Original scientific paper

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 46, No. 4(2016), 219 – 228

MDE-based Rapid DSE of multi-core embedded systems: The H.264 Decoder Case Study

Manel Ammar[1], Mouna Baklouti[1], Maxime Pelcat[2], Karol Desnos[2] and Mohamed Abid[1]

[1]CES Laboratory, National Engineering School of Sfax, Sfax, Tunisia
[2]IETR, INSA Rennes, CNRS UMR 6164, UEB, Rennes, France

Abstract: The recent advances in Unified Modeling Language (UML) give a valuable milestone for its application to modern embedded systems design space exploration. However, it is essential to remember that UML is unable to solve the difficulty associated with embedded systems analysis, but it only provides standard modeling means. A reliable Design Space Exploration (DSE) process which suits the peculiarities of complex embedded systems design is necessary to complement the use of UML for design space exploration. In this article, we propose a Model Driven Engineering-based (MDE) co-design flow that combines high-level data-intensive application analysis with rapid prototyping. In order to specify the embedded system, our methodology relies on the Modeling and Analysis of Real-Time and Embedded Systems (MARTE) UML profile. Moreover, the present contribution uses the Parameterized and Interfaced Synchronous Dataflow (πSDF) Model-of-Computation (MoC) and a model based on the IP-XACT standard as intermediate levels of abstraction to facilitate the analysis step in the co-design flow. The rapid prototyping process relies on the πSDF graph of the application and a system-level description of the architecture. This paper presents our Hw/Sw co-specification methodology, including its support for gradual refinement of the high-level models towards lower levels of abstraction for design space exploration purposes.

Keywords: Co-Design; MP2SoC; MDE; MARTE; πSDF; S-LAM; PREESM; SoC

Hiter DSE večjedrnih vgrajenih sistemov na osnovi MDE: Primer H.264 dekoderja

Izvleček: Najnovejši napredki poenotenega modelirnega jezika (UML) ponujajo pomembne mejnike pri raziskovanju načrtovalskega prostora modernih vgrajenih sistemov. Poudariti pa je potrebno, da ULM ne rešuje problemov analize vgrajenih sistemov temveč določa le standard pri njihovem načrtovanju. Zanesljivo raziskovanje načrtovalskega prostora (DSE), ki ustreza posebnostim kompleksnih vgrajenih sistemov je potrebno za dopolnilno uporabo ULM. V članku predlagamo modelno gnan načrtovalni potek na osnovi inženirskega pristopa, ki združuje analizo podatkovno intenzivne aplikacije na visokem nivoju s hitrim izdelavi prototipov. Določitev vgrajenega sistema temelji na ULM profilu modeliranja in analize vgrajenih sistemov v realnem času. Dodatno, predlagana rešitev vključuje parametiziran in z vmesnikom sinhroniziran (πSDF) model toka podatkov (MoC) in model na osnovi IP-XACT standarda vmesnih nivojev za pospešitev korakov analize v poteku načrtovanja. Hitra izdelava prototipov temelji na πSDF grafih aplikacije in na opisu arhitekture sistemskega nivoja. Članek predstavlja programsko/strojno metodologijo, skupaj s podporo postopne izboljšave od modelov višjih nivojev do nizkih nivojev abstrakcije raziskovanja načrtovalskega prostora.

Ključne besede: so-načrtovanje; MP2SoC; MDE; MARTE; πSDF; S-LAM; PREESM; SoC

** Corresponding Author's e-mail: manel.ammar@ceslab.org*

1 Introduction

At the present time, Massively Parallel Multi-Processors System-on-Chip (MP2SoC) are commonly dedicated to data-intensive processing applications where huge amounts of data are handled in a regular way by means of repetitive computations. As performance presents an important feature of emerging MP2SoCs, the design of such systems should meet strict time-to-market and cost constraints, while holding the guarantee of rising performance through parallelism.

Performance relies on a diverse set of factors (granularity of the application, model of the architecture, partitioning and allocation choices, etc.) and parameters (number of processing units, memory sizes, etc). Design Space Exploration (DSE) means adjusting these factors and parameters while taking into account a set of metrics (execution time, latency, throughput, energy, etc.) to find the optimal combination between the MP2SoC architecture and the data-intensive processing application at an early phase of the system design. The DSE of complex embedded systems involves three issues which are:

- The modeling effort: which depends on the specification methodology
- The evaluation effort: which depends on the performance estimation techniques and tools
- The results accuracy: which depends on the exploration strategies that reduce the vast design space while reaching accurate performance numbers

Research on the DSE of modern applications running on complex System-on-chip (SoC) is still emerging. Several design frameworks have been suggested enabling high-level system specification. Based on the Model Driven Engineering (MDE) guidelines, the Unified Modeling Language (UML) [1] semantics and the Modeling and Analysis of Real-Time and Embedded Systems (MARTE) [2] profile annotations, these frameworks guarantee a model-based specification methodology that stresses the use of models in the embedded systems development life cycle and argues automation via meta-modeling, model transformation and code generation techniques.

In addition, state-of-the art DSE frameworks rely on different evaluation techniques and exploration strategies. A common practice of embedded systems performance estimation in these approaches [3, 4, 5] is simulation. Although the simulation approach is more accurate, it is often time-consuming to be involved inside the design space exploration loop. Moreover, it requires well-defined rich input models imposing extensive specification efforts. The COMPLEX framework [3], for example, uses MDE foundations for the co-design of embedded systems, but, it directly generates executable files of the system to run a simulation-based DSE process. Providing such an executable model at an early phase of the design process may introduce an unjustified burden when making early design decisions.

On the contrary, analytical design space exploration approaches [6, 7, 8] do not depend on simulators or on running code on real hardware. They rather take high-level specification of the embedded application, combine it with high-level model of the architecture and perform a static analysis to obtain performance measurements for this combination. For this reason, we propose a purely analytical approach based on the high-level analysis of the embedded system. While building a simulation model is computationally costly, analytical estimations can be considered to accelerate the design process.

In this paper, we propose an automatic approach that takes advantage from MDE and MARTE and defines two levels of abstraction that alleviate the analysis and generation of data-intensive processing applications running on multi-processor architectures. The first level is based on a novel extension of the famous Synchronous Data Flow (SDF) [9] Model-of-Computation (MoC), the Parameterized and Interfaced Synchronous Dataflow (πSDF) [10] model. Another level is introduced in our platform-based co-design flow facilitating IP integration, architecture generation and system analysis. This level complies with a model based on the IP-XACT standard [11] named System-Level Architecture Model (S-LAM) [12]. High-level MARTE-based specification of the parallel architecture can be then refined in an MDE-based process to produce S-LAM description of the platform. These two abstraction levels were integrated with the PREESM [13] system-level rapid prototyping tool in an MDE-based co-design flow for complex embedded systems design.

In previous work [14], the UML/MARTE methodology for modeling the data-parallel application and the automatic generation of the πSDF specification have been presented. In [15] the automatic generation from the UML/MARTE specification of the S-LAM description of the architecture was explained. In this paper, a complete overview of the proposed DSE flow is given in Section 2. Moreover, the paper contributes new features not addressed in previous work, specifically the integration of the PREESM rapid prototyping tool inside the DSE framework and the validation of the proposed DSE methodology using a more complex case study (the H.264 decoder) which will be addressed in Section 3.

2 The proposed co-design flow

Accurate performance numbers can be reached at the cost of very detailed modeling. On the other hand, a moderate effort for modeling leads to a high-level evaluation task, but the accuracy is lost. In the proposed co-design framework, a balanced tradeoff has been made between design space exploration performance and accuracy allowing for extremely rapid system-level analysis while still yielding reliable estimations. In fact, our framework (Figure 1) is a complete and automatic Computer-Aided Design (CAD) tool for the co-specification, design space exploration and code generation of MP2SoC systems that totally relies on MDE techniques. Being based on the Eclipse framework, front-end, transformation chains and back-end tools are grouped together in a fully-integrated flow.

2.1 UML/MARTE front-end: modeling concepts

The proposed co-specification methodology supports the description of the architecture, the application, the allocation, and the deployment within a unified UML model. Description of the architecture, the application, and the allocation are declared by means of UML classes (class diagram and composite structure diagram) annotated with MARTE stereotypes (Table 1). Deployment of software and hardware IPs, describing implementation details of application tasks and architecture components, are described using the UML deployment diagram decorated by stereotypes of our proposed deployment profile as shown in Table 1.

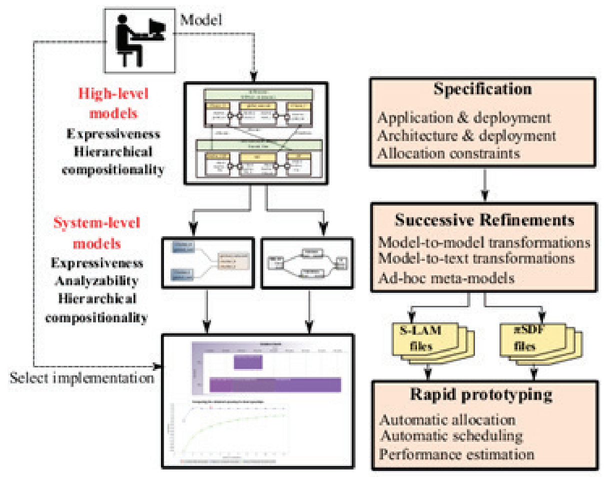


Figure 1: Our proposed co-design flow

2.2 Refinements and abstraction levels inside the proposed flow

Two MoCs were introduced in the proposed co-design flow to prune the design-space exploration step: πSDF and S-LAM.

2.2.1 The πSDF MoC

Generally, MoCs can be evaluated based on their expressiveness and analyzability. SDF MoC proved to be a very successful mean providing a good degree of expressiveness while offering a lot of potential analysis [16] [17].

This combination makes this MoC very motivating in the domain of multimedia applications for embedded systems since throughput, storage requirements and latency can be easily estimated using analysis methods. Being able to specify complex hierarchic and parametric dataflow-based applications, the πSDF MoC extends the SDF MoC while preserving expressiveness and analyzability features. This MoC promotes rapid design space exploration and reconfigurable resource allocation of heterogeneous multicore systems. A πSDF graph is a directed graph represented by a tuple G=(A,F,I,π,Δ), where A is a set of actors and F is a set of FIFOs. The hierarchical compositionality mechanism is based on the set of hierarchical interfaces I. Furthermore, dynamism in πSDF relies on π and Δ, which describes respectively the set of parameters and their dependencies.

2.2.2 The S-LAM MoC

To be properly analyzed and prototyped, the hardware architecture part of a given embedded system needs to be described at system-level. The S-LAM MoC, which facilitates such specifications, allows a simple and ex-

Table 1: Used MARTE subset for the architecture (HW), the application (Sw), and the allocation models

Concept	Stereotype	Package	Model
Processing resource	HwProcessor	MARTE:HRM	Hw
Storage resource	HwMemory	MARTE:HRM	Hw
Communication resource	HwCommunicationResource	MARTE:HRM	Hw
Task	SwSchedulableResource	MARTE:SRM	Sw
Communication port	FlowPort	MARTE:GCM	Hw, Sw
Repetitive component	Shaped	MARTE:RSM	Hw, SW
Complex link topology	Tiler	MARTE:RSM	Hw, SW
Complex link topology	Reshape	MARTE:RSM	Hw, SW
Simple allocation	Allocate	MARTE:Alloc	allocation
Repetitive allocation	Distribute	MARTE:Alloc	allocation
Hardware component	HwResource	MARTE:HRM	Hw
Hardware IP	HwIP	Deployment	deployment
Software IP	SwIP	Deployment	deployment

pressive description while enabling rapid simulations. Being compatible with the IP-XACT model, the S-LAM meta-model does not use the entire IP-XACT meta-model, but it exploits a sub-set of concepts that capture the needed information for the exploration phase [15].

There are two main motivations behind the use of S-LAM as intermediate abstraction level:
Simplicity: S-LAM knows nothing about the implementation details of each component of the hardware architecture while detailing its primary properties
Compositional: S-LAM makes the hierarchical description of the system possible which facilitates the specification of massively parallel architectures complex structure.

2.2.3 Refinements using transformation chains

Three transformation chains were defined in our co-design flow. The first transformation generates the πSDF description of the data-parallel application. The second transformation chain generates an S-LAM compliant description of the parallel architecture. And the third chain generates a scenario file, the third design entry of the rapid prototyping tool. The implementation of a transformation flow in the MDE approach relies on the definition of ad-hoc meta-models for each abstraction level. For this reason, three meta-models were proposed: the MARTE and Deployment meta-model, the πSDF meta-model and the S-LAM meta-model. In addition, model-to-model and model-to-text transformations were defined inside the transformation chains as depicted in Figure 2. In our approach, model-to-model transformation rules are defined using the QVTO language [18] and model-to-text transformation rules are described using the Acceleo tool. Following the MDE principles, an automatic transformation was developed to generate a MARTE-compliant model from a UML-based specification. The first model-to-model transformation produces generic models of the application, the architecture and the allocation conform to the MARTE meta-model.

The MARTE to πSDF transformation chain [14] takes as input the generic application model resulting from the first transformation and generates as output a πSDF specification which conforms to the πSDF meta-model. The S-LAM transformation chain [15] generates a model that conforms to the S-LAM meta-model taking as entry point the generic architecture model.
The generated πSDF and S-LAM models should be processed by two model-to-text transformations to produce .pi and .slam files of the application and the architecture. The generation of a scenario file aims at separating the algorithm and architecture constraints from system-level models.

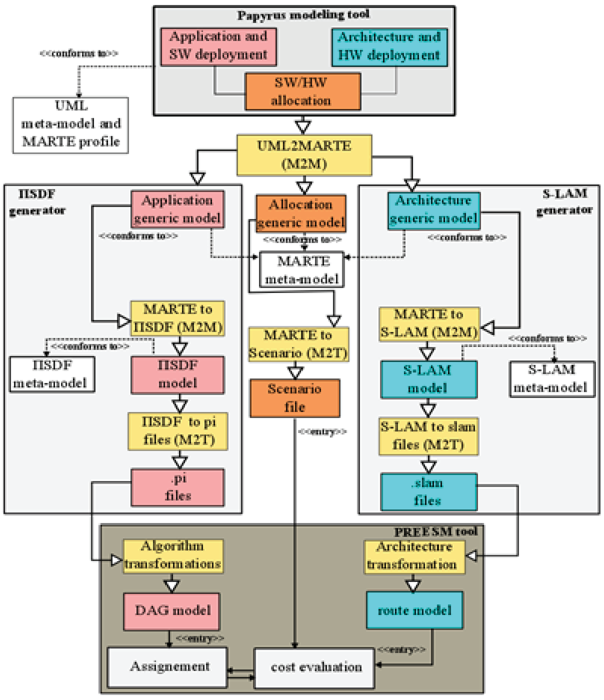


Figure 2: Refinements and abstraction levels inside the proposed flow

2.3 Rapid prototyping with PREESM

Based on the previously described steps (co-specification, successive refinements), we described a multi-level design space exploration methodology that relies on high-level models and refinement chains to enhance the rapid analysis of high-performance embedded systems. The final step in the proposed approach is the rapid prototyping of the πSDF/S-LAM combination using PREESM which will be described in this section.

The flexible rapid prototyping process in PREESM consists of exploring the design tradeoffs at system-level while taking into account system constraints and objectives present in the scenario file. The central feature of the rapid prototyping method is the multi-core scheduler. Before starting the scheduling phase, PREESM performs two transformations aiming to expose the parallelism of the application and the architecture. In the one hand, the πSDF graph is transformed into a Hierarchical SDF, then into a Homogeneous SDF and finally into a Directed Acyclic Graph (DAG). The latter will be processed by the scheduler. On the other hand, a route model is generated from the S-LAM model aiming to facilitate the allocation task. The NP-complete scheduling process in PREESM consists of two separate operations:

- Assignment: relies on the DAG model of the application to assign actors to operators

- Cost evaluation: relies on the route model and the scenario to estimate the cost of the proposed solution

Such a scheduling process must satisfy both the data dependencies between tasks of the application and the execution constraints imposed by the execution platform. It also increases predictability, and allows precise performance estimations. At the end of the scheduling process, a Gantt chart of the execution is displayed plotting the optimal schedule. Memory storage requirements and speedup values are also estimated and plotted in different charts.

3 Experimental results

Our objective is to illustrate the effectiveness of the proposed co-design framework in terms of rapidity and accuracy of the exploration results. The H.264 decoder application [19], a typical data-intensive signal processing application, is chosen to demonstrate the efficiency of the proposed exploration tools. We mainly focus on a coarse-grain parallelization technique implemented in the literature [20] and try to predict the advantage of running such complex application on massively parallel architectures.

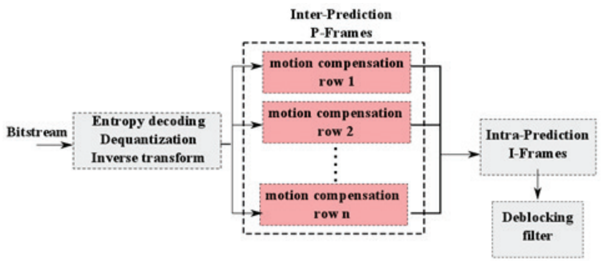


Figure 3: Parallel motion compensation application block diagram

3.1 The H.264/AVC decoder

Among numerous video compression standards, H.264 seems to be very effective in terms of compression and quality. Providing a compression efficiency gain of 50% compared to previous standards, the H.264 codec proves its effectiveness in high definition systems as well as low resolution devices. The H.264 AVC decoder splits each frame of a given video sequence into macroblocks (blocks of 16 × 16 pixels). These macroblocks are decoded in raser scan order using intra-prediction, inter-prediction and deblocking filter.

With the uncontrollably evolution of video resolutions, the processing time of this decoder keeps increasing. Executing such complex application on parallel cores should solve this problem. However, dependencies, data coherency and synchronization introduced in the intra-prediction, inter-prediction and deblocking filter kernels are challenging characteristics making the parallelization task very hard.

In recent years, coarse-grain and fine-grain parallelization techniques were proposed. Coarse-grain methods allow decoding groups of pictures, frame or slices in parallel. Fine-grain techniques investigate smaller units named macroblocks.

In this case study, we aim to study the motion compensation parallelization technique [20]. This technique divides the frame into rows of independent macroblocks as shown in Figure 3. The motion compensation stage is processed for each row of the frame. The decoder process begins with the entropy decoding. Then, dequantization and inverse transformation are executed on the resulting data. Afterward, every row of macroblocks of a frame is inter-predicted (motion compensation task). At the end, the deblocking filter is applied.

3.2 The target model of the architecture

MP2SoC, as presented in Figure 4, is composed of a parametric number of processing elements (PE), grouped into two clusters. The clusters can communicate via a global interconnection network. The first cluster, cluster0, contains one processing element connected to its local memory and can act as a global controller of the architecture. Inside the second cluster, each processing element is connected to its local memory and can communicate to other processors via a local network. The presented architecture [21] is parametric and configurable to satisfy a wide range of systematic signal processing applications. Its design is based on IP assembly approach.

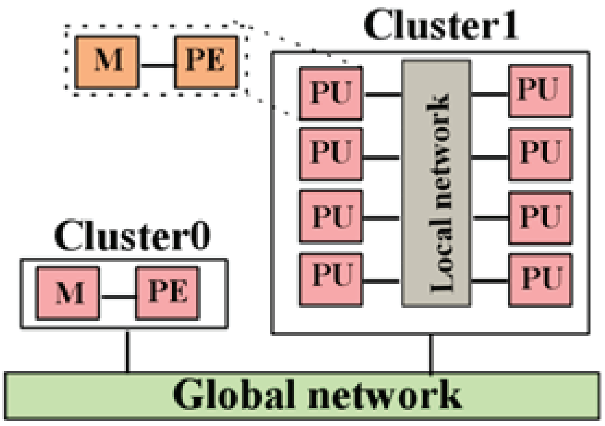
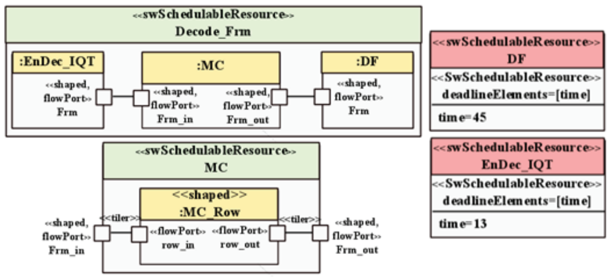


Figure 4: MP2SoC architecture

3.3 Modeling the H.264 decoder

In Figure 5, the shaped stereotype associated with the instance of the MC_Row class denotes the data-parallelism of the application. Each repetition of the motion compensation task consumes one input pattern and produces one output pattern. A pattern corresponds to a row of macroblocks.



<<swSchedulableResource>>
Decode_Frm
:EnDec_IQT
<<shaped, flowPort>> Frm
:MC
<<shaped, flowPort>> Frm_in
<<shaped, flowPort>> Frm_out
:DF
<<shaped, flowPort>> Frm
<<swSchedulableResource>>
MC
<<shaped>>
:MC_Row
<<shaped, flowPort>> Frm_in
<<tiler>>
<<flowPort>> row_in
<<flowPort>> row_out
<<tiler>>
<<shaped, flowPort>> Frm_out
<<swSchedulableResource>>
DF
<<SwSchedulableResource>>
deadlineElements=[time]
time=45
<<swSchedulableResource>>
EnDec_IQT
<<SwSchedulableResource>>
deadlineElements=[time]
time=13

Figure 5: Parallel motion compensation in UML

Table 2 summarizes the multiplicities of consumed and produced patterns inside the Decode_Frm and MC classes where X and Y present the number of macroblocks in the horizontal and vertical directions, and nbrow specifies the number of rows processed in parallel. To guarantee accuracy of the exploration results, the deadlineElements attribute of the SwSchedulableResource stereotype is used to specify how much of program execution time each elementary task used as seen in Figure 5.

3.4 Modeling the MP2SoC architecture

Figure 6 shows the UML specification of the MP2SoC architecture. The main component of the architecture, named MP2SoC_Architecture, is composed of two clusters connected via a global network. The Cluster_1 class encloses a parametric number of processing units (PU) specified using the Shaped repetition concept. The Tiler connector (whose attributes are not shown in this figure for the simplicity purpose) between the global_net port of the PU and the global_net port of the Cluster_1 specifies how processing units are regularly connected to the global network.

3.5 Partial allocation view of the case study

Data-parallel splitting of the motion compensation process while leaving parts of the application sequential requires a constrained allocation view that will guide the rapid prototyping process. In Figure 7, the main component of the H.264 decoder and the main component of the hardware architecture are displayed. Since parallel processing is not needed for the EnDec_IQT and DF tasks, they are completely mapped on the processing unit of Cluster_0 via the Allocate links. The data-parallel splitting of the MC_Row task imposes the distribution of the repetitions of this task onto the processing units of Cluster_1 using the Distribute stereotype.

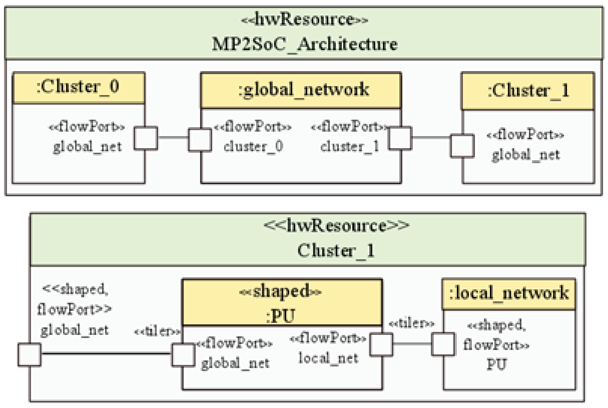


Figure 6: MP2SoC UML models

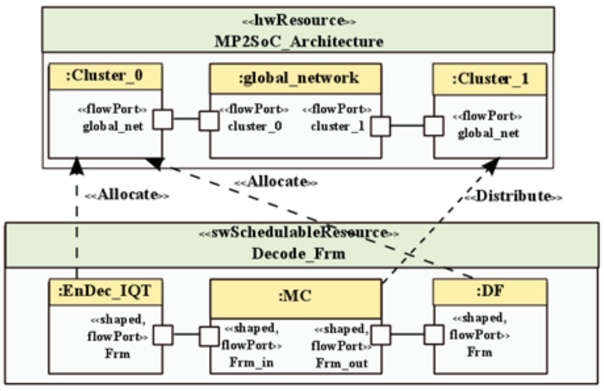


Figure 7: Allocation for the motion estimation parallelization

3.6 Deployment modeling

Figure 8 presents the deployment of the PE elementary component onto the PE_IP artifact stereotyped hwIP. This class is deployed on the Nios II processor. The Nios

Table 2: Multiplicities inside the parallel motion

Class	EnDec _IQT	MC	MC	DF	MC_Row	MC_Row
Port	Frm	Frm_in	Frm_out	Frm	row_in	row_out
Multiplicity	X × Y	X × Y	X × Y	X × Y	X × (Y ÷ nbrow)	X × (Y ÷ nbrow)

II processor IP is provided with the hardware library, associated with our framework, which contains processor, memory and communication network IPs. While the filePath attribute facilitates the generation of the MP2SoC source code, the vlnv attribute guides the S-LAM generation process as it gathers required IP properties.

Our framework integrates a source code generator that produces the implementation of a given MP2SoC architecture. Currently, Nios II-based systems can be directly generated from a parameterized specification of the architecture in terms of processor numbers.

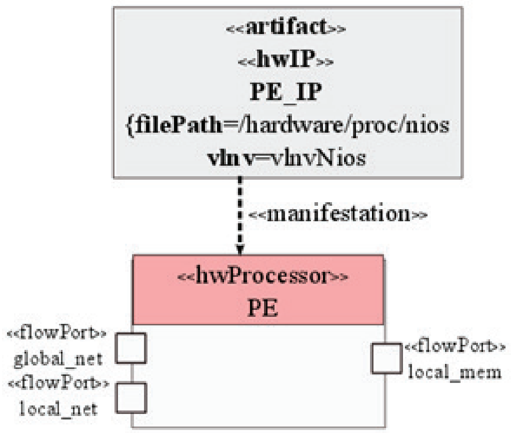


Figure 8: Deployment of the PE

3.7 Executing transformation chains within the case study

3.7.1 Generation of πSDF files

Executing the πSDF transformation chain on the parallel motion compensation application generates one πSDF file for each hierarchic class: Decode_Frm.pi (Figure 9(a)) and MC.pi (Figure 9(b)) files. The hierarchic structure of the application is conserved during the transformation phase for the two applications.

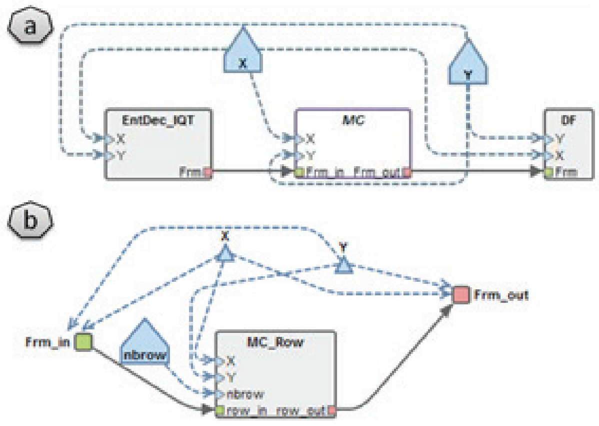


Figure 9: .pi files of the parallel motion compensation application

3.7.2 Generation of S-LAM files

Different configurations of MP2SoC were generated varying the number of processing units (by changing the shape value of the PU class). For the rapid prototyping of the parallel motion compensation application, four architectures were produced containing 2, 4, 8 and 16 processing units in the Cluster1. Generating a complete MP2SoC architecture containing 8 processing units is illustrated in Figure 10.

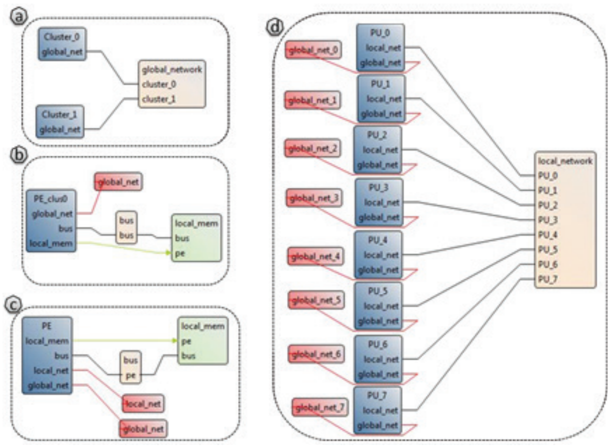


Figure 10: S-LAM files of the MP2SoC architecture

3.8 Exploration results in the case study

In the parallel motion compensation approach, the motion compensation task for each row of one P-frame is executed in parallel on different cores. We experiment this parallelization method using the CIF (352 × 288) resolution.

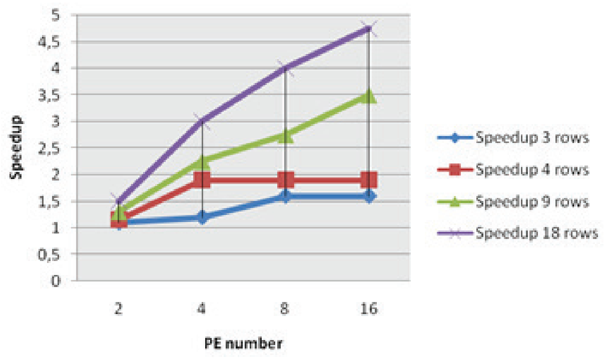


Figure 11: Speedup of the parallel motion compensation parallelization method

In CIF resolution, each frame has 22 horizontal macroblocks and 18 vertical macroblocks. Figure 11 shows the average speedup of the motion compensation task for different number of rows. For the CIF resolution, the maximum speedup of 4.75 is reached using 16 processing units and 18 rows. The speedup decreases as the number of rows decreases for the same processing unit number. In fact, decreasing the row number leads to increasing the number of macroblocks inside each row, the fact that slows down the execution time. In contrary, increasing the row number intensifies the speedup. For example, doubling the row number (from 9 to 18) improves the speedup with 45% in an eight processing units-based architecture. In fact, the scheduler distributes a set of rows of small size on the processing units, once the parallel execution of these rows completes rapidly, it distributes another amount of rows.

The speedup for 18 rows is around 4 when eight processing units are used. Doubling the number of processing units rise the speedup to 4.75 which cannot be considered as efficient as expected since barely a poor improvement of 18% is gained. The main reasons are the extra-time needed for synchronization between processing units and the big amount of data transfer overhead that intensifies the execution time.

4 Conclusion

The starting point of our study is to adapt a methodology for the co-design of complex embedded systems. Previous research works in the co-design domain focus on simulation for system analysis. While some other researches promote elevation of design abstraction levels, they do not benefit from the advantages offered by the MARTE profile and the novel πSDF MoC. The contribution of this paper is the definition of an MDE-based flow that takes as input the UML diagrams specified with the MARTE profile and transforms them into intermediate models corresponding to the πSDF and S-LAM models. These intermediate models add additional semantics and techniques, with the intended goal of analyzing the application, exploring the design space of possible implementations and generating the system implementation.

Component-based approach provides means to decompose a complex system into simpler components. As we have seen in this paper, our framework takes advantage of this approach for the specification of the data-intensive application and the massively parallel architecture. The complex structure of data-intensive applications makes them suitable to compositional or hierarchical design. Compositional design of MP2SoC architectures can be done using hardware components consisting of elementary or composite classes arranged in a hierarchical manner. In Section 3, we described a compositional specification technique that is totally based on the composite structure diagram concepts where a complex application is divided into simpler tasks and a given MP2SoC architecture is specified based on a bottom-up approach that builds the hierarchy of the architecture using elementary components assembling. To benefit from component-based design for MP2SoC systems, the scheduler should be addressed for high performance applications running on clusters.

The static scheduling algorithms implemented within the PREESM scheduler assign tasks to computing resources before applications are executed. At compilation time, task execution time and communication time are supposed to be known and specified as discussed in Section 3. These algorithms, including the list scheduling and the FAST algorithms, are mainly dedicated to scheduling tasks on multi-core systems. Prototyping the H.264 decoder using the scheduling kernel of PREESM brings some limitations including:
Limitations in code generation: the current PREESM code generation supports exclusively static πSDF graphs. A new code generation based on a runtime system name Spider and supporting all πSDF features is currently studied
Lack of energy and area cost estimation: performance is evaluated based on two metrics, throughput and latency. Although the optimization of these constraints is vital when dealing with high-performance applications, power consumption and area occupation remains more important with the ever increasing number of cores inside MP2SoC systems.

Mapping task graph nodes onto clusters means clustering. The task graph clustering approach [22] for scheduling massive parallel tasks on cluster-based architectures seems to be effective for MP2SoC systems. Researches in this field try to combine clustering algorithms with power consumption reduction [23, 24]. These efforts use emerging power reduction techniques, for example, the Dynamic Voltage and Frequency Scaling (DVFS) [25] and try to adapt them for the cluster-based systems. Integrating such technique into the PREESM scheduler seems to be a good direction to support the composition characteristic of the architecture and the application. Another motivating point is that these techniques are based on a DAG description of the application [26], which is the entry point of the PREESM scheduler.

The PREESM scheduler, as seen in Section 2, divides the assignment and cost evaluation tasks into two sub-modules. One advantage of this approach is that additional heuristics, like power and area, can be easily integrated within the cost evaluation kernel. Since the S-LAM model of the application and the scenario file enclosing system constraints are the inputs of the cost evaluation task, additional values needed for the power and area estimation need to be generated from high-level UML models and encapsulated into these files. These values include the processor frequency, hardware resources occupation area, and power constraints, etc. Attributes associated with stereotypes of the MARTE profile will be used to specify these values.

5 References

1. Object Management Group. Unified Modeling Language specification, version 2.1.
2. Object Management Group. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, version 1.0.
3. K. Grüttner, P. A. Hartmann, K. Hylla, S. Rosinger, W. Nebel, F. Herrera, et al, "The COMPLEX reference framework for HW/SW co-design and power management supporting platform-based design-space exploration," MICROPROCESS MICROSY, vol.37, no.8, pp.966-980, 2013.
4. C. Silvano, et al., "Multicube: Multi-objective design space exploration of multi-core architectures," IEEE Computer Society Annual Symposium on VLSI, pp. 488–493, 2010.
5. A. Pimentel, C. Erbas, S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," IEEE Transactions on Computers, vol.55, no.2, pp.99–112, 2006.
6. F. A. M. do Nascimento, M.F.S. Oliveira, F.R. Wagner, "A model-driven engineering framework for embedded systems design," Innovations in Systems and Software Engineering, vol.8, pp.19-33, Mars 2012.
7. B. Schatz, F. Holzl, T. Lundkvist, "Design-space exploration through constraint based model-transformation" 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems, ECBS 2010, pp. 173–182, 2010.
8. L. G. Murillo, M. Mura and M. Prevostini, "MDE Support for HW/SW codesign: A UML-based design flow," In Advances in Design Methods from Modeling Languages for Embedded Systems and SoC's, Springer Netherlands, pp. 19-37, 2010.
9. E. Lee and D. Messerschmitt, "Synchronous data flow," Proceedings of the IEEE, vol. 75, no. 9, p 1235-1245, September 1987.
10. K. Desnos, M. Pelcat, J.F. Nezan, S. Bhattacharyya and S. Aridh, "PiMM: Parameterized and Interfaced Dataflow Meta-Model for MPSoCs Runtime Reconfiguration," International Conference on Embedded Computer Systems: Architecture, Modeling and Simulation (SAMOS XIII), Samos, Greece, July 2013.
11. IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tools Flows, IEEE Std 1685-2009, Feb. 2010, pp. C1-360.
12. M. Pelcat, J. F. Nezan, J. Piat, J. Croizer, and S. Aridhi,"A system-level architecture model for rapid prototyping of heterogeneous multicore embedded systems," In Proceedings of DASIP conference., September 2009.
13. M. Pelcat, K. Desnos, J. Heulot, C. Guy, J.-F. Nezan, and S. Aridhi,"Preesm: A dataflow-based rapid prototyping framework for simplifying multicore DSP programming," In 6 th European Embedded Design in Education and Research Conference, EDERC 2014, pp. 36-40, 2014.
14. M. Ammar, M. Baklouti, M. Pelcat, K. Desnos and M. Abid, "MARTE to πSDF transformation for data-intensive applications analysis," In Conference on Design & Architectures for Signal & Image Processing, DASIP, October 2014.
15. M. Ammar, M. Baklouti, M. Pelcat, K. Desnos and M. Abid, "Automatic Generation of S-LAM Descriptions from UML/MARTE for the DSE of Massively Parallel Embedded Systems, " 2015 Software Engineering Research, Management and Applications conference, SERA 2015 (to be appeared).
16. A.-H. Ghamarian, M. C W Geilen, S. Stuijk, T. Basten, A. J M Moonen, M.J.G. Bekooij, B.D. Theelen, M.R. Mousavi, "Throughput Analysis of Synchronous Data Flow Graphs," Sixth International Conference on Application of Concurrency to System Design, ACSD 2006, pp.25-36, June 2006
17. K. Desnos, M. Pelcat, J.-F. Nezan, S. Aridhi, "Memory bounds for the distributed execution of a hierarchical Synchronous Data-Flow graph," International Conference on Embedded Computer Systems, SAMOS 2012, pp.160-167, July 2012.
18. P. Guduric, A. Puder, R. Todtenhofer, "A Comparison between Relational and Operational QVT Mappings," In the 6th International Conference on Information Technology: New Generations, ITNG '09, pp.266-271, April 2009.
19. J. V. Team, "Advanced video coding for generic audiovisual services," ITU-T Rec. H, vol. 264, pp. 14496–10.
20. E. Baaklini, S. Rethinagiri, H. Sbeity, et al, "Scalable row-based parallel H.264 decoder on embedded multicore processors," In Signal, Image and Video Processing, pp. 1-15, 2014.
21. M. Baklouti, and M. Abid, "Multi-Softcore Architecture on FPGA," International Journal of Reconfigurable Computing, 2014.
22. X. Qin and H. Jiang, "A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters," Journal of Parallel and Distributed Computing, vol. 65, no. 8, pp. 885-900, 2005.

23. G. L. Valentini, W. Lassonde, S. U. Khan, et al, "An overview of energy efficiency techniques in cluster computing systems," Cluster Computing, vol. 16, no. 1, pp. 3-15, 2013.
24. Z. Zong, A. Manzanares, X. Ruan, et al, "EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters," IEEE Transactions on Computers, vol. 60, no. 3, pp. 360-374, 2011.
25. L. Wang, G. Von Laszewski, J. Dayal, et al, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," In10th IEEE/ACM International Conference on: Cluster, Cloud and Grid Computing, CCGrid 2010, pp. 368-377, 2010.
26. L. WANG, S. U. KHAN, D. CHEN, et al,"Energy-aware parallel task scheduling in a cluster," Future Generation Computer Systems, vol. 29, no. 7, pp. 1661-1670, 2013.

Arrived: 25. 04. 2016
Accepted: 13. 09. 2016