LATIN-AMERICAN JOURNAL OF COMPUTING (LAJC), Vol VIII, Issue 1, January 2021

Síntesis de Sistemas de Conmutación Mediante Permutación de Tablas de Código Gray (Método PGC)

Switching Systems Synthesis Method Using Permuted Gray Code Tables (PGC Method)

ARTICLE HISTORY

Received 06 August 2020
Accepted 02 November 2020

César Troya-Sherdek
Faculty of Applied Science
International University of Ecuador
Quito, Ecuador
cesartroyasherdek@gmail.com
https://orcid.org/0000-0002-4274-2649

Valentin Salgado-Fuentes
Department of Mechanical Engineering
Technical University of Denmark
Kgs. Lyngby, Denmark
vasafu@mek.dtu.dk

Jaime Molina
Department of Mechanical Science
Kachariy Higher Technical Institute
Quito, Ecuador
jaime.molina@itk.edu.ec

Gustavo Moreno
Department of Electronic Science
Kachariy Higher Technical Institute
Quito, Ecuador
gustavo.moreno@itk.edu.ec

C. Troya-Sherdek, V. Salgado-Fuentes, J. Molina and G. Moreno, "Switching Systems Synthesis Method Using Permuted Gray Code Tables (PGC Method)", Latin-American Journal of Computing (LAJC), vol. 8, no. 1, 2021.

Síntesis de Sistemas de Conmutación Mediante Permutación de Tablas de Código Gray (Método PGC)

Switching Systems Synthesis Method Using Permuted Gray Code Tables (PGC Method)

César Troya-Sherdek
Faculty of Applied Science
International University of Ecuador
Quito, Ecuador
cesartroyasherdek@gmail.com

Valentin Salgado-Fuentes
Department of Mechanical Engineering
Technical University of Denmark
Kgs. Lyngby, Denmark
vasafu@mek.dtu.dk

Jaime Molina
Department of Mechanical Science
Kachariy Higher Technical Institute
Quito, Ecuador
jaime.molina@itk.edu.ec

Gustavo Moreno
Department of Electronic Science
Kachariy Higher Technical Institute
Quito, Ecuador
gustavo.moreno@itk.edu.ec

Resumen— Encontrar la función más corta en los sistemas de conmutación es una necesidad para el desarrollo de sistemas automáticos eficientes. Actualmente, existen varias metodologías que tienen como objetivo solucionar esta necesidad con diferentes técnicas. Este artículo propone una nueva metodología para encontrar una fórmula proposicional que describa un problema de un sistema de conmutación utilizando varias tablas de verdad que se basan en una original, estas tablas se generan utilizando los principios y permutaciones del Código Gray. Como se mostrará, el código utilizado tiene una relación directa con los caminos hamiltonianos, donde cada permutación es una conexión diferente en un hipervolumen y cada nodo se representa como una combinación de bits. Para verificar y validar el método, se desarrolló un algoritmo utilizando el MATLAB y se comparó con las soluciones del software Boole-Deusto. Finalmente, se presentan ejemplos de ejecución, comparación de costos computacionales y propuestas de trabajos futuros.

Palabras Clave— *Caminos hamiltonianos, código Gray, funciones booleanas, hipercubo, problemas discretos, sistemas de conmutación*

Abstract— Finding the shortest function on switching systems is a necessity for the development of efficient automatic systems. Currently, several methodologies aim to solve this need with different techniques. This article proposes a new methodology to find a propositional formula that describes a switching system problem using several truth tables which are based on an original one; these tables are generated using Gray Code principles and permutations. As it will be shown, the used code has a direct relation to the Hamiltonian paths, where each permutation is a different connection in a hypervolume, and each node is represented as a bit combination. An algorithm was developed using MATLAB and compared with the solutions from the software Boole-Deusto to verify and validate the applicability and implementation of the method. Finally, examples of execution, computational cost comparison and future work proposals are presented.

Keywords— *Boolean functions, discrete problems, Gray Code, Hamiltonian Paths, hypercube, switching systems.*

I. INTRODUCTION

The solution of switching systems problems is of increasing importance in the development of modern technologies as well as in the implementation of automated control strategies. Thus, some authors like P. Roth [1], Quine [2] or Karnaugh [3] made much effort to improve the efficiency of these solutions. In propositional logic, a truth table defines a problem and a propositional formula (also known as truth function or Boolean function), can be obtained to describe any given truth table. Several methods can be used to obtain this formula, e.g. Veitch chart, Karnaugh map [4], minterms, maxterms [5] or Boolean algebra. However, as recognized by Quine [2], "The quest is to descry a technique to find the shortest truth-function formula" or by Veitch [6] "The problem is how to depict a Boolean function of "n" variables so the human eye can quickly see how to simplify the function". Veitch and Karnaugh used graphical methods, but higher-order problems present severe difficulties since the method involves human inspection.

This work proposes a new method to find a propositional formula that describes a switching system problem. Based on the original truth table, Gray code principles and permutations [7] are used to generate multiple truth tables. Gray codes are named after Frank Gray who in 1947 patented the idea of generating a binary codification that is used in applications that depends essentially on the bits looping. They are represented as a function G(i) where the consequential G(i+1) differ in exactly one bit [8], it is essential to note that the permuted tables that start with Gray code structure will maintain that property in all permutations. The advantage of using Gray codes to rearrange the truth tables, as will be explained in the method, lies on generating clusters that can be grouped and simplified using Boolean algebra theorems like identities and complements.

To verify and validate the proposed method, an algorithm using MATLAB 2014b is developed and tested with 2 to 7 bits logic tables. The computations are performed on a 64-bit architecture Intel XEON E3-1505M 2.80 GHz with 32 Gb RAM personal workstation capable of achieving a propositional formula that adequately solves the original switching system truth table. The solutions from the current method are compared with the solutions from the software Boole-Deusto [9].

II. METHOD

The Gray code tables present a single bit variation in each row, allowing to identify groups (2n members) of bits with a 'true' logic output that can be simplified. By permuting the columns of the Gray table, all possible clusters appear. The approach used has a direct relation with the Hamiltonian paths in hypercubes [10] where each permutation is a different path, and each node (vertex) is represented as a row (bit combination) of the truth table as can be seen in Fig. 1 [11]; transforming a multidimensional analysis into a unidimensional one. The right side of Fig. 1 shows a hypercube, whose vertices represent all possible combinations of the permutation. The vertices with a filled circle are the returned true outputs, and the vertices with the empty circle are the false outputs. Also, the arrows show the Hamiltonian path for the given Gray code truth table permutations that are represented in the left side of Fig. 1.

A	B	C	D	
0	0	0	0	1
1	0	0	0	1
1	0	0	1	1
0	0	0	1	1
0	1	0	1	1
1	1	0	1	1
1	1	0	0	1
0	1	0	0	1
0	1	1	0	0
1	1	1	0	0
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
1	0	1	1	1
1	0	1	0	0
0	0	1	0	0

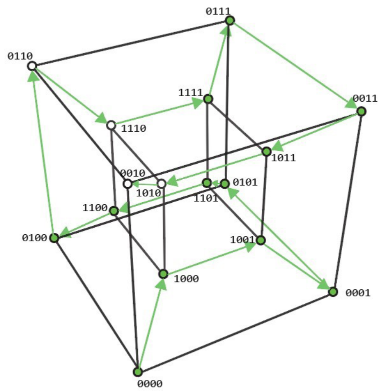

Fig. 1 Multidimensional representation of a Gray code table permutation.

The algorithm of the proposed method consists of four basic steps: preparation, generation, depuration and output. To illustrate the operation, a three-bit logic table that corresponds to the unidimensional output array XT = (0,1,0,1,1,0,0,1) will be used. For the first step, the original 'Gray table' must be generated along with all the possible permuted tables and each one of the eight-element logic output 'X' must be assigned to the corresponding input combination. The columns from the original 'Gray table' are named alphabetically using capital letters and used as independent arrays, for the example they will be: AT = (0,0,0,0,1,1,1,1); BT=(0,0,1,1,1,1,0,0); CT = (0,1,1,0,0,1,1,0). The number of permutations can be determined using (1) as described by Benavides [12], where the interchangeable values 'r' are the same than the number of bits 'n' so it can be expressed only as the factorial of 'n!'.

$$NTables = \frac{n!}{(n-r)!} \quad (1)$$

The permuted tables are assembled by concatenating the 'n' columns in the order given by the permutations and reassigning the labels of the columns to its original order; each row has also assigned its corresponding logic value of 'X' based on the input bits configuration. With the 'n!' tables filled, the generation stage begins, the objective is to mark the pairs of input sets combinations that give a 'true' logic output. Moreover, is important to keep track of the input sets that give a 'true' logic output

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2021

but never get in pairs with other combination through all the permutations, these sets are going to be known as 'elusive sets'. Fig. 2 shows the assembled permuted tables corresponding to the example; the dashed lines point out the selected input sets, the continuous vertical line indicates the corresponding true values and the continuous horizontal lines keep track of the 'elusive sets'.

1° PERMUTATION

A	B	C	X
0	0	0	0
1	0	0	1
1	1	0	0
0	1	0	0
0	1	1	1
1	1	1	1
1	0	1	0
0	0	1	1

2° PERMUTATION

A	B	C	X
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
0	1	1	1
1	1	1	1
1	1	0	0
0	1	0	0

3° PERMUTATION

A	B	C	X
0	0	0	0
0	1	0	0
1	1	0	0
1	0	0	1
1	0	1	0
1	1	1	1
0	1	1	1
0	0	1	1

4° PERMUTATION

A	B	C	X
0	0	0	0
0	0	1	1
1	0	1	0
1	0	0	1
1	1	0	0
1	1	1	1
0	1	1	1
0	1	0	0

5° PERMUTATION

A	B	C	X
0	0	0	0
0	0	1	1
0	1	1	1
0	1	0	0
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1

6° PERMUTATION

A	B	C	X
0	0	0	0
0	1	0	0
0	1	1	1
0	0	1	1
1	0	1	0
1	1	1	1
1	1	0	0
1	0	0	1

Fig. 2 Assembled permuted tables.

The pairs of input sets selected are reorganized in a two-column array called 'global combinations' while the 'elusive sets' occupy a namesake unidimensional array, as shown in Fig. 3a. In the depuration stage, the repeated sets must be deleted regardless of the order that the combinations were found (i.e., 011 111 is the equivalent of 111 011). The result of this operation is shown in Fig. 3b.

a)

GLOBAL COMBINATIONS

A	B	C	A	B	C
0	1	1	1	1	1
0	0	1	0	1	1
0	1	1	1	1	1
1	1	1	0	1	1
0	1	1	0	0	1
1	1	1	0	1	1
0	0	1	0	1	1
0	1	1	0	0	1

Elusive sets

A	B	C
1	0	0

b)

GLOBAL COMBINATIONS

A	B	C	A	B	C
0	1	1	1	1	1
0	0	1	0	1	1

Elusive sets

A	B	C
1	0	0

Fig. 3 a) Global combinations and Elusive sets. b) Depurated Global combinations and Elusive sets.

The output stage is required to translate the outcome of the depuration stage into the traditional Boolean algebra notation using the cleaned 'global combinations' array where the unchanged bits are subjected to a logical 'AND' whereas each row of the array to a logical 'OR'. On the 'elusive sets', the bits of each row are subjected to a logical 'AND' while the rows to a logical 'OR'. In both arrays, those bits with a 'false' logical state take the denied label (~), and bits with a 'true' logical state take only the label of the bit. The first combination changes bit 'A' while bits 'B' and 'C' remain the same, so the rule for this combination gives the output (B·C). In the second row, the bit 'B' changes, the bit 'A' remains with '0' and the bit 'C' remains with '1' so the logical output is (~ A · C). In the row from the 'elusive sets', the output is (A · ~ B · ~ C). The complete output is the logical OR of the previous rules: (~ A · C) + (B · C) + (A · ~ B ·~ C).

In some cases, there are associations involving two configurations sets that have already been marked; these are called 'ghost sets' and most commonly appear with four or larger numbers of input bits such as XT= (1,1,0,1,1,0,0,1,1,1,0,1,1,0,0, 1) which corresponds to a four-bit function. Fig. 4a shows the 'global combinations' array for this logical output after removing the repeated sets. To remove the 'ghosts sets' is necessary to find clusters of logical bits that repeat their groupings. In Fig. 4a, the binary combinations that are used more than once are marked with '1'. On the other hand, the first appearance of each combination and those that appear only once are marker with '0' (β column in Fig. 4a). To mark a combination of sets as selected at least one of the two sets must be marked with a zero. Continuous lines in Fig. 4a note the 'cleaned global combinations'. A clearer representation of this can be seen in Fig. 4b, where the global combinations are placed in the Karnaugh map format. The continuous lines indicate the 'cleaned global combinations' and the shaded sections represent the 'ghost sets'. It is important to note that in order to optimize the cleaning of 'ghosts sets' it is necessary to reorder the table of 'global combinations' by placing the 'elusive sets' (dashed lines) at the end of the table, these are represented by pairs of identical combinations in Fig 4a. The final output will be: (A·~C~D) + (A·C·D) + (~B·~C·D) + (~A·C·D) + (~A·~C·~D).

GLOBAL COMBINATIONS

A	B	C	D	β	A	B	C	D	β
1	1	0	0	0	1	0	0	0	0
1	1	1	1	0	1	0	1	1	0
0	0	0	1	0	1	0	0	1	0
0	0	1	1	0	0	1	1	1	0
0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	1	1
0	0	0	1	1	0	0	1	1	1
1	0	0	0	1	0	0	0	0	1
0	1	0	0	1	1	1	0	0	1
1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	0	0	0	1
0	0	1	1	1	1	0	1	1	1
0	0	0	0	1	0	0	0	1	1
1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1
1	1	0	0	1	1	1	0	0	1
0	1	0	0	1	0	1	0	0	1

(a)

CD \ AB	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	1	1	1
10	0	0	0	0

(b)

Fig. 4 a) Ghost sets depuration. b) Ghosts sets equivalent in Karnaugh maps.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2021

The importance of placing the 'elusive sets' at the end of the 'global combinations' table before executing the last depuration stage is to achieve a higher degree of simplification in the results, avoiding that the bits are selected individually generating functions equally equivalent but with more significant extension and complexity. To assess the effectiveness of the method in the treatment of the 'ghost sets', more cases with a different number of input bits were tested. Besides, unbalance cases with logic outputs containing more true outputs (1's) than false outputs (0's) and conversely, were used to make sure that proper simplification is accomplished. In these cases, sparse, random or uniform location of the true outputs (1's) was also considered.

Two examples corresponding to 3-bit combinations are presented to verify the entire method and its operation. The first example has an input XT= (0,0,0,1,1,0,0,1) and the second example is defined by the input XT= (1,0,0,1,1,0,0,1). Furthermore, the computational effort of the algorithm was analyzed to know how efficient the method could be in comparison with other fore-mentioned methods. However, even when the number of bits of the test cases is constant, the location of the true outputs (1's) changes the procedure performed in the depuration step and the running time needed. Therefore, to have a measurement that can be used as a benchmark, the input bit (1,0) was used in different computations but increasing with the number of bits; e.g. (1,0,1,0,1,0,1,0) for 3 bits, (1,0,1,0,1,0,1,0,1,0,1,0) for 4 bits, etc. The reason to use this logic output combination is due to the result in all the cases is the denied last variable (~ C and ~ D respectively). With only one variable as an answer, the eliminations of the repeated combinations in the depuration step increase accordingly with the number of bits and the running time. Finally, to increase the efficiency of the method, a more in-depth analysis was carried out to find a way to qualify the yielded permutations from the generation step based on better simplification perspectives.

III. RESULTS

A. Example number 1

From the vector XT= (0,0,0,1,1,0,0,1), six existing permutations are generated and presented in Fig. 5.

0	0	0	0
1	0	0	1
1	1	0	0
0	1	0	0
0	1	1	1
1	1	1	1
1	0	1	0
0	0	1	0

(a) 1st

0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	0
0	1	1	1
1	1	1	1
1	1	0	0
0	1	0	0

(b) 2nd

0	0	0	0
0	1	0	0
1	1	0	0
1	0	0	1
1	0	1	0
1	1	1	1
0	1	1	1
0	0	1	0

(c) 3rd

0	0	0	0
0	0	1	0
1	0	1	0
1	0	0	1
1	1	0	0
1	1	1	1
0	1	1	1
0	1	0	0

(d) 3rd

0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	0
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1

(e) 5th

0	0	0	0
0	1	0	0
0	1	1	1
0	0	1	0
1	0	1	0
1	1	1	1
1	1	0	0
1	0	0	1

(f) 6th

Fig. 5 Permutations - example 1.

From each of these permutations the local combinations are extracted and grouped in the table of 'global results' in Fig. 6, it is crucial to note the presence in the table of six identical rows with the combination [1 0 0 - 1 0 0] indicating a recursive 'elusive set'.

1	1	1	0	1	1
0	1	1	1	1	1
1	1	1	0	1	1
0	1	1	1	1	1
1	0	0	1	0	0
1	1	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	0	1	0	0
1	0	0	1	0	0
1	0	0	1	0	0

Fig. 6 Global Results - example 1.

Fig. 7a presents the results of performing the purification of repeated combinations on the global results, only four of the fourteen originals rows remained after the first filtration stage. Three of these results will remain in the form of 'elusive sets', however, at the time of performing the procedure described previously to eliminate 'ghost sets' the last two rows are discarded because they present redundant combinations that do not provide direct information to the solution, this final depuration result is presented in Fig. 7b.

1	1	1	0	1	1
1	0	0	1	0	0
1	1	1	1	1	1
0	1	1	0	1	1

(a) Global Results without repetitions

1	1	1	0	1	1
1	0	0	1	0	0

(b) Global Results without ghost set

Fig. 7 Depurations results - example 1.

Finally, the translation to traditional algebra notation is done by analyzing the bits that do not change in each row; the first combination changes only in the first bit while the second one remains constant in all of them, applying a logical OR (+) connection between the rows, the final Boolean function solution is (B·C) + (A·~B·~C).

B.Example number 2

From the vector XT= (1,0,0,1,1,0,0,1), six existing permutations are generated and presented in Fig. 8. It is interesting to note in these permutations that all true outputs (logic 1s) can be grouped in one or some of the tables, therefore, in the filtered results it will be observed that all 'elusive set' were eliminated.

0	0	0	0
1	0	0	1
1	1	0	0
0	1	0	0
0	1	1	1
1	1	1	1
1	0	1	0
0	0	1	0

(a) 1st

0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	0
0	1	1	1
1	1	1	1
1	1	0	0
0	1	0	0

(b) 2nd

0	0	0	0
0	1	0	0
1	1	0	0
1	0	0	1
1	0	1	0
1	1	1	1
0	1	1	1
0	0	1	0

(c) 3rd

0	0	0	0
0	0	1	0
1	0	1	0
1	0	0	1
1	1	0	0
1	1	1	1
0	1	1	1
0	1	0	0

(d) 3rd

0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	0
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1

(e) 5th

0	0	0	0
0	1	0	0
0	1	1	1
0	0	1	0
1	0	1	0
1	1	1	1
1	1	0	0
1	0	0	1

(f) 6th

Fig. 8 Permutations - example 2.

In Fig. 9 the 'local results' were grouped into the 'global results' array and sorted by placing the 'elusive sets' at the end (criterion presented in Fig 4), those being the last six rows of the table before the purification stage.

1	0	0	0	0	0
1	1	1	0	1	1
0	0	0	1	0	0
1	0	0	0	0	0
0	1	1	1	1	1
0	0	0	1	0	0
1	1	1	0	1	1
0	1	1	1	1	1
1	1	1	1	1	1
0	1	1	0	1	1
1	1	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	0	1	0	0

Fig. 9 Global Results - example 2.

The first filtering step presented in Fig. 10a eliminates the repeated combinations between rows moving from fourteen combinations to only five, of which three remain to be 'elusive sets', then in Fig. 10b the results of eliminating 'ghost sets' are presented, only two final combinations were conserved, and all remaining elusive sets were eliminated.

1	0	0	0	0	0
1	1	1	0	1	1
1	1	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0

(a) Global Results without repetitions

1	0	0	0	0	0
1	1	1	0	1	1

(b) Global Results without ghost sets

Fig. 10 Depurations results - example 2.

Restructuring the results into the Boolean algebra format yields to the function (~B·~C) + (B·C). Both presented examples were compared with the Boole-Deusto software, and the solutions achieved the same function, confirming the operation and effectiveness of the methodology.

C. Computational effort & qualification study

Table I compiles the average running time of the algorithm, measured for 2, 3, 4, 5, 6 and 7 bits. As can be seen, the running time increases exponentially with the increment of inputs bits 'n' due to the augment of tables (n!). This behaviour can be represented with (2) obtained through a regression method. With the equation, a simplified expression of the computational cost of the algorithm is achieved.

Bits	Time [s]
2	0.0828
3	0.1115
4	0.1804
5	1.1327
6	23.2123
7	1996.5976

Table I Computational time cost.

$$t = (9.8338294 \cdot 10^{-9}) \cdot e^{(3.7191 \cdot n)} \quad (2)$$

The qualification study yielded multiple truth tables sorted using Gray code (permutations), each one of them with unique 'true output' (logic 1's) clusters as shown with the continuous vertical lines in Fig. 11.

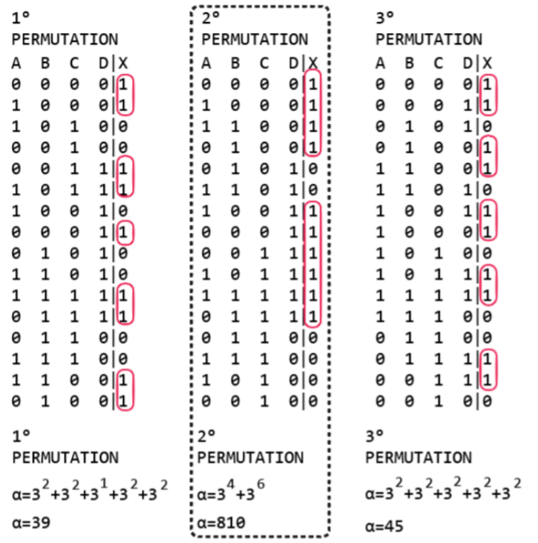

Fig. 11 Rating equation applied to 3 permutation tables.

From this clustering, (3) can be developed where: 'm' represents the number of groups found on each table and 'n' is the number of true values in each group, the result of such qualification is exemplified in Fig. 11

$$\alpha = \sum_{i=0}^{i=m} (3^{n_i}) \quad (3)$$

The second permutation has higher simplification potential with an alpha value of 810; it should be noted that, in Fig 11, only 3 of the 24 permutations were used.

IV. DISCUSSION

In the case of the method effectiveness, some cases were detected where the degree of simplification obtained was not fully accomplished compared to other methods. For instance, in the 4-bit function previously presented (Fig. 4a), the solution with the proposed method deduced from the 'global combination' table was: (A·~ C·~D) + (A·C·D) + (~B·~C·D) + (~A·C·D) + (~A·~C·~D) while the equivalent solution obtained by Boole-Deusto was: (~C·~D) + (C·D) + (~B·~C). It can be deduced that the method yields accurate results but not as effective as other methods, so an improvement in the depuration stage is needed.

Regarding the performance of the method, the time used for the programmed algorithm to solve the functions is longer than the one needed by Boole-Deusto. However, it is considered that significant improvements can be achieved and these results can be reduced considerably by improving steps taken on the algorithm, e.g. analyzing only the truth table with the best perspectives of simplification as explained below. Also, (2) and the results of the datasets were compared to well-known growth rate models and datasets presented by [13], this comparison confirms that the proposed model behaviour should be similar to an exponential model.

The qualification procedure could allow reducing the computational time by discarding unnecessary analyzes on inconvenient permutations. During the various testing stages, a characteristic behaviour has been observed in the way in which the 'local results' are organized in the 'global results' array. Therefore, it is considered that by applying (3) to qualify the permutations would be possible to establish a combination order that allows a more profound simplification by eliminating redundant tables that do not contribute new information to the resolution. Nonetheless, the improvement of the output stage would be covered in further studies.

V. CONCLUSION

This paper proposes the so-called PGC method to find a propositional formula of a switching system problem by using Gray code principles and Boolean algebra. The main advantage of the proposed method is that it does not require in-depth knowledge of Boolean algebra, and unlike graphical methods, the outcome does not require visual inspection. Moreover, the method is simple to implement and deploy in any programming tool since it does not require complex development techniques or advanced levels of analysis.

The proper operation of the method was demonstrated by comparing solutions obtained using the implementation described with manual methods and Boole-Deusto software. Although in some cases, the solution obtained did not represent the best possible result, an adequate degree of simplification was achieved, and all outputs obtained by the different methods are correspondingly equivalent. Furthermore, an equation that correlates the amount of time that the algorithm needs to solve a problem based on the number of logical inputs helps to estimate the computational time of the analyzed system before its deployment. Even though the computational time of the algorithm might be more significant than other methods, a possible step of the implementation has been identified as the future step of optimization for future developments of the PGC method.

Finally, the described method could also be extended to solve sequential logic problems, decision trees, route optimization, reduction of logic circuits or even for the academic purpose of using a flat interpretation of Hamiltonian hypercubes.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of M.Sc. Richard Andrade and M.Sc. Jose Beltran for their useful and valuable advice and mentorship. Their comments and suggestions during the performance of the project have helped the team to direct the focus of the analysis in the right track.

REFERENCES

[1] J. P. Roth, "Algebraic topological methods for the synthesis of switching systems. I.," Transactions of the American Mathematical Society 88.2, pp. 301-326, 1958.
[2] W.V. Quine "A way to simplify truth functions." The American mathematical monthly 62.9, pp. 627-631, 1955.
[3] M. Karnaugh, "The map method for synthesis of combinational logic circuits," Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 72.5, pp. 593-599, 1963.
[4] R.B Hurley, "Probability maps," IEEE Transactions on Reliability 12.3 pp. 39-44, 1963.
[5] Astola, Jaakko, and R. Stankovic. "Fundamentals of Switching Theory and Logic Design A Hands-on Approach," Springer, 2006.
[6] E.W. Veitch, "A chart method for simplifying truth functions," Proceedings of the ACM national meeting (Pittsburgh), 1952.
[7] Bitner, R James., Gideon Ehrlich, and E.M. Reingold. "Efficient generation of the binary reflected Gray code and its applications," Communications of the ACM 19.9, pp. 517-521, 1976.
[8] W. Press, et al. Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing. Cambridge university press. 1992.
[9] J. Zubia, J. García, Sanz Martínez, and S. Borja, "BOOLE-DEUSTO, la aplicación para sistemas digitales," 2001.
[10] J. Dybizbanski, and A. Szepietowski, "Hamiltonian paths in hypercubes with local traps," Information Sciences 375, pp. 258-270, 2007.
[11] K. Sankar, V. Jaya, M. Pandharipande, and P. S. Moharir. "Generalized gray codes," Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS IEEE. 2004.
[12] D. Benavides, "Diseño, implementación y evaluación de unidades didácticas de matemáticas en MAD 2," pp. 265. 2019.
[13] L. Egghe and I. Ravichandra Rao, "Classification of growth models based on growth rates and its applications," Scientometrics 25.1, pp. 5-46, 1992.

AUTHORS

César Troya-Sherdek

César David Troya Sherdek, Marketing / Sales operations / Business Intelligence Analyst in General Motors Ecuador, Associate professor at ITK Instituto Tecnológico Kachary for the electronics department, Cum Laude in Mechatronic Engineering from the International University of Ecuador, MBAc from ADEN University, has worked in research groups in the aeronautics area, aerospace, mathematics and computing, also lectured on data science and artificial intelligence.

Valentin Salgado-Fuentes

Valentin Salgado Fuentes- Born in Quito - Ecuador in 1991 and Graduated from the SEK International University of Ecuador in 2014 as a Mechanical Engineer with a specialization in Energy and Control processes. In 2016 he moved to Denmark to study a Master degree in Engineering Design and Applied Mechanics at the Technical University of Denmark (DTU). Since 2018, he is a PhD student at the Section of Thermal Energy at DTU developing advance numerical models of complex thermal systems.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2021

Jaime Molina

Jaime Vinicio Molina Osejos, Professor at SEK International University Auxiliary Investigator by the Senescyt, Master in Design, Production and Industrial Automation. Leader of the Laboratory of Metallography and Industrial, Coordinator of the Master's Degree in Mechanical Design, Manufacturing of Vehicle Autoparts (2016 - 2018). Coordinator of the Careers of Mechanical Engineering in: Design and Materials (2012 - 2014), Member of the Research Committee of the UISEK (2011) Professor since 2010 of in SEK International University. and Kachariy Higher Technological Institute

Gustavo Moreno

Gustavo Adolfo Moreno Jimenez is Professor at ITK Instituto Tecnologico Kachary director of Electronics area. He received his Master of Science in Technology Management from Marshall University (United States), his Master in Pedagogy and University Management from SEK University (Chile), and his Bachelor in Science at Electronic Engineering from ESPE University (Ecuador). He is a Senescyt certified Investigator, winner of "Ideas Bank" Senescyt Award in 2015, and winner of "Teaching Best Practices" SEK University Award in 2017.

C. Troya-Sherdek, V. Salgado-Fuentes, J. Molina and G. Moreno, "Switching Systems Synthesis Method Using Permuted Gray Code Tables (PGC Method)", Latin-American Journal of Computing (LAJC), vol. 8, no. 1, 2021.