El Naga’s Transitions Technique and its Advantages Compared to the Conventional Method of Designing Sequential Circuits

Nagi M. El Naga, Halima M. El Naga
California State University, Northridge/California State Polytechnic University, Pomona

Abstract

In most textbooks of logical circuits’ design, only one logical sequential circuits design method has been documented. In this paper, it will be referred to this method as the conventional or traditional method. This conventional method of designing sequential circuits does not provide a mean for the designer to check the correctness of his design until the implementation phase. In this case, if the design fails to do the desired function, the designer has to go through a debugging phase of both the design and the implementation, which could be a big waste of the designer time. In this paper, El Naga's Transitions technique, as a method of designing logical sequential circuit will be first introduced and its advantages compared to the conventional method are presented. This technique is based on the use of the four transitions: α, the transition from 0 to 1, β, the transition from 1 to 0, *I*, the transition from 1 to 1, and φ, the transition from 0 to 0. This technique provides the designer of logical sequential circuits with various testing algorithms that check the correctness of almost every step in the design procedure. If the provided testing algorithms are followed after each step of the design, the final design will almost be error free. The testing algorithms are presented and discussed.

Although, asynchronous counters have been discussed in almost every logical circuits design textbook, they all failed to provide the readers with a systematic procedure to design them. This is due to the fact that the conventional method lacks the means to provide such procedure. Based on El Naga's Transitions technique presented in this paper, a systematic procedure to design asynchronous counters can be driven.

I. Introduction

In this section, a design of a simple synchronous sequential circuit, a sequencer, using the traditional method will be presented and its disadvantages will then be discussed. Starting the design from the state table shown in Figure 1, the design steps using the traditional method are as follows:

Present State				Next State			
A	B	C	D	A	B	C	D
0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	0	1	0
1	0	1	0	1	0	0	1
1	0	0	1	0	0	1	0
0	0	1	0	1	0	0	0
1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	1
0	1	0	1	0	0	0	0

Figure 1. Sequencer State Table

Step 1: Identify each of the present states (incount states) using a minterm.
Step 2: Determine the general don't care terms (out of count states).
Step 3: Decide on the type of flip-flops to be used in the design. In this example, JK flip-flop is assumed to be selected.
Step 4: Using the excitation table for the selected type of flip-flop, develop an input table for each flip-flop (A, B, C, D) used in the design. Each input table will contain as many columns as the selected type of flip-flop has data inputs. For this example, a column for the *j* input and a column for the *k* input need to be developed. These columns are filled with the binary values (0, 1, x) to be applied to each input to achieve the transition from the present state to the next state of the flip-flop.
Step 5: Construct a Karnaugh map for each input of each flip-flop.
Step 6: From these maps, using the standard function minimization processes, derive the optimum input equations for each flip-flop.
Step 7: Draw the schematic diagram.

The results of Step 1 and Step 4 are shown in Figure 2. The general don't care terms (out of count states) of Step 2 are given by the following equation:

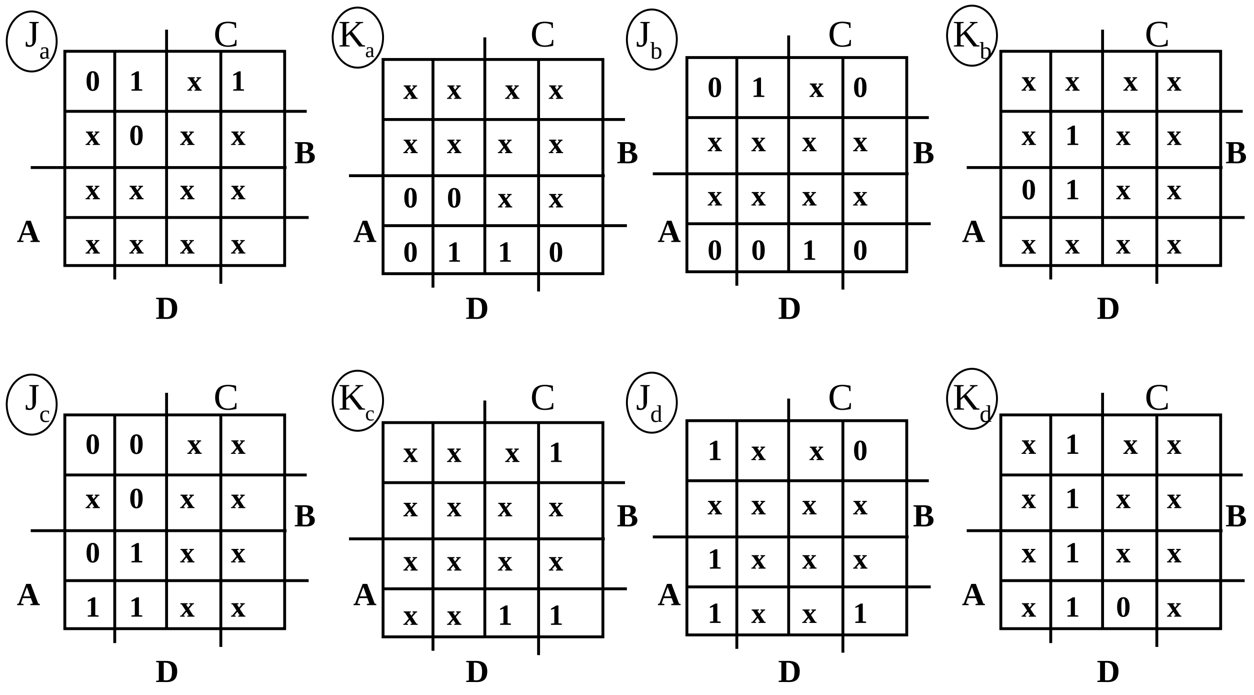
General Don't Care Terms = $P_3 + P_4 + P_6 + P_7 + P_{14} + P_{15}$ ……..…………(1)

The Karnaugh maps of the flip-flops inputs, the results of Step 5, are shown in Figure 3. The flip-flops input equations driven from these maps are given below:

$j_a = C + \overline{B}D$ ………(2) $k_a = \overline{B}D$ ……….(3)
$j_b = CD + \overline{A}D$ ………(4) $k_b = D$ ……..…(5)
$j_c = AD + A\overline{B}$ ………(6) $k_c = 1$ ………...(7)
$j_d = A + \overline{C}$ ………(8) $k_d = \overline{C}$ ……….. (9)

	Present State				Next State				A		B		C		D	
	A	B	C	D	A	B	C	D	j_a	k_a	j_b	k_b	j_c	k_c	j_d	k_d
P_0	0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
P_1	0	0	0	1	1	1	0	0	1	x	1	x	0	x	x	1
P_{12}	1	1	0	0	1	1	0	1	x	0	x	0	0	x	1	x
P_{13}	1	1	0	1	1	0	1	0	x	0	x	1	1	x	x	1
P_{10}	1	0	1	0	1	0	0	1	x	0	0	x	x	1	1	x
P_9	1	0	0	1	0	0	1	0	x	1	0	x	1	x	x	1
P_2	0	0	1	0	1	0	0	0	1	x	0	x	x	1	0	x
P_8	1	0	0	0	1	0	1	1	x	0	0	x	1	x	1	x
P_{11}	1	0	1	1	0	1	0	1	x	1	1	x	x	1	x	0
P_5	0	1	0	1	0	0	0	0	0	x	x	1	0	x	x	1

Figure 2. Sequencer State and Flip-Flops Input Tables



J_a C B A D
0 1 x 1 / x 0 x x / x x x x / x x x x
K_a C B A D
x x x x / x x x x / 0 0 x x / 0 1 1 0
J_b C B A D
0 1 x 0 / x x x x / x x x x / 0 0 1 0
K_b C B A D
x x x x / x 1 x x / 0 1 x x / x x x x
J_c C B A D
0 0 x x / x 0 x x / 0 1 x x / 1 1 x x
K_c C B A D
x x x 1 / x x x x / x x x x / x x 1 1
J_d C B A D
1 x x 0 / x x x x / 1 x x x / 1 x x 1
K_d C B A D
x 1 x x / x 1 x x / x 1 x x / x 1 0 x

Figure 3. Maps for Flip-Flops Inputs

Using equations 2-9 to draw the schematic diagram, Step 7, completes the design of the sequencer.

Although the method explained above, or one of its variation, is the only logical sequential circuits design methodology documented in text books, it has the following drawbacks:

1. In Step 4, mistakes could be made during the development of the flip-flops' input tables. Unfortunately, there isn't any rule that could be applied to detect them. They will only be detected when the final product does not work properly after a complete implementation is done, which is very costly.
2. In Step 5, while filling up the Karnaugh maps from the flip-flops' input tables, binary information could be misplaced. This, if not detected, produces a final product that does not work properly. Unfortunately, there isn't any rule that could be applied to detect that either.
3. If for any reason, it is needed to design the above sequencer using any other type of flip-flops, steps 4 to 7 need to be repeated for the new type. This is necessary because the binary information in the flip-flops' input tables and Karnaugh maps are dependent on the type of the flip-flop.

El Naga's Transition methodology solves these problems and more as will be discussed in the following sections.

II. Basic definition:

During the transition from one state to another a flip-flop can go through one of four possible transitions, which are defined as:

1. α, the transition from 0 to 1,
2. β, the transition from 1 to 0,
3. *I,* the transition from 1 to 1, and
4. φ, the transition from 0 to 0.

The transition methodology is based on the use of these four transitions. For a particular data input of a specific type of flip-flop, any of the above transition could be either *essential* transition, *don't care* transition, or *zero* transition. A transition is defined as *essential* transition for specific data input of a flip-flop if it is necessary to apply logical *1* to that input to see the flip-flop going through this transition. A transition is defined as *don't care* transition for specific data input of a flip-flop if it doesn't matter whether to apply logical *1* or *0* to that input to see the flip-flop going through this transition. A transition is defined as *zero* transition for specific data input of a flip-flop if it is necessary to apply logical *0* to that input to see the flip-flop going through this transition. For example, for an RS flip-flop to go through α transition, 0 to 1, it is necessary to apply logical 1 to the S input and logical 0 to the R input. Therefore, α transition is considered to be *essential* transition for the S input and *zero* transition for the R input.

To explain how to use this method to design synchronous sequential circuit, we need to define an excitation equation of each data input of each type of flip-flop in terms of these four transitions. The excitation equation of data input consists of two parts separated by a "+" sign. The first part represents a list of the *essential* transitions of this input. The second part starts with D.C., an abbreviation of don't care, followed by a list of the *don't care* transitions of this input included between brackets. The general don't care terms are considered as part of the *don't care* transitions list and are represented by X. Therefore, the *essential* transitions and the *don't care* transitions are included in the excitation equation while the *zero* transitions are not included at all. According to this definition, the excitation equations of all data inputs of all types of flip-flops are listed below:

1. RS flip-flop:

$$S = \alpha + \text{D.C. } [\, I \,,\ X \,] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$
$$R = \beta + \text{D.C. } [\varphi \,,\ X \,] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

2. T flip-flop:

$$T = \alpha , \beta + \text{D.C. } [\, X \,] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

3. JK flip-flop:

$$J = \alpha + \text{D.C. } [\beta \,, I \,,\ X \,] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$
$$K = \beta + \text{D.C. } [\alpha \,, \varphi \,,\ X \,] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(6)$$

4. D flip-flop:

$$D = \alpha \,, I + \text{D.C. } [\, X \,] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

For example, from Equation 5, for the J input, α is *essential* transition, β and *I* are *don't care* transitions while φ is the only *zero* transition, which is not included in the excitation equation of the input.

III. El Naga's Transitions Methodology for the Design of Synchronous Sequential Circuit

In this section, the same sequencer, designed in section I of this paper using the traditional method will be designed using the transition method. Starting the design from the state table shown in Figure 1, the design steps using the transition method are as follows:

Step 1: Identify each of the present states (incount states) using a minterm.

Step 2: Determine the general don't care terms (out of count states).

Step 3: Develop a transition table for each flip-flop (A, B, C, D) used in the design. Each transition table will contain one column regardless of the type of flip-flop to be used in the design. These columns are filled with the transitions (α,β,*I*, φ), which each flip-flop will go through from the present states to the next states.

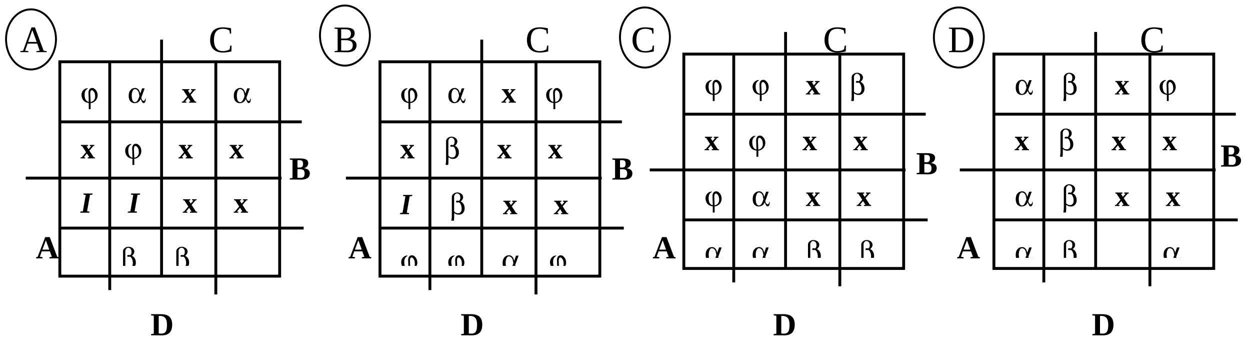
The results of Step 1 and Step 3 are shown in Figure 4. The general don't care terms of Step 2 are given by the following equation:

General Don't Care Terms = $P_3 + P_4 + P_6 + P_7 + P_{14} + P_{15}$ ……..…………(10)

Step 4: Construct a Karnaugh map for each input of each flip-flop. Each map is filled with the transitions (α,β,*I*, φ) from the corresponding flip-flop transition table. The General Don't Care terms of Equation 10 are located in the maps and represented by x's. The developed maps are shown in Figure 5.

	Present State A B C D	Next State A B C D	A	B	C	D
P_0	0 0 0 0	0 0 0 1	φ	φ	φ	α
P_1	0 0 0 1	1 1 0 0	α	α	φ	β
P_{12}	1 1 0 0	1 1 0 1	*I*	*I*	φ	α
P_{13}	1 1 0 1	1 0 1 0	*I*	β	α	β
P_{10}	1 0 1 0	1 0 0 1	*I*	φ	β	α
P_9	1 0 0 1	0 0 1 0	β	φ	α	β
P_2	0 0 1 0	1 0 0 0	α	φ	β	φ
P_8	1 0 0 0	1 0 1 1	*I*	φ	α	α
P_{11}	1 0 1 1	0 1 0 1	β	α	β	*I*
P_5	0 1 0 1	0 0 0 0	φ	β	φ	β

Figure 4. Sequencer State and Transition Tables



A			
φ	α	x	α
x	φ	x	x
I	*I*	x	x
	β	β	

B			
φ	α	x	φ
x	β	x	x
I	β	x	x
φ	φ	α	φ

C			
φ	φ	x	β
x	φ	x	x
φ	α	x	x
α	α	β	β

D			
α	β	x	φ
x	β	x	x
α	β	x	x
α	β		α

Figure 5. Maps for the Sequencer

It should be noticed that all the work done in the previous four steps depends only on the sequencer, which is defined by the state table, and has nothing to do with the type of flip-flop that will be selected for the design. Therefore, unlike the traditional design method, the change of the type of flip-flop at any time later on will not require the repetition of any of the above steps, which is one of the great advantages of this method.

Step 5: Decide on the type of flip-flops to be used in the design. In this example, JK flip-flop is selected.

Step 6: From the maps developed in Step 4 and shown in Figure 5, derive the optimum input equations for each flip-flop.

The process of optimizing a function using a map filled with transition is very similar to that of a map filled with binary values (0, 1, x). In case of a map filled with transition, the *essential* transitions of the flip-flop input under consideration are treated the same way as *1*'s are treated in a map filled with binary values. At the same time, the *don't care* transitions of the flip-flop input under consideration, together with the General Don't Care terms, are treated the same way as x's are treated in a map filled with binary values. I.e. the adjacent *essential* transition entries should be combined together, with the help of the *don't care* transitions and the General Don't Care terms, in the minimum number of groups. Each group should be as large as possible the same way as *1*'s are grouped in a map filled with binary values, and the way these groups are read is also the same.

To demonstrate this process, the map of flip-flop B is selected from Figure 5. From Equation 5, α is the *essential* transition and β, ***I***, and **x** are the don't care for the **j** input. There are two *essential* transitions, α's, which are grouped as shown in Figure 6. That will produce the optimum function given in Equation 11. On the other hand, from Equation 6, β is the *essential* transition and α, φ, and **x** are the don't care for the **k** input. There are two *essential* transitions, β's, which are grouped as shown in Figure 7 to produce the input function given in Equation 12. Although two copies of the map are used here to demonstrate how the flip-flop's input functions are driven, only one copy of the map is indeed needed to drive all the input functions for all types of flip-flops.

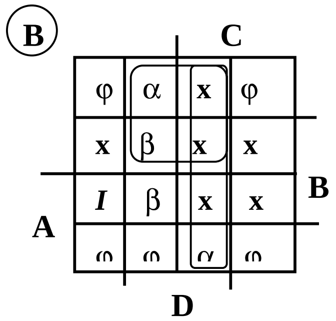


Figure 6. J_b input Equation

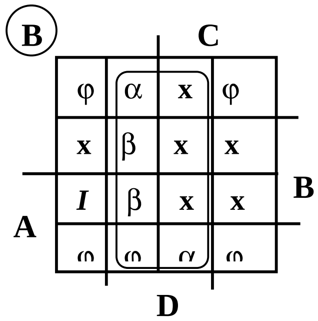


Figure 7. K_b input Equation

$j_b = CD + \overline{A}D$ ....(11)

$k_b = D$ ....(12)

Applying this procedure on the other three maps in Figure 5, the following set of flip-flop input functions will be driven:

$j_a = C + \overline{B}D$ ………(13) $k_a = \overline{B}D$ ……….(14)

$j_b = CD + \overline{A}D$ ………(15) $k_b = D$ ……..…(16)

$j_c = AD + A\overline{B}$ ………(17) $k_c = 1$ ………...(18)

$j_d = A + \overline{C}$ ………(19) $k_d = \overline{C}$ ……….. (20)

Step 7: Use the above set of equations, draw the schematic diagram, which is the last step in the design process.

IV. Transition Method’s Design Verification Rules

The Transition method provides a number of rules that can be used to verify the correctness of the work done in the major steps of the design. After constructing the transition table, Step 3, the following rules could be applied to check the correctness of the information tables in case of a **closed loop** counter or sequencer:

1. For every α transition there must be β transition. Therefore, in every transition table the number of α transition and number of β transition should be the same. If they are different, this means there is an error that needs to be corrected.
2. Since a flip-flop can not go through a α transition twice without going through a β transition in between, then in each transition table α and β transitions should alternate.
3. After going through a α transition, the flip-flop will contain “**1**”. Therefore, in each transition table α transitions can only be followed by β or ***I*** transitions. Similarly, β transitions can only be followed by α or φ transitions.

After constructing and filling a Karnaugh map with the transitions, it is possible to check if any of these transitions is misplaced. For example, for the Karnaugh map of flip-flop B, B divides the maps into two halves. One half represent B and the other half represent the complement of B ($\overline{B}$). In all the states in the half that represents B, B has a value 1. During the transition from any of these states, this value of 1 can stay 1, ***I*** transition, or change to 0, β transition. Therefore, in this half of the map only β and ***I*** transitions and **X** should be found. If α or φ transition is found in this half this would be an indication that it is misplaced. Similarly, in the second half of the map only α and φ transitions and **X** should be found. If β or ***I*** transition is found in this half this would be an indication that it is misplaced.

If any of the above rules fails, it means there is an error that needs to be corrected.

V. Conclusion

In this paper, El Naga's Transitions technique, as a method of designing logical sequential circuit is introduced and compared to the conventional method. This transition method has some advantages compared to the traditional method, which are summarized in the following:

1. Once it is well understood, the Transition method requires much less and easier work compared to that of the traditional method. The design of a sequencer using all types of flip-flops requires the construction of six maps for each flip-flop if the traditional method is used, while only one map is required if the Transition method is used. In this case, all input functions of all types of flip-flops can be driven out of this single map.

2. The Transition method provides a number of rules that can be used to verify the correctness of the work done in the major steps of the design, which are discussed above, while the traditional method does not provide any. This could save a lot of time and work.

3. The traditional method lacks the means to provide a systematic procedure to design asynchronous counters, while based on El Naga's Transitions technique presented in this paper, a systematic procedure to design asynchronous counters can be driven.

Bibliography

1. Nagi M. El Naga, Lecture Notes, Electrical and Computer Engineering Department, California State University, Northridge, California, 1999.
2. John F. Wakerly, Digital Design Principles & Practices, Third Edition, Prentice Hall, 2000.
3. Charles H. Roth, Jr., Fundamentals of Logic Design, West Publishing Company, 1992.

NAGI ELNAGA
Nagi El Naga is a Professor of Electrical and Computer Engineering at California State University, Northridge. His research interests include computer architecture, computer arithmetic, multiprocessors performance evaluation and Error correcting and Detecting Systems. Worked as a consultant in the area of digital system design. He received his B.S. and M.Sc. degrees in Electrical Engineering from Ain Shams University in 1970 and 1974 and his Ph.D. in Computer Engineering from the University of Waterloo, Canada in 1978.

HALIMA El NAGA
Halima El Naga is an Associate Professor of Electrical and Computer Engineering at California State Polytechnic University, Pomona. Her research interests include parallel processing, computer architecture and memory systems for shared memory multiprocessors. She received her MS in Electrical Engineering from California State University, Northridge in 1987and her Ph.D. in Computer Engineering from the University of Southern California in 1999.

V. Conclusion