ISSN: 2456-043X

International Journal of Engineering Science and Generic Research (IJESAR)
Available Online at www.ijesar.in
Volume 3; Issue 6; November-December; 2017; Page No. 46-51

Implementation of High Speed Single Error Correction–Double Adjacent Error Correction (72, 64) Codes

[1] Shaik Khadar Basha, [2] K.S.N. Vittal

[1]Student at QIS Institute of Technology, Ongole, AP, India.

[2]Assistant Professor at QIS Institute of Technology, Ongole, AP, India.

Abstract

Single error correction and double-adjacent error correction (SEC–DAEC) codes are a type of error correction codes (ECCs) capable of correcting single and double-adjacent errors. The existing methods (39, 32) Per-Bit Joined-Pattern Correction. In these technique having more delay, to overcome this problem (72, 64) SEC–DAEC with pipeline method. This brief proposes methods to optimize the decoder of SEC–DAEC codes when implemented in an FPGA, reducing the resource utilization when compared with the conventional implementations. The implementation of two codes, a (72, 64) SEC–DAEC and a (72, 64) SEC–DAEC with pipeline, show that the proposed designs reduce the resource utilization of the correction circuit when comparing it with the conventional designs.

Index Terms: Error correction codes (ECCs), field-programmable gate arrays (FPGAs), lookup tables (LUTs), single error correction and double-adjacent error correction (SEC–DAEC).

1. INTRODUCTION

Field-programmable gate arrays (FPGAs) are already used in many different application areas, including space and avionics [1], [2]. Their low cost and reprogramming capabilities [3] will continue to make them an option for this type of missions. Single event upsets (SEUs) are important causes of errors in electronics, especially in harsh environments. Multiple cell upsets (MCUs) can also occur, so more than one bit is changed. Usually, the affected bits are close to each other and, in many cases, they are adjacent. Some studies [4], [5] reported double error MCUs in static random access memories (SRAMs). Specific techniques are developed to recover the original information when SEUs or MCUs appear. Error correction codes are one of those techniques. They are widely used in memories [6]. Linear block codes [7] are very popular due to the low complexity of the decoding circuitry. Single error correction–double error detection (SEC–DED) codes have been traditionally used when there is no need to correct multiple errors or when enough interleaving distance is used. However, interleaving can impact area, delay, and aspect ratio [4], [8], which can be a problem in embedded memories. When interleaving is not applicable, single error correction and double-adjacent error correction (SEC–DAEC) can be used. SEC–DAEC codes have been proposed to protect memories [8], [9]. Some FPGAs include SEC–DED protection for the internal Block-RAMs. The system implemented in the FPGA may require the use of SEC–DAEC to better protect either the Block RAMs or an external SRAM memory. The FPGA logic can be used to provide this SEC–DAEC support. In the case of BRAMs, the logic implementing the protection may need to be replicated to protect each BRAM.

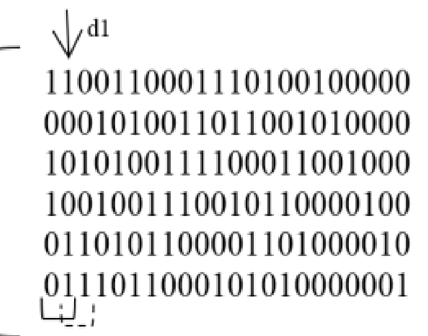


Fig. 1: Example of an SEC–DAEC check matrix

Maximizing FPGA resource utilization helps in reducing area wastage. This brief presents a strategy to optimize the design of SEC–DAEC decoders when implementing them into Xilinx14.7-series FPGAs. The design reduces the utilization of resources by taking advantage of the regular structure of the FPGA. This approach may be used with other FPGAs and codes that share similar characteristics. The rest of this brief

Corresponding author: Shaik Khadar Basha

is structured as follows. First, the SEC–DAEC code is described, and the conventional decoding approaches are presented. Then, the features of the selected FPGA are described. Later, the proposed designs are presented. Finally, they are compared with the conventional implementation.

2. EXISTING DESIGN

A. SEC–DAEC Codes

Linear codes (n, k) convert a data word of k bits into a code word of n bits, adding n-k parity bits using a specific algorithm. These codes can be represented by their parity-check matrix H (with n columns and n-k rows). This matrix is used to process the received words to obtain a result called syndrome. Fig. 1 shows an example of the parity-check matrix of a linear code. The first 16 bits correspond to the data bits, and the other 6 bits correspond to the parity-check bits. The calculated syndrome is a vector of length equal to the number of parity bits in the code. If the syndrome is all zeros, the received word corresponds to a code word, and no error is present. Otherwise, an error occurred. When the syndrome matches one of the columns of the H matrix, a single error (SE) is detected, and can be corrected by flipping back the appropriate bit. For instance, based on the matrix from Fig. 1, a syndrome of 100011 matches the second column (marked with an arrow), so an SE is detected in that bit and will be corrected by inverting its value. In SEC–DAEC codes, double-adjacent errors (DAEs) can also be corrected.

The syndrome obtained in those cases matches the XOR of two of the matrix adjacent columns. The error is corrected by changing the corresponding bit positions. In the previous example, the check matrix from Fig. 1 is derived from an SEC–DAEC code. When a syndrome of 001111 is calculated, a DAE is corrected in the first and second bits as that syndrome is the XOR of the first and second columns of the matrix (101100 XOR 100011), marked with a solid line in Fig. 1. Thus, we can say that syndromes 100011 (the second column of the matrix), 001111 (XOR of the first and second columns), and 101000 (XOR of the second and third columns) indicate that an error occurred in the second bit of the code word, and it must be corrected. This can be replicated for any data bit. Parity bits typically do not need to be corrected. If the syndrome matches one of the parity bit positions or the XOR of two adjacent parity columns, then we can state that no error occurred in the data bits even with a nonzero syndrome.

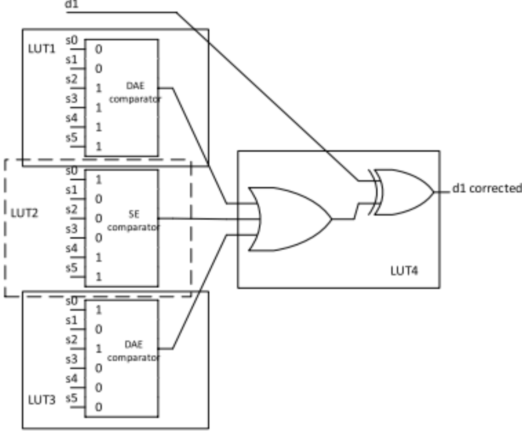


Fig. 2: SEC–DAEC correction for the second bit in the (22, 16) code.

The conventional implementation of the correction circuit is shown in Fig. 2, using the second bit of the example (d1). The syndrome bits (s0 to s5) are compared with the patterns that identify an error in this second bit. The first and third comparators check against the DAE patterns, while the second one evaluates the SE pattern. An OR gate identifies if the comparators detected an error. An XOR gate corrects the error in the second bit (d1) if needed.

B. Logic Structure of FPGAs

FPGAs can be adapted to specific designs and modified when needed due to their architecture based on reprogrammable logic blocks. Lookup tables (LUTs) are one of the components of these logic blocks, used to implement the logic functions of the design. The number and characteristics of the LUTs and other elements depend on the manufacturer and the product family. Many of these manufacturers provide additional flexibility, making the n-input LUTs capable of performing more than one function simultaneously. Xilinx 5-, 6-, and 7-series FPGAs incorporate six-input LUTs that can implement a six-input function or two five-input functions if all inputs are shared [10]. Altera Corporation [11] also provides similar capabilities. This flexibility can be used to optimize the area utilization. LUT minimization by technology mapping is NP-hard [12], [15]. A number of algorithms have been proposed to reduce area utilization in LUT-based FPGAs [13], [14], with some of them [15], [16] addressing the multi-output functions provided by some of these FPGAs. Manufacturers are still conducting research in this

© 2017 IJESAR. All Rights Reserved.

topic [17]. However, tools provided by these manufacturers still seem to fail in taking full advantage of these approaches or else those algorithms are not effective for the decoder designs under study.

C. Per-Bit Joined-Pattern Correction With Six Parity Bits

The first code has been introduced in Fig. 1, and it is a (22, 16) code with n = 22, k = 16, and 6 parity bits. A syndrome equal to the XOR of two adjacent columns implies that a DAE was detected in the bits located on those column positions. The conventional design shown in Fig. 2 can be implemented into a Xilinx 14.7 -series FPGA in a very simple way, using a six-input LUT for every comparator and another LUT for the OR and XOR gates (using four inputs). To correct a single bit, four different LUTs are used, as shown in Fig. 2. Therefore, a maximum of 64 LUTs are needed, assuming no optimization when mapping is done.

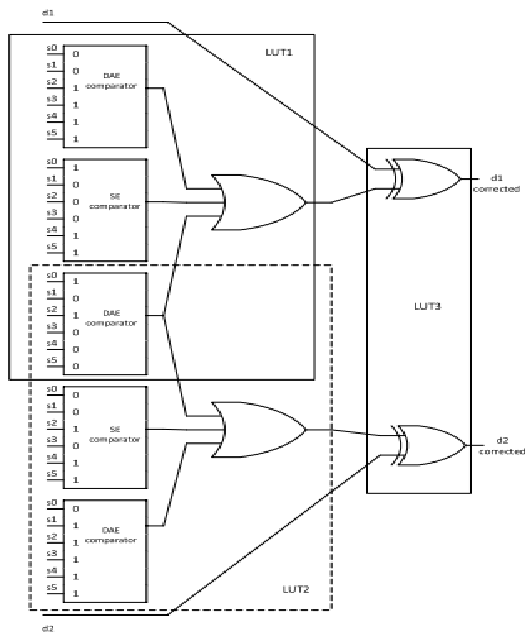


Fig. 3: SEC–DAEC (22, 16) per-bit joined-pattern correction optimization.

The first optimization consists of performing a per-bit joined-pattern correction. A single LUT can be used to output a “1” value when the bit has to be corrected and “0” otherwise by checking all the patterns for that bit simultaneously. As an example, Fig. 3 shows how to group the three comparators and the OR gate of the second bit into a single six-input LUT (the third bit correction is also drawn). In addition, the XOR gate that corrects the error has only two inputs, so a single five-input LUT (with two outputs) can be used to correct two bits simultaneously. This is represented by LUT3 in Fig. 3. Two inputs of this LUT correspond to the bits to be corrected (d1 and d2), and the others are the outputs of the six-input LUTs for those bits. Therefore, 3 LUTs would be needed for every 2 bits, 24 LUTs to correct the 16 data bits.

D. Per-Bit Joined-Pattern Correction With Seven Parity Bits

The optimization is more complicated when using seven parity bits. As an example, we will next analyze the (39, 32) SEC–DAEC code from [9], with 32 data bits and 7 parity bits (see Fig. 4).

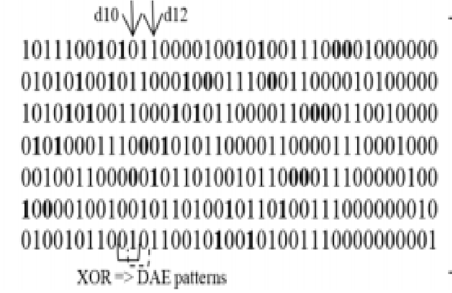


Fig. 4: Check matrix for an SEC–DAEC (39, 32) code

The main difference with the previous case is that the syndrome no longer fits into a single six-input LUT. The conventional implementation for (39, 32) would be similar to the design shown in Fig. 2 for (22, 16), but with a fifth input in LUT4 corresponding to the additional bit from the syndrome (and modifying the associated circuitry). This implies that 4 LUTs are required to correct each bit, up to 128 LUTs for the 32 data bits. In the real implementation. The per-bit joined-pattern correction. With the six-input LUT restriction commented above, the proposed design requires the value of a syndrome bit to be moved to the second LUT.

The value of this bit must be shared by the three patterns of that data bit (the SE and the two double error syndromes). For example, in column 11 of Fig. 4, corresponding to the SE pattern for the 11th bit (d10), the fifth bit is 0 (0110001). At the same time, the DAE patterns for the 11th bit (XOR of the adjacent columns, marked in Fig. 4) have also a “0” in this fifth row (1101001 and 1010011). Fig. 5 shows the resulting design for column 11 (bit d10) by moving syndrome bit s4 (fifth bit) to LUT3. An AND gate is added to check the output of the first LUT and the expected shared value of s4. A NOT gate is needed only when the expected value of the

© 2017 IJESAR. All Rights Reserved.

syndrome bit (s4 in this case) is 0. Every column in this matrix (Fig. 4) shares at least one bit value for the three patterns (marked in bold). There is a second optimization that can be applied when the same bit is moved to the second LUT for two different columns.

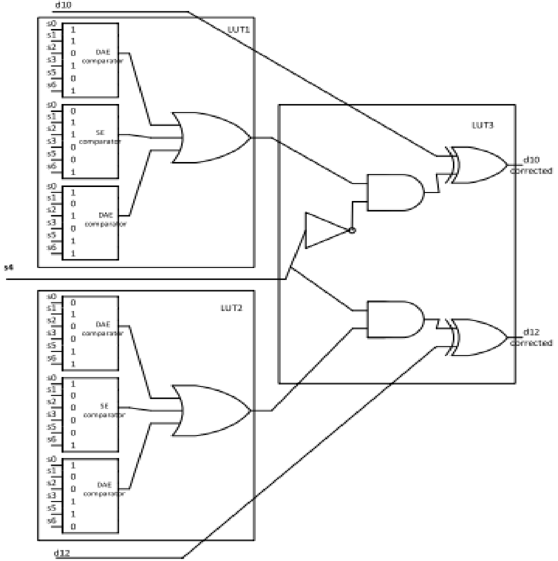


Fig. 5: SEC–DAEC (39, 32) per-bit joined-pattern correction optimization for the bits d10 and d12.

In Fig. 5, columns 11 and 13 place the fourth syndrome bit in the second LUT. Rearranging the design so that these bits are checked together, it is possible to allocate the second step (additional syndrome bit check and correction) of the process for both bits in a single five-input LUT (LUT3 in Fig. 5). Inputs are the data bits to be corrected (d10 and d12), the syndrome bit that is shared by the three patterns of both columns (s4) and the outputs of the first comparators. Three LUTs are needed for every two data bits to perform the correction. Consequently, 48 LUTs are enough to correct the 32 data bits.

E. **Multibit Joined-Pattern Correction with Six Parity Bits**

It is possible to introduce an extra optimization to the six-bit parity codes. It is based on the possibility of moving a bit to the second-step LUT, together with the fact that only four inputs were used in the correction LUT (LUT3) shown in Fig. 3. Looking at the matrix presented in Fig. 1, all three patterns for the second column (d1) share a “0” in the second bit (001111, 100011, and 101000). This bit (s1) can be shifted to the second LUT using the same idea that was presented for seven-bit parity codes. Again, the circuit to correct the data can be rearranged, so that those data bits moving the same syndrome bit can be placed together. In this scenario, not only the second LUT can be merged, but also the first LUT, which only has five inputs now, can be shared by both columns, as shown in Fig. 6, for the second and fourth columns (d1 and d3). With this multi bit joined-pattern correction approach, only two LUTs are used for every two bits.

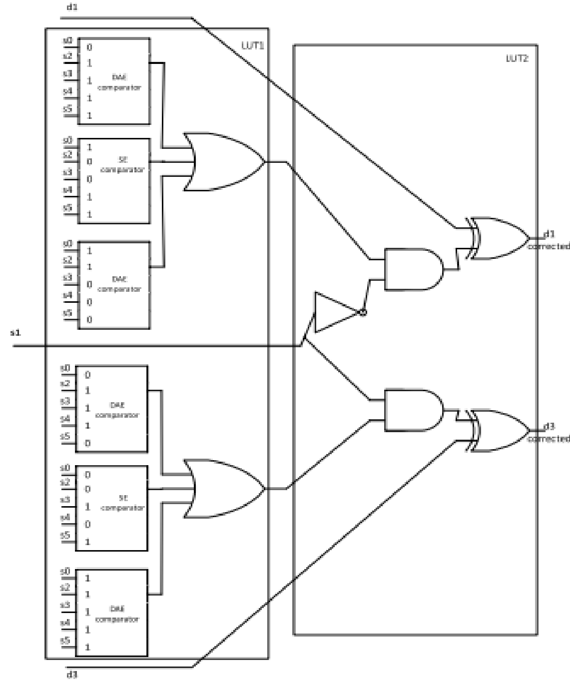


Fig. 6: SEC–DAEC (22, 16) multi bit joined-pattern correction optimization for the bits d1 and d3.

Looking at the matrix shown in Fig. 1, all the columns have at least one bit that is shared between single and DAE patterns except the seventh one (000111, 110110, and 001001). In addition, column scan be placed in groups of two, based on the same shared bit being chosen. As pointed out, the seventh column cannot be optimized, and, as we have an even number of data columns, another column also stays in the first optimization design, with three LUTs to perform the decoding. Moving the bits from the first LUT to the second one implies that the equation of both LUTs changes. A careful analysis needs to be done to avoid introducing errors in the equations.

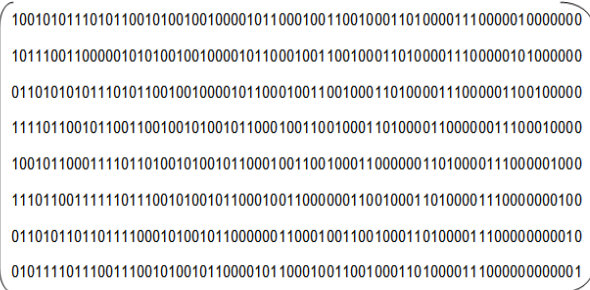
3. PROPOSED DESIGN

A. Per-Bit Joined-Pattern Correction With Eight Parity Bits

We can use the (72, 64) SEC–DAEC code presented in [9], with 64 data bits and 8 parity bits. The pre-Bit

© 2017 IJESAR. All Rights Reserved.

Joined –Pattern Correction with Eight Parity Bits is shown in Fig8.This scenario is more difficult to optimize due to these eight parity bits.



100101011101011001010010010000101100010011001000110100001110000010000000
101110011000001010100100100001011000100110010001101000011100000101000000
011010101011101011001001000010110001001100100011010000111000001100100000
111101100101100110010010100101100010011001000110100001100000001110010000
100101100011110110100101001011000100110010001100000011010000111000001000
111011001111110111001010010110001001100000011001000110100001110000000100
011010110110111100010100101100000011000100110010001101000011100000000010
010111101110011100101001011000010110001001100100011010000111000000000001

Fig. 7: Check matrix for an SEC–DAEC (72, 64) code

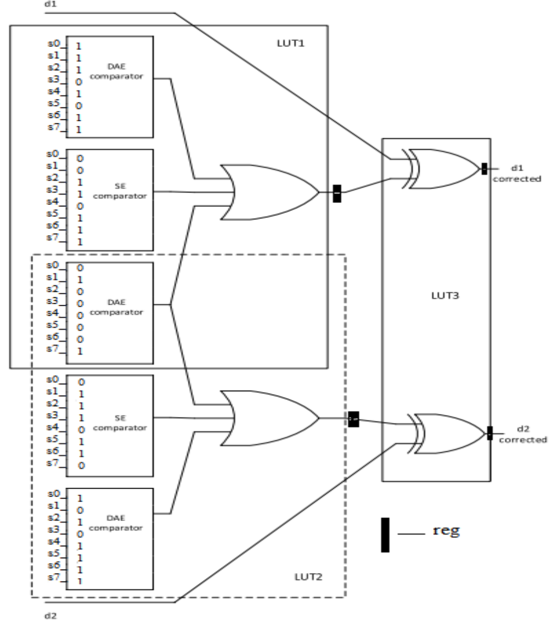


Fig. 8: SEC–DAEC (72, 64) per-bit joined-pattern correction optimization for the bits d11and d12.

The per-bit joined-pattern correction can still be done, but it can only be applied to columns with two bits that can be moved to the second LUT (those that share the same value for the three patterns). There are less columns in these codes that fulfill this requirement. In addition, the second step cannot be merged into a single LUT for two different data bits, as there would be six inputs (two from the first stage LUTs, two from the data bits to be corrected, and two from the shared syndrome bits) in this second stage instead of five. Here we can use pipelining concept to reduce the delay of the circuit. In this proposed design we can use registers at the outputs of the or gate operation and also final results. Future work will focus on the identification of additional strategies for codes with 64-bit or wider word sizes.

4. SIMULATION AND SYNTHESIZE RESULTS

Fig. 9: SEC–DAEC (72, 64) per-bit joined-pattern correction optimization for the bits d10 and d12 Simulation Wave Forms.

Synthesize result:-

	SEC–DAEC (39, 32) per-bit joined-pattern	SEC–DAEC (72, 64) per-bit joined-pattern
Delay	**6.209 ns**	**4.283**

5. CONCLUSION

In this brief, we have presented a method to optimize the decoders for SEC–DAEC codes in FPGAs in terms of resource utilization by taking advantage of the specific characteristics of this type of codes, and the special features included in modern FPGAs. The implementation results for two codes, a (39, 32) SEC–DAEC and a (72, 64) SEC–DAEC, show that the proposed designs reduce the resource utilization of the correction circuit when comparing it with the conventional designs. Although only SEC–DAEC codes have been studied, other codes that correct more than two adjacent errors (e.g., triple adjacent error correction) may also benefit from this strategy. It can be applied to other FPGAs with a different LUT configuration (such as Altera Stratix eight-input fracturable LUTs) protecting different code widths (e.g., the one with eight parity bits). Future work will address this possibility and the utilization of different FPGAs to compare the results and enhance the optimization of codes with wider words. Synthesis algorithms in future tool releases from FPGA manufacturers may address this optimization, considering that there is a compromise between the runtime in executing the algorithm and the global optimization achieved. The development of a more formal framework and application will also be covered in the future.

© 2017 IJESAR. All Rights Reserved.

REFERENCES

1. Xilinx Solutions for Aerospace & Defense Applications, accessed on Apr. 27, 2016. [Online]. Available: http://www. xilinx.com/esp/mil_aero/collateral/presentations /xsol_aero_def_appl.pdf
2. Land and J. Elliot, (2009) "Architecting ARINC 664, Part 7 (AFDX) solutions," Xilinx, Inc., San Jose, CA, USA, Appl. Note XAPP1130 (v1.0.1), accessed on Apr. 27, 2016. [Online]. Available: http://www. xilinx.com/support/documentation/application_n otes/xapp1130.pdf
3. Fiethe, H. Michalik, C. Dierker, B. Osterloh, and G. Zhou, "Reconfigurable system-on-chip data processing units for space imaging instruments," in Proc. Design, Autom. Test Eur. Conf., vol. 1. 2007, pp. 1–6.
4. S. Baeg, S. Wen, and R. Wong, "SRAM interleaving distance selection with a soft error failure model," IEEE Trans. Nucl. Sci., vol. 56, no. 4, pp. 2111–2118, Aug. 2009.
5. R. K. Lawrence and A. T. Kelly, "Single event effect induced multiple-cell upsets in a commercial 90 nm CMOS digital technology," IEEE Trans. Nucl. Sci., vol. 55, no. 6, pp. 3367–3374, Dec. 2008.
6. C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, Mar. 1984.
7. S. Lin and D. J. Costello, Jr., Error Control Coding, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall. 2004.
8. A. Neale and M. Sachdev, "A new SEC-DED error correction code subclass for adjacent MBU tolerance in embedded memory," IEEE Trans. Device Mater. Rel., vol. 13, no. 1, pp. 223–230, Mar. 2013.
9. A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. 25th IEEE VLSI Test Symp., May 2007, pp. 349–354.
10. 7 Series FPGAs Configurable Logic Block. User Guide, Xilinx, Inc., San Jose, CA, USA, Nov. 2014.
11. "FPGA architecture," Altera Corp., San Jose, CA, USA, White Paper WP-01003-1.0 (v 1.0), Jul. 2006.
12. A. H. Farrahi and M. Sarrafzadeh, "Complexity of the lookup-table minimization problem for FPGA technology mapping," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 13, no. 11, pp. 1319–1332, Nov. 1994.
13. J. H. Anderson, Q. Wang, and C. Ravishankar, "Raising FPGA logic density through synthesis-inspired architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 3, pp. 537–550, Mar. 2012.
14. A. Mishchenko, A. S. Chatterjee, and R. K. Brayton, "Improvements to technology mapping for LUT-based FPGAs," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 26, no. 2, pp. 240–253, Feb. 2007.
15. Y. Hu, V. Shih, R. Majumdar, and L. He, "FPGA area reduction by multi-output function based sequential resynthesis," in Proc. 45th ACM/IEEE Design Autom. Conf., Jun. 2008, pp. 24–29.
16. Y.-Y. Liang, T.-Y. Kuo, S.-H. Wang, and W.-K. Mak, "ALMmap: Technology mapping for FPGAs with adaptive logic modules," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 31, no. 7, pp. 1134–1139, Jul. 2012.

Authors Details:

	Shaik Khadar Basha has completed his B.Tech in Electronics and Communication Engineering from Malineni Lakshmaiah Engineering College, Singarayakonda, JNTUK affiliated college in 2013. He is pursuing his M.Tech in VLSI and Embedded Systems from QIS Institute of Technology, Ongole, JNTUK affiliated college.
	K.S.N. Vittal currently working as Assistant Professor, Department of E.C.E, QIS Institute of Technology, Ongole. He has more than 6 years of teaching experience. He received the B.Tech degree in Electronics & Communications Engineering from Jawaharlal Nehru Technological University in 2008. He received M.TECH degree from K L University in 2012. His research interests include Analog design and Low power design.

© 2017 IJESAR. All Rights Reserved.