International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

IJIEMR Transactions, online available on 25th Sept2017. Link
:http://www.ijiemr.org/downloads.php?vol=Volume-6&issue=ISSUE-8

Title: **VLSI DESIGN OF AN AREA EFFICIENT ARCHITECTURE OF DSP ACCELERATOR USING DADDA ALGORITHM**

Volume 06, Issue 08, Pages: 327– 333.
Paper Authors

DR.UDARA YEDUKONDALU,SASI PRIYA MUSUNURI ,DR. DOLA SANJAY.S

Ramachandra College of Engineering, Eluru, A.P., India

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

VLSI DESIGN OF AN AREA EFFICIENT ARCHITECTURE OF DSP ACCELERATOR USING DADDA ALGORITHM

[1]DR. UDARA YEDUKONDALU, [2]SASI PRIYA MUSUNURI , [3]DR. DOLA SANJAY.S

[1]HOD & Professor, Dept. of ECE, Ramachandra College of Engineering, Eluru, A.P., India

[2]M.Tech Student, Dept. of ECE, Ramachandra College of Engineering, Eluru, A.P., India

[3]Principal, Ramachandra College of Engineering, Eluru, A.P., India

[1]yedukondalu.udara@gmail.com, [2]musunuri.priya@gmail.com, [3]dicedola@gmail.com

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 gain in area-delay product and also in energy consumption compared with the state-of-art flexible data paths.

Key Words: DSP, CS, Flexible Accelerator

I. 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. 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 in order to increase ASICs' flexibility without significantly compromising their performance.High-performance flexible data paths 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 or specified in a predefined behavioral template library. 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. The domain-specific architecture generation algorithms vary the type and number of computation units achieving a customized design structure. Flexible architectures were proposed exploiting ILP and operation chaining. Recently, Ansaloniet adopted aggressive operation chaining to enable the computation of entire sub expressions using multiple ALUs with heterogeneous arithmetic features.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. The proposed architecture compromising flexible computational units that support the execution of a large set of operation templates found in DSP kernels. The proposed accelerator architecture delivers average gain in area- delay product and also in energy consumption compared with the state-of-art flexible data paths.

II. CARRY SAVE ARITHMETIC

CS representation has been widely used to design fast arithmetic circuits due to its inherent advantage of eliminating the large carry-propagation chains. CS arithmetic optimizations 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 or the DFG is transformed usingthe distributive property. Thus, the CS optimization approaches have limited impact on DFGs dominated by multiplications, e.g., filtering DSP applications.We tackle the limitation by exploiting the CS to modified Booth (MB) recoding each time a multiplication needs to be performed within a CS-optimized data path. Thus, the computations throughout the multiplications are processed using CS arithmetic and the operations in the targeted data path are carried out without using any intermediate carry-propagate adder for CS to binary conversion, thus improving performance.

III. FLEXIBLE ACCELARATOR

The proposed flexible accelerator architecture is shown in Fig. 1. Each FCU operates directly on CS operands and produces data in the same form 1 for direct reuse of intermediate results. Each FCU operates on 16-bit operands. Such a bit-length is adequate for the most DSP data paths, 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 CS to Bin 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

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

data communication patterns per kernel) can be mapped onto the proposed architecture using post-RTL data path interconnection sharing techniques. 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

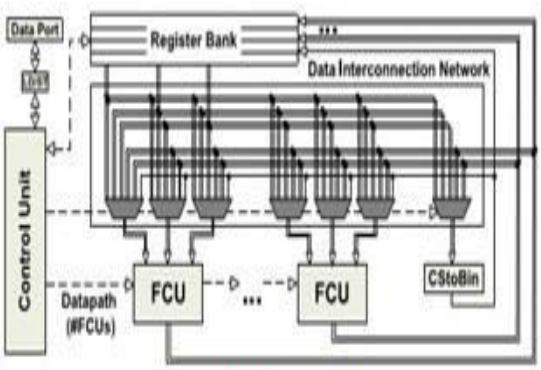


Fig. 1 Flexible data path

A.Structure of the Proposed Flexible Computational Unit

The structure of the FCU (Fig. 2) has been designed to enable high-performance flexible operation chaining based on a library of operation templates. Each FCU can be configured to any of the T1–T5 operation templates shown in Fig. 3. The proposed FCU enables intra template operation chaining by fusing the additions performed before/after the multiplication and performs any partial operation template of the following complex operations

$$W^* = A \times (X^*+Y^*) + K^*$$
$$W^* = A \times K^* + (X^* + Y^*).$$

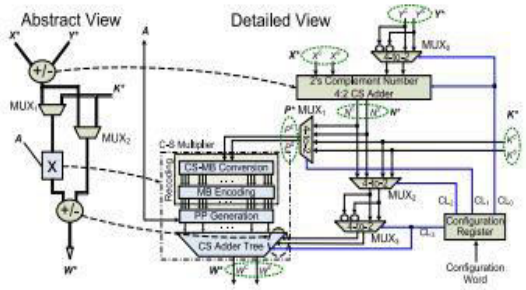


Fig. 2 Flexible Computational Unit

The following relation holds for all CS data X∗={XC,XS}=XC+XS. The operand A is a two's complement number. The alternative execution paths in each FCU are specified after properly setting the control signals of the multiplexers MUX1 and MUX2 (Fig. 2). The multiplexer MUX0 outputs Y∗ when CL0 = 0(i.e., X∗+Y∗ is carried out) or Y∗ when X∗−Y∗ is required and CL0 =1. The two's complement 4 2 CS adder produces the N∗=X∗+Y∗ when the input carry equals 0 or the N∗=X∗−Y∗ when the input carry equals 1. The MUX1 determines if N∗(1) or K∗(2) is multiplied with A.TheMUX2specifies if K∗(1) or N∗(2)is added with the multiplication product. The multiplexer MUX3accepts the output of MUX2 and its 1's complement and outputs the former one when an addition with the multiplication product is required (i.e., CL3=0) or the later one when a subtraction is carried out (i.e., CL3=1). The 1-bit ace for the subtraction is added in the CS adder tree. The multiplier comprises a CS-to-MB module, which adopts a recently proposed technique to recode the 17-bit P∗ in its respective MB digits with minimal carry propagation. The multiplier's product consists of 17 bits. The multiplier includes a compensation method for reducing the error imposed at the product's accuracy by the

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

truncation technique. However, since all the FCU inputs consist of 16 bits and provided that there are no overflows, the 16 most significant bits of the 17-bit W∗ (i.e., the output of the Carry-Save Adder (CSA) tree, and thus, of the FCU) are inserted in the appropriate FCU when requested.

B.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 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 CS to Bin 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 has been adopted considering the mobility 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

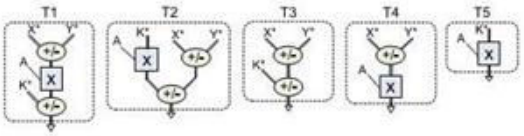


Fig. 3 FCU template library

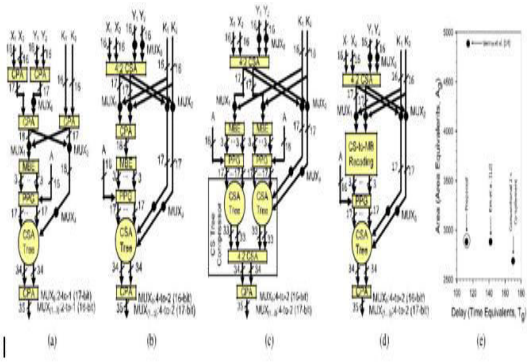
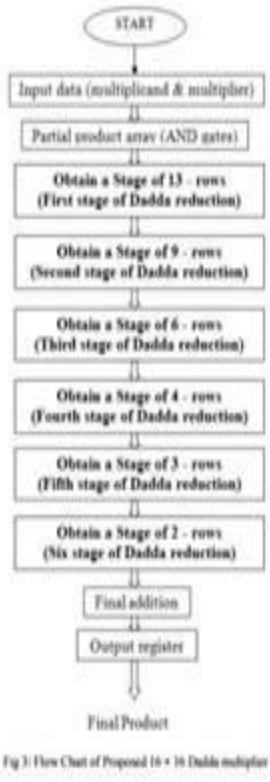


Fig.4. Typical chaining of addition–multiplication–addition operations reflecting T1 template of Fig. 3. Its design is based on (a) two's complement arithmetic,
(b) CS optimizations of [12], (c) CS optimizations with multiplication distribution, and (d) incorporating the CS-to-MB encoding concept. (e) Positioning of the proposed approach with respect to the two's complement one and the CS optimizations

IV. DADDA MULTIPLIER

The Dadda multiplier is a hardware multiplier design, invented by computer scientist Luigi Dadda in 1965. It is slightly faster (for all operand sizes) and requires fewer gates (for all but the smallest operand sizes) than array multiplier. Dadda multipliers have the same 3 steps 1. Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding N2 results. Depending on position of the multiplied bits, the wires carry different weights, for example wire of bit carrying result of a2b3 is 32. 2. Reduce the number of partial products to two layers of full and half adders. 3. Group the wires in two numbers, and add them with a conventional adder.The proposed multiplier 16x16 Dadda multiplier requires six reduction stages with intermediate matrix heights of 13, 9,6,4,3 and finally 2.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org



The Dadda scheme essentially minimizes the number of adder stages required to perform the summation of the partial products. This is achieved by using full and half adders to reduce the number of rows in the matrix of bits at each summation stage by a factor of 3/2.The results in a final matrix consisting of two rows of bits which must be summed using a multiple-bit adder (e.g. a ripple-carry or carry look-ahead adder). The corresponding circuit for a multiplier using this scheme shown in Fig 2.

V. SIMULATION

This simulation is run by Xilinx ISE Design Suite. Fig. 5 shows the simulation result for Flexible control unit and Fig. 6 shows the Area results

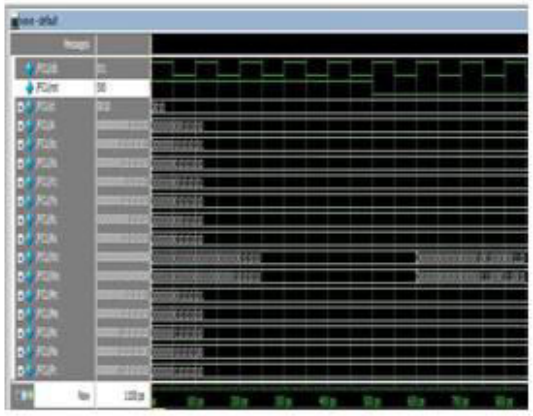


Fig. 5 Flexible Control Unit

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	824	4656	17%
Number of Slice Flip Flops	128	9312	1%
Number of 4 input LUTs	1472	9312	15%
Number of bonded IOBs	185	232	79%
Number of GCLKs	1	24	4%

Fig. 6 Area Result

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

[11] 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.

B. Parhami, *Computer Arithmetic: Algorithms and Hardware*

Designs. Oxford, U.K.: Oxford Univ. Press, 2000.

[12] Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, 1965.

[13] An efficient floating-point multiplier design using combined booth and dada algorithms. 30th June 2014. Vol. 64 No.3

[14] Design and Analysis of CMOS Based DADDA Multiplier. IJCEM, VOL .16 ISSUE 6, NOV 2013

Authors profile:

Yedukondalu Udara

Received the B.Tech in Electronics & Communication Engineering from VRSEC, Vijayawada in 2001 and M.Tech in I & CS from JNTU, KANINADA in 2007. Ph.D. He obtained doctorate from ANU College of Engineering and Technology. His area of interest comprises is Low Power VLSI Design. Presently, he is working as Head of the department and professor, Department of Electronics and Communication Engineering, Ramachandra College of Engineering.

Sasi Priya Musunuri

Received the B.Tech in Electronics & Communication Engineering from Rama Chandra college of Engineering affiliated to JNTU, KAKINADA in 2014 and pursing M.Tech in VLSI from Ramachandra College of Engineering, Andhra Pradesh. Her research interests include VHDL, Verilog modeling of digital circuits and testing, Schematic Diagram and Simulation, Digital circuits using Mentor Graphics, Pyxis and Calibre.

Dola Sanjay. S

Received the B.Tech in Electronics & Communication Engineering from UBDTCE, (Govt College) Davanagere, Karnataka in 1999, M.Tech in Applied Electronics from Dr.MGR University, Chennai in 2007 and Ph.d from JNTUA, Anantapur, AP in 2007. He has Over 14+ years experience in Teaching with reputed Engineering colleges. Insightful experience as principal in-charge principal, Vice principal, All India Council for Technical Education(AICTE) , Local Inspection Committee (LIC), Facts Finding Committee (FFC), National Board of Accreditation (NBA). Presently he is working as Principal in Ramachandra College of Engineering.