

SOFTWARE RADIO APPROACH FOR RE-CONFIGURABLE MULTI-STANDARD RADIOS

Jörg Brakensiek¹, Bernhard Oelkrug¹, Martin Bücken¹, Dirk Uffmann¹, A. Dröge¹, M. Darianian¹, Marius Otte²

¹Nokia Research Center, Meesmannstr. 103, 44 807 Bochum, Germany, jorg.brakensie@nokia.com
²University of Dortmund, Information Processing Lab, 44221 Dortmund, marius.otte@uni-dortmund.de

Abstract - next generation wireless systems will lead to an integration of existing networks, forming a heterogeneous network. Re-configurable systems will be the enabling technology sharing hardware resources for different purposes. This paper will highlight the requirements of a re-configurable multi-standard terminal from the physical-layer point of view. A re-configurable architecture consisting of algorithm domain specific accelerators, allowing autonomous complex digital signal processing without interference from a microprocessor, will be explained. Performance comparison numbers with latest Digital Signal Processors will show the effectiveness of the proposed architecture.

Keywords - Software Defined Radio, multi-standard, re-configurable, terminal, heterogeneous network, ASIP, hardware accelerator, SoC, classes of algorithms

I. INTRODUCTION

Future wireless communication systems will allow the separation of transfer requests over a wireless channel from the used transmission access technology. A wide range of communication systems and standards has been introduced, which can be organised in a layered structure as shown in Figure 1.

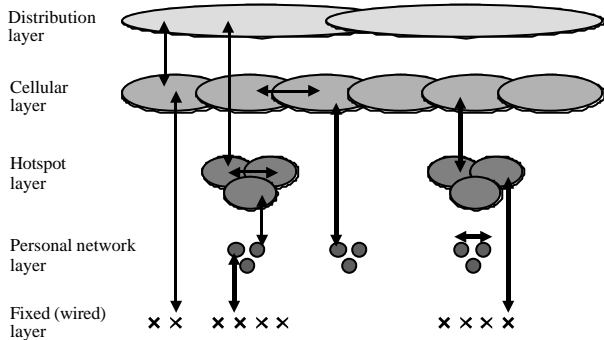


Figure 1: Heterogeneous Network Structure [1]

These layers provide a hierarchical view on the achievable network structure.

- Distribution layer: global coverage, with full mobility and large cells, downlink communication
- Cellular layer: full coverage, full mobility, global roaming, individual communication links
- Hot spot layer: local coverage, restricted mobility, individual high data rate links
- Personal network layer: short-range coverage, restricted mobility, point-to-point communication links

- Fixed (wired) layer: point access, high data rate links
- In a heterogeneous network scenario, systems are connected by implementing handover procedures, which can either be horizontal, i.e. between systems belonging to the same layer, or vertical, i.e. between systems belonging to different layers. The different handover possibilities are demonstrated in Figure 1 with horizontal and vertical arrows.

These heterogeneous network architectures require new mobile terminals and terminal architectures. In chapter II the requirements and constraints of this new terminal will be presented. Chapter III will explain the proposed terminal architecture based on a software defined radio approach. A demonstrator platform is described in chapter IV.

II. RE-CONFIGURATION OF A MULTI-STANDARD TERMINAL

Heterogeneous network architectures require systems, which are supporting more than one access technology. Inside such a network, any user, application or service will be able to choose the network resource based on parameters, which are important for the current transfer request:

- available and supported access technologies
- guaranteed bandwidth
- supported mobility
- access time
- transfer costs (€/Bit)
- needed power consumption (pJ/Bit)

Terminals may use the data transfer possibilities in a very flexible manner. The data transfer itself is handled transparently; i.e. the transfer is done using one or more different access technologies, invisible for the user.

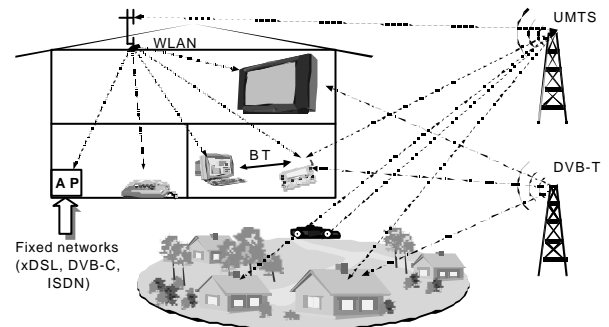


Figure 2: Heterogeneous Network Application Scenario

A heterogeneous network scenario is depicted in Figure 2, showing a mobile multi-standard terminal (e.g. PDA), linked to different networks, providing the following exemplary services:

- store data on a file server via Bluetooth
- transfer videos to the TV via Wireless LAN
- send a note to a friend via UMTS
- download broadcast IP data via DVB-T

A. Re-configuration Options

Re-configuration towards different standards can be done on different timelines, i.e. the re-configuration strategy is heavily dependent on the number of expected re-configurations during the product lifetime and the time period between two consecutive re-configurations. Different levels of re-configuration can be defined [3]:

- *Commissioning*: Configuration is done only at the time of product shipping, this would be more a configuration than a true re-configuration
- *Re-configuration with downtime*: Re-configuration is done only a few times during the product lifetime and will take some time, where the system is switched off
- *Re-configuration on a per call basis*: Re-configuration is done dynamically on a per call based decision, without any major downtime
- *Re-configuration per timeslot*: Re-configuration is done highly dynamically, with very fine time granularity, i.e. re-configuration will be done often during a single call

The flexible usage of multi-mode terminals will require a re-configuration on a per call basis, which will be done invisible for the user. Re-configuration on a time slot basis is not realistic, as this would require synchronisation of the entire heterogeneous network. Re-configuration with downtime will restrict the usage and is comparable with exchangeable modules (e.g. PCMCIA cards).

B. Re-configuration Architectures

The main basic re-configuration architectures have been identified leading to a new architecture approach, as shown in Figure 3.

1) One bit re-configuration

The simplest re-configuration would be to switch between standards on a very high level. This would require an independent baseband implementation for each system. At some place the incoming data stream is multiplexed to the dedicated baseband processing chain and the processed data is given back to the data sink. The amount of needed information for re-configuration is at minimum one single bit, defining the multiplexer behaviour.

Though this approach may be good enough for some dedicated basestation applications, it is not applicable for terminal applications as power consumption and area efficiency are the dominating factors for high volume, mobile devices.

2) Software Defined Radio

The most flexible re-configurable architecture is a Software Defined Radio (SDR) one [2], where the same device can be re-programmed to support various standards.

Software defined radio architectures are building a general-purpose communication platform for existing wireless access technologies, including future extensions and enhancements. Therefore the hardware platform has to provide sufficient processing power and communication bandwidth for future needs.

The openness of this approach concerning enhancements and new features results in an overhead of available resources, which is infeasible from a terminal point of view. In addition, high-bandwidth communication systems such as 54 MBit/s Wireless LAN would require a massively parallel implementation of microprocessor and DSP resources in order to fulfill the performance requirements. They would contribute mainly to the power consumption and area overhead of the system.

3) Re-configuration by parameterisation

Therefore we propose a new terminal approach, in which similarities and differences between the standards will be identified and parameterised. The basic algorithms, underlying the systems, have been evaluated and classes of algorithms with their corresponding parameter sets have been identified. Dedicated hardware will be developed now for the common baseband structures. Differences are accommodated by reconfiguration based on the defined parameters. Re-configuration capabilities will be provided only as much as needed, but as few as possible. This approach will be named *Software Re-configurable Radio (SRR)*.

The system architecture, which is underlying this SRR, has to be scalable and modular in order to allow future extension of the system, without changing the main part of the system architecture itself. Therefore, SRR can be implemented efficiently and power optimised.

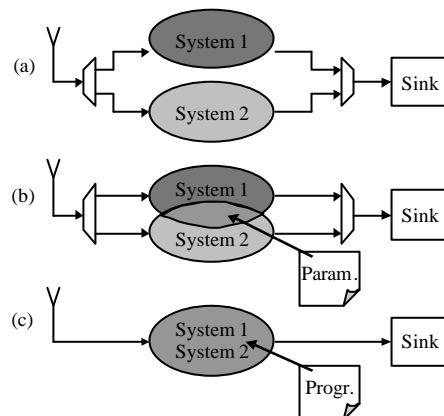


Figure 3: Re-configuration Architectures (2 standards)
 (a) One bit re-configuration, (b) Re-configuration by parameterisation (c) Software Defined Radio

III. MULTI-STANDARD ARCHITECTURE

The digital baseband architecture of a software re-configurable terminal, e.g. for UMTS and Wireless LAN has to deal with the following algorithmic challenges:

- Different modulation schemes: OFDM for Wireless LAN and W-CDMA for UMTS
- Different data rates, up to 54 Mbit/s for Wireless LAN and 2 Mbit/s for UMTS

Re-configuration of the digital baseband in this case means using the same hardware platform for both standards, with as little standard specific (i.e. parallel) hardware as possible.

A. Approach for Re-configuration

Different technologies are available supporting re-configuration:

- *Digital signal processors (DSP)*: Even though the achievable clock frequency is very high, the overall speed is not sufficient to handle complex high-speed algorithms. For data transport application the general purpose ALUs and MACs are not used efficiently and also for dedicated data processing applications only specific instructions are needed. Even more, DSPs are not very much scalable.
- *Field programmable gate arrays (FPGA)* provide a lot of flexibility but the achievable speed is restricted and power consumption is high. For data transport applications most of the logic resources are not used, whereas the routing resources are not used efficiently in data processing applications.

The approach, which is followed in this paper, can be best described as an algorithm specific instruction set processor (ASIP). Normally ASIP means application specific instruction set processor, but as algorithms are defining the processing complexity of an application, algorithm is the more suitable description of the intended functionality. In contrast to a DSP, these ASIPs are implementing only a limited instruction set, but since the instruction set is adapted to the algorithm needs, it will be used very efficiently. The architecture will allow easy scalability towards the needed system performance.

The intention of the architecture is to free the microprocessor and/or the DSP from doing complex and performance intensive calculations. Therefore the ASIP can be seen as an independent accelerator attached to the system bus, as depicted in Figure 4. The accelerator will do these operations, running autonomously without interaction from the microprocessor/DSP. This approach can be scaled to a multitude of accelerators as needed.

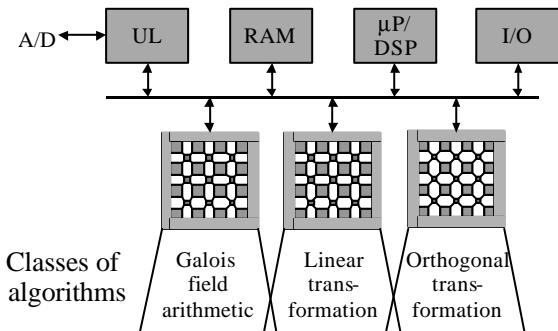


Figure 4: Re-configurable Architecture

B. Concept of a Field of Processing Elements

The concept to accelerate complex algorithms can be derived by looking at a relatively simple example as shown in Figure 5, which represents the signal flow graph of an 8-point Fast-Fourier-Transformation (FFT).

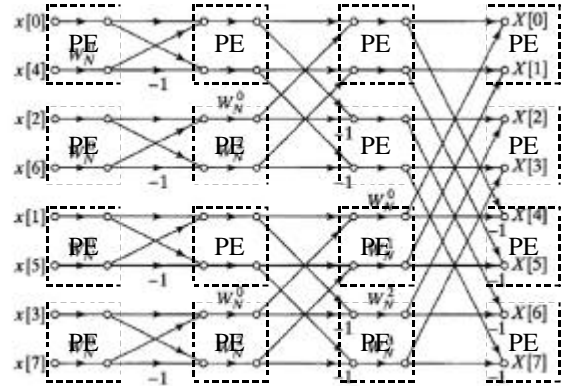


Figure 5: FFT Signal Flow with Atomic Operations

It is obvious that this signal flow graph consists of many atomic but common operations, which are in this case complex additions/subtractions and a complex multiplication. These common operations can be grouped together in a single processing element (PE). The PEs of this 8-FFT signal flow graph are shown in Figure 5. The interconnection scheme between the PEs is showing the typical butterfly structure.

A first accelerator approach, which is based on interconnected atomic operations, would be the implementation of a field of processing elements. Performance increase compared to a general purpose DSP could be achieved through the massive parallel implementation.

This approach is suffering from the following problems:

- Full flexibility of the PE interconnection scheme is highly complex in order to allow local and global interconnections of each PE output with each PE input.
- Size of the PE field is dependent on the size of the algorithm (changes of the algorithm parameter will lead to an inefficient usage of the whole field of processing elements).
- Huge I/O bandwidth is needed in order to keep the whole field running.

C. Concept of a Virtual Field of Processing Elements

To overcome the above described restrictions the accelerator models the complete field of processing elements as a *virtual* field and implements only a limited number of physical processing elements. This set of physical PEs subsequently computes the complete virtual field. The whole virtual field is calculated after N_{sweep} calculations (a so-called sweep), which is given in the following equation:

$$N_{\text{sweep}} = \frac{\text{Virtual field size}}{\text{Number of physical PE}} \quad (1)$$

Assuming pipeline stages on the interconnection lines between every two PEs, the order of execution does not have any influence on the computation result. These intermediate results, algorithm input and output values are stored in a central memory block (Data RAM). In a single sweep the PEs are fed from the Data RAM out of the *active*-memory banks and at the same time the PE results are stored in the *shadow*-memory banks. After every complete sweep the functionality of both memory banks is swapped. With this mechanism no intermediate value is overwritten. The read addresses for the input values of the PEs are provided by a configuration memory. Write addresses are incremented in a pre-determined fashion from a finite state machine (FSM) controller. Throughout these memory operations, the interconnection scheme of the virtual field of processing elements is accomplished by memory addressing. Even complex interconnection schemes can be realised and re-configuration of this interconnection scheme can easily be done through exchange of the Configuration RAM content. Figure 6 shows the accelerator architecture.

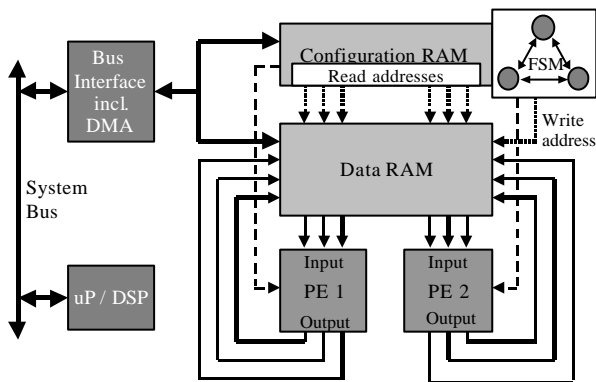


Figure 6: Hardware Accelerator Architecture

Beside their data input, the processing elements have additional configuration inputs connected to the Configuration RAM. They are used inside the PE to switch between different modes, allowing different operations within the same PE. Therefore the content of the Configuration RAM can be seen as the execution code (microcode) of the ASIP. The configuration vector is changed on a clock cycle basis, allowing very dynamic re-configuration within the processing elements.

The number of processing elements for an accelerator is not limited by the architecture. It is scalable in order to support as much processing elements as needed for the performance requirements of the dedicated algorithms.

A DMA capable bus interface allows to autonomously load data into the Data RAM and to store the data results into any memory location on the attached bus. The accelerator is activated externally by e.g. the microprocessor. Both together, the Configuration RAM and the FSM, then manage an equivalent to multiple different subprogram executions and loops, as they can be found in standard microprocessors. After the completion of the processing, which may last one or more sweeps, an interrupt is asserted.

D. Performance Comparison

Performance comparisons to leading edge DSPs have proven that the algorithms using specialised PEs are computed much more cycle effective and the scalable architecture allows even greater speedups compared to DSPs with only few execution units. [4]

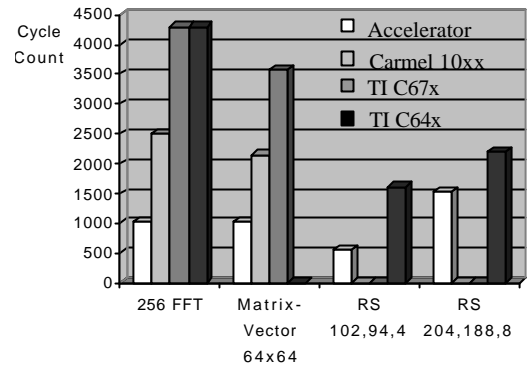


Figure 7: Performance Comparison

The Reed-Solomon Codec for the TI C64 shows the C-entry performance. The accelerator in this comparison has 4 PEs for the RS and two PEs for the FFT and matrix-vector-multiplication. Performance numbers are taken from the DSP data sheets (as far as available).

E. Comparison with Other Existing Architectures

Due to the increasing demand for re-configurable signal processing platforms, many different approaches have been and are still investigated in university and commercial research projects and have led to different accelerator architectures. The most important criterion for a comparison with our proposal is the functionality of the re-configuration units.

Due to the regular structure of data paths in DSP applications, most data flow machines have coarse-grained re-configurable interconnections. On the other hand, granularity, number and re-configurability of execution units vary widely. Typical SoC oriented re-configurable architectures connect an on-chip general-purpose processor core with coarse-grained re-configurable units:

- The *Chameleon Re-configurable Communications Processor* uses a 32-bit re-configurable fabric that serves as the reconfiguration area and is connected to an on-chip ARC RISC core. The fabric is divided into a dozen tiles, each of it consisting of local memory blocks, re-configurable data path units, 2 16x24 multipliers and a control unit.
- The *Xtensa Configurable Processor* is basically a general-purpose processor with an extendible instruction set. Furthermore it can also be extended by soft IP-cores that are tailored for the specific application domain.
- The *MorphICs Dynamically Re-configurable Architecture* is organised as a series of re-configurable

kernel processing elements controlled by a general-purpose processor. Each re-configurable element is also coarse-grained and data path oriented

Compared to them our approach is less general but also less complex. Due to the restriction to one class of algorithms (each class corresponds to one PE implementation) we get a better area efficiency and less power consumption. This is due to the fact that the developed building blocks meet the precise needs of the applications the system is designed for. The accelerator concept proposed in this paper has an additional advantage over the commercial products. Due to the separation of arithmetic blocks (PEs) and data path re-configuration (Data and Configuration RAM) and due to the fixed and simple interface of the PEs, it is quite easy to develop and embed new arithmetic units, optimised for a limited class of algorithms. In contrast to the other data flow machines we do not use dedicated configurable switching elements to manage data paths, but we use a very flexible memory interface that virtually connects the PEs. Therefore we achieve quasi-unlimited routing capabilities and almost 100% utilisation of the arithmetic units.

IV. DEMONSTRATOR

To proof the proposed architecture, a demonstrator platform has been developed using the ARM Integrator/AP AHB ASIC Development Platform. The platform contains:

- One core module with the ARM920T microprocessor core, AHB system bus bridge, an interrupt controller and memory
- Up to four logic modules with Virtex2000E FPGAs, memory, and a full connection to the AHB

The entire demonstrator system is shown in Figure 8.

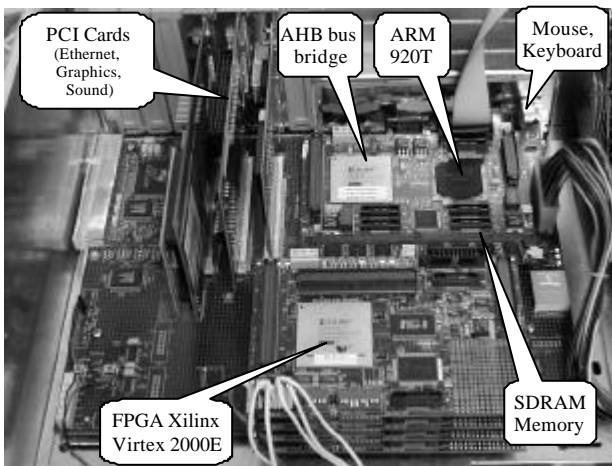


Figure 8: Demonstrator Platform

The accelerators are implemented on the FPGA modules, connected via the AHB to the ARM920. Accelerators for three different classes of algorithms have been developed and implemented so far, which can be used for the following algorithms or applications:

- FFT/IFFT [5]
- Beamforming

- Reed-Solomon coding [6]

The demo application is running on the ARM with Linux as the operating system. Custom made kernel modules allow easy access for any application to the accelerators embedded into the FPGA.

V. CONCLUSION

The demand for multi-standard system solutions has increased the need for re-configurable terminal architectures. In order to increase the flexibility of the system, without adding to much re-configuration overhead, a concept of accelerators has been proposed, which are targeted and optimised for a class of algorithms.

The proposed accelerator architecture allows re-configuration on different levels.

- PE level: Dynamic re-configuration of the PE using the configuration input.
- Accelerator level: Slow dynamic re-configuration of the accelerator, changing the Configuration RAM content.
- SoC level: Static re-configuration of microprocessor execution code, defining the accelerator usage.

VI. ACKNOWLEDGEMENTS

This project is partly funded from the German Ministry of Education and Research (BMBF) in the "Mobile, Re-configurable Multi-standard Terminal for UMTS and WLAN (MoReTeX)" project, which is a part of the umbrella "Software Defined Radio Based Architecture Studies for Re-configurable Mobile Communication Systems (RMS)".

VII. REFERENCES

- [1] R. Becher, M. Dillinger, M. Haardt, W. Mohr, Broad-Band Wireless Access and Future Communication Networks. Proceedings of the IEEE, Vol. 89, No 1, January 2001.
- [2] J. Mitola, The Software Radio Architecture. IEEE Communications Magazine, May 1995.
- [3] J. Brakensiek, R. Wittmann, M. Darianian, Software Defined Radio Technology for Multstandard Terminals, 2nd Karlsruhe Workshop on Software Radios, March 2002.
- [4] B. Oelkrug, M. Bucker, D. Uffmann, A. Dröge, J. Brakensiek, M. Darianian, Programmable Hardware Accelerator for Universal Telecommunication Applications, 2nd Karlsruhe Workshop on Software Radios, March 2002.
- [5] H. Lange, M. Bucker, O. Franzen, H. Schröder: *Reconfigurable Multiply-Accumulate-Based Processing Element*. IEEE Workshop on Heterogeneous reconfigurable Systems on Chip, Hamburg, April 2002.
- [6] S. Roy, M. Bucker, W. Wilhelm, B. S. Panwar: *Reconfigurable Hardware Accelerator for a Universal Reed-Solomon Codec*. 1st IEEE Int. Conference on Circuits and Systems for Communication, St. Petersburg, June 2002.