A PROGRAMMABLE BASEBAND PLATFORM FOR SOFTWARE-DEFINED RADIO

Hans-Martin Bluethgen, Cyprian Grassmann, Wolfgang Raab, Ulrich Ramacher,
Josef Hausner, Infineon Technologies AG, 81609 Munich, Germany,
Hans-Martin.Bluethgen@infineon.com

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. The programming environment of this platform consists of an application programming interface (API), compiler and debugger, and a virtual prototype of the hardware. 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.

1. INTRODUCTION

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 and T-Mobile US in this space, two of the largest mobile carriers in the US.

As with the further evolution of 2G through 4G communication systems the plethora of standards will be combined into powerful super-standards like WCDMATDD and HSDPA – these super-standards varying with world regions – the system architect's task is undergoing a big change. In this paper, we concentrate on the baseband part of cell phones, although the RF and application processing sections share with the baseband part the unique opportunity for innovating the architecture as well as the customer interface.

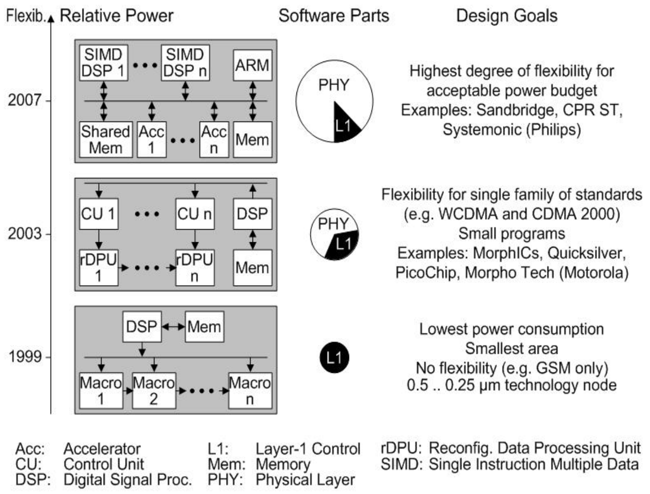
Flexib.
Relative Power
Software Parts
Design Goals
2007
SIMD DSP 1
SIMD DSP n
ARM
Shared Mem
Acc 1
Acc n
Mem
PHY
L1
Highest degree of flexibility for acceptable power budget
Examples: Sandbridge, CPR ST, Systemonic (Philips)
2003
CU 1
CU n
DSP
rDPU 1
rDPU n
Mem
PHY
L1
Flexibility for single family of standards (e.g. WCDMA and CDMA 2000)
Small programs
Examples: MorphICs, Quicksilver, PicoChip, Morpho Tech (Motorola)
1999
DSP
Mem
Macro 1
Macro 2
Macro n
L1
Lowest power consumption
Smallest area
No flexibility (e.g. GSM only)
0.5 .. 0.25 µm technology node
Acc: Accelerator
CU: Control Unit
DSP: Digital Signal Proc.
L1: Layer-1 Control
Mem: Memory
PHY: Physical Layer
rDPU: Reconfig. Data Processing Unit
SIMD: Single Instruction Multiple Data

Figure 1: Trend of flexibility

In the recent past, the design criteria were chosen with regard to what could be realized economically with 0.5 – 0.13µm CMOS technologies. This lead to architectures with minimal area and power consumption. Macros absorbed the compute-intensive signal-processing parts of the physical layer whereas layer-1 control processing was executed on a DSP (Figure 1). 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 macros to an intolerable height. Instead, the idea emerged of emulating the macros by a small number of reconfigurable data path units with adjacent small control units. This way, the firmware increased significantly, the programs could be written by the designers of the chip, only, and the customer had to be involved into the partitioning of the system into hardware and software. Furthermore, the size as well as the number of these data paths would grow with the number of standards and applications, which, in turn, would increase control-overhead and area, let alone the complexity of the programming model.

This innovative architecture style had been first picked up by start-up companies (Quicksilver, Morpho Technologies, Morphics, Mercury, Picochip, to name a few). A success story for reconfigurable computing and communication is still lacking, however, [1]. Not only are area and power consumption inferior to alternative architectures – there is no approach at hand for programming reconfigurable architectures in the wake of ever-increasing demands for flexibility.

Thus, to develop an 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, as requested by a terminal manufacturer, it was imperative to pursue new ways.

2. PLATFORM ARCHITECTURE

Let us remember the challenge: 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 large, 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 2 and described in detail 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 [2].

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 [3][4].

The cluster of SIMD cores is accompanied by dedicated programmable processors for channel encoding and decoding as well as filtering operations. 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.

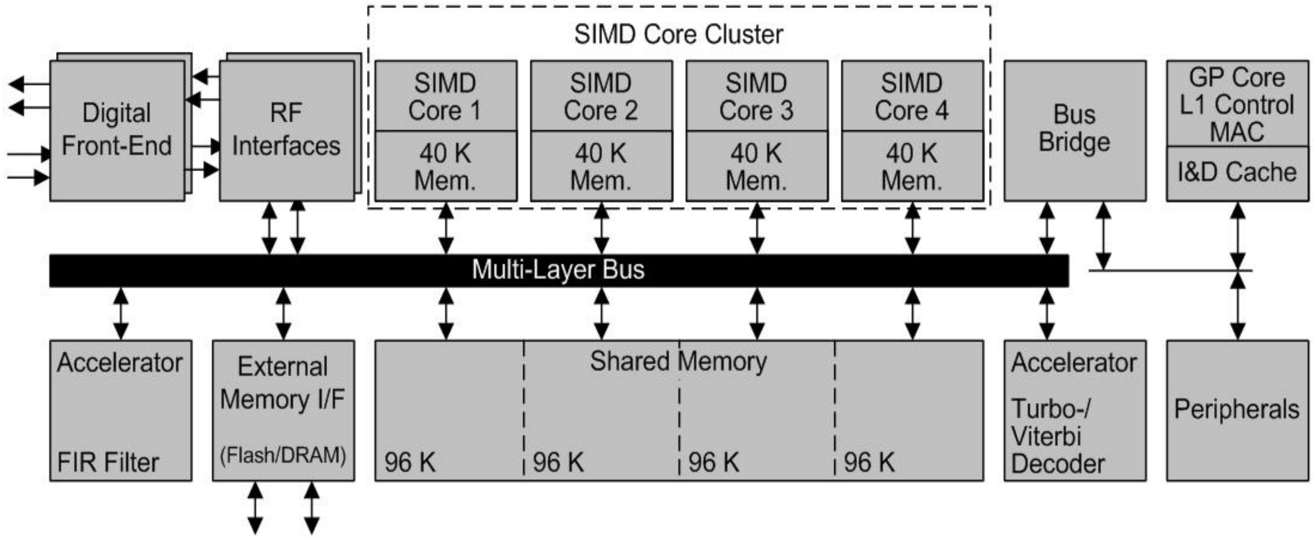

Figure 2: Baseband processing platform

2.1. SIMD core

The SIMD core (Figure 3) is based upon previous work [6] and has been simplified as well as extended for the application in communication systems. Each SIMD core contains four processing elements (PEs) 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 pipeline of the PEs' execution units is four stages long, which is used to relax the timing requirements for the memories, reduce the memories' supply voltage and thereby the power consumption. However, a long pipeline leads to stalls in case of data dependencies between instructions from the same task. That is why here each of the pipeline stages contains an instruction of one of four separate tasks. Four SIMD cores each running four tasks results in a task-level parallelism of 16.

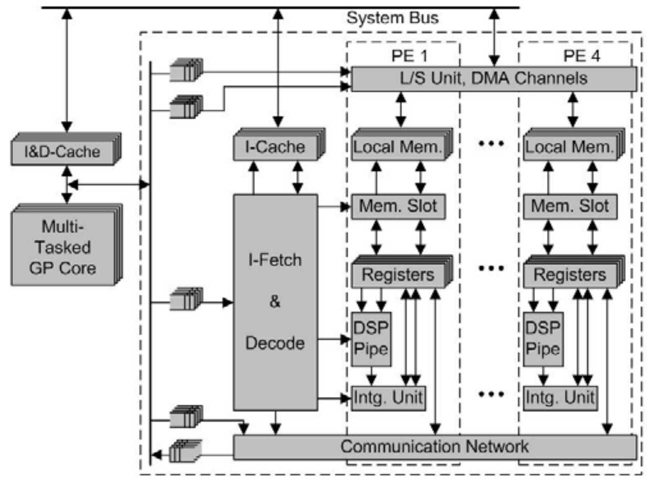

Figure 3: SIMD core block diagram

2.2. Accelerators

Table 1 and Table 2 show, respectively, parameters of channel coding algorithms and FIR filters used in different standards. The channel coding and filter processors must support all these parameters. 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. Figure 4 and Figure 5 show block diagrams of the channel coding and filter accelerators, respectively.

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.

Table 1: Channel coding parameters for selected standards

	Turbo Decoder		Viterbi Decoder		
	UMTS	CDMA 2000	UMTS	CDMA 2000	802.11a
max. data rate	384 kbps	307 kbps	64 kbps	38.4 kbps	24 Mbps
Code rate	1/3	1/3, 1/5	1/2, 1/3	1/2..1/6	1/2
number of trellis states	8	8	256	256	64
decoder type	SISO	SISO	hard decision or SOVA	hard decision or SOVA	hard decision
parallel butterflies	8	8	8	8	8
recursion direction	fwd, bck	fwd, bck	fwd	fwd	fwd
forward window size	~30	~30	–	–	flexible
backward window size	~30	~30	~45 (SOVA)	~45 (SOVA)	–
stopping criterion	yes	yes	no	no	no

Table 2: FIR filter parameters for selected standards

	filter length	data word length	coeff. word length	sample rate
WCDMA	19..25	8..10	8..10	7.68 MHz
WLAN 802.11	15..21	10..12	10..12	20 MHz / 22 MHz

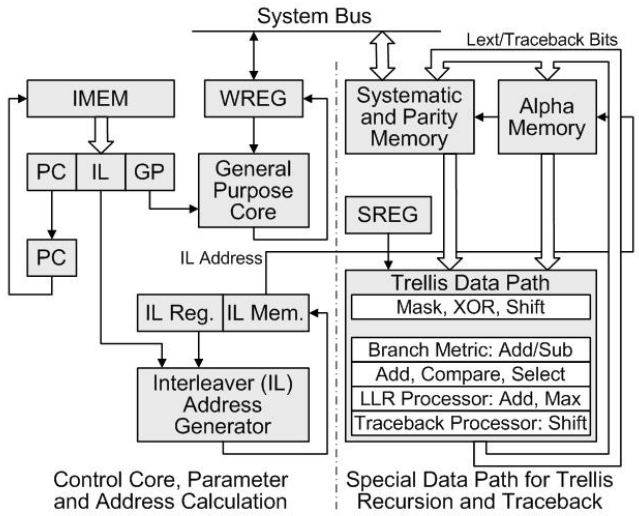

Figure 4: Channel coding accelerator

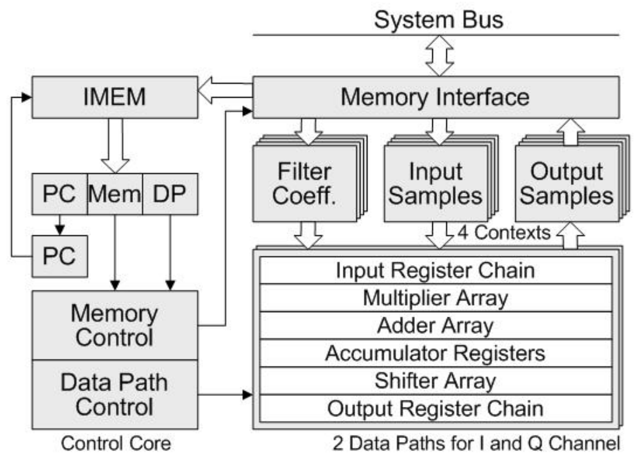

Figure 5: FIR filter accelerator

3. SOFTWARE ENVIRONMENT

It is important to note that the hardware platform architecture resembles a computer architecture. To keep the simplicity of the programming model of this SDR approach, a suitable software development environment is required that enables the programmer to exploit the potential of the hardware easily, however, without knowing its 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 SDR platform, each component bearing a research problem of its own.

3.1. Customer interface

The API provides access to functions from an 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 these functions, the API interface is the borderline for the intellectual property owned by the ODM and OEM, respectively.

3.2. SIMD compiler

The SIMD compiler is a crucial component of the SDR platform programming environment. It extracts data parallelism out of a C program and maps it onto a parallel processor. To support the extraction of data parallelism, the input to the compiler is C code with data-parallel C extensions (DPCE). 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.

3.3. Scheduling

The SIMD compiler extracts data parallelism out of a single piece of program and maps the program 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 [1]. Our approach is to start with a manual partitioning and scheduling of the representative standards that will be part of the SDR demonstrator. From this, a methodology and heuristics shall be derived for the construction of a scheduling tool.

3.4. Real-time operating system

The application software requires a real-time operating system (RTOS) running on the layer-1 controller core and on each SIMD core. The RTOS contains all 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 prior to execution as was described before. Nevertheless, it would be more flexible if the task schedule were determined during run time so that the system can react automatically to varying load conditions. 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 develop a methodology to generate the necessary operating system and device drivers automatically.

3.5. System design and virtual prototyping

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 coarse-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 baseband chip.

In this context, it is also necessary to perform simulation of the virtual prototype with reasonable speed. The options under consideration are the distributed simulation on parallel workstations or the mapping onto a prototyping hardware based on FPGAs.

4. 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. In the 802.11b standard the most critical timing is the short inter-frame space (SIFS), in particular. Figure 6 illustrates the SIFS time. Disregarding the time the signal takes through the RF stages of a terminal the SIFS denotes the time from the last sample of a frame coming into the baseband through the ADC until the first sample of an acknowledgement frame is sent to the DAC. The profiling was carried out in two steps. Firstly, the SIFS time was distributed equally over all functions contained in the signal processing chain. Secondly, each function was profiled in detail on the cycle-accurate simulator in order to find out all optimization potential both in hardware and software.

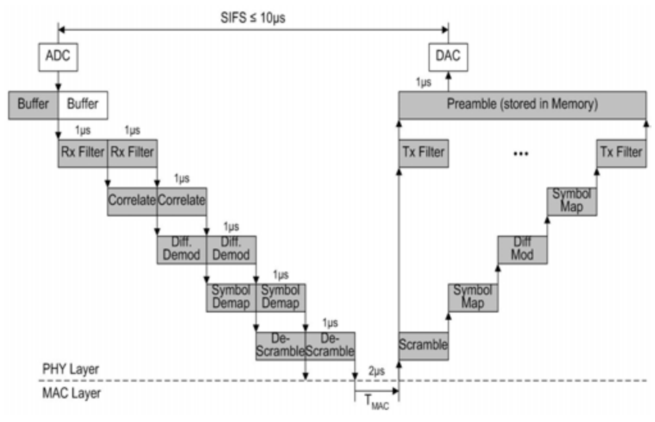

Figure 6: Illustration of the short inter-frame space (SIFS)

The detailed profiling showed the following results.

1. Using a virtual prototype for the detailed profiling is crucial for hardware optimization because it allows for short iteration cycles.
2. The overhead for synchronization between tasks running in different threads is very costly.
3. Mapping of tasks onto the hardware has to be done very carefully. It should be supported by an automatic tool in order to find an optimum mapping which fulfils the throughput and latency requirements.

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 standard in terms of complexity.

5. CONCLUSION

The next few years will see a transition from dual-band single-standard to multi-band multi-standard terminals. This revolutionary innovation will drive the further evolution of baseband processing. Flexibility and simplicity of the programming model turn out to be the decisive design criteria for the baseband architects. 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. Hence, software-defined radio technology is a key enabling technology for future cognitive radios. These will have to be able to utilize otherwise unused spectrum by transmitting in temporal or spectral gaps and to recognize and support a variety of single standards. A respective research project including mobile carriers, OEMs and universities is in preparation.

6. ACKNOWLEDGEMENT

The authors wish to thank N. Bruels, U. Hachmann, D. Langen, M. Loew, M. Richter, M. Sauermann, and A. Schackow for their work and contributions to this project.

7. REFERENCES

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

[2] J. Glossner et al., "A Software-Defined Communications Baseband Design", IEEE Communications Magazine, Jan. 2003.

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

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

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

[6] 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.