Journal of Advances in Mathematics and Computer Science

30(1): 1-22, 2019; Article no.JAMCS.45009

ISSN: 2456-9968

(Past name: British Journal of Mathematics & Computer Science, Past ISSN: 2231-0851)

Derivation of a Scalable Solution for the Problem of Factoring an n-bit Integer

Ali Muhammad Rushdi[1*], Sultan Sameer Zagzoog[1] and Ahmed Said Balamesh[1]

[1]Department of Electrical and Computer Engineering, King Abdulaziz University, P. O. Box 80204, Jeddah 21589, Saudi Arabia.

Authors' contributions

This work was carried out in collaboration among the three authors. Author AMR designed the study, performed the analysis, solved the examples and wrote the preliminary manuscript. Author SSZ managed the literature search and drew the figures. Author ASB contributed to the analysis and solutions of the examples and provided useful insight about the interrelationships among problems of different sizes and various solution approaches. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JAMCS/2019/45009

Editor(s):

(1) Dr. Kai-Long Hsiao, Associate Professor, Taiwan Shoufu University, Taiwan.

(2) Dr. Feyzi Basar, Professor, Department of Mathematics, Fatih University, Turkey.

Reviewers:

(1) Iroju Olaronke, Adeyemi College of Education, Nigeria.

(2) Iouliia Skliarova, University of Aveiro, Portugal.

(3) Ibrahim Senturk, Ege University, Turkey.

Complete Peer review History: http://www.sciencedomain.org/review-history/27962

Original Research Article

Received: 15 October 2018
Accepted: 12 December 2018
Published: 24 December 2018

Abstract

The problem of integer factorization is ubiquitous in scientific and engineering applications including the challenging task of cryptanalysis. This problem is intractable but might admit real-time hardware solutions for small bit sizes. This paper suggests manual and automated scalable solutions for integer factorization based on equation solving over big Boolean algebras. The manual solution is illustrated over a form of 8-variable Karnaugh maps that is highly regular and modular. This solution covers the problem of 6 bits, which includes the problems of 5, 4, and 3 bits as special cases. Moreover, the automated solution is implemented, and subsequently its results are presented and discussed briefly. These results show the notorious evolution of the temporal and spatial complexities as the number of input bits increases. Based on the automated solution, the largest possible hardware circuit obtained via the automated solution is to be constructed, verified and tested. Such a hardware implementation (e.g., FPGA implementation) could serve as a ready real-time look-up solution not only of the pertinent problem but also of all smaller problems.

**Corresponding author: E-mail: arushdi@kau.edu.sa, arushdi@yahoo.com, arushdi@ieee.org, alirushdi@gmail.com;*

Keywords: Manual and automated scalable solutions; integer factorization; Boolean-equation solving; modular Karnaugh map; algorithmic implementation.

1 Introduction

The problem of integer factorization is ubiquitous in scientific applications, and it is particularly prominent in the cryptanalysis of the RSA cryptosystem [1-4]. This problem is intractable, and none of the many sophisticated algorithms for solving it has an under-exponential temporal complexity. Currently, there are many attempts to handle this problem in real time *via* hardware solutions. Some of these attempts are based on the extension of propositional logic to higher-order logic such as first-order predicate logic. These attempts involve Boolean functional synthesis and the utilization of Skolem functions [5-9]. Other attempts require the enlargement of two-valued Boolean algebras to a 'big' Boolean algebra [10-12], an approach to be pursued further herein. Other notable approaches for the hardware solution of integer factorization are also available, and continuous innovations in such approaches are being offered with no end in sight [13-28].

Our present paper aims to obtain a hardware solution for integer factorization that is based on solving Boolean equations over 'big' Boolean algebras [29-52]. However, our solution herein has three advantages over earlier solutions, namely: (a) it is general and scalable, (b) it is automated, and (c) it is realizable via current hardware technologies such as that of FPGA. Of course, scalability is not absolute. It will reach a limit due the bottleneck imposed by time and memory limitations. The solution is pictorially insightful thanks to efficient utilization of modern versions of the Karnaugh map [53-62].

The organization of the rest of this paper is as follows. Section 2 modifies our earlier hardware solution for integer factorization using Boolean-equation solving [10-12]. The modifications introduced make this solution scalable and consequently readily amenable for clear and straightforward algorithmic formulation. Section 3 solves the (6, 5, 3) factorization problem, in which a 6-bit integer **X** is factored as a product of a 5-bit integer **Y** and a 3-bit integer $\mathbf{Z}$. We solve this problem with the aid of a special form of the 8-variable Karnaugh map (taken from [62, 63]) that has better regularity and modularity than other forms of the 8-variable map such as the ones used in [11, 64-68]. Section 4 outlines our algorithmic implementation of integer factorization, with a code listing in Matlab given in appendix A. Section 4 also reports typical results obtained by running the given code. Section 5 concludes the paper.

2 General Scalable Formulation

We have earlier discussed the problem of factoring an integer **X** into a product $\mathbf{Y} * \mathbf{Z}$ of the two integers **Y** and **Z** subject to the constraints ($\mathbf{Y} \geq \mathbf{Z}$) and ($\mathbf{Z} \geq \mathbf{1}$) [10, 11]. For an even bit size of **X,** say $2n$, the bit size of **Y** and **Z** are $(2n-1)$ and n, respectively. However, when we multiply **Y** and **Z** of such sizes, we produce an **X** of bit size $(2n-1)+n=3n-1$. In our earlier work [10, 11], we have forbidden bit sizes of **X** exceeding $2n$, a feature that did not allow full scalability for our solution. In our present paper, we will *temporarily* allow bit sizes of **X** to be $(3n-1)$, thereby securing full scalability for our solution, in the sense that our solution for the $2n$ problem includes solutions for all smaller valid problem (down to a bit size of 3 for **X**) as special cases. To solve our current problem, in which the triad (**X**, **Y**, **Z**) has bit sizes $(2n, 2n-1, n)$, we need a multiplication table for the $(2n-1)-bit$ **Y** with the $n-bit$ **Z**, and allow entries of the table $T(\mathbf{Y}, \mathbf{Z}) = \mathbf{Y} * \mathbf{Z}$ to be each of $(3n-1)$ bits. For convenience, we arrange the input domain of this table to render it a Karnaugh-map layout, *i.e.,* to employ a reflected binary coding (grey coding) for each of its two dimensions.

The multiplication table constitutes the initial specification for our problem, namely

$$g_0(\mathbf{Y}, \mathbf{Z}) = 1 \qquad (1)$$

Where

$$g_0 = \wedge_{i=0}^{(3n-2)} (\mathrm{X}_i \,\mathcal{O}\, \mathrm{T}_i(\mathbf{Y},\mathbf{Z})) = \wedge_{i=0}^{(3n-2)} \mathrm{X}_i^{\mathrm{T}_i(\mathbf{Y},\mathbf{Z})} \tag{2}$$

is an ANDing over all the $(3n-1)$ bits i of $\mathbf{X}$. The XNOR function $\mathrm{X}_i \,\mathcal{O}\, \mathrm{T}_i(\mathbf{Y},\mathbf{Z})$ equals the complemented literal $\overline{\mathrm{X}}_i$ when $\mathrm{T}_i(\mathbf{Y},\mathbf{Z}) = 0$ and equals the un-complemented literal X_i when $\mathrm{T}_i(\mathbf{Y},\mathbf{Z}) = 1$. The initial function $g_0(\mathbf{Y},\mathbf{Z})$ is now replaced by the final-specification function (equated to 1)

$$g(\mathbf{Y},\mathbf{Z}) = g_0(\mathbf{Y},\mathbf{Z})\, \mathrm{I}(\mathbf{Z} > 1)\, \mathrm{I}(\mathbf{Y} \geq \mathbf{Z}) \tag{3}$$

where the symbol I(event) denotes the Boolean indicator for that event, namely

$$\mathrm{I(event)} = \begin{cases} 1 & \textit{if the event occurs} \\ 0 & \textit{if the event does not occur} \end{cases} \tag{4}$$

The two functions $g_0(\mathbf{Y},\mathbf{Z})$ and $g(\mathbf{Y},\mathbf{Z})$ are defined as $g_0\colon \mathrm{B}^{3n-1} \to \mathrm{B}$, and $g\colon \mathrm{B}^{3n-1} \to \mathrm{B}$ where B is the ‘big’ Boolean algebra $\mathrm{B} = \mathrm{FB}(\mathbf{X})$, *i.e.,* it is the free Boolean algebra with $(3n-1)$ generators $\mathrm{X}_{(3n-2)}, \mathrm{X}_{(3n-3)} \dots, \mathrm{X}_1, and\ \mathrm{X}_0$. This Boolean algebra is of $\mathrm{K} = 2^{(3n-1)}$ atoms and 2^{K} elements. The Boolean equation (1) is now solved by constructing the auxiliary function $\mathrm{G}(\mathbf{Y},\mathbf{Z},\mathbf{p})$ according to the rules given or demonstrated in [10, 11, 40, 50-52]. It is straightforward to use $\mathrm{G}(\mathbf{Y},\mathbf{Z},\mathbf{p})$ to deduce the value of $\mathrm{Y}_i(\mathbf{X},\mathbf{p}), 0 \leq i \leq (2n-2), and\ \mathrm{Z}_i, 0 \leq i \leq (n-1)$ subject to a consistency condition (to be derived also from $\mathrm{G}(\mathbf{Y},\mathbf{Z},\mathbf{p})$). We do not need to develop $\mathrm{G}(\mathbf{Y},\mathbf{Z},\mathbf{p})$ fully, as we do not need to find the parameters associated with atoms that have non-zero $\mathrm{X}_i, 2n \leq i < (3n-1)$. This means that we work with the first 2^{2n} atoms out of the $2^{(3n-1)}$ atoms. Our solutions will not involve $\mathrm{X}_i, 2n \leq i < (3n-1)$ (which are set identically zero) and will involve only the first $2n$ bits $(\mathrm{X}_i, 0 \leq i < 2n)$.

The aforementioned solution for the $(2n, 2n-1, n)$ problem includes all valid smaller problems down to the (3, 2, 2) problem as special cases. The next problem is the $(2n-1, 2n-2, n)$, which involves **X** of an odd rather than even bit size and its Karnaugh maps are of sizes that are *one half* those of the preceding $(2n, 2n-1, n)$ problem. For this latter problem, **X** has an initial odd bit size of $(3n-2)$, but only $(2n-1)$ bits are retained at the end. The problem next to this problem is the $(2n-2, 2n-3, n-1)$ problem. This problem involves **X** of an even bit size, and its Karnaugh maps are of sizes that are one *quarter* those of the preceding problem. For this problem **X** has a bit size of $(3n-4)$ at the outset, which reduces to $(2n-2)$ at the end.

3 Solution of the (6, 5, 3) Problem

Fig. 1 shows a decimal-entered multiplication table for the two integers $\mathbf{Y} = (\mathrm{Y}_4, \mathrm{Y}_3, \mathrm{Y}_2, \mathrm{Y}_1, \mathrm{Y}_0)$ and $\mathbf{Z} = (\mathrm{Z}_2, \mathrm{Z}_1, \mathrm{Z}_0)$. The table is cast in an 8-variable Karnaugh map layout. For convenience, we depict every map column by two decimal values of $\mathbf{Y} = (2^4\mathrm{Y}_4 + 2^3\mathrm{Y}_3 + 2^2\mathrm{Y}_2 + 2^1\mathrm{Y}_1 + 2^0\mathrm{Y}_0)$ in its cells $(0 \leq \mathbf{Y} \leq 31)$. The first value is $(2^4\mathrm{Y}_4 + 2^3\mathrm{Y}_3 + 2^2\mathrm{Y}_2 + 2^1\mathrm{Y}_1)$. It is determined by the four horizontal variables of the map, and is valid for cells in which $\mathrm{Y}_0 = 0$ (cells outside the Y_0 domain). The second value is equal to the previous value augmented by 1 and is valid for cells in which $\mathrm{Y}_0 = 1$ (cells inside the Y_0 domain, highlighted by green shading lines). We also depict every two consecutive map rows by a single value of $\mathbf{Z}(0 \leq \mathbf{Z} \leq 7)$, where $\mathbf{Z} = 2^2\mathrm{Z}_2 + 2^1\mathrm{Z}_1 + 2^0\mathrm{Z}_0$. Note that the *left half* of the map in Fig. 1 describes the next smaller problem, the (5, 4, 3) problem. The *top left quarter* of this half depicts the next smaller problem, the (4, 3, 2) problem. Finally, the smallest valid problem, the (3, 2, 2) problem, is represented by, again, the left half of the previous problem. We use bold boundaries and various shadings to distinguish the smaller maps used for the smaller problems in Fig. 1.

Table 1. Orthonormal tags used with composite integers less than 64 with multiple factorizations. A thick line denotes the lower boundary for the n-bit problem, where *n* = 3, 4, 5, and 6, respectively

Integer	Corresponding atom (only atoms with $\bar{X}_7\bar{X}_6$ are retained)	Multiplicity of non-trivial factorizations	Set of orthonormal tags
12	$\bar{X}_7\bar{X}_6\bar{X}_5\bar{X}_4X_3X_2\bar{X}_1\bar{X}_0$	2	$\{p_1, \bar{p}_1\}$
16	$\bar{X}_7\bar{X}_6\bar{X}_5X_4\bar{X}_3\bar{X}_2\bar{X}_1\bar{X}_0$	2	$\{p_2, \bar{p}_2\}$
18	$\bar{X}_7\bar{X}_6\bar{X}_5X_4\bar{X}_3\bar{X}_2X_1\bar{X}_0$	2	$\{p_3, \bar{p}_3\}$
20	$\bar{X}_7\bar{X}_6\bar{X}_5X_4\bar{X}_3X_2\bar{X}_1\bar{X}_0$	2	$\{p_4, \bar{p}_4\}$
24	$\bar{X}_7\bar{X}_6\bar{X}_5X_4X_3\bar{X}_2\bar{X}_1\bar{X}_0$	3	$\{p_5p_6, p_5\bar{p}_6, \bar{p}_5\}$
28	$\bar{X}_7\bar{X}_6\bar{X}_5X_4X_3X_2\bar{X}_1\bar{X}_0$	2	$\{p_7, \bar{p}_7\}$
30	$\bar{X}_7\bar{X}_6\bar{X}_5X_4X_3X_2X_1\bar{X}_0$	3	$\{p_8p_9, p_8\bar{p}_9, \bar{p}_8\}$
32	$\bar{X}_7\bar{X}_6X_5\bar{X}_4\bar{X}_3\bar{X}_2\bar{X}_1\bar{X}_0$	2	$\{p_{10}, \bar{p}_{10}\}$
36	$\bar{X}_7\bar{X}_6X_5\bar{X}_4\bar{X}_3X_2\bar{X}_1\bar{X}_0$	4	$\{p_{11}p_{12}, p_{11}\bar{p}_{12}, \bar{p}_{11}p_{12}, \bar{p}_{11}\bar{p}_{12}\}$
40	$\bar{X}_7\bar{X}_6X_5\bar{X}_4X_3\bar{X}_2\bar{X}_1\bar{X}_0$	3	$\{p_{13}p_{14}, p_{13}\bar{p}_{14}, \bar{p}_{13}\}$
42	$\bar{X}_7\bar{X}_6X_5\bar{X}_4X_3\bar{X}_2X_1\bar{X}_0$	3	$\{p_{15}p_{16}, p_{15}\bar{p}_{16}, \bar{p}_{15}\}$
44	$\bar{X}_7\bar{X}_6X_5\bar{X}_4X_3X_2\bar{X}_1\bar{X}_0$	2	$\{p_{17}, \bar{p}_{17}\}$
45	$\bar{X}_7\bar{X}_6X_5\bar{X}_4X_3X_2\bar{X}_1X_0$	2	$\{p_{18}, \bar{p}_{18}\}$
48	$\bar{X}_7\bar{X}_6X_5X_4\bar{X}_3\bar{X}_2\bar{X}_1\bar{X}_0$	4	$\{p_{19}p_{20}, p_{19}\bar{p}_{20}, \bar{p}_{19}p_{20}, \bar{p}_{19}\bar{p}_{20}\}$
50	$\bar{X}_7\bar{X}_6X_5X_4\bar{X}_3\bar{X}_2X_1\bar{X}_0$	2	$\{p_{21}, \bar{p}_{21}\}$
52	$\bar{X}_7\bar{X}_6X_5X_4\bar{X}_3X_2\bar{X}_1\bar{X}_0$	2	$\{p_{22}, \bar{p}_{22}\}$
54	$\bar{X}_7\bar{X}_6X_5X_4\bar{X}_3X_2X_1\bar{X}_0$	3	$\{p_{23}p_{24}, p_{23}\bar{p}_{24}, \bar{p}_{23}\}$
56	$\bar{X}_7\bar{X}_6X_5X_4X_3\bar{X}_2\bar{X}_1\bar{X}_0$	3	$\{p_{25}p_{26}, p_{25}\bar{p}_{26}, \bar{p}_{25}\}$
60	$\bar{X}_7\bar{X}_6X_5X_4X_3X_2\bar{X}_1\bar{X}_0$	5	$\{p_{27}p_{28}p_{29}, p_{27}p_{28}\bar{p}_{29}, p_{27}\bar{p}_{28}p_{29}, p_{27}\bar{p}_{28}\bar{p}_{29}, \bar{p}_{27}\}$
63	$\bar{X}_7\bar{X}_6X_5X_4X_3X_2X_1X_0$	2	$\{p_{30}, \bar{p}_{30}\}$

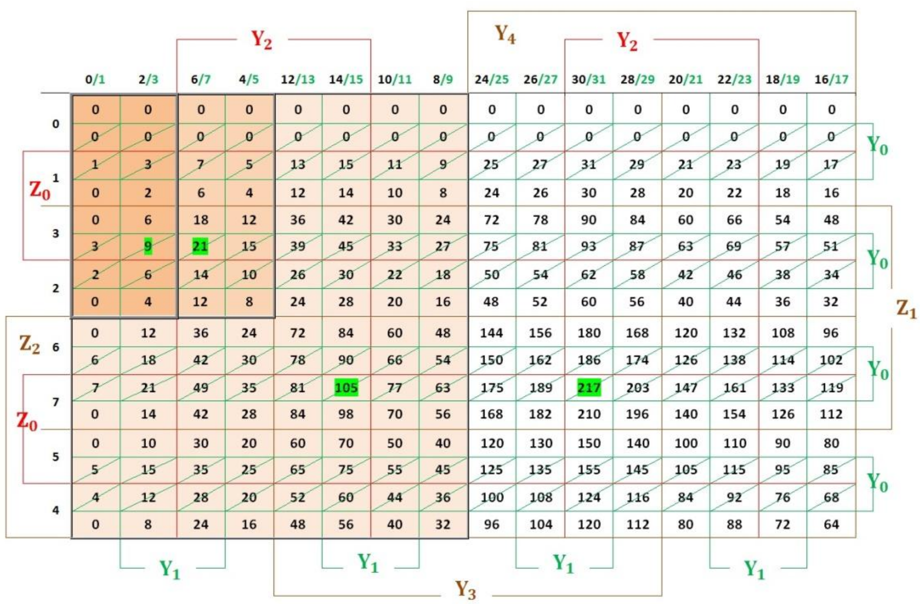
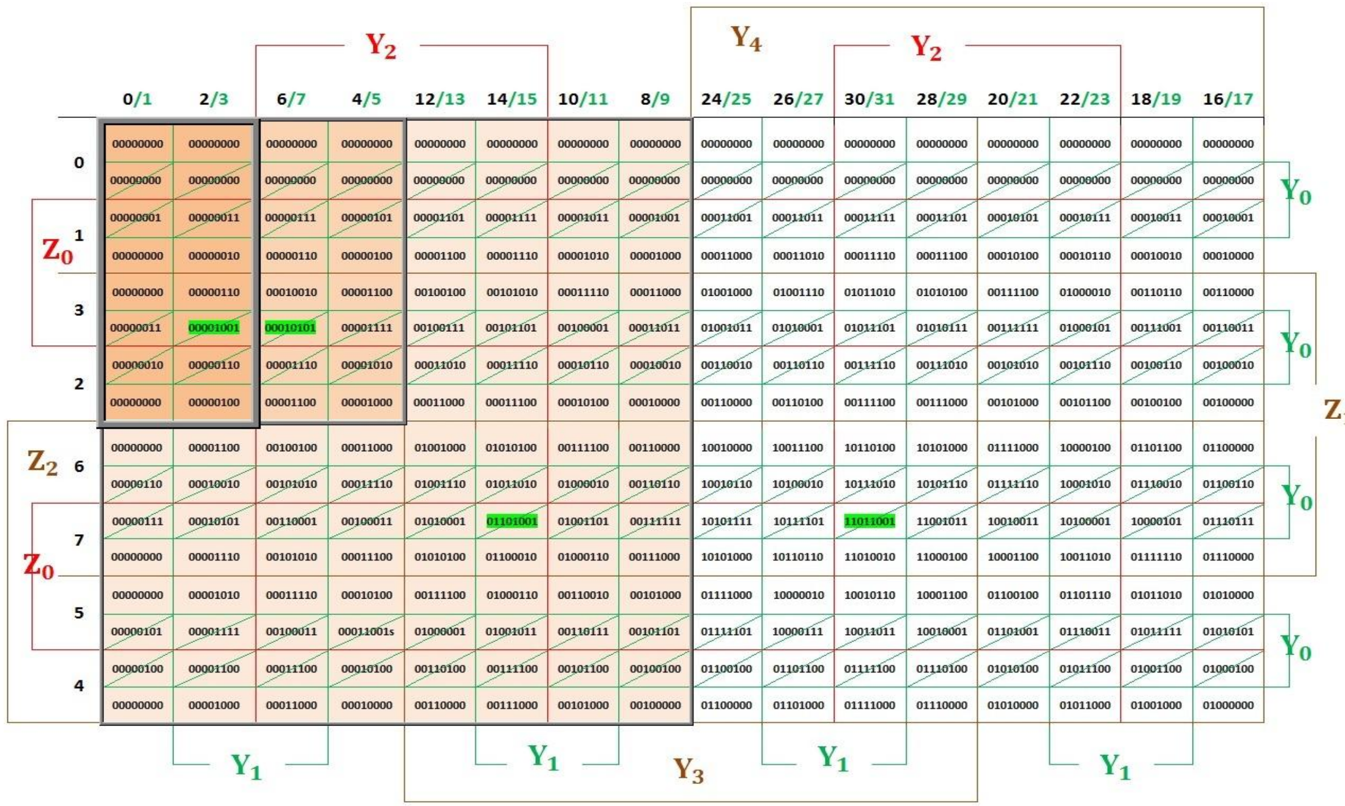


Fig. 1. The multiplication table for the (6, 5, 3) problem including its (5, 4, 3), (4, 3, 2), and (3, 2, 2) sub-problems

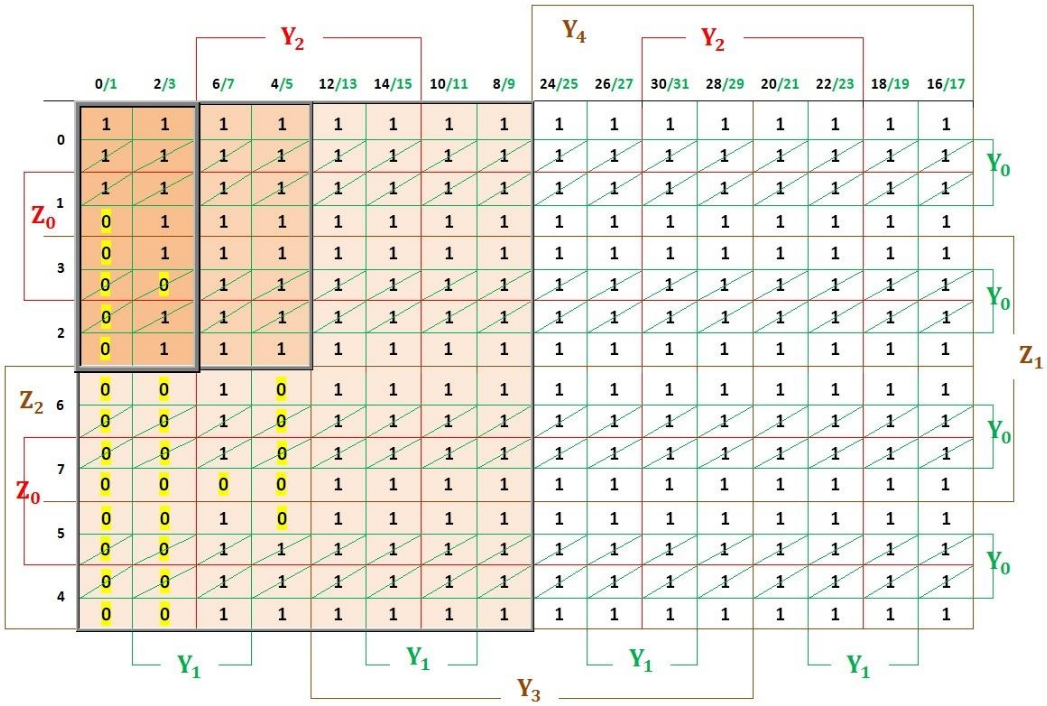
The table has the layout of an 8-variable Karnaugh map, where every column is topped by two possible decimal values for $(Y_4Y_3Y_2Y_1Y_0)$ and every two consecutive rows are labelled by a common decimal value for $(Z_2Z_1Z_0)$. Map entries represent the product $(Y_4Y_3Y_2Y_1Y_0) * (Z_2Z_1Z_0)$ in decimal. For convenience, composite numbers less than 64 having multiple factorizations are highlighted in red



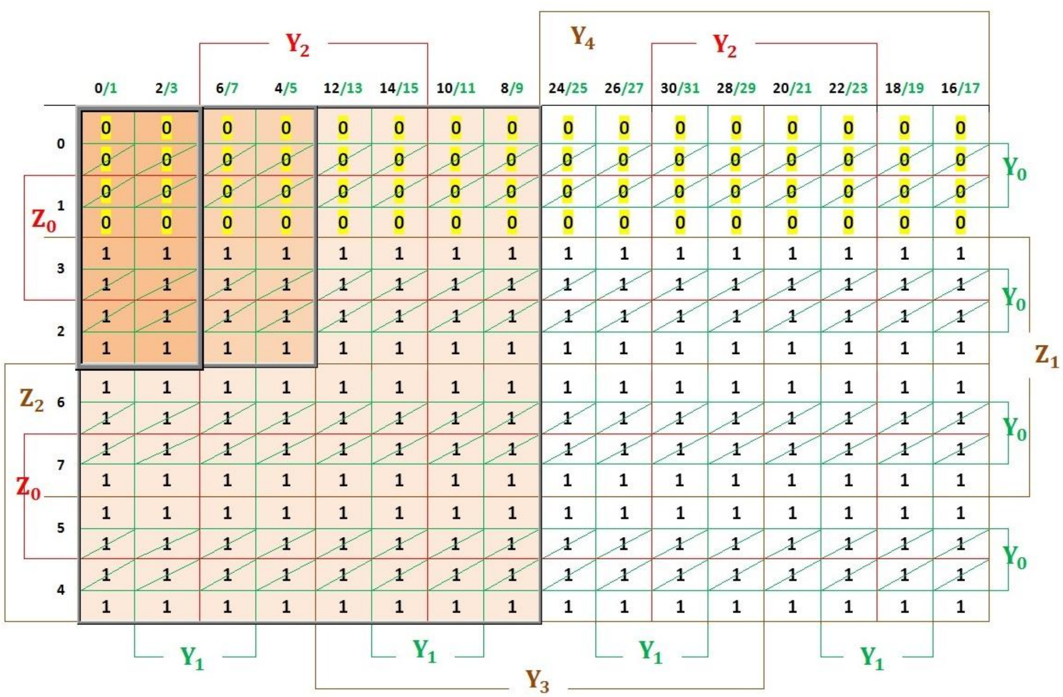
$\boldsymbol{g_0}(\mathbf{Y}, \mathbf{Z})$

Fig. 2. The multiplication table or map for the (6, 5, 3) problem including its (5, 4, 3), (4, 3, 2) and (3, 2, 2) sub-problems

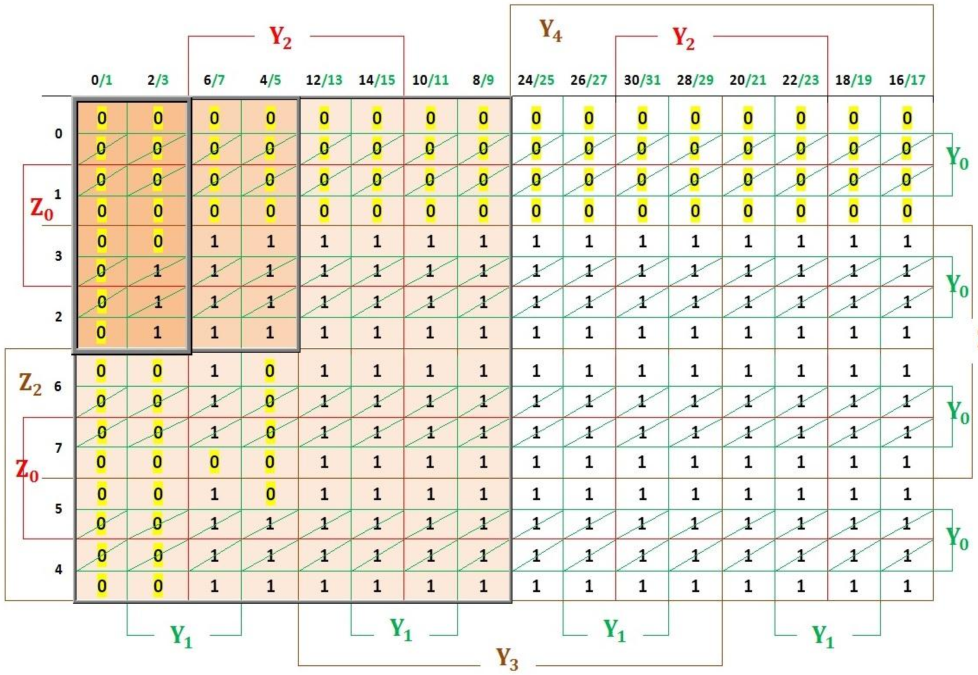
Map entries represent the product $(Y_4Y_3Y_2Y_1Y_0) * (Z_2Z_1Z_0)$ in binary (8 bits). This map also represents the initial-specification function $g_0(Y, Z)$ with binary strings understood to depict corresponding atoms. The largest entries in the map and its sub-maps (highlighted in green) are 217, 105, 21, and 9, and represent, 11011001, 01101001, 00010101, and 00001001 or, equivalently; $X_7X_6\bar{X}_5X_4X_3\bar{X}_2\bar{X}_1X_0$, $\bar{X}_7X_6X_5\bar{X}_4X_3\bar{X}_2\bar{X}_1X_0$, $\bar{X}_7\bar{X}_6\bar{X}_5X_4\bar{X}_3X_2\bar{X}_1X_0$, and $\bar{X}_7\bar{X}_6\bar{X}_5\bar{X}_4X_3\bar{X}_2\bar{X}_1X_0$.



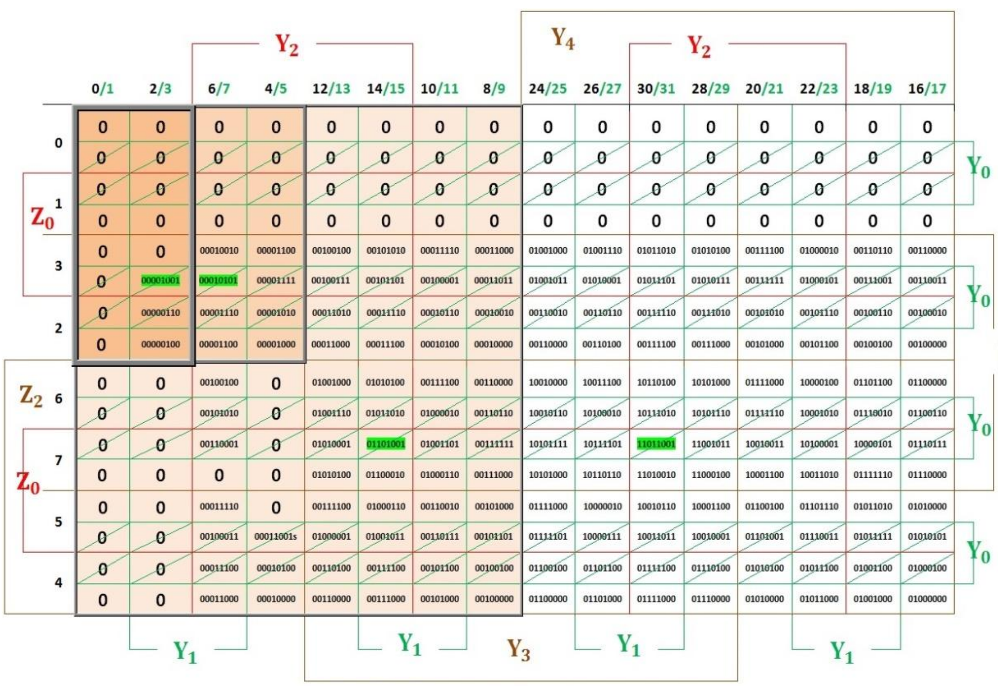
(a) $\mathbf{I(Y \geq Z)}$



(b) $\mathbf{I(Z > 1)}$



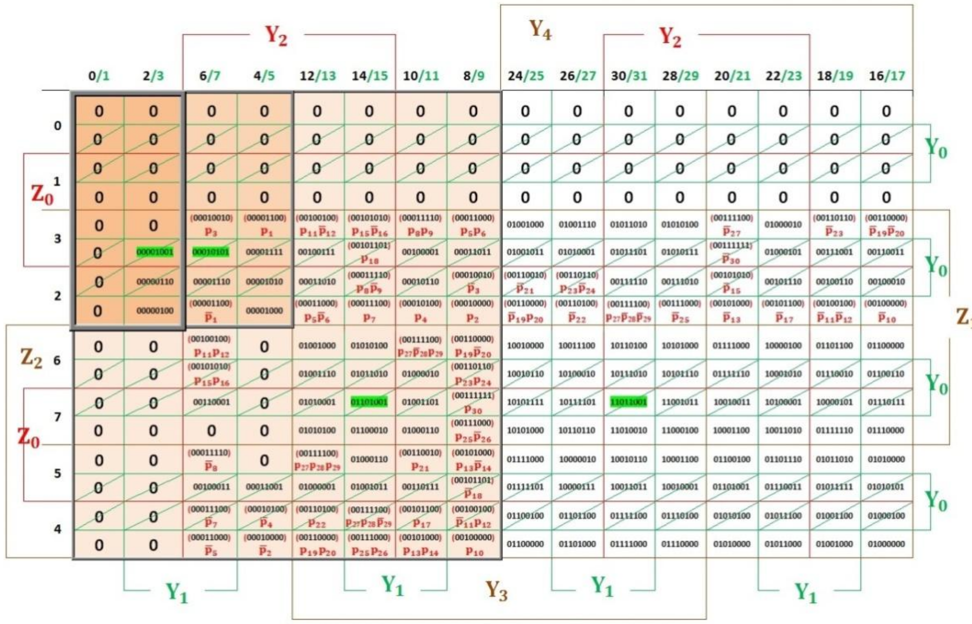
(c) $\mathbf{I(Z > 1)\, I(Y \geq Z)}$



(d) $\boldsymbol{g}(\mathbf{Y}, \mathbf{Z})$

Fig. 3. Evolution of the specification function $\boldsymbol{g}(\mathbf{Y}, \mathbf{Z})$

A non-zero entry in the $g(Y, Z)$ map symbolizes a single atom of the 256 atoms of $FB(X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0)$. For example, the entry 00111000 (binary for 56) denotes $(\bar{X}_7 \bar{X}_6 X_5 X_4 X_3 \bar{X}_2 \bar{X}_1 \bar{X}_0)$



Curtailed $\mathbf{G}(\mathbf{Y}, \mathbf{Z}, \mathbf{p})$

Fig. 4. The curtailed auxiliary function for the (6, 5, 3) problem

Fig. 2 is a replica of Fig .1, in which the entries $\mathbf{X}$ are converted from decimal to 8-bit binary representation. The figure can also be understood to represent the initial-specification function $g_0(\mathbf{Y}, \mathbf{Z})$ provided every string of bits $(T_7T_6T_5T_4T_3T_2T_1T_0)$ is understood to indicate the corresponding atom $(X_7^{T_7}X_6^{T_6}X_5^{T_5}X_4^{T_4}X_3^{T_3}X_2^{T_2}X_1^{T_1}X_0^{T_0})$. For example, the cell corresponding to $\mathbf{Y} = 31$ and $\mathbf{Z} = 7$ has an entry of $31 * 7 = (217)_{10} = (11011001)_2$ which is understood to represent $X_7X_6\overline{X}_5X_4X_3\overline{X}_2\overline{X}_1X_0$.

Fig. 3 describes the evolution of $g_0(\mathbf{Y}, \mathbf{Z})$ into $g(\mathbf{Y}, \mathbf{Z})$. Fig. 3(a) represents the indicator $I(\mathbf{Y} \geq \mathbf{Z})$, while Fig. 3(b) represents the indicator $I(\mathbf{Z} > 1)$. Fig. 3(c) is the product of Fig. 2, Fig. 3(a), and Fig. 3(b) and hence represents $g(\mathbf{Y}, \mathbf{Z})$. Fig. 4 represents the auxiliary function $G(\mathbf{Y}, \mathbf{Z}, \mathbf{p})$, curtailed to include parameters for the pertinent atoms among the 64 atoms in which $X_7 = X_6 = 0$. The transition from the specification function $g(\mathbf{Y}, \mathbf{Z})$ to the auxiliary function $G(\mathbf{Y}, \mathbf{Z}, \mathbf{p})$ is achieved according to the procedure in [10-12,40]. It is accomplished with the aid of Table 1, which identifies sets of orthonormal tags to be associated with atoms of multiple appearances in the map of $g(\mathbf{Y}, \mathbf{Z})$ in Fig. 3(d). Atoms beyond the initial 64 atoms are ignored in writing the final solution and its consistency condition, which turn out to be exactly the same as in Rushdi et al. [11]. For brevity, we have not repeated the logical expressions for the outputs $\mathbf{Z}$ and the consistency condition in this paper.

4 Algorithmic Implementation of Integer Factorization

Our manual work on integer factorization has to be stopped at the (6, 5, 3) problem. The next larger problem (namely, the (7, 6, 4)) has an input domain of 6+4=10 variables and would be considerably difficult (albeit, not totally impossible) for a Karnaugh-map treatment. Our solution procedure, on the other hand, is algorithmic in nature, and is amenable to coding as a computer program. In fact, we did write such a program in Matlab based on the knowledge accumulated throughout the manual solution of small problems. The program correctness was verified for the four problems solved manually, namely, the (3, 2, 2), (4, 3, 2), (5, 4, 3) and (6, 5, 3) problems. Fig. 5 outlines the scheme of the manual and automated approaches used herein. The figure also indicates directions for forthcoming work.

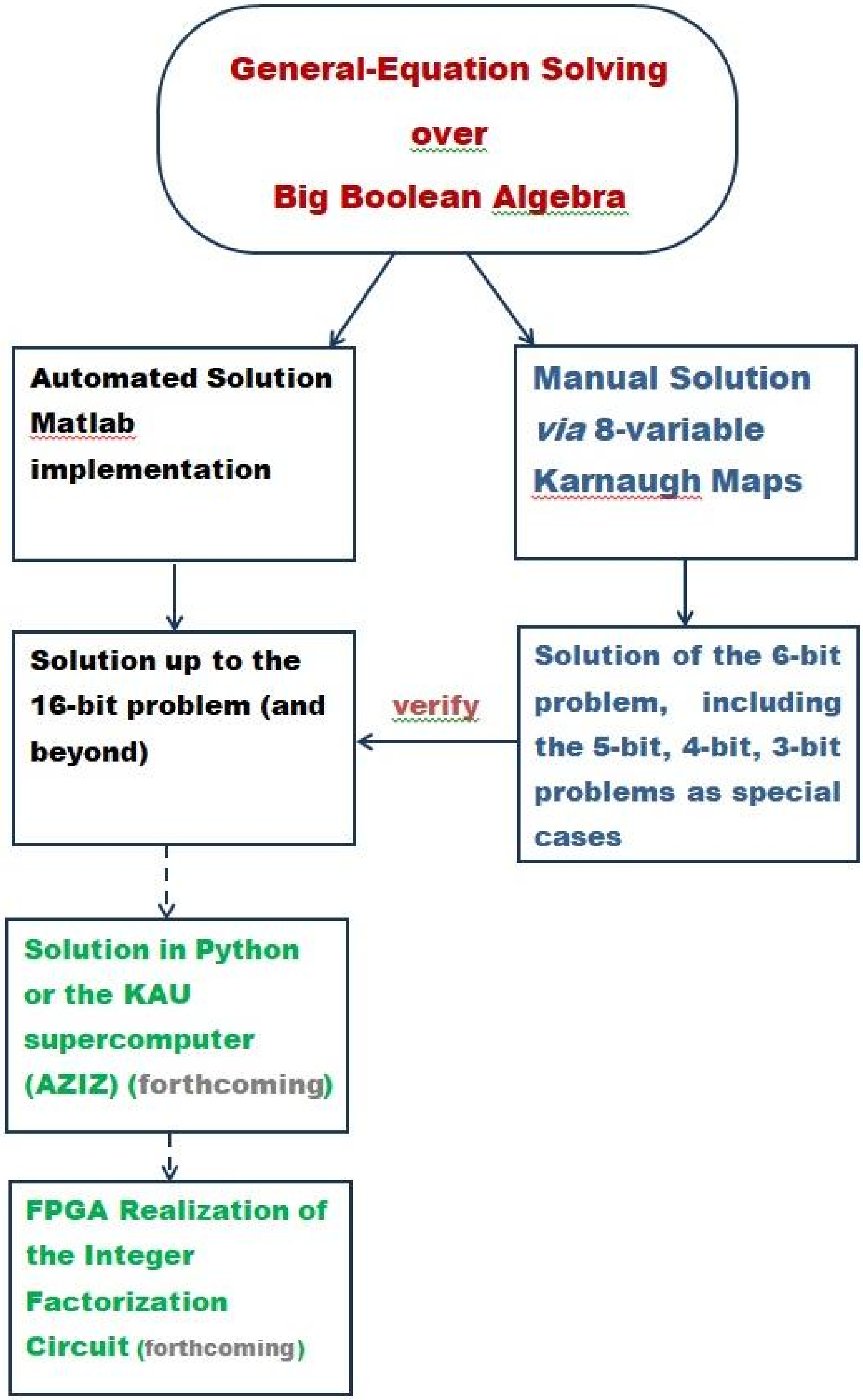


Fig. 5. An overview of our scheme for handling integer factorization

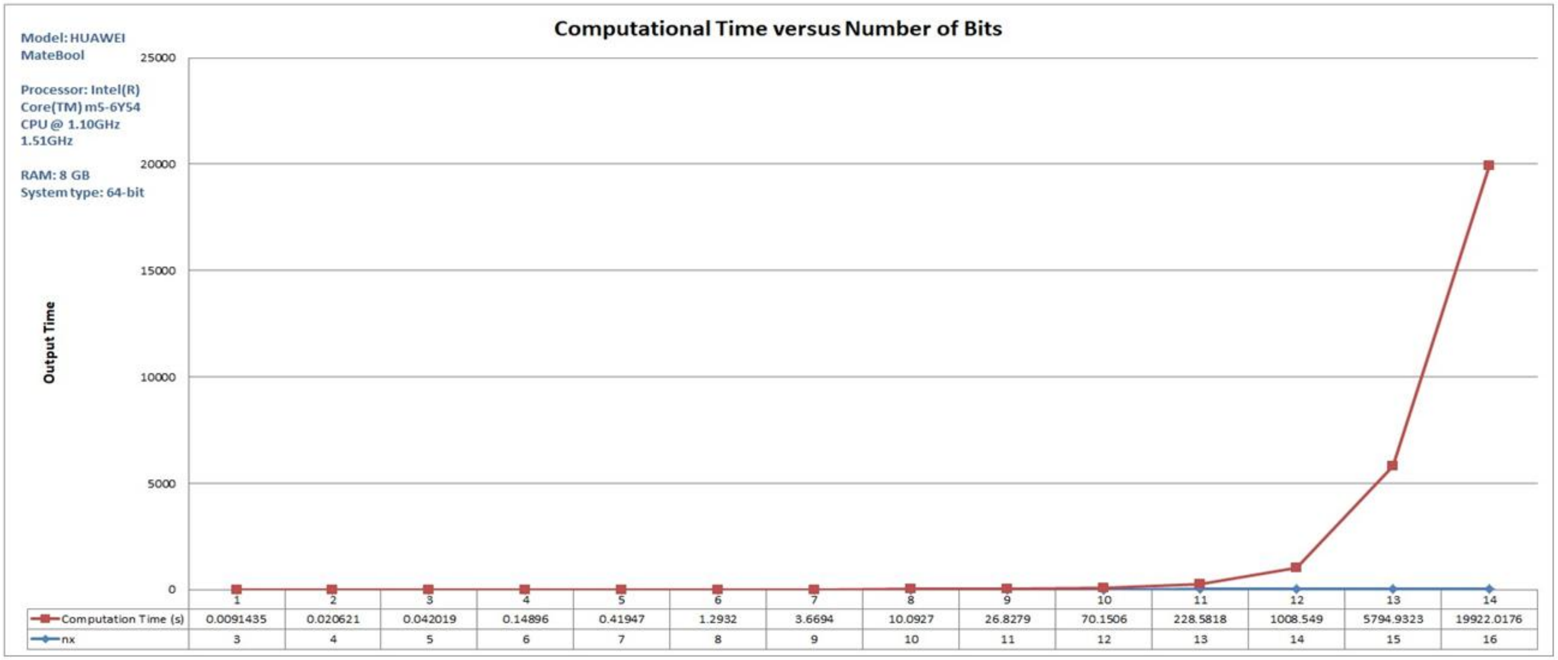


Fig. 6(a). Temporal complexity expressed as computational time versus the number of input bits

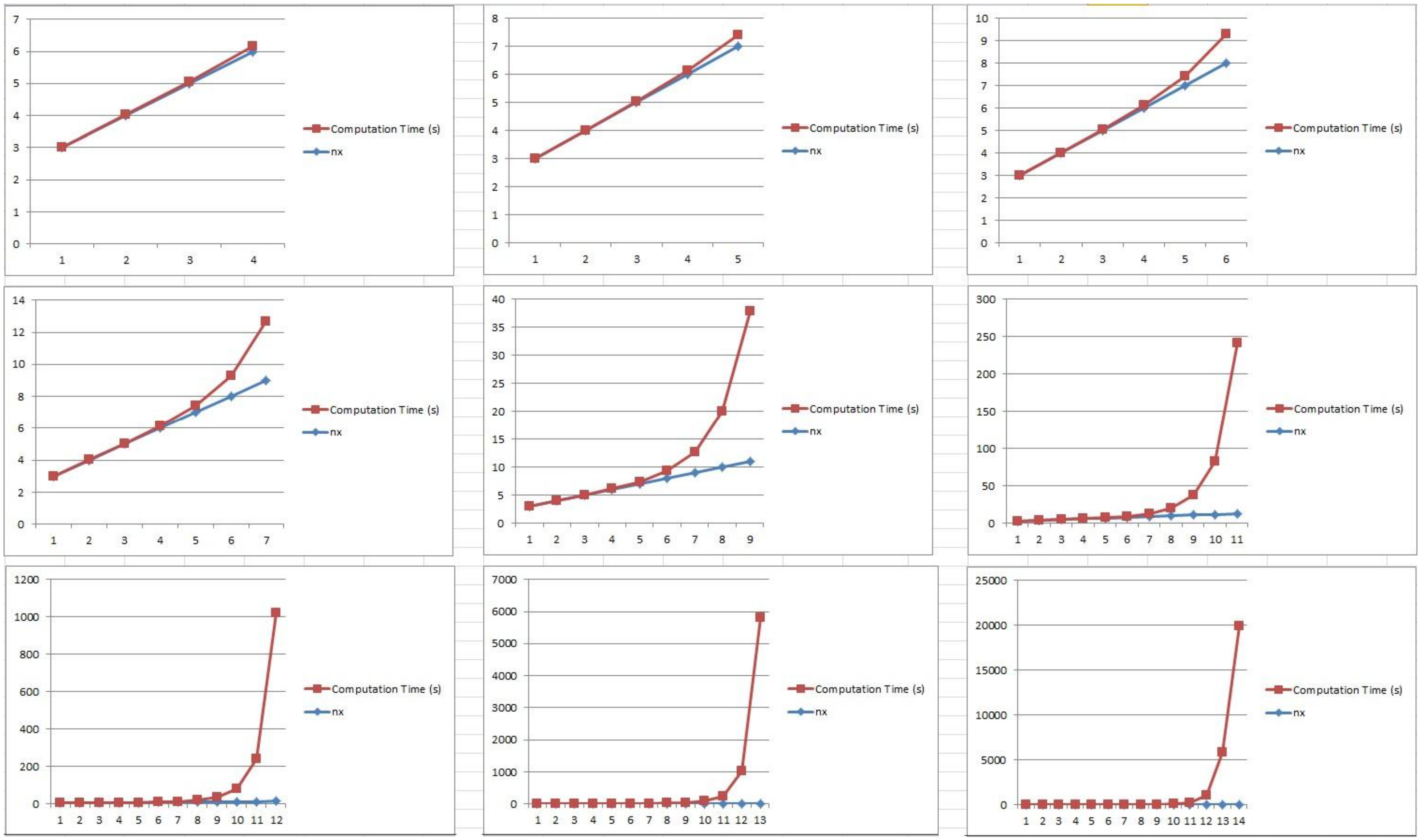


Fig. 6(b). Snapshots of the evolution of the temporal complexity in Fig. 6(a) as the number of input bits increases

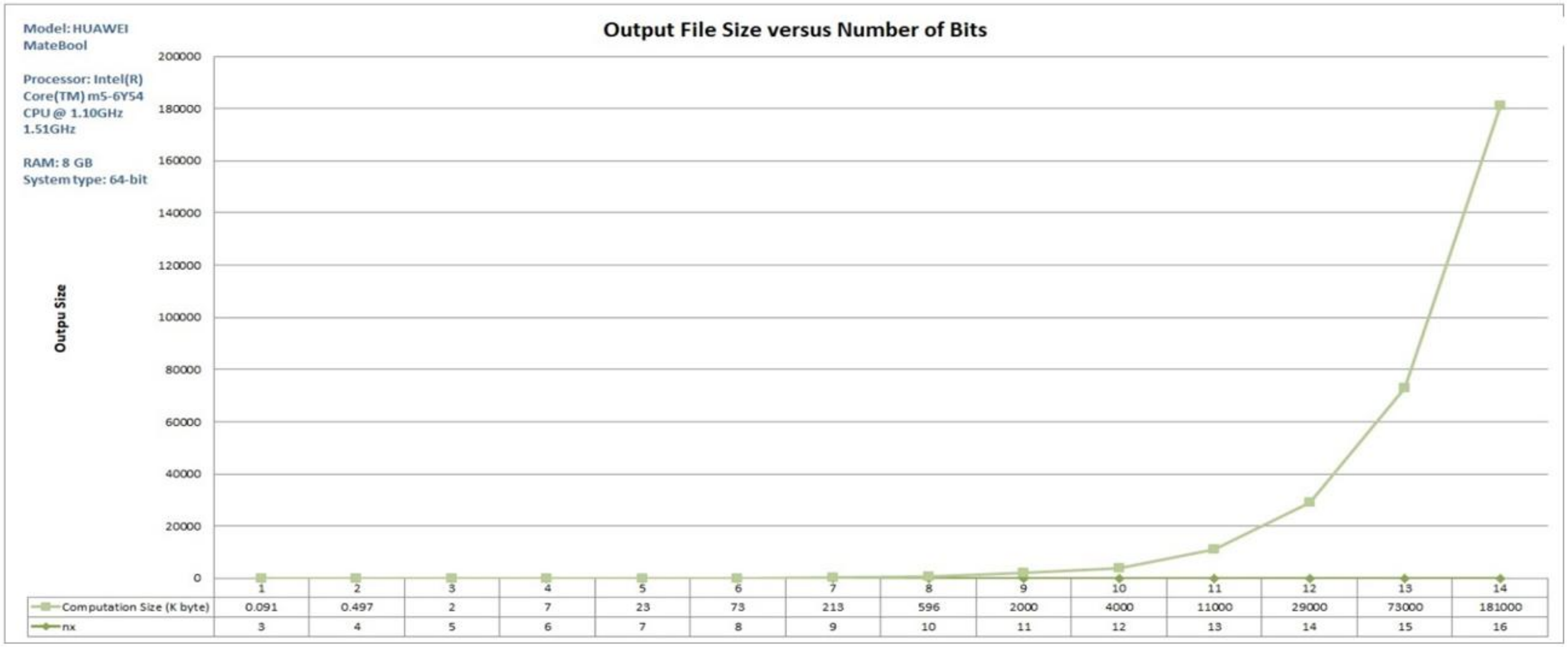


Fig. 7(a). Spatial complexity expressed as output file size versus the number of input bits

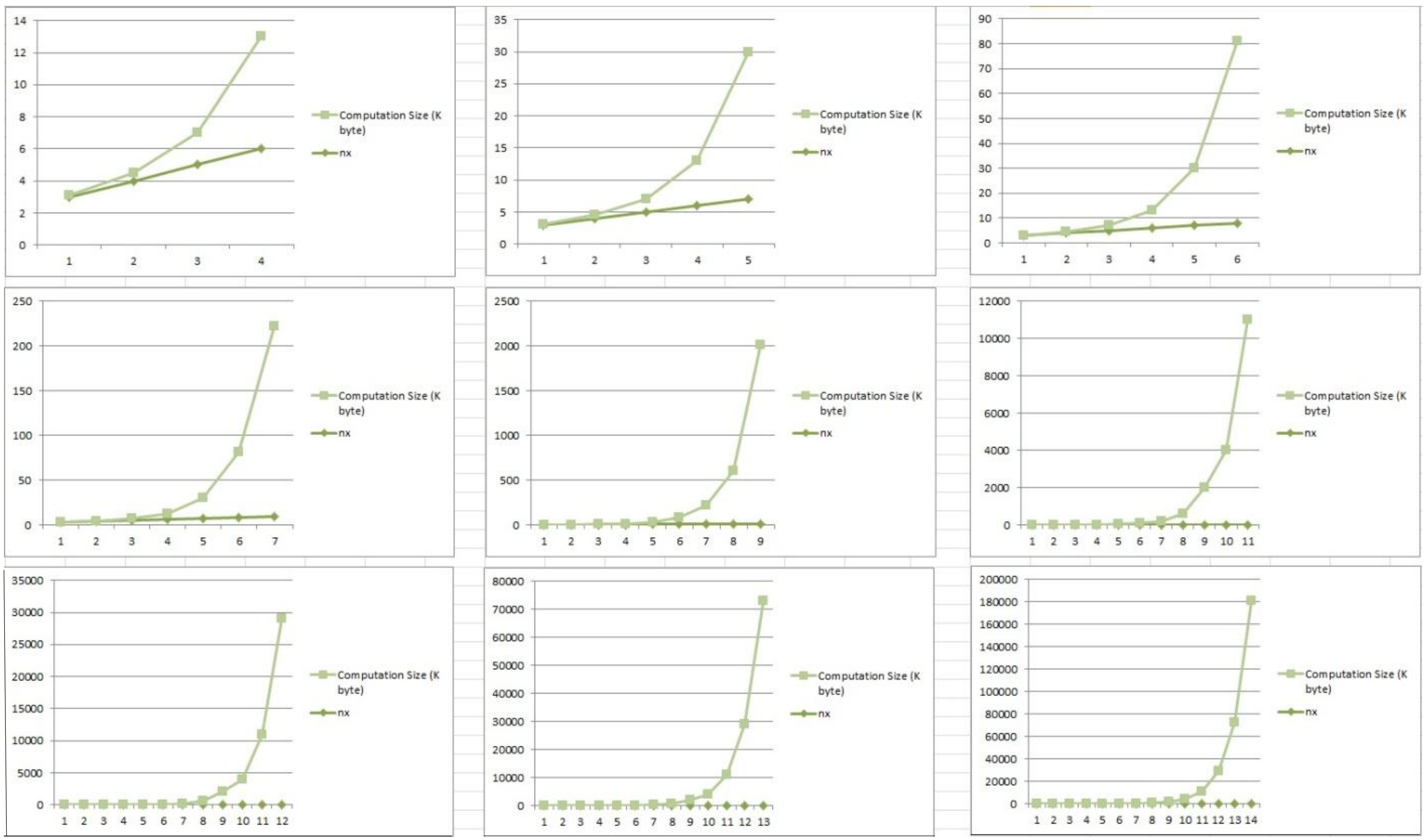


Fig. 7(b). Snapshots of the evolution of the spatial complexity in Fig. 7(a) as the number of input bits increases

We plan to try a new automated version using Python on AZIZ (King Abdulaziz University's super computer). The next step is to make an FPGA realization of the largest problem we manage to solve. So far, we were able to run the program for the 20-bit problem. The program might be used for larger problems as well, but both its execution time and output size are exponentially increasing, as clearly indicates by Figs. 6 and 7. The notorious "Curse of Dimensionality" is vividly demonstrated by the snapshots in Figs. 6(b) and 7(b) which show the evolution of the temporal and spatial complexities as the number of input bits increases.

5. Conclusions

Integer factorization is an intractable problem that might be handled in real time for small problems via hardware solution. Such a solution requires the extension of propositional logic to higher-order logics (e.g., first-order predicate logic) or the enlargement of two-valued Boolean algebra to a 'big' Boolean algebra. The paper derives a hardware circuit that factorizes a 6-bit integer **X** into two integers **Y** and **Z** of sizes 5 bits and 3 bits, respectively. The paper demonstrates that the resulting solution of the integer-factorization problem above includes the solutions of smaller problems as special cases. The paper builds on the experience gained in solving the 6-bit problem to design and implement a Matlab program to solve the general n-bit problem. The largest possible hardware circuit obtained via the automated solution is to be constructed, verified and tested. Such a hardware implementation (e.g., FPGA implementation) serves as a ready real-time look-up solution not only of the pertinent problem but also of all smaller problems.

Our contribution in this paper is admittedly a modest one and pertains mainly to a formulation of the integer factorization problem as a problem of Boolean equation-solving. This formulation gives a better insight into the problem and may provide opportunities to simplify the solution in the future. Our current solution method employs a strategy implicitly equivalent to that of a look-up table (albeit, with a more efficient enumeration). That is a main reason for the quick growth in the complexity of the solution.

The complexity in our solution comes from two sources. One involves the task of finding the Boolean expressions for the solution. This task is slow, but it has to be done once, and only once, for a problem of a given size. The other source of complexity is the size of the obtained expressions for the solutions. This is repeatedly unavoidable and would make a hardware implementation impractical for large problems due to large memory and time requirements. A possible way to reduce the complexity of our solution is to find the smallest factor (greater than 1) of the integer to be factorized and then, recursively, find the next factor, and so on.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Menezes A, Oorschot P, Vanstone S. Handbook of applied cryptography, CRC Press Company, New York, NY, USA; 1997.

[2] Crandall R, Pomerance C. The ubiquity of prime numbers, Chapter 8. In Prime Numbers A Computational Perspective Second Edition Springer, New York, NY, USA; 2005.

[3] Rushdi AMA, Alsheikhy AA. A pedagogical multi-key multi-stage package to secure communication channels. Journal of Qassim University: Engineering and Computer Sciences. 2017;10(2):105-124.

[4] Ahmad W, Rushdi AMA. A new cryptographic scheme utilizing the difficulty of big Boolean satisfiability. International Journal of Mathematical, Engineering and Management Sciences (IJMEMS). 2018;3(1):47-61.

[5] John AK, Shah S, Chakraborty S, Trivedi A, Akshay S. Skolem functions for factored formulas. In Proceedings of the 15th Conference on Formal Methods in Computer-Aided Design. FMCAD Inc. 2015;73-80.

[6] Fried D, Tabajara LM, Vardi MY. BDD-based Boolean functional synthesis. In International conference on computer aided verification. Springer International Publishing. 2016;402-421.

[7] Akshay S, Chakraborty S, John AK, Shah S. Towards parallel Boolean functional synthesis. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, Berlin, Heidelberg. 2017;337-353.

[8] Tabajara LM, Vardi MY. Factored Boolean functional synthesis. Formal Methods in Computer-Aided Design, FMCAD 2017, Vienna, Austria; 2017.

[9] Akshay S, Shah S, John A, Chakraborty S. Going beyond verification: Boolean function synthesis. Power Point Presentation; 2017.
Available:http://www.cfdvs.iitb.ac.in/workshop17/synthesis.pdf
(Accessed on December 24, 2017)

[10] Rushdi AM, Zagzoog SS. Design of a digital circuit for integer factorization via solving the inverse problem of logic. Advances in International Journal of Mathematical, Engineering and Management Sciences (IJMEMS). 2018;26 (3):1-14.

[11] Rushdi AM, Zagzoog SS, Balamesh AS. Design of a hardware circuit for integer factorization using a big Boolean algebra. Advances in International Journal of Mathematical, Engineering and Management Sciences (IJMEMS). 2018;27(1):1-25.

[12] Rushdi AM, Zagzoog SS. Derivation of all particular solutions of a 'big' Boolean equation with applications in digital design. Current Journal of Applied Science and Technology. 2018;27(3):1-16.

[13] Kim HJ, Mangione-Smith WH. Factoring large numbers with programmable hardware. In Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays. ACM. 2000;41-48.

[14] Bernstein DJ. Circuits for integer factorization: A proposal, manuscript; 2001.
Available:http//cr.yp.to/papers.html#nfscircui

[15] Lenstra AK, Shamir A, Tomlinson J, Tromer E. Analysis of Bernstein's factorization circuit. In International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg. 2002;1-26.

[16] Geiselmann W, Steinwandt R. A dedicated sieving hardware. In International workshop on public key cryptography. Springer, Berlin, Heidelberg. 2003;254-266.

[17] Shamir A, Tromer E. Factoring large numbers with the TWIRL device. In Annual International Cryptology Conference. Springer, Berlin, Heidelberg. 2003;1-26.

[18] Geiselmann W, Steinwandt R. Yet another sieving device. In Cryptographers' Track at the RSA Conference. Springer, Berlin, Heidelberg. 2004;278-291.

[19] Geiselmann W, Shamir A, Steinwandt R, Tromer E. Scalable hardware for sparse systems of linear equations, with applications to integer factorization. In International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg. 2005;131-146.

[20] Geiselmann W, Januszewski F, Köpfer H, Pelzl J, Steinwandt R. A simpler sieving device: Combining ECM and TWIRL. In International Conference on Information Security and Cryptology. Springer, Berlin, Heidelberg. 2006;118-135.

[21] Izu T, Kogure J, Shimoyama S. CAIRN 3: An FPGA implementation of the sieving step with the lattice sieving. Proc. of the 2007 Special-purpose Hardware for Attacking Cryptographic Systems (SHARCS 2007). 2007;33-39.

[22] De Meulenaer G, Gosset F, De Dormale GM, Quisquater JJ. Integer factorization based on elliptic curve method: Towards better exploitation of reconfigurable hardware. In Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on. IEEE. 2007; 197-206.

[23] De Meulenaer G, Gosset F, De Dormale GM, Quisquater JJ. Elliptic curve factorization method: Towards better exploitation of reconfigurable hardware. In SHARCS Workshop Record. 2007;21-31.

[24] Yu H, Bai G. An efficient method for integer factorization. In Circuits and Systems (ISCAS), 2015 IEEE International Symposium on. IEEE. 2015;73-76.

[25] Ahuja NA, Subedar M, Lee Y, Tickoo O. A factorization approach for enabling structure-frommotion/SLAM using integer arithmetic. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017;554-562.

[26] Bos JW, Naehrig M, Pol JVD. Sieving for shortest vectors in ideal lattices: A practical perspective. International Journal of Applied Cryptography. 2017;3(4):313-329.

[27] Monaco JV, Vindiola MM. Integer factorization with a neuromorphic sieve. In Circuits and Systems (ISCAS), 2017 IEEE International Symposium on. IEEE. 2017;1-4.

[28] Monaco JV, Vindiola MM. Factoring integers with a brain-inspired computer. IEEE Transactions on Circuits and Systems I: Regular Papers. 2018;65(3):1051-1062.

[29] Hammer PL, Rudeanu S. Boolean methods in operations research and related areas. Springer Verlag, Berlin, Germany; 1968.

[30] Rudeanu S. Boolean functions and equations. North-Holland Publishing Company & American Elsevier, Amsterdam, the Netherlands; 1974.

[31] Brown FM, Rudeanu S. Recurrent covers and Boolean equations, Colloq. On Lattice Theory, Szeged, Hungary, Published in Colloquia Mathematica Societatis Janos Bolyai, North Holland Publishing Company, Amsterdam, The Netherlands. 1983;55-86.

[32] Brown FM. Boolean reasoning: The logic of Boolean equations. Kluwer Academic Publishers, Boston, USA; 1990.

[33] Tucker JH, Tapia MA. Solution of a class of Boolean equations. Proceedings of IEEE Southeastcon 95, New York, NY, USA. 1995;1:106-112.

[34] Woods S, Casinovi G. Efficient solution of systems of Boolean equations. Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design. 1996;542-546.

[35] Brusentsov NP, Vladimirova YS. Solution of Boolean equations. Computational Mathematics and Modeling. 1998;9(4):287-295.

[36] Rudeanu S. Lattice functions and equations. Springer Verlag, London, UK; 2001.

[37] Rushdi AM, Ba-Rukab OM. Low-cost design of multiple-output switching circuits using map solutions of Boolean equations. Umm Al-Qura University Journal: Science, Medicine and Engineering. 2003;15(2):59-79.

[38] Rushdi AM. Efficient solution of Boolean equation using variable-entered Karnaugh maps. Journal of King Abdulaziz University: Engineering Sciences. 2004;15(2):21-29.

[39] Rudeanu S. Boolean sets and most general solutions. Information Sciences. 2010;180:2440-2447.

[40] Rushdi AM, Amashah MH. Using variable–entered Karnaugh maps to produce compact parametric general solutions of Boolean equations. International Journal of Computer Mathematics. 2011; 88(15): 3136-3149.

[41] Crama Y, Hammer PL. Boolean functions: Theory, algorithms, and applications. Cambridge University Press, Cambridge, United Kingdom; 2011.

[42] Rushdi AM. A comparison of algebraic and map methods for solving general Boolean equations. Journal of Qassim University: Engineering and Computer Sciences. 2012;5(2):147-173.

[43] Rushdi AMA, Albarakati HM. The inverse problem for Boolean equations. Journal of Computer Science. 2012;8(12):2098-2105.

[44] Rushdi AM, Albarakati HM. Using variable-entered Karnaugh maps in determining dependent and independent sets of Boolean functions. Journal of King Abdulaziz University: Computer and Information Technology. 2012;1(2):45-67.

[45] Rushdi AMA, Amashah MH. Purely-algebraic versus VEKM methods for solving big Boolean equations. Journal of King Abdulaziz University: Engineering Sciences. 2012;23(2):75-85.

[46] Rushdi AMA, Albarakati HM. Prominent classes of the most general subsumptive solutions of Boolean equations. Information Sciences. 2014;281:53-65.

[47] Rushdi AMA, Al-Qwasmi MA. Formal derivation of a particular input of a single AND (OR) gate in terms of its output and other inputs. Journal of King Abdulaziz University: Engineering Sciences. 2015;26(2):51-62.

[48] Rushdi AMA, Ahmad W. Satisfiability in 'big' Boolean algebras via Boolean-equation solving, Journal of King Abdulaziz University: Engineering Sciences. 2016;28(1):3-18.

[49] Rushdi AMA, Ahmad W. A novel method for compact listing of all particular solutions of a system of Boolean equations. British Journal of Mathematics & Computer Science. 2017;22(6):1-18.

[50] Rushdi AMA. Handling generalized type-2 problems of digital circuit design via the variable-entered Karnaugh map. International Journal of Mathematical, Engineering and Management Sciences (IJMEMS). 2018;3(4):392-403.

[51] Rushdi AMA, Ahmad W. Digital circuit design utilizing equation solving over ‘big’ Boolean algebras. International Journal of Mathematical, Engineering and Management Sciences (IJMEMS). 2018;3(4):404-428.

[52] Rushdi AMA, Ahmad W. A comparison of the methods of Boolean-equation solving and input-domain constraining for handling type-2 problems of digital circuit design. Current Journal of Applied Science and Technology, 2018, 29(2):1-15.

[53] Rushdi AM. Symbolic reliability analysis with the aid of variable-entered Karnaugh maps. IEEE Transactions on Reliability, 1983;32(2):134-139.

[54] Rushdi AM. Improved variable-entered Karnaugh map procedures. Computers and Electrical Engineering. 1987;13(1):41-52.

[55] Rushdi AM, Al-Yahya HA. A Boolean minimization procedure using the variable-entered Karnaugh map and the generalized consensus concept. International Journal of Electronics. 2000;87(7):769-794.

[56] Rushdi AM, Al-Yahya HA. Further improved variable-entered Karnaugh map procedures for obtaining the irredundant forms of an incompletely-specified switching function. Journal of King Abdulaziz University: Engineering Sciences. 2001;13(1):111-152.

[57] Rushdi AM. Prime-implicant extraction with the aid of the variable-entered Karnaugh map. Umm Al-Qura University Journal: Science, Medicine and Engineering. 2001;13(1):53-74.

[58] Rushdi AMA, Ghaleb FAM. The Walsh spectrum and the real transform of a switching function: A review with a Karnaugh-map perspective. Journal of Qassim University: Engineering and Computer Sciences. 2014;7(2):73-112.

[59] Rushdi AM, Rushdi MA. Switching-algebraic analysis of system reliability, Chapter 6 in Ram M, Davim P. (Editors), Advances in Reliability and System Engineering. Springer International Publishing, Cham, Switzerland, 2017;139-161.

[60] Rushdi AMA. Utilization of Karnaugh maps in multi-value qualitative comparative analysis. International Journal of Mathematical, Engineering and Management Sciences. 2018;3(1):28-46.

[61] Rushdi RA, Rushdi AM. Karnaugh-map utility in medical studies: The case of fetal malnutrition. International Journal of Mathematical, Engineering and Management Sciences (IJMEMS). 2018;3(3): 220-244.
Available:www.ijmems.in/ijmems—volumes.html

[62] Dean KJ. An extension of the use of Karnaugh maps in the minimization of logical functions. Radio and Electronic Engineer. 1968;35(5):294-296.

[63] Motil JM. Views of digital logic & probability via sets, numberings; 2017.
Available:http://www.csun.edu/~jmotil/ccSetNums2.pdf

[64] Booth TM. The vertex-frame method for obtaining minimal proposition-letter formulas. IRE Transactions on Electronic Computers. 1962;2:144-154.

[65] Halder AK. Karnaugh map extended to six or more variables. Electronics Letters. 1982;18(20):868-870.

[66] Rushdi AM, Al-Khateeb DL. A review of methods for system reliability analysis: A Karnaugh-map perspective. In Proceedings of the First Saudi Engineering Conference, Jeddah, Saudi Arabia. 1983; 1:57-95.

[67] Rushdi AM. Overall reliability analysis for computer-communication networks. In Proceedings of the Seventh National Computer Conference, Riyadh, Saudi Arabia. 1984;23-38.

[68] Rushdi AM. On reliability evaluation by network decomposition. IEEE Transactions on Reliability, Corrections: ibid, 34(4),319. 1984;33(5):379-384.

APPENDIX A

Integer-Factorization Matlab Code

test01.m

```
close all;clear;clc;

nx=6;
%find n such that nn=2n or nn=2n-1

n=ceil(nx/2);

nz=n;
ny=nx-1;
c=zeros(1,2^nx);

d=[];
db=[];

for z=2:2^nz-1
        for y=z:2^ny-1
                if y*z<=2^nx-1
                        d=[d;y z y*z 0];
                        c(y*z+1)=c(y*z+1)+1;
                end
        end
end

ps=getparams(c);
k=find(c>=2);

for i=1:length(k)
        kk=find(d(:,3)==k(i)-1);
        d(kk,4)=(1:c(k(i)))';
end

ybits=mydec2bin(d(:,1),ny);
zbits=mydec2bin(d(:,2),nz);

for i=1:ny
        ex=['y' num2str(ny-i) '='];
        k=find(ybits(:,i)==1);
        dd=d(k,:);
        for j=1:size(dd,1)
        txt='';
        xbits=mydec2bin(dd(j,3),nx);
                for q=1:nx
                txt=[txt 'x' num2str(nx-q)];
                        if xbits(q)==0
                                txt=[txt ''''];
```

```
                end
            end
            if dd(j,4)>0
                atom=dd(j,3);
                param=ps{atom+1}{dd(j,4)};
                txt=[txt ' ' param];
            end
        if j>1, txt = [ ' + ' txt]; end
            ex=[ex txt];
        end
disp(ex);
end
```

getparams.m

```
function ps=getparams(c)

cc=c(c>=2);

nparams=sum(ceil(log2(cc)));

k=1;
for i=1:length(c)
    if c(i)<=1
        ps{i}='';
    else
        np=ceil(log2(c(i)));
        nmult=c(i);
        ps{i}=spantree(np,nmult,k);
        k=k+np;
    end
end
```

mydec2bin.m

```
function y=mydec2bin(x,b)

y=dec2bin(x,b);
y=double(y)==49;
y=double(y);
```

spantree.m

```
function pp=spantree(np,nmult,vindex)

%np=3;
%nmult=5;
%vindex=27;

tree=[-1 0 0 0 0];

node=1;
while true
```

```
          nnodes=size(tree,1);
          nendnodes=sum(tree(:,4));

          if nendnodes>=nmult, break; end
          if tree(node,2)==0 && tree(node,5)<np
                    tree(node,4)=0;
                    nnodes=nnodes+1;
                    newnode=[node 0 0 1 tree(node,5)+1];
                    tree(node,2)=nnodes;
                    tree=[tree;newnode];
                    node=nnodes;
          elseif tree(node,3)==0 && tree(node,5)<np
                    nnodes=nnodes+1;
                    newnode=[node 0 0 1 tree(node,5)+1];
                    tree(node,3)=nnodes;
                    tree=[tree;newnode];
                    nendnodes=nendnodes+1;
                    node=nnodes;
                    if nendnodes>=nmult, break; end
          else
                    node=tree(node,1);
          end
          if size(tree,1)>8, break; end
end

k=find(tree(:,4)==1);
for i=1:length(k)
          node=k(i);
          pp{i}='';
          while true
                    depth=tree(node,5);
                    px=['p' num2str(vindex+depth-1)];
                    pre=tree(node,1);
                    if tree(pre,3)==node
                              px=[px ''''];
                    end
                    pp{i}=[px pp{i}];
                    node=pre;
                    if pre==1, break; end
          end
end
```

© 2019 Rushdi et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://www.sciencedomain.org/review-history/27962