[1.]S. KANNAPPAN, [2.]S. Aruna MASTANI

A SURVEY ON MULTI-OPERAND ADDER

[1-2.]Jawaharlal Nehru Technological University (JNTUA), Ananthapuramu, INDIA

Abstract: The processors are required to perform computationally intense operations in the modern data world and an Arithmetic Unit (AU) is the heart of the processor that contributes for its performance. It's an ever ending research to optimize the AU w.r.t. architecture area and latency for improving the performance of the processor. Adder forms the basic Building block of any AU, and in particular addition of multiple operands is required in complicated arithmetic operations like multiplication, convolution, Transforms etc. Hence existing circuits for efficient Multi-operand adder in terms of circuit area and delay are discussed. The Multi-Operand Adders is optimized in many ways; the most popular methods are discussed in brief with respect to their performance. Memory-based computing is becoming an important approach to achieve fastness and cost effectiveness in contrast to conventional logical approach, and Distributed Arithmetic (DA) is a process of replacing logic elements with small memories or LUTs to optimize the circuit complexity. This paper presents various methods of Multi-Operand Adders and various optimizations for Multi-Operand Addition.
Keywords: carry save adder, compressor, generalized parallel counter, adder tree

INTRODUCTION TO OPTIMIZATION OF ARITHMETIC COMPUTATIONS

The optimization of Arithmetic Units are started with the adder more particularly Multi-Operand Adder in case of computationally intense applications. Most of the Multi operand adder implementations are based on optimizing either compressor logic for reduction in partial sum or the optimization of tree structure for reducing the carry propagation. The existing techniques are effective for addition of less number of operands only. The Area and delay of the circuit increase linearly for addition of large number of operands. The addition of large operands was implemented using bit partition technique. In 1973, S. Singh and R. Waxman described a scheme for multiple operand addition and multiplication [16], applying the bit-partitioning technique for adding 'K' number of operands of N bits each. In this scheme each partition contains m-bits, where $m = \lceil \log_2(K-1) \rceil$. The final sum is obtained in $m + 1$ addition cycles; the overall delay depends on delay of addition cycle. It is inefficient for addition of large number of operands.

In the tree structured adder the delay is reduced but area increased with number of operands. In serial addition of multi operands the number of clock cycles (called Latency) increase with increase in number of operands so, the existing implementations of multi operand adders are not effective in optimizing area and delay for large number of operands. The various techniques of Multi-Operand Adders using Tree structure is discussed in Section II & III, the Multi-Operand Adder using Generalized Parallel Counters is discussed in Section-IV.

COMPRESSOR TREE STRUCTURE OF ADDERS

The Multi-Operand Adders are generally implemented in two methods i.e Array Adders and Adder Tree structure. In Array Adder structure, two operands are added and output is added with third operand and continues the chain of addition until to get final sum output. It requires 'K' number of adder levels for addition of 'K' operands. But in case of Adder Tree structure the number of levels to add 'K' operands is less than that of Array Adders. It groups 'K' number of operands into sets of two operands. All the sets are added parallel in one level. The sum outputs from first level again grouped into sets of two operands and perform addition. This process continues until to get two operands and added in last level to obtain final sum. In each level it reduces number of operands to half. Therefore it requires $\log_2 K$ levels. The Adder Tree structure is faster than Adder Array structure with same resources consumed by both configurations. But the Array Adder is having regular routing than Adder Tree structure.

The Ripple Carry Adder (RCA) or Carry Look Ahead Adder (CLA) are two general Carry Propagate Adders used in the above methods i.e. Array Adder, Adder Tree is Carry Propagate Adder. The delay of their CPA depends on bit length of operand. For N-bit operand the of RCA proportional to N and for CLA it is proportional to$\log_2 N$. To reduce the delay these adders where implemented on FPGA by using dedicated carry chains [8]. The RCA on FPGA using fast carry chain is simpler than any other CPA topologies at an expense of high hardware cost [9]. The pipelining technique can be applied more effectively RCA [1]. The delay of Adder Tree using CPA is high due to carry propagation along the bit length.

Carry Save Adder tree is used as another approach for implementing Multi-Operand Adders. Here the carry is directly propagated to next level instead of propagating in the same as in case of CPA. The advantage of Carry Save Adder (CSA) tree is utilized in ASIC implementation due to flexible routing. The critical path delay can be minimized by optimizing the interconnection between Full Adders. But to implement on FPGA the Ripple Carry Adder tree is

preferred than CSA adder tree. When CSA tree is implemented on FPGA it become slower than RCA tree due to routing delay of CSA. However, a straightforward implementation on FPGAs [6] roughly requires double hardware than a carry-ripple adder, and does not exploit the fast carry chain to improve speed. This was partially solved by the compressors which compress the operands more than CSA tree [5], [7].

In 2005, R. D Kenney et al. [10] introduced and analyzed three techniques for performing fast operands addition [10]. Multi-operand adder designs are constructed and synthesized for 6 to 12 input operands. In 2005, J. Villalba et al. have studied the on–line addition of multiple operands for conventional, carry–save (CS) and/or Signed–Digit (SD) numbers [11]. They have proposed a new and key element called on-line Full Adder which is used in CSA tree to perform Multi-Operand Addition serially. The on-line Full Adder contains a 1-bit delay register at the sum bit output of FA. An external delay element also inserted at every skipped level in tree structure to maintain proper timing. The on-line Full Adder is used to reduce the hardware resources and cycle time.

In 2009, Manuel Ortiz et al. [12] have implemented 3:2 and 4:2 compressors using Dedicated carry chain in FPGA. When more than three operands are to be added the 4:2 compressors is preferred because it fully utilizes the logic of the slice and achieves high speed than 3:2 compressor with same resources. In 2009, William Kamp et al. [13] has implemented the basic 3:2 and 4:2 compressors using dedicated carry chain for redundant numbers to eliminate carry propagation. So that the delay is maintained nearly constant even for large operand width targeting low cost FPGAs.

In 2013 J. Harmigo et al. [14] was proposed a complete compressor tree using the dedicated fast carry chain unlike basic compressors developed in [12], [13]. They developed a linear array of carry save adder by mapping to preceding stages of FAs to conventional Ripple Carry Adders (RCA) or ternary adders. The delay is much smaller with equivalent resources compared to Adder Tree using RCA. Due to shorter carry chain it would be fastest choice for combinatorial multiple operand adders but it is inefficient as it is has irregular shaped compressor trees.

WALLACE TREE ADDER

In 2017, S.D. Thabah et al. [15] have analyzed different types of Multi-Operand adders in terms of propagation delay, power consumption and resource utilization. From the synthesis reports they have concluded that Wallace Tree Adders are the high speed and less power consuming circuit.

Even though the Carry Save Adder tree based on Carry Propagate Adders provide easy for implementation, Wallace tree adder is an efficient architecture for implementing Multi-Operand Adder tree which gives lowest propagation delay and least power consumption compared to other implementations even for increasing number of operands and bit size. However Wallace tree architecture is not a regular structure so, for large number of operands the circuit complexity increases along with routing Delay.

MULTI-OPERAND ADDERS USING GENERALIZED PARALLEL COUNTERS

The concept of using Compressor trees has a history in the design of arithmetic units for microcomputers since 1964. C. Wallace has proposed a compressor tree structure called Wallace tree [17] to optimize the multiplier, and L. Dadda in 1965 has given with the concept of counting number of ones in a column of binary bits by a tree structure called Dadda tree [18]. The concept of compressor is to avoid the slow carry propagation by passing the carry to the next compressor stage instead of propagating it within the same stage, which guarantees speed-up in Application Specific Integrated Circuits (ASICs) or custom ICs. The carry save arithmetic is not suitable for implementation on FPGA for a long time. As the CSA tree has large routing so that it become slower on FPGA because the dedicated carry chains available in modern FPGAs which are significantly faster than FPGA's routing fabric. However, compressors are implemented using dedicated carry chains as discussed in the previous sections it was first shown by Parandeh-Afshar et al. [7] in 2008 that a significant delay improvement can be obtained by using compressor trees on FPGAs by using the concept so called Generalized Parallel Counters (GPCs). The Multi-Operand Adder using GPCs provide a better utilization of the look-up tables (LUTs). They achieved delay reductions of about 30% while having a slight resource overhead of 5%. However, the design of the compressor tree is much more complex compared to the simple classic algorithms from Dadda [18] or Bickerstaff [21]. They provided a heuristic [7] as well as an exact integer linear programming (ILP) method [20].

In 2015, Burhan Khurshid et al. [22], implemented Generalized Parallel Counter (GPC) on Look up Tables. The Generalized Parallel Counters (GPCs) are used in constructing high speed compressor trees along with specialized fast carry chain in existing methods. The fast carry chain is eliminated from the previous existing GPC structure and has focused on achieving efficient mapping of GPCs on FPGAs by using only general Look-up table (LUT) fabric. The resulting structures are purely combinational and cannot be efficiently pipelined to achieve the potential FPGA performance. So the delay cannot be optimized for large number of operands.

In 2018 Martin Kumm et al. [25] was developed "Advanced compressor Tree Synthesis for FPGA" in which it proposed to combine Improved Generalized Parallel Counters and Integer Linear Programming to optimize the delay and resource utilization. The improved GPC is a combination of GPC for column compression and a 4:2 row compressor. This gives a better result in terms of delay and resource utilization. Therefore many approaches for Multi-Operand Adder were proposed in literature on Column compressor for counting one's in a column of bits, Generalized Parallel Counters (GPCs) also called multicolumn counters (different weighted column bits), which are also mapped with LUT's and some row compressor i.e 4:2 (four binary numbers reduced to two binary numbers) and optimization algorithms (Integer Linear Programming) for compressors.

CONCLUSIONS

For adding large number of operands, the Multi-Operand Adder using carry save arithmetic require more delay and complex hardware. Carry save addition is a row wise addition. To optimize area and delay column wise addition i.e compressors is an alternative approach. These compressors are generalized as Generalized Parallel Counters (GPC's) to add multiple operands in parallel. But choosing required number of GPC's and optimizing these GPC's is an overhead. Many optimization algorithms were combined with GPC's for better performance. And also these GPC's are combined with FPGA carry chain logic for efficient implementation on FPGA devices. Some of the techniques are combination of row compressors and mapping of GPC's on Look Up Tables in FPGA with optimization algorithms. In the recent techniques Distributed Arithmetic is adopted for implementing multi-Operand Addition in Filters where small distributed memory elements are used to store pre-computed values and achieving required value with additional hardware. These memory based implementations are well suited for both ASIC as well as FPGA. In future, Artificial Intelligence (AI) will become necessary for most the applications in the modern word. The AI requires a large number of calculations, mappings with less time as well as with smaller circuit chips. Memory based computing will become a choice for deep learning and Machine learning in AI.

References

[1] S. Yu and E. E. Swartzlander, "DCT implementation with distributed arithmetic", IEEE Transactions on Computers, vol. 50, no. 9, pp. 985–991, Sept. 2001.
[2] T.-S. Chang, C. Chen, and C.-W. Jen, "New distributed arithmetic algorithm and its application to IDCT," IEE Proceedings Circuits, Devices and Systems, vol. 146, no. 4, pp. 159–163, Aug. 1999.
[3] T.-S. Chang and C.-W. Jen, "Hardware-efficient implementations for discrete function transforms using LUT-based FPGAs," IEE Proceedings Circuits, Devices and Systems, vol.146, no. 6, pp. 309–315, Nov. 1999.
[4] F. de Dinechin, H. D. Nguyen and B. Pasca, Pipelined FPGA Adders, LIP Research Report no. ensl-00475780, Apr. 2010.
[5] J. Hormigo, M. Ortiz, F. Quiles, F. J. Jaime, J. Villalba and E.L. Zapata, Efficient Implementation of Carry-Save Adders in FPGAs, 20th IEEE international Conference on Application-Specific Systems, Architectures and Processors, pp. 207–210, Jul. 2009.
[6] P. M. Martinez, V. Javier, and B. Eduardo, On the design of FPGA-based Multioperand Pipeline Adders, XII Design of Circuits and Integrated System Conference, 1997.
[7] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Efficient Synthesis of Compressor Trees on FPGAs," in Asia and South Pacific Design Automation Conference (ASPDAC). IEEE, 2008, pp. 138–143.
[8] Xilinx Inc., Virtex-6 User Guide, 2009, http://www.xilinx.com/.
[9] S. Xing and W. H. Yu, FPGA Adders: Performance Evaluation and Optimal Design, IEEE Design and Test of Computers, vol. 15, no. 1, pp. 24–29, Jan.- Mar. 1998.
[10] R. D. Kenney and M. J. Schulte, "High-Speed Multioperand Decimal Adders", IEEE Transactions on Computers, vol. 54, no. 8, pp. 953-963, Aug. 2005.
[11] J. Villalba, J. Hormigo, J. M. Prades and E. L. Zapata, "On–line Multioperand Addition Based on On–line Full Adders*", in Proc. Int. Conf. on Application-Specific Systems, Architecture Processors (ASAP'05), pp. 322-327, 2005
[12] M. Ortiz, F. Quiles, J. Hormigo, F. J. Jaime, J. Villalba, and E. L. Zapata, "Efficient Implementation of Carry-Save Adders in FPGAs," in IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP), 2009, pp. 207–210.
[13] W. Kamp, A. Bainbridge-Smith, and M. Hayes, "Efficient Implementation of Fast Redundant Number Adders for Long Word- Lengths in FPGAs," in 2009 International Conference on Field- Programmable Technology (FPT). IEEE, 2009, pp. 239–246.
[14] J. Hormigo, J. Villalba, and E. L. Zapata, "Multioperand Redundant Adders on FPGAs," submitted to IEEE Transactions on Computers, vol. 62, no. 10, pp. 2013– 2025, 2013.
[15] S. D. Thabah; M. Sonowal and P. Saha ,"EXPERIMENTAL STUDIES ON MULTI-OPERAND ADDERS", INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS VOL. 10, NO. 2, JUNE 2017
[16] S. Singh and D. Waxman, "Multiple Operand Addition and Multiplication", IEEE Transactions on Computers, vol. C-22, no. 2, pp. 113-120, Feb. 1973.
[17] C. Wallace, "A Suggestion for a Fast Multiplier," IEEE Transactions on Electronic Computers, no. 1, pp. 14–17, 1964.
[18] L. Dadda, "Some Schemes For Parallel Multipliers," Alta Frequenza, vol. 45, no. 5, pp. 349–356, 1965.

[19] A. Omondi and B. Premkumar, Residue Number Systems: Theory and Implementation. Imperial College Press, 2007.
[20] A. R. Meo, "Arithmetic Networks and Their Minimization Using a New Line of Elementary Units," submitted to IEEE Transactions on Computers and currently under review, vol. C-24, no. 3, pp. 258–280, 1975.
[21] K.A.C. Bickerstaff, M. Schulte, and E.E. Swartzlander, "Reduced area multipliers," in Application-Specific Array Processors, 1993
[22] Suhas B. Shirol, S. Ramakrishna and Rajashekar B. Shettar, "Design and Implementation of Adders and Multiplier in FPGA Using ChipScope: A Performance Improvement", Information and Communication Technology for Competitive Strategies pp 11-19, 31 August 2018
[23] Duncan J. M. Moss , David Boland, and Philip H. W. Leong, "A Two-Speed, Radix-4, Serial–Parallel Multiplier", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, (Volume: 27 , Issue: 4 , April 2019) Page no. 769 – 777.
[24] Martin Kumm and Johannes Kappauf." Advanced Compressor Tree Synthesis for FPGAs", IEEE Transactions on Computers (Volume: 67 , Issue: 8 , Aug. 1 2018).

ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering
ISSN: 2067-3809