

How to Design a Sequential Counter

Charles B. Cameron, CDR, USN

October 31, 2002

Contents

1	Introduction	2
2	Specifying the Count Sequence	2
3	Generating a State Table	2
4	Minimizing the Logic Using Karnaugh Maps	4
5	Implementing the Design Using D-Flip-flops	4
6	Implementing the Design Using JK-Flip-flops	5
7	Implementing the Design Using T-Flip-flops	7
8	Comparison of the Three Implementations	9

List of Figures

1	Implementation of the Synchronous Counter Using D-Flip-flops .	4
2	Implementation of the Synchronous Counter Using JK-Flip-flops.	6
3	Implementation of the Synchronous Counter Using T-Flip-flops. .	8

List of Tables

1	Initial State Table	2
2	State Table Including Next States	3
3	State Table Including D-Flip-flop Inputs Required	3
4	Karnaugh Maps for D-Flip-flop Implementation	4
5	Activation Table for a JK-Flip-flop	5
6	State Table Including JK-Flip-flop Inputs Required	5
7	Karnaugh Maps for Implementing f Using JK-Flip-flops	6
8	Activation Table for a T-Flip-flop	7

9	State Table Including T-Flip-flop Inputs Required	7
10	Karnaugh Maps for Implementing f Using T-Flip-flops	8

1 Introduction

The purpose of this document is to illustrate the design process for implementing a sequential circuit to generate an arbitrary sequence of output numbers. The restriction imposed for simplicity is that a number appear only once in the sequence and that the sequence is repeated indefinitely. We also illustrate a means for initializing the counter to an arbitrary initial state using switches or pushbuttons.

2 Specifying the Count Sequence

A desired sequence can come from nearly any approach. From the point of view of this article, it is completely arbitrary. Our example will specify a 3-bit number sequence which, in decimal, is 5, 7, 3, 2, and 6, repeated indefinitely. In binary this is 101, 111, 011, 010, and 110.

3 Generating a State Table

To start with, write all the numbers from 0 to $2^n - 1$ in a column and place their binary equivalents beside them, as shown in Table 1. We shall label the binary bits A , B , and C . These represent the bits of the 3-bit number from most significant bit to least significant bit. For some of these rows, we care

Current State			
	A	B	C
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

Table 1: Initial State Table

what state the counter goes to next. We'll assume that for any states not in the list 5, 7, 3, 2, and 6 that we do not care about what follows them. This rests on the belief that these states will never occur. We can cause them not to occur by initializing the counter to one of the states in the list—say, 5 or 6—and trusting

that no gamma rays will strike the counter and put it into one of the forbidden (undesirable) states.

The next step in the process, then, is to expand our table to specify the next states, as shown in Table 2. For example, looking at line 5, we see that the next state should be 7, or 111 in binary. Similarly, in row 7 we see that the next state should be a 3, or 011 in binary.

Current State				Next State			
	A	B	C		A	B	C
0	0	0	0	×	×	×	×
1	0	0	1	×	×	×	×
2	0	1	0	6	1	1	0
3	0	1	1	2	0	1	0
4	1	0	0	×	×	×	×
5	1	0	1	7	1	1	1
6	1	1	0	5	1	0	1
7	1	1	1	3	0	1	1

Table 2: State Table Including Next States

The next step is to add in the signals needed to force the flip-flops to assume the desired values. This is particularly easy to do with D-flip-flops since the D -input of a D-flip-flop is exactly the same as the value we want the flip-flop to take on.

Current State				Next State				Flip-flop Controls		
	A	B	C		A	B	C	D_A	D_B	D_C
0	0	0	0	×	×	×	×	×	×	×
1	0	0	1	×	×	×	×	×	×	×
2	0	1	0	6	1	1	0	1	1	0
3	0	1	1	2	0	1	0	0	1	0
4	1	0	0	×	×	×	×	×	×	×
5	1	0	1	7	1	1	1	1	1	1
6	1	1	0	5	1	0	1	1	0	1
7	1	1	1	3	0	1	1	0	1	1

Table 3: State Table Including D-Flip-flop Inputs Required

We now have a specification for the control signals we need. There are three functions to implement: D_A , D_B , and D_C .

4 Minimizing the Logic Using Karnaugh Maps

We need three Karnaugh maps, one for each of the functions D_A , D_B , and D_C .

		C	
D_A	0	1	
00	×	×	
AB 01	1	0	
11	1	0	
10	×	1	
$D_A = \overline{B} + \overline{C}$			

		C	
D_B	0	1	
00	×	×	
AB 01	1	1	
11	0	1	
10	×	1	
$D_B = \overline{A} + C$			

		C	
D_C	0	1	
00	×	×	
AB 01	0	0	
11	1	1	
10	×	1	
$D_C = A$			

Table 4: Karnaugh Maps for D-Flip-flop Implementation

5 Implementing the Design Using D-Flip-flops

An implementation for the equations derived in Table 4 appears in Figure 1.

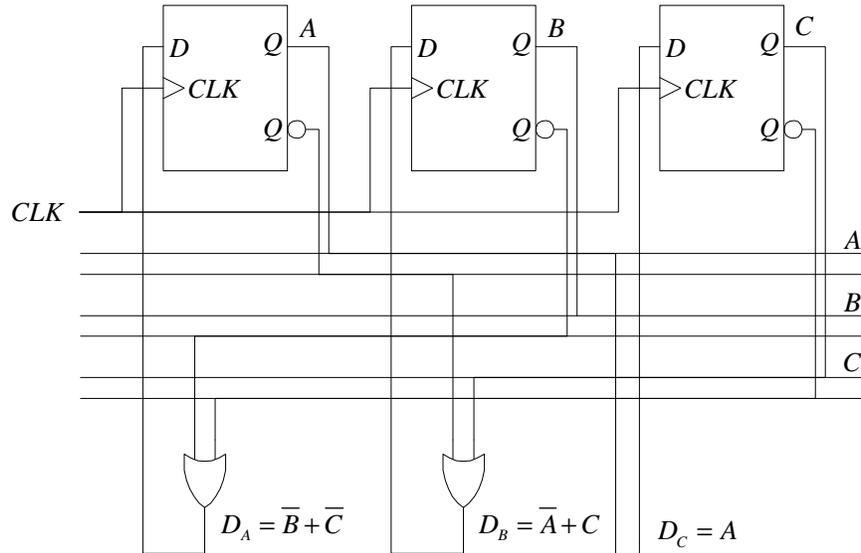


Figure 1: Implementation of the Synchronous Counter Using D-Flip-flops

6 Implementing the Design Using JK-Flip-flops

To change the design from one using D-flip-flops to one using JK-flip-flops requires considering what it takes to force a JK-flip-flop from one state to another. Table 5 shows this.

Output States		Inputs Required	
Q_{old}	Q_{new}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Table 5: Activation Table for a JK-Flip-flop

We can use this information to fill in the columns of a revised state table. Whereas Table 3 showed the control signals required to operate D-flip-flops, Table 6 shows the control signals required to operate JK-flip-flops. There are twice as many control signals because each flip-flop now has two control signals, not just one. This is unappealing superficially but the extra investment in functions often results in less complex logic circuitry. However, we won't find out unless we go through the design process.

	Current State				Next State			Flip-flop Controls					
	A	B	C		A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	×	×	×	×	×	×	×	×	×	×
1	0	0	1	×	×	×	×	×	×	×	×	×	×
2	0	1	0	6	1	1	0	1	×	×	0	0	×
3	0	1	1	2	0	1	0	0	×	×	0	×	1
4	1	0	0	×	×	×	×	×	×	×	×	×	×
5	1	0	1	7	1	1	1	×	0	1	×	×	0
6	1	1	0	5	1	0	1	×	0	×	1	1	×
7	1	1	1	3	0	1	1	×	1	×	0	×	0

Table 6: State Table Including JK-Flip-flop Inputs Required

In Table 7 the controls of Table 6 have been placed in Karnaugh maps to facilitate obtaining minimal logic equations, shown below each Karnaugh map.

Figure 2 shows our implementation of the synchronous counter using JK-flip-flops. Circuitry to permit the state to be initialized to $101_2 = 5_{10}$ has been added using the preset P and clear C functions of these JK-flip-flops. The P input of flip-flop A , the C input of flip-flop B , and the P input of flip-flop C are asserted whenever the $START$ signal is applied, forcing the binary value 101_2 into the three-flip-flop counter immediately, without waiting for a high-going $CLOCK$ -transition. Otherwise these inputs are tied to ground via resistor R .

		C
J_A	0	1
00	×	×
AB	01	1 0
11	×	×
10	×	×
$J_A = \overline{C}$		
		C
K_A	0	1
00	×	×
AB	01	×
11	0	1
10	×	0
$K_A = BC$		

		C
J_B	0	1
00	×	×
AB	01	×
11	×	×
10	×	1
$J_B = 1$		
		C
K_B	0	1
00	×	×
AB	01	0 0
11	1	0
10	×	×
$K_B = A\overline{C}$		

		C
J_C	0	1
00	×	×
AB	01	0 ×
11	1	×
10	×	×
$J_C = A$		
		C
K_C	0	1
00	×	×
AB	01	×
11	×	0
10	×	0
$K_C = \overline{A}$		

Table 7: Karnaugh Maps for Implementing f Using JK-Flip-flops

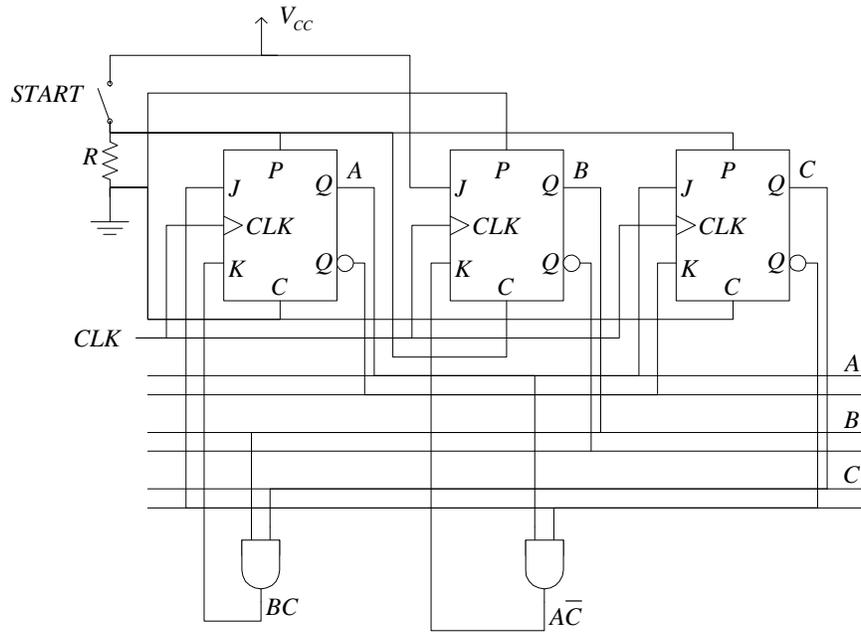


Figure 2: Implementation of the Synchronous Counter Using JK-Flip-flops. Circuitry to put the counter into state $101_2 = 5_{10}$ has been added using the preset and set functions of these JK-flip-flops.

The other three asynchronous inputs are never asserted, so they are always tied to ground. These are the C inputs of flip-flops A and C and the P input of flip-flop B .

7 Implementing the Design Using T-Flip-flops

To change the design from one using D-flip-flops or JK-flip-flops to one using T-flip-flops requires considering what it takes to force a T-flip-flop from one state to another. Table 8 shows this.

Output States		Inputs Required
Q_{old}	Q_{new}	T
0	0	0
0	1	1
1	0	1
1	1	0

Table 8: Activation Table for a T-Flip-flop

We can use this information to fill in the columns of a revised state table. Whereas Table 3 showed the control signals required to operate D-flip-flops, Table 9 shows the control signals required to operate T-flip-flops. There are just as many control signals because, as with D-flip-flops, each flip-flop has only one control signal, not two as was the case with JK-flip-flops.

	Current State				Next State			Flip-flop Controls		
	A	B	C		A	B	C	T_A	T_B	T_C
0	0	0	0	×	×	×	×	×	×	×
1	0	0	1	×	×	×	×	×	×	×
2	0	1	0	6	1	1	0	1	0	0
3	0	1	1	2	0	1	0	0	0	1
4	1	0	0	×	×	×	×	×	×	×
5	1	0	1	7	1	1	1	0	1	0
6	1	1	0	5	1	0	1	0	1	1
7	1	1	1	3	0	1	1	1	0	0

Table 9: State Table Including T-Flip-flop Inputs Required

In Table 10 the controls of Table 9 have been placed in Karnaugh maps to facilitate obtaining minimal logic equations, shown below each Karnaugh map.

Figure 3 shows our implementation of the synchronous counter using T-flip-flops. Circuitry to permit the state to be initialized to $110_2 = 6_{10}$ has been added using the preset \bar{P} and clear \bar{C} functions of these T-flip-flops. Note that we initialize the counter in the JK-flip-flop implementation to 5_{10} , not 6_{10} . This

T_A		C	
	0	1	
00	×	×	
AB 01	1	0	
11	0	1	
10	×	0	
			$T_A = \overline{A}\overline{C} + ABC$

T_B		C	
	0	1	
00	×	×	
AB 01	0	0	
11	1	0	
10	×	1	
			$T_B = A\overline{C} + \overline{B}$

T_C		C	
	0	1	
00	×	×	
AB 01	0	1	
11	1	0	
10	×	0	
			$T_C = A\overline{C} + \overline{A}C$

Table 10: Karnaugh Maps for Implementing f Using T-Flip-flops

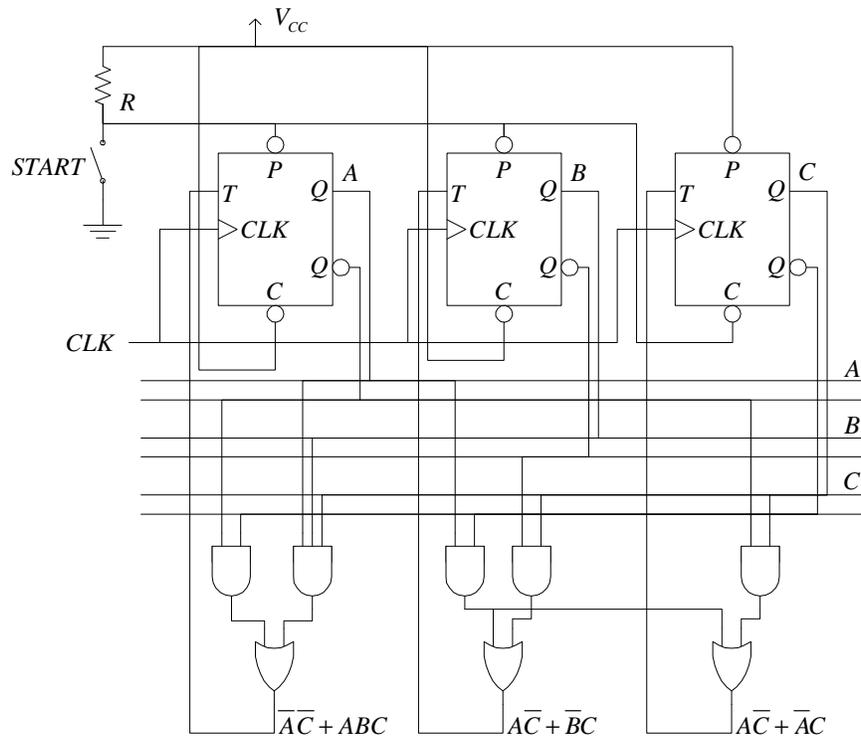


Figure 3: Implementation of the Synchronous Counter Using T-Flip-flops.

As with the JK-flip-flop implementation, this implementation contains circuitry to put the counter into an initial state. In this case, we chose an initial state $110_2 = 6_{10}$, just to illustrate that any initial state is easy to obtain. As before, we use the asynchronous preset and clear functions of these flip-flops, with the difference that these are now T-flip-flops, not JK-flip-flops. Also, we have made the asynchronous inputs active-low inputs rather than active-high inputs to illustrate what is needed to accommodate this change. Since the term $A\overline{C}$ appears twice, we can simply use the output of the gate that computes it in the two places it is required.

change is simply to show that it is easy to start with any desired initial state, as long as it is in the list of desired states. Also, the flip-flops used here have active-low asynchronous inputs, not active-high asynchronous inputs as our JK-flip-flops did. Thus the P input of flip-flop A , the P input of flip-flop B , and the C input of flip-flop C are asserted whenever the $START$ signal is applied, forcing the binary value 110_2 into the three-flip-flop counter immediately, without waiting for a high-going $CLOCK$ -transition. Otherwise these inputs are tied to V_{CC} via resistor R . The other three asynchronous inputs are never asserted, so they are always tied to V_{CC} . These are the C inputs of flip-flops A and B and the P input of flip-flop C .

Our implementation shows that we have eliminated one 2-input AND gate by noting that the term $A\bar{C}$ is needed as part of both T_B and T_C . We can compute it once and apply it in the two places it is needed, saving a gate.

8 Comparison of the Three Implementations

A comparison of the control-signal logic in the three implementations of the synchronous counter shows which designs require the most circuitry. Generally, more circuitry requires more debugging effort and so is to be avoided.

This comparison shows an unusual result: the JK-flip-flop implementation takes two 2-input AND gates, compared to two 2-input OR gates for the D-flip-flop implementation. This is unusual because the use of JK-flip-flops usually leads to *fewer* input gates, not the same number or more.

The comparison also reveals another unusual result: the T-flip-flop implementation takes substantially more gates than either of the other two implementations. It requires four 2-input AND gates, three 2-input OR gates, and one 3-input AND gate. It is more common for T-flip-flop implementations to be simpler than their corresponding D-flip-flop implementations.

In general, there is no way to predict which of several possible equivalent implementations will be most economical unless this kind of detailed design is done. Here, we could pick either the D-flip-flop or the JK-flip-flop implementation as the most economical, at least in the sense that they require less debugging and less hardware.