Co-modelling and simulation with multilevel of granularity for real time electronic systems supervision

Manel Khlif, M. Shawky

Co-modelling and simulation with multilevel of granularity for real time electronic systems supervision

Manel KHLIF, Mohamed SHAWKY
Heudiasyc UMR 6599
Université de Technologie de Compiègne
Centre de Recherches de Royallieu
BP 20529
60205 COMPIEGNE cedex FRANCE
Firstname.lastname@hds.utc.fr

Abstract

This paper describes a new modelling and simulation approach in order to observe the system behaviour and eventual faults. It is based on hardware-software co-modelling with multi-level of granularity to get multi-level of simulation speed. Our modelling approach let integrate the hardware specifications to the functional model in order to establish also hardware supervision models after HW/SW co-modelling and simulation.

To reach this objective we have to have a relationship between the desired accuracy of the supervision and the level of granularity of the hardware-software model, for every ECU (Electronic Computing Units) that belongs to an automotive embedded architecture.

1. Introduction

The technological development encourages the car manufacturers to propose advanced functions for driving assistance that involve more than one computing unit. In fact, a computing unit uses information issued from sensors or other computing units, illustrating a "system with distributed functions". In a vehicle, the functions are sometimes distributed on several components or subsystems (computing units, wires, sensors, actuators…), communicating with a specific interconnection network [15]. However, one of the disadvantages of this distribution is the difficulty of the real time supervision to detect and localize a hardware fault.

To bring out the advantages of a highly distributed architecture, we propose a modelling methodology that benefits from the existing link between the software and the hardware platforms.

Our contribution in this paper is related to the very first phase of model-based diagnosis [9]. We reproduce the appropriate behaviour of the system in a set of comprehensible models showing at the same time the hardware and the software behaviours (*Figure 1*).

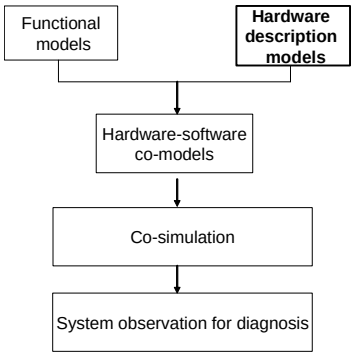

Figure 1. Hardware and software models for system observation for diagnosis

To present our contributions, this paper is structured as follows:

First, we present the need of hardware-software co-modelling for fault detection.

Then, we present the related works done in the field of the HW-SW co-design.

In section 4, we explain our contribution. We show a relationship between the accuracy required for fault detection and the level of granularity required for the hardware-software co-model. We try to find the good compromise between fault detection accuracy and the simulation accuracy.

In section 5, we use SystemC as a working environment for the hardware-software co-modelling of our embedded electronic architecture. We present the results in sections 6 and 7.

Finally, in the last section we conclude this paper and present our future works.

2. Need of Hardware-Software co-modelling

Usually, car manufacturers use software models expressing the embedded functions. However, when the hardware architecture reaches a complicated level of functional distribution, it becomes difficult to a diagnosis designer to maintain the hardware-software link for each sub-function of the system in the diagnosis model, especially in systems with highly distributed functions.

We believe that every sub-function has a link with at least one hardware sub-component, and a hardware fault appears in the system as a functional fault. So if we detect the functional fault we can localize the hardware fault if we know exactly the existing link between the sub-function and the hardware sub-component (*Figure 2*).

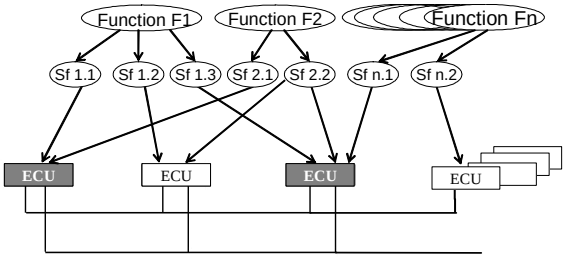

Figure 2. software sub-functions and hardware components links

The electronic distributed architecture that we study is embedded on board of a vehicle. Its "Software Architecture" describes the implementation of application functions proposed to us by a car manufacturer using the "Hardware Architecture" that represents all the hardware resources: it is composed of a set of ECUs connected by the interconnection bus CAN (Controller Area Network)[10].

Every ECU is composed by a processor, a memory, CAN interface and eventually Inputs/Outputs. We aim at modelling the functional distribution on this hardware architecture to simulate distributed real time software-hardware architecture. It allows the supervision designer to use the co-models for simulations to test and observe the system behaviour, off-line without being intrusive.

The existing techniques of fault detection and diagnosis, especially of electronic systems, are mainly based on functions and algorithms without hardware components description (architecture and behaviour), and need to be more accurate by expressing simultaneously both hardware and software behaviours, in order to lead to more accurate results of fault detection.

In the next section, we present some research works adding hardware information to the functional model in order to enhance systems manipulations.

3. Hardware-Software co-design for supervision

Within the context of co-design methodologies, concurrent hardware and software techniques have been proposed in the literature employing for example SpecC to add more detail at the specification level [3]. The adoption of various formal languages for co-simulation, like SDL and C [5] is mainly used for the design of reactive systems like telecom systems such as wireless protocols employing different and standardized formalisms. On the other hand, functional models compatibles with HDL (Hardware description language) models [11] are used to get accurate hardware specifications using for example RTL (Register Transfer Level) level of modelling. However, when it concerns HW-SW co-design for tests and observation, system's behaviour and properties are specified in a single formal language such as in [7] and [8].

In this paper, we are not interesting by dependability analysis (fault simulation, estimation of optimal diagnostic strategies, etc,), but we focus on on co-modelling the appropriate behaviour of a system for diagnosis.

4. Multilevel of granularity co-modelling

Our objective is also to develop a relationship between the accuracy of the expected on-line fault detection and the level of granularity of software-hardware co-modelling (*Figure 3*).

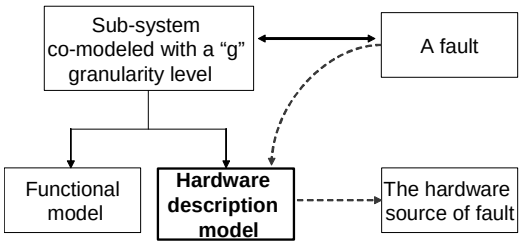

Figure 3. Relationship between a sub-system co-model and hardware source of fault

As a first step, we have to model the hardware architecture as a set of hardware sub-components, and then we have to model the software platform as a set of sub-functions allocating them to the modelled hardware sub-systems. As a second step, we schedule the priorities of the sub-functions, allocating the needed accuracy level of supervision to each priority. Then, we allocate a granularity level of co-modelling to every priority, in order to co-model every sub-function with a corresponding granularity level (*Figure 4*).

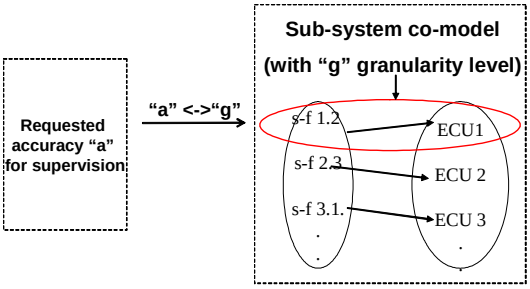

Figure 4. Granularity-accuracy relationship

5. Working environment

5.1 SystemC

SystemC is a language built in standard C++ by extending the language with the use of class libraries. SystemC addresses the need for a system design and verification language that spans hardware and software.

One of the most promising systemC advantages is hardware-software co-modelling to develop virtual platforms, because it supports a unified language of hardware-software modelling [1].

We have also selected SystemC as a working environment because it has many advantages:

- SystemC allows hierarchical modelling to express the multilevel of granularity modelling
- It allows HW/SW modelling with the same language
- The models could be easily connected to any other hardware models [13], or functional models (e.g. in Simulink) [12][14]
- SystemC environment includes also a simulator: it consists of a C++ library and an event-based motor for simulation
- Any C or C++ library can be included in a HW/SW co-model

Hence, we can describe the appropriate behaviour of the electronic embedded system with different levels of hierarchy. Thus, every sub-system that should be under supervision can be hierarchically co-modelled.

5.2 The TLM modelling

Transaction-level modelling (TLM) is a high-level approach to model digital systems where details of communication among modules are separated from the details of the implementation of functional units or of the communication architecture. Communication mechanisms such as busses or FIFOs are modelled as channels, and are presented to modules using SystemC interface classes [6].

Cycle Accurate TLM use case describes a category of levels of abstraction that could be cut into finer divisions. A Cycle Accurate (CA) model is a TLM model that represents the stage of communication refinement, in which communication is modelled accurately down to the level of bus cycles, and even clock cycles. CA modelling allows hardware verification, evaluating the state of every component in every cycle and running software device drivers. CA simulation speed varies between 10 and 100 KHz. A CA model consists of a set of processes that run once per cycle. This fits with the use of SC_METHOD processes and non-blocking calls.

In [4], Ghenassia, F. shows that TLM projects do not require a lot of effort and time to be correctly modelled compared to RTL projects. In fact, a Cycle Accurate project may need approximately half of the time compared to an RTL project for its realization.

For these advantages, we have used the CA level to co-model our HW/SW platform as shown in the next section.

6. Co-modelling using multilevel of granularity

We co-modelled with TLM, hierarchically, each hardware-software sub-system beginning by the highest level of granularity. Thus, on each level of granularity, we find a set of models representing at the same time the functional behaviour and the hardware architecture. Another interesting feature is in simulation; it is possible to switch between two or more levels of granularity according to the criticality level of eventual faults and the diagnosis system needs (e.g.: functions priorities).

6.1 ECUs and CAN bus modelling

The whole architecture consists of n ECUs communicating through the CAN network. In this part of the work, we have modelled the CAN protocol real-time behaviour to realize communications between ECUs models. We have simplified the details to ease the modelling; by implementing a virtual arbiter in the bus. With the Transaction modelling, the communication between components is described as function calls.

Each ECU is master and slave at the same time and has one bidirectional port in each module (*Figure 5*). It is used to send orders to the bus (Requests) and getting data and information from the bus (Responses). Each ECU that wants to send a message sends a request to the bus. If at least 2 ECUs request a bus transmission at the same time (i.e. in a time shorter than a bus cycle), the bus arbiter selects the most important message by comparing arbitration fields in the two messages.

Only one clock is used for all processors when the level of granularity is high and the accuracy of the model for simulation is set to the ECU clock cycle.

It is important to note that full CAN protocol is used only in models with high level of granularity, expressing transactions between ECUs. With a more accurate level of granularity, the processor and the memory models of every ECU are wrapped into SystemC modules in order to communicate with other devices such as CAN controllers. Anyway, the transactions are kept cycle accurate.

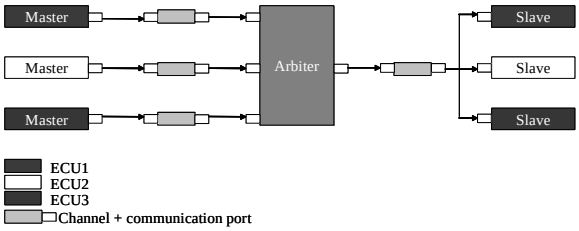

Figure 5. CAN bus TLM model

6.2 Interfacing hardware-software models

The communication between hardware and software modules should be done through messages exchange. This message exchange is using a shared interface between the two sides implementing the appropriate methods for messages exchange and eventually using blocking calls (*Figure 6*).

We have co-modelled in SystemC a sub-function of Smart Distance Keeping (SDK); a given function from a truck manufacturer. “SDK” is the name given to the “enhanced Adaptive Cruise Control (ACC) function”, mainly to distinguish a regulation only based on a fixed distance of 50m (ACC compliant to the regulations for heavy trucks) from a regulation that takes the relative vehicles dynamics into account. Thus, the SDK maintains a safe headway time, i.e the inter-vehicle distance is varying as a function of the velocity and is likely to be less than the legal 50 meters at intermediate speeds an embedded radar detects continuously the target [2].

*Figure **6*** shows an example of the embedded radar interface sending the acquired date to an ECU (the hardware module), which is responsible for extracting the current distance between the two vehicles from the received data, to compare it with the safety distance.

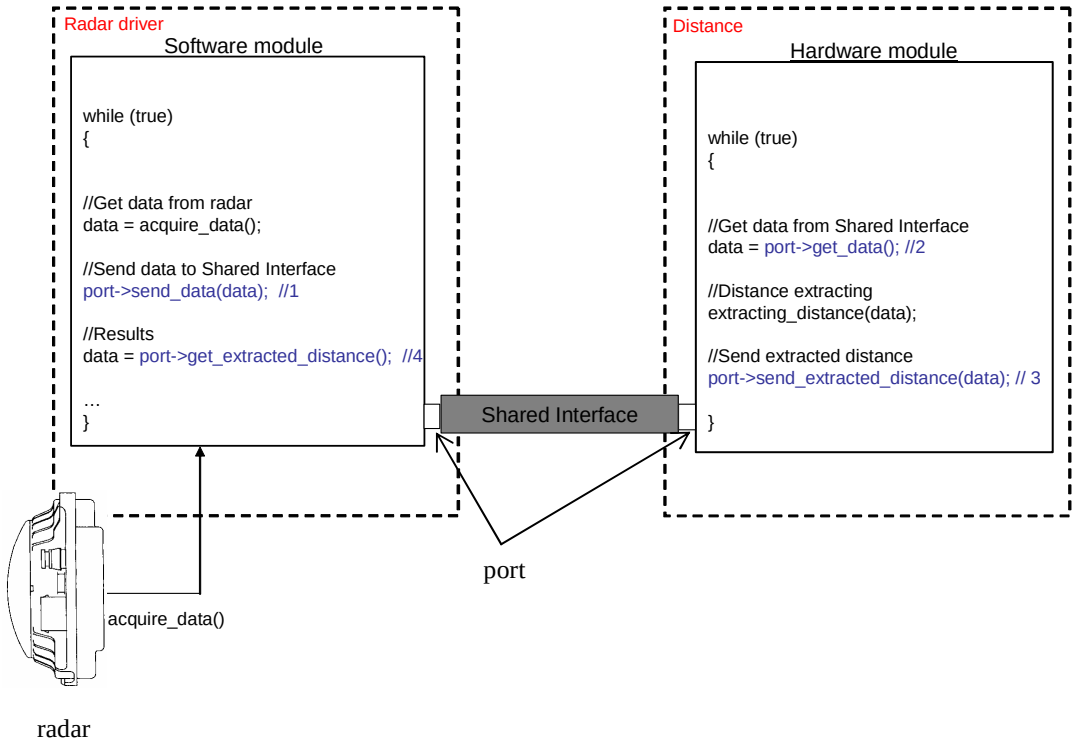

Figure 6. HW/SW interfacing

7. Test and validation

The presented modelling method allows the supervision designer to simulate and test the system off-line with various levels of accuracy without being intrusive, or to re-use the TLM models for real time embedded simulation for fault detection. If the supervision designer uses the same models for the on-board simulation for fault detection, the duration of fault detection based on the simulation of a model in a level "i" of granularity is calculated as follows (1):

With N is the number of components (level *i*)

$$\text{Fault detection duration (level } i\text{)} <= \sum_{j=1}^{N} \text{simulation duration (j)} \quad (1)$$

Thanks to the high-speed systemC simulation compared to RTL simulations, the supervisory system decreases the fault detection duration (see section 5).

We have tested our approach with Smart Distance Keeping system . The supervisory system can simulate the different SystemC co-models beginning by the highest level of hierarchy. In case of incoherence detection in a level of the hierarchy, SystemC brings the advantage of co-simulating the system with a more accurate level in order to increase the accuracy of incoherence detection and eventually localization. This possibility of different levels of simulation allows us to rapidly situate a path to the detected fault (*Figure 7*).

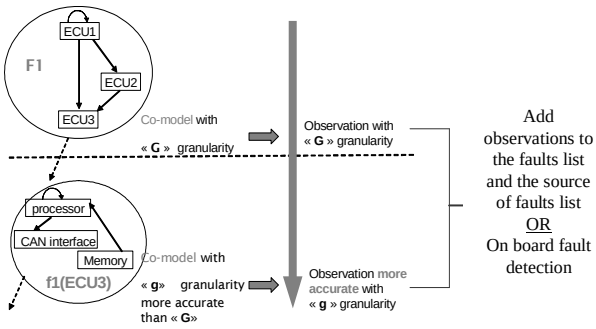

Figure 7. Multilevel of granularity co-simulation

It is important to note that this approach of modelling for diagnosis is simple to implement when the embedded functions are cyclic, as progressive and repetitive simulations may be done with real input values.

8. Conclusions and perspectives

We presented in this paper a co-modelling technique with multilevel of granularity under SystemC. Cycle accurate models of the CAN bus, CAN interfaces and ECUs including memories and processors has been co-modelled with different levels of granularity. We have shown the impact of a multi-granularity hardware-software model for the hardware real-time fault detection. This result shows that such modelling method is suitable to simulate distributed real time software-hardware architecture and allows the supervision designer to re-use the TLM models for embedded simulations for real time supervision or to simulate and test the system off-line without being intrusive.

As future work, we aim to extend the multilevel granularity modelling to other computing platforms, like SoC (System on Chip) for example; a technology that is gaining interest for car manufacturers. A SoC is a heterogeneous system, composed of software, hardware, requiring special development environments for specification, simulation, verification, compilation and execution. SoC are characterized by the re-use of different components (processors, memory, communication networks) connected on the same chip. The TLM is very suitable to model the architecture of a SoC, in order to enable development of the embedded software in advance of the hardware, and to carry out analyses early in the design cycle.

9. References

[1] Cai L. and Daniel G, 2003. Transaction level modelling: An overview. In Hardware/Software Co design and System Synthesis, *Report 03-10, Center for Embedded Computer Systems*, University of California, p. 466-471.

[2] Claeys X. and al, 2003. Chauffeur Assistant Functions. *Report restricted to RENAULT TRUCKS*, Contract number IST-1999-10048, Lyon, FRANCE.

[3] Gajski D-D. and al, 2000. SpecC: Specification Language and Methodlogy, *Kluwer Academic Publishers*.

[4] Ghenassia, F. (ed.), 2005. Transaction-Level Modelling with SystemC. TLM Concepts and Applications for Embedded Systems. *Springer*. ISBN 0-387-26232-6.

[5] Gioulekas, F. and al, 2005. Heterogeneous system level co-simulation for the design of telecommunication systems. *Journal of Systems Architecture. 51*, p. 688-705.

[6] Grötker T. and al, 2002. System Design with SystemC. *Springer*, Chapter 8, p. 131. ISBN 1402070721.

[7] Csertan. G, and al, 1994. Modelling of Fault-Tolerant Computing Systems. *In Proceedings of the 8th Symposium on Microcomputers and Applications, uP'94*, Budapest, Hungary, p.78-83.

[8] Csertan. G, and al, 1995. Dependability Analysis in HW-SW codesign. *In Proceedings of the IEEE International Computer Performance and Dependability Symposium, IPDS'95*, Erlangen, Germany, p. 316-325.

[9] Hamscher W. and al, 1992. Readings in model-based diagnosis. *Morgan Kaufmann*, isbn: 1-55860-249-6, San Francisco, CA, USA.

[10] Paret D, 1999. Le bus CAN description : de la théorie à la pratique. *Dunod* , ISBN-10: 2100047647, ISBN-13: 978-2100047642.

[11] Wong, S.Y, 1998. Hardware/software co-design language compatible with VHDL. *IEEE, WESCON/98*, Anaheim, CA, USA, p. 78 -83.

[12] Boland J-F and al. 2004. Using Matlab and Simulink in a SystemC verification Environment. *2nd North American SystemC User's Group*. Santa Clara, CA, USA

[13] Bombana, M.; Bruschi, F. 2003. SystemC-VHDL co-simulation and synthesis in the HW domain. *Design, Automation and Test in Europe Conference and Exhibition*, pp. 101-105, Messe Munich, Germany.

[14] Czerner F. And Zellmann J. 2002. Modelling Cycle-Accurate Hardware with Matlab/Simulink using SystemC. *6th European SystemC Users Group Meeting (ESCUG)*. Stresa, Ilalia.

[15] Paret D. 2007. Multiplexed Networks for Embedded Systems: CAN, LIN, FlexRay, Safe-by-Wire, *Wiley*, ISBN: 978-0-470-03416-3