[195]

# Reconfigurable Processor Architectures for Mobile Phones

Martin Vorbach

PACT XPP Technologies AG
Muthmannstr. 1
D-80939 Munich, Germany
martin.vorbach@pactcorp.com

Jürgen Becker

Universitaet Karlsruhe (TH)
Institut fuer Technik der Informationsverarbeitung
D-76128 Karlsruhe, Germany
becker@itiv.uni-karlsruhe.de

**Abstract**

*This paper describes a new dynamically Configurable System-on-Chip (CSoC) concept and integration, consisting of an ARM7 EJS processor-core, a coarse-grain 4x4 XPP-array from PACT XPP Technologies AG, and application-tailored global/local memory topology with efficient Amba-based communication interfaces. The system and introduced CSoC architecture is optimized for mobile communication algorithm scenario. The paper gives an overview on the overall system concept, the hardware datapath structures and their integration as well as discussing some selected application implementation results within this target area.*

## 1 Introduction and Motivation

Systems-on-Chip (SoCs) has become reality now, driven by fast development of CMOS VLSI technologies. Complex system integration onto one single die introduce a set of various challenges and perspectives for industrial and academic institutions. Important issues to be addressed here are cost-effective technologies, efficient and application-tailored hardware/software architectures, and corresponding IP-based EDA methods. Due to exponential increasing CMOS mask costs, essential aspects for the industry are now flexibility and adaptivity of SoCs. Thus, in addition to ASIC-based, one new promising type of SoC architecture template is recognized by several academic [2] [16] [17] [18] [19] [20] and first commercial versions [4] [5] [6] [8] [10] [11] [13]: Configurable SoCs (CSoCs), consisting of processor-, memory-, probably ASIC-cores, and on-chip reconfigurable hardware parts for customization to a particular application. CSoCs combine the advantages of both: ASIC-based SoCs and multichip-board development using standard components [3].

This contribution provides the description of a CSoC project, integrating the dynamically reconfigurable eXtreme Processing Platform (XPP) from PACT [10] [11], [12] (see figure 3). The XPP architecture realizes a new runtime re-configurable data processing technology that replaces the concept of instruction sequencing by configuration sequencing with high performance application areas envisioned from embedded signal processing to co-processing in different DSP-like application environments.
The adaptive reconfigurable data processing architecture consist of following components:

- Processing Array Elements (PAEs), organized as Processing Arrays (PAs),
- a packet oriented communication network,
- a hierarchical Configuration Manager (CM) tree, and
- a set of I/O modules.

This supports the execution of multiple data flow applications running in parallel. A PA together with one low level CM is referred as PAC (Processing Array Cluster). The low level CM is responsible for writing configuration data into the configurable objects of the PA. Typically, more than one PAC is used to build a complete XPP device. Doing so, additional CMs are introduced for configuration data handling. With an increasing number of PACs on a device, the configuration hardware assumes the structure of a tree of CMs. The root CM of the tree is called the supervising CM or SCM. This unit is usually connected to an external or global RAM.
The basic concept consists of replacing the Von-Neumann instruction stream by automatic configuration sequencing and by processing data streams instead of single machine words, similar to [1]. Due to the XPP's high regularity, a high level compiler can extract instruction level parallelism and pipelining that is implicitly contained in algorithms

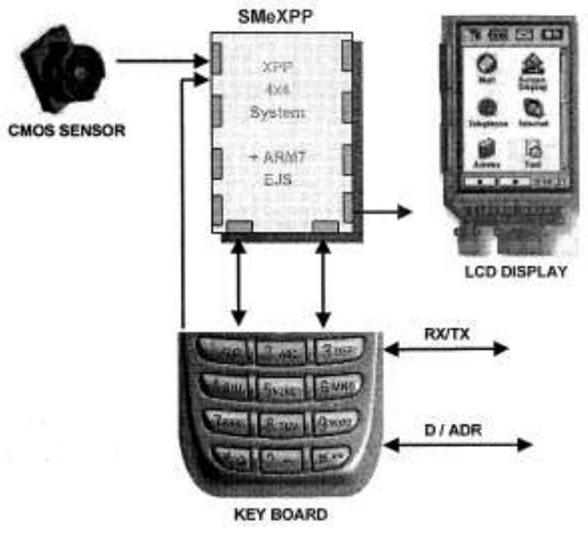

**Figure 1:** XPP System Integration - Overview

[12]. The XPP can be used in several fields, e.g. as image/ video processing, encryption, and baseband processing of next generation wireless standards, e.g. to realize also *Software Radio* approaches. 3G systems, i.e. based on the UMTS standard, will be defined to provide a transmission scheme which is highly flexible and adaptable to new services. Relative to GSM, UMTS and IS-95 will require intensive layer 1 related operations, which cannot be performed on today´s processors [14] [15]. Thus, an optimized HW/SW partitioning of these computation-intensive tasks is necessary, whereas the flexibility to adapt to changing standards and different operation modes (different services, QoS, BER, etc.) has to be considered. Therefore, selected computation-intensive signal processing tasks have to be migrated from software to hardware implementation, e. g. to ASIC or coarse-grain reconfigurable hardware parts, like the XPP architecture.

## 2 XPP-based CSoC Architecture

Our CSoC architecture (figure 2) consists of a 4x4 XPP-core from PACT, one ARM7 EJS processor-core, and different memory modules and interfaces. The main communication bus is chosen to be the AHB from ARM. The size of the XPP architecture will be 16 ALU-PAEs, a 4x4-array. To get an efficient coupling of the XPP architecture to AHB, suitable AHB-bridges are implemented which connects IO-interfaces of the XPP to the AHB bussystem.

The processor on the introduced CSoC version is a ARM7 EJS core. Additional modules can easily be added using the on-chip AMBA AHB/APB busses, whereas local memory modules on the CSoC is used to store the ARM programs, data for XPP computation and XPP configurations. The bandwidth of the AHB bus system can be scaled, e.g. dependent on the required throughput the clock frequency is variable from 25-100Mhz ain order to reduce the power consumption in an optimized way. The interface of the memory modules to the AHB is realized as a slave. That's because this module is a passive module only and can not start any kind of transactions on the AHB. Moreover, there will be external RAM interfaces implemented, which allows to connect external memory to the CSoC. In figure 2 the general concept for the Amba-based interface integration of the ARM/XPP/Memory modules is shown. In addition the ARM7 EJS and the XPP have accesses to private ROM/SRAM memories (not shown in figure 2), which reduces the traffic on the multilayer Amba-bussystem and provides more concurrency in the overal data transfer patterns. Each of the parallel operating high-throughput bridges connecting the different ARM/XPP/Memory modules can achieve a high and variable data throughput, which is sufficient for multimedia-based applications like MPEG-4 algorithms applied to video data in PAL-standard format (see section 4).
The communication between the CSoC and the outside world could be realized throught a master/slave AHB/PCI host bridge. The AHB master ability admits the direct transfers from PCI to internal RAM without involvement of the ARM7 EJS processor.

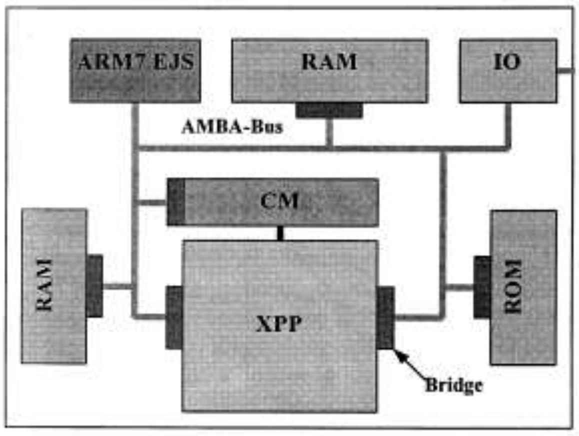

**Figure 2:** AMBA AHB to CM Interface structure

## 3 eXtreme Processing Platform - XPP

The XPP architecture is based on a hierarchical array of coarse-grain, adaptive computing elements called *Processing Array Elements (PAEs)* and a *packet-oriented communication network*. The strength of the XPP technology originates from the combination of array processing with unique, powerful run-time reconfiguration mechanisms. Since configuration control is distributed over several *Configuration Managers (CMs)* embedded in the array, PAEs can be configured rapidly in parallel while neighboring PAEs are processing data. Entire applications can be configured and run independently on different parts of the array. Reconfiguration is triggered externally or even by special event signals originating within the array, enabling self-reconfiguring designs. By utilizing protocols implemented in hardware, data and event packets are used to process, generate, decompose and merge streams of data. The XPP has some similarities with other coarse-grain reconfigurable architectures like the KressArray [21] or Raw Machines [22] which are specifically for stream-based applications. XPP's main distinguishing features are its automatic packet-handling mechanisms and sophisticated hierarchical configuration protocols.

### 3.1 Array Structure

An XPP device contains one or several *Processing Array Clusters (PACs)*, i.e. rectangular blocks of PAEs. Each PAC is attached to a CM responsible for writing configuration data into the configurable objects of the PAC. Multi-PAC devices contain additional CMs for configuration data handling, forming a hierarchical tree of CMs. The root CM is called the supervising CM or SCM. The XPP architecture is also designed for cascading multiple devices in a multi-chip. A CM consists of a state machine and internal RAM for configuration caching. The PAC itself contains a configuration bus which connects the CM with PAEs and other configurable objects. Horizontal busses carry data and events. They can be segmented by configurable switch-objects, and connected to PAEs and special I/O objects at the periphery of the device.
A PAE is a collection of PAE objects. The typical PAE shown in figure 3 contains a BREG object (back registers) and an FREG object (forward registers) which are used for vertical routing, as well as an ALU object which performs the actual computations. The ALU object's internal structure is shown on the bottom left-hand side of the figure. The ALU implemented performs common fixed-point

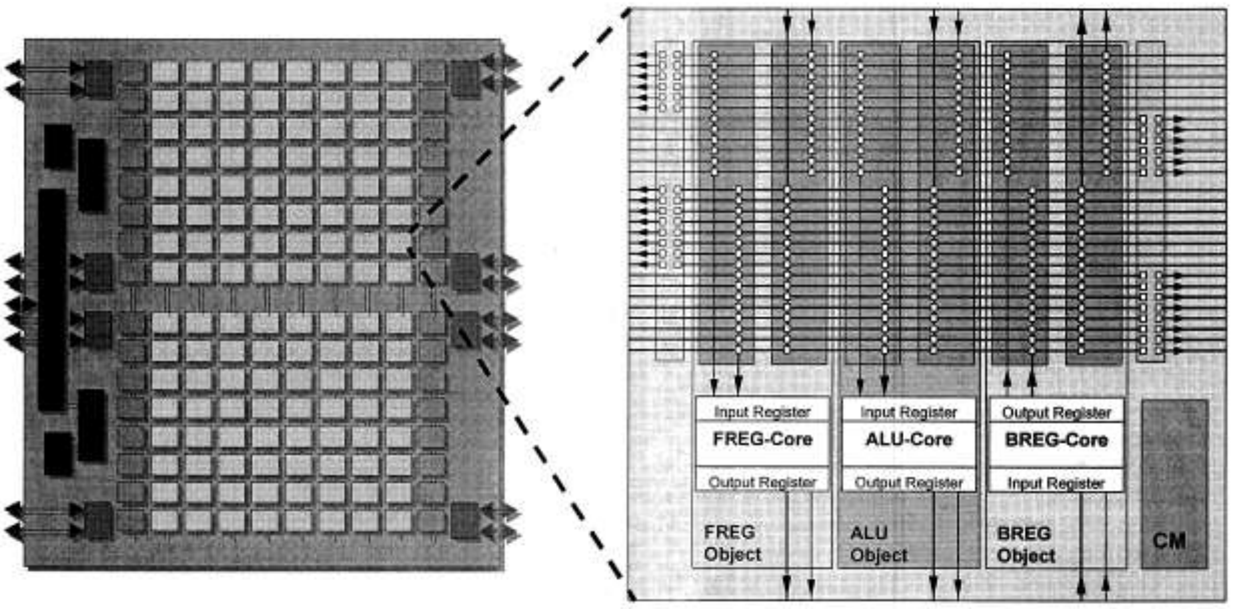

**Figure 3:** XPP64 Architecture Overview and Structure of one ALU PAE module

| Algorithm | XPP16 | DSP IP |
|---|---|---|
| MPEG Video 2D DCT (8x8) | 19 nWattSeconds | 51 nWattSeconds |
| Real 16 Tap FIR Filter (40 Samples) | 12 nWattSeconds | 49 nWattSeconds |
| 256-point FFT | 360 nWattSeconds | 453 nWattSeconds |

Frequency = 100 MHz

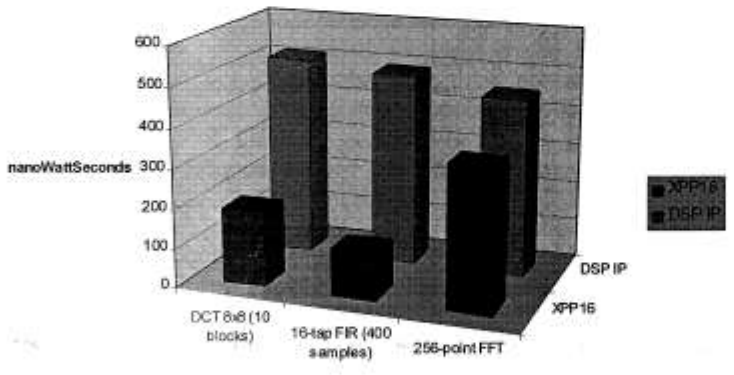

**Figure 4:** Energy Consumption Benchmarks Comparison of XPP16 and DSP Processor

arithmetical and logical operations as well as several special three-input opcodes like multiply-add, sort, and counters. Events generated by ALU objects depend on ALU results or exceptions, very similar to the state flags of a classical microprocessor. A counter, e.g., generates a special event only after it has terminated. The next section explains how these events are used. Another PAE object implemented in the prototype is a memory object which can be used in FIFO mode or as RAM for lookup tables, intermediate results etc. However, any PAE object functionality can be included in the XPP architecture.

## 3.2 Packet Handling and Synchronization

PAE objects as defined above communicate via a packet-oriented network. Two types of packets are sent through the array: data packets and event packets. Data packets have a uniform bit width specific to the device type.
In normal operation mode, PAE objects are self-synchronizing. An operation is performed as soon as all necessary data input packets are available. The results are forwarded as soon as they are available, provided the previous results have been consumed. Thus it is possible to map a signal-flow graph directly to ALU objects, and to pipeline input data streams through it. The communication system is designed to transmit one packet per cycle. Hardware protocols ensure that no packets are lost, even in the case of pipeline stalls or during the configuration process. This simplifies application development considerably. No explicit scheduling of operations is required. Event packets are one bit wide. They transmit state information which controls ALU execution and packet generation. For instance, they can be used to control the merging of data-streams or to deliberately discard data packets. Thus conditional computations depending on the results of earlier ALU operations are feasible. Events can even trigger a self-reconfiguration of the device as explained below.

## 3.3 Configuration

The XPP architecture is optimized for rapid and user transparent configuration. For this purpose, the configuration managers in the CM tree operate independently, and therefore are able to configure their respective parts of the array in parallel. Every PAE stores locally its current configuration state, i.e. if it is part of a configuration or not (states

„configured"'or „free"). If a configuration is requested by the supervising CM, the configuration data traverses the hierarchical CM tree to the leaf CMs which load the configurations onto the array. The leaf CM locally synchronizes with the PAEs in the PAC it configures. Once a PAE is configured, it changes its state to „configured". This prevents the respective CM from reconfiguring a PAE which is still used by another application. The CM caches the configuration data in its internal RAM until the required PAEs become available. Hence the CMs' cache memory and the distributed configuration state in the array enables the leaf CMs to configure their respective PACs independently. No global synchronization is necessary.
While loading a configuration, all PAEs start to compute their part of the application as soon as they are in state „configured". Partially configured applications are able to process data without loss of packets. This concurrency of configuration and computation hides configuration latency. Additionally, a prefetching mechanism is used. After a configuration is loaded onto the array, the next configuration may already be requested and cached in the low-level CMs' internal RAM. Thus it need not be requested all the way from the SCM down to the array when PAEs become available.

| | SMeXPP | Application Processor |
|---|---|---|
| Performance | 2000 MIPS | 500 MIPS |
| # MACs | 16 | 4 |
| Power | < 0,4 mW/MHz | 0,5 – 0,8 mW/MHz |
| Frequency | 52 MHz | 208 MHz |
| Battery Lifetime | > 4x | 1x |
| On Chip Memory | 4 MBit | 6 MBit |
| Off Chip Memory | 0 MB | 16 MB |

| | SMeXPP | Application Processor |
|---|---|---|
| Performance | 2000 MIPS | 500 MIPS |
| Chip Cost | 4 ... 6 | 6 ... 8 |
| External Memory | 0 | 16 ... 32 MB |
| Memory Cost | 0 | 8 ... 10 |
| Package | <100 I/O | >200 I/O |
| Package Cost Benefit | -1 | 0 |
| Cost Benefit | -11 ... 13 | 0 |

**Figure 5:** Technical and Commercial Trade-offs of SMeXPP CsoC

## 4 Mobile Communication Application - Examples Discussion -

PACT did in the year 2000 a first evaluation board based on 0.25 µm technology for their XPP 128 chips. Based thereupon, promising performance results compared to a parallel VLIW type DSP of Texas Instruments were obtained. [10], [11].
Within the application area of future mobile phones desired and important functionalities are gaming, video compression for multimedia messaging, polyphone sound (MIDI), etc. Therefore, a flexible, low cost hardware platform with low power consumption is needed for realizing neccesary computation-intensive algorithms parts. Thus, PACT implemented several of these functionalities onto the cost-efficient 4x4 XPP array size, e.g. a 256-point FFT, a real 16 tap FIR filter, and a video 2d DCT (8x8) for MPEG-4 systems. The last functionality will be discussed here. The energy consumption benchmarks comparisons between an XPP16 architecture and a state-of-the-art DSP Processor for the above mentioned funtionalities are given in figure 4. The SMeXPP combination of ARM-7 EJS and XPP with efficient RAM-topology promises also a high boost in performance and flexibility. The technical and commercial trade-offs of this SMeXPP solution is shown in figure 5. First digital TV application performance results were obtained by evaluating corresponding MPEG-4 algorithm mappings onto the introduced ARM/XPP CSoC and based on the 0.13 µm CMOS technology synthesis results. Based on this coarse-grain CSoC version, performance/cost results of an MPEG-4 application is currently under imple-

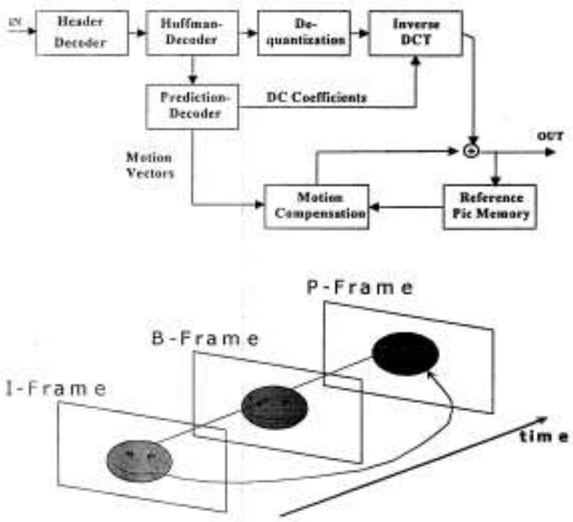

**Figure 6:** Main MPEG-4 Algorithm Modules

mentation, whereas the Inverse DCT (see figure 6) applied to 8x8 pixel blocks can be performed by an 4x4 XPP-Array in 74 clock cycles. Since the IDCT is one of the most complex operations in MPEG-4 algorithms, the preliminary clock frequency of 100 Mhz based on 0.13 µm CMOS technology integration is sufficient for this real-time digital TV application scenario.

## 5 Conclusions

The paper described the dynamically reconfigurable XPP architecture and coarse-grain hardware structures. The focus was given to its Configurable System-on-Chip (CSoC) integration concept, whereas an ARM7 EJS processor-core, a coarse-grain 4x4 XPP-array from PACT XPP Technologies AG, and application-tailored global/local memory topologies are interfaced with efficient Amba-based communication structures. The hardware/software system and introduced SMeXPP CSoC architecture is optimized for mobile communication algorithm scenario. The paper has given an overview on the overall system concept, the hardware datapath structures and their integration as well as discussing some selected application performance results within the area of mobile phones, e.g. the implementation discussion of a video 2d DCT (8x8) in MPEG-4 systems suitable for mobile multimedia messaging. From the explained technical and commercial trade-offs such a SMeXPP solution is outperforming traditional processor-based approaches, e.g. from performance, cost and risk minimization aspects.

## 6 References

[1] R. W. Hartenstein, J. Becker et al.: A Novel Machine Paradigm to Accelerate Scientific Computing; Special issue on Scientific Computing of Computer Science and Informatics Journal, Computer Society of India, 1996.

[2] J. Becker, T. Pionteck, C. Habermann, M. Glesner: Design and Implementation of a Coarse-Grained Dynamically Reconfigurable Hardware Architecture; in: Proc. of IEEE Computer Society Annual Workshop on VLSI (WVLSI 2001), Orlando, Florida, USA, April 19-20, 2001

[3] J. Becker (Invited Tutorial): Configurable Systems-on-Chip (CSoC); in: Proc. of $9^{th}$ Proc. of XV Brazilian Symposium on Integrated Circuit Design (SBCCI 2002), Porto Alegre, Brazil, September 5-9, 2002

[4] Xilinx Corp.: http://www.xilinx.com/products/virtex.htm.

[5] Altera Corp.: http://www.altera.com

[6] Triscend Inc.: http://www.triscend.com

[7] Triscend A7 Configurable System-on-Chip Platform - Data Sheet http://www.triscend.com/products/dsa7csoc_summary.pdf

[8] LucentWeb] http://www.lucent.com/micro/fpga/

[9] Atmel Corp.: http://www.atmel.com

[10] PACT Corporation: http://www.pactcorp.com

[11] The XPP Communication System, PACT Corporation, Technical Report 15, 2000

[12] V. Baumgarte, F. Mayr, A. Nückel, M. Vorbach, M. Weinhardt: PACT XPP - A Self-Reconfigurable Data Processing Architecture; The 1st Int'l. Conference of Engineering of Reconfigurable Systems and Algorithms (ERSA'01), Las Vegas, NV, June 2001

[13] Hitachi Semiconductor: http://semiconductor.hitachi.com/news/triscend.html

[14] Peter Jung, Joerg Plechinger., "M-GOLD: a multimode basband platform for future mobile terminals",CTMC'99, IEEE International Conference on Communications, Vancouver, June 1999.

[15] Jan M. Rabaey: System Design at Universities: Experiences and Challenges; IEEE Computer Society International Conference on Microelectronic Systems Education (MSE'99), July 19-21, Arlington VA, USA

[16] S. Copen Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, R. Laufer "PipeRench: a Coprocessor for Streaming Multimedia Acceleration" in ISCA 1999. http://www.ece.cmu.edu/research/piperench/

[17] MIT Reinventing Computing: http://www.ai.mit.edu/projects/transit dpga_prototype_documents.html

[18] N. Bagherzadeh, F. J. Kurdahi, H. Singh, G. Lu, M. Lee: "Design and Implementation of the MorphoSys Reconfigurable Computing Processor "; J. of VLSI and Signal Processing-Systems for Signal, Image and Video Technology, 3/ 2000

[19] Hui Zhang, Vandana Prabhu, Varghese George, Marlene Wan, Martin Benes, Arthur Abnous, "A 1V Heterogeneous Reconfigurable Processor IC for Baseband Wireless Applications", Proc. of ISSCC2000.

[20] Pleiades Group: http://bwrc.eecs.berkeley.edu/Research/Configurable_Architectures/

[21] R. Hartenstein, R. Kress, and H. Reinig. A new FPGA architecture for word-oriented datapaths. In Proc. FPL'94, Prague, Czech Republic, September 1994. Springer LNCS 849.

[22] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, and P. Finch. Baring it all to software: Raw machines. IEEE Computer, pages 86-93, September 1997

[23] ARM Corp.: http://www.arm.com/arm/AMBA