Design and Implementation of a Laboratory Prototype for Boolean Function Simplification in Real Time

Umair Butt[1], Zeeshan Azmat[1], Jason Gu[2], Muhammad Usman Asad[2], Umar Farooq[1,2], Valentina Emilia Balas[3], Khurrum Karim Qureshi[4] and Ghulam Abbas[5]

[1]Intelligent Systems Laboratory & Automation Facility (ISLAF), IEECE, PU, 54590, Lahore, Pakistan; umar.ee@pu.edu.pk, zeeshan.ee@pu.edu.pk

[2]Department of Electrical and Computer Engineering, Dalhousie University, Halifax, B3H 4R2, Canada; mh549096@dal.ca, jason.gu@dal.ca, umar.farooq@dal.ca

[3]Department of Automatics and Applied Software "Aurel Vlaicu" University of Arad, Arad, Romania; valentina.balas@uav.ro

[4]Department of Electrical Engineering, King Fahad University of Petroleum & Minerals, Dhahran, Saudi Arabia; kqureshi@kfupm.edu.sa

[5]Department of Electrical Engineering, The University of Lahore, 1-KM Defence Road, Lahore,54000, Punjab, Pakistan; ghulam.abbas@ee.uol.edu.pk

Abstract: The aim of this study is to develop a three-variable Karnaugh-Map (K-MAP) in hardware. K-MAPs are extensively used in digital logic design for simplifying Boolean expressions. Here, we automate the process of Boolean expression simplification by designing a K-MAP hardware circuit, that can show the simplified expression in real-time. The hardware is comprised of input, combinatory, comparator, and output units. The input unit accepts minterms from the DIP switches. These minterms are processed in combinatory unit against all the possible combinations of boxes in power-of-two. The output from the boxes with higher power-of-two is kept in the comparator unit and the result is displayed on LEDs in the output unit. The proposed hardware design can be used in a classroom setting to teach the concepts of Boolean function simplification and verify the practice problems.

Keywords: K-Map; Boolean Algebra; Digital Logic Design; Electronics

1 Introduction

Karnaugh maps (K-MAP) are widely used for simplifying Boolean functions, which are represented as the sum of minterms or the product of max-terms format. The major advantage of this technique lies in the systematic nature of the simplification process, as opposed to the classical Boolean algebra [1-3]. Literature reveals that K-MAPs have been used in a variety of applications such as secure transmission systems [4], database normalization [5], reliability engineering [6], pattern classification [7], power electronics [8], chemical kinetics [9], software engineering [10], and others. A number of modifications to the K-MAP algorithm have also been proposed which make it even more versatile e.g., a method is presented in [11] to introduce the traceability feature of minterms when mapped from the Boolean expression onto the K-MAP. This add-on will benefit students and instructors alike when reviewing the problem at a later stage. A generalized K-MAP technique is proposed in [12] to lessen the difficulty of simplifying Boolean expressions having more than 6 variables. A tree of four variable K-MAPs is constructed and the notion of K-Don't care is introduced along with backtracking to simplify the Boolean expression. Another lower complexity version, Yasser (Y-MAP) is proposed in [13]. The capability of K-MAP is extended in [14] to yield optimal multi-level logic designs based on XOR patterns. The modified K-MAP is used for the minimization of spintronic and memristive logic functions in [15]. The concept of combining boxes on the K-MAP is introduced in [16] which results in a more simplified Boolean expression than the standard combination scheme.

In our present work, we reimagine the process of Boolean expression simplifications by K-MAP as a proper functional device that utilizes the fundamentals of K-MAP to yield simplified expressions in real time. With our proposed device, students can simply consult the truth table and feed the potential minterms as inputs to the device which shall give a simplified expression at the output. The process of construction of the proposed hardware prototype can be used as a case study in an undergrad digital logic design course. In addition, it can be used in a classroom environment to verify the practice problems. To the best of the authors' knowledge, the proposed hardware design of K-MAP has not been presented before.

This paper is organized as follows: The proposed hardware design is presented in Section 2. The software and hardware implementation results are included in Section 3, followed by Conclusions in Section 4.

2 Proposed Hardware Design

To start with, we assume that the Boolean function is represented in the sum of the minterms format. The possible minterms appearing in the 3-variable K-map are shown in Table 1. These 8 minterms form the input to the proposed hardware and user can enter minterms of their choice through 8 selection switches, where the HIGH status of a switch S_x indicates the presence of minterm, m_x in the Boolean expression. The selected minterms are passed to the combinatory unit which generates 27 Boolean expressions (as shown in Appendix A) corresponding to all possible combinations of boxes in a 3-variable K-map. The co-existence of various combinations, as yielded by the combinatory unit, is eliminated in the comparator unit. The final simplified Boolean expression is displayed on LEDs, where a particular LED represents one of the 27 possible terms in the output Boolean expression. The detailed design of combinatory and comparator units will be presented next.

Table 1
Truth table 3-variable K-Map [2]

A	B	C	MINTERM	EXPRESSION
0	0	0	**m_0**	**A`B`C`**
0	0	1	**m_1**	**A`B`C**
0	1	0	**m_2**	**A`BC`**
0	1	1	**m_3**	**A`BC**
1	0	0	**m_4**	**AB`C`**
1	0	1	**m_5**	**AB`C**
1	1	0	**m_6**	**ABC`**
1	1	1	**m_7**	**ABC**

2.1 Combinatory Unit

The design of the combinatory unit is based on the fact that the boxes or squares in the K-Map are combined according to the integer powers of 2. To this end, all possible combinations and the respective Boolean expressions are determined, as outlined below.

2.1.1 Combination of 8-Boxes

Since the assumed Boolean expression is a function of three variables, there is only one way for combining 8 boxes on the K-map, as depicted in Fig. 1. Subsequently, we obtain only one expression i.e., $f_8(A,B,C) = \sum(0,1,2,3,4,5,6,7)$.

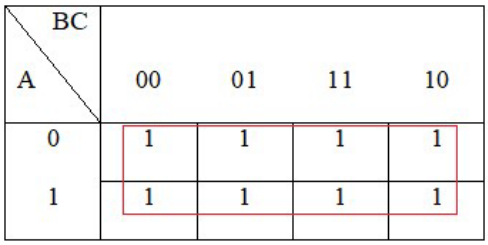


Figure 1
Possibility of combining 8-boxes

2.1.2 Combination of 4-boxes

Now, we consider possible ways of combining 4 boxes. It is found that there are 6 such combinations which result in a total of 6 expressions, out of which 2 are obtained through horizontal grouping of minterms i.e., $f_{4,1}(A,B,C)=\sum(0,1,3,2)$, $f_{4,2}(A,B,C)=\sum(4,5,7,6)$, while 3 are obtained through vertical grouping of minterms i.e., $f_{4,3}(A,B,C)=\sum(0,1,4,5)$, $f_{4,4}(A,B,C)=\sum(1,3,5,7)$, $f_{4,5}(A,B,C)=\sum(3,2,7,6)$, and the remaining is due to the minterms at the edges of the K-map i.e., $f_{4,6}(A,B,C)=\sum(0,2,4,6)$. This process is shown graphically in Fig. 2.

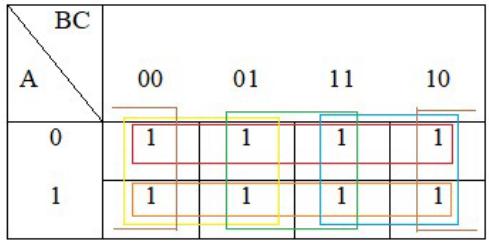


Figure 2
Possibility of combining 4-boxes

Red Box: $f_{4,1}=A'$, Orange Box: $f_{4,2}=A$, Yellow Box: $f_{4,3}=B'$, Green Box: $f_{4,4}=C$, Blue Box: $f_{4,5}=B$, Brown Box: $f_{4,6}=C'$

2.1.3 Combination of 2-boxes

Now, we search all possible ways of combining 2 boxes at a time. This yields a total of 12 expressions, out of which 6 are obtained through horizontal grouping of minterms i.e., $f_{2,1}(x,y,z)=\sum(0,1)$ $f_{2,2}(x,y,z)=\sum(1,3)$,

$f_{2,3}(x,y,z)=\sum(3,2)$ $f_{2,4}(x,y,z)=\sum(4,5)$, $f_{2,5}(x,y,z)=\sum(5,7)$, $f_{2,6}(x,y,z)=\sum(7,6)$, while 4 are obtained through vertical grouping of minterms i.e., $f_{2,7}(x,y,z)=\sum(0,4)$, $f_{2,8}(x,y,z)=\sum(1,5)$, $f_{2,9}(x,y,z)=\sum(3,7)$, $f_{2,10}(x,y,z)=\sum(2,6)$, and the remaining 2 are due to the minterms at the edges i.e., $f_{2,11}(x,y,z)=\sum(0,2)$, $f_{2,12}(x,y,z)=\sum(4,6)$. The pictorial representation is given in Fig. 3.

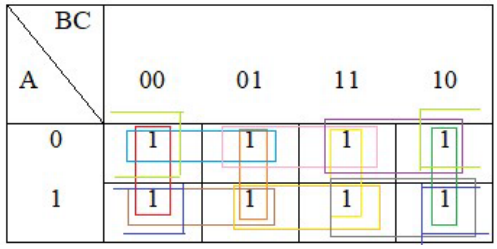


Figure 3
Possibility of combining 2-boxes

Blue Box: $f_{2,1}=A'B'$, Pink Box: $f_{2,2}=A'C$, Purple Box: $f_{2,3}=A'B$, Brown Box: $f_{2,4}=AB'$, Gold Box: $f_{2,5}=AC$, Gray Box: $f_{2,6}=AB$, Red Box: $f_{2,7}=B'C'$, Orange Box: $f_{2,8}=B'C$, Yellow Box: $f_{2,9}=BC$, Green Box: $f_{2,10}=BC'$, Light Green Box: $f_{2,11}=A'C'$, Dark Blue Box: $f_{2,12}=AC'$

2.1.4 Combination of 1-boxes

Here, we treat each box separately (Fig. 4) which results in a total of 8 expressions. These Boolean functions correspond to the expressions of minterms, as shown in Table 1 i.e., $f_{1,1}=m_0$, $f_{1,2}=m_1$, $f_{1,3}=m_2$, $f_{1,4}=m_3$, $f_{1,5}=m_4$, $f_{1,6}=m_5$, $f_{1,7}=m_6$, $f_{1,8}=m_7$.

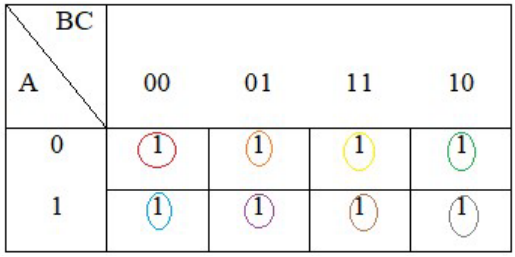


Figure 4
Possibility of combining 1-boxes

Red Box: $f_{1,1}$ **A`B`C`**, Orange Box: $f_{1,2}$ **A`B`C**, Yellow Box: $f_{1,3}$ **A`BC**, Green Box: $f_{1,4}$ **A`BC`**, Blue Box: $f_{1,5}$ **AB`C`**, Purple Box: $f_{1,6}$ **AB`C**, Brown Box: $f_{1,7}$ **ABC**, Gray Box: $f_{1,8}$ **ABC`**

2.2 Comparator Unit

To retain only the simplified terms, outputs from the combinatory units are processed in comparator unit which eliminates the extra terms. For instance, let $g(A,B,C)=\sum(0,2,4,6)$ be the Boolean function to be simplified. This function will activate the following outputs of combinatory units: $f_{1,1}, f_{1,3}, f_{1,5}, f_{1,7}, f_{2,7}, f_{2,10}, f_{2,11}, f_{2,12}, f_{4,6}$. However, the simplified expression should only contain $f_{4,6}$. Thus, comparator unit is imperative to address the overlapped boxes in K-map and such overlapping's in K-map are resolved in following ways:

2.2.1 Comparator Unit for 2-boxes and 4-boxes

The basis for this unit is explained through previous example of simplifying the Boolean function g. Since $f_{4,6}=C'$ is activated, it should inhibit the outputs $f_{2,7}, f_{2,10}, f_{2,11}, f_{2,12}$ as they all contain $f_{4,6}$ i.e. $f_{4,6}=f_{2,7}+f_{2,10}+f_{2,11}+f_{2,12}=C'(B'+B+A'+A)=C'$. Thus, for this specific example, the simplified outputs of 4boxes and 2boxes combinations are generated as:

$$O_{4,6}=f_{4,6}, O_{2,7}=f'_{4,6}f_{2,7}, O_{2,10}=f'_{4,6}f_{2,10}, O_{2,11}=f'_{4,6}f_{2,11}, O_{2,12}=f'_{4,6}f_{2,12}$$

Continuing in this manner, we eliminate the possibilities of generation of extra 2-box-terms by forming the final outputs as:

$$O_{2,1}=f_{2,1}f'_{4,1}f'_{4,3}f'_{8}, O_{2,2}=f_{2,2}f'_{4,1}f'_{4,4}f'_{8}, O_{2,3}=f_{2,3}f'_{4,1}f'_{4,5}f'_{8}, O_{2,4}=f_{2,4}f'_{4,2}f'_{4,3}f'_{8},$$

$$O_{2,5}=f_{2,5}f'_{4,2}f'_{4,4}f'_{8}, O_{2,6}=f_{2,6}f'_{4,2}f'_{4,5}f'_{8}, O_{2,7}=f_{2,7}f'_{4,3}f'_{4,6}f'_{8}, O_{2,8}=f_{2,8}f'_{4,3}f'_{4,4}f'_{8},$$

$$O_{2,9}=f_{2,9}f'_{4,5}f'_{4,4}f'_{8}, O_{2,10}=f_{2,10}f'_{4,5}f'_{4,6}f'_{8}, O_{2,11}=f_{2,11}f'_{4,1}f'_{4,6}f'_{8}, O_{2,12}=f_{2,12}f'_{4,2}f'_{4,6}f'_{8},$$

Here, f_8 is active when all the (input) minterms are high. The final outputs of the 4-box-terms are generated as:

$$O_{4,1}=f_{4,1}f'_{8}, O_{4,2}=f_{4,2}f'_{8}, O_{4,3}=f_{4,3}f'_{8}, O_{4,4}=f_{4,4}f'_{8}, O_{4,5}=f_{4,5}f'_{8}, O_{4,6}=f_{4,6}f'_{8},$$

2.2.2 Comparator Unit for 1-boxes

Continuing previous example of simplifying the Boolean function g, the combinatory outputs $f_{1,1}, f_{1,3}, f_{1,5}, f_{1,7}$ are eliminated by virtue of their generation mechanism, which is based on the fact that a minterm has three neighbors on a three variable K-map. A particular combinatory output from 1box unit will only be activated if all its neighboring entries on K-map are zero. This implies that all combinatory outputs $f_{1,1}, f_{1,3}, f_{1,5}, f_{1,7}$ will be inactivated since one of the neighbors of each minterm is zero e.g., $m_1 = 0$ leads to $f_{1,1} = 0$, and $m_3 = 0$ leads to $f_{1,3} = 0$. By generalizing this process, we obtain the outputs of 1box units as:

$$O_{1,1} = m_0 m_1' m_2' m_4', O_{1,2} = m_1 m_0' m_3' m_5', O_{1,3} = m_3 m_1' m_2' m_7',$$
$$O_{1,4} = m_2 m_3' m_6' m_0', O_{1,5} = m_4 m_0' m_5' m_6', O_{1,6} = m_5 m_4' m_7' m_1',$$
$$O_{1,7} = m_7 m_5' m_6' m_3', O_{1,8} = m_6 m_7' m_2' m_4'.$$

3 Results

The proposed K-MAP solver is first tested in MULTISIM environment and the complete circuit diagram is shown in Fig. 5. The following ICs are used for implementing the framework: 74LS04(Not Gate), 74LS08(2 input And Gate), 74LS11(3 input And Gate) and 74LS21(4 input And Gate). After the circuit is implemented on MULTISIM, we verify its output for different cases. The results are recorded in the form of a video which contains the following 5 cases in sequence (video can be downloaded from ISLAF YouTube channel at https://youtu.be/j5H6NrICyTw).

3.1 Case-1:

3.1.1 Find Boolean Expression for Sum of Minterms (m_0, m_2, m_3, m_4, m_5, m_6, m_7)

A \ BC	00	01	11	10
0	1	1	1	1
1	1	1	1	1

In this case, only $O_8 = f_8$ will be active while all other outputs $O_{4,1},...,O_{4,6}$, $O_{2,1},...,O_{2,12}$, and $O_{1,1},...,O_{1,8}$ will be zero either due to the presence of complement of f_8 in each of these outputs (AND operation) or by virtue of the mechanism of generation.

3.2 Case-2:

3.2.1 Find Boolean Expression for Sum of Minterms (m_0, m_2, m_3, m_4, m_5, m_6)

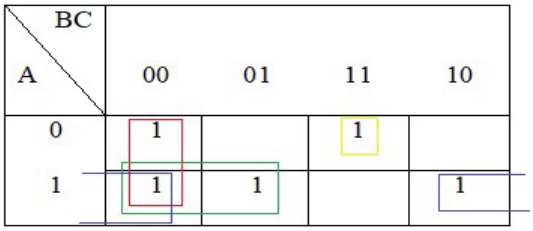
A \ BC	00	01	11	10
0	1		1	1
1	1	1		1

Red Box: **C'**, Blue Box: **A'B**, Green Box: **AB' : Sum of Product = C'+AB' + A'B**

In this case, 1-combination outputs and 8-combinations outputs are clearly zero. Only one 4-combination output $O_{4,6}$ is active which inhibits the 2-combinations outputs $O_{2,7}$ and $O_{2,10}$ due to the presence of the complement of $f'_{4,6}$ in these terms. The only 2-combination outputs, $O_{2,3} = f_{2,3}f'_{4,1}f'_{4,5}f'_8$, and $O_{2,4} = f_{2,4}f'_{4,2}f'_{4,3}f'_8$, will be active, as $f_{2,1}, f_{2,2}, f_{2,5}, f_{2,6}, f_{2,8}, f_{2,9}, f_{2,11}, f_{2,12}$ are all zero.

3.3 Case-3

3.3.1 Find Boolean Expression for Sum of Minterms (m_0, m_3, m_4, m_5, m_6)

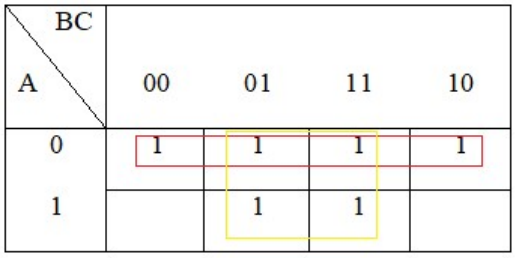


Red Box: **B'C', A'BC,** Green Box: **AB',** Blue Box: **AC' Sum of Product = B'C' + AB' + AC' + A'BC**

In this case, 8-combination output is zero while only 1-combination output $O_{1,3} = m_3 m_1' m_2' m_7'$ is active due to three surrounding inactive minterms. Furthermore, there is no contribution from 4-combination outputs. However, 2-combination outputs $O_{2,4} = f_{2,4} f_{4,2}' f_{4,3}' f_8'$, $O_{2,7} = f_{2,7} f_{4,3}' f_{4,6}' f_8'$, and $O_{2,12} = f_{2,12} f_{4,2}' f_{4,6}' f_8'$, will be active while all other 2-combinations outputs are zero due to the inactivation of their respective functions, $f_{2,x}$.

3.4 Case-4

3.4.1 Find Boolean Expression for Sum of Minterms (m_0, m_1, m_2, m_3, m_5, m_7)



Red Box: **A'**, Yellow Box: **C : Sum of Product = A' + C**

In this case, 8-combination and 1-combination outputs are clearly zero. The 2-combination outputs are also zero, either due to the activation of certain 4-box combination outputs or the inactivation of respective Boolean functions, $f_{2,x}$

$f_{4,1}$, and $f_{4,4}$

resulting in the activation of outputs, $O_{4,1}$, and $O_{4,4}$, respectively.

3.5 Case-5

3.5.1 Find Boolean Expression for Sum of Minterms (m_0, m_3, m_5, m_6)

A \ BC	00	01	11	10
0	1		1	
1		1		1

Red Box: **A'B'C'**, Yellow Box: **AB'C**, Orange Box: **A'BC**, Green Box: **ABC'**

Sum of Product = A'B'C + A'BC + AB'C+ABC'

In this case, Boolean functions pertaining to 8-combination, 4-combinations, and 2-combinations boxes are all inactive. The only activity is generated by the 1-combination boxes who's neighboring minterms are inactive i.e., outputs $O_{1,1}, O_{1,3}, O_{1,5}, O_{1,6}$ will be HIGH.

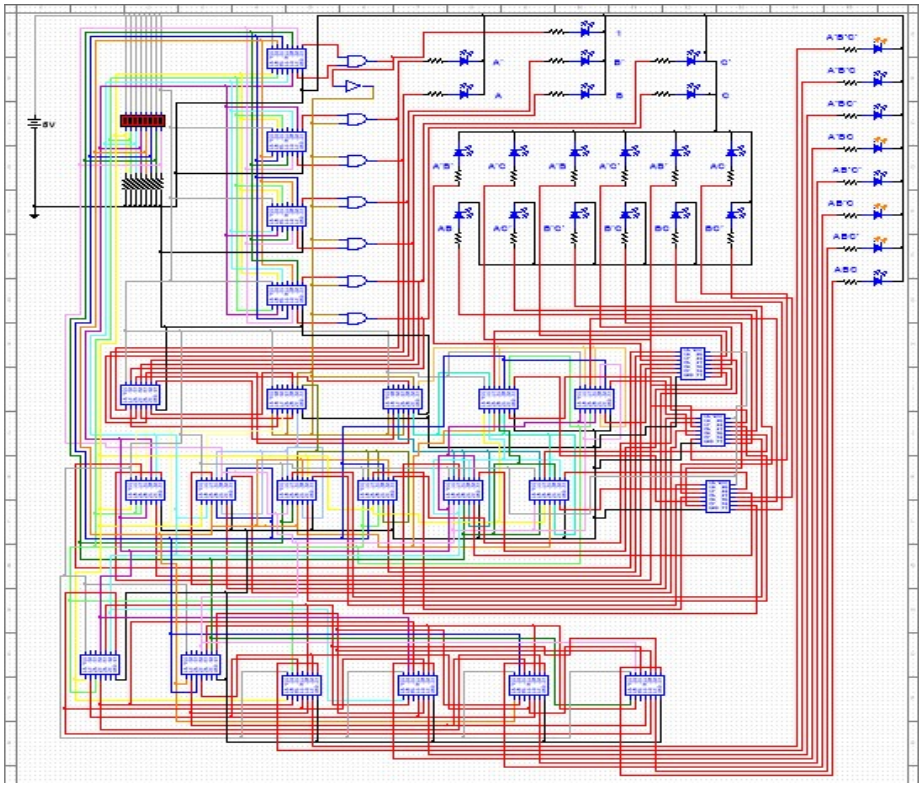


Figure 5
Proposed hardware circuit for real time simplification of Boolean functions

Finally, hardware of the proposed K-MAP solver is constructed and is depicted in Fig. 6 (a). The circuit is verified for different cases e.g., when all the input minterms are active (Fig. 6(b)), when only one of the input minterms is active (Fig. 6(c)), and when four of the input minterms are active (Fig. 6(d), Fig. 6(e)).

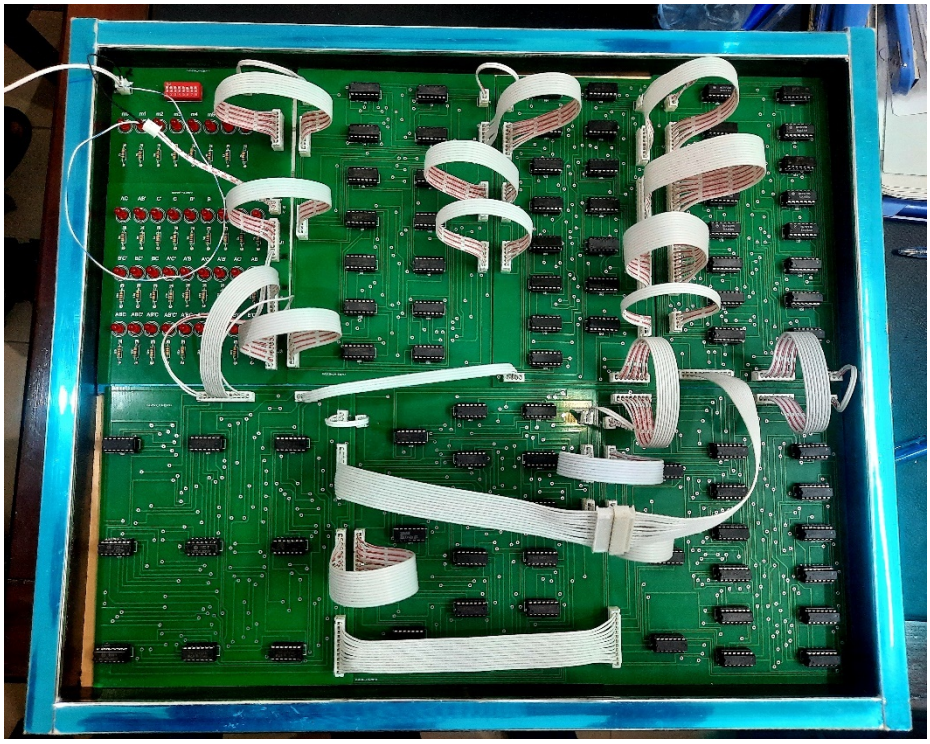


Figure 6

(a)

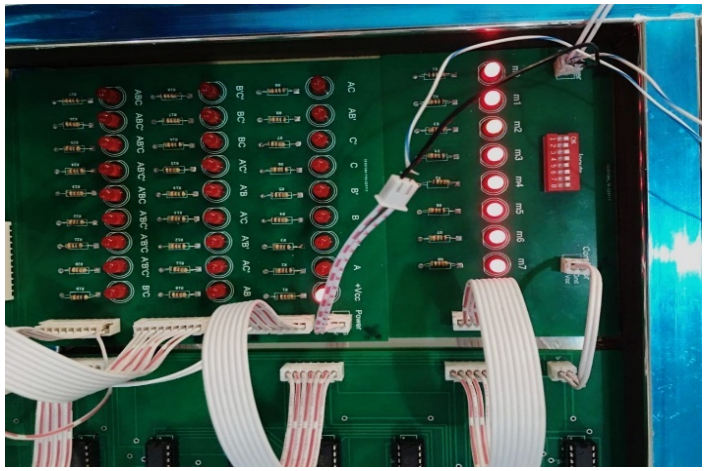


Figure 6

(b)

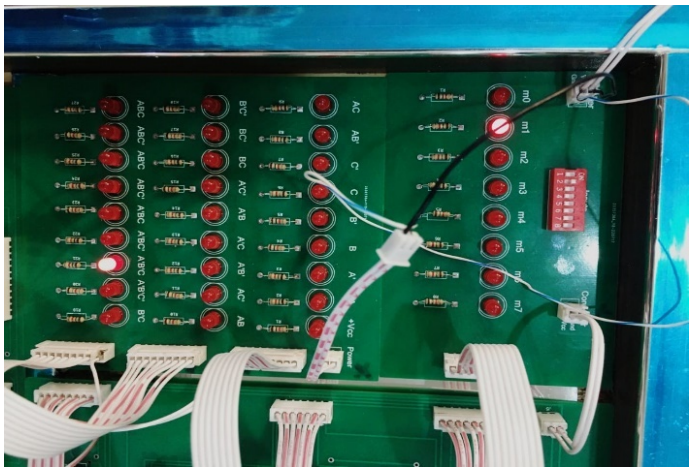


Figure 6

(c)

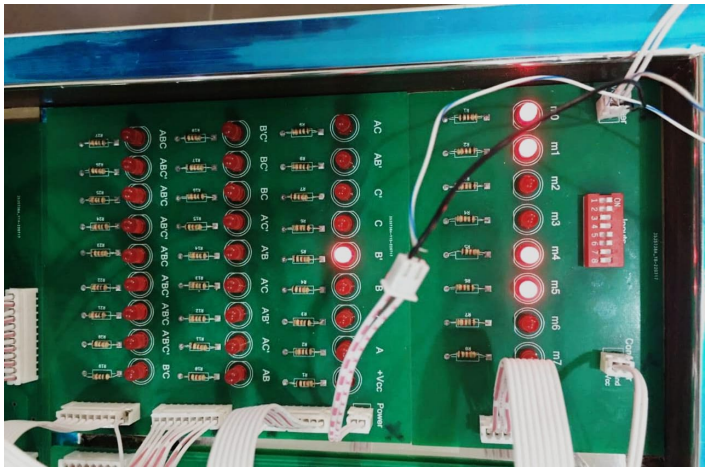


Figure 6

(d)

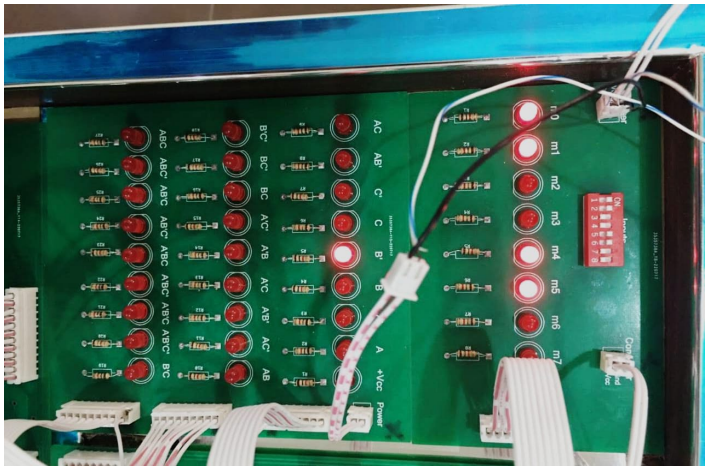


Figure 6

(e)

Hardware of the proposed K-MAP solver (a) Hardware unit (b) Input-Output unit showing result of $\sum(m_0,...,m_7)$ (c) Input-Output unit showing result of $\sum(m_1)$ (d) Input-Output unit showing result of $\sum(m_0,m_2,m_4,m_6)$ (e) Input-Output unit showing result of $\sum(m_0,m_1,m_4,m_5)$

Conclusions

In this paper, we have presented a hardware design for determining the simplified Boolean expressions through K-MAP. The circuit accepts minterms from DIP switches and processes them in the combinatory and comparator units to yield the simplified Boolean expression, which is displayed on LEDs. The interesting aspect of the proposed hardware, is the utilization of K-MAP concepts to develop a hardware version of K-MAP. The proposal can be included in undergrad course on digital logic design, as a case study.

In the future work a hardware version of higher order K-MAPs with Don't-care conditions, will be developed.

Acknowledgement

This work was supported by NSERC.

References

[1] M. Karnaugh, 'The map method for synthesis of combinational logic circuits', *Transactions of the American Institute of Electrical 162 Engineers, Part I: Communication and Electronics*, Vol. 72, pp. 593-599, 1953

[2] M. Mano, *Digital Design*, 5th Edition. Pearson Publishers, 2013

[3] U. Farooq, *Lecture notes on digital logic design*, University of the Punjab, Lahore, 2012

[4] K. N. Plataniotis and S. Stergiopoulos, 'Karnaugh map 2,2 secret sharing scheme for color images', *16th International Conference on Digital Signal Processing*, pp. 1-6, 2009

[5] D. J. Russomanno and R. D. Bonnell, 'A pedagogical approach to database design via Karnaugh maps', *IEEE Transactions on Education*, Vol. 42, pp. 261-270, 1999

[6] R. G. Bennetts, 'Analysis of reliability block diagrams by Boolean techniques', *IEEE Transactions on Reliability*, Vol. 31, pp. 159-166, 1982

[7] T. Hsu and S. Wang, 'The K1-map reduction for pattern classification', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 616-622, 1997

[8] J. Chen *et al*., 'A single-phase step-up seven-level inverter with a simple implementation method for level-shifted modulation schemes', *IEEE Access*, Vol. 7, pp. 146552-146565, 2019

[9] L. Ge *et al*., 'A formal combinational logic synthesis with chemical reaction networks', *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2017

[10] L. Yu and W. Tsai, 'Test case generation for Boolean expressions by cell covering', *IEEE Transactions on Software Engineering*, Vol. 44, pp. 77-99, 2018

[11] M. E. Holder, 'A modified Karnaugh map technique', *IEEE Transactions on Education*, Vol. 48, pp. 206-207, 2005

[12] V. C. Prasad, 'Generalized Karnaugh map method for Boolean functions of many variables', *IETE Journal of Education*, pp. 11-19, 2017

[13] Y. S. Abdalla, 'Introducing the Yasser-map as an improvement of the Karnaugh-map for solving logical problems', *IEEE International Conference on Electrical, Computer and Communication Technologies*, 2015

[14] R. F. Tinder, 'Multilevel logic minimization using K-map XOR patterns', *IEEE Transactions on Education*, Vol. 38, pp. 370-375, 1995

[15] V. Vyas *et al*., 'Karnaugh map method for memristive and spintronic asymmetric basic logic functions', *IEEE Transactions on Computers*, Vol. 70, pp. 128-138, 2021

[16] M. Garrido, 'Simplifying Karnaugh maps by making groups of non-power-of-two elements', *Circuits, Systems, and Signal Processing,* Vol. 41, pp. 5895-5902, 2022