A Cross-Layer Reliability Design Methodology for Efficient, Dependable Wireless Receivers

CHRISTINA GIMMLER-DUMONT and NORBERT WEHN, University of Kaiserslautern

Continued progressive downscaling of CMOS technologies threatens the reliability of chips for future embedded systems. We developed a novel design methodology for dependable wireless communication systems which exploits the mutual trade-offs of system performance, hardware reliability, and implementation complexity. Our cross-layer approach combines resilience techniques on hardware level with algorithmic techniques exploiting the available flexibility in the receiver. The overhead is minimized by recovering only from those hardware errors that have a strong impact on the system behavior. We apply our new methodology on a double-iterative MIMO-BICM receiver which belongs to the most complex systems in current communication standards.

Categories and Subject Descriptors: B.8.0 [Performance and Reliability]: General; B.7.1 [Integrated Circuits]: Types and Design Styles—*Algorithms implemented in hardware; VLSI (very large scale integration)*

General Terms: Reliability, Design, Performance, Algorithms

Additional Key Words and Phrases: Design for dependability, cross-layer reliability, wireless communication systems, MIMO-BICM systems, iterative MIMO detection and channel decoding, algorithmic error resilience

ACM Reference Format:

Christina Gimmler-Dumont and Norbert Wehn. 2014. A cross-layer reliability design methodology for efficient, dependable wireless receivers. *ACM Trans. Embedd. Comput. Syst.* 13, 4s, Article 137 (March 2014), 29 pages.

DOI: <http://dx.doi.org/10.1145/2584666>

1. INTRODUCTION

Technology scaling has reached a point at which process and environmental variabilities are no longer negligible, and can no longer be hidden from system designers, as the exact behavior of CMOS devices becomes increasingly less predictable. This will show in the form of static and dynamic variations, time-dependent device degradation and early life failures, sporadic timing errors, radiation-induced soft errors, and lower resilience to varying operating conditions [Nowka et al. 2008]. Already today, conservative margining, guard banding, and conservative voltage scaling come at a large cost. Only turning away from conservative worst-case design methodologies for a 100% reliable physical hardware layer will make further downscaling of CMOS technologies a profitable endeavor [DAC Roundtable 2010; Mitra et al. 2011]. This calls for radically new cross-layer design concepts.

Until today, these problems have mostly been addressed at the lower design levels. At the higher levels, systems are typically designed under the premise of fault-free underlying hardware. Only in extremely critical applications, such as avionics, where the system cost is less important than its dependability, Triple Modular Redundancy

This work was partly supported by the German Research Foundation (DFG) as part of the priority program “Dependable Embedded Systems” (SPP 1500).

Authors’ addresses: C. Gimmler-Dumont (corresponding author) and N. Wehn, Microelectronic Systems Design-Research Group, University of Kaiserslautern, Germany; email: gimmler@eit.uni-kl.de.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

2014 Copyright is held by the author/owner(s)

1539-9087/2014/03-ART137

DOI: <http://dx.doi.org/10.1145/2584666>

(TMR) and similar techniques are employed on a system level. Thus, to no big surprise, the large body of related work focuses on low-level techniques to present higher abstraction and design levels with a practically error-free platform built from potentially faulty elements.

To make a platform resilient against transient or permanent faults, built-in redundancy or built-in self-recovery techniques have to be employed. They all come at the cost of chip area, power consumption, reduced system throughput, or other implementation-related metrics. Lower implementation cost, especially with regard to energy consumption, can be obtained when a degradation of hardware reliability to a certain degree is tolerated. In fact, energy consumption and dependability of integrated circuits can be seen as strongly interrelated problems: by decreasing the operating voltage, the energy efficiency increases but at the same time the dependability decreases. Thus, energy efficiency and dependability have to be carefully traded off against each other.

An error-resilient architecture that can be seen as practically error free can be composed of protected components. Applications for these platforms can be implemented in a traditional way, still assuming fault-free operation of the underlying hardware. In addition to this horizontal integration, recent research also evaluates the additional potential of a vertical integration of error resilience on the application level with platforms having a reduced reliability. True *cross-layer optimization* approaches do not only exploit the fact that some important classes of algorithms are inherently error tolerant, but also adapt applications and hardware architectures jointly to achieve the best possible trade-offs. This article focuses on cross-layer optimization for wireless communication systems.

1.1. Existing Approaches for the Design of Dependable Systems

The need to detect the occurrence of errors and provide a recovery mechanism is common to all resilience approaches. Memories were the first devices which employed built-in redundancy and error resilience intensively to increase production yield. In logic circuits, majority voting and TMR are trivial but very costly solutions. More advanced circuit-level techniques have been proposed, tailored to protect particular components. Examples are Razor flip-flops [Ernst et al. 2003] to detect signal delay errors, techniques to reduce interconnect noise [Elgamel and Bayoumi 2003], self-checking mechanisms for combinational circuits [Nicolaidis 1999], error correcting codes for memories [Gherman et al. 2009], and techniques of built-in soft-error resilience for correcting soft errors in latches and flip-flops [Krishnamohan and Mahapatra 2004; Mitra et al. 2007]. Error Detection and Correction (EDC) codes, first used in memories, are also applied for interconnect networks, for example, in Murali et al. [2005] and Ejlali et al. [2007].

Application-level dependability was identified as the key to minimize the implementation overhead for error detection and recovery. However, compared to the techniques for individual components described previously, the available base of research is still small. A large number of important and relevant applications such as probabilistic algorithms and audio and video processing show a certain, inherent error resilience [Bau et al. 2007; May et al. 2008], which does not require the underlying hardware platform to be completely error free. This tolerance towards various kinds of errors can be attributed to one of the following categories.

—*Algorithmic resilience* is given when a certain amount of errors can be tolerated by the algorithm itself without rendering the outcome of the calculations invalid. This is typical of probabilistic applications and iterative algorithms. They can be found,

for instance, in the field of Recognition, Mining, and Synthesis (RMS) and in wireless systems. A further application is algorithms which can tolerate statistical behavior, such as fixed-point DSP algorithms and numerical calculations as found in many signal processing applications.

- Cognitive resilience* stems from the interaction of an application with a human being like in audio and video processing. Here, errors are tolerable as long as the user cannot discern quality differences, or accepts them as a trade-off for a longer battery lifetime, for example.

Taking into account the application-inherent error resilience at the architectural level clears the way for a further reduction of the architectural overhead for error detection and recovery, and for techniques such as Voltage OverScaling (VOS), where the voltage is reduced beyond the point at which error-free operation of the circuit is guaranteed. By providing an application with only the minimum sufficient degree of hardware dependability required to achieve the desired reliability on the application level, the energy efficiency can be optimized.

The potential of interrelating application performance and resilience has been shown in several publications. In Shanbhag et al. [2010], the Algorithmic Noise Tolerance (ANT) approach is proposed in which an unreliable, ultra energy-efficient main block executes most of the computations. This main block can make intermittent errors, as long as they occur infrequently. A reliable error control block detects and corrects the errors. Application resilience against errors in the Least Significant Bits (LSBs) of numerical values has been analyzed by, for example, Karakonstantis et al. [2007] and Shanbhag [2002]. Other examples are the techniques of stochastic logic [Qian et al. 2011], probabilistic switching [Palem 2005], significance-driven computation [Mohapatra et al. 2009], probabilistic CMOS [George et al. 2006], or Chakrapani et al. [2007]. The scalable effort hardware design [Chippa et al. 2010] identified and made control knobs available on algorithm, architecture, and circuit level which allow tuning after fabrication to improve the energy efficiency. The HERQULES framework [Karakonstantis et al. 2010] presented an optimization framework for design space exploration of trade-offs between energy consumption by VOS and implementation complexity demonstrated by change of quantization and different adder structures. In all these cases, power savings are achieved by the toleration of computational errors on the architectural level, which have no or negligible impact on the application performance.

In contrast to approaches shown in earlier studies, the Error-Resilient System Architecture (ERSA) [Bau et al. 2007; Leem et al. 2010] is a cross-layer approach for probabilistic applications from the RMS domain, which adapts the platform and the applications jointly. ERSA is a multiprocessor system-on-chip consisting of many undependable cores for fast, energy-efficient, but error-prone calculations, accompanied by a number of dependable cores for operations that do not tolerate errors. Interconnects and memories are protected using standard methods. Calculations performed by the undependable cores are accepted depending on lightweight sanity checks. Investigations on RMS algorithms (K-means clustering, Bayes-net structure learning, belief propagation algorithms like Low-Density Parity-Check (LDPC) decoding) show that ERSA can maintain a high decoding performance even at comparatively high error rates. While ERSA focuses on single algorithms only, the approach presented in Khajeh et al. [2008] considers a wireless video transmission system and takes this idea even a step further, investigating the potential of tuning the resilience of the application. Power savings are achieved by, on the one hand, a lower compression rate of the video (application layer) and, on the other hand, selective protection on the network layer and aggressive voltage scaling on the physical layer. This approach was extended in

Table I. State-of-the-Art Publications in Error-Resilient Wireless Communication Systems

Reference	System/Component	Error Locations
[Hussien et al. 2010]	Viterbi decoder	input LLR memory
[Eltawil and Kurdahi 2006]	Turbo decoder	input LLR memory
[Karakonstantis et al. 2012]	Turbo decoder	H-ARQ LLR memory
[Geldmacher et al. 2011]	Turbo equalization	all LLR memories
[Novak et al. 2010]	MIMO-BICM, open loop	decoder input LLR memory
[May et al. 2010b]	Turbo decoder	bus communication
[Abdallah and Shanbhag 2009]	Viterbi decoder	recursion unit
[Liu et al. 2009]	Viterbi decoder	recursion unit
[Brehm et al. 2012]	Turbo decoder	all LLR memories, recursion unit
[May et al. 2008]	LDPC decoder	memories, functional units, controller, network
[Kairy et al. 2012]	MIMO detector	complex detector input memory
[Gimmmler-Dumont et al. 2012a]	MIMO-BICM, closed loop	all LLR + complex data memories

Khajeh et al. [2012] by the development of an analytical error model for embedded memories and an adaptive power management policy.

These are the first cross-layer approaches to resilient systems. However, they mainly exploit runtime adaptivity as a means of energy optimization, for example, by Dynamic Voltage and Frequency Scaling (DVFS), thus trading off system quality and implementation performance (energy, area). In order to exploit the full potential of reliability-aware cross-layer design, we propose to add the hardware reliability as a third design criterion. This gives rise to additional trade-offs of system performance and system dependability.

1.2. State-of-the-Art in Dependable Wireless Systems

Wireless communication systems are an excellent application to investigate all the mutual effects of tuning system performance, system dependability, error resilience, and implementation complexity in combination. While runtime adaptivity is common in today's communications standards like HSDPA (High-Speed Downlink Packet Access) or LTE (Long-Term Evolution) [Third Generation Partnership Project 2008], which specify higher data throughput rates for higher Signal-to-Noise Ratios (SNR), these techniques have never been applied to increase system error resilience with respect to hardware failures. To date, only few publications deal with error resilience in wireless communication systems. These are discussed in the remainder of this section.

The recently published research articles deal primarily with the question whether existing wireless receivers are already error resilient against hardware errors and how their resilience can be improved with little implementational overhead. The main focus of research on dependability in wireless communication systems (see Table I) lies on the error resilience of the channel decoder. Channel decoding algorithms qualify for resilience analysis due to their probabilistic behavior and their purpose of correcting errors. According publications analyze transient errors in the memories [Hussien et al. 2010; Eltawil and Kurdahi 2006; Geldmacher et al. 2011; Novak et al. 2010] or consider timing errors [Abdallah and Shanbhag 2009; Liu et al. 2009]. So far, only very few articles consider both types of errors, for example, May et al. [2008] and Brehm et al. [2012]. A detailed discussion of the aforementioned publications can be found in Brehm et al. [2012]. Here, we restrict ourselves to a discussion of the main trends in the related literature.

Many existing works propose aggressive VOS to improve the system energy efficiency [Eltawil and Kurdahi 2006; Hussien et al. 2010; Geldmacher et al. 2011; Novak et al. 2010]. The resulting errors can cause a severe degradation of the system performance. Therefore, different methods are proposed to counterbalance these additionally induced errors. However, these works often neglect that the implementation of, for example, more complex computation units for the adjacent channel decoder/equalizer [Hussien et al. 2010; Geldmacher et al. 2011] or the error protection of the memory contents [Novak et al. 2010] result in throughput and/or power penalties. A fair judgment of the effectiveness of aggressive voltage scaling on system level requires these penalties to be put into relation to the achievable power savings. In contrast to this, Karakonstantis et al. [2012] propose to increase the manufacturing yield by correcting errors in the input memory via the system's hybrid automatic repeat request (H-ARQ).

Up to now, publications concentrate on the channel decoder component and its dominating memory block, the so-called LLR (Log-Likelihood Ratio) input buffer. However, iterative channel decoders also contain large internal memories which can suffer from unreliable storage [May et al. 2008; Brehm et al. 2012]. In addition, wireless receivers may comprise large iterative systems in which the channel decoder is only one component among others, such as iterative Multiple-Input Multiple-Output (MIMO) detection, turbo synchronization, or turbo equalization. These large systems also contain data buffers for signal processing which are not based on likelihoods, but store, for example, information about channel characteristics. Errors in even the less significant quantization bits of this information may have a large influence on the overall behavior of the receiver. Every bit in this data will always affect several information bits and is thus more critical than errors in likelihood data [Gimmler-Dumont et al. 2012a].

All state-of-the-art publications have in common that they mostly utilize static resilience techniques to combat the effects of unreliable hardware, for example, error protection of memories. Static methods have the disadvantage to permanently decrease the system performance in at least one of the terms of throughput, area, or power consumption, even when no errors occur. In most cases, it is preferable to counterbalance a short-term hardware error with a dynamic mitigation technique, which is only activated when errors are detected.

More importantly, all these publications present only single solutions to individual parts of the reliability problem. To date, there is no methodology for the design of dependable wireless systems that combines all these efforts.

1.3. Contributions

This article has three main contributions.

- We present a new methodology for the design of dependable wireless communication systems by exploiting the mutual trade-offs of system performance, hardware reliability, and implementation complexity (Section 2). In our reliability-aware cross-layer approach the system responds dynamically at runtime to the current reliability and service requirements.
- We present a flexible architecture for double-iterative MIMO-BICM receivers supporting many of the current communication standards (Section 3). Energy efficiency and error resilience are strongly correlated. Thus, we focus strongly on the design of a highly energy-efficient, weakly programmable MIMO detector. Our novel architecture achieves energy efficiency through the support of several kinds of detection algorithms, allowing a trade-off of communications performance and throughput at runtime. Offering this flexibility is a necessary requirement for the proposed cross-layer reliability methodology.

Fig. 1. Wireless communication systems suffer from different sources of errors: channel errors, quantization errors, errors from suboptimal algorithms, and hardware errors.

—We propose application-specific resilience techniques for double-iterative MIMO-BICM receivers (Section 4). These techniques are realized on different layers of abstraction ranging from low-level to high-level techniques. A reliability control unit chooses the techniques offering just the necessary reliability with the smallest overhead. To the best of our knowledge, this is the first analysis on the error resilience of MIMO detection and double-iterative MIMO-BICM receivers.

2. METHODOLOGY: ERROR MITIGATION USING DYNAMIC RESILIENCE ACTUATORS

Wireless communication systems have an inherent error resilience. They are designed to recover the originally transmitted data sequence in spite of errors that occur during transmission over a noisy channel; see Figure 1. To achieve a reliable transmission, today's communication systems use advanced Forward Error Correction (FEC) techniques, which employ redundancy that the sender adds to the actual information prior to transmission. This redundancy combined with information about channel characteristics is used on the receiver side to correct transmission errors. Implementation efficiency of communication systems in hardware mandates quantization of data values and the use of suboptimal algorithms, that is, algorithms that generate results which deviate from the theoretically correct values. Both can be seen as further sources of errors in addition to the noise on channel. In the same way, errors induced by hardware faults can be considered as yet another error source in a communication system.

Modeling of hardware errors is crucial for the design of dependable systems. Radiation, thermal effects, aging, or process or parameter variations cause distortions on a physical level which can be modeled by probabilistic bit flips according to the Resilience Articulation Point (RAP) model [Herkersdorf et al. 2013]. Depending on its location, a bit flip can have very different effects. An error in the controller, for example, usually leads to a system malfunction, whereas individual errors in the memories or the dataflow are often inherently corrected by a wireless receiver [May et al. 2008; Gimpler-Dumont et al. 2012a]. Efficiency in terms of area and energy will be achieved by recovering only from those errors which have a significant impact on the system output and by choosing the layer on which the treatment of these error results in the least overhead.

Dynamic approaches for error resilience also have to monitor the current hardware status. This monitoring can be done on different abstraction layers. Examples are Error Detection Sequential (EDS) circuits on microarchitectural layer. EDS circuits are very popular [Das et al. 2009], however, they require pre- and post-silicon calibration. Monitors on higher abstraction layers are application specific and normally more efficient. For example, Brehm et al. [2012] proposed to detect timing errors with a small additional hardware block which mimics the critical path under relaxed timing constraints. The result of the mimic hardware is compared to the normally operating unit.

Fig. 2. The standard communication flow dynamically adjusts communication parameters to achieve the required QoS, for example, the code rate in Hybrid-ARQ systems.

Deviations indicate timing errors. For a turbo and convolutional code decoder, the mimic hardware only required 0.7% of the decoder area. In this article we focus on resilience techniques which are employed after hardware errors have been detected, not on the detection methods themselves.

Many state-of-the art publications utilize low-level static resilience techniques to combat the effects of unreliable hardware, for example, ECC protection of memories, Razor flip-flops, or stochastic logic [Qian et al. 2011]. Static methods have the disadvantage of permanently decreasing the system performance in at least one of the terms of throughput, area, or power, even when no errors occur. In May et al. [2008] for example, the static protection of a complete LDPC decoder for WiMax/WiFi resulted in an area overhead of 21%.

Dynamic techniques often use available hardware resources or have very low additional costs as we will show in Section 4. However, error detection circuits result in additional costs. When comparing static and dynamic methods, this additional cost has to be taken into account. In general, the choice of the protection method will also depend on the expected hardware error statistics as we will demonstrate in the next paragraph. Eventually, a combination of static and dynamic protection will likely result in the least overhead.

2.1. The Dynamic Behavior of Wireless Systems

Modern wireless communication standards like LTE provide mechanisms to monitor and dynamically adapt to changes in the Quality-of-Service (QoS). The QoS in a wireless transmission system is typically defined as the bit error rate with respect to a given signal-to-noise ratio. If the desired QoS cannot be achieved for the current transmission channel, communication parameters like code type, code rate, etc., are adjusted to improve the communications performance (see Figure 2(a)). A good example for this dynamic behavior is the hybrid automatic repeat request (H-ARQ), which is used in wireless communication standards such as LTE, HSPA. These systems typically

transmit blocks of data at a high data rate and with little error protection, that is, with a very high code rate. If the decoder fails, the transmission of additional data is requested until the block is correctly decoded. Note that such a retransmission does not contain the same data as before. Instead, different information will be sent every time, which had been punctured on the transmitter side before. The additional information decreases the data rate but at the same time increases the probability that the block can be correctly decoded at the receiver. Figure 2(b) shows the throughput of a H-ARQ system over different SNR values. For high SNR values, decoding succeeds after the first transmission, that is, the channel decoder can correct all errors, and a high throughput is obtained. With a decreasing SNR, more and more blocks require additional transmissions and the throughput is lowered. The system dynamically adapts the code rate and the throughput for each block.

This example shows how wireless receivers adapt dynamically to changes in the transmission channel, that is, varying SNR, and correct transmission errors. The question is how this idea can be applied to the case of hardware errors. Several research groups have looked into this direction recently (see Section 1.2). They have shown that low rates of hardware errors in a wireless receiver are not visible on the system level. This is due to the fact that for low SNR the channel errors dominate and for high SNR the channel decoder can correct the hardware errors. For moderate hardware error rates, some dynamic high-level techniques exist, for example, increasing the number of decoder iterations to counterbalance the impact of hardware errors. However, for very high error rates on the hardware level, a purely software-based mitigation is not possible. An increase of reliability can generally be achieved by either *static low-level techniques*, like, for example, Razor flip-flops, triple-modular redundancy, or by *dynamic high-level techniques* which exploit the flexibility of the receiver, such as increase of decoder iterations, or a combination of both. To their advantage, dynamic techniques are mainly algorithmic changes, which can be controlled by software and do not require a more costly change of the underlying hardware.

Consequently, it is possible to use high-level techniques to mitigate hardware errors in wireless communication systems. However, transmission through wireless channels is unreliable and the channel quality changes over the time. Channel noise and hardware noise may change independently from each other. In good channel conditions, we can use a part of the error correction capability of the receiver to combat hardware errors if needed. When the channel quality is very poor, all high-level techniques are needed to obtain the required QoS, and hardware errors have to be counterbalanced by static low-complexity methods. This is also demonstrated in Figure 3(b). When the hardware reliability is very high, no action has to be taken. High amounts of hardware errors cannot be overcome using dynamic techniques exclusively. A combination of dynamic and static techniques is mandatory. When the channel quality is very poor, only static techniques are available. For medium noise levels, there are potential trade-offs between dynamic and static techniques.

2.2. Concept of Dynamic Resilience Actuators

In current standards, like HSDPA or LTE, the QoS is dynamically adjusted at runtime, for example, higher data throughput rates are specified for higher SNR. This is due to the fact that the computational requirements on the different algorithms decrease with higher SNR in order to enable higher throughput. In future technologies the negotiated QoS may also depend on the reliability of the receiver hardware under given operating conditions. This leads to an entirely new paradigm—adaptive QoS with respect to communication reliability and hardware reliability. An illustration of this is the possibility to relax reliability requirements on the underlying hardware instead of providing a higher throughput at high SNR. An example for this is the use of

(a) extended dynamic QoS flow: the reliability control unit chooses the resilience actuators which result in the least overhead and, thus, in an energy-efficient design

(b) depending on the current hardware reliability and channel quality, different static and dynamic techniques (resilience actuators) are available to mitigate the impact of hardware errors (entries in cyan color quantify the example in Section 2.3)

Fig. 3. Our new methodology integrates seamlessly into the existing QoS flow of today's communication systems. The available resilience techniques depend on the current channel quality and hardware reliability.

voltage overscaling, where the voltage is reduced beyond the point at which fault-free operation of the circuit is guaranteed in order to lower the power consumption of the receiver. In this way, QoS, hardware reliability, and implementation efficiency can be traded off against one another at runtime.

In Brehm et al. [2012], we presented how this new paradigm can be integrated into the existing QoS flow of wireless communication systems. Figure 3(a) shows the extended version of the original QoS flow from Figure 2(a). Low rates of hardware errors are implicitly corrected by a wireless receiver. In that case no further action is required. A higher rate of hardware errors results in a degradation of the QoS and, thus, can be detected by the standard QoS flow. The standard QoS flow is already error resilient by itself, as it dynamically adjusts the communication parameters to obtain a certain QoS. In most cases, however, it will be cheaper in terms of energy to correct a temporary hardware error by the activation of a dynamic protection mechanism than by changing the communication parameters as, for example, a H-ARQ-based correction is very costly with respect to energy consumption.

A degradation of the QoS can be caused by either channel errors or hardware errors. A differentiation of these two error sources is not possible with the existing QoS monitoring system only. Therefore, it is necessary to monitor the reliability status of each hardware component. However, exact information on gate level is not required. Single bit-flips in the data path, for example, are often mitigated by the algorithmic error resilience of the receiver. Application-specific detection circuits like the reduced-size ACS unit for turbo decoding proposed in Brehm et al. [2012] can indicate the status of one component with only a small overhead.

We introduced a reliability control unit which activates one or several *resilience actuators* according to the current monitoring status. A resilience actuator is a dynamic protection mechanism which can increase the error resilience either on component or

on system level. Resilience actuators can be found on hardware level and on software level. So far, we identified four classes of actuators. On the lowest level, we can change the *hardware operating point*, for example, the supply voltage or the clock frequency. The trade-off between supply voltage, clock frequency, and power consumption is well studied in the literature. Another possibility is the use of *low-level hardware techniques*, such as the selective protection of critical parts, quantization, or setting erroneous likelihood values to zero; compare May et al. [2008]. Many algorithms have parameters which can be changed at runtime. Advanced channel decoders, for example, work iteratively. The number of iterations is a parameter which can easily be changed for each individual block by the software. For many components, we have a choice of different algorithms, starting from optimal algorithms with a high complexity down to suboptimal algorithms with a very low complexity, which offers a trade-off between QoS and implementation efficiency. The *choice of parameters and algorithms* is another class of actuators [Brehm et al. 2012]. There also exist resilience actuators on *system level*. Adjusting the communication parameters, such as by choosing a channel code with a better error correction capability, improves the error resilience, but the effects are not immediate. A faster solution is to shift complexity between different components when one of the components has a low hardware reliability. It is important to note that resilience actuators are only activated when hardware errors cause a degradation of the QoS.

In general, different actuators or combinations of actuators are suited to deal with different types of hardware errors. Normally, it is preferable to use actuators which do not require changes inside the components or which can be implemented with low complexity. Each actuator offers a different trade-off between hardware reliability, QoS, and implementation performance (throughput, energy). Based on the channel quality and the respective requirements on QoS, throughput, and energy, the reliability control chooses those actuators which will best fulfill the requirements. Therefore, it is mandatory to characterize each actuator with regard to its influence on communications performance, throughput, area, and energy overhead. Sometimes, the reliability requirements necessitate the use of resilience actuators which have a severe effect, for example, on the system throughput. In these cases, the reliability control also needs actuators which trade off throughput and communications performance. The big advantage of this reliability extension is the dynamic protection of the wireless receiver, which is only activated when necessary.

2.3. An Example

The last section introduced our new methodology in a very general way. The trade-off between channel quality and hardware resilience and the choice of the resilience actuators are application specific and cannot be quantified in a general fashion. However, in this paragraph, we demonstrate our methodology on a concrete example in order to make it more seizable.

The system memories add substantially to the die area of an iterative MIMO-BICM receiver (this will be introduced in Section 3). Memories are very susceptible to hardware errors due to their dense and highly optimized layouts. In Gimmmler-Dumont et al. [2012a], we analyzed the impact of hardware errors in the different system memories on the system performance of a MIMO-BICM system. We found out that, especially the memories containing complex-valued data, namely the channel information and the received vectors, are very sensitive. Figure 4 shows the degradation of the communications performance when errors are injected in the channel information memory. Up to a bit error probability of $p_b = 10^{-6}$ the degradation is negligible for the typical Frame Error Rates (FERs) of a wireless system. Afterwards, the performance decreases gradually with an increasing p_b .

Fig. 4. The system communication performance is gradually decreasing for random bit-flips in the channel information memory. p_b depicts the bit error probability.

Table II. Quantitative Comparison of Resilience Actuators for the System Memories of an Iterative MIMO-BICM Receiver

resilience actuator	impacts	tolerated hardware (un-) reliability	tolerated hardware error probability
algorithmic error resilience	area +0% power +0% throughput -0%	-200mV supply voltage	10^{-6}
1 outer iteration	area +0% power +0% throughput -75%	-300mV supply voltage	$4 \cdot 10^{-5}$
1-bit error correction code	area +30% power +30% throughput -0%	-400mV supply voltage	$8 \cdot 10^{-4}$
8T memory cells	area +25% power +0% throughput -0%	-500mV supply voltage	$8 \cdot 10^{-3}$ for the equivalent 6T memory cells

We assume that the memory errors result from supply voltage drops which occur regularly during power state switching. Several resilience actuators exist which can be applied for different degrees of hardware unreliability in order to mitigate the impact of the hardware errors on the system performance [Kleeberger et al. 2013]. Table II lists them with their influence on area, power consumption, and throughput and their error resilience. In Figure 3(b), the actuators are arranged according to our methodology. No action has to be taken as long as there is a high hardware reliability, that is, voltage drops of no more than 200mV. Within this area, the receiver shows an inherent algorithmic error resilience. For a decreased reliability in which voltage drops up to 300mV occur, we can react on the highest level by increasing the number of iterations in order to regain communications performance. For transient errors, this leads only to a temporary throughput degradation without loss of communications performance. When errors occur with a high probability $p_b > 5 \cdot 10^{-5}$, high-level resilience actuators cannot provide the necessary resilience. On a lower level, the contents of the memory can be protected by a simple 1-bit error correction code. The resilience can be even further increased on technology level by employing 8-transistor (8T) memory cells instead of 6-transistor (6T) cells resulting in a smaller implementation overhead.

8T memory cells can even tolerate voltage drops of 500mV. However, the increase in area and power is in both cases permanent.

3. APPLICATION: DOUBLE-ITERATIVE MIMO-BICM RECEIVER

Multiple-antenna or MIMO systems have the potential to increase the data rate of wireless communication systems. They belong to the most advanced systems in the upcoming 4G and 5G communication standards, and their very high complexity is a challenge for any hardware implementation. Indeed, only few implementations of these iterative receivers exist as of now. To demonstrate our novel methodology, we chose to apply it to a double-iterative MIMO-BICM transmission system.

Cross-layer reliability design requires a deep knowledge of the application under consideration. Therefore, we first introduce the application of our choice and its implementation, before we discuss the results of our new methodology in Section 4.

In the remainder of this Section

- we introduce the system model for MIMO-BICM systems and the algorithms employed in the MIMO detector,
- we present an architecture framework for a flexible, double-iterative MIMO-BICM receiver, which supports different channel codes from the WiMax, WiFi, and LTE standards,
- we present a highly energy-efficient, weakly programmable MIMO detector architecture, and
- we characterize all system components according to their algorithmic flexibility and their algorithmic error resilience.

Energy efficiency and error resilience are strongly correlated problems. Therefore, algorithmic flexibility is a mandatory requirement for error-resilient architectures.

3.1. MIMO-BICM System Model

Typically, a channel code provides redundancy, which allows the correction of transmission errors in the receiver. An interleaver between channel encoder and modulator reduces dependencies between neighboring bits. The modulated symbols are multiplexed to an array of antennas and then transmitted in parallel to increase the data rate. This system setup is called a MIMO-BICM (bit-interleaved coded modulation) system. On the receiver side, a MIMO detector decouples the multiple transmission streams, and the channel decoder corrects errors that have been induced by noise on the communication channel. The most advanced receiver techniques combine the MIMO detector and the channel decoder in an iterative feedback loop to further improve the communications performance of the receiver [Hochwald and ten Brink 2003]. These two blocks exchange likelihood values, which reflect their confidence in the results of their computations. The channel decoder can be iterative itself (and often is), which results in a double-iterative receiver structure. The number of iterations is dynamic and depends strongly on the respective system state and QoS requirements.

Figure 5 shows a model of such a MIMO-BICM system, which is used for the following discussion. The source generates a random infoword $\mathbf{u}$ of length K_c which is encoded by the channel encoder. The interleaved codeword $\mathbf{X}^N$ consists of N_c bits which are linearly grouped into N subblocks $\mathbf{x}_n$.

$$\mathbf{X}^N = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_N) \quad (1)$$

Each subblock $\mathbf{x}_n$ consists of Q coded bits.

$$\mathbf{x}_n = (x_{1,n}, x_{2,n}, \dots, x_{q,n}, \dots, x_{Q,n}), \quad x_{q,n} \in \{-1, +1\} \quad (2)$$

Fig. 5. System model of a MIMO transmission chain using a bit-interleaved coded modulation scheme with iterative MIMO detection and channel decoding in the receiver.

Each $\mathbf{x}_n$ is mapped directly to a complex symbol $s = \text{map}(\mathbf{x}_n)$, chosen from a 2^Q -ary QAM modulation scheme. M_T symbols are combined in one transmission vector $\mathbf{s}_t$, where M_T is the number of transmit antennas.

$$\mathbf{s}_t = (s_{1,t}, s_{2,t}, \dots, s_{m,t}, \dots, s_{M_T,t}) \quad (3)$$

The whole modulated sequence is represented by

$$\mathbf{S}^T = (\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_t, \dots, \mathbf{s}_T). \quad (4)$$

T time slots are needed to transmit all symbols of one codeword. The transmission of vector $\mathbf{s}_t$ in time step t for M_T transmit antennas and M_R receive antennas is modeled by

$$\mathbf{y}_t = \mathbf{H}_t \cdot \mathbf{s}_t + \mathbf{n}_t \quad (5)$$

with $\mathbf{H}_t$ the channel matrix of dimension $M_T \times M_R$ and $\mathbf{n}_t$ the noise vector of dimension M_R , the entries of which are zero-mean and unit variance Gaussian variables. The elements of $\mathbf{H}_t$ are modeled as independent, complex, zero-mean Gaussian random variables. Their real and imaginary parts are independent variables, each with variance $\sigma^2 = N_0/2$. It is assumed that $\mathbf{H}_t$ is ergodic, that is, its entries change independently after each channel use. Furthermore, the MIMO detector is assumed to know $\mathbf{H}_t$, and all employed antenna constellations are symmetric with $M_T = M_R = M$. The received vectors $\mathbf{y}_t$ are subsumed in the matrix $\mathbf{Y}^T$

$$\mathbf{Y}^T = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_t, \dots, \mathbf{y}_T) \quad (6)$$

with

$$\mathbf{y}_t = (y_{1,t}, y_{2,t}, \dots, y_{m,t}, \dots, y_{M_R,t}). \quad (7)$$

Before the decoding starts, the channel preprocessing applies a QR decomposition on $\mathbf{Y}^T$ and $\mathbf{H}_t$. This results in the transformed received vectors $\hat{\mathbf{Y}}^T$ and updated channel matrices $\mathbf{R}_t$. The decoding process iterates over the MIMO detector and the channel decoder, which exchange probability information about the codeword. The soft-in-soft-out MIMO detector determines the likelihood of the bits for each received vector $\hat{\mathbf{y}}_t$ using the a priori information $\mathbf{L}_t^a$ from the channel decoder. Only the extrinsic information $\lambda^e = \lambda - \mathbf{L}^a$ is passed on to the channel decoder.

The channel decoder processes the whole codeword at a time. It uses the interleaved a priori information λ^a from the MIMO detector for the calculation of the estimated information bit sequence $\hat{\mathbf{u}}$ and the a posteriori logarithmic likelihood ratios (LLRs) Λ of the codeword. The extrinsic information $\mathbf{L}^e = \Lambda - \lambda^a$ is returned to the MIMO detector,

thus closing the iterative loop. Channel decoders for advanced channel codes like turbo and LDPC (Low-Density Parity-Check) codes employ belief propagation algorithms. These are iterative algorithms that exchange likelihood values within an iteration. Thus, the complete system is a double-iterative system consisting of an outer loop (MIMO detector, channel decoder) and an inner loop for channel decoding.

3.2. MIMO Detection Algorithm Review

In this section we review the basics of tree-search-based MIMO detection algorithms. The MIMO detector processes the transformed received vectors $\hat{\mathbf{y}}_t$ (see Section 3.1). A received symbol vector $\hat{\mathbf{y}}_t$ can be seen as a weighted superposition of the entries of $\mathbf{s}_t$, disturbed by Gaussian noise. The task of the MIMO detector is the equalization and separation of the originally sent sequence of symbols $\mathbf{s}_t$. The MIMO detector works on one received vector $\hat{\mathbf{y}}_t$ at a time.

For all detection-related explanations, the time indices of $\mathbf{y}$, $\mathbf{H}$, etc., are dropped for ease of notation. Even if not mentioned specifically for each equation, the vectors $\mathbf{s}$ and $\mathbf{x}$ are always the complex representation and the bit representation, respectively, of the same symbol vector. $x_{q,m}$ denotes the q th bit of the m th symbol in $\mathbf{s}$.

For iterative detection and decoding the MIMO detector computes logarithmic likelihood values (LLRs) for each bit.

$$\lambda(x_{q,m}) = \ln \frac{P(x_{q,m} = +1|\mathbf{y})}{P(x_{q,m} = -1|\mathbf{y})} \quad (8)$$

These LLR values can be computed by the Max-Log-Map approximation [Hochwald and ten Brink 2003].

$$\begin{aligned} \lambda(x_{q,m}) \approx & \min_{\forall \mathbf{s} | x_{q,m}=+1} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 - N_0 \sum_{\forall q,m} \ln P(x_{q,m}) \right\} \\ & - \min_{\forall \mathbf{s} | x_{q,m}=-1} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 - N_0 \sum_{\forall q,m} \ln P(x_{q,m}) \right\} \end{aligned} \quad (9)$$

A detailed derivation of Eq. (9) can be found in, for example, Vikalo et al. [2004]. An interpretation for (9) is that the LLR value $\lambda(x_{q,m})$ is derived from the most likely symbol vectors $\mathbf{s}$, with $x_{q,m}$ being +1 or -1, respectively. The metric $d(\mathbf{s})$ is a measure for the likelihood that $\mathbf{s}$ was actually the originally transmitted sequence, given the characteristics of the channel $\mathbf{H}$ and the received sequence $\mathbf{y}$.

$$d(\mathbf{s}) = \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 - N_0 \sum_{\forall q,m} \ln P(x_{q,m}) \quad (10)$$

Small values of $d(\mathbf{s})$ relate to a high probability of $\mathbf{s}$ being sent.

Calculating all possible $d(\mathbf{s})$ to determine (9) quickly becomes infeasible for a larger number of receive and transmit antennas and/or higher-order modulations as the complexity grows with 2^{QM} . Therefore, many suboptimal algorithms with lower complexity were devised [Gimmmler-Dumont et al. 2012b]. Most of them are based on a tree search. In order to map the metric calculations (10) on a tree structure, the channel matrix $\mathbf{H}$ is decomposed into a unitary matrix $\mathbf{Q}$ and an upper-triangular matrix $\mathbf{R}$. The Euclidean distance is rewritten as

$$\|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 = \|\mathbf{y}' - \mathbf{R}\mathbf{s}\|^2 \quad (11)$$

Fig. 6. Detection problem represented by a tree for the modulation alphabet $\{-1, +1\}$ (BPSK) and M_T antennas.

with $\mathbf{y}' = \mathbf{Q}^H \mathbf{y}$. Eq. (10) is replaced by the equivalent metric

$$d(\mathbf{s}) = \|\mathbf{y}' - \mathbf{R}\mathbf{s}\|^2 - N_0 \sum_{\forall q,m} \ln P(x_{q,m}). \quad (12)$$

The triangular structure of $\mathbf{R}$ allows the recursive calculation of $d(\mathbf{s})$

$$d_m = d_{m+1} + \gamma_m(s^{(m)}) \quad (13)$$

with the starting point $d_{M+1} = 0$ and $d(\mathbf{s}) = d_1$. The metric update $\gamma_m(s^{(m)})$ depends on the partial symbol vector $s^{(m)} = (s_m, s_{m+1}, \dots, s_M)$.

$$\gamma_m(s^{(m)}) = \left| y'_m - \sum_{j=m}^M R_{m,j} s_j \right|^2 - N_0 \sum_{q=1}^Q \ln P(x_{q,m}) \quad (14)$$

This equation can be further simplified by introducing the interference-reduced symbol y''_m , which is the same for all children of a node.

$$y''_m = y'_m - \sum_{j=m+1}^M R_{m,j} s_j \quad (15)$$

$$\gamma_m(s^{(m)}) = |y''_m - R_{m,m} s_m|^2 - N_0 \sum_{q=1}^Q \ln P(x_{q,m}) \quad (16)$$

The recursive calculation of Eq. (13) can be represented by a tree with $M + 1$ levels, as shown for the modulation alphabet $\{-1, +1\}$ in Figure 6. The root node corresponds to d_{M+1} and each leaf node corresponds to the metric $d(\mathbf{s})$ of one possible vector $\mathbf{s}$. Each level corresponds to the detection of one symbol s_m . Branches are labeled with an element of the modulation alphabet. When advancing from a parent node to a child node, the child node's metric d_m is calculated from the metric of its parent d_{m+1} and the branch metric γ_m . Sorted QR decomposition [Wubben et al. 2001] and MMSE preprocessing [Wubben et al. 2003] are used as additional techniques to reduce the complexity.

Based on this tree search, many different MIMO detection algorithms exist. The main differences between the algorithms can be described by how they traverse the tree, for example, breadth-first, depth-first, or metric-first techniques, and how branches of the tree are pruned. In general, these algorithms exhibit different communications performances and implementation complexities.

Within this article, we consider two algorithms: MMSE Successive Interference Cancellation (SIC) [Wubben et al. 2001] and soft-input soft-output sphere detection [Hochwald and ten Brink 2003]. These two algorithms lie on opposite ends of the

Fig. 7. Generic architecture framework for an iterative MIMO-BICM receiver including main building blocks and system memories.

ranges of communications performance and throughput. For the MMSE-SIC detection, the tree is traversed only once from top to bottom. On each layer, only the best child node is extended. The hard output MMSE-SIC result equals the sequence of chosen child nodes. In order to obtain a soft output a single-input single-output demapping operation has to be performed for each layer in the tree. MMSE-SIC detection leads to very short processing times and, thus, a very high throughput at the cost of a severely reduced communications performance.

Soft-input soft-output sphere detection is a depth-first search. In the computation of Eq. (9) it considers all symbol vectors $\mathbf{s}$, which lie inside a sphere of radius r around the received vector $\mathbf{y}$, that is, for which $d(\mathbf{s}) < r$. Whenever a partial metric d_i exceeds the sphere radius, the corresponding part of the tree is excluded from the search. The number of processed nodes in the tree is dynamic and depends on the current channel realization. When a large radius is chosen, sphere detection offers near-optimal communications performance at the cost of a low throughput. The throughput can be increased by reducing the radius, which, however, will lead to a degradation of the communications performance.

3.3. System Architecture

In this section we present an architecture framework for iterative MIMO-BICM receivers including implementation results for its components. Multiple-antenna systems are combined with different types of channel codes in the existing standards. WiFi features LDPC codes and convolutional codes, whereas LTE supports only the trellis-based convolutional and turbo codes. WiMax supports all three kinds of channel codes. Therefore, we mapped the iterative receiver structure from Figure 5 onto a general architecture framework which allows us to plug in different MIMO detectors and channel decoders [Gimmmler-Dumont et al. 2012b].

The framework—shown in Figure 7—connects the main building blocks via several system memories. The complex-valued channel information $\mathbf{H}_t$ and $\mathbf{R}_t$ is stored in MAT_H and MAT_R respectively, the complex-valued received information $\mathbf{y}_t$ and $\hat{\mathbf{y}}_t$ in Y_VEC and Y_HAT. DET_IN, DET_OUT, and DEC_IN contain the exchanged LLR values. During the inner iterations of the channel decoder the values in DEC_IN might be updated and the original information is no longer available after decoding. Thus, the a posteriori LLR values λ have to be stored in DET_OUT in order to be able to extract the extrinsic information $\mathbf{L}^a$ for the next iteration of the detector. The size

Table III. Design Overview for Receiver Components

Component	Clock frequency	Area	Throughput	Power
Channel preprocessing	300 MHz	$0.14mm^2$	29-43 Mbit/s	12mW
MIMO detector	300 MHz	$0.21mm^2$	40-480 Mbit/s	15mW
WiMax/WiFi LDPC decoder	385 MHz	$1.17mm^2$	50-300 Mbit/s	389mW
LTE Turbo decoder	300 MHz	$1.52mm^2$	150 Mbit/s	243mW
Channel decoding ASIP	300 MHz	$0.57mm^2$		53mW
turbo decoding mode			21 Mbit/s	
convolutional decoding mode			16 Mbit/s	
System memories LTE	—	$1.37mm^2$	—	—
System memories WiMax/WiFi	—	$0.21mm^2$	—	—

All data for 65nm, after place & route with worst case PVT.

of the memories is determined by the longest codeword size in the LTE standard (18432 bits) and the supported modulation schemes (4-, 16-, and 64-QAM). The area of each memory in a 65nm process technology is shown in Figure 7. The total area of all system memories is $1.37mm^2$.

Table III shows implementation results for all components of the iterative receiver [Nazar et al. 2010; Gimmler-Dumont et al. 2012b]. We include results for different channel decoders to be able to support the different standards with MIMO transmission modes: WiMax, WiFi [Alles 2010], and LTE [May et al. 2010a; Brehm et al. 2011]. The size of the system memories is determined by the largest block length in each communication standard. For example, LTE turbo codes include up to 18432 bits. In this case, the system memories require approximately 40% of the total system area. For WiMax/WiFi, the maximum block length is only 2304 bits which results in a much smaller area for the system memories. The power consumption of the memories is not neglectable when compared to the other components [Gimmler-Dumont et al. 2012b]. However, it depends heavily on the number of inner and outer iterations and is thus not included in Table III.

All designs were synthesized in a 65nm low-power bulk CMOS standard cell library. Target frequency after place-and-route is 300 MHz, which is typical of industrial designs (with the exception of WiMax/WiFi LDPC decoder). State-of-the-art designs consider the following Process, Voltage, and Temperature (PVT) parameters: Worst Case (WC, 1.1V, 125°C), Nominal Case (NOM, 1.2V, 25°C), and Best Case (BC, 1.3V, -40°C). Synthesis was performed with the Synopsis Design Compiler in topographical mode, place-and-route (P&R) with the Synopsys IC Compiler. Synthesis as well as P&R were performed with Worst-Case PVT settings of the 65nm library to create a reference point. Experiments with with Nominal Case have shown that the throughput can be increased by a factor between 1.5 and 2.0 [Gimmler et al. 2012]. Depending on their algorithmic resilience, some parts of the receiver can already be operated in NOM without visible performance loss. The system memories, for example, could be operated with a lower supply voltage without increasing the system's frame error rate (see Table II).

3.4. MIMO Detector Architecture

There exist a wide range of suboptimal algorithms for MIMO detection that allow trade-offs of communications performance versus implementation complexity and energy consumption. Almost all detection algorithms can be mapped to a search in a tree structure, as shown in Section 3.2. In this section, we present an energy-efficient, weakly programmable MIMO detector architecture which can process different

algorithms. Most tree search algorithms can be constructed by only five coarse-grained operations.

- The *enumeration unit* determines the visiting order for the children of one particular node.
- The *interference reduction unit* computes the interference-reduced symbols according to Eq. (15).
- The *metric computation unit* computes the recursive metric according to Eq. (16).
- The *node administration unit* stores all intermediate nodes with their results and chooses the node which is visited next in the tree.
- The *minima administration unit* stores the results of leaf nodes and updates the minima for the calculation of Eq. (9).

We presented an architecture based on these kernels which is able to perform most of the existing algorithms [Gimmmler-Dumont and Wehn 2013]. The complexity of these algorithms covers all classes from linear up to almost exponential complexity, for example, linear Successive Interference Cancellation (SIC), fixed effort detection [Wong et al. 2002; Barbero and Thompson 2008], or sphere detection. As all algorithms are performed with the same algorithmic kernels, the overhead for flexibility is very small. In the literature, mainly highly optimized architectures are presented which perform exactly one algorithm, such as Borlenghi et al. [2011] and Studer et al. [2011]. There exist only a few processor architectures which are able to perform different algorithms. In Jafri et al. [2009], the processor architecture is based on very small-grained operations, that is, complex number operations. In Chen et al. [2012], a processor architecture based on matrix operations is presented which performs maximum ratio combining, linear MMSE detection, and MMSE-SIC detection. However, none of the processor architectures supports tree search algorithms which offer the best communications performance. Thus, they only cover a limited range of the possible channel scenarios.

The implementation results for the detector are shown in Table III. Two examples of detector configurations are illustrated in Figure 8. First, we show how to combine our five operations to perform an MMSE-SIC in Figure 8(a). For this detection algorithm, the tree is only traversed once from top to bottom. After the *interference reduction*, the *enumeration unit* rounds the symbol y_i'' to the closest modulation symbol. The result of this rounding operation is fed back to the *interference reduction* for the next layer in the tree. For hard output MMSE-SIC detection, the detector output equals the results of the *enumeration unit*. For soft output, the *enumeration unit* continues the enumeration of the closest symbols around y_i'' . For all enumerated symbols, the Euclidean distances from the interference-reduced symbol are computed in the *metric calculation unit* and then stored in the *minima administration unit*. MMSE-SIC detection offers a very high constant throughput (compare Table IV) at a reduced communications performance.

The second example in Figure 8(b) shows the configuration for a depth-first or sphere search. Especially for iterative receivers, the soft-input soft-output sphere detector offers the best communications performance. During runtime, throughput can be traded off against communications performance by adjusting the sphere radius. However, due to the nature of the depth-first search, the throughput is dynamic and varies with channel conditions and over the outer iterations. The configuration in Figure 8(b) utilizes all five basic operations. All operations can be performed in parallel if data is available. The *enumeration unit* determines a sequence of the best children of a node. The *metric calculation* computes the recursive metric for each node. The enumeration is stopped when a maximum number is reached or when one of the child nodes violates the radius constraint. Intermediate nodes which fulfill the radius constraint are stored together with their metrics in the *node administration* until their processing is continued. Leaf nodes are stored in the *minima administration*. Whenever possible, the *interference*

Fig. 8. The presented MIMO detector architecture consists of five basic blocks, which can be configured for different detection algorithms.

Table IV. Throughput of Different Detection Algorithms at a Clock Frequency of 300 MHz

detection algorithm	4×4 , 16-QAM	4×4 , 64-QAM
MMSE-SIC hard-output	480 Mbit/s	720 Mbit/s
MMSE-SIC soft-output	390 Mbit/s	600 Mbit/s
soft-input soft-output sphere detection	40-120 Mbit/s	35-120 Mbit/s

reduction unit processes a node from the *node administration* and passes the result to the *enumeration* unit. This recursive loop continues until all nodes within the sphere have been computed. In contrast to other depth-first sphere decoders (e.g., Burg et al. [2005] and Witte et al. [2010]) which employ a one-node-per-cycle architecture, the presented architecture computes two nodes per cycle. This is a novel approach that doubles the throughput compared to state-of-the-art implementations. The throughput results for the different algorithms are summarized in Table IV. Apart from these two types of algorithms, the detector architecture can also process breadth-first search algorithms, such as Barbero and Thompson [2008].

In contrast to the existing approaches, this is the first weakly programmable MIMO detector architecture which offers just the necessary flexibility. In this way, the detection algorithm can be chosen and parameterized at runtime according to the current channel conditions and QoS requirements. This approach leads to a highly energy-efficient implementation. The flexibility can also be used to improve the error resilience of an iterative MIMO-BICM receiver, as we will demonstrate in Section 4.

3.5. Classification of Components

In this section, we discuss characteristics of all components regarding their (algorithmic) error resilience and their flexibility. Where available, we also mention previous

results. This classification serves as input for the identification of application-specific resilience actuators in the following section.

3.5.1. Channel Preprocessor. The channel preprocessor performs a QR decomposition and a matrix-vector multiplication. The circuit is highly dataflow oriented and has a limited flexibility only: The QR decomposition of the channel matrix can be done for 2×2 or 4×4 matrices, and the number of detection vectors that are processed before the channel matrix is updated can be adjusted. Our architecture, which was presented in Gimmmler-Dumont et al. [2013], runs a sorted QR decomposition with MMSE preprocessing for a 4×4 channel matrix in 167 clock cycles. The channel preprocessing is only performed once, even if several outer iterations between detector and decoder are done. An error which occurs in this unit is therefore propagated through all outer iterations.

For the QR decomposition, we chose the modified Gram-Schmidt process [Golub and Van Loan 1996] due to its simplicity and stability when working with finite precision values. Apart from this stability, the algorithm exhibits no error resilience. From an algorithmic point of view, the only solution for error mitigation is therefore recomputation in case of hardware errors. As due to throughput constraints the recomputation might not be feasible, the channel preprocessing has to be protected mainly by low-level techniques. An analysis of low-level protection of the channel preprocessing is out of scope of this article but will be addressed in a future study.

3.5.2. Channel Decoder. Channel decoders correct errors that were induced by noise on the communication channel. They are usually very flexible in order to support different code rates and block lengths. The most advanced decoders work iteratively by exchanging likelihood values. These iterations necessitate large memories and a large memory bandwidth, making the designs memory dominated. Channel decoders have an inherent error resilience in consistence with their primary purpose: error detection and correction. Several studies show that channel decoders can correct hardware errors up to a certain degree, such that on a system level no effects are visible. It was found that the most efficient technique to improve the error resilience is to adjust the number of decoder iterations. Important results of these publications have been summarized in Section 1.2.

3.5.3. MIMO Detector. The task of the MIMO detector is to separate and demap the individually transmitted symbol streams. The architecture from Section 3.4 processes antenna constellations up to 4×4 and QAM modulation schemes up to 64-QAM. Furthermore, its configurability allows it to execute different detection algorithms ranging from high-throughput suboptimal to near-optimal algorithms. In contrast to channel decoding, MIMO detection has no implicit error correction capability. However, its algorithmic flexibility can be exploited for trade-offs which adapt the error resilience on system level. We will discuss resilience actuators for MIMO detection and their influence on the overall system in Section 4.

4. IDENTIFICATION OF RESILIENCE ACTUATORS

This section demonstrates our novel methodology (see Section 2) on the state-of-the-art double-iterative MIMO-BICM receiver which we presented in the previous section. Section 4.1 is dedicated to the identification of resilience actuators for MIMO detection. Error resilience techniques for channel decoding have already been thoroughly investigated (compare Sections 3.5.2 and 1.2). However, there are potential trade-offs on system level between MIMO detection and channel decoding, which we will discuss in Section 4.2. Except for the hardware operating point, we will restrict ourselves to application-specific resilience actuators. Universal, already established low-level hardware techniques can be applied to any application and result in a constant overhead. In

this section we focus on dynamic resilience techniques, which can be switched on and off as necessary and which do not have a large impact on implementation complexity and energy consumption.

4.1. Resilience Actuators for Soft-Input Soft-Output MIMO Detection

This section presents resilience actuators for MIMO detection. As MIMO detectors have no inherent error correction capability, algorithmic changes inside the detector component cannot improve the error resilience. This has to be done either on system level or by changing the hardware operating point. These actions usually have a negative influence on system throughput and/or communications performance. Therefore, we also introduce algorithmic resilience actuators, enabling a trade-off of throughput and communications performance in order to counterbalance these effects.

4.1.1. Hardware Operating Point. When timing errors occur, the clock frequency can be reduced or the supply voltage can be increased to make the circuit faster. However, both approaches require additional control circuits and energy. The trade-off between supply voltage and energy is well understood. The number of bit-flips in a memory, for example, strongly depends on the voltage: Increasing the supply voltage decreases the soft error rate. According to Dixit and Wood [2011], the soft error rate drops by about 30% when the operating voltage is increased by 100 mV compared to the nominal voltage. Changing the hardware operating point offers a trade-off between reliability and energy consumption, which is often used for voltage overscaling. In the Introduction, we cited several publications in which voltage overscaling is employed for achieving an energy reduction.

4.1.2. Adjustment of Detection Quality. Changing the detection quality offers a trade-off between communications performance and throughput but has no direct influence on the error resilience. However, a higher throughput augments the available time budget and thus offers a higher potential for error resilience. In the following paragraphs, we will describe which options we have to influence the detection quality.

There exist a variety of MIMO detection algorithms with different degrees of suboptimality regarding communications performance and detection complexity. We introduced two corner cases already in Section 3.2: soft-input soft-output sphere detection and MMSE-SIC detection. The first has a very high communications performance at a low throughput, whereas these characteristics are reversed for MMSE detection.

The soft-input soft-output MIMO detection algorithm introduced in Section 3.2 offers two control parameters, which allow us to trade-off detection quality for throughput. The detection algorithm performs a tree search in which a likelihood metric (12) for each possible transmission vector is computed recursively. After each recursion step, that is, in each layer of the tree, the metric is compared to a given radius. The computation of metrics which exceed this radius is aborted and all following branches of the tree are pruned. Increasing this radius leads to a more precise detection as more transmission vectors are considered in the final LLR computation. At the same time the computational complexity is increased because more metrics have to be calculated. Thus, the radius is a dynamic control parameter to trade off communications performance for throughput.

The tree search is executed depth first, where the child nodes of each valid node have a visiting order which is determined by an enumeration method. This order ensures that child nodes which are likely to result in better metrics will be visited first. Once the metric of a child node exceeds the radius, the remaining child nodes are discarded as well, because they, too, will likely lie outside of the radius. Normally, only a small part of child nodes is visited during the search. In order to simplify the enumeration process in the hardware implementation, we restricted the maximum number of enumerated child

Fig. 9. Our MIMO detector architecture allows us to dynamically change the detection quality at runtime, offering large trade-offs between communications performance and throughput.

nodes to ten, which does not lead to a degradation of the detection quality. However, the maximum number of enumerated children can be further decreased, resulting again in a trade-off between communications performance and throughput.

A third possibility to change the detection quality is the use of a suboptimal algorithm, such as MMSE-SIC detection. In contrast to sphere detection, MMSE detection has a constant throughput at a low detection complexity, but the communications performance is also reduced. The architecture we presented in Section 3.4 is weakly programmable. It can be configured at runtime to execute different detection algorithms, for example, hard- and soft-output MMSE-SIC detection, or soft-input soft-output sphere detection.

The resulting trade-off for a 4×4 antennas, 16-QAM system employing a 64-state convolutional code is shown in Figure 9. The throughputs are compared with regard to a fixed communications performance of a Frame Error Rate (FER) of 10^{-2} . The achievable SNR range spreads over more than 7dB while the throughput varies between 40Mbit/s and 480Mbit/s. This resilience actuator uses the available algorithmic flexibility and thus has only a negligible influence on power and area consumption.

4.1.3. External LLR Manipulations. In this paragraph, we describe methods on hardware level which can dynamically increase the error resilience of the MIMO detector. Instead of accessing the MIMO detector directly, we propose low-complexity techniques that work only on the LLR-input and -output values of the detector. LLR values have a high robustness against hardware errors. They represent likelihood values for information bit decisions. A negative value represents a logical “1” and a positive value represents a logical “0”. The absolute value is a measure for the reliability of this decision. If an LLR value is equal to zero, it contains no information. Thus, the most important information is stored in the sign bit. As long as this sign bit is not compromised, the core information is still correct and the channel decoder can correct the hardware errors. These characteristics offer two further possibilities to reduce the impact of hardware errors without intervening with the interior of the detector component.

Channel preprocessing and MIMO detection work on the basis of MIMO vectors, not codewords. When a hardware error is detected in the processing chain of a MIMO

vector, we can treat the according LLR values at the output of the MIMO detector as zero, which represents an equal likelihood for 0 or 1. This prevents at least the propagation of the errors. This technique is known from channel decoders, and was presented for an LDPC decoder in May et al. [2008]. If the rate of hardware errors is not too high, the channel decoder can recover the lost information. Furthermore, this technique reduces the average power consumption of the MIMO detector. When, for example, an error occurs within the channel preprocessor, we do not process this MIMO vector in the detector, but instead set the resulting LLR values to zero. This energy reduction might then be spent to improve the performance of the channel decoder, for example, by increasing the number of decoder iterations.

During feedback iterations, we can employ a kind of vector-based iteration control on the input LLR values [Zhang et al. 2010]. The LLR values are a measure for the likelihood of the bit decisions. If the absolute value of all LLR values for one MIMO vector lies above a threshold, we can assume that this vector is already correctly decoded and use it as output for the detector without doing an actual detection. In Zhang et al. [2010], a complexity reduction in the MIMO detector of approximately 25% was achieved without loss of communications performance. This technique is most effective when errors occur in the detector, because the number of detections is reduced to a minimum and the throughput is increased.

The additional complexity for the realization of these low-level techniques is negligible compared to the area and power consumption of the complete detector. Our detector architecture instantiates approximately 80 comparators, 80 adders, and 3 multipliers for the data path (without controller and precomputation of certain values). The additional operations for a vector based iteration control (for example, one adder and one comparator) thus result in a very low overhead. Setting values to zero requires even less hardware. In this way, the average power consumption can be decreased while the communications performance stays almost identical.

4.2. Resilience Actuators on System Level

Instead of increasing the reliability of components individually, the problem can also be tackled on system level. The double-iterative structure of a MIMO-BICM receiver offers several high-level possibilities to combat the unreliability of its components. We present the most promising techniques in the remainder of this section.

4.2.1. Iteration Control Mechanisms. An iteration control typically monitors exchanged values in an iterative system and checks stopping conditions to detect the convergence of the processed block. In Gimmler-Dumont et al. [2012a] we analyzed the impact of memory errors on the system behavior of an iterative MIMO system. We observed that errors in any of the memories before the MIMO detector have an increased impact on the communications performance if the incorrect values are processed repeatedly during the outer iterations. Furthermore, the hardware errors caused by radiation accumulate over time. Therefore, it is always preferable to reduce the decoding time of a block.

An iteration control stops the iterative decoding as early as possible. It may detect correctly decoded blocks as well as blocks which cannot be decoded at all, thus minimizing decoding costs in terms of energy and throughput. There exist a number of low-complexity stopping criteria for iterative MIMO systems [Gimmler et al. 2010]. They are based on thresholds that balance the false alarm rates and false detection rates. These thresholds can be adjusted dynamically at runtime allowing, for example, a higher number of blocks to be stopped if hardware errors have occurred. Employing iteration control on system level increases the average throughput but—depending

Table V. Algorithmic Complexity for Different Iteration Control Criteria
[Gimmmler et al. 2010]

Criterion	Operations per information bit and iteration
Sign-Change Ratio	2x XOR, 2x ADD
Mean-Reliability	2x ADD
Re-Encoding	6x XOR, 2x XOR, 2x ADD
Mutual information	2x LUT, 2x ADD
Convolutional Decoder	398x ADD, 266x COMPARE

In comparison to channel decoding, the additional effort for iteration control is negligible.

Fig. 10. Example for complexity shifting in a double-iterative MIMO-BICM receiver by varying the number of outer loop and channel decoder iterations.

on the choice of the threshold values—does not necessarily have an influence on the communications performance.

In Gimmmler et al. [2010], it was possible to reduce the number of outer iterations to an average below 2 (from a maximum of 10) without sacrificing communications performance. A further throughput increase is possible by allowing a degradation of communications performance. The additional effort for an iteration control is very low compared to channel decoding (compare Table V).

4.2.2. Complexity Shifting between Components. An example for a global algorithmic adaptation is to shift the complexity between system components: When a building block cannot compensate an error locally, the system convergence can still be achieved by increasing the computational effort of other building blocks. Such a shift can be achieved, for instance, between the channel decoder and the MIMO detector, leveraging the outer feedback loop. When the MIMO detector is not able to counterbalance the impact of hardware errors, the number of channel decoder iterations and/or the number of outer loop iterations can be increased in order to maintain the communications performance.

Figure 10 shows the frame error rate for a 4×4 antennas, 16-QAM system employing a WiMax-like LDPC code where complexity shifting can be used. We compare the frame error rate for different numbers of decoder iterations and outer iterations. Let

Fig. 11. Implementation efficiency of MIMO detector, WiMax/WiFi LDPC decoder, and two system configurations using the efficiency metrics from Kienle et al. [2011].

us consider the case when the receiver is performing 3 outer iterations and 5 LDPC iterations. When the MIMO detector suffers from hardware errors, for example, due to a temperature increase, we can temporarily shift more processing to the LDPC decoder by performing only 2 outer iterations and 20 LDPC iterations. The new configuration provides the same communications performance.

The question is how such a shift changes the energy efficiency of the MIMO receiver. Figure 11 shows the implementation efficiency of a MIMO-BICM receiver and its components. The red curve shows the efficiency of the MIMO detector for different search radii and in the MMSE-SIC configuration. The blue curve shows the efficiency of an LDPC decoder running with different number of iterations. The LDPC decoder is a flexible decoder which supports all code rates and code lengths from WiMax and WiFi standard. The yellow and the green curves show the system efficiency for different numbers of outer iterations with 5 LDPC iterations and 20 LDPC iterations respectively.

With the help of this graph, we can quantify the influence of a complexity shift: when changing from 3 outer and 5 LDPC iterations to 2 outer and 20 LDPC iterations, the energy efficiency of the system is reduced by approximately 50%. However, the same communications performance is achieved and when the temperature in the MIMO detector decreases, the reliability control can return to the original configuration.

4.2.3. Shifting of Error Correction Capability between Components. Typically, MIMO detector and channel decoder are designed and implemented independently of each other. The MIMO transmission scheme provides a large data rate but has no error correction abilities. The error correction capability is solely provided by the channel code to improve the error rate performance of the transmission system. From a system point of view, the MIMO detector does not work on completely independent data as there are dependencies from the overlaying channel code. However, this diversity cannot be exploited by the detector as the channel interleaver hides the code structure from the detector and because most channel code constraints span over many MIMO detection vectors.

Kienle [2008] introduced a small block code in each MIMO detection vector in order to generate a small diversity gain in the MIMO detector while simplifying the outer channel code to keep the overall coding rate constant. While this approach targeted the decoding complexity, it can also be used to increase the error resilience of the MIMO detector. Each parity check in one MIMO vector improves the error correction capabilities of the detector. On a system level, the diversity gain can be split between detector and decoder dynamically, thus allowing the system to react dynamically to changing hardware error rates. The only drawback of this approach is that the diversity separation has to be done on the transmitter side, which causes a high delay.

5. CONCLUSIONS

Technology scaling is leading to a point where traditional worst-case design is no longer feasible. In this article, we presented a new methodology for the design of dependable wireless systems. We combined cross-layer reliability techniques to treat hardware errors with the least possible overhead leading to a high energy efficiency. Application-specific resilience actuators together with low-level techniques offer the ability to respond to the changing requirements on reliability and quality-of-service. We illustrated our new methodology on a state-of-the-art double-iterative MIMO-BICM receiver which belongs to the most complex systems in modern communication standards.

We identified dynamic resilience actuators on all layers of abstraction. Each actuator offers a trade-off between communications performance, implementation performance (throughput, power), and error resilience. Any actuator which trades off communications performance for throughput, for example, the sphere radius, can be reused to increase the error resilience, when combined with a reduction of the clock frequency. Throughput and error resilience are thus closely related. As we have shown, algorithmic resilience actuators offer a great potential for dynamic trade-offs between communications performance, implementation performance, and error resilience. To the best of our knowledge, this publication shows for the first time the strong mutual dependencies between these three design metrics in a wireless receiver.

ACKNOWLEDGMENTS

The authors thank Christian Brehm and Matthias May for many insightful discussions.

REFERENCES

- R. A. Abdallah and N. R. Shanbhag. 2009. Error-resilient low-power viterbi decoder architectures. *IEEE Trans. Signal Process.* 57, 12, 4906–4917.
- M. Alles. 2010. Implementation aspects of advanced channel decoding. Ph.D. dissertation. University of Kaiserslautern.
- L. G. Barbero and J. S. Thompson. 2008. Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems. *IEEE Trans. Vehic. Technol.* 57, 5, 2804–2814.
- J. Bau, Q. Jacobson, R. Hankins, B. Saha, A. Tabatabai, and S. Mitra. 2007. Error resilient system architecture (ERSA) for probabilistic applications. In *Proceedings of the IEEE International Workshop on Silicon Errors in Logic – System Effects*.
- F. Borlenghi, E. M. Witte, G. Ascheid, H. Meyr, and A. Burg. 2011. A 772mbit/s 8.81bit/nj 90nm CMOS soft-input softoutput sphere decoder. In *Proceedings of the IEEE Asian Solid State Circuits Conference (A-SSCC'11)*. 297–300.
- C. Brehm, T. Ilseher, and N. When. 2011. A scalable multi-ASIP architecture for standard compliant trellis decoding. In *Proceedings of the International SoC Design Conference (ISOC'11)*. 349–352.
- C. Brehm, M. May, C. Gimmmler, and N. When. 2012. A case study on error resilient architectures for wireless communication. In *Proceedings of the 25th International Conference on Architecture of Computing Systems (ARCS'12)*. 13–24.

- A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei. 2005. VLSI implementation of MIMO detection using the sphere decoding algorithm. *IEEE J. Solid-State Circ.* 40, 7, 1566–1577.
- L. N. Chakrapani, P. Korkmaz, B. E. S. Akgul, and K. V. Palem. 2007. Probabilistic system-on-a-chip architectures. *ACM Trans. Des. Autom. Electron. Syst.* 12, 3, 1–28.
- X. Chen, A. Minwegen, Y. Hassan, D. Kammmler, S. Li, T. Kempf, A. Chattopadhyay, and G. Ascheid. 2012. FLEXDET: Flexible, efficient multi-mode MIMO detection using reconfigurable ASIP. In *Proceedings of the 20th Annual IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM'12)*. 69–76.
- V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. 2010. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC'10)*. 555–560.
- DAC Roundtable. 2010. Designing chips without guarantees. *IEEE Des. Test Comput.* 27, 5, 60–67.
- S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw. 2009. RazorII: In situ error detection and correction for PVT and SER tolerance. *IEEE J. Solid-State Circ.* 44, 1, 32–48.
- A. Dixit and A. Wood. 2011. The impact of new technology on soft error rates. In *Proceedings of the IEEE International Reliability Physics Symposium (IRPS'11)*.
- A. Ejlali, B. M. Al-Hashimi, P. Rosinger, and S. G. Miremadi. 2007. Joint consideration of fault-tolerance, energy-efficiency and performance in on-chip networks. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'07)*. 1–6.
- M. A. Elgamel and M. A. Bayoumi. 2003. Interconnect noise analysis and optimization in deep submicron technology. *IEEE Circ. Syst. Mag.* 3, 4, 6–17.
- A. M. Eltawil and F. J. Kurdahi. 2006. System redundancy: A means of improving process variation yield degradation in memory arrays. In *Proceedings of the International Symposium on VLSI Design, Automation and Test*. 1–4.
- D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. 2003. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proceedings of the 36th International Symposium on Microarchitecture*. 7–18.
- J. Geldmacher, K. Hueske, and J. Gotze. 2011. Turbo equalization for receivers with unreliable buffer memory. In *Proceedings of the IEEE Vehicular Technology Conference (VTC'11)*. 1–5.
- J. George, B. Marr, B. E. S. Akgul, and K. V. Palem. 2006. Probabilistic arithmetic and energy efficient embedded signal processing. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES'06)*. 158–168.
- V. Gherman, S. Evain, M. Cartron, N. Seymour, and Y. Bonhomme. 2009. System-level hardware-based protection of memories against soft-errors. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'09)*. 1222–1225.
- C. Gimmmler, F. Kienle, C. Weis, N. Wehn, and M. Alles. 2012. ASIC design of a gbit/s LDPC decoder for iterative MIMO systems. In *Proceedings of the International Computing, Networking and Communications Conference (ICNC'12)*. 192–197.
- C. Gimmmler, T. Lehnigk-Emden, and N. Wehn. 2010. Low-complexity iteration control for MIMO-BICM systems. In *Proceedings of the 21st IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'10)*.
- C. Gimmmler-Dumont, C. Brehm, and N. When. 2012a. Reliability study on system memories of an iterative MIMO-BICM system. In *Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration*.
- C. Gimmmler-Dumont, F. Kienle, B. Wu, and G. Masera. 2012b. A system view on iterative MIMO detection: Dynamic sphere detection versus fixed effort list detection. *VLSI Des. J. Special issue Flexible Radio Des.: Trends Challenges Digital Baseband Implement*, 2.
- C. Gimmmler-Dumont, P. Schlaefel, and N. When. 2013. ASIC implementation of a modified QR decomposition for tree search based MIMO detection. In *Proceedings of the 4th IEEE Latin American Symposium on Circuits and Systems*.
- C. Gimmmler-Dumont and N. When. 2013. An energy-efficient weakly programmable MIMO detector architecture. *Adv. Radio Sci.* 11, 131–136.
- G. H. Golub and C. F. Van Loan. 1996. *Matrix Computations* 3rd Ed. The Johns Hopkins University Press.
- A. Herkersdorf, V. Kleeburger, S. Nassif, U. Schlichtmann, C. Weis, and N. When. 2013. Cross-layer dependability modeling and abstraction in systems on chip. In *Proceedings of the 9th IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE'13)*.
- B. M. Hochwald and S. Ten Brink. 2003. Achieving near-capacity on a multiple-antenna channel. *IEEE Trans. Comm.* 51, 389–399.

- A. M. A. Hussien, M. S. Khairy, A. Khajeh, K. Amiri, A. M. Eltawil, and F. J. Kurdahi. 2010. A combined channel and hardware noise resilient viterbi decoder. In *Proceedings of the Record of the 44th Asilomar Conference on Signals, Systems and Computers (ASILOMAR'10)*. 395–399.
- A. R. Jafri, D. Karakolah, A. Baghdadi, and M. Jezequel. 2009. ASIP-based flexible MMSE-IC linear equalizer for MIMO turbo-equalization applications. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'09)*. 1620–1625.
- M. S. Kairy, C.-A. Shen, A. M. Eltawil, and F. Kurdahi. 2012. Error resilient imo detector for memory-dominated wireless communication systems. In *Proceedings of the IEEE Global Communications Conference (GlobeCom'12)*.
- G. Karakonstantis, N. Banerjee, K. Roy, and C. Chakrabarti. 2007. Design methodology to trade off power, output quality and error resiliency: Application to color interpolation filtering. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'07)*. 199–204.
- G. Karakonstantis, G. Panagopoulos, and K. Roy. 2010. HERQULES: System level cross-layer design exploration for efficient energy-quality trade-offs. In *Proceedings of the ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED'10)*. 117–122.
- G. Karakonstantis, C. Roth, C. Benkeser, and A. Burg. 2012. On the exploitation of the inherent error resilience of wireless systems under unreliable silicon. In *Proceedings of the IEEE Design Automation Conference (DAC'12)*.
- A. Khajeh, M. Kim, N. Dutt, A. M. Eltawil, and F. J. Kurdahi. 2008. Cross-layer co-exploration of exploiting error resilience for video over wireless applications. In *Proceedings of the IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia (ESTImedia'08)*. 13–18.
- A. Khajeh, M. Kim, N. Dutt, A. M. Eltawil, and F. J. Kurdahi. 2012. Error-aware algorithm/architecture co-exploration for video over wireless applications. *ACM Trans. Embed. Comput. Syst.* 11S, 1.
- F. Kienle. 2008. Low-density MIMO codes. In *Proceedings of the 5th International Symposium on Turbo Codes and Related Topics*. 107–112.
- F. Kienle, N. Wehn, and H. Meyr. 2011. On complexity, energy- and implementation-efficiency of channel decoders. *IEEE Trans. Comm.* 59, 12, 3301–3310.
- V. Kleeberger, C. Gimmmler-Dumont, C. Weis, A. Herkersdorf, D. Mueller Gritschneider, S. Nassif, U. Schlichtmann, and N. When. 2013. A cross-layer technology-based study of the impact of memory errors on system resilience. *IEEE Micro* 33, 4, 46–55.
- S. Krishnamohan and N. R. Mahapatra. 2004. A highly-efficient technique for reducing soft errors in static CMOS circuits. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'04)*. 126–131.
- L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra. 2010. ERSA: Error resilient system architecture for probabilistic applications. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'10)*. 1560–1565.
- Y. Liu, T. Zhang, and J. Hu. 2009. Design of voltage overscaled low-power trellis decoders in presence of process variations. *IEEE Trans. VLSI Syst.* 17, 3, 439–443.
- M. May, M. Alles, and N. When. 2008. A case study in reliability-aware design: A resilient LDPC code decoder. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'08)*. 456–461.
- M. May, T. Ilseher, N. Wehn, and W. Raab. 2010a. A 150mbit/s 3GPP LTE turbo code decoder. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'10)*. 1420–1425.
- M. May, N. Wehn, A. Bouajila, J. Zeppenfeld, W. Stechele, A. Herkersdorf, D. Ziener, and J. Teich. 2010b. A rapid prototyping system for error-resilient multi-processor systems-on-chip. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'10)*. 375–380.
- S. Mitra, K. Brelsford, Y. M. Kim, H.-H. K. Lee, and Y. Li. 2011. Robust system design to overcome CMOS reliability challenges. *IEEE J. Emerg. Selected Topics Circ. Syst.* 1, 1, 30–41.
- S. Mitra, M. Zhang, N. Seifert, T. M. Mak, and K. S. Kim. 2007. Built-in soft error resilience for robust system design. In *Proceedings of the IEEE International Conference on Integrated Circuit Design and Technology (ICICDT'07)*. 1–6.
- D. Mohapatra, G. Karakonstantis, and K. Roy. 2009. Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator. In *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'09)*. ACM Press, New York, 195–200.
- S. Murali, T. Theodorides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli. 2005. Analysis of error recovery schemes for networks on chips. *IEEE Des. Test Comput.* 22, 5, 434–442.
- G. L. Nazar, C. Gimmmler, and N. When. 2010. Implementation comparisons of the QR decomposition for MIMO detection. In *Proceedings of the 23rd Symposium on Integrated Circuits and System Design (SBCCI'10)*. ACM Press, New York, 210–214.

- M. Nicolaidis. 1999. Time redundancy based soft-error tolerance to rescue nanometer technologies. In *Proceedings of the 17th IEEE VLSI Test Symposium*. 86–94.
- C. Novak, C. Studer, A. Burg, and G. Matz. 2010. The effect of unreliable LLR storage on the performance of MIMO-BICM. In *Proceedings of the Record of the 44th Asilomar Conference on Signals, Systems and Computers (ASILOMAR'10)*. 736–740.
- K. Nowka, S. Nassif, and K. Agarwal. 2008. Characterization and design for variability and reliability. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC'08)*. 341–346.
- K. V. Palem. 2005. Energy aware computing through probabilistic switching: A study of limits. *IEEE Trans. Comput.* 54, 9, 1123–1137.
- W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. 2011. An architecture for fault-tolerant computation with stochastic logic. *IEEE Trans. Comput.* 60, 1, 93–105.
- N. Shanbhag. 2002. Reliable and energy-efficient digital signal processing. In *Proceedings of the 39th Design Automation Conference (DAC'02)*. 830–835.
- N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones. 2010. Stochastic computation. In *Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC'10)*. 859–864.
- C. Studer, S. Fateh, and D. Seethaler. 2011. ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation. *IEEE J. Solid-State Circ.* 46, 7, 1754–1765.
- Third Generation Partnership Project. 2008. 3GPP TS 36.212 V8.5.0; 3rd generation partnership project; Technical specification group radio access network; Evolved universal terrestrial radio access (E-UTRA); Multiplexing and channel coding (release 8). www.3gpp.org.
- H. Vikalo, B. Hassibi, and T. Kailath. 2004. Iterative decoding for MIMO channels via modified sphere decoding. *IEEE Trans. Wirel. Comm.* 3, 6, 2299–2311.
- E. M. Witte, F. Borlenghi, G. Ascheid, R. Leupers, and H. Meyr. 2010. A scalable VLSI architecture for soft-input soft-output single tree-search sphere decoding. *IEEE Trans. Circ. Syst. II: Express Briefs* 57, 9, 706–710.
- K. W. Wong, C. Y. Tsui, R. S. K. Cheng, and W. H. Mow. 2002. A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'02)*, vol. 3.
- W. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer. 2003. MMSE extension of V-blast based on sorted QR decomposition. In *Proceedings of the 58th IEEE Vehicular Technology Conference (VTC'03)*, vol. 1. 508–512.
- D. Wubben, R. Bohnke, J. Rinas, V. Kuhn, and K. D. Kammeyer. 2001. Efficient algorithm for decoding layered space-time codes. *Electron. Lett.* 37, 22, 1348–1350.
- D. Zhang, I.-W. Lai, K. Nikitopoulos, and G. Ascheid. 2010. Informed message update for iterative MIMO demapping and turbo decoding. In *Proceedings of the International Symposium on Information Theory and its Applications (ISITA'10)*. 873–878.

Received August 2012; revised May 2013; accepted November 2013