A Reconfigurable Application Specific Instruction Set Processor for Convolutional and Turbo Decoding in a SDR Environment

Timo Vogt, Norbert Wehn

Microelectronic Systems Design Research Group
University of Kaiserslautern, 67663 Kaiserslautern, Germany
{vogt, wehn}@eit.uni-kl.de

Abstract

Future mobile and wireless communication networks require flexible modem architectures to support seamless services between different network standards. Hence, a common hardware platform that can support multiple protocols implemented or controlled by software, generally referred to as software defined radio (SDR), is essential. This paper presents a family of dynamically reconfigurable application-specific instruction-set processors (ASIP) for the application domain of channel coding in wireless communication systems. As a weakly programmable IP core, it can implement trellis based channel decoding in a SDR environment. It features binary convolutional decoding, and turbo decoding for binary as well as duobinary turbo codes for all current and upcoming standards.

The ASIPs consist of a specialized pipeline with 15 stages and a dedicated communication and memory infrastructure. Logic synthesis revealed a maximum clock frequency of 400 MHz and a total area of 0.42 mm^2 for a 65 nm technology. Simulation results for Viterbi and turbo decoding demonstrate maximum throughput of 196 and 34 Mbps, respectively, and outperforms existing SDR based approaches for channel decoding.

1 Introduction

In recent years, we have seen the emergence of an increasing number of wireless protocols. Incorporating many of these protocols, hand-held wireless devices represent already today a convergence of many disparate features, including wireless communication, real-time multimedia, and interactive applications, into a single platform. Next generation mobile communication networks will feature new services, especially *multi-access interoperability*, with increased data throughput. *Thus flexibility becomes a dominant aspect for the transceiver*. To provide the flexibility for supporting seamless services between various wireless networks, there is a high demand for a common hardware platform that can support multiple protocols implemented or controlled by software, generally referred to as *software defined radio*.

The focus of this paper is put on *channel decoding in mobile and wireless communications systems in the context of SDR*. Channel decoding is the central processing task of the outer modem. Here convolutional codes (CC), binary turbo codes (bTC), and duobinary turbo codes (dbTC) are established techniques. The implementation complexity of the encoding algorithms is negligible. The decoding algorithms, however, have a very high computational complexity. In [14] it was shown that channel decoding contributes, depending on the implementation platform, about 40% to the total computational complexity of the physical layer of a UMTS or a WLAN 802.11a system. Similar results are obtained by [19, 9, 7] for other systems. Hence, it is essential to provide efficient implementations for this task.

Most of today's platforms for digital baseband processing support one or just a few standards, with an embedded channel decoder coprocessor engine especially designed for the respective outer modem. E.g., the system-on-chip "Greenside" platform from STMicroelectronics [21, 24] integrates two hardwired decoder coprocessors (COPRO in Figure 1) for turbo and convolutional decoding of EDGE, UMTS and CDMA2000. Other examples for dedicated coprocessors supporting CC and bTC for UMTS are [5, 3]. Such architectures are not suited for SDR since they do not provide the required flexibility.

Recently, platforms for SDR [14, 18, 1, 10, 8, 20, 15] were presented. Most of these platforms are single or multiple SIMD (single-instruction multiple-data) vector processing engines to support computationally intensive signal processing tasks. However, in channel decoding algorithms, calculations are not the bottleneck. Efficient internal data management is key. Thus such architectures are not well suited for channel decoding. An architecture optimized for just this task is preferable, rather than utilizing the same type of processor for the whole modem. This approach is similar to today's modems that use hardwired coprocessors for channel decoding (see Figure 1). Our intention is to re-

978-3-9810801-3-1/DATE08 © 2008 EDAA

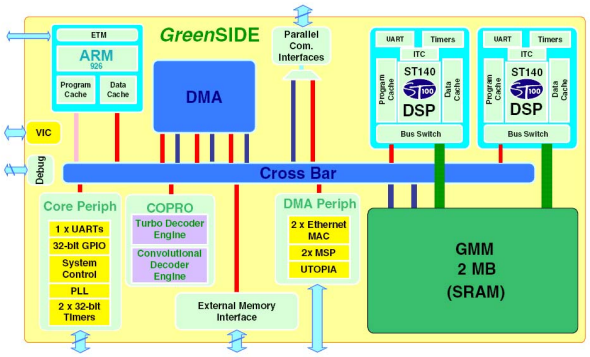

Figure 1. Baseband modem system-on-chip "Greenside" from STMicroelectronics [21, 24]

place the dedicated coprocessors, which are central building blocks of the platform, by an ASIP which can be considered as a *weakly programmable* processor. We will show that efficient utilization of application knowledge yields efficient and flexible architectures, as confirmed in a design study on a scalable reconfigurable channel decoder for future wireless handsets [12]. High performance combined with the advantages of processors, namely instruction level flexibility and a programming model, are achieved by ASIPs.

In [16] the first ASIP targeting the channel coding domain of UMTS turbo codes was presented. It is based on a customizable RISC processor, Tensilica's Xtensa. The gain in processing speed of this ASIP was limited by the classical RISC structure and its general memory architecture that is fixed for the Xtensa platform. Total freedom in pipeline and memory architecture design gives room for further improvement. Moreover, it allows to add application specific *run-time reconfigurability* to the ASIP approach: thus the flexibility requirements can be balanced between instruction level flexibility and reconfigurability.

An ASIP using this approach was proposed in [17], but it only targets binary and duobinary turbo codes with a maximum of 8 states. Convolutional codes, which have in case of large number of states similar computational complexity as turbo codes, and turbo codes with 16 states are not supported. The ASIP presented in this paper, named FlexiTreP (*Flexi*ble *Tre*llis *P*rocessor), resolves this lack of code support and furthermore achieves a higher data throughput, as will be shown in Section 4. It is a revised and strongly enhanced version of the processor introduced in [22]. The enhancements are in both functionality and performance. FlexiTreP now represents a whole *family* of processors since it can be tailored to the application scenario by *design-time configurability*. It combines run-time reconfigurability with design-time configurability. *FlexiTreP now implements turbo and convolutional decoding for all existing and emerging standards in the field of mobile wireless communication systems.* Moreover, detailed implementation results for FPGA and ASIC technologies are presented

Standard	Codes	States	Rates	Blocksizes	Throughput
GSM	CC	16,64	1/4...1/2	...876	...12 kbit/s
EDGE	CC	64	1/4...1/2	...870	...62 kbit/s
UMTS	CC	256	1/4...1/2	...504	...32 kbit/s
	bTC	8	1/3	...5114	...2 Mbit/s
CDMA2000	CC	256	1/6...1/2	...744	...28 kbit/s
	bTC	8	1/5...1/2	...20730	...2 Mbit/s
HSDPA	bTC	8	1/2...3/4	...5114	...14.4 Mbit/s
LTE	bTC	8	1/3	...5114	...100 Mbit/s
DAB	CC	64	1/4	none	...1.1 Mbit/s
DVB-H	CC	64	1/2...7/8	1624	...32 Mbit/s
DVB-T	CC	64	1/2...7/8	1624	...32 Mbit/s
DVB-RCT	dbTC	16	1/2,3/4	...648	...31 Mbit/s
IEEE802.11a/g	CC	64	1/2...3/4	...4095	...54 Mbit/s
IEEE802.16	CC	64	1/2...7/8	...4800	...20 Mbit/s
(WiMAX)	dbTC	8	1/3...7/8	...4800	...20 Mbit/s

Table 1. Selection of standards and channel codes

for a number of ASIP instances configured for different application scenarios. The fabrication of an ASIP instance in a 65nm ASIC technology is in progress.

2 Decoder Requirements

A careful analysis of channel codes incorporated in communication standards reveals that most of them are using binary convolutional and binary and duobinary turbo codes. Although block sizes, polynomials, and coding rates differ both within one coding scheme as well as between the coding schemes, it is possible to find commonalities for convolutional and turbo decoders which can be exploited for the decoder's architecture. The decoding algorithms of low density parity check (LDPC) codes vary substantially from CC and TC and are therefore not considered here.

Table 1 summarizes the relevant standards. Each standard has its own parameters for the channel encoder, such as different constraint lengths, generator polynomials, and, in case of turbo codes, different interleaving patterns. Thus, the ASIP has to support a wide range of coding parameters.

Turbo decoding either utilizes the Soft-Output Viterbi Algorithm (SOVA) or the Maximum Aposteriori Probability (MAP) algorithm. Both of these algorithms process soft intrinsic input and produce soft extrinsic output. However, the SOVA has a degradation of at least 0.3dB in communications performance compared to the MAP algorithm, and it also hits the noise floor earlier than its competitor. Hence, the MAP algorithm is preferable over the SOVA. For convolutionally encoded data, the standard decoding is performed by the Viterbi Algorithm (VA). In contrast to SOVA and MAP algorithm, it generates hard decision output values.

In summary, the following specifications were derived which have to be provided by the ASIP to fullfil the flexibility requirements for SDR systems:

- combined decoder for binary CC and binary and duobinary TC decoding,
- VA and Max-Log-MAP decoding for CC,
- support of $N = 16 \ldots 256$ states for CC,

- support of $N = 4 \ldots 16$ states for bTC,
- support of $N = 8 \ldots 16$ states for dbTC,
- arbitrary feedback and generator polynomials,
- rate flexibility by internal depuncturing of punctured codes,
- high throughput, low latency.

The throughput requirements vary strongly with the communication protocols and standards. In a mobile service like HSDPA, for instance, the maximum throughput for the turbo decoder is 14.4 Mbps, but only 3.6 Mbps are implemented today. We target a throughput of at least the maximum of HSDPA. If the required throughput is not achieved by a single processor, e.g. in a WLAN system, FlexiTreP can be configured for a multiprocessor decoder system.

3 Architecture

The ASIP is based on an architecture for hard and soft output convolutional decoding, first published in [22]. Here the decoding algorithm is programmable, and the code structure of the convolutional code is dynamically reconfigurable. The new ASIP presented in this paper inhibits the same concept but is extended to implement binary and duobinary turbo decoding. The interface is now well defined for a network-on-chip approach [23], and additional pipeline stages allow to implement interleaving and deinterleaving of extrinsic data. Hence, the iterative decoding process required for turbo decoding can now be implemented directly by the ASIP. Furthermore, the ASIP was extended for MAP decoding of duobinary codes with 16 states, and it now also features depuncturing functionality for nearly arbitrary punctering schemes. This functionality is unique for a channel decoder since it is usually performed before channel decoding. Since FlexiTreP implements this task inherently during decoding, overall processing time, latency, memory area, and energy is reduced.

Two major guidelines governed the design of the ASIP: first, an optimization for turbo decoding since this is the more demanding task, and second, a maximum reutilization of both computational hardware and memory for turbo and convolutional decoding.

The ASIP constitutes a *weakly programmable IP* block in a multi-core system. As such, it will always serve as a *slave* to a master processor (similar to the baseband modem in Figure 1). The interface, depicted in Figure 2, can be split into a system, a data-in and data-out, and a configuration interface. The input and output data can be *streamed* to and from the ASIP to support a data-flow process. Two special signals control the global data flow (*sleep, wakeup*). If in sleep mode, the datapath pipeline and all internal memories besides the currently accessed I/O memories are powered down. The configuration interface is used to write the channel code parameters to the Dynamically Reconfigurable Channel Code Control (DRCCC) unit, to load a new

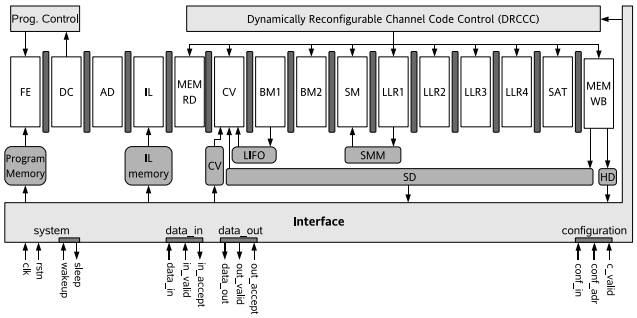

Figure 2. FlexiTreP general architecture, consisting of an interface, memories, a dynamically reconfigurable channel code control, a program control, and an instruction pipeline with 15 stages.

program, or to load a new deinterleaver table into the IL memory. The configuration interface is, in contrast to the data input and output interfaces, memory mapped.

As already mentioned, flexible data routing and data management is key for channel decoding. Thus the ASIP is based on a distributed memory concept, composed of various memories allocated to specific pipeline stages. Besides the channel value (CV) memory for the input and the hard decision (HD) memory for the output data, two additional memories are reserved to store internal data only: a state metrics memory (SMM) and a last-in-first-out (LIFO) which serves as buffer for channel and a-priori values. The soft decision (SD) memory serves for internal data storage, namely the a-priori values for turbo decoding or the decision bits of the local survivors of the Viterbi decoder, as well as for data output. Note that all memories are switched off when not accessed to reduce power consumption.

The DRCCC unit defines the structure of the convolutional (component) code that is in use. It controls the internal data routing of the datapath and is look-up-table (LUT) based. The unit includes a *shadow* and a *working* configuration. The working registers hold the configuration that is actually in use. The shadow registers allow for loading a new configuration without effecting the actual data processing. The content of the shadow registers is transferred to the working registers by a special instruction within one clock cycle, thus allowing a fast context switch between different codes. Moreover, the DRCCC unit allows for efficient multi-context instructions, reducing the instruction width from 65 to 24 bits which considerably saves area and energy.

The datapath pipeline consists of 15 stages and is depicted in detail in Figure 3. The first two stages are instruction fetch (FE) and instruction decode (DC). The third stage (AD) generates the addresses for the memory accesses in subsequent pipeline stages. The IL-pipeline stage accesses the IL memory, if necessary, to perform interleaving of the a-priori information upon read in the next pipeline stage (MEM). The DRCCC controls in the CV stage if one

	design-time	run-time
code classes	x	
code structure		x
decoding algorithm	x	x
memories	x	x

Table 2. Run-time and design-time configurability of FlexiTreP.

or three a-priori values are written to the a-priori register (APR) for binary or duobinary turbo codes, respectively. The CV stage also reads the channel values that are needed to compute the branch metrics. Note that the channel values themselves are not interleaved since Berrou's turbo decoding scheme is implemented [4]. All values required for the branch metric calculation are buffered in the LIFO buffer during the first recursion. They are retrieved from this buffer during the second. The buffering reduces energy consumption since memory access to the larger CV and AP/SUR memories are replaced by buffer accesses. The Viterbi algorithm does not utilize the LIFO buffer. The following pipeline stages perform the data manipulation required for turbo, MAP, or Viterbi decoding. The BM1 stage computes up to 16 branch metrics in parallel, or performs the trace back operation of the VA (not shown in Figure 3). The branch metrics are shuffled according to the DRCCC in stage BM2 and are utilized in SM to compute up to 16 state metrics in parallel [22]. The state metrics are directly forwarded from the output register of the SM pipeline stage to its input. Simultaneously they can be stored to the SMM for later use. This allows for fast computation of the metric recursions. In case of $N > 16$, a load store architecture is implemented: intermediate state metrics are loaded from the SMM to a pipeline register, processed, and then written back from the pipeline register to the SMM. A single trellis recursion with $N = 256$ states can thus be computed in 16 consecutive steps. The soft-output computation of the Log-MAP algorithm is pipelined due to critical path reduction and is located in stages LLR1 to LLR4. The SAT-stage computes the extrinsic information and saturates these values to eight bit. The stages LLR1 to SAT are idle during VA-operation. Finally, either the soft or the hard decoded output values are stored to a memory.

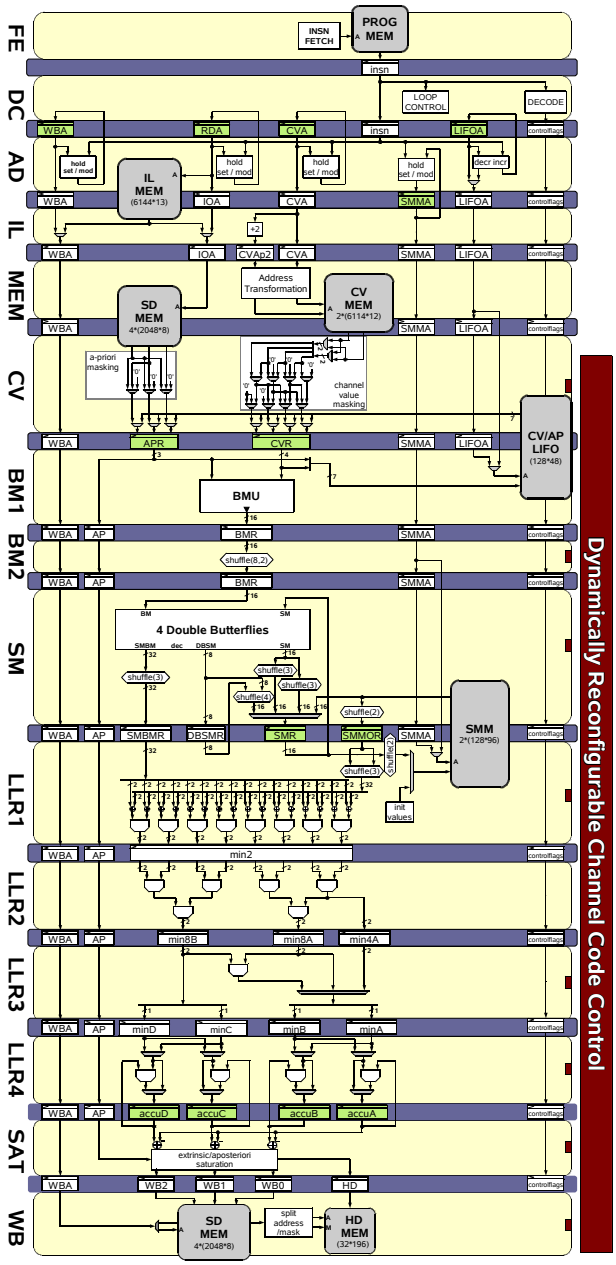

Figure 3. FlexiTreP: overall pipeline architecture configured for CC and TC decoding using the MAP algorithm

4 Results

The FlexiTreP pipeline architecture was modelled in LISA and verified with the ProcessorDesigner tool-set from Coware Inc [6]. This model is parametrizable to support the full set of codes as discussed before, or only a subset. Thus we provide design-time and run-time configurability (see Table 2).

Various VHDL instances of the FlexiTreP pipeline generated by the ProcessorGenerator tool were synthesized with the Synopsys Design Compiler. A 65 nm low power low leakage standard cell library with worst case conditions (1.10V, 120°C) is used as target technology. The total gate count of the different ASIP instances without memories is listed in Table 3. The maximum clock frequency for the FlexiTreP with full functionality is 400 MHz. It can be increased if only a subset of codes is supported, as shown in Table 3. The area overhead of the ASIP with full flexibility is 46% compared to a dedicated 16-state bTC decoder ASIP. Table 3 also shows that the soft output convolutional decoding comes with hardly any additional cost (only 2.4% logic area) compared to an instance that allows only hard output

Functionality	ASIC (65 nm, 1.10V, 120°C) Size [μm^2]	Frequency [MHz]	FPGA (Xilinx xc4vlx80-12) Size [Slices]	Frequency [MHz]
Full Functionality (with soft output for CC)	109320	400	7012	109
bTC, dbTC, and VA (without soft output for CC)	106762	400	6683	112
bTC, dbTC (duobinary with $N = 8$ only)	88966	415	5494	117
bTC	74391	450	4207	135

Table 3. ASIC and FPGA synthesis results for various instances of the ASIP with support of different code sets: Full functionality supports binary turbo codes (bTC), duobinary turbo codes (dbTC), and convolutional decoding as in Section 2.

	binary CC		bTC	dbTC
States	MAP best / typ [Mbps]	Viterbi [Mbps]	best / typ [Mbps]	best / typ [Mbps]
4	186 / 170	-	18.6 / 17	-
8	186 / 170	-	18.6 / 17	37.2 / 34
16	186 / 170	196	18.6 / 17	18.6 / 17
32	— / 27	78	—	—
64	— / 14	44	—	—
128	— / 5	23	—	—
256	— / 1.6	12	—	—

Table 4. Core throughput of FlexiTreP with full functionality for convolutional and turbo codes.

convolutional decoding with the VA (bTC, dbTC, and VA). The ASIP was also synthesized for a Xilinx Virtex4 FPGA and verified on a rapid prototyping platform. The results are shown in Table 3 as well. Here a maximum clock frequency of 109 MHz was achieved with full code support.

The total capacity of all memories used for the final implementation is 286 kbits, and the total area is around 0.31 mm^2. Maximum information block sizes of 6144 for bTC and 8192 for dbTC and CC at a coding rate $R = 1/2$ can be supported. Note that the memories can be taylored to the actual application scenario at design and run-time. The total gate count of the ASIP core in the 65nm standard cell technology without memories providing full functionality is 53 kilogate equivalents (kGE). Compared to the processors of [16] and [17] with 104 kGE and 93 kGE for the core's logic, respectively, the FlexiTreP saves more than 45% of the area, but provides much higher flexibility.

The core throughput achieved by FlexiTreP for convolutional and turbo decoding (TC at 5 iterations) for the different state numbers is listed in Table 4. In the *best* case, no acquisition is required, e.g., in case of block sizes smaller than 128 and 256 data bits for binary and duobinary codes, respectively. The numbers for the *typical* case consider windowing with acquisition. For MAP decoding of constraint lengths larger than five, the windows become very small. This disqualifies decoding without acquisition, and hence only typical throughput numbers are relevant. The throughput for Viterbi decoding is hardly effected by windowing due to very large window sizes (1024 and larger). The relatively large difference between 16 and 32 states constitutes in the load-store processing of the state metrics during partial parallel processing of a trellis segment.

The rate dependent input and output delay is not considered in Table 4. The total throughput can be computed from the core throughput and the code rate R as

$$throughput_{tot} = \frac{1}{throughput_{core}^{-1} + (\frac{R}{4} + \frac{1}{32})/400MHz}$$

assuming an I/O clock frequency of 400 MHz. This results, e.g., in a total throughput of 16.4 Mbps for a rate 1/3 binary turbo code with 5 iterations.

Table 5 summarizes turbo decoder implementations for UMTS turbo code applications on different state-of-the-art target platforms. The clock frequencies listed are maximum values. They differ, among other things, because the target technology is not the same. However, the FlexiTreP outperforms the other processor implementations even if they all run with the same clock frequency. This high performance is achieved due to the high internal memory bandwidth, the specialized data path pipeline, and the internal data shuffling and reordering mechanisms.

5 Conclusion

Application specific flexibility is mandatory to meet the flexibility and performance requirements of SDR. In this paper we presented a family of weakly programmable IP cores for the application domain of trellis based channel decoding: FlexiTreP. It features Viterbi and Max-Log-MAP processing for binary convolutional codes, and binary and duobinary turbo decoding of all actual and upcoming standards. In addition the ASIP provides the flexibility to adapt to evolving and new standards.

The dynamically reconfigurable ASIP approach allows for high throughput and small area while preserving flexibility. The presented ASIP can furthermore by configured during run-time to the application scenario. Several instances were generated and synthesized for ASIC as well as FPGA technology. Maximum throughput at 400MHz is 196 Mbps for Viterbi and 34 Mbps for (duobinary) turbo decoding. It thus outperforms state-of-the-art solutions targeting SDR.

6 Acknowledgments

This work has been supported by the DFG within the Schwerpunktprogramm "Rekonfigurierbare Rechensysteme".

	Implementation Technology	Size	Clock freq.	cycles/ (bit*MAP)	Throughput @ 5 iter
Conf. RISC[16]	ASIC (130nm)	104kGE	133 MHz	9	1.4 Mbps
Clustered VLIW[11]	VitexII-4000	517 slices	80 MHz	8	1 Mbps
XiRisc[13, 2]	ASIC (130nm)	—	100 MHz	100	0.1 Mbps
SODA[25]	ASIC (180nm)	1000kGE	400 MHz	20	2 Mbps
ASIP [17]	ASIC (90nm)	97kGE	335 MHz	6.5	5 Mbps
FlexiTreP	ASIC (65nm)	53kGE	400 MHz	2.35	17 Mbps
FlexiTreP	Virtex4	7012 slices	109 MHz	2.35	4.6 Mbps

Table 5. Comparison of turbo decoder implementations for UMTS bTC. FlexiTreP is implemented with full functionality here.

References

[1] Stretch. http://www.stretchinc.com.

[2] A. Baschirotto, R. Castello, F. Campi, G. Cesura, M. Toma, R. Guerrieri, R. Lodi, L. Lavagno, and P. Malcovati. Baseband Analog Front-End and Digital Back-End for Reconfigurable Multi-Standard Terminals. *Circuits and Systems Magazine, IEEE*, 6(1):8–28, First Quarter 2006.

[3] F. Berens, G. Kreiselmaier, and N. Wehn. Channel Decoder Architecture for 3G Mobile Wireless Terminals. In *Proc. 2004 Design, Automation and Test in Europe (DATE '04)*, Paris, France, Feb. 2004.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. In *Proc. 1993 International Conference on Communications (ICC '93)*, pages 1064–1070, Geneva, Switzerland, May 1993.

[5] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. M. Davis, G. Woodward, C. Nicol, and R. Yan. A Unified Turbo/Viterbi Channel Decoder for 3GPP Mobile Wireless in 0.18-μm CMOS. *IEEE Journal of Solid-State Circuits*, 37(11):1555–1564, Nov. 2002.

[6] Co-Ware. http://www.coware.com.

[7] M. M.-C. T. T. for Integrated Broadband Cellular Systems. Synthesis report on the performance and complexity obtained with the HW and SW platforms (D7.2). http://ist-matrice.org.

[8] J. Glossner, D. Iancu, M. Moudgill, G. Nacer, S. Jinturkar, and M. Schulte. The Sandbridge SB3011 SDR Platform. In *Joint IST Workshop on Mobile Future and the Symposium on Trends in Communications (SympoTIC '06)*, pages ii–v, 24-27 June 2006.

[9] M. Hosemann, R. Habendorf, and G. P. Fettweis. Hardware-Software Codesign of a 14.4Mbit - 64 State - Viterbi Decoder for an Application-Specific Digital Signal Processor. In *Proc. IEEE Workshop on Signal Processing Systems 2003 (SIPS'03)*, Seoul, Korea, Aug. 2003.

[10] IMEC. Scientific Report 2006: Software Defined Radio Flexible Air Interface. www.microelektronica.be/ wwwinter/mediacenter/en/SR2006/681340.html, 2006.

[11] P. Ituero and M. Lopez-Vallejo. New Schemes in Clustered VLIW Processors Applied to Turbo Decoding. In *Application-specific Systems, Architectures and Processors, 2006. ASAP '06. International Conference on*, pages 291–296, Sept. 2006.

[12] G. Krishnaiah, N. Engin, and S. Sawitzki. Scalable Reconfigurable Channel Decoder Architecture for Future Wireless Handsets. In *Proc. 2007 Design, Automation and Test in Europe (DATE '07)*, 2007.

[13] A. LaRosa, C. Passerone, F. Gregoretti, and L. Lavagno. Implementation of a UMTS Turbo-Decoder on a dynamically reconfigurable platform. In *Proc. 2004 Design, Automation and Test in Europe (DATE '04)*, Paris, France, Feb. 2004.

[14] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. SODA: A Low-power Architecture For Software Radio. In *Proc. 33rd International Symposium on Computer Architecture (ISCA'06)*, pages 89–101, 2006.

[15] E. Matu, H. Seidel, T. Limberg, P. Robelly, and G. Fettweis. A gflops vector-dsp for broadband wireless applications. In *Conference 2006, IEEE Custom Integrated Circuits*, pages 543–546, 10-13 Sept. 2006.

[16] H. Michel, A. Worm, Münch, and N. Wehn. Hardware/Software Trade-offs for Advanced 3G Channel Coding. In *Proc. 2002 Design, Automation and Test in Europe (DATE '02)*, Paris, France, Mar. 2002.

[17] O. Muller, A. Baghdadi, and M. Jezequel. ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding. In *Proc. 2006 Design, Automation and Test in Europe (DATE '06)*, Munich, Germany, Mar. 2006.

[18] PACT XPP Technologies. PACT home page. www.pactcorp.com.

[19] C. Pan, N. Bagherzadeh, A. Kamalizad, and A. Koohi. Design and Analysis of a Programmable Single-Chip Architecture for DVB-T Base-Band Receiver. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 468–473, 2003.

[20] U. Ramacher. The Future of Mobile Computing. In *Proc. 6th International Symposium on Multiprocessor Systems-on-Chips (MPSoC'06)*, pages 281–294, Estes Park, Colorado, USA, August 2006.

[21] ST Microelectronics Greenside Platform. http://www.st.com/stonline/products/literature/ta/11145.htm.

[22] T.Vogt and N.Wehn. A Reconfigurable Application Specific Instruction Set Processor for Viterbi and Log-MAP Decoding. In *Proc. IEEE Workshop on Signal Processing (SIPS'06)*, pages 142–147, Banff, Canada, October 2006.

[23] P. Vivet, D. Lattard, F. Clermidy, E. Beigne, C. Bernard, Y. Durand, J. Durupt, and D. Varreau. FAUST, an Asynchronous Network-on-Chip based Architecture for Telecom Applications. In *Proc. 2007 Design, Automation and Test in Europe (DATE '07)*, 2007.

[24] T. Vogt, N. Wehn, and P. Alves. A Multi-Standard Channel-Decoder for Base-Station Applications. In *Proc. 17th Symposium on Integrated Circuits and System Design (SBCCI) 2004*, pages 192–197, Porto de Galinhas, Brazil, Sept. 2004.

[25] L. Yuan, S. Mahlke, M. Trevor, C. Chaitali, R. Alastair, and F. Krisztian. Design and Implementation of Turbo Decoders for Software Defined Radio. In *Proc. IEEE 2006 Workshop on Signal Processing Systems (SiPS)*, 2006.