ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Multioperand Redundant Adders on FPGAs

Kashivisalakshi.T
M.Tech Student,
Department of ECE,
KITS for Women's, kodad, T.S, India.

MS.B.Jyothirmayee
Associate Professor,
Department of ECE,
KITS for Women's, kodad, T.S, India.

ABSTRACT:

Although redundant addition is widely used to design parallel multi operand adders for ASIC implementations, the use of redundant adders on Field Programmable Gate Arrays (FPGAs) has generally been avoided. The main reasons are the efficient implementation of carry propagate adders (CPAs) on these devices (due to their specialized carry-chain resources) as well as the area overhead of the redundant adders when they are implemented on FPGAs. This paper presents different approaches to the efficient implementation of generic carry-save compressor trees on FPGAs. They present a fast critical path, independent of bit width, with practically no area overhead compared to CPA trees. Along with the classic carry-save compressor tree, we present a novel linear array structure, which efficiently uses the fast carry-chain resources. This approach is defined in a parameterizable HDL code based on CPAs, which makes it compatible with any FPGA family or vendor. A detailed study is provided for a wide range of bit widths and large number of operands. Compared to binary and ternary CPA trees, speedups of up to 2.29 and 2.14 are achieved for 16-bit width and up to 3.81 and 3.11 for 64-bit width.

Keywords:

Computer arithmetic, reconfigurable hardware, compressor trees, multi operand addition, redundant representation, carry-save adders.

1. INTRODUCTION:

For years, the use of dedicated decimal hardware has been limited to mainframe business computers [25, 6, 4] and handheld calculators at the low end [7]. Very recently a renewed interest in providing hardware acceleration for decimal arithmetic has emerged [9]. It has been boosted by the numerically intensive computing requirements of new commercial, financial and Internet applications, such as e-commerce and e-banking. Because of the perspectives of a more widespread use of decimal processing, the revised IEEE 754-2008 Standard for Floating-Point [19] incorporates a specification for decimal arithmetic. Besides, the advances in FPGA technology have opened up new opportunities to implement efficient floating-point coprocessors for specialized tasks [28] for a fraction of the cost of an ASIC implementation. Thus, even though most of the current research on decimal arithmetic is targeted at high-performance VLSI design [10, 12, 14, 20, 21, 26, 31], a few works present implementations of decimal arithmetic units for FPGAs [3]. The design of decimal units for FPGAs faces several challenges: first, the inherent inefficiency of decimal representations in systems based on two-state logic and a complex mapping of the decimal arithmetic rules into boolean logic. On the other hand, the special built-in characteristics of FPGA architectures make it difficult to use many well-known methods to speedup computations (for example, carry-save and signed-digit arithmetics). Therefore, it may be preferable to develop specific decimal algorithms more suitable for FPGAs rather than adapting existing ones targeted for ASIC platforms. In this context, we present the algorithm, architecture and FPGA implementation of a novel unit to perform fast addition of a large amount of decimal (BCD) fixed-point or integer operands. This operator is also of key importance for other arithmetic operations such as decimal multiplication and division. Among the novel contributions of this work are the following:

2. METHODOLOGY:

The design of portable devices requires consideration for peak power consumption to ensure reliability and proper operation.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

However, the time averaged power is often more critical as it is linearly related to the battery life. There are four sources of power dissipation in digital CMOS circuits: switching power, short-circuit power, leakage power and static power. The following equation describes these four components of power

$$P_{avg} = P_{switching} + P_{short-circuit} + P_{leakage} + P_{static}$$
$$= \alpha C_L V_{dd} V_s f_{ck} + I_{sc} V_{dd} + I_{leakage} V_{dd} + I_{static} V_{dd}$$

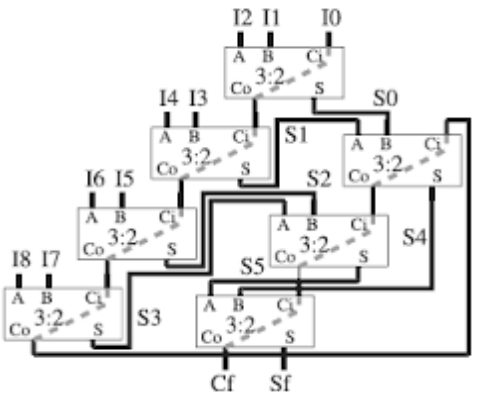
low voltage:

Power consumption is linearly proportional to voltage swing (Vs) and supply voltage (Vdd) as indicated in Eq. (2.5). For most CMOS logic families, the swing is typically rail-to-rail. Hence, power consumption is also said to be proportional to the square of the supply voltage, Vdd Therefore, lowering the Vdd is an efficient approach to reduce both energy and power, presuming that the signal voltage swing can be freely chosen. This is, however, at the expense of the delay of circuits. The delay, td, can be shown to be proportional to. The exponent is between 1 and 2. It tends to be closer to 1 for MOS transistors that are in deep sub-micrometer region, where carrier velocity saturation may occur. increases toward 2 for longer channel transistors. The current technology trends are to reduce feature size and lower supply voltage. Lowering Vdd leads to increased circuit delays and therefore lower functional throughput.

Smaller feature size, however, reduces gate delay, as it is inversely proportional to the square of the effective channel length of the devices. In addition, thinner gate oxides impose voltage limitation for reliability reasons. Hence, the supply voltage must be lowered for smaller geometries. The net effect is that circuit performance improves as CMOS technologies scale down, despite of the Vdd reduction. Therefore, the new technology has made it possible to fulfil the contradicting requirements of low power and high throughput In this Section we present prior work on binary and decimal multi-operand addition and analyse different representative FPGA implementations, discussing their associated advantages and costs. Based on the conclusions extracted from this survey, in Section 3 we present a proposal that leads to more efficient implementations of decimal multi-operand adders in FPGA platforms.

3. IMPLEMENTATION:

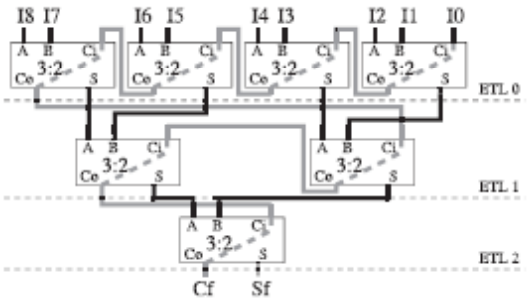
In this section, we present different approaches to efficiently map CS compressor trees on FPGA devices. In addition, approximate area and delay analysis are conducted for the general case. Let us consider a generic compressor tree of NOP input operands with N bit width each. We also assume the same bit width for input and output operands. Thus, input operands should have previously been zero or sign extended to guarantee that no overflow occurs3.1. Regular CS compressor tree design The design of a multi operand CS compressor tree attempts to reduce the number of levels in its structure. The implementation of a generic CS compressor tree requires [NOP/2]-1 4:2 compressors (because each one eliminates two signals), whereas a carry-propagate tree uses NOP-1 CPAs (since each one eliminates one signal) .If we bear in mind that a 4:2 compressor uses practically double the amount of resources as CPAs [28],both trees basically require the same area. On the other hand, the speed of a compressor tree is determined by the number of levels required. In this case, because each level halves the number of input signals, the critical path delay (D) is approximately L4:2=[log2(Nop)]-1 (1) D=L4:2 . d4:2 (2) Where L4:2 is the number of levels of the compressor tree and d4:2 is the delay of a 4:2 compressor level (including routing). We now generalize this idea to compressors of any size by proposing a different approach based on linear arrays. This reduces the critical path of the compressor tree when it is implemented on FPGAs with specialized carry-chains. 3.2 Linear Array Structure In the previous approach, specialized carry resources are only used in the design of a single 4:2 compressor, but these resources have not been considered in the design of the whole compressor tree structure. However, in our case, given the two output words of each adder (sum-word and carry-word), only the carry-word is connected from each CSA to the next, whereas the sum words are connected to lower levels of the array



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Fig. 1 shows an example for a 9:2 compressor tree designed using the proposed linear structure, where all lines are N bit width buses, and carry signal are correctly shifted. For the CSA, we have to distinguish between the regular inputs (A and B) and the carry input (Ci in the figure), whereas the dashed line between the carry input and output represents the fast carry resources. With the exception of the first CSA, where Ci is used to introduce an input operand, on each CSA Ci is connected to the carry output (Co) of the previous CSA, as shown in Fig. 1.Thus, the whole carry-chain is preserved from the input to the output of the compressor tree (from I0 to Cf). First, the two regular inputs on each CSA are used to add all the input operands (Ii).

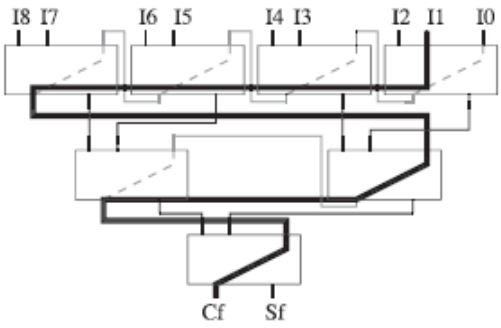
When all the input operands have been introduced in the array, the partial sum-words (Si) previously generated are then added in order (i.e. the first generated partial sums are added first) as shown in Fig. 1.In this way, we maximize the overlap between propagation through regular signals and carry-chains. Regarding the area, the implementation of a generic compressor tree based on N bit width CSAs requires Nop – 2 of these elements (because each CSA eliminates one input signal).Therefore, considering that a CSA could be implemented using the same number of resources as a binary CPA (as shown below), the proposed linear array, the 4:2 compressor tree, and the binary CPA tree have approximately the same hardware cost



Time model of the proposed CS 9:2 compressor tree.In relation to the delay analysis, from a classic point of view our compressor tree has Nop - 2 levels. This is much more than a classic Wallace tree structure and, thus, a longer critical path. Nevertheless, because we are targeting an FPGA implementation, we temporarily assume that there is no delay for the carry-chain path. Under this assumption, the carry signal connections could be eliminated from the critical path analysis and our linear array could be represented as a hypothetical tree, as shown in Fig. 2 (where the carry-chain is represented in gray).

To compute the number of effective time levels (ETL) of this hypothetical tree, each CSA is considered a 2:1 adder, except for the first, which is considered a 3:1 adder. Thus, the first level of adders is formed by the first [(Nop-1)/2] CSAs (which correspond to partial addition of the input operands). This first ETL produces [(Nop-1)/2]partial sum-words that are added to a second level of CSAs (together with the last input operand if Nop is even) and so on, in such a way that each ETL of CSAs halves the number of inputs to the next level. Therefore, the total ETLs in this hypothetical tree are L= [log2(Nop-1)] (3) and the delay of this tree is approximately L times the delay of a single ETL. The delay of the carry-chain is comparatively low, but not null. Let us consider just two global values for the delay: dcarry, which is the delay for the path between the carry inputs (Ci) of two consecutiveCSAs (see Fig. 2); and dsum, which is the delay from one general input of a CSA (Aor B) to a general input of a directly connected CSA, i.e., the time taken by the data to go from an ETL to the next one (see Fig. 2).

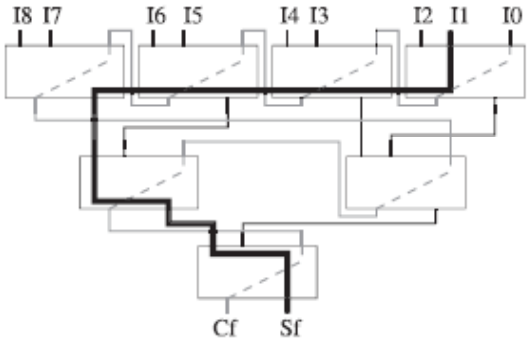
Even under this simplified scenario, it is unfeasible to obtain a general analytical expression for the delay of our compressor tree structure. On each ETL, the propagation through carry-chains and the general paths are overlapped and this overlap depends on multiple factors. First, it depends on the relative relationship between the values of dcarry and dsum (which is associated with the FPGA family used). Second, it depends on the number of operands that affect both the delay of the carry-chain of each ETL and the internal structure of the hypothetical tree. Even though the former could be expressed as an analytical formula, the latter cannot be expressed in this way (especially when Nop -1 is not a power of two). However, it is possible to bound the critical path delay by considering two extreme options.



Critical path of the proposed 9:2 compressor tree for linear array behavior.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

One extreme situation occurs when the delay of the whole carry-chain corresponding to each ETL (dcarry x the number of CSAs of the ETL) is always greater than the delay from an ETL to the next one (dsum). In this case, the timing behavior corresponds to a linear array and the critical path is represented in Fig. 3. Initially, the first carry out signal is generated from I1, I2, I3 in the first CSA and then the carry signal is propagated through the whole carry-chain until the output. Thus, the delay of the critical path has two components corresponding to the generation of the first carry signal and the propagation through the carry-chain. If we characterize the delay from a general input to the carry output in the first CSA (including later routing) as dsum, then the estimated lower bound for the delay of the compressor tree is Dlow=dsum+(Nop-3). dcarry (4) dsum is usually one order of magnitude greater than dcarry.



The other extreme situation occurs when the delay of the carry-chain on each ETL (dcarry x the number of CSAs of the ETL) is always less than the delay from an ETL to the next one (dsum). In this case, we obtain the hypothetical tree presented in Fig. 2. The critical path is shown in Fig. 4. It begins as in the previous case: A first carry generation from the general inputs and carry propagation through the carry-chain of the first ETL, because all internal general paths of the CSAs corresponding to the first ETL are updated in parallel and they need the carry input to generate the output signals. However, due to the initial premise, and because sum signals arrive at the first CSAs of each ETL earlier than at the last ones, after the first ETL the critical path goes from one ETL to the next one through the general routing. Therefore, the delay of the critical path has three components corresponding to the generation of the first carry signal, propagation through the carry-chain of the first ETL, and propagation across the remaining of the ETLs. In this case, taking into account that the delay between the carry input and the sum output (including later routing) is approximately dcarry, the estimated upper bound for the delay of the compressor tree is dsum x the number of ETLs + dcarry x the number of CSAs of the first ETL, that is Dup= [log2(Nop-1)].dsum+(Nop-1)/2.dcarry (5) This scenario is very frequent because if the delay through the carry-chain of the first ETL is less than dsum, then the next ETLs hold this condition.Thus,only the following condition: dsum > (6) needs to be fulfilled. Therefore, for values of Nop up to 2dsum/dcarry, the delay of the linear array compressor tree is very close to Dup. However, for greater values of Nop, the delay of the compressor tree is between Dlow and Dup, because the hypothetical structure of the compressor tree is a mix of both situations: The first CSAs form a linear array until the delay of the carry-chain in an ETL is lower than dsum, and then the remaining ones form a hypothetical tree. 3.3 High-Level Implementation A high-level description of an FPGA design using HDL presents some significant advantages, such as portability among different families (even from different brands), easier generalization, lower error production during the design process, and so on. A disadvantage is that control over the final implementation is lost, which could produce unexpected results. Thus, the HDL code needs to be carefully selected, especially when specific inner resources of the FPGA are used. In fact, a low-level design of the basic building block, which is instantiated from the higher level circuit, is sometimes the only way to achieve efficient implementations. Thus, the design is anchored to a specific FPGA inner structure.

4. DISCUSSION:

To measure the effectiveness of the designs presented in this paper, we have developed two generic VHDL modules implementing the proposed compressor tree structures: First, the linear array implemented by using CPAs (binary and ternary) and, second, the 4:2 compressor tree using the design of the compressor presented in [28]. Both modules provide the output result in CS format and allow the selection of different parameters such as: The number of operands (Nop), the number of bits per operand (N), and the basic building blocks (i.e., binary or ternary adder) for the linear array. For the purposes of comparison, similar modules, which implement classic adder tree structures based on binary CPAs and ternary CPAs, have also been developed. All these modules were simulated using Modelsim SE 6.3f and they were synthesized using Xilinx ISE 9.2, targeting Spartan-3A, Virtex-4, and Virtex-5 devices.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

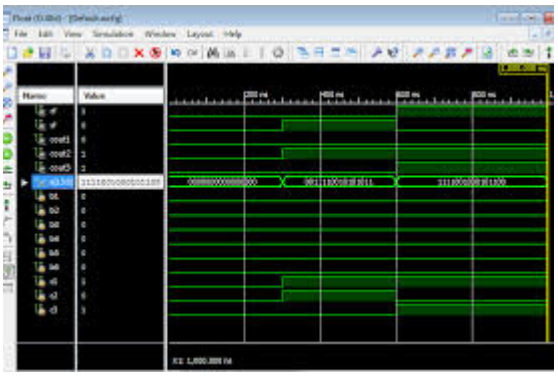
A generic ternary adder module was designed following the recommendations of Xilinx [46], because this adder is not automatically supported by ISE 9.2. Furthermore, to investigate their portability, compressor trees based onn-ternary CPAs were also synthesized to target the Altera Stratix-II family. In this case, the ternary adders are directlyinstantiated at a high level. We now summarize the main results obtained in this study. A. Results on FPGA Families with Support for Binary CPAs For the sake of simplicity, of the two FPGA families tested (Spartan-3A and Virtex-4), only the results corresponding to the Virtex-4 family are presented, because the results are very similar for both families.

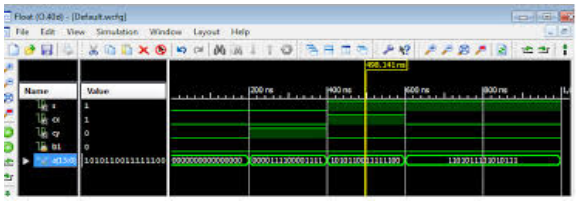
On these FPGAs, the compressor trees based on ternary adders are not efficiently implemented, and thus, we have only tested the ones based on binary adders. For purposes of clarity, let us denote as CPA tree the classic tree structure based on CPAs, OUR array the proposed linear array structure based on 3:2 CSAs, and 4:2 tree the classic tree structure based on a 4:2 compressors. Regarding the area, Fig. 8 shows the number of LUTs required by the different compressor tree structures when Varying Nop from 4 to 128 operands, for 16- and 64-bit widths.

With the exception of 4- and 5- operand compressor trees, which we consider separately, the area used for the three compressor trees is very similar and varies linearly with the number of operands and the bit width, as expected. Specifically, the area of CPA tree and OUR array is practically identical, whereas the 4:2 tree requires a little more area (up to 6 percent for a 16-bit width and up to 2 percent for a 64-bit width), due to the implementation of boundary bits on the 4:2 CA. Let us now consider the cases of four and five operands.

The CPAs involved in the implementation of OUR array are only 2- and 3-bit width for all operand sizes. Given this small size, the synthesis tool implements these CPAs by exclusively using LUTs, and not the specialized carry-chain, because this producesfaster circuits. As a consequence, there is an increase in area for these particular cases (as shown in Fig. 8), which could be eliminated by manually designing low-level CPAs or by changing the synthesis tool. On the other hand, this faster CPA implementation leads to more significant speedups for these cases, as shown in the following.

6. Experimental Results:





To measure the effectiveness of the designs presented in this paper, we have developed two generic VHDL modules implementing the proposed compressor tree structures: First, the linear array implemented by using CPAs (binary and ternary) and, second, the 4:2 compressor tree using the design of the compressor presented in [28]. Both modules provide the output result in CS format and allow the selection of different parameters such as: The number of operands (Nop), the number of bits per operand (N), and the basic building blocks (i.e., binary or ternary adder) for the linear array. For the purposes of comparison, similar modules, which implement classic adder tree structures based on binary CPAs and ternary CPAs, have also been developed. All these modules were simulated using Modelsim SE 6.3f and they were synthesized using Xilinx ISE 9.2, targeting Spartan- 3A, Virtex-4, and Virtex-5 devices. A generic ternary adder module was designed following the recommendations of Xilinx,because this adder is not automatically supported by ISE 9.2. Furthermore, to investigate their portability, compressor trees based on ternary CPAs were also synthesized to target the Altera Stratix-II family. In this case, the ternary adders are directly instantiated at a high level. For the sake of simplicity, of the two FPGA families tested (Spartan-3A and Virtex-4), only the results corresponding to the Virtex-4 family are presented, because the results are very similar for both families. On these FPGAs, the compressor trees based on ternary adders are not efficiently implemented, and thus, we have only tested the ones based on binary adders.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

For purposes of clarity, let us denote as CPA tree the classic tree structure based on CPAs, OUR array the proposed linear array structure based on 3:2 CSAs, and 4:2 tree the classic tree structure based on a 4:2 compressors.

5. Concluding Remarks:

Efficiently implementing CS compressor trees on FPGA, in terms of area and speed, is made possible by using the specialized carry-chains of these devices in a novel way. Similar to what happens when using ASIC technology, the proposed CS linear array compressor trees lead to marked improvements in speed compared to CPA approaches and, in general, with no additional hardware cost. Furthermore, the proposed high-level definition of CSA arrays based on CPAs facilitates ease-of-use and portability, even in relation to future FPGA architectures, because CPAs will probably remain a key element in the next generations of FPGA. We have compared our architectures, implemented on different FPGA families, to several designs and have provided a qualitative and quantitative study of the benefits of our proposals

6.ACKNOWLEDGMENTS:

I am Kashivisalakshi.T and would like to thank the publishers, researchers for making their resources material available. I am greatly thankful to Assistant Prof: Miss.B.Jyothirmayee for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

7.REFERENCES:

[1] B. Cope, P. Cheung, W. Luk, and L. Howes, "Performance Comparison of Graphics Processors to Reconfigurable Logic: ACase Study," IEEE Trans. Computers, vol. 59, no. 4, pp. 433-448,Apr. 2010.

[2] S. Dikmese, A. Kavak, K. Kucuk, S. Sahin, A. Tangel, and H. Dincer,"Digital Signal Processor against Field Programmable Gate ArrayImplementations of Space-Code Correlator Beamformer for SmartAntennas," IET Microwaves, Antennas Propagation, vol. 4, no. 5,pp. 593-599, May 2010.

[3] S. Roy and P. Banerjee, "An Algorithm for Trading off Quantization Error with Hardware Resources for MATLAB-based FPGA Design,"IEEE Trans. Computers, vol. 54, no. 7, pp. 886-896, July 2005.

[4] F. Schneider, A. Agarwal, Y.M. Yoo, T. Fukuoka, and Y. Kim,"A Fully Programmable Computing Architecture for MedicalUltrasound Machines," IEEE Trans. Information Technology inBiomedicine, vol. 14, no. 2, pp. 538-540, Mar. 2010.

[5] J. Hill, "The Soft-Core Discrete-Time Signal Processor Peripheral[Applications Corner]," IEEE Signal Processing Magazine, vol. 26,no. 2, pp. 112-115, Mar. 2009.

[6] J.S. Kim, L. Deng, P. Mangalagiri, K. Irick, K. Sobti, M. Kandemir,V. Narayanan, C. Chakrabarti, N. Pitsianis, and X. Sun, "AnAutomated Framework for Accelerating Numerical Algorithmson Reconfigurable Platforms Using Algorithmic/ArchitecturalOptimization," IEEE Trans. Computers, vol. 58, no. 12, pp. 1654-1667, Dec. 2009.

[7] H. Lange and A. Koch, "Architectures and Execution Modelsfor Hardware/Software Compilation and their System-LevelRealization," IEEE Trans. Computers, vol. 59, no. 10, pp. 1363-1377, Oct. 2010.

[8] L. Zhuo and V. Prasanna, "High-Performance Designs for LinearAlgebra Operations on Reconfigurable Hardware," IEEE Trans.Computers, vol. 57, no. 8, pp. 1057-1071, Aug. 2008.

[9] C. Mancillas-Lopez, D. Chakraborty, and F.R. Henriquez, "ReconfigurableHardware Implementations of Tweakable EncipheringSchemes," IEEE Trans. Computers,, vol. 59, no. 11, pp. 1547-1561, Nov. 2010.

[10] T. Guneysu, T. Kasper, M. Novotny, C. Paar, and A. Rupp,"Cryptanalysis with COPACOBANA," IEEE Trans. Computers,vol. 57, no. 11, pp. 1498-1513, Nov. 2008.

[11] I. Kuon and J. Rose, "Measuring the Gap between FPGAs andASICs," IEEE Trans. Computer-Aided Design of Integrated Circuitsand Systems, vol. 26, no. 2, pp. 203-215, Feb. 2007.

[12] M. Frederick and A. Somani, "Beyond the Arithmetic Constraint:Depth-Optimal Mapping of Logic Chains in LUT-Based FPGAs,"Proc. ACM/SIGDA Int'l Symp. Field Programmable Gate Arrays,pp. 37-46, 2008.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[13] T. PreuBer and R. Spallek, "Enhancing FPGA Device Capabilitiesby the Automatic Logic Mapping to Additive Carry Chains," Proc.Int'l Conf. Field Programmable Logic and Applications (FPL), pp. 318-325, 2010.

[14] S. Gao, D. Al-Khalili, and N. Chabini, "Implementation of LargeSize Multipliers Using Ternary Adders and Higher OrderCompressors," Proc. Int'l Con. Microelectronics (ICM), pp. 118-121, 2009.

[15] S. Gao, D. Al-Khalili, and N. Chabini, "FPGA Realization of HighPerformance Large Size Computational Functions: Multipliers andApplications," Analog Integrated Circuits and Signal Processing,vol. 70, no. 2, pp. 165-179, Feb. 2011.

Author's Details:

Ms.Kashivisalakshi.T. MTech student, in M.Tech Student, Dept of ECE in KITS for women's,kodad, T.S, India

MS.B.Jyothirmayee. working as a Assistant at ECE in KITS for women's,kodad, T.S, IndiaJNTUH Hyderabad. she has 3 years of UG/PG Teaching Experience