INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 2, Issue 3 OCT 2014

DESIGNING OF HIGH SPEED MULTI OPERAND ADDER TREE

YASWANTH SAGAR SILUMOJU

P.G.Scholor Vlsi-Design, Jothismathi College Of Engineering And Technology.

R.ANIL KUMAR

Guide And Coordinator, Asst.Prof, Ece Department, Jothismathi College Of Engineering And Technology

ABSTRACT—This paper describes a novel multiple-operand redundant binary adder .Redundant addition is widely used to design parallel multioperand 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.

1.INTRODUCTION

THEuse of Field Programmable Gate Arrays (FPGAs) to implement digital circuits has been growing in recentyears. 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 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. 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 nonredundant 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.

2. 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 Nbit 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 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

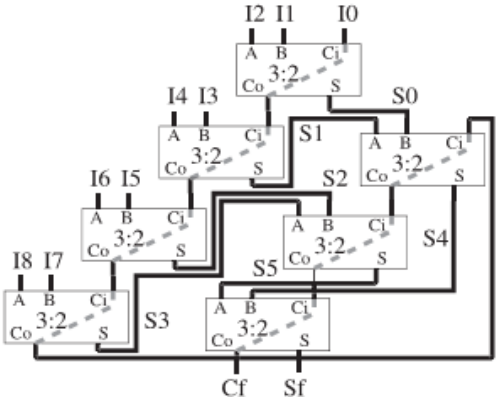

Fig. 1. N-bit width CS 9:2 compressor tree based on a linear array of CSAs.

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.

$$L_{4:2} = [\log_2(N_{op})] - 1 \quad (1)$$

$$D \sim L_{4:2} \cdot d_{4:2} \quad (2)$$

whereL4:2is the number of levels of the compressor tree and d4:2is 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 (AandB) 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 Ciis used to introduce an input operand, on each CSA Ciis 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.

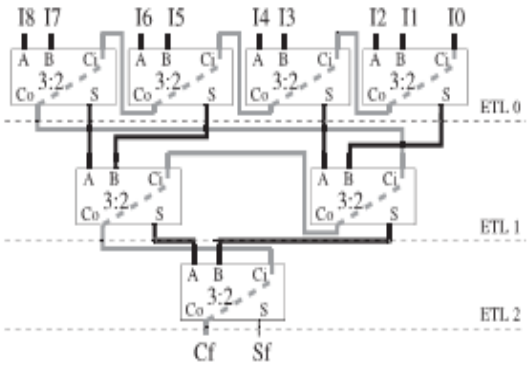

Fig 2.Proposed 9:2 compressor adder tree

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 Nbit 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.

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[(Nop-1)/2], which is almost half the amount needed in the binary case. On the other hand, the number of levels is

$$L3{:}1=[\text{Log}_3(N_{op})], \quad (3)$$

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.

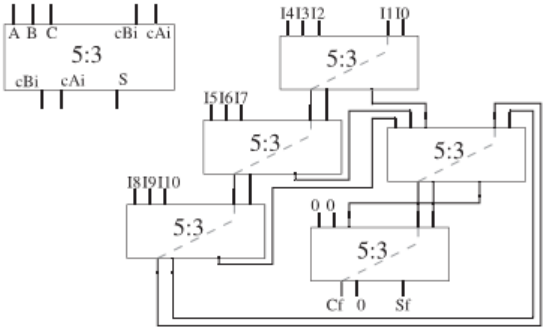

Fig. 3. CS 11:2 compressor tree based on a linear array of 5:3compressors.

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.

3.IMPLEMENTATION RESULTS AND COMPARISON

To measure the effectiveness of the designs presented inthis 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. 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.

4. CONCLUSIONS

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.

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,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/Architectural Optimization,"IEEE Trans. Computers, vol. 58, no. 12, pp. 1654-1667, Dec. 2009.

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

[8] 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.

[9] C. Mancillas-Lopez, D. Chakraborty, and F.R. Henriquez, "Reconfigurable Hardware Implementations of Tweakable Enciphering Schemes," 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.