15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7, 2007, copyright by EURASIP

A REGULAR INTERCONNECTION SCHEME FOR EFFICIENT MAPPING OF DSP KERNELS INTO RECONFIGURABLE HARDWARE

Sotiris Xydis, George Economakos and Kiamal Pekmestzi

School of Electrical and Computer Engineering, National Technical University of Athens
Iroon Polytexneiou 9, GR-15780 Athens, Greece
phone: + (30) 210-7223653, fax: + (30) 210-7722428, email: sxydis@microlab.ntua.gr
web: www.microlab.ntua.gr

ABSTRACT

This paper presents a design technique for coarse grained reconfigurable cores targeting mostly DSP applications. The proposed technique inlines flexibility into custom Carry-Save-Arithmetic (CSA) datapaths exploiting a stable and canonical interconnection scheme. The canonical interconnection is revealed by a uniformity transformation imposed on the basic architectures of CSA multipliers and CSA chain-adders/subtracters. The design flow for the implementation of the core is analyzed in detail, and the advanced mapping opportunities are presented. The paper concludes with the experimental results showing that our architecture performs an average latency reduction of 32.63%, compared with datapaths of primitive computational resources, with sufficient hardware utilization.

1. INTRODUCTION

The advent of Reconfigurable Computing [4] has generated a whole new research field in the area of digital design. The new computational model augments the available logical density of the circuits, binding the spatial (high parallelism) and the temporal (high programmability-flexibility) models. Along with the high integration densities provided by the current ASIC technologies, we are able to design dynamic configurable hardware systems on a single chip (*Configurable System-on-Chip*, CSoC) [12, 13, 14].

A significant number of reconfigurable architectures have already been proposed, varying mostly on the granularity's degree. An overview of the most popular reconfigurable architectures can be found in [7]. Fine-grained architectures [7, 8] favor bit-level operations and mapping universality, but suffer from high reconfiguration delays and power consumption. Coarse-grained architectures [13, 14] eliminate the disadvantages of fine-grained ones and preserve universality at most cases, but operate only on word-length data formats. Recently, hybrid architectures [10] have been proposed which try to combine the benefits of the two above approaches. All these solutions propose new architectures to enable dynamic hardware reconfiguration. The aim of this work is to enable reconfigurability in already known circuits which incorporates a significant degree of computation density, such as array multipliers.

In this paper, an area and time efficient reconfigurable arithmetic unit (RAU), targeting mainly the DSP domain, is introduced. In precise, onto the datapath of a 16x16 carry-save multiplier we mapped the behaviors of carry-save adder and subtracter enabling single or chained operations. The interconnection scheme, between the cells of the flexible datapath, remains stable at each operation case, through the appliance of the a uniformity transformation.

The proposed architecture provides fast implementations for the set of mapped operations due to the CSA-based logic, which eliminates the time consuming carry-propagation. The fast arithmetic operations and the stable interconnection scheme rise the opportunities for efficient operation chaining. Operation chaining is a well known synthesis technique, from the field of the High Level Synthesis (HLS), which removes the intermediate registers between data-dependent operations improving the total delay of the combined units.

2. RELATED WORK

Many coarse grained reconfigurable architectures have been proposed in bibliography. The Morphosys reconfigurable system [13] is a complete reconfigurable SoC implemented at the layout level [11]. It incorporates a 32-bit RISC processor and a 8x8 array of coarse-grained reconfigurable cells for efficient mapping of DSP applications. The basic reconfigurable cell is universal consisting of an ALU and a MAC unit. The SoC also incorporates a DMA-controller and a Frame buffer for fast data transfers between the memory and the reconfigurable array module.

A coarse grain reconfigurable architecture, which targets DSP applications, is proposed in [5], enabling efficient template-based operation chaining. Every node of the applications' DFG is mapped on a computational resource. The templates are implemented by interconnecting appropriately a number of computational cells. Template-chaining is performed by using a flexible inter-template interconnection network. Although, the proposed architecture seems to have performance gains, the area overheads imposed by the basic template cell architecture are not negligible.

Morphable multipliers proposed in [3] are multi-mode (morphable) functional units (MFU) based on the exploitation of the compressors' timing slack in tree multipliers. The system is configured to perform two operand multiplication or addition. Each desired mode is analyzed separately and the adder chains that will form the adder mode is produced. However, a severe constraint is that only sharing between single addition and multiplication operations is considered, so the applicability set of the produced designs is strict enough.

In [2] synthesis techniques of Multi-Mode (MM) Cores are presented, targeting DSP embedded systems. The initial specification of the desired configurations is formatted as separate DFGs, one for each configuration. The main objective is to design low-power MM units following the conventional ASIC design-flow of existed industrial synthesis tools.

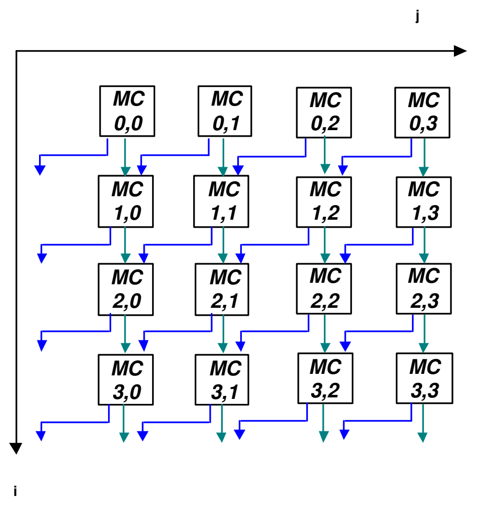


Figure 1: A 4×4 multiplier

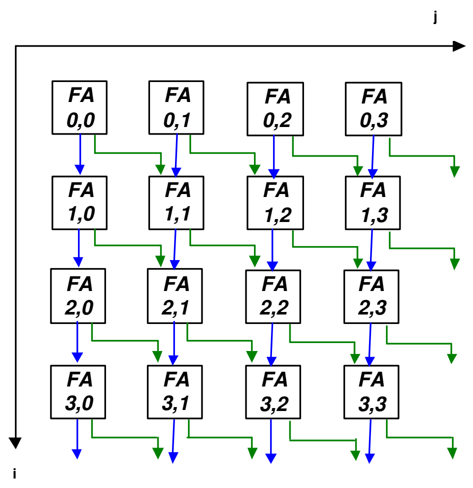


Figure 2: A 4×4 chain-adder

3. EFFICIENT INTERCONNECTION SCHEME

The generality of most reconfigurable architectures is achieved by both interconnection's and functional block's reconfigurability. For interconnections, classical FPGAs use switch matrices while coarse grained reconfigurable arrays use complex bus-based and highly multiplexer-based resources. These complex interconnection schemes impose a significant area overhead and also a reconfiguration delay overhead due to the augmented number of configuration bits needed to control the interconnections.

DSP applications are based mainly on four operative modes, namely, addition, subtraction, multiplication and shifting. This can be easily confirmed by profiling of the the main DSP kernels' dataflow-graphs. Thus, the ability of performing the previously mentioned primitive operations, maintaining temporal flexibility and stable/simple interconnection structure, can expose high area and reconfiguration delays gains comparing with classical coarse grained architectures. This paper describes a technique to map the main DSP operations onto the structure of a modified array multiplier, while maintaining the efficient routing between the initial multiplier's basic cells.

Consider a $N \times N$ array multiplier based on Carry-Save-Adders (CSA) [9]. The array consists of N^2 multiplier cells (MCs) and their interconnections. The structure of such a multiplier is given in figure 1 for $N = 4$. The overall architecture can be described by the following relations:

$$Mul\_Cell_{i,j} : a_j \times b_i \times si_{i,j} \times ci_{i,j} \rightarrow so_{i,j}, co_{i,j} \qquad (1a)$$

$$si_{i,j} = 0, \qquad i = 0 \qquad (1b)$$

$$si_{i,j} = so_{i-1,j+1}, \qquad i \in \{1, N-1\} \qquad (1c)$$

$$ci_{i,j} = 0, \qquad i = 0 \qquad (1d)$$

$$ci_{i,j} = co_{i-1,j}, \qquad i \in \{1, N-1\} \qquad (1e)$$

$$a_j, b_i, si_{i,j}, ci_{i,j}, so_{i,j}, co_{i,j} \in \{0,1\} \qquad (1f)$$

Relation 1a defines that each basic cell is a multi-input-multi-output function targeting the Boolean domain, as imposed by relation 1f. Relations 1b to 1e refer to the routing properties of the CSA multiplier's architecture.

The structure of an array of CSA adders for chain addition is very similar. It also consists of N^2 cells and the basic component of full addition is present. An illustrative example of 4×4 CSA chain-adder is given in figure 2. The corresponding relations are the following:

$$Add\_Cell_{i,j} : a_j \times si_{i,j} \times ci_{i,j} \rightarrow so_{i,j}, co_{i,j} \qquad (2a)$$

$$si_{i,j} = x_{0,j}, \qquad i = 0 \qquad (2b)$$

$$si_{i,j} = so_{i-1,j}, \qquad i \in \{1, N-1\} \qquad (2c)$$

$$ci_{i,j} = y_{0,j}, \qquad i = 0 \qquad (2d)$$

$$ci_{i,j} = co_{i-1,j}, \qquad i \in \{1, N-1\} \qquad (2e)$$

$$a_j, x_{0,j}, y_{0,j}, si_{i,j}, ci_{i,j}, so_{i,j}, co_{i,j} \in \{0,1\} \qquad (2f)$$

The same relations also apply in chain-subtraction, with only difference in the basic cell's truth table. The routing schemes for both the array multiplier and the chain-adder (or chain subtracter), denoted by relations 1c, 1e, 2c and 2e, present specific dissimilarities, as also shown in figures 1 and 2. Mapping together multiplication, addition, subtraction, chain addition and chain subtraction operations on a single circuit, and changing dynamically the configuration between these behaviors can lead to a high performance functional unit, especially for the DSP domain. However, the combinative mapping of the above configurations in a straightforward manner suffers from the described routing dissimilarity, and from the functional dissimilarity occurring between each basic cell configuration context.

Routing dissimilarity imposes a complex interconnection scheme between the cells. Actually, a 2 to 3 switch circuit is needed for every $cell_{i,j}$ to route appropriately the cell's outputs. This scheme is area inefficient, augments the architecture's configuration word-length and imposes a highly complex interconnection routing between the cells.

To tackle the routing dissimilarity we propose a technique based on a uniformity transformation. The transformation applies on the primary inputs' bit-order and on the

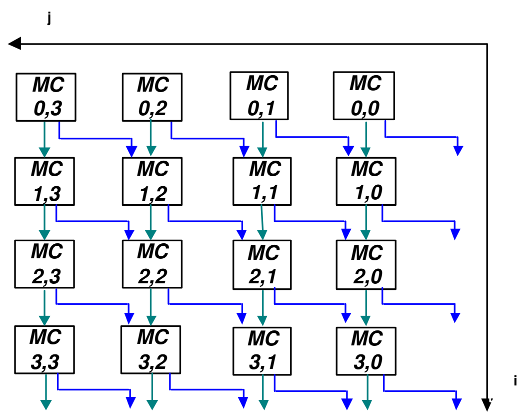
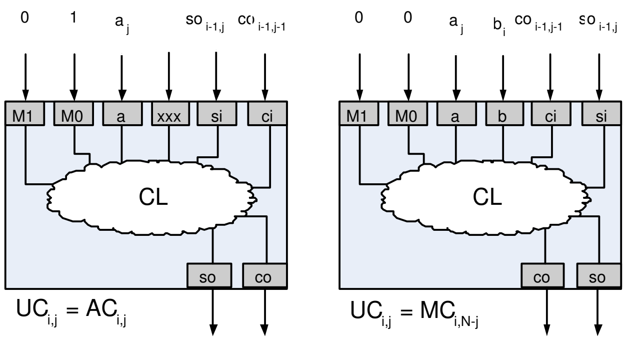


Figure 3: Mirror architecture of array multiplier



(a) AC (b) MC

Figure 4: Port mapping of the unified cell

cell's input and output ports and results to a stable routing scheme among all configurations enabling an efficient combined mapping of the desired behaviors.

The proposed transformation is completed in three stages. At the first stage, a reference architecture is selected. The reference architecture can follow the interconnection scheme of either the array multiplier or the chain-adder (the chain-subtracter has the same interconnection scheme). The interconnection scheme of the reference architecture will remain stable throughout all operation modes. What is changing is the functionality of the basic cell, which is a unified cell (UC) functioning as either a multiplication cell (MC), or as an addition cell (AC), or as a subtraction cell (SC), depending on a set of selection signals $\{M0, M1\}$.

The two interconnection schemes are equally canonical and enable efficient VLSI implementation. The selection is based on the demands of the applications that will be mapped onto the reconfigurable architecture. For example, if one application requires a large number of multiplication operations, it is preferable to select the multiplication scheme as reference architecture. The chain-adder scheme will be selected if additions or subtractions occur more often. The demands of an application set can be found through profiling. For the rest of the paper, without loss of generality, we consider reference architecture the interconnection scheme of the chain-adder (as found in our experiment suite).

In the subsequent stage, we reverse the bit-order of the multiplier's input $a_j, j \in [0, N-1]$, and the mapping order of the basic cells to preserve the correct functionality (figure 3). The resultant architecture is mirror symmetrical to the initial. The relations for the mirrored architecture are obtained by substituting $j = N - j$ in the relations for the multiplier:

$$a_{N-j} \times b_i \times si_{i,N-j} \times ci_{i,N-j} \rightarrow so_{i,N-j}, co_{i,N-j} \tag{3a}$$

$$si_{i,N-j} = 0, \qquad i = 0 \tag{3b}$$

$$si_{i,N-j} = so_{i-1,N-j-1}, \qquad i \in \{1, N-1\} \tag{3c}$$

$$ci_{i,N-j} = 0, \qquad i = 0 \tag{3d}$$

$$ci_{i,N-j} = co_{i-1,N-j}, \qquad i \in \{1, N-1\} \tag{3e}$$

$$a_{N-j}, b_i, si_{i,N-j}, ci_{i,N-j}, so_{i,N-j}, co_{i,N-j} \in \{0,1\} \tag{3f}$$

These new relations indicate that each $cell_{i,j}$'s input signals for both chain-adder and mirrored-multiplier structures are issued by the same cells of the previous row. Therefore, the routing dissimilarity constraint is partially relaxed, due to the fact that the input signals now have the same sources. However, each source is still attached to different ports on the sink cell. The different port attachment imposes the need for two 2-to-1 multiplexers under each cell to drive the inputs in a proper way. Elimination of these multiplexers is performed by the third stage of the proposed transformation.

In the UC structure, a straightforward port assignment generates the need of the internal multiplex usage. This happens because port *si* in the chain-adder's case has input from the right above cell, but in multipliers case it has inputs from the above left cell. An equivalent condition holds for the port *ci*. To overcome this limitation, we propose to keep steady the unified cell's external port interface and alter the inter-cell port mapping. The altering can be made as a simple permutation operation on the UC's internal port instantiation mapping, as shown in figure 4.

These three stages form the proposed uniformity transformation. The transformation exposes a canonical routing scheme for efficient mapping of multiple arithmetic behaviors, on the same architecture, minimizing the routing complexity and circuits criticality imposed by interconnection dissimilarities. The feasibility of multiple behavior mapping generates opportunities of flexible, run time configurable area/time efficient architectures targeting the DSP domain.

4. DSP APPLICATION MAPPING

Based on the technique of the previous section, we have constructed a *Reconfigurable Arithmetic Unit* (RAU) on which we mapped DSP kernels. The RAU consists of a 16×12 array of UCs, a context register, four input reversing-order modules, a carry propagate adder/subtracter and a register file for intermediate variables' storage.

The architecture can be instantaneously configured either as one multi-cycle pipelined functional unit or as three standalone and independent single cycle computational units. Its 3-stage pipelined structure permits high throughput applications to be mapped and a high utilization degree between the independent stages. At the beginning of each stage there is a multiplexer based intermediate line which enables the proper routing of the pipelines inputs. The configuration is guided by the context register which drives all the allocated mul-

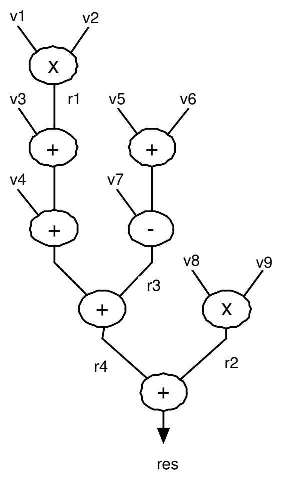


Figure 5: Application DFG

tiplexers. A 16×12 multiplication can be mapped as a 3 cycle pipelined CSA multiplier. It can be clocked by high frequency clocks because of the its pipelined structure and its CSA architecture. Comparing the maximum clock periods between the RAU and an un-pipelined CSA multiplier, the former yields in ASIC technology about 3× (or 2,3× in FPGA technology) faster clock rates.

Several DSP applications can be mapped onto the RAU. For example, consider the DFG of figure 5. Figure 6 shows a possible mapping. In each control step 3 processing units are available Seq 1, Seq 2, Seq 3, corresponding to the architecture's pipeline stages. The multiplication operations are performed in three control steps (indexes a, b and c) and the final result is obtained from the last pipeline segment. The storage requirements are reduced due to operation chaining and next-stage data forwarding capabilities.

5. EXPERIMENTAL RESULTS

In this section, we demonstrate the effectiveness of our architecture in terms of latency reduction and hardware utilization. The testbench suite includes seven application DFGs, coming from the DSP domain (benchmarks found in [1] and a 4 point FFT). Initially, the DFGs and the synthesizable RTL implementations of all benchmarks were evaluated using the SPARK HLS framework [6]. They were coded in C inserted into SPARK with the resource constraint of 2 ALU and 2 multiply units. This resource constraint was selected because of the similarities in hardware functionality and complexity with our architecture. Next we formed an experimental setup for our architecture, a 4-way pipelined 16x16 RAU with the first three pipeline segments being configurable and the last non-configurable. A top level FSM module was instantiated, to emulate the data busses and the configuration bus, providing appropriate input data and configuration words to the reconfigurable kernel. The DFGs, obtained from SPARK, were manually mapped to our reconfigurable architecture.

The clock period of the primitive resource datapath, T_{pr}, was set to the delay of the multiplier unit. We assume that the ALU unit operates in $T_{pr}/2$ of the period. Our reconfigurable arithmetic unit is able to operate in $\simeq T_{pr}/4$ due to the pipelined architectural structure that it integrates. So, taking into account the extra multiplexer lines and the unified cell

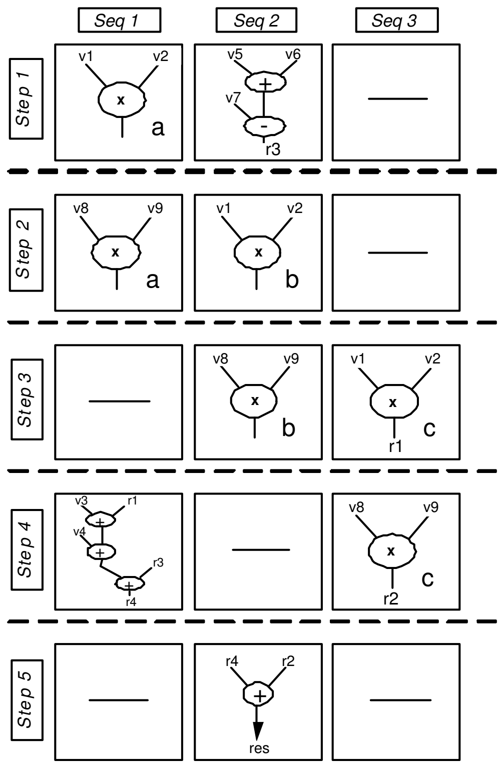


Figure 6: DFG mapping onto the proposed architecture

structure introduced in RAU's datapath, a more precise value of the clock period is $T_{RAU} = T_{pr}/3$.

Figure 7 shows comparative performance results with the datapaths generated by SPARK. Our architecture achieves latency reduction in all cases. The speedups range between 27%-54%, with an average speedup of $\simeq$ 32.63%. The DCT benchmark presents the highest performance gain due to the fact that many ALU-based operation chain opportunities have been found in it. Furthermore, these operation chaining opportunities can be extended by allowing the same computation to be performed simultaneously in more than one computational segment of the RAU. This feature is fully supported by our architecture and enabled in all benchmarks. Accordingly, the lowest performance gains are reported for the FIR4 and FIR7 filter mainly because of the tight data dependencies between multiplication and ALU operations,

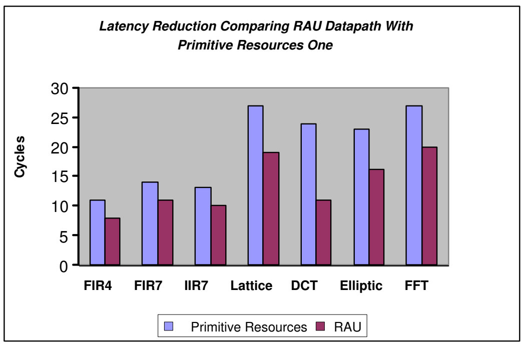


Figure 7: Latency reduction using the RAU datapath

Table 1: Latency and usage result comparison

DFG	DFG nodes	Reconf. Arith. Unit	Usage ratio A	Usage ratio B
Fir4	7	8	62.5	66.5
Fir7	17	10	67.5	70
Iir7	18	10	71	75
Lattice	24	19	58	61
DCT	44	11	84.5	97
Elliptic	39	16	48	64.6
FFT	41	20	65	69

minimizing the operation chaining in RAU.

Table 1 shows the number of DFG nodes extracted from each benchmark, the clock cycles needed for execution onto the proposed reconfigurable architecture and finally its usage ratios. The usage ratio metric depicts the utilization of RAU's computational segments. Usage Ratio A considers all the four computational segments along with the last non-configurable segment, while Usage Ratio B comprises only the utilization of the three reconfigurable ones. In both cases, usage ratio is calculated according to the type:

$$RAU_{Usage} = \frac{(\#Cycles \cdot PEs) - Idle\,PEs}{\#Cycles \cdot PEs} \cdot 100\% \qquad (4)$$

The *PEs* is the number of available computational segments while the *#Cycles* refers to the number of cycles required to schedule each DFG onto RAU. The *Idle PEs* is the sum of the idle computational segments in each cycle.

Usage Ratio A ranges from 48% to 84.5%, with an average value of 65.2%. Respectively, Usage Ratio B ranges from 61% to 97%, with an average value of 71.9%. In general, the utilization degree of the RAU's resources is high in both cases. All the computational segments have been utilized in every DFG example. The difference between the average values of Usage Ratio A and Usage Ratio B is reasonable enough, since the last non-configurable computational stage is used only at multiplication operations.

The Elliptic filter performed the lowest value of Usage Ratios among all benchmarks, in one case even below 50% (48% Usage Ratio A). This happens because the Elliptic filter's DFG is dominated by node sequences of low parallelism. This fact along with the high operation chaining opportunities revealed by the structure of the specific DFG limit the number of reconfigurable PEs used in parallel per cycle. However, considering all other cases the Elliptic filter can be considered as an exception to the general rule of high utilization degrees.

6. CONCLUSION AND FUTURE WORK

In this paper a technique to generate a reconfigurable architecture with a canonical interconnection scheme has been presented. The resulting architecture achieves significant performance gains over a set of DSP benchmarks. Our future work will focus on the development of an automated mapping methodology for efficient operation scheduling and binding onto our architecture, taking into consideration the RAU's specific features.

REFERENCES

[1] CDFG toolset, http://poppy.snu.ac.kr/CDFG/cdfg.html.

[2] L. Chiou, S. Bhunia, and K. Roy. Synthesis of Application-Specific Highly Efficient Multi-mode Cores for Embedded Systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 4(1):168–188, 2005.

[3] Chiricescu, Schuette, Glinton, and Schmit. Morphable Multipliers. In *Proc. of the International Conference on Field Programmable Logic and Applications*, pages 647–656, 2002.

[4] K. Compton and S. Hauck. Reconfigurable Computing: A Survey of Systems and Software. *ACM Computing Surveys*, 34(2):171–210, 2002.

[5] 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*, 25(6):1154–1163, June 2006.

[6] S. Gupta, N. Savoiu, N. Dutt, R. Gupta, and A. Nicolau. Using Global Code Motions to Improve the Quality of Results for High-Level Synthesis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(2), Feb. 2003.

[7] R. Hartenstein. A Decade of Reconfigurable Computing: A Visionary Retrospective. In *Proc. of ACM/IEEE Design Automation and Test in Europe Conference*, pages 642–649, 2001.

[8] S. Hauck, T. Fry, M. Hosler, and J. Kao. The Chimaera Reconfigurable Functional Unit. In *Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines*, pages 87–96, 1997.

[9] K. Hwang. Computer Arithmetic. In *John Wiley and Sons*, 1979.

[10] R. Kastner, S. Ogrenci-Memik, E. Bozorgzadeh, and M. Sarrafzadeh. Instruction Generation for Hybrid Reconfigurable Systems. *ACM Transactions on Design Automation of Electronic Systems*, 7(4):605–627, 2002.

[11] M.Lee, H. Singh, G. Lu, N. Bagherzadeh, and F. Kurdahi. Design and Implementation of the MorphoSys Reconfigurable Computing Processor. *in Journal of VLSI Signal Processing Systems, Kluwer*, 24:147–164, 2000.

[12] A. Olugbon, S. Khawam, T. Arslan, I. Nousias, and I. Lindsay. An AMBA AHB-based Reconfigurable SoC Architecture using Multiplicity of Dedicated Flyby DMA Blocks. In *Proc. of ASP-DAC*, pages 1256–1259, 2005.

[13] H. Singh, M. Lee, F. K. G. Lu, N. Bagherzadeh, and E. Filho. Morphosys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications. *in IEEE Trans. on Computers*, 49(5):465–481, May 2000.

[14] S. Wallner. A configurable System-on-Chip Architecture for Embedded and Real-Time Applications: Concepts, Design and Realization. *Elsevier Journal of Systems Architecture*, 51(6-7):350–357, 2005.