# A review of the present state of art in FPGA-Based Adders

Nasser Lotfivand [1,a], Mohd Nizar Hamidon [1,2,b], Maryam Mohd Isa [2,c], Nasri Sulaiman [2,d], Vida Abdolzadeh [3,e]

[1.] Institute of Advanced Technology, University Putra Malaysia, 43400 Serdang, Selangor, Malaysia
[2.] Department of Electronic Engineering, University Putra Malaysia, 43400 Serdang, Selangor, Malaysia
[3.] Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran
[a] Ahoora4444@hotmail.com , [b] mnh@eng.upm.edu.my , [c] misa@eng.upm.edu.my , [d] nasri@eng.upm.edu.my , [e] S.V.Abdolzadeh@Iaut.ac.ir

**Abstract:** Adders are the most fundamental arithmetic circuits that are used in processors and play key role in VLSI circuits. Power consumption and speed of these circuits are important quality factors for high performance integrated processing circuits. Floating-point operators, integer multipliers, and modular adders need large adders. On the other hand, Field programmable gate arrays (FPGAs) due to the excellent features such as low power consumption, flexibility, reusability, reasonable cost, easy upgrading, have become a favored platform for VLSI design. At this article recent advances and state of the art techniques in FPGA-Based Adders are reviewed.
[Lotfivand N, Hamidon MN, Isa MM, Sulaiman N, Abdolzadeh V. **A review of the present state of art in FPGA-Based Adders**. *Life Sci J* 2012;9(3):1234-1238] (ISSN:1097-8135). http://www.lifesciencesite.com. 175

**Keywords:** Adder; FPGA; FPGA-Based Adder

## 1. Introduction

Adders are building block in digital processing systems. Design and implementation of these circuits have received much research attention from designers. A large spectrum of architecture is available for adders such as ripple-carry, carry-look-ahead, carry-skip adders [1], [2]. Utilizing of each approach in design is depending on the application, throughput and latency. When base on the application the latency is not design major factor ripple-carry architecture can be used while the other two architectures are proper to perform high throughput with small latency. In some of systems assigned adders are expected to have high throughput while latency is not so limitation factor, at such systems it may be commodious to follow architectures that are less complicated than carry-skip or carry-look-ahead adders [1], [2].

In this paper, we discuss the various designs of FPGA-based adders.

## 2. FPGA technology

Nowadays Field Programmable Gate Array (FPGA), have produced in varicose processing platforms, with high speed microprocessors, memories and data transfer links. FPGAs have an array of logic modules, programmable routing resources and input/output blocks. FPGAs in compare with CPU have low power dissipation. CPUs run applications as a flow of instructions but FPGA segments application into several optimized and independent logic blocks. A study about CPU and FPGA has been done in [4].

## 3. State of art FPGA-based adders

One of the earliest structural algorithm and design procedure for a 32-bit FPGA-based adder was introduced by Hashemian (1995). This algorithm was based on the carry select technique and for fast response operands was divided into slices. The slice carries by a parallel processing technique were transferred into a multiplexer based structure. By this parallel processing, the final carry terms was produced logarithmically, rather than linearly. This logarithmic approach was for the carry propagation delays decreasing while the data size increase. Base on this algorithm one gate delay is added to the overall time for the addition with each doubling of the operand size [5].

A fault-tolerant adder was offered by Alderighi et al. (2001). On this design fault tolerance at lower design costs and by shifting and rotating the operands given in input to the replicated ALUs, and by adopting a scheme for the full-adder block was accomplishing [6].

Morris et al. (2005) reported two FPGA-based reduction methods for reducing multiple sets of sequentially delivered, floating point values in optimal time without stalling the pipeline. The serial method was a time-division multiplexed implementation of a full binary reduction tree (figure 1). In the parallel method two α-stage adders had used. One of the adders was for reducing all the values for a given set. After arriving the last value in a set, there were multiple partial reductions in the pipeline, if all adders were busy, new values were buffered until an adder became available [7].

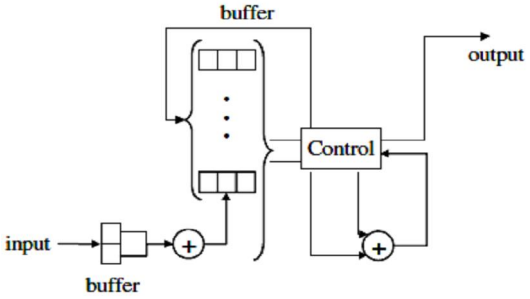


Figure 1: Serial Reduction Architecture

A design methodology for floating-point adder with leading-one predictor (LOP) was presented by Malik & Ko (2005).Figure 2 shows an implementation of this algorithm. LOP was for prediction of the shift amount for post normalization in parallel with the addition. Shifter was for pre-normalization and post-normalization. The LOP was the critical path for the addition operation [8].

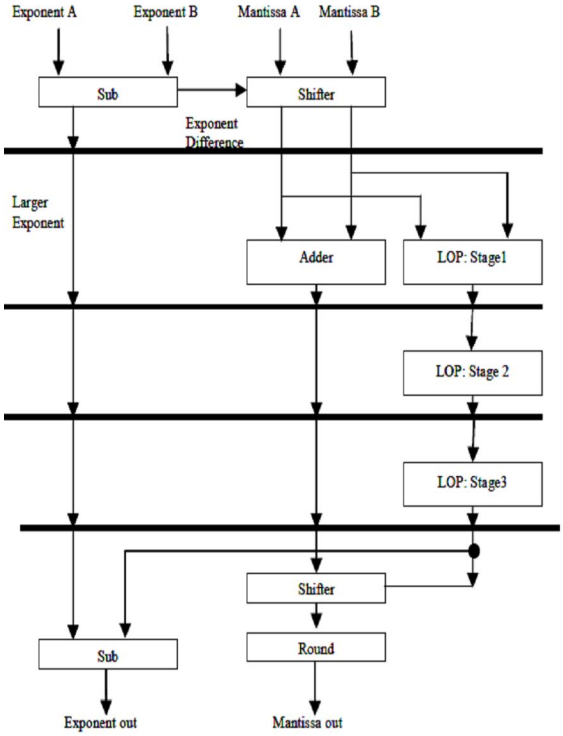


Figure 2: implementation of floating point adder algorithm

Figure 3 depict Maslennikowa et al. (2006) offered structure for the q-operands multi-operand modular adders. These adders were based on a carry-propagate adder tree and had read-only memory (ROM) units for correction of partial results [9].

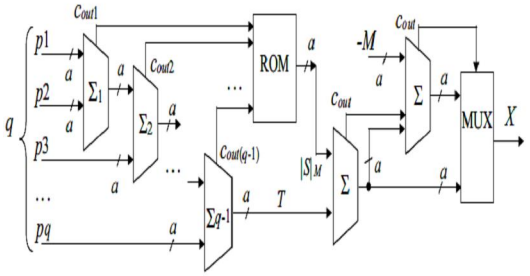


Figure 3: multi-operand modular adders' structure

For designing floating point components in FPGAs, Karlstrom et al. (2006) offered a method. This method was based on using parallel normalization approach to reduce the number of pipeline stages needed to perform the normalization operation. Figure 4 shows the adder architecture. In the first step, the operands are compared and swapped (if require) and the smallest number enters the path with the alignment shifter. Also the implicit one is added at this step, if the input operands are non-zero. In the second step, by the exponent difference, the smallest number is shifted down so that the exponents of both operands match. Add or sub operations are executed in the next step. The final step is the normalization. At this method, due to the earlier comparison and swap step, sub operation never causes a negative result [10].

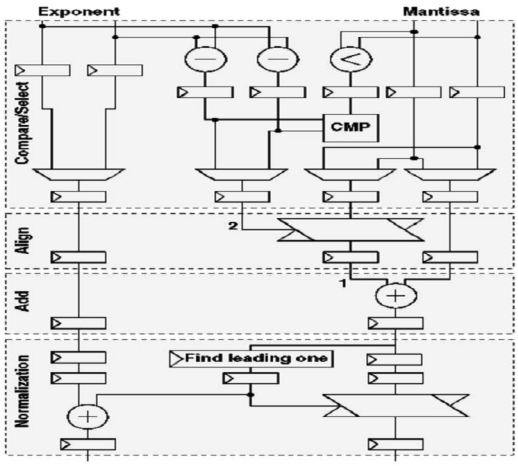


Figure 4: Karlstrom's adder architecture

Kikkeri & Seidel (2007) reported a double precision floating-point adder on the full gate-level verification and FPGA implementation without considering parameterized floating-point adder implementations. For optimization in the design several methods such as nonstandard separation into two paths, unification of rounding cases for addition and subtraction, sign magnitude computation of a difference based on one's complement subtraction, and circuits for approximate counting of leading zeros from borrow-save representation had used [11]. Ng et al. (2008) described an adder for bit-stream signal processing. This circuit was customized for quad-level sigma-delta modulated signals (figure 5). This adder was based on ripple carry adder and one bit of output was fed back to the adder to suppress the truncation error [12].

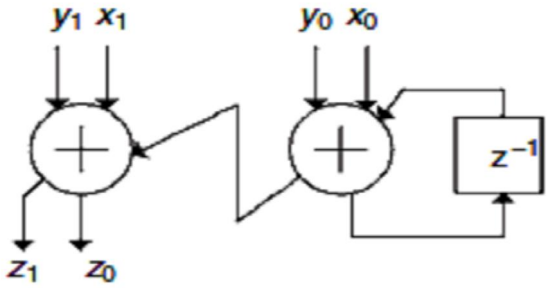


Figure 5: Quad-level bit-stream adder

A 3-input floating-point adder was reported by Guntoro & Glesner (2008). With the purpose of distributing the critical paths and improving the performance the design was based on a 5-level pipeline stage [13]. Malik et al. (2008) studies showed that the standard floating-point adder algorithm is area-efficient, but has more levels of logic and greater overall latency. Leading-one predictor algorithm adds parallelism to the design and thus reduces levels of logic significantly, but because of added hardware and significant routing delays it does not significantly improve overall latency in FPGAs [14].

Yousuf & Najeeb-ud-din (2008) introduced a methodology for carry select adder. In this methodology, sum was calculated for carry-in of '0'and other sum for carry-in of '1'. These sums were calculated by making use of one XOR gate and an inverter. Final sum-out was obtained by making use of multiplexer whose strobe signal was the carry of the previous stage (Cin). Likewise, Carry-out is generated by making use of a multiplexer whose strobe signal is Sum0. Further; optimization of the proposed logic was made by replacing each logic element of the proposed logic with NAND gates. Figure 6 illustrate a carry select adder [15].

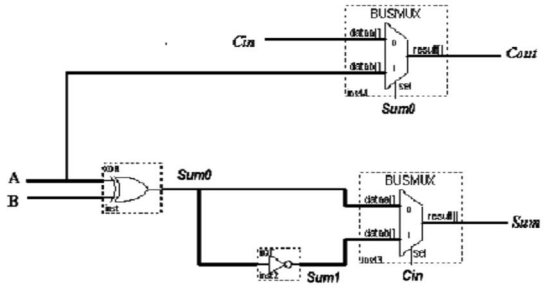


Figure 6: Basic Carry Select Adder Cell

A study on the carry-chain type BCD adders was reported by Biou et al. (2009). Base on this study for big operands the decimal adder works faster than an equivalent binary implementation and furthermore the coding / decoding processes are no more needed. The time delays for BCD adders are slightly better while the hardware requirements, depending on algorithm selection, range from three to four times that of a binary ripple-carry adder. For very great numbers the time saving is more significant [16].

Ortiz et al. (2009) by taking advantage of the specialized carry-logic studied implementations of carry-save adders on FPGA devices. They showed that it is possible to implement redundant adders with a hardware cost close to that of a carry propagate adder. Specifically, for 16 bits and bigger word lengths, redundant adders are clearly faster and have an area requirement similar to carry propagate adders. Among all the redundant adders had been studied, the 4:2 compressor was the fastest one, presented the best exploitation of the logic resources within FPGA slices and the easiest way to adapt classical algorithms to efficiently fit FPGA resources [17]. Liu et al. (2009) different parallel prefix trees used in the design of an end-around carry (EAC) adder targeting FPGA technology [18].

Kamp et al. (2009) introduced adders with a redundancy in representation to eliminate carry propagation, providing near constant addition delay irrespective of the operand width [19]. Rani et al. (2009) introduced a fast adder based on Quaternary Signed Digit (QSD) number system. In QSD, each digit can be represented by a number from -3 to 3 and carry free addition and other operations on a large number of digits such as 64, 128, or more can be implemented with constant delay and less complexity [20].

Bystritskaya et al. (2010) investigated 36-bit ripple-carry, carry-skip, carry-select and carry-look-ahead adders intended for using in field programmable gate arrays. This study showed that

36-bit ripple carry adder has the maximal delay but, in spite of its minimal size, parameter β (β equals to a product of maximal delay and a total number of transistors of adder) is maximal, i.e. the advantage in area does not compensate for the loss in speed. Carry-select adder has maximal parameter β amongst the remaining adders: its size is maximal, but speed is less than that shown by carry-look-ahead adder. The usage of such adder is not profitable in presence of any limiting factors [21].

Bhattacharjee et al. (2011) did a study on low power arithmetic circuits for digital signal processing (DSP) applications in respect to delay, power requirement and implementation costs of the different 8, 16, 32 and 64 bit circuits that can be realized for implementing the basic fixed-point arithmetic units in FPGA [22]. Nguyen et al. (2011) did a study on FPGA-specific arithmetic optimizations for the mapping of carry select and carry-increment adders targeting the hardware carry chains of FPGAs. Different trade-offs between latency and area was explored [23].

Preußer et al. (2011) presented the carry-compact addition scheme. While its central concept was inspired by the carry look-ahead addition, it was distinguished by its internal use of compacted pseudo-addends and its selective formation of compaction groups. A carry-compact addition already outperforms the basic ripple-carry adder for operand widths starting at 50 bits [24].

Martinez et al. (2012) introduced a fault tolerant parallel-prefix adder with capability of both fault detection and correction. This design was using a Sparse Kogge-Stone (SKS) Adder. In figure 7 red highlighted parts shows the error correction and detection logic [25].

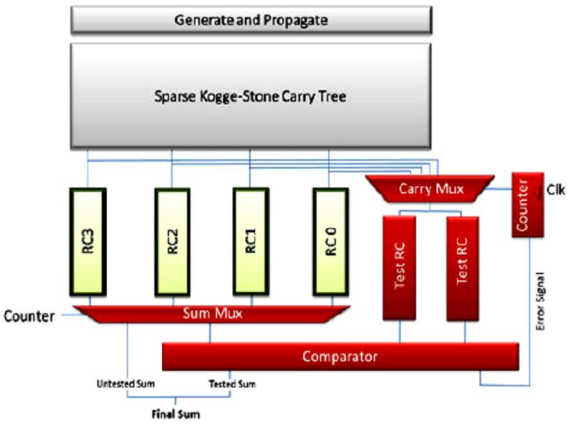


Figure 7: Block diagram of Fault Tolerant Sparse Kogge-Stone Adder

## 4. Conclusion

The recent growth in the number of available resources on FPGAs makes them excellent candidates in many computing applications. FPGA can be used for the applications that require high speed, high precision floating point arithmetic. At this article the state of art techniques in FPGA-Based Adders was discussed.

**Corresponding Author:**
Nasser Lotfivand
Institute of Advanced Technology, University Putra Malaysia, 43400 Serdang, Selangor, Malaysia
E-mail: Ahoora4444@hotmail.com

## References

1. J.M. Muller, Arithmetique des Ordinateurs ( Paris: Masson, 1989).
2. I. Koren, Computer Arithmetic Algorithms (Englewood Cliffs, N.J.: Prentice-Hall, 1993).
3. Hartenstein R., A decade of reconfigurable computing: a visionary retrospective, Proc. Conf. On Design, Automation and Test in Europe, 2001, 642 – 649.
4. Craven S., P. Athanas, Examining the Viability of FPGA Supercomputing, Journal on Embedded Systems, 2007, doi:10.1155/2007/93652
5. Hashemian R., An algorithm and design procedure for high speed carry select adders using FPGA technology, Proc. 37th Midwest Symposium on Circuits and Systems, 1994.
6. Alderighi M., D'Angelo S., Metra C., Sechi G.R., Novel fault-tolerant adder design for FPGA-based systems, Proc. 7th Conf. On-Line Testing Workshop, 2001, 54-58.
7. Morris G.R., Zhuo L., Prasanna V.K., High-performance FPGA-based general reduction methods , Proc. 13th IEEE Conf. on Field-Programmable Custom Computing Machines, 2005, 323 – 324.
8. Malik A., Seok-Bum Ko, Effective implementation of floating-point adder using pipelined LOP in FPGAs, Proc. Conf. on Electrical and Computer Engineering, 2005, 706 – 709.
9. Maslennikowa N., Maslennikow O., Berezowski R., Lienou J.-P., Design Of Fpga-based Multi-operand Modular Adders For Residue Number System Converters, Proc. Conf. on Mixed Design of Integrated Circuits and System, 2006, 264 – 268.
10. Karlstrom P., Ehliar A., Liu D., High Performance, Low Latency FPGA based

Floating Point Adder and Multiplier Units in a Virtex 4, Proc. Conf. on Norchip, 2006, 31 – 34.
11. Kikkeri N., Seidel P.-M., An FPGA Implementation of a Fully Verified Double Precision IEEE Floating-Point Adder, Proc. IEEE Conf. on Application -specific Systems, Architectures and Processors, 2007, 83 – 88.
12. Ng C.W., Wong N., Ng T.S., Quad-level bit-stream adders and multipliers with efficient FPGA implementation, Electronics Letters, 44(12), 2008, 722 – 724.
13. Guntoro A., Glesner M., High-performance fpga-based floating-point adder with three inputs, Proc. Conf. on Field Programmable Logic and Applications, 2008, 627 – 630.
14. Malik A., Dongdong Chen, Younhee Choi, Moon Lee, Seok-Bum Ko, Design tradeoff analysis of floating-point adders in FPGAs, Canadian Journal of Electrical and Computer Engineering, 33(3), 2008, 169 – 175.
15. Yousuf R., Najeeb-ud-din, Synthesis of carry select adder in 65 nm FPGA, Proc. IEEE Conf. on TENCON, 2008, 1 – 6.
16. Bioul G., Vazquez M., Deschamps J.P., Sutter G., Decimal addition in FPGA, Proc. 5th Conf. on Programmable Logic, 2009, 101 – 108
17. Ortiz M., Quiles F., Hormigo J., Jaime F.J., Villalba J., Zapata E.L., Efficient Implementation of Carry-Save Adders in FPGAs, Proc. 20th IEEE Conf. on Application-specific Systems, Architectures and Processors, 2009, 207 – 210.
18. Feng Liu, Forouzandeh, F.F., Mohamed, O.A., Gang Chen, Xiaoyu Song, Qingping Tan, A Comparative Study of Parallel Prefix Adders in FPGA Implementation of EAC, Proc. IEEE Conf. on Digital System Design, Architectures, Methods and Tools, 2009 , 281 – 286.
19. Kamp W., Bainbridge-Smith A., Hayes M., Efficient implementation of fast redundant number adders for long word-lengths in FPGAs, Proc. Int. Conf. on Field-Programmable Technology, 2009, 239 – 246.
20. Rani R., Singh L.K., Sharma N.,FPGA implementation of fast adders using Quaternary Signed Digit number system, Proc. Int. Conf. on Emerging Trends in Electronic and Photonic Devices & Systems, 2009, 132 – 135.
21. Bystritskaya N.A., SmolyannikovI.A., Kurganskii S.I., Investigation of properties of 36-bit adders for creation of DSP blocks on FPGA, Proc. Int. Conf. on Micro/Nanotechnologies and Electron Devices (EDM), 2010, 143 – 146.
22. Bhattacharjee S., Sil S., Basak B., Chakrabarti A., Evaluation of power efficient adder and multiplier circuits for FPGA based DSP applications, Proc. Int. Conf. on Communication and Industrial Application (ICCIA), 2011, 1 – 5.
23. Hong Diep Nguyen, Pasca B., Preusser T.B., FPGA-Specific Arithmetic Optimizations of Short-Latency Adders, Proc. Int. Conf. on Field Programmable Logic and Applications (FPL), 2011, 232 – 237.
24. Preußer T.B., Zabel M., Spallek R.G., Accelerating Computations on FPGA Carry Chains by Operand Compaction Preusser, Proc. 20th IEEE Symposium on Computer Arithmetic (ARITH), 2011, 95 – 102.
25. Martinez Chris D., Bollepalli L. P. Deepthi, Hoe David H. K., A fault tolerant parallel-prefix adder for VLSI and FPGA design, Proc. 44th Southeastern Symposium on System Theory (SSST), 2012, 121 – 125.

1/26/2012