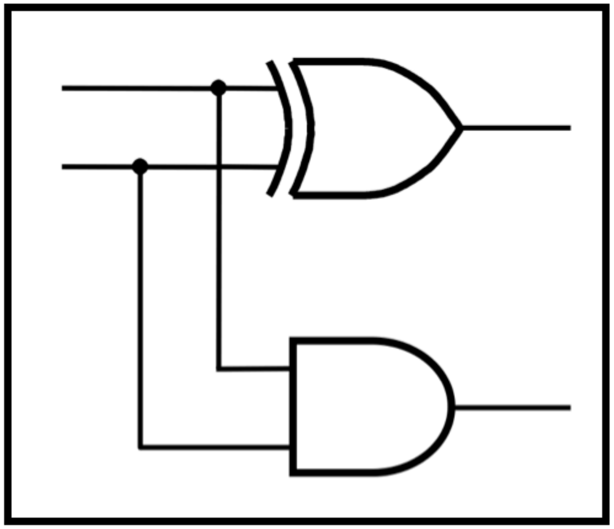


CprE 281: Digital Logic

Instructor: Alexander Stoytchev

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

Counters & Solved Problems

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

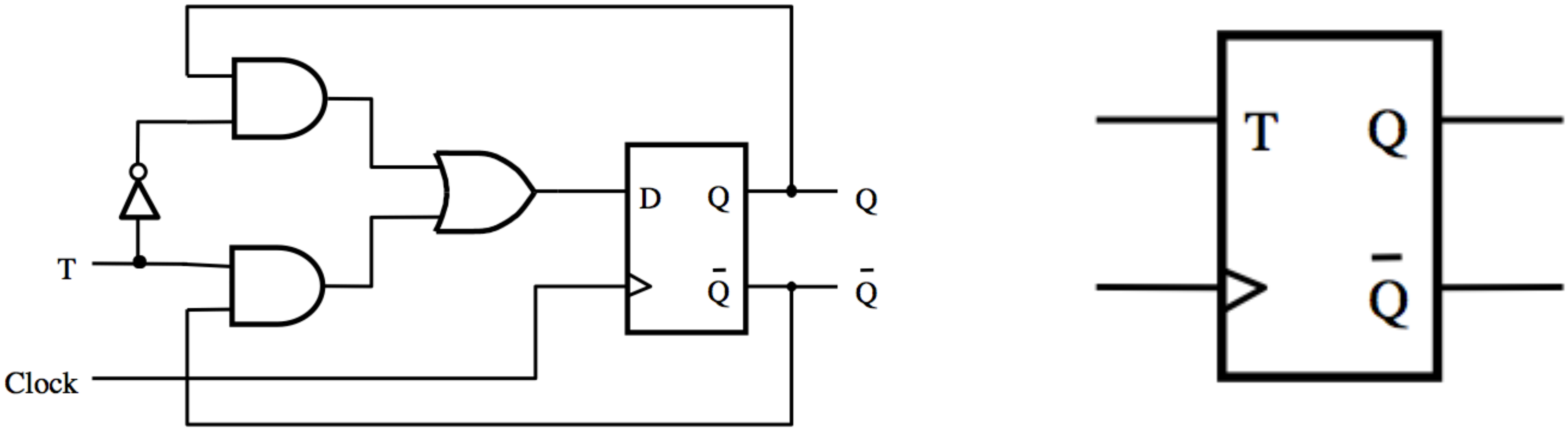
Administrative Stuff

- Homework 9 is out

- It is due on Monday Nov 5

Counters

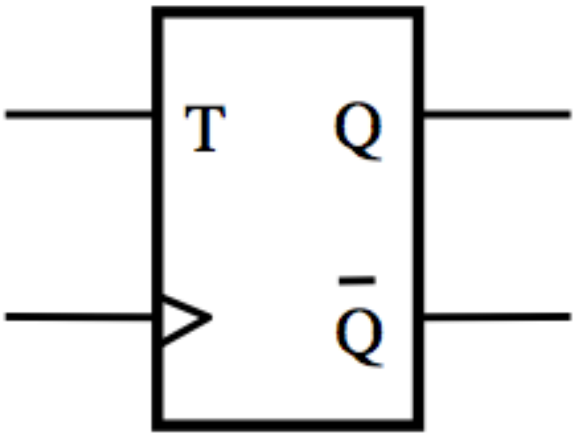
T Flip-Flop
(circuit and graphical symbol)



[Figure 5.15a,c from the textbook]

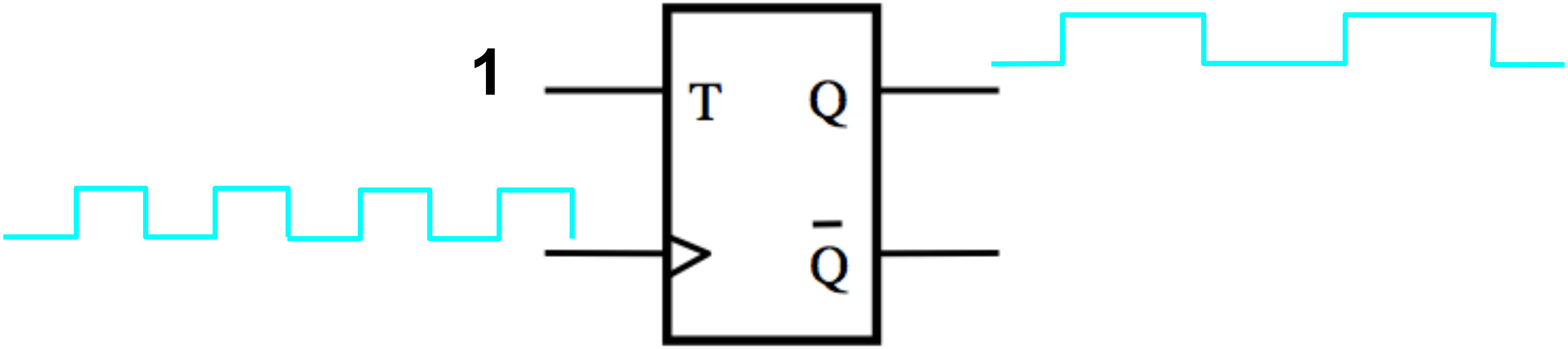
The output of the T Flip-Flop

divides the frequency of the clock by 2

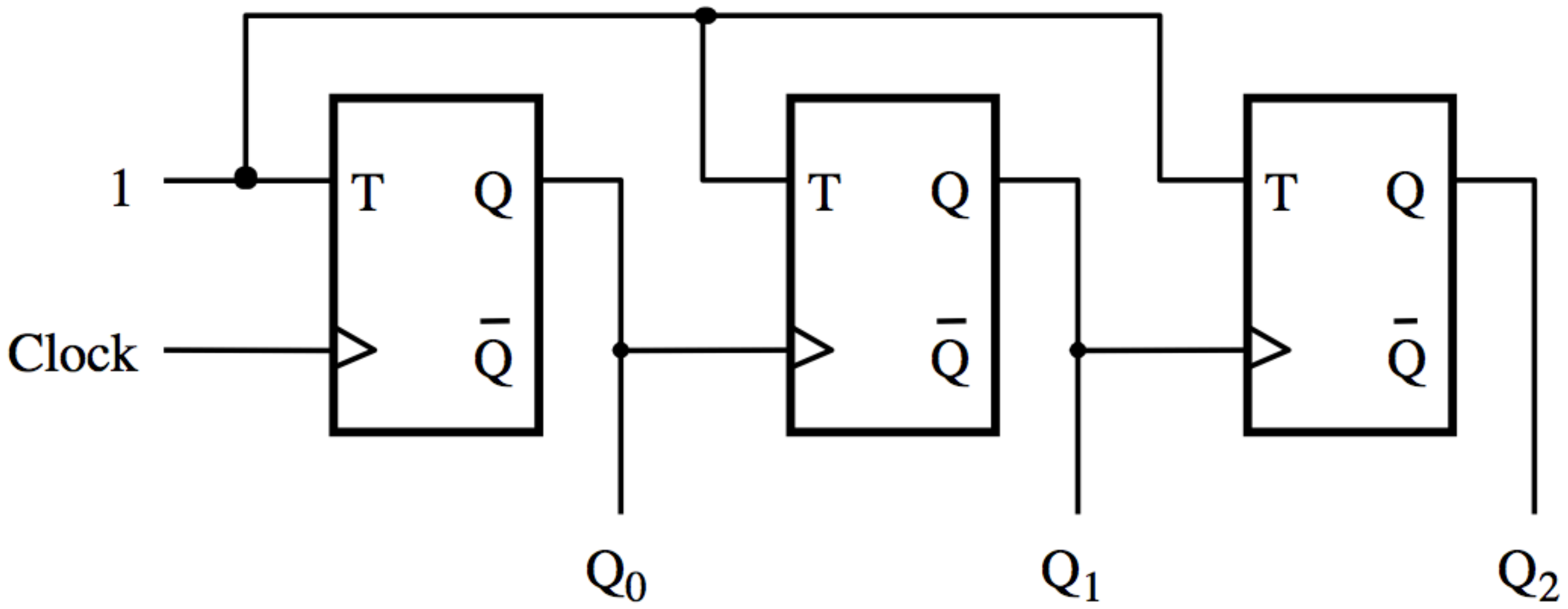


The output of the T Flip-Flop

divides the frequency of the clock by 2

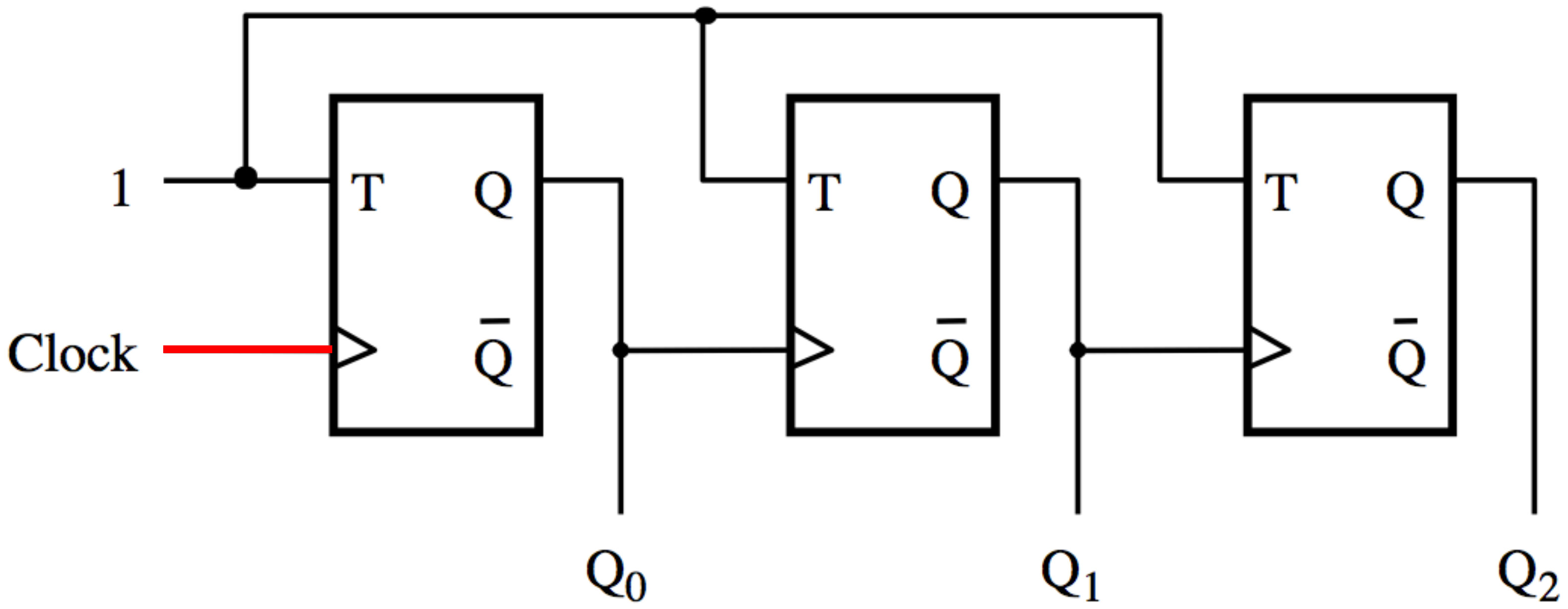


A three-bit down-counter



[Figure 5.20 from the textbook]

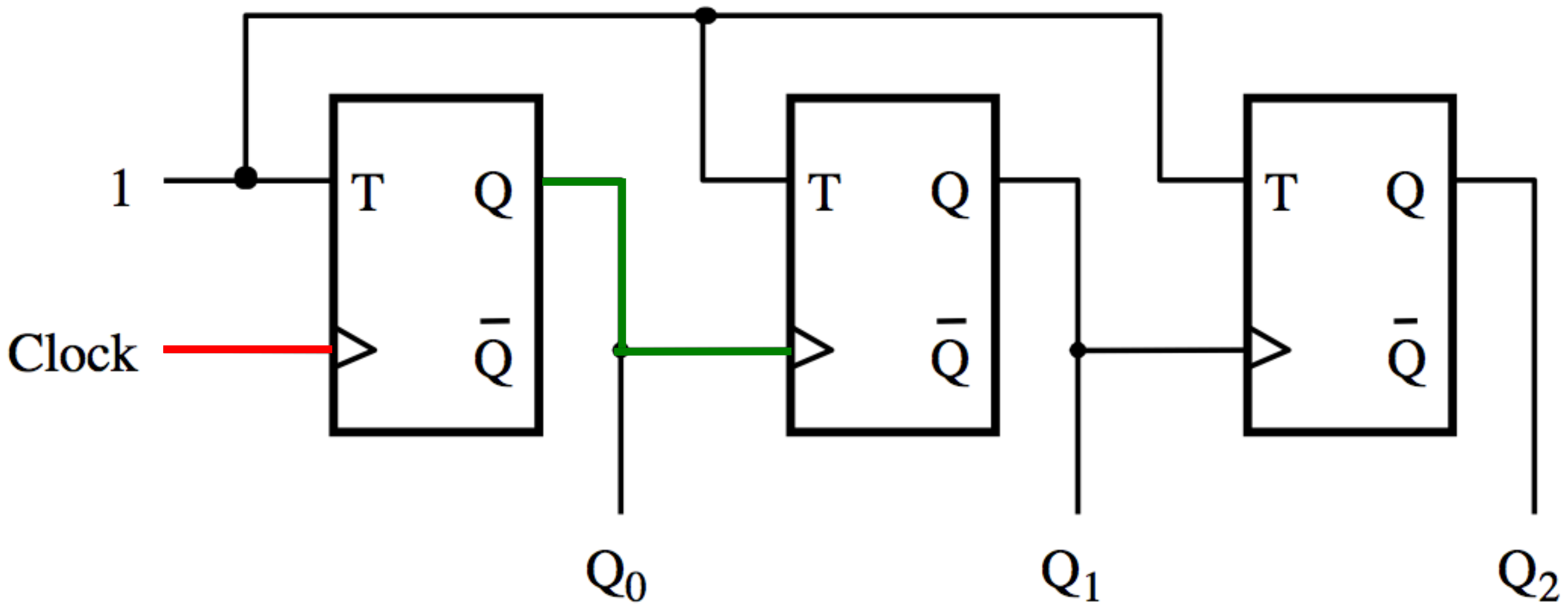
A three-bit down-counter



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

[Figure 5.20 from the textbook]

A three-bit down-counter

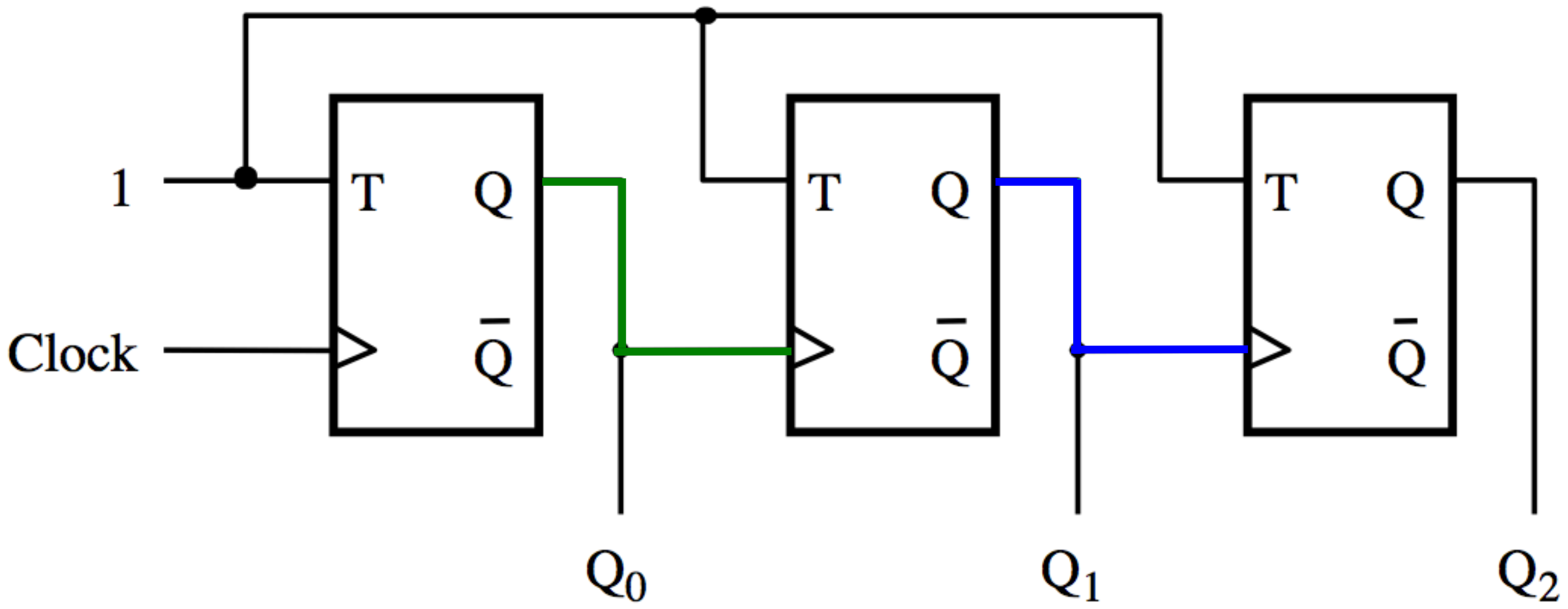


The first flip-flop changes on the positive edge of the clock

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

[Figure 5.20 from the textbook]

A three-bit down-counter



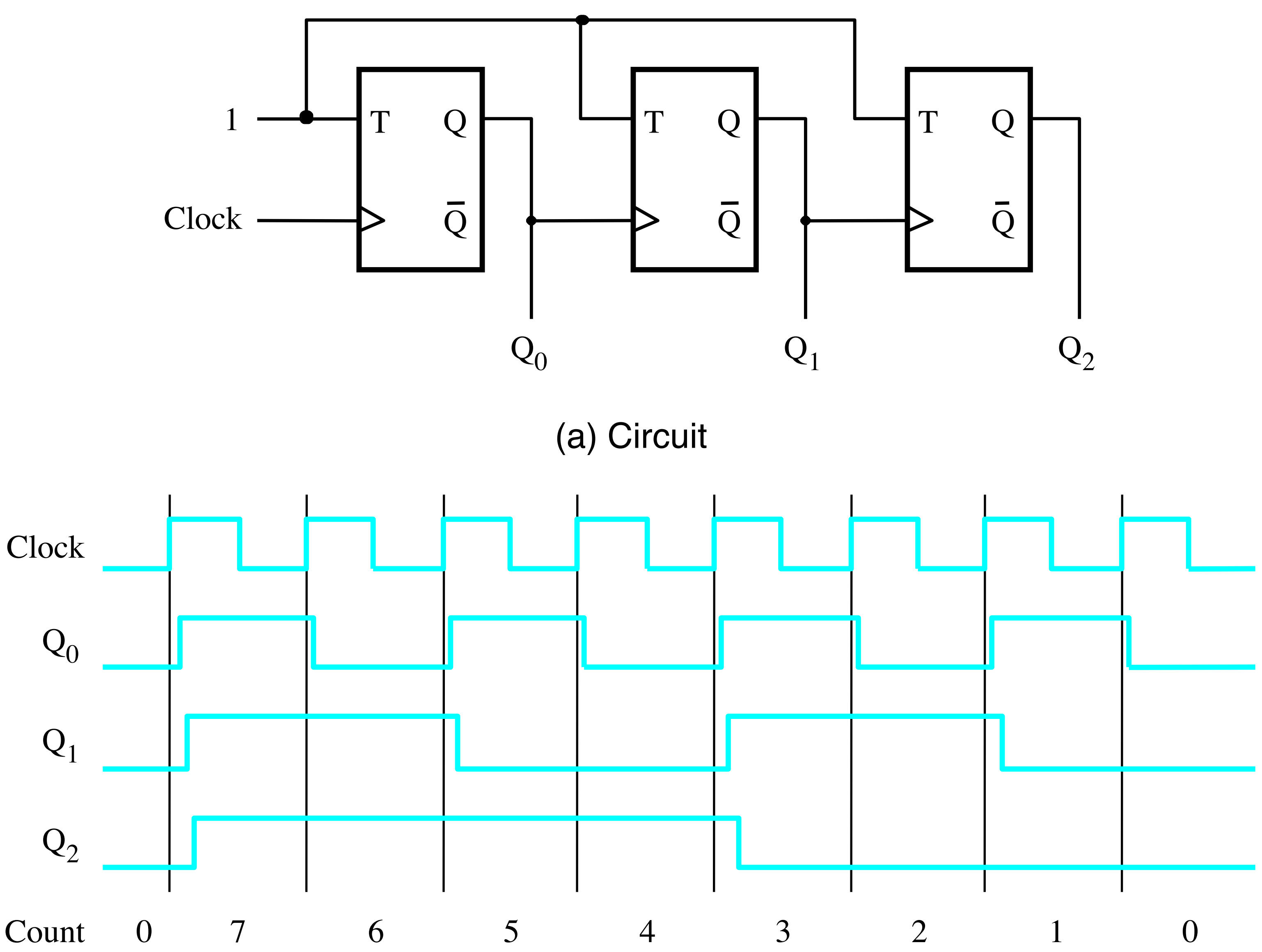
The first flip-flop changes on the positive edge of the clock

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

The third flip-flop changes on the positive edge of Q_1

[Figure 5.20 from the textbook]

A three-bit down-counter

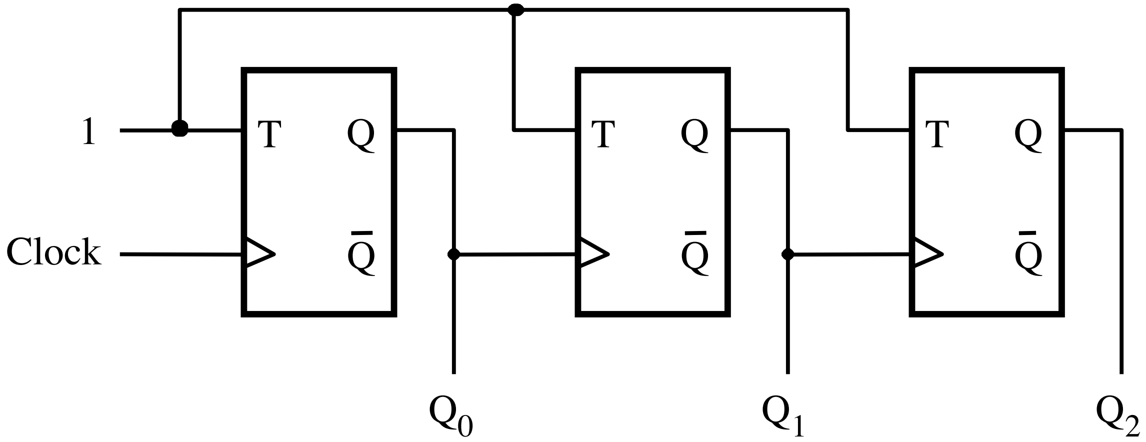


(a) Circuit

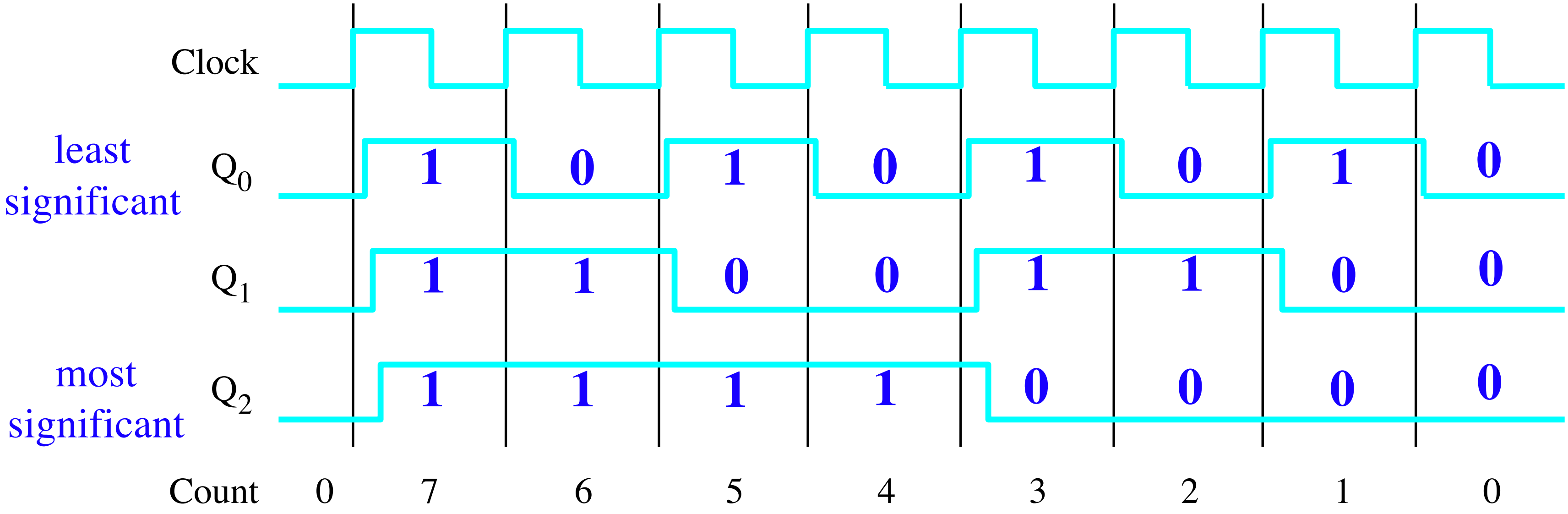
(b) Timing diagram

[Figure 5.20 from the textbook]

A three-bit down-counter

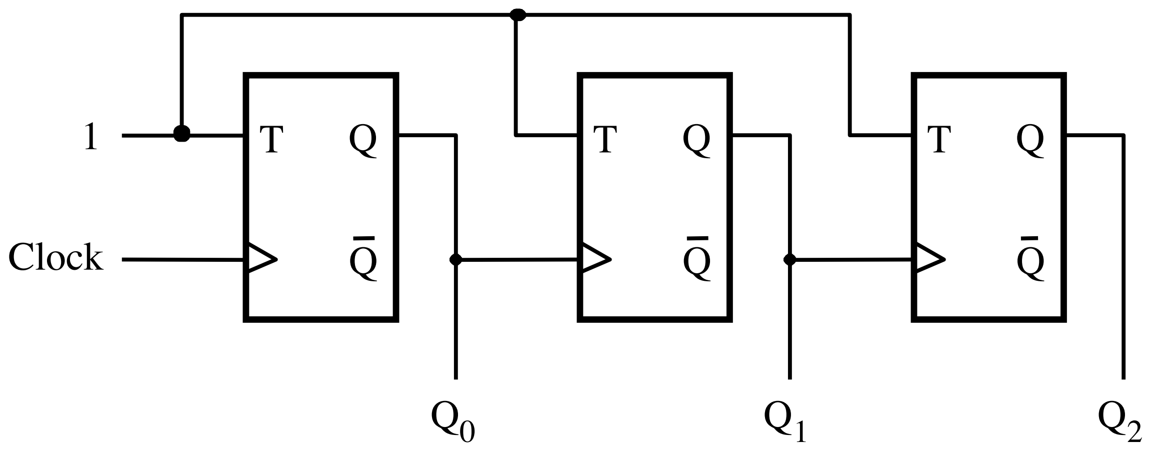


(a) Circuit

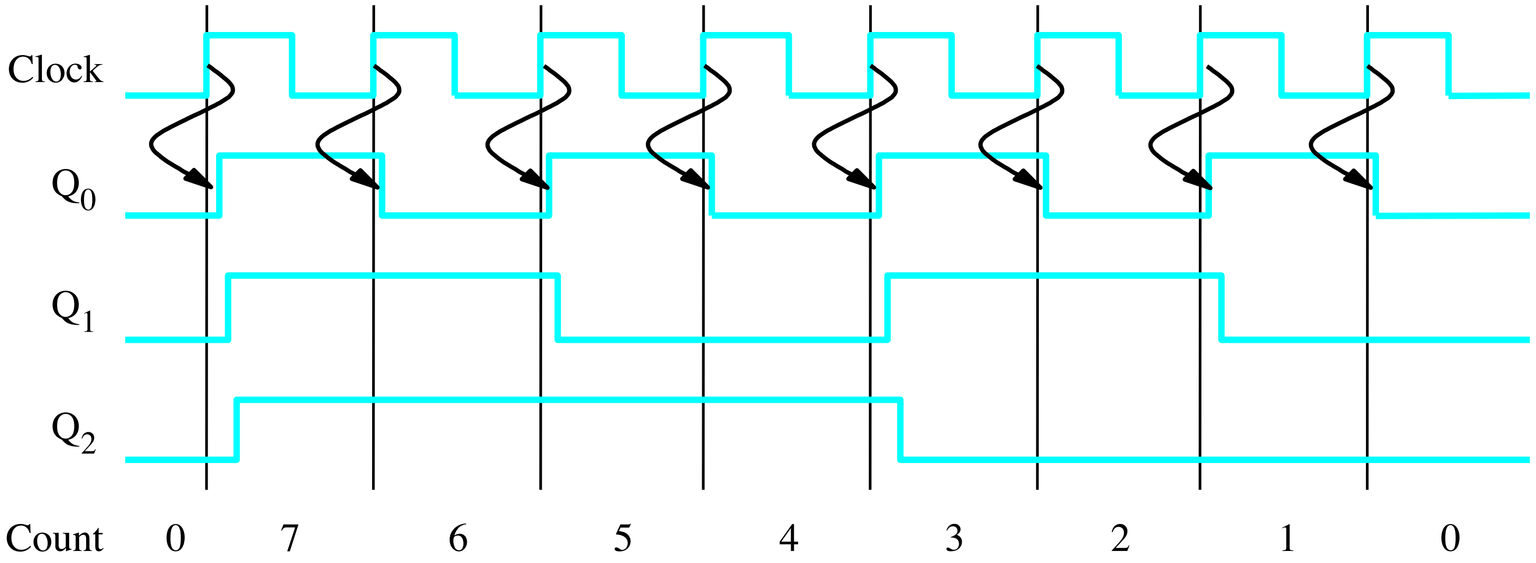


(b) Timing diagram

A three-bit down-counter



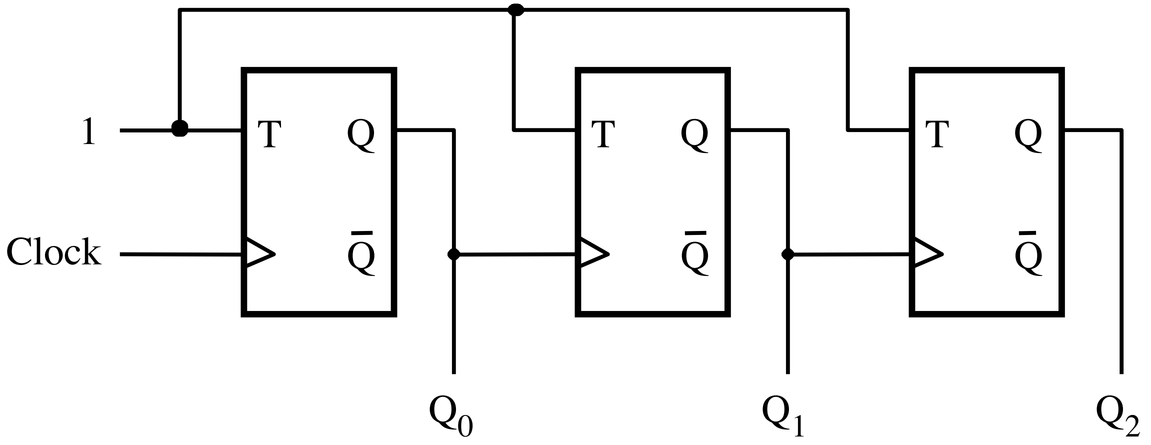
(a) Circuit



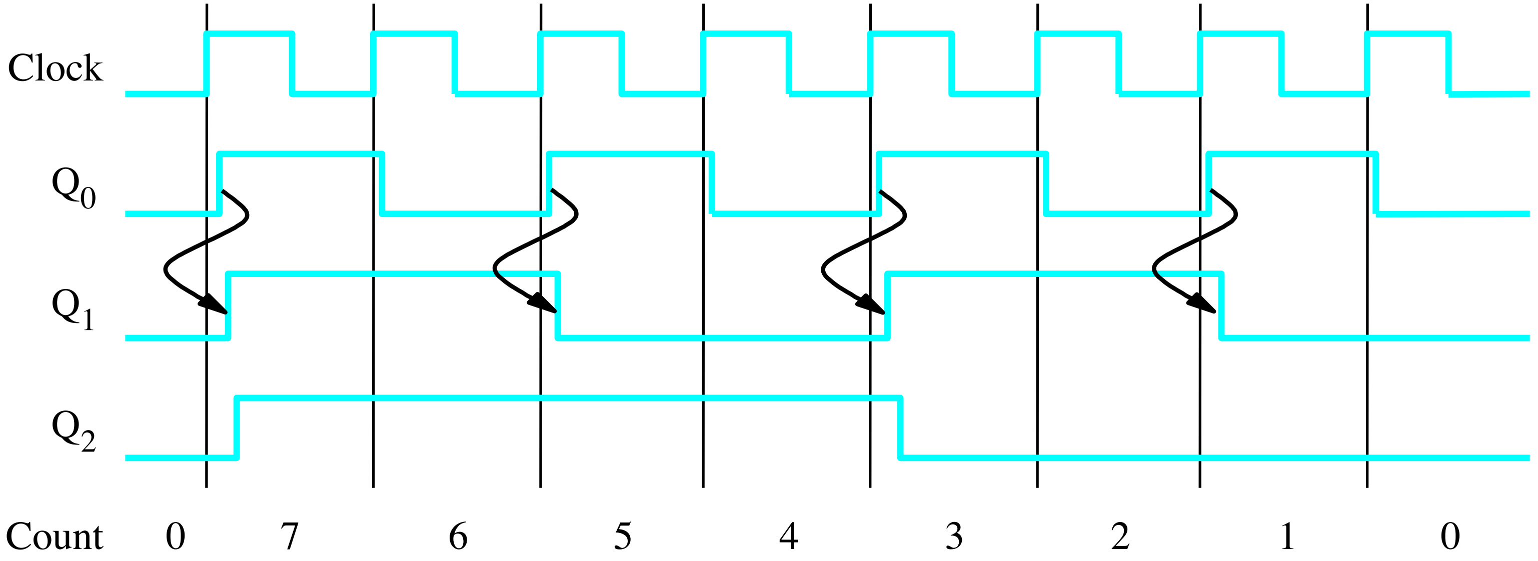
Q_0 toggles on the positive clock edge

(b) Timing diagram

A three-bit down-counter



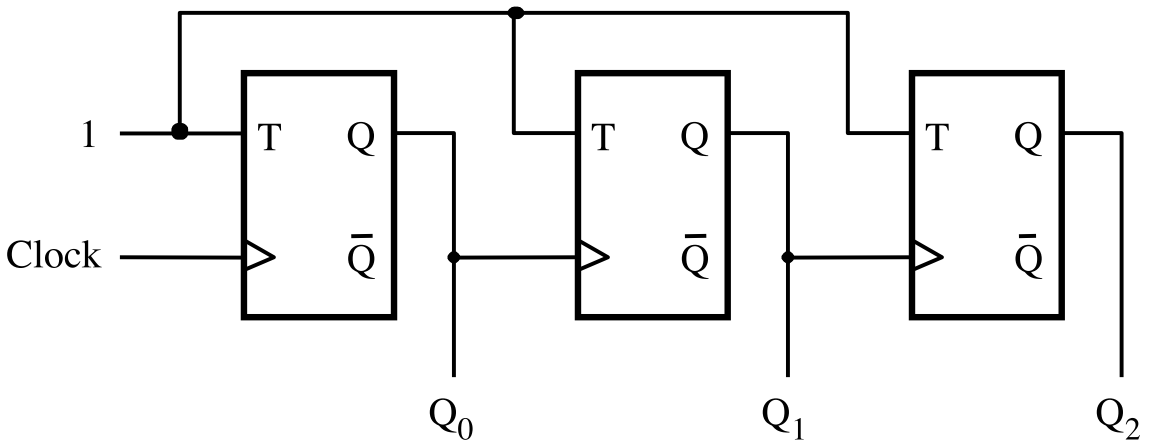
(a) Circuit



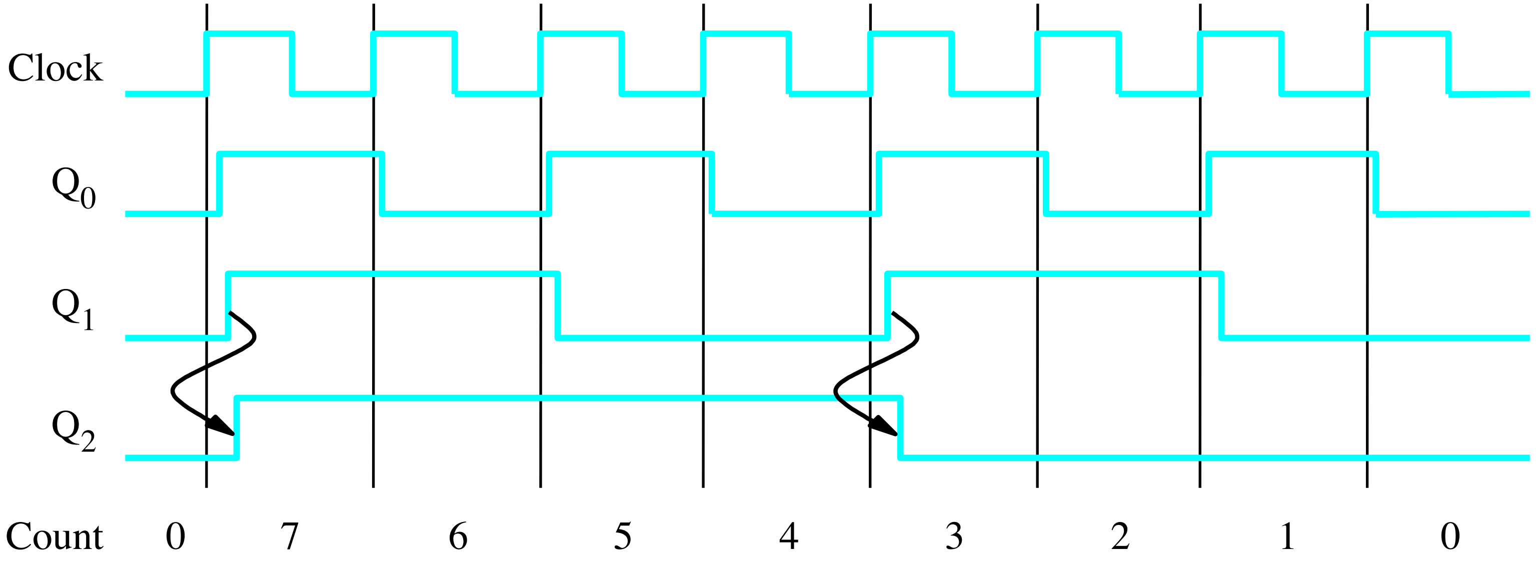
Q_1 toggles on the positive edge of Q_0

(b) Timing diagram

A three-bit down-counter



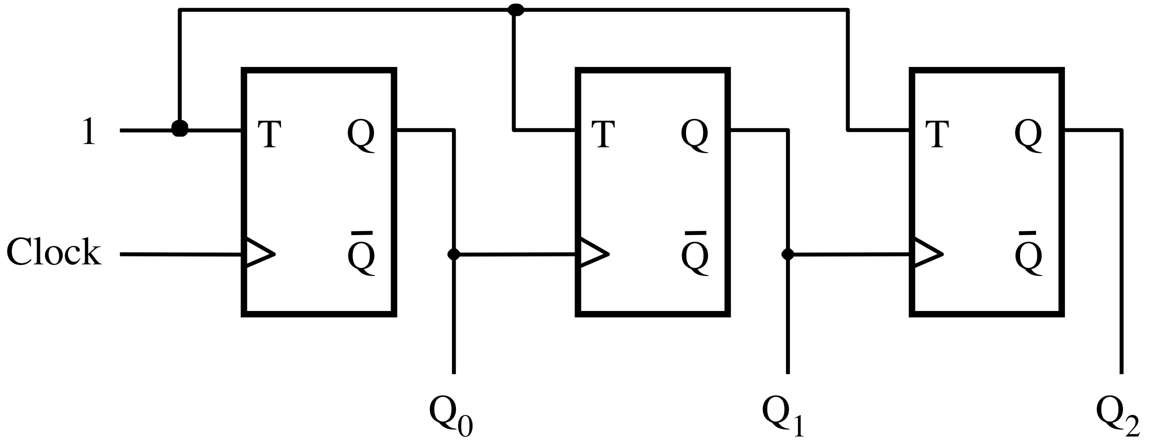
(a) Circuit



Q_2 toggles on the positive edge of Q_1

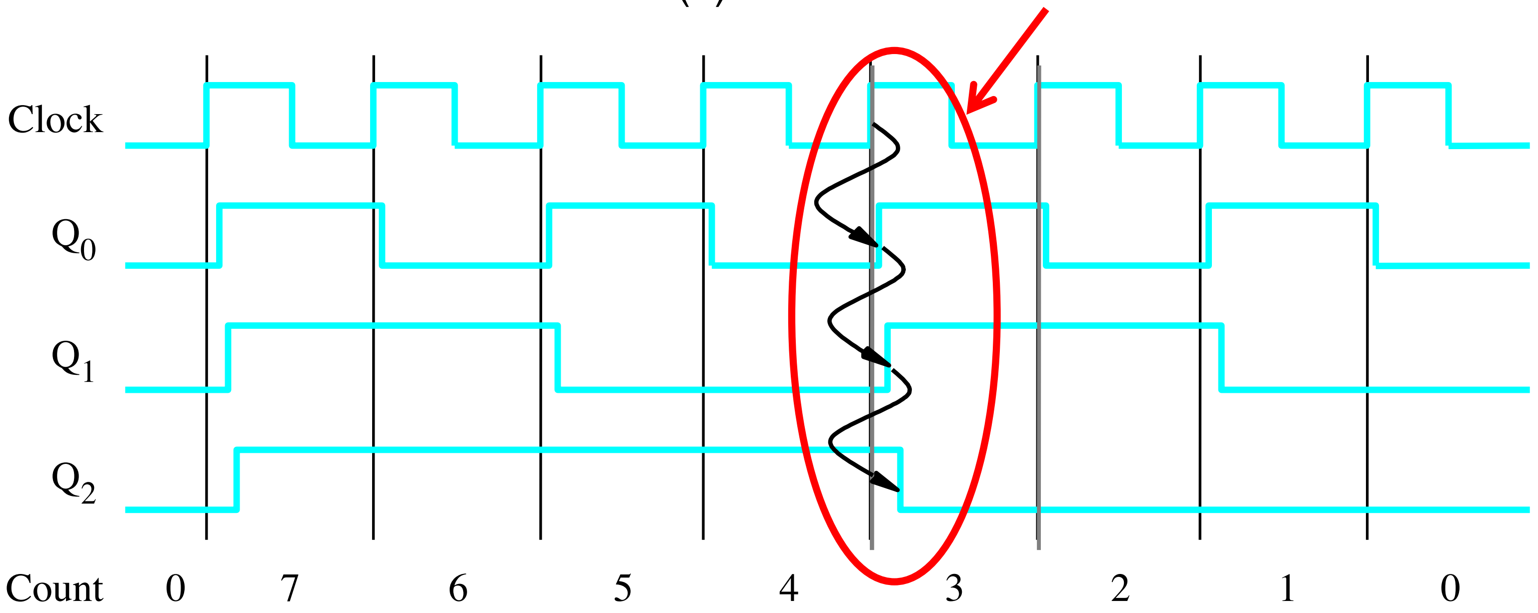
(b) Timing diagram

A three-bit down-counter



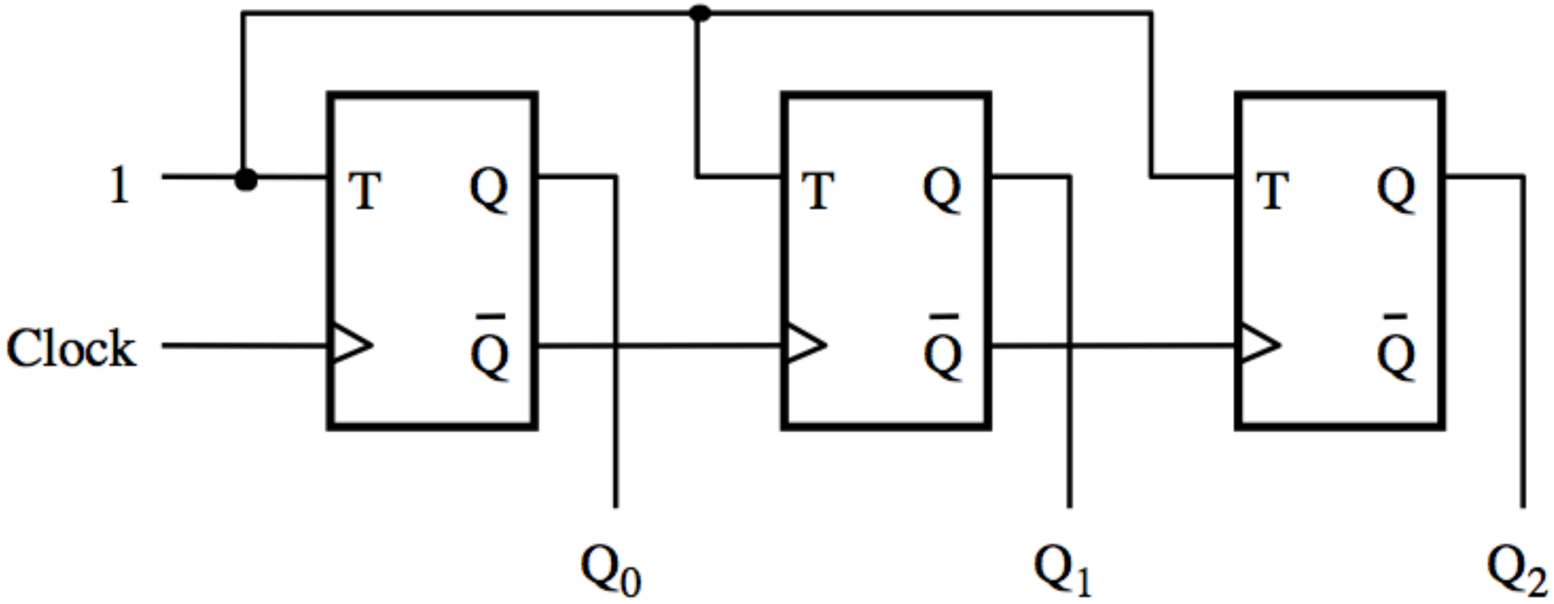
(a) Circuit

The propagation delays get longer



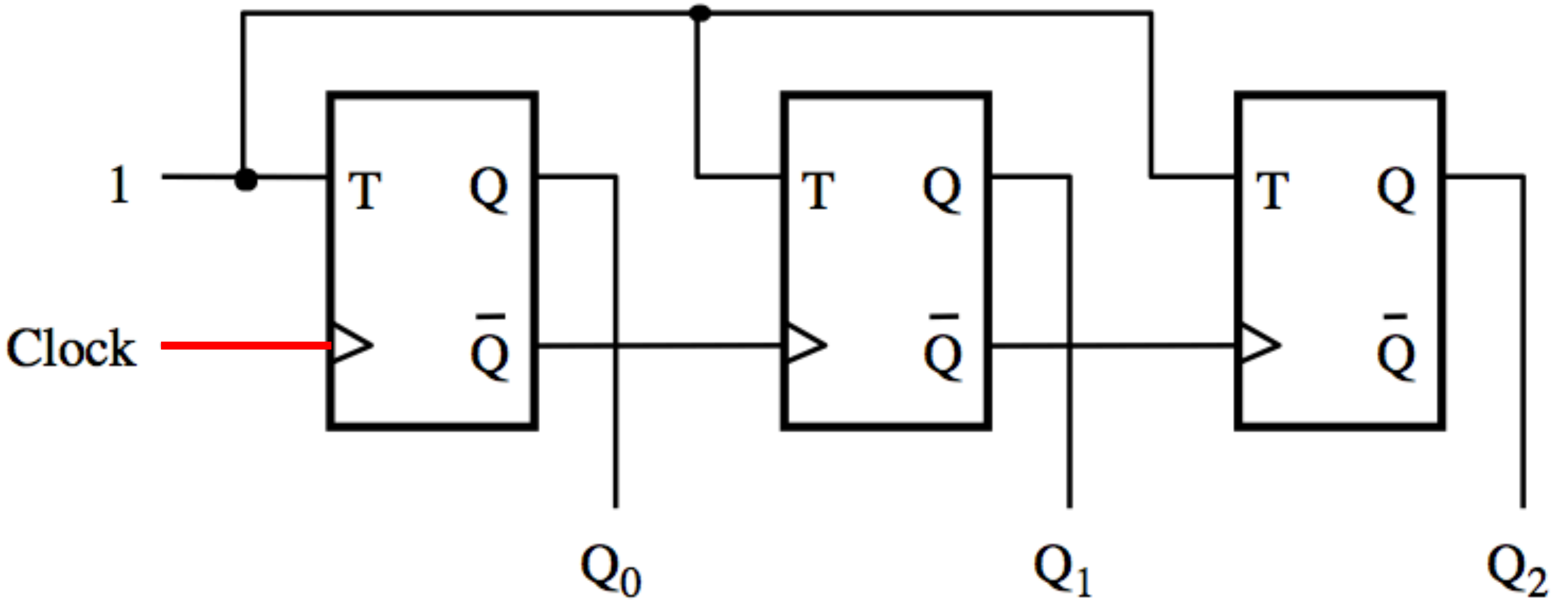
(b) Timing diagram

A three-bit up-counter



[Figure 5.19 from the textbook]

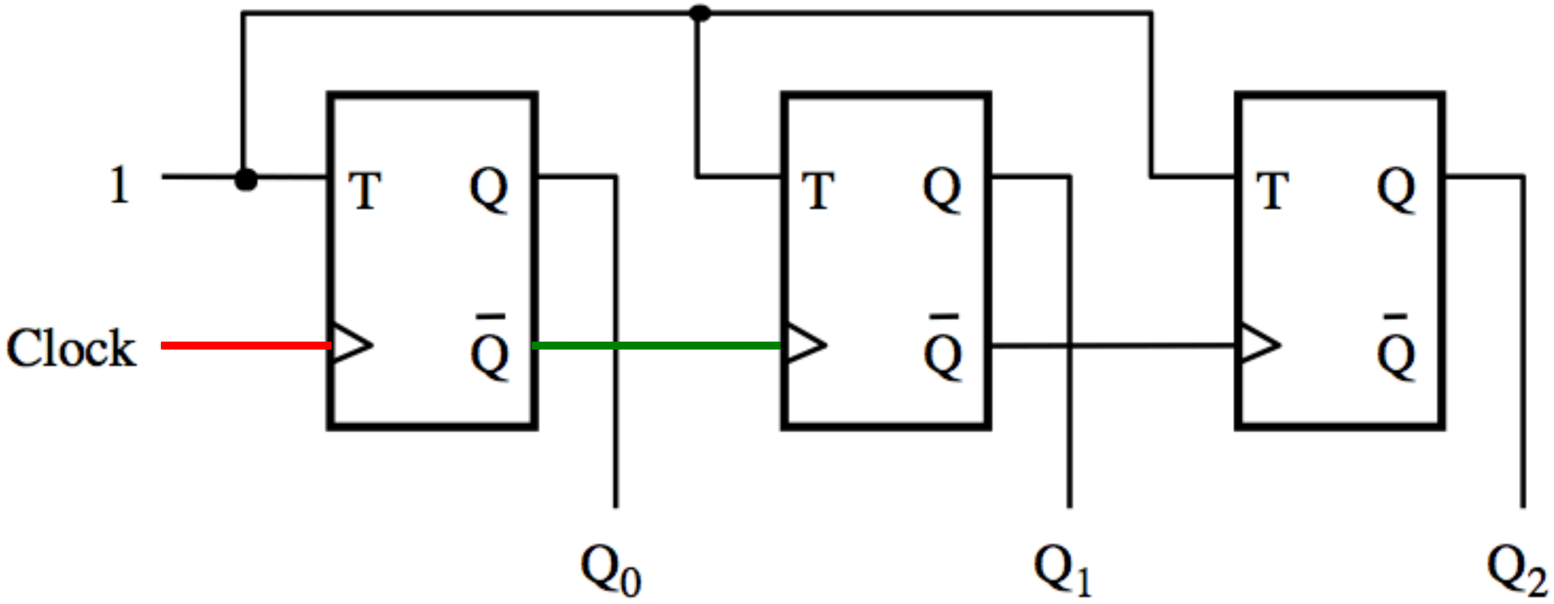
A three-bit up-counter



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

[Figure 5.19 from the textbook]

A three-bit up-counter

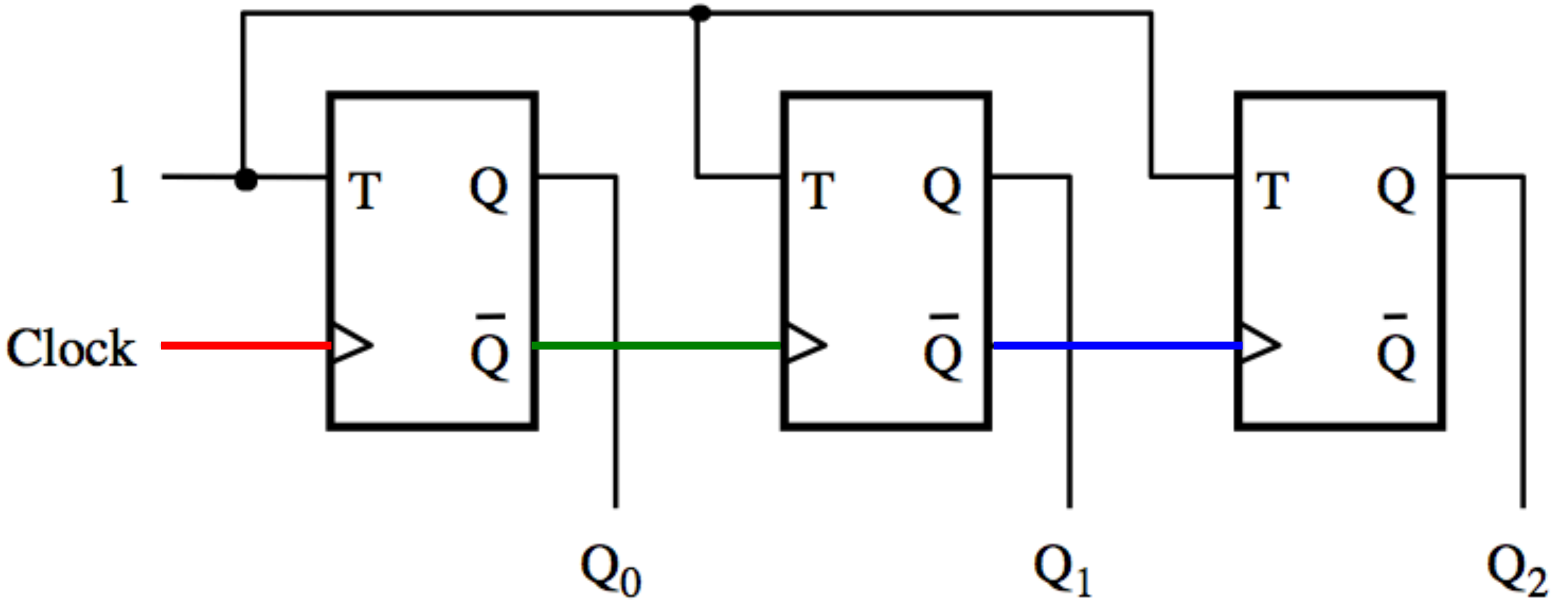


The first flip-flop changes on the positive edge of the clock

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

[Figure 5.19 from the textbook]

A three-bit up-counter



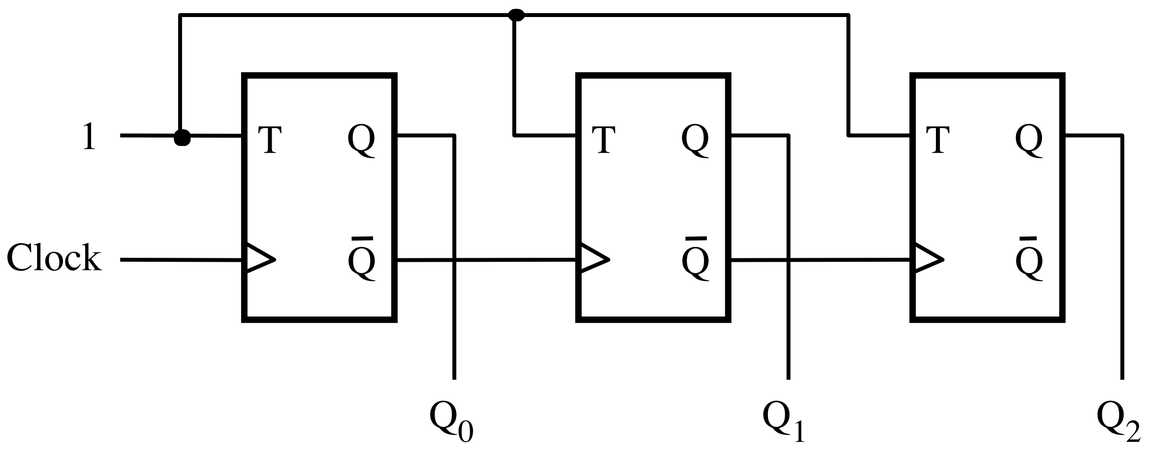
The first flip-flop changes on the positive edge of the clock

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

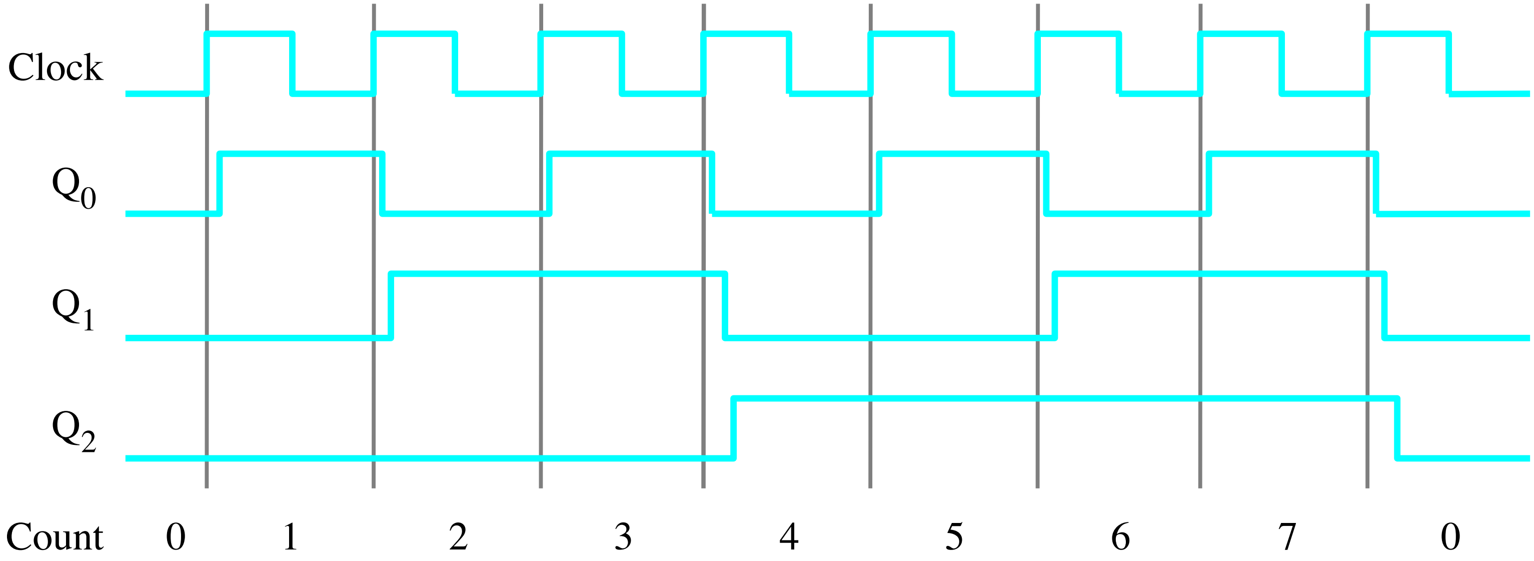
The third flip-flop changes on the positive edge of $\overline{Q}_1$

[Figure 5.19 from the textbook]

A three-bit up-counter



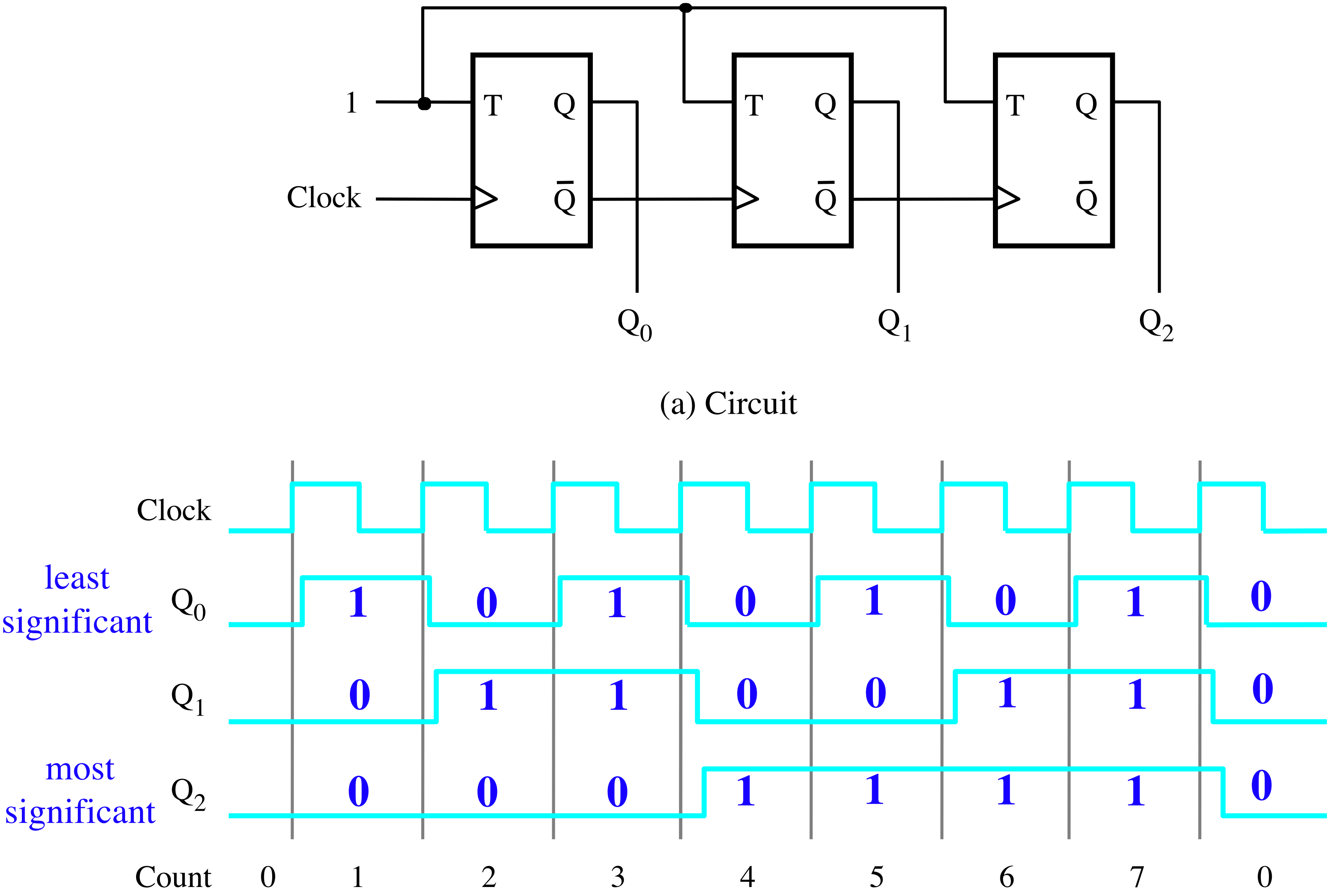
(a) Circuit



(b) Timing diagram

[Figure 5.19 from the textbook]

A three-bit up-counter

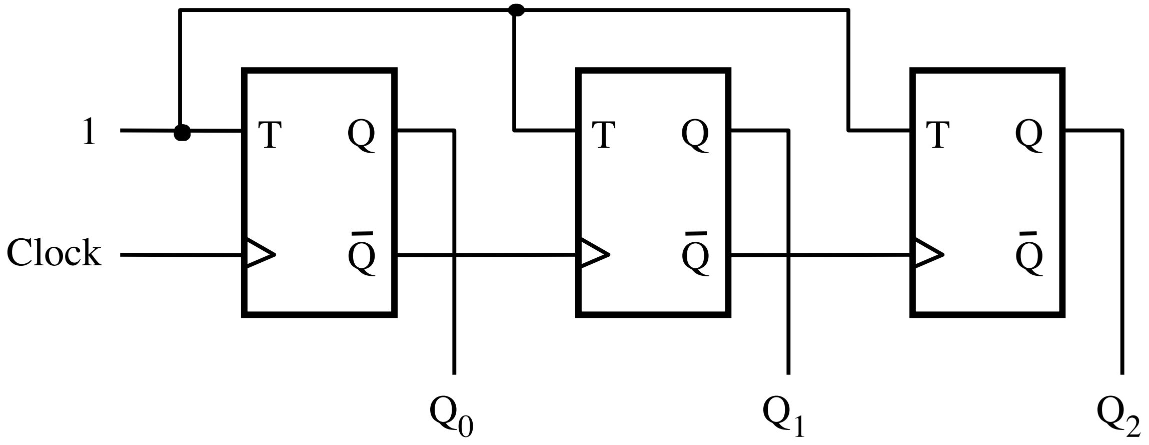


(a) Circuit

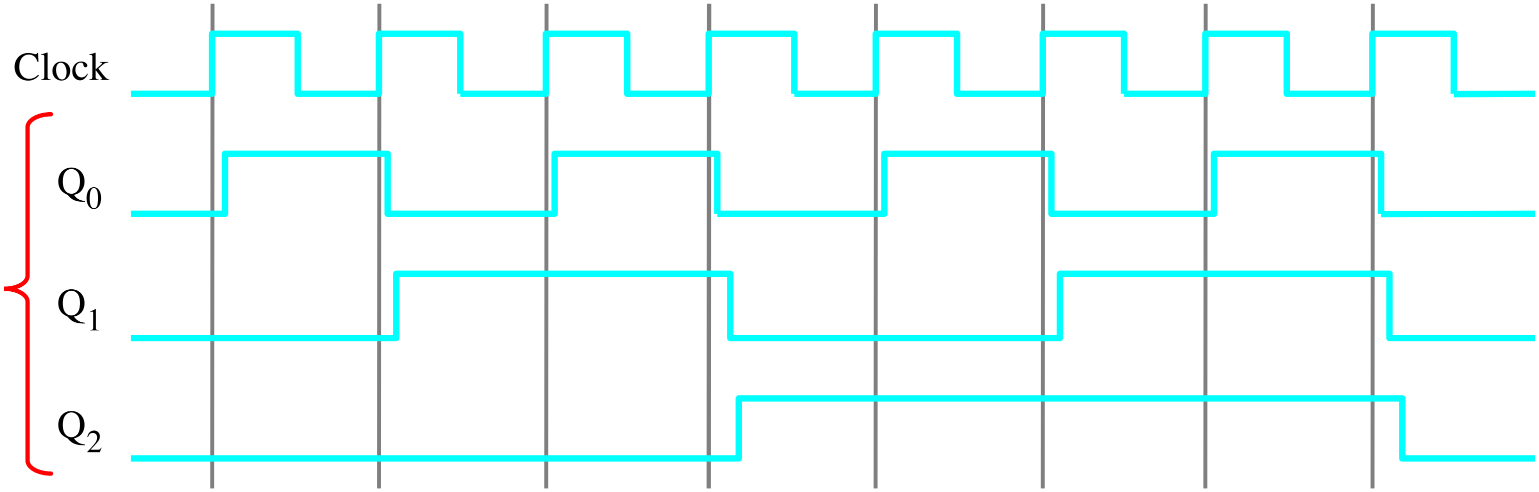
(b) Timing diagram

[Figure 5.19 from the textbook]

A three-bit up-counter



(a) Circuit

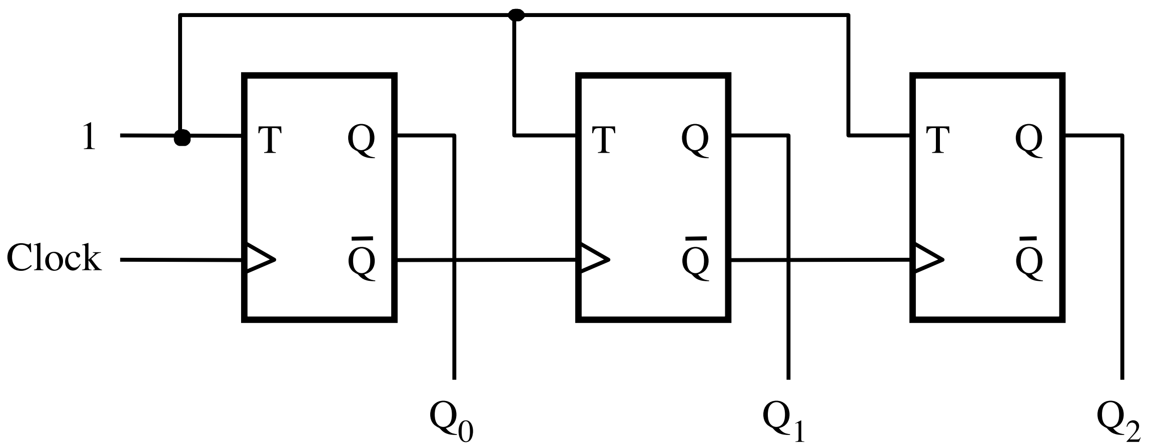


The count is formed by the Q’s

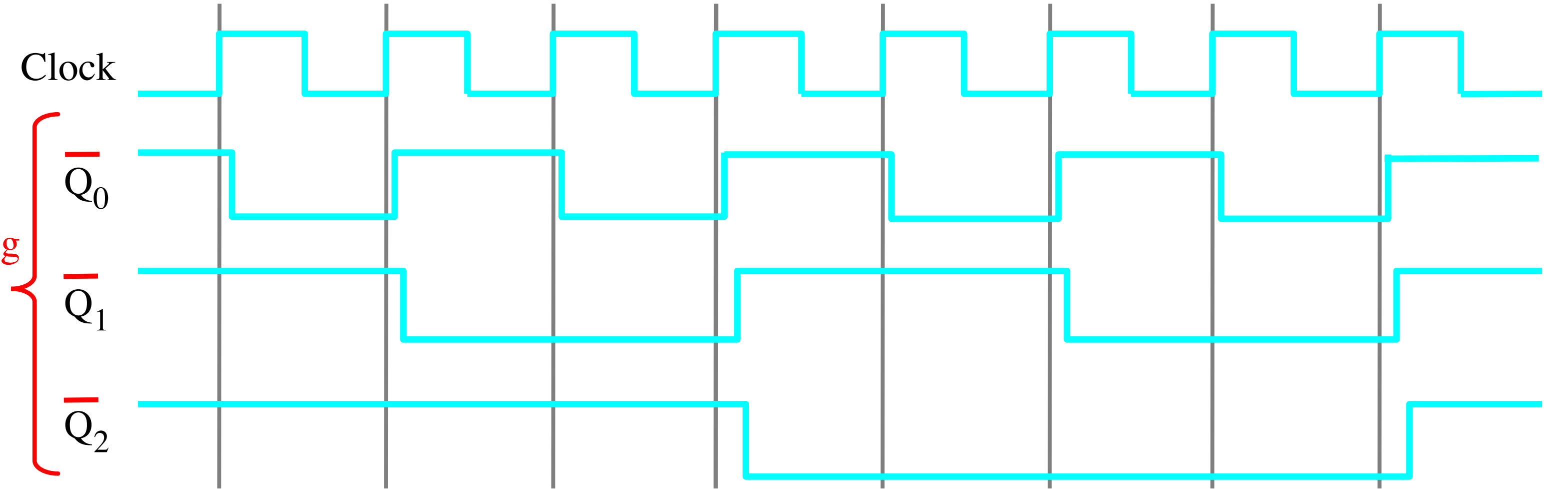
(b) Timing diagram

[Figure 5.19 from the textbook]

A three-bit up-counter



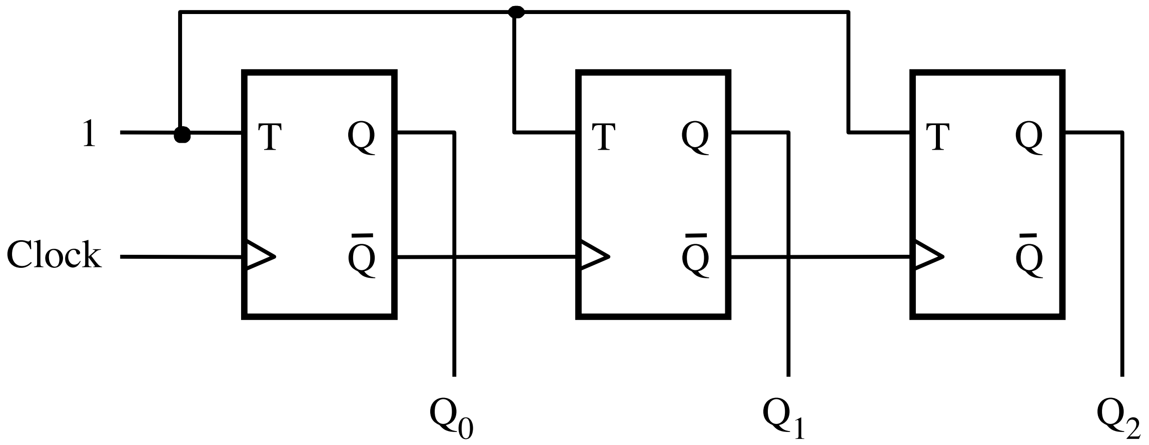
(a) Circuit



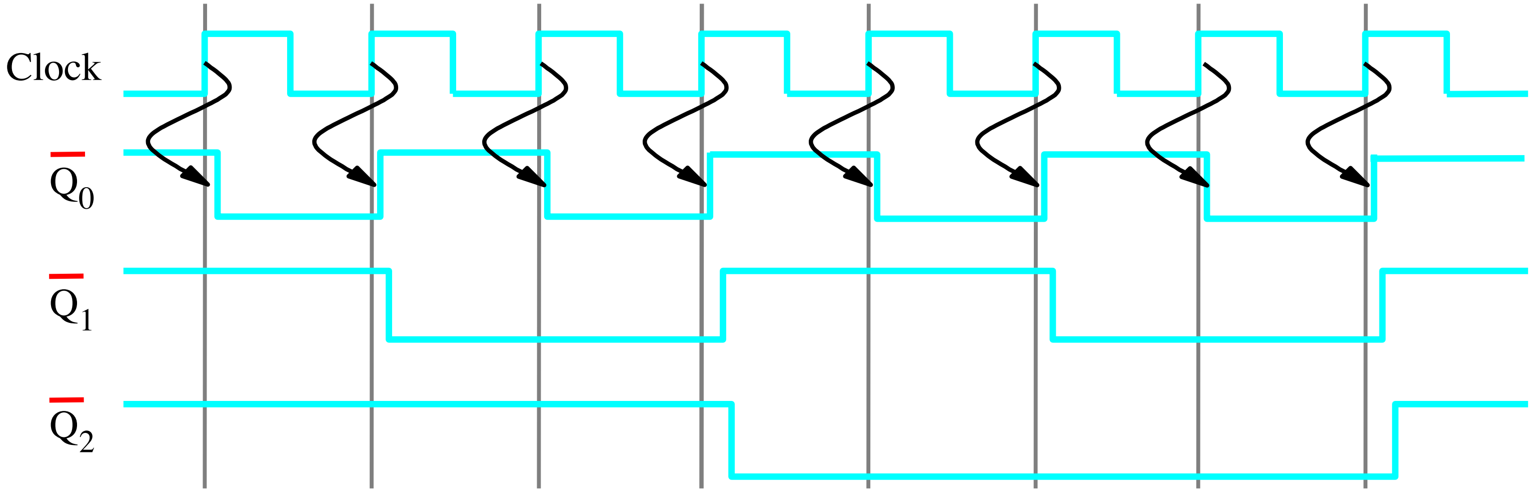
The toggling is done by the $\overline{Q}$'s

(b) Timing diagram

A three-bit up-counter



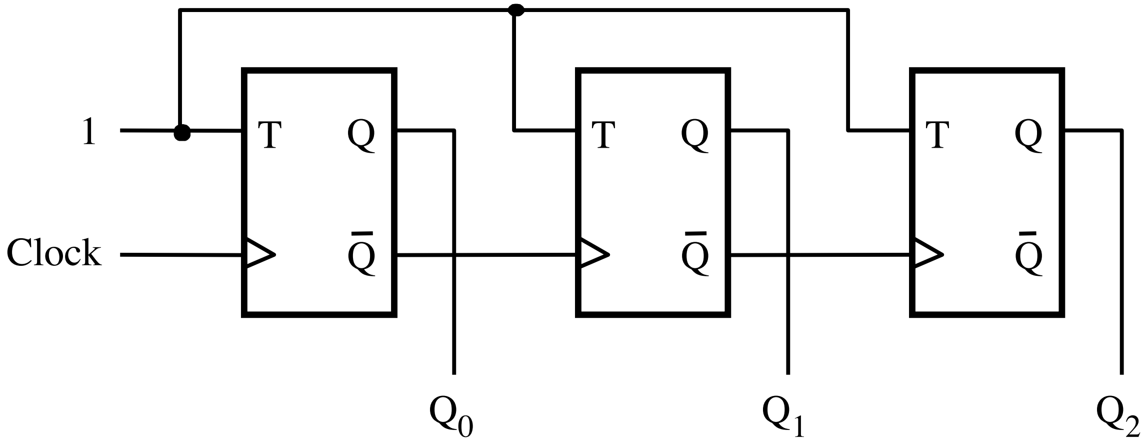
(a) Circuit



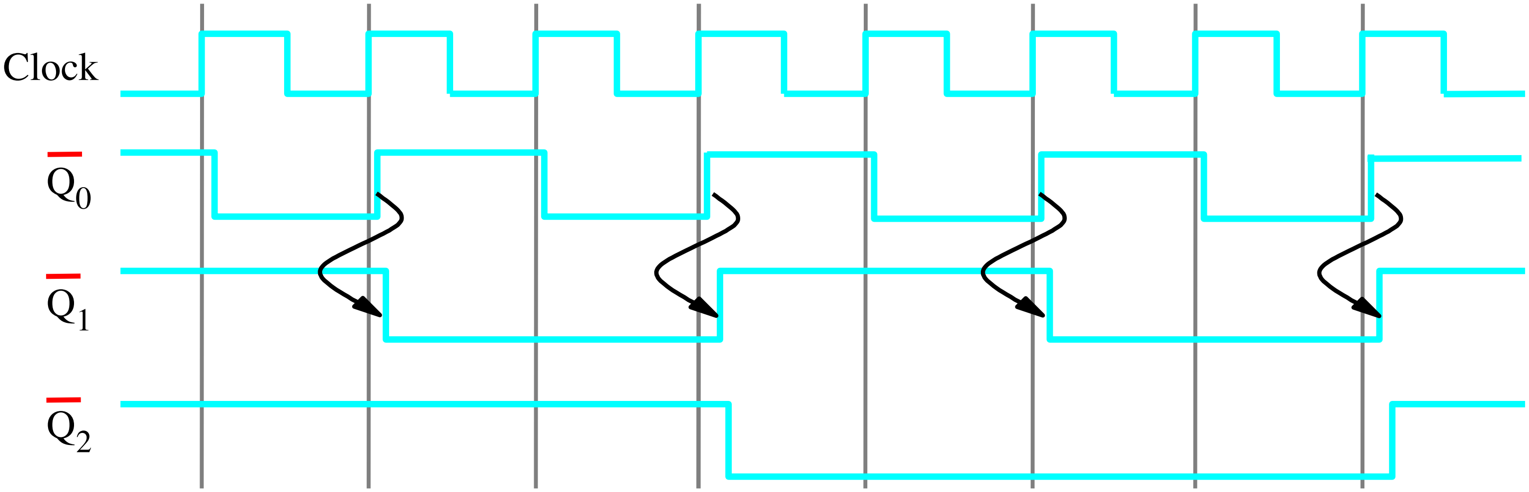
$\overline{Q}_0$ toggles on the positive clock edge

(b) Timing diagram

A three-bit up-counter



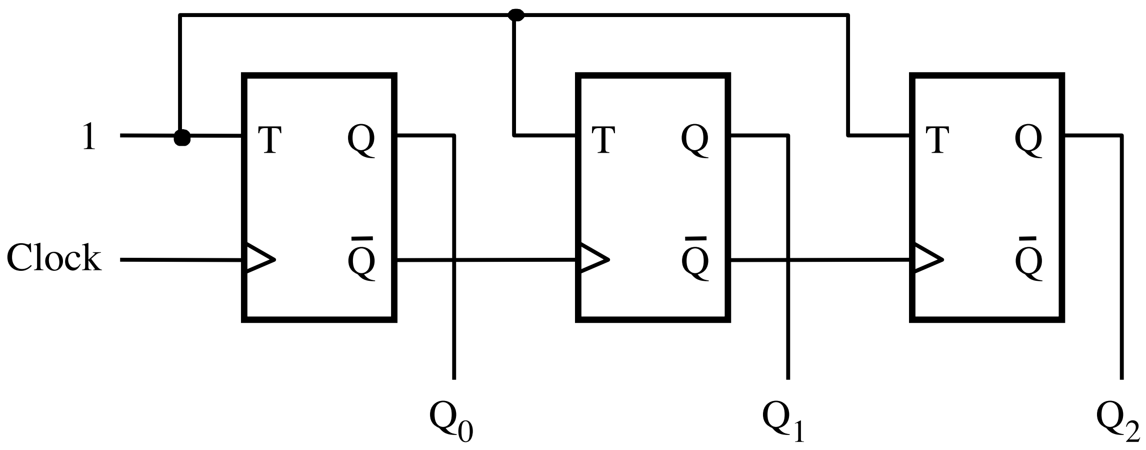
(a) Circuit



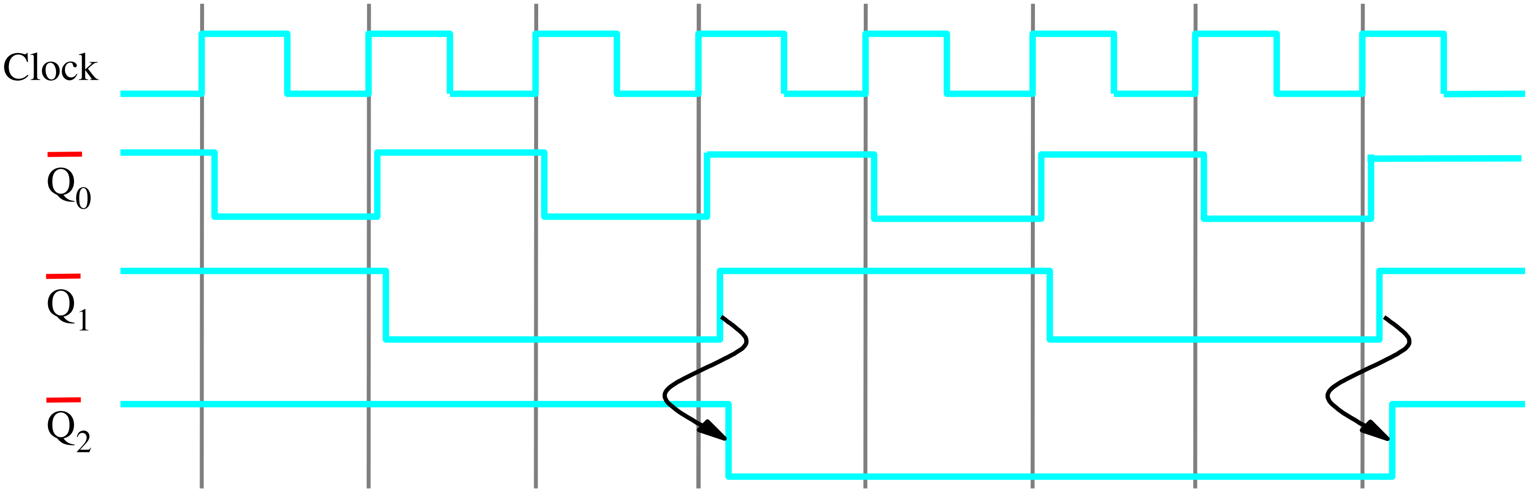
$\overline{Q_1}$ toggles on the positive Edge of $\overline{Q_0}$

(b) Timing diagram

A three-bit up-counter



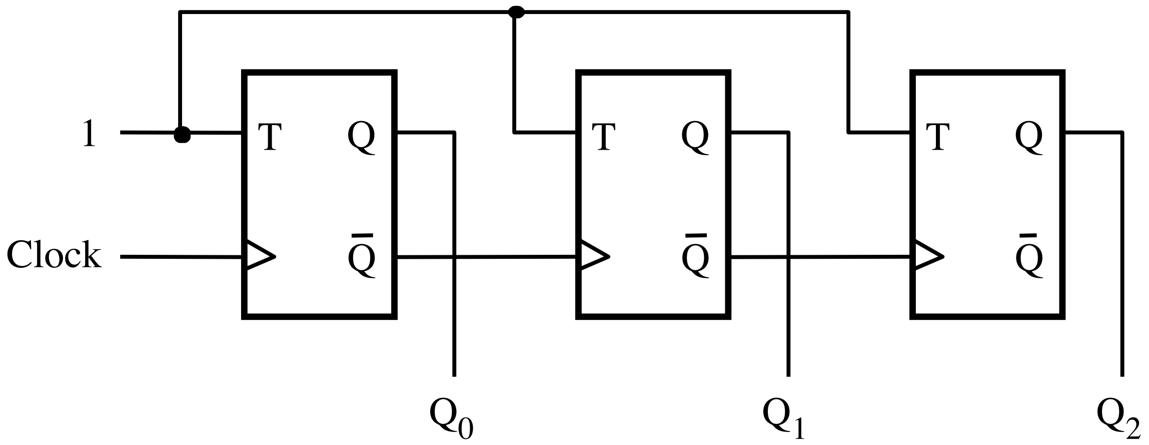
(a) Circuit



$\overline{Q}_2$ toggles on the positive edge of $\overline{Q}_1$

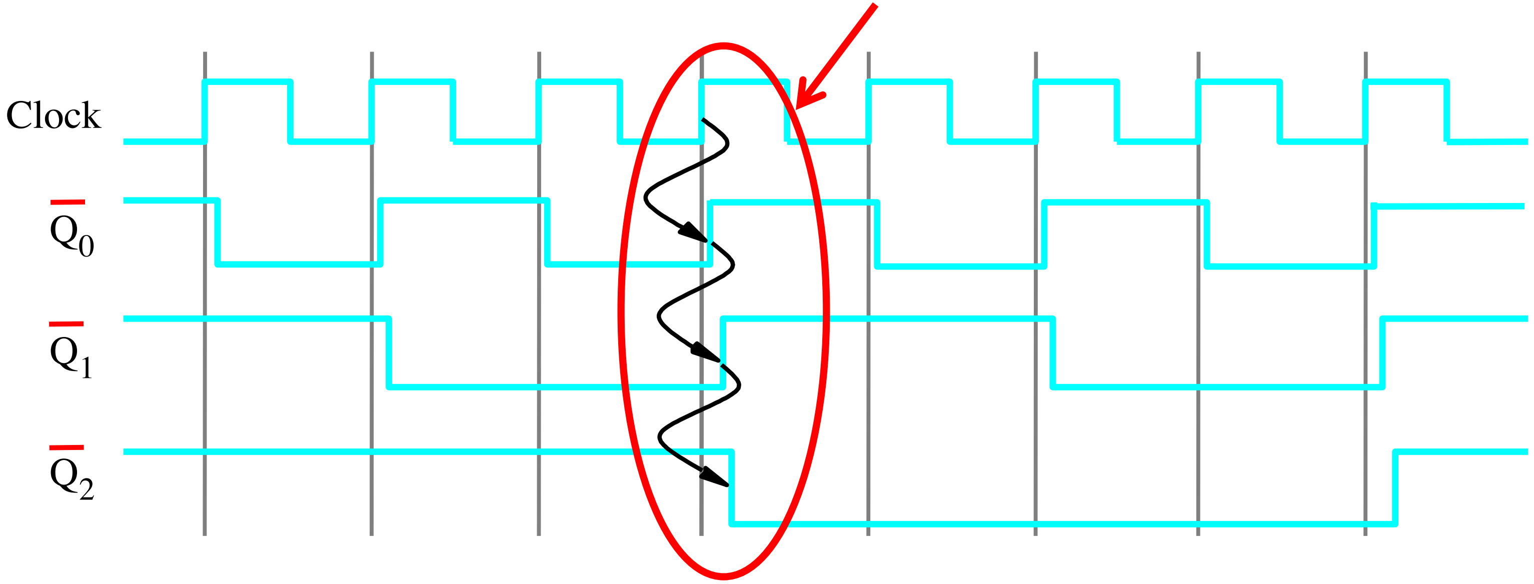
(b) Timing diagram

A three-bit up-counter



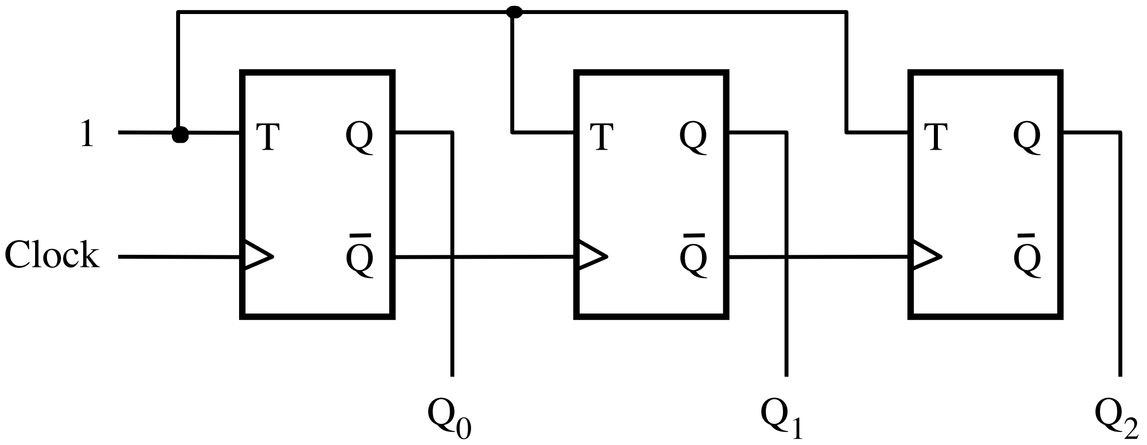
(a) Circuit

The propagation delays get longer



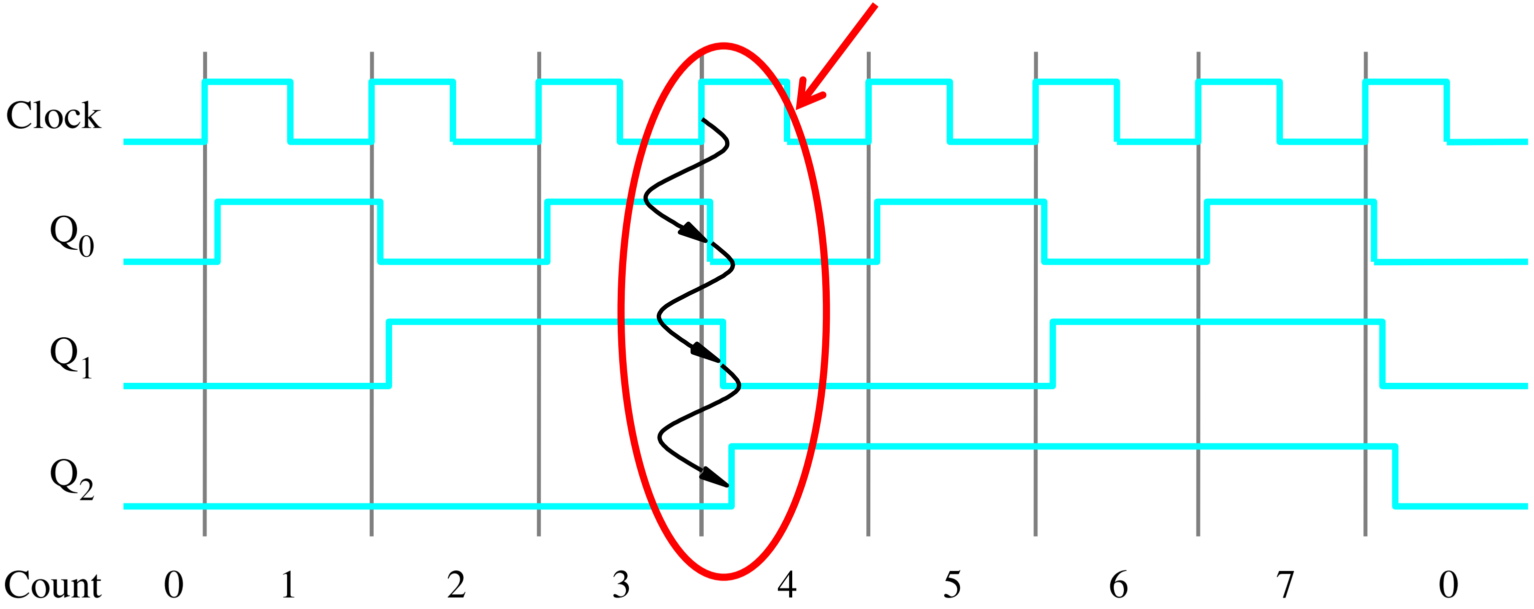
(b) Timing diagram

A three-bit up-counter



(a) Circuit

The propagation delays get longer

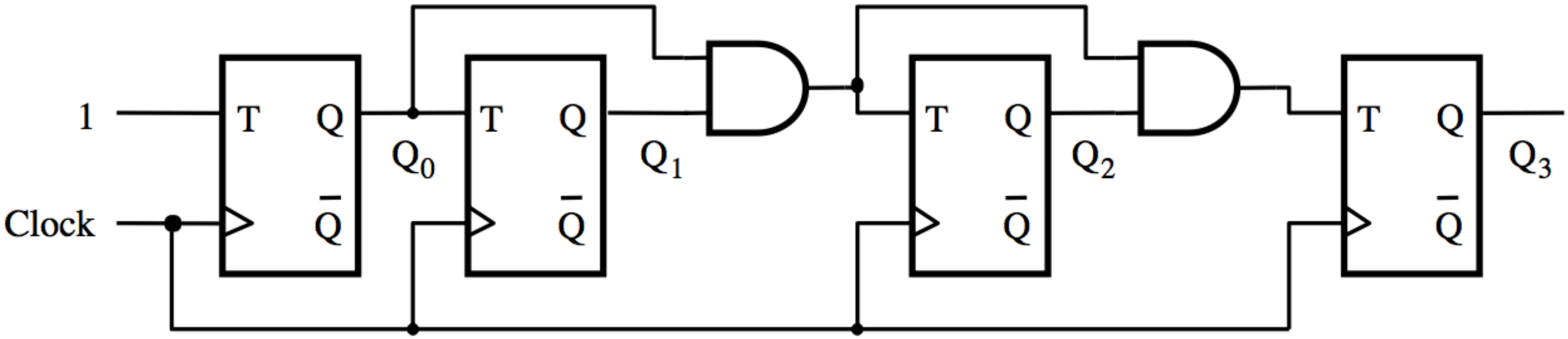


(b) Timing diagram

[Figure 5.19 from the textbook]

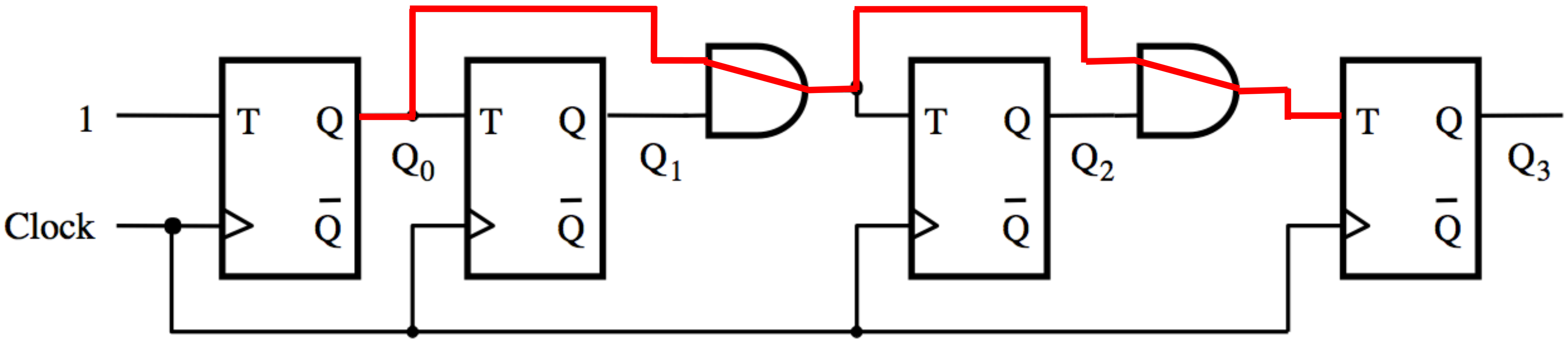
Synchronous Counters

A four-bit synchronous up-counter



[Figure 5.21 from the textbook]

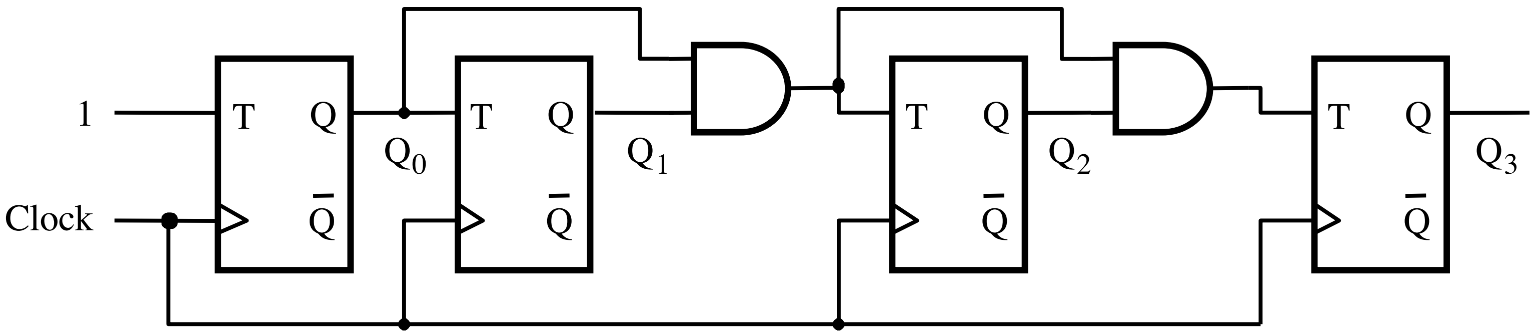
A four-bit synchronous up-counter



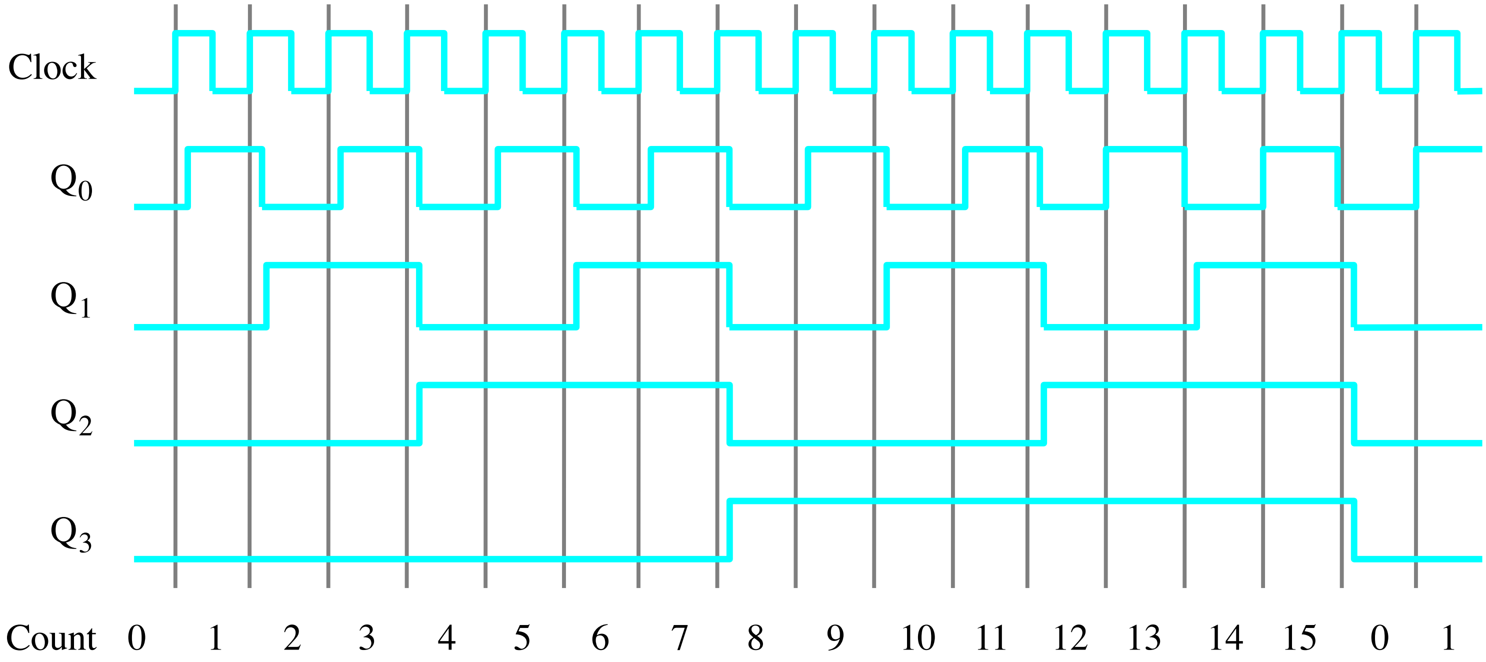
The propagation delay through all AND gates combined must not exceed the clock period minus the setup time for the flip-flops

[Figure 5.21 from the textbook]

A four-bit synchronous up-counter



(a) Circuit

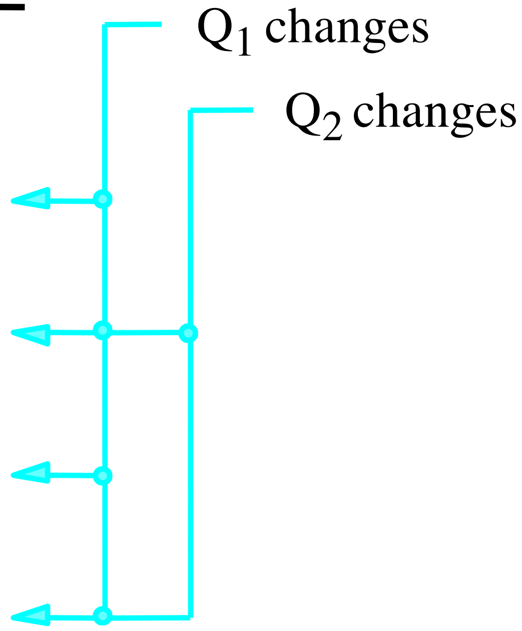


(b) Timing diagram

[Figure 5.21 from the textbook]

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



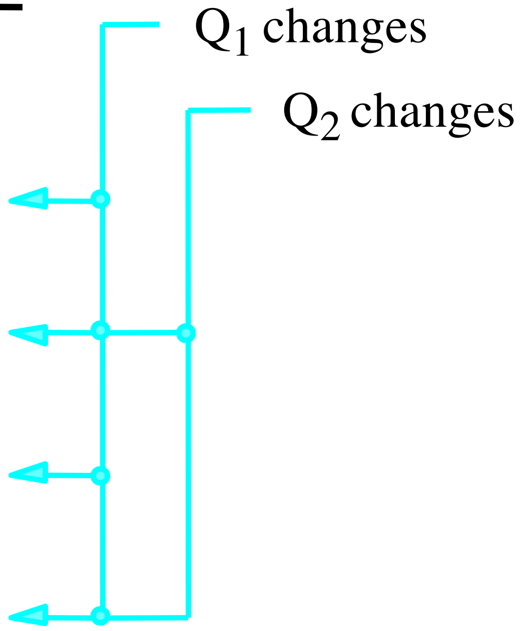
Q_1 changes

Q_2 changes

[Table 5.1 from the textbook]

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



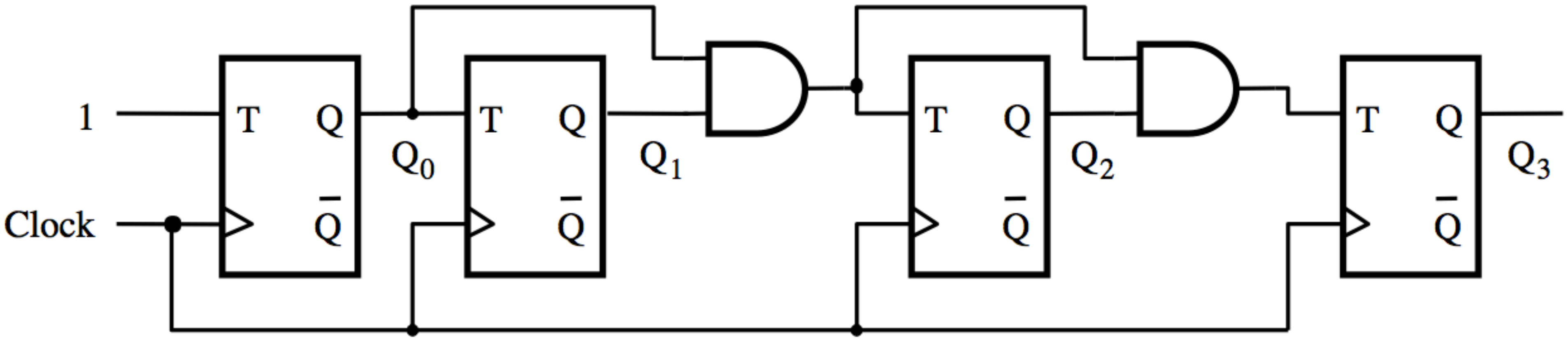
$T_0 = 1$

$T_1 = Q_0$

$T_2 = Q_0 Q_1$

[Table 5.1 from the textbook]

A four-bit synchronous up-counter



$T_0 = 1$

$T_1 = Q_0$

$T_2 = Q_0 Q_1$

[Figure 5.21 from the textbook]

In general we have

$T_0 = 1$

$T_1 = Q_0$

$T_2 = Q_0 Q_1$

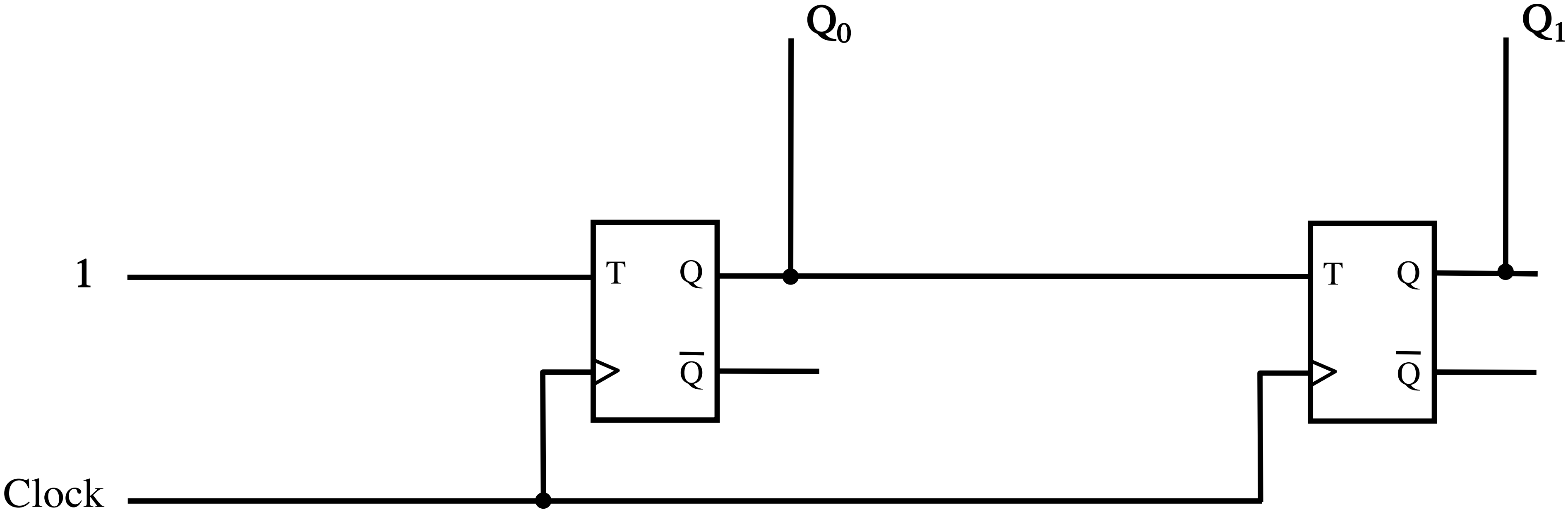
$T_3 = Q_0 Q_1 Q_2$

…

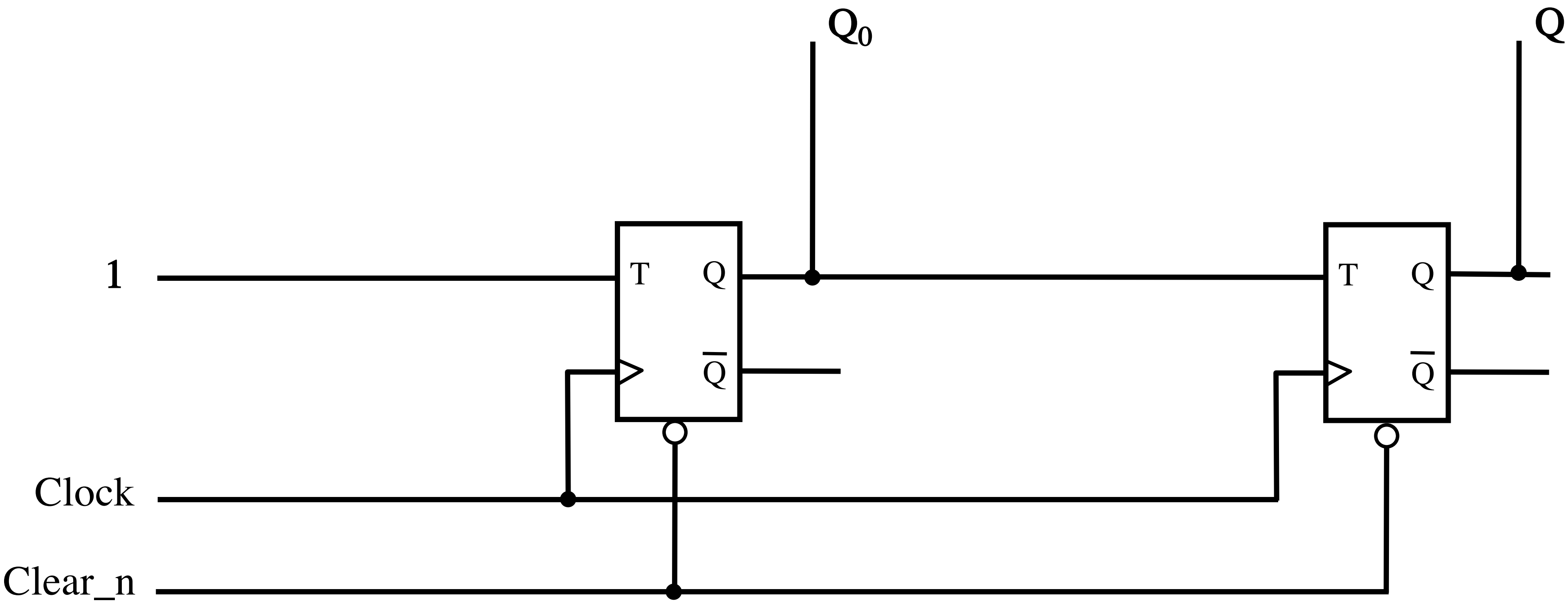
$T_n = Q_0 Q_1 Q_2 \ldots Q_{n-1}$

Synchronous v.s. Asynchronous Clear

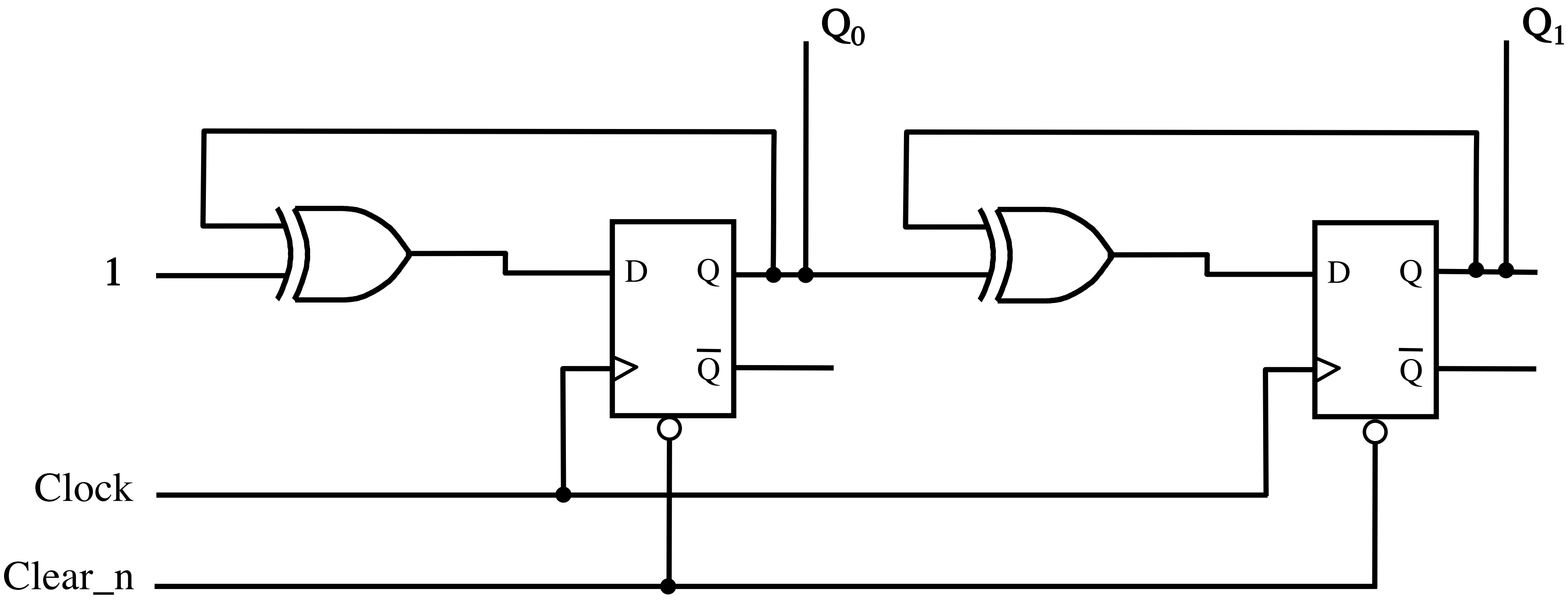
2-Bit Synchronous Up-Counter (without clear capability)



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

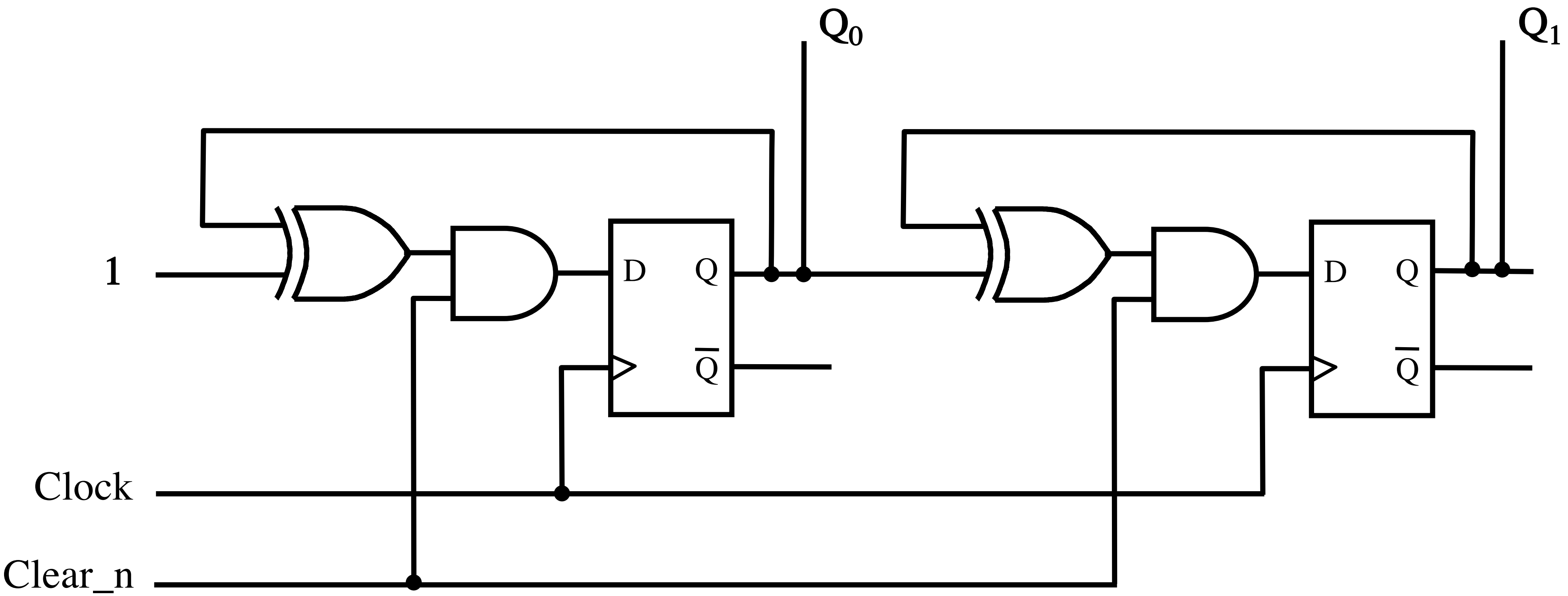


2-Bit Synchronous Up-Counter (with asynchronous clear)



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

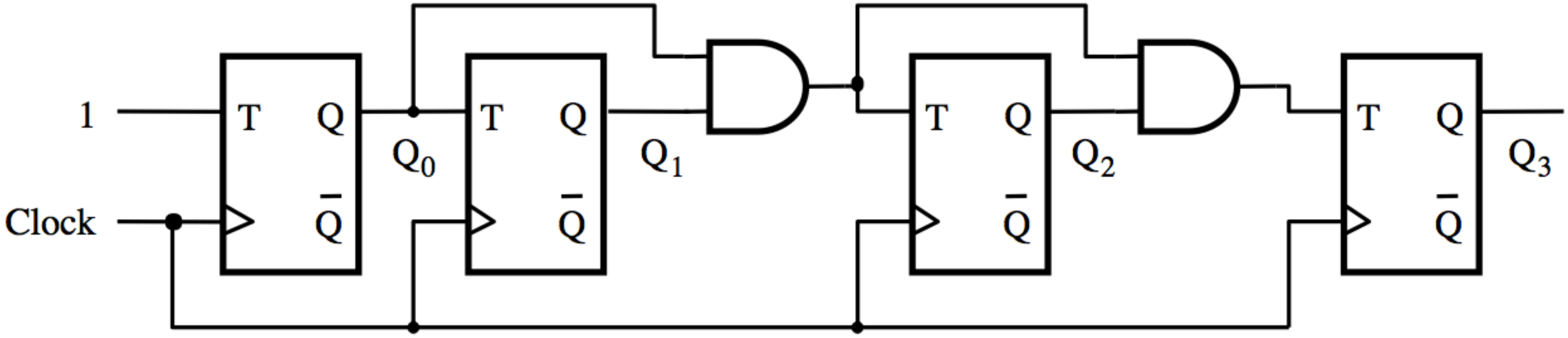
2-Bit Synchronous Up-Counter (with synchronous clear)



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

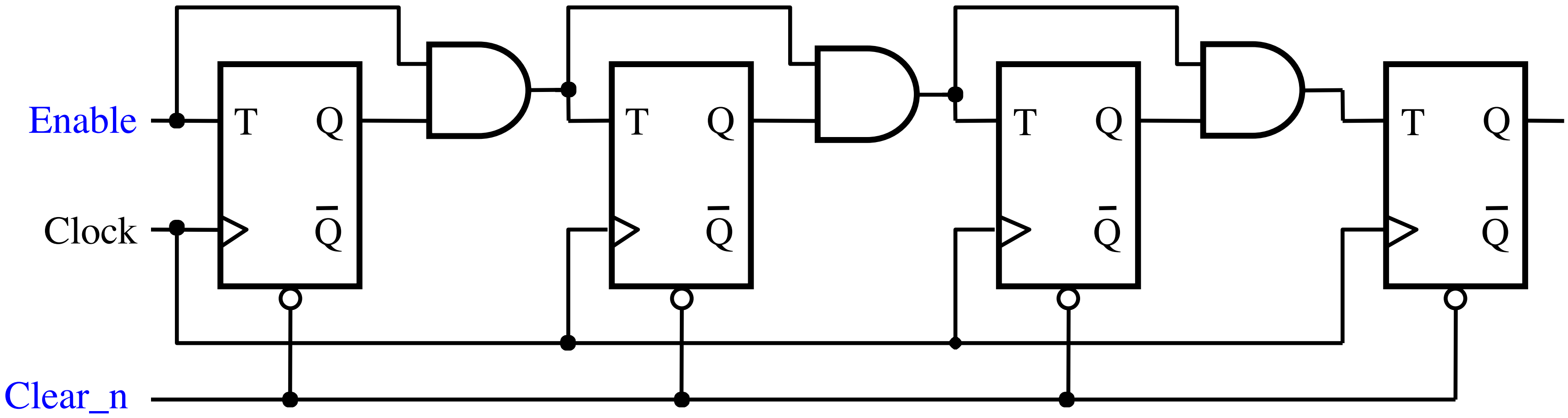
Adding Enable Capability

A four-bit synchronous up-counter



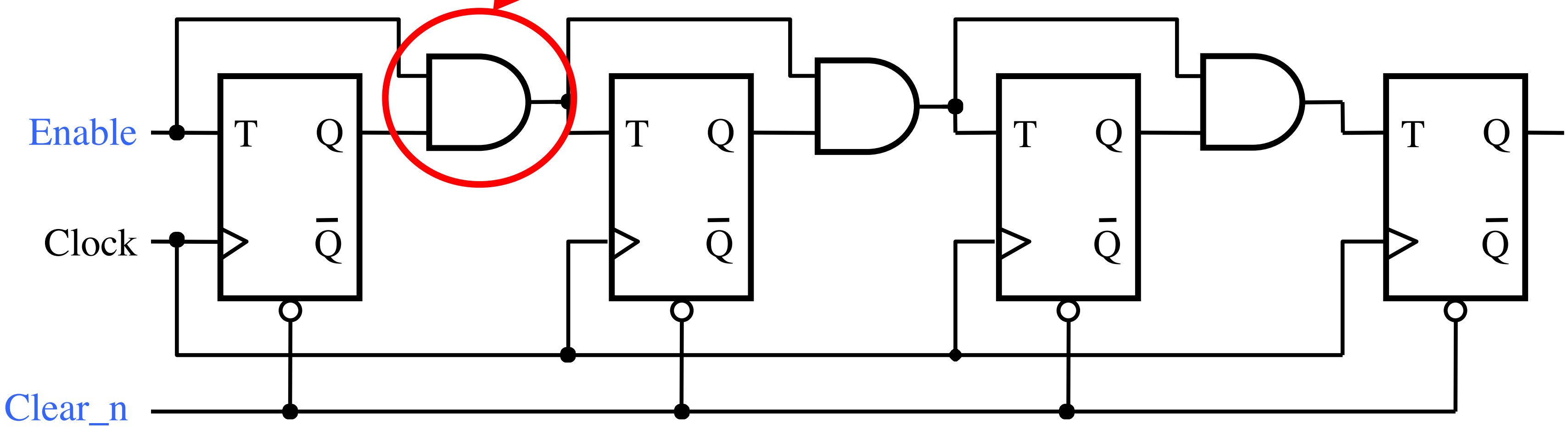
[Figure 5.21 from the textbook]

Inclusion of Enable and Clear Capability



[Figure 5.22 from the textbook]

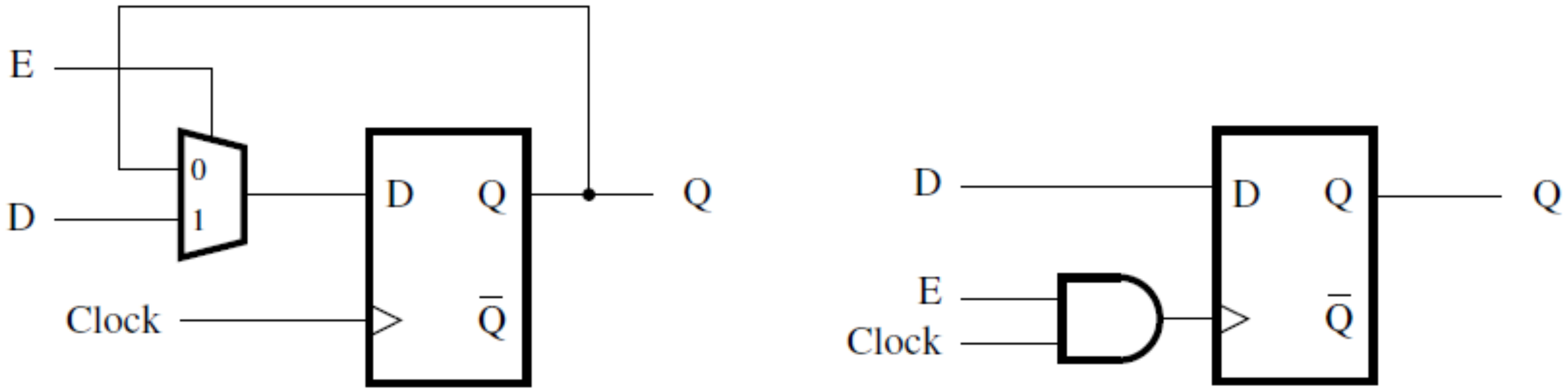
Inclusion of Enable and Clear Capability



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

[Figure 5.22 from the textbook]

Providing an enable input for a D flip-flop



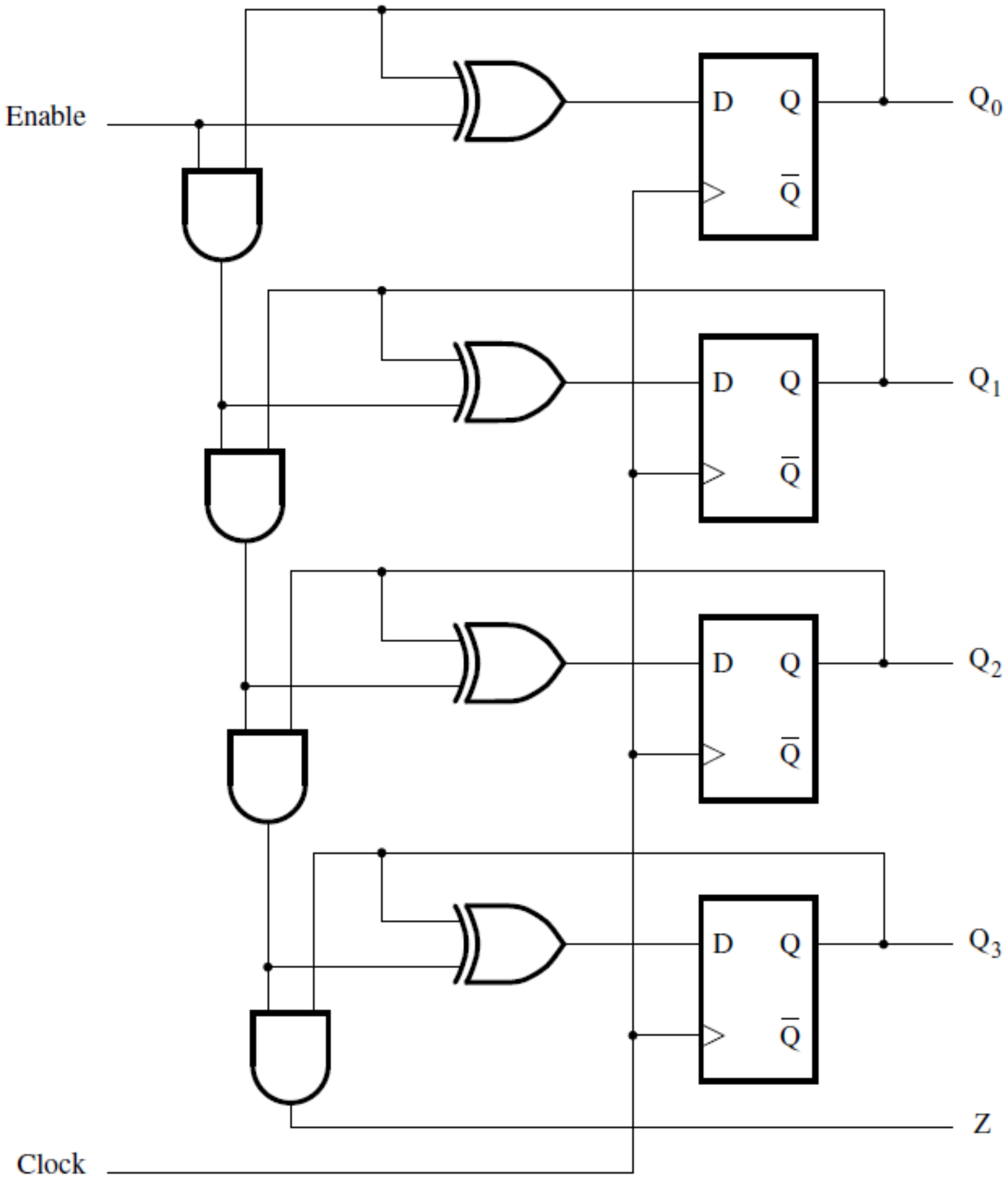
(a) Using a multiplexer

(b) Clock gating

[Figure 5.56 from the textbook]

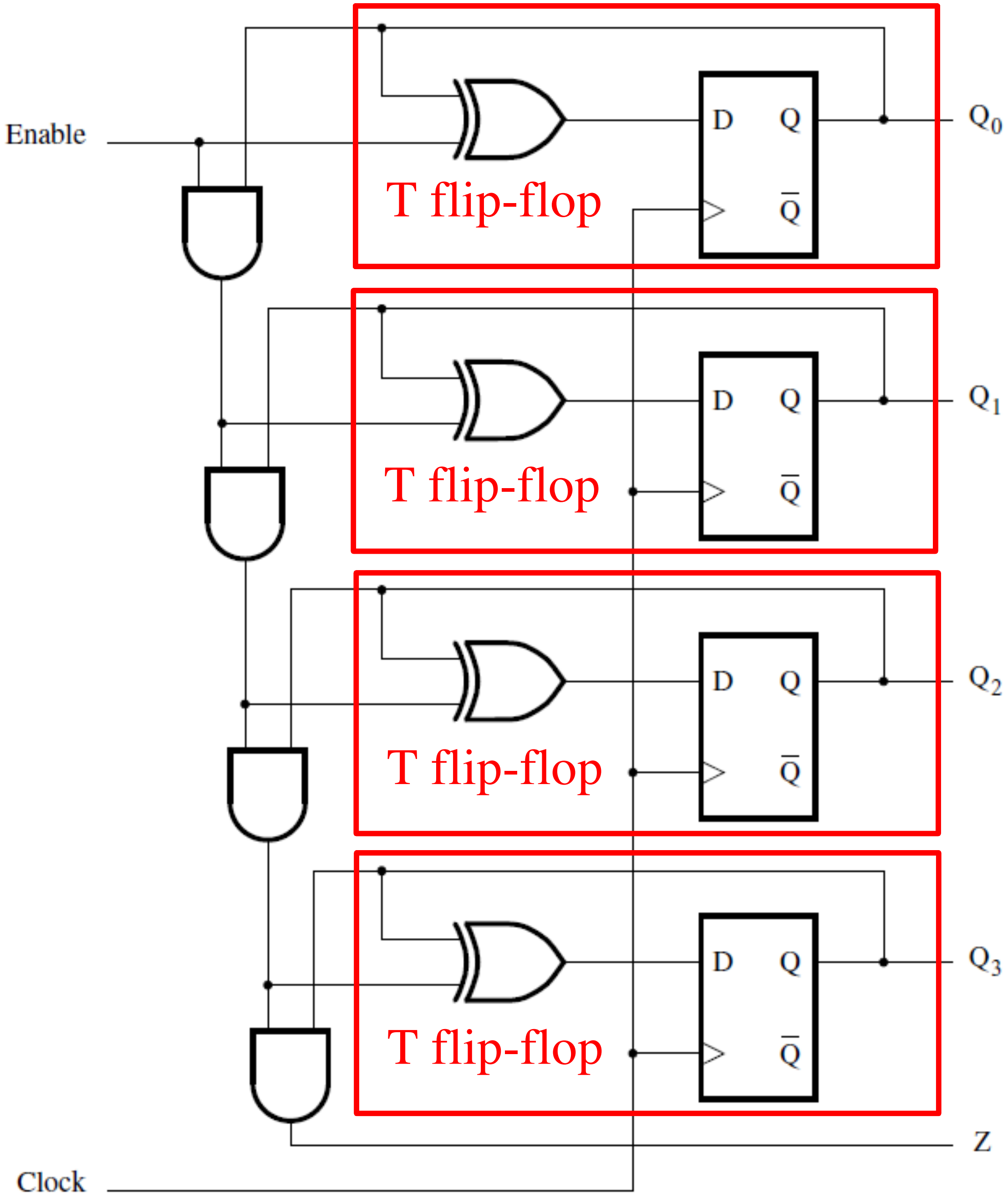
Synchronous Counter
(with D Flip-Flops)

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



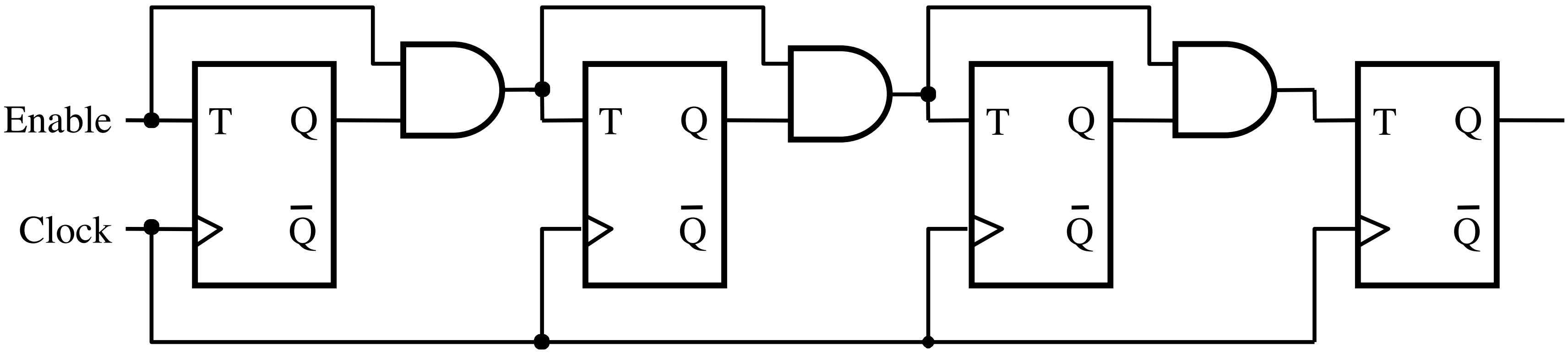
[Figure 5.23 from the textbook]

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

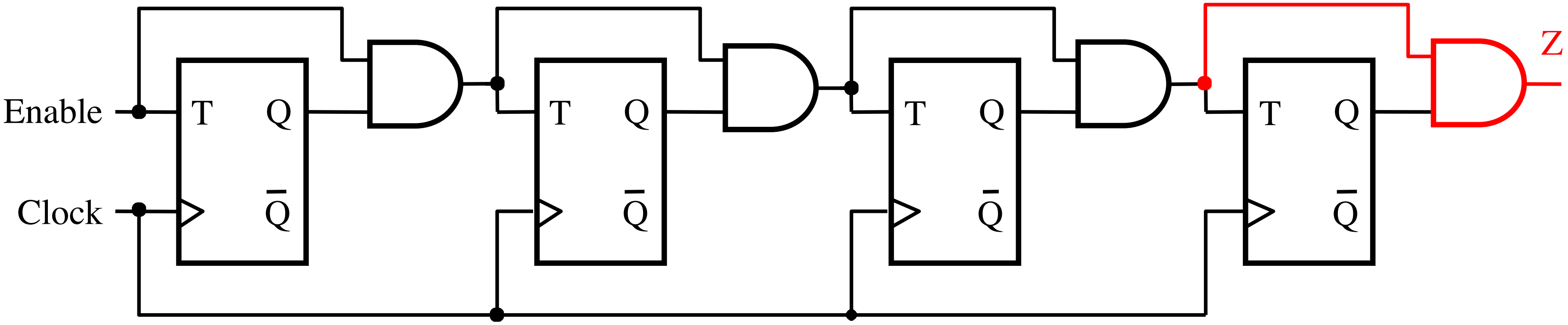


[Figure 5.23 from the textbook]

Equivalent to this circuit with T flip-flops



Equivalent to this circuit with T flip-flops

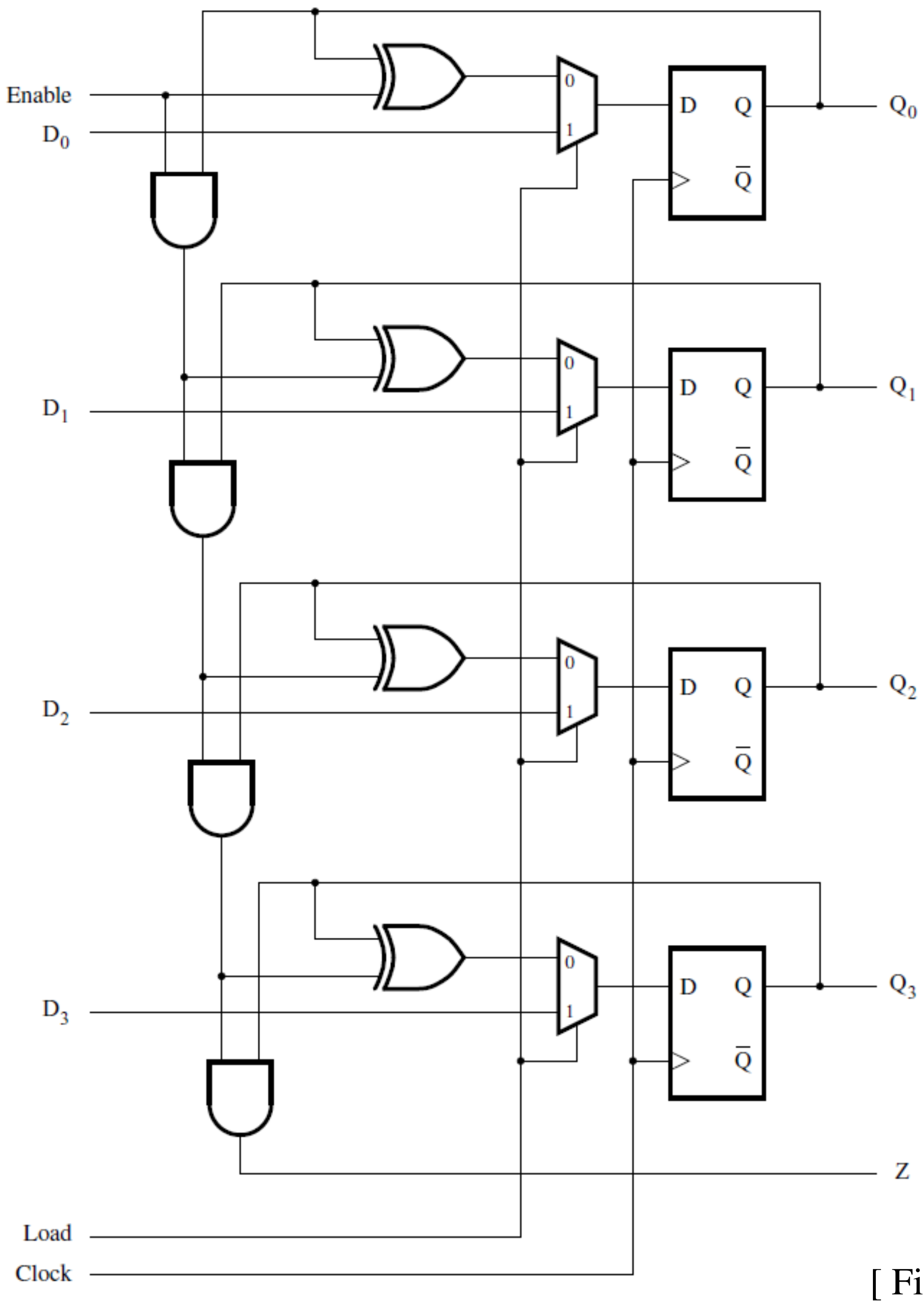


But has one extra output called Z, which can be used to connect two 4-bit counters to make an 8-bit counter.

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

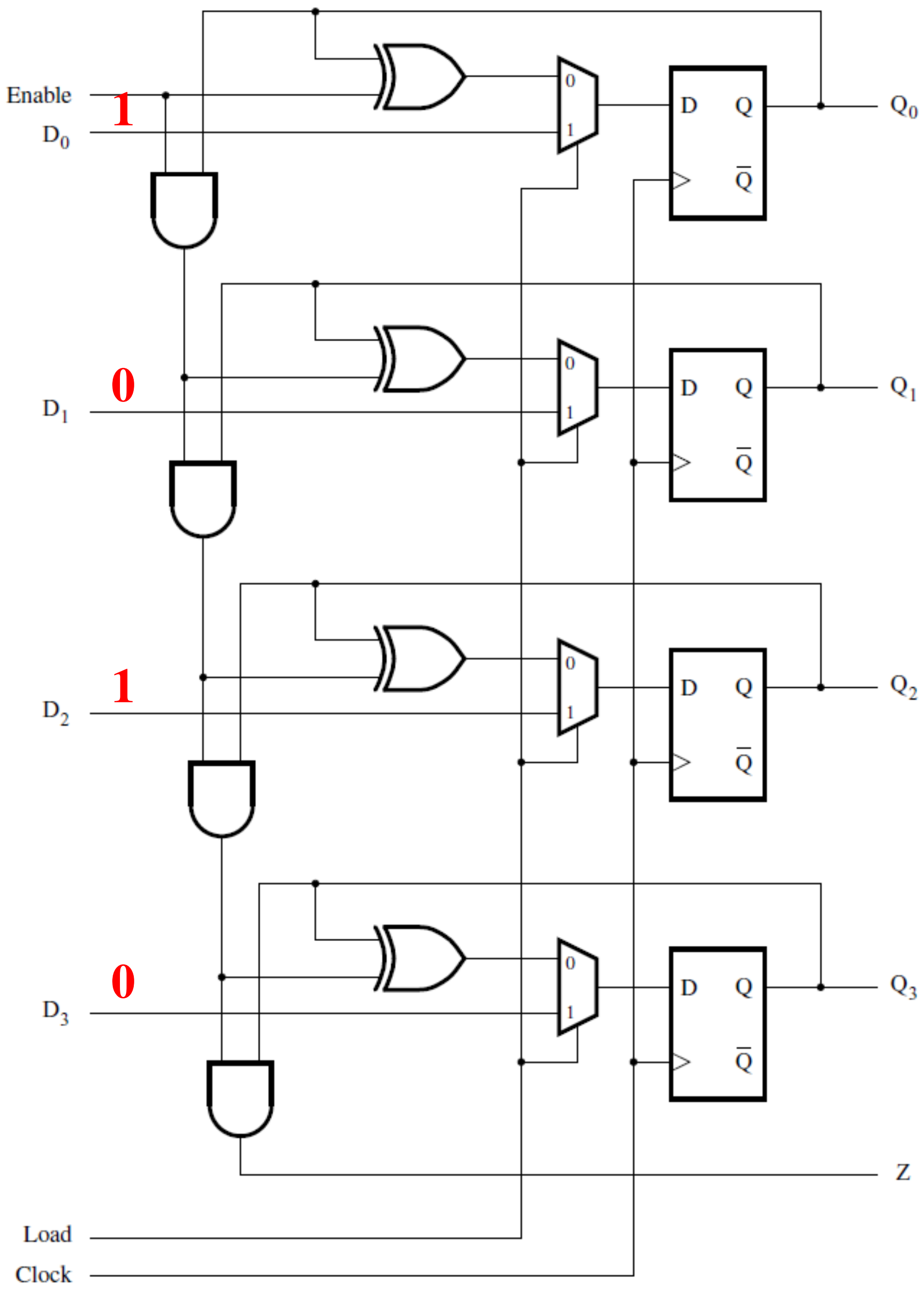
Counters with Parallel Load

A counter with parallel-load capability



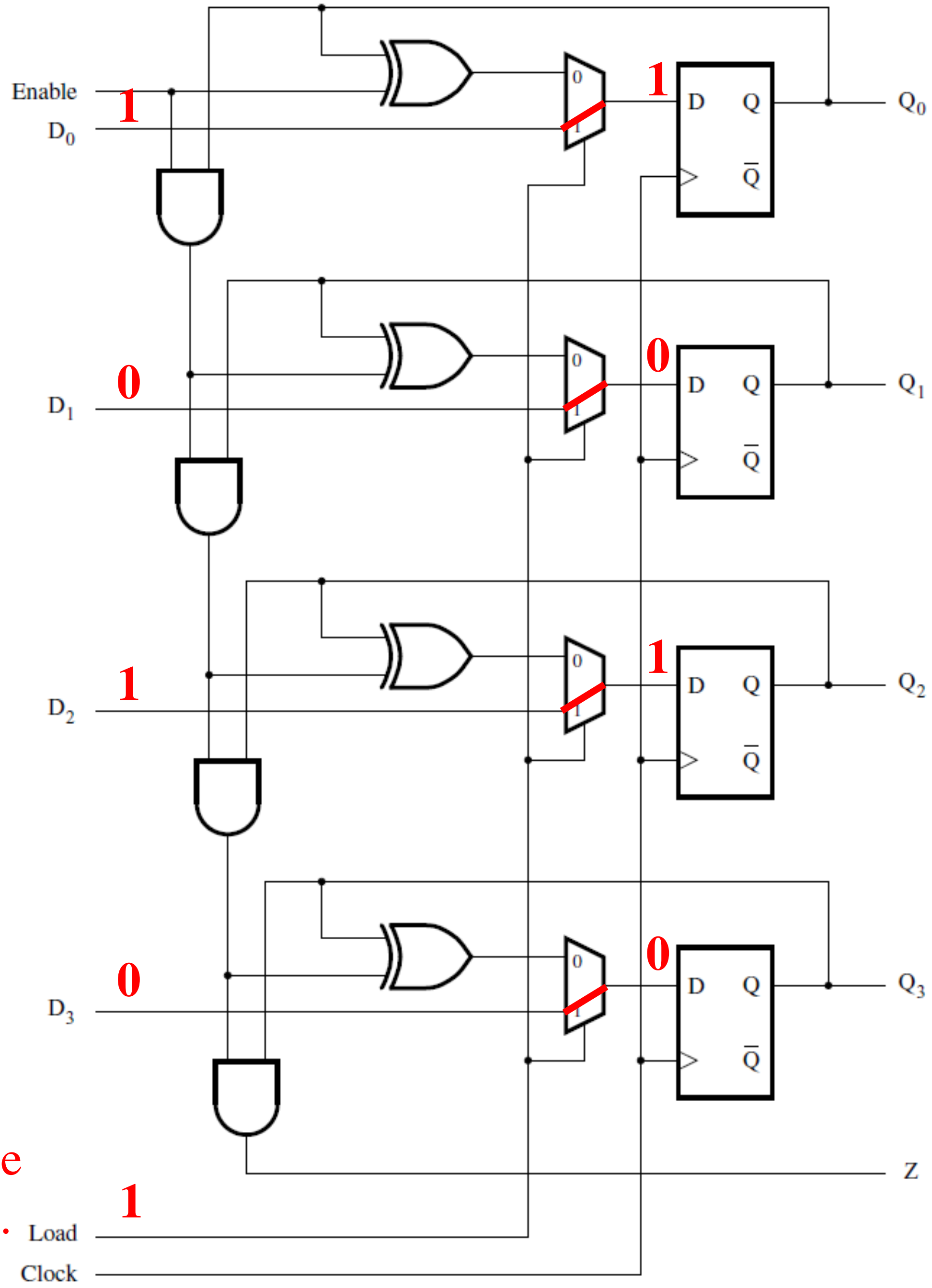
[Figure 5.24 from the textbook]

How to load the initial count value



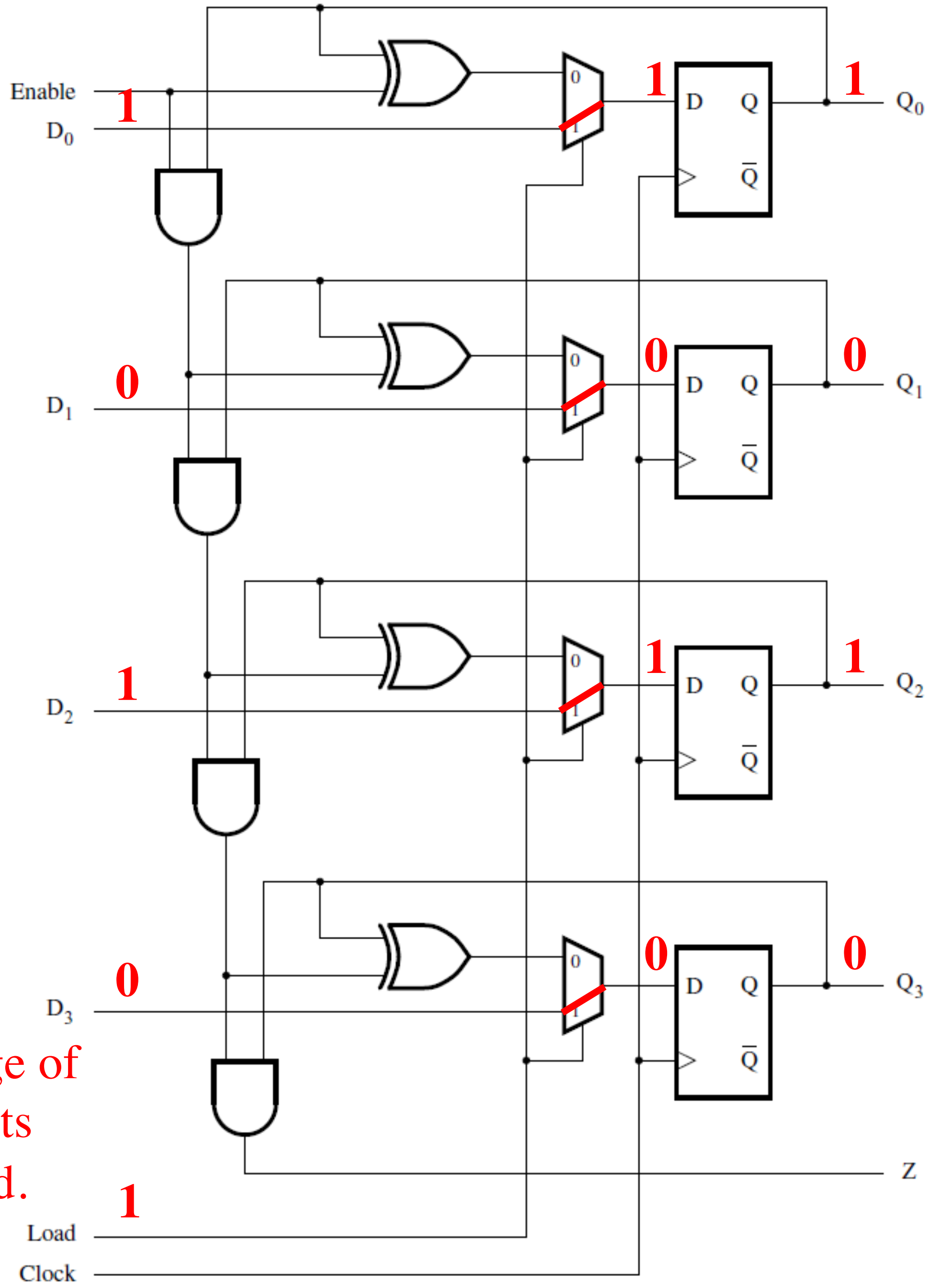
Set the initial count on the parallel load lines (in this case 5).

How to zero a counter



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

How to zero a counter



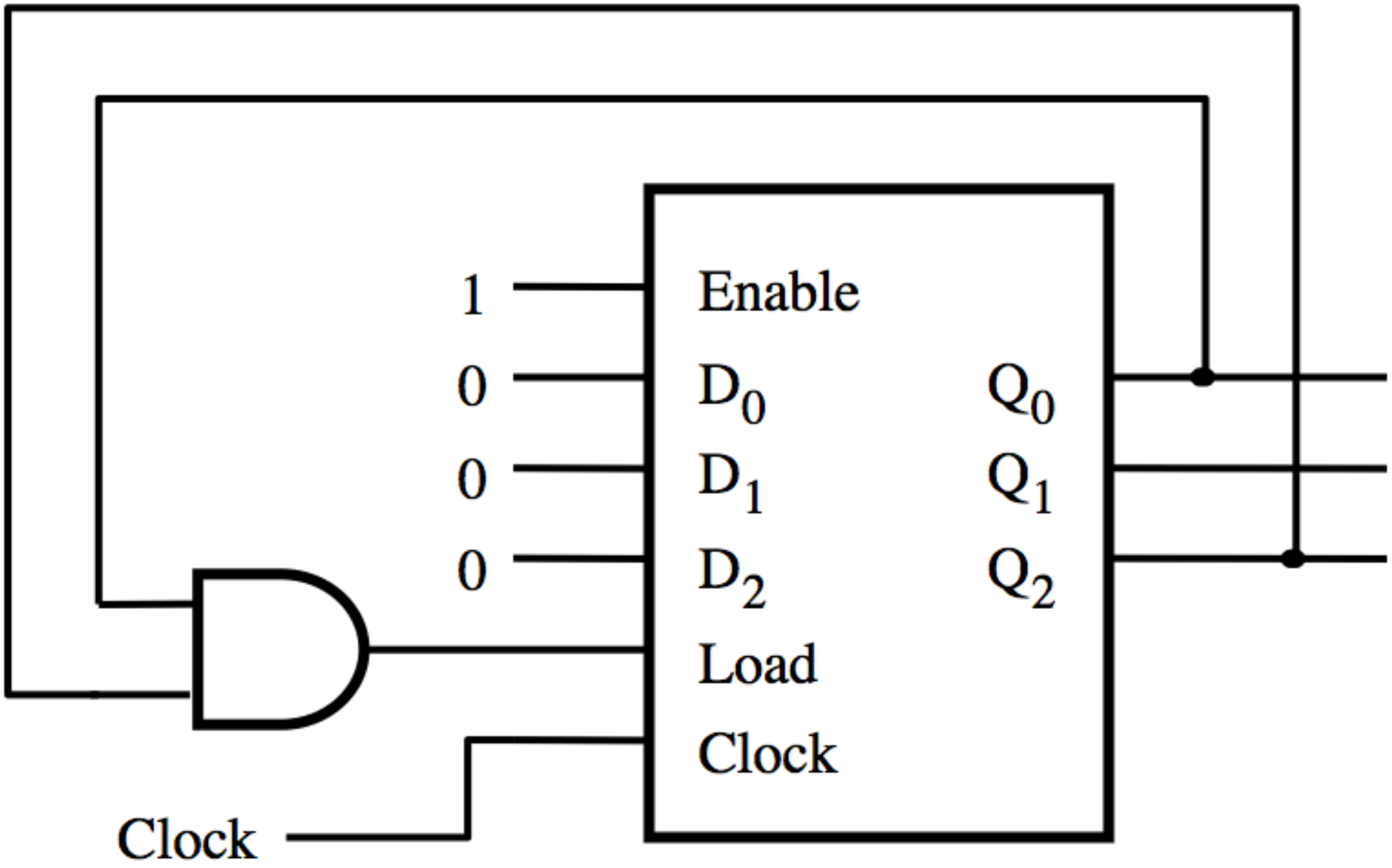
When the next positive edge of the clock arrives, the outputs of the flip-flops are updated.

Reset Synchronization

Motivation

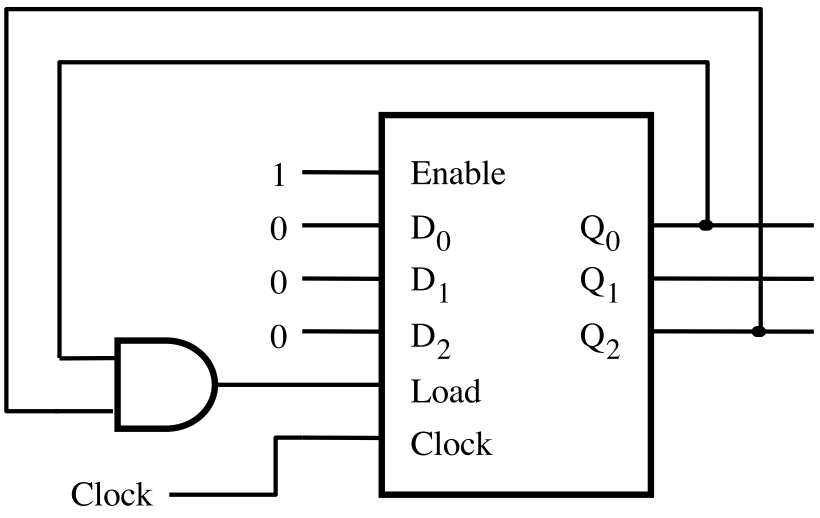
- An n-bit counter counts from 0, 1, …, 2^n-1
- For example a 3-bit counter counts up as follow
  - 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, …
- What if we want it to count like this
  - 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, …
- In other words, what is the cycle is not a power of 2?

What does this circuit do?

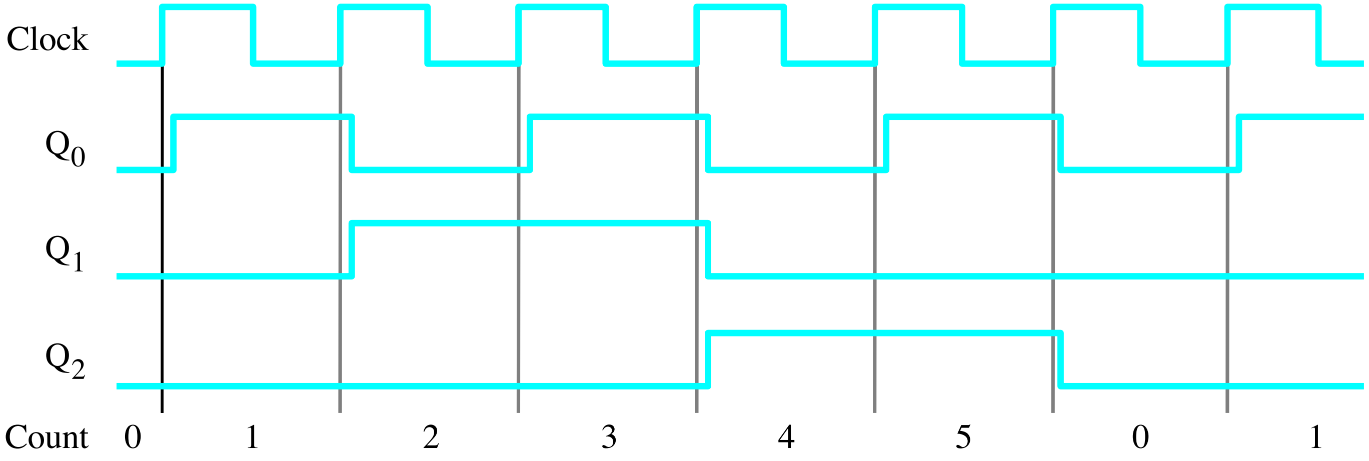


[Figure 5.25a from the textbook]

A modulo-6 counter with synchronous reset



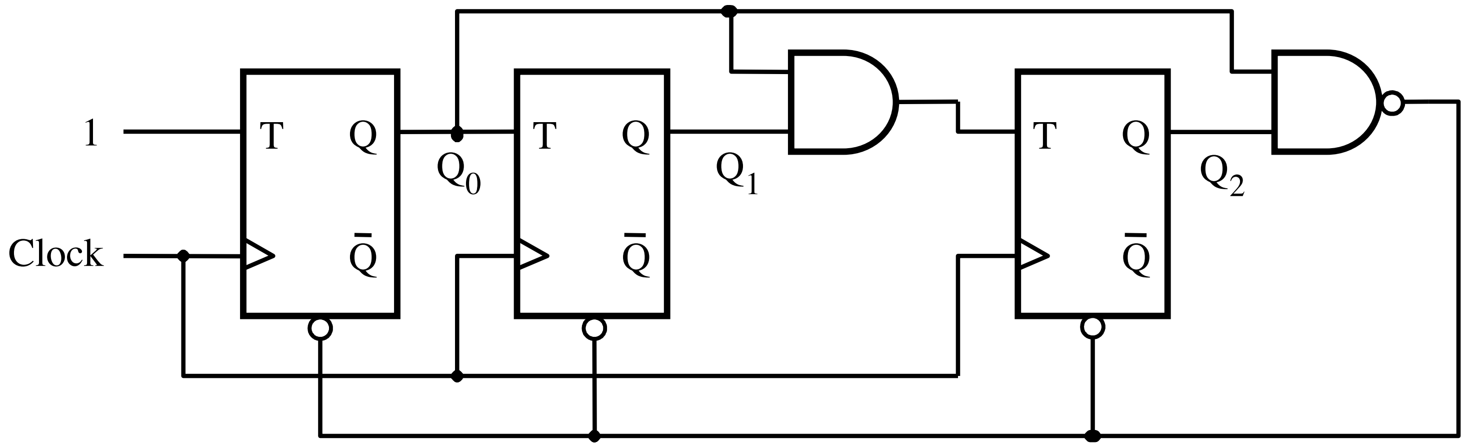
(a) Circuit



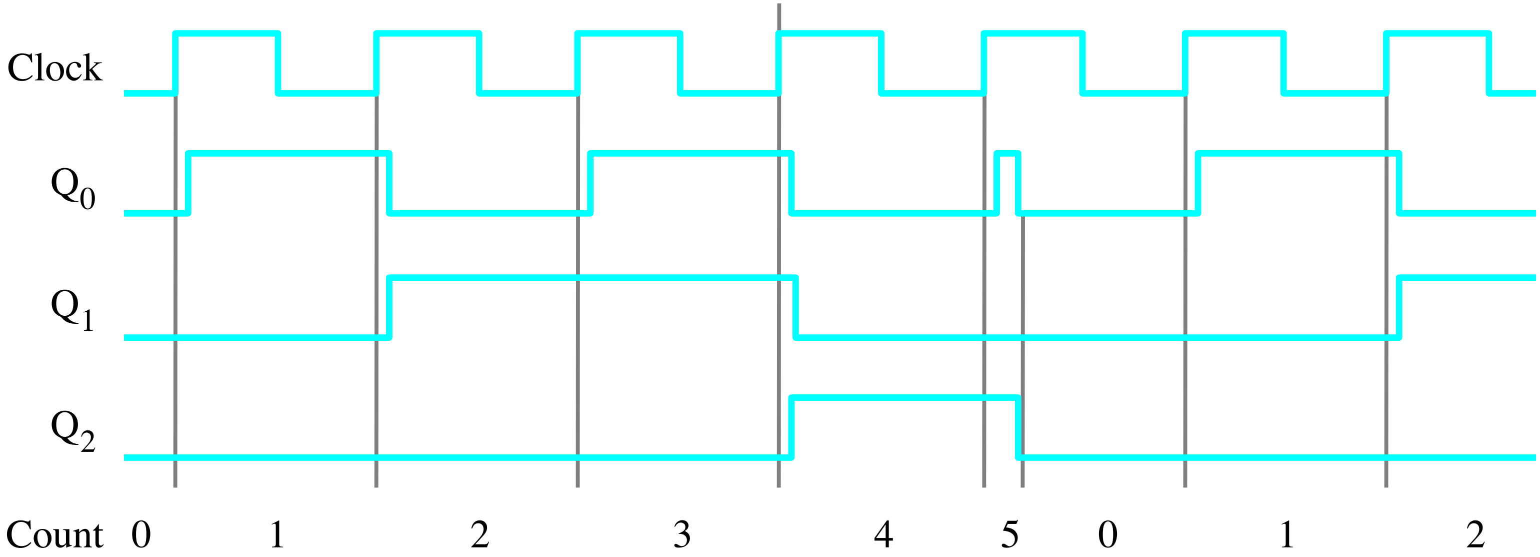
(b) Timing diagram

[Figure 5.25 from the textbook]

A modulo-6 counter with asynchronous reset



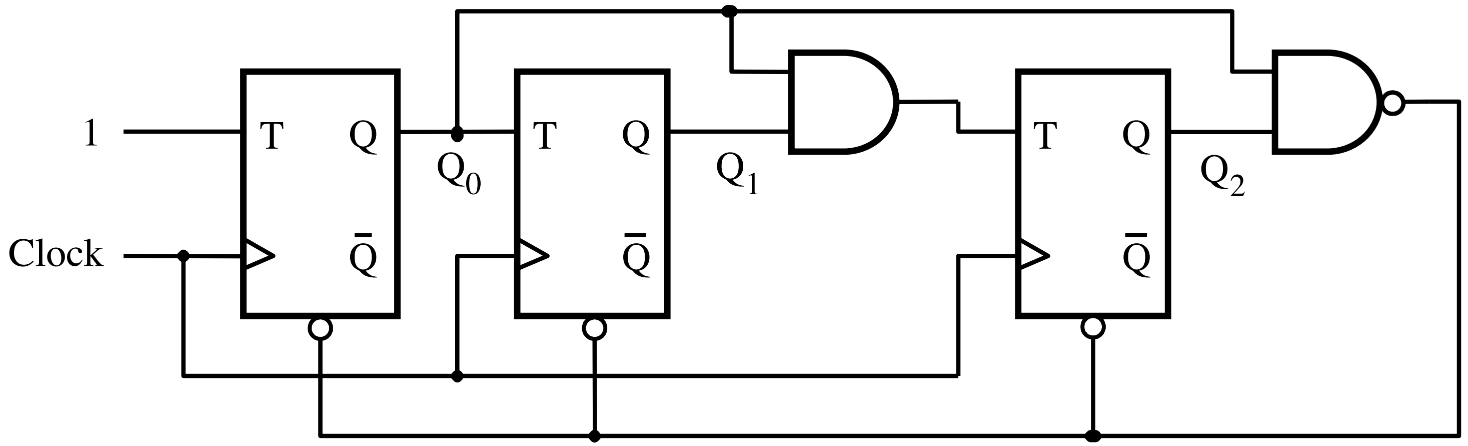
(a) Circuit



(b) Timing diagram

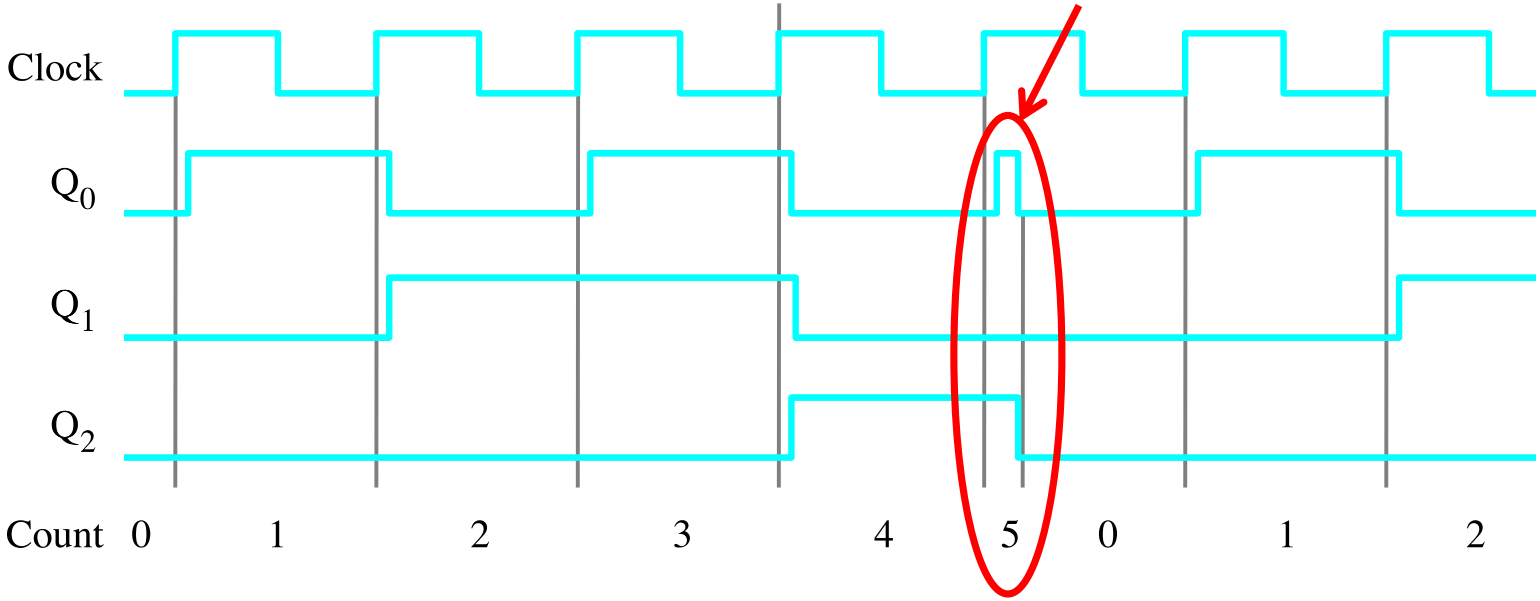
[Figure 5.26 from the textbook]

A modulo-6 counter with asynchronous reset



(a) Circuit

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



(b) Timing diagram

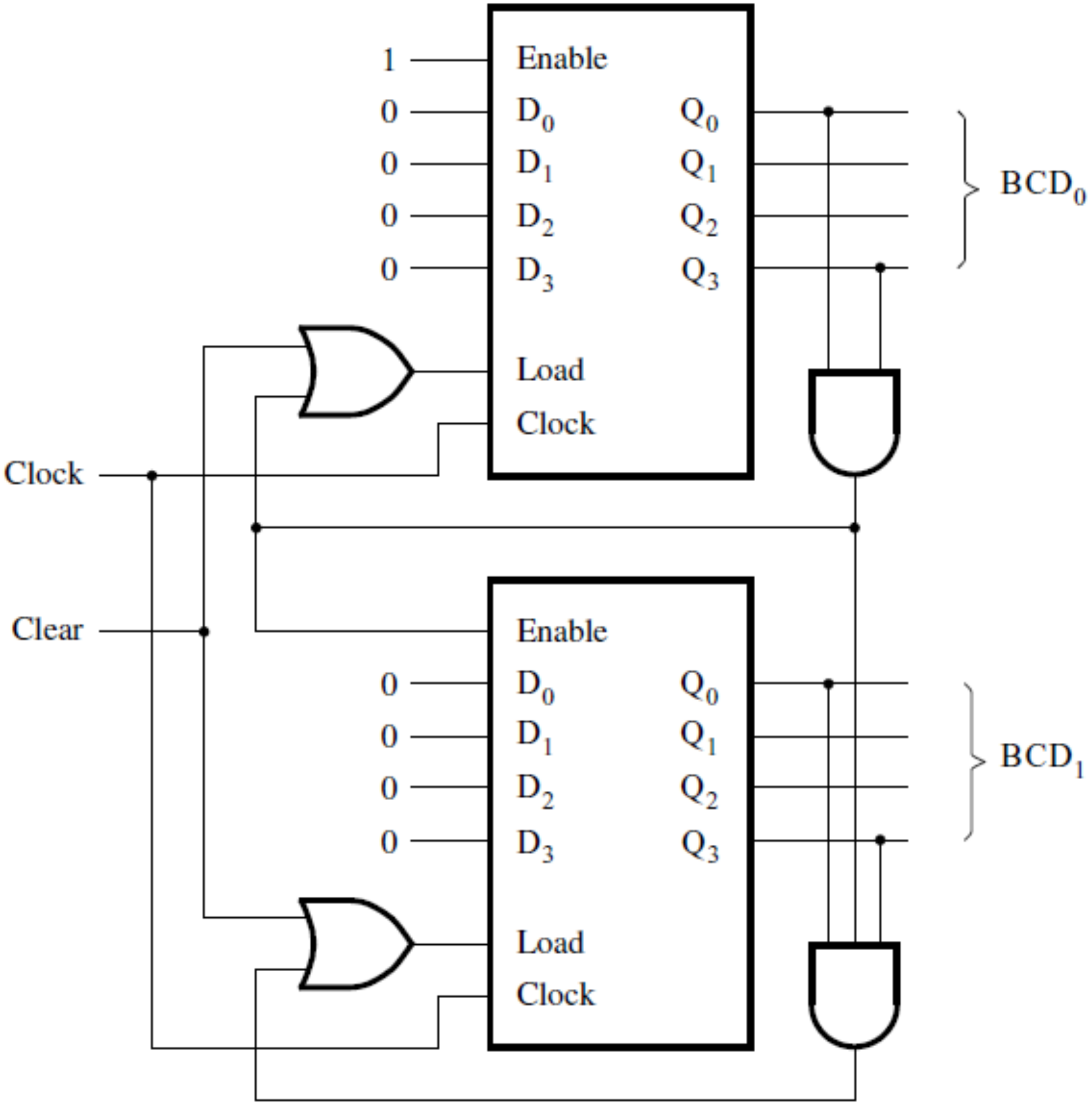
[Figure 5.26 from the textbook]

Other Types of Counters
(Section 5.11)

A two-digit BCD counter

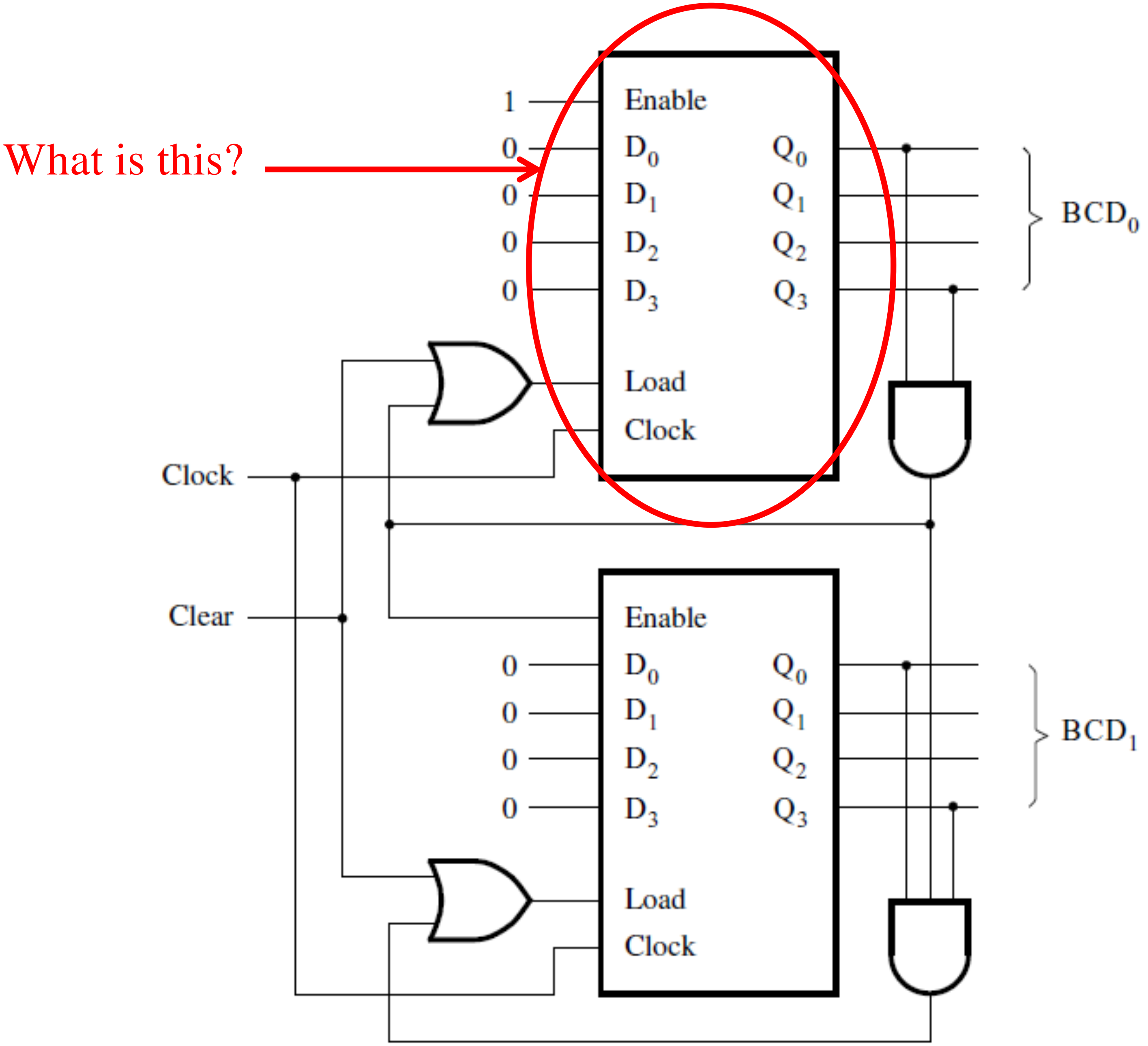
- 2: Parallel-load four-bit counter
  - Figure 5.24
- Each counts in binary
  - 0-9
- Resets generated on 9
  - Reset by loading 0's
- Second digit enabled by a 9 on first counter

A two-digit BCD counter



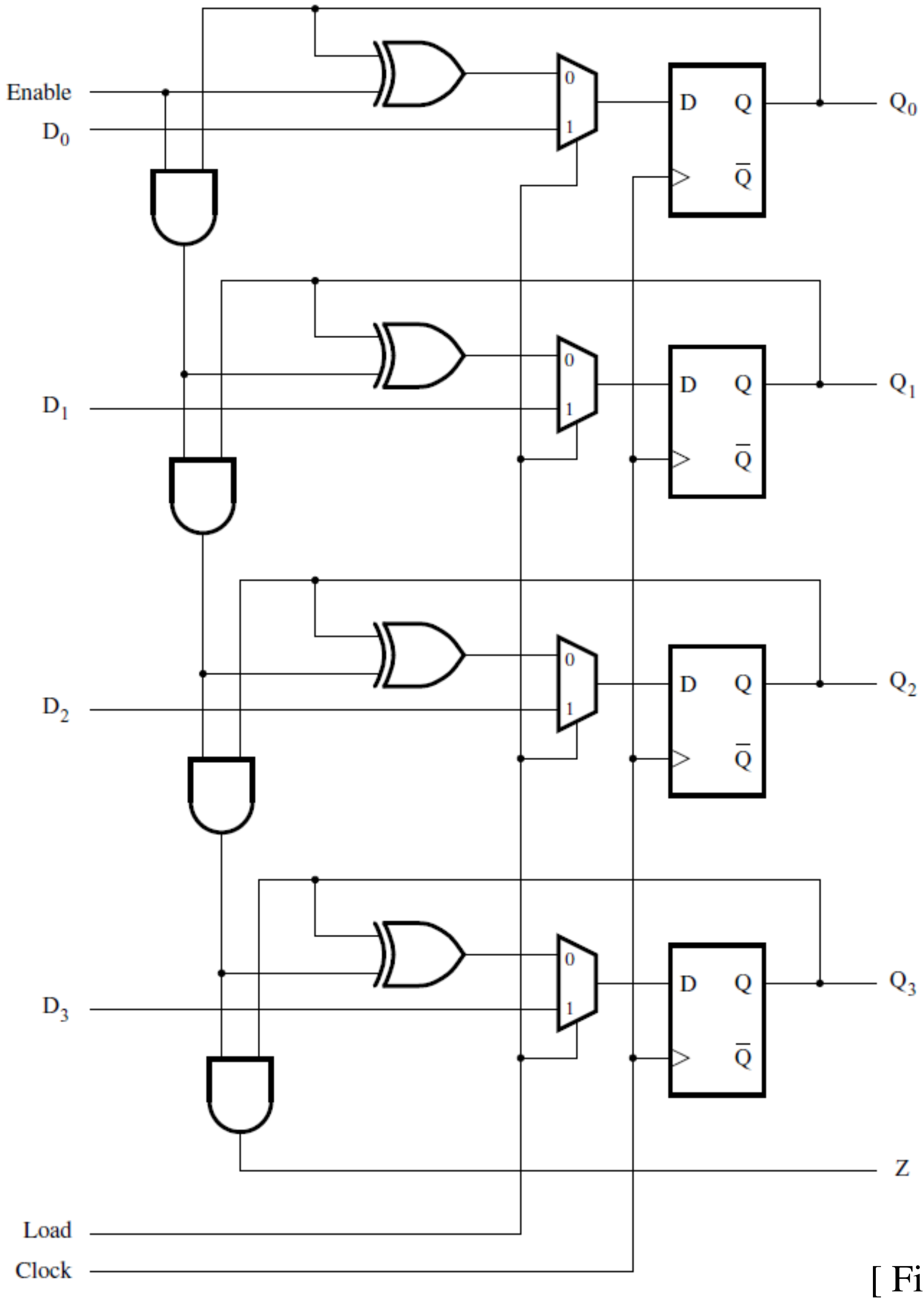
[Figure 5.27 from the textbook]

A two-digit BCD counter



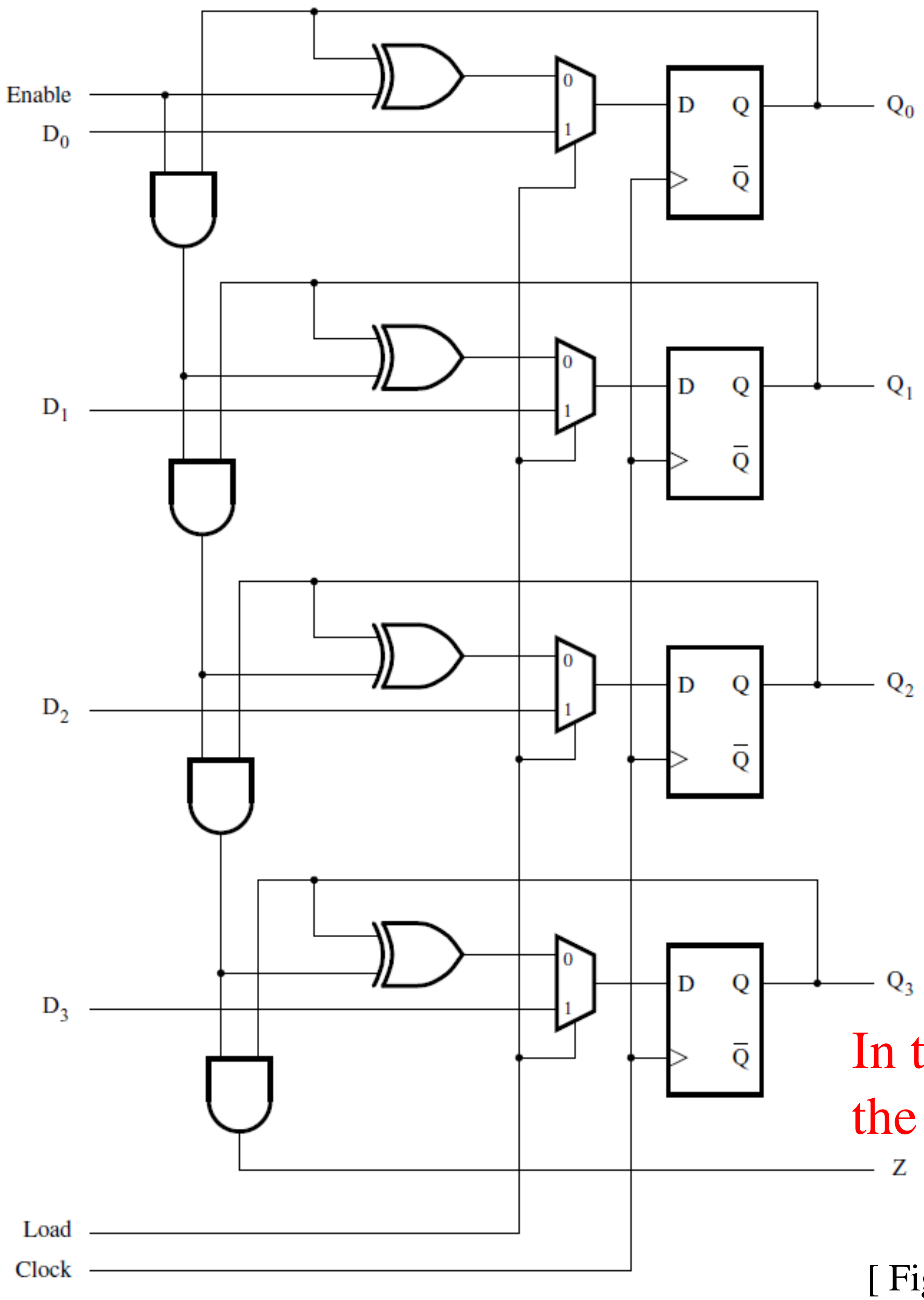
[Figure 5.27 from the textbook]

It is a counter with parallel-load capability



[Figure 5.24 from the textbook]

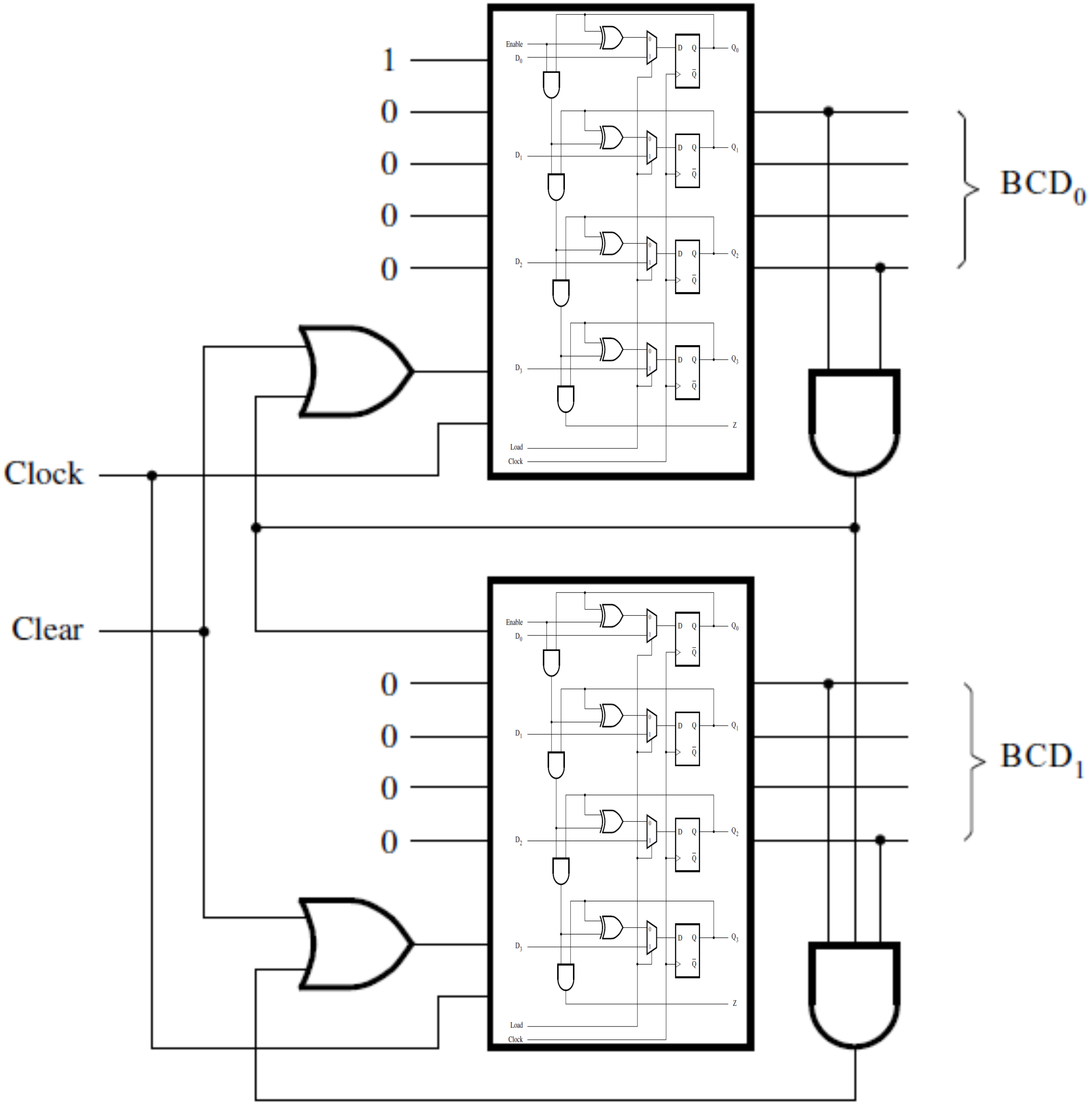
It is a counter with parallel-load capability



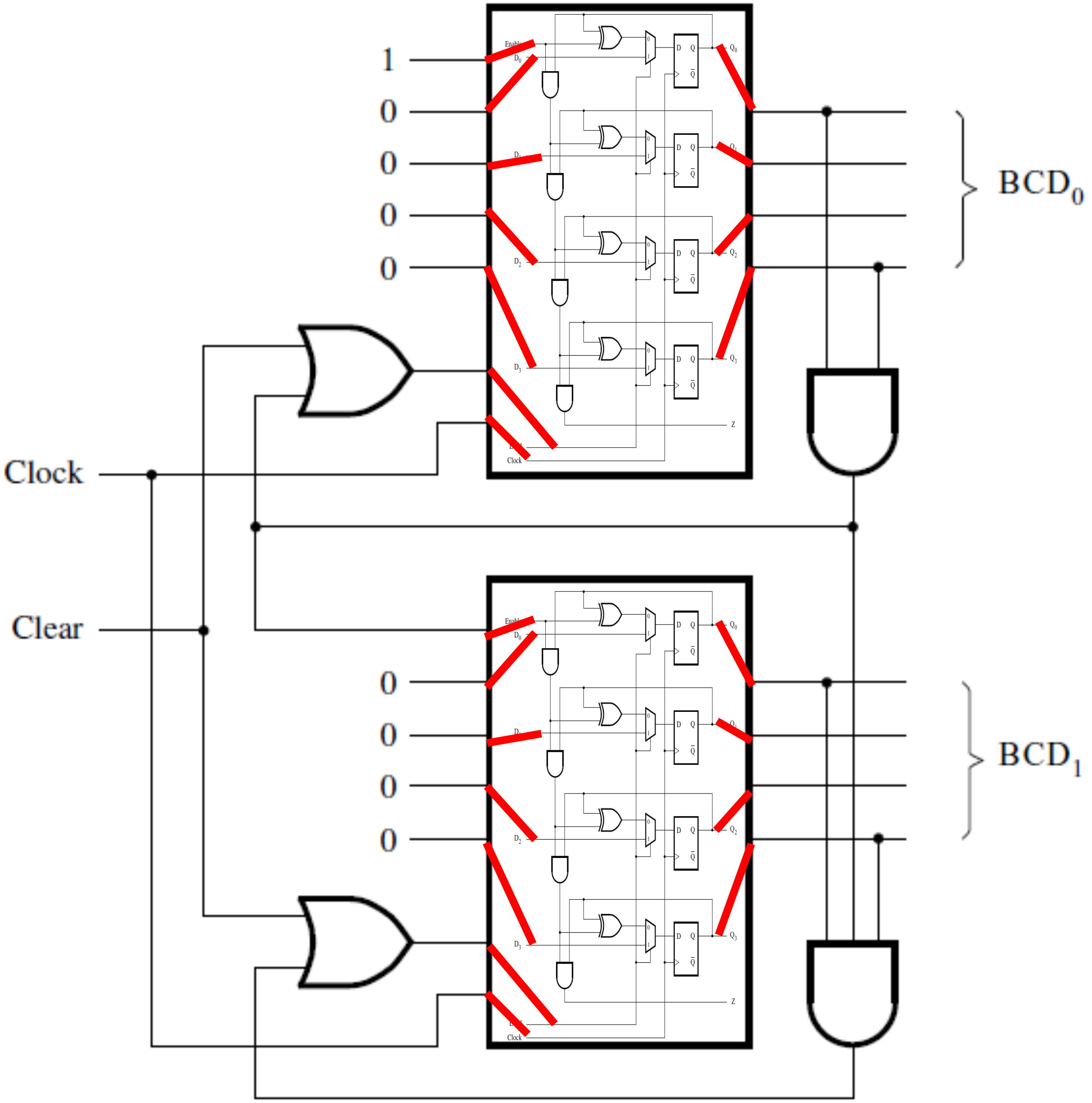
In this case,
the z output is ignored

[Figure 5.24 from the textbook]

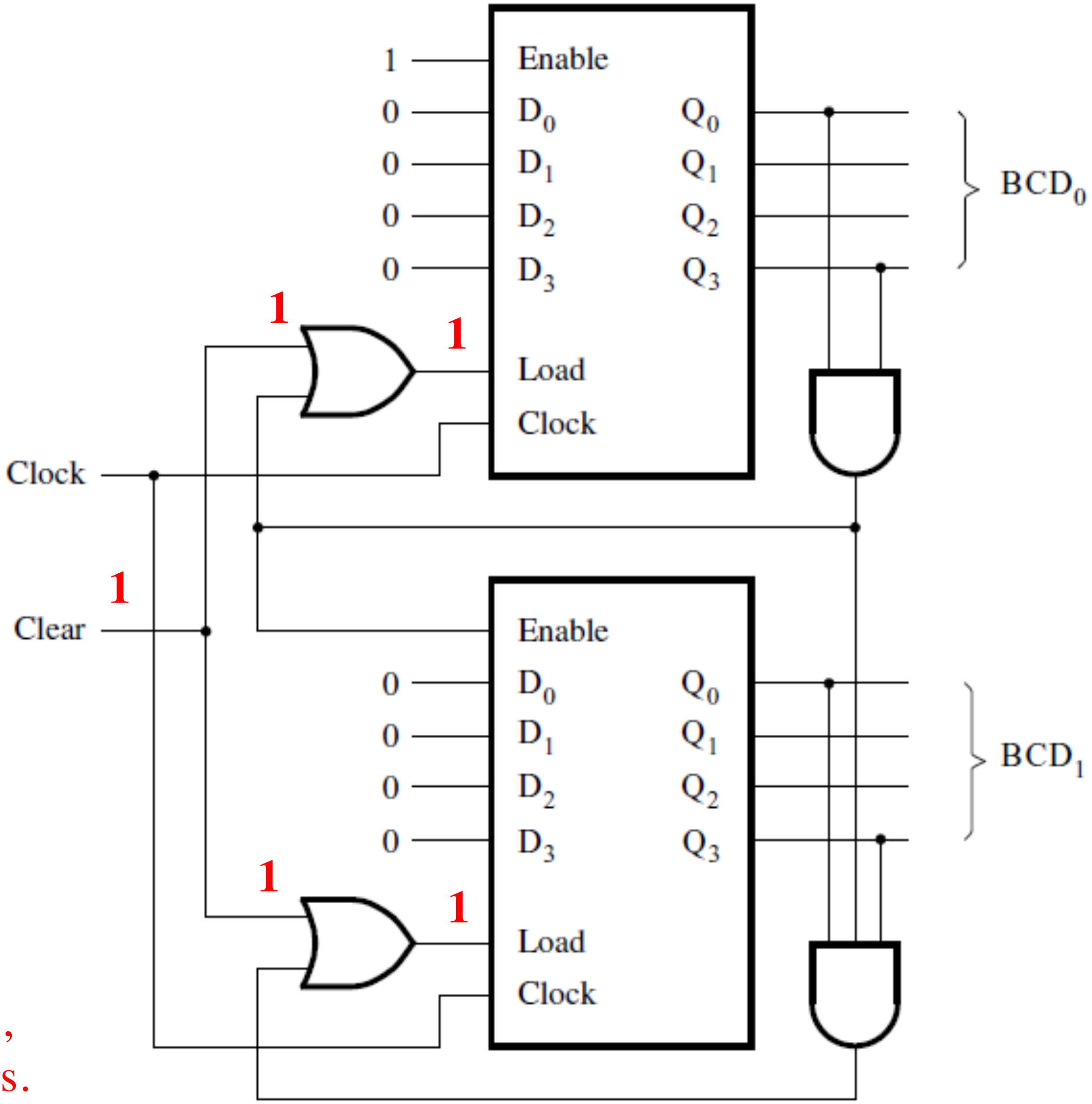
A two-digit BCD counter



A two-digit BCD counter



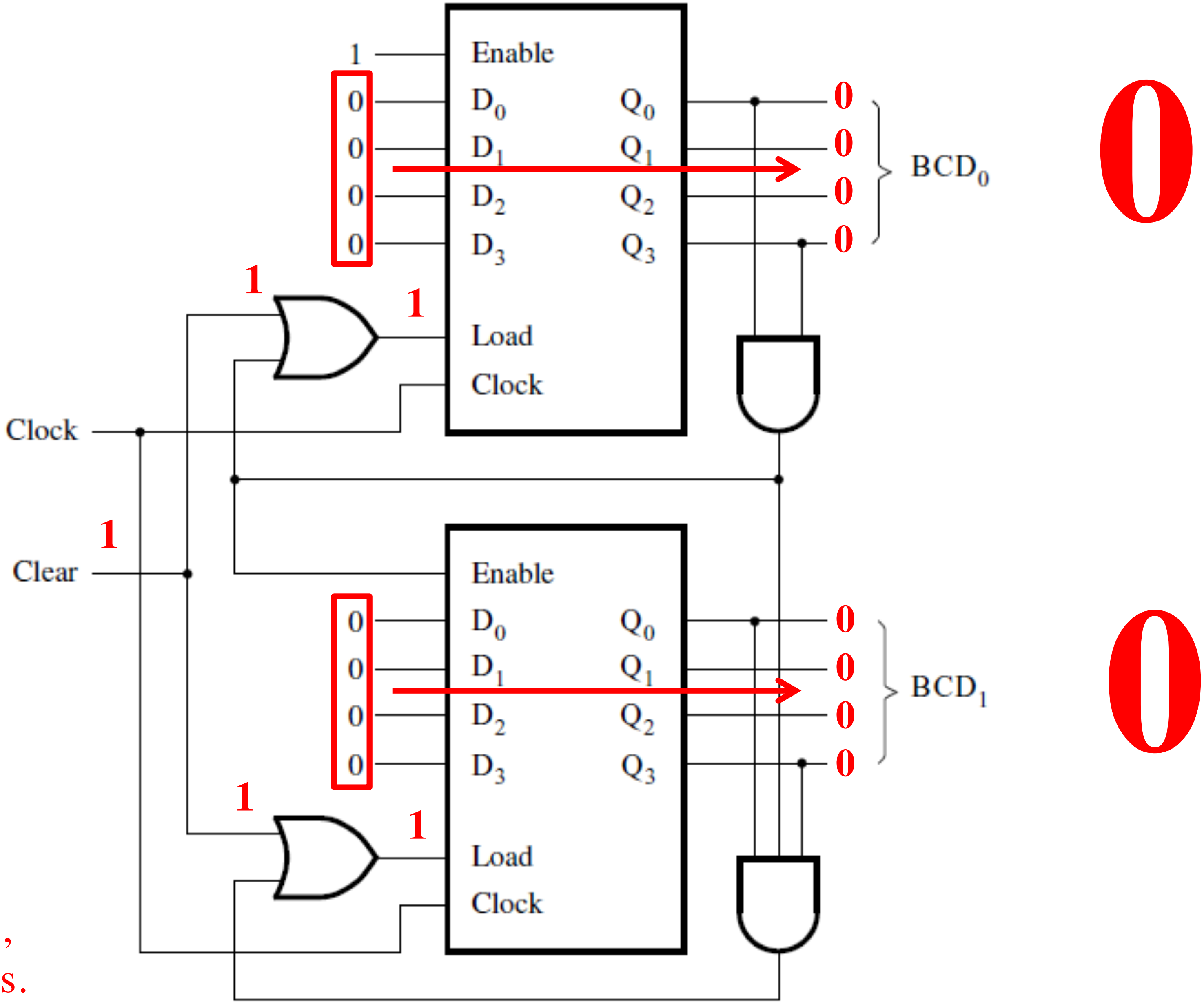
Zeroing the BCD counter



Setting "Clear" to 1,
zeroes both counters.

[Figure 5.27 from the textbook]

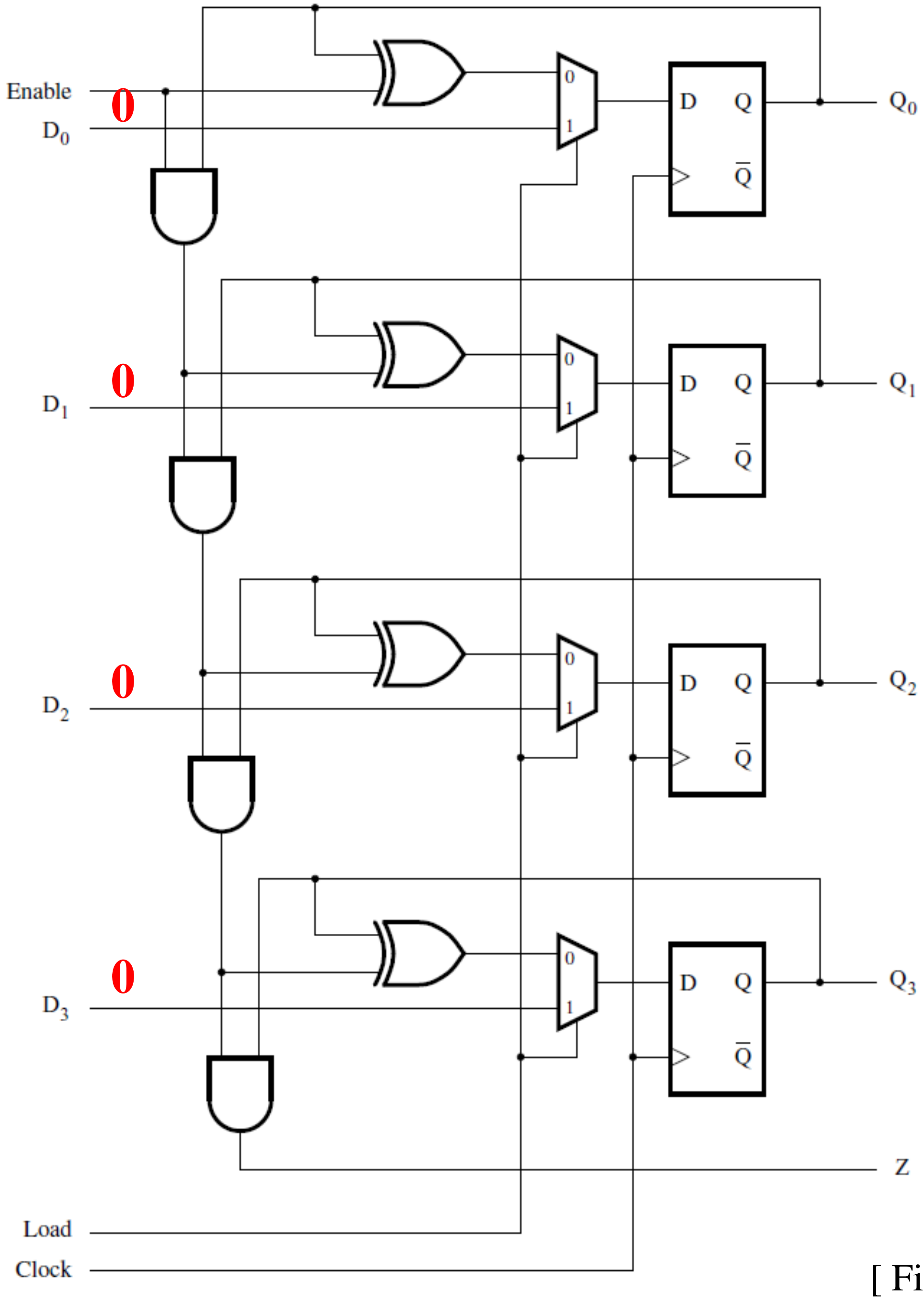
Zeroing the BCD counter



Setting "Clear" to 1,
zeroes both counters.

[Figure 5.27 from the textbook]

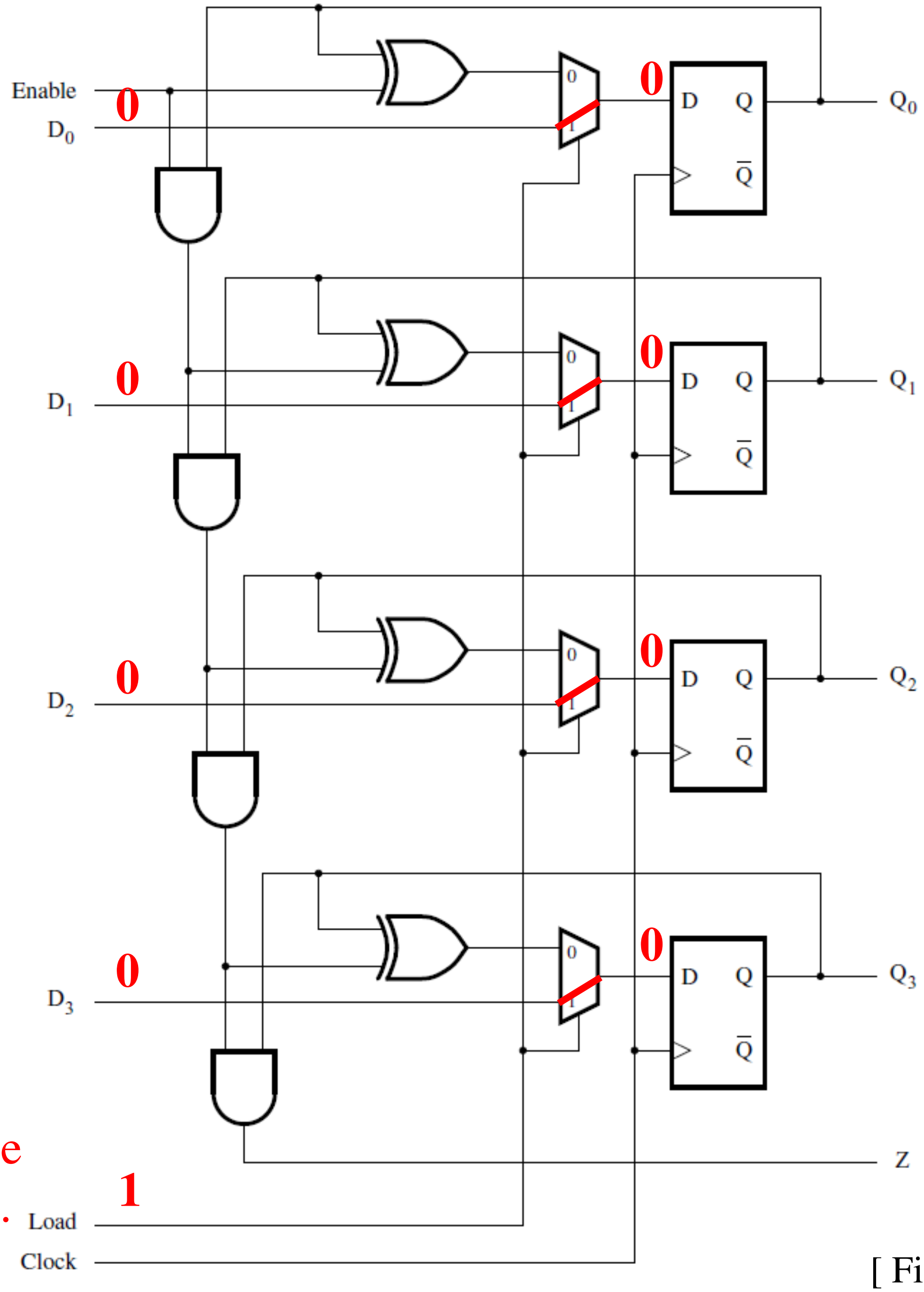
How to zero a counter



Set all parallel load input lines to zero.

[Figure 5.24 from the textbook]

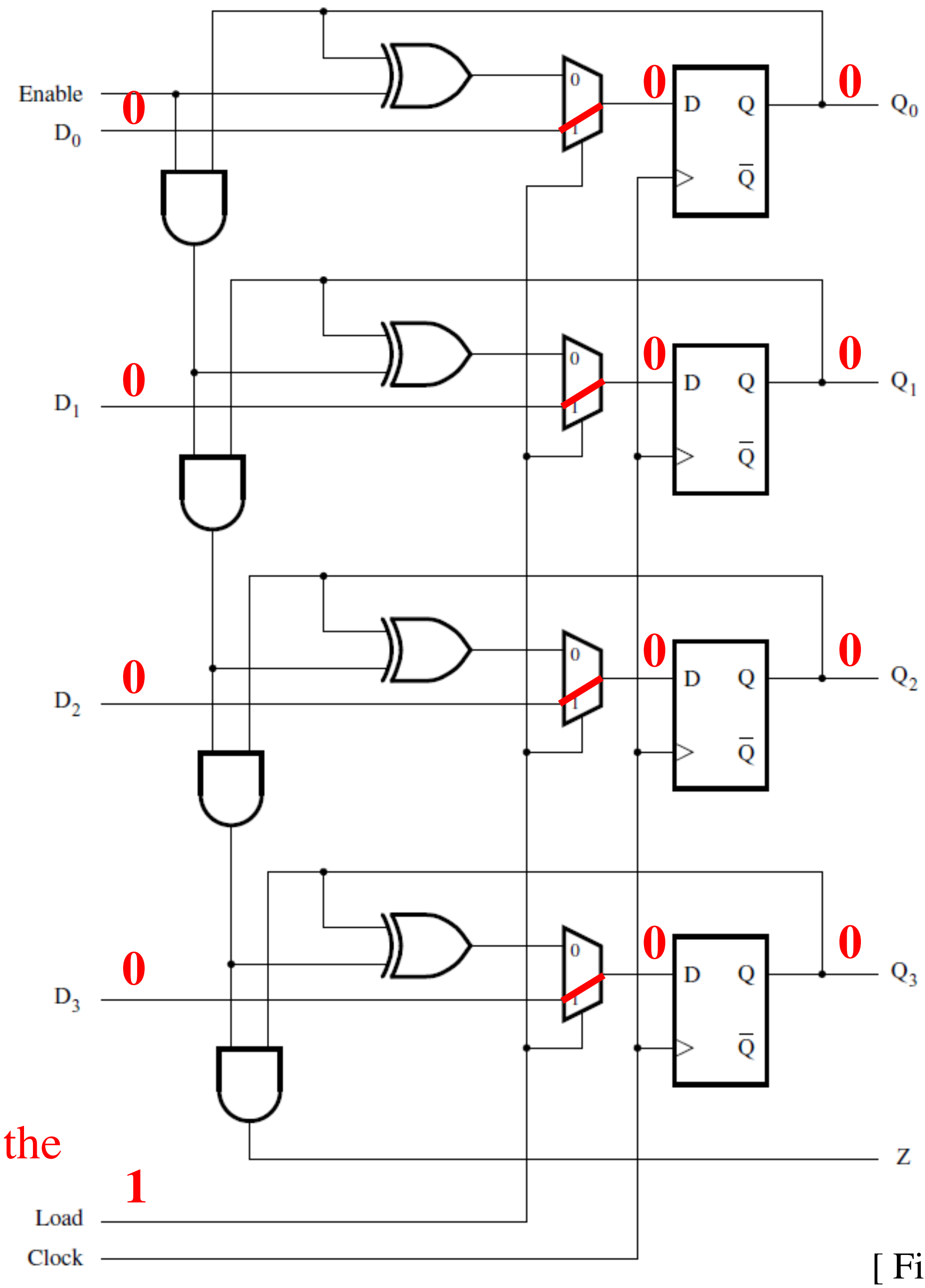
How to zero a counter



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

[Figure 5.24 from the textbook]

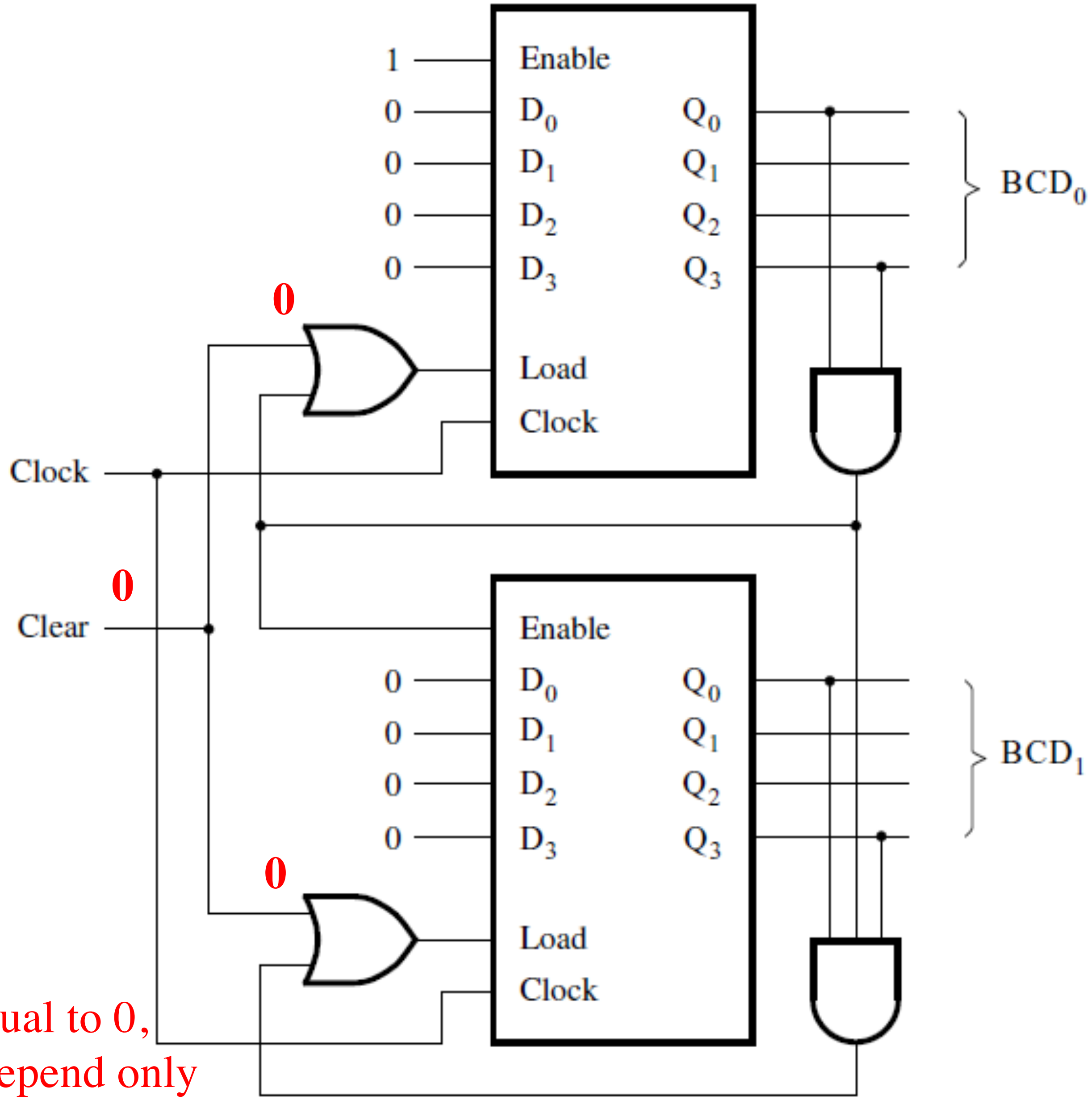
How to zero a counter



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

[Figure 5.24 from the textbook]

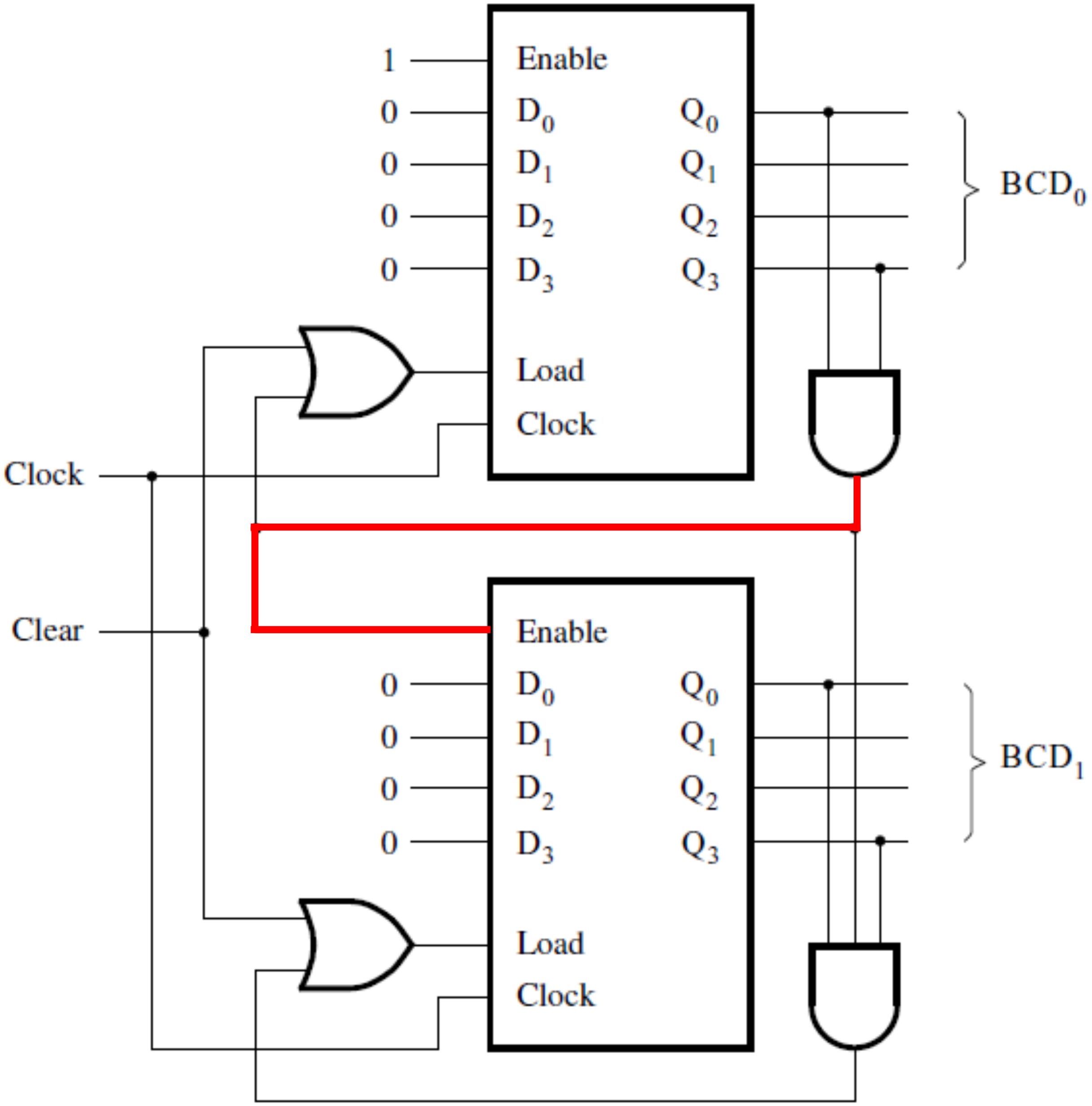
When Clear = 0



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

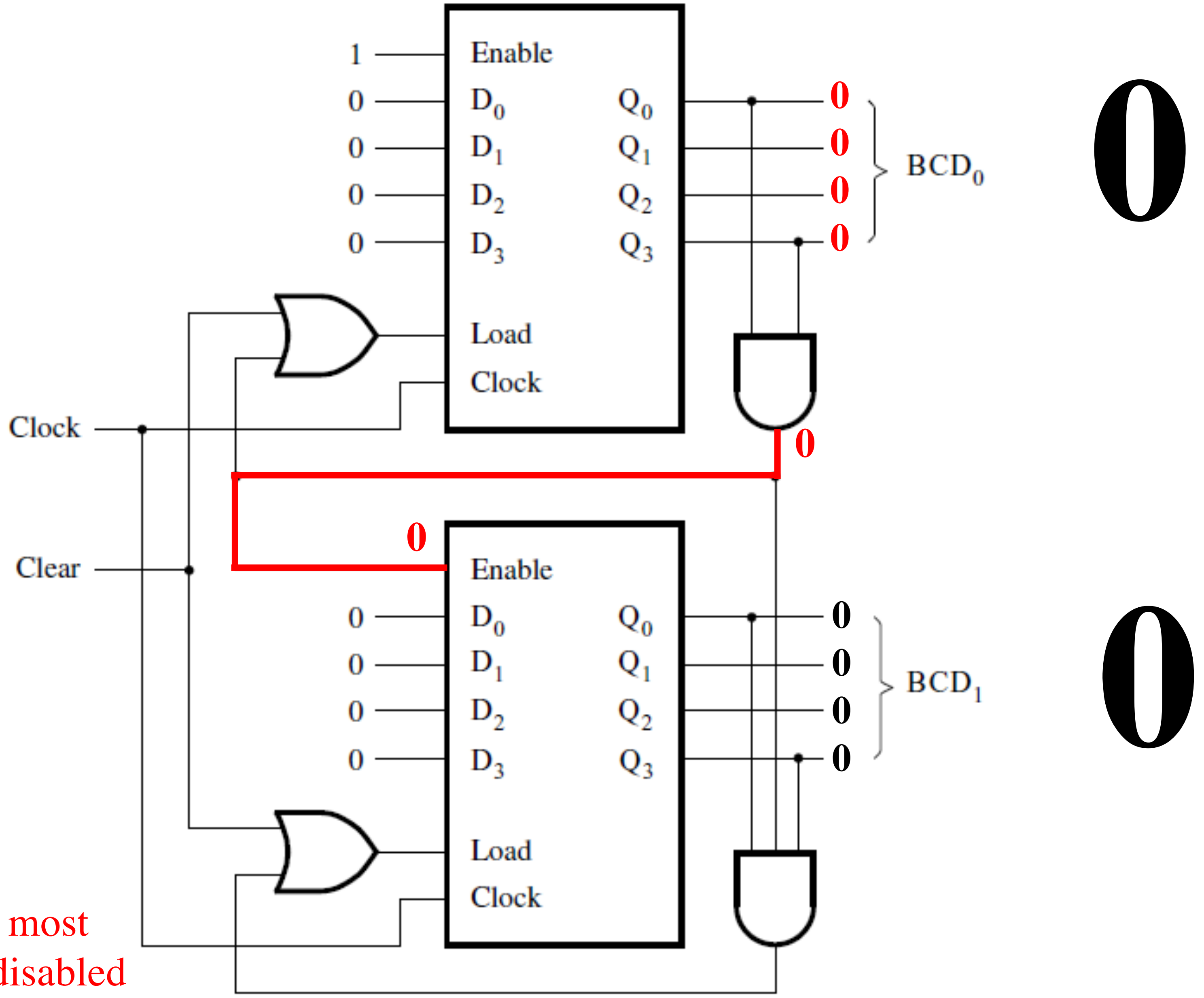
[Figure 5.27 from the textbook]

Enabling the second counter



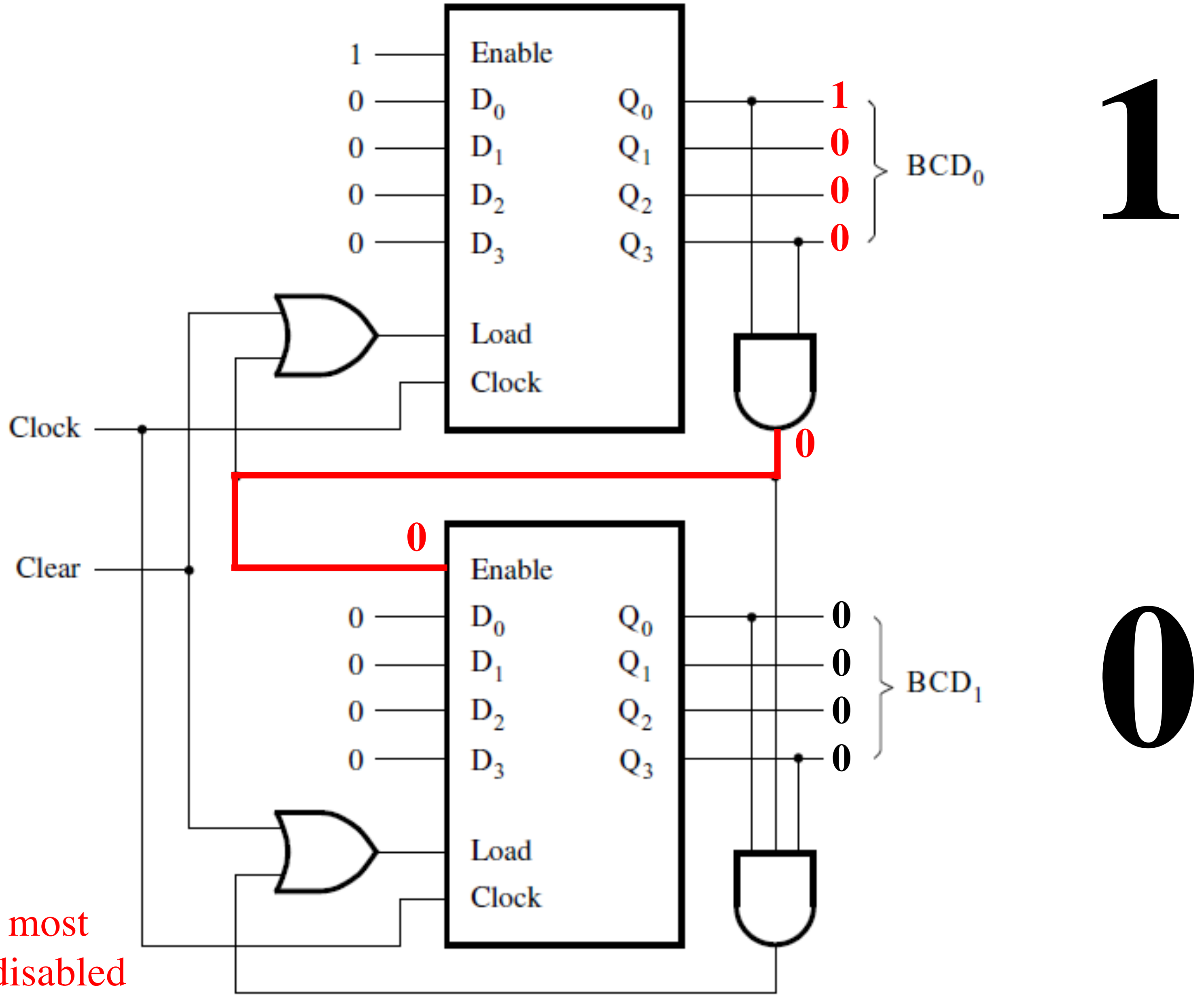
[Figure 5.27 from the textbook]

Enabling the second counter



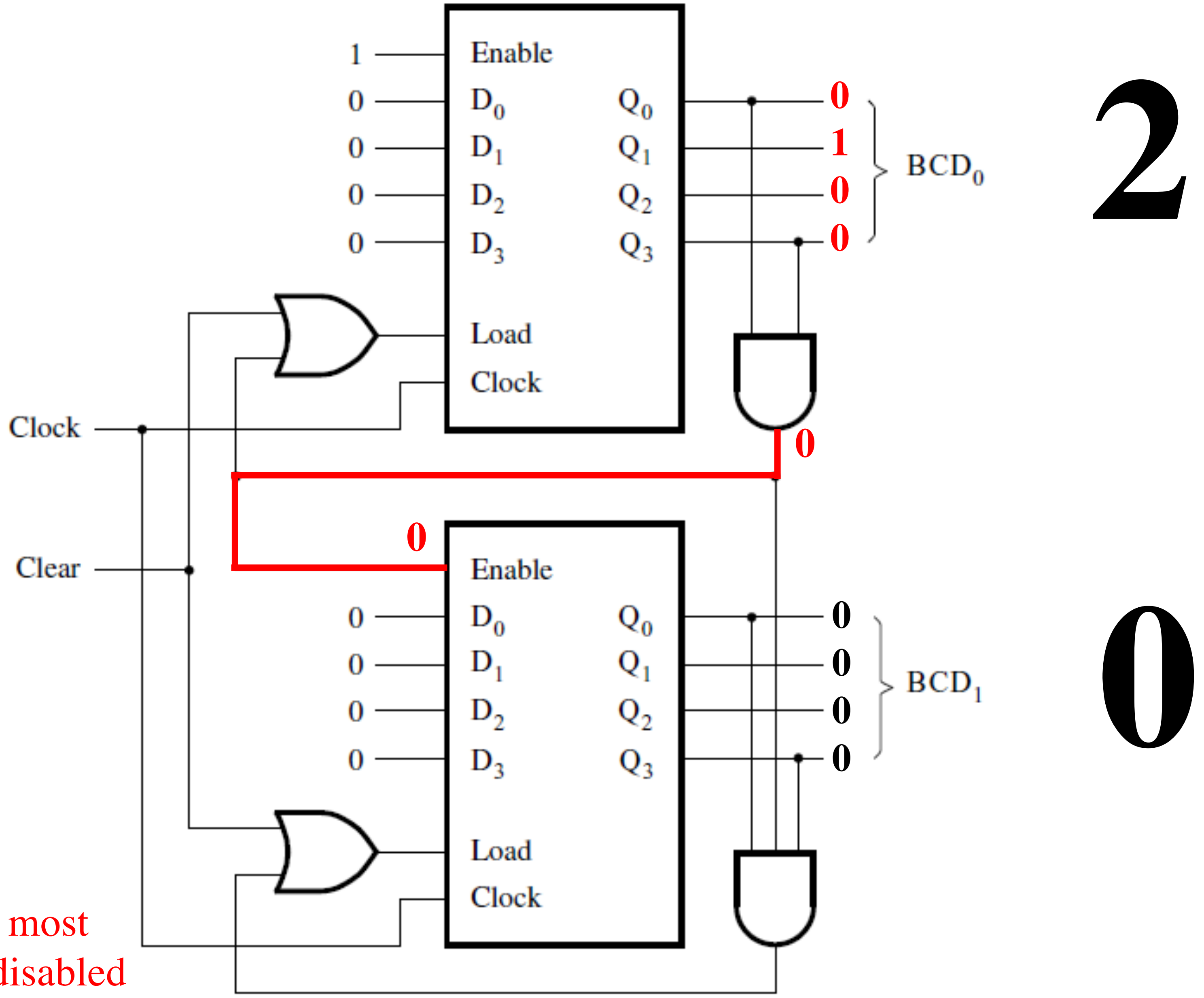
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



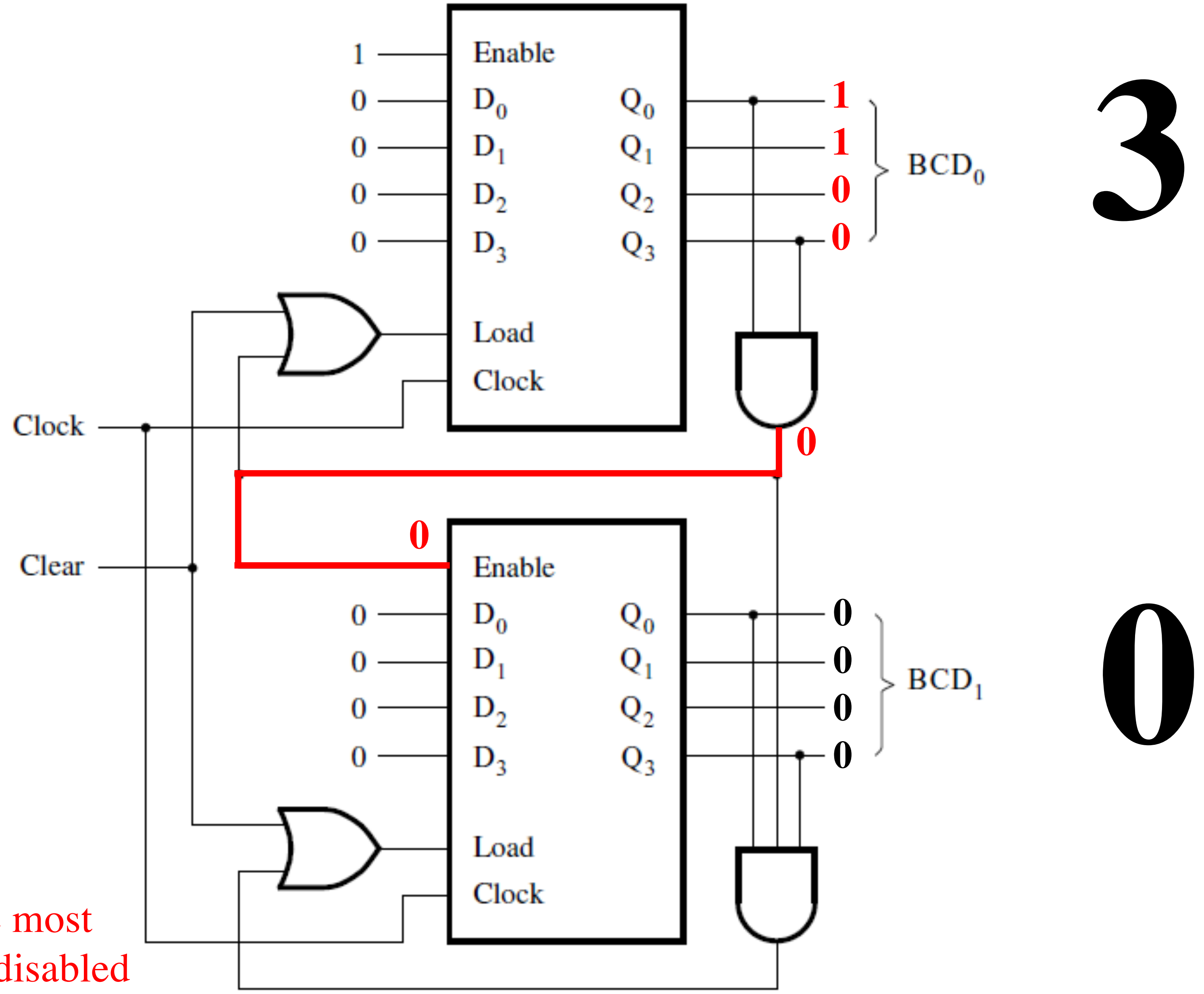
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



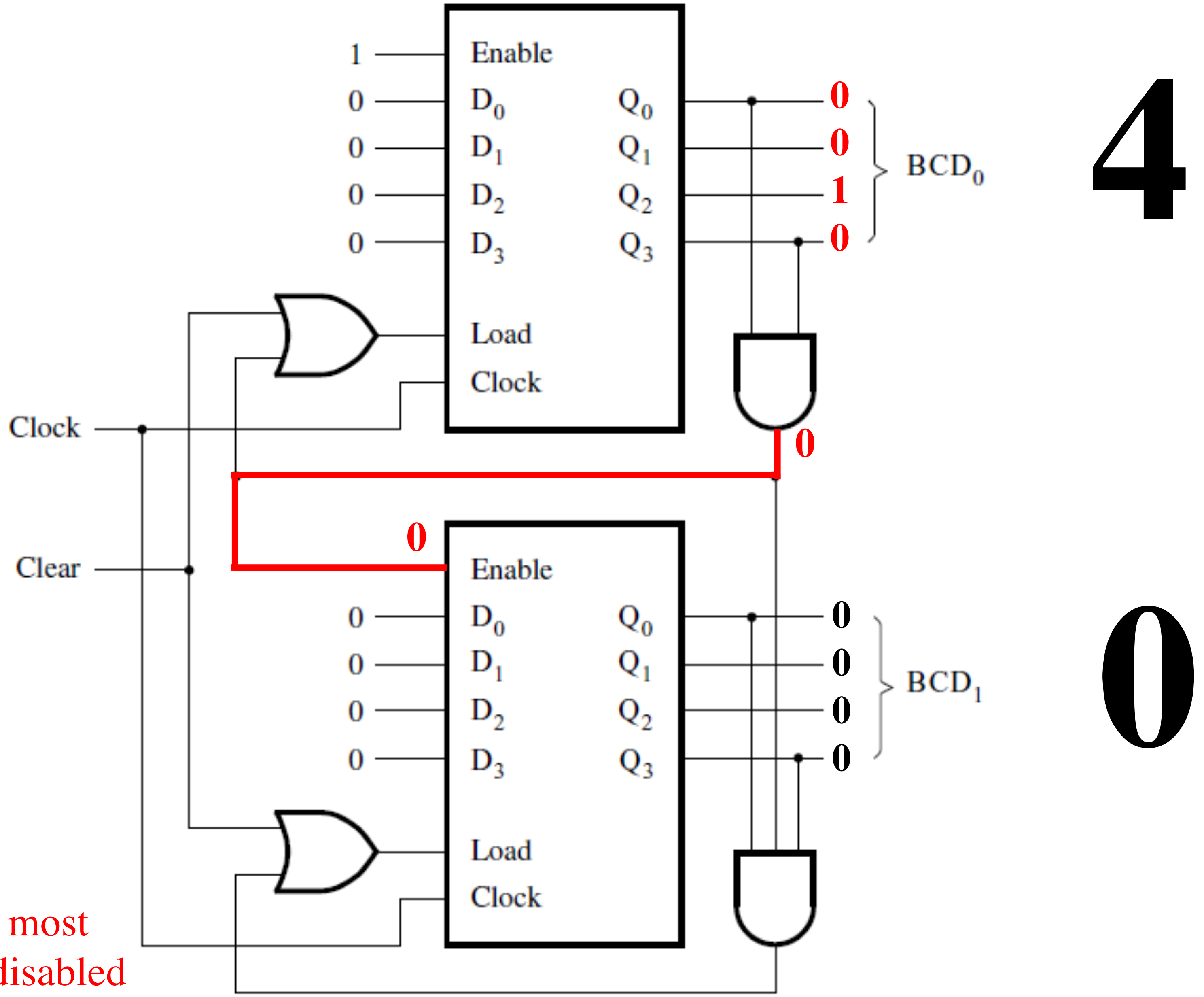
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



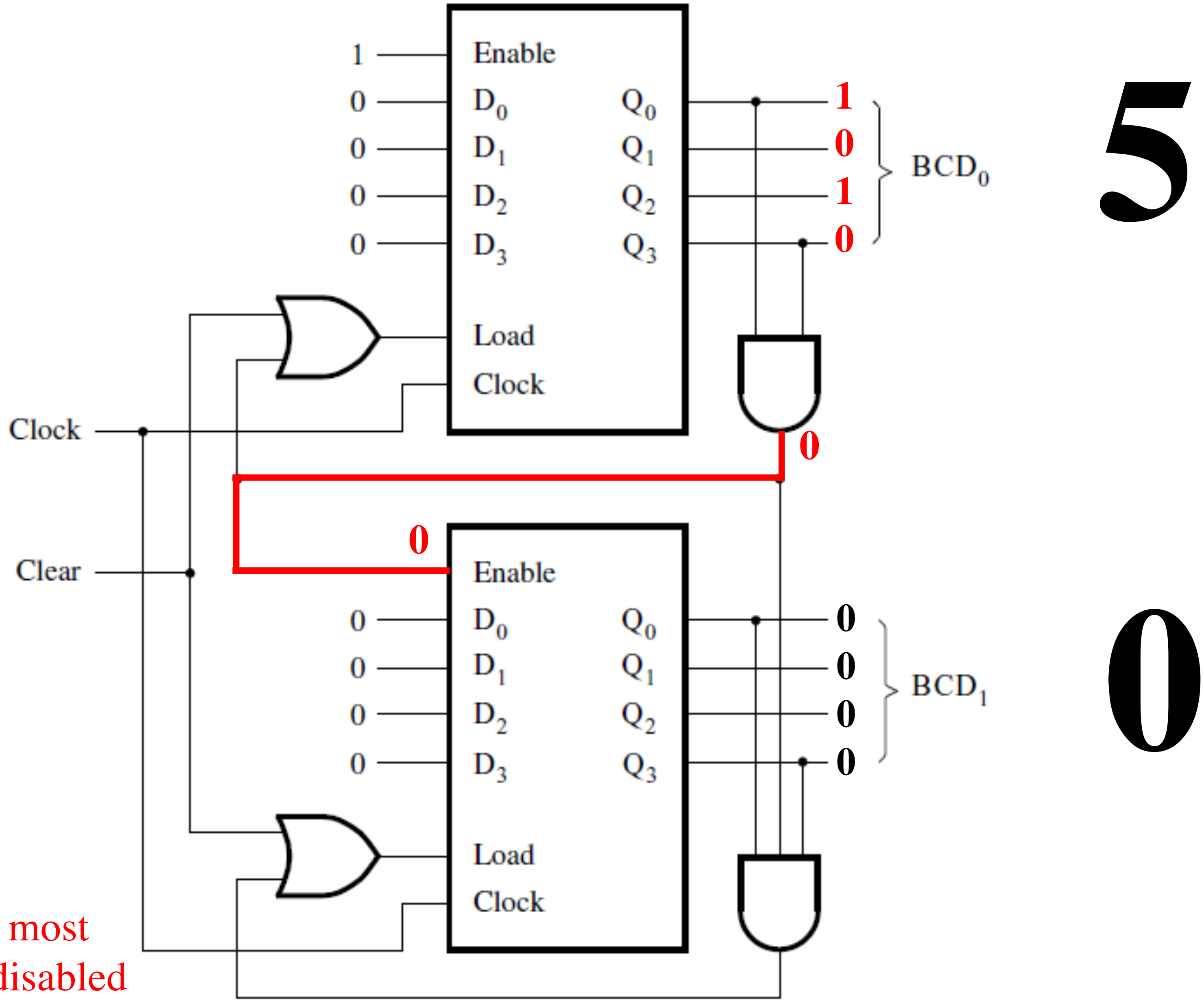
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



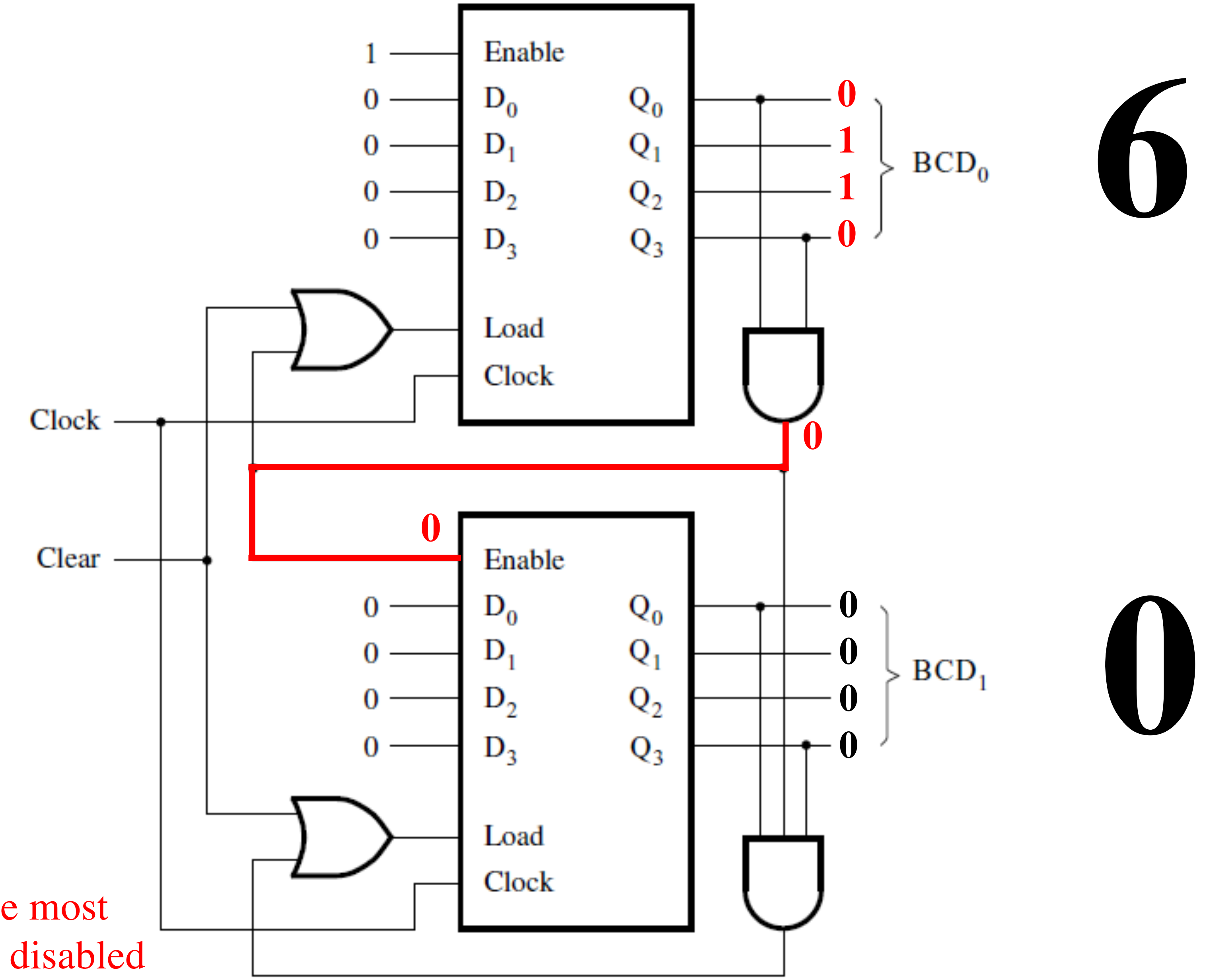
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



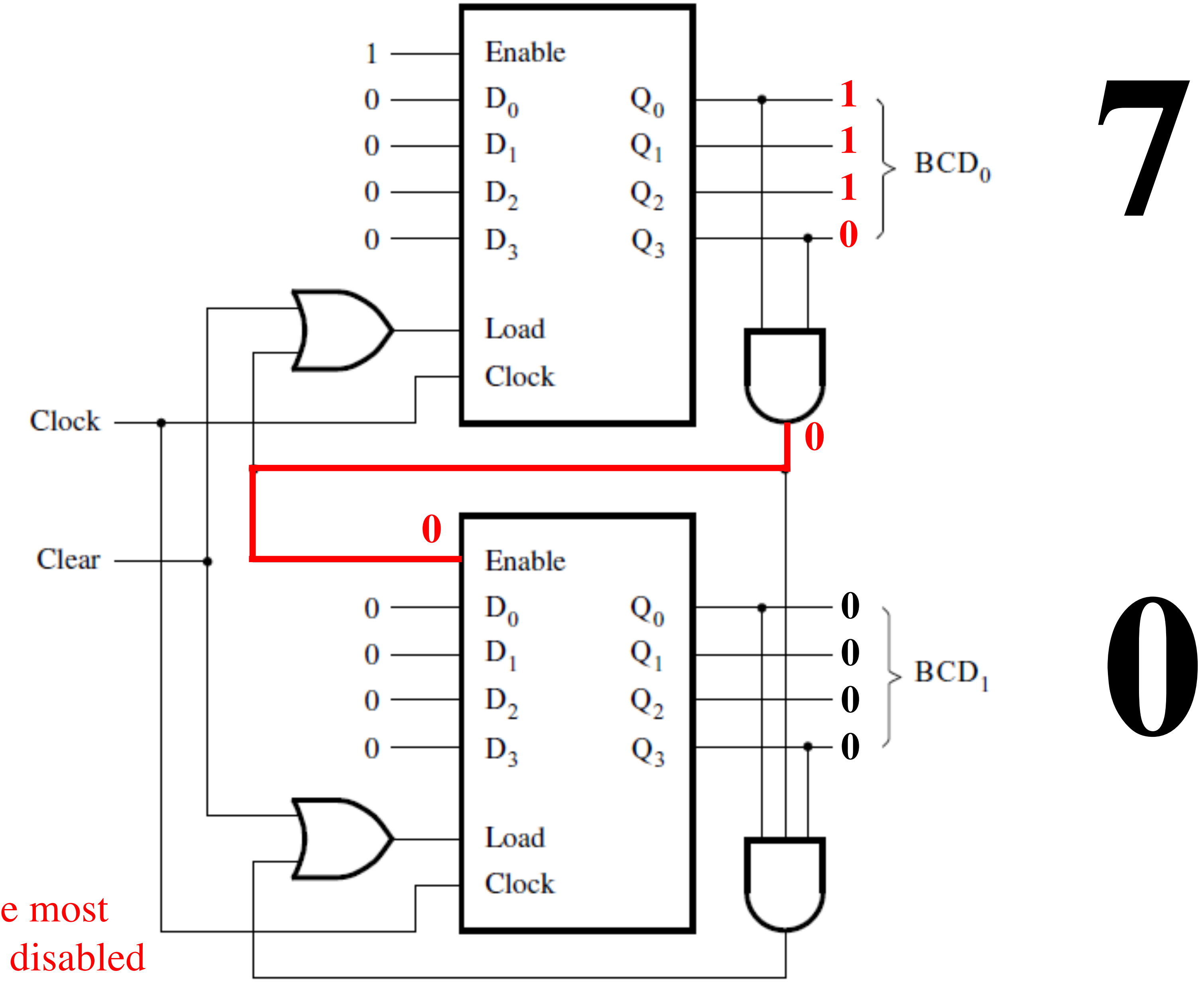
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



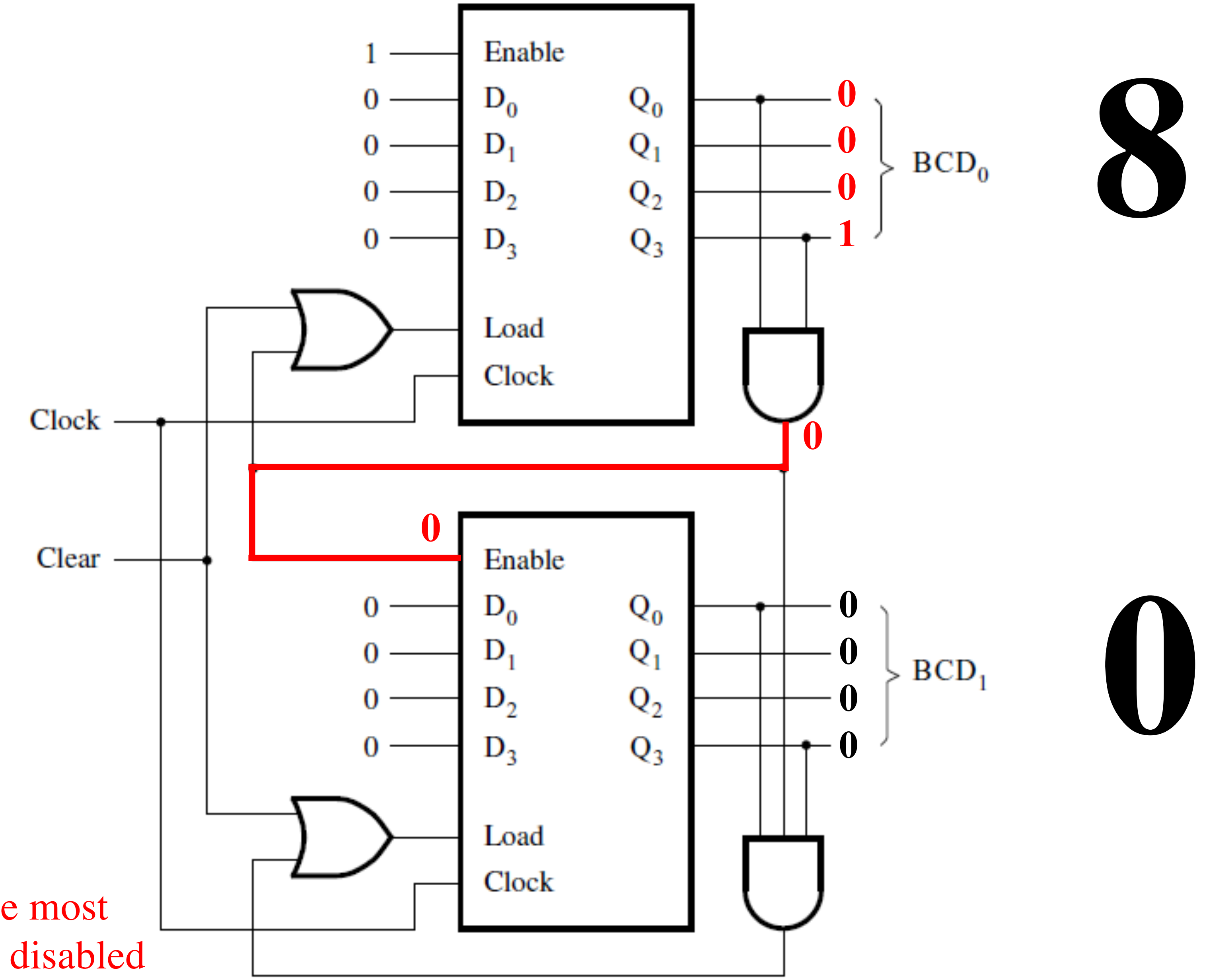
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



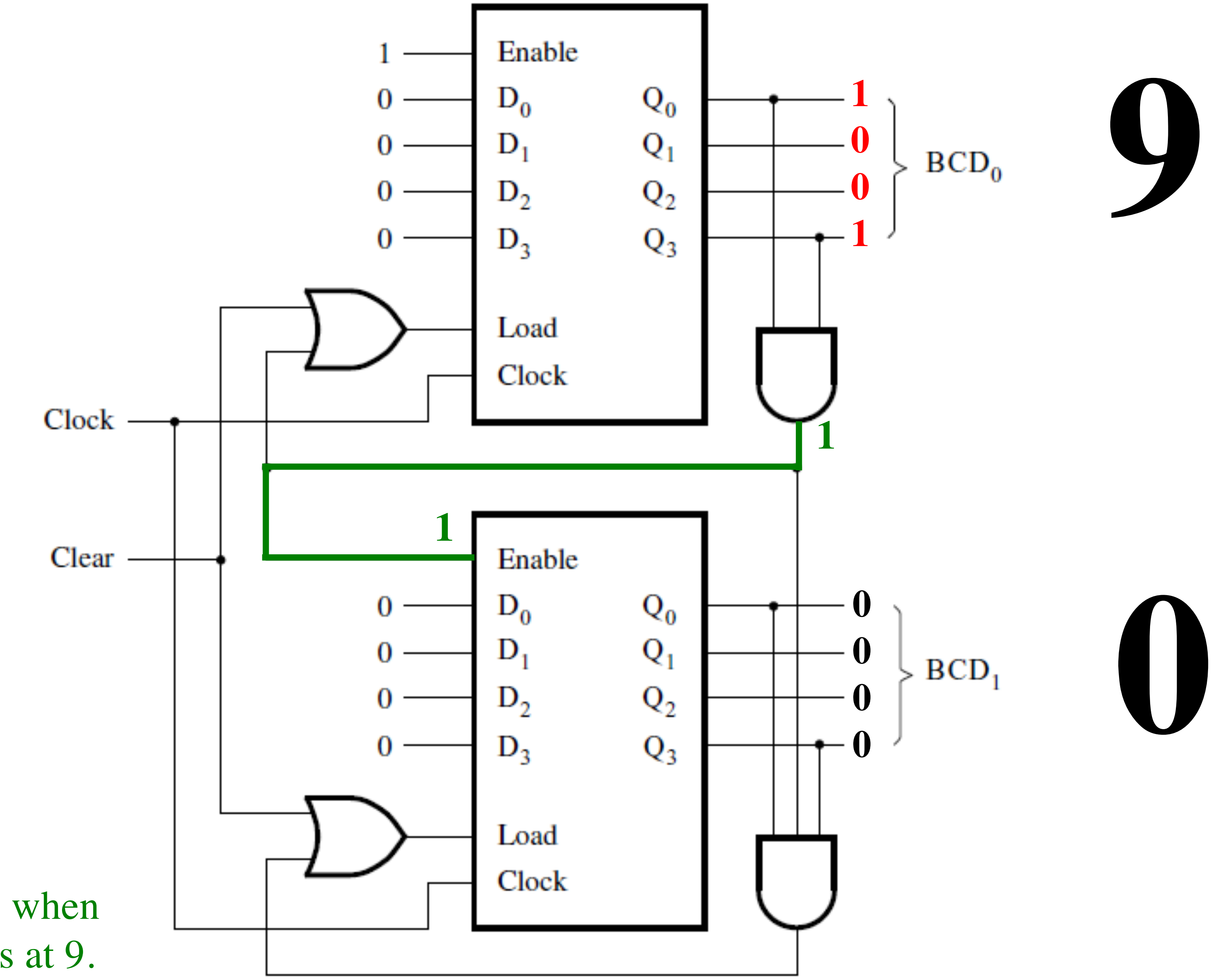
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



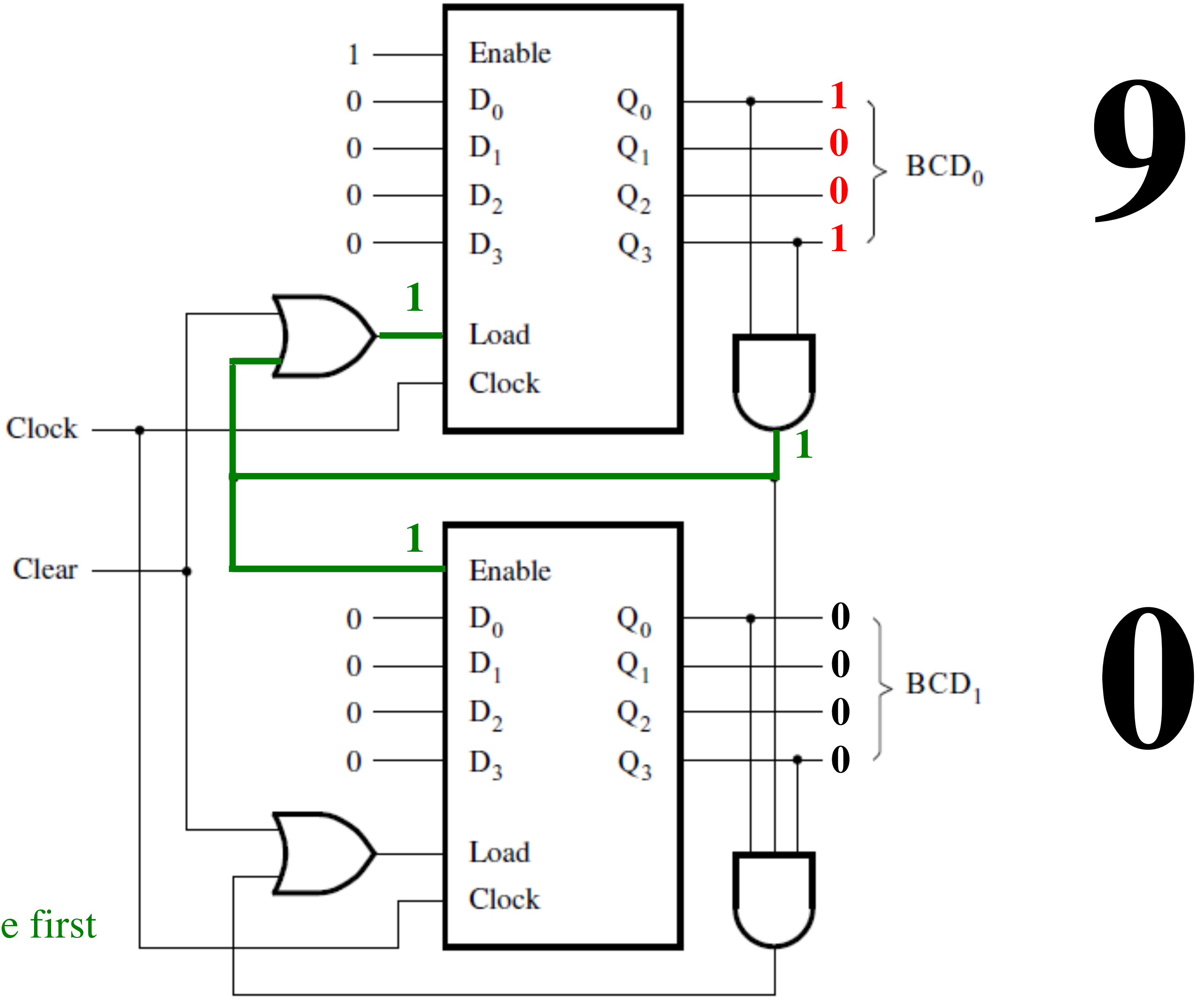
The counter for the most significant digit is disabled most of the time.

Enabling the second counter



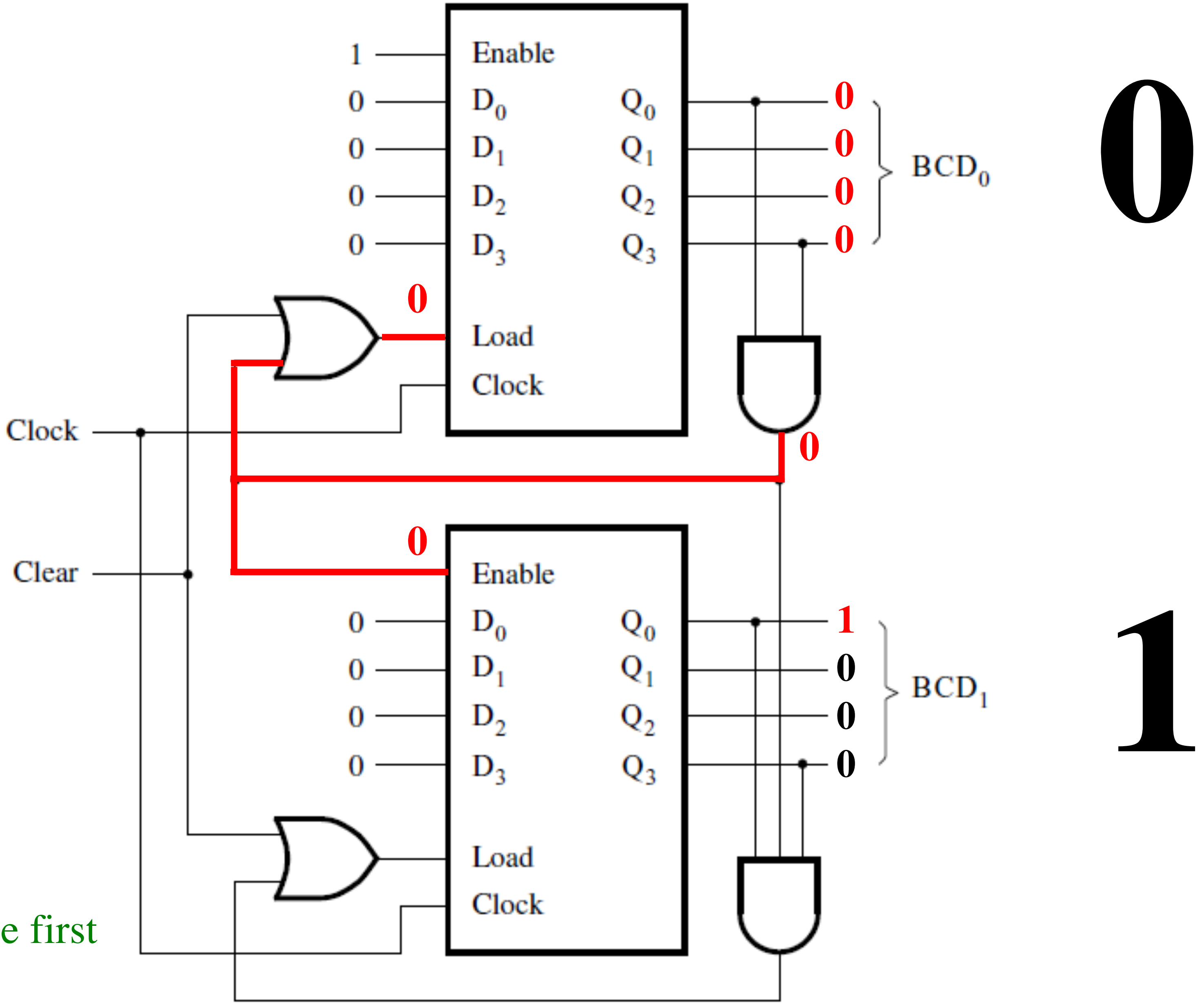
It is enabled only when the first counter is at 9.

Enabling the second counter



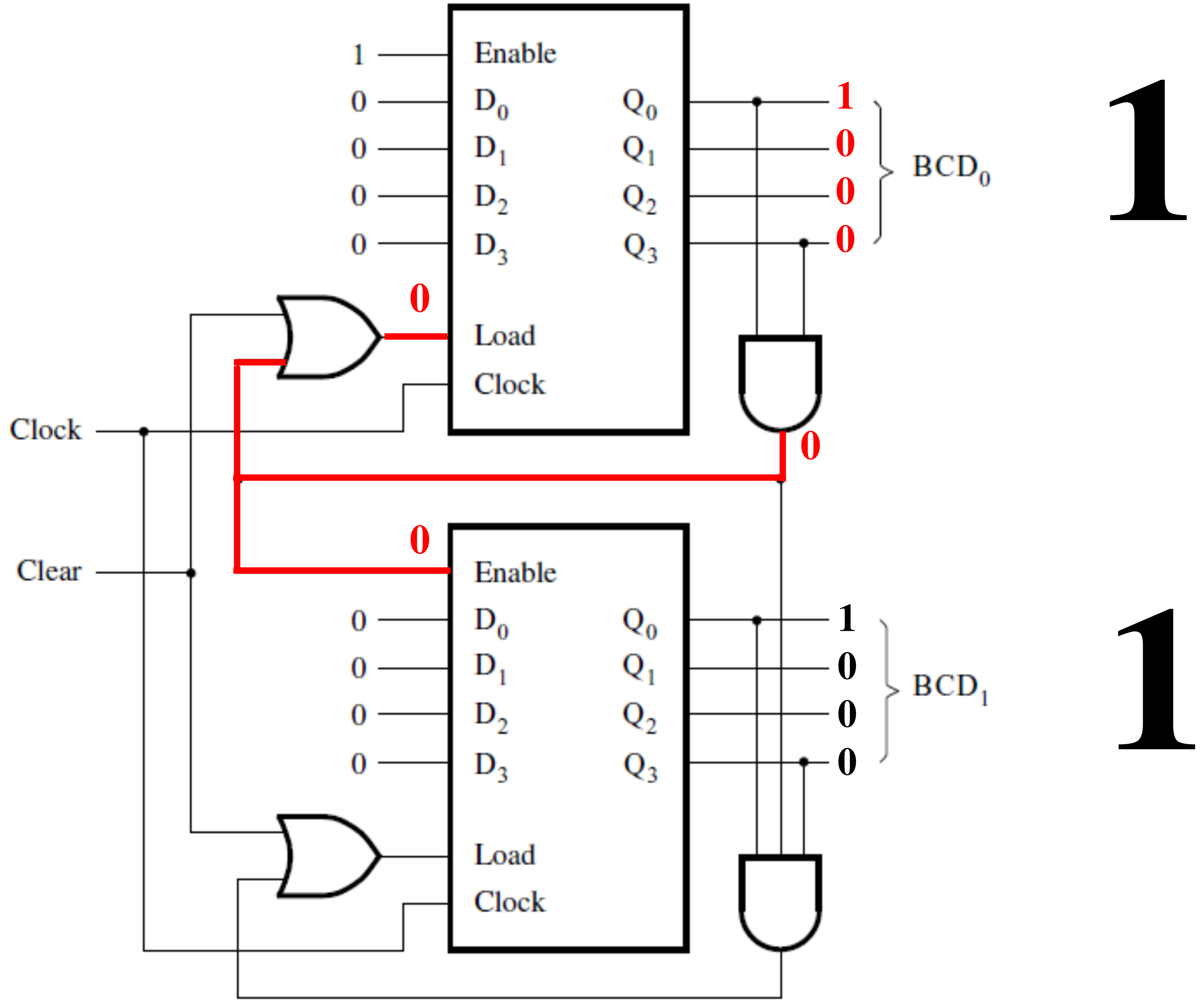
At the same time the first counter is reset.

Enabling the second counter

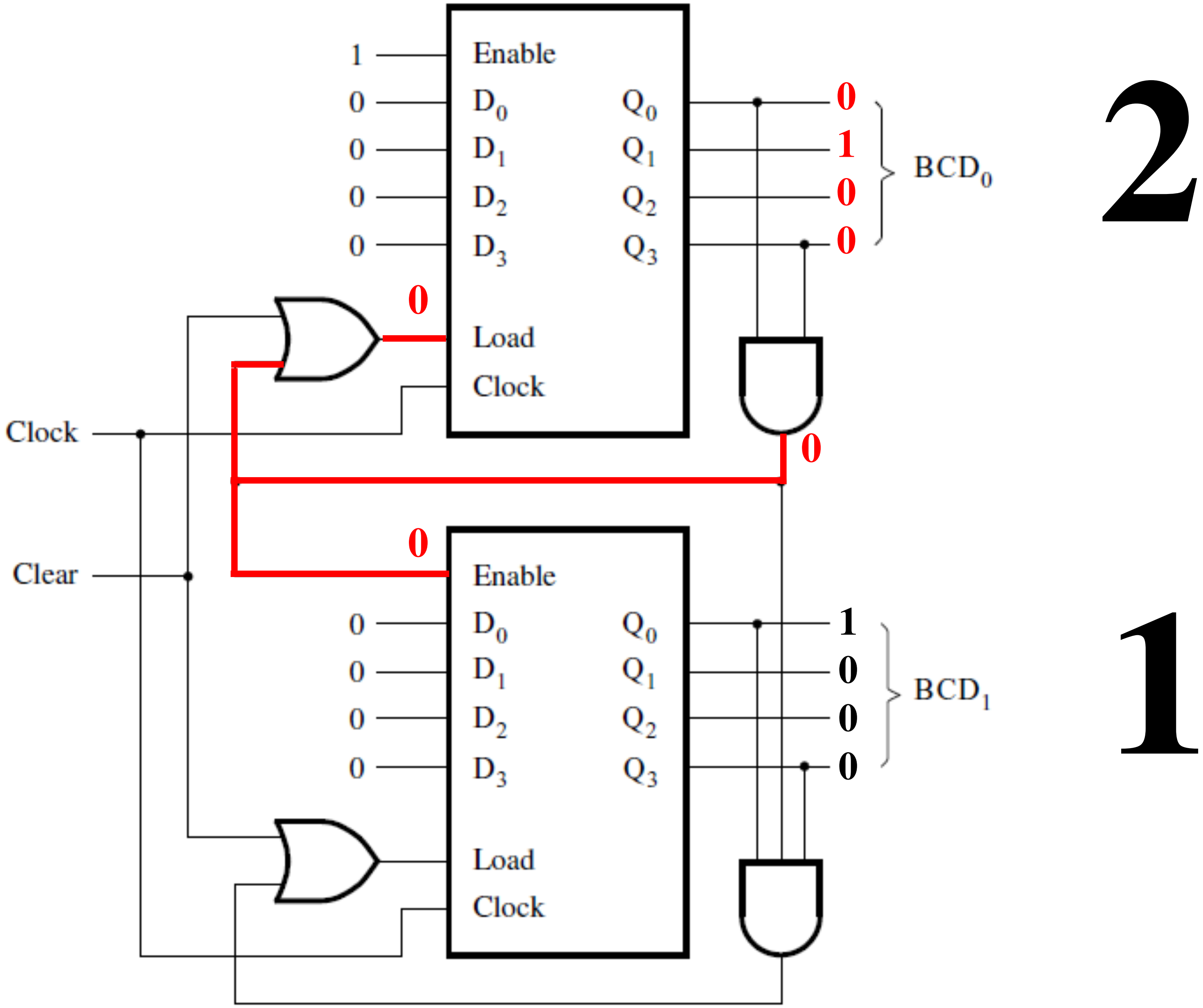


At the same time the first counter is reset.

Enabling the second counter

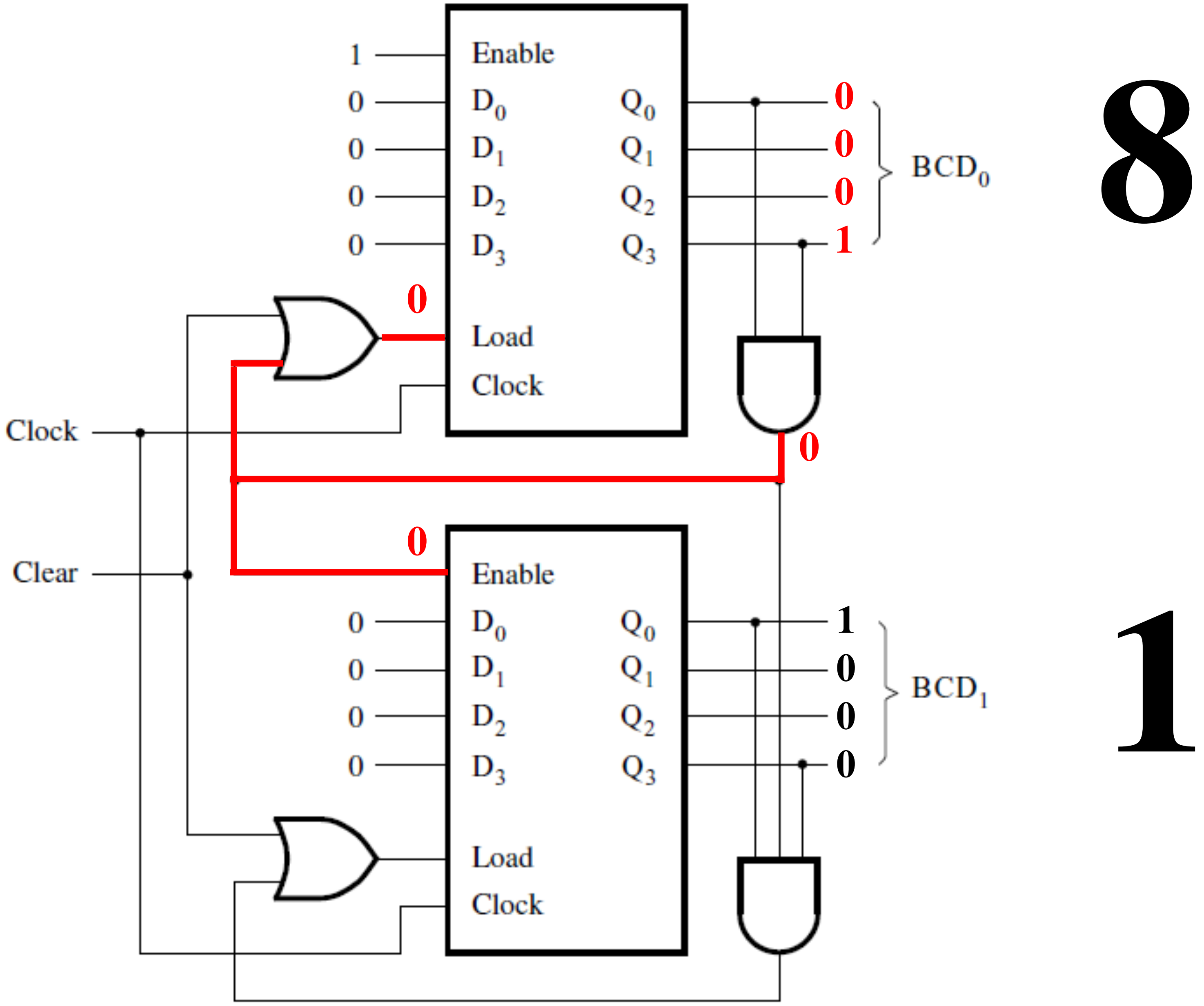


Enabling the second counter

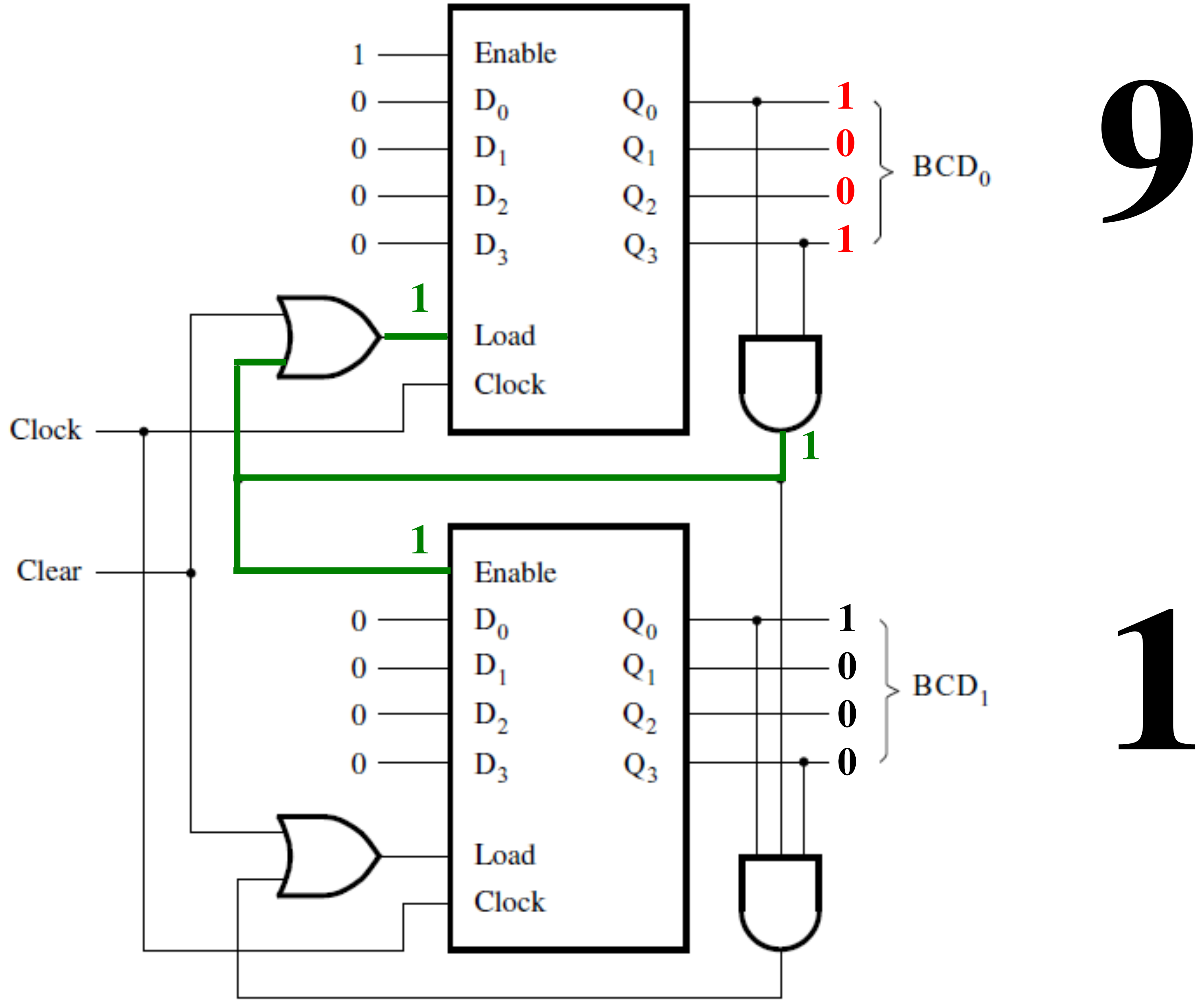


• • •

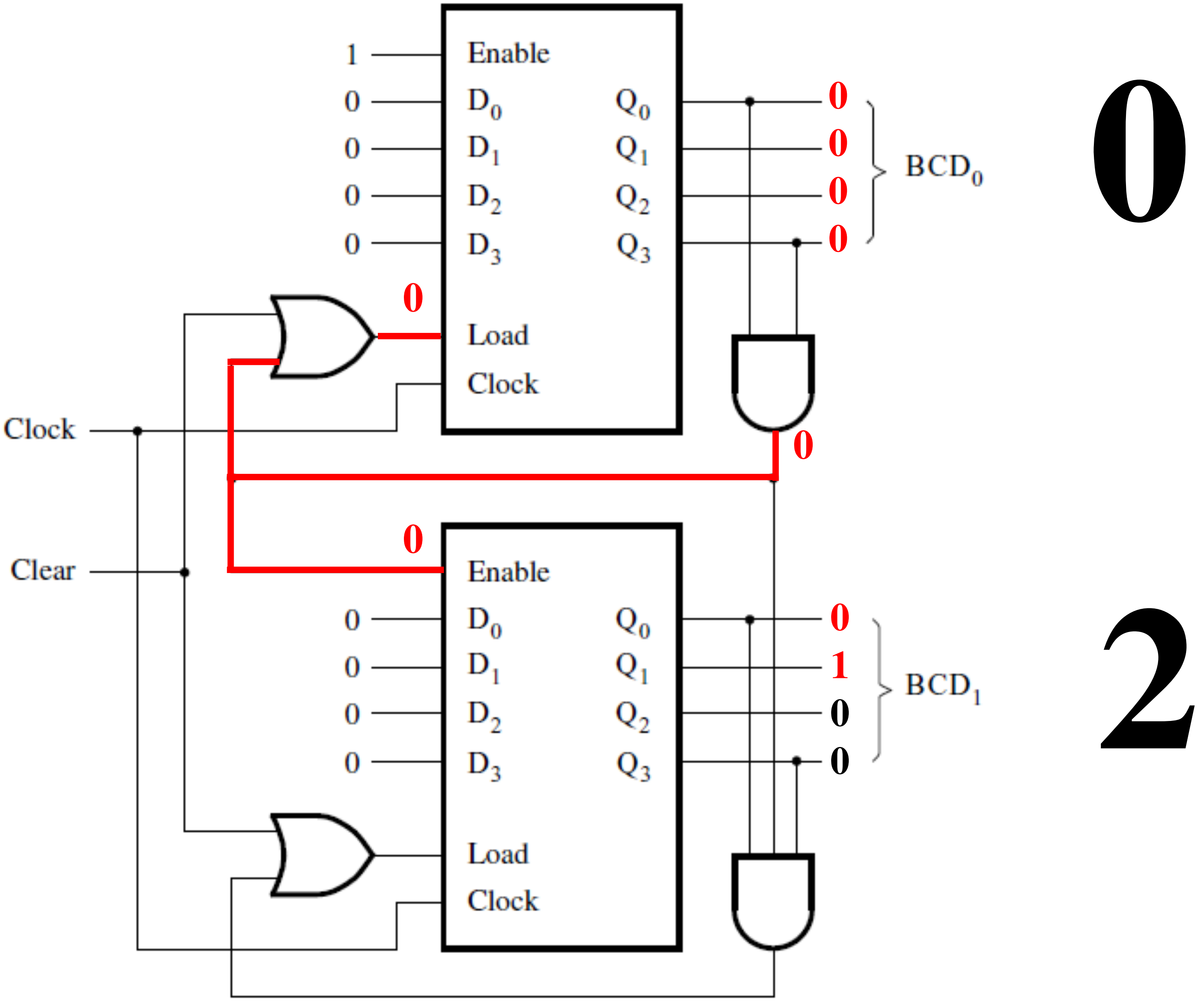
Enabling the second counter



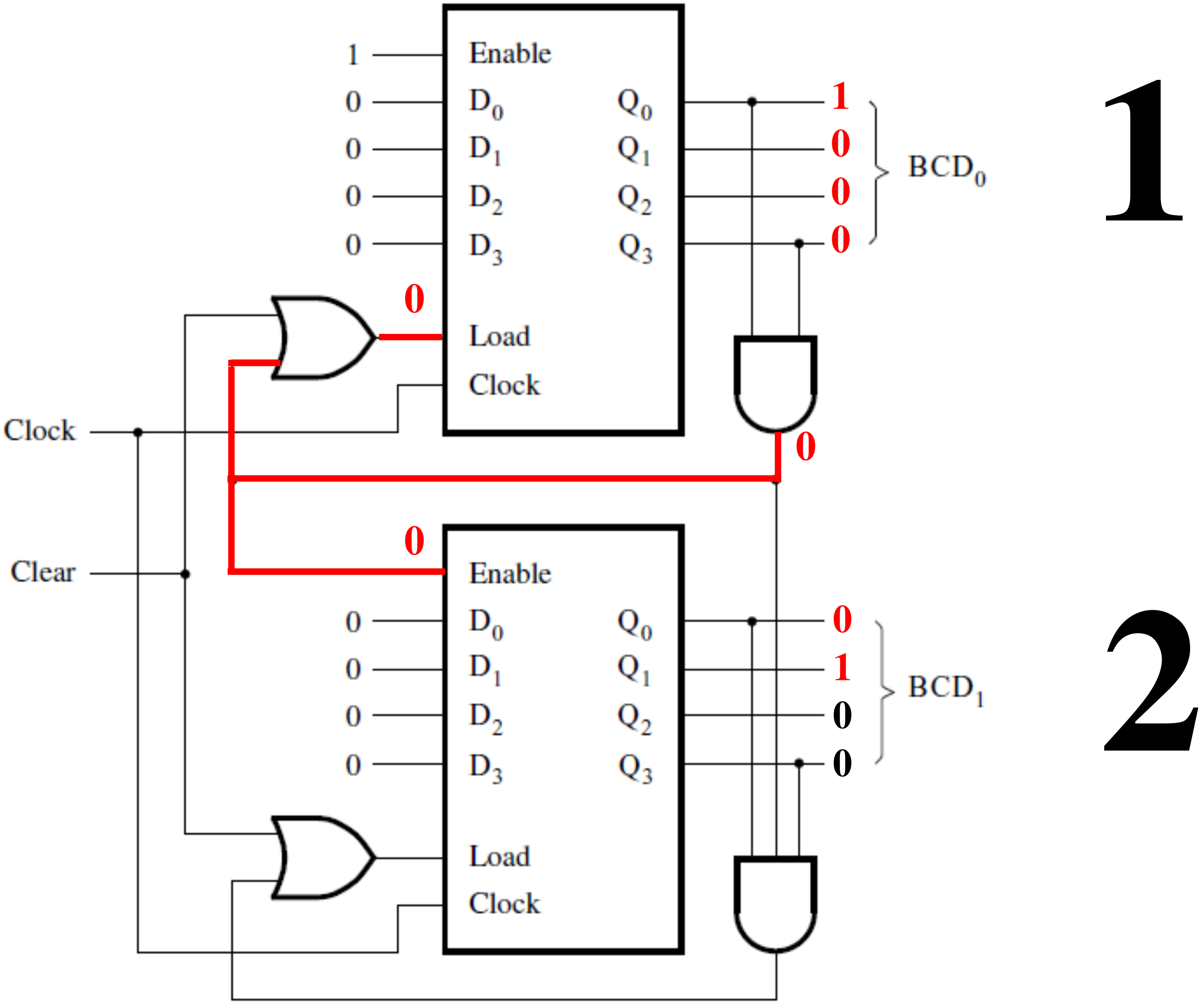
Enabling the second counter



Enabling the second counter

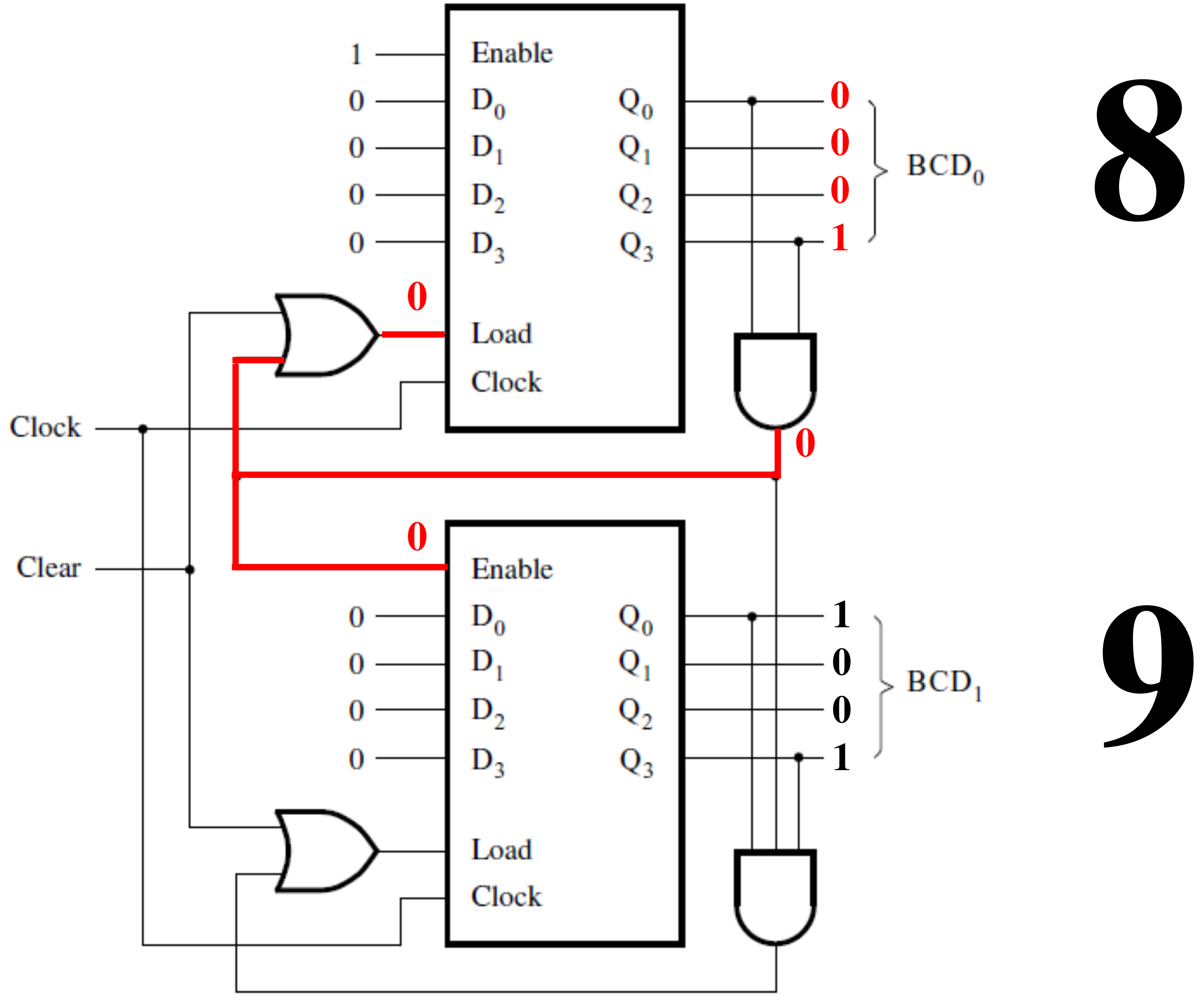


Enabling the second counter

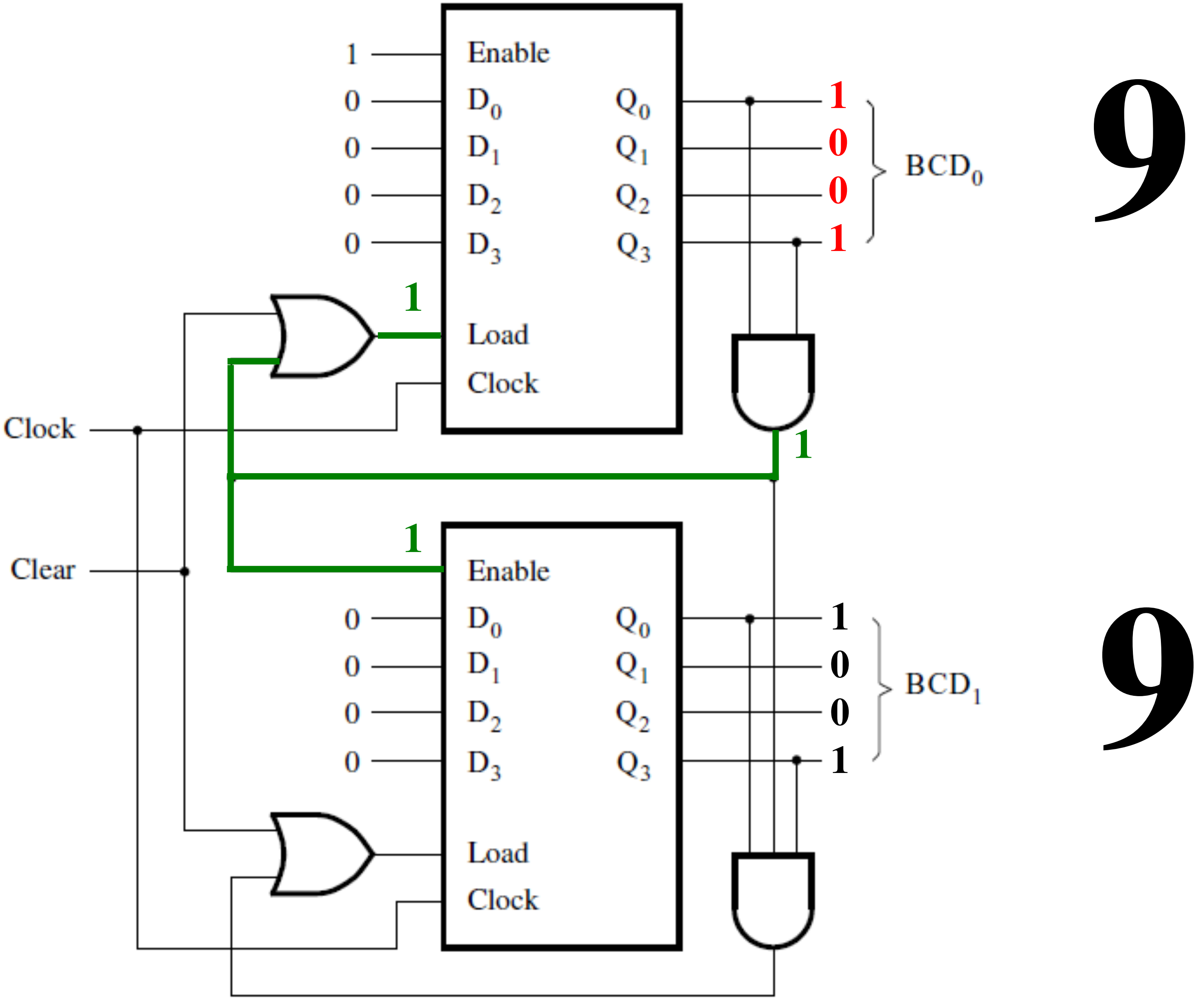


• • •

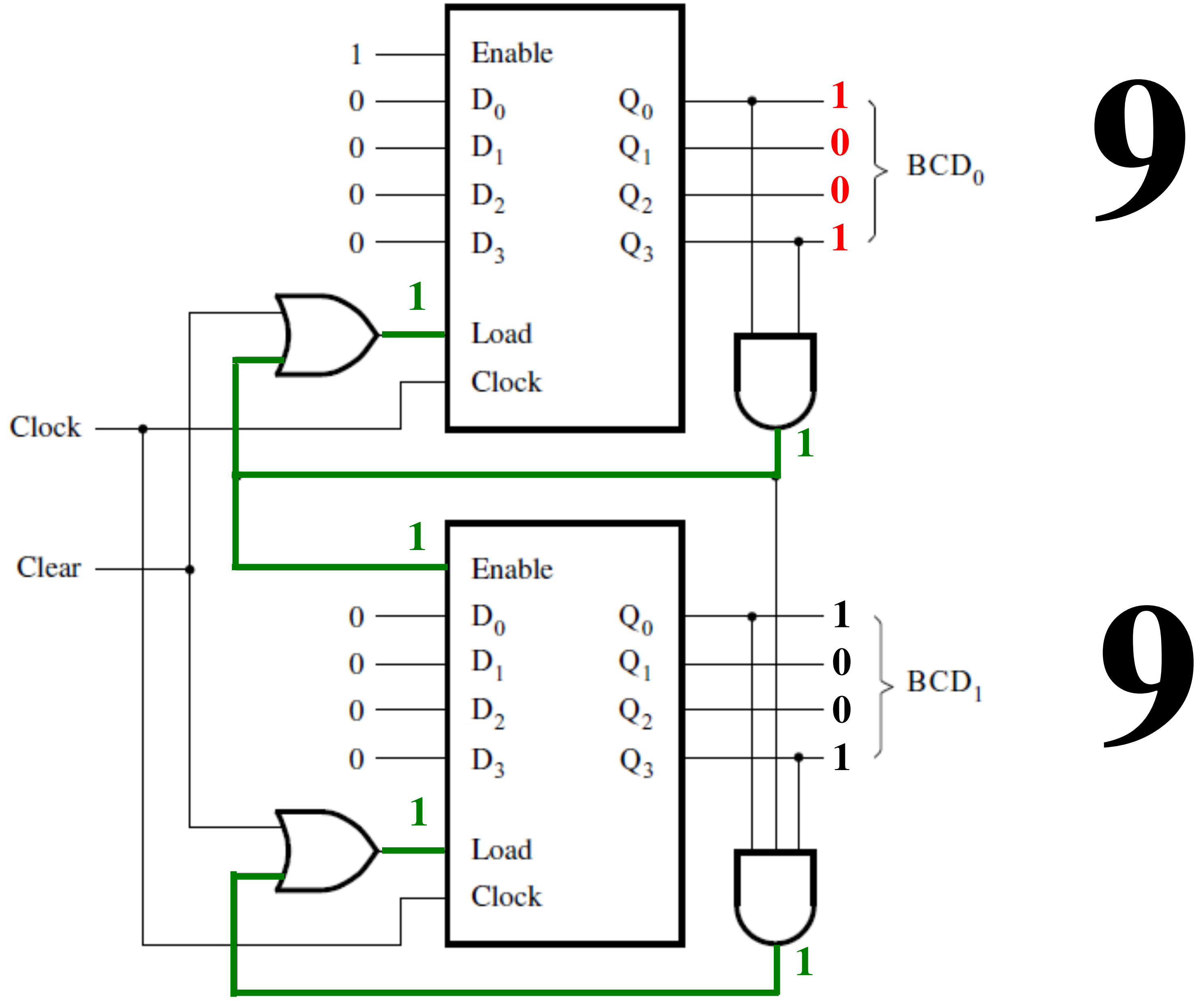
Enabling the second counter



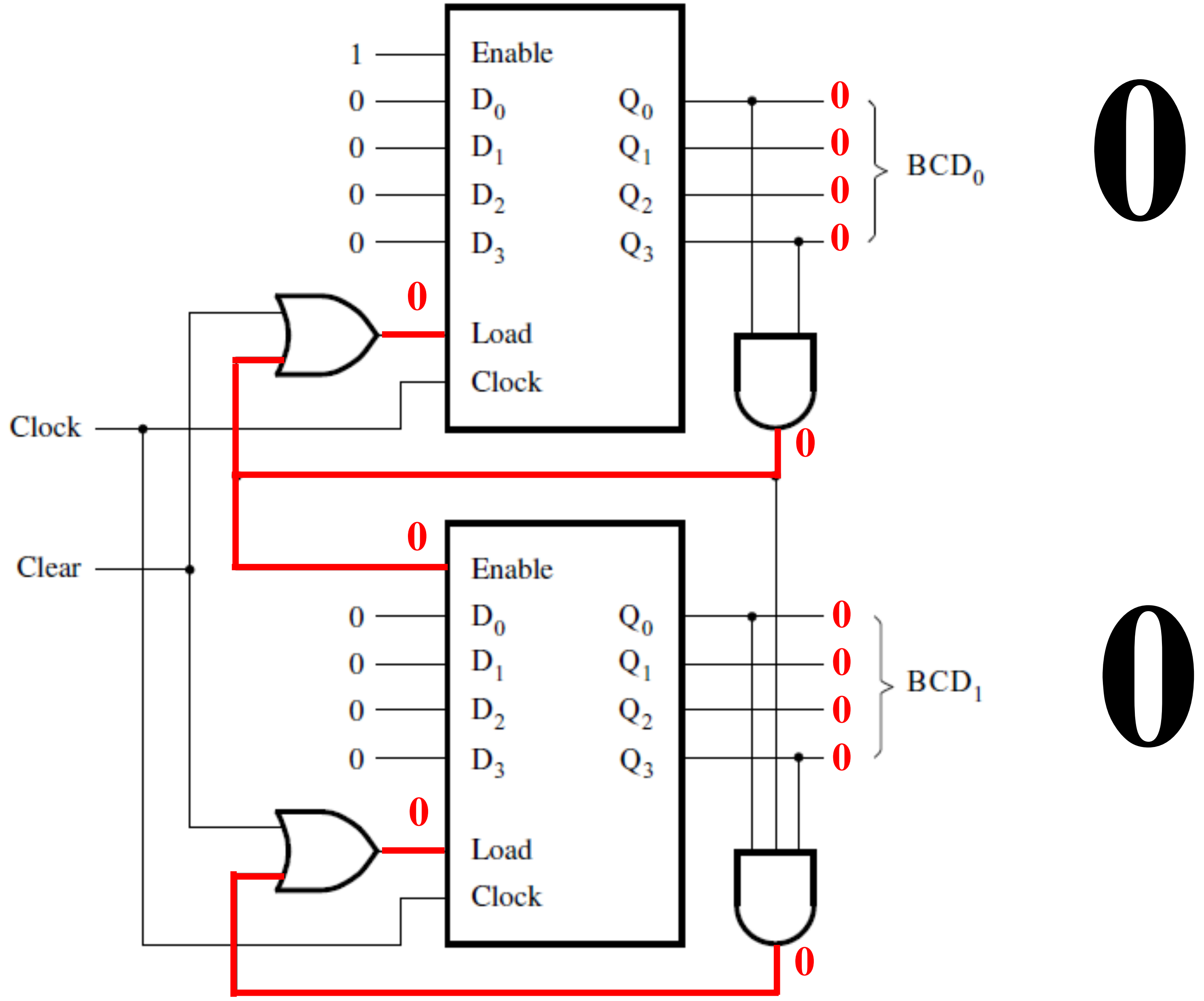
Enabling the second counter



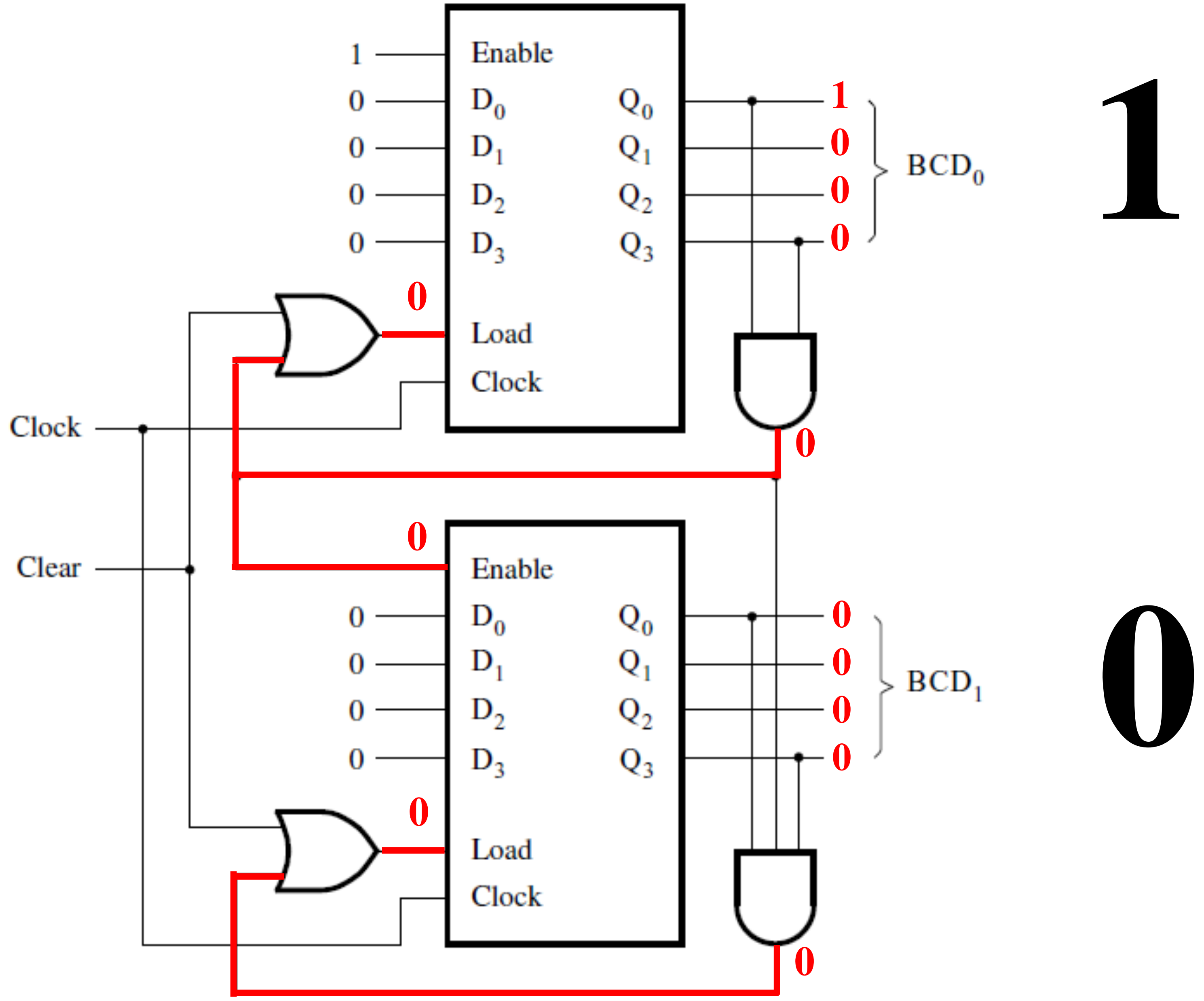
Enabling the second counter



Enabling the second counter

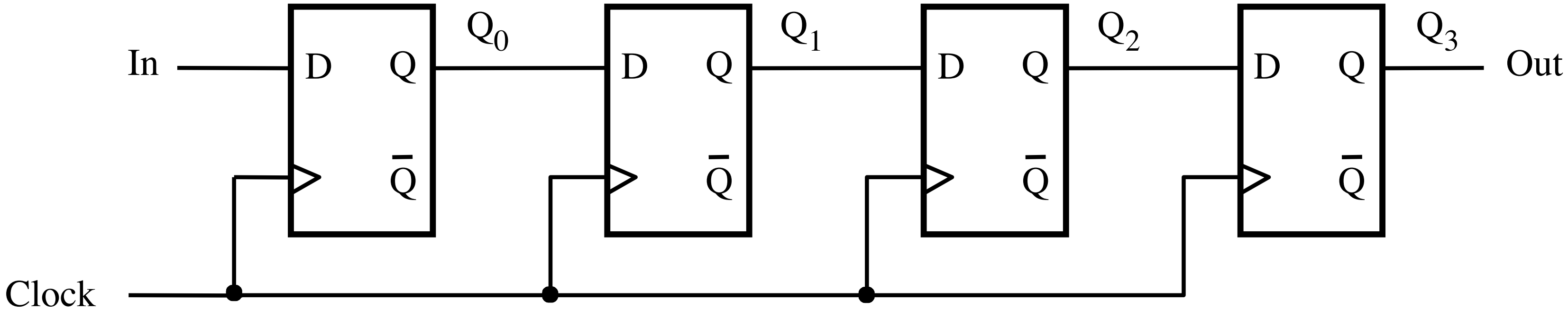


Enabling the second counter



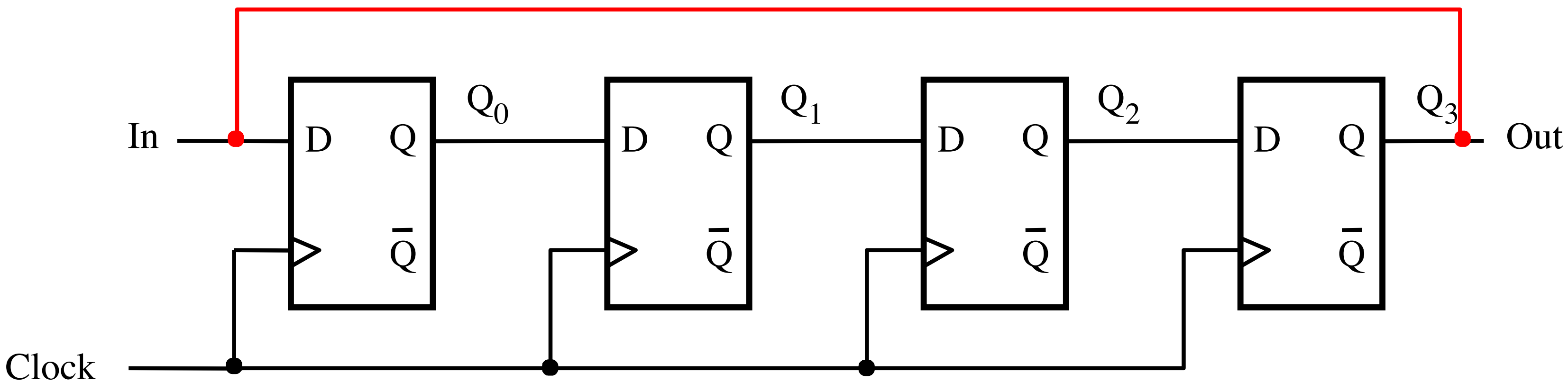
Ring Counter

How to build a 4-bit ring counter



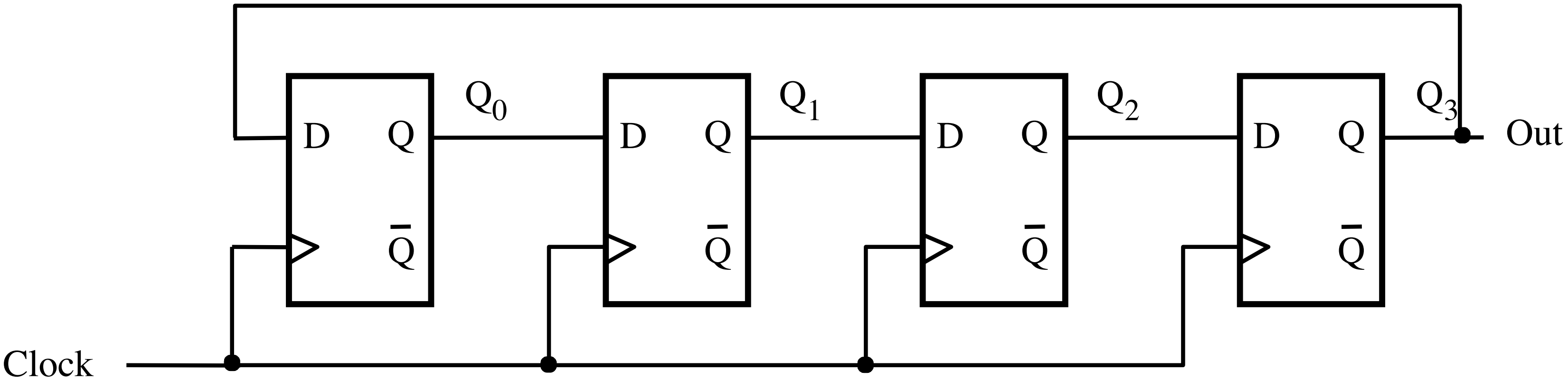
To build a ring counter we start with a shift register.

How to build a 4-bit ring counter



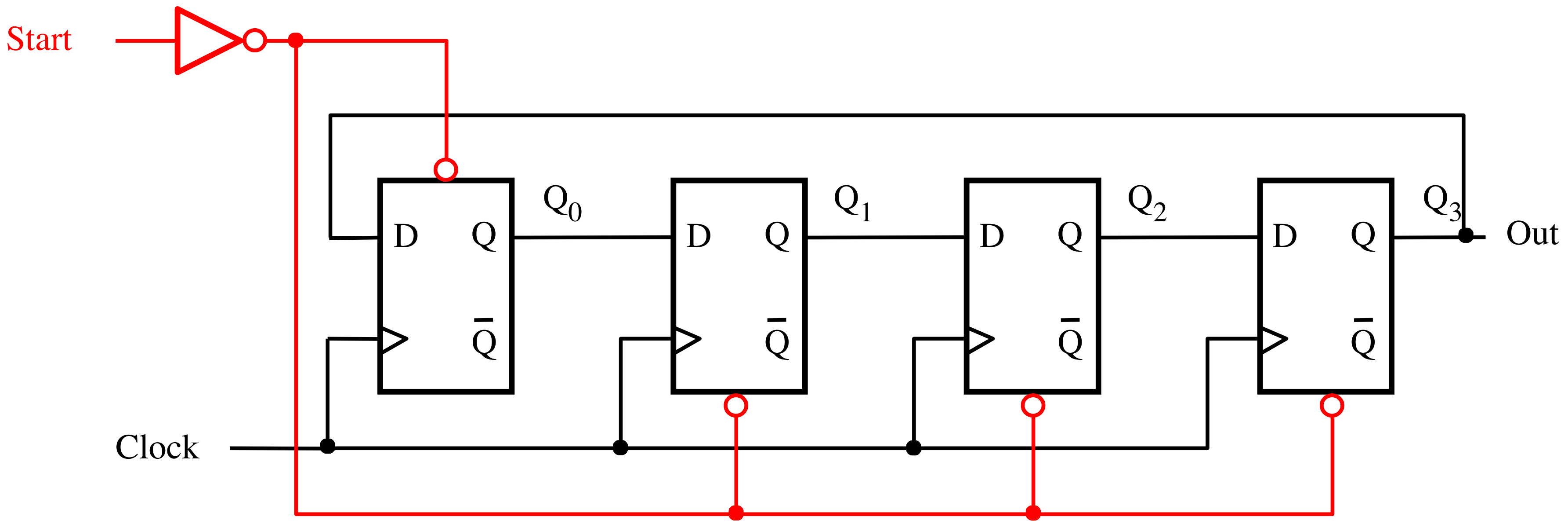
Next, add a loop from the last flip-flop to the first…

How to build a 4-bit ring counter



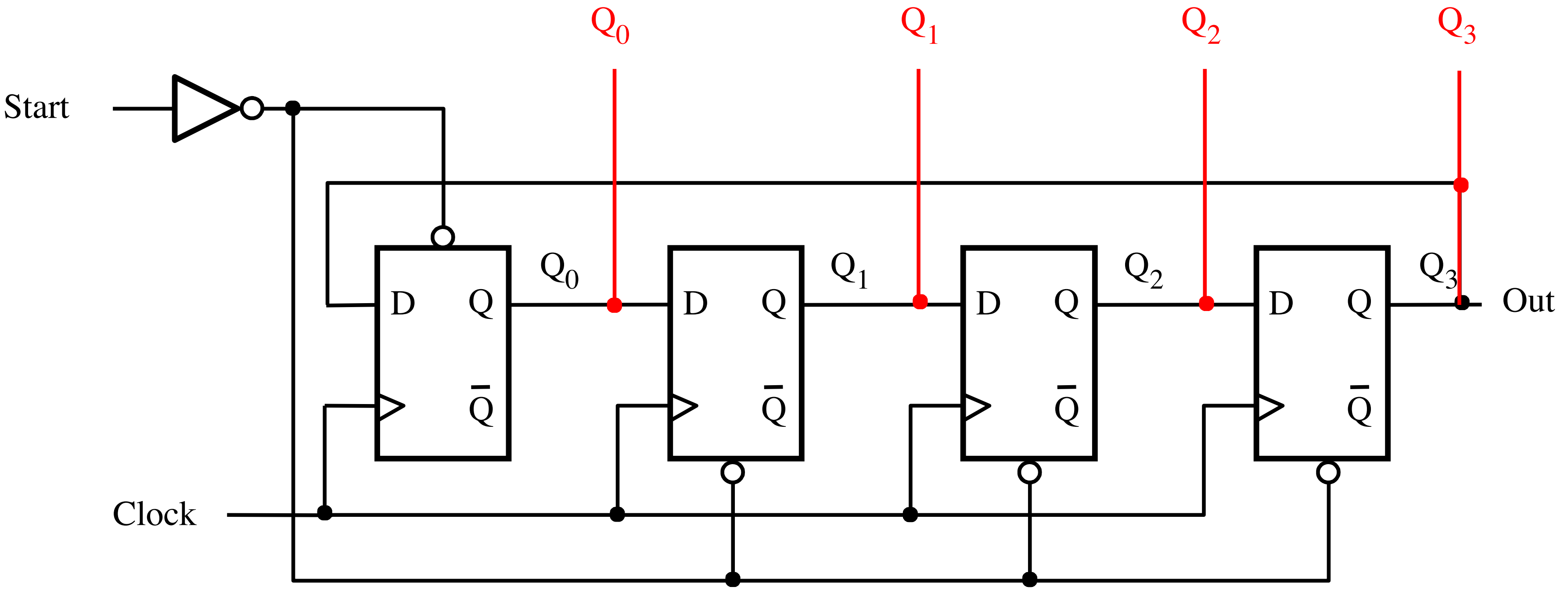
... and remove the In input line.

How to build a 4-bit ring counter



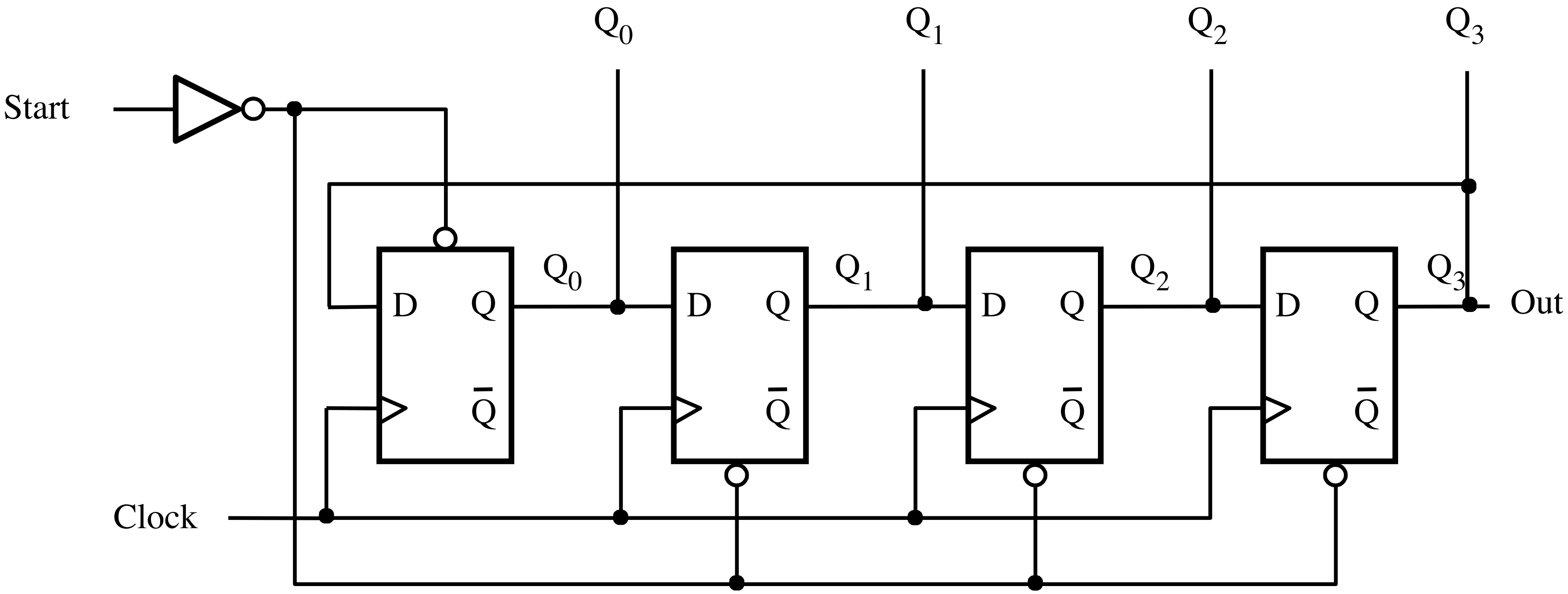
Also, add a start input that goes inverted to preset_n of the first flip=flop and to clear_n of all remaining flip-flops.

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.

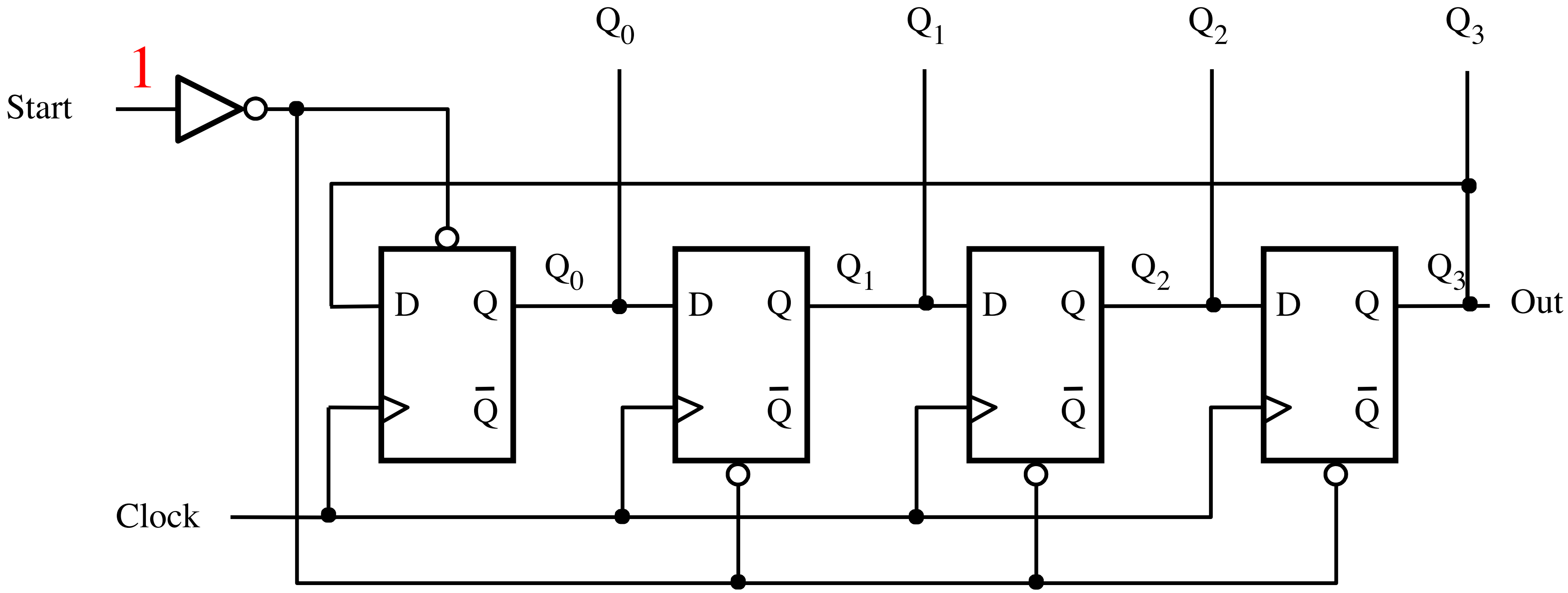
4-bit ring counter

- **There is only one 1 on the outputs of the four flip-flops**

- **The counting sequence is: 1000, 0100, 0010, 0001, 1000, …**

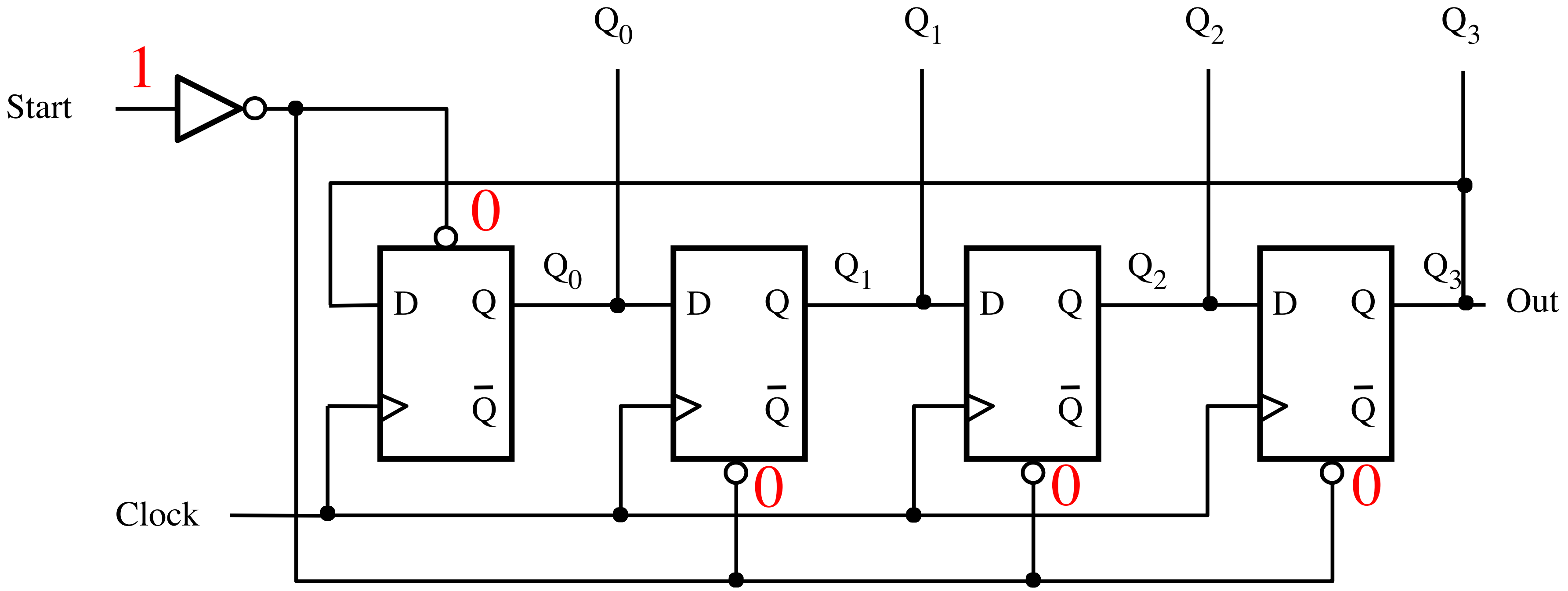
- **To reset the counter**
  - **set start to 1 for a short period of time**
  - **This sets the four outputs to 1000**

4-bit ring counter: How does it work



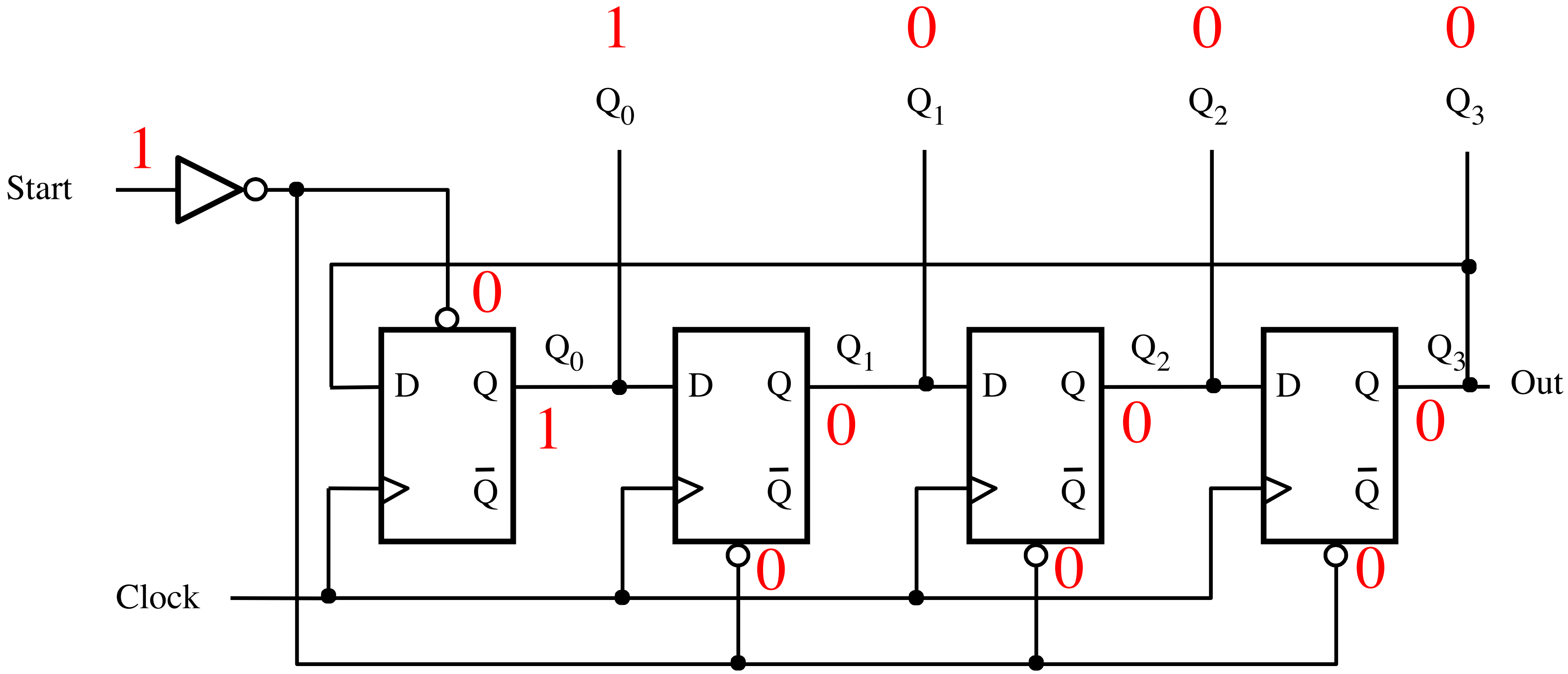
To initialize the counter set Start to 1.

4-bit ring counter: How does it work



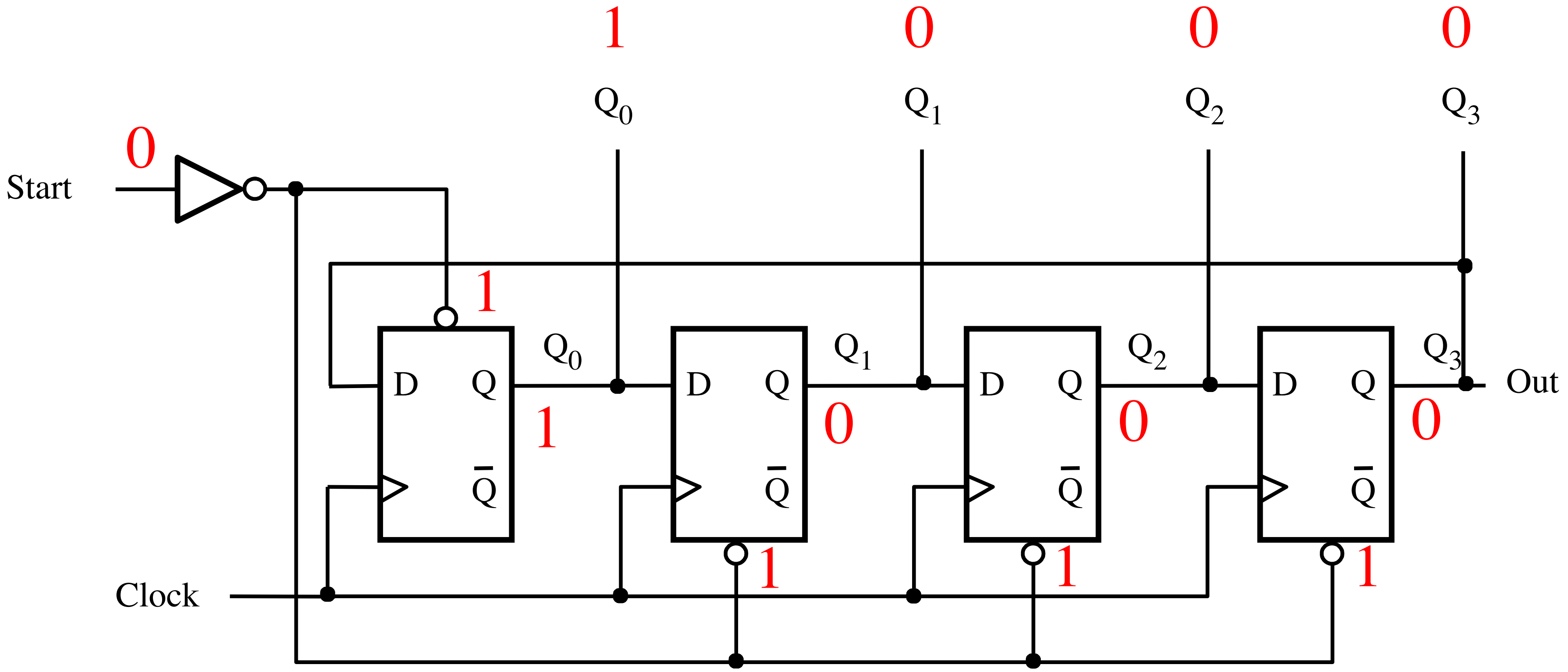
After the NOT gate, this 1 goes as 0 to preset_n of the first flip-flop and to clear_n of all remaining flip-flops.

4-bit ring counter: How does it work



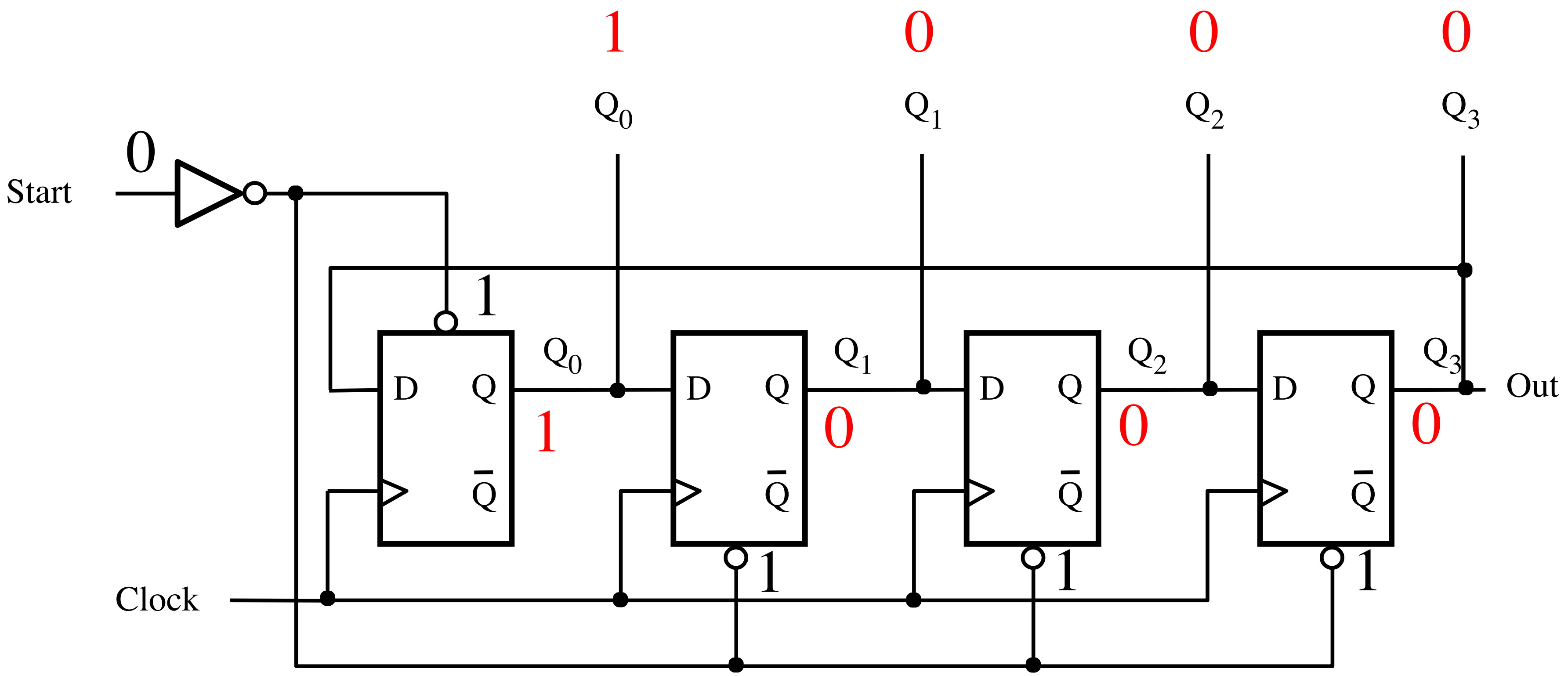
This sets the output pattern to 1000,
i.e., only the first bit is one and the rest are zeros.

4-bit ring counter: How does it work



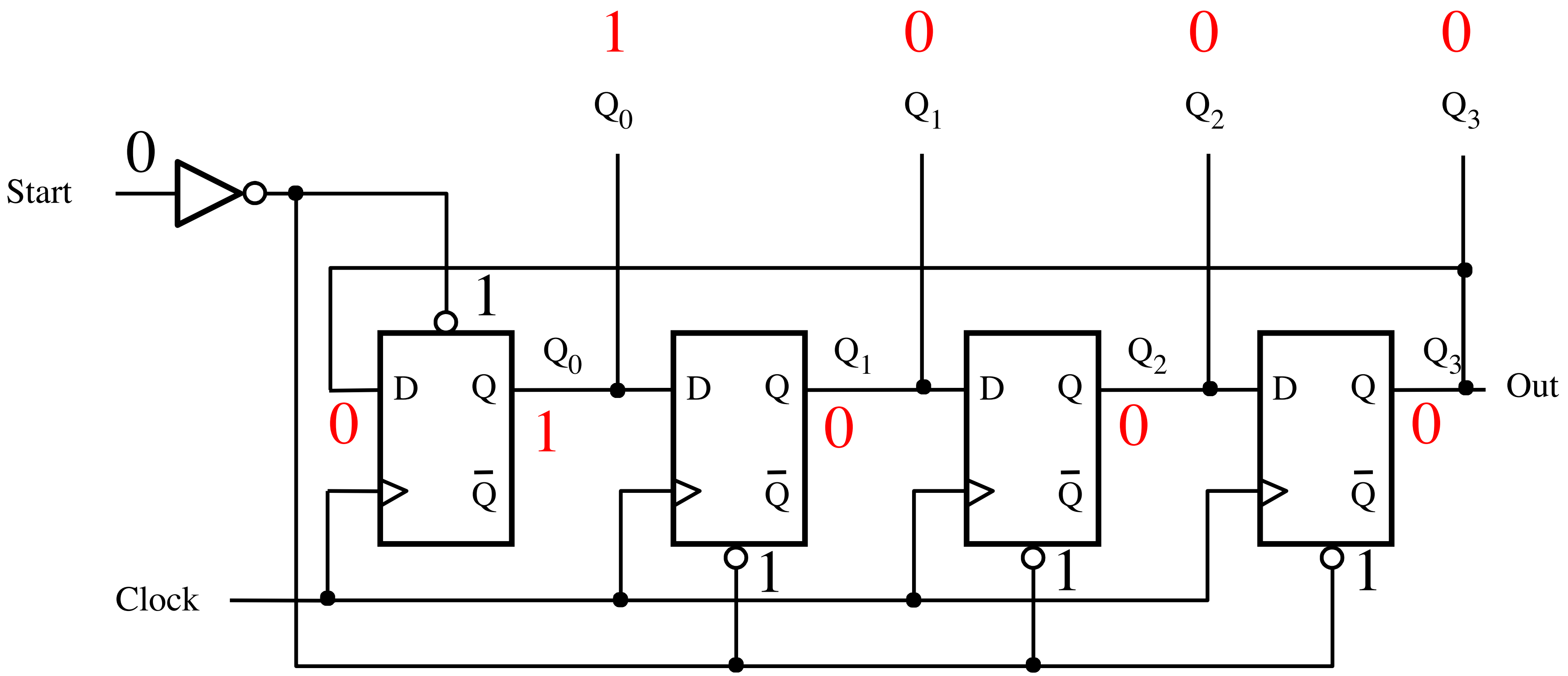
Setting Start to 0 has no effect on the outputs,
because both preset_n and clear_n are sensitive only to 0.

4-bit ring counter: How does it work



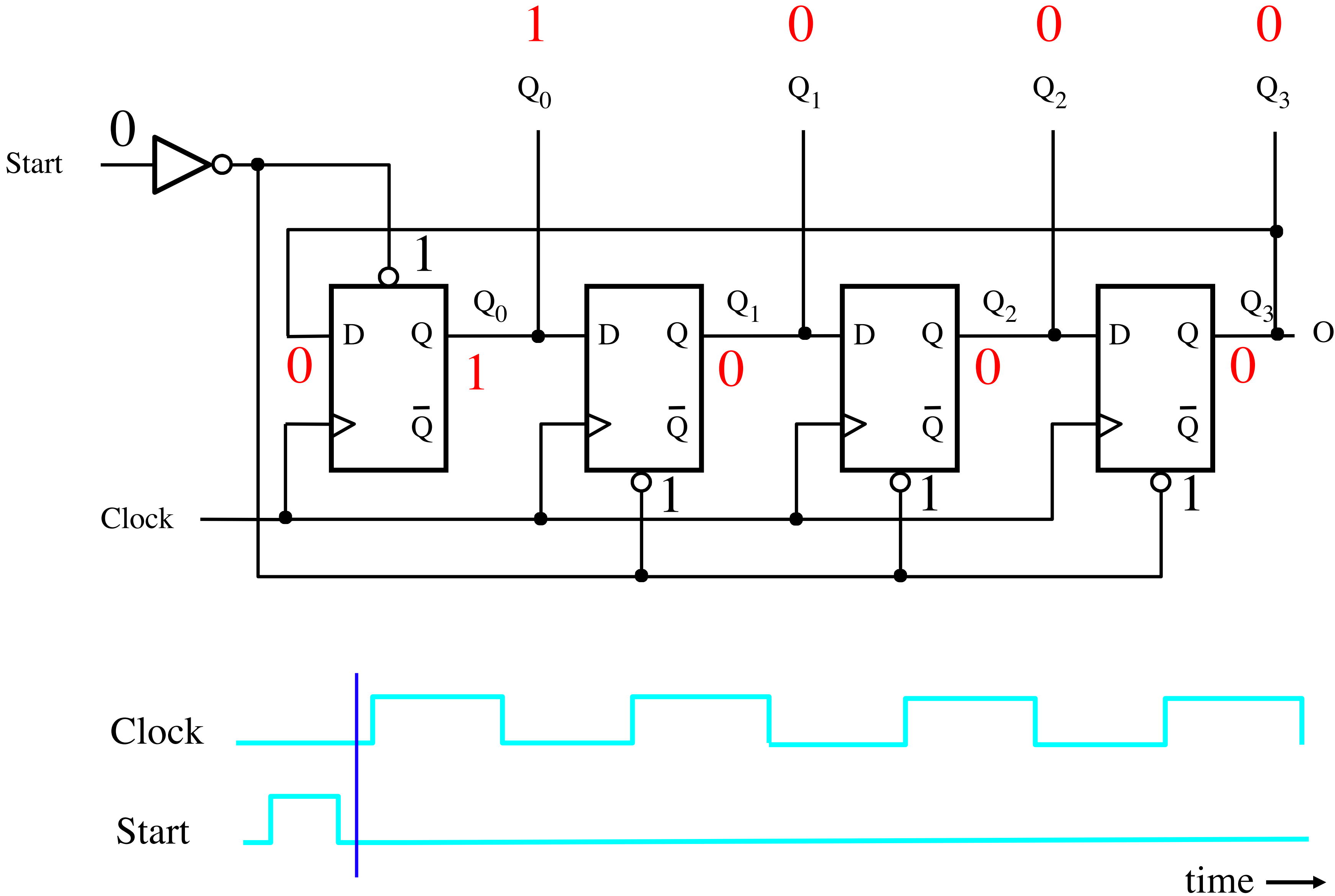
The initialization does not depend on the clock since both preset_n and clear_n bypass the gates of the latches in the flip-flops.

4-bit ring counter: How does it work

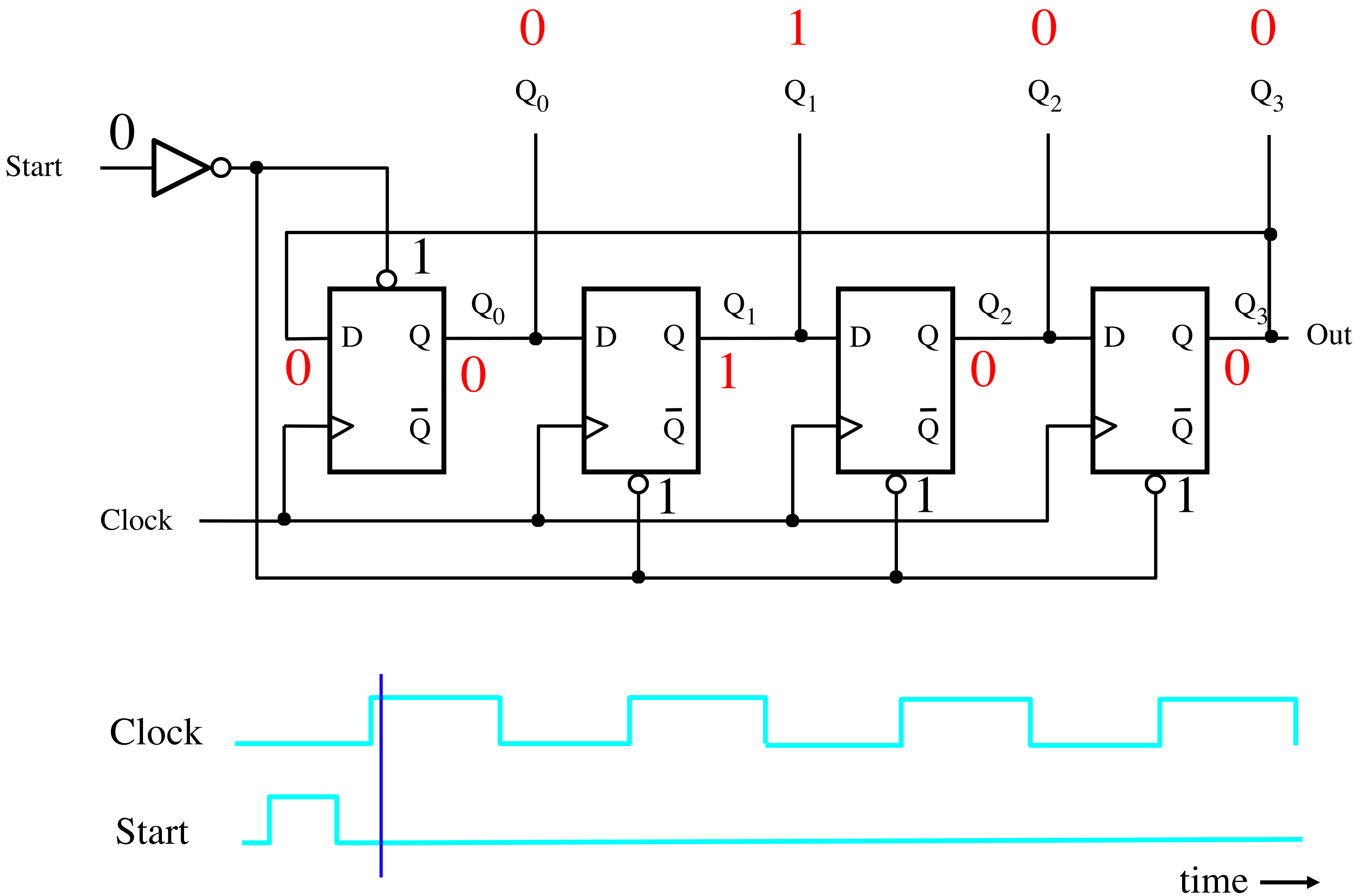


That last 0 loops back to the D input of the first flip-flop.

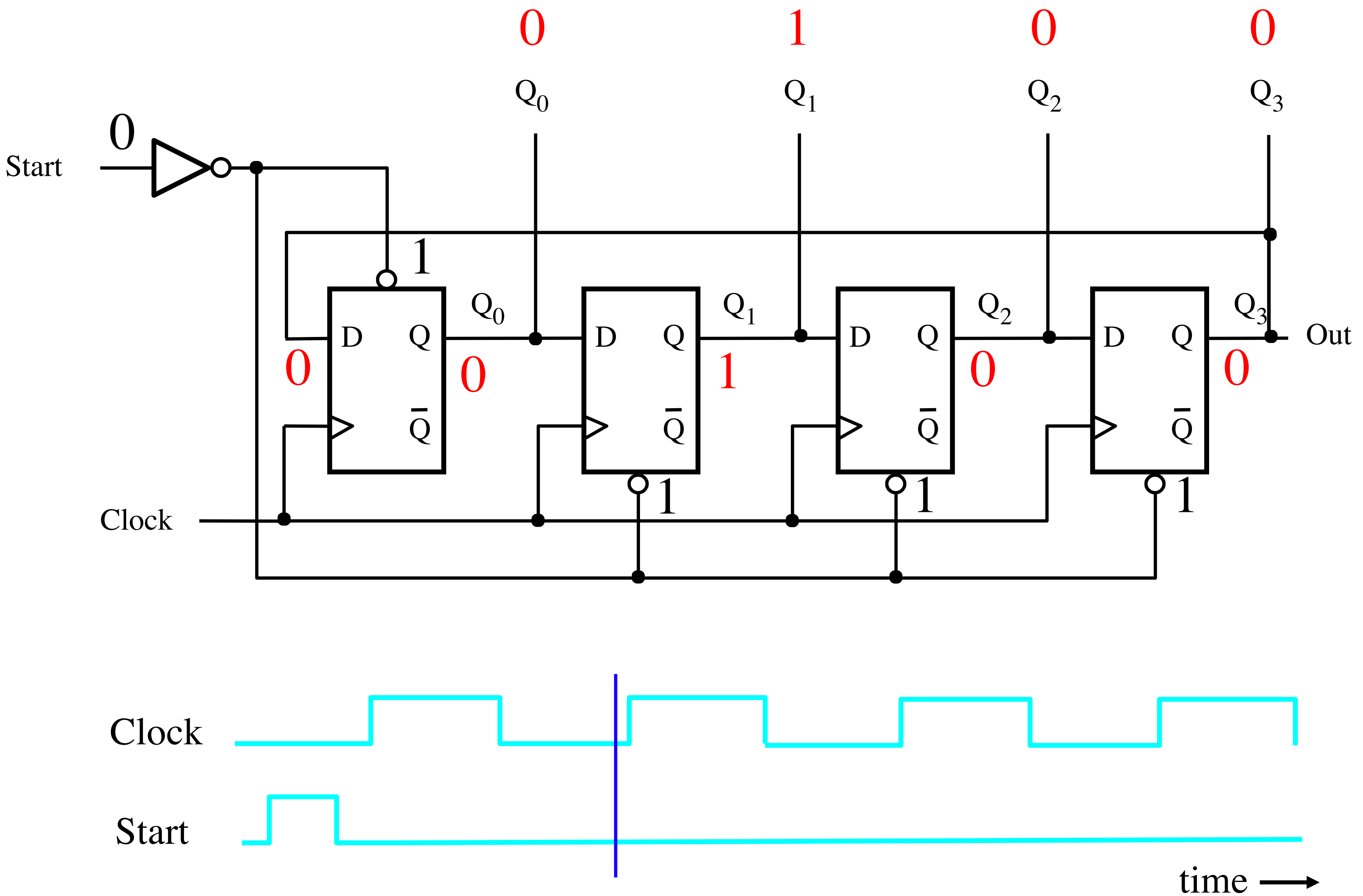
4-bit ring counter: How does it work



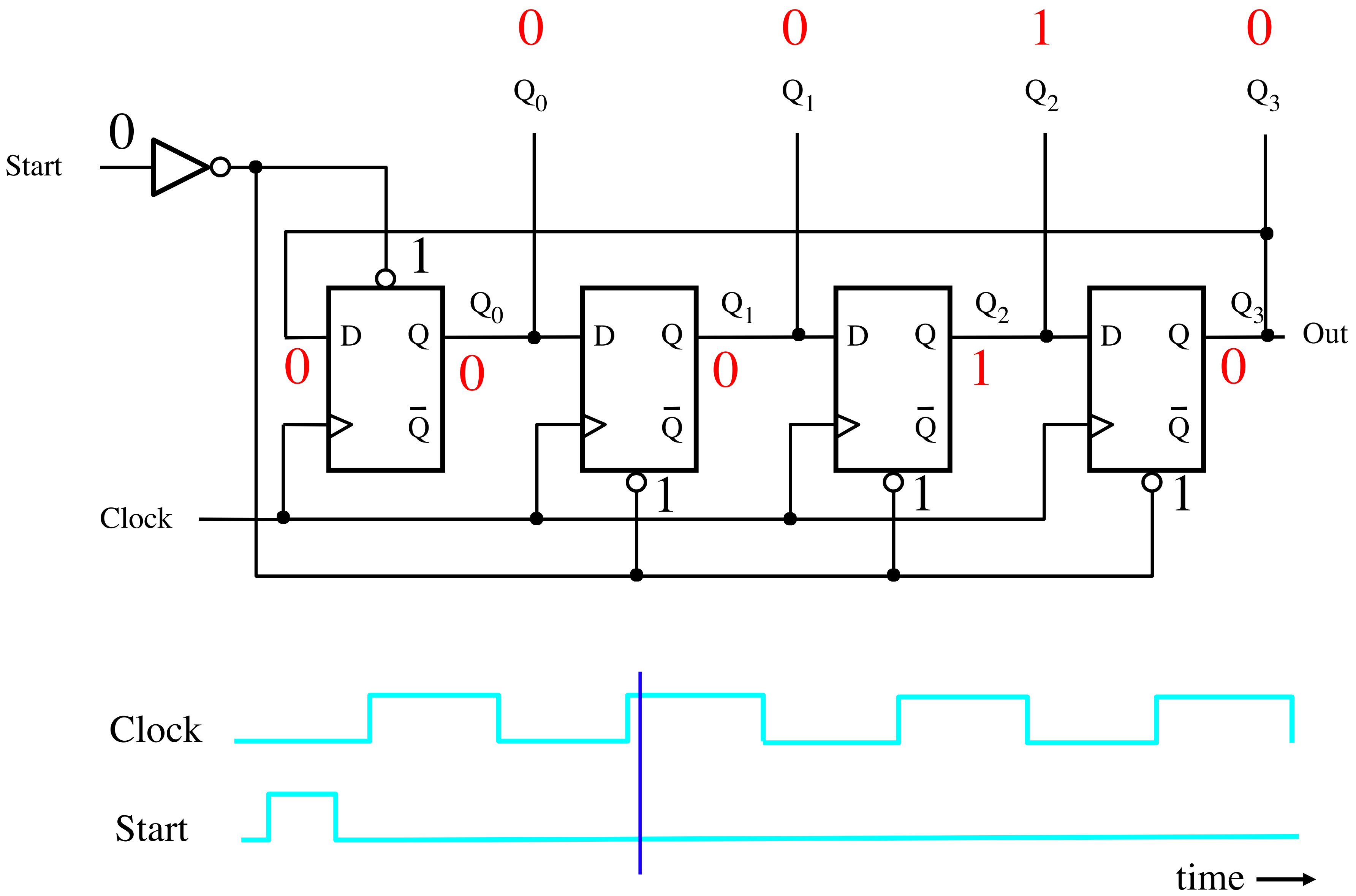
4-bit ring counter: How does it work



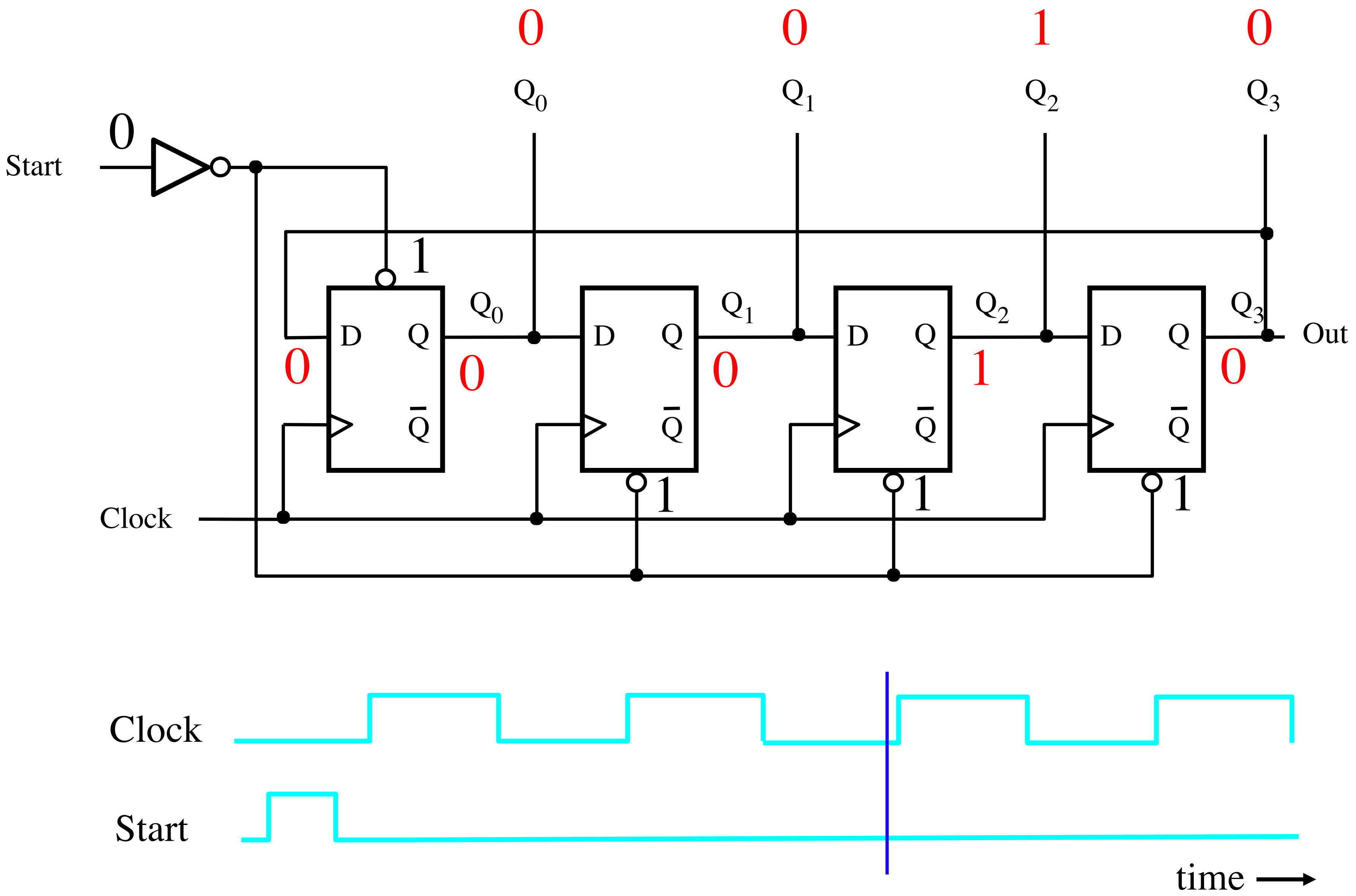
4-bit ring counter: How does it work



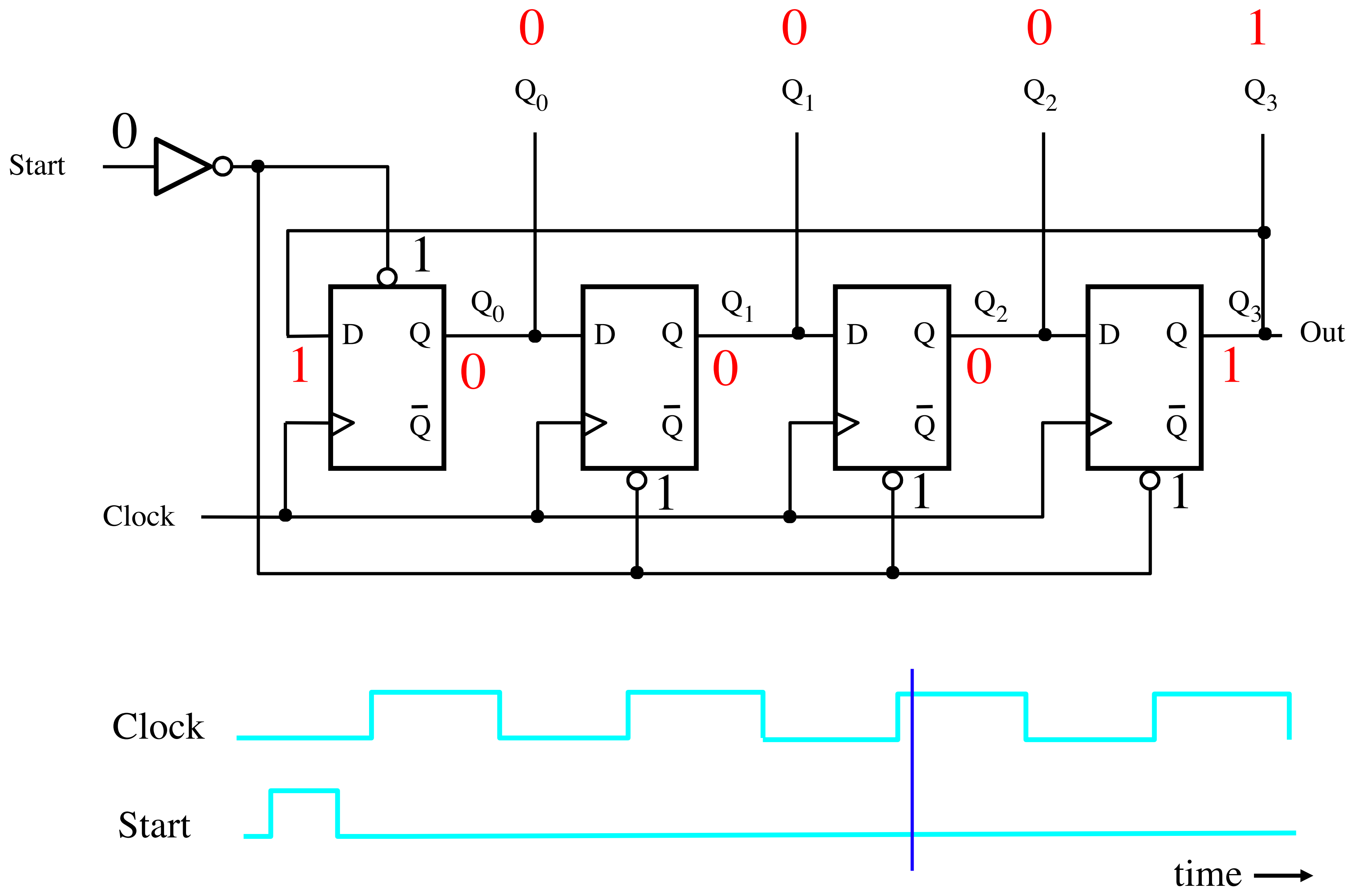
4-bit ring counter: How does it work



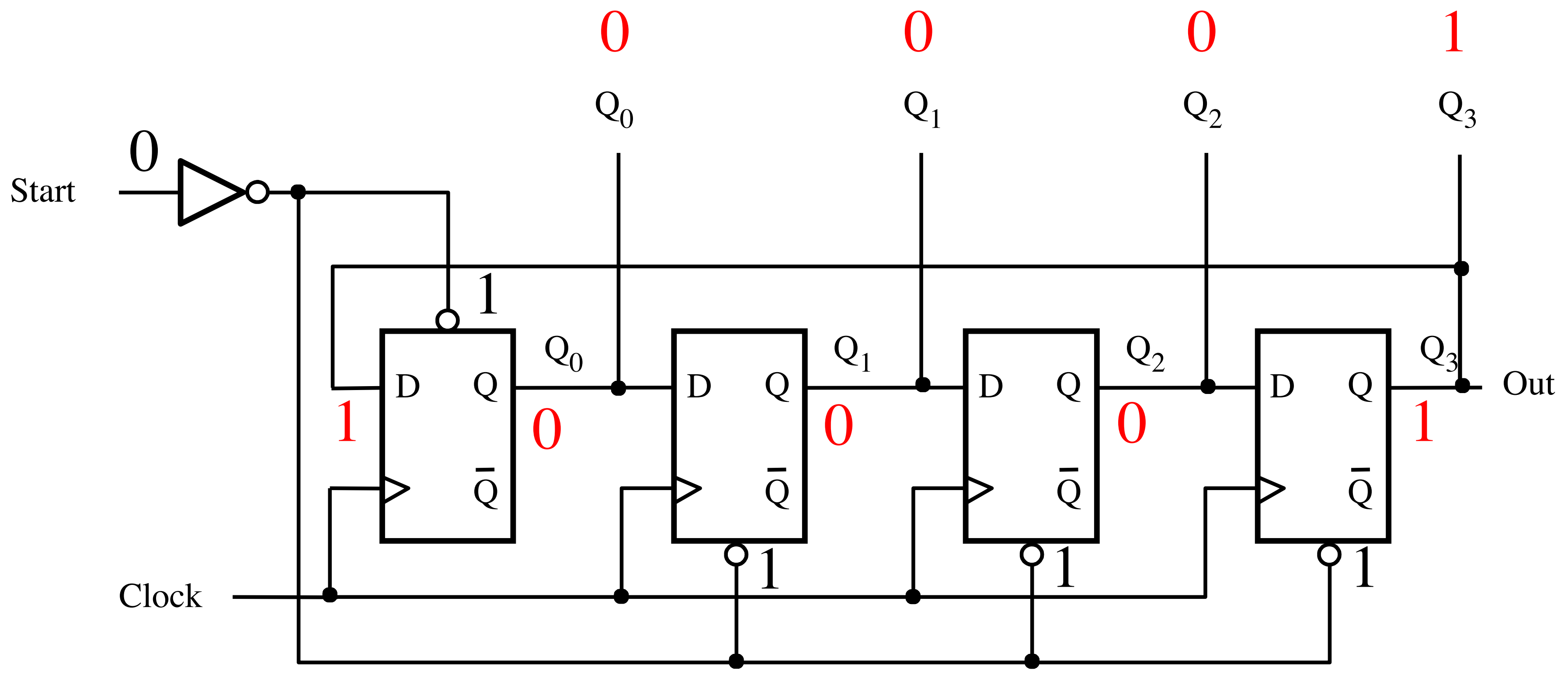
4-bit ring counter: How does it work

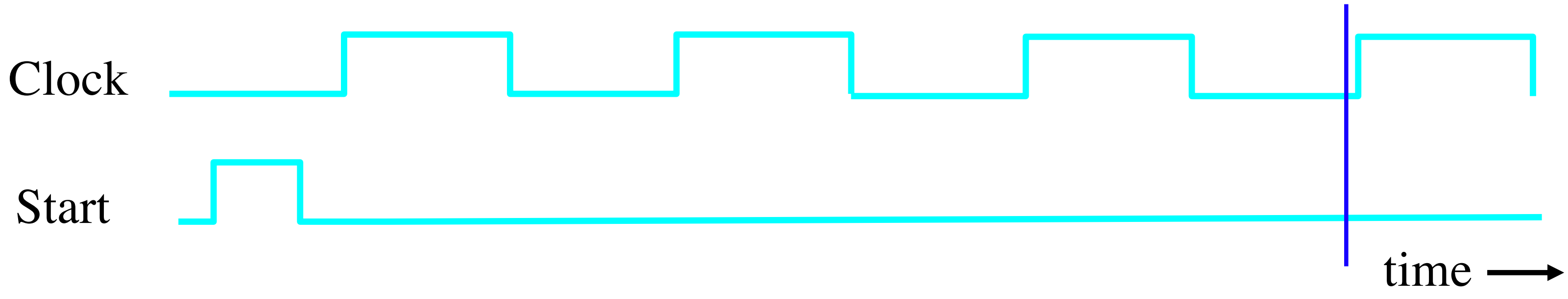


4-bit ring counter: How does it work

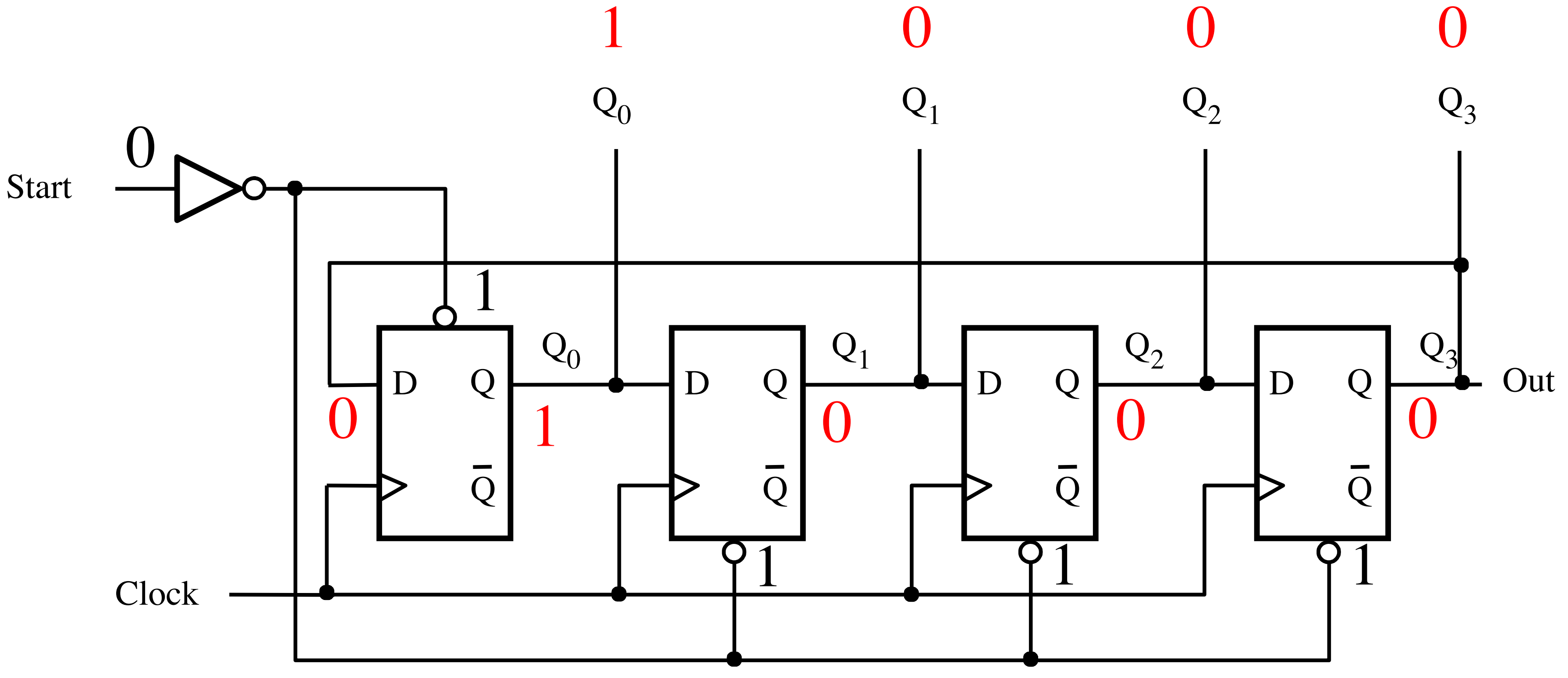


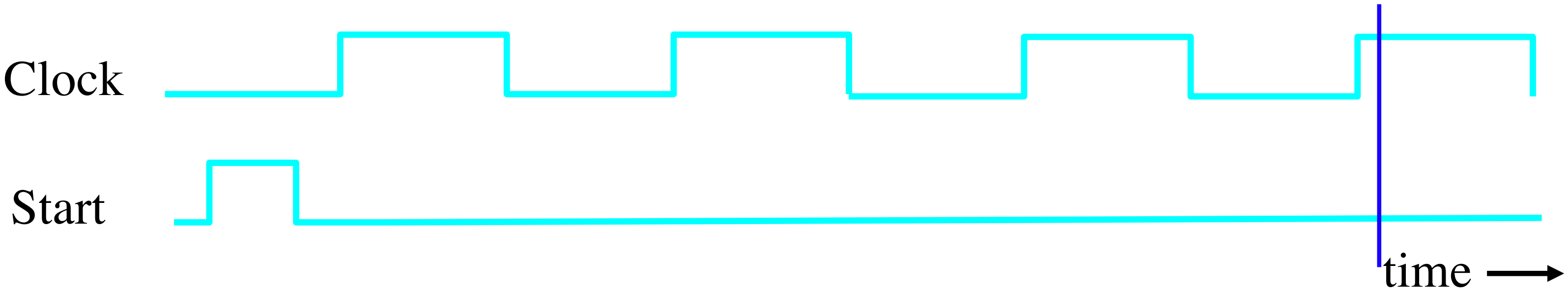
4-bit ring counter: How does it work



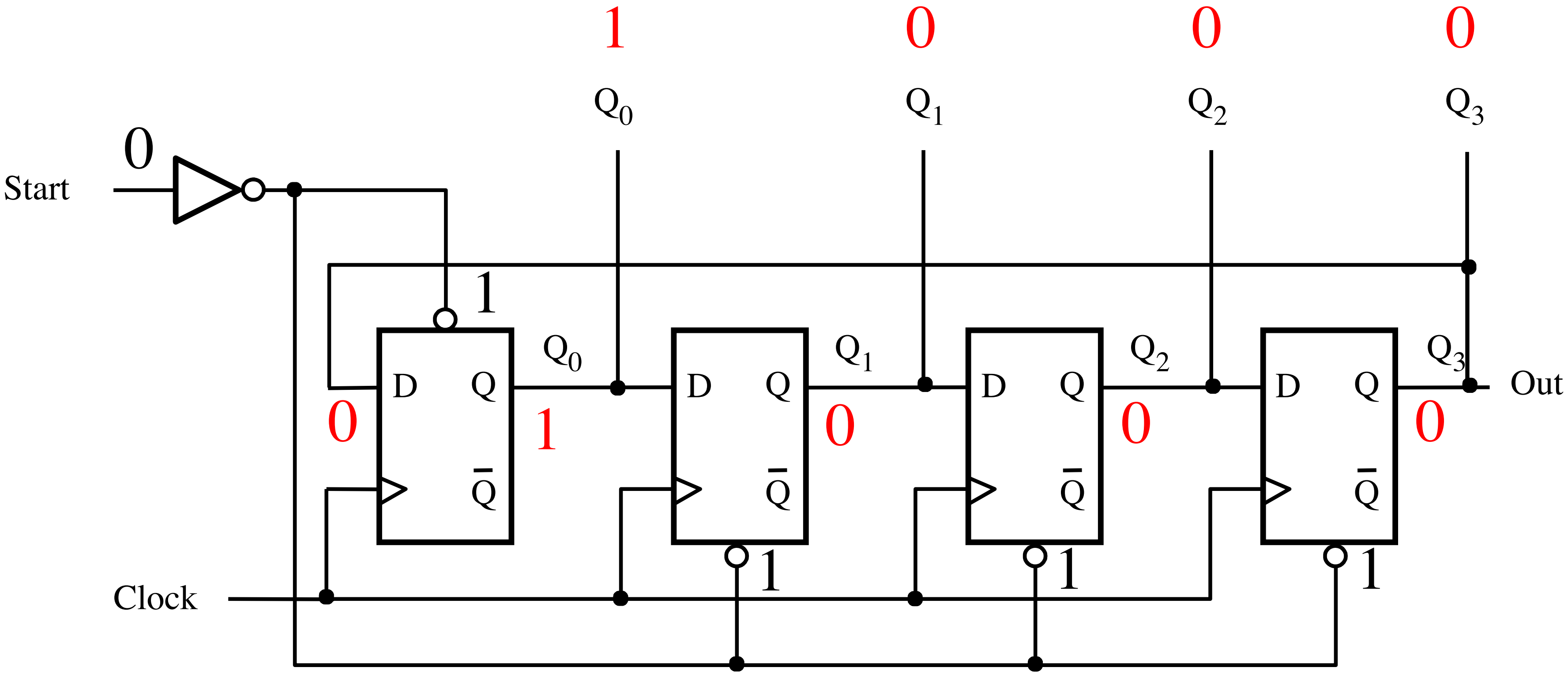


4-bit ring counter: How does it work



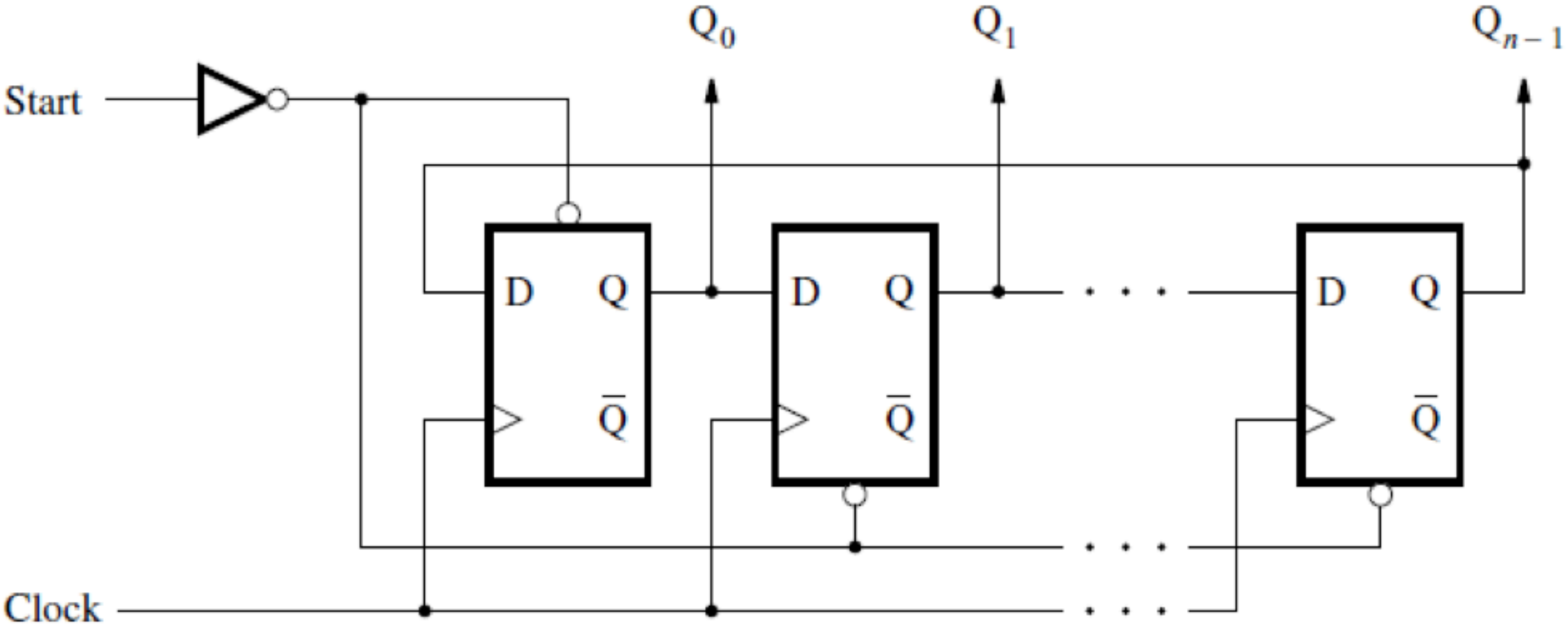


4-bit ring counter: How does it work



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

n-bit ring counter



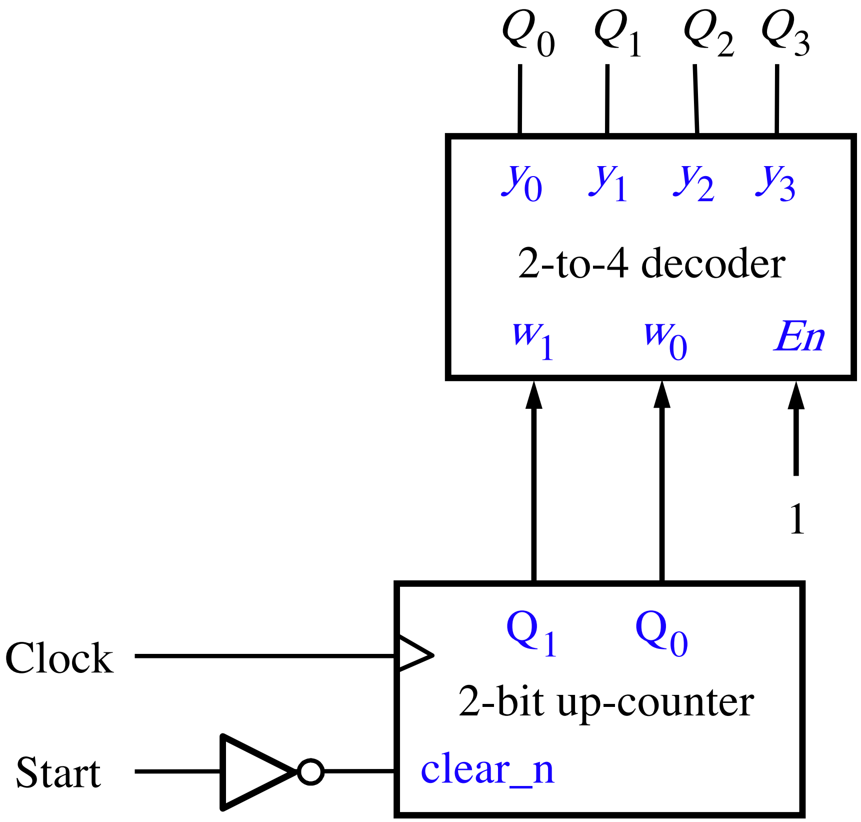
[Figure 5.28a from the textbook]

Ring Counter
(alternative implementation)

Alternative version of a 4-bit ring counter

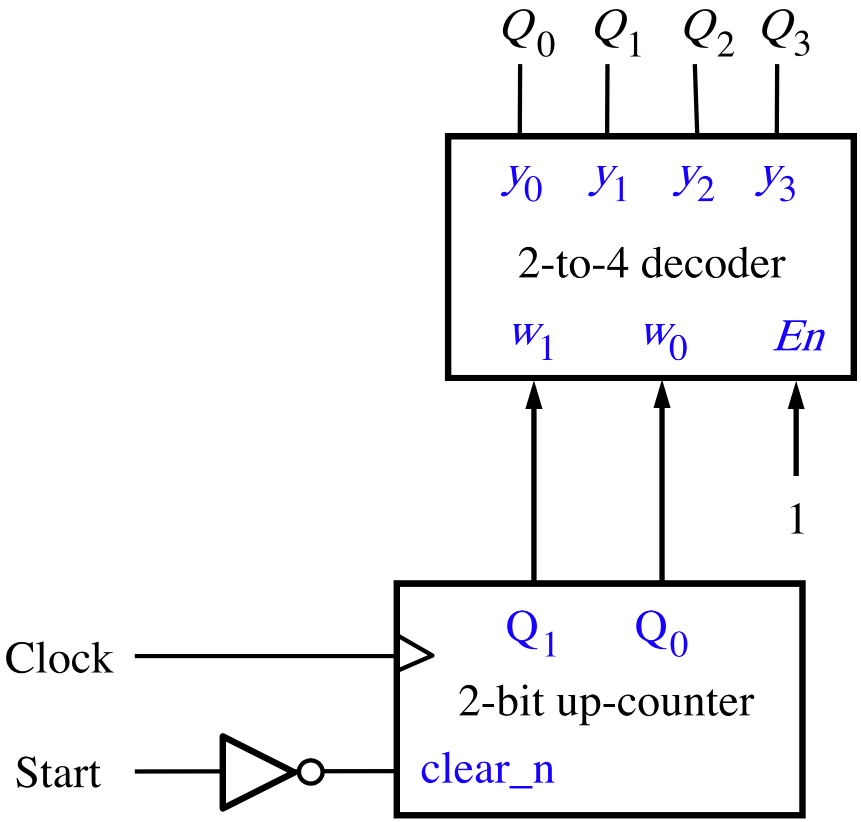
- This implementation uses 2-bit up-counter followed by a 2-to-4 decoder.
- The counter cycles through 00, 01, 10, 11, 00, …
- Recall that the outputs of the decoder are one-hot encoded. Thus, there is only one 1 on its outputs.
- Because the output of the counter is the input to the decoder, the outputs 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

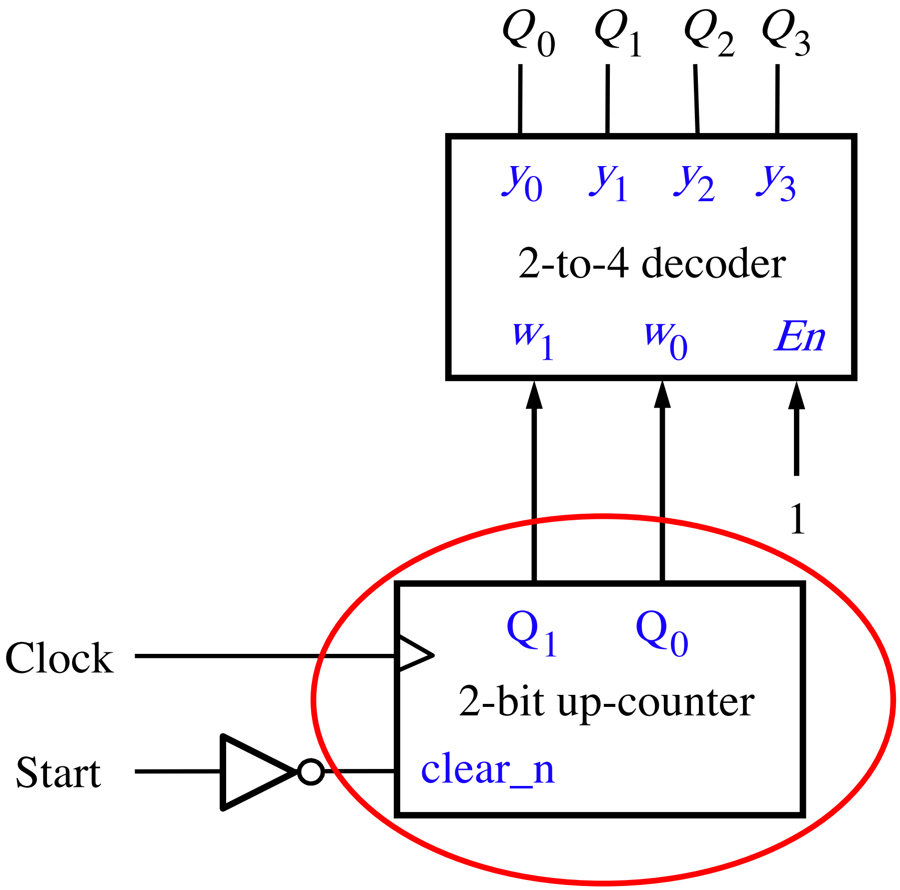


[Figure 5.28b from the textbook]

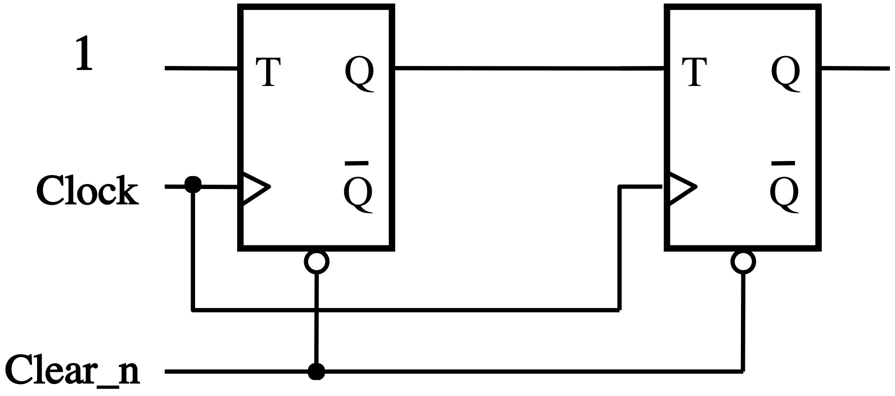
What are the components?



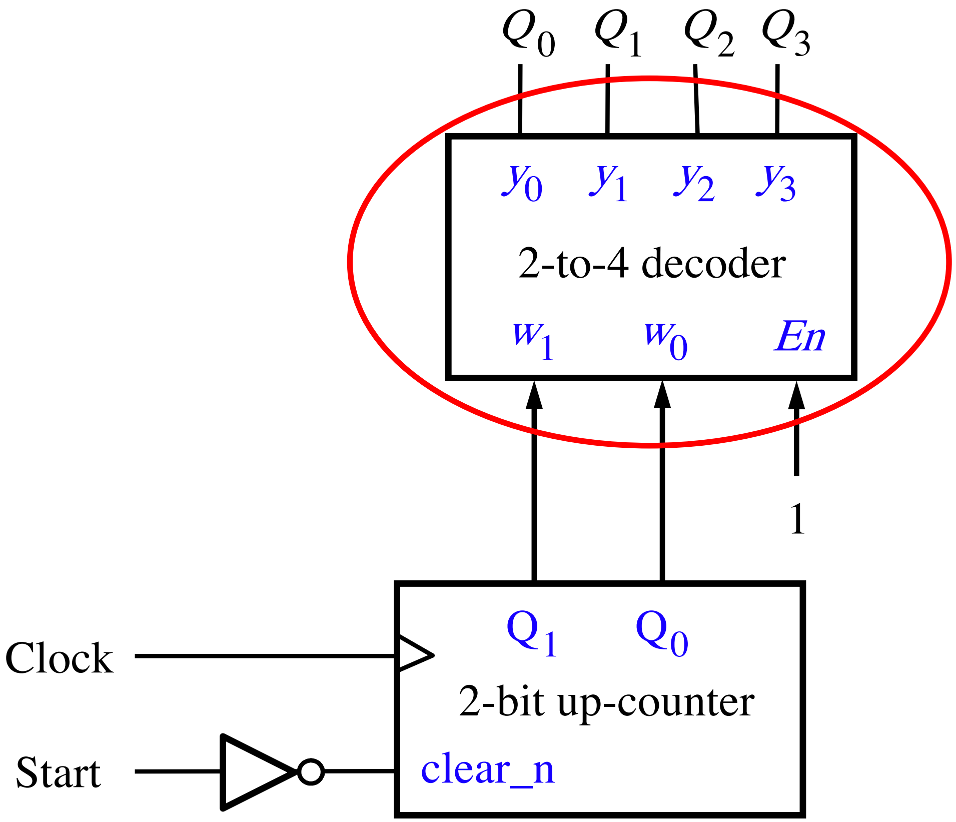
2-Bit Synchronous Up-Counter



2-Bit Synchronous Up-Counter

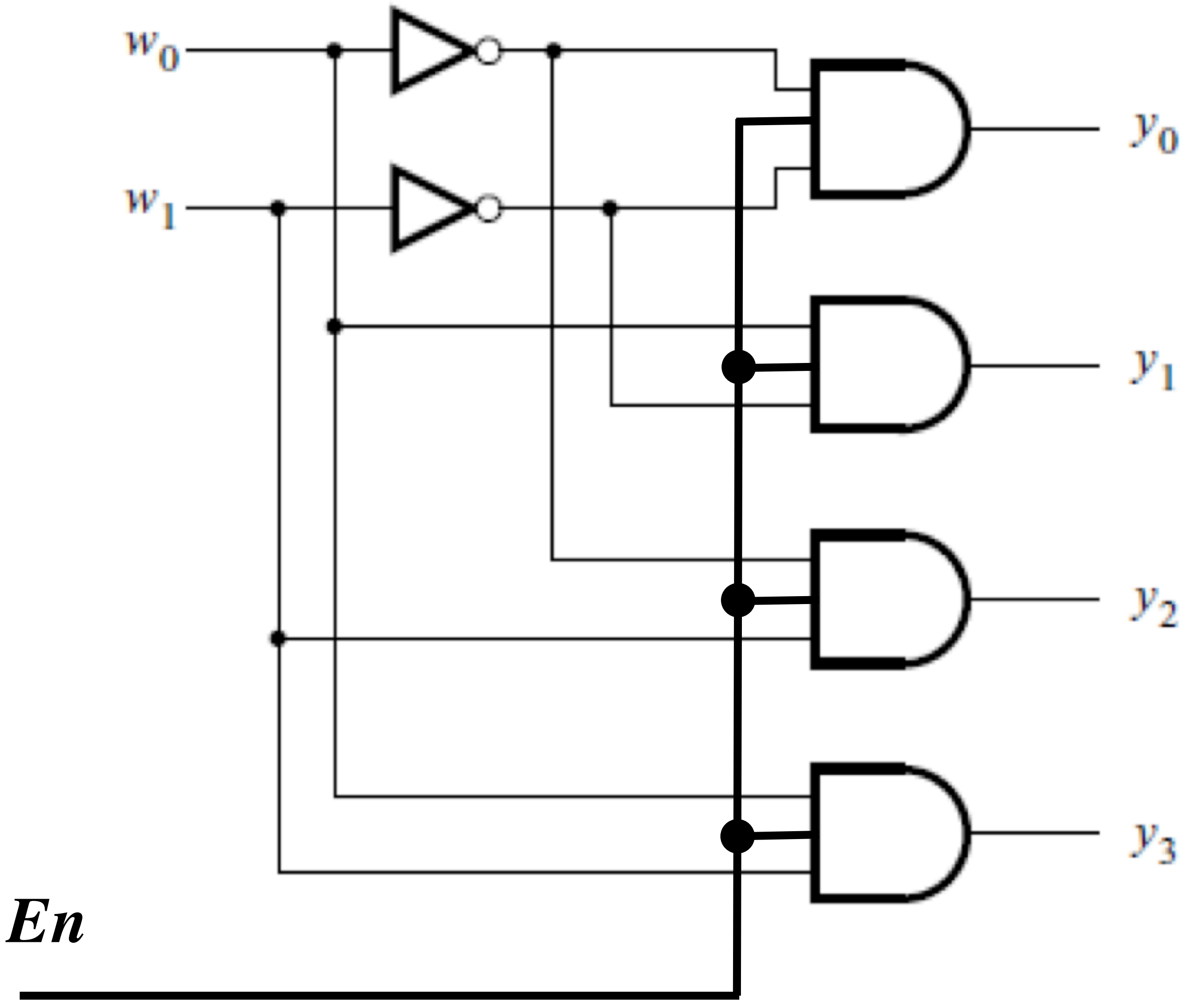


2-to-4 Decoder with Enable Input



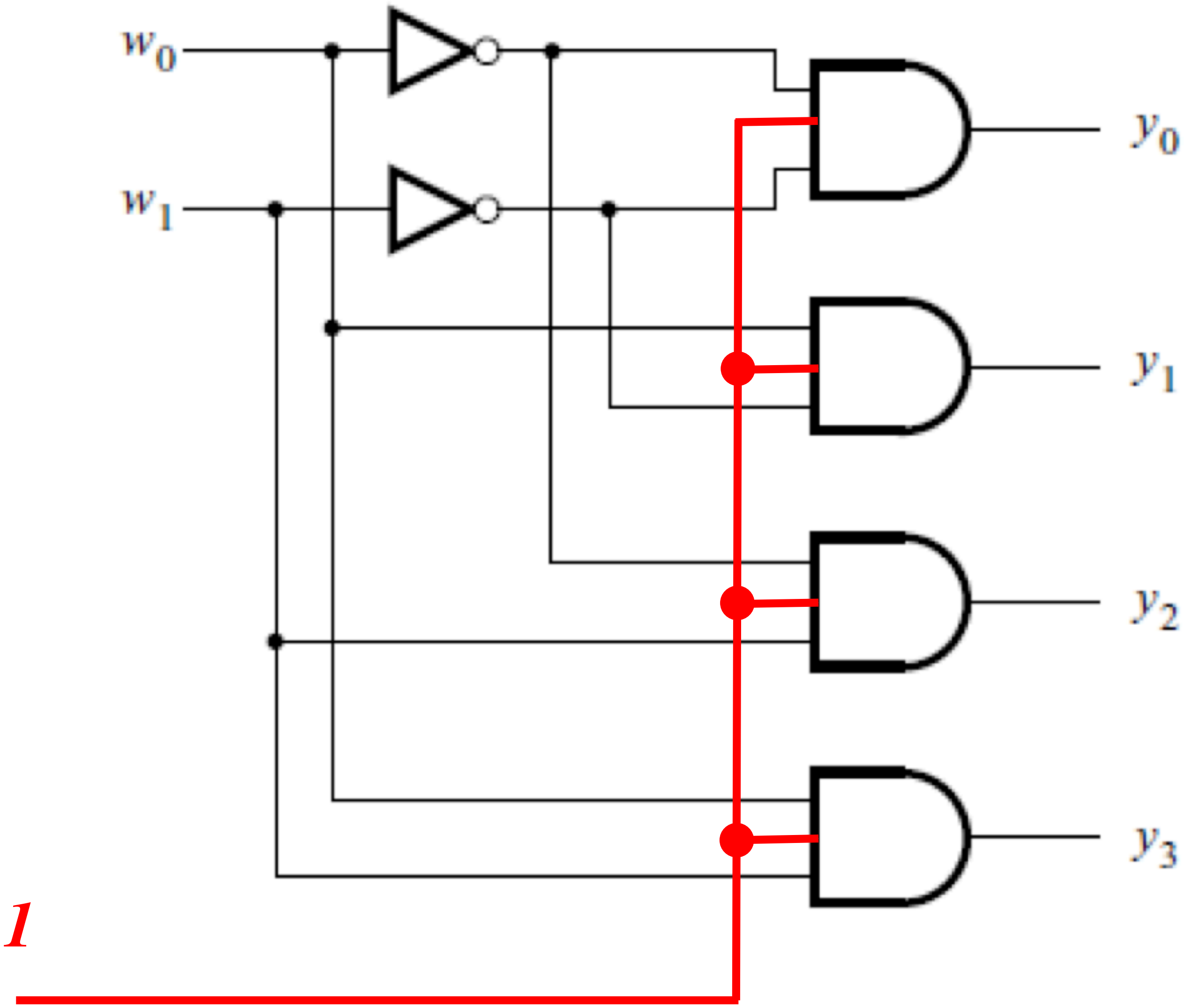
[Figure 5.28b from the textbook]

2-to-4 Decoder with Enable Input



[Figure 4.14c from the textbook]

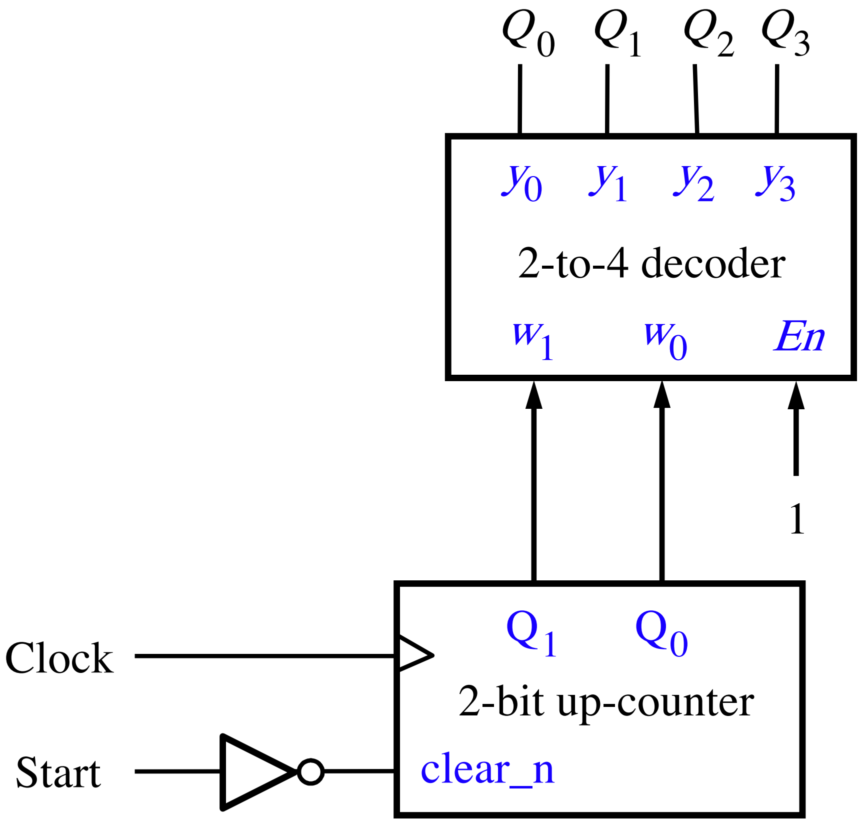
2-to-4 Decoder with Enable Input



(always enabled in this example)

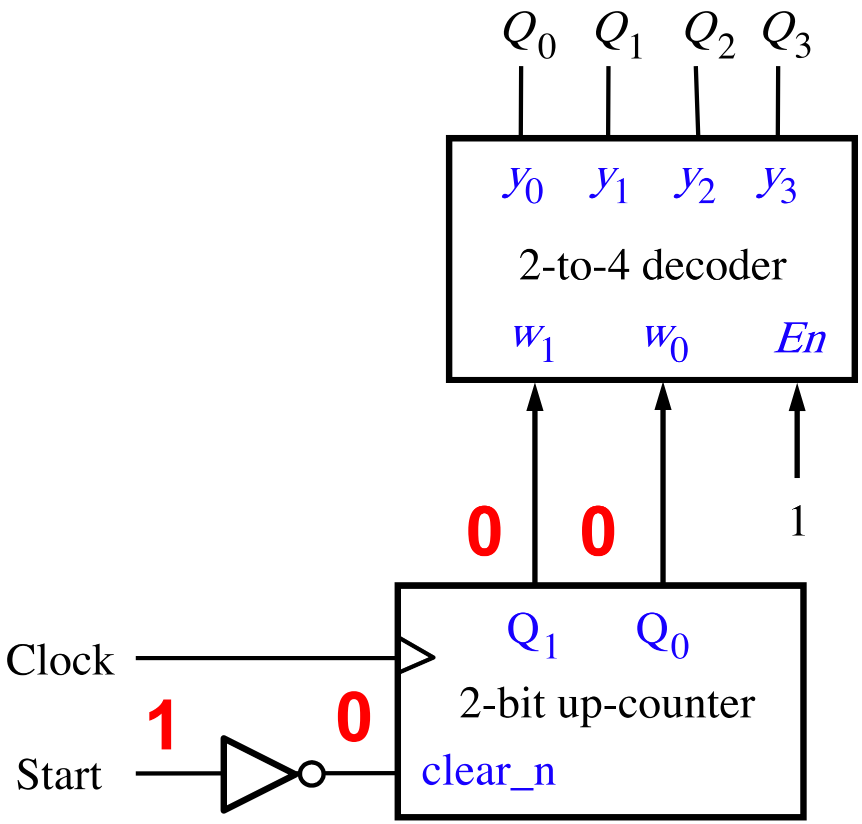
[Figure 4.14c from the textbook]

How Does It Work?



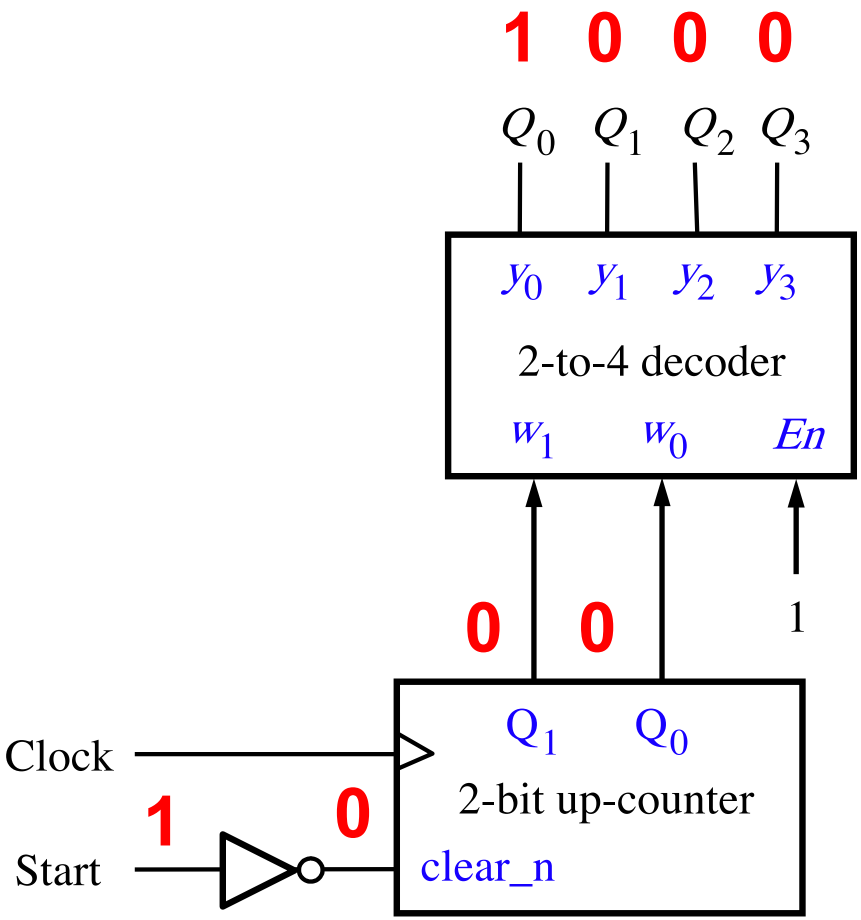
[Figure 5.28b from the textbook]

How Does It Work?



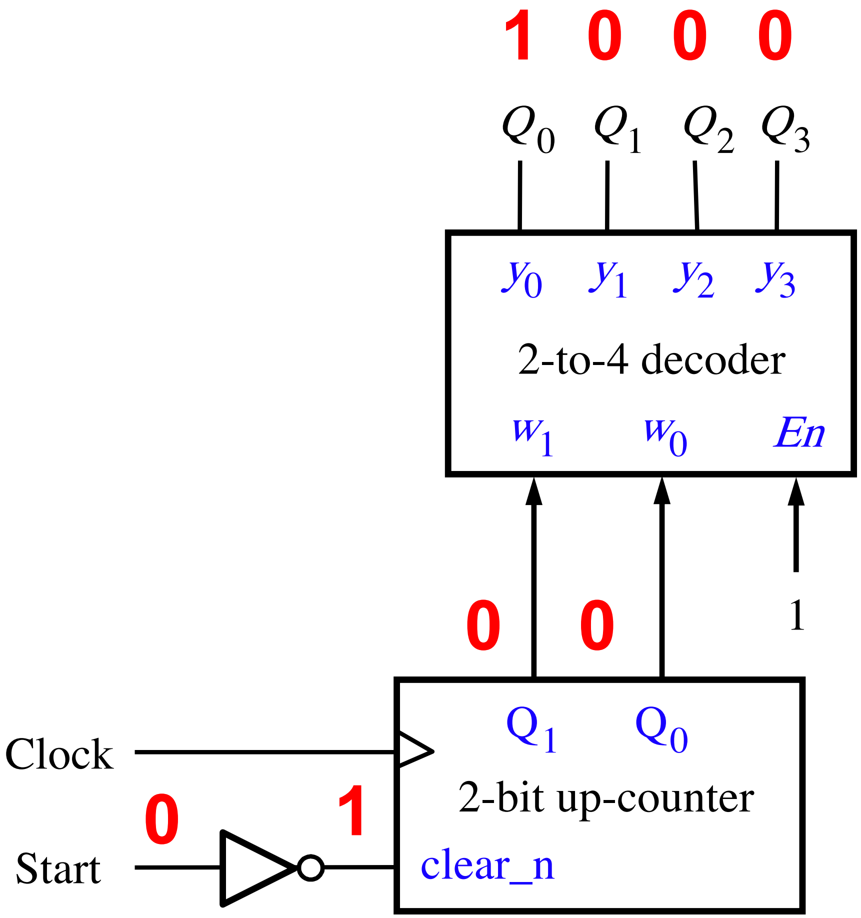
To initialize this circuit set Start to 1, which sets the counter to 00.

How Does It Work?



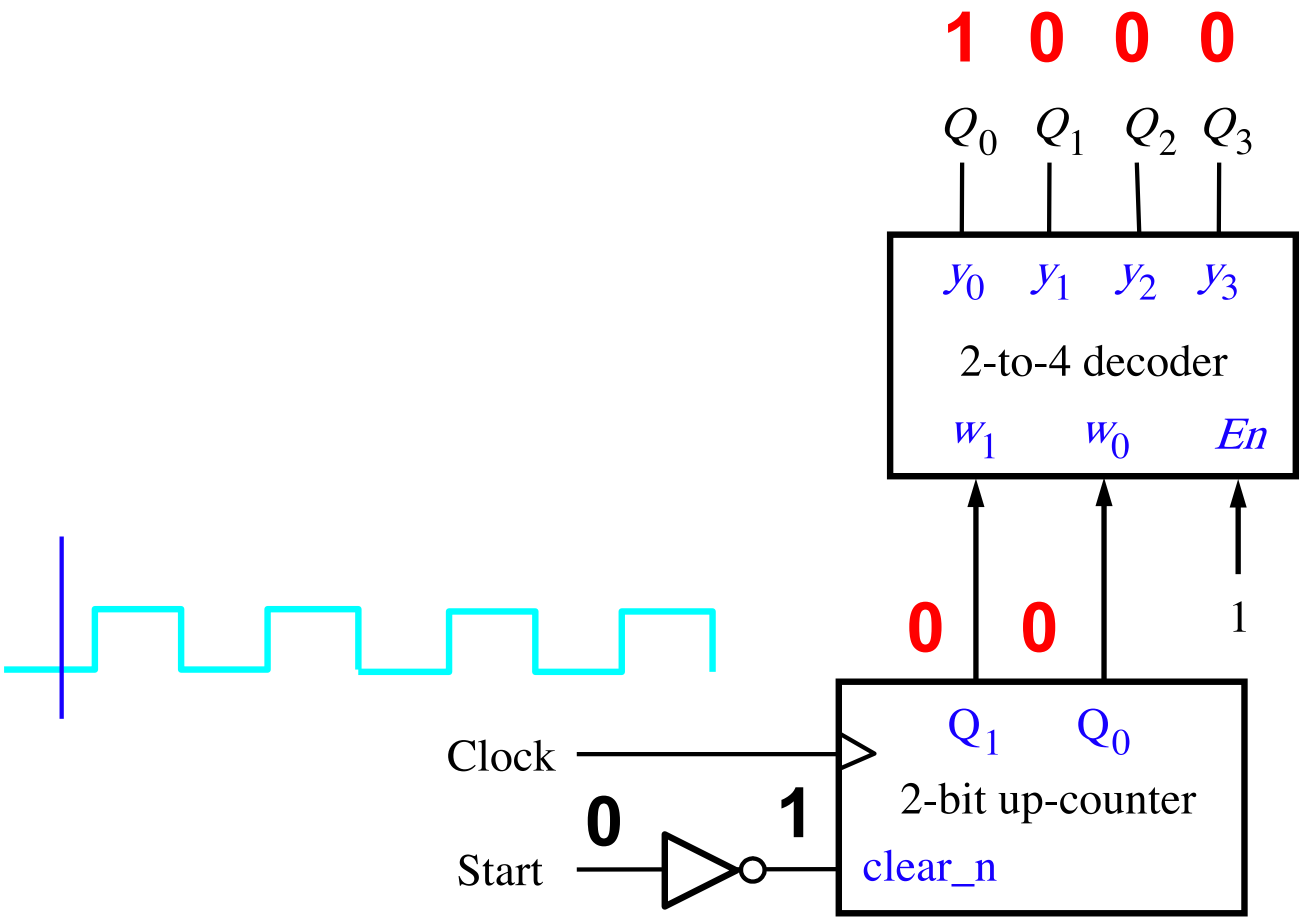
This sets the outputs of the decoder to the start of the counting sequence.

How Does It Work?

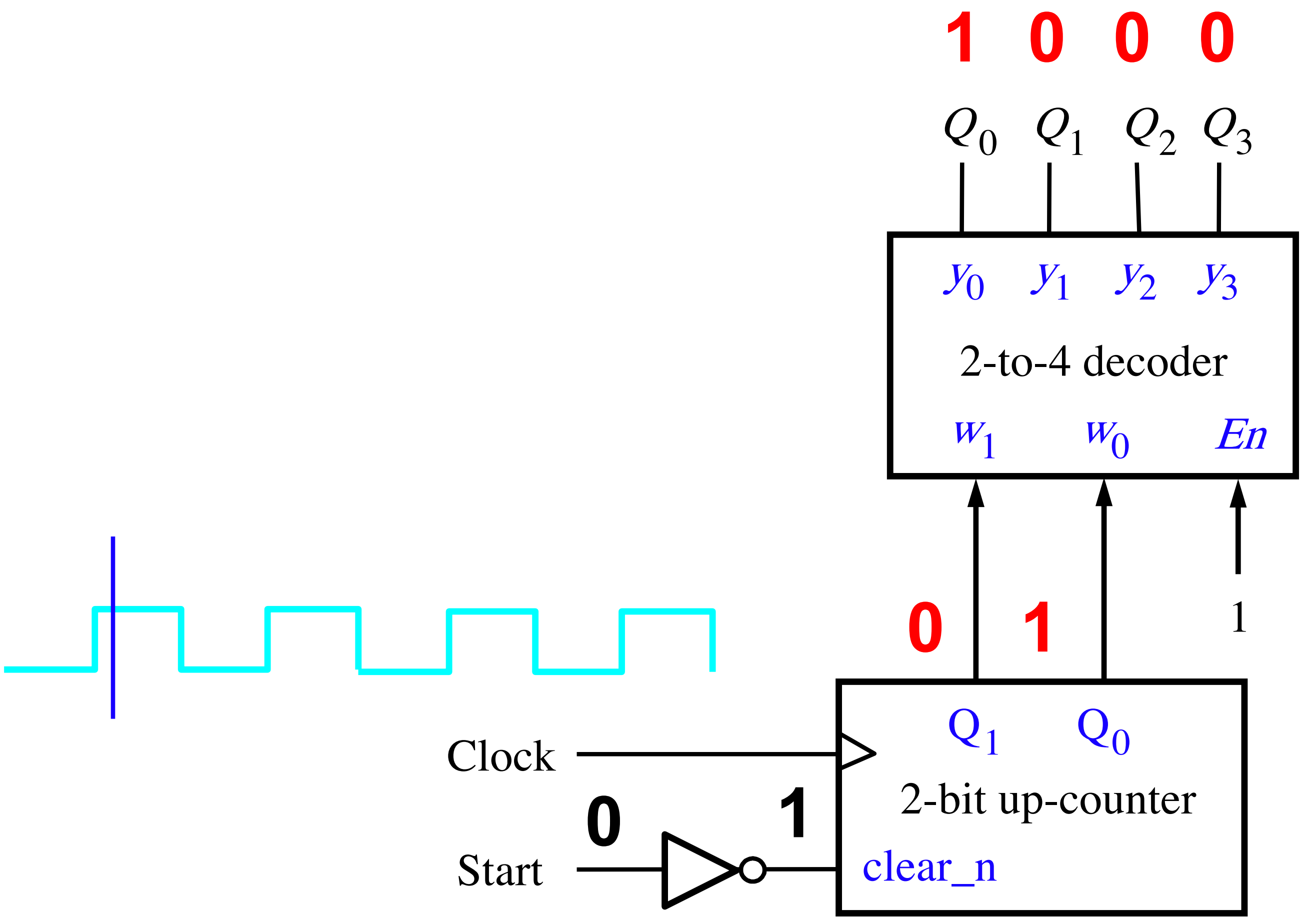


When Start is equal to 0, clear_n has no effect.

How Does It Work?

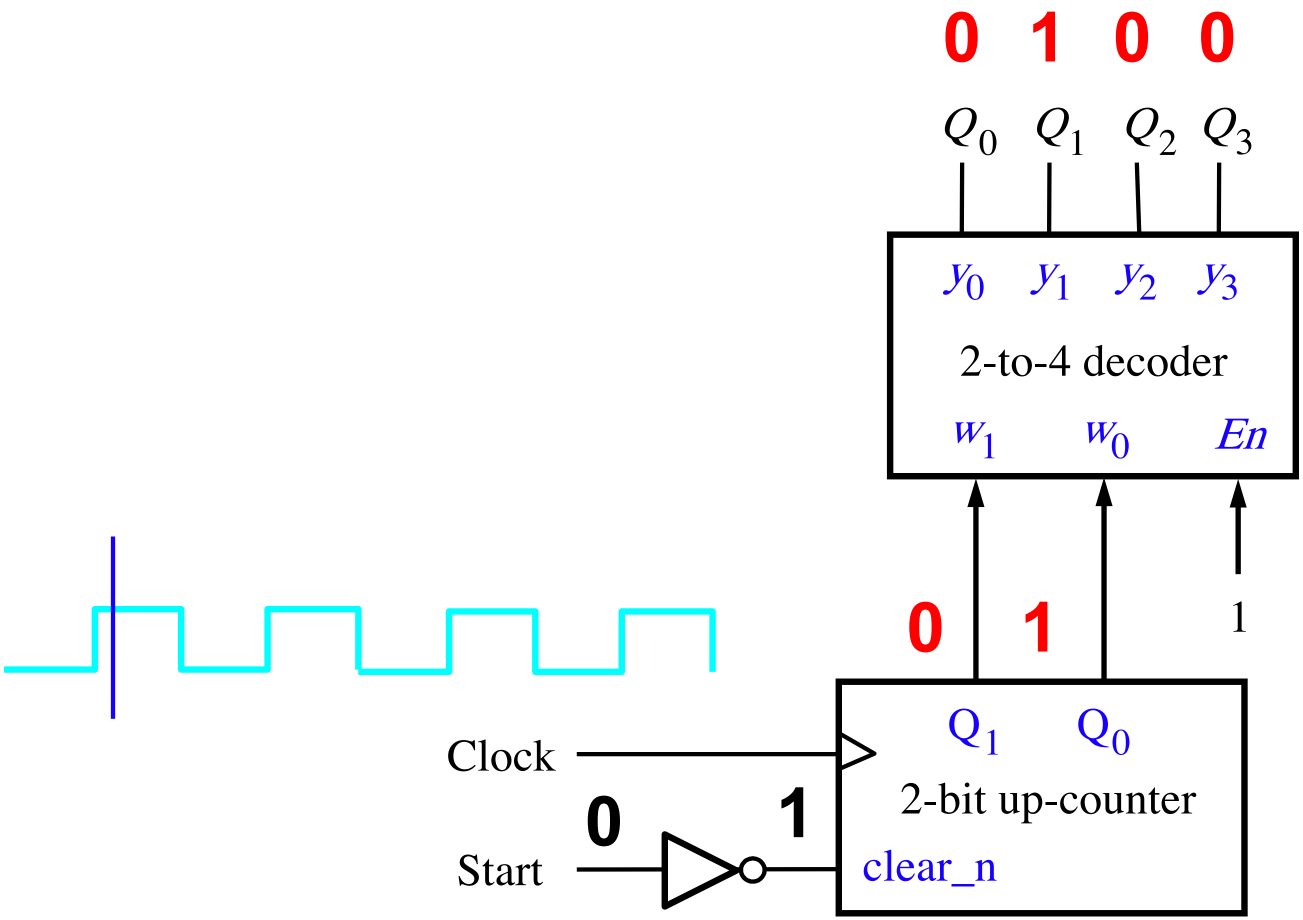


How Does It Work?



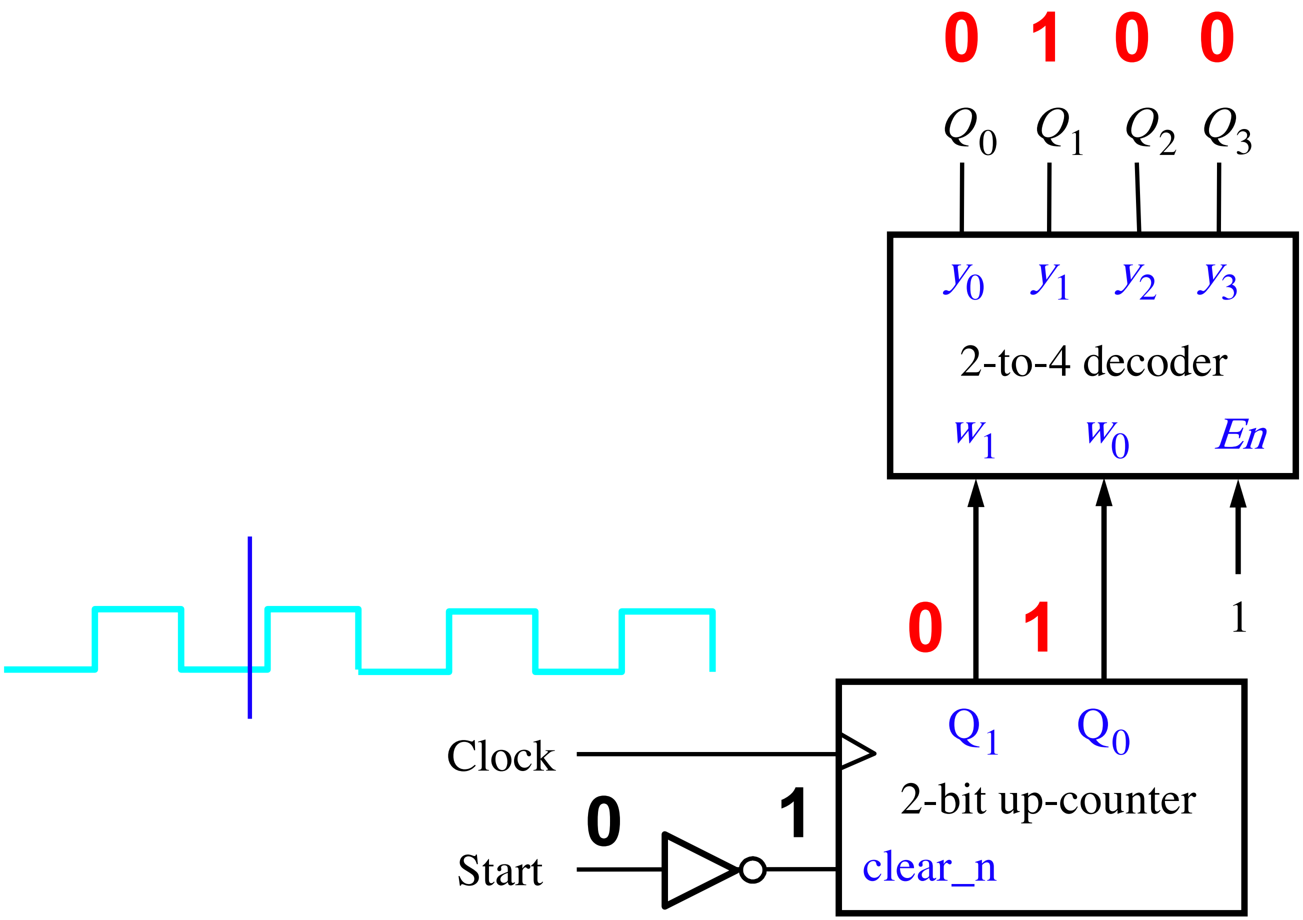
The counter increments the count on the positive clock edge ...

How Does It Work?

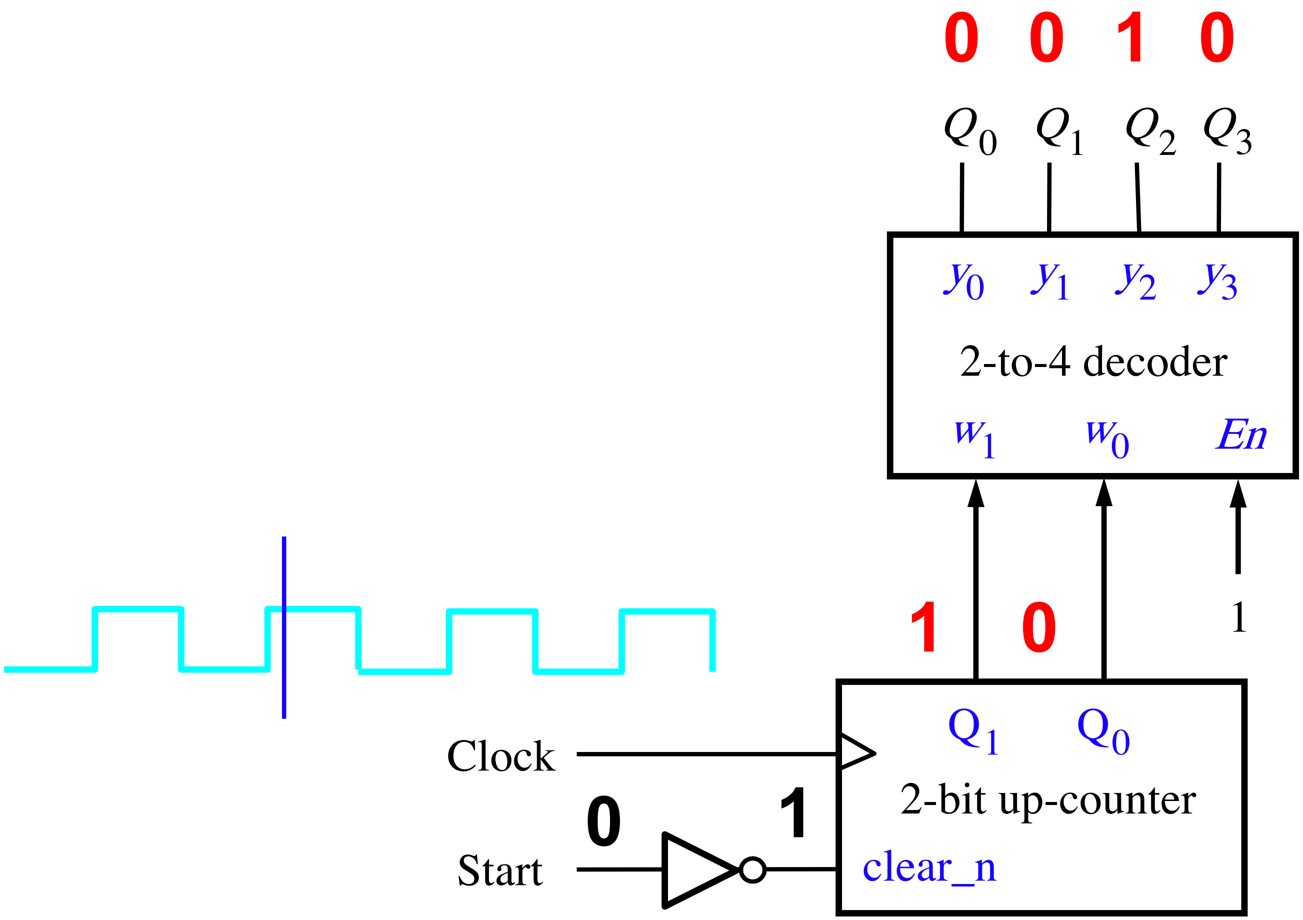


… this updates the outputs of the decoder (which are one hot encoded).

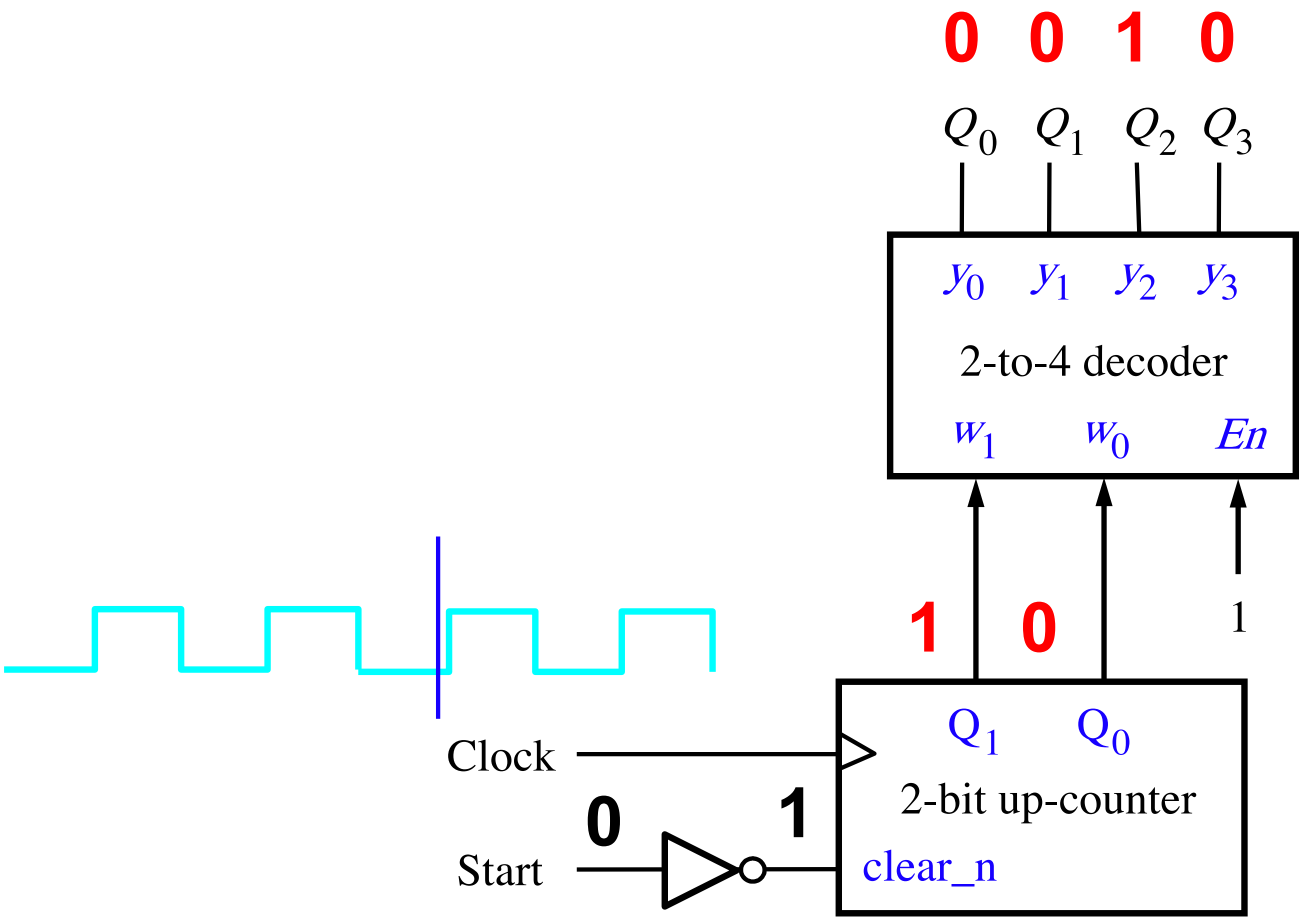
How Does It Work?



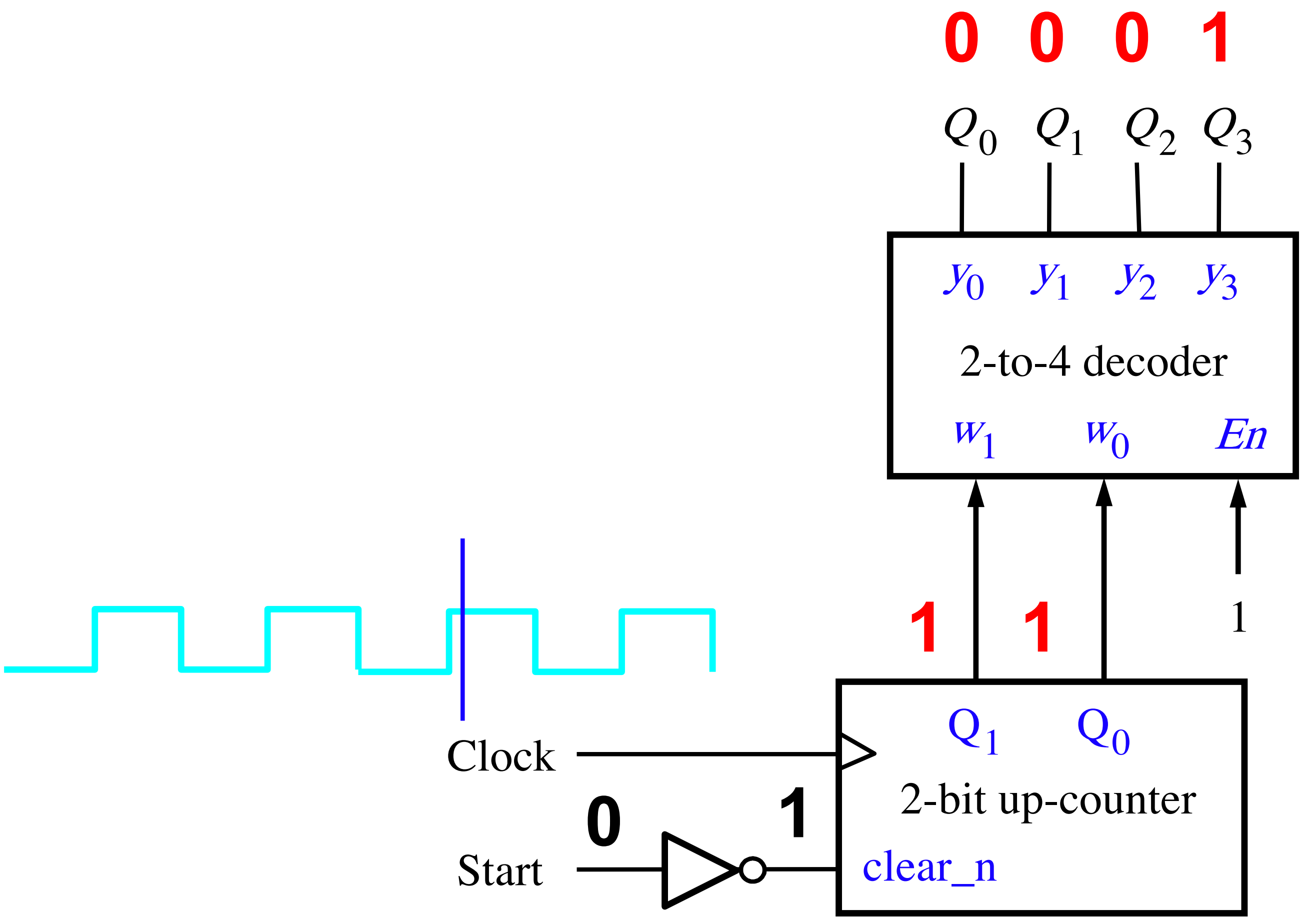
How Does It Work?



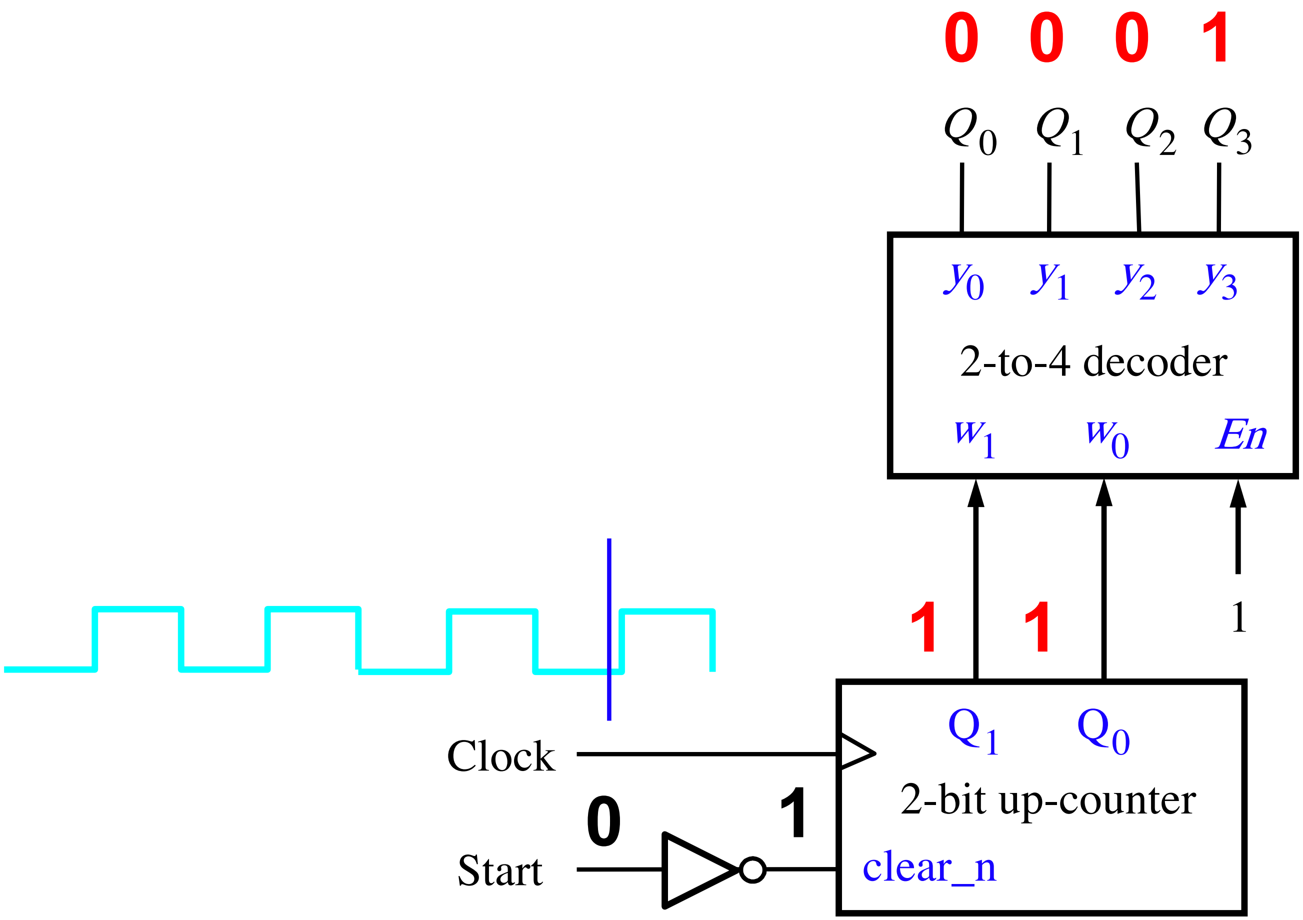
How Does It Work?



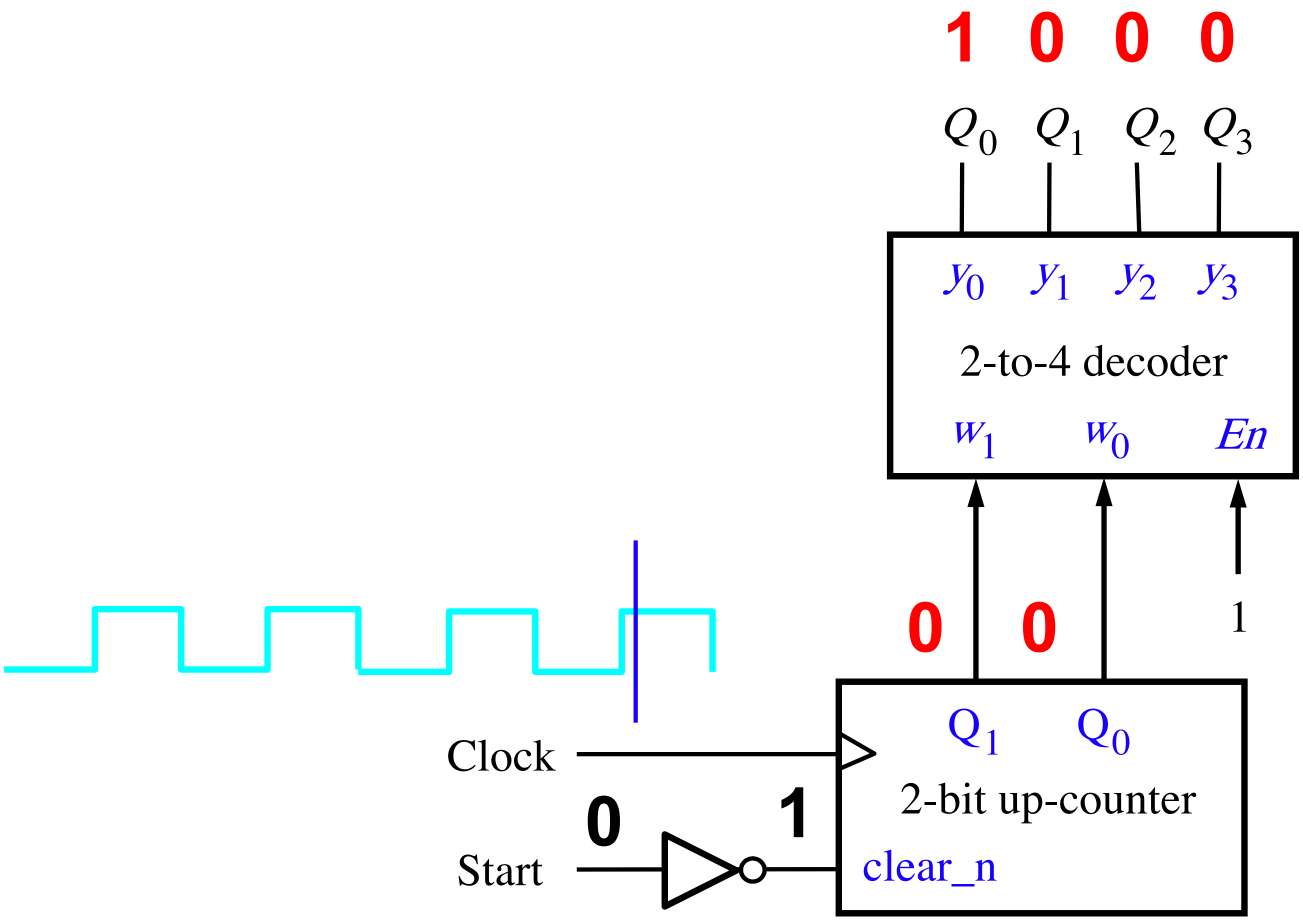
How Does It Work?



How Does It Work?



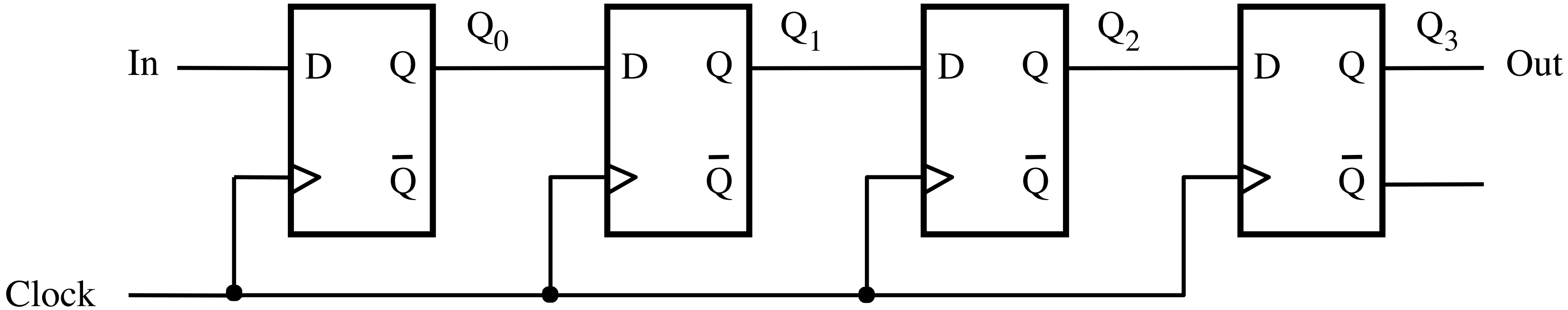
How Does It Work?



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

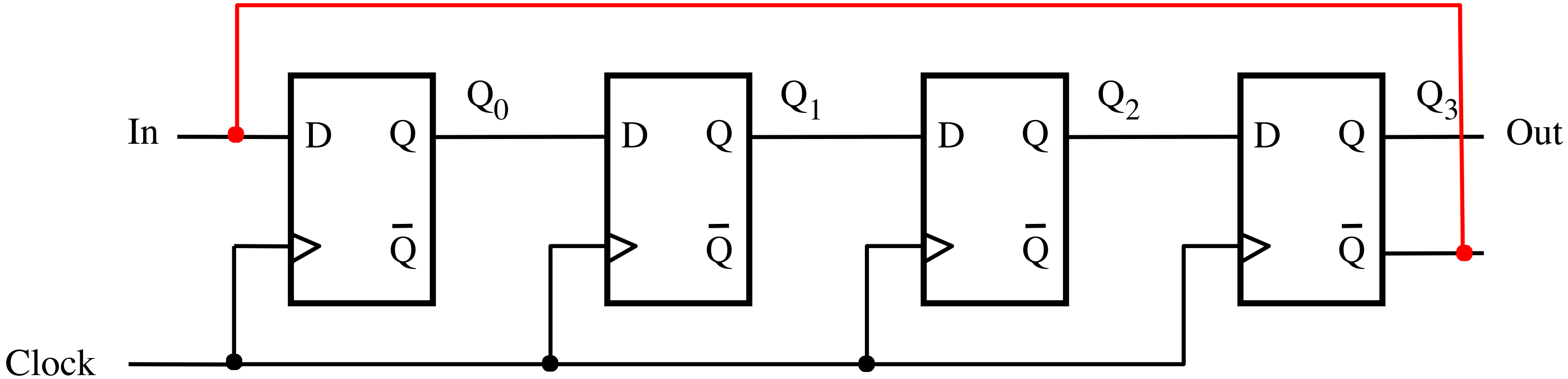
Johnson Counter

How to build a 4-bit Johnson counter



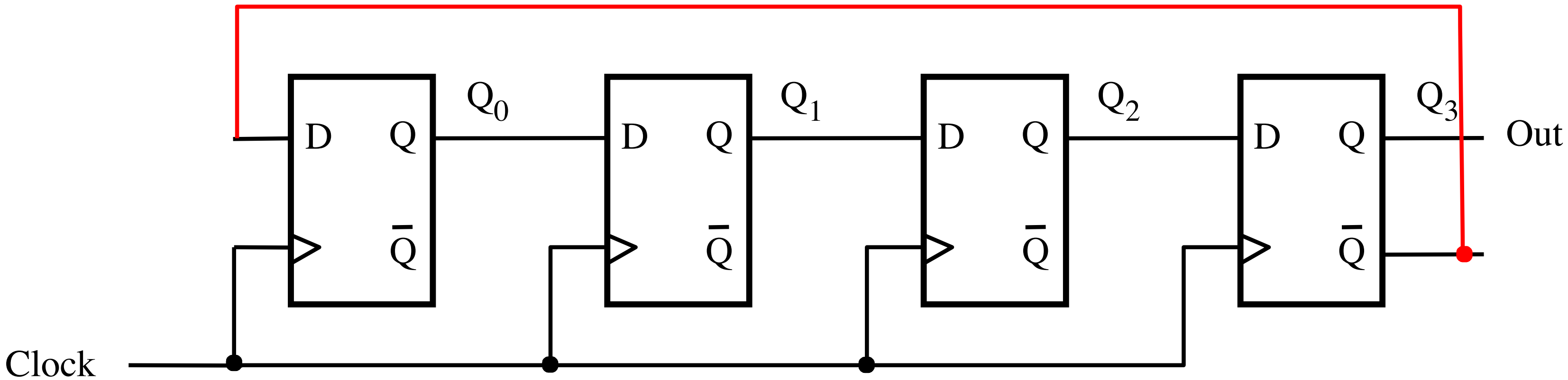
To build a Johnson counter start with a shift register.

How to build a 4-bit Johnson counter



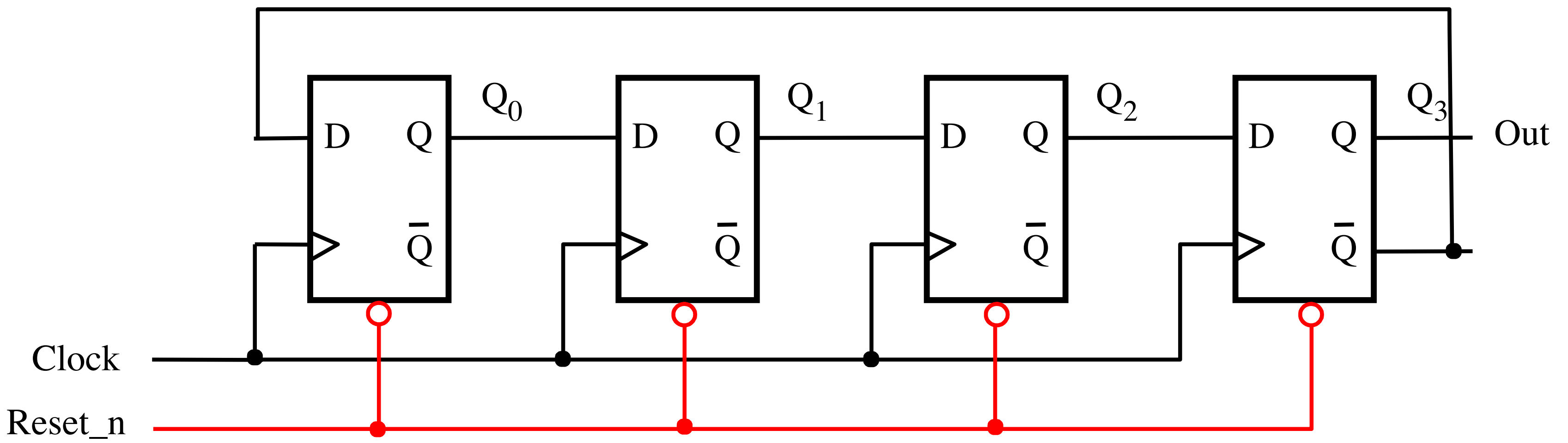
Next, add a loop from the $\overline{Q}$ output of the last flip-flop to the first…

How to build a 4-bit Johnson counter



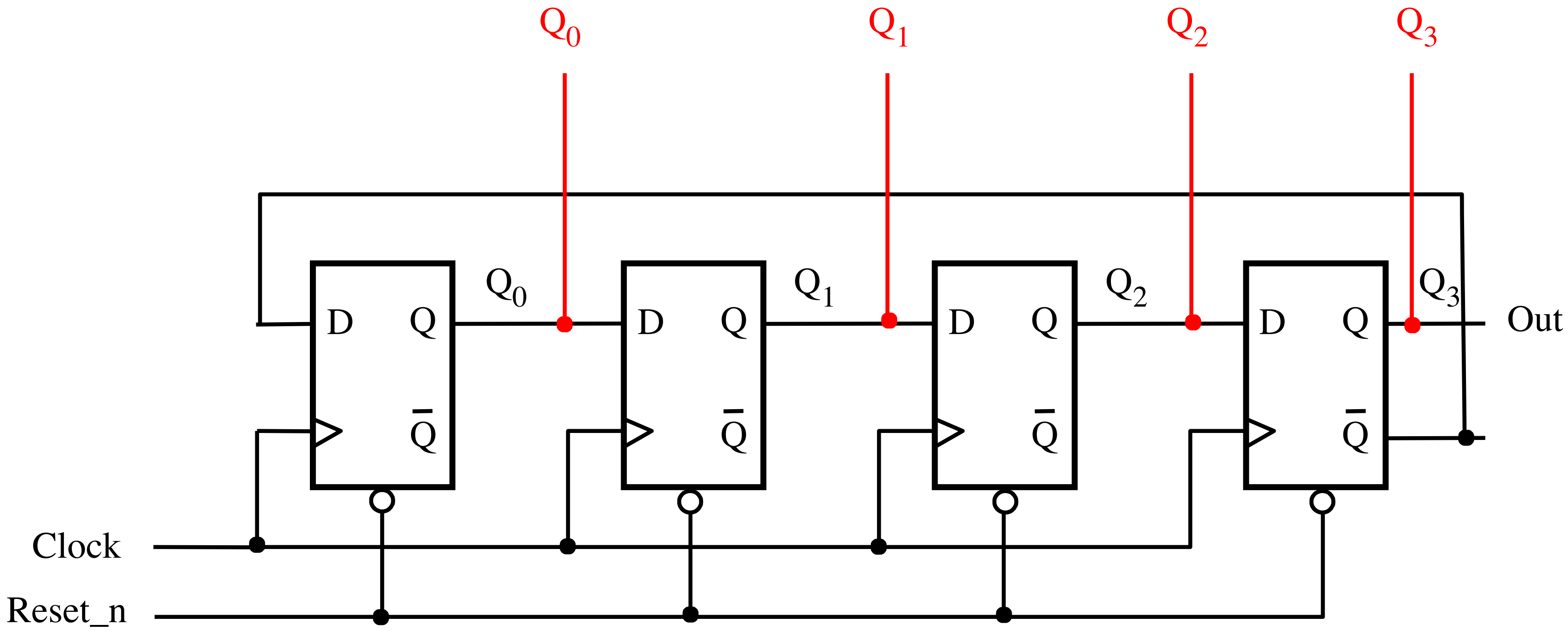
... and remove the In input line.

How to build a 4-bit Johnson counter



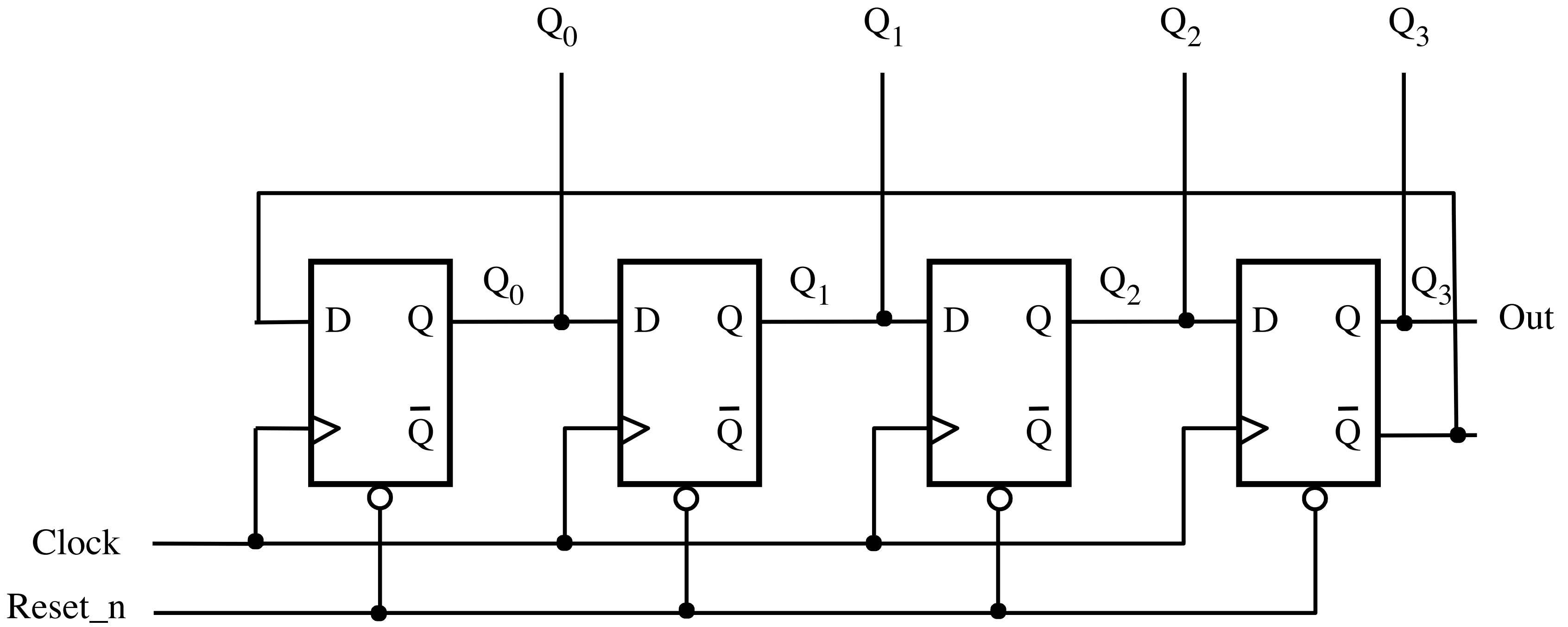
Also, add a Reset_n line that goes to clear_n of all flip-flops.

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

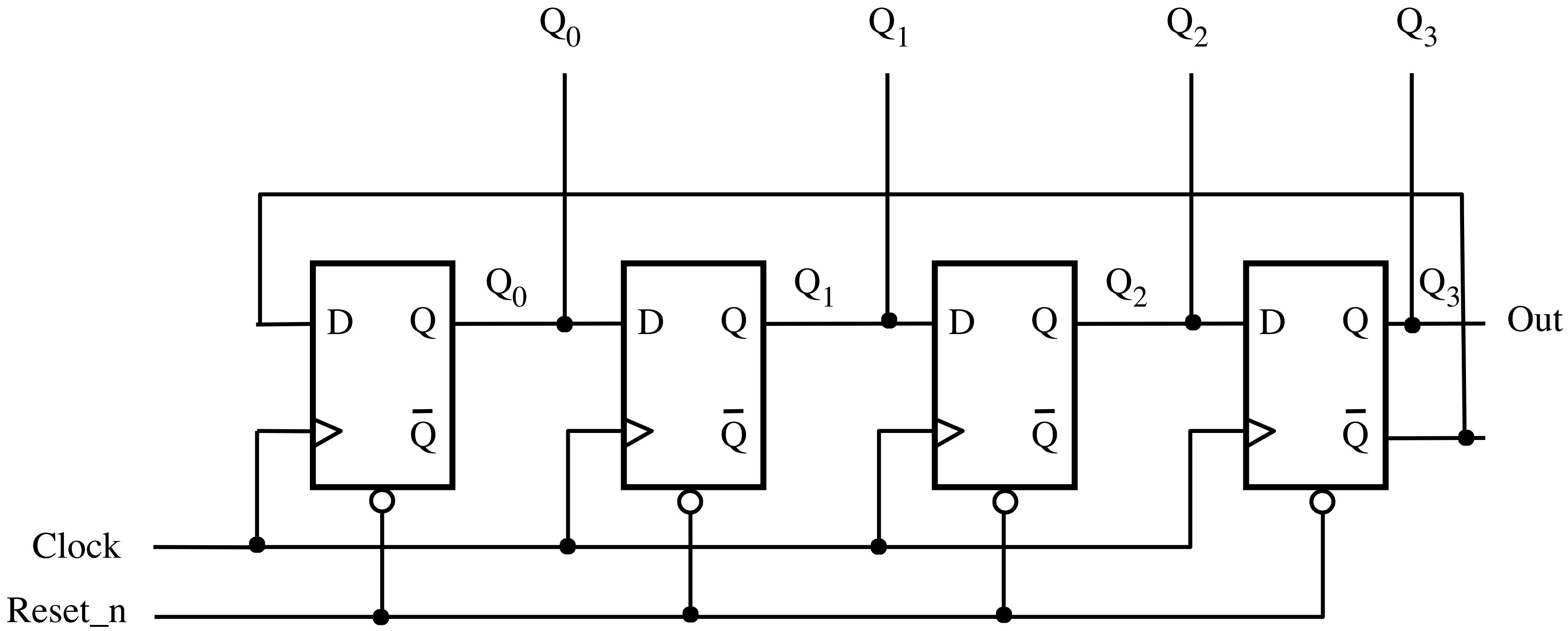


This is the final circuit diagram.

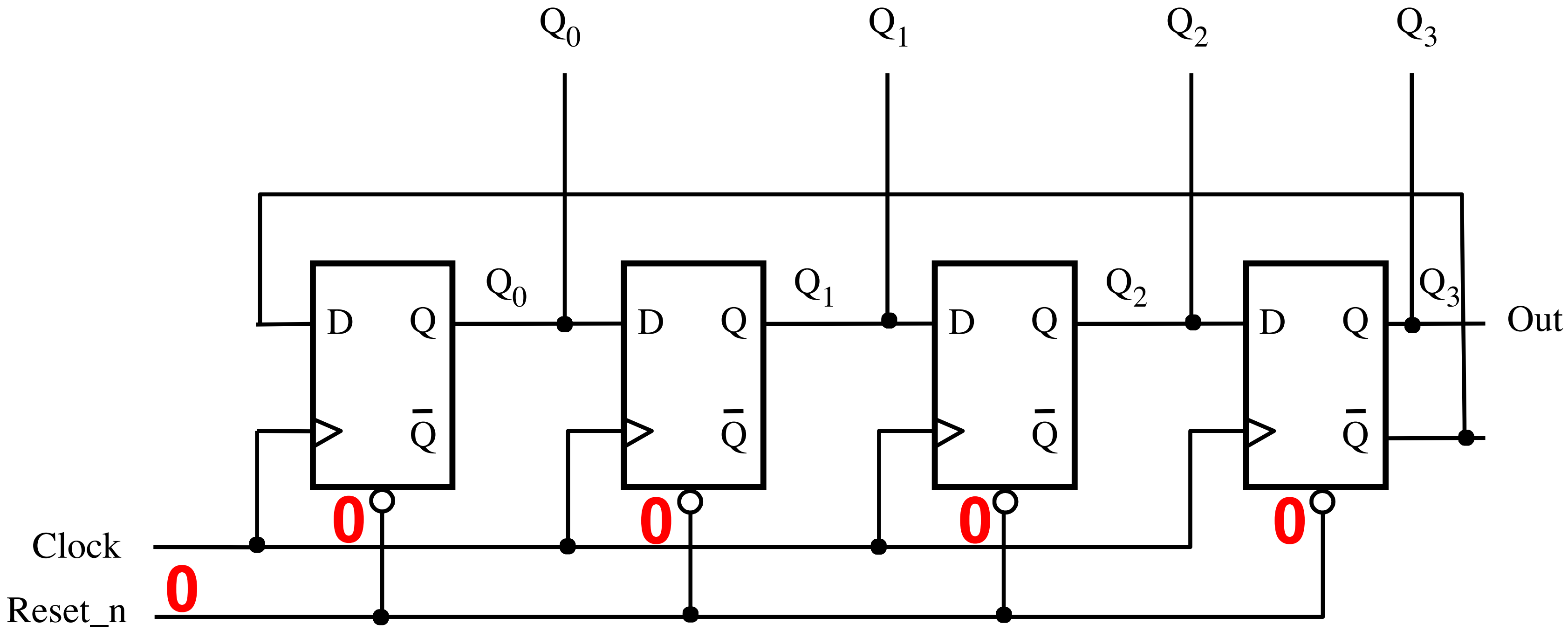
4-Bit Johnson Counter

- 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?

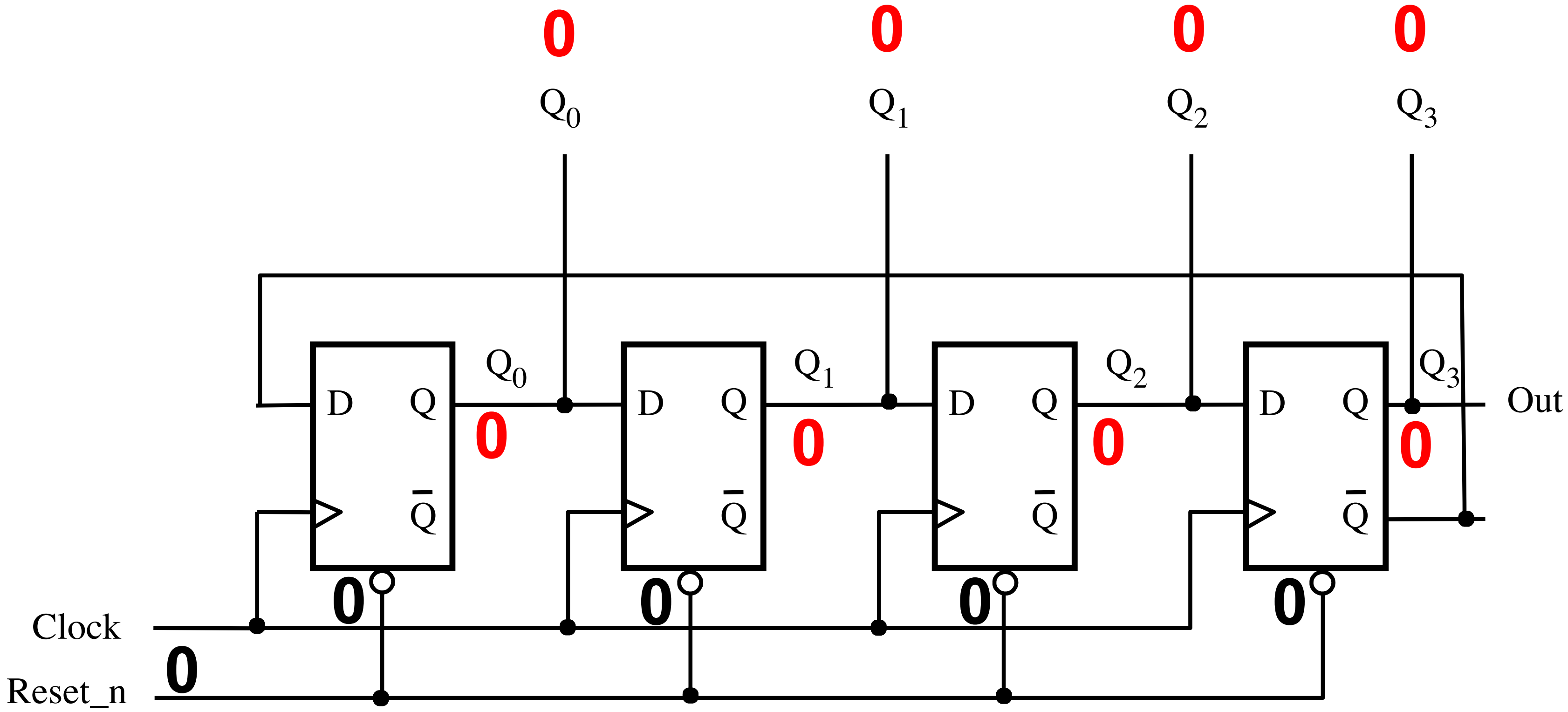


Initialization: How does it work?



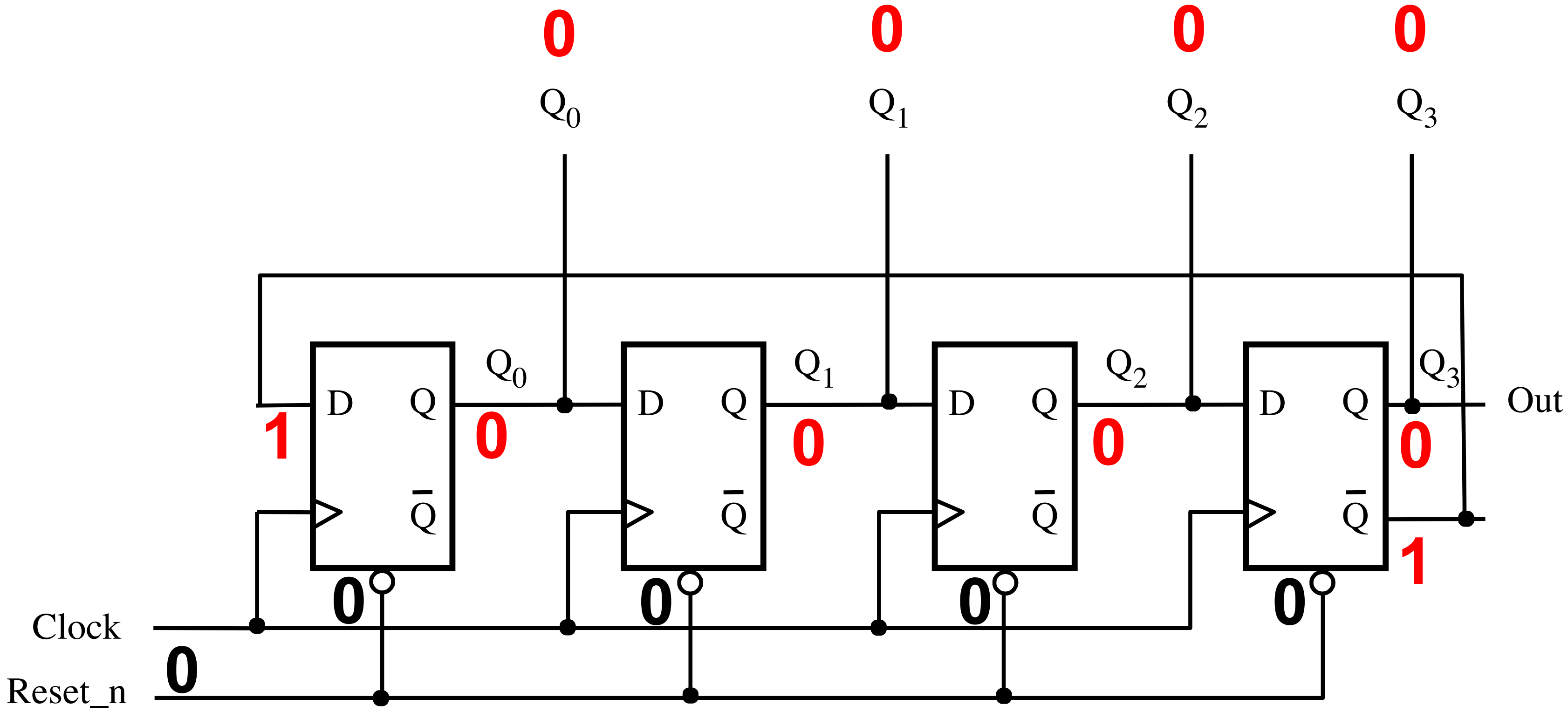
To initialize the Johnson counter set Reset_n to 0 ...

Initialization: How does it work?



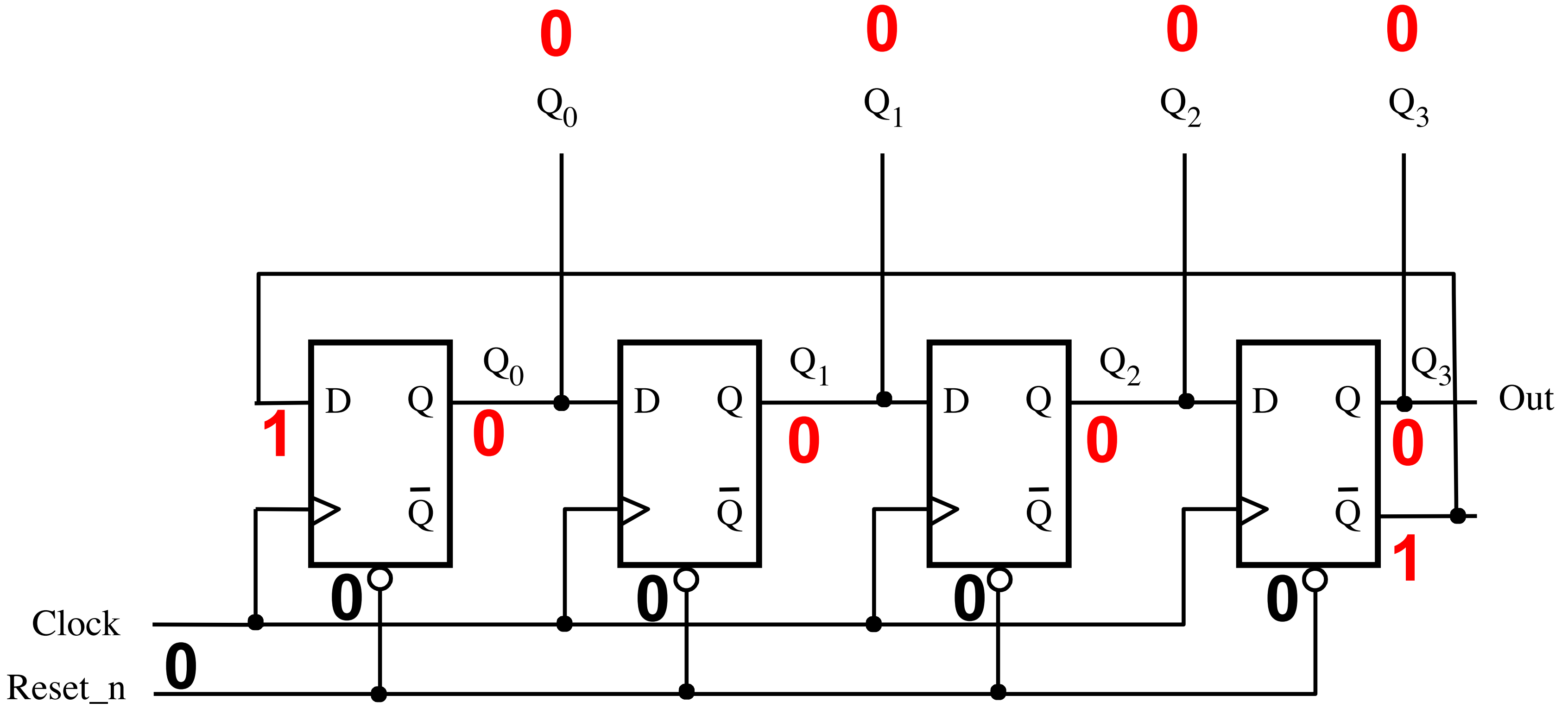
... this zeros the outputs of all flip-flops ...

Initialization: How does it work?

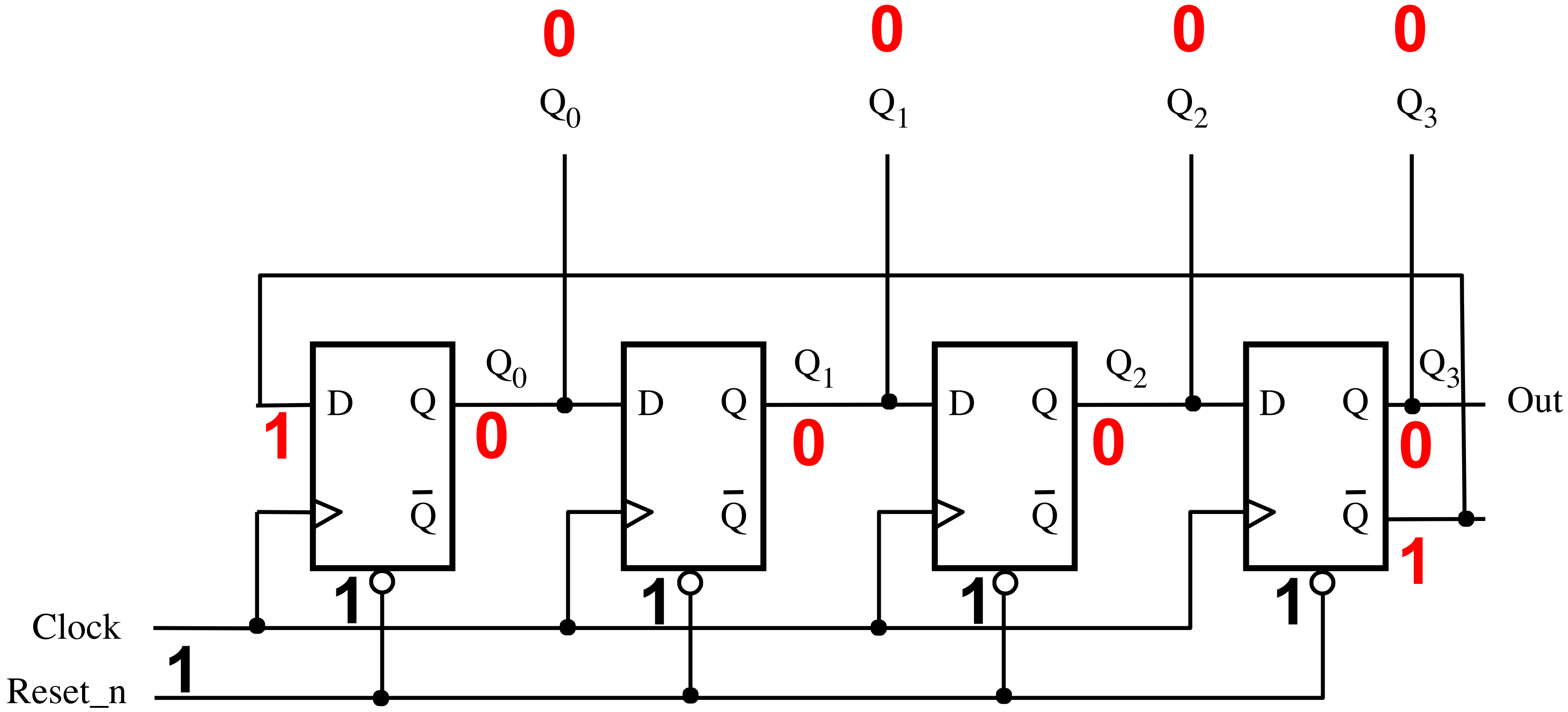


... and also sets the $\overline{Q}$ output of the last flip-flop to 1.

Counting: How does it work?

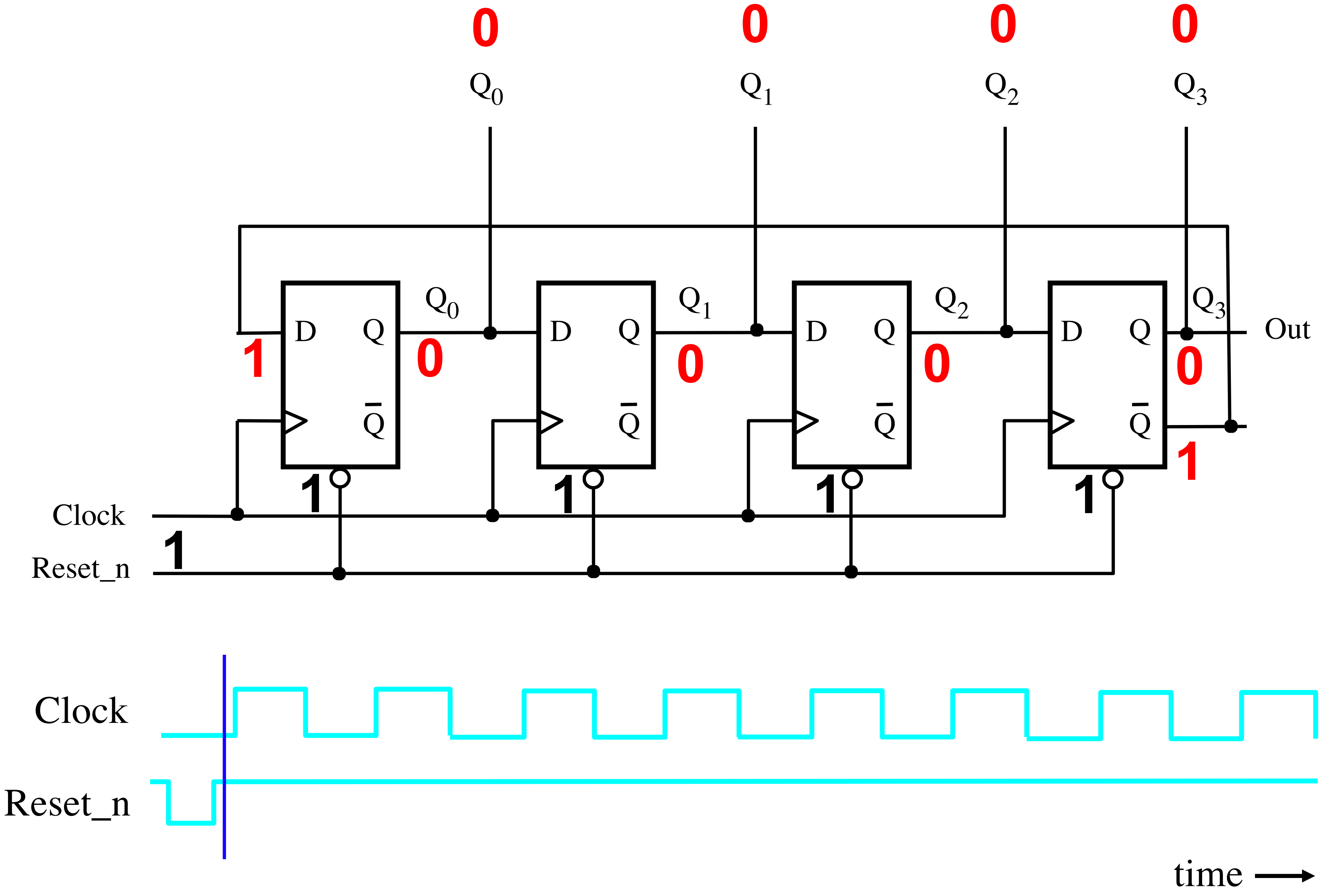


Counting: How does it work?

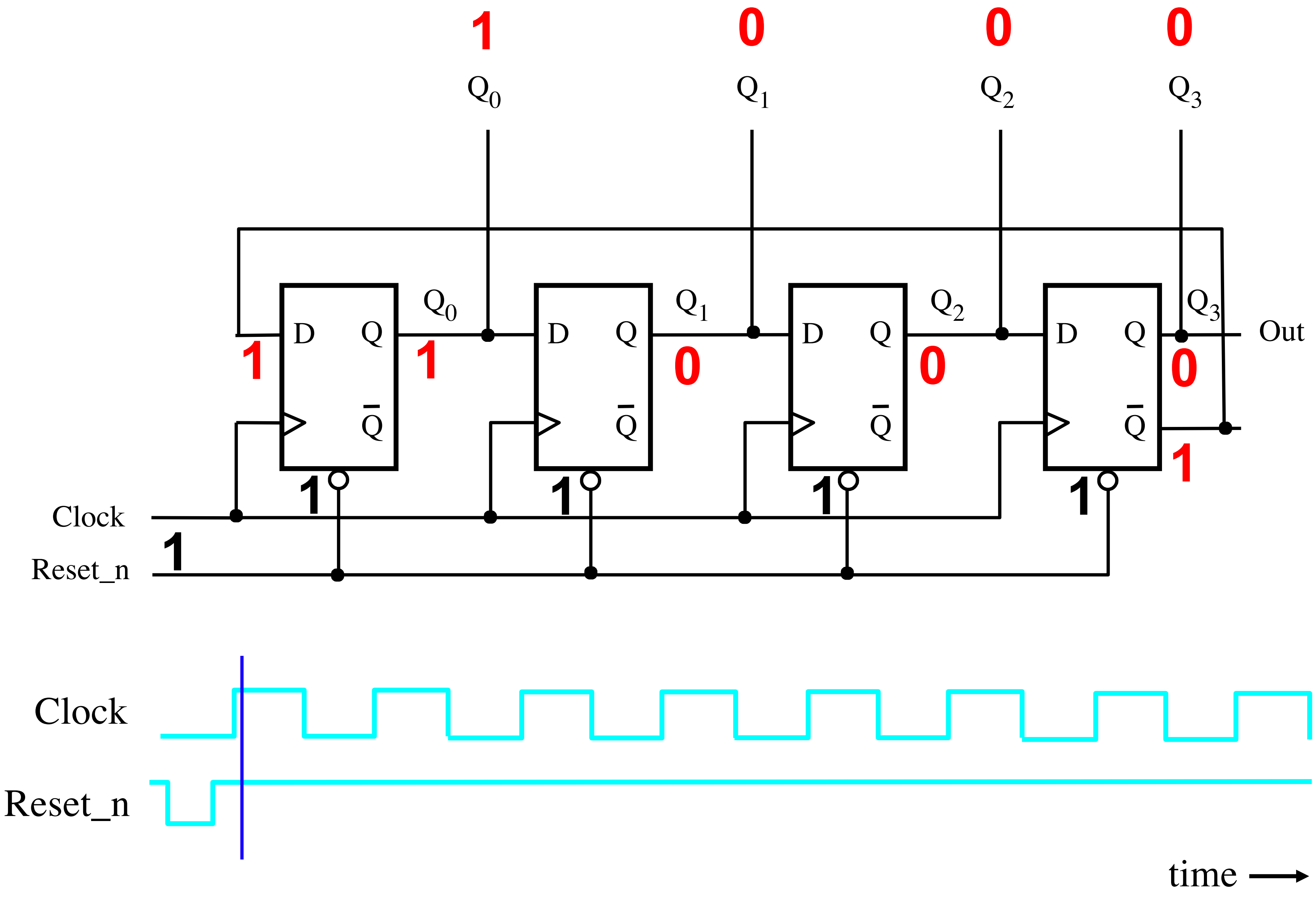


To start counting, Reset_n needs to be set to 1.

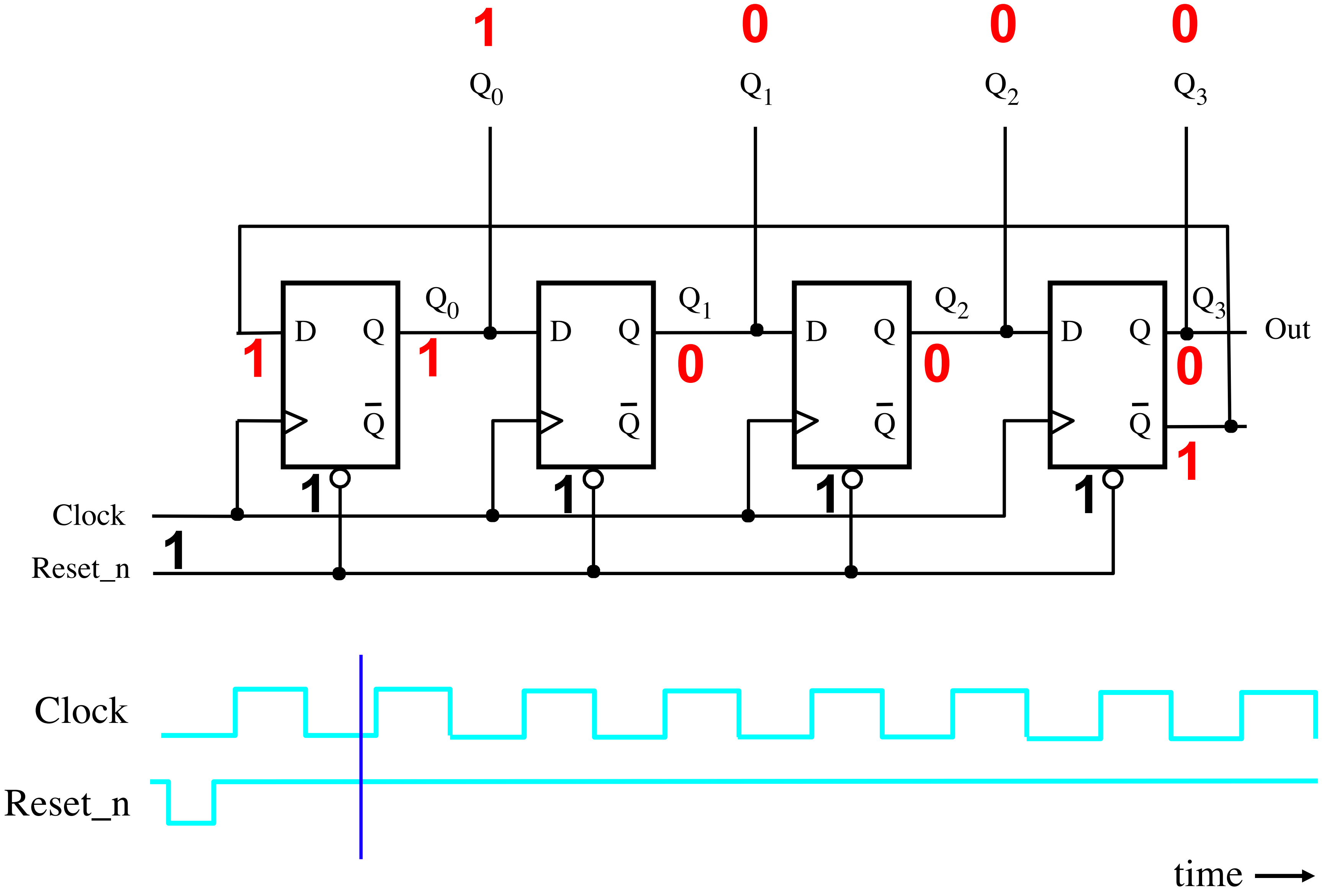
Counting: How does it work?



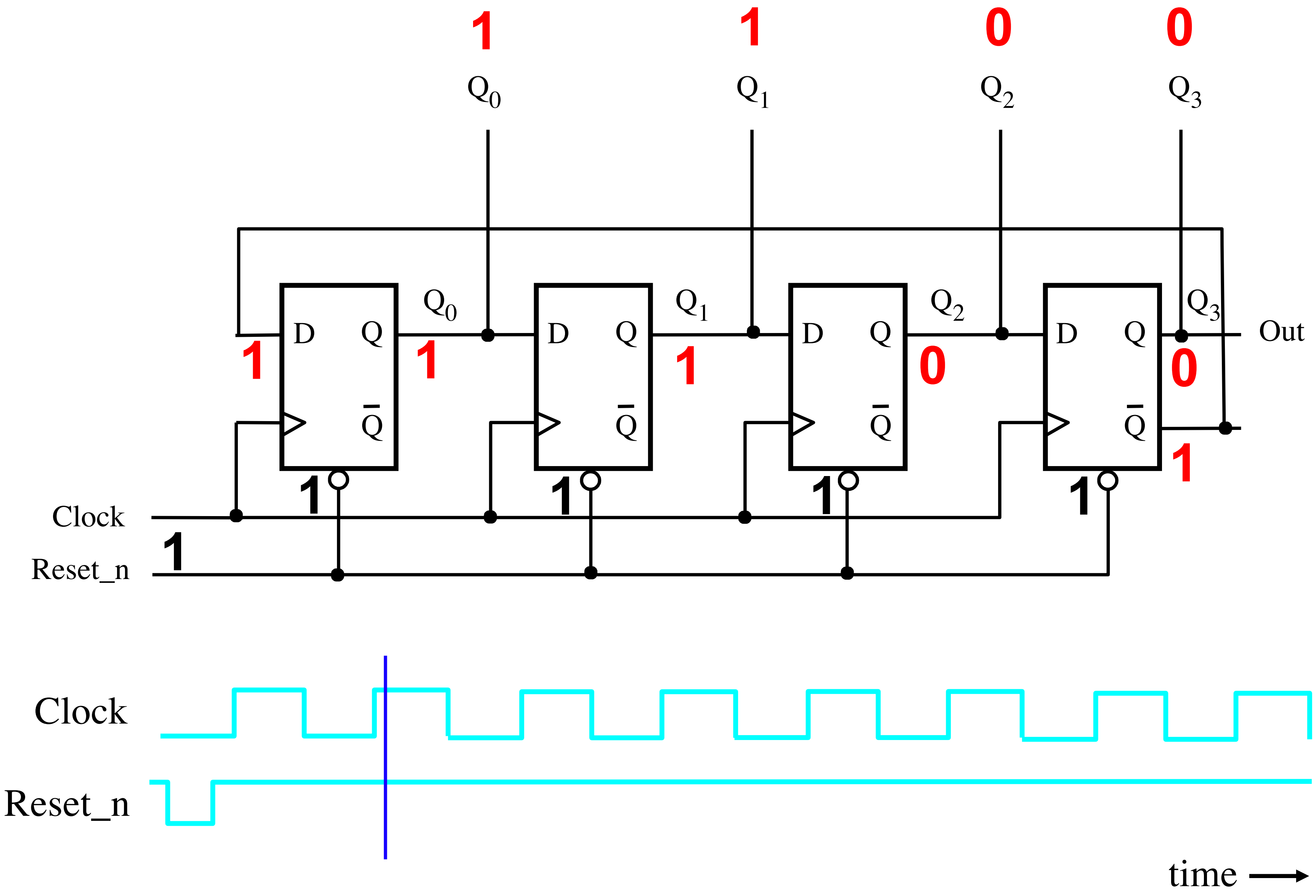
Counting: How does it work?



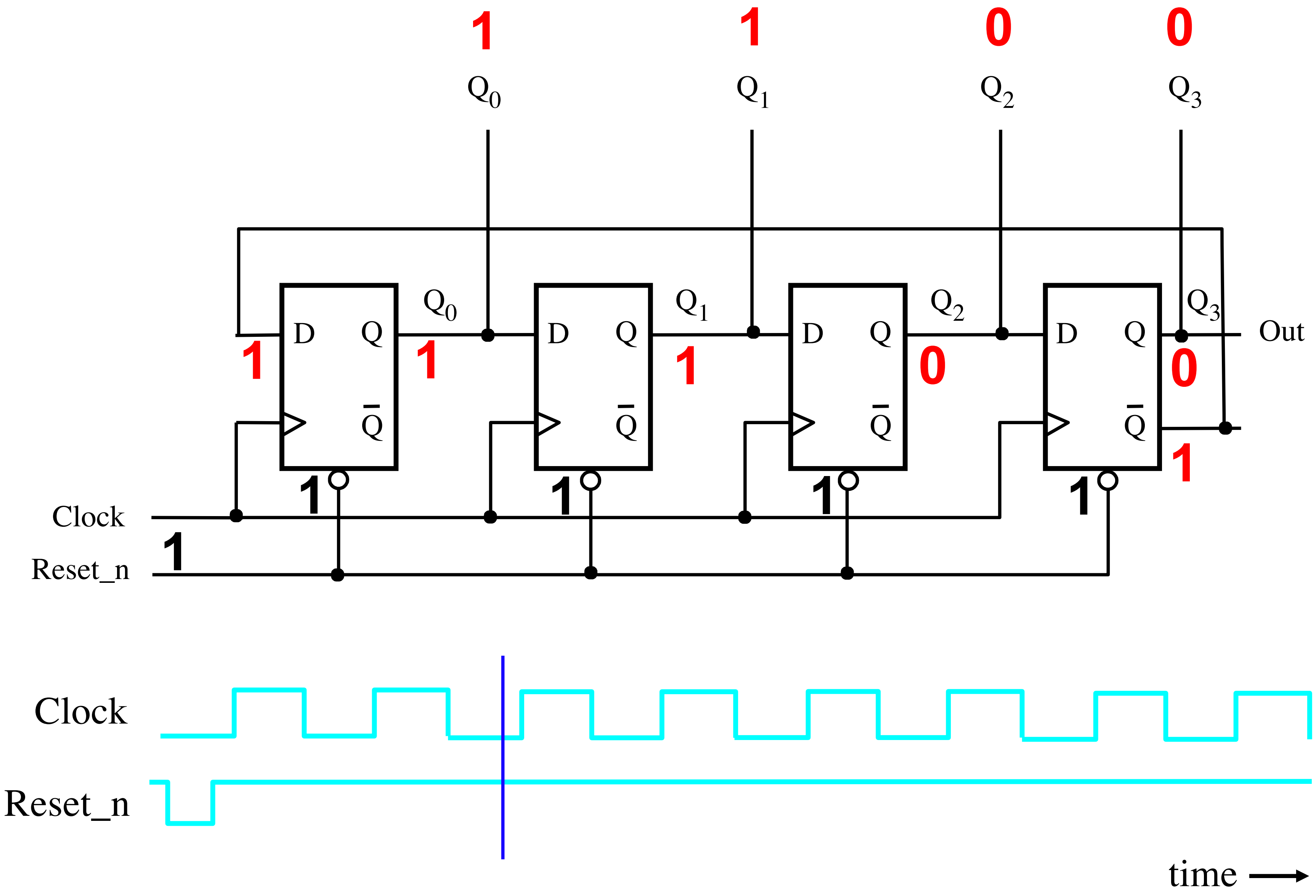
Counting: How does it work?



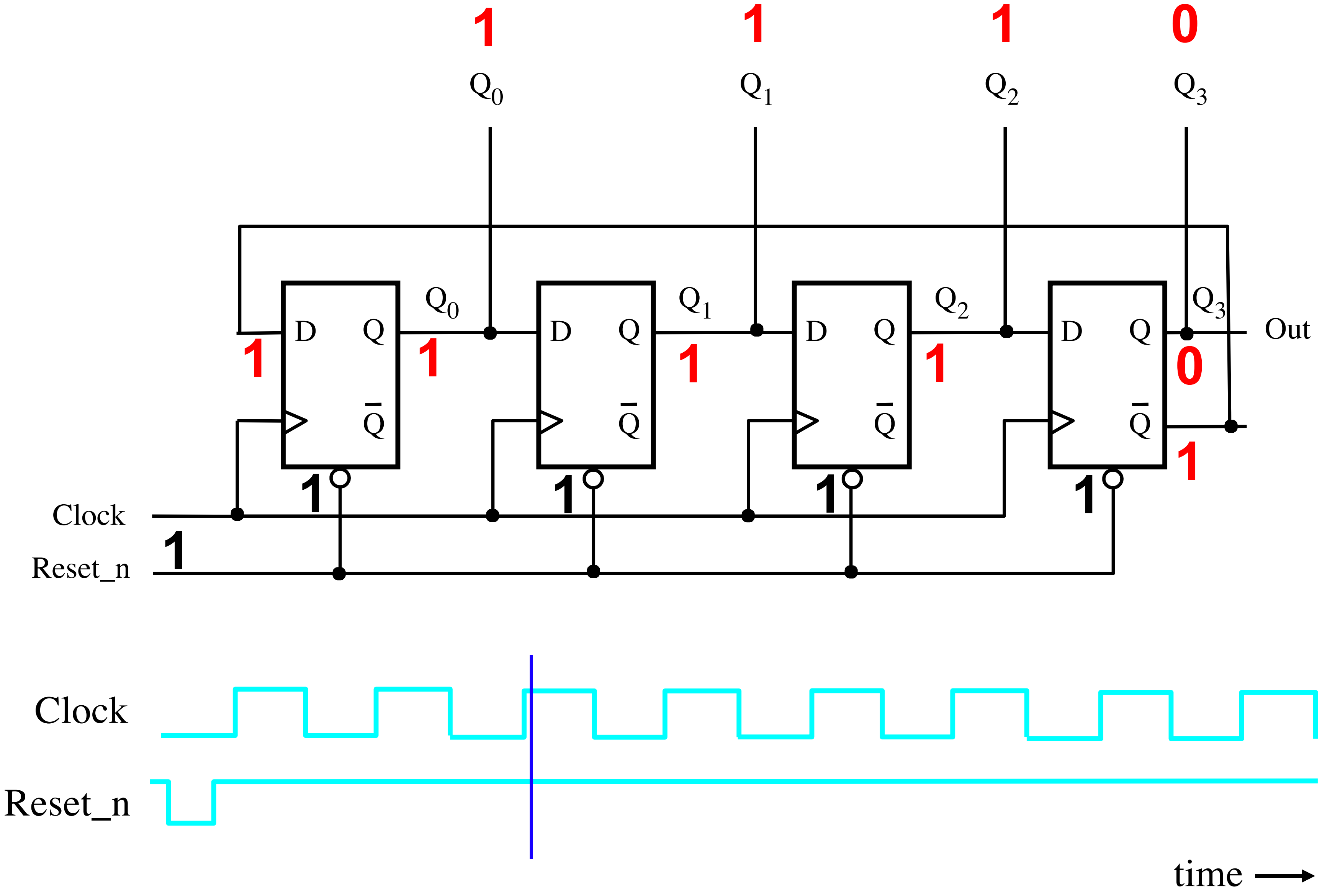
Counting: How does it work?



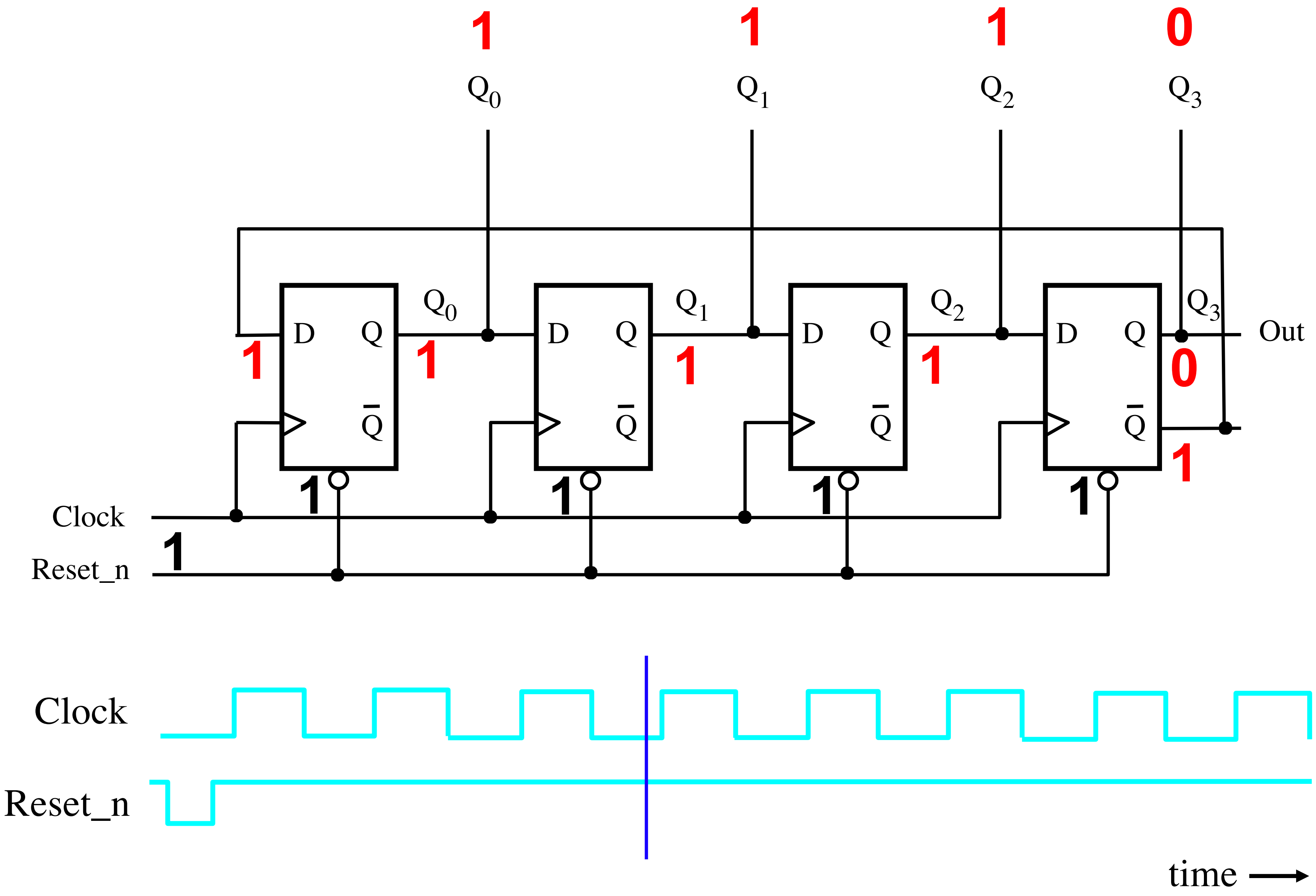
Counting: How does it work?



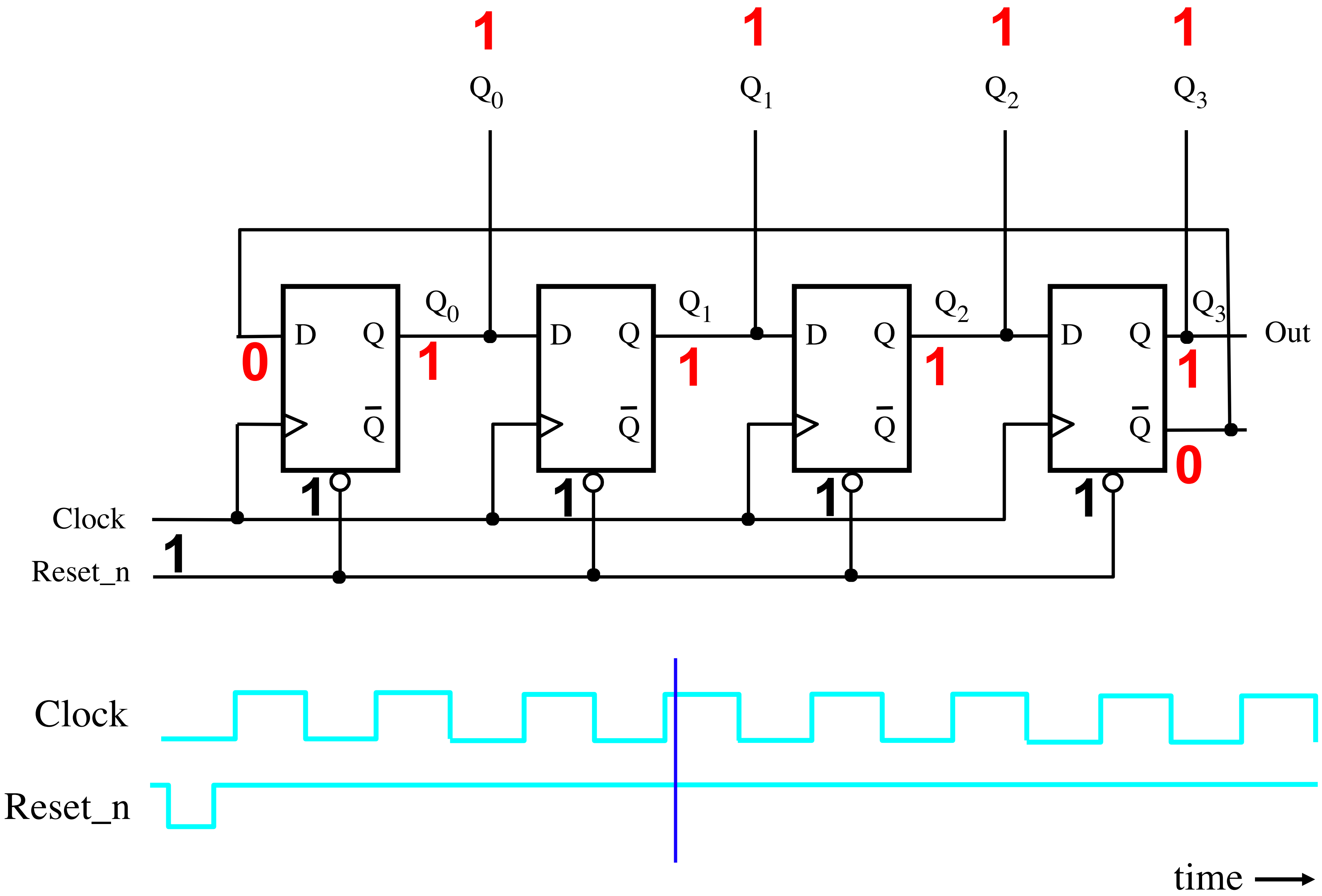
Counting: How does it work?



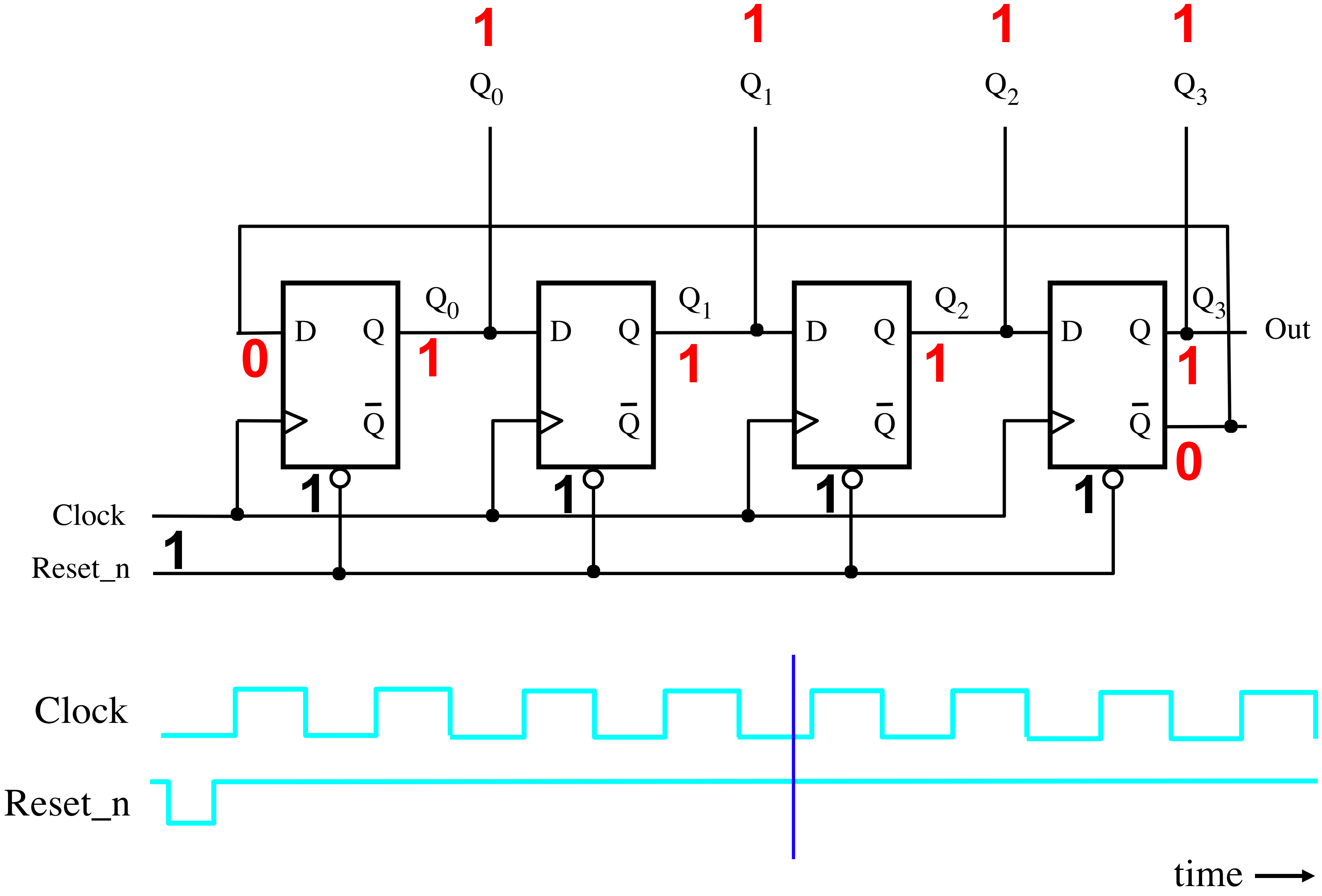
Counting: How does it work?



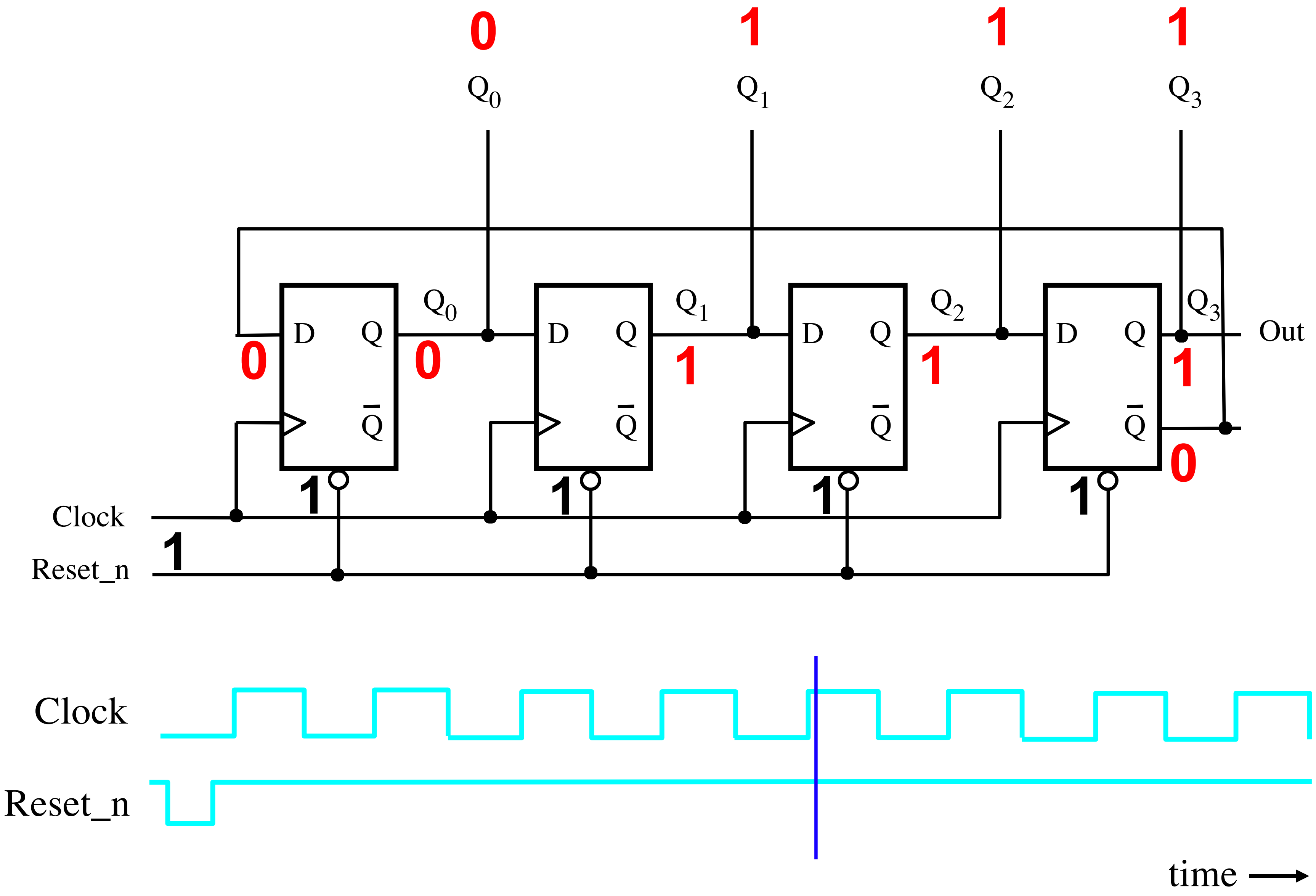
Counting: How does it work?



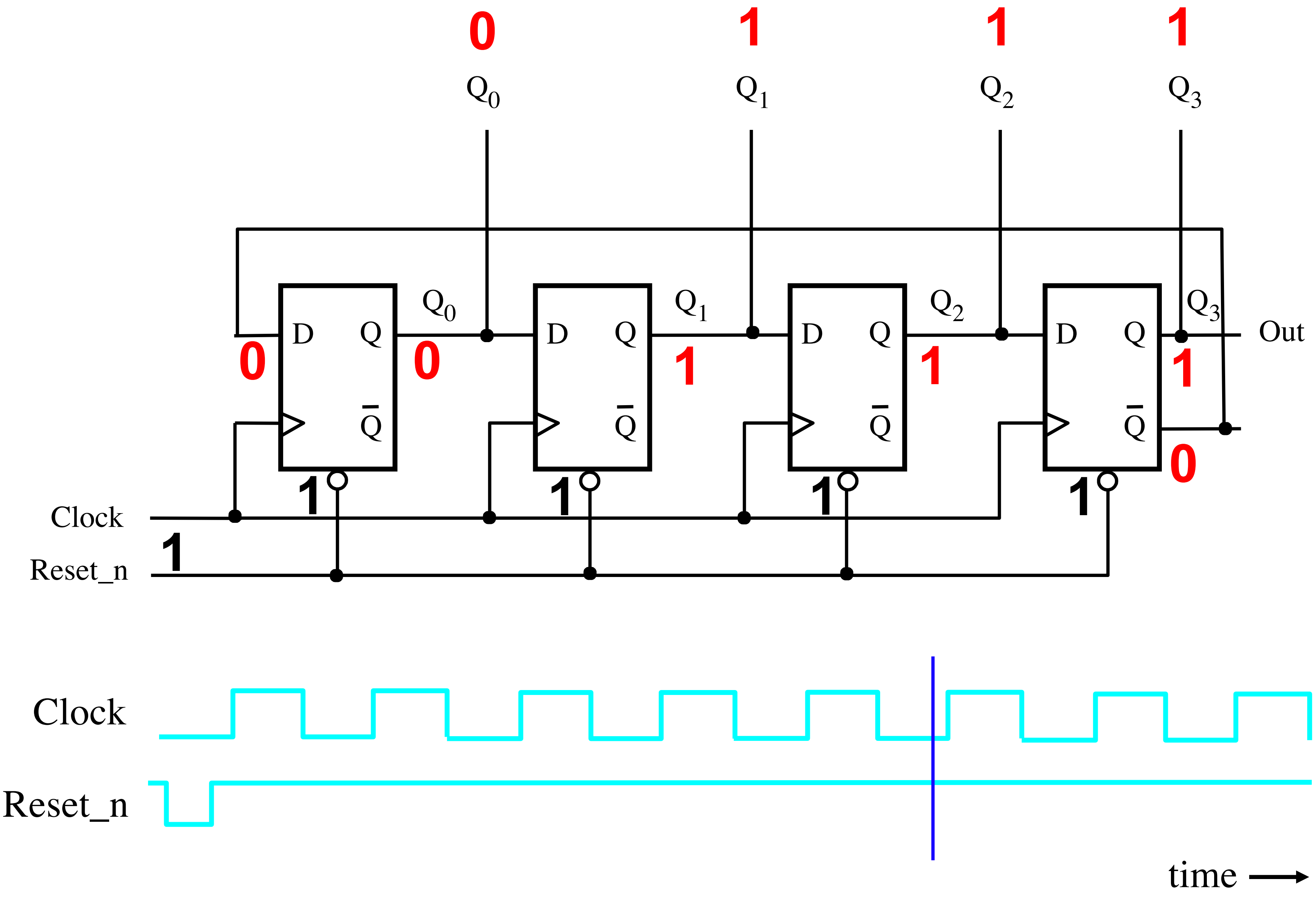
Counting: How does it work?



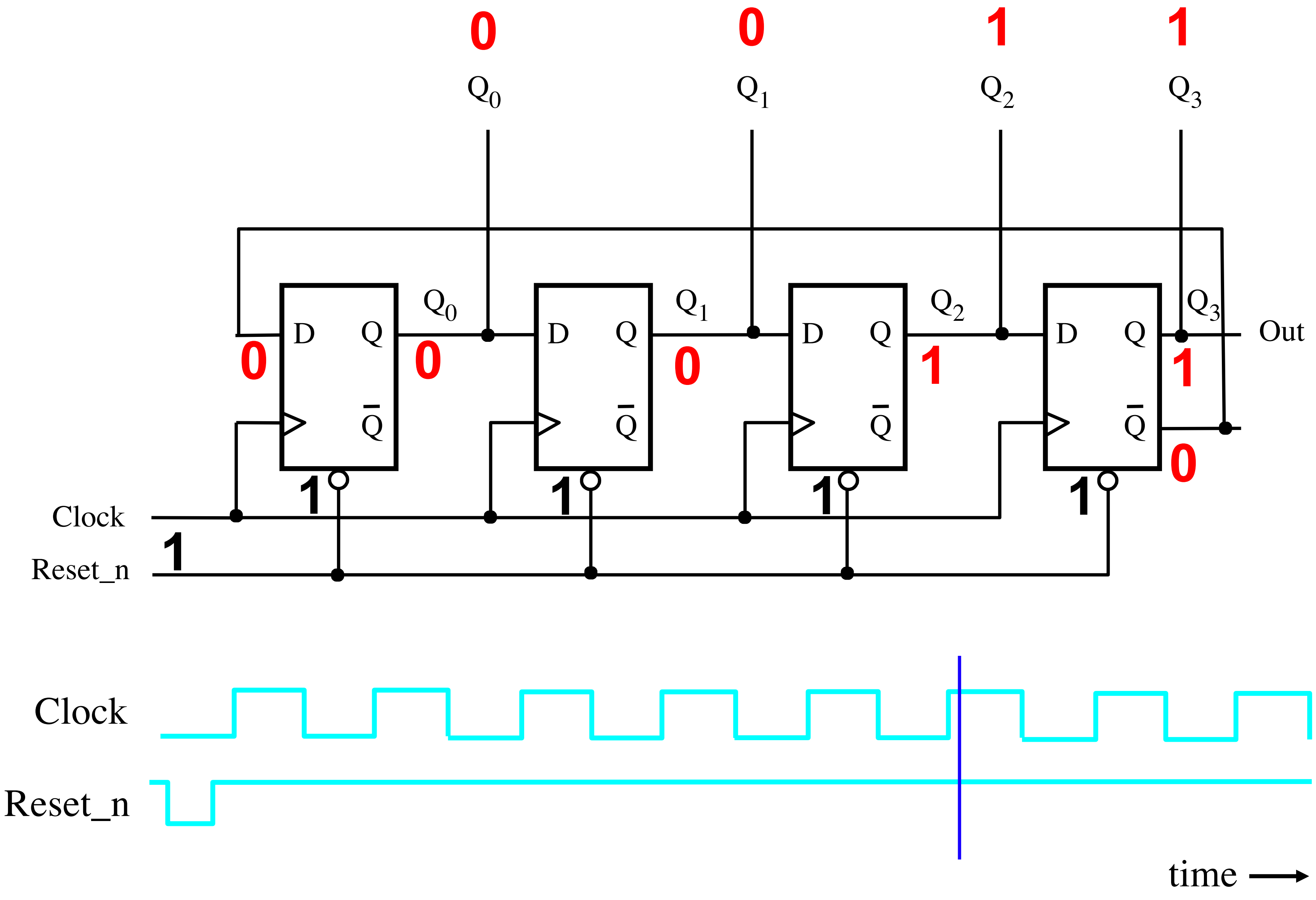
Counting: How does it work?



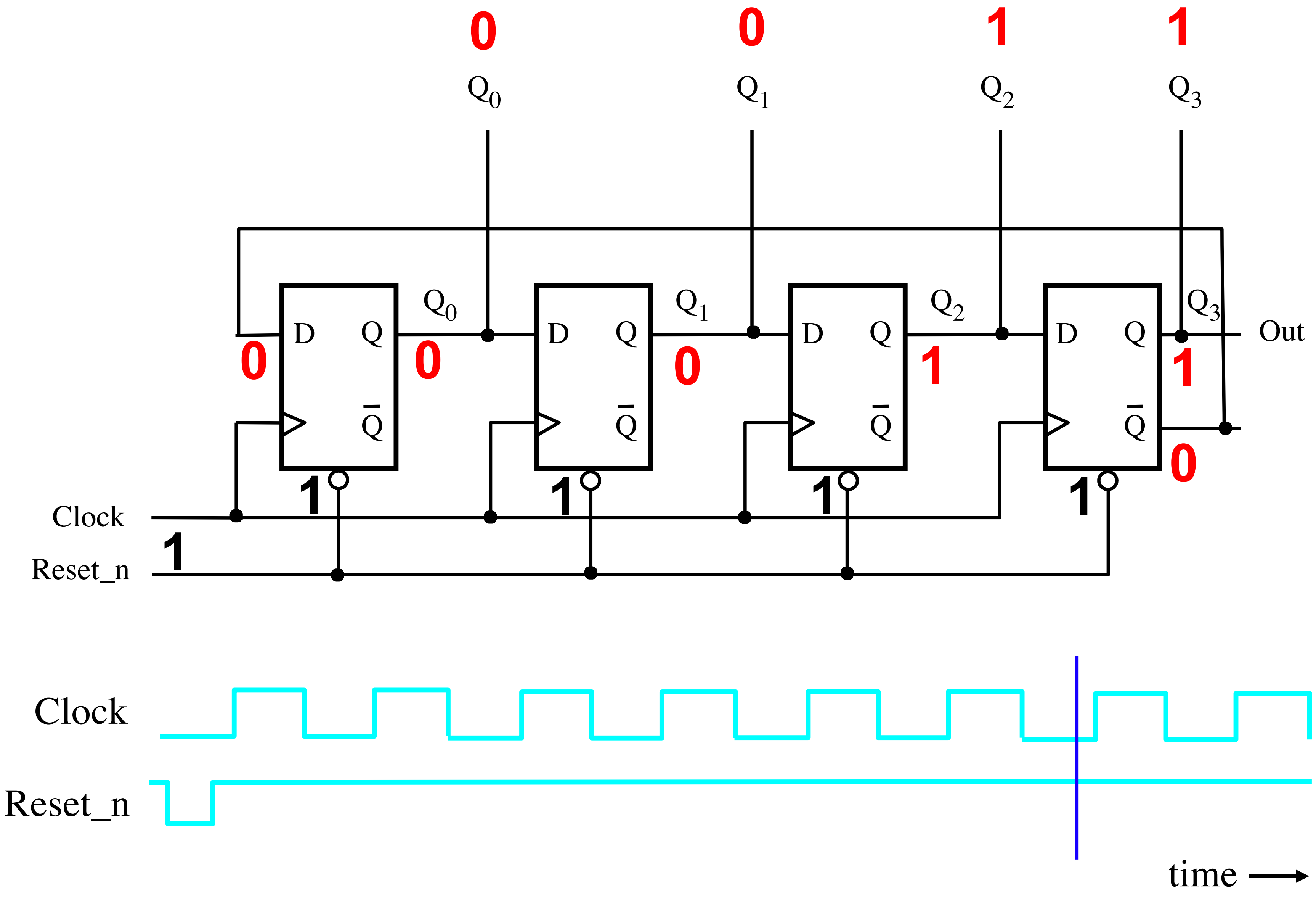
Counting: How does it work?



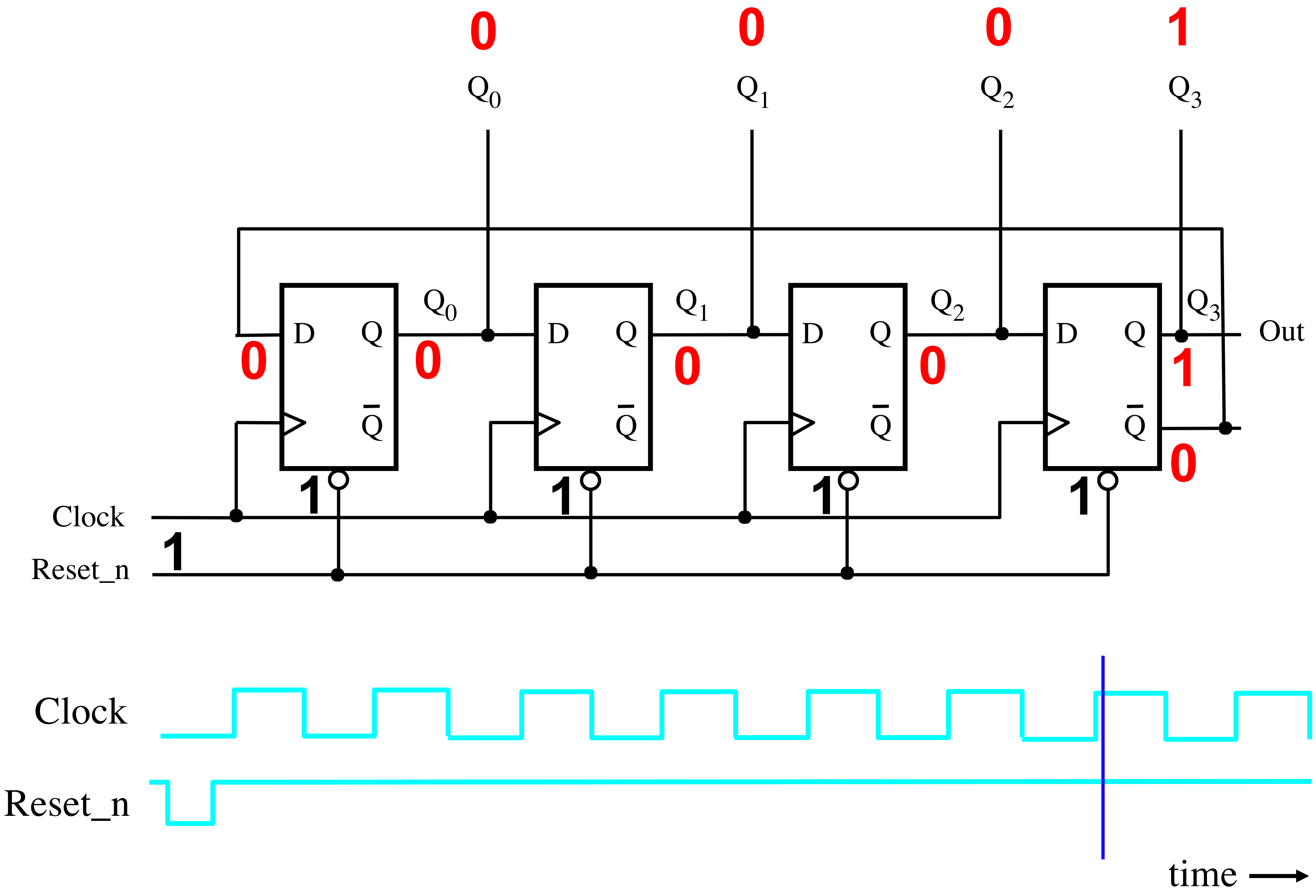
Counting: How does it work?



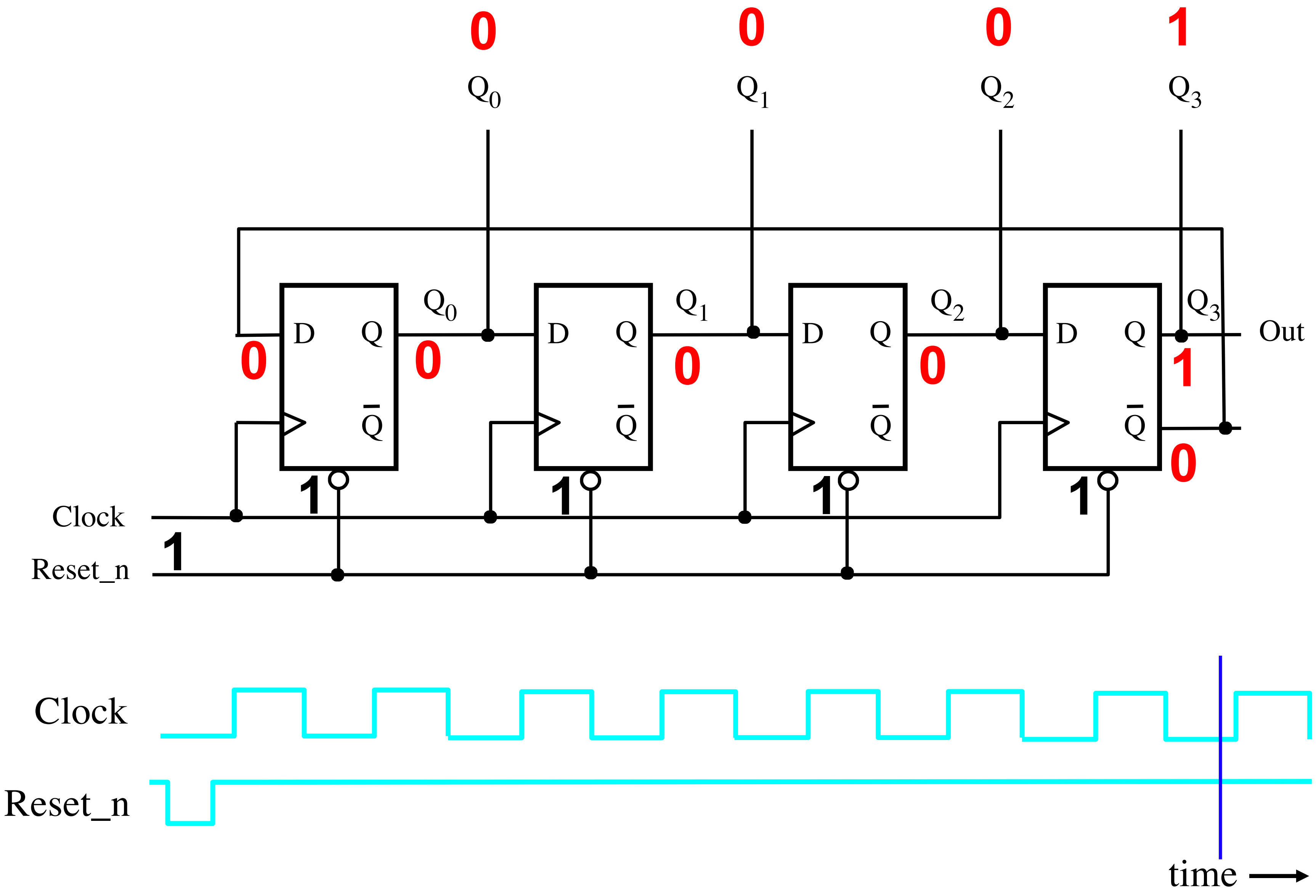
Counting: How does it work?



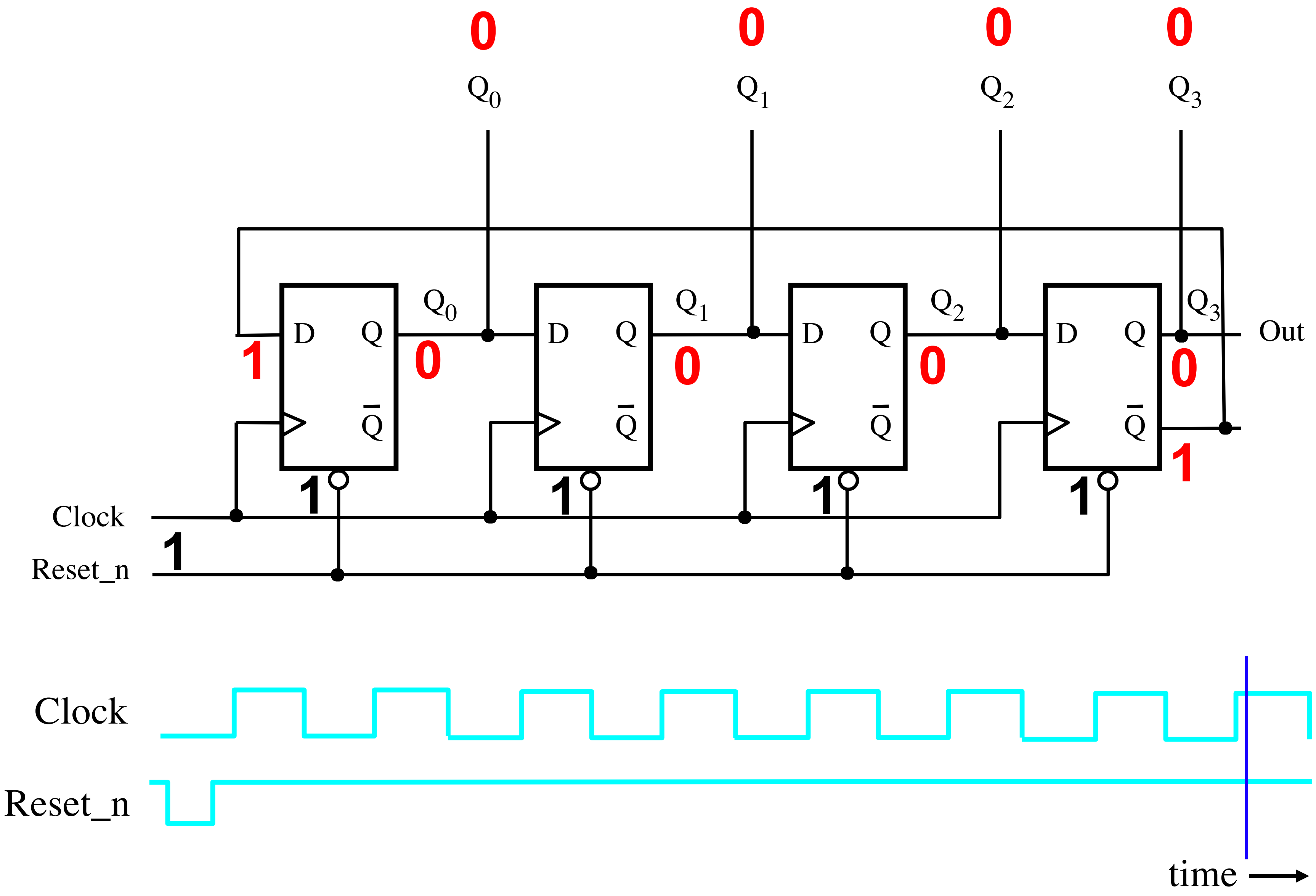
Counting: How does it work?



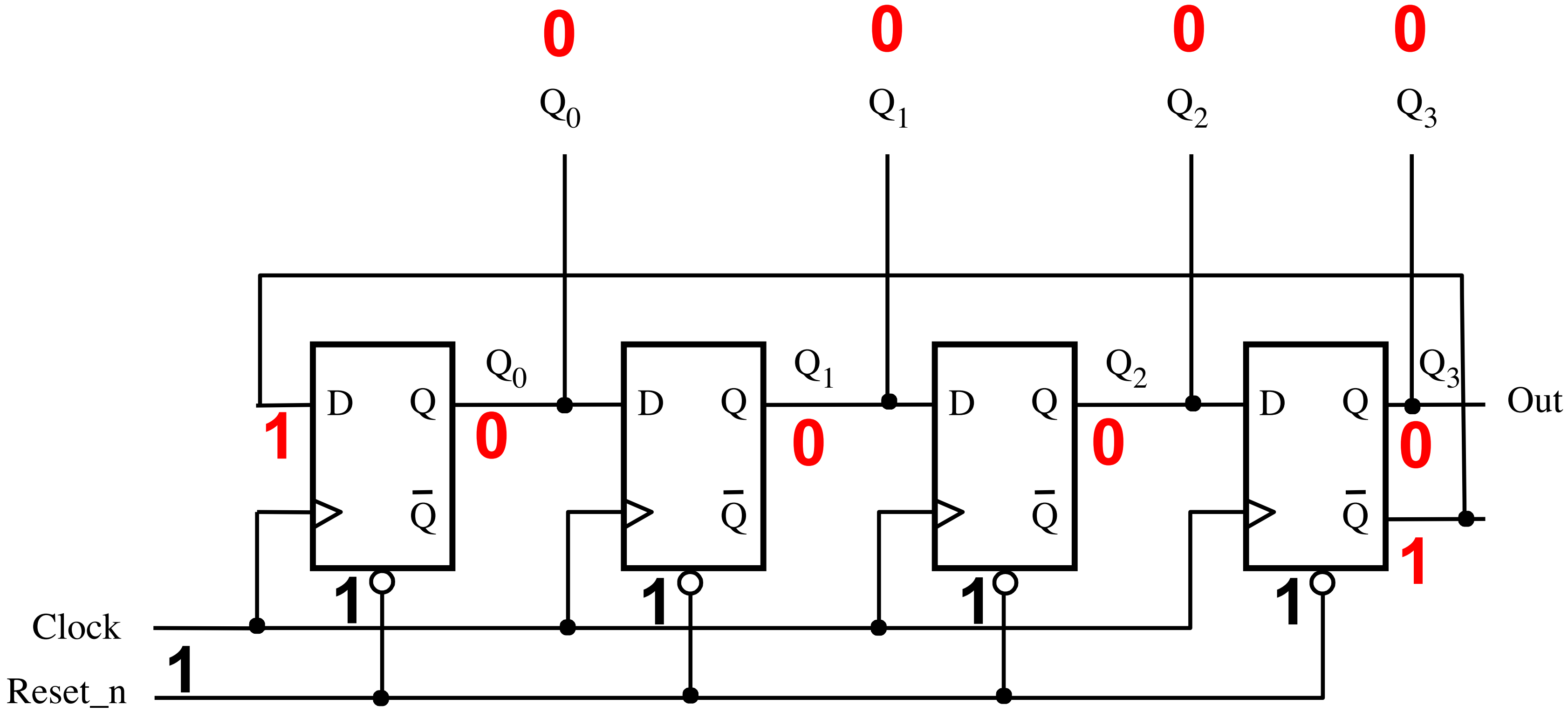
Counting: How does it work?



Counting: How does it work?

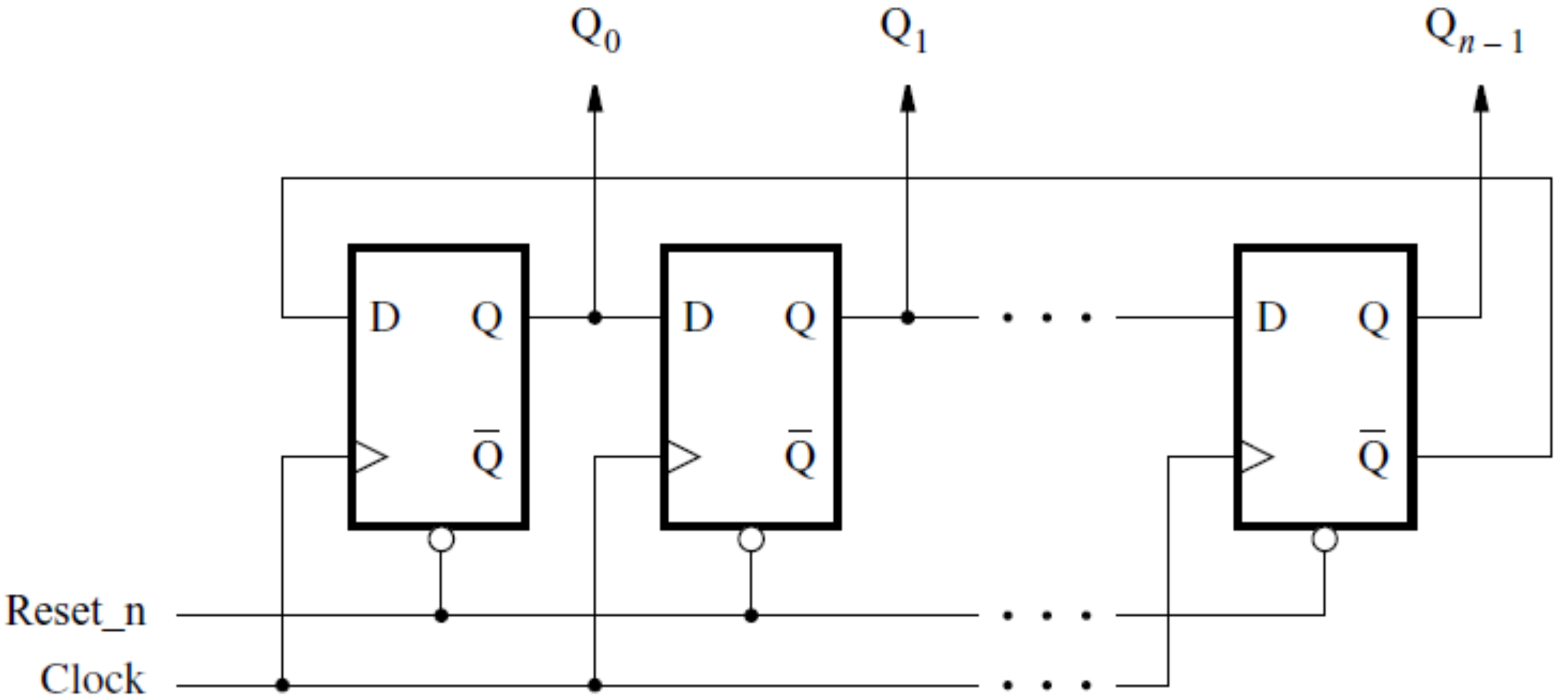


Counting: How does it work?



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

n-bit Johnson Counter



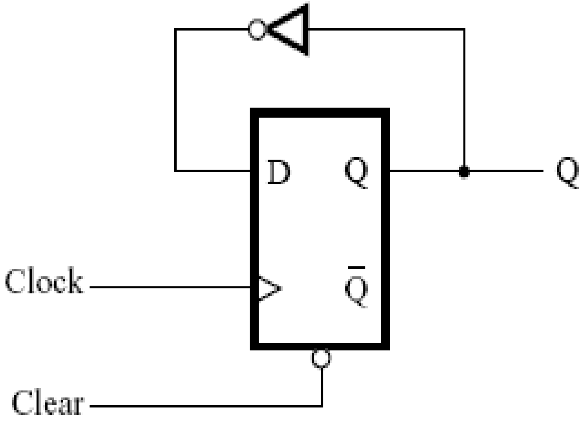
[Figure 5.29 from the textbook]

Timing Analysis of Flip-Flop Circuits
(Section 5.15)

Timing Review

- t_{su}: setup time
- t_h: hold time
- t_{cQ}: propagation delay

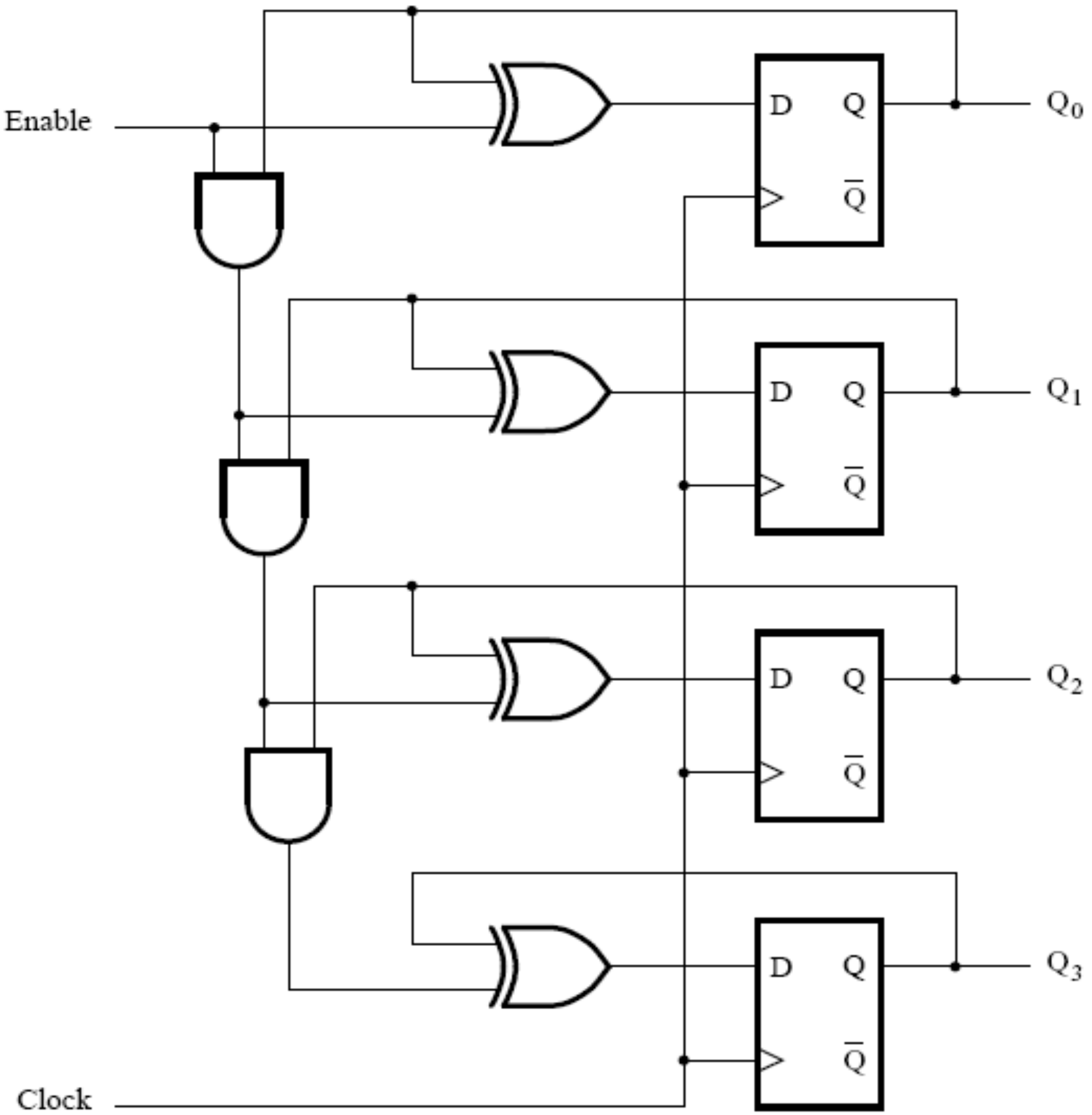
Timing Example



- t_{su}: 0.6ns
- t_h: 0.4ns
- t_{cQ}: 0.8ns to 1.0ns
  - Which value to use?
- Logic gate delay: 1+0.1k
  - k is equal to the number of inputs
- $T_{min} = t_{su} + t_{cQ} + t_{not} = 0.6 + 1.0 + 1.1 = 2.7$ns
- $F_{max} = 1/T_{min} = 370.37$ MHz

- Check for hold violations
  - Fastest Q can change = $t_{cQ} + t_{not} = 0.8 + 1.1 = 1.9$ns
  - 1.9ns > 0.4ns therefore no hold violations

Timing Example: 4-bit counter



[Figure 5.67 from the textbook]

Timing Example: 4-bit counter

- Look for longest path
  - Q0 to Q3
- Propagation delay of Q0
- 3 AND propagation delays
- 1 XOR propagation delay
- Setup delay for Q3

- $T_{min} = 1.0 + 3(1.2) + 1.2 + 0.6 = 6.4$ns
- $F_{max} = 1/6.4$ns $= 156.25$MHz

- Check for hold violations
  - Fastest Q can change = $t_{cQ} + t_{XOR} = 0.8 + 1.2 = 2$ns
  - 2.0ns > 0.4ns therefore no hold violations

Timing Example: Clock Skew

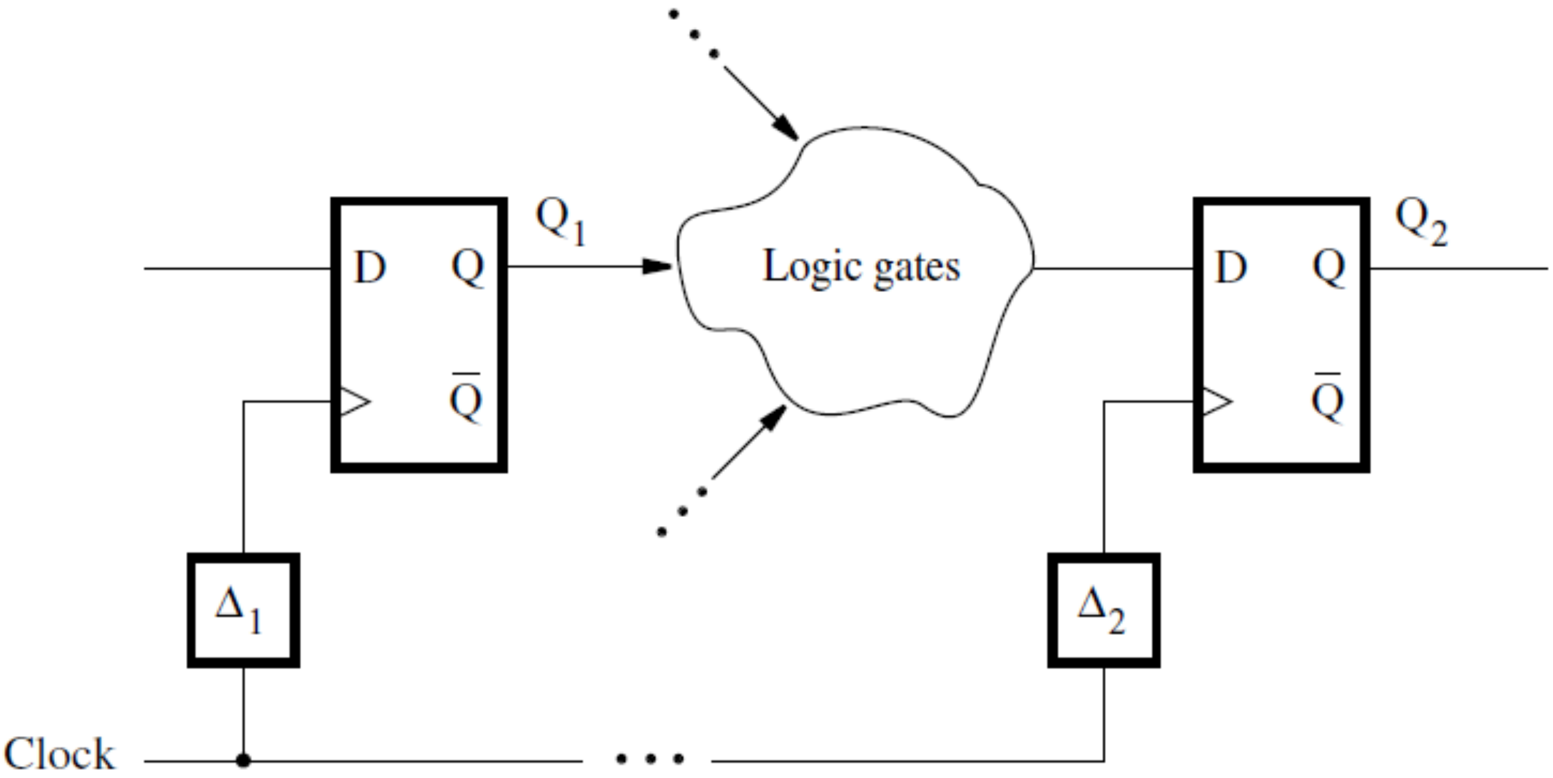


Figure 5.68. A general example of clock skew.

Skew Timing Example: 4-bit counter

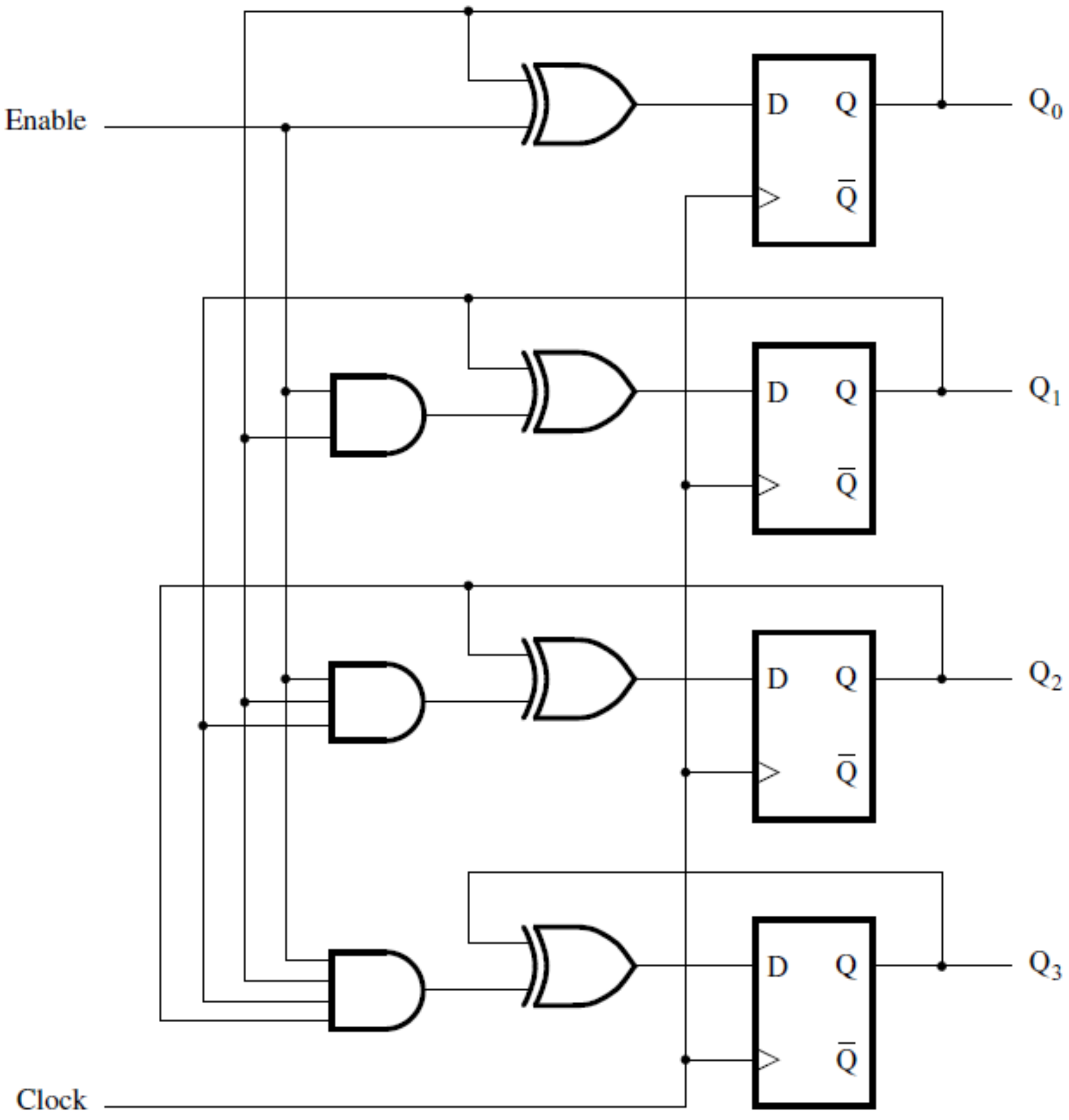
- Q3 now has a clock slew delay: 1.5ns
  - T = 1.0 + 3(1.2) + 1.2 + 0.6 - 1.5 = 4.9ns
- Now might not be the longest path
- Check Q0 to Q2
  - T = 1.0 + 2(1.2) + 1.2 + 0.6 = 5.2ns
- F_{max} = 1/5.2ns = 192.31 MHz

Example 5.22

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.**

A faster 4-bit counter



[Figure 5.75 from the textbook]

Faster 4-bit Counter

- Longest path: Q0 to Q3

- $T_{min} = t_{cQ0} + t_{AND} + t_{XOR} + t_{su}$
  $= 1.0 + 1.4 + 1.2 + 0.6 = 4.2\text{ns}$

- $F_{max} = 1/4.2\text{ns} = 238.1\text{ MHz} > 156.25\text{ MHz}$

Reaction Timer Circuit
(Section 5.14)

Problem Statement

- Want to design a reaction timer

- The circuit turns on a light (LED)

- A person then presses a switch

- Measure the elapsed time from when the LED is turned on until the switch is pressed

Clock Divider

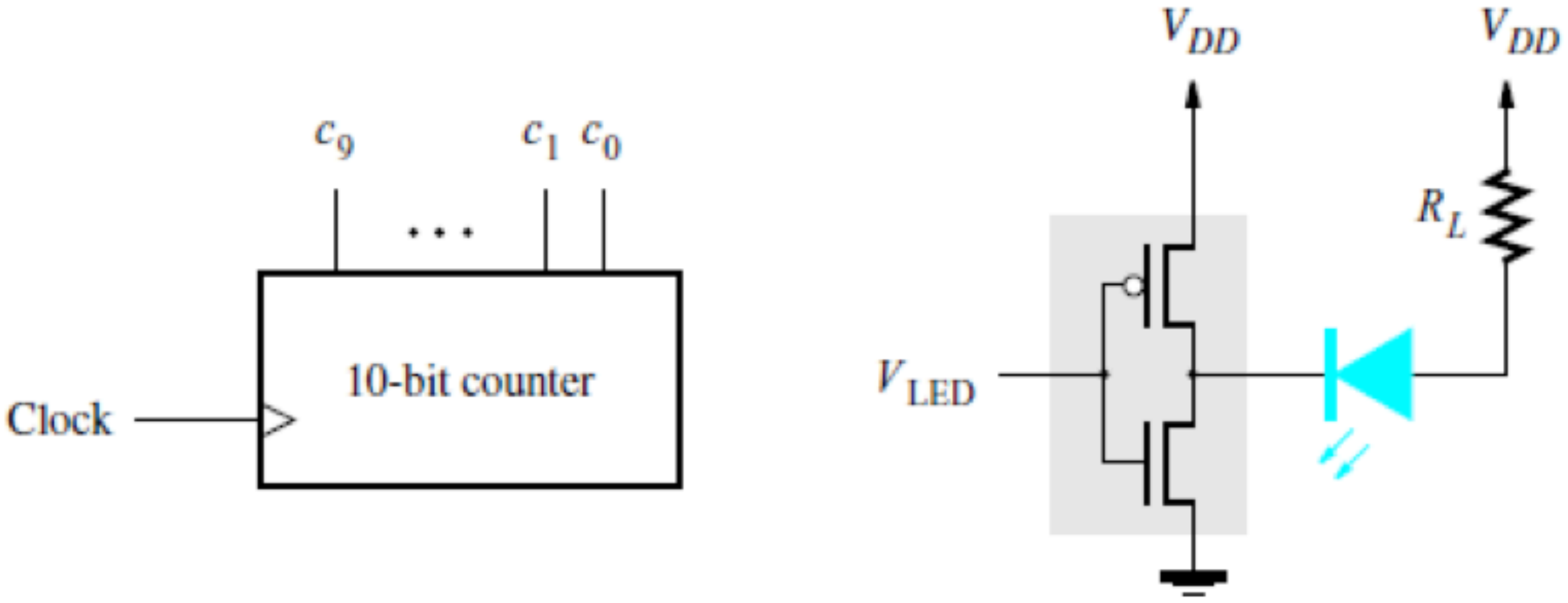
- **Input: 102.4kHz**

- **Output: 100Hz**

- **10-bit Counter to divide**

- **Output Frequency = 102.4k / 2^{10} = 100Hz**

A reaction-timer circuit



(a) Clock divider

(b) LED circuit

[Figure 5.61 from the textbook]

Functionality of circuit

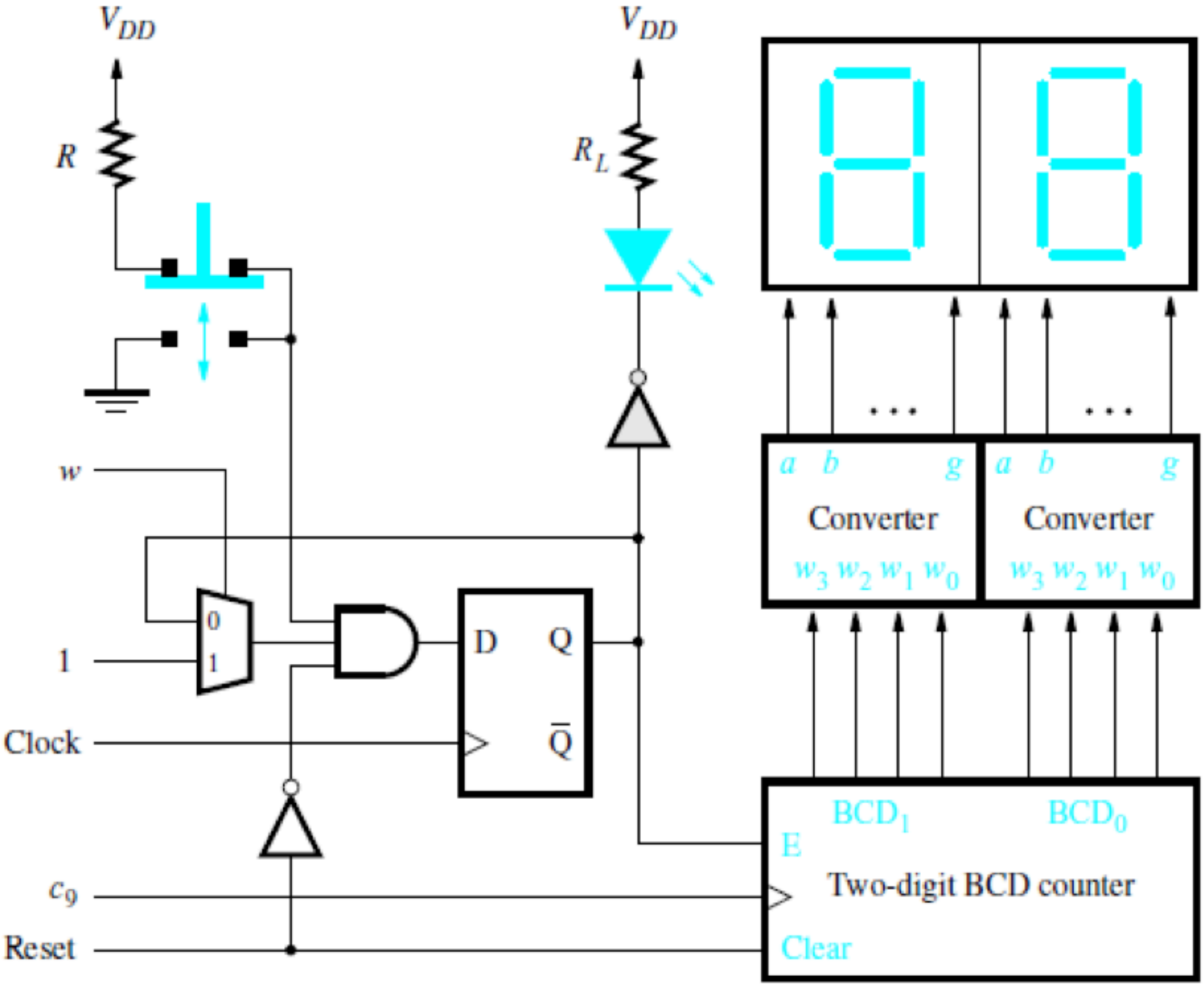
- **Push switch**
  - **Nominally 1**

- **DFF to keep track of the state**

- **Two-digit BCD counter**
  - **Output goes to converters to a 7-segment display**

- **Start-up**
  - **Assert the Reset signal**
    - **Clears counter**
    - **Clears flip-flop**
  - **Assert w=1 for one cycle**
  - **Once switch is hit**
    - **Clears flip-flop**
    - **Stops counting**

Push-button switch, LED, and 7-segment displays

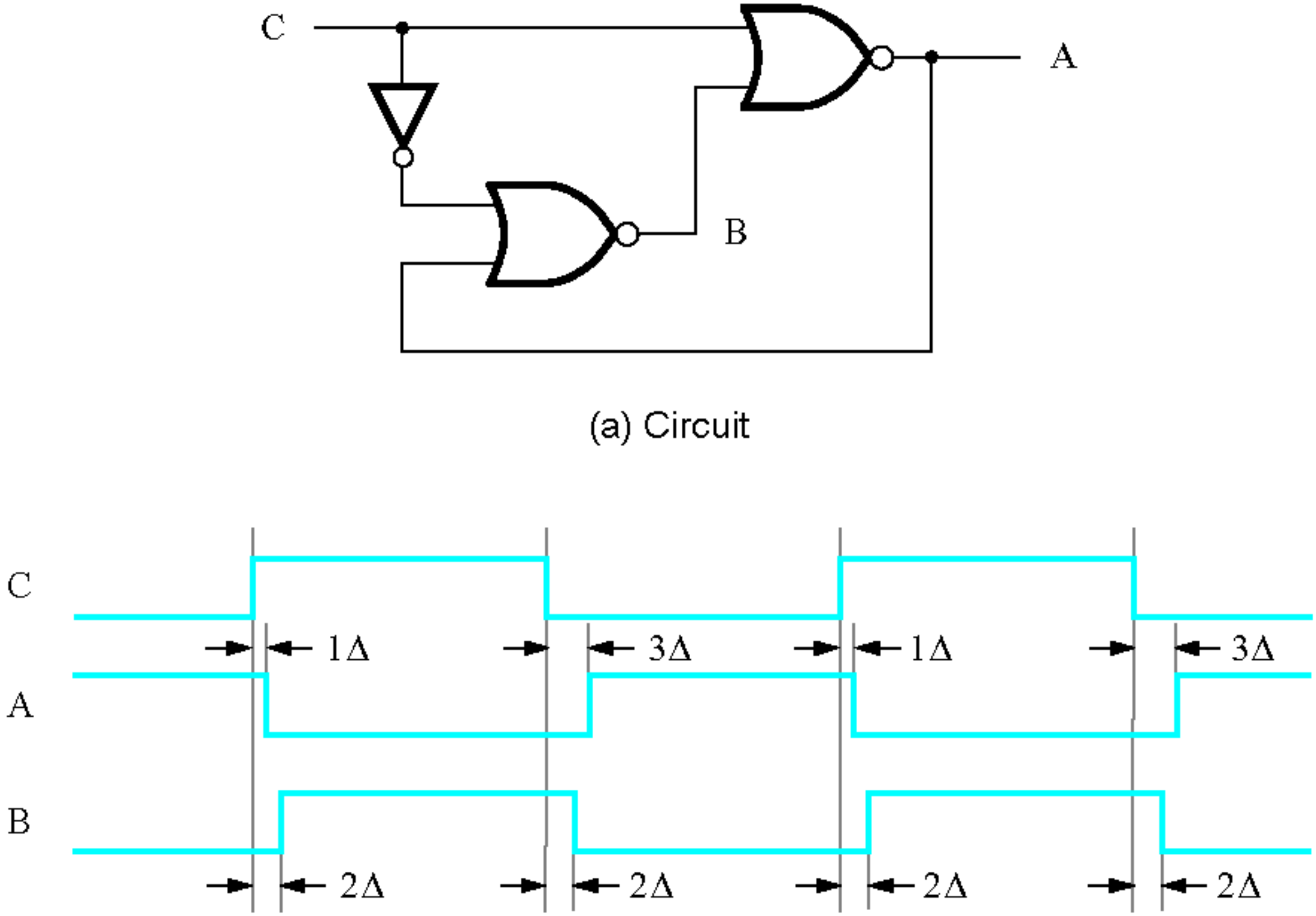


[Figure 5.61c from the textbook]

Examples of Solved Problems (Section 5.17)

Example 5.18

Circuit for Example 5.18



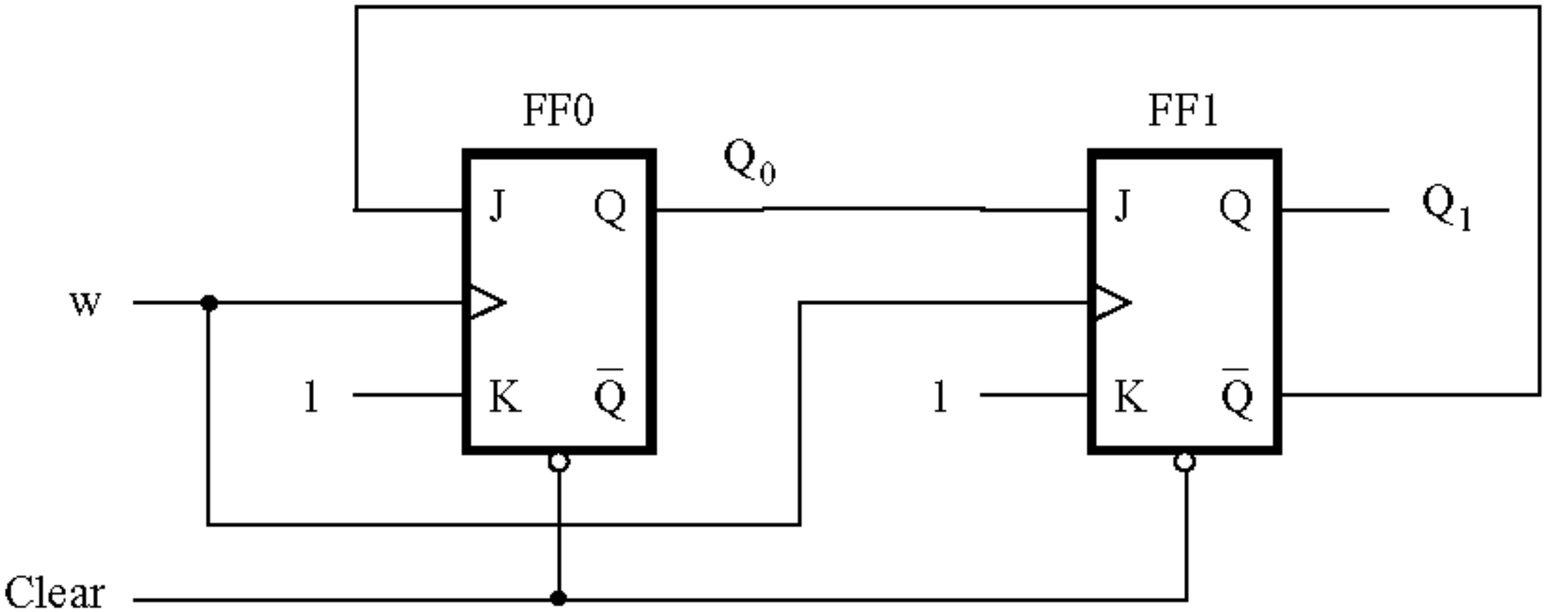
(a) Circuit

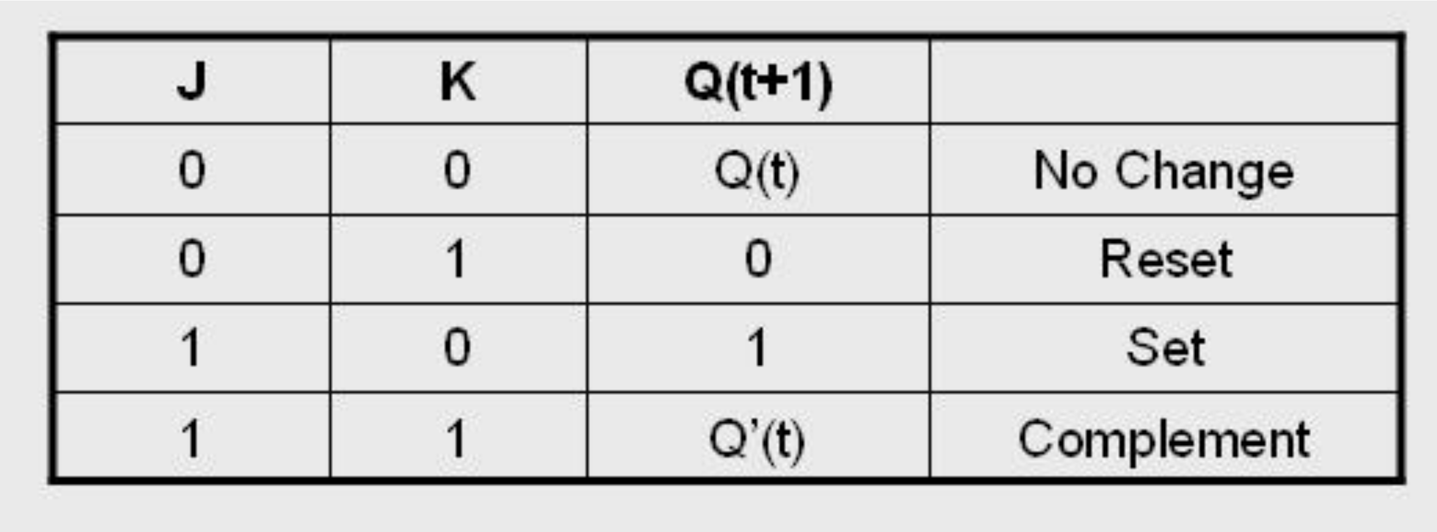
(b) Timing diagram

[Figure 5.70 from the textbook]

Example 5.19

Circuit for Example 5.19

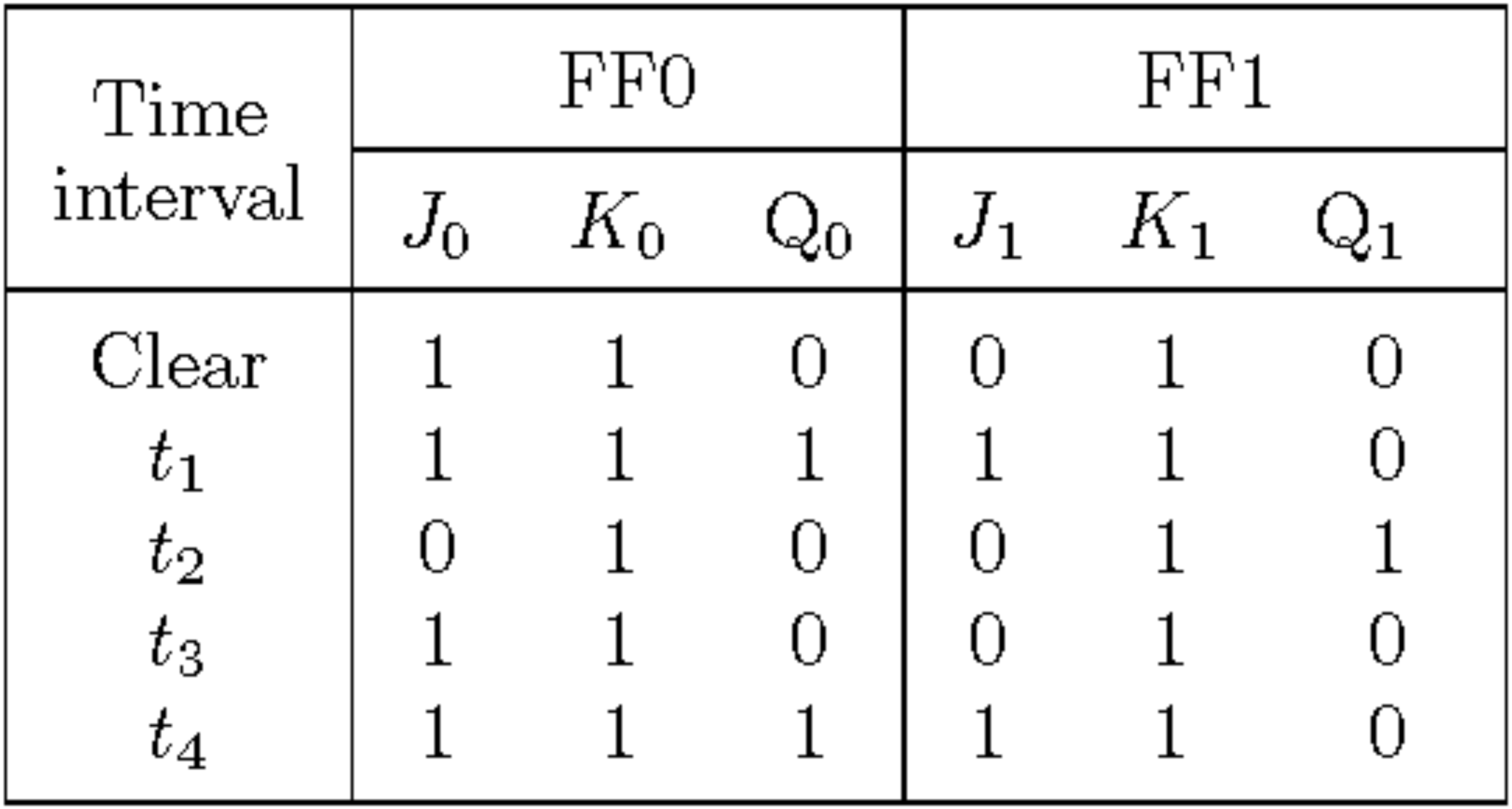




J	K	Q(t+1)	
0	0	Q(t)	No Change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

[Figure 5.71 from the textbook]

Summary of the behavior of the circuit



Time interval	FF0			FF1		
	J_0	K_0	Q_0	J_1	K_1	Q_1
Clear	1	1	0	0	1	0
t_1	1	1	1	1	1	0
t_2	0	1	0	0	1	1
t_3	1	1	0	0	1	0
t_4	1	1	1	1	1	0

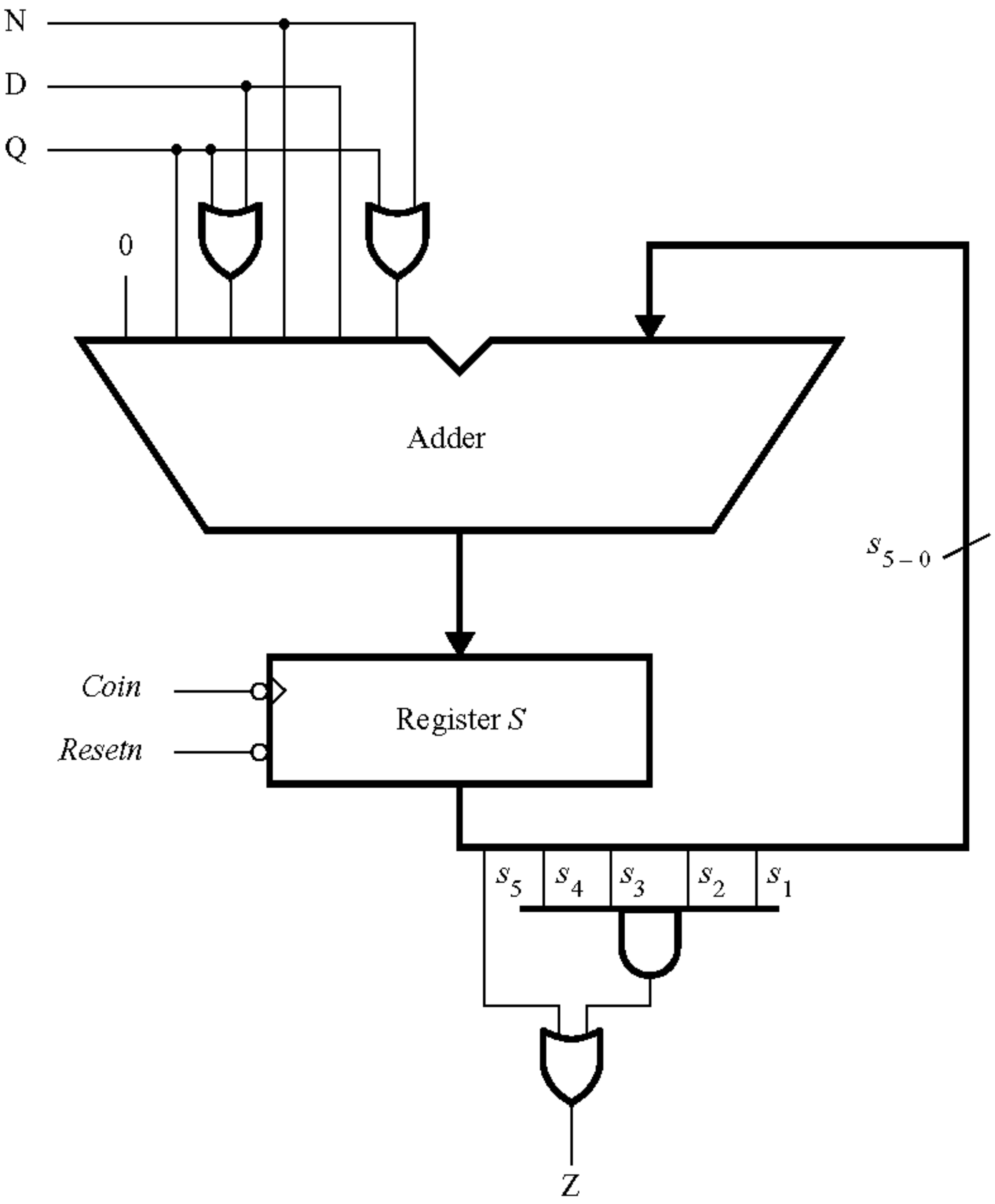
[Figure 5.72 from the textbook]

Example 5.20

Vending Machine Example

- Inputs N, D, Q, Coin, Resetn
  - N, D, Q: nickel, dime, quarter
  - Coin: pulsed when a coin is entered
    - Used to store values into register
  - Resetn: resets the register value to zero
- Add up new coin with old value
  - Store new sum into old value register
- See if total is above thirty cents
  - If so output Z goes high

Circuit for Example 5.20



[Figure 5.73 from the textbook]

Questions?

THE END