Journal of Advances in Mathematics and Computer Science

26(3): 1-14, 2018; Article no.JAMCS.39285
ISSN: 2456-9968
(Past name: British Journal of Mathematics & Computer Science, Past ISSN: 2231-0851)

Design of a Digital Circuit for Integer Factorization *via* Solving the Inverse Problem of Logic

Ali Muhammad Ali Rushdi[1*] and Sultan Sameer Zagzoog[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 between the two authors. Author AMAR designed the study, performed the analysis, solved the example and wrote the manuscript. Author SSZ managed the literature search and drew the figures. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JAMCS/2018/39285
Editor(s):
(1) Feyzi Basar, Professor, Department of Mathematics, Fatih University, Turkey.
Reviewers:
(1) Anuj Kumar Goel, Maharishi Markandeshwar University, India.
(2) Fatih Başçiftçi, Selçuk University, Turkey.
(3) Nikos Petrellis, Technological Educational Institute of Thessaly, Greece.
Complete Peer review History: http://www.sciencedomain.org/review-history/23038

Received: 9th November 2017
Accepted: 23rd January 2018
Published: 5th February 2018

Original Research Article

Abstract

In standard problems of digital circuit design, a switching function (two-valued Boolean function) is specified declaratively as a (usually incomplete) asserted relation $\mathrm{R}(\mathbf{X},\mathbf{Z})$, or equivalently as an equation $\mathrm{R}(\mathbf{X},\mathbf{Z}) = 1$, where $\mathbf{X}$ and $\mathbf{Z}$ are inputs and outputs, respectively. To obtain such a function constructively, one might use Boolean-function synthesis (which enlarges propositional logic to first-order predicate logic), or use a 'big' Boolean algebra (which acts as an enlargement of switching algebra). This paper explores the utility of Boolean-equation solving in handling the hard or intractable problem of integer factorization by constructing a hardware circuit that achieves this purpose in real time (at least for reasonably large bit sizes). The feasibility of the proposed scheme is verified via the manual solution of the smallest possible problem. However, the results obtained are really encouraging, as they can be automated in a straightforward fashion. A sequel forthcoming paper will treat the scaling, complexity, and automation issues, and will, in particular, determine the upper limit on the bit size that can be treated by the current technique.

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

Keywords: 'Big' Boolean algebra; digital design; Integer factorization; Boolean function decomposition.

1 Introduction

Design of digital circuits is customarily accomplished in the realm of Switching Algebra (two-valued Boolean Algebra B_2) or equivalently in the realm of Propositional Logic (PL) [1-4]. With contemporary digital design increasing in sophistication and complexity, it became necessary to enlarge this realm to a more powerful entity including it as a special case. Fig. 1 proposes two schemes for enlarging B_2 or PL into a more powerful domain, while Table 1 compares these two schemes. The first scheme leads to *Boolean-Function Synthesis,* which starts by a relational specification $R(\mathbf{X}, \mathbf{Z})$ among the inputs $\mathbf{X}$ and outputs $\mathbf{Z}$, synthesizes each output as a function of the inputs such that the specification holds. Such a synthesis involves some special functions of mathematical logic called Skolem Functions [5-8]. The second competitive (albeit probably less well known) scheme resorts to the use of 'big' finite Boolean algebras, which are complemented distributive lattices, each with n generators ($n \geq 1$), $N = 2^n$ atoms, and $2^{2^n} = 2^N$ elements.

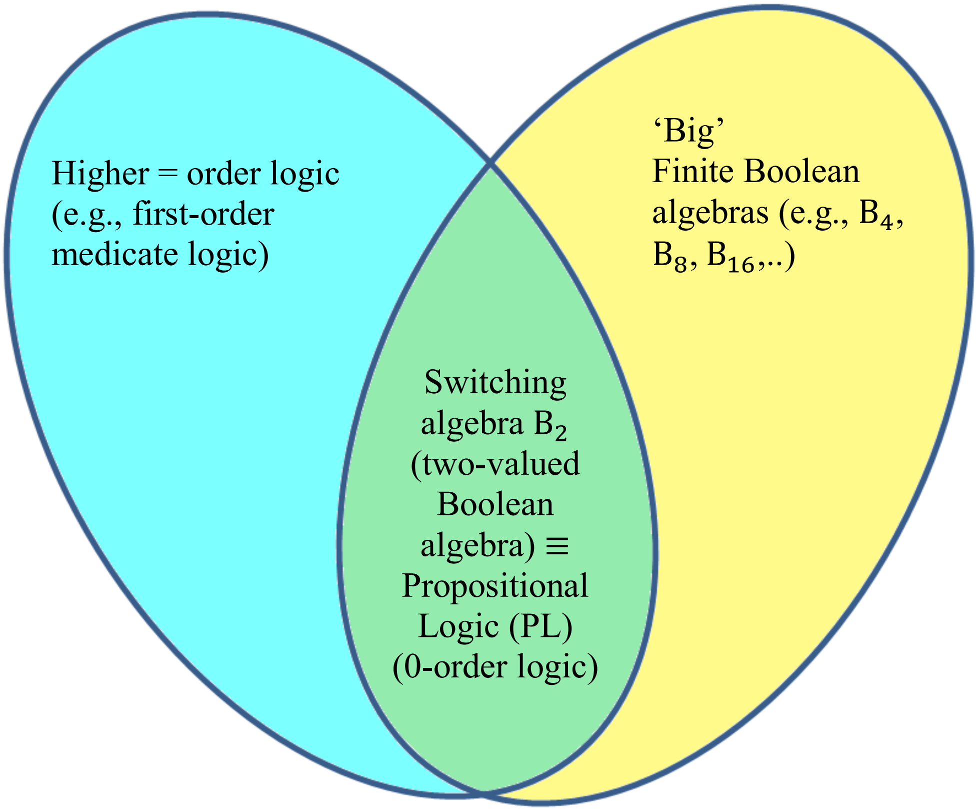


Fig. 1. The relation among various algebras and logics that might be needed in contemporary digital circuit design

Table 1. Comparison for two schemes of enlarging the domain of digital circuit design

Scheme	**Boolean-Function synthesis**	**Boolean-Equation solving**
Enlargement for two-valued Boolean algebra B_2 (propositional logic)	Higher-order logics (e.g., first-order predicate logic).	'big' finite Boolean algebras (e.g., B_4, B_8, B_{16}, …).
Declarative specification	A desired input-output relation $\mathrm{R}(\mathbf{X},\mathbf{Z})$ between input variables $\mathbf{X} \in \mathrm{B_2}^m$ and output variables $\in \mathrm{B_2}^n$. According to the Principle of Assertion, asserting this relation is equivalent to equating it to 1 [10]	An equation originally stated as $\mathrm{R}(\mathbf{X},\mathbf{Z}) = 1$ but subsequently viewed as $\mathrm{R}(\mathbf{Z}) = 1$ where R is a 'big' Boolean function $\mathrm{R} = \mathrm{B}^n \rightarrow \mathrm{B}$, and $\mathrm{B} = \mathrm{FB}(\mathbf{X})$ is a 'big' Boolean algebra of m generators, $M = 2^m$ atoms, and $2^M = 2^{2^m}$ elements.
Constructive solution	Synthesize a function $\mathbf{F} = \mathrm{B_2}^m \rightarrow \mathrm{B_2}^n$ such that for every $\mathbf{X}$ if there is a value of $\mathbf{Z}$ such that $\mathbf{Z} = \mathbf{F}(\mathbf{X})$ then $\mathrm{R}(\mathbf{X},\mathbf{F}(\mathbf{X})) = 1$, i.e., specify outputs $\mathbf{Z}$ as a function $\mathbf{Z}(\mathbf{X})$ of input $\mathbf{X}$ such that $\mathrm{R}(\mathbf{X},\mathbf{Z})$ holds (evaluate to true). Such a function is refused to as a *skolem function* for $\mathbf{Z}$ in $\mathrm{R}(\mathbf{X},\mathbf{Z})$ [5-8]	Solve the 'big' Boolean equation $\mathrm{R}(\mathbf{Z}) = 1$ for $\mathbf{Z}$ as a function $\mathbf{Z}(\mathbf{X})$ expressing the outputs $\mathbf{Z}$ in terms of the inputs $\mathbf{X}$.
Treatment of inputs $\mathbf{X}$ that admit no outputs $\mathbf{Z}$	For values of $\mathbf{X}$ that do not admit any value of $\mathbf{Z}$ such that $\mathrm{R}(\mathbf{X},\mathbf{Z})$ holds, the value of $\mathbf{F}(\mathbf{X})$ is inconsequential, i.e., we do not care what the function outputs.	The Boolean-equation technique identifies $\mathbf{X}$ values that do not admit $\mathbf{Z}$ values via a specific consistency condition that (possibly) annihilates the atoms in $\mathrm{FB}(\mathbf{X})$ covesponding to these $\mathbf{X}$ values which consequently forces $\mathrm{FB}(\mathbf{X})$ to collapse to a subalgebra. The technique adds these values as don't-care to the solutions $\mathbf{Z}$.

In passing, we stress that the two aforementioned schemes are essentially equivalent. The equivalence of the two schemes stems from an axiom peculiar to the Calculus of Propositions, called the Principle of Assertion [9,10] which states that "To say that a proposition is true is to state the proposition itself, " namely

$$[\mathrm{A} = 1] = \mathrm{A}, \tag{1}$$

Consequently, it is possible in the Calculus of Propositions to dispense entirely with equations.

One of the motivating problems for the first scheme is the problem of integer factorization, which is ubiquitous in scientific applications, including, in particular, the celebrated RSA Cryptosystems [11]. Though this problem has many sophisticated algorithms in practice, it is known to be a hard or intractable problem. Its best solvers in Boolean function synthesis have been able to solve only up to 12 bits [12].

The purpose of this paper is to handle the problem of integer factorization using the aforementioned second scheme. We demonstrate the feasibility, and expose the details, of this second scheme by manually solving a toy problem of 4 bits only. We point out the possibility of automating the solution, so that it might be applied in the design of larges factorization circuits. Such a design is expected to be limited only by the finiteness of the computational sources available.

The organization of the rest of this paper is as follows. In Section 2, we digress a little bit to give a quick review of the postulates of Boolean algebras and useful facts about them. Section 3 is the main contribution of this paper as it gives a detailed solution of the 4-bit integer factorization problem, using recently developed techniques for solving Boolean equations [10,13-27]. Section 4 concludes the paper.

2 A Quick Review of Pertinent Concepts

To make the paper self-contained, we briefly review some of the concepts and facts needed herein. The reader is also advised to consult the excellent texts by Rudeanu [14] and Brown [10]. More details are also available in [19-27]. Admittedly, the review included herein could be considered trivially warranted, but it hopefully saves the reader the trouble of collecting scattered (albeit well known) information. Moreover, this information is rendered more comprehensible via pictorial visualization.

2.1 Postulates of a Boolean Algebra

A *Boolean algebra* is a quintuple $B = (B, \vee, \wedge, 0, 1)$ in which B is a set, called the carrier; $\vee$ and $\wedge$ are binary operations on B, and the zero (0) and unit (1) elements are *distinct* members of B (that are not necessarily the only members of B), with certain postulates on commutativity, distributivity, identities and complementation being satisfied. These postulates are given herein as dual pairs.

1. Commutative Laws. For all a, b in B,

$$\boldsymbol{a} \vee \boldsymbol{b} = \boldsymbol{b} \vee \boldsymbol{a} \qquad \boldsymbol{a}\wedge\boldsymbol{b} = \boldsymbol{b}\wedge\boldsymbol{a}$$

2. Distributive Laws. For all a, b, c in B,

$$\boldsymbol{a}\vee(\boldsymbol{b}\wedge\boldsymbol{c}) = (\boldsymbol{a}\vee\boldsymbol{b})\wedge(\boldsymbol{a}\vee\boldsymbol{c}) \qquad \boldsymbol{a}\wedge(\boldsymbol{b}\vee\boldsymbol{c}) = (\boldsymbol{a}\wedge\boldsymbol{b}) \vee (\boldsymbol{a}\wedge\boldsymbol{c})$$

3. Identities. For all a in B,

$$\mathbf{0} \vee \boldsymbol{a} = \boldsymbol{a} \qquad \mathbf{1} \wedge \boldsymbol{a} = \boldsymbol{a}$$

4. Complements. To any element a in B there corresponds a unique element $\bar{\boldsymbol{a}}$ in B such that

$$\boldsymbol{a} \vee \bar{\boldsymbol{a}} = \mathbf{1} \qquad \boldsymbol{a} \wedge \bar{\boldsymbol{a}} = \mathbf{0}$$

2.2 Facts about Boolean Algebras:

1. Every element X of B has a *unique complement* $\overline{X}$.
2. There is a *partial-order* or *inclusion* ($\leq$) relation on B that is *reflexive*, *anti-symmetric*, and *transitive*.
   (a) reflexive: $\boldsymbol{a} \leq \boldsymbol{a}$
   (b) anti-symmetric: $\{\boldsymbol{a} \leq \boldsymbol{b}\, ,\ \boldsymbol{b} \leq \boldsymbol{a}\}$ ==> $\boldsymbol{a} = \boldsymbol{b}$
   (c) transitive: $\{\boldsymbol{a} \leq \boldsymbol{b}\, ,\ \boldsymbol{b} \leq \boldsymbol{c}\}$ ==> $\{\, \boldsymbol{a} \leq \boldsymbol{c}\, \}$
3. A Boolean algebra B enjoys many useful properties such as *associativity*, *idempotency*, *constants*, *absorption*, *involution*, *de Morgan's, reflection*, *consensus*, *inclusion and duality*. These properties might be detailed as follows:

Property 1 (Associativity):

$$\boldsymbol{a} \vee (\boldsymbol{b}\vee\mathbf{c}) = (\boldsymbol{a}\vee\mathbf{b})\vee \mathbf{c} \qquad \boldsymbol{a}\wedge(\boldsymbol{b}\wedge\mathbf{c}) = (\boldsymbol{a}\wedge\mathbf{b})\wedge\mathbf{c}$$

Property 2 (Idempotency):

$$\boldsymbol{a} \vee \boldsymbol{a} = \boldsymbol{a} \qquad \boldsymbol{a}\wedge\boldsymbol{a} = \boldsymbol{a}$$

Property 3 (Constants):

$$\boldsymbol{a} \vee \mathbf{1} = \mathbf{1} \qquad\qquad \boldsymbol{a} \wedge \mathbf{0} = \mathbf{0}$$

Property 4 (Absorption):

$$\boldsymbol{a} \vee (\boldsymbol{a} \wedge \boldsymbol{b}) = \boldsymbol{a} \qquad\qquad \boldsymbol{a} \wedge (\boldsymbol{a} \vee \boldsymbol{b}) = \boldsymbol{a}$$

Property 5 (Involution):

$$\overline{(\overline{\boldsymbol{a}})} = \boldsymbol{a}$$

Property 6 (De Morgan's Laws):

$$\overline{(\boldsymbol{a} \vee \boldsymbol{b})} = \overline{\boldsymbol{a}} \wedge \overline{\boldsymbol{b}} \qquad\qquad \overline{(\boldsymbol{a} \wedge \boldsymbol{b})} = \overline{\boldsymbol{a}} \vee \overline{\boldsymbol{b}}$$

Property 7 (Reflection):

$$\boldsymbol{a} \vee (\overline{\boldsymbol{a}} \wedge \boldsymbol{b}) = \boldsymbol{a} \vee \boldsymbol{b} \qquad\qquad \boldsymbol{a} \wedge (\overline{\boldsymbol{a}} \vee \boldsymbol{b}) = \boldsymbol{a} \wedge \boldsymbol{b}$$

Property 8 (Consensus):

$$(\boldsymbol{a} \wedge \boldsymbol{b}) \vee (\overline{\boldsymbol{a}} \wedge \boldsymbol{c}) \vee (\boldsymbol{b} \wedge \boldsymbol{c}) = (\boldsymbol{a} \wedge \boldsymbol{b}) \vee (\overline{\boldsymbol{a}} \wedge \boldsymbol{c}) \qquad (\boldsymbol{a} \vee \boldsymbol{b}) \wedge (\overline{\boldsymbol{a}} \vee \boldsymbol{c}) \wedge (\boldsymbol{b} \vee \boldsymbol{c}) = (\boldsymbol{a} \vee \boldsymbol{b}) \wedge (\overline{\boldsymbol{a}} \vee \boldsymbol{c})$$

Property 9 (Inclusion):

$$\boldsymbol{a} \leq \boldsymbol{a} \vee \boldsymbol{b} \qquad\qquad \boldsymbol{a} \wedge \boldsymbol{b} \leq \boldsymbol{a}$$

Property 10 (The principle of duality):

Every identity deducible from the postulates of a Boolean algebra is transformed into another identity if (i) the operations ∨ and ∧ (ii) the left and right members of inclusions, and (iii) the identity-elements 0 and 1 are interchanged throughout. The postulates themselves, together with the foregoing properties, provide good examples of the duality-principle. Because of that principle, only one of each of the statement-pairs above need be established; the other member of the pair follows by duality.

4. A Boolean algebra B is a *complemented distributive lattice* whose 0 and 1 values are *distinct.(Therefore B_1 does not exist in our analysis).*
5. A nonzero element Z of B is said to be an *atom* of B if and only if for every $X \in B$, the condition $X \leq Z$ implies that $X = Z$ or $X = 0$.
6. Every *finite* Boolean algebra B is *atomic*, i.e. for every nonzero element $X \in B$, there is some atom Z such that $Z \leq X$. This viewpoint rejects the case $\{0 = 1\}$ as a contradiction, and ignores the possibility of an *atomless* algebra B_1 in which $\{0 = 1\}$ is accepted!
7. Examples of Boolean algebras include the algebra of classes (subsets of a set), the algebra of propositional functions, the arithmetic Boolean algebra, the switching or two-element Boolean algebra, as well as big Boolean algebras, [10].
8. Boolean algebras with the same number of elements are *isomorphic*.
9. Every finite Boolean algebra B has 2^m elements, where m is the cardinality of (number of elements in) the set of atoms of B. We distinguish Boolean algebras larger than the two-valued one (the switching algebra B_2, m =1) by naming them big Boolean algebras.
10. A Boolean function f: $B^n \rightarrow B$,(with a domain B^n and range B) where B is a carrier of 2^m elements, is uniquely determined by a truth table or a Karnaugh map partially representing f for the *restricted domain* $\{0, 1\}^n$ which is a *strict* subset of the complete domain B^n.
11. The elements of B are named in terms of a *minimum* number of *abstract* variables or generators $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_k)$, with the elements of B taken as the elements of the *free Boolean algebra* $FB(\mathbf{Y}) = FB(\mathrm{Y}_1, \mathrm{Y}_2, \ldots, \mathrm{Y}_k)$ which is isomorphic to the Boolean algebra of switching functions of k variables, and possesses 2^{2^k} elements. The smallest big Boolean algebra B_4 has a single generator a, two atoms

$\bar{a}$ and a, and 4 partially-ordered elements ($0 \leq \{\bar{a}, a\} \leq 1$) that are the 4 switching functions of one variable. A 4-dimensional hypercube lattice can be used to visualize the big Boolean algebra B_{16} which has two generators a and b, four atoms $\bar{a}\bar{b}, \bar{a}b, a\bar{b}$ and ab, and 16 partially-ordered elements that are the 16 switching functions of 2 variables. A cubic lattice represents the big Boolean algebra B_8 which still has two generators a and b, but only three atoms (say $\bar{a}\bar{b}$, $\bar{a}b$, and $a\bar{b}$), and 8 partially-ordered elements. Note that B_8 can be obtained from B_{16} by nullifying, one of its atoms.

Fig. 2 demonstrates many of the above postulates and facts by displaying the lowest order finite Boolean algebras as complemented distributive lattices or hypercubes (occasionally hypocubes! or simply cubes) [21].

2.3 Big Boolean Algebras are Unavoidable

Big Boolean algebras cannot be avoided [10]. The use of big Boolean algebras in the analysis and design of switching systems is unavoidable, even if unrecognized, at least when using algebraic methods [10,28]. We will see shortly that to solve $\mathrm{R}(\mathbf{X}, \mathbf{Z}) = 1$ for $\mathbf{Z}$ as a function of $\mathbf{X}$ necessitates the use of big Boolean algebra $\mathrm{FB}(\mathbf{X})$.

2.4 Differences between Big Boolean Algebras and the Two-Valued One

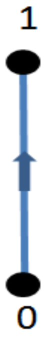
The two-valued Boolean algebra has properties not shared by big Boolean algebras. For example, Brown [10] points out that the following conclusions

$$\{\, xy = 0 \,\} ==> \{\, Either\ x = 0\ or\ y = 0 \,\}$$
$$\{\, x \vee y = 1 \,\} ==> \{\, Either\ x = 1\ or\ y = 1 \,\}$$

are valid only in B_2. However, in $B_4 = \{0, 1, \propto, \bar{\propto}\}$ $xy = 0$ or $x \vee y = 1$ could be satisfied by $x = \propto, y = \bar{\propto}$ and it is not necessary for x to be 0 .

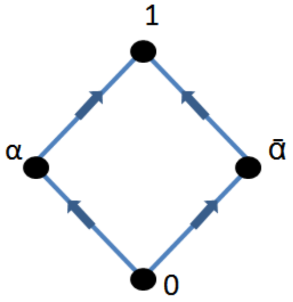
0=1 (contradiction)

The Boolean algebra B_1 (which is not allowed herein)



0 ≤ 1

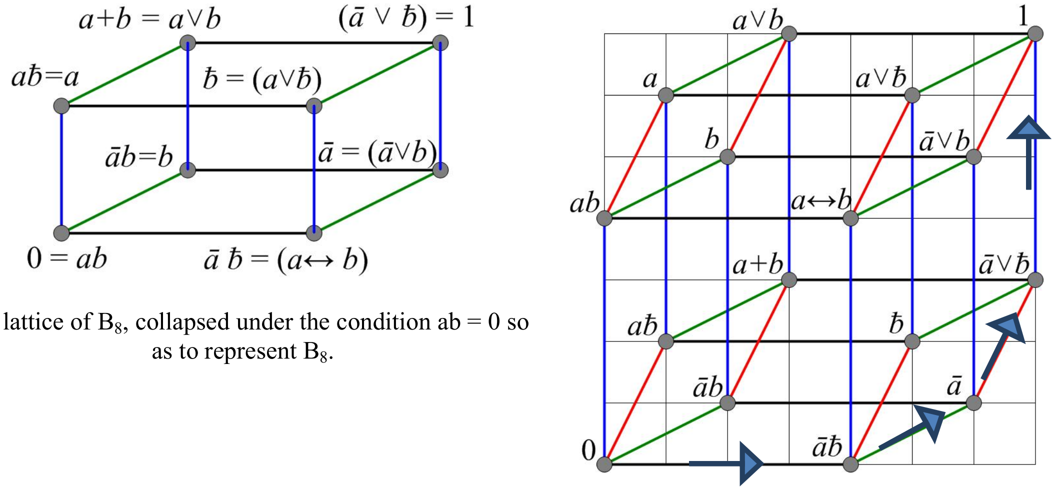
The lattice of B_2



The lattice of B_4

$0 \leq \begin{Bmatrix} \propto \\ \bar{\propto} \end{Bmatrix} \leq 1$

$\propto$, $\bar{\propto}$ not comparable



The lattice of B_8, collapsed under the condition ab = 0 so as to represent B_8.

A hypercube lattice indicating the partial ordering among the 16 elements of B_{16}.(for example $0 \le \bar{a}\bar{b} \le \bar{a} \le \bar{a}\vee\bar{b} \le 1$)

Fig. 2. Visualization of the lattice structure of the few lowest-order Boolean algebras, including the atomless B_1 (rejected herein), the switching algebra B_2 and the 'big' Boolean algebras B_4, B_8 and B_{16}.

3 Digital Design for an Integer-Factorization Circuit

The multiplication of an n-bit integer **Y** and an m-bit integer **Z** produces an integer **X** of $(n+m)$ bits. We consider the inverse of this operation which is the factorization of an integer **X** of $2n$ bits into two factors **Y** and **Z**. To avoid factoring **X** into a product of itself with 1, we impose the restrictions $(\mathbf{Y} > 1)$ and $(\mathbf{Z} > 1)$. To avoid duplicate factorizations due to commutativity $(\mathbf{Y} * \mathbf{Z} = \mathbf{Z} * \mathbf{Y})$, we impose the additional restriction $(\mathbf{Y} \ge \mathbf{Z})$. Since **Z** can be as small as 2, the integer **Y** can be as large as $(\mathbf{X}/2)$, and hence might occupy up to $(2n-1)$ bits. Since $(\mathbf{Y} \ge \mathbf{Z})$, the number **X** should satisfy $(\mathbf{X} \ge \mathbf{Z}^2)$, and hence **Z** might occupy up to n bits. The sizes of the integers **X**, **Y**, and **Z** are therefore $2n$, $(2n-1)$, and n bits, respectively.

The value $n = 1$ is not admissible since the smallest **Z** $(i.e., 2)$ requires 2 bits. For illustrative purposes, we demonstrate herein the smallest possible problem for which $n = 2$ so that the triple $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is of sizes $(4, 3, 2)$ bits. This problem can be initially illustrated by four 5-variable Karnaugh maps, but they will be grouped together as a single multi-entered Karnaugh map.

With a problem of the above sizes, we need a multiplication table of inputs $\mathbf{Y} \le 7$ and $\mathbf{Z} \le 3$ that produces a product up to $\mathbf{X} \le 21$. This multiplication table is rendered a Karnaugh-map representation in Fig. 3, with the decimal values of the inputs $\mathbf{Y}, \mathbf{Z}$ and output **X** highlighted in red. The map also translates these decimal values into binary values distinguished in black. Since the binary values of **X** are given in 4 bits, the map in Fig. 3 is a multi-entered map and is equivalent to four different maps for the binary variables X_3, X_2, X_1, and X_0 of course, the representation of $\mathbf{X} > 15$ in 4 bits fail.

Fig. 4(a) translates Fig. 3 to an initial specification of the problem in the form of an equation

$$g_0(\mathbf{Y}, \mathbf{Z}) = g_0(Y_2, Y_1, Y_0, Z_1, Z_0) = 1 \qquad (2)$$

with the function $g_0 \colon B^5 \to B$ constructed over the 'big' Boolean algebra $B = FB(\mathbf{X})$, *i.e.,* it is the free Boolean algebra with the four generators X_3, X_2, X_1, and X_0. This Boolean algebra is of $2^4 = 16$ atoms and

2^{16} = 65536 elements. The function g_0 is characterized by discriminants or Karnaugh map entries given for a specific value of **Y** and **Z** by

$$g_0(\mathbf{X}, \mathbf{Z}) = \wedge_{i=1}(\mathrm{X}_i \odot \mathrm{X}_i(\mathbf{Y}, \mathbf{Z})) \qquad (3)$$

where

$$\mathrm{X}_i \odot \mathrm{X}_i(\mathbf{Y}, \mathbf{Z}) = \mathrm{X}_i^{\mathrm{X}_i(\mathbf{Y},\mathbf{Z})} \qquad (4)$$

is equal to X_i (uncomplemented) if $\mathrm{X}_i(\mathbf{Y}, \mathbf{Z}) = 1$ and equals $\overline{\mathrm{X}_i}$ (complemented) if $\mathrm{X}_i(\mathbf{Y}, \mathbf{Z}) = 0$. Here, we allow a (hopefully forgivable) abuse of notation by using the same symbol X_i to denote a certain variable *per se*, and also to denote a particular constant value $\mathrm{X}_i(\mathbf{Y}, \mathbf{Z})$ specified as an entry (0 or 1) in a particular cell $(\mathbf{Y}, \mathbf{Z})$ in the map of Fig. 3. Based on the above discussion, we obtain the Karnaugh map for g_0 in Fig. 4(a). To complete the problem specifications, we need to replace g_0 by g given by

$$g(\mathbf{Y}, \mathbf{Z}) = g_0(\mathbf{Y}, \mathbf{Z})\ \mathrm{I}(\mathbf{Y} > 1)\ \mathrm{I}(\mathbf{Z} > 1)\ \mathrm{I}(\mathbf{Y} \geq \mathbf{Z})\ \mathrm{I}(\mathbf{X} \leq 15), \qquad (5)$$

where the symbol $\mathrm{I}(\text{event})$ is a Boolean indicator for that event, i.e., it is 1 if the event occurs and 0 if it does not occur. We already discussed the necessity for the requirements $(\mathbf{Y} > 1)$, $(\mathbf{Z} > 1)$ and $(\mathbf{Y} \geq \mathbf{Z})$. The extra condition $\mathrm{I}(\mathbf{X} \leq 15)$ is needed to ensure that X is properly represented in 4-bits (as indicated earlier, Fig. 3 misrepresents the integers 18 and 21). It is straightforward to note that

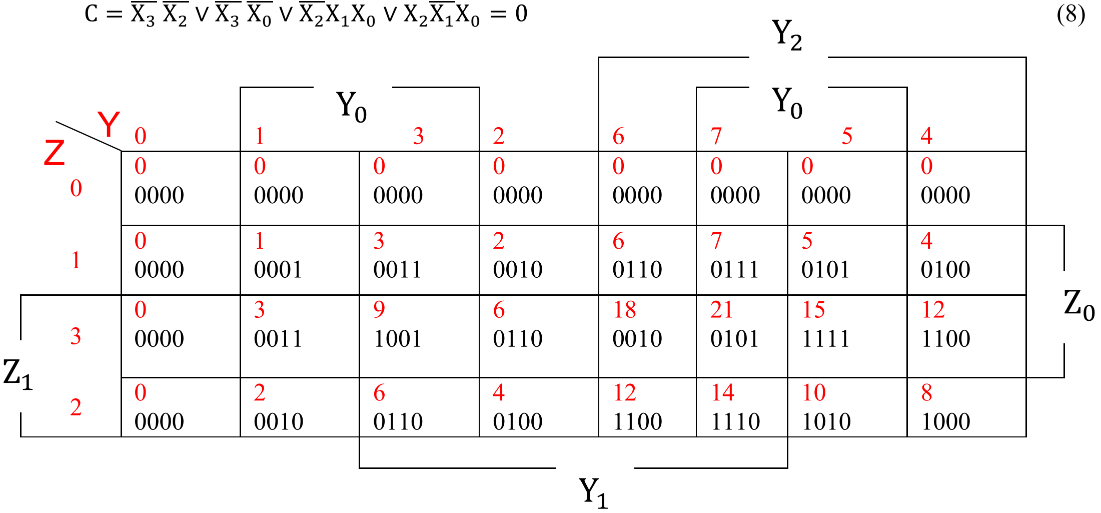
$$\mathrm{I}(\mathbf{Z} > 1)\ \mathrm{I}(\mathbf{Y} \geq \mathbf{Z}) \Rightarrow \mathrm{I}(\mathbf{Y} > 1) \qquad (6)$$

and hence equation (5) is simplified to

$$g(\mathbf{Y}, \mathbf{Z}) = g_0(\mathbf{Y}, \mathbf{Z})\ \mathrm{I}(\mathbf{Z} > 1)\ \mathrm{I}(\mathbf{Y} \geq \mathbf{Z})\ \mathrm{I}(\mathbf{X} \leq 15), \qquad (7)$$

The remaining parts of Fig. 4 explain the evolution of the map for g in $(\mathbf{Y}, \mathbf{Z})$ once the function $g(\mathbf{Y}, \mathbf{Z})$ in (7) is obtained. It is straightforward to solve it via recently developed techniques for solving Boolean equations (see, e.g., [21, 23-26]. First we construct the auxiliary function $\mathrm{G}(\mathbf{Y}, \mathbf{Z}, \mathbf{P})$ in Fig. 4(f) and identify the atoms not asserted in Fig. 4(f) in Fig. 5 to be nullified as the consistency condition

$$\mathrm{C} = \overline{\mathrm{X}_3}\,\overline{\mathrm{X}_2} \vee \overline{\mathrm{X}_3}\,\overline{\mathrm{X}_0} \vee \overline{\mathrm{X}_2}\mathrm{X}_1\mathrm{X}_0 \vee \mathrm{X}_2\overline{\mathrm{X}_1}\mathrm{X}_0 = 0 \qquad (8)$$



Z \ Y	0	1	3	2	6	7	5	4
0	0 0000	0 0000	0 0000	0 0000	0 0000	0 0000	0 0000	0 0000
1	0 0000	1 0001	3 0011	2 0010	6 0110	7 0111	5 0101	4 0100
3	0 0000	3 0011	9 1001	6 0110	18 0010	21 0101	15 1111	12 1100
2	0 0000	2 0010	6 0110	4 0100	12 1100	14 1110	10 1010	8 1000

$\mathrm{X}_3(\mathbf{Y}, \mathbf{Z})\ \mathrm{X}_2(\mathbf{Y}, \mathbf{Z})\ \mathrm{X}_1(\mathbf{Y}, \mathbf{Z})\ \mathrm{X}_0(\mathbf{Y}, \mathbf{Z})$

Fig. 3. Karnaugh-map representation for $\mathrm{X} = (\mathrm{X}_3\ \mathrm{X}_2\ \mathrm{X}_1\ \mathrm{X}_0)_2$ as a product $\mathrm{Y} * \mathrm{Z} = (Y_2Y_1Y_0) * (Z_1Z_0)$. Both inputs Y and Z and output *X* are expressed in decimal notation (red font) and in equivalent binary notation (black font). For the binary notation the map is a multi-entered map, and is equivalent to four (single-entered) maps.

Y_2

Y_0 Y_0

$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$
$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}x_0$	$\overline{x_3}\,\overline{x_2}x_1x_0$	$\overline{x_3}\,\overline{x_2}x_1\overline{x_0}$	$\overline{x_3}x_2x_1\overline{x_0}$	$\overline{x_3}x_2x_1x_0$	$\overline{x_3}x_2\overline{x_1}x_0$	$\overline{x_3}x_2\overline{x_1}\,\overline{x_0}$
$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}x_1x_0$	$x_3\overline{x_2}\,\overline{x_1}x_0$	$\overline{x_3}x_2x_1\overline{x_0}$	$\overline{x_3}\,\overline{x_2}x_1\overline{x_0}$	$\overline{x_3}x_2\overline{x_1}x_0$	$x_3x_2x_1x_0$	$x_3x_2\overline{x_1}\,\overline{x_0}$
$\overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0}$	$\overline{x_3}\,\overline{x_2}x_1\overline{x_0}$	$\overline{x_3}x_2x_1\overline{x_0}$	$\overline{x_3}x_2\overline{x_1}\,\overline{x_0}$	$x_3x_2\overline{x_1}\,\overline{x_0}$	$x_3x_2x_1\overline{x_0}$	$x_3\overline{x_2}x_1\overline{x_0}$	$x_3\overline{x_2}\,\overline{x_1}\,\overline{x_0}$

Z_0 Z_1

Y_1

$(a)\ g_0(Y_2, Y_1, Y_0, Z_1, Z_0)$

Y_2

Y_0 Y_0

Z								
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1

Z_0 Z_1

Y_1

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

Y_2

Y_0 Y_0

Z \ Y	0	1	3	2	6	7	5	4
0	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1
3	0	0	1	0	1	1	1	1
2	0	0	1	1	1	1	1	1

Z_0 Z_1

Y_1

$(\boldsymbol{c})\ \mathbf{I}(\mathbf{Y} \geq \mathbf{Z})$

Y_2

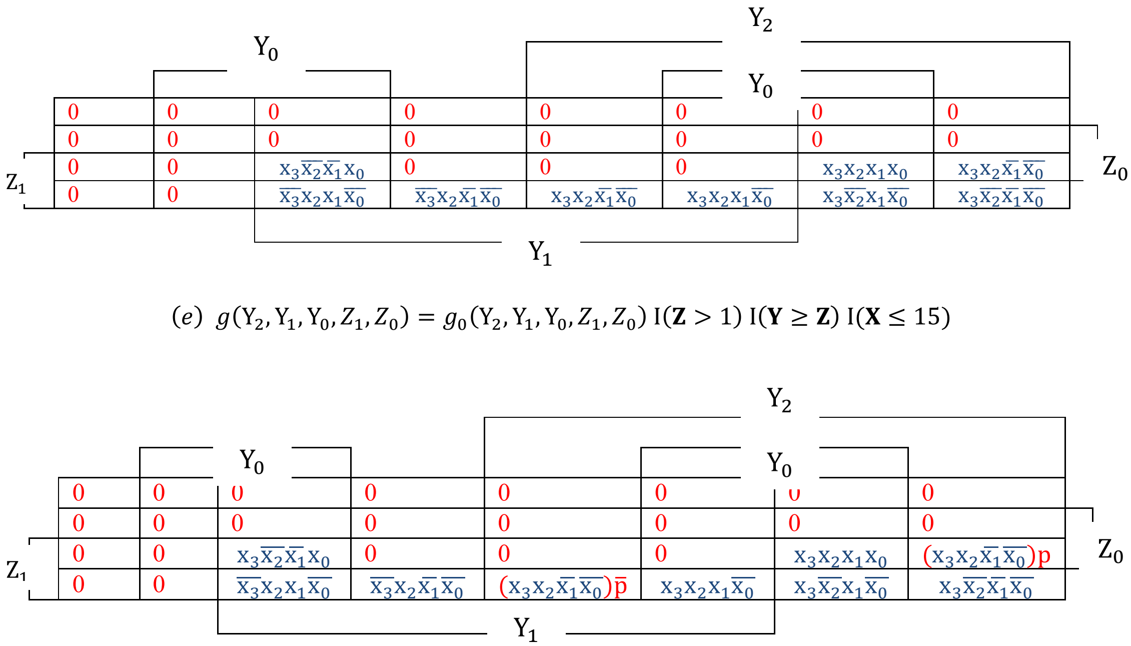
Y_0 Y_0

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

Z_0 Z_1

Y_1

$(d)\ \mathrm{I}(\mathbf{X} \leq 15)$



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	$X_3\overline{X_2}\overline{X_1}X_0$	0	0	0	$X_3X_2X_1X_0$	$X_3X_2\overline{X_1}\overline{X_0}$
0	0	$\overline{X_3}X_2X_1\overline{X_0}$	$\overline{X_3}X_2\overline{X_1}\overline{X_0}$	$X_3X_2\overline{X_1}\overline{X_0}$	$X_3X_2X_1\overline{X_0}$	$X_3\overline{X_2}X_1\overline{X_0}$	$X_3\overline{X_2}\overline{X_1}\overline{X_0}$

$(e)\ g(Y_2, Y_1, Y_0, Z_1, Z_0) = g_0(Y_2, Y_1, Y_0, Z_1, Z_0)\ I(\mathbf{Z} > 1)\ I(\mathbf{Y} \geq \mathbf{Z})\ I(\mathbf{X} \leq 15)$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	$X_3\overline{X_2}\overline{X_1}X_0$	0	0	0	$X_3X_2X_1X_0$	$(X_3X_2\overline{X_1}\overline{X_0})p$
0	0	$\overline{X_3}X_2X_1\overline{X_0}$	$\overline{X_3}X_2\overline{X_1}\overline{X_0}$	$(X_3X_2\overline{X_1}\overline{X_0})\overline{p}$	$X_3X_2X_1\overline{X_0}$	$X_3\overline{X_2}X_1\overline{X_0}$	$X_3\overline{X_2}\overline{X_1}\overline{X_0}$

$(f)\ G(X_2, X_1, X_0, Y_1, Y_0, p)$

Fig. 4. Evaluation of a function g equated to 1 that represents all problem specifications, and subsequent evaluation of the corresponding auxiliary function G.

Subsequently, the desired solution is read from the map of $G(\mathbf{X}, \mathbf{Y}, \mathbf{p})$ as [19, 24]

$$Y_2 = X_3\overline{X_0} \vee X_3X_2X_1 \vee d(C) \tag{9a}$$

$$Y_1 = X_2X_1\overline{X_0} \vee \overline{X_3}X_2\overline{X_0} \vee X_3\overline{X_2}\,\overline{X_1}X_0 \vee X_3X_2\overline{X_1}\,\overline{X_0}\,\overline{p} \vee d(C) \tag{9b}$$

$$Y_0 = X_3X_1\overline{X_0} \vee X_3X_2X_1 \vee X_2X_1\overline{X_0} \vee X_3\overline{X_2}\,\overline{X_1}X_0 \vee d(C) \tag{9c}$$

$$Z_1 = X_3\overline{X_0} \vee X_2\overline{X_0} \vee X_3X_2X_1 \vee X_3\overline{X_2}\,\overline{X_1}X_0 \vee d(C) \tag{9d}$$

$$Z_0 = X_3\overline{X_2}\,\overline{X_1}X_0 \vee X_3X_2X_1X_0 \vee X_3X_2\overline{X_1}\,\overline{X_0}\,p \vee d(C) \tag{9e}$$

Equations (9) constitute a faithful (albeit incompletely specified) solution. They can be used to produce a completely-specified solution of desired features (such as compactness). The single parameter p in (9) can be either considered belonging to the underlying Boolean algebra or to the two-valued Boolean algebra [19, 23, 24]. Therefore, the parametric solution (9) is equivalent to two particular solutions. Fig. 6 displays these two particular solutions in compact form. These two solutions are in agreement with Fig. 5. In fact, the particular solutions can be directly deduced from Fig. 5 for our current toy problem. However, extensions to Fig. 5 cannot be used to construct general parametric solutions for lager problems.

If we have as input $(X)_{10} = 13$, i.e. , $X_3X_2X_1X_0 = 1101$, our circuit will check to find that $X_2\overline{X_1}X_0 = 1$ and hence $C = 1$. This means that the consistency condition is not satisfied (i.e., it is a contradiction $1 = 0$) which indicates that 13 is a prime number that cannot be factored (other than to a product of 1 and itself). When C is found to be 1, the circuit refrains from reporting values for **Y** and **Z** (since such values are meaningless).

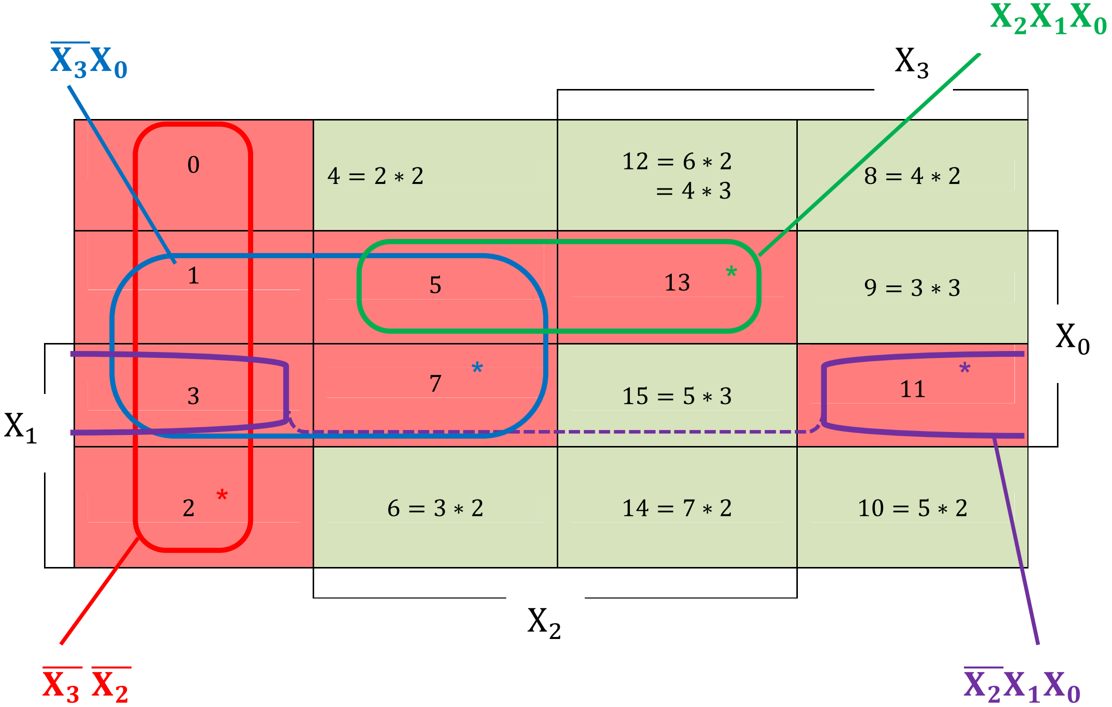
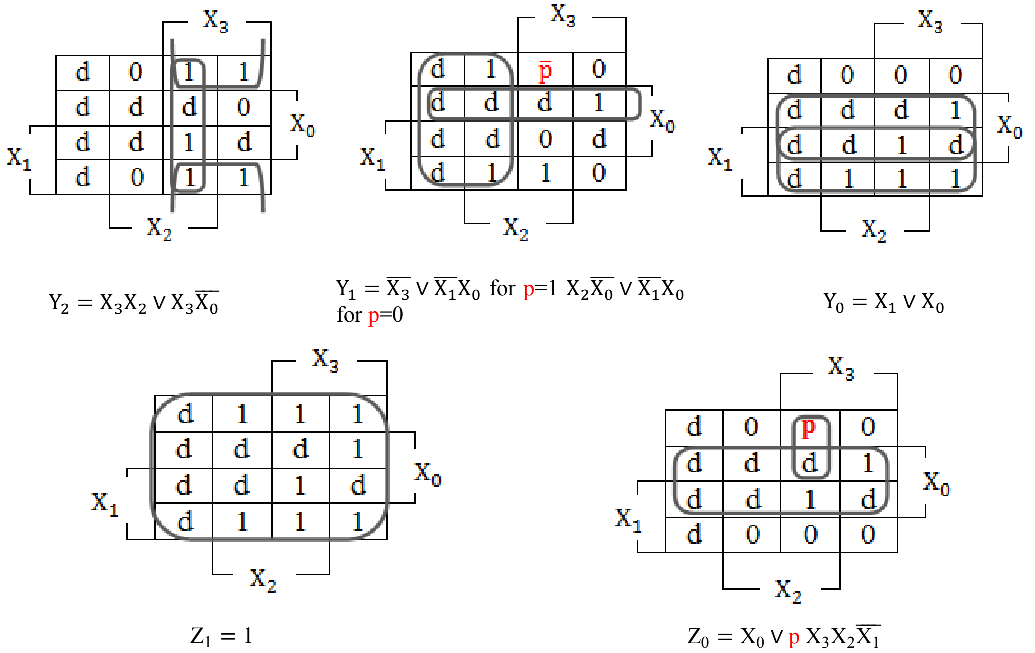


Fig. 5. Identification of atoms not asserted in Fig. 4(f) for nullification as a consistency condition. Cells representing composite numbers (4, 6, 8, 10, 12, 14, 15) are painted green, while cells depicting 0 or 1 together with the prime number (2, 3, 5, 7, 11, 13) are colored red. These red cells have don't cares n the maps of Fig. 6.



$Y_2 = X_3X_2 \vee X_3\overline{X_0}$

$Y_1 = \overline{X_3} \vee \overline{X_1}X_0$ for p=1 $X_2\overline{X_0} \vee \overline{X_1}X_0$ for p=0

$Y_0 = X_1 \vee X_0$

$Z_1 = 1$

$Z_0 = X_0 \vee p\ X_3X_2\overline{X_1}$

Fig. 6. The two particular solutions for the factorization problem (in agreement with Fig. 5)

4 Conclusions

This paper explores a novel technique for hardware implementation of the task of integer factorization. Such an implementation has the promise of achieving this hard task for reasonably large bit sizes in real time. The technique offered herein is an alternative for a technique already in use that employs predicate logic to achieve Boolean-function synthesis. By contrast, our technique employs Boolean-equation solving over ‘big’ Boolean algebra, *i.e.*, an algebra larger than the two-valued Boolean algebra. Though the example solved herein is only a 4-bit toy problem, it suffices to demonstrate the success of the technique and to set the stage for its automated implementation.

Our paper is the first exploration of the use of Boolean-solving techniques in integer factorization. This paper must be supplemented with a study of the scaling issue (whether the used technique can handle large-size problems) and the complexity issue (how much time and memory are needed for such large-size problems). A sequel forthcoming paper will treat the scaling, complexity, and automation issues, and will, in particular, determine the upper limit on the bit size that can be treated by the current technique. We reiterate that the problem of integer factorization is definitely a hard intractable problem, and that its best solvers using Boolean function synthesis is known to have handled only up to 12 bits. Our forthcoming work is required to decide whether our novel technique can surpass this bit size or not.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Muroga S. Logic design and Switching theory, Wiley, New York, NY, USA; 1979.

[2] Gregg JR. Ones and Zeros: Understanding Boolean algebra, Digital Circuits, and the Logic of Sets, IEEE PRESS, New York, NY, USA; 1998.

[3] Crama Y, Hammer PL. Boolean Functions: Theory, Algorithms, and Applications, Cambridge University Press, Cambridge, United Kingdom; 2011.

[4] Brown SD, Vranesic Z. Fundamentals of digital logic with Verilog Design, 3rd Ed., McGraw-Hill, New York, NY, USA; 2014.

[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. 2016;402-421. Springer International Publishing.

[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, October 2-6; 2017.

[9] Couturat L. L'algebre de la Logique. Paris: Scientia, 1905. English translation (by Lydia G. Robinson): Open Court Pub. Co., Chicago & London; 1914.

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

[11] 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.

[12] 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)

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

[14] Rudeanu S. Boolean Functions and Equations, North-Holland Publishing Company & American Elsevier, Amsterdam, the Netherlands; 1974.

[15] Rushdi AM. Using variable-entered Karnaugh maps to solve Boolean equations. International Journal of Computer Mathematics. 2001;78(1):23-38.

[16] Rudeanu S. Algebraic methods versus map methods of solving Boolean equations. International Journal of Computer Mathematics. 2003;80(7):815-817.

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

[18] Baneres D, Cortadella J, Kishinevsky M. A recursive paradigm to solve Boolean relations. IEEE Transactions on Computers. 2009;58(4):512-527.

[19] 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.

[20] 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.

[21] 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.

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

[23] Rushdi AMA, Al-Qwasmi. 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):51-64.

[24] 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.

[25] Rushdi AMA, Ahmad W. Satisfiability in Big Boolean algebras via Boolean-equation solving. Journal of King Abdulaziz University: Engineering Sciences. 2017;28(1).

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

[27] Rushdi AMA, Ahmad W. Digital circuit design utilizing equation solving over 'big' Boolean algebras. International Journal of Mathematical, Engineering and Management Sciences. 2018;3(3).

[28] 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(3).

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/23038