International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.4, No.8, August 2015 DOI:10.15693/ijaist/2015.v4i8.182-185

Carry-Save Arithmetic Using Generalized Parallel Counter

G.Shalini
Assistant Professor
Department of Electronics and communication Engineering
Jayaram college o f engineering and technology

Abstract-Carry-save arithmetic occurs naturally in a variety of DSP application. "field programmable counter array" (FPCA), an accelerator for carry-save arithmetic intended for integration into an FPGA as an alternative to DSP blocks. In addition to multiplication and multiply accumulation, the FPCA can accelerate more general carry-save operations, such as multi-input addition and multipliers that have been fused with other adders. A set of arithmetic-oriented data flow transformations that can be applied to a computation in order to maximize the use of carry-save arithmetic. Each cluster of adders is then replaced with a compressor tree. It is reduces n integers , down to two, sum and carry.A carry-propagate adder (CPA) has two-input adder, then performs the final addition, , to compute the result. FPCA accelerates a wider variety of applications than DSP blocks and improves performance, area utilization, and energy consumption.

***IndexTerms*—Carry-save arithmetic, field-programmable gate array (FPGA), generalized parallel counter (GPC).**

I. INTRODUCTION

FPGA performance is lacking for arithmetic circuits. Generally, arithmetic circuits do not map well onto lookup tables (LUTs), the primary building block for general logic in FPGAs. To address this concern, FPGAs offer two solutions: First, LUTs are now tightly integrated with fast carry chains that perform efficient carry-propagate addition; second, FPGAs contain DSP blocks that perform multiplication and multiply accumulation (MAC). Although an improvement over LUTs alone, these enhancements lack generality; specifically, they cannot effectively accelerate carry-save arithmetic.Carry-save arithmetic is a technique to add sets of numbers that eliminates much of the carry propagation that would otherwise occur. Carry-save arithmetic has been the method of choice for partial-product reduction in parallel multipliers for more than 40 years [1], [6]. More recently, [2] developed a set of arithmetic-oriented data flow transformations that can be applied to a computation in order to maximize the use of carry-save arithmetic.

These transformations systematically reorder the operations in a circuit in order to cluster disparate adders together and to merge adders with the partial-product-reduction trees of parallel multipliers. Each cluster of adders is then replaced with a *compressor tree*, i.e., a circuit that reduces n integers,Ao.A1……An-1 , down to two,S (sum) andC (carry), such that

$$S+C = \sum_{i=0}^{n-1} A_i$$

A carry-propagate adder (CPA), i.e., a two-input adder, then performs the final addition, S+C , to compute the result. Aside from the transformations of [2], compressor trees occur naturally in a variety of applications [3]–[4]. The arithmetic capabilities of FPGAs are not well attuned to the needs of carry-save arithmetic. Programmable LUTs have been augmented with fast carry chains that are good building blocks for CPAs but cannot be used for carry-save arithmetic. The fastest methods to synthesize compressor trees on FPGA general logic [5], [7] do not use the carry chains except for the final CPA.

FPGAs also integrate DSP blocks, which perform integer multiplication and MAC. Although useful, DSP blocks cannot accelerate multi-input addition; likewise, when the transformations merge multipliers with adders, the resulting operation can no longer map onto a DSP block. That being said, certain multiplication operations whose bitwidths do not match up well with the bitwidths of the DSP blocks are faster when performed on the general logic of an FPGA [8].

This paper advocates the use of a field programmable counter array (FPCA) for carry-save arithmetic on FPGAs. The FPCA is a programmable accelerator that can be integrated into an FPGA as an alternative to DSP blocks. An early FPCA, introduced[9] , is a lattice of m:n counters. An m:n counter is a circuit that takes m input bits, counts the number of them that are set to 1, and outputs the result, a value in the range [0,m], as an n-bit unsigned binary number. The number of output bits is

$$n=[\log_2(m+1)$$

The FPCA described in this paper, in contrast, is built using generalized parallel counters (GPCs) , an extended type of counter that can sum bits having different input ranks; GPCs are built using counters as building block

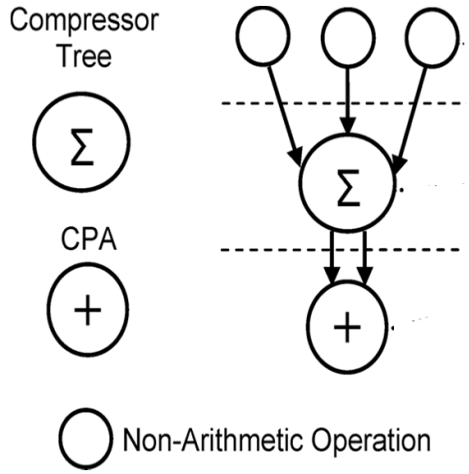


Fig. 1 shows our approach.A circuit transformed as described previously is partitioned into a (set ofcompressor tree(s) with corresponding CPA(s) and a set of nonarithmetic operations. The compressor tree is mapped onto an FPCA, which is embedded
within a larger FPGA.

II.RELATED WORK

A.Commercial FPGA Architectures and Mapping

This section summarizes the arithmetic features in the *Altera Stratix III* and *Xilinx Virtex-5* FPGAs, both of which are high-end FPGAs realized in 65-nm CMOS technology. The logic architectures of both of these FPGAs feature six-input LUTs with carry chains that perform efficient carry-propagate addition without using the routing network. The *Stratix III* carry chain is a ripple-carry adder; the *Virtex-5* carry chain includes an XOR gate and a multiplexor (mux) which enable carry-lookahead addition. *Stratix II* introduced a method to combine the LUTs with the carry chain to perform ternary (three-input) addition, which remained in place for the *Stratix III*; the *Virtex-5* similarly supports ternary addition. Due to the peculiar nature of FPGA architectures, it has long been thought that multi-input addition is best realized using trees of adders rather than compressor trees. The use of ternary adders rather than binary (two-input) adders could reduce the height of the trees, thereby reducing delay and/or pipeline depth.

B. FPGA Enhancements to Improve Arithmetic Performance

Numerous enhancements for FPGAs have been proposed in the past, particularly to improve arithmetic performance. For example, several researchers have proposed hard IP cores: application- specific integrated circuit (ASIC) components that implement common operations that are embedded into the FPGA. The most prevalent of these IP cores include block memories, Although the FPCA is similar in principle to the IP cores described previously, it is not completely hard: It is programmable and has its own routing network.

Although it is intended to implement just one class of circuits—compressor trees, the FPCA is flexible and is not fixed to a specific bitwidth; this distinguishes the FPCA,Recently proposed an alternative FPCA architecture. Theirs is radically different than the one described here; the most important distinguishing feature is that it uses direct programmable connections but does not employ a global routing network; as such, it offers less flexibility than the architecture proposed here, but with the potential of reduced delay, area, and power consumption due to the absence of global routing. Future work will compare and contrast these two architectures to better understand the differences between them.

III GPCs

A *generalized parallel counter (GPC)* is a counter that can sum bits having different ranks; all of the input bits of the same rank are referred to as a *column*. In principle, we could design a GPC that produces more than one output bit of each rank; to simplify our heuristic, however, we only consider GPCs that produce a single output bit of each rank. An *m:n* counter can implement a GPC by connecting all input bits of rank *i* to *2i* inputs of the counter. Of course, it is also possible to construct a GPC from basic gates as well; it turns out that *k*-input GPCs map quite well onto *k*-LUTs.

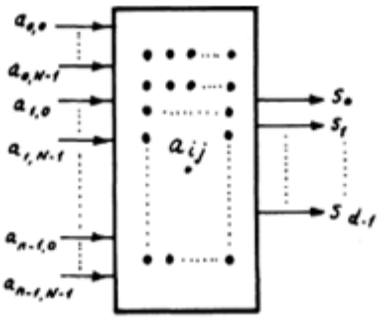


Fig. 2. (n X N; d) counter scheme.

We assume that a GPC has at most *M* input bits and *N* output bits. Formally, a GPC is a tuple: *(KN-2, KN-1, …, K0; S)*, where the output *S* is an *N*-bit number. There are *N-1* columns, from rank *0* to *N-2*, with *Ki > 0* input bits in the *ith* column.

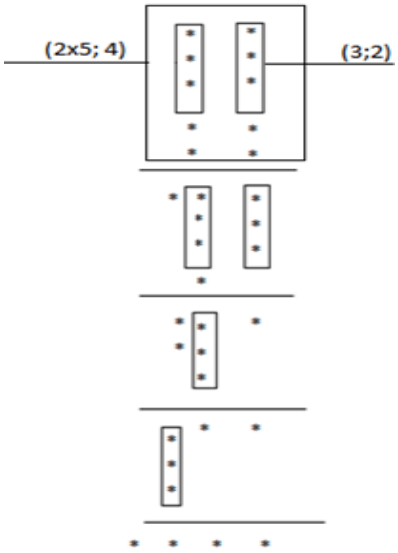


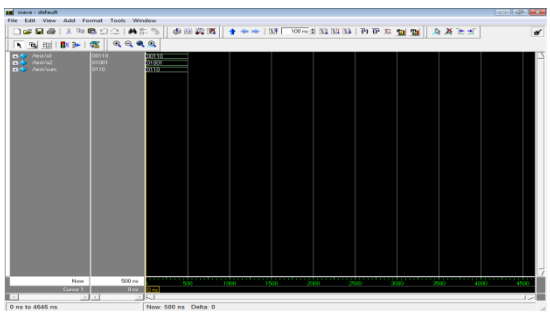
Fig.3. Synthesis of a counter (2x 5; 4)

Counters of this type are denoted as: (n X N; d) and are graphically represented as shown in Fig. 2.Equal columns consume a rectangular matrix segment of n columns by N rows. This type of counter (that they call a maximally efficient equal column counter) is very useful in the synthesis of parallel digital multipliers because in the reduction scheme it produces a uniform output matrix while consuming the largest possible regular portion of the input matrix.

Fig. 3 shows the reduction scheme for the construction of the generalized parallel counter (2 X 5, 4) using primitive networks (3;2) and it also shows the interconnections of these primitive networks.

IV RESULT

A scheme for generating generalized equal column counters from smaller ones has been described. The method proposed gives a specific extension and (2x5;4) counter was obtained using generalized parallel counter



V. CONCLUSION AND FUTURE WORK

The FPCA is a programmable IP core that can accelerate compressor trees on FPGAs. For parallel multiplication, the FPCA retains many of the advantages of the embedded multipliers in the DSP blocks; however, it suffers a disadvantage in terms of area utilization because the partial products must be synthesized on the FPGA general logic. For parallel multiplication, the FPCA retains most of the benefits of the embedded multipliers in the DSP blocks, while providing a variable-bitwidth solution for multiplication operations that do not match the fixed bitwidth of the DSP blocks. Moreover, the FPCA can accelerate multi-input addition operations, while the DSP blocks cannot, particularly when used in conjunction with transformations to expose compressor trees at the application level. Furthermore, the FPCA reduces the critical path delay and energy consumption compared to the best methods to synthesize compressor trees on the FPCA.

The DSP block will generally outperform the FPCA for applications containing many multiplications whose bitwidths match precisely that of the ASIC multipliers in the embedded DSP blocks and for which the transformations are ineffective. For virtually all other applications that contain compressor trees—naturally or via transformation, the FPCA performs
significantly better than current FPGAs.

REFERENCE

[1] L. Dadda, “Some schemes for parallel multipliers,” *Alta Freq.*, vol.
34,pp.349-356 Mar.1965.

[2] A. K. Verma, P. Brisk, and P. Ienne, “Data-flow transformations to maximize the use of carry-save representation in arithmetic circuits,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1761–1774, Oct. 2008.

[3] S. Mirzaei, A. Hosangadi, and R. Kastner, “FPGA implementation ofhigh speed FIR filters using add and shift method,” in *Proc. Int. Conf. Comput. Des.*, San Jose, CA, Oct. 2006, pp. 308–313.

[4] A. Shams, W. Pan, A. Chandanandan, and M. Bayoumi, “A high-performance 1D-DCT architecture,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Geneva, Switzerland, May 2000, vol. 5, pp. 521–524.

[5] H. Parandeh-Afshar, P. Brisk, and P. Ienne, “Efficient synthesis of compressor trees on FPGAs,” in *Proc. Asia-South Pacific Des. Autom. Conf.*, Seoul, Korea, Jan. 2008, pp. 138–143

6] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Improving synthesis of compressor trees on FPGAs via integer linear programming," in *Proc. Int. Conf. Des. Autom. Test Eur.*, Munich, Germany, Mar. 2008, pp. 1256–1261.

[7] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[8] P. Brisk, A. K. Verma, P. Ienne, and H. Parandeh-Afshar, "Enhancing FPGA performance for arithmetic circuits," in *Proc. Des. Autom. Conf.*, San Diego, CA, Jun. 2007, pp. 404–409.

[9] S. Dormido and M. A. Canto, "Synthesis of generalized parallel counters," *IEEE Trans. Comput.*, vol. C-30, no. 9, pp. 699–703, Sep. 1981.

[10] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "A novel FPGA logic block for improved arithmetic performance," in *Proc. Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, Feb. 2008, pp. 171–180.

[11]C.S Wallace"A suggestion for a fastmultiplier".IEEETrans.Electron.comp .no.6.754,Dec 1964.

[12] S. Dormido and M. A. Canto, "An upper bound for the synthesis of generalized parallel counters," *IEEE Trans. Comput.*, vol. C-31, no. 8, pp. 802–805, Aug. 1982.

[13] *"Stratix III Device Handbook, Vol. 1 and 2"* Altera Corporation, San Jose, CA, Feb. 2009.

[14] *"Virtex-5 User Guide"* Xilinx Corporation, San Jose, CA, 2007. [Online]. Available: http://www.xilinx.com/

[15] *"Virtex-5 FPGA Xtreme DSP DesignConsiderations"* Xilinx Corporation, San Jose, CA, Jan. 2009. [Online]. Available: http://www.xilinx. com/

[16] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel, "A hybrid ASIC and FPGA architecture," in *Proc. Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2002, pp. 187–194.

[17] P. Jamieson and J. Rose, "Architecting hard crossbars on FPGAs andincreasing their area-efficiency with shadow clusters," in *Proc. IEEE Int. Conf. Field Programmable Technol.*, Kitakyushu, Japan, Dec. 2007, pp. 57–64.

[18] M. J. Beauchamp, S. Hauck, K. D. Underwood, and K. S. Hemmert, "Architectural modifications to enhance the floating-point performance of FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 2, pp. 177–187, Feb. 2008.

[19] R. Kastner, A. Kaplan, S. O. Memik, and E. Bozorgzadeh, "Instruction generation for hybrid-reconfigurable systems," *ACM Trans. Des.Autom. Electron. Syst.*, vol. 7, no. 4, pp. 602–627, Oct. 2002.

20] K. Leijten-Nowak and J. L. van Meerbergen, "An FPGA architecture with enhanced datapath functionality," in *Proc. Int. Symp. Field Programmable Gate Arrays*, Monterey, CA, Feb. 2003, pp. 195–204.

[21] M. T. Frederick and A. K. Somani, "Multi-bit carry chains for high-performance reconfigurable fabrics," in *Proc. Int. Conf. Field Programmable Logic Appl.*, Madrid, Spain, Aug. 2006, pp. 1–6.

AUTHOR PROFILE

G.SHALINI is a assistant professor in electronics and communication department in Jayaram college of engineering and technology.Het area of interest is Image processing.