A Novel MPSoC Interface and Control Architecture for Multi-Standard RF Transceivers

Siegfried Brandstätter, and Mario Huemer, *Senior Member, IEEE*

***Abstract*—The introduction of new mobile communication standards, enabling the ever growing amount of data transmitted in mobile communication networks, continuously increases the complexity of control processing within radio frequency (RF) transceivers. Since this complexity cannot be handled by traditional approaches, this article focuses on the partitioning of RF transceiver systems and on the implementation of application-specific components to introduce an advanced multiprocessor system-on-chip (MPSoC) interface and control architecture which is able to fulfill the requirements of future RF transceiver integrations. The proposed framework demonstrates a high degree of scalability, flexibility, and reusability. Consequently, the time to market for products can be reduced and fast adaptations to the requirements of the market are feasible. Moreover, the developed application-specific components achieve improved or at least equivalent performance results compared to common architectures while the silicon area can be reduced. This characteristic has positive effects on the costs as well as on the power consumption of the RF transceiver.**

***Index Terms*—RF transceiver, hard real-time controlling, MPSoC, on-chip synchronization, on-chip communication, application-specific processor design, application-specific NoC design.**

I. Introduction

THIS article focuses on the design of an advanced multiprocessor system-on-chip (MPSoC) interface and control architecture for multi-standard radio frequency (RF) transceivers. Certain technical aspects of this architecture have already been published in [1], [2], and [3]. While the focus of these publications is limited to the technical aspects themselves, the following elaboration presents a general view on the topic and describes in detail the interaction of these technical aspects. Additionally, the work provides a survey of modern multi-standard RF transceivers which describes the features and the internal control tasks. This survey allows to clarify the requirements of future interface and control architectures.

A. Motivation

For a long time the development of RF transceivers was associated with pure analog design. Nowadays, this statement is not true anymore. RF transceivers have become complex mixed signal systems-on-chip (SoCs) including analog signal processing, digital signal processing, and control processing. The digital interface and control architecture of an RF transceiver, which mainly involves the control processing, is used to control the internal and external, analog and digital modules of the device.

Since the beginning of RF transceiver integration, the complexity of controlling these modules increased continuously. Table I shows this trend on the basis of various facts. Starting in 2003, the effort for implementing an interface and control architecture was quite low because almost all devices only supported a single mobile communication standard. In the following years companies tried to integrate multiple mobile communication standards and the support of multiple frequency bands in a single-chip integrated circuit (IC) [4]. Moreover, a lot of tasks, which were originally processed within the baseband IC, were moved to the RF transceiver. For the shown device in column two, which supports the Global System for Mobile Communications (GSM) and Universal Mobile Telecommunications System (UMTS) standards, the complexity of controlling increased tremendously compared to the device in column one. Due to the increasing size of the digital domain within RF transceivers, the next development step introduced smaller technology nodes at the cost of more complexity in the digital domain to enhance the analog signal processing. For the shown devices in column three and four, the complementary metal-oxide-semiconductor (CMOS) C11RF technology node was replaced by the CMOS C65LP technology node. In 2011, a new generation of RF transceivers was able to additionally support the Long Term Evolution (LTE) standard [5], the successor of the UMTS standard. Again, a huge increase in complexity can be recognized.

B. Evolution of Concepts and State-of-the-Art

1) Flatten Logic based Architectural Approach: At the beginning of RF transceiver integration for GSM systems, the interface and control architectures were kept very simple since the devices only integrated the analog parts of the receive and transmit paths. As stated in [1], in such system architectures the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) are placed on the baseband IC or on separate ICs. Hence, the signal data interface is usually represented by a multiplexed differential analog I/Q interface which is shared between the receive and transmit signals. Additionally, a serial digital interface and two dedicated signals are used to control the RF transceiver, to enable the system clock generation, and to provide the system clock to the baseband.

S. Brandstätter is with DMCE GmbH & Co KG, 4040 Linz, Austria, Email: SiegfriedX.Brandstaetter@intel.com

M. Huemer is with the Institute of Signal Processing, Johannes Kepler University Linz, 4040 Linz, Austria, Email: Mario.Huemer@ieee.org

This work has been supported by the Austrian COMET-K2 program of the Linz Center of Mechatronics (LCM), and was funded by the Austrian federal government and the federal state of Upper Austria.

Manuscript received July XX, 2014; revised XXX.

TABLE I
COMPLEXITY INCREASE OF RF TRANSCEIVER INTEGRATION.

Year	**2003**	**2007**	**2009**	**2011**
Supported Standard	GSM	GSM, UMTS	GSM, UMTS	GSM, UMTS, LTE
Technology Node	CMOS C11RF	CMOS C11RF	CMOS C65LP	CMOS C65LP
Human Resources[a]	0.5 Persons	11 Persons	21 Persons	30 Persons
Total Size of Digital Domain	11 kGate	490 kGate	780 kGate	1000 kGate
Memory Size	-	257 kBit	1030 kBit	2196 kBit
Die Size	6 mm²	20 mm²	18 mm²	36 mm²

[a] Number of persons involved in developing the interface and control architecture over the project period.

The interface and control architecture of the RF transceiver itself includes a register set which is connected to the serial digital control interface. Considering that the configuration registers within this set are directly programmed by the baseband, this kind of controlling is called register based programming model. The configuration registers act as input for the flatten control logic which generates the control signals for the analog blocks of the receive and transmit paths. In most cases, the implementation of the control logic is very simple consisting of direct links between the configuration registers and the control signals, hardwired logic, and delay mechanisms. Since shared resources, such as bus systems, are not used in this kind of architecture, events can be triggered exactly by accessing the configuration registers over the serial digital control interface.

But of course, this simplicity has several drawbacks. Control signals may be derived from combinations of other control signals. Therefore, side effects and cross impacts have to be considered carefully, even for small changes in the system. These interdependences of control signals also increase the effort of verification. Furthermore, the partitioning of the design and implementation work into smaller work packages appears to be quite difficult.

2) Hierarchical Modular based Architectural Approach: While the differential analog I/Q data interface became an industrial standard, digital data interfaces were emerging in parallel [1]. According to [6], these digital data interfaces provide development benefits for both systems, the baseband and the RF transceiver. The performance of the RF transceiver can be increased by supporting the implementation of analog and digital filters. Furthermore, the integration of digital modulators, such as the polar modulator, requires a digital data interface to gain an optimum performance. The baseband, on the other hand, can be partitioned to exclude all analog functions which work at high frequencies, e.g. the ADCs which sample the I/Q data signals. Due to the fact that digital functions are more independent from technology nodes than analog ones, the baseband can be upgraded to new technology nodes with less effort.

Since the interconnection between the baseband and RF transceiver ICs should be generic and independent from architectural decisions, the DigRF v1.12 digital interface [7] was standardized. This interconnection includes two serial digital interfaces. One is shared between the receive and transmit data by multiplexing the interface signals, and the other one is used to control the RF transceiver. Two additional signals are needed to enable the system clock generation and to provide the system clock to the baseband. Moreover, the interface provides a strobe signal which can be used by the baseband to exactly trigger events within the RF transceiver.

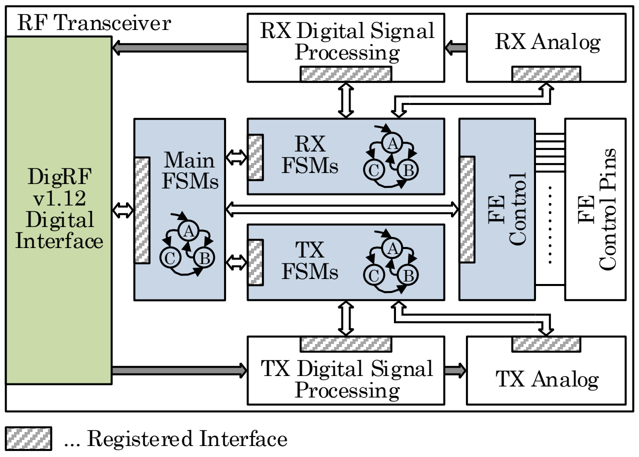

Fig. 1. Hierarchical modular based architectural approach for RF transceivers.

In the following hierarchical modular based architectural approach not only the ADC and the DAC but also the digital signal processing of the receive and transmit paths is placed on the RF transceiver IC. Furthermore, the control of the external analog front-end (FE), including the RF switch, the power amplifier (PA), and the low-noise amplifier (LNA), is integrated, too. As a result, the device is now responsible for the timing on the air interface. This leads to additional requirements for the interface and control architecture concerning real-time. According to the 3rd Generation Partnership Project (3GPP) specification [8], the maximum tolerance on the timing of a GSM transmit burst is ±1 symbol period. Due to the fact that this budget must be arranged over the whole signal path, including the baseband, the interface, and the RF transceiver, customers typically specify a timing tolerance of ±1/4 symbol periods (±0.92 µs) for the RF transceiver.

Since this interface and control architecture not only controls the receive and transmit paths and the external analog FE but also processes calibration and compensation algorithms at startup, a flatten logic based architectural approach would be far too ineffective. Thus, a partitioning into smaller work packages is required. This “divide and conquer” approach is also reflected in the interface and control architecture which can be seen in Figure 1. The control tasks are assigned to four major modules: the main finite-state machines (FSMs) block, the receiver (RX) and transmitter (TX) FSMs blocks, and the FE control block. The interconnections between these blocks, including the digital signal processing and analog modules, are implemented as registered interfaces. This means that the higher hierarchical layer writes the control information to the register sets of the lower hierarchical layer which processes this information. In order to save silicon area, these register sets are typically accessed over a bus system which is shared by several FSMs. In case of multiple masters accessing the bus system at the same time, this fact can lead to delays and consequently to timing violations on the air interface.

The main advantage of this interface and control architecture is the encapsulation of the blocks which reduces the risks of side effects and cross impacts. Moreover, the hierarchical approach affords the implementation of generic receive and transmit paths with the main requirement of fulfilling the mobile communication standard. Subsequently, the higher hierarchical layers can be tailored to different customer requirements.

3) High-Level Programming based Architectural Approach: The introduction of UMTS led to the demand for a new digital data interface which is capable of handling the arising data rates [1]. However, the specification of the new interface not only accounted higher data rates but also the upcoming diversity approach in the RX, the multi-standard RF transceiver integration, and the demand for less interconnections between the baseband and RF transceiver ICs.

As result of these requirements, the DigRF v3.09 digital interface [9] was standardized. The two differential digital paths of this interconnection, one for transmitting and one for receiving, are capable of handling various prioritized logical channels for configuration, timing control, and data. Like for the other interfaces, two additional signals are required to enable the system clock generation and to provide the system clock to the baseband. In addition to the register based programming model, this interface provides a high-level programming model which is used to optimize the performance for multi-standard use case scenarios and to simplify the configuration.

In order to meet the requirements of the UMTS standard, the tolerances for the timing specification on the air interface are tightened. According to the 3GPP specification of UMTS [10], the maximum timing error on the initial transmission shall be less than or equal to 1.5 chip periods. This budget must be arranged over the whole signal path and therefore, customers usually specify a timing tolerance of $\pm 1/4$ chip periods (± 65 ns) for the RF transceiver.

The interface and control architecture, which is used to fulfill the requirements and possibilities of the new standardized interface, is very similar to the one described in Section I-B2 with the only difference of an additional module named macro decoder. This block represents the higher hierarchical layer which is introduced to support high-level programming. The macro decoder decodes the configuration macros, which are sent over the DigRF v3.09 digital interface to the RF transceiver, and generates a list of registers including their values. These precalculated values are written to the configuration registers of the main FSMs block on a cycle accurate basis after receiving a time accurate strobe (TAS) macro. The functionality of the remaining blocks stays the same, but special attention has to be paid to the adapted timing tolerance on the air interface when using a shared bus system to access the registers of lower hierarchical layers.

II. Survey of Modern Multi-Standard RF Transceivers

A. Future Trends

As stated in Section I-A, plenty of RF transceiver innovations have been introduced in the past decades. Nowadays, the rapidly growing mobile communication market even calls for more advanced concepts in order to be prepared for the future. The following sections address four important scopes of RF transceiver design and describe their future trends.

1) System Partitioning: When talking about system partitioning, it is important to distinguish between the physical partitioning and the functional partitioning of mobile communication platforms. Since the beginning of RF transceiver integration, companies try to reduce the number of external components needed to implement a mobile communication platform by arranging them in ICs. This approach reduces the costs and the required printed circuit board (PCB) area and enhances the performance of the platform. Considering their relation to RF functionality, many of the former external devices are integrated into the RF transceiver IC.

The functional partitioning of mobile communication platforms is mostly associated with the partitioning of tasks between the baseband and the RF transceiver. In the course of time, plenty of tasks, which were originally processed within the baseband, were moved to the RF transceiver. There is a general tendency to encapsulate all functionalities, which are related to RF and the air interface, in the RF transceiver. In this case, the baseband can simply request the transmission or reception of a data stream for a mobile communication standard on a predefined frequency band and channel.

2) Connectivity: These days almost all RF transceivers for mobile communication are able to handle the GSM and UMTS standards. Upcoming integrations also support the current releases of the LTE standard. But in general, mobile devices, such as mobile phones and laptop computers, demand further connectivity besides mobile communication. They acquire wireless local area network (WLAN) connections, and receive digital terrestrial television, digital audio broadcasts, and global positioning system (GPS) satellite information. Furthermore, Bluetooth connections and near field communication (NFC) are required.

As stated in [11], current techniques make use of separate platforms with its own RF transceiver and baseband to support these standards. In order to overcome the resulting disadvantages, future techniques require to share and encapsulate functionalities across platforms. This not only concerns the RF transceiver and baseband ICs but also the external analog FEs and the antennas.

3) Receive and Transmit Architectures: One of the most important choices to be made is the architecture of the receive and transmit paths within the RF transceiver. In the past decades, plenty of studies and examples have been presented on heterodyne, homodyne, low intermediate frequency (IF), wideband IF, and other architectures. All of them have certain benefits and drawbacks for a specific application [12].

Nowadays, almost all modern RF receivers for mobile multi-standard and multi-band communication make use of direct-conversion architectures. These architectures offer the highest potential for flexibility but also incorporate a number of issues which are not, or the least, less relevant in heterodyne architectures. In order to overcome these issues, additional algorithms are required to implement direct-conversion architectures. Further concepts, which have been emerging in the recent years, are the polar modulator and the polar transmitter. These TX concepts provide a very promising power added efficiency (PAE) by processing the amplitude and phase information in two separated paths.

4) Technology Node: These days, almost all RF transceivers for mobile communication are integrated using CMOS technology. This effort is motivated by low costs and a significant capacity for on-chip integration. In order to obtain higher packing densities, higher circuit speeds, and a lower power dissipation, the ever increasing size of the digital domain even gives rise to the usage of smaller technology nodes for integrating RF transceivers. While this trend shows great benefits for the digital domain, it implicates significant drawbacks for the analog design. The analog domain suffers from less analog voltage headroom, signal-to-noise ratio (SNR), available power, and dynamic range [13]. These challenges need to be addressed by technology enhancement, circuit optimization, and architectural adaption as well as by digitally assisted analog design.

B. Internal Control Tasks

After presenting the future trends of modern multi-standard RF transceivers, the following sections characterize the internal control tasks which are implicated in this evolution. Due to the different nature of the processed algorithms and techniques, the internal control tasks demand different real-time and processing performance requirements from the architecture. A useful classification, which considers the time of execution, specifies three different groups of tasks.

1) Preconfiguration Tasks: The preconfiguration tasks are triggered by the baseband to prepare the reconfiguration of the operational mode within the RF transceiver. The processing scheme in Figure 2 demonstrates that the device has to be enabled first. After a short wake-up phase the baseband sends a configuration macro to the RF transceiver. The reception of this macro activates the preconfiguration tasks which decode the configuration macro and precalculate the required parameters for the modules of the device. Basically, the preconfiguration tasks only precalculate and store these parameters but do not apply any changes to the operational settings of the RF transceiver.

The real-time requirements in terms of timing jitter, which are demanded from the interface and control architecture to process the preconfiguration tasks, are not excessively high. This is due to the fact that these tasks do not access the registers which are related to the operational settings of the RF transceiver. On the other hand, the worst-case execution times (WCETs) of the preconfiguration tasks must fulfill certain limits. Therefore, the interface and control architecture must provide an adequate processing performance for these tasks.

2) Reconfiguration Tasks: After the RF transceiver has finished its preconfiguration, the baseband is allowed to activate the reconfiguration tasks. As depicted in Figure 2, these tasks are triggered by the baseband sending a TAS macro. The main function of the triggered tasks is the reconfiguration of the operational mode by applying the precalculated and stored parameters to the internal and external, analog and digital modules of the RF transceiver. Additionally, the tasks are responsible for performing certain calibration and compensation algorithms.

Changing the operational settings of the RF transceiver directly influences the activities on the air interface. As stated in Section I-B, the activities of certain mobile communication standards define very tight tolerances for the timing specification on the air interface. In order to fulfill these specifications, the real-time requirements for processing the reconfiguration tasks are very high. Most of them even require cycle accurate accesses to the register sets with a very low startup jitter. Another challenge arises from the fact that several of the tasks may be processed in parallel. Considering that the parameters for the reconfiguration of the operational mode are precalculated and stored, the processing performance requirements of the tasks are not excessively high. Only the calibration and compensation algorithms demand a certain processing performance to enable a fast startup of the modules.

3) Run-Time Tasks: As soon as the interface and control architecture has applied the precalculated and stored parameters to the modules of the RF transceiver, the system is configured for a certain operational mode. During the operation of this mode, the architecture has to perform numerous run-time tasks. In general, these tasks are started and processed periodically by the interface and control architecture itself. In this context, relevant examples are measurement tasks, tasks for maintaining the operational mode, and tasks for compensating unwanted effects which may change over time.

The processing scheme in Figure 2 demonstrates that the run-time tasks apply changes to the operational settings of the RF transceiver which directly influence the activities on the air interface. Therefore, the real-time requirements for processing the run-time tasks are as high as the real-time requirements for processing the reconfiguration tasks. Since various run-time tasks may be processed concurrently, the architecture must provide additional means for a parallel execution of tasks. Certain run-time tasks may embed algorithms with a high computational complexity. Thus, the interface and control architecture of the RF transceiver has to provide an adequate performance for processing these tasks.

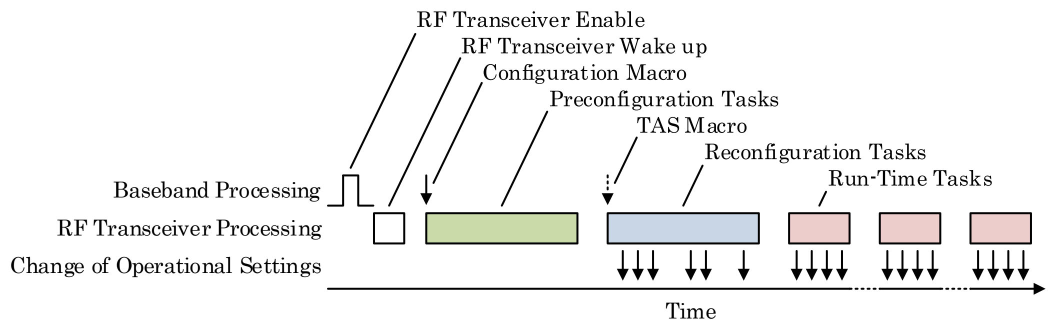

Fig. 2. Processing scheme of the internal control tasks.

C. Conceptual Requirements for Interface and Control Architectures

The continuous enhancements of modern multi-standard RF transceivers have a great influence on the conceptual requirements for future interface and control architectures of these devices. Furthermore, the time to market for wireless products is getting shorter and shorter since the enhancements and the improvements of mobile communication systems progress at an enormous speed. The interface and control architectures described in Section I-B have the common disadvantage that their reusability and flexibility are very restricted. Hence, late adaptations to the mobile communication standard or failures frequently result in huge costs for the implementation, the verification, and potential new tape-outs.

Considering these circumstances, the conceptual requirements for future interface and control architectures are obvious:

- Introduce scalability for future RF transceiver integrations,
- Maximize the flexibility and the reusability,
- Speed up the time to market,
- Minimize the power consumption and the silicon area effort,
- Allow different customer requirements,
- Introduce a universal debug and trace concept, and
- Allow an RF transceiver integration within the baseband IC.

III. A Novel Architectural Approach

A. Partitioning

In order to cope with the increasing complexity of the control processing, future interface and control architectures have to provide considerably more processing performance but also flexibility for late adaptations to the mobile communication standard as well as for modifications in case of failures. According to Pollacks Rule [14], using a centralized controlling concept with a single processing element involves severe disadvantages because the performance increase by enhancing a programmable microarchitecture is roughly proportional to the square root of the increase in complexity.

In order to overcome this barrier, future interface and control architectures have to implement a distributed controlling concept [1]. This means that the internal control tasks are distributed to several processing elements. In this context, the novel architectural approach defines units which not only include the required analog and digital signal processing blocks but also elements which are used for control processing. Since the internal control tasks are encapsulated and processed within the unit they are associated with, the RF transceiver can be partitioned into stand-alone units which are capable of controlling themselves after providing certain parameters.

This approach introduces another essential advantage. The partitioning allows to power-down all units of the RF transceiver which are not required for its current operation. This enables a great potential for saving power, especially for RF transceiver designs which incorporate a high number of units. For example, an RF transceiver implementing a multiple-input multiple-output (MIMO) architecture can separately power-down its RX and TX units.

Basically, the advanced partition generates a framework for future interface and control architectures which is able to fulfill the stated requirements. Figure 3 depicts this framework which is reduced to the interface unit, the RX unit, and the TX unit. The interface unit is a particular unit since it represents the gateway to the device. Thus, the unit not only embeds elements of the interface between the baseband and the RF transceiver, but it also incorporates blocks which are required for the overall control processing of the system. The RX and TX units, in contrast, incorporate elements which are only related to their functionality. In general, an RF transceiver incorporates a larger number of units than displayed in this framework. Examples are several RX and TX units for MIMO architectures, several RX units for receive diversity, additional RX and TX units for connectivity besides mobile communication, and units for controlling the external analog FE. In terms of control processing, all these units are similar to the presented RX and TX units, and are therefore omitted in this diagram.

In general, a distributed controlling concept involves various challenges which include the processing of the internal control tasks, the on-chip synchronization, and the on-chip communication. The following sections address these challenges and describe the respective blocks in the developed framework.

B. Processing Elements

Programmable processing elements are the key for gaining the flexibility required in future interface and control architectures. This flexibility allows modifications of the control

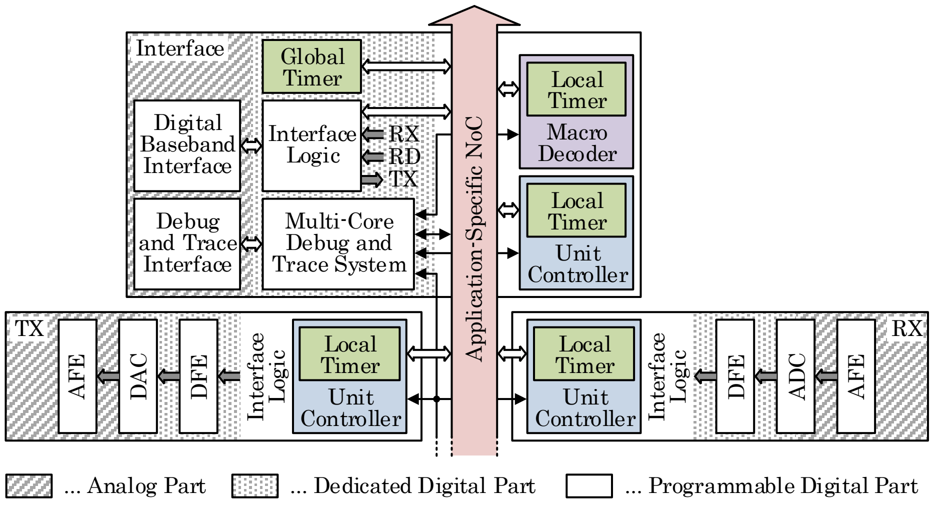

Fig. 3. Framework of the novel architectural approach reduced to the interface unit, the RX unit, and the TX unit.

processing for late adaptations to the mobile communication standard, the corrections of failures, or the implementation of different customer requirements. Considering the real-time and processing performance requirements, it can be seen that the three groups of internal control tasks demand two different requirement sets.

While the real-time requirements for processing the preconfiguration tasks are not excessively high, most of the reconfiguration and run-time tasks demand cycle accurate accesses to the register sets with a very low startup jitter. Moreover, the processing elements must provide means for a parallel execution of reconfiguration and run-time tasks. With respect to the processing performance, the WCETs of the preconfiguration tasks are the crucial factor. In order to fulfill these two sets of requirements, the novel architectural approach introduces two different processing elements.

1) Unit Controller: The unit controller is dedicated to the processing of the reconfiguration and run-time tasks. In order to encapsulate these tasks within the unit they are associated with, each unit embeds a unit controller which is used to process the respective tasks. Using this concept, the units are capable of controlling themselves after providing the essential parameters.

As depicted in Figure 3, the interface unit also embeds a debug and trace interface, and a multi-core debug and trace system to fulfill the conceptional requirement of a universal debug and trace concept. These concepts are nowadays widely researched and used in MPSoCs [15]. Thus, the multi-core debug and trace system is taken as a given and is not further discussed in this work.

The unit controller, in contrast, has specific requirements for processing the reconfiguration and run-time tasks. Typically, these requirements are fulfilled by implementing separate elements to handle the cycle accurate accesses to the register sets and the parallelism of the tasks. But using this approach involves two drawbacks.

- Due to the communication between the elements the overall execution time and hence the power consumption is increased.
- In case of processing several parallel and independent internal control tasks the handling of the available resources is very complex.

For these reasons, we propose an application-specific core which enables a cycle accurate execution of parallel tasks on a single resource [2]. Using this architecture, the computation of the internal control tasks and the cycle accurate accesses to the register set need not be separated. More details about this application-specific core are given in Section V.

2) Macro Decoder: The macro decoder is dedicated to the processing of the preconfiguration tasks which involve the decoding of the configuration macros and the precalculation of the required parameters. Since these tasks are related to the high-level programming model of the baseband interface, the processing element is embedded in the interface unit of the RF transceiver. Figure 3 demonstrates that the macro decoder is also linked to the multi-core debug and trace system.

Similar to the unit controller, the macro decoder has to be programmable in order to provide the needed flexibility. But the preconfiguration tasks demand less real-time capabilities in terms of timing jitter than the reconfiguration and run-time tasks do. On the other hand, a high processing performance is favored to speed up the preconfiguration phase of the RF transceiver. Considering these requirements, an off-the-shelf general purpose processor can be used for processing the preconfiguration tasks. This solution avoids huge costs for designing, implementing, and verifying a customized architecture. In order to complete the design of the novel architectural approach, a master student evaluated and verified a well known open source processor called OpenRISC1200 [16]. The results of this work can be found in [17].

C. On-Chip Synchronization

Even though the novel architectural approach is based on stand-alone units which are capable of controlling themselves after providing certain parameters, several associated reconfiguration and run-time tasks may be distributed to different unit controllers. Due to these dependencies and the fact that changes to the operational settings demand high real-time requirements in terms of timing jitter, a synchronization mechanism between the tasks is required.

A well known approach, which is often used in real-time critical applications, is the introduction of a common time base [18]. As illustrated in Figure 3, the framework makes uses of global and local timers to introduce this feature. In general, the interface unit is the first unit to be powered up when the RF transceiver is enabled. Therefore, the global timer, which acts as the master by defining the common time base, is located in this unit. The local timers, which represent the local references, are directly embedded within the macro decoder and the unit controllers. Since these processing elements make use of the local timers, the distributed internal control tasks can be synchronized to the common time base. Due to the fact that the clocking concept of the targeted architecture makes use of a single clock source, a continuous synchronization of the local timers to the global timer is not required. The synchronization process must only be triggered whenever a unit is powered up. In this case, the baseband or the interface unit within the RF transceiver signals the global timer to broadcast the common time base using the on-chip communication system. The local timers of the enabled units make use of the received common time base to synchronize their local references. Detailed information about distributing the common time base using the on-chip communication system are provided in the following sections.

D. On-Chip Communication

A fundamental challenge of designing a distributed controlling concept is the on-chip communication. As stated in the previous section, the novel architectural approach targets an implementation of stand-alone units which are capable of controlling themselves. Nevertheless, in advance to the activation of the reconfiguration tasks certain parameters must be provided to the units and the TAS information has to be distributed for triggering the reconfiguration tasks. Furthermore, various associated reconfiguration and run-time tasks may need to exchange certain measurement values and control parameters between each other. Moreover, the communication system must provide means to synchronize the local timers to the global timer.

As shown in Figure 3, the framework implements an application-specific network-on-chip (NoC) to facilitate the required on-chip communication. The main motivation for utilizing a NoC structure is the increasing number of units, which have to be interconnected within the proposed framework, paired with the different types of required communication services. Of course, various communication systems, e.g. bus systems, may provide these features, but previous developments have shown that these systems are more complex, less flexible, less structured, and less scalable. As a consequence, they are more prone to errors in case of modifications. In general, the classification of the on-chip communication within the interface and control architecture is a key factor to identify the requirements and restrictions for the application-specific NoC. On closer examination it can be seen that the novel architectural approach requires four different communication services to implement a fully functional framework. These four services are presented in Table II.

The TAS distribution service is required to distribute the TAS information to the unit controllers for triggering the reconfiguration tasks. Due to the high real-time requirements in terms of timing jitter demanded by the reconfiguration tasks, the highest priority is assigned to this service. The time distribution service, having the second highest priority, is required to synchronize the local timers to the global timer. Like the TAS distribution service, the time distribution service requires a constant packet latency and is only used for broadcasting by a single source.

Another service, called messaging service, provides broadcast and point-to-point communication to the processing elements, the interface logic, and the global timer. This service is required by the reconfiguration and run-time tasks to exchange information during their operation. The cycle accurate synchronization between these tasks is assured by a time stamp based communication. In order to separate the on-chip communication of the preconfiguration tasks, a further communication service, called random read/write access service, is introduced to distribute the precalculated parameters to the unit controllers after decoding the configuration macro. Furthermore, the baseband interface is allowed to use this service for debugging purposes.

When looking at the requirements and restrictions of these services, it appears that using a common NoC architecture, which provides quality of service (QoS), is likely the best approach. But as stated in [3], a common network based architecture of the NoC does not lead to an optimum result. Hence, we propose an application-specific approach which is tailored to the specific requirements and restrictions of the interface and control architecture. A description of this application-specific NoC is given in the following section.

IV. Application-Specific Network-on-Chip

A. Concept

1) Design Decisions: As stated in the previous section, the requirements and restrictions of the needed communication services obviously target an implementation of a common NoC architecture which provides QoS [19]. Most of these QoS based NoCs support differentiated services and make use of virtual channels to implement the required priority based scheduling and allocation policies. But a common NoC architecture, which provides QoS, shows several disadvantages.

- The implementation of virtual channels multiplies the number of input and output buffers. Therefore, introducing QoS has a great impact on the silicon area as well as on the power consumption of the design.
- A QoS based NoC cannot guarantee a constant packet latency which is needed to broadcast the TAS and time information. Hence, additional hardware structures are required to assure a constant latency from sending to receiving the packets.
- Common NoC architectures experience a rather high packet latency which is typically reduced by operating at

TABLE II
SPECIFICATION OF THE COMMUNICATION SERVICES.

Communication Service	Priority	Broadcast	Point-to-Point	Number of Sources
TAS Distribution	0	$*$		1
Time Distribution	1	$*$		1
Messaging	2	$*$	$*$	$N_{\text{UnitController}}$ + 3
Random Read/Write Access	3		$*$	2

higher clock frequencies [20]. But due to the frequency restrictions of the interface and control architecture, a larger number of wait states has to be expected especially for random read accesses.

In order to overcome these drawbacks, an application-specific NoC is implemented for the novel architectural approach. Based on the partitioning of the RF transceiver into stand-alone units which include one or several communication clients, e.g. a unit controller or a macro decoder, using a fat-tree based topology for the NoC is likely the best approach. In general, the minimum number of routers can be achieved by implementing a MinRoot architecture [21].

When looking at the specified communication services in Table II, it appears that various requirements and restrictions can lead to a reduction of the complexity within the NoC. The following are the key factors.

- The TAS and time distribution services only require a broadcast mechanism.
- The TAS and time distribution services are only used by a single source each.
- The random read/write access service is only used by two sources.

Since the TAS and time distribution services are only used by a single source, a special packet injection point at the root of the fat-tree based topology is chosen for these services. By applying this technique called "root injection", the paths from the leafs to the root of the system can be eliminated. This leads to a lower packet latency and a saving of silicon area. Considering that only a broadcast mechanism is required, two dedicated distribution systems are used in parallel to the NoC instead of implementing QoS for these services. Even though there is an additional logic and routing overhead, which consumes silicon area, the expected area effort is less compared to using QoS.

A further enhancement belongs to the random read/write access service. The bus enhanced network-on-chip (BENoC) architecture [22] makes use of a bus system which operates in parallel to a common NoC. The bus structure concurrently functions as a low latency broadcast and multicast capable medium which is primarily used for short latency signaling and multicast services. The analysis shows that a bus enhanced NoC is the optimum solution for the novel architectural approach, too. Of course, the bus structure can only be implemented efficiently because there are only two sources for the random read/write access service, the macro decoder and the interface logic. These sources are the only masters on the shared bus system.

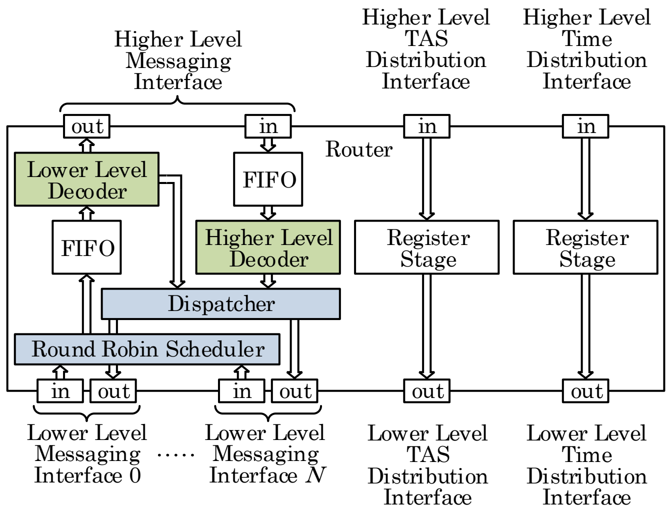
Higher Level Messaging Interface
Higher Level TAS Distribution Interface
Higher Level Time Distribution Interface
out
in
in
in
Router
Lower Level Decoder
FIFO
FIFO
Higher Level Decoder
Register Stage
Register Stage
Dispatcher
Round Robin Scheduler
in
out
in
out
out
out
Lower Level Messaging Interface 0
Lower Level Messaging Interface N
Lower Level TAS Distribution Interface
Lower Level Time Distribution Interface

Fig. 5. Router microarchitecture of the bus enhanced MinRoot NoC with root injection.

Figure 4 demonstrates the final result of the optimization process: the bus enhanced MinRoot NoC with root injection. In this example diagram the application-specific NoC connects 13 unit controllers, the macro decoder, the interface logic, and the global timer.

2) Architecture: The bus system of the application-specific NoC implements the basic features of the WISHBONE Revision B4 standard [23] which are extended by the optional features of data selection, pipelining, and locking. Due to the fact that the design includes two layers, the bus system is capable of accepting simultaneous slave requests from multiple masters. In the event of two masters requesting the same slave, a priority arbitration scheme is applied. Furthermore, the architecture does not imply any register stages and therefore, requests are transferred immediately.

As described in the previous section, the network architecture of the application-specific NoC implements dedicated physical channels for the TAS distribution service, the time distribution service, and the messaging service. Figure 5 depicts the microarchitecture of a router which is located below the root of the MinRoot topology. Since the TAS and time distribution services are only used for broadcasting, the dedicated channels do not require a complex switching technique or routing algorithm. Only register stages are implemented between the higher and lower level interfaces to avoid long combinational paths.

The switching technique and the routing algorithm of the messaging service, in contrast, require more sophisticated mechanisms. Basically, this service makes use of wormhole

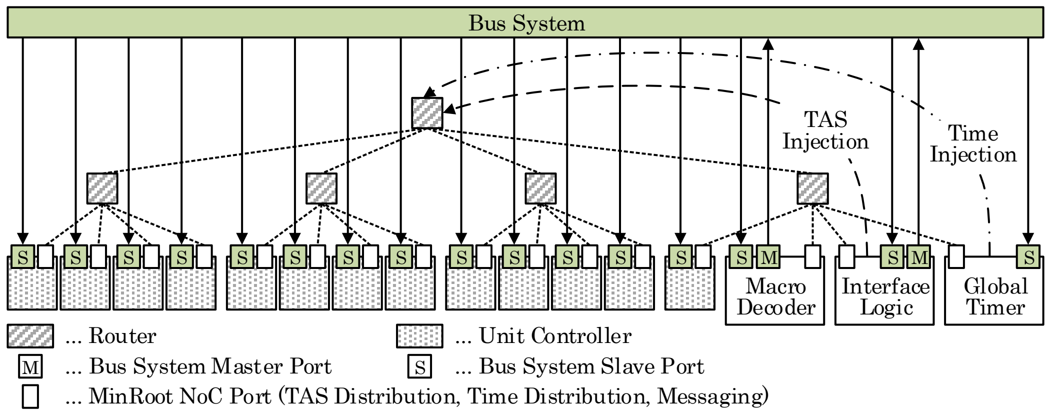
Bus System
TAS Injection
Time Injection
S
M
Macro Decoder
Interface Logic
Global Timer
... Router
... Unit Controller
... Bus System Master Port
... Bus System Slave Port
... MinRoot NoC Port (TAS Distribution, Time Distribution, Messaging)

Fig. 4. Bus enhanced MinRoot NoC with root injection.

switching. In order to accept packets from multiple routers or communication clients located at the lower level of the topology, the router microarchitecture implements a round robin scheduler which forwards the packets to the respective first-in first-out (FIFO). The actual routing decisions are taken within the lower and higher level decoders by analyzing the destination address which is stored in the first flow control unit (flit), the header flit, of every packet. Broadcast packets and packets not belonging to the subnet are always routed to the higher level messaging interface of the router. The other packets, which belong to the subnet, are forwarded to the dispatcher with additional information about the requested lower level messaging interface. In general, this routing algorithm describes a static, distributed, and minimal behavior. The dispatcher, implementing two layers, is able to forward one packet from the higher level decoder and one from the lower level decoder concurrently. Certainly, this is only possible if the packets have different requested lower level messaging interfaces. In case of two packets having the same requested lower level messaging interface, the packet from the higher level decoder is forwarded first.

B. Benchmarking

1) Environment: As stated in Section IV-A1, the obvious solution for providing the communication services required within the proposed framework would be implementing a QoS based NoC. Hence, the reference NoC, which is used to benchmark the application-specific NoC, utilizes a native QoS approach.

In general, the reference implementation features the same MinRoot architecture like the application-specific NoC. Thus, the number of routers in the system is the same. But compared to the optimized NoC the native one neither implements root injection nor a bus enhancement. In fact, the reference NoC makes use of priority based scheduling and allocation policies to enable the prioritized communication of the required services. Like most NoCs which provide differentiated services, the design implements these policies by applying virtual channels.

Basically, the environment for benchmarking both systems is based on the setup depicted in Figure 4. Additionally, the following considerations are used to operate the NoCs.

- The bus system of the application-specific NoC features an address port width of 32 Bit and a data port width of 16 Bit.
- The router interconnections of both NoCs are set to 16 Bit. The only exception is the router interconnection of the time distribution interface used in the application-specific NoC which is only one bit wide.
- The size of the lower and higher level FIFOs is set to 2 flit and 6 flit, respectively.
- The size of the packets injected to the messaging interface, the TAS distribution interface, and the time distribution interface of the application-specific NoC is 6 flit (96 Bit), 2 flit (32 Bit), and 20 flit (20 Bit), accordingly. The QoS based NoC accepts packets with a size of 6 flit (96 Bit).
- The destination addresses of the injected packets and the destination addresses of the random read/write accesses are uniformly distributed. This means that the probability of addressing a particular communication client is equal for all of them.

The clock frequency of both systems is set to 104 MHz. This is also the target clock frequency for synthesizing the NoCs using the CMOS C65LP technology library which specifies a 65 nm technology especially designed for low power and RF design.

2) Packet Throughput: Figure 6 demonstrates the system throughput capabilities of the application-specific NoC and the QoS based NoC. In these diagrams the horizontal axis depicts the bit rate, which the simulation environment tries to inject, and the vertical axis shows the bit rate which is accepted by the NoC.

The system throughput capabilities of the application-specific NoC show that the bit rate of the time distribution service is limited to approximately 100 MBit/s. This is due to the fact that the time distribution service is distributed over a single serial wire. The TAS distribution service, the random read/write access service, and the messaging service provide a maximum bit rate of approximately 1.7 GBit/s, approximately 1.8 GBit/s, and approximately 2.1 GBit/s, respectively. The overall system throughput, which equals the sum of through-

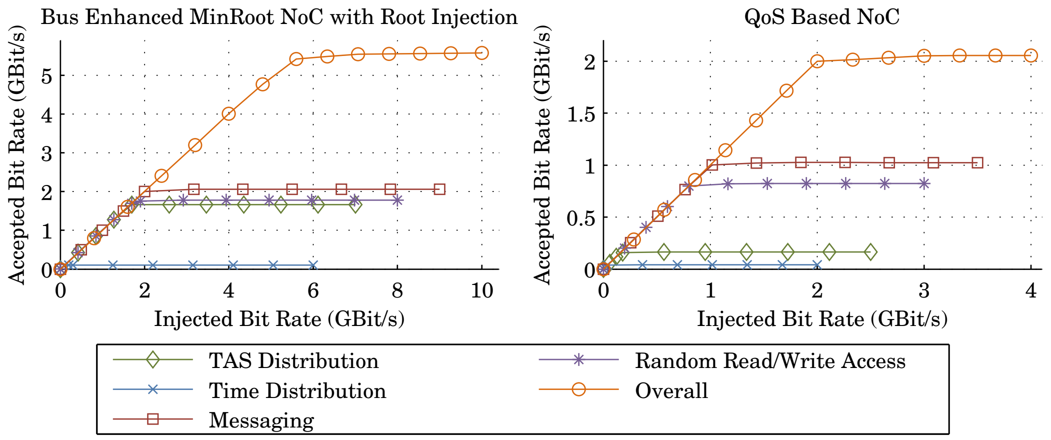

Fig. 6. Throughput capabilities of the bus enhanced MinRoot NoC with root injection and the QoS based NoC.

puts provided by the particular communication services, has a saturation point of approximately 5.5 GBit/s.

The system throughput capabilities of the QoS based NoC depict that the overall bit rate has a saturation point of approximately 2.1 GBit/s which is quite similar to the throughput of the messaging service provided by the application-specific NoC. This is caused by the fact that the basic principle of the QoS based router microarchitecture is very similar to the part of the application-specific router microarchitecture which provides the messaging service. Since the application-specific NoC implements dedicated physical channels and structures for every communication service, the overall system throughput is approximately 3.4 GBit/s higher than the overall system throughput of the QoS based NoC.

3) Packet Latency: Another benchmark figure of an on-chip communication system is the average latency of transferring packets. This value is of significant interest for applications which demand a certain level of real-time performance.

The average packet latencies of the application-specific NoC and the QoS based NoC, depending on the accepted bit rate, are illustrated in Figure 7. The diagram of the application-specific NoC shows that two important requirements, which are stated in Section III-D, are met by the application-specific design.

- The TAS and time distribution services generate a constant packet latency of 4 clock cycles and 22 clock cycles, accordingly. This is due to the fact that the NoC provides dedicated physical channels and structures for the root injection of these communication services.
- Since the random read/write access service is provided by the bus enhancement of the NoC, the average access latency of this communication service is very low. Assuming that the slave implements a synchronous interface and that the master interface is registered, the access latency shows values between only 1 clock cycle and 2 clock cycles.

The average packet latency of the messaging service, on the other hand, demonstrates a typical NoC behavior. Increasing the bit rate leads to a higher average packet latency which reaches its maximum at the peak throughput capability of approximately 5.5 GBit/s. This is due to the fact that all router interconnections of the fat-tree based topology provide an equal throughput capability. If this throughput capability is fully utilized, packets are blocked within the routers and the average packet latency increases.

Unlike the application-specific NoC, the QoS based NoC demonstrates average packet latencies for all provided communication services depending on the accepted bit rate. This behavior is caused by sharing the physical channels and structures among all communication services. It has to be kept in mind that additional hardware structures are required to provide a constant latency for the distribution of the TAS and time information. Considering their high priorities, the TAS and time distribution services and the messaging service provide the lowest average packet latency of approximately 12 clock cycles and approximately 15 clock cycles at the maximum throughput capability of approximately 2.1 GBit/s. In general, the random read/write access service has the lowest priority. Therefore, packets belonging to this service are blocked within the routers until all packets with higher priority have been transferred. At the maximum throughput capability this behavior leads to a rather high average packet latency of approximately 140 clock cycles.

4) Silicon Area: The silicon area effort of an on-chip communication system is a further important characteristic. Due to the fact that the silicon area is tightly coupled to the power consumption of a system, this figure is of significant interest for evaluating low power designs.

Table III demonstrates the silicon area figures of the application-specific NoC and the QoS based NoC measured in gate equivalents. It can be seen that the effort for implementing the application-specific NoC is very low. This is caused by the fact that the specific requirements and restrictions of the communication services are considered and that dedicated physical channels and structures are used to implement these services.

The QoS based NoC, in contrast, shows a gate count which is more than three times the gate count of the application-specific NoC. As stated before, the main reason for this growth are the multiple FIFOs needed to implement the

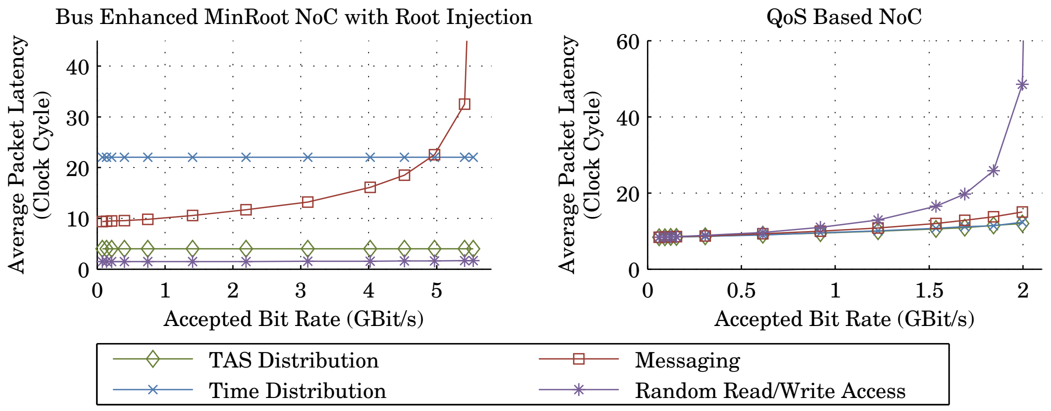

Fig. 7. Average packet latencies of the bus enhanced MinRoot NoC with root injection and the QoS based NoC.

TABLE III
SILICON AREA COMPARISON OF THE BUS ENHANCED MINROOT NOC WITH ROOT INJECTION AND THE QOS BASED NOC.

Communication Service	**Bus enhanced MinRoot NoC with Root Injection**	**QoS based NoC**
TAS Distribution	928 Gate	43.359 kGate
Time Distribution	97 Gate	
Messaging	7895 Gate	
Random Read/Write Access	4015 Gate	
	12.935 kGate	43.359 kGate

virtual channels. The results of the synthesis indicate that these FIFOs allocate approximately 55 % of the overall silicon area. Another important factor has to be kept in mind when comparing the silicon area of both systems: The gate count of the QoS based NoC does not include the additional hardware structures which are needed to provide a constant packet latency for the TAS and time distribution services.

V. APPLICATION-SPECIFIC RISC CORE

A. Concept

1) Instruction Set Architecture: The first attempt of implementing an application-specific core for the novel architecture approach utilized a very long instruction word (VLIW) architecture. But an analysis of this design, called soft finite-state machine core (SoftFSMC), showed that most instructions of the evaluated benchmark algorithm are not able to use the provided parallelism resulting in a very low code density.

Considering this disadvantage, the further development focused on an application-specific reduced instruction set computer (RISC) core called virtual finite-state machine core (vFSMC). Typically, RISC instruction sets do not include complex instructions, especially instructions which combine memory accesses with arithmetic and logical operations [24]. These load-store architectures allow the implementation of well balanced instruction pipelines which can be clocked at high frequencies. On the other hand, the code density can be increased by implementing arithmetic and logical instructions which support these memory accesses. The following sections also demonstrate that the effort for achieving cycle accurate multithreaded processing is quite low when implementing the accesses to the memories within the instruction pipeline.

Therefore, the instruction set architecture (ISA) of the novel vFSMC is based on the ISA of the Texas Instruments MSP430 [25] which includes several memory addressing modes for every arithmetic and logical instruction. In order to increase the performance of the core and to ease the complexity of the instruction pipeline, the following modifications are applied to the native ISA.

- The destination addressing mode of the two-operand instruction format is reduced to register addressing. This is due to the fact that only one word can be fetched from the data memory per clock cycle.
- A move instruction format, which allows to specify a memory location for both operands, is introduced. Consequently, move operations from one memory location to another one can be specified by a single instruction.
- Decimal add and subtract instructions are removed since they are not required for the targeted application.
- Single bit shift instructions are replaced by multi-bit shift instructions to reduce the execution time and to increase the code density.
- Multiply instructions are embedded within the instruction set. Due to the architecture of the instruction pipeline, the results of these instructions are written to the general purpose registers R14 and R15.
- A loop instruction, which allows the iteration of the successive instruction, is inserted. This concept increases the code density in case of executing the same instruction repeatedly.
- Considering that the core implements a hardware multithreaded architecture, mutex instructions are added to handle multiple accesses to shared areas within the ex-

ternal resources.
- Interrupt related instructions are removed because the threads, which are executed on the vFSMC, are not interruptible.

Similar to the ISA of the MSP430, the ISA of the vFSMC makes use of a register file which provides 16 registers with a width of 16 Bit. Furthermore, the ISAs of both cores support the same addressing modes which include register addressing, displacement addressing, absolute addressing, register indirect addressing, postincrement addressing, and immediate addressing.

A major difference between both cores refers to the addressing of different word sizes. Basically, the ISA of the MSP430 provides byte and word addressing functionalities. The ISA of the vFSMC extends these functionalities by introducing a bit field addressing mode which is able to overcome the drawbacks of accessing bit fields within the data memory, the register sets, or the register file of the core itself. The difficulty of accessing bit fields is well known in optimized systems which keep the bit width of operands and parameters as short as possible to save memory and register space. While the MSP430 requires several instructions to perform an operation on a bit field, the ISA of the vFSMC is able to define a single atomic instruction which performs an operation on a bit field in a single instruction cycle.

2) Hardware Multithreading: Hardware multithreading is commonly used to increase the utilization of a single core by applying thread-level parallelism (TLP) [24]. The target of the vFSMC, in contrast, is not to maximize the utilization of the core but to apply cycle accurate execution to each thread. Due to the scalar architecture of the core, using fine-grained temporal multithreading is likely the best approach. This multithreading technique requires a microarchitecture which is able to issue an instruction from a different thread at every clock cycle.

However, the vFSMC implements a special type of fine-grained temporal multithreading which executes the threads in an ascending order even if they are idle. The execution of four threads, using ordered fine-grained temporal multithreading, is depicted in Figure 8. The diagram shows that the threads are processed on a fixed time grid which allows a cycle accurate execution of every thread without interfering each other. Compared to the active threads, the idle threads consume time slots but their state is never modified.

A further advantage of this multithreading technique is the elimination of all hazards which are caused by internal dependencies within the instruction pipeline. Of course, this is only assured if the number of supported threads is equal or greater than the number of pipeline stages. Considering that the design of the vFSMC follows this rule, the additional hardware structures for detecting and resolving hazards are not required.

Since the vFSMC implements a shared instruction pipeline, the resulting instruction execution frequency of each thread depends on the number of supported threads. It can be expressed by

$$f_{\text{Thread}} = \frac{f}{N_{\text{Thread}}} \tag{1}$$

where f specifies the clock frequency of the core and N_{Thread} represents the number of supported threads which has to be equal or greater than the number of pipeline stages.

3) System Architecture: Even though the vFSMC is based on the ISA of the MSP430, both core architectures differ widely. Thus, different system architectures are required to embed the cores. A major difference between the MSP430 and the application-specific RISC core refers to their memory architecture. While the native design of the MSP430 implements a Von Neumann architecture, the design of the vFSMC makes use of a pure Harvard architecture to enable ordered fine-grained temporal multithreading. Basically, the ISA of the core allows a maximum instruction length of 64 Bit. For this reason, the width of the instruction interface is extended from 16 Bit to 64 Bit. This extension allows to fetch an entire instruction per clock cycle and therefore, a constant stream of instructions can be provided to the instruction pipeline.

The variable instruction length, provided by the ISA of the vFSMC, allows a segmentation of a single instruction in two consecutive instruction memory words. Consequently, the core requires a line cache memory which holds the most significant 48 Bit of the previously fetched instruction memory word for each thread. Considering that the cache interface must be able to concurrently issue a read access and a write access, this memory has to provide simultaneous read and write accesses.

In order to ensure a constant stream of instructions through the instruction pipeline without any stalls, the design of the vFSMC makes use of concurrent read and write accesses through the data interface. Hence, storages connected to this interface have to provide simultaneous read and write accesses, too. Moreover, all resources connected to the vFSMC have to accept requests from the core without any wait states to enable the cycle accurate execution of every thread.

4) Core and Pipeline Architecture: As shown in Figure 9, the vFSMC implements a three stage instruction pipeline which includes a fetch stage, a decode and load stage, and an execute and store stage. Additionally, the core provides a multiply unit which is an extension to the instruction pipeline.

To enable ordered fine-grained temporal multithreading, every thread owns a dedicated register file which is embedded in the global register file. According to the ordered execution of threads, the fetch stage generates thread identifiers in an ascending order. These identifiers are passed through the instruction pipeline and therefore, each stage and the multiply unit are able to determine the relation of the currently processed instruction for accessing the correct thread register file.

Besides generating the thread identifiers, the fetch stage provides the control information which is required to access the instruction memory and the line cache memory. Moreover, the stage is responsible for handling loop operations which are indicated by the described loop instruction. The requested data from the instruction memory and the line cache memory is directly passed to the decode and load stage which assembles the instruction segments. Subsequently, the instruction is decoded and, if required, the source operand is loaded from the data memory or the thread register file.

If necessary, the execute and store stage extracts the speci-

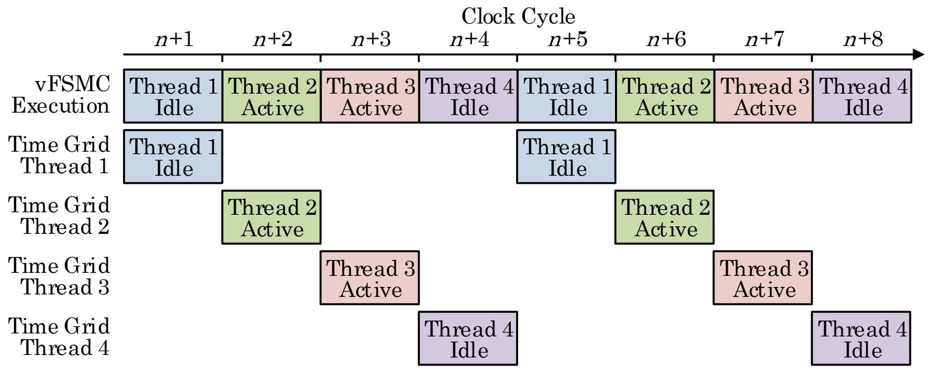

Fig. 8. Execution of four threads on the vFSMC using ordered fine-grained temporal multithreading.

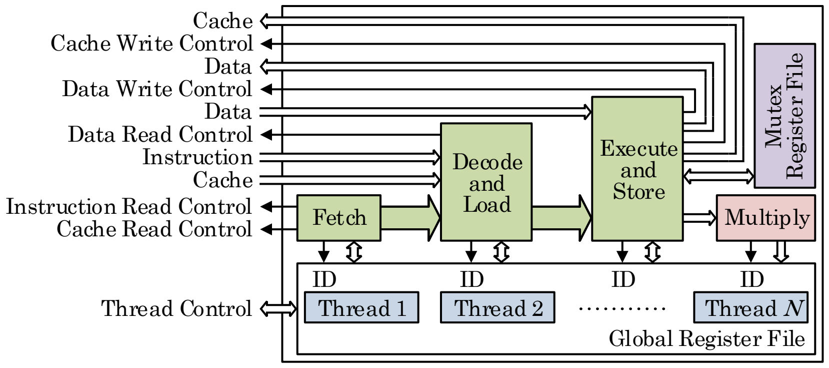

Fig. 9. Core and pipeline architecture of the vFSMC.

fied bit field from the previously loaded source operand. Furthermore, the second source operand, which is only required for two-operand instructions, is accessed in the thread register file. After performing the operation the stage makes use of the destination operand and the possibly specified bit field to store the result to the data memory or the thread register file. In addition, it writes the most significant 48 Bit of the fetched instruction memory word to the line cache memory. In case of processing a branch instruction, the execute and store stage generates the target address and the program counter (PC) register is updated based on the branch condition.

In addition to the global register file, the vFSMC embeds a mutex register file. This register file is exclusively accessed by the execute and store stage to handle the stated mutex instructions. In order to remove all hazards from the instruction pipeline of the vFSMC which are caused by internal dependencies, the minimum number of supported threads is set to three. If less threads are utilized, one or two thread register files can be forced to zero and their respective threads to idle. The benchmark figure in Section V-B4 illustrates that this approach avoids the introduction of flip flops and therefore, the silicon area can be scaled linearly with the number of supported thread.

B. Benchmarking

1) Environment: In general, three competitive cores are used to analyze the performance of the vFSMC. This group includes the power optimized reduced instruction set computer core (PORC) [26], the ARM Cortex-M0 [27], and the SoftFSMC.

The PORC is a power optimized 16-Bit processor which is compatible with the native ISA of the Texas Instruments MSP430. In order to allow a fair competition, the core makes uses of a hardware multiplier which is embedded as a peripheral. The ARM Cortex-M0 is the smallest ARM processor available. It implements a high performance 32-Bit central processing unit (CPU) which provides a small silicon area and a low power consumption. As already stated above, the SoftFSMC was the first approach of implementing a core for cycle accurate multithreaded processing. It embeds a 128-Bit VLIW architecture and utilizes the same ordered fine-grained temporal multithreading technique like the vFSMC. Since the core is only analyzed at a behavioral level, silicon area figures are not available for this core.

Common benchmark algorithms are used to evaluate the performance of general purpose processors, however, these standardized algorithms are typically not suitable to demonstrate the capabilities of an application-specific design. Therefore, the automatic gain control (AGC) algorithm, which is a well known and challenging methodology used within RF transceivers, is used to benchmark the cores. This algorithm controls the gain of the receive path in order to provide a stream of data samples with an optimally tuned amplitude. Basically, choosing the AGC algorithm for the UMTS standard to analyze the performances of the cores has various reasons. On the one hand, the WCET of this algorithm has to be reduced to a minimum to increase the performance of the

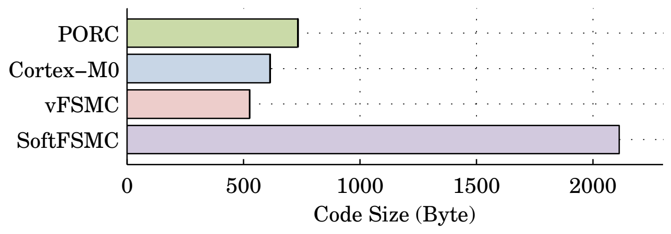

Fig. 10. Comparison of the code sizes needed to implement the AGC algorithm.

RF transceiver. On the other hand, the timing for updating the parameters of the receive path is tightly constrained. This characteristic emphasizes the real-time capabilities of the evaluated cores.

The clock frequency of all systems is set to 104 MHz. This is also the target clock frequency for synthesizing the PORC and the vFSMC using the CMOS C65LP technology library.

2) Code Size: A very important benchmark figure of an ISA is its code density. This characteristic is of significant interest for embedded microprocessor systems because the size of the instruction memory directly influences the silicon area, the power consumption, as well as the costs.

Figure 10 illustrates the code sizes which are required to implement the AGC algorithm for the PORC, the Cortex-M0, the vFSMC, and the SoftFSMC. As already mentioned above, the SoftFSMC offers a very low code density. This is due to the fact that the core implements a VLIW architecture which enables the parallel execution of multiple operations. But for the selected algorithm most instructions are not able to use the provided parallelism. The PORC and the Cortex-M0, in contrast, show a rather good performance.

However, the ISA of the application-specific RISC core is most efficient and generates a code size of only 527 Byte. This value is approximately 30 % lower than the code size required for the PORC and approximately 15 % lower than the code size needed for the Cortex-M0. The analysis shows that the code size reduction is mainly caused by the following features of the ISA: the arithmetic and logical instructions which are able to directly access memory operands, the bit field addressing mode, and the multi-bit shift instructions.

3) Execution Time: The processing performance of a core is one of the most important benchmark figures. This characteristic is evaluated by comparing the execution times of the AGC algorithm.

In order to evaluate the single-threaded execution times of the vFSMC and the SoftFSMC, the number of threads supported by these hardware multithreaded architectures is set to the minimum number of three. Furthermore, two thread register files are forced to zero and their respective threads to idle. This approach generates comparable single-threaded designs. The diagram in Figure 11 demonstrates that the single-threaded execution times of the PORC, the Cortex-M0, the vFSMC, and the SoftFSMC are quite similar.

However, the vigorous processing performance of the application-specific RISC core is demonstrated in case of executing several internal control tasks in parallel. Due to

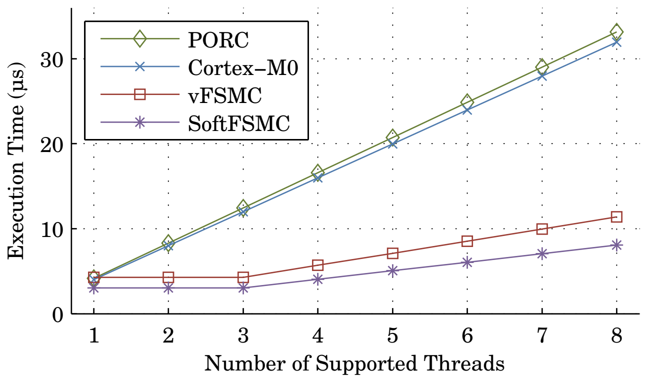

Fig. 11. Comparison of the execution times needed to process the AGC algorithm.

the fact that the vFSMC and the SoftFSMC implement a hardware multithreaded architecture, processing these tasks can be distributed to various dedicated threads. The single-treaded architectures of the PORC and the Cortex-M0, on the other hand, have to process the internal control tasks sequentially. In case of the following analysis, several AGC algorithms are processed in parallel or sequentially depending on the architecture of the core. This scenario can be found in RF transceivers which implement a MIMO architecture or receive diversity in the RX.

Figure 11 demonstrates that for processing one to three algorithms the execution times of the vFSMC and the SoftFSMC show a constant value. The reason for this behavior is the minimum number of supported threads which is set to three for both cores. For processing more than three AGC algorithms, the execution times of the vFSMC and the SoftFSMC increase by approximately 1.4 µs and approximately 1 µs per additional algorithm, respectively. The execution times of the PORC and the Cortex-M0, in contrast, increase linearly with the number of sequentially processed algorithms. For processing three AGC algorithms the gap between the execution times of the PORC and the vFSMC is already approximately 8.2 µs. The excellent multithreaded processing performance of the vFSMC is even more effective in case of processing a higher number of algorithms. On the one hand, this increase in processing performance is induced by the special features of the ISA which are already stated in Section V-B2. On the other hand, the ordered fine-grained temporal multithreading technique is able to remove all hazards from the instruction pipeline of the core which are caused by internal dependencies.

4) Silicon Area: Apart from the code density of the ISA and the processing performance of the microarchitecture, the silicon area effort of a core is a further important characteristic. Since this figure is tightly related to the power consumption of a system, it is of significant interest for evaluating low power designs. Moreover, it is used to measure the effort of hardware resources. In order to provide realistic gate counts with respect to the static timing analysis, all required memories are embedded in the synthesis. However, the gate counts presented in this benchmark only refer to the cores themselves and do not include the memories. As already mentioned above,

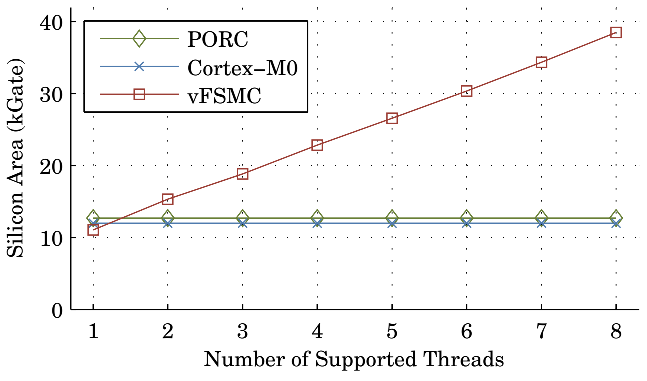

Fig. 12. Silicon area comparison of the PORC, the Cortex-M0, and the vFSMC.

silicon area figures for the SoftFSMC are not available because the core is only analyzed at a behavioral level.

Figure 12 shows the silicon area efforts of the PORC, the Cortex-M0, and the vFSMC measured in gate equivalents. Due to the fact that the PORC and the Cortex-M0 do not support multiple threads by hardware, their gate counts are constant and do not dependent on the number of supported threads. The synthesis results of the PORC demonstrate a value of approximately 12.7 kGate including the hardware multiplier which is embedded as a peripheral. As stated in [27], the gate count of the Cortex-M0 is rated at a value of approximately 12 kGate.

The application-specific RISC core, in contrast, implements a dedicated register file for every supported thread. As described in Section V-A4, the minimum number of supported threads is set to three. If less threads are utilized, one or two thread register files can be forced to zero and their respective threads to idle. Using this approach, the synthesis tool introduces constants instead of flip flops, and the gate count can be scaled almost linearly even if only one or two threads are used. In general, the number of supported threads can be chosen freely in advance to synthesizing the design to meet the exact requirements of the internal control tasks. Starting at a value of approximately 11 kGate for supporting a single thread, the gate count of the vFSMC increases by approximately 3.9 kGate for each additionally supported thread. This low silicon area effort is mainly introduced by the clear design of the instruction pipeline which excludes the hardware structures for detecting and resolving hazards caused by internal dependencies.

VI. Conclusion

This article focuses on the design of interface and control architectures for RF transceivers which are used to control the internal and external, analog and digital modules. Considering that the increasing complexity of these architectures cannot be handled by traditional concepts, the partitioning of RF transceiver systems and the implementation of application-specific components are evaluated. The basic knowledge about the topic is introduced by illustrating the evolution of interface and control architectures, and by analyzing the future trends and the internal control tasks of modern multi-standard RF transceivers.

The novel architectural approach is based on a distributed controlling concept which partitions the RF transceiver into stand-alone units capable of controlling themselves. Due to the strict real-time requirements, the architecture makes use of a common time base which is needed to synchronize linked tasks of different units. The on-chip communication of the system is enabled by an application-specific NoC which is tailored to the specific requirements of RF transceivers. In order to fulfill the requirements of the internal control tasks, an application-specific RISC core, which provides cycle accurate multithreaded processing, is introduced.

The examination of the novel architectural approach shows great benefits. The partitioning of the system provides an excellent scalability and the concept encapsulates the entire functionality of each unit which has positive effects on the reusability. In general, the high degree of scalability, flexibility, and reusability allows to reduce the time to market for RF transceivers and enables fast adaptations to the requirements of the market. Furthermore, the application-specific components feature a conclusive performance. Compared to common architectures, the application-specific designs are able to provide better or at least equivalent performance results while the silicon area can be reduced. This characteristic has positive effects on the costs as well as on the power consumption of the RF transceiver.

The coherent concept of the novel MPSoC interface and control architecture still has to prove itself in the product environment. But the advantages of the architecture already attract attention, and particular considerations are incorporated in upcoming RF transceiver integrations.

References

[1] S. Brandstätter, B. Neurauter, and M. Huemer, "A novel architectural approach for control architectures in RF transceivers," in *Proceedings of the 2010 IEEE International SOC Conference (SOCC)*, Las Vegas, Nevada, USA, September 2010, pp. 407–412.

[2] S. Brandstätter and M. Huemer, "VFSMC - A Core for cycle accurate multithreaded Processing in hard real-time Systems-on-Chip," in *Proceedings of the 2011 IEEE International SOC Conference (SOCC)*, Taipei, Taiwan, September 2011, pp. 312–317.

[3] ——, "An Application-Specific Network-on-Chip for Control Architectures in RF Transceivers," in *Proceedings of the 2012 International Conference on Embedded Computer Systems (SAMOS)*, Samos, Greece, July 2012, pp. 68–75.

[4] D. Wenzel, "System Architectures in Multi-Mode Mobile Terminals," in *Proceedings of the 2007 European Conference on Wireless Technologies (ECWT)*, Munich, Germany, October 2007, pp. 16–19.

[5] Intel Corporation, "Intel at Mobile World Congress 2011," Press Release, February 2011. [Online]. Available: http://newsroom.intel.com/servlet/JiveServlet/download/1831-42-3782/MWC_factsheet_2011.pdf

[6] B. Wilkins, "The Benefits of Standard Radio-to-Baseband Digital Interfaces," *High Frequency Electronics*, vol. 2, no. 4, pp. 38–40, July 2003.

[7] MIPI, "DigRF Baseband / RF Digital Interface Specification, Version 1.12," Standard, February 2004.

[8] 3GPP, "Digital cellular telecommunications system (Phase 2+); Radio subsystem synchronization (3GPP TS 45.010 version 7.4.0 Release 7)," Standard, April 2008.

[9] MIPI, "DigRF Dual-Mode 2.5G / 3G Baseband / RF IC Interface Standard, Version 3.09," Standard, November 2006.

[10] 3GPP, "Universal Mobile Telecommunications System (UMTS); Requirements for support of radio resource management (FDD) (3GPP TS 25.133 version 8.2.0 Release 8)," Standard, April 2008.

[11] P. Hooijmans, "Architectures for mobile RF convergence and future RF transparency," *RFDESIGN*, pp. 18–24, February 2006.

[12] A. Springer, L. Maurer, and R. Weigel, “RF System Concepts for Highly Integrated RFICs for W-CDMA Mobile Radio Terminals,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 1, pp. 254–267, January 2002.

[13] J. Pekarik, D. Greenberg, B. Jagannathan, R. Groves, J. Jones, R. Singh, A. Chinthakindi, X. Wang, M. Breitwisch, D. Coolbaugh, P. Cottrell, J. Florkey, G. Freeman, and R. Krishnasamy, “RFCMOS Technology from 0.25μm to 65nm: The State of the Art,” in *Proceedings of the 2004 IEEE Custom Integrated Circuits Conference (CICC)*, Orlando, Florida, USA, October 2004, pp. 217–224.

[14] F. Pollack, “New Microarchitecture Challenges in the Coming Generations of CMOS Processing Technologies (Keynote),” in *Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture (MICRO)*, Haifa, Israel, November 1999.

[15] ARM, “Debug and Trace for Multicore SoCs,” White Paper, September 2008. [Online]. Available: http://www.arm.com/files/pdf/CoresightWhitepaper.pdf

[16] OpenCores, “OpenRISC 1200 IP Core Specification (Preliminary Draft),” Manual, May 2012. [Online]. Available: http://openrisc.net/or1200-spec.html

[17] W. Ahmed, S. Brandstätter, and M. Huemer, “Verifying Open Source CPU Cores using Instruction Set Simulators in OVM Environments,” in *Proceedings of the 2011 Austrochip*, Vienna, Austria, September 2011, pp. 45–50.

[18] N. Suri, M. Hugue, and C. Walter, “Synchronization Issues in Real-Time Systems,” *Proceedings of the IEEE*, vol. 82, no. 1, pp. 41–54, January 1994.

[19] M. Horchani, M. Atri, and R. Tourki, “Design of a NoC-router guaranteeing QoS based on Virtual Channels reservation,” in *Proceedings of the 2nd International Conference on Signals, Circuits and Systems (SCS)*, Nabeul, Tunisia, November 2008.

[20] Arteris, “A comparison of Network-on-Chip and Busses,” *Design and Reuse*, May 2005.

[21] M. Ali, J. Jamali, and A. Khademzadeh, “MinRoot and CMesh: Interconnection Architectures for Network-on-Chip Systems,” *World Academy of Science, Engineering and Technology*, vol. 54, pp. 354–359, 2009.

[22] R. Manevich, I. Walter, I. Cidon, and A. Kolodny, “Best of Both Worlds: A Bus Enhanced NoC (BENoC),” in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, San Diego, California, USA, May 2009, pp. 173–182.

[23] OpenCores, “WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores (Revision B4),” Standard, June 2010. [Online]. Available: http://cdn.opencores.org/downloads/wbspec_b4.pdf

[24] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed. Morgan Kaufmann Publishers Inc., 2007.

[25] Texas Instruments, “MSP430™Ultra-Low-Power Microcontrollers,” Booklet, 2012. [Online]. Available: http://www.ti.com/lit/sg/slab034v/slab034v.pdf

[26] W. Guscheh, M. Schutti, M. Siegel, and T. Pühringer, “Power Optimierung auf Entwurfsebene von einem 16-Bit RISC-Prozessor für mobile Endgeräte,” in *Proceedings of the 2007 Austrochip*, Graz, Austria, October 2007, pp. 143–148.

[27] ARM, “ARM® Cortex™-M0,” Booklet, 2009. [Online]. Available: http://www.mcu-related.com/pdf/CortexM0-flyer.pdf

Siegfried Brandstätter was born in Salzburg, Austria, in 1982. He received the Dipl.-Ing. (FH) degree in hardware/software systems engineering from the Upper Austria University of Applied Sciences, Hagenberg, Austria, in 2006. From 2007 to 2013, he investigated interface and control architectures for multi-standard RF transceivers in cooperation with DMCE, a subsidiary company of Intel Mobile Communications. In 2013 he received the Dr.techn. (Ph.D.) degree in information and communication technology from the University of Klagenfurt, Klagenfurt, Austria. His current research interests are focused on real-time controlling, networks-on-chip, and application-specific processors.

Mario Huemer (SM’-2007) was born in Wels, Austria, in 1970. He received the Dipl.-Ing. degree in mechatronics and the Dr.techn. (Ph.D.) degree from the Johannes Kepler University of Linz, Austria, in 1996 and 1999, respectively. From 1997 to 2000, he was a research assistant at the Institute for Communications and Information Engineering at the University of Linz, Austria. From 2000 to 2002, he was with Infineon Technologies Austria, research and development center for wireless products. From 2002- 2004 he was a lecturer at the University of Applied Sciences of Upper Austria, and from 2004-2007 he was Associate Professor for Electronics Engineering at the University of Erlangen-Nuremberg, Germany. In 2007 Mario Huemer moved to Klagenfurt, Austria, to establish the Chair of Embedded Systems and Signal Processing at Klagenfurt University as a full professor. From 2012 to 2013 he served as dean of the Faculty of Technical Sciences. Since September 2013 he is head of the newly founded Institute of Signal Processing at the Johannes Kepler University of Linz, Austria.

His research interests are adaptive and statistical signal processing, signal processing architectures and implementations, as well as mixed signal processing and control with applications in communications, radio frequency and baseband integrated circuits, battery- and power management for mobile devices, and sensor signal processing. Within these fields he published more than 160 papers. In 2000 Mario Huemer received the German ITG and the Austrian GIT award for dissertations, and in 2010 the Austrian Kardinal Innitzer award in natural sciences. His review work includes national and European research projects as well as international journals. Since 2009 he is member of the editorial board of the ”International Journal of Electronics and Communications (AEUE)”.

Mario Huemer is member of the IEEE, the German Society of Information Technology (ITG), and the Austrian Electrotechnical Association (OVE).