



# **CprE 281: Digital Logic**

**Instructor: Alexander Stoytchev**

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

# Counters & Solved Problems

*CprE 281: Digital Logic*  
*Iowa State University, Ames, IA*  
*Copyright © 2013*

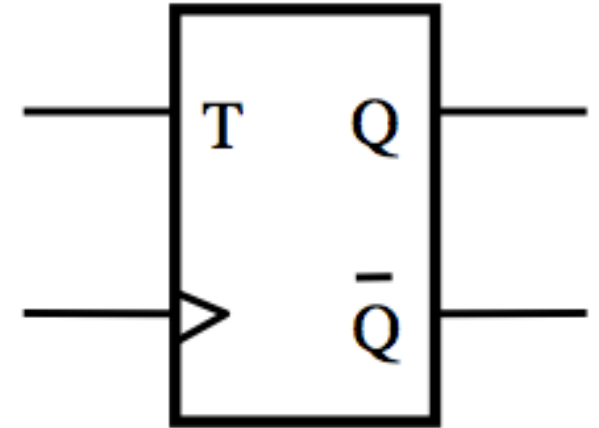
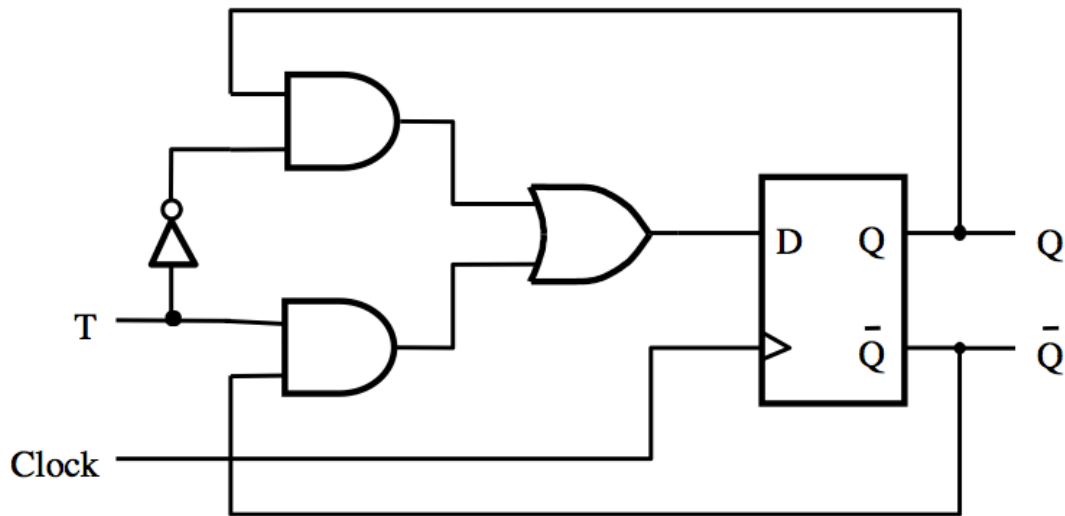
# **Administrative Stuff**

- **Homework 9 is due today**
- **Homework 10 is due on Wednesday Nov 8 @ 10pm**

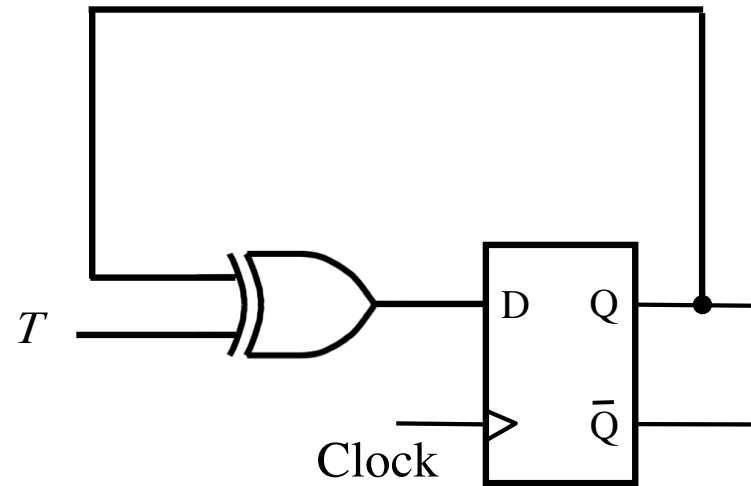
# Counters

# T Flip-Flop

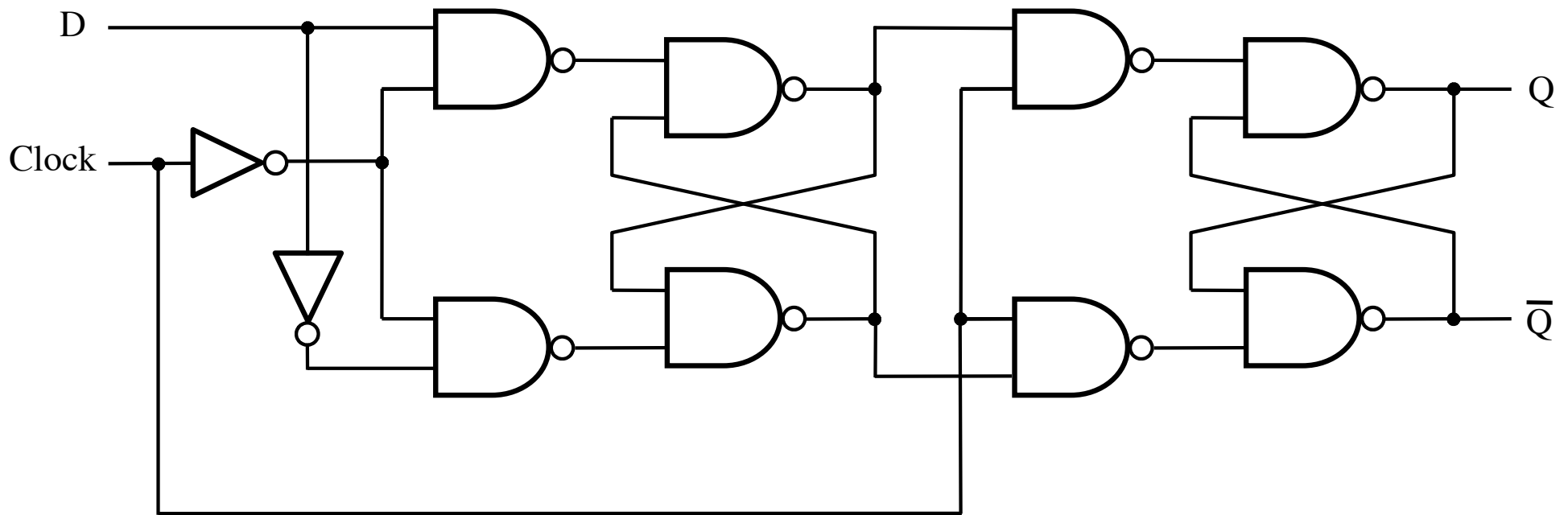
(circuit and graphical symbol)



# Yet Another Way to Draw a T Flip-Flop

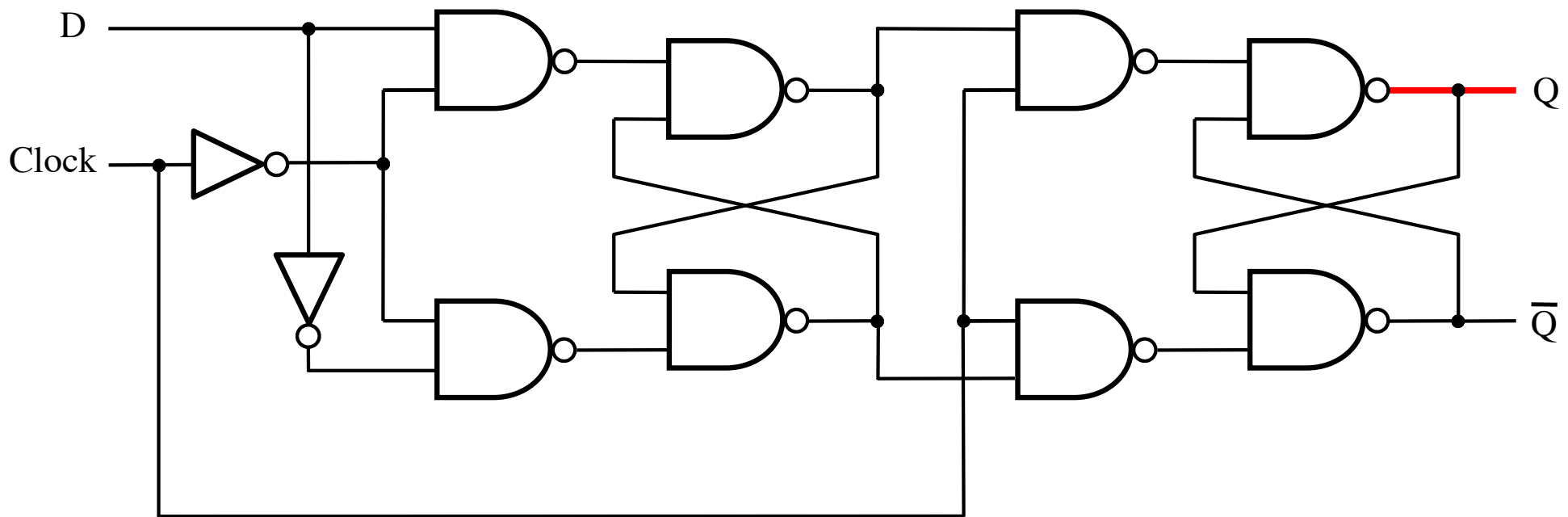


# The Complete Wiring Diagram for a Positive-Edge-Triggered D Flip-Flop



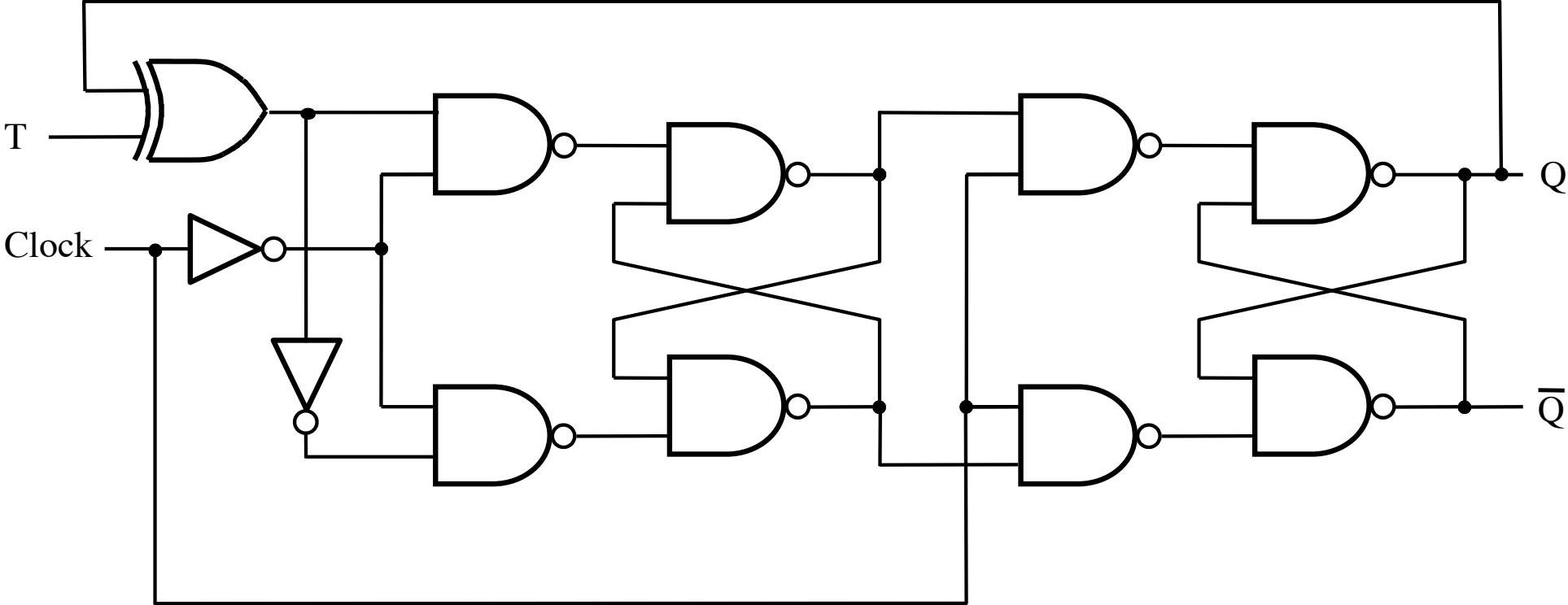
# The Complete Wiring Diagram for a Positive-Edge-Triggered D Flip-Flop

We need all of this just to store 1 bit!

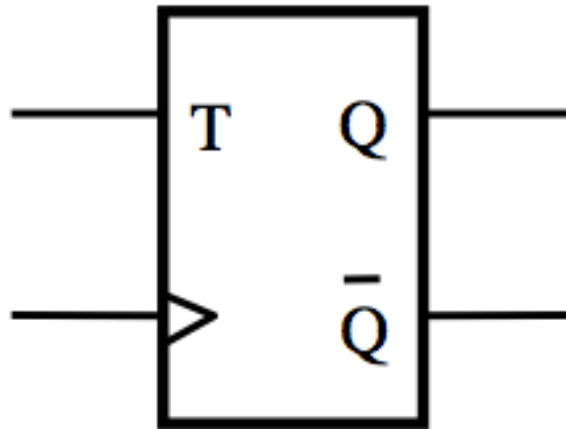




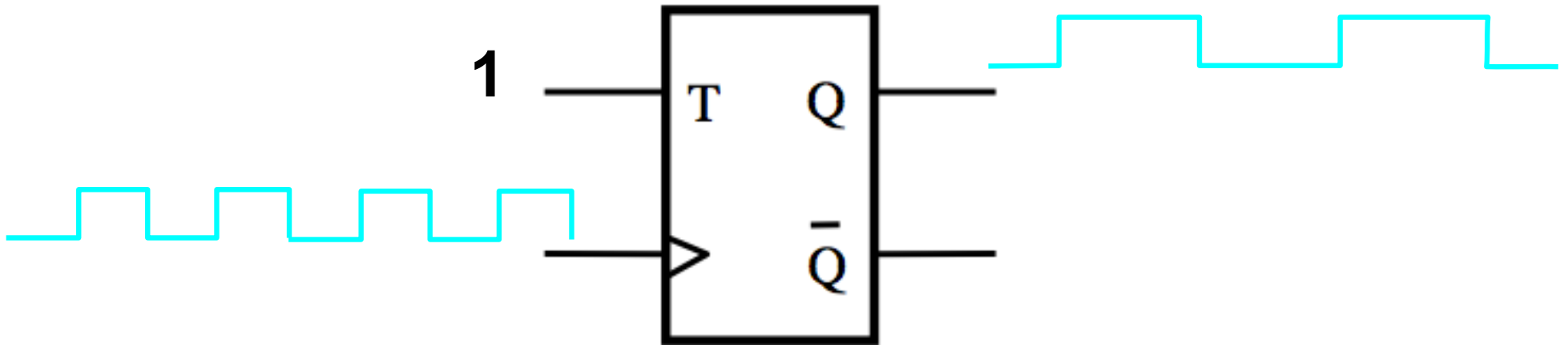
# The Complete Wiring Diagram for a Positive-Edge-Triggered T Flip-Flop



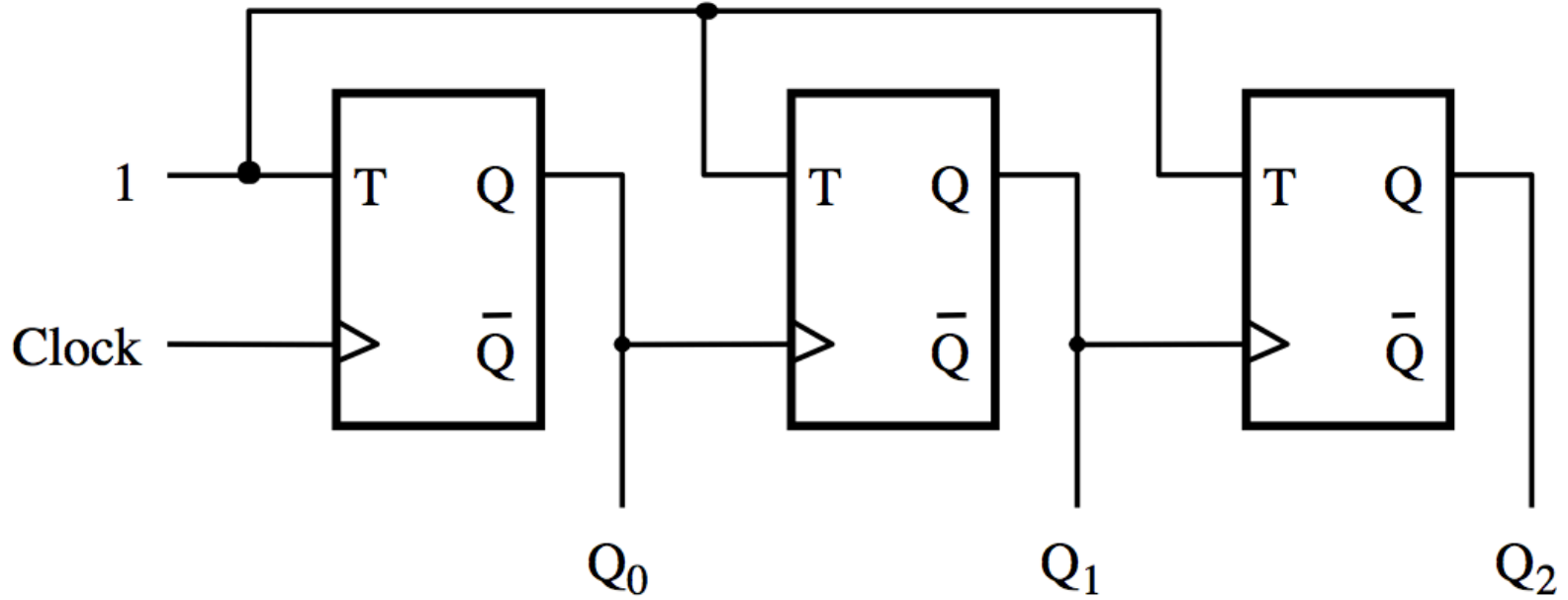
**The output of the T Flip-Flop  
divides the frequency of the clock by 2**



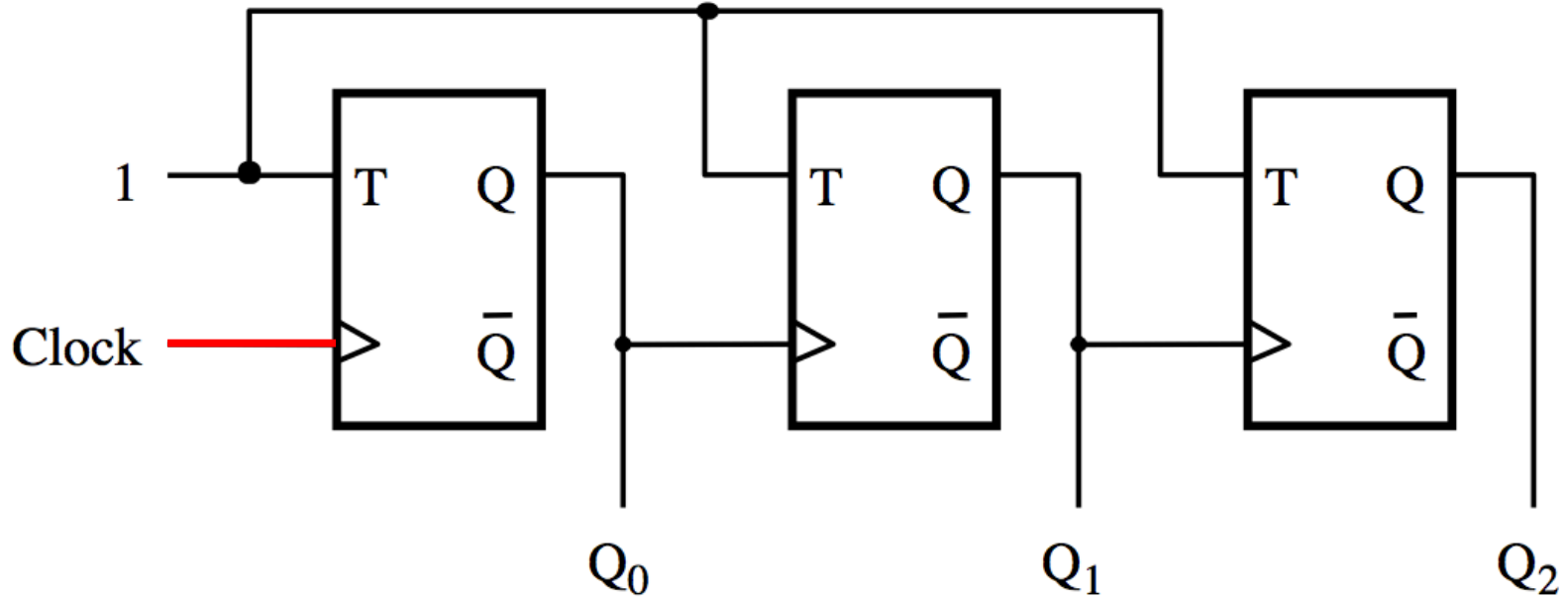
**The output of the T Flip-Flop  
divides the frequency of the clock by 2**



# A three-bit down-counter

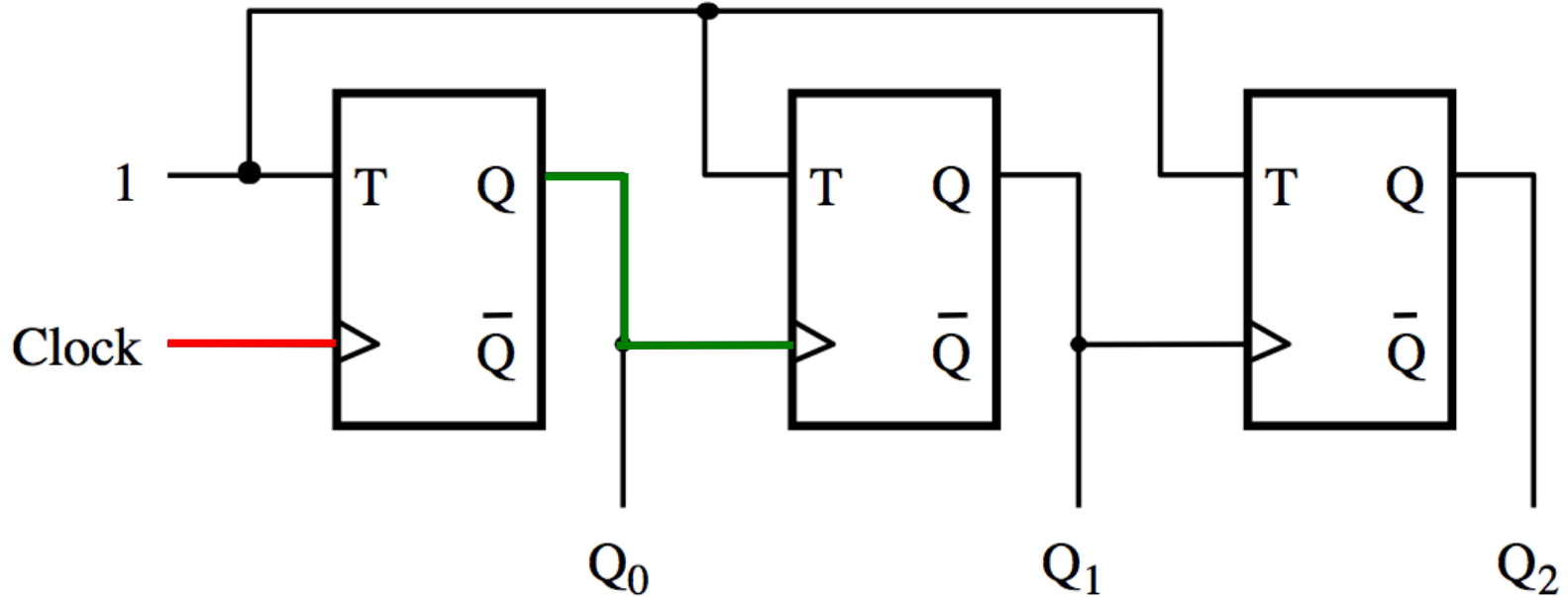


# A three-bit down-counter



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

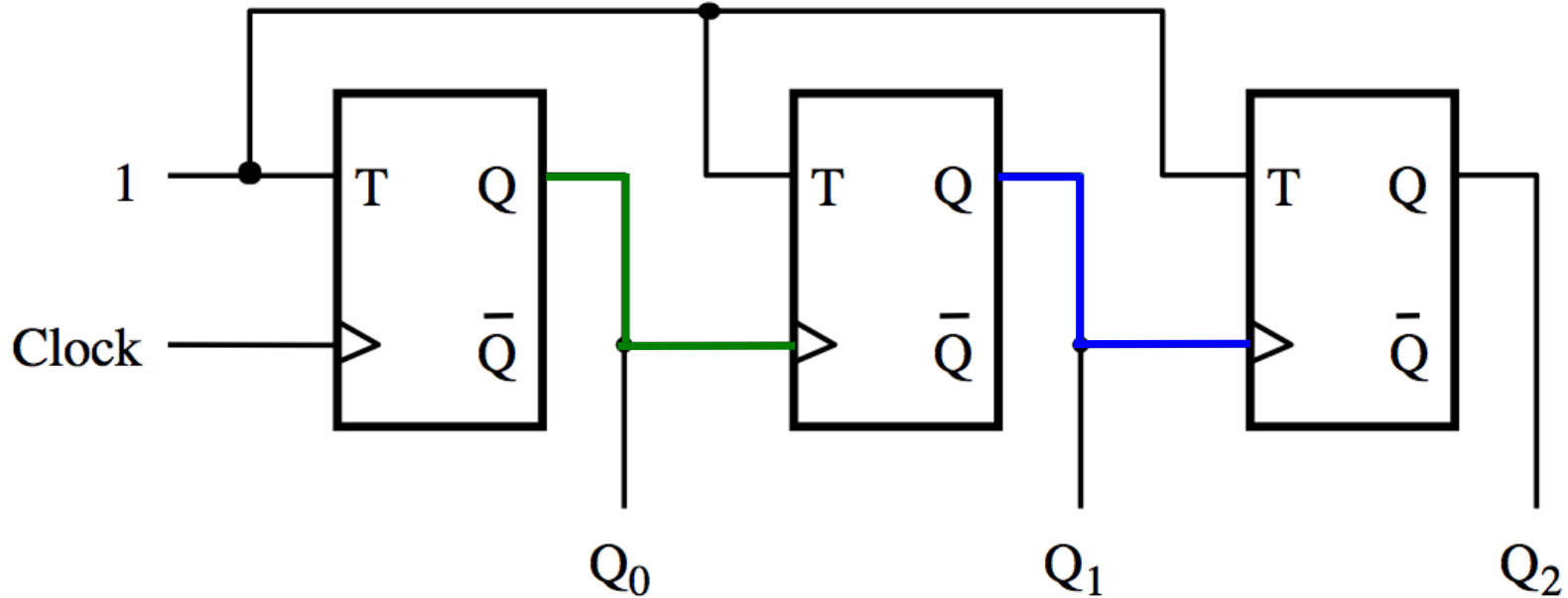
# A three-bit down-counter



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

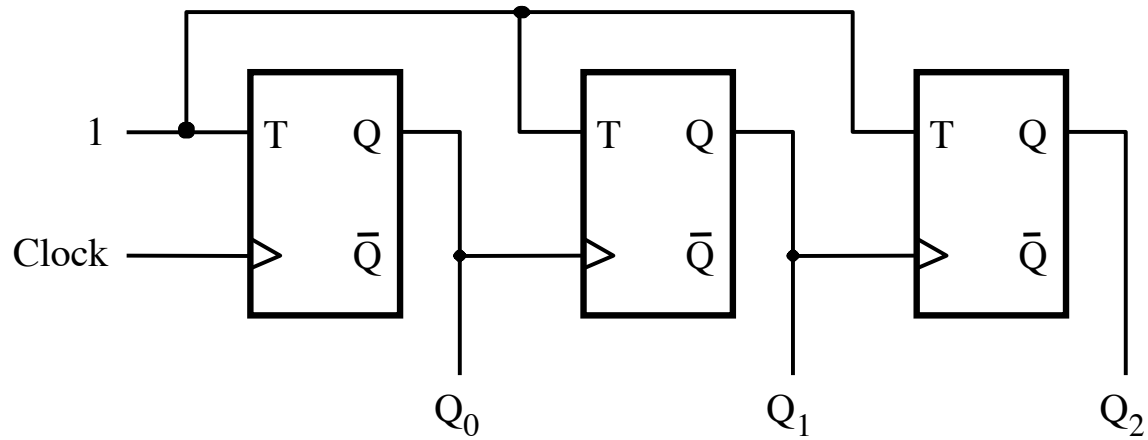


The first flip-flop changes on the positive edge of the clock

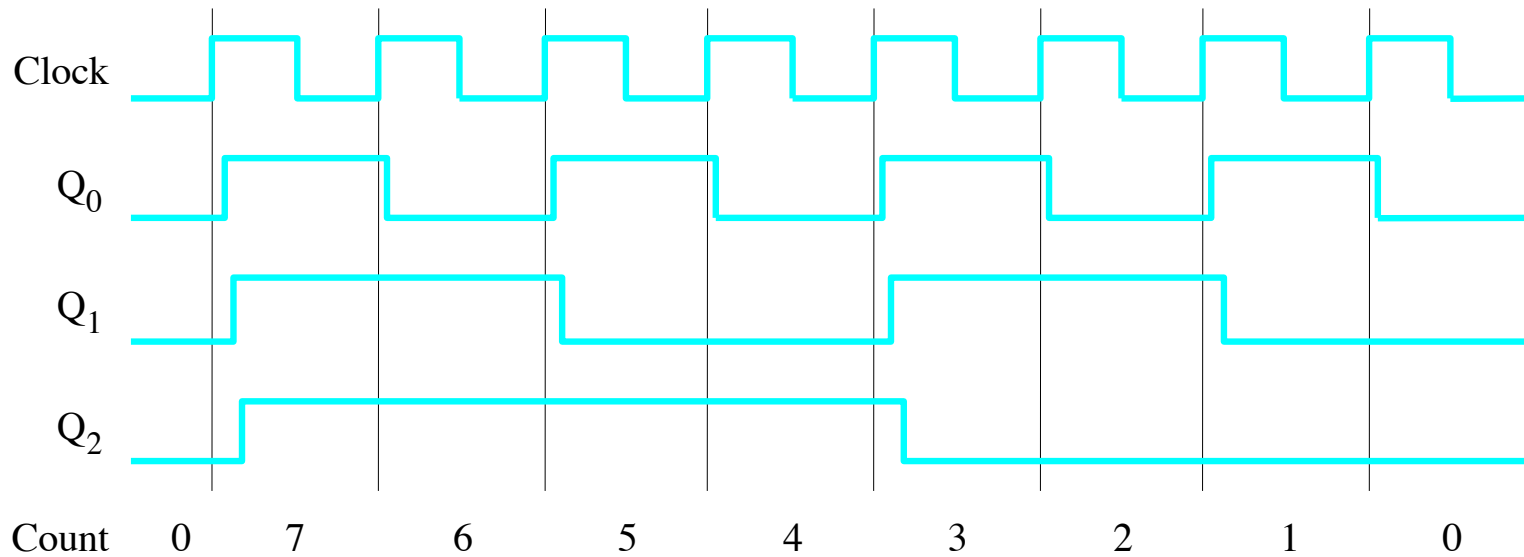
The second flip-flop changes on the positive edge of Q<sub>0</sub>

The third flip-flop changes on the positive edge of Q<sub>1</sub>

# A three-bit down-counter



(a) Circuit

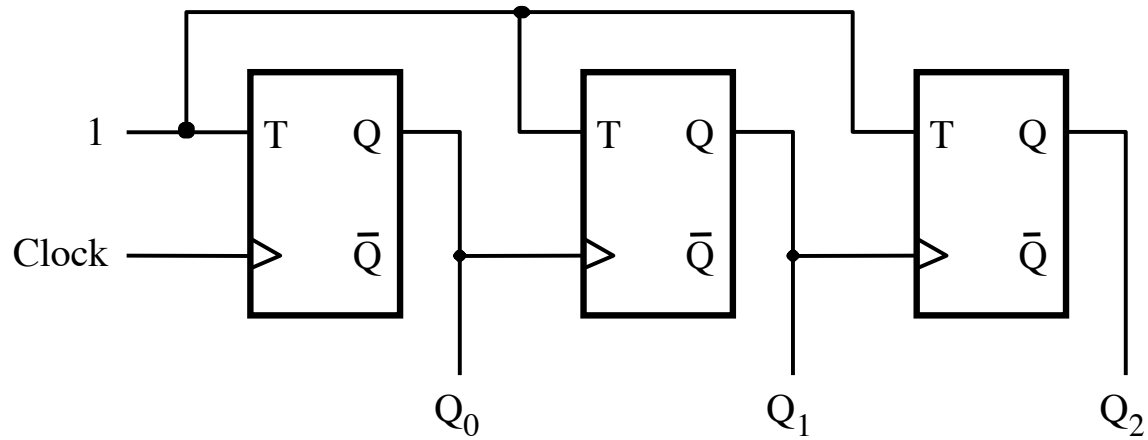


(b) Timing diagram

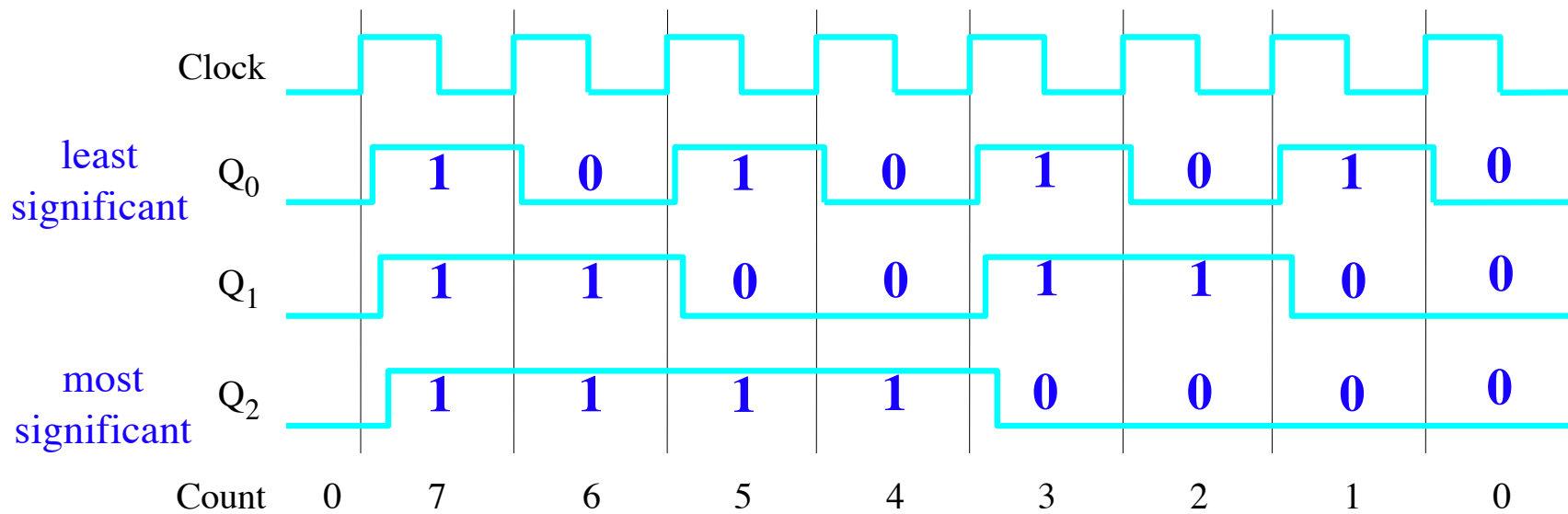
[ Figure 5.20 from the textbook ]



# A three-bit down-counter

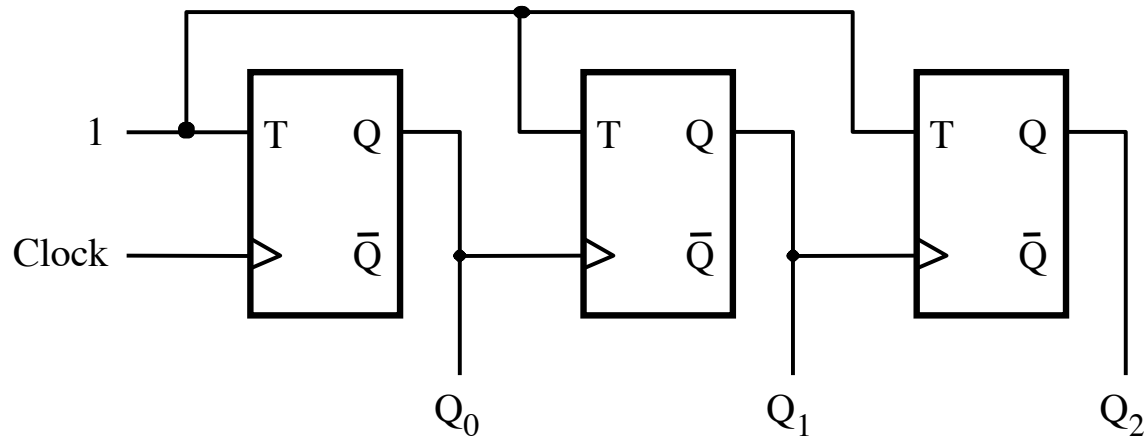


(a) Circuit

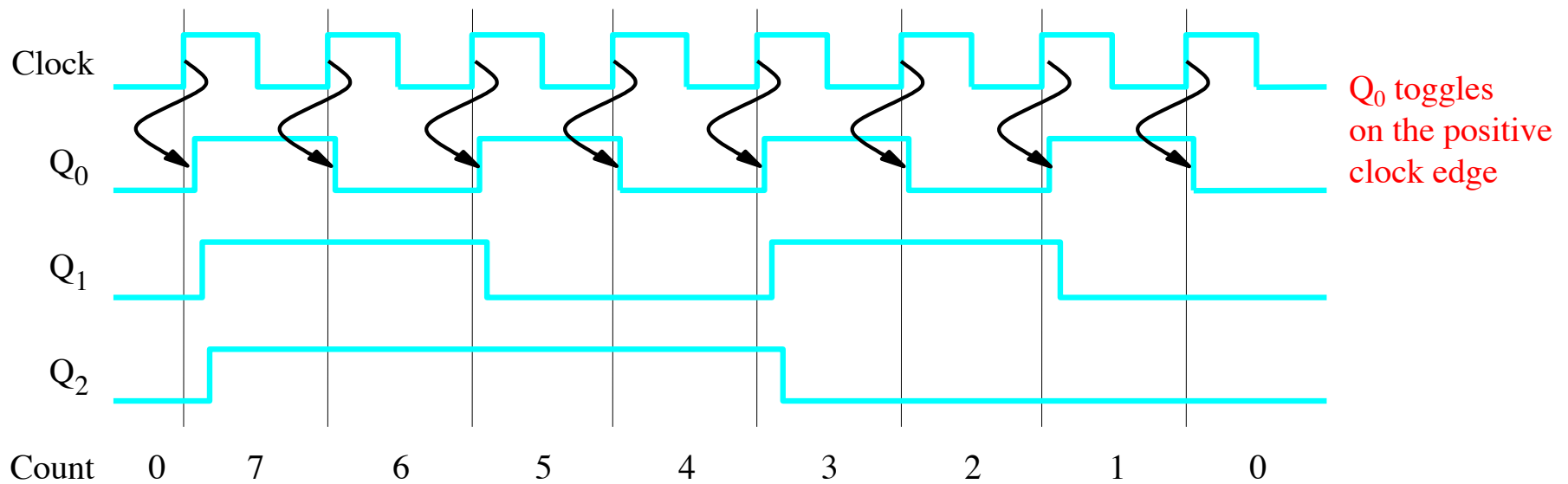


(b) Timing diagram

# A three-bit down-counter

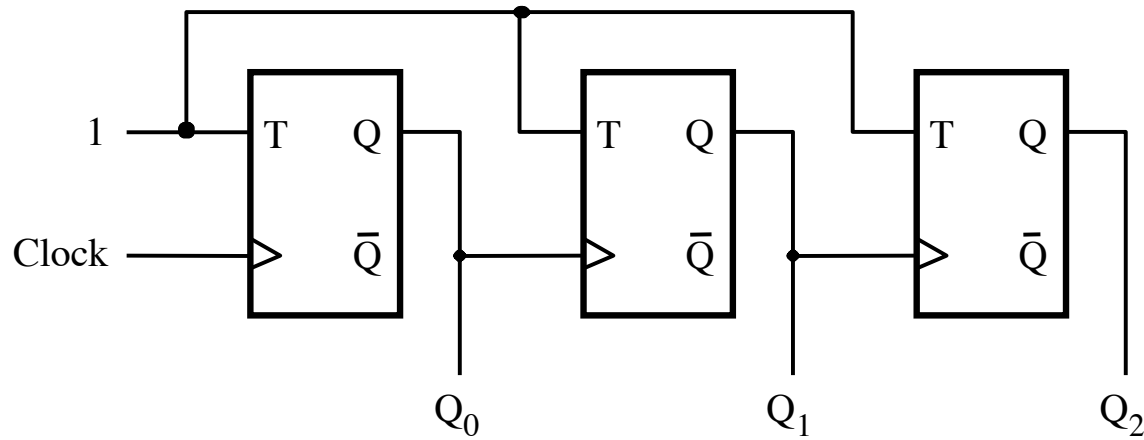


(a) Circuit

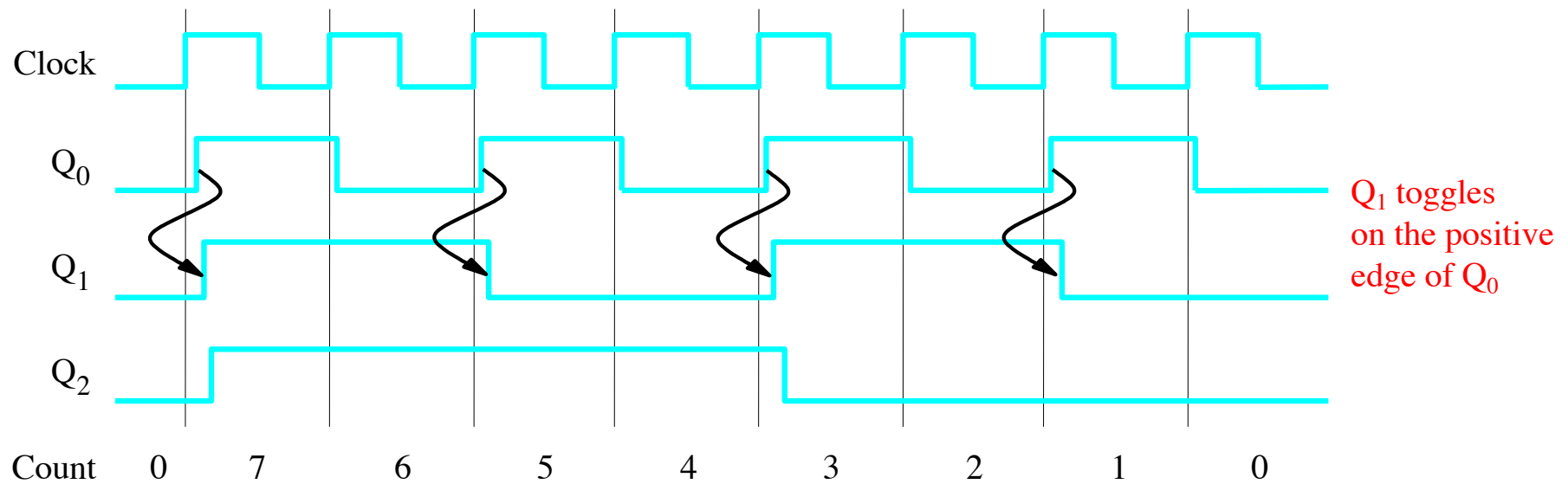


(b) Timing diagram

# A three-bit down-counter

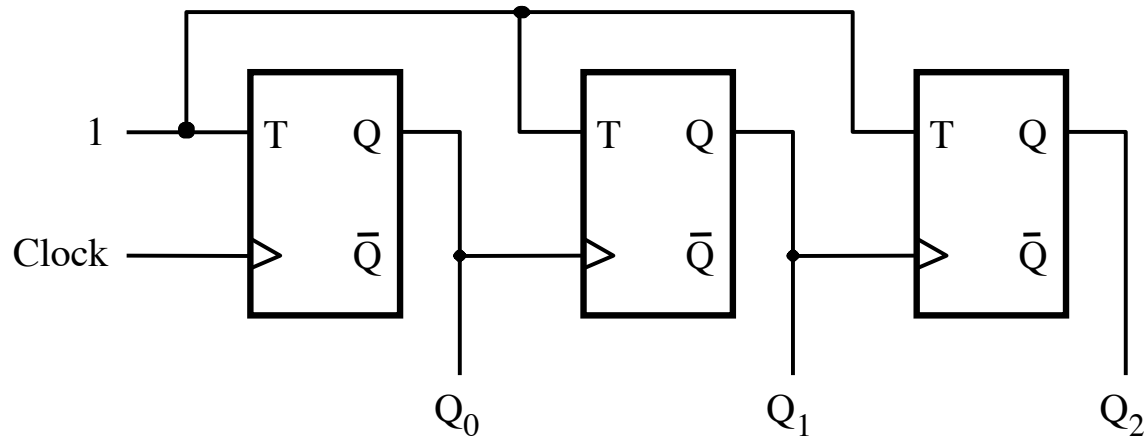


(a) Circuit

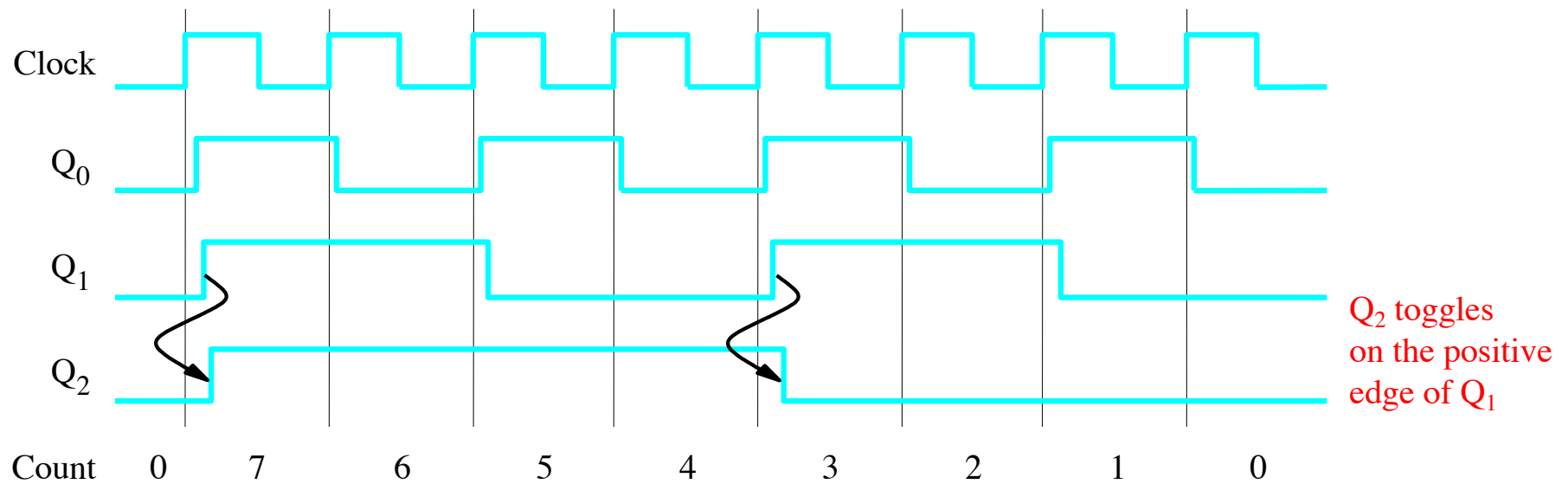


(b) Timing diagram

# A three-bit down-counter

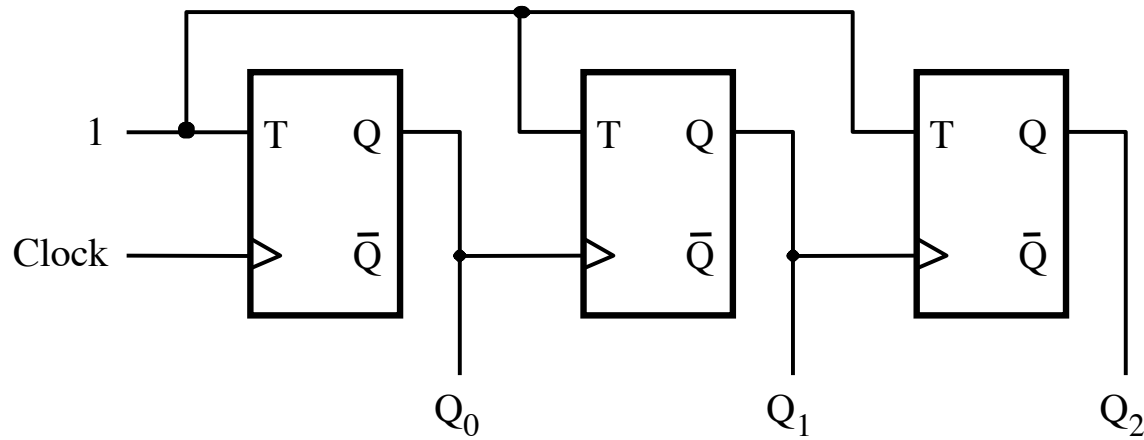


(a) Circuit



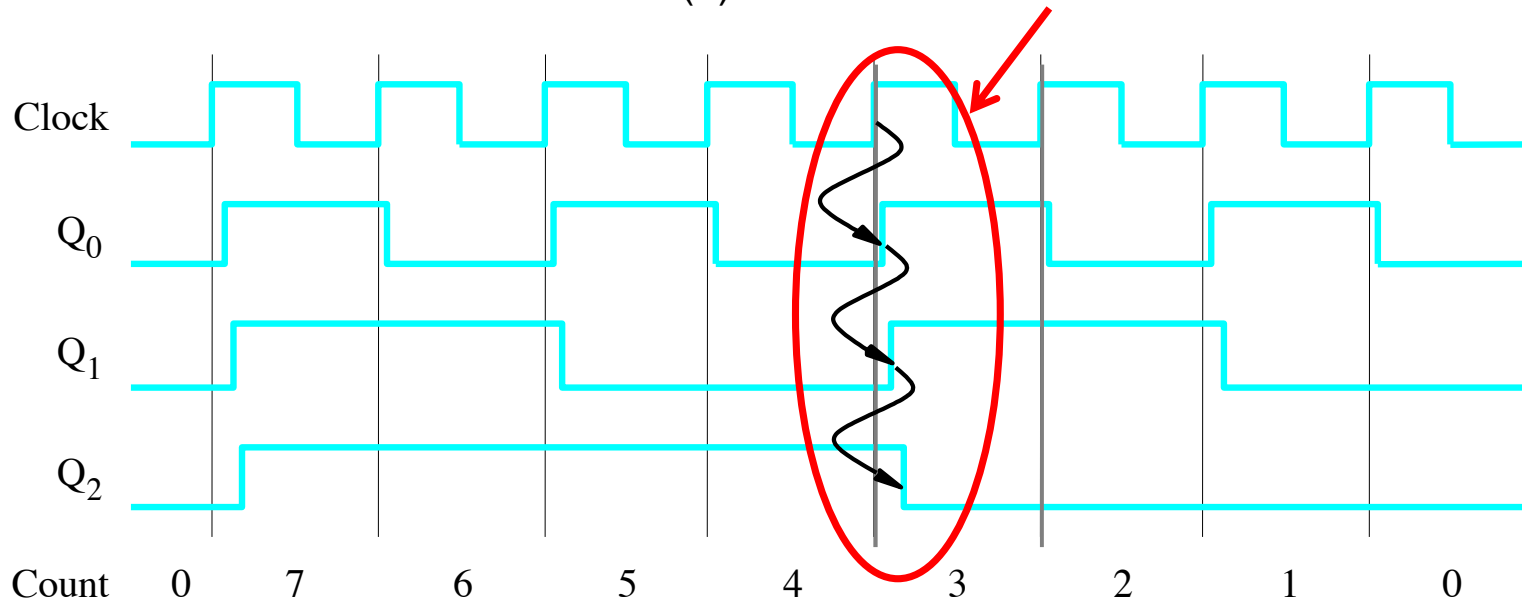
(b) Timing diagram

# A three-bit down-counter



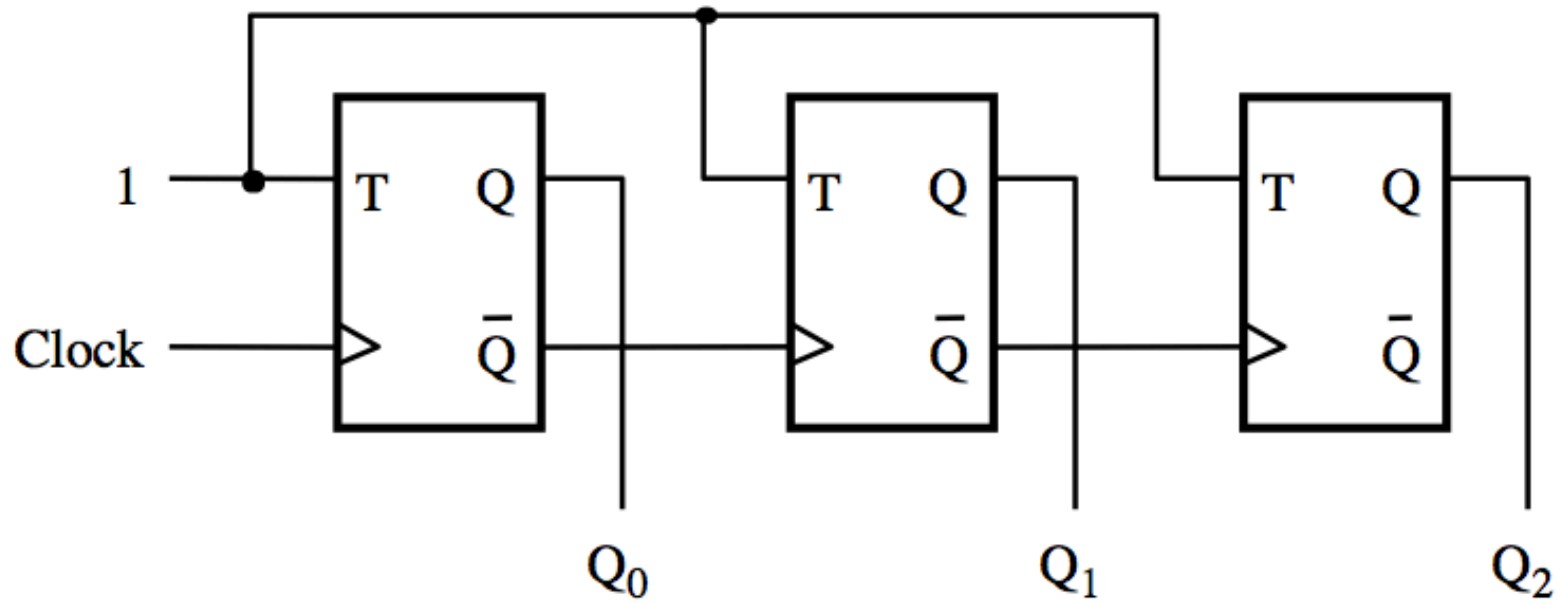
(a) Circuit

The propagation delays get longer

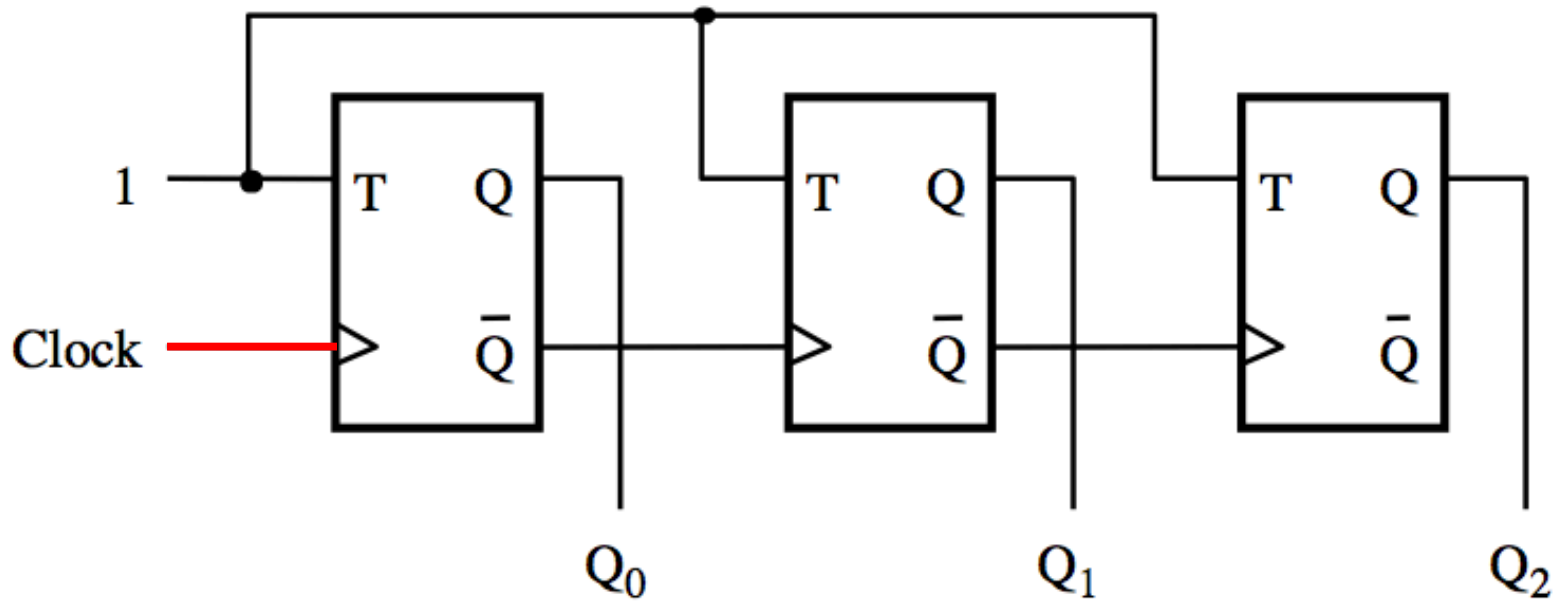


(b) Timing diagram

# A three-bit up-counter

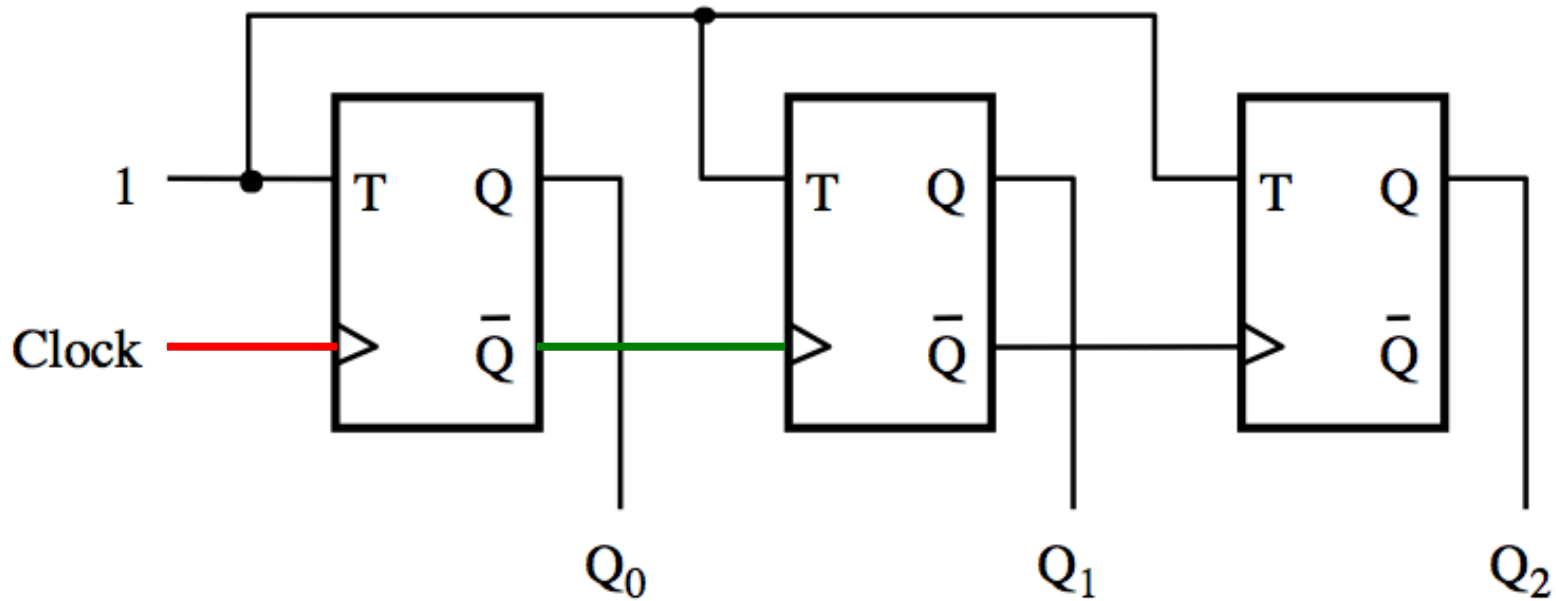


# A three-bit up-counter



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

# A three-bit up-counter

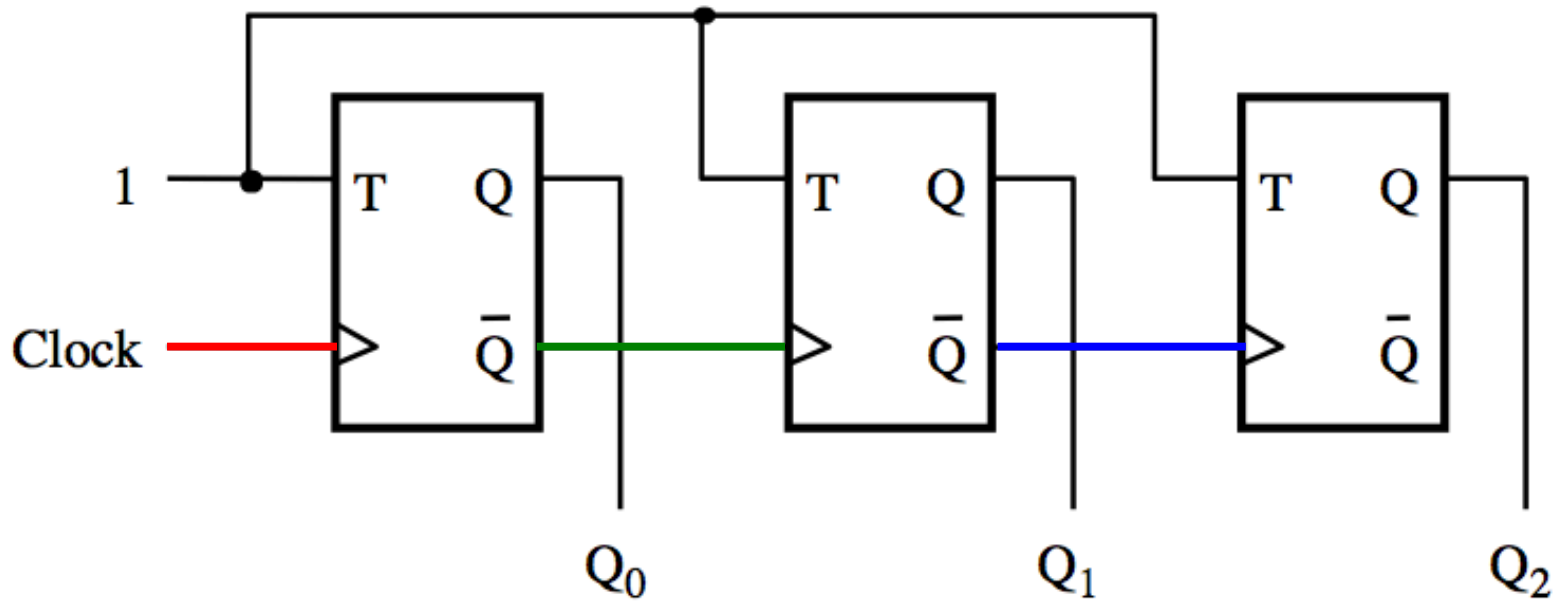


The first flip-flop changes  
on the positive edge of the clock

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



# A three-bit up-counter

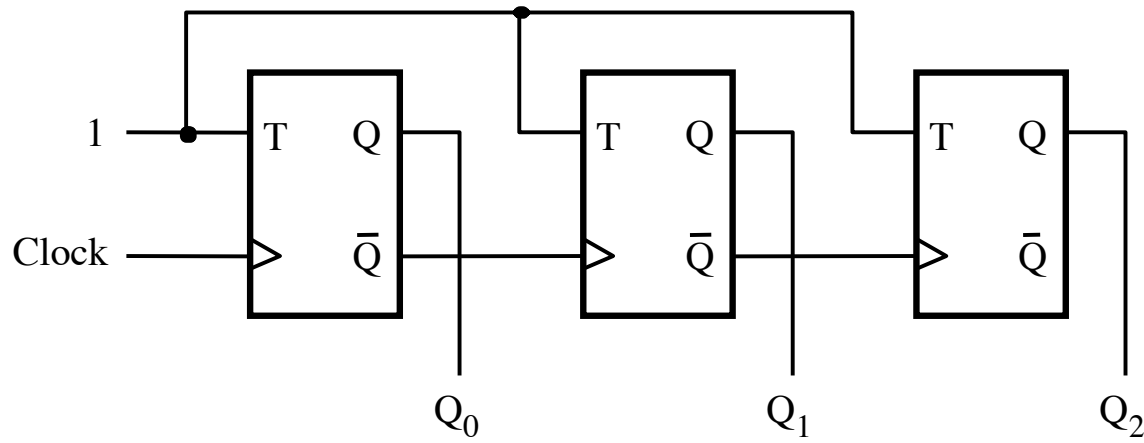


The first flip-flop changes  
on the positive edge of the clock

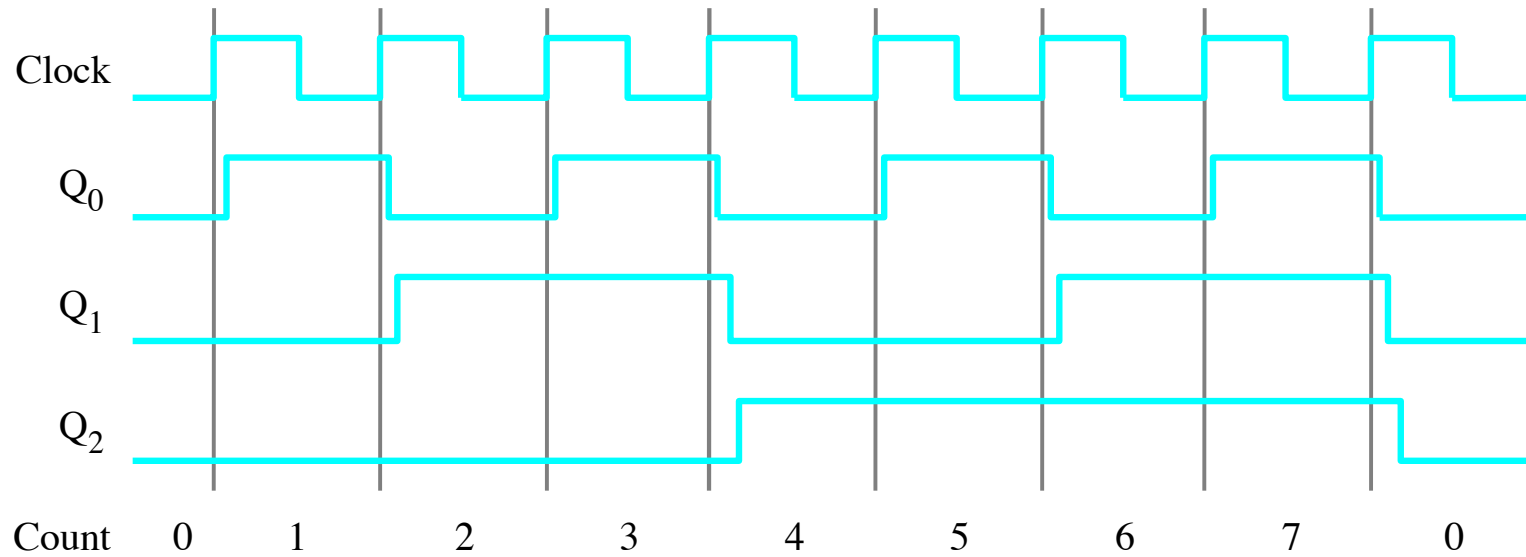
The second flip-flop changes  
on the positive edge of  $\bar{Q}_0$

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

# A three-bit up-counter

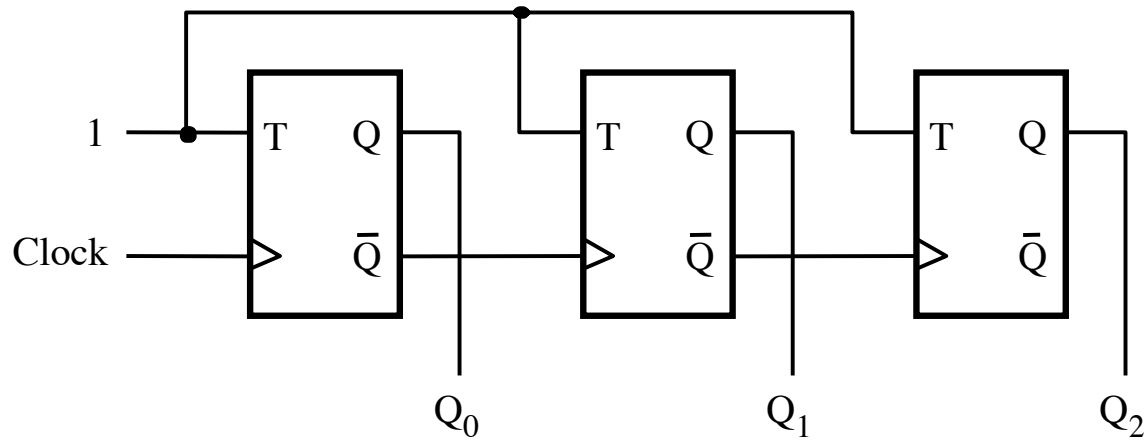


(a) Circuit

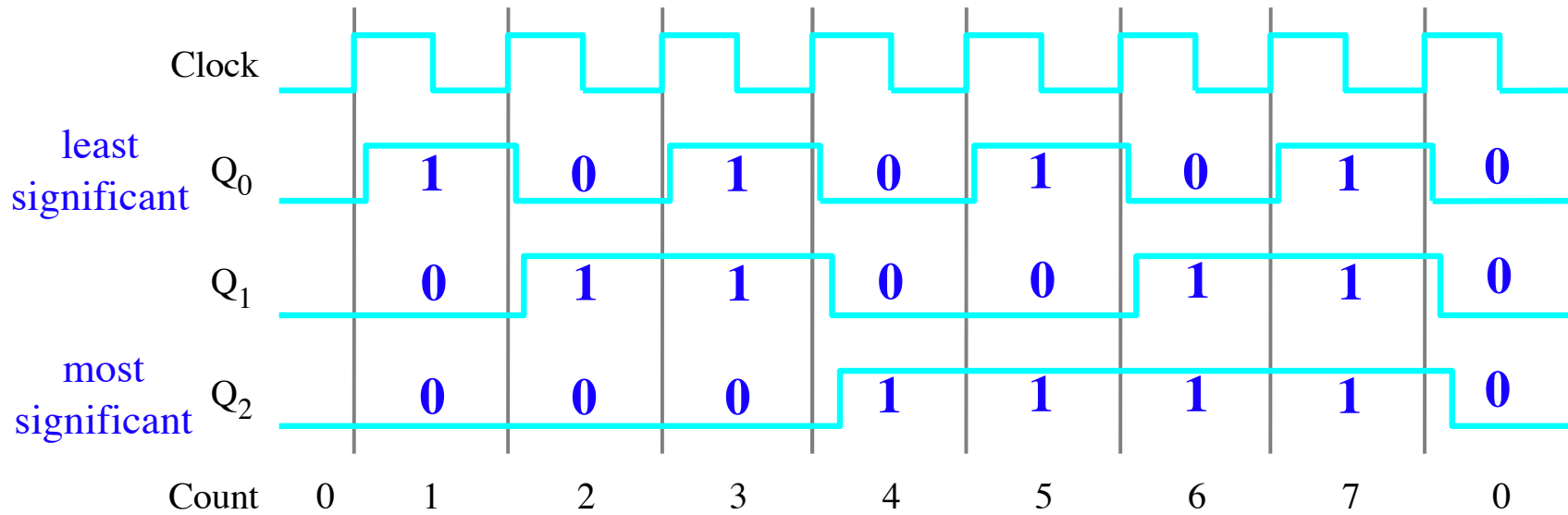


(b) Timing diagram

# A three-bit up-counter

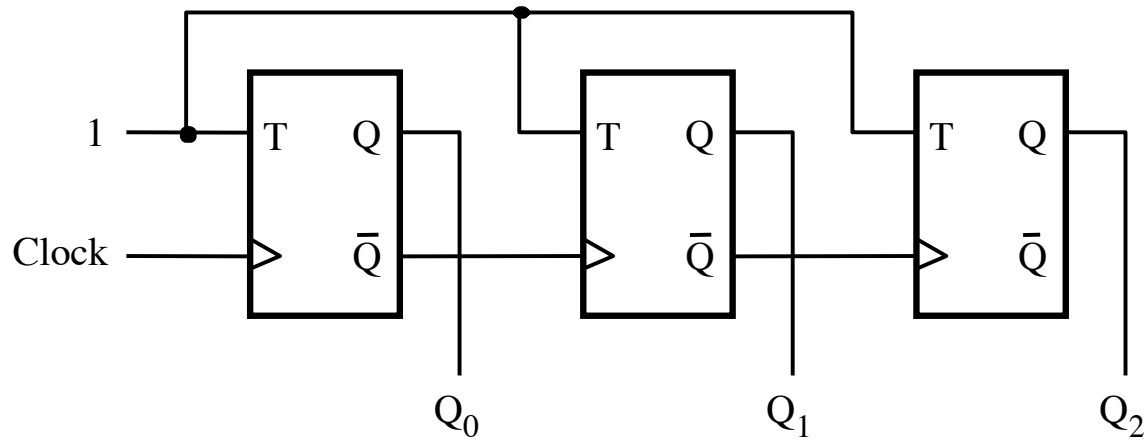


(a) Circuit

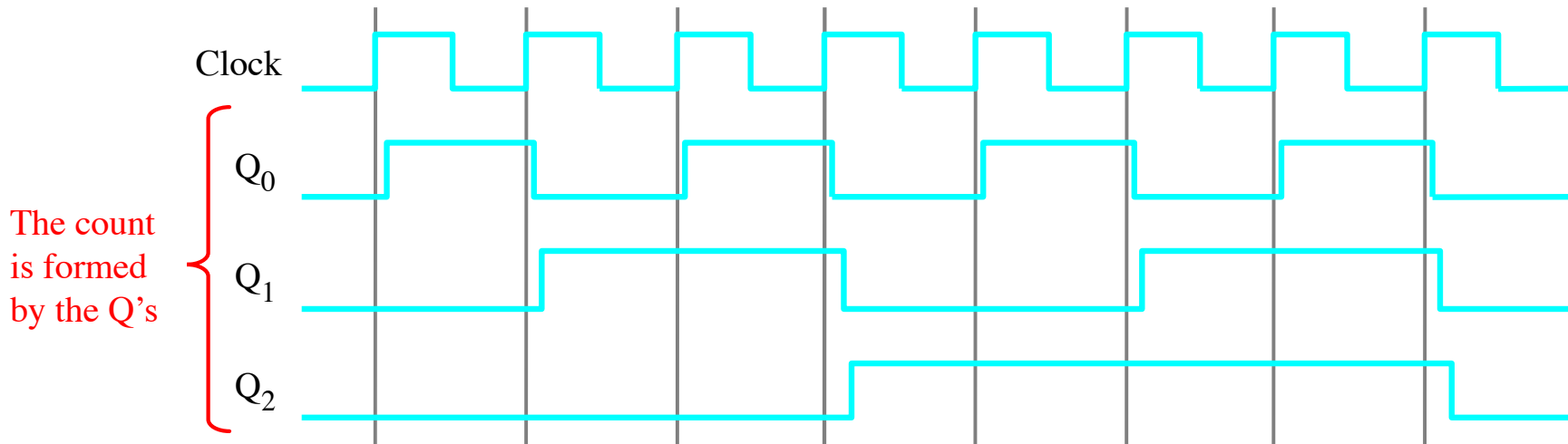


(b) Timing diagram

# A three-bit up-counter

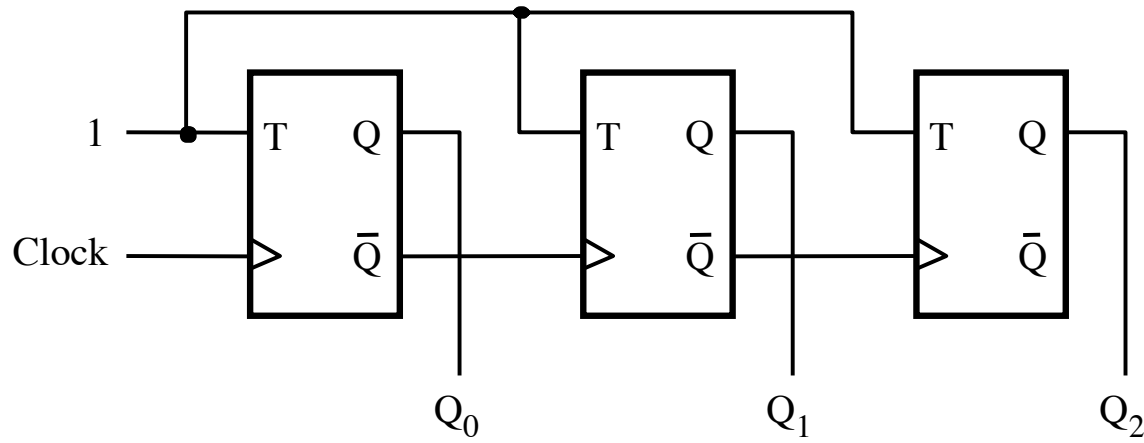


(a) Circuit

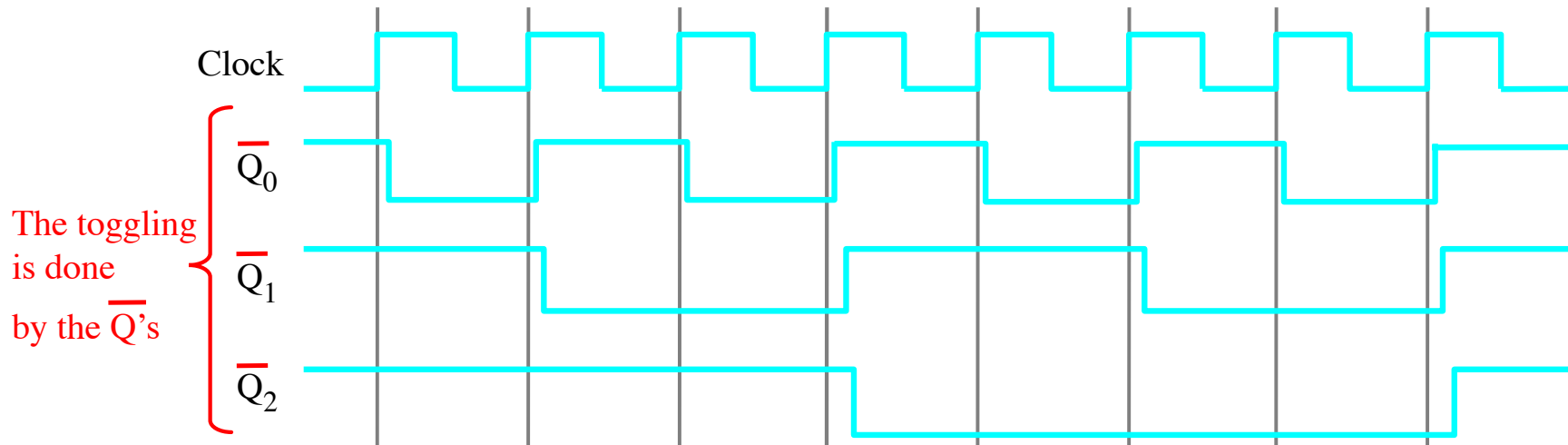


(b) Timing diagram

# A three-bit up-counter

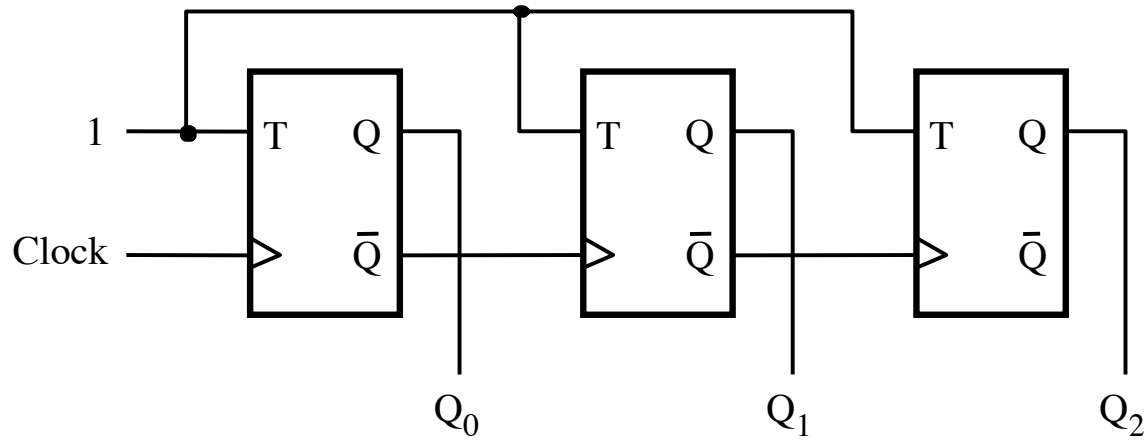


(a) Circuit

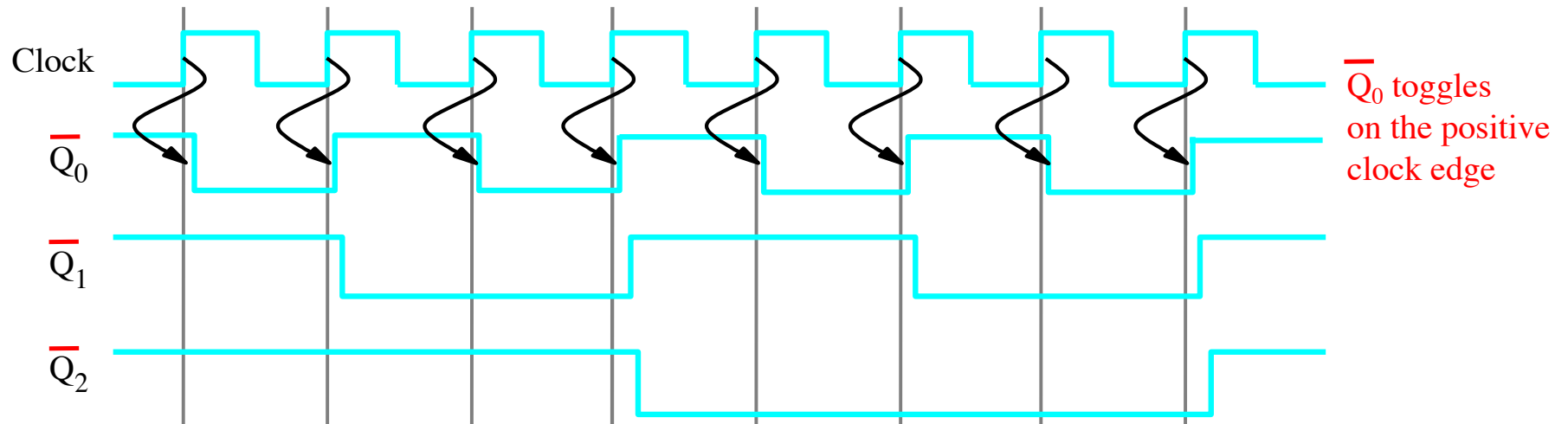


(b) Timing diagram

# A three-bit up-counter

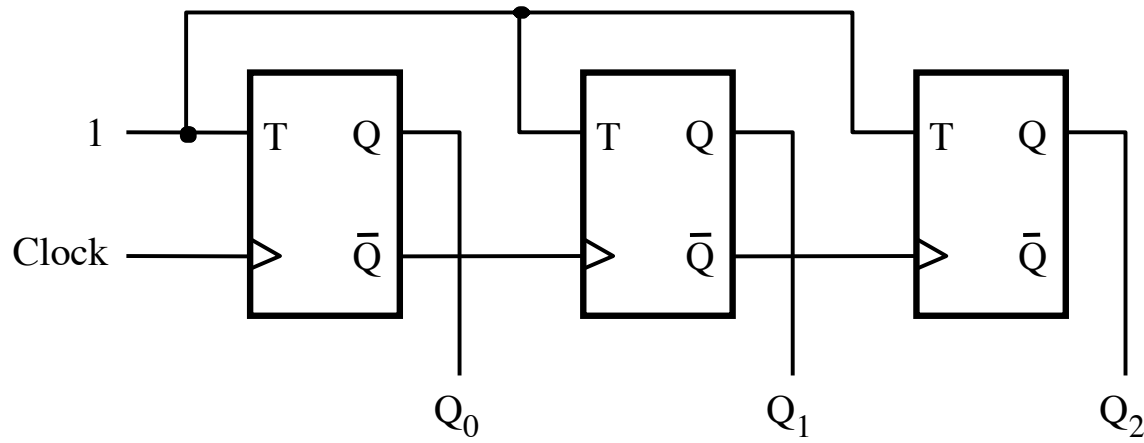


(a) Circuit

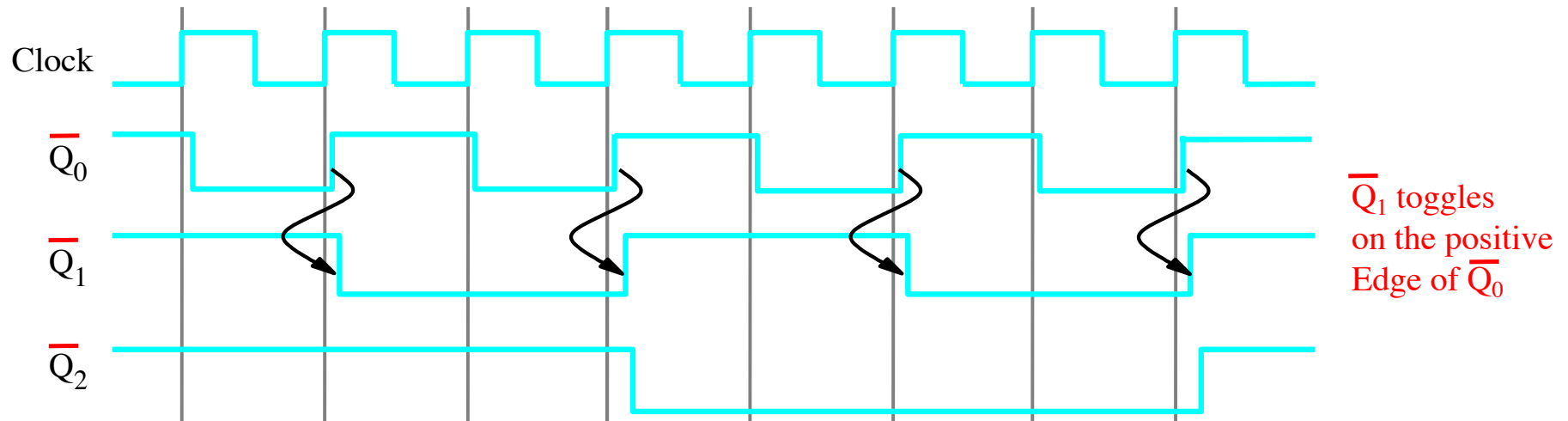


(b) Timing diagram

# A three-bit up-counter

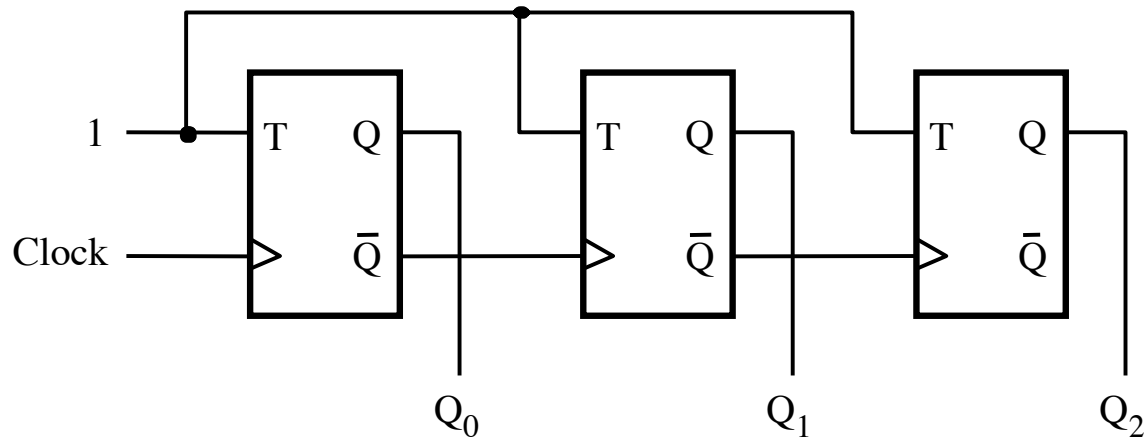


(a) Circuit

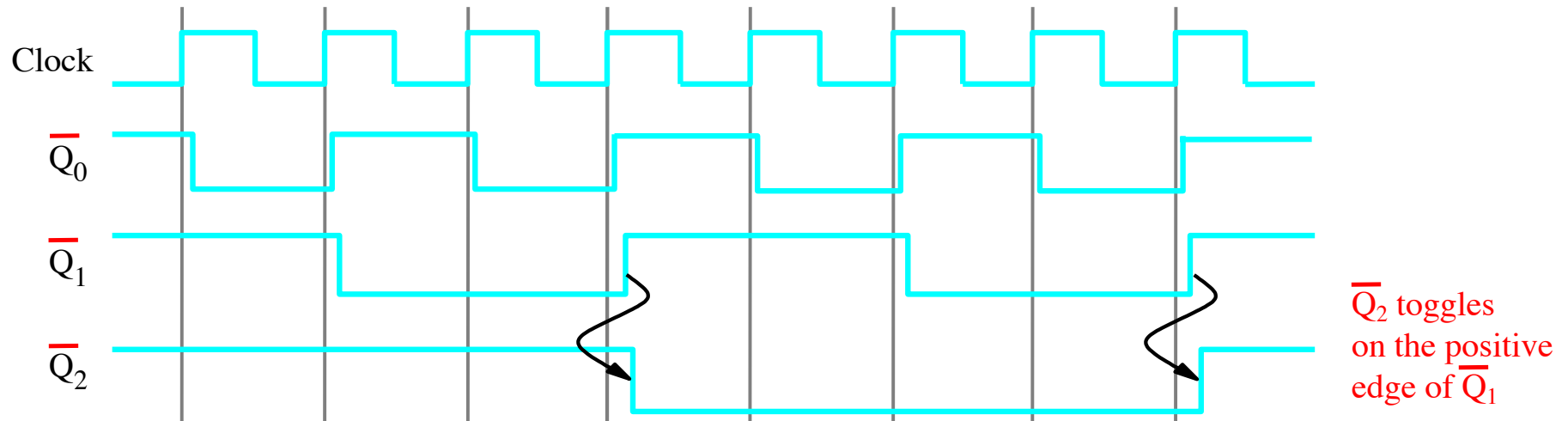


(b) Timing diagram

# A three-bit up-counter



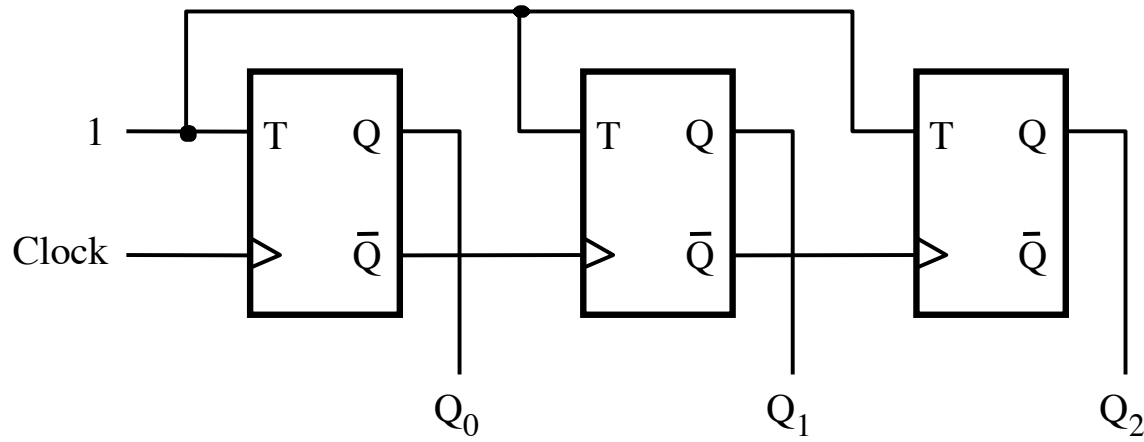
(a) Circuit



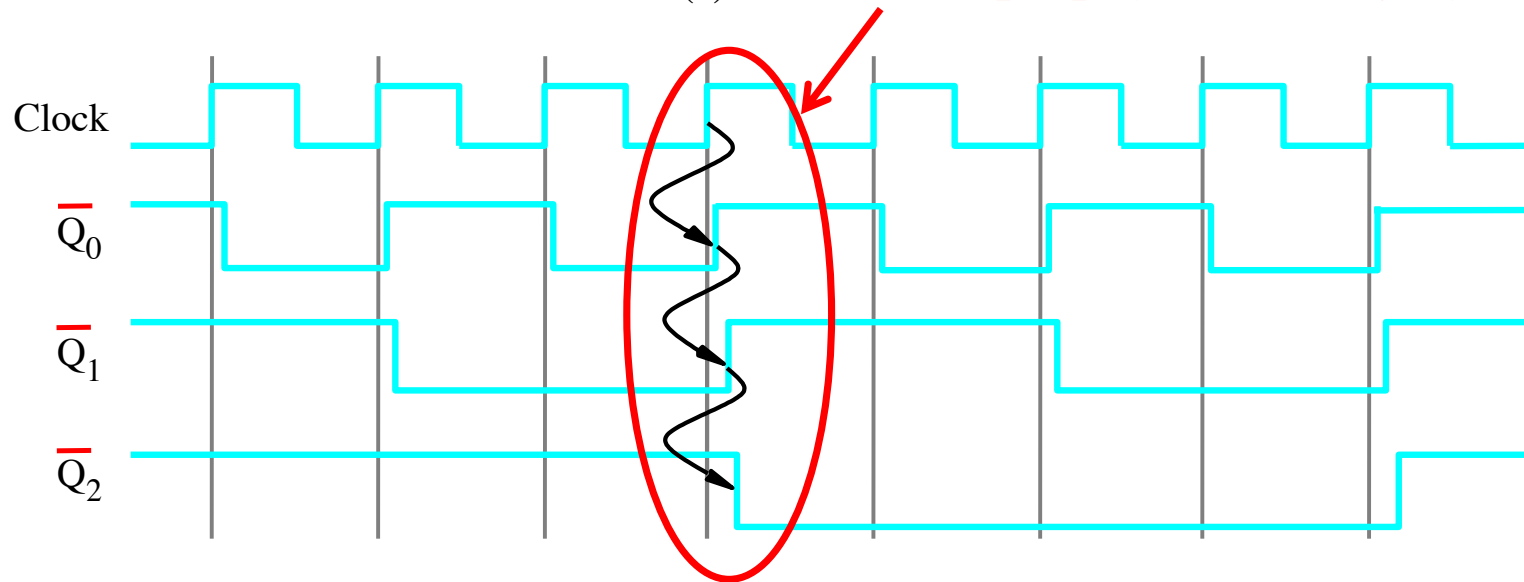
(b) Timing diagram



# A three-bit up-counter

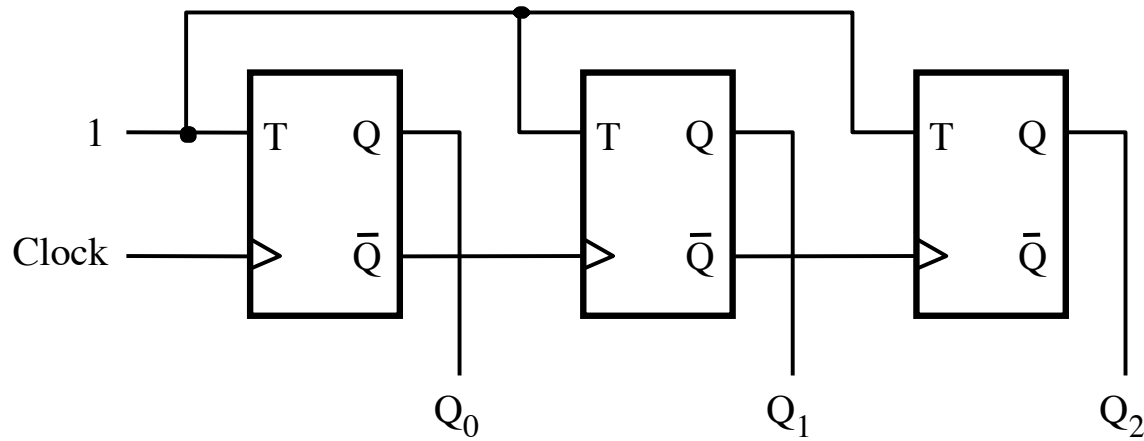


(a) Circuit **The propagation delays get longer**

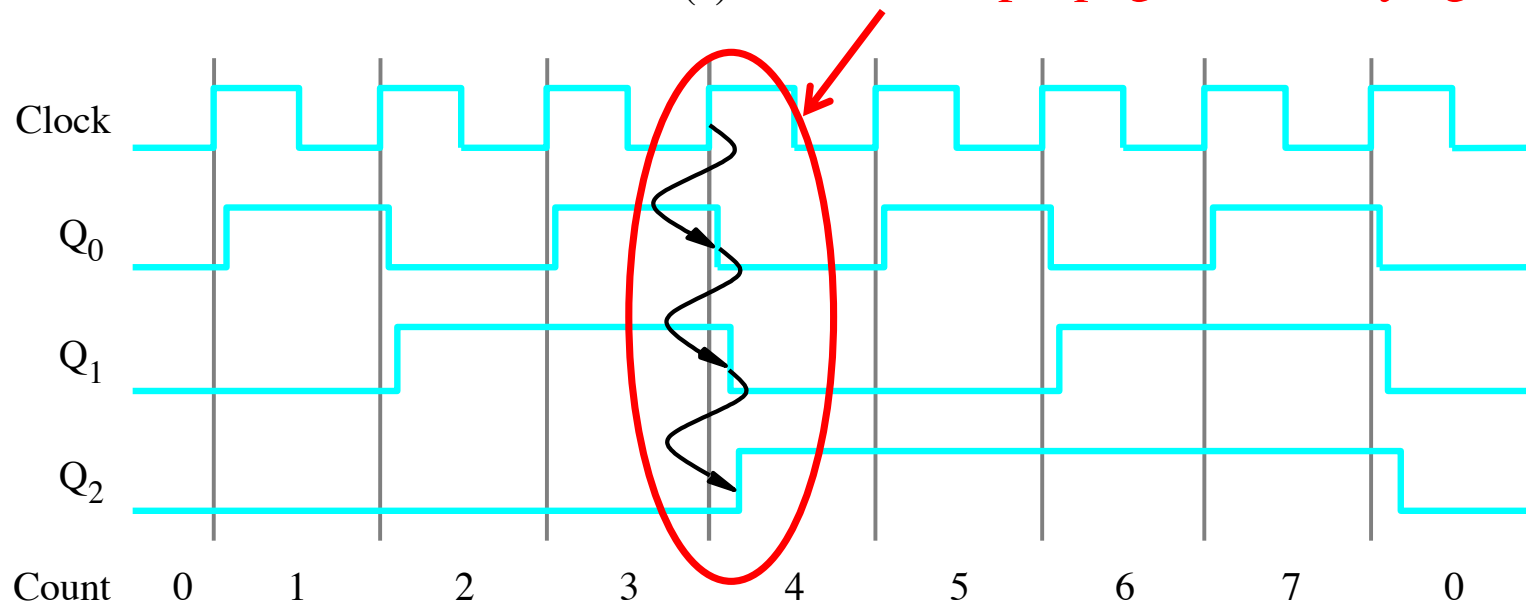


(b) Timing diagram

# A three-bit up-counter



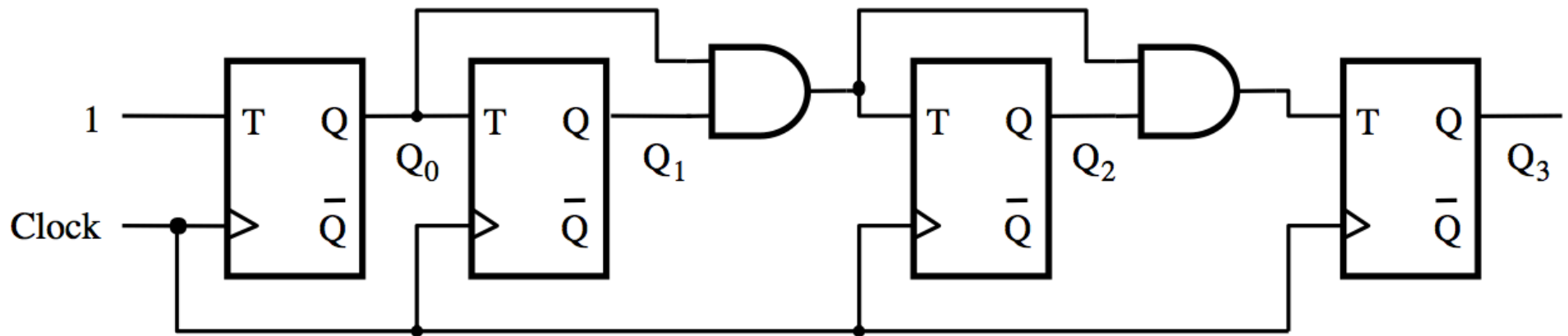
(a) Circuit **The propagation delays get longer**



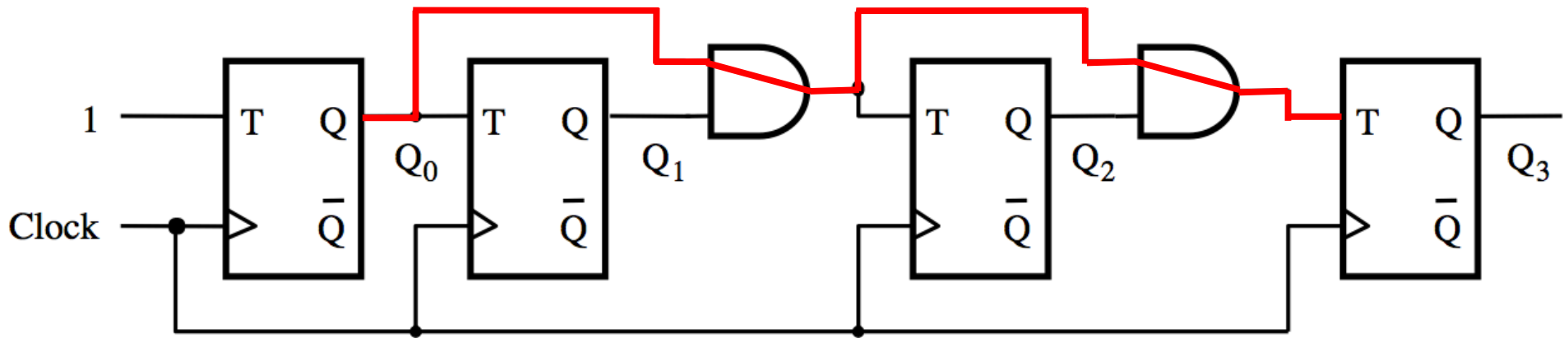
(b) Timing diagram

# **Synchronous Counters**

# A four-bit synchronous up-counter

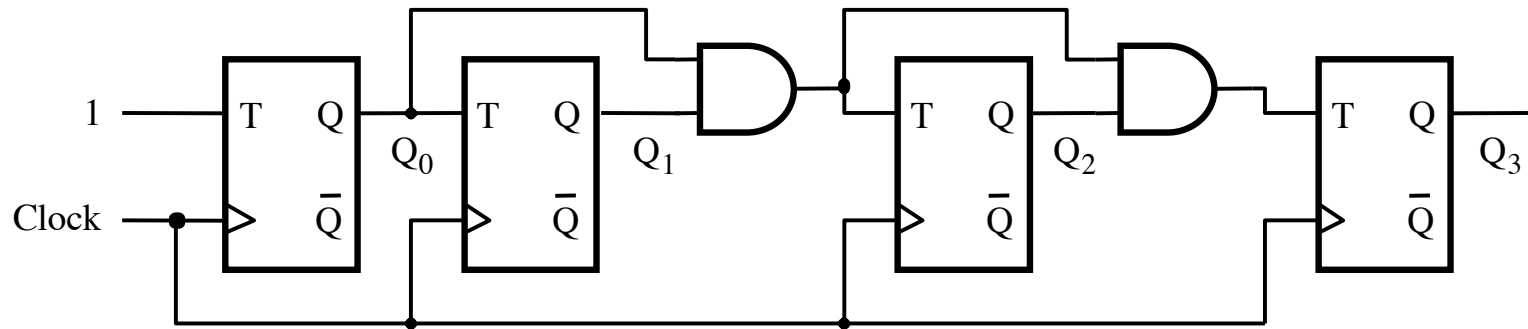


# A four-bit synchronous up-counter

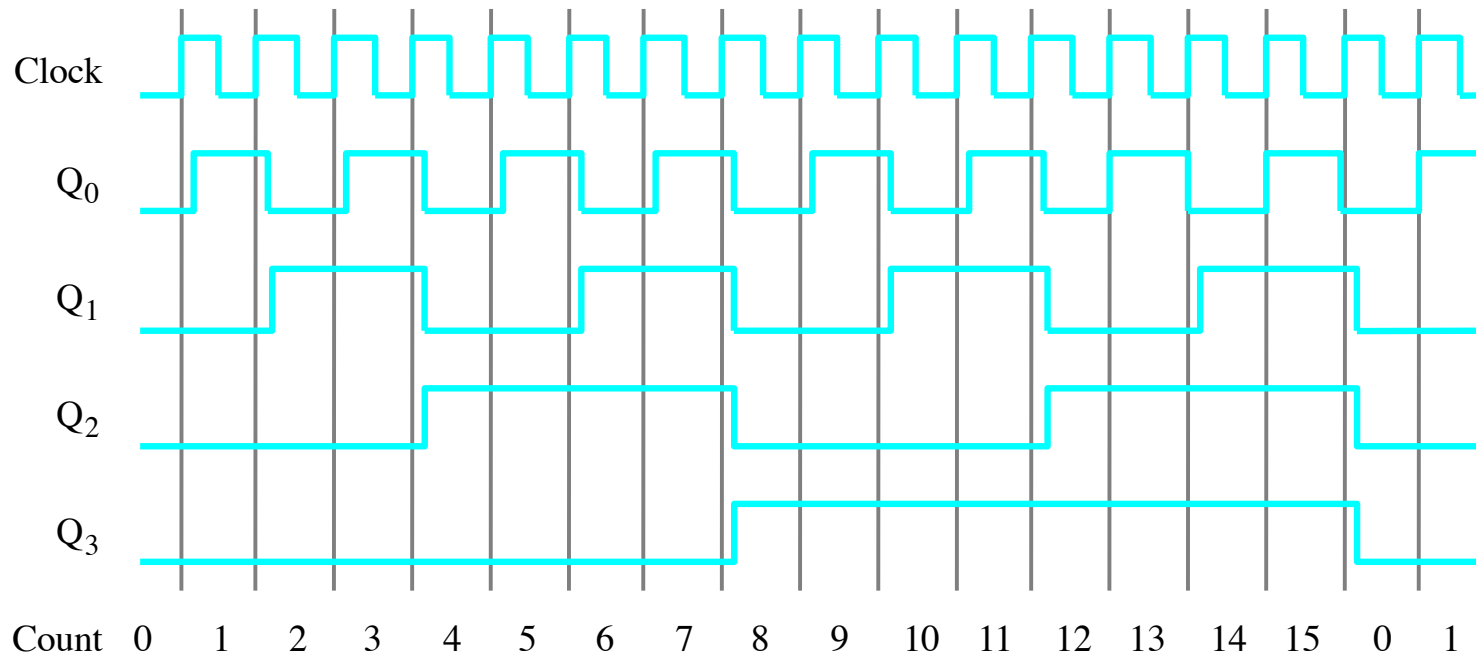


The propagation delay through all AND gates combined must not exceed the clock period minus the setup time for the flip-flops

# A four-bit synchronous up-counter



(a) Circuit



(b) Timing diagram

# Derivation of the synchronous up-counter

Clock cycle	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
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

Q<sub>1</sub> changes

Q<sub>2</sub> changes

# Derivation of the synchronous up-counter

Clock cycle	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
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

Q<sub>1</sub> changes

Q<sub>2</sub> changes

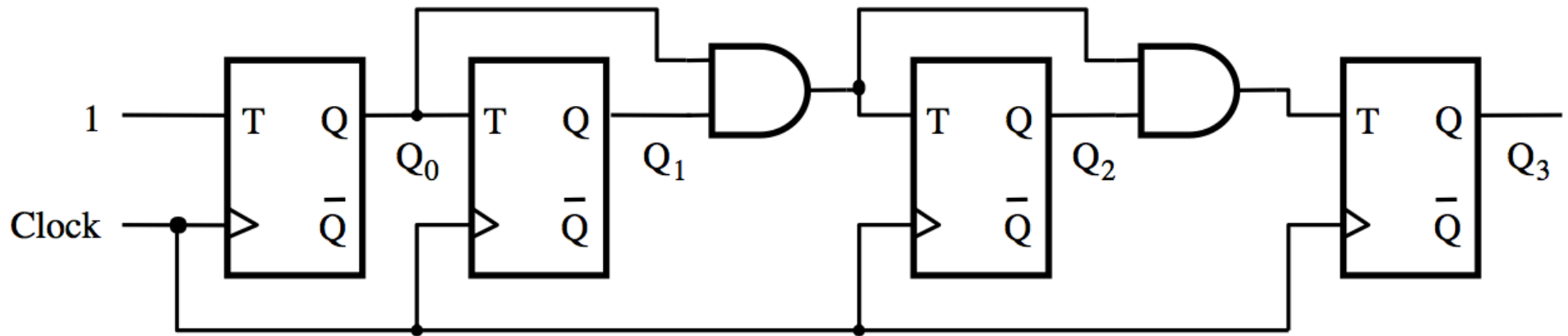
$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0 Q_1$$



# A four-bit synchronous up-counter



$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0 Q_1$$

# In general we have

$$T_0 = 1$$

$$T_1 = Q_0$$

$$T_2 = Q_0 Q_1$$

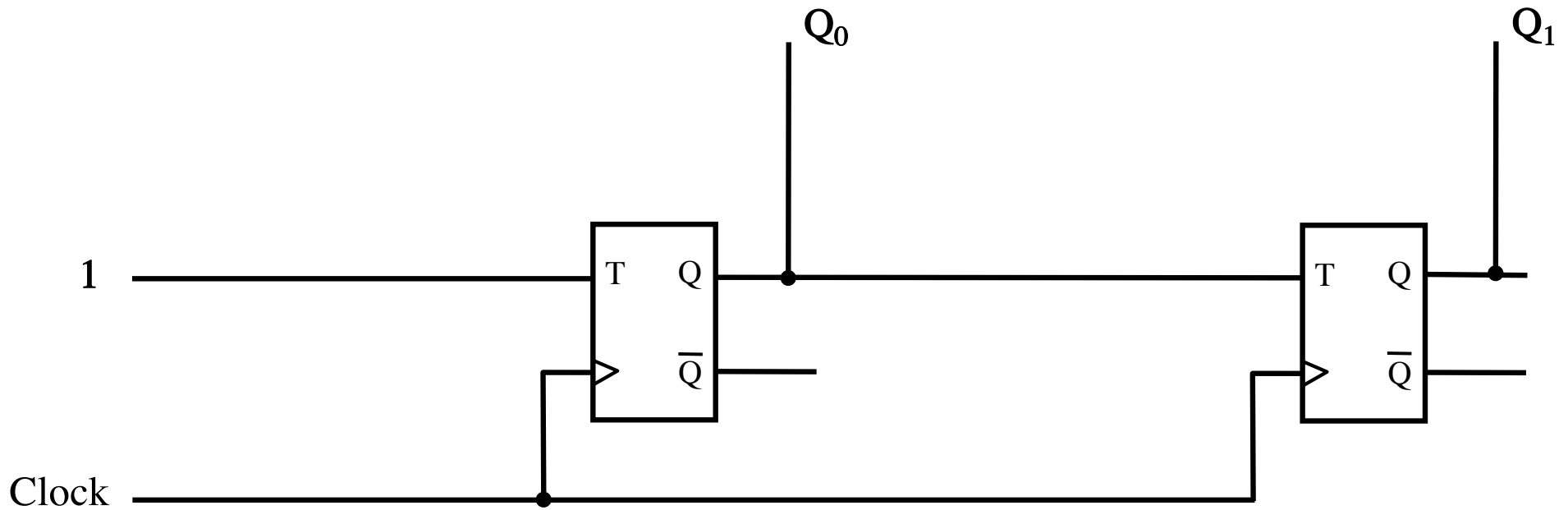
$$T_3 = Q_0 Q_1 Q_2$$

...

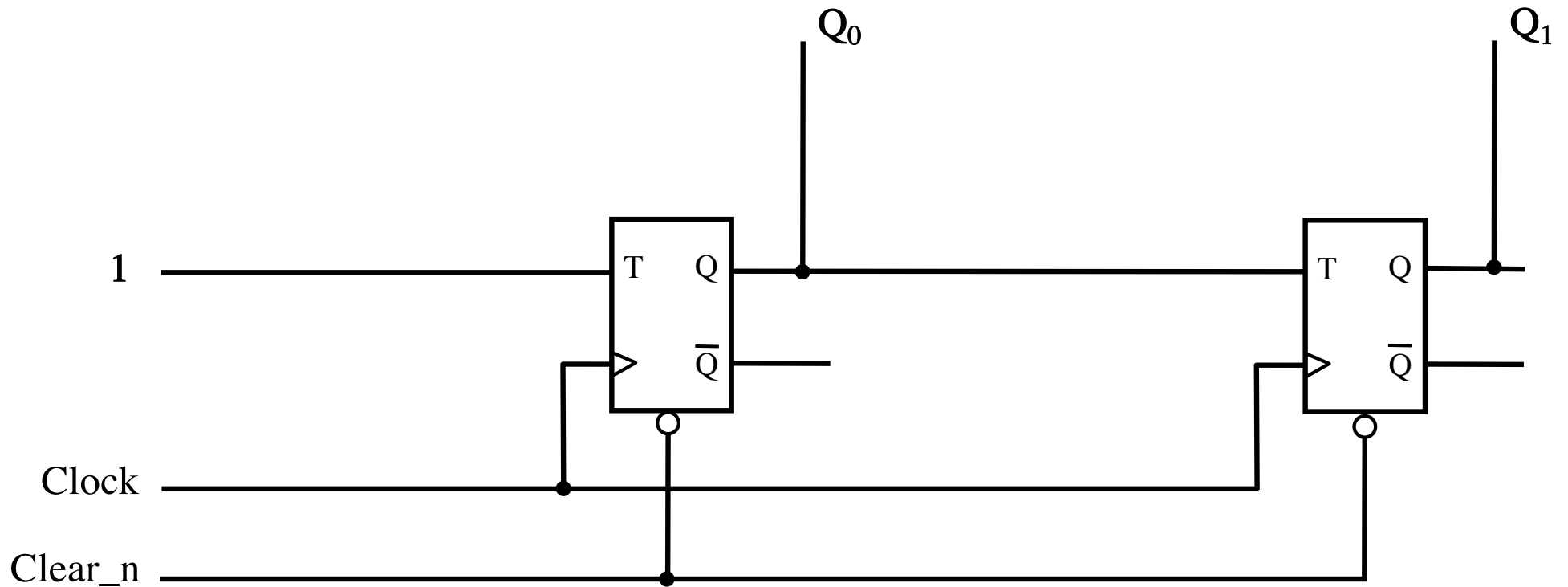
$$T_n = Q_0 Q_1 Q_2 \cdots Q_{n-1}$$

# **Synchronous v.s. Asynchronous Clear**

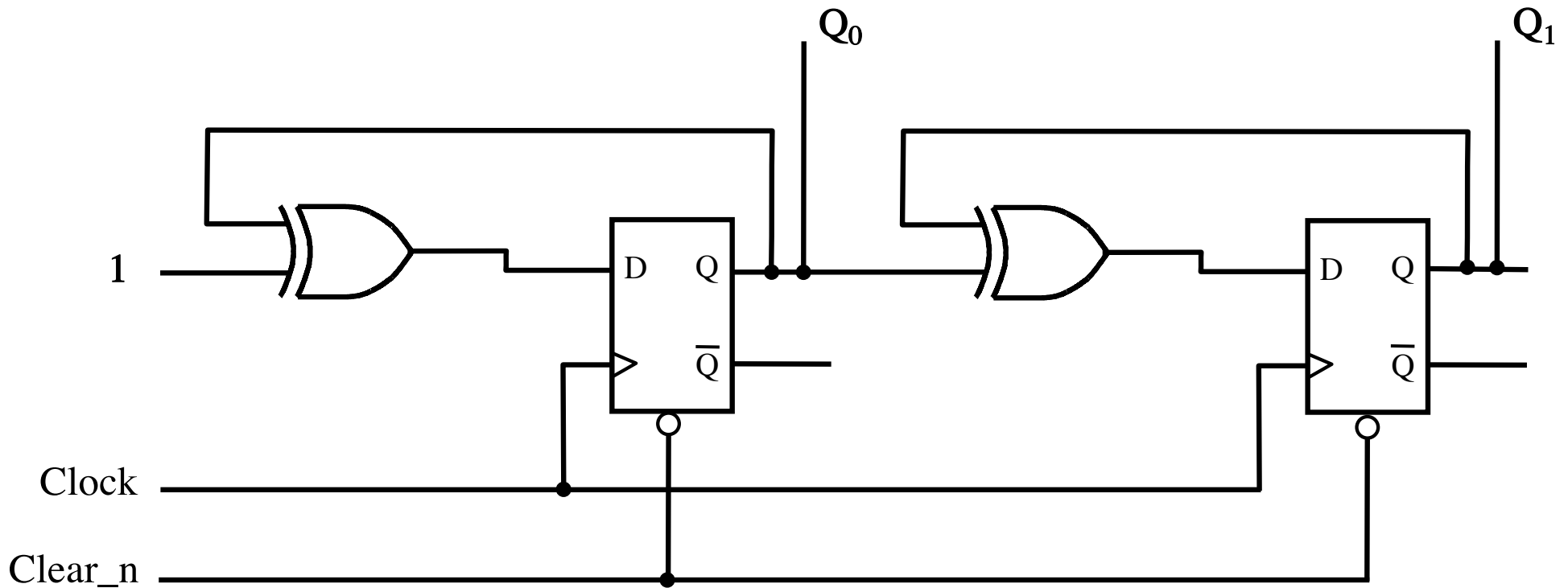
# 2-Bit Synchronous Up-Counter (without clear capability)



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

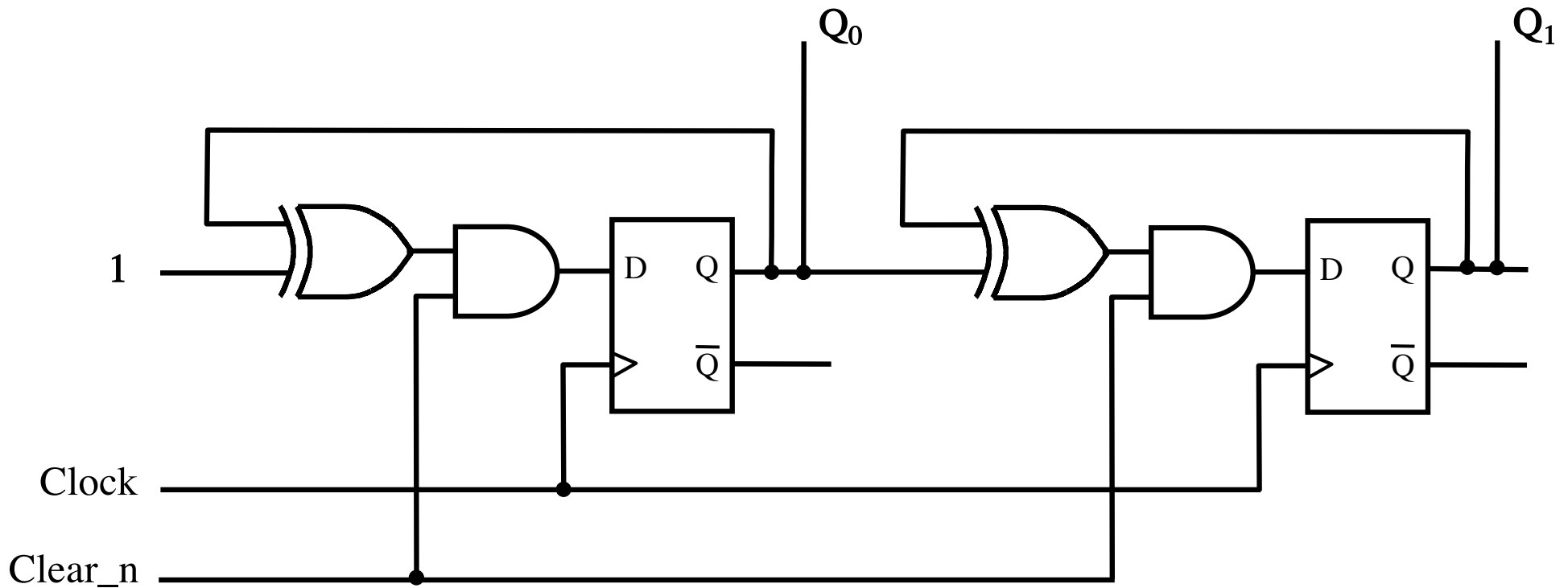


# 2-Bit Synchronous Up-Counter (with asynchronous clear)



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

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

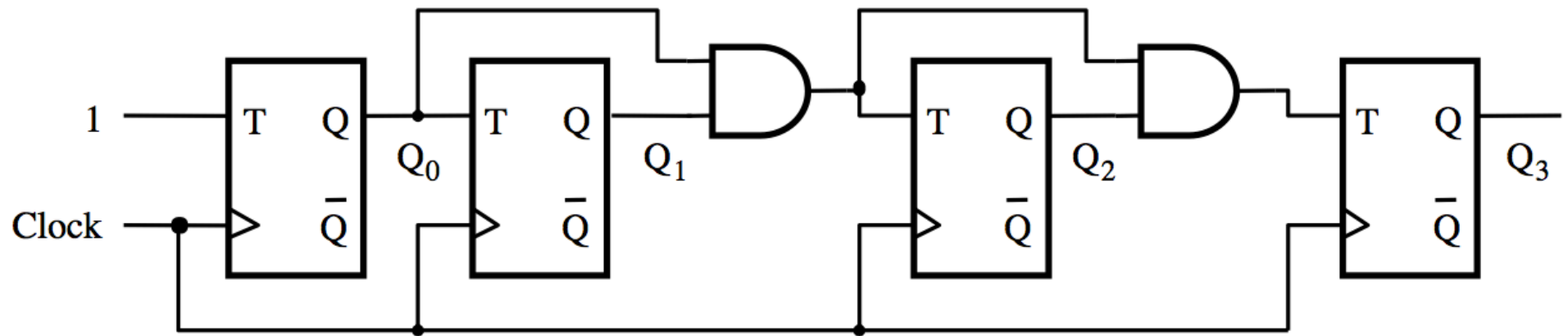


This counter can be cleared only on the positive clock edge.

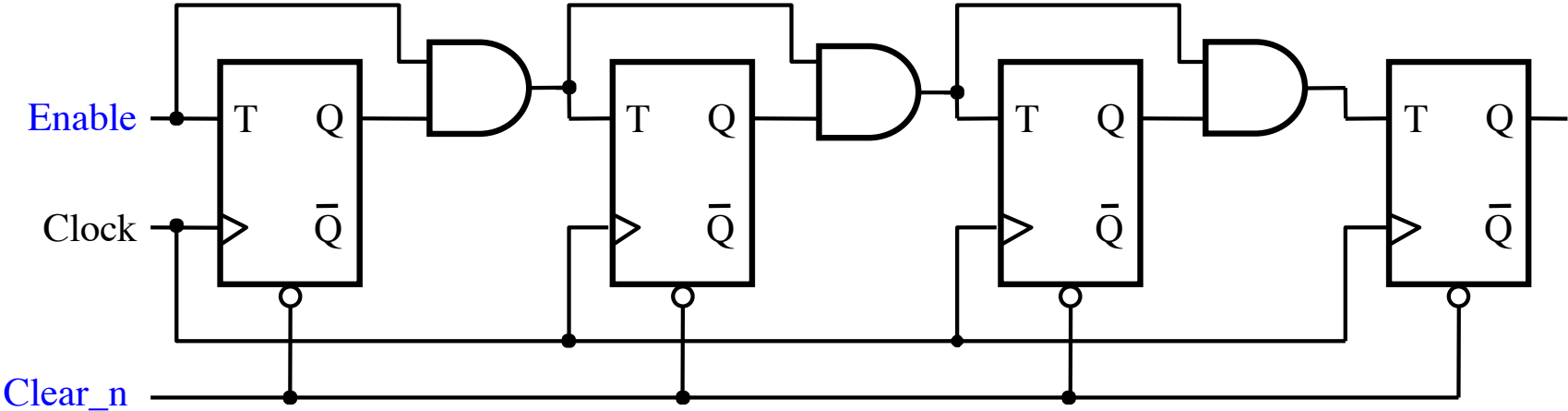
# **Adding Enable Capability**



# A four-bit synchronous up-counter



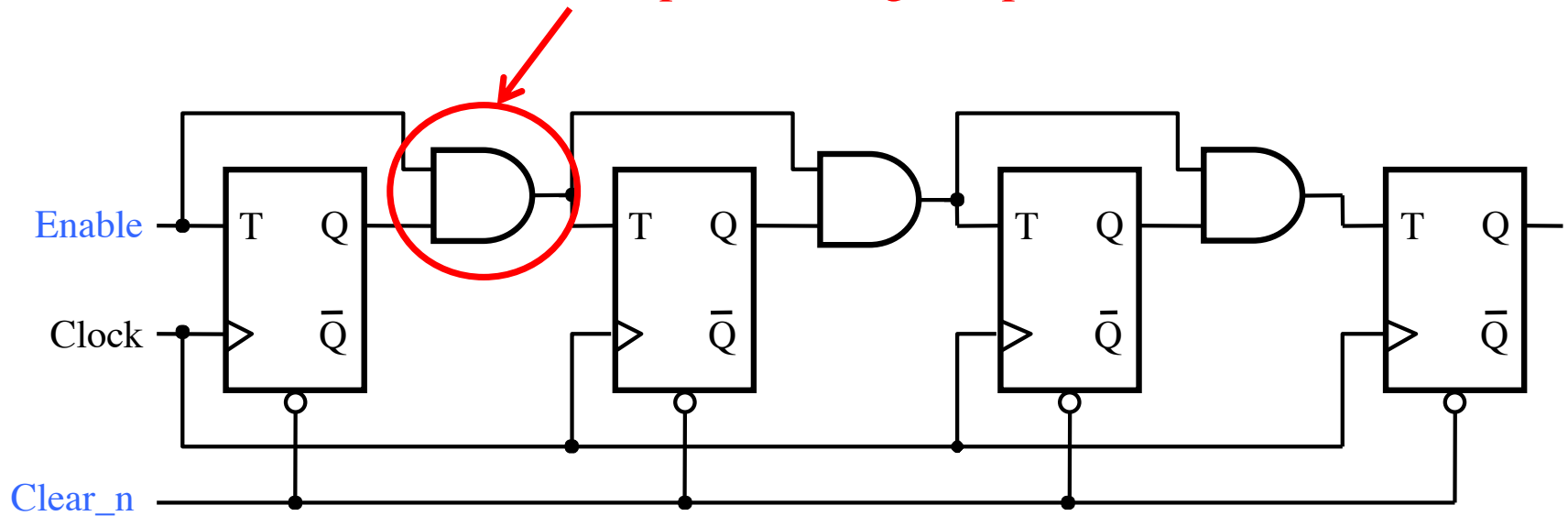
# Inclusion of Enable and Clear Capability



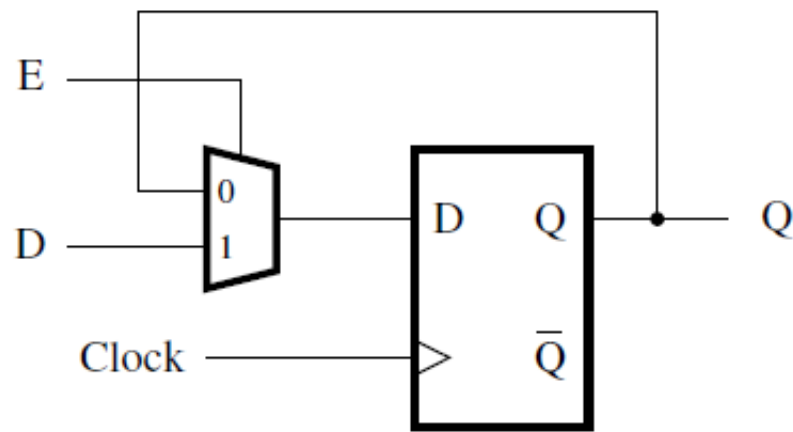
[ Figure 5.22 from the textbook ]

# Inclusion of Enable and Clear Capability

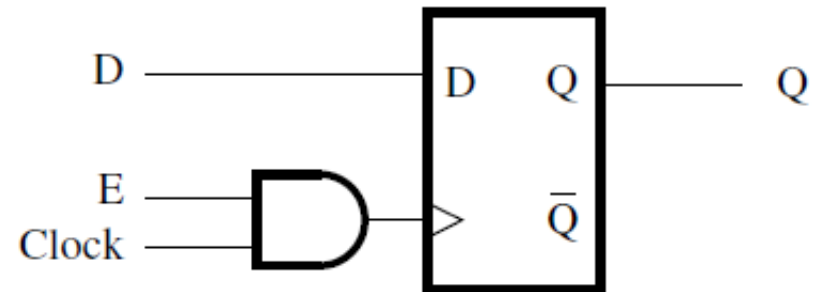
This is the new thing relative to the previous figure, plus the clear\_n line



# Providing an enable input for a D flip-flop



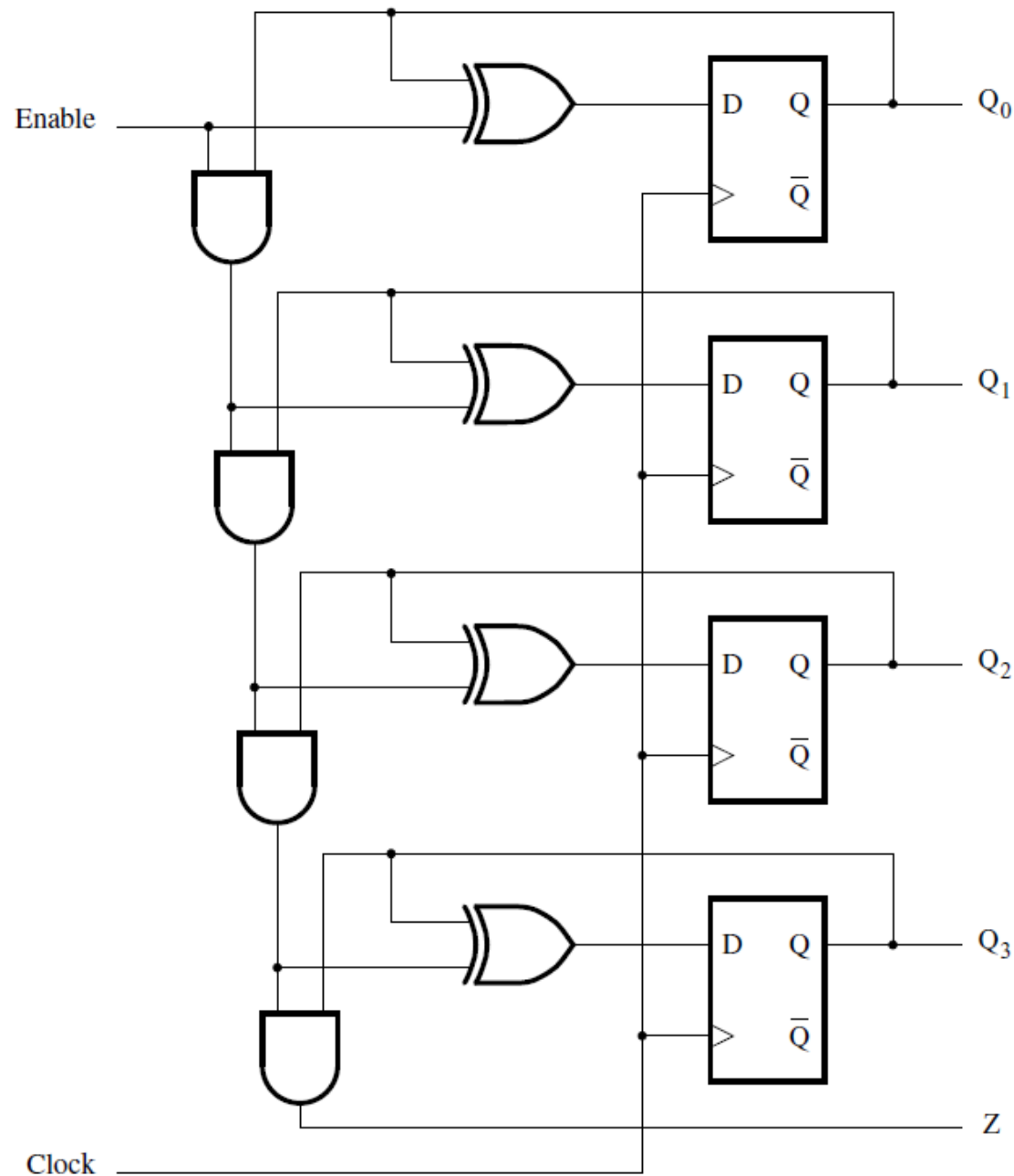
(a) Using a multiplexer



(b) Clock gating

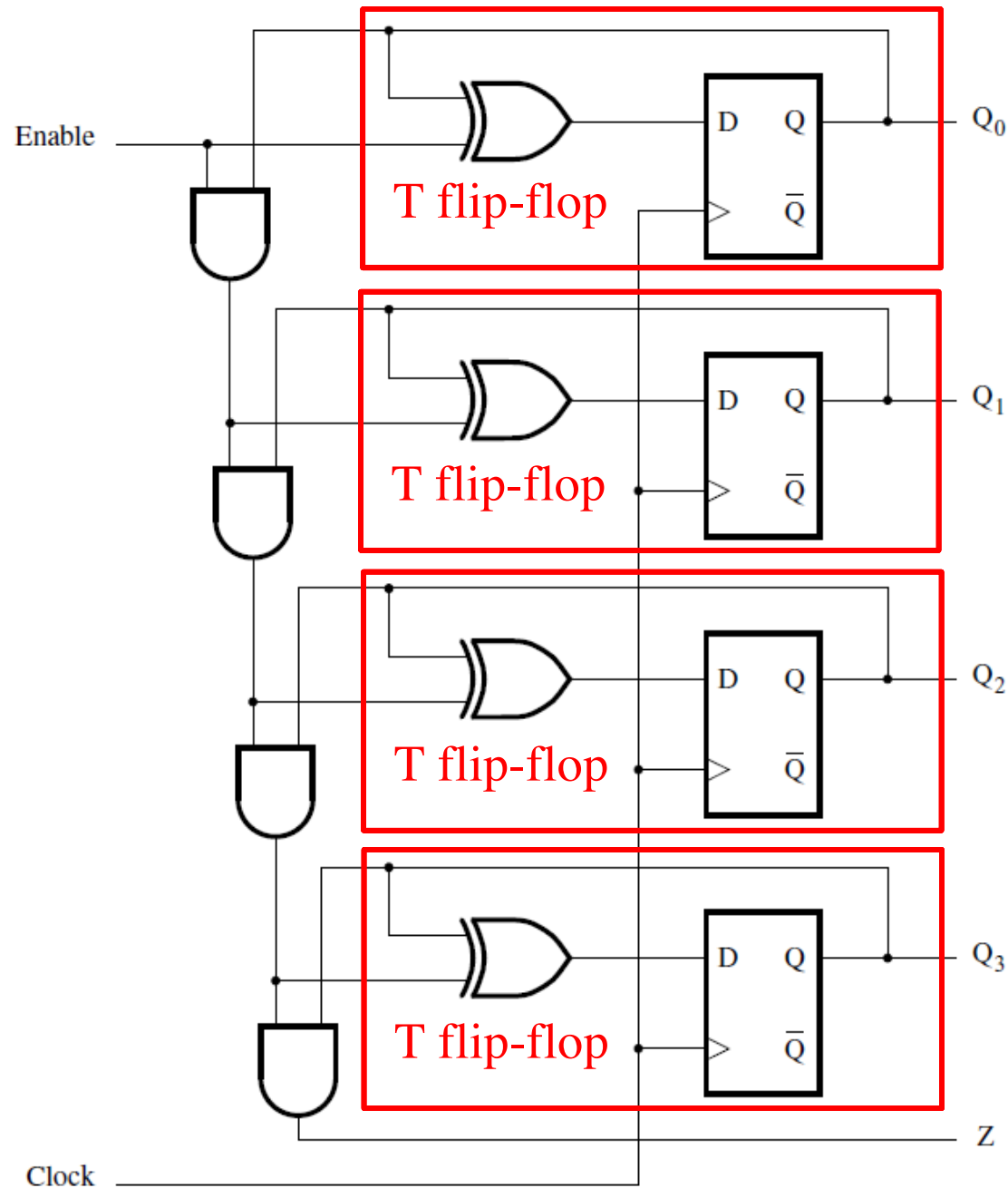
# **Synchronous Counter (with D Flip-Flops)**

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



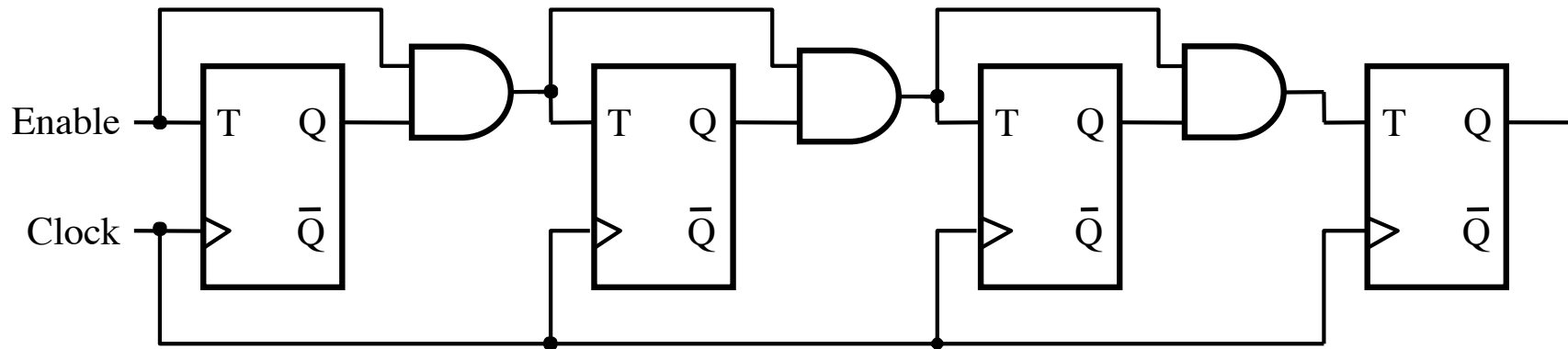
[ Figure 5.23 from the textbook ]

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



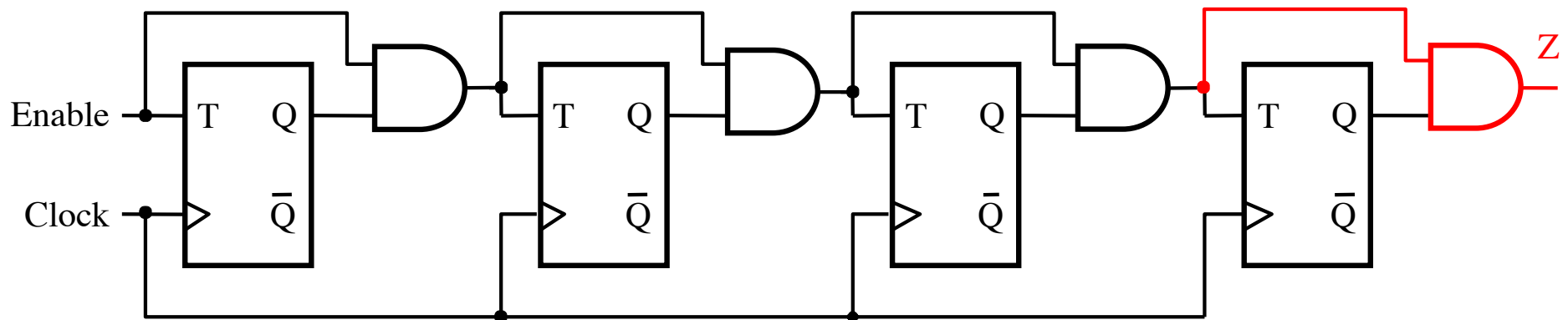
[ Figure 5.23 from the textbook ]

# Equivalent to this circuit with T flip-flops





# Equivalent to this circuit with T flip-flops



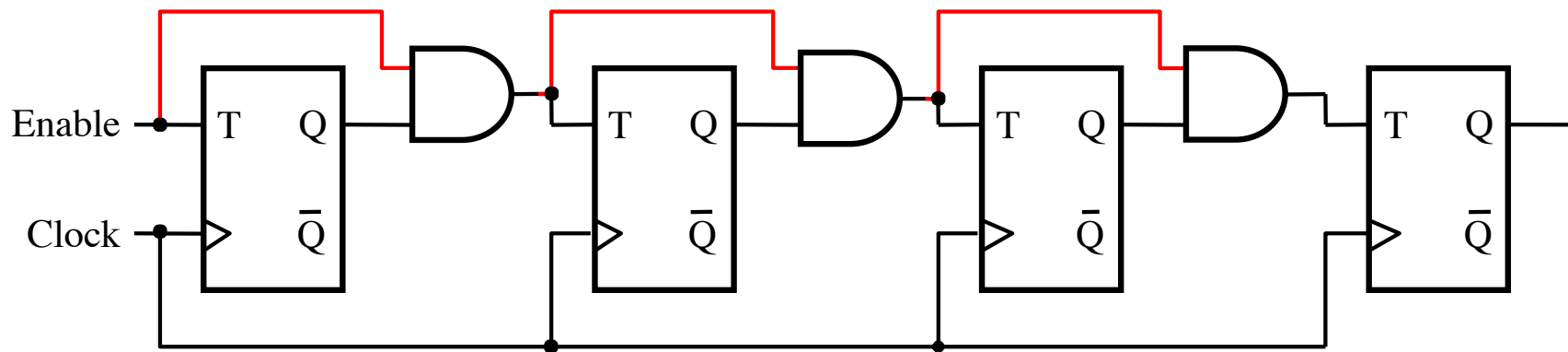
But has one extra output called Z, which can be used to connect two 4-bit counters to make an 8-bit counter.

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

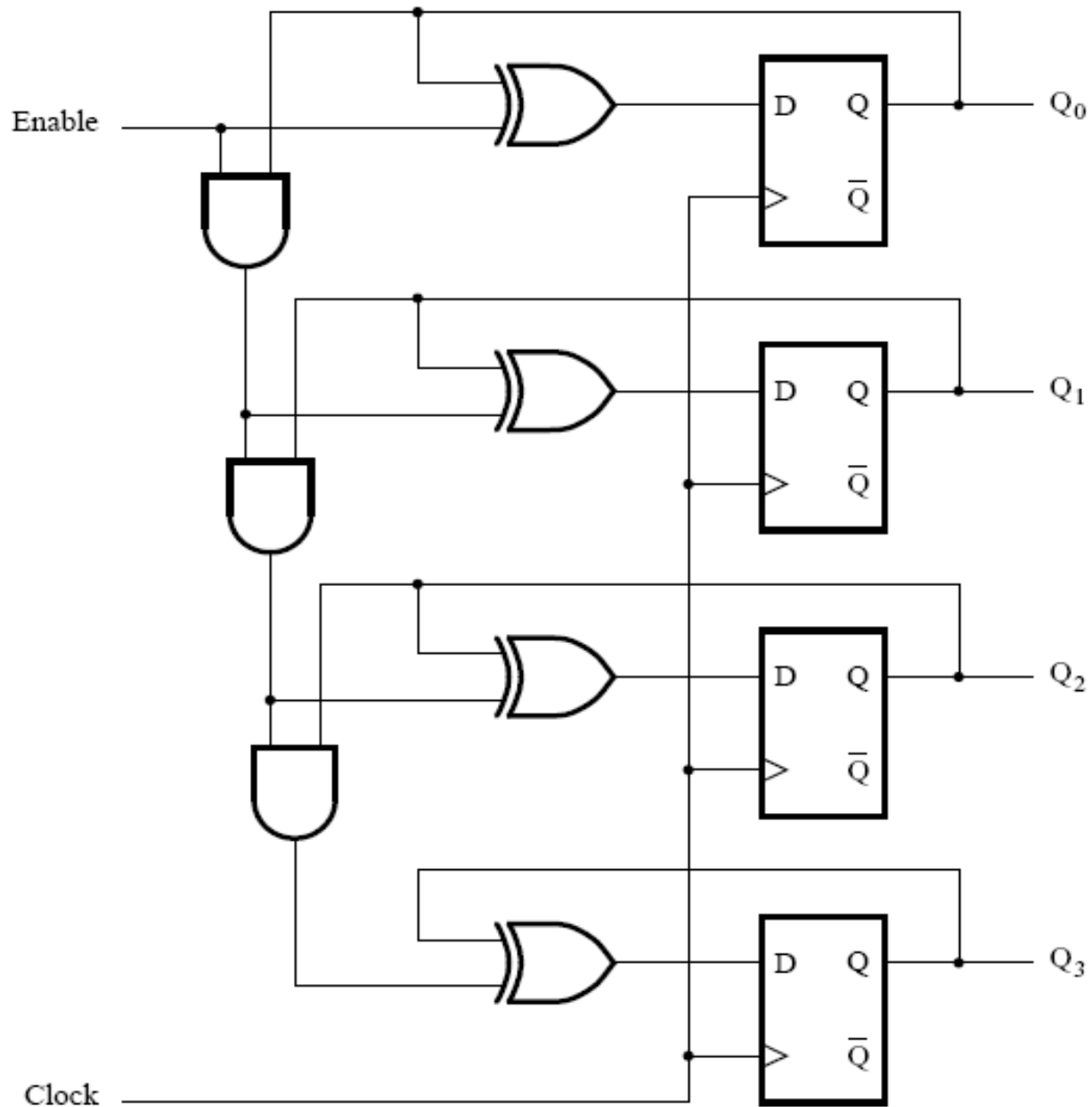
# Faster 4-bit Counter

- **Want to increase the speed of the 4-bit counter?**
- **Use a similar method as the one used in 4-bit adder.**
- **Replace the series of 2-input AND gates with AND gates with more input lines.**

# Is it possible to reduce this delay?

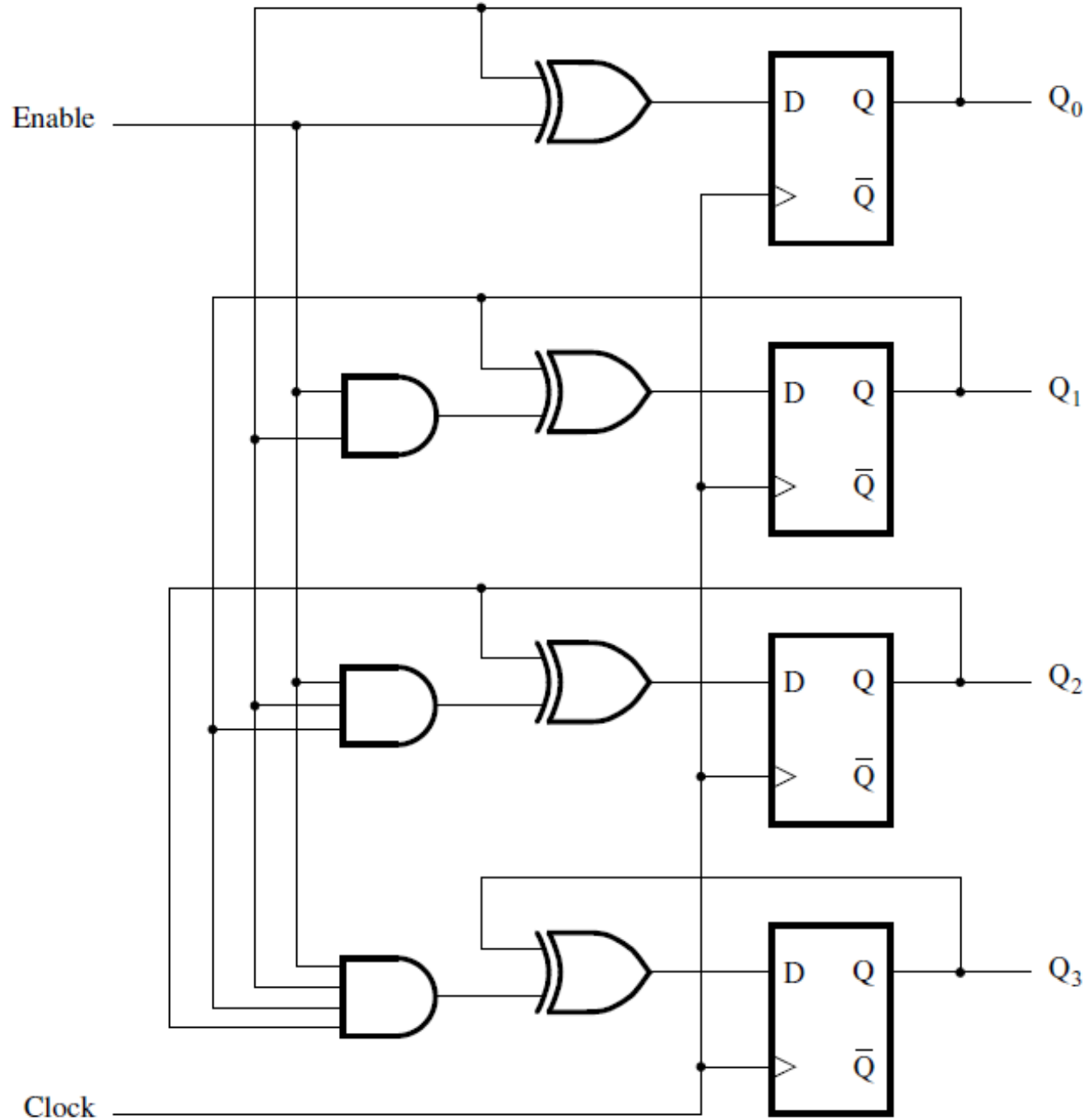


# Equivalent 4-bit counter



[ Figure 5.67 from the textbook ]

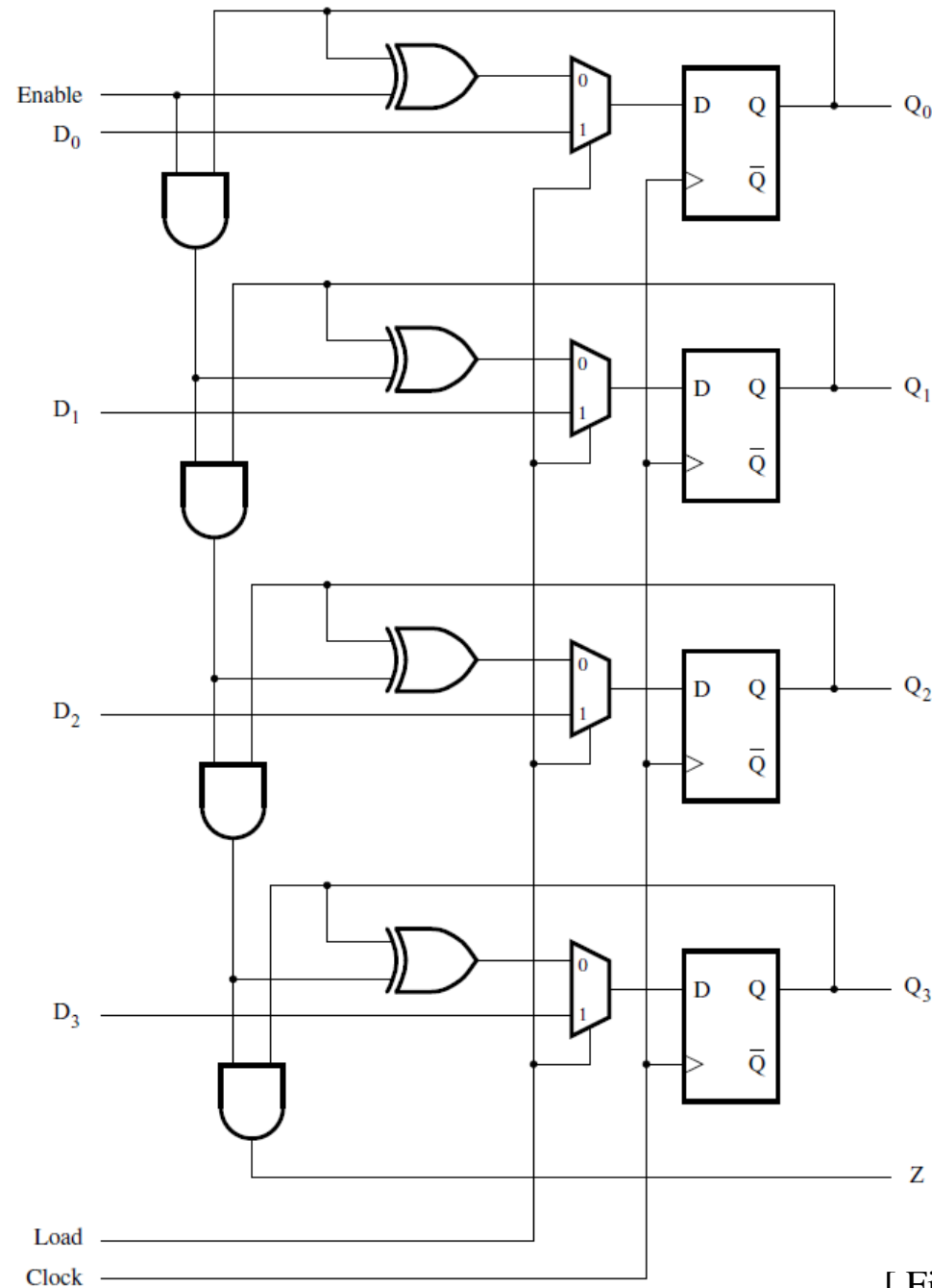
# A faster 4-bit counter



[ Figure 5.75 from the textbook ]

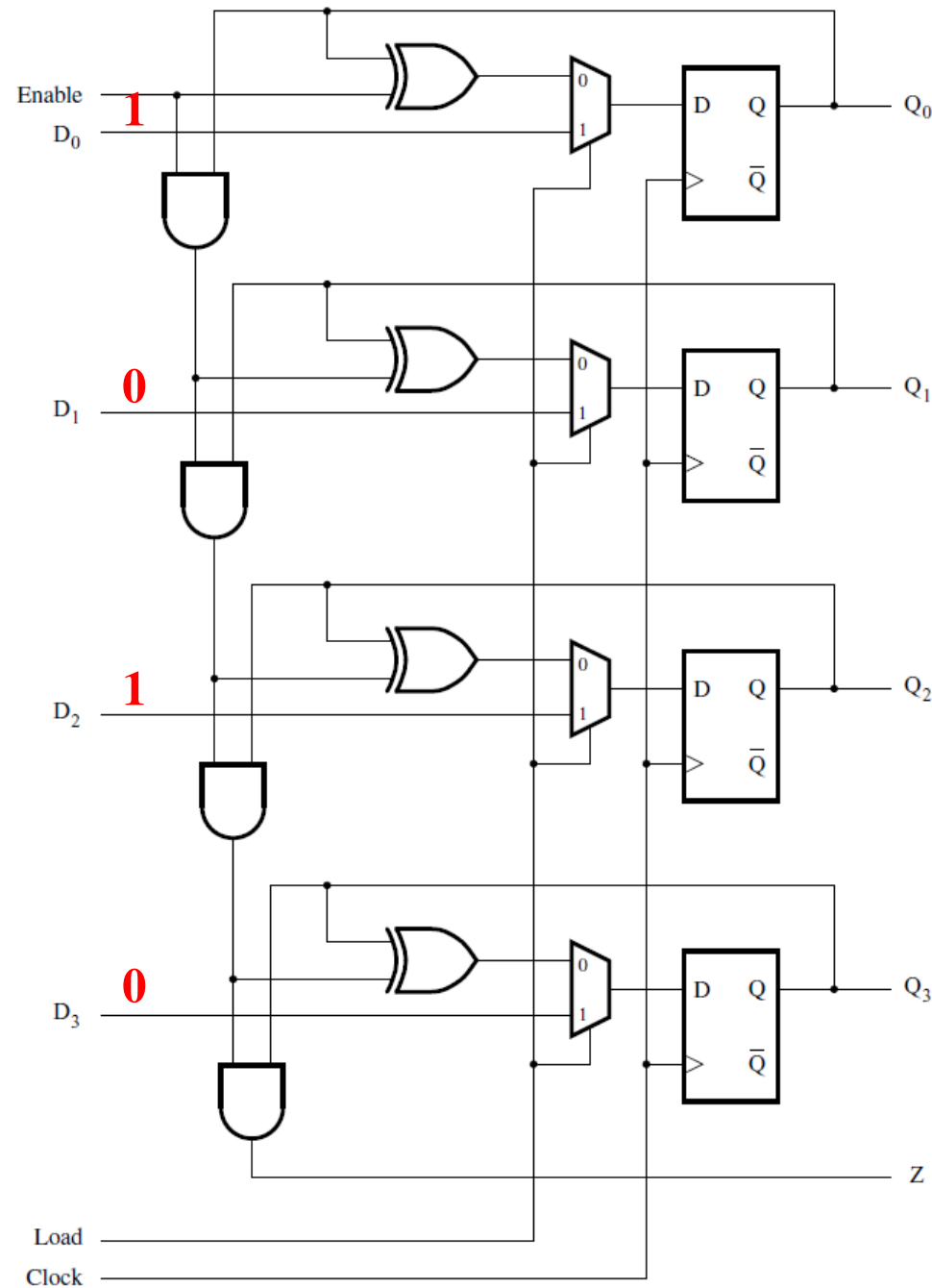
# **Counters with Parallel Load**

# A counter with parallel-load capability



[ Figure 5.24 from the textbook ]

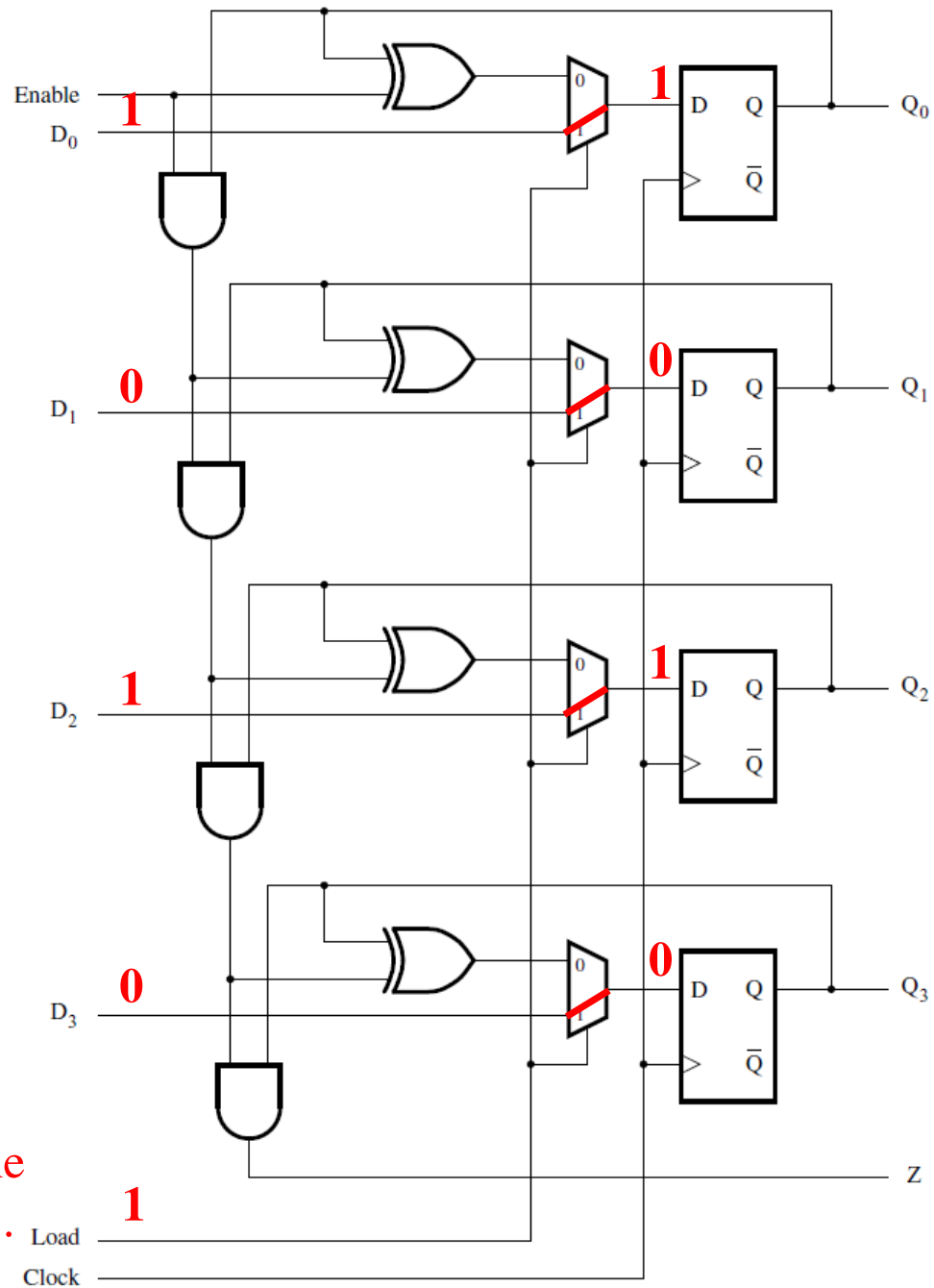
# How to load the initial count value



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

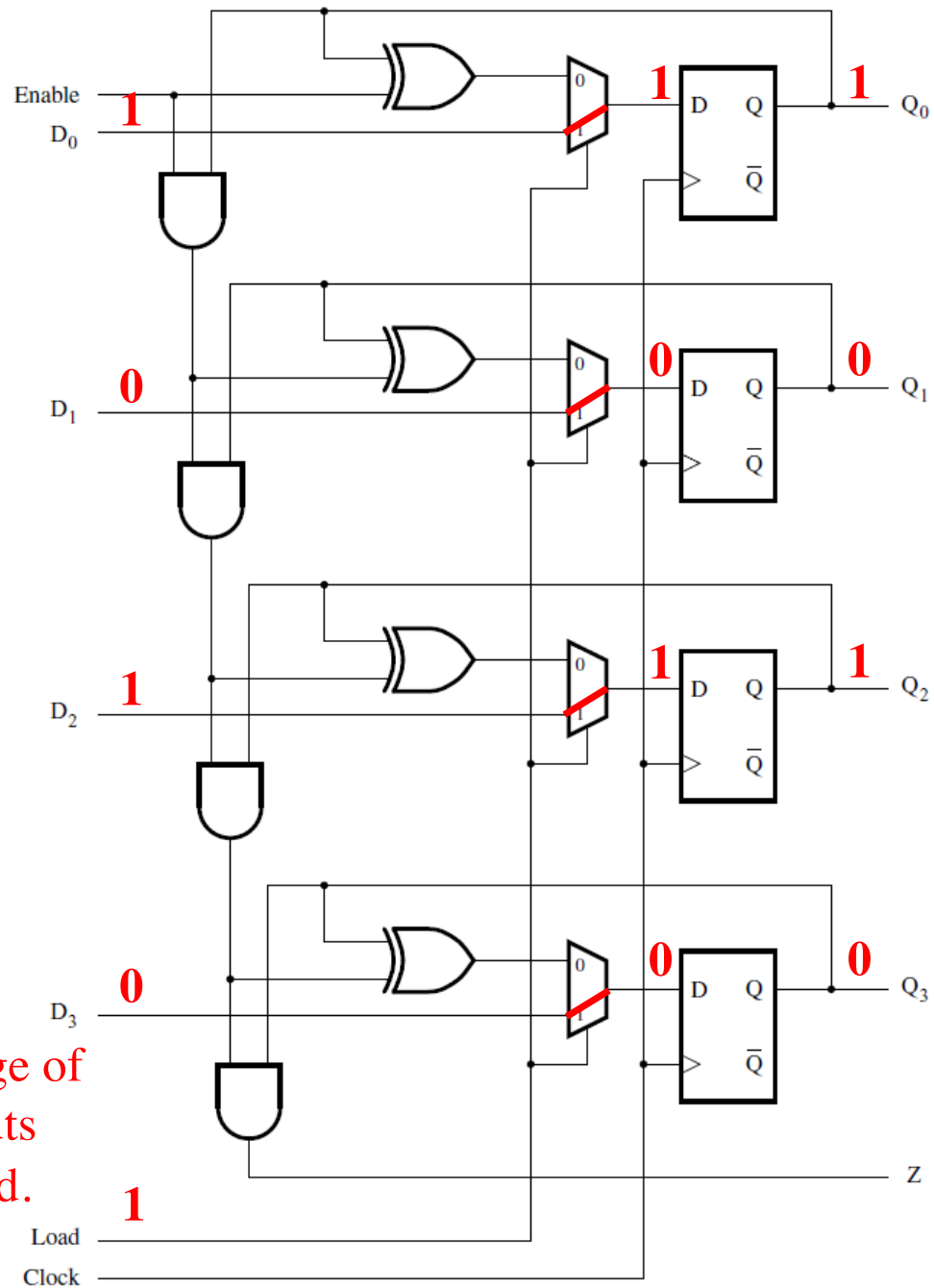


# How to zero a counter



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

# How to zero a counter



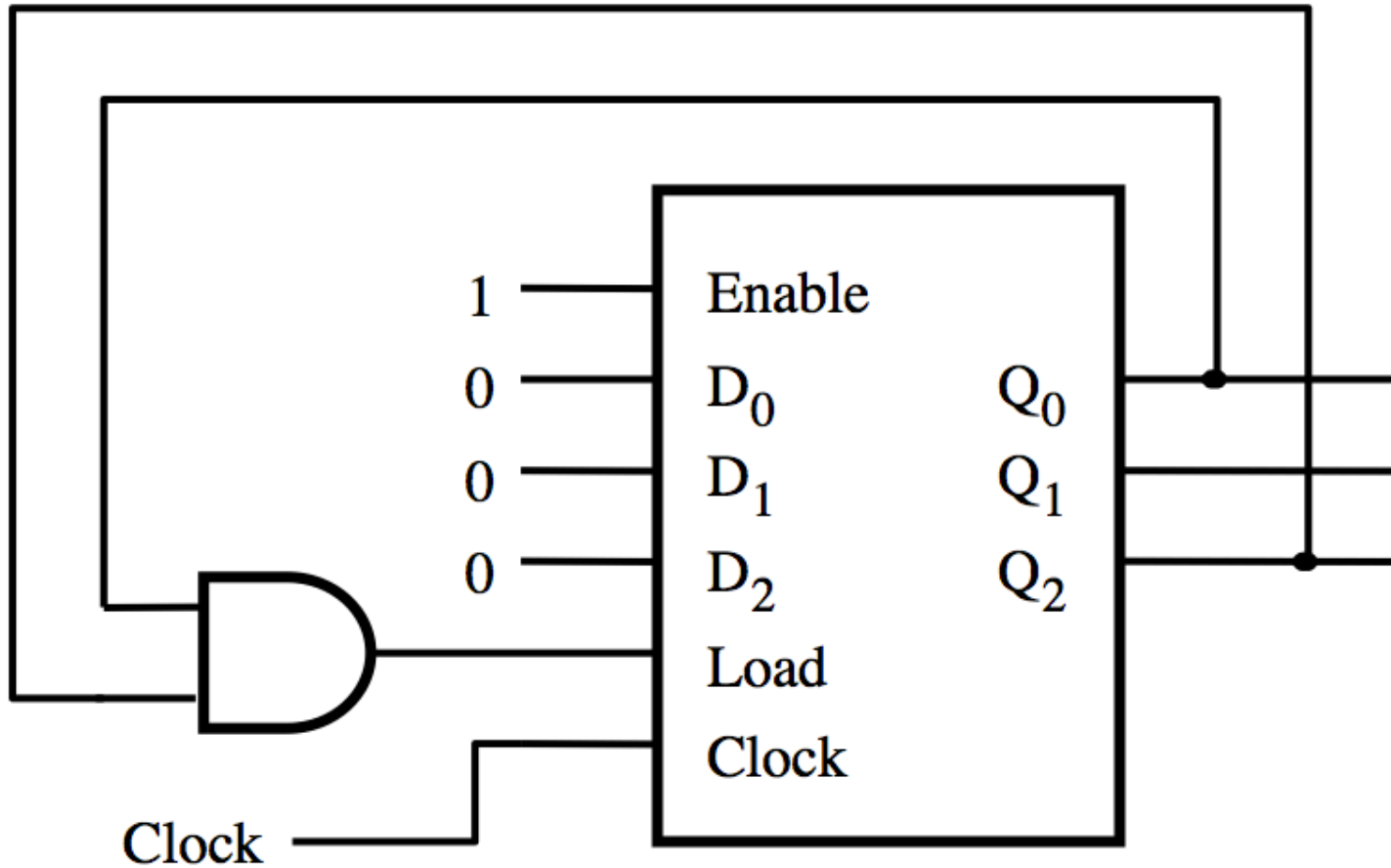
When the next positive edge of the clock arrives, the outputs of the flip-flops are updated.

# **Reset Synchronization**

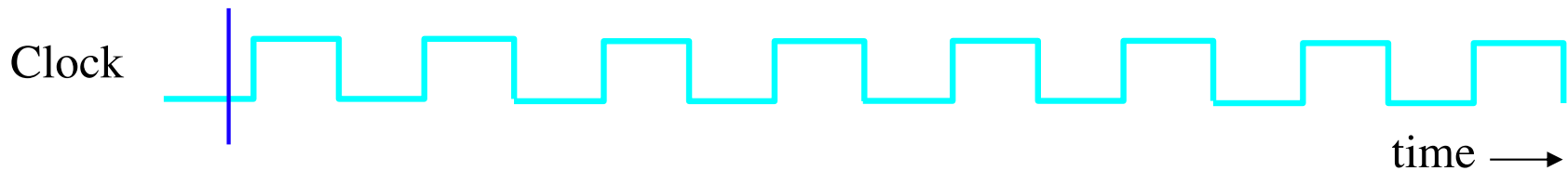
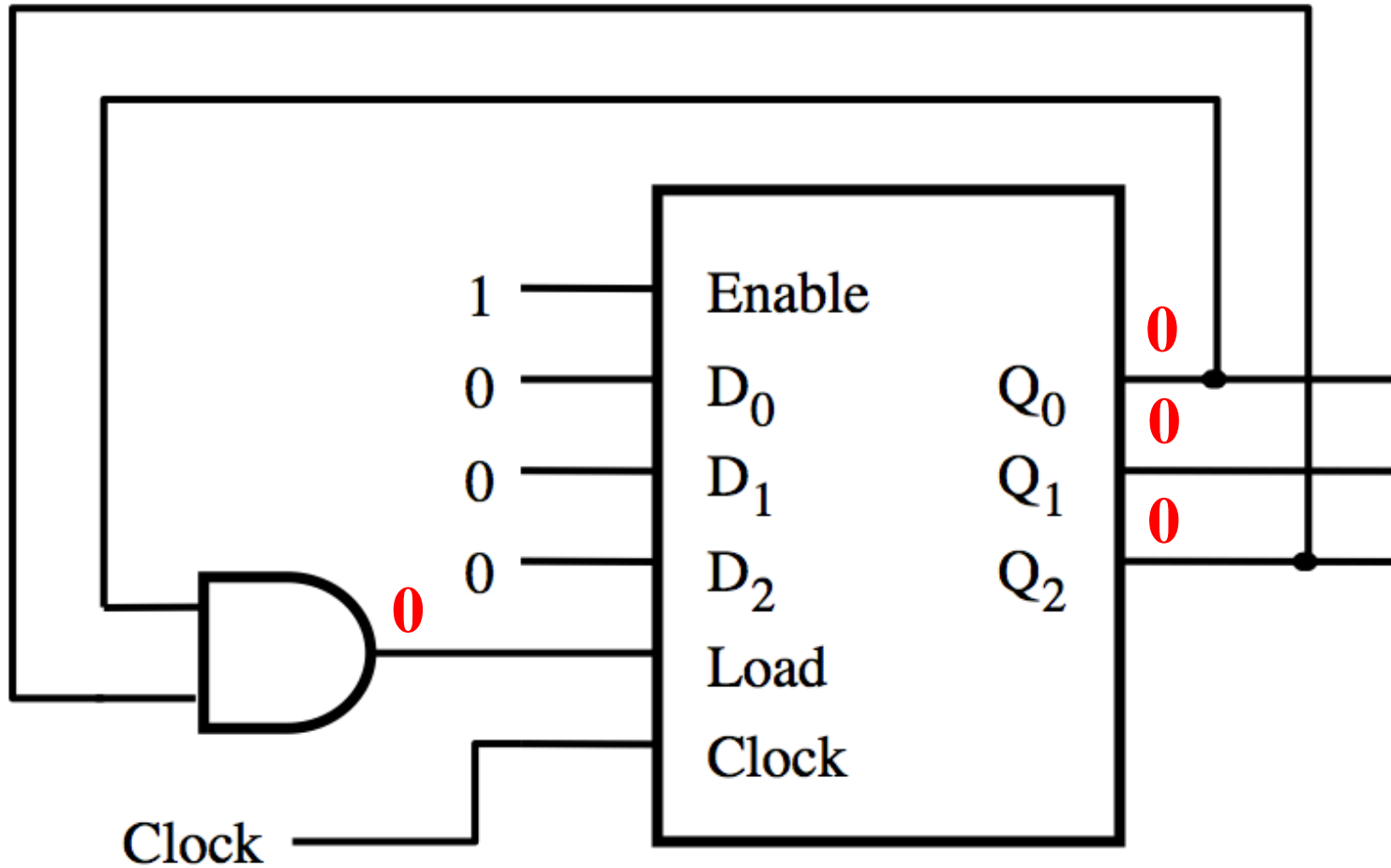
# Motivation

- **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?**

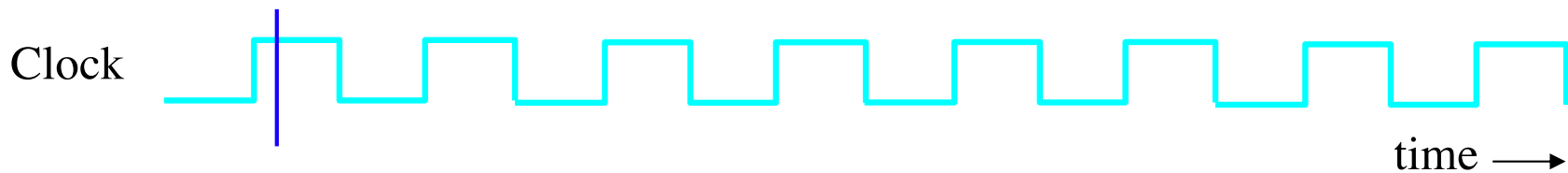
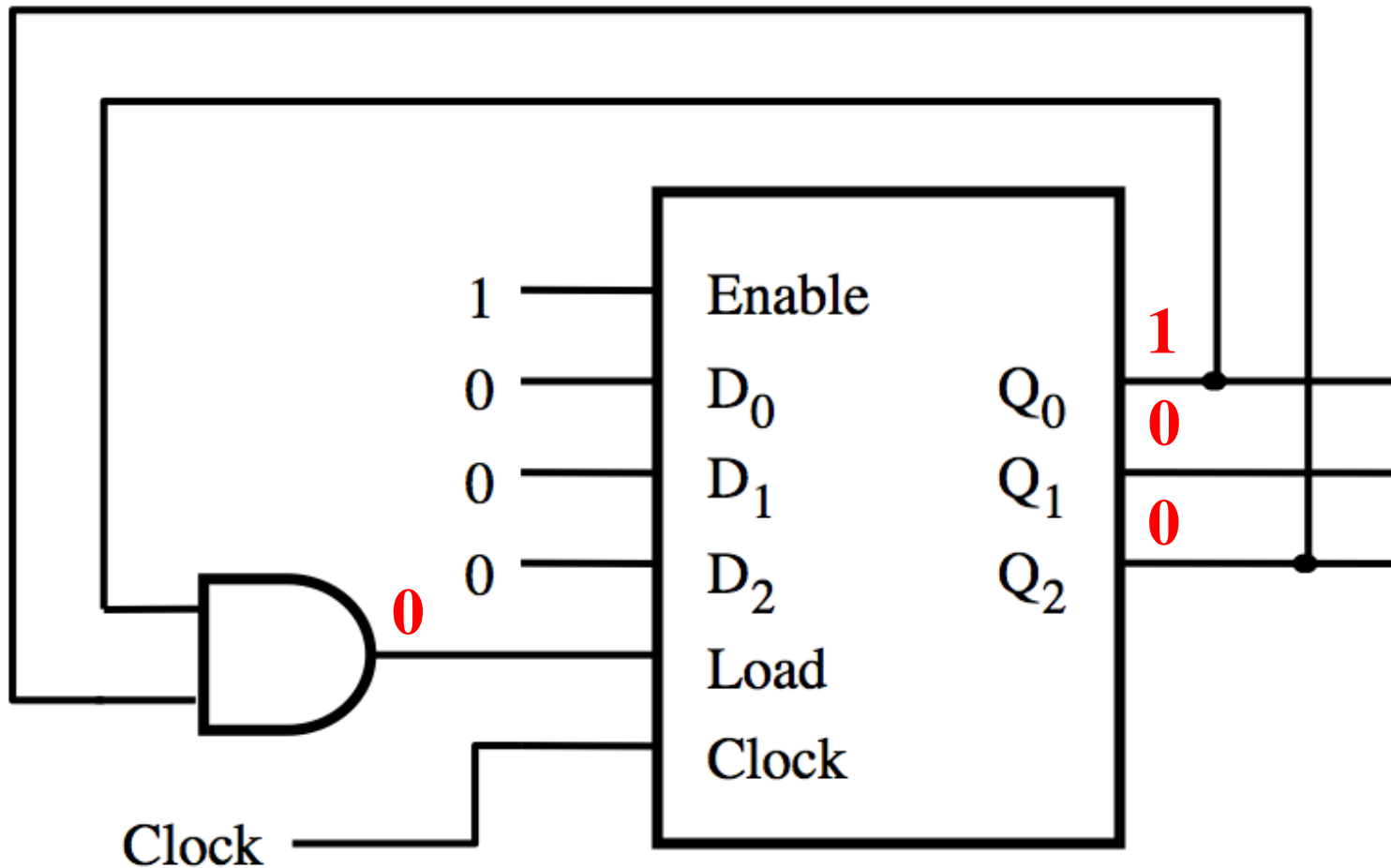
# What does this circuit do?



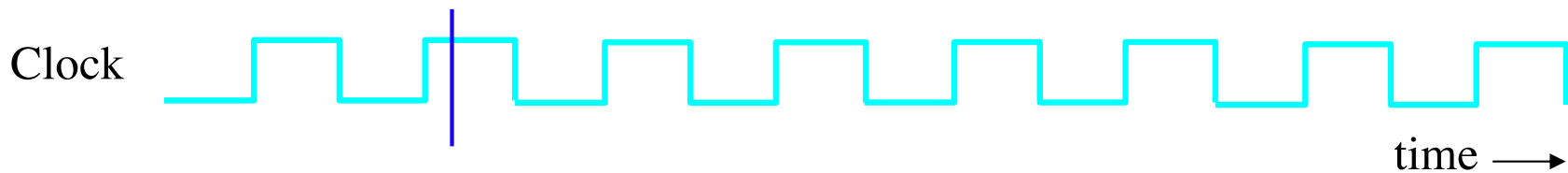
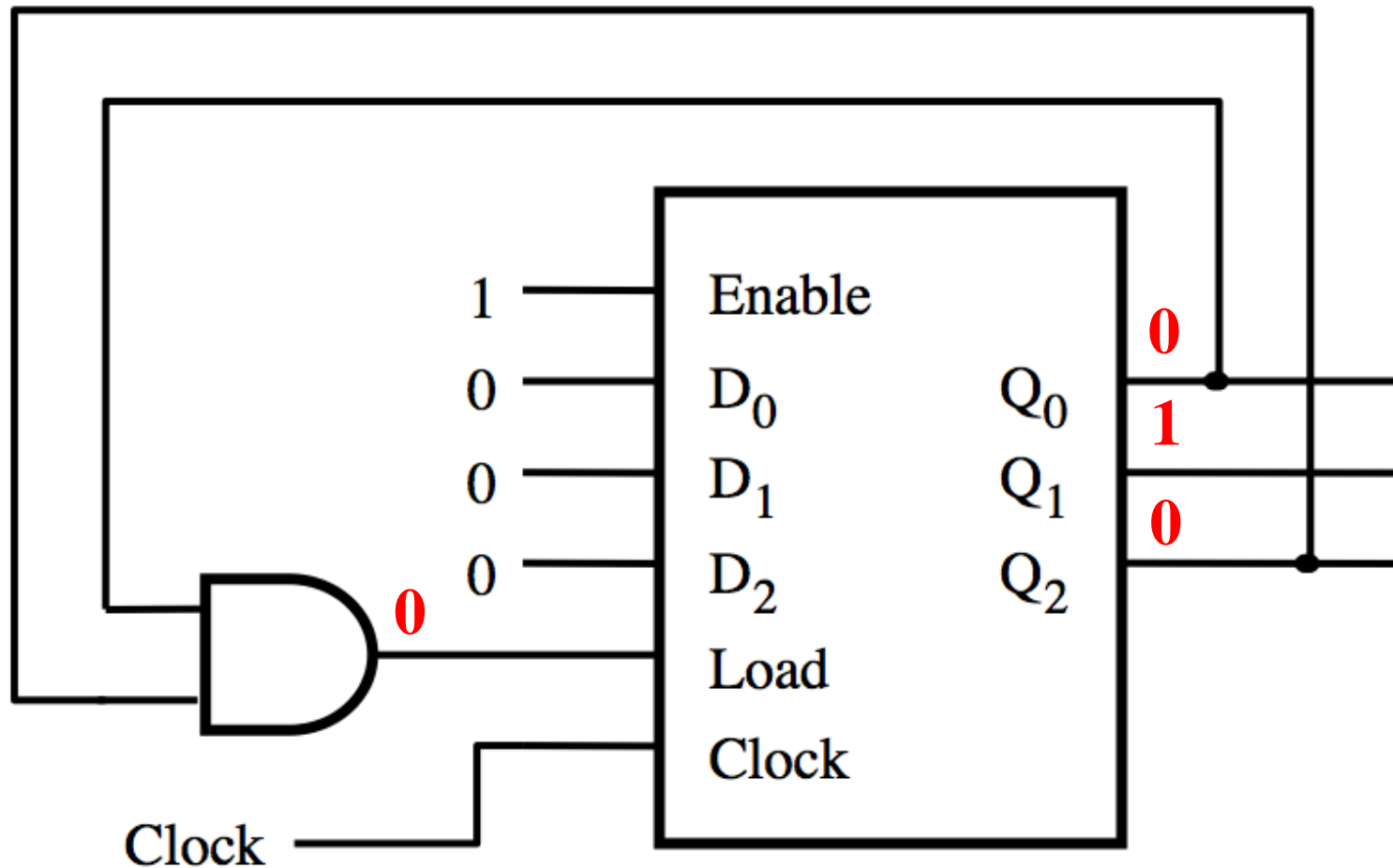
# What does this circuit do?



# What does this circuit do?

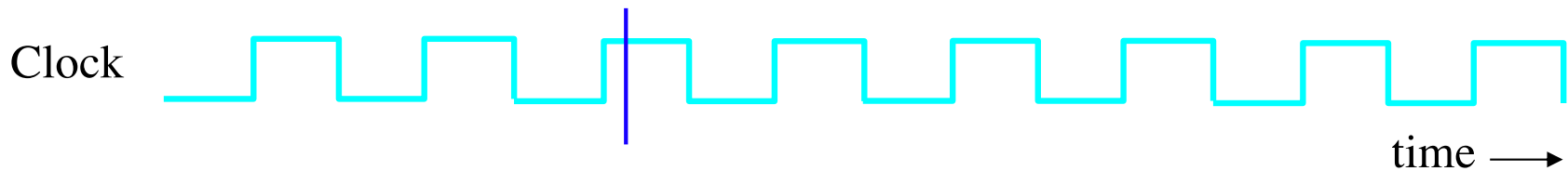
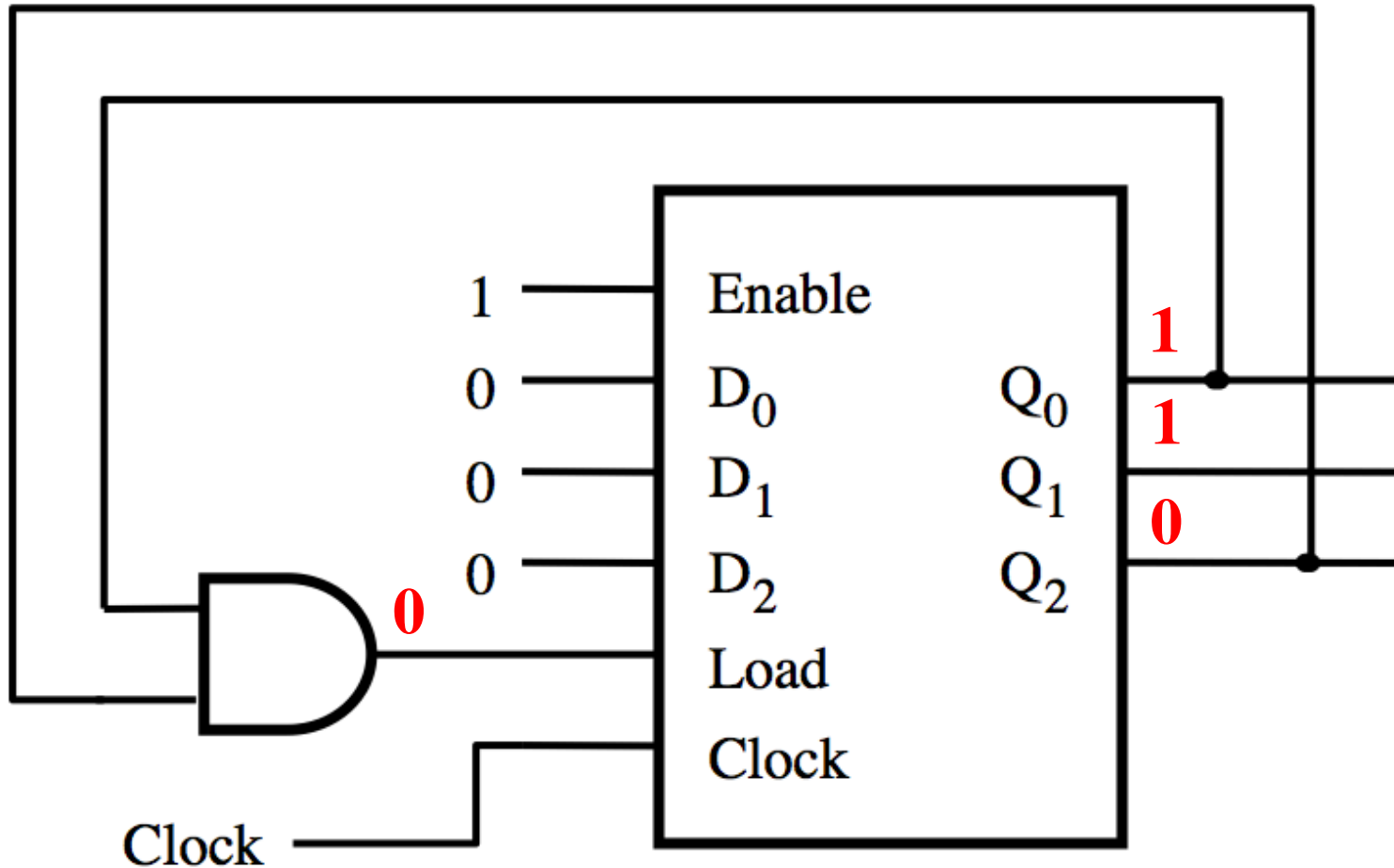


# What does this circuit do?

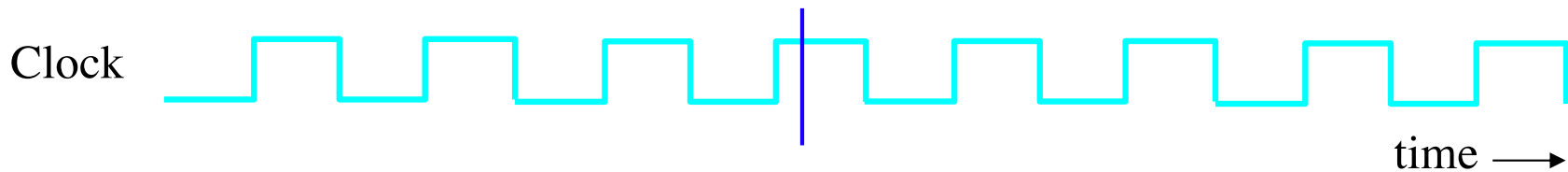
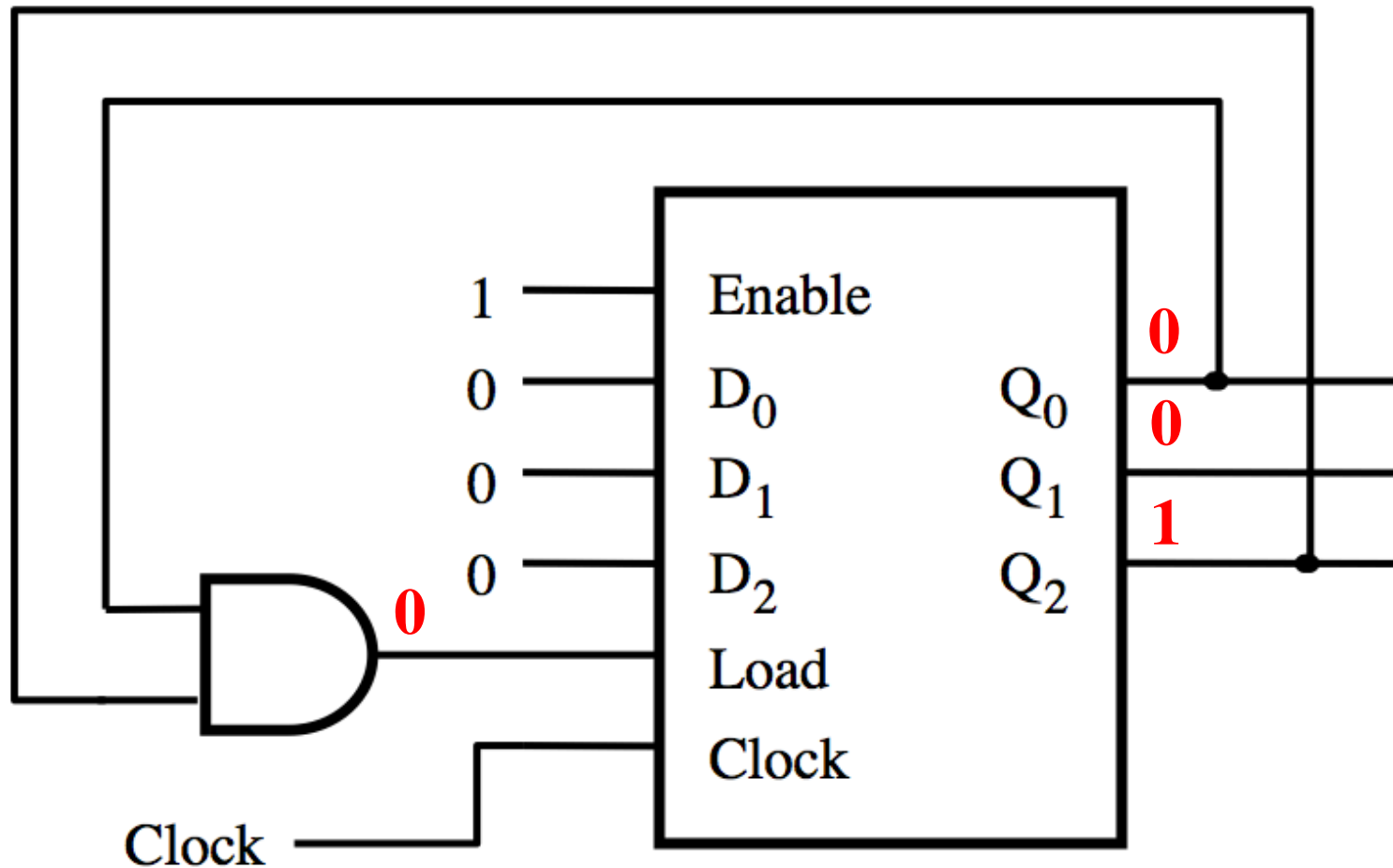




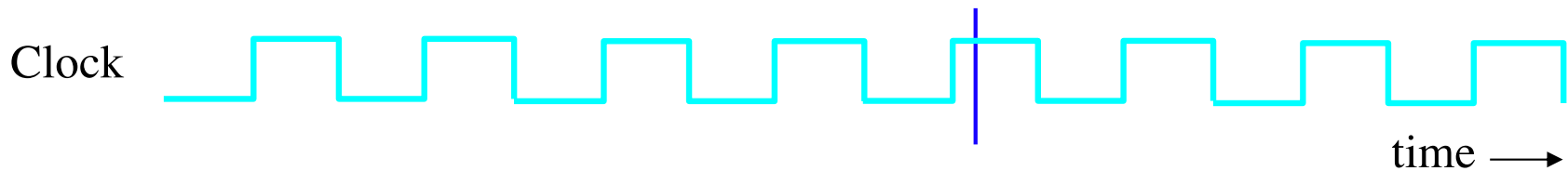
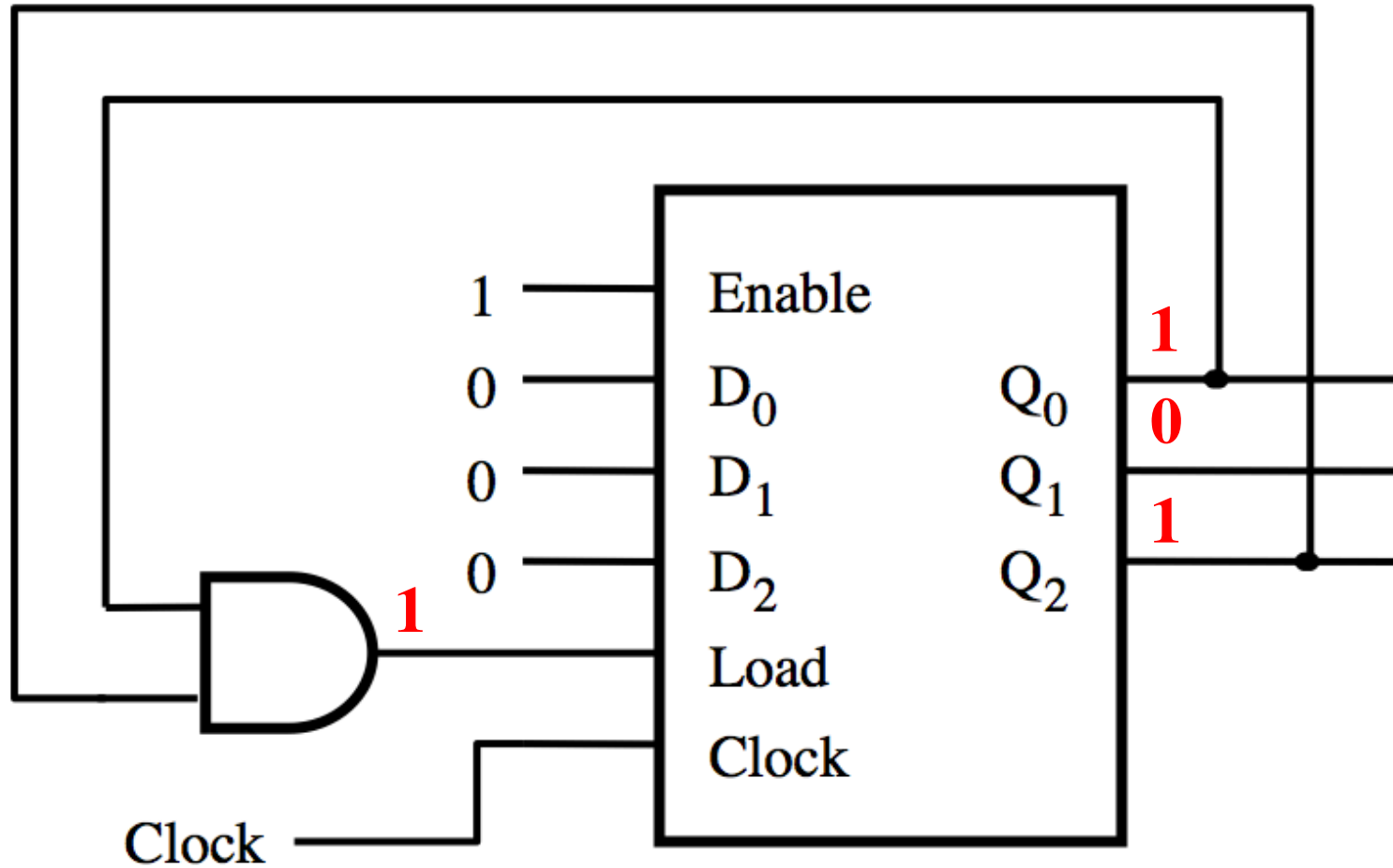
# What does this circuit do?



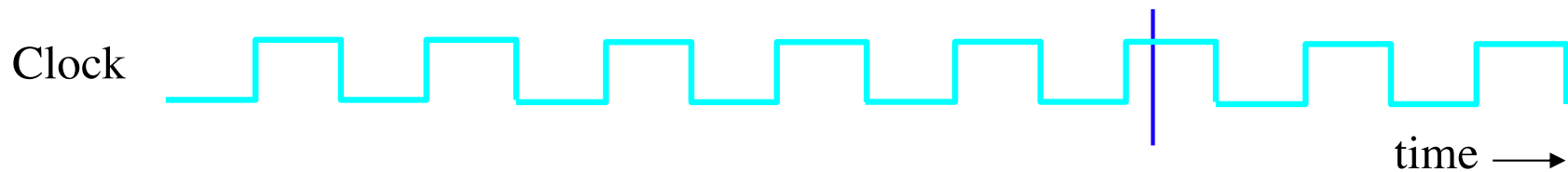
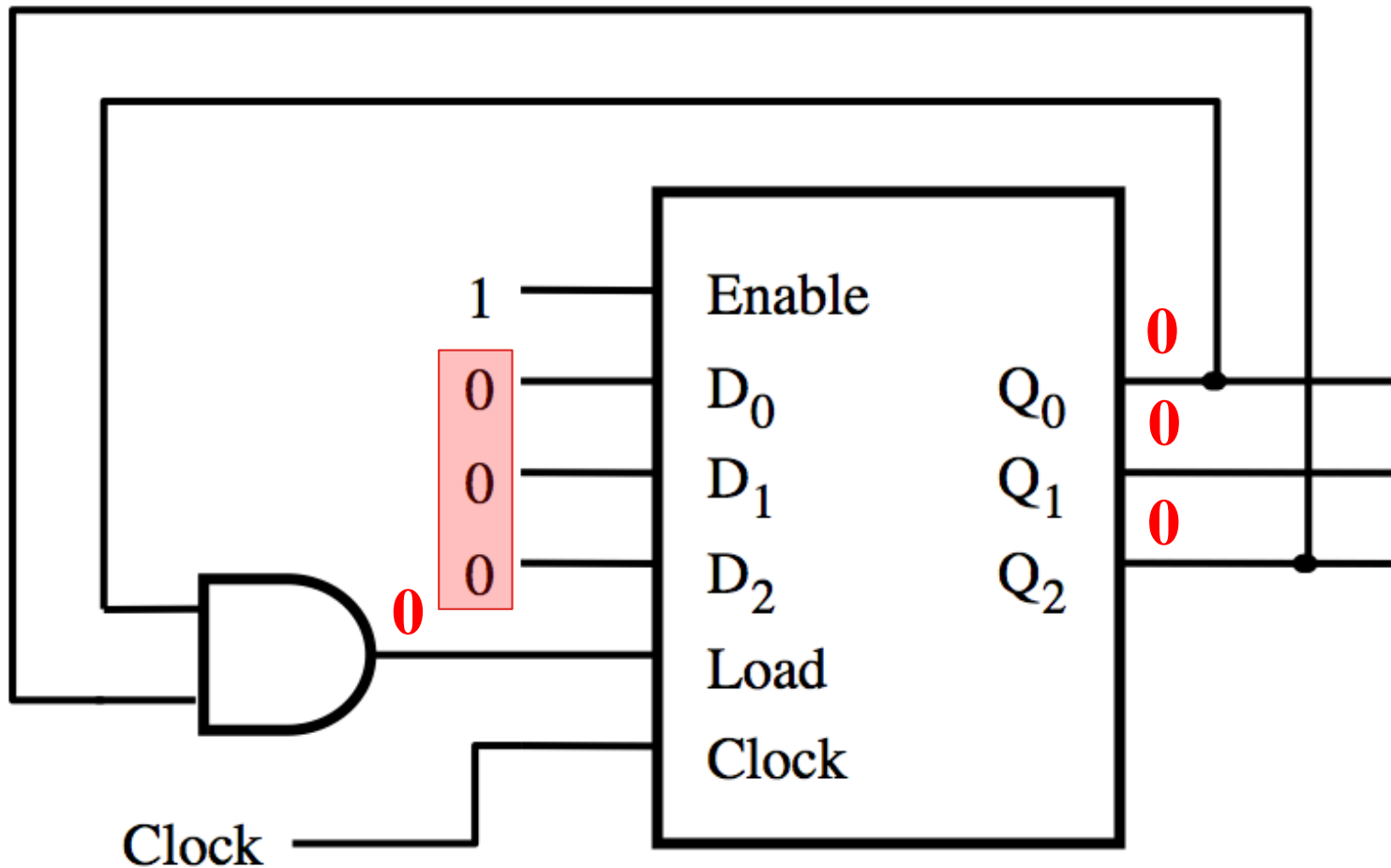
# What does this circuit do?



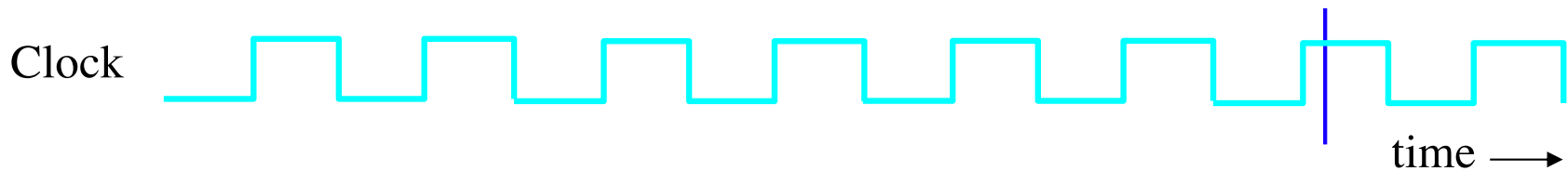
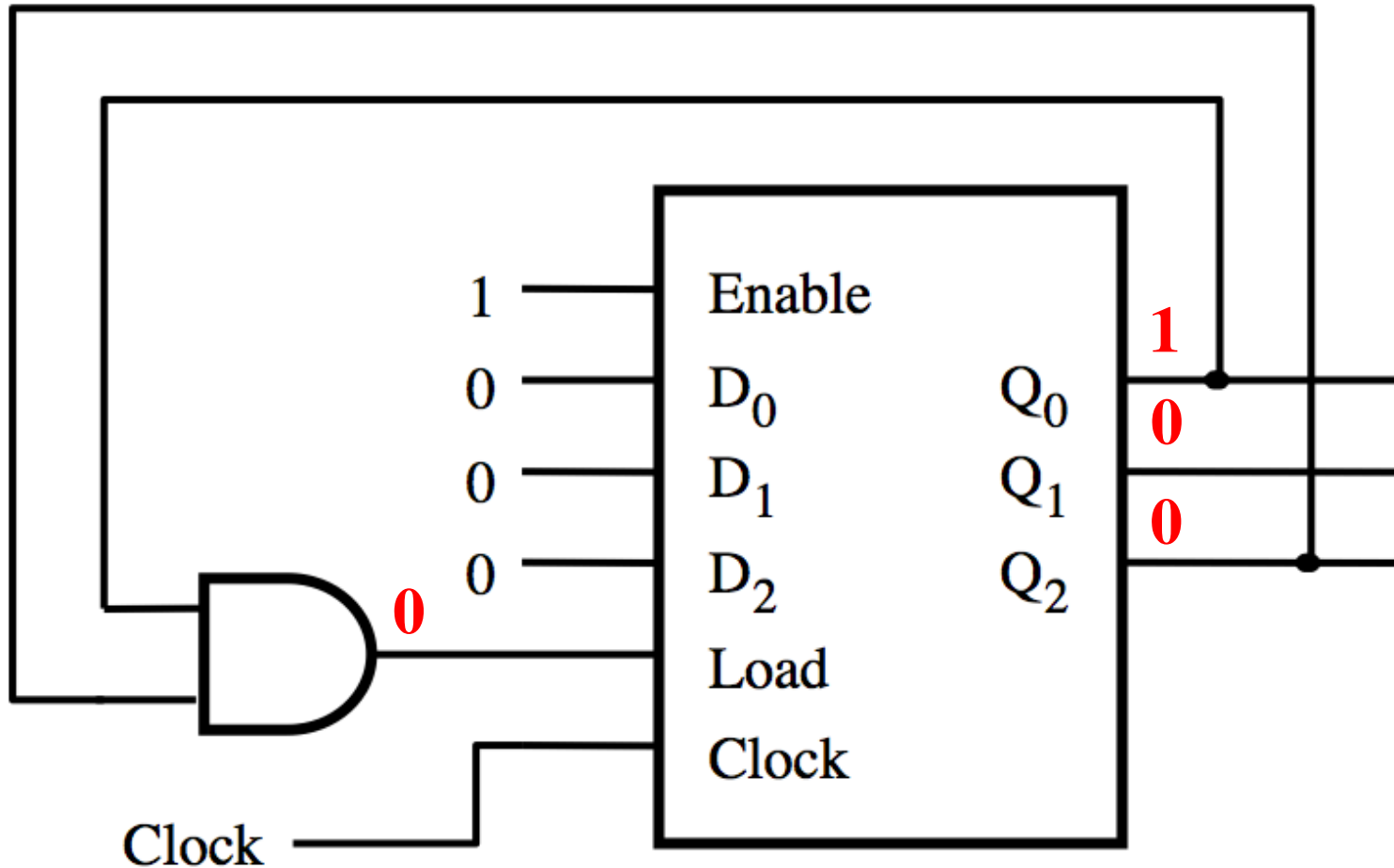
# What does this circuit do?



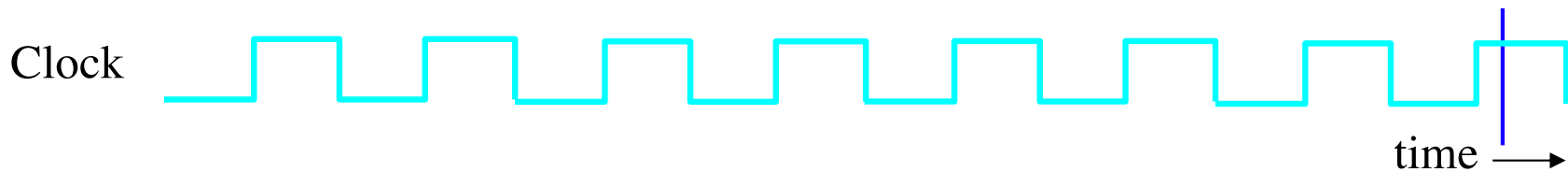
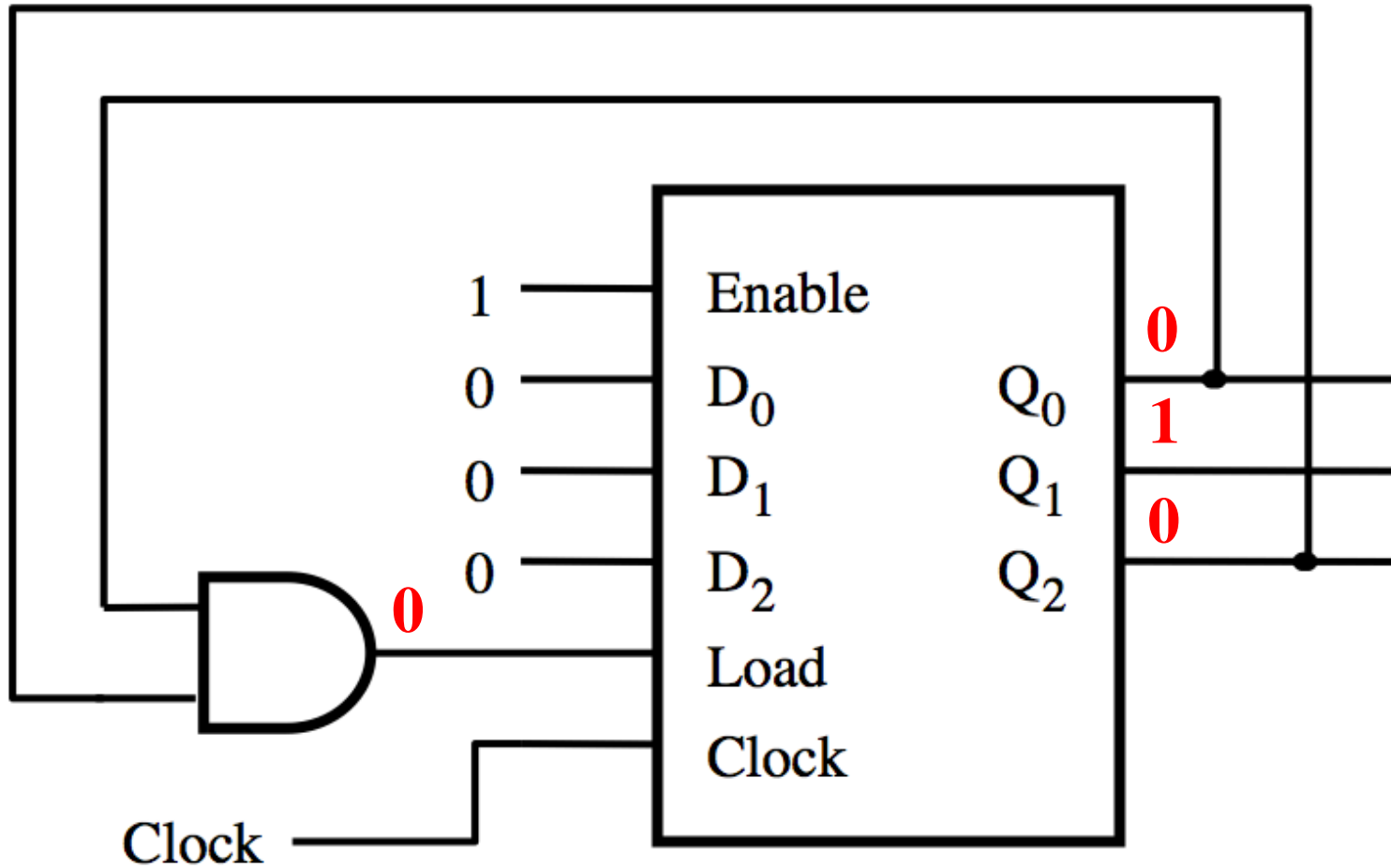
# What does this circuit do?



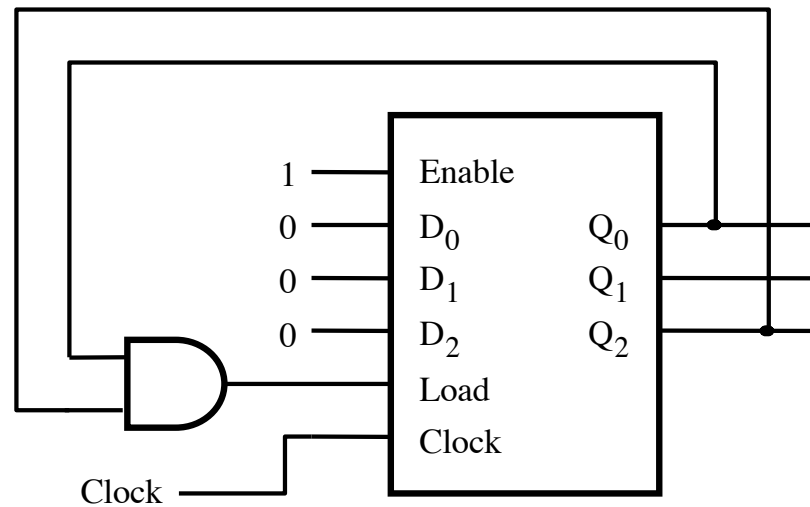
# What does this circuit do?



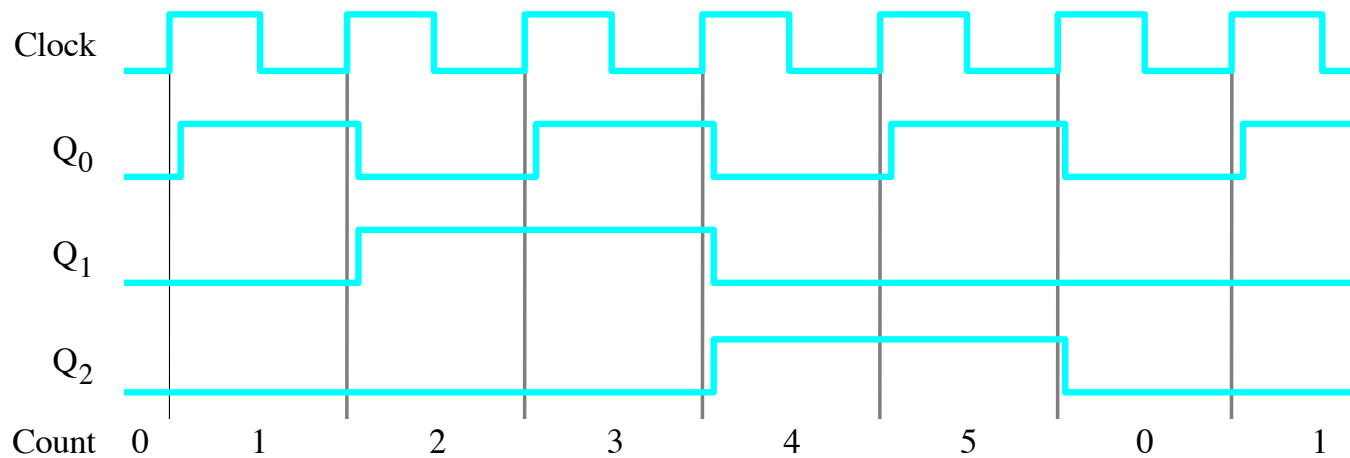
# What does this circuit do?



# A modulo-6 counter with synchronous reset

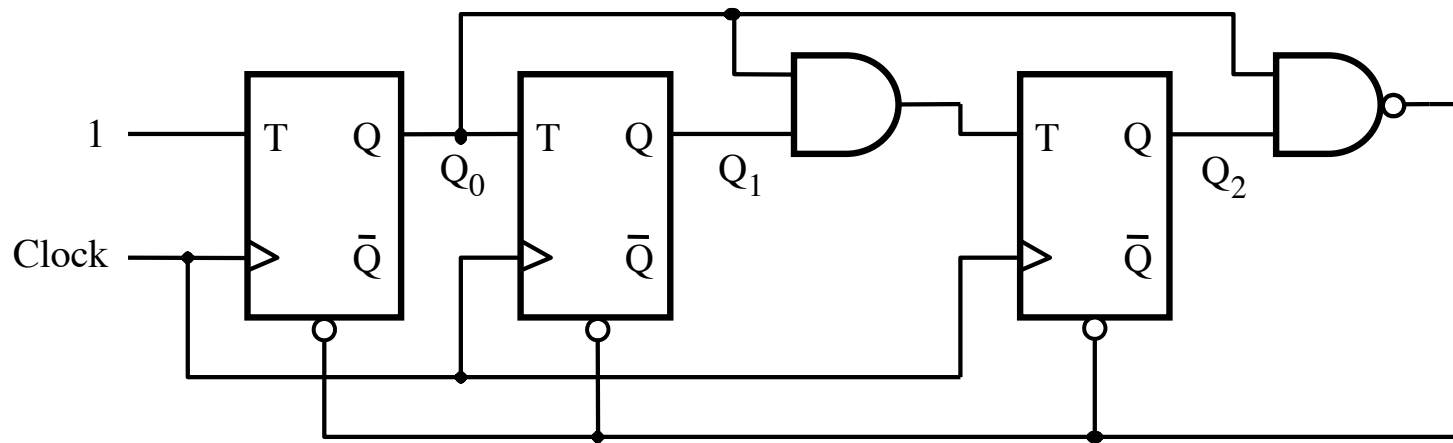


(a) Circuit

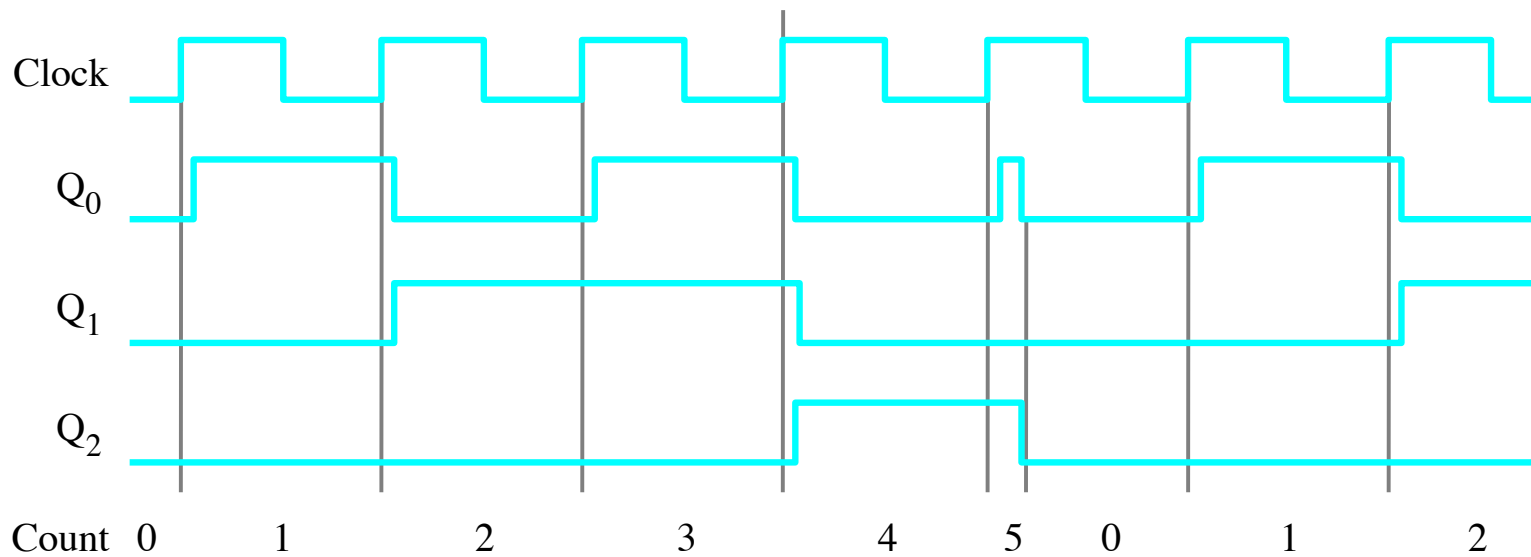


(b) Timing diagram

# A modulo-6 counter with asynchronous reset



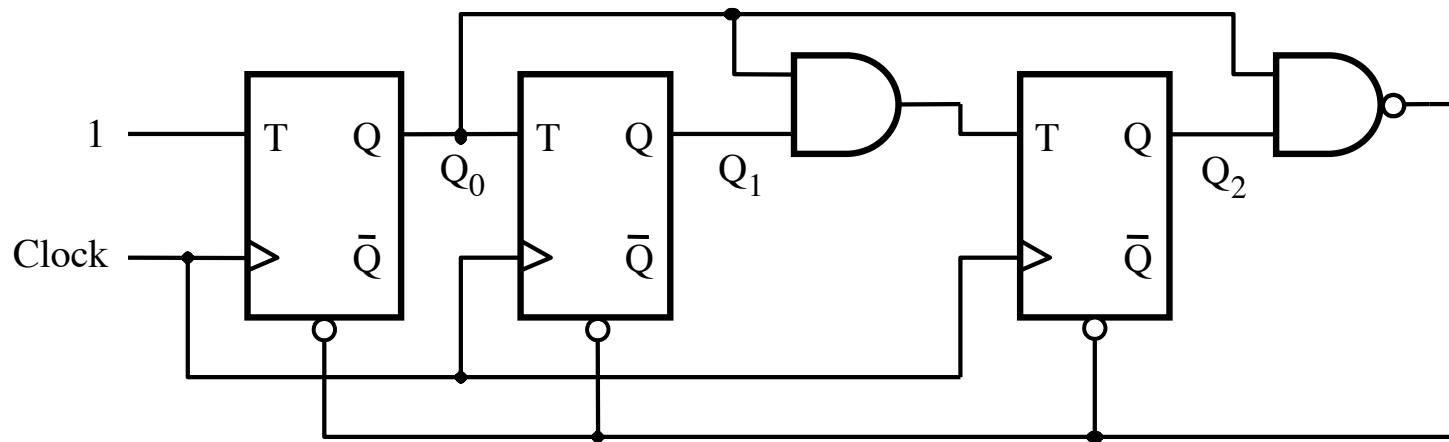
(a) Circuit



(b) Timing diagram

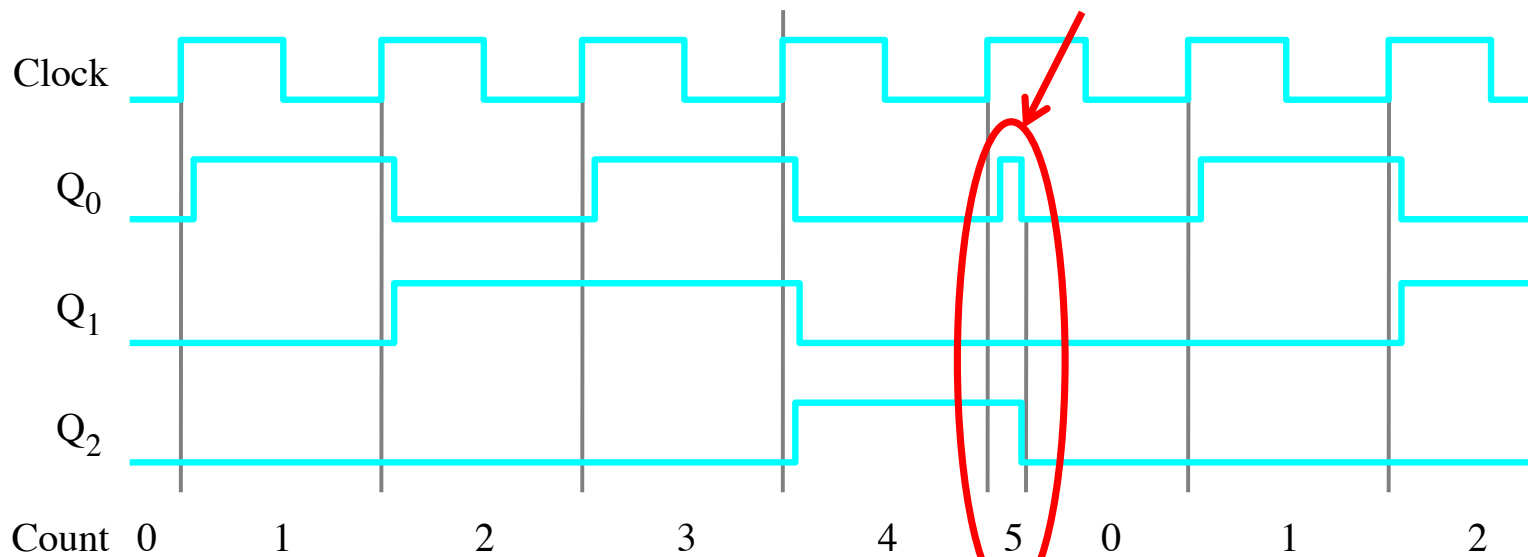


# A modulo-6 counter with asynchronous reset



(a) Circuit

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



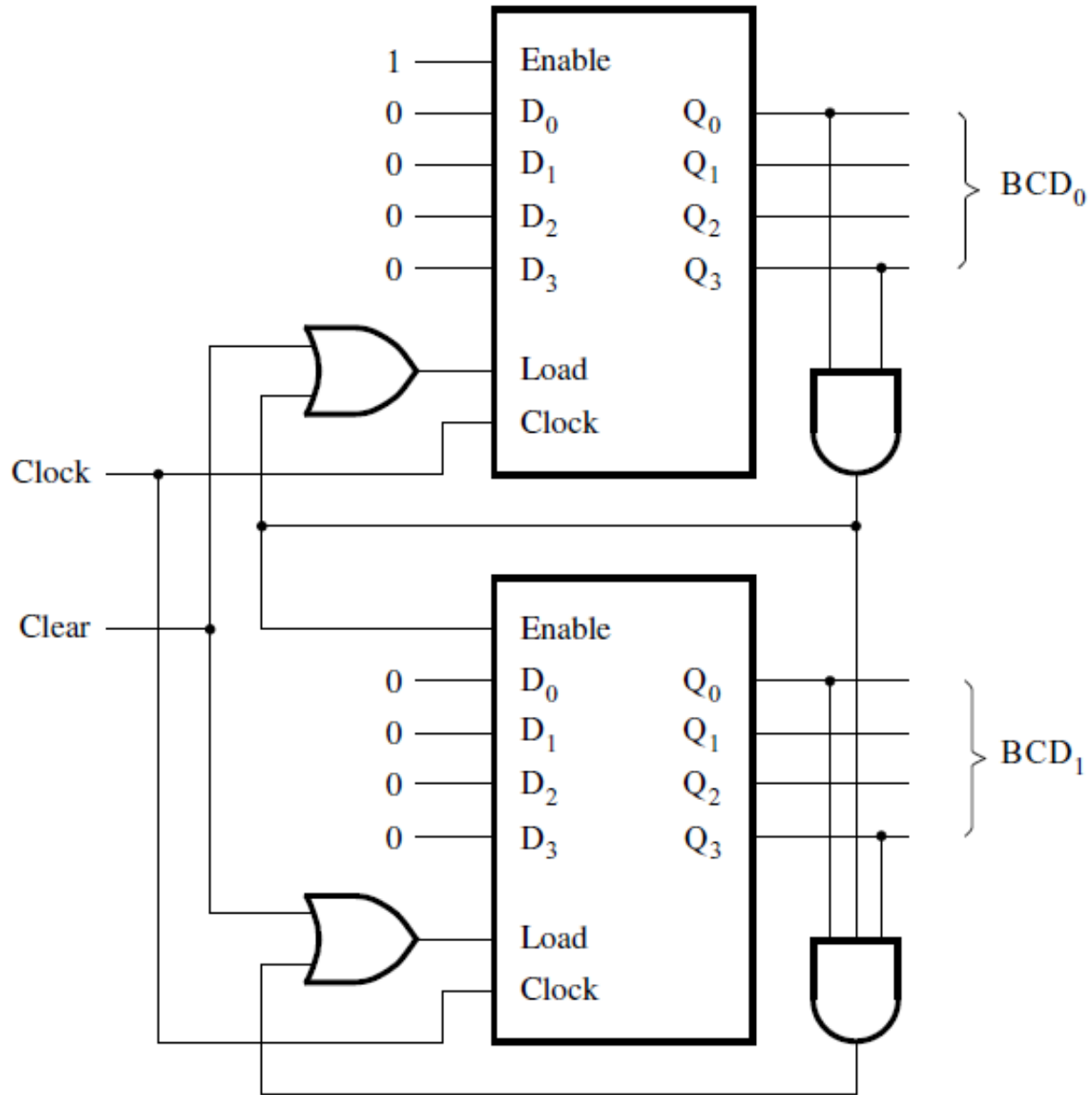
(b) Timing diagram

# **Other Types of Counters (Section 5.11)**

# A two-digit BCD counter

- **Use Two Parallel-load four-bit counters**
  - Figure 5.24
- **Each counts in binary**
  - 0-9
- **Resets generated on 9**
  - Reset by loading 0's
- **Second digit enabled by a 9 on first counter**

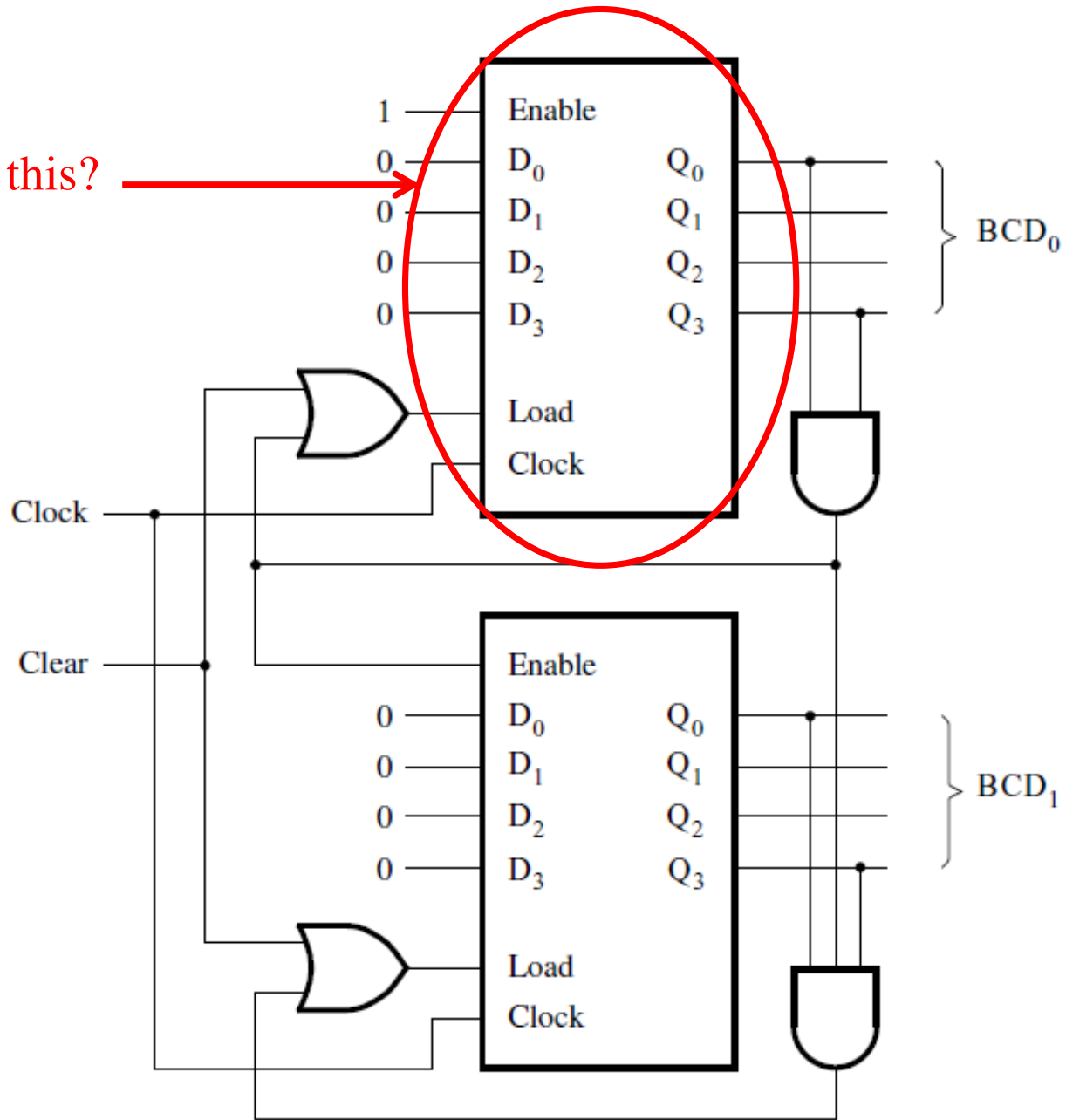
# A two-digit BCD counter



[ Figure 5.27 from the textbook ]

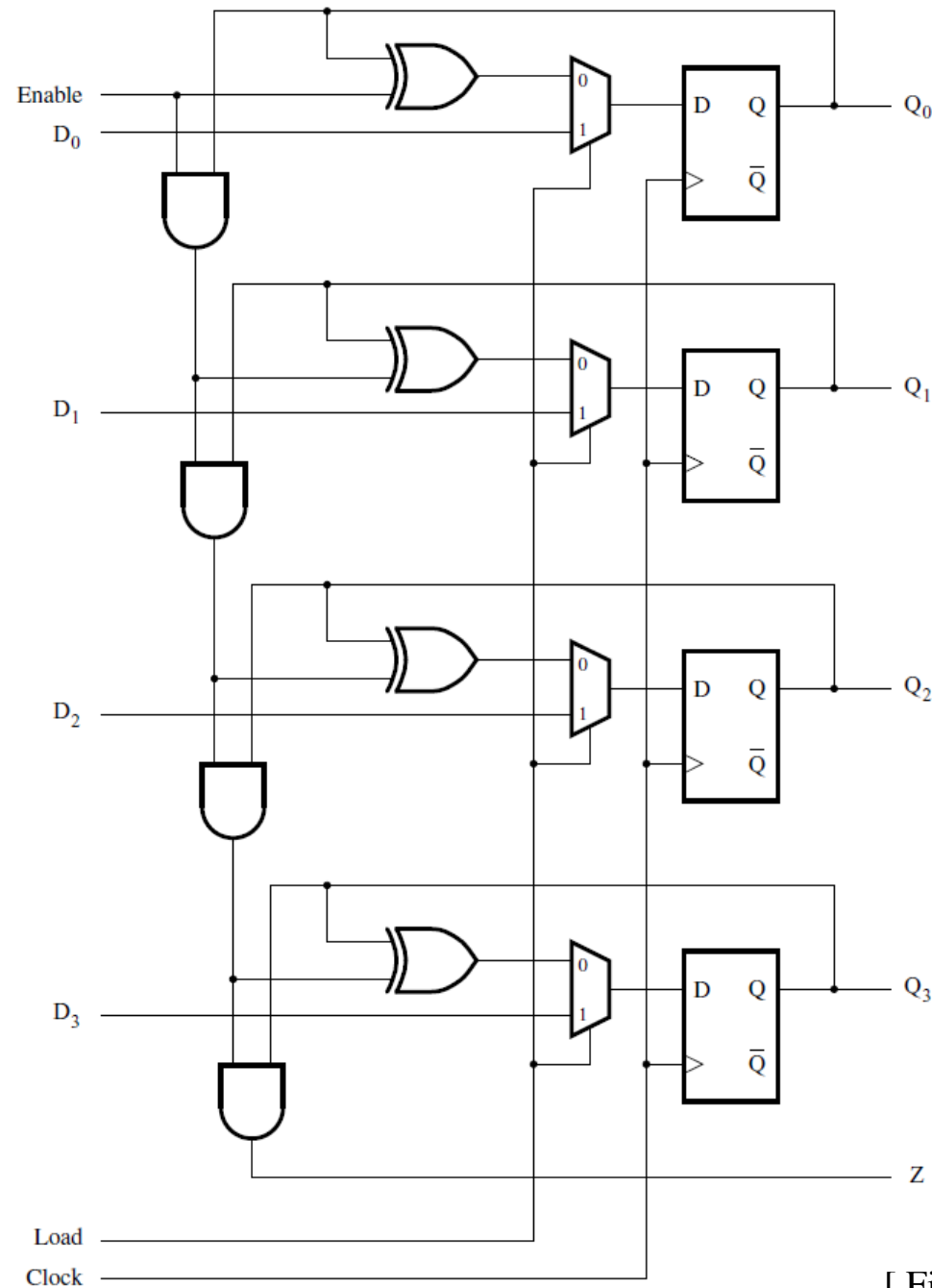
# A two-digit BCD counter

What is this? →



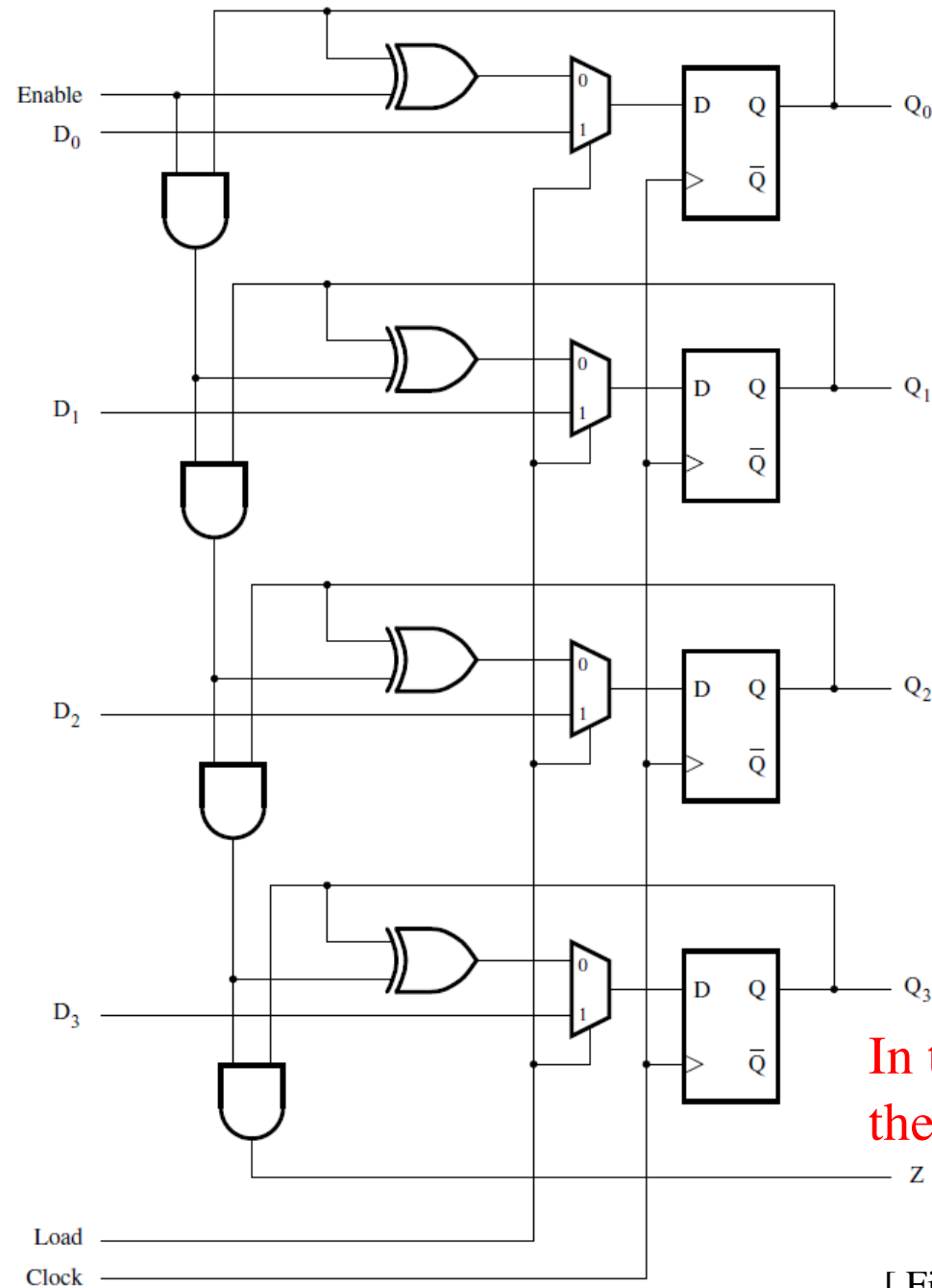
[ Figure 5.27 from the textbook ]

# It is a counter with parallel-load capability



[ Figure 5.24 from the textbook ]

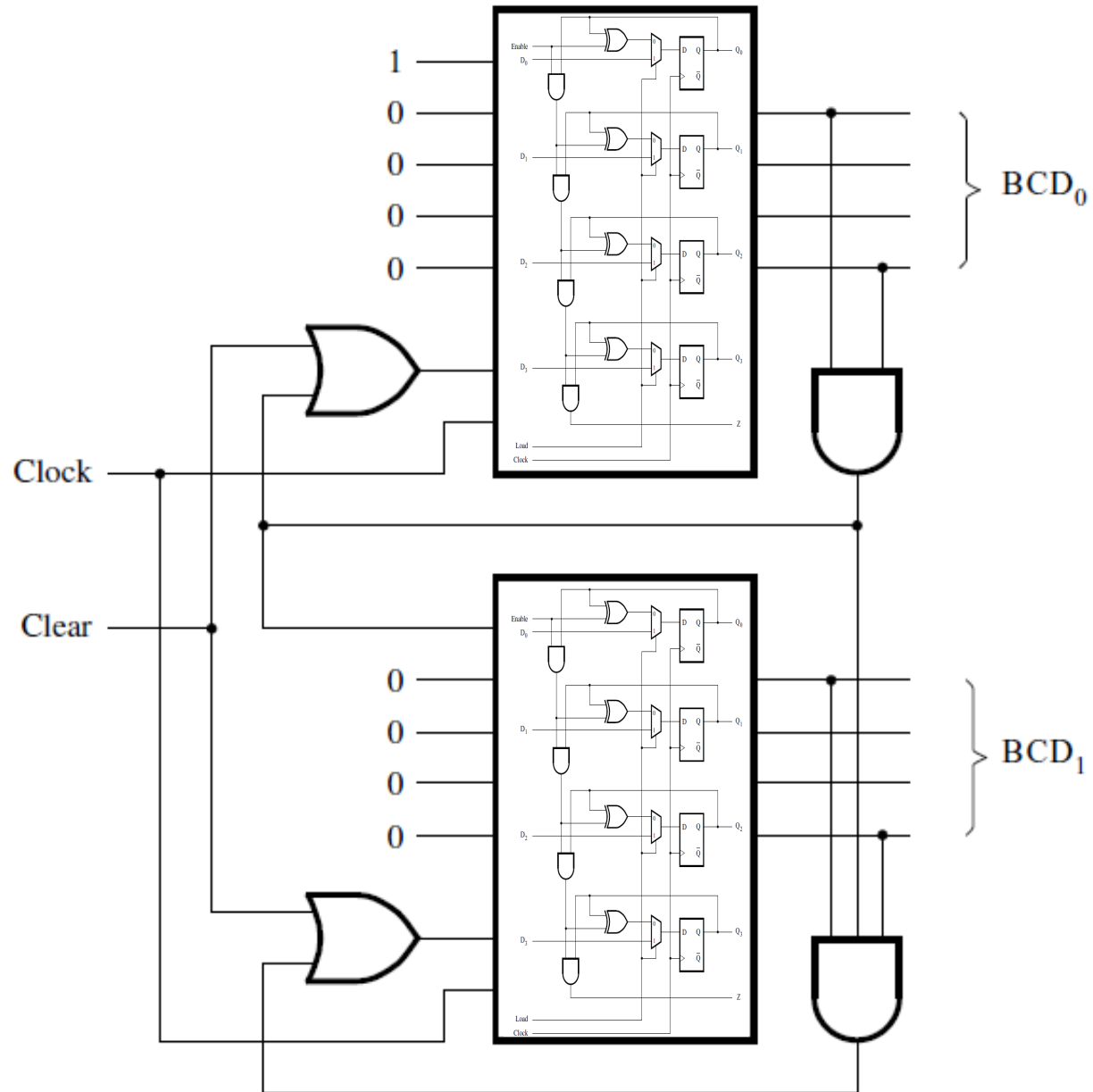
# It is a counter with parallel-load capability



In this case,  
the z output is ignored

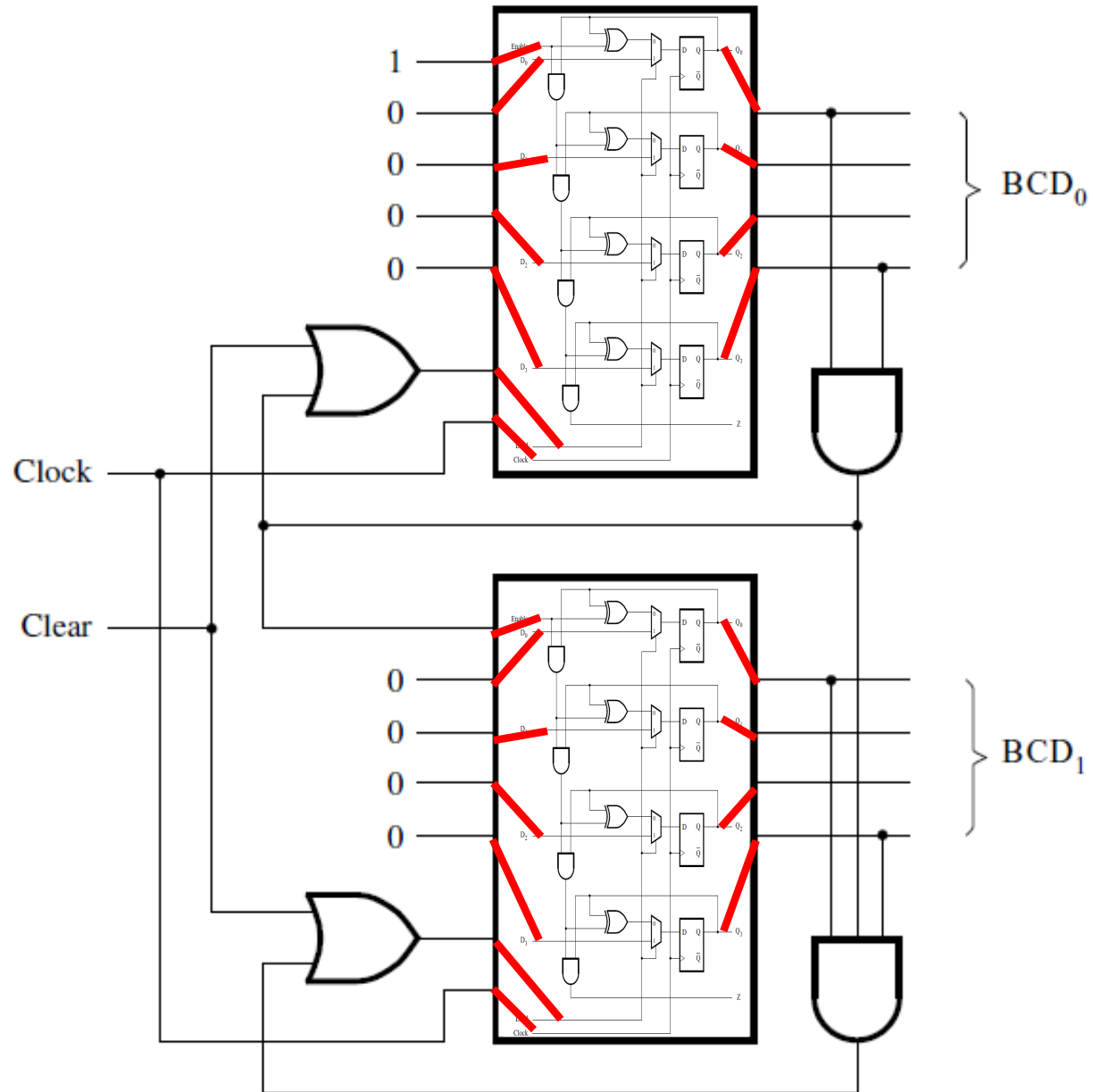
[ Figure 5.24 from the textbook ]

# A two-digit BCD counter

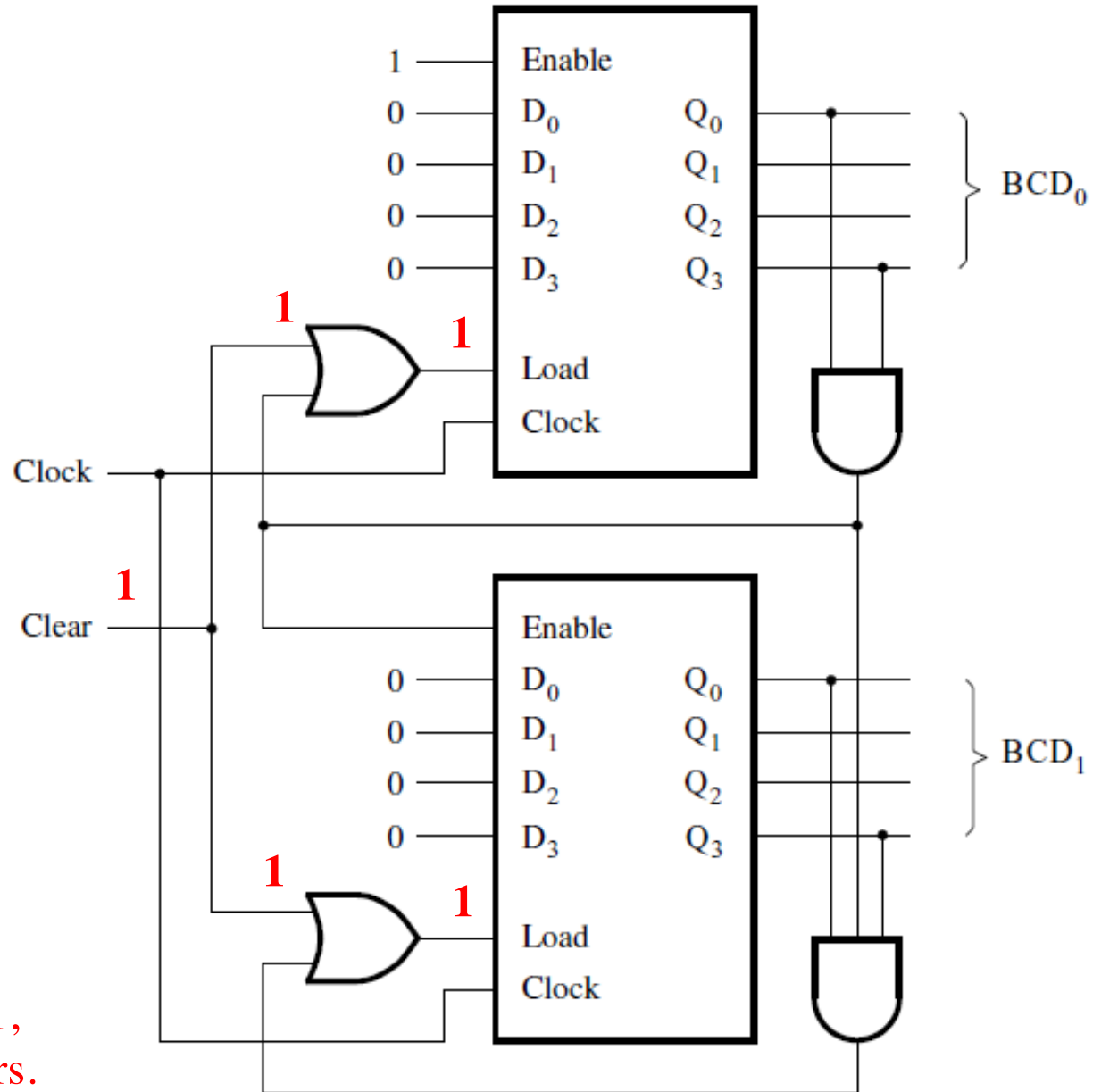




# A two-digit BCD counter

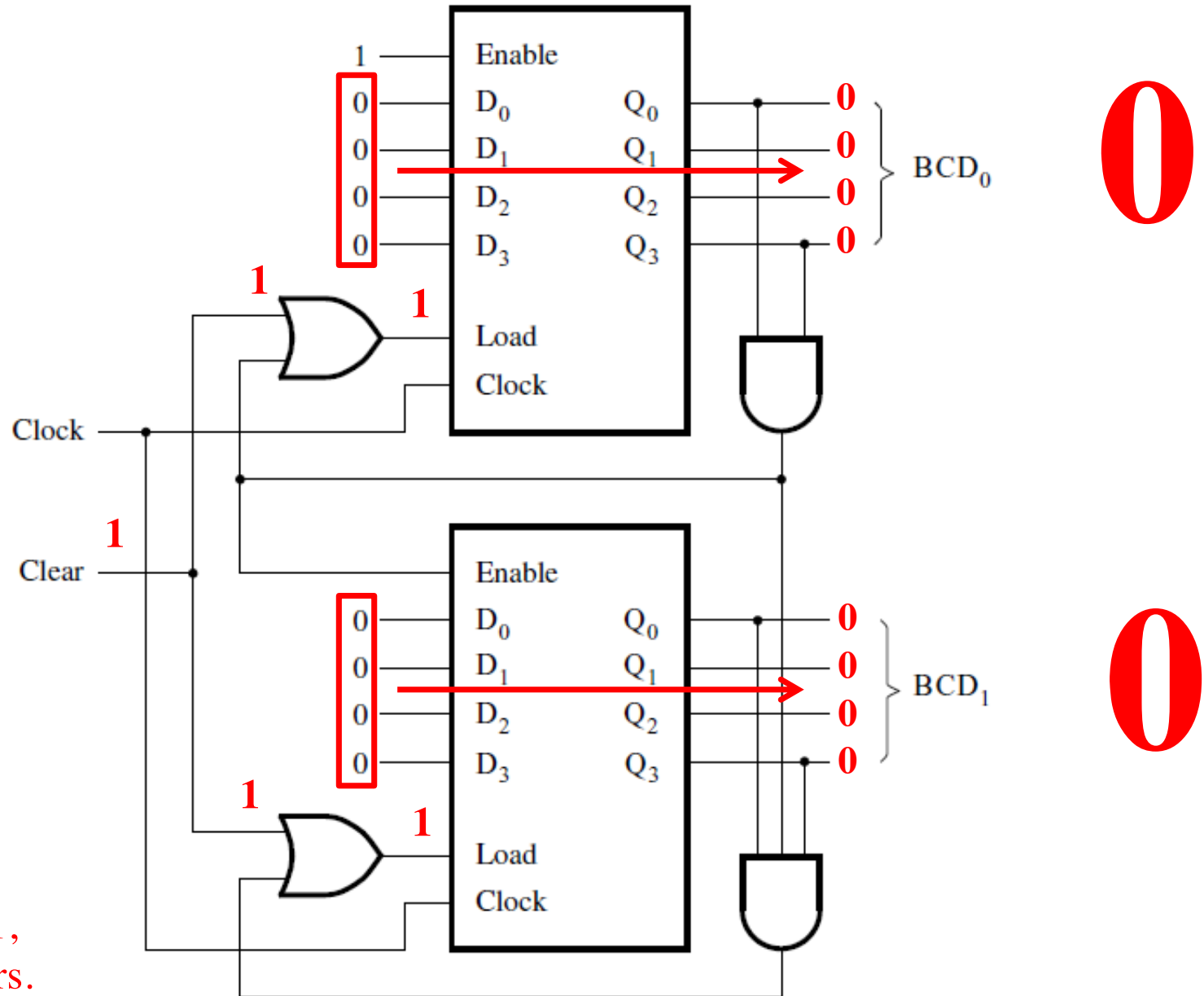


# Zeroing the BCD counter



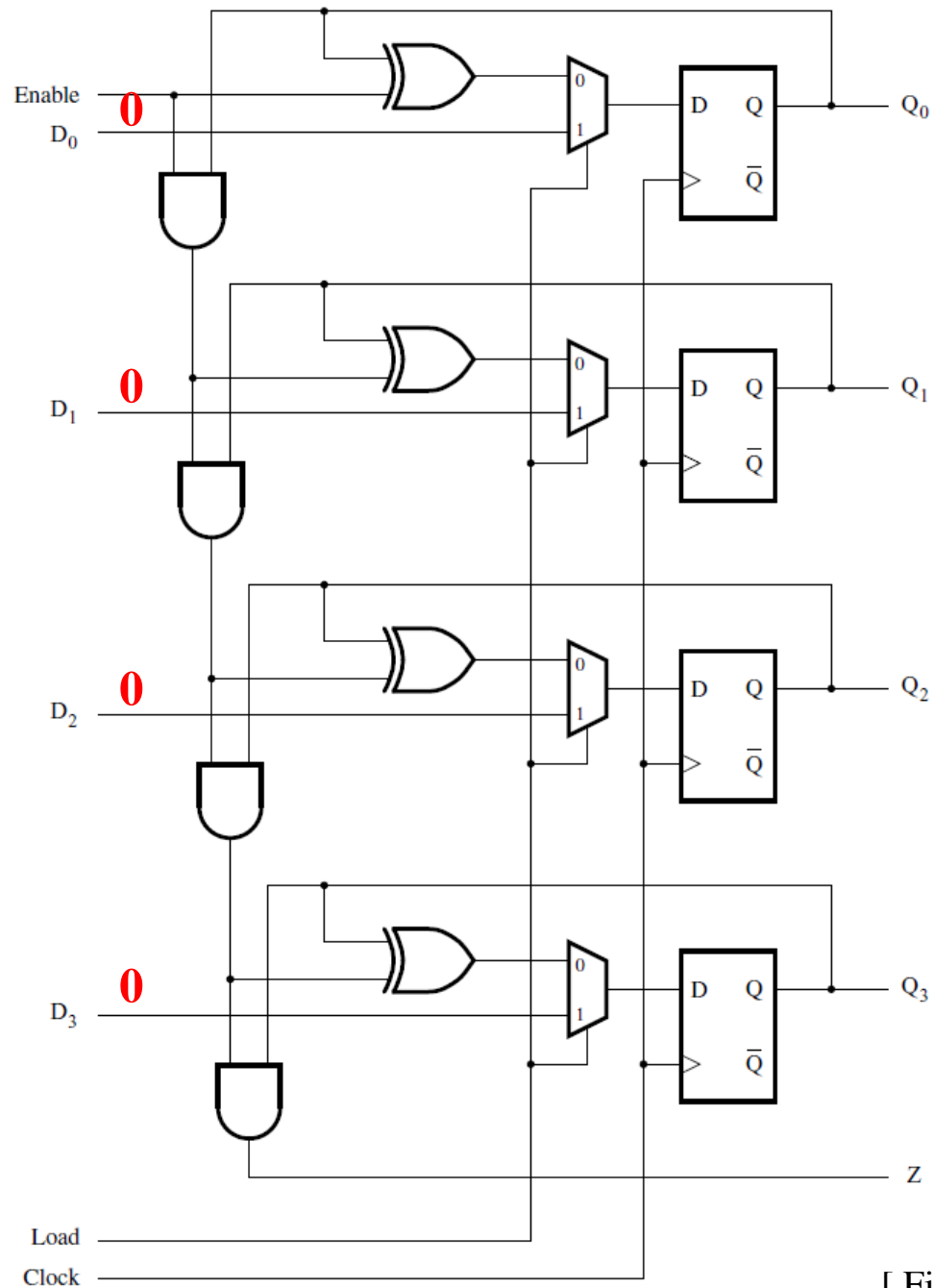
Setting "Clear" to 1,  
zeroes both counters.

# Zeroing the BCD counter



Setting "Clear" to 1,  
zeroes both counters.

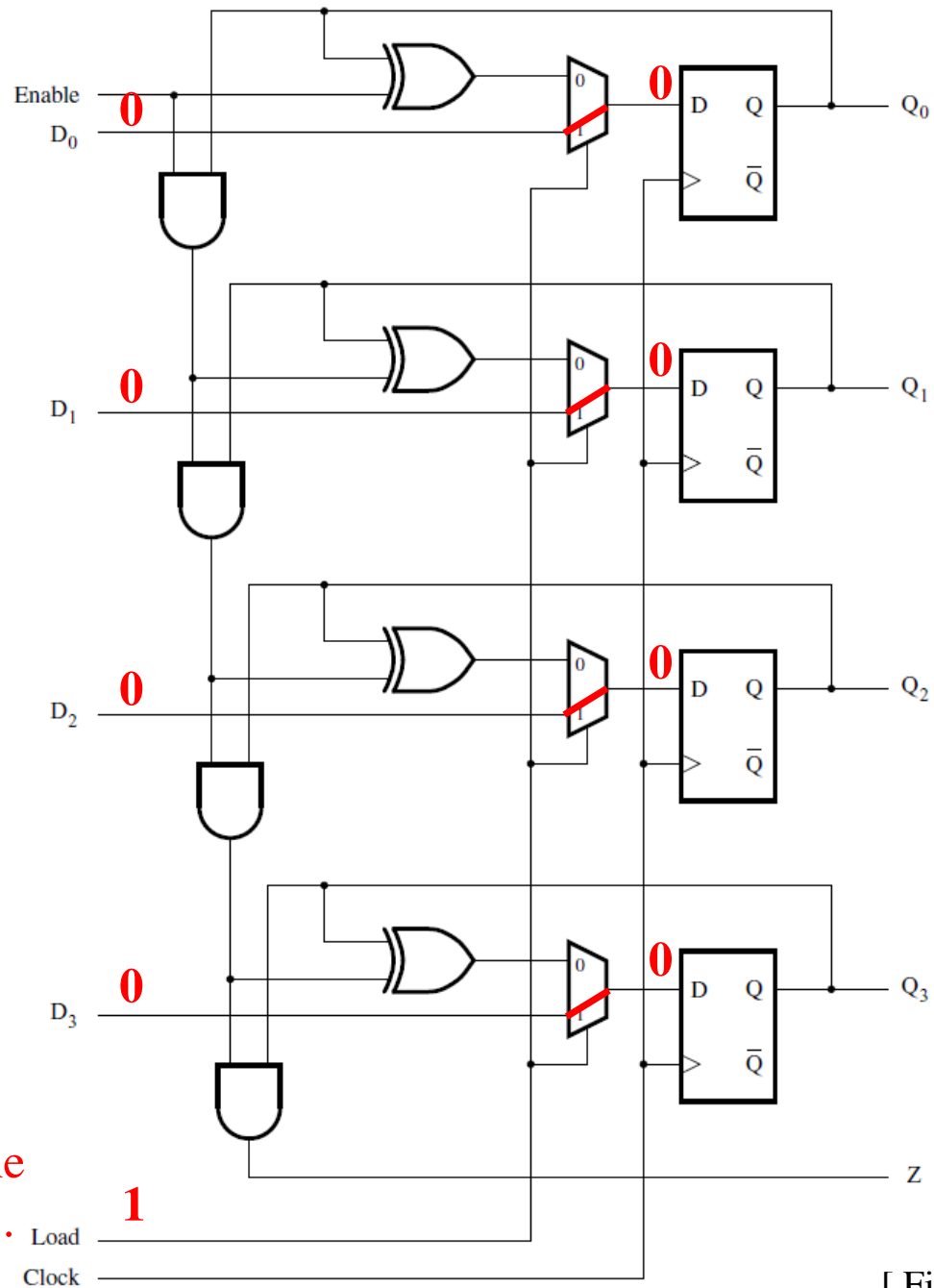
# How to zero a counter



Set all parallel load input lines to zero.

[ Figure 5.24 from the textbook ]

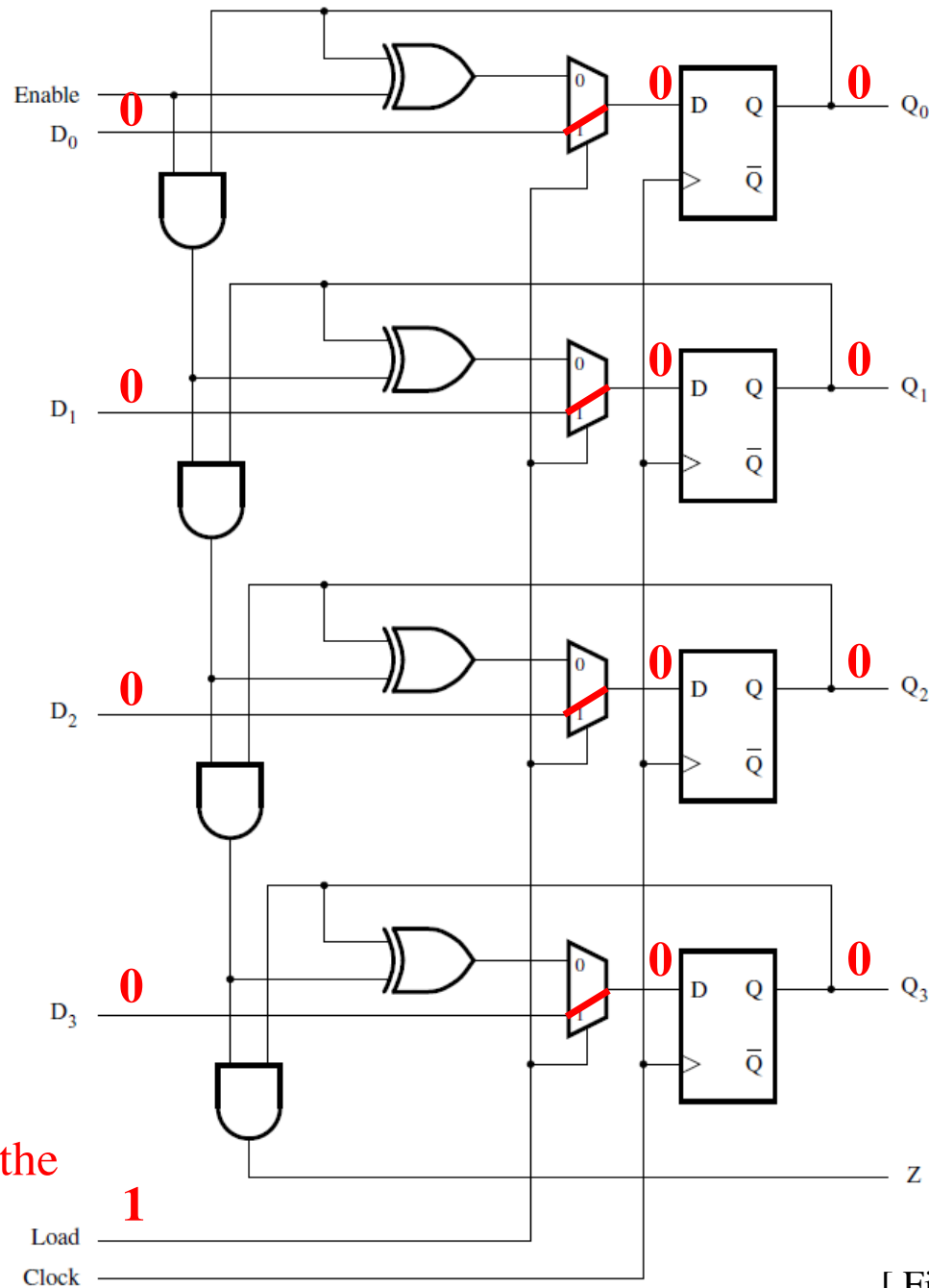
# How to zero a counter



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

[ Figure 5.24 from the textbook ]

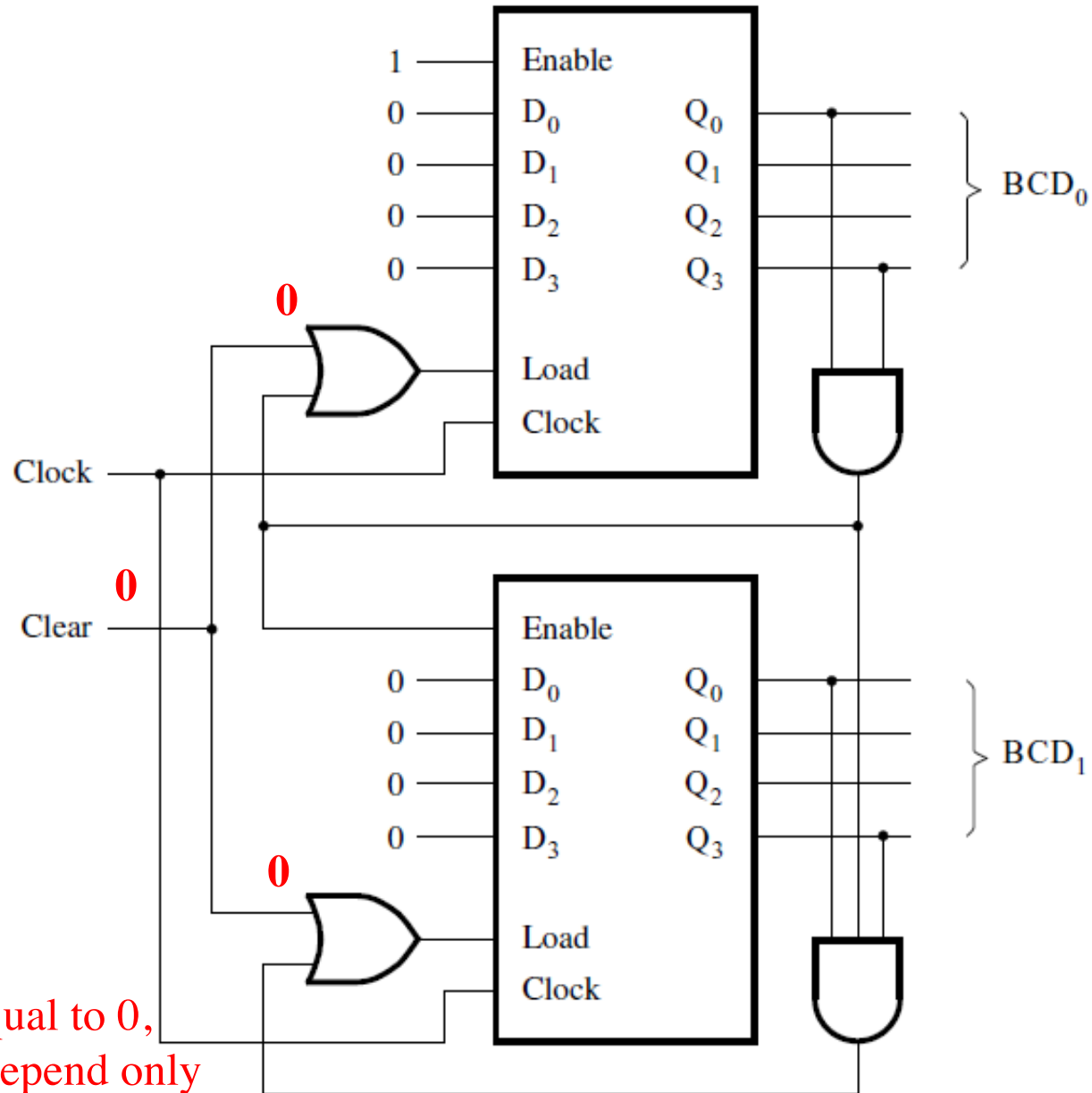
# How to zero a counter



When the positive edge of the clock arrives, all outputs are set to zero together.

[ Figure 5.24 from the textbook ]

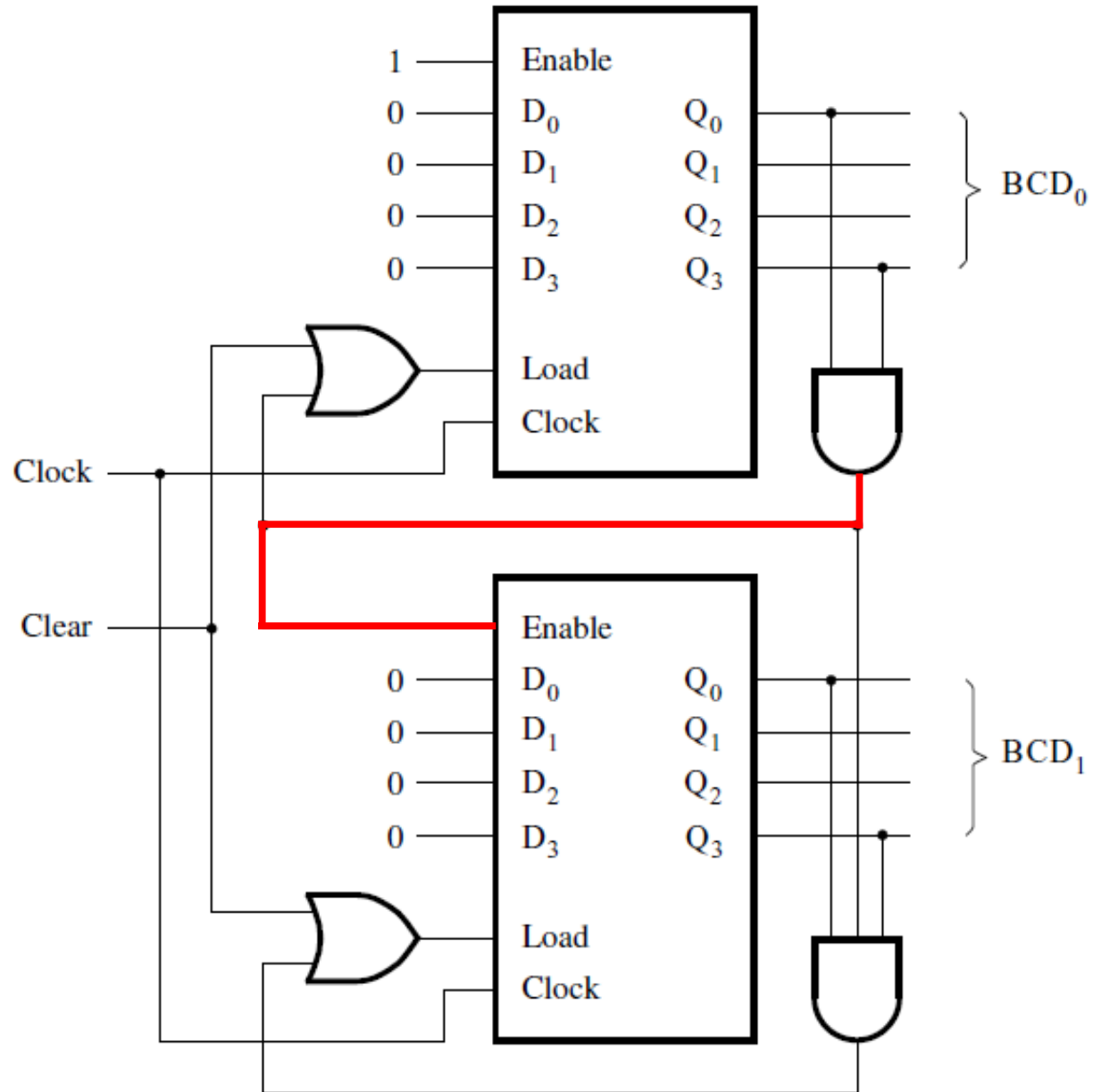
# When Clear = 0



When "Clear" is equal to 0, the two OR gates depend only on the feedback connections.

[ Figure 5.27 from the textbook ]

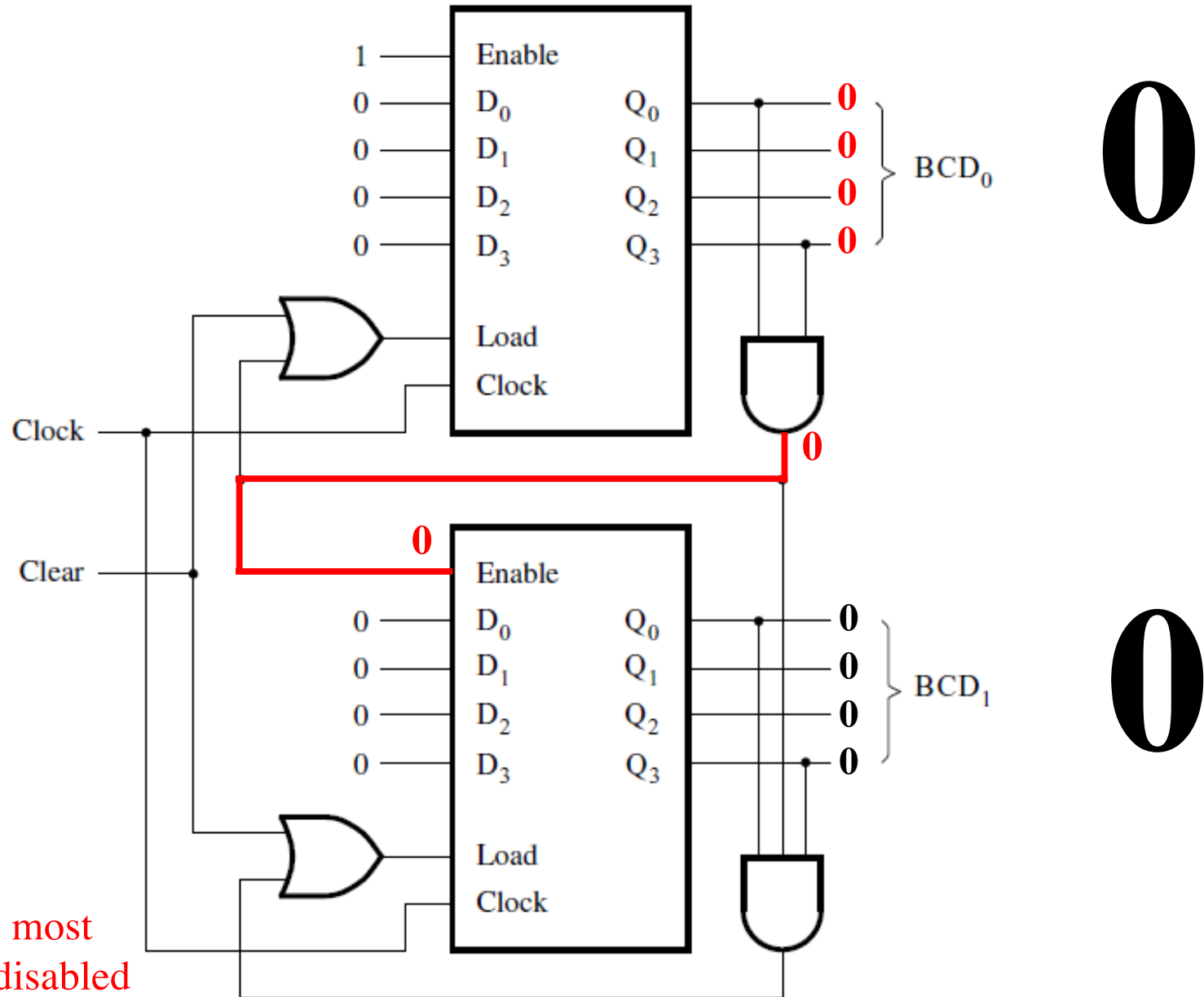
# Enabling the second counter



[ Figure 5.27 from the textbook ]

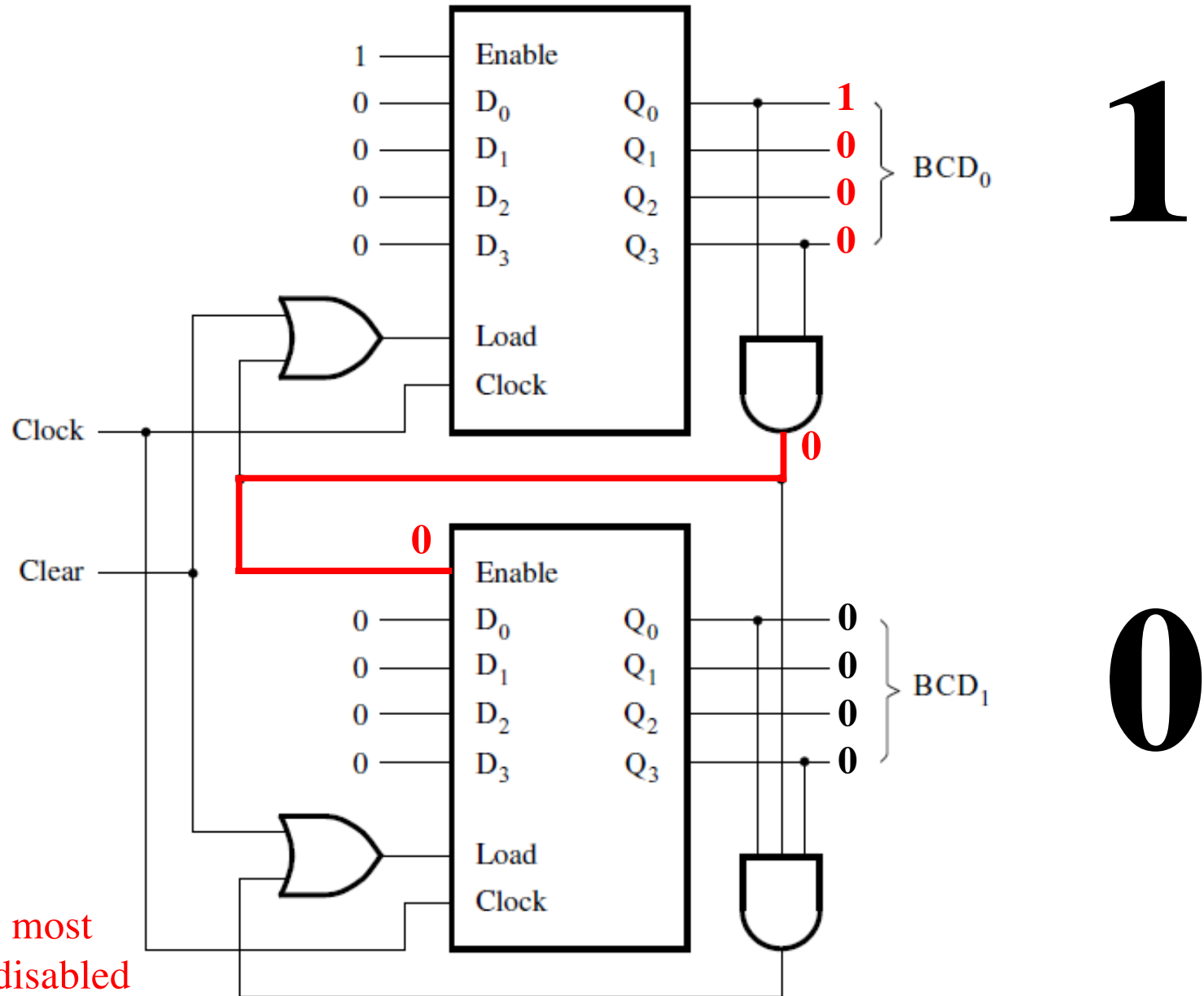


# Enabling the second counter



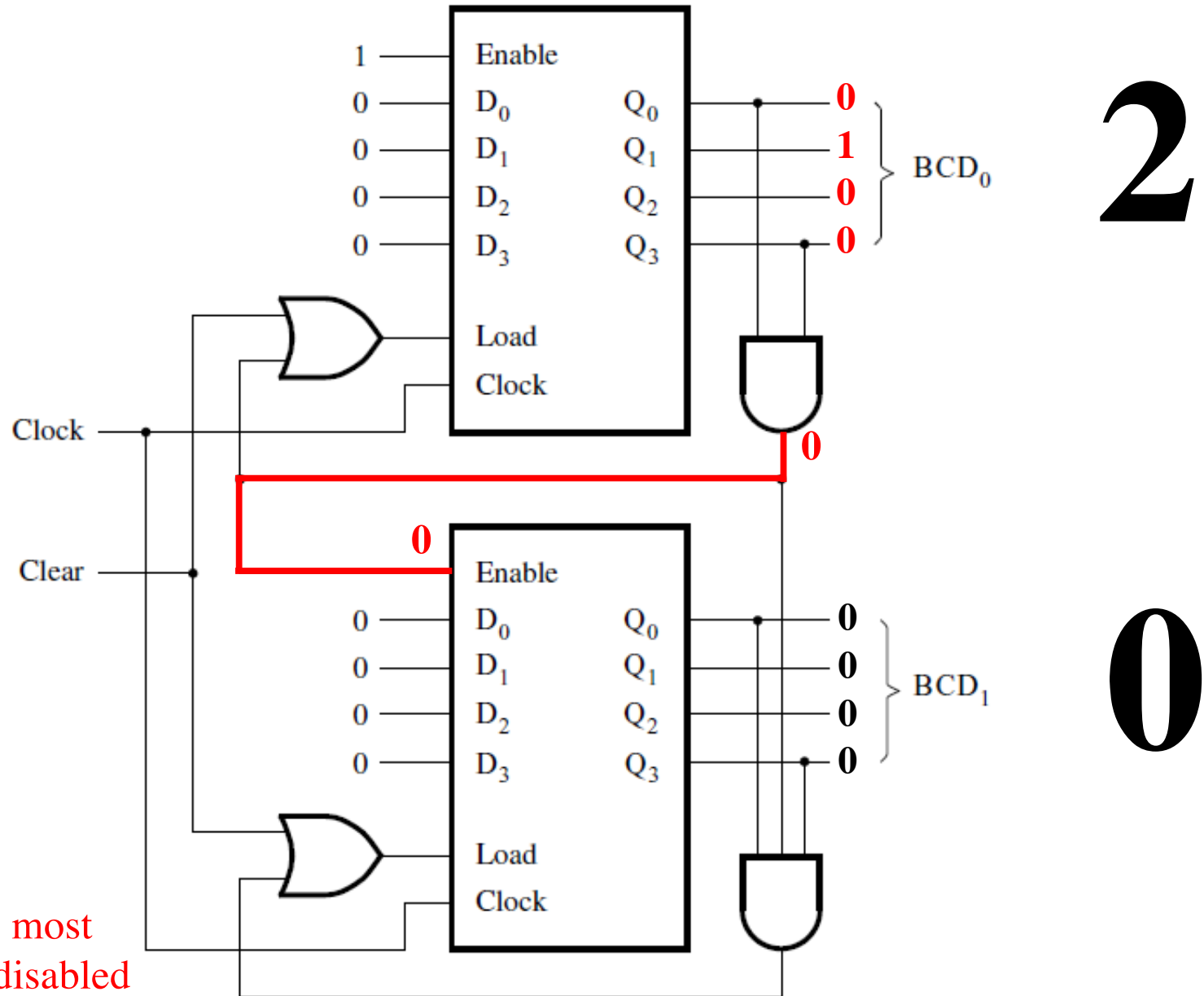
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter



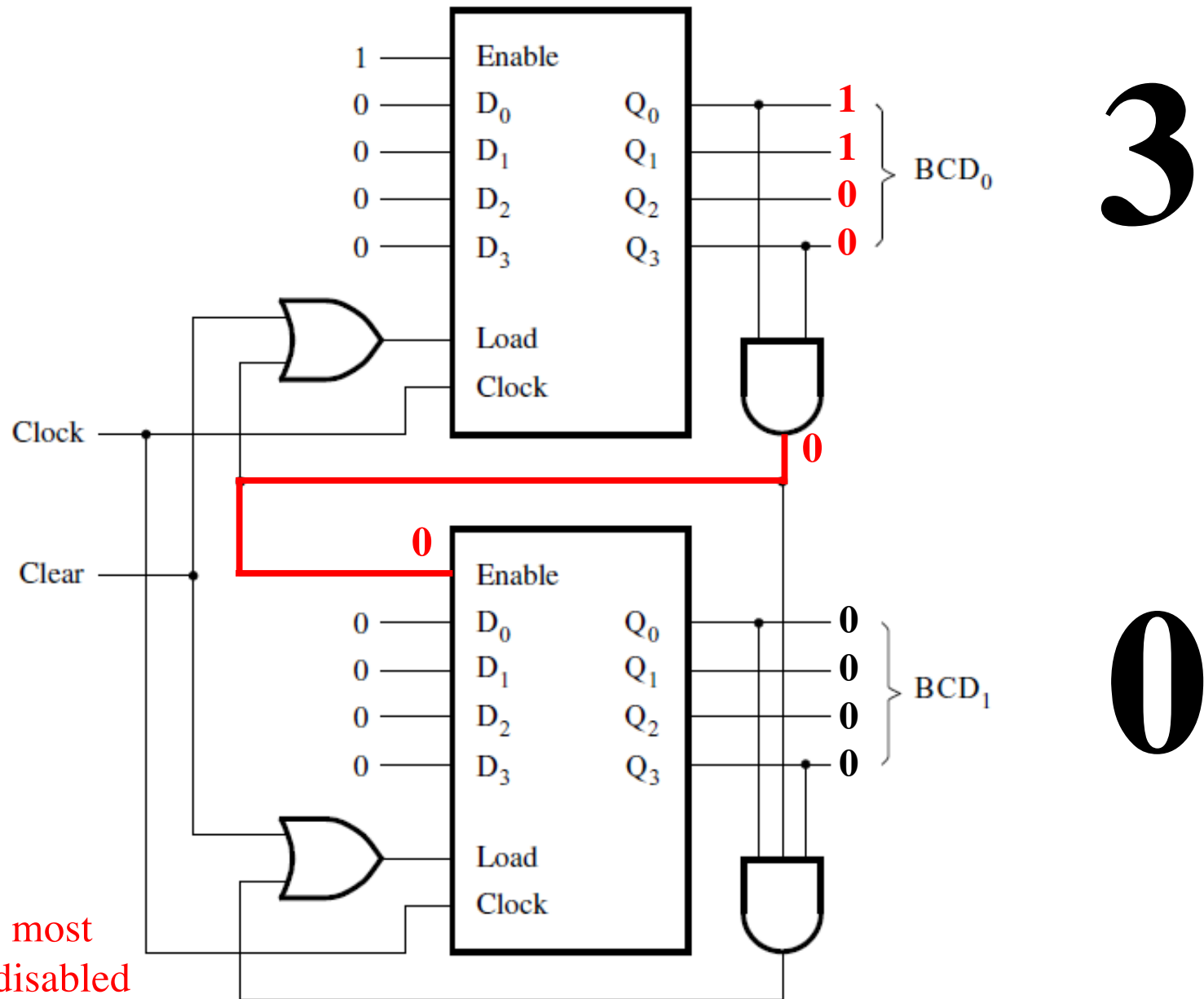
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter



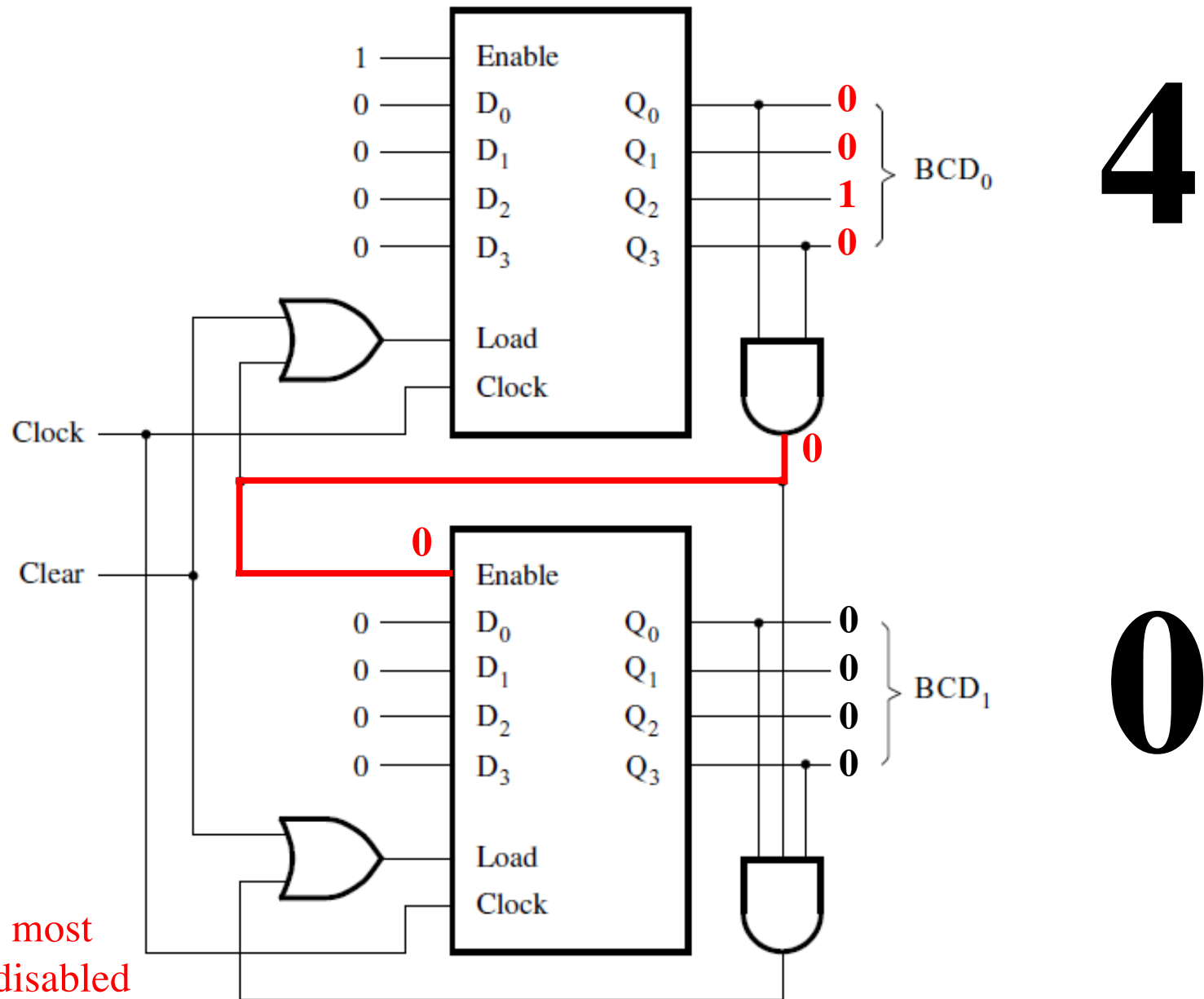
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter



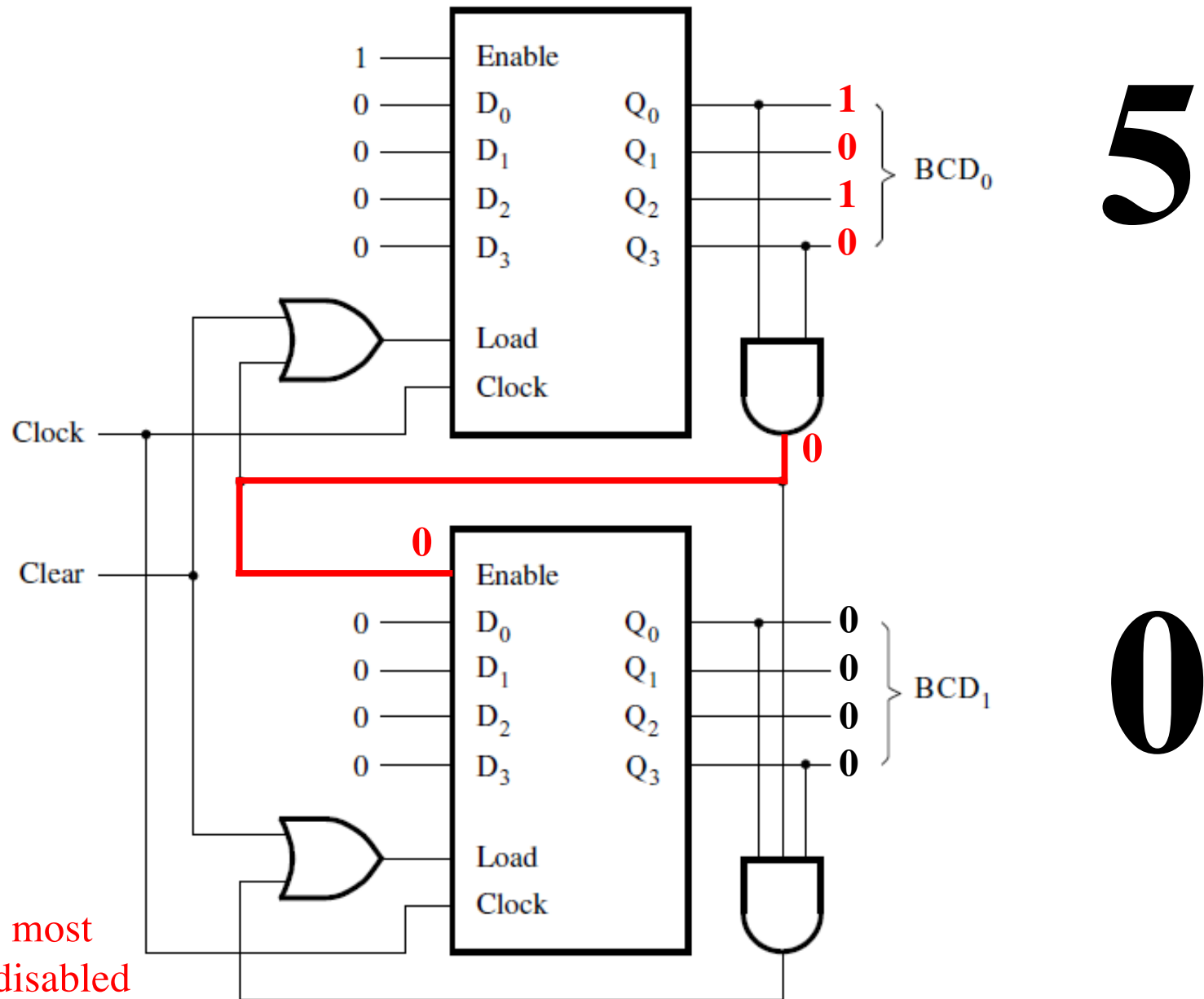
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter



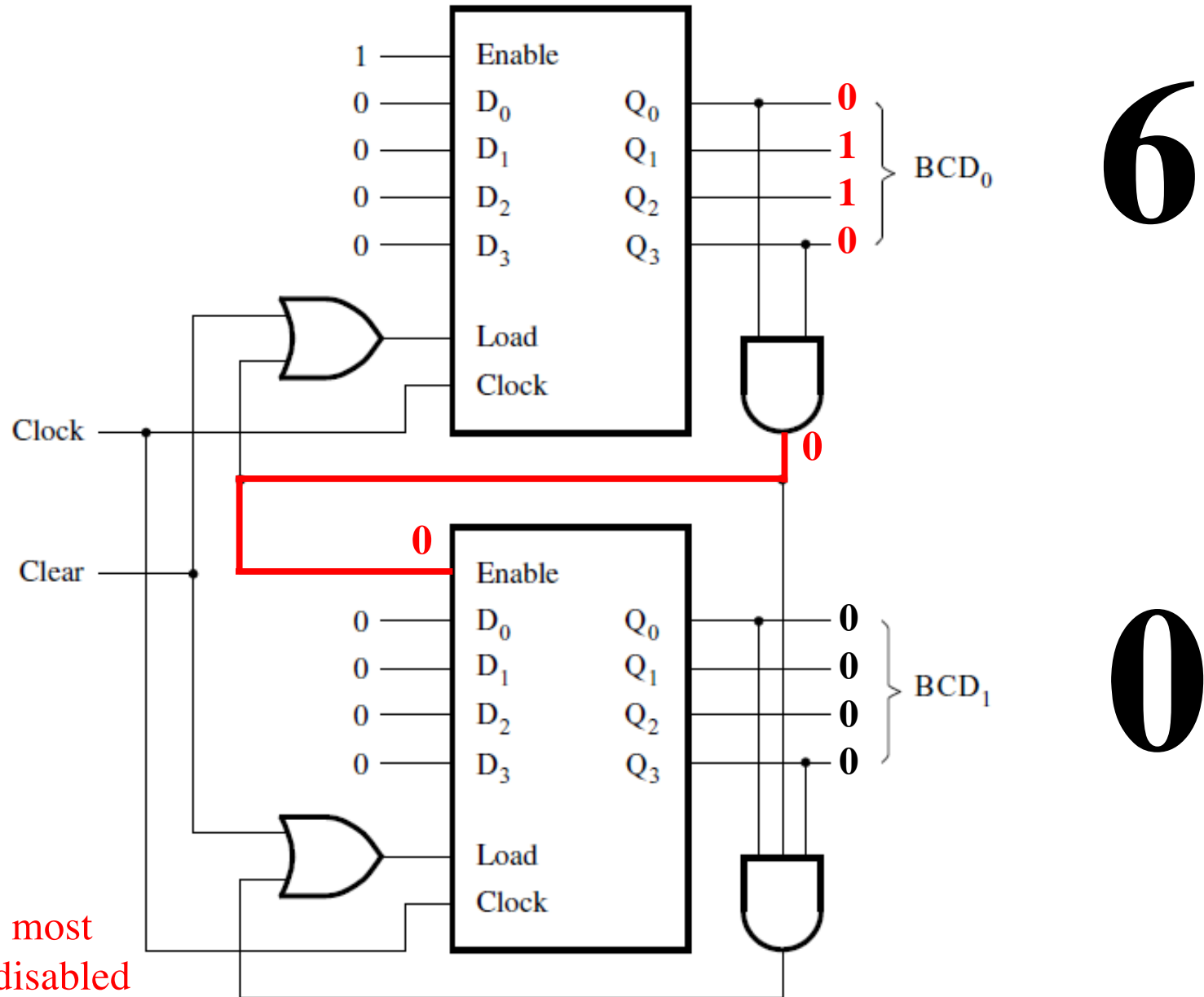
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter



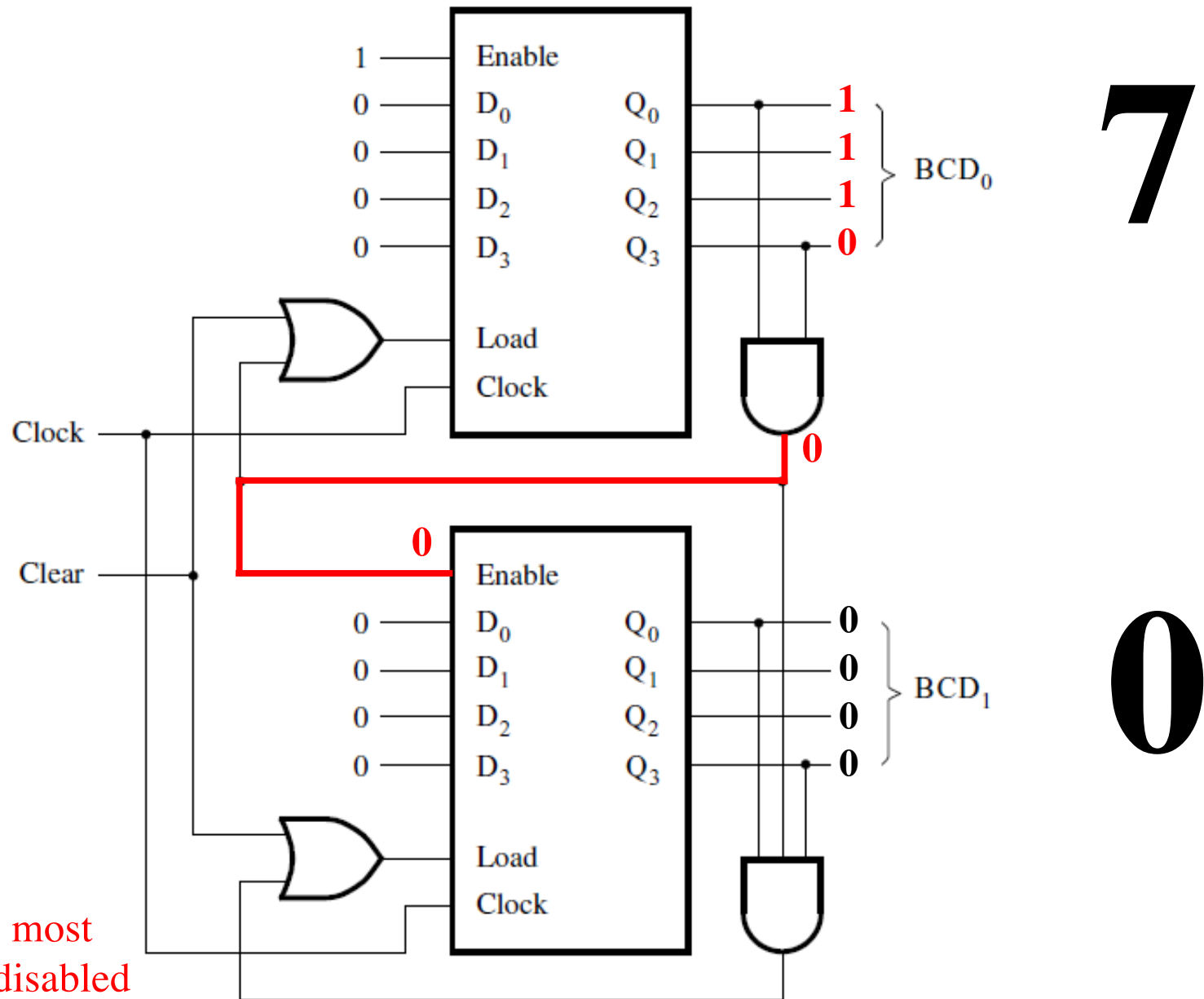
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter



The counter for the most significant digit is disabled most of the time.

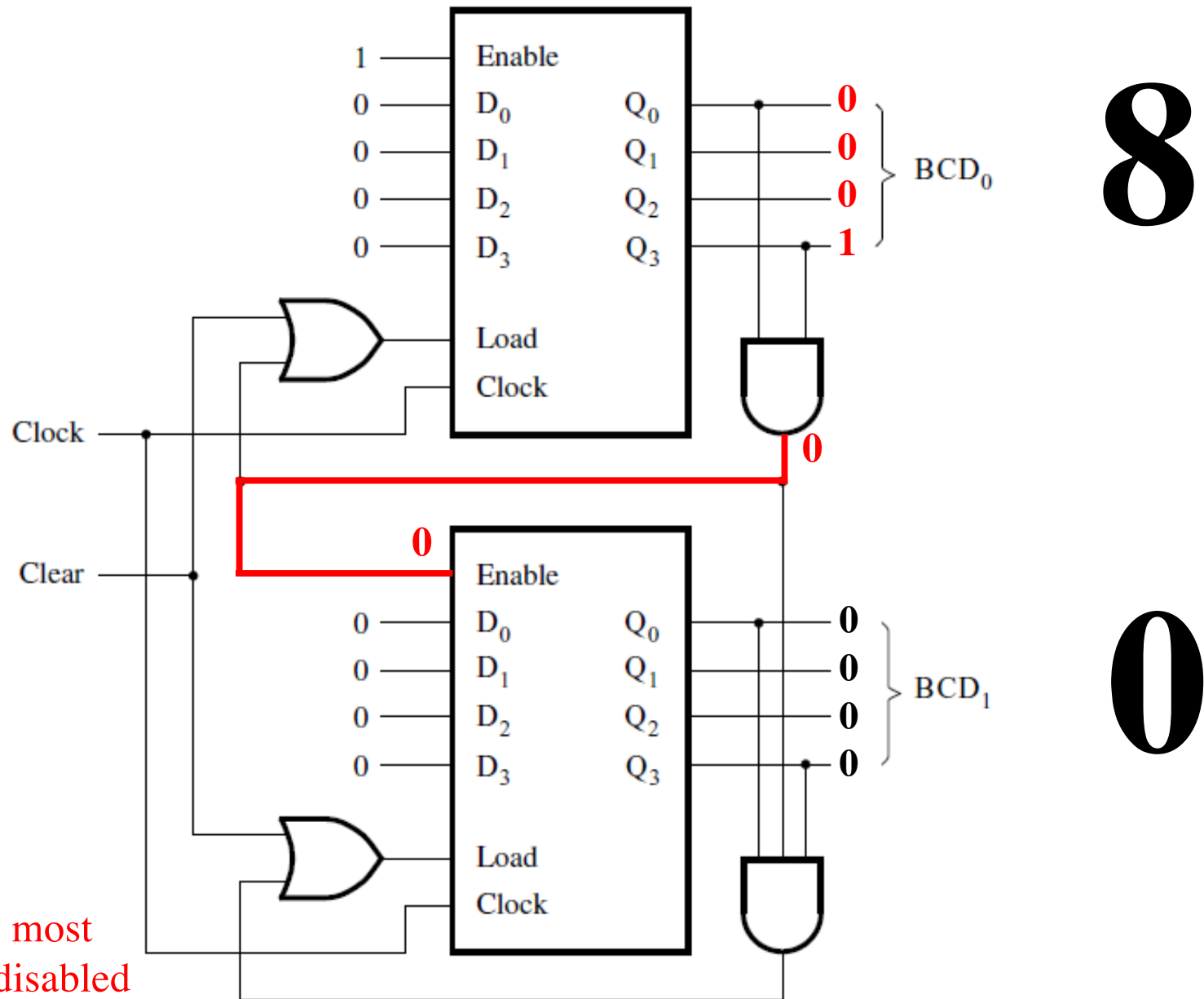
# Enabling the second counter



The counter for the most significant digit is disabled most of the time.

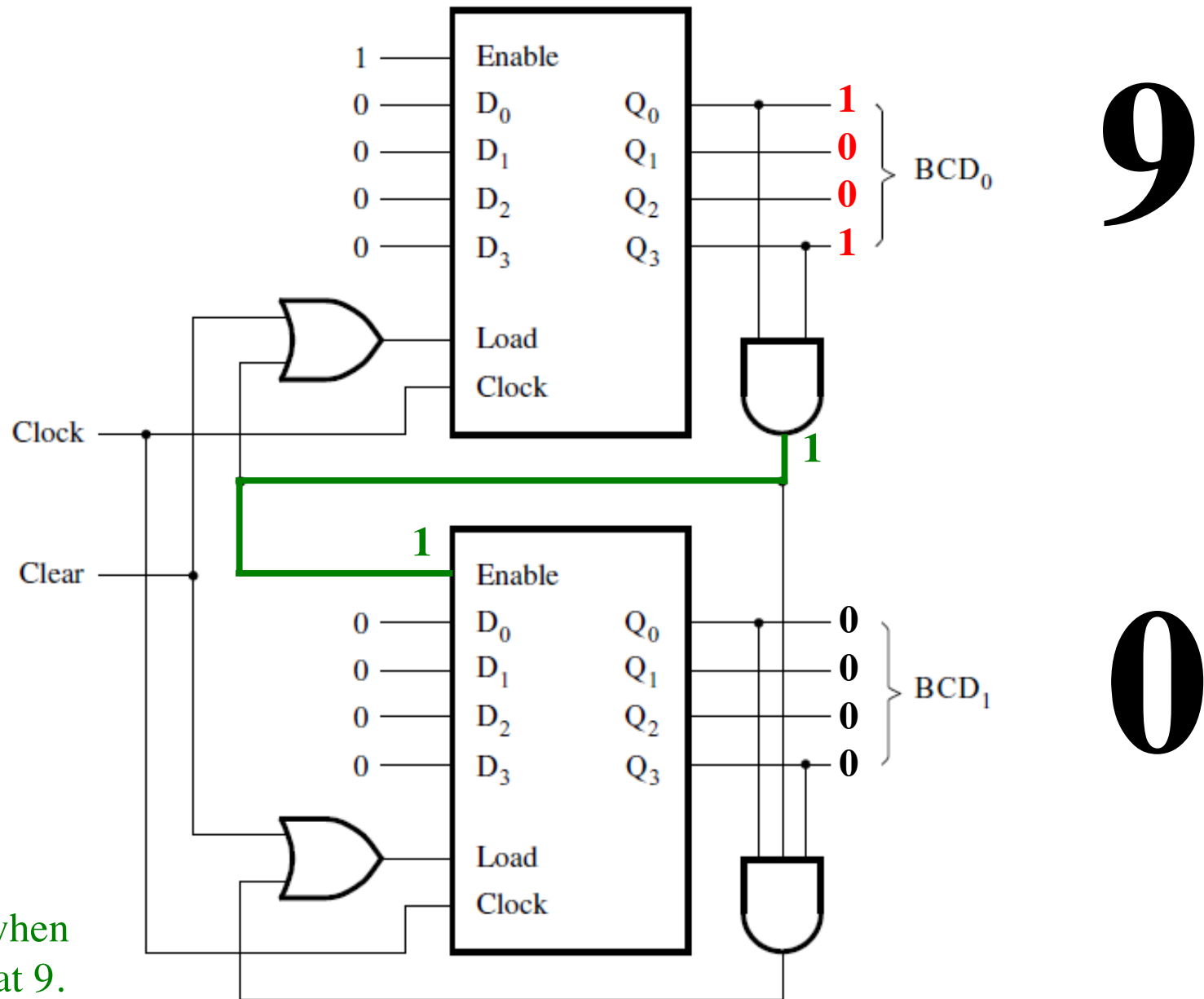


# Enabling the second counter



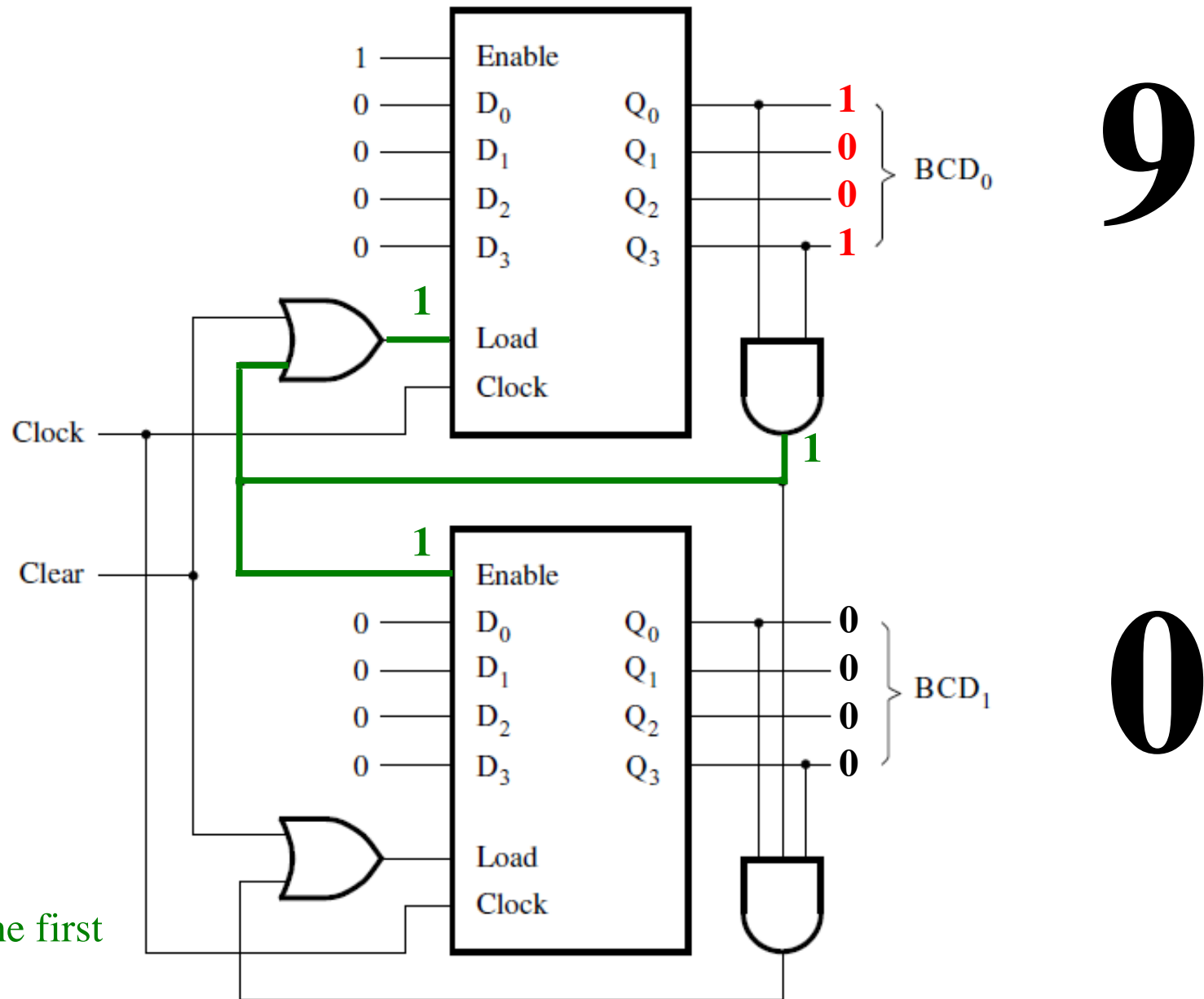
The counter for the most significant digit is disabled most of the time.

# Enabling the second counter

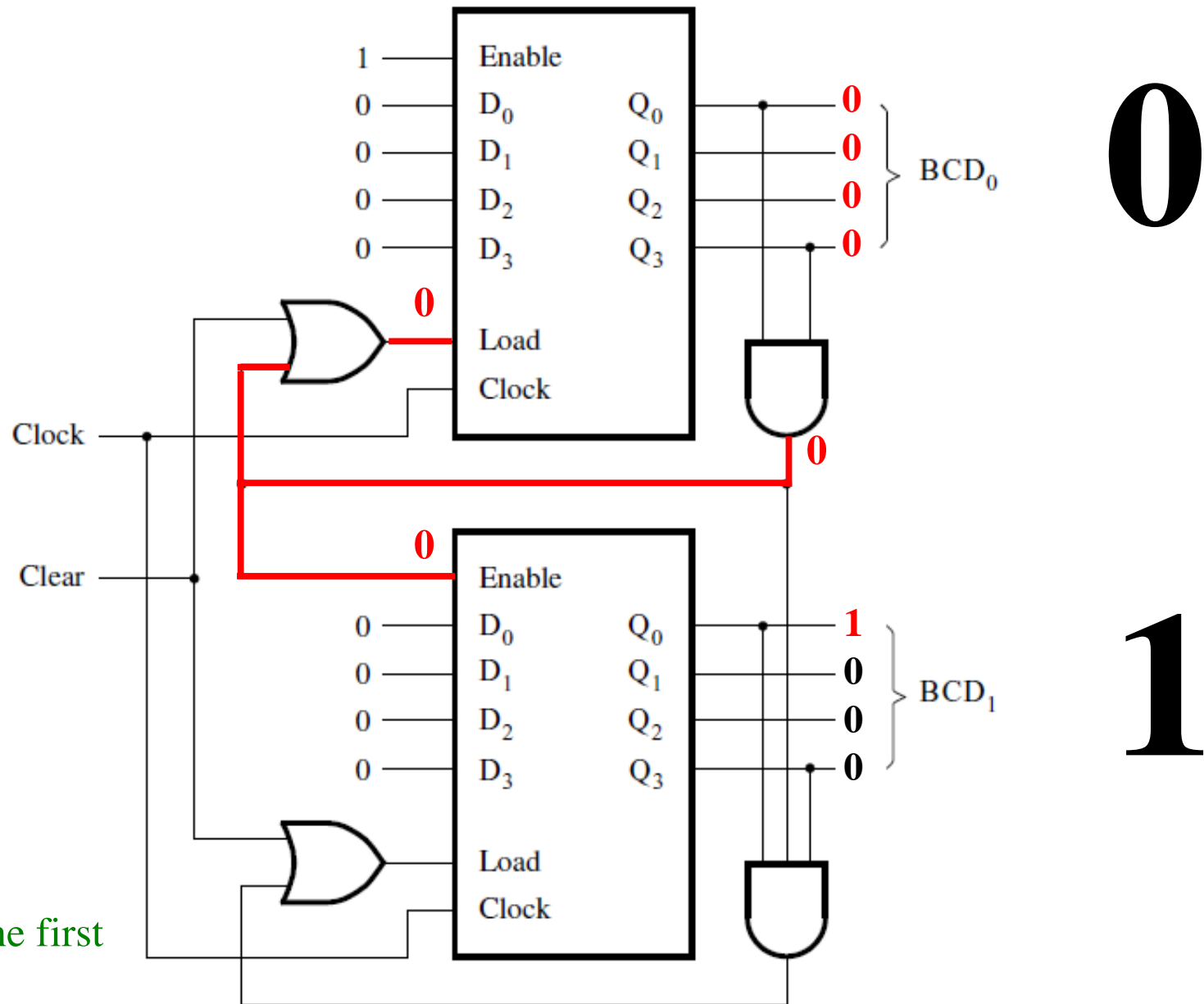


It is enabled only when the first counter is at 9.

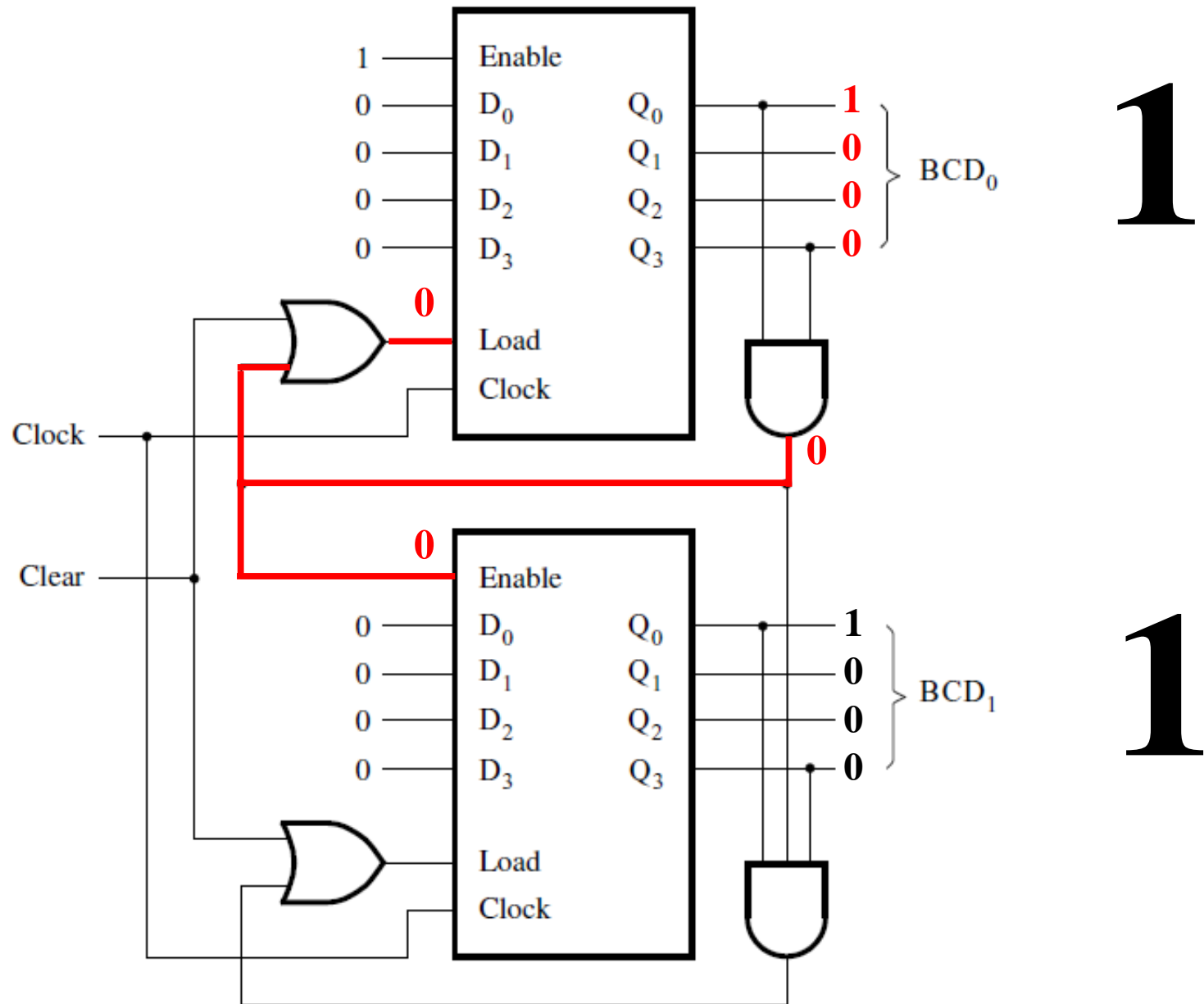
# Enabling the second counter



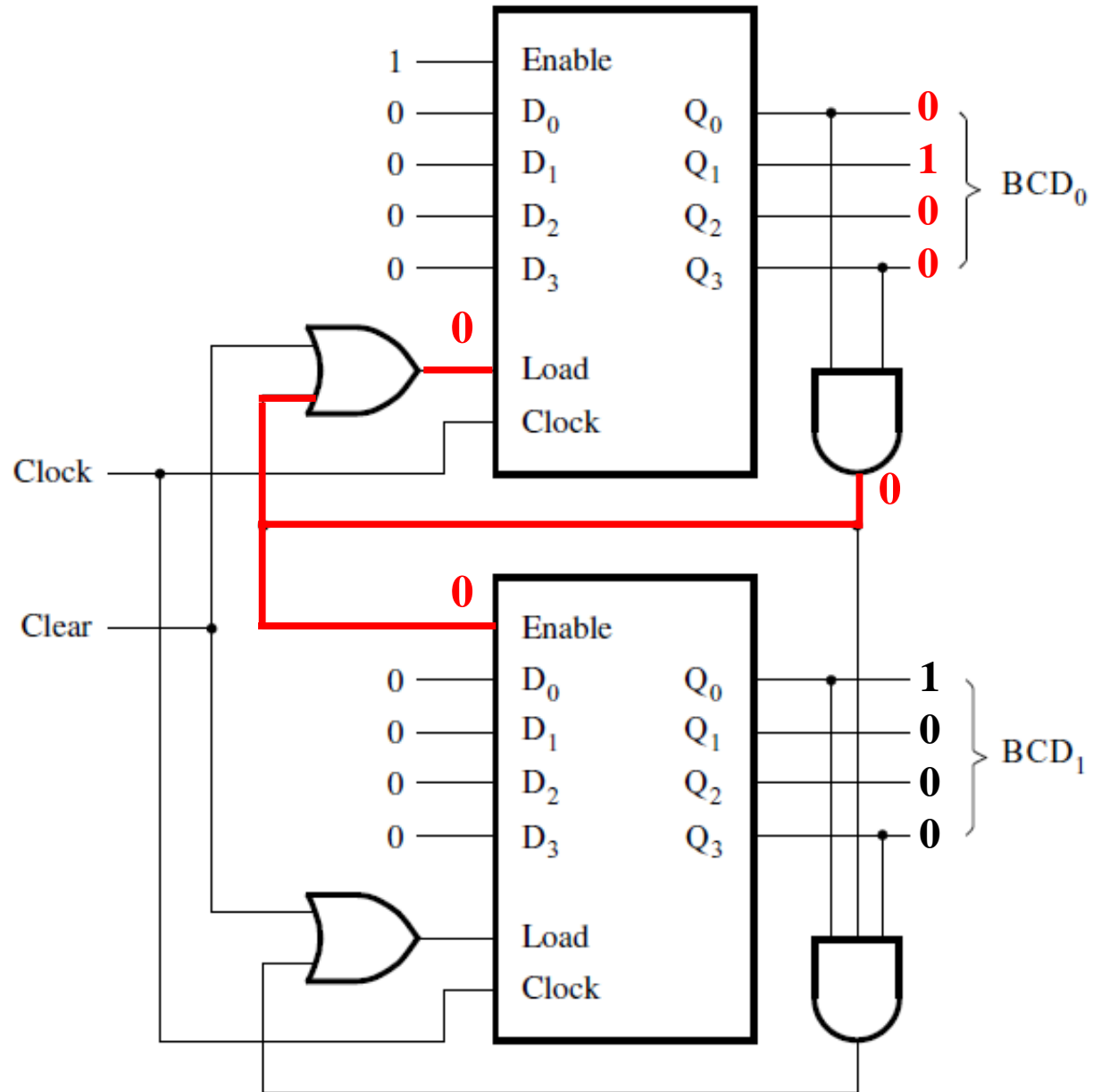
# Enabling the second counter



# Enabling the second counter



# Enabling the second counter

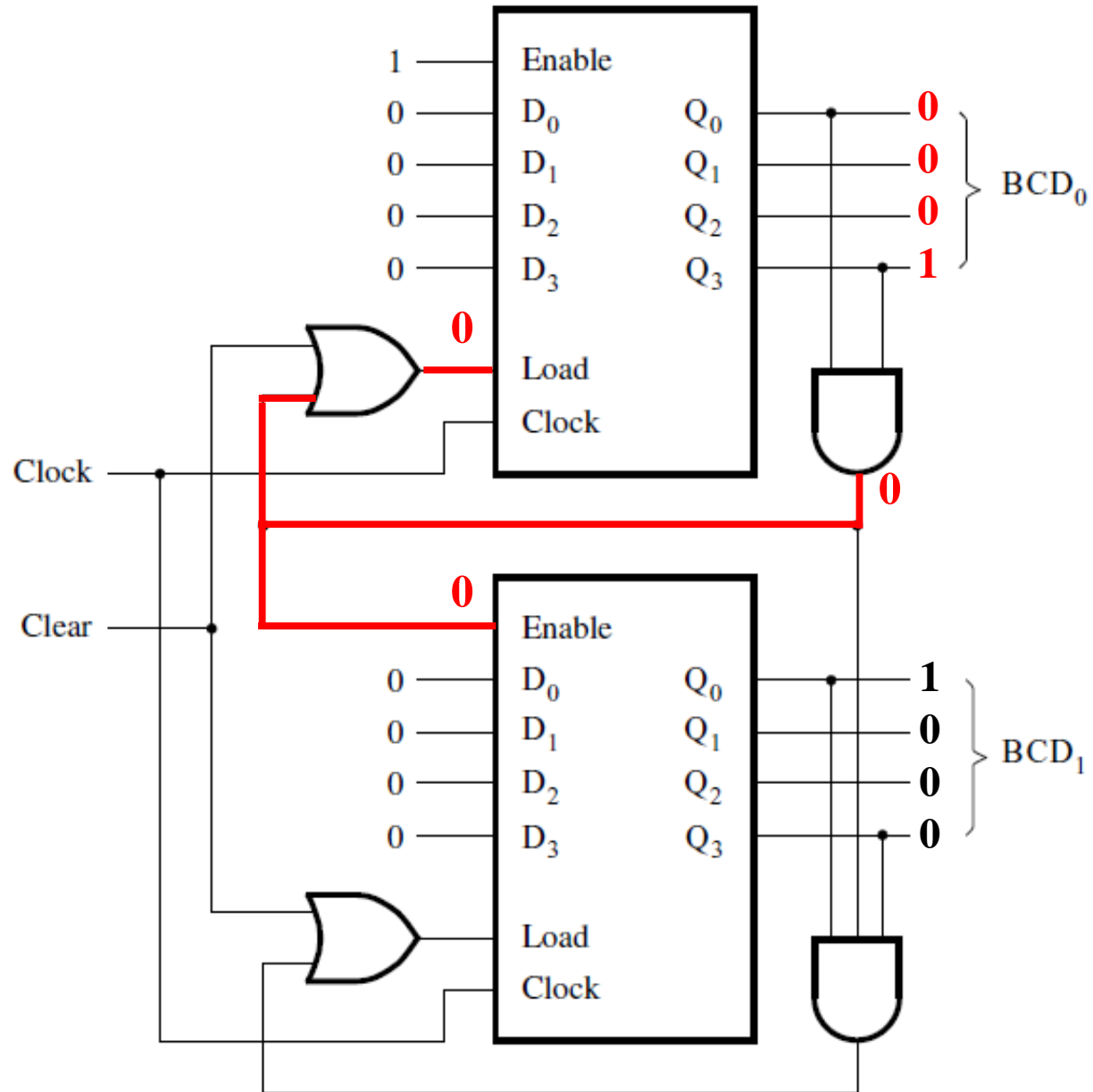


2

1



# Enabling the second counter

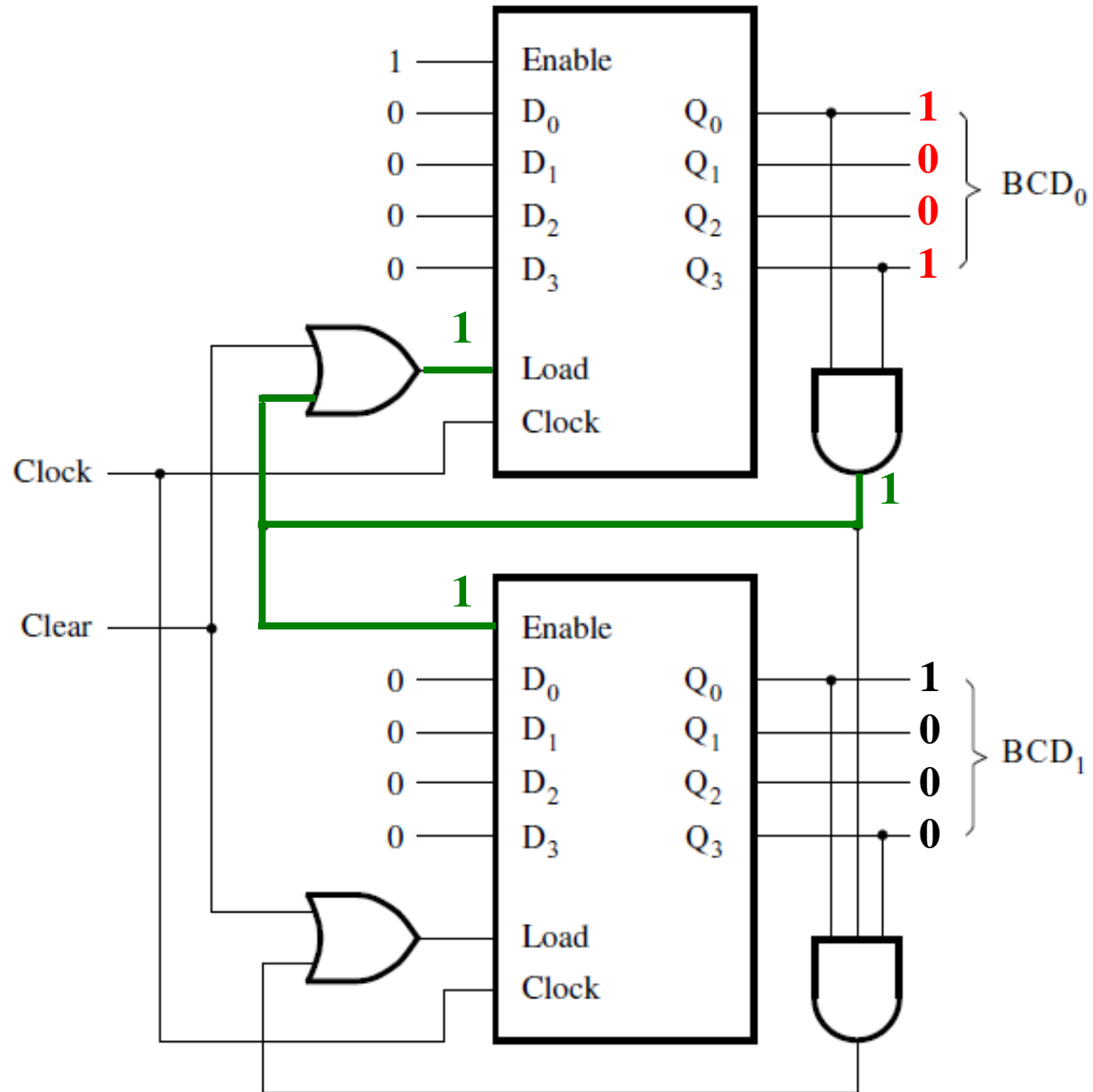


8

1



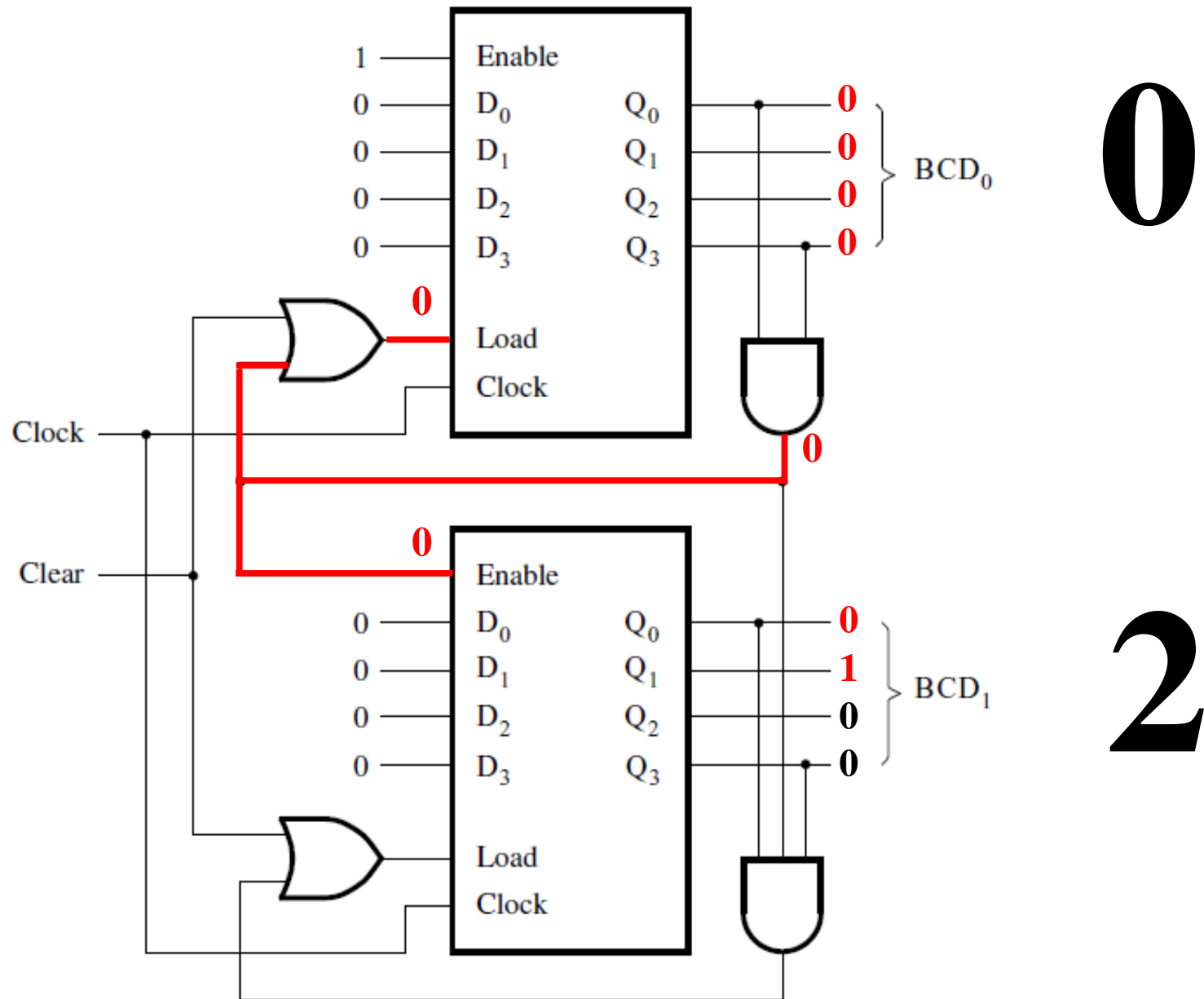
# Enabling the second counter



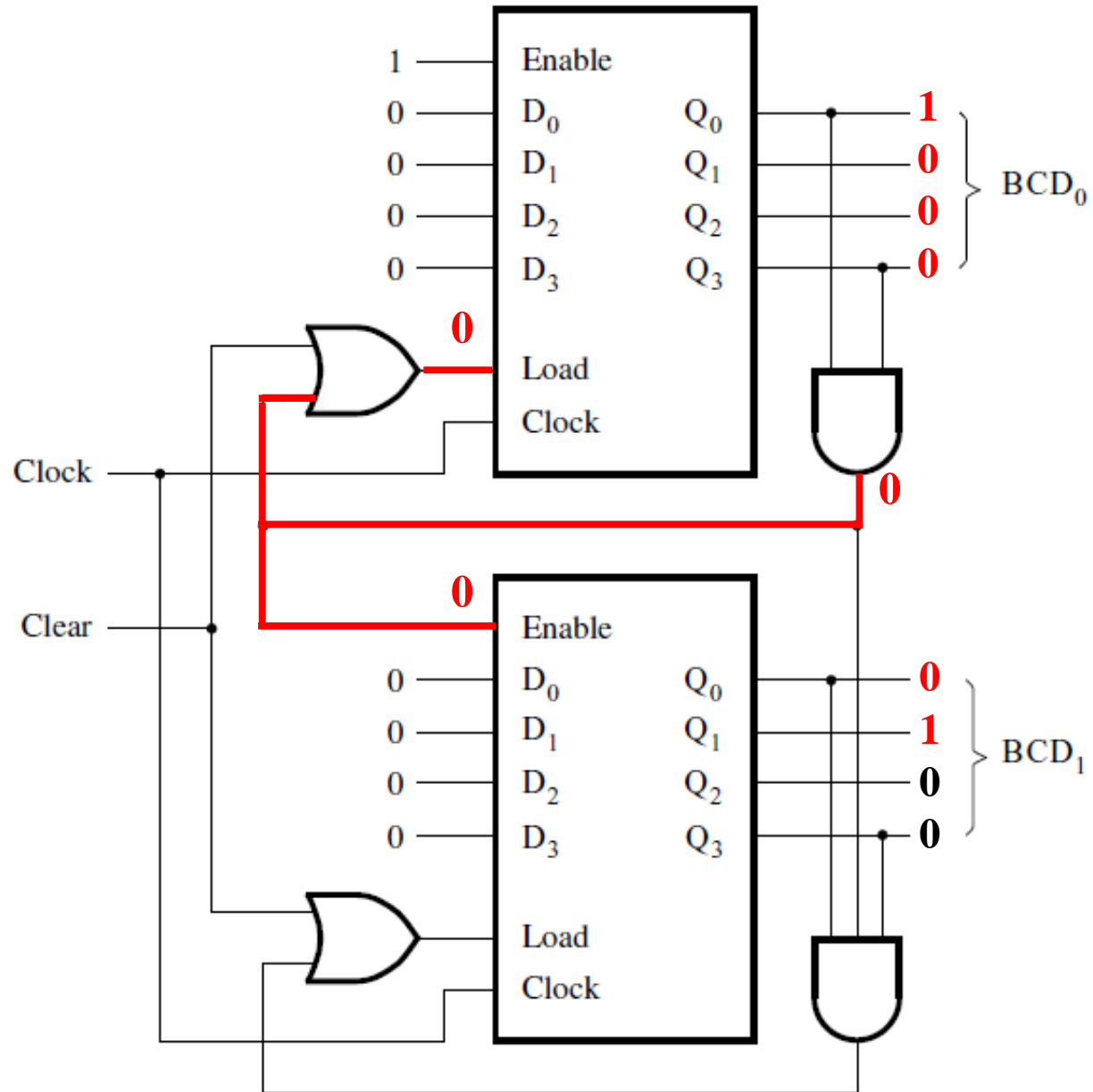
9

1

# Enabling the second counter



# Enabling the second counter

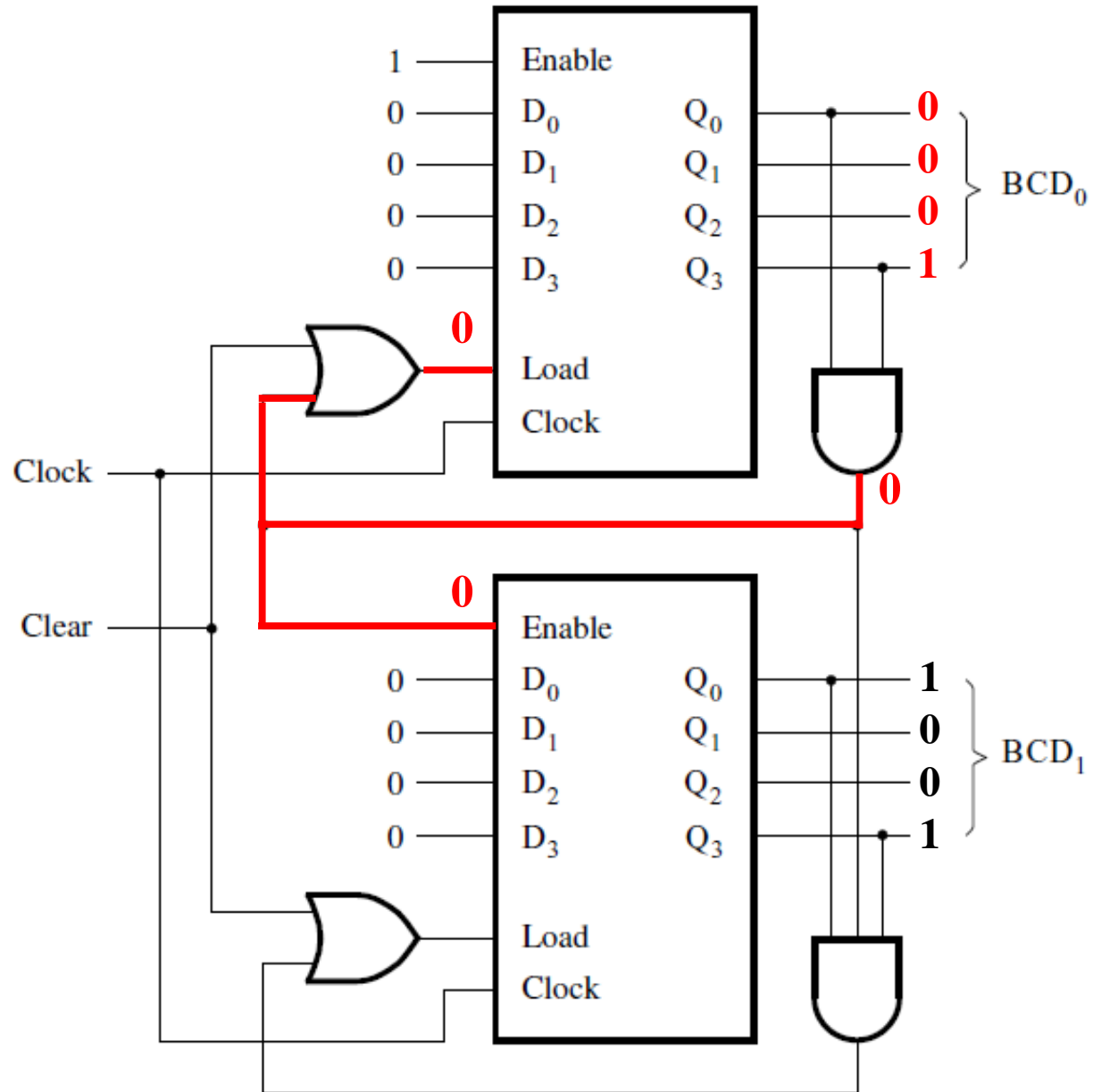


1

2



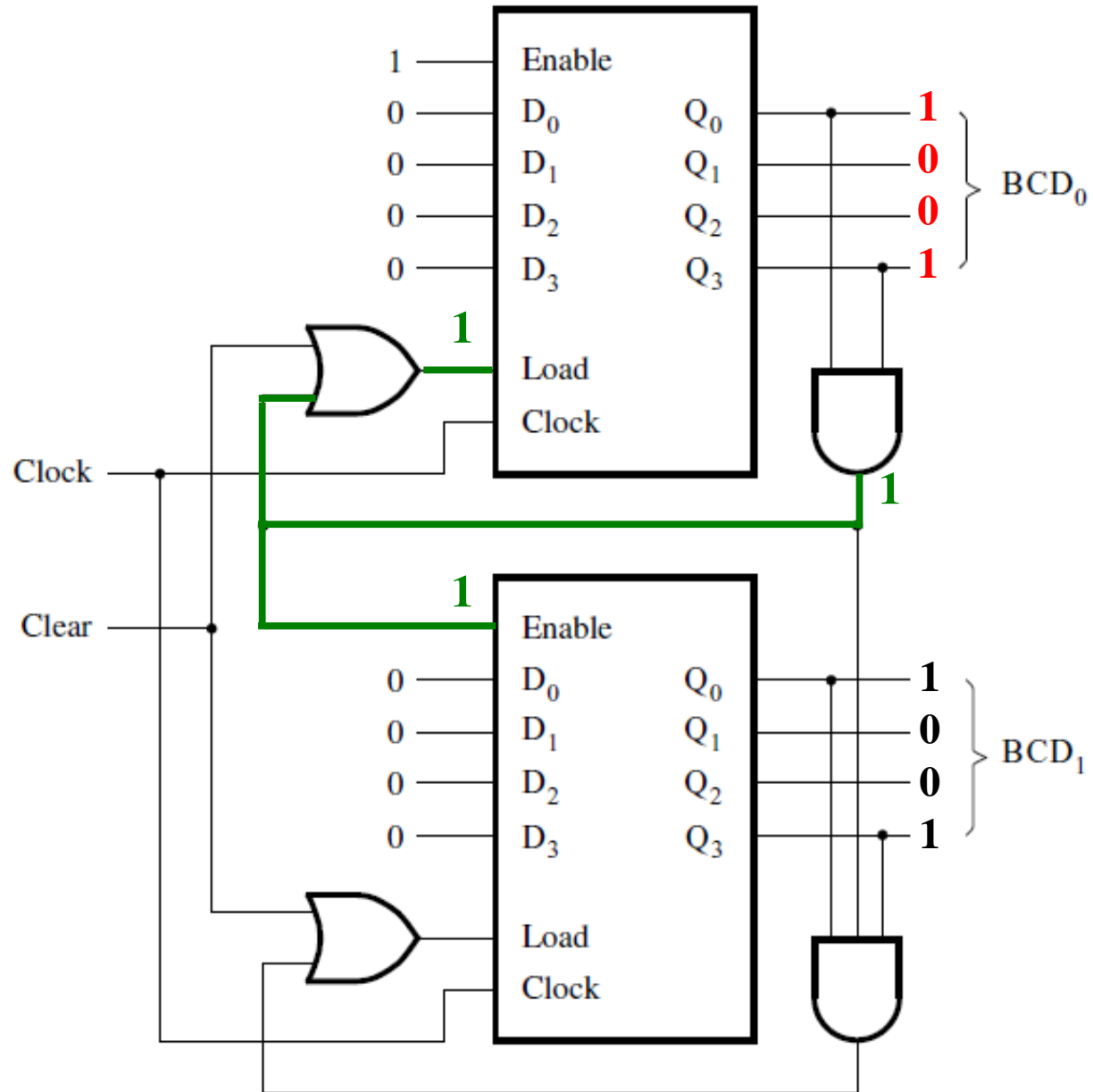
# Enabling the second counter



8

9

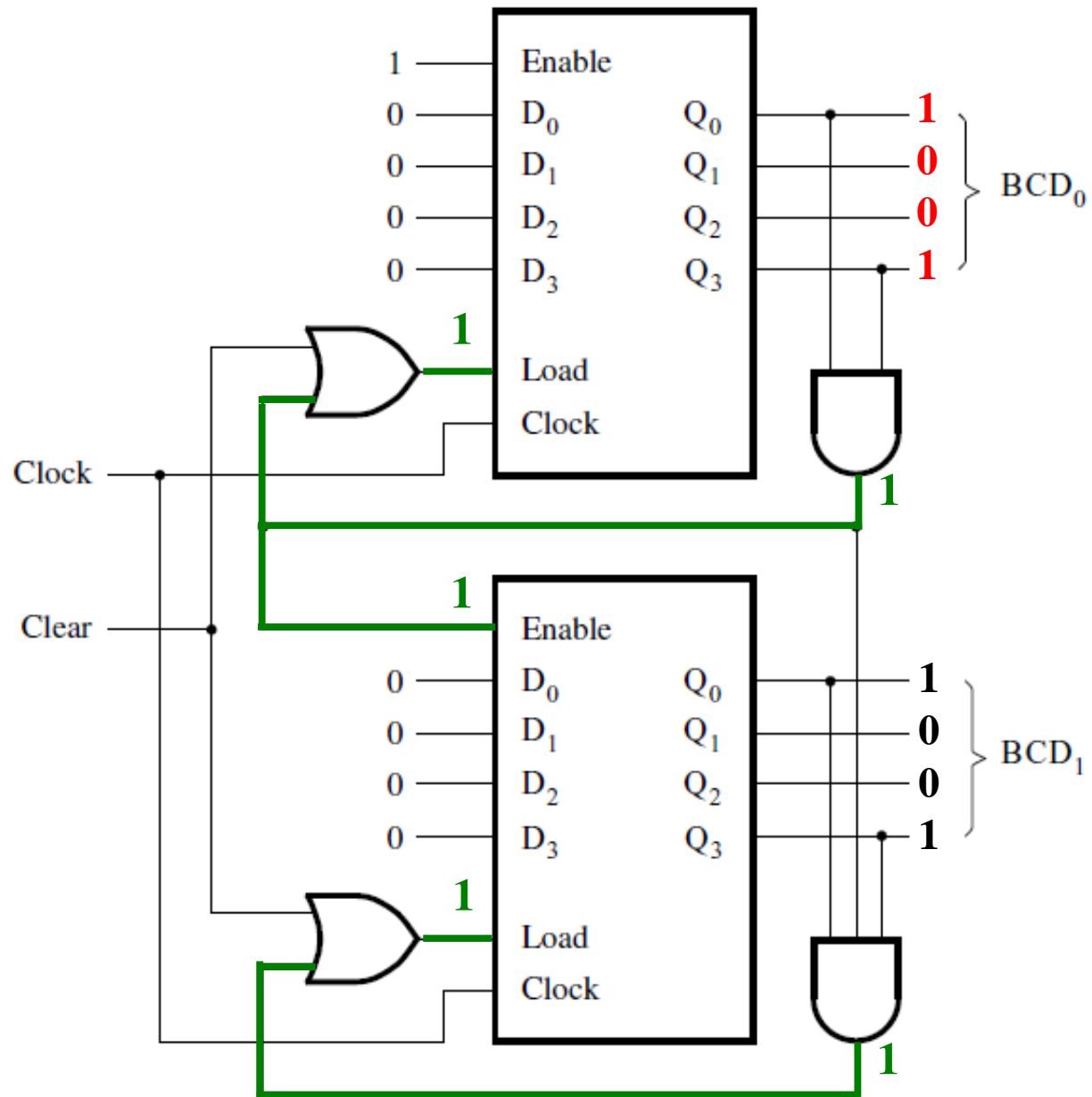
# Enabling the second counter



9

9

# Enabling the second counter



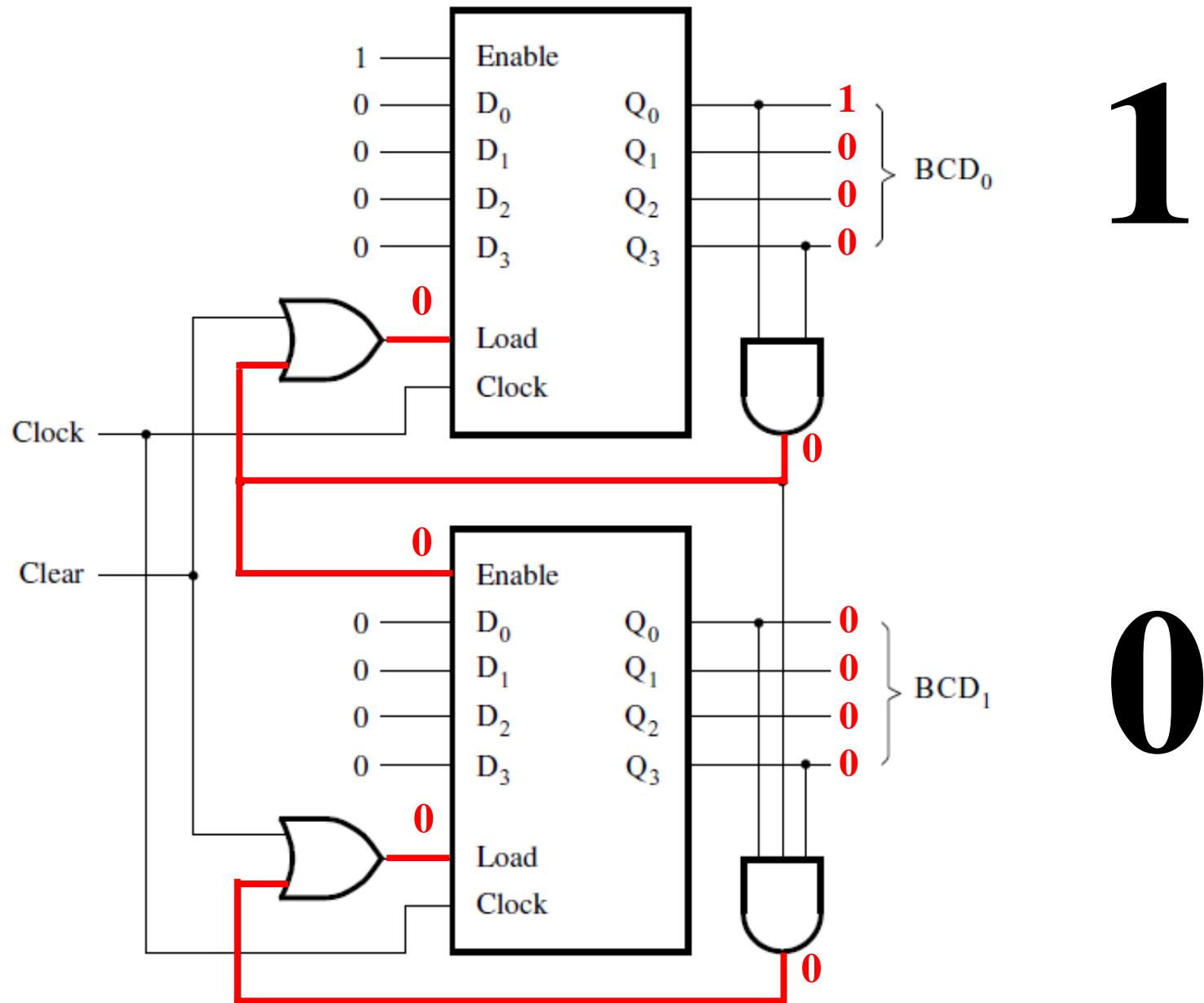
9

9



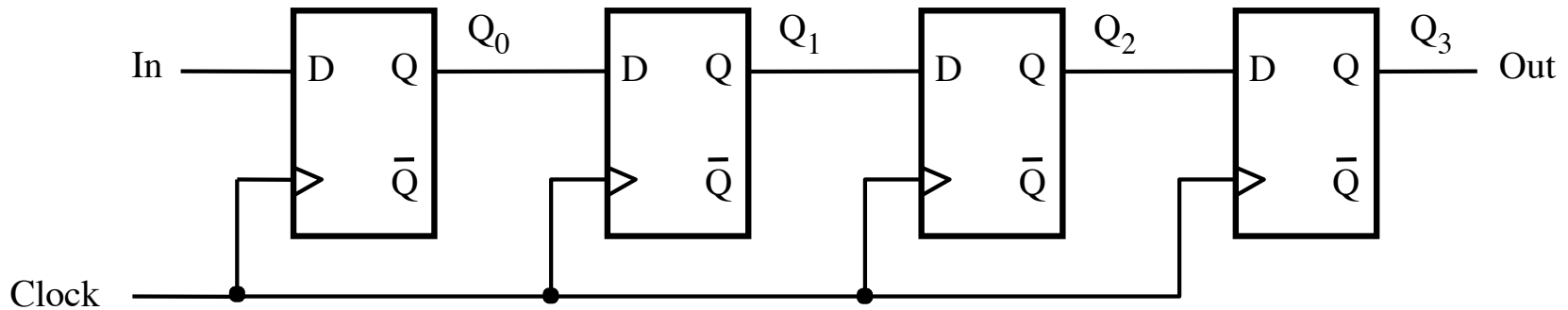


# Enabling the second counter



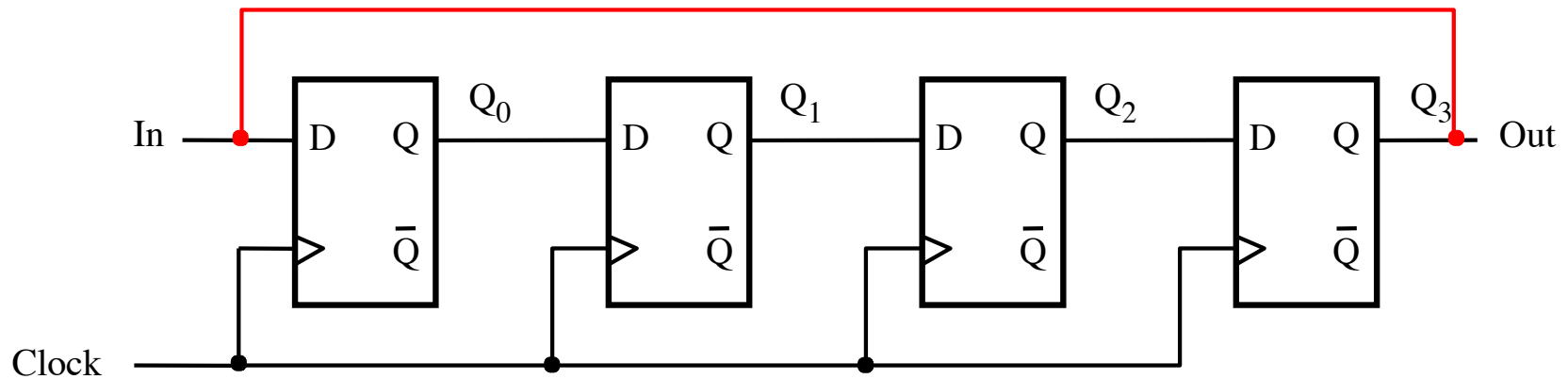
# Ring Counter

# How to build a 4-bit ring counter



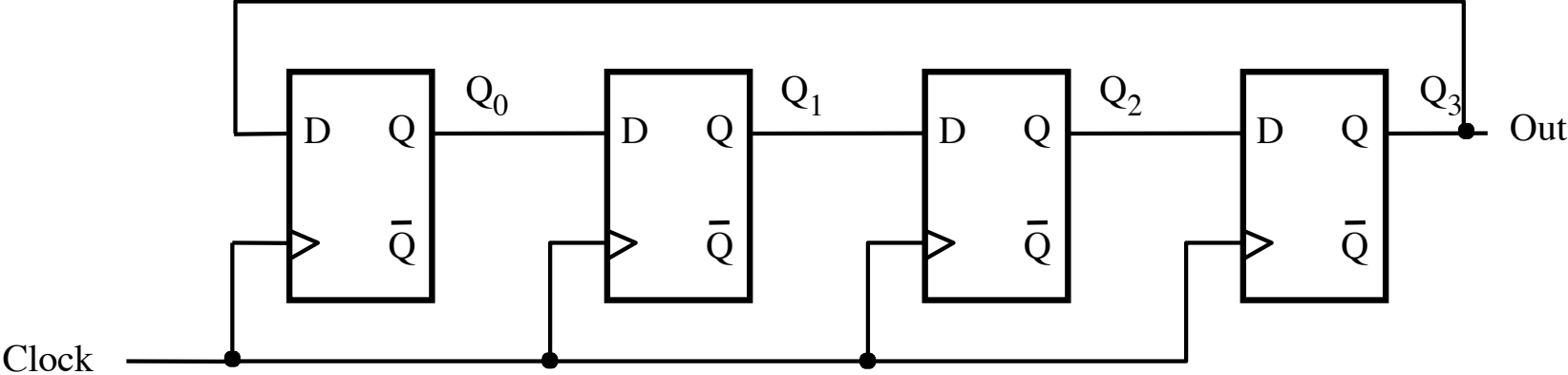
To build a ring counter we start with a shift register.

# How to build a 4-bit ring counter



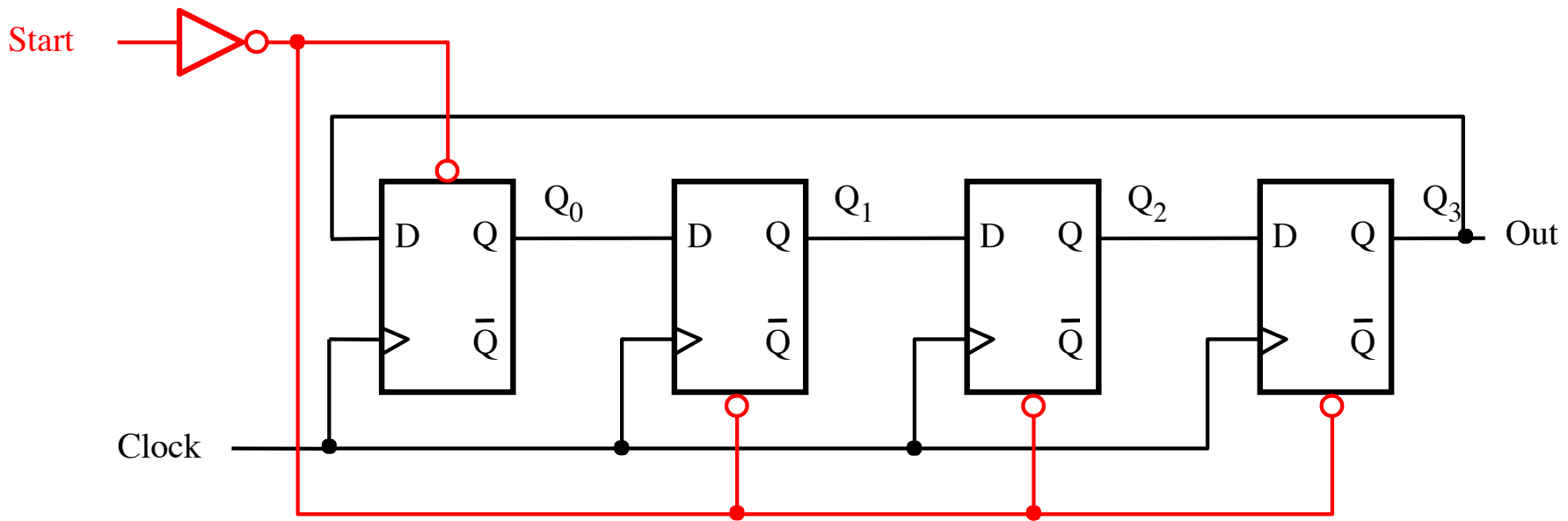
Next, add a loop from the last flip-flop to the first...

# How to build a 4-bit ring counter



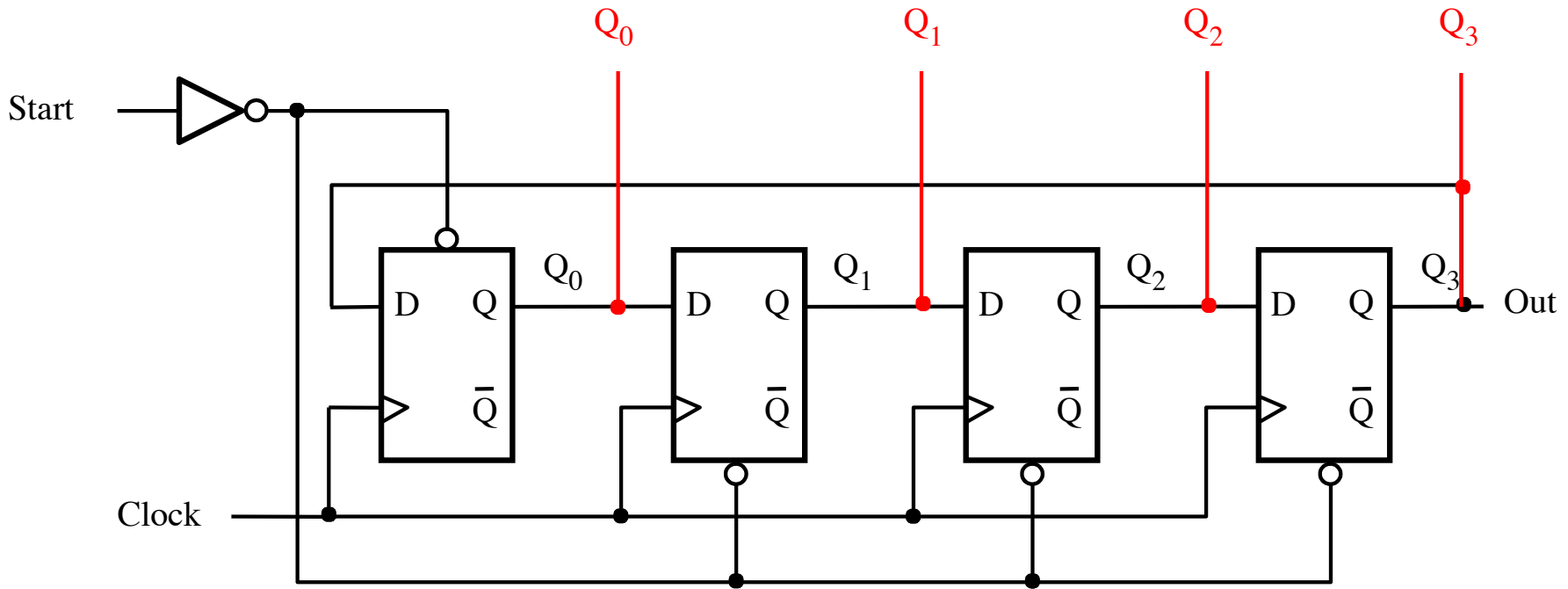
... and remove the In input line.

# How to build a 4-bit ring counter



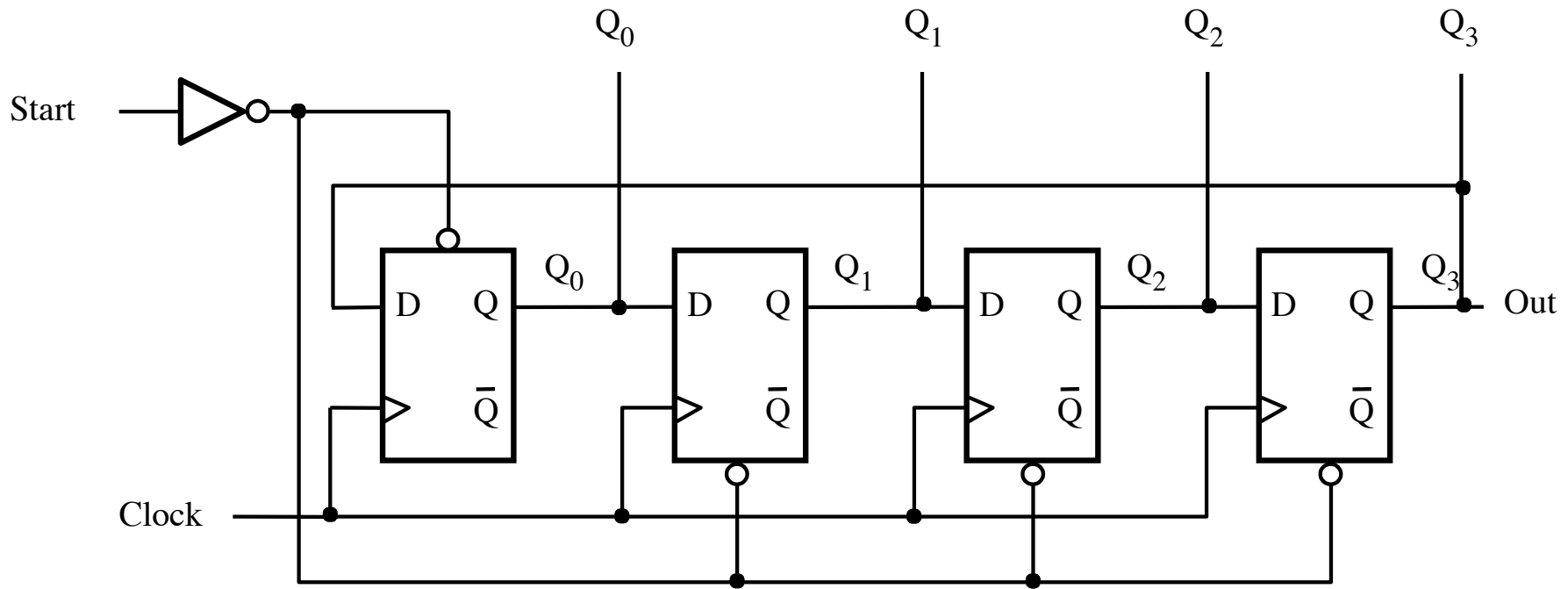
Also, add a start input that goes inverted to preset\_n of the first flip=flop and to clear\_n of all remaining flip-flops.

# How to build a 4-bit ring counter



Finally, extend the output lines that form the count number.

# How to build a 4-bit ring counter



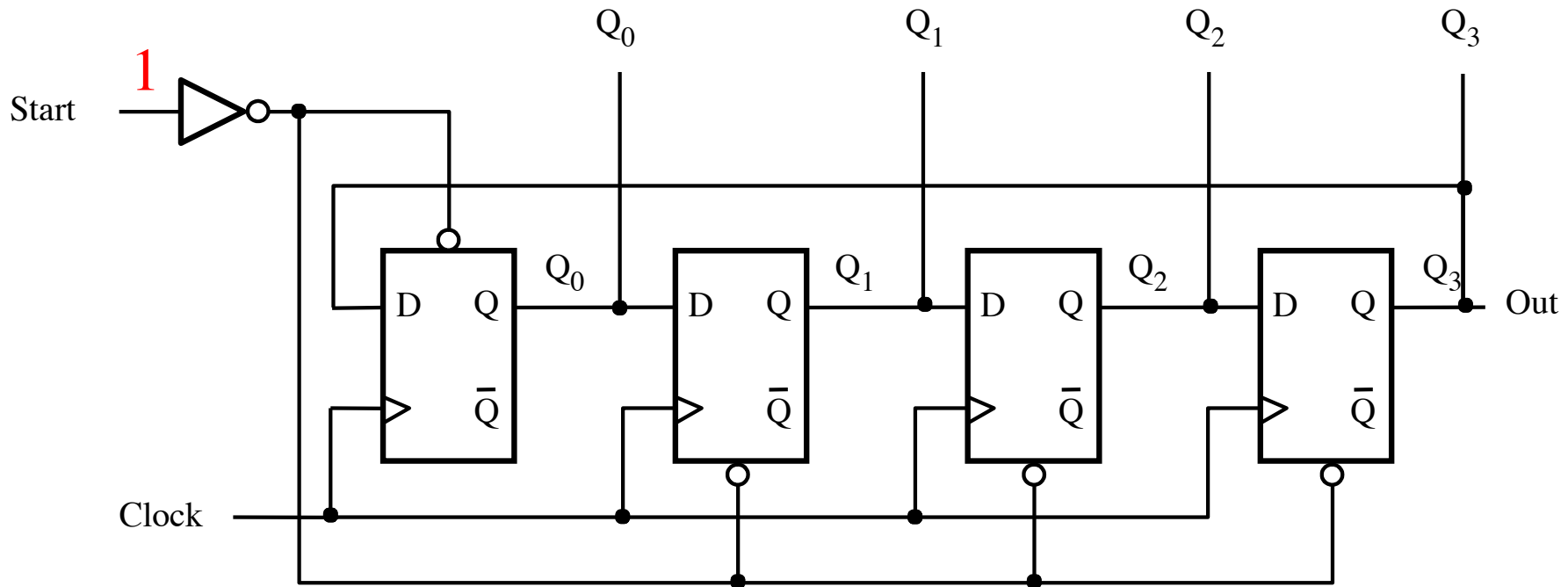
This is the final circuit diagram.



# 4-bit ring counter

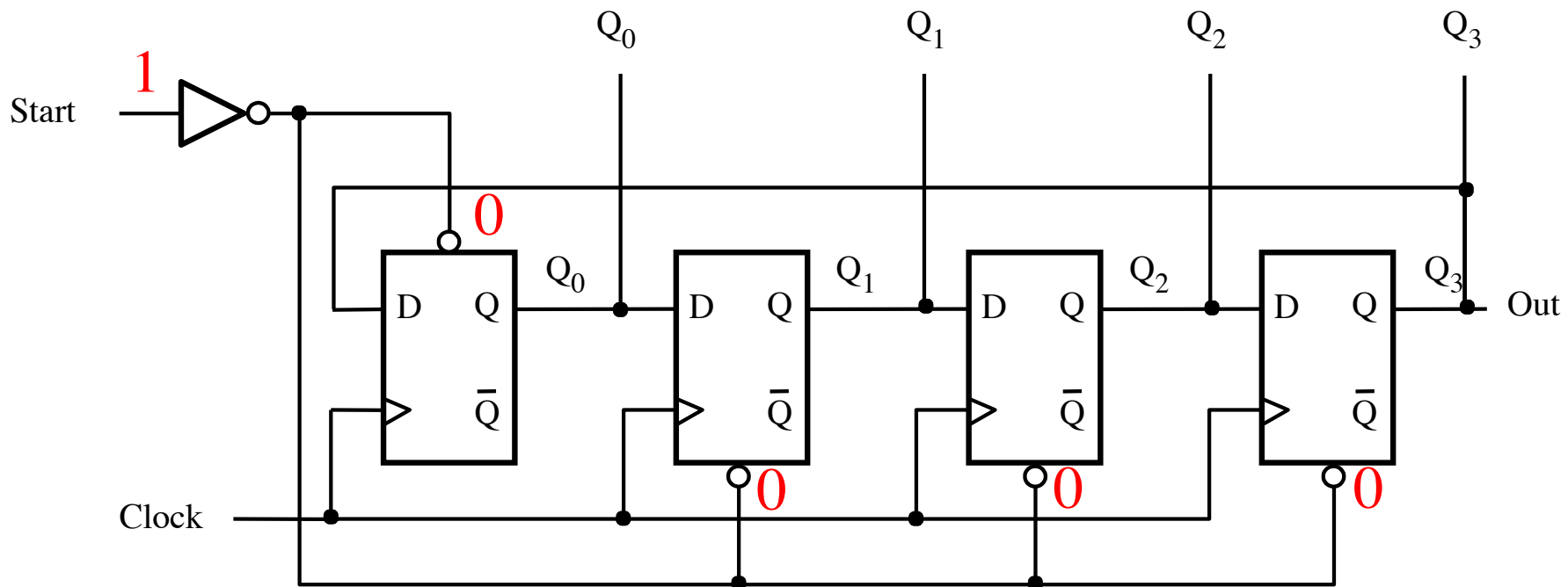
- There is only one 1 on the outputs of the four flip-flops
- The counting sequence is: 1000, 0100, 0010, 0001, 1000, ...
- To reset the counter
  - set start to 1 for a short period of time
  - This sets the four outputs to 1000

# 4-bit ring counter: How does it work



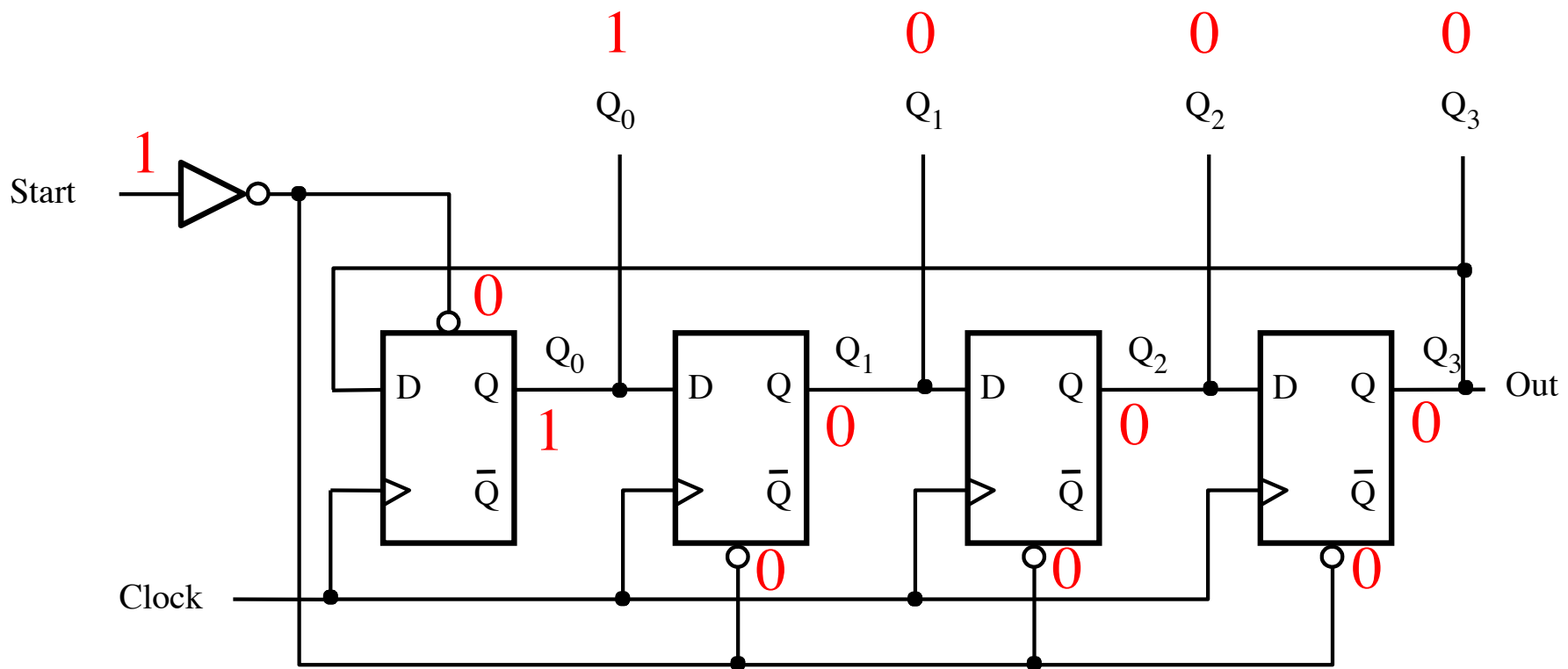
To initialize the counter set Start to 1.

# 4-bit ring counter: How does it work



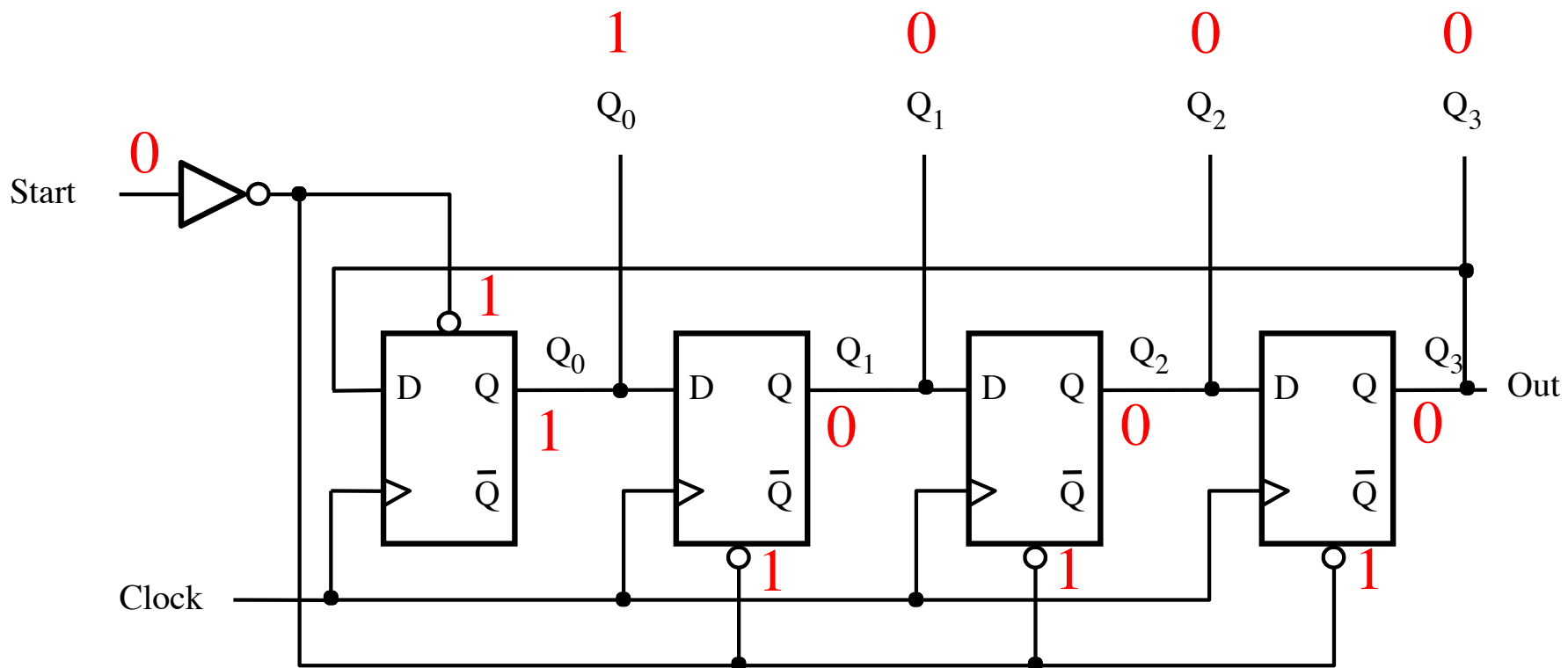
After the NOT gate, this 1 goes as 0 to preset\_n of the first flip-flop and to clear\_n of all remaining flip-flops.

# 4-bit ring counter: How does it work



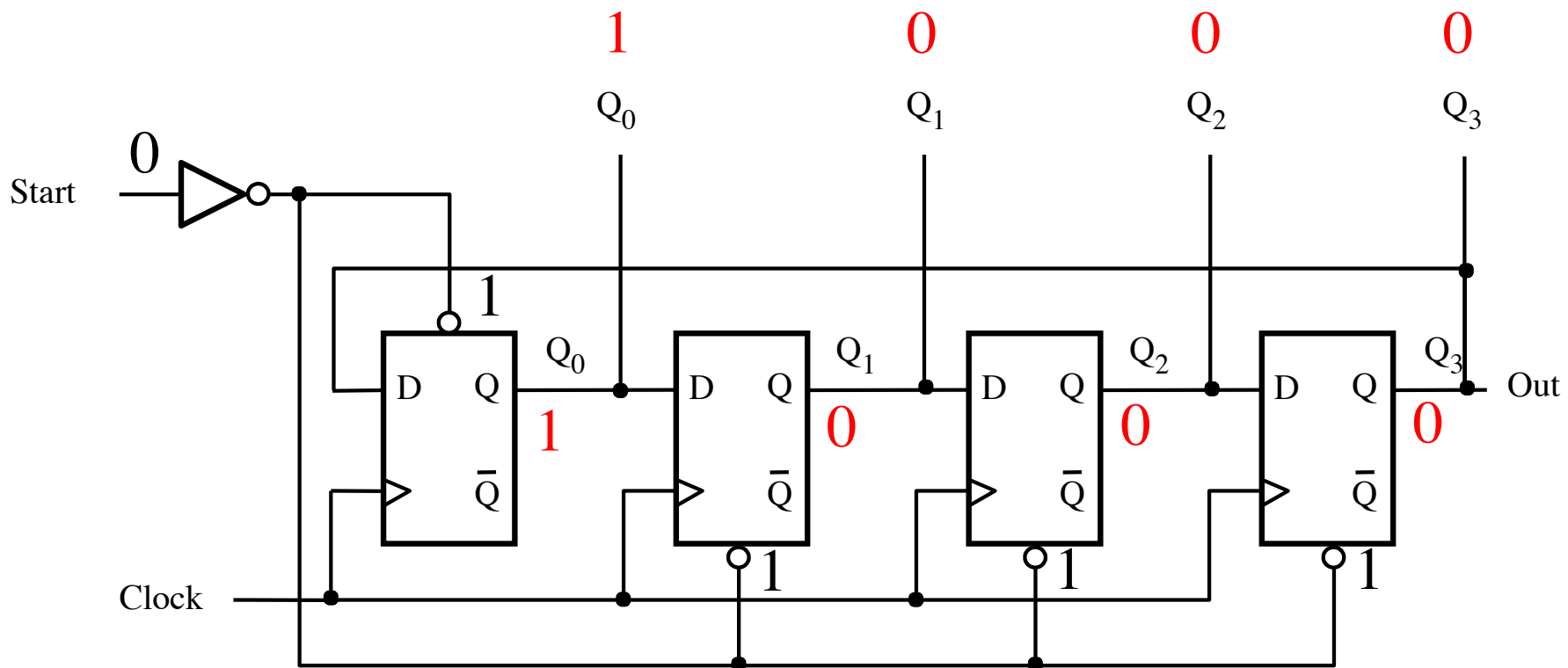
This sets the output pattern to 1000,  
i.e., only the first bit is one and the rest are zeros.

# 4-bit ring counter: How does it work



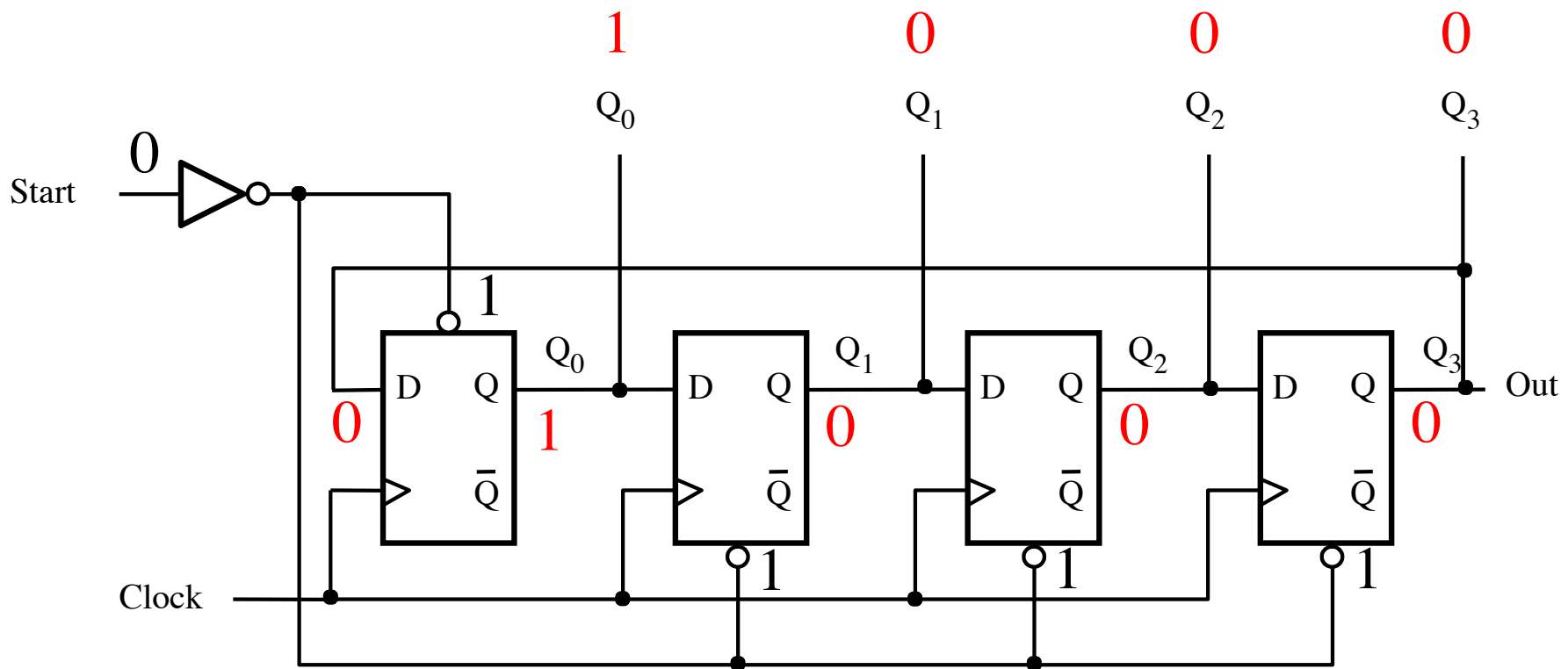
Setting Start to 0 has no effect on the outputs,  
because both preset\_n and clear\_n are sensitive only to 0.

# 4-bit ring counter: How does it work



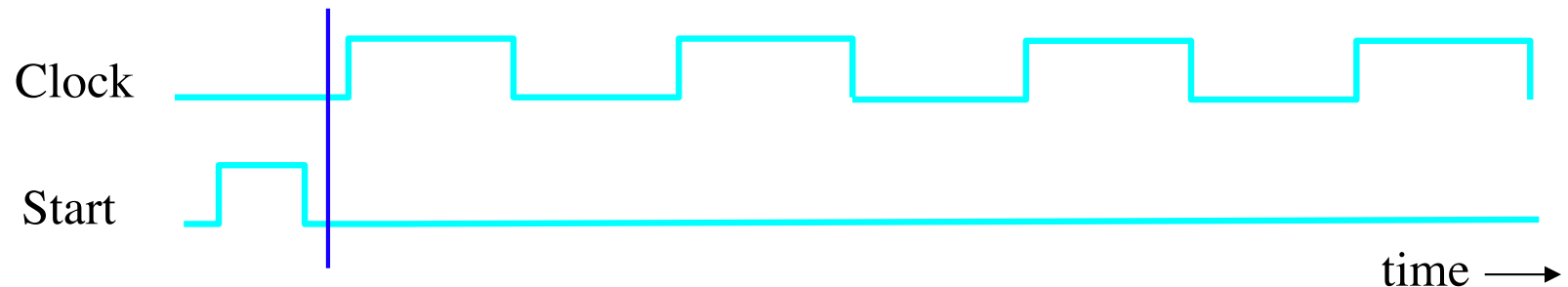
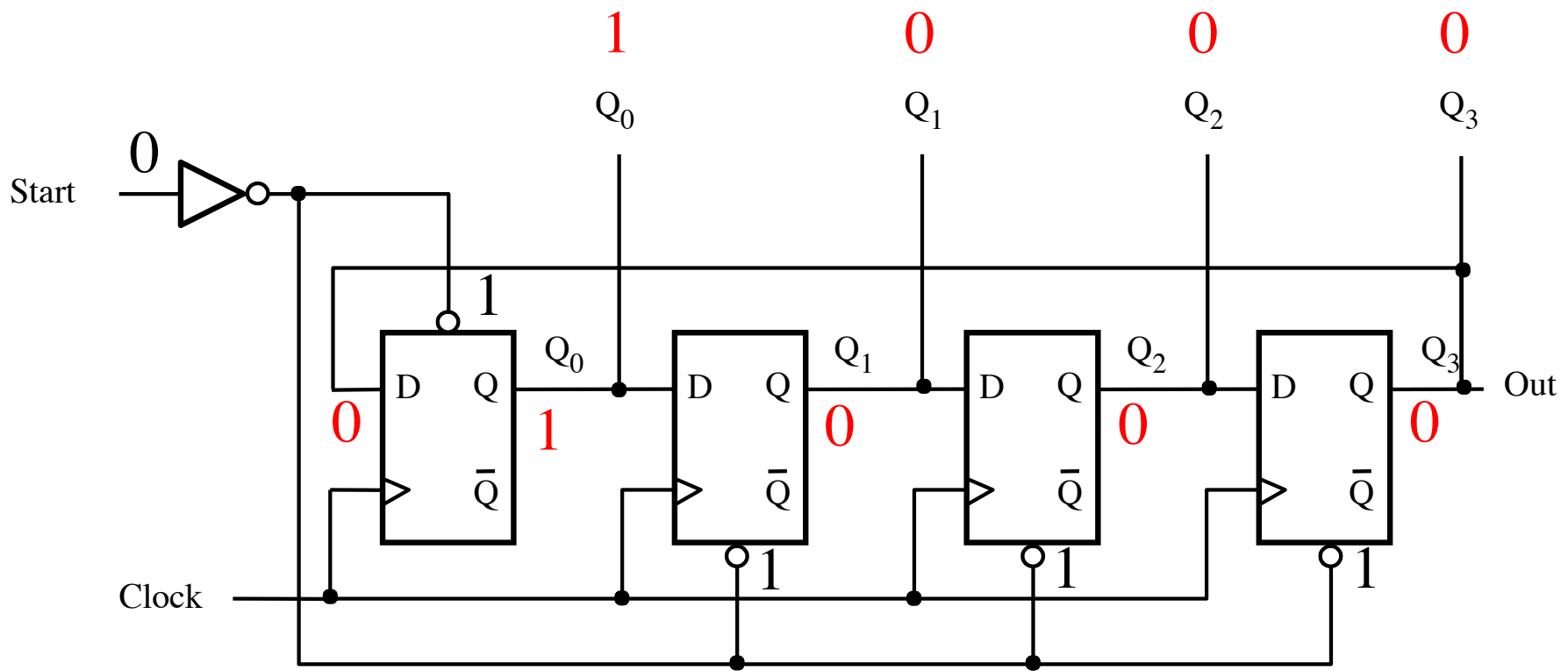
The initialization does not depend on the clock since both `preset_n` and `clear_n` bypass the gates of the latches in the flip-flops.

# 4-bit ring counter: How does it work



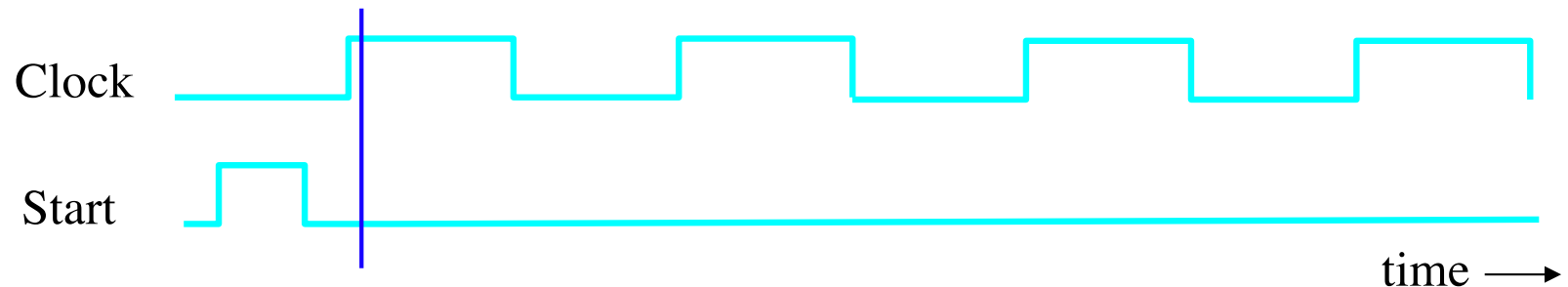
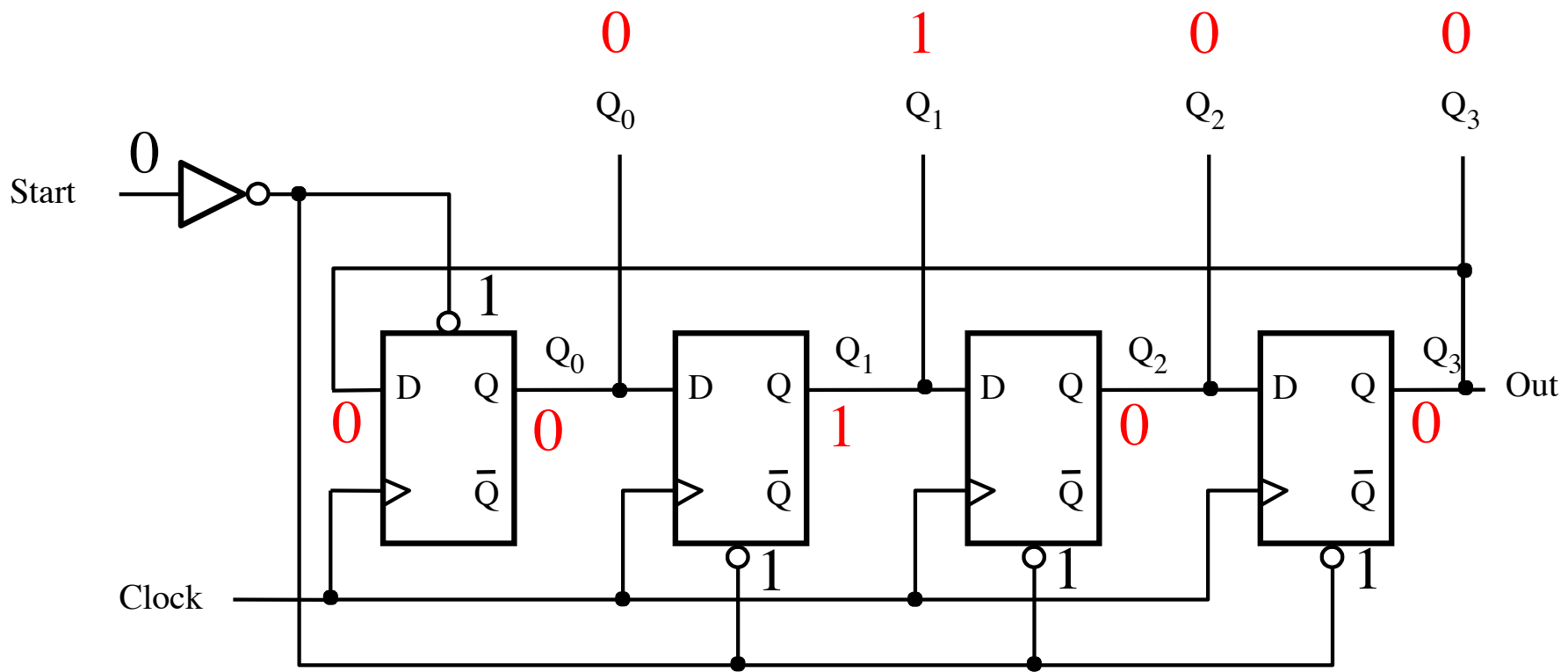
That last 0 loops back to the D input of the first flip-flop.

# 4-bit ring counter: How does it work

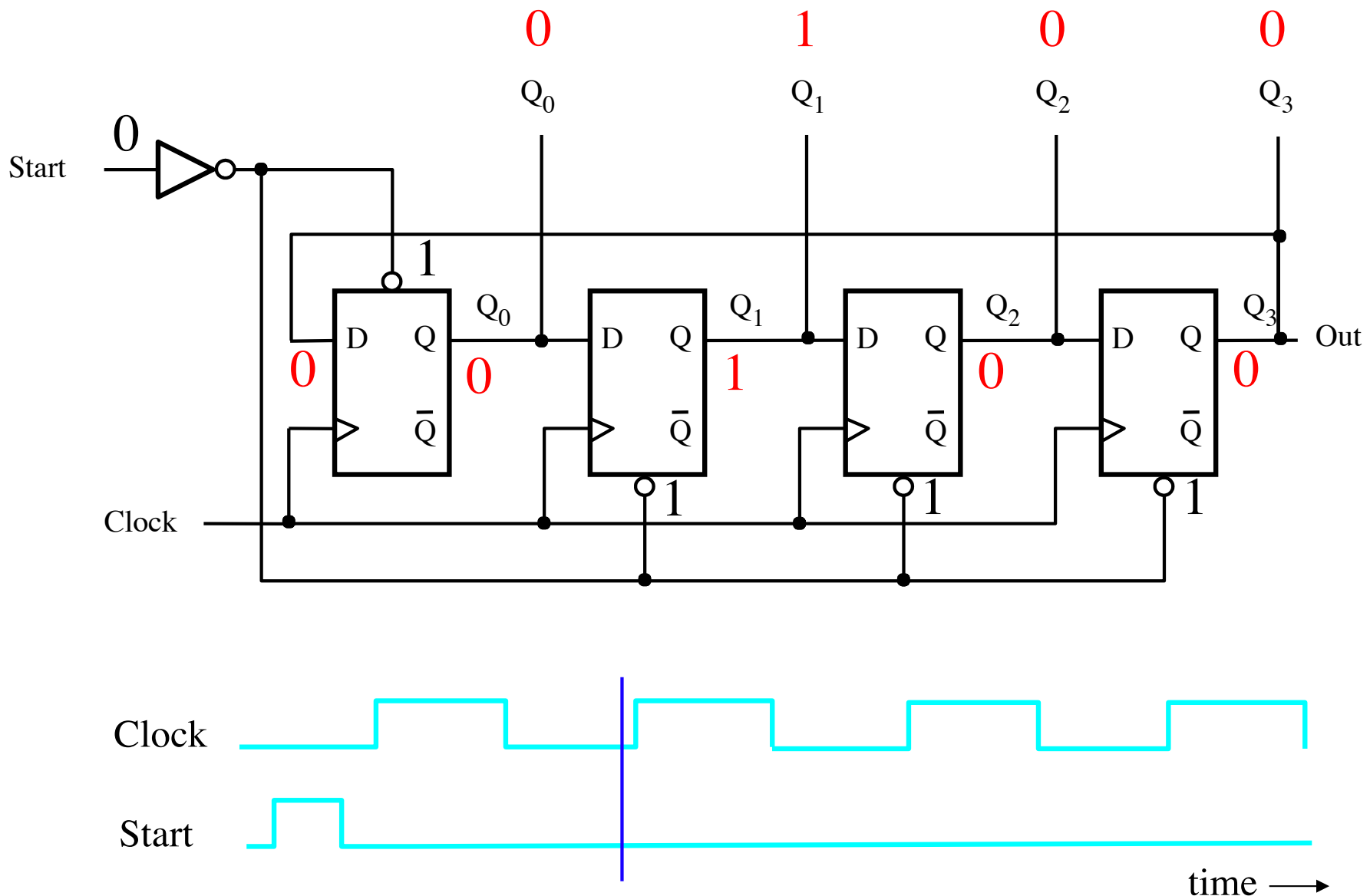




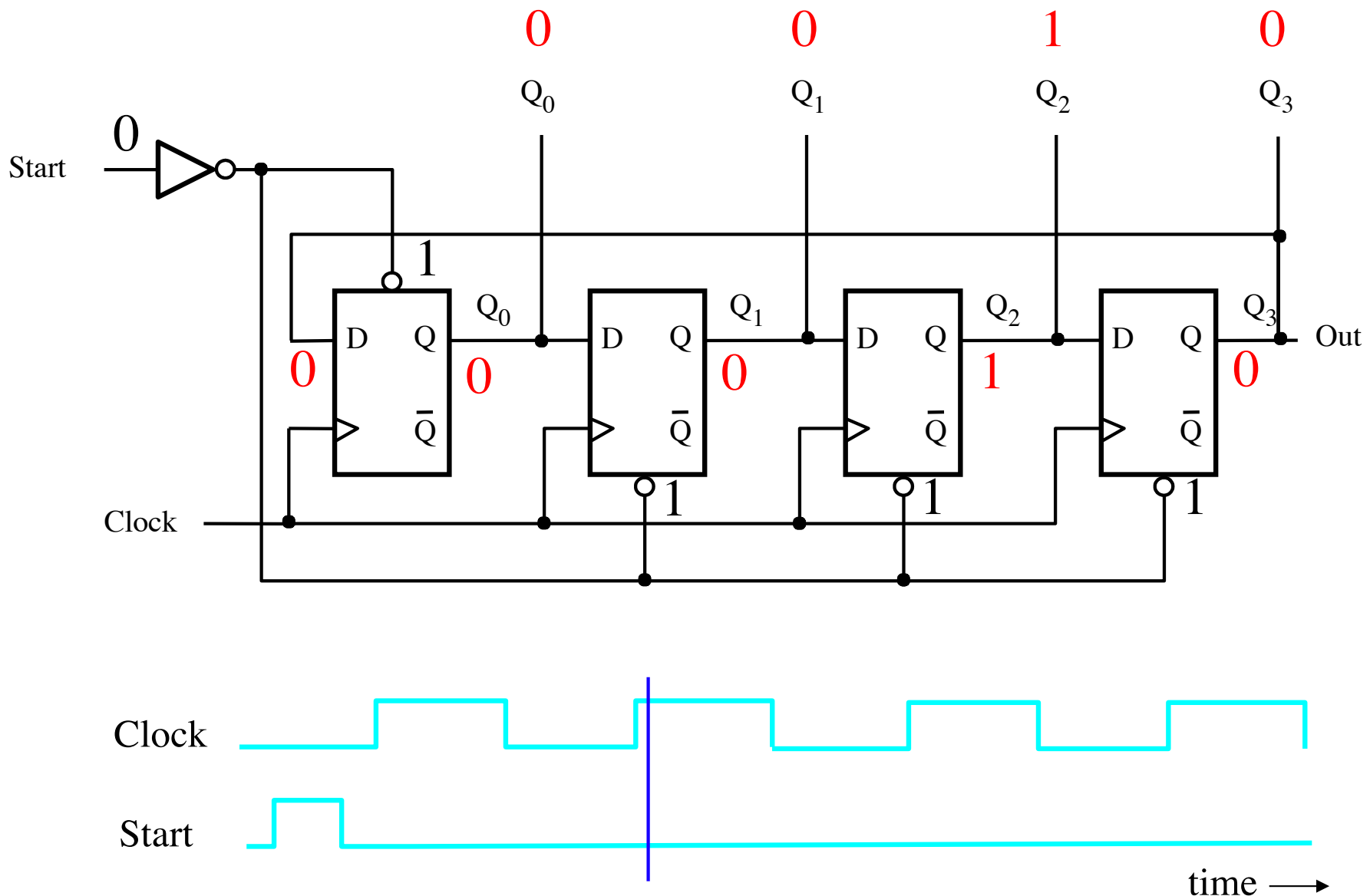
# 4-bit ring counter: How does it work



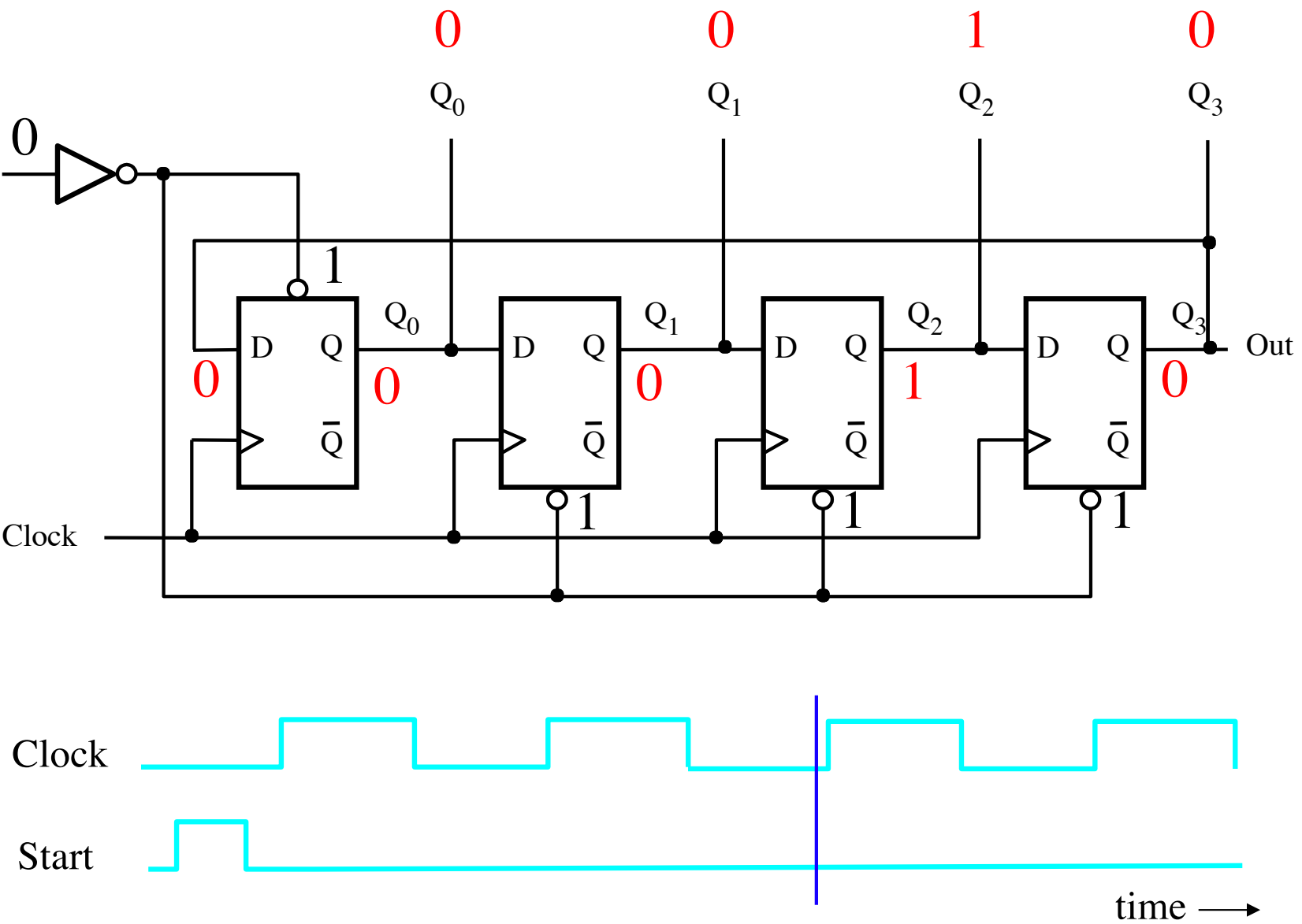
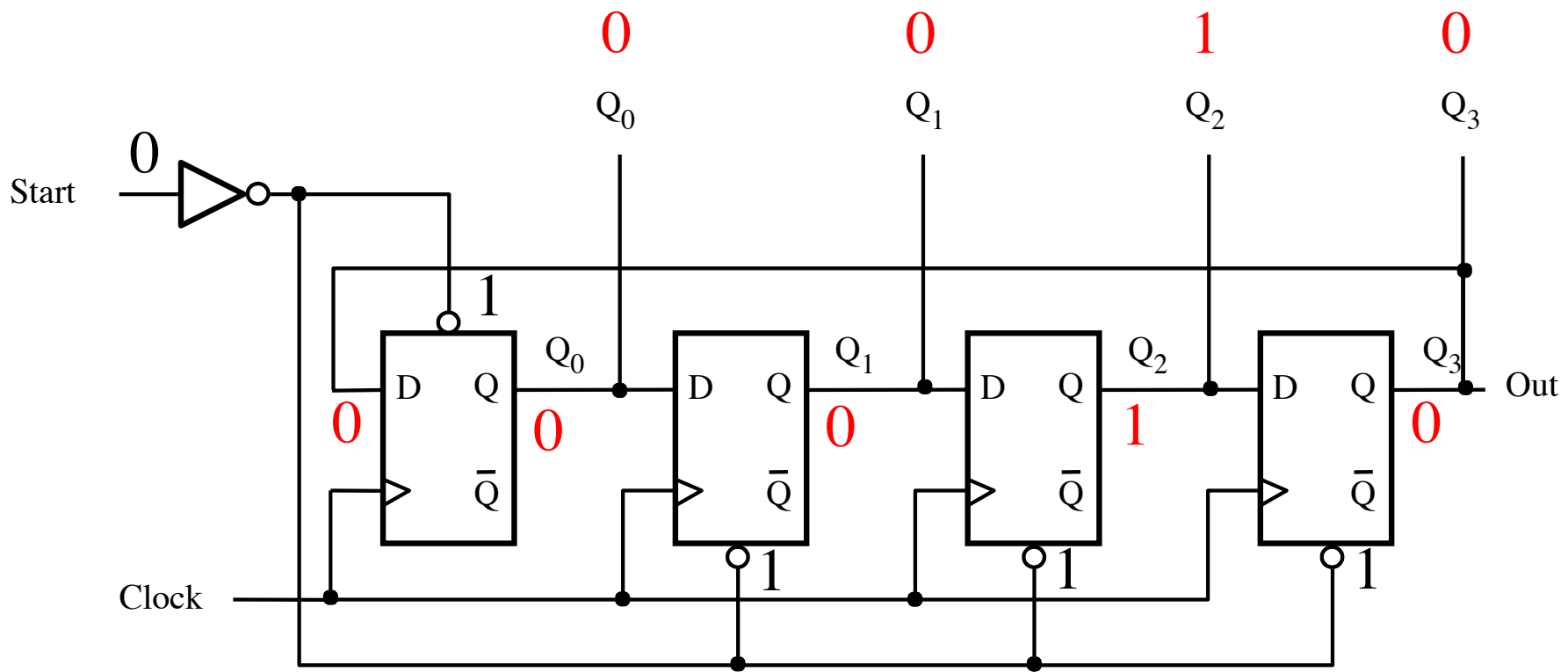
# 4-bit ring counter: How does it work



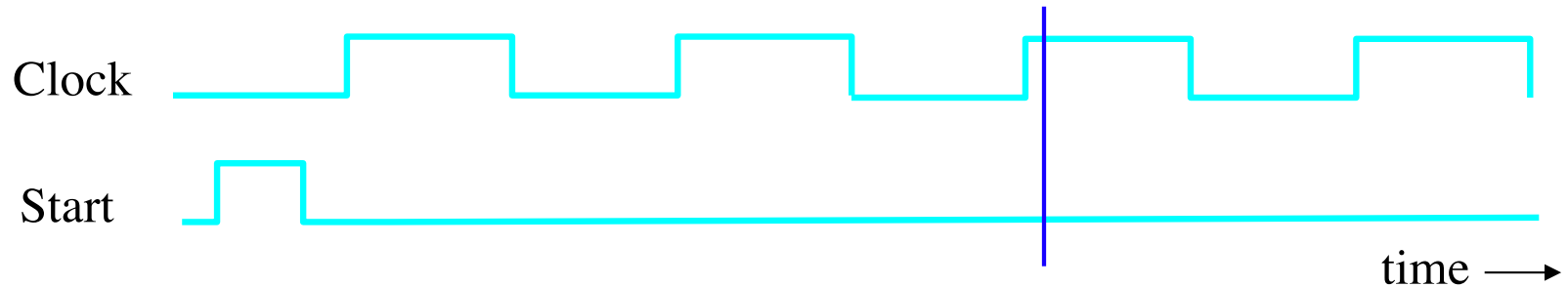
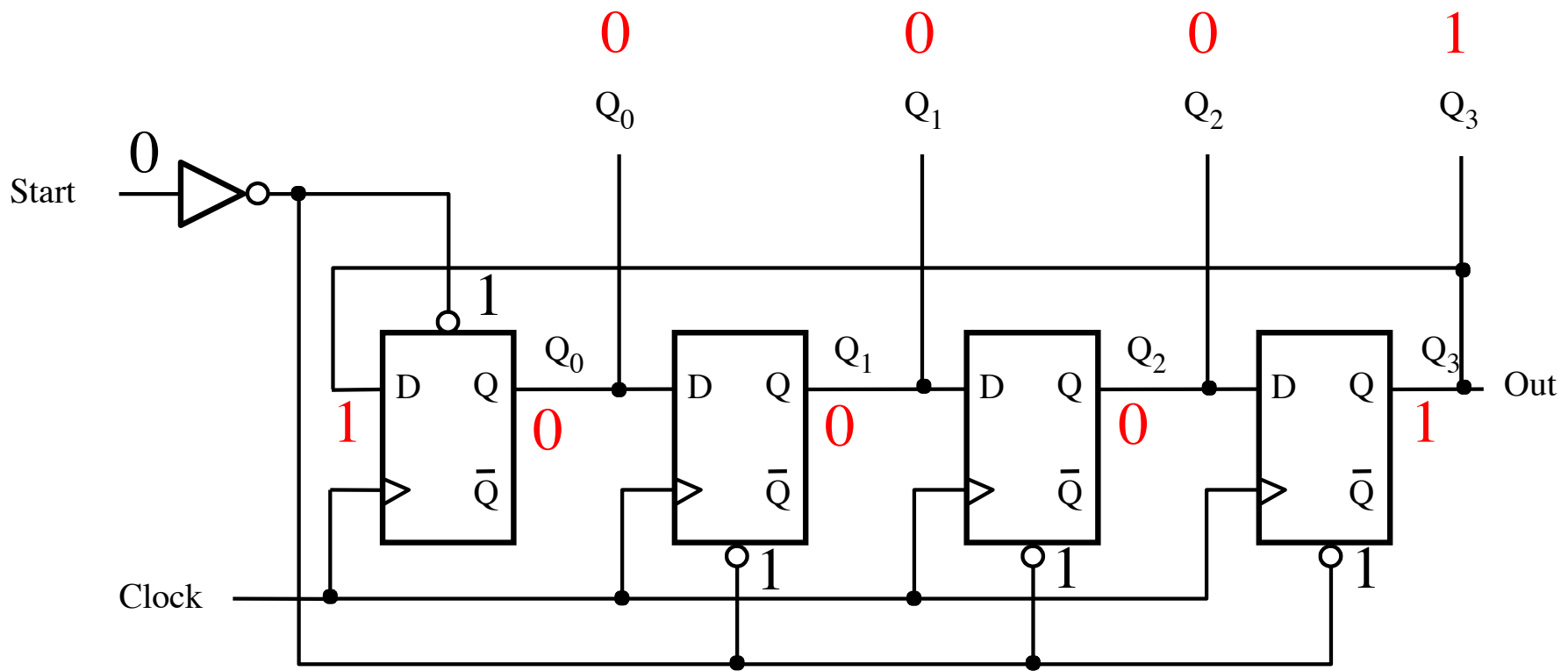
# 4-bit ring counter: How does it work



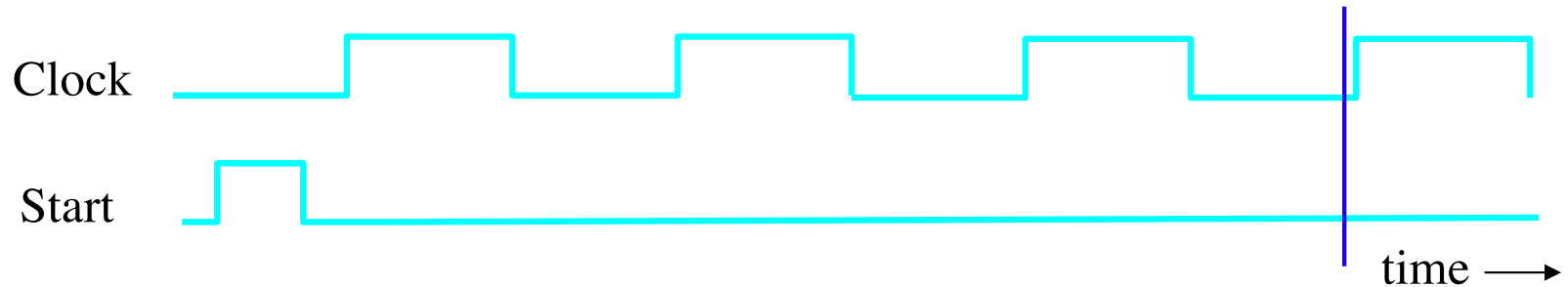
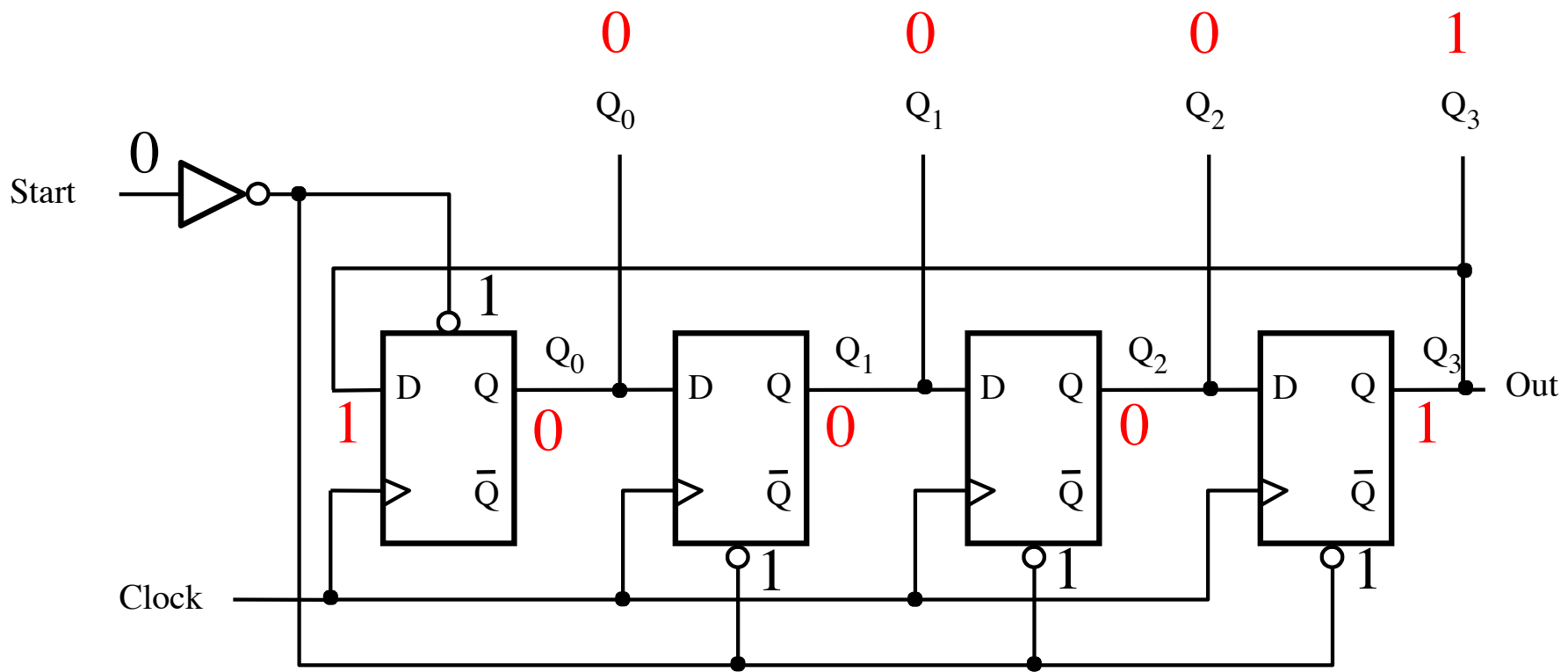
# 4-bit ring counter: How does it work



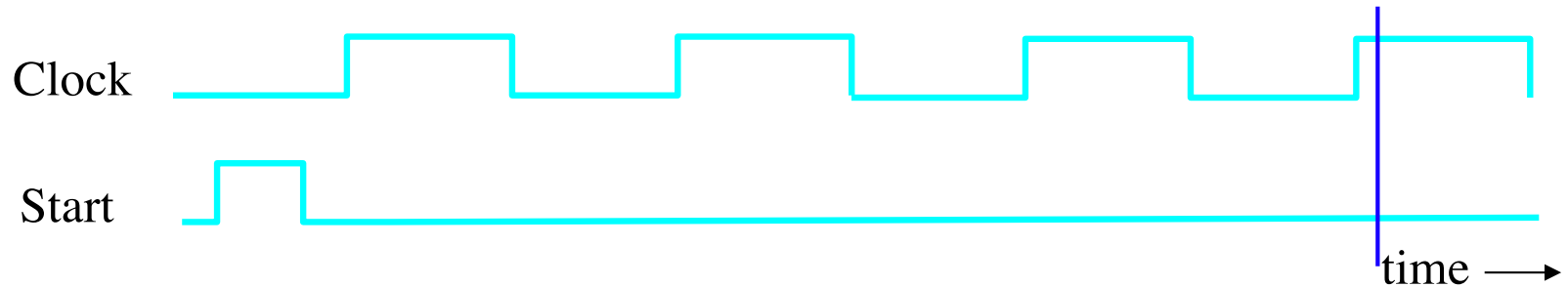
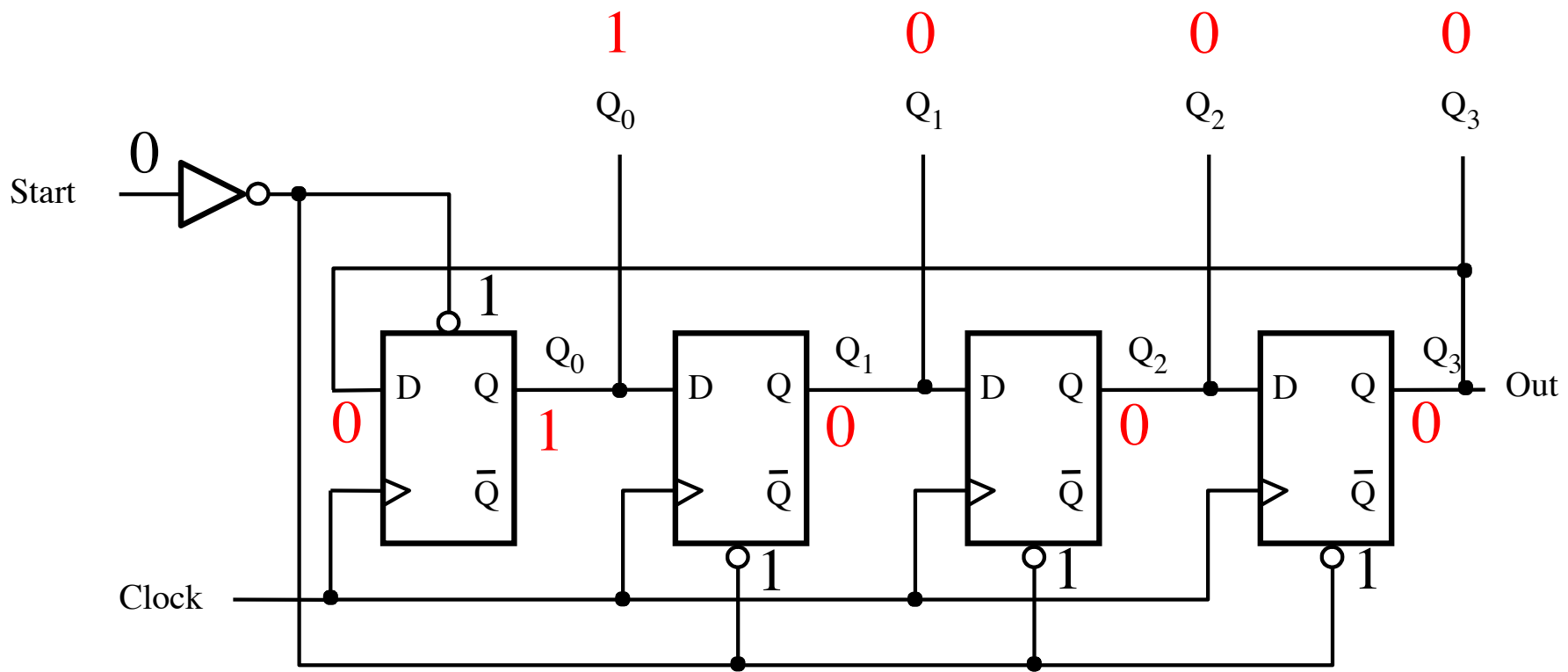
# 4-bit ring counter: How does it work



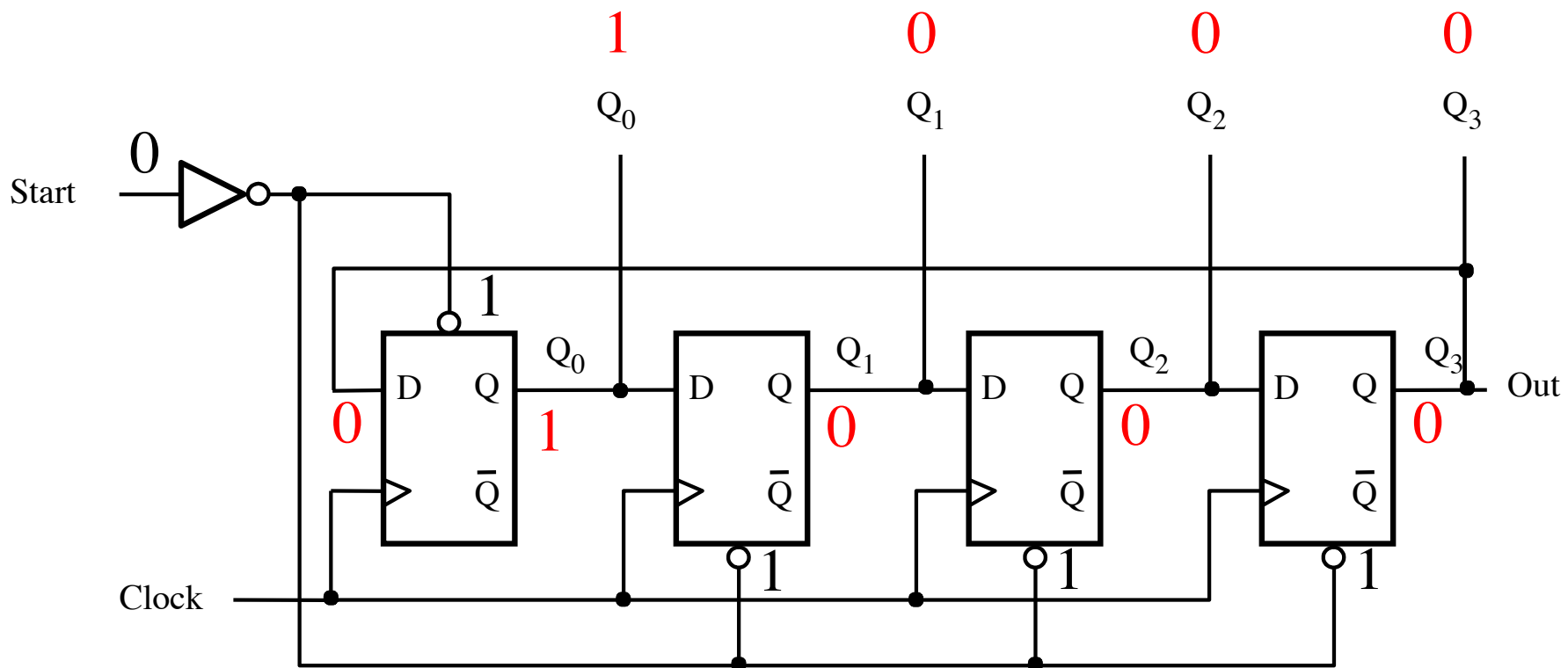
# 4-bit ring counter: How does it work



# 4-bit ring counter: How does it work



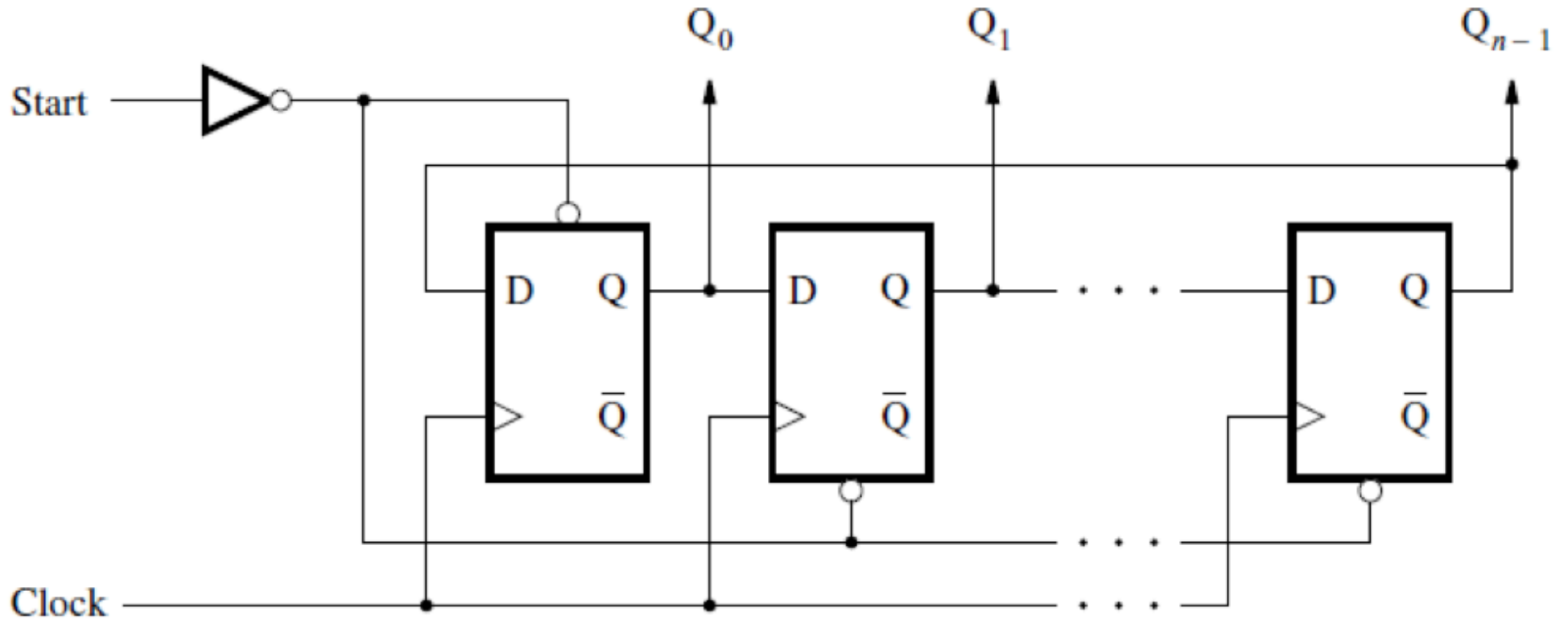
# 4-bit ring counter: How does it work



It is back to the start of the counting sequence,  
which is: 1000, 0100, 0010, 0001.



# n-bit ring counter

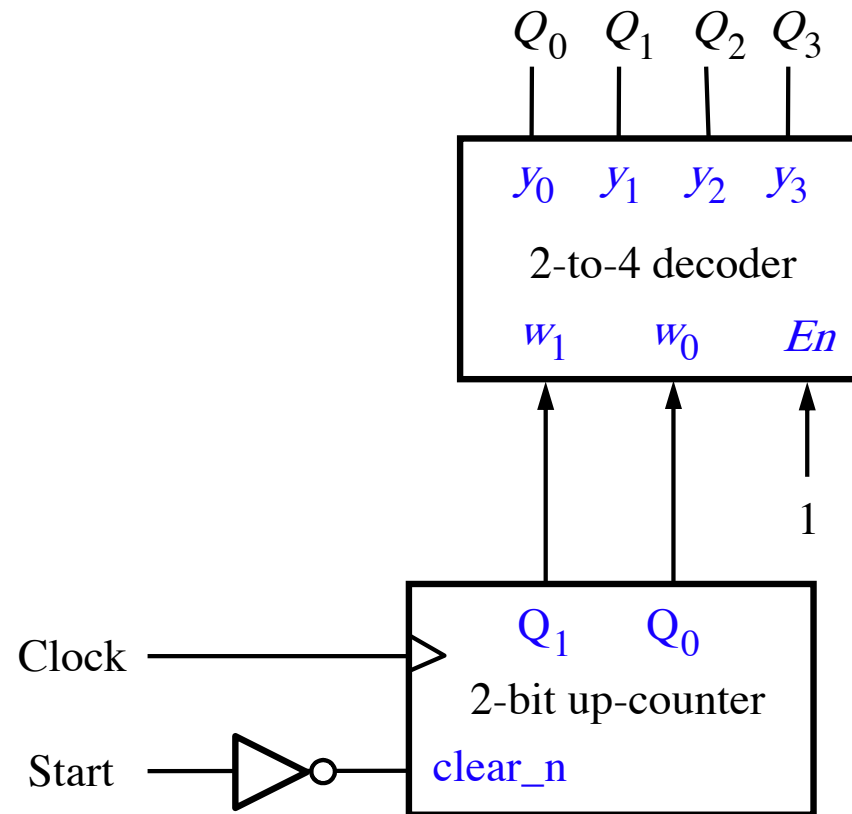


# **Ring Counter (alternative implementation)**

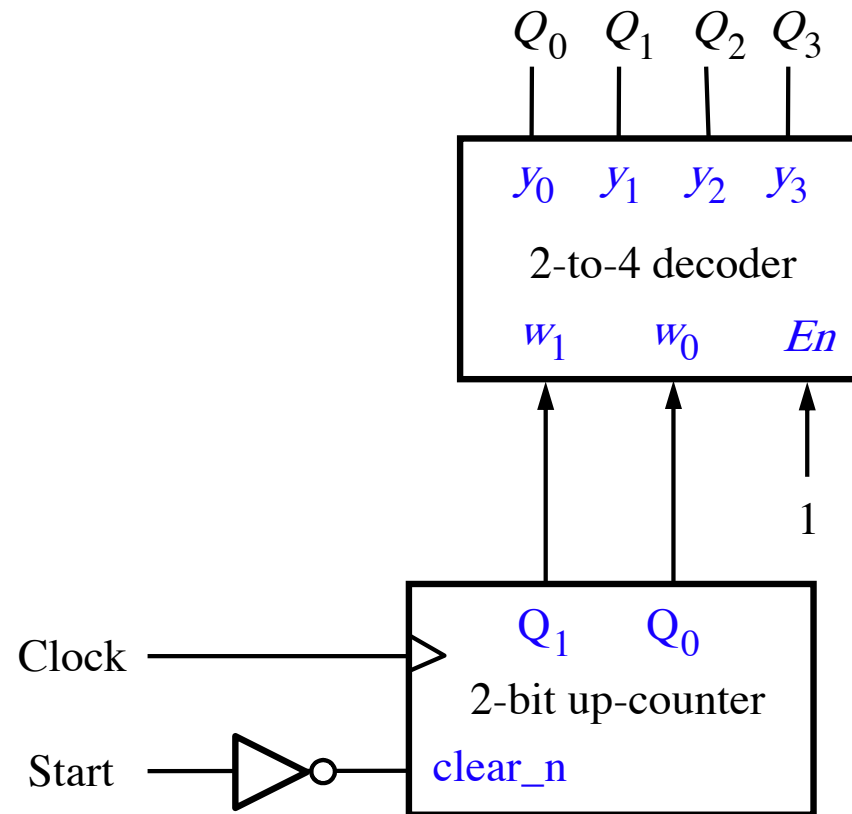
# Alternative version of a 4-bit ring counter

- This implementation uses 2-bit up-counter followed by a 2-to-4 decoder.
- The counter cycles through 00, 01, 10, 11, 00, ...
- Recall that the outputs of the decoder are one-hot encoded. Thus, there is only one 1 on its outputs.
- Because the output of the counter is the input to the decoder, the outputs of the decoder cycle through: 1000, 0100, 0010, 0001, 1000, ...
- This is the counting sequence for a ring counter.

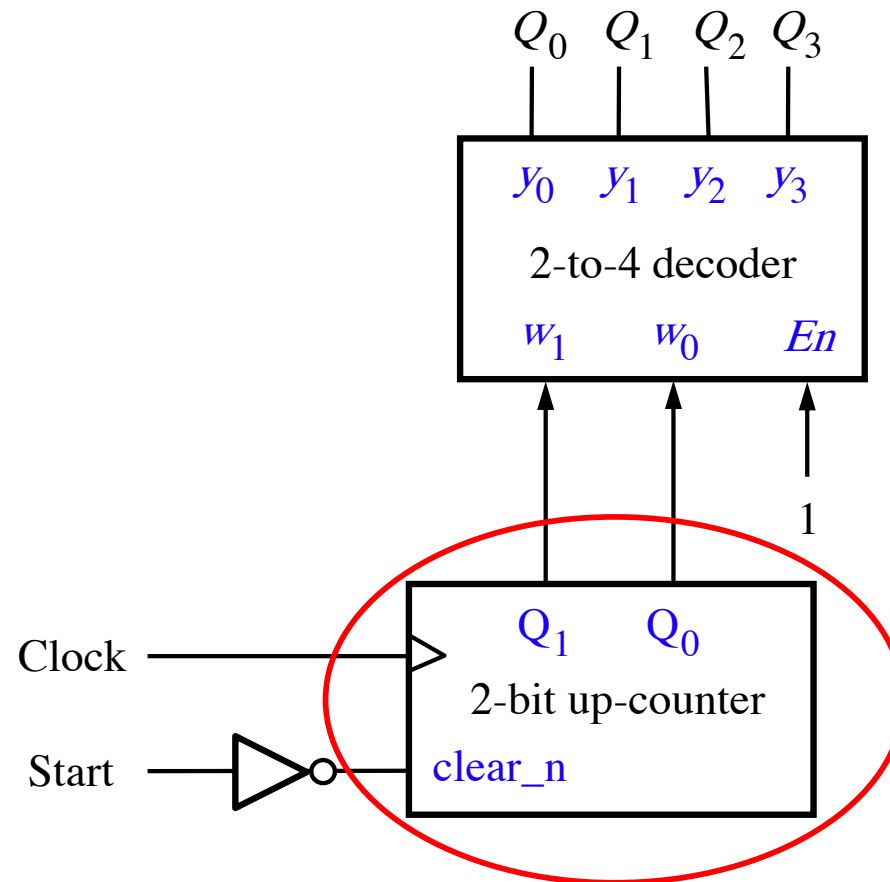
# Alternative version of a 4-bit ring counter



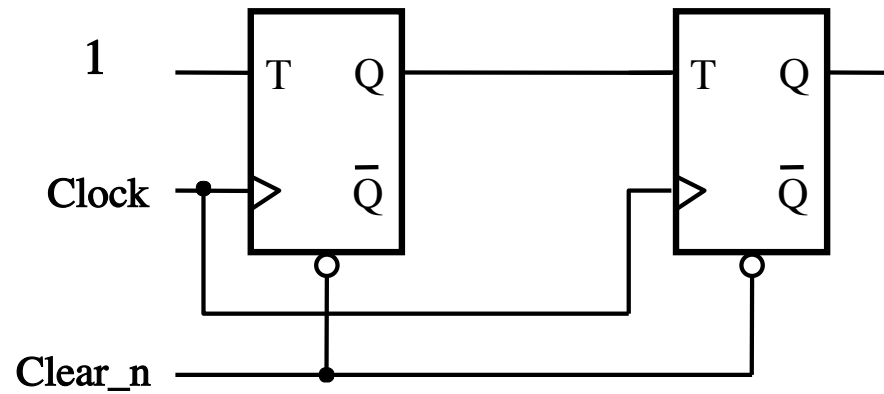
# What are the components?



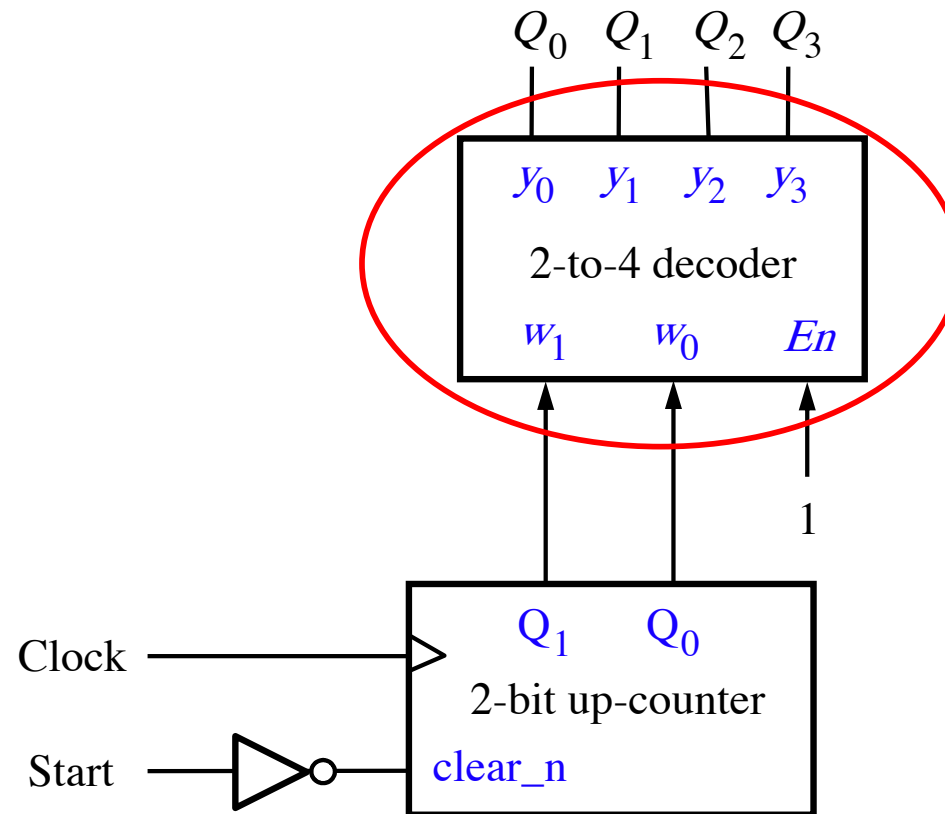
# 2-Bit Synchronous Up-Counter



# 2-Bit Synchronous Up-Counter



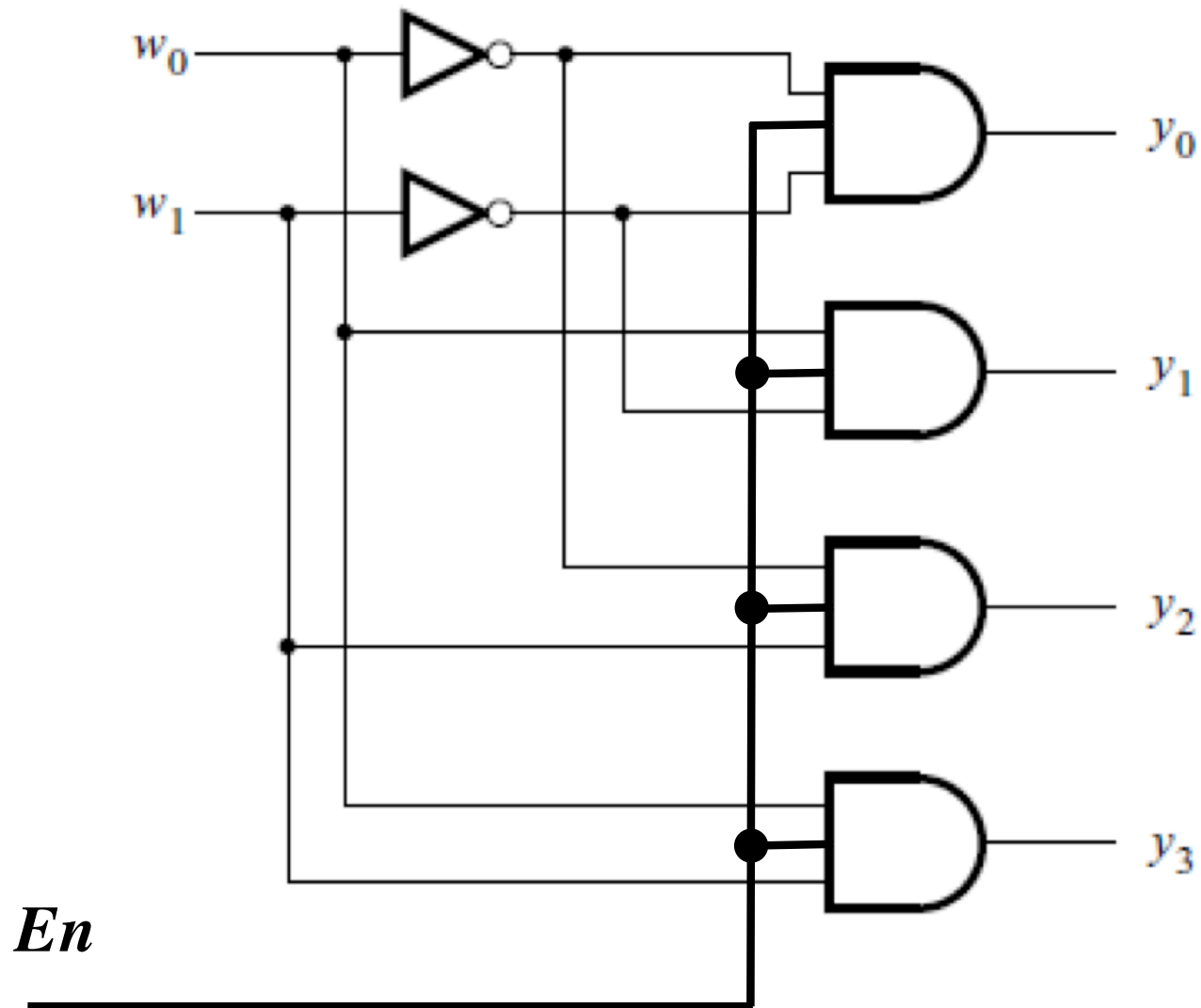
# 2-to-4 Decoder with Enable Input



[ Figure 5.28b from the textbook ]

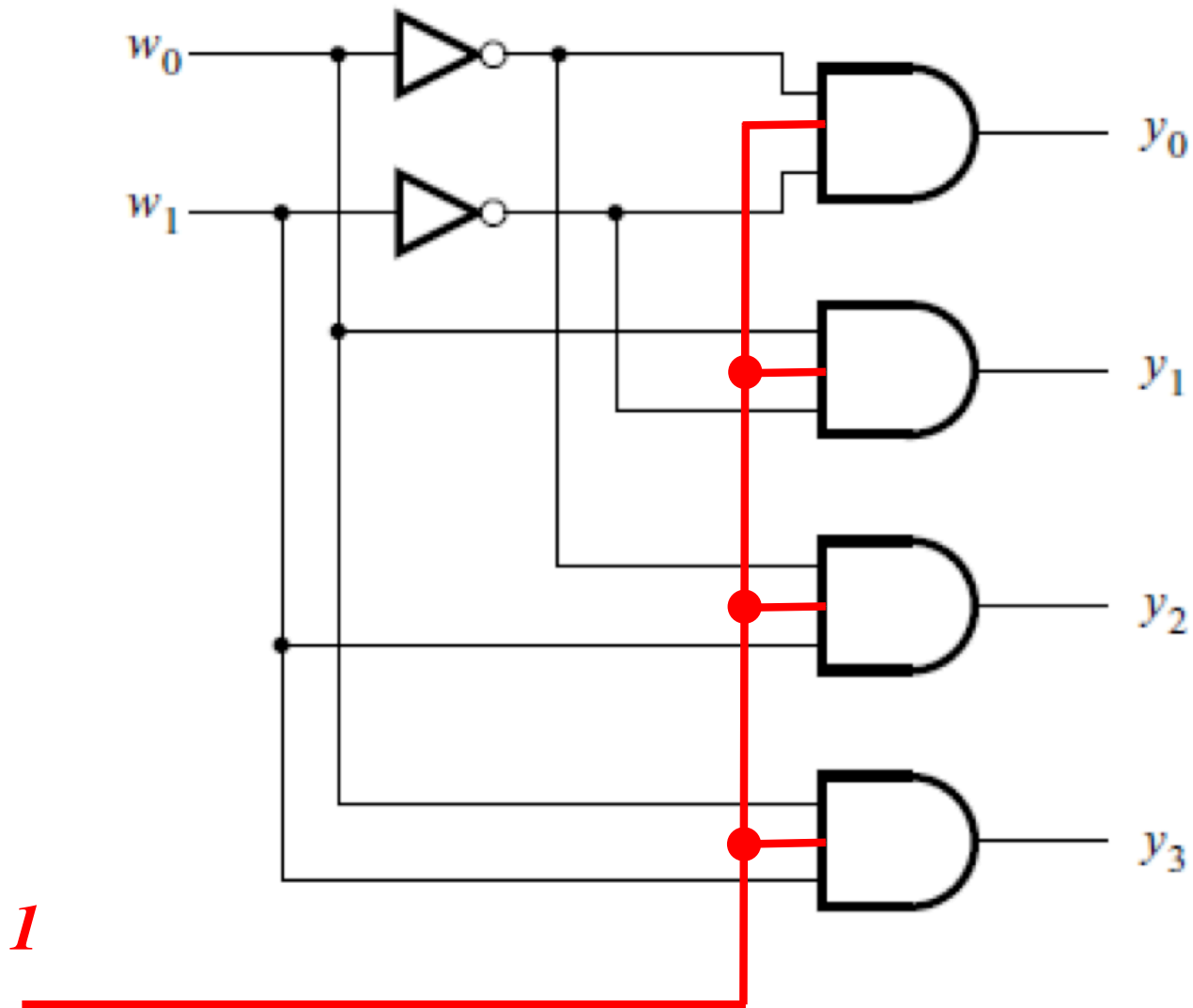


# 2-to-4 Decoder with Enable Input



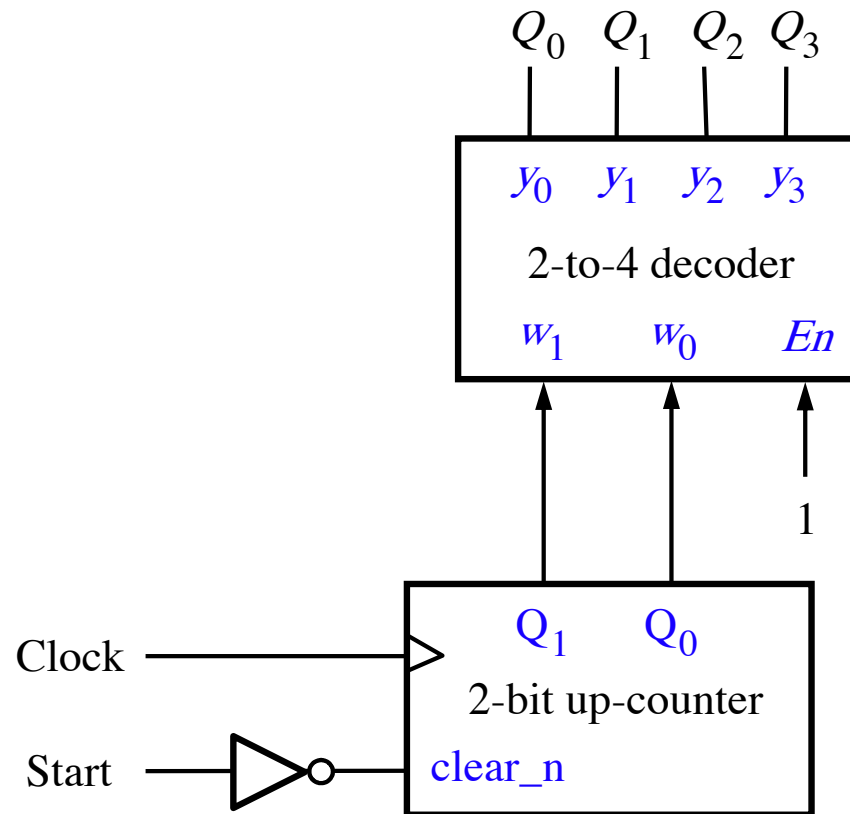
[ Figure 4.14c from the textbook ]

# 2-to-4 Decoder with Enable Input

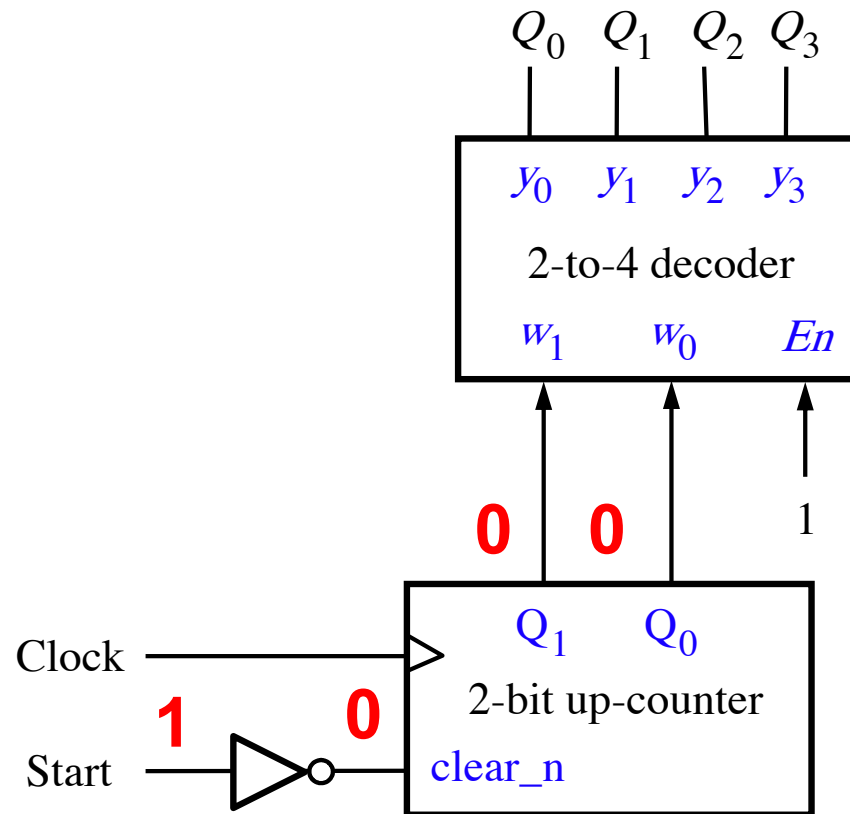


(always enabled in this example)

# How Does It Work?

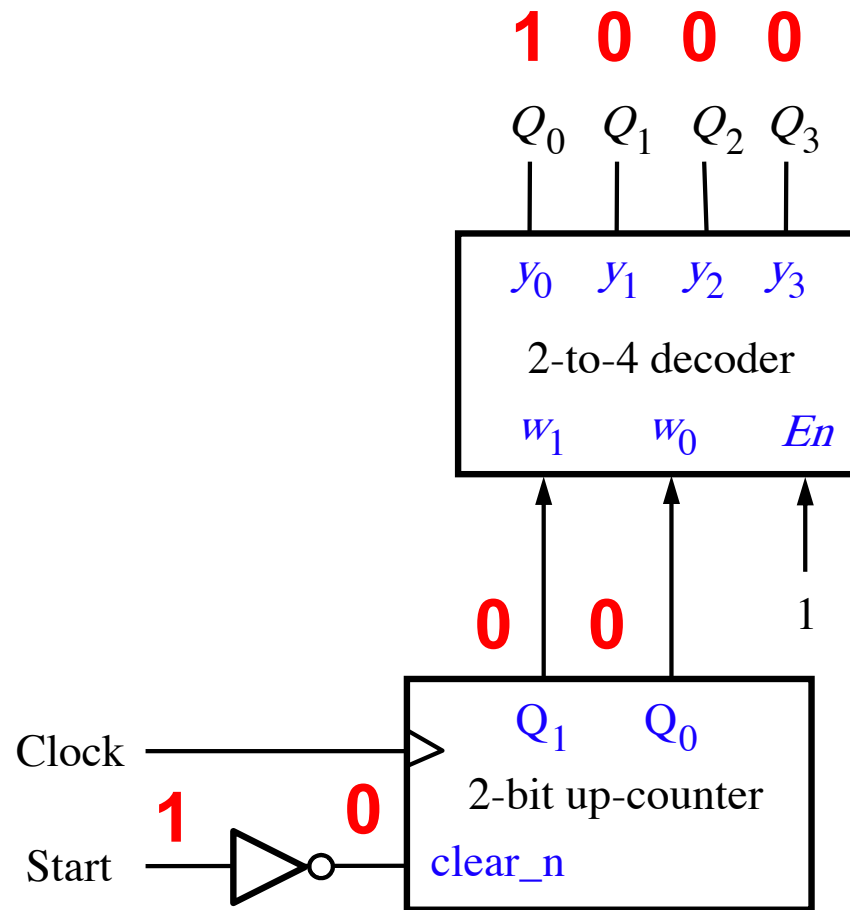


# How Does It Work?



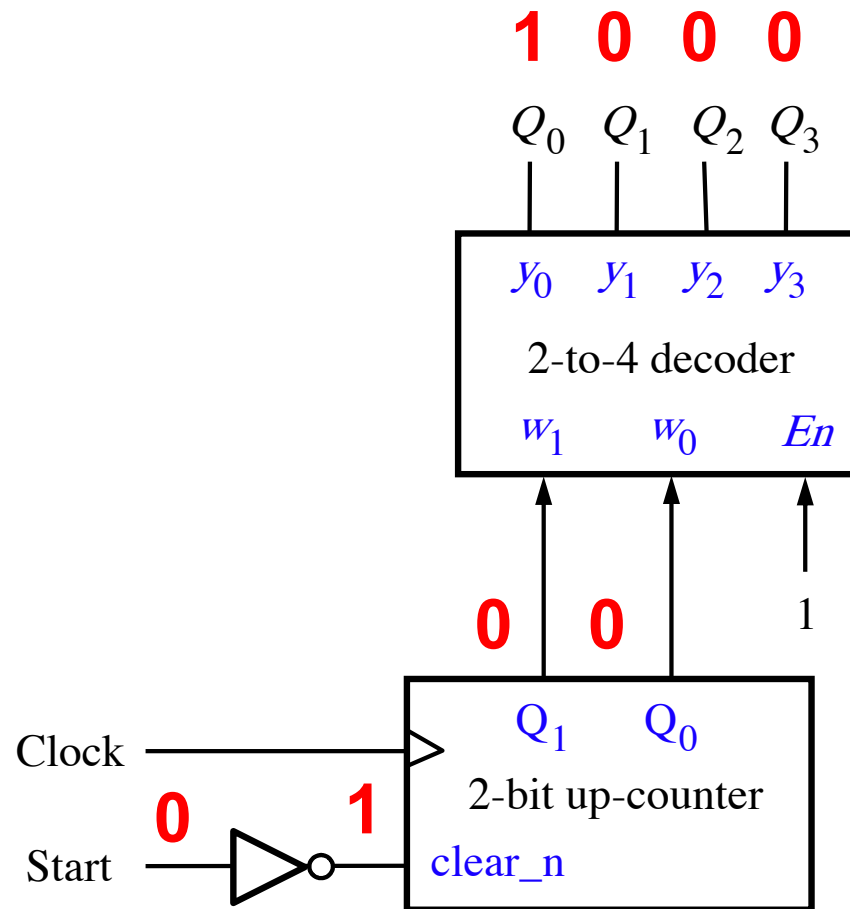
To initialize this circuit set Start to 1, which sets the counter to 00.

# How Does It Work?



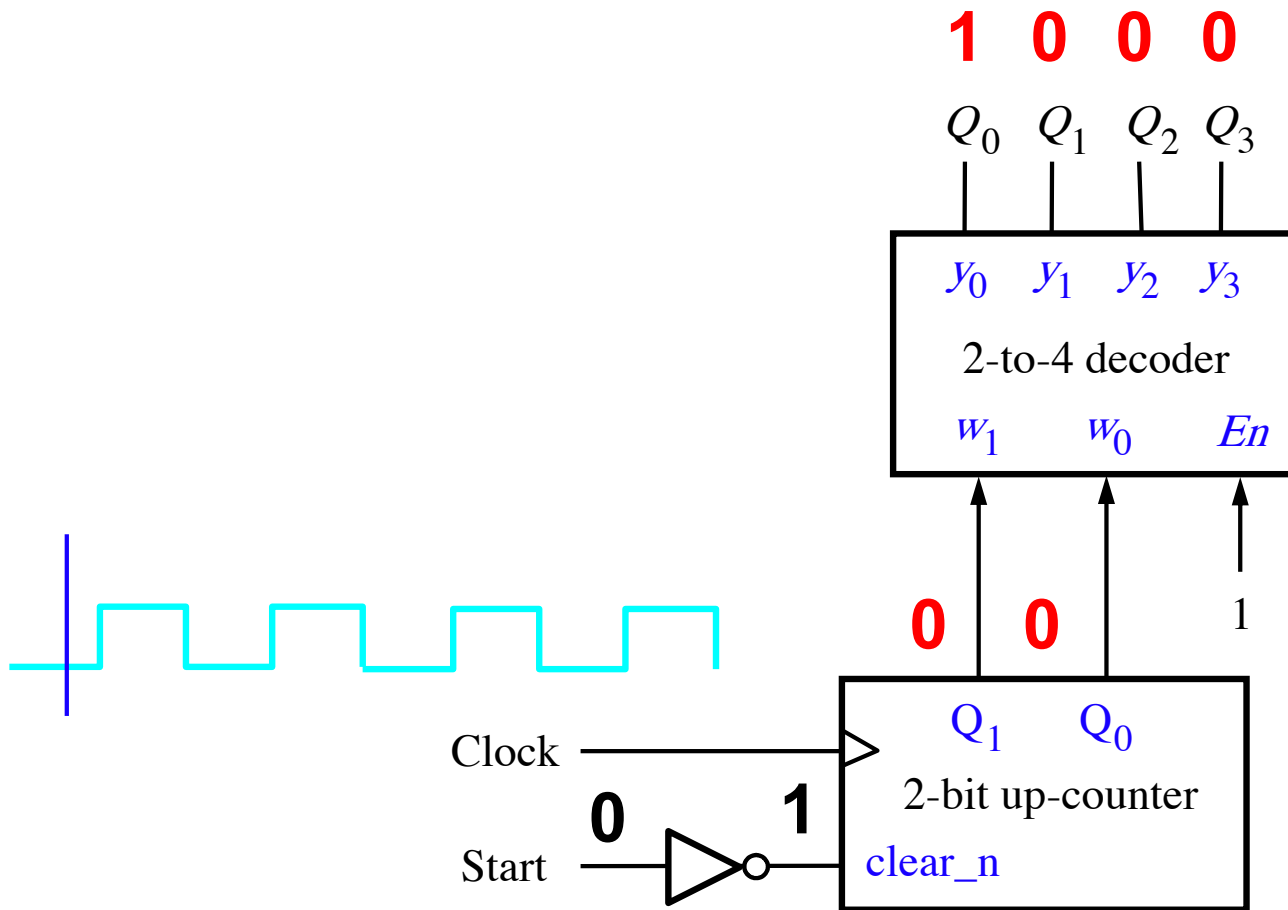
This sets the outputs of the decoder to the start of the counting sequence.

# How Does It Work?

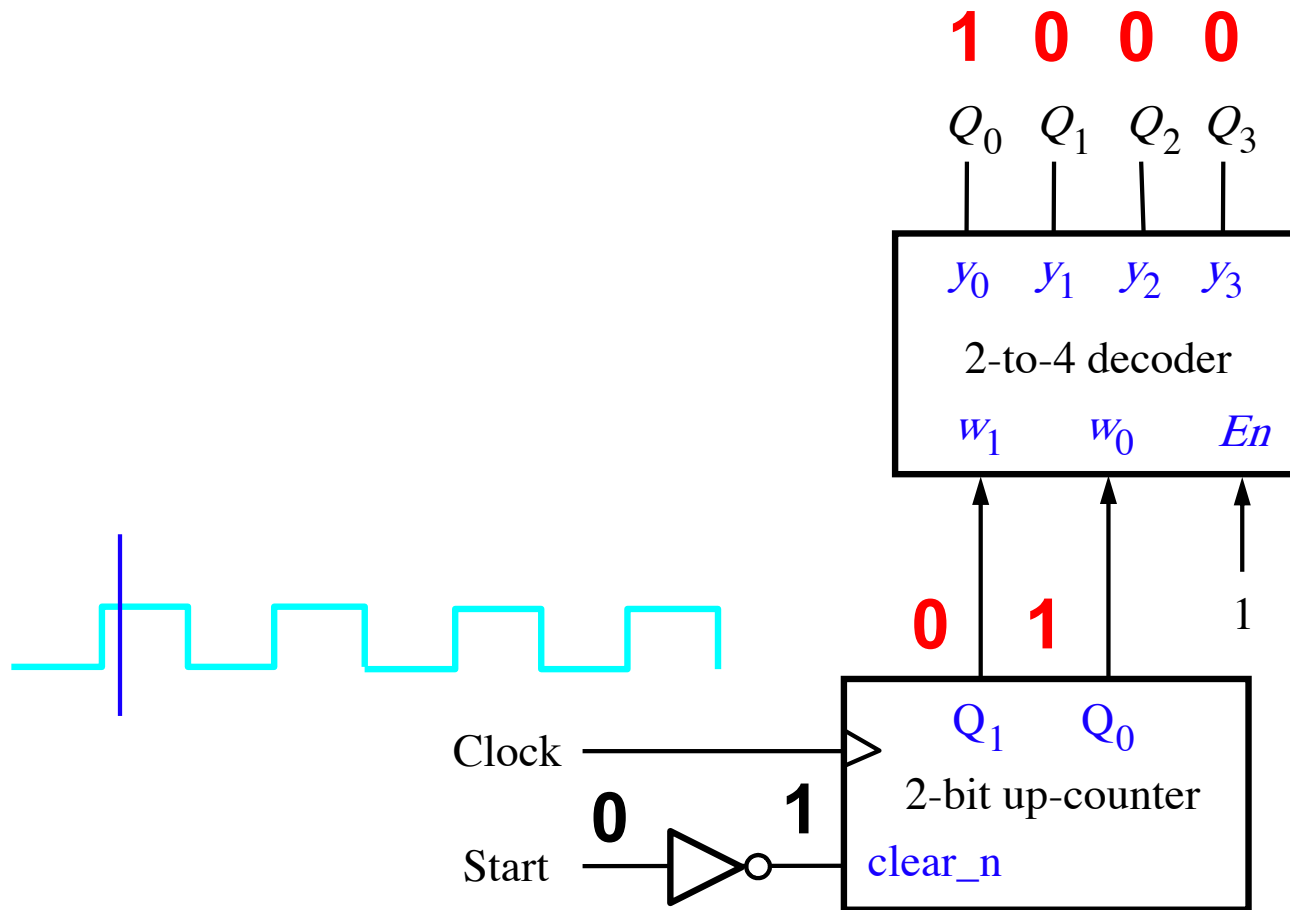


When Start is equal to 0, clear\_n has no effect.

# How Does It Work?



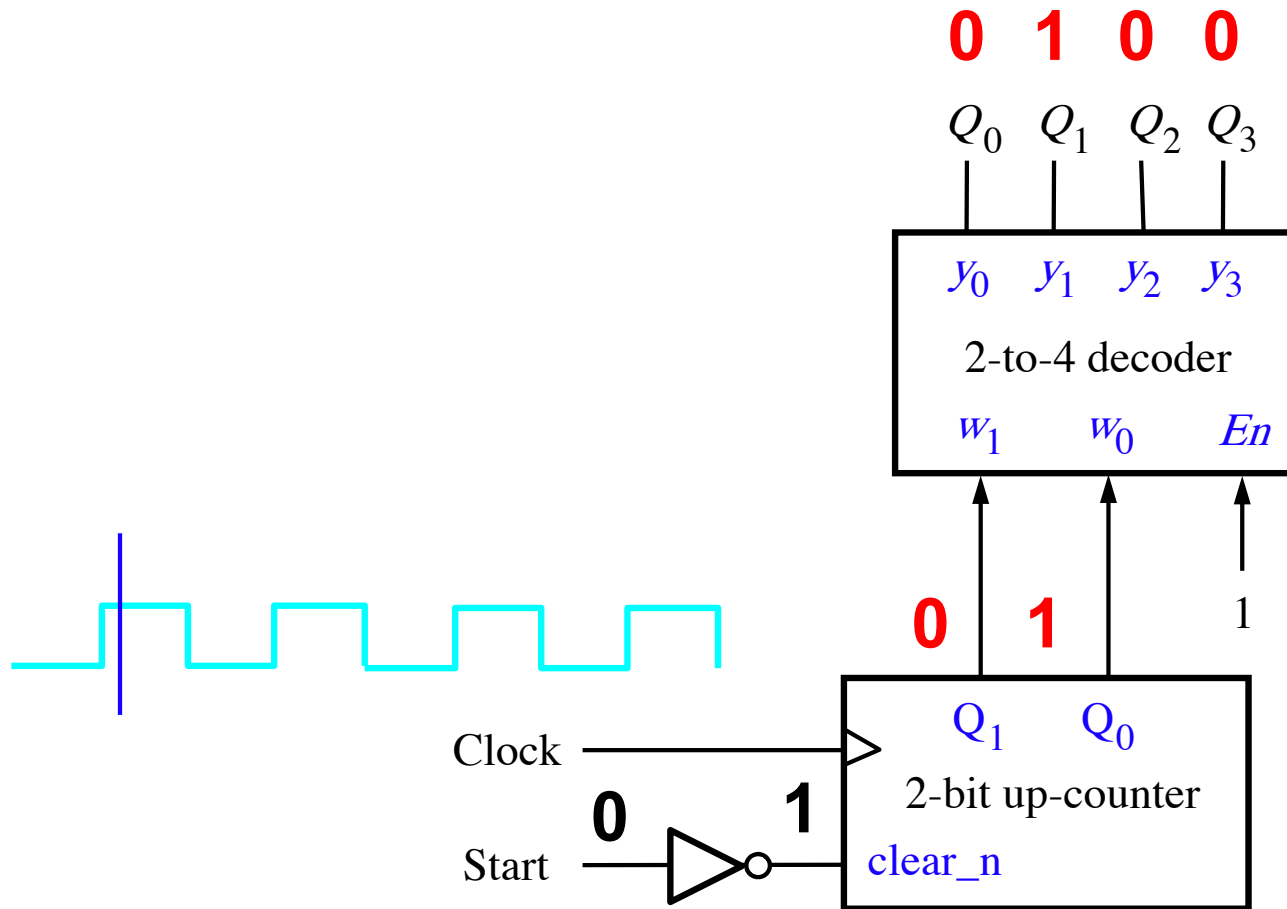
# How Does It Work?



The counter increments the count on the positive clock edge ...

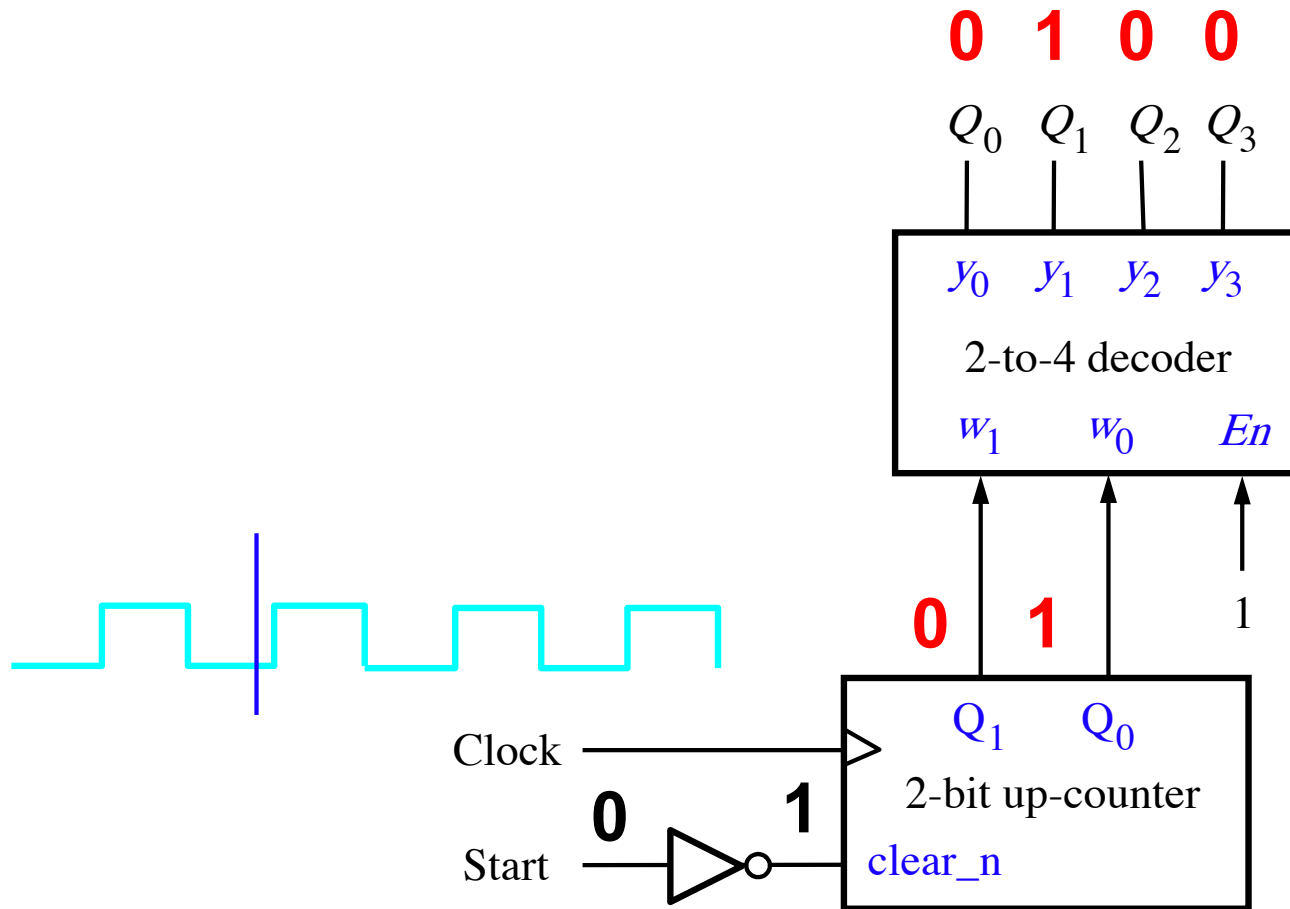


# How Does It Work?

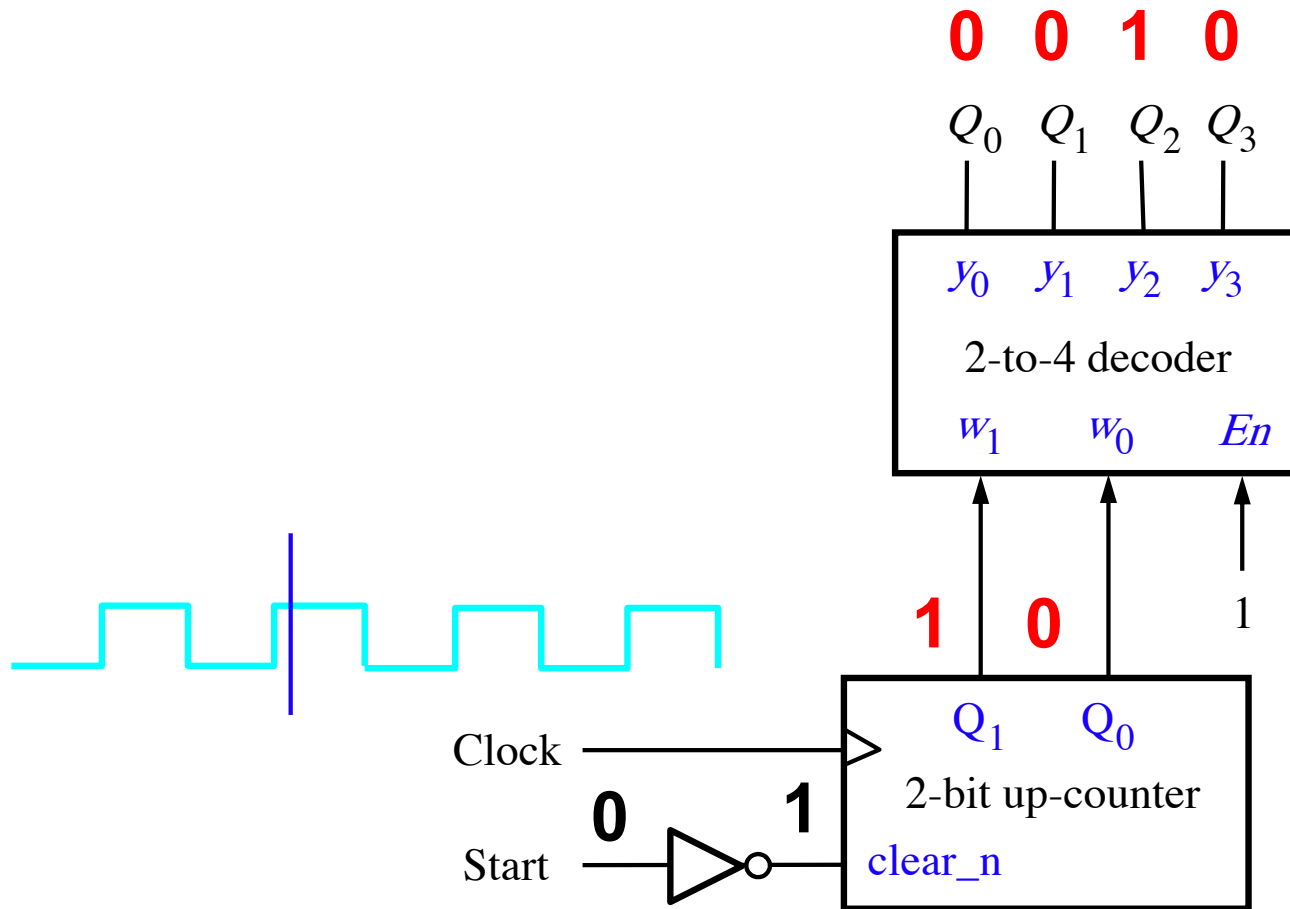


... this updates the outputs of the decoder (which are one hot encoded).

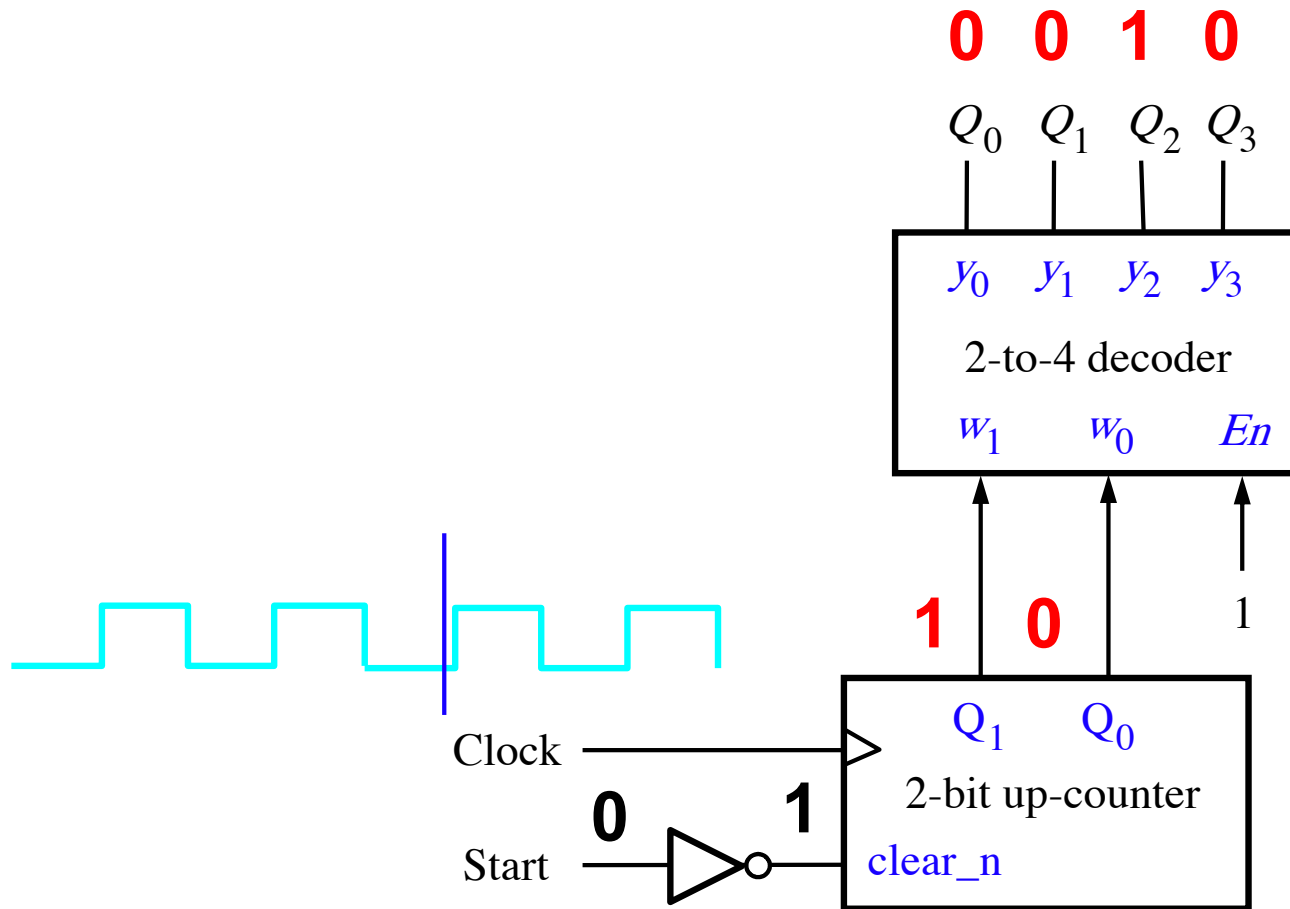
# How Does It Work?



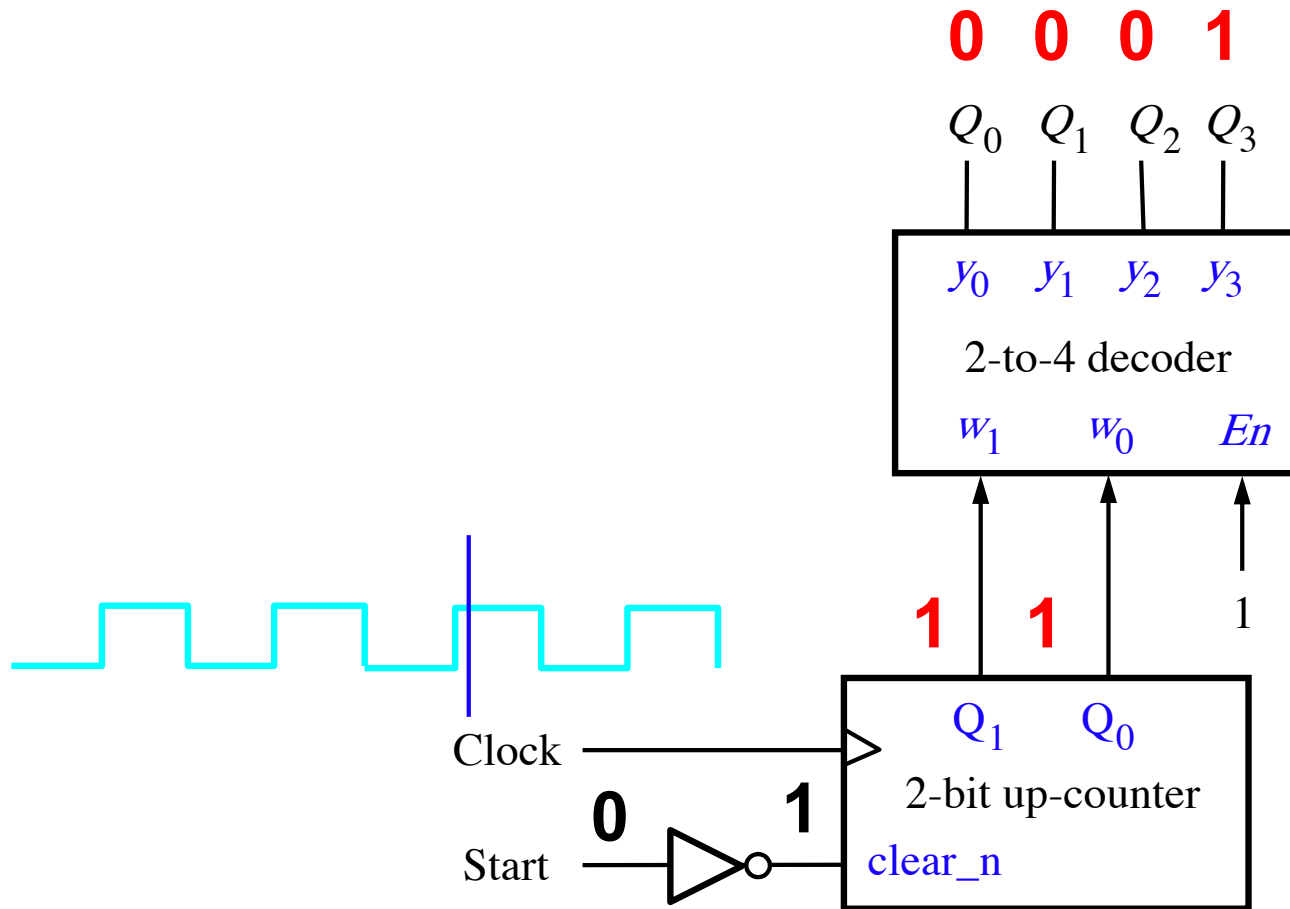
# How Does It Work?



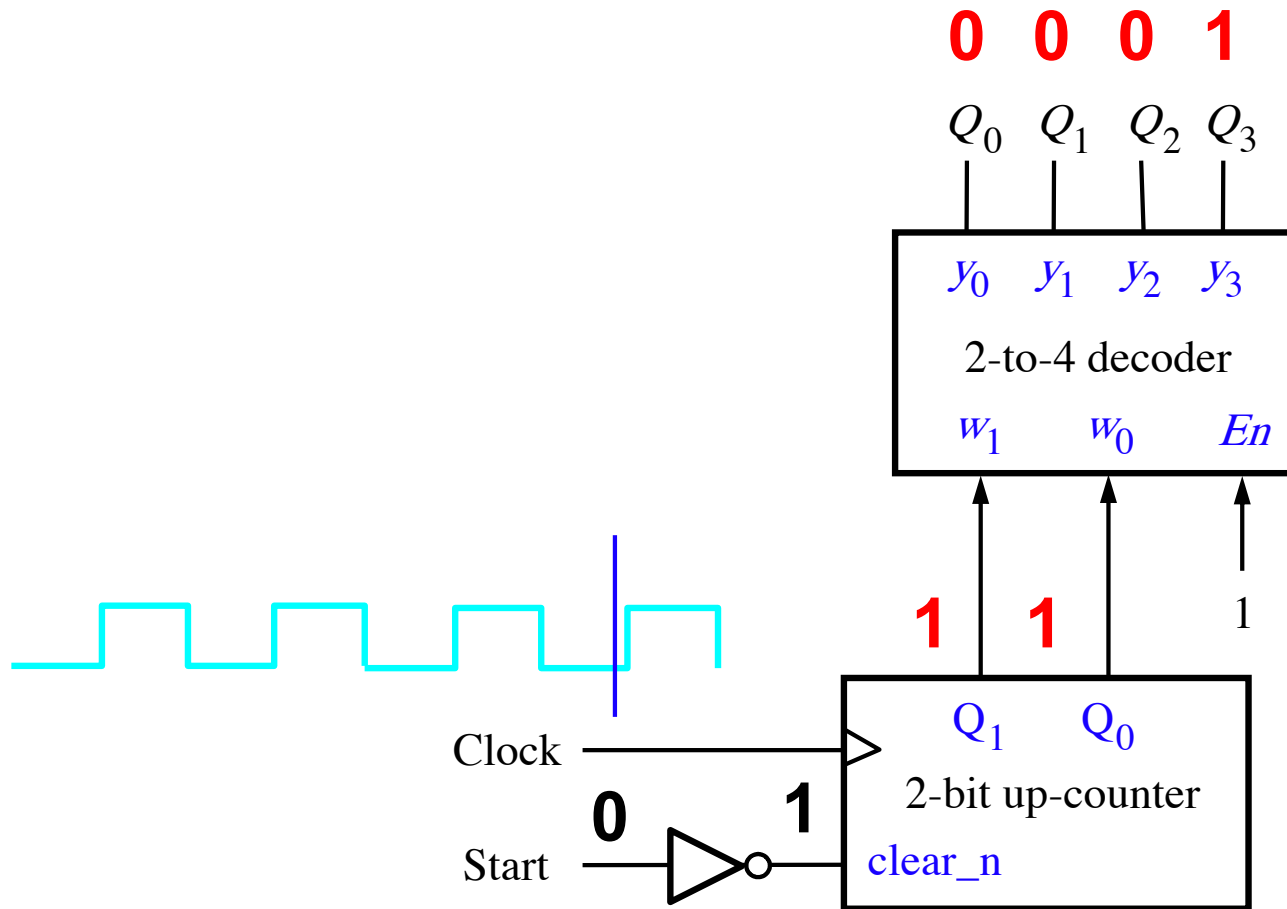
# How Does It Work?



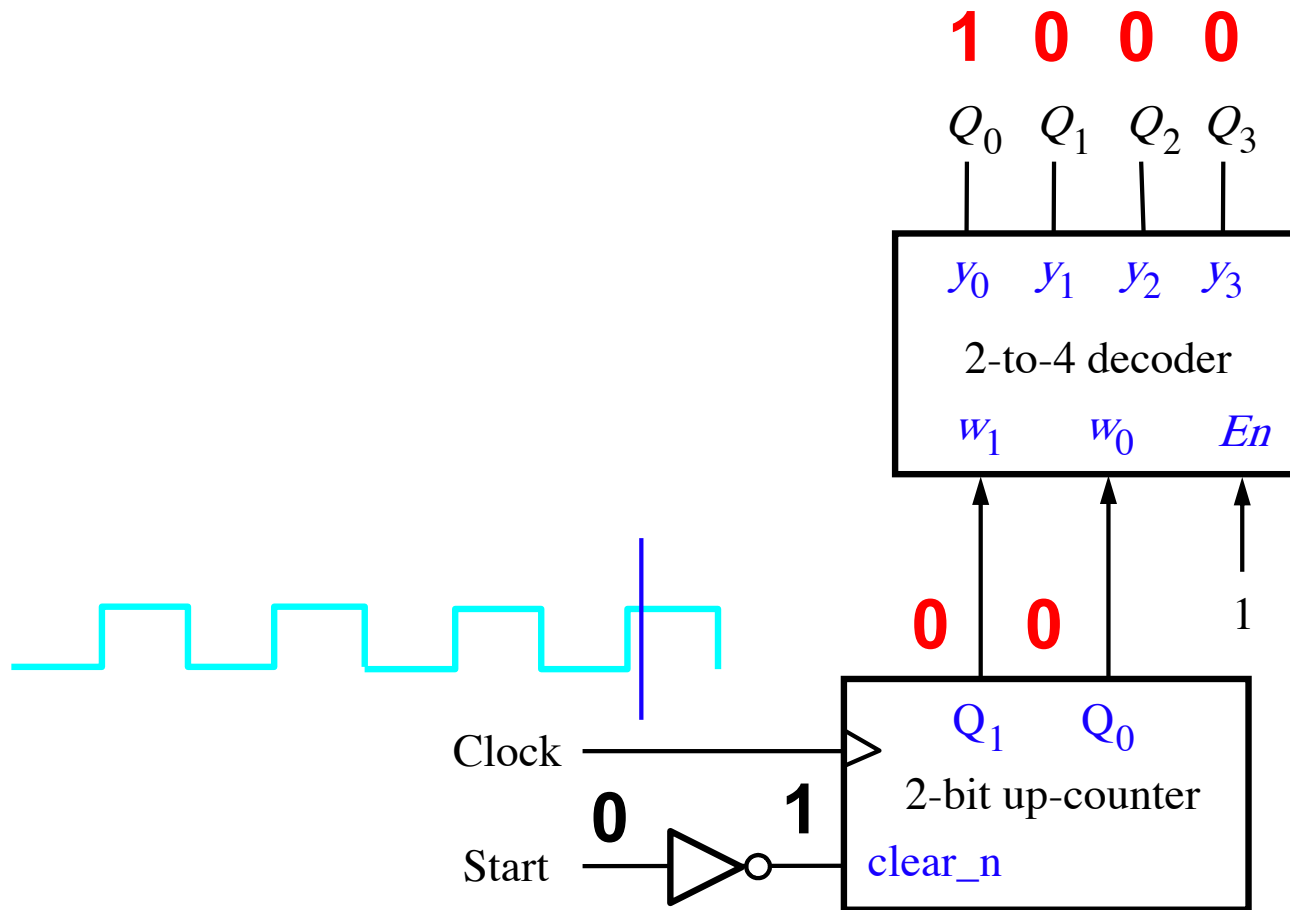
# How Does It Work?



# How Does It Work?



# How Does It Work?

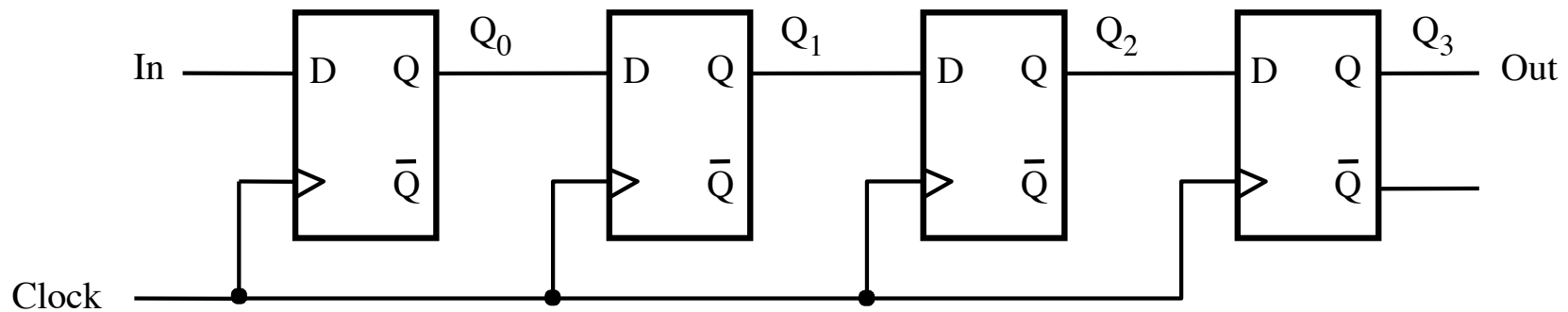


It is back to the start of the counting sequence,  
which is: 1000, 0100, 0010, 0001.

# **Johnson Counter**

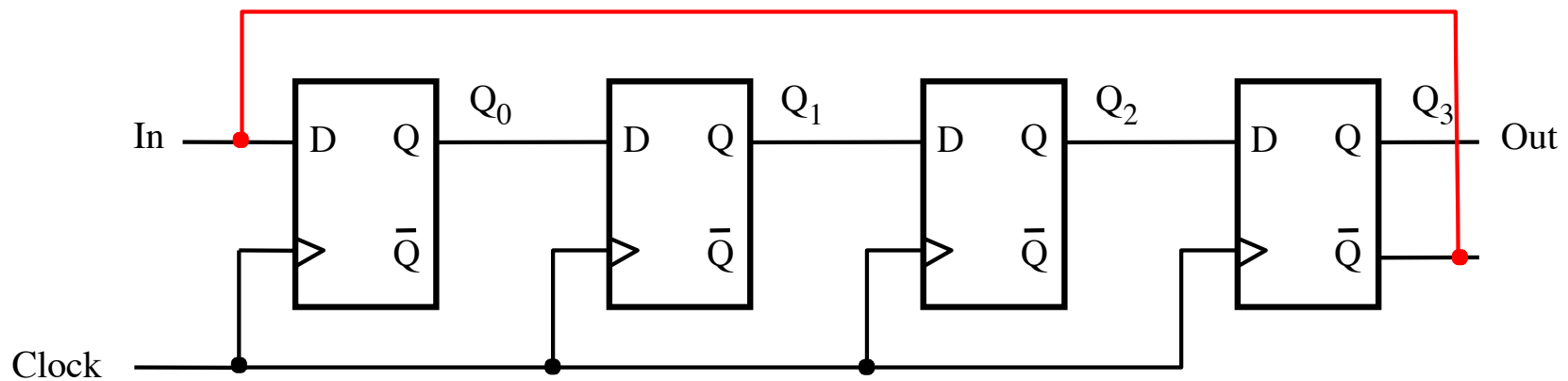


# How to build a 4-bit Johnson counter



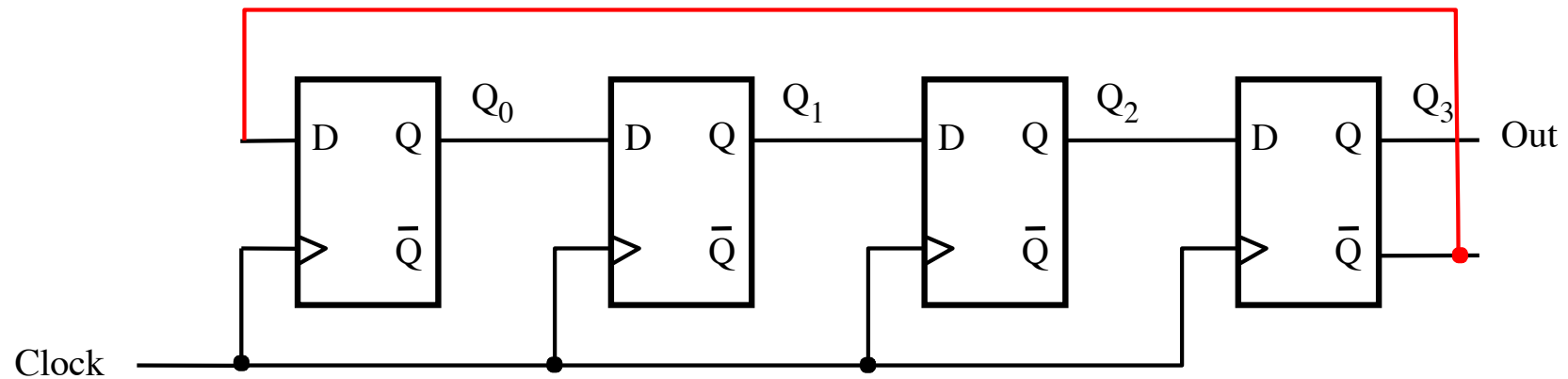
To build a Johnson counter start with a shift register.

# How to build a 4-bit Johnson counter



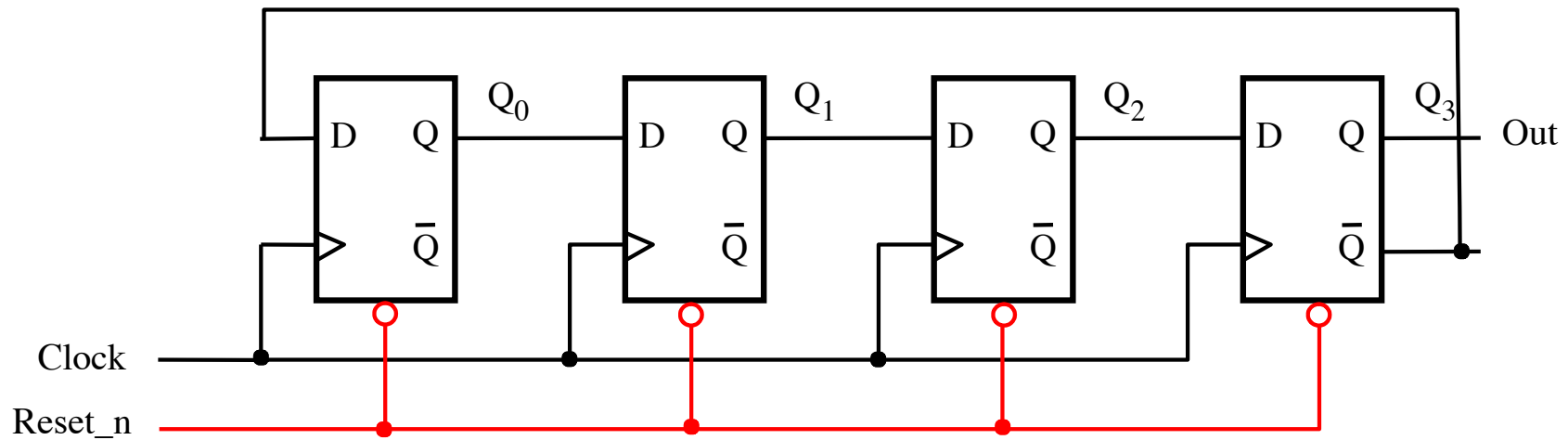
Next, add a loop from the  $\bar{Q}$  output of the last flip-flop to the first...

# How to build a 4-bit Johnson counter



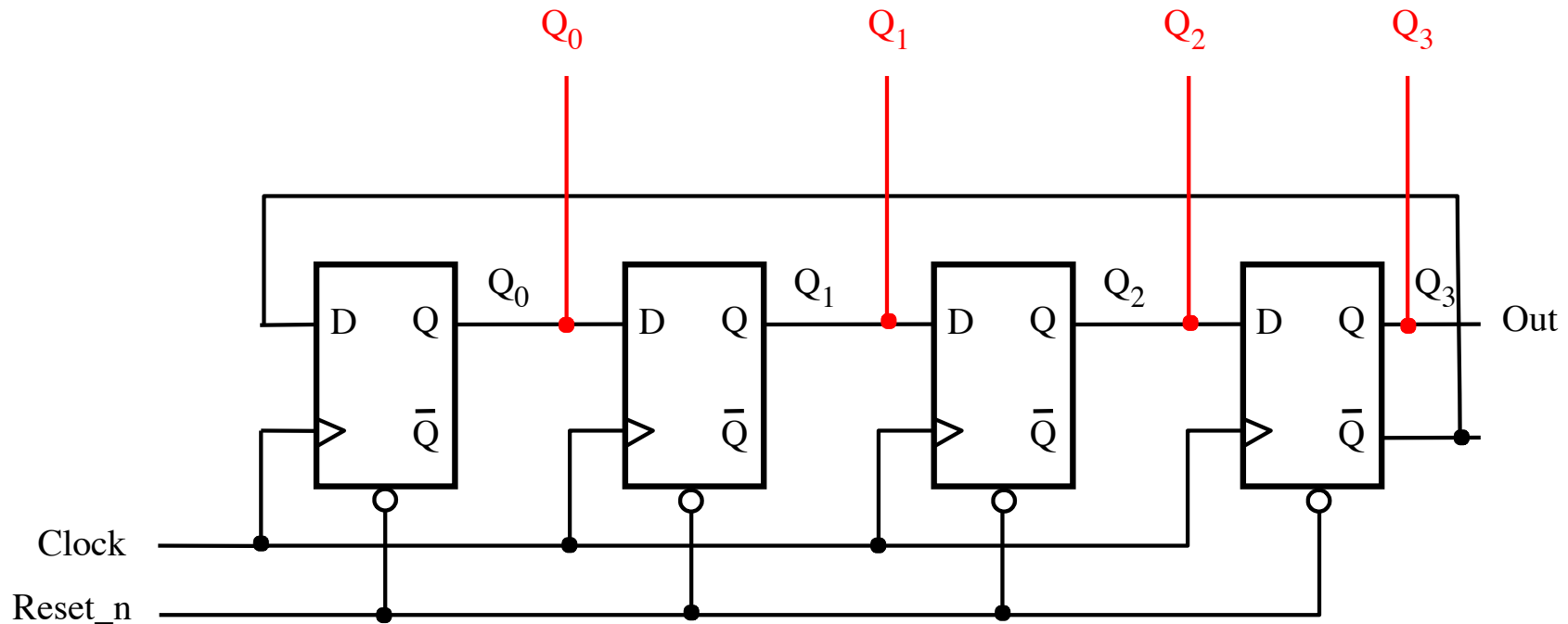
... and remove the In input line.

# How to build a 4-bit Johnson counter



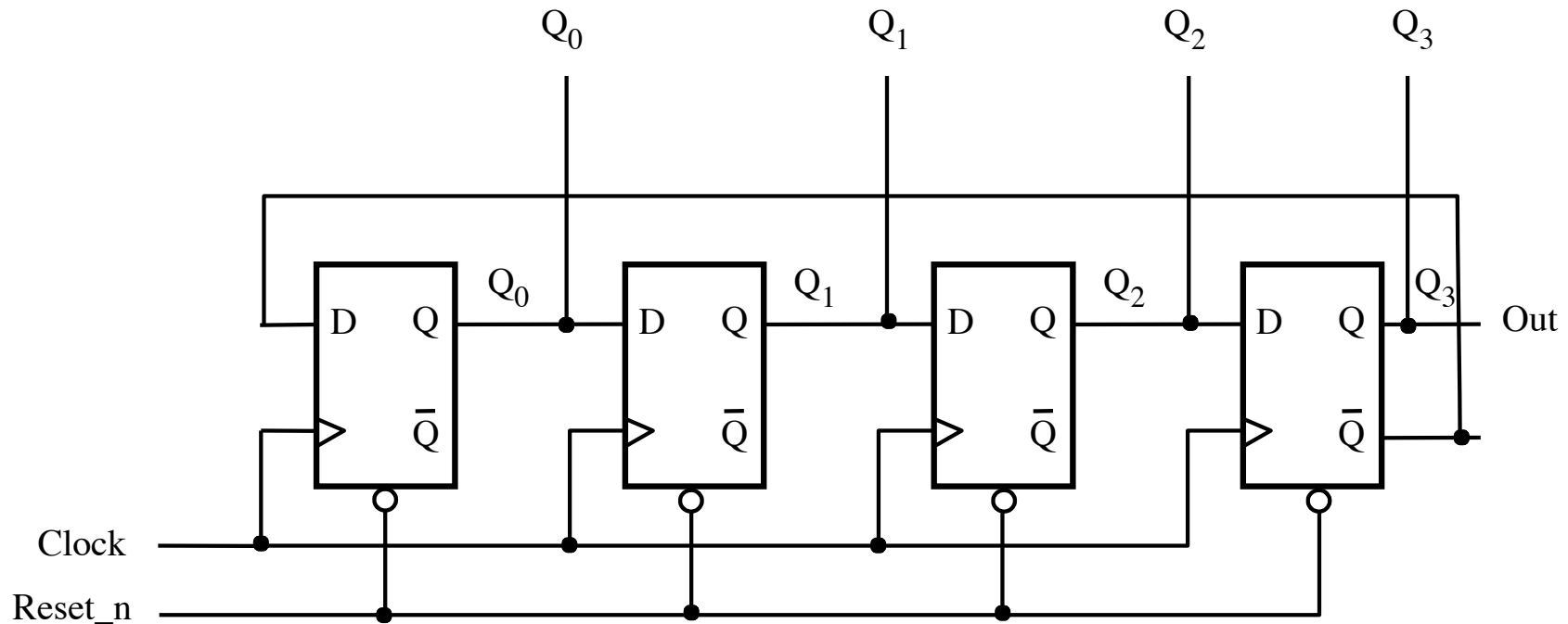
Also, add a Reset\_n line that goes to clear\_n of all flip-flops.

# How to build a 4-bit Johnson counter



Finally, extend the output lines that form the count number.

# How to build a 4-bit Johnson counter

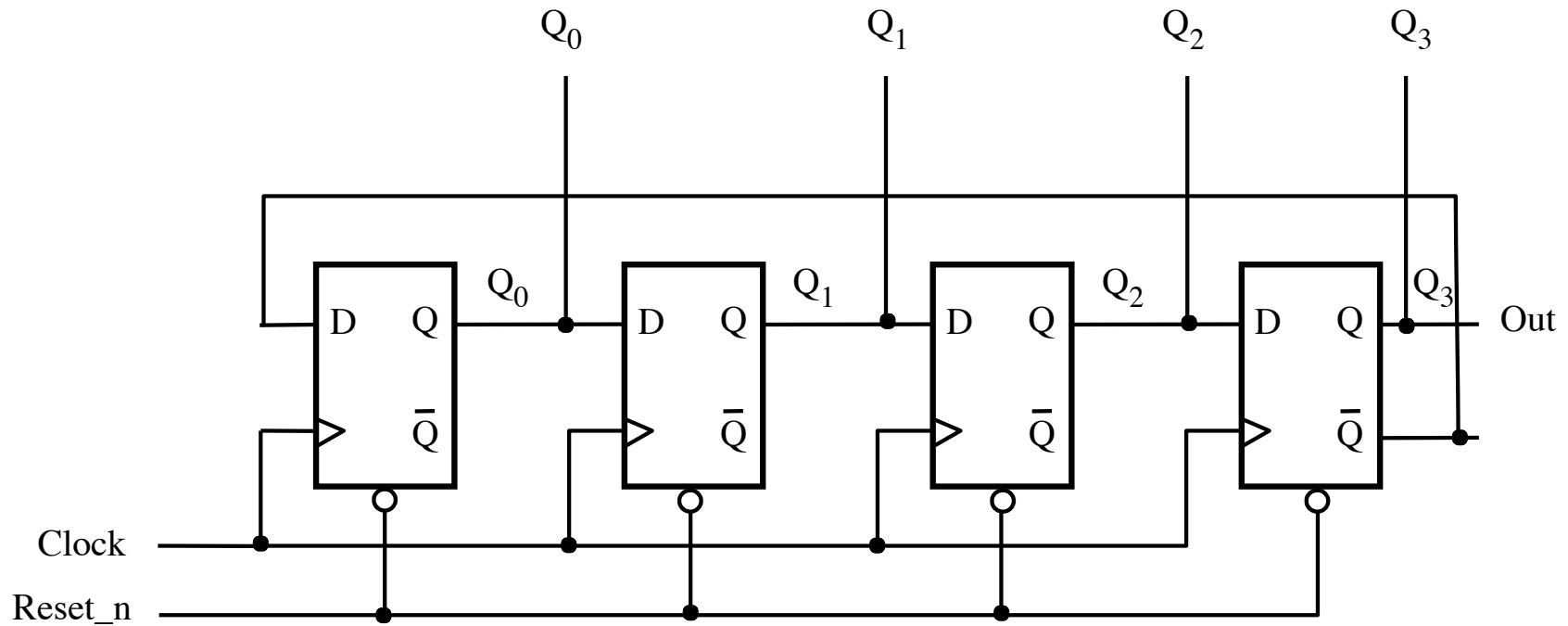


This is the final circuit diagram.

# 4-Bit Johnson Counter

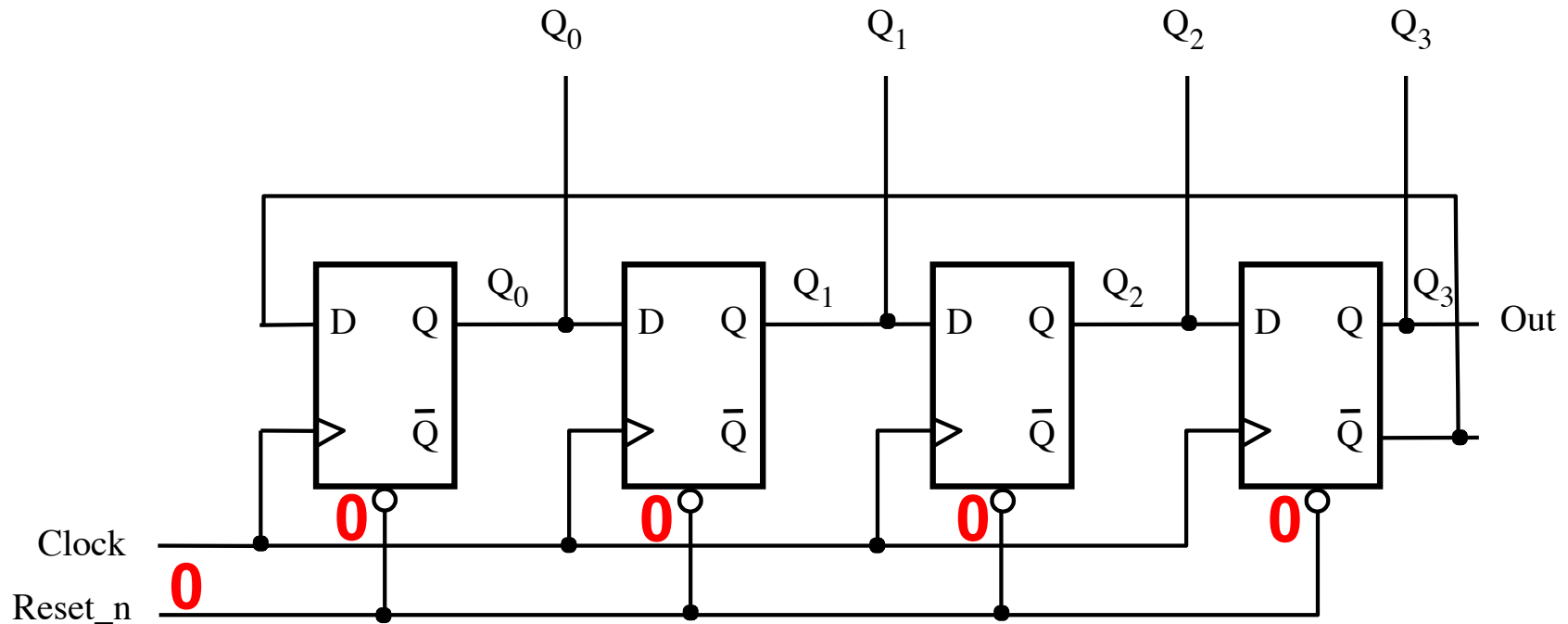
- Only 1-bit changes at a time
- Start with a reset of all flip-flops
- The counting sequence is:  
0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000
- An n-bit Johnson counter has a counting sequence of length  $2n$

# Initialization: How does it work?



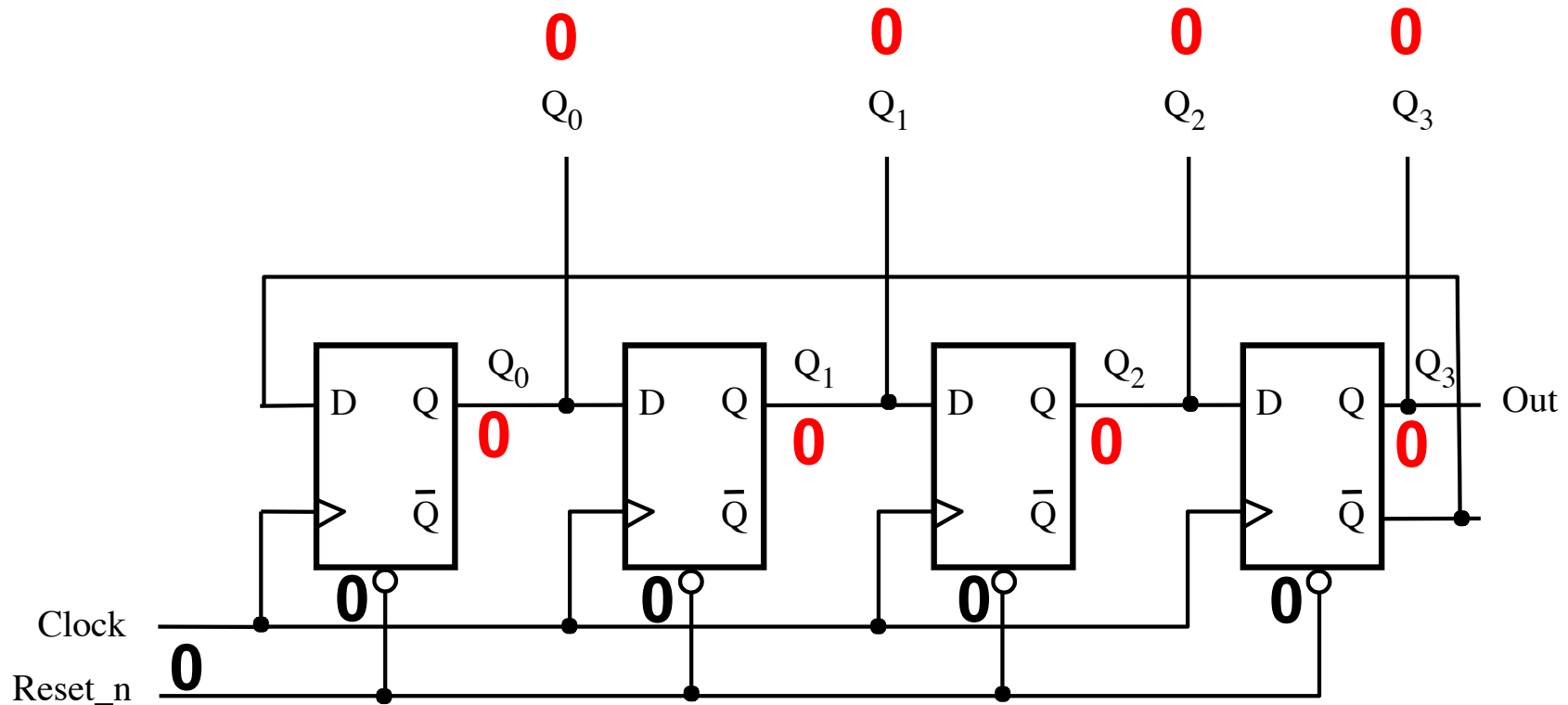


# Initialization: How does it work?



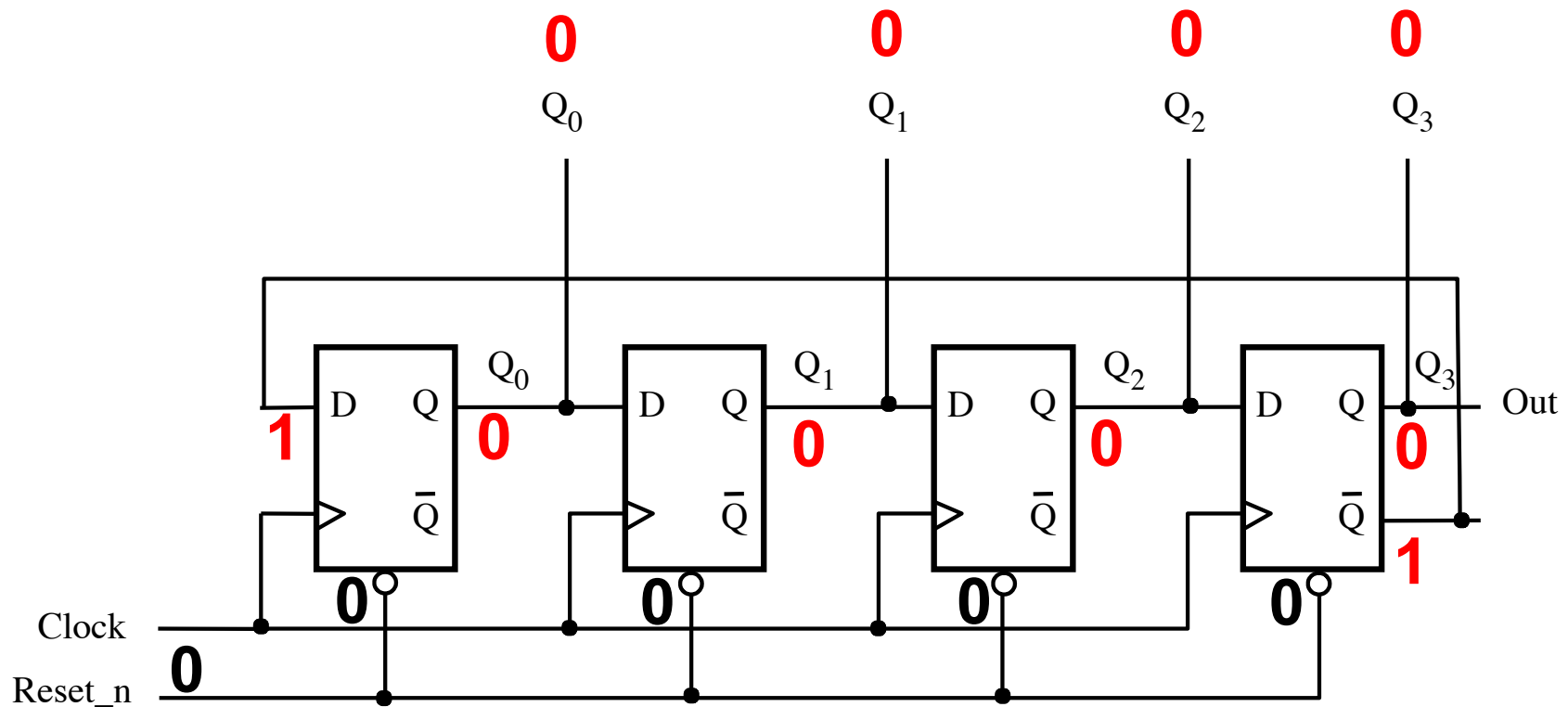
To initialize the Johnson counter set  $Reset\_n$  to 0 ...

# Initialization: How does it work?



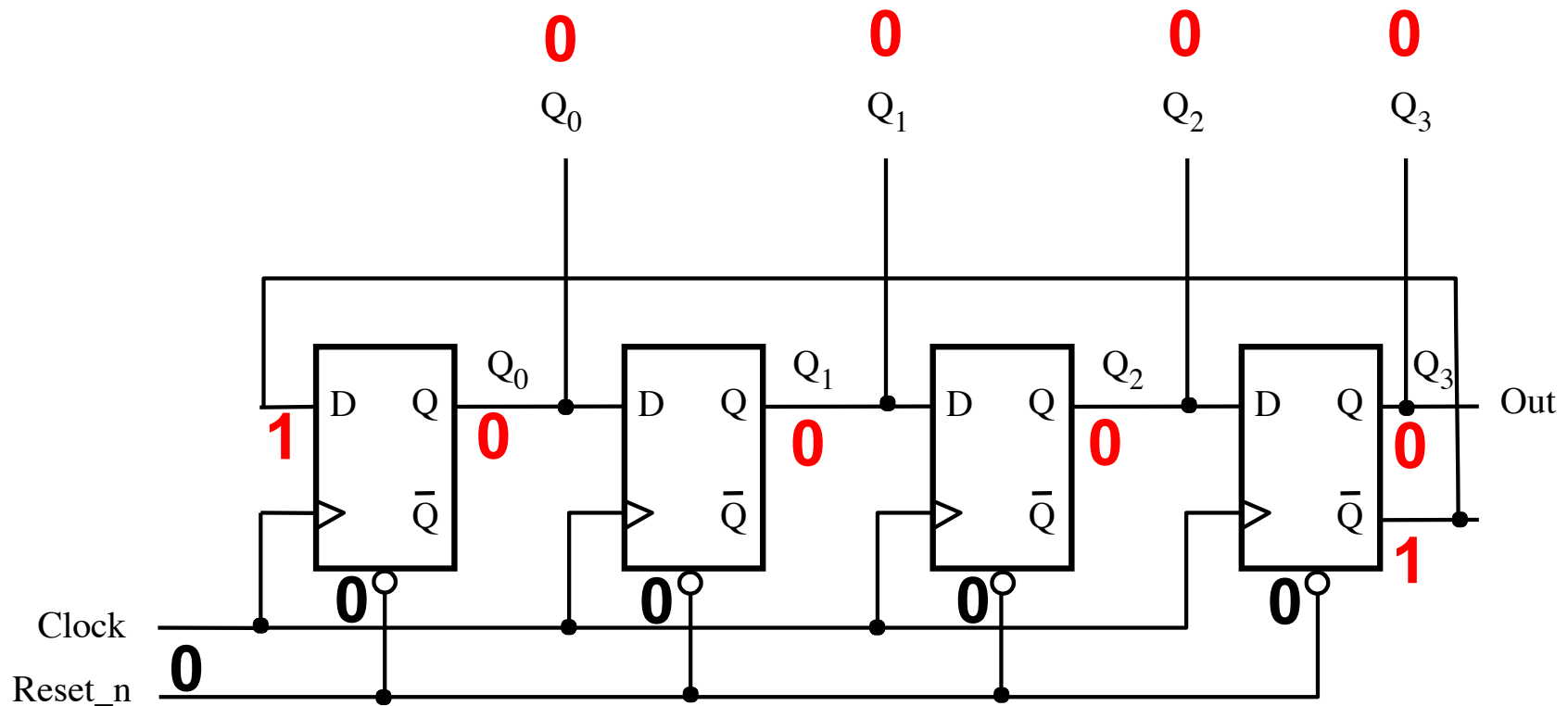
... this zeros the outputs of all flip-flops ...

# Initialization: How does it work?

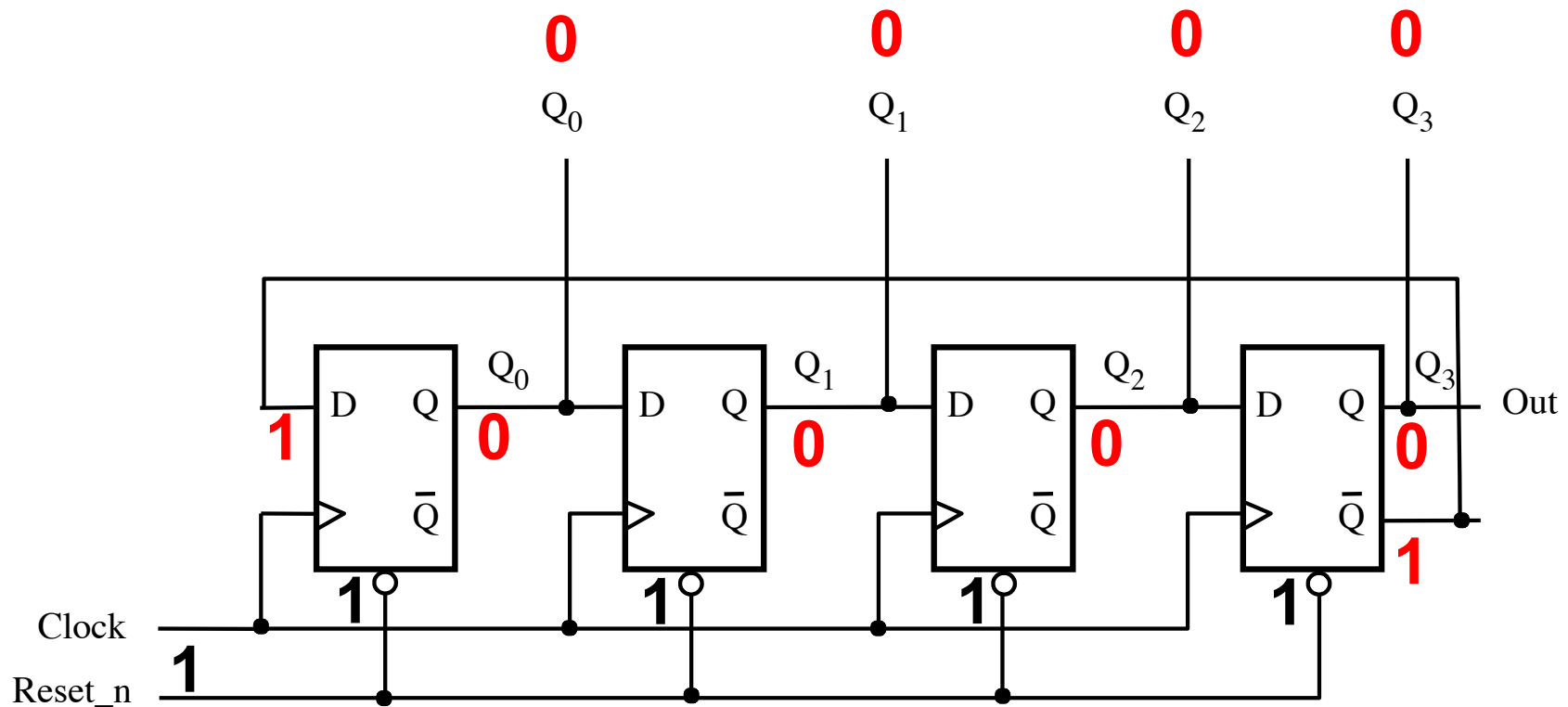


... and also sets the  $\bar{Q}$  output of the last flip-flop to 1.

# Counting: How does it work?

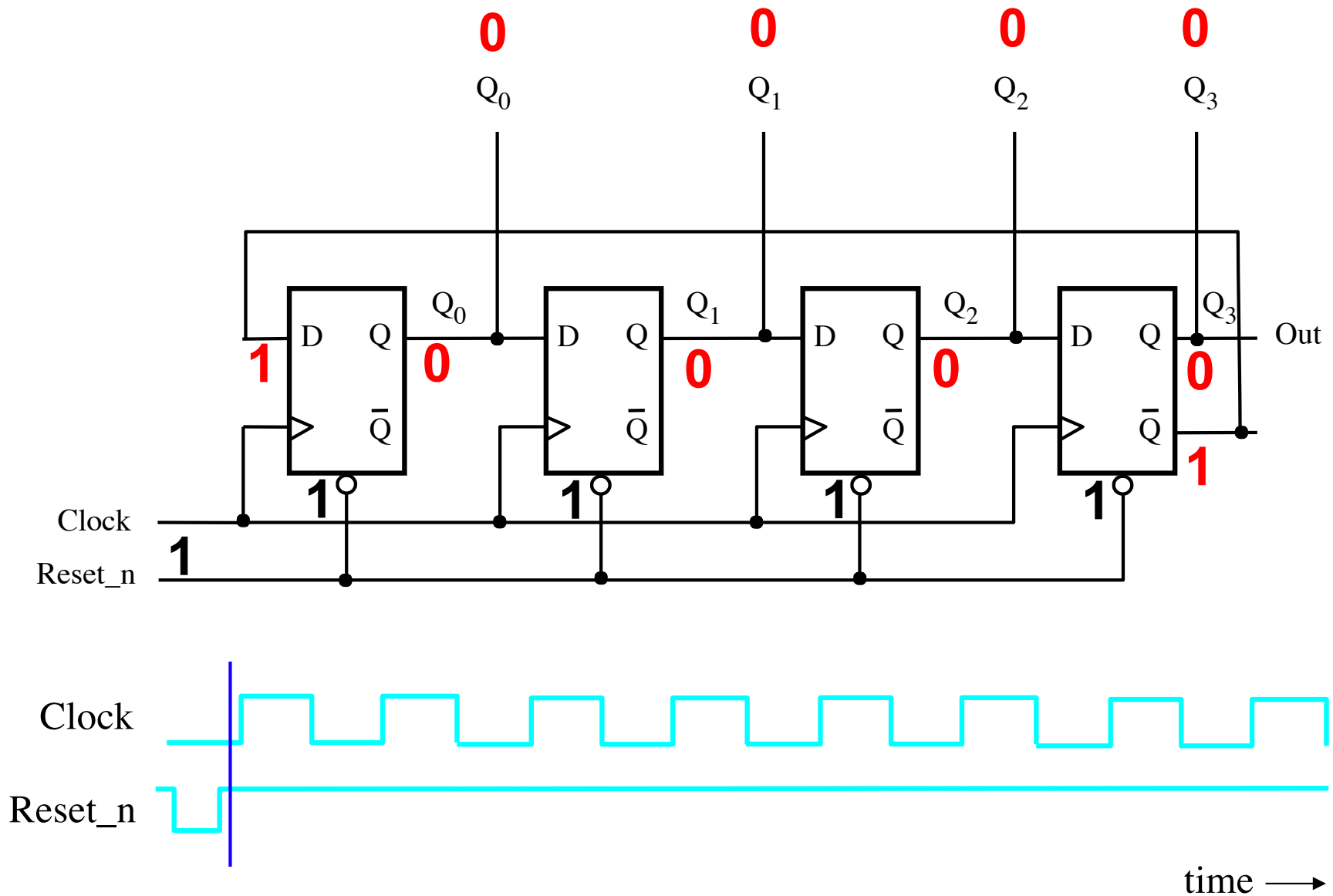


# Counting: How does it work?

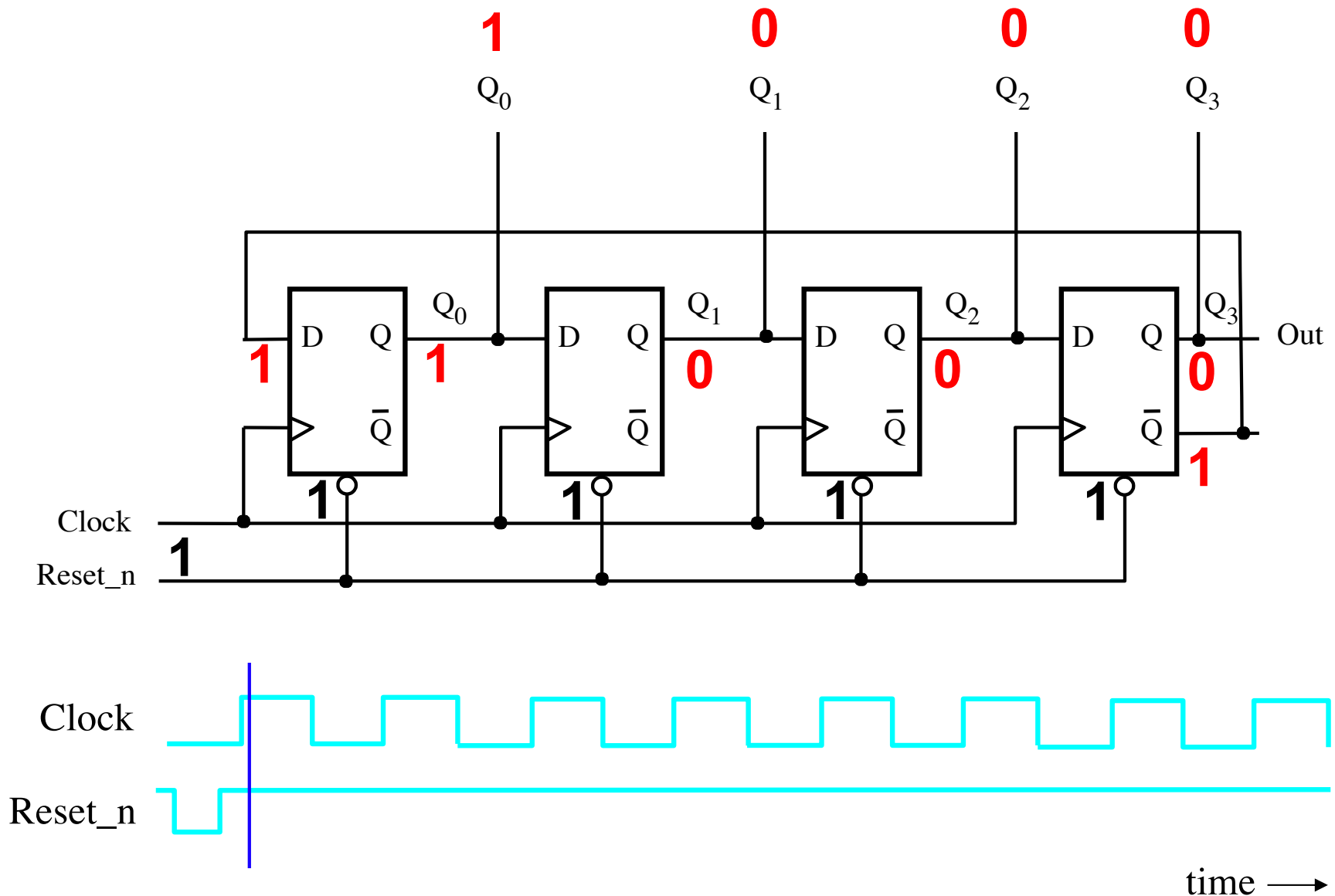


To start counting, Reset\_n needs to be set to 1.

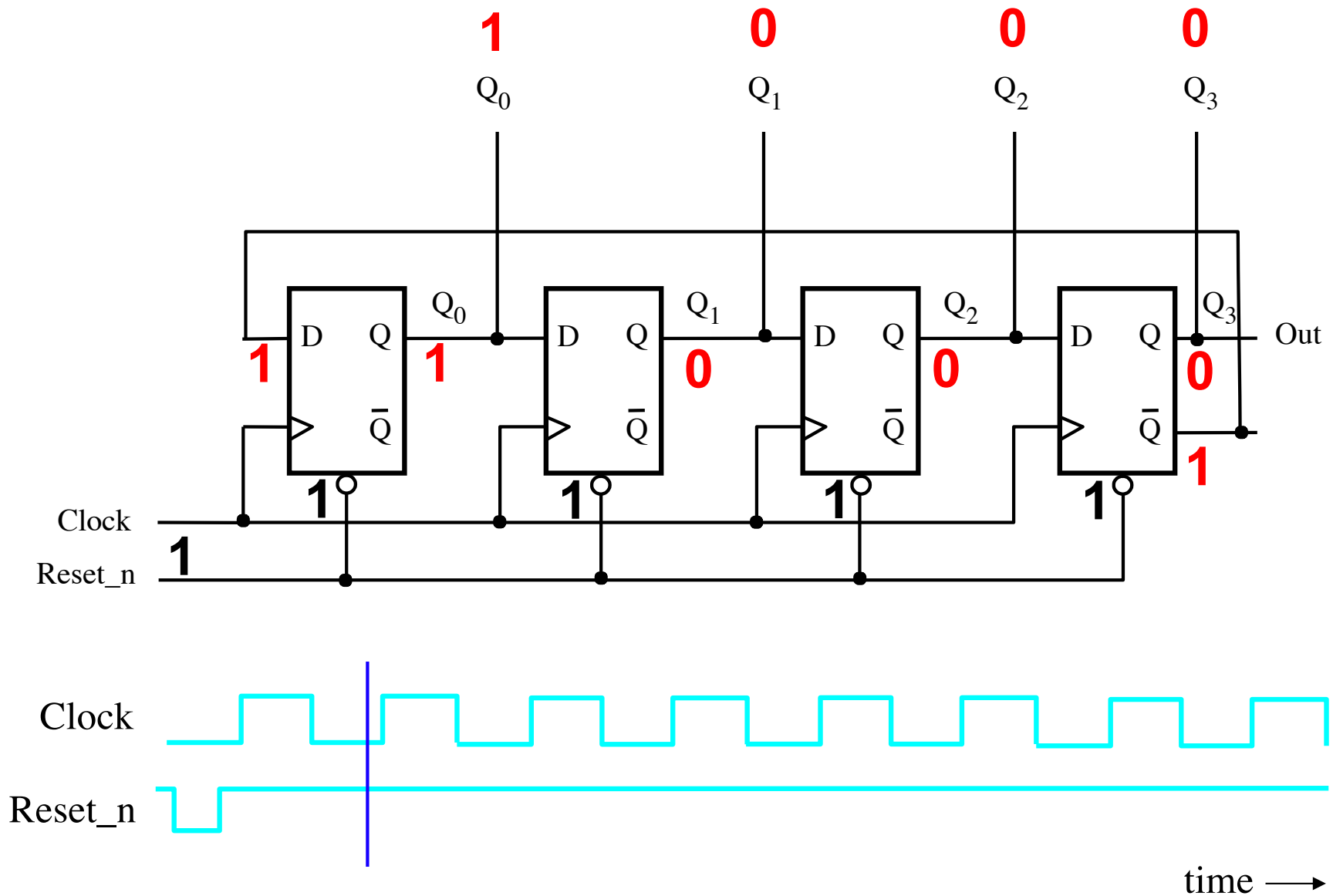
# Counting: How does it work?



# Counting: How does it work?

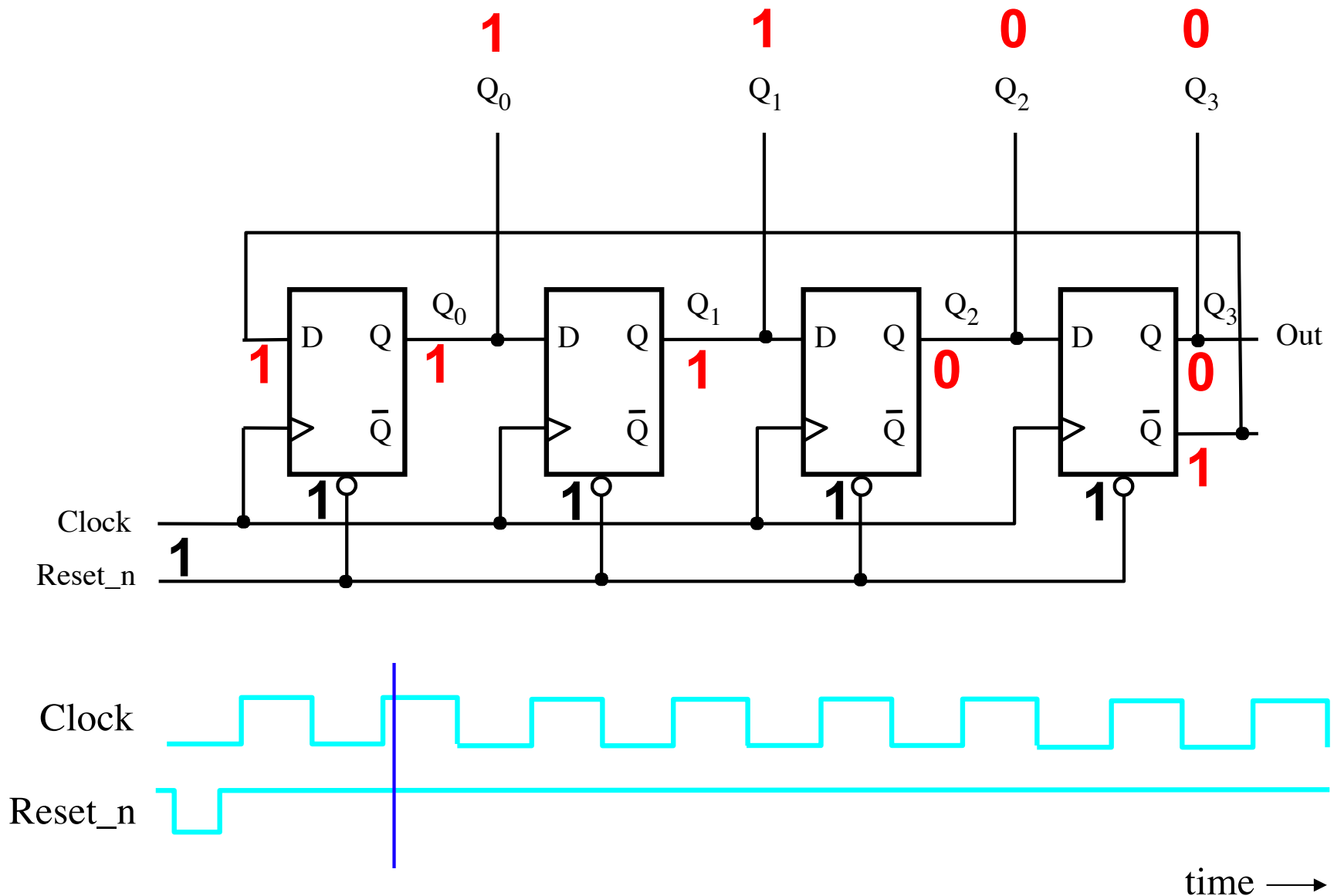


# Counting: How does it work?

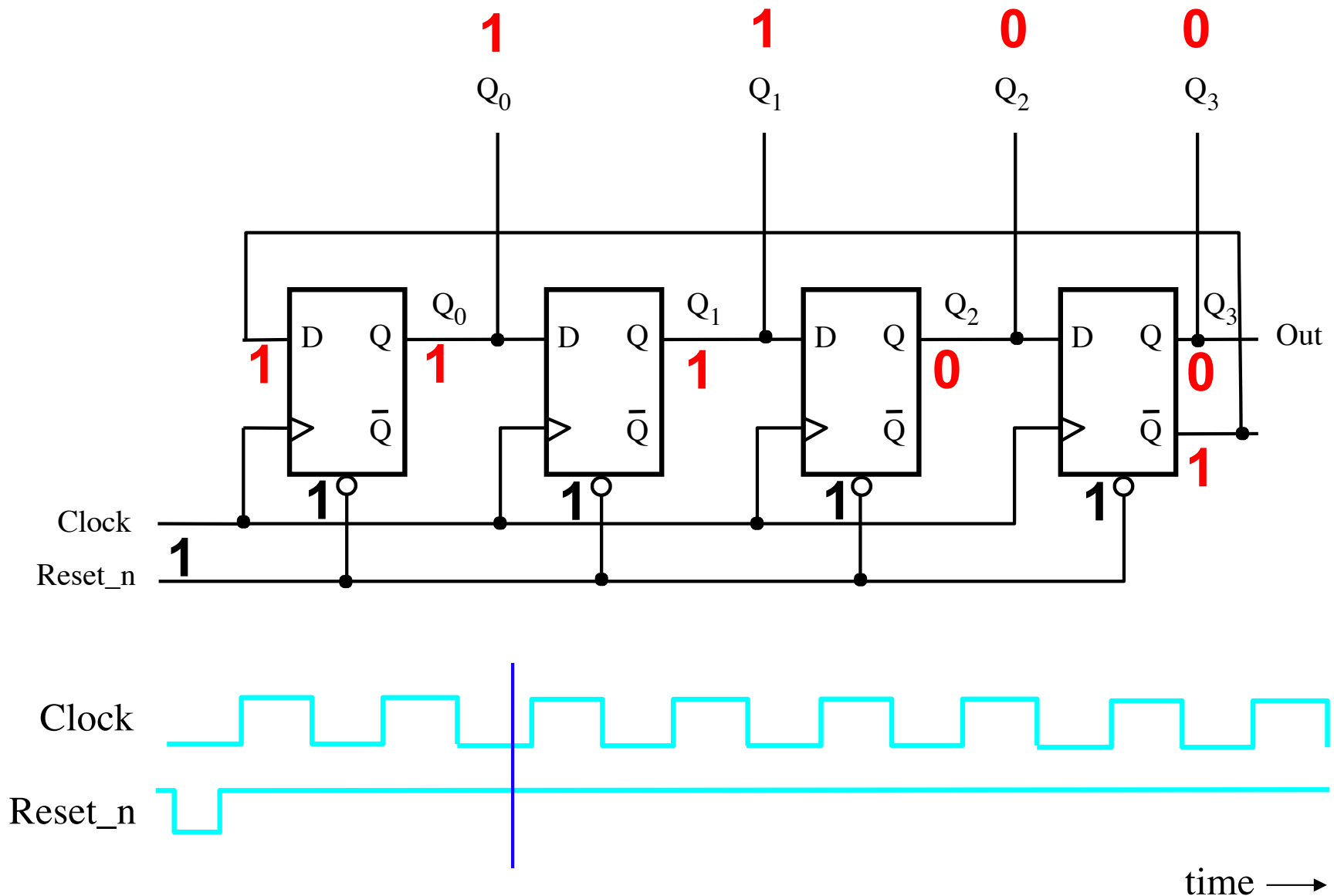




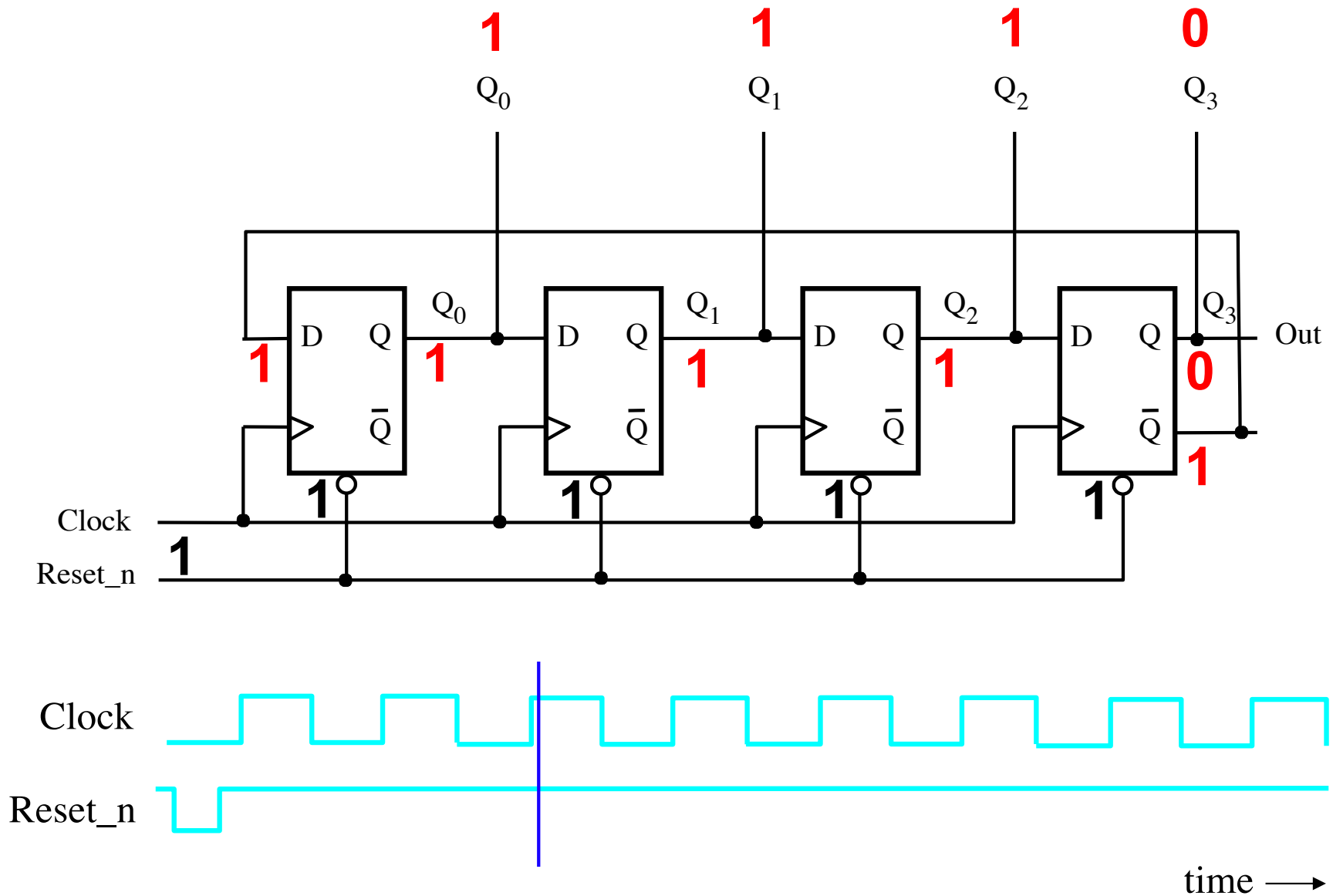
# Counting: How does it work?



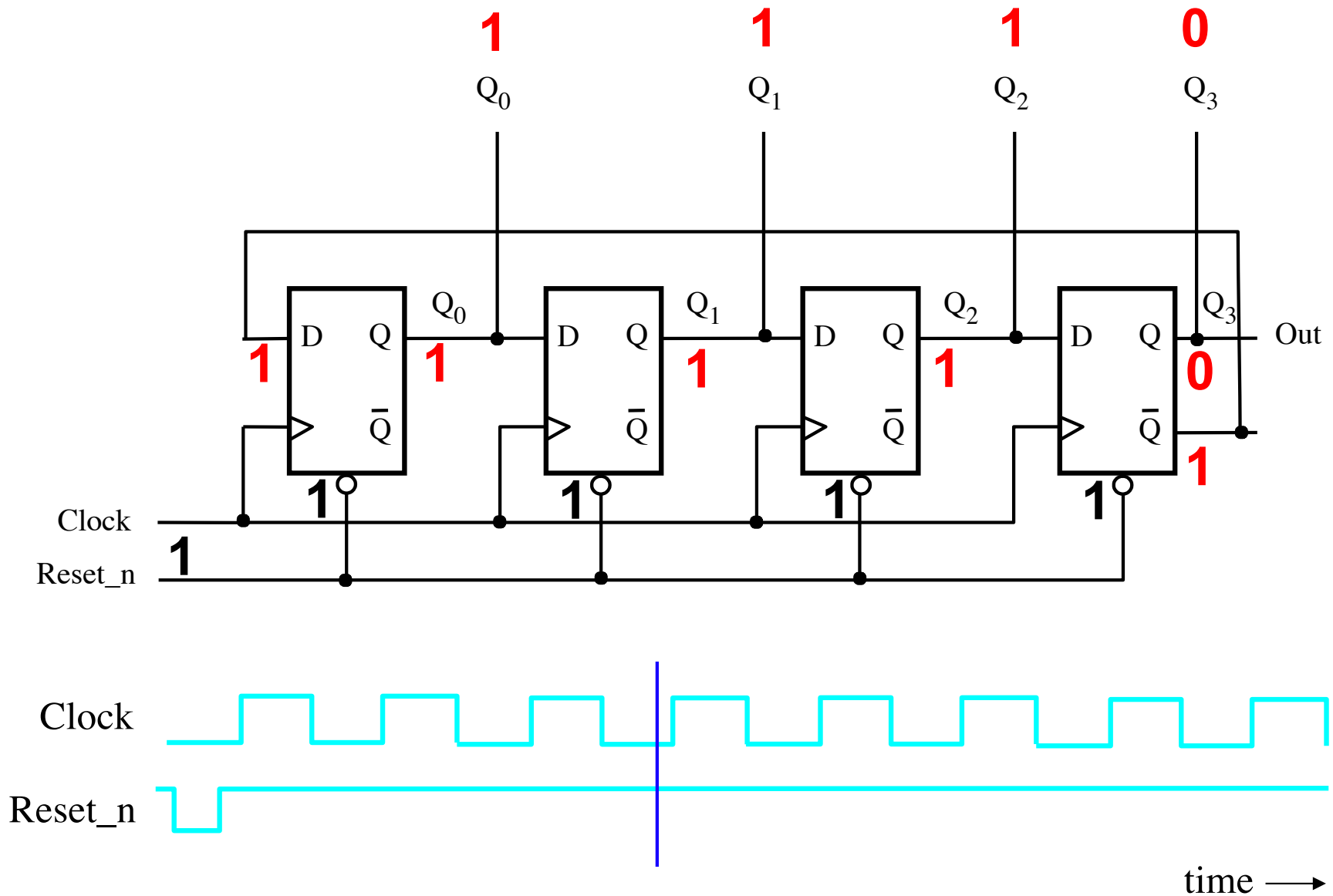
# Counting: How does it work?



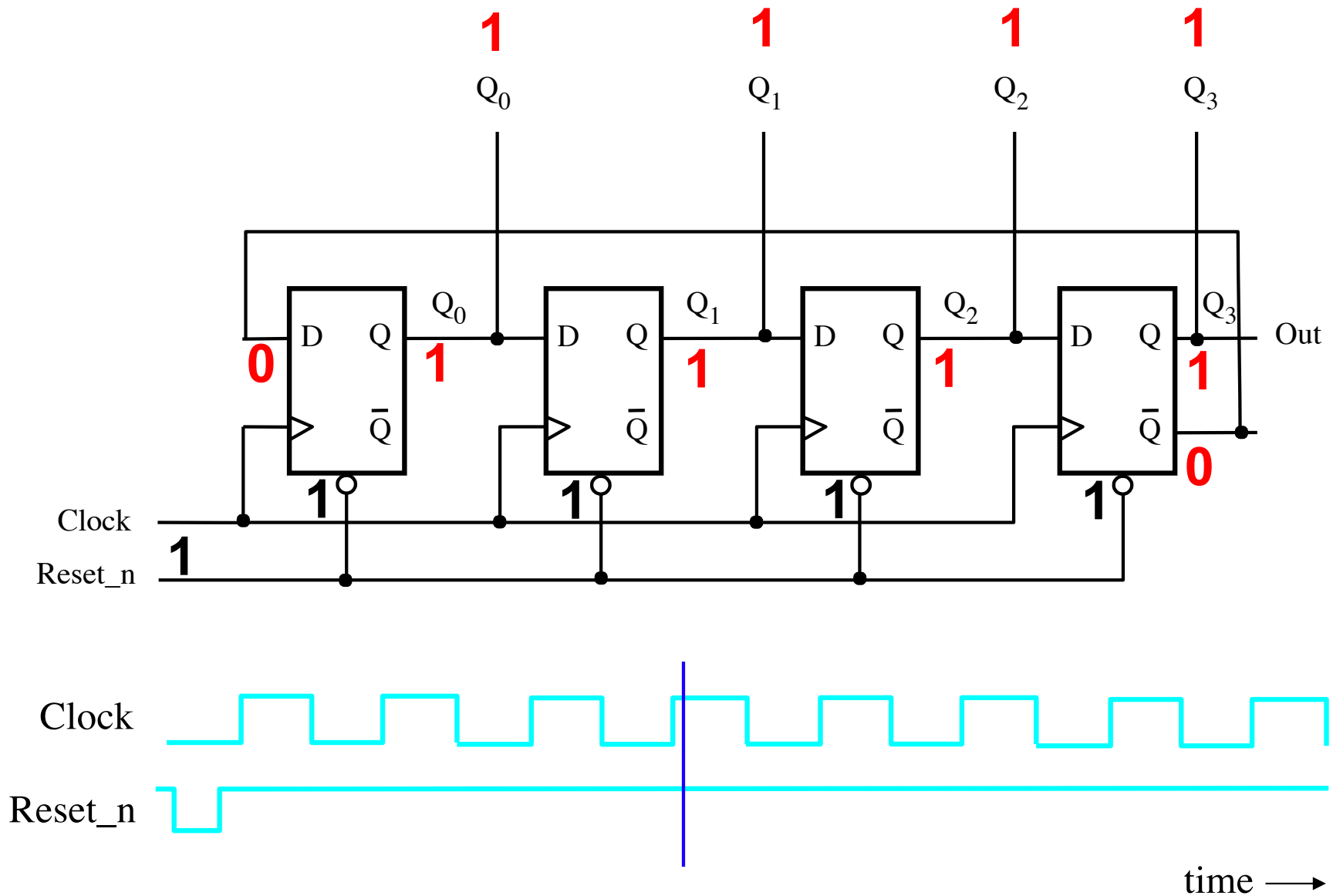
# Counting: How does it work?



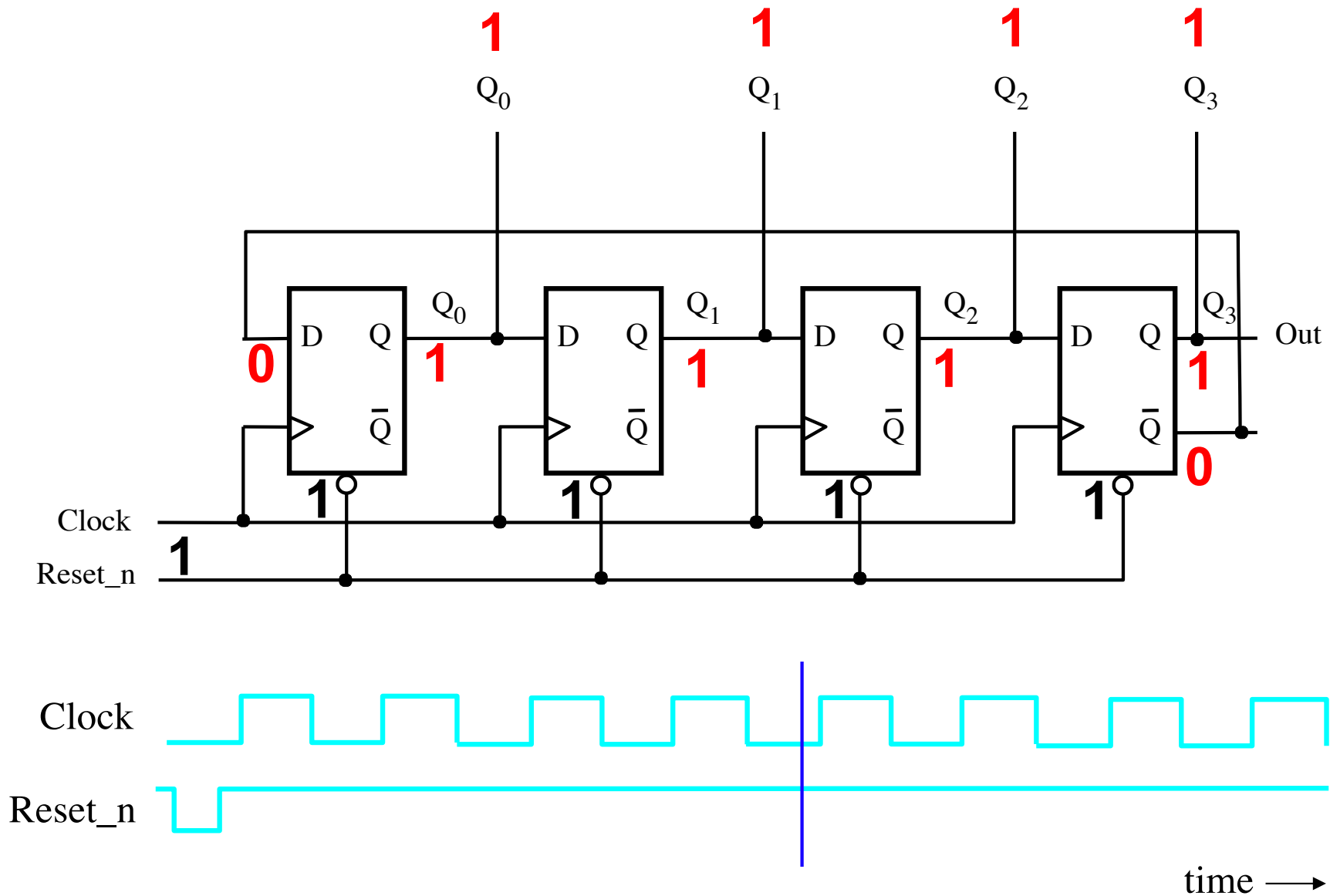
# Counting: How does it work?



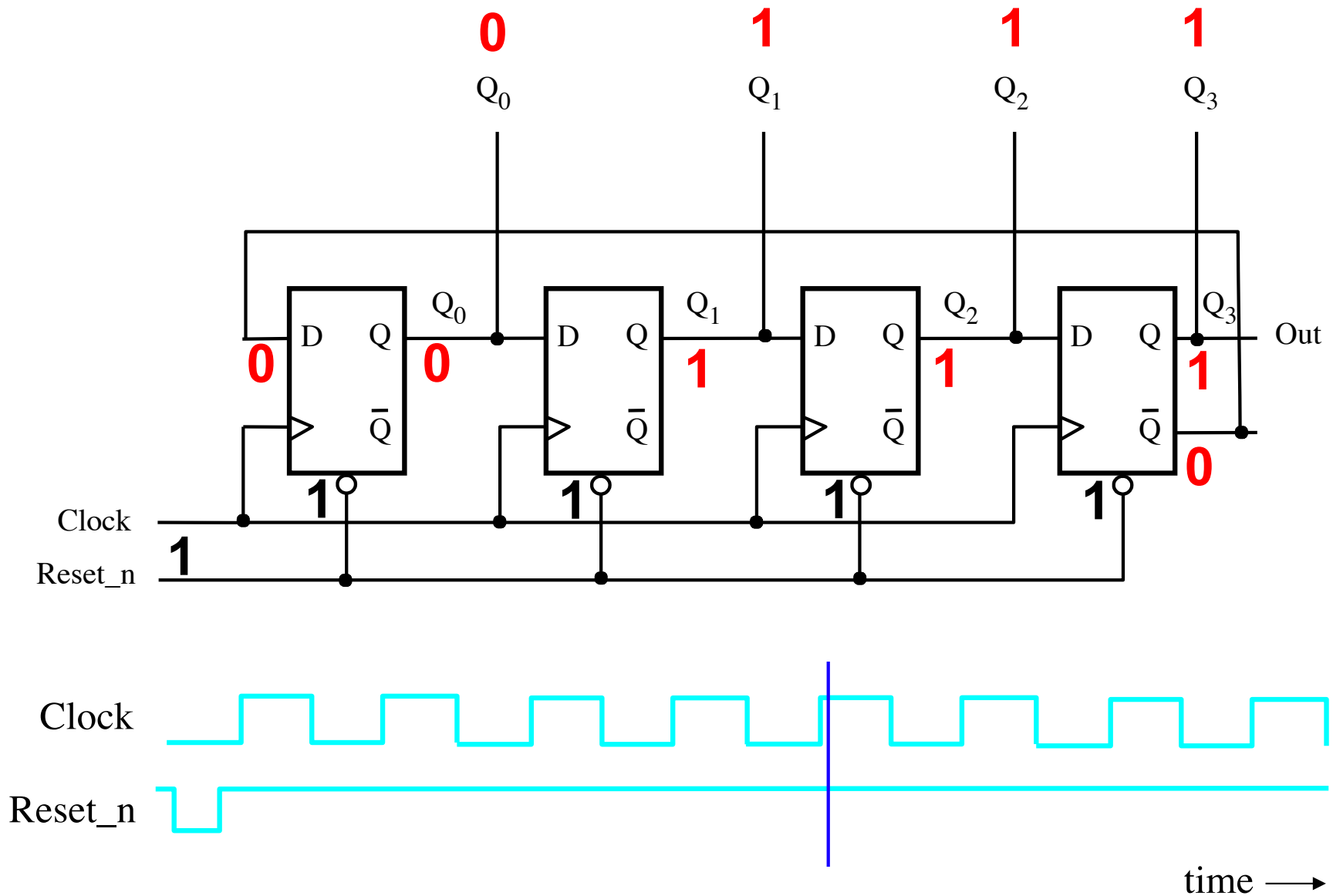
# Counting: How does it work?



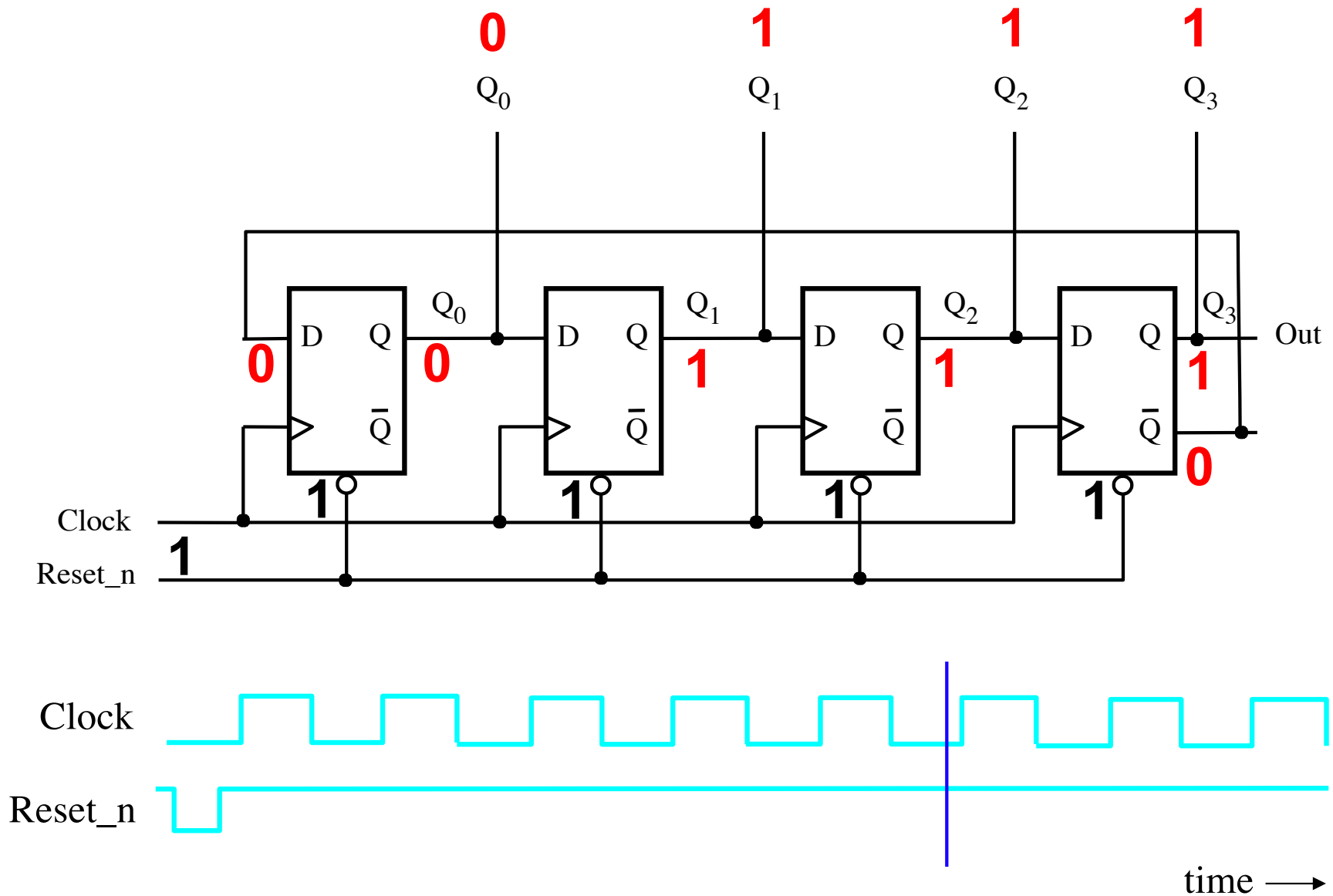
# Counting: How does it work?



# Counting: How does it work?

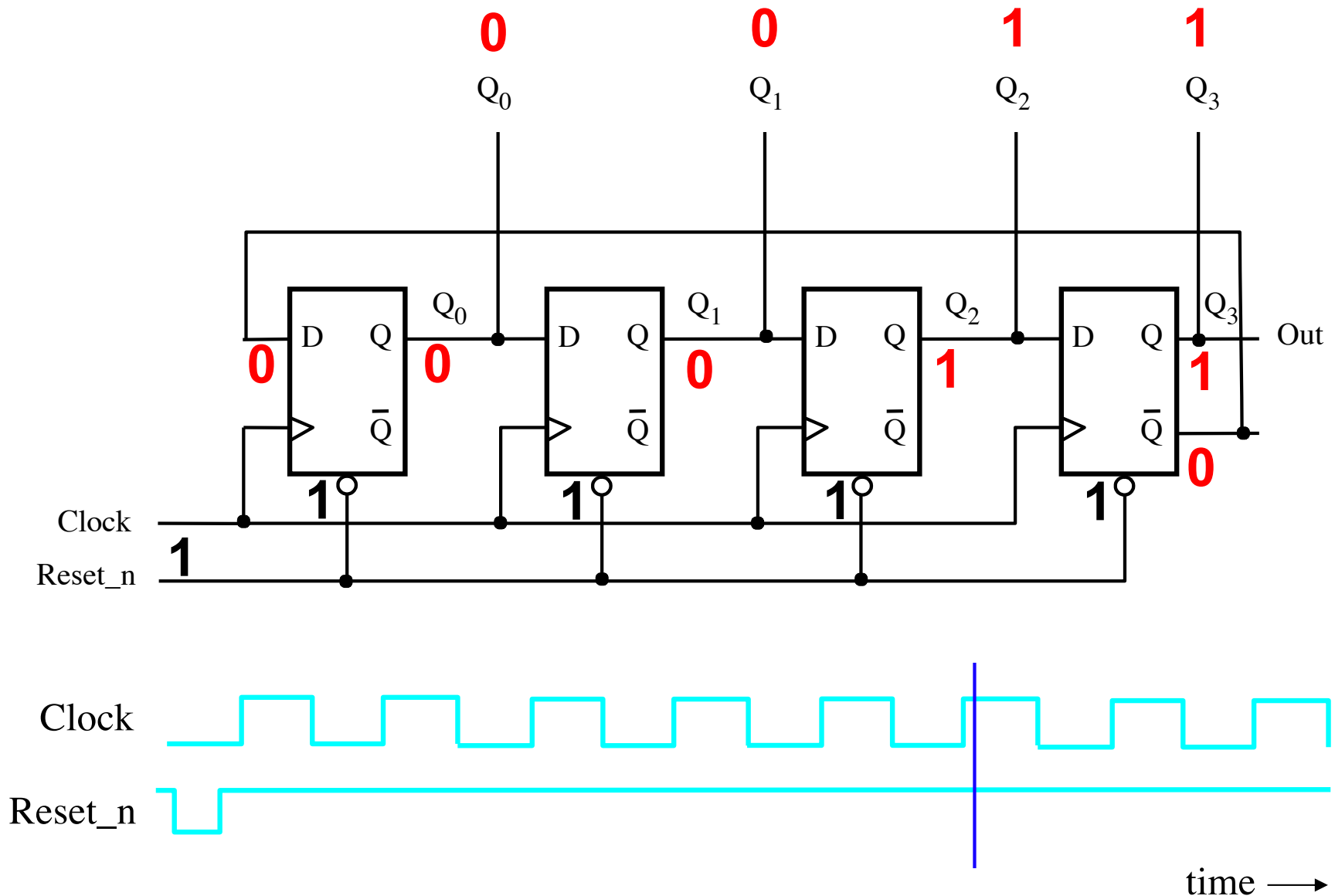


# Counting: How does it work?

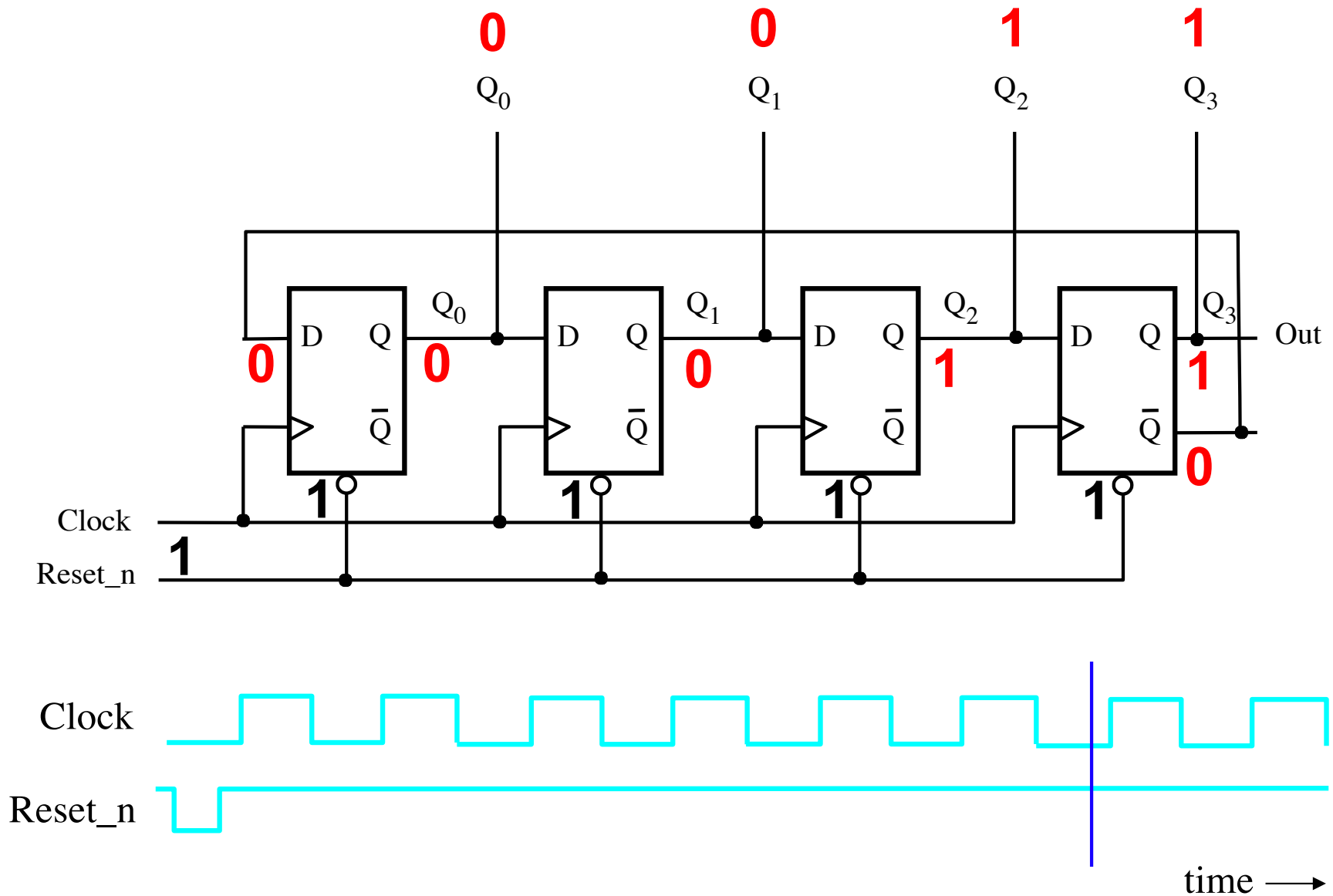




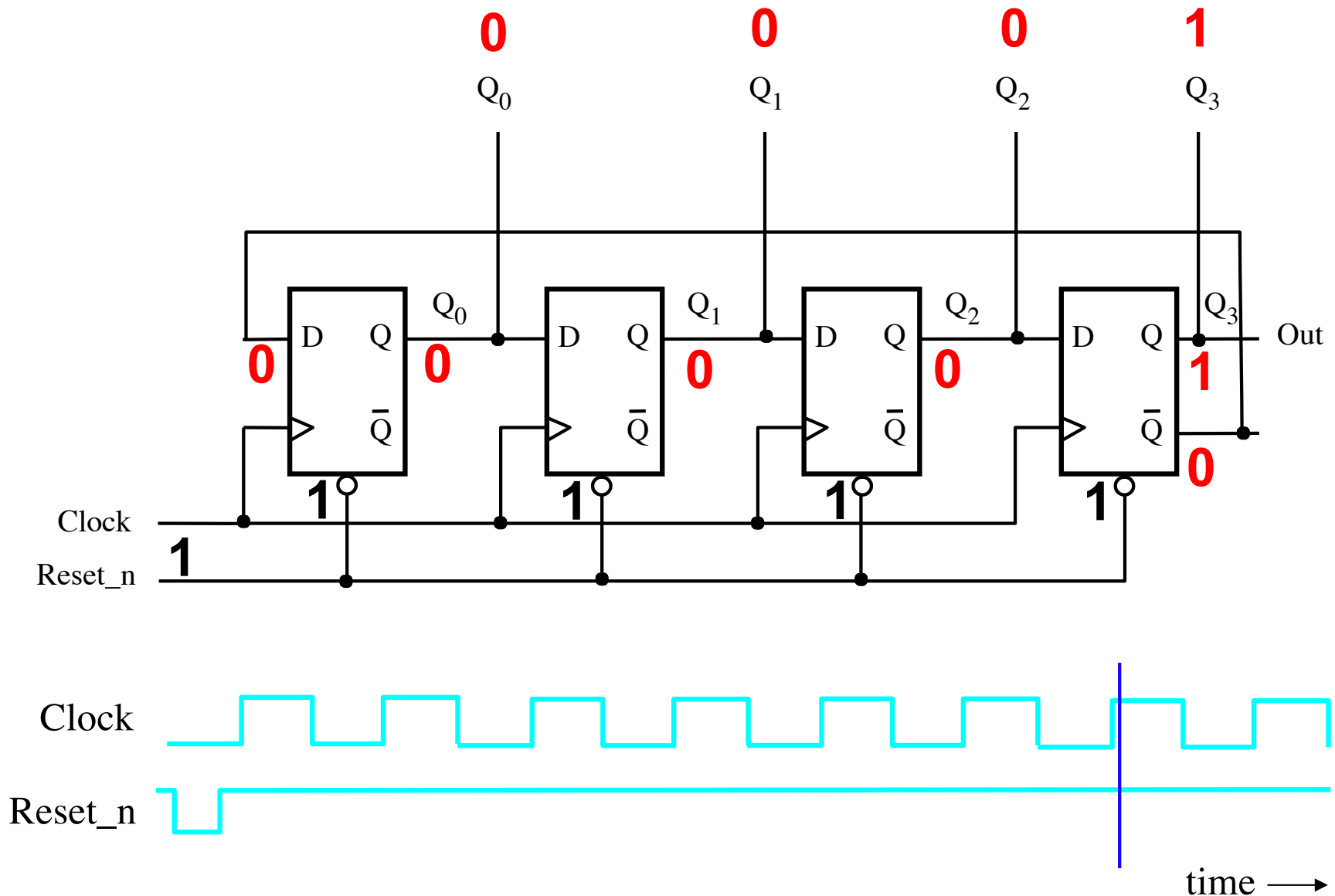
# Counting: How does it work?



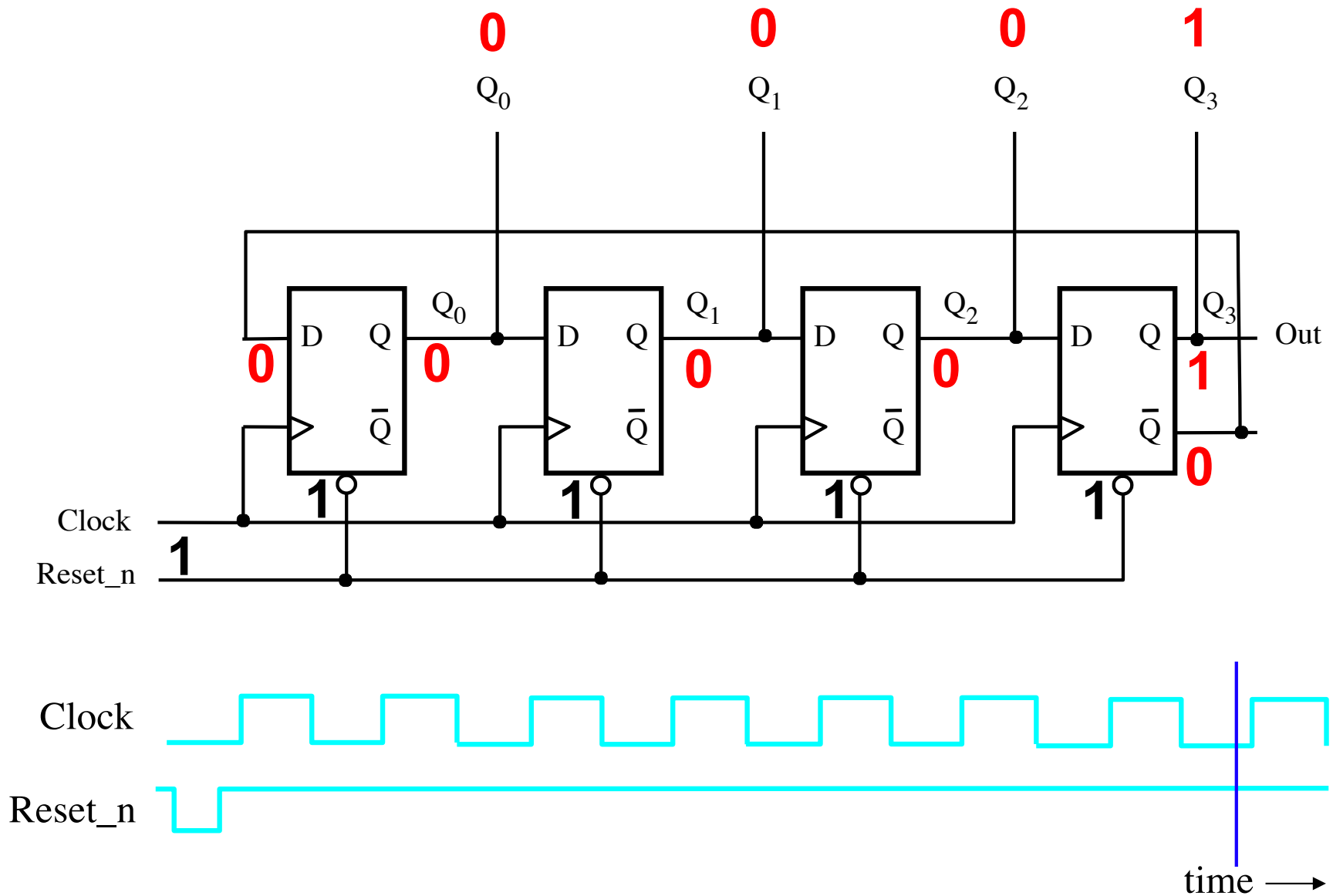
# Counting: How does it work?



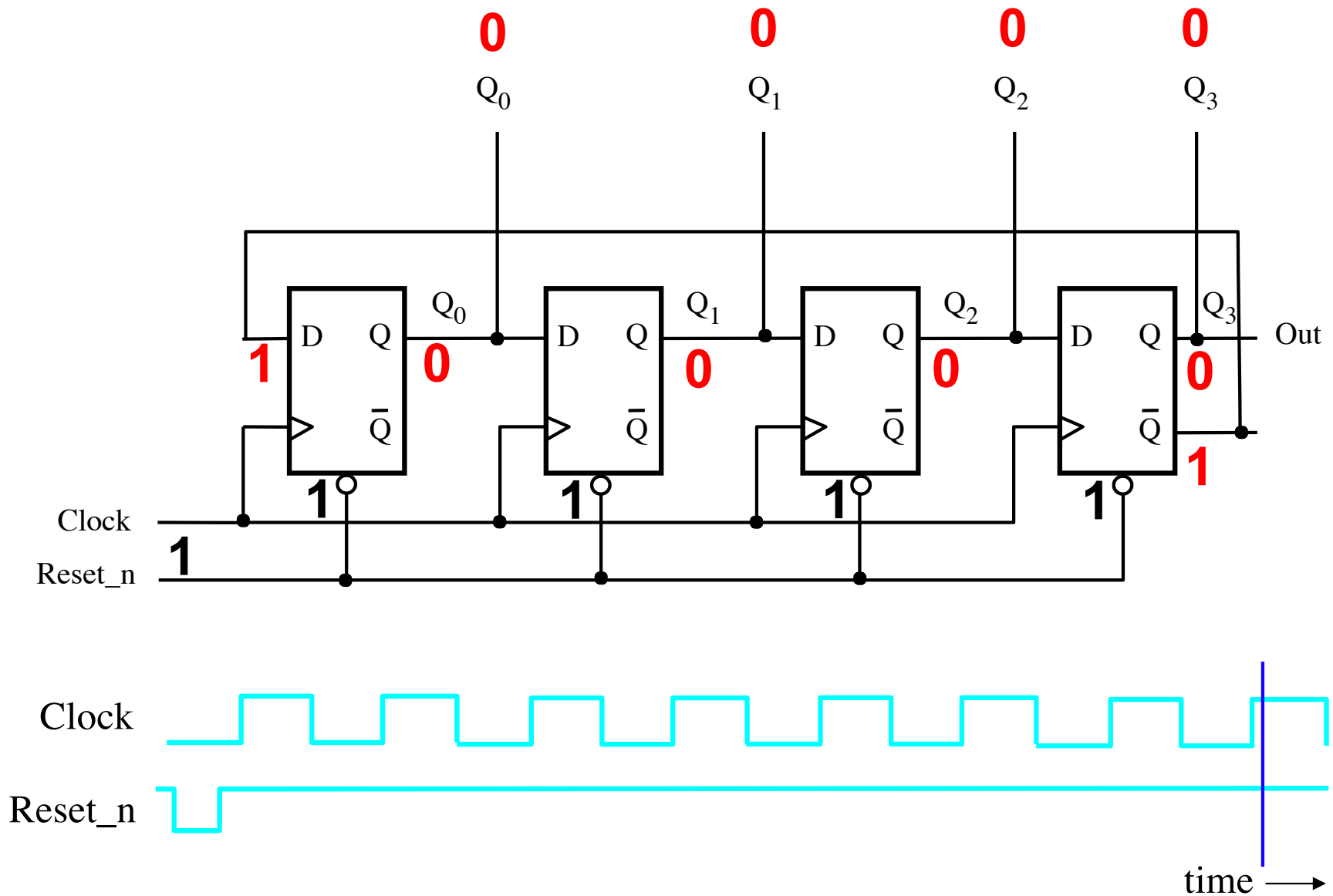
# Counting: How does it work?



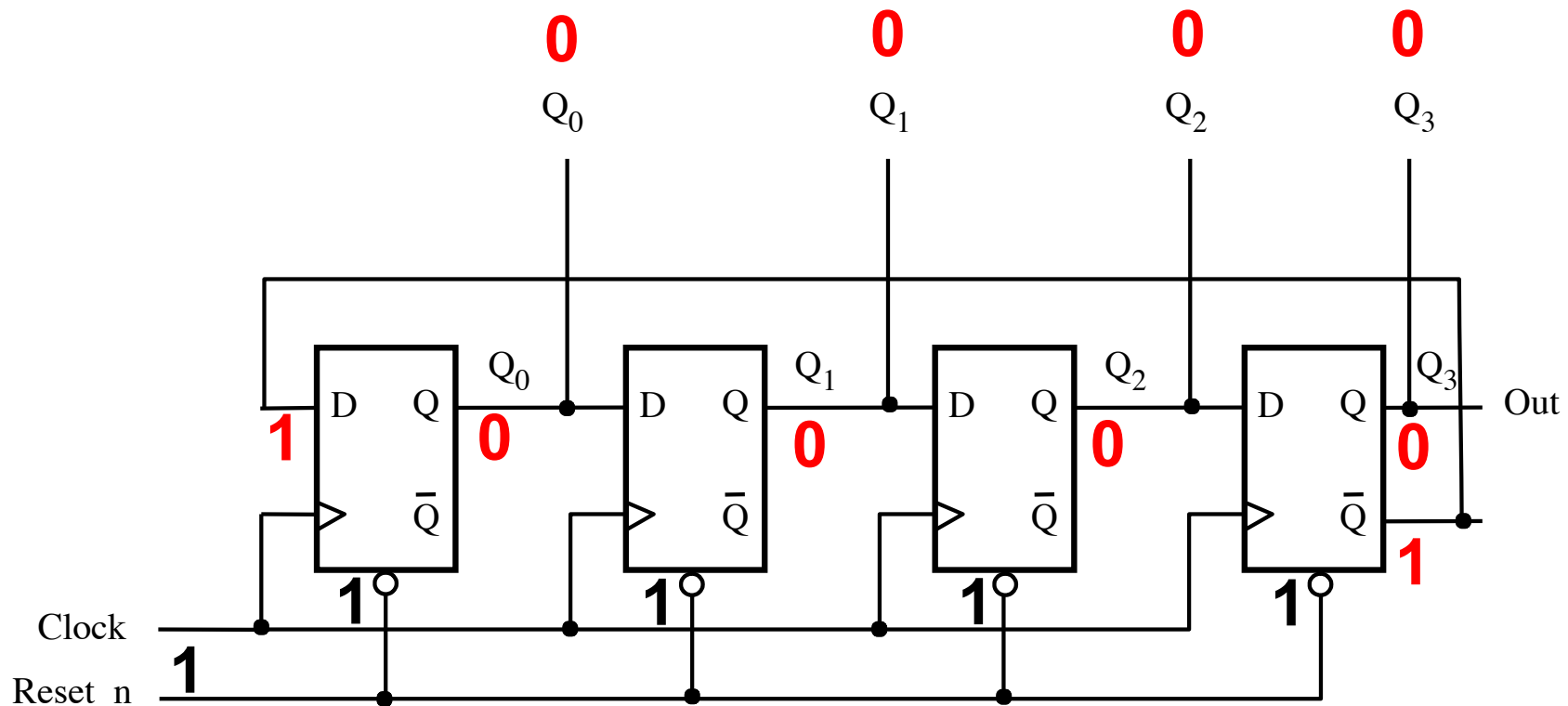
# Counting: How does it work?



# Counting: How does it work?

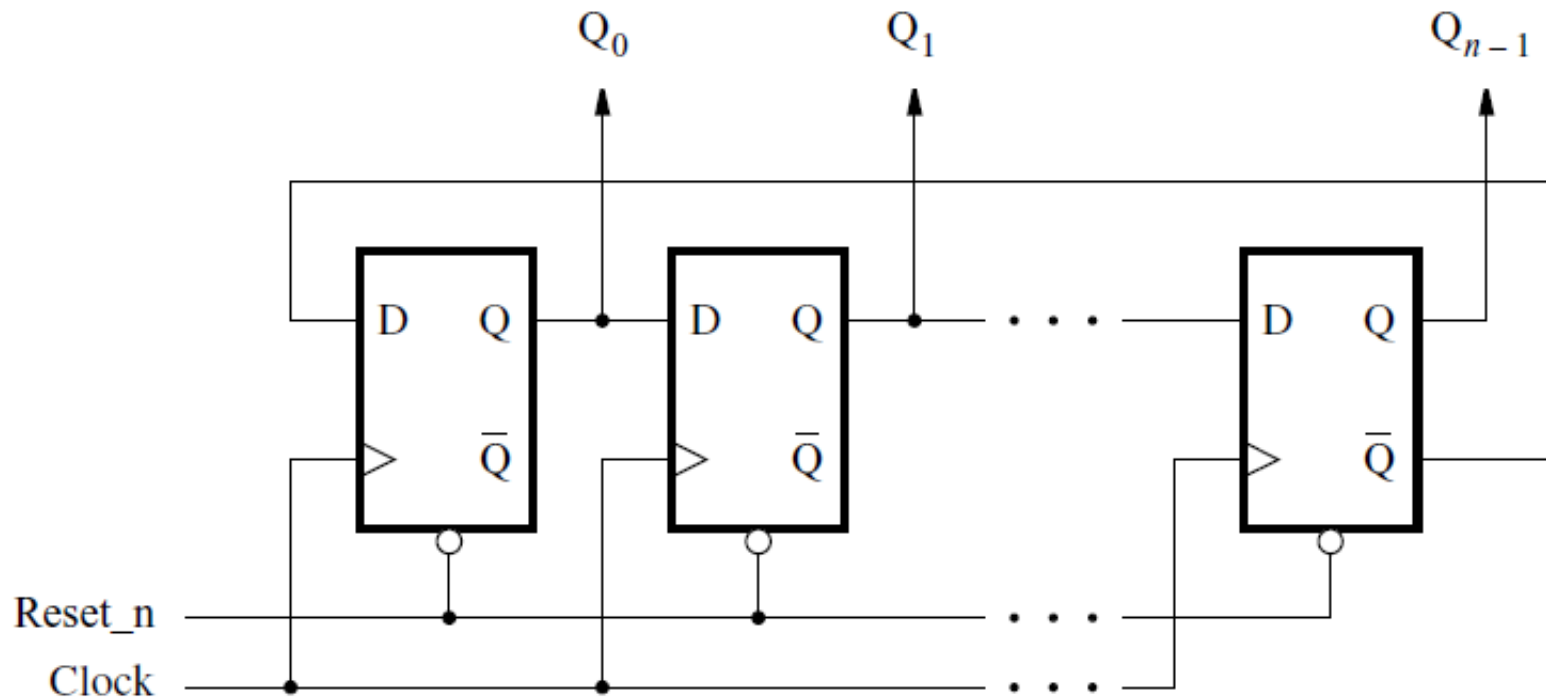


# Counting: How does it work?



It is back to the start of the counting sequence, which is:  
0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001.

# n-bit Johnson Counter



[ Figure 5.29 from the textbook ]

**Questions?**



**THE END**