# Petri Nets tools integration through Eclipse

Adilson Arcoverde, Jr.
Centro de Informática
Universidade Federal de Pernambuco
aoaj@cin.ufpe.br

Gabriel Alves Jr.
Centro de Informática
Universidade Federal de Pernambuco
gaaj@cin.ufpe.br

Ricardo Lima
Departamento de Sistemas Computacionais
Universidade de Pernambuco
ricardo@dsc.upe.br

## ABSTRACT

The lack of a standard file format to represent Petri net limits the reuse of models among Petri net tools. In order to reduce the impact of this limitation the Petri Net Markup Language (PNML) has been proposed. PNML is an interchange file format for Petri nets based on XML. This paper presents a framework, called EZPetri, based on PNML. EZPetri is a *perspective* and a set of plug-ins of the Eclipse platform. The union of Eclipse and PNML has demonstrated itself to be an effective instrument for integrating Petri net tools and applications. The paper discusses the principles of the EZPetri, and presents one application integrated into the framework: *Power Consumption Analysis Framework*. Such application has been developed with no knowledge about EZPetri. This is a demonstration of the integration facilities provided by the framework. EZPetri is thus a fertile ground for combining existing Petri net tools and applications into a single environment, offering the Petri net community a new perspective of integration.

## 1. INTRODUCTION

Petri net is a powerful specification language useful for modelling concurrent, asynchronous, distributed, parallel, non deterministic, and/or stochastic systems [16]. It is also a formal specification technique with powerful methods for qualitative and quantitative analysis [15]. Since their introduction by C. A. Petri in 1960, Petri nets have been widely applied in many fields of science and industry.

There are several Petri net tools available for specific types of net (high-level[7], timed[19], stochastic[5], etc.). The lack of a standard has forced Petri net tool designers to create their own file format. Thereby, a model created through a Petri net tool cannot be read by other tools. This assertion is true even for tools supporting the same Petri net type.

This situation has motivated the Petri net community toward creating a Petri net interchange format. In order to define a single XML-based file for any Petri net type, many proposals were presented during the International Conference on Application and Theory of Petri nets in 2000. The most prominent of them was the Petri Net Markup Language (PNML)[8].

The development of a computational tool to support PNML would be the next step. Such a tool is supposed to provide facilities for integrating existing Petri net tools through importing/exporting functions.

This paper presents a new Petri net PNML-based tool, called EZPetri. It takes advantage of the plug-in technology offered by the Eclipse platform[13].

## 2. THE EZPETRI PROJECT

EZPetri is an extendable Eclipse-based tool suite that supports editing Petri nets, as well as importing/exporting Petri nets from/to different Petri net tools. It takes advantage of the plug-in technology of Eclipse to couple existing Petri net tools and to implement new functionalities.

For instance, one may decide to build a new analysis method for Time Petri nets. Instead of implementing a new graphical interface, the developer may reuse all features already defined, such as editors, compilers, etc, and maintain the focus on what really matters: the new analysis method.

PNML forms the kernel of EZPetri. It means that any Petri net type may be represented through the PNML format in the EZPetri environment. Thus, it glues together the integration facilities provided by Eclipse with the PNML interchange format.

The project contributes to reducing the gap between members of the Petri net community who use different Petri net types, tools and file formats. Moreover, EZPetri improves productivity in the development of new products by offering several functionalities on a single development platform. Figure 1 depicts the current state of the EZPetri project. It also depicts the adopted design strategy. Each plug-in accesses a single and shared PNML file.

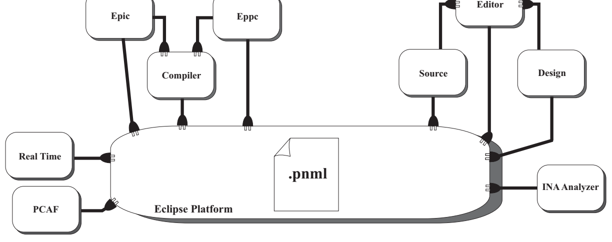


Figure 1: EZPetri Architecture

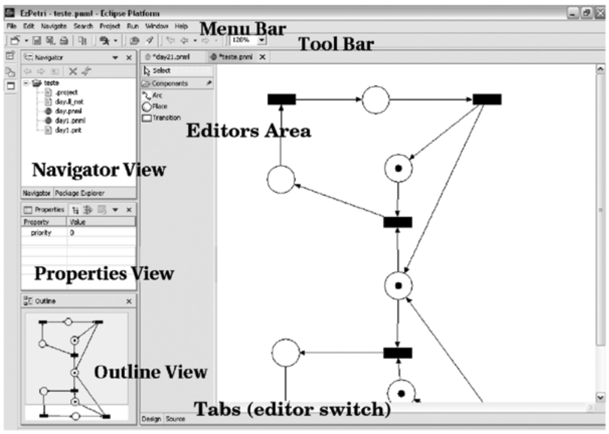


**Figure 2: EZPetri Views and Editors**

## 3. EZPETRI PERSPECTIVE

The design of the EZPetri perspective took into account a problem known as *loss of context.* The problem occurs when a user does not know where they are in the *User Interface* (UI) or where to go to complete a task. A frequent cause of the problem is the inclusion of an excessive number of *views* and *editors* in the perspective. For instance, an object action may differ between two distinct *views* or menu items may vary with *views.* Therefore, the EZPetri perspective reduced the number of *views* and *editors* as much as possible. This decision yielded an intuitive (consistent) platform. As can be seen in Figure 2, the EZPetri perspective is composed of three *views*: *navigator view*, *properties view*, and *outline view.* There is also the *editor area*, the *menu bar* and *tool bar.*

The *menu bar* includes functionalities found in many commercial tools: to create new files, help, arrange views and so on. The *tool bar* works as a shortcut for functionalities most used on the menu bar. Some options of these bars are context-sensitive and are available only when a specific view or editor is focused.

The *navigator view* enables users to show the workspace area. It is useful to manage projects, folders and files. In addition to ordinary tasks, such as creating a new folder or invoking an editor to modify a file, specific functionalities are available. For example, the user may right-click a PNML file, choose the *Compile To* option, and translate PNML into the file format of a specific Petri net tool.

The *property view* is useful to show and modify attributes of selected objects. For instance, one may select a place and edit its name, marking, etc.

The *outline view* displays attributes of specific files. A tree structure is usually adopted in the outline view to represent attributes. A non-conventional usage of outline view is provided by the EZPetri editor. It presents an overview of the whole Petri net model (see Figure 2), when the EZPetri graphical editor is activated. Such a view is useful for large models. The user may choose the part of the specification to be presented in the *editor area* by clicking on the corresponding point of the outline view.

The *editor area* is a large blank box where specific file formats are manipulated. EZPetri provides a *multipage editor* which is the parent of several editors. It uses tabs to switch between different child editors, i.e., source (PNML), and the graphical editor. When the user presses over *Design* tab in the parent multipage editor, the graphical editor will be displayed.

All editors in the *multipage editor* are supposed to implement a synchronization interface. Such an interface defines methods to translate the editor contents into PNML, and vice-versa. When one selects an editor through the *multipage editor* tab, the parent editor executes the method in the corresponding interface to synchronize the contents of the currently opened editor and the requested editor. In order to not compromise performance, synchronization is performed only when either users switch between editors, or the work is saved.

## 4. THE MODELLING ENVIRONMENT

EZPetri contains a set of plug-ins for the graphical edition of three types of Petri nets (place-transition Petri nets, time Petri nets, and stochastic Petri nets). The type information in the PNML tag is used by the editor to identify the Petri net type.

These editors provide functionalities found in many Petri net graphical editors: drawing by select-and-click, drag and drop, resize, undo, redo, zoom in, zoom out, select all, select all of the same type, etc. The editor gives flexibility for changing the source/target of an arc by dragging it to another valid source/target. Some functionalities are enabled in specific situations. For instance, *select all of the same type* is enabled only when one transition or place, but never both at the same time, is selected.

The editor includes an *overview* of the Petri net model in the *outline* view. The overview is a small view with a picture of the whole Petri net model. The corresponding area clicked in the overview becomes visible in the *editor area.* This innovative functionality is useful for large projects. It allows users to rapidly move to a specific point of the model by clicking on the corresponding area of the *outline* view. It avoid the usage of the concept of *page*, adopted by some tools (i.e. PNK). *pages* make the model difficult to understand and edit, since designers are supposed to manage several windows to work with a single net.

*Alignment* functionalities are useful to organize objects in the model. Selecting two or more objects and right-clicking over them, shows a popup menu that provides a number of alignment functionalities. For instance, *Align Left* aligns the left side of all selected objects with the left side of the last selected objects. This includes object labels. The last selected object is identified by a black border. Figure 3 exemplifies the usage of the alignment functionality.

Similarly, *same size* functionality applies the same width and height to all selected object. The width and height used will be that of the last object selected.

The graphical editor was developed using the Eclipse plug-in *Graphical Editing Framework*(GEF)[6]. GEF assists the task of building several GUIs.

## 5. COMPILERS

EZPetri compilers are used to translate from a specific description format into PNML, or vice-versa. By translating from a file format into PNML, it is possible to reuse nets modelled in different tools. The inverse process is useful to export a PNML model to other Petri net tools. Notice that is possible to convert from a given file format into another

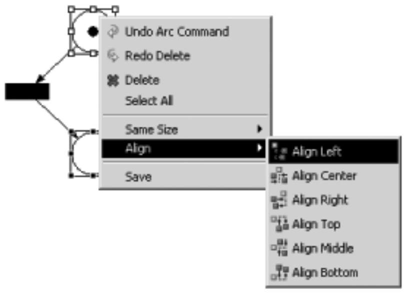


**Figure 3: Aligning left two places of the net**

using PNML as an intermediated representation. The Document Object Model (DOM) [20] was adopted to manipulate the PNML file.

To guarantee compilers reuse, they were developed and packaged in two different modules: *core* and *ui*. The *core* provides translation functionality. It compiles from a specific format into PNML, and vice-versa. The *ui* provides a popup menu option. When requested by the user, this menu invokes the compiling feature provided by the *core*.

Two compilers have been developed: *EZPetri PNML INA Compiler* (Epic) and *EZPetri PNML PEP Compiler* (Eppc). Currently, both Epic and Eppc support Place/Transition Petri nets. Additionally, Epic supports Time Petri nets.

Epic converts the *Integrated Net Analyzer* (INA) [17] file format into PNML, and vice-versa. INA is a well known Petri net tool It provides a rich set of analysis techniques.

Eppc translates the *Programming Environment based on Petri nets* (PEP) [1] formats into PNML, and vice-versa. PEP is a comprehensive set of modelling, compilation, simulation and verification components, linked together within a Tcl/Tk graphical user interface.

## 6. INA INTEGRATION

As discussed before, INA is one of the most used analysis tool for Petri nets. In this section we will present the way this tool was integrated into EZPetri.

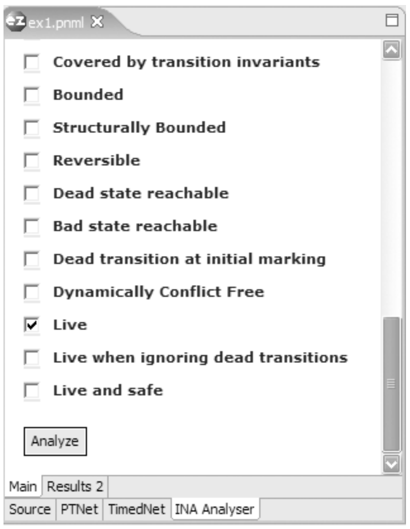
INA is a shell tool and the interaction with this program is made through menu options entered by command line. Figure 4 shows a piece of INA initial dialog. For instance, if A is pressed at this point, the loaded net will be analyzed.

```
Do You want to
   edit ? .......................................E
   fire ? .......................................F
   analyse ? ....................................A
   reduce ? .....................................R
   read the session report ? ....................S
   delete the session report ? ..................D
   change options ? .............................O
   quit ? .......................................Q
choice >
```

**Figure 4: A piece of INA's initial dialog**

Two important INA files were used for the integration purpose. The COMMAND.ina file is generated when the user decides to quit INA. All the commands entered during an INA session are stored into this file. If this file exists when INA starts, the question *Same procedure as last time? Y/N* is presented. Answering $<Y>$, repeats the execution of the commands entered the previous INA session. Another file generated during an INA session is named SESSION.ina. It contains every analysis result or deduction obtained during the session.

Currently, some formal analysis methods provided by INA are supported directly from EZPetri. Choosing the analysis tab in the EZPetri editor, it is possible to analyze some formal properties of the modelled net (see Figure 6).


ex1.pnml

- [ ] Covered by transition invariants
- [ ] Bounded
- [ ] Structurally Bounded
- [ ] Reversible
- [ ] Dead state reachable
- [ ] Bad state reachable
- [ ] Dead transition at initial marking
- [ ] Dynamically Conflict Free
- [x] Live
- [ ] Live when ignoring dead transitions
- [ ] Live and safe

**Figure 5: EZPetri INA analyzer**

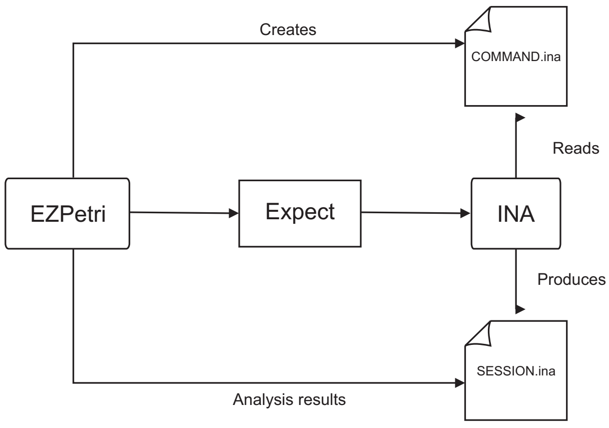


**Figure 6: EZPetri INA analyzer**

In order to invoke the analysis functions provided by INA, EZPetri explicitly generates the `COMMAND.ina` file. The set of commands included in the file depends on the properties selected in the EZPetri INA analyzer (see Figure 6). The **Expect**[12], which is employed to simulate keyboards interactions, executes the INA program[1]. The `SESSION.ina` file produced by INA analysis is shown to EZPetri user.

Figure 7 depicts the analysis of *liveness* and *ordinary* properties of net presented in Figure **??**.

[1]The Expect is executed using the *java.lang.Runtime* and *java.lang.Process* classes.

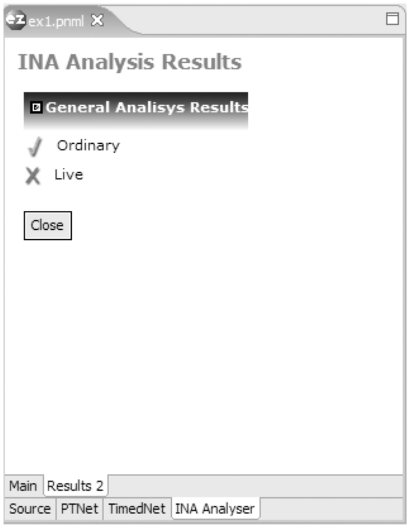


Figure 7: Result of the EZPetri INA analyzer

## 7. POWER CONSUMPTION ANALYSIS FRAMEWORK - PCAF

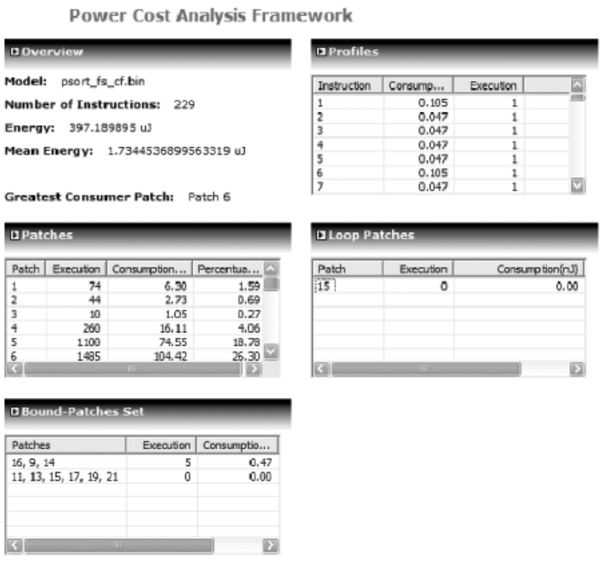
In power critical embedded systems, energy consumption may be regarded to the processor and to specific hardware device. Despite of processor hardware optimization, processor consumption is affected by the software dynamic behavior [18], meaning that the software power analysis is crucial. In order to improve designs, methods for design space exploration play an important. For instance, they are useful to improve the performance and energy consumption. Many embedded computing applications are power critical. Some examples are portable medical instruments, notebook computers, personal digital assistant and cellular phones.

A new approach based on Colored Petri Net for analyzing software power cost was proposed in [9]. This work explores Colored Petri Nets as a formal description language. The simulation mechanism is used to capture behavioral patterns that characterize energy consumption. The atomic behavior associated to consumption is the processor instruction set. During the program execution flow, each instruction changes the internal processor context. The program flow is modelled as a Colored Petri Net, where each processor state is modelled as a place, the internal context as a data structure within a token, and instructions as transitions that processes this token. The instruction behavior is defined by an associated CPN-ML code (a Standard ML Language subset) [10].

Some CPN processor architecture models [3][2] are based on the architectural/RTL hardware model, i.e. based on its internal functional units (pipeline stages, Icache and Dcache). In contrast, this approach models processor only by its instruction set. Models based on architectural/RTL hardware have to be feed with very detailed hardware consumption data related with implementation technologies. Such information is not often available to the system designer [14]. On other hand, instruction-level power model can be implemented through simple physical measures [11]. The proposed CPN model explicitly represents control and data dependencies, hence allowing mapping consumption features to software structures.

In order to integrate this work to EZPetri, the *Power Cost Analysis Framework* (PCAF) has been developed. It implements functions such as codes entities identification (*Patches and Clusters*) and consumption analysis. The result generated by the analysis is shown in a GUI as charts and tables. Figure 8 exemplifies a PCAF result. Additionally, a portable document format (pdf) file containing the result is generated.

To reach the result, several operations are executed by this framework. It starts by loading the machine-code file, which is translated into a CPN model. The next step is to send the model to the CNP-Tools proxy. This is performed through a TCP/IP channel. Then, the proxy loads the model in the CPN-Tools. A TCP/IP connection is established between the PCAF and the proxy using the Comms/CPN[4]. The CPN-Tools analyzes the model and generates the energy consumption information. Finally, the proxy sends such information to PCAF, which mounts the charts and tables presented to the user.


Power Cost Analysis Framework

Overview

**Model:** psort_fs_cf.bin

**Number of Instructions:** 229

**Energy:** 397.189895 uJ

**Mean Energy:** 1.7344536899563319 uJ

**Greatest Consumer Patch:** Patch 6

Profiles

| Instruction | Consump... | Execution |
|---|---|---|
| 1 | 0.105 | 1 |
| 2 | 0.047 | 1 |
| 3 | 0.047 | 1 |
| 4 | 0.047 | 1 |
| 5 | 0.047 | 1 |
| 6 | 0.105 | 1 |
| 7 | 0.047 | 1 |

Patches

| Patch | Execution | Consumption... | Percentua... |
|---|---|---|---|
| 1 | 74 | 6.30 | 1.59 |
| 2 | 44 | 2.73 | 0.69 |
| 3 | 10 | 1.05 | 0.27 |
| 4 | 260 | 16.11 | 4.06 |
| 5 | 1100 | 74.55 | 18.78 |
| 6 | 1485 | 104.42 | 26.30 |

Loop Patches

| Patch | Execution | Consumption(nJ) |
|---|---|---|
| 15 | 0 | 0.00 |

Bound-Patches Set

| Patches | Execution | Consumptio... |
|---|---|---|
| 16, 9, 14 | 5 | 0.47 |
| 11, 13, 15, 17, 19, 21 | 0 | 0.00 |

Figure 8: Tables presented in the user interface

## 8. CONCLUSIONS

The paper has described a framework for the development of Petri net applications and integrating existing Petri net tools. The framework joins together the PNML interchange file format with extension facilities provided by the Eclipse platform. The combination of these technologies has demonstrated itself to be an effective instrument for integrating Petri net tools and applications.

A description of the EZPetri architecture and its implementation have been provided. The paper has detailed the EZPetri perspective, including their *editors* and *views*. The EZPetri modelling environment has also been briefly discussed.

The successful and quick integration of two existing Petri net tools, namely INA[17] and PEP[1], into EZPetri has demonstrated the potential of the framework for importing/exporting Petri net file formats of other Petri net tools. It must be highlighted that the PEP and INA plug-ins required less than sixteen hours (each) to be implemented.

Additionally, a graphical frontend for the INA tool has been implemented. Using this INA frontend is possible to invoke several formal analysis methods provided by INA. This closer integration between INA and EZPetri increases designers productivity.

We have demonstrated that EZPetri can be extended to incorporate Petri net applications. In particular, we have described a Petri net based application, named *Power Cost Analysis Framework - PCAF* developed with no knowledge of EZPetri and their integration into the framework.

We believe that the EZPetri framework offers more than just another tool for the Petri net community. It offers them a perspective of real integration. Through collaboration, corporate professionals, researchers, members of academia, and individual developers can further the goal of producing interoperable Petri net based products and offerings.

## 9. ADDITIONAL AUTHORS

Additional authors:
*Paulo Maciel* (Centro de Informática - Universidade Federal de Pernambuco, email: prmm@cin.ufpe.br)
*Meuse de Oliveira Jr.* (Centro de Informática - Universidade Federal de Pernambuco, email: mnoj@cin.ufpe.br)

## 10. REFERENCES

[1] B. Best and B. Grahlmann. Pep - more than a petri net tool. In *LCNS*, volume 1055, pages 397–401. Springer-Verlag, 1996.

[2] F. Burns, A. Koelmans, and A. Yakovlev. Wcet analysis of superscalar processor using simulation with coloured petri nets. *International Journal of Time-Critical Computing Systems,.*

[3] F. Burns, A. Koelmans, and A. Yakovlev. Modelling of superscala processor architectures with design/CPN. In Jensen, K., editor, *Daimi PB-532: Workshop on Practical Use of Coloured Petri Nets and Design/CPN, Aarhus, Denmark, 10-12 June 1998*, pages 15–30. Aarhus University, 1998.

[4] G. Gallasch and L. M. Kristensen. Comms/CPN: A communication infrastructure for external communication with design/CPN. In *3rd Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'01) / Kurt Jensen (Ed.)*, pages 75–90. DAIMI PB-554, Aarhus University, Aug. 2001. InternalNote: Submitted by: hr.

[5] P. Haas. *Stochastic Petri Nets: Modelling, Stability, Simulation.* Springer-Verlag, New York, 2002.

[6] R. Hudson. *How to get started with the GEF.* http://www.eclipse.org/gef, 2003.

[7] K. Jensen. An introduction to high-level petri nets. In *Int. Symp. on Circuits and Systems, Proceedings, Kyoto, Japan*, volume 2, pages 723–726, New York, 1985. IEEE. NewsletterInfo: 25.

[8] M. Jüngel, E. Kindler, and M. Weber. Towards a generic interchange format for petri nets. *In 21st International Conference on Application and Theory of Petri Nets Aarhus.*, June 26-30 2000.

[9] M. N. Junior, P. Maciel, R. Barreto, and F. Carvalho. Towards a software power cost analysis framework using colored petri net. In *PATMOS 2004*, volume 3254, pages 362–371. LNCS Kluwer Academic Pubishers, September 2004.

[10] L. Kristensen, S. Christensen, and K. Jensen. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer: Special section on coloured Petri nets*, 2(2):98–132, 1998.

[11] T. Laopoulos, P. Neofotistos, C. Kosmatopoulos, and S. Nikolaidis. Current variations measurements for the estimation of software-related power consumption. *IEEE Instrumentation and Measurement Technology Conference*, May 2002.

[12] D. Libes. Expect: Scripts for controlling interactive processes. *Computing Systems*, (4), 1991.

[13] D. Lipcoll, D. Lawrie, and A. Sameh. *Eclipse Platform Technical Overview.* Object Technology International Inc., July 2001.

[14] S. D. Marcello Lajolo, Anand Raghunathan and L. Lavagno. Cosimulation-based power estimation for syste-on-chip design. *IEEE Transactions on Very Large Scale Integration (VLSI) System*, 10(3):253–266, June 2002.

[15] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[16] G. Rozemberg and W. Reisig. Informal introduction to petri nets. 1998.

[17] P. Starke and S. Roch. *INA - Integrated Net Analyzer - Version 2.2.* Humbolt Universität zu Berlin - Institut für Informatik, 1999.

[18] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Transactions on Very Large Scale Integration Systems*, 2(4):437–445, December 1994.

[19] J. Wang. *Timed Petri Nets, Theory and Application.* Kluwer Academic Publishers, 1998.

[20] World Wide Web Consortium (W3C) (ed.), http://www.w3.org/DOM. *Document Object Model*, 2000.