An Efficient Methodology to Design Multi-branch Sequencers Using El Naga's Transitions Technique

Nagi M. El Naga,
California State University, Northridge

Halima M. El Naga
California State Polytechnic University, Pomona

Abstract

Designing a multi-branch sequencer using the conventional method of designing sequential circuits is a very long process that might take few hours and it does not provide a mean for the designer to check the correctness of his/her design until the implementation phase. Using the conventional method, it is very hard for teachers to present this subject and very hard for the students to understand it. In this paper, an efficient procedure to design these type of sequencers based on the use of El Naga's Transitions technique is 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 procedure cuts the design time by more than 80%. This technique also 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 innovated design technique that is presented in this paper makes the process of designing, teaching and understanding multi-branch sequencers much simpler.

I. Introduction

In this section, the four transitions used in El Naga's Transitions technique [1] are first presented. Then, the excitation equation of each data input of each type of flip-flop is derive n in terms of these four transitions [2].

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 transitions could be either an *essential* transition, *don't care* transition, or a *zero* transition. These are explained in the following:

1. ***Essential Transition*****:** A transition is defined as an *essential* transition for a specific data input of a flip-flop if it is necessary to apply a logical *1* to that input to see the flip-flop go through this transition.
2. ***Don't care Transition*****:** A transition is defined as a *don't care* transition for a specific data input of a flip-flop if it doesn't matter whether to a logical *1* or a logical *0* is applied to that input to see the flip-flop go through this transition.
3. ***Zero Transition*****:** A transition is defined as a *zero* transition for a specific data input of a flip-flop if it is necessary to apply a logical *0* to that input to see the flip-flop go through this transition.

For example, to see an RS flip-flop go through α transition, 0 to 1, it is necessary to apply a logical 1 to the S input and a logical 0 to the R input. Therefore, the α transition is considered an *essential* transition for the S input and a *zero* transition for the R input.

The excitation equation of a data input of a flip-flop 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:

$$S = \alpha + \text{D.C. } [I, X] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$
$$R = \beta + \text{D.C. } [\varphi, X] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

2. T flip-flop:

$$T = \alpha, \beta + \text{D.C. } [X] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

3. JK flip-flop:

$$J = \alpha + \text{D.C. } [\beta, I, X] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$
$$K = \beta + \text{D.C. } [\alpha, \varphi, X] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

4. D flip-flop:

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

For example, in Equation 4, for the J input, α is an *essential* transition, β and *I* are *don't care* transitions. φ is the only *zero* transition, and, as such is not included in the excitation equation of the input.

II. El Naga's Transitions Methodology for the Design of Synchronous Sequential Circuit [1]

In this section, the steps to design a closed loop synchronous sequencer using the El Naga's transition method are reviewed [1]. These steps are:

Step 1: Identify each of the states (incount states) using a minterm.
Step 2: Determine the general don't care terms (out of count states).
Step 3: Construct a Karnaugh map for each flip-flop. Each map is filled with the transition (α,β,*I*, φ) the corresponding flip-flop will go through in every state. The General Don't Care terms are located in the maps and represented by x's.
Step 4: Decide on the type of flip-flops to be used in the design.
Step 5: From the maps developed in Step 4, derive the optimum input equations for each flip-flop.

The process of optimizing a function using a map filled with transitions is very similar to that of a map filled with binary values (0, 1, x). In case of a map filled with transitions, the *essential* transitions of the input of the flip-flop 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.

III. The Design of a Multi-branch Sequencers Using El Naga's Transitions Technique

In this section, the procedure to design a multi-branch synchronous sequencer using El Naga's Transition technique is presented. To demonstrate this design methodology, the sequencer defined by the state diagram shown in Figure 1 will be considered. This state diagram could represent a multi-cycle control unit of a digital computer. In this sate diagram, the state "0001" has three possible next states depending on the values of the control signals S_1 and S_2. This state will be identified as the "*branching state*". With exception of the branching state each of the other states has only one next state. The control signals S_1 and S_2 have only effect on the branching state. At the branching state, if S_1 and S_2 have the value "11" the next state will be "0110", if they have the value "10" the next state will be "0011" and if they have the value "01" the next state will be "1100". In this case, it is assumed that S_1 and S_2 will never be set to the value "00". Therefore, the term $\overline{S}_1\ \overline{S}_2$ is considered as a don't care term for the control signals S_1 and S_2.

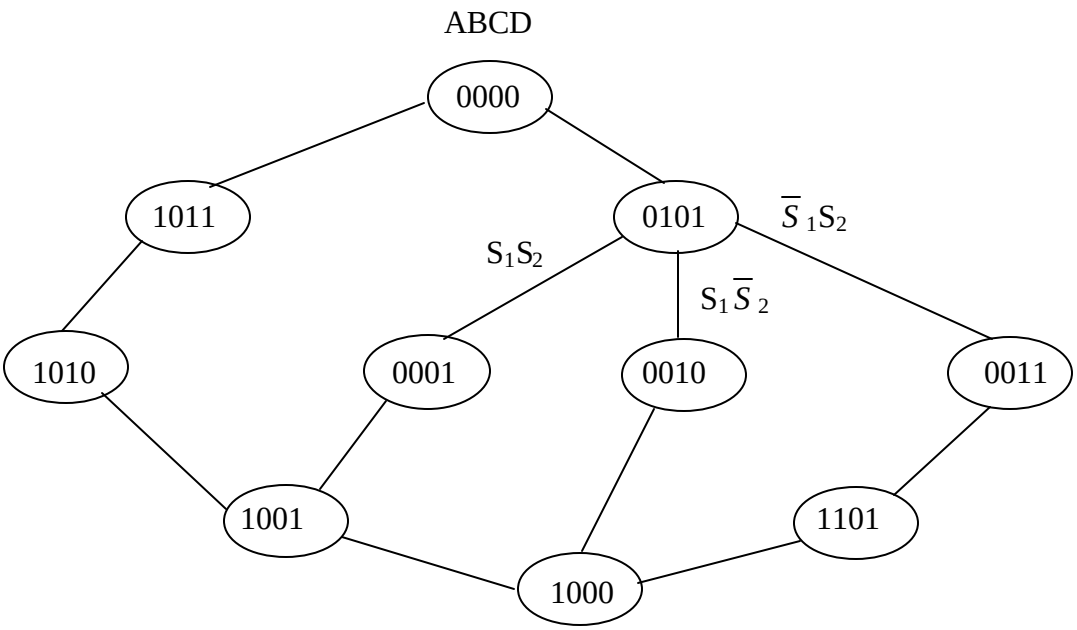


Figure 1. Sequencer State Diagram

The procedure to design a multi-branch synchronous sequencer using El Naga's Transition technique consists of the following steps:

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

The result of this step is shown in Figure 2.

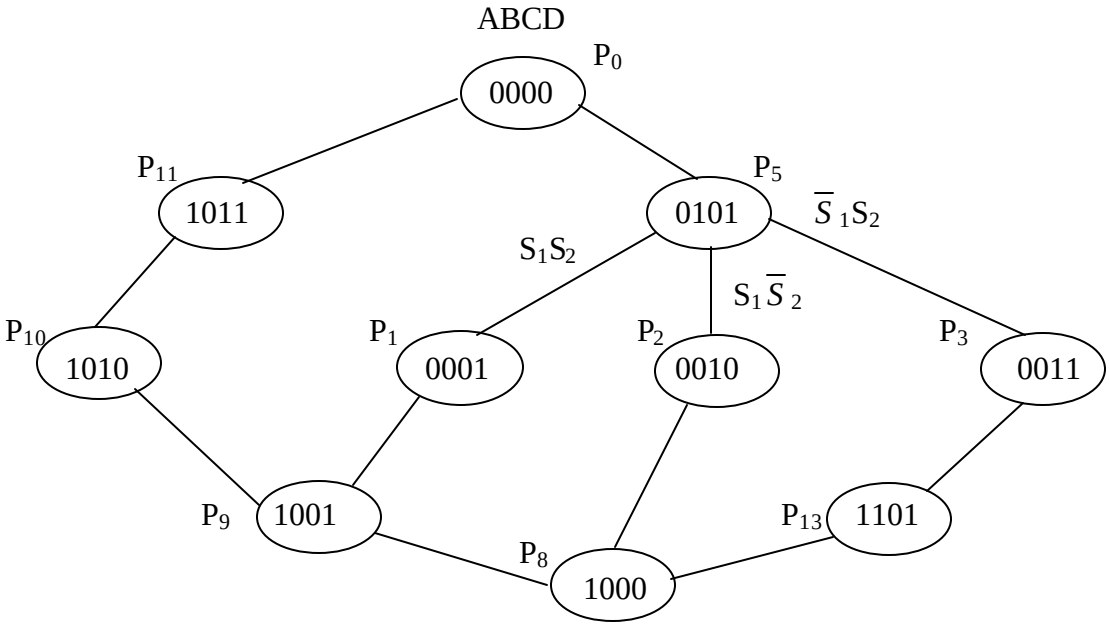


Figure 2. Sequencer State Diagram

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

The general don't care (G.D.C.) terms for this sequencer are given below.

$$G.\,D.\,C. = P_4, P_6, P_7, P_{12}, P_{14}, P_{15}$$

Step 3: *Construct a Karnaugh map for each flip-flop. Each map is filled with the transition (a,b,I, j) the corresponding flip-flop will go through in every state. The General Don't Care terms are located in the maps and represented by x's.*

With exception of the branching state, each of the other states has only one next state. Therefore, for each of these states, each flip-flop will go through one of the four transitions depending on the value of the flip-flop in that state and its value in the next state. This transition should be located in the corresponding state location in the Karnaugh map of the flip-flop.

At the branching state, a flip can go through the same transition in all branches if it has the same value in all next states. In this case, this transition, which could be any of the four transition (α, β, I, φ), should be located in the branching state location in the Karnaugh map of the flip-flop. In this case, the branching control signals have no effect on this flip-flop and the Karnaugh map of this flip-flop should be treated as if there is no branching. For an example, at the branching state flip-flop **A** goes through j transition in all branches since its value in the branching state is "**0**" and its value in all next states is also "**0**". Similarly, at the branching state flip-flop **B** goes through b transition in all branches since its value in the branching state is "**1**" and has the same value, "**0**", in all next states.

On the other hand, a flip-flop can go through two different transitions at the branching state if it has different values in the next states. In this case, regardless of how many branches there are, if the flip-flop has a value "**0**" in the branching state the two transitions will be a and j and if the flip-flop has a value "**1**" in the branching state the two transitions will be b and **I.** In this case, a zero is located in the branching state location in the Karnaugh map of the flip-flop and the two transitions, with each one ANDed with its control expression, are written beside the map. The control expression of any transition is the some of the signals that control the branches through which the flip-flop will go through this transition. For an example, at the branching state flip-flop **C** goes through "j " transition in the branch controlled by S_1S_2 and goes through "a " transition in the other two branches. Therefore, the control expression of each of these two transitions is as follows:

$$j \;\; S_1S_2$$
$$a \;\; (S_1\overline{S}_2 + \overline{S}_1S_2)$$

By using the don't care term $\overline{S}_1\,\overline{S}_2$, these will simplify to:

$$j\ S_1S_2$$
$$a\ (S_1\overline{S}_2 + \overline{S}_1S_2 + \overline{S}_1\ \overline{S}_2\ ) \Longrightarrow a\ (\ \overline{S}_1 + \overline{S}_2\ )$$

Similarly, at the branching state flip-flop **D** goes through "b" transition in the branch controlled by $S_1\overline{S}_2$ and goes through "I" transition in the other two branches. Therefore, the control expression of each of these two transitions is as follows:

$$b\ S_1\overline{S}_2$$
$$I\ (S_1S_2 + \overline{S}_1S_2\ )$$

By using the don't care term $\overline{S}_1\ \overline{S}_2$ these will simplify to:

$$b\ (S_1\overline{S}_2 + \overline{S}_1\overline{S}_2) \Longrightarrow b\ (\overline{S}_2)$$

$$I\ (S_1S_2 + \overline{S}_1S_2\ ) \Longrightarrow I\ (S_2)$$

The result of applying this step to the sequencer is shown in Figure3.

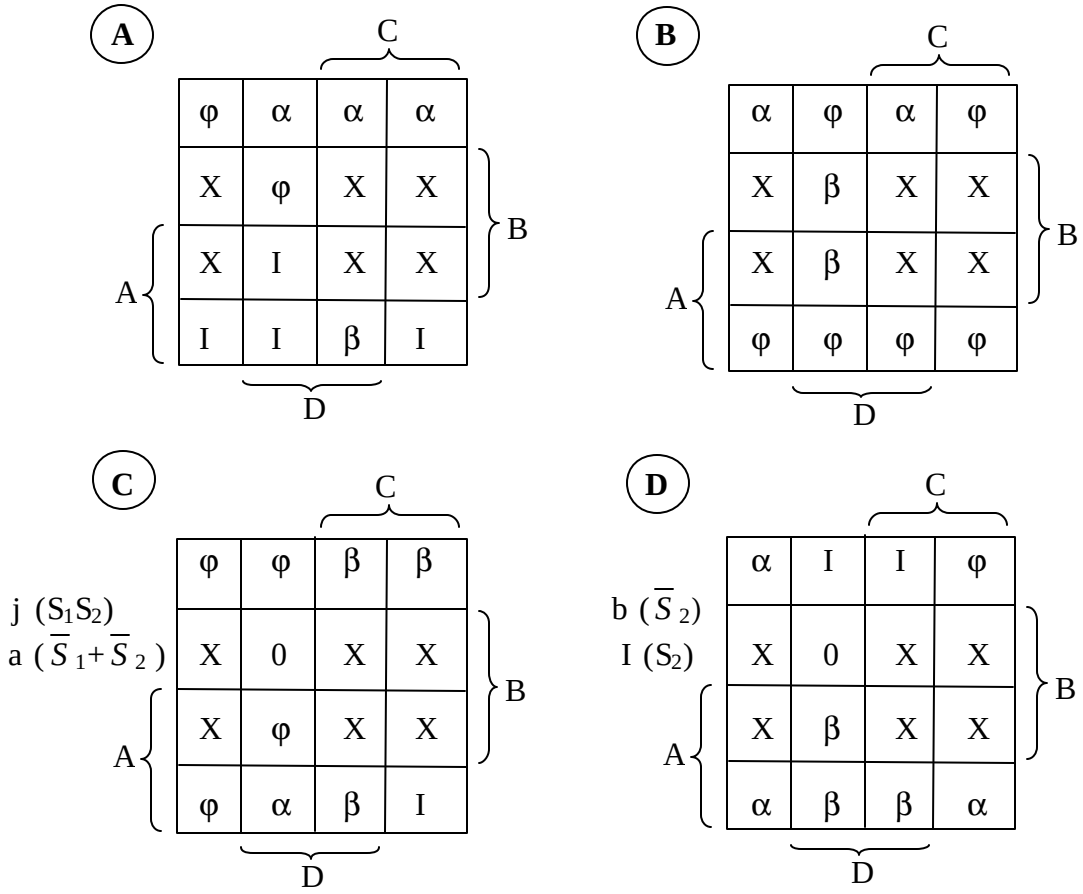


Figure3. Sequencer Karnaugh Maps

Step 4: *Decide on the type of flip-flops to be used in the design*.

Let assume that JK flip-flop is selected to be used in this design.

Step 5: *From the Karnaugh maps developed in Step 3, derive the optimum input equations for each flip-flop.*

As mentioned earlier, if at the branching state, a flip-flop goes through the same transition in all branches, as for flip-flops A and B in the example, then the branching control signals have no effect on this flip-flop and the Karnaugh map of this flip-flop should be treated as if there is no branching [1].
On the other hand, if at the branching state, a flip-flop can go through two different transitions, as for flip-flop C and D in our example, then the way to derive the input equation from the Karnaugh map for each input of the flip-flop depends on which case we have for that input. Depending on the relation between the input of the flip-flop under consideration and the two transitions which the flip-flop can go through at the branching state, we could have one of the following three cases:

Case 1: if one transition is *essential* and the other transition is *zero* transition for that input.
Case 2: if the two transitions are *don't care* transitions for that input.
Case 3: if one transition is *don't care* and the other transition is *zero* transition for that input.

The case for each input of each type of flip-flops for each of (α, φ) and (β, I) pair of transitions is given in Table 1.

	JK		RS		T	D
	J	K	R	S		
(α, φ)	Case 1	Case 2	Case 3	Case 1	Case 1	Case1
(β, I)	Case 2	Case 1	Case 1	Case 3	Case 1	Case1

Table 1.The Relation Between the Flip-flops Inputs and the Three Cases

The way to derive the input equation from the Karnaugh map for a flip-flop input in **case 1** is done in two steps. In the first step, the "0" is kept in the branching state location in the map and the essential transitions outside the branching state for that input are grouped and read as normal. In the second step, the "0" in the branching state location in the map is replaced by the essential transition which is written outside the map. Then this transition is grouped, in the largest possible group, with the don't care and other essential transitions. This group is read, ANDed with the control expression of the essential

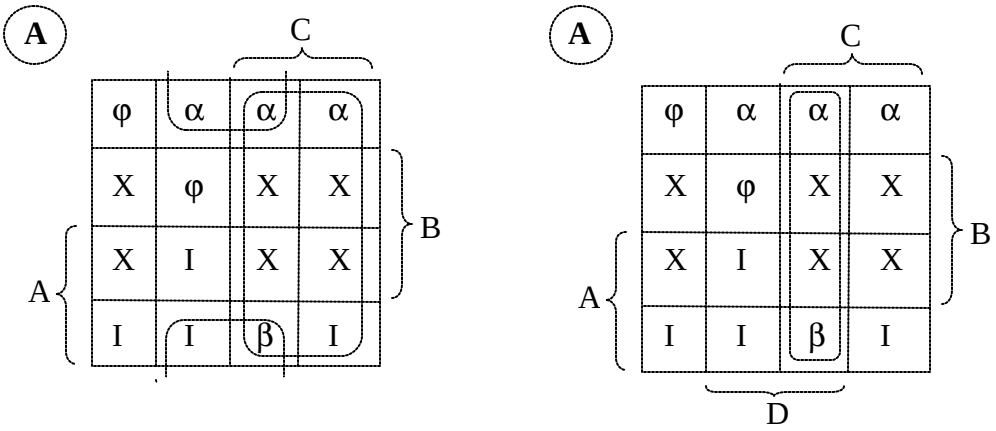
transition and then added to the logical expression generated in the first step. This final expression will represent the input equation to that input of the flip-flop.

The way to derive the input equation from the Karnaugh map for a flip-flop input in **case 2** is very simple. Just replace the "0" in the branching state location in the map by "x" and then derive the input equation from this map as if there is no branching.

The way to derive the input equation from the Karnaugh map for a flip-flop input in **case 3** is also very simple. Just keep the "0" in the branching state location in the map and then derive the input equation from this map as if there is no branching. In this case, the "0" is treated as a *ZERO* transition.

The last design step is applied to the above sequencer and the results are presented in the following.

1. For flip-flop A: Since flip-flop "A" goes through one transition at the branching state its Karnaugh map is treated as if there is no branching. The input equations to this flip-flop are given below.

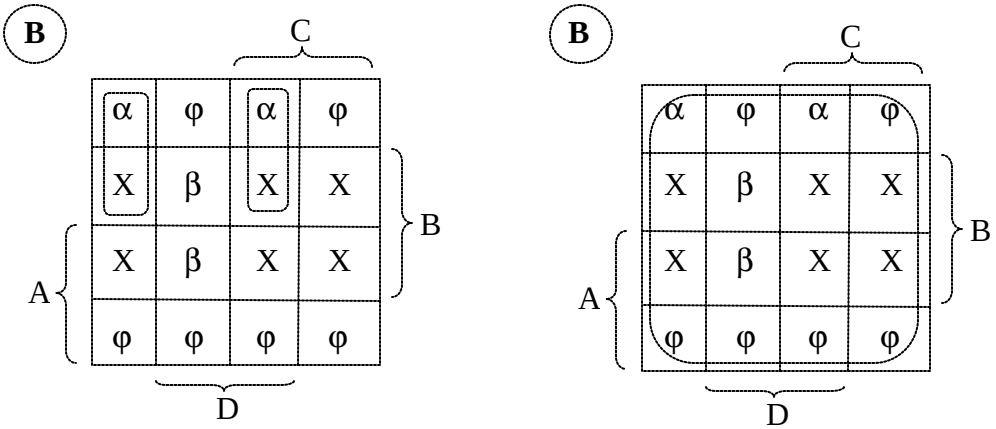


a. Groups for J_a input b. Groups for K_a input

Figure 4. Groups for Flip-flop A

$J_a = C + \overline{B}\ D$ $K_a = C\ D$

2. For flip-flop B: Since flip-flop "**B**" also goes through one transition at the branching state its Karnaugh map is treated as if there is no branching. The input equations to this flip-flop are given below.



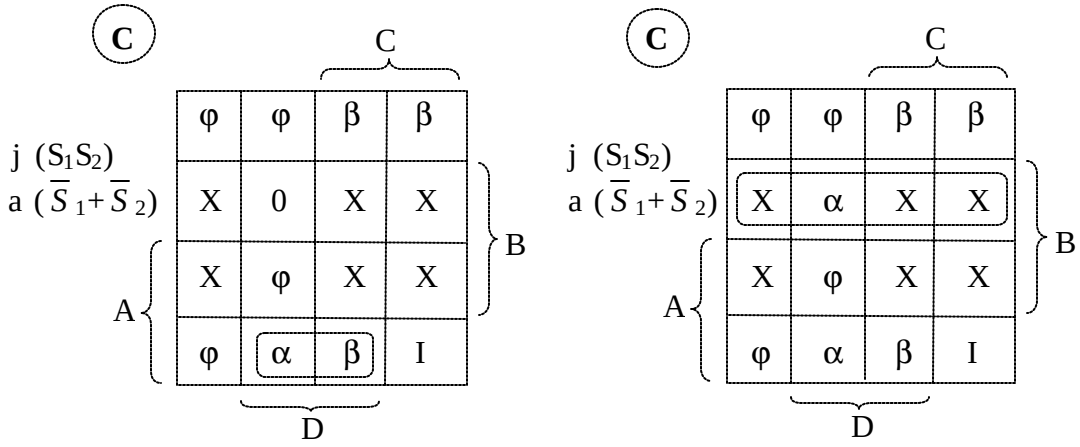
a. Groups for J_b input b. Groups for K_b input

Figure 5. Groups for Flip-flop B

$$J_b = \overline{A}\,\overline{C}\,\overline{D} + \overline{A}CD \qquad K_b = 1$$

3. For flip-flop C: Since flip-flop "**C**" has two transitions (α and φ) at the branching state, its two inputs are handled separately.

3a. J_c Input: For the J input and (α, φ) pair of transition it is Case 1, which is handled in two steps as discussed before. The input equation to this input is given below.



a. Groups for the first step of Case 1 for J_c input

b. Groups for the second step 2 of Case 1 for J_c input

Figure 6. Groups for J_c Input

$$J_c = A\,\overline{B}D + (\overline{S}_1 + \overline{S}_2)\overline{A}B$$

3b. K_c Input: For the K input and (α, φ) pair of transition it is Case 2, which is handled as discussed before. The input equation to this input is given below.

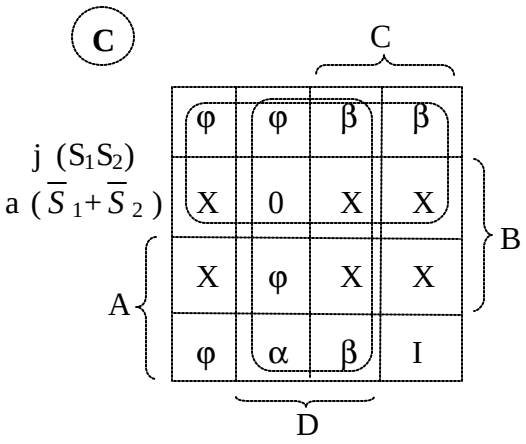


Figure 7. Groups for K_c Input

$$K_c = \overline{A} + D$$

4. For flip-flop D: Since flip-flop "**D**" has two transitions (β and I) at the branching state, its two inputs are handled separately.

4a. J_d Input: For the J input and (β, I) pair of transition it is Case 2, which is handled as discussed before. The input equation to this input is given below.

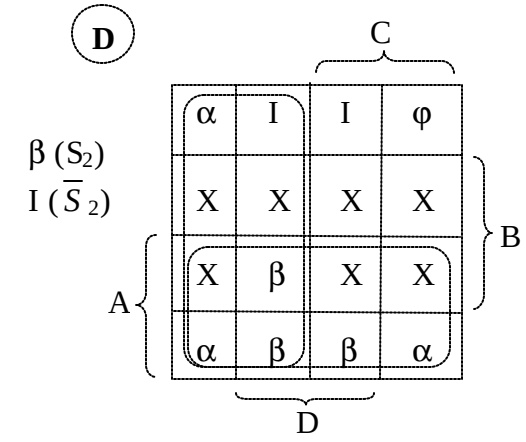
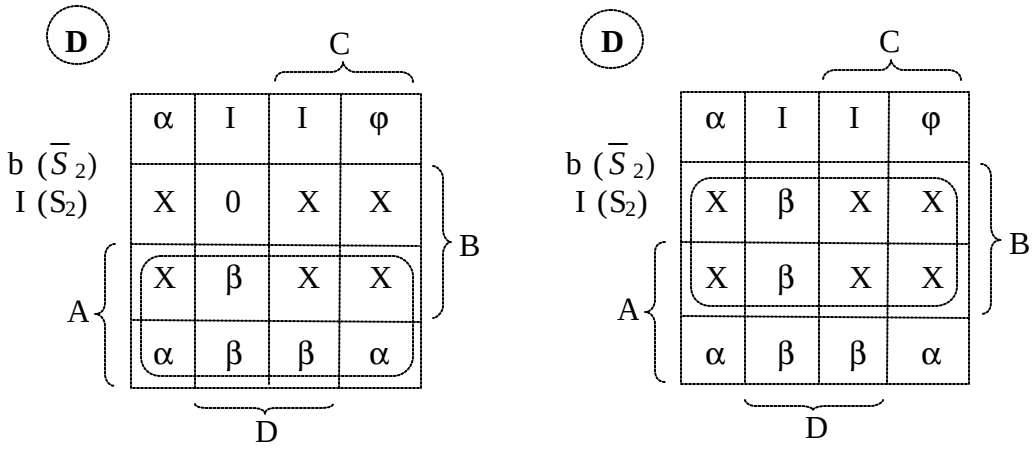


Figure 8. Groups for J_d Input

$$J_d = A + \overline{C}$$

4b. K_d Input: For the K input and (β, I) pair of transition it is Case 1, which is handled in two steps as discussed before. The input equation to this input is given below.



a. Groups for the first step of Case 1 for K_d input

b. Groups for the second step 2 of Case 1 for K_d input

Figure 9. Groups for K_d Input

$$K_d = A + \overline{S}_2 B$$

That conclude the design of this sequencer.

IV. Conclusion

In this paper, an efficient procedure to design multi-branch sequencers based on the use of El Naga's Transitions technique is presented. 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 time and easier work compared to that of the traditional method.

2. The design of a sequencer similar to the one presented in this paper using all types of flip-flops requires the construction of **six six-variable** maps for each flip-flop if the traditional method is used, while only **one four-variable** map is required if the Transition method is used. In this case, all input functions of all types of flip-flops can be derived out of this single map as explained in this paper.

3. While using the conventional method of designing sequential circuits does not provide any mean for the designer to check the correctness of his/her design until the implementation phase, 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[1]. This could save a lot of time and work.

4. Since designing a multi-branch sequencer using the conventional method of designing sequential circuits is a very long process that might take few hours, it is very hard for teachers to present this subject and very hard for the students to understand it based on the use of this method. For several years I have been easily teaching my students

designing multi-branch sequencers using the Transition method and it has been very well understood and perceived by them.

In conclusion, the innovated design technique that is presented in this paper makes the process of designing, teaching and understanding multi-branch sequencers much simpler.

Bibliography

1. El Naga, Nagi M. and Halima M. El Naga. "El Naga's Transitions Technique and its Advantages Compared to the Conventional Method of Designing Sequential Circuits," Proc. 2000 ASEE Annual Conference & Exposition, **June 18-21, 2000**, St. Louis, MO.
2. *"A Systematic Design Procedure for Asynchronous Counters Using El Naga's Transitions Technique,"* Proc. 2001 ASEE Annual Conference & Exposition,, June 24-27, 2001, Albuquerque, New Mexico.
3. El Naga, Nagi M. Lecture Notes, Electrical and Computer Engineering Department, California State University, Northridge, California, 1999.
4. John F. Wakerly, Digital Design Principles & Practices, Third Edition, Prentice Hall, 2000.

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, multiprocessor performance evaluation and Error detecting and correcting Systems. Dr. El Naga 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 B.S. degree in Electrical Engineering from Ain Shams University in 1977, M.S. in Electrical and Computer from California State University, Northridge in 1987 and her Ph.D. in Computer Engineering from the University of Southern California in 1999.