ENERGY-PERFORMANCE EXPLORATION OF A CGA-BASED SDR PROCESSOR

David Novo (IMEC, Leuven, Belgium; novo@imec.be); Bruno Bougard (IMEC, Leuven, Belgium; bougardb@imec.be); Praveen Raghavan (IMEC, Leuven, Belgium; ragha@imec.be); Thomas Schuster (IMEC, Leuven, Belgium; schuster@imec.be); Hong-Seok Kim (Samsung; hong-seok.kim @samsung.com) ; Ho Yang (Samsung; hoyang@samsung.com); Liesbet Van der Perre (IMEC, Leuven, Belgium; vdperre@imec.be)

ABSTRACT

Software-Defined Radio (SDR) enables cost-effective multi-mode terminals. However, the growing complexity of the new communication standards together with the reduced energy budget required by battery-powered devices challenge architects. Coarse Grain Array (CGA) processors are strong candidates to undertake both high-performance and low power.

In this paper, we present an extensive energy-performance exploration of a CGA-based SDR processor. We stress the importance of trading off different sources of parallelism, such as data and instruction level parallelism, to achieve the required performance at minimum energy cost. The resulting instantiation is able to execute the symbol-based base-band processing of a 108 Mbps Space Division Multiplexing (SDM) OFDM WLAN receiver with an estimated average power consumption of 110 mW in 90nm CMOS technology.

1. INTRODUCTION

Future handhelds will need to concurrently support a wide variety of wireless communication standards. This can not be implemented by the traditional approach of multiplying hardware and software without increasing the cost of the terminal. Software Defined Radio (SDR), where the baseband processing is deployed on programmable and/or reconfigurable hardware, promises short time-to-market, rapid product derivate development and long product cycle [1]. However, the challenge still resides in the instantiation of specific architectures able to cope with high complex wireless algorithms while keeping programmability and reasonable battery life time.

In order to achieve the required high performance at reasonable energy budget, architecture parallelism must be increased [2]. Parallelism can be provided in several ways: by adding more issue slots [3], wider SIMD (Single Instruction Multiple Data) datapaths [4] or hybrid combination of both [5, 6]. Given that the appropriate combination of parallelism sources is application-dependent, fast exploratory methodologies, which enable cheap iterations in early stages of the design flow, become crucial to guarantee the efficiency and attractiveness of the SDR approach.

In this paper, we present an extensive energy-performance exploration of a CGA templated architecture, which is a strong candidate to undertake the demanding SDR requirements [7]. Since both Data Level Parallelism (DLP) and Instruction Level Parallelism (ILP) have different affects, we explore both types of parallelism and their influence on performance and energy. We also illustrate that by varying processor micro-architecture to trade off DLP and ILP, both software cycle count and hardware critical path are considerably affected. Consequently, optimization metrics in terms of absolute time and absolute energy, instead of clock cycles and power, must be considered for such architecture explorations. For this reason we model each of the CGA architecture components to extract basic performance and energy metrics, based on which we develop a parametrical model of broader design space exploration. The exploratory methodology is applied for the optimization of a CGA processor targeting the symbol-based baseband processing of a 108 Mbps SDM OFDM WLAN receiver.

The remainder of the paper is organized as follows. In Section 2, we introduce the CGA framework. In Section 3, we present the selected application driver. In Section 4 we detail the exploration methodology. We show the results and further analyze them in Section 5. Finally, in Section 6 we draw our conclusions.

2. CGA ARCHITECTURE EXPLORATION FRAMEWORK

ADRES [8], the considered CGA framework, consist of a templated array of densely interconnected functional units which have local register-file and configuration memory (loop buffer). A limited subset of those units is instead connected to a shared multi-ported register-file, enabling their operation also as standard VLIW (Very Long Instruction Word) processor (see Fig. 1).

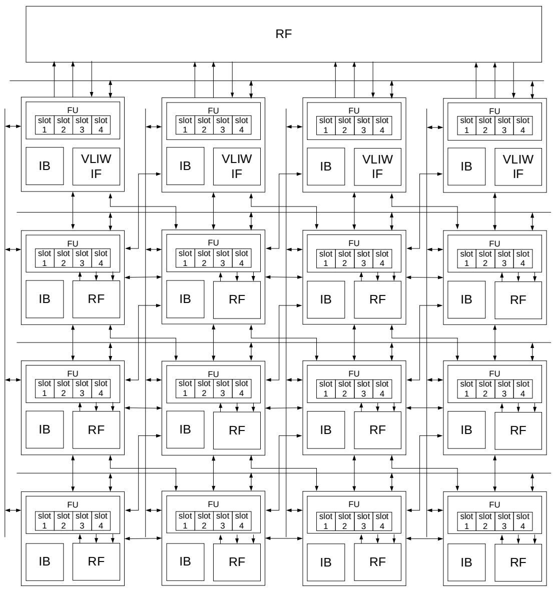

Figure 1. 4x4 ADRES instance with SIMD4.

A retargetable C compiler used for the application mapping, called DRESC [9], targets both the VLIW and CGA modes. DRESC transparently maps data flow dominated loops into the CGA whereas schedules the rest of the code in the VLIW part. The compiler supports different architectures described on an XML architecture file. Application source code can therefore be compiled directly onto the coarse-grained reconfigurable processor. DRESC exploits loop-level parallelism to achieve high ILP by modulo scheduling, a widely used software pipelining technique [10]. Modulo scheduling executes multiple iterations of a loop in parallel. On the other hand, DLP extraction is not yet automated and is inserted into the program by the programmer using intrinsic C-functions.

3. APPLICATION DRIVER

In wireless communications the link capacity can be dramatically improved using multiple antennas. When both the transmitter and the receiver are equipped with multiple antennas, this is commonly referred to as Multiple Inputs, Multiple Outputs (MIMO) communication. SDM is a type of MIMO encoding which transmits two streams in the same bandwidth and type slot with two different antennas. The interference is cancelled either by pre-coding the signal at the transmitter, or by interference cancellation at the receiver. We consider the second option. We identify the SDM mode of IEEE 802.11.n (see Fig. 2) as the currently most compute intensive mode for SDR terminals [11]. The final ADRES instance should, at least, be capable to run this mode respecting latency (SIFS < 16 µs) and real-time constrains (average symbol processing time < 4µs) [12].

SDM with receiver processing splits the user data into two streams of OFDM symbols. Each stream is transmitted synchronously by an antenna (doubling the data rate for the user). These streams experience delay, fading and inter-stream interference while traveling through the channel. At the receiver, two antennas receive both streams summed up together. Based on these received symbols, the receiver recovers the transmitted streams.

The considered functionality, highlighted in Fig. 2, deploys the continuous symbol-based processing. The latter starts just after the coarse synchronization which spots the OFDM symbols and consists of 4 major kernels:

- *FFT*: 64 points complex Fast Fourier Transform (FFT).
- *Tracking*: Fine frequency synchronization that corrects the error of the initial coarse estimate.
- *Spatial equalizer*: SDM equalizer that neutralizes the channel distortion. Its implementation is based on complex matrix multiplications.
- *Demapper*: 64-QAM demapper that translates from constellation symbols to bits.

A part from the aforementioned kernels, our receiver application also contains the non-kernel code (glue code). Steering only the kernel code can lead to architecture optimizations that blow up the glue code execution time. This can result in unexpected overhead and therefore expensive iterations in late stages of the design flow. Hence we stress the importance of considering the whole application to drive, even at this high level, architectural explorations.

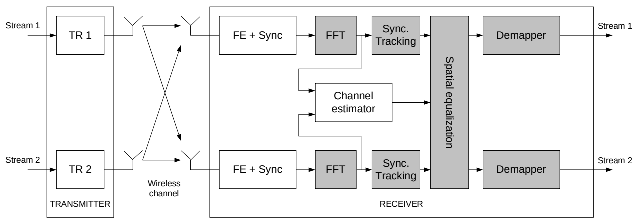

Figure 2. SDM WLAN OFDM receiver.

4. EXPLORATION METHODOLOGY

With the increasing complexity of wireless platforms, the main design challenges are related to design methodology [13]. Consequently, new approaches are crucial to enable fast but still meaningful explorations.

In this section we introduce the methodology applied in our extensive energy-performance aware architecture exploration. We first give a global overview of the methodology flow, providing later further details on the energy and delay estimation steps.

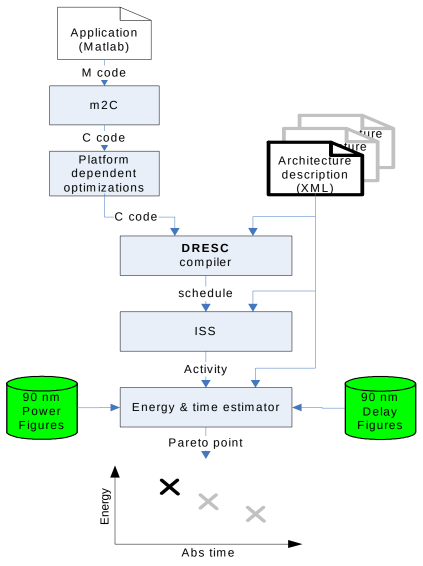

Figure 3. Exploration methodology flow.

4.1. Methodology flow

The starting point of our methodology flow (see Fig. 3) is a Matlab [14] description of the application. Then, the targeted functionality is automatically translated to C by means of the Real-Time Workshop toolbox from MathWorks [14]. To maximize the mapping efficiency of the generated code, it is advantageous to manually apply a number of source-code transformations [10]. For example, among others, nested loops should be replaced with a single loop (loop coalescing), array elements operated inside the loop should be dumped into integers, loops should have one exit and function calls within loops must be in-lined. The optimized C code is then compiled for the different architectures instances using the DRESC compiler. The resulting schedule is simulated with the Instruction Set Simulator (ISS) to both check the correctness and generate activity traces. This activity information, together with delay and energy information of the current architecture instance are the inputs to generate a point in the exploration space. Varying the architecture parameters, one can construct an image of the design space where only the Pareto optimal points need to be preserved.

4.2. Energy estimates

One possible power estimation approach is the so-called Physical-Level Power Analysis (PLPA) methodology. The latter, based on the analysis of the switching activity of all circuit nodes of the architecture, gives accurate estimates. However, PLPA requires a detailed description of the processor implementation at transistor level, which normally is only available late in the design cycle, being therefore unsuitable for early fast explorations. In order to estimate the power at a reasonable computation effort, we advocate for a Functional-Level approach [13] where an abstraction of the CGA processor core, depicted in Fig. 4, is considered**.** The latter splits the architecture in the following functional elements:

- *IF*: Fetching of VLIW instruction words and dispatching of atomic instructions. We assume NOP compression.
- *DM*: Multi-banked scratchpad memory. We assume one memory per Load/Store unit.
- *FR*: First row of reconfigurable cells. This includes all the Functional Units (FUs) connected to the Global Register File (GRF), their Context Memory (CM) fetching, and the GRF.
- *OR*: All the rows but the first row. This includes all the FUs connected to the Local Register Files (LRF), their CM fetching, and the LRF.
- *IC*: Interconnect network

We have synthesized the components of the FU and RFs with Synopsys Physical Compiler targeting state-of-the-art 90nm technology. We considered worst-case design corner for synthesis and nominal corner (V_{DD}=1V) for power estimation. The energy consumptions of the memories (DM, CM and IF blocks) have been obtained from appropriate memory datasheet. Finally, to characterize the IC contribution, the energy consumption per active connection has been extracted from the place & routing of a representative instance. The latter is then extrapolated for other architecture instances.

To estimate the energy required to execute a given application, the energy estimator script weights the aforementioned figures with the activity information extracted by the instruction set simulator (ISS)**.**

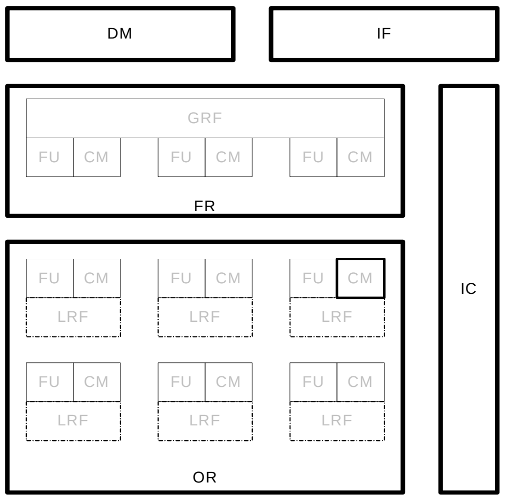

Figure 4. Architecture partitioning for power analysis.

4.2. Execution time estimate

The ISS reports the exact number of cycles required to execute a given application. This metric would be enough to compare the performance of different applications running on the same processor instance. However, in our exploration we also vary architecture parameters that affect the critical path and consequently the clocking frequency of the architecture instance. Thus, in order to correctly compare different architectures, we need to transform the reported cycles to absolute execution time. To do so, we have identified the critical path of our architecture template in the VLIW units, more concretely in the necessary operand forwarding network. This specific part of the architecture is therefore further explored with the RTL synthesis flow presented in [16]. RTL implementation of the VLIW part out of our XML architecture description is generated. After synthesis with Synopsys, the maximum clocking frequency for every instance is obtained. The timing and power consumption of several architectures obeying at different combinations of number of VLIW issues and word widths is generated. With the latter information a library is instantiated.

5. EXPLORATION AND ANALYSIS

This section reports and analyzes the results obtained with the proposed CGA energy-performance aware architecture exploration. We first justify the architectural parameters considered in this initial high level exploration. In the following subsections, we describe concrete affects related to ILP and DLP variations on the architecture instance. Later, we analyze the latency-performance trade off resulted from the software pipelined approach. Finally, we discuss the results on the Time-Energy Pareto space.

5.1. Exploration strategy

ADRES template provides a large number of parameters that can be varied. We advocate for a gradual architecture exploration, starting with the dimensioning of the parameters that have bigger impact in performance and energy consumption of the architecture. For the actual experiment we only vary the size of the array and the width of the SIMD datapath. Varying only these parameters we show variations of up to 65% in performance and up to 85% in energy. We do not consider other parameters like interconnect topology in this paper. Our previous work [17] shows that different interconnect topologies can influence up to 35% of the performance and 30% of the energy consumption. The application of such partitioning considerably reduces the search space and thus the design time.

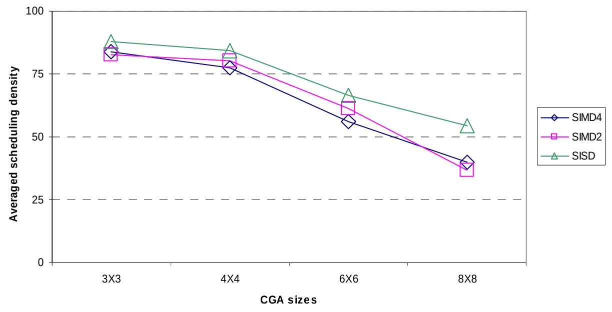

Figure 5. The efficiency of the compiler is reduced while increasing the size of the CGA .

5.2. ILP exploration

In a CGA, the only architectural way to modify the achievable ILP is by changing the size of the array. The DRESC compiler can extract the ILP from the code in the scope of a loop and map it on the architecture using modulo scheduling. Fig 5 shows the evolution of the averaged scheduling density (over the considered kernels) while increasing the size of the CGA. We observed that from the 4x4 array onwards, the scheduling density starts to decay drastically. The reason is that by increasing the size, the scheduling problem, solved by heuristics, becomes more complex and the risk of finding a suboptimal schedule increases. Moreover, for a given piece of code, the amount of instructions that can be scheduled in parallel is limited, as dependencies break the amount of ILP present. In Fig. 5 we also observe that for the SISD (Single Instruction, Single Data) case, the decay of the scheduling density is weaker than for the other curves. This is because the compiler converts the DLP present in the algorithm to ILP. As soon as the DLP is exploited (SIMD2 or SIMD4), the number of independent operations that can be scheduled in parallel is further reduced and hence the scheduling density experiences a more pronounced decay.

5.3. DLP exploration

The DLP extraction in the ADRES framework fully relies on the programmer, who models the vector operations using intrinsic functions in the application C code.

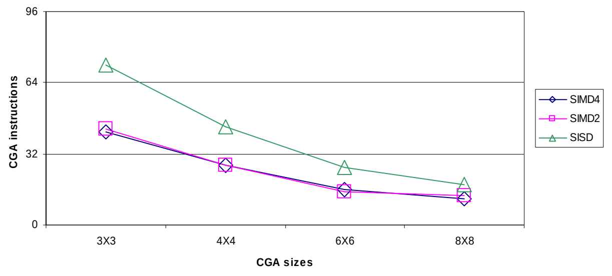

Figure 6. Complex ISA reduces the instruction memory footprint.

We select 16 bits as subword length since it provides enough precision to accommodate the targeted SDR processing. For the DLP exploration, we instantiate 3 different degrees of data parallelism:

- SISD: single instruction operates on single subword. This results in a 16 bit architecture.
- SIMD2: single instruction operates on 2 subwords, real and imaginary part of a complex word. This results in a 32 bit architecture.
- SIMD4: single instruction operates on 4 subwords, real and imaginary part of 2 complex words. This results in a 64 bit architecture.

Most of the baseband processing operates on complex data (I and Q data). Thus we upgrade the Instruction Set Architecture (ISA) of the SIMD2 and SIMD4 architectures to support complex arithmetic [16].
This simplifies the application dataflow reducing the complexity of the scheduling problem, as there are fewer instructions to be scheduled (see Fig 6).

5.4 Performance-latency trade off

The software pipelining approach that increases the application ILP comes with a known drawback: the processing efficiency depends on the number of iterations of a loop. In the software pipeline, like in hardware pipeline, we need to fill in the pipeline (prologue stage) before fully utilizing all the allocated resources (kernel stage). Then, once we provide the last input, we still have to wait until the pipeline is emptied (epilogue stage) to obtain the result. More the iterations to execute, lower is the overhead due to the epilogue and prologue in the overall cycle count. To increase the number of loops iterations, we consider the processing of several OFDM symbols in parallel. Fig. 7 shows the effect, in symbol time and latency, of parallelizing the execution of multiple symbols. We observe that going from 4 to 8 processed symbols in parallel, the symbol execution time is slightly reduced while the latency is almost doubled. This indicates that from 4 parallel symbols on, the weight of the prologue and epilogue on the overall cycle count is negligible. Hence we consider the processing of 4 symbols in parallel as the optimal configuration for execution time. However, processing multiple symbols together increases the latency and also the data memory footprint requirement. Thus, once at design time the memory size is fixed to accommodate several symbols (4 symbols in our concrete example), the still existing trade off between latency and symbol time can be exploited at run-time. For this reason we have written our code on such a way that the number of symbols to process is an input parameter of our software which controls the loop iterations. This enables the adaptability of the processing efficiency to the current latency requirement.

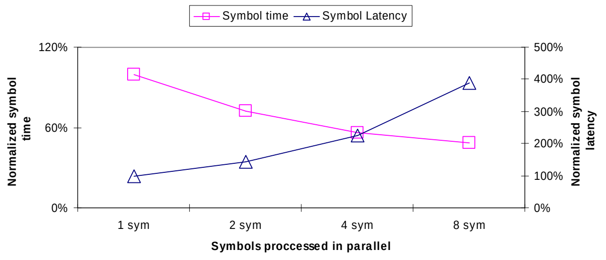

Figure 7. Performance-latency trade off.

5.5. Energy vs. execution time

In order to choose the most suitable architecture, the selection of appropriate figures of merit is crucial. We consider the average energy and execution time of a single OFDM symbol. Area has not been considered since it is supposed to be largely dominated by memories, which are not affected by our exploration.
Fig. 8 shows the Time-Energy Pareto curves obtained in this architecture exploration. The performance axis (X axis) indicates the absolute time required for the processing of one OFDM symbol. The latter should be, in average, below 4 µs to guarantee real-time demodulation. The Y axis corresponds to the energy consumed in the processing of a single OFDM symbol. In the space defined by these two axes we have plotted the possible design points of the studied CGA instances. We have two representations for every architecture instance, one considering the processing of a single OFDM symbol (solid line) and a second one considering the processing of 4 symbols in parallel (dashed line). At run-time, when the conditions allow long latency, the processor can drastically improve its energy efficiency by moving from the solid working point to the dashed one. We represent with an identical marker the points that belong to the same array size and vary only the width of the datapath (from left to right: SIMD4, SIMD2 and SISD).
Larger arrays produce longer prologues/epilogues as well as the overhead in terms of both performance and energy due to these prologues and epilogues. Therefore they experience a remarkable improvement when changing the processing mode from single symbol to 4 symbols in parallel. We also observe, especially in small arrays, a clear benefit of widening the SIMD datapath. The latter importantly improves the performance while the energy consumption is maintained, thereby leading to architectures with higher energy efficiency.
From a certain array size on, no performance improvement is obtained due to the reduction in scheduling density and achievable clock frequency. Moreover, this increase in size leads to an energy penalty. Hence architectures such as 6x6 and 8x8 arrays are not interesting instances of the ADRES template to efficiently run the considered application driver.

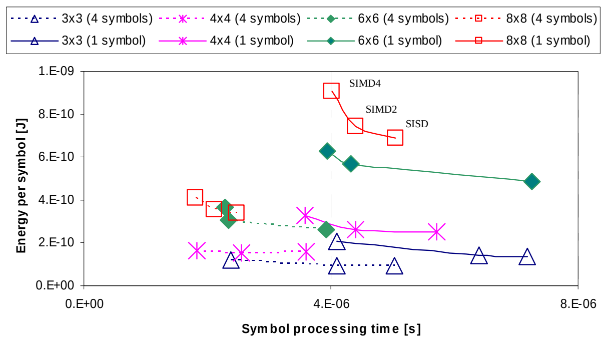

Figure 8. Explored architecutes represented in the "Time-Energy" Pareto space.

The 3x3 and 4x4 SIMD4 instances are Pareto-optimum points offering different energy-performance trade offs. The 3x3 SIMD4 instance consumes less energy than the 4x4 SIMD but also delivers less performance. We select the 4x4 SIMD4 instance as it offers real-time processing when demodulating a single OFDM symbol (hard latency constraints). The latter executes the symbol-based base-band processing of a 108 Mbps SDM OFDM WLAN receiver with an estimated average power consumption of 110 mW running at 330 MHz in 90nm CMOS technology.

6. CONCLUSIONS

In the race towards SDR terminals, energy efficiency is the figure of merit. The latter can be largely affected by increasing the parallelism on processor architectures. In this paper, we demonstrated that finding right combination of the different sources of architectural parallelism is fundamental. This combination is application/domain dependent and consequently we stress the importance of selecting complete and representative application drivers.
CGAs are strong candidates to deliver high energy efficiency while providing the required performance. Nevertheless, most CGAs like ADRES are templated, being able to generate an infinite amount of different instances, though just few of them can be considered optimal. Consequently fast methodologies, like the proposed in this paper, are crucial to drive designers' choice in this large design space.

7. ACKNOLEDGEMENTS

The authors want to thank their colleagues in the SDR FLAI and ADRES projects for technical support.
This research has been carried out in the context of IMEC's multimode multimedia program which is partly sponsored by Samsung.

8. REFERENCES

[1] Glossner, J.; Moudgill, M.; lancu, D., "The sandbridge SDR communications platform," *SympoTIC '04*, pp. ii–ix, 24-26 Oct. 2004.
[2] J. Rabaey, "Silicon Platforms for the next generation wireless systems - What role does reconfigurable hardware play?," *FPL 2000*, pp. 277-285, Aug. 2000
[3] SiliconHive, Philips Research, http://www.siliconhive.com
[4] K Van Berkel, F. Heindle, P. Meuwissen, K. Moeren and M. Weiss, "Vector Processing as an Enabler for Software-Defined Radio in Handsets from 3G+WLAN Onwards," *SDR Technical Conference*, pp. 125-130, Nov. 2004.
[5] Texas Instruments "TMS320C6000 CPU and Instruction Set", http://www.ti.com.
[6] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, K. Flautner, "SODA: A Low Power Architecture For Software Radio," *ISCA'06*, pp. 89-101, June 2006.
[7] Novo, D.; Moffat, W.; Derudder, V.; Bougard, B., "Mapping a multiple antenna SDM-OFDM receiver on the ADRES coarse-grained reconfigurable processor," SIPS*'06*, pp. 473-478, Nov. 2006.
[8] B. Mei, S. Vernalde, D. Verkest, H. De Man and R. Lauwereins, "ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse Grained Reconfigurable Matrix," *FPL* 2003, pp. 61-70, Sept. 2003
[9] B. Mei, S. Vernalde, D. Verkest, H. De Man and R. Lauwereins, "DRESC: A Retargetable Compiler for Coarse-Grained Reconfigurable Architectures," *FPL'02*, pp. 166-174, Sept. 2002.
[10] R. B. Rau, "Iterative Modulo Scheduling," *HP Lab*, Tech Report: HPL-94-115, 1995.
[11] Van der Perre, L., "Broadband WLANs: setting the limits for SDR platforms", *WWRF15 Meeting*, 2005.
[12] MathWorks, http://www.mathworks.com/
[13] Aarno Parssinen, "Keynote talk: System Design for Multi-Standard Radios," *ISSCC'06*, February, 2006.
[14] IEEE 802.11, http://grouper.ieee.org/groups/802/11/
[15] Laurent, J.; Julien, N.; Senn, E.; Martin, E., "Functional level power analysis: an efficient approach for modeling the power consumption of complex processors", *DATE'04*, Vol. 1, pp. 666-667, Feb. 2004.
[16] T. Schuster et al., "Subword-Parallel VLIW Architecture Exploration for Multimode Software Defined Radio", *SIPS'06*. Oct. 2006.
[17] A. Lambrechts, P. Raghavan, M. Jayapala, F. Catthoor and D. Verkest, "Energy-Aware Interconnect Exploration of Coarse-Grained Reconfigurable Processors," *WASP'05*, September, 2005.