Asynchronous-Channels and Time-Domains Extending Petri Nets for GALS Systems

Filipe Moutinho and Luís Gomes

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

Abstract. A specific class of Petri nets was extended with Asynchronous-Channels (ACs) and Time-Domains (TDs) to support Globally-Asynchronous Locally-Synchronous (GALS) systems' modeling, analysis and implementation. This non-autonomous class of Petri nets is targeted to support the development of automation and embedded systems using a model-based development approach. It benefits from a tool chain framework previously developed, covering the whole development flow, from specification to hardware and software deployment. With the extended Petri net class is possible to model GALS systems, and use the specification to generate the corresponding state space supporting the behavior verification and providing valuable information for implementation.

Keywords: GALS embedded systems, Model-based development, Petri nets.

1 Introduction

With the increase in the number of requirements, embedded systems are becoming larger and more complex. Synchronous specifications are widely used in hardware and software systems development due to simplicity in the verification and synthesis processes. Using software platforms it is common not to reach the desired processing performance, requiring a full or a partially hardware implementation. In hardware, large synchronous designs with the need for high clock frequencies are complex to develop. This can occur because it is difficult to make a proper clock tree distribution, and the signal propagation time may be higher than the clock period. High power consumption and Electromagnetic Interference (EMI) are also common problems of large synchronous circuits, that can be minimized with the use multiple synchronous components. In software, multiple components also enables the number of clocks (processor clock ticks per second) reduction and as a consequence power consumption reduction. Distributed embedded systems are a possible solution for complex embedded system; also allowing the reuse of old previously designed components.

Globally-Asynchronous Locally-Synchronous (GALS) systems proposed in [1] are intrinsically distributed systems and combine advantages of synchronous systems with asynchronous systems. Synchronous systems are easier to develop and rely on a set of available tools. On the other hand, asynchronous systems are faster, with lower power consumption and higher performance. In GALS systems, each local component is synchronous with a local clock tick, which determines its evolution; as each component has a different clock domain, the global system is asynchronous. Interaction can occur through asynchronous *wrappers*, such as those proposed in [2].

Petri net classes have been proposed by several authors to develop embedded systems through a model-based development approach. The Input-Output Place-Transition (IOPT) Petri net [3] is one of those classes, with an available tool framework allowing: (1) models edition; (2) models partition [4] (producing a set of synchronous sub-models interconnected through synchronous communication channels and supporting the application of hardware-software co-design techniques in embedded systems design); (3) automatic generation of the state space for properties verification; (4) automatic generation of C and VHDL codes for implementing system controllers; (5) the generation of Graphical User Interfaces.

However, since we need to face distributed implementation and to accommodate different time domains associated with the components of the GALS system, it is necessary to handle asynchronous communication between components, where specific asynchronous *wrappers* can be used to assure robust communication. As the IOPT net class does not allow GALS systems specification, the following research question appear: *How to specify GALS systems using the IOPT net class, in order to verify GALS systems properties, to support behavior verification and to obtain the required information for components and asynchronous wrappers implementation?*

This paper presents an extension to the IOPT net class, introducing Asynchronous-Channels (ACs) and Time-Domains (TDs), making possible the specification of GALS systems through the extended IOPT net class. From this specification it is possible to generate the associated state space. Properties verification through the state space will help to determine if the models specify the desired behavior and to obtain required information to implement components and asynchronous *wrappers.*

2 Contribution to Value Creation

Using a model-based development approach to embedded systems, together with its implementation as a GALS system, enables the design and implementation of more complex systems, better documented, in less time, in a more automatic way, and benefiting from reusability of models and code. In this sense, the model-based development approach and this work in particular contribute with added value for the system development. In addition, the system when implemented as a GALS system, instead of being implemented as a global synchronous system, might have less EMI and power consumption. To develop reliable systems is required to guarantee the proper behavior of the embedded system, where this work gives an important contribution, extending the IOPT net class with the ability of specifying GALS systems, supporting its documentation, verification and implementation.

3 Related Work

GALS embedded systems development presents greater challenges when compared to synchronous embedded systems development, making the development method even more crucial in the final system quality, time-to-market, reusability, etc. Model-based development approaches proposed by several authors (such as in [5, 6, 7, 8, 9]) in the recent years, for embedded systems development, may be an appropriate approach in the development of GALS systems.

Some authors, like in [10], proposed textual languages for GALS systems specification and verification, while others (such as in [11, 12]) used graphical-based descriptions. In [11], the Place/Transition net class (P/T nets, an autonomous Petri net class) [13] is extended with localities. It is used to model and make the behavioral analysis of GALS systems. Localities are assigned to transitions, making them part of specific components of the GALS system.

The IOPT net class [3] extended with ACs and TDs is considered in this work to support the complete development flow of GALS systems, and not only system specification and verification, like in [11]. The IOPT net class was chosen based on its characteristics that make it suitable for modeling automation and embedded systems. It benefits from availability of a tool chain framework, used in this work to support model edition, partitioning, properties verification and automatic generation of C and VHDL codes for implementing GALS system components. In [14], the IOPT net class (not extended) was used to specify GALS systems, where a set of sub-models was used to specify a set of components, and the interaction between components was modeled through single places. The use of IOPT nets as was done in [14] has two limitations: (1) it is not possible to use two (separate) sub-models to specify a single component; and (2) single places between components do not allow the specification of asynchronous communication between components, as the maximal step execution within each component, separately, is not assured.

4 The IOPT Petri Net Class

The IOPT net [3] is a class of Petri nets that extends the well known P/T net class [13] with inputs, outputs and a set of additional characteristics. Inputs are used to model the interaction between the environment and the system (making this class non-autonomous); outputs are used to represent system actions in the environment. IOPT nets have synchronous execution (the system evolution takes place at specific instants of time controlled by a clock tick) and a maximal-step executable semantics, which means that all enabled and ready transitions at a specific instant of time will fire. A transition is enabled when the number of tokens in places from incoming arcs are equal or bigger than the weight of the corresponding arc connecting the place to the transition. A transition is ready when its guard is true and all input events occur.

In order to benefit from Model Driven Architecture (MDA – an initiative from Object Management Group) artifacts and infrastructure, an IOPT Ecore representing IOPT models was proposed in [15].

A distributed embedded system with (two) components in interaction can be specified through an IOPT net model. Fig. 1 (at the left and at the center) presents two distinct ways to do it. But in both is not possible to specify components with distinct time domains, disabling GALS systems specifications. In addition, the synchronous channel (see [4]) of the left model considers a zero time delay between *T1* and *T2* firing, making it unsuitable to specify GALS components interaction. Furthermore, using a specification through events, like in the center model, the output event *#Z* and the input event *#A* should be related, but in IOPT net it is not possible to do it.

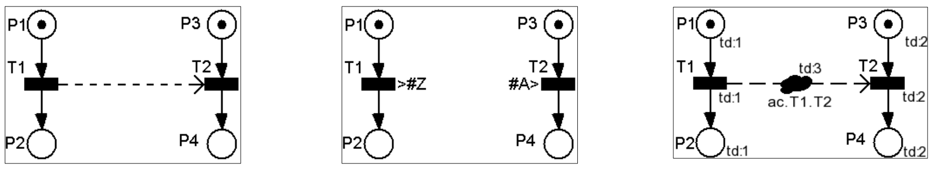

Fig. 1. A Petri net with a synchronous channel (left), a Petri net with a two components interacting through events (center), and a GALS system model using AC and TDs (right).

5 ACs and TDs Extending the IOPT Net Class

Introducing a new annotation attribute referring the Time-Domain (TD) of each node of the IOPT net (places and transitions) it is possible to associate each node to a specific component. In addition, replacing in left model of Fig. 1 the synchronous channel, or in center model of Fig. 1 the communication events, by an Asynchronous-Channel (AC), the right model of Fig. 1 is obtained. Each AC (represented by a dashed arrow with a cloud in the middle) has a specific TD.

All nodes of an IOPT net model, directly or indirectly connected through arcs to a transition of a specific component, must belong to the same component of the transition. In the right model of Fig. 1, nodes *P1*, *T1* and *P2* belong to component one with TD *1* (*td:1*), nodes *P3*, *T2* and *P4* belong to component two with TD *2* (*td:2*), and the AC named *ac.T1.T2* has TD *3* (*td:3*).

5.1 Definition

An AC always connects two transitions with two different TDs. One transition is the master and sends events to the other transition (the slave), events pass through the AC. In right model of Fig. 1, *T1* is the master transition and belongs to component one with TD *1* (*td:1*), *T2* is the slave transition and belongs to component two with TD *2* (*td:2*). An IOPT Petri net extended with ACs and TDs can be defined by

$$IOPT2GALS = (IOPT, ACs, TDs) \ , \tag{1}$$

where: (1) an IOPT Petri net is defined as in [3]; (2) ACs are a set of Asynchronous-Channels; and (3) TDs are a set of time domains.

$$IOPT = (P, T, A, TA, M, weight, weightTest, priority, isg, ie, oe, osc) \ . \tag{2}$$

$$ACs \subseteq (T \times T) \ . \tag{3}$$

$$AC \subseteq (t_m \times t_s) \ . \tag{4}$$

$$TDs = TDs_p \cup TDs_t \cup TDS_{ac} \ . \tag{5}$$

t_m is the master and t_s is the slave, such that $(t_m \in T) \wedge (t_s \in T) \wedge (t_m \neq t_s)$. $TDs_p : P \to IN$, $TDs_t : T \to IN$, and $TDs_{ac} : ACs \to IN$.

The IOPT Ecore proposed in [15] was extended in order to include ACs and TDs. Fig. 2 presents the new package extending the IOPT Ecore. Two annotations were inserted: (1) the *AsynchronousChannel* and the *TimeDomain*. An IOPT net *Page* can have one or more *AsynchronousChannels*. An *AsynchronousChannel* has a *TimeDomain* and links one master transition to one slave transition. Master and slave transitions must belong to different components (with different time domains). When modeling GALS systems, IOPT net *Nodes* belong to specific GALS components (identified by its time domain).

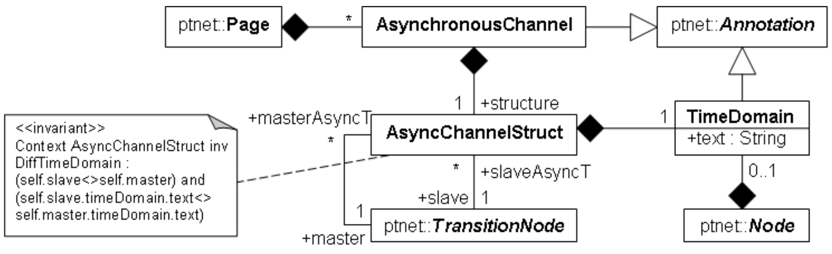

Fig. 2. Asynchronous-Channels package extending IOPT net Ecore.

5.2 AC Executable Semantics

Considering Fig. 1 (right), each time the master transition fires, an event is sent to the slave transition through the AC. The time spent between master and slave transition (always different from zero, contrary to what happens in the synchronous channels) depends on the AC TD. The proposed executable semantics considers that the slave component consumes the received events in the next execution tick.

The executable semantics of ACs can be described using IOPT nets, in two distinct (and equivalent) ways (Fig. 3): in the left model, using synchronous channels [4] (represented by dashed arrows between transitions); or in the right model using a test arc (represented by a line with an arrow in the middle) between a place and a transition (also known as read arc). In both models: (1) each time master transition fires tokens are inserted in *P5*; (2) transition *T4* models asynchronous nature of the channel, with the specific TD of the AC (*td:3*), it consumes tokens from P5 and insert tokens in *P6*; (3) in the next clock tick cycle of component two with TD *2* (*td:2*), tokens are removed from *P6* through *T5* (left) or *T3* (right) and *T2* if enabled, fires. Using one of the models of Fig. 3 to describe the behavior of ACs, it will be possible to analyze the generated state space, getting through the maximal bound of AC places (*P5* or *P6*), the buffer length of the communication channels implementing the ACs. This information is very important for a robust implementation of the whole system.

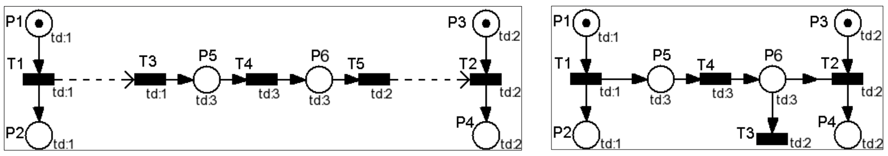

Fig. 3. AC model using synchronous channels (left) and AC model using a test arc (right).

6 Validation

An IOPT net editor supporting ACs and TDs was developed as a textual editor automatically generated from the extended Ecore in Eclipse Modeling Framework (EMF). This editor guarantees well-formed models in fully concordance with the IOPT Ecore metamodel. A set of examples was used to validate the proposed ACs and TDs. Due to space limitations, in this paper is presented a very simple one, modeling a manufacturing system with one machine and two conveyor belts. Each conveyor belt feeds the machine with one type of components, two components are needed to build a piece. Output signals *#M1* and *#M2* make the conveyor belts move. Two sensors (input events *#S1* and *#S2*) detect components arriving. After components arrive, output event *#Build* is generated by the system controller, putting the machine to work. Input event *#Done* indicates the end of machine building process.

The system was first specified through a (centralized) IOPT net. After the model edition and translation into Petri Net Markup Language (PNML) format, the model from the manufacturing system, was divided into three sub-models using the net split tool [4], in order to implement the distributed controller with three components: (1) component *C1* controlling the machine, (2) component *C2* controlling one conveyor belt, and (3) component *C3* controlling the second conveyor belt.

In order to move away from synchronous paradigm, and include different time domains for the generated components, synchronous channels were replaced by the proposed ACs and each IOPT Petri net node was associated with one TD (one component of the GALS system). The GALS system model of the distributed manufacturing system is presented in Fig. 4.

The generated PNML [15] was used to feed the state space generator tool for GALS systems based on the algorithm proposed in [16], which generates state spaces from IOPT models of GALS systems, allowing property verification of the behavior of the global GALS system (including each component behavior and its interaction).

From the state space and performing queries, was verified that the system has the desired properties: no deadlocks; the machine build a new piece when both components are available; etc. It was also verified that the maximal bound of all places of the IOPT net is one, which means that the length of implementation registers and *wrapper* buffers is equal to one. Due to space limitation is not possible to present the generated state space and the performed queries. VHDL code for hardware and C code for software implementations were automatically generated from the PNML file to implement each GALS system component, using the tools [17, 18].

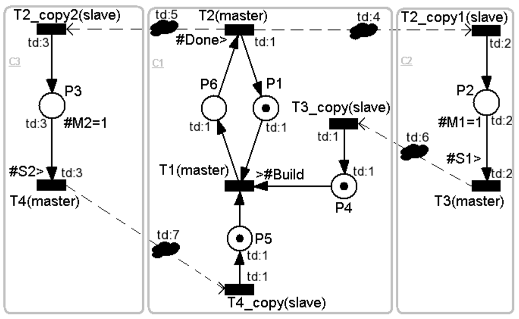

Fig. 4. The IOPT Petri net model with ACs and TDs modeling a GALS system.

7 Conclusions and Future Work

With the proposed ACs and TDs extending IOPT nets it is possible to specify GALS systems behavior. This class is used in a model-based development approach to verify GALS systems properties, supporting behavior verification and implementation.

The proposed extension was validated with several examples, where GALS systems were initially modeled: (1) with a set of models specifying a set of components, interacting through events; or (2) with one centralized model and then partitioned using the net splitting operation, and interacting through synchronous channels. In both approaches models rely on a synchronous paradigm, which means that all components have to be synchronous within the same clock domain. The TDs and ACs proposed in this paper allowed the development of distributed implementations with components at different clock domains, and its interaction.

The new tool used to generate the global state space of GALS systems modeled through extended IOPT nets (with ACs and TDs) will be publicly available in the near future. The generated state space allows properties verification of the entire system (as if it is a single synchronous system). The tool will have a comprehensive interface allowing queries on the state space, and will be integrated in the tool chain framework currently under development, including a new graphical editor supporting ACs and TDs edition.

Acknowledgment. The first author work is supported by a Portuguese FCT (Fundação para a Ciência e a Tecnologia) grant, ref. SFRH/BD/62171/2009.

References

1. Chapiro, D.M.: Globally-Asynchronous Locally-Synchronous Systems, Ph.D. Thesis: Stanford University, (1984)
2. Bormann, D. S., Cheung, P.Y.K.: Asynchronous wrapper for heterogeneous systems, International Conference on Computer Design (ICCD), (1997)
3. 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)
4. Costa, A., Gomes, L.: Petri net partitioning using net splitting operation, in Proceedings of the 7th IEEE International Conference on Industrial Informatics, Cardiff, UK, (2009)
5. Schatz, B., Pretschner, A., Huber, F., Philipps, J.: Model-based development of embedded systems, in Advances in Object-Oriented Information Systems, Springer, France, (2002)
6. 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)
7. Borcsok, J., Chaaban, W., Schwarz, M., Sheng, H., Sheleh, O., Batchuluun, B.: An automated software verification tool for model-based development of embedded systems with Simulink, in XXII International Symposium on Information, Communication and Automation Technologies (ICAT 2009), Bosnia, (2009)
8. Bunse, C., Gross, H.G., Peper, C.: Applying a model-based approach for embedded system development, in Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, Washington, DC, USA, (2007)
9. Gomes, L., Fernandes, J.: Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation, IGI Global's, (2009)
10. Carloni, L.P., Sangiovanni-Vincentelli, A.L.: A formal modeling framework for deploying synchronous designs on distributed architectures, in In FMGALS: Formal Methods for Globally Asynchronous Locally Asynchronous Architecture. Elsevier, (2003)
11. 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)
12. Suhaib, S., Mathaikutty, D., Shukla, S.K.: Dataflow architectures for GALS, Electronic Notes in Theoretical Computer Science, 200:33--50, (2008)
13. Reisig, W.: Petri nets: an introduction, Springer-Verlag New York, Inc., NY, USA, (1985)
14. Moutinho, F., Gomes, L., Barbosa, P., Barros, J.P., Ramalho, F., Figueiredo, J.: A. Costa, and A. Monteiro, Petri Net Based Specification and Verification of Globally-Asynchronous-Locally-Synchronous System, in 2nd DOCEIS - Doctoral Conference on Computing Electrical and Industrial Systems. Springer, (2011)
15. Ribeiro, J., Moutinho, F., Pereira, F., Barros, J.P., Gomes, L.: An Ecore based Petri net Type Definition for PNML IOPT Models, INDIN'2011 - 9th IEEE International Conference on Industrial Informatics, Caparica, Lisbon, Portugal, (2011)
16. Moutinho, F., Gomes, L.: State Space Generation Algorithm for GALS Systems Modeled by IOPT Petri Nets, 37th Annual Conf. of the IEEE Industrial Electr. Society, Australia, (2011)
17. Gomes, L., Rebelo, R., Barros, J., Costa, A., Pais, R: From Petri net models to C implementation of digital controllers, Proceedings of the ISIE'2010 - IEEE International Symposium on Industrial Electronics, Bari, Italy, (2010)
18. Gomes, L., Costa, A., Barros, J., Lima, P.: From Petri net models to VHDL implementation of digital controllers, 33rd Annual Conf. of IEEE Industrial Electr. Society, Taiwan, (2007)