Overview of the 4S Project

Gerard Smit, Eberhard Schüler[*], Jürgen Becker[†], Jérôme Quévremont[‡], Werner Brugger[§]
contact University of Twente the Netherlands

Abstract — **In this paper an overview of the EU-FP6 "Smart Chips for Smart Surroundings" (4S) [7] project is given. The overall mission of the 4S project is to define and develop efficient (ultra low-power), flexible, reconfigurable core building blocks, including the supporting tools, for future ambient systems. Dynamic reconfiguration offers the flexibility and adaptability needed for future ambient devices, it provides the efficiency needed for these systems, it enables systems that can adapt to rapidly changing environmental conditions, it enables communication over heterogeneous wireless networks, and it reduces risks: reconfigurable systems can adapt to standards that may vary from place to place or standards that have changed during and after product development.**

Index Terms — **Coarse-grain reconfigurable, ambient systems, low power, SoC, DRM, MPEG-4.**

I. Introduction

The overall mission of the 4S project (Smart Chips for Smart Surroundings) is to define and develop efficient (ultra low-power), flexible, reconfigurable core building blocks for future ambient systems including the supporting tools. As an application we have chosen a concrete worldwide broadcast radio application (DRM) and MPEG-4 video that can be used in an ambient system scenario.

Ambient systems (also known as ambient intelligence or ubiquitous computing) are networked embedded systems intimately wirelessly integrated with everyday environments and supporting people in their activities. These systems will create a smart surrounding for people to facilitate and enrich daily life and increase productivity at work. It is likely that these systems will be quite different from current computer systems, and will have to be based on radically new architectures comprising a set of reconfigurable "building blocks" (IP blocks) and flexible interconnection mechanisms. These components often have conflicting requirements; they have to be flexible, adaptive as well as energy-efficient and low-cost.

Hence, the systems architecture of future ambient devices poses a lot of challenges: these devices have a very small energy budget, they are always operational (although quite often in a low-power mode), are small in size but might require a performance that exceeds the levels of current PDA computers. State-of-the-art processor architectures cannot provide the processing power required by a fully operational ambient device given the tight energy limitations. To realize ambient devices within the energy budget flexible and highly efficient architectures are needed. Moreover, without significant energy reduction techniques and energy-efficient adaptive architectures, battery-life constraints will severely limit the capabilities of these devices. The development of such architectures and supporting tools is at the core of the 4S project.

II. Heterogeneous Reconfigurable Computing

Reconfigurable systems offer the required flexibility and can adapt processing resources dynamically to the demand of applications. Our Heterogeneous Reconfigurable SoC (HR-SoC) consists of: bit-level reconfigurable tiles (e.g. embedded FPGAs), word-level reconfigurable tiles, and general-purpose programmable tiles (DSPs and microprocessors). The tiles are interconnected by a suitable NoC.

Typically, some algorithms are more suited for bit-level reconfigurable architectures (e.g. PN-code generation), others for DSP-like implementations and others for word-level reconfigurable platforms (e.g. FIR filters or FFT algorithms).

The programmability of the architecture enables the system to be targeted at multiple applications within the target domain. The reconfigurable tiles and firmware can be upgraded at any time (even when the system is already installed).

A HR-SoC combines performance, flexibility and energy-efficiency. It supports high performance through massive parallelism, it matches the computational model of the algorithm to the granularity of the processing entity, it can operate at minimum supply voltage and clock frequency and so provides energy-efficiency and flexibility at the right granularity only where needed and desirable.

Until recently only a few reconfigurable architectures have been proposed for wireless devices and to our knowledge none for ambient devices. Most reconfigurable architectures were targeted at simple glue logic or at dedicated high-performance computing. However, there are a number of reasons for using a HR-SoC in future ambient devices e.g.:

- Standards evolve quickly; this means that future systems have to have the flexibility and adaptivity to adapt to slight changes in the standards. By using reconfigurable architectures instead of ASICs costly re-designs can be avoided. For ambient devices standards are still under

[*] PACT Germany
[†] University of Karlsruhe Germany
[‡] Thales Communications, France
[§] Atmel Germany

development, therefore devices need to be highly flexible and adaptable to new developments. Downloadable reconfigurations (long-term reconfiguration) enable new or adapted services / standards on existing devices,

- The costs of designing complex ASICs is growing rapidly, in particular the mask costs of these chips is very high. With reconfigurable processors it is expected that less chips have to be designed (re-usability/adaptivity of silicon area), so companies can save dramatically on mask costs and due to higher production volumes the purchase costs for the end users will be reduced,
- The system can adapt to the environment and to the needs of the user rather than having the user adapt to a fixed system (the ambient intelligence approach). When the system can adapt - at run-time - to the environment significant power-saving can be obtained and less transmission and processing power is needed,
- Heterogeneous reconfigurable modules containing digital as well as analogue building blocks (e.g. programmable filter banks and multi-standard advanced synthesizer blocks, including fully integrated VCOs) will result in solutions for flexible on-chip RF front ends, covering a multitude of services for upcoming applications with one set of hardware,
- Finally, traditional (DSP) algorithms are rather static. The emergence of new ambient applications that require sophisticated adaptive, dynamical algorithms has drawn renewed attention to run-time dynamic reconfiguration.

The blocks cover both digital and mixed signal structures applicable to communication devices. Furthermore, we develop efficient NoCs to interconnect all tiles of the SoC.

The overall important characteristic is the life-time of a communication stream. We aim to develop a SoC for a multimedia terminal where we can assume that the data streams are semi-static and have periodic behavior. This means that for a long period of time subsequent data items of a stream follow the same route. This will last for seconds and more, because a user will listen to its radio or has a phone conversation for a considerable time. However, the control system might change some settings of processes due to changing environmental conditions.

According to the type of services required, the following types of traffic can be distinguished in the network:

- GT (*guaranteed throughput*) this is the part of the traffic for which the network has to give real-time guarantees (i.e. guaranteed bandwidth, bounded latency).
- BE (*best effort*) this is the part of the traffic for which the network guarantees only fairness but does not give any bandwidth and timing guarantees.

In our proposed NoC we support both GT and BE traffic.

A. Prototyping

Within the 4S project we are developing two prototypes. The first prototype called BCVP (Basic Concept Verification Platform) is constructed with off-the-shelf components. It contains a Dimitri chip [3] (containing two ARM9 cores, two DDCs, a Viterbi decoder and peripheral IO), ZBT DRAM, a Xilinx XC2V3000 or XC2V8000 FPGA and a PACT/XPP64-A1 [4]. The second prototype called HiCVP (Highly Integrated Concept Verification Platform) consists of a highly integrated SoC. This chip is in its definition phase and will probably contain one ARM926 core [6], boot memory, Viterbi decoder, two DDCs (Digital Down Converters), two A/D converters, peripheral IO, and four Montium tiles [5].

III. Application domain

A. DRM

As a key application for the 4S project we have chosen digital broadcast radio (in particular DRM) and MPEG-4 video. The DRM (Digital Radio Mondiale) standard [1][2] has been adopted by the ETSI at a European level and by the IEC (International Electrotechnical Committee) at a worldwide level. DRM offers digital radio broadcast in three frequency bands up to 30 MHz (long, medium and short waves).
DRM brings important improvements compared to existing analogue broadcasts in the above mentioned frequency bands: stereophonic sound, several audio services on one channel, FM-like sound quality without fading, low rate data transmission, additional services, ease of use for the listener, etc. Depending upon the frequency band, a transmitter can cover a region, a country or even reach any point in the world.

B. MPEG-4 video

MPEG-4 video is an ISO/IEC standard developed by the MPEG (Moving Picture Experts Group), the committee that also developed the standards known as MPEG-1 and MPEG-2. MPEG-4 builds on the proven success of three fields:

- Digital television;
- Interactive graphics applications (synthetic content);
- Interactive multimedia (World Wide Web, distribution of video and access to content)

In the 4S-project we will mainly concentrate on the visual part of MPEG-4.

IV. Design trajectory

Proper development tooling is essential for programmable devices. This is a major requirement for the system engineer to program the reconfigurable device.

Reconfigurable processors substantially reduce development cycles and costs normally associated with ASIC design, including nonrecurring engineering (NRE) costs, mask sets, fabrication runs, and perhaps most importantly, respins. However, controlling the development time and costs in a reconfigurable processor design requires a comprehensive set of tools – a design environment with a graceful flow from systems design to executable files that configure the reconfigurable architecture and a run-time system that maps the processes to processors.

In fact the availability of high-level design entry tooling is critical for the viability of any reconfigurable architecture. In the 4S project we will develop methods and techniques to

support the mapping of typical algorithms found in the ambient intelligence application domain onto heterogeneous reconfigurable architectures. These techniques have to identify the characteristic properties of the algorithm at hand and match these with the characteristic properties of the different target technologies (analogue, bit-level reconfigurable, word-level reconfigurable, programmable etc.).

Figure 1 shows the design trajectory of the 4S project, the compile-time design flow and the run-time flow (controlled by the RTOS). The design-time tool chain is based on existing tools for the tiles, with possible extensions. This allows the integration of the implementation results of the various tiles providing co-simulation, combined power estimation and performance characteristics. For each task, a set of precompiled functions with tile-specific characteristics concerning power, tile utilization and performance is provided.

At run-time the operating system (RTOS) dynamically selects the required task from the set. The decision which of the available tasks from the set is utilized is based on the actual needs of the application. The selection criteria can be current power constraints (e.g. low battery), utilization of resources of the hardware platform by other applications (e.g. the coarse-grained reconfigurable tile is currently utilized by another application) or user demands (e.g. user wishes higher audio quality). In this section we give an overview of the design methodology and of the existing and developed tools.

A. Task graph

The whole software trajectory starts with a high-level task-level system description of the application. We assume that the applications are written in C/C++ in terms of task graphs consisting of functional processes with standard 4S inter-process communication primitives.

As a first step in the design methodology, the application software, written in C/C++, can be simulated and validated on a functional verification platform. The advantage of this early simulation is that the overall application structure can be verified independently of the actual functional implementation. Note that by writing applications as communicating processes the programmer automatically does the (manual) partitioning.

B. Compiling individual functional processes to processors

Functional processes can be implemented on various hardware/software tiles (e.g. implementation of a 256pFFT on an ARM, on a Montium or on an embedded FPGA); the 4S inter-process communication primitives must be mapped onto specific NoC capabilities.

The next step is that individual functional processes are compiled or synthesized to appropriate processing tiles. Individual functional processes might have functional equivalent process implementations on more than one processing platform. At design-time all these implementations will be generated. At run-time the operating systems decides which implementation will be used depending on available tiles, QoS and energy constraints. The tools for compiling processes to processing tiles are not developed in this project, but we assume they are available. However, when small hardware changes require adaption of the tools, this will be done inside of the project, but it is not the main focus of the 4S project.

C. Annotation of process implementations

The in the previous step derived implementations of the processes will be annotated with performance characteristics (e.g. number of clock cycles, energy consumption, memory requirements, average load on a processing element). These performance figures are used by the run-time system to find the most optimal processing element for each process [9].

For most processing tiles there are tools available to derive the performance figures, but for other processing tiles the performance figures will be derived in the 4S project (either measured or derived from datasheets). Table 1 shows an example of process characterizations.

In addition to that the inter-process communication primitives have to be annotated with performance characteristics e.g. throughput [bits/sec], maximum latency [s],

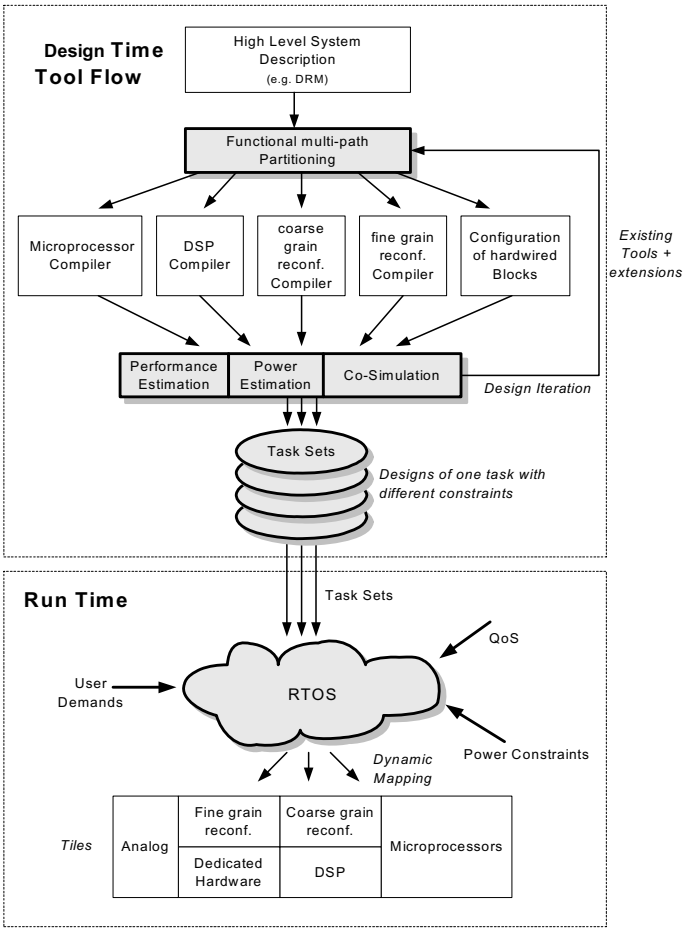

Figure 1: Software design flow and innovation (shadow: main issues for 4S)

etc. This annotation cannot be done automatically as some figures like energy consumption are data dependent.

Process	Processing tile	Performance characterization			
		[mW/ MHz]	Clock cycles	Memory [bytes]	Load [%]
Process A (Correlation)	ARM9	1	100	1k	50 %
	Montium	0.1	50	2k	100 %
	FPGA	0.4	30	2.5k	10 %
Process B (FFT)	ARM9	3	800	10k	60%
	Montium	0.5	200	10k	100%
Process C (Viterbi)	ASIC	0.05	10	2k	100%
	ARM9	0.9	300	15k	10%
	Montium	0.3	200	5k	50%
	FPGA	0.1	75	8k	25%

Table 1: Example of performance characterization

Modeling the overall system is beneficial and necessary for the project in multiple ways. Simply relying on existing tools is not enough for the highly heterogeneous system envisaged; the close co-operation of vastly different modules and the development of algorithms for optimization of application mapping with respect to a set of parameters (energy consumption, QoS, etc.) can be considerably eased by the use of a comprehensive system model that combines characterizations provided by the aforementioned tools.

As modeling language, SystemC is used because of its flexibility and the wide range of abstraction levels covered. It is suitable for high-level interface definition as well as low-level (nearly) hardware accurate modeling.

D. Run-time tools

To support run-time adaptive behaviour, trade-offs between different parameter sets should be made to determine the most optimal set for the current situation. In the 4S project we introduce a run-time control system, which is based on a model that selects at run-time a set of parameters that minimizes the cost, while satisfying the requested quality.

The run-time system consists of a collection of tools that are controlled by a distributed operating system called OSYRES [8]. The task of OSYRES is to start a new application graph by allocating processes to processing elements and application channels to NoC links. Finding the right processor for a certain process and finding the appropriate communication path is performed by the spatial mapping tool (SMIT) [9]. Based on the result of SMIT, OSYRES will install the required processes (which might mean reconfiguration or program re-loading) and will initiate the right communication mechanisms.

This instantiation of an application is performed when an application is started, however, when certain events happen the mapping might be reconsidered and/or communication links might be rerouted. Events that might trigger a (re-)mapping could be:

- the user starts an extra application that needs to be mapped on processing tiles,
- the user decides to kill an application which frees its occupied processing tiles,
- the QoS of the wireless link might change and therefore extra functionality (e.g. extra filtering) has to be performed that needs extra processing resources,
- the user might want to listen to another broadcast station that happens to use another set of parameters, and therefore the baseband processing tasks have to be updated.
- on a regular interval the system could do a test whether the current mapping is still sufficiently optimal.

Changing of the mapping can mean that an entire application graph needs to be removed and replaced by another graph or that only a single process in a process graph is changed or moved to another tile.

V. First Results and Conclusion

Currently the first prototype of 4S (BCVP) is operational. The OSYRES operating system is running on the two ARM9 cores. A first FPGA board is also operational. The FPGA board can be used for functional verification of sub-modules (e.g. the Montium core and the NoC), but can also be used as an interface to the PACT/XPP. The Montium is running on the FPGA at 9 MHz, and shows the same results as predicted by the RTL simulation. The specification and design of the HiCVP chip is work in progress, and will be finalized end 2005.

It is envisaged that in the long run, work performed within this project will lay the foundations for the development of a new range of ultra low-power components, architectures, tools, guidelines and standards that underpins the future development of ambient systems.

Acknowledgment

This work is part of the 4S project that has been supported by the Sixth European Framework Programme under project number IST 001908.

References

[1] "Digital Radio Mondiale (DRM); System Specification", ETSI, ES 201 980 V2.1.1, Nov. 2003.
[2] http://www.drm.org
[3] J. Quévremont, M. Sarlotte, B. Candaele, "Development process of a DRM digital broadcast SoC receiver platform", Annales des Télécommunications, Sept-Oct 2004.
[4] "XPP64-A1 Reconfigurable Processor – Datasheet", Rev.1.1, PACT XPP Technologies AG.
[5] P. M. Heysters, "Coarse-grained reconfigurable processors", CTIT Ph.D.-thesis series No. 04-66, 2004.
[6] http://www.arm.com
[7] http://www.smart-chips.net
[8] OSYRES-Operating framework for reconfigurable embedded systems, see www.ti-wmc.nl/downloads/Product_Sheet_OSYRES.pdf
[9] Lodewijk T. Smit, Gerard J. M. Smit, Johann L. Hurink, Hajo Broersma, Daniel Paulusma, and Pascal Wolkotte. "Run-time mapping of applications to a heterogeneous reconfigurable tiled system on chip architecture" in Proceedings of the International Conference on Field-Programmable Technology, pages 421-424, December 2004.