Chapter 2
Counters

Latest update: 08/01/2024

Table of Contents

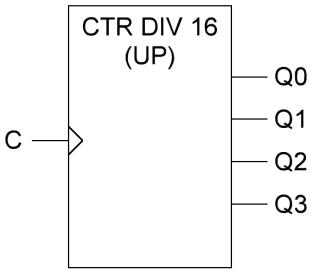
I. Definitions

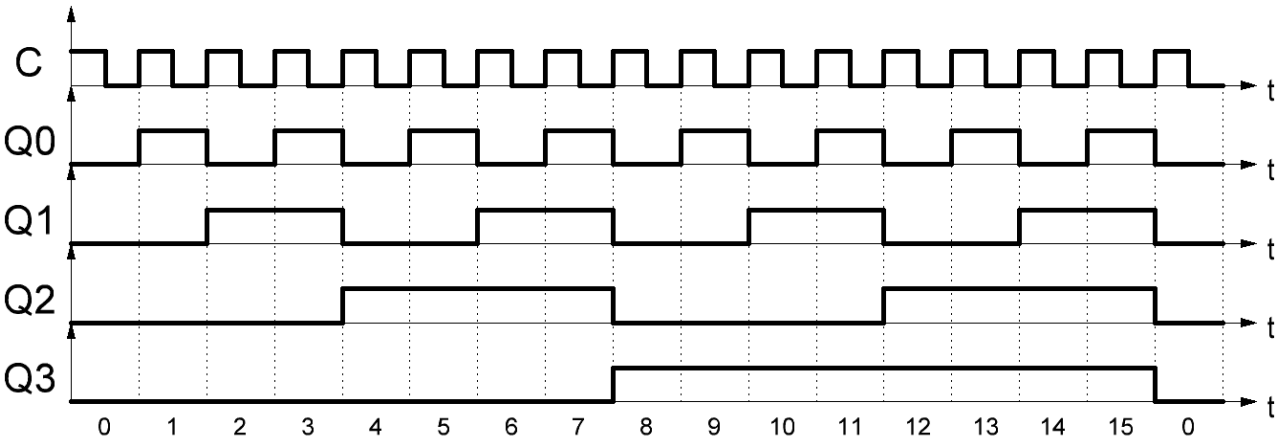
1. Binary Up Counters

A binary up counter is a sequential circuit with a clock input and an *n*-bit output. This output represents a natural binary number, which is incremented by one each time the counter is clocked.

We say that an up counter is 'modulo *m*' when it counts from **0** to ***m* – 1**. For instance:
- A modulo-8 up counter counts from 0 to 7.
- A modulo-10 up counter counts from 0 to 9.
- A modulo-16 up counter counts from 0 to 15.

Example of a modulo-16 up counter:





It is noteworthy that an output (Q_n) toggles on each **falling edge** of the previous output (Q_{n-1}):
- *Q1* toggles on each **falling edge** of *Q0*.
- *Q2* toggles on each **falling edge** of *Q1*.
- *Q3* toggles on each **falling edge** of *Q2*.

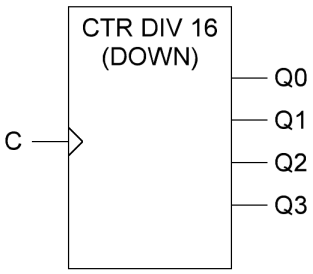
Notes:
- A 'modulo-10 up counter' can also be called: 'decade counter', 'decimal counter' or 'BCD counter'.
- A 'modulo-*m* up counter' with $m = 2^n$ is a **'full-sequence counter'**.
- A 'modulo-*m* up counter' with $m \neq 2^n$ is a **'truncated-sequence counter'**.

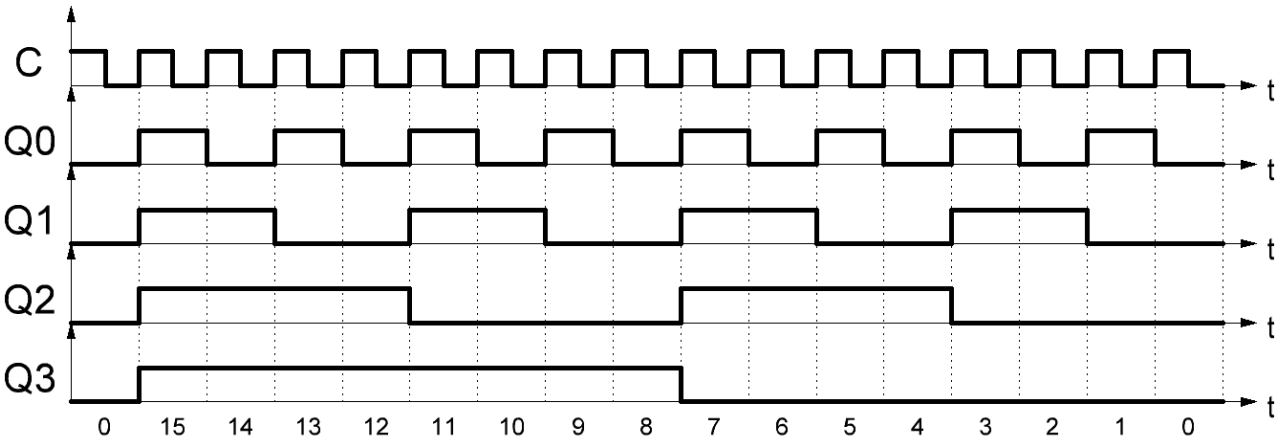
2. Binary Down Counters

A binary down counter is a sequential circuit with a clock input and an *n*-bit output. This output represents a natural binary number, which is decremented by one each time the counter is clocked.

We say that a down counter is 'modulo *m*' when it counts backwards from ***m* – 1** to **0**. For instance:
- A modulo-8 down counter counts backwards from 7 to 0.
- A modulo-10 down counter counts backwards from 9 to 0.
- A modulo-16 down counter counts backwards from 15 to 0.

Example of a modulo-16 down counter:





It is noteworthy that an output (Q_n) toggles on each **rising edge** of the previous output (Q_{n-1}):
- *Q1* toggles on each **rising edge** of *Q0*.
- *Q2* toggles on each **rising edge** of *Q1*.
- *Q3* toggles on each **rising edge** of *Q2*.

Notes:
- A 'modulo-10 down counter' can also be called: 'decade down counter', 'decimal down counter' or 'BCD down counter'.
- A 'modulo-*m* down counter' with $m = 2^n$ is a **'full-sequence down counter'**.
- A 'modulo-*m* down counter' with $m \neq 2^n$ is a **'truncated-sequence down counter'**.

3. Other Types of Counters

Actually, a counter can count through any arbitrary sequence. A sequence is not necessarily a natural binary sequence. For instance, a Gray (up) counter counts through the Gray code. A sequence can also be designed to satisfy the special needs of a circuit.

II. Designing Asynchronous Counters

1. Principle

In a binary counter (up or down), the frequency of an output (Q_n) is half the frequency of its previous output (Q_{n-1}):

- The frequency of *Q1* is half the frequency of *Q0*.
- The frequency of *Q2* is half the frequency of *Q1*.
- And so forth.

The frequencies are successively divided by two (see the timing diagram of a binary up counter – I.1 above). This can also be seen in a natural binary sequence:

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Therefore, the key principle of a counter is to divide frequencies by two. This can be easily achieved by connecting divide-by-two circuits in series.

Now, we have to understand what a divide-by-two circuit is and how it can be built.

2. Divide-by-Two Circuits or Toggle Flip-Flops

A divide-by-two circuit, **commonly called a 'toggle flip-flop'**, divides the frequency of its input by two. There are several ways to build one; the most common include either a D or a JK flip-flop:

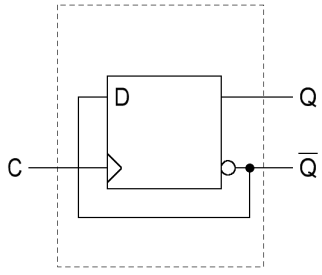


Figure 1

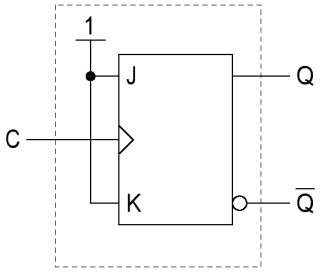
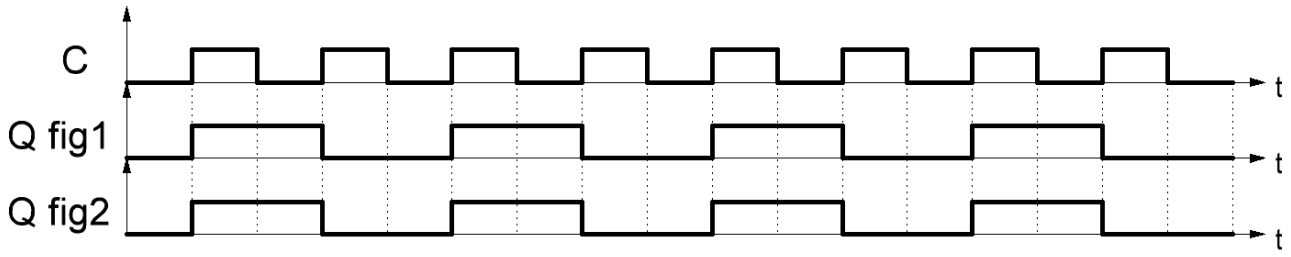
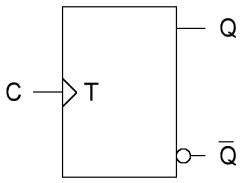


Figure 2

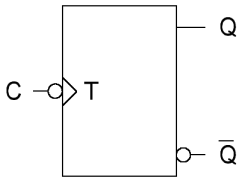
The timing diagrams of these two circuits are identical:

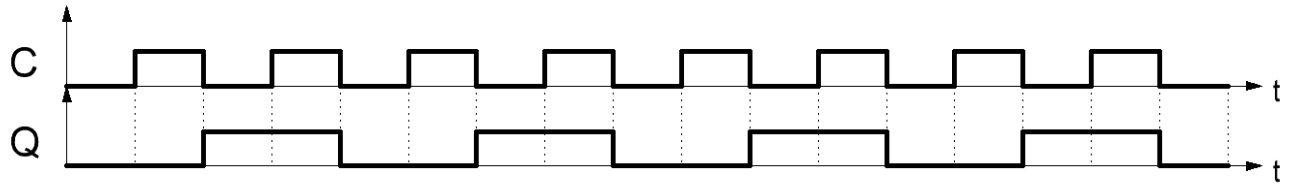


We can associate the toggle flip-flop with the following circuit symbol:



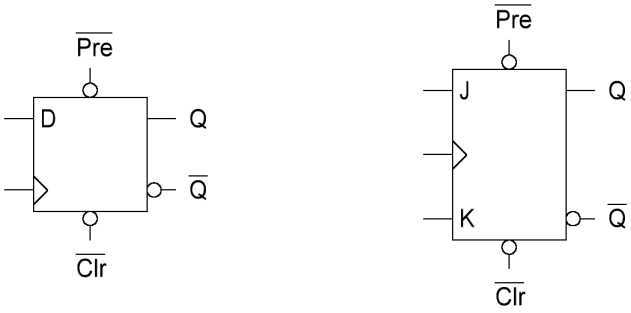
This toggle flip-flop is positive-edge-triggered, but if we use a D or JK negative-edge-triggered flip-flop, we can build a negative-edge-triggered toggle flip-flop. Its circuit symbol and timing diagram are as follows:



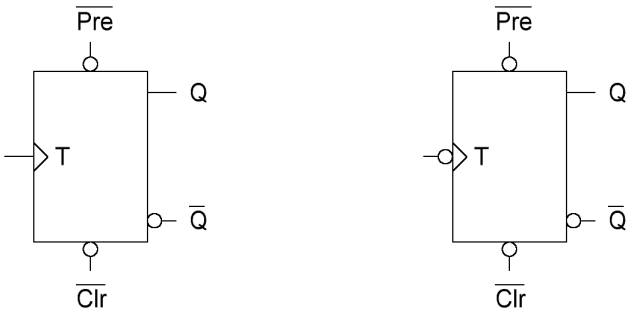


D or JK flip-flops usually come with two asynchronous active-low inputs called $\overline{clear}$ and $\overline{preset}$ (or $\overline{reset}$ and $\overline{set}$). In normal operation, both of these inputs are 1s. When $\overline{preset} = 0$, Q is set to 1, and when $\overline{clear} = 0$, Q is reset to 0. These inputs are unconditional, that is to say they override every other input of the flip-flop.

Their symbols are as follows:



Therefore, we can add these inputs to our toggle flip-flops:



These preset and clear inputs are not always required. Thus, when we do not need them, we will not represent them on diagrams assuming that they will always be inactive ($\overline{clear} = \overline{preset} = 1$).

3. Full-Sequence Counters

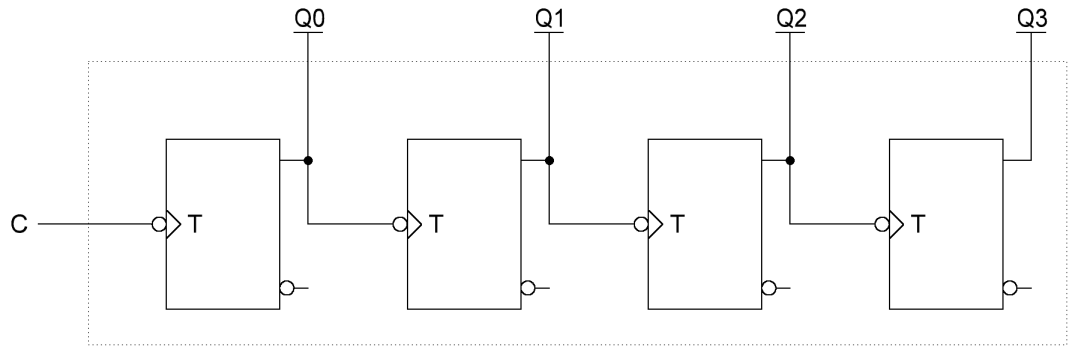
3.1. Up Counters

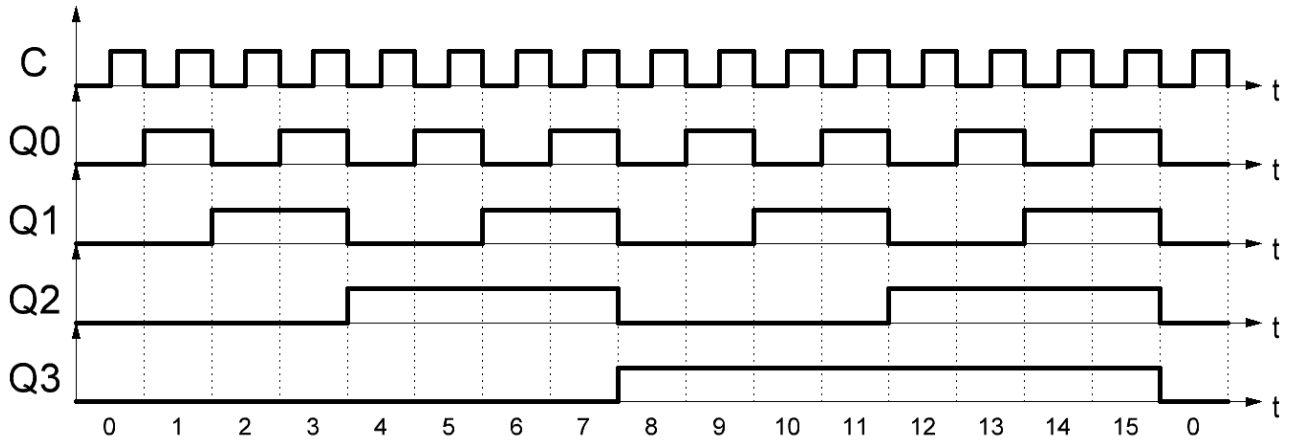
A full-sequence up counter is an *n*-bit up counter with a modulo of 2^n. It cycles through the natural binary sequence from 0 to $2^n - 1$.

To design an *n*-bit up counter, we need *n* toggle flip-flops. Then, we should connect them in series so that an output (Q_n) toggles on each **falling edge** of its previous output (Q_{n-1}).

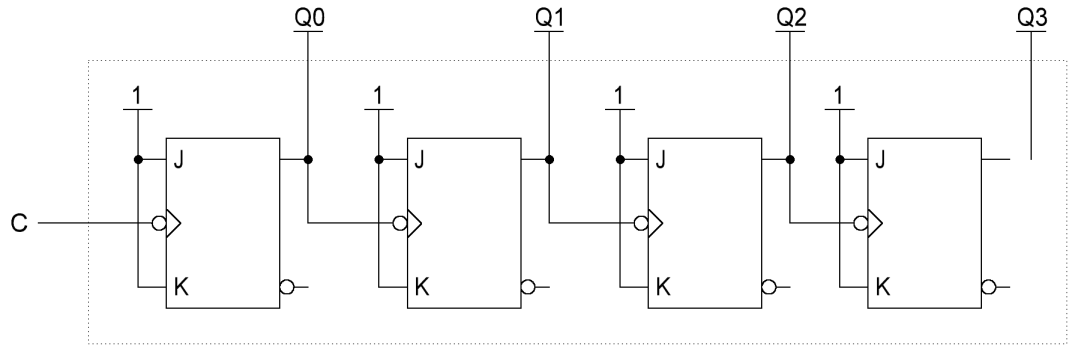
For instance, to design a 4-bit up counter, we should connect 4 toggle flip-flops in series so that:
- *Q1* toggles on each **falling edge** of *Q0*.
- *Q2* toggles on each **falling edge** of *Q1*.
- *Q3* toggles on each **falling edge** of *Q2*.

If we use negative-edge-triggered flip-flops, this is what the circuit and timing diagrams should look like:

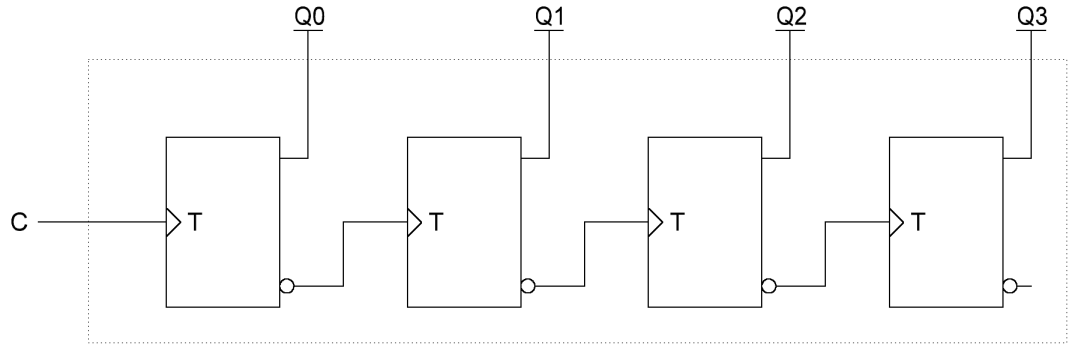




For example, if the toggle flip-flops are made up of JK flip-flops, the circuit diagram is as follows:



Of course, we can also use positive-edge-triggered flip-flops:

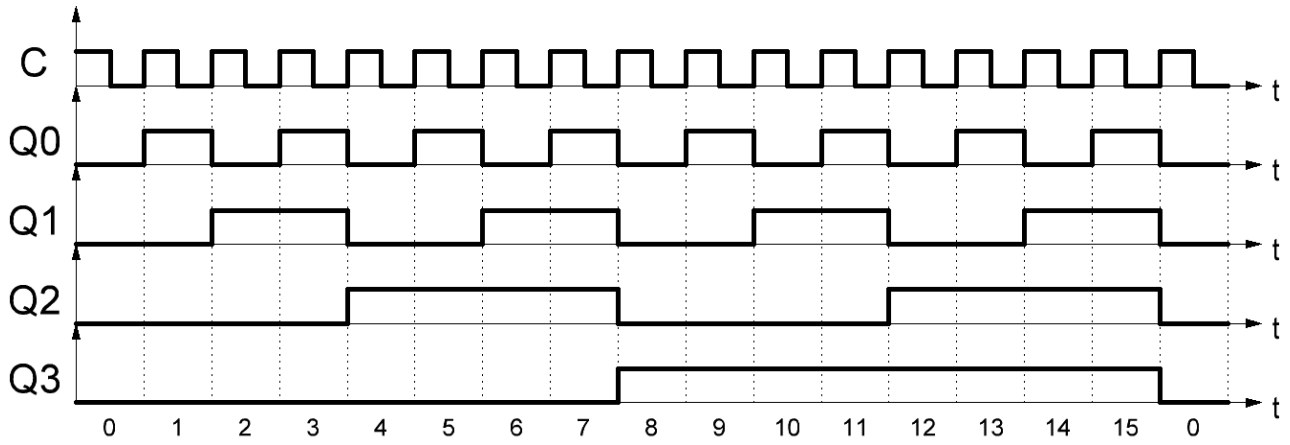


The circuit above is still an up counter because:

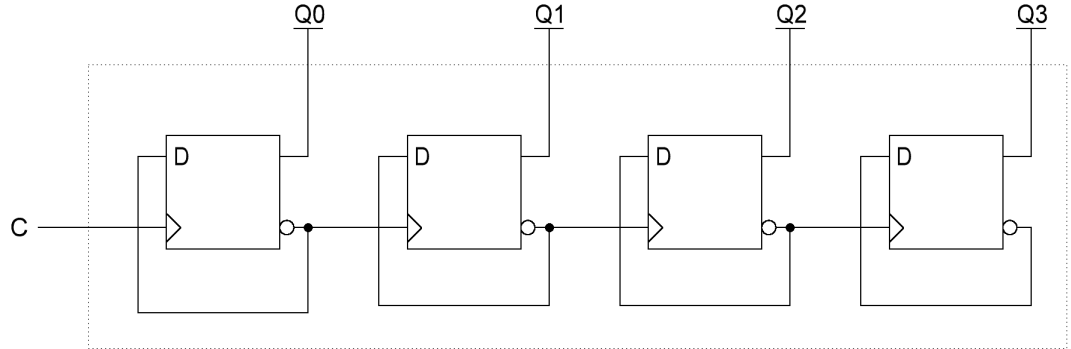
- *Q1* toggles on each rising edge of $\overline{Q0}$ (that is to say, on each **falling edge** of *Q0*).
- *Q2* toggles on each rising edge of $\overline{Q1}$ (that is to say, on each **falling edge** of *Q1*).
- *Q3* toggles on each rising edge of $\overline{Q2}$ (that is to say, on each **falling edge** of *Q2*).

It is noteworthy that in an up counter, there is always **one bubble** on the wire that connects an output (Q_n or $\overline{Q_n}$) to the next clock input.

Here is the timing diagram:



For example, if the toggle flip-flops are made up of D flip-flops, the circuit diagram is as follows:



3.2. Down Counters

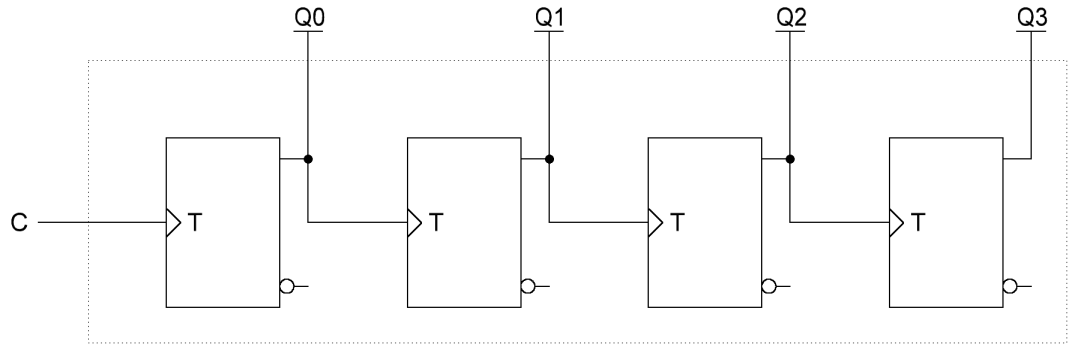
A full-sequence down counter is an *n*-bit down counter with a modulo of 2^n. It cycles through the natural binary sequence from $2^n - 1$ to 0.

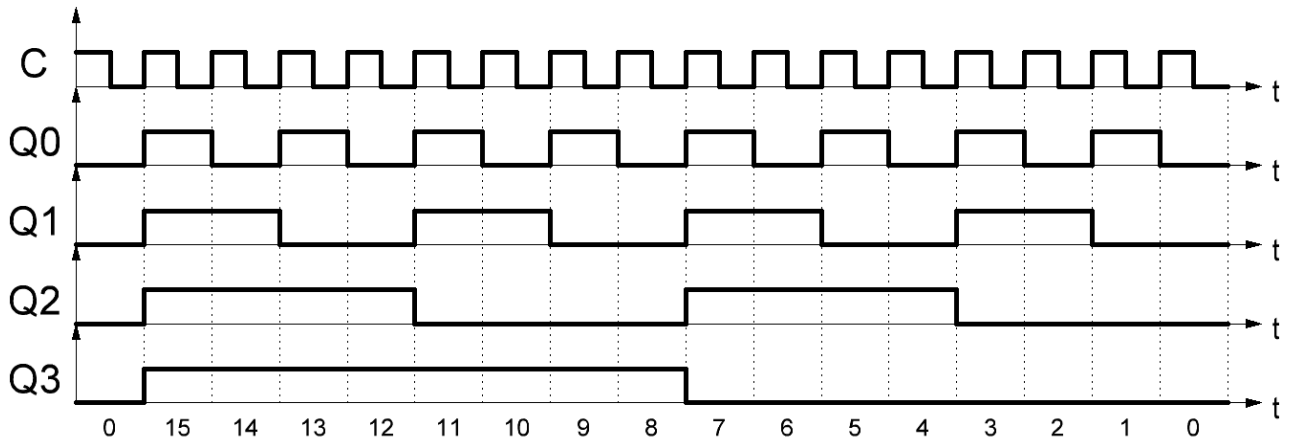
To design an *n*-bit down counter, we need *n* toggle flip-flops. Then, we connect them in series so that an output (Q_n) toggles on each **rising edge** of its previous output (Q_{n-1}).

For instance, to design a 4-bit down counter, we should connect 4 toggle flip-flops in series so that:

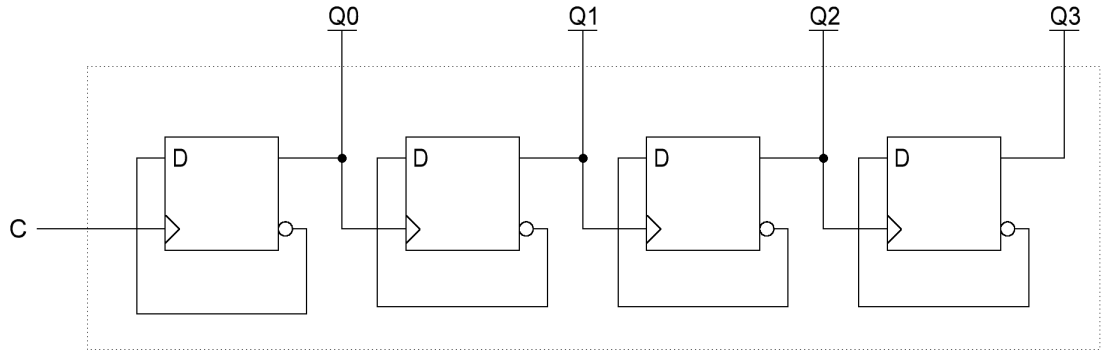
- *Q1* toggles on each **rising edge** of *Q0*.
- *Q2* toggles on each **rising edge** of *Q1*.
- *Q3* toggles on each **rising edge** of *Q2*.

If we use positive-edge-triggered flip-flops, this is what the circuit and timing diagrams should look like:

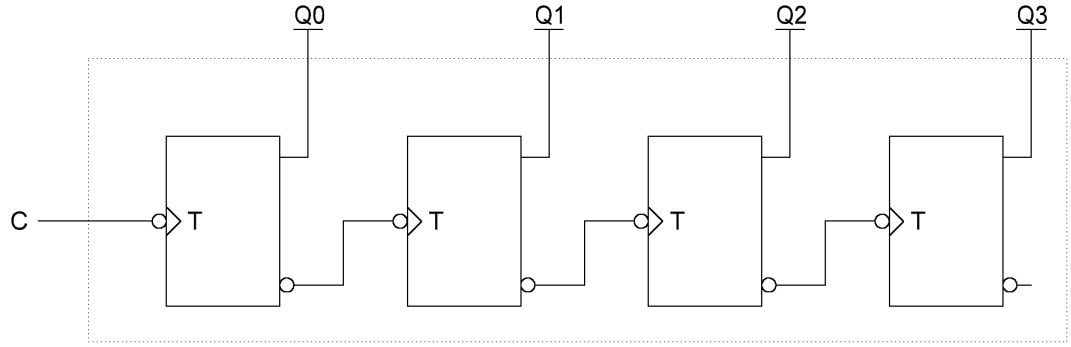




For example, if the toggle flip-flops are made up of D flip-flops, the circuit diagram is as follows:



Of course, we can also use negative-edge-triggered flip-flops:

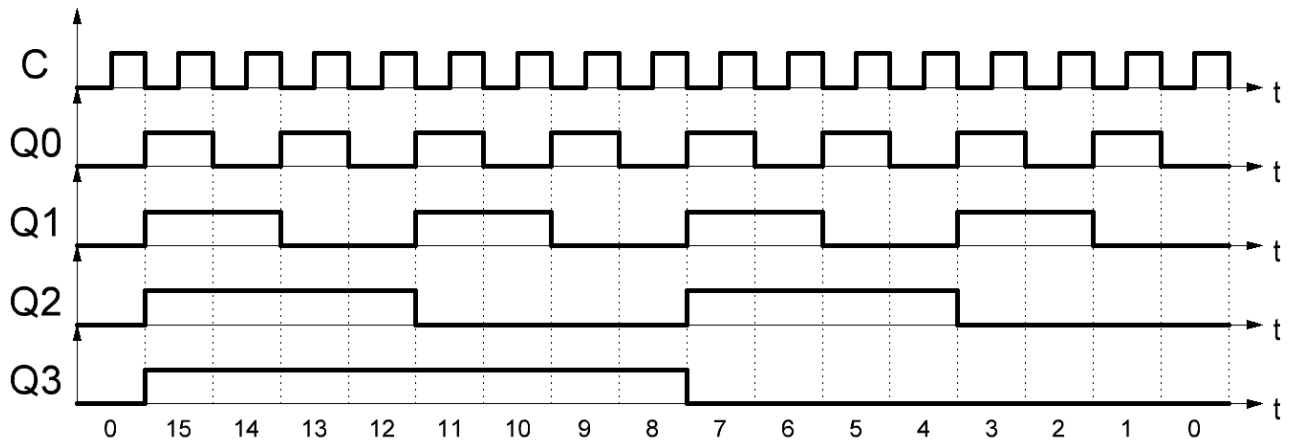


The circuit above is still a down counter because:

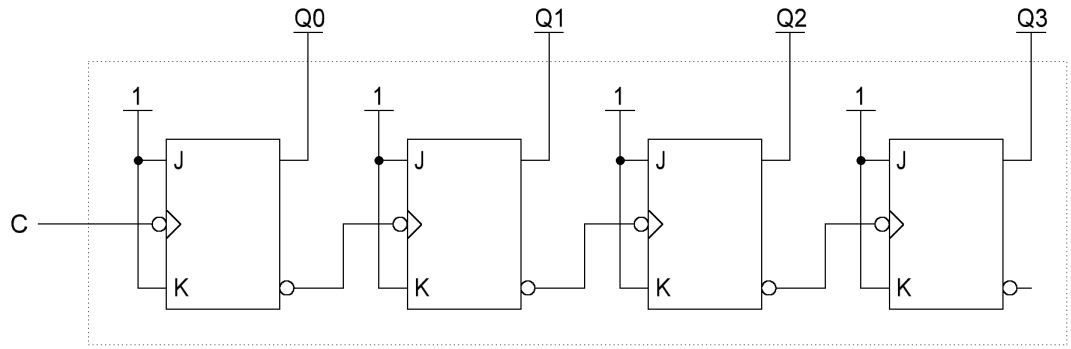
- *Q1* toggles on each falling edge of $\overline{Q0}$ (that is to say, on each **rising edge** of *Q0*).
- *Q2* toggles on each falling edge of $\overline{Q1}$ (that is to say, on each **rising edge** of *Q1*).
- *Q3* toggles on each falling edge of $\overline{Q2}$ (that is to say, on each **rising edge** of *Q2*).

It is noteworthy that in a down counter, there are either **two bubbles or none at all** on the wire that connects an output (Q_n or $\overline{Q}_n$) to the next clock input.

Here is the timing diagram:



For example, if the toggle flip-flops are made up of JK flip-flops, the circuit diagram is as follows:



4. Truncated-Sequence Counters

4.1. Up Counters

A truncated-sequence up counter is an *n*-bit up counter with a modulo lower than 2^n. It cycles through the natural binary sequence from 0 to a value lower than $2^n - 1$.

The first step in designing a truncated-sequence counter is to build a full-sequence counter. Then, the value of the modulo must be detected and replaced by the value 0.

For instance, a modulo-11 up counter is a truncated-sequence counter because it requires a 4-bit output, which has 16 combinations, when only 11 are needed. In other words, it cycles through from 0 to 10 instead of cycling through from 0 to 15.

To design it, we have to build a modulo-16 up counter and replace the value 11 by the value 0:

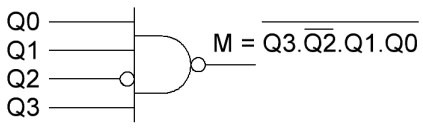
Mod-16:	0	1	2	3	4	5	6	7	8	9	10	**11**	12	13	14	15
Mod-11:	0	1	2	3	4	5	6	7	8	9	10	**0**	1	2	3	4

↑

The value 11 must be replaced by the value 0.

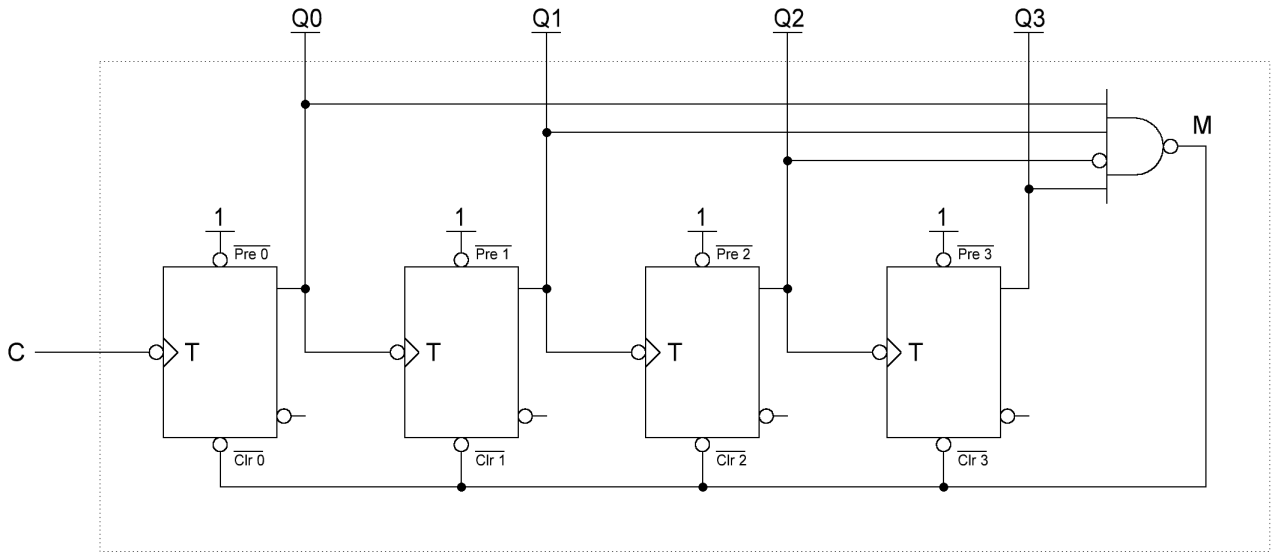
To set the 4-bit output of the counter to 0, we can use the asynchronous active-low clear inputs ($\overline{clear}$).

To detect the value 11 we can use a 4-input NAND gate. The output of the NAND gate is called *M*:

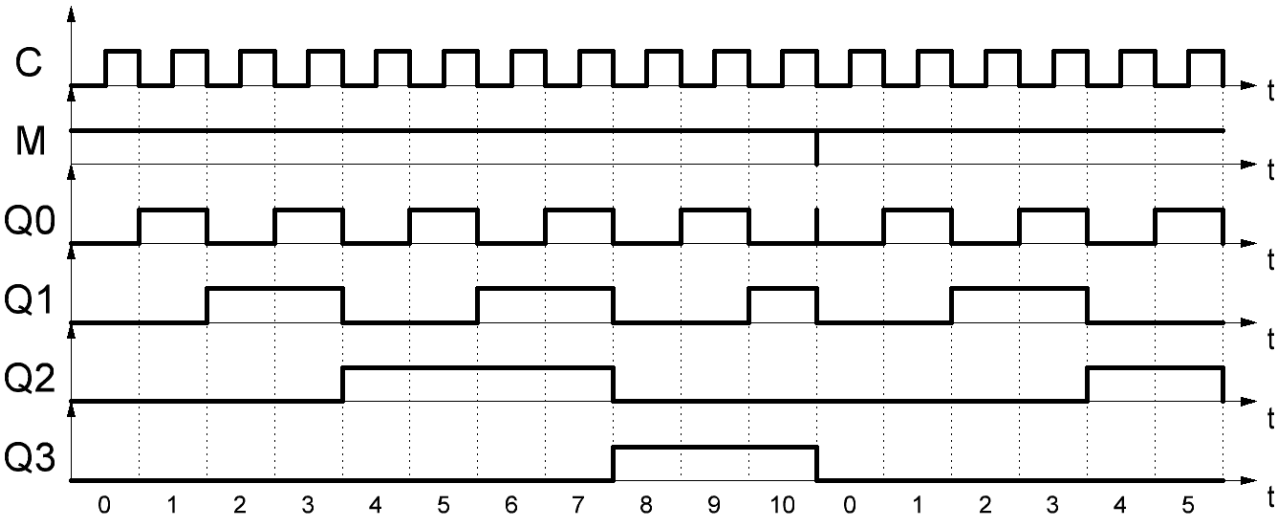


$M = 0$ if and only if $Q_{3:0} = 1011_2 = 11_{10}$.

If we use negative-edge-triggered flip-flops, this is what the circuit diagram should look like:



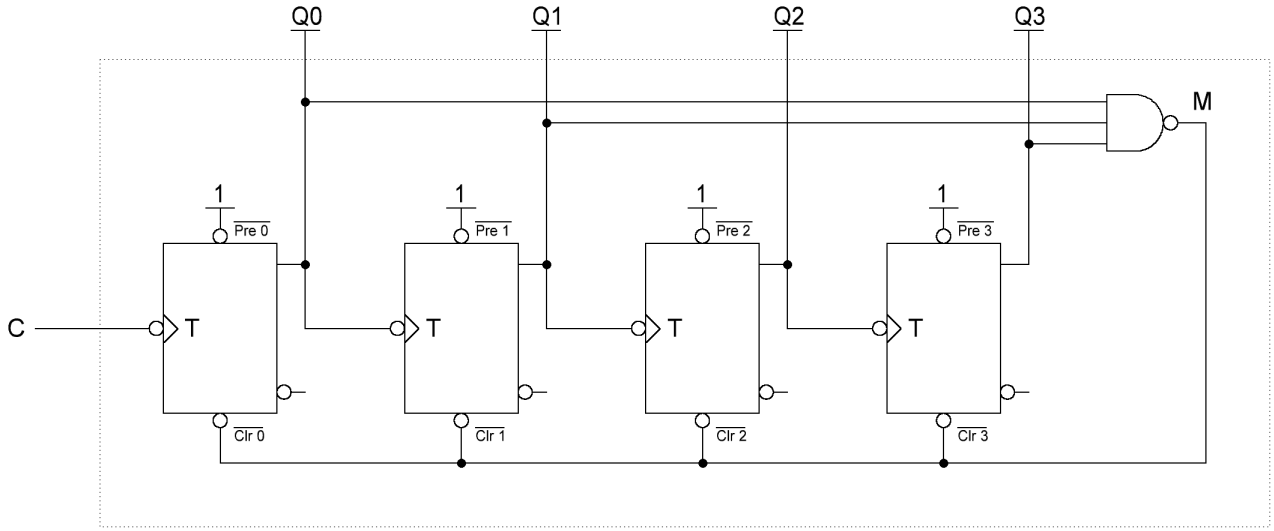
As soon as the counter reaches 11, *M* is set to 0. Therefore, the clear inputs become active and the outputs are reset to 0. Then, *M* goes back to 1, the clear inputs become inactive and the counter resumes counting.



The value 11 exists momentarily and will be long enough to reset all the flip-flops to 0. In practice, the reset pulse lasts a few nanoseconds and is related to the performance of the components.

We can optimize the detection of 11 slightly. It can be noticed that $Q3 = Q1 = Q0 = 1$ when the output is either 11 ($Q2 = 0$) or 15 ($Q2 = 1$). Since 15 never appears in our sequence, we can conclude that when $Q3 = Q1 = Q0 = 1$, the output of the counter is 11. In other words, the value of $Q2$ is irrelevant in this case.

After being optimized, here is what the circuit diagram should look like:



4.2. Down Counters

A truncated-sequence down counter is an *n*-bit down counter with a modulo lower than 2^n. It cycles through the natural binary sequence from a value lower than $2^n - 1$ to 0.

The first step in designing a truncated-sequence down counter is to build a full-sequence down counter. Then, the maximum value must be detected and replaced by the value of the modulo minus 1.

For instance, a modulo-11 down counter is a truncated-sequence counter because it requires a 4-bit output, which has 16 combinations, when only 11 are needed. In other words, it cycles through from 10 to 0 instead of cycling through from 15 to 0.

To design it, we have to build a modulo-16 down counter and replace the value 15 by the value 10:

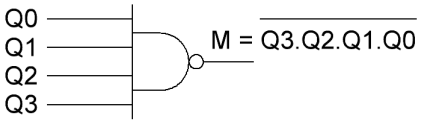
Mod-16:	10	9	8	7	6	5	4	3	2	1	0	**15**	14	13	12	11
Mod-11:	10	9	8	7	6	5	4	3	2	1	0	**10**	9	8	7	6

↑

The value 15 must be replaced by the value 10.

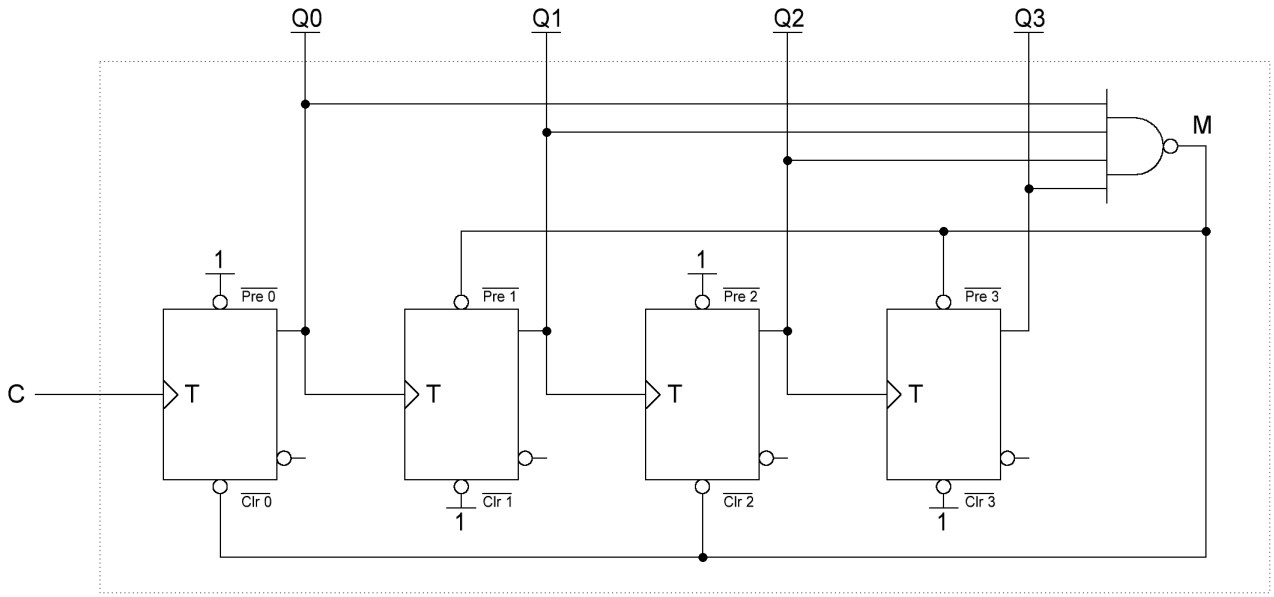
To set the 4-bit output of the counter to 10, we can use the asynchronous active-low preset and clear inputs ($\overline{preset}$ and $\overline{clear}$).

To detect the value 15 we can use a 4-input NAND gate. The output of the NAND gate is called *M*:

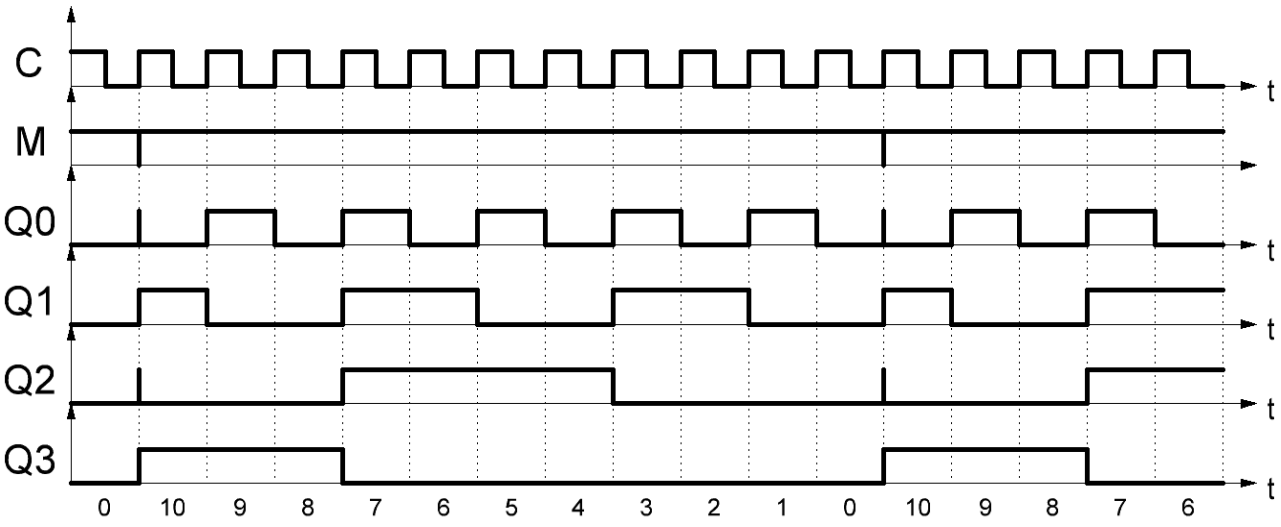


$M = 0$ if and only if $Q_{3:0} = 1111_2 = 15_{10}$.

If we use positive-edge-triggered flip-flops, this is what the circuit diagram should look like:



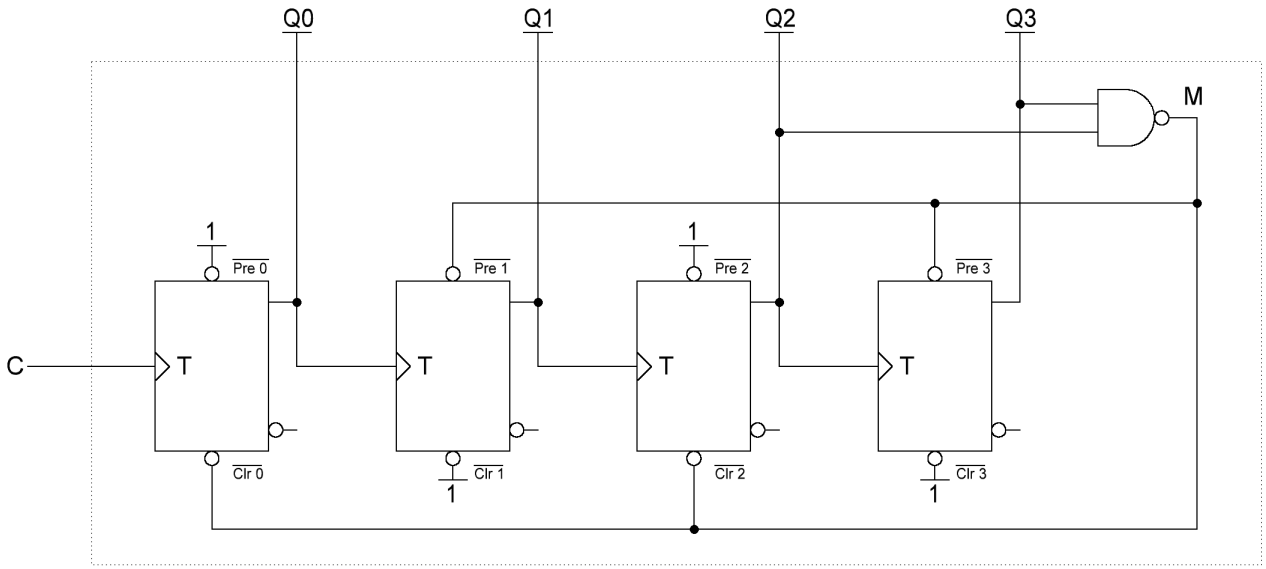
As soon as the counter reaches 15, *M* is set to 0. Therefore, *Q0* and *Q2* are reset to 0 and *Q1* and *Q3* are set to 1; that is to say the 4-bit output is set to 10 ($10_{10} = 1010_2$). Then, *M* goes back to 1, the preset and clear inputs become inactive and the down counter resumes counting.



The value 15 exists momentarily and will be long enough to set the 4-bit output to 10. In practice, the set and reset pulses last a few nanoseconds and are related to the performance of the components.

We can optimize the detection of 15 slightly. It can be noticed that *Q3* = *Q2* = 1 when the output is greater than or equal to 12. Since 12, 13 and 14 never appear in our sequence, we can conclude that when *Q3* = *Q2* = 1, the output of the counter is 15. In other words, the values of *Q0* and *Q1* are irrelevant in this case.

After being optimized, here is what the circuit diagram should look like:



5. Summary

- To design an asynchronous (up or down) counter you have to connect toggle flip-flops in series. The output of the first flip-flop triggers the output of the second flip-flop, the output of the second flip-flop triggers the output of the third flip-flop and so forth. This is why this type of counter is also called a ‘rippled counter’.

- In an up counter, an output (Q_n) toggles on each **falling edge** of the previous output (Q_{n-1}). This can be achieved by connecting Q_n or $\overline{Q_n}$ to the next clock input so that there is only one bubble on the wire.

- In a down counter, an output (Q_n) toggles on each **rising edge** of the previous output (Q_{n-1}). This can be achieved by connecting Q_n or $\overline{Q_n}$ to the next clock input so that there are either two bubbles or none at all on the wire.

- To design a truncated-sequence up counter, the value of the modulo must be detected and replaced by 0.

- To design a truncated-sequence down counter, the maximum value of the n-bit output ($2^n - 1$) must be detected and replaced by the value of the modulo minus 1.

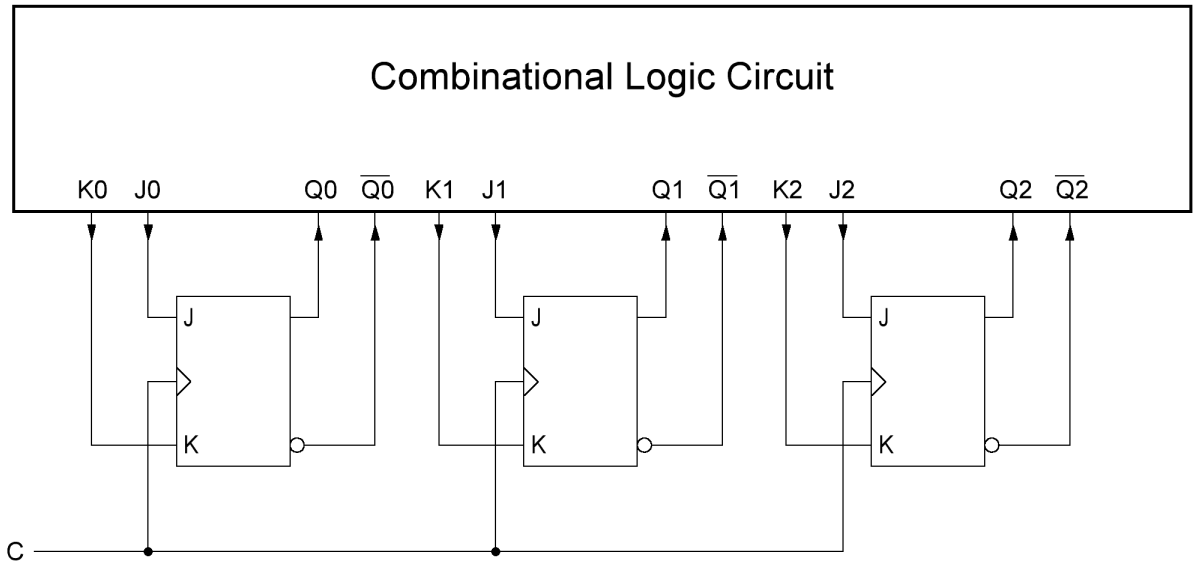
III. Designing Synchronous Counters

1. Principle

A synchronous counter is made up of several D or JK flip-flops that are all connected to the same clock signal. Therefore, every output changes at the same time.

The inputs and outputs of the flip-flops are connected to a combinational logic circuit, which is designed to generate a particular sequence. Any arbitrary sequence is possible.

For instance, let us take three JK flip-flops:



The outputs of the flip-flops are the inputs of the combinational logic circuit; and the outputs of the combinational logic circuit are the inputs of the flip-flops.

Designing the combinational logic circuit comes down to determining the expressions of its outputs (*K0, J0, K1, J1*, etc.) in terms of its inputs (*Q0, Q1*, etc.).

To do this, we need to use excitation tables.

2. Excitation Tables

An excitation table displays the input states required to have a given output transition.

2.1. The Excitation Table of a JK Flip-Flop

The excitation table of a JK flip-flop can be deduced from the truth table of a JK flip-flop.

Excitation Table

	Q(t)	Q(t+1)	J	K
①▸	0	0	0	Φ
②▸	0	1	1	Φ
③▸	1	0	Φ	1
④▸	1	1	Φ	0

Truth Table

	C	J	K	Q	
①▸	↑	0	0	q	◂④
①▸	↑	0	1	0	◂③
②▸	↑	1	0	1	◂④
②▸	↑	1	1	$\bar{q}$	◂③

To determine the states of *J* and *K* for each line, you have to answer the following question: what could the states of *J* and *K* be when the output changes from *Q(t)* to *Q(t+1)*?

Line ① : Transition of *Q* from 0 to 0
- no change ($J = 0, K = 0$)
- reset to 0 ($J = 0, K = 1$)

Line ② : Transition of *Q* from 0 to 1
- toggle ($J = 1, K = 1$)
- set to 1 ($J = 1, K = 0$)

Line ③ : Transition of *Q* from 1 to 0
- toggle ($J = 1, K = 1$)
- reset to 0 ($J = 0, K = 1$)

Line ④ : Transition of *Q* from 1 to 1
- no change ($J = 0, K = 0$)
- set to 1 ($J = 1, K = 0$)

2.2. The Excitation Table of a D Flip-Flop

The excitation table of a D flip-flop is very easy to obtain because a D flip-flop copies the value from the input *D* to the output *Q*.

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

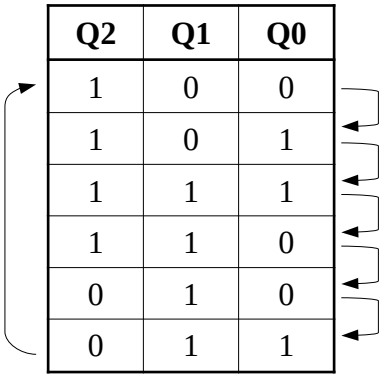
To determine the state of *D* for each line, you have to answer the following question: what could the state of *D* be when the output changes from *Q(t)* to *Q(t+1)*? Obviously: *Q(t+1) = D*.

3. Synchronous Counters Using JK Flip-Flops

We can design any arbitrary sequence as long as there are no identical combinations.

Let us take the example below:

We want to design a synchronous counter that cycles through the following sequence:



Q2	Q1	Q0
1	0	0
1	0	1
1	1	1
1	1	0
0	1	0
0	1	1

Using the excitation table, we can determine the inputs of the JK flip-flops.

Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
1	0	0	Φ	0	0	Φ	1	Φ
1	0	1	Φ	0	1	Φ	Φ	0
1	1	1	Φ	0	Φ	0	Φ	1
1	1	0	Φ	1	Φ	0	0	Φ
0	1	0	0	Φ	Φ	0	1	Φ
0	1	1	1	Φ	Φ	1	Φ	1

According to the excitation table of a JK flip-flop, when Q0 rises from 0 to 1, J0 is 1 and K0 is either 0 or 1.

From this table, we can now determine the most simplified expressions of *J0*, *K0*, *J1*, *K1*, *J2* and *K2* in terms of *Q0*, *Q1* and *Q2*.

- In an obvious way:
  - **K0 = Q1**
  - **J1 = Q0**
  - **J2 = Q0**

- Using Karnaugh maps:

J0	Q1 Q0 = 00	01	11	10
Q2 = 0	Φ	Φ	Φ	1
Q2 = 1	1	Φ	Φ	0

J0 = $\overline{Q1}$ + $\overline{Q2}$

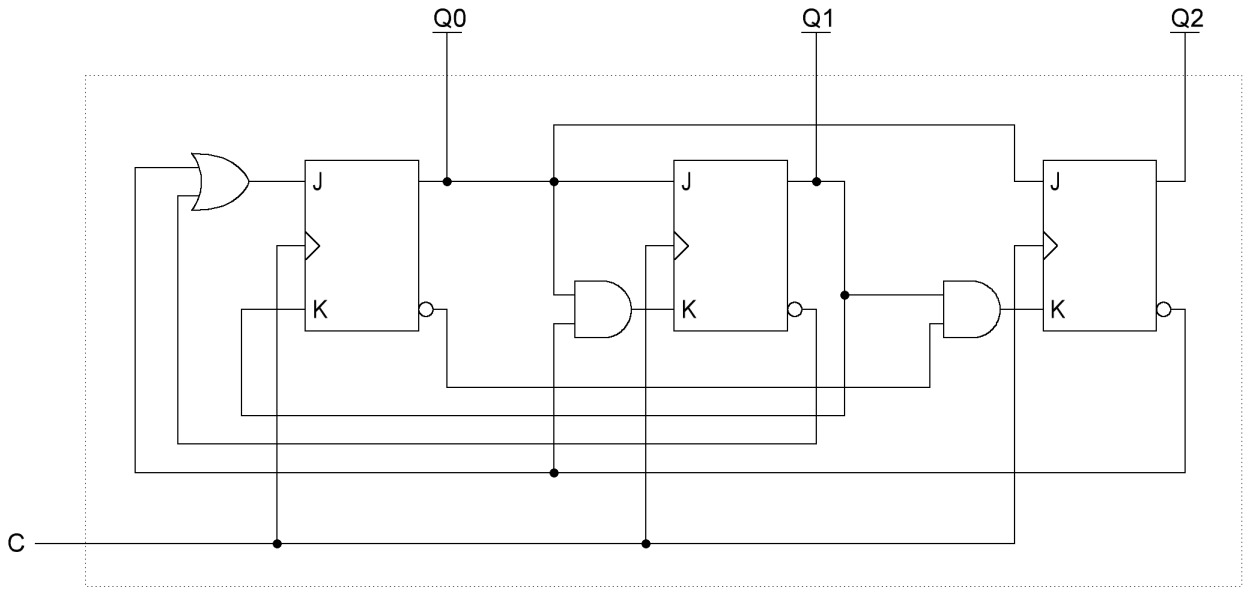
K1	Q1 Q0 = 00	01	11	10
Q2 = 0	Φ	Φ	1	0
Q2 = 1	Φ	Φ	0	0

K1 = Q0.$\overline{Q2}$

K2	Q1 Q0 = 00	01	11	10
Q2 = 0	Φ	Φ	Φ	Φ
Q2 = 1	0	0	0	1

K2 = $\overline{Q0}$.Q1

Here is what the circuit diagram should look like:



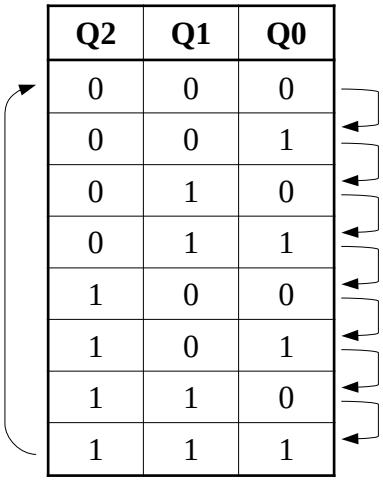
We can use either positive- or negative-edge-triggered flip-flop, depending on which edge we want to synchronize the counter with.

4. Synchronous Counters Using D Flip-Flops

The method of designing a synchronous counter using D flip-flops is very similar to the method using JK flip-flops. You just have to replace the *J* and *K* inputs by the *D* inputs. We can design any arbitrary sequence as long as there are no identical combinations.

Let us take the example below:

We want to design a modulo-8 synchronous counter that cycles through the following sequence:



Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Using the excitation table, we can determine the inputs of the D flip-flops.

Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

According to the excitation table of a D flip-flop, when Q0 rises from 0 to 1, D0 is 1.

From this table, we can now determine the most simplified expressions of *D0*, *D1*, and *D2* in terms of *Q0*, *Q1* and *Q2*.

- In an obvious way:
  - **D0 = $\overline{\mathbf{Q0}}$**

- Using Karnaugh maps:

D1	Q1 Q0 = 00	01	11	10
Q2 = 0	0	1	0	1
Q2 = 1	0	1	0	1

D1 = $\overline{Q0}$.Q1 + Q0.$\overline{Q1}$

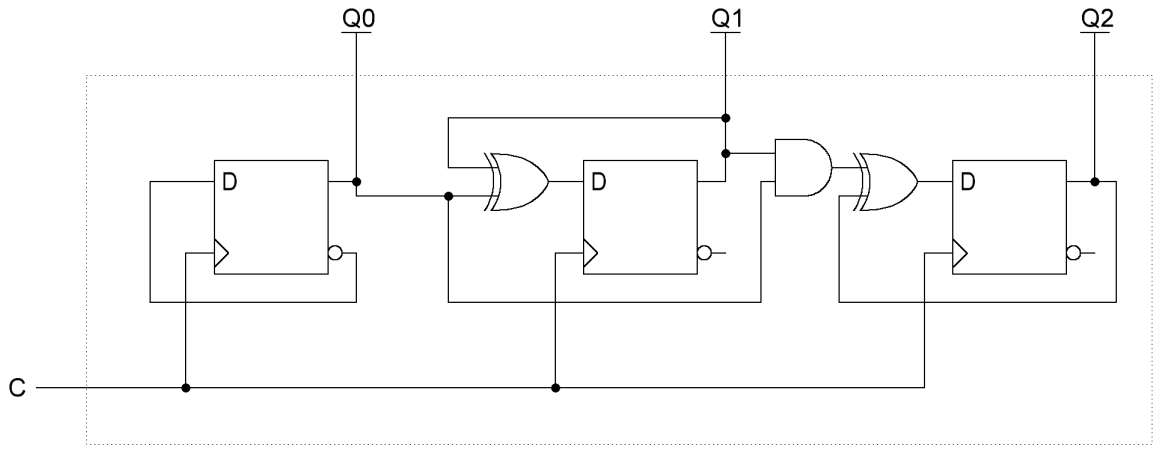
D2	Q1 Q0 = 00	01	11	10
Q2 = 0	0	0	1	0
Q2 = 1	1	1	0	1

D2 = $\overline{Q2}$.Q1.Q0 + Q2.$\overline{Q1}$ + Q2.$\overline{Q0}$

These expressions can be simplified once more using EXCLUSIVE OR.

- **D1 = Q0 ⊕ Q1**
- D2 = $\overline{Q2}$.Q1.Q0 + Q2.($\overline{Q1}$ + $\overline{Q0}$)
  D2 = $\overline{Q2}$.Q1.Q0 + Q2.($\overline{Q1.Q0}$)
  D2 = Q2 ⊕ (Q1.Q0)

Here is what the circuit diagram should look like:



We can use either positive- or negative-edge-triggered flip-flop, depending on which edge we want to synchronize the counter with.