Invited paper: Parallel programming and Run-time Resource management Framework for Many-core Platforms: The 2PARMA Approach

C. Silvano[1], W. Fornaciari[1], S. Crespi Reghizzi[1], G. Agosta[1], G. Palermo[1], V. Zaccaria[1], P. Bellasi[1], F. Castro[1], S. Corbetta[1], E. Speziale[1], D. Melpignano[2], J.M. Zins[2], H. Hübert[3], B. Stabernack[3], J. Brandenburg[3], M. Palkovic[4], P. Raghavan[4], C. Ykman-Couvreur[4], I. Anagnostopoulos[5], A. Bartzas[5], D. Soudris[5], T. Kempf[6], G. Ascheid[6], J. Ansari[6], P. Mähönen[6], B. Vanthournout[7]

[1] DEI Politecnico di Milano, Italy, [2] STMicroelectronics, France, [3] Fraunhofer HHI, Germany, [4] IMEC vzw, Belgium and IBBT, Belgium, [5] ICCS National Tech. University of Athens, Greece, [6] RWTH Aachen University, Germany, [7] Synopsys, Belgium

***Abstract*—Real-time applications, hard or soft, are raising the challenge of unpredictability. This is an extremely difficult problem in the context of modern, dynamic, multiprocessor platforms which, while providing potentially high performance, make the task of timing prediction extremely difficult. Also, with the growing software content in embedded systems and the diffusion of highly programmable and re-configurable platforms, software is given an unprecedented degree of control on resource utilization. The 2PARMA project aims at overcoming the lack of parallel programming models and run-time resource management techniques to exploit the features of many-core processor architectures.**

The main goals of the 2PARMA project are: the definition of a parallel programming model combining component-based and single-instruction multiple-thread approaches, instruction set virtualisation based on portable byte-code, run-time resource management policies and mechanisms as well as design space exploration methodologies for Many-core computing architectures.

I. Introduction

The current trend in computing architectures is to replace complex super-scalar architectures with many processing units connected by an on-chip network able to accommodate such a high number of cores, satisfying the needs for communication and data transfers. This trend is mostly dictated by inherent silicon technology frontiers, which are getting as closer as the process densities levels increase. The number of cores to be integrated in a single chip is expected to continue to rapidly increase in the coming years, moving from Multi-core to Many-core architectures. This trend will require a global rethinking of software and hardware approaches.

Multi-core architectures are nowadays prevalent in general purpose computing and in high performance computing. In addition to dual- and quad-core general purpose processors, more scalable multi-core architectures are widely adopted for high-end graphics and media processing. Such platforms are becoming widespread as silicon technology develops in the sub-50nm nodes. The transition to multi-core is almost a forced choice to escape the silicon efficiency crisis caused by the looming power wall, the application complexity increase and the design complexity gap under tightening time-to-market constraints. While multi-core architectures are common in general-purpose and domain-specific computing, there is no one-size-fits-all solution. General-purpose multi-cores are still designed to deliver outstanding single-thread performance under very general conditions in terms of workload mix, memory footprint, runtime environment and legacy code compatibility. These requirements lead to architectures featuring a few complex, high-clock speed mega-cores with complex instruction sets, deep pipelines, non-blocking multi-level caches with hardware-supported coherency and advanced virtualization support. Today, we see a trend towards many-core fabrics, with a throughput oriented memory hierarchy featuring software-controlled local memories, FIFOs and specialized DMA engines. As a result, an SoC platform today is a highly heterogeneous system. It integrates a general-purpose multi-core CPU, and a number of domain-specific many-core subsystems. Examples of such emerging multi-core platforms are the Intels SCC [1] and STs Platform 2012 [2].

System-level design and optimization of computing systems is a highly challenging task. Especially since such systems are becoming more and more complex, from both hardware as well as software perspectives [3]. Over the last few years, the main focus in the design of computing systems has been to provide good performance and at the same time achieve low-power consumption. To achieve optimal results, a good coordination between hardware and software design is required. Therefore, memory-intensive applications running on embedded platforms (e.g., multimedia) must be closely linked to the underlying Operating System (OS) and efficiently utilize the available hardware resources. Putting all this together, it is clear that developing a complete, working system is an integration nightmare [3].

The 2PARMA project focuses on the design of a class of

This work is supported by the E.C. funded FP7-248716 2PARMA Project, www.2parma.eu

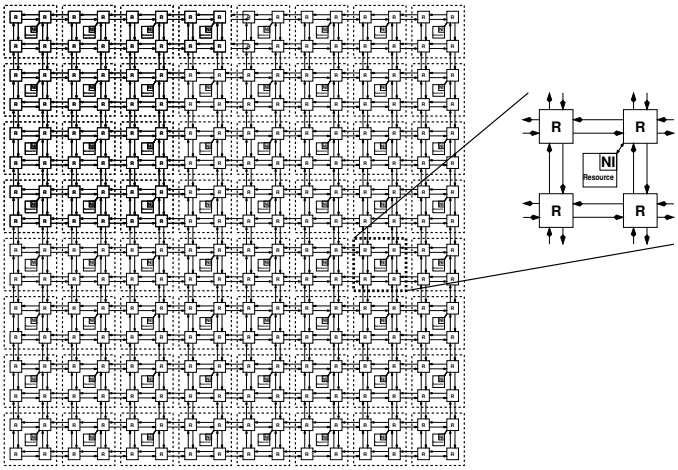

Fig. 1. 2PARMA Many-Core Computing Fabric Template.

parallel and scalable computing processors, which we call Many-core Computing Fabric (MCCF) template. This template is composed of many homogeneous processing cores connected by an on-chip network as shown in Figure 1. Benefits of Many-core Computing Fabric architectures include finer grained possibilities for energy efficiency paradigms, local process variations accounting, and improved silicon yield due to voltage/frequency island isolation possibilities. To exploit these potential benefits, effective run-time power and resource management techniques are needed. Moreover the Many-core Computing Fabric offers customisation capabilities to extend and to configure at run-time the architectural template to address a variable workload.

The 2PARMA project aims at overcoming the lack of parallel programming models and run-time resource management techniques to exploit the features of many-core processor architectures focusing on the definition of a parallel programming model combining component-based and single-instruction multiple-thread approaches, instruction set virtualisation based on portable bytecode, run-time resource management policies and mechanisms as well as design space exploration methodologies for Many-core Computing Fabrics.

The above scientific and technical objectives are intended to meet some of the main challenges in computing system research:

- To improve performance by providing software programmability techniques to exploit the hardware parallelism;
- To explore power/performance trade-offs and to provide runtime resource management and optimisation;
- To improve system reliability in terms of lifetime and yield of hardware components by providing transparent resource reconfiguration and instruction set virtualisation;
- To increase the productivity of the process of developing parallel software by using semi-automatic parallelism extraction techniques and extending the OpenCL programming paradigm for parallel computing systems.

The rest of this paper is organized as follows. Section II, introduces the main technological and research goals of the project. Section III, describes the target Many-Core Computing Fabric platforms while Section IV, describes the application scenarios selected for the project. Finally, Section V, draws some conclusions and highlights the expected project outcomes.

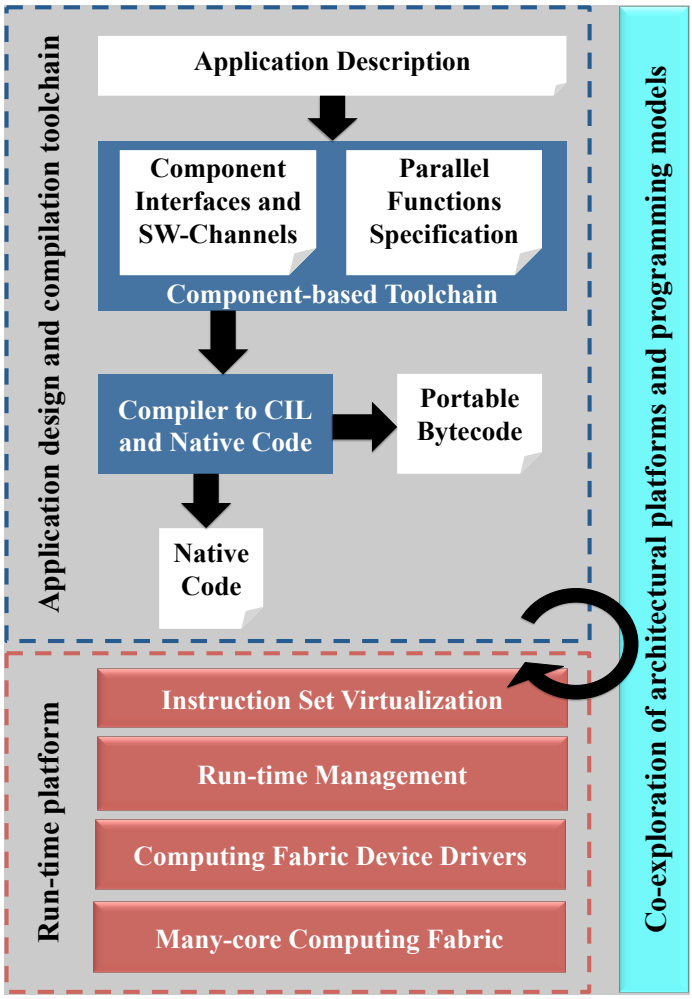

Fig. 2. Overview of the 2PARMA Design Flow.

II. Design Flow

The main goals of the 2PARMA project related to the analysis and development of the complete software layer able to exploit the features of future many-core processor architectures presented in the previous section. This goal has been tackled from several standpoints as presented in the following subsections.

The tool environment and design flow of the 2PARMA project is shown in Figure 2. The basic idea behind the 2PARMA project is to combine the automatic extraction of parallelism to dynamic compilation to exploit the management of system resources at runtime.

A. Programmability of Many-core Computing Fabrics

Data-parallel programming has been studied for a long time in connection with Single Instruction/Multiple Data (SIMD or vector) machines, and has been the object of a renewed interest in the last decade, due to the tremendous growth, both in terms of performance and number of units sold, of hardware acceleration technologies (such as Graphic Processor Units or GPUs) that can be used, besides their primary role, for the execution of highly parallel workloads of a general purpose nature.

The form of parallelism exposed by such a programmed system essentially consists of executing the same sequence of instructions on multiple collections of data. In their primary role, indeed, modern GPUs process fine-grained, data-parallel workloads consisting of thousands of independent threads executing vertex, geometry, and pixel-shader program threads concurrently. While being comparable to SIMD machines, current programmable GPUs employ a slightly different execution paradigm which is dubbed Single Instruction/Multiple Thread (SIMT).

2PARMA project tackles the issue of programmability of Multi-core Computing Fabrics at both the programming language and Operating System level. On one hand, it leverages the increasingly popular Component-Based Software Engineering (CBSE) and develops parallelism extraction techniques to identify opportunities for parallelisation at a high level in the design phase; 2PARMA then employs extensions of existing standards for parallel programming, such as OpenCL, to express data parallelism for Many-core Computing Fabrics. On the Operating System level, 2PARMA provides the means to define and deploy logic peripherals to the Many-core Computing Fabric, preserving isolation among them and efficient communication between host and Computing Fabric. The 2PARMA intends providing developers with comfortable tools and programming environments aiming at increasing software cycles productivity with respect to current, mainly manual, methodologies.

The proposed improvements build upon existing work on GPGPU architectures [4], since the 2PARMA platforms can be programmed using tools and languages commonly used for such architectures (e.g., OpenCL). However, since Many-core Computing Fabrics enjoy a greater level of efficiency in handling control flow than GPGPUs, techniques specific to ccNUMA architectures have also been developed for the efficient implementation of collective operations [5].

B. Runtime Resource Management

The development of new computing systems requires tuning of the software applications to specific hardware blocks and platforms as well as to the relevant input data instances. The behaviour of these applications heavily relies on the nature of the input data samples, thus making them strongly data-dependent. For this reason, it is necessary to extensively profile them with representative samples of the actual input data. An important aspect of this profiling is done not only at the dynamic data type level, which actually steers the designers' choice of implementation of these data types, but also at the functional level. We characterize the software metadata that these optimizations require, and we present a methodology, as well as appropriate techniques, to obtain this information from the original application. Equally important is for the designer to have a good knowledge of the platform characteristics. With both this information at hand (software metadata and platform characteristics) the designer can characterize the runtime behaviour of the application and determine its working modes and the reconfiguration overheads.

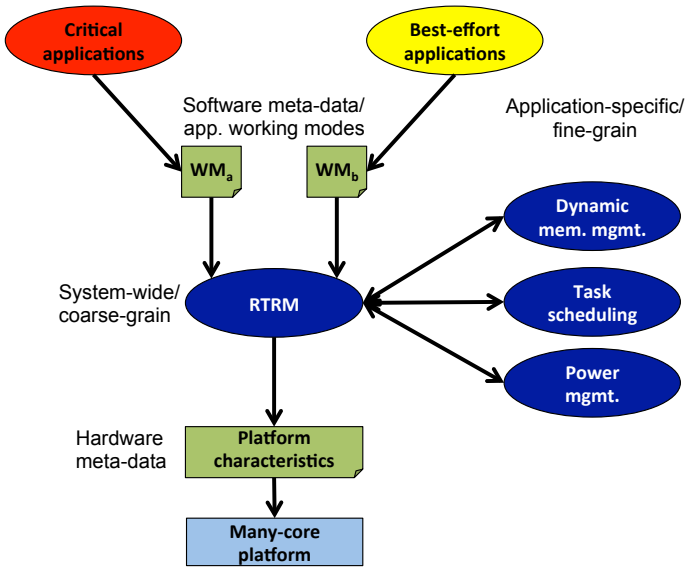

Fig. 3. Overview of the 2PARMA RunTime Resource Management Framework.

The 2PARMA's RTRM framework, targets the optimization of both computing fabric resources usage and applications' Quality-of-Service (QoS) requirements. Specifically regarding Power Management, 2PARMA investigates on the OS, services supporting runtime management. The main available OS frameworks related to power, hot-spots and process variation management have been analyzed and their characteristics have been compared to define the base for the future design of a new framework for supporting the QoS-based runtime management of a generic many-core computing platform. To support the new power manager, both application behaviour and computing platform characteristics should be identified, described and properly reported to the runtime manager framework. This involved a deeper investigation on two main aspects: the platform description and the interfaces with applications.

The RTRM framework is composed of a set of modules providing services to different "classes of users" (depicted in Figure 3). Two of these users are represented by applications and target-platforms. Applications usually have different Working Modes (WM), each one defining expected Quality-of-Service (QoS) and corresponding needs in terms of computational resources (e.g. processing elements, memory, bandwidth, etc). Target-platforms define a set of available resources, each one with specific: features, operating modes, monitoring and con-

trol points. Moreover, the platform architecture could define a specific functional relationship between available resources (e.g. clusters of processing elements and a certain memory hierarchy).

The RTRM is a component placed in between applications and the target-platform, which is in charge of managing applications access to platform available resources. This management is a quite complex activity generally aimed at meeting contrasting goals: maximizing applications' performance while reducing energy consumption. How this double goal could be obtained is behind the scope of this paper. Here it is important to understand that the RTRM tool should be supported on its role by the applications and the target-platform. Indeed, it is required for both these "users" to provide the framework with some information that could be effectively exploited to accomplish its task.

The overall structure of the required information, coming from applications and target-platforms (meta-data) should satisfy three main design goals:

1) **Completeness**: all the information required to properly support the RTRM management activities should be considered and represented.
2) **Portability**: the meta-data should describe both applications and target platform properties independently from each other. Indeed, for the success of the final solution it is considered interesting to support a "write once and run everywhere" approach. Thus, it should be possible to define application meta-data independently of the target platform they will run on.
3) **Simplicity**: the information required by the RTRM represents an "overhead" for developers of both applications and platforms, thus it is important to identify a solution that is as much as possible effortless to be used. Even better if the solution could be defined as an extension of the classical design and development flow of each system abstraction layer.

Overall, these requirements must be considered to properly define the collection of meta-data and the interfaces to acquire them from the applications and the target platform.

C. Design Space Exploration

Continuous adaptation and runtime management require large amount of information on the system and the applications to take effective and timely decisions. 2PARMA goes beyond traditional design space exploration (DSE) by defining a methodology to provide synthetic information about the points of operation of each application with respect to the subsets of resources available to it. Design space exploration methodologies developed in 2PARMA provide also architectural customisation to support parallel programming models, especially communication and memory mapping.

Focusing on the combined optimisation of parallel programming models and architectural paramters for many-core platforms, it is expected that conventional or state-of-the-art profiling techniques cannot be used for the task of analysing and profiling. Profiling memory accesses on a cycle accurate

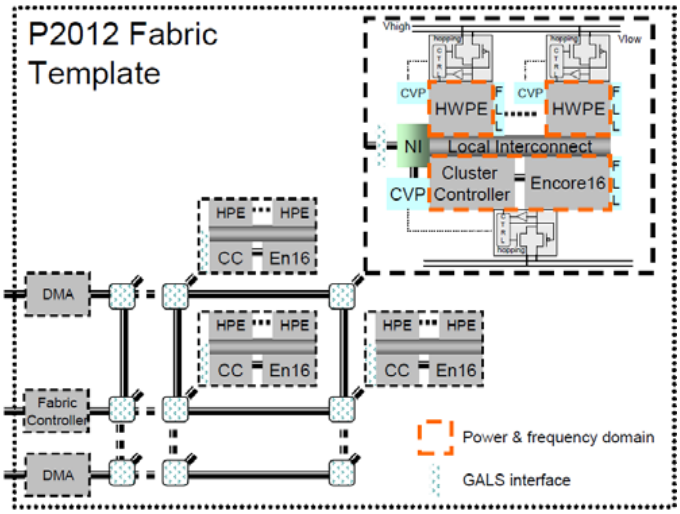

Fig. 4. Platform 2012 Template.

basis [6] is not sufficiently supported by available profiling tools due to the fact that only shared memory architectures were modelled at the time. Moreover, many core platforms will be built upon completely different interconnection networks requiring new profiling techniques taking into account connection topologies. The influence on the overall system performance of the implemented connection topology and the resulting fragmentation of memory accesses abroad the distributed memory will be evaluated by a set of tools. Within the Project the methodology described in [6] will be extended to be applicable for a wide range of architectural approaches including network on chip architectures. The tool called NoCTrace takes the system architecture described in SystemC as an input and guides the user through the process of non-intrusively integrating monitoring probes into the system model. Based on the profiling methodologies, it will be possible to get an in depth view of how parallel programming models behave on many core platforms. The results will be used to co-optimise the programming model and the architecture of the target platform.

III. Platforms and Applications

The 2PARMA project focuses on the Many-core Computing Fabric (MCCF) template composed of many homogeneous processing cores connected by an on-chip network. The 2PARMA project will demonstrate methodologies, techniques and tools by using innovative hardware platforms provided and developed by the partners, including the Platform 2012 an early implementation of Many-core Computing Fabric provided by STMicroelectronics and the many-core COBRA platform provided by IMEC.

A. STMicroelectronics Platfrom 2012

The P2012 program is cooperation among STMicroelectronics and CEA and aims at designing and prototyping a regular computing fabric able to improve manufacturing yield.

Platform 2012 (P2012) is a high-performance programmable accelerator whose architecture meets requirements for next generation SoC products at 32nm and beyond.

Platform 2012 (P2012) aims at moving a significant step forward in accelerator architecture video, and next-generation immersive multimodal applications such as computational photography, and augmented reality. P2012 is an area- and power-efficient many-core computing fabric, and it provides an architectural harness that eases integration of hardwired accelerators. The P2012 computing fabric os highly modular, as it is based on multiple clusters implemented with independent power and clock domains, enabling aggressive fine-grained power, reliability and variability management. Clusters are connected via a high-performance fully-asynchronous network-on-chip (NoC), which provides scalable bandwidth, power efficiency and robust communication across power and clock domains.

Figure 4 shows the Platform 2012 computing fabric, made of a variable number of tiles that can be easily replicated to provide scalability. Each tile includes a computing cluster with its memory hierarchy and a communication engine. Tiles are connected through a regular network on chip. The whole computing fabric operation is coordinated by a fabric controller and is connected to the SoC host subsystem through a dedicated bridge, with DMA capabilities. Clusters of the fabric can be isolated to reduce power consumption (or to switch-off a faulty element) and frequency/voltage scaling can be applied in active mode. Also, as soon as 3D stacking is used, it is envisaged that the NoC could provide high-bandwidth access to a stacked layer of embedded memory, whose dimension will also be determined according to the SoC needs. The P2012 computing fabric is connected to a host processor such as the ARM Cortex A9, via a system bridge. The fabric is in this way exposed to legacy operating systems like the GNU/Linux OS.

B. IMEC's ADRES-based COBRA Platform

The IMECs COBRA platform is an advanced platform template targeting 4G giga-bit per second wireless communication. One instance of the platform is shown in Figure 5. This platform can be customized to handle very high data rates as well as low throughputs in a scalable way.

Digital Front End (DFS) has been updated through more programmable and sensing DIFFS components. There is also an outer modem (OMD) of the platform by introducing flexible FlexFEC cores next to Viterbi ASICs. There are 4 DIFFS cores and 2 additional FlexFECs next to 2 Viterbi ASICs that are referred as Binary Convolutional Codes (BCC) in the COBRA platform). We improved the baseband cores by going to multi-threaded version of the cores. Multithreading introduced missing level of parallelism between the ILP and TLP. The AMBA bus is used only for the control plane of the platform, whereas in the data plane we have introduced a novel interconnection structure through crossbars. Crossbars are controlled by the interconnect controller which is a simple communication assist processor. Thus the architecture is shifted towards a more

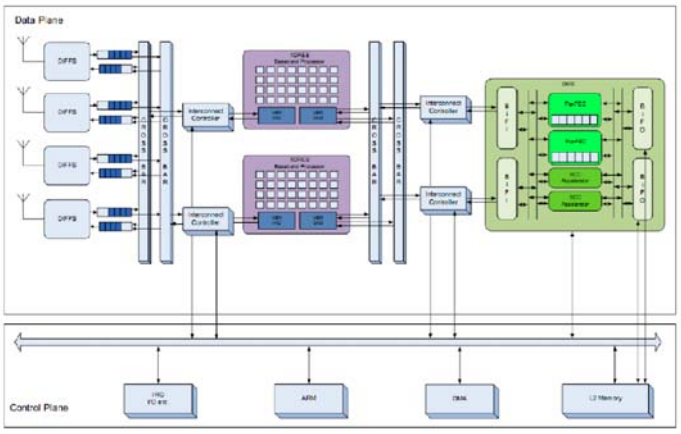

Fig. 5. IMEC's COBRA Platform.

streaming oriented one, which is beneficial for current and future wireless applications. In the COBRA platform we focus also on enabling run-time reconfigurability, which is a must for run-time control oriented research.

In this platform, besides the type and the size of each core, the number of each type of core can be selected based on the different standards that need to be supported on the platform. For example for the highest throughput modes for Wireless LAN 802.11n 4x4 MIMO, the number of DIFFS cores may be 4 and two (multi-threaded) ADRES cores and two FlexFEC cores. In case the platform has to support only a low end Wireless LAN SISO standard or a basic LTE SISO reception, one DIFFS core, ADRES core and one FlexFEC would be sufficient to meet the requirements of this mode.

IV. Applications

The Many-Core Computing Fabric template is designed as a coprocessor for computationally intensive applications in high-end embedded scenarios. To prove its effectiveness, and the effectiveness of the design flow and tools produced in the 2PARMA project, it is necessary to employ real world applications of considerable industrial impact. These applications will be engineered, optimised and specialised using the methodologies described in Section II, and tested on the two target implementations of the Many-Core Computing Fabric template. In this Section, we introduce the three applications chosen for the 2PARMA project: Scalable Video Coding (SVC), Cognitive Radio, and Multi View Video (MVV).

A. Scalable Video Coding (SVC)

SVC [7] also known as layered video coding has already been included in different video coding standards in the past.The new SVC standard provides scalability at the bit-stream level and supports functionalities such as bit-rate, format and power adaption as well as graceful degradation in poor transmission environments. An SVC encoder is capable of creating a scalable bit-stream, which contains multiple representations of the same video source, in a single run. Dependent to the conditions of transmission or the capabilities

of the device, parts of this bit-stream can be removed, so that the SVC decoder can select which part of the video stream is decoded.

Three different modes of scalability exist within the given application: temporal, spatial and peak signal to noise ratio (PSNR). Temporal scalability denotes the capability to support different frame-rates, while spatial scalability defines the existence of multiple video resolutions. Finally PSNR/fidelity scalability allows supporting multiple visual fidelities within the same scalable video data bit-stream. Of course all possible combinations of these three modes can be used for the encoding of the video data source.

The SVC application makes use of a layered approach for decoding the video stream. In order to minimize the data transmission, new inter-layer encoding techniques have been introduced by SVC. These new techniques imply that for the decoding of one specific video representation the corresponding SVC layer and all lower dependency layers must be processed by the SVC decoder. Therefore, every additional SVC enhancement layer increases the amount of data that has to be parsed and decoded. Accordingly this increases the required resources on a device needed for the video playback.

SVC represents one of the potential killer applications. Due to its scalable nature it is very well suited to demonstrate e.g. the different concepts of runtime resource management. SVC is not only scalable in respect to its coding features. It can also be used and implemented for scalability techniques regarding processing power, power consumption or resource management. In this case e.g. the scalability of the media bitstream will be reflected by the available processing power. On the other hand this scalability in terms of needed processing power can be used to minimize power consumption by limiting the available processing power accordingly to the given power budget. Scalable multi media signal processing on scalable computing platforms will be one of the most challenging research topics in this field in the future.

B. Cognitive radio (MAC and Physical Layer)

From the domain of wireless communications, a cognitive radio application will be provided by RWTH Aachen University, which includes both physical and MAC-layer processing. Especially, the low latency as well as high throughput and reconfiguration requirements of state-of-the-art wireless communication standards makes the cognitive radio application as a highly appropriate use case for the 2PARMA project and its parallel programming models.

Physical layer design is mostly characterized by a well-defined structure with substantially varying functionalities based on data flow models. Furthermore, the underlying algorithms are computationally intensive and time critical, therefore the idea pursued in this application is to define and explore these algorithmic kernels to represent characteristics of the functional blocks and by appropriately assembling these kernels, run-time reconfiguration of the physical layer can be executed to construct a complete wireless standard.

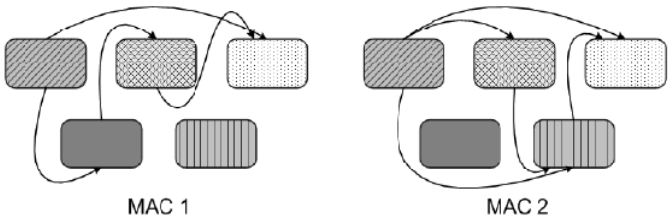

Fig. 6. MAC composition based on the common kernel functionalities.

By a thorough investigation of different MAC-layers, the most common kernel functionalities have been identified, which are referred to as the decomposition of MACs. The decomposition of MACs into their building block functionalities is carried out after a comprehensive analysis of existing MAC protocols from different classes of wireless networks. The MAC layer kernel functionalities are identified based on the commonalities among the protocols. The idea is to compose highly dynamic cognitive MAC solutions from a set of these kernel components based on the requirements of the applications and QoS demands. In order to bind the individual components and to coordinate the control and data flow among the decomposed MAC components, a Wiring Engine is designed. The wiring engine in the MAC-processor allows dynamically linking different MAC-layer components together at the run-time to result in suitable MAC functionalities. This approach allows fast on-the-fly reconfiguration, which is a must for cognitive MACs. The concept of composing MACs using the same set of kernel components is shown in Figure 6 below.

In order to allow efficient PHY and MAC processing and exploit parallelization, a novel tool-chain has been designed and implemented [8], [9] in the 2PARMA project. Our tool-chain enables on-the-fly reconfiguration and efficient resource management while respecting the time-critical nature of the data and control transfer among different PHY/MAC components in the cognitive radio application.

C. Multi-View Video

With the current development of electronic, network, and computing technology, Multi-View Video (MVV) becomes a reality and overcomes the following limitations of conventional single video. First, single view video only provides one view direction for an event at any time instance, while users may want to watch the event from different directions. Second, users are in a passive position. Even in a live video service, users can only watch the pre-selected video contents whereas no or little interactivity exists between the users and the content capturing. Third, recording an event from one fixed/dynamic view direction is not always the best way, from both visual experience and event representation criteria. For example, in a high action sports game or in an exercise diagnosis, audience and instructors often want to watch the video from comprehensive views, which gives them better experience or helps them to make a correct judgment.

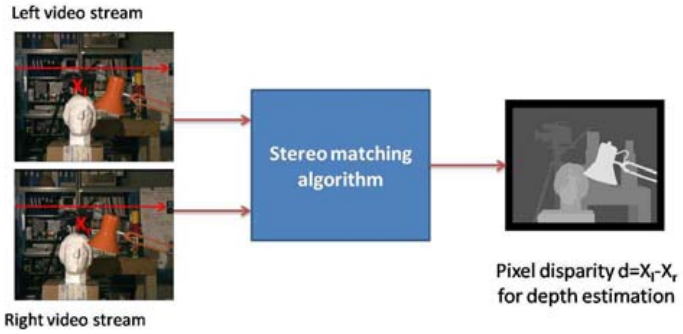

Fig. 7. Applications/Architectures Integration.

MVV refers to a set of N temporal synchronized video streams coming from cameras that capture the same scene from different viewpoints. The availability of multiple views of the scene broadens the application field: it extends the sensation of classical 2D video, it allows the user to freely choose a viewpoint (Free Viewpoint Video), and it provides a 3D depth impression of the scene (3D television).

Owing to the massive amount of involved data and the extensive processing, MVV presents research issues in video coding, image processing, computer vision, and computer graphics. Although MVV systems are diverse, several modules are common, such as MVV coding, compressing multi-view visual content, depth estimation (or stereo matching), estimating the pixel-based scene geometry, and view synthesis, synthesizing any desired virtual view given the captured multi-view scene. To ensure the success of an MVV system, these modules need not only to meet the challenging functional requirements, but also to run efficiently at interactive rates on the commodity platforms.

The application, considered in the 2PARMA project, is the cross-based stereo matching algorithm [10], where two aligned left and right cameras are assumed for the sake of clarity. Given a pair of images from two aligned cameras and from any stereo pair of pixels p in both left and right images, the depth of p is inferred from the pixel disparity, being the difference on the x coordinate of p in the stereo pair. Once the disparity is obtained for each pixel, the disparity map can be visualized in grayscale encoding (see Figure 7). The value selection for the parameters in this algorithm has an impact on the accuracy of the disparity map, in trade-off with the processing complexity, and hence the image rate, performance and the power consumption. The optimal combination of parameter values depends on the image content, which is unknown at design time. Hence these parameters need to be monitored preferably at run time.

V. Summary and project outcomes

The 2PARMA project tackles the issue of programming and managing a Many-Core Computing Fabric a novel architectural template represented within the project by STM Platform 2012 and IMEC Cobra architectures.

A design flow has been defined, starting with the high-level implementation of the application and leading to runtime management of the application execution, in a highly-variable context where multiple applications compete for resources. Design space exploration and profiling techniques close the feedback loop, helping the designer in refining the application for each target platform. The basic idea behind the 2PARMA project is to combine the automatic extraction of parallelism to dynamic compilation to exploit the management of system resources at runtime.

Finally, in order to prove the effectiveness of the design flow and tools produced in the 2PARMA project, high-impact real world applications are employed.

References

[1] T. G. Mattson, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, and S. Dighe, "The 48-core scc processor: the programmer's view," in *Proc. of SC*. IEEE Computer Society, 2010, pp. 1–11.

[2] STMicroelectronics and CEA, "Platform 2012: A Many-core programmable accelerator for UltraEfficient Embedded Computing in Nanometer Technology," 2010.

[3] A. Sangiovanni-Vincentelli, "Quo vadis, sld? reasoning about the trends and challenges of system level design," *Proc. of IEEE*, vol. 95, no. 3, pp. 467 –506, 2007.

[4] A. D. Biagio and G. Agosta, "Improved programming of gpu architectures through automated data allocation and loop restructuring," in *Proceedings of the PARMA Workshop (ARCS2010 Workshop)*. VDE Verlang, 2011.

[5] E. Speziale, A. D. Biagio, and G. Agosta, "An optimized reduction design to minimize atomic operations in shared memory multiprocessors," in *Proceedings of the 16th HiPS Workshop, IPDPS 2011*, 2011.

[6] H. Hübert and B. Stabernack, "Profiling-based hardware/software co-exploration for the design of video coding architectures," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 19, pp. 1680–1691, November 2009.

[7] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[8] X. Zhang, J. Ansari, G. Yang, and P. Mähönen, "Trump: Supporting efficient realization of protocols for cognitive radio networks," in *Proceedings of IEEE DySPAN*, 2011.

[9] V. Ramakrishnan, E. M. Witte, T. Kempf, D. Kammler, G. Ascheid, R. Leupers, H. Meyr, M. Adrat, and M. Antweiler, "Efficient and portable sdr waveform development: the nucleus concept," in *Proceedings of the 28th IEEE conference on Military communications*, ser. MILCOM'09. IEEE Press, 2009, pp. 918–924.

[10] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 19, pp. 1073–1079, July 2009.