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

Design of High speed multi-redundant adders using Compressors

Kagitha Bapuji
M.Tech Student,
Department of ECE,
NIMRA Institute of Science and Technology.

R.Veera Bhadraiah
Asst Professor,
Department of ECE,
NIMRA Institute of Science and Technology.

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 parameter is able 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.This work estimates the performance of the proposed designs in terms of delay, area are implemented in Xilinx ISE.

1. Introduction:

Area and power reduction in data path logic systems are the main area of research in VLSI system design. High-speed addition and multiplication has always been a fundamental requirement of high-performance processors and systems. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. The major speed limitation in any adder is in the production of carries and many authors have considered the addition problem.Multi-operand addition appears in many algorithms, such as multiplication 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 thecarry-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 carrypropagation 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.

2. CS COMPRESSOR TREES ON FPGAs:

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 multi-operand CS addition.

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

2.1 Regular CS Compressor Tree Design:

The classic design of a multi-operand 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, be efficiently implemented on Xilinx FPGAs [28]. The implementation of a generic CS compressor tree requires dNop= 2 e 1 4:2 compressors (because each one eliminates two signals), whereas a carry-propagate tree uses basic building block.

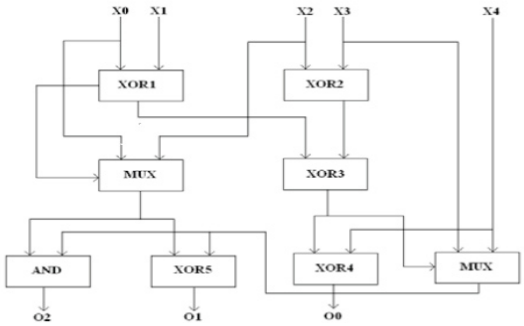


Figure 1:

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 (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 bð Nop 1 Þ=2 c CSAs (which correspond to partial addition of the input operands). This first ETL produces bð Nop 1 Þ=2 c 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 ¼ dlog2 ðNop 1 Þe ; ð3 Þ and the delay of this tree is approximately L times the delay of a single ETL. The other assumes a carry-in of 1, selecting which adder had the correct assumption via the actual carry-in yields the desired result.

One extreme situation occurs when the delay of the whole carry-chain corresponding to each ETL (dcarry 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. 4. 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 þ ðNop 3 Þ dcarry: ð4 Þ As mentioned, dsum is usually one order of magnitude greater than dcarry. Thus, the initial condition could be partially fulfilled for compressor trees with high Nop, but only in the first ETLs, because the number of CSAs on each ETL is nearly halved compared to the previous one. In fact, because the last ETL has only one CSA (ETLo in Fig. 2), this condition (i.e., in all ETLs) can only be completely fulfilled if d sum < dcarry, which is not possible on FPGAs.

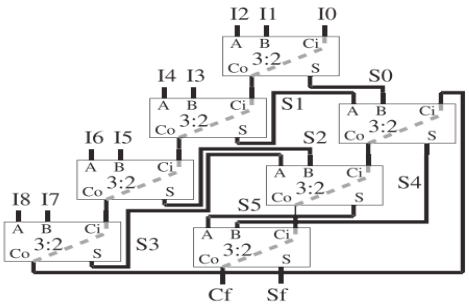


Figure 2: Regular Fixed Size CSLA

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 [24]. 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. 2 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. 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. Thus, the first level of adders is formed by the first bðNop 1 Þ=2 c CSAs (which correspond to partial addition of the input operands). This first ETL produces bðNop 1 Þ=2 c 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 ¼ dlog2 ðNop 1 Þe ; ð3 Þ and the delay of this tree is approximately L times the delay of a single ETL. H O R M I G O E T A L . : M U L T I O P E R A N D R E D U N D A N T A D D E R S O N F P G A S 2015 Fig. 2. N-bit width CS 9:2 compressor tree based on a linear array of CSAs. Fig. 3.

Time model of the proposed CS 9:2 compressor tree. However, 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 consecutive CSAs (see Fig. 3); and d , which is the delay from one general input of a CSA (A or 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. 3). 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 d carry 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.

One extreme situation occurs when the delay of the whole carry-chain corresponding to each ETL (dcarry 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. 4. 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.

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

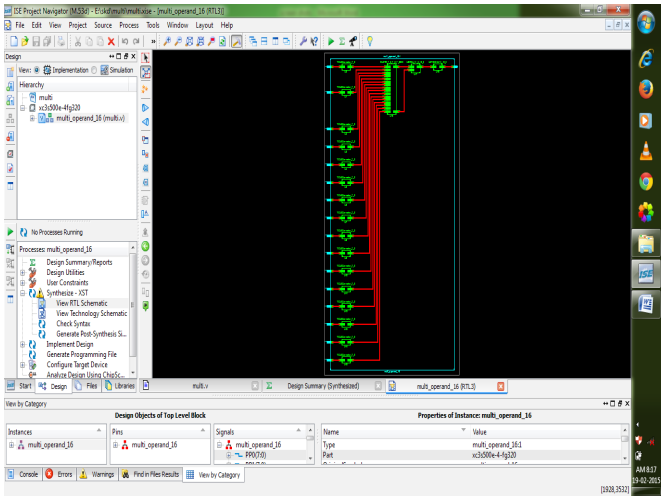
If we characterize the delay from a general input to the carry output in the first CSA (including later routing) as dsum, then the estimated lower bound for the delay of the compressor tree is Dlow dsum þ ðNop 3 Þ dcarry: ð4 Þ As mentioned, dsum is usually one order of magnitude greater than dcarry. Thus, the initial condition could be partially fulfilled for compressor trees with high Nop, but only in the first ETLs, because the number of CSAs on each ETL is nearly halved compared to the previous one. In fact, because the last ETL has only one CSA (ETLo in Fig. 2), this condition (i.e., in all ETLs) can only be completely fulfilled if dsum < dcarry, which is not possible on FPGAs.

Therefore, although (4) is an underestimated lower bound, it reflects the behavior of the first ETLs for high Nop, as described below. The other extreme situation occurs when the delay of the carry-chain on each ETL (dcarry 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. 3. The critical path is shown in Fig. 5. 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 nextone 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 the number of ETLs þ dcarry the number of CSAs of the first ETL, that is Dup dlog2 ðNop 1 Þe dsum þ Nop 1 2 dcarry: ð5 Þ 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 thefollowing condition: dsum >Nop 1 2 dcarry; ð6 Þ 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.

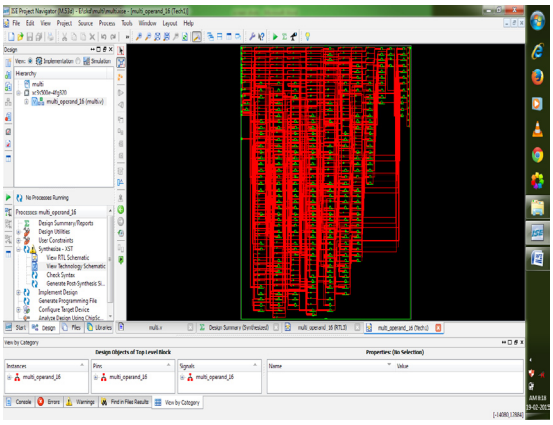
However, for greater values of Nop, the delay of the compressor tree is between Dlow and Dup, because the hypothetical structure of the compressor tree is a mix of both situations: The first CSAs form a linear array until the delay of the carry-chain in an ETL is lower than dsum, and then the remaining ones form a hypothetical tree. In any case, the behavior of the delay related to the number of operands has two additive components: 1) a discontinuous logarithmic variation due to the number of ETLs of the hypothetical tree; and 2)a linear variation 2 0 1 6 IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 10, OCTOBER 2013 Fig. 4. Critical path of the proposed 9:2 compressor tree for linear array behavior. Fig. 5. Critical path of the proposed 9:2 compressor tree for tree behavior. related to the propagation through either the carry-chain of the first ETL (for low Nop) or the linear array part (for high Nop). In addition, increasing Nop causes the growth of the last term, which could occasionally reduce the value of the first term, because first ETLs of the tree part could become part of the linear structure. As a result, logarithmic behavior is dominant for low values of Nop, whereas linear behavior prevails for high values. On the other hand, due to the interaction of these two terms, the curve of the delay in relation to Nop is smooth, instead of the increment by steps produced in a classic tree, as shown in Section 4. If we compare this approach to the one based on 4:2 compressors, the hypothetical tree of the former generally has one more level than the latter. In addition, the linear array compressor tree also has a delay associated with the linear component, whereas the 4:2 compressor tree does not.

3. SCREEN SHOTS:

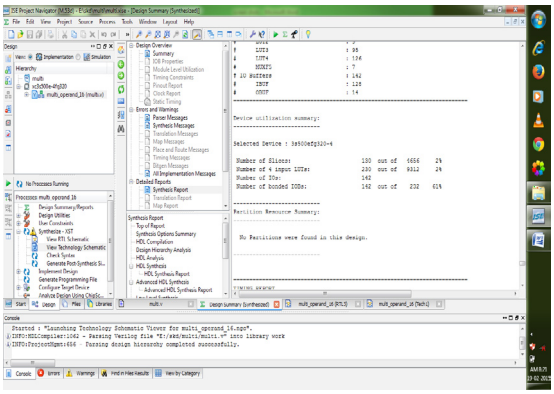


ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

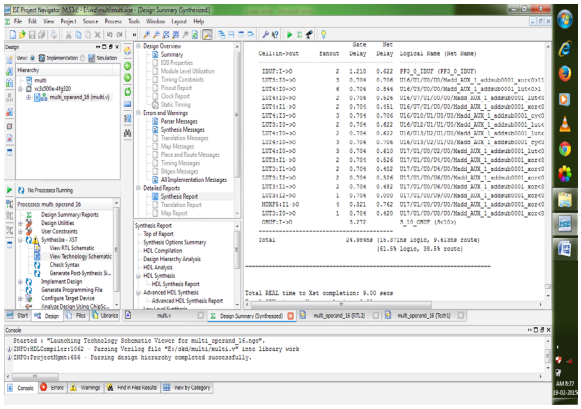
RTL SCHAMATIC:



TECHNOLOGY SCHAMATIC:



DESIGN UTILISATION SUMMARY:



DELAY:

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,July2005.

[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, "An Automated Framework for Accelerating Numerical Algorithms on Reconfigurable Platforms Using Algorithmic/ Architectural Optimization," IEEE Trans. Computers, vol. 58, no. 12, pp. 1654-1667, Dec.2009.

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

[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.

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