The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland

USING HIERARCHICAL STRUCTURING MECHANISMS WITH PETRI NETS FOR PLD BASED SYSTEM DESIGN

Luis GOMES, João-Paulo BARROS

Universidade Nova de Lisboa & UNINOVA,
P-2825-114 Monte de Caparica, PORTUGAL, *<lugo, jpb>@uninova.pt*

Abstract. *This paper addresses the use of hierarchical model structuring mechanisms for the design of embedded systems (in the sense of reactive real-time systems), using Reactive Petri nets. Relevant characteristics of Reactive Petri nets are briefly presented and their main roots are identified, namely Coloured Petri nets, Interpreted and Synchronised Petri nets.*
Two structuring techniques will be presented. The first one concerns with the integration of bus signals representation into non-autonomous Petri net models and the second one with a graphical structuring mechanism named by horizontal decomposition. This mechanism relies on the usage of macro-nodes, which have sub-nets associated with them. Three types of macro-nodes were used: macro-place, macro-transition and macro-block. The execution of the model is accomplished through the execution of the flat representation of the model.
At the end, a case study is presented around a controller of a 3-cell FIFO system. High-level and low-level Petri net models were used and compared for that purpose. The implementation of the referred controller was done using programmable logic devices, namely PALs and CPLDs.

Key Words. *Petri nets; Hierarchical structuring mechanisms; Programmable Logic Devices*

1. INTRODUCTION

It is widely acknowledged that the support for specific model structuring mechanisms is of the most importance for a complex system designer. In this sense, the use of techniques to build up compact models through the use of a "divide to conquer" strategy are commonly accepted as necessary. Common techniques use different levels of abstraction enabling the construction of the model in an incremental way.

From the point of view of the hardware design, the concepts of modular modelling have been widely used for several decades. Also from the point of view of software design, the effects of structured programming, modular programming and of object-oriented programming have been supporting the reusability and robustness of code. Complementarily, the use of data structures could help towards compactness and expressiveness of the produced specification.

These concepts and tendencies have been ported for the Petri net family of formalisms through a large number of proposals integrating hierarchies and modularity into Petri nets [4] [3] [1] [2]. Some of them support translation into simpler models (which means that they have the same modelling power), while other ones do not.

From our point of view, the concepts proposed in [4] are of major importance, namely substitution transitions (probably the most used hierarchical structuring mechanism; it is also used in Coloured Petri nets [1] and in Design/CPN, the most used Petri net tool), and substitution places (this mechanism plays a dual role related with substitution transitions; the execution of the model is accomplished through the flat model obtained after the fusion of the two-levels of nets. It is important to stress that it is not allowed to directly connect a substitution place to a substitution transition; although already identified in [4], it was considered there not to be a serious restriction. We disagree with that assumption.

This paper does not present, or use, some other important model-structuring mechanisms. Among them, one can mention the concept of depth, extensively used in Statecharts, and integrated in Reactive Petri net class [2] (which led to the concept of vertical decomposition structuring mechanism).

2. ABOUT REACTIVE PETRI NETS

This section briefly presents a non-autonomous high-level Petri net model, referred as Reactive Petri net model (RPN) [2]. In the following paragraphs, the model characteristics will be briefly presented in a non-formal way, around two components: the autonomous and the non-autonomous parts. In what concerns with the autonomous part of the model, Coloured Petri nets is the reference model [1]. The transition firing semantics is also similar.

The non-autonomous part in its turn can be divided in two main parts: input-output modelling and other execution issues. In the second group, one can found priorities associated with transitions, automatic conflict resolution and time modelling. From the point of view of this paper, only the input modelling is relevant. Interpreted and synchronised Petri nets [5] constitute the references to the input and output modelling. In this sense, input is modelled through event conditions associated to transitions. Output actions can be associated to markings or to transition firings.

The transitions firing rule was changed so as to consider the new input dependency. As such, for a transition to fire it must fulfil two conditions: it must be enabled, by the existence of a specific binding of tokens presented in the input places, and it must be ready, the external input evaluation must be true. Every transition enabled and ready will be fired. This means that the maximal step is always used. It is the followed approach in the synchronised Petri nets and interpreted Petri nets models.

3. PROPOSED HIERARCHICAL STRUCTURING MECHANISMS

Most of the known hierarchical structuring techniques emphasises the autonomous part of the model. Also, most of them use the flat model for the execution of the model; in this sense, the hierarchical structure is not reflected at the implementation level. This is also the case for the structuring mechanisms proposed in this paper.

3.1. From signals to buses support

Explicit modelling of interdependencies between the autonomous and non-autonomous parts of the model can be obtained in two situations:

- case I): binding variables can be used in event conditions;
- case II): input signals can be used in arc inscriptions, which implies that input signals can determine characteristics of coloured tokens.

From the designer point of view, this characteristic could be of major interest in terms of the compactness of the produced model. It has to be stressed that it is possible to find a flat model that is behaviourally equivalent to the model with mutual-dependent attributes, as far as the cardinality of the variables involved in those cross-references is finite. This will also be shown in the following paragraphs.

Case I situation is presented in Figure 1(a). There, a vector of input signals *a[i]* is associated with one transition. The specific binding of the tokens may select a specific input signal from the array. The flat model can be found replicating the transition as well as the arcs and inscriptions by a factor equal to the number of elements of the vector (two in the figure). The Figure 1(b) shows the flat model behaviourally equivalent to the model of Figure 1(a).

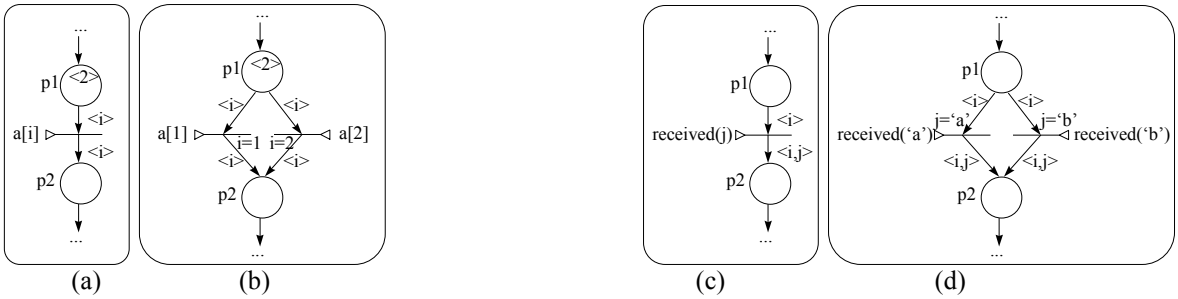

Figure 1 - Translation of specific inscriptions into flat models.

Case II situation is presented in Figure 1(c), where an external event will impose part of its own characteristics into generated tokens coloured attributes. The example models the synchronisation between a local activity (represented by p1 marking) and an external activity that emits an event with some attached information (in the example, two possible values are considered). The Figure 1(d) shows the behaviourally equivalent flat model. It has to be stressed that one needs to know in advance the type and cardinality of the external information in order to be able to produce the behaviourally equivalent flat model.

3.2. Horizontal decomposition

The horizontal decomposition mechanism is defined in the "common" way used by top-down and bottom-up approaches, supporting refinements and abstractions, and is based on the concept of module. The module is modelled in a separated net, stored in a page; every page may be used several times in the same design. The pages with references to the modules are referred as super-page (upper-level pages), while the pages containing the module model are referred as sub-pages (lower-level pages).

The execution of the model is accomplished through the flat model, which was the motivation for the naming of this decomposition type. This type of hierarchical decomposition only uses autonomous characteristics of the model. So, in this sub-section, non-autonomous characteristics, like signals, are irrelevant.

The nodes of the net model related with hierarchical structuring are named by macro-nodes. Three types of macro-nodes are foreseen: macro-place, macro-transition (also used by Hierarchical Coloured Petri nets [1]) and macro-block. Every macro-node will have an associated sub-page that can be referred as a macro-net. Distinctive graphical notations are used for the representation of macro-nodes: macro-places are represented by a double circle

(or ellipse), macro-transitions by a rectangle and macro-blocks are represented half-macro-place and half-macro-transition. Due to space limitation, detailed examples are not presented, nor associated detailed semantics.

A macro-place corresponds to a (macro-)node where all the arcs are connected to transitions, which means that the boundary of the macro-net (i.e. of the sub-page) is composed by a set of places. A macro-transition corresponds to a (macro-)node where all the arcs are connected to places. This means that the boundary of the sub-net should be composed by transitions; however, due to the necessary coloured binding evaluation for those transitions, the actual connections to input places and arc inscriptions are needed. So, in the sub-page associated with the macro-transition, references to places of the super-page have to be included. However, from the point of view of the execution of the model, the places connected to a macro-transition exist only at the super-page level; the places presented at the sub-page are void and will be merged with the associated places at the super-page.

As far as one macro-transition can not be executed like an ordinary transition and also that a macro-place cannot be considered as an ordinary place (which means that it cannot hold a marking), they act just as a graphical modelling convenience enabling the designer to structure the graphical model in an expressive way. In this sense, it is interesting to consider the use of graphical modules that can have arcs to/from places and transitions at the same page. They correspond to modules in a more general way and we name them macro-blocks. As far as this representation is not an executable specification (at this level), the bipartite intrinsic characteristic of Petri nets is not violated.

The following steps compose the merging process of the sub-page into the super-page:

- References of places and transitions used by the sub-page will be eventually changed in order to produce unique labels;
- One copy of the sub-page is inserted at the super-page; the macro-node reference is removed;
- The arcs connected with a macro-place semantics are connected to the referred boundary place; for arcs connected with a macro-transition semantics, void boundary places of the sub-page are merged with the associated places at the super-page (arcs and associated arc inscriptions in the sub-page are kept).

The interface of the macro-net (i.e. of the sub-page) is composed by a set of boundary places. As referred, this set of places will constitute the glued points between the sub-page and the super-page. However, when the sub-page boundary place is associated with a macro-place (or with a connection to a place in a macro-block), the boundary place is a common place (in the sense, that it can be marked, for instance). If the sub-page boundary place is associated with a macro-transition (or with a connection to a transition in a macro-block), the boundary place is a void place (in the sense, that it can not be marked, for instance). This characteristic allows the direct interconnection of macro-nodes, which can be of most interest in real-world applications, as we will try to emphasise in the following section.

4. CASE STUDY

An example illustrating the application of the proposed macro-nodes is presented. It is a 3-cell FIFO (first-in-first-out) system controller for assembly activities, referred in [2]. The assembly cell system has a conveyor to transport objects to the different cells. Each cell has presence sensors to detect objects on its in- and outputs, connected to the variables *in[1..3]* and *out[1..3]*. Each cell also has conveyor movement control, through the variables *move[1..3]*.

Figure 2 shows a simplified characterisation of the system controller, and two Reactive Petri net models as well. At the right hand side model, coloured characteristics support an easy modification of the model to accommodate a different number of cells. In the simplified model at the centre, a module-centric model based on the cell concept is presented; each cell is modelled by a macro-block. Associated detailed low-level model can be found in Figure 3. The usage of macro-blocks enables a more intuitive modelling and direct interpretation of the corresponding systems presented at the plant.

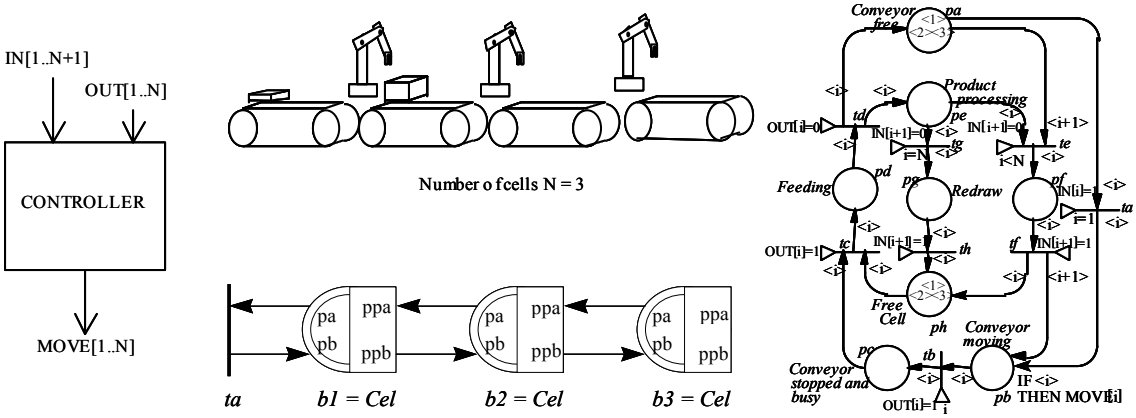

Figure 2 - N-cell FIFO system model.

The final goal is to implement the referred controller through a programmable logic device (PALs or CPLDs). As far as it is not possible to use a systematic direct translation procedure from high-level nets to hardware implementation, several possibilities were analysed to accomplish the goal.

Using the horizontal decomposition mechanism, it is possible to model the system in different ways, using high-level or low-level nets. Due to space limitations, it is not possible to present different examples. One is briefly presented considering the use of low-level Petri nets, where an equivalent model was obtained; it is presented at Figure 3. Three areas are identified associated with the different cells (which means to a different token in terms of the initial model).

Although the example complexity is relatively low, and so no general conclusions could be taken, different possible solutions were implemented and tested using PLDs (simple PLDs like PALs and CPLDs 9536 and 95108 from Xilinx). Possible implementation strategies rely on the following attitudes:

- Start with the coloured model and build up the associated space state, which is behaviourally equivalent to the original model; then implement the associated state machine in hardware using well-known techniques;
- Unfold the coloured model into low-level nets (as the one presented in Figure 3), using automatic tools; with the low-level model, different approaches are possible:
  - direct implementation based on direct translation of each node into hardware structure (translating places by memory elements and transitions by combinatorial logic);
  - indirect implementation based on the associated state space;
  - partitioning of the model, based on the net characteristics, and parallel implementation of the several parts.

Among the different solutions tested and associated with the last referred method, for instance the development of a hardware module associated with the macro-blocks presented in Figure 2 (centre), one other was based on place invariant computation. In the referred example, based

on the model of Figure 3, one can get four invariants with very interesting characteristics: i) the four place invariants cover all the places of the net, and ii) each invariant can be seen as a state machine (as far as only one and only one place of the invariant is marked at a time). In this sense, the implementation of the system can be based on the partitioning of the model into four concurrent state machines, which were trivially implemented in a PLD.

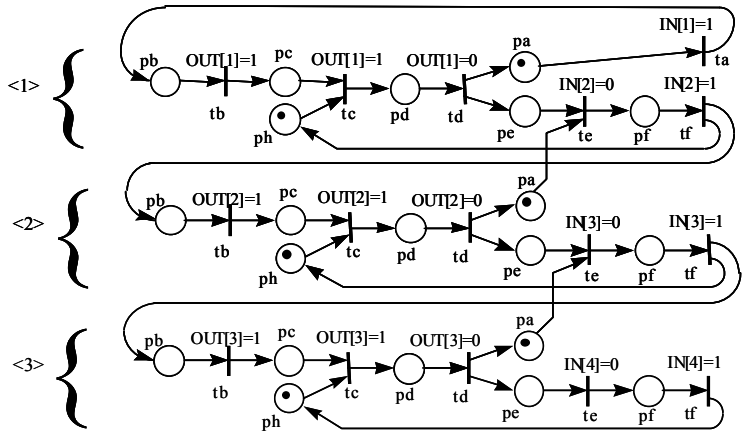

Figure 3 - Low-level representation of the 3-cell FIFO system.

5. CONCLUSIONS

In this paper, Petri net-based digital systems design was addressed. Two techniques were presented towards a better model structuring. Their usage was successfully validated through an example of a controller for a low-to-medium complexity system, which was implemented based on programmable logic devices (PALs and CPLDs). The first technique introduces the representation of arrays of signals into the Petri net model, while the second one uses the concept of module, which can be used associated with special kind of nodes, named by macro-nodes. Three kinds of macro-nodes were proved to be of interest.

ACKNOWLEDGMENTS

The presentation of the work at the DESDes'01 Workshop is supported by the IST project 10258 DOTS – "Distributed Object Telecontrol Systems and Networks".

REFERENCES

[1] K. Jensen; 1992; "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 1"; Springer-Verlag; ISBN 3-540-55597-8

[2] L. Gomes; 1997; "Redes de Petri Reactivas e Hierárquicas - integração de formalismos no projecto de sistemas reactivos de tempo-real"; in Portuguese; PhD Thesis; Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia; 318 pages

[3] Luca Bernardinello, Fiorella De Cindio; 1992; "A Survey of Basic Net Models and Modular Net Classes"; in "Advances in Petri Nets 1992"; Lecture Notes in Computer Science; G. Rozenberg (Ed.); Springer-Verlag

[4] P. Huber, K. Jensen, R.M. Shapiro; 1990; "Hierarchies in Coloured Petri Nets"; in "Advances in Petri Nets 1990", Lecture Nores in Computer Science LNCS 483, G. Rozenberg (Ed.); pp. 313-341

[5] R. David, H. Alla; 1992; "Petri Nets & Grafcet; Tools for modelling discrete event systems"; Prentice Hall International (UK) Ltd; ISBN 0-13-327537-X.