Technical Report DAIMI/PB-530, March 1998.

Digital Systems Synthesis from Petri Net Descriptions[*]

Norian Marranghello[†]
Datalogisk Institut
Århus Universtitet
norian@daimi.aau.dk

Abstract. *The design of digital systems has reached a degree of complexity that virtually prevents their effective realization without computer aided design tools. Several languages were already proposed to be used in such tools, each with the objective of capturing as much hardware characteristic as possible. During about the last fifteen years the importance and use of Petri net as a language for modeling digital systems have greatly increased. Many computer aided design tools dealing with Petri nets for the analysis, verification and synthesis of this sort of hardware have been recently developed as well. With such a growing importance in mind, this report aims at presenting an overview of the research going on the application of Petri nets to the description of digital systems and the synthesis of the corresponding hardware from these descriptions.*

1 Introduction

The automation of the synthesis process of digital systems is an issue whose importance steadily grows with the constant increase in the complexity of such systems. The use of Petri nets in the modeling and simulation of complex systems has proved very worthwhile. Petri nets have already been successfully used for the specification, analysis, and synthesis of digital systems for several years by many groups around the world. The main goal of this report is to provide an overview of the research going on the synthesis of digital systems from Petri net descriptions.

The first step in the design of a digital system is its specification. The very first ideas are usually presented in a natural language. The task of the designer is to develop the idea into a collection of hardware and/or software components to perform the required function(s).

---

[*] The relevant portions of this report, mainly those in section 3, were submitted for revision by the individuals referred to in each paragraph. All comments received from them were incorporated in the text. Even so, some misunderstanding may still remain. Therefore, whatever comments about other people's work, included herein, should be regarded as reflecting the unique impression and understanding of their work by the author of this report.

[†] On leave from the Department of Computer Science, São Paulo State University, Brazil; with a post-doctoral fellowship from the Fundação de Amparo à Pesquisa do Estado de São Paulo, Brazil, grant FAPESP-96/11164-7.

Kishinevsky et al.[01] present an example anecdote that, in my opinion, illustrates very well the vagueness of such kind of specification. Their illustrative story is about the following: *A soldier in the front comes to his commanding officer with the idea of building a weapon to reach far beyond the enemies' lines so as to cut their supplies. Weakened, the opposing military forces would more easily be defeated. The soldier was immediately sent to the nation's president. When asked how to build such a magnificent weapon the soldier replied that his duty was to suggest the idea. Likewise, in order to develop it into a practical device there was a full team of designers much more qualified than him.* Although Kishinevsky and co-workers offer a slightly different version of the joke, using it in a somewhat unlike context too, it serves very well to show the level of abstraction in which an initial specification can be proposed. The designer can be presented with specifications suchlike the one above and has to eventually come up with a system to carry out the required set of functions.

Considering such a broadness it can be argued that the steps leading from an initial specification towards a final implementation of a system may be thought of as a process of synthesis. This is certainly the case that the scope of the terms "final implementation of a system" and "process synthesis" are too coarse. Thus, more specific definitions for them are in order. These two terms can be represented by three words that will be explained in the sequel, namely: system, implementation and synthesis.

Let's begin by saying that all through this report the word *system* will mean a digital system, i.e., those systems that manipulate discrete elements of information, unless strictly stated on the contrary.

The second word, *implementation*, can have two different scopes, i.e., it is used both in the software and hardware contexts. In the former context it is assumed to be the generation of a piece of code written in some computer programming language. This code should be ready to use, if a low-level language is employed, or directly convertible to a machine understandable language, when the code is produced in a high-level programming language. In the latter context it is interpreted in this report as the actual generation of an integrated circuit directly into a silicon or some other suitable substrate.

The third word, *synthesis*, needs a bit more elaborate and longer explanation, for which some help from Gajski and co-workers is greatly welcome. In this report it is adopted the more general definition of synthesis as the process of refining a description given in some higher level of abstraction into another one at a lower level of abstraction. Considering the above definition of synthesis and the division of the design space as proposed by Gajski and Kuhn[02], it is possible to define high- and low-level synthesis as being "a translation from a behavioral description into a structural description"[03], and a translation from a structural description into a geometrical one, respectively.

It is possible to identify several levels of abstraction within each region of the design space in which to describe a digital system. From a more theoretical point of view down to a more

concrete one at least four levels of abstraction can be recognized in the composition of the structural region, namely the architectural, register-transfer, logic, and circuit levels. At the architectural level the structure of a system is described as a set of processors, memories, controllers and interfaces. At the register-transfer level the structure of the system is split into a data path and a control unit. The data path is composed of a set of registers to store information and some functional units to manipulate the data while transferring them among the registers. The control unit is a combinational circuit used for sequencing the actions in the data path. At the logic level the structure of the system is described in terms of logic gates and their timing relationships. At the circuit level the system is described through a set of transistors with the corresponding types, parameters and sizes.

The state-of-the-art approach to synthesize digital systems consists in the utilization of the so called co-design techniques. In this case the design process usually starts with a behavioral specification of the system. The initial specification is progressively refined in order to obtain the best match between its software and hardware components. After a series of refinements, the synthesis of the system is concluded with the implementation of both its software and hardware constituent parts.

This report is focused on the synthesis of hardware. Therefore, the reader interested in how to synthesize software or in more details about co-design is referred to the literature on the subject[04-12].

As for the hardware portion, the behavior of the system is initially provided in a suitable hardware description language. At the beginning of the design process the system is described from such a behavioral perspective. Then, a transformation from the behavioral to a structural viewpoint, frequently at the architectural level, is carried out. Thereupon, this description is gradually refined to lower levels of abstraction. As a rule, the architectural description of the system is further detailed down to the register-transfer level. At this point, the description of the system consists of an interconnection of registers, multiplexers and some other high-order logic entities, including some time dependence among them as well. At this stage, the so called high level synthesis process is completed. This is also, the level of detail in which most existing co-design systems provide the description of the hardware.

Next, an intermediate step converting the register-transfer description into a logic level one is needed. It is considered here as a changeover procedure because it is not clear at the moment whether this is part of a high- or low-level synthesis system. Existing synthesis approaches, at either level, may include or not this kind of transformation. Anyhow, the result is a description of the system in terms of (possibly simple) logic gates and a reasonably detailed timing relationship among such entities.

Typically, during the low level synthesis process the gates constituting the logic level description of the system are converted to transistors, whose parameters, suchlike sizes and types, are computed taking into account the timing information available. From the circuit level structural

description a geometric one is then contrived. Analogously to the structural region of the design space, the geometrical one can be seen from several levels of abstraction. Nevertheless, let's consider here that the resulting geometrical description be a layout of the final circuit, containing a profile of the geometric shapes to be used in the manufacturing of the desired system. Each of these shapes, such as rectangles, squares, and so on, represents a particular type of substrate and was conveniently sized in order to reflect the characteristics of each transistor in the proposed hardware.

Perhaps VHDL is the most popular hardware description language at the moment. Nevertheless, Petri nets are becoming continually more popular for the description of digital systems. This is especially true in the case of asynchronous circuits, where a great likeness of some fundamental principles to those of Petri nets can be observed. During the last 15 years or so the importance and use of Petri nets as a design aid to digital systems have greatly increased. Two main reasons can be identified as catalysts of such a trend. On the one hand, Petri nets constitute a language capable of capturing causality relations, concurrency of actions and conflicting conditions from digital systems in a natural and convenient way. On the other hand, Petri nets embed a theory that allows the description, analysis and verification of digital systems in a formal yet easy to use basis.

The rest of this report is organized as follows. In section two a brief review of the main Petri net related techniques used during the design of digital systems is provided. In section three an overview of the work going on the subject title is given. In section four a high-level Petri net extension is briefly described and some ideas that may lead to the development of a synthesis system are presented. Final comments are offered in section five.

2 Techniques

In this section it is offered an overview of the main Petri net related techniques used during the design of digital systems. The reader may find some resemblance with the expositions by Yakovlev and co-workers[13, 14]. This is indeed the case. Notably this section of the report was positively influenced by their notes. As a matter of fact, this section is mostly a summary of the relevant parts of those articles, which is included here for two reasons: first to serve as a reference for the reader not quite familiar with some specific terms, second for the sake of completeness of the present overview.‡

It is assumed that the reader has a basic knowledge both of Petri nets and of hardware. Those unfamiliar with Petri nets could have profit from the report by Murata[15] and the books by

‡ It is worth raising a point here: as a summary, the present text cannot be as complete as the original papers, its virtue should be to reduce a many-pages reading to 2 or 3 pages, highlighting what the author of the summary regards as more relevant. Furthermore, This text is not only a summary, it includes some personal views and interpretations of the author. Thus, at no moment should the reader understand the present text as containing anything else but the author's personal view and understanding of the several cited works.

Reisig[16] and by Jensen and Rozenberg[17]. Those unfamiliar with hardware are referred to the books by Gajski et al.[03] and by Kishinevsky et al.[01]. Those familiar with both topics that want more detail on the material presented here are referred to the papers by Yakovlev et al. cited in the former paragraph. Finally, those familiar with the specific field of application of Petri nets to the synthesis of digital systems may well want to skip the reading of this section.

The design process of a digital system comprises roughly three parts: modeling, analysis/verification, and synthesis; each of which is informally explained below.

The formal verification of a system requires it to be described through a formal model. The modeling language considered here is Petri net, which is a bipartite directed graph, i.e., a graph with two kinds of vertices: local states (usually referred to as places) of the net, graphically represented by ellipses, and actions (usually referred to as transitions) of the net, graphically represented by rectangles. Each transition is connected to one or more input places through directed edges, which have their origin at a place, and end at the considered transition. Each transition is also connected to one or more output places through directed edges, which originate at the transition, and end at the output place(s). The global state of the system (also known as net marking) is represented by the assignment of tokens to each of the net places, resulting in a general net marking. A change to the net marking can be seen as the variation of the token assignments, i.e., by moving the tokens around through the places. A change in the net marking corresponds to the execution of actions (known as transition firing or occurrence) according to the following basic rules:

(a) a transition is enabled if each of its input places has at least one token;
(b) any enabled transition can occur, and its firing is represented by removing a token from each of the corresponding input places and inserting a new token in each of its output places; and
(c) enabled transitions can occur concurrently as long as they are independent, i.e., use different tokens.

According to such a definition, a Petri net can be represented by a tuple $PN = (P, T, F, M_o)$ where P is the set of places, T is the set of transitions, F is a flow relation defining directed edges connecting either places to transitions or transitions to places, and M_o is the initial marking of the net. Next, a labeled Petri net is defined as a Petri net in which every transition is labeled with a symbol from an alphabet A, originating the corresponding labeling function $L : T \rightarrow A$. In order to bring this theory closer to digital systems a signal transition graph (STG) is defined as a Petri net whose transitions are labeled with the names of binary signal transitions, i.e., the alphabet considered is the set $A = \{a^+, a^-, a^{\sim}\}$. The elements of the alphabet correspond to the upward (a^+), downward (a^-) and either ($a^{\sim}$) transition of signal a. Then, an STG can be viewed as a triple $STG=(PN, A, L)$, where its components are defined as above. As an example, Fig.1 presents de modeling of an inverter both by Petri nets and STGs. In Fig.1(b) it is assumed that the state of the system, from which the Petri net fragment was extracted, was reset so that the output y is equal to zero; it is also assumed that in some other part of the system an action producing a one-

to-zero ($x : 1 \rightarrow 0$) transition on signal x will eventually occur, thus, removing the token from place $x = 1$, generating a token in place $x = 0$, and enabling the upward ($y+$) transition of signal y.

The model produced must be analyzed, i.e., the designer needs to answer some questions about the behavior of the system. Furthermore, it is desirable to verify some characteristics of the system; or putting it in another way, it is desirable to rigorously prove that the designed system possesses some formally stated attributes. In order to do so, three approaches are considered, namely: reachability graphs, symbolic traversal, and partial orders.

Reachability analysis consists in producing a graph containing all possible markings of the net and all possible sequences of firings. The main problem with this method is that the number of possible states grows enormously even for small examples, causing an effect known as the state explosion problem. To avoid this problem Valmari[18] proposed a method, named stubborn sets. In this method only one of a group of concurrently enabled transitions is analyzed at a time. Nonetheless, in order to overcome the state explosion problem, only part of the reachability graph is generated. Consequently, not all properties of the system can be analyzed. For instance, Valmari proved that his method can always detect deadlocks of the system. However, by hiding some states of the system, it may not be possible to prove the completeness of the state coding (CSC) property[19], which means that it may not be feasible to verify that the logic circuit corresponding to the modeled system is implementable. Furthermore, as the states of the system are manipulated and not all of them are generated this procedure may conceal the causal ordering of events preventing a timing analysis of the model.

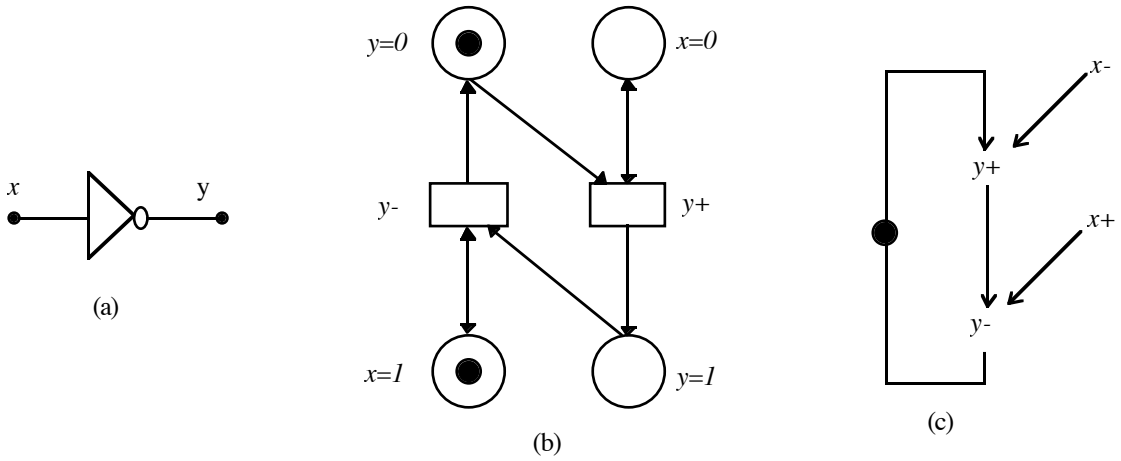

Figure.1 - Example of an inverter logic gate (a);
its description by a Petri net fragment (b);
and the corresponding STG representation (c).

The second approach starts by constructing a k-variable boolean function where each variable represents one place of the net and expresses the binary condition of existence or not of a token in the corresponding place. In such a way the net is described through a characteristic function, which in turn is represented as a binary decision diagram (BDD)[20]. A transition function, transforming markings into one another, is computed for each enabled transition at each marking. By successively applying the transition function to each marking, the symbolic image of the full state-space is generated. As compared to the reachability analysis, the symbolic traversal approach has a better performance due to boolean characterization. Covering all reachable markings it is also possible to prove the CSC property with this technique. Nonetheless, there still may be not enough information about the causality ordering of events in order to proceed a timing analysis of the model.

The partial ordering representation is given by the Petri net unfolding[21, 22]. Informally, the unfolding of a net is its transformation into another equivalent one in which the cycles are broken and extra places are inserted to preserve the semantics of the original net. With this approach there is included enough information to cover all possible markings. As the cycles in the original net are broken, it could be the case that the information about the causal relationship of events became hidden by such cuts. Though, Semenov and Yakovlev[23] used it for the analysis of timed models of asynchronous circuits. In order to do that, time-state classes are associated to the markings produced allowing the timing analysis.

The third part of the design process is the actual synthesis of the desired system. To this purpose two approaches are possible. They are the syntax directed and the STG-based approaches, both of which are commented bellow.

The syntax directed synthesis starts with a description of the system as a labeled Petri net. The components of this net are effectively replaced with circuit elements. For this to be feasible the net must be at least bounded, if not 1-safe, so that the generated circuit have a meaningful implementation. There are two ways to generate the required circuit, they are the place-to-latch and the event-based mappings. The former maps each place in the net to a memory latch (flip-flops in the case of 1-safe nets and up/down counters for bounded ones) and each transition to some appropriate random logic at the input of the corresponding latches. This can be used to synthesize relatively large circuits as long as the constraints on area and speed are not very critical. The latter is an extension of Patil's mapping. He used six elements and eight modules composed of these elements to represent Petri net structures[24-26]. This extension, uses a seventh component, named decision-wait, that allows the synchronization of events in different groups of mutually exclusive signals. This gives way to the mapping of unsafe, although bounded, Petri nets. Syntax directed approaches require that all transformations be done at the Petri net level. The reason for this is that doing any change at the circuit level would at least risk destroying the semantic soundness of the circuit in respect to the verified net.

A complimentary technique takes the system specification in terms of an STG, and generates its state graph. The state graph of an STG is the binary encoded form of the reachability graph of

the corresponding labeled Petri net. Using boolean minimization techniques a boolean function is derived from the state graph. Thus, the desired circuit can be generated through one of a number of methods available[27, 28].

3 Applications

The approach adopted by a group in Linköping, Sweden, was implemented in the CAMAD high level synthesis system[29]. The system is specified in a sub-set of the VHDL language named S'VHDL[30]. Such specification is then mapped onto an extended timed Petri net (ETPN) model consisting of two separate but related parts, namely, a control structure and a data path[31]. The available parallelism is extracted and a register transfer level description of the hardware is produced[32]. The data path is represented as a directed graph where the nodes capture data manipulation units such as data storage elements and arithmetic operators, and the arcs represent the interconnection of these elements. The control structure is represented as a timed Petri net with restricted firing rules, as defined in Peng's Ph.D. work[33]. Furthermore, the control structure communicates by issuing control signals to and receiving conditional signals from the data path. On the one hand, the data path graph is a register transfer level description extracted directly from the VHDL specification. On the other hand, the timed Petri net description of the control structure as well as the corresponding communication signals are usually implemented as hardware through the synthesis of one or more finite state machines, depending on the style chosen by the designer of the system.

A group in Paris, France[34], developed an approach to the high level synthesis of embedded systems based on the description of the system in VHDL[35], and the translation of the description to a model named interpreted and timed Petri nets (ITPN)[36] for formal verification of its properties. A system of boolean equations from the Petri net description is constructed first. System properties are then investigated using symbolic analysis of such equations[37]. Finally, the system is synthesized into a register level VHDL version for implementation with ordinary synthesis tools for VLSI circuits[38].

A group in Lisbon, Portugal, is developing a framework for the design of complex reactive systems; applications to programmable controllers were developed. The system is defined as a reactive Petri net[39] which is a Petri net class based on colored Petri nets (CPNs for short)[40], synchronous interpreted Petri nets[41] and StateCharts[42]. The model includes transition priorities and hierarchical structure constructs[43]. The analysis of the model is performed using the associated state space. The system is still under elaboration, however it is a goal of the development team to support the implementation of hardware on programmable logic components such as CPLDs and FPGAs, using direct and indirect synthesis of the Petri net model, within a co-design development environment.

The work described in this and the next three paragraphs reflects some results of international cooperation that is described here as being centered in Zielona Gora, Poland. This is the outcome of several different cooperations established by Prof. Adamski (Technical University of Zielona

Gora) with groups in Germany (Ilmenau), Portugal (Minho), and the UK (Bristol). The picture is about as follows: Prof. Adamski started research in Poland and had good links with the group headed by Prof. Fengler, in Ilmenau. Later he started collaborating with Prof. Dagless, in Bristol, UK. Afterwards, he spent some time as a professor at the University of Minho. At this stage he collaborated with two teams: one headed by Prof. Proença and the other by Prof. Monteiro, establishing a link between the groups in Ilmenau and Minho. Now, so to speak, a traditional cooperation of Zielona Gora with both Ilmenau and Minho seems to remain strong.

One of the works mentioned in the beginning of the previous paragraph is going on the synthesis of distributed discrete controller systems using a formalism based on Petri nets and knowledge based theories[44]. Such a synthesis system generates rule based formal behavioral specifications of logic controllers in the so called Petri net specification format (PNSF)[45]. The rule based specification is a description of the system used for the symbolic verification of its characteristics. Such a description can afterwards be transformed into a format, e. g. VHDL[46], accepted by some commercial field programmable compilers available, which are then used to synthesize the controller[47]. The PNSF description is directly mapped into the hardware library[48].

Another work describes an electronic computer aided design framework (SCBA) for the specification, validation, and high level synthesis of synchronous interpreted Petri nets (SIPN) based controller[49]. SIPN is an extension of ordinary Petri nets, to accomplish the specification of asynchronous parallel controllers. Within this framework, reachability graph analysis of the SIPN description is used to investigate the properties of the modeled system[49]. A compiler automatically translates the SIPN specification into a register transfer level VHDL description that is used for simulation and synthesis purposes by a package that accepts the VHDL subset generated by the compiler[41]. Recently, the SIPN model was improved to include hierarchies and object orientation concepts[50]. The new model, named shobi-PN, has been developed to include both the parallel controllers and the data path behavioral representation[51]. The shobi-PN model also extends the SIPN in the hierarchical mechanisms, since it supports both the use of hierarchy in the parallel controller specification (macronodes) and the use of hierarchy in the data path descriptions (macrotokens), exploiting the use of objects to model the data path resources[52]. The SOFHIA design flow supports the shobi-PN specifications by reusing the CONPAR/VHDL compiler and the SCBA environment and by generating SIPN specifications written in the CONPAR intermediate language[53].

In a third work[54], a methodology for the high level synthesis of synchronous controllers was presented. A Petri net formal specification of the controller is the starting point of the design process. The Petri net specification is represented by a binary decision diagram (BDD). The behavioral analysis and the verification of the properties of the modeled system are accomplished by boolean manipulation on the characteristic functions of the BDD representation. Using the result of a symbolic traversal of the Petri net and syntax directed techniques, a register transfer level description of the circuit is generated. The logic circuit is finally optimized and synthesized as either a field programmable gate array (FPGA) or into ASIC technology.

An abounding groundwork on the theoretical aspects of the design and synthesis of speed-independent asynchronous circuits has been done along the years. A strong research branch started some 20 years ago with the work of Rosenblum on Signal Graphs[55] a model that evolved such as defined by Rosenblum and Yakovlev[56]. Independently, Chu investigated Signal Transition Graphs[57], a model similar to Signal Graphs. The term Signal Transition Graphs is now commonly used to identify this form of notation. The relationship between STGs and the original Muller's model of speed-independent cirtuits has been studied by Kishinevsky et al.[01] and by Yakovlev et al.[58]. Centered in such a model Cortadella, Kishinevsky, Kondratyev, Lavagno and Yakovlev developed a theory[59-61] that was used by Cortadella in the implementation of the PETRIFY computer aided design tool[62]. A brief description of such a tool is presented in the following paragraph.

The tool PETRIFY was implemented by Cortadella at the Technical University of Catalunya, in Barcelona, Spain[62]. In PETRIFY the behavior of the system can be described as a Petri net, an STG or a transition system (TS). A TS is a special kind of state graph in which arcs are labeled with abstract names of events[61]. The initial description of the system is analyzed and a simpler one is produced by PETRIFY either in Petri net notation or as another STG. The digital system is analyzed utilizing state-based models such as those described by Cortadella et al.[61]. After the optimization of the specification the synthesis, i.e., the final gate selection from a given technology library is performed through the boolean function manipulation of the state graph information. A net list of a speed-independent logic level circuit corresponding to the initial behavioral specification is generated as the end product of PETRIFY.

Besides the work directly related to PETRIFY, the above people also made independent research related to both Petri nets and the design of asynchronous circuits. For example:

Cortadella worked with Pastor on a structural method for the synthesis of speed-independent circuits from STGs[63]. Cortadella also worked with Roig on the verification of asynchronous circuits specified with STGs. This verification approach is implemented in the tool VERSIFY[64].

Kishinevsky, Kondratyev and Taubin developed research on the synthesis of speed independent circuits from Petri net like models with OR-causality[01, 65]. Their concepts were implemented in the tool FORCAGE[01]. They also worked on a method for the verification of asynchronous circuits by unfoldings[66], which is implemented within the logic synthesis tool SIS[67].

Lavagno made research on the synthesis of asynchronous circuits with bounded delay model, starting from STGs, while at the University of California at Berkeley, USA[68, 69]. This model is implemented in the logic synthesis tool SIS[67].

Yakovlev, together with Koelmans and Lavagno, developed a two-stage methodology of designing asynchronous control circuits from Petri net descriptions[70], conducted research with Petrov[71]

on the synthesis of speed-independent circuits from STGs and with Semenov and Cortadella's group on the synthesis and verification of asynchronous circuits using unfoldings[23, 72].

4 Proposal

A high-level Petri net extension was developed by our group, in Brazil, in order to describe the characteristics of multiprocessor architectures. The analysis of systems described in such a model can be performed by the use of a computer program implemented to accomplish this end[73]. The main characteristics of this model are represented in Fig.2 and can be summarized as follows:

- The tokens are represented by natural numbers.
- Each place has a domain representing the set of tokens allowed to be there.
- To each transition there may be associated:
  - operations to be effected on the net variables;
  - boolean expressions whose absence default to true; and
  - time intervals indicating the minimum and maximum delays for the actual firing of the transition after its enabling, whose absence default to imediate firing.
- Two kinds of branches may connect places to transitions:
  - simple input branches, which connect one place to one transition and are labled with two values (min-max) representing the minimum and the maximum number of tokens to be removed from the input place when the transition fires; and
  - conjunctive input branches, which connect several places to one transition, labled in the same way as the simple input branches, however, removing the same token set from each and every input place when the transition fires.
- Output branches connecting transitions to places may have boolean expressions describing the token set to be put into the corresponding output place when the transition fires (the default output function is the union of all sets extracted from the input places).
- The conditions for the enabling of a transition are:
  - the cardinality of the token set present in each input place must match the one stablished by the corresponding input branch;
  - the boolean expression of the transition must evaluate to true; and
  - the token set resulting from the evaluation of the output function must have an empty intersection with the token set present in the corresponding output place.
- The firing rule that must be satisfied by the enabled transition in order that it is allowed to fire is that the lower time limit of the candidate transition must be lower than or equal to the lowest among the upper time limits of all enabled transitions.
- The effect of a transition firing is:
  - exclusion of a certain token set from the input places;
  - addition of the appropriate token set to the corresponding output places;
  - setting of the transition operations results; and
  - updating of the net times.

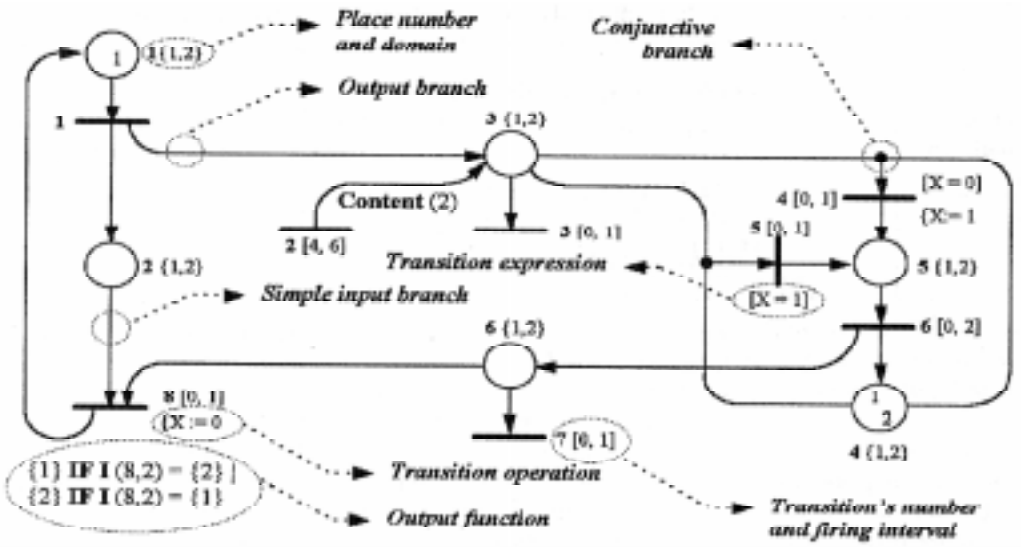

Figure.2 - Example of the characteristics of our high-level net

The simulation of a net is performed by changing classes, each of which representing a possible global state of the modeled system. Each net class is composed by a net marking, the corresponding time domain, and the values of the net variables. The relationship among the classes is described by a reachability graph. The nodes of such a graph represent the classes reachable from a given initial class. The nodes are connected by directed arcs labeled with the transition whose firing transformed the original class into the destination one. The token sets removed from each input place are also attached to the corresponding arc label.

After achieving the proper results, which may be confirmed through simulation analysis, the net model of the system should be converted to a low level Petri net in order to allow for a direct mapping of the net onto a circuit description, i.e., so that a direct synthesis of the system can be performed. Such an equivalent low level net can be obtained in a way similar to the one described by Jensen[74]. CPNs are far more elaborate, having many more facilities than our model, i.e., to certain extent CPNs can be considered as a super set of our model. Then, basically every transformation presented by Jensen apply to our model[§]. However, there are some minor details that are presented in our model and, to my knowledge, are not defined in the theory of CPNs.

The main difference between our model and CPNs is the case of conjunctive branches. These branches can be converted by using the analogy that follows. Each input place of the high level

[§] This statement must be formally proved before it can be taken as a truth.

net is converted to as many places in the low level net as there are tokens in the domains of the equivalent high level net places. The transitions in the low level net should match all possible combinations of tokens from the input places. This means that if one has two input places, such as places 3 and 4 in Fig.2, and these two places have two different tokens each (the domain of each place is {1,2} in Fig.2), they would have to be transformed into four different places and five different transitions in the low level net, as shown in Fig.3. Note that all transitions in Fig.3 correspond to transition 4 in Fig.2. This transformation can be optimized if the branch expressions are taken into consideration and only those combinations corresponding to valid outcomes for each expression are converted to structural objects of the low level net.

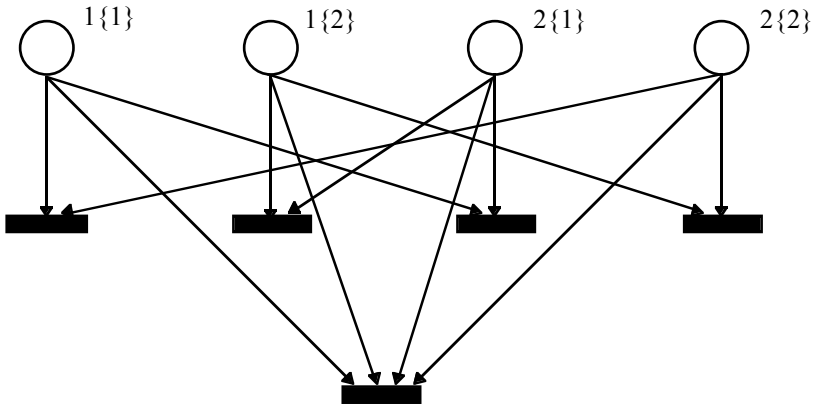

Figure.3 - Example of a piece of a low level Petri net.

After such a transformation, the resulting low level Petri net can be used to perform a direct synthesis of the system. This synthesis process can be done by using a one-to-one mapping of the Petri net structure onto a set of pre-defined hardware components. At this phase, methods similar to the one proposed by Patel[75] could be used. Patel's approach consists of four major steps, respectively involving: the production of a delay free net; the mapping of this net onto a hardware structure; the minimization of the hardware implementation; and the technology mapping with the optimization of the hardware implementation.

5 Conclusions

In this report it was presented an overview of the main research going on the application of Petri nets to the description of digital systems and the synthesis of the corresponding hardware from these descriptions. This overview is by no means claimed to be complete. To try to ensure a good coverability of the field I broadcast a message to the Petri net community, through the Petri net mailing list, asking everyone working in the field to let me know about their work. I tried to reflect in this report all replies that I received, as well as the data that I found with my own search. Therefore, it is my belief that most of the work in the field has been covered. At the end

of this report I presented some ideas that perhaps may lead to a fully Petri net based digital systems synthesis tool.

I would like to conclude this report by once more, so to speak, borrowing a quotation from the book by Kishinevsky et al.[01]. Their citation is attributed to G.Lichtenberg and reads suchlike this: *But who can stop me from picking a word here and a meaning there and then merging them together?*

6 Acknowledgments

I would like to express my gratitude to everybody that directly or indirectly contributed with the information that allowed me to produce this report. Special thanks are due to Dr. Kurt Jensen for hosting my stay in Denmark, to Dr. Søren Christensen for the review of the draft report, to Dr. Jaroslaw Mirkowski for feeding me with many pointers to people in the field as well as for the review of the draft report, to Prof. Marian Andrzej Adamski for several comments on his work in Poland and abroad, to Prof. Michael Kishinevsky and to Dr. Alexandre Yakovlev for many comments on the work described in the last seven paragraphs of section 3. Besides that I would like to thank Dr. Alexandre Yakovlev for having fed me with a lot of useful information since the very beginning of this report, his hints helped me find a way in the always fuzzy outset of this kind of search. Thanks are also due to everybody mentioned in section 3, whose names I do not list here to avoid forgetting somebody, but who deserve my highest acknowledgments for having so kindly reviewed my (sometimes fuzzy) thoughts about their work.

7 References

[01] M.Kishinevsky, A.Kondratyev, A.Taubin and V.Varshavsky *Concurrent Hardware: The Theory and Practice of Self-timed Design* John Wiley & Sons (ISBN:0471935360), 1994.

[02] D.Gajski and R.Kuhn *Guests Editors' Introduction* IEEE Computer Magazine, 16(12)11-14, 1983.

[03] D.Gajski, G.Nutt, A.Wu and S.Lin *High Level Synthesis - Introduction to Chip and System Design* Kluwer Academic Publishers (ISBN:0792391942), 1992.

[04] J.Gong, D.Gajski and S.Bakshi *Model Refinement for Hardware/Software Co-design* ACM Transactions on Design Automation of Electronic Systems, 2(1)22-41, 1997.

[05] P.Maciel, T.Maciel, E.Barros and W.Rosenstiel *A Petri Net Approach to Compute Load Balance in Hardware/Software Codesign* to appear in the Proceedings of the Conference on High Performance Computing, Boston, Massachusetts, U.S.A., April, 1998.

[06] G.de Micheli (Editor) *Special Issue on Hardware/Software Co-design* IEEE Proceedings, vol.85, no.3, March 1997.

[07] P.Pype *A New Approach to Hardware/Software Co-design for System Level Integration* XII Congresso da Sociedade Brasileira de Microeletrônica, http://www.dsif.fee.unicamp.br/sbmicro/papers/tutor01.pdf, 10pp., 1997.

[08] E.Stoy and Z.Peng *Inter-domain Movement of Functionality as a Repartitioning Strategy for Hardware/ Software Co-design* Journal of Systems Architecture, vol.43, pp.87-98, 1997.

[09] V.Catania, M.Malgeri and M.Russo *Applying Fuzzy Logic to Co-design Partitioning* IEEE Micro Magazine, 17(3)62-70, 1997.

[10] P.Eles, Z.Peng, K.Kuchcinski and A.Doboli *System Level Hardware/Software Partitioning Based on Simulation Annealing and Tabu Search* Journal of Design Automation for Embedded Systems, vol.2, pp.5-32, 1996.

[11] E.Stoy *A Petri Net Based Unified Representation for Hardware/Software Co-design* Licenciate Thesis, Linköping University, Sweden, LiU-Tek-Lic 1995:21, 88pp., 1995.

[12] J.Rozenblit and K.Buchenrieder (Eds.) *Co-design: Computer-aided Software/ Hardware Engineering* IEEE Press, (ISBN:0780310497), 1995.

[13] A.Yakovlev, A.Koelmans, A.Semenov and D.Kinniment *Modelling, Analysis and Synthesis of Asynchronous Control Circuits Using Petri Nets* Integration - The VLSI Journal, 21(3)143-170, 1996.

[14] A.Yakovlev and A.Koelmans *Petri Nets and Digital Hardware Design* to appear in Advances in Petri Nets, Lecture Notes in Computer Science, Springer Verlag, 1998.

[15] T.Murata *Petri Nets: Properties, Analysis and Applications* Proceedings of the IEEE, 77(4)541-580, 1989.

[16] W.Reisig *A Primer in Petri Net Design* Springer Verlag (ISBN:3540520449), 1992.

[17] K.Jensen and G.Rozenberg (Eds.) *High-Level Petri Nets, Theory and Applications* Springer Verlag (ISBN:354054125X), 1991.

[18] A.Valmari *Stubborn Attack on State Explosion* Formal Methods in System Design 1(1)297-322, 1991.

[19] T.A.Chu *Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications* Ph. D. Thesis, Massachusetts Institute of Technology, MIT/LCS/TR-393, 189pp., 1987.

[20] R.Bryant *Graph-based Algorithms fro Boolean Function Manipulation* IEEE Transactions on Computers, 35(8)677-691, 1986.

[21] K.McMillan *Trace Theoretic Verification of Asynchronous Circuits Using Unfoldings* Proceedings of the 7th International Conference on Computer Aided Verification, pp.180-194, P.Wolper (Ed.), Springer Verlag, 1995.

[22] J.Esparza, S.Römer and W.Vogler *An Improvement of McMillan's Unfolding Algorithm* Institut für Informatik, Technische Universität München, SFB-Bericht Nr.342/12/95-A, 16pp., 1995.

[23] A.Semenov and A.Yakovlev *Verification of Asynchronous Circuits Using Time Petri Net Unfolding* Proceedings of the 33rd ACM/IEEE Design Automation Conference, pp.59-62, 1996.

[24] S.Patil and J.Denis *The Description and Realization of Digital Systems* Digest of Papers of the 6th Annual IEEE Computer Scociety International Conference, pp.223-226, September, 1972.

[25] S.Patil *Circuit Implementation of Petri Nets* Computation Structures Group Memo 73, Project MAC, Massachusetts Institute of Technology, USA, 15pp., December 1972.

[26] S.Patil *Cellular Arrays for Asynchronous Control* Conference Record of the 7th Annual Workshop on Microprogramming, pp.178-185, Palo Alto, CA, U.S.A., 1974.

[27] A.Kondratyev, M.Kishinevsky, B.Lin, P.Vanbekbergen and A.Yakovlev *Basic Gate Implementation of Speed-independent Circuits* Proceedings of the 31st ACM/IEEE Design Automation Conference, pp.56-62, 1994.

[28] M.Sawasaki, C.Y.-Couvreur and B.Lin *Externally Hazard-free Implementation of Asynchronous Circuits* Proceedings of the 32nd ACM/IEEE Design Automation Conference, pp.718-724, San Francisco, CA, U.S.A., 1995.

[29] Z.Peng and K.Kuchcinski *Automated Transformation of Algorithms into Register-transfer Level Implementation* IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 13(2)150-166, 1994.

[30] P.Eles, K.Kuchcinski, Z.Peng and M.Minea *Synthesis of VHDL Concurrent Processes* Proceedings of the European Design Automation Conference, pp.540-545, 1994.

[31] Z.Peng and A.Törne *A Petri Net Based Modelling and Synthesis Technique for Real-time Systems* Proceedings of the 5th Euro-Micro Workshop on Real Time Systems, 1993.

[32] P.Eles, K.Kuchcinski and Z.Peng *Synthesis of Systems Specified as Interacting VHDL Processes* Integration - The VLSI Journal, 21(3)113-138, 1996.

[33] Z.Peng *A Formal Methodology for Automated Synthesis of VLSI Systems* Ph. D. Thesis, nr.170, Department of Computer and Information Sciences, Linköping University, Sweden, 1987.

[34] I.Augé, R.Bawa, P.Guerrier, A.Greiner, L.Jacomme and F.Pétrot *User Guided High Level Synthesis* Proceedings of the IX IFIP International Conference on VLSI, August, 1997.

[35] R.Bawa and E.Encrenaz *A Platform for the Formal Verification of VHDL Programs* Proceedings of the 4th International Workshop on Symbolic Methods and Applications in Circuit Design, Louvain, Belgique, 1996.

[36] R.Bawa and E.Encrenaz *A Tool for Translation of VHDL Descriptions into a Formal Model and its Application to Formal Verification and Synthesis* Lecture Notes in Computer Science, vol.1135, pp.471-474, 1996.

[37] E.Encrenaz *A Symbolic Relation for a Subset of VHDL'87 Descriptions and its Application to Symbolic Model Checking* Proceedings of the IFIP WG10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods, pp.328-342, Lecture Notes in Computer Science, vol.987, Springer Verlag, 1995.

[38] R.Bawa and L.Jancomme *Synthèse de Descriptions Comportementales Séquencielles en Conformité Avec la Sémantique VHDL* Actes de Colloque CAO de Circuits Intégrés et Systèmes, pp.303-306, 1997.

[39] L.Gomes and A.Garção *Programmable Controller Design Based on a Synchronized Colored Petri Net Model and Integrating Fuzzy Reasoning* Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science, vol.935, pp.218-237, 1995.

[40] K.Jensen *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use* Springer Verlag, vol.1 (ISBN:3540609431), 1992; vol.2 (ISBN:3540582762), 1994; and vol.3 (ISN:3540628673), 1997.

[41] J.Fernandes, A.Pina and A.Proença *Simulação e Síntese de Controladores Paralelos Baseados em Redes de Petri* Anais do VII Simpósio Brasileiro de Arquiteturas de Computadores - Processamento Paralelo, pp.481-492, 1995.

[42] D.Gajski, F.Vahid, S.Narayan and J.Gong *Specification and Design of Embedded Systems* Prentice Hall Inc. (ISBN:0131507311), 1994.

[43] L.Gomes *Sobre Algumas Atividades de Pesquisa na Uninova* Private Communication, 1997.

[44] M.Adamski, J.Monteiro, W.Fengler and A.Wendt *Distributed Petri Net-based Discrete Controller Systems* Proceedings of an International Conference on Control, vol.2, pp.777-782, Oporto, Portugal 1996.

[45] T.Kozlowski, E.Dagless, J.Saul, M.Adamski and J.Szajna *Parallel Controller Synthesis Using Petri Nets* IEE Proceedings, part E - Computers and Digital Techniques, 142(4)263-271, 1995.

[46] M.Adamski and J.Monteiro *Petri Net Modeling and VHDL Simulation of Descrete Mechatronic Systems* Proceedings of the International IEE Conference on Mechatronics, vol.1, pp.397-402, Guimarães, Portugal, 1996.

[47] M.Wegrzyn, M.Adamski and J.Monteiro *Reconfigurable Logic Controller with FPGA* Proceedings of the 4th IFAC Workshop on Algorithms and Architectures for Real Time Control, Algarve, Portugal, 1997.

[48] W.Wegrzyn, M.Adamski and J.Monteiro *VHDL Simulation of Xilinx-FPGA-based Concurrent Controller* Proceedings of a Workshop on Application of Programmable Logic, pp.12-15, Lisbon, Portugal, 1996.

[49] J.Fernandes, M.Adamski and A.Proença *VHDL generation from hierarchical Petri Net Specifications of Parallel Controllers* IEE Proceedings, part E - Computers and Digital Techniques, 144(2)127-137, 1997.

[50] R.Machado, J.Fernandes and A.Proença *Redes de Petri e VHDL na Prototipagem Rápida de Sistemas Digitais* 4º Encontro Nacional do Colégio de Engenharia Eletrotécnica, Ordem dos Engenheiros de Portugal, 1997.

[51] R.Machado, J.Fernandes and A.Proença *Specification of Industrial Digital Controllers with Object-Oriented Petri Nets* IEEE International Symposium on Industrial Electronics, Guimarães, Portugal, 1997.

[52] R.Machado, J.Fernandes and A.Proença *An Object-Oriented Model for Rapid Prototyping of Data Path/Control Systems - A Case Study* To appear in the 9th IFAC Symposium on Information Control in Manufacturing, Nancy and Metz, France, 1998.

[53] R.Machado, J.Fernandes and A.Proença *SOFHIA: A CAD Environment to Design Digital Control Systems* In C. Delgado Kloos and E. Cerny (Eds.) Proceedings of the XIII IFIP Conference on Computer Hardware Description Languages and their Applications, pp. 86-88, Chapman & Hall, 1997.

[54] K.Bilinski and E.Dagless *High-level Synthesis of Synchronous Parallel Controllers* Proceedings of the 17th International Conference on Application and Theory of Petri Nets, J.Billington and W.Reisig (Eds.), Lecture Notes in Computer Science, vol.1091, pp.93-112, 1996.

[55] L.Ya.Rosenblum *The Signal Graph Language for the Modeling of Exchange Protocols and Aperiodic Circuits* Proceedings of the Simulation of Digital Control and Computer Systems, Sverdlovsk, IMM, 1981 (in Russian). This work is cited as reference 35 of the book by Kishinevsky et al. [01].

[56] L.Rosenblum and A.Yakovlev *Signal Graphs: from Self-timed to Timed Ones* Proceedings of the International Workshop on Timed Petri Nets, pp.199-206, Torino, Italy, 1985.

[57] T.A.Chu *On the Models for Designing VLSI Asynchronous Digital Systems* Integration: the VLSI journal, 4(1)99-113, 1986.

[58] A.Yakovlev, L.Lavagno and A.Sangiovanni-Vincentelli *A Unified Signal Transition Graph Model for Asynchronous Control Circuit Synthesis* Formal Methods in System Design, 9(3)139-188, November 1996.

[59] J.Cortadella, M.Kishinevsky, A.Kondratyev, L.Lavagno and A.Yakovlev *Complete State Encoding Based on the Theory of Regions* International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu, Japan, 1996.

[60] J.Cortadella, M.Kishinevsky, A.Kondratyev, L.Lavagno and A.Yakovlev *Methodology and Tools for State Encoding in Asynchronous Circuit Synthesis* Proceedings of the 33rd ACM/IEEE Design Automation Conference, pp.63-66, 1996.

[61] J.Cortadella, M.Kishinevsky, L.Lavagno and A.Yakovlev *Synthesizing Petri Nets from State-based Methods* Proceedings of the International Conference on Computer Aided Design, pp.164-171, 1995. Also published as technical report UPC-DAC-95-09, Universitat Politècnica de Catalunya, 36pp., 1995.

[62] J.Cortadella, M.Kishinevsky, A.Kondratyev, L.Lavagno and A.Yakovlev *Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers* Proceedings of the 11th Conference on Design of Integrated Circuits and Systems, pp.205-210, 1996.

[63] E.Pastor, J.Cortadella, A.Kondratyev and O.Roig *Structural Methods for the Synthesis of Speed Independent Circuits* Proceedings of the European Design and Test Conference, March, 1996.

[64] O.Roig *The home page of VERSIFY* - http://www.ac.upc.es/vlsi/versify, VLSI CAD Group, Technical University of Catalunya. When this report was writen the last updated had occured in November, 1996.

[65] A.Yakovlev, M.Kishinevsky, A.Kondratyev and L.Lavagno *OR Causality: Modelling and Hardware Implementation* Proceedings of the 15th International Conference on Application and Theory of Petri Nets, J.Billington and W.Reisig (Eds.), Lecture Notes in Computer Science, vol.815, pp., 1994.

[66] A.Kondratyev, M.Kishinevsky, A.Taubin and S.Ten *Analysis of Petri Nets by Ordering Relations in Reduced Unfoldings* Formal Methods in System Design, 12(1)5-38, 1997.

[67] E.M.Sentovich, K.J.Singh, L.Lavagno, C.Moon, R.Murgai, A.Saldanha, H.Savoj, P.R.Stephan, R.K.Brayton and A.L.Sangiovanni-Vincentelli *SIS: A System for Sequential Circuit Synthesis* Technical Report UCB/ERL M92/41, University of California at Berkeley, May 1992.

[68] L.Lavagno *Synthesis and Testing of Bounded Wire Delay Asynchronous Circuits from Signal Transition Graphs* PhD thesis, University of California at Berkeley, Technical Report UCB/ERL M92/140, November 1992.

[69] L.Lavagno and A.L.Sangiovanni-Vincentelli *Algorithms for Synthesis and Testing of Asynchronous Circuits* Kluwer Academic Publishers, 1993.

[70] A.Yakovlev, A.Koelmans and L.Lavagno *High Level Modelling and Design of Asynchronous Interface Logic* IEEE Design and Test of Computers, pp.32-40, Spring 1995.

[71] A.V.Yakovlev and A.Petrov *Petri Nets and Parallel Bus Controller Design* Proceedings of the 11th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science, vol., pp.244-263, 1990.

[72] A.Semenov, A.Yakovlev, E.Pastor, M.Peña and J.Cortadella *Synthesis of Speed Independent Circuits from STG-unfolding Segment* Proceedings of the 34th ACM/IEEE Design Automation Conference, pp.16-21, 1997.

[73] W.L.A. de Oliveira *Proposta de uma Rede de Petri de Alto Nível Incluindo Temporização* Master Dissertation, São Paulo State University at Rio Preto, Department of Computer Science, 131pp., 1997.

[74] K.Jensen *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use* Springer Verlag, vol.1, pp.78-85, (ISBN:3540609431), 1992.

[75] M.R.K.Patel *Random Logic Circuit Implementation of Extended Timed Petri Nets* Microprocessing and Microprogramming, 30(1-5)313-320, North Holland, 1990.