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

MODELING AND VERIFICATION OF SEQUENTIAL CONTROL PATHS USING PETRI NETS

Werner ERHARD, Andreas REINSCH and Torsten SCHOBER

Friedrich-Schiller University Jena, Dept. of CS, Ernst-Abbe-Platz 1-4,
D-07740 Jena, Germany, <*erhard, reinsch, schober*>*@informatik.uni-jena.de*

Abstract. *In this paper a design methodology based on interpreted Petri nets is applied to the functional verification of complex sequential control paths. Starting from a Petri net model with free choice structure and control engineering interpreted (CEI) net components, the verification of structural and behavioural properties as well as functional simulation is performed. A series of analysis strategies is derived to enable an automatic functional verification of a Petri net model. From the results of functional verification a set of modeling guidelines could be provided.*

Keywords. *Petri net, digital design, functional verification, sequential control path.*

1. PETRI NET BASED DIGITAL SYSTEM DESIGN

To model digital systems using Petri nets, one has to make two important choices concerning the net specification and the net interpretation. The variety to model a system in a structural way is determined by the net specification. By means of ordinary Place/Transition Petri nets it is simple to express both sequential and concurrent events. In particular the structural sub class of Free-Choice Petri nets (FCPN) that covers state machine and marked graphs provides clear structural modeling facilities. A comprehensive overview regarding net classes and its properties can be found in [7]. On the other hand the chosen net specification should enable the application of analysis methods for structural and behavioural net analysis. In this respect FCPN are particularly suitable too. Many results of Petri net theory can be efficiently applied to analyze properties of FCPN.

1.1. Modeling digital systems using CEI Petri nets

Due to net interpretation, the components of a Petri net are assigned to components of a digital system. Hence every net interpretation of a Petri net creates a Petri net model. There exist several net interpretations in different technical domains that either map Petri net components to elementary digital components or small sub nets to digital modules [8, 6, 1]. In [5] Control Engineering Interpreted (CEI) Petri nets have been introduced. This net interpretation performs a direct mapping between a system of communicating finite state machines and a hardware realization as shown in figure 1.

A CEI Petri net N_{SIPN} is defined as a tuple $N_{SIPN} = (P, T, F_{pre}, F_{post}, M_0, G, X, Y, Q_t, Q_p)$. Every transition t_i is assigned to an AND-gate and every place p_i is assigned to a memory cell. Consequently, a marked place $p_i = (e_i, a_i)$ symbolizes an active memory cell. Transition activating and place marking are realized by two mappings Q_t and Q_p. So, in addition to the usual firing rule, transitions are activated by guards G_i. These guards arise from logically conjuncted input signals x_i and express signal processing in a digital system. Output signals a_i can be used to enable state transition $p_i \rightarrow p_j$

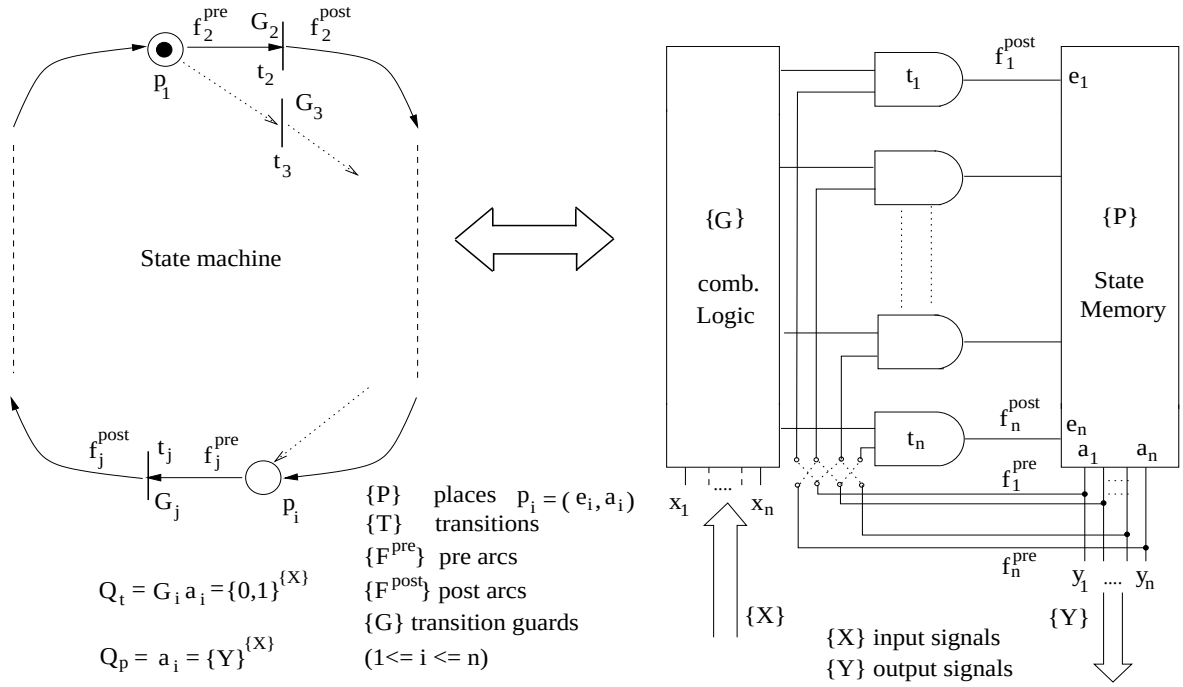

Figure 1: *CEI Petri net and its hardware realization*

or for output signal processing. In case of state transition, $a_i = 1$ is conjuncted with G_j at an arbitrary transition t_j. Then p_i gets low before p_j gets high and hence a new state is activated. Concurrent behaviour is represented by different state memory modules, each of sequential behaviour, communicating through shared transitions. The signal transition in a CEI Petri net shows that the token flow of the Petri net equivalently represents the signal flow of the digital system.

This is one of the major objectives in the hardware design methodology proposed in [2]. By means of CEI Petri nets with Free-Choice structure it is possible to model digital systems in a *transparent* way, such that system behaviour is equivalently reproduced by a simple structured Petri net model. Furthermore, behaviour and structure of the Petri net model is extensively analyzable if the Petri net model is retransformed into a FCPN. Therefore all transition guards G_i are eliminated.

To summarize at this point the approach of this work can be schematized as in figure 2. Starting with net specification FCPN and net interpretation CEI Petri net the modeling process can be performed. Afterwards the Petri net model is unannotated to a FCPN to enable net analyses. Results obtained by net analysis should be interpreted for the Petri net model to conclude a statement concerning its functional verification.

1.2. Petri net based design flow

Owing to high acceptance of commercial CAE-Systems and hardware description languages (HDL) it is not preferable to create a design flow that solely is based on Petri nets. Therefore in the design flow two entities enable a transformation between constructs of a HDL and Petri net components and vice versa. Consequently, there are several opportunities to specify a digital system. On top of figure 3 HDL input as well as graphical or textual Petri net input is proposed. For large designs high level Petri nets (HLPN) such as hierarchical Petri nets or colored Petri nets can be incorporated if it is possible to unfold the HLPN to a low level Petri net as FCPN. Once a Petri net model is created simulation and analysis methods can be applied to verify the design functionally. At this point Petri nets have two great advan-

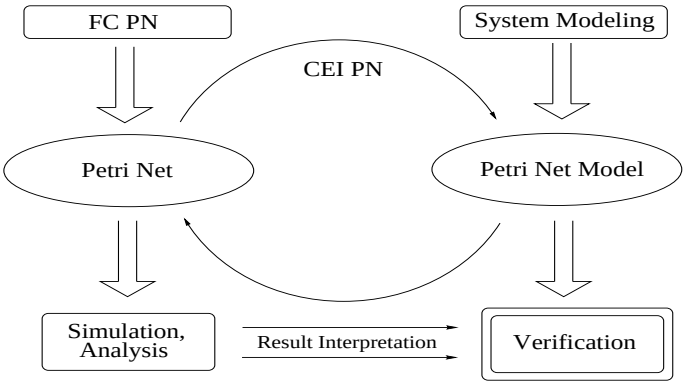

Figure 2: *Modeling and verification of digital systems using Petri nets*

tages. First because of its graphical concepts rough design errors can be detected easily by functional simulation. Beyond that an exhaustive analysis of structural and behavioural properties leads to a formal design verification rather than simulation of test pattern. A functionally verified design is transformed to dedicated HDL constructs to enable logic synthesis by means of conventional CAE tools. After logic synthesis simulation and analysis methods can be applied again to simulate and verify the timing behaviour of the design. Now deterministic and stochastic timed Petri nets are applied.
Thus it is shown that the design flow is based on Petri nets but nevertheless is embedded in a conventional design flow. To put the new design steps into practice the tool development environment "Petri Net Kernel (PNK)" [4] and the "Integrated Net Analyzer (INA)" [9] are used.

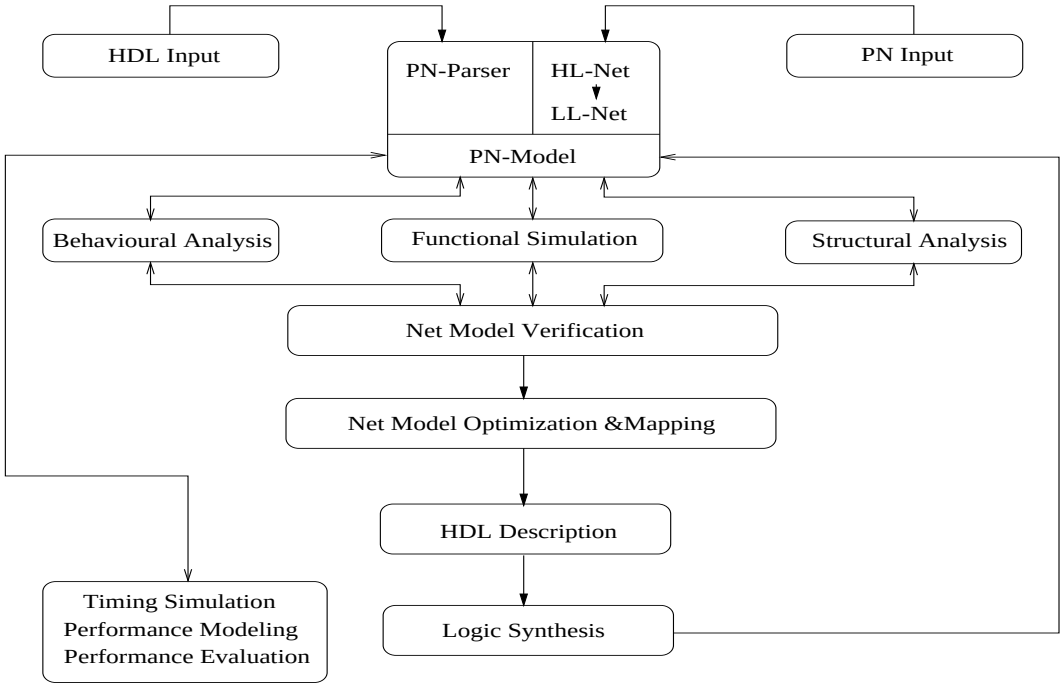

Figure 3: *Design flow for Petri net based digital system design*

2. FUNCTIONAL VERIFICATION OF PETRI NET MODELS

For functional verification of a modeled digital system a set of properties are studied by means of Petri net analysis. The analysis results are interpreted as properties of Petri net model. In a more practical way it is necessary to propose some requirements and goals that must be obtained for functional verification. Then a series of analysis strategies is derived to enable an automatic detection, localization and removal of modeling errors. As an outcome of applied analysis strategies some modeling guidelines could be derived. Adhering to these modeling guidelines enables approaching a functionally correct design already at early modeling cycles. In this work analysis strategies and modeling guidelines for functional verification of sequential control paths are suggested.

2.1. Requirements and Goals

Requirements for functional verification are expressed as Petri net properties of the analyzed Petri net. Net analysis is performed with the aid of tool INA as structural analysis and as reachability analysis. At first Petri net analysis has to ensure boundness and especially 1-boundness. In the Petri net model boundness determines that the modeled design has a finite state space. Control paths with an infinite number of states are not close to reality. Moreover 1-boundness is claimed, since the digital system should be modeled in a transparent way. If every place of th Petri net contains at most one token, logical values “high” and “low” are distinguishable for the memory cells that are represented by places. An unbounded Petri net is not analyzable any further. Liveness is the next necessary Petri net property for functional verification and it is interpreted as the capability to perform a state transition in any case. Every transition of the Petri net can be fired at any reachable marking. So if liveness is preserved the Petri net and thus the Petri net model are not deadlocked. If in a live Petri net the initial state is reachable then the Petri net is reversible. To reflect the structure of a sequential control path the Petri net should have state machine structure. In this case not any transition of the Petri net is shared. Every transition has exactly one predecessor place and one successor place. Therefore in a state machine generation and consumption of tokens is avoided. Complex sequential control paths such as control path of a sequential microprocessor consist of a system of strongly connected state machines. That includes decomposability into small partial nets with state machine structure. When a Petri net is 1-bounded, live and it has state machine structure then a very transparent Petri net model has been created. Every state in the Petri net model can be assigned to a state of the control path. Thus the reachability tree is rather small and represents exactly the number of control states. From the implementation point of view this state encoding is called one hot encoding.

Places with more than one successor transition generate conflict situations. If several post-transitions of a marked place are enabled and one transition fires, then all other transitions are disabled. Hence the Petri net is not persistent and also it is not predictable what transition will fire. Transition guards are able to solve conflicts because it represents an additional firing condition that is required to perform a state transition. So transitions in conflict can get unique using transition guards and no behaviour ambiguity remains. When a pre-place of a transition also appears as its post-place then there is a self loop in the Petri net. Self loops can give a structural expression to model external signal processing distinctly. It has to be clarified in the modeling process if any and which self loops are desired to emphasize external signal processing. Thus self loops can be detected and assigned to that situations and others are marked as modeling errors and should be removed.

2.2. Analysis Strategies

Table 1 summarizes all derived analyses strategies regarding detected modeling errors and affected Petri net properties. An automatic verification of Petri net models using $S1 \ldots S9$ can be performed according to figure 4. Using PNK and INA the analysis is efficiently implementable. It is supposed, that at “Start” a

reachability tree is derived or at least a part of it to determine boundness or unboundness of the analyzed Petri net. Strategy $S1$ detects shared transitions with more than one post-place or transitions without pre-place that cause unbounded places and hence an unbounded Petri net.

Strategy S1:

1. If Petri net N is unbounded, then
   (a) Determine all shared transitions t_i, $\mid t_i\bullet \mid > 1$ by means of $\bullet p_j$, $\forall p_j \in P \Rightarrow$ generate list $\{t_P^i\}$.
   (b) Determine transitions without a pre-place Ft_i0 by evaluating $\bullet t_i$, $\forall t_i \in T$.
2. Check $t_i = \bullet p_j$, $\forall p_j \in P$: if $t_i \in \{t_P^i\} \vee t_i = Ft_i0 \Rightarrow p_j$ is unbounded. Pre-transition t_i of place p_j produces tokens if $t_i \in \{t_P^i\}$ or $t_i = Ft_i0$ and hence unboundness of p_j is caused by t_i.

$S2$ and $S3$ are applied to detect and localize dead transitions that caused by lack of strong connectedness or by shared transitions with more than one pre-place. It is possible that the analyzed Petri net is 1-bounded and live but it shows no state machine structure. In this case all generated tokens are consumed within a marked graph, that is a Petri net structure in which no place is shared. Strategy $S4$ is applied to localize such shared transitions. By means of strategy $S5 \ldots S7$ dead transitions are detected and localized caused by one sided nodes as mentioned in the table. In the last two analysis strategies $S8$ and $S9$ transition guards are used to interpret conflicts and self loops within the Petri net model. A detailed description of all analysis strategies and its application within a case study is given in [3].

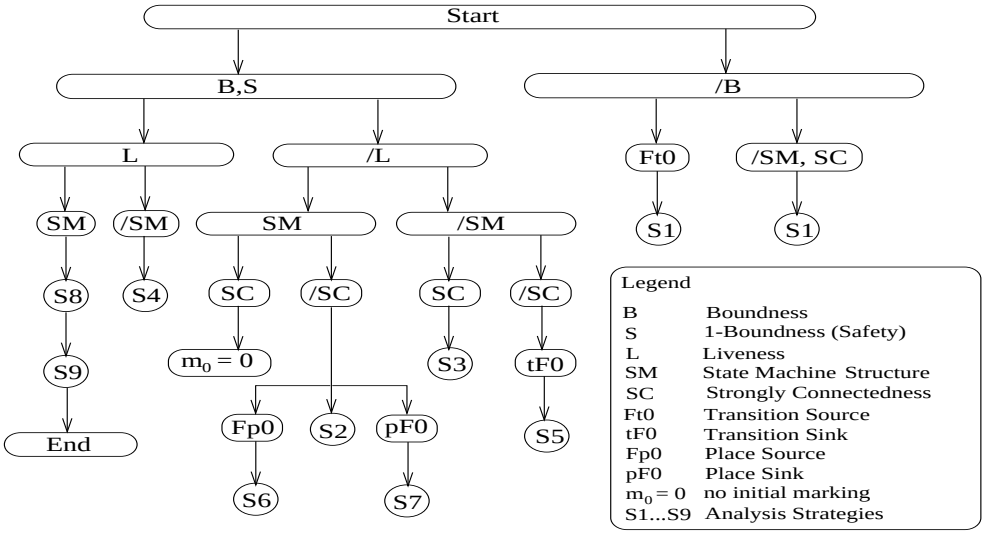

Figure 4: *Application scheme of derived analysis strategies*

According to the proposed analysis strategies it is possible to derive modeling guidelines that affect the modeling process and assist the designer to create a system model reflecting the desired functionality.

Modeling Guidelines:

1. Avoidance of shared transitions.
2. Avoidance of one-sided nodes.
3. Ensuring strong connectedness.
4. Removal of conflicts using transition guards:
   (a) All post-transitions of a shared place are provided with guards.
   (b) Every post-transition of a shared place is provided with an unique guard.

analysis strategy	modeling error	affected property
S1	generation of tokens or transition without pre-place	boundness
S2	not strongly connected	liveness
S3	consumption of tokens	state machine structure, liveness
S4	generation and consumption of tokens	state machine structure
S5	transition without post-place	state machine structure, liveness
S6	place without pre-transition	liveness
S7	place without post-transition	liveness
S8	self loops	pureness
S9	conflicts	static & dynamic conflict free

Table 1: *Summary of analysis strategies*

3. CONCLUSIONS

This work introduces a Petri net based hardware design methodology for modeling and functional verification of digital systems. System modeling is performed by means of Free-Choice Petri nets (FCPN) and control engineering interpreted (CEI) Petri nets. Using Petri net analysis techniques functional verification of Petri net models is obtained by analysis of Petri net properties and a suitable interpretation concerning the Petri net model. It is shown that a Petri net based design flow can be embedded in a conventional hardware design flow. For the functional verification of sequential control paths a series of analysis strategies is provided. Using these analysis strategies it is possible to detect and localize modeling errors automatically. Finally a set of modeling guidelines is determined to avoid modeling errors at early design cycles.

References

[1] J. Cortadella et. al. Hardware and Petri Nets: Application to Asynchronous Circuit Design. *Proc. of the* 21^{st} *ICATPN, LNCS 1825, Springer Verlag, 2000.*

[2] W. Erhard, A. Reinsch und T. Schober. Petri-Netz-basierter Entwurf asynchroner rekonfigurierbarer Systeme. *Tagungsband 15. GI/ITG-Fachtagung - Architektur von Rechensystemen ARCS'99, 4.-7.10.1999, Jena*.

[3] W. Erhard, A. Reinsch and T. Schober. Formale Verifikation sequentieller Kontrollpfade mit Petrinetzen. *Berichte zur Rechnerarchitektur, Band 7, Nr. 2, 2001*.

[4] E. Kindler and M. Weber. The Petri Net Kernel: An Infrastructure for Building Petri Net Tools. *Proc. of the* 20^{th} *ICATPN, LNCS 1643, Springer Verlag, 1999.*

[5] R. König und L. Quäck. *Petri-Netze in der Steuerungstechnik.* Verlag Technik Berlin, 1988.

[6] D. Misunas. Petri-Nets and Speed-Independent Design. *Communication of the ACM, 16(8):474-479, 1973.*

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

[8] S. Patil. Coordination of Asynchronous Events. *ScD Thesis, Dept. of Elec. Eng., MIT, Cambridge, Mass., May 1970.*

[9] P. H. Starke and S. Roch. Integrated Net Analyzer INA, Version 2.2. *LS Automaten und Systemtheorie, Institut für Informatik, Humboldt Universität zu Berlin, Dezember 2000.*