Introduction to the Conversion of Flip-Flops

July 18, 2016 by Sneha H.L.

This article describes the steps necessary to convert a given flip-flop into a desired flip-flop using the example of an SR-to-JK flip-flop conversion.

Introduction

Flip-flops are bi-stable single-bit memory devices which are one among the numerous digital components used in sequential systems.

The various kinds of flip-flops in existence are SR flip-flop, JK flip-flop, D flip-flop, and T flip-flop. Each of them exhibit unique characteristics and hence result in different output for the same combination of input states. Hence, whenever we want one flip-flop to mimic the behaviour of the other, we need to resort to a flip-flop conversion technique.

Creating an Excitation Table

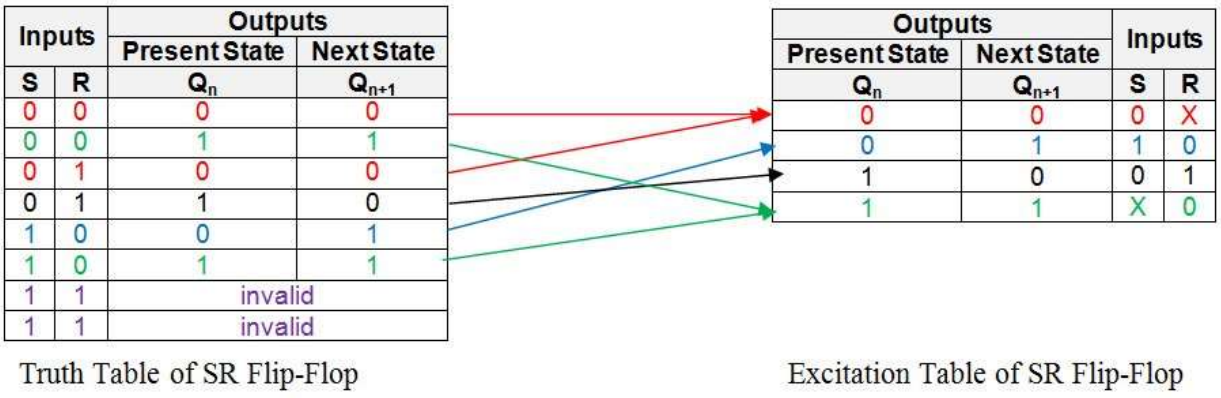
The output of a flip-flop at a given instant depends on both its input(s) as well as its present-state, defined by the information summarized and presented by its truth table. In other words, in the truth table of a flip-flop, the next-state output will be the last column. That column is determined by the combination of bits in its preceding columns, which will be the input(s) followed by its present-state.

Now, just imagine that we want to know the sequence of the input combination which results in a definite output state. The information pertaining to this can be obtained by back-tracing (in terms of columns) the information presented by the truth table of the flip-flop. That is, we will have the first two columns as the

present- and the next-states of the flip-flop which will be followed by the column(s) representing the flip-flop inputs.

Such a table can be aptly referred to as an "excitation table" as it indicates the excitations to be provided at the input pins of the flip-flop to result in the expected outcome for a known present-state.

The concept explained can be further made clear by the following example, where we obtain the excitation table for the SR flip-flop from its truth table:



Inputs		Outputs	
		Present State	Next State
S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	invalid	
1	1	invalid	

Truth Table of SR Flip-Flop

Outputs		Inputs	
Present State	Next State		
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation Table of SR Flip-Flop

Figure 1: Truth table and excitation table of an SR flip-flop

The first row in the truth table above shows that the present- and the next-states of the flip-flop will be 0 and 0 if its inputs are S = 0 and R = 0.

The same output combination also appears even when the inputs are given as S = 0 and R = 1 as evident from the third row of the truth table. This indicates that, in order to obtain the SR flip-flop's output as 0, we must drive the input pin, S, to low (i.e. S = 0) while the other input, R, may be pulled either low or high (i.e. R = 0 or 1), provided its present state is 0. In other words, the input combination S = 0 and R = X (Don't Care) results in the next-state of the flip-flop to be 0 from its current state, which is equal to 0.

Now, note that the same information is successfully conveyed by the entries (shown in the red colour) in the first row of the excitation table.

Similarly, the present-state and next-state combination of 0 and 1 is obtained for the SR flip-flop when its inputs are S = 1 and R = 0. This information is concisely represented by the second row of the excitation table (shown in the blue colour).

Following on the same grounds, we find that to obtain present- and next-states of the flip-flop as 1 and 0, we should have S = 0 and R = 1, as indicated by the entries in the black colour corresponding to the third row of the excitation table.

Lastly, note that S can be either 1 or 0 (i.e. S = X) and R should be 0 in order to obtain the present- and next-states of the flip-flop as 1 and 1. This is shown by the green colour entries in the fourth row of its excitation table.

Having done this, all the information present in the truth table is transferred appropriately into the excitation table, completing it.

By employing the same procedure, the excitation tables can be obtained for all other types of flip-flops viz., JK flip-flop, D flip-flop, and T flip-flop as shown by Figures 2, 3 and 4, respectively:

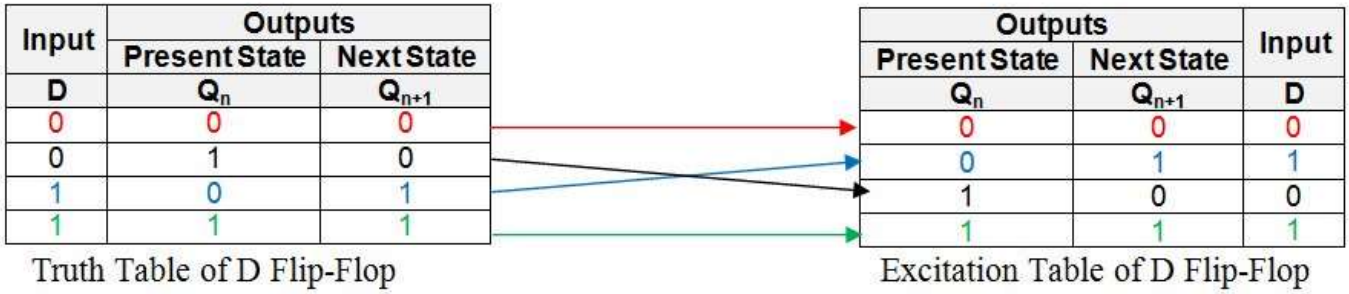
Inputs		Outputs	
		Present State	Next State
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table of JK Flip-Flop

Outputs		Inputs	
Present State	Next State		
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation Table of JK Flip-Flop

Figure 2: Truth table and excitation table of a JK flip-flop



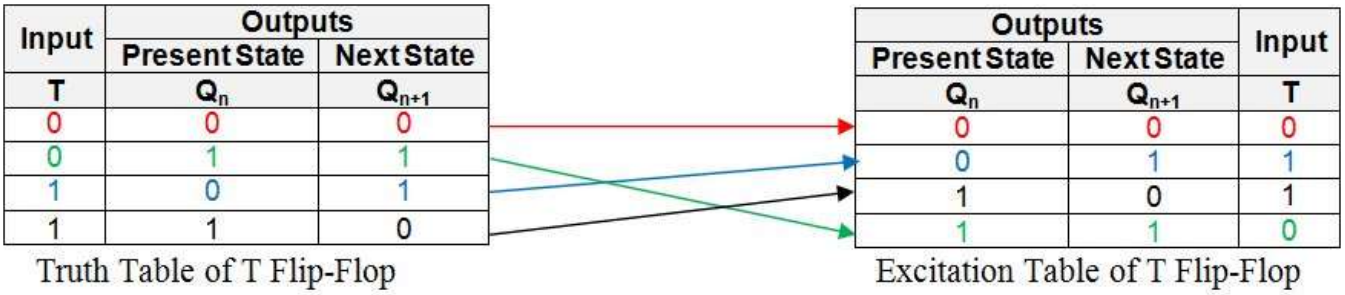
Input	Outputs	
	Present State	Next State
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Truth Table of D Flip-Flop

Outputs		Input
Present State	Next State	
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table of D Flip-Flop

Figure 3: Truth table and excitation table of a D flip-flop



Input	Outputs	
	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of T Flip-Flop

Outputs		Input
Present State	Next State	
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table of T Flip-Flop

Figure 4: Truth table and excitation table of a T flip-flop

Meaning of "Desiring a Flip-Flop"

When we say, "We want a particular flip-flop", it means that we are clear about the flip-flop outcomes for the given combination of inputs for each case of its present-state. That is, we have the truth-table information of the desired flip-flop in our hands. However, the expected outcome cannot be obtained directly from the given flip-flop as its behaviour for the same combination of input states and the present-state will be different (most of the time).

Hence, one needs to determine the sequence of the input bits in the given flip-flop which results in the same output (for a definite present-state) as that in the case of the desired flip-flop. As mentioned before, this information is readily present in the form of the entries in the excitation table of the given flip-flop.

Knowing this, the next step would be to merge the information presented by the excitation table of the given flip-flop with the information present in the truth table of the desired flip-flop. This can be done by filling the entries from the excitation table of the given flip-flop into the appropriate rows of the truth table corresponding to the desired flip-flop by adding additional column(s) which represent the input(s) of the given flip-flop. When done so, we will get a new table which we can refer to as a "Conversion Table**":**

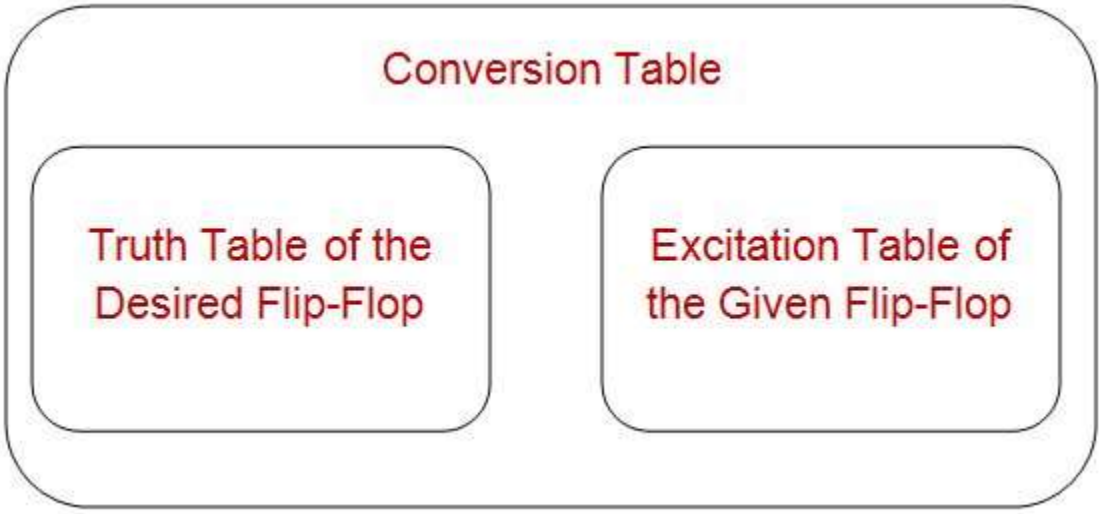


Figure 5: Structure of a conversion table

For example, the conversion process of the SR flip-flop into a JK flip-flop is initiated by writing the truth table for the JK flip-flop as shown by the yellowish enclosure in Figure 6. Here, it is seen that the first row has the present- and the next-states of the flip-flop as 0 and 0 (the red entries in the truth table).

Now we look at the excitation table of the SR flip-flop (shown in the right-side of Figure 6) which has a row indicating the present- and the next-states of the SR flip-flop to be 0 and 0. As seen by the red entries in the excitation table, this corresponds to the first row for which the inputs are S = 0 and R = X.

The same information is placed into the first row of the JK flip-flop's truth table by adding two more columns, S and R (as shown by the pink enclosure in Figure 6), to result in an SR-to-JK Conversion Table:

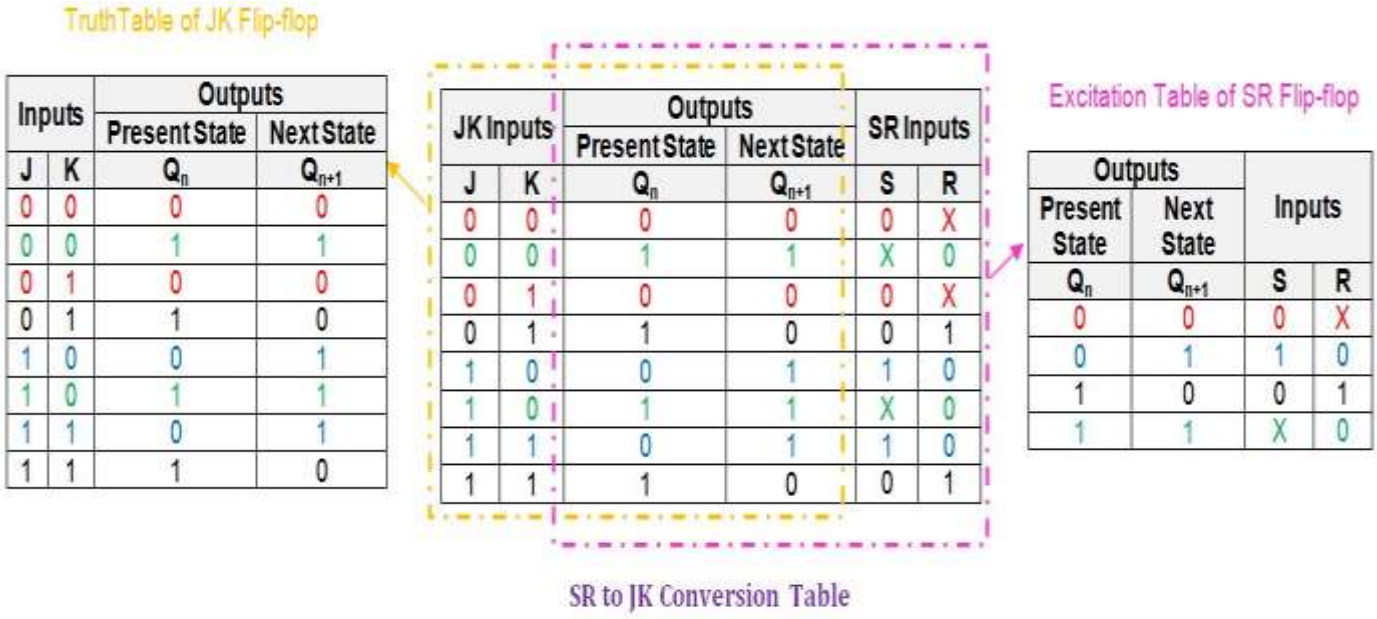


Figure 6: SR to JK conversion table

Similarly, the second row of the JK flip-flop has the present- and the next-states as 1 and 1 which correspond to the fourth row of the SR flip-flop's excitation table (shown as the green entries in the respective tables). The values for the S and R inputs corresponding to this output combination is seen to be X and 0, respectively, which are filled into the second row of the conversion table (shown in green again).

By following the same procedure for each and every row, the entire table can be filled to obtain a completed SR-to-JK flip-flop conversion table, as shown in the center of Figure 6.

Using a K-Map to Obtain Logical Expressions

Having obtained the conversion table, the next step is to arrive at the logical expressions for the inputs of the given flip-flop in terms of the inputs of the desired flip-flop and the present-state.

Further, you should take care to obtain the minimal logical expression so as to facilitate the design of the circuit with the least possible gates. This objective can be achieved by employing any of the suitable Boolean algebraic simplification techniques like that of the K-map technique.

According to this, for the example under consideration, we need to obtain the expressions for the inputs S and R in terms of J, K, and Q_n. This can be done by employing the K-map simplification technique as shown in Figure 7:

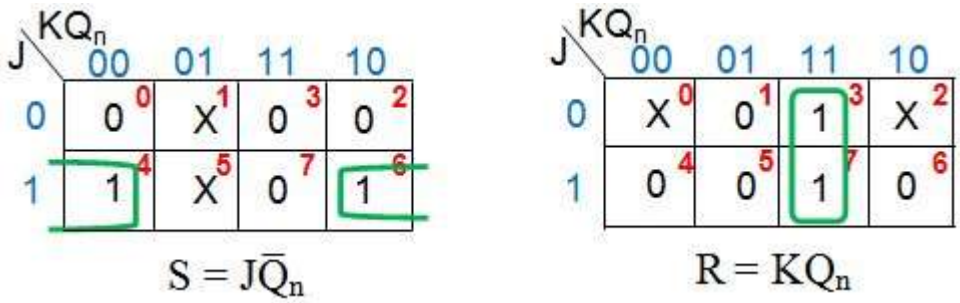


Figure 7: K-map simplification for the inputs of the SR flip-flop in terms of J, K, and Q_n

This results in the expression for S input as $J\bar{Q}_n$ and for R input as KQ_n.

An SR-to-JK Flip-Flop Conversion

At last, by using the information provided by the logical expression obtained, we can re-design the connections to be provided at the input pins of the given flip-flop. This can be achieved either by just manipulating the connections and/or by adding additional combinational circuit(s), inorder to make the given flip-flop functionally equivalent to the desired flip-flop.

In the example provided, the logical expressions obtained for S and R indicate two things:

- The input pin S of the SR flip-flop is to be fed by the output of a two-input AND gate which is driven by J and $\overline{Q}_{n.}$
- The input pin R of the SR flip-flop is to be fed by the output of a two-input AND gate which is driven by K and $Q_{n.}$

Both of these must be satisfied in order to make an SR flip-flop behave like a JK flip-flop as shown in Figure 8:

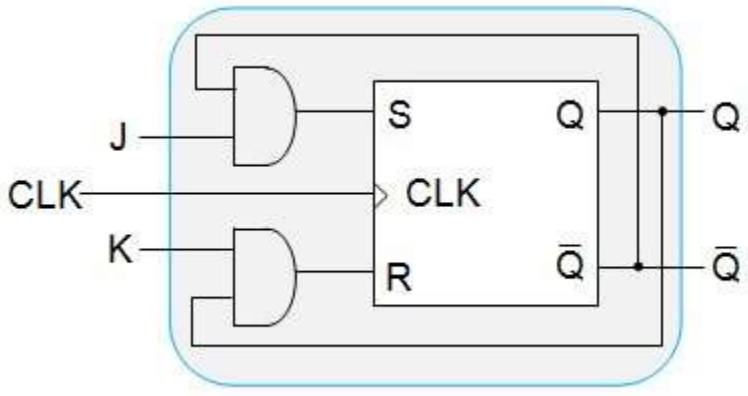


Figure 8: An SR flip-flop behaving like a JK flip-flop

In Review

In this article, we have presented a detailed procedure which can be carried out to convert any of the given flip-flops into any other type of flip-flop. We have shown this using an example wherein an SR flip-flop is made functionally equivalent to JK flip-flop.

However, it is to be noted that similar steps will also hold good for mutual conversion between any kind of flip-flops available. We can also cross-verify whether the conversion process was successful or not. These things will be covered in Part 2 of this article.

SR-to-D and SR-to-T Flip-Flop Conversions

July 28, 2016 by Sneha H.L.

This article will teach you how to verify flip-flop conversions for SR-to-JK flip-flops. It will also guide you through the conversion and verification processes for SR-to-D and SR-to-T flip-flops.

Previous Articles in This Series

- Conversion of Flip-Flops — Part I

In Part I of this article, we discussed the detailed procedure involved in converting an SR flip-flop into a JK flip-flop. Here, we will discuss the method which can be used to verify the conversion process for the same example. We'll also briefly explain the conversion and verification techniques for the conversion of (i) an SR flip-flop into D-type and (ii) an SR flip-flop into T-type.

A Brief Review

In the first part of this article, we had discussed the steps to be followed to convert an SR flip-flop into a JK flip-flop in detail. The process of conversion indicated that, in order to achieve the objective, we would need to do two things:

1. Drive the input, S, by the output of a two-input AND gate which has its inputs as J and $\bar{Q}_n$
2. Drive the input, R, by the output of a two-input AND gate whose inputs are K and Q_n

As a result, the digital system designed in Part I is as shown by Figure 1:

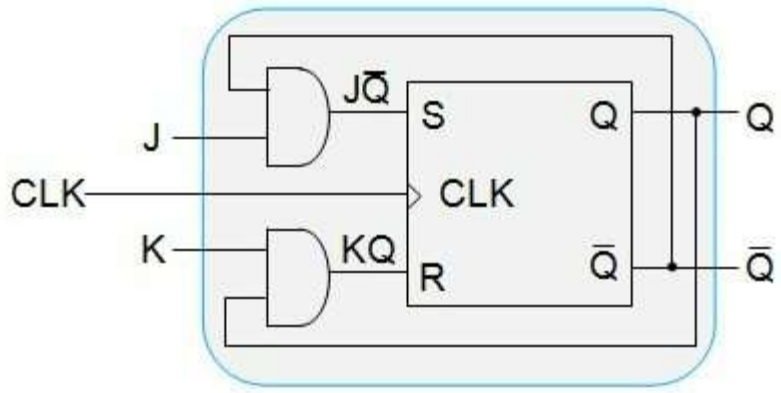


Figure 1: An SR Flip-Flop converted to a JK Flip-Flop

However, we did not verify our design in order to make sure what we obtained is what we needed. Although this would seem to be extra work, it is worth of being done so as to confirm that the system we've designed works as expected.

Verification

As is the case for most digital systems, this design can be verified by exciting it with random inputs and checking whether the outcome will be the same as that of our prediction. This process, when done systematically, will lead to the "truth-table mode of verification" technique.

Anatomy of a Verification Table

For example, the system shown in Figure 1 can be verified by writing its truth table as shown in Figure 2, which can be referred to as the verification table of an SR-to-JK flip-flop conversion.

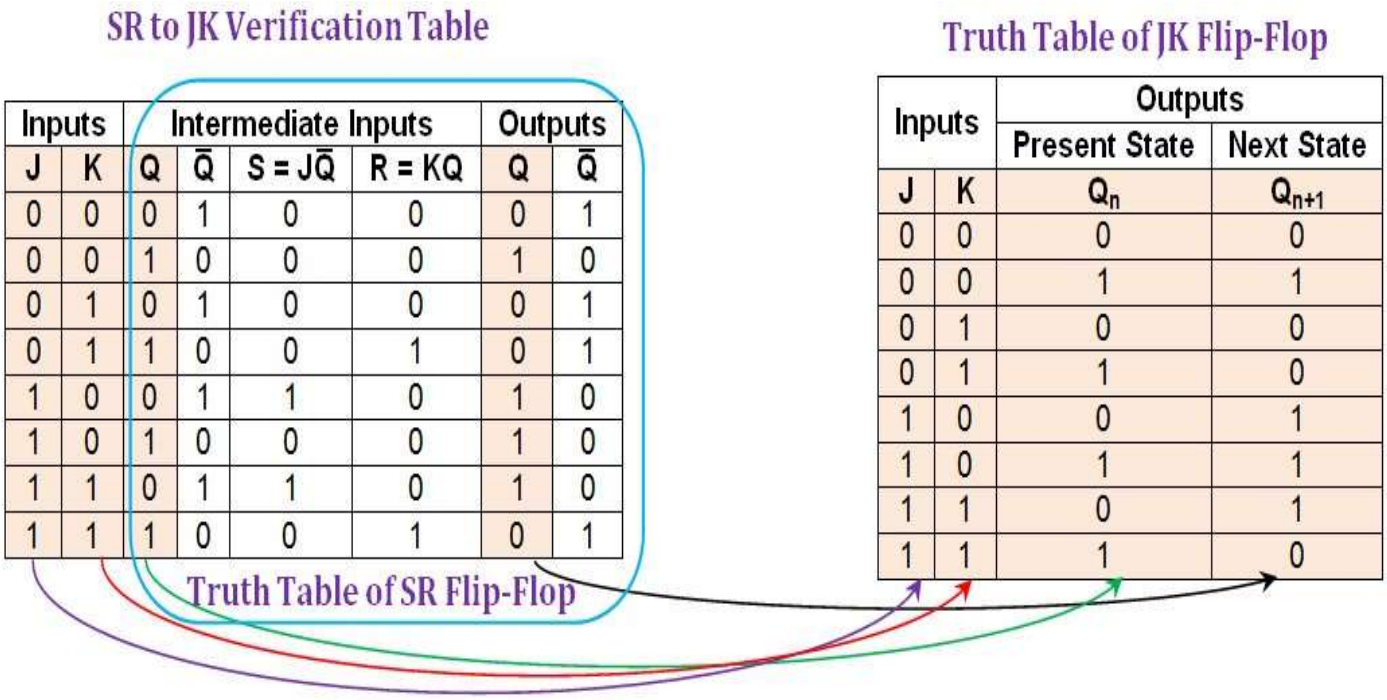


Figure 2: Comparison between an SR-to-JK verification table and the truth table of JK flip-flop. Click to enlarge.

Here, the first two columns designated as "inputs" indicate the input pins which are to be driven by the user (note that these are only the inputs of the desired flip-flop).

The next set of four columns is designated as "intermediate inputs". Among these, the first column (Q) represents the present state of the flip-flop. The next column ($\bar{Q}$) represents its negation. The following two columns ($S=J\bar{Q}$ and $R=KQ$) indicate the bit patterns driving the inputs of the SR (given) flip-flop.

Finally, we have two more columns designated as "outputs" (Q and $\bar{Q}$) which show the output-bit and its negation for the inputs provided.

Filling Out a Verification Table

First, let us consider the case where the user pulls down both J and K inputs, i.e., $J = K = 0$.

Now, if the present state of the flip-flop, Q, is 0 (which implies that its inverted bit, $\bar{Q}$, is 1), then the input pins S and R will be driven by 0 (as $S = J\bar{Q}$) and 0 (as $R = KQ$), respectively. For this combination, the output-bit of the SR flip-flop will remain unchanged and thus one gets the output bit as 0 and its inverted bit as 1, as indicated by the output columns in the first row of the SR-to-JK verification table (shown in Figure 2).

On the other hand, we would have obtained the output bit as 1 and its inversion as 0 for the same case, provided the present-state, Q, of the flip-flop was 1. This is indicated by the second row of the SR-to-JK verification table.

Similarly, all other output entries in the table can be filled.

Once done, a close observation of the same reveals the following two points:

1. The columns of the table spanning from that of the third to the last yield nothing but the truth table of the SR flip-flop where (i) the S and R columns represent its

inputs, (ii) the Q column under the "intermediate inputs" section represents its present-state column, and (iii) the Q column under the "outputs" section represents its next-state column (the columns shown within the blue enclosure in Figure 2).

2. The first three columns of the verification table along with its penultimate column form the truth table of the JK flip-flop (as indicated by the purple, red, green, and black arrowed pointers, respectively, in Figure 2). This indicates that the designed system behaves identically to a JK flip-flop for any combination of input states and present state. Thus, we can conclude that our aim of converting the given SR flip-flop into the desired JK flip-flop was successful in a complete sense.

A similar process of conversion (presented in Part I) and verification methodology can be applied for any combination of flip-flop types. We'll demonstrate this in the following two examples wherein a given SR flip-flop is converted into D- and T-types.

Conversion of an SR-to-D Flip-Flop

The process is initiated by obtaining the SR-to-D conversion table – a table which incorporates the information present in the excitation table of the SR flip-flop into the truth table of the D flip-flop. This is shown in Figure 3:

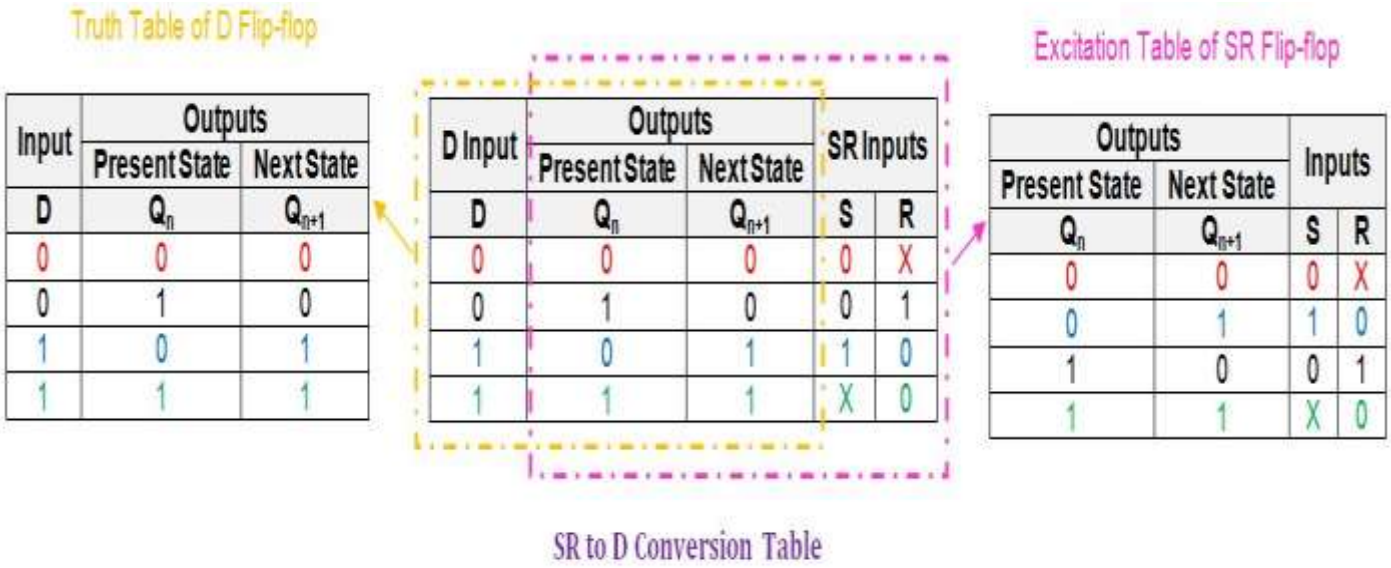


Figure 3: The breakdown of an SR-to-D conversion table. Click to enlarge.

Next, we need to obtain the logical expressions for the S and R input pins in terms of D and the present-state literal, Q_n. We can do this using a simplification technique like that of K-maps. You can learn more about this technique here.

In this case, the technique yields a K-map as shown in Figure 4:

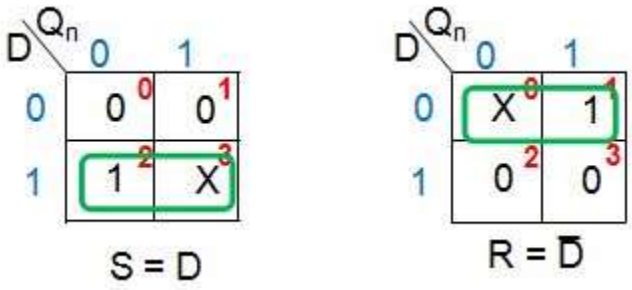


Figure 4: K-map simplification for the conversion of an SR-to-D flip-flop

From Figure 4, we can conclude that the given SR flip-flop can be made functionally equivalent to a D flip-flop by driving its S and R inputs by D and $\bar{D}$, respectively. Thus, the required digital system can be designed by using a single NOT gate as shown by Figure 5:

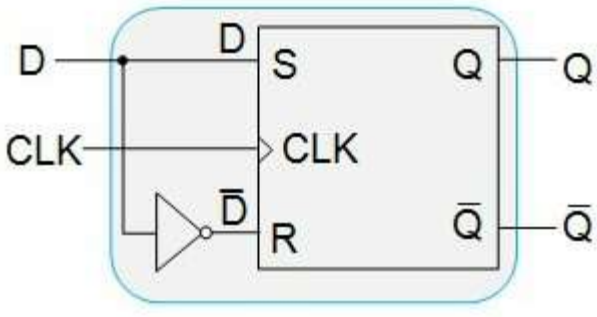
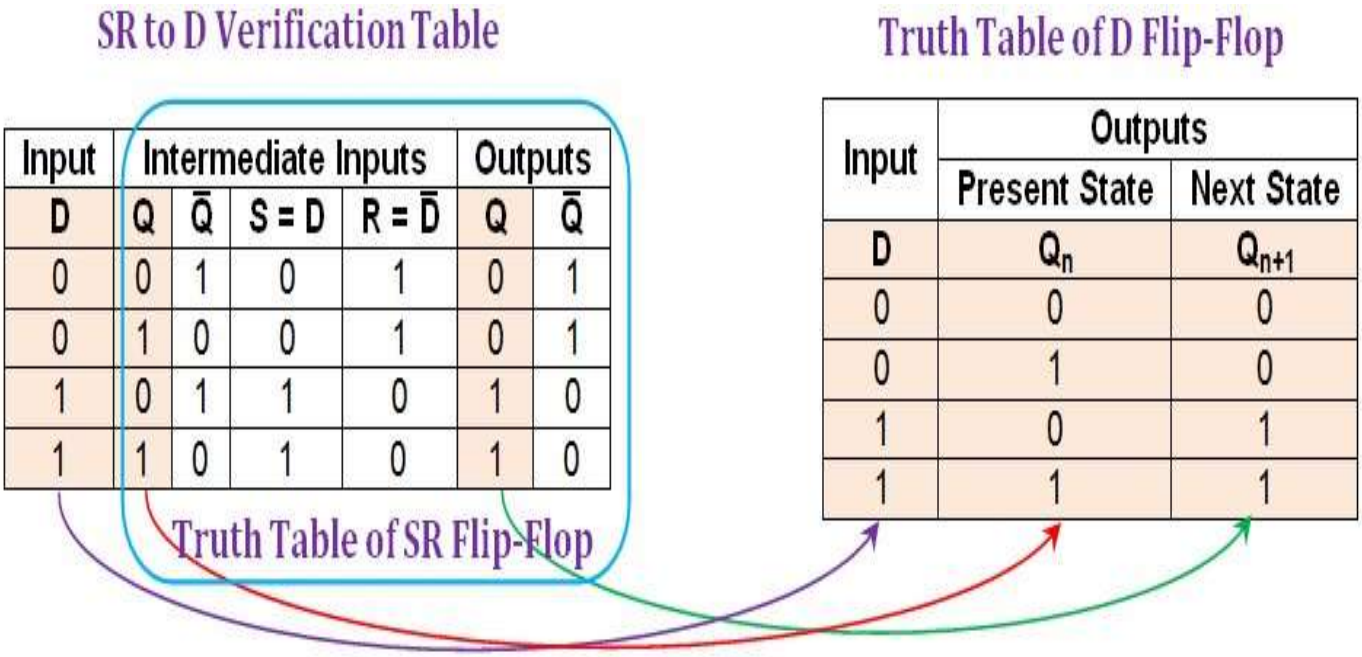


Figure 5: An SR flip-flop functioning as a D flip-flop

Having designed the system, our next step is to verify its functionality by using an SR-to-D verification table as shown in Figure 6:



Input	Intermediate Inputs				Outputs	
D	Q	$\bar{Q}$	S = D	R = $\bar{D}$	Q	$\bar{Q}$
0	0	1	0	1	0	1
0	1	0	0	1	0	1
1	0	1	1	0	1	0
1	1	0	1	0	1	0

Input	Outputs	
	Present State	Next State
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Figure 6: Comparison between an SR-to-D verification table and the truth table of D flip-flop. Click to enlarge.

From the verification table shown in Figure 6, it is evident that the entries in its first, second, and sixth columns (shaded in beige) are identical to the entries found in the truth table of the D flip-flop. This indicates that the system designed using the given SR flip-flop will behave exactly as a D flip-flop.

Conversion of an SR-to-T Flip-Flop

In order to convert the given SR flip-flop into T-type, we have to first write the SR-to-T conversion table, which is shown in Figure 7:

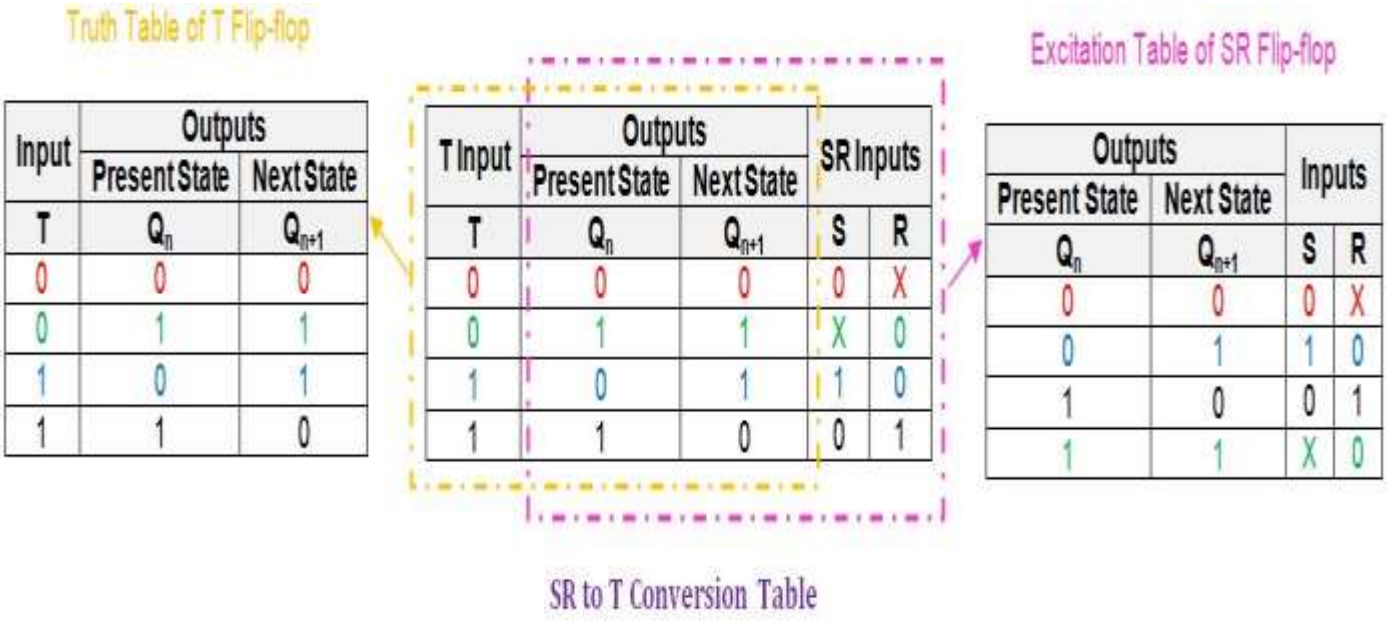


Figure 7: An SR-to-T conversion table. Click to enlarge.

Now, we need to express the inputs S and R in terms of T and the present-state literal, Q_n. This can be accomplished by simplifying their logical expressions using the K-map technique (Figure 8).

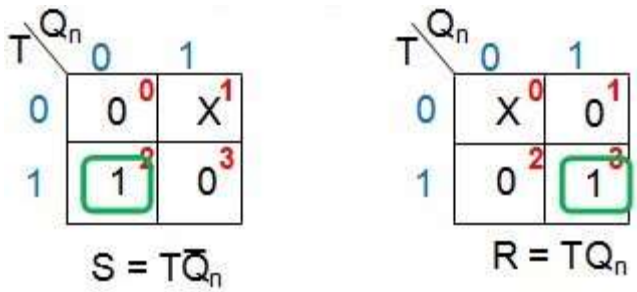


Figure 8: K-map simplification for S and R inputs in terms of T and Q_n

From Figure 8, it can be noted that the SR flip-flop can be made to function as a T flip-flop with two actions:

1. Connect the S input to the output of a two-input AND gate which is driven by the user-provided input, T, and the negation of the flip-flop's present-state, $\overline{Q}_n$

2. Connect the R input to the output of a two-input AND gate which is driven by the user-defined input, T, and the present-state of the flip-flop, Q_n

Thus the resultant digital system would be as shown in Figure 9:

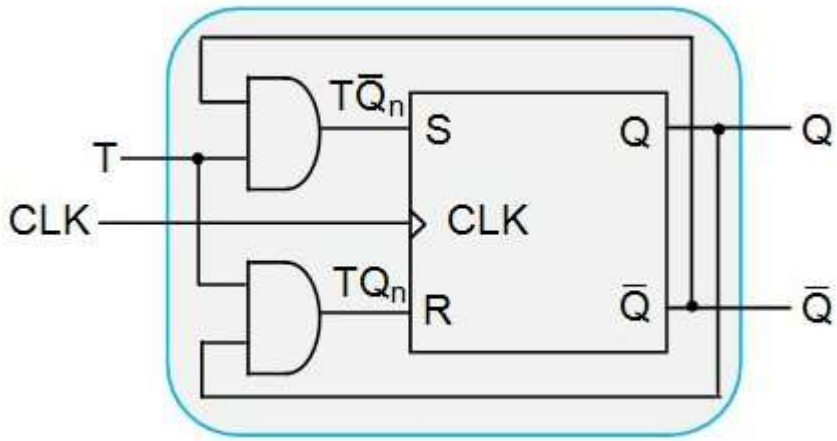


Figure 9: An SR flip-flop functioning as a T flip-flop

Now we shall check our conversion technique by writing the SR-to-T verification table, which is shown in Figure 10:

SR to T Verification Table

Input	Intermediate Inputs				Outputs	
T	Q	$\bar{Q}$	$S = T\bar{Q}$	$R = TQ$	Q	$\bar{Q}$
0	0	1	0	0	0	1
0	1	0	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	1	0	1

Truth Table of SR Flip-Flop

Truth Table of T Flip-Flop

Input	Outputs	
	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Figure 10: The comparison between an SR-to-T verification table and the truth table of a T flip-flop. Click to enlarge.

Here it is seen that the entries in the first, second, and the sixth columns of the SR-to-T verification table (shaded in beige) match exactly the entries in the truth table of the T flip-flop. From this, one can conclude that the system designed converts the given SR flip-flop into a T flip-flop. This indicates a successful conversion process.

An Overview

In this article, we explained the process of verifying the flip-flop conversion technique for the SR-to-JK flip-flop conversion that was explained in Part I.

Further, the same conversion and verification techniques were applied to two more examples wherein the given SR flip-flop was converted into a D flip-flop and a T flip-flop.

In Part III of this series, we will present the conversion of a JK flip-flop to other flip-flop types and also verify the conversions.

Conversion of JK Flip-Flops

August 10, 2016 by Sneha H.L.

This article teaches you how to convert a given JK flip-flop circuit to other types of flip-flops while verifying the process of conversion.

Previous Articles in This Series

- Conversion of Flip-Flops — Part I
- Conversion of Flip-Flops — Part II

A Quick Review

The previous two parts of this series discussed the method of conversion and verification processes for (i) SR-to-JK flip-flop, (ii) SR-to-D flip-flop and (iii) SR-to-T flip-flop. Continuing it, here, we apply the same techniques for converting the given JK flip-flop to SR-, D- and T-types, while verifying the conversion. Please read Part I and Part II before continuing.

Conversion of a JK-to-SR Flip-Flop

Step 1: Write the JK-to-SR conversion table

The first step in converting a JK-to-SR flip-flop would be to write a JK-to-SR conversion table as shown in Figure 1.

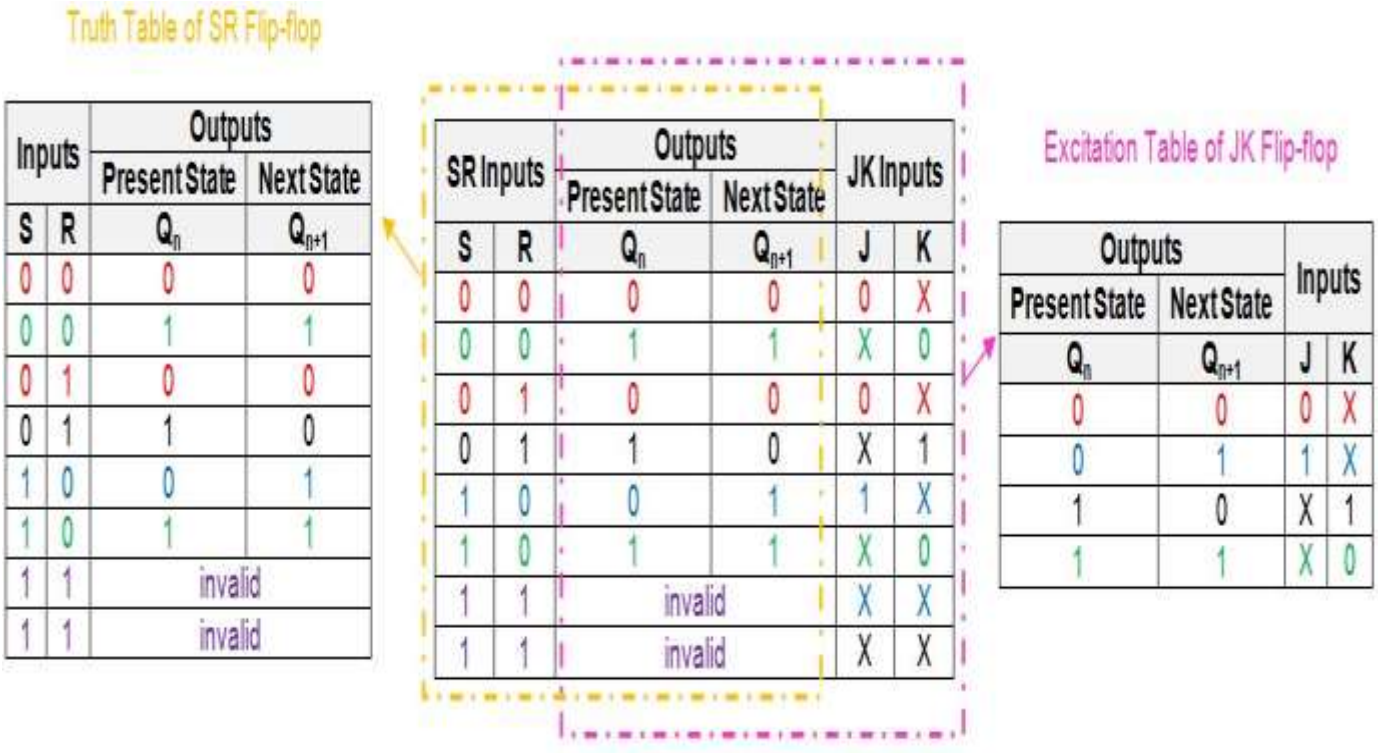


Figure 1: JK-to-SR conversion table. Click to enlarge.

The intention behind this step is to represent the information presented by the truth table of the SR flip-flop and the excitation table of the JK flip-flop in a common table.

However, note that the last two rows of the table have "don't cares" as the entries in their J and K columns. This is because the input combination of S = 1 and R = 1 is invalid in the case of an SR flip-flop.

Step 2: Simplify the logical expressions for the inputs

Now, simplify the logical expressions for the inputs of the given flip-flop (J and K) in terms of the inputs of the desired flip-flop (S and R) and the flip-flop's present-state, Q_n.

This can be done by following any logical simplification technique like that of the K-map. (For reference, I have covered in a previous article how to use the K-map method of simplification.)

Figure 2: K-map simplification for J and K inputs in terms of S, R and Q_n

Figure 2 shows that the given JK flip-flop behaves readily as an SR flip-flop—it needs neither additional circuitry nor the manipulation of the connections. So the required digital system (Figure 3) will be nothing but the given JK flip-flop.

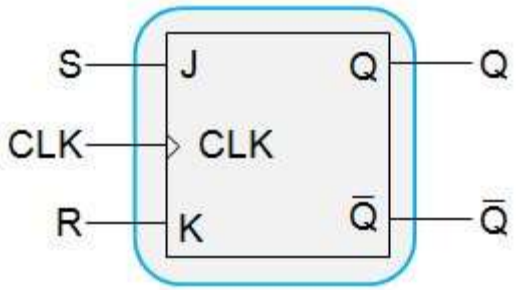


Figure 3: A JK flip-flop behaving as an SR flip-flop

Step 3: Verification

The next step is to verify our design using a JK-to-SR verification table as shown in Figure 4. The approach employed is simple: write the truth table for the designed system and compare it with the truth table of the desired flip-flop.

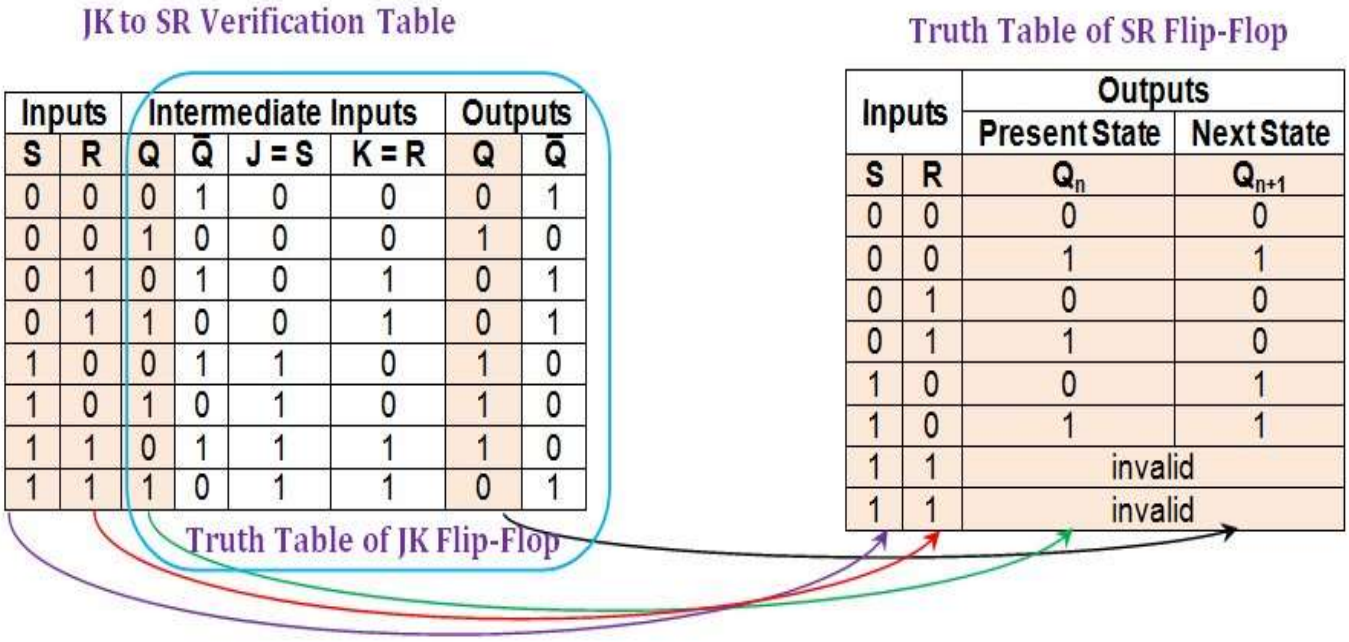


Figure 4: Comparison between the JK-to-SR verification table and the truth table of an SR flip-flop.

Click to enlarge.

From the figure, it is evident that the entries in the first, second, third, and seventh columns of the JK-to-SR verification table (shaded in beige) are consistent with the entries found in the truth table of the SR flip-flop. Note that the last two rows of the verification table, which seem to differ, can be considered equivalent. This is because the "invalid" in the truth table of the SR flip-flop signifies that the flip-flop is not supposed to be excited by the inputs S = 1 and R = 1. This can also be interpreted as follows: one cannot be sure of the outcome when both the inputs of the SR flip-flop are driven high. This means that the output can be either high or low; thus, the entries in the last two rows of our verification table are acceptable.

Hence we can conclude that the given JK flip-flop can function also as an SR flip-flop.

Now that we're familiar with the steps required to convert and verify these flip-flops, we'll run through two more examples a little bit more quickly.

Conversion of a JK-to-D Flip-Flop

This conversion process is initiated by writing the JK-to-D conversion table as shown in Figure 5.

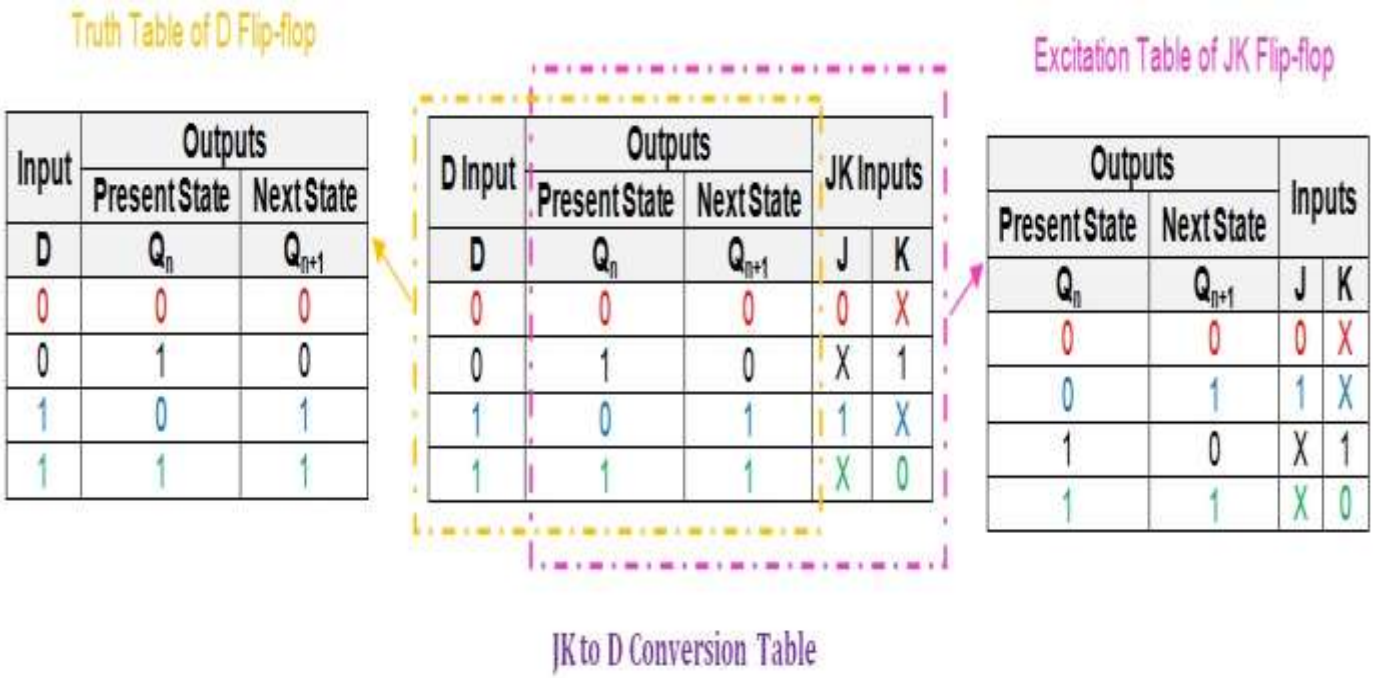


Figure 5: JK-to-D conversion table. Click to enlarge.

Next, let us use a K-map to obtain the logical expressions for the inputs J and K in terms of D and Q_n.

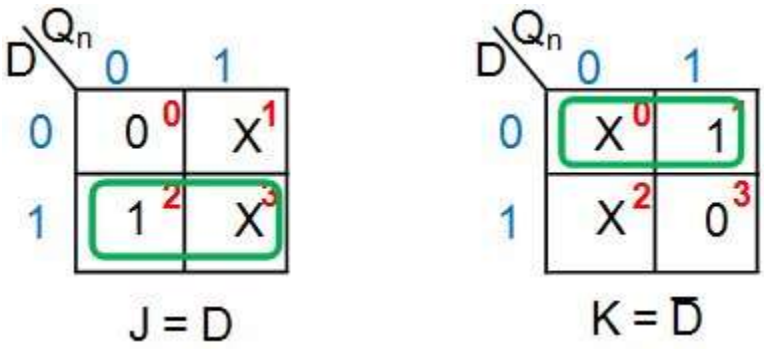


Figure 6: K-map simplification for J and K inputs in terms of D and Q_n

From Figure 6, it can be seen that the given JK flip-flop can be converted into a D-type flip-flop by driving its J and K input pins with the D input and its negation, respectively. Thus the additional hardware component required would be a NOT gate, resulting in the digital system shown in Figure 7.

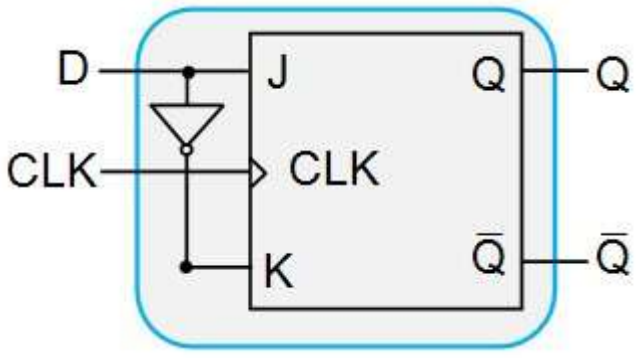


Figure 7: JK flip-flop designed to behave as a D flip-flop

Now, we shall verify our system so as to ensure that it behaves like we expect it to. For this, let us construct the JK-to-D verification table as shown in Figure 8.

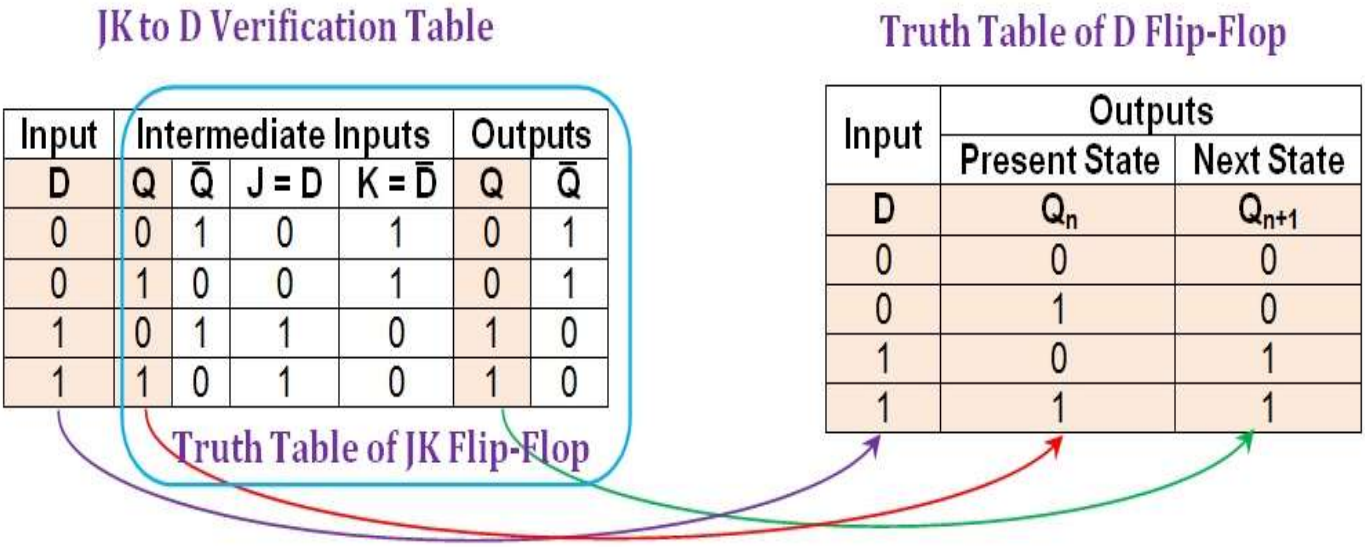


Figure 8: Comparison between the JK-to-D verification table and the truth table of a D flip-flop. Click to enlarge.

From the figure, it can be clearly seen that the entries in the first, second, and sixth columns of the JK-to-D verification table (shaded in beige) are the same as those in the D flip-flop's truth table. Thus, it can be concluded that the conversion process of JK flip-flop into D-type was successful.

Conversion of JK to T Flip-Flop

In order to convert the given JK flip-flop into a T flip-flop, we'll again begin with the initial requirement of obtaining the corresponding conversion table. You can see this table in Figure 9.

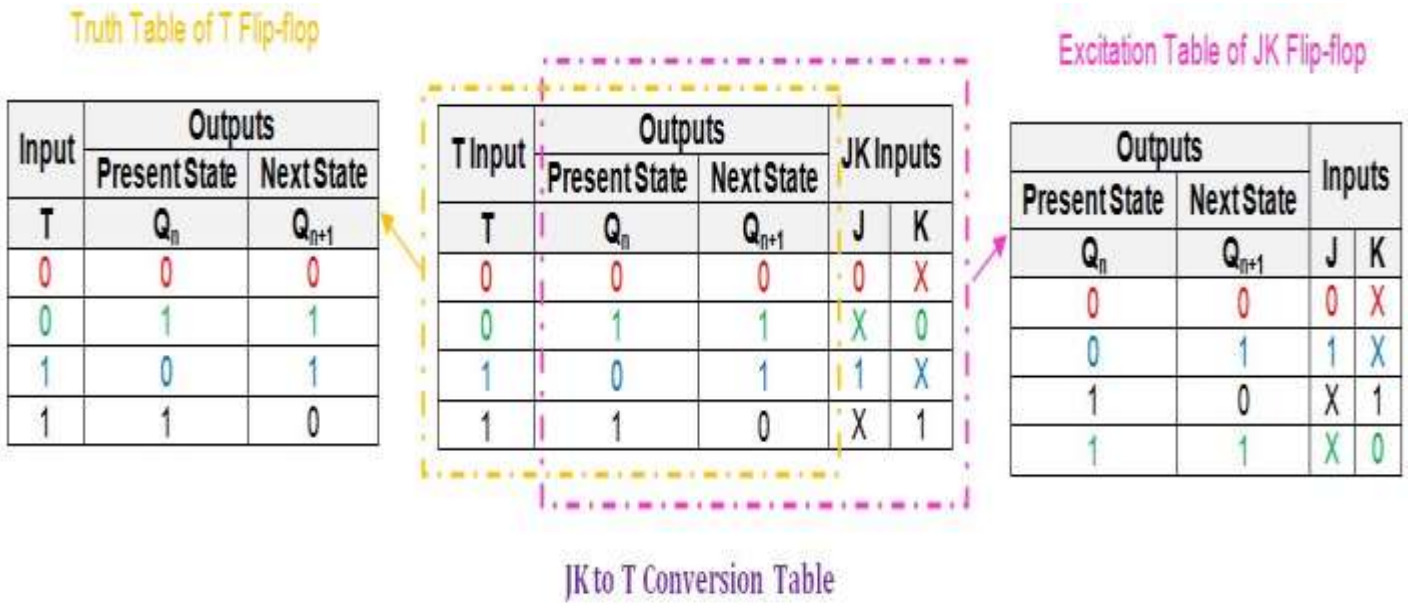


Figure 9: JK-to-T conversion table. Click to enlarge.

The next step is to express the inputs of the given JK flip-flop (J and K) in terms of the input of the desired T flip-flop and the present state (T and Q_n, respectively).

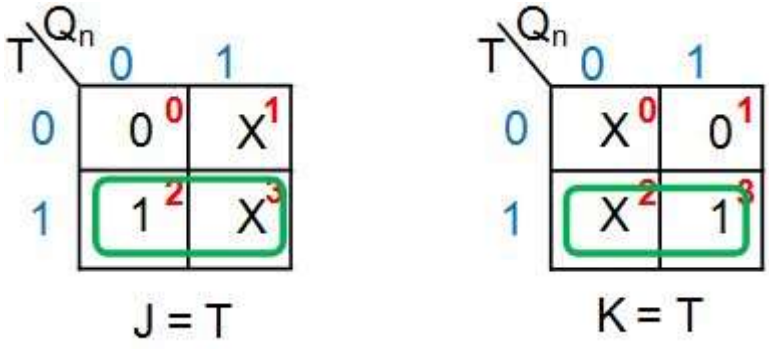


Figure 10: K-map simplification for J and K inputs in terms of T and Q_n

Figure 10 shows that in order to convert the given JK flip-flop into a T flip-flop, it's enough just to drive both of its input pins (J and K) with the input T. This results in the digital system shown in Figure 11.

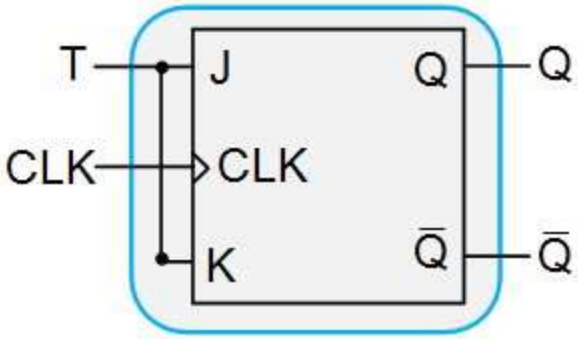
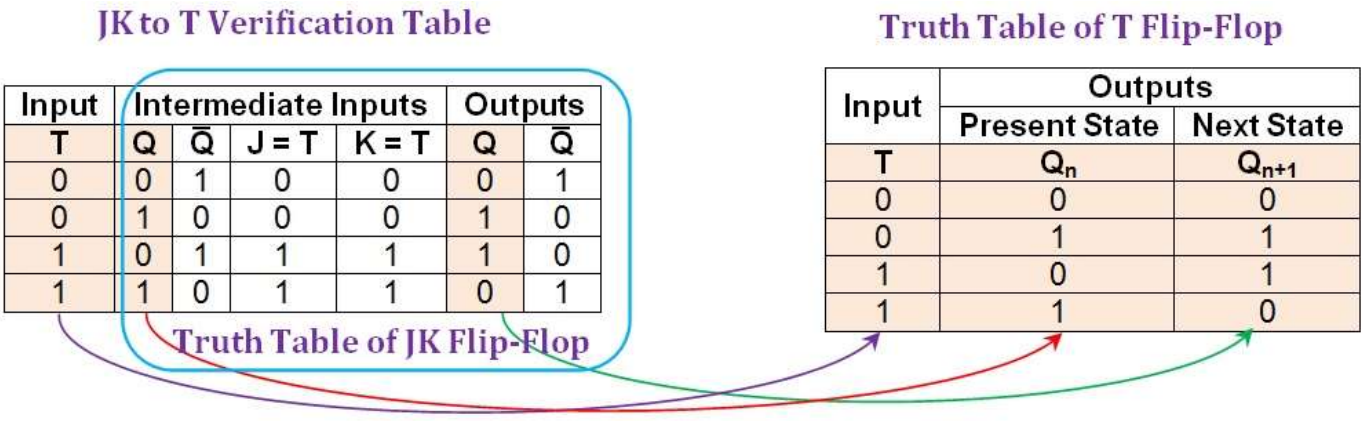


Figure 11: A JK flip-flop designed to behave as a T flip-flop

Lastly, verification of the completed conversion process can be performed using the JK-to-T verification table, as shown in Figure 12.



Input	Intermediate Inputs				Outputs	
T	Q	$\bar{Q}$	J = T	K = T	Q	$\bar{Q}$
0	0	1	0	0	0	1
0	1	0	0	0	1	0
1	0	1	1	1	1	0
1	1	0	1	1	0	1

Input	Outputs	
	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Figure 12: Comparison between the JK-to-T verification table and the truth table of a T flip-flop. Click to enlarge.

Here it can be seen that the first, second, and penultimate columns of the JK-to-T verification table (shaded in beige) are in agreement with the columns of the truth table of the T flip-flop. Hence, the given JK flip-flop functions equivalently to a T flip-flop for any combination of the input and the present state.

Summary

In this article, we have seen the processes associated with converting a JK flip-flop to SR-, D- and T-type flip-flops and then verifying the conversion.

Part IV of this series will discuss converting a given D flip-flop to SR-, JK- and T flip-flops and also present the verifications for these conversions.

Conversion of D Flip-Flops

August 25, 2016 by Sneha H.L.

This article presents the conversions of D-type flip-flops into SR-, JK- and T-types. It also explains how to verify the conversion process.

Previous Articles in This Series

- Conversion of Flip-Flops — Part I
- Conversion of Flip-Flops — Part II
- Conversion of Flip-Flops — Part III — JK Flip-Flops

Introduction

Here, we will briefly present the methodology to convert the given D flip-flop into (i) an SR flip-flop, (ii) a JK flip-flop and (iii) a T flip-flop; this process is discussed in detail in Part I of the series.

Next, we will verify the resultant systems using the technique described in detail in Part II of the series.

Conversion of a D to SR Flip-Flop

The process of converting the given D flip-flop into an SR-type is initiated by obtaining a table which represents both the information present in the truth table of the SR flip-flop as well as the information conveyed by the excitation table of the D flip-flop. Such a table is referred to as the D-to-SR conversion table and is shown in Figure 1.

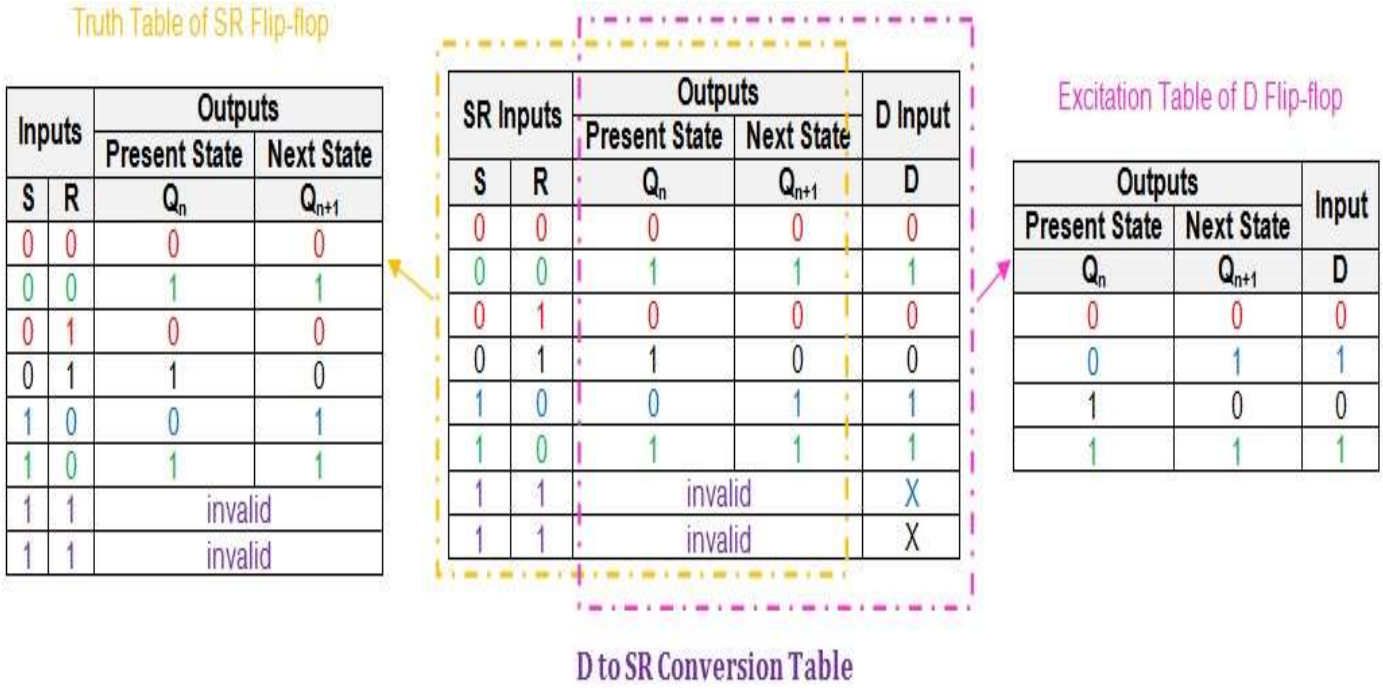


Figure 1: D-to-SR conversion table. Click to enlarge.

Here we note that the last two rows of the conversion table have X (Don't Cares) in the "D Input" column. This is because with an SR flip-flop the input combination of S = R = 1 is invalid (because the output will be unpredictable).

Our next step will be to obtain the logical expression for the input of the given D flip-flop in terms of the inputs of the desired flip-flop, S and R, and the present-state, Q_n. However, while doing so, we need to simplify the Boolean expression as much as possible using a suitable simplification technique, such as the K-map. The Karnaugh method is covered in detail here.

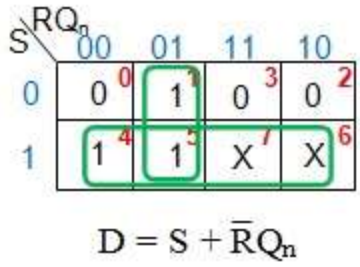


Figure 2: K-map Simplification for D input

From Figure 2, the simplified logical expression for the D input is found to be $S + \bar{R}Q_n$. This means that, in order to make the given D flip-flop behave like the desired SR flip-flop, we need to AND Q_n with the negation of the user-defined input R and then OR the result with the user-defined input S.

Thus the additional combinational circuit required would be one NOT gate, one AND gate, and one OR gate. The resultant system, designed using these components, is shown in Figure 3.

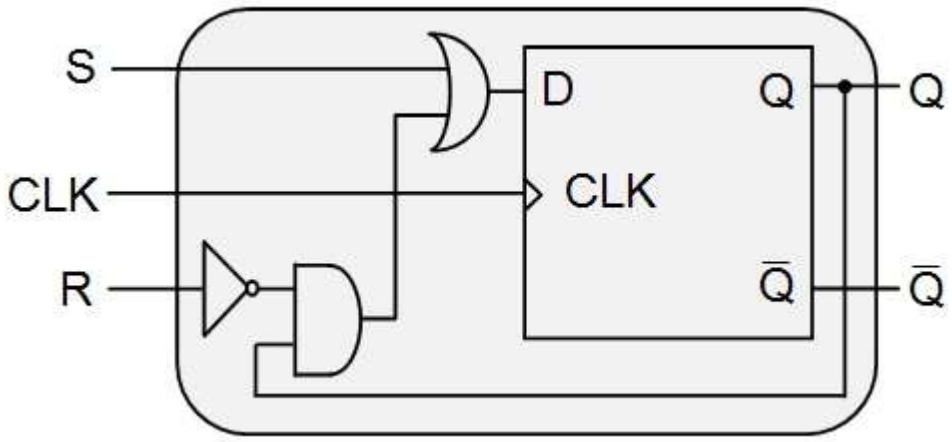


Figure 3: D flip-flop behaving as an SR flip-flop

Upon completion of the conversion process, we need to move on to the verification process. Here, we need to write the truth table for the designed system and compare its entries with those in the truth table of the SR (desired) flip-flop.

D to SR Verification Table

Inputs		Intermediate Inputs					Outputs	
S	R	Q	$\bar{Q}$	$\bar{R}$	$\bar{R}Q$	D = S + $\bar{R}Q$	Q	$\bar{Q}$
0	0	0	1	1	0	0	0	1
0	0	1	0	1	1	1	1	0
0	1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0	1
1	0	0	1	1	0	1	1	0
1	0	1	0	1	1	1	1	0
1	1	0	1	0	0	1	1	0
1	1	1	0	0	0	1	1	0

Truth Table of D Flip-Flop

Truth Table of SR Flip-Flop

Inputs		Outputs	
		Present State	Next State
S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	invalid	
1	1	invalid	

Figure 4: Comparison between the D-to-SR verification table and the truth table of an SR flip-flop. Click to enlarge.

The figure shows that all the entries in the first, second, third, and eighth columns (shaded in beige) of the D-to-SR verification table are consistent with those present in the truth table of the SR flip-flop. The last two rows appear to differ, but they can be considered equivalent because an SR flip-flop's outputs can be either high or low as a result of the invalid input combination. Actually, we have designed a system that is better than an SR flip-flop because it has predictable output behavior when both inputs are high.

The verification table indicates that the conversion process was a success: The given D flip-flop was made functionally equivalent to the desired SR flip-flop.

Conversion of D to JK Flip-Flop

The given D flip-flop can be converted into a JK flip-flop by using a D-to-JK conversion table as shown in Figure 5. This table collectively represents the data of both the truth table of the JK flip-flop and the excitation table of the D flip-flop.

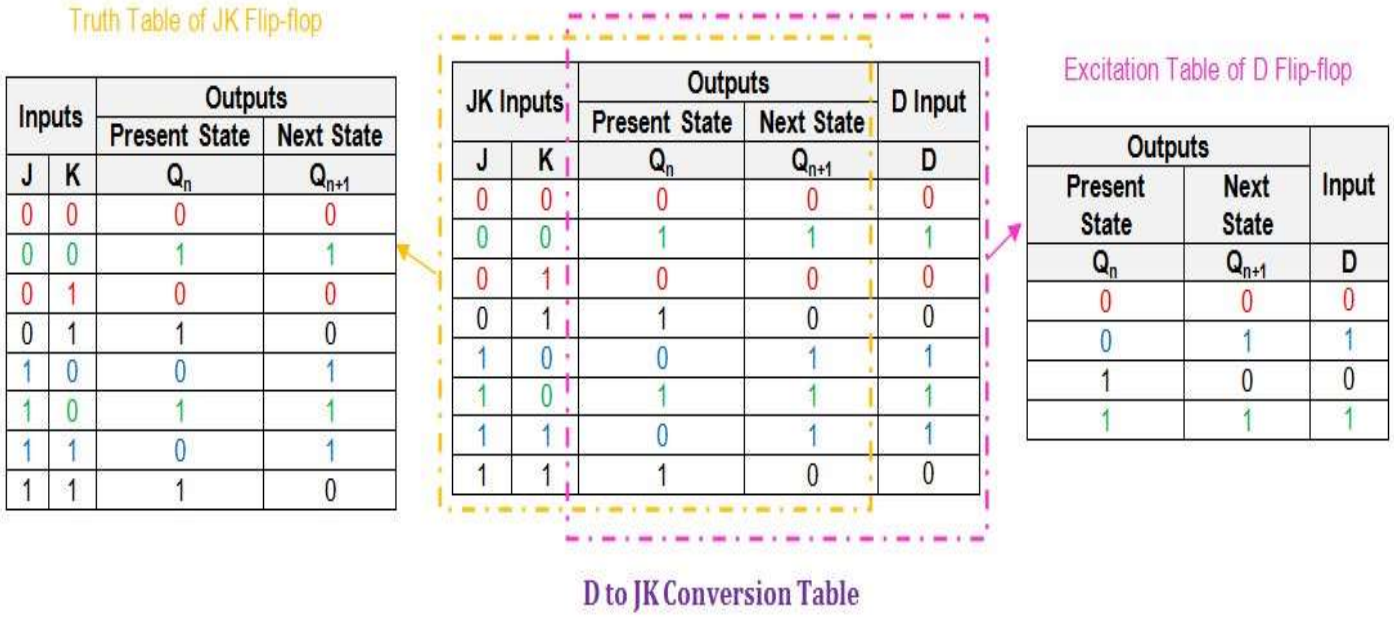


Figure 5: D-to-JK conversion table. Click to enlarge.

Following this, we need to simplify the expressions for the D-input in terms of J, K, and Q_n. We will again employ the K-map technique.

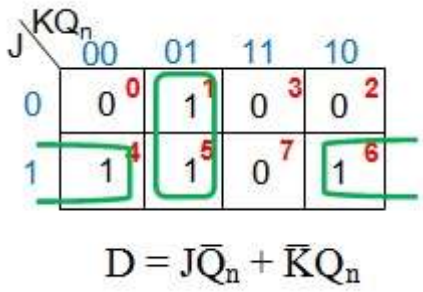


Figure 6: K-map simplification for D input in terms of J, K, and Q_n

Figure 6 shows that, in order to convert the D flip-flop into a JK flip-flop, its D input needs to be driven by the output of a two-input OR gate which has its inputs as

1. J ANDed with the negation of the present-state Q_n (i.e., $\bar{Q}_n$)
2. Negation of K ($\bar{K}$) ANDed with the present-state Q_n

This indicates that we would require

1. One NOT gate—to negate K
2. Two AND gates—one to obtain $J\bar{Q}_n$ and the other to get $\bar{K}Q_n$
3. One OR gate—to obtain the D input given by $J\bar{Q}_n + \bar{K}Q_n$

Thus, the resultant system would be as shown in Figure 7.

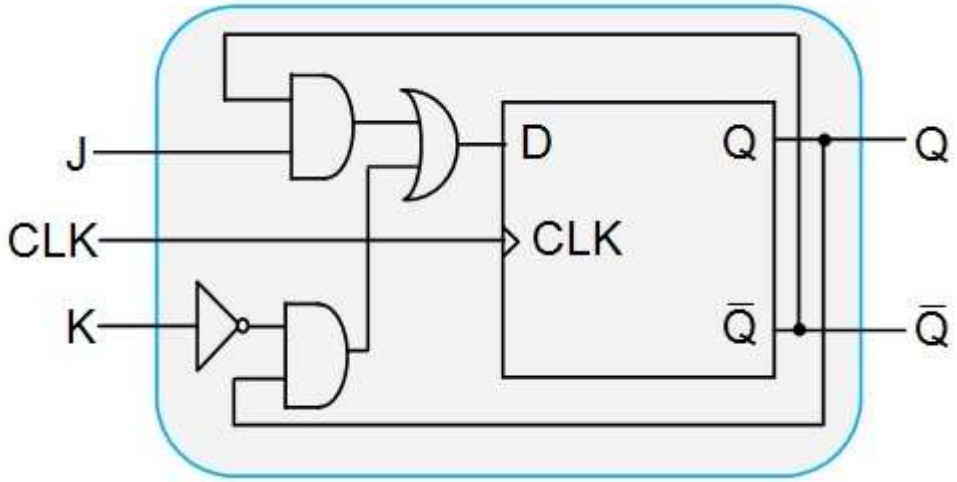


Figure 7: The D flip-flop designed to behave as a JK flip-flop

Lastly, let us verify whether our designed system behaves as we expect it to using a D-to-JK verification table, shown in Figure 8.

D to JK Verification Table

Inputs		Intermediate Inputs						Outputs	
J	K	Q	$\bar{Q}$	$\bar{K}$	$J\bar{Q}$	$\bar{K}Q$	$D = J\bar{Q} + \bar{K}Q$	Q	$\bar{Q}$
0	0	0	1	1	0	0	0	0	1
0	0	1	0	1	0	1	1	1	0
0	1	0	1	0	0	0	0	0	1
0	1	1	0	0	0	0	0	0	1
1	0	0	1	1	1	0	1	1	0
1	0	1	0	1	0	1	1	1	0
1	1	0	1	0	1	0	1	1	0
1	1	1	0	0	0	0	0	0	1

Truth Table of D Flip-Flop

Truth Table of JK Flip-Flop

Inputs		Outputs	
		Present State	Next State
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Figure 8: Comparison between the D-to-JK verification table and the truth table of a JK flip-flop. Click to enlarge.

Figure 8 shows that the first, second, third, and ninth columns of the D-to-JK verification table (shaded in beige) have entries which are identical to those in the columns of the JK flip-flop's truth table. This indicates that the given D flip-flop behaves exactly as a JK flip-flop for every combination of inputs and the present-state.

Thus, we can conclude that the conversion process was successful.

Conversion of D to T Flip-Flop

In order to convert the given D flip-flop into a T-type, we need to obtain the corresponding conversion table, as shown in Figure 9. Here, the information in the excitation table of the D flip-flop is inserted as a part of the T flip-flop's truth table.

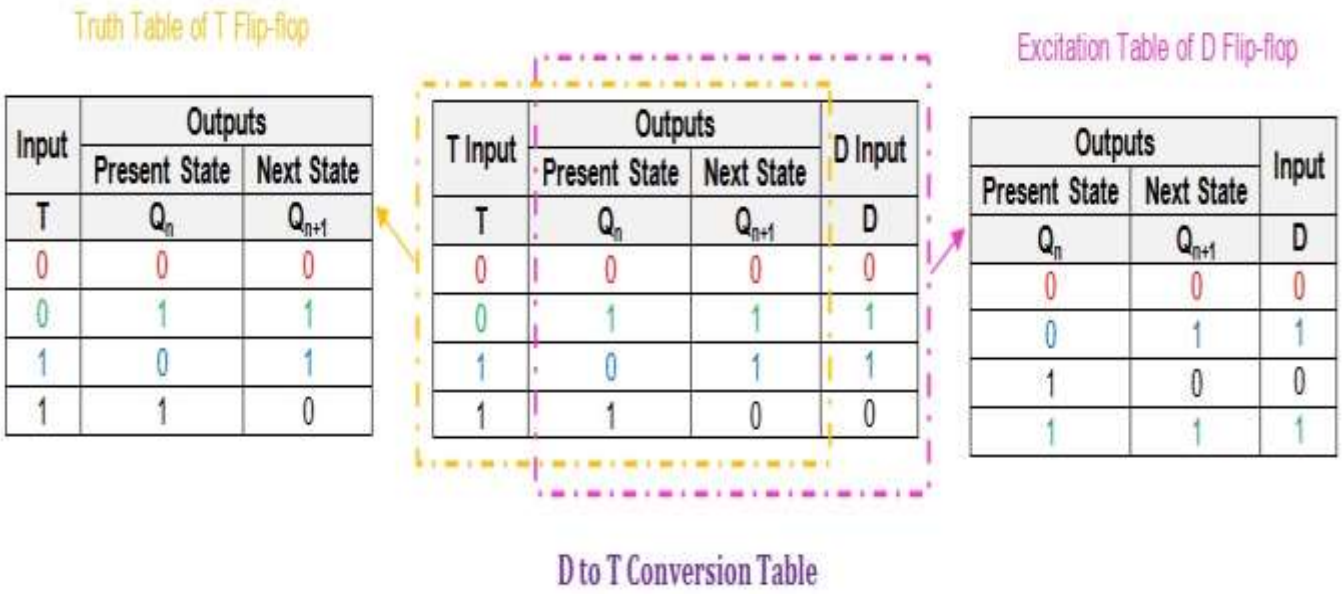


Figure 9: D-to-T conversion table. Click to enlarge.

Having obtained the conversion table, the next step is to express the input, D, in terms of T and Q_n.

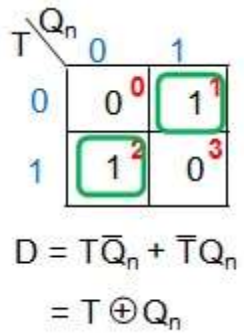


Figure 10: K-map simplification for D in terms of T and Q_n

From Figure 10, we see that in order to convert the given D flip-flop into a T-type, we need to drive its input pin (D) by the output of an XOR gate whose inputs are T and Q_n. This would lead to the new digital system which is shown in Figure 11(a).

If we must confine ourselves to only NOT, OR, and AND gates, we will need to follow these steps:

i. Use an AND gate to AND the user-defined input, T, and the negation of the flip-flop's present-state Q_n.
ii. Use another AND gate to AND the present-state of the flip-flop with the negation of T (obtained as an output of a NOT gate).
iii. OR together the outputs of the two AND gates using a two-input OR gate.

This leads to the digital system shown in Figure 11(b).

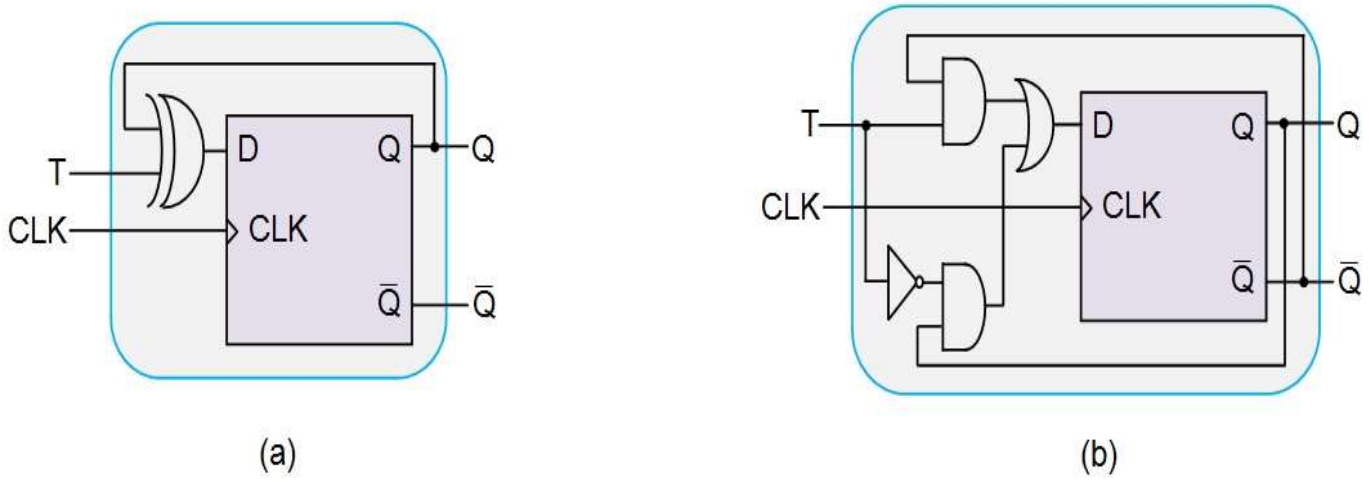


Figure 11: D flip-flop designed to behave as a T flip-flop using (a) an XOR gate and (b) only NOT, OR, and AND gates. Click to enlarge.

The next and final step is to verify the conversion process using the D-to-T verification table, shown in Figure 12.

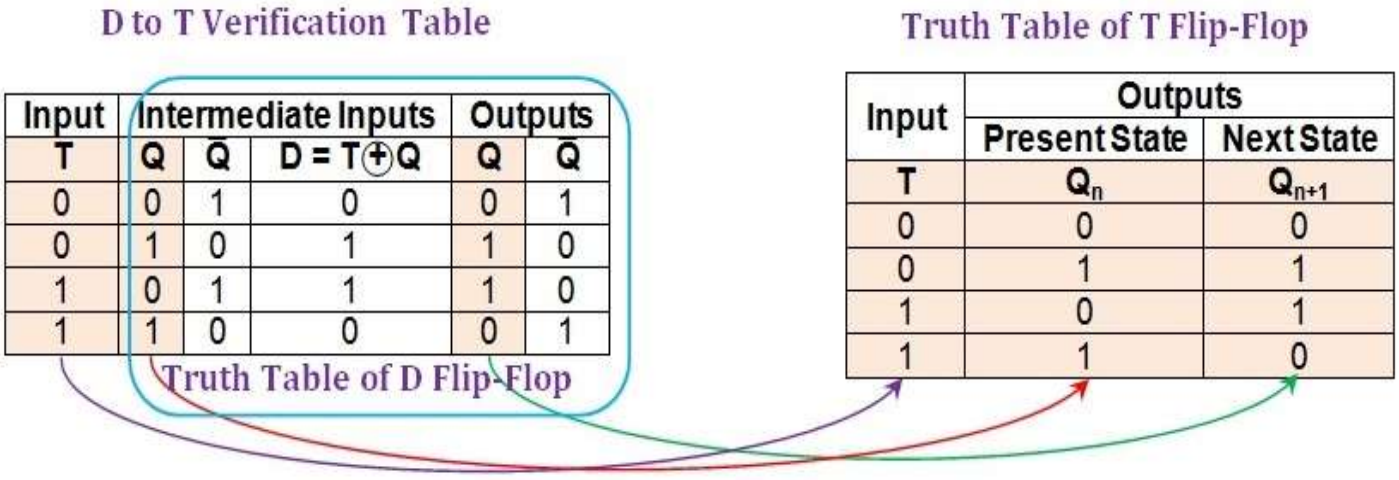


Figure 12: Comparison between the D-to-T verification table and the truth table of a T flip-flop. Click to enlarge.

From the figure, it can be seen that the first, second, and penultimate columns (shaded in beige) of the D-to-T verification table are identical to the columns in

the truth table of the T flip-flop. This indicates a successful conversion process, i.e., the given D flip-flop behaves exactly as a T flip-flop.

Although we have verified the system designed in Figure 11(a), the conclusion is valid also for the design shown in Figure 11(b) because

$$A \oplus B = \bar{A}B + A\bar{B}$$

Summary

This article presents the methodology used to convert a given D flip-flop into SR-, JK-, and T-type flip-flops. It also discusses the verification process for each of these conversions.

The next part of the series (Part V) will cover the conversion of a T flip-flop into other types.

Conversion of T Flip-Flops

September 14, 2016 by Sneha H.L.

Here we convert the given T flip-flop into SR-, JK- and D-types, and we also verify the process of conversion.

Introduction

This article covers the steps involved in converting a given T flip-flop into SR-, JK-, and D-type flip-flops. We also present a verification technique for these conversions; the verification process allows us to ensure that the designed systems provide the desired functionality.

Previous Articles in This Series

Please refer to the previous parts in this series, particularly the first two, for a detailed explanation of the process:

- Introduction to the Conversion of Flip-Flops
- SR-to-D and SR-to-T Flip-Flop Conversions
- Conversion of JK Flip-Flops
- Conversion of D Flip-Flops

Conversion of a T to an SR Flip-Flop

In order to convert a given T flip-flop into SR-type, we need to combine the information presented in the SR flip-flop's truth table and the information in the T flip-flop's excitation table into a common table. This can be referred to as a T-to-SR conversion table and is as shown in Figure 1.

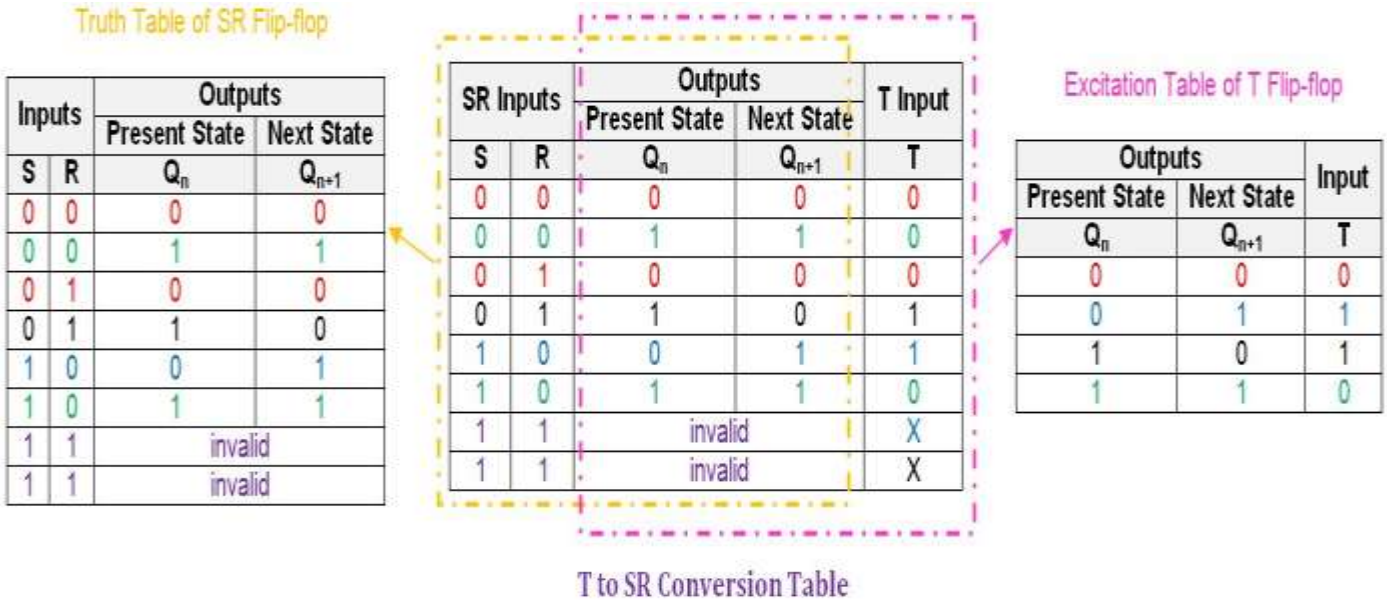


Figure 1: T-to-SR conversion table. Click to enlarge.

Notice the don't care (X) entries in the last two rows of the conversion table's "T input" column. These indicate that when both inputs (S and R) are driven high, the output of the SR flip-flop is unpredictable (owing to the "race around condition").

Next, we should express the input of the given flip-flop in terms of the present-state, Q_n, and the input(s) of the desired flip-flop. This can be done by using a suitable simplification technique, such as the K-map (discussed in detail in a separate article).

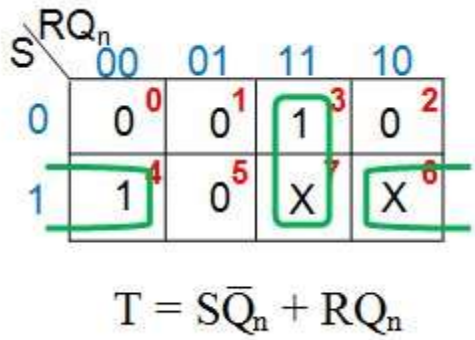


Figure 2: K-map simplification for the T input in terms of S, R, and Q_n

Figure 2 shows that the simplified logical expression for the T input in terms of S, R, and Q_n is $S\bar{Q}_n + RQ_n$.

Designing Your System

To make the given T flip-flop functionally equivalent to the desired SR flip-flop, we need to AND $\bar{Q}n$ with the user-defined input S and also AND Qn with the user-provided input R. The results of these AND operations are then ORed together.

Thus, we require two AND gates and one OR gate to convert the T flip-flop to an SR-type, as shown in Figure 3.

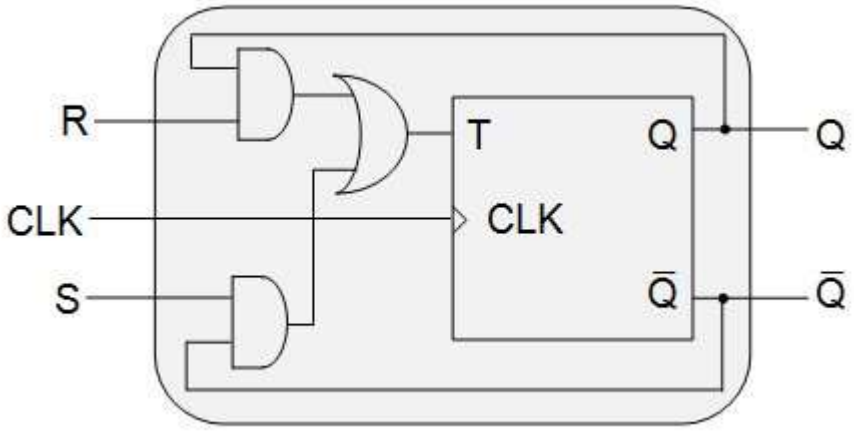


Figure 3: T flip-flop providing the functionality of an SR flip-flop

Verification

Having designed the system, we will now verify that the conversion process was successful. This can be accomplished by means of the truth table–based verification technique. The process involves comparison between the truth table of the desired (SR) flip-flop and the verification table for the T-to-SR conversion, as shown in Figure 4.

T to SR Verification Table

Inputs		Intermediate Inputs					Outputs	
S	R	Q	$\bar{Q}$	$S\bar{Q}$	RQ	$T = S\bar{Q} + RQ$	Q	$\bar{Q}$
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	1	0	1	0	0	0	0	1
0	1	1	0	0	1	1	0	1
1	0	0	1	1	0	1	1	0
1	0	1	0	0	0	0	1	0
1	1	0	1	1	0	1	1	0
1	1	1	0	0	1	1	0	1

Truth Table of T Flip-Flop

Truth Table of SR Flip-Flop

Inputs		Outputs	
		Present State	Next State
S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	invalid	
1	1	invalid	

Figure 4: Comparison between the T-to-SR verification table and the truth table of an SR flip-flop. Click to enlarge.

Figure 4 shows that the values in the first, second, third, and eighth columns (shaded in beige) of the T-to-SR verification table are consistent with those in the SR flip-flop's truth table. Thus, the conversion process was successful. The last two rows at first seem inconsistent, but they are indeed acceptable because an SR flip-flop's outputs can be either high or low when both inputs are logic high. Actually, the converted T flip-flop is better than an SR flip-flop because it has predictable output states even for the invalid input combination.

Conversion of a T to a JK Flip-Flop

We begin with the T-to-JK conversion table (see Figure 5), which combines the information in the JK flip-flop's truth table and the T flip-flop's excitation table.

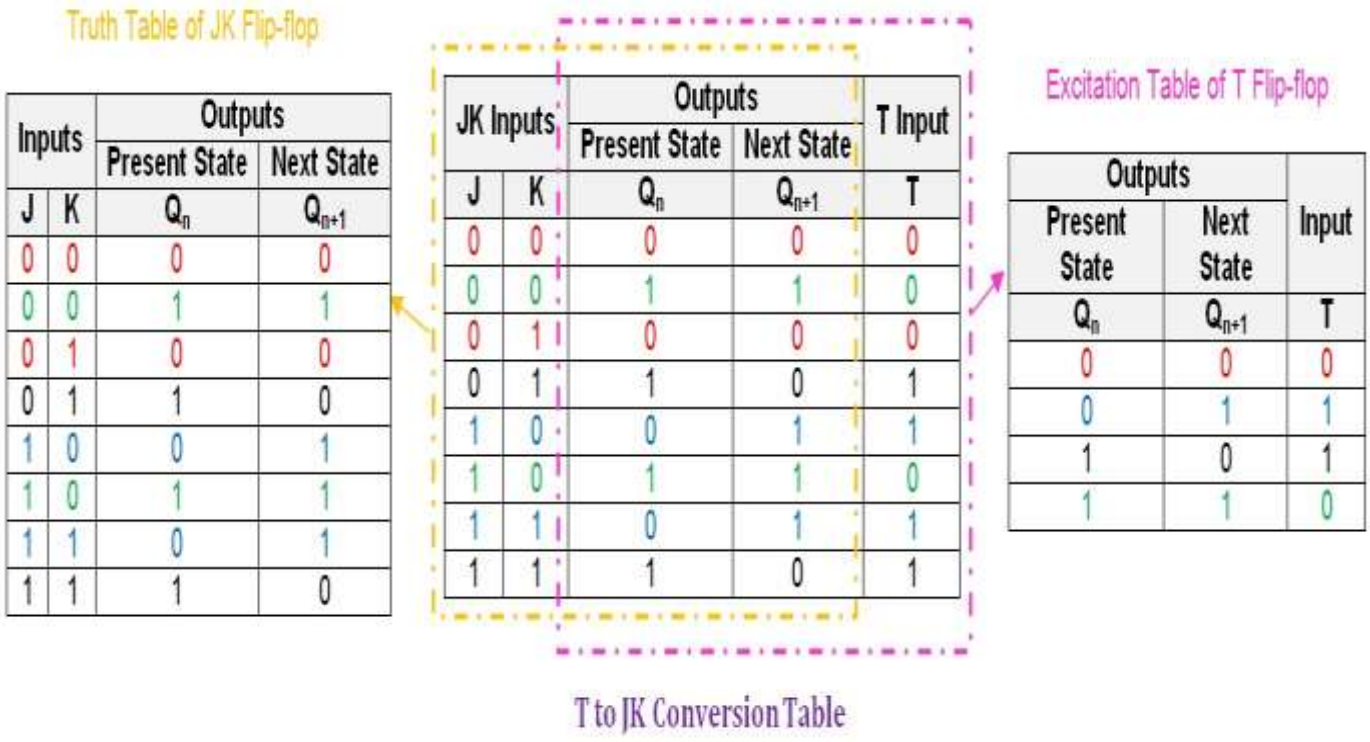


Figure 5: T-to-JK conversion table. Click to enlarge.

Next, we need to obtain the simplified Boolean expression for the T input in terms of J, K, and Q_n.

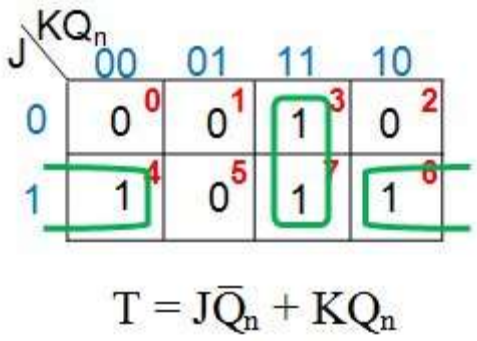


Figure 6: K-map simplification for the T input in terms of J, K, and Q_n

Figure 6 shows the expression for the T input as $J\overline{Q}_n + KQ_n$. This means that to convert the T flip-flop into a JK flip-flop, the T input is driven by the output of a two-input OR gate which has as inputs

1. J ANDed with the negation of the present-state Q_n, i.e., $\overline{Q}_n$
2. K ANDed with the present-state, Q_n

Thus, we will need two AND gates and one OR gate, as shown in Figure 7.

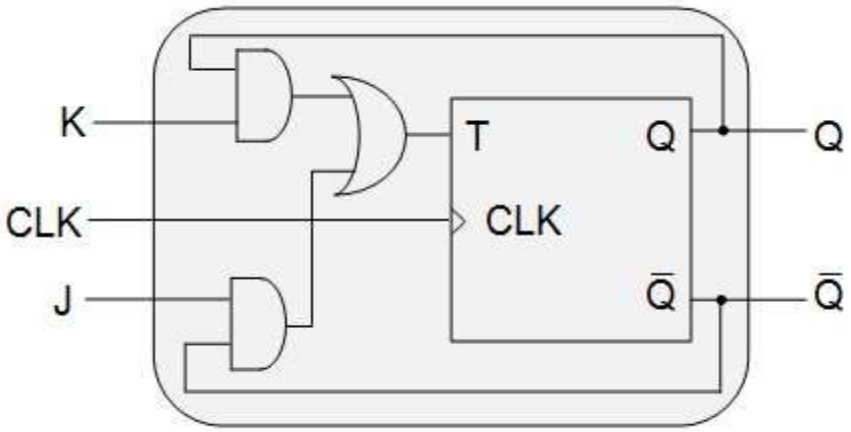


Figure 7: T flip-flop designed to behave as a JK flip-flop

The final step is to verify whether the system behaves as we expect it to. This can be done using a T-to-JK verification table, shown in Figure 8. Here we can compare the entries in the verification table to the truth table of the JK flip-flop.

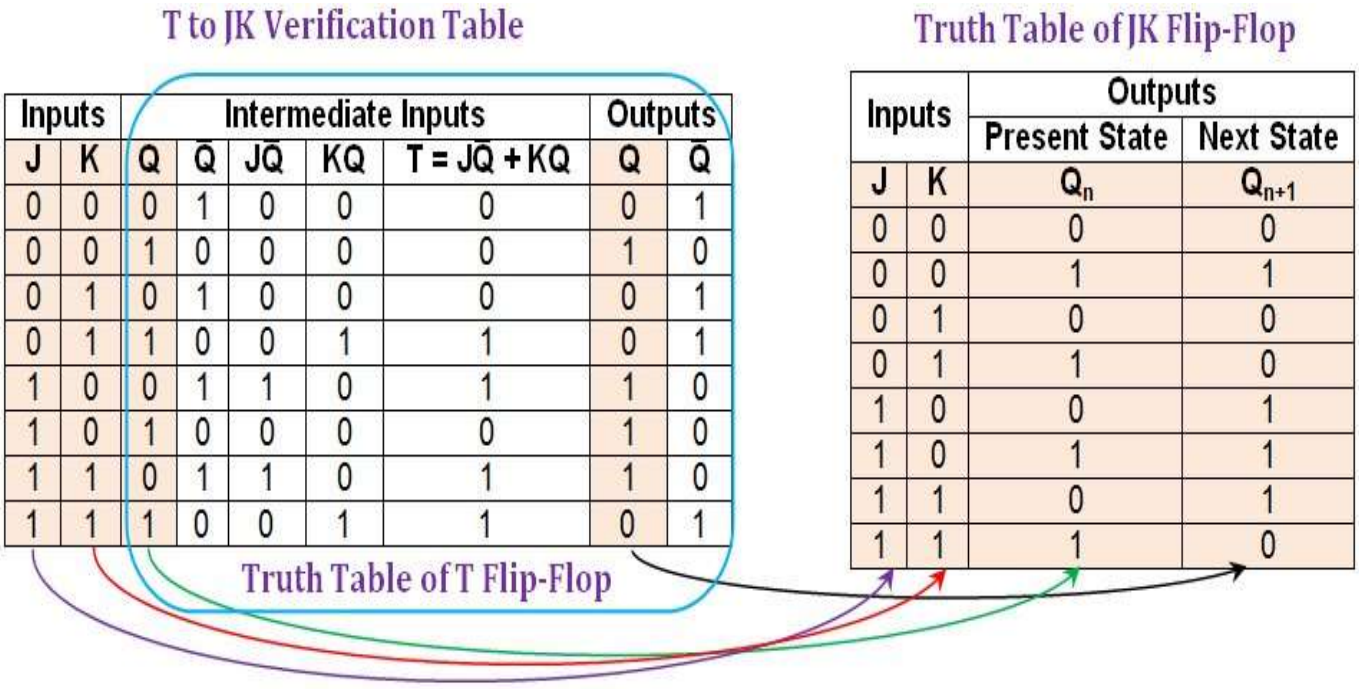


Figure 8: Comparison between the T-to-JK verification table and the truth table of a JK flip-flop. Click to enlarge.

The entries in the first, second, third, and eighth columns (shaded in beige) of the T-to-JK verification table agree with those in the truth table of the JK flip-flop. This indicates that the given T flip-flop has become functionally equivalent to the desired JK flip-flop.

Conversion of a T to a D Flip-Flop

We begin by writing the T-to-D conversion table (see Figure 9).

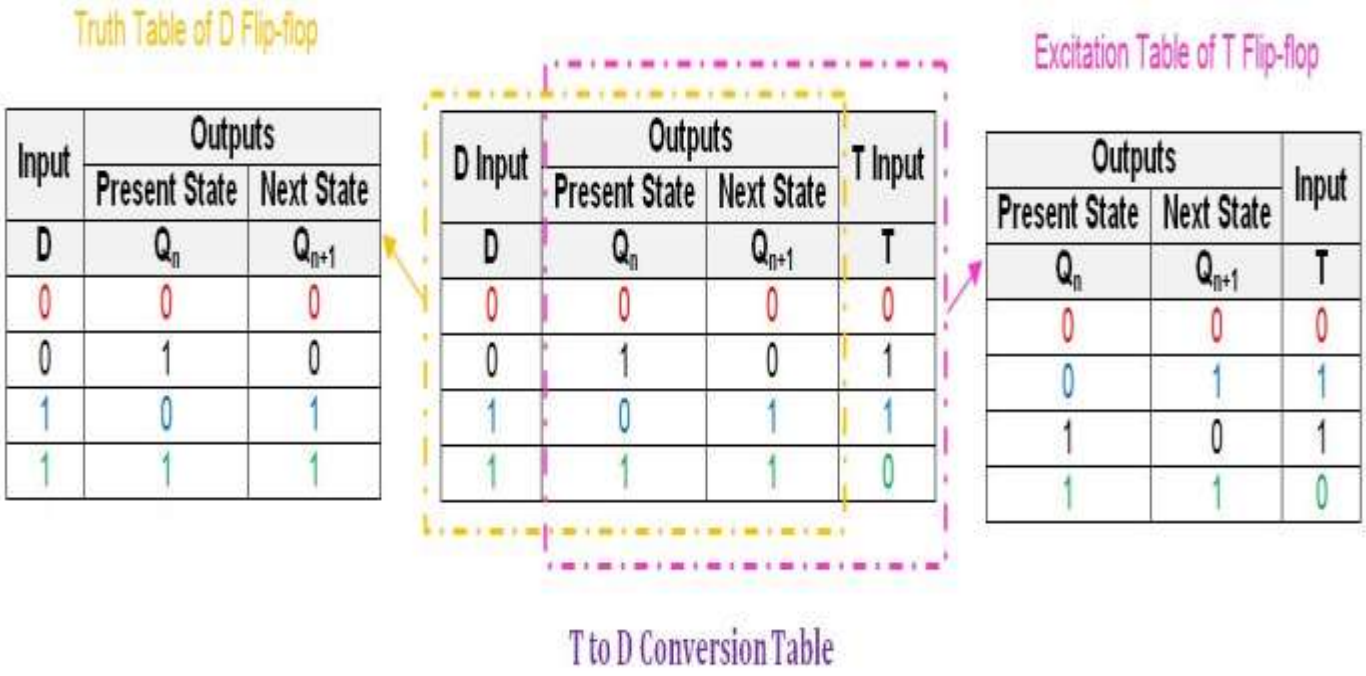
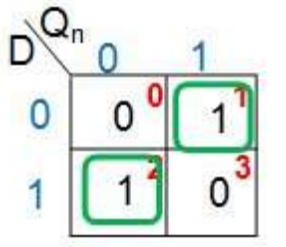


Figure 9: T-to-D conversion table. Click to enlarge.

Once this is done, we need to express the input, T, in terms of the user-defined input, D, and the flip-flop's present-state, Q_n. We will again use the K-map simplification technique.



$$T = D\bar{Q}_n + \bar{D}Q_n$$

$$= D \oplus Q_n$$

Figure 10: K-map simplification for the T input in terms of D and Q_n

Figure 10 shows that, in order to make the given T flip-flop functionally equivalent to a D flip-flop, we need to drive its input pin, T, with the output of an XOR gate whose inputs are D and Q_n. This will lead to the new digital system shown in Figure 11(a). Figure 11(b) shows a system which is functionally equivalent to that of Figure 11(a) but is designed using only NOT, AND, and OR gates.

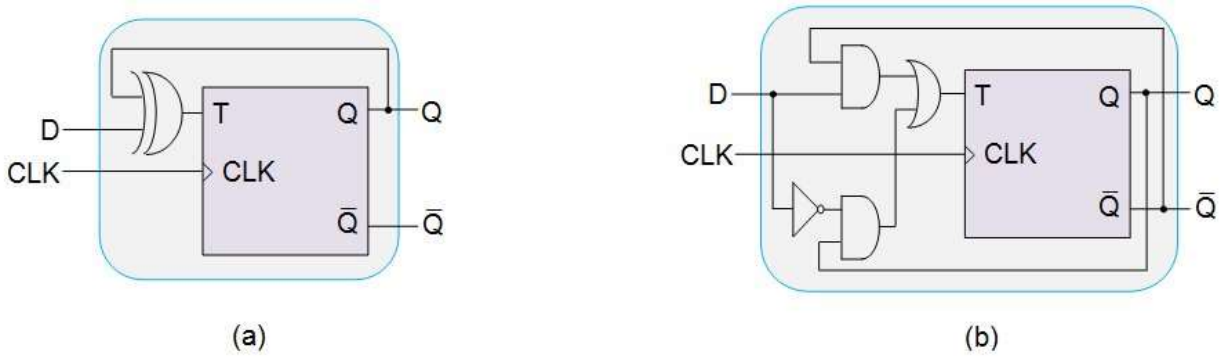


Figure 11: T flip-flop designed to behave as a D flip-flop using (a) an XOR gate and (b) NOT, AND, and OR gates

Finally, in order to ensure that the designed system behaves as expected, we will write a T-to-D verification table, shown in Figure 12.

T to D Verification Table

Input	Intermediate Inputs			Outputs	
D	Q	Q̄	T = D⊕Q	Q	Q̄
0	0	1	0	0	1
0	1	0	1	0	1
1	0	1	1	1	0
1	1	0	0	1	0

Truth Table of T Flip-Flop

Truth Table of D Flip-Flop

Input	Outputs	
	Present State	Next State
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Figure 12: Comparison between the T-to-D verification table and the truth table of a D flip-flop. Click to enlarge.

Figure 12 shows that the entries in the first, second, and fifth columns (shaded in beige) of the T-to-D verification table are the same as those in the D flip-flop's truth table. Thus, we have successfully converted the given T flip-flop into a D-type flip-flop.

Note that although we have presented the verification for the system shown in Figure 11(a), the conclusion is valid for the system in Figure 11(b) because the two systems are logically equivalent:

$$A \oplus B = \overline{A}B + A\overline{B}$$

Summary

In this article, we have explained the process of converting a T flip-flop into SR-, JK-, and D-type flip-flops. We have also verified our new systems in order to confirm that the conversion processes were successful.