# Embedded Processor Characteristics Specification Through Multiobjective Evolutionary Algorithms

K.Ghali [1] and O.Hammami [2]
ENSTA
32 Bvd Victor
75739 Paris
FRANCE
{ghali, hammami}@ensta.fr
[1] IEEE Student member [2] IEEE member

***Abstract*—The design of a superscalar microprocessor for a given workload is a tremendous task by itself due to the numerous parameters involved and the ranges of their possible values. If power consumption and area are also to be considered then the problem is even more complicated and requires a suitable framework and methodology for exploring the vast multidimensional space for such a problem. In this paper we propose such a framework based on multi-objective evolutionary algorithms and demonstrate its use on a significant size example.**

*Index Terms*—Embedded, multi-objective, processor.

## I. INTRODUCTION

Intellectual properties are increasingly used in system on chip designs and involve complex function such as microprocessors, DSP and various specialized functions. The advent of system level design methodologies and languages stimulate the use of soft IP described in a high level language such as SystemC and which are therefore more amenable to parameterization. Several vendors propose parameterizable microprocessors with associated software tools. However parameterization is workload driven as well as design constraints dominated by for example chip area and power consumption. It is impossible to manually explore the huge space of the all possible configurations and clearly an automatic tool for such a purpose would be of high value. The problem at hand is the automatic exploration and solutions search in a multidimensional solution space for the selection of the best characteristics of an ASIP (Application Specific Integrated Processor).

## II. MOEA and NSGA-II

Multi-objective evolutionary algorithms (MOEA) tackle problems with multiple objectives and produce a set of optimal solutions known as the Pareto-optimal solutions. The MOEA share many of the basic features of genetic algorithms such as the concept of population, individuals, fitness function although in a multidimensional space. A number of multi-objective evolutionary algorithms have been proposed over the past decade [2,3] among them the PAES and SPEA algorithms. Recently a new MOEA the NSGA-II algorithm [4] was proposed which outperforms both PAES (Pareto-archived evolution strategy) and SPEA (strength Pareto EAs) multi-objective EAs. We selected the NSGA-II algorithm to conduct experimental studies in the framework of embedded processor microarchitecture features selection under the constraints of power consumption, execution time and area. The NSGA-II was implemented using the associated NSGA-II Library and accordingly modified to interface with various software tools used to evaluate the three main parameters: (1) power consumption, (2) area estimation (3) execution time. The NSGA-II algorithm was executed on a pool of 48 networked PCs and produced very quick results.

### *A. Genome Structure*

The characteristics of a genome includes cache organization, number of floating point and integer functional units, pipeline organization (fetch, decode, issue).

### *B. Power Consumption*

Power consumption is estimated with the help of the Wattch tool. Wattch is a framework for analyzing and optimizing microprocessor power dissipation at the architecture-level. Wattch is 1000X or more faster than existing layout-level power tools, and yet maintains accuracy within 10% of their estimates as verified using industry tools on leading-edge designs.

### *C. Area Estimation*

Area estimation of superscalar processor configuration was approximated by the sum of caches area, the floating point units and individual computation units. The first was evaluated with the CACTI tool, the second with the FUPA Stanford utility and finally the computation units were evaluated based on published data in the literature.

### *D. Execution Time*

Execution time was evaluated the SimpleScalar superscalar simulator which is thoroughly described in the next section.

The three values are the values which characterize the evaluation of an individual in the framework of NSGA-II. Classification of all the solutions is based on those characteristics.

0-7803-7912-8/03/$17.00 © 2003 IEEE

## III. TURBO-CODERS

Turbo coders have been proposed for several embedded applications in the field of wireless third generation mobile telecommunications (UMTS), satellite and deep spaced communications and even for disk drives. All those applications need high coding gains and performances close to Shannon's limit in terms of error bit rate. Those gains and performances can be achieved through the use of convolutional concatenated codes. A concatenated encoder is composed of two or more recursive and systematic convolutional encoders connected in an encoding network. Due to the high computation of turbo-codes several IP and VLSI implementation have been proposed to achieve high throughput. In this paper we selected the turbo-coding as our workload on which to apply the NSGA-II algorithm for the purpose of an embedded processor software implementation.

## IV. SUPERSCALAR SIMULATOR: SIMPLESCALAR

SimpleScalar is an execution driven cycle accurate instruction set simulator (ISS) of a superscalar microprocessor [10]. A complete development chain (compiler, debugger, profiler) comes with the tool which allows the quick porting of any ANSI C application to SimpleScalar. The SimpleScalar toolset is composed of agcc compiler ported for the SimpleScalar architecture which generates SimpleScalar binary files. The SimpleScalar assembler and loader along with the necessary ported libraries produce SimpleScalar executables that can be fed directly into one of the provided simulators. The simulator themselves are compiled with the host platform's native ANSI C compiler. The support libraries can be modified in that case the glibc source must be installed, modified and built. The simulators come equipped with their own loader and thus you do not need to build the GNU binary libraries to run simulations. The toolset comes with a variety of simulators ranging from untimed functional simulators to cycle-accurate complex simulators.

**Table 1 – SimpleScalar Simulators**

| Simulator | Description |
|---|---|
| *sim-fast* | Functional simulator (No time accounting) |
| *sim-safe* | Functional simulator + alignment and access permissions checking |
| *sim-cache/sim-cheetah* | cache simulators |
| *sim-profile* | Functional simulator + profiler |
| *sim-outorder* | Cycle accurate speculative out of order superscalar simulator |

The above tools represent the SimpleScalar toolset.

The SimpleScalar microprocessor micro-architecture is derived from the MIPS-IV ISA with a semantics which is a superset of MIPS with a few exceptions: (1) delay slots are not used therefore instructions succeeding load, stores and control transfers are not executed, (2) load and stores support 2 addressing modes: indexed and auto-increment/decrement (3) a square root instruction is included and (4) there is an extended 64 bits instruction encoding.

The SimpleScalar microprocessor models an out-of-order superscalar architecture based on a RUU (Register Update Unit). The RUU exploits a reorder buffer to automatically rename registers and hold the results of pending instructions. However, completed instructions are retired in program order to the register file. The microarchitecture supports speculative execution. The memory system uses a load/store queue and store values are placed in this queue if the store is speculative. Load instructions are dispatched to the memory system when the addresses of all previous stores are known and loads may be satisfied either by the memory system or by an earlier store value reading in the queue if their addresses match. Speculative loads may generate cache misses but speculative TLB misses stall the pipeline until the branch condition is known. The rich set of the various tunable cache parameters of the processor are described in the following tables.

**Table 2 – Processor Caches Parameters**

| Cache Parameters | Default value | comments |
|---|---|---|
| *Cache parameters (nsets, bsize assoc, , repl)* | D: 128/32/4/l<br>I: 512/32/1/l | Basic cache parameters: associativity, block size nbr of sets, replacement policy |
| *dl1lat* | 1 cycle | Hit latency of L1 cache |
| *dl2lat*<br>*lat* | 6 cycles<br>18 cycles | Hit latency of L2 cache<br>Main memory access latency |
| *width* | 8 bytes | Main memory width |

The parameters of processor core are described in Table.3

**Table 3 – Processor Core Parameters**

| Processor Parameters | Default Values | Comments |
|---|---|---|
| *decode width* | 4 | decode width |
| *issue width* | 4 | max issue width |
| *ruu* | 16 | capacity of the RUU in instructions |
| *lsq* | 8 | capacity of the load/store queue |
| *ifqsize* | 4 | fetch width |
| *ialu* | | Nbr of int ALUs |
| *imult* | 1 | Nbr of int mult/dividers |
| *memports* | 2 | Nbr of L1 cache ports |
| *fpalu* | 4 | Nbr of floating point ALUs |
| *fpmult* | 1 | Nbr of floating point mult/dividers |

## V. EXPERIMENTAL RESULTS

We applied the NSGA-II algorithm on the Turbo-decoder application and explored the impact of population size and number of generations. The population size value set was P = {13, 23, 33} and the number of generations value set was G = {5, 10, 20, 50, 100, 200}. All the results are described below.

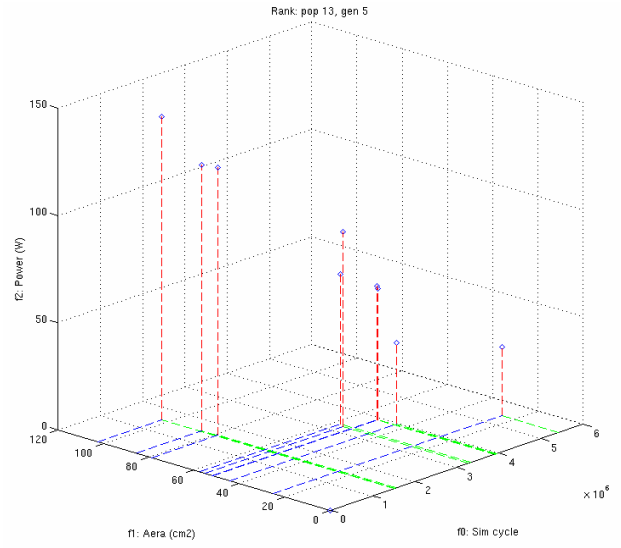


Fig.1 For 5 generations – popsize = 13

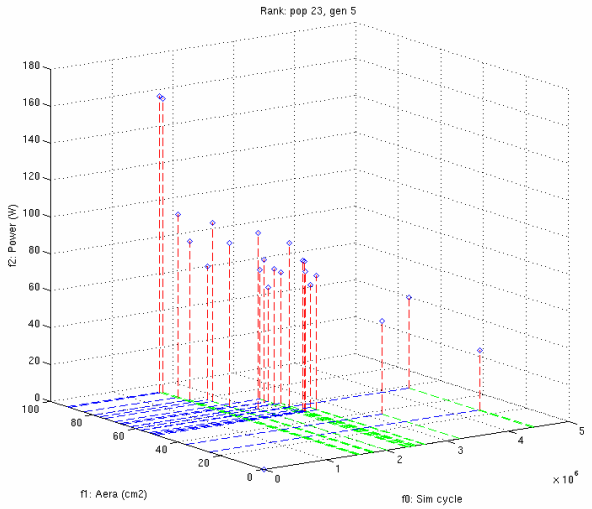


Fig. 2 For 5 generations – popsize = 23

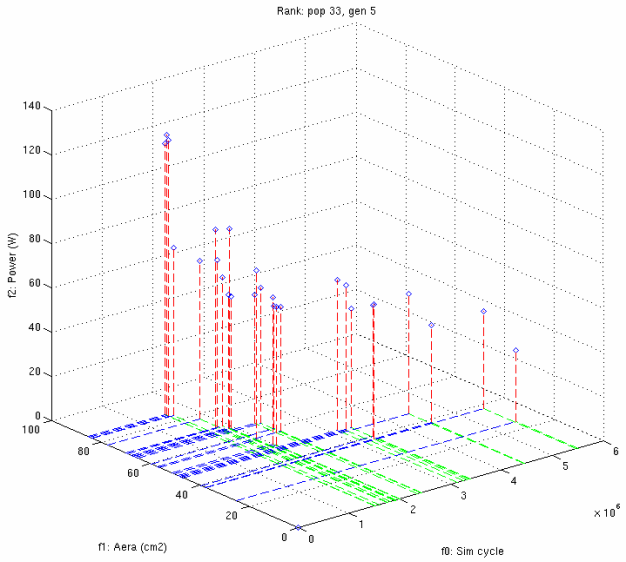


Fig.3 For 5 generations – popsize = 33

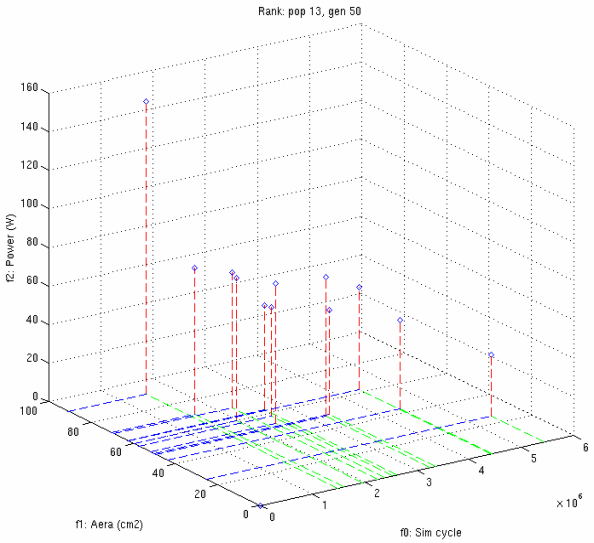


Fig.4 For 50 generations – popsize = 13

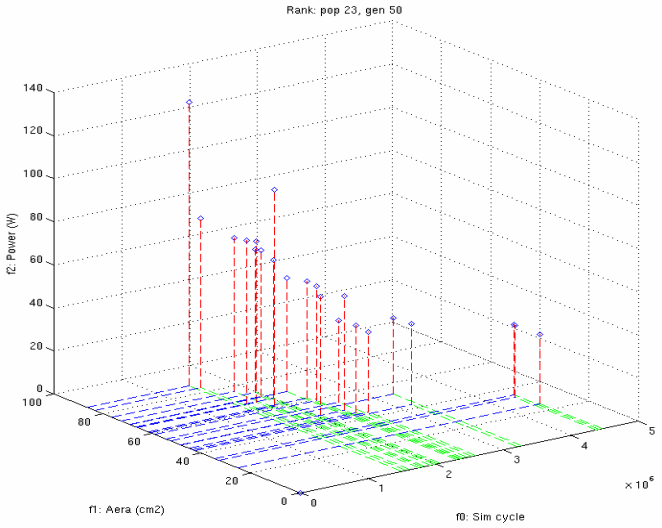


Fig.5 For 50 generations – popsize = 23

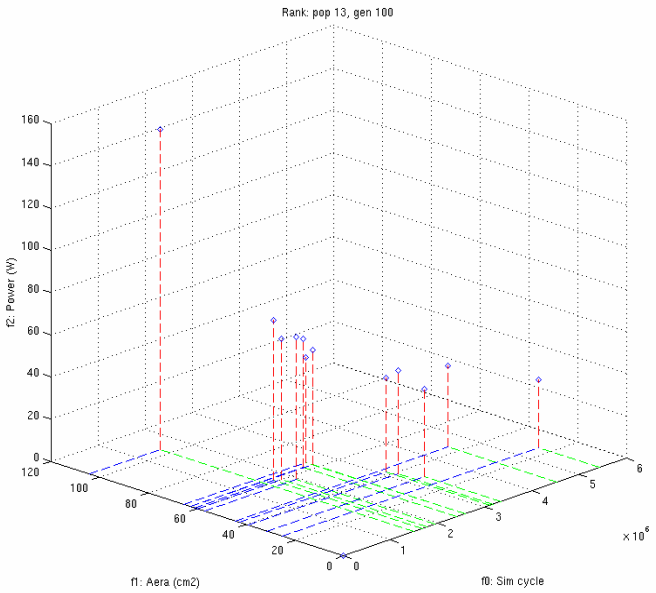


Fig.7 For 100 generations – popsize = 13

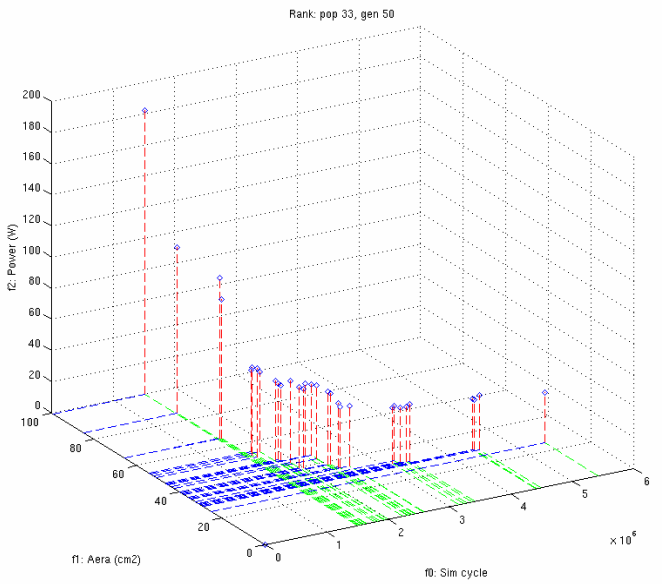


Fig.6 For 50 generations – popsize = 33

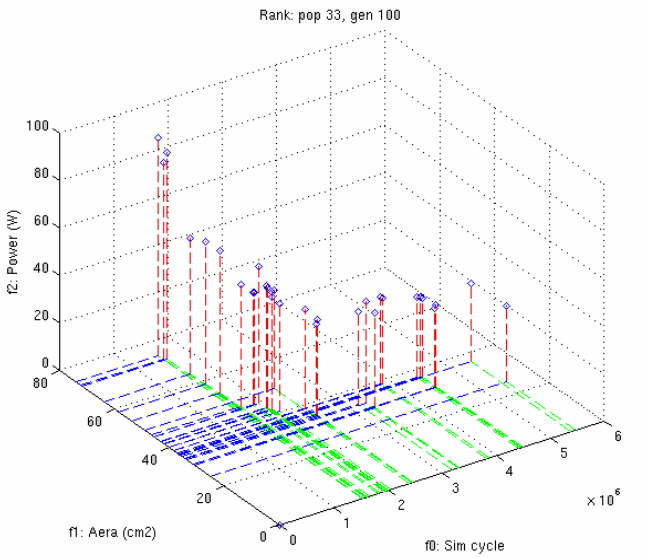


Fig.8 For 100 generations – popsize = 23

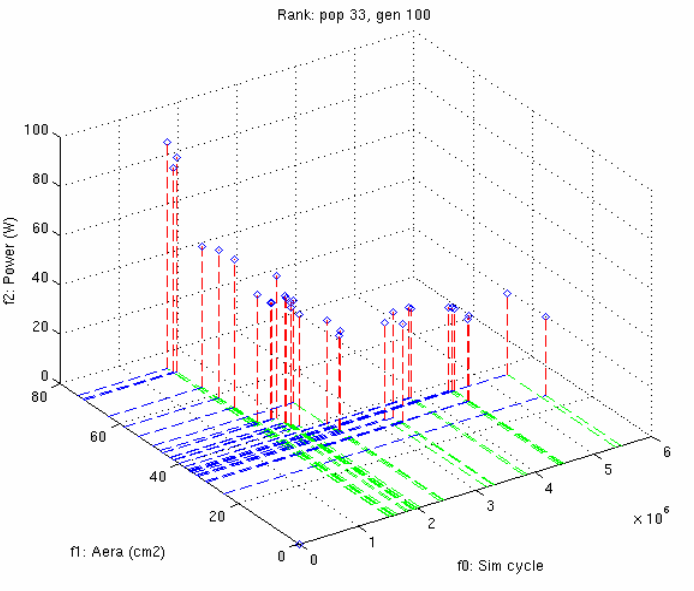


Fig.9 For 100 generations – popsize = 33

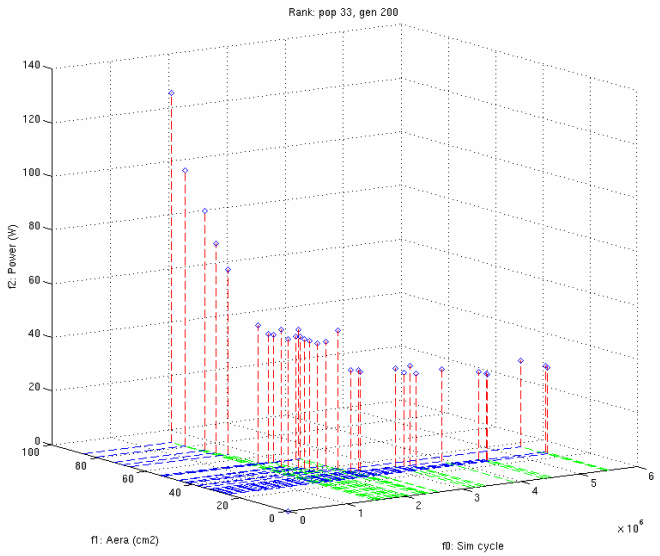


Fig.11 For 200 generations – popsize = 33

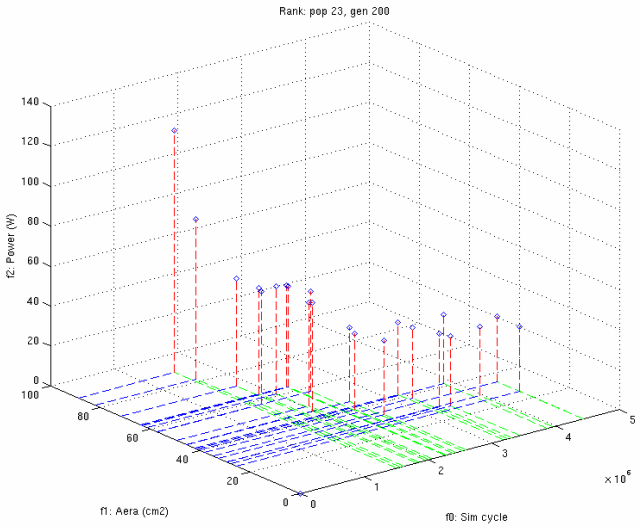


Fig.10 200 generations – popsize = 23

## VI. CONCLUSION

We presented in this paper a method for the production of ideal genome configurations in a multidimensional space for the optimization of execution time, surface area and power consumption. This methodology can easily be extended to take into account more micro-architectural parameters and more applications. Clearly the impact of population size affects the quality and density of the results as demonstrated in this paper. Those results stimulate an incremental approach to design automation where simulation accuracy is only emphasized on useful configurations.

## VII. 0.REFERENCES


[1] Luc Semeria, Andrew Seawright, Renu Mehra, Daniel Ng, Arjuna Ekanayake, Barry Pangrle, *'RTL C-Based Methodology for Designing and Verifying a Multi-Threaded Processor'*, 39th DAC, june 10-14, 2002.

[2] C. A. Coello Coello, D.A. Van Veldhuizen, G. B. Lamont , Evolutionary Algorithms for Solving Multi-Objective Problems , Vol 5, **Genetic Algorithms and Evolutionary Computation** Kluwer Academic Publishers; ISBN: 0306467623

[3] E.Zitzler, K.Deb and L.Thiele, ,'' Multiobjective Evolutionnary Algorithms: A Comparative Case Study and the Strength Pareto Approach '', IEEE Trans. On Evolutionary Computation'', pp.257-271, 3(4), 1999.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, T. Meyarivan, *'A Fast and Elistist Multi-Objective Genetic Algorithm: NSGA-II'*, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur .

[5] J.Soerensen, P.Birk and Z.Zvonar, "New Challenges for Integrated Circuits Solutions", Wireless Personal Communications, Vol. 17, pp. 149-153, June 2001.

[6] F.Vahid and T.Givargis, “Platform Tuning for Embedded Systems Design”, IEEE *Computer*, Vol.34, No.3, March 2001, pp. 112-114.

[7] B.Ramakrsihan Rau and M.S.Schlansker, “ Embeded Computer Architecture and Automation”, IEEE Computer, Vol.34, No.4, April 2001.

[8] K.Keutzer, S.Malik, A.Richard Newton, J.M.Rabey and A.Sangiovanni-Vincentelli, “ System-level Design: Orthogonalization of Concerns and Platform-Based- Design”, IEEE Trans. On Computer Aided-Design of Integrated Circuits and Systems, Vol.19, No.12, Dec.2000.

[9] T.Y.Yen and al., “Hardware-Software Co-synthesis of Distributed Embedded Systems” , Kluwer, 1997.

[10] R.E.Gonzalez, “Xtensa: A Configurable and Extensible Processor”, IEEE Micro, Mar./Apr. 2000, pp.60-70.

[11] Simplescalar Manual: **www.simplescalar.org**

[12] M.F.Jacome and H.P.Peixoto, “ A Survey of Digital Design Reuse”, IEEE Design & Test of Computers, pp.98-107, May-June 2001.

[13] D. Brooks, V. Tiwari, and M. Martonosi, "***Wattch: A Framework for Architectural-Level Power Analysis and Optimizations***," Proc. 27th. Int'l Symp. on Computer Architecture, pp. 83--94, June 2000.

[14] Patrick Robertson, Peter Hoeher, Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding, European Trans. On Telecommun.,Vol. 8, No. 2, March-April 1997, p. 119-125.

[15] Joachim Hagenauer, Elke Offer, and Lutz Papke, Iterative decoding of binary block and convolutional codes, IEEE Trans. Inform. Theory, Vol. 42,No. 2, March 1996, p. 429-445.

[16] G.De Micheli, R. Ernst, and W. Wolf, *Readings in Hardware/Software Codesign*, Morgan Haufmann Publisher, San Mateo, CA, 2001.

[17] A. Wang, E. Killian, D. Maydan, and C. Rowen, “Hardware/software instruction set configurability for system-on-chip processor” in Proc. Design Automation Conf., June 2001, pp.184-188.

[18] Xtensa microprocessor, Tensilica nc.(http://www.tensilica.com).

[19] ARCtangent processor, Arc International (http://www.arc.com).

[20] Jazz DSP, Improv Systems Inc. (http://www.improvsys.com).

[21] SP-5flex DSP core, 3DSP Corp. (http://www.3dsp.com).

[22] R. Sucher, “Carmel: A configurable long instruction word DSP core” in Microprocessor Forum, Oct. 1998.

[23] J. A. Fisher, “Customized instruction sets for embedded processors” in Proc Design Automation Conf. June 1999, pp. 253-257.

[24] R. Cloutier and D.E. Thomas, “Synthesis of pipelined instruction set processors” in Proc. Design automation Conf., June 1993.

[25] I. J. Huang and A. M. Despain, “Generating instruction sets and microarchitectures from applications” in Proc. Int. Conf. Computer-Aided Design, Nov. 1994, pp. 391-396.

[26] A. Kitajima, M. Itoh, J. Sato, A. Shiomi, Y. Takeuchi, and M. Imai, “Effectiveness of the ASIP design system PESA-III in design of pipelined processors”, in Proc. Asia south Pacific Design Automation Conf., Jan. 2001, pp. 649-654.

[27] . Choi, J. H. Yi, J. Y. Lee, I. C. Park, and C. M. Kyng, “Exploiting intellectual properties in ASIP designs for embedded DSP software” in Proc. Design Automation Conf., June 1999, pp. 939-944.

[28] A. Pyttel, A. Sedlmeier, and C. Veith, « PSCP : A scalable parallel ASIP architecture for reactive systems” in Proc. Design Automation Test Europe Conf., Mar. 1998.

[29] V. S. Lapinski, Algorithms for Compiler-assisted Design Space Exploration of Clustred VLIW ASIP datapaths, Ph D. thesis, University of Texas at Austin, May 2001.

[30] S. Aditya, B. R. Rau, and V. Kathail, “Automatic architectural synthesis of VLIW and EPIC processors”, in Proc. Int. Symp. System-level Synthesis, Nov. 1999, pp. 107-113.

[31] W. Zhao and C. A. Papachristou, “An evolution programming approach on multiple behaviours for the design of application specific programmable processors”, in Proc. European design Test Conf., Mar. 1996, pp. 144-150.

[32] K. Kim, R. Karri, and M. Potkonjak, “Synthesis of application specific programmable processors“, in Proc. Design Automation Conf. June 1997, pp. 353-358.

[33] K. Kucukcakar, “An ASIP design methodology for embedded system”, In Proc. Int. Symp. HW/SW Codesign . May 1999, pp. 17-21.

[34] H. Choi, J. S.Kim, C. W. Yoon, I. C. Park, S. H. Hwang, and C. M. Kyung, “Synthesis of application specific instructions for embedded DSP software”, IEEE Trans. Computers, vol. 48, no. 6, pp. 603-614, June 1999.

[35] I. J. Huang and A. M. Despain, “Synthesis of instruction sets for pipelined microprocessors”, in Proc. Design Automation Conf. ,June 1994.