A Model Driven Co-Design Approach for High Performance Embedded Systems Dedicated to Transport

Sébastien Le Beux, Philippe Marquet, Jean-Luc Dekeyser

LIFL, INRIA Lille-Nord Europe and University of Lille

France

E-mail: Sebastien.Le-Beux@polymtl.ca

Abstract: This paper demonstrates that the Model Driven Engineering approach is reliable for the development of codesign environments dedicated to embedded systems. From a UML model of a data parallel application, the Gaspard environment is able to generate an hardware accelerator which executes the modeled application. All the potential parallelism of a given application is modeled in UML and is exploited in order to generate a powerful hardware system. Gaspard is dedicated to intensive signal processing applications. Therefore, many automotive sector applications could benefit of the Gaspard environment.

Keywords: Model Driven Engineering, embedded systems, data parallelism, FPGA, intensive signal processing.

Sébastien Le Beux is currently post-doctoral researcher at the École Polytechnique of Montreal, Canada. He received a PhD. in computer science from the University of Lille in 2007. His research interests include the design of parallel, embedded and reconfigurable architectures, model driven engineering and the mapping of intensive signal processing applications onto network on chip.

Philippe Marquet is currently an assistant professor at the University of Lille, France and a researcher within the INRIA, the French institute for research in computer science. Philippe MARQUET received a Ph.D. in Computer Science from the University of Lille in 1992. His research interests include the design of parallel, embedded and reconfigurable architectures, the definition of programming models, languages and compilers dedicated to parallel computing. He also worked on the definition and implementation of real-time operating systems for SMP architectures. He (co-)advised 10 Ph.D thesis.

Jean-Luc Dekeyser received his PhD degree in computer science from the University of Lille in 1986, he was a fellowship at CERN Geneva. After a few years at the Supercomputing Computation Research Institute in Florida State University, where he worked on high performance computing for Monté-Carlo methods in High Energy Physics, he joined in 1988 the University of Lille in France as an assistant professor. There he worked on data parallel paradigm and vector processing. He created a research group working on High Performance Computing in the CNRS lab in Lille. He is currently Professor in computer science at University of Lille. He is heading the DaRT INRIA project. His research interests include embedded systems, System on Chip co-design, high performance computing, model driven engineering.

1. Introduction

Complexity of the automotive embedded applications continuously increases. Anti-collision radar, vehicle tracking and inter-vehicle communication are some examples of such applications which have to be embedded in a system.

Most of the time, embedded systems have real-time constraints and have to satisfy various constraints, such as energy consumption, surface, etc. The “System on Chip” (SoC) allow to design mix hardware and software embedded systems. SoC are realized on a single chip, therefore, they help to solve embedded systems constraints. The current SoC generation already contains several processors on a single chip: a general purpose processor (ARM, etc.), DSP dedicated to digital signal processing applications, ASIC for very specific and systematic tasks and reconfigurable components, FPGA, which add flexibility to a SoC. The rising integration capabilities of the chips allows to design more “parallel” SoC, where several homogeneous or heterogeneous processors communicate according to bus or complex Network on Chip (NoC). The next SoC generation has to provide more and more computing power and still has to satisfy the energy consumption constraints.

The rising complexity of the automotive application and the rising SoC integration capabilities are the trend. However, it is an ambitious challenge to efficiently execute such complex

applications onto such complex architectures. In order to successfully face this challenge, it becomes necessary to relevantly exploit the specificity of hardware and software elements contained in embedded systems. A hardware element corresponds to an architectural component in a SoC (*i.e.* processors, memories, etc.), a software elements corresponds to a task in an application. Some approach aim to specify the software independently from the hardware in order to facilitate the design of embedded system. The software and the hardware are tested together only at the end of the design process, they can reveal some major problem that could have been detected earlier with a co-design approach. The co-design approach aims to specify, at the same time and in the same environment, the hardware and the software embedded in a system. For this purpose, it is necessary to model both the hardware and the software in the unified environment in order to accelerate the design process and to increase the productivity of the SoC designers.

This paper extends the works presented in [6] and demonstrates the reliability of the Gaspard codesign environment to develop high performance systems. Gaspard is introduced in Section 2; this environment is definitely oriented towards regular parallel applications and architectures. Section 3 presents the high level modeling of a purely data parallel application used in an embedded automotive radar. From this high level model, the VHDL code is automatically produced and synthesized onto FPGA. On-road experimentation results of the overall embedded system are provided in Section 4, demonstrating the reliability of our approach. The conclusion presents the extension of our work for control-flow and mix data-flow/control-flow applications.

2. Co-Design Approach

In classical approach, the software and hardware designs are realized separately and are tested together only at the end of the design process. The conjoint development process of the software and the hardware, called co-design, allows handling complex systems design and promotes their perennity in order to enhance their productivity. The main objective is to manage the growing complexity of the embedded systems thanks to a co-design approach. Indeed, the design space exploration, the compilation processes and the prototyping are fundamental challenges in co-design. The design space exploration processes allows finding hardware architecture convenient for a specific application. The performance analysis provides relevant information on the embedded system behavior, which is provided to design space exploration process [1]. The prototyping validates this performance analysis.

This new approach also constitutes a challenge for the development of the software tools dedicated to the co-design. Indeed, the co-design of a SoC manipulates the two technologies (software and hardware) and covers the different process cycles, from the specification towards the realization.

2.1 Model driven engineering

The Object Management Group (OMG: http://www.omg.org/) advocates the use of the Model Driven Engineering [14] (MDE) in order to clearly separate the specification level from the implementation level. The use of the MDE for SoC co-design offers a new opportunity to design embedded systems. MDE allows separating the application and the architecture specifications from the languages (platforms) used to realize or execute the embedded system. This allows to reuse models and components at high abstraction levels.

In MDE, abstraction levels are specified with metamodels. A metamodel gathers the set of concepts and relations between the concepts used to describe a model at a given abstraction level. A model is then said to *be conform to* a metamodel. Generally speaking, a metamodel defines the syntax of its models, like a language defines its grammar. In MDE, a model transformation [5] is a compilation process which transforms a *source* model into a *target* model. The source and the target models being respectively conform to the source and the target metamodels.

A model transformation relies on a set of *rules*. Each rule clearly identifies concepts in the source metamodel in order to create concepts respecting the target metamodel. Such a decomposition facilitates the extension and the maintainability of a compilation process: new rules extend the compilation process and each rule can be modified independently from the other rules. Model transformations allow to transform a source model at a given abstraction level into a destination model at a lower abstraction level. Therefore, they make possible the deployment of the system according to the targeted implementation (SystemC, VHDL, etc.).

2.2 The Gaspard environnement

Gaspard is entirely developed according to the MDE approach: the application, the architecture and the association of the application onto the architecture are completely independent from any implementation details. For instance, a same high level specification of an application can be simulated with SystemC, synthesized using VHDL, executed onto multiprocessors architecture using Open/Fortan and verify thanks to the Synchronous language.

For this purpose, Gaspard follows the "Y-chart" [9], as illustrated in Figure 1. MDE helps to specify the abstraction levels (which are associated to a metamodel) and the model transformations [5] between these abstraction levels. In this paper, we focus on the chain that generates VHDL (this allows FPGA synthesis). The transformation chain is the following: UML→Deployed→RTL→VHDL. UML corresponds to the high level specification of systems, Deployed corresponds to the first internal representation, RTL corresponds to the description of an hardware accelerator able to execute a given application and VHDL corresponds to the VHDL code of this hardware accelerator. Deployed, RTL and VHDL are automatically generated thanks to model transformations (represented with arrows). The overall metamodels and model transformations allow the opportunity to express/manipulate regular aspects.

Indeed, Gaspard is definitely oriented toward the data parallel paradigm and its usage in embedded systems for intensive signal processing applications. A specific attention is provided to the regular aspects in applications and multiprocessors system on chip (MPSoC). The placement of an application (software) onto an architecture (hardware) also requires regular aspects that are exploited during the compilation process.

3. Co-Design for Transport

The automotive domain also requires tools and techniques in order to realize SoC based embedded systems. These tools have to contribute in the maintenance and have to warranty a high level safety of such systems despite their growing complexity.

3.1 The ModEasy project

The Interreg ModEasy project2 (which stands for MOdel Driven dEsign for Automotive Safety embedded sYstems, http://www.lifl.fr/modeasy/) aims to develop software tools and techniques to aid the design of reliable embedded systems using advanced software technologies. The tools are evaluated on applications of the automotive domain, such as reactive cruise control [11] and anti-collision radar [13, 6] but can be applicable for generic embedded systems in any safety and mission critical applications in the wider industrial domain. The project intends to reduce development and production costs while maintaining existing high dependability and safety levels as embedded systems become more complex for many existing and new products.

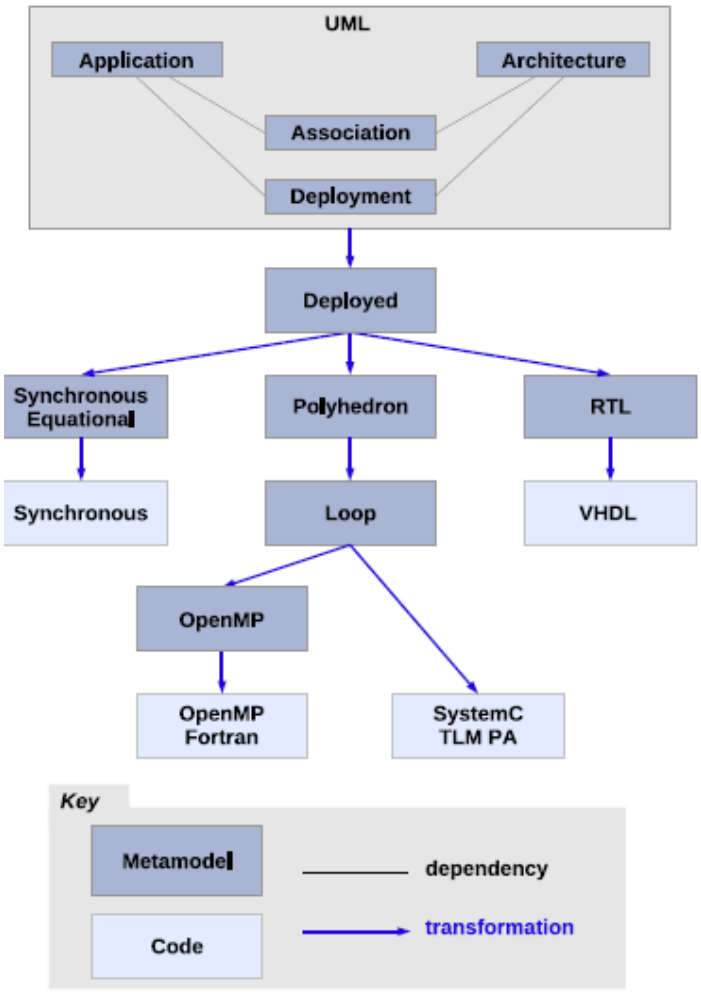


Figure 1. The Gaspard environment: from a high level specification, model transformations allow to generate several targets implementations.

The aim of the project is to develop tools, which are to generate low level implementations (*co-design synthesis*) from high level specifications of automotive embedded systems (*co-design specification*). The detailed goals of the ModEasy project are to:

- evaluate the specific requirements of the automotive embedded systems;
- formalize the high level specification of these requirements into metamodels thanks to the MDE methodology. Such metamodels are implemented in UML with profiles [4];
- specify the low abstraction level metamodels which correspond to the targeted implementations: SystemC for a multiprocessor SoC simulation and VHDL for a hardware accelerator execution;
- develop transformation rules in order to ensure the automatic generation of the low level implementations from the high level specification;
- allow FPGA implementations for the generated hardware accelerators;
- evaluate the developed tools for cruise control and anti-collision radar systems;
- prototype these systems onto FPGA and realize on-road experimentation.

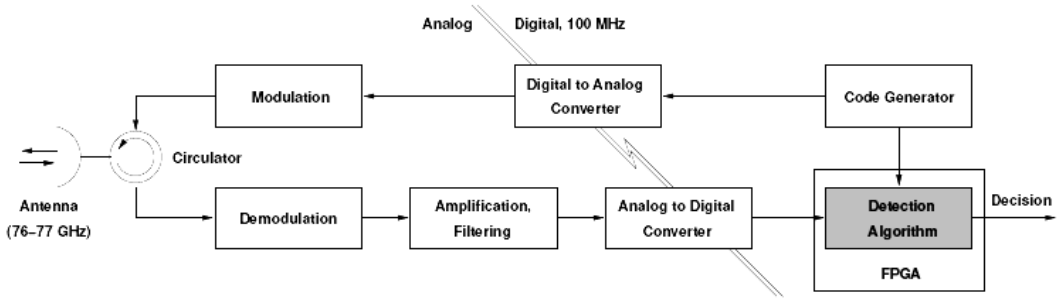


Figure 2. The studied detection algorithm in an anti-collision radar system.

The following section deals with some results of the project, which concern a detection algorithm embedded in an anti-collision radar system. For this purpose, we have:

- model in UML the detection algorithm;
- generate an hardware accelerator from the UML model;
- synthesize the hardware accelerator onto FPGA;
- integrate the FPGA in the overall anti-collision radar system;
- evaluate the performance of the embedded system during on-road experimentation.

3.2 Anti-collision radar system

Anti-collision radar systems use radar waves to detect obstacles. Such system emits a wave thanks to an antenna, as illustrated in Figure 2. When the emitted wave hits an obstacle (other vehicles, animals, etc.), it is re-emitted in the direction of the antenna. The system compares the received wave with the emitted one, searching for similarities resulting from the presence of an obstacle in front of the vehicle. Computing the time spent between the emission and the reception of the wave, the system can determine the distance between the car and the obstacle. By periodically computing this distance, the system can also estimate the relative speed of the car and the obstacle. Thus, detecting an obstacle comes down to one task: performing the correlation of an emitted and a received wave. This obstacle detection is placed in an embedded system context, where resources are limited. This leads us to use FPGA technologies, which offer flexible solutions for the implementation of hardware accelerator that perform the correlation.

The correlation is the most resources and time consuming part of the overall anti-collision radar system: it receives a sample each 10 ns and executes thousands operations for each received sample. The computing power required for the execution of such data parallel applications is considerable but they can be efficiently executed by hardware accelerators [15, 10]. However, the development of full custom execution units is costly and error prone since it requires hardware designer experts, even when using the usual high level synthesis tools. We believe that Gaspard is able to facilitate the development of such hardware accelerators for the studied automotive system. Indeed, it can be generated from a high level specification (*i.e.* independent from the implementation details of the hardware accelerators) of the algorithm used to detect the obstacles. The studied detection algorithm is expressed by the following equation:

$$C_{cy}(j) = \sum_{i=0}^{1023} c(i) \cdot y(i+j)$$

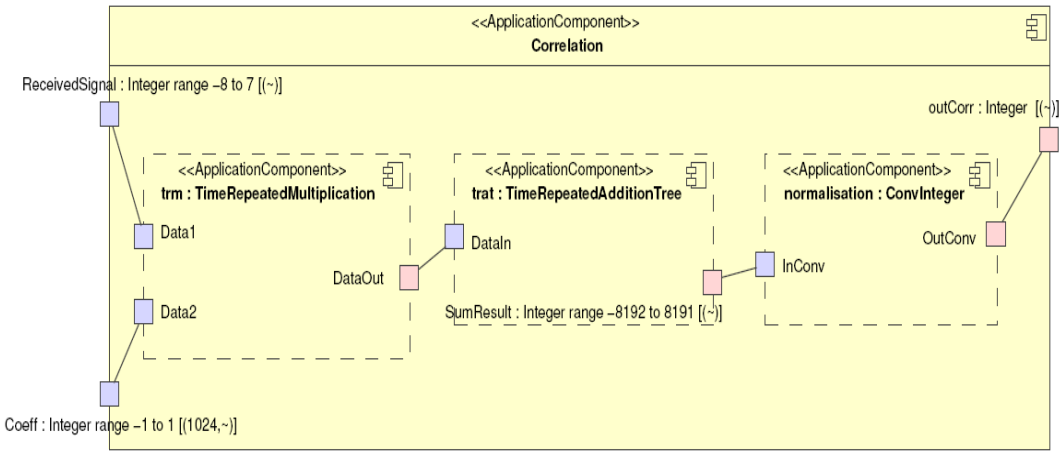


Figure 3. Top view of the detection algorithm: each component instance represents a task and each connector expresses a data dependency.

Ccy(j) corresponds to the correlation results, c(i) is the reference code and y(i+j) corresponds to the received signal. This algorithm and its hardware implementation have already been studied [15], but our aim is to generate these hardware accelerators from high level models. The following presents the UML model of this detection algorithm.

3.3 UML modeling of the detection algorithm

We describe the UML model of the detection algorithm with a top-down approach.

3.3.1 Overall view of the detection algorithm

For each received sample, the detection algorithm computes 1024 multiplications and realizes an addition with the results provided by the overall multiplications. These two majors steps are modeled in Figure 3: *trm* realizes the multiplications and *trat* performs the summation. The task *ConvInteger* standardizes the output of the detection algorithm in order to produce integer values. The connectors represent the data dependencies between the several tasks. The detection algorithm has two input ports: *ReceivedSignal* represents the received signal (coming from the antenna in the embedded system) and *Coeff* corresponds to the reference code. *outCorr* is the output of the detection algorithm (*i.e.* the results).

3.3.2 Repetition of the multiplication

Figure 4 represents the UML model of the multiplications: the *TimeRepeatedMultiplication* component expresses the temporal data dependencies while *RepeatedMultiplication* expresses the spacial data dependencies. The data flowing through *inDataM* (from *rm*) correspond to the shift of the data flowing through *Data1*. This shift is expressed thanks to the tiler that connect *Data1* to *inDataM*. The tilers are precisely defined in [3], they allow to express complex data dependencies in data parallel applications.

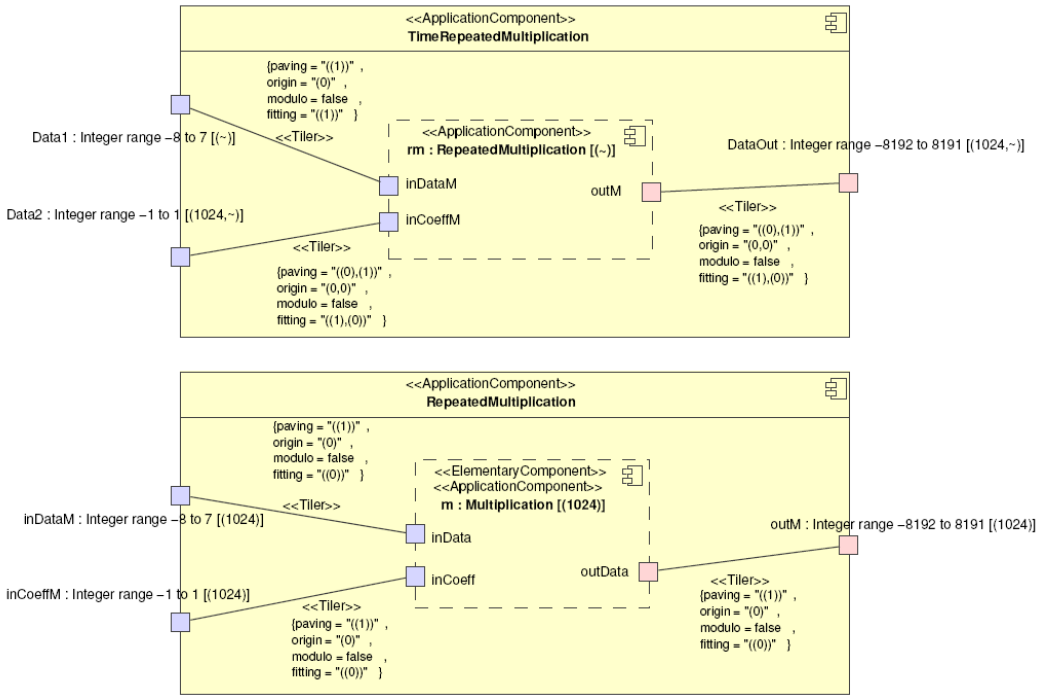


Figure 4. UML model of the multiplications: the top side of the figure expresses the temporal data dependencies, the bottom side expresses the spacial ones.

A tiler allows to model data dependencies that links a M-dimension data array to a N-dimension data pattern. These data dependencies are not limited to compact and parallel to axis patterns. Usually, data dependencies are expressed within indexes. Indexes are tedious to manipulate and error prone when directly provided by users: their complexity dramatically increases with the number of dimensions and the shape of the pattern. Tilers do not cause this inconvenience. The spacial data dependencies, illustrated on the bottom side of Figure 4 are expressed in the same manner. The repetition space of the task *m* is expressed thanks to the multiplicity [(1024)].

3.3.3 Pipelined additions

Figure 5 represents the UML model, which expresses an addition of 1024 data (previously computed by the multiplications). *InAdditionTree* is the input port of the addition tree, its shape is [(1024)]. *Result* is the output port (the result of the addition), it is a scalar.

Each pipeline stage of the addition tree realizes a partial addition of the overall addition. The shape of the ports between the pipeline stages reduces until a scalar is obtained from the initial 1024 input data (1024 → 512 … 2 → 1). The bottom side of Figure 5 represents the first pipeline stage. This component partially adds 1024 data into 512 data. In order to realize the partial addition, the task *a1* (an elementary component) is repeated 512 times. This task is elementary and is linked to an IP (Intellectual Properties). An IP provides the behavior of an elementary component. The links between an elementary component and an IP are managed during the deployment specification step (Figure 1) like detailed in the following.

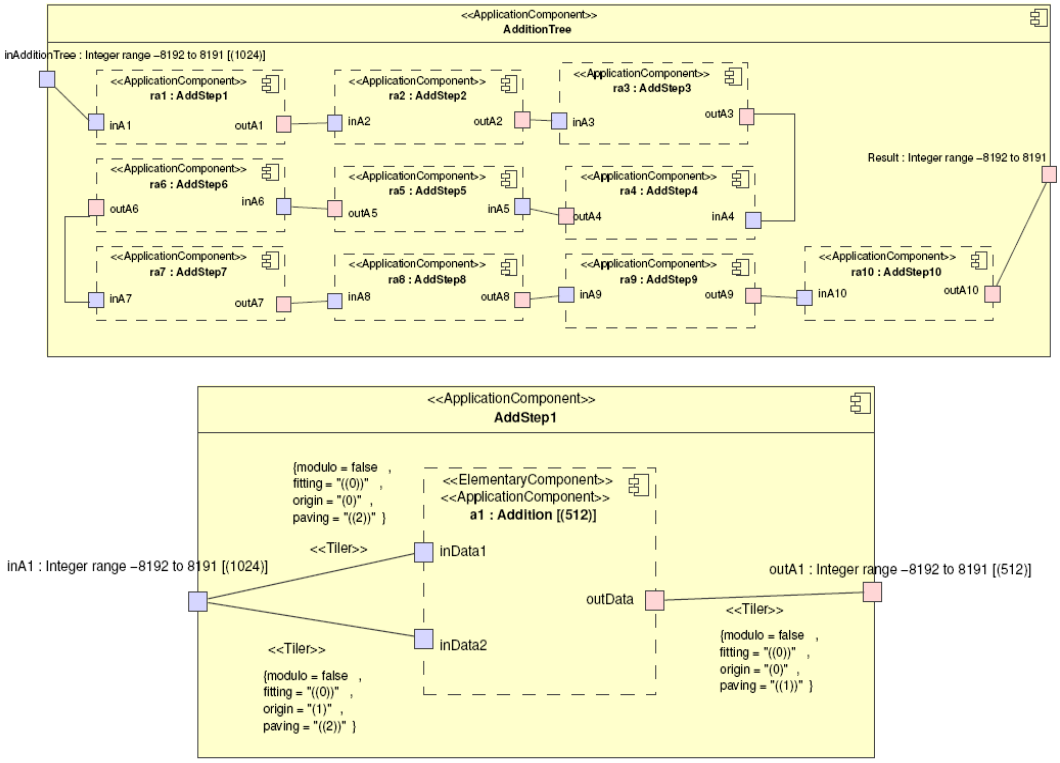


Figure 5. UML model of the addition: the top side illustrates the addition tree while the bottom side illustrates a data parallel task in this addition tree.

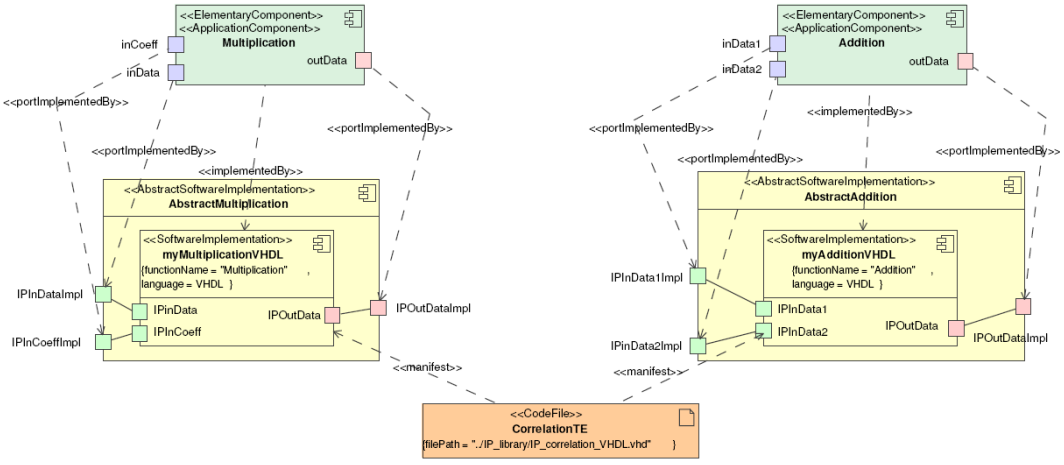


Figure 6. The elementary components are deployed onto IP contained in a library.

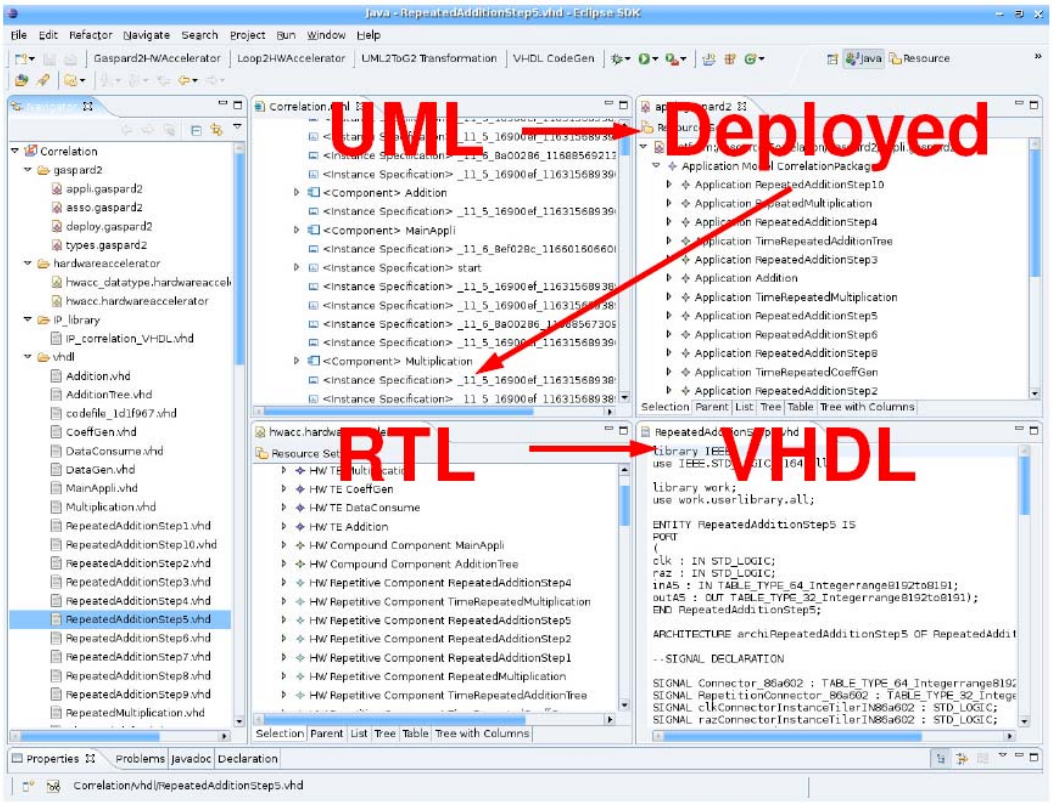


Figure 7. Automatic generation of a VHDL code from a UML model. The resulting VHDL code describes an hardware accelerator able to execute the modeled detection algorithm in UML.

3.3.4 Deployment of the elementary components

The elementary components from the previously detailed model (an addition and a multiplication) correspond to the leaf of a tree (*i.e.* the tree represents the overall algorithm). The behavior of leafs is provided during the deployment specification step [2].

Figure 6 illustrated the deployment of the addition and the multiplication components. The top side of the figure represents these elementary components, the center of the figure shows the IP and the bottom side of the figure illustrates the code-file of the IP. Elementary components and their ports are deployed thanks to the (ImplementedBy and PortImplementedBy connectors. The file-path of the IP code is provided thanks to a manifest connector. In this example, the same code file contains the multiplication and the addition IP.

4. From UML to Road…

Gaspard is able to transform a UML model in order to generate a hardware accelerator. Figure 7 illustrates the integration of Gaspard into the Eclipse environment [7]. The UML model corresponds to the detection algorithm previously detailed. This model is transformed in a Deployed model which corresponds to the first internal representation of the application. A RTL is then automatically generated, it corresponds to an abstract representation of the hardware accelerator. Details about the transformation from a deployed model into a RTL model can be found in [12]. The last step is the VHDL code generation, the resulting VHDL code describes the hardware accelerator and it is directly simulable and synthetically according to commercial

synthesis tools. The following section provides simulation and synthesis results provided within such commercial tools.

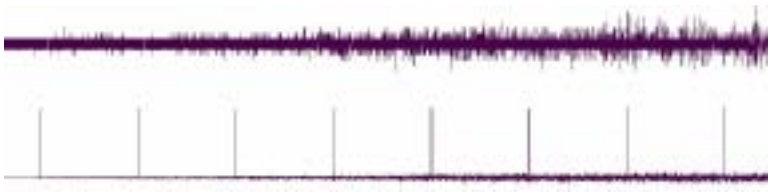


Figure 8. Simulation results of the hardware accelerator.

4.1 VHDL simulation

Figure 8 illustrates the correct behavior of the generated hardware accelerator. The top side of the figure illustrates the input of the hardware accelerator (the received wave) and the bottom side of the figure represents the output (*i.e.* the result of the detection algorithm). The peaks demonstrate the right behavior of the hardware accelerator: the hardware accelerator is able to detect the obstacles in front of the anti-collision radar system.

4.2 VHDL synthesis

This section presents the synthesis results of the generated VHDL code. This section is illustrated with code excerpts and different views of the synthesized design.

4.2.1 Data dependencies

Expressing the overall application in UML, Gaspard identifies that the temporal data dependencies of the detection algorithm are efficiently implemented in hardware with shift register. The top left-hand side of Figure 9 presents an excerpt of the corresponding VHDL code. The **dInTIN1db 1** signal allows to create a shift register able to shift data 1024 times. In the **process**, the **loop** statement corresponds to the description of the shift register itself, its input is the signal **InTIN1db(1)**. The outputs are connected to the different pipeline stage of the shift register. The bottom left-hand side of Figure 9 represents a partial synthesis result of the shift register. The left-hand side represents the input of the shift register and the right-hand side represents the outputs that are connected to shifted data.

The spacial data dependencies are managed in the same manner. However, while shift register are necessary for temporal data dependencies, simple connectors can perform spatial data dependencies. The excerpt of the VHDL code illustrated on the top right-hand side of Figure 9 describes such data dependencies. This code can be directly synthesized onto FPGA, as illustrated on the bottom right hand side.

Starting from a UML description of the tilers, Gaspard automatically generates such VHDL code. For this purpose, we manage all the specificities of the tiler (which are detailed in Section 3.3.2).

4.2.2 Addition tree

The top side of Figure 10 illustrates the 6 last stages of the addition tree in the generated hardware accelerator. They partially realize the addition in the detection algorithm. A tree topology and the reduction of data arrays from a pipeline stage to another are clearly identified. The bottom left-hand side of Figure 10 illustrates a VHDL code excerpt of the 8th pipeline stage (*i.e.* in the addition tree). This code corresponds to a multiple instantiation of the **Addition** computing unit; this allows to execute four additions in parallel. The bottom right-hand side of

Figure 10 represents the synthesis results for this 8th pipeline stage (which include the previously detailed VHDL code excerpt). The marks **1** and **5** correspond to the input and output ports, the marks **2x** and **4** identify the components that resolve data dependencies. The marks **3x** represent the four elementary tasks, which realize additions in a parallel manner.

```
...
process (clk,InTIN1db)
begin
   if clk'event and clk = '1' then
      delayedInputTIN1db_1(1024) <= InputTIN1db(1);
      for i in 1 to 1023 loop
         dInputTIN1db_1(i) <= dInputTIN1db_1(i+1);
      end loop;
   end if;
end process;

Signal_InTIN1db_1 <= dInputTIN1db_1;
...
```

```
...
OutputTIN1160709(6) <= InputTIN1160709(11);
OutputTIN1160709(7) <= InputTIN1160709(13);
OutputTIN1160709(8) <= InputTIN1160709(15);
OutputTIN1160709(9) <= InputTIN1160709(17);
OutputTIN1160709(10) <= InputTIN1160709(19);
OutputTIN1160709(11) <= InputTIN1160709(21);
OutputTIN1160709(12) <= InputTIN1160709(23);
OutputTIN1160709(13) <= InputTIN1160709(25);
OutputTIN1160709(14) <= InputTIN1160709(27);
OutputTIN1160709(15) <= InputTIN1160709(29);
...
```

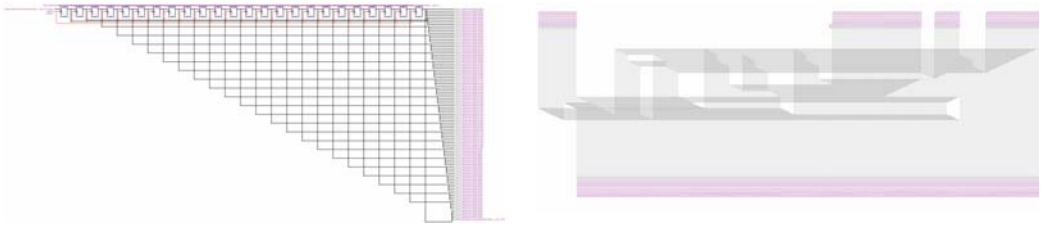
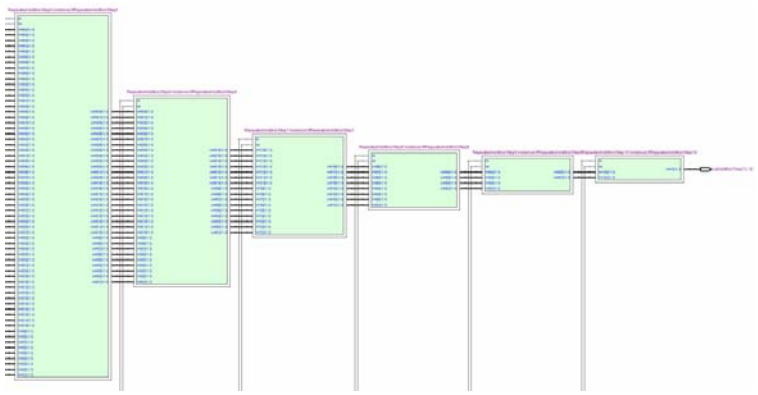


Figure 9. Synthesis results of data dependencies.



```
...
genit0 : for it0 in 1 to 4 generate
   instanceOfAddition : Addition
      port map(
      clk => clkConnectora8,
      raz => razConnectora8,
      outData =>RepConnector_f72499(it0) ,
      inData1
=>RepConnector_17a95c6(it0) ,
      inData2 =>RepConnector_ff22ed(it0)) ;
end generate;
...
```

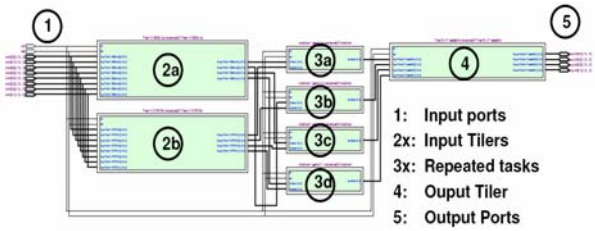


Figure 10. Synthesis results for the pipelined addition tree.

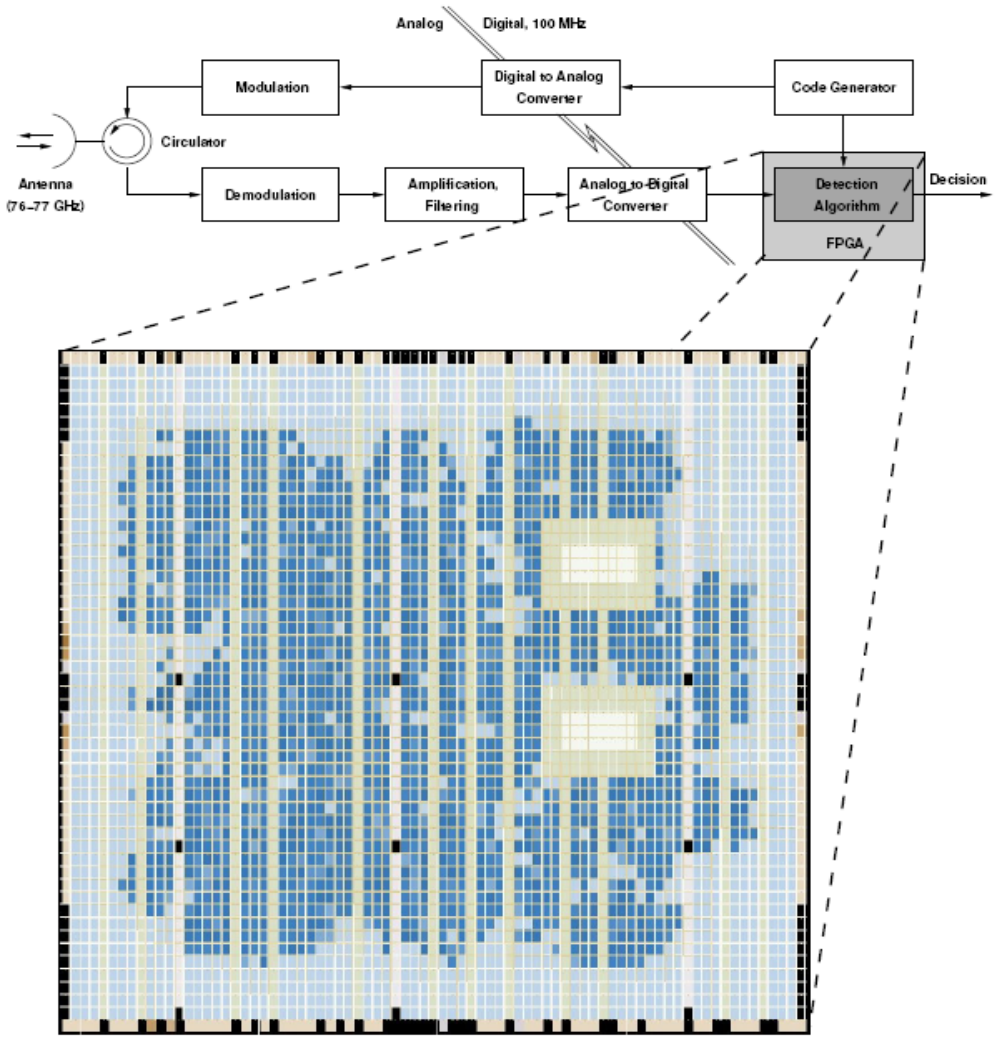


Figure 11. FPGA implementation of the hardware accelerator and integration of the FPGA in the embedded system.

4.2.3 FPGA implementation and experimentation

The generated hardware accelerator has been synthesized onto a FPGA Stratix2S60 from the Altera company. This FPGA has been integrated into the overall anti-collision radar system, as illustrated in Figure 11.

In the scope of the ModEasy project, the embedded system has been tested during on-road experimentation. The tests have been realized with the INRETS of Lille (Institut National de Recherche sur les Transports et leur Sécurité: http://www.inrets.fr/infos/centres/ct_vascq.html) and the IEMN of Valenciennes (Institut d'Electronique de Microélectronique et de Nanotechnologie : http://www.iemn.univ-lille1.fr). The tests have validated the right behavior of the overall embedded system and the hardware accelerator (generated from the UML model) ensures the obstacle detection. Figure 12 illustrates the on-road experimentation. Others tests was also performed by the IEMN in order to use the anti-collision radar with a cruise control and a GPS, resulting in a mix dataflow and control-flow embedded systems [8]. While the data-flow part of the system was automatically generated, the control-flow part was written by hand.

Some extensions of the Gaspard environment are currently realized in order to generate such mix data-flow and control-flow embedded systems from UML models.

Figure 12. On road experimentation of the overall embedded system.

5. Conclusion

This paper presents the Gaspard co-design environment, which is dedicated to the embedded systems. From a UML model of an application, Gaspard generates the VHDL code corresponding to a hardware accelerator able to execute the modeled application. The VHDL code can be synthesized and implemented onto FPGA according to synthesis tools. The UML models are independent from any implementation details. Therefore, one can model an application and generate the corresponding hardware accelerator without being expert in the design of hardware accelerators.

We demonstrate the reliability of the Gaspard environment for data parallel automotive applications. For this purpose, a detection algorithm embedded in anti-collision radar was modeled in UML. The hardware accelerator generated by Gaspard was synthesized onto FPGA. After simulation processes and validations, the hardware accelerator was experimented on-road. This experimentation validated the right behavior of the hardware accelerator and the overall embedded system.

6. Acknowledgements

We acknowledge the LT'2007 staff for their recommendation in the submission of this extended variant of our works.

REFERENCES

1. ATITALLAH, R. B., S. NIAR, J.-L. DEKEYSER, **MPSoC Power Estimation Framework at Transaction LevelModeling**. In The 19th International Conference on Microelectronics (ICM 2007), Cairo, Egypt, Dec. 2007.
2. ATITALLAH, R. B., E. PIEL, S. NIAR, P. MARQUET, J.-L. DEKEYSER, Multilevel MPSoC simulation using an MDE approach. In: IEEE International SoC Conference (SoCC 2007), Hsinchu, Taiwan, Sept. 2007.

3. BOULET, P., A**rray-OL revisited, multidimensional intensive signal processing specification**, Research Report RR-6113, INRIA, Feb. 2007.

4. CUCCURU, A., J.-L. DEKEYSER, P. MARQUET, P. BOULET, T**owards UML 2 extensions for compact modeling of regular complex topologies**. In: MoDELS/UML 2005, ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica, Oct. 2005.

5. CZARNECKI, K., S. HELSEN, **Classification of model tranformation approaches**. In: Proceeding of OOPSLA Workshop on Generative Techniques in the Context of Model Driven Architecture, 2003.

6. DEKEYSER, J.-L., S. LE BEUX, P. MARQUET, **Une approche mod`ele pour la conception conjointe de systèmes embarquées hautes performances dédiés au transport**. In: Workshop International: Logistique & Transport (LT'2007), Sousse, Tunisie, Nov. 2007.

7. **eclipse.org.** Eclipse. http://www.eclipse.org, 2005.

8. HILLALI, Y. E., A. RIVENQ, O. LABBANI, J. ROUVAEN, **Study of a Intelligent Cruise Controlwith GPS and Radar**. In: International ModEasy'07 Workshop, Barcelona, Spain, Sept. 2007. http://www2.lifl.fr/modeasy/workshop.html.

9. GAJSKI, D. D., R. KUHN, **Guest editor introduction: New VLSI-tools**, IEEE Computer, 16(12):11–14, Dec.1983.

10. HILLALI, Y. E., **Etude et réalisation d'un système de communication et de localisation, basé sur les techniques d'étalement de spectre aux transports guidés**, PhD thesis, University of Valenciennes, 2005. (In French).

11. LABBANI, O., E. RUTTEN, J.-L. DEKEYSER, S**afe design methodology for an intelligent cruise control system with GPS**. In: *6*4th IEEE Vehicular Technology Conference (VTC 2006), Montréal, Québec, Canada, Sept. 2006.

12. LE BEUX, S., **Un flot de conception pour applications de traitement du signal systématique implémentées sur FPGA à base d'Ingénierie Dirigée par les Modèles**, Thèse de doctorat (PhD Thesis), Laboratoire d'informatique fondamentale de Lille, Université des sciences et technologies de Lille, France, Dec. 2007.

13. LE BEUX, S., P. MARQUET, O. LABBANI, J.-L. DEKEYSER, **FPGA implementation of embedded cruise control and anti-collision radar**. In: DSD'2006, 9th Euromicro conference on digital system design, Dubrovnik, Croatia, Aug. 2006.

14. SEIDEWITZ, E., **What models mean**, IEEE Softw., 20(5):26–32, 2003.

15. TESSIER, R., W. BURLESON, R**econfigurable computing for digital signal processing: A survey**, The Journal of VLSI Signal Processing, 28(1-2):7–27, Dec. 1999.