Sequential Logic Notes

Andrew H. Fagg

Digital logic circuits composed of components such as AND, OR and NOT gates and that do not contain loops are what we refer to as *stateless*. In other words, the output that the circuit produces **only** depends on the current inputs, and does not depend on any previous inputs. These types of circuits are also referred to as *combinatorial circuits*.

However, as we design computing systems, we have a need for our circuits to maintain some record of what has happened before. For example, as you type letters into your favorite editor, you want your editor to remember what you have typed. In your computer, these data are stored in a variety of memories (including your disk drive and your Random Access Memory, or RAM).

In general, a circuit that maintains a memory of the past is referred to as a *sequential logic circuit*. At the heart of these circuits are logic devices called *latches* and *flip-flops* that store a single bit of data until the bit is overwritten at a later (often arbitrary) time.

The concepts outlined in the following sections will be important for our later understanding of several key ideas, including:

- binary numbers,
- software control of the configuration of a microcontroller,
- software manipulation of the voltage state of a microcontroller's output pin,
- the generation of precise timing signals in a microcontroller, and
- the implementation of basic mathematical operations in a microprocessor.

1 D-Type Flip-Flops

One logic device that stores data is called a *D-Type flip-flop*. Here, *flip-flop* refers to the fact that the stored bit can be "flipped" back and forth between a logic 0 and a logic 1 state. *D* refers to the type of input provided to the device, shown in Figure 1. The outputs of the

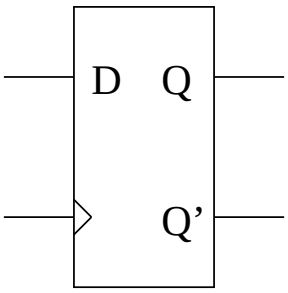

Figure 1: A D-Type Flip-Flop.

device are Q and Q', where Q is the value that has been stored by the flip-flop and Q' is the NOT of the stored value. The inputs of the device are the *data input* (D) and the *clock* (the triangular symbol).

Under most conditions, the D flip-flop does not change its output value (also called its *state*), even as the D input changes. However, when the clock signal transitions from logic 1 to logic 0, the value presented at D **just prior** to the transition is stored by the flip-flop. The outputs Q and Q' are then changed to reflect this stored value.

Figure 2 shows an example of how the flip-flop responds to various inputs. Here, the initial state of Q is given as logic 0 (and hence $Q' = 1$). In addition, the CLK and D signals are given. Q over time is determined by 1) its initial value, and 2) the input signals CLK and D. In particular, Q changes **only** when CLK transitions from logic 1 to 0 (these times are indicated by the dotted vertical lines). At the first transition, D is 1, which results in Q changing to 1. At the second clock transition, D is also 1. However, since $Q = 1$ already, we do not observe a change. Note that D temporarily flips to low between transitions 1 and 2. This does not affect the state of the flip-flop. At transition 3, D is at 0, which causes Q to change to 0. Finally, at transition 4, D is again at 0. As a result, Q remains in the 0 state.

2 Frequency Divider

The circuit shown in Figure 3 implements a form of *frequency divider* of the clock signal. In this circuit, the Q' output signal is used as input into D. Suppose that at a given time, the current state of the flip-flop is $Q = 0$. Because $Q' = 1$, D is also 1. When the clock input transitions from 1 to 0, the value presented to D is copied to Q. In this case, Q becomes 1.

Now, because $Q' = 0$, the value presented to the D input is also 0. When the clock again transitions from 1 to 0, this value is copied to Q. Hence, Q becomes 0. This process repeats

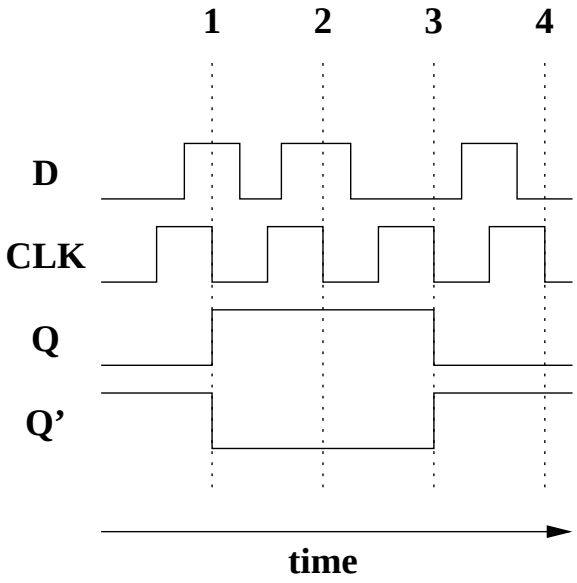

Figure 2: A Flip-Flop Example.

itself every two cycles of the clock signal. The corresponding timing diagram is shown in Figure 4. Note that Q and Q' are **always** opposites of one-another (this is a property of the flip-flop). Also note that the frequency of the signal on Q is exactly half that of the clock, CLK.

3 Shift Register

The circuit shown in Figure 5 is composed of three flip-flops and implements a form of *shift register*. Note that the CLK signal is distributed to all of the flip-flops. When this signal transitions from 1 to 0, the three flip-flops simultaneously copy the value that is presented to their respective D input to Q. Specifically, as a result of the clock transition, $X0$ will take on the value that was presented on Y, $X1$ will take on the value that was previously $X0$, and $X2$ will take on the value that was previously $X1$.

As an example, consider the timing diagram in Figure 6. Here, the initial state is $X2 = 0$, $X1 = 0$ and $X0 = 0$. The CLK and Y values are given. The subsequent states of $X2$, $X1$ and $X0$ are determined by these input signals and by their initial states. On the first transition of CLK from 1 to 0, the value of Y is copied to $X0$, as shown by the upper-left dashed arrow (at coordinate $[Y, 1]$). This results in a change of $X0$ from 0 to 1. Likewise,

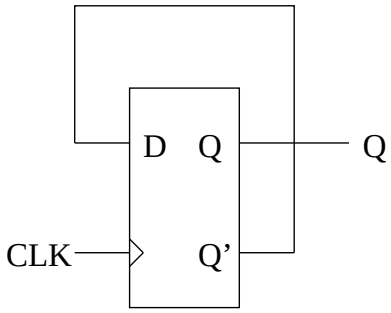

Figure 3: Frequency Divider Circuit.

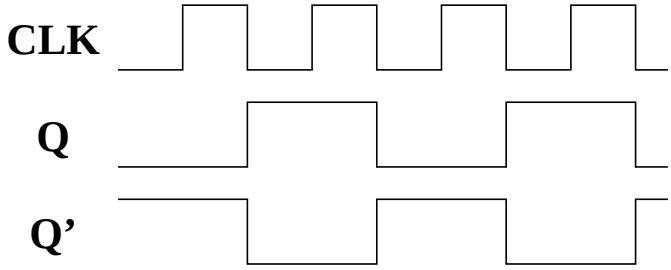

Figure 4: Timing digram for the frequency divider circuit. Note that the initial state of Q (and hence Q') is given as an initial condition. The subsequent states are determined by this initial state, the CLK signal, and the circuit itself.

the old value of $X0$ is copied to $X1$ (arrow at coordinate $[X1, 1]$), leaving $X1$ in a state of 0. Finally, the old value of $X1$ is copied to $X2$ (coordinate $[X2, 1]$), leaving $X2$ in a state of 0.

On the second transition, the same copy operations are performed: Y (which is now 0), is copied to $X0$, the old value of $X0$ (1) is copied to $X1$, and the old value of $X1$ (0) is copied to $X2$. This same pattern continues through clock transitions 3 and 4.

Note that one can interpret the X's as individual bits in a 3-bit binary number: $X2, X1, X0$, or just X. Looking again at Figure 6, the initial value of X is binary 000 (decimal 0). After transition 1, the value is 001 (decimal 1). After transition 2, the value is 010 (decimal 2). After transitions 3 and 4, the value is 100 (decimal 4) and 001 (decimal 1), respectively.

Because a bit value is copied at each transition from one Xi to the next one $(i + 1)$, we refer to this as a *left shift register*. When $Y = 0$, each transition adds an extra 0 on the right-hand-side of the binary value. This operation is equivalent to multiplying the binary

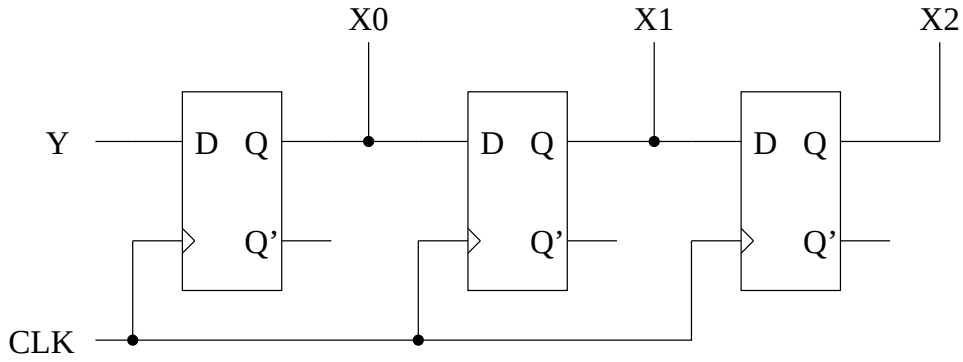

Figure 5: Shift register circuit.

value by 2.[1] Note that when the binary value is larger than or equal to 4 (when $X2 = 1$), multiplying by 2 effectively produces a *carry* that is lost by this circuit. For example, when $X = 100$, shifting left leaves us with $X = 000$ (as opposed to $X = 1000$). This is an inherent limitation of all digital representations of integers: there is always a maximum value that we can represent, and is determined by how many bits that are used to represent the number and whether the number can also capture negative values.

One can also design a complimentary circuit that shifts the bits from left to right. Such a *right shift register* implements the mathematical operation of dividing by 2 and dropping the remainder.

4 Binary Counters

Binary counter circuits implement the mathematical operation of adding 1 to a value. Table 1 lists the first eight binary values, starting at 0. Our binary counter circuit will start at some initial condition (say, row i). After the clock transition from 1 to 0, the value should correspond to row $i+1$. When $B = 111$, the next integer in the sequence would be $B = 1000$. However, in this example, we only have 3 bits to work with and the carry to the 4^{th} bit is lost. Hence, the next value following $B = 111$ is $B = 000$.

If we consider time as moving from one row to another at a regular pace, then $B0$ exhibits a signal at a particular frequency. Looking at bit $B1$, this signal is half the frequency of $B0$. Furthermore, $B1$ changes state every time $B0$ changes from 1 to 0. The same is true for $B2$:

[1] An analogy: in decimal, or *base 10*, adding a 0 to the right-hand-side of a number is equivalent to multiplying it by 10.

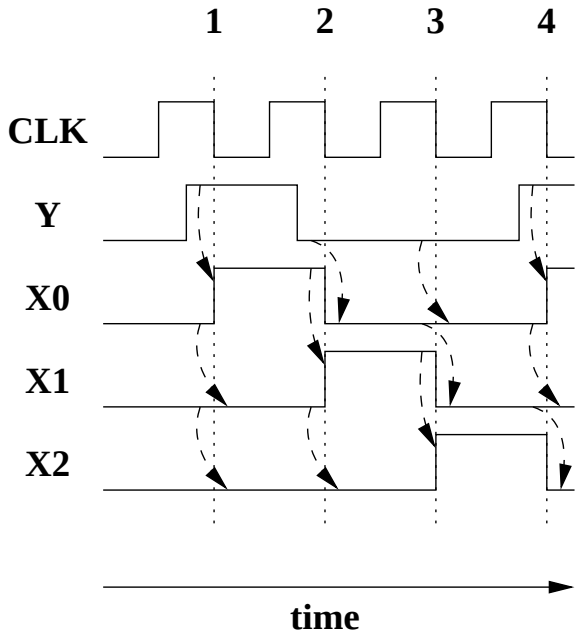

Figure 6: Shift register timing example. The dashed arrows show how the bit values in one time period affect the bit values in the next period.

it operates at half the frequency of $B1$ and it changes state every time $B1$ changes from 1 to 0.

4.1 Ripple Counter

The *ripple counter* exploits this frequency division property of the bits in the counting sequence. Figure 7 shows a 3-bit ripple counter. Here, the CLK signal is only connected to the clock input of the leftmost flip-flop (that stores $B0$). As we noted about the counting sequence, $B1$ will flip state every time $B0$ transitions from 1 to 0. Hence, we can use $B0$ as the clock input to the middle flip-flop (that stores $B1$). Likewise, $B1$ can be used as the clock input to the rightmost flip-flop ($B2$).

Figure 8 shows the timing behavior of our circuit. In this example, the initial state of $B = 000$ is given. The evolution of this state as a function of time is determined by this initial state and the CLK signal. The *ripple counter* name comes from the fact that a *carry* (in the mathematical sense) will "ripple down" the circuit from left to right. When bit i is 1 and its clock input transitions from 1 to 0, it produces a carry that causes 1 to be added to

Binary Value			Decimal Value
$B2$	$B1$	$B0$	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Table 1: Binary values from 0 to 7.

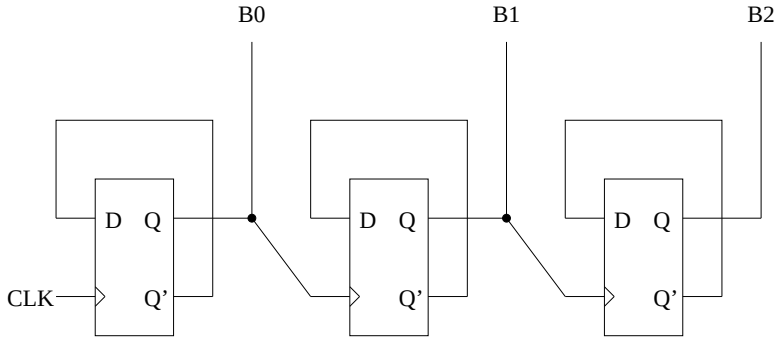

Figure 7: Ripple Counter Circuit.

bit $i+1$. In turn, this may result in a carry that is passed on to bit $i+2$. This rippling effect is shown in the small delay in response by $B1$ with respect to $B0$ and of $B2$ with respect to $B1$.

Although the ripple counter design is very simple, a possible disadvantage is the fact that the flip-flops do not all change state at the same time. In fact, bit i can only change state **after** $i-1$ changes state. Although these delays may be on the order tens of nanoseconds, for some applications, they can be unacceptable. This brings us to a different design – one in which all of the bits change state at the same time.

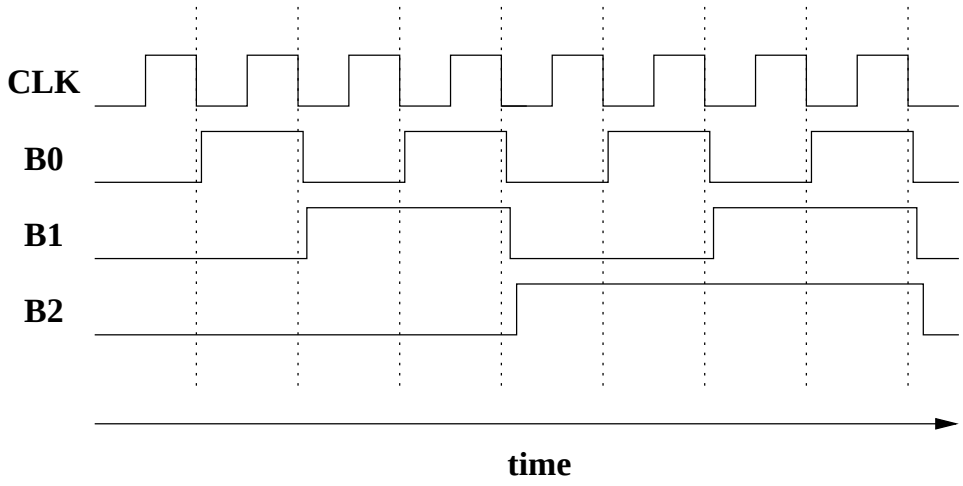

Figure 8: Timing diagram for the ripple counter circuit.

4.2 Sequential Counter

One possible implementation of a 3-bit *sequential counter* is shown in Figure 9. Here (as in many sequential circuits), the clock inputs to all of the flip-flops receive the same CLK signal. This will ensure that the flip-flops change state at exactly the same time. Analyzing the temporal behavior of sequential circuits is a two-step process. First, assume a constant state for all of the flip-flops and no change in state of the clock signal. Given the flip-flop state, and possibly other external inputs (though none are shown in this circuit), we compute the values of each of the D inputs. Second, given the D inputs, we assume that CLK transitions from logic 1 to 0, resulting in a copy of the D values to Q. This process is then repeated.

As an example, assume that the initial state of the counter is $B = 000$ in Figure 9. This implies that $D2 = 0$, $D1 = 0$ and $D0 = 1$. On the subsequent clock tick (from 1 to 0), this results in a new counter state of $B = 001$. In this new state, $D2 = 0$, $D1 = 1$ and $D0 = 0$. On the next clock tick, the state then becomes $B = 010$.

Assuming an initial state of $B = 011$, then $D2 = 1$, $D1 = 0$ and $D0 = 0$. Hence, the next state is $B = 100$. In this state, $D2 = 1$, $D1 = 0$ and $D0 = 1$. The next state is $B = 101$.

Assuming an initial state of $B = 111$, then $D2 = 0$, $D1 = 0$ and $D0 = 0$. Hence, the next state is $B = 000$.

Assuming an initial counter state of $B = 000$, Figure 10 shows the corresponding timing diagram. This diagram is virtually identical to that of the ripple counter, except that there

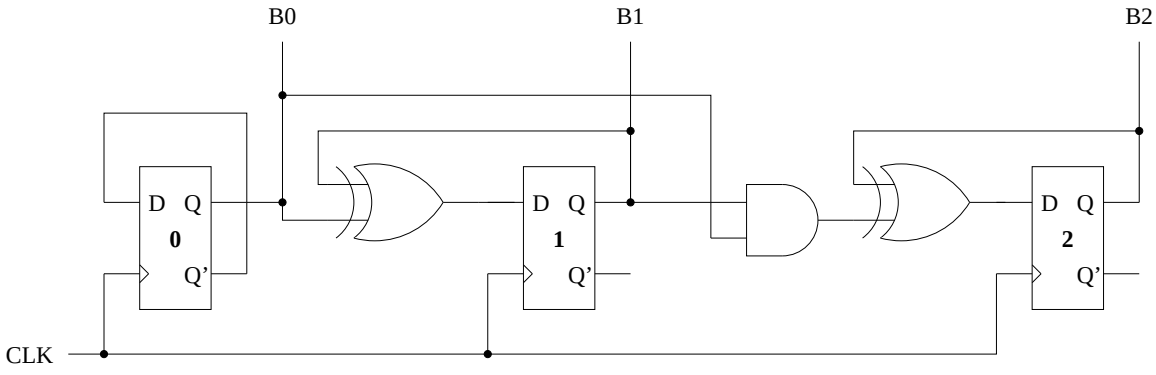

Figure 9: Sequential Counter Circuit.

is no delay in response between the bits. Note, however, there is a real delay between the CLK signal changing from 1 to 0 and the response of the flip-flops. However, this delay is not shown for convenience.

The behavior of a sequential circuit can be analyzed more formally by constructing a truth table that describes what happens in every state. This table will have one row for every possible bit state and external input (an example of an external input is coming in the next section). For each row, we show the D value for each flip-flop.

Table 2 shows the truth table that corresponds to the circuit shown in Figure 9. Note that we have already discussed some of these rows above. The truth table captures a complete picture of each of the possible states and what the next state will be, as encoded by the D inputs. Hence, this table gives us a direct way of reading off the sequence of states given an initial state. This is something that we can do without constructing a timing diagram. For this example, $B = 011$ is followed by $B = 100$, which is in turn followed by $B = 101$.

4.3 Up-Down Counter

An *up-down counter* is a logical device that will either add or subtract one from a stored value, depending upon the value of an external control input. Figure 11 illustrates the implementation of a 3-bit up-down counter. As before, $B2$, $B1$ and $B0$ represent the values that are stored by the counter (and are also outputs from the circuit). Control signal X determines whether one is added ($X = 0$) or subtracted ($X = 1$) from the stored value.

While this circuit is more complicated that the previous one, we can get a handle on how it behaves by focusing on what each of the D inputs is as a function of the current state and of the external output X. We will do this by constructing a truth table that contains a total

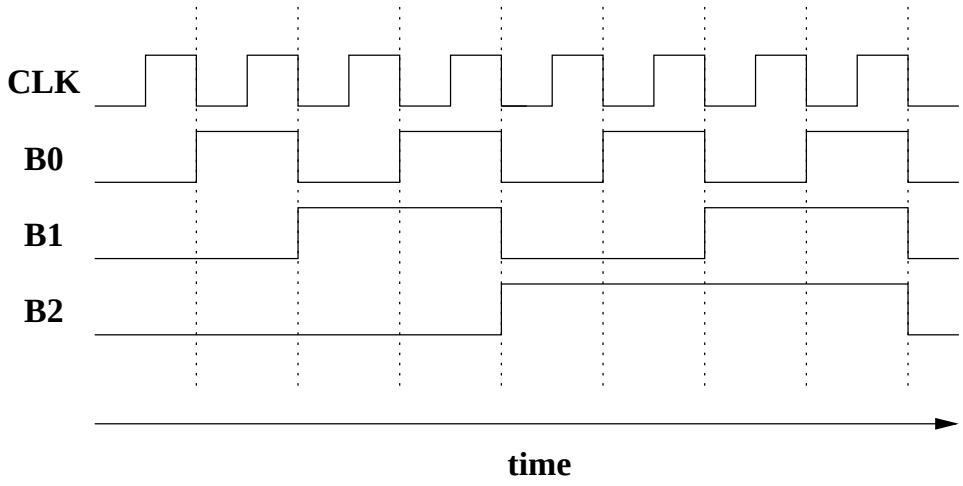

Figure 10: Timing diagram for the sequential counter circuit.

of 16 rows (one external input + 3 state bits implies 2^4 cases). The completed truth table is shown in Table 3. However, the reader should confirm the correct values of the D's for each of the 16 cases.

First, we will focus on $B0$. Note that $D0$ is directly connected to $Q0'$. Hence, the next state of this flip-flop will **always** be the opposite of its previous state.

For $B1$, note that $D1$ only receives inputs from $Q0$, $Q1$ and X. Hence, we have only 8 cases to consider, since the circuit will do the same thing no matter the state of $Q2$. As you analyze this part of the circuit, a useful pair Boolean rules is: $X \oplus 0 = X$ and $X \oplus 1 = \bar{X}$.

Finally, for $B2$, we must consider all 16 cases separately.

Given Table 3, we can then determine the sequence of states that will be produced (one state for each clock tick) based on an initial state and the control input X. Suppose an initial state of $B = 101$ and $X = 0$. The next states will be $B = 110$, $B = 111$ and $B = 000$. In contrast, assume an initial state of $B = 010$ and $X = 1$. The next states will be $B = 001$, $B = 000$ and $B = 111$.

State			Next State		
$B2$	$B1$	$B0$	$D2$	$D1$	$D0$
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

Table 2: Truth table for the sequential counter circuit.

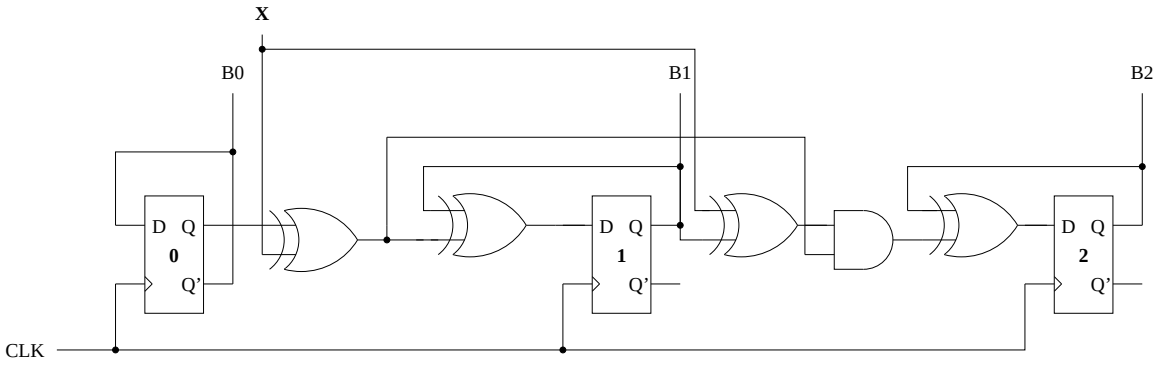

Figure 11: A sequential up-down counter implementation.

Input	State				Next State			
X	$B2$	$B1$	$B0$	Value	$D2$	$D1$	$D0$	Value
0	0	0	0	0	0	0	1	1
0	0	0	1	1	0	1	0	2
0	0	1	0	2	0	1	1	3
0	0	1	1	3	1	0	0	4
0	1	0	0	4	1	0	1	5
0	1	0	1	5	1	1	0	6
0	1	1	0	6	1	1	1	7
0	1	1	1	7	0	0	0	0
1	0	0	0	0	1	1	1	7
1	0	0	1	1	0	0	0	0
1	0	1	0	2	0	0	1	1
1	0	1	1	3	0	1	0	2
1	1	0	0	4	0	1	1	3
1	1	0	1	5	1	0	0	4
1	1	1	0	6	1	0	1	5
1	1	1	1	7	1	1	0	6

Table 3: Truth table for the sequential counter circuit. *Value* is the decimal equivalent of the corresponding binary value.