A Low-Power ASIP for IEEE 802.15.4a Ultra-Wideband Impulse Radio Baseband Processing

Christian Bachmann*, Andreas Genser*, Jos Hulzink†, Mladen Berekovic‡ and Christian Steger*

*Institute for Technical Informatics, Graz University of Technology
Inffeldgasse 16, 8010 Graz, Austria

†IMEC Netherlands, Holst Centre
High Tech Campus 31, 5656 AE Eindhoven, The Netherlands

‡IDA, TU Braunschweig
Hans-Sommer-Str. 66, 38106 Braunschweig, Germany

***Abstract*—The IEEE 802.15.4a amendment has introduced ultra-wideband impulse radio (UWB IR) as a promising physical layer for energy-efficient, low data rate communications. A critical part of the UWB IR receiver design is the low-power implementation of the digital baseband processing required for synchronization and data decoding. In this paper we present the development of an application-specific instruction-set processor (ASIP) that is tailored to the requirements defined by the baseband algorithms. We report a number of optimizations applied to the algorithms as well as to the hardware architecture. This enables performance increases up to a factor of 122x and energy consumption decreases up to 90x as compared to a 16-bit baseline architecture. Furthermore, this ASIP offers greater flexibility due to programmability as compared to an ASIC implementation.**

I. Introduction

Within IMEC's Human++ research program [1], ultra-low power radio transceivers are identified as the key components in future wireless sensor networks for mobile healthcare applications. IEEE 802.15.4a ultra-wideband impulse radio (UWB IR) represents a potential communication scheme for these low-power radios.

UWB IR transceivers typically make use of an analog/digital partitioning: An analog radio front-end is used to transmit and receive modulated RF signals. The baseband signals are encoded and decoded by a digital baseband processor as depicted in Fig. 1.

In first generation systems an ASIC implementation has been employed for digital baseband processing [2]. This implementation has been extended to an application-specific instruction-set processor (ASIP), offering greater flexibility as well as better energy-efficiency [3]. Both implementations are, however, operating on a predecessor modulation scheme and do not support the latest revision of the standard. This paper presents the design and implementation of a new baseband ASIP supporting the low-power and low-rate modes of the novel IEEE 802.15.4a standard amendment [4]. In addition to the previous ASIP, basic channel estimation and a rake receiver structure are being considered.

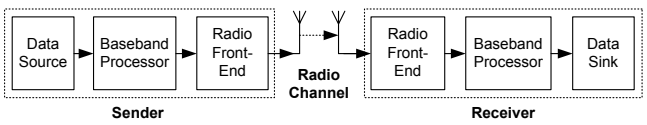

Fig. 1. Baseband processing principle

Section II gives a brief introduction to IEEE 802.15.4a ultra-wideband impulse radio and baseband algorithms. The design process of the baseband ASIP is described in Section III. Furthermore, the design methodology is outlined. In Section IV various optimizations are presented. Finally, Section V illustrates implementation results and in Section VI conclusions are drawn.

II. IEEE 802.15.4a Ultra-Wideband Impulse Radio

Ultra-wideband impulse radio communication is based on the emission of very short pulse waveforms, exhibiting wideband spectral characteristics in the frequency domain. Due to the stringent UWB emission limits (e.g., [5]), the energy contained in a single pulse is very low. In order to be still able to extract information out of UWB transmitted data and to fight noise and interference, a technique called *spreading* is used. A symbol to be transmitted is represented by a number of consecutive pulses instead of a single pulse. The pseudo-random bit sequence determining the polarity of these pulses is referred to as *spreading code*.

In order to restore the original bit, i.e. to *despread* the sequence of pulses, a cross-correlation operation

$$y = \sum_{i=1}^{N} (x[i] \cdot C[i]) \tag{1}$$

of N pulses x with the spreading code C, also of length N, is calculated. The value of the decision variable y determines the value of the original bit.

978-3-9810801-5-5/DATE09 © 2009 EDAA

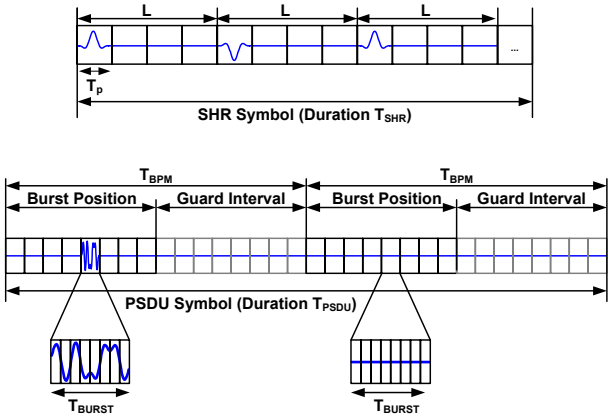

Fig. 2. SHR (top) and PSDU (bottom) symbols

A. IEEE 802.15.4a Frame Structure

In IEEE 802.15.4a UWB IR systems data is being transmitted in frames consisting of three major sub-parts:

- Synchronization header (SHR) / Preamble
- Data header (PHR)
- Data unit, i.e. the actual payload data (PSDU)

The synchronization header, also referred to as *preamble*, is being transmitted in order to aid receiver algorithms in timing acquisition and frame synchronization. It also serves the purpose of channel estimation and gain control setting optimization. This preamble is composed of *SHR symbols* that contain a number of isolated pulses as depicted in Fig. 2.

The data unit and its header are constructed out of *PSDU symbols*. Each symbol is able to carry two bits of information: One bit is coded in the symbol half in which a burst, i.e. a concatenation of pulses, occurs (burst position modulation, BPM). Another bit is used to determine the polarity (phase) of the burst itself (binary phase shift keying, BPSK). The state of a linear feedback shift register (LFSR) varies the spreading code for each transmitted PSDU symbol. This spreading code determines the burst sequence as well as the exact burst position within one of the symbol halves ("hopping code"). A PSDU symbol is depicted in Fig. 2.

B. Baseband Algorithms

When designing the UWB IR baseband algorithms, the frame structure as pointed out in Section II-A has to be taken into account. The different sub-parts of a frame are processed by different sub-algorithms.

1) Synchronization / Timing Acquisition: The synchronization phase is commonly composed of noise estimation, signal detection (coarse acquisition), fine acquisition, channel estimation and end-of-preamble search sub-states [7]:

- Initialization & Noise Estimation
  All necessary data structures are initialized and a threshold value corresponding to current noise levels is computed.
- Signal Detection (SD)
  The purpose of the signal detection state is to detect the presence of an ultra-wideband impulse radio signal within input data that is corrupted by noise.
- Fine Acquisition (FA) & Channel Estimation
  The fine acquisition algorithm is used to optimize synchronization and to detect false positives during signal detection. Furthermore, channel estimation is integrated into this algorithm.
- End-Of-Preamble (EOP) Search
  The EOP search algorithm detects the end of the preamble (synchronization header) and the start of the actual payload data within the UWB IR frame.

2) Payload Decoding (PD) / PSDU Data Despreading: Once all synchronization algorithms have been successfully passed, the decoding of the actual payload data can be carried out. A selective-rake receiver structure is employed in order to achieve higher signal-to-noise ratios. The channel coefficients previously derived by the channel estimation algorithm are used as weights within the rake receiver structure.

III. Design of a Baseband ASIP

The methodology used for designing this ASIP employs a gradual exploration process covering multiple processor architectures. The use of the following architectures for baseband processing is investigated:

- Scalar RISC
- Vector RISC
- Vector Very Long Instruction Word (VLIW)

Each of these architectures is adapted to optimize the execution of the baseband algorithms.

A. ASIP Design Flow

In the design process the IP Designer [6] tool suite by Target Compiler Technologies as depicted in Fig. 3 was employed.

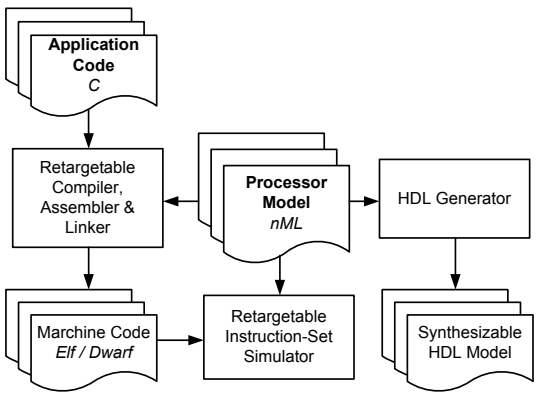

Fig. 3. Target tools environment [6] used for baseband ASIP design

It allows for fast architectural exploration due to *retargetability*. The high-level description of the processor facilitates rapid architectural changes.

The highly optimizing C-compiler, the (dis)assembler and the linker, are used to generate machine code for a given processor architecture. This code is then executed and benchmarked on the cycle-accurate instruction-set simulator. If performance requirements are met, a HDL model can be generated out of the same high-level description.

B. Architectural Exploration

We use a scalar 16-bit Harvard RISC processor as depicted in Fig. 4 as the starting point in the architectural design process. It comprises a 16-bit arithmetic logical unit (ALU), a 16 $\times$ 16-bit register file (RF), load-store unit, branch unit and instruction decoder. Furthermore, a custom extension unit is used to speed up scalar UWB operations.

Due to the fact that the cross-correlation, which is heavily employed by large parts of the baseband algorithms, is an ideal candidate for parallelization, the move to a vector processor seems obvious. A single instruction multiple data (SIMD) architecture is hence used to investigate the impact of data parallelism on execution performance. The vector architecture as depicted in Fig. 4 is derived from the scalar architecture but additionally includes a vector ALU, a vector register file and vector data memory. In addition to standard vector operations, dedicated UWB application-specific instructions are enabled through the use of a custom extension unit and a spreading code register.

In order to support real-time processing of IEEE 802.15.4a UWB IR baseband data, the algorithmic functionality specified above has to be executed on the baseband architecture obeying several timing limitations. The minimum time spans of SHR and PSDU symbols as well as the amount of data sampled per symbol have to be taken into account. Maximum input data rates of 1250.0 Mbit/s for SHR symbols[1] and 624.4 Mbit/s for PSDU symbols[1] have to be sustained by the architecture. Both, scalar and vector architecture, fail to meet these data rates for moderate clocking frequencies (<250 MHz).

[1]These numbers are derived from timing requirements set forth in the standard [4], an assumed ADC resolution of 5 bit as well as algorithm-dependent settings.

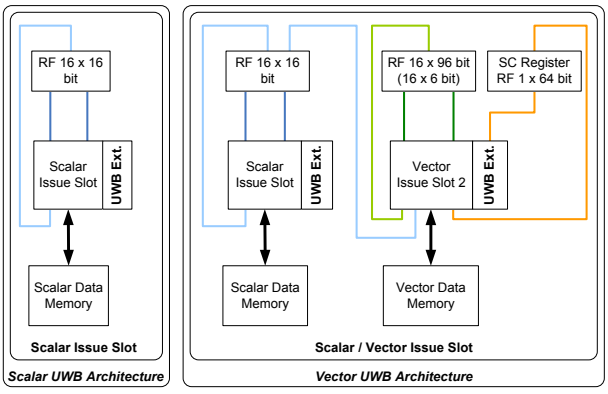

Fig. 4. Architectural overview of scalar (left) and vector baseband processors (right)

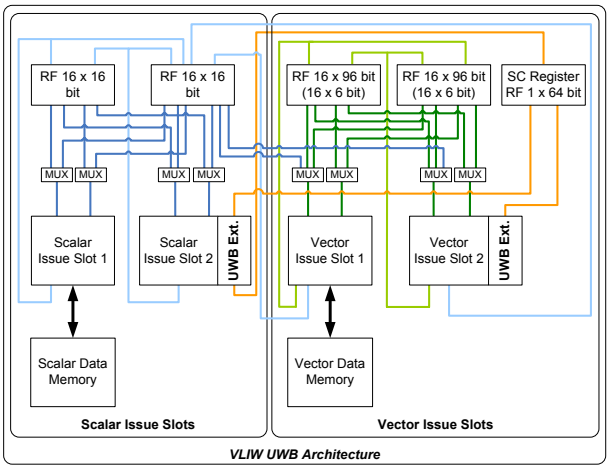

Fig. 5. Architectural overview of final VLIW baseband processor

To achieve sufficient throughput, our proposed UWB baseband architecture resembles a four issue-slot VLIW architecture, consisting of two slots containing scalar and control functional units and two slots containing vector units. Functional units specially tailored to the requirements of baseband processing are present in the design and can be accessed through custom instructions. Register-file sizes are adapted to the resolution of the ADCs used in the radio front-end as well as to algorithm requirements. Similar to the vector processor described above, a dedicated spreading code register is present. Fig. 5 presents an architectural overview of the final baseband processor.

IV. ASIP Optimizations

A. Vector Correlation

A vital part of UWB baseband processing is the cross-correlation of input data and spreading code as expressed in Equation 1. The multiplications inside the sum can be calculated independently from each other, hence enabling their parallel computation. Due to the binary or ternary structure of the spreading code the multiplication itself can be implemented as invert/replace operation.

Furthermore, the different sub-sums can partly be calculated in parallel, resulting in an adder-tree of depth $log2(n)$ where n is the number of vector elements.

B. Custom Instructions

The instruction-set of the VLIW processor is extended by a number of custom instructions as listed in Table I. Scalar operations are used to load, read out and modify the spreading code register. Furthermore, special arithmetic operations are present in the form of the addition of absolute values as well as the emulation of a linear feedback shift register (LFSR).

Custom vector instructions are used to implement the parallelized computation of the cross-correlation as pointed out

TABLE I
LIST OF IMPLEMENTED CUSTOM INSTRUCTIONS

Instruction	Input par.	Outp. par.
Addition of absolute values	(scl,scl)	(scl)
Spreading code load[a]	(scl)	(-)
Spreading code store[b]	(-)	(scl)
Spreading code rotate[a]	(-)	(-)
Spreading code shift-load[a]	(scl)	(-)
LFSR clocking	(scl)	(scl)
Vector SC correlation (PSDU)	(vec)	(scl)
Vector SC correlation (SHR)	(vec)	(scl)
Vector shift	(vec)	(vec)

[a]Internal spreading code register is modified.
[b]Data is read from internal spreading code register.

in Section IV-A. Different variants for header as well as payload decoding, operating on different representations of the spreading code, are available. An additional vector logic operation is present in the form of a vector shift instruction.

C. Algorithmic Optimizations

The baseband algorithms are adapted to the architectural extensions implemented in the ASIP. The synchronization as well as the payload decoding algorithms harness the cross-correlation custom instructions as well as vectorization. Expensive mathematical operators such as divisions, squares and roots can be replaced by suitable shift and absolute value add instructions. Furthermore, by exploiting the size of the register files, a multi-buffer synchronization approach can be implemented. This speeds-up the signal detection state by correlating more than one input data vector during each synchronization phase.

D. Low-Power Techniques

Techniques for low-power logic design employed within the baseband processor are *clock gating* and *operand isolation*. By applying the clock gating technique [8] the distribution of the clock signal to inactive modules of the processor is avoided. The clock signal is then turned off, reducing the power dissipation due to switching activity.

Operand isolation [9], often also referred to as *guarded evaluation*, is used to avoid the propagation of switching activity to inactive parts of the architecture's datapath. Thus, power dissipation due to unnecessary activity is eliminated in unused modules. Both types of logic optimization are performed automatically by the Target HDL generator [10].

V. EXPERIMENTAL RESULTS

Experimental results for evaluating the impact of architectural optimizations are structured into *performance* and *power consumption metrics*. Performance metrics for different processor architectures have been determined by means of cycle count measurements on the instruction-set simulator provided by the Target tool suite. For that purpose, various versions of the baseband algorithms have been executed on the different baseband processor architectures as introduced in Section III-B.

In order to obtain power consumption characteristics of the final UWB baseband processor, the VHDL model generated by the ASIP tool flow, extended with the user primitives for the custom functional units, is synthesized as well as placed and routed for a TSMC 90nm process including memories. The baseband algorithms are then simulated on the back-annotated netlist of the placed and routed processor including parasitics using Cadence NCSim 5.7. The value change data (VCD) information obtained during these simulations is used to obtain power consumption values with Synopsys PrimeTime PX Z2007.

A. Performance Results

For evaluating the overall performance of an algorithm-architecture combination, the different sub-algorithms have been profiled while processing an entire UWB IR frame. Fig. 6 and Fig. 7 depict the worst-case amount of cycles consumed by the different sub-parts of the baseband algorithm. In Fig. 6 the cycle counts for the execution on the basic scalar architecture are shown. Fig. 7 illustrates the cycle counts for processing the same data executing the optimized algorithms on the optimized vector VLIW baseband processor. It can be seen that the cycle count numbers have been drastically reduced due to these optimizations.

Fig. 8 and Fig. 9 illustrate the impact of different optimizations on performance. For this, the baseband algorithms were profiled on the vector VLIW architecture while gradually enabling optimizations in the algorithms. The improvements

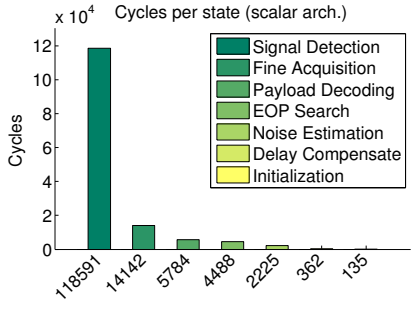

Fig. 6. Cycles per algorithm on scalar architecture

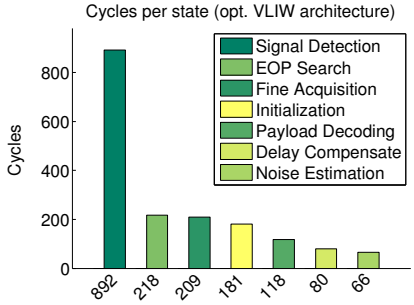

Fig. 7. Cycles per algorithm on vector VLIW architecture

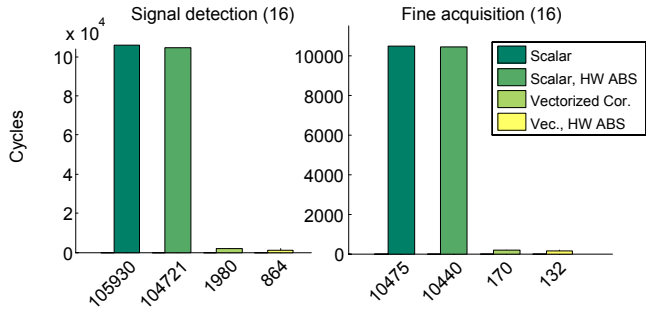

Fig. 8. Degrees of performance optimization for synchronization (vector VLIW architecture)

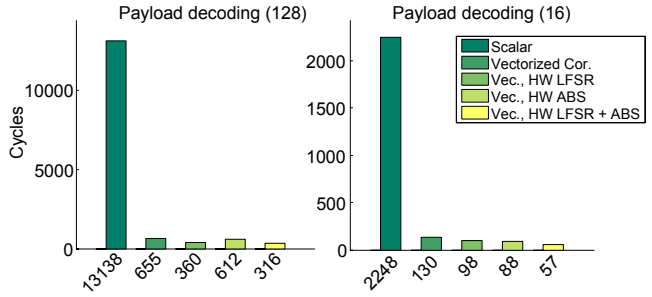

Fig. 9. Degrees of performance optimization for payload decoding (vector VLIW architecture)

due to vectorization and enabling different custom instructions, i.e. vector correlation, addition of absolute values and LFSR emulation (see Section IV-B), are shown as compared to a scalar implementation. For the signal detection and fine acquisition algorithms cycle count reductions by factors of 122x and 79x can be achieved. For the payload decoding algorithm the combined use of all architectural improvements enables a reduction by factors of 39x (spreading code length 16) and 41x (SC length 128).

It is observed that the signal detection during synchronization is the most cycle-intensive state, hence determining the required clock frequency. Only due to all architectural optimizations, real-time processing of baseband data at a relatively moderate clocking frequency of 225 MHz is made feasible. At this frequency, a computational performance of 7.65 GOPS is achieved by the VLIW architecture as compared to 0.23 GOPS and 3.6 GOPS for scalar and vector architecture respectively.

B. Power and Energy Results

Out of active and idle power consumption as well as the execution time profiling information, energy values and average power consumption values per algorithm state are calculated.

Fig. 10 summarizes the average power values obtained for different baseband algorithms running on the final UWB baseband processor. It can be seen that during the execution of the signal detection (SD) and the EOP search algorithms for a

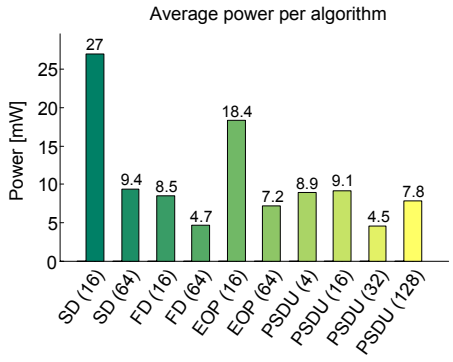

Fig. 10. Average power consumption of different baseband sub-algorithms (vector VLIW architecture)

spreading code length of 16, the average power consumption is quite high. This is due to the fact that in these phases the computational effort is high while the time span available for processing is very limited. For the fine acquisition (FA) and the decoding of the actual payload data (PD) on the other hand, the duty cycle is considerably smaller, resulting in a lower average power consumption.

Fig. 11 and Fig. 12 depict the decrease of energy consumption for different versions of the synchronization and the payload decoding algorithms. All algorithms are being profiled on the final vector VLIW baseband architecture but make use of different processor optimizations. The impact of vectorization as well as the benefit of enabling different custom

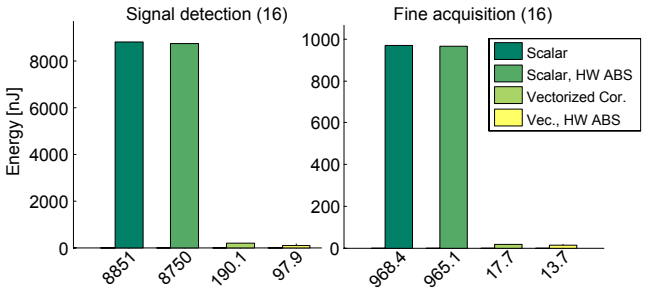

Fig. 11. Energy optimization results for synchronization (vector VLIW architecture)

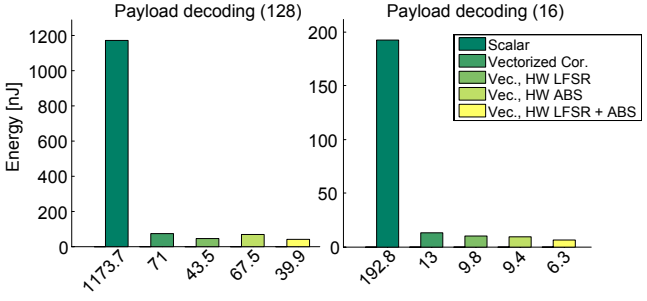

Fig. 12. Energy optimization results for payload decoding (vector VLIW architecture)

instructions, as introduced in Section IV-B, is shown. Due to the shorter run-times, energy consumption is reduced by a factor of 90x for signal detection algorithm and 70x for the fine delay search state. For the payload decoding, improvements by a factor of 30x for a spreading code length of 16 and a factor of 29x for a SC length of 128 are achieved as compared to the unoptimized version. These improvements enable energy-efficient baseband processing.

C. Area Results

The overall area of the final baseband processor (vector VLIW) after synthesis, place and route, is 0.77 mm^2. Program, scalar data and vector data memories contribute to almost 60% of this area. Large data memories are required to buffer incoming data of the radio front-end. A large part of the processor core itself is used by the vector register file (58%) that is especially needed for synchronization. The UWB IR functional units implementing the custom instructions as mentioned in Section IV-B only contribute to a small part ($\approx$3%) of the core area.

VI. Conclusions

Novel wireless sensor networks require ultra-low power radios for data communications. The IEEE 802.15.4a ultra-wideband impulse radio amendment represents a promising physical layer for energy-efficient, low data rate communications. In this paper we present an application-specific processor architecture tailored to the needs of digital baseband processing.

Within the design process, various optimizations on different levels of abstraction are introduced. By exploiting these optimizations and migrating from a scalar to a vector and finally to a vector VLIW architecture, performance speed-ups in the range of 39x to 122x for payload decoding and signal detection algorithms are presented. Furthermore, energy consumption can be decreased by a maximum of 30x for payload decoding and 90x for signal detection as compared to the baseline implementation.

We conclude that an energy-efficient, low-power baseband ASIP implementation for IEEE 802.15.4a is feasible while still offering a greater amount of flexibility due to programmability than an ASIC.

Acknowledgement

We would like to thank Target Compiler Technologies and IMEC for their support in the course of this project.

References

[1] B. Gyselinckx, R. Vullers, C. Hoof, J. Ryckaert, R. Yazicioglu, P. Fiorini, and V. Leonov. Human++: Emerging Technology for Body Area Networks. *VLSI-SoC*, 2006.

[2] M. Badaroglu, C. Desset, J. Ryckaert, V. D. Heyn, G. V. der Plas, P. Wambacq, and B. V. Poucke. Analog-digital partitioning for low-power UWB impulse radios under CMOS scaling. *EURASIP*, 2006.

[3] J. Govers, J. Huisken, M. Berekovic, O. Rousseaux, F. Bouwens, M. D. Nil, and J. L. van Meerbergen. Implementation of an UWB Impulse-Radio Acquisition and Despreading Algorithm on a Low Power ASIP. *HiPEAC*, 2008.

[4] IEEE. *P802.15.4a Draft Amendment to IEEE Standard for Information technology - Telecommunications and information exchange between systems*, 2006.

[5] FCC. *Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems*, 2002.

[6] G. Goossens, D. Lanneer, W. Geurts, and J. Van Praet. Design of ASIPs in multi-processor SoCs using the Chess/Checkers retargetable tool suite. *Intl. Symp. on System-on-Chip*, 2006.

[7] C. Desset, M. Badaroglu, J. Ryckaert, and B. van Poucke. UWB Search Strategies for Minimal-Length Preamble and a Low-Complexity Analog Receiver. *SPAWC*, 2006.

[8] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and S. Kaijian. *Low Power Methodology Manual*. Springer, 2007.

[9] M. Munch, B. Wurth, R. Mehra, J. Sproch, and N. Wehn. Automating RT-level operand isolation to minimize power consumption in datapaths. *DATE*, 2000.

[10] G. Goossens, J. Van Praet, D. Lanneer, and W. Geurts. Ultra-Low Power? Think Multi-ASIP SoC!. *IP-07*, 2007.