16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

A FLEXIBLE ARCHITECTURE FOR DSP APPLICATIONS COMBINING HIGH PERFORMANCE ARITHMETIC WITH SMALL SCALE CONFIGURABILITY

Sotiris Xydis, Isidoros Sideris, George Economakos and Kiamal Pekmestzi

School of Electrical and Computer Engineering
National Technical University of Athens
Athens 15773, Greece
Email: {sxydis, isidoros, geconom, pekmes}@microlab.ntua.gr

ABSTRACT

This paper presents the architecture of a flexible and high performance unit for DSP applications. The proposed architecture operates based on fast Carry-Save (CS) arithmetic. A mapping methodology, for datapaths composed with the proposed flexible units, is also presented. It exploits the incorporated features of the proposed units and enables fast computations, high operation densities and advanced data reusability. Experimental results shown that several DSP algorithms can be mapped onto the proposed architecture with high efficiency delivering in average, latency gains of 36.56% and 45.76% compared to the MAC and the primitive resources based datapaths, respectively.

1. INTRODUCTION

As the VLSI fabrication technology continues to scale down the design of complex configurable hardware systems has became a reality. In contrast with the classical FPGA-like reconfigurable architectures [1], coarse-grained architectures have been proposed in the bibliography [2], which preserve flexibility and deliver high speed implementations along with low power consumption.

Recently, the notion of domain-specific reconfigurable logic has been introduced [3]. It includes the idea of providing only a certain amount of reconfiguration capability to the developing ASIC, according to the desired applications' specific needs. The Digital Signal Processing (DSP) domain seems to be an extremely promising area for the integration of domain-specific reconfigurable methods and architectures, since it is dominated by data-path intensive applications with lightweight control logic.

In order to improve DSP algorithms' implementations, Multiply-Accumulator (MAC) units are often allocated. Many approaches have been presented for the design of fixed MACs [4], [5]. They focus on performance and power optimizations, but they do not take into account any reconfiguration/flexibility properties.

Recently, Tatas et al [6] have proposed a reconfigurable MAC architecture which is based on a 16×16 multiplier's decomposition and it can be reconfigured to adapt its structure to the application needs, trading bitwidth for array size. In [7] a reconfigurable MAC unit is introduced. It performs one 32-bit multiplication or two 16-bit multiplications by splitting the multiplier in two pieces. Both approaches propose MAC architectures, configuring only the bitwidth of the operators.

Considering datapaths with only MAC units allocated, DSP applications such as DCT, FFT, symmetrical FIR filters etc cannot be efficiently mapped. Several application specific circuits have been proposed in [8], [9], [10] for optimized implementation of these applications. Bruguera [9] proposed the usage of a carry-save (CS) to signed-digit (SD) recoding scheme in order to fuse in an efficient way addition operations prior to multiplication ones. Definitions of carry-save and signed-digit arithmetic representations can be found in [11]. In [10] an area efficient CS to SD recoding scheme has been presented which recodes the initial CS data to digits proper for Booth encoding. Nevertheless, no notion of flexibility has been reported for any of these type of contributions.

In this paper, motivated by the above observations, we propose a flexible and high performance architecture able to execute efficiently a large set DSP operation templates. High performance is achieved by exploiting the CS arithmetic format which eliminate the time consuming carry-propagation. Flexibility is enabled based on small scale configurability, by inserting a limited number of multiplexors. The proposed architecture actually merges the MAC and the fused addition-multiplication datapath in one unit. It combines pre- and post-multiplication addition, and it extends the number of input data operands by allowing computations on CS data formats. A mapping flow based on well known High Level Synthesis (HLS) heuristics [12] is also proposed, which exploits the dense and flexible structure of the proposed architecture. It have to be mentioned that, the proposed architecture targets the ASIC domain space rather the FPGA one [1], since it is composed by components of coarser granularity than the basic elements of classical FPGA devices.

The remainder of the paper is organized as follows. In Section 2 the proposed FAMA architecture is presented in detail. The mapping of DSP application kernels to FAMA based datapaths is presented in Section 3. Section 4 refers to experimental results. Finally, Section 5 concludes the paper.

2. THE FLEXIBLE FAMA ARCHITECTURE

2.1 FAMA Architecture Description

The proposed architecture is based on a carry-save Fused Add-Multiply-Add (FAMA) unit. FAMA's general functionality can be described by the following equations:

$$Z^* = N^* \times A + K^* \tag{1a}$$

$$N^* = X^* + Y^* \tag{1b}$$

$$X^* = \{X^C, X^S\} = X^C + X^S \tag{1c}$$

The superscripted star, *, denotes a redundant representation composed of two numbers both in 2s complement form (defined as 2's complement CS form/representation). Each of

the Z^*, N^*, Y^*, K^* is formed like X^*. The quantities Z^C, Z^S, N^C, N^S, X^C, X^S, Y^C, Y^S, K^C, K^S and A are all 2's complement conventional binary numbers.

The primary inputs of the unit, X^*, Y^*, K^*, can be either in 2's complement conventional binary or in 2's complement CS form. The latter enables the direct reusability of the unit's output, eliminating the need for the latency intensive conversion from the CS representation to the conventional binary form.

The overall architecture of the flexible FAMA unit is depicted in Fig. 1. It operates on data of 16-bit wordlength, since 16-bits enable sufficient accuracy for the majority of modern DSP applications [13]. It consists of 1) a 2's complement numbers 4:2 CS adder (CSA), for the addition of the input data (X^*, Y^*), 2) the recoding scheme, 3) a tree based adder for the addition of the partial products, 4) the final CS accumulation unit implemented by a 4:2 CSA and 5) a configuration register.

The upper 4:2 CSA (see Fig. 3) computes the $N^* = X^* + Y^*$ which can be used either in the pre- or the post-multiplication addition. Input A has to be in 2's complement binary numeric representation.

At next, the 2's complement CS formed data, N^* or K^*, are driven to the recoding unit. The recoding unit performs the conversion from CS format to SD format and subsequently the conversion of each signed digit to the Sign-Magnitude (SM) digit representation. The recoding unit is presented in more detail in Section 2.2. For now, we support signed digits, D_j, ranging between [-1, 1], but this scheme can easily be extended (i.e in a Modified's Booth) for even more efficient implementations of the multiplier.

The Sign-Magnitude (SM) [11] digits $(sgn(D_j), |D_j|)$ are passed through the partial product generation component together with the input A. The Signed Partial Products (SPP_js) are generated based on the following logic function.

$$SPP_j = (A \oplus sgn(D_j)) \cdot |D_j| \cdot 2^j \tag{2}$$

If the magnitude of the signed digit D_j is zero then the partial product is also zero. If $|D_j| = 1$ the input A is

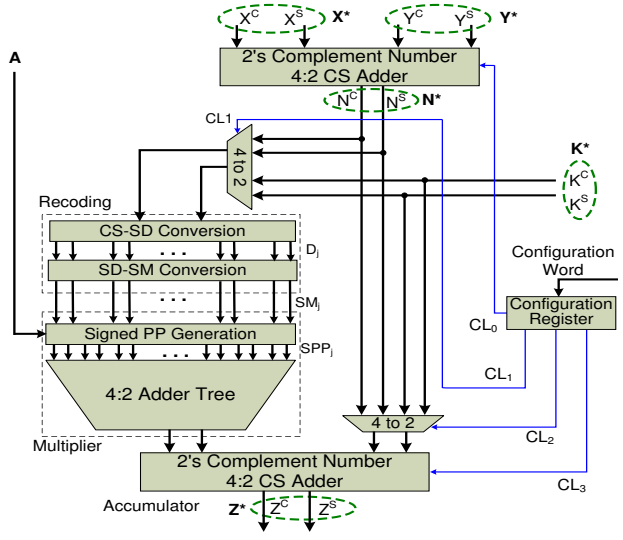

Figure 1: The configurable FAMA architecture.

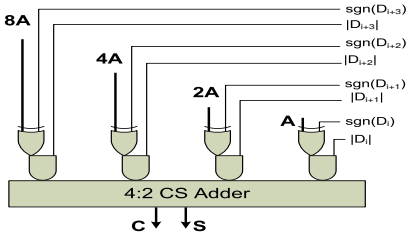

Figure 2: The Partial Product Generation and the 4:2 CSA Component of the Tree based Adder.

"*XORed*" with the $sgn(D_j)$. That is, if the D_j has a negative value, $sgn(D_j) = 1$, the input X is complemented. When $sgn(D_j) = 0$, the D_j has a positive value and remains the same (Fig. 2). In case of negative D_j, the "*XORed*" A has to be incremented by 1, to form the correct 2's complement inverted number. This increment is performed inside the tree CS adder, taking into account the weights of the SPP_j's bits.

We decide to use a tree based multiplier trading with a more canonical multiplication scheme, because tree multipliers deliver high performance gains. For 16-bit data, we need 3 levels of 4:2 CS adders. In case of Modified Booth encoding 2 levels of 4:2 adders are sufficient. The main principles can be adapted in other multiplication schemes, i.e array multipliers, in a straightforward manner. Because the paper focuses on the overall architecture and the mapping opportunities it reveals, we neglect further details about the core multiplier's circuit.

The final CS accumulation unit is a 4:2 CSA with two fixed inputs (the multipliers output bits) and two configurable inputs. All the inputs of the accumulation unit are in 2's complement CS format. The configured inputs are either the N^* derived from the initial 4:2 CSA or an independent CS input number (K^*).

The configuration register controls the operation mode of the unit and the signs of the input numbers, by driving with proper control logic bits (CL_i) the multiplexors and the sign selection inside each 4:2 CSA module. It is loaded in a cycle by cycle basis.

2.2 The Recoding Module

The recoding module enables the addition prior to multiplication to be performed without introducing any propagation delay. The detailed structure of the recoding module, for $n = 4$-bit data, is illustrated in Fig. 3. The module can be partitioned in three stages.

The first stage implements the addition of four 2's complement binary number resulting at a CS format. It consists of two lines of full adder (FA) cells. Given that the most significant bit in the 2's complement form has a negative value, the two left-most cells are slightly modified. They are composed by FA cells with inverters at the negative values' input and output ports [8]. Actually, this stage is implemented by the 4:2 2's complement CS adder (Fig. 3). The CS representation of $X^* + Y^*$ is generated according to the next relation:

$$FA_{0,j}(X_j^S, X_j^C, Y_j^C) = \{S_{0,j}, C_{0,j+1}\} \tag{3a}$$

$$FA_{1,j}(S_j, C_j, Y_j^S) = \{S_j, C_{j+1}\} \tag{3b}$$

where the $S_{0,j}$ and the $C_{0,j+1}$ bits are the sum and carry bits of the full adder at the first row. $FA_{0,j}$ and the S_j, C_{j+1} bits

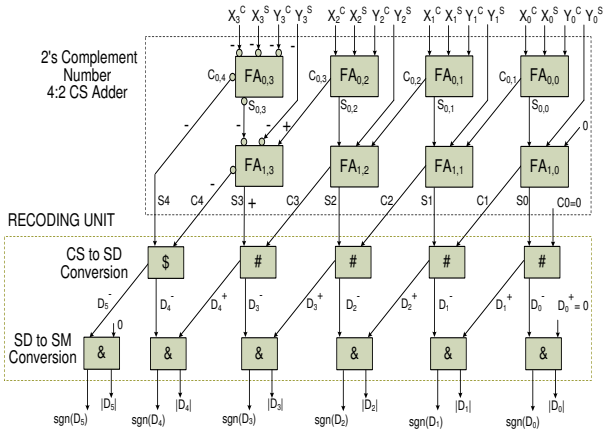

Figure 3: A 4-bit 4:2 CS Adder and Recoding Unit.

are the bits which form the 2's complement CS form, respectively. The result with $n+1$ digits is also in 2's complement CS form.

At next, the CS form has to be recoded to the SD format. This recoding operation is performed by the second stage, CS to SD conversion. This module comprises of $n+1$ cells which operate on each CS input digit. The first n cells are identical (the # modules in Fig. 3). Their truth table is depicted in Fig. 4 (Table A). The SD representation of the input data is generated according to the next relations:

$$D_j^- = S_j \oplus C_j \tag{4a}$$

$$D_{j+1}^+ = S_j + C_j \tag{4b}$$

where D_j^- and D_{j+1}^+ have negative and positive value, respectively. Each D_j digit consists of the (D_j^+, D_j^-) bits and ranges between [-1,1].

Taking into account that the most significant CS digit $(C_{0,n}, C_{1,n})$ is formed by 2 bits with negative values, the logic function of the left-most cell ($) is different and its logic function is described by Table B in Fig. 4.

The last recoding stage (SD to SM) encodes the SD digits to their SM format. For $n+1$ CS digits the CS-SD module generates $n+2$ output digits and the last stage SD-SM requires $n+2$ cells of type &. At each SM digit $sgn(D_j)$ represents the D_j's sign and the $|D_j|$ represents the D_j's magnitude. The truth table of the cell of type & is given in Table C and its logic function is:

$$sgn(D_j) = D_j^- \tag{5a}$$

$$|D_j| = D_j^- \oplus D_j^+ \tag{5b}$$

The most significant SM digit is generated from the negative valued SD_n^- bit and a 0 positive valued bit. So the consistency

Table A: Truth Table of #

Si	Ci	Vi	D_i^-	D_{i+1}^+
0	0	0	0	0
0	1	1	1	1
1	0	1	1	1
1	1	2	0	1

Table B: Truth Table of $

Si	Ci	Vi	D_i^-	D_{i+1}^-
0	0	0	0	0
0	1	-1	1	0
1	0	-1	1	0
1	1	-2	0	1

Table C: Truth Table of &

D_i^-	D_i^+	Vi	sgn	$\|D_i\|$
0	0	0	0	0
0	1	1	0	1
1	0	-1	1	1
1	1	0	1	0

Figure 4: The Recoding Truth Tables.

of the SM representation is maintained. The output digits of the SD to SM recoding unit are forwarded, together with the input A, to the signed partial product generation unit (Fig. 2), described in Section 2.1 to initialize the multiplication process.

It is worth to be noticed that the 4:2 CSA, the CS to SD and the SD to SM conversion modules are carry-propagation-less circuits, contributing to the critical path by the delay of 3 full adders, approximately.

3. APPLICATION MAPPING ONTO FAMA DATAPATHS

The proposed FAMA architecture enables the execution of complex and computationally expensive operation templates. Recent research activities [14], [15] from the HLS field have shown that DSP applications can benefit in terms of performance and power consumption by using resources that implement efficiently computational templates of chained operations. In [15] a large number of different template-based hardware resources is generated augmenting the system's heterogeneity. In [14] the authors proposed an generalized coarse-grained reconfigurable template, named FCC, to tackle the heterogeneity problem of such synthesized datapaths. However, each FCC reconfigurable cell comprises a large number of computational elements (4 ALU units and 4 multipliers) which in each control step are not fully utilized.

The FAMA architecture is an intermediate solution enabling lower heterogeneity of the synthesized datapaths comparing with the [15] approach, without spanning so many hardware resources as in [14]. Each FAMA resource supports a large library of operation templates. In Fig. 5 the FAMA's template library is illustrated, considering primary inputs in the 2's complement CS form. FAMA delivers high speed computations exploiting the Carry-Save arithmetic operation without surcharging the datapath's controller due to its small configuration word.

The mapping procedure onto datapaths composed by FAMA units consists of 3 major steps. At first, the Control-Data-Flow-Graph (CDFG) of the application's initial specification is extracted. The CDFG captures the control flow of the application between the Basic Blocks (BBs) and the data flow in each BB. The BBs are executed sequentially. When the execution proceeds between the BBs proper control signals are generated to drive the control flow of the computation. The operation level parallelism is exploited inside each BB by the constructing the BB's Data-Flow-Graph (DFG).

After the CDFG's extraction each BB's DFG is parsed and a clustering procedure takes place in order to transform the initial DFG into a FAMA based DFG (Fig. 6). The inputs and the output of each DFG node are assumed to be in 2's complement CS format in order to be consistent with the

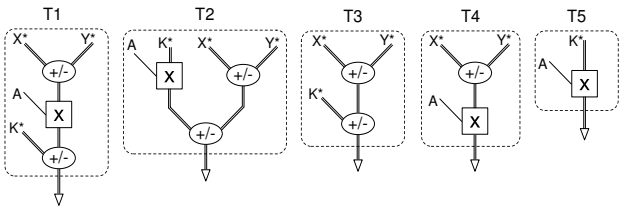

Figure 5: FAMA's Template Library.

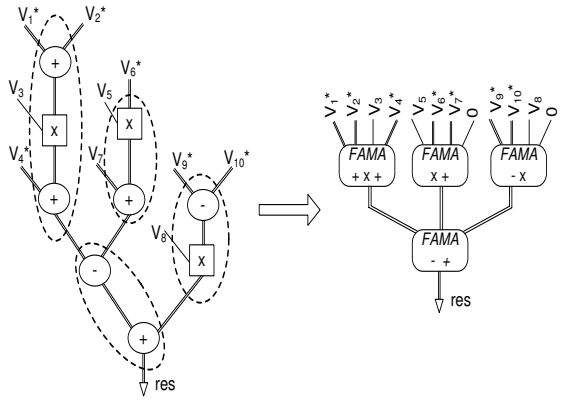

Figure 6: Clustering Procedure based on DFG Covering.

general execution case. Conventional binary inputs can be modeled as a partial case. The clustering is based on the coverage of the DFG in respect to the template library of Fig. 5. The FAMA based DFG (F-DFG) expresses the initial specification in terms of the new resource space. In case that there is a conflict during the clustering process (i.e in which FAMA cluster has to belong a primitive operation node, when two clusters overlap) a refereed criterion is evaluated. In our case, this refereed criterion drives the conflict nodes to the cluster that encapsulates the larger number of DFG nodes belonging in the initial DFG's critical path. Due to the facts that 1) the FAMA's output is in 2's complement CS data format and 2) that the FAMA's multiplier can hold only one CS formatted operand, when an edge between two operations clustered in different sets is identified and if the sink DFG node is a multiplication operation, this edge is marked to indicate to the scheduler that a conversion (=simple addition with carry-propagation) is required from the CS format to the conventional binary form.

The new data dependencies of the FAMA based DFG are calculated and the graph is fed to a modified resource constrained List-Based Scheduler [12], which considers resources of the FAMA type. The modified scheduler distributes the nodes of the F-DFG among the control-steps and the available hardware resources based on the mobility (=ALAP-ASAP value) of each node. The nodes with smaller mobility are executed first. The modified scheduler also respects the marked edges of the F-DFG interleaving a arithmetic conversion step in the proper cases and takes into account the CS formatted intermediate results. The scheduler has been designed for mixed resource allocation scenarios composing of FAMA units and conventional binary primitive resource circuits. In case that all the resources are based on CS arithmetic the scheduler is simplified, given that it has not to consider the CS formatted intermediate results.

4. EXPERIMENTAL RESULTS

This section demonstrates the advanced mapping capabilities offered by the proposed architecture. Our benchmark suite includes five DSP computational intensive kernels shown at the first column of Table 1. The DSP kernels are represented by their innermost Data-Flow Graphs (DFGs).

We experiment with three different resource allocation scenarios. An upper-bound constraint of 3 ALU units has been imposed in each scenario. The first scenario considers the allocation of 2 FAMA units, the second scenario the allocation of 2 MAC units, while the third scenario considers the resource constraint of 2 Wallace multipliers. Under the above resource constraints, the benchmark DFGs were scheduled with a mobility based list-scheduler [12].

The time-frame for one control step was set separately for each allocation scenario based on the component with the longest critical delay found in each scenario. Accuracy of 16-bit has been considered. So, for the FAMA-based datapath the clock-cycle was set to 5.2 ns. For the MAC based datapath the cycle delay was set to 5.4 ns, taking into account the carry-propagation final adder delay overhead, while for the multiplier-based datapath the clock period was set to 4 ns. The aforementioned clock cycle periods are based on [16] for the TSMC 0.13 standard cell library [17], and they were coarsely estimated considering the delays of the individual components which form each datapath.

Table 1 shows the number of DFG nodes for each benchmark and the utilization ratio of the two FAMA instances $(FAMA_A, FAMA_B)$. $FAMA_A$ has high utilization ratio of 100%. $FAMA_B$ unit has a significantly lower utilization of 61.76% in average. The main reason that this happens is the mapping strategy we apply considering $FAMA_A$ as the primary FAMA unit. In case of a 7 taps symmetrical FIR (Finite Impulse Response) filter (Sym FIR7), where the utilization ratio of $FAMA_B$ is zero, only one FAMA unit is allocated. Maximum utilization of both FAMA resources is observed in the 4-point FFT (Fast-Fourier Transform) benchmark, because it incorporates many alu-mul-alu operations along with high parallelism features.

Table 2 records the number of allocated resources in datapaths composed by 1) FAMA units, 2) MAC and ALU units, 3) Wallace multipliers (MULs) and ALU units. It can be seen that FAMA based datapaths, unlike the MAC and MUL based datapaths, are able to execute the overall DFG computation without any extra allocated hardware units. Thus, using FAMA units high performance datapaths (Fig. 7) with low heterogeneity and high hardware utilization can be implemented. In case of the final results have to be in conventional binary format, a carry-propagate adder can be allocated for the final CS to binary conversion. There is no need to use a faster type of adder, considered that the critical path of a FAMA unit is much longer than the critical path of a carry-propagate adder.

Table 3 illustrates the comparative results for the three resource assignment scenarios. The execution cycles and the cycle gain, using FAMA units, are reported for each scenario case. In average the FAMA based datapath delivers cycle gains of 28.54% compared with the MAC based datapaths and of 82.82% compared with the datapaths composed by primitive resources. FAMA based datapaths of Elliptic filter and FFT delivers the highest cycle gains. For the 8-point Dis-

Table 1: DFG's Node Count and Usage in the FAMA based Datapath.

Benchmark	DFG nodes	$FAMA_A$ Utilization	$FAMA_B$ Utilization
Fir11	21	100%	40%
Sym FIR7	20	100%	0%
FFT	18	100%	100%
DCT	24	100%	88.8%
Elliptic	39	100%	80%

Table 2: Allocated Resources using FAMAs, MACs and MULs.

Benchmark	No. of **FAMA**	No. of ALUs	No. of **MAC**	No. of ALU	No. of **MUL**	No. of ALU
Fir11	2	0	2	0	2	2
Sym FIR7	1	0	1	1	2	2
FFT	2	0	2	3	2	3
DCT	2	0	2	3	2	3
Elliptic	2	0	2	3	2	3

Table 3: Comparative Performance Results (No. of Cycles).

Benchmark	FAMA Based	MAC Based	Cycle Gain (%)	MUL Based	Cycle Gain (%)
Fir11	9	9	0	12	33.3
Sym FIR7	7	8	14.2	8	14.2
FFT	7	9	28.5	14	100
DCT	9	9	0	15	66.6
Elliptic	5	10	100	15	200
Average	-	-	**28.54%**	-	**82.82%**

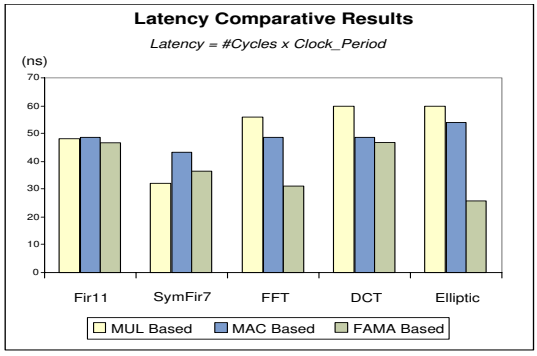

Figure 7: Comparative Latency Results.

crete Cosine Transform (DCT) benchmark, the FAMA based datapath is scheduled with the same cycle count as the MAC based. This is happened due the fact that the initial additions are performed by FAMA units configured to operate the T3 operation template (Fig. 5). However, it has to be mentioned that only two FAMA units have been allocated, while in case of the DCT MAC-based datapath three extra ALU resources has been considered (Table 2).

Comparative latency results are depicted in Fig. 7. An average latency gain of 36.56 % and 45.76 % is achieved by FAMA based datapaths in comparison with the other two allocation scenarios. Elliptic filter presents the largest latency reduction with two FAMA units required, while the MAC based datapath needs 2 MACs and 3 extra ALU units. At FIR11 and Symmetrical FIR7 filters, the FAMA's datapath delivers latency reduction at significantly smaller ranges because MAC based datapath are highly tuned for this type of applications.

5. CONCLUSION

In this paper, we presented a flexible hardware architecture which enables the execution of complex DSP operation templates. A mapping procedure onto the the proposed flexible architecture was also introduced.

Future work will focus onto the low level implementation and the extensive experimentation of the proposed architecture to evaluate more robustly its hardware characteristics.

REFERENCES

[1] Xilinx Spartan Series, www.xilinx.com.

[2] R. Hartenstein, "A decade of reconfigurable computing: A visionary retrospective," in *Proc. of ACM/IEEE Design Automation and Test in Europe Conference*, 2001, pp. 642–649.

[3] T. S. Baloch, I. Ahmed, "Domain-specific reconfigurable array targeting discrete wavelet transform for system-on-chip applications," in *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 3*, 2005, p. 161.

[4] P. Stelling, V. Oklobdzija, "Implementing Multiply-Accumulate Operation in Multiplication Time," in *proccedings of 13th IEEE Symposium on Computer Arithmetic*, 1997, p. 99.

[5] Yuyun Liao, D.B. Roberts, "A High-Performance and Low-Power 32-bit Multiply-Accumulate Unit with Single-Instruction-Multiple-Data (SIMD) Feature," *Solid-State Circuits IEEE Journal*, vol. 37, no. 7, pp. 926–931, July 2002.

[6] K. Tatas, G. Koutroumpezis, D. Soudris, A. Thanailakis, "Architecture Design of a Coarse-Grain Reconfigurable Multiply-Accumulate Unit for Data-Intensive Applications," *Integration the VLSI Journal*, vol. 40, no. 2, pp. 74–93, Feb 2007.

[7] Y. Li and J. Chen, "Reconfigurable architecture of a high performance 32-bit MAC Unit for embedded DSP," in *Proc. of ASICON 2003*, 2003, pp. 1285–1288.

[8] K. Pekmestzi, "Complex Number Multipliers," *IEE Proceedings*, vol. 136, pp. 70–75, Jan. 1989.

[9] J. Bruguera and T. Lang, "Implementation of the FFT Butterfly with Redundant Arithmetic," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 717–723, Oct. 1996.

[10] W. Yeh, C. Jen, "A High Performance Carry-Save to Signed-Digit Recoder for Fused Addition-Multiplication," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing - ICASSP*, vol. 6, 2000, pp. 3259–3262.

[11] K. Hwang, "Computer Arithmetic," in *John Wiley and Sons*, 1979.

[12] G. DeMicheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1994.

[13] G. Constantinides, P. Cheung, and W. Luk, *Synthesis And Optimization Of DSP Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[14] M. Galanis, G. Theodoridis, S. Tragoudas, and C. Goutis, "A High Performance Data-Path for Synthesizing DSP Kernels," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1154–1163, June 2006.

[15] R. Kastner, S. Ogrenci-Memik, E. Bozorgzadeh, and M. Sarrafzadeh, "Instruction Generation for Hybrid Reconfigurable Systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 7, no. 4, pp. 605–627, 2002.

[16] www.synopsys.com/products/designware.

[17] Artisan Components, TSMC 0.13 Library Databook.