International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

Design of Multioper and Adders Using Different Compressors Based on FPGA

Mrs. B. Swetha
Assistant Professor, Department of VLSI(ECE), Sri Venkateshwara Engineering College,Suryapet,Nalgonda,India

Ms. V. Pallavi
PG Student, Department of VLSI(ECE),Sri Venkateshwara Engineering College,Suryapet,Nalgonda,India

Abstract-
Although redundant addition is widely used to design parallel multioper and 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.

Index Terms—Computer arithmetic; reconfigurable hardware; multioperand addition; redundant representation; carry-save adders

INTRODUCTION

THE use of Field Programmable Gate Arrays (FPGAs) to implement digital circuits has been growing in recent years. In addition to their reconfiguration capabilities, modern FPGAs allow high parallel computing. FPGAs achieve speedups of two orders of magnitude over a general-purpose processor for arithmetic intensive algorithms[1]. Thus, these kinds of devices are increasingly selected as the target technology for many applications, especially in digital signal processing [2], [3], [4], [5],hardware accelerators [6], [7], [8], cryptography [9], [10]and much more. Therefore, the efficient implementation ofgeneralized operators on FPGAs is of great relevance.The typical structure of an FPGA device is a matrix of configurable logic elements (LEs), each one surrounded by interconnection resources. In general, each configurable element is basically composed of one or several n-input lookup tables (N- LUT) and flip-flops. However, in modern FPGA architectures, the array of LEs has been augmented by including specialized circuitry, such as dedicated multipliers, block RAM, and so on. In [11], the authors demonstrate that the intensive use of these new element reduces the performance GAP between FPGA and ASIC implementations. One of these resources is the carry-chain system, which is used to improve the implementation of carry propagate adders

International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

(CPAs). It mainly consists of additional specialized logic to deal with the carry signals, and specific fast routing lines between consecutive LEs, as shown in Fig. 1. This esource is presented in most current FPGA devices from low-cost ones to high-end families, and it accelerates the carry propagation by more than one order of magnitude compared to its implementation using general resources. Apart from the CPA implementation, many studies have demonstrated the importance of using this resource to achieve designs with better performance and/or less area requirements, and even for implementing nonarithmetic circuits [12], [13].Multioperand addition appears in many algorithms, such as multiplication [14], [15], filters [16], [17], SAD [18], and others [1], [8], [19], [20], [21]. To achieve efficient
implementations of this operation, redundant adders are extensively used [22], [23]. Redundant representation reduces the addition time by limiting the length of the carry-propagation chains. The most usual representations are carry-save (CS) and signed-digit (SD). A CS adder (CSA)
adds three numbers using an array of Full-Adders (FAs), but without propagating the carries. In this case, the FA is usually known as a 3:2 counter. The result is a CS number, which is composed of a sum-word and a carry-word. Therefore, the CS result is obtained without any carry propagation in the time taken by only one FA. The addition of two CS numbers requires an array of 4:2 compressors, which can be implemented by two 3:2 counters. The conversion to non redundant representation is achieved by adding the sum and carry word in a conventional CPA [24]. However, due to the efficient implementation of CPAs, the use of redundant adders has usually been rejected when targeting FPGA technology. A direct implementation of a 3:2 counter usually doubles the area requirements of its equivalent CPA and improved speed is only noticeable for long bit widths. Nevertheless, several recent studies have demonstrated that redundant adders can be efficiently mapped on FPGA structures, reducing area overhead and improving speed, as described in Section 2. Despite the important advances represented by these previous studies, the solutions proposed require either (or sometimes both) the use of a sophisticated heuristic to generate each compressor tree or a low-level design. The latter impedes portability, because it is highly dependent on the inner structure.

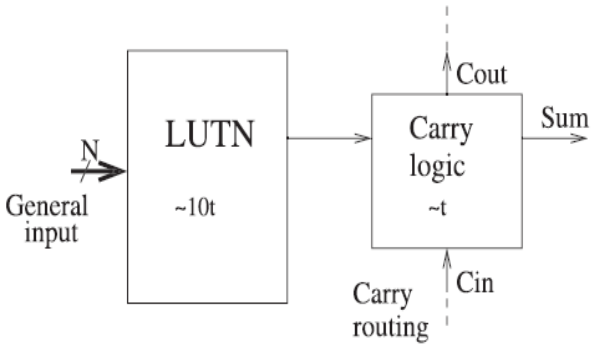


Fig. 1. General scheme of dedicated carry-chain resources included in modern FPGA devices.

In addition, their area and speed could be improved, because the use of a specialized fast carry-chain is very limited. In this paper, we study the efficient implementation of multioperand redundant compressor trees in modern FPGAs by using their fast carry resources. Our approaches strongly reduce delay and they generally present no area overhead compared to a CPA tree. Moreover, they could be defined at a high level based on an array of standard CPAs. As a consequence, they are compatible with any FPGA family or brand, and any improvement in the CPA system of future FPGA families would also benefit from them. Furthermore, due to its simple structure, it is easy to design a parametric HDL core, which allows synthesizing a compressor tree for any number of operands of any bit width. Compared to previous approaches, our design presents better performance, is easier to implement, and offers direct portability.
The rest of the paper focuses on CS representation, because the extension to SD

International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

representation could be simply achieved by inverting certain input and output signals from and to the compressor tree, as was demonstrated in [25]. Since it is unnecessary to make any internal changes to the array structure, these small modifications do not significantly modify compressor tree performance. The remainder of this paper is organized as follows: Section 2 reviews previous work on redundant addition on FPGAs. In Section 3, we present our proposals for implementing multioperand redundant compressor trees on FPGAs and a theoretical analysis of their performance. In Section 4, we compare the results of implementation using different approaches. Finally, the conclusions are presented in Section 5.

3.1 Regular CS Compressor Tree Design

The classic design of a multioperand CS compressor tree attempts to reduce the number of levels in its structure. The 3:2 counter or the 4:2 compressor are the most widely known building blocks to implement it [43]. We select a 4:2 compressor as the basic building block, because it could be efficiently implemented on Xilinx FPGAs [28]. 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

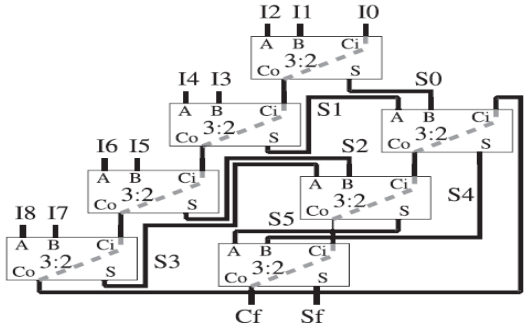


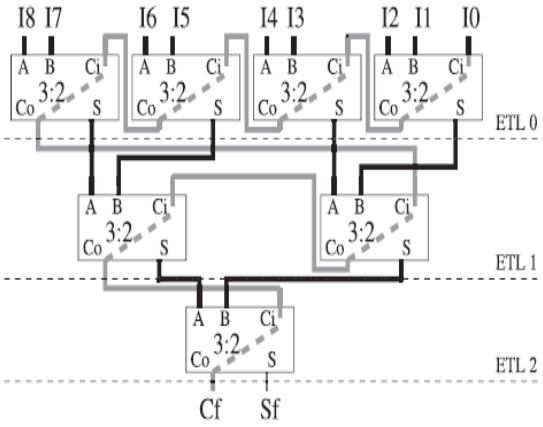
Fig. 2. N-bit width CS 9:2 compressor tree based on a linear array of csa's designed using the proposed linear structure, where all lines are N bit width buses, and carry signal are correctly shifted. For the CSA, we have to distinguish between the regular inputs (A and B) and the carry input (Ci in 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. 2.

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. 2. 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) [24]. 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.

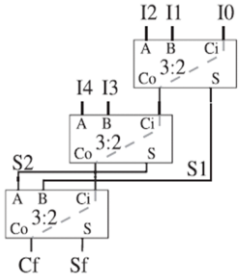
In relation to the delay analysis, from a classic point of view our compressor tree has Nop -2 levels. This is much more than a classic Wallace tree structure and, thus, a longer critical path. Nevertheless, because we are targeting an FPGA implementation, we temporarily assume that there is no delay 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. 3 (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.

International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

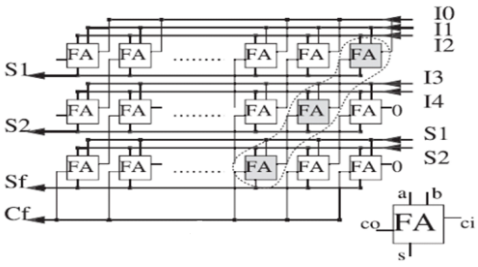
Thus, the first level of adders is formed by the first [(Nop _ 1)/2] CSAs (which correspond to partial addition of the input operands). This first ETL produces [(Nop _ 1)/2]partial sum-words that are added to a second level of CSAs together with the last input operand if Nop is even) and so on, in such 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 [log2(Nop _ 1)] and the delay of this tree is approximately L times the delay of a single ETL.



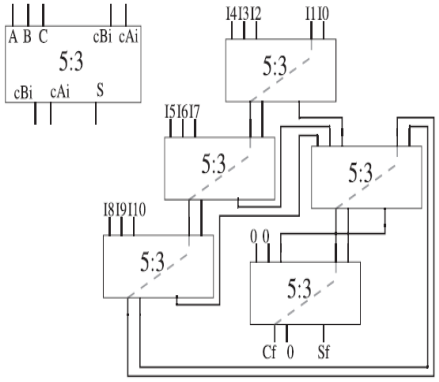
fg. 3. Time model of the proposed CS 9:2 compressor tree



CS 5:2 compressor tree based on a linear array



CS 5 a linear array carry save adder



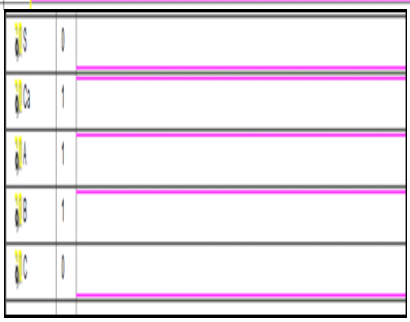
CS 11:2 compressor tree

IMPLEMENTATION RESULTS AND COMPARISON

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

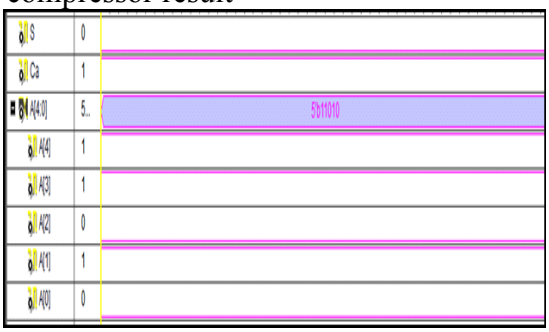
International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

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 [46], 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. We now summarize the main results obtained in this project.



11-2 compressor result
9-2 compressor result

5-2 compressor result



5-3 compressor result

APPLICATIONS AND ADAVANTAGES

ADVANTAGES:

- Reduces delay
- Reduces area
- Increases speed
- Direct portability
- Easy to implement

APPLICATIONS:

- Microchip manufacturing
- Power management
- Arithmetic units

CONCLUSION

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

International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

proposed CS linear array compressor trees lead to marked improvements in speed compared to CPA approaches and, in general, with no additional hardware cost.

FUTURESCOPE

Furthermore, the proposed high-level definition of CSA arrays based on CPAs facilitates eas ofuse and portability, even in relation to future FPGA architectures, because CPAs will probablremain 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.

REFERENCES

[1.] B. Cope, P. Cheung, W. Luk, and L. Howes, "Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case 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 Array Implementations of Space-Code Correlator Beamformer for Smart Antennas," 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 Medical Ultrasound Machines," IEEE Trans. Information Technology in Biomedicine.

[5.] J. Hill, "The Soft-Core Discrete-Time Signal Processor Peripheral [Applications Corner]," IEEE Signal Processing Magazine.

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

Mrs.B.SWETHA
ASSISTANT PROFESSOR
DEPARTMENT OF VLSI (ECE)

Ms.V.PALLAVI
M.TECH