Review of Computer Engineering Research
2020 Vol. 7, No. 1, pp. 27-37
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/journal.76.2020.71.27.37
© 2020 Conscientia Beam. All Rights Reserved.

INTERACTIVE ALGORITHMS FOR THE VERIFICATION OF THE EQUALITY BETWEEN COMPLEX AND SIMPLIFIED BOOLEAN-ALGEBRA EXPRESSIONS IN DIGITAL DECODERS

Tochukwu Arinze Ikwunne[1+]
Samuel Obiora Okide[2]

[1]*Alex Ekweme Federal University Ndufu-Alike Ikwo, Nigeria.*
Email: Ikwunne.arinze@funai.edu.ng Tel: +2348038653851
[2]*Nnamdi Azikiwe University Awka, Nigeria.*
Email: so.okide@unizik.edu.ng Tel: +2348056520945

(+ Corresponding author)

Article History
Received: 14 January 2020
Revised: 18 February 2020
Accepted: 20 March 2020
Published: 13 April 2020

Keywords
Digital decoder
Boolean algebra expression
Seven-segmented display
Combinational logic gate
Verification
Algorithms
Applications.

ABSTRACT

This work highlights the use of an algorithm in evaluating and verifying a complex Boolean expression that are used in fabricating digital decoder systems. Digital decoders are built by human beings and, unfortunately, humans make mistakes. Both design errors and faulty implementation may lead both the hardware and software components of systems to behave in unexpected ways, which in turn may lead to business losses and even risky situations. Karnough maps, Boolean algebra theorems and laws are some of the techniques that can be used to simplify and reduce complex Boolean algebra expressions and truth table can be used to confirm that the reduced Boolean algebra expression is the same as the original, complex Boolean algebra expression. However, generating the truth table manually is tedious, especially when the Boolean algebra equation or expression has many Boolean variables. Therefore, this work presents novel algorithms to verify and evaluate complex Boolean expression from the fabricated decoder circuit.

Contribution/Originality: This study documents the importance of using an algorithmic approach in checking and evaluating the equality of a complex Boolean expression and its simplified form from the fabricated decoder circuit. This is to ensure that the hardware and software systems are designed with reduced or no errors.

1. INTRODUCTION

A Digital Decoder Integrated Circuit (DDIC) is a device that converts a digital format into another. A typical example of the DDIC device is the conversion of the Binary Coded Decimal (BCD) to Seven-Segment Display Decoder. Seven-segment Light Emitting Diode (LED) type provides a very convenient way of displaying information in numerical form, letters or even alphanumerical characters. 74LS47 decoder can produce the required numbers of HEX from 0 to 9 and display the correct combination of LED segments. Moreover, the seven-segment values for "0" through "9" has four inputs. The inputs are used to determines which of the segments on a seven-segment LED display should be on or off. The seven-segment display are shown in Figure 1. The seven-segment display in Figure 1 shows how to derive the Boolean expressions to build a driver circuit. It's application in digital systems are present in this modern society. An average person might use thousands of digital devices in each lifetime. Consumer electronics such as decoders, digital cameras, mobile phones, televisions, stereo equipment, and microwave ovens are examples of these devices. Moreover, hardware and software systems can control technological products such as missiles, airplanes, elevators, medical devices, ships, etc. These electronic devices are built and fabricated by human beings. The device hardware and software components are expressed using Boolean algebra expression. Digital software and hardware components of a system is one of the applications of Boolean algebra. A one-to-one correspondence exits between a digital circuits and Boolean expressions. Boolean expressions have a digital circuit design and vice versa. However, humans make mistakes that results in hardware and software systems misbehaviour, which in turn may lead to loss of business and even risky situations. Consequently, Karnough maps, Boolean algebra theorems and laws were applied in simplifying complex Boolean algebra expression. In order to determine the equality between a complex and simplified Boolean algebra, truth table are applied in confirming that the two Boolean algebra are the same. However, it is quite difficult to produce a truth table that has many variables. Example is shown in Table 1 that involves four input variables. Therefore, this work presents the use of novel algorithms in order to evaluate the equality between a simplified and complex Boolean algebra expression. It is necessary to state that the word, Boolean, was coined by George Boole. He is a mathematician that has done a classical work on logic. Boolean algebra contains a set of two possible values, two binary operators and one unary operator. The set satisfies five properties such as commutativity, associativity, distributivity, existence of identity and complement [1]. In Table 2, displays binary operators of Boolean algebra, AND, OR and one unary operator, NOT [2]. The evaluation of the equality between a complex and simplified Boolean expression is demonstrated in the design of the seven-segmented decoder display.

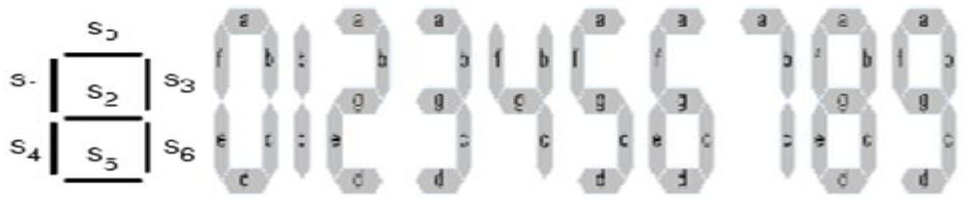

Figure-1. Seven Segment Display and Values for "0" Through "9".

Table-1. 7-segment display with 4-input truth table.

Digit	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Note: 1=ON, 0=OFF.

Where:

S_0 (=a) = A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D = A + C + BD + B'D',

S_1 (=f) = A'B'C'D' + A'BC'D' + A'BC'D + A'BCD' + AB'C'D' + AB'C'D = A + C'D' + BD' + BC',

S_2 (=g) = A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + AB'C'D'+ AB'C'D= A + BC' + B'C + CD',

S_3 (=b) = A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BCD + AB'C'D' + AB'C'D = B'+ C'D' + CD,

S_4 (=e) = A'B'C'D' + A'B'CD' + A'BCD' + AB'C'D' = B'D' + CD',

S_5 (=d) = A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD' + AB'C'D' + AB'C'D = A + CD' + BC'D + B'C + B'D',

S_6 (=c) = A'B'C'D' + A'B'C'D + A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D'+ AB'C'D = A + B + C'+ D.

Table-2. Binary operators of boolean algebra.

And	0	1	Or	0	1
0	0	0	0	0	1
1	0	1	1	1	1

Note: " ' ": '0 = 1, '1 = 0.

2. LITERATURE REVIEW

Boolean algebra is applied in and out of computer science domain. In the field of psychiatry, for example, Boolean algebra operations are linked to show how operations of nervous system work. Boolean algebra variants satisfy the properties of fuzzy sets [3]. Boolean algebra laws simplify both long and complex logic expressions [4]. Human errors are bound in the design and implementation of digital systems, the need to verify the equality between a complex Boolean algebra expression that have been simplified is very important. Therefore, it is required that formal verification is used in checking that a computerized system satisfies a specification that describes the correct behaviour of the system. This is to ensure that the hardware and software systems are designed with reduced or no errors. This important discipline of formal verification started since early 1980's and was referred to as model checking [5]. Model checking have been the essential part of the system development cycle in the design of Very Large Scale Integration (VLSI) circuits [6] and has been used to assist the design and implementation of telecommunication systems software [7]. Moreover, μ-calculus is one of the notable model-checking techniques. There are studies on μ-calculus with Boolean Equation Systems that uses model checking [8-15] but, the computational complexity of the μ-calculus model checking problem is dissonant, and no polynomial time algorithm has been discovered. Therefore, Mader [16] provides a broad study of the properties of Boolean equation systems. She shows how the μ-calculus model-checking problem are solved in terms of Boolean equation systems. In addition, Mader provided algebraic manipulations as a proof system for solving general Boolean equation systems problems. This led an initial steps to an iterative algorithm to solve general form Boolean equation systems called Gauß elimination. Groote and Willemse [17] show how a μ-calculus formula can be transformed into a defined parameter to define Boolean equation system. According to Groote and Willemse [18]; Groote and Willemse [19] various solution methods for parameterized Boolean equation systems were studied. Truth table were used to evaluate logical expressions in order to determine truth table results with an arithmetic version [20]. However, the computational complexity of evaluating the equality of a complex Boolean algebra and its simplified form is still dissonant. Thus, the need to develop further solution and methods that are efficient in practice. Therefore, this work presents a novel algorithm that do not only verifies Boolean algebra equations but evaluate Boolean algebra expressions used in digital decoders.

3. EXPERIMENTAL AND COMPUTATIONAL DETAILS

3.1. Correspondence between Simplified Blinking Circuits and Boolean Functions

LED (Light Emitting Diode) is a semiconductor light emitting diode. A small light is emitted when current can pass through LED, which gives impression to the users that circuit is active. The basic idea was driven from a common cathode 7-segment LED display using combinational logic circuit. The first aspect of this circuit is decoder. A decoder is a combinational circuit that converts a binary or BCD (Binary Coded Decimal) number to the corresponding 7-Segment Display. This combinational logic circuit is a system of logic gates consisting of outputs and inputs. The output depends only on the present state of the inputs. The logic circuit is designed with 4 (four) inputs and 7(seven) outputs, each representing an input to the display integrated circuit (IC) as shown in Figure 2 It is necessary to state that the design of the logic circuits requires a good knowledge of Boolean algebra and logic gates. The original complex Boolean algebra expressions of the seven-segment decoder are given in Figure 3. There are three methods of simplifying complex Boolean expression, these are a: algebraic, b: Karnough map and c: Quine-McCluskey. Karnough's map is used in this work. Using Karnough's map, logic circuitry for each input to the display with their simplified Boolean expressions are designed in Figure 4. There are lots of applications using LEDs. These applications can be found in highways to indicate traffic lights, signaling, and showing moving pictures in an advertisement board.

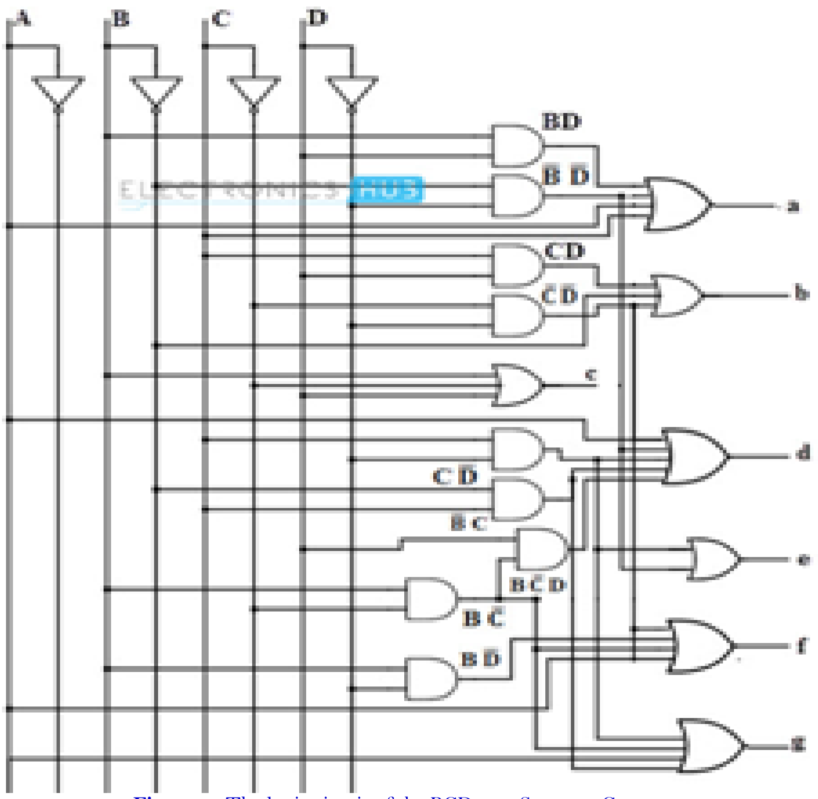

Figure-2. The logic circuit of the BCD to 7-Segment Converter.

$$S_0\,(=a) = A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D$$

$$S_1\,(=f) = A'B'C'D' + A'BC'D' + A'BC'D + A'BCD' + AB'C'D' + AB'C'D$$

$$S_2\,(=g) = A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + AB'C'D' + AB'C'D,$$

$$S_3\,(=b) = A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BCD + AB'C'D' + AB'C'D$$

$$S_4\,(=e) = A'B'C'D' + A'B'CD' + A'BCD' + AB'C'D'$$

$$S_5\,(=d) = A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD' + AB'C'D' + AB'C'D$$

$$S_6\,(=c) = A'B'C'D' + A'B'C'D + A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D$$

Figure-3. The seven-segment decoder complex Boolean algebra expressions.

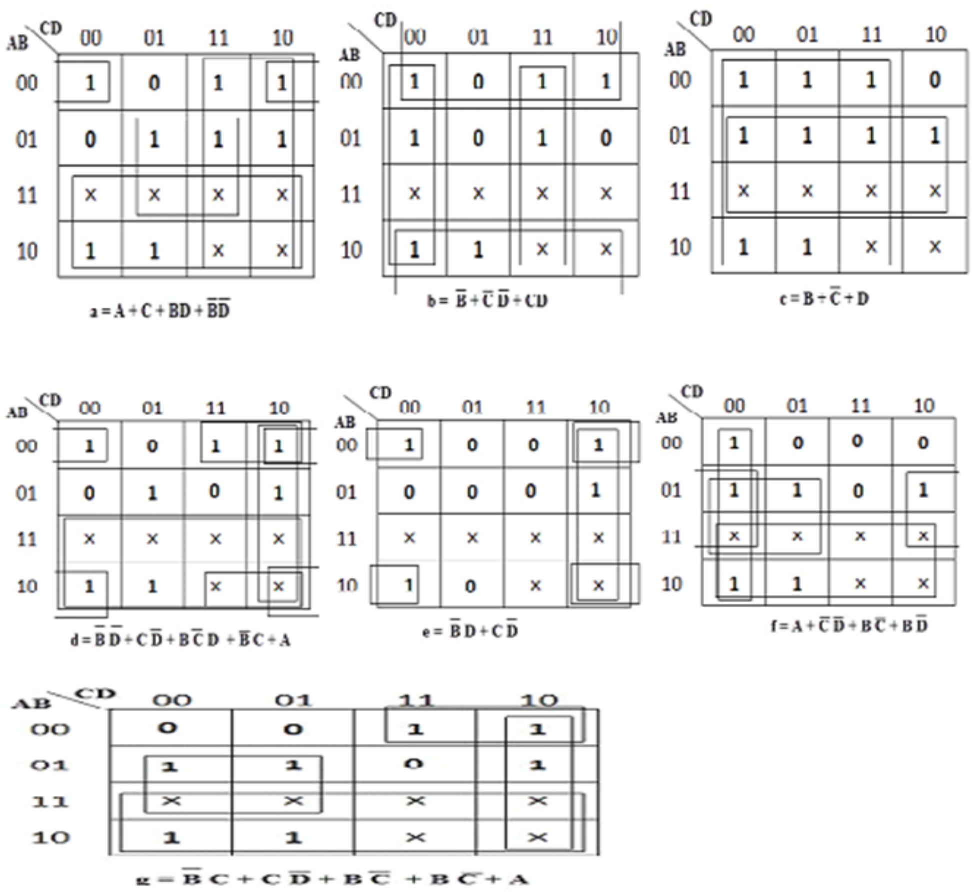
$a = A + C + BD + \bar{B}\bar{D}$
$b = \bar{B} + \bar{C}\bar{D} + CD$
$c = B + \bar{C} + D$
$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$
$e = \bar{B}\bar{D} + C\bar{D}$
$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$
$g = \bar{B}C + C\bar{D} + B\bar{C} + B\bar{C} + A$

Figure-4. Karnough's Map simplification of the 7 Segment Decoder.

4. RESULTS AND DISCUSSION

4.1. Algorithms for Evaluating Complex Boolean algebra Expression in 7-Segment Decoder

The complex Boolean expressions in the segment decoder and their simplified forms have been stated in Figure 3. The authors developed a novel algorithm for verifying these expressions. The essence of using the algorithms is because of the difficult nature of manually using truth tables that have many Boolean variables.

4.2. Expression

A set of logical values with Boolean algebra in the S_0 (=a) of the seven-segment display. Suppose A, B, C and D are the elements of the set S_0, algorithm establishes that complex Boolean algebra expression: A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D equals (=) the simplified Boolean algebra expression: A + C + BD + B'D' is follows below:

```
Boolean expression 4.2a (Boolean A, B, C,D)
1   Read in data
1.1 Captures A
1.2 Captures  B
1.3 Captures C
1.4 Captures D
2 Determine expression4.2a
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND NOT B AND C AND NOT D
2.3 thirdterm = NOT A AND NOT B AND C AND D
```

```
2.4 fourthterm = NOT A AND B AND NOT C AND D
2.5 fifthterm = NOT A AND B AND C AND NOT D
2.6 sixthterm = NOT A AND B AND C AND D
2.7 seventhterm = A AND NOT B AND NOT C AND NOT D
2.8 eighthterm = A AND NOT B AND NOT C AND D
2.9 left = firstterm OR secondterm OR thirdterm OR fourthterm OR fifthterm OR sixthterm OR sevethterm OR eightterm
2.10 right = A OR C OR (B AND D) OR (NOT B AND NOT D)
2.11 IF variables in the left = variables in the right THEN
2.11.1 result equals "The Expression Holds"
ELSE
2.11.2 result equals "The Expression does not Hold"
3. Display result
```

4.3. Expression

A set of logical values as Boolean algebra in S_1 (=f) of the seven-segment display. Suppose A, B, C and D are members of the set, the algorithm that can be used to verify the complex Boolean algebra expression: A'B'C'D' + A'BC'D' + A'BC'D + A'BCD' + AB'C'D' + AB'C'D is equal to (=) the simplified Boolean algebra expression: A + C'D' + BD' + BC' is follows below:

```
Boolean expression4.3f (Boolean A, B, C,D)
1 Read in  data
1.1 Captures A
1.2 Captures  B
1.3 Captures C
1.4 Captures  D
2 Determine expression4.3f
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND B AND NOT C AND NOT D
2.3 thirdterm = NOT A AND B AND NOT C AND D
2.4 fourthterm = NOT A AND B AND C AND NOT D
2.5 fifthterm = A AND NOT B AND NOT C AND NOT D
2.6 sixthterm = A AND NOT B AND NOT C AND D
2.7 left = firstterm OR secondterm OR thirdterm OR fourthterm OR fifthterm OR sixthterm
2.8 right = A OR (NOT C AND NOT D) OR (B AND NOT C)
2.9 IF variables in the left = variables in the right THEN
2.9.1 result equals "The Expression Holds"
ELSE
2.9.2 result equals "The Expression does not Hold"
3. Display result
```

4.4. Expression

A set of logical values as Boolean algebra in S_2 (=g) of the seven-segment display. Suppose A, B, C and D are members of the set, the algorithm that can be used to verify the complex Boolean algebra expression: A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + AB'C'D'+ AB'C'D is equal to (=) the simplified Boolean algebra expression: A + BC' + B'C + CD' is follows below:

Boolean expression 4.4g (Boolean A, B, C,D)

```
1 Read in  data
1.1 Captures A
1.2 Captures B
1.3 Captures C
1.4 Captures D
2 Determine expression4.4g
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND NOT B AND C AND D
2.3 thirdterm = NOT A AND B AND NOT C AND NOT D
2.4 fourthterm = NOT A AND B AND NOT C AND D
2.5 fifthterm = A AND NOT B AND NOT C AND NOT D
2.6 sixthterm = A AND NOT B AND NOT C AND D
2.7 left = firstterm OR secondterm OR thirdterm OR fourthterm OR fifthterm OR sixthterm
2.8 right = A OR (B AND NOT C) OR (C AND NOT D)
2.9 IF variables in the left equals variables in the right THEN
2.11.1 result equals "The Expression Holds"
ELSE
2.11.2 Result equals "The Expression does not Hold"
3. Display result
```

4.5. Expression

A set of logical values as Boolean algebra in S_3 (=b) of the seven-segment display. Suppose A, B, C and D are members of the set, the algorithm that can be used to verify the complex Boolean algebra expression: A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BCD + AB'C'D' + AB'C'D equals (=) the simplified Boolean algebra expression: B'+ C'D' + CD is follows below:

Boolean expression 4.5b (Boolean A, B, C,D)

```
1 Read in  data
1.1 Captures A
1.2 Captures B
1.3 Captures C
1.4 Captures D
2 Determine expression 4.5b
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND NOT B AND NOT C AND D
2.3 thirdterm = NOT A AND NOT B AND C AND NOT D
2.4 fourthterm = NOT A AND NOT B AND C AND D
2.5 fifthterm = NOT A AND B AND NOT C AND NOT D
2.6 sixthterm = NOT A AND B AND C AND D
2.7 sevethterm = A AND NOT B AND NOT C AND NOT D
2.8 eighthterm = A NOT B AND NOT C AND D
2.7 left = firstterm OR secondterm OR thirdterm OR fourthterm OR fifthterm OR sixthterm OR seventhterm OR
eighthterm
2.8 right = NOT B OR (NOT C AND NOT D) OR (C AND D)
```

```
2.9 IF variables in the left equals variables in the right THEN
2.9.1 result equals "The Expression Holds"
ELSE
2.9.2 result equals "The Expression does not Hold"
3. Display result
```

4.6. Expression

A set of logical values as Boolean algebra in S_4 (=e) of the seven-segment display. Suppose A, B, C and D are members of the set, the algorithm that can be used to verify the complex Boolean algebra expression: A'B'C'D' + A'B'CD' + A'BCD' + AB'C'D' equals (=) the simplified Boolean algebra expression: B'D' + CD' is follows below:

```
Boolean expression 4.6e (Boolean A, B, C,D)
1 Read in  data
1.1 Captures A
1.2 Captures B
1.3 Captures C
1.4 Captures D
2 Determine expression4.6e
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND NOT B AND C AND NOT D
2.3 thirdterm = NOT A AND B AND C AND NOT D
2.4 fourthterm = A AND NOT B AND NOT C AND NOT D
2.5 left = firstterm OR secondterm OR thirdterm OR fourthterm
2.6 right = (NOT B AND NOT D) OR (C AND NOT D)
2.7 IF variables in the left = variables in the right THEN
2.7.1 result equals"The Expression Holds"
ELSE
2.7.2 result equals "The Expression does not Hold"
3. Display result
```

4.7. Expression

A set of logical values as Boolean algebra in S_5 (=d) of the seven-segment display. Suppose A, B, C and D are members of the set, the algorithm that can be used to verify the complex Boolean algebra expression : A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D + A'BCD' + AB'C'D' + AB'C'D equals (=) the simplified Boolean algebra expression: A + CD' + BC'D + B'C + B'D' is follows below:

```
Boolean expression4.7d(Boolean A, B, C,D)
1 Read in  data
1.1 Captures  A
1.2 Captures B
1.3 Captures C
1.4 Captures D
2 Determine expression4.7d
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND NOT B AND C AND NOT D
2.3 thirdterm = NOT A AND NOT B AND C AND D
```

```
2.4 fourthterm = NOT A AND B AND NOT C AND D
2.5 fifthterm = NOT A AND B AND C AND NOT D
2.6 sixthterm = A AND NOT B AND NOT C AND NOT D
2.7 seventhterm = A AND NOT B AND NOT C AND D
2.8 left = firstterm OR secondterm OR thirdterm OR fourthterm OR fifthterm OR sixthterm OR sevethterm
2.9 right = A OR (C AND NOT D) OR (B AND NOT C AND D) OR (NOT B AND C) OR (NOT B AND NOT D)
2.10 IF variables in the left = variables in the right THEN
2.10.1 result equals "The Expression Holds"
ELSE
2.10.2 result equals"The Expression does not Hold"
3. Display result
```

4.8. Expression

A set of logical values as Boolean algebra in S_6 (=c) of the seven-segment display. Suppose A, B, C and D are members of the set, the algorithm that can be used to verify the complex Boolean algebra expression : A'B'C'D' + A'B'C'D + A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D'+ AB'C'D equals (=) the simplified Boolean algebra expression: A + B + C'+ D is follows below:

```
Boolean expression4.8c(Boolean A, B, C,D)
1 Read in  data
1.1 Captures A
1.2 Captures B
1.3 Captures C
1.4 Captures D
2 Determine expression4.8c
2.1 firstterm = NOT A AND NOT B AND NOT C AND NOT D
2.2 secondterm = NOT A AND NOT B AND NOT C AND D
2.3 thirdterm = NOT A AND NOT B AND C AND D
2.4 fourthterm = NOT A AND B AND NOT C AND NOT D
2.5 fifthterm = NOT A AND B AND NOT C AND D
2.6 sixthterm = NOT A AND B AND C AND NOT D
2.7 seventhterm = NOT A AND B AND C AND D
2.8 eighthterm = A AND NOT B AND NOT C AND NOT D
2.9 ninethterm = A AND NOT B AND NOT C AND D
2.10 left = firstterm OR secondterm OR thirdterm OR fourthterm OR fifthterm OR sixthterm OR sevethterm OR
eightterm OR ninethterm
2.11 right = A OR B OR (NOT C) OR D
2.12 IF variables in the left = variables in the right THEN
2.12.1 result equals "The Expression Holds"
ELSE
2.12.2 result equals "The Expression does not Hold"
3. Display result
```

These algorithms can be implemented using Java programming language. These algorithms can be implemented in Java. The reason for choosing Java was that Java programming provides Boolean data type, together with all the relevant Boolean operators. The algorithms have been implemented in a Java class and tested separately. All these Boolean expressions used in fabricating the 7-segment decoder circuit.

5. CONCLUSION

We established in this paper the importance of Boolean algebra applications towards the design of computer hardware and software. We presented novel algorithms approaches in checking and evaluating complex Boolean expression from the fabricated decoder circuit. These algorithms have been implemented and verified using Java programming language.

Funding: This study received no specific financial support.
Competing Interests: The authors declare that they have no competing interests.
Acknowledgement: Both authors contributed equally to the conception and design of the study.

REFERENCES

[1] D. Tan, "On the structure of finite Boolean algebra," *Journal of Mathematical Sciences & Mathematics Education*, vol. 5, pp. 20-24, 2011.

[2] O. E. Oguike, "Algorithms for verifying variants of boolean algebra equations and expressions," *African Journal of Computing & ICT*, vol. 6, pp. 1-8, 2013.

[3] M. Dhar, "Fuzzy sets towards forming boolean algebra," *International Journal of Energy, Information and Communications*, vol. 2, pp. 1-22, 2011.

[4] L. Rompis, "Boolean agebra for xor gates," *Journal of Computer Science and Applications*, vol. 1, pp. 14 - 16, 2013.

[5] E. Clarke and E. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Proceedings of Workshop on Logics of Programs, Lecture Notes in Computer Science, Springer-Verlag*, 1981, pp. 52–71.

[6] J. Burch, E. M. Clarke, K. McMillan, D. Dill, and L. Hwang, "Symbolic model checking: 1020 states and beyond," in *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science*, 1990, pp. 428–39.

[7] G. Holzmann, *Design and validation of computer protocols*. Englewood Cliffs: Prentice Hall, Pearson Technology Group USA, 1990.

[8] E. E. A. and C. Lei, "Efficient model checking in the fragments of the propositional μ-calculus," presented at the Symposion on Logic in Computer Science, IEEE Computer Society Press, 1986.

[9] D. Long, A. Browne, E. Clarke, S. Jha, and W. Marrero, "An improved algorithm for the evaluation of fixpoint expressions," in *Proceedings of International Conference on Computer Aided Verification (CAV'1994), Lecture Notes in Computer Science, Springer-Verlag*, 1994, pp. 338–350.

[10] H. Seidl, "Fast and simple nested fixpoints " *Information Processing Letters*, vol. 59, pp. 303–308, 1996. Available at: https://doi.org/10.1016/0020-0190(96)00130-5.

[11] X. Liu, C. R. Ramakrishnan, and S. A. Smolka, "Fully local and efficient evaluation of alternating fixed points," in *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, Springer Verlag*, 1998, pp. 5–19.

[12] H. R. Andersen and B. Vergauwen, "Efficient checking of behavioural relations and modal assertions using fixed-point inversion," in *Proceedings of Conference on Computer Aided Verification, Lecture Notes on Computer Science, Springer-Verlag,* , 1995, pp. 142–154.

[13] R. Cleaveland and B. Steffen, "Computing behavioural relations logically," in *Proceedings of the 18th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Springer-Verlag*, 1991, pp. 127–138.

[14] K. Larsen, "Efficient local correctness checking," in *Proceedings of Conference on Computer Aided Verification, Lecture Notes on Computer Science, Springer-Verlag,* , 1992, pp. 30–43.

[15] M. R. . "A generic on-the-fly solver for alternation-free boolean equation systems," in *Proceedings of the 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003), Lecture Notes in Computer Science, Springer Verlag*, 2003, pp. 81–96.

[16] A. Mader, "Verification of modal properties using boolean equation systems," PhD Thesis, Technical University of Munich, 1997.

[17] J. F. Groote and T. Willemse, "A checker for modal formulas for processes with data," in *Proceedings of Formal Methods for Components and Objects, Second International Symposium (FMCO'2003), Lecture Notes in Computer Science, Springer-Verlag*, 2004, pp. 223–239.

[18] J. F. Groote and T. Willemse, "Parameterised boolean equation systems," in *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR'2004), Lecture Notes in Computer Science, Springer-Verlag*, 2004, pp. 308–324.

[19] J. F. Groote and T. Willemse, "Parameterised boolean equation systems," *Theoretical Computer Science*, vol. 343, pp. 332–369, 2005.

[20] M. Azram, "Arithmetic version of boolean algebra," in *Proceeding of 2nd IEEE Internation Conference on Computer Science and Information Technology*, 2009, pp. 79 – 80.