Int. J. Communications, Network and System Sciences, **2011, 4, 436-446**
doi:10.4236/ijcns.2011.47052 Published Online July 2011 (http://www.SciRP.org/journal/ijcns)

Scientific Research

System-Level Performance Evaluation of Very High Complexity Media Applications: A H264/AVC Encoder Case Study

Hajer Krichene Zrida[1], Abderrazek Jemai[2], Ahmed C. Ammari[3], Mohamed Abid[1]
[1]*Computer & Embedded Systems Laboratory, National School of Engineers of Sfax, Sfax University, Sfax, Tunisia*
[2]*LIP2 Laboratory, Faculty of Science of Tunis, Tunis, Tunisia*
[3]*Research Unit in Materials Measurements and Applications, National Institute of Applied Sciences and of the Technology, Carthage University, Carthage, Tunisia*
E-mail: *hajer_kri@yahoo.co.nz*
Received March 23, 2011; *revised May* 20, 2011; *accepted June* 10, 2011

Abstract

Given the substantially increasing complexity of embedded systems, the use of relatively detailed clock cycle-accurate simulators for the design-space exploration is impractical in the early design stages. Raising the abstraction level is nowadays widely seen as a solution to bridge the gap between the increasing system complexity and the low design productivity. For this, several system-level design tools and methodologies have been introduced to efficiently explore the design space of heterogeneous signal processing systems. In this paper, we demonstrate the effectiveness and the flexibility of the Sesame/Artemis system-level modeling and simulation methodology for efficient performance evaluation and rapid architectural exploration of the increasing complexity heterogeneous embedded media systems. For this purpose, we have selected a system level design of a very high complexity media application; a H.264/AVC (Advanced Video Codec) video encoder. The encoding performances will be evaluated using system-level simulations targeting multiple heterogeneous multiprocessors platforms.

Keywords: System-Level Performance Evaluation, Embedded Systems Design Space Exploration Tools, the Sesame/Artemis Design Tool, a Parallel H.264/AVC Video Encoder

1. Introduction

The architectural complexity of System-on-Chip (SoC)-based embedded systems, as well as the design re- quirements regarding real-time performance, high flexibility, low power consumption and cost greatly complicate the system design. Nowadays, the classical design methods, typically starting from a single application specification, become short used for designing such an embedded system. In order to resolve the increasing design complexity, researchers have recently come up with a new design concept called system-level design [1]. For this purpose, a new generation of system-level tools and methodologies has been introduced to efficiently explore the design space of heterogeneous signal processing systems. Each tool/methodology directly reflects a well-defined design flow.

The Y-chart layer's based approach, considered as the most popular approach for designing multimedia oriented systems, is already being followed in most recent system-level design works [1]. It tries to improve the shortcomings of the classical HW/SW co-design approach by abandoning the usage of low-level (instruction-level or cycle-accurate) simulators for the design space exploration at an early stage of the flow, and abandoning a single system specification to describe both hardware and software parts. Indeed, the Y-chart methodology recognizes a clear separation between an application model, an architecture model and an explicit mapping step to relate the application model to the architecture model. The application model describes the functional behavior of an application, independent of architectural specifics like the HW/SW partitioning or timing characteristics. The architecture model defines the architecture resources, captures their timing characteristics, and then simulates the performance consequences of the application events

(communication and computation operations) for software (programmable components) and hardware (reconfigurable/dedicated) executions.

As showed in **Figure 1**, the Y-chart general design scheme is composed of four steps [1]. The first step "Application Modeling" aims to capture a functional specification of the system in the form of a set of benchmark applications. The second step "Architecture Modeling" consists in modeling the target architecture by the resources available in the system. In embedded systems, these resources typically are processors, operating systems, buses and memories. After that, the parallel application processes are mapped onto the resources of the architecture. The result of the mapping step is an implementation of the system which can be used as an input for the performance analysis step. Typically, based on the Y-chart approach principle, a system designer studies the set of benchmark applications, makes some initial calculations, and proposes the architecture. The designer then evaluates and compares several instances of the platform by mapping each application onto the platform architecture by means of performance analysis. The resulting performance numbers may inspire the designer to improve the architecture, restructure the application, or change the mapping. The possible designer actions are shown with the light bulbs in **Figure 1**.

The outline of the paper is as follow. In Section 2, we first present the underlying properties of some different tools. Based on these most important design criterions considered in our comparative synthesis, the Sesame software framework is selected among the best system-level design methodologies. In Section 3, we describe the main features, tools, and methods provided by the Sesame/ Artemis simulation and modeling environment. Section 4 presents a complexity analysis of the H.264/AVC standard and reviews a performed previous work for the development of an optimized parallel encoder model. Sesame is used in Section 5 to evaluate the encoder performance targeting multiprocessors architectures and will show up the effectiveness and the flexibility of this design methodology.

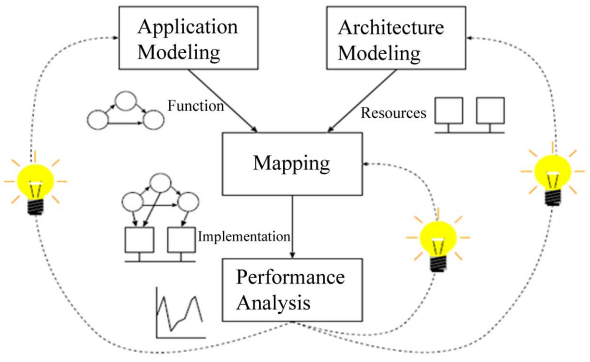

Figure 1. The Y-chart: a general scheme for heterogeneous system design.

2. System-Level Exploration Tools Comparative Synthesis

In the literature, there are a number of exploration environments that facilitate the system-level design space exploration by providing support for mapping a behavioral application specification to an architecture platform model [1]. Although all the system-level design methodologies are created to be used in the same field: designing embedded systems at high system level, there exist wide diversity among them. In **Table 1**, we summarize the most interesting design properties of the some representative ones.

The study found that selected methodologies and tools as shown in **Table 1** differ from each other in first their HW/SW design approach. Some of them, like the Ptolemy tool [2], don't support a layered abstraction level design approach and use a single specification including both functional behavior and architecture models. Others support the platform-based design approach (like Metropolis [3]) or the top-down design methodology (like SpecC [4]). However, it is demonstrated that the Y-chart layer's based approach, which is followed by several recent works, became nowadays the most popular and used for designing heterogeneous multiprocessors embedded systems.

Although the differences among the seven tools are not absolute, the features shown in **Table 1** indicate that the most preferable methodologies/tools are Metropolis, VCC [4], and Sesame-like [5,6], because they have the largest amount of positive marks ("+"). By elimination, the Metropolis environment is excluded of the most preferable methodologies list since it does not facilitate explicitly the Y-chart approach. Between the VCC and Sesame tools, we observe that the mixed-level simulation is only supported by the Sesame tool. For this, we have opted for selecting the Artemis/Sesame methodology to implement at system-level the H.264/AVC video encoding application on a multiprocessor SoC-based architecture. Indeed, the system-level modeling and simulation framework Sesame/Artemis is developed to directly reflect the Y-chart design approach. It provides several methods and tools to quickly and separately build the application process network model, the target architecture model, and the mapping model of the application onto the architecture.

Currently, Sesame has been evaluated for the design of two medium complexity media applications: an MPEG-2 decoder and a variant of M-JPEG encoder [7,8]. Our objective in this paper is to use this methodology for the

Table 1. Overview of some properties of presented tools and methodologies.

Methodology/tool	Ptolemy	Polis	Metropolis	SpecC	VCC	SystemC	Sesame/Artemis
Y-chart supported	-	+	-	-	+	x	+
MoC variety supported	+	x	+	x	+	x	-
Dynamic performance models	+	+	+	+	+	+	+
Formal analysis and verification	-	+	+	+	x	x	x
Reusability supported	+	+	+	+	+	+	+
Complex applications domains supported	x	-	+	x	+	+	+
Target architecture variety supported	x	-	+	+	+	+	+
All abstraction levels supported	-	+	+	+	+	+	+
HW synthesis and IP Integration supported	-	+	+	+	x	-	+
Mixed simulation supported	-	x	x	x	-	+	+
Automatic HW/SW partitioning supported	-	-	-	-	-	-	-
Automatic mapping	-	-	+	-	-	-	-
Rapid prototyping	x	+	+	+	+	+	+

Legend: True, +; False, -; May be, x.

design of very complex media systems. The H.264/AVC reference video encoder represents an example of a very complex case study typical of the multimedia domain. It has been designed with the goal of enabling significantly improved compression performance relative to all existing video coding standards [9]. Such a standard uses advanced compression techniques that in turn, require high computational power [10]. Implementing a H.264/AVC video encoder for an embedded SoC is thus a big challenge since this encoder requires very high computation power to achieve real-time encoding. In this study, both modeling and mapping stages of the Sesame design flow are explored for an optimal H.264/AVC encoder implementation verifying constraints. This will demonstrate the effectiveness and the flexibility of the methods and tools provided by this methodology for rapid system-level design space exploration of complex embedded systems.

3. The Sesame/Artemis Simulation and Modeling Environment

In this section, we will briefly describe the Sesame/Artemis simulation and modeling environment [11,12]. The required software model layers are first presented. The implementation of these layers is based on specific tools. A brief description of these tools is given along with the application in the literature of these tools to some medium complexity multimedia systems.

3.1. The Sesame Layer's Software Model

Using the Sesame system-level design software framework, three software specification model layers are required: the application process network layer, the target architecture layer, and the layer for mapping the application onto the architecture, as showed in the **Figure 2** [6].

3.1.1. Application Modeling Layer

Applications in Sesame are modeled using the Kahn Process Network (KPN) model of computation in which parallel processes, implemented in a high-level language, communicate with each other via unbounded FIFO channels [13]. Each process is executed sequentially. Reading from channels is blocking; writing to channels is non-blocking. The execution of a Kahn Process Network is deterministic, meaning that for a given input always the same output is produced and the same workload is generated, irrespective of the execution schedule. The model fits nicely with signal processing applications, as it can model stream processing with the guarantee that no data is lost. The key characteristic of the KPN model is that it specifies an application in terms of distributed control and distributed memory which allows us to map the application onto a multiprocessor platform in a systematic and efficient way.

3.1.2. Architecture Modeling Layer

An architecture model is constructed from generic building blocks provided by a library containing template performance models for processors, co-processors, memories, buffers, busses, and so on. The evaluation of architecture is performed by simulating the performance consequences of the application model events that are mapped onto the architecture model. This requires each process and channel of the Kahn process network to be associated with, or mapped onto, one component of the architecture model. When executed, each Kahn process generates a trace of events, and these event traces are routed towards a specific component of the architecture model through a trace event queue. A Kahn process places its application events into this queue while the corresponding architecture component consumes them (**Figure 2**).

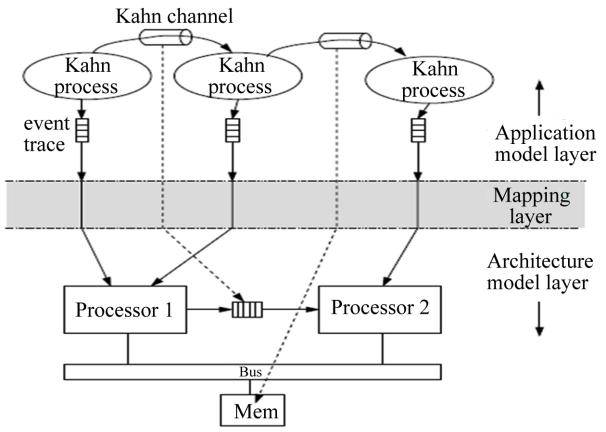

Figure 2. The three layers within Sesame: the application model layer, the architecture model layer, and the mapping layer.

3.1.3. Mapping Layer

The mapping layer maps the event traces generated by the Kahn processes of an application model onto the resources in the architecture model. In addition, it maps the Kahn communication channels onto communication resources at the architecture level. Each Kahn channel can be thus mapped onto a point-to-point FIFO channel between two processors or onto a software buffer in shared memory. As showed in **Figure 2**, it is possible to map multiple Kahn processes onto a single architecture component (e.g., in the case of a programmable component). Such mappings require the events from the event traces that are mapped onto the same architecture resource to be scheduled. This scheduling is also performed by the mapping layer [7].

3.2. Sesame Implementation Tools

In the previous section, we have seen that the Sesame software structure is composed of three layers: the application layer, the architecture layer, and the mapping layer which is an interface between the two previous ones. All three layers in Sesame are composed of components which should be instantiated and connected using some form of object creation and initialization mechanism, as shown in **Figure 3**. This allows reusing of code and guarantees the flexibility to easily manipulate the model based on performance results as dictated by the Y-Chart methodology (**Figure 1**). The three models layers are implemented by the following tools:

3.2.1. YML Modeling Language

Sesame was developed to guarantee a rapid construction of the simulation models thought the use of libraries of pre-built architecture simulation components. In order to enable quick modification, a flexible description format for the interconnection of these components is required. For this, the YML (Y-chart Modeling Language) is defined to create the structure of Sesame's simulation models. YML is an XML-based language. Using XML is attractive because it is simple and flexible, reinforces reuse of model descriptions, and comes with good programming language support. The core elements of YML are network, node, port, link, and property [6].

3.2.2. Application PNRunner Simulator

Sesame's application simulator is called PNRunner, or Process Network Runner. PNRunner implements the semantics of Kahn process networks in C++. It reads an YML application description file and executes the correspondent application model. The PNRunner execution allows generating a trace of application events (trace API) to drive an architecture simulation (**Figure 3**). Using this API, PNRunner can send application events (communication and computation operations) to the architecture simulator where their performance consequences are simulated. Hence, application/architecture trace-events co-simulation is possible.

3.2.3. Architecture Pearl Simulator

The target architecture model in Sesame is implemented in the Pearl discrete event simulation language [14]. Pearl is a small but powerful object-based language which provides easy construction of abstract architecture models and fast performance simulation. It has a C-like syntax with a few additional primitives for simulation purposes. A Pearl program is a collection of concurrent objects which communicate with each other through synchronous or asynchronous message-passing. After sending an asynchronous message, the sending object continues execution, while waiting for a synchronous reply message from the receiver.

3.3. Medium Complexity Media Case Studies

The Sesame modeling and simulation methodology has been applied to two medium complexity media applications: an MPEG-2 decoder and a variant of an M-JPEG encoder [7,8]. These both studies have been performed at the black-box architecture model level and showed promising results. These media applications have been implemented on various multiprocessors architectures models. For these architectures, different hardware-software partitioning, application to architecture mappings, processor speeds, and interconnect structures (bus, Crossbar, and Omega networks) are evaluated [7]. Based on the obtained execution performance results, the Crossbar model is demonstrated better in terms of the measured number of frames per second than the Omega net-

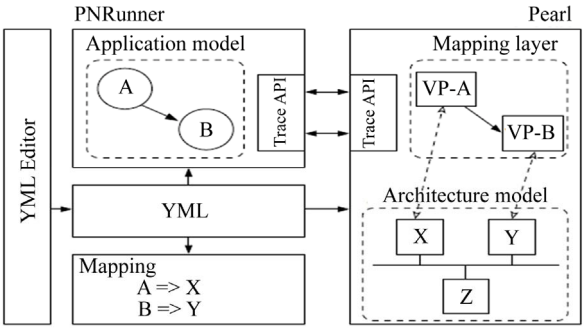

Figure 3. Sesame software tools overview.

work and common bus structures (about 5% faster than the Omega network) [7].

A lot of design space exploration has been so considered for getting the optimal system design. For these medium complexity media applications, the performance evaluation is straightforward and very fast obtained. For all the used configurations, the obtained simulation times range from 5 to 10 seconds. This is many orders of magnitude faster than using classical RTL-level simulators and is very acceptable for design space exploration. Given this, the next sections aim to further demonstrate the effectiveness and the flexibility of the Sesame methods and tools even for the design of very complex media applications. This will be performed by using this methodology for the design and performance evaluation of a H.264/ AVC reference encoder targeting multiple heterogeneous multiprocessors platforms.

4. The H.264/AVC Video Encoder Case Study

The H.264/AVC has been designed with the goal of enabling significantly improved compression performance relative to all existing video coding standards [9]. Such a standard uses advanced compression techniques that in turn, require high computational power [10]. Implementing a H264 video encoder for an embedded SoC requires very high computation power to achieve real-time encoding. This section first presents a complexity analysis of the H.264/AVC reference encoder in comparison to M-JPEG application case. Then, it reviews a performed previous work to get an optimized parallel model of the encoder using an appropriate high-level independent target-architecture parallelization approach.

4.1. Complexity Analysis of the H264/AVC Reference Video Encoder

The complexity of the H.264/AVC encoder application depends on the algorithm, the encoding option tools, the input sequences and the architecture in which it is implemented. In a previous work [15], we performed a complete high level performance and complexity analysis of a H.264/AVC video encoding application. The experiments have been performed on a General-Purpose Processor (GPP) 1.6 GHZ INTEL Centrino platform using the JM 10.2 software reference version [16] with a main profile @ level 4. For an optimal balance between the encoding efficiency and the implementation cost, a proper use of the H.264/AVC tools has been proposed to maintain an acceptable performance while considerably reducing complexity. Using the obtained optimal encoding tools for a very low bit rate 7 frames QCIF "bridge far" sequence, the computing time for the encoding process on the GPP platform is of 15.2 seconds. The associated complexity in frames per second is of 2.16 fps. For this test sequence, the peak memory usage is also measured using the "memprof" GNU profiler [17]. For the used sequence, the obtained peak memory cost is of 5.02 MB. This result refers to none optimized source code. Applying platform independent memory optimizations through C level code transformations may be used to get an optimized memory and algorithmic version of the reference code.

In comparison to the Motion JPEG application presented in Section 2, the non optimized H.264/AVC reference encoder is about two to three orders of magnitude more complex in terms of computing time and memory usage. To illustrate this, the dynamic instruction distribution by operation types have been obtained for both applications using an "objdump" utility and are reported in **Figure 4**. For the H.264/AVC, the dominated instructions types are the "arithmetic" and the "Memory" (Load/ Store) operations. Actually, these results confirm the very high complexity of this new standard, the potential memory allocation needed and the high volume of computation required. The SoC implementation of such a complex application will point out the outcome of the Sesame methodology for the design of such complex systems.

4.2. High Level Parallel Specification of a H.264/AVC Video Encoder

To speedup the computing of this encoder, a multiprocessor implementation is probably needed. Prior to this implementation, the sequential encoding reference C source code [16] should be transformed into concurrent KPN tasks communicating via dedicated FIFO channels. The goal of this step is to extract the available task-parallelism from the application by splitting compute nodes as far as possible to get a valid parallel KPN model of the encoder. For an optimal design flow, it is our aim to provide a parallel specification of the application which forms a good starting point for mapping onto different

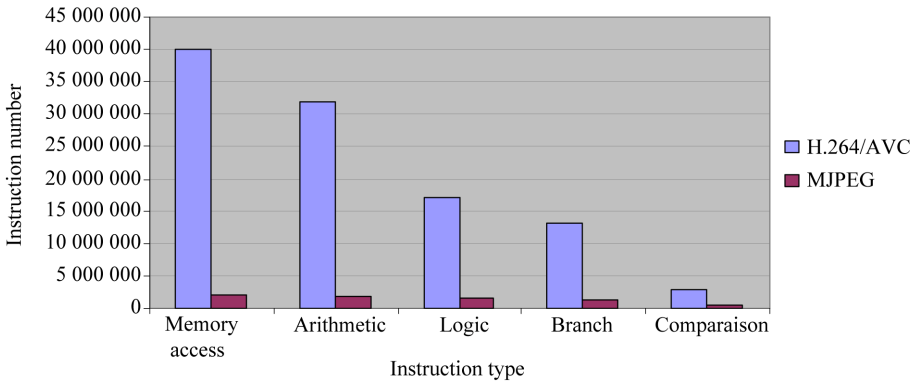

Figure 4. The dynamic instruction distribution by operation types for both the H.264/AVC and M-JPEG applications.

systems-on-chip platforms. To do so, we proposed in a previous work a high-level independent target-architecture parallelization approach [18,19] to get an optimized parallel model of the encoder with the best computation and communication workload balance.

The proposed parallelization approach is based on the use of the KPN/YAPI [20] parallel programming models of computation, and the selection of a fine-grain Macro-Block communication granularity level. The key characteristic of the approach is the simultaneous exploration of the two predominant concepts of parallelism; the data-level partitioning and the task-level splitting and merging. This means that communication and computation workload analysis are needed to provide a global guidance when optimizing concurrency between processes. In general, when the concurrency bottlenecks are identified, task and data levels splitting and/or merging are performed for better distributing the computing workload over the processes. For the most computational-expensive tasks, data splitting is proposed for a better concurrency optimization [18].

Given the proposed parallelization approach, the Task Level Parallelism (TLP) is first considered. The goal of this step is to extract the available task-parallelism by splitting compute nodes as far as possible to get a first starting valid parallel KPN model of the encoder. For this case, the encoder block diagram [21] has served as a starting point for extracting the task-level parallelism. Then to get a parallel implementation of the encoder with the best computation and communication workload balance, different steps of task level splitting or merging and data level splitting are used to derive in a structured way a final optimized model. Further details on the different steps used are given in [19]. Finally, the optimal model obtained is given in **Figure 5**. This figure shows that the low-complexity DCT, Quantification, Decoder, and Filter modules have been merged into only one "Dct_Dec_Filter" process. For the most computational-expensive Motion estimation and compensation "Mec" task, a data partitioning strategy has been considered to distribute the computing of this process into three "Mec1", "Mec2", and "Mec3" processes with tripling of the associated Input/Output FIFO channels.

Given the "Gprof" [22] computation profiling results of the obtained parallel model reported in **Figure 6**, it is clear that the final proposed model has good concurrency properties with an acceptable computation and communication workload balance.

5. System Level Performance Evaluation of the H.264/AVC Video Encoder

This section will show up the effectiveness and the flexibility of the Sesame system level design methodology for efficient and rapid design space exploration of such complex systems. For this, the base target architecture and the mapping strategy are first presented. The sesame system level design is then used for performance evaluation and design space exploration of the encoder targeting multiple multiprocessors architectures. Finally, the efficacy of the methodology is evaluated for the design of very complex systems.

5.1. The Base Target Architecture and Mapping

Starting with the Sesame system-level design methodology presented in Section 2, three software model specifications are required: the application process network model, the target architecture model, and the mapping model of the application onto the architecture. For this, the optimized parallel model of the H.264/AVC encoder of **Figure 5** is first ported to the Sesame framework. This has been performed by transforming the previously validated YAPI model into a C++ PNRunner network model. The obtained network model is then simulated with the PNRunner simulator to generate a computational and communication event traces of the application execution, called trace-event queues [6].

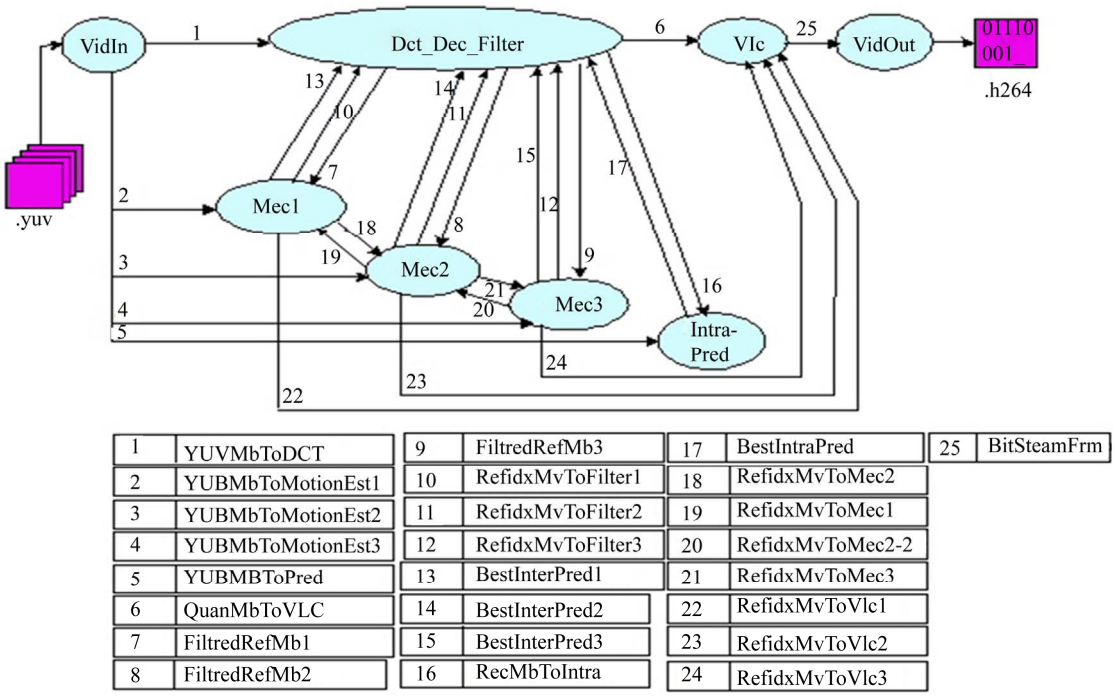

Figure 5. Proposed optimized parallel KPN model of the H.264 encoder.

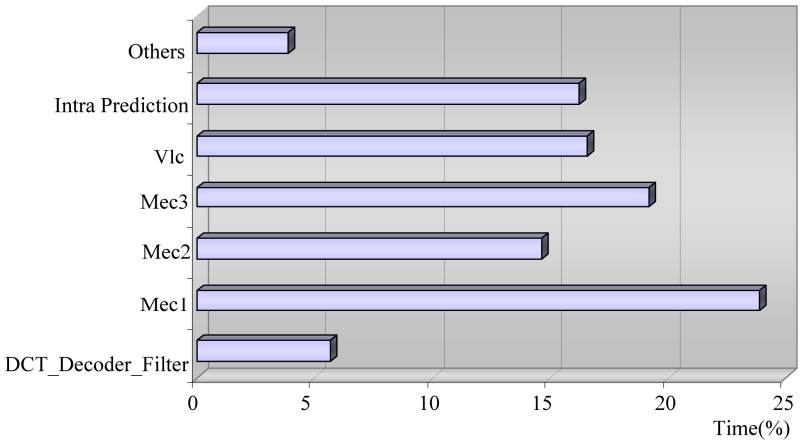

Figure 6. Computation workload profiling of the final parallel model.

Parallel to the application model specification, the target architecture is modeled with the Pearl object-based simulation language. The Sesame environment provides a small library of architecture black-box base models: processing cores, a generic bus, a generic memory, and several interfaces for connecting these base model building blocks. Once a target architecture model is validated, a trace-driven co-simulation of the application events traces queues mapped to the architectural components is carried out. Such a co-simulation requires an explicit mapping of the KPN processes and channels to the particular components of the target architecture. More than one KPN process can be mapped to a same processor as the system simulator automatically schedules the events from the different queues.

In our case, the base target architecture is given in **Figure 7** that represents a multiprocessor platform communicating with a shared DRAM memory through a common bus. For this platform, we have used general purpose processors (assumed to be MIPS R3000), and assumed a conservative timing of 100 ns to read/write a 64-bit word from/to DRAM. The instruction latencies for the MIPS R3000 microprocessors components were estimated using technical documentation. Communication between components is performed through buffers in shared memory.

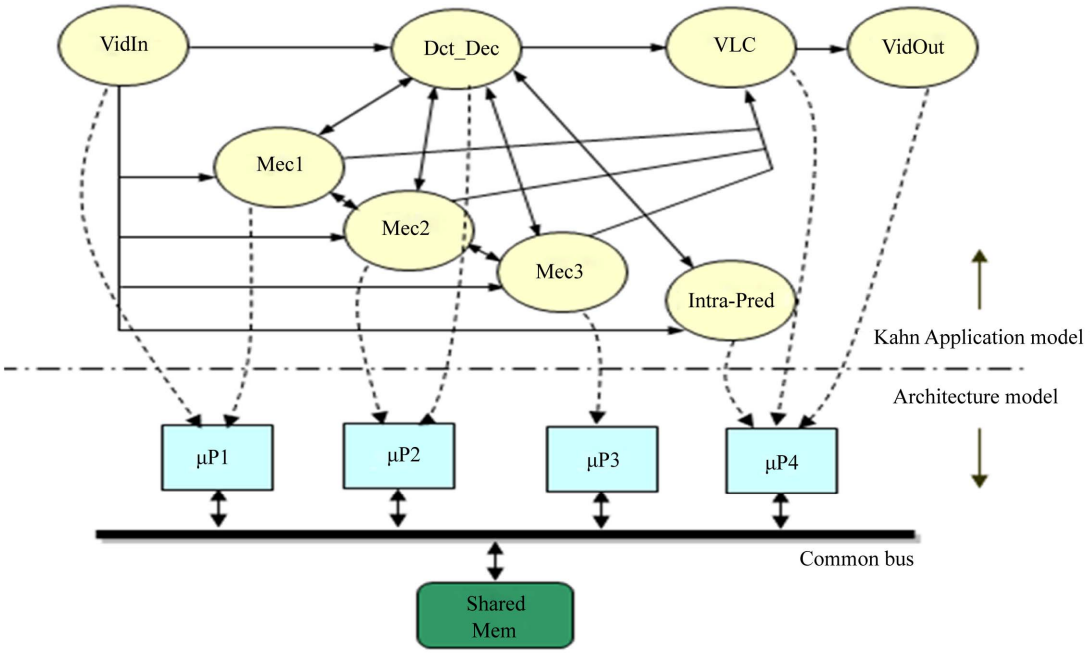

Figure 7. H.264 encoder's application to architecture mapping.

For sufficient design space exploration, several platform models are used. The platforms differ by the number of used processors. One platform is used with two processors; a second is with four and a third is tested with six processors. Given the optimized parallel model of the H.264/AVC encoder, the Sesame design space exploration consisted in changing the mapping combination or adding another architectural component without touching the H.264 encoding parallel specification since this application presents already good concurrency properties with an acceptable computation and communication workload balance [19]. When such a system modification is performed, we have to recompile first the hardware architecture, and then the entire system to regenerate the new YML files of the target architecture and mapping layers. Adding a new architectural component consist in acceding through an "YMLEditor" YML graphical editor to the Sesame library of black-box components models and after that making by simple clicks its addition to the architecture model and its connection to the bus. The "YMLEditor" editor is also used to project quickly the application tasks and Kahn communications channels on the different architecture resources.

Mapping application processes to this platform has been decided explicitly given the obtained computation and communication load distribution results of **Figure 6**. For the bi-processor platform example, the total computation load has been distributed between the two processors. The "Mec1", "Mec2", and "Dct_Dec_Filter" processes are mapped to one, and all the others to the second processor. The mapping strategy used with the four processors platform is showed in **Figure 7**. In this case, first, the most complex "Mec1", "Mec2", "Mec3", and "Intra-Pred" tasks are mapped separately to each used core to guarantee a competitive execution between them. Then, the "Dct_Dec_Filter" process is added to run with the "Mec2" process on the same core. The "Vlc" is also added to the "Intra-Pred" process and is mapped to the fourth processor.

5.2. System Level Performance Evaluation and Design Space Exploration

After having mapped the PNRunner optimized H264/AVC network model to the different used platforms with two four and six microprocessors, the performance analysis step is performed by system-level simulations. In all the experiments, the input test video sequence consists of YUV frames captured in a QCIF resolution of 176 * 144 pixels. The simulation results of the QCIF "Bridge-close" sequence H.264/AVC encoding process are obtained for the different used platforms and are presented in the following **Figure 8**. It is clear from this figure, that the encoding performances obtained in frames per second are getting better linearly when the number of simulated microprocessors is increased. For each case, as the application model is considered to be optimal, the execution/communication performances gain may be improved by changing the mapping policy or/and the platform architecture.

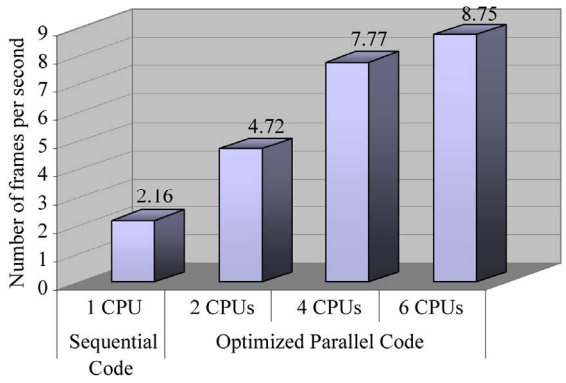

Figure 8. H.264/AVC encoding performances vs. simulated processors with the common bus structure.

To modify the architecture, a designer can also explore the use of other communication models or enhance the architecture with hardware components using appropriate HW/SW partitioning. For example, for the four-processor platform with the common bus structure, performance numbers for the execution/communication workload is obtained for each used architecture component. The obtained results are shown in **Figure 9**. For each component, a bar shows the breakdown of the time spent on reading/writing, being busy and being idle. Given this figure, it is obvious that the computation cost is much more important than the time spent in reading/writing from/to the shared memory. The communication and computation loads are nearly balanced for all the used components. Such a result confirms the good concurrency properties of the proposed optimized parallel application model and the appropriate used mapping policy. However, the times being idle are too much important in comparison with the times being busy for all the architecture components. This has caused probably a substantial degradation of the final encoding performances. Given the important amount of data communicated between processes for this encoding process, it is clear that the common memory bus structure constitutes a serious communication bottleneck. Indeed, the very important data dependency between processors requires a potential memory access and allocation for the read/write operations. For a common-bus-based multiprocessor architecture, this causes a saturation of bus and thus a lot of time spent in waiting to read/write data from/to other component.

For further design space exploration and in order to reduce the communication bottleneck observed for the common-bus-based architecture, others inter processors communication structures and topologies should be tested. In the Sesame framework, a Crossbar and an Omega network Pearl model structures are implemented [7]. Given this, we selected in our experiments the crossbar switch structure in replacement of the common bus model. A Pearl simulation model of a 4 × 4 crossbar switch is implemented, as shown in **Figure 10**. For the obtained architecture of **Figure 10**, the processors communicate with each other over the crossbar. The memory is distributed per processor and resides in the Virtual Buffers (VBs). Data is written to the virtual buffer associated with the writing processor. Only reads are forwarded over the crossbar, and, it is possible to use it for write calls also. The performance results are obtained for the different used platforms and are presented in the following **Figure 11**. As shown in **Figure 11**, the use of the Crossbar structure come up with a substantial performance encoding gain in frames per second (fps) in comparison with the common bus architecture. In effect, we achieved the 9.6 fps with six processors (MIPS R3000) connected via a 4 × 4 crossbar communication model. In addition, for the four-processor platform, the execution/communication workload is obtained for each used component. The obtained results are reported in **Figure 12**. The performance numbers statistics of **Figure 12** clearly show that the components spend much more time being busy doing work and more less time waiting for reading and writing. This confirms the performance gain obtained.

5.3. Evaluation of the Methodology for Rapid Design Space Exploration

The Sesame framework has been used for the design of a very complex media application verifying constraints. Given the complexity of case studied, the previous section outlined the difficulty to evaluate one design using detailed clock cycle-accurate simulators. For the system level design case, the simulation times did not take more then 5 minutes for all the used configurations. Measurements have been done on a General-Purpose Processor (GPP) platform based on an INTEL Centrino 1.6 GHZ with 512 MB RAM memory running a Linux operating system. In comparison to classical RTL-level simulators, this is many orders of magnitude faster and is acceptable for design space exploration.

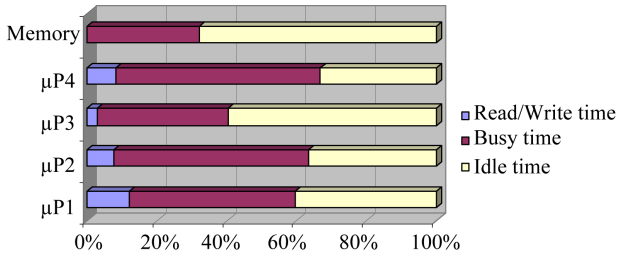

Figure 9. Reading-Writing/Execution/Idle statistics for the common-bus-based architecture.

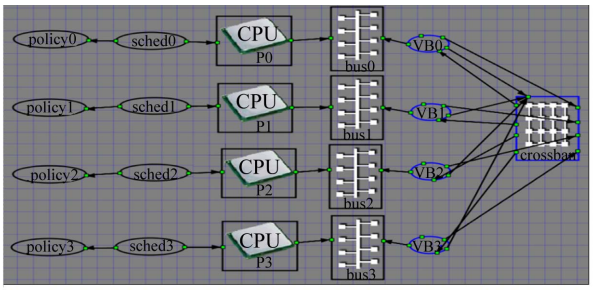

Figure 10. Used Crossbar-structure-based four processors architecture model.

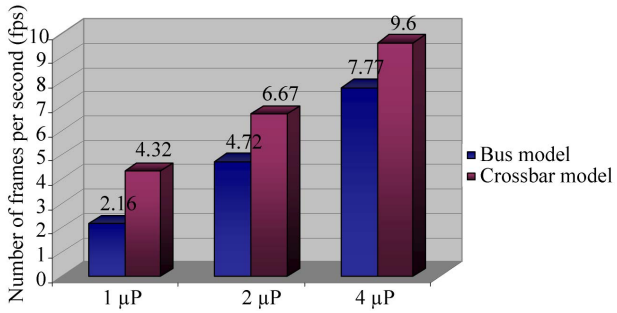

Figure 11. H.264 encoding performances vs. simulated processors with the Crossbar model.

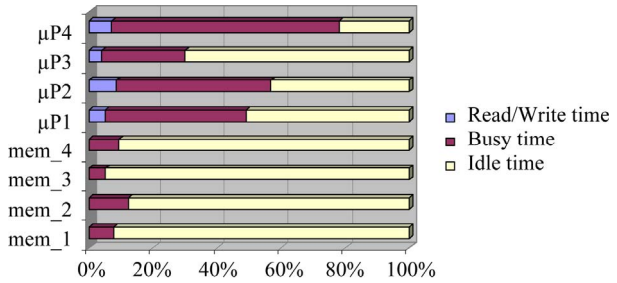

Figure 12. Reading-Writing/Execution/Idle statistics for the crossbar-model-based architecture.

Due to the simplicity and expressive power of Sesame's Pearl simulation language, modeling all the platform architectures has been rapidly performed. Indeed, the system-level modeling relieves the designer from low level implementation details. Performance evaluation at high abstraction levels makes it possible to control the speed, required modeling effort, and attainable accuracy of the simulations. This enables to efficiently explore the large design space in the early design stages. Applying more detailed models at a later stage allows focused architectural exploration.

Finally, we find that the Sesame methodology facilitates the performance analysis of embedded systems architectures in a way that directly reflects the Y-chart design approach. Essential in this modeling methodology is that an application model is independent from architectural specifics, assumptions on hardware/software partitioning, and timing characteristics. Thus, the application is studied in isolation by means of a functional (behavioral) software model written in a high level language. Given the complexity of the case studied, an appropriate parallelization methodology has been separately and undependably proposed to get the optimal application model with the best computation and communication workload balance [19]. This results in a good starting model with primary estimations of its performance requirements. As a result, a single optimally and separately designed application model has been used to exercise different mappings onto a range of architecture models. This clearly demonstrates the strength of decoupling application models and architecture models and it enables the reuse of both types of models.

6. Conclusions

In this paper, we motivated the use of the Sesame/Artemis system-level design methodology for efficient architectural exploration of the increasing complexity heterogeneous embedded media systems. The case studied is concerned with an optimal H.264/AVC encoder SoC design verifying constraints. The complexity analysis of the H.264/AVC reference encoder confirmed the very high complexity of this new standard, the potential memory allocation needed and the high volume of computation required. For the design of this complex media system, the outcome, the effectiveness and the flexibility of the methods and tools provided by the Sesame methodology have been clearly illustrated. Both modeling and mapping stages of the Sesame design flow are explored. A lot of design space exploration has been considered for getting an optimal design. For all the used configurations, the simulation times did not take more then 5 minutes for all the used configurations. In addition, due to the simplicity and expressive power of the architecture specification language, modeling all the proposed platform architectures has been rapidly performed. This enables to efficiently explore the large design space in the early design stages.

7. References

[1] C. Erbas, "System-Level Modeling and Design Space Exploration for Multiprocessor Embedded System-on-Chip Architectures," Ph.D Thesis, University of Amsterdam, Amsterdam, 2006.

[2] E. A. Lee, *et al.*, "Overview of the Ptolemy Project," Technical Memorandum UCB/ERL M01/11, University of California, Berkeley, May 2001.

[3] X. Chen, H. Hsieh and F. Balarin, "Verification Approach of Metropolis Design Framework for Embedded Sys-

tems," *International Journal of Parallel Programming*, Vol. 34, No. 1, February 2006, pp. 3-27. doi:10.1007/s10766-005-0002-x

[4] L. Cai and D. D. Gajski, "C/C++ Based System Design Flow Using SpecC, VCC and SystemC," Technical Report CECS_02_30, 14 June 2002.

[5] J. Coffland, "SESAME Users Guide," Technical Report, University of Amsterdam, 5 April 2006. http://sesamesim.sourceforge.net/docs/SESAME/SESAME_Users_Guide.html

[6] J. E. Coffland and A. D. Pimentel, "A Software Framework for Efficient System Level Performance Evaluation of Embedded Systems," *Proceedings of the* 2003 *ACM Symposium on Applied Computing*, Melbourne, March 2003. doi:10.1145/952532.952663

[7] A. D. Pimentel, S. Polstra, F. Terpstra, A. W. van Halderen, J. E. Coffland and L. O. Hertzberger, "Towards Efficient Design Space Exploration of Heterogeneous Embedded Media Systems," Technical Report, Department of Computer Science, University of Amsterdam, 2001.

[8] P. van der Wolf, P. Lieverse, M. Goel, D. La Hei and K. Vissers, "An MPEG-2 Decoder Case Study as a Driver for a System Level Design Methodology," *Proceedings of the* 7*th International Workshop on Hardware/Software Codesign*, Rome, 3-5 May 1999, pp. 33-37. doi:10.1145/301177.301196

[9] E. G. Iain and H. Richardson, "264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia," John Wiley &Sons Ltd, Hoboken, 2003.

[10] M. Alvarez, A. Salami, A. Ramirez and M. Valero, "A Performance Characterization of high Definition Digital Video Decoding Using H264/AVC," *Proceedings of IEEE International*, *Symposium on Workload Characterization*, Austin, 6-8 October 2005, pp. 24-33.

[11] A. D. Pimentel, P. Lieverse, P. van der Wolf, L. O. Hertzberger and E. F. Deprettere, "Exploring Embedded-Systems Architectures with Artemis," *IEEE Computer*, Vol. 34, No. 11, November 2001, pp. 57-63.

[12] P. Lieverse, P. van der Wolf, E. F. Deprettere and K. A. Vissers, "A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems," *Journal of VLSI Signal Processing for Signal, Image and Video Technology*, Special Issue on SiPS'99, Vol. 29, No. 3, November 2001, pp. 197-207.

[13] G. Kahn, "The Semantics of a Simple Language for Parallel Programming," In: J. L. Rosenfeld, Ed., *Information Processing*, *Proceedings of the IFIP Congress* 74, North-Holland Publishing Co., Stockholm, 5-10 August 1974.

[14] H. Muller, "Pearl: A Language for Architecture Simulation," 25 February 1993. http://www.pearl.org/

[15] H. Krichene Zrida, A.C. Ammari, A. Jemai and M. Abid, "Performance/Complexity Analysis of a H264 Video Encoder," *International Review on Computers and Software*, Vol. 2, No. 4, July 2007, pp. 401-414.

[16] H264 Reference Software Version JM 10.2, November 2005. http://iphome.hhi.de/suehring/tml/.

[17] MemProf—Profiling and leak detection, July 2008. http://www.gnome.org/projects/memprof/

[18] H. Krichene Zrida, A. C. Ammari, A. Jemai and M. Abid. "A YAPI-KPN Parallel Model of a H264/AVC Video Encoder," *Proceedings of* the 4*th IEEE International Conference on Ph.D Research in Microelectronics and Electronics*, Istanbul, 22-25 June 2008, pp. 109-112.

[19] H. K. Zrida, A. Jemai, A. C. Ammari and M. Abid, "High Level H.264/AVC Video Encoder Parallelization for Multiprocessor Implementation," *Proceedings of the* 12*th ACM/ IEEE Design Automation and Test in Europe Conference and Exhibition*, Nice, 20-24 April 2009, pp. 940- 945.

[20] E. A. Kock, G. Essink, W. J. M. Smits, P. van der Wolf, J. Y. Brunel, W. M. Kruijtzer, P. Lieverse and K. A. Vissers, "YAPI: Application Modeling for Signal Processing System," *Proceeding* 37*th Design Automation Conference*, Los Angeles, 5-9 June 2000, pp. 402-405. doi:10.1109/DAC.2000.855344

[21] R. Schäfer, T. Wiegand and H. Schwarz, "The Emerging H264/AVC Standard," EBU Technical Review, January 2003.

[22] S. L. Graham, P. B. Kessler and M. K. McKusick, "Gprof: A Call Graph Execution Profiler," *Proceedings of the SIGPLAN*' 82 *Symposium on Compiler Construction*, Boston, 23-25 June 1982. http://sourceware.org/binutils/docs/gprof/(October 2009)

