

**HIGH SPEED MULTIOPERAND REDUNDANT ADDERS USING  
COMPRESSOR TREES**NALLA PETA.ANIL<sup>(1)</sup>, W. YASMEEN<sup>(2)</sup><sup>(1)</sup>*Pg Scholar, Vlsi System Design, Intellectual Institute Of Technology, Ap, India*<sup>(2)</sup>*Assistant Professor, Intellectual Institute Of Technology, Ap, India*

---

**Abstract-** Here we present 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, here 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.

---

**Index Terms**—Computer arithmetic, multioperand addition, redundant representation, carry-save adders

---

**I. 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. Therefore, the efficient implementation of generalized operators on

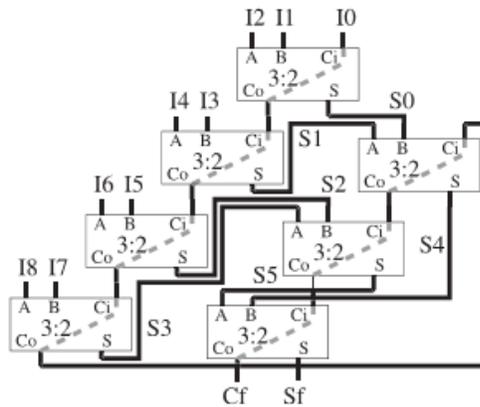
FPGAs are of great relevance. The typical structure of an FPGA device is a matrix of configurable logic elements (LEs), each one surrounded by interconnection resources. In general, each configurable element is basically composed of one or several n-input lookup tables (N- LUT) and flip flops. However, in modern FPGA architectures, the array of LEs has been augmented by including specialized circuitry, such as dedicated multipliers, block RAM, and so on. The intensive use of these new elements 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 (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 resource 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 non arithmetic circuits. Multioperand addition appears in many algorithms, such as multiplication, filters, SAD, and others. To achieve efficient implementations of this operation, redundant adders are extensively used. Redundant representation reduces the addition time by limiting the length of the carry-propagation chains. The most usual representation 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]. 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.

**II. CS COMPRESSOR TREES**

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. A more accurate Analysis for specific examples is provided in Section 4. 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 occurs. A detailed analysis of the number of leading guard bits required for multioperand CS addition.

**2.1 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 counters or the 4:2 compressors 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  $\lceil \text{Nop}/2 \rceil - 1$  4:2 compressors (because each one eliminates two signals), whereas a carry-propagate tree uses



1. N-bit width CS 9:2 compressor tree based on a linear array of CSAs. Nop-1CPAs (since each one eliminates one signal) [24]. 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.

$$L_{4:2} = \lceil \log_2(\text{Nop}) \rceil - 1 \tag{1}$$

$$D \sim L_{4:2} \cdot d_{4:2} \tag{2}$$

where  $L_{4:2}$  is the number of levels of the compressor tree and  $d_{4:2}$  is the delay of a 4:2 compressor level (Including routing). This structure is constructed assuming a similar delay for all paths inside each 4:2

Compressor. Nevertheless, in FPGA devices with dedicated carry resources, the delay from the carry Input to the carry output and the routing to the next carry input is usually more than one order of Magnitude faster than the rest of the paths involved in connecting two FAs (see Fig. 1). Thus, the connection of FAs through the carry-chain should be preserved as much as possible to obtain fast circuits. In fact, this is the idea behind the structure of the 4:2 compressor presented for Xilinx FPGA. 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.

**2.2 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. To optimize the use of the carry resources, we propose a compressor tree structure similar to the classic linear array of CSAs. 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. Fig. 2 shows an example for a 9:2 compressor tree designed using the proposed linear structure, where all lines are Nbit 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 (Cin 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  $C_{i0}$  is used to introduce an input operand, on each CSA  $C_{i1}$  is connected to the carry output ( $C_o$ ) 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  $I_0$  to  $C_f$ ). First, the two regular inputs on each CSA are used to add all the input operands ( $I_i$ ). 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.

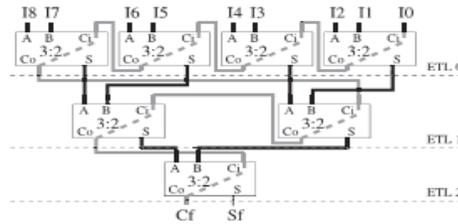


Fig 2. Proposed 9:2 compressor adder tree

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.

**2.3 Improvement for Ternary Adders**

To improve the performance of multioperand addition, the newest On these FPGAs, a ternary adder requires the same amount of resources as a simple 2-input adder while showing a similar speed. Since each ternary adder eliminates two operands, the number of adders required for a compressor tree is  $\lceil (Nop-1)/2 \rceil$ , which is almost half the amount needed in the binary case. On the other hand, the number of levels is

$$L3:1 = \lceil \log_3(Nop) \rceil, \quad (3)$$

which is considerably faster than the one based on binary adders. Therefore, the ternary adder is preferred to implement multi operand parallel addition when targeting these devices. We now present how our linear array compressor tree design is adapted to take advantage of this new resource.

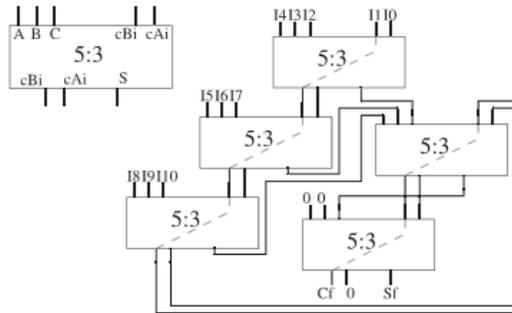


Fig.3. CS 11:2 compressor tree based on a linear array of 5:3 compressors.

This new compressor tree design (see example in Fig.7) could also be implemented at a high-level Description using ternary CPAs as a basic building block (see Fig. 6).n ternary adders of N Add-bit width Diagonally arranged are required to implement it (simplifying the extreme cases). Once again, the most Significant sum-bit of each ternary adder comprises the sum-word of the compressor tree, whereas the last cB out is the final carry-word. Except for the ending adders, each ternary adder sums one bit of each Operand and partial sum, varying the bit weight depending on its relative position.

**III. 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 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.

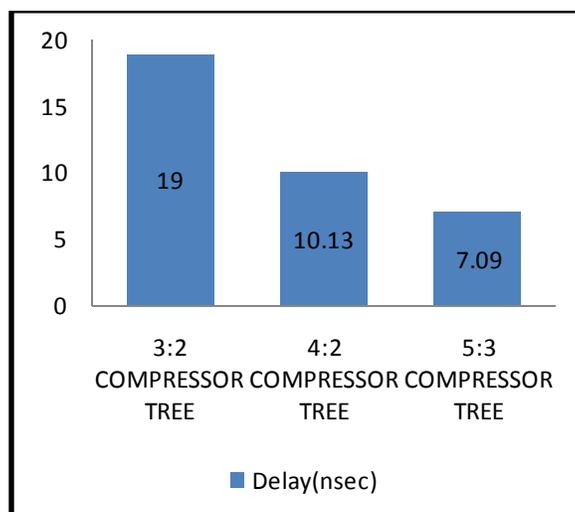


Fig.4 Delay comparison of compressor trees with 16 bit inputs

#### IV. CONCLUSIONS

Efficiently implementing CS compressor trees on FPGA, in terms of area and speed, is made possible by using the specialized carry-chains. While comparing among 3:2 and 4:2 and 5:3 compressor trees.

The proposed CS linear array compressor trees lead to marked improvements in speed compared to CPA approaches. 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.

#### REFERENCES

- [1] P. Brisk, A. Verma, P. Ienne, and H. Parandeh-Afshar, "Enhancing FPGA Performance for Arithmetic Circuits," Proc. ACM/IEEE 44th Design Automation Conf. (DAC '07), pp. 334-337, 2007.
- [2] S. Dormido and M. Canto, "Synthesis of Generalized Parallel Counters," IEEE Trans. Computers, vol. 30, no. 9, pp. 699-703, Sept. 1981.
- [3] P. Komerup, "Reviewing 4-to-2 Adders for Multi-Operand Addition," J. VLSI Signal Processing, vol. 40, pp. 143-152, 2005.
- [4] H. Parandeh-Afshar, A. Verma, P. Brisk, and P. Ienne, "Improving FPGA Performance for Carry-Save Arithmetic," IEEE Trans. Very Large Scale Integration Systems, vol. 18, no. 4, pp. 578-590, Apr. 2010.
- [5] M. Ortiz, F. Quiles, J. Hormigo, F. Jaime, J. Villalba, and E. Zapata, "Efficient Implementation of Carry-Save Adders in FPGAs," Proc. IEEE 20th Int'l Conf. Application-Specific Systems, Architectures and Processors (ASAP), pp. 207-210, 2009.
- [6] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Exploiting Fast Carry-Chains of FPGAs for Designing Compressor Trees," Proc. Int'l Conf. Field Programmable Logic and Applications (FPL), pp. 242-249, aug. 2009.
- [7] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Efficient Synthesis of Compressor Trees on FPGAs," Proc. Asia and South Pacific Design Automation Conf. (ASPDAC), pp. 138-143, 2008.
- [8] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Improving Synthesis of Compressor Trees on FPGAs via Integer Linear Programming," Proc. Int'l Conf. Design, Automation and Test in Europe (DATE '08), pp. 1256-1261, 2008.
- [9] Hadi Parandeh-Afshar, Philip Brisk, Paolo Ienne: "An FPGA Logic Cell and Carry Chain Configurable as a 6:2 or 7:2 Compressor". ACM Transactions on Reconfigurable Technology and Systems, Vol. 2, No. 3, Article 19, Pub. date: September 2009.
- [10] A. Cevrero, P. Athanopoulos, H. Parandeh-Afshar, A.K. Verma, H.S.A. Niaki, C. Nicopoulos, F.K. Gurkaynak, P. Brisk, Y. Leblebici, and P. Ienne, "Field Programmable Compressor Trees: Acceleration of Multi-Input Addition on FPGAs," ACM Trans. Reconfigurable Technology Systems, vol. 2, pp. 13:1-13:36, June 2009.

#### **ACKNOWLEDGMENT**

**FIRST AUTHOR: Nallapeta Anil** received the B.Tech degree in Electronics and Communication Engineering in the year 2012 and pursuing M.Tech degree in VLSI System design from Intellectual institute of technology. her area of interests includes Communication and VLSI Design

**SECOND AUTHOR: W.Yasmeen** received the B.Tech degree in Electronics and Communication Engineering and M.Tech degree in VLSI. Currently, She is working as Assistant Professor in the Department of Electronics and Communication Engineering, Intellectual institute of technology, Ananthapuramu. Having three years of teaching experience. Area of interest includes VLSI design, vhdl and verilog .