Augmenting High-Level Petri Nets to Support GALS Distributed Embedded Systems Specification

Filipe Moutinho, Luís Gomes

Augmenting High-level Petri Nets to Support GALS Distributed Embedded Systems Specification

Filipe Moutinho and Luís Gomes

Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Portugal
UNINOVA – CTS, Portugal
fcm@uninova.pt, lugo@fct.unl.pt

Abstract. High-level Petri net classes are suited to specify concurrent processes with emphasis both in control and data processing, making them appropriate to specify distributed embedded systems (DES). Embedded systems components are usually synchronous, which means that DES can be seen as Globally-Asynchronous Locally-Synchronous (GALS) systems. This paper proposes to include in high-level Petri nets a set of concepts already introduced for low-level Petri nets allowing the specification of GALS systems, namely time domains, test arcs and priorities. Additionally, this paper proposes external messages and three types of (high-level) asynchronous communication channels, to specify the interaction between distributed components based on message exchange. With these extensions, GALS-DES can be specified using high-level Petri nets. The resulting models include the specification of each component with well-defined boundaries and interface, and also the explicit specification of the asynchronous interaction between components. These models will be used not only to specify the system behavior, but also to be the input for model-checking tools (supporting its verification) and automatic code generation tools (supporting its implementation in software and hardware platforms), giving a contribution to the model-based development approach and hardware-software co-design of DES based on high-level Petri nets.

Keywords: Distributed embedded systems, GALS systems, model-based development, high-level Petri nets, asynchronous-channels.

1 Introduction

This work gives a contribution to the use of high-level Petri nets in the model-based development approach of distributed embedded systems (DES). Model-based development approaches [1, 2, 3, 4] and hardware-software co-design techniques [5] have been providing several development methods for embedded systems. Often supported by tools, these methods can provide benefits such as the reduction in the development time and in the number of development errors (bugs). DES are composed by a set of embedded systems (components) in interaction, which may be geographically distributed or not (implemented in a single implementation platform or chip). When these components, which may be hardware or software components, are synchronous with a specific clock tick (or clock signal), the DES is globally-asynchronous locally-synchronous (GALS) [6].

Several modeling formalisms, such as State-Diagrams, StateCharts [7] and Petri nets [8], have been used in model-based development approaches to develop embedded systems. Given that Petri nets are appropriate to specify concurrency, they were chosen in this work as the modeling formalism for DES. Low-level Petri net classes are appropriate to specify "control dominated" systems, whereas high-level Petri net classes are appropriate to specify not only control but also "data processing".

Low-level Petri nets were extended and used in [9, 10, 11] to specify GALS systems. Additionally, several works (such as in [12, 13, 14, 15]) addressed the development of distributed systems using low-level Petri nets. But no works are known, where globally-asynchronous locally-synchronous distributed embedded systems (GALS-DES) are explicitly specified through high-level Petri nets.

The work presented in this paper is integrated in a PhD work, and contributes to answer the following research question: *How to extend high-level Petri net classes to allow the explicit specification of GALS-DES?* An explicit specification should include the specification of each component, its interface, and the specification of the asynchronous interaction between components.

This paper presents an extension to high-level Petri nets, which contributes both to the model-based development and to the hardware-software co-design of distributed embedded systems that are also globally-asynchronous locally-synchronous (GALS-DES). The extended high-level Petri net models are intended to support not only the system specification, but also to be used as input in model checking and automatic code generators tools for hardware and software platforms.

Section 2 mentions how this paper contributes to the Internet of Things, section 3 presents the proposed extension for high-level Petri nets, and finally in section 4 conclusions and future work are presented. Due to space limitations it was not possible to present any complete application example in this paper.

2 Relationship to Internet of Things

The spread of embedded systems by devices, machines and infrastructures around us, and the possibility to interconnect via Internet or mobile connections, enabled the creation of many types of distributed embedded systems. The development of distributed systems when compared with the development of centralized systems is a challenging task, due to the interaction of concurrent components and also due to the large size/complexity of the overall system. This paper aims to contribute to the development of such systems using a model-based development approach and high-level Petri net classes, which are suitable to specify concurrent controllers with control and data processing capabilities. High-level Petri nets were augmented in this work to allow the specification of distributed embedded systems, and are also intended to be the input for verification and automatic code generation tools.

3 Extending High-level Petri Nets for GALS-DES

High-level Petri nets were defined in the international standards ISO/IEC 15909-1 and ISO/IEC 15909-2 [16]. Briefly presented in Fig. 1, high-level Petri nets (*HLCoreStructure*) are an extension of low-level Petri nets (*PNMLCoreModel*). It is important to note that the international standard does not define a concrete syntax for high-level Petri nets, and that in this paper is used a concrete syntax similar to the one used in Colored Petri nets [17].

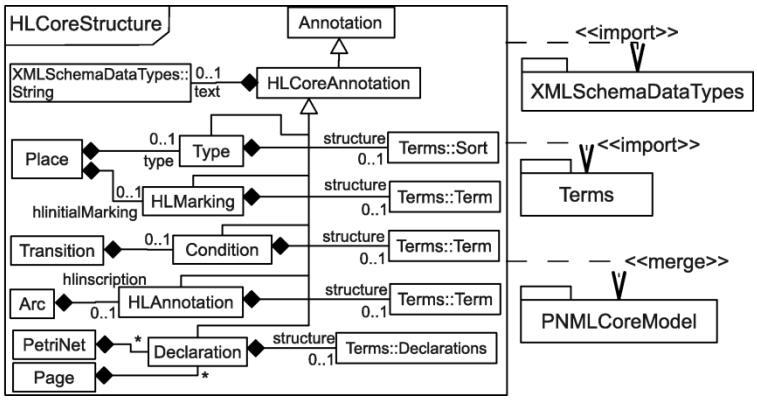

Fig. 1. The UML class diagram of the *HLCoreStructure* package, image adapted from [16].

This paper extends high-level Petri nets with a set of concepts to support the development of GALS-DES with emphasis both on control and data processing. The proposed concepts are presented in Fig. 2, which is represented by an UML class diagram and by a set of constraints expressed in the Object Constraint Language (OCL). Some of the concepts (test arcs, priorities, and time domains) have already been proposed for low-level Petri nets in [18, 11], whereas the other concepts (external messages and three types of asynchronous channels) are new.

Test arcs and priorities proposed here for high-level Petri nets, were proposed in [18] to support conflict arbiters. Test arcs (also known as read arcs) do not remove tokens from places, and are represented in Figs. 3, 4, and 5, by arcs with an arrow in the middle. When two or more transitions are in conflict, the one with higher priority (lower value) will fire. Priorities are represented in Fig. 5, by *p:1* and *p:2*.

3.1 Execution Semantics and Time-domains

The concept of time-domains was proposed in [11] to extend the IOPT-net class [18], which is a low-level Petri net class. This class without time-domains has (globally) synchronous and maximal-step execution semantics, which means that transitions can only fire at specific time instants (given by a clock tick) and that all transitions that are enable and ready (see [18]) to fire (and also not in conflict) at a specific clock tick,

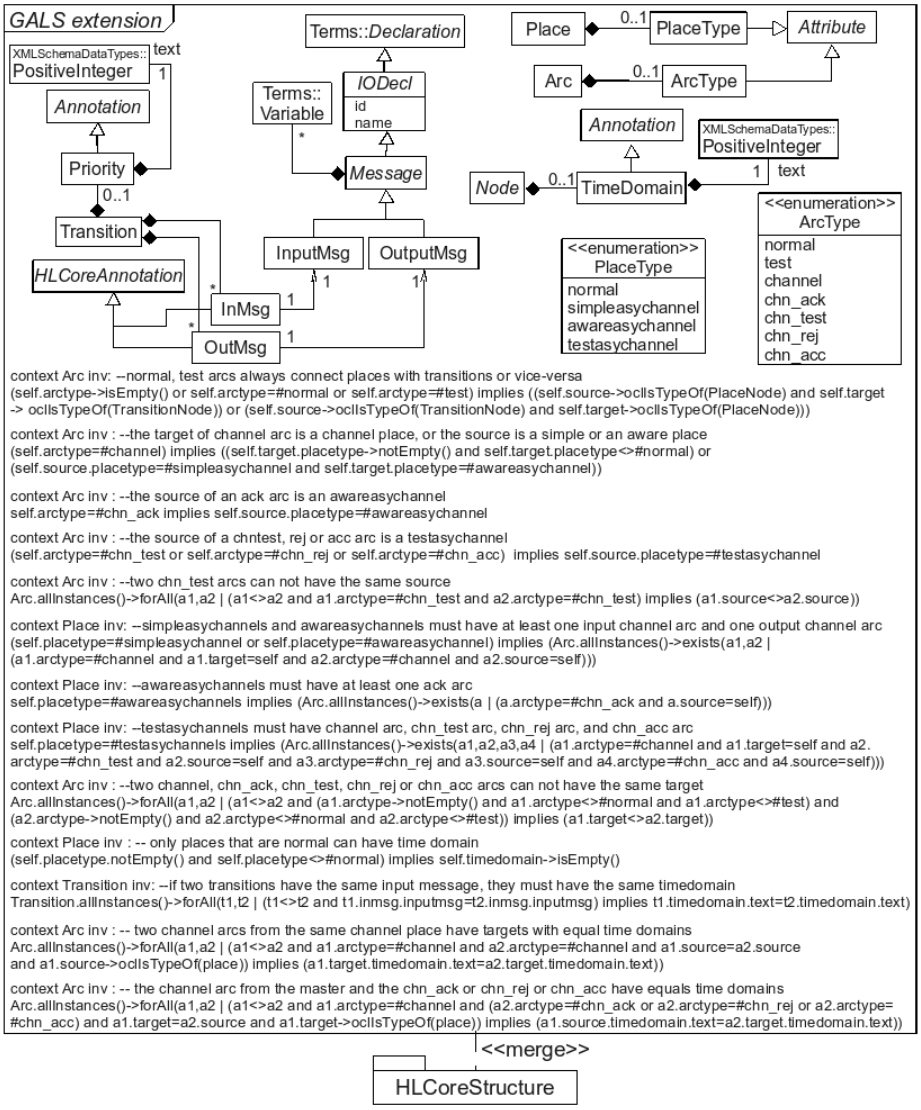

Fig. 2. The proposed *GALS extension* package.

will fire simultaneously. The IOPT-net class extended with time-domains allows the specification of globally-asynchronous locally-synchronous systems, because all transitions with a specific time-domain are “locally synchronous” and have a “locally maximal-step” execution semantics. The concept of time-domain associates each Petri net node (transition or place) with a specific component, which is synchronous with a specific clock tick. Transitions with different time-domains belong to different components (synchronous with distinct clock signals).

The concept of time-domain when added into high-level Petri net classes, introduces the globally-asynchronous locally-synchronous execution semantics into high-level Petri nets, allowing the specification locally synchronous components (synchronous with specific clock signals). Fig. 2 presents the concept of time-domain added into high-level Petri nets, where Petri net nodes (transitions and places) can have an associated time-domain (represented as a node annotation). An OCL specifies that if two transitions have the same input message, they must have equal time-domains, because an input message cannot be an input of several components. Fig. 3 presents a high-level Petri net model with two components, each one specified by a specific time-domain (represented by *td:1* and *td:2*).

3.2 Asynchronous-channels

Time-domains are used to identify components, whereas asynchronous-channels enable their interaction specification (the exchange of data between components). Three types of directed asynchronous communication channels are proposed in this paper: (1) Simple-Asynchronous-Channel (SAC); (2) Aware-Asynchronous-Channel (AAC); and (3) Test-Asynchronous-Channel (TAC). To ensure that messages are delivered by the same order that are sent, asynchronous channels have FIFO (First In, First Out) semantics. All the proposed channels assume that all sent messages eventually arrive (sometime in the future) at the destination component.

The models at the right hand side of Figs. 3, 4, and 5 present the semantics of each channel and the models at the left hand side present their representation (clouds connected to transitions by dashed arcs). Although it is only presented for SAC (in the right model from Fig. 3), the behavioural model of the other two channels (right models from Figs. 4 and 5) also have a *reset* transition to prevent the messages identifier to grow indefinitely. Due to space restrictions the channels proposed in Fig. 2 will be briefly described.

Each SAC sends messages from one (master) transition to a set of (slave) transitions. All slaves have the same time-domain (belong to the same component), which is different from master time-domain. Fig. 3 (left) presents a high-level Petri net model with a SAC, and its behavior is presented in Fig. 3 (right) model.

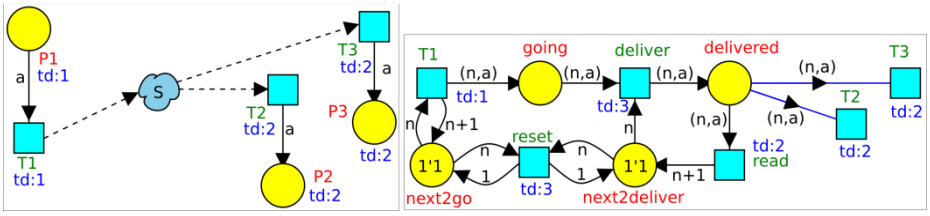

Fig. 3. A high-level Petri net model with a Simple-Asynchronous-Channel (at the left) and its behavioral specification using a high-level Petri net model (at the right).

In an AAC, messages are sent as in a SAC, but an acknowledge is sent when the message is read by destination component (even if the message is ignored). Fig. 4 (left) presents a model with an AAC, and its behavior is presented in Fig. 4 (right).

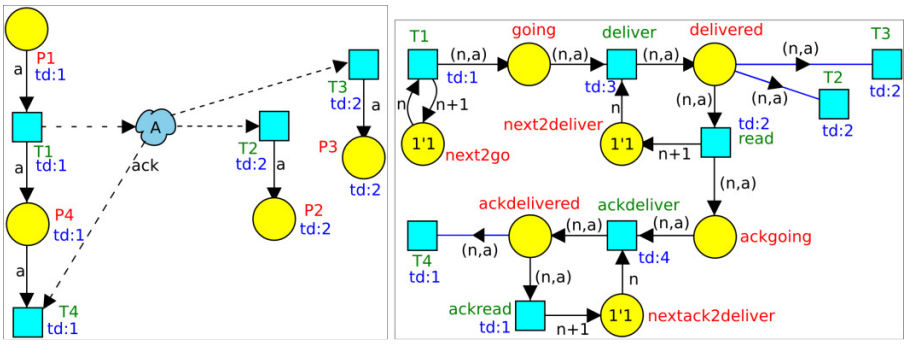

Fig. 4. A high-level Petri net model with an Aware-Asynchronous-Channel (at the left) and its behavioral specification using a high-level Petri net model (at the right).

Finally, in a TAC messages are sent as in a SAC, but when the message is read by destination component, an accepted (*acc*) message (if the message fired the transition) or a rejected (*rej*) message (if the message is ignored) is sent back. Fig. 5 (left) presents a model with a TAC, and the TAC behavior is presented in Fig. 5 (right) model. Each SAC or AAC can have several slave transitions, whereas each TAC can only have one slave transition.

Any Petri net model with these three types of communication channels can be specified by a Petri net model without channels, replacing the channels by their behavioral models (right models from Figs. 3, 4 and 5).

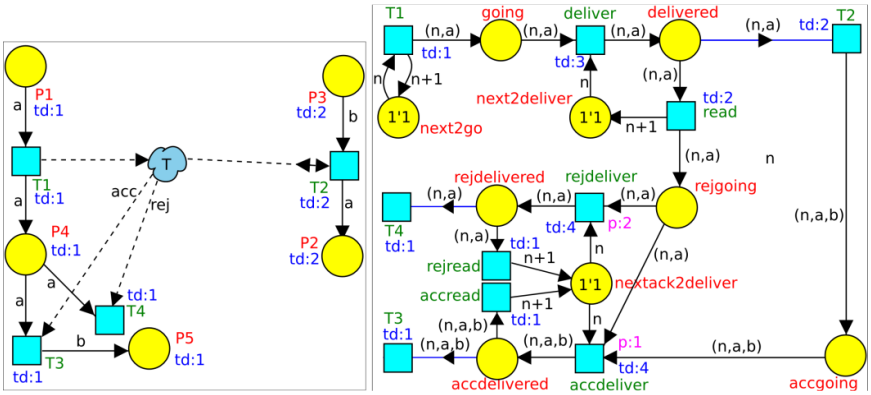

Fig. 5. A high-level Petri net model with a Test-Asynchronous-Channel (at the left) and its behavioral specification using a high-level Petri net model (at the right).

The GALS-DES specification using high-level Petri net classes extended with time-domains and asynchronous-channels have a strong mathematical definition (not presented in this paper) and a well defined execution semantics, allowing its use as input for model-checking tools, to verify the extended high-level Petri net models proprieties.

3.3 Messages

Input and output messages are proposed to specify the components interface (to allow the specification of the interaction between the components and the environment or between the components and the communication channels). After GALS-DES specification and verification, and before components implementation, asynchronous channels must be removed, and input and output messages must be inserted into Petri nets. Inserted input and output messages are associated with master and slave transitions (previously connected with asynchronous channels). The resulting models with input and output messages can be used as input for automatic code generator tools for software and hardware platforms.

Input and output messages are proposed in this paper as presented in Fig. 2. *Messages* are *Petri nets* or *Pages* declarations, and are associated with transitions. Both input and output messages can have associated *Variables*, which defines the carried data. When an input message occurs, the associated transitions fire if enable and ready (the carried data influence transition bindings). Output messages are generated when associated transitions fire. An output message can carry data obtained from place marking.

4 Conclusions and Future Work

This extension gives a contribution to the use of high-level Petri nets in the model-based development approach of distributed embedded systems. The identification of components sub-models is made through the use of time-domains; the interaction between components is specified by asynchronous communication channels; and external messages (input and output messages) are the components interface.

Some of the concepts (such as time-domains) proposed in this paper for high-level Petri nets had already been proposed in the past for low-level Petri nets, whereas other concepts (such as the three types of asynchronous communication channels with FIFO semantics and the external messages) as far as we know, are new.

During our experience using these communication channels, it was possible to specify all encountered communication scenarios between components of globally-asynchronous locally-synchronous distributed embedded systems.

As future work we intend to extend the tool chain framework for a low-level Petri net class (online available at *http://gres.uninova.pt/)*, to allow the edition, the model-checking, and the automatic code generation of distributed embedded systems using high-level Petri nets.

Acknowledgments. This work was partially financed by Portuguese Agency "FCT - Fundação para a Ciência e a Tecnologia" in the framework of project PEst-OE/EEI/UI0066/2011. The first author was supported by a FCT grant, ref. SFRH/BD/62171/2009.

References

1. Schatz, B., Pretschner, A., Huber, F., Philipps, J.: Model-based development of embedded systems. In: Bruel, J.-M., Bellahsène, Z. (eds.) Advances in Object-Oriented Information Systems, LNCS, vol. 2426, p.298. Springer, Heidelberg (2002)
2. Rust, C., Kleinjohann, B.: Modeling Intelligent Embedded Real-Time Systems using High-Level Petri Nets. In: Proceedings of the forum on design languages FDL. (2001)
3. De Niz,D., Bhatia, G., Rajkumar, R.: Model-Based Development of Embedded Systems: The SysWeaver Approach. In: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium, Washington, DC, USA (2006)
4. Gomes, L., Fernandes, J. (eds.): Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation. In: IGI Global's. ISBN 978-1-60566-750-8 (2009)
5. Wolf, W.H.: Hardware-software co-design of embedded systems [and prolog]. In: Proceedings of the IEEE, vol. 82, no. 7, pp. 967-989 (1994)
6. Chapiro, D. M.: Globally-Asynchronous Locally-Synchronous Systems. Ph.D. Thesis: Stanford University (1984)
7. Harel, D.: Biting the silver bullet: toward a brighter future for system development, Computer , vol.25, no.1, pp. 8-20 (1992)
8. Reisig, W.: Petri nets: an introduction, Springer-Verlag New York, Inc., NY, USA (1985)
9. Nielsen, M., Sassone, V., Srba, J.: Towards a Notion of Distributed Time for Petri Nets. In Proceedings of the 22nd International Conference on Application and Theory of Petri Nets (ICATPN '01), José Manuel Colom and Maciej Koutny (eds.). Springer-Verlag, London, UK, 23-31 (2001)
10. Kleijn, H., Koutny, M., Rozenberg, G.: Processes of Petri nets with localities, Technical Report CS-TR-941, School of Computing Science, Newcastle upon Tyne, UK, (2006)
11. Moutinho, F., Gomes, L.: Asynchronous-channels and time-domains extending Petri nets for GALS systems. In Camarinha-Matos, L.M., Shahamatnia, E., Nunes, G. (eds) Technological Innovation for Value Creation. IFIP AICT, vol. 372, pp. 143–150. Springer Berlin, Heidelberg (2012)
12. Hopkins, R.: Distributable nets. In Rozenberg, G. (ed.) Advances in Petri Nets, ser. LNCS, vol. 524, pp. 161–187. Springer Berlin, Heidelberg (1991)
13. Badouel, E., Caillaud, B., Darondeau P.: Distributing finite automata through Petri net synthesis. Formal Asp. Comput., vol. 13, no. 6, pp. 447–470 (2002)
14. Van Glabbeek, R., Goltz, U., Schicke, J.-W.: On synchronous and asynchronous interaction in distributed systems. In: CoRR, vol. abs/0901.0048 (2009)
15. Van Glabbeek, R., Goltz, U., Schicke-Uffmann, J.-W.: On distributability of Petri nets. In: Birkedal, L. (ed.) Foundations of Software Science and Computational Structures, ser. LNCS, vol. 7213, pp. 331–345. Springer Berlin, Heidelberg (2012)
16. Hillah, L., Kindler, E., Kordon, F., Petrucci, L., Treves, N.: A primer on the Petri Net Markup Language and ISO/IEC 15909-2. In: Petri Net Newsletter, vol. 24, no. 76, pp. 9–28 (2009) (originally presented at the 10th International workshop on Practical Use of Colored Petri Nets and the CPN Tools – CPN'09)
17. Jensen, K., Kristensen, L.M., Wells., L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. In: International Journal on Software Tools for Technology Transfer (STTT)9(3-4), pp. 213-254 (2007)
18. Gomes, L., Barros, J., Costa, A., Nunes, R.: The Input-Output Place-Transition Petri Net Class and Associated Tools. In: Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN'07), Vienna, Austria (2007)