Theory of Logic Circuits

Laboratory manual

Exercise 10

Counters

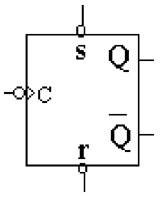
© 2008 Urszula Stańczyk, Piotr Czekalski (edt.)

1. Introduction

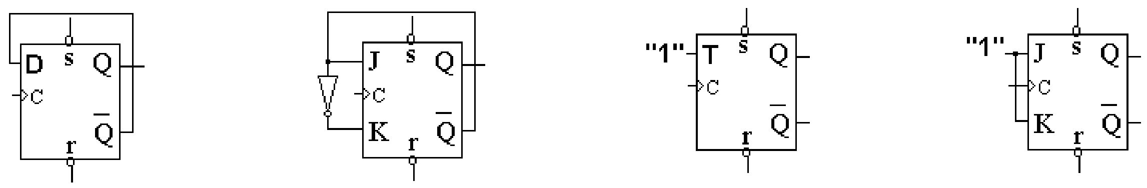
Counters are such circuits that count occurrences of clock events or generate certain patterns. These circuits may be either asynchronous or synchronous.

2. Asynchronous counters

In asynchronous counters the control inputs of flip-flops are not driven by the same signal source.



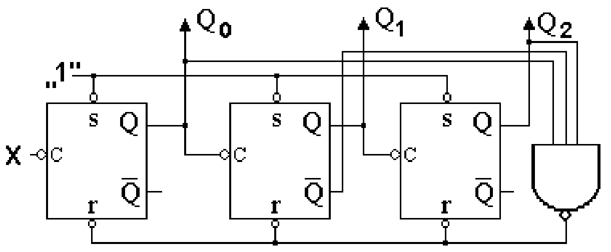
The common element used in a design of such a counter is a simplified symbol of a flip-flop. By such symbol we understand any flip-flop working as a frequency divider by 2. It may be any of the following possibilities.

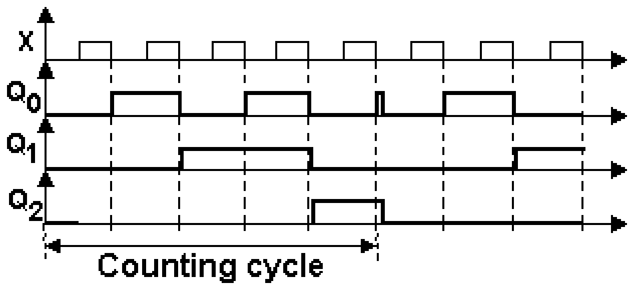


Such elements connected together into a cascade, in which one triggers another, create an asynchronous counter (such counters are also called serial counters). The number of states of such a counter is 2^N, where N equals to the number of flip-flops used in a construction. The number of states may be reduced by detecting some state on the counter outputs and resetting the counter afterwards, as shown in the example below.

2.1. Example

Complete a timing chart for a circuit given and find the number of its states in a counting cycle.





The counter is mod 5 as it counts from zero to four, detects 5 on its outputs and resets itself.

3. Synchronous counters

In synchronous counters the system clock drives the control inputs to all the flip-flops in the design and as a result all flip-flops change their state at the same time (and that is why such counters are also called parallel).
To design a synchronous counter we may use one of two methods. The first one bases on the fact that a synchronous counter is a synchronous sequential circuit and as such it may as well be designed using Huffman's method. The second method is called a present state-next state table method. Both will be shown in example below.

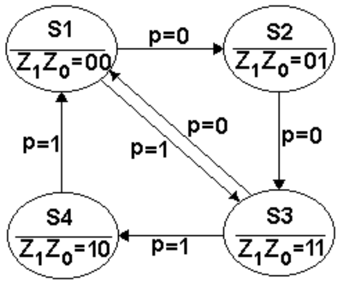
3.1. Example

Design a parallel synchronous programmable counter, which for a programming signal p=0 counts in the code 00→01→11, while for p=1 it counts 11→10→00. Present a solution in a form of the input excitation functions for JK and T flip-flops.

We will solve this task in three versions:

a) Without illegal state recovery,
b) With synchronous illegal state recovery,
c) With asynchronous illegal state recovery.

For version a) we will use Huffman's method. In the first step it is best to draw a state diagram for a circuit.



Then we change this graph into a flow map as follows.

	p = 0	1	Z_1Z_0
S1	S2	S3	00
S2	S3		01
S3	S1	S4	11
S4		S1	10

As we need all states we cannot reduce or merge this table.

Then the next step is to encode states. The best optimal way is to do it, if possible, with the states of outputs, then we will not need any additional output combinational block.

Notice here, that for our example the states of outputs correspond to the natural way of describing a Karnaugh map (Gray code) but it does not, however, happen always.
We assume the codes: S1 – (00) Q_1Q_0, S2 – (01) Q_1Q_0, S3 – (11) Q_1Q_0, S4 – (10) Q_1Q_0 and receive a composite Karnaugh map, from which we get the output functions $Z_1=Q_1$ $Z_0=Q_0$ and two Karnaugh maps for Q_1 and Q_0.

	p		
$Q_1^tQ_0^t$	0	1	Z_1Z_0
00	01	11	00
01	11		01
11	00	10	11
10		00	10

$Q_1^{t+1}Q_0^{t+1}$

	p	
$Q_1^tQ_0^t$	0	1
00	0	1
01	1	
11	0	1
10		0

Q_1^{t+1}

	p	
$Q_1^tQ_0^t$	0	1
00	1	1
01	1	
11	0	0
10		0

Q_0^{t+1}

Then using a method of a flip-flop excitation table and variations of the maps above, we get the maps for J_1K_1 and T_0.

	p	
$Q_1^tQ_0^t$	0	1
00	0-	1-
01	1-	
11	-1	-0
10		-1

J_1K_1

	p	
$Q_1^tQ_0^t$	0	1
00	1	1
01	0	
11	1	1
10		0

T_0

$$J_1 = Q_0 + p$$
$$K_1 = \overline{Q_0} + \overline{p}$$
$$T_0 = Q_0 \cdot Q_1 + \overline{Q_1} \cdot \overline{Q_0} = \overline{Q_1 \oplus Q_0}$$

Our counter has two illegal states. For p=0 it is Z_1Z_0=10, for p=1 it is Z_1Z_0=01. As stated above, the recovery from illegal states may be done in an asynchronous or synchronous way.
Asynchronous illegal state recovery assumes the use of the asynchronous setting and resetting inputs of flip-flops (that is why it is called “asynchronous”). Once the illegal state is detected, appropriate asynchronous inputs are activated and the counter returns to the legal state, usually the first one in the counting cycle.
Notice here, that we do not always reset the flip-flops. We reset the counter – i.e. we set its first state, but it does not have to be “00”.

For version b) – i.e. the counter with synchronous illegal state recovery we will use the present state-next state table method.

p	$Q_1^tQ_0^t$	Q_1^{t+1} Q_0^{t+1}
0	00	01
	01	11
illegal	10	00
	11	00
1	00	11
illegal	01	11
	10	00
	11	10

In this method we create the table where on the left we write all internal states of a counter, on the right the appropriate next states. We know them when we assume that the internal states correspond directly to the states of counter outputs.
As we are supposed to ensure the synchronous illegal state recovery, for each one of such states we write as a next state the first one in the counting cycle, i.e. for p=0 it is Q_1Q_0=00, p=1 it is Q_1Q_0=11.

Basing on this, we may create Karnaugh maps for Q_1^{t+1} and Q_0^{t+1}, and then, using one of known methods receive the input excitation tables for flip-flops.

There is, however, another possibility, which makes the process of designing faster. In this approach the present state-next state table is created with a slight but a significant difference. First, on the left we put all internal states of a counter but in order in which they should appear on the output, with illegal states at the end of a list. Once we have it, we should notice that each row of the table is a next state for the previous row, so we don't need any additional columns for writing the next states. Then knowing the succession of states and the types of flip-flops we may find the truth tables for them as follows.

p	Q_1Q_0	J_1K_1	T_0
0	00	0-	1
	01	1-	0
	11	-1	1
illegal	10	-1	0
1	11	-0	1
	10	-1	0
	00	1-	1
illegal	01	1-	0

For the current state 00 the next is 01, so the first flip-flop should remain 0 (J_1K_1=0-) and the second should be change its state to 1 (T_0=1). For the current state 01 the next is 11, so the first flip-flop should be set (J_1K_1=1-) and the second should remain at 1 (T_0=0), and so on.

p \ $Q_1^tQ_0^t$	00	01	11	10
0	0-	1-	-1	-1
1	1-	1-	-0	-1

J_1K_1

p \ $Q_1^tQ_0^t$	00	01	11	10
0	1	0	1	0
1	1	0	1	0

T_0

When we create Karnaugh maps for J_1K_1 and T_0, and compare them to those from version a) of our example we should notice that the only difference is that these maps have cells corresponding to illegal states filled, but actual functions in their minimal forms are the same.
It means that even the version without self-correction has such synchronous mechanism in itself. It does not, however, happen always.

As synchronous illegal state recovery requires a whole clock cycle to make it work, sometimes the asynchronous version is preferred, as it acts immediately - once the illegal state is detected it activates the proper asynchronous inputs and sets the legal state.

For c) version of our counter we take the part of a design done in an a) version – i.e. the counter without self-correction mechanism and to this we need only to add the asynchronous illegal state recovery.
If the number of illegal states is small, the solution may be achieved just by considering which inputs and when should be activated. When this number is greater, it is best to create a Karnaugh map in order to get an optimal solution.
In our example there are only two illegal states. For p=0 it is Q_1Q_0=10, for p=1 it is Q_1Q_0=01. In the first case we should return to the state "00", in the second to "11". It means that in both cases we do not need to change the state of Q_0, so its sr inputs should be left unconnected. As for Q_1, we need to keep in mind that these asynchronous inputs are active with "0", so when we want to set a flip-flop we need "0" on s input, if we want to reset a flip-flop we need "0" on r input.

For both illegal states we have to detect them and this is best done with AND function, for activity with “0” with NAND function. So:

$\overline{s_1} = \overline{p \cdot \overline{Q_1} \cdot Q_0}$ and $\overline{r_1} = \overline{\overline{p} \cdot Q_1 \cdot \overline{Q_0}}$

which corresponds to setting the flip-flop when p=1 and Q_1Q_0=01 and resetting the flip-flop when p=0 and Q_1Q_0=10.

Notice here, that our counter in this last version has both asynchronous and synchronous mechanisms of illegal state recovery, but usually it is not the case.

There is one more group of counters that needs pointing out. These counters have some states on the outputs repeated within one counting cycle. For example a counter with states 00→01→11→01. As for the current state “01” the next should be in the first case “11”, in the second “00”, it is not possible to get a solution in a straightforward way. In order to make it possible we have to add one auxiliary state variable – i.e. a flip-flop that will remember in which stage of a counting cycle the counter is in. Then the internal states may be as follows: 000→001→111→101. When sending on the output only two less significant bits we get the required sequence.
The most significant bit is “0” to the first occurrence of the repeated state “01”, then changes to “1” for the rest of the counting cycle.
Of course, it is only one of many possible solutions.

4. Tasks to be performed during laboratory

1. Design a reverse counter mod 3 in three versions:
    a. Without self-correction mechanism,
    b. With the synchronous illegal state recovery,
    c. With the asynchronous illegal state recovery.
2. Design a programmable counter, which for a programming signal p=0 counts in the code 00→11→01, for p=1 counts 11→00→10, in three versions:
    a. Without self-correction mechanism,
    b. With the synchronous illegal state recovery,
    c. With the asynchronous illegal state recovery.
3. Design a counter with states 00→01→00→10 with the synchronous illegal state recovery.
4. Design an asynchronous counter mod 6.

For all tasks given, the available logical elements are D and JK flip-flops, 2- and 4-input NAND gates, 2-input NOR gates, 2-input XOR gates.

5. Instructions to follow

1. Solve all tasks before the exercise.
2. Implement the circuits specified by your supervisor (using given elements).
3. Present working circuits to your supervisor for acceptance.