Towards a Verification Approach for Reconfigurable Embedded Systems

Fatma Krichen[1,2], Amal Gassara[1], Bechir Zalila[1], and Mohamed Jmaiel[1]
[1]ReDCAD, University of Sfax, Tunisia
[2]IRIT, University of Toulouse, France
Email: Fatma.Krichen@irit.fr

***Abstract*—This paper proposes a verification approach of reconfigurable Distributed Real-time Embedded (DRE) systems. Since dynamic reconfiguration significantly complicates the execution of such a system, it is necessary to ensure the preservation of non-functional properties after applying a such reconfiguration. The proposed approach allows thus verifying non-functional properties at design time. Using our verification approach, the designer can easily verify these properties without deep knowledge of existing verification techniques.**

I. Introduction

Embedded systems are typically constituted of heterogeneous components including both hardware and software elements. Dynamic reconfiguration consists of architectural or behavioral modifications of a system during execution. It significantly complicates both development and execution of DRE systems. In Fact, these systems have strict time and resource constraints which should be respected. Maintaining their non-functional properties is needed when reconfigurations are applied. Verifying non-functional properties of these systems is thus required and particularly for reconfigurable ones.

In this paper, we propose a suitable approach for verifying the following non-functional properties: CPU usage, meeting of thread deadline, memory usage and bandwidth usage.

The rest of this paper is organized as follows. In Section II, we describe our proposed model-based verification approach. As a proof of concept, a GPS (Global Position System) case study is considered in Section III. In Section IV, we briefly review some related work that address the verification of non-functional properties of real-time embedded systems. Finally, Section V concludes this paper and presents some future work.

II. Model-Based Verification Approach

We propose an integrated verification approach of reconfigurable DRE systems. Based on models, our approach enables to check whether a such system satisfies non-functional properties using well-defined formalisms. In the following, we describe the modeling and verification stages.

A. Modeling of reconfigurable DRE systems

A reconfigurable system introduces a set of configurations (e.g. modes). These configurations should be specified using architectural style which captures and characterizes all system configurations. The architectural style is identified by structured components, connectors and structural constraints. Hence, a configuration belonging to an architectural style consists of a set of structured component instances and connectors of the corresponding style and respects its structural constraints. To verify in the next step the non-functional properties, the architectural style should be allocated on the hardware architecture. As the hardware architecture is unchanged, the allocation is made from software architecture models to execution supports using well-defined allocation policies (as already detailed in a previous work [1]).

We introduce a new meta-model called *ArchStyle* to describe the previous defined software concepts of reconfigurable DRE systems. Figure 1 defines the architectural style by introducing the *ArchitecturalStyle* meta-class which allows to present multi-state applications. We also introduce the *Allocation* meta-class to specify the allocation of architectural style to execution supports. This allocation has non-functional and allocation constraints that should be respected. The execution supports will be specified using MARTE profile.

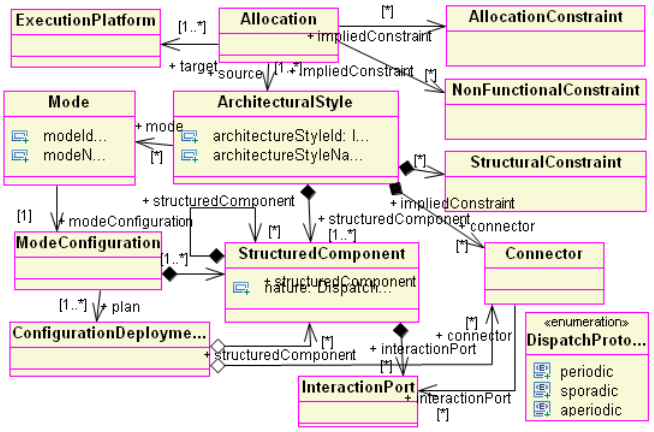

Fig. 1. The *ArchStyle* meta-model

B. Verification of non-functional properties

The second stage in our proposed approach consists in checking non-functional properties of such a system at design time. Since a reconfigurable system is composed of a set of configurations, the verification of non-functional properties should be performed for each configuration. But this procedure is enough hard especially when having a non-predefined

978-1-4673-2713-8/12/$31.00 ©2012 IEEE

number of configurations. For this purpose and based on the specified architectural style, we define the property-specific Worst Case Execution Instance (WCEI) to verify the corresponding property. We assume that if the property is satisfied at the WCEI, it is so satisfied for all configurations belonging to the architectural style. Our approach takes in consideration three kinds of thread:

- Periodic thread is characterized by a constant time interval between two successive activations. It is defined by a deadline (D_p), a period (P_p) and an execution requirement (C_p).
- Sporadic thread can be executed at arbitrary times with defined minimum inter-arrival times between two consecutive activations (P_{sp}). It is also identified by a deadline (D_{sp}) and an execution requirement (C_{sp}).
- Aperiodic thread is activated only once and it is characterized by an arrival time and an execution requirement (C_{ap}).

Our verification framework deals with performance properties like CPU usage, meeting of thread deadlines, memory usage and bandwidth usage.

The WCEI of the CPU usage and meeting of thread deadlines is deduced based on the following points: each CPU is allocated by a maximum number of threads (i.e. structured component instances) and the worst case execution of sporadic threads is taken into consideration (i.e. sporadic threads will be considered as periodic threads). To verify both CPU usage and meeting of thread deadlines, we use the RMS scheduling algorithm [2] and the Cheddar framework [3]. The CPU usage test consists in comparing the processor usage factor to a given bound. The processor usage factor is computed with the formula $\sum_{i=0}^{n}(Ci/Pi)$ [2], where n is the number of threads, Ci is the execution time of thread *i* and Pi is the period of thread *i*. Using the RMS scheduling algorithm, we verify that the processor usage factor is less than $n(2^{1/n}-1)$ according to [2], where n is the number of threads. The deadline meeting test consists in verifying that thread response time computed by simulation respects its deadline. In fact, Cheddar verifies the meeting of deadlines and indicates the threads that do not meet their deadlines.

The WCEI of the memory usage is obtained by considering the maximum number of threads using their associated memories and therefore by the highest usage rate of memories. At design level, each structured component is characterized by the property *memory size* which presents the memory footprint. Periodic and sporadic threads permanently allocate memory during system execution while aperiodic threads allocate memory only when they exist. Based on our approach, the memory usage verification consists in checking at the corresponding WCEI that the used memory size of threads being executed on the same node does not exceed the node memory size at each instant.

The WCEI of the bandwidth usage is deduced based on the two following points: each bus is allocated by a maximum number of software connections and all software connections will be executed at the same time. A component-based architecture is characterized by logical connections between components. These connections can be local or distributed. We verify the existence of a hardware bus for each software connection and the absence of bandwidth overflow for each hardware bus. For this purpose, an algorithm was implemented.

TABLE I
NON-FUNCTIONAL PROPERTIES OF GPS SYSTEM COMPONENTS

Structured Component	Nature	Period Deadline	WCET	memory size
Receiver	sporadic	100 ms	20 ms	0.9 MB
Position	sporadic	100 ms	20 ms	0.5 MB
TreatmentUnit	sporadic	100 ms	20 ms	0.75 MB
Decoder	sporadic	100 ms	20 ms	0.1 MB
Encoder	sporadic	100 ms	20 ms	0.5 MB
GpsSatellite	periodic	400 ms	30 ms	0.9 MB
GpsControlBase	periodic	400 ms	30 ms	0.9 MB

III. CASE STUDY

To illustrate our verification approach, we consider as a classical case study: a GPS (Global Positioning System) [4]. The satellite sends to earth an encrypted signal which contains various information useful for localization and synchronization. The control base receives and sends information to satellites in order to synchronize the clocks of satellites. For simplicity's sake, many functions of this case study have been omitted. We only detail the architecture of the terminal. Both satellite and control base are represented by basic components. Each component is specified with a set of non-functional properties defined as target values with the *StructuredComponent* Stereotype. The table I presents the properties of components. Figure 2 shows the allocation of GPS architectural style to GPS terminal hardware and GPS satellite hardware. The top part of Figure 2 shows the architectural style of GPS while the down part presents the hardware part (e.g. GPS terminal node and GPS satellite node). The memory size of the GPS Terminal node is 250 MB while the memory size of the GPS Satellite node is 200 MB. The frequency of each processor is 800 MHz while the bandwidth of each bus is 200 b/s.

After the specification of the considered case study, we pass to check the non-functional properties. The DRE models elaborated within the modeling stage represent the input of the verification stage in our proposed framework. Figure 3 which presents a part of the output of the integrated Cheddar framework shows that the CPU usage property is well checked for the *cpu1*, *cpu2* and *cpu* processors. The memory usage and the bandwidth usage are also checked for the GPS terminal node and the GPS satellite node.

IV. RELATED WORK

Several approaches allowing the verification of real-time embedded systems have been proposed based on various formalisms and formal specification languages.

The classic scheduling theory [2], such as Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF), is widely used for the verification of real-time embedded systems. RMS is a fixed priority scheduling algorithm while

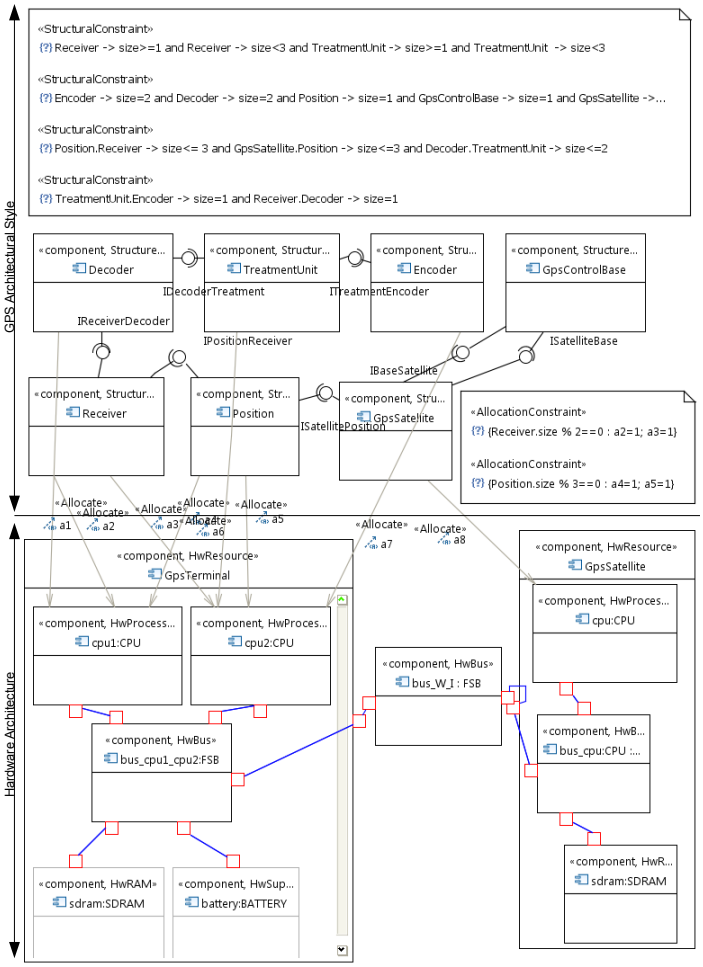

Fig. 2. Allocation of the architectural style to GPS terminal hardware and GPS satellite hardware

```
Scheduling feasibility, Processor cpu1:CPU :
- The base period is 2000 (see [18], page 5).
- Processor utilization factor with period is 0.40400 (see [1], page 6)
- In the preemptive case, with RM, the task set is schedulable because

Scheduling feasibility, Processor cpu2:CPU :
- The base period is 5000 (see [18], page 5).
- Processor utilization factor with period is 0.64440 (see [1], page 6)
- In the preemptive case, with RM, the task set is schedulable because

Scheduling feasibility, Processor cpu:CPU :
- The base period is 1000 (see [18], page 5).
- Processor utilization factor with period is 0.00600 (see [1], page 6)
- In the preemptive case, with RM, the task set is schedulable because
```

Fig. 3. The CPU usage verification in the GPS system

EDF is a dynamic priority (deadline driven) scheduling algorithm. EDF can reach 100% of CPU consumption while RMS does not exceed 69% of CPU consumption when we have an important number of threads. Contrary to EDF, RMS has a simple implementation. Moreover, RMS behaves better than EDF in the case of overload and is easily implanted with classical operating systems. In another part, a real-time scheduling tool called Cheddar [3] is proposed in order to check thread temporal constraints of real-time embedded systems. It provides feasibility tests as well as a scheduling simulation engine. Feasibility tests allow the verification of task temporal constraints without computing the scheduling of the application while the scheduling simulation engine computes first the scheduling of the application and then applies event analyzers to check temporal properties. Cheddar is designed to be open and flexible. In addition, the TASM language [5] aims at capturing three key aspects of real-time system behavior: functional behavior, temporal behavior and resource usage. It allows the verification of temporal properties such as worst-case execution time and resource properties such as memory. The temporal properties are analyzed using a translation of TASM specifications to timed automata.

The previous presented approaches ensure the formal verification of non-functional properties of static real-time embedded systems. The verification of reconfigurable ones is not considered. In this direction, the authors in [6] present an integrated model-based development approach allowing the modeling as well as the formal verification of dynamic adaptation behaviour of embedded systems. However, this work treats only the behavior adaptation of embedded systems. Moreover, the formalism SAS is very complex and requires many skills.

V. Conclusion and Future Work

In this paper, we proposed an approach allowing the verification of non-functional properties of reconfigurable DRE systems. Our approach combines different formalisms. We used RMS algorithm and Cheddar framework to verify both CPU usage and meeting of thread deadlines. We defined two algorithms to verify both the memory and bandwidth usage. We developed an ECLIPSE plug-in which presents a graphical editor and integrates Cheddar framework.

Our framework allows to transparently verify non-functional properties of systems at design time without having knowledge for verification techniques. However, our framework may be extended to support additional non-functional properties. As future work, we plan to propose a model-driven approach ensuring the development of reconfigurable DRE systems.

References

[1] F. Krichen, B. Hamid, B. Zalila, and M. Jmaiel, "Towards a model-based approach for reconfigurable distributed real time embedded systems," in *Proceedings of the 5th European Conference on Software Architecture*. Springer, 2011.

[2] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, no. 1, 1973.

[3] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *Proceedings of the 2004 annual ACM SIGAda international conference on Ada*. ACM, 2004.

[4] F. Krichen, B. Zalila, M. Jmaiel, and B. Hamid, "A middleware for reconfigurable distributed real-time embedded systems," in *Proceedings of the ACIS International Conference on Software Engineering Research, Management and Applications SERA (selected papers)*, ser. Studies in Computational Intelligence. Springer, 2012.

[5] M. Ouimet, G. Berteau, and K. Lundqvist, "Modeling an Electronic Throttle Controller Using the Timed Abstract State Machine Language and Toolset," in *Workshops and Symposia at MoDELS in Software Engineering*. Springer, 2006.

[6] R. Adler, I. Schaefer, T. Schuele, and E. Vecchi, "From Model-Based Design to Formal Verification of Adaptive Embedded Systems," in *Proceedings of the 9th International Conference on Formal Engineering Methods*. Springer, 2007.