AN EXPLORATION METHODOLOGY FOR VLIW ARCHITECTURE TARGETING MULTIMODE WIRELESS BASEBAND PROCESSING

[1]*Thomas Schuster*, [12]*Bruno Bougard*, [1]*David Novo Bruna*, [3]*Emil Matus*,
[1]*Liesbet Van der Perre*, [3]*Gerhard Fettweis*

Thomas.Schuster@imec.be
[1]IMEC, Kapeldreef 75, B-3001 Leuven, Belgium
[2]ESAT, K. U. Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
[3]IfN, TU-Dresden, Helmholtzstrasse 18, D-01062 Dresden, Germany

ABSTRACT

The wide range of future and existing standards in wireless communications calls for flexible solutions. A way to introduce flexibility to digital radio systems is to replace hardwired components by programmable devices.
This paper describes the exploration of a VLIW architecture targeting the domain of multimode wireless baseband processing. Target is to explore the tradeoff between flexibility, performance and power consumption.
The Coware® LISATek™ tool suite for automated embedded processor design has been used to create a processor model with scalable number of functional units and RTL generation capability. Furthermore, a C-compilation framework based on IMPACT was adapted to enable the mapping of a Fast Fourier Transformation and parts of a SDM-OFDM receiver. The code was optimized with special instruction supported in C by intrinsic functions. To estimate the costs of the implemented optimization, the generated RTL has been synthesized in 90nm technology.
It is shown that a wide energy-performance tradeoff can be ranged by varying the processor configurations. Moreover, by using special instructions for complex arithmetic, power consumption and execution time of the benchmarked kernels can be reduced by factor two.

1. INTRODUCTION

Over the last decade, digital radio systems have gained a large acceptance, opening up entirely new markets and pushing the development of new products and services. This is especially true in mobile personal communications where standards and demands are evolving quickly. Short time to market is therefore crucial, pushing for more flexible implementation. Software Defined Radio (SDR), where most of the modulation and demodulation processing is carried out on programmable and/or reconfigurable hardware, has been introduced as the ultimate way to achieve flexibility and thus short time-to-market for radio products [1]. Although SDR are getting wide-spread in communication infrastructure, deploying them in battery-powered handheld devices makes energy-efficiency a vital feature. Traditionally, programmable devices such as general purpose processors and Digital Signal Processors (DSP) have an energy-efficiency one or two order(s) of magnitude lower than dedicated ASIC solution. Commonly, Application Specific Instruction-set Processors (ASIP) [2] are considered most suitable for a good energy-performance tradeoff. They offer the flexibility of a programmable solution, while remaining energy efficient through a particularly targeted instruction set. However, SDR processors have potentially to support multiple standards, so that they cannot be as far application optimized as traditional ASIPs are. An appropriate SDR processor is domain specific, with a domain restricted to baseband processing.
In this paper, we focus on SDR domain-specific VLIW architectures. To enable extensive architecture exploration with representative SDR benchmarks, we have set up a methodology for instruction set exploration based on a generic VLIW architecture. The LISATek suite for automated embedded processor design and automation is used to create the cycle accurate model of a homogeneous VLIW processor with a scalable number of functional units (FUs). To enable the mapping and profiling of SDR benchmarks, we use an IMPACT based C compilation framework. As benchmark, we consider a Fast Fourier Transformation (FFT) and parts of a SDM-OFDM receiver. By introducing special instructions for SIMD and complex arithmetic, we target our instruction set towards the SDR domain. To estimate the implementation costs of the proposed optimizations and achieve a suitable tradeoff between flexibility, performance and power consumption, RTL-code generated from LISATek is synthesized using Synopsys® tools and state-of-the-art 90nm technology. Next to a physical compilation including an automated placement analysis, power estimation is carried out. Therefore, the synthesized designs are back annotated with realistic switching information, provided by simulations of the previously mapped applications. Based on that, the energy-performance tradeoff can be analyzed.

2. TARGET ARCHITECTURE

One of many approaches to exploit parallelism in signal processing is VLIW (Very Long Instruction Word) [3]. VLIW processors are especially successful on the embedded processor market. They are specialized on the exploitation of ILP and in opposite to other DSP architectures, dispose of mature C compiler support. This is an important advantage for the development of complex multi-mode systems and has influenced our decision to consider VLIW as baseline architecture for SDR.

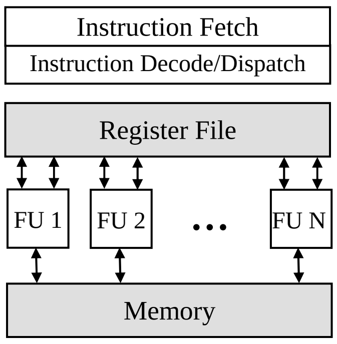
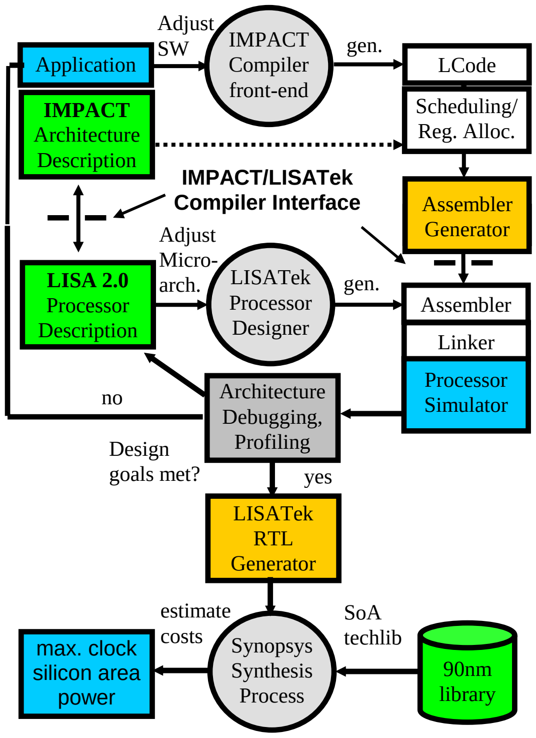


Fig. 1: VLIW architecture with N homogeneous FUs

Our target architecture (fig. 1) mainly consists of a scalable number of FUs and a shared multiport register file. All FUs have a homogeneous instruction set and are directly connected to a common memory hierarchy. The processor has a load/store architecture. The data paths are 32-bit wide and pipelined in 6 stages. The register file contains 64 32-bit general purpose registers and 32 1-bit predicate registers. The predicate registers are used for conditional instruction execution in EPIC style [4]. The flexible number of FUs in the design and their homogeneous character allows the exploration of different levels of ILP. The 32-bit data paths can be split in order to support 2x 16-bit SIMD instructions and benefit from potential DLP.

3. METHODOLOGY FOR INSTRUCTION SET EXPLORATION

Based on the proposed architecture template, we carry out an instruction set exploration targeting the SDR domain. Therefore, we widely follow the design flow proposed in [5]. The LISATek tool-suite for automated embedded processor design and automation has been used to create a cycle accurate instruction-set simulator. A typical design in LISATek starts from a high-level processor description in LISA Language for Instruction Set Architecture and is then iteratively refined into a RTL implementation model. The successful compilation of a LISA processor model enables the generation of software development tools. LISATek provides a retargetable assembler, linker, disassembler, processor simulator and a C compiler [6]. Because the C compiler shipped with LISATek is not usable for our kind of target architecture, it has been replaced by an IMPACT based compiler framework developed at IMEC [7] [8] [9].



Application
Adjust SW
IMPACT Compiler front-end
gen.
LCode
IMPACT Architecture Description
Scheduling/ Reg. Alloc.
IMPACT/LISATek Compiler Interface
Assembler Generator
Adjust Micro-arch.
LISA 2.0 Processor Description
LISATek Processor Designer
gen.
Assembler
Linker
Processor Simulator
no
Architecture Debugging, Profiling
Design goals met?
yes
LISATek RTL Generator
estimate costs
SoA techlib
max. clock silicon area power
Synopsys Synthesis Process
90nm library

Fig. 2: LISATek based exploration flow

By following the design flow presented in figure 2 an instruction set exploration can be carried out. New instructions simply need to be encoded in the LISA processor description and introduced to the compiler as intrinsic functions. Afterwards, they can be directly used to optimize C programmed applications. LISATek provides the profiling tools to determine the effects of the implemented optimizations on the program execution.
However, fundamental architectural information such as silicon area, power consumption and maximum clock frequency can only be collected by means of simulations on RTL and Gate level. We use the LISATek RTL generator to produce synthesizable RTL and a mainly Synopsys® based tool-chain (Physical Compiler™, PrimePower™) to perform gate-level synthesis and RTL power simulation. In order to achieve realistic power estimates, the designs are back-annotated with switching activity from compiled SDR applications. That way, we observe the costs of architectural features such as instruction set extensions and have an immediate feedback for design decisions.

4. INSTRUCTION SET EXTENSIONS

Algorithms for wireless baseband processing usually operate on complex input samples. Hence, we decided to focus on instruction set extensions for complex arithmetic. Due to the fact that a precision of 16 binary digits is sufficient for almost all tasks in wireless baseband processing, the small set of special instructions presented in table 1 promises high speed up and high data path utilization.

op-code	**intrinsic function**
brev	bit reversed index calc.
cadd	2x16bit vector addition
csub	2x16bit vector subtraction
cmul	complex multiplication
casr	complex arith. shift right
ccon	conjugate complex

Tab. 1: special instructions for optimization

5. SOFTWARE MAPPING AND PROFILING

Two software mapping experiments are carried out. First, we benchmark a 64-point Radix-4 Fast Fourier Transform (FFT) with decimation in time (DIT). The FFT is one of the most important operations in digital signal processing. It is needed in all OFDM-based transmission schemes and therefore important for SDR. As a second more sophisticated benchmark, we use the C code of a Space Division Multiplex (SDM) OFDM receiver. Therefore, we consider 5 of the kernels introduced in [10]. Table 2 gives an overview about these kernels and shows their dominant operations.

Kernel A	Channel Estimation Stage 1 (IDFT) complex matrix multiplication
Kernel B	Channel Estimation Stage 2 (DFT) complex matrix multiplication
Kernel C	Noise Estimation complex square norms
Kernel D	MMSE algorithm complex matrix inversion
Kernel E	One tap compensation (8x) complex vector multiplication

Tab. 2: Kernels for SDM-OFDM experiment

Both test benches, FFT and SDM-OFDM receiver, are mapped to VLIW processors with different numbers of FUs. Moreover, all kernels have been optimized using the small set of special instructions introduced in section 4.

Table 3 shows the simulation cycle counts for the 64-point FFT. The benchmarks for the SDM kernels are presented in table 4 and 5.

FFT64	1 FU	2 FU	4 FU	6 FU	8 FU
standard	4368	2259	1195	914	886
optimized	1725	905	494	388	388

Tab. 3: Cycle counts 64 Point Radix-4 FFT

SDM 4FU	A	B	C	D	E
standard	8480	11154	663	2057	8168
optimized	1840	3083	404	914	3402

Tab. 4: Cycle counts SDM-OFDM, VLIW 4FUs

SDM 8FU	A	B	C	D	E
standard	5957	7848	473	2153	6032
optimized	1453	2856	336	991	2704

Tab. 5: Cycle counts SDM-OFDM, VLIW 8FUs

Because all our test benches operate on complex data, we naturally win by introducing special instructions for complex arithmetic. For almost all configurations speed ups (in terms of cycle counts) of more than factor two are achieved. It can also be observed that for up to 4 FUs the cycle counts decrease proportionally to the number of FUs in the design.

6. IMPLEMENTATION COST ESTIMATION

To achieve a realistic cost estimate for the different processor configurations, we generate RTL from LISATek and use a Synopsys synthesis flow based on Physical Compiler, PrimePower and a 90 nm technology library. Physical Compiler is a block-level timing convergence tool that combines proprietary synthesis, placement and static timing analysis. It is used to estimate the maximum clock rate of our designs. PrimePower is intended for a full-chip, gate-level power analysis. We use it for an activity-based RTL power simulation. Realistic switching activity is provided by the FFT test bench introduced in section 5.
Figure 3 shows the maximum clock rates of the design. It can be seen that our intrinsics cause an average frequency drop of 70 MHz and only a small area overhead (Tab. 7). The average power consumption (Tab. 6) depends on the instruction set complexity and the processor utilization. For a fair assessment we need to look at execution time and energy consumption of a full benchmark. Figure 4 presents these data for a 64 Point FFT and different processor configurations. It can be seen that within our exploration space a processor with 4 FUs delivers the best energy-performance tradeoff. It also becomes visible that by introducing our special instruc-

tion the tradeoff curve is shifted to the origin. We gain a factor 2 in performance and energy efficiency.

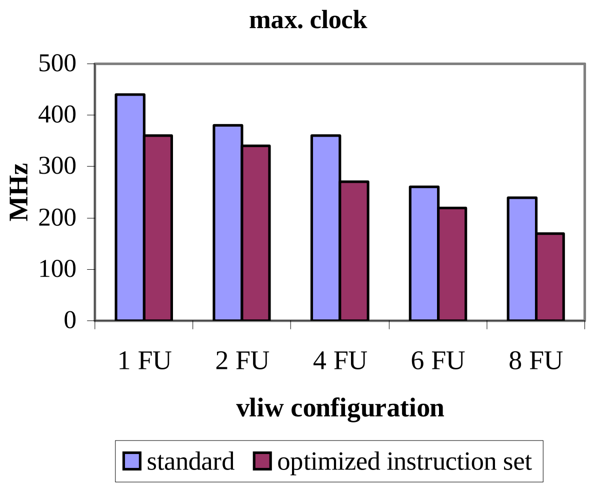


Fig. 3: maximum clock frequency

Power [mW]	1 FU	2 FU	4 FU	6 FU	8 FU
standard	27,7	48,75	83,64	92,12	126
optimized	31,92	55,03	81,71	86,7	99,54

Tab. 6: average power for 64 Point FFT

Area [mm^2]	1 FU	2 FU	4 FU	6 FU	8 FU
standard	0,20	0,37	0,71	0,97	1,41
optimized	0,21	0,43	0,77	1,10	1,53

Tab. 7: area after preliminary placement

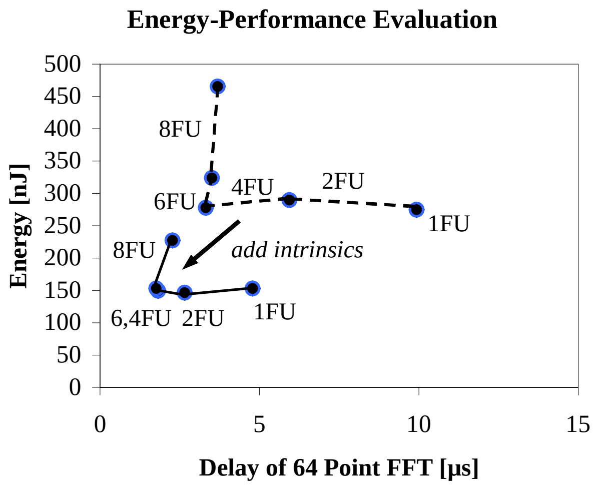


Fig. 4: energy-performance optimum

7. CONCLUSION

We have set up a LISATek based exploration methodology for VLIW DSP architecture. By scaling the number of FUs and introducing a set of special instructions, we have optimized a processor instance for the domain of SDR. Various kernels such as a FFT and parts of a SDM-OFDM receiver have been mapped and profiled.
To assess the costs of the implemented optimizations, we have generated RTL from LISATek. Gate-level synthesis and RTL power simulations have been carried out.
Our results show that within the exploration space a processor configuration with 4 FUs and special instructions for complex arithmetic leads to a Pareto optimum tradeoff between energy and performance. A FFT over 64 complex samples can be computed in less then 2 µs and with energy below 200 nJ. In a configuration with 4 FUs and with a specialized instruction set, our VLIW processor delivers sufficient performance for SISO WLAN IEEE 802.11 a/g.

ACKNOWLEDGEMENT

This research has been carried out in the context of IMEC's multimode multimedia program which is partly sponsored by Samsung and Freescale.

REFERENCES

[1] H. Capelle, B. Bougard, "Low power SDR baseband implementation for digital broadcasting receivers," IST Mobile & Wireless Communication Summit, 2005

[2] H. Meyr, "System-on-chip for communications: the dawn of ASIPs and the dusk of ASICs," SIPS 2003

[3] S. Agarwala, "A 600-MHz VLIW DSP," IEEE Journal of Solid-State Circuits, Volume 37, Issue 11, 2002

[4] D. I. August, D. A. Conners, "Integrated Predicated and Speculative Execution in the IMPACT EPIC Architecture," ISPASS, 2000

[5] A. Hoffmann, H.Meyr, R. Leupers, "Architecture Exploration for Embedded Processors with LISA," Kluwer Academic Publishers, 2002

[6] M. Hosemann, G. Cichon, "Implementing a Receiver for Terrestrial Digital Video Broadcasting in Software on an Application-Specific DSP," SIPS, 2004

[7] B. Mei, S.Vernalde, "DRESC: A retargetable compiler for coarse-grained reconfigurable architectures," International Conference on Field Programmable Techn., 2002

[8] The IMPACT Group, http://crhc.uiuc.edu/impact

[9] Q. B. Olaniran, "Emulation of the intermediate representation in the IMPACT compiler," Master Thesis University of Illinois at Urbana-Champaign, 1997

[10] D. Novo Bruna, "Mapping a multiple antenna SDM-OFDM receiver on the ADRES Coarse-Grained Reconfigurable Processor," SIPS, 2005