GENERALIZED FLIP-FLOP INPUT EQUATIONS BASED ON A FOUR-VALUED BOOLEAN ALGEBRA

Jerry H. Tucker
Aerospace Electronics System Division
NASA Mail Stop 488
Langley Research Center
Hampton, Virginia 23681
j.h.tucker@larc.nasa.gov

Moiez A. Tapia
Professor of Electrical and Computer Engineering
P.O. Box 248294
University of Miami
Coral Gables, Florida 33124-0640
mtapia@eng.miami.edu

***Abstract* - A procedure is developed for obtaining generalized flip-flop input equations, and a concise method is presented for representing these equations. The procedure is based on solving a four-valued characteristic equation of the flip-flop, and can encompass flip-flops that are too complex to approach intuitively. The technique is presented using Karnaugh maps, but could easily be implemented in software.**

INTRODUCTION

A procedure is developed for obtaining the generalized input equations of a flip-flop from the description of that flip-flop. The method relies on Boolean calculus concepts [1-4] and techniques for solving Boolean equations [5-9]. Specifically the method is based on solving a four-valued characteristic equation of a flip-flop for the two-valued input variables to the flip-flop. A representation is presented for concisely expressing the generalized input equations for any flip-flop. This representation enables finite-state machines to be synthesized using only a single set of Karnaugh maps, one for each state variable, for any combination of different types of flip-flops. The method can be applied not only to the simple commonly used flip-flops but is equally applicable to more complex flip-flops. The techniques presented can easily be extended to express the effect of clocks, and to precisely describe the combination of synchronous and asynchronous inputs.

FLIP-FLOP DESCRIPTION

A flip-flop, with output y and inputs, $\mathbf{x} = x_1,\ldots,x_i,\ldots,x_n$, can be described by the state diagram shown in Fig. 1. The various transitions of the flip-flop are specified by the functions $f_0,\ldots,f_3$. All of these are functions of the input variables $x_1,\ldots,x_i,\ldots,x_n$. For some flip-flops, there may be conditions under which the inputs, $x_1,\ldots,x_i,\ldots,x_n$, are constrained not to assume certain values. For these constrained conditions no transition is specified by f_0, f_1, f_2, or f_3. For example, the set-reset flip-flop requires that both the set and reset inputs are not "1" at the same time.

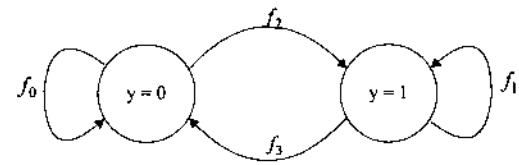

Fig. 1. The state diagram of a flip-flop.

Let δy be a four-valued Boolean variable capable of assuming a value of "0", "1", "Δ", or "∇". δ*y* will be used to specify the relationship between the present value, *y*, and the next value, *Y*, of the output of the flip-flop. This relationship is shown in Table 1.

Table 1. The definition of δy, where Y is the next value of y.

y	Y	δy
0	0	0
0	1	Δ
1	0	∇
1	1	1

OBTAINING THE MAPS

The flip-flop input equations could be obtained by expressing the flip-flop characteristic equation in terms of four-valued δy and solving this equation for each x_i, $0 \le i \le n$. Instead of expressing this four-valued characteristic equation algebraically, we will express the same information using a Karnaugh map for δy. This δ*y*-map is actually more useful for our purpose than writing an equation for δy. The method of obtaining the δy-map is given by Algorithm 1.

Algorithm 1. To obtain the δ*y*-map, from a flip-flop's state table or the state diagram, follow the steps given below:

1. Construct a Karnaugh map for δy with coordinates $x_1,\ldots,x_i,\ldots,x_n, y = (\mathbf{x}, y)$.
2. On the δ*y*-map obtained above, select each cell individually. For each selected cell, two cases are

possible. (a). If the next state, Y, is not specified for the (x, y) coordinates of the selected cell, enter a "-" in that cell. (b). If the next state, Y, is specified for the (x, y) coordinates of the selected cell, enter in that cell the appropriate value (0, Δ, ∇, or 1) of δy as specified in Table 1.

The "-" in step 2(a) above is used to specify conditions that cannot occur because of constraints on the inputs.

Once the δy-map of a flip-flop has been constructed, it can be used to obtain a map for any particular x_i. For presenting the procedure to obtain this x_i-map, it is convenient to let the vector $\mathbf{z}_i = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, represent a vector comprising all input variables other than some selected input variable, x_i. Using this notation the δy-*map* represents some function of x_i, y, and $\mathbf{z}_i$ as in (1).

$$\delta y = \delta y(x_i, y, \mathbf{z}_i) \tag{1}$$

The problem is to solve (1) for x_i. This results in a general equation for x_i, expressed as a function of δy and $\mathbf{z}_i$ as shown in (2).

$$x_i = x_i(\delta y, \mathbf{z}_i) \tag{2}$$

The procedure for doing this is an extension of the method introduced in [9]. It is convenient to define a $\mathbf{z}_i$-group to be a group of Karnaugh map cells all with the same $\mathbf{z}_i$ coordinate. We will use $\boldsymbol{a}$ to represent some particular value of $\mathbf{z}_i$, and α to represent some particular value of δy. Thus, α can be one of four values "0", "1", "Δ", or "∇". The concept of $\mathbf{z}_i$-group leads to two obvious lemmas:

Lemma 1: On the δy-map and the x_i-map, each $\mathbf{z}_i$-group contains four cells.

Proof: On the δy-map the coordinates of cells in a given $\mathbf{z}_i$-group differ in only x_i and y; thus, four cells are specified. On the x_i-map, the coordinates of cells in a given $\mathbf{z}_i$-group differ in only δy. Since δy can assume four values, four cells are specified.

Lemma 2: On the δy-map, a particular value of δy appears at most twice in any $\mathbf{z}_i$-group.

Proof: Any particular δy value determines the value of y; therefore, there are two cells, with coordinates differing only in x_i, that can contain the particular δy value.

With this background, the procedure for obtaining the x_i-map from the δy-*map* can be given.

Algorithm 2. To obtain the x_i-map from the δy-map.

1. Construct a map for x_i with coordinates (δy, $\mathbf{z}_i$).
2. On the δy-map and the x_i-map, form all possible $\mathbf{z}_i$-groups.
3. Considering each $\mathbf{z}_i$-group separately; use Table 2 to determine the value to enter in each cell of the x_i-map.

Table 2: The rules for obtaining the x_i-map from the δy-map. α is some particular value (0, 1, Δ, or ∇), of δy, and $\boldsymbol{a}$ is some particular value of $\mathbf{z}_i$.

Number of δy-map cells in ($\mathbf{z}_i = \boldsymbol{a}$)-group containing α.	Value to be entered in x_i-map cell with coordinate (α, $\boldsymbol{a}$)
0	-
1	x_i coordinate of δy-map cell containing α
2	d

Example 1.

Consider the Set-reset flip-flop with the state diagram shown in Fig. 2.

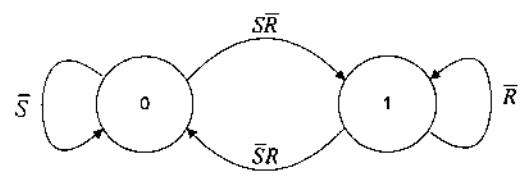

Fig. 2. The state diagram for the set-reset flip-flop.

The Karnaugh map for Y is shown in Fig. 3(a). The equivalent δy-map can be obtained by applying Algorithm 1 and is shown in Fig. 3(b).

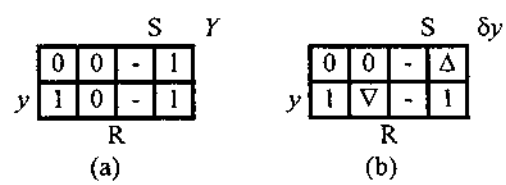

Fig. 3(a) The next value of y Karnaugh map for the set-reset flip-flop, (b) The δy Karnaugh map for the set-reset flip-flop. The "-" is used to indicate the constraint that SR = 0.

The S and R maps can be obtained from the δy-map by using Algorithm 2. As shown in Fig. 4(a) and Fig. 4(b), the S-map is obtained by forming the two R-groups on the δy-map. One R-group has a coordinate of R = 0, the other has a coordinate of R = 1. The R-group with coordinate of R = 1 determines the values to be entered in the four bottom cells of the S-map. The R-group with coordinate of R = 0 determines the values to be entered in the top four cells of the S-map. Considering the four top cells, the values are

obtained as follows: In the (R = 0)-group on the δy-map there is one "0" with an S coordinate of "0"; therefore, a "0" is entered in the top δy = 0 cell of the S-map. The (R = 0)-group of the δy-map contains a single "Δ" with an S coordinate of "1"; therefore, a "1" is entered in the top δy = Δ cell of the S-map. The (R = 0)-group of the δy-map has two cells containing "1"; therefore, "d" is entered in the top δy = 1 cell of the S-map. No cells in the (R = 0)-group of the δy-map contains "∇"; therefore "-" is entered in the top δy = ∇ cell of the S-map. A similar procedure is used to obtain the values to be entered in the (R = 1)-group of the S-map.

To solve for R, we form S-groups on the δy-map and again apply Algorithm 2. The result is shown in Fig. 4(c) and Fig. 4(d).

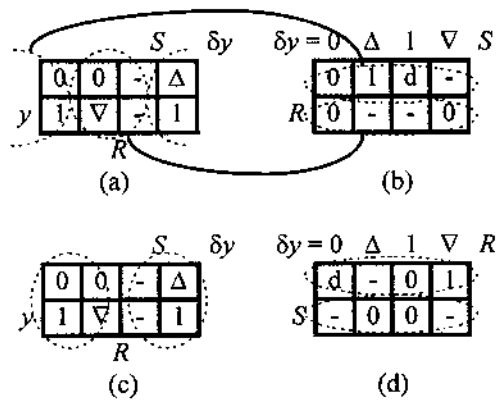

Fig. 4(a) The δy-map with R-groups circled. (b) The S-map with R-groups circled. (c) The δy-map with S-groups circled. (d) The R-map with S-groups circled.

OBTAINING THE INPUT EXPRESSIONS

Once the maps for S and R have been obtained, equations for S and R can easily be derived. These equations will be expressed using a variation of notation introduced in [7]. We will first describe the notation in its simplest form and generalize it later. The simpler form of the notation is sufficient whenever the x_i input equation is independent of z_i. When this is the case, x_i can be expressed in the form

$$x_i(\delta y) = \sum_{\delta y}(\alpha_1,\ldots,\alpha_r;d,\alpha_{r+1},\ldots,\alpha_s) \qquad (3)$$

Where α_j, $0 \le j \le s$, is some particular value (0, Δ, 1, or ∇) of δy. The generalized input expression given by (3) is obtained from the δy-map description of the flip-flop with inputs x_i and z_i. Once obtained, (3) can be used to realize the logic equations to implement a finite state machine. These implementation equations are obtained using the δy-maps of the state variables of the finite state machine. To do this, the expression given in (3) is interpreted as follows: The equation for x_i is obtained from a state variables δy-map by assigning to x_i, the value of "1" for the coordinates of all δy-map cells containing $\alpha_1,\ldots,\alpha_r$. The value of x_i is a don't-care, "d", for the coordinates of all cells containing $\alpha_{r+1},\ldots,\alpha_s$, or "d". For the coordinates of all other cells, the value of x_i is "0". The procedure for obtaining (3) is given below.

Algorithm 3. Obtaining the expression for x_i when x_i does not depend on the other input variables.

1. Form the four δy-groups on the x_i-map. These four groups are the (δy = 0)-group, the (δy = 1)-group, the (δy = Δ)-group and the (δy = ∇)-group.
2. Verify that each δy-group that contains a "1" does not contain a "0" or a "d", and that each δy-group that contains a "d" does not contain a "0" or a "1". Unless this condition is satisfied, x_i is not independent of the other input variables and can not be expressed in the simple form given by (3).
3. If a δy-group contains "1," use that δy value as one of the $\alpha_1,\ldots,\alpha_r$ symbols.
4. If a δy-group contains "d," use that δy value as one of the $\alpha_{r+1},\ldots,\alpha_s$ symbols.

To illustrate the use of this algorithm return to the set-reset flip-flop example. Applying Algorithm 3 to the S-map in Fig. 4 results in the generalized input expression for S:

$$S(\delta y) = \sum_{\delta y}(\Delta;d,1) \qquad (4)$$

The above equation is interpreted as follows: For any δy-map, S must be "1" for the coordinates of all δy-map cells containing a "Δ", and S is don't-care for the coordinates all δy-map cells containing a "d" or a "1".

Applying Algorithm 3 to the R-map in Fig. 5 results in the expression for R:

$$R(\delta y) = \sum_{\delta y}(\nabla;d,0) \qquad (5)$$

SUCCESSIVE ELIMINATION

In the above example, the expression for S was obtained from the δy-map, and then the expression for R was also obtained from the δy-map. As an alternative, the R-map could have been obtained from the S-map using the successive elimination method presented in [7]. This process of successive elimination will not always result in an identical input expression as produced by obtaining each input equation directly from the δy-map, but the expressions obtained in each case will always be equivalent. In general, using successive elimination will reduce the amount of computation and simplify the result for some inputs. This

general process of successive elimination is described below:

Algorithm 4. Given a map for x_j, $j \neq i$, with coordinates of x_i, $\mathbf{z}_i$, and δy, the map for x_i with coordinates of $\mathbf{z}_i$ and δy can be obtained as follows:

1. Construct a map for x_i with coordinates δy and $\mathbf{z}_i$.
2. On the x_j -map, form all possible $(\delta y, \mathbf{z}_i)$-groups.
3. For each $(\delta y, \mathbf{z}_i)$-group on the x_j map, enter the values, as shown in Table 3, in the corresponding cells of the x_i map.

Table 3: The rules for mapping a $(\delta y, \mathbf{z}_i)$-group on the x_j-map into the corresponding cell on the x_i-map. The "*" represents "0", "1", or "d".

Contents of the $(\delta y, \mathbf{z}_i)$-group on the x_j-map.		Value to be entered in the $(\delta y, \mathbf{z}_i)$ cell of the x_i-map.
$(x_i = 0)$ cell	$(x_i = 1)$ cell	
-	-	-
-	*	1
*	-	0
*	*	d

An intuitive understanding of the reason for the entries in Table 3 can be obtained as follows: When both the cells of some $(\delta y, \mathbf{z}_i)$-group on the x_j-map contain a "-", then, for that value of $(\delta y, \mathbf{z}_i)$, the variable x_i can assume neither the value "0" or "1". Thus, a "-" is entered in the $(\delta y, \mathbf{z}_i)$ cell of the x_i-map to indicate that the δy and $\mathbf{z}_i$ values of these coordinates are not permitted. If only one of the two cells of some $(\delta y, \mathbf{z}_i)$-group on the x_j-map contains a "-", then for that value of $(\delta y, \mathbf{z}_i)$, the only value x_i can assume is the x_i coordinate of the cell that does not contain the "-". If no cell of some $(\delta y, \mathbf{z}_i)$-group on the x_j-map contains a "-", then there is no restriction on the value of x_i; therefore, for that value of $(\delta y, \mathbf{z}_i)$, the variable x_i can assume the value of either "0" or "1".

The result of applying successive elimination to the set-reset flip-flop to obtain the R-map from the S-map is shown in Fig. 5. Applying Algorithm 3 to the R-map in Fig. 4(d) results in the same expression as given in (5). In this example the S-map has coordinates of only R and δy; thus, there are no $\mathbf{z}_i$ variables.

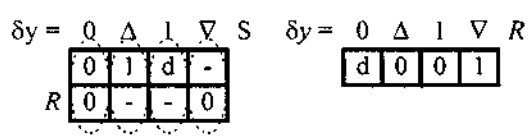

Fig. 5. Obtaining the R-map from the S-map by successive elimination. The δy-groups are circled on the S-map.

COMPLEX FLIP-FLOPS

The representation given by (3) is sufficient for most commonly used flip-flops; however, more complex flip-flops require an extension to the notation. This extension is necessary to take into account the effect of interdependence between various inputs. In general, the input equation for x_i can be expressed as

$$x_i(\delta y, \mathbf{z}) = \sum_{\delta y}\left(u_1, \ldots, u_r; d, u_{r+1}, \ldots, u_s\right) \tag{6}$$

where each u_j, $1 \leq j \leq s$, can be one of three forms. The there forms for u_j are: α_j, $\alpha_j \cdot a_j$, or a_j where a_j is some function of $\mathbf{z}_i$ or a subset of $\mathbf{z}_i$, and α_j is some particular value (0, Δ, 1, or ∇) of δy. A cell on the δy-map is said to be specified by u_j if any of the following conditions are satisfied: (1) u_j is of the form α_j, and the cell contains α_j. (2) u_j is of the form $\alpha_j \cdot a_j$, the cell has a $\mathbf{z}_i$ coordinate covered by a_j, and the cell contains α_j. (3) u_j is of the form a_j, and the cell has a $\mathbf{z}_i$ coordinate covered by of a_j. With this notation, the expression given by (6) is interpreted as follows: x_i is obtained from the a state variables δy-map by assigning to x_i, the value of "1" in all cells specified by $u_1, \ldots, u_r$. The value of x_i is a don't-care, "d" in all δy-map cells containg a "d", or specified by $u_{r+1}, \ldots, u_s$. In all other cells x_i is assigned a value of "0". It should be observed that with this notation, terms such as $0 \cdot x_1\bar{x}_2$ may appear inside the summation in (6). This term is used simply to express the condition when "$\delta y = 0$ and $x_1\bar{x}_2 = 1$". It does not imply that the term can be simplified to "0".

The expression given by (6) is obtained from the x_i-map as follows:

Algorithm 5. Obtaining the expression for x_i from the x_i-map with coordinates δy and $\mathbf{z}_i$.

1. Form the four δy-groups on the x_i map.
2. Each "1" on the x_i-map must be covered by some $u_1, \ldots, u_r$. If a $(\delta y = \alpha)$-group contains only the symbols "1" and "-", then the "1's" in that δy-group may be covered by $u_j = \alpha$. If a $(\mathbf{z}_i = a)$-group contains only the symbols "1" and "-", then the "1's" in that group may be covered by $u_j = a$. If a $(\delta y = \alpha)$-group contains a "1" that is not covered by the above, then that "1" must be covered by $u_j = \alpha \cdot a$, where a is the $\mathbf{z}_i$ coordinate of the cell containing the "1" or a subset of $\mathbf{z}_i$ coordinate of a group containing the "1" and other cells containing only the symbols "1" and "-".
3. Each "d" on the x_i-map must be covered by some $u_{r+1}, \ldots, u_s$. The rules for covering the "d" are the same as those given in step 2 with the "1" replaced by "d".

Example 2

Consider a DE flip-flop. This flip-flop is similar to a standard D flip-flop except that the D input is only enabled when the E input is "1". When E is "0", the flip-flop remains in its current state. The δy-map for the DE flip-flop is shown in Fig. 6.

		D		δy
	0	0	Δ	0
y	1	∇	1	1
		E		

δy = 0	Δ	1	∇	D
	d	-	d	-
E	0	1	1	0

Fig. 6. The δy-map and the D-map for the DE flip-flop.

The D-map in Fig. 6 is obtained from the δy-map in Fig. 6 using Algorithm 2. Note that the (δy = 1)-group on the D-map contains both "d" and "1", and the (δy = 0)-group on the D-map contains both "d" and "0". Thus, D cannot be independent of E. Using Algorithm 5, the expression below is obtained for D.

$$D(\delta y,E) = \sum_{\delta y}\left(\Delta, 1\cdot E; d, \overline{E}\right) \tag{7}$$

The above expression is interpreted as follows: D must have the value of "1" for the coordinates of those δy-map cells containing "Δ". The $1\cdot E$ term includes "·". This indicates that D must be assigned the value of "1" for the δy-map coordinates of those cells containing "1" provided E is assigned the value of "1" for those same δy-map coordinates. D is don't-care for the δy-map coordinates of those cells of the δy-map containing a "d" or for those coordinates where E is assigned a value of "0". The expression for E can be obtained from the δy-map or it can be obtained by elimination from the D-map. Fig. 7 shows the E-map as shown below:

δy =	0	Δ	1	∇	E
	d	1	d	1	

Fig. 7. The E-map obtained by successive elimination from the D-map shown in Fig. 6.

Since the E-map in Fig. 7 is not a function of other input variables, the expression for R can be obtained using Algorithm 2. This results in

$$E = \sum_{\delta y}(\Delta, \nabla; d, 1, 0) \tag{8}$$

CONCLUSION

Despite the widespread use of computer tools for synthesis of finite state machines, occasions still arise for manual synthesis. Flip-flop input expressions obtained using techniques presented here enable sequential circuits to be synthesized for virtually any flip-flop using only one δy-map for each state variable. The method presented for obtaining the input equations for flip-flops is considerably more powerful and general than previous techniques. The effect of clocks and a combination of both synchronous and asynchronous inputs can easily be handled using this method. The presented method may thus aid in the development of improved synthesis techniques.

In addition to providing an example of the benefit of applying Boolean equation techniques to a classical problem in switching theory, this paper is also of interest because of its utilization of a four-valued Boolean variable to describe conventional two valued logic.

REFERENCES

[1] J. H. Tucker, M. A. Tapia, and A. W. Bennett, "Boolean Differentiation and Integration Using Karnaugh Maps," Proceedings of IEEE Southeast-Conference, Williamsburg, Va., April 4-7, 1977.

[2] M. A. Tapia and J. H. Tucker, "Complete Solution of Boolean Equations," IEEE Transactions on Computers, Vol. C-29, July, 1980, pp. 662-665.

[3] J. H. Tucker, M. A. Tapia, and A. Wayne Bennett, "Boolean Integral Calculus for Digital Systems," IEEE Transactions on Computers, Vol. C-34, No. 1, January 1985, pp. 78-81.

[4] J. H. Tucker, M. A. Tapia, and A. W. Bennett, "Boolean Integral Calculus," Applied Mathematics and Computation Vol 26, 1988, pp. 201-236.

[5] S. Rudeanu, *Boolean Functions and Equations*, North Holland Publ. Co. & American Elsevier, Amsterdam - Londan - New York, 1974.

[6] F. M. Brown, *Boolean Reasoning The Logic of Boolean Equations*, Kluwer Academic Publishers, 1990.

[7] J. H. Tucker and M. A. Tapia, "Using Karnaugh Maps to Solve Boolean Equations by Successive Elimination," Proceedings of IEEE Southeastcon, pp. 589-591, Birmingham, Al. April 13-15, 1992.

[8] J. H. Tucker and M. A. Tapia, "Minimum Parameter Solution of Switching Equations," Proceedings of IEEE Southeastcon, pp. 180-184, Miami, Fl. April 11-13, 1994.

[9] J. H. Tucker and M. A. Tapia, "Solution of a Class of Boolean Equations," Proceedings of IEEE Southeastcon, March 26-29, 1995.