Title: Design of Flexible Reconfigurable Architecture for DSP Apllications

Volume 07, Issue 04, Pages: 192 – 199.

Paper Authors

V.Mounika, K.Swapna
Department of ECE, Vaagdevi College of Engineering,Bollikunta Warangal, Telangana.

DESIGN OF FLEXIBLE RECONFIGURABLE ARCHITECTURE FOR DSP APLLICATIONS

V.Mounika[1], K.Swapna[2]

[1]Pg Scholar, Department of ECE, Vaagdevi College of Engineering,Bollikunta Warangal, Telangana.

[2]Associate professor, Department of ECE, Vaagdevi College of Engineering, Bollikunta Warangal,Telangana

ABSTRACT: Hardware acceleration has been proved an extremely promising implementation strategy for the digital signal processing (DSP) domain. Rather than adopting a monolithic application-specific integrated circuit design approach, in this brief, we present a novel accelerator architecture comprising flexible computational units that support the execution of a large set of operation templates found in DSP kernels. We differentiate from previous works on flexible accelerators by enabling computations to be aggressively performed with carry-save (CS) formatted data. Advanced arithmetic design concepts, i.e., recoding techniques, are utilized enabling CS optimizations to be performed in a larger scope than in previous approaches. Extensive experimental evaluations show that the proposed accelerator architecture delivers average gains of up to 61.91% in area-delay product and 54.43% in energy consumption compared with the state-of-art flexible data paths.

INTRODUCTION

Modern embedded systems target high-end application domains requiring efficient implementations of computationally intensive digital signal processing (DSP) functions. The incorporation of heterogeneity through specialized hardware accelerators improves performance and reduces energy consumption [1]. Although application-specific integrated circuits (ASICs) form the ideal acceleration solution in terms of performance and power, their inflexibility leads to increased silicon complexity, as multiple instantiated ASICs are needed to accelerate various kernels. Many researchers have proposed the use of domain-specific coarse-grained reconfigurable accelerators [2]–[9] in order to increase ASICs' flexibility without significantly compromising their performance. High-performance flexible data paths [2], [4], [6], [7], [10] have been proposed to efficiently map primitive or chained operations found in the initial data-flow graph (DFG) of a kernel. The templates of complex chained operations are either extracted directly from the kernel's DFG [10] or specified in a predefined behavioral template library [4], [6], [7]. Design decisions on the accelerator's data path highly impact its efficiency. Existing works on coarse-grained reconfigurable data paths mainly exploit architecture-level optimizations, e.g., increased instruction-level parallelism (ILP) [2]–[5], [7]. The domain-specific architecture generation algorithms of [5] and [9] vary the type and number of computation units achieving a customized design structure. In [2] and [4], flexible architectures were proposed exploiting ILP and operation chaining. Recently, Ansaloni *et al.* [8] adopted aggressive operation chaining to enable the computation of entire sub expressions using multiple ALUs with heterogeneous arithmetic features. Manuscript received July 25, 2014; revised October 16, 2014 and December 12, 2014; accepted January 8, 2015. The authors are with the Department of Electrical and Computer Engineering, Color versions of one or more of the figures in this paper are available The aforementioned reconfigurable architectures exclude arithmetic optimizations during the architectural synthesis and consider them only at the internal circuit structure of primitive components, e.g., adders, during the logic synthesis [11]. However,research activities [12]–[14] have shown that the arithmetic optimizations at higher abstraction levels than the structural circuit one significantly impact on the data path performance. In [12], timing-driven optimizations based on carry-save (CS) arithmetic were performed at the post-Register Transfer Level (RTL) design stage. In [13], common sub expression elimination in CS computations is used to optimize linear DSP circuits. Verma *et al.* [14] developed transformation techniques on the application's DFG to maximize the use of CS arithmetic prior the actual data path synthesis. The aforementioned CS optimization approaches target inflexible datapath, i.e., ASIC, implementations. Recently, Xydis *et al.* [6], [7] proposed a flexible architecture combining the ILP and pipelining techniques with the CS-aware operation chaining. However, all the aforementioned solutions feature an inherent limitation, i.e., CS optimization is bounded to merging only additions/subtractions. A CS to binary conversion is inserted before each operation that differs from addition/subtraction, e.g., multiplication, thus, allocating multiple CS to binary conversions that heavily degrades performance due to time-consuming carry propagations. In this brief, we propose a high-performance architectural scheme for the synthesis of flexible hardware DSP accelerators by combining optimization techniques from both the architecture and arithmetic levels of abstraction. We introduce a flexible datapath architecture that exploits CS optimized templates of chained operations. The proposed architecture comprises flexible computational units (FCUs), which enable the execution of a large set of operation templates found in DSP kernels. The proposed accelerator architecture delivers average gains of up to 61.91% in area-delay product and 54.43% in energy consumption compared to state-of-art flexible datapaths [4], [7], sustaining efficiency toward scaled technologies.

II. CARRY-SAVE ARITHMETIC: MOTIVATIONAL OBSERVATIONS AND LIMITATIONS

CS representation [15] has been widely used to design fast arithmetic circuits due to its inherent advantage of eliminating the large carry-propagation chains. CS arithmetic optimizations [12], [14] rearrange the application's DFG and reveal multiple input additive operations (i.e., chained additions in the initial DFG), which can be mapped onto CS compressors. The goal is to maximize the range that a CS computation is performed within the DFG. However, whenever a multiplication node is interleaved in the DFG, either a CS to binary conversion is invoked [12] or the DFG is transformed using the distributive property [14]. Thus, the aforementioned CS optimization approaches have limited impact on DFGs dominated by multiplications, e.g., filtering DSP applications.

III. PROPOSED FLEXIBLE ACCELERATOR

The proposed flexible accelerator architecture is shown in Fig. 1.

Each FCU operates directly on CS operands and produces data in the same form1 for direct reuse of intermediate results. Each FCU operates on 16-bit operands. Such a bit-length is adequate for the most DSP datapaths [16], but the architectural concept of the FCU can be straightforwardly adapted for smaller or larger bit-lengths.

The number of FCUs is determined at design time based on the ILP and area constraints imposed by the designer. The CStoBin module is a ripple-carry adder and converts the CS form to the two's complement one. The register bank consists of scratch registers and is used for storing intermediate results and sharing operands among the FCUs. Different DSP kernels (i.e., different register allocation and data communication patterns per kernel) can be mapped onto the proposed architecture using post-RTL datapath interconnection sharing techniques [9], [17], [18]. The control unit drives the overall architecture (i.e., communication between the data port and the register bank, configuration words of the FCUs and selection signals for the multiplexers) in each clock cycle.

DFG Mapping Onto the Proposed FCU-Based Architecture:

In order to efficiently map DSP kernels onto the proposed FCU-based accelerator, the semiautomatic synthesis methodology presented in [7] has been adapted. At first, a CS-aware transformation is performed onto the original DFG, merging nodes of multiple chained additions/subtractions to 4:2 compressors. A pattern generation on the transformed DFG clusters the CS nodes with the multiplication operations to form FCU template operations (Fig. 3). The designer selects the FCU operations covering the DFG for minimized latency. Given that the number of FCUs is fixed, a resource-constrained scheduling is considered with the available FCUs and CStoBin modules determining the resource constraint set. The clustered DFG is scheduled, so that each FCU operation is assigned to a specific control step. A list-based scheduler [21] has been adopted considering the mobility2 of FCU operations. The FCU operations are scheduled according to descending mobility. The scheduled FCU operations are bound onto FCU instances and proper configuration bits are generated. After completing register allocation, a FSM is generated in order to implement the control unit of the overall architecture

THEORETICAL ANALYSIS

In this section, we provide a theoretical analysis of the proposed approach based on the unit gate model3 [22]. The critical template of the proposed FCU is the T1 of Fig. 3 and reflects an addition–multiplication–addition operation chaining (AMADFG). Fig. 4(a) shows the AMADFG when all operands are in two's complement form. Fig. 4(b) shows how [12] optimizes the AMADFG. Fig. 4(c) illustrates how [14] distributes the multiplication operation over the CS formatted data. The proposed approach in Fig. 4(d) incorporates the CS-to-MB recoding unit. We assume 16-bit input operands for all the designs and, without loss of generality, we do not consider any truncation concept during the multiplications. Fig. 4(e) shows the area-delay tradeoffs of all the alternative designs. As shown, the proposed design solution is the most effective among all the design alternatives.

CONCLUSION

In this brief, we introduced a flexible accelerator architecture that exploits the incorporation of CS arithmetic optimizations to enable fast chaining of additive and multiplicative operations. The proposed flexible accelerator architecture is able to operate on both conventional two's complement and CS-formatted data operands, thus enabling high degrees of computational density to be achieved. Theoretical and experimental analyses have shown that the proposed solution forms an efficient design tradeoff point delivering optimized latency/area and energy implementations.

REFERENCES

[1] P. Ienne and R. Leupers, *Customizable Embedded Processors: Design Technologies and Applications*. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[2] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems," *J. Supercomput.*, vol. 26, no. 3, pp. 283–308, 2003.

[3] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in *Proc. 13th Int. Conf. Field Program. Logic Appl.*, vol. 2778. 2003, pp. 61–70.

[4] M. D. Galanis, G. Theodoridis, S. Tragoudas, and C. E. Goutis,"A high-performance data path for synthesizing DSP kernels," *IEEE Trans. Comput.-Aided*

Design Integr. Circuits Syst., vol. 25, no. 6, pp. 1154–1162, Jun. 2006.

[5] K. Compton and S. Hauck, "Automatic design of reconfigurable domainspecific flexible cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 5, pp. 493–503, May 2008.

[6] S. Xydis, G. Economakos, and K. Pekmestzi, "Designing coarse-grain reconfigurable architectures by inlining flexibility into custom arithmetic data-paths," *Integr., VLSI J.*, vol. 42, no. 4, pp. 486–503, Sep. 2009.

[7] S. Xydis, G. Economakos, D. Soudris, and K. Pekmestzi, "High performance and area efficient flexible DSP datapath synthesis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 3, pp. 429–442, Mar. 2011.

[8] G. Ansaloni, P. Bonzini, and L. Pozzi, "EGRA: A coarse grained reconfigurable architectural template," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 1062–1074, Jun. 2011.

[9] M. Stojilovic, D. Novo, L. Saranovac, P. Brisk, and P. Ienne, "Selective flexibility: Creating domain-specific reconfigurable arrays," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 5, pp. 681–694, May 2013.

[10] R. Kastner, A. Kaplan, S. O. Memik, and E. Bozorgzadeh, "Instruction generation for hybrid reconfigurable systems," *ACM Trans. Design Autom. Electron. Syst.*, vol. 7, no. 4, pp. 605–627, Oct. 2002.

[11] [Online]. Available: http://www.synopsys.com, accessed 2013.

[12] T. Kim and J. Um, "A practical approach to the synthesis of arithmetic circuits using carry-save-adders," *IEEE Trans. Comput.-0 Aided Design Integr. Circuits Syst.*, vol. 19, no. 5, pp. 615–624, May 2000.

[13] A. Hosangadi, F. Fallah, and R. Kastner, "Optimizing high speed arithmetic circuits using three-term extraction," in *Proc. Design, Autom. Test Eur. (DATE)*, vol. 1. Mar. 2006, pp. 1–6.

[14] 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.

[15] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford, U.K.: Oxford Univ. Press, 2000.

[16] G. Constantinides, P. Y. K. Cheung, and W. Luk, *Synthesis and Optimization of*

DSP Algorithms. Norwell, MA, USA: Kluwer, 2004.

[17] N. Moreano, E. Borin, C. de Souza, and G. Araujo, "Efficient datapath merging for partially reconfigurable architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 7, pp. 969–980, Jul. 2005.