On Generating VHDL Descriptions from Aspect-Oriented UML/MARTE Models

Marco Aurélio Wehrmeister
Graduate Program in Applied Computing (PPGCA)
Federal University of Technology - Paraná (UTFPR)
Av. Sete de Setembro, 3165 — 80230-901 Curitiba, Brazil
wehrmeister@utfpr.edu.br

Marcela Leite
Graduate Program in Applied Computing (PPGCA)
Santa Catarina State University (UDESC)
Instituto Federal Catarinense (IFC Araquari)
P.O Box 21 – 89245-000 Araquari, Brazil
marcela.leite@ifc-araquari.edu.br

***Abstract*—This paper discusses an approach to generate VHDL descriptions from high-level specifications, specifically UML/MARTE models that include aspect-oriented semantics. Standard UML diagrams describe the handling of functional requirements, whereas crosscutting concerns associated with the non-functional requirements are handle by aspects. UML-to-VHDL transformation is performed automatically by a script-based code generation tool named GenERTiCA. For that, mapping rules scripts define how to generate VHDL constructs from model elements, including the implementation of aspects adaptations. The generated VHDL description does not require any manual modification, in order to be fully synthesized onto a FPGA device. Some case studies have been performed to evaluate the proposed approach, however, this paper discusses the line-following robot implemented as a FPGA-based embedded system. An improvement in system design has been obtained, namely an increase in system performance and a better utilization of FPGA reconfigurable resources. Such positive results are related to a better modularization of components achieved by using the proposed high-level approach.**

I. Introduction

Nowadays, an increasing number of embedded systems are being delivered to the final customers with *Field-Programmable Gate Array* (FPGA) devices interconnected with traditional IC components (e.g. processors and memory) into the system hardware platform. Programmability, flexibility and performance are some of the reasons to add a FPGA device to perform some tasks in the final embedded systems [1][2].

This hardware/software co-design and partitioning of system task, as well as the increasing number of services demanded from modern embedded systems, increases design complexity of FPGA-based embedded systems. New approaches are demanded in order to manage such a complexity. An old but still valid idea is the increase of the abstraction level used during design. Both academia and industry are seeking for new approaches that use high-level specifications such as, for instance, *Unified Modeling Language* (UML) models annotated with stereotype from its profile for *Modeling and Analysis of Real-Time and Embedded Systems* (MARTE). *Model-Driven Engineering* (MDE) [3][4][5] make intensive use of models, in order to manage complexity and also other project constraints, since engineers need lesser focus on technological details.

Since late 90's, one can find some approaches, e.g. [6], that are able to generate VHDL descriptions from UML models. In [7] and [8], an UML model (specially its class diagram) is used as source of information to generate a VHDL description containing only the system structure, i.e. its components and their connections. System behavior is commonly generated from state digram as in [9]. The main drawback of such approaches is that it is common to find VHDL statements within the UML model, decreasing its abstraction level. Furthermore, state diagrams are not usually used by software engineers, and hence, co-design of embedded system components may be hindered, increasing design time.

Aspect-oriented Model-Driven Engineering for Real-Time systems (AMoDE-RT) [5] is a MDE approach that proposes the intensive use of UML/MARTE models, from which system implementation can be automatically generated, including software/hardware implementation of embedded system components. In [10], AMoDE-RT has been extended to support the generation of VHDL descriptions, including both system structure and behavior. System behavior is specified in a platform-independent fashion via sequence diagrams rather than state diagrams, since the former is easily understood by both hardware and software engineering teams. However, the approach proposed in [10] has some limitations: (i) it supports only few classes/objects per system; (ii) only 1-to-1 associations are supported; (iii) the generated behavior does not differentiate synchronous and asynchronous methods[1] calls used within sequence diagrams. (iv) engineers need to include unnecessary details in the UML model in order to generate the VHDL description as complete as possible. (v) depending on the amount of such details, generated VHDL descriptions need some minor manual modifications before they can be synthesized by a synthesis tool. This work improves [10] and proposes a new set of mapping rules for the UML-to-VHDL transformation, including the support of some object-oriented constructions which have not been initially supported, as well as the support for sequence diagrams full semantics.

In addition, in the embedded systems domain, non-functional requirements (e.g. deadlines, energy consumption, reduced footprint, communication latency, etc.) introduce crosscutting concerns in system design and implementation. Such concerns are not properly handled by traditional approaches (e.g. object-orientated, component-based approaches) due to the functional decomposition [11], [12] enforced by such approaches. This decomposition schema leads to modularization problems of crosscutting concerns, since their handling cannot be encapsulated within single units, which, in turn, results in scattered and tangled handling. Aspect-Oriented Programming [11] addresses crosscutting concerns modularization issues in software components by proposing

[1] In this text, a "method" encapsulates an object behavior that is executed when its associated message is received from other object.

special constructs called *aspects*. VHDL descriptions are somehow similar to a software source code files; they describe components structure and behavior using a language based on functional decomposition. Hence, VHDL descriptions may present the same modularization issues related to crosscutting non-functional requirements [13], [14], [15].

AMoDE-RT supports aspect-oriented concepts in UML/-MARTE models, through the *DERAF aspects* [16]. Thus, another contribution of this work is the VHDL implementation of some DERAF aspects. Aspects adaptations have been specified as mapping rules scripts that generate VHDL constructs to provide crosscutting concerns handling. By using the approach proposed, it is possible to generate a fully functional and synthesizable VHDL description from a more abstract UML model – there is no need for manual modifications on the generated descriptions. Engineers can generate both hardware and software source code from the same UML model [5], facilitating hardware/software co-design.

In order to demonstrate the feasibility of the proposed approach, as well as discusses the achieved improvements, this paper discusses the design of a FPGA-based implementation of a line-following robot. Results show a better utilization of FPGA resources, and also a small impact in project performance. In comparison with the results presented in [10], a greater amount of source code lines was generated from the UML model, which, in turn, it may lead to an effort decrease for creating the embedded system implementation, specially when designing larger systems. In addition, this case study has demonstrated that by using AMoDE-RT approach, it is possible not only to deal with non-functional requirements earlier in the design cycle, but also to obtain automatically a fully synthesizable VHDL description[2].

The rest of this text is organized as follows: section II discusses related works; section III describes how AMoDE-RT generates VHDL descriptions from aspect-oriented UML models; section IV outlines the line-following robot case study and discusses the obtained results. Finally, section V draws some conclusions and discusses directions for future works.

II. Related Works

Generating VHDL descriptions from high-level models has been proposed in various works [17], [18], [10], [8], [19]. For instance, COLA (*COmponent LAnguage*) tool allows system high-level specification and also the generation of a VHDL description from such a specification [18]. The generated VHDL description comprises the hardware structure and behavior, which is extracted from state machine diagrams. However, COLA has its own formal modeling semantics, which is not a standard, hindering its adoption in comparison with UML-based approaches/tools.

On the other hand, GASPARD is a tool that generates VHDL descriptions from UML/MARTE models [17], [8]. GASPARD uses concepts from logical view of MARTE *Hardware Resource Modeling* (HRM) package. Such a tool focuses on modularization and components mapping. It is necessary to define templates, which are static elements, in order to generate the system VHDL description. As a consequence, GASPARD generates only entity and components mapping. In addition, it does not support the code generation for system behavior.

[2]The generated VHDL description was synthesized, uploaded and executed on a FPGA development kit without any manual modification.

In [19], an approach is proposed aiming the generation of VHDL descriptions for system behavior validation. Sequence diagrams are used to specify system interactions, from which VHDL description is generated. Project constraints are specified in sequence diagram as MARTE stereotypes, defining system non-functional requirements subject to validation. However, that work generates VHDL files only for system validation; it does not generate VHDL description for the system functional requirements.

Recently, an increasing number of research works propose the use of Aspect-Oriented Programming (AOP) in hardware design. The use of AOP in conjunct with VHDL language is analyzed in [13]. This work identified some elements in VHDL which may be subject to aspects adaptations. According to that work conclusions, possible *join points* in VHDL are *process* and variables and signals assignments. These places are suitable for code injection, specially *before*, *after* and *around* each join point [13].

AspectVHDL language is an extension that includes AOP concepts in VHDL syntax [14]. AspectVHDL language allows the definition of *join point* for procedures, functions, data type definitions, entity architecture, and processes sensitivity list. However, in order to be able to define *join points* and hence to use AspectVHDL, the system VHDL description must be structured in functions and procedures.

This work differs from mentioned related works, since AMoDE-RT is based on MDE techniques and high-level UML/MARTE models, as well as it is supported by a flexible code generation tool such as GenERTiCA. The VHDL generation approach proposed in [10] has been enhanced through a proper support for high-level constructs available in UML. System behavior is completely specified by sequence diagrams, including the specification of non-functional requirements using MARTE stereotypes. This diagrams are used for generating the VHDL components behavior. Moreover, crosscutting non-functional requirements handling is speficied in a high-level and independent of platform way. This work proposes and demonstrates VHDL implementation of these high-level aspects. Hence, it is possible to generate a complete VHDL description from UML/MARTE high level specification. Such a description is synthesizable and fully functional, according to system requirements specified in the UML model.

III. From Aspect-Oriented UML models to VHDL descriptions

Aspect-oriented Model-Driven Engineering for Real-Time systems (AMoDE-RT) [5], [16] is an UML-based MDE approach that promotes the use of *Platform Independent Models* (PIM) as the main artifacts to specify embedded real-time system requirements, including system structure, behavior, constraints and non-functional properties. One of its main contributions is the separation of concerns when handling functional and non-functional requirements. For that, AMoDE-RT supports concepts of the *Aspect-Oriented Software Development* (AOSD) within UML/MARTE models by means of the *Distributed Embedded Real-time Aspect Framework* (DERAF).

Furthermore, in AMoDE-RT, system implementation is obtained automatically from UML/MARTE models by means of model transformations [5]. *Generation of Embedded Real-Time Code based on Aspects* (GenERTiCA) tool [20] supports this automatic generation of source code. GenERTiCA im-

plements a script-based code generation approach that uses a set of scripts to map UML elements into constructs, services, and/or APIs provided by a given target platform. Engineers may specify mapping rules scripts for distinct target platforms, including both software and hardware (i.e. using *Hardware Description Languages*).

In addition, it is important to highlight that GenERTiCA does not only generate source code but also performs aspects weaving. In other words, GenERTiCA weaves aspects adaptations into the generated code, allowing the use of aspects in model, even though the target implementation language does not support such concepts. Aspects adaptation are implemented as mapping rules scripts, enabling their portability for different platforms or/and applications. An adaptation may occur in two ways: in model level or in code level. At model level, adaptations may include or change elements in DERCS model [20] generated from UML/MARTE specification). These modification are accessible to mapping rules during code generation step. On the other hand, at code level, modifications affect directly the generated code without changing the input model.

A. Functional Requirements

A set of mapping rules to generate VHDL code from UML/MARTE models has already been proposed in a previous work [10]. However, as mentioned, [10] has some limitations; specially, when one considers the potential increase in the specification abstraction level by using object-oriented features, e.g. encapsulation and inheritance. Besides improving design complexity management, by using such abstractions, engineers may improve artifacts[3] modularity, and hence, their reuse.

In order to address the already mentioned issues, a new set of mapping rules have been created, as depicted in Table I. First column shows the UML meta-model elements; second column depicts the VHDL elements mapping, as proposed in [10]; and, third column shows the modifications proposed in this work for UML-to-VHDL mapping. As one can note, classes are mapped into entities and their related architectures. Moreover, in addition to the support for encapsulation and inheritance, an important contribution of these new mapping rules is the support for synchronous and asynchronous message sending.

Furthermore, it is worth mentioning that this work supports UML structural elements (e.g. classes and objects) and behavioral elements (e.g. interactions and actions). Some stereotype from *Hardware Resource Modeling* (HRM) and *Software Resource Modeling* (SRM) of MARTE profile are also covered, as described in Table I. Moreover, it is possible to extend the mapping rules to add different packages and elements. In summary, 29 scripts of mapping rules have been created, totalizing 2365 code lines. By using these mapping rules scripts, engineers are allowed to use high-level object-oriented concepts supported in UML: encapsulation by means of get/set methods; 1-to-n relationships such as aggregations and compositions; inheritance/generalization relationships; full semantics of sequence diagrams, such as combined fragments, synchronous and asynchronous method calls. For additional details, readers are encouraged to consult [21].

B. Non-Functional Requirements

AMoDE-RT uses DERAF aspects to handle crosscutting concerns in a platform independent way within UML/MARTE models. The high-level semantics of DERAF allow its aspects to be implemented in distinct target platforms, including those that do not support AOSD concepts [5]. Due to space constraints, this section provides only an overview of the VHDL implementation of three DERAF aspects used in the case study. Details on how aspects adaptations are implemented in VHDL have been discussed in [22].

TABLE I. MAPPING CONCEPTS FROM UML/MARTE TO VHDL

UML		VHDL[10]	New Mapping
Class	Structure	Entity-Arch. Pair	Entity-Architecture Pair
	Association	Entity Ports (1-to-1)	Components (1-to-n)
	Generalization	-	Components for concrete classes; functions, attributes and/or methods from parent class(es)
Attributes	Public	Entity Ports	-
	Private	Signals	Ports, Signals or Constants
	Read-Only	-	Constants
Data Types	Enumeration	-	Enumerations within packages
	Composite	-	Vectors
Method/ Operation	Behavior	Process	Process or *inline* code
	Message Occurrence Specification	Entity Ports	Signals assignment or process activation
	TimedEvent stereotype	-	"Active" Process

1) PeriodicTiming: PeriodicTiming aspect [16] deals with the periodic execution of one or more *active objects*[4] behaviors. A similar behavior is achieved triggering periodically a *process* within VHDL entities. Therefore, the proposed VHDL implementation of *PeriodicTiming* considers VHDL entities as active objects, and each of periodic behavior as a process. The execution frequency of periodic processes is controlled via a clock divider component associated to the entity. Therefore, in summary, *PeriodicTiming* aspect adapts the generated code by including a clock divider for each affected entity, as well as creates the interconnection logic between this component and the affected processes.

The clock divider is included as a FPGA platform component through mapping rules of platform configuration. Such a component is included in the generated VHDL files when *PeriodicTiming* aspect is used in the UML model. In fact, *PeriodicTiming* mapping rules scripts do not define a clock divider component; they only use its services. The target platform library should provide resources for aspect implementation related to the non-functional requirements handling [5]. Such an approach makes *PeriodicTiming* aspect portable to other platforms, which have their own mechanisms to control the execution frequency of active objects. According to AMoDE-RT approach, embedded system specifications (i.e. models) must be platform independent, in order to allow the reuse of model elements in distinct projects and with different target platforms. Therefore, when GenERTiCA identifies that aspects have been specified in the UML/MARTE model, it includes the necessary platform services used in their implementation, i.e. mapping rules scripts. In this case, the clock divider component is included as a FPGA platform component by means of platform configuration mapping rules, but only when *PeriodicTiming* aspect is used in the model.

2) DataFreshness: Data freshness [23] can be understood as the temporal validity of real-time data. Such kind of

[3] e.g. model, source code, description of components, etc.

[4] An *active object* executes autonomously its behaviors in parallel with other active objects. A *passive object* only executes its behaviors in response to method calls from other objects either active or passive.

non-functional requirement is important for some classes of embedded system applications, such as control systems. In such systems, data can only be used whether it is temporally valid. For instance, before using data coming from other components (e.g. sensors), it is necessary to check whether these data are updated. To address such a non-functional requirement, *Datafreshness* aspect [16] associates timestamps with controlled data, checking them before using such data.

VHDL implementation of *DataFreshness* creates constants representing the temporal validity of affected object attributes. An additional process is created within the entity that represents the affected object. This process controls whether values are updated by using a one-bit signal, which, in turn, is checked before any reading access of the controlled attribute.

However, it is worth mentioning that this approach is one possible implementation of *Datafreshness* aspect. Other engineers may handle data freshness non-functional requirements in a different way. This VHDL implementation has been proposed as proof-of-concept for AMoDE-RT aspect-oriented MDE approach, and hence, more efficient implementations are indeed feasible.

3) COPMonitoring: Computer Operating Properly Monitoring (COPMonitoring) aspect deals with the identification of faults in system components (hardware and software). This aspect adds a mechanism to monitor system components, checking whether they are executing correctly. For that, monitored components must periodically communicate with such a mechanism to inform they are executing without problems. When any component fails to accomplish such a communication, it is considered faulty and the system is warned on such an issue. Thus, the system may take any measure to overcome the problems caused by such a faulty component.

The proposed VHDL implementation of *COPMonitoring* aspect consists in a *watchdog timer*, an interrupt mechanism, and all glue logic that implements the mentioned behavior. One watchdog timer component instance is created for each affected entity, which, in turn, must handle the watchdog timeout interruption, i.e. it must handle the faults identified by the watchdog timer. In this implementation, when a watchdog timeout occurs, the system is reset. On the other hand, the behavior of the affected entity is modified to include the logic that will reset the watchdog timer.

Watchdog components shall be provided by the component library of the chosen target FPGA platform. As it happened with the clock divider component in the *PeriodicTiming* aspect, when GenERTiCA identifies that *COPMonitoring* aspect is specified in the UML model, all required services are included into the generated VHDL description. Morover, it is important to highlight that *COPMonitoring* the first aspect that deal with faults within DERAF framework. It is an additional contribution of this work.

IV. Case Study and Results

This work has already been validated through some case studies. This paper presents a line-following robot which has been designed using AMoDE-RT approach and implemented as an ASIP on a Xilinx Spartan-6 FPGA development kit. The robot is composed of two infrared sensors: one detects the left-hand side of the line, while the other one detects the right-hand side. An Arduino Uno board was used as a Analog-to-Digital Converter. It converts sensors analog signals into digital signals, which, in turn, are sent to FPGA board input ports. The robot has two servomotors to spin left/right wheels. The movement control system commands the robot to turn left/right based on sensors input.

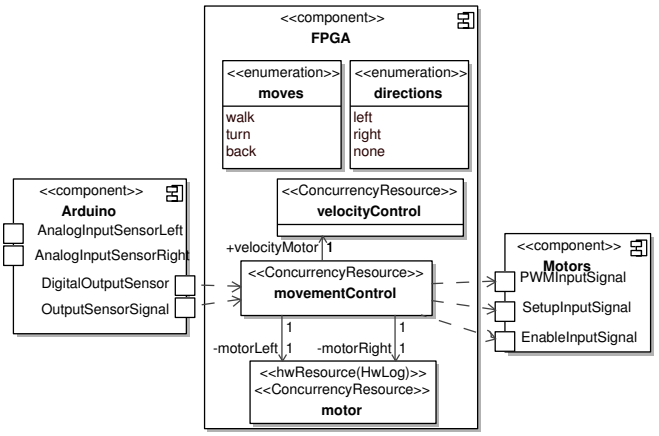

Fig. 1. Composite Structure Diagram of Robot Control Systems

System requirements have been specified in the UML/-MARTE model following AMoDE-RT modeling guidelines [16]. Thereafter, a VHDL description has been generated by GenERTiCA, using the created UML/MARTE model and the proposed set of VHDL mapping rules as input. Figure 1 shows the composite structure diagram of this robot control system. As one can see, three active objects are implemented on FPGA, which is connected to the Arduino and wheels servomotors. *MovementControl* is responsible for controlling the robot direction. *Motor* generates a PWM signal to control the rotation speed of left/right wheels, and *VelocityControl* implements a speed ramp in order to smooth the robot movement.

This project presents three non-functional requirements: (i) active objects must be periodically activated according to the execution frequency specified in *TimedEvent* stereotype; (ii) servomotor faults must be handled within the robot control system; (iii) sensor and motor data have temporal validity. These requirements has been handled through *PeriodicTiming*, *COPMonitoring* and *DataFreshness* aspects. Figure 2 presents the *Aspects Crosscutting Overview Diagram* (ACOD) showing these DERAF aspect.

In this evaluation, the line-following robot has been designed in two versions. The first one does not use DERAF aspects in UML/MARTE model. Non-functional requirements handling has been implemented manually, i.e. the generated VHDL description was manually modified in order to include these requirements handling. This implementation was done by the same designer that made the implementation of aspects, used in second version. On the other hand, in the second version, DERAF aspects have been used, and the complete VHDL description has been automatically generated. There was no manual modification in the generated VHDL description. It is worth pointing out that both versions have implemented the same set of functional and non-functional requirements.

Thereafter, ISE WebPack[5] tool was used to compile and synthesize the two VHDL descriptions into a Spartan 6 FPGA (XC6LX16-CS324). Synthesis results are presented in Table

[5] www.xilinx.com

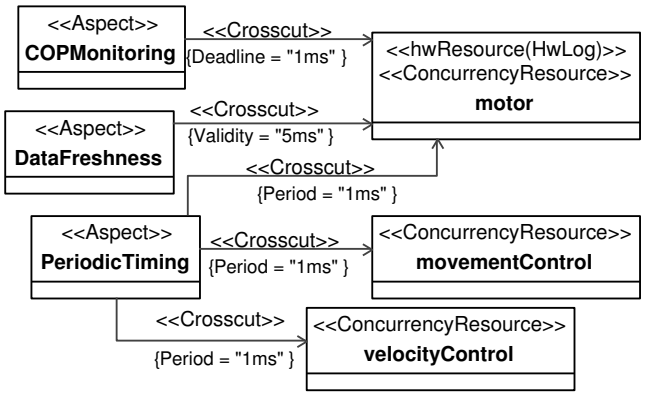

Fig. 2. ACOD diagram of line-following robot

II. First column shows analyzed metrics. Column "FPGA resources" presents the total amount of resources available in FPGA. Third and fourth columns depict metrics for, respectively, "object-oriented" version (V1) and the aspect-oriented version (V2). VHDL description of both version were generated by GenERTiCA. However, as mentioned, V1 was modified manually in order to include non-functional requirements handling; and V2 was fully generated automatically due to the use of DERAF aspects and the proposed VHDL mapping rules. Finally, last column presents variation (percentage) in presented metrics of the two versions.

As one can see, Lines of Code (LOC) increased ca. 16% in V2. However, the amount of utilized FPGA hardware has doubled, since used FPGA resources increases ca 184% on average[6]. Such a result indicates that, for VHDL descriptions, an increase in LOC cannot be related to a proportional increase in FPGA resources utilization. In software, on the other hand, this relation can be established, since the binary code size usually increases proportionally to LOC.

Furthermore, even though the generated system ASIP in V2 has a large increase in FPGA resources utilization, its performance is better: it presents a shorter critical path (6.22% decrease in minimum period in comparison with V1), leading to a greater operation frequency (6.65% increase). This improvement may had been achieved due to a better modularization of system elements achieved by using DERAF aspects, since concerns are better separated among components as indicated in *TR* metric presented in Table III. Such a modularization improvement has also impacted positively on utilization ratio of FFs and LUTs per slice: 2.02 FF/slice and 3.16 LUT/slice in V1; and 2.65 FF/slice and 3.62 LUT/slice. In other words, it could indicate that the place and routing algorithms of the synthesis tool use more efficiently the available slices in V2, since the increase in FF/LUT is not proportional to the increase of slices. However, it is necessary to perform a more detailed analysis to confirm such hypotheses. In larger systems, this difference in utilization ratio may impact strongly on system performance, i.e. the higher the FF/LUT utilization ratio, the lesser the amount of used slices and better the system overall performance. Moreover, these results are different from those presented in [15], where aspects had a lesser impact on area usage (3%) but did not affect system performance. Finally, it is worth pointing out that the trade-off between performance and

[6]300% increase in IOB metric in V2 was included in this average amount

TABLE II. SYNTHESIZED SYSTEM METRICS

	FPGA resources	V1 Manual	V2 DERAF	
Metrics	**Available**	**Used**	**Used**	**Variation**
Slices	2.278	123	262	113%
Flip-Flops (FF)	18.224	248	693	179.43%
Look-Up Table (LUT)	9.112	389	949	143.95%
Input/Output Blocks (IOBs)	232	20	80	300%
Lines of Code (LOC)		416	482	15.86%
Minimum period (ns)		4.64	4.35	-6.22%
Maximum Frequency (Mhz)		215.50	229.85	6.65%

TABLE III. IMPACT OF DERAF ON LINE-FOLLOWING ROBOT DESIGN

Aspect	LOC	LOWC	LOAC	CDLOC	TR	AB
Periodic Timing	206	277	34	44	15.88%	2.08
COPMonitoring	206	299	98	50	16.72%	0.94
DataFreshness	206	319	84	28	8.77%	1.34

Note: *LOC* - Lines of Code; *LOWC* - Lines of Woven Code; *LOAC* - Lines of Adaptation Code; *CDLOC* - Concern Diffusion over LOC; *TR* - Tangling Ratio; *AB* - Aspectual Bloat

FPGA area depends on system requirements and constraints. Hence, engineers shall chose a design approach that meets the projects requirements and constraints.

Table III shows reusability analysis of DERAF aspects and also their impact on the system design in V2. The first three columns indicate the number of code lines of: (i) the whole description without any aspect (*LOC*), (ii) the whole description with aspects (*LOWC*), and (iii) all aspect adaptation (*LOAC*). The remaining columns indicate AOSD-specific metrics. *CDLOC* [24] indicates the number of context switches between functional- and non-functional requirements handling code. CDLOC is used to calculate the *Tangling Ratio (TR)* [25]. According to [25], *TR* indicates how much functional and non-functional concerns are intermixed. The obtained average value of 13.79% is a good indication in comparison with results presented in [25]. *AB* column [24] shows the increase in the number of aspect-related code lines into the final system implementation after performing the aspects weaving process. AB indicates also the impact of aspects on reusability as well as on the final system. In this sense, *COPMonitoring* shows a lower *AB*, meaning a low impact on the system and on its reuse, since it affects fewer elements. However, *COPMonitoring* presents the highest *TR*. Intuitively, AB metric may be improved in larger systems, specially when *COPMonitoring* affects a larger amount of elements.

V. FINAL REMARKS

This paper discussed an approach that allows the automatic generation of fully synthesizable VHDL descriptions from high-level UML/MARTE models. By applying MDE and AOSD, the proposed approach deals with functional and non-functional requirements in a platform independent way, opening room for hardware and software co-design of embedded systems. One of the main contributions of this work is the definition of UML-to-VHDL mapping rules, which have been implemented as code generation scripts for the GenERTiCA tool. These mapping rules include the support for key object-oriented features supported in UML, namely, encapsulation, inheritance, and 1-to-n associations, as well as the support for synchronous and asynchronous method calls (from sequence diagrams). In addition, a VHDL implementation of some DERAF aspect have been created, in order to handle with crosscutting concerns related to system non-functional requirements. Such a handling code is woven into functional

requirements code. Hence, it is important to highlight that, in comparison with [10], a greater amount of VHDL statements is generated, resulting in complete and fully synthesizable VHDL descriptions. Engineers do not need to modify manually the generated VHDL files, since our experiments demonstrated that the generated VHDL description was synthesized without any error into a FPGA device.

In order to assess the proposed work, this papers presented the design of the control systems of a line-following robot, which has been implemented as an ASIP on a FPGA development kit. AMoDE-RT has been successfully applied, impacting positively in system implementation, for instance, in components modularization (*TR metric* is lower than 20%) and system performance (circuit frequency increased 6.65%). It is worth mentioning that occupied FPGA resources increased in the aspect-oriented version of this system. However, slices are used more efficiently; this may lead to a better FPGA resources usage in larger systems without penalizing system performance. On the other hand, as DERAF aspects led to a better modularization of crosscutting concerns, their usage opens room for a good reutilization rate. As AMoDE-RT advocates for using platform independent specifications, e.g. UML models and DERAF aspects, design complexity management may be improved due to an enhanced opportunity for hardware/software co-design, as well as reuse of artifacts [5].

Despite the proposed work has been successfully applied int design of a FPGA-based embedded system, the proposed VHDL implementation of DERAF aspects has presented some issues. Some aspects adaptation implementation have interfered in the project specification, e.g. *DataFreshness* reads signal value associated to an OUT port, but such a situation is forbidden in VHDL. To overcome this problem, this port type had to be changed to INOUT, so that this signal could be read internally within the entity. Other issue is related to the interfere among aspect adaptations. For instance, a signal created in *COPMonitoring* had interfered the implementation created for *DataFreshness*, and hence, *COPMonitoring* implementation, i.e. its mapping rule scripts, had to be modified.

As future work, more case studies are already being performed. A similar analysis is going to be executed, in order to demonstrate the suitability and feasibility of using platform independent aspects to design FPGA-based embedded real-time systems. New rules for UML elements which are not yet covered must be defined, e.g. state machine diagrams and their relationship with sequence diagrams. In addition, other DERAF aspects are going to be implemented in VHDL. However, for that, it is important to obtain or create a set of reusable soft IP components, in order to implement aspects adaptations using their services.

REFERENCES

[1] E. Monmasson and M. Cirstea, "FPGA Design Methodology for Industrial Control Systems - A Review," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1824 –1842, aug. 2007.

[2] F. Salewski and A. Taylor, "Systematic considerations for the application of FPGAs in industrial applications," in *IEEE International Symposium on Industrial Electronics*, 2008, pp. 2009–2015.

[3] D. Hästbacka *et al.*, "Model-driven development of industrial process control applications," *Journal of Systems and Software*, vol. 84, no. 7, pp. 1100–1113, 2011.

[4] P. Mohagheghi *et al.*, "An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases," *Empirical Software Engineering*, vol. 18, no. 1, pp. 89–116, 2013.

[5] M. A. Wehrmeister, C. E. Pereira, and F. Rammig, "Aspect-oriented model-driven engineering for embedded systems applied to automation systems," *IEEE Transactions on Industrial Informatics.*, vol. 9, no. 4, pp. 2373–2386, Nov 2013, doi:10.1109/TII.2013.2240308.

[6] W. McUmber and B. H. C. Cheng, "Uml-based analysis of embedded systems using a mapping to vhdl," in *IEEE International Symposium on High-Assurance Systems Engineering*, 1999, pp. 56–63.

[7] J. Vidal *et al.*, "A co-design approach for embedded system modeling and code generation with uml and marte," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, April 2009, pp. 226–231.

[8] M. Elhaji *et al.*, "System level modeling methodology of noc design from uml-marte to vhdl," *Design Automation for Embedded Systems*, vol. 16, no. 4, pp. 161–187, 2012.

[9] S. Wood *et al.*, "A model-driven development approach to mapping uml state diagrams to synthesizable vhdl," *IEEE Transactions on Computers*, vol. 57, no. 10, pp. 1357–1371, Oct 2008.

[10] T. Moreira *et al.*, "Automatic code generation for embedded systems: From UML specifications to VHDL code," in *Proc. of 8th IEEE International Conference on Industrial Informatics (INDIN)*, July 2010, pp. 1085–1090, doi:10.1109/INDIN.2010.5549590.

[11] G. Kiczales *et al.*, "Aspect-oriented programming," in *Proc. of European Conference on Object-Oriented Programming*. Berlin: Springer-Verlag, 1997, pp. 220–242.

[12] Elrad *et al.*, "Discussing aspects of AOP," *Communications of the ACM*, vol. 44, no. 10, pp. 33–38, Oct. 2001.

[13] M. Engel and O. Spinczyk, "Aspects in hardware: what do they look like?" in *Proceedings of the 2008 AOSD workshop on Aspects, components, and patterns for infrastructure software*, ser. ACP4IS '08. New York, NY, USA: ACM, 2008, pp. 5:1–5:6.

[14] M. Meier, S. Hanenberg, and O. Spinczyk, "AspectVHDL Stage 1: The prototype of an aspect-oriented hardware description language," in *Proceedings of the 2012 workshop on Modularity in Systems Software*, ser. MISS '12. New York, NY, USA: ACM, 2012, pp. 3–8.

[15] T. Muck, M. Gernoth, W. Schroder-Preikschat, and A. Frohlich, "A Case Study of AOP and OOP Applied to Digital Hardware Design," in *Brazilian Symposium on Computing System Engineering (SBESC)*, nov. 2011, pp. 66 –71, doi: 10.1109/SBESC.2011.23.

[16] M. A. Wehrmeister, E. P. de Freitas, A. P. D. Binotto, and C. E. Pereira, "Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems," *Mechatronics*, 2014, *to appear.* doi:10.1016/j.mechatronics.2013.12.008.

[17] I. Quadri *et al.*, "MARTE based modeling approach for Partial Dynamic Reconfigurable FPGAs," in *Proc.of Embedded Systems for Real-Time Multimedia*, 2008, pp. 47–52.

[18] Z. Wang *et al.*, "A model driven development approach for implementing reactive systems in hardware," in *Proc.of Specification, Verification and Design Languages (FDL)*, 2008, pp. 197 –202.

[19] E. Ebeid, D. Quaglia, and F. Fummi, "Generation of vhdl code from uml/marte sequence diagrams for verification and synthesis," in *Euromicro Conference on Digital System Design*, 2012, pp. 708–714.

[20] M. Wehrmeister, E. Freitas, C. Pereira, and F. Rammig, "Genertica: A tool for code generation and aspects weaving," in *Intl. Symp. Object-Oriented Real-Time Distributed Computing (ISORC)*. IEEE Computer Society, 2008, pp. 234–238, doi:10.1109/ISORC.2008.67.

[21] M. Leite, C. V. Damiani, and M. A. Wehrmeister, "Enhancing automatic generation of VHDL descriptions from UML/MARTE models," in *Proc. of International Conference on Industrial Informatics (INDIN'14)*. Piscataway, NY, USA: IEEE Eletronics Society, 2014, pp. 1–5.

[22] M. Leite and M. A. Wehrmeister, "Aspect-oriented model-driven engineering for FPGA/VHDL based embedded real-time systems," in *Intl. Symp. Object-Oriented Real-Time Distr. Computing (ISORC)*. IEEE Computer Society, 2014, pp. 261–268, DOI: 10.1109/ISORC.2014.45.

[23] Burns and Wellings, "HRT-HOOD: A structured design method for hard real-time systems," *Real-Time Systems*, vol. 6, no. 1, pp. 73–114, 1994.

[24] E. Figueiredo *et al.*, "On the maintainability of aspect-oriented software: A concern-oriented measurement framework," in *12th European Conference on Software Maintenance and Reengineering*, 2008, pp. 183–192.

[25] J. a. M. Cardoso *et al.*, "LARA: an aspect-oriented programming language for embedded systems," in *Proceedings of the 11th annual international conference on Aspect-oriented Software Development*, ser. AOSD '12. New York, NY, USA: ACM, 2012, pp. 179–190.