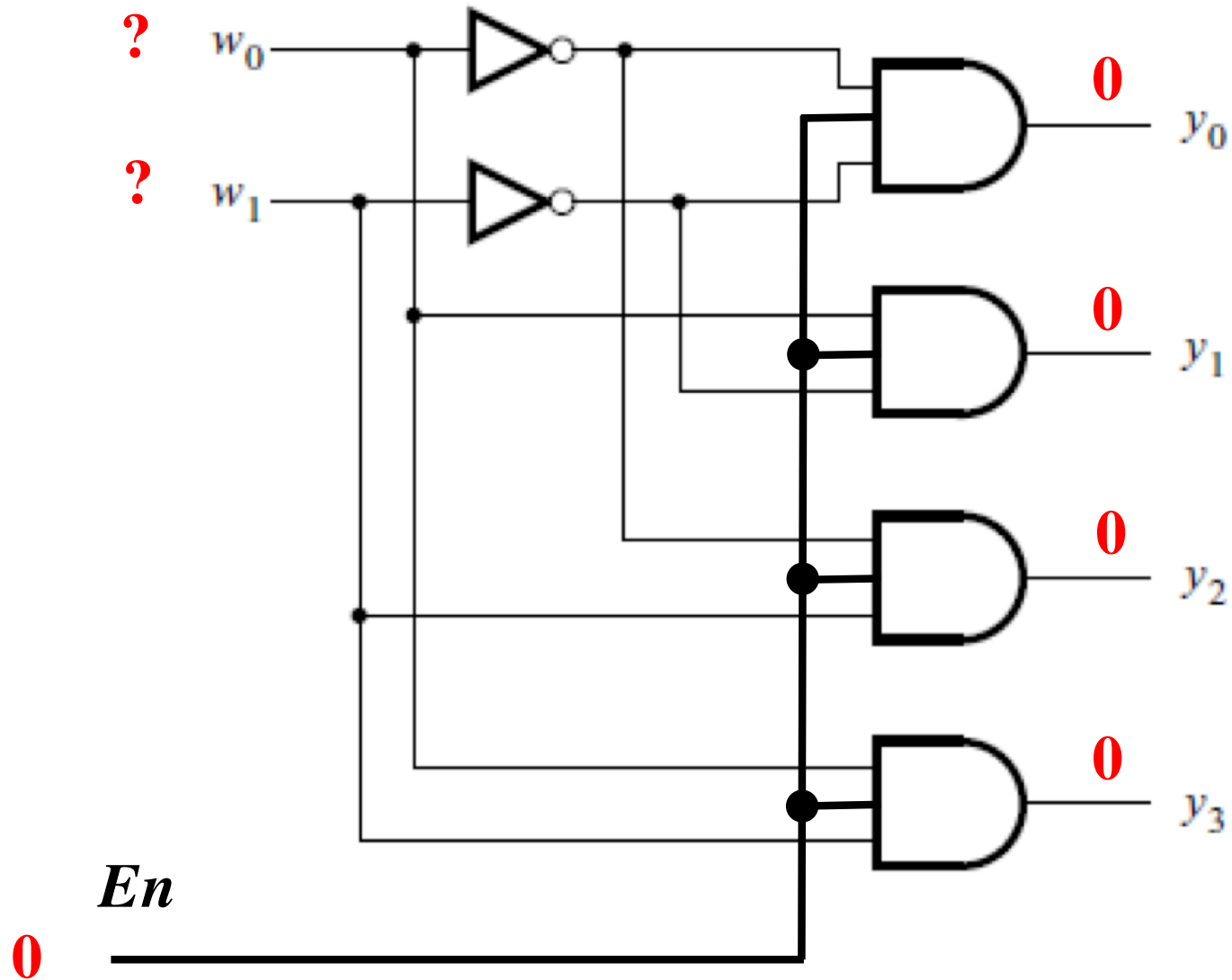
2-to-4 Decoder with Enable Input



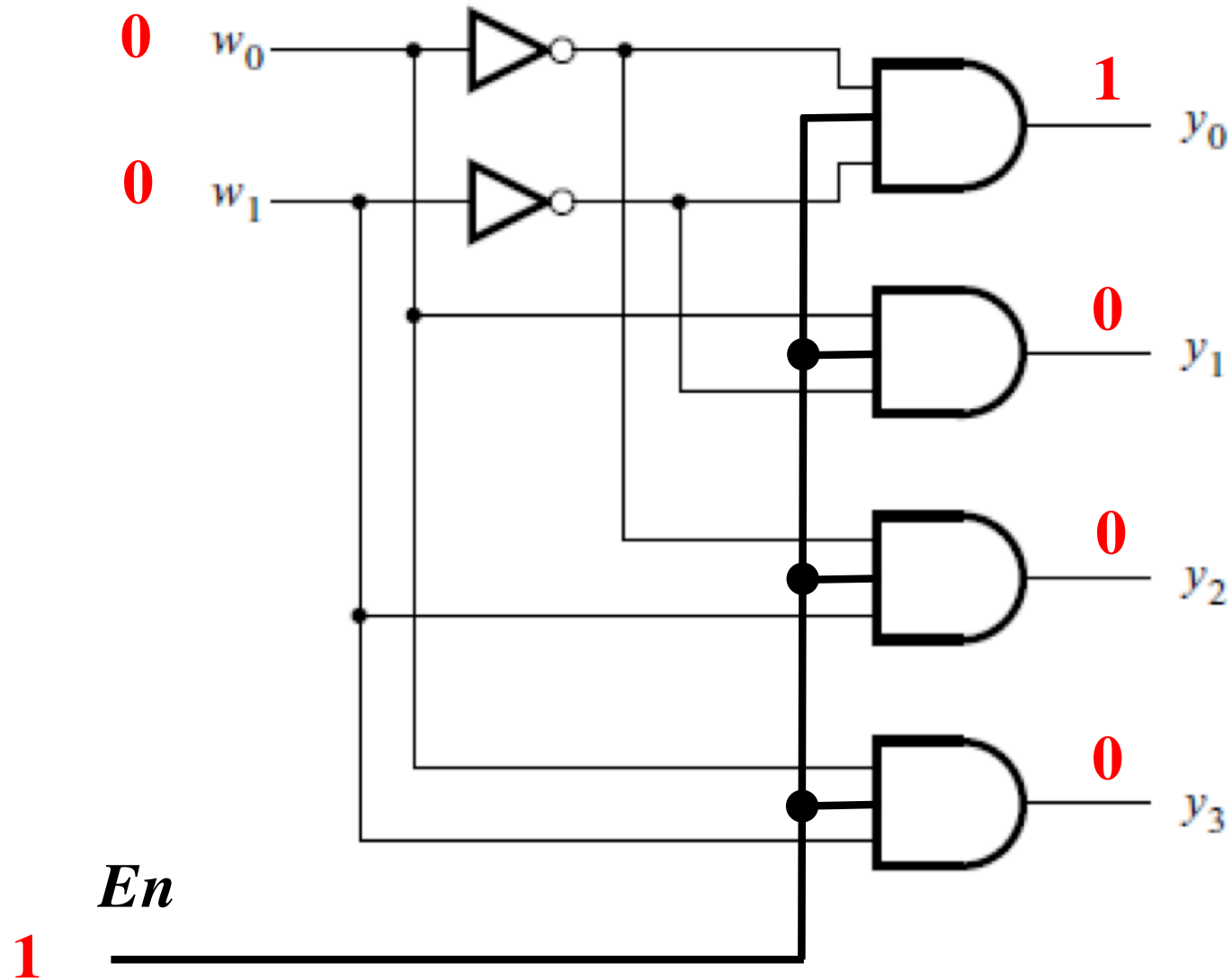
2-to-4 Decoder with Enable Input



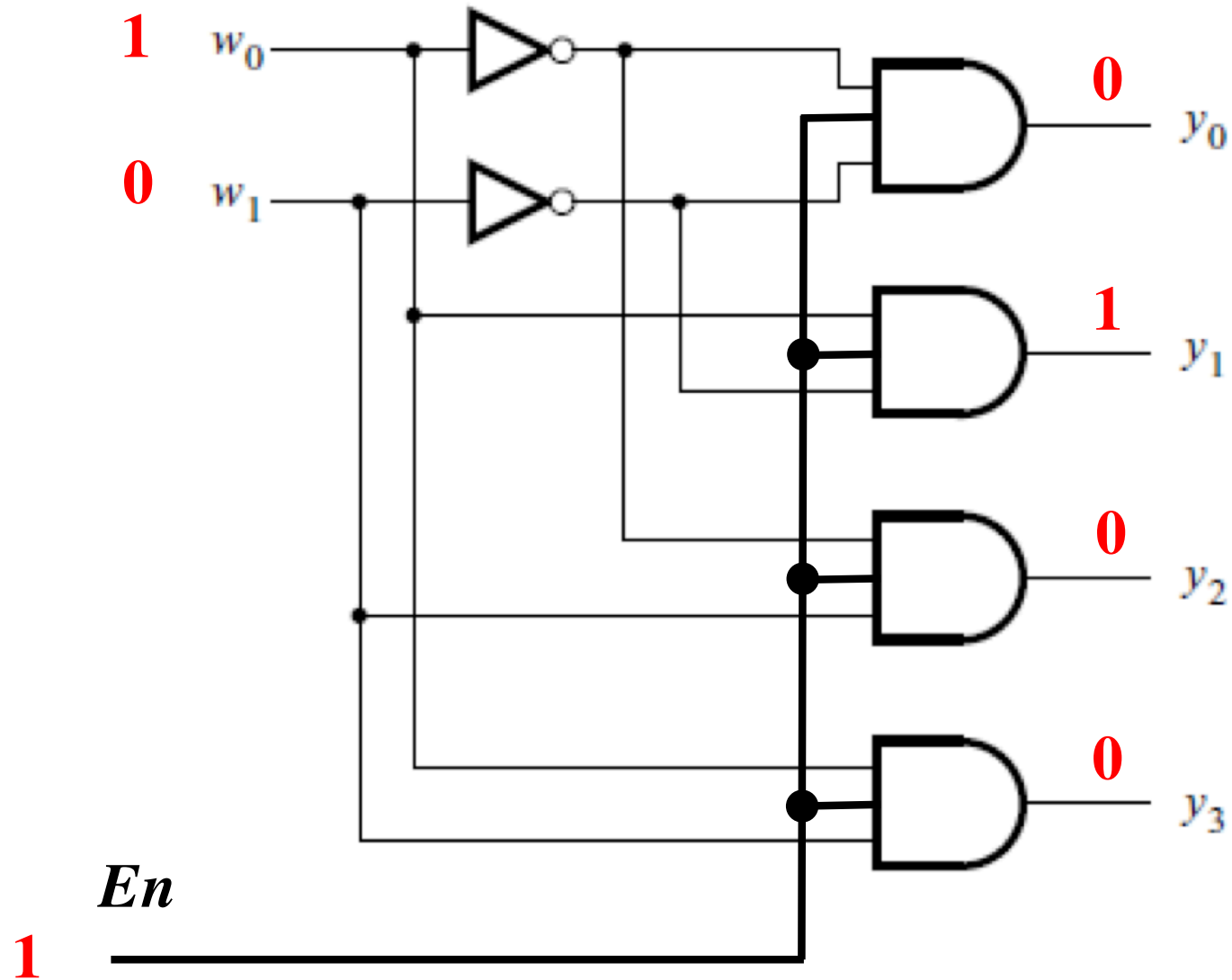
2-to-4 Decoder with Enable Input



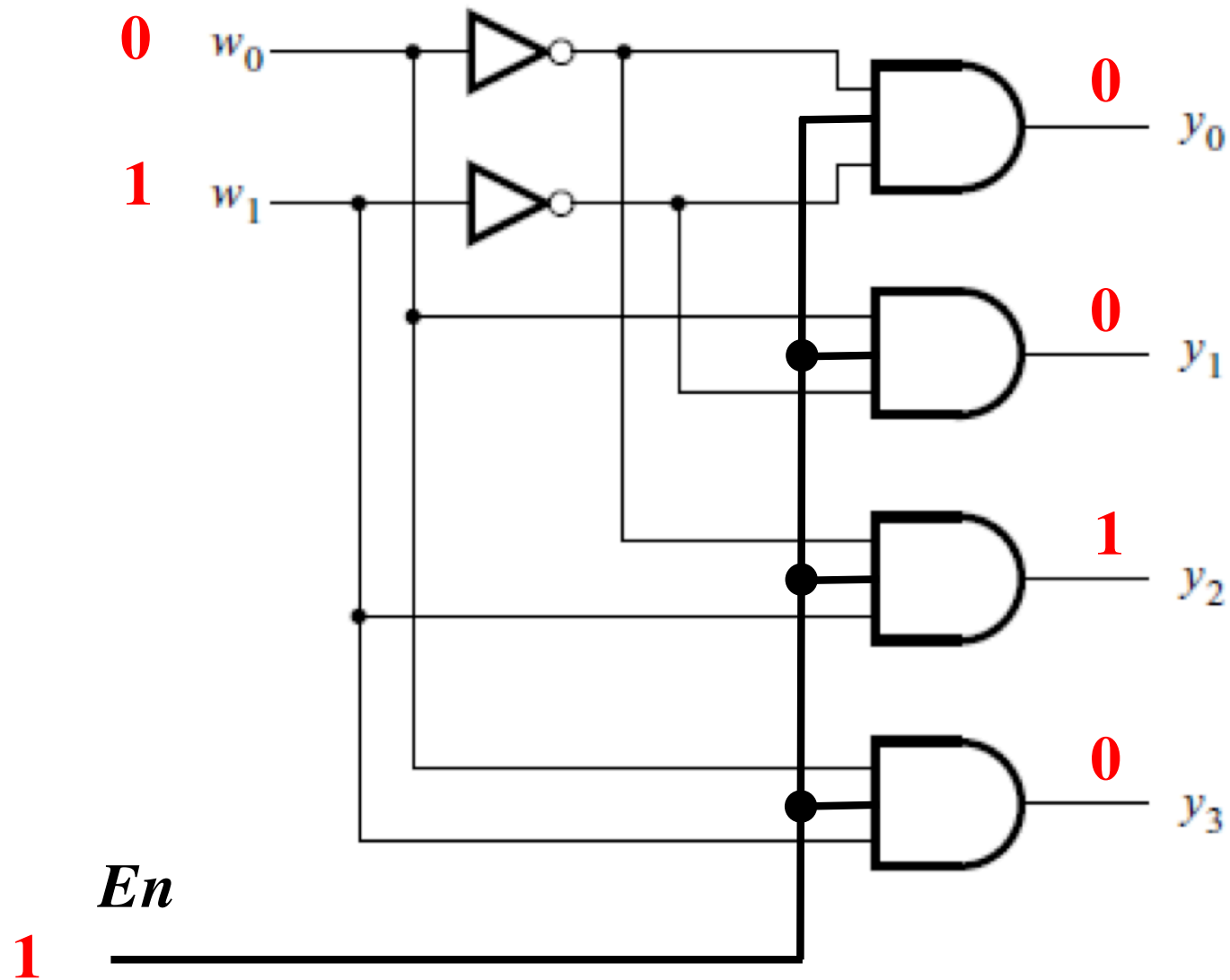
2-to-4 Decoder with Enable Input



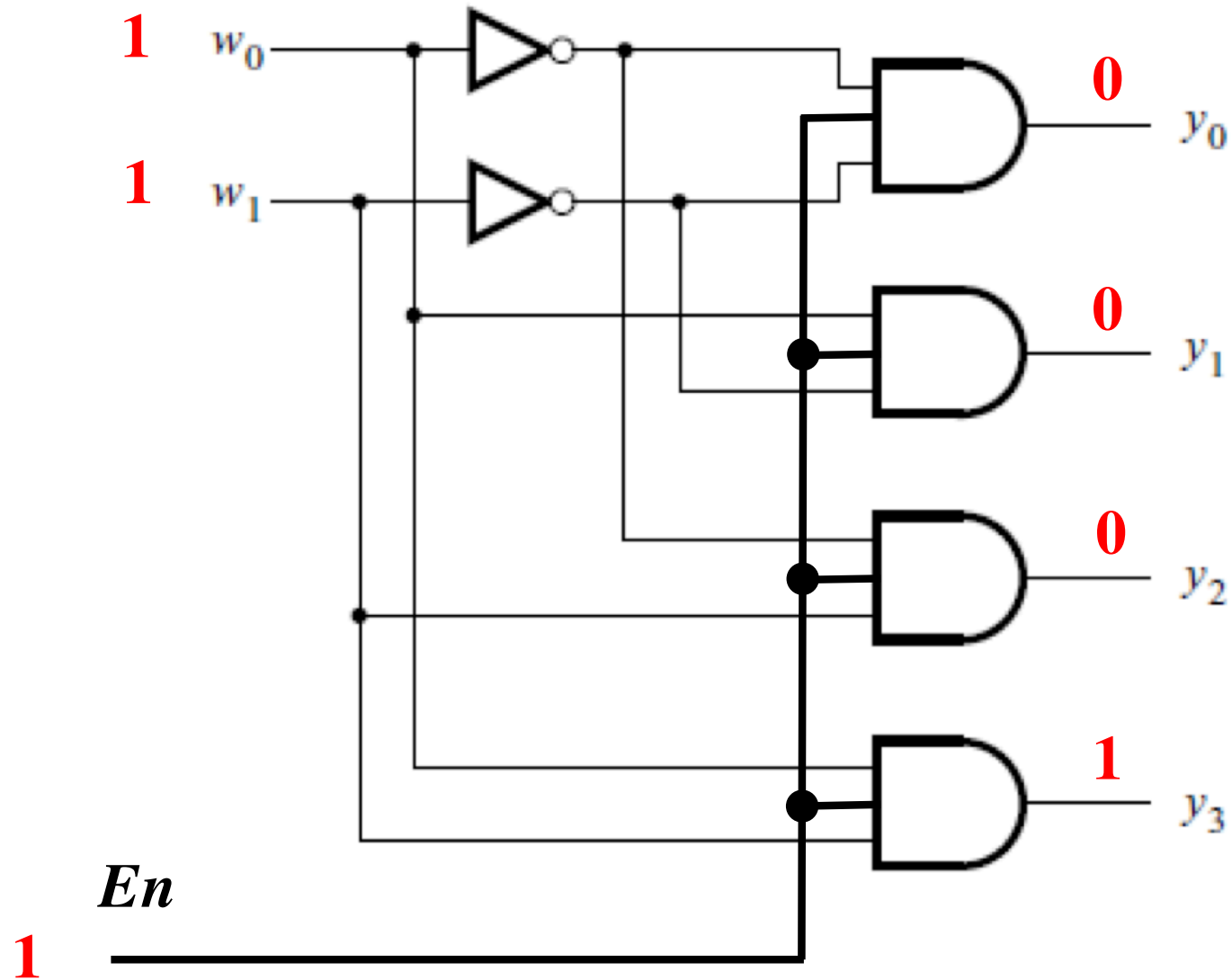
2-to-4 Decoder with Enable Input



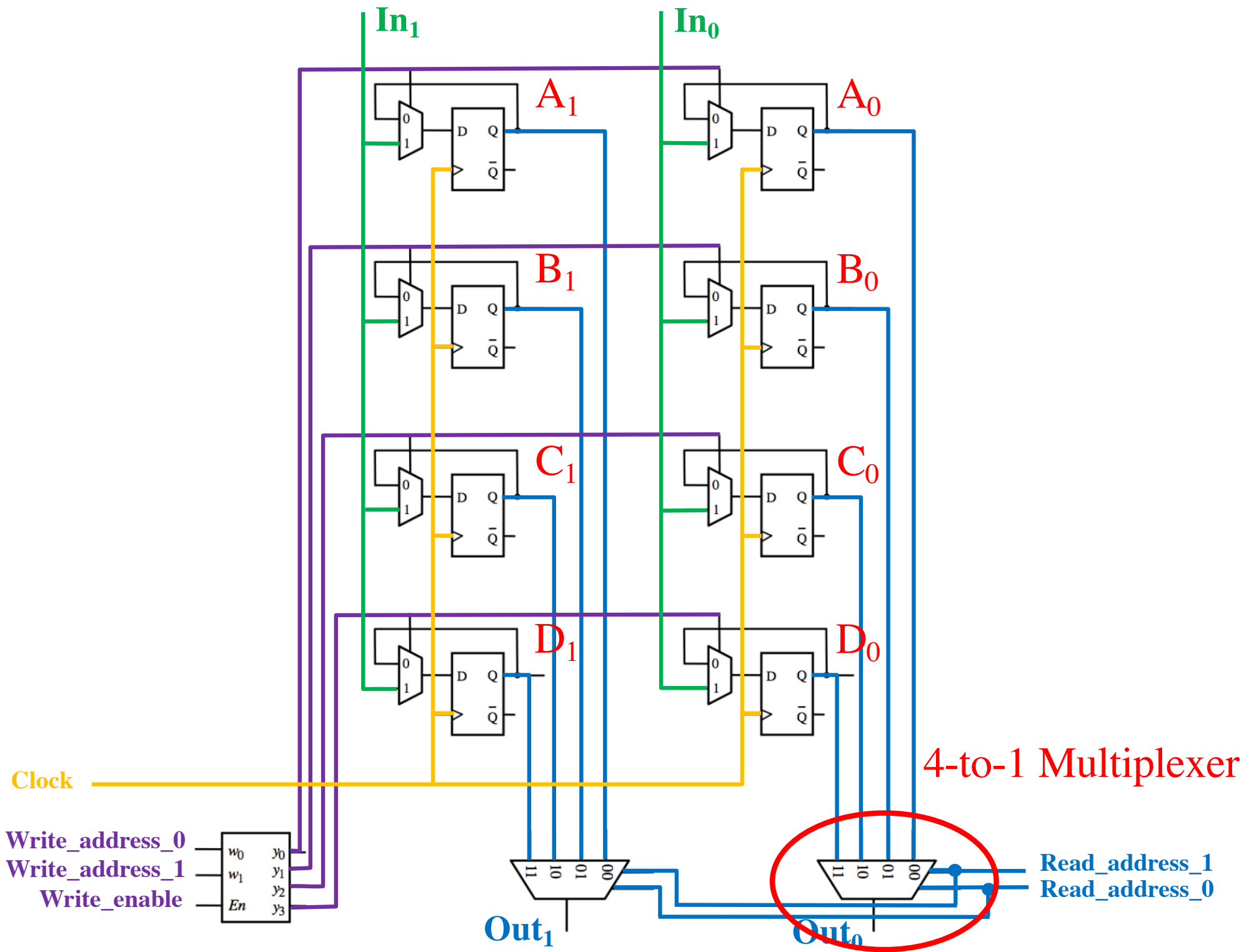
2-to-4 Decoder with Enable Input

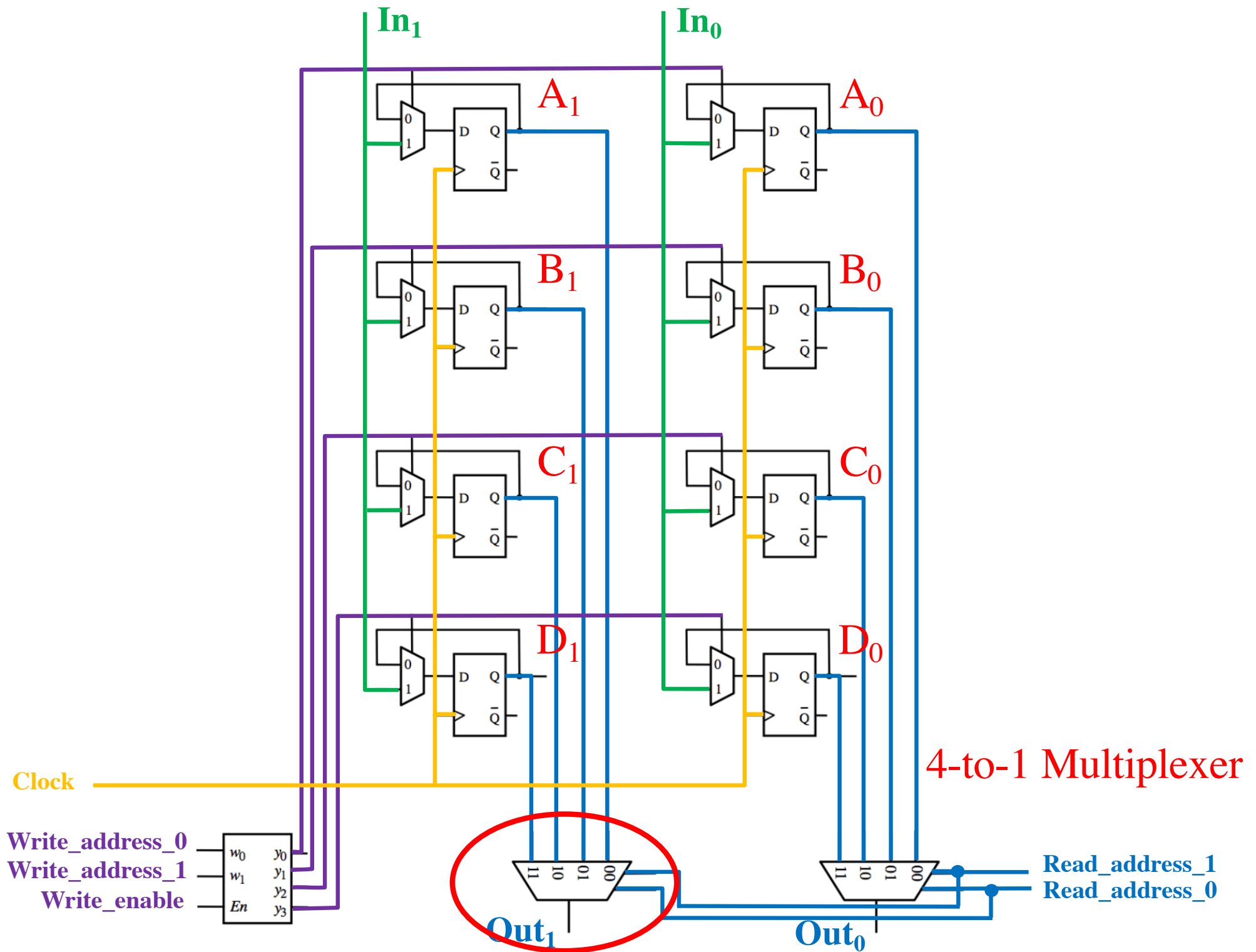


2-to-4 Decoder with Enable Input

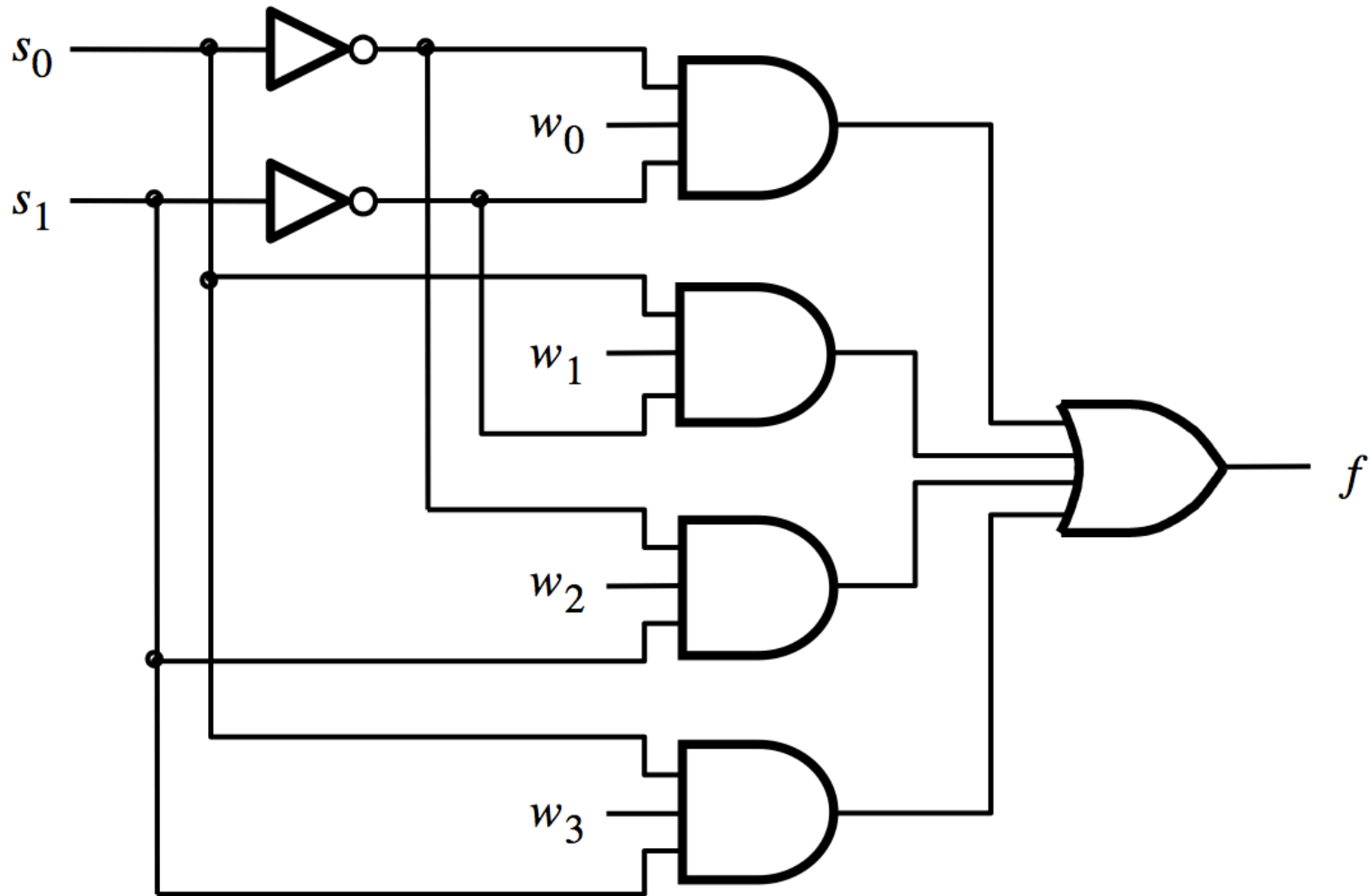


4-to-1 Multiplexer



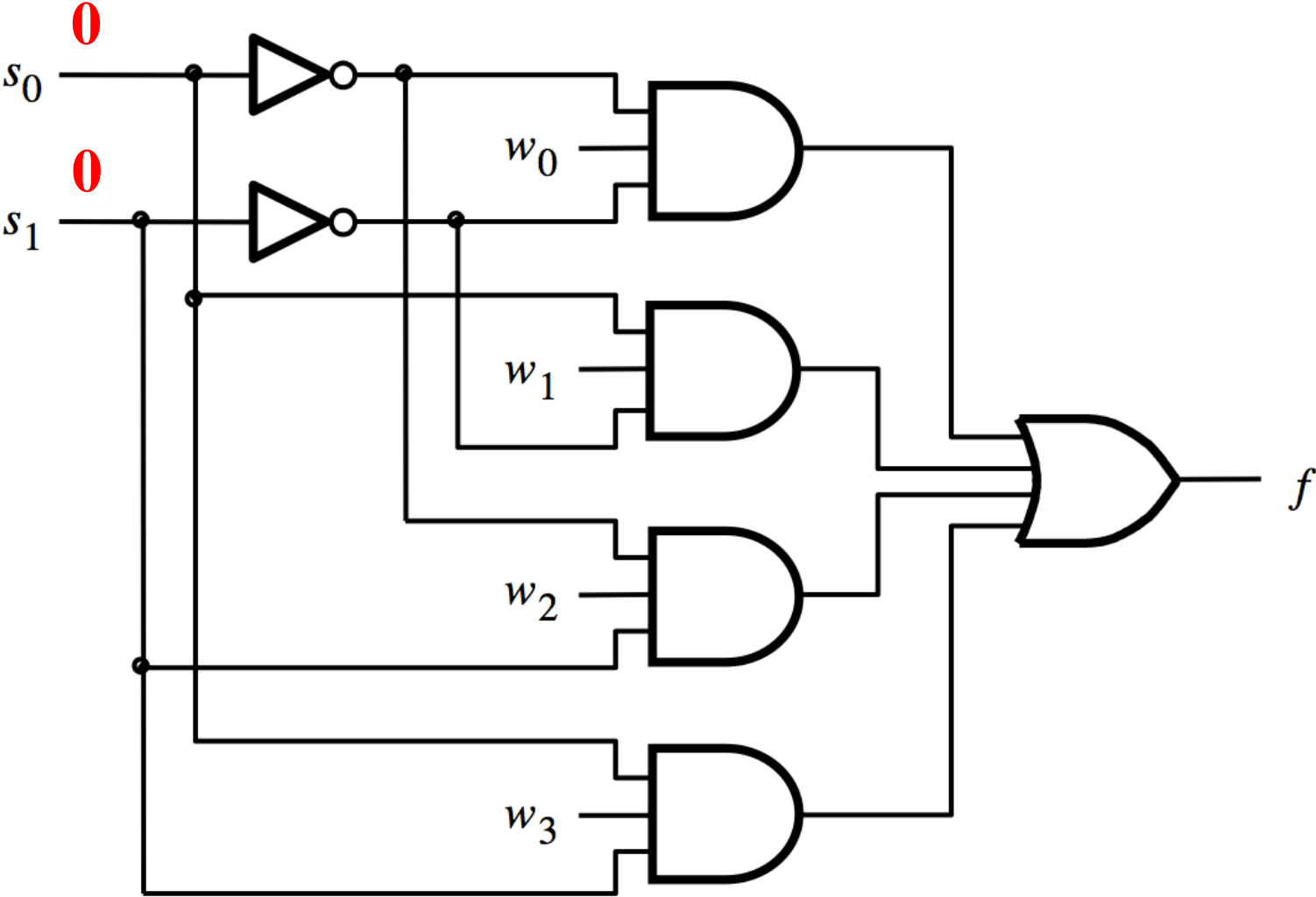


4-to-1 Multiplexer

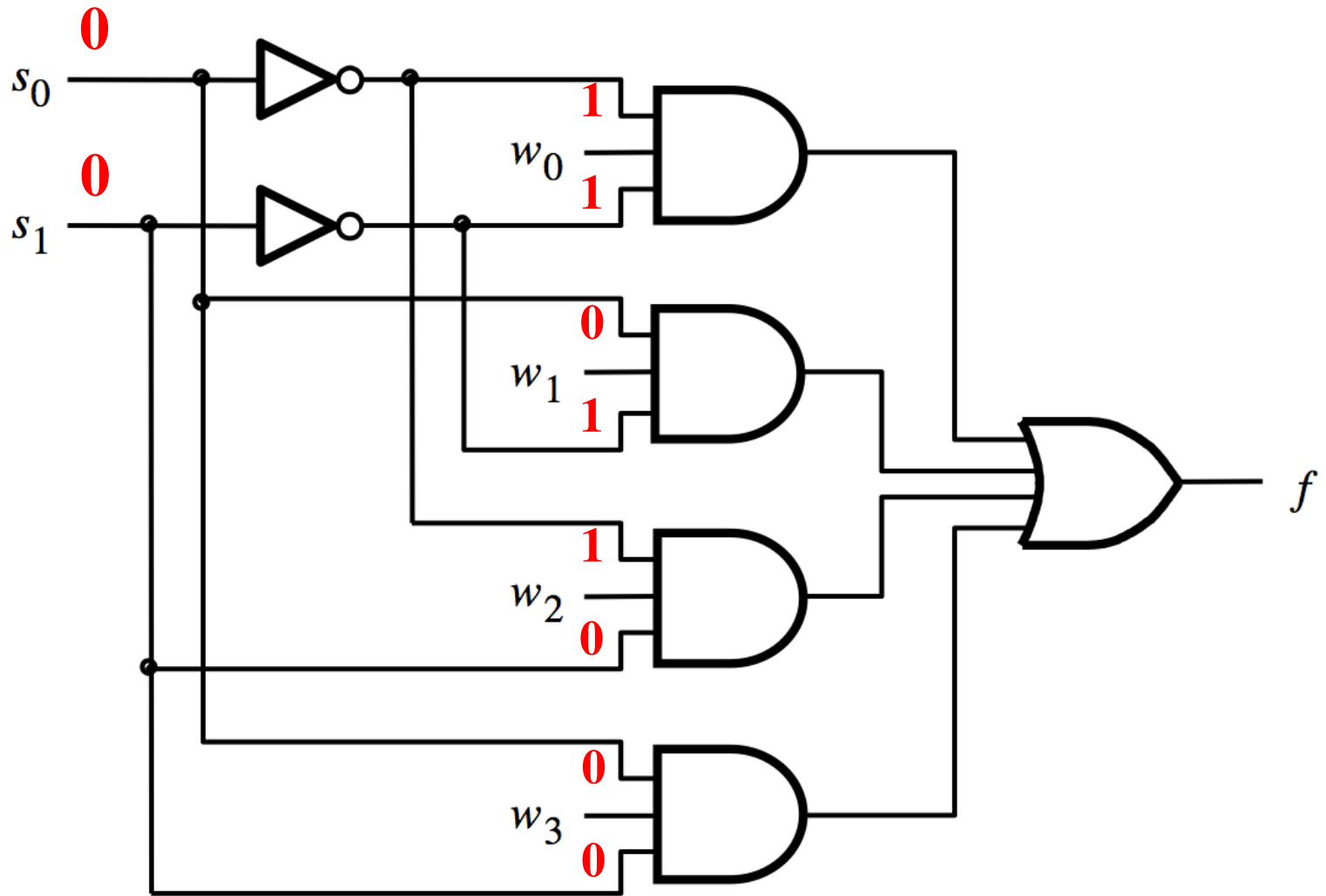


$$f = \overline{s_1} \overline{s_0} w_0 + \overline{s_1} s_0 w_1 + s_1 \overline{s_0} w_2 + s_1 s_0 w_3$$

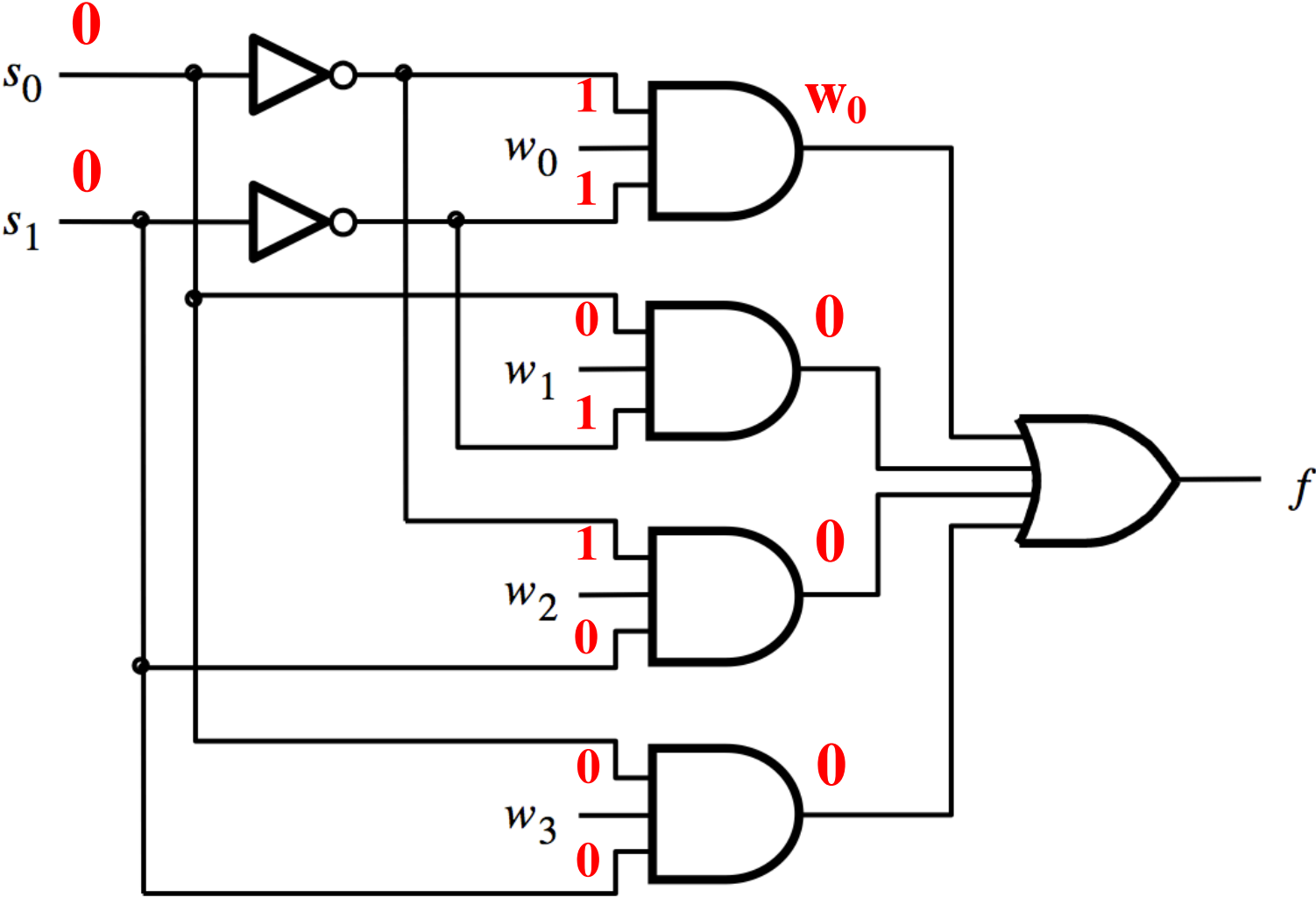
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=0$)



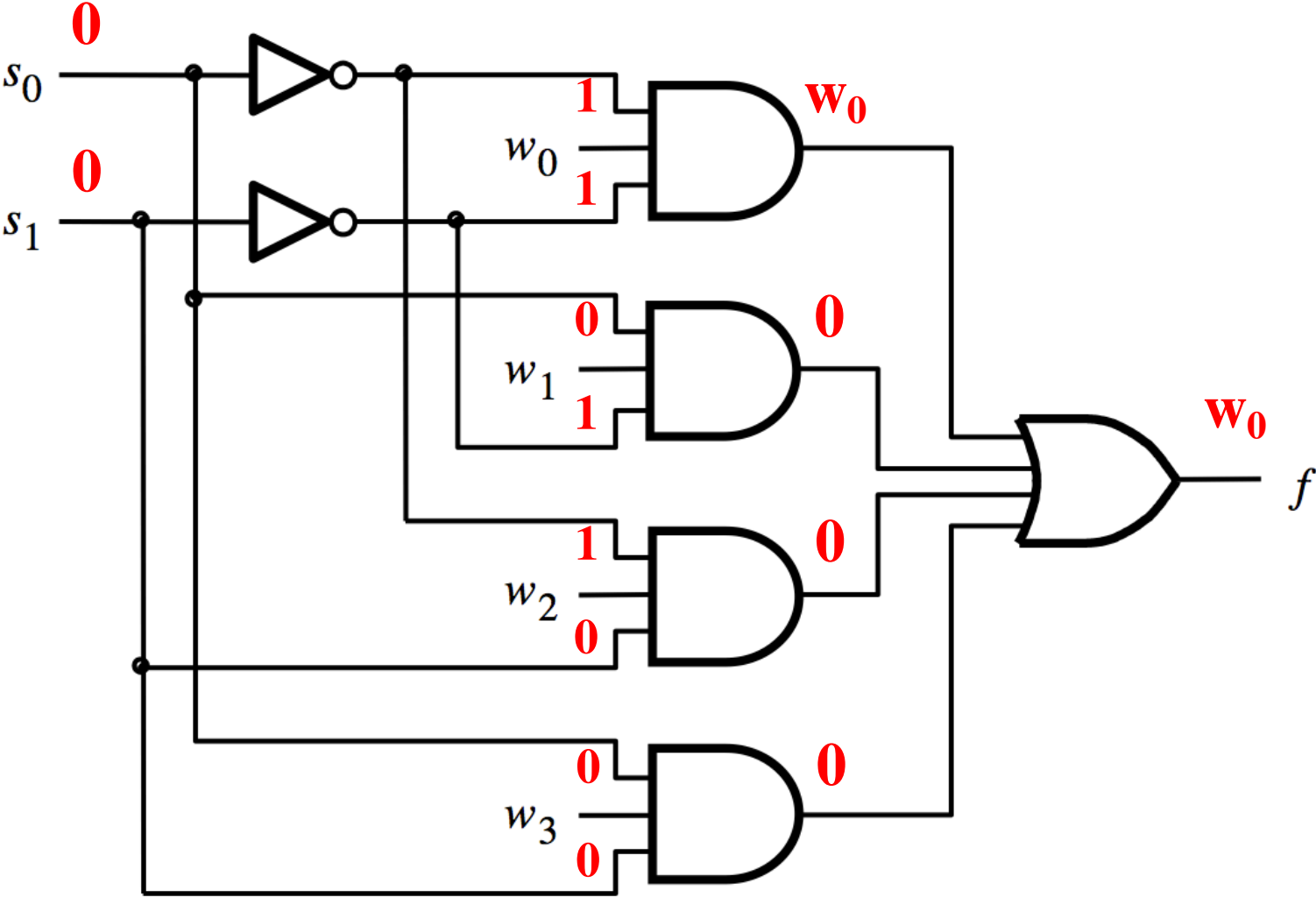
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=0$)



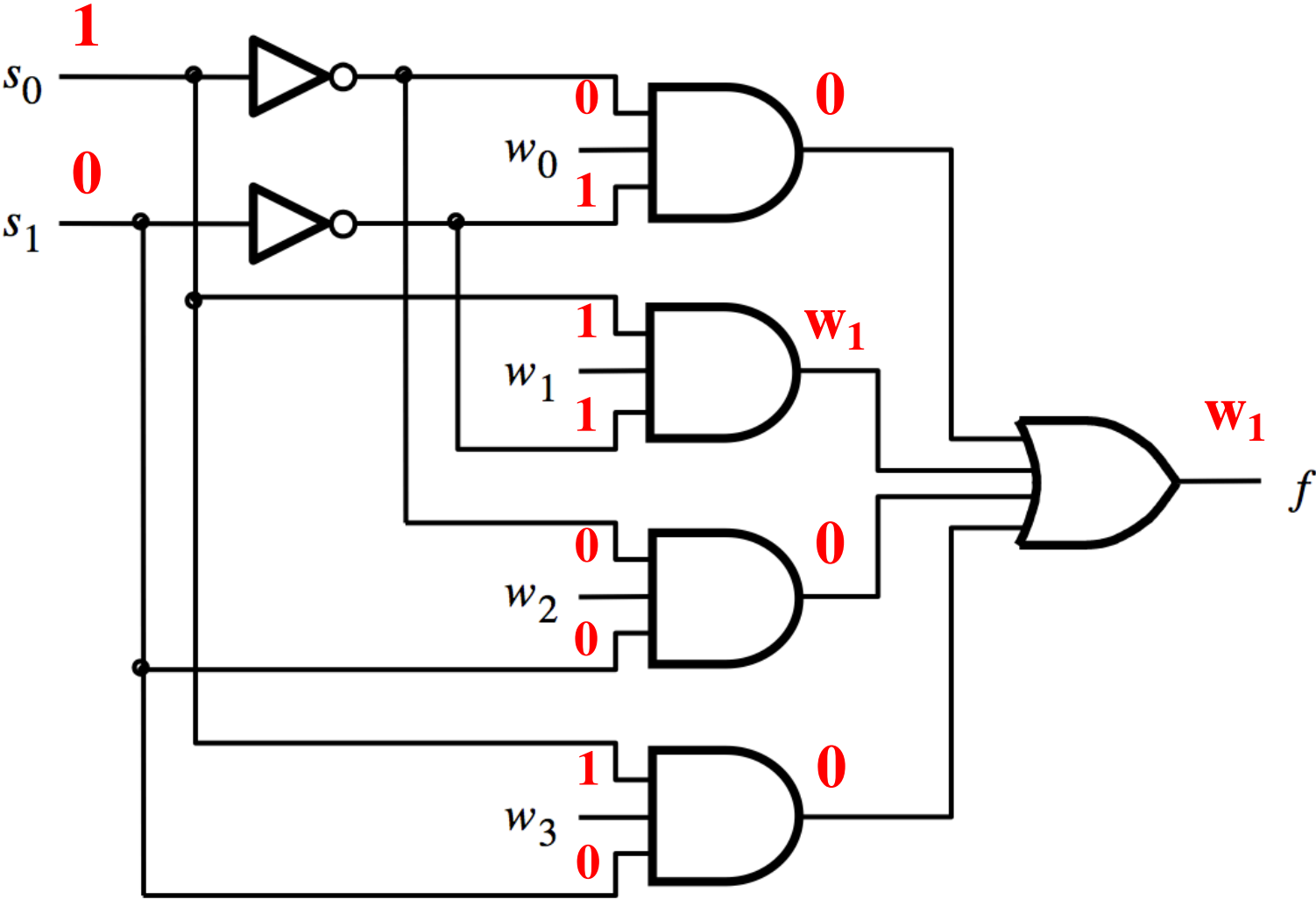
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=0$)



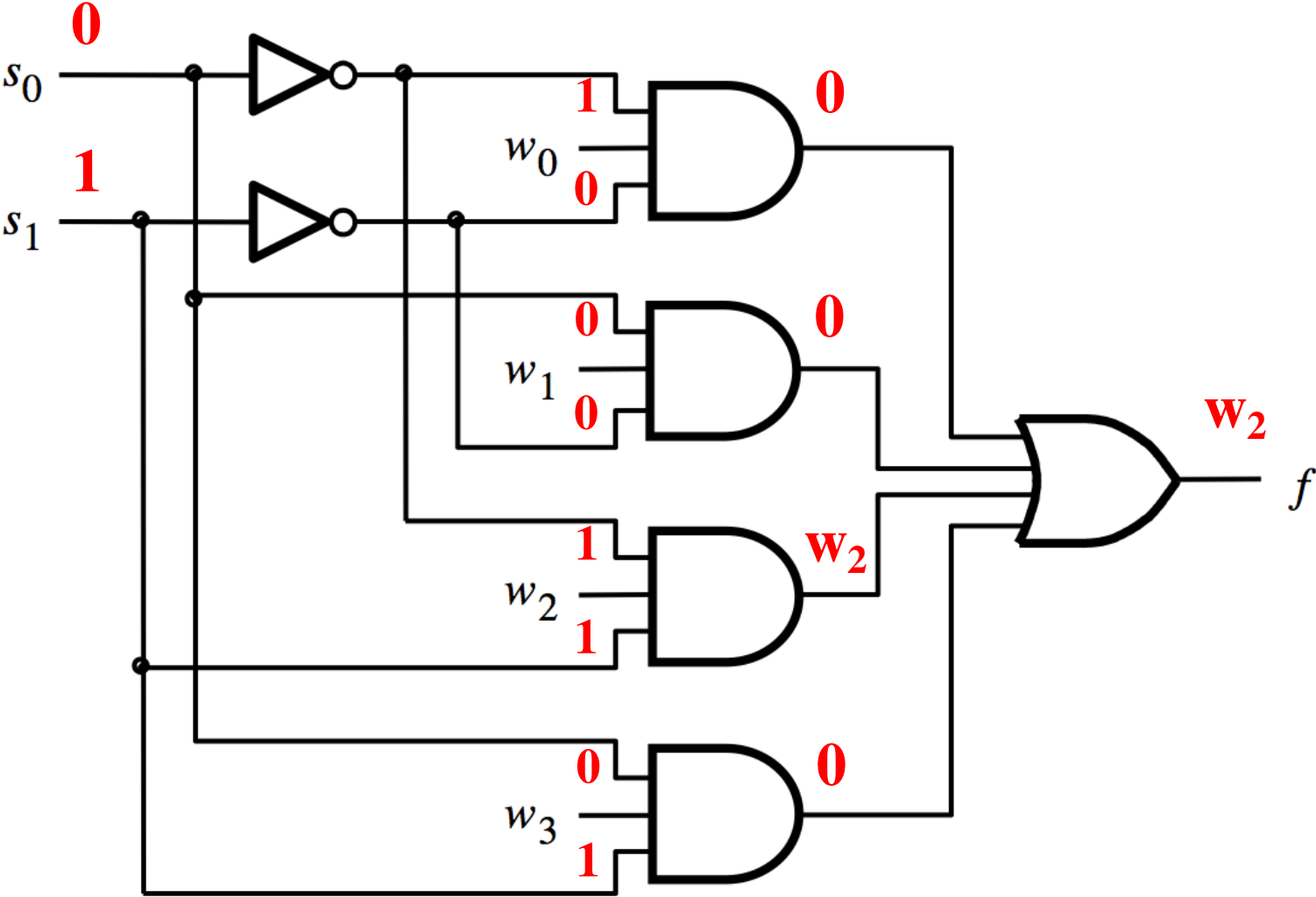
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=0$)



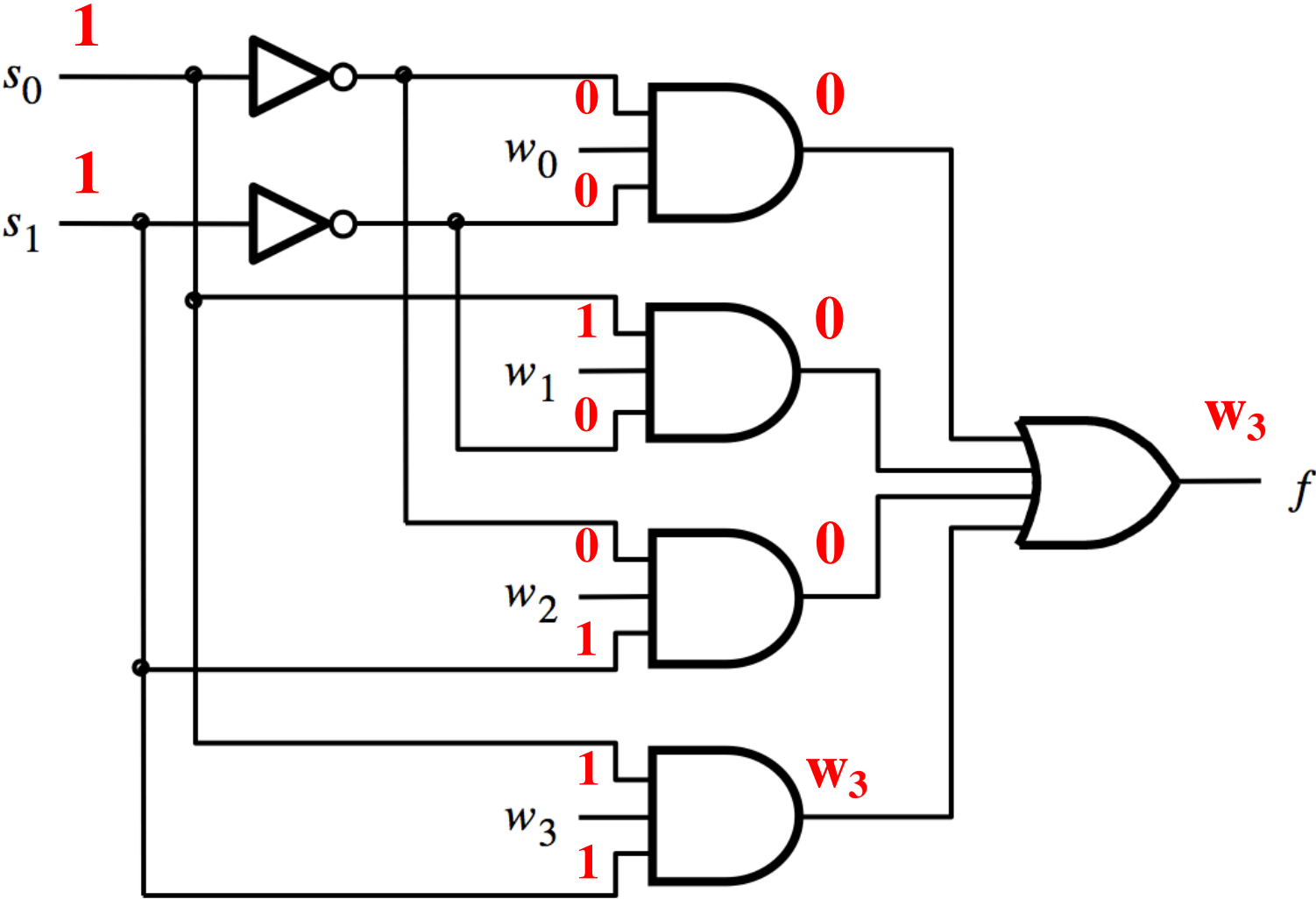
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=1$)



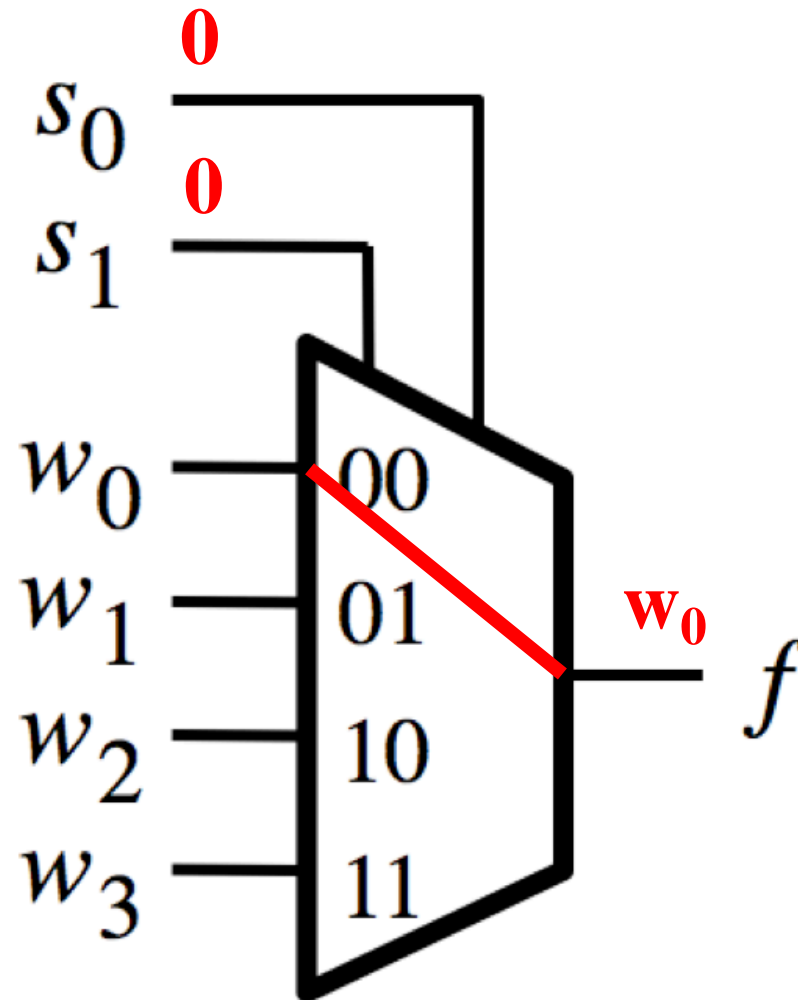
Analysis of the 4-to-1 Multiplexer ($s_1=1$ and $s_0=0$)



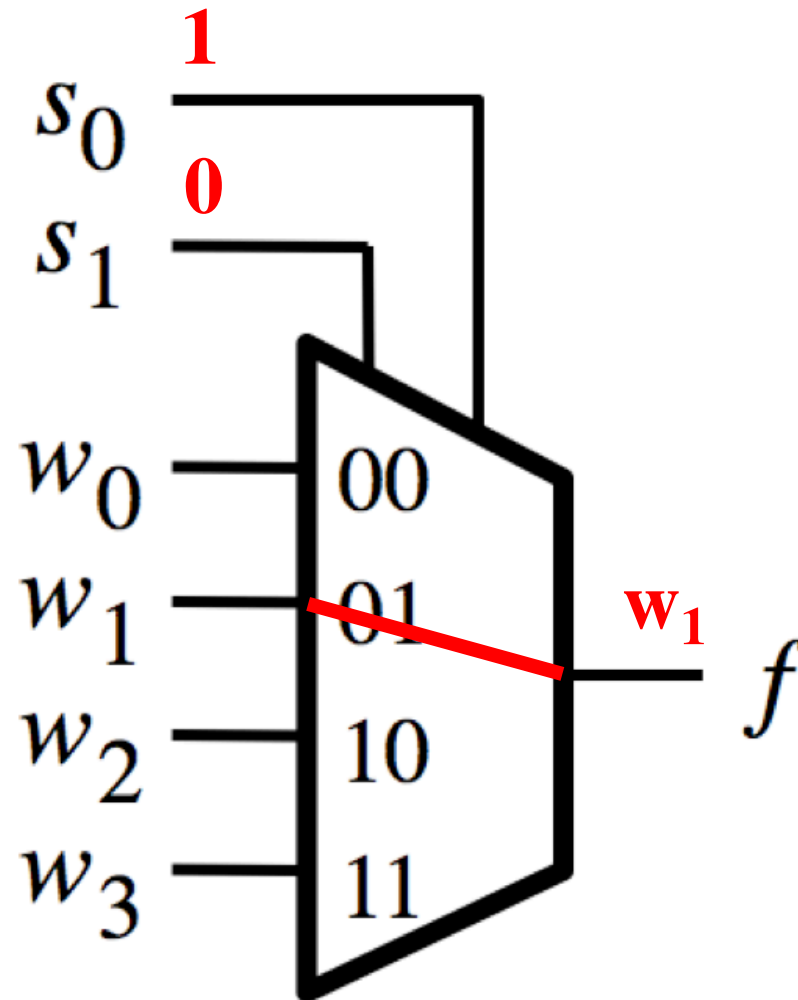
Analysis of the 4-to-1 Multiplexer ($s_1=1$ and $s_0=1$)



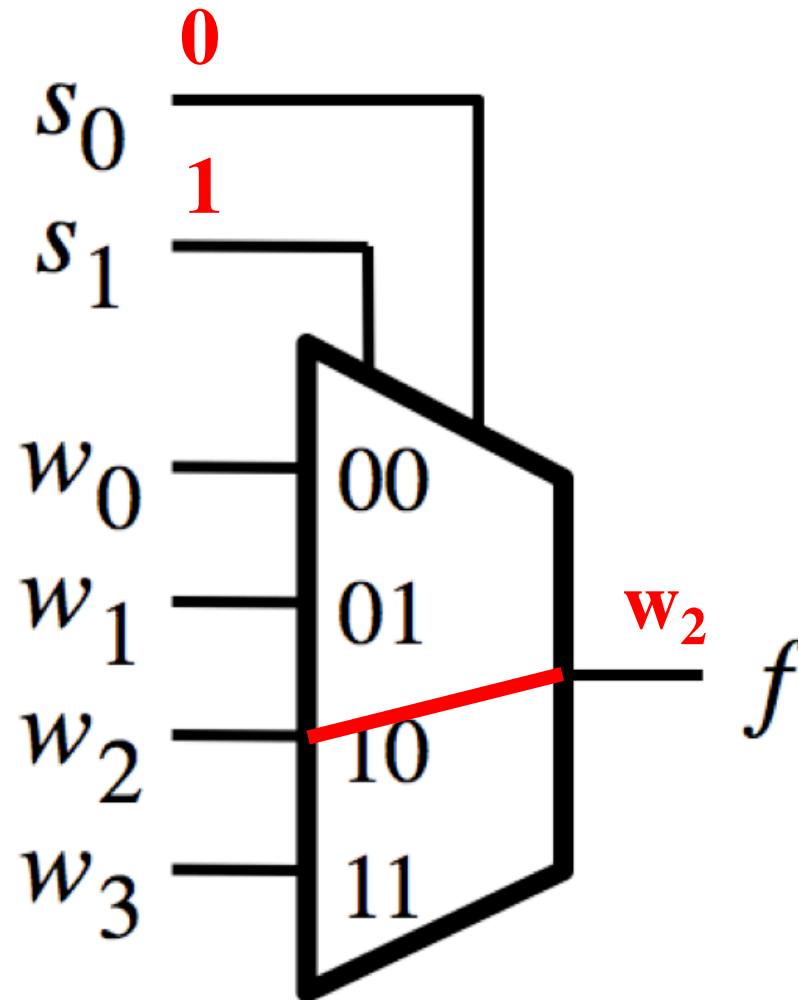
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=0$)



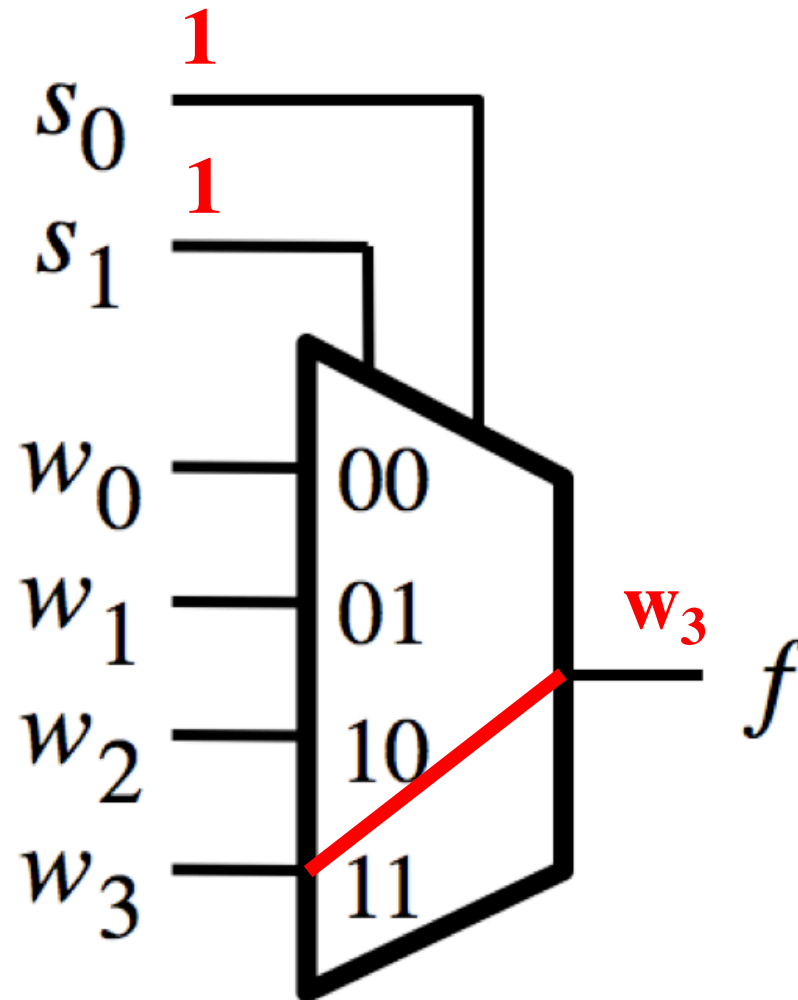
Analysis of the 4-to-1 Multiplexer ($s_1=0$ and $s_0=1$)



Analysis of the 4-to-1 Multiplexer ($s_1=1$ and $s_0=0$)



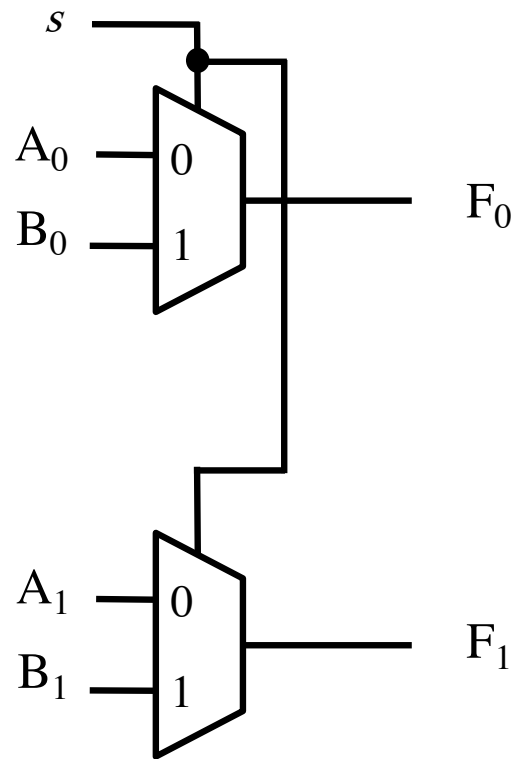
Analysis of the 4-to-1 Multiplexer ($s_1=1$ and $s_0=1$)



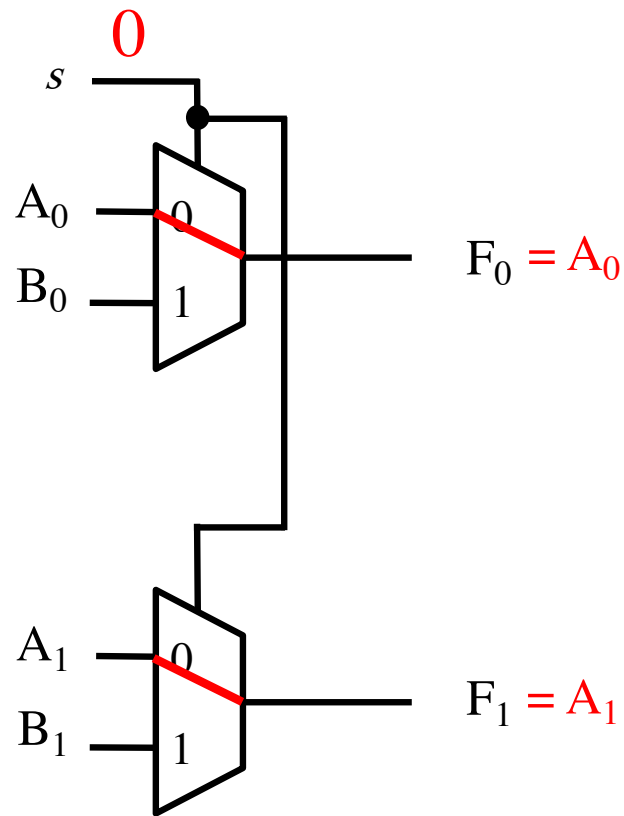
Multiplexer Tricks

(select one of two 2-bit numbers)

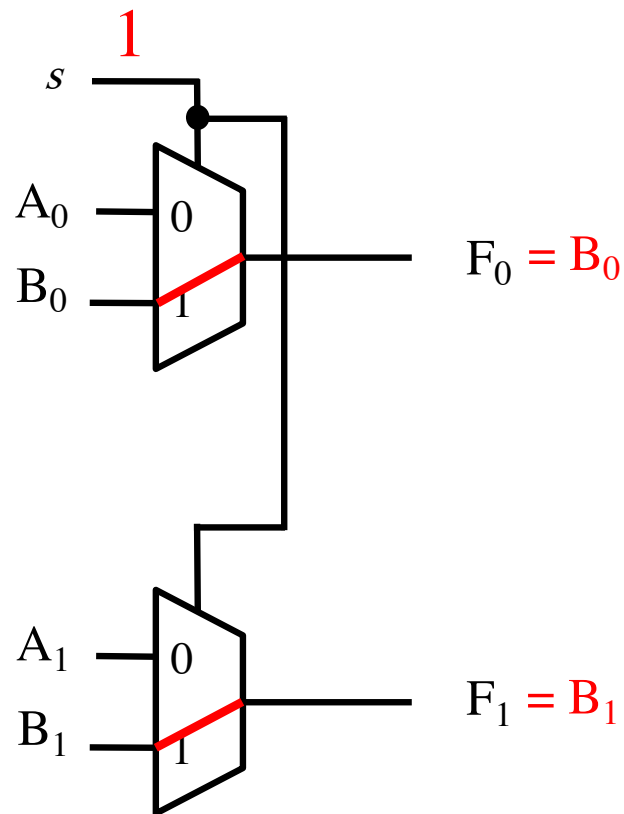
Select Either $A=A_1A_0$ or $B=B_1B_0$



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



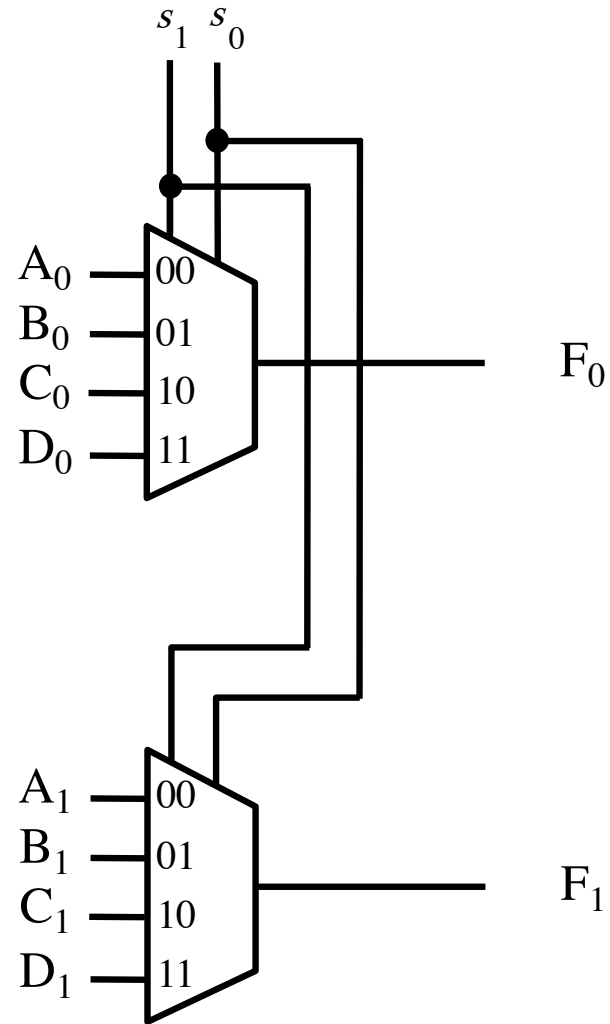
Select Either $A=A_1A_0$ or $B=B_1B_0$



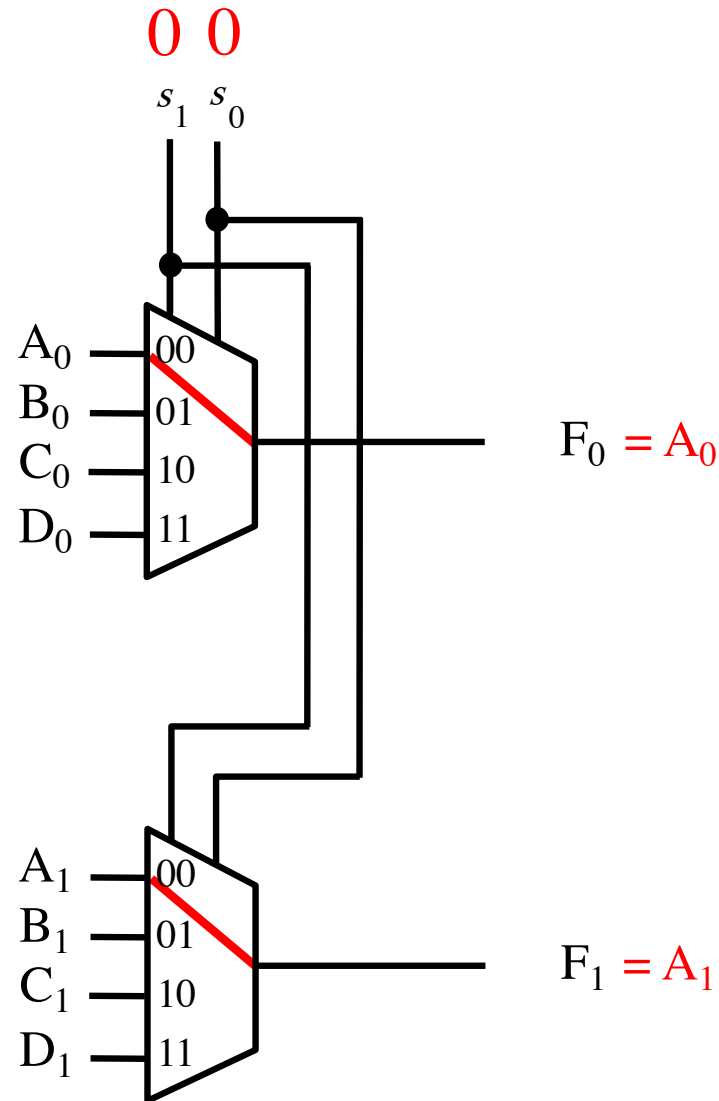
Multiplexer Tricks

(select one of four 2-bit numbers)

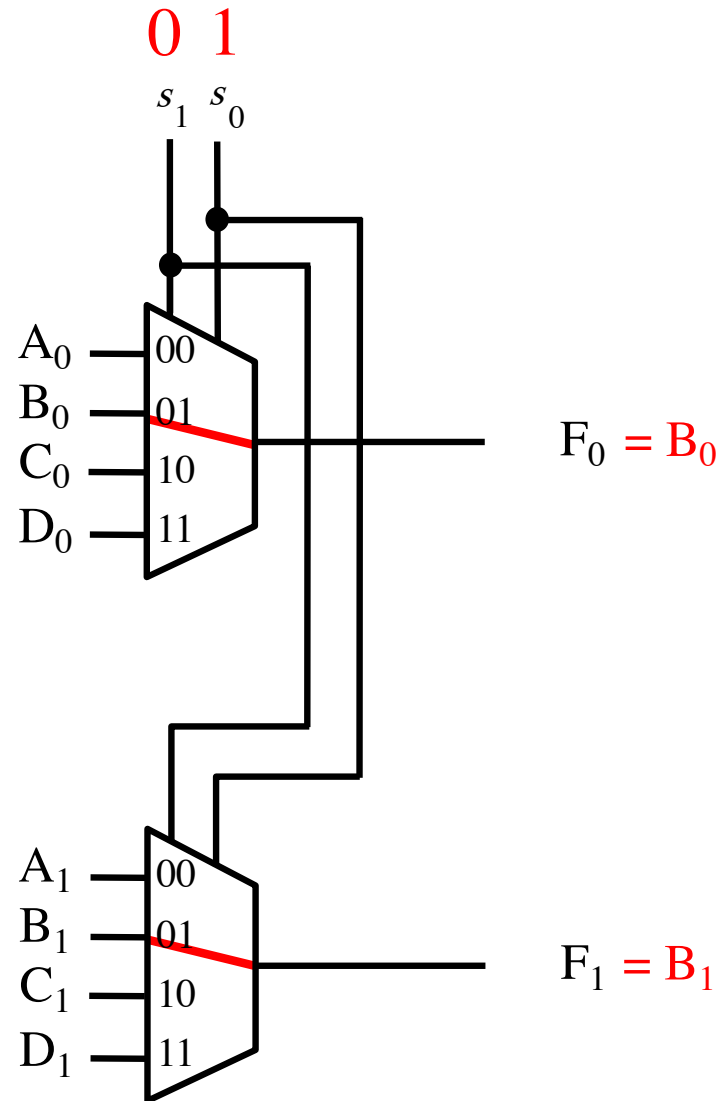
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



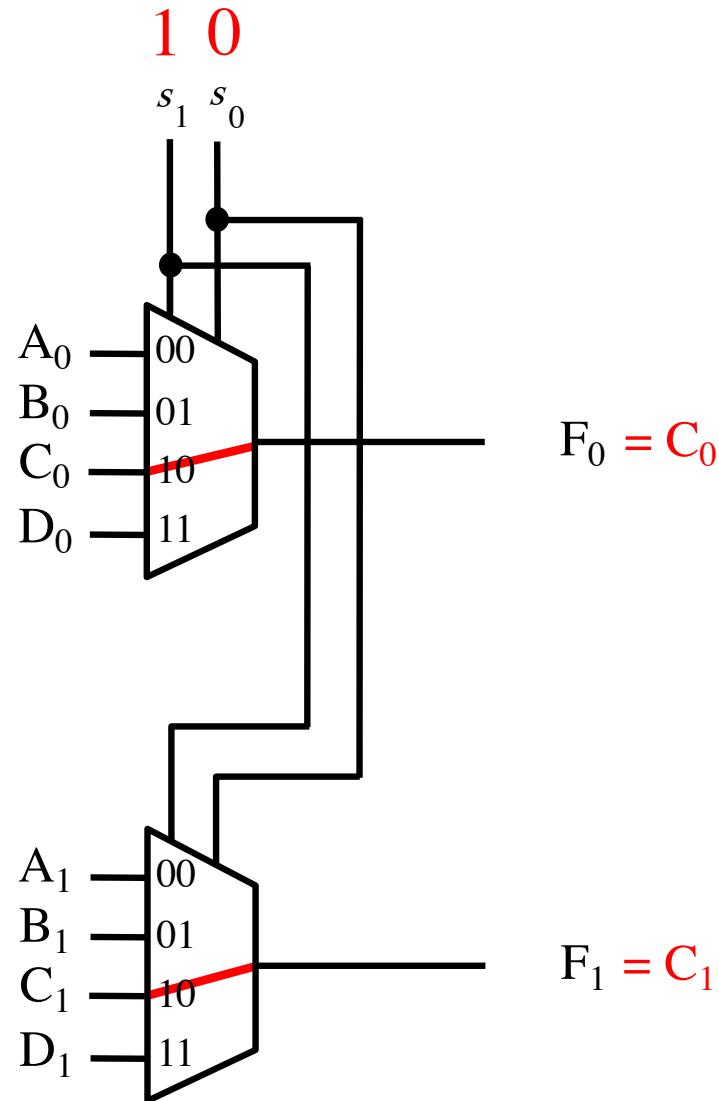
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



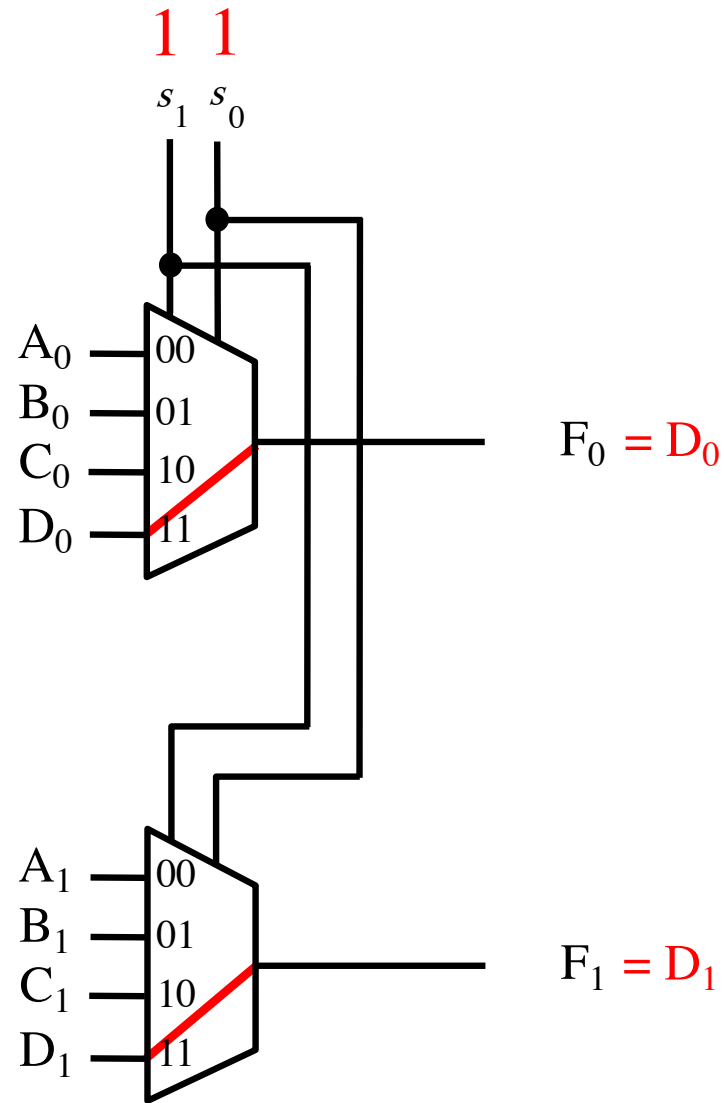
Select $A=A_1A_0$ or $B=B_1B_0$ or $C=C_1C_0$ or $D=D_1D_0$



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 $D=D_1D_0$

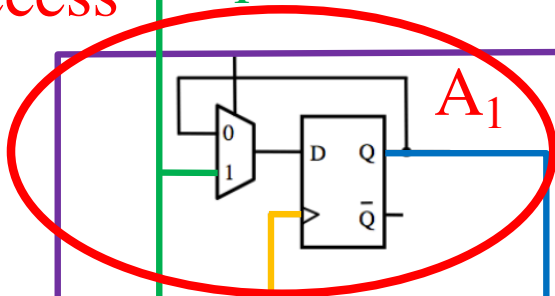


1-bit Parallel-Access Register

1-bit Parallel-Access Register

In₁

In₀



A₁

A₀

B₁

B₀

C₁

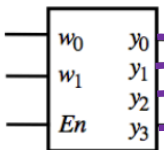
C₀

D₁

D₀

Clock

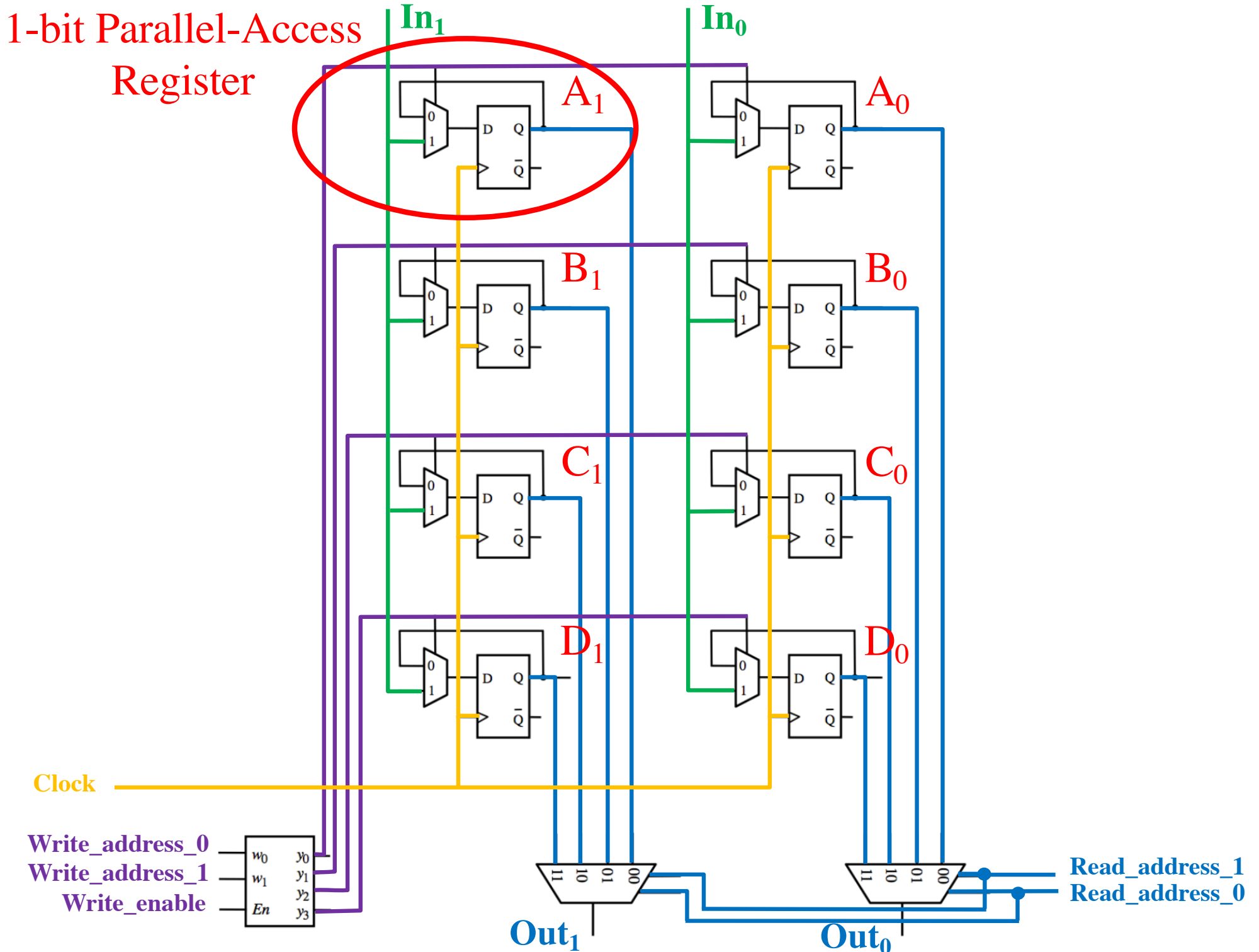
Write_address_0
Write_address_1
Write_enable



Out₁

Out₀

Read_address_1
Read_address_0



1-bit Parallel-Access Register

x 8

In₁

In₀

A₁

A₀

B₁

B₀

C₁

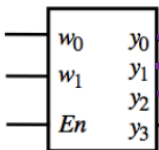
C₀

D₁

D₀

Clock

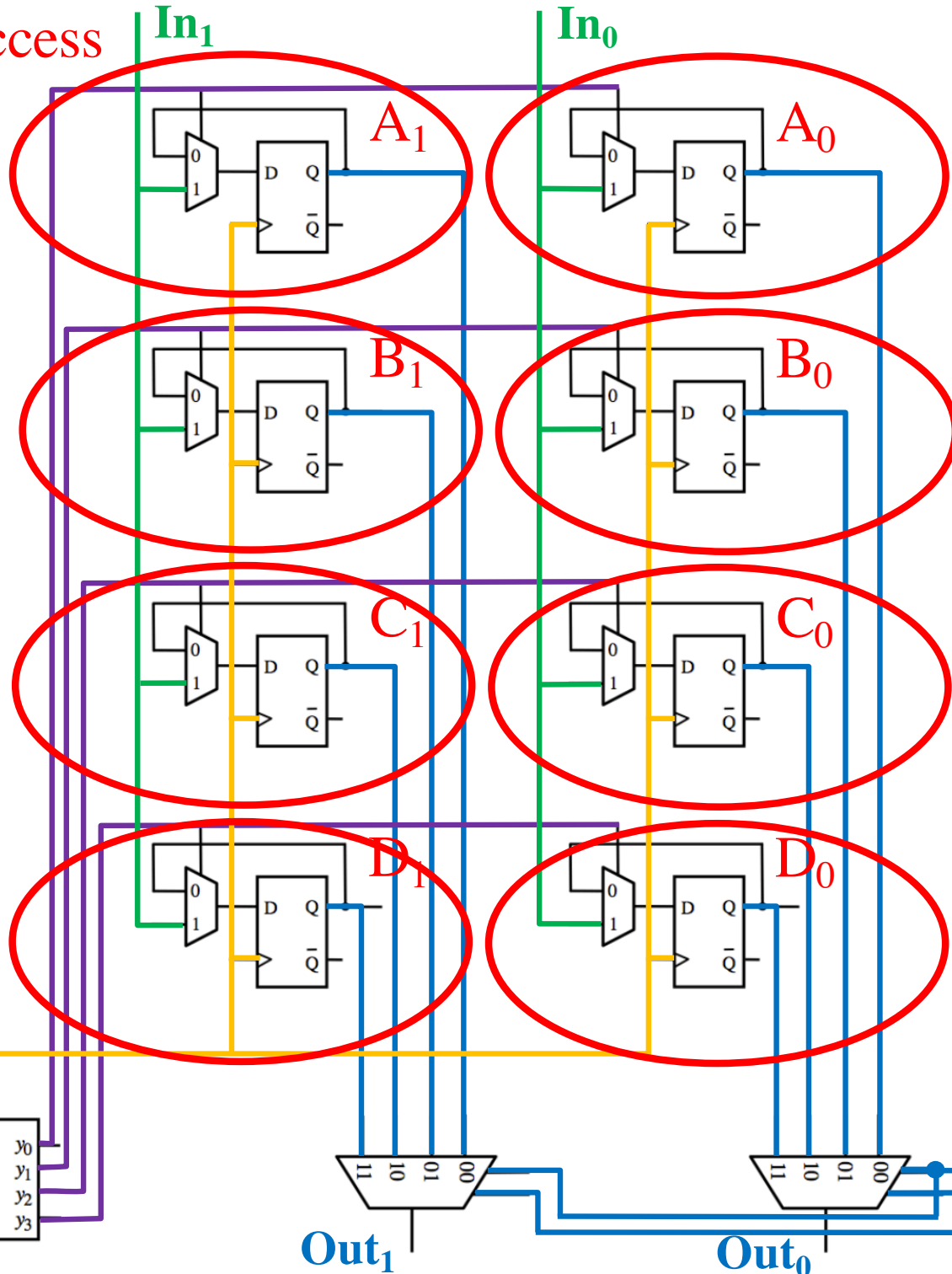
Write_address_0
Write_address_1
Write_enable



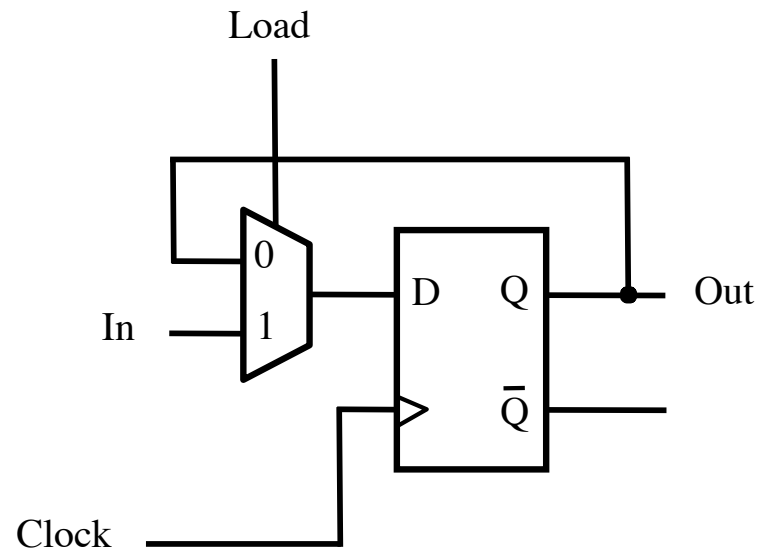
Out₁

Out₀

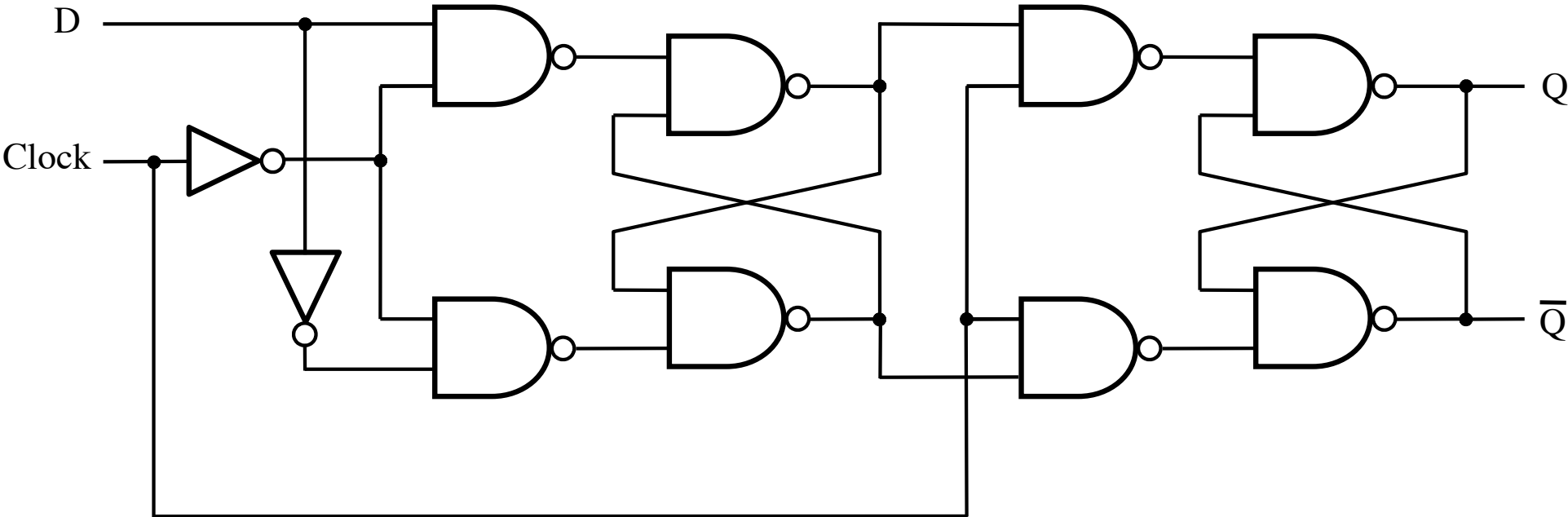
Read_address_1
Read_address_0



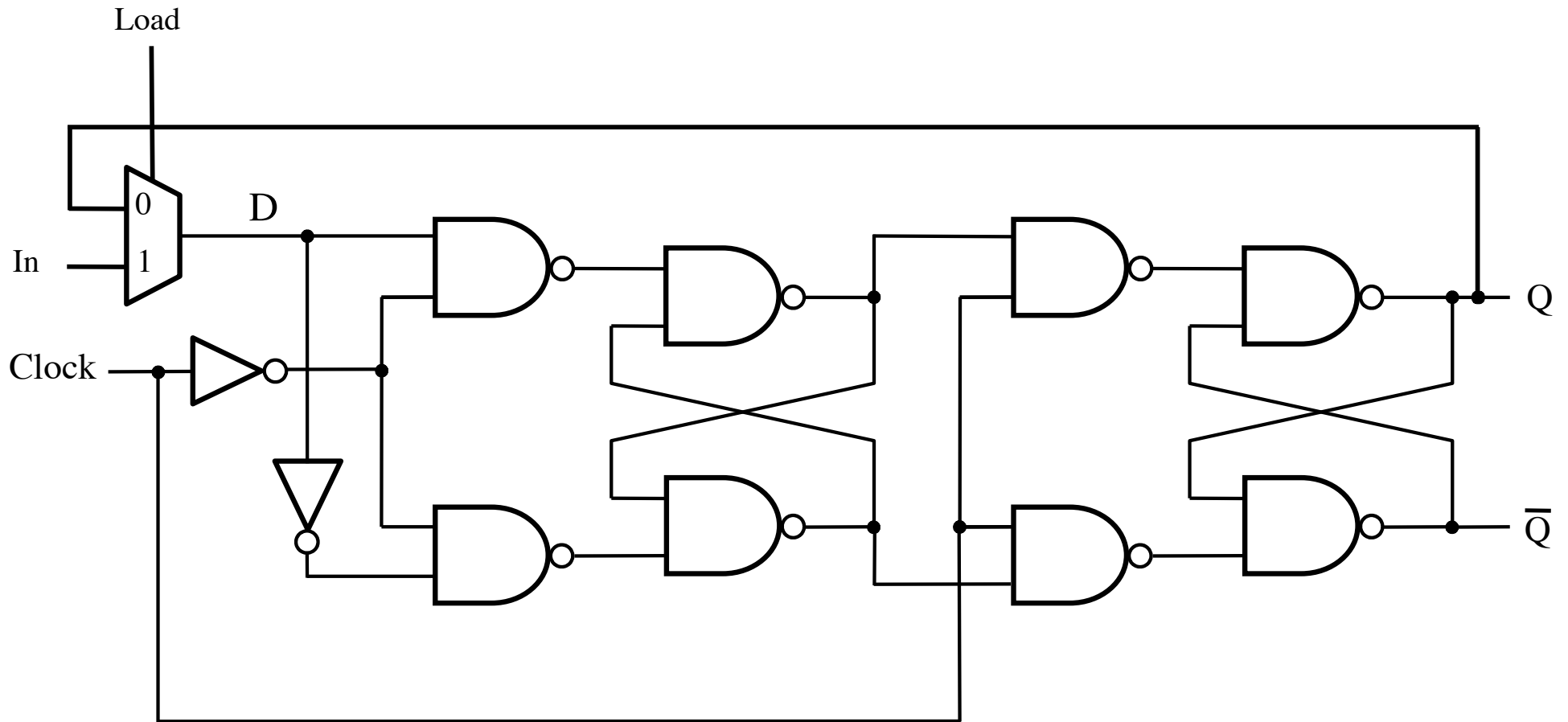
1-Bit Parallel-Access Register



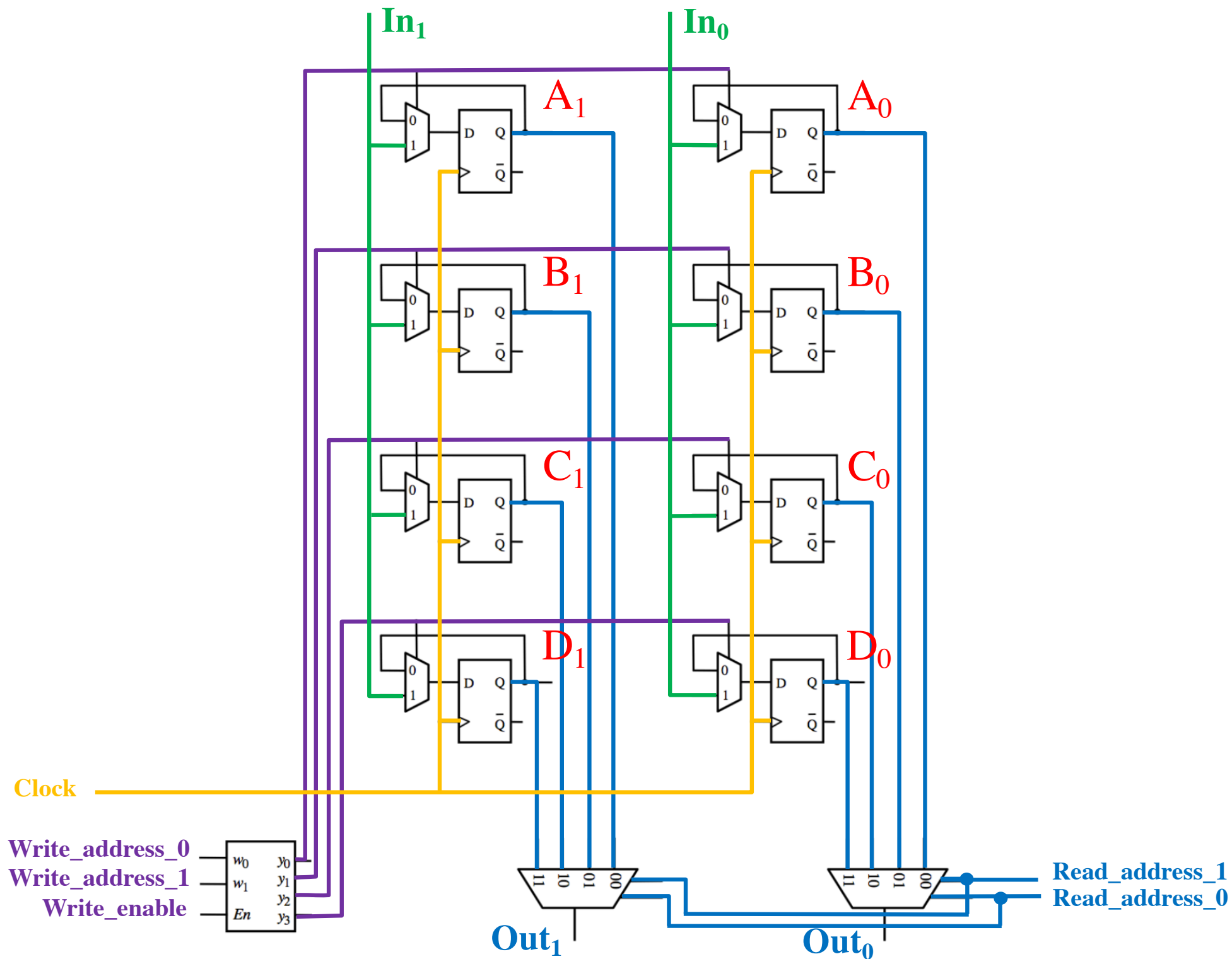
Positive-Edge-Triggered D Flip-Flop



1-bit Parallel-Access Register

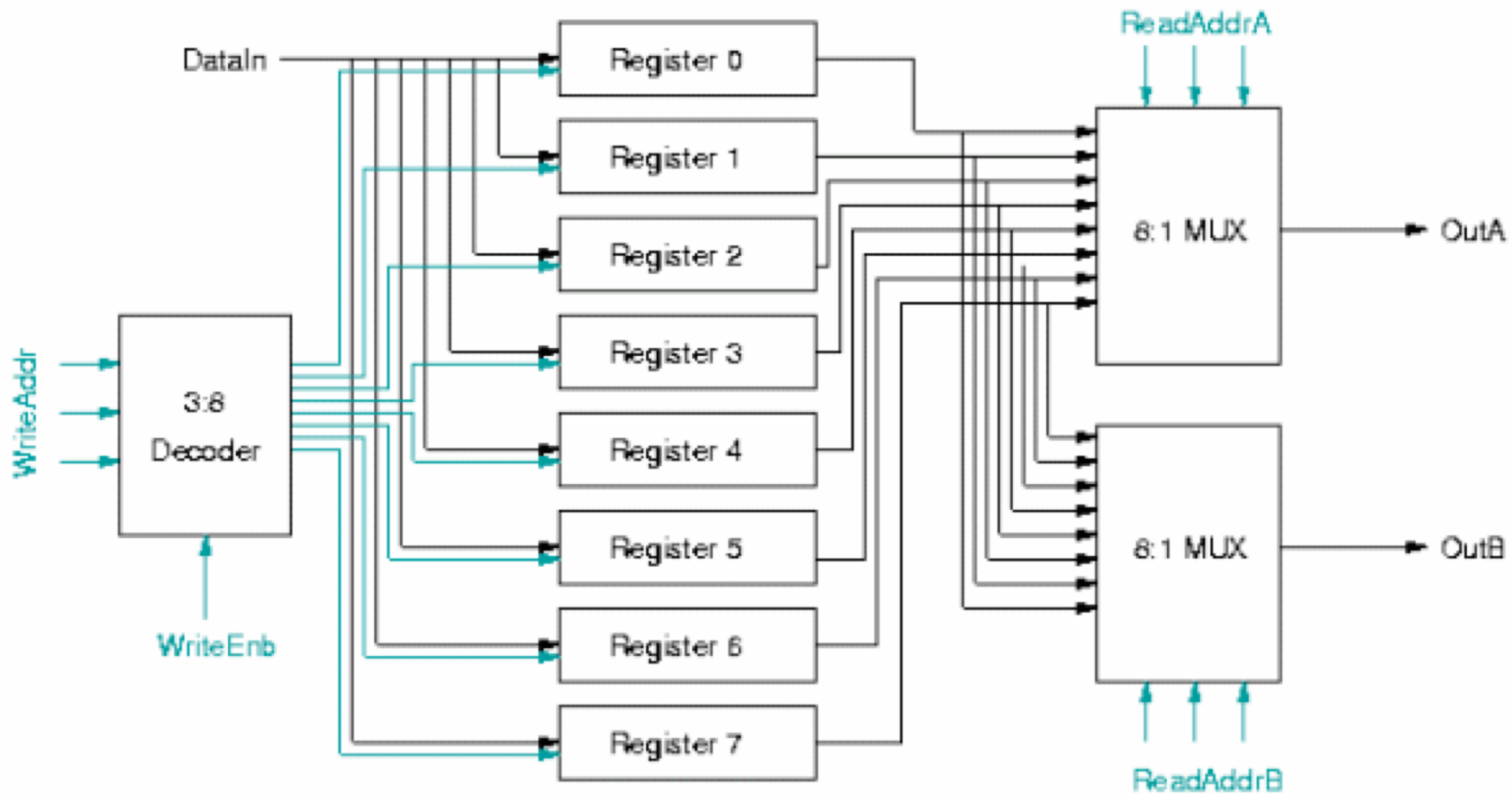


Putting it all Together



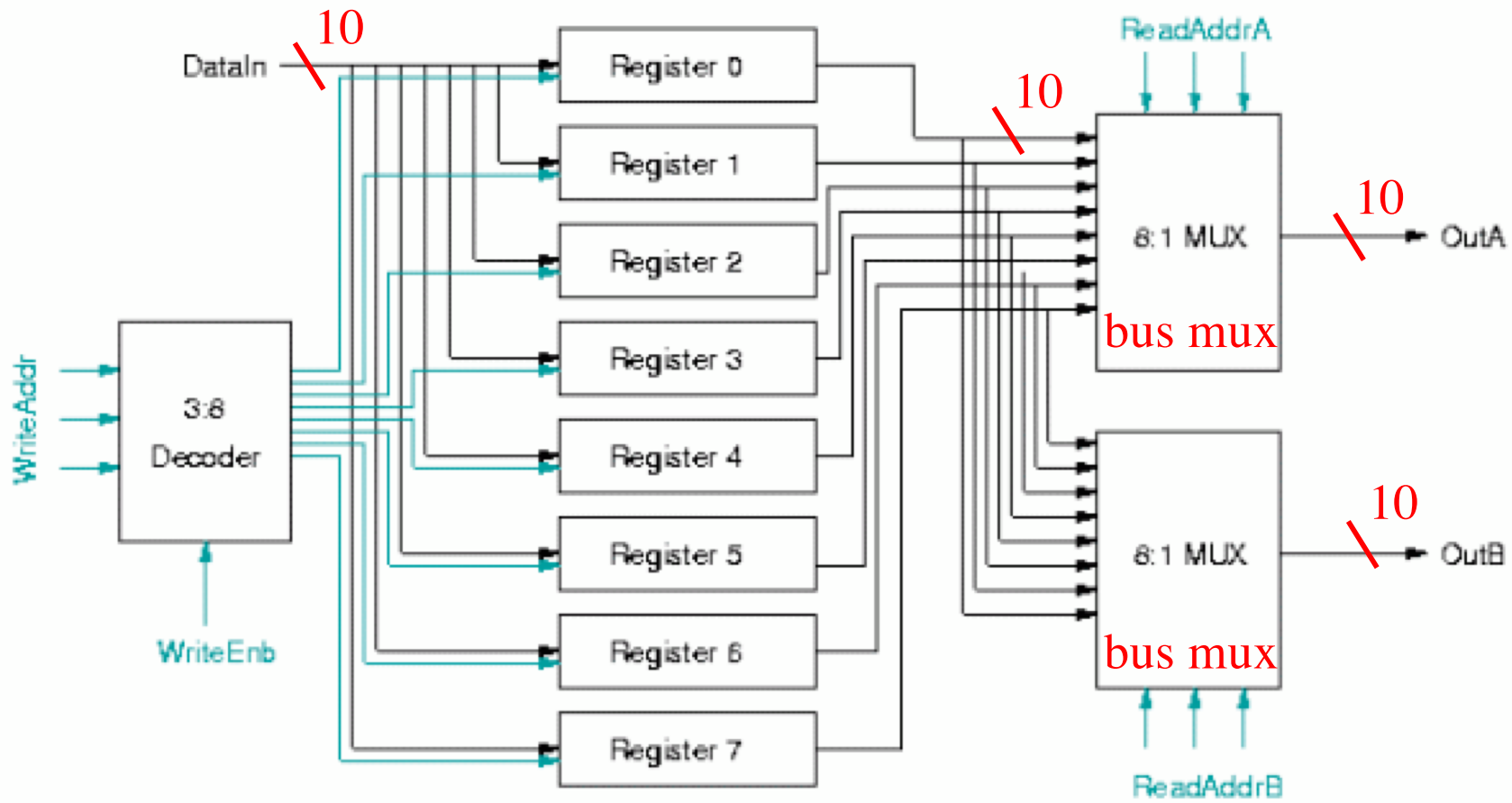
Register file with eight 10-bit registers and two read ports

Register file with 2 read ports



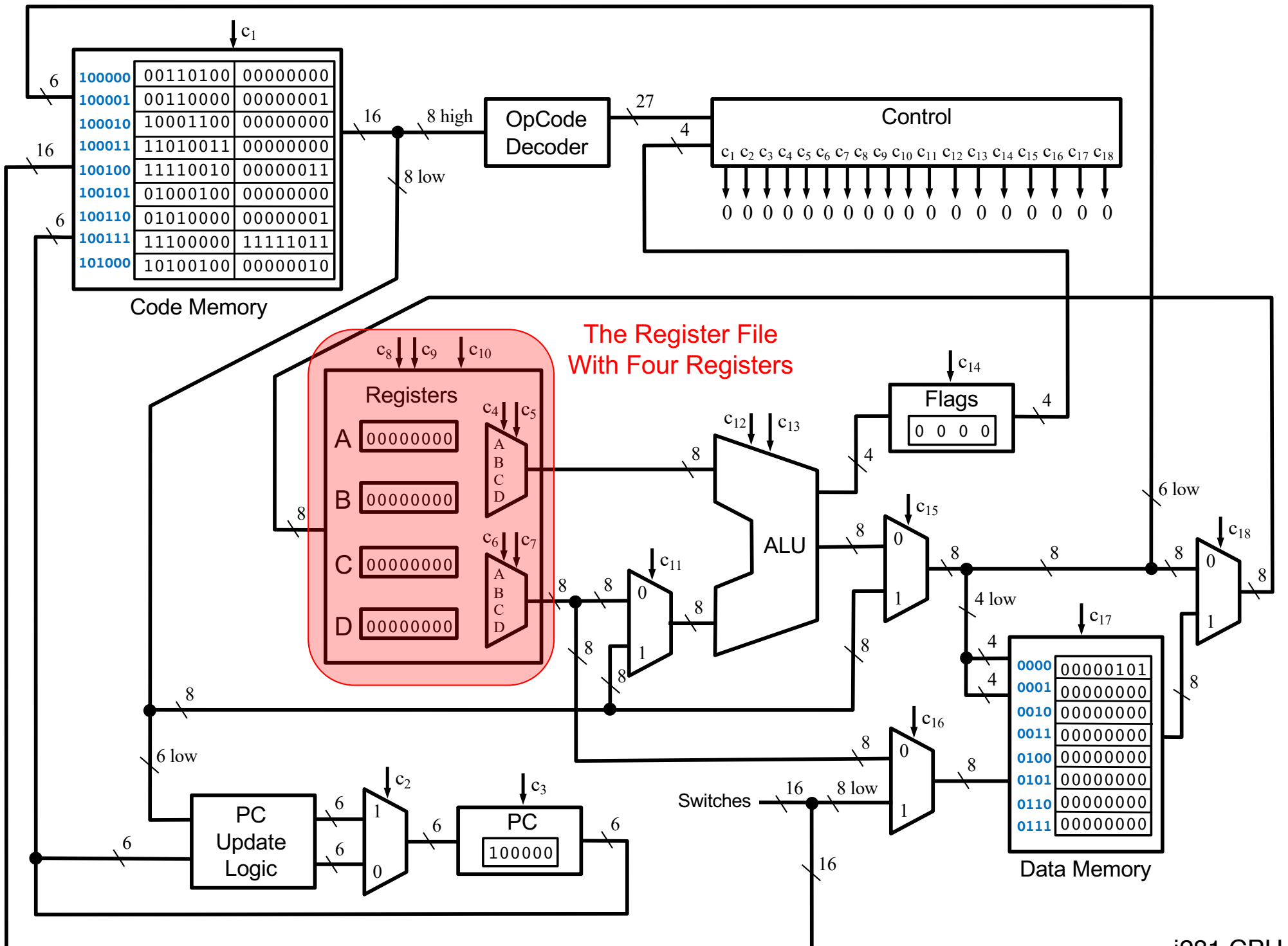
Gray lines are 1-bit signals
Black lines are 10-bit signals

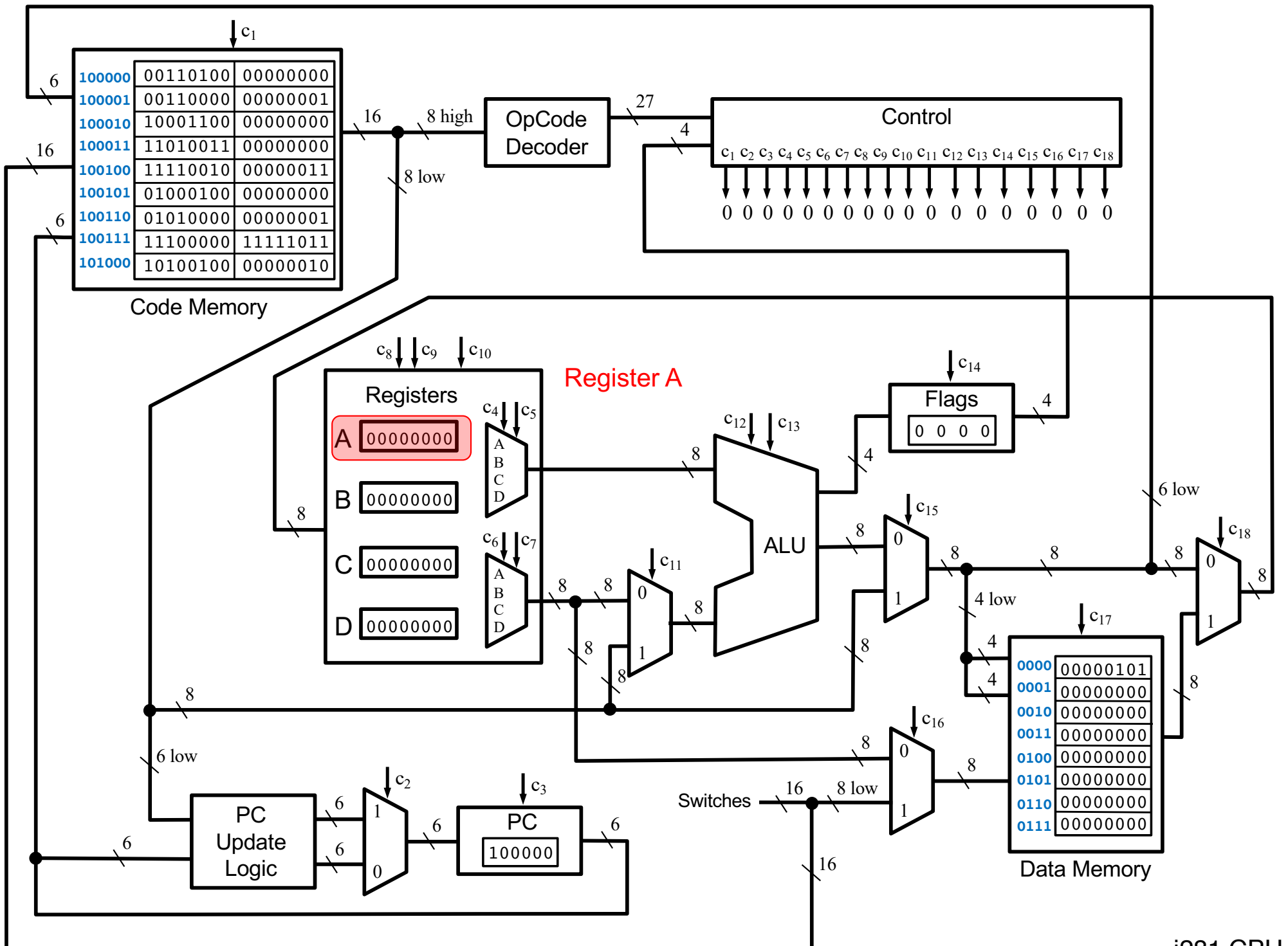
Register file with 2 read ports

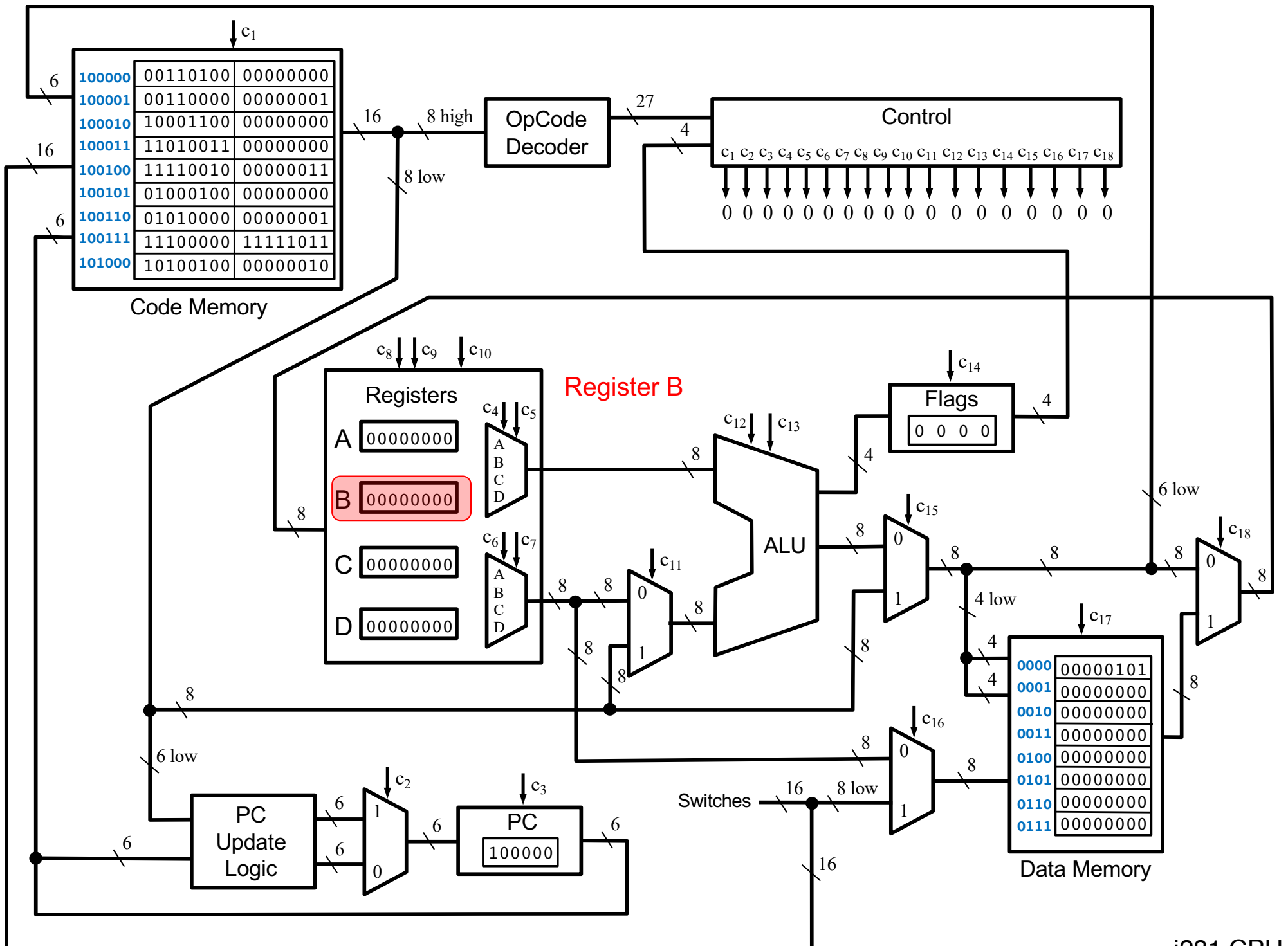


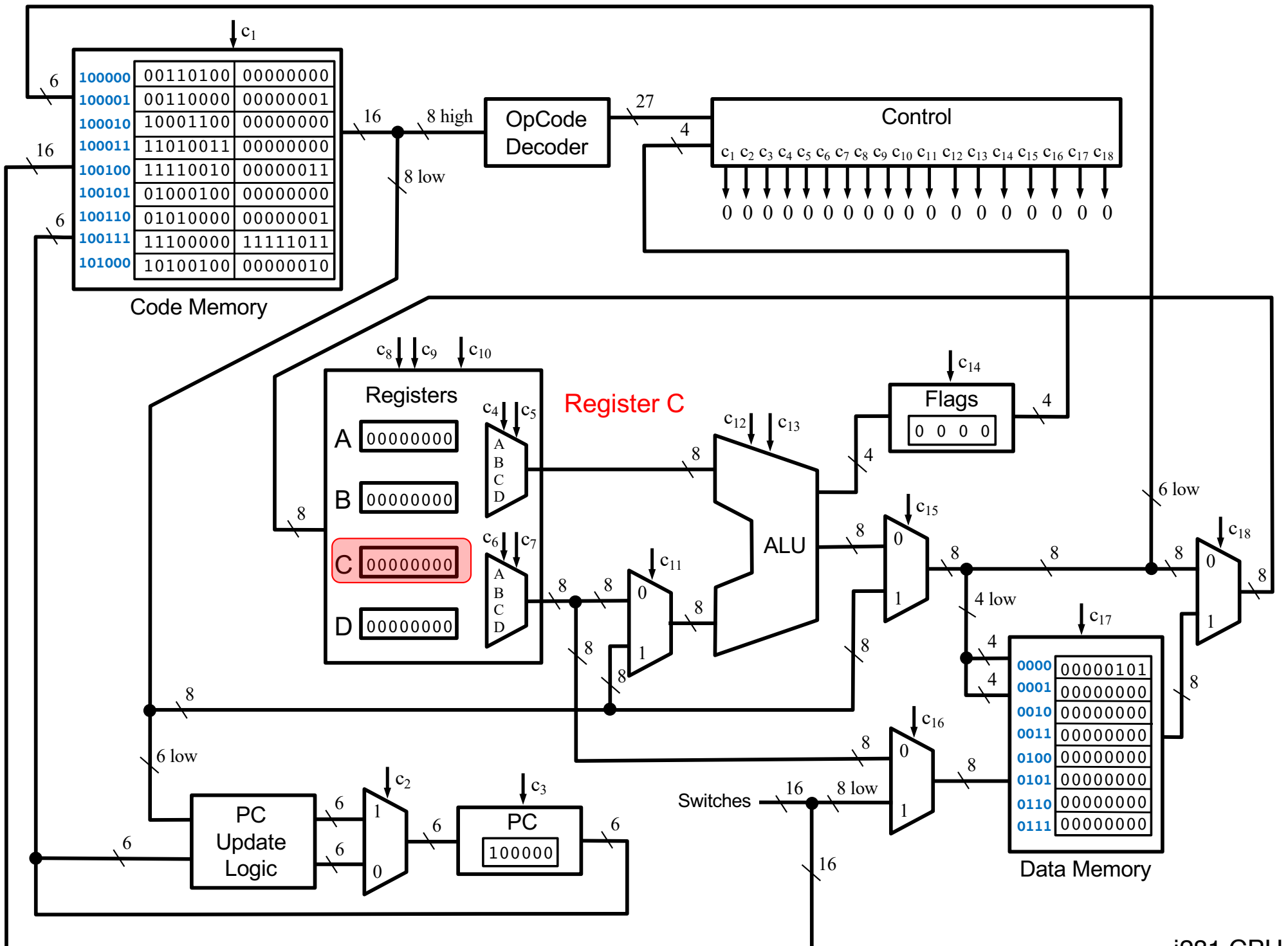
Gray lines are 1-bit signals
Black lines are 10-bit signals

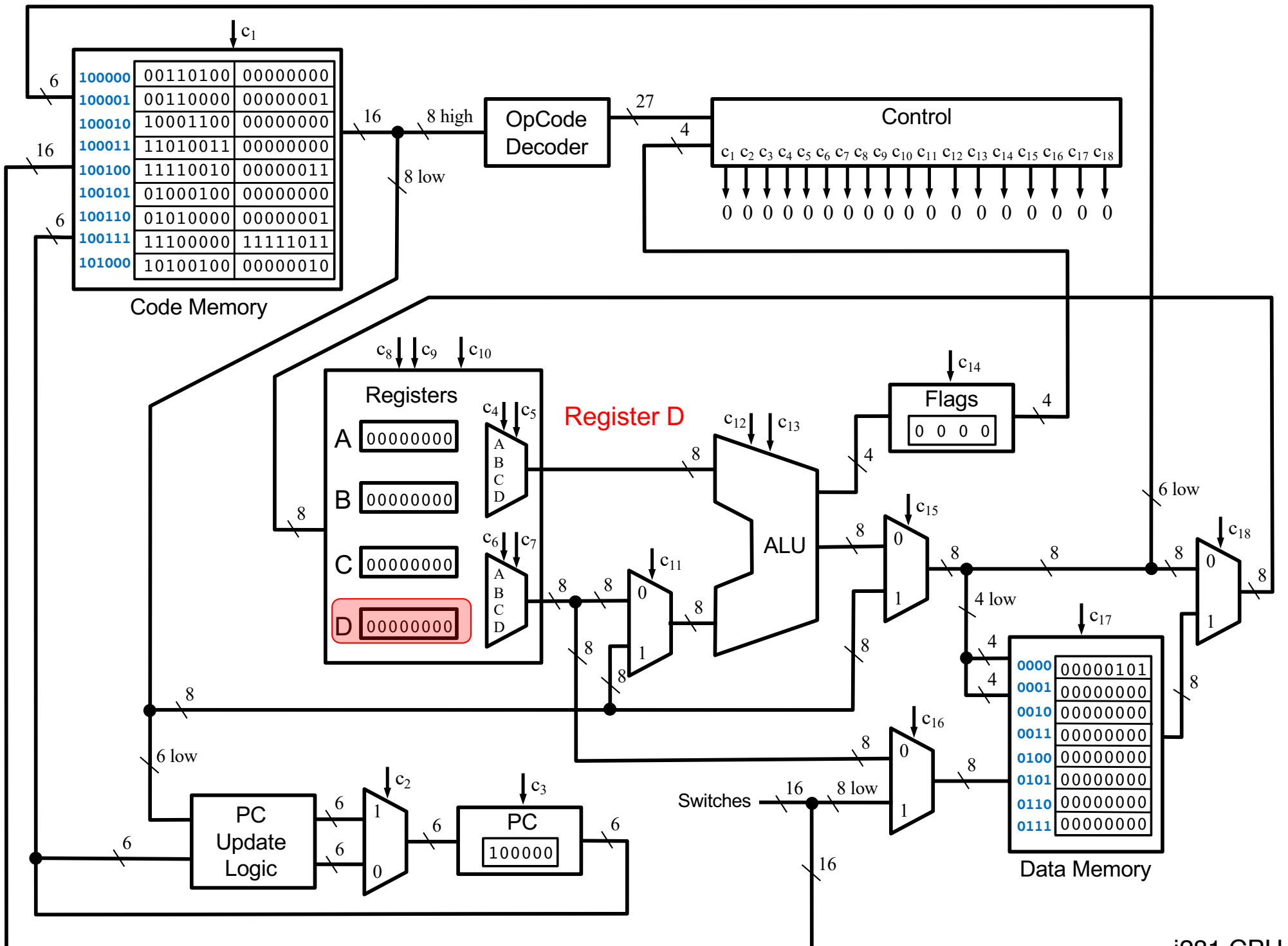
**The i281 CPU has a register file
with four 8-bit registers
and two read ports**



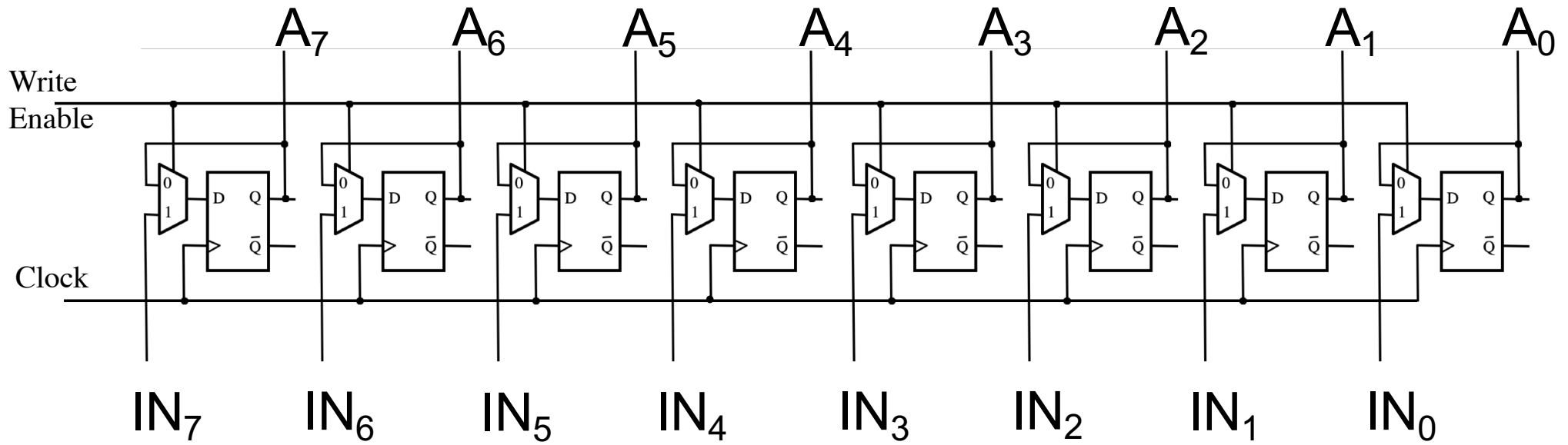






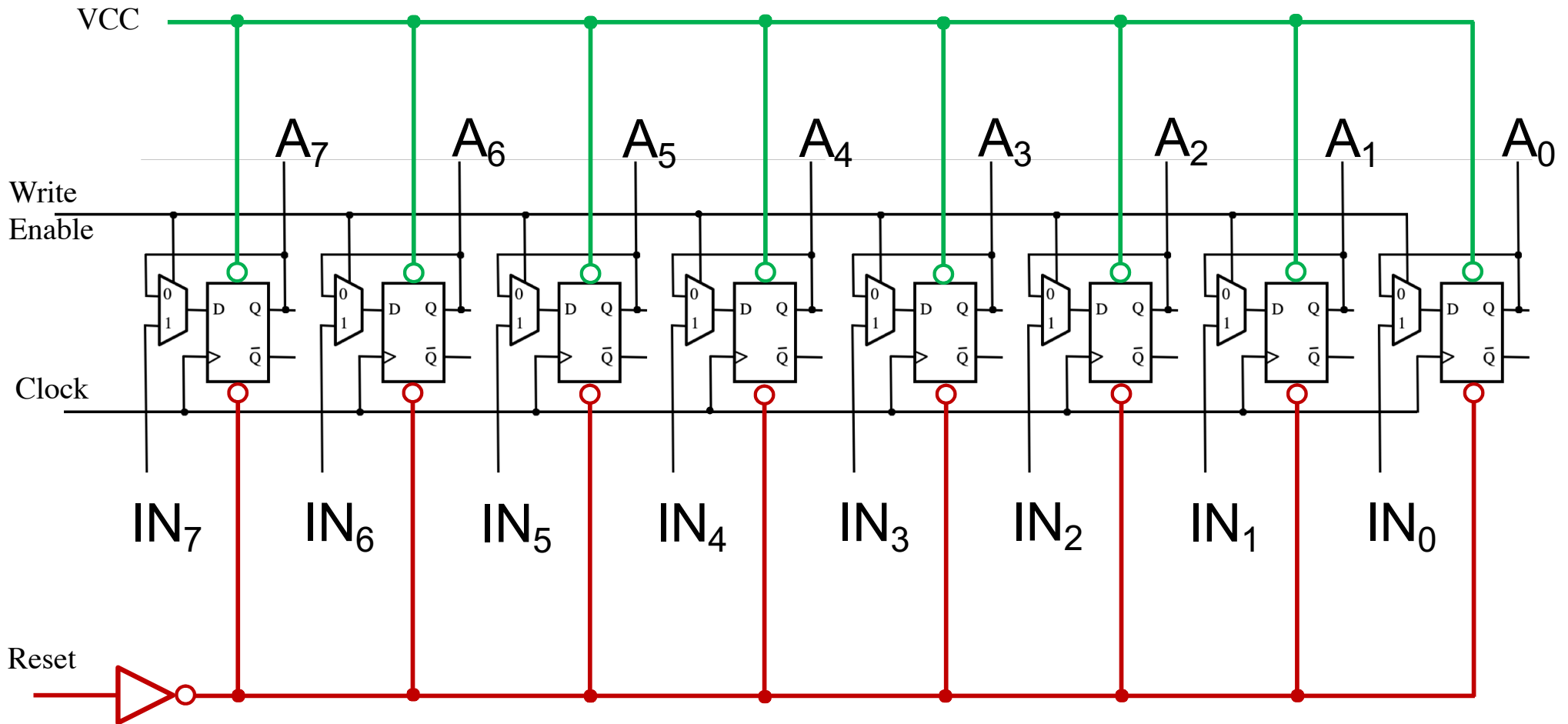


Register A

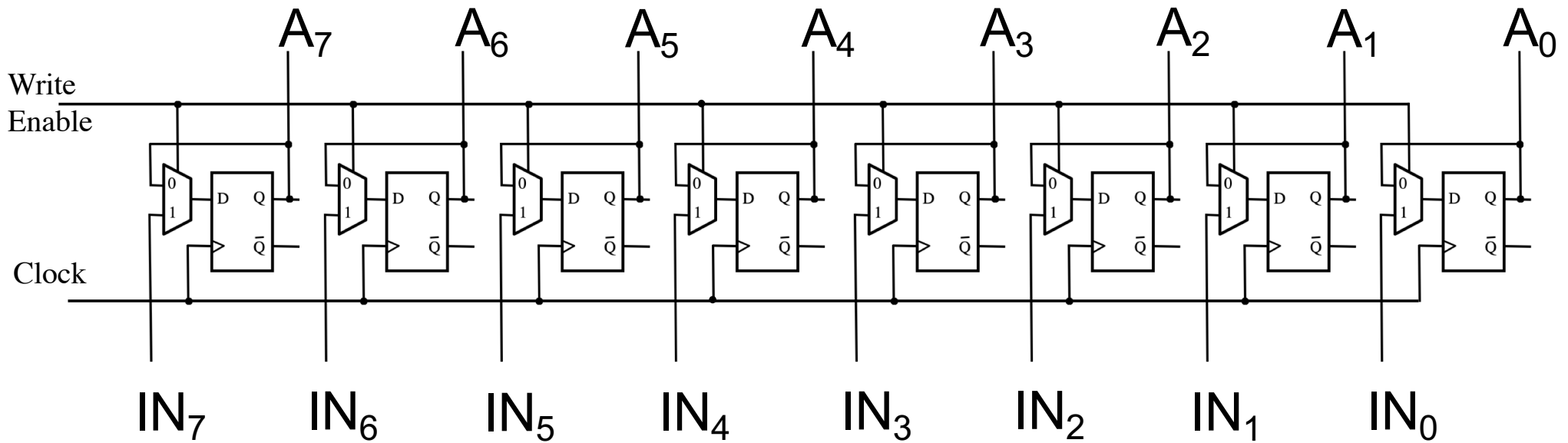


8-Bit Parallel-Access Register

Register A

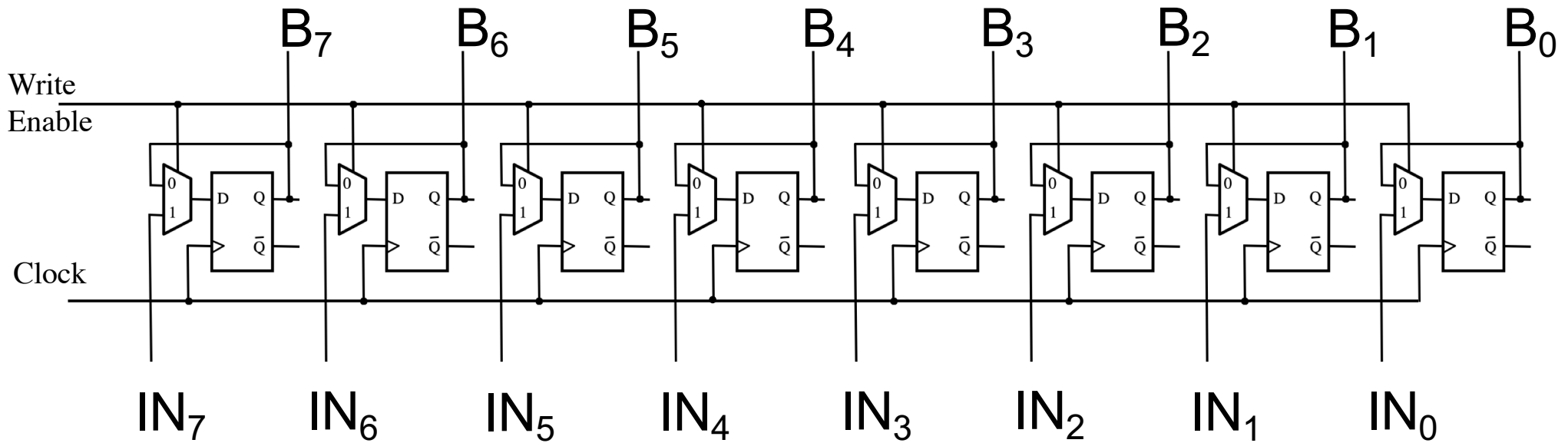


Register A



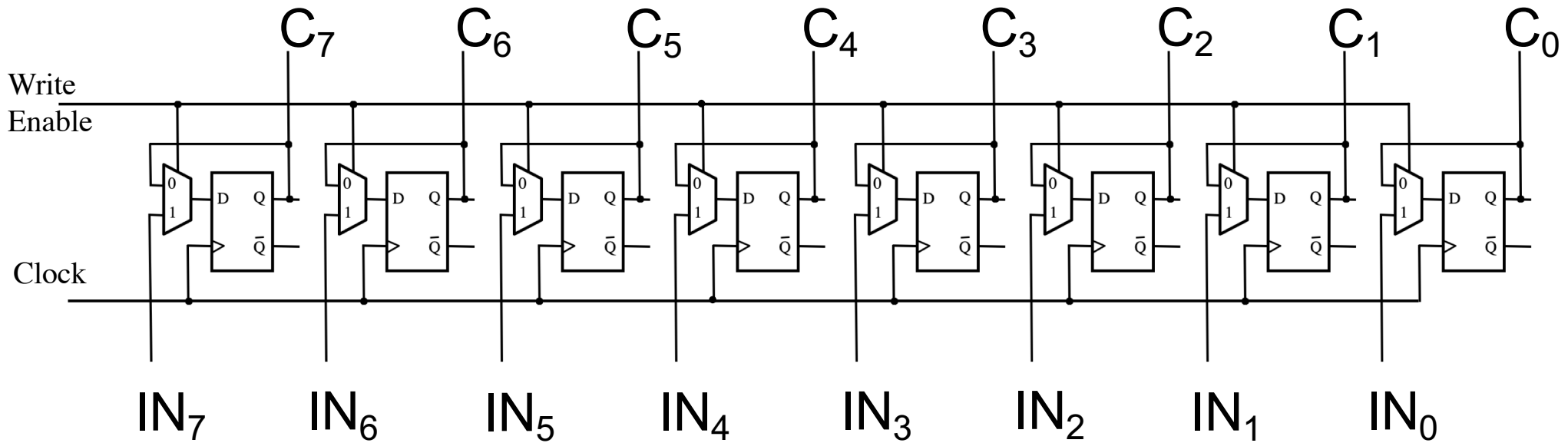
8-Bit Parallel-Access Register

Register B



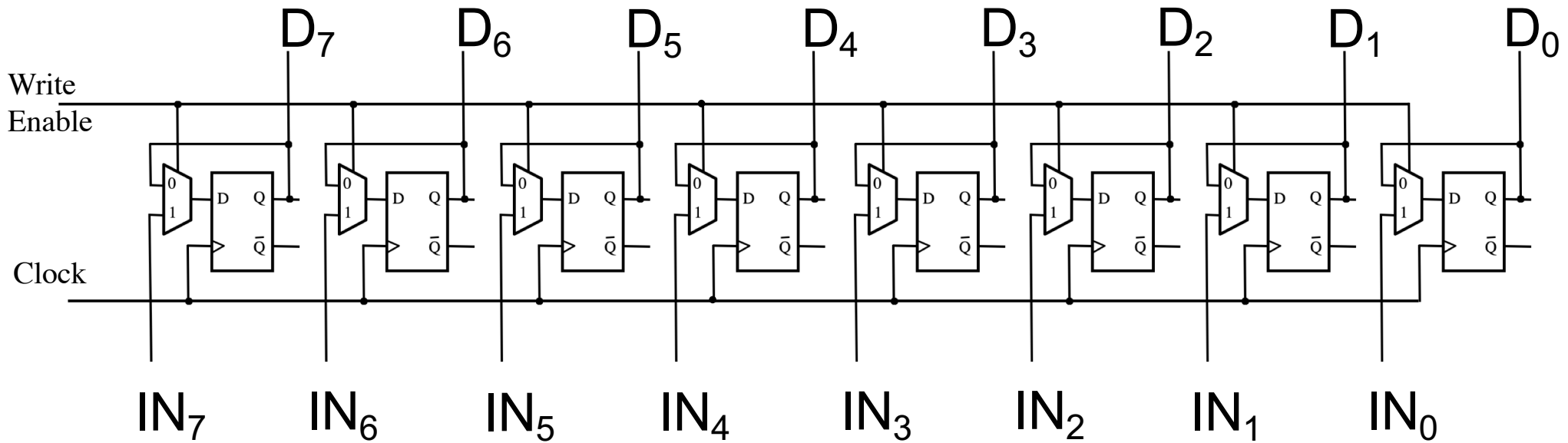
8-Bit Parallel-Access Register

Register C

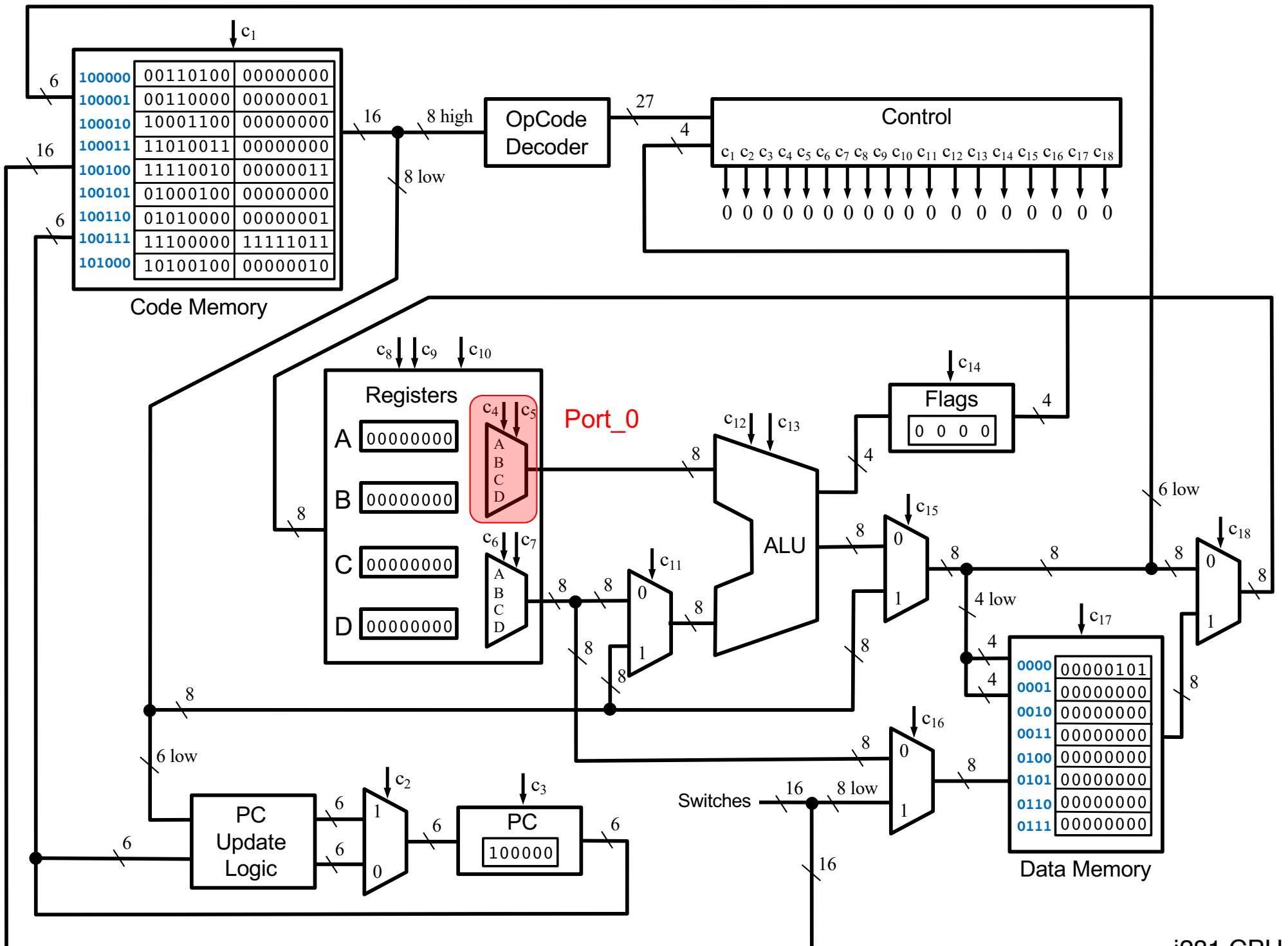


8-Bit Parallel-Access Register

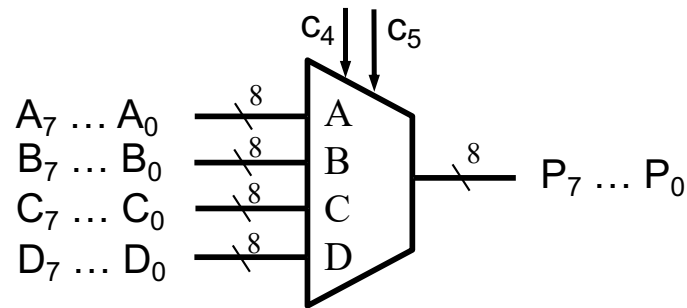
Register D

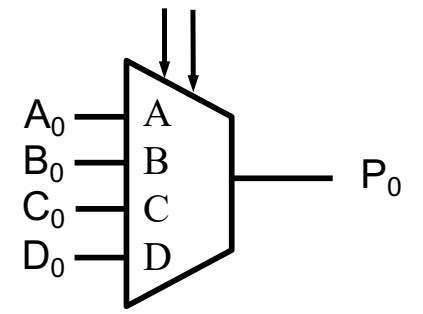


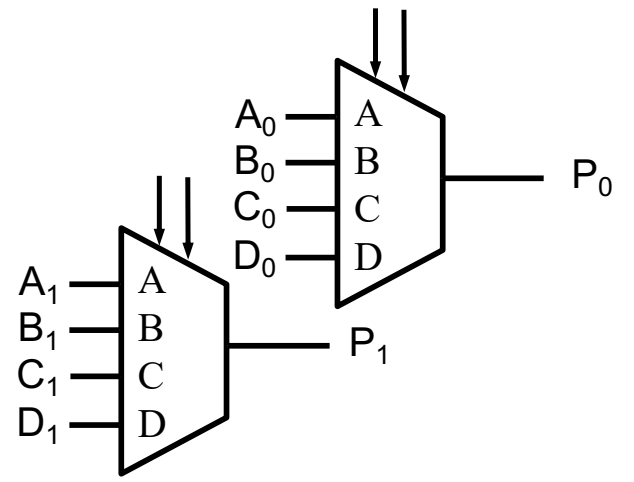
8-Bit Parallel-Access Register

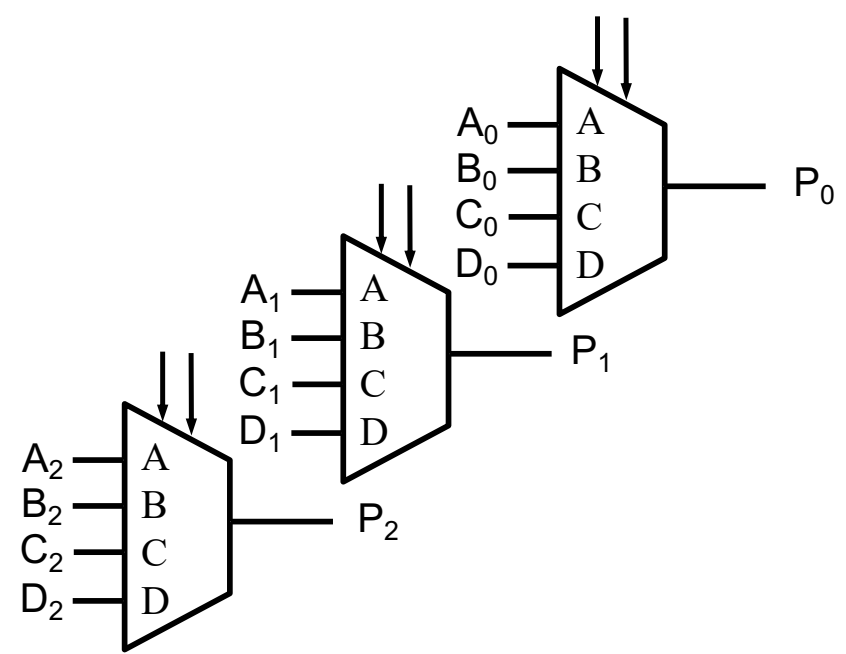


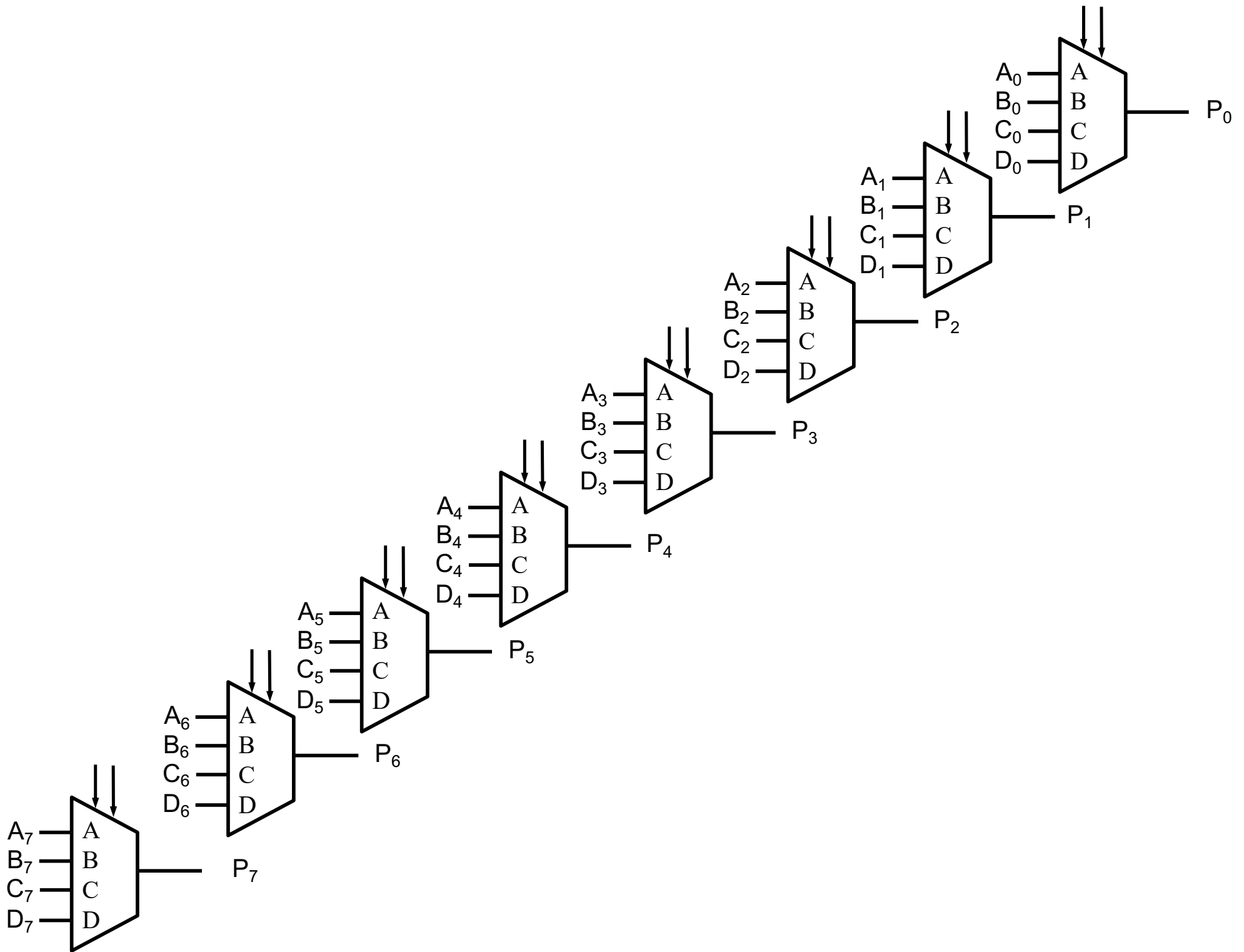
4-to-1 Bus Multiplexer (with 8-bit lines)

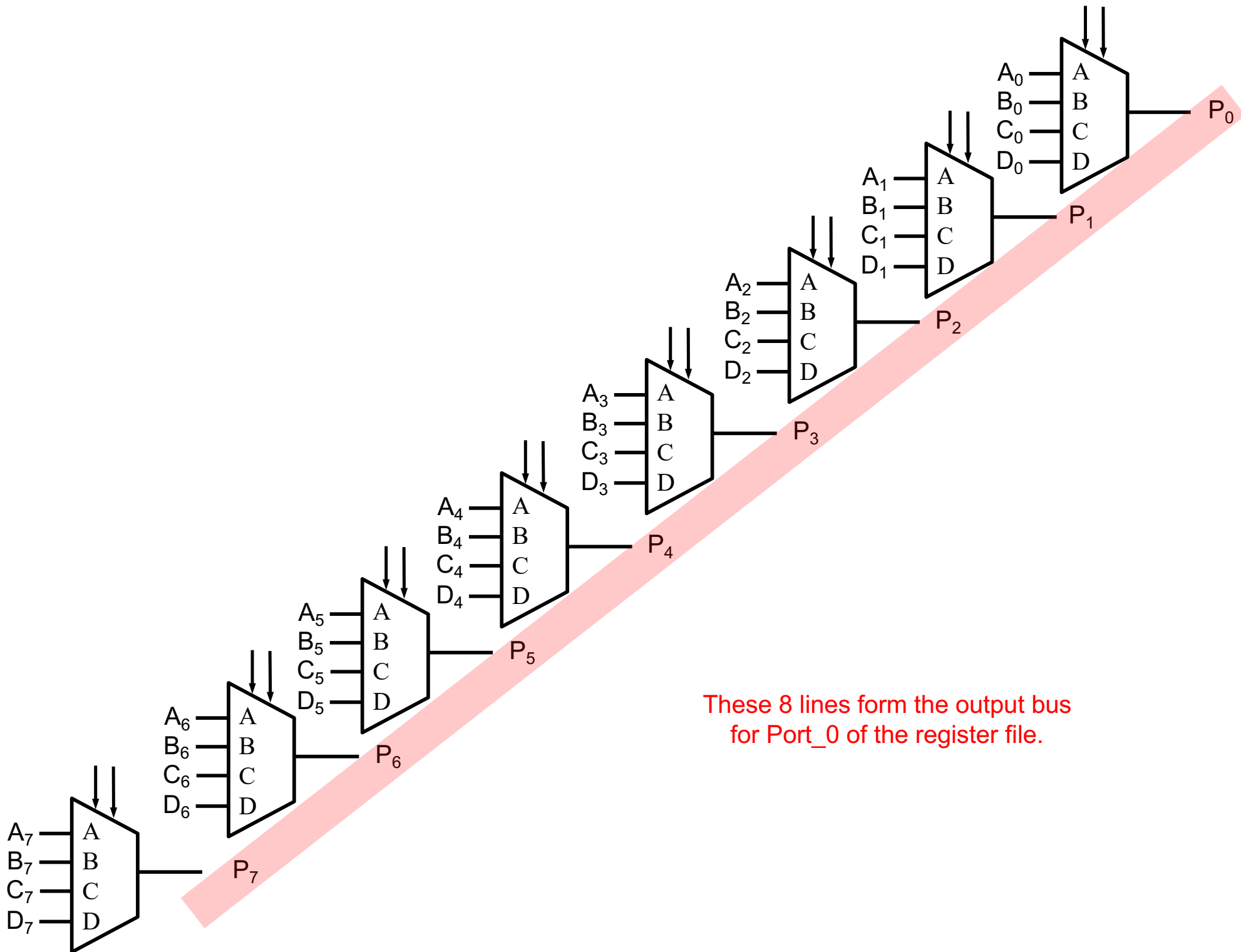


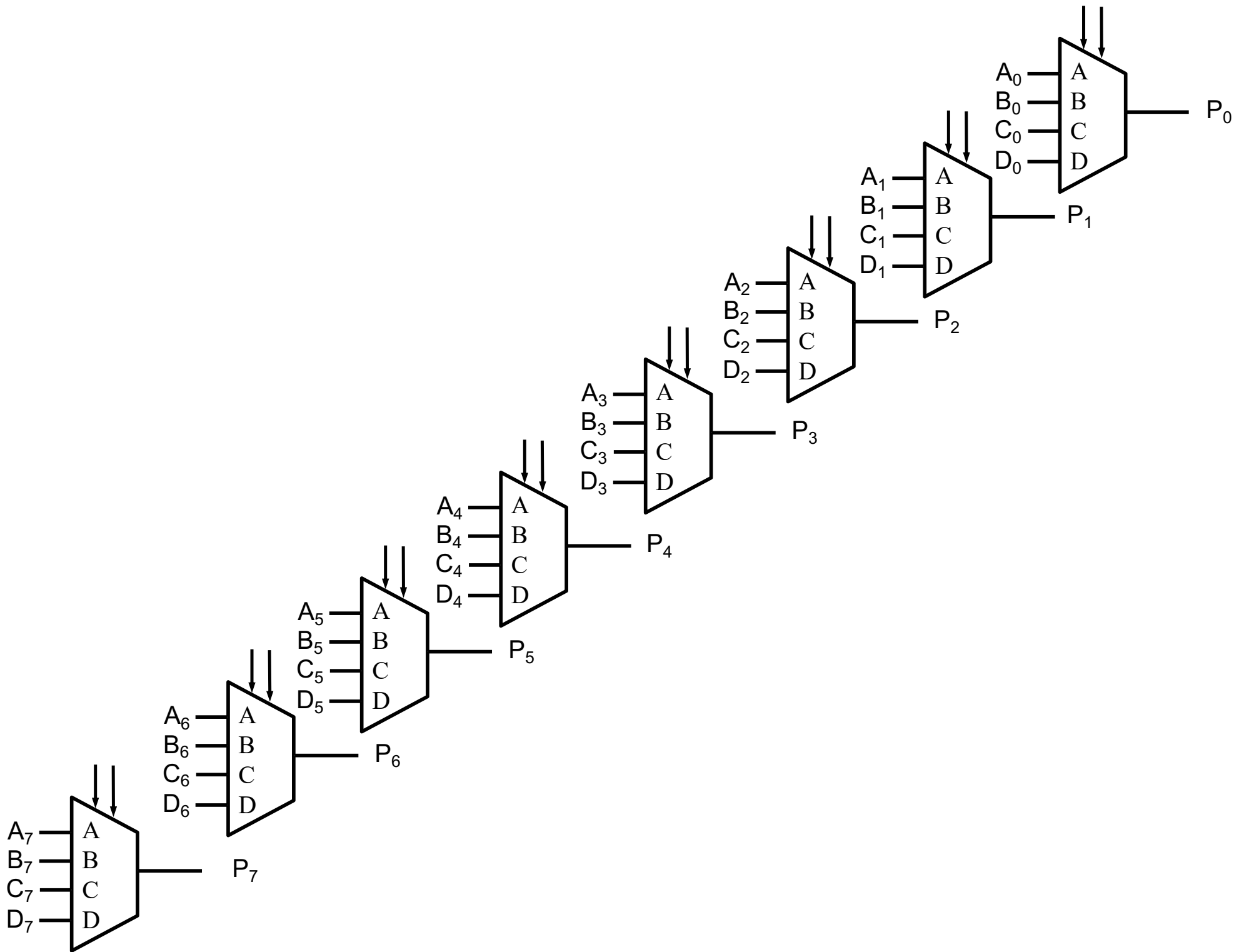


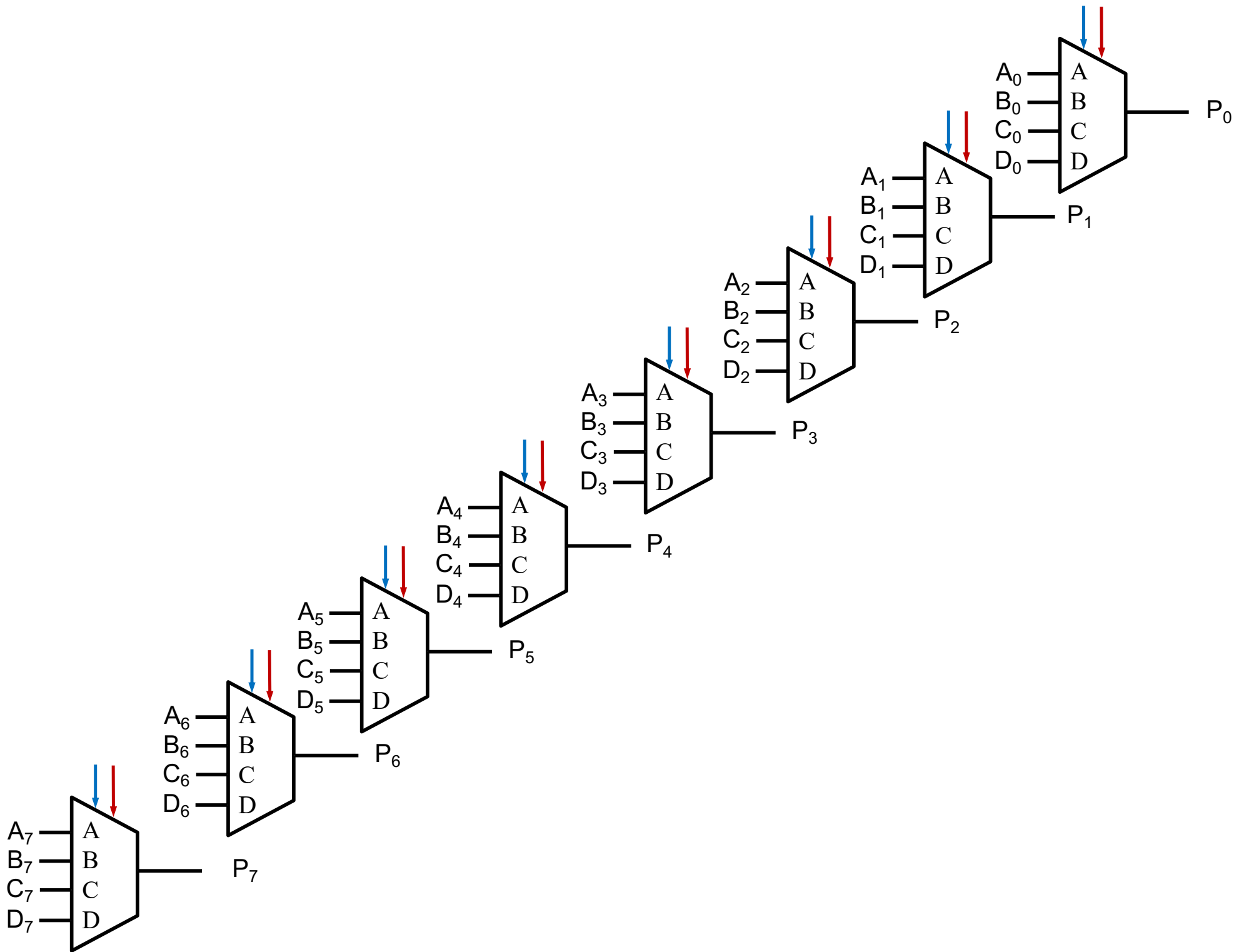


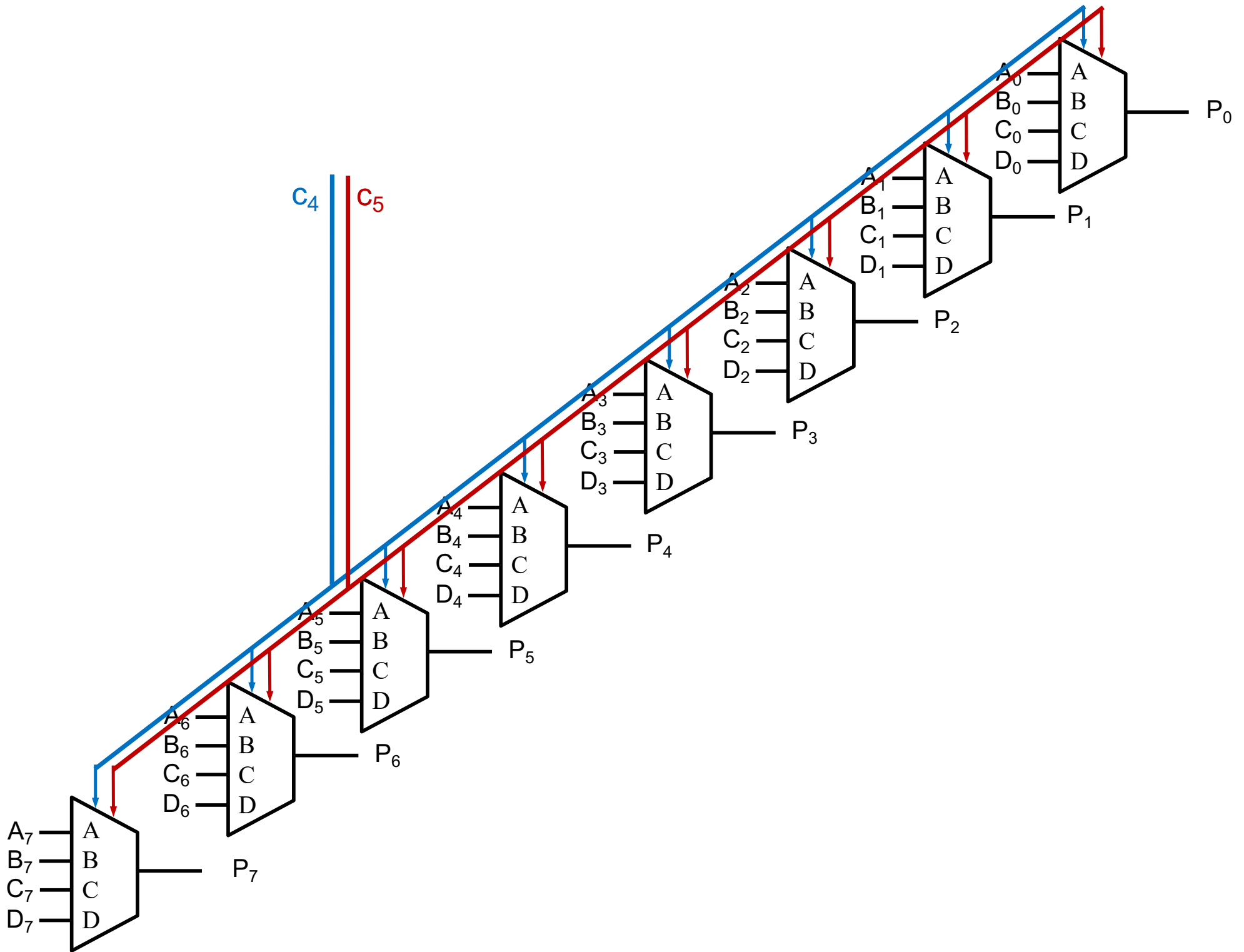


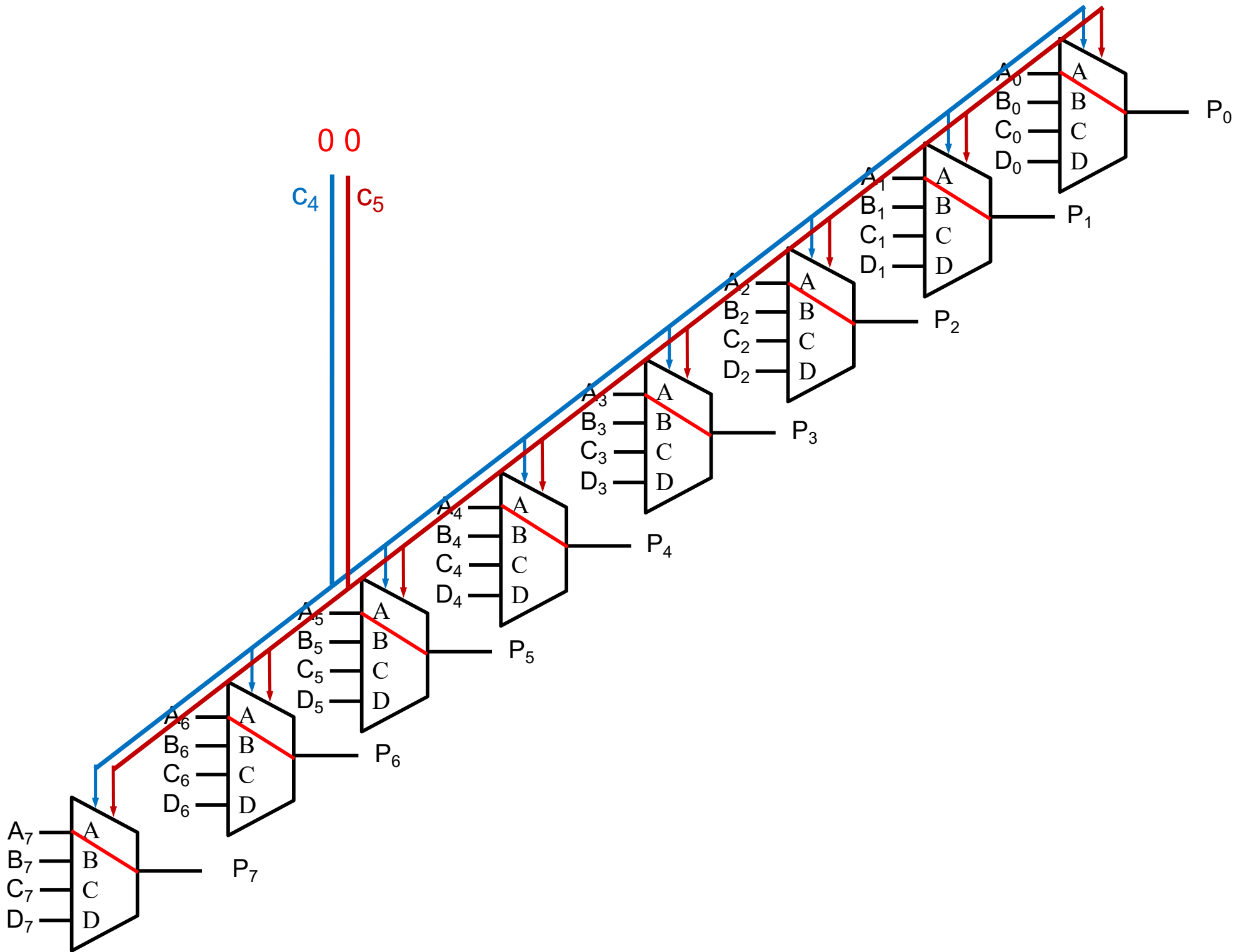


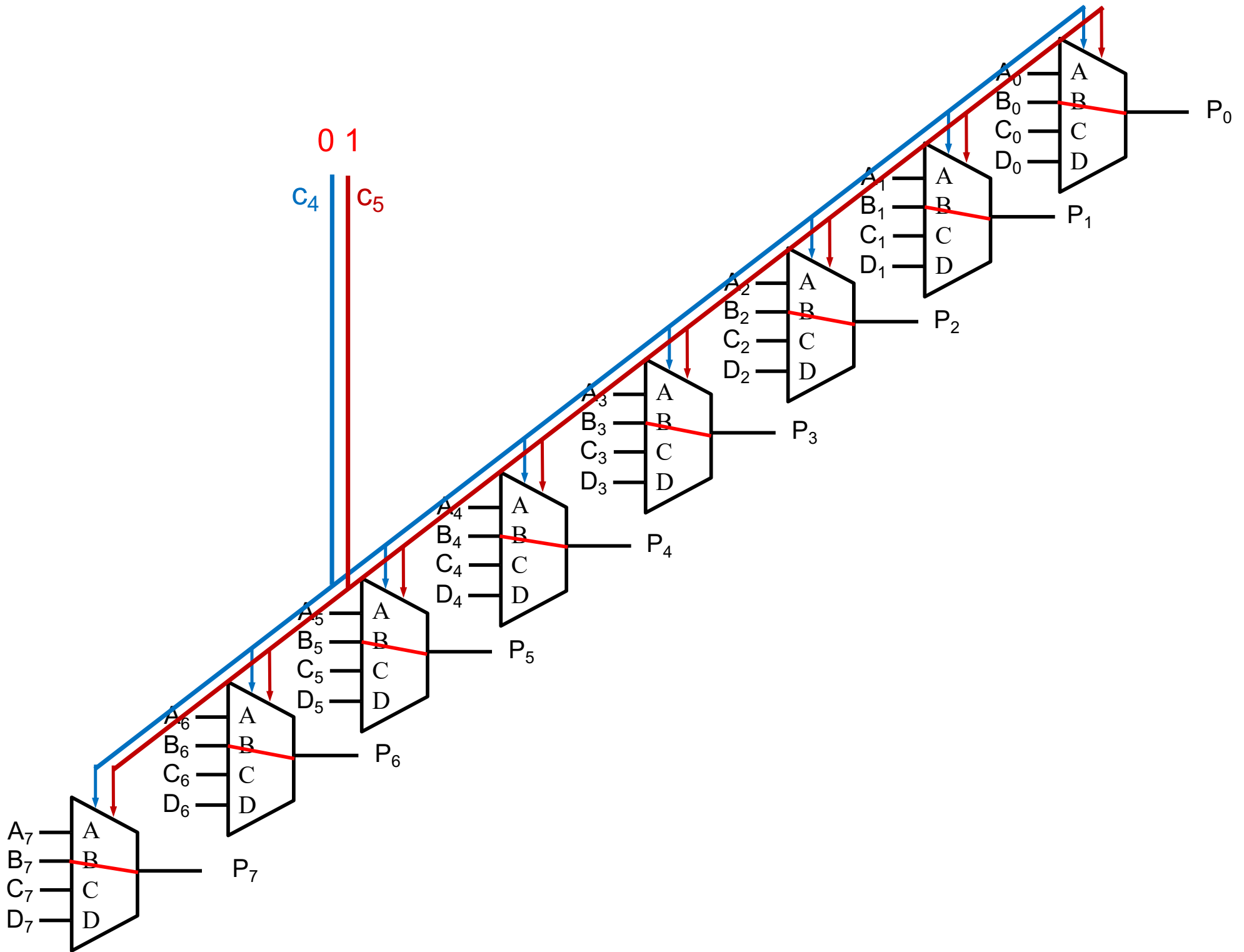


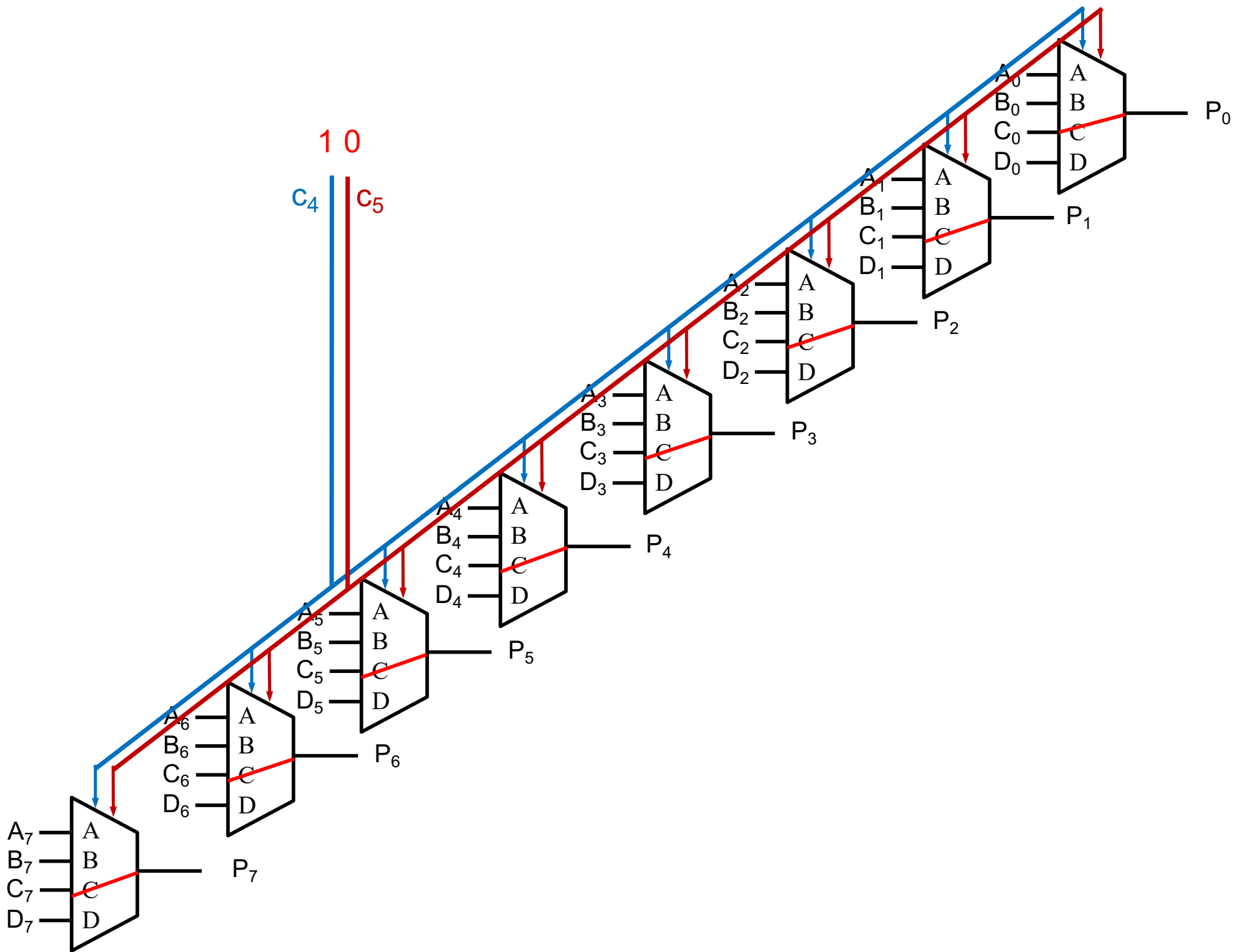


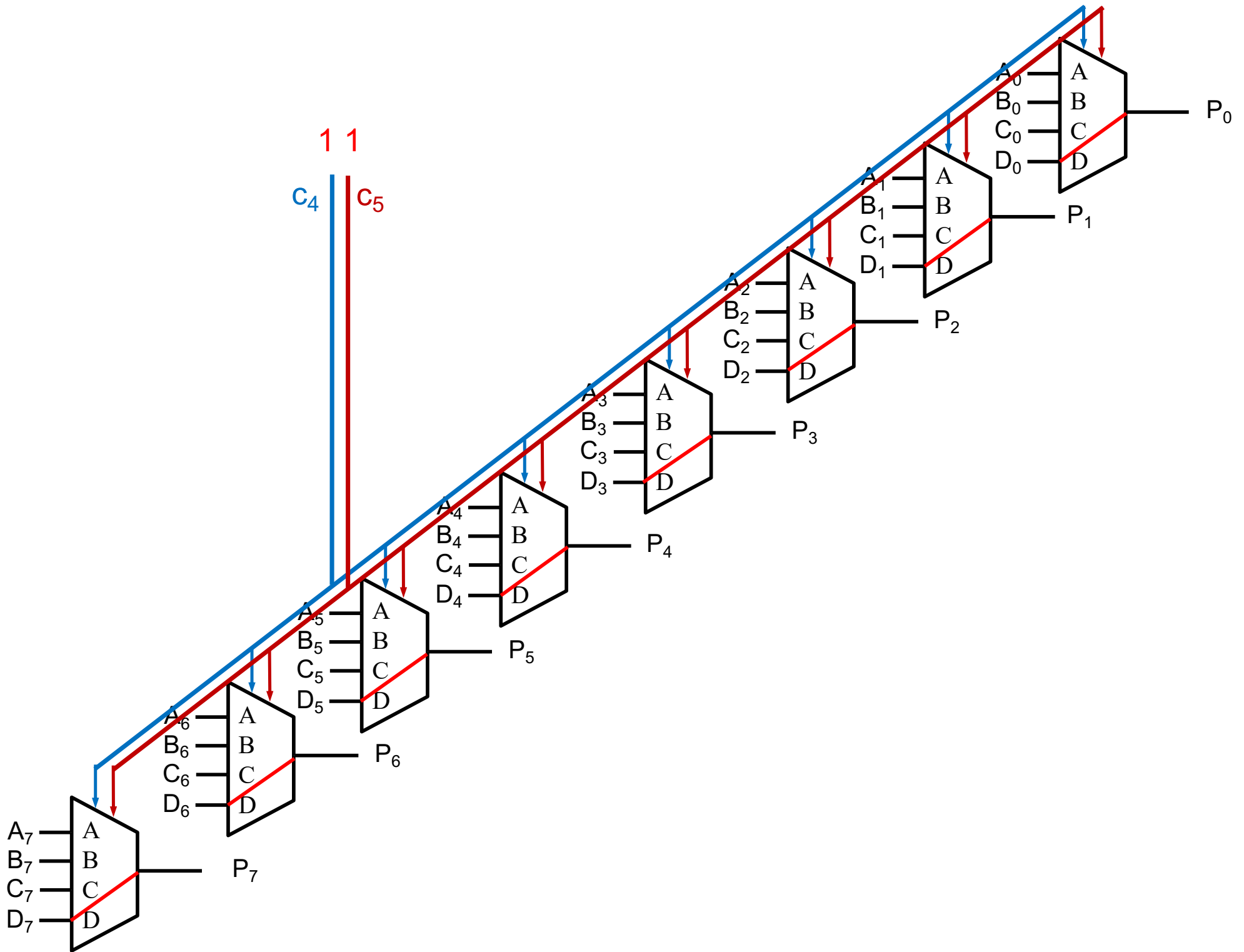


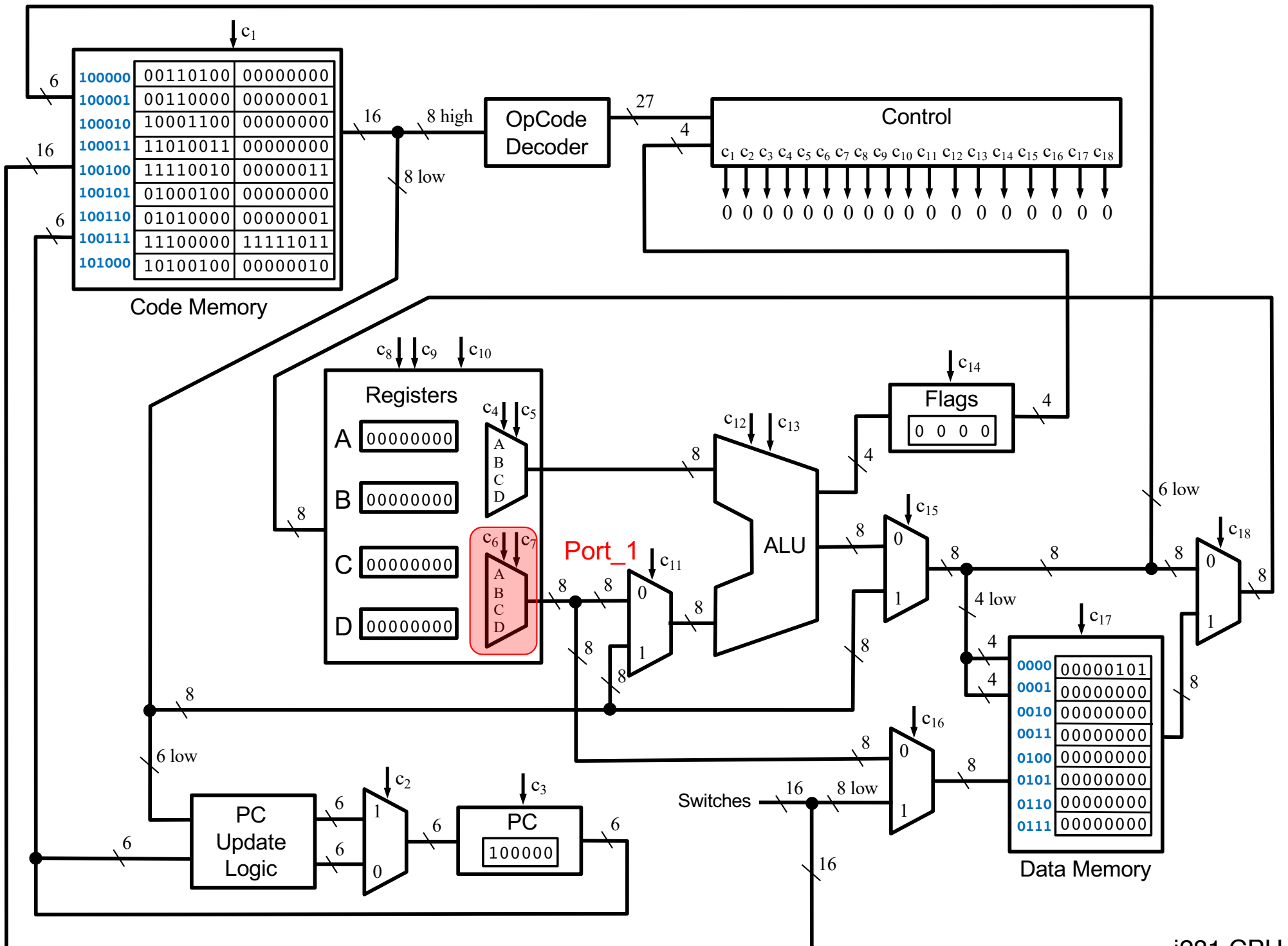




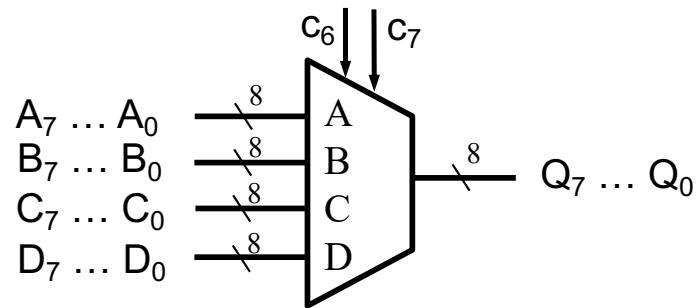


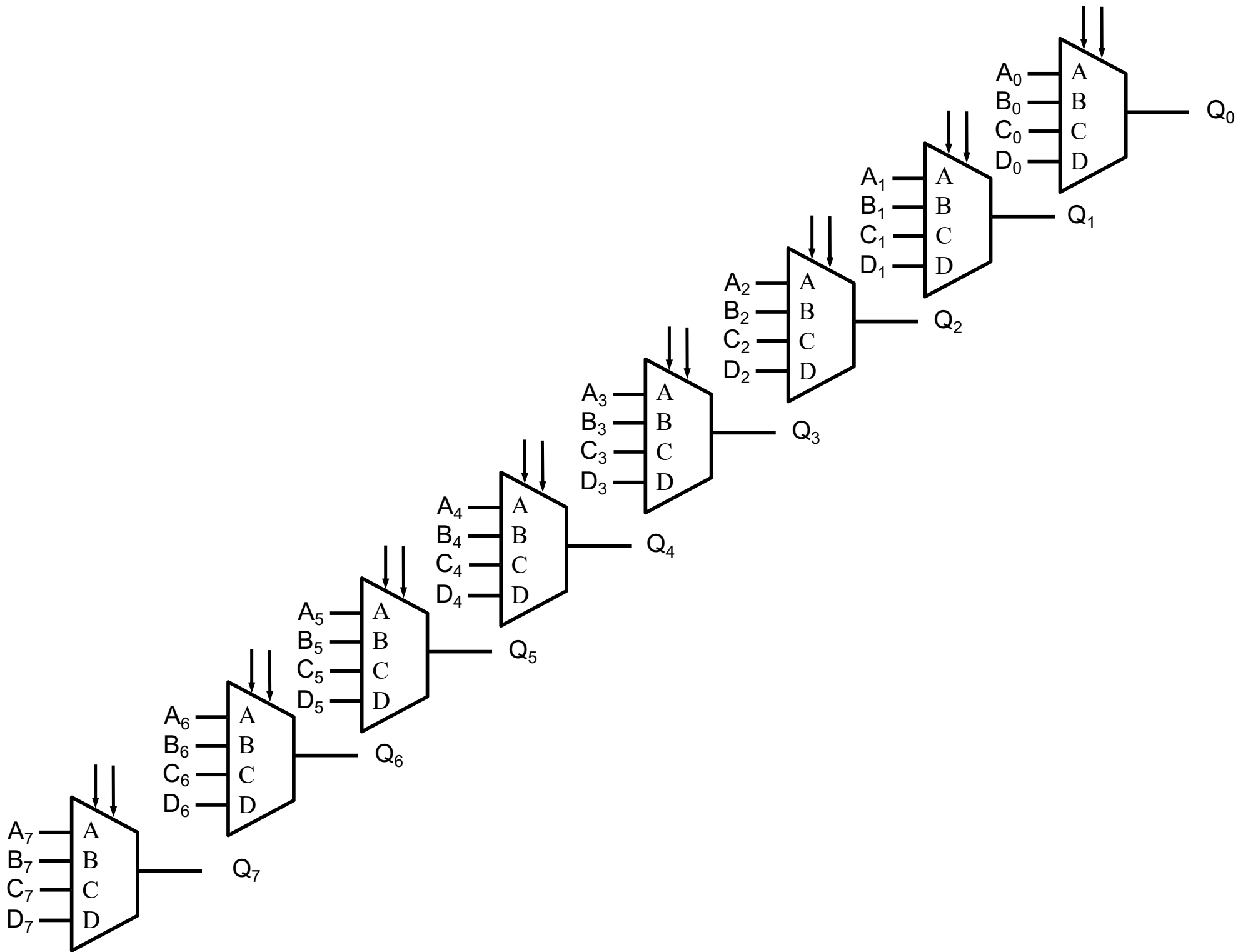


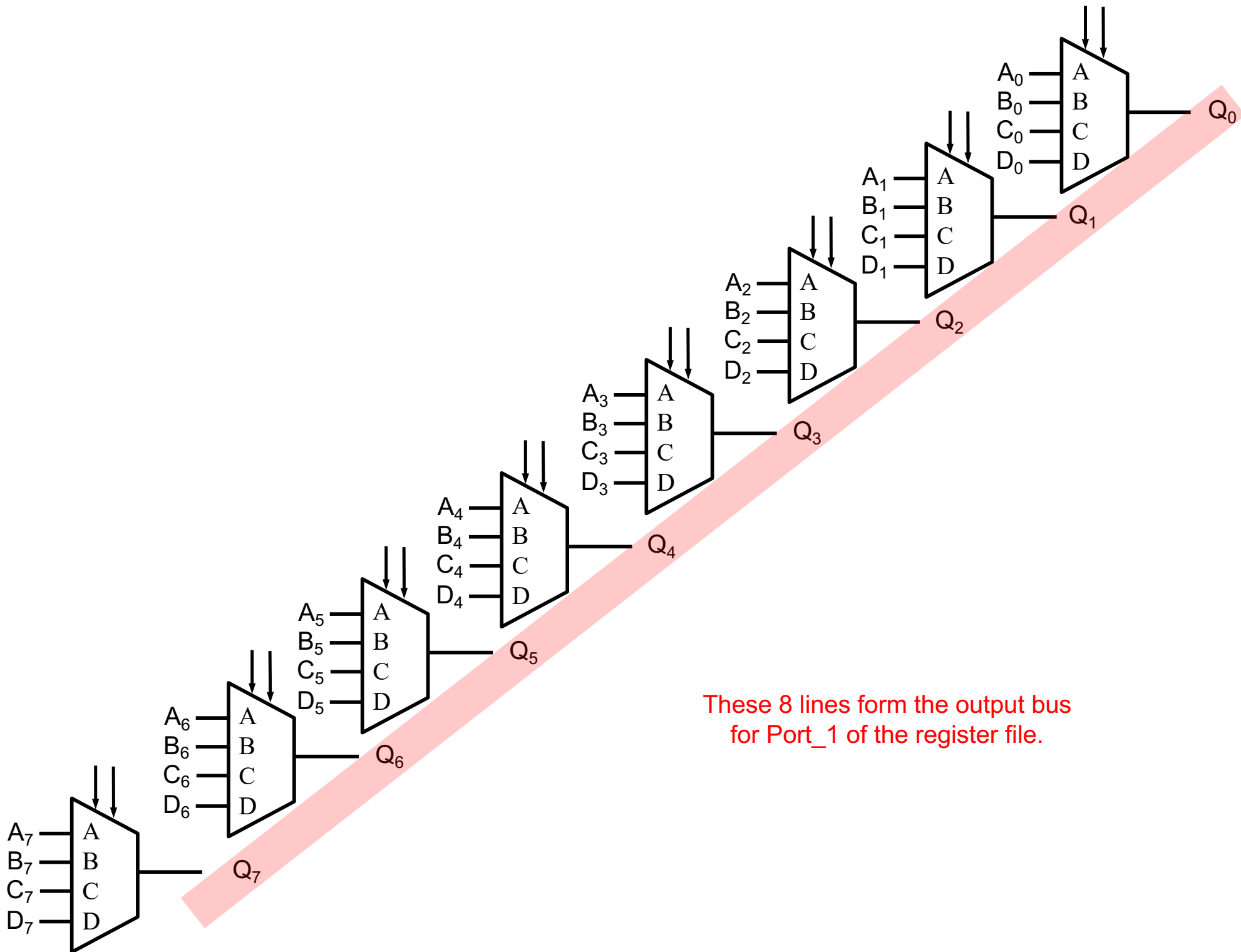




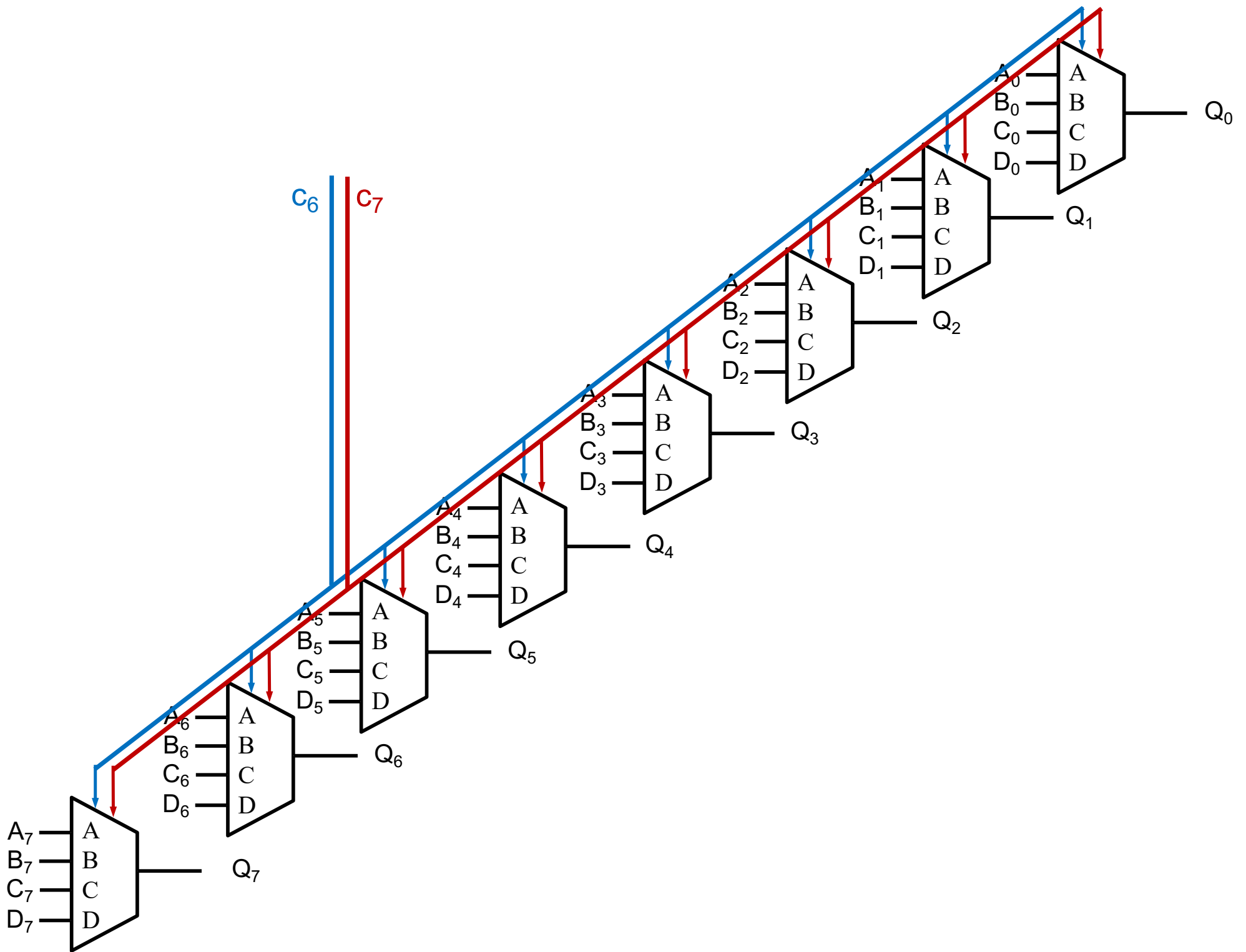
4-to-1 Bus Multiplexer (with 8-bit lines)

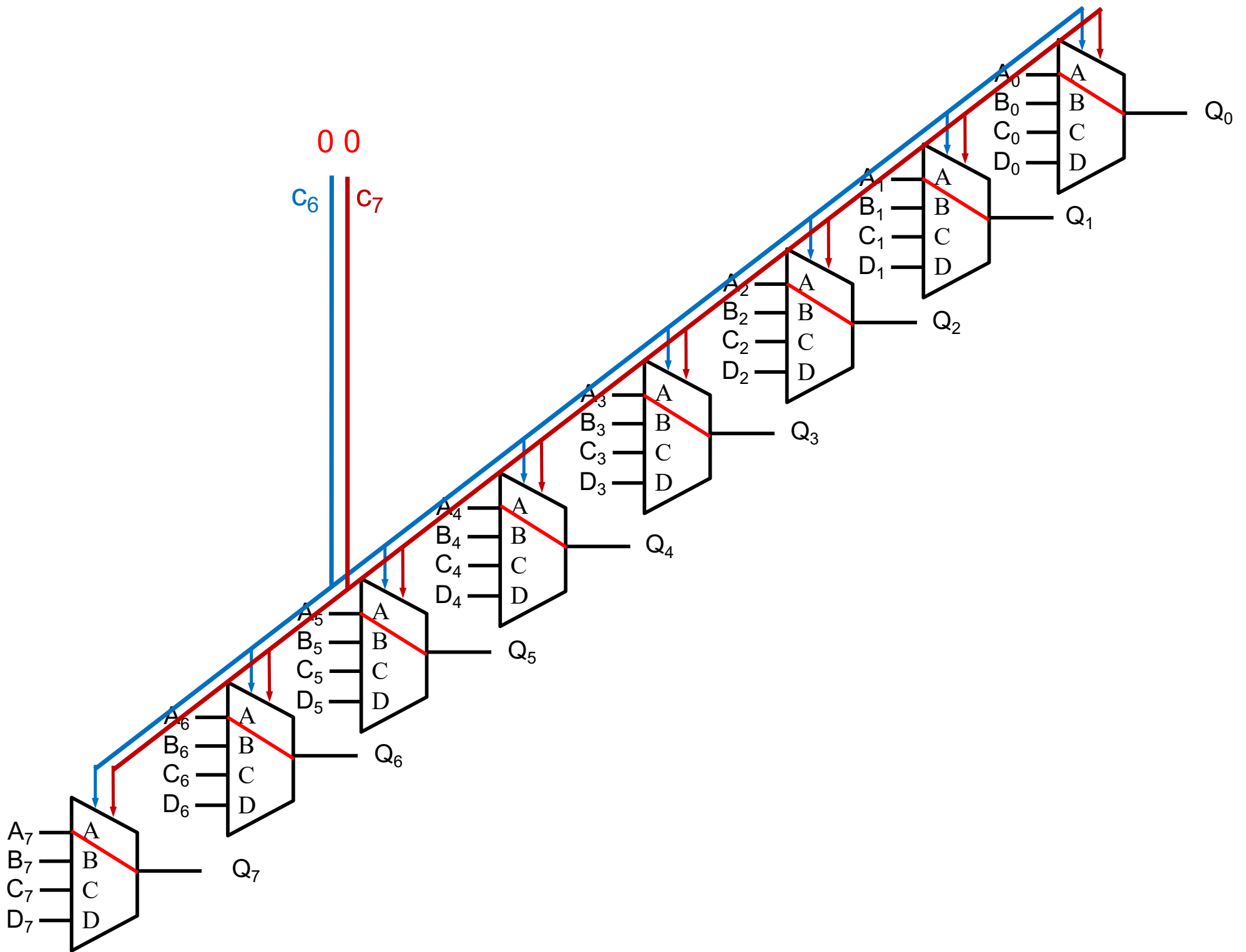


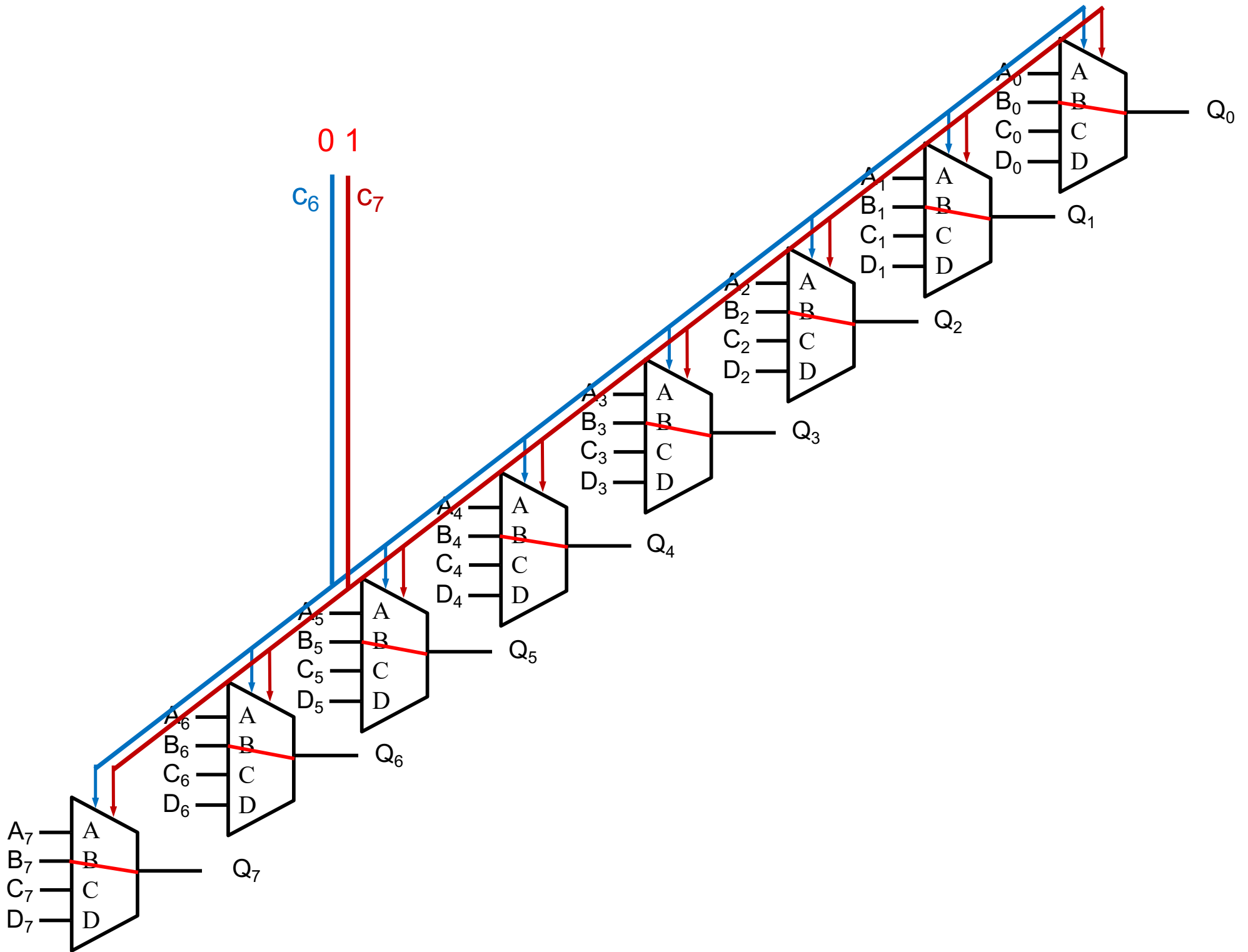


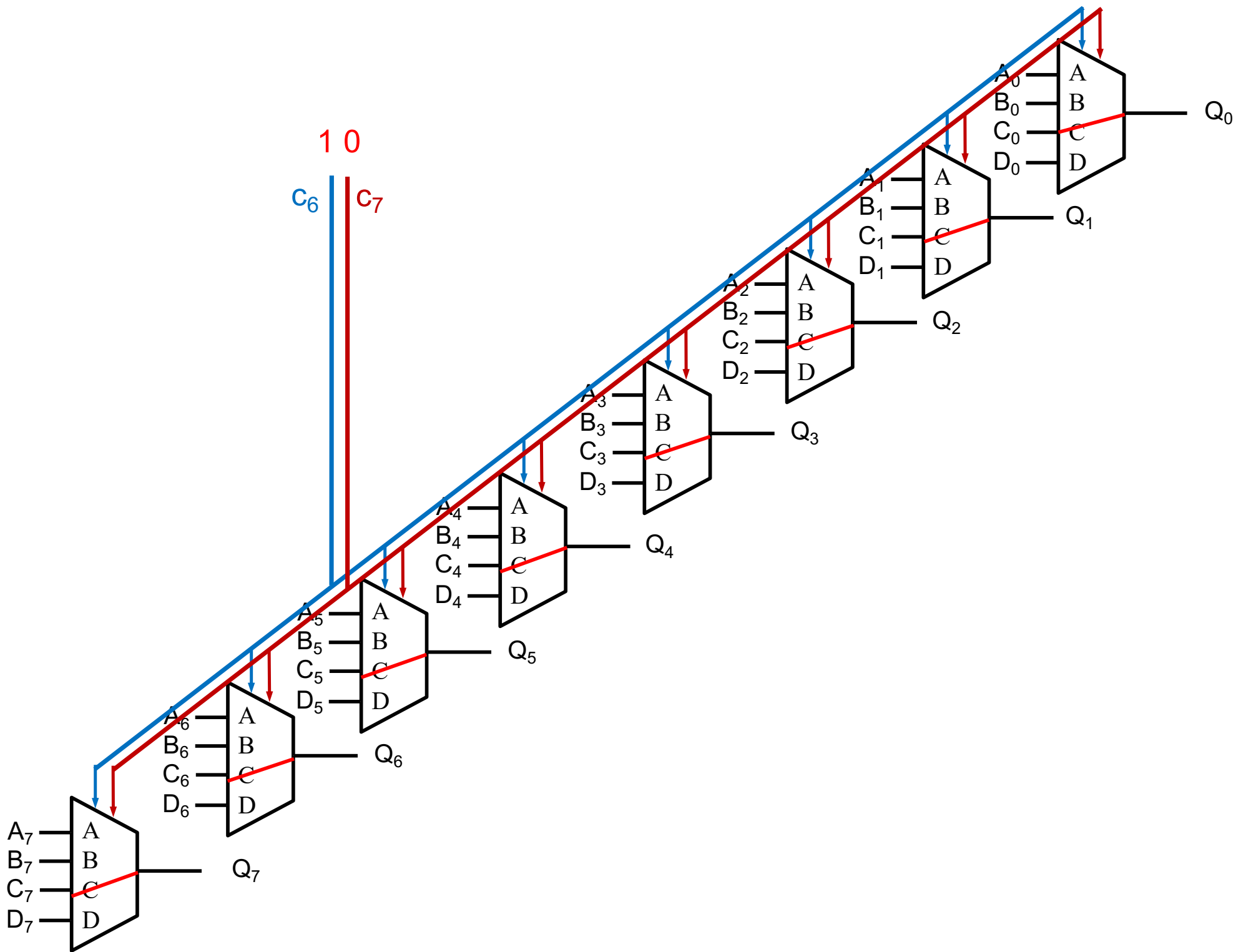


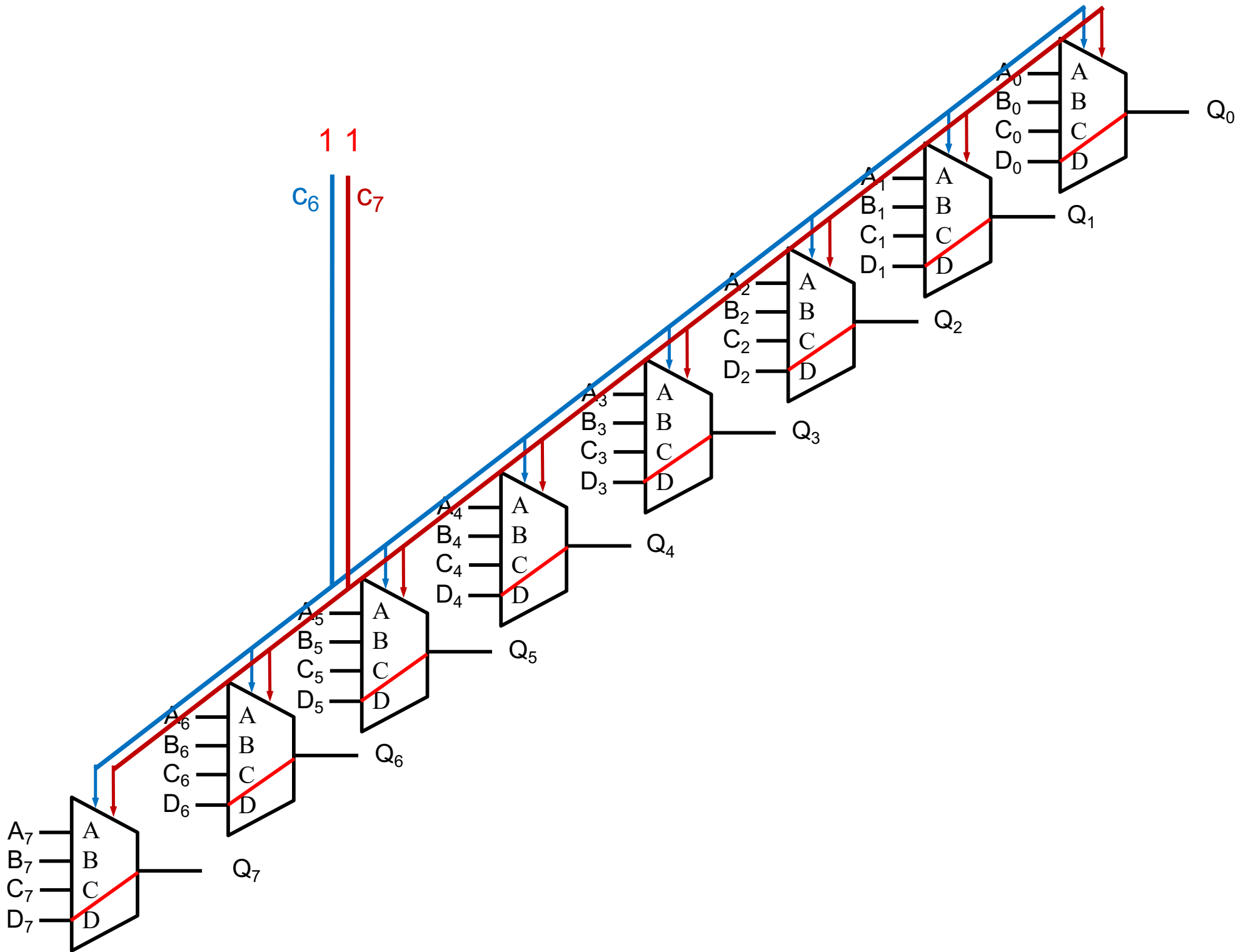
These 8 lines form the output bus for Port_1 of the register file.

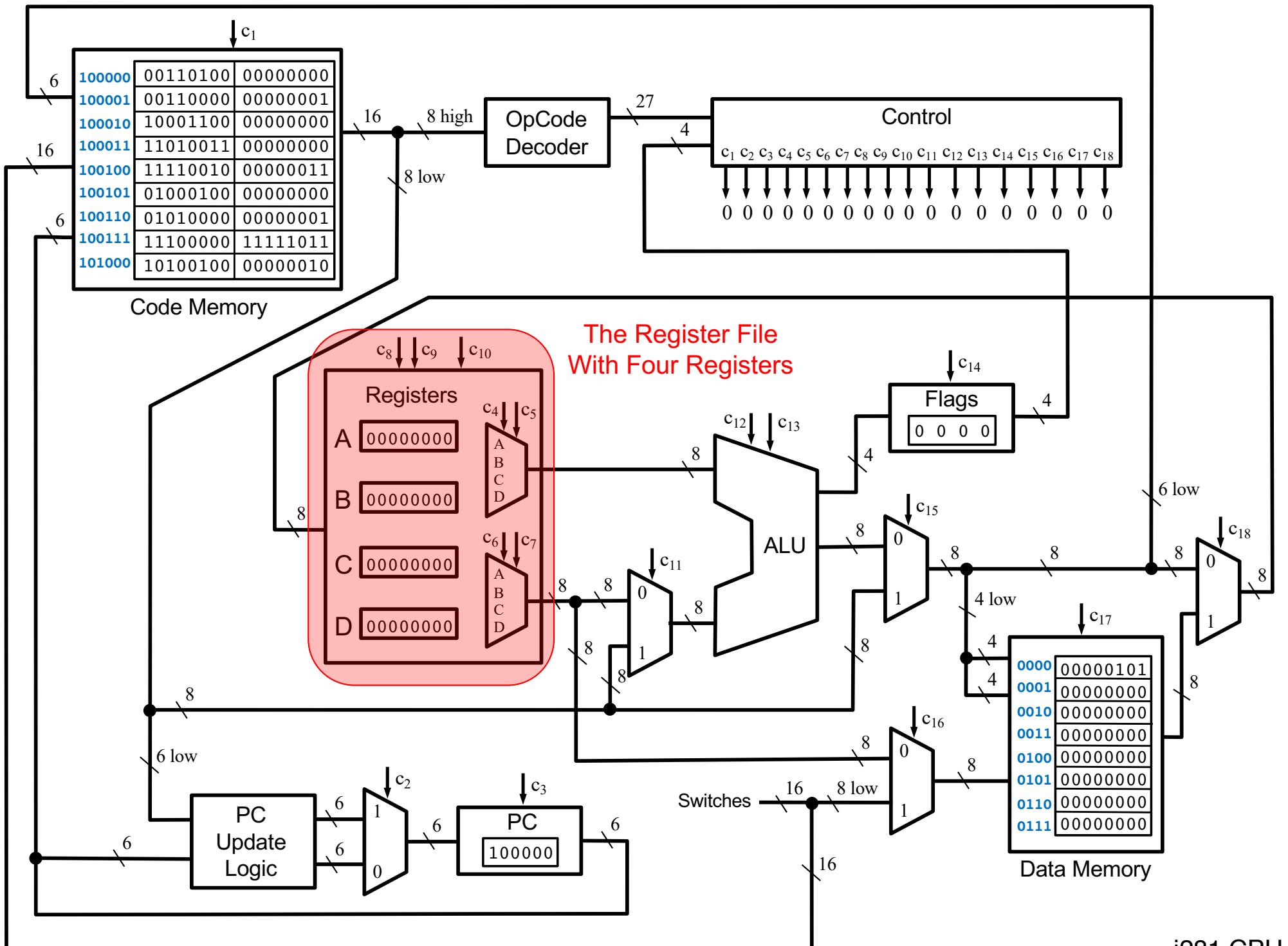








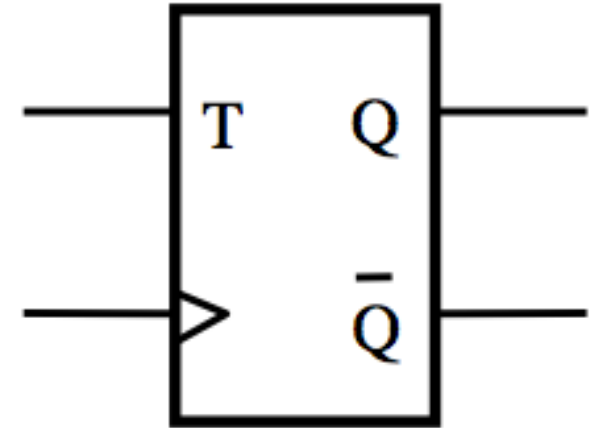
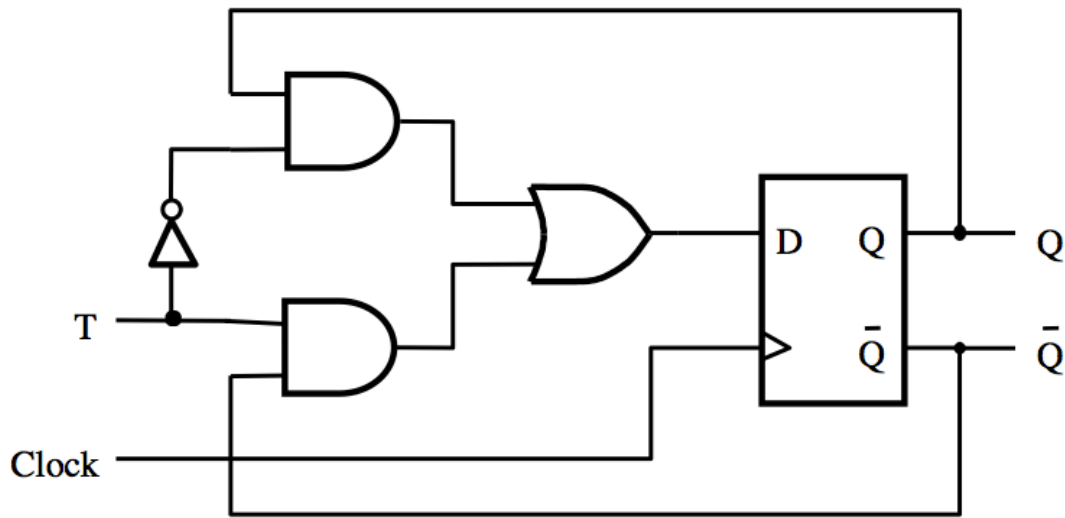




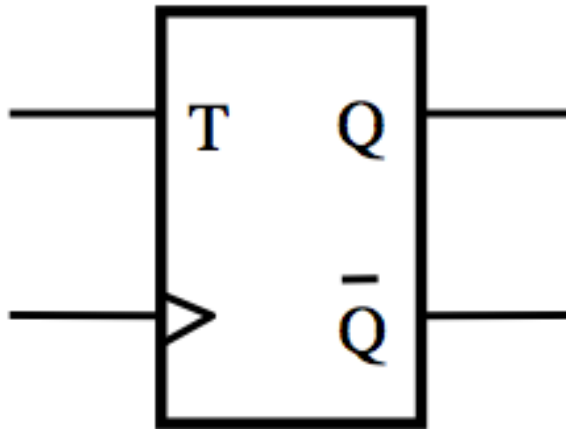
Counters

T Flip-Flop

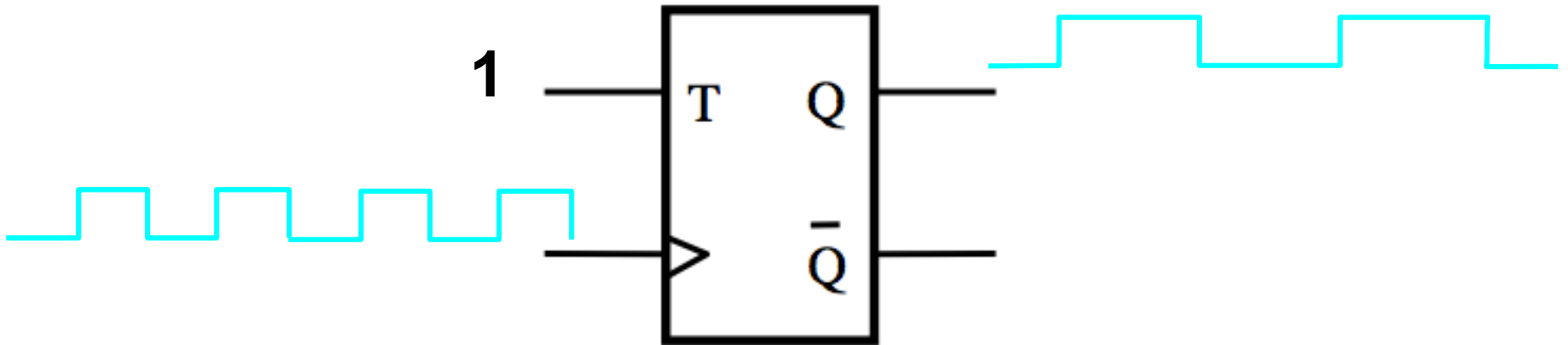
(circuit and graphical symbol)



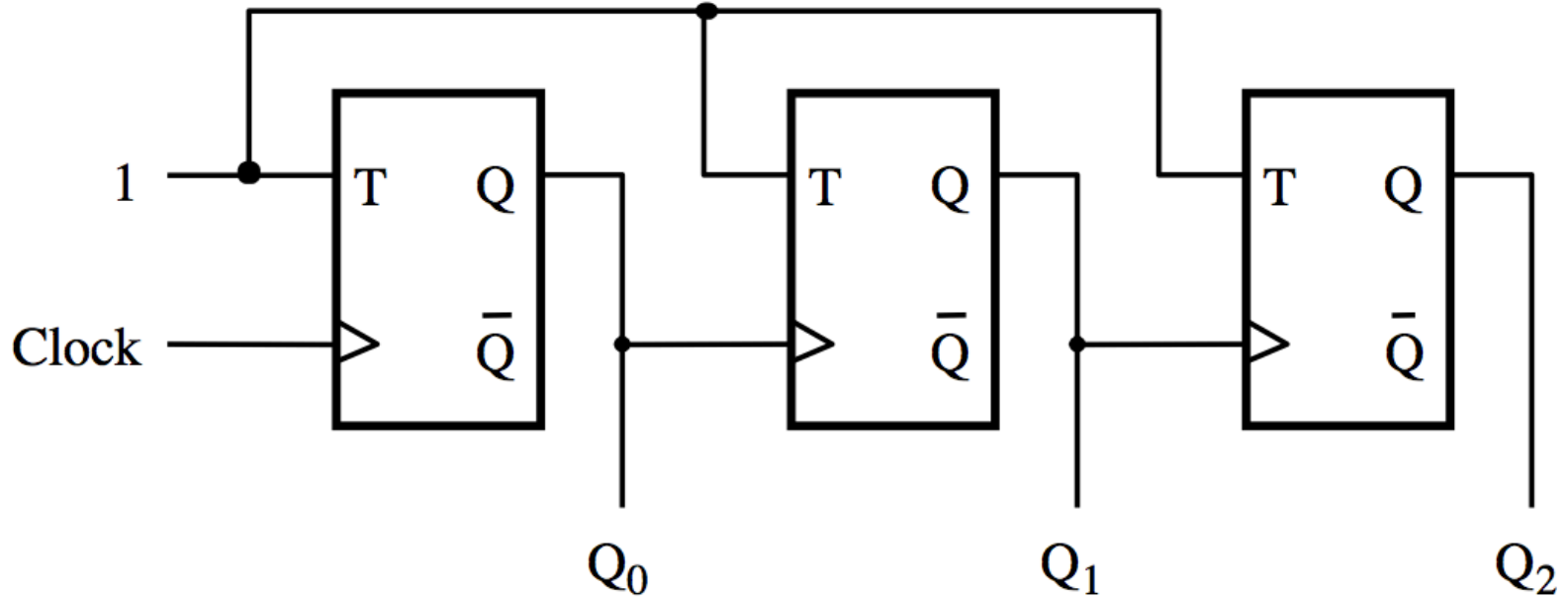
**The output of the T Flip-Flop
divides the frequency of the clock by 2**



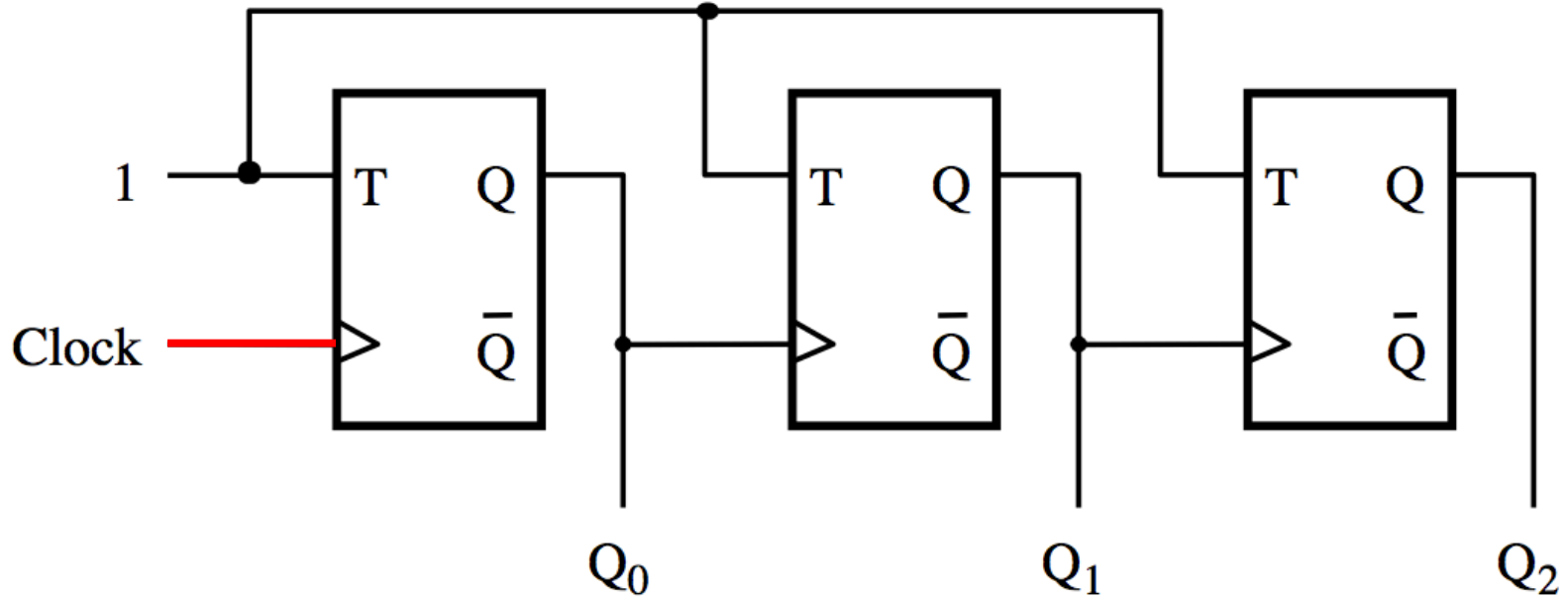
**The output of the T Flip-Flop
divides the frequency of the clock by 2**



A three-bit down-counter

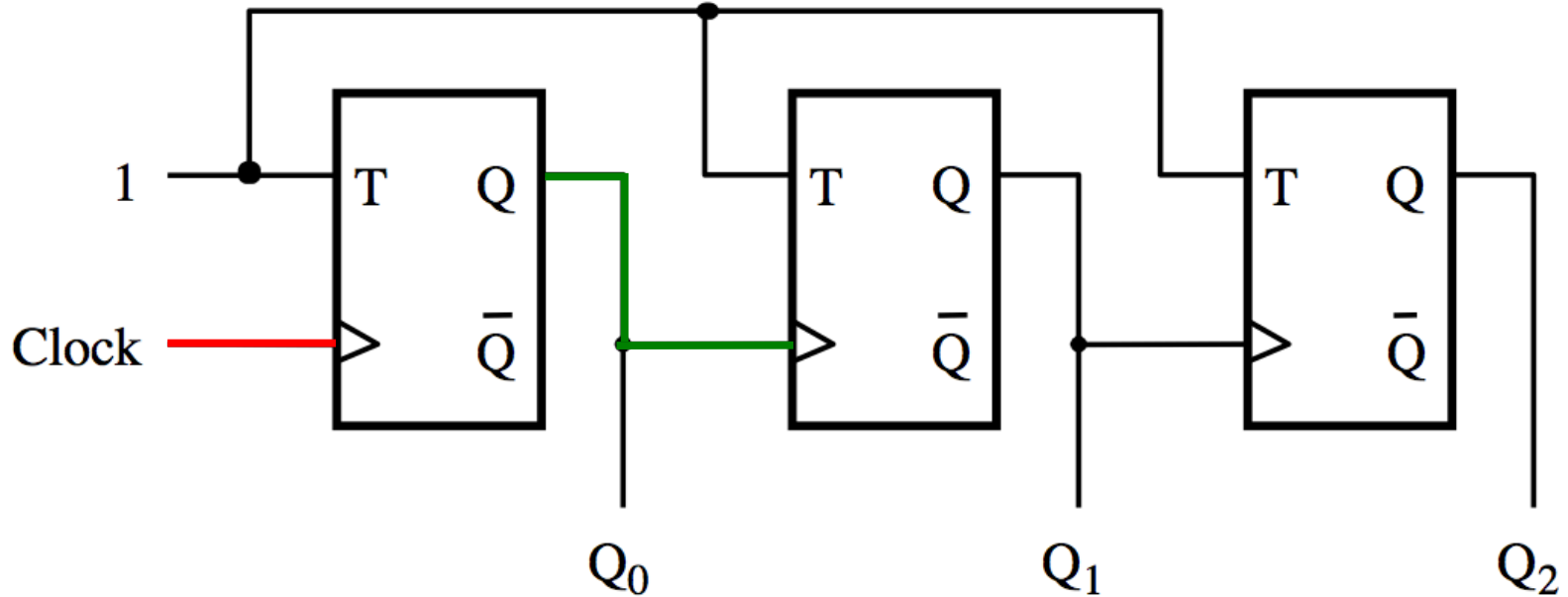


A three-bit down-counter



The first flip-flop changes
on the positive edge of the clock

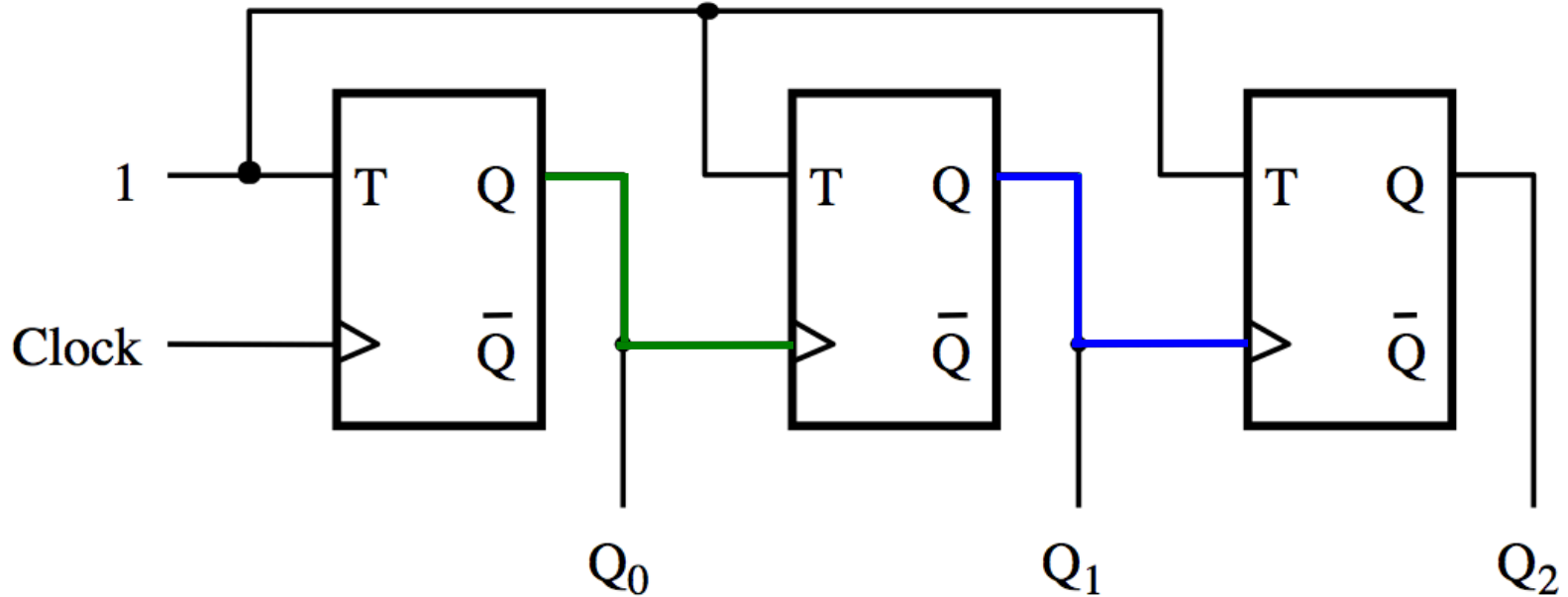
A three-bit down-counter



The first flip-flop changes
on the positive edge of the clock

The second flip-flop changes
on the positive edge of Q_0

A three-bit down-counter

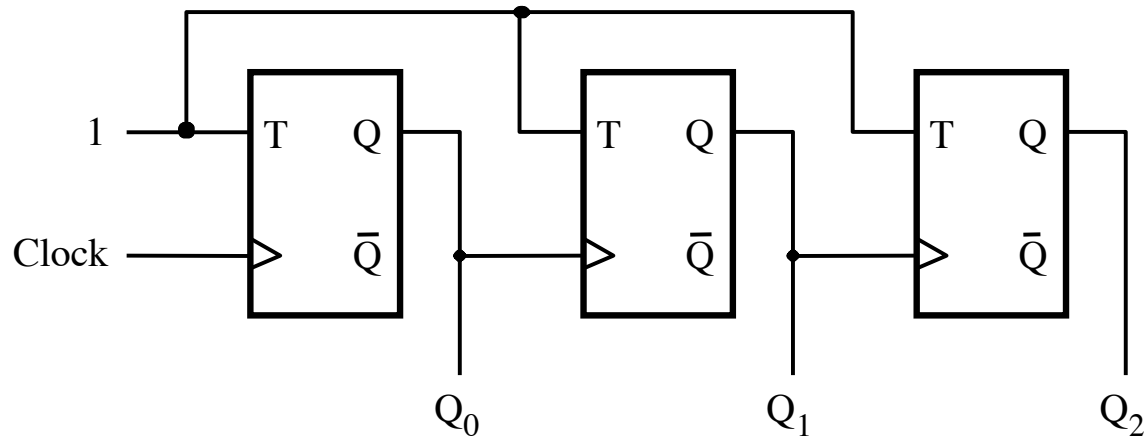


The first flip-flop changes
on the positive edge of the clock

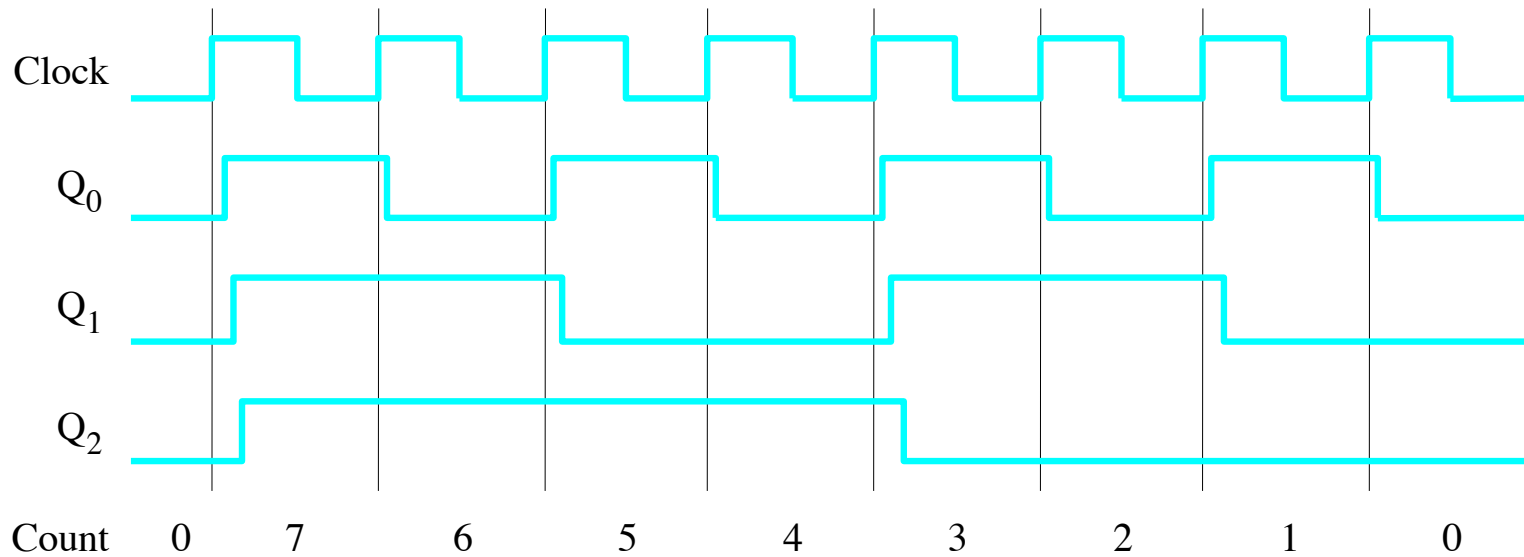
The second flip-flop changes
on the positive edge of Q₀

The third flip-flop changes
on the positive edge of Q₁

A three-bit down-counter



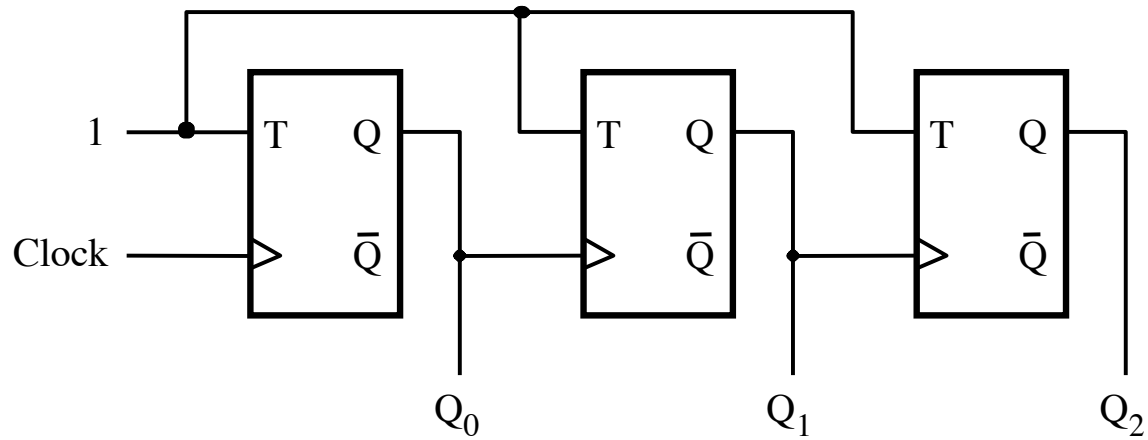
(a) Circuit



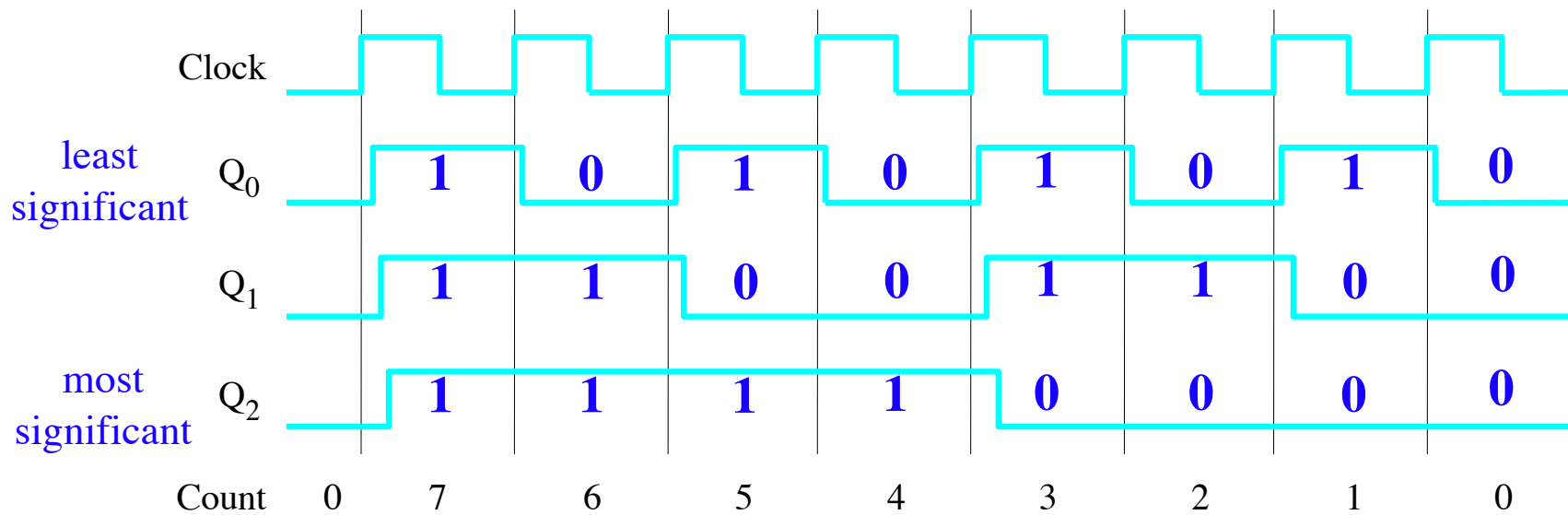
(b) Timing diagram

[Figure 5.20 from the textbook]

A three-bit down-counter

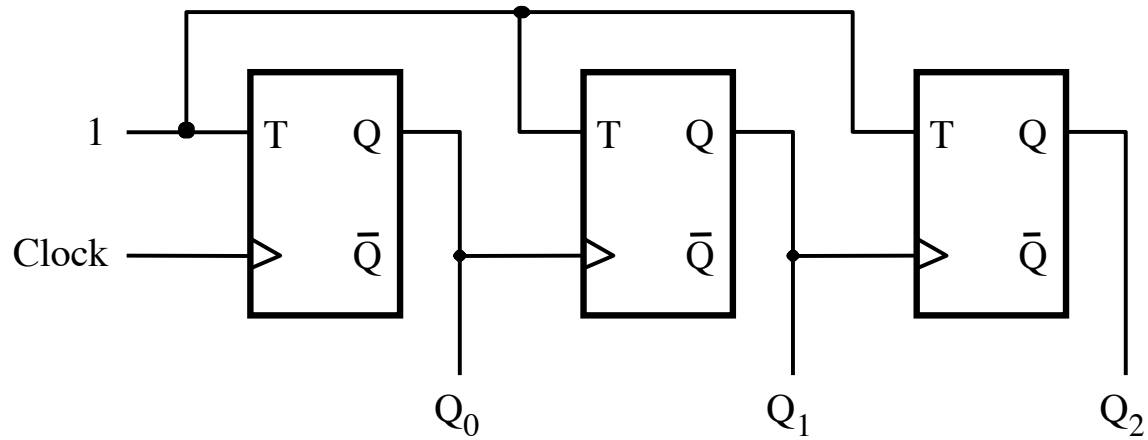


(a) Circuit

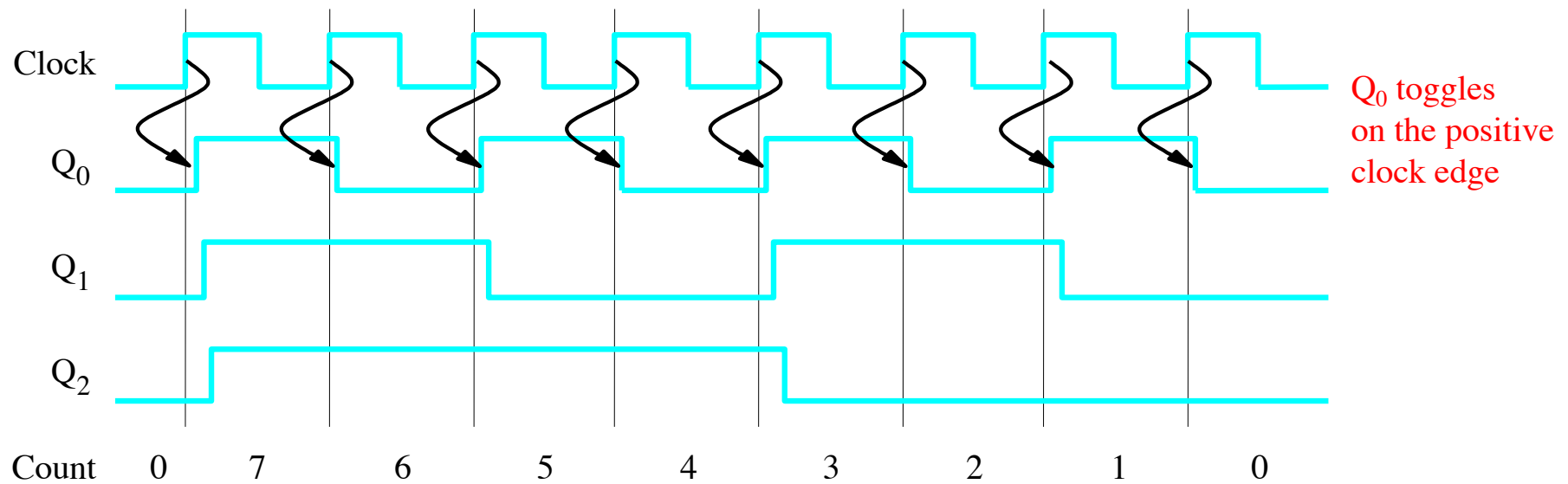


(b) Timing diagram

A three-bit down-counter

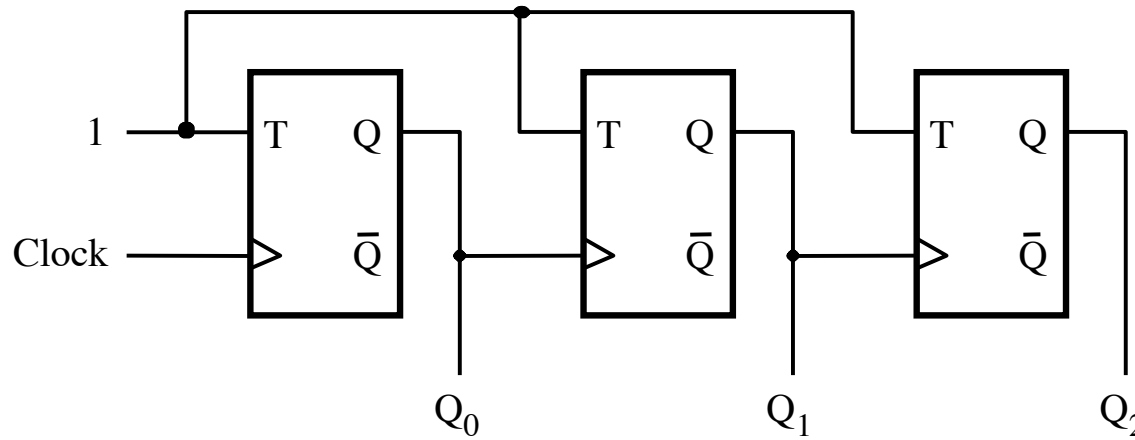


(a) Circuit

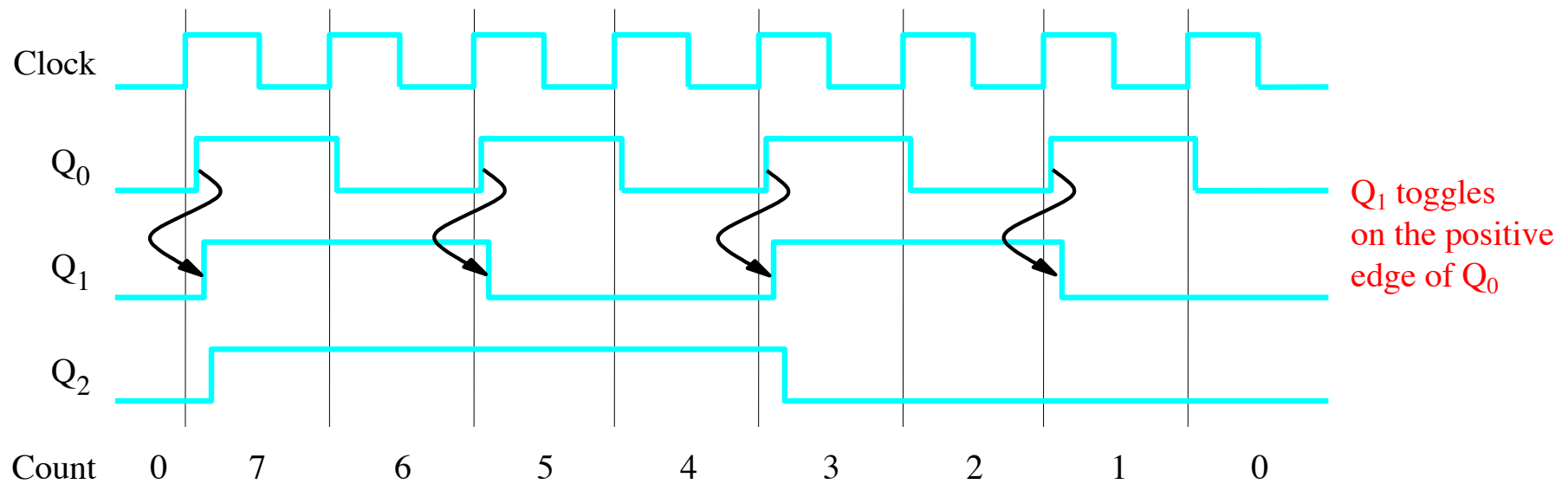


(b) Timing diagram

A three-bit down-counter

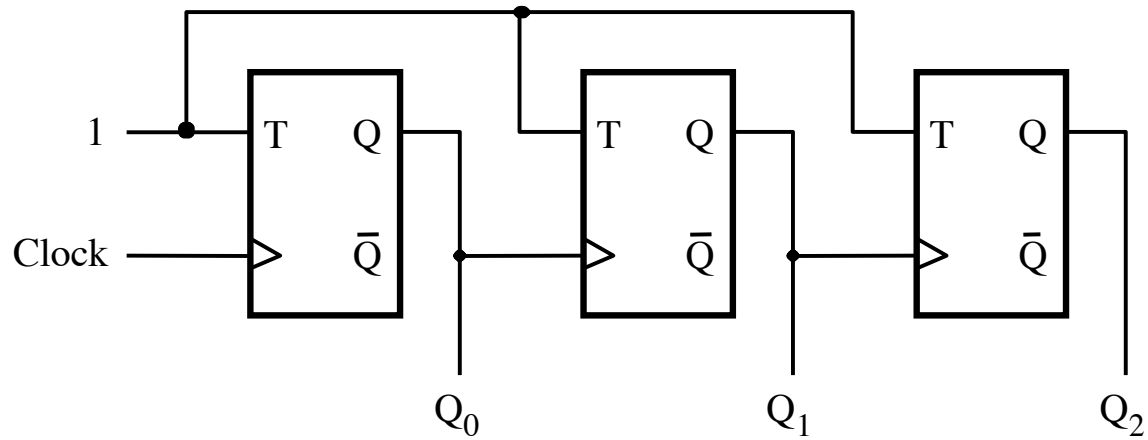


(a) Circuit

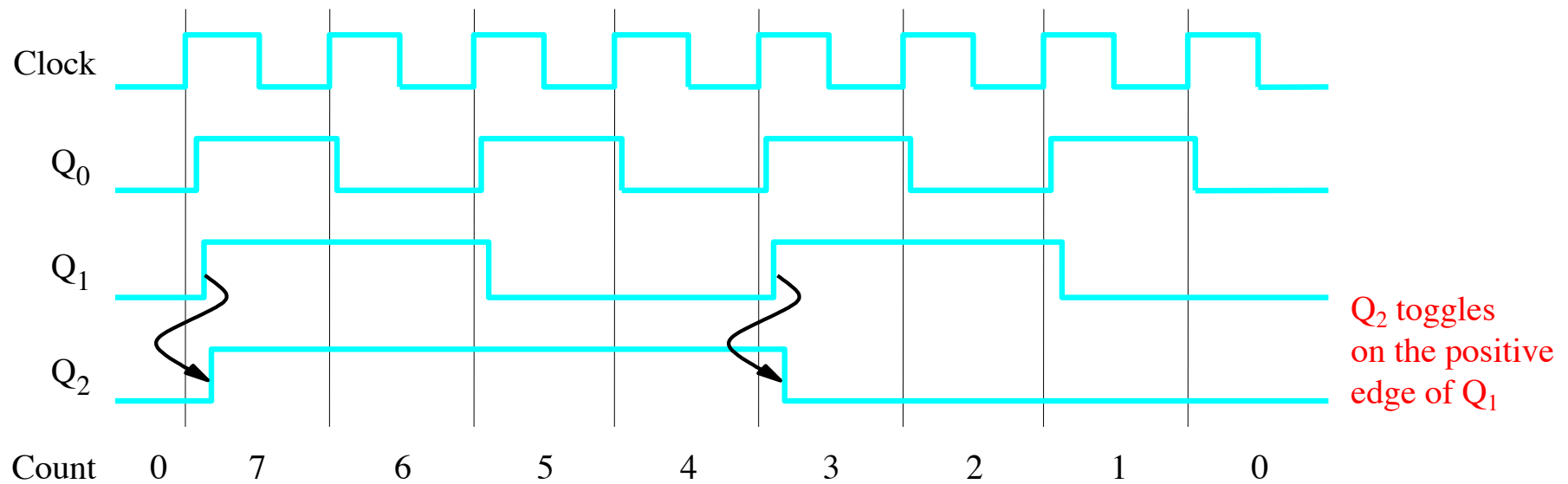


(b) Timing diagram

A three-bit down-counter

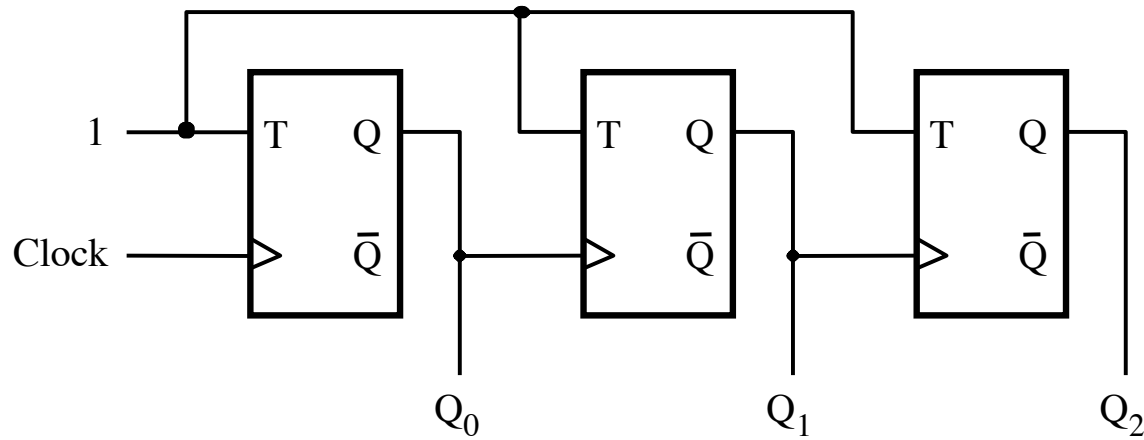


(a) Circuit



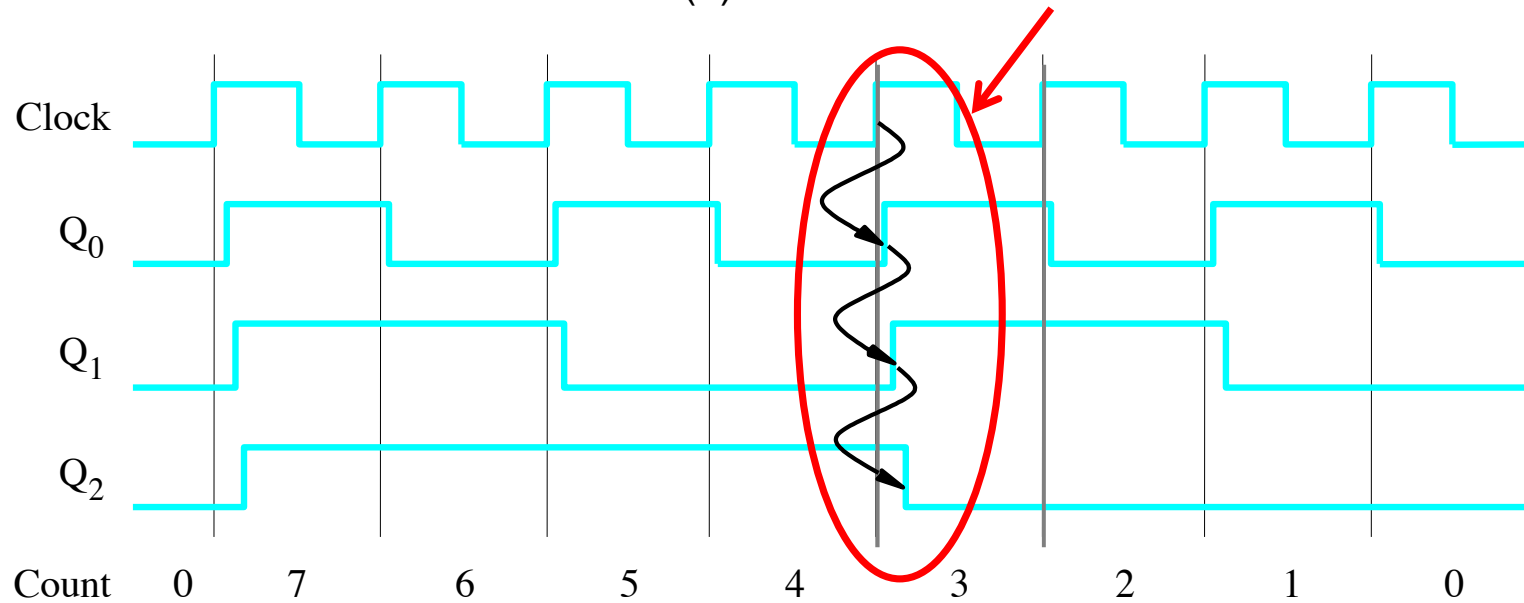
(b) Timing diagram

A three-bit down-counter



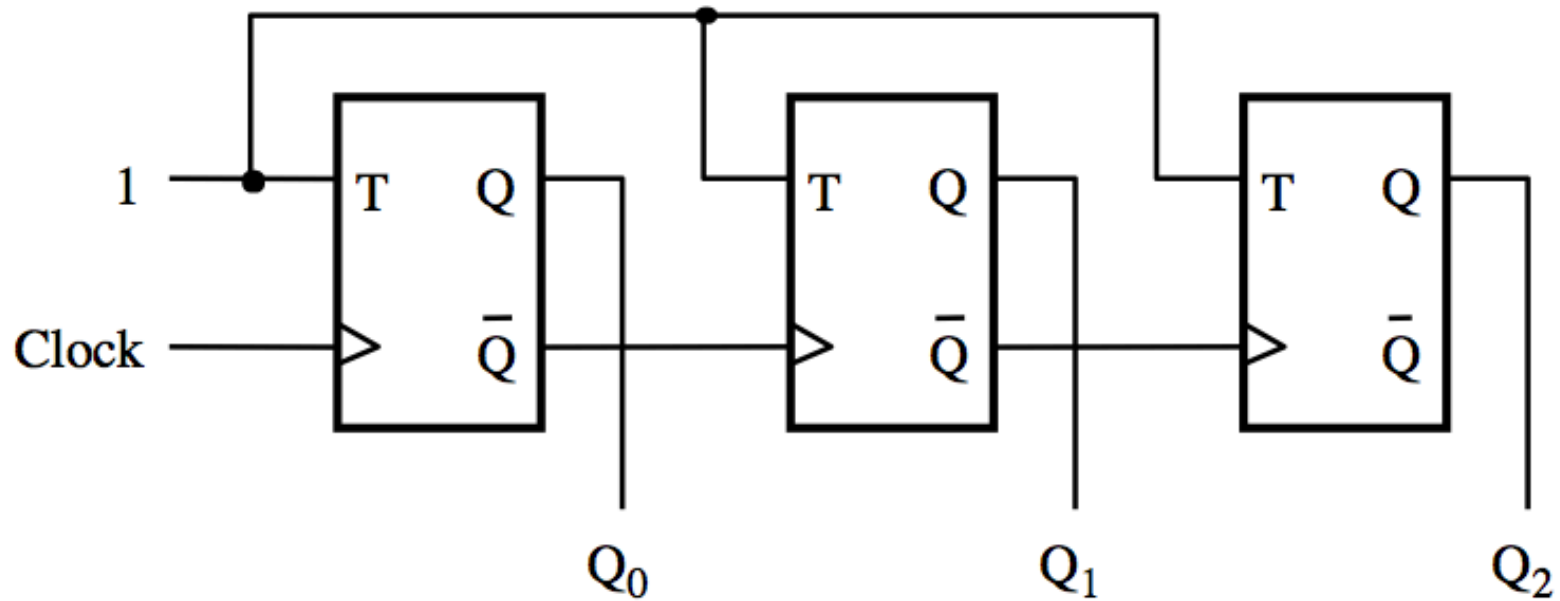
(a) Circuit

The propagation delays get longer

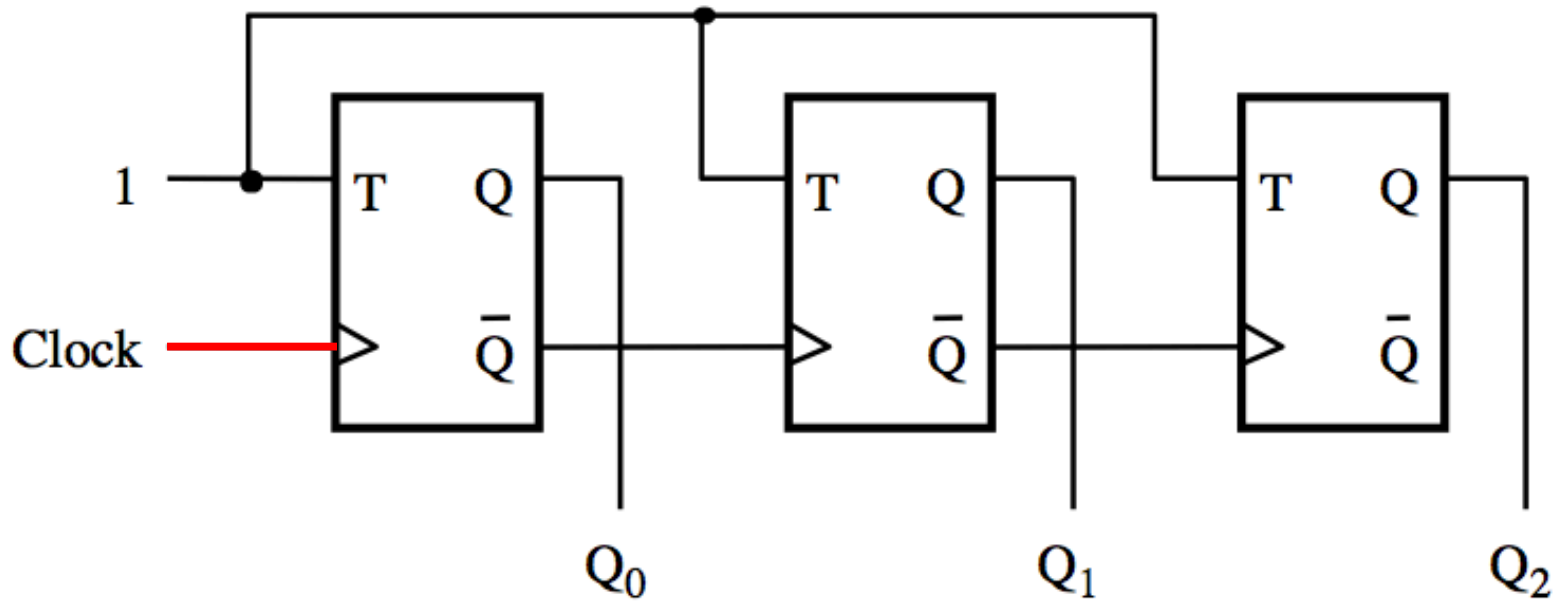


(b) Timing diagram

A three-bit up-counter

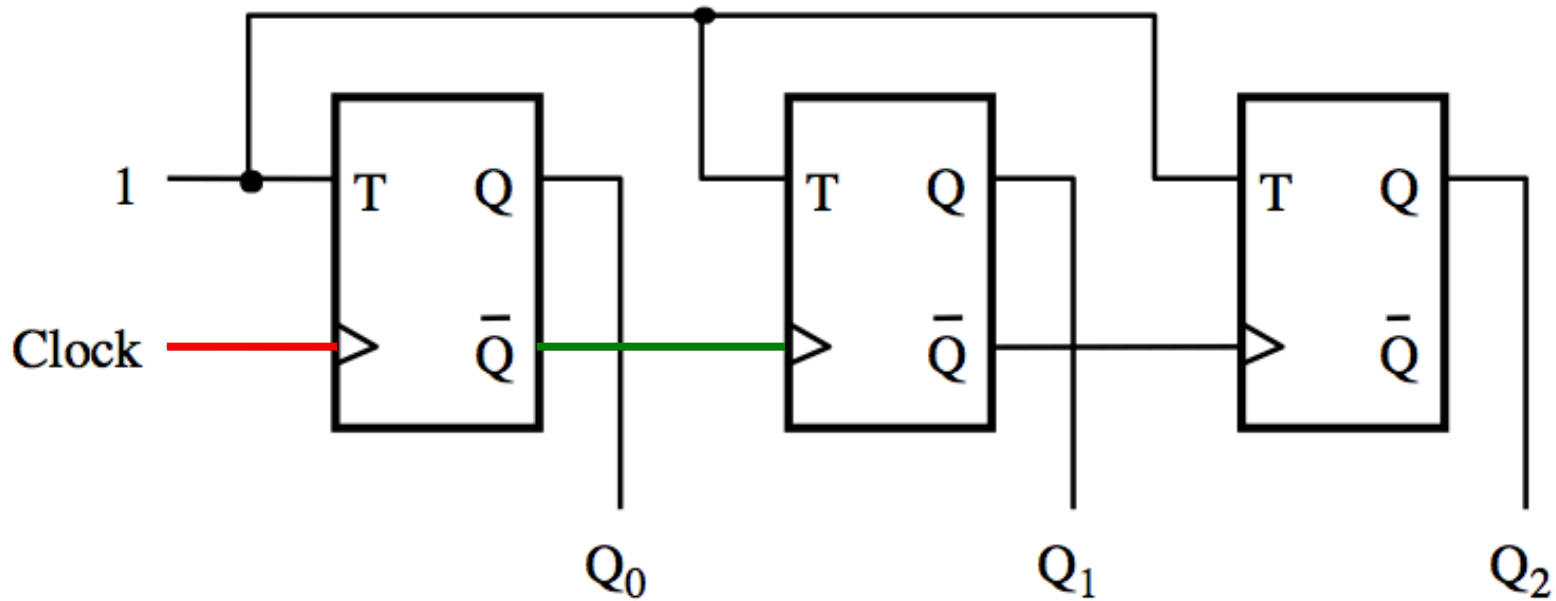


A three-bit up-counter



The first flip-flop changes
on the positive edge of the clock

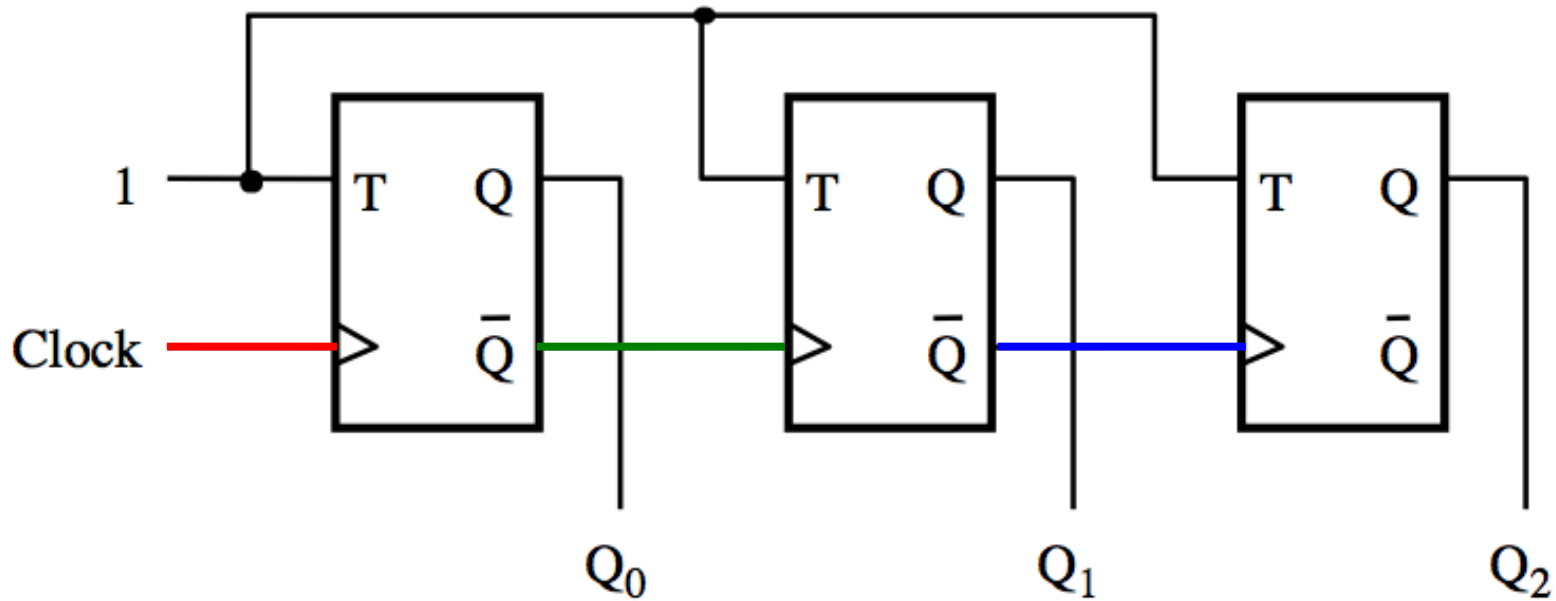
A three-bit up-counter



The first flip-flop changes on the positive edge of the clock

The second flip-flop changes on the positive edge of \bar{Q}_0

A three-bit up-counter

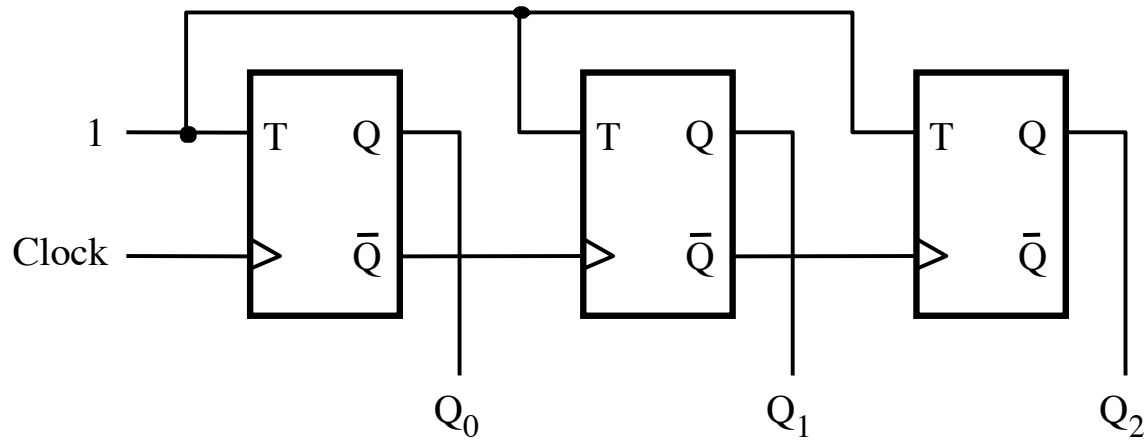


The first flip-flop changes on the positive edge of the clock

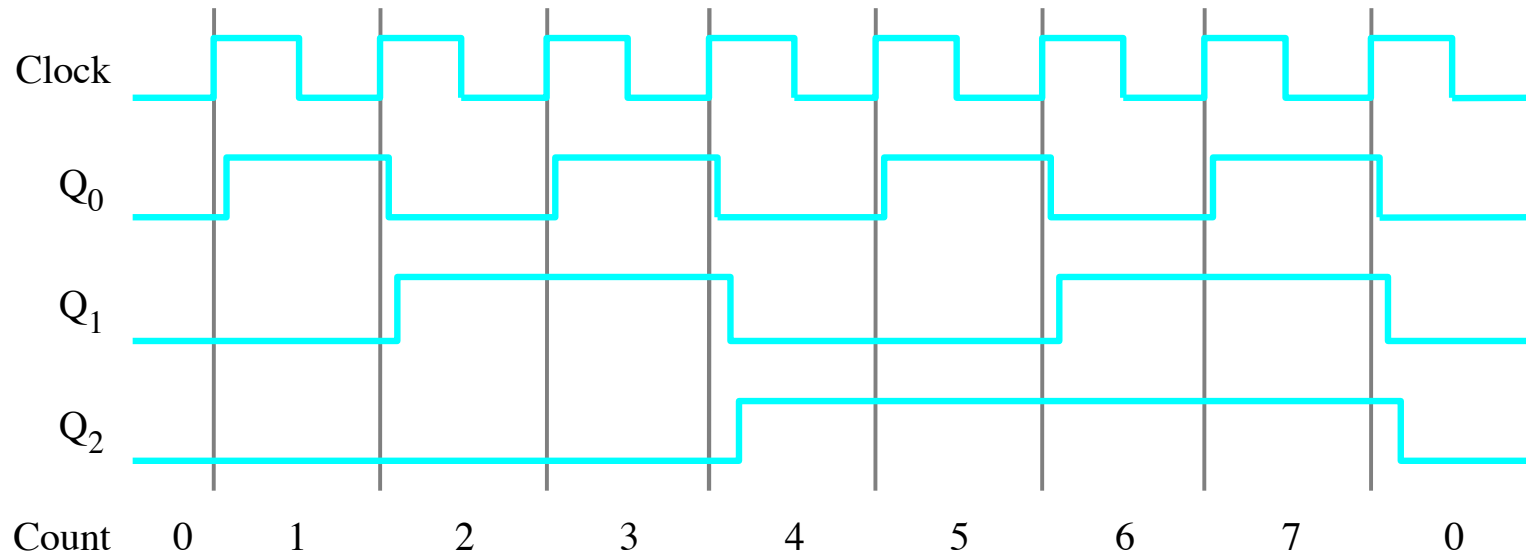
The second flip-flop changes on the positive edge of \bar{Q}_0

The third flip-flop changes on the positive edge of \bar{Q}_1

A three-bit up-counter

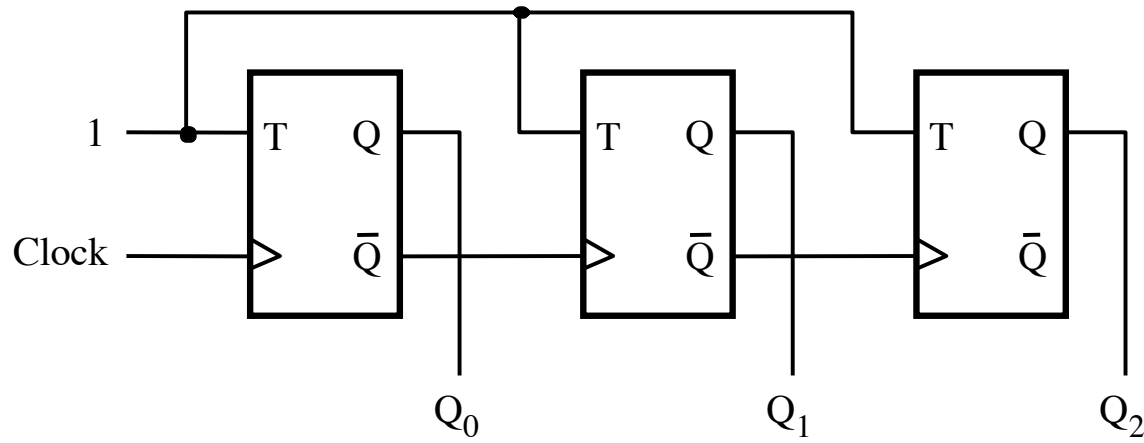


(a) Circuit

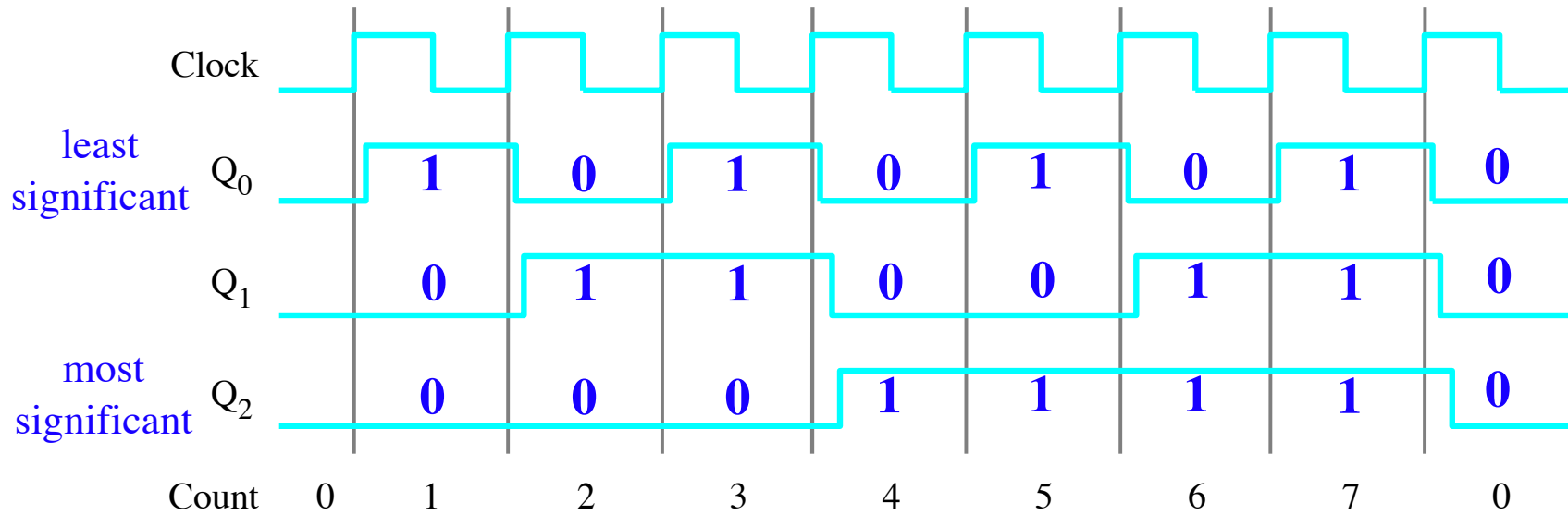


(b) Timing diagram

A three-bit up-counter

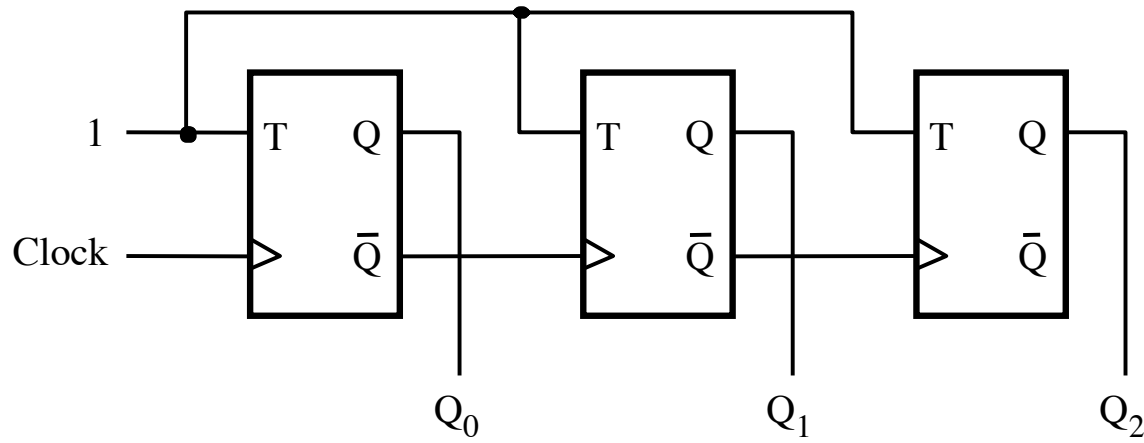


(a) Circuit

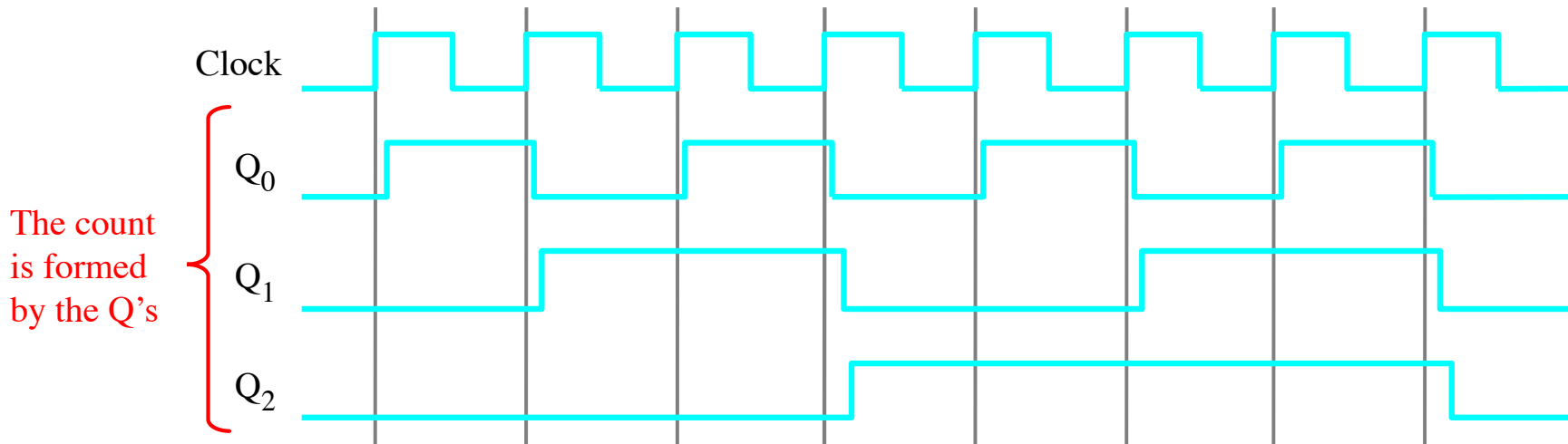


(b) Timing diagram

A three-bit up-counter

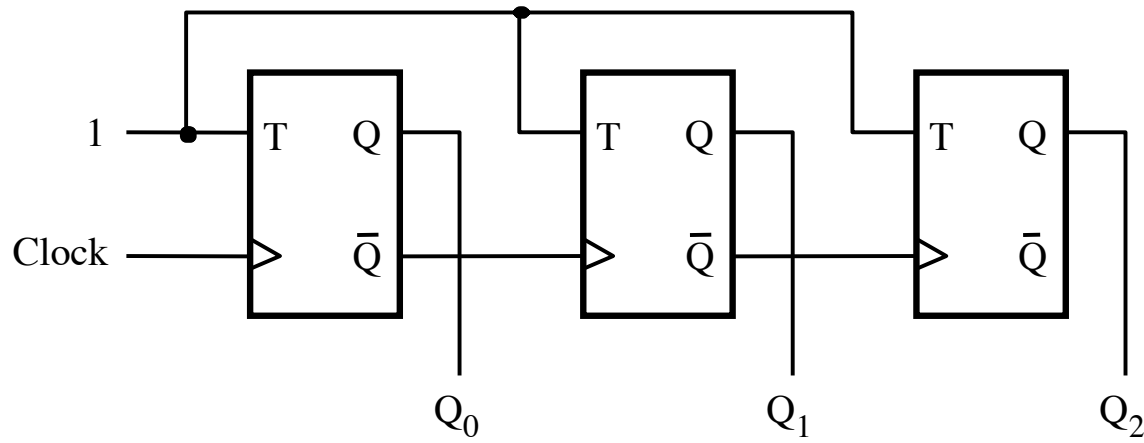


(a) Circuit

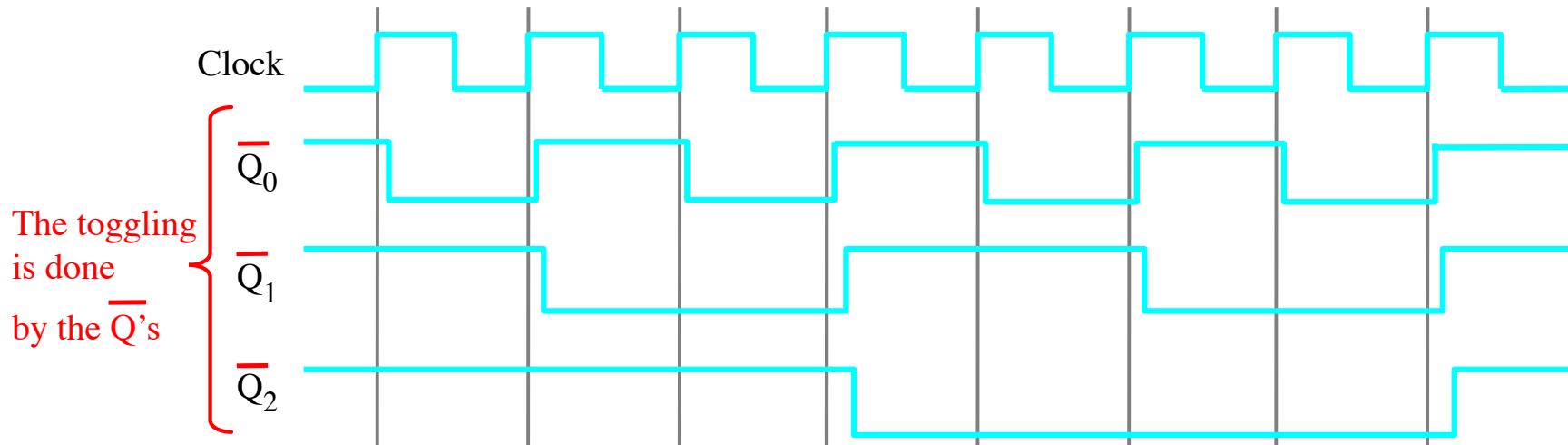


(b) Timing diagram

A three-bit up-counter

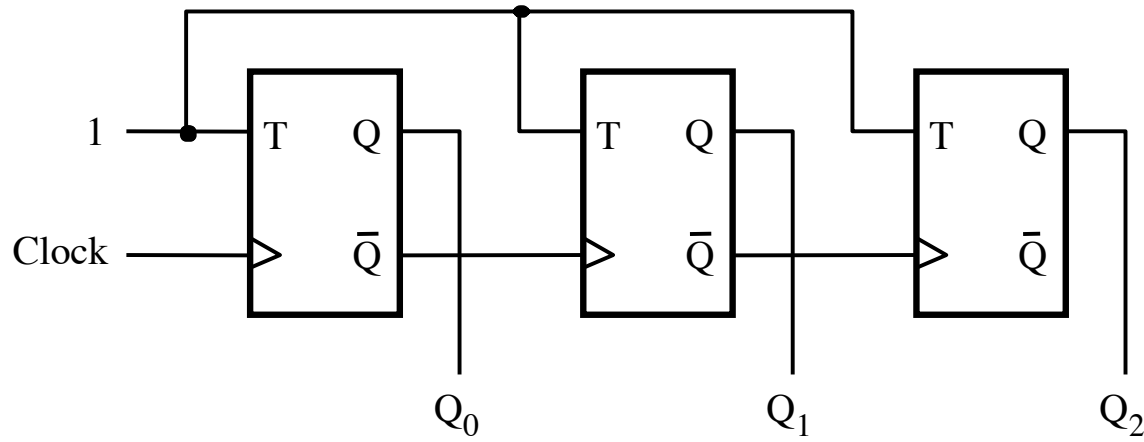


(a) Circuit

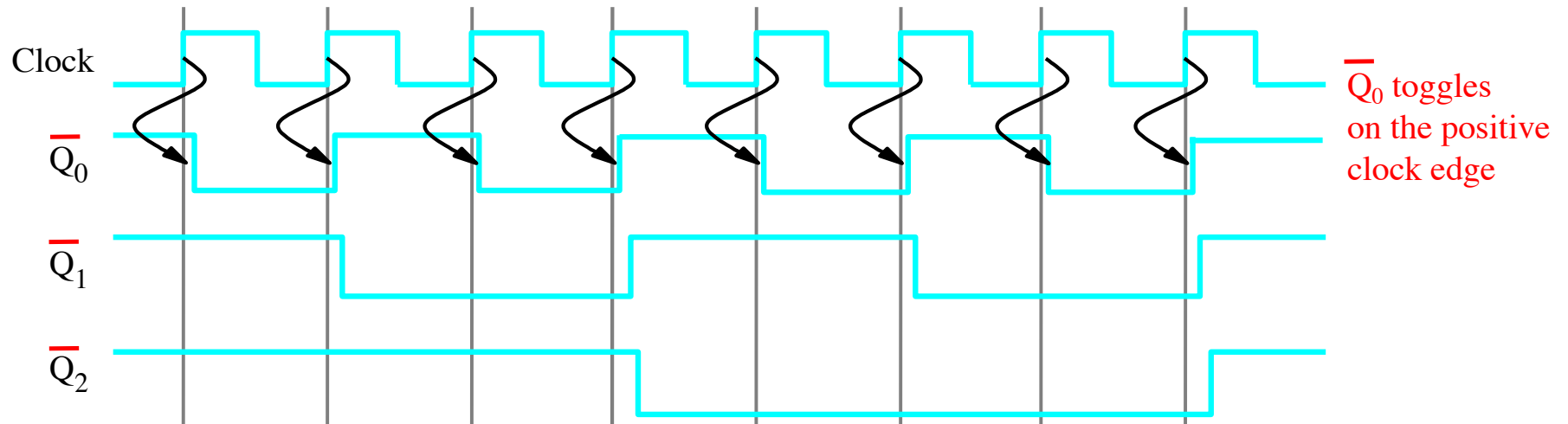


(b) Timing diagram

A three-bit up-counter

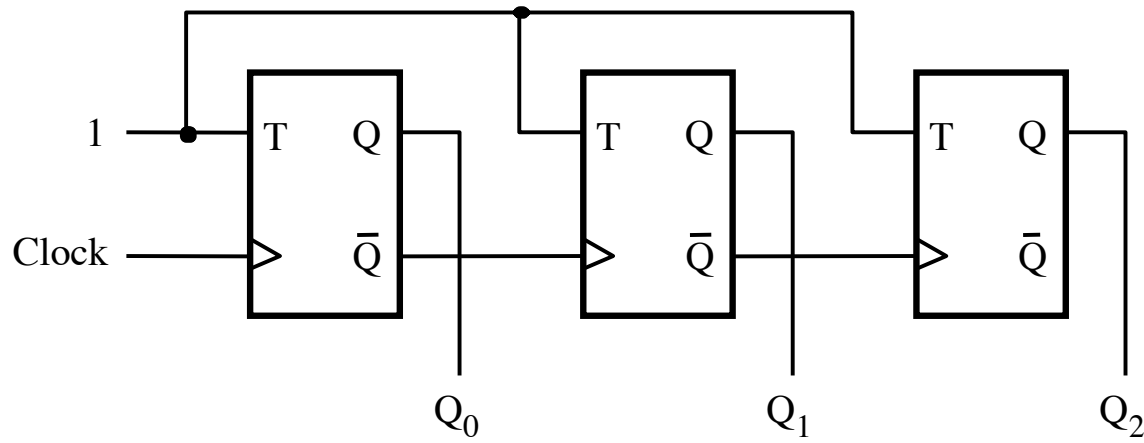


(a) Circuit

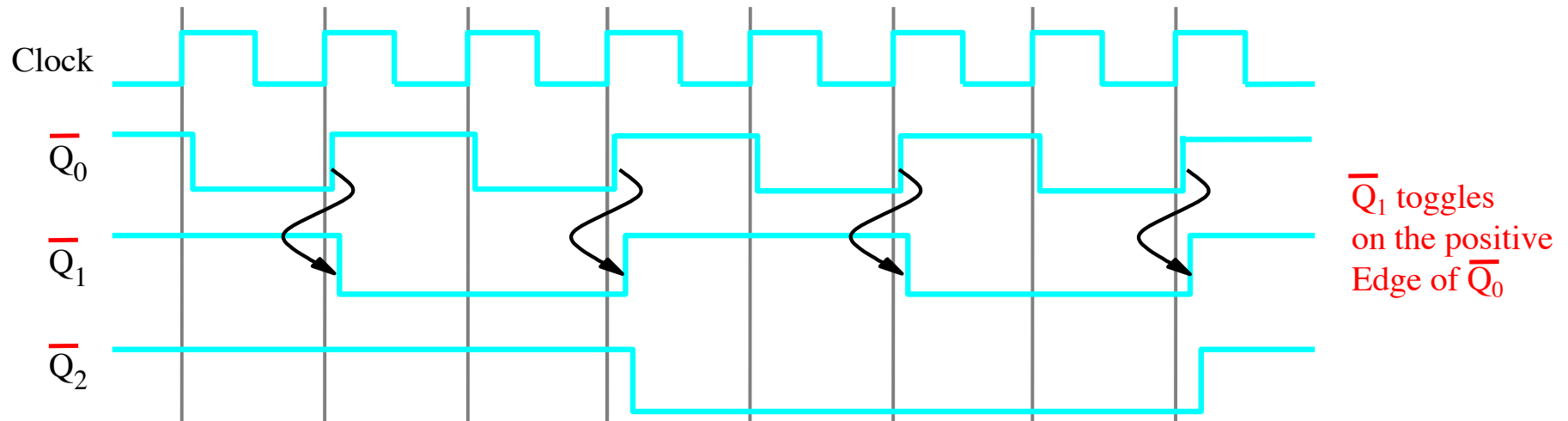


(b) Timing diagram

A three-bit up-counter

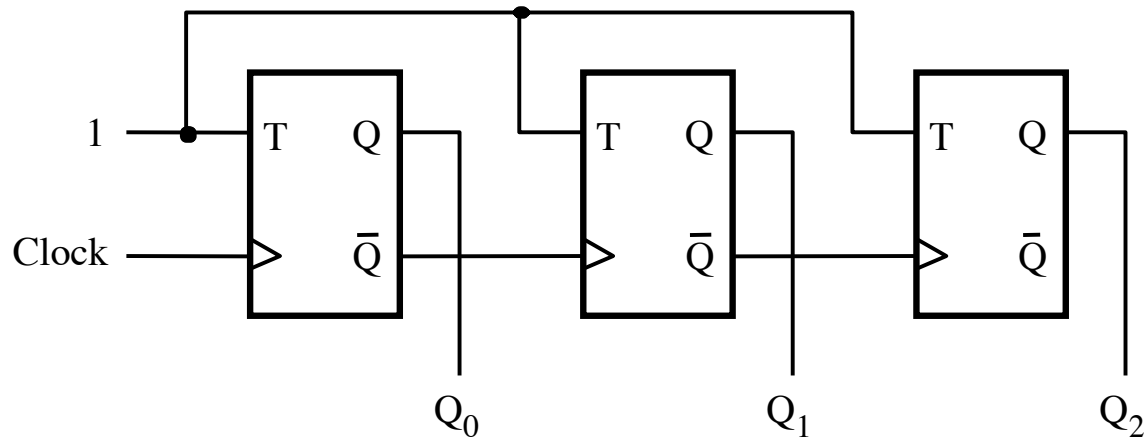


(a) Circuit

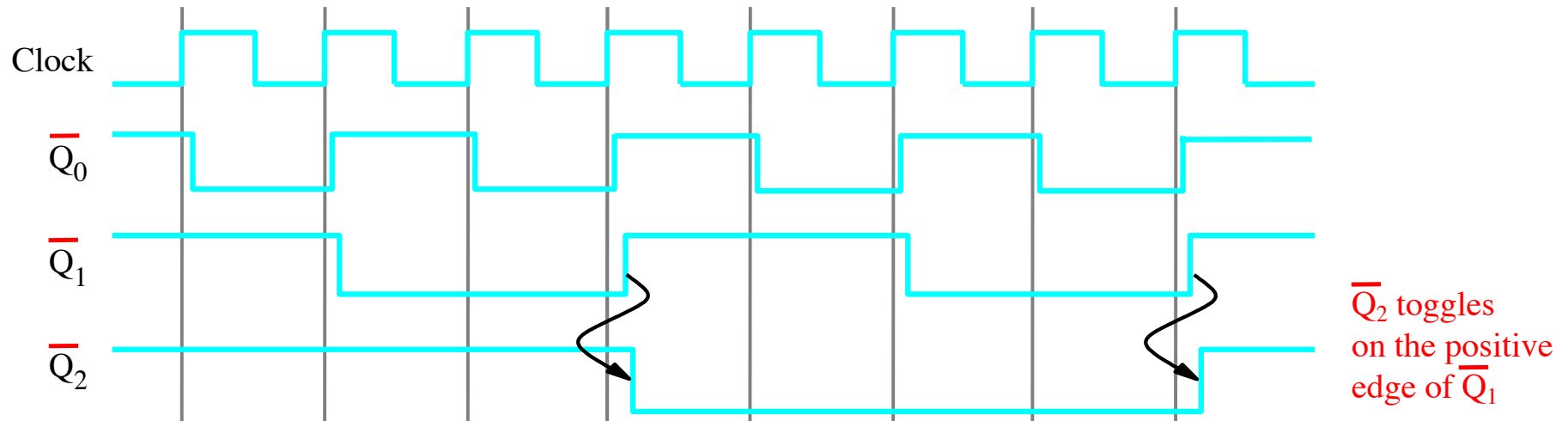


(b) Timing diagram

A three-bit up-counter

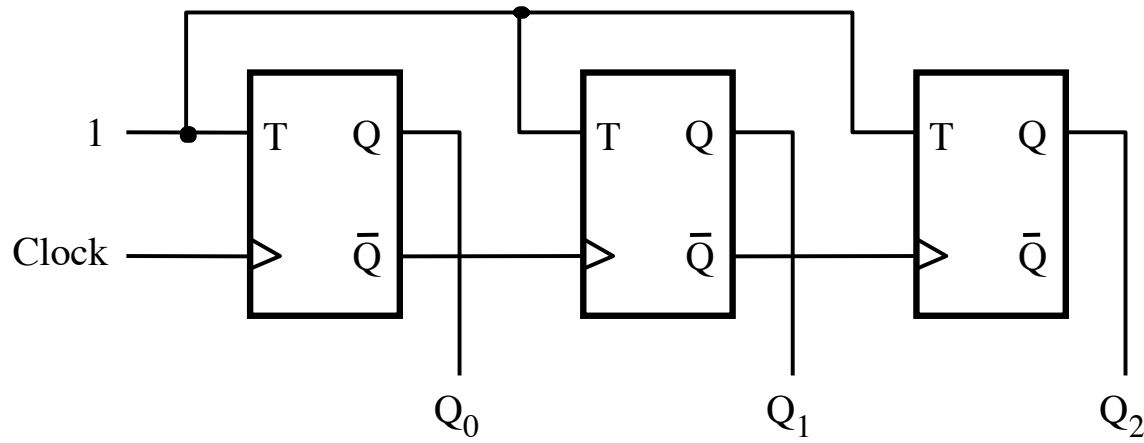


(a) Circuit

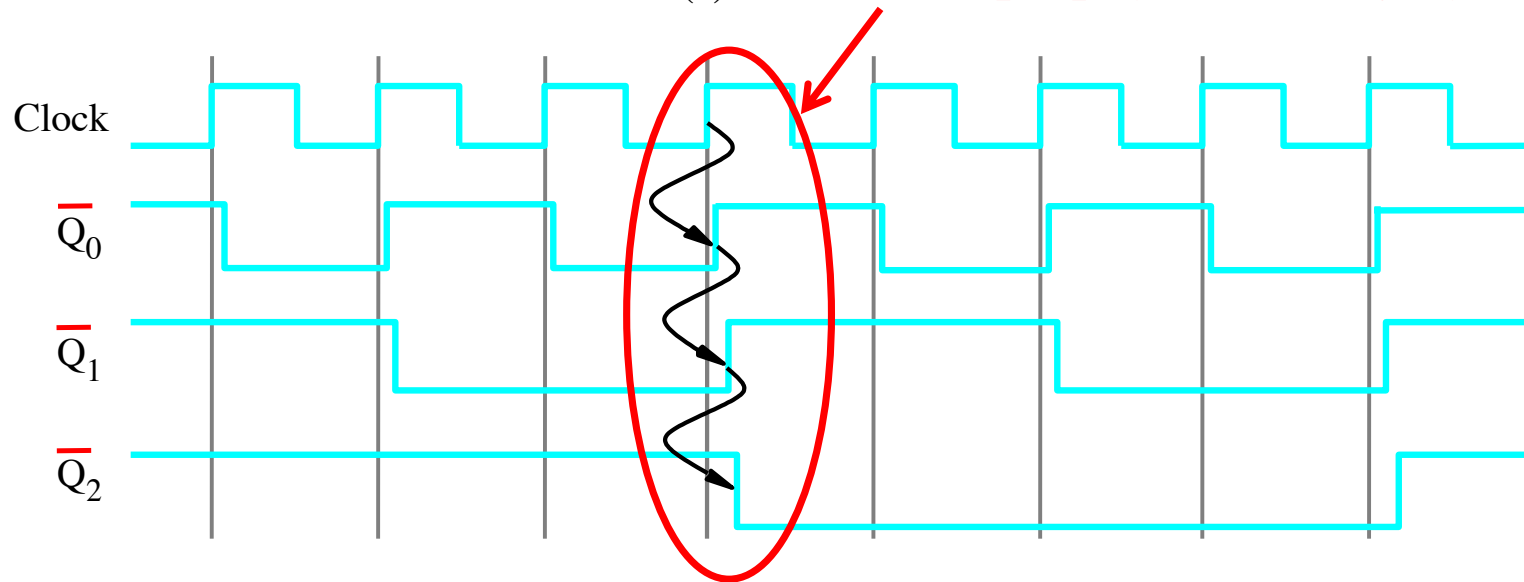


(b) Timing diagram

A three-bit up-counter

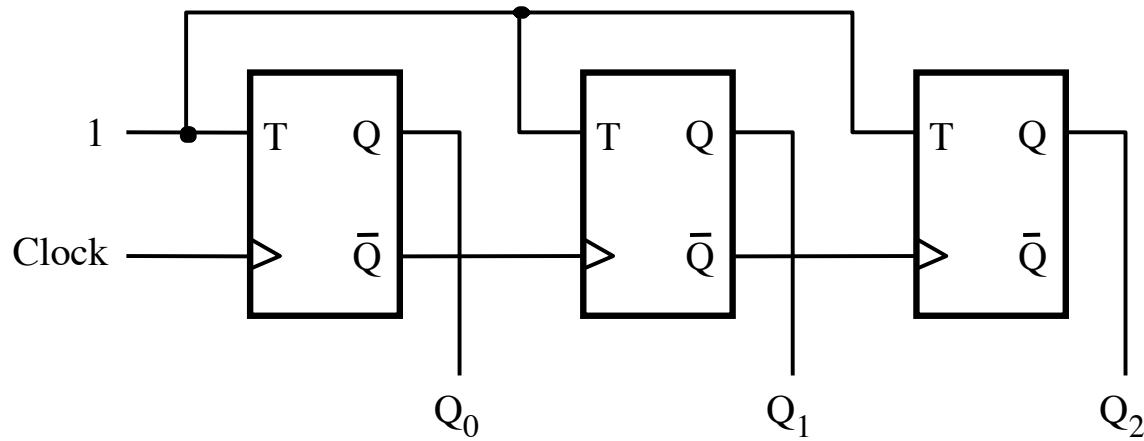


(a) Circuit **The propagation delays get longer**

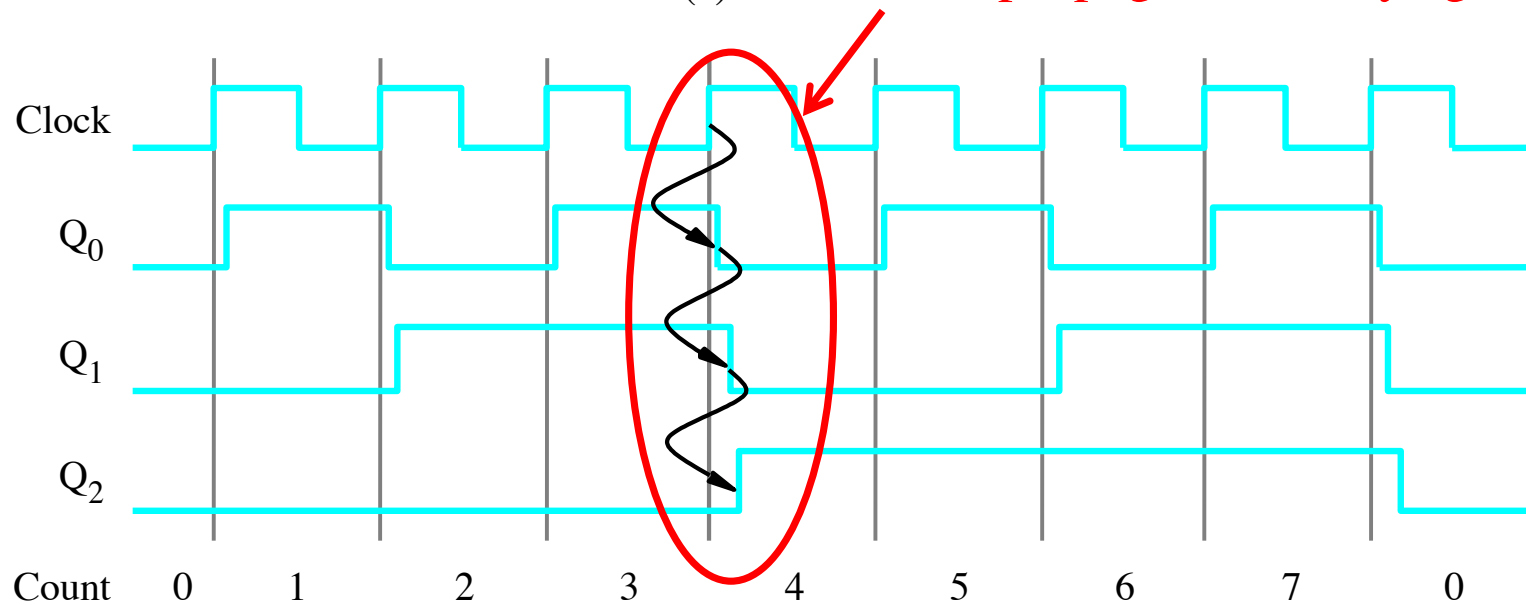


(b) Timing diagram

A three-bit up-counter



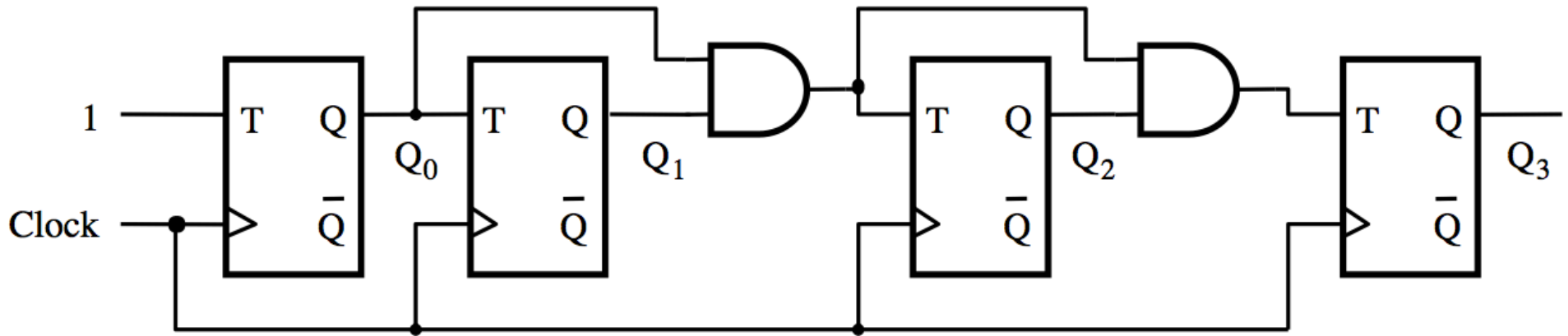
(a) Circuit **The propagation delays get longer**



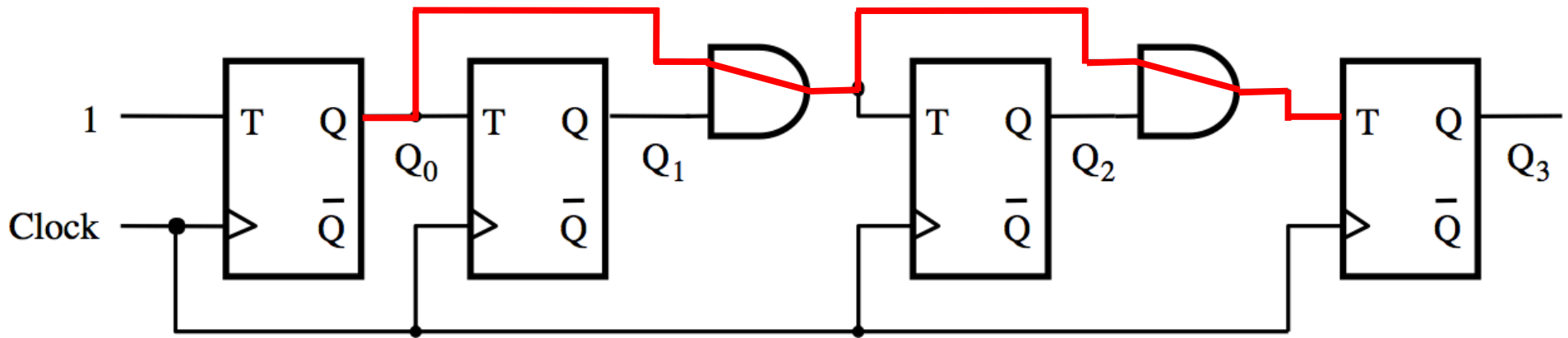
(b) Timing diagram

Synchronous Counters

A four-bit synchronous up-counter

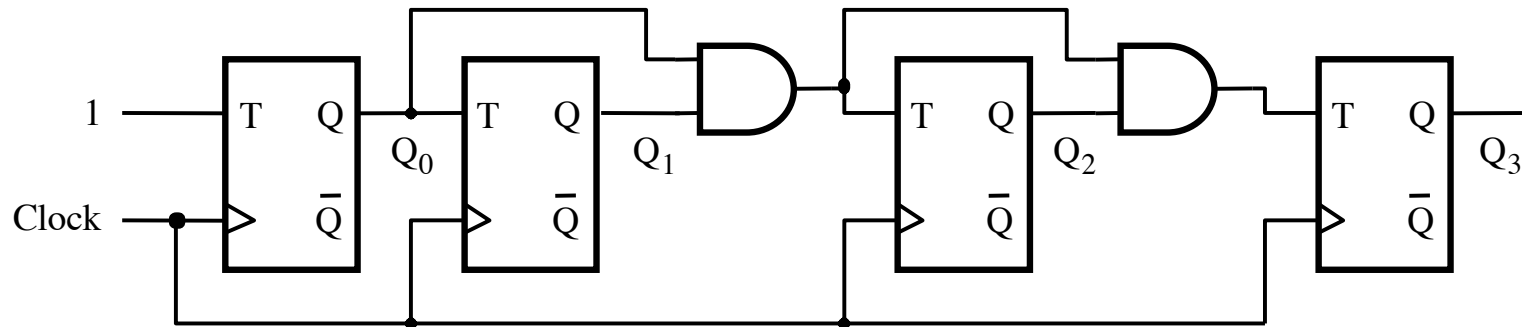


A four-bit synchronous up-counter

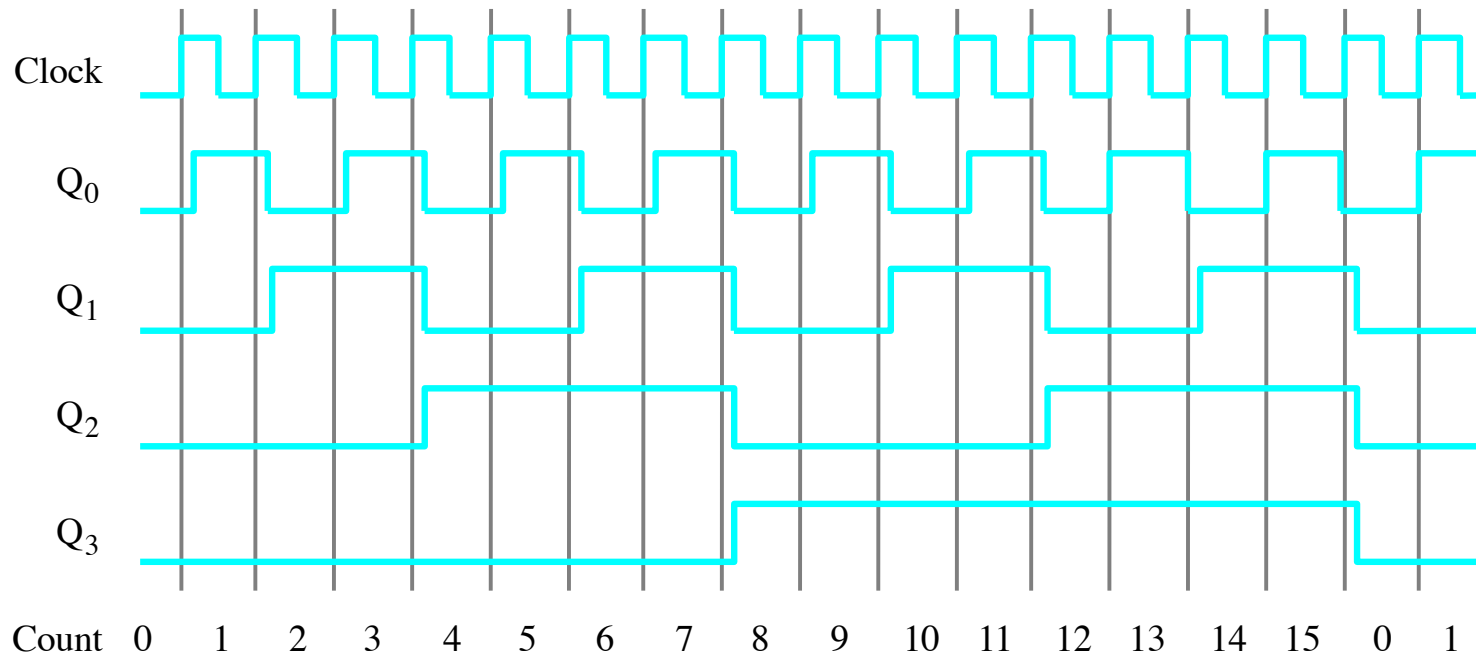


The propagation delay through all AND gates combined must not exceed the clock period minus the setup time for the flip-flops

A four-bit synchronous up-counter



(a) Circuit



(b) Timing diagram

Derivation of the synchronous up-counter

Clock cycle	Q ₂	Q ₁	Q ₀
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Q₁ changes

Q₂ changes

Derivation of the synchronous up-counter

Clock cycle	Q ₂	Q ₁	Q ₀
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Q₁ changes

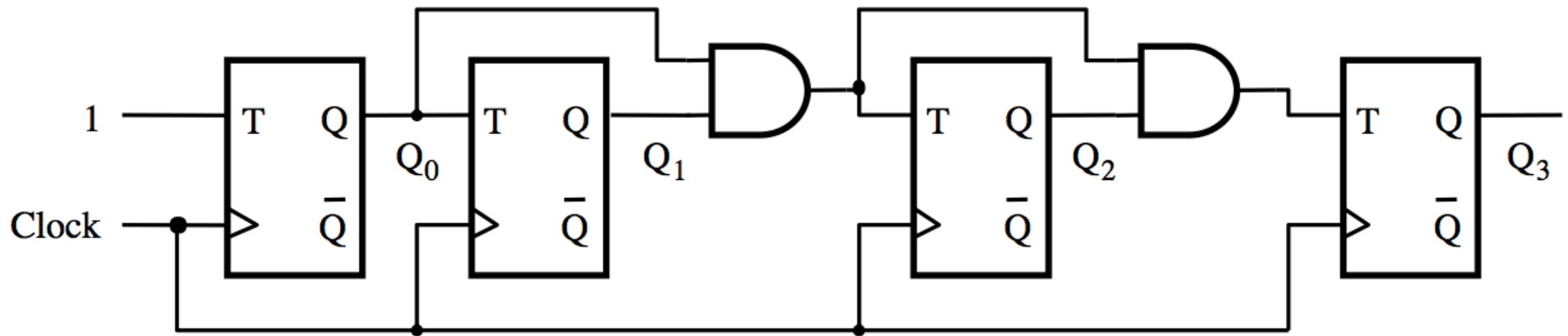
Q₂ changes

$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0 Q_1$$

A four-bit synchronous up-counter



$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0 Q_1$$

In general we have

$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0 Q_1$$

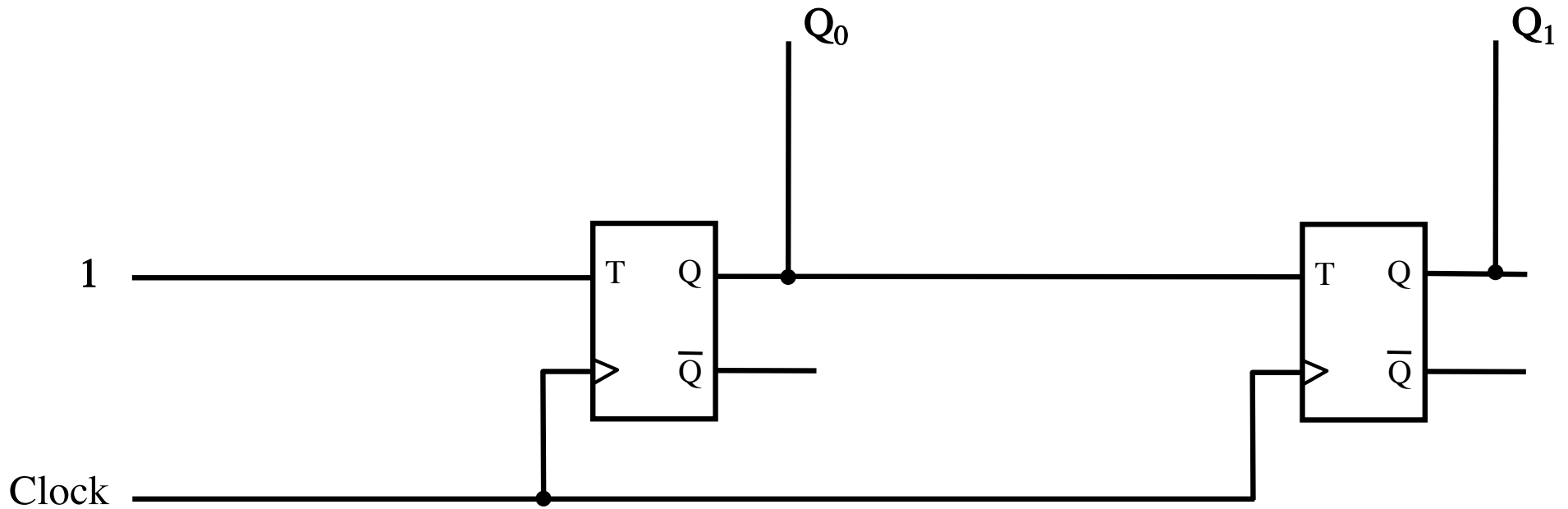
$$T_3 = Q_0 Q_1 Q_2$$

...

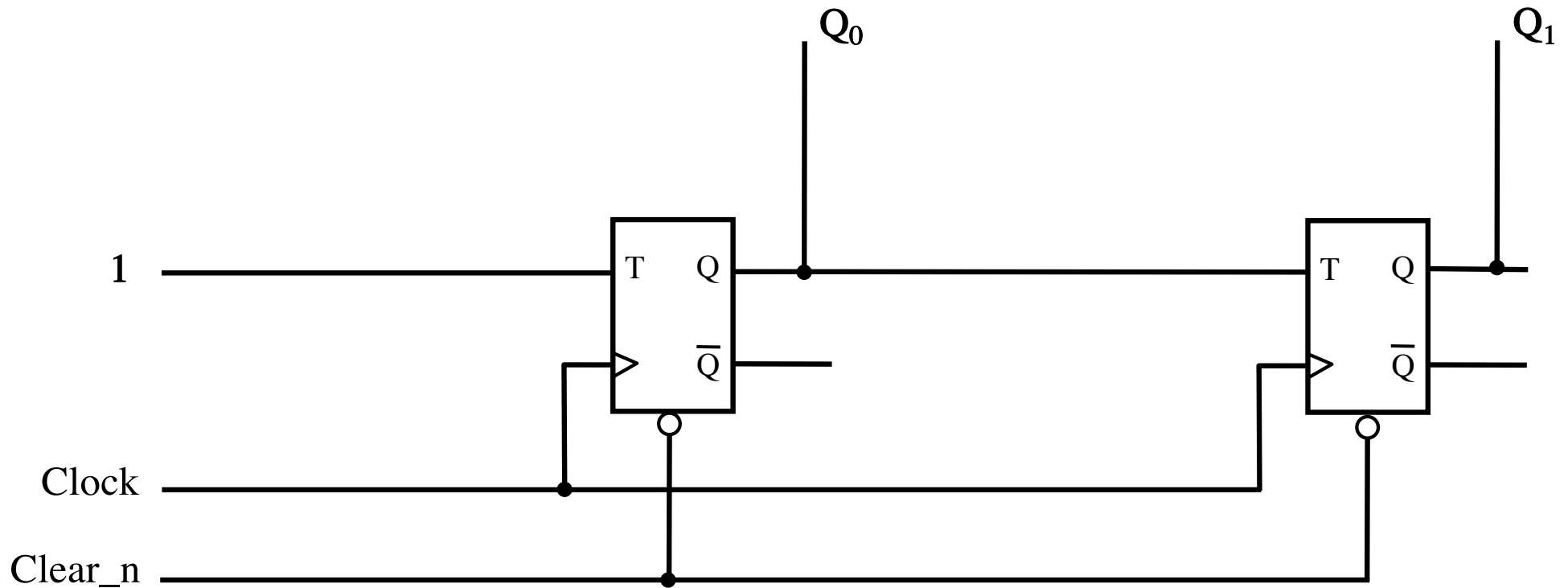
$$T_n = Q_0 Q_1 Q_2 \cdots Q_{n-1}$$

Synchronous v.s. Asynchronous Clear

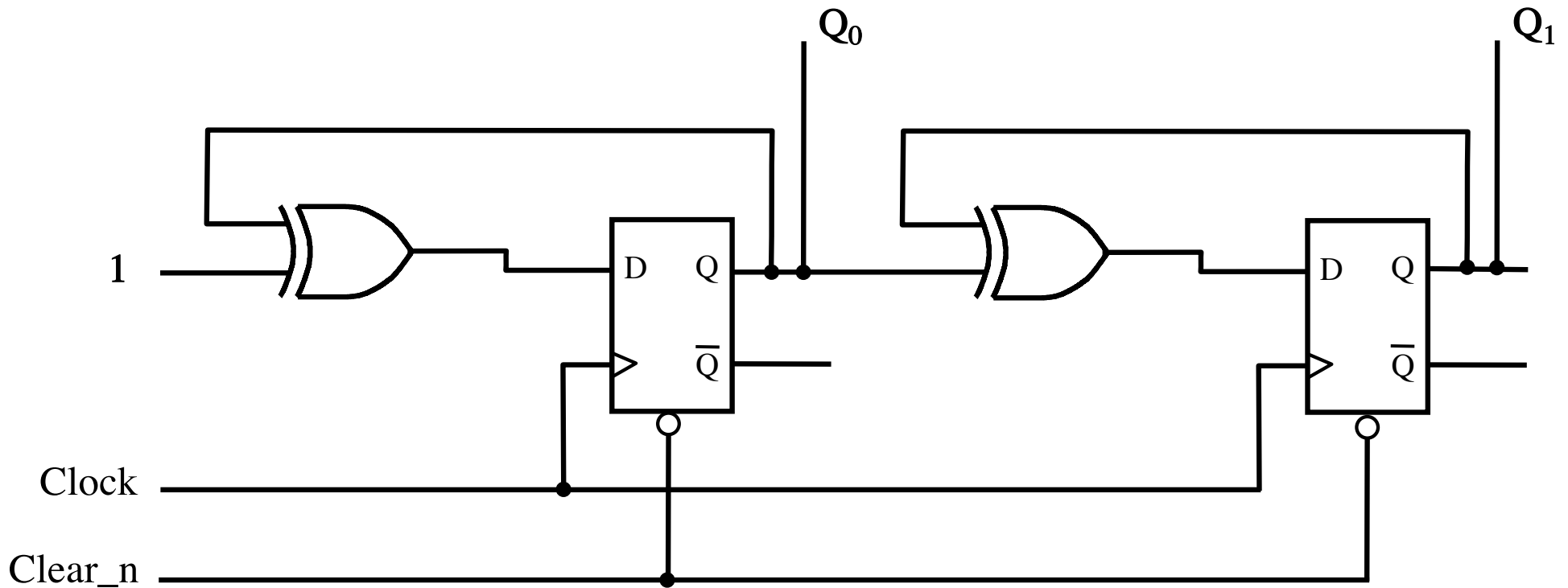
2-Bit Synchronous Up-Counter (without clear capability)



2-Bit Synchronous Up-Counter (with asynchronous clear)

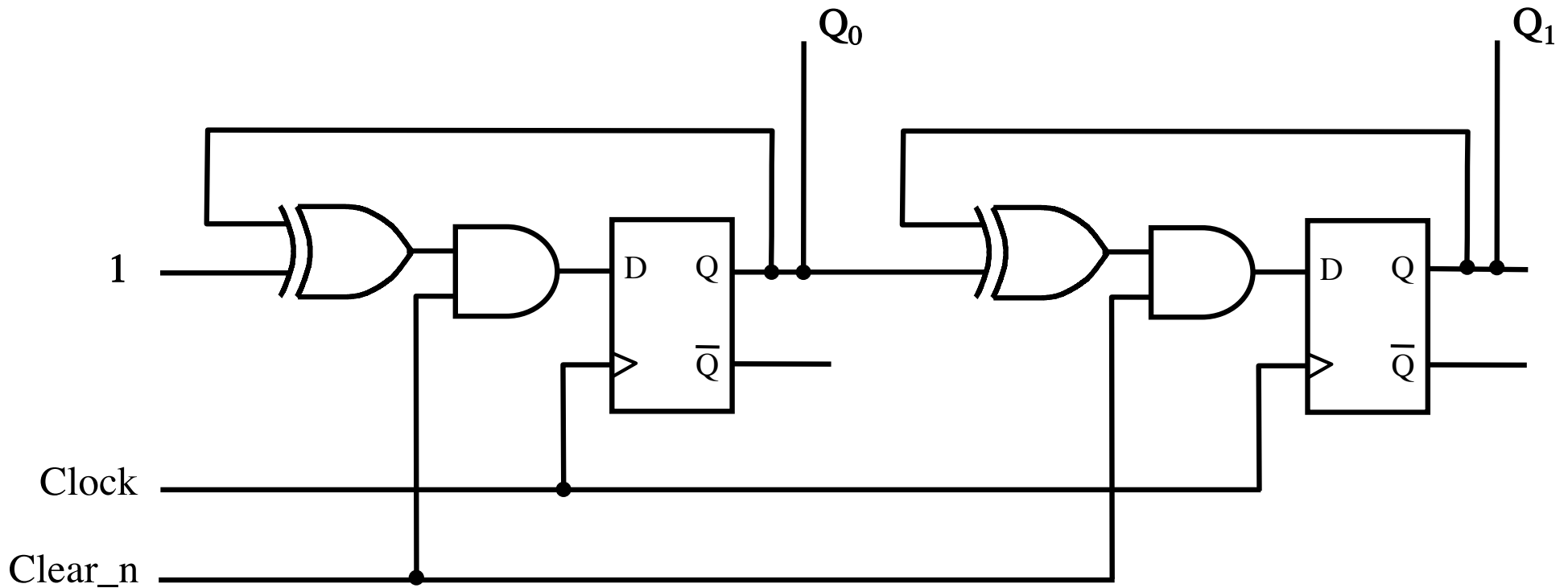


2-Bit Synchronous Up-Counter (with asynchronous clear)



This is the same circuit but uses D Flip-Flops.

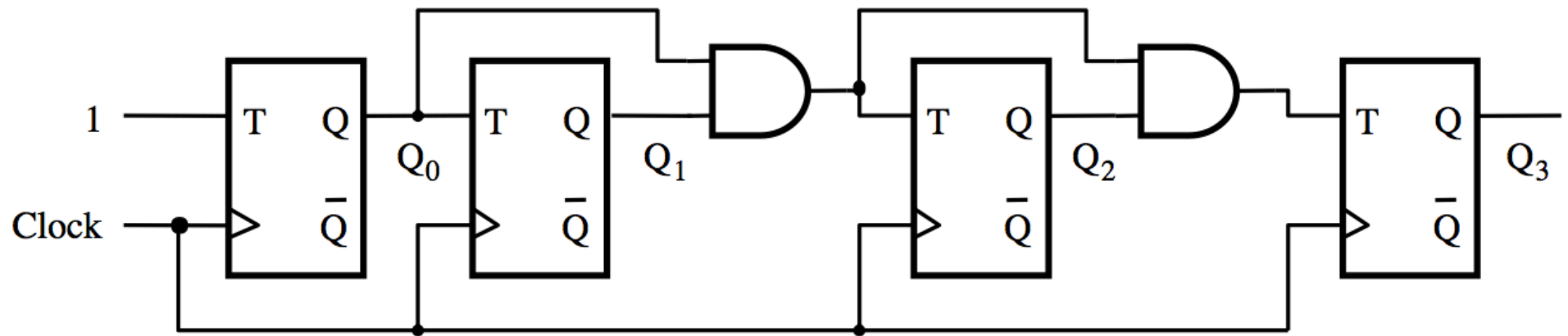
2-Bit Synchronous Up-Counter (with synchronous clear)



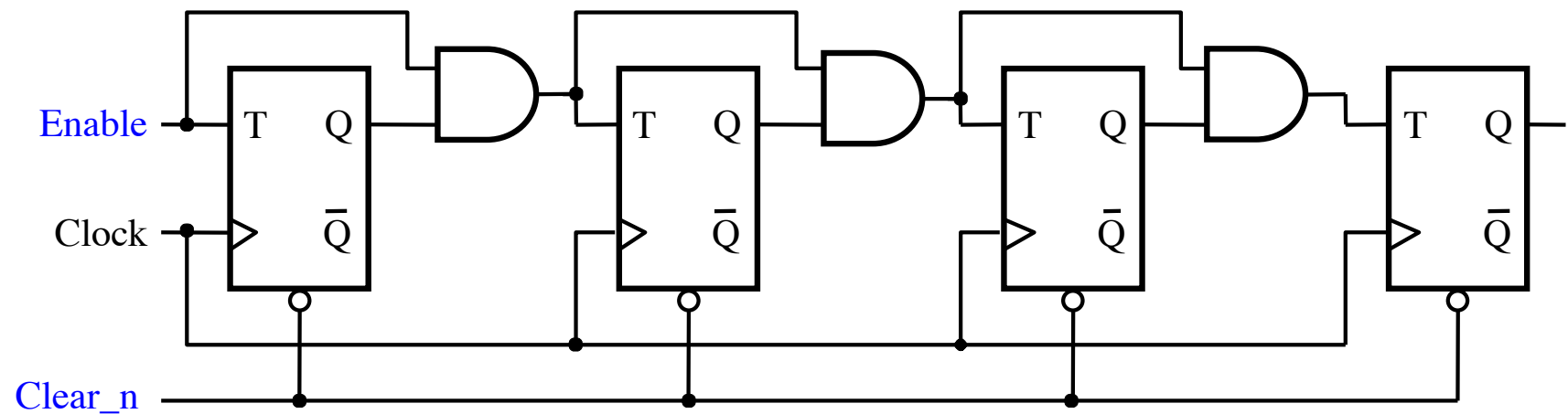
This counter can be cleared only on the positive clock edge.

Adding Enable Capability

A four-bit synchronous up-counter



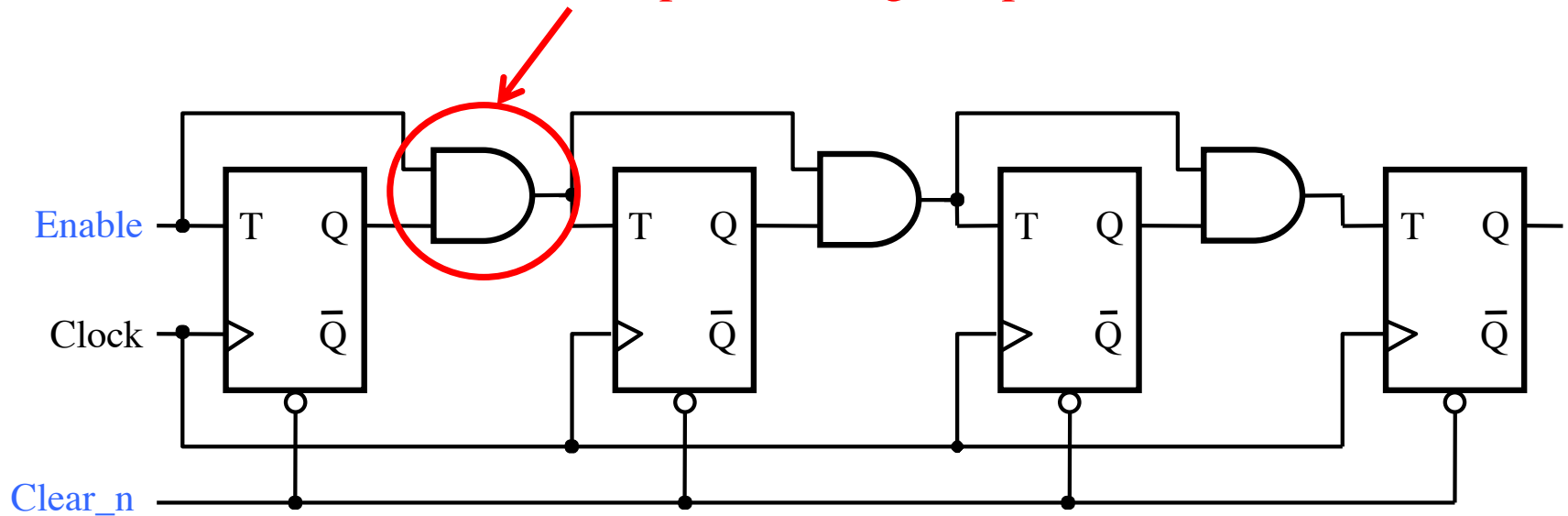
Inclusion of Enable and Clear Capability



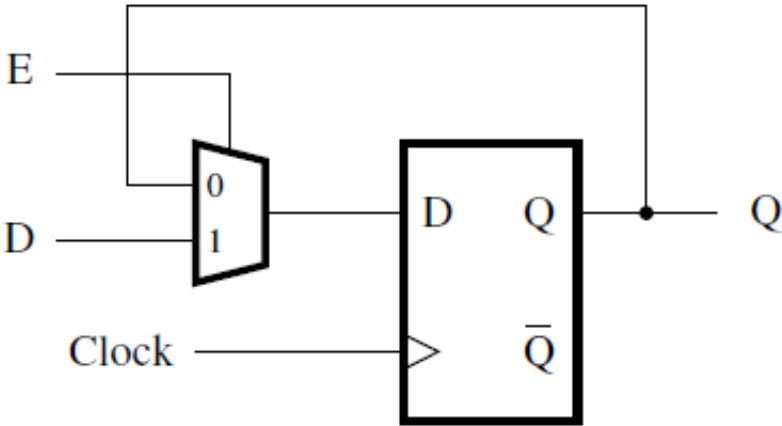
[Figure 5.22 from the textbook]

Inclusion of Enable and Clear Capability

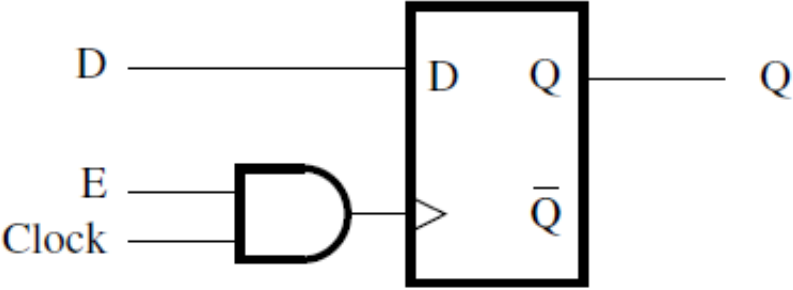
This is the new thing relative to the previous figure, plus the clear_n line



Providing an enable input for a D flip-flop



(a) Using a multiplexer

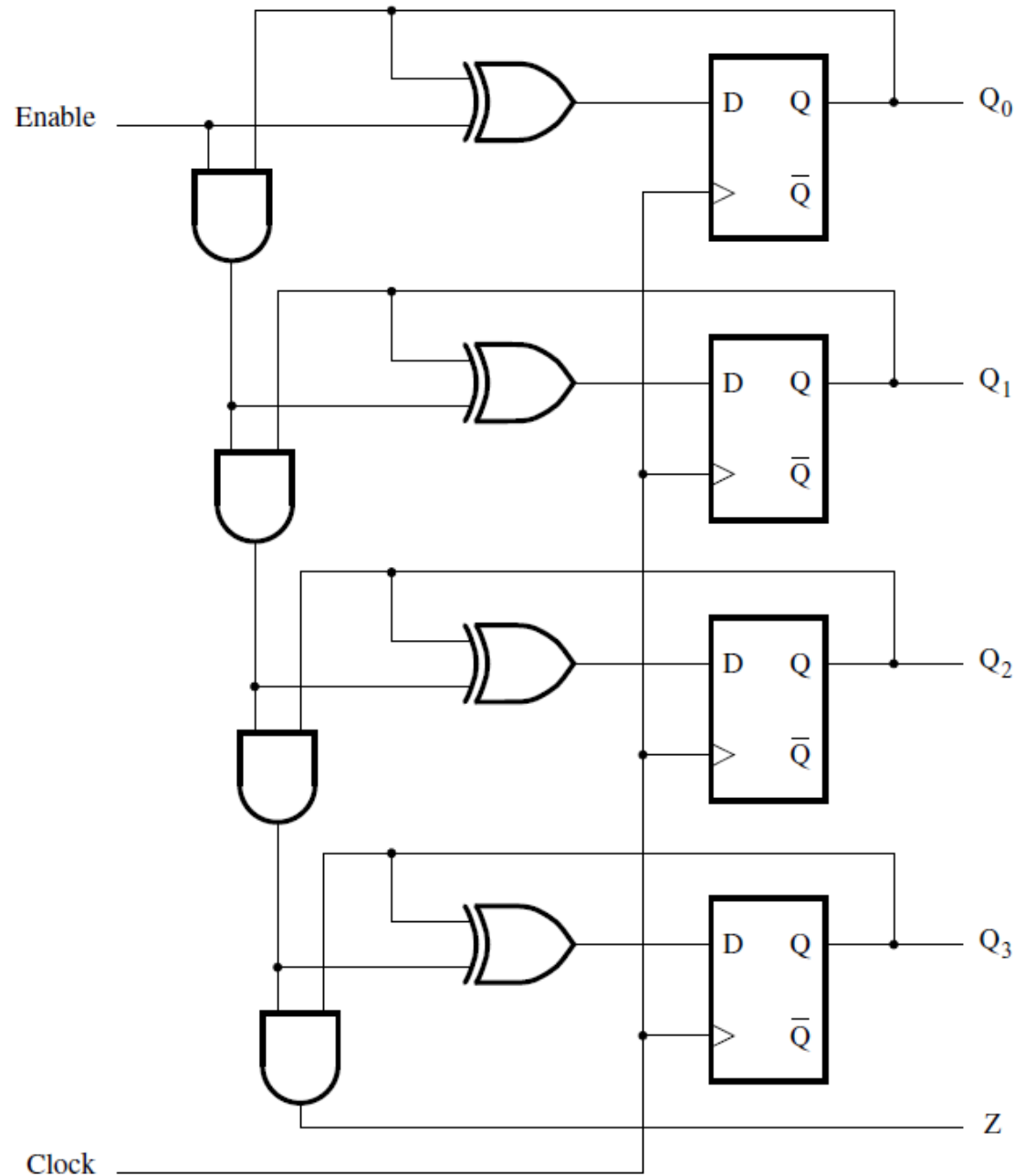


(b) Clock gating

[Figure 5.56 from the textbook]

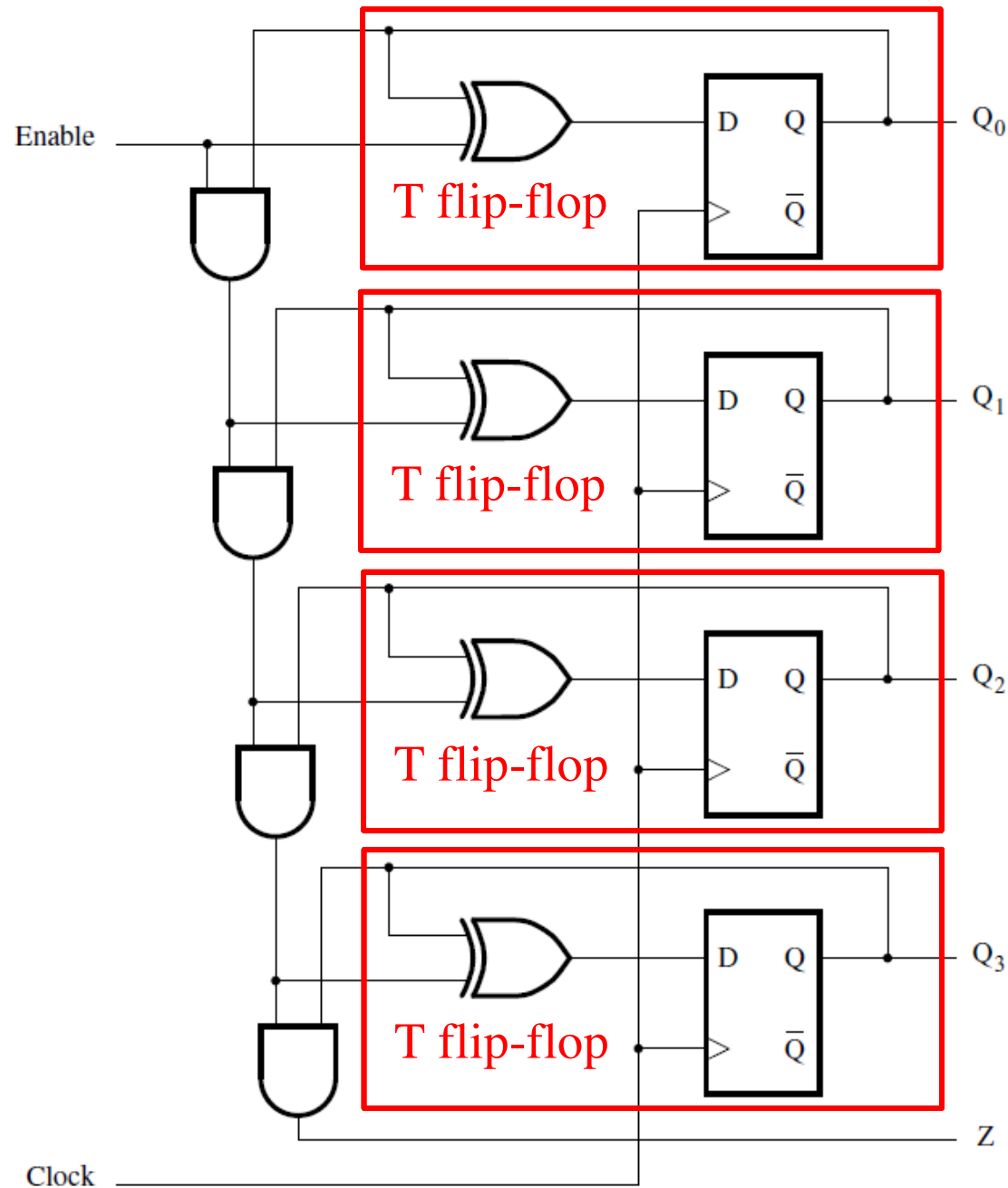
Synchronous Counter (with D Flip-Flops)

A 4-bit up-counter with D flip-flops



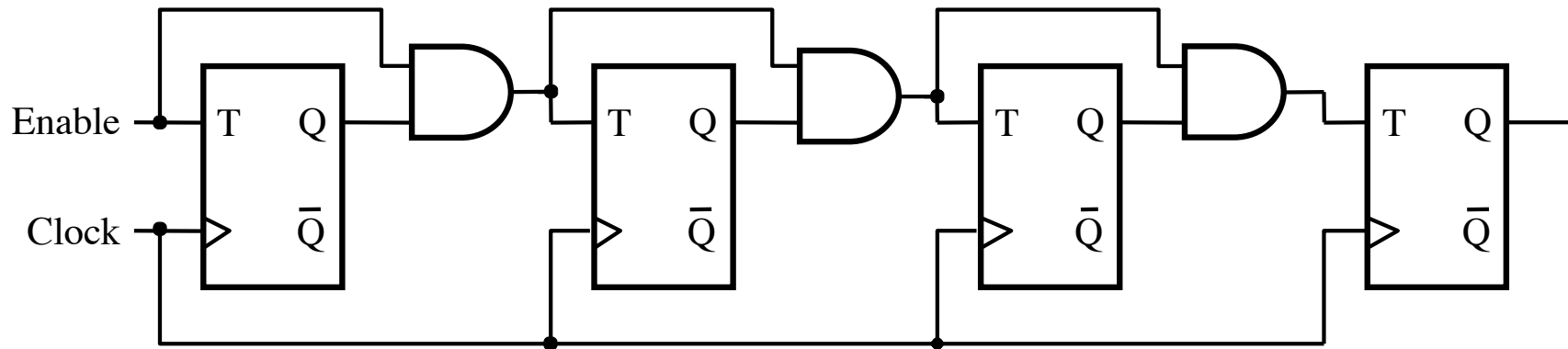
[Figure 5.23 from the textbook]

A 4-bit up-counter with D flip-flops

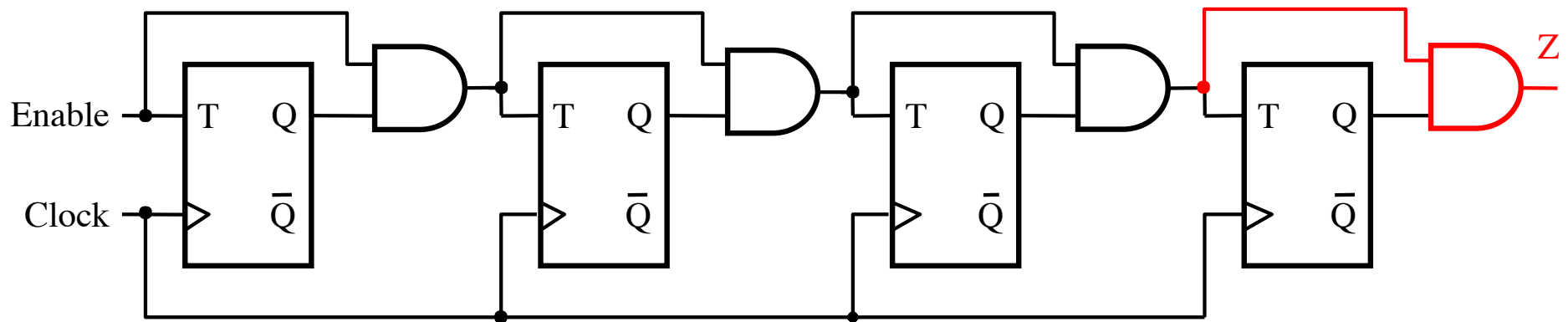


[Figure 5.23 from the textbook]

Equivalent to this circuit with T flip-flops



Equivalent to this circuit with T flip-flops

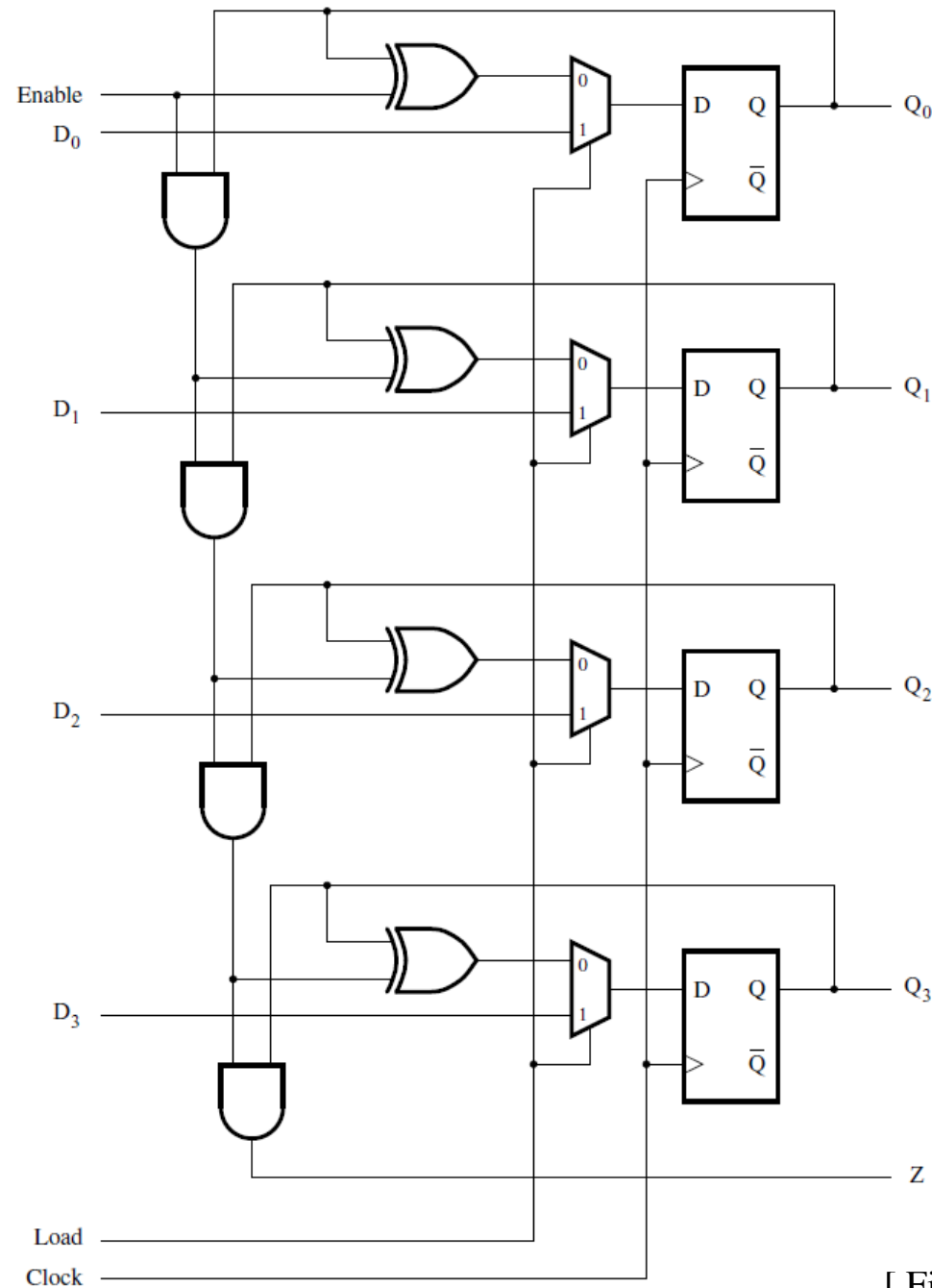


But has one extra output called Z, which can be used to connect two 4-bit counters to make an 8-bit counter.

When $Z=1$ the counter will go to 0000 on the next clock edge, i.e., the outputs of all flip-flops are currently 1 (maximum count value).

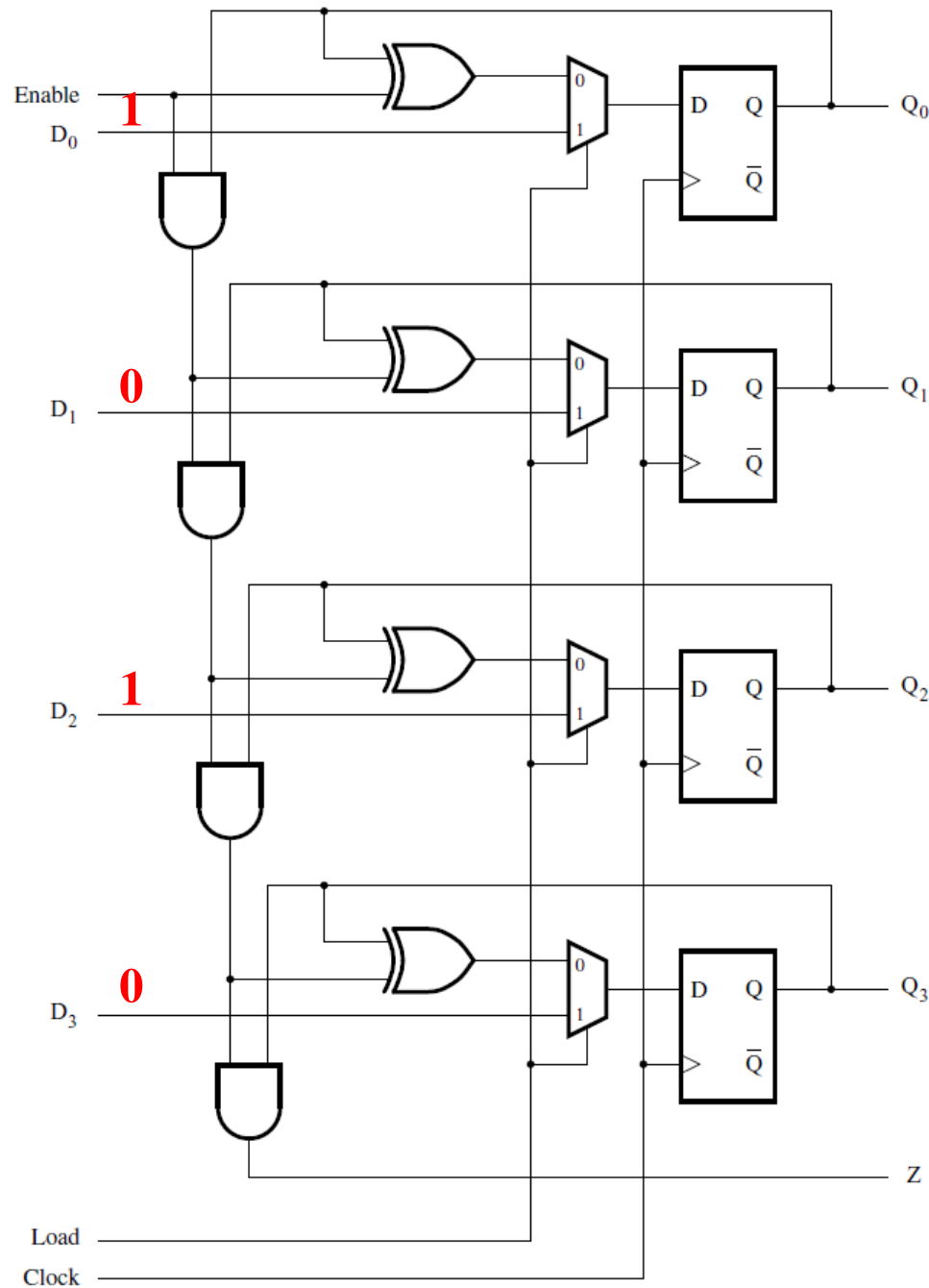
Counters with Parallel Load

A counter with parallel-load capability



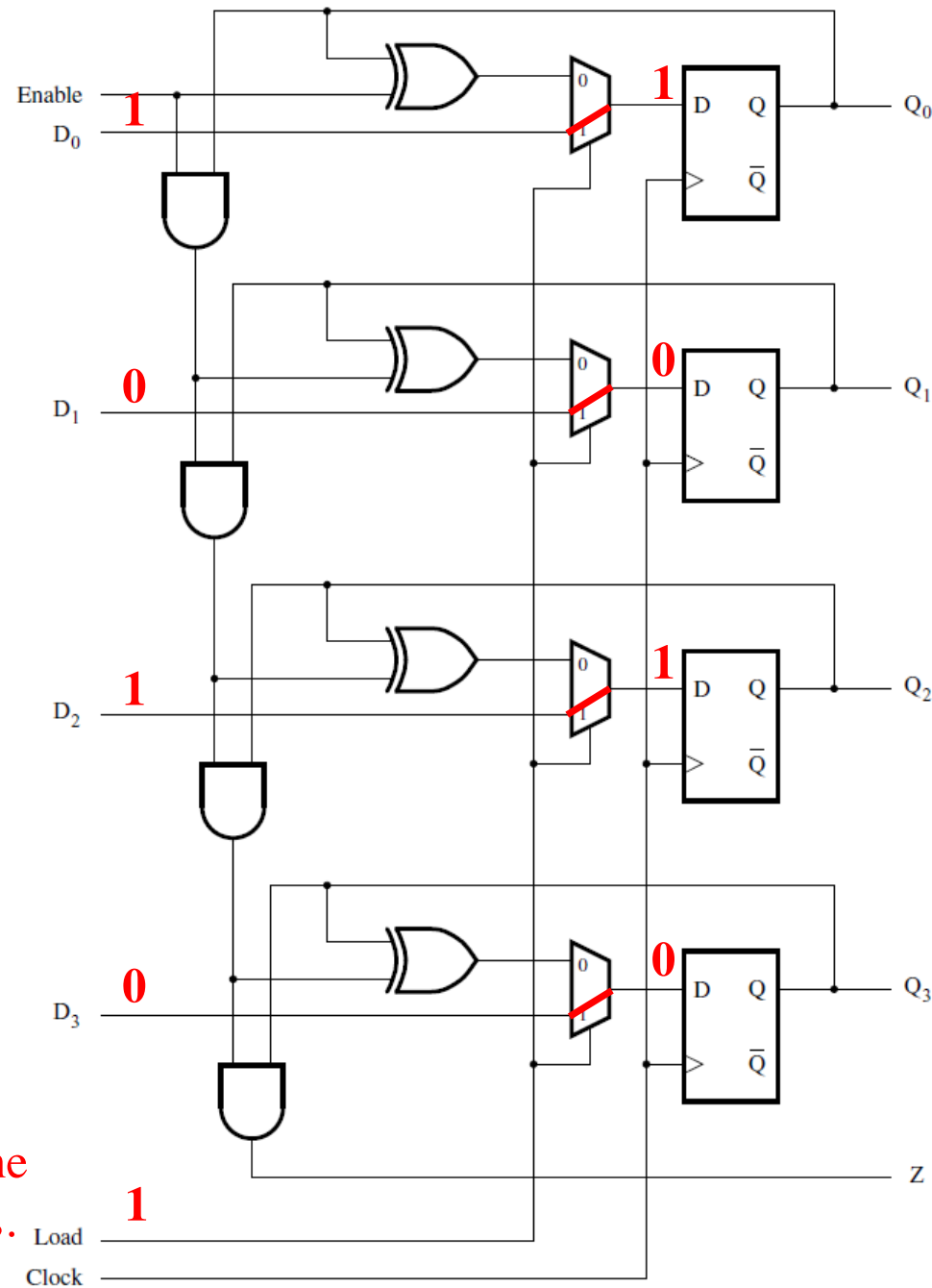
[Figure 5.24 from the textbook]

How to load the initial count value



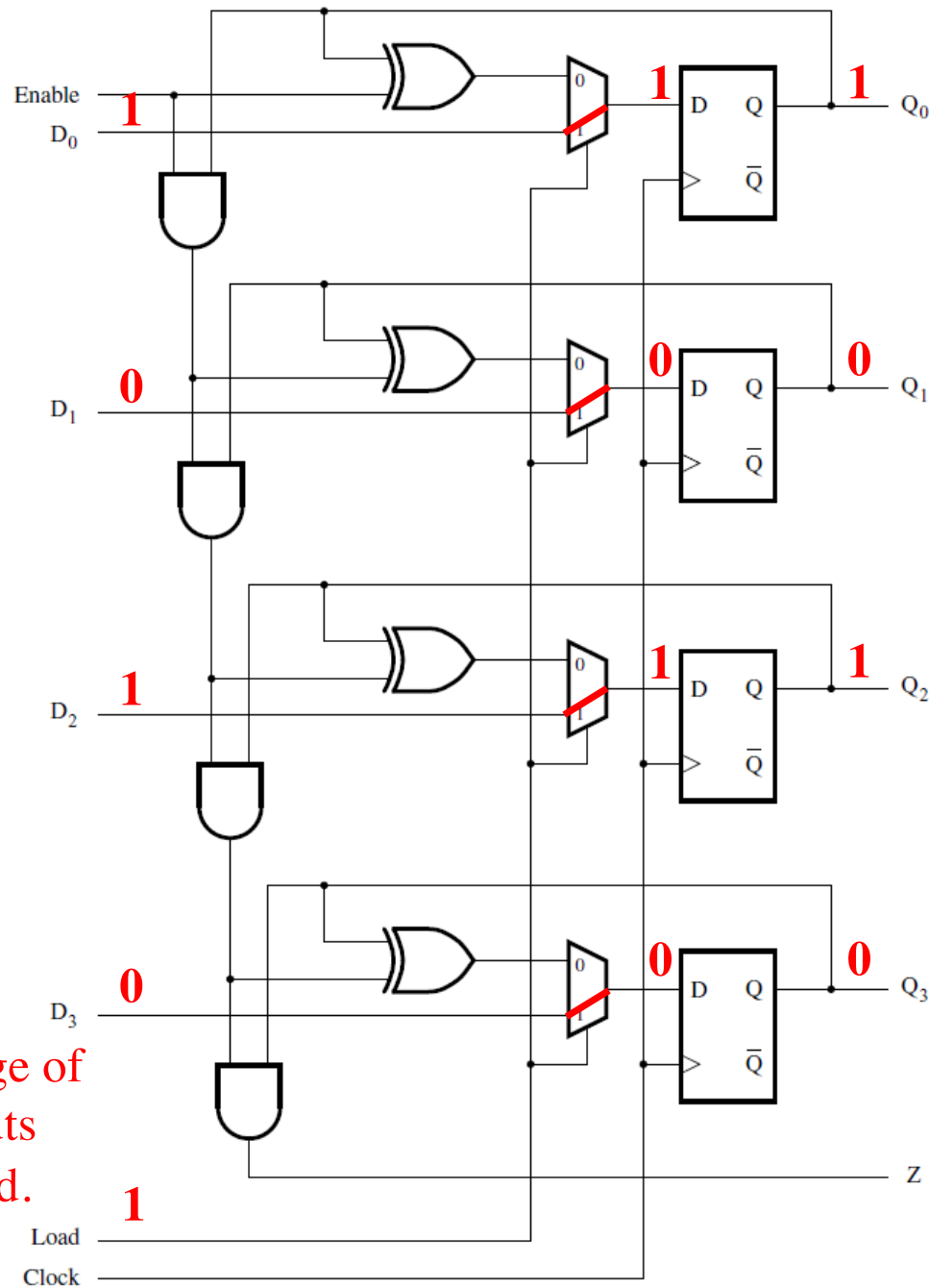
Set the initial count on the parallel load lines (in this case 5).

How to zero a counter



Set "Load" to 1, to open the "1" line of the multiplexers.

How to zero a counter



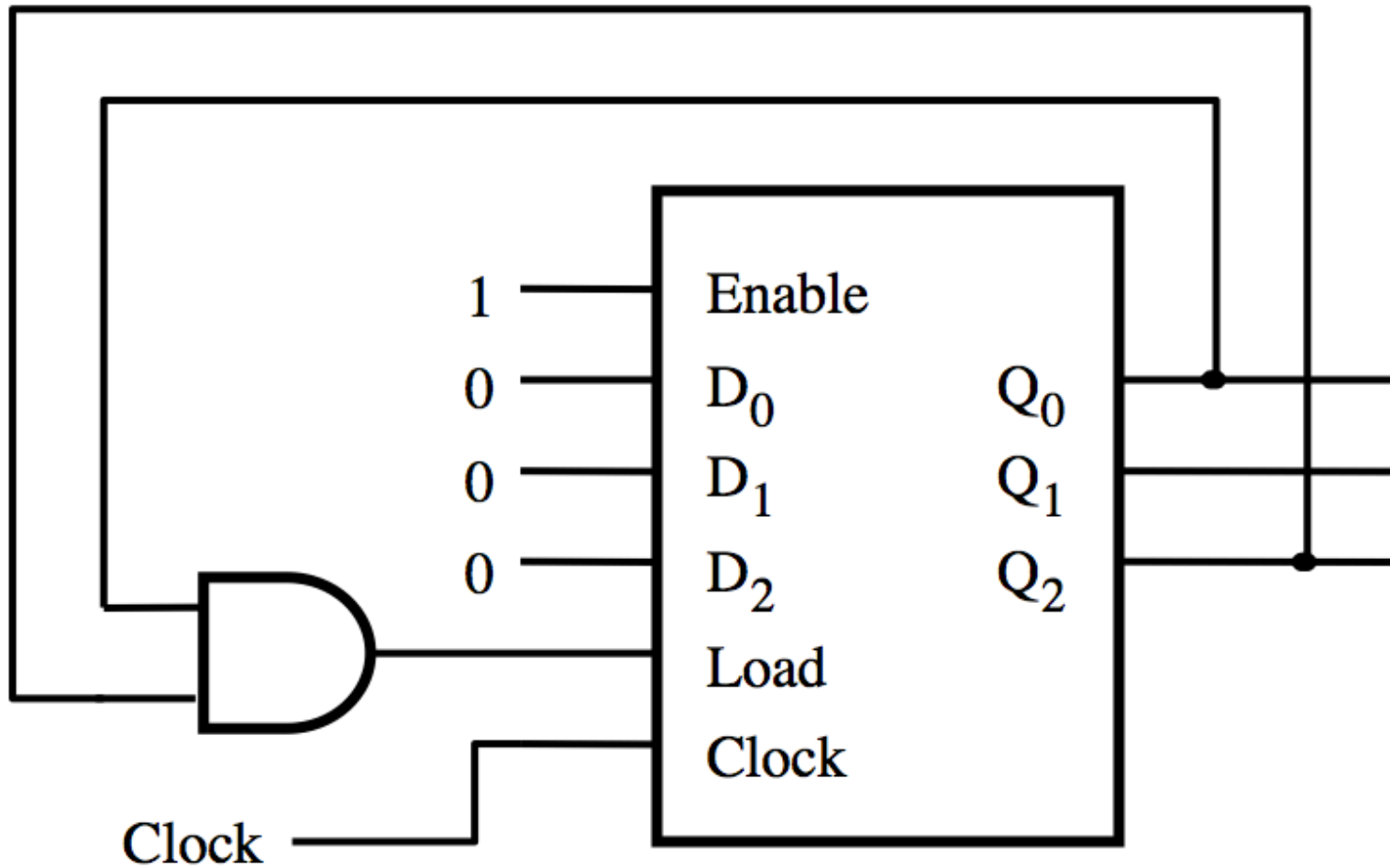
When the next positive edge of the clock arrives, the outputs of the flip-flops are updated.

Reset Synchronization

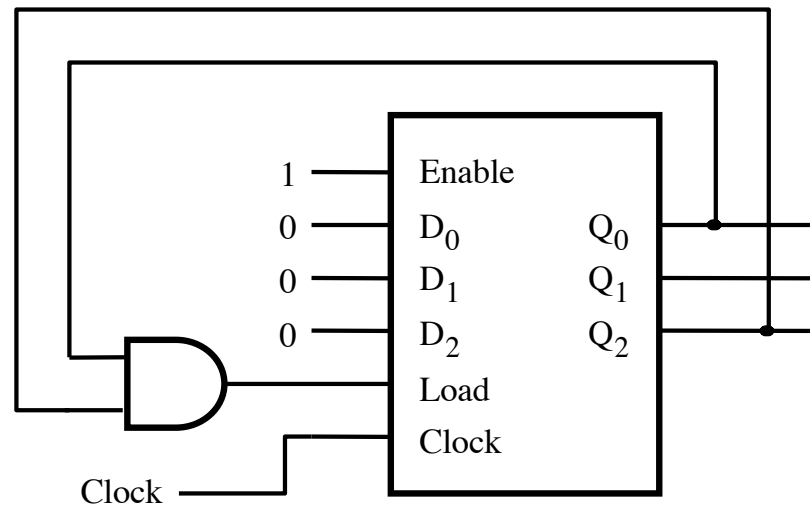
Motivation

- **An n-bit counter counts from 0, 1, ..., 2^n-1**
- **For example a 3-bit counter counts up as follow**
 - **0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, ...**
- **What if we want it to count like this**
 - **0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, ...**
- **In other words, what is the cycle is not a power of 2?**

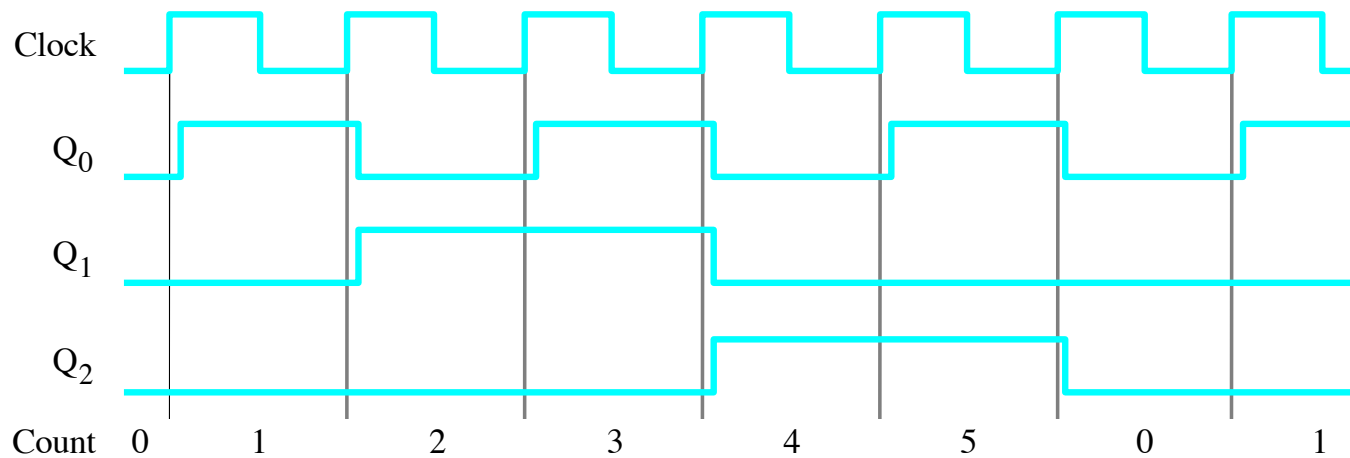
What does this circuit do?



A modulo-6 counter with synchronous reset

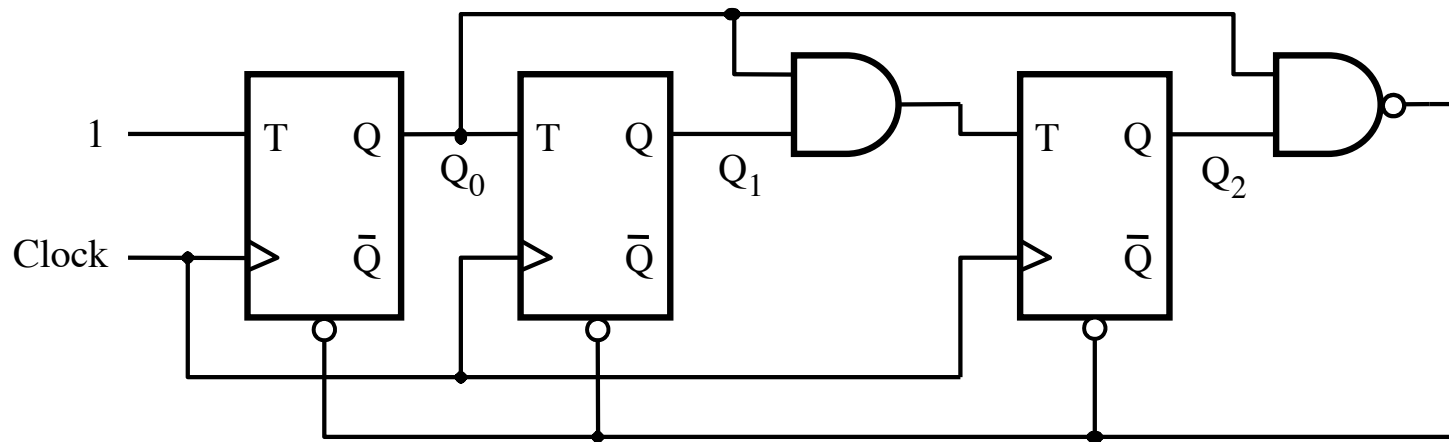


(a) Circuit

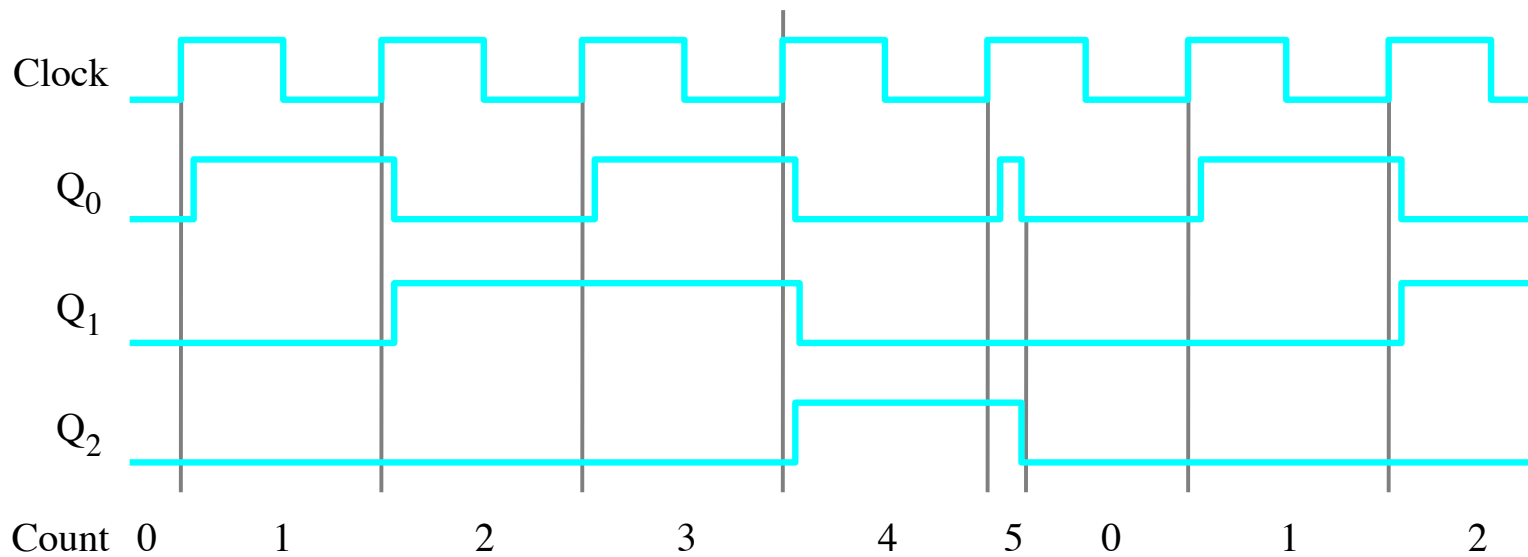


(b) Timing diagram

A modulo-6 counter with asynchronous reset

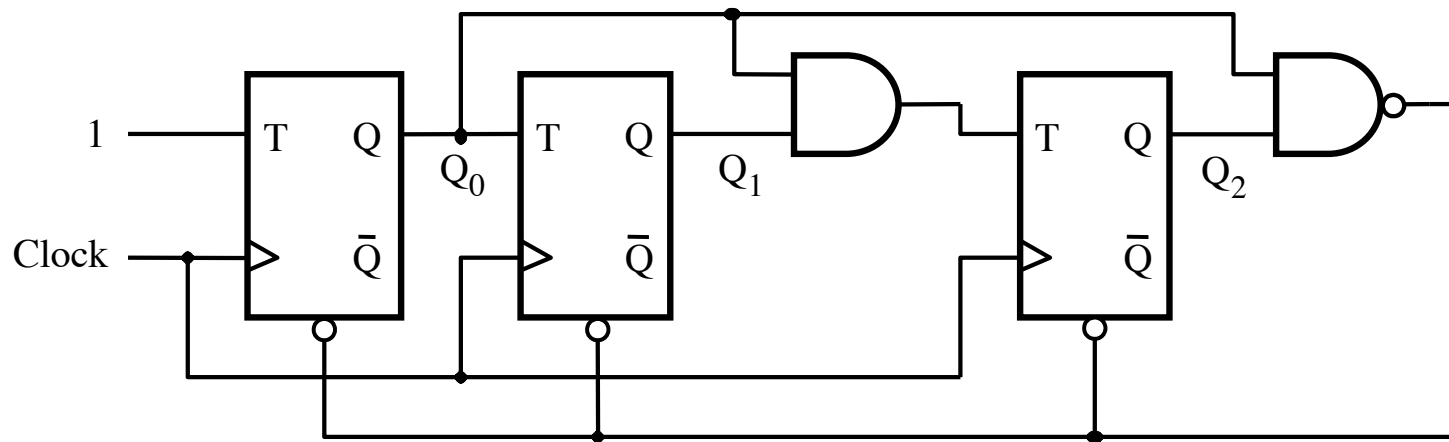


(a) Circuit



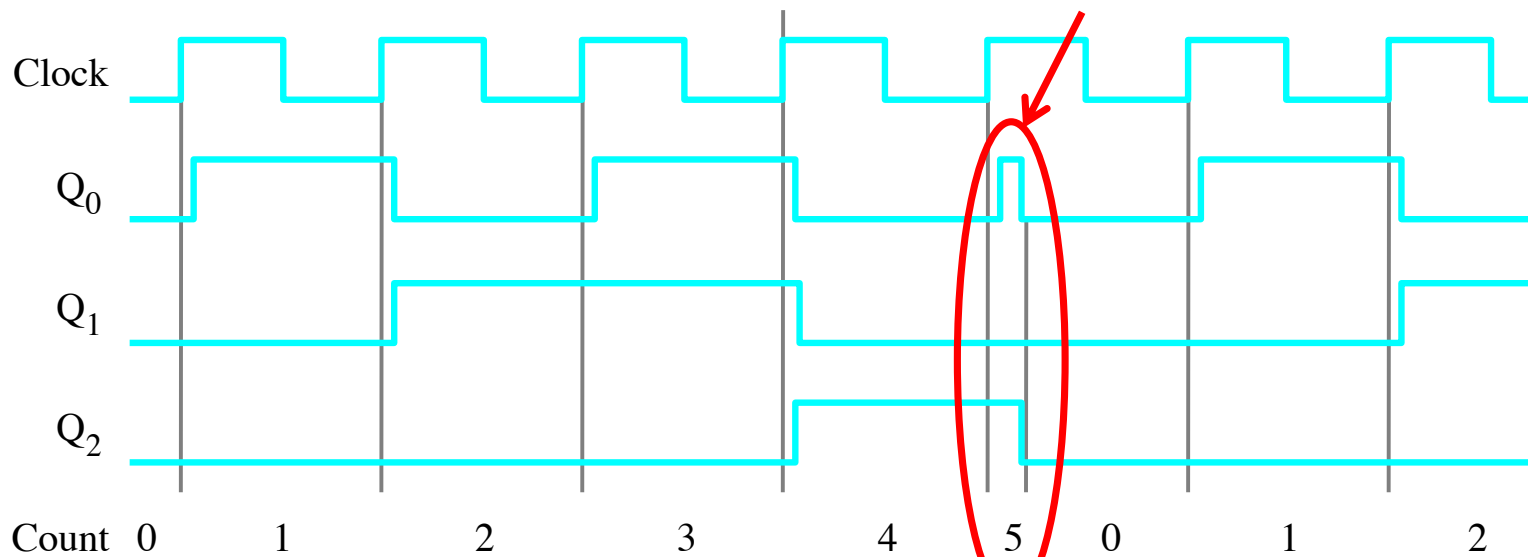
(b) Timing diagram

A modulo-6 counter with asynchronous reset



(a) Circuit

The number 5 is displayed for a very short amount of time



(b) Timing diagram

Questions?

THE END