SYSTEM LEVEL ARCHITECTURE EXPLORATION FOR RECONFIGURABLE SYSTEMS ON CHIP

Konstantinos Masselos

Imperial College of Science Technology and Medicine, Exhibition Road, London, SW7 2BT, United Kingdom
k.masselos@imperial.ac.uk

Nikolaos S. Voros

INTRACOM Telecom Solutions
254 Panepistimiou str., 26443, Patra, Greece
voni@intracom.gr

Yang Qu, Kari Tiensyrjä

VTT Electronics
P.O.Box 1100, FIN-90571 Oulu, Finland
Yang.Qu@vtt.fi, kari.tiensyrja@vtt.fi

Miroslav Cupak, Luc Rijnders

IMEC
Kapeldreef 75, B-3001 Leuven, Belgium
cupac@imec.be, rijnders@imec.be

Marko Pettissalo

Nokia Technology Platforms
P.O.Box 50, FIN-90571 Oulu, Finland
marko.pettissalo@nokia.com

ABSTRACT

During the last years, a new type of Systems-on-Chip called, Reconfigurable Systems-on-Chip (RSoCs), has appeared. The design of such systems is a complex task and requires innovative methods to support the development process. In this paper, we present two alternative approaches for the efficient architecture exploration of RSoCs, based on SystemC language and on OCAPI-xl environment. The approaches introduced, allow early evaluation of alternative mappings of system's functionality onto different architectures. As a result, the time consuming iterations from lower design stages are eliminated, and reduced design time is achieved. The paper proves the effectiveness of the proposed approaches through three different case studies, borrowed from complementary domains.

1. INTRODUCTION

The inclusion of reconfigurable hardware in Systems-on-Chip offers several advantages. It combines computation efficiency and flexibility in the use of hardware resources over time, allows area optimization through sharing of hardware resources in time among different tasks, post fabrication upgrading of functionality (improving time to market) and post-fabrication bug fixing.

The design of a Reconfigurable System-on-Chip (RSoC) is not a trivial task. To obtain an efficient implementation, extended design flows are needed to cope with the reconfiguration aspects and overheads (reconfiguration time and power, and memories for configurations storage). More important, efficient architecture exploration methods are essential to ensure correct architecture decisions early in the design cycle and eliminate time consuming iterations from low level design stages in case constraints are not met.

Many high-level design approaches for reconfigurable systems rely on compilation of C or C-like descriptions of applications targeting on reconfigurable hardware [1]. Recently, a few co-compilation and co-synthesis type design approaches for reconfiguration have been published [2]. UML has been widely used for system specification and documentation. In [3, 4], researchers have presented different approaches to extend the UML to the reconfigurable system design domain.

Reconfigurable hardware brings a new dimension to system partitioning. The dynamic reconfiguration requires partitioning to address both temporal and spatial dimensions. Such an automatic partitioning is, in the general case, still an unsolved problem. Nevertheless, in specific cases, solutions for temporal partitioning [5], for task scheduling [6] and for context management [7] have been proposed.

As far as architecture exploration and mapping are concerned, there are two approaches supported by the existing commercial design flows: (a) the *tool oriented design flow*, and (b) the *language oriented design flow*. Example of tool oriented design flow is the N2C by CoWare [8]. The design flows supported by such tools work well on traditional hardware/software solutions. The incorporation of new reconfigurable parts is not possible without unconventional trickery. Examples of language oriented design flows are OCAPI-xl [9] and SystemC [10]. Especially for the latter, since it promotes the openness of the language and the standard, the addition of a new domain can be made to the core language itself. However, the method mostly preferred is to model the basic constructs required for modeling and simulation of reconfigurable hardware, using basic constructs of the language. In this way, the language compatibility with existing tools and designs is preserved. SystemC and OCAPI-xl extensions for reconfigurable SoC design are discussed in detail in Sections 2 and 3 respectively.

The rest of the paper is organized as follows: In Sections 2 and 3, architecture exploration methods based on SystemC language and on OCAPI-xl environment are detailed respectively. Case studies' results from three different domains are described in Section 4. Finally, conclusions are drawn in Section 5.

2. SYSTEMC BASED ARCHITECTURE EXPLORATION FOR RSOCS

The SystemC-based approach focuses on the system partitioning and design space exploration for run-time reconfigurable SoCs. In the SystemC-based approach, a new architecture is defined partly based on an existing architecture and partly using the system specification as input. We assume the initial architecture and the HW/SW partition are given at the beginning of system-level design. The SystemC extension is designed to work with a SystemC model of the existing device to suit the design considering Run-Time Reconfiguration (RTR) hardware. The provided support in the SystemC approach include: a) estimation and analysis support for design space exploration and system partitioning, and b) reconfiguration modeling using standard mechanisms of SystemC, and a transformation tool to automatically generate SystemC models of the reconfigurable hardware.

Estimation approach The estimation approach starts from function blocks represented using C-language and produces hardware execution time and resource utilization estimates for each function block. The estimator works on one function block at a time. The SUIF-based [11] front-end pre-processor is used to extract Control-Data Flow Graphs (CDFG) from the C code. Then a set of high-level synthesis tasks is carried out to produce the estimates. ASAP and ALAP scheduling is used to determine the critical paths, from which we estimate the execution time. A modified version of Force-Directed Scheduling (FDS) is used to estimate the hardware resources required for the tasks. Finally, allocation algorithms are used to estimate the hardware resources required for interconnections with multiplexer types of interconnection units. The current estimator targets a Virtex2-like FPGA in which the main resources are LookUp-Tables (LUTs) and multipliers.

Modeling Reconfiguration Overhead A SystemC model, called Dynamically ReConfigurable Fabric (DRCF) component, is developed to model the configuration overhead. Different features associated with the reconfigurable technology are not directly modeled in the DRCF component. Instead, the DRCF component contains the functions that describe the behavior of the reconfiguration process and relates the performance impact of the reconfiguration process to a set of parameters. Thus, by tuning the parameters, designers can easily evaluate the trade-offs between different technologies without going into implementation details. At the moment, the following parameters are available for designers: a) the length of the required memory space, which represents the size of the context, and b) delays associated with the reconfiguration process in addition to delays of memory transfers.

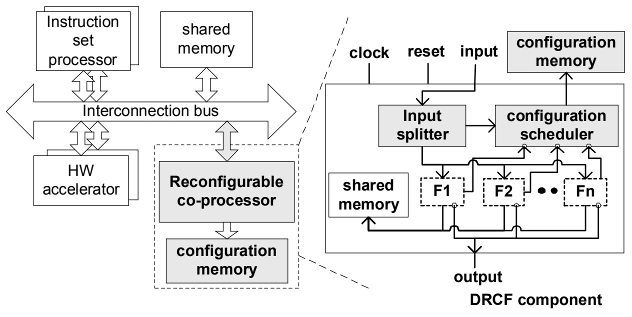

Fig. 1 System level modeling for RSoC.

In order to enable the DRCF component to capture and understand incoming messages, the SystemC modules of the candidate components must implement the *read()*, *write()*, *get_low_addr()* and *get_high_addr()* interface methods. In fact, these interface methods are very common for bus slave modules in transaction-level models.

A general model of a reconfigurable SoC is shown in Fig. 1. The DRCF component is a single hierarchical SystemC module, which instantiates the candidate components (F1 to Fn). Each candidate component occupies separate system address space and is an individual SystemC module. The Configuration Scheduler (CS) monitors the operation states of the candidate components and controls the configuration process. The Input Splitter (IS) is an address decoder and it manages all incoming

Interface-Method-Calls (IMCs). If it detects an incoming IMC to an unloaded task, it informs the CS that the task should be loaded and hold the IMC until the loading is done. Otherwise, the IMC is forwarded to the target module.

In order to reduce the coding effort, we have developed a tool that can automatically transform SystemC modules of the candidate components into a DRCF component. A script file is used to guide the transformation process and record the parameters, e.g. configuration latency. The outputs are SystemC files of a new reconfigurable SoC system, in which those specified candidate components are replaced with a DRCF component.

3. OCAPI-XL ARCHITECTURE EXPLORATION FOR RSOCS

OCAPI-xl is a C++ based design environment for development of concurrent, heterogeneous HW/SW applications. It abstracts away the heterogeneity of the underlying platform through an intermediate-language layer that provides a unified view on SW and HW components. The language is directly embedded in C++ via a creatively designed set of classes and overloaded operators [9], and has an abstraction level between assembler and C.

OCAPI-xl's design-flow starts at high (typically C/C++) level and goes all the way down to the implementation in a sequence of incremental steps. The OCAPI-xl design flow can be divided as follows:

1. Identification of code suitable for parallelisation
2. Partitioning of the single-threaded C/C++ code into parallel tasks
3. Mapping of the functional model from step 2 onto the architecture
4. Refinement of selected processes to OCAPI-xl embedded language, which can then be used in HW, as well as in SW scenarios.

At the stages from 2 to 4, OCAPI-xl provides the designer with simulation results as well as quantitative figures of system throughput, activity, performance etc. (i.e. it provides important feedback directing new refinement steps).

OCAPI-xl extension for the design of RSoC The OCAPI-xl based approach demonstrates the ability to model and analyse system partitioning and mapping already at the system level. Relevant design steps have been enhanced to include reconfigurability aspects. The provided support for the OCAPI-xl related approach includes:

- Software process scheduling extension.
- High-level modelling of context switching.

Software Processes Scheduling Extension In the high-level software model of computation, concurrency is considered at the processor level. This means that for every process there is a separate processor assumed. Naturally, in real life this will typically not be the case. In realistic software implementation an operating system allows all the processes to be assigned to the same software processing resource. So, from the performance point of view, the processes are not running concurrently, but they are sequentialized by the operating system scheduler onto the processing unit. To model such behaviours in the OCAPI-xl performance model, a separate process type, *procManagedSW*, has been introduced. To be able to create a process of the type procManagedSW, the designer must create a scheduling object. This scheduler will perform the actual sequentialization of all the processes Fig. 2 illustrates sequentialization of three processes exploiting Round-Robin scheduler. The user has the opportunity to exploit a predefined set or define its own scheduler for targeted operating system.

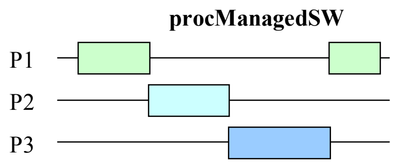

Fig. 2 Sequentializing computation over time.

It is important to realize that switching between the different SW tasks is not penalty-free. In order to come to most accurate performance results, context-switching overhead is also considered in the performance model. The user can define extra context switching time for every process created, which is then applied to that process during the OCAPI-xl simulation.

High-Level Modeling of Context Switching The ability to reschedule a task either in HW or SW is an important asset in a RSoC. To support this feature, high-level implementation and management of HW/SW relocatable tasks in OCAPI-xl have been modelled. The aim was to model a pre-emptive relocation of tasks from the reconfigurable logic to the SW and vice versa. The model supports spatial temporal scheduling in HW and SW. Within this model, it is in principle also possible to model resource sharing at the HW level, by replacing one task with another on the same physical reconfigurable resource and by adding the appropriate contexts.

4. DESIGN CASES

4.1 WCDMA

A WCDMA detector [12] has been selected as a design case to validate the SystemC-based approach. The target is an RTR-type of implementation and the implementation platform was the VP20FF1152 development board, which contains one Virtex2P XC2VP20 FPGA (predefined RSoC). The whole WCDMA base-band receiver system is depicted in Fig. 3. The case study focuses on the detector portion (gray area in Fig. 3) of the receiver and a limited set of the full features were taken into account. The detector case used 384 kbits/s user data rate without handover.

The design started from C representation of the system. The estimation approach and prototype tool described in Section 2 has been used. Based on the resource estimates, the final dynamic context partition is as following: the channel estimator is assigned to one context (1387 LUTs), and the other three processing blocks are assigned to a second context (1078 + 463 + 287 = 1828 LUTs). This partition results in balanced resource utilization and less communication traffic across the contexts.

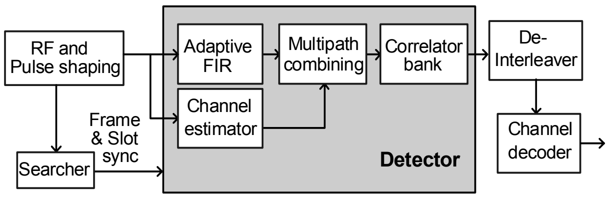

Fig. 3 WCDMA receiver baseband system.

A fixed system has been created, in which all of the four detector functions are mapped onto separate hardware accelerators and the scheduling task is mapped onto a software task that runs on the embedded PowerPC processor core. The prototype tool mentioned in Section 2 has been used to automatically generate the DRCF component from the fixed system. The SystemC models are described at the transaction level, in which the workload is derived based on the estimation results but with manual adjustment. The performance simulation shows that the system requires 2 reconfiguration requests per slot. When the configuration clock is running at 33 MHz, the reconfiguration latency is 2.73 ms for 16 bits configuration bit-width.

Vendor-specific tools were used in the system refinement and implementation phases. The SystemC-based approach does not produce RTL synthesizable code. C code and RTL-VHDL code were manually created. The reconfiguration was implemented using SystemACE CF solution and the control function of the SystemACE module was inserted into the SW code. The module-based partial reconfiguration design flow is used to implement the two contexts in the Virtex II Pro. In the implementation, 920 LUTs and 4 Block RAMs are required for the context containing the channel estimator, and 1254 LUTs, 6 Block RAMs and 12 Block Multipliers are required for the other context. The static part requires 1199 LUTs and 25 Block RAMs. 21 bus macros are used to connect the static part and dynamic contexts. The size of the partial bit streams generated for the context-1 and the context-2 are 278k bytes and 280k bytes respectively. When the system is running at 100 MHz, the decoding time of one slot of data is 9.66 ms including the reconfiguration latency.

Evaluation A fixed hardware and a pure software implementation have been developed as reference designs. In the fixed-hardware implementation, the processing time for decoding one slot of data is 1.06 ms at 100MHz, but the design doubles the resource requirement compared to the RTR system. In the full software implementation, the processing time for one slot of data is 294.6 ms, over 30 times compared to the RTR system. This does not fulfill the real-time requirements.

The WCDMA detector design case is based on sample by sample processing and is not ideal for demonstrating the RTR benefits. It was selected for the validation of the SystemC-based design methodology. Through the design case, the estimation approach and the DRCF modeling approach have shown their usefulness by providing reasonably accurate results without going into low-level implementation. With C code and test data that were available in the WCDMA detector design case, the design at the system-level took less than a week.

4.2 Wireless LAN

In this section, the prototyping of a HIPERLAN/2 [13] RSoC on ARM integrator prototyping platform is described. ARM Integrator includes the ARM AHB motherboard, the core modules containing ARM processors and the logic modules containing FPGAs.

An ANSI C model has been developed for the MAC and physical layers' functionality of the system. The size of the model is 20000 lines of code. Using the ANSI-C model as input, OCAPI-xl models of the HIPERLAN/2 MAC and physical layers have been developed. For the high level exploration, high-level OCAPI-xl processes (procHLHW, procHLSW and procManagedSW) have been used. Using the performance estimation (in terms of execution cycles) capabilities of OCAPI-xl different mappings of HIPERLAN/2 tasks on hardware and software have been evaluated and the most promising solution has been identified.

Based on (a) the system level architecture exploration, (b) the analysis of the HIPERLAN/2 computational complexity and (c) performance constraints, two core modules and two logic modules have been allocated for the realization of the HIPERLAN/2 system. Each core module includes an ARM7TDMI processor and each logic module includes a Xilinx Virtex E 2000 FPGA. The architecture of the ARM Integrator instance that has been selected for the realization of the HIPERLAN/2 system is shown in Fig. 4. The first core module acts as protocol processor realizing the major part of the HIPERLAN/2 DLC functionality. The second core module realizes the lower part of the HIPERLAN/2 MAC functionality and also controls the operation of the baseband processing part. The first logic module (bottom FPGA) realizes the frequency and data domain parts of the baseband receiver. The second logic module (top FPGA) realizes the baseband transmitter, the time domain blocks of the baseband receiver, the interface to MAC and a slave interface to an AMBA bus.

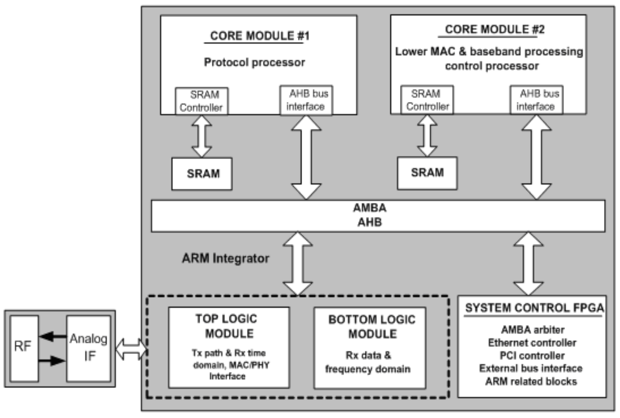

Fig. 4 Architecture of selected ARM Integrator platform instance.

In the next step the high level OCAPI-xl model has been refined by changing processes' types from high level to low level (procOCAPI1 and procANSIC). This allowed a cycle accurate simulation of the complete system functionality and confirmation that timing constraints are met. After this step, hardware and software development started. The ARM development tools have been used to map the developed C++ code on the ARM processors' of the targeted platform. The code and the data for the tasks are stored in SDRAM memory. The size of the code running on the protocol processor is 1.4 Mbytes while the size of the code running on the second processor is 50 Kbytes. Both ARM processors operate at 50 MHz.

A typical FPGA design flow has been adopted for the realization of the tasks assigned on the platform's logic modules starting from VHDL. The total utilization of the bottom logic module (FPGA) is 85% (93 I/Os, 14923 function generators, 12164 CLB slices, 6368 flip flops/latches). The total utilization of the top logic module is 89% (312 I/Os, 16527 function generators, 11252 CLB slices, 8544 flip flops/latches). The size of the configuration files for the two FPGAs is 1,2 Mbytes. Two clocks of 40 and 80 MHz are driven in each FPGA.

Evaluation The performance results presented above from the realization of the HIPERLAN/2 system on the ARM Integrator platform are expected to improve in a reconfigurable SoC implementation. This is due to the overheads introduced by the ARM Integrator platform architecture. The use of the proposed system level architecture exploration approach allowed the definition of an efficient architecture that satisfied the targeted performance constraints. No time consuming iterations and feedback loops from the low level implementation stages for architecture modifications were required.

4.3 MPEG4 Decoder

MPEG-4 Video Decoder (see Fig. 5) has been selected to demonstrate that OCAPI-xl based approach, as described in Section 3, is a highly suitable approach for designing the reconfigurable SoCs at the system level.

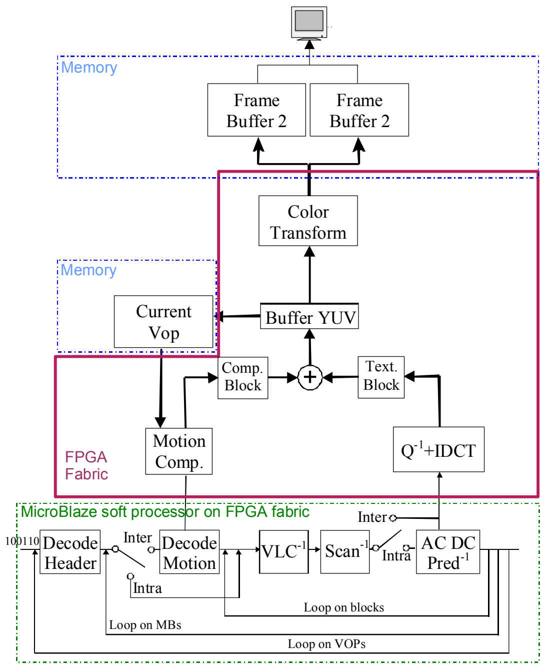

Fig. 5 Block diagram of the simple profile MPEG-4 video decoder with indication of final HW/SW partitioning

The initial C/C++ code was obtained from Final Draft International Standard sources. After the optimisation phase, the design proceeded with modelling of performance estimation in OCAPI-xl environment. A number of small tests were done on the board to find the operator execution times and memory access times and to select proper memory architecture. Taking into account the analysis results from optimisation phase, the most CPU time and memory access demanding blocks of the decoder were selected to become the candidates for HW acceleration. To demonstrate early RSoC design evaluation, the design was modelled in two flavours:

- Configured as pure SW version of MPEG-4 decoder running on soft processor core.
- Configured as HW accelerated version, where most cycle demanding blocks have been implemented in reconfigurable HW.

After a SW processes annotation, the performance estimation started with simulation of the pure SW version of the decoder. For HW accelerated version of the decoder, the OCAPI-xl processes have been redefined to high-level hardware (HLHW) type. Comparing the average frame time with pure SW version, the speed-up of factor 4.2 was estimated. The final HW/SW partitioning of the MPEG-4 video decoder for both versions is shown in Fig. 5.

Evaluation The key benefit shown by this design case is demonstration of ability of high-level simulation-based performance estimation and evaluation of context switching between the different computation resources. Based on the performance estimation results, it is possible to construct a trade-off curve for considered HW/SW partitions. This gives the designer unique opportunity to evaluate at early stage of the design process, which of the components is beneficial to implement in reconfigurable HW and which ones will be running in SW. As a result, design time costly lower-level iterations are eliminated.

The accuracy of estimations has been measured by comparing the difference between the estimated performance and board performance. The difference is 8% which is acceptable for the most relevant test sequence.

The design was mapped on Xilinx's Multimedia Development Board containing xc2v2000 Virtex-II FPGA with a single embedded MicroBlaze soft processor gaining 46% utilization (5000 slices) for HW accelerator and 71% utilization (7703 slices) for the whole decoding system. It uses 33% of available 18x18 multipliers, 76% block RAMs and 52% of LUTs. Synplify-Pro FPGA synthesis tool and vendor specific toolsets have been used in the implementation phase.

5. CONCLUSIONS

For the efficient design of complex RSoCs within strict design time constraints, efficient design methodologies at the early design stages are required. The proposed architecture design space exploration methods rely on the use of SystemC language and OCAPI-xl environment. Through appropriate extensions, they address reconfiguration issues at a high level. The effectiveness of the proposed methods has been proven through their application on three real life systems, borrowed from different domains: a WCDMA system, a WLAN system and an MPEG4 decoder.

6. REFERENCES

[1] G. Venkataramani, W. Najjar, F. Kurdahi, N. Bagherzadeh, W. Bohm, J. Hammes, "Automatic Compilation to a Coarse Grained Reconfigurable System-on-Chip", *ACM Transactions on Embedded Computing Systems*, vol. 2, No. 4, 2003, pp. 560-589.

[2] J. Becker, R. Hartenstein, "Configware and Morphware - Going Mainstream", *Journal of System Architecture*, vol. 49, 2003, pp. 127-142.

[3] D. Fröhlich, B. Steinbach, T. Beierlein, "UML-Based Co-Design for Run-Time Reconfigurable Architectures*", In 2003 Forum on Specification & Design Languages*, pp. 285-296.

[4] T. Schattkowsky, W. Mueller, A. Rettberg, "A Model-Based Approach for Executable Specifications on Reconfigurable Hardware", *In 2005 Design Automation and Test in Europe Conference*, pp. 692-697.

[5] C. Bobda, "Synthesis of Dataflow Graphs for Reconfigurable Systems using Temporal Partitioning and Temporal Placement" Dissertation, University of Paderborn, 2003.

[6] J. Noguera, R. M. Badia, "System-Level Power-Performance Trade-offs in Task Scheduling for Dynamically Reconfigurable Architectures", *In 2003 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 73 – 83.

[7] R. Maestre, F. J. Kurdahi, M. Fernandez, R. Hermida, N. Bagherzadeh, H. Singh, "A Framework for Reconfigurable Computing: Task Scheduling and Context Management", *IEEE Transactions on VLSI Systems*, vol. 9, issue 6, 2001, pp. 858 – 873.

[8] CoWare Inc (2004) Available: http://www.coware.com

[9] OCAPI-xl (2004) Available: http://www.imec.be/ocapi/welcome.html

[10] SystemC (2004) Available: http://www.systemc.org

[11] R. P. Wilson et. Al, "SUIF: An Infrastructure for Research on Parallelizing and Optimizing Compilers", In 1994 7th ACM SIGPLAN symposium on Principles and practice of parallel programming, pp. 37-48.

[12] M. J. Heikkila, "A novel blind adaptive algorithm for channel equalization in WCDMA downlink", *In 2001 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, pp. 41- 45.

[13] ETSI, "Broadband Radio Access Networks (BRAN); HIPERLAN type 2; Physical (PHY) layer, v 1.2.1", 2000