A Software-Programmable Multiple-Standard Radio Platform

Hans-Martin Blüthgen, Cyprian Graßmann and Ulrich Ramacher
Infineon Technologies, Corporate Research, CPR ST, Munich.

***Abstract*— Future wireless terminals will have to be multi-band, multi-standard and able to execute multiple standards concurrently. In this paper we describe a flexible and programmable baseband platform for a large variety of mobile and WLAN standards. For the SDR platform architecture our primary design goal was to find the most flexible and easy-to-program solution within a specified power budget. The result is an architecture consisting of a cluster of four single-instruction multiple-data (SIMD) DSP cores each containing four processing elements and operating at 300 MHz. The cluster of SIMD cores is accompanied by dedicated processors for filtering operations, and channel encoding and decoding. Beside a programming environment for this platform consisting of an application programming interface (API), compiler and debugger, and a virtual prototype of the hardware, we investigate alternatives for a model based design approach Profiling results for the digital signal processing software performing the PHY layer of IEEE 802.11b on the virtual prototype underline the feasibility of our approach.**

***Index Terms*—multi-standard, multi-mode, baseband, SDR platform**

I. INTRODUCTION

Flexibility is becoming one of the most important necessities in future terminals for mobile communications and wireless networks. Recently, the need for more flexibility has gained more attention in the literature (e.g. [2,4,5]). In contrast to today's dual-band single-standard cell phones, future wireless terminals will have to be multi-band, multi-standard and able to do handover between multiple standards and execute them concurrently.

The UMTS Forum estimates that there will be 20.5 million WiFi users by the year 2005 5.3 million of which will also be 3G users. Hence, public WiFi presents a positive market opportunity for mobile operators. The seriousness of this technology is evident by the activities undertaken by AT&T wireless, T-Mobile US in this space, two of the largest mobile carriers in the US.

With the further evolution of 2G-4G communication systems more and more standards have to be supported by and integrated into wireless terminals. With the multitude of existing and evolving standards on the one hand and the need to react to market requirements quickly on the other the system architect's task is undergoing a big change. In the recent past, the design criteria were chosen with regard to what could be realized economically with 0.5–0.13-µm CMOS technologies. This lead to architectures with minimal area and power consumption employing optimized dedicated circuits. With the advent of new standards and the shift to ubiquitous communication, continuation of this style of design would have meant to increase the number of dedicated circuits to an intolerable height. Instead, the idea emerged of using common reconfigurable or programmable circuits covering more than one standard in the RF section as well as in the digital baseband. For the digital baseband, solutions based on a small number of reconfigurable data path units with adjacent small control units have been investigated [1]. The drawback of this kind of solution is especially the complexity of the programming model. In consequence, programs have to be written by the designers of the chip, only, and the customer has to be involved into the partitioning of the system into hardware and software. A success story for reconfigurable computing and communication is still lacking and there is no approach at hand for programming reconfigurable architectures in the wake of ever-increasing demands for flexibility.

The challenge is to develop a system architecture for UMTS FDD at 384 kb/s, CDMA2000 1x DV, GSM/GPRS/EDGE class 12, IEEE 802.11b, IEEE 802.11g (at reduced data rate, e.g. 24 Mb/s), Bluetooth, DAB and GPS.

II. SYSTEM PERSPECTIVE

In general, communication systems consist of analog and digital signal processing sections. The digital section is further partitioned into a digital front-end and the baseband (Figure 1). There is no clear rule for which components of the digital signal processing chain are assigned to the digital front-end or baseband, respectively. For the realization of such a system as a chip set this partitioning usually introduces interfaces between chip boundaries.

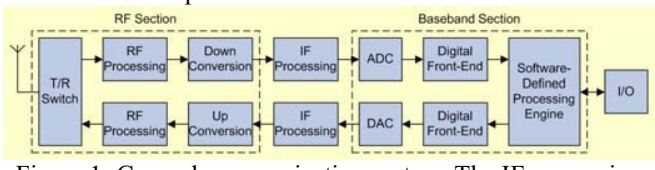

Figure 1: General communication system. The IF processing section is omitted for direct conversion.

The current trend is to have an interface on the receive side with a data rate of two times the chip or symbol rate. On the transmit side the interface has a chip or symbol rate interface. The ideal multi-standard radio would have a fully flexible baseband, one fully programmable digital front-end, and a

wide-band radio frequency (RF) section which is able to cover all necessary frequency bands.

III. RF Section

In principal it is possible to build a fully flexible RF front-end out of programmable or tunable components. These components are wideband antennas, amplifiers, mixers and oscillators, tunable filters, and highly accurate digital synthesizers. However, each of the standards that need to be covered for a multi-standard radio comes with very tight constraints which are difficult to meet even for the single-standard case. Usually, using wideband components means compromising some performance compared to the narrow-band solution optimized for the single-standard case. There are currently efforts to combine the analog signal paths of related communication standards into a single flexible RF front-end. Examples of related communication standards are GSM/GPRS/ WCDMA. However, it is unlikely that the analog signal paths of different families of communication standards (e.g. wireless LAN and mobile phones) can be combined in the near term. Figure 2 shows the block diagram of a multi-standard radio system featuring multiple RF chips for the different standards. This concept can be easily extended if more than two signal paths or completely different air interfaces are required.

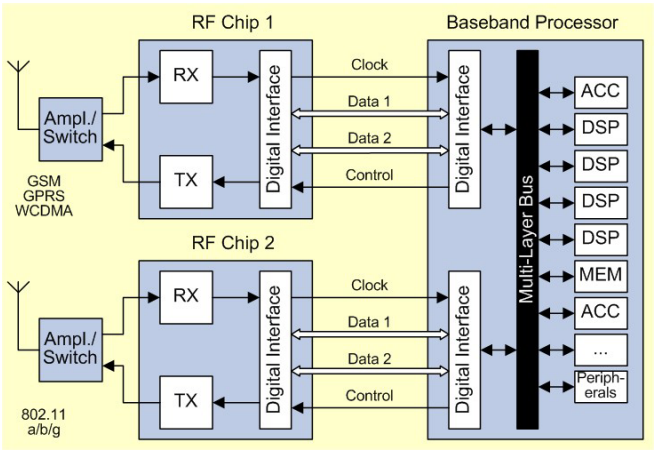

Figure 2: Multi-standard radio system.

IV. Baseband Architecture

The challenge in the development of an optimum architecture for the baseband processor was, that simplicity of programming model and flexibility (capability of executing applications not considered at time of specification) added two more parameters, besides area and power that spanned the architecture design space and made the search for an optimal solution considerably more difficult. Giving highest weight to area and power would mean to search for a solution based upon reconfigurable architectures. To avoid the pitfalls of those, we assumed that customer and chip designer could agree on a reasonable upper limit for area and power, say, 40 mm^2 and 200 mW for execution of UMTS or 802.11b, alternatively. Then, we would be free to design an architecture, which is built for highest flexibility and simplest programming model.

For this goal, the ideal solution would be a single DSP with sufficiently high clock. Instead, because of the limits of a 90-nm technology, the entry point into design space exploration is set by the smallest possible number of general-purpose DSPs working at highest possible clock frequency and V_{dd} so that the power budget is not exceeded. Then, the programming model is as simple as possible and flexibility at highest possible level. If the area would turn out to be too high, extensions to the general-purpose DSP instruction set have to be added, meaning in the worst that an accelerator is invoked. This way, flexibility and simplicity of programming model are compromised at the least, and the area and power budget is kept. With these iteration rules, we arrived at the solution depicted in Figure 3 and described in details below.

In consequence, the next-generation baseband system turned into a software-defined radio system that executes programs and is programmed by the customer by means of high-level APIs with adjacent libraries. This revolutionary innovation will drive the evolution of the baseband systems of the next generation [3].

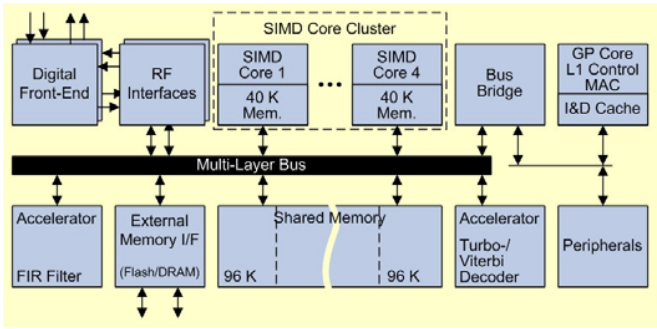

Figure 3: Baseband platform.

With the entry point into the architecture design space chosen as explained above, our estimations on power consumption resulted in an architecture consisting of a cluster of four single-instruction multiple-data (SIMD) DSP cores. This kind of DSP core is particularly suited for the computationally complex algorithms in communication systems [6,7].

The SIMD core (Figure 4) is based upon previous work [8] and has been simplified as well as extended for the application in communication systems. Each SIMD core contains four processing elements and operates with a clock frequency of 300 MHz. It supports special instructions like saturating operations and finite-field arithmetic, and long-instruction word (LIW) features for performing arithmetic operations and memory accesses in parallel. The instruction pipeline is four stages long, which is used to relax the timing requirements for the memories, reduce the memory's supply voltage and thereby the power consumption. However, a long pipeline leads to data dependencies between instructions from the same task. That is why here each of the four pipeline stages contains an instruction of one of four separate tasks. Four SIMD cores each running four tasks result in a task-level parallelism of 16. The cluster of SIMD cores is accompanied by dedicated programmable processors for filtering operations as well as channel encoding and decoding [9].

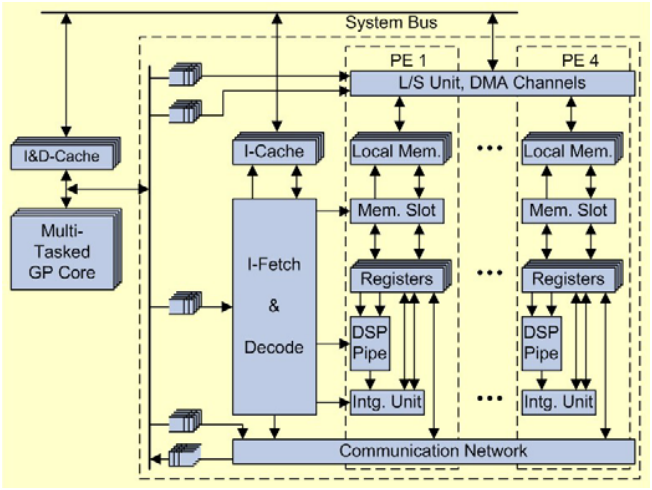

Figure 4: SIMD core.

These dedicated processors account for almost half of the total processing power of the entire SDR platform. In addition, there is an ARM processor for the execution of the protocol stacks. It is important to note that the use of dedicated processors does not compromise flexibility here. The algorithms run on the processors are common to most communication systems and usually are located in the same position within the systems.

Although dedicated processors are employed it is necessary that all parameters of channel coding algorithms and FIR filters used in different standards can be supported. In addition, both dedicated processors must be able to run at least two standards concurrently. Instead of implementing a separate macro for each of the supported modes the processors are based on fine-grain instructions, e.g. for arithmetic operations (add/sub) or bit manipulation. This retains the maximum level of flexibility for the entire platform.

The accelerators' flexibility can be exploited in two ways. Firstly, it is possible to modify the functionality within a single standard. Secondly, to assign more computational resources to one standard and use algorithms that are more sophisticated. Examples are the use of more iterations in Turbo decoding or a larger number of taps for FIR filtering.

V. Software Architecture

It is important to note that the hardware platform architecture resembles a computer architecture. To keep the simplicity of the programming model of this multi-standard radio approach, a suitable software development environment is required that enables the programmer to exploit the potential of the hardware easily, however, without knowing their details. The development environment consists of an application-programming interface (API), compiler and debugger, a real-time operating system, and a cycle-accurate SystemC simulation of the multi-standard radio platform.

The API provides access to functions from a SDR library and their parameters, peripherals, and operating system functions. The SDR library contains simple functions like convolution, complex functions like a RAKE receiver, and complete virtual radio engines like WCDMA.

As the SDR architect must know theses functions, the API interface is the borderline for the intellectual property owned by the ODM and OEM, respectively.

The SIMD compiler is a crucial component of the SDR platform programming environment. It maps C code with data-parallel C extensions (DPCE) onto a parallel processor. The compiler splits the code into a sequential part for the SIMD core controller and generates machine code for the array of processing elements. Simplicity of the programming model requires the compiler to generate machine code that is efficient enough even for performance-critical functions so that no assembler code must be written. The mapping must be done such that the throughput for each of the four processing elements is maximized and the communication bandwidth between the processing elements is minimal. This simple optimization goal turns into a formidable research problem with the increase of architectural variables: various word widths, cache size of each PE, depth of PE pipeline, number of registers, PE-PE communication bandwidth, and more.

The SIMD compiler maps basic blocks represented as functions onto one SIMD core. What is left is the task distribution and scheduling onto the platform of parallel SIMD cores and dedicated processors. Simplicity of the programming model again mandates this process to be automated or at least tool-supported. Nevertheless, it is essential that the scheduling result is efficient in terms of minimum synchronization overhead and maximum utilization of computing resources.

There is an inherent partitioning of a communication system into tasks coming from the algorithms (scrambling, modulation, FIR filter, etc.), which have to be performed. The computational complexity of these tasks shows a huge variance. Usually tasks close to the A/D and D/A converters performing the chip or sample rate processing demand much more computing power than tasks for the bit processing. However, for the mapping onto a parallel platform the system has to be partitioned into a set of balanced tasks with similar run time. This implies that computationally complex tasks must be split into separate tasks if throughput requirements cannot be met. The split of tasks has to take into account their data dependencies. Less complex tasks can be joined into a single combined task making it possible for the compiler to optimize beyond task boundaries.

The automation of this process of mapping and scheduling is a difficult research problem. Our approach is to start with a manual partitioning and static scheduling of the representative standards that will be part of a SDR demonstrator. From this, a methodology and heuristics are derived for the construction of a partitioning and static scheduling tool.

The application software requires a real-time operating system (RTOS) running on the layer-1 controller core and on each SIMD core. The RTOS API is limited to the necessary functions for task creation and synchronization, interrupt-handling, access to peripherals, and input/output. The task scheduling of a parallel program can be done statically, prior to execution as was described before. Nevertheless, parallel execution of multiple standards will also demand for run-time scheduling capabilities. For this, efficient scheduling algorithms will be elaborated and integrated into the operating system.

For future architecture space exploration, we would like to use different processors. In order to reduce the design time for the virtual prototype, we developed a technique to generate the operating system and device drivers automatically.
The iterative methodology of architecture space exploration, which was described above, requires examining and profiling for multiple points in the architecture space. In order to do this in short time and with precise results, it is necessary to use a virtual prototype of the entire system rather than using course-grain models or even designing real hardware. Here, the virtual prototype is based on a cycle-accurate SystemC simulation of the hardware platform with the application software and the operating system running on it. The virtual prototype also allows a programmer to write applications for mobile equipment well ahead of fabrication of the final base band chip. To perform simulations of the virtual prototype with reasonable speed a distribution of the simulation onto parallel workstations and the mapping onto a prototyping hardware based on FPGAs are under consideration.

VI. System Function Modeling

For the design entry it is desirable to have a description of the system function together with all constraints like throughput and latency. It is important that the system description itself does not limit the solution space of the final realization but captures all real-time requirements. This means that the system function description must be equally valid for realizations based on embedded parallel processors, dedicated hardware macros, or even reconfigurable logic like FPGA. Therefore, the system function description must be independent of the architecture onto which the system is finally mapped.
Tools which are used for system function modeling and simulation today are for example: Simulink, COSSAP, SPW, or ML-Designer. None of these tools supports all requirements for an architecture-independent system function description. Models of computation used for the simulation of models are e.g. data flow models with fixed sample rates, or event-driven. The models of computation are not always suitable for an architecture-independent modeling. E.g. data flow models with fixed data rates describe the function in a way that reflects the behavior of a hardware implementation more than a software implementation. Moreover, it is not always possible to describe systems containing subsystems with different models of computation. Fixed sample rates for simulation as it is used in e.g. Simulink are more targeted towards hardware implementation. Some don't support efficient implementation of control, e.g. as state machines. None of the tools supports the annotation of additional information like the constraints mentioned before.

VII. Code generation

Our analysis of the WLAN 802.11b software implementation showed a ratio in the order of five between lines of code for control and lines of code for digital signal processing. It is a straightforward task to manually write and verify optimized code for the digital signal processing tasks, even in low-level assembler language. These functions can be part of a library of generic signal processing functions which can be re-used and customized by parameters for each instantiation.

Writing the control code, however, i.e. the code for task scheduling, synchronization, buffer management, and so on, around the digital signal processing functions is tedious and error-prone, especially, for parallel processor architectures. Using a fixed schema for the implementation of the control code, the design process is substantially accelerated, as all the buffers, synchronization primitives and their arrangement in a parallel program can be generated automatically (e.g. by the generation of multiple thread functions). It allows a quick assessment of different alternatives for a partitioning of the system onto parallel target hardware architectures and avoids error prune, manual modifications of the software implementation which can easily consume person months to run free of errors.

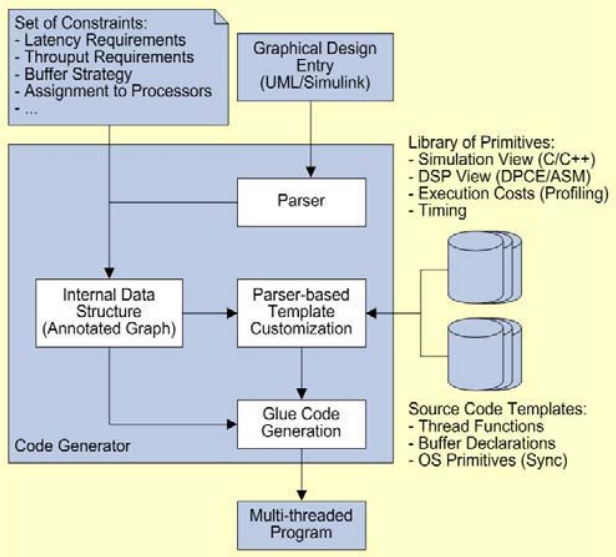

Figure 5: Code generation process.

Currently, we investigate a template based code generation process, which allows us to generate multithreaded C-code for simplified receive and transmit chains of WLAN 802.11b. Along with the completion of receive and transmit chain models the code generator will be extended and completed, to be used for other radio standards, like WCDMA. The basic code generation process is shown in Figure 5. The structure of the model is extracted from a Simulink MDL-file. Additionally, the behavior of functional components and data types and sizes of the connections between them are taken from files describing these components in a generic way.

From the description of the system functions it is than possible to generate the program code for specific architectures automatically. Basically the structural information of the system is translated into control code. The (generic) code of the digital signal processing functions is taken from the library and customized according to the parameters specified in the system description. Beside Simulink other Development tools may be used for the specification and simulation of the system functions. Candidates which we review are SPW form CoWare, CoCentric from Synopsis, Ptolemy from UC Berkeley and

UML tools which support UML profiles.

Some of these tools are already able to generate code for embedded targets from system models. However, only single processors or single threads are supported. There is no tool available right now that is able to generate efficient code for embedded parallel processors. We are currently investigating tools that fulfill all requirements stated above [10].

VIII. Application Profiling

In order to prove the feasibility of our approach, we started with a detailed profiling of physical layer software running on the virtual prototype of our hardware platform. The first communication standard we investigated was WLAN 802.11b. WLAN standards, in general, show the most demanding requirements regarding throughput and latency.

The profiling is carried out in two steps. Firstly each function of the receive- and transmit-chain is profiled in detail on the cycle-accurate simulator in order to find out the optimization potential both for hardware and software. Secondly the single functions are embedded into a multithreaded control and the resulting application is profiled on the cycle accurate virtual prototype of the base band architecture. The virtual prototype allows tracing the accesses to the synchronization primitives, without changing the application behavior. Figure 6 shows a line for each process taking part in the processing of a receive chain.

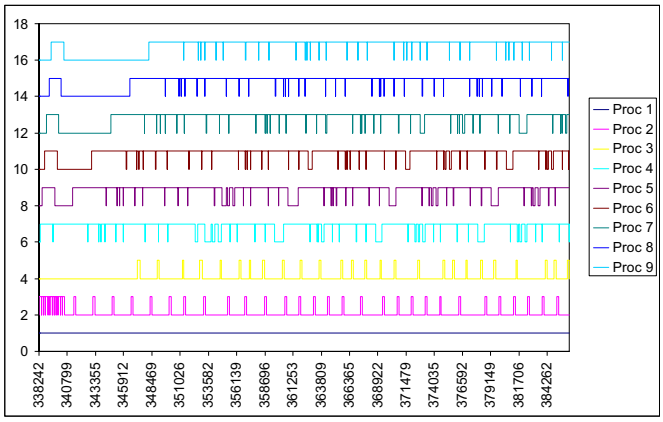

Figure 6: Traces of parallel WLAN 802.11b implementation

We developed a tool which automatically analyses these traces to determine the critical path in the application and to determine the necessary number of threads needed to meet throughput requirements and to determine the actual latency and throughput requirements of the current implementation.

If further optimizations are needed, the analysis results allow focusing these optimizations onto the critical sections of the design. Therefore a cycle accurate simulation, using a virtual prototype for the detailed profiling is crucial for a fast and efficient hardware optimization.

All in all our profiling results show that there is no blocking point in principle for this approach. Currently, profiling is also carried out for the UMTS physical layer which is one of the most demanding standards in terms of complexity.

IX. Conclusion

For the baseband of future multi-standard multi-band cell phones, flexibility and simplicity of the programming model turn out to be the decisive design criteria. Although baseband processing will be growing in number of tasks, our research has shown that the respective performance requirements will be able to be accommodated in a 90-nm CMOS technology by parallel programmable SIMD DSPs with few adjacent dedicated processors. New tools are required for system function capturing and code generation for this parallel computer architecture. Profiling results of representative communication standards show the feasibility of this approach.

Acknowledgment

Contributions by the following colleagues are highly appreciated: N. Brüls, Y. Fonin, U. Hachmann, D. Langen, M. Löw, W. Raab, M. Richter, M. Sauermann, A. Schackow, A. Troya.

References

[1] U. Ramacher, H.-M. Blüthgen, “The Architectural Challenge of Next-Generation Baseband Systems”, ResearchTrends, Oct. 2003.

[2] T. Arnaud, “Multi-Standard Receiver Architecture and Circuits”, European Solid-State Circuits Conference, Lisbon, Sep. 2003.

[3] U. Ramacher, “Next Generation Embedded Communication Systems: Reconfigurability, Flexibility and Programmability”, Intel Corp. Hillsboro/Oregon, On Chip Reconfigurable Computing and Communications Workshop, May 2003.

[4] J. Glossner et al., “A Software-Defined Communications Baseband Design”, IEEE Communications Magazine, Jan. 2003.

[5] R. Wilson, “Intel Tips Plans for Reconfigurable Radio Architecture”, EE Times, 22 Jan. 2003.

[6] J.-P. Giacalone, “Trends in Programmable DSP Architecture for new Generation Wireless Modems”, European Solid-State Circuits Conference, Lisbon, Sep. 2003.

[7] G. Fettweis, M. Bolle, J. Kneip, M. Weiss, “OnDSP: A New Architecture for Wireless LAN Applications”, Embedded Processor Forum, San Jose, May 2002.

[8] W. Raab, N. Brüls, U. Hachmann, J. Harnisch, U. Ramacher, C. Sauer, A. Techmer, “A 100-GOPS Programmable Processor for Vehicle Vision Systems”, IEEE Design & Test of Computers, vol. 20, no. 1, Jan. 2003.

[9] H.-M. Bluethgen, C. Grassmann, W. Raab, U. Ramacher, J. Hausner, “A Programmable Baseband Platform for Software-Defined Radio”, SDR Technical Conference, Phoenix, Nov. 2004.

[10] C. Grassmann, M. Sauermann, H.-M. Bluethgen, U. Ramacher, “System-Level Hardware Abstraction for Software-Defined Radios”, SDR Technical Conference, Phoenix, Nov. 2004.