University of Southampton

Electronics and Computer Science

Nine Month Report:

Petri Net oriented modelling and synthesis for Embedded Systems

Mauricio Varea

June, 2000

Summary

The aim of this research is to investigate and develop a modelling technique for the internal representation of embedded system specifications. The first part of the report reviews briefly the various modelling techniques that have been reported in the literature and conclude that the use of Petri Nets is a suitable technique for systems with a high degree of communication, as is the case of Hardware/Software embedded systems. The second part of the report introduces the basic principles, definitions and execution rules of a new type of modelling technique based on Petri Nets called NOEMI, that stands for *Petri Net Oriented Embedded Systems Modelling*. This technique was formalised in a paper submitted to Design, Automation and Test in Europe (DATE) - *2001* conference, which is included at the end of this report.
A brief comparison with some recently introduced Petri Net oriented techniques is done, in order to show that the technique is at least as efficient as state of the art. Finally, a number of areas for future work are identified.

Chapter 1

Introduction

Traditional methodologies that have been applied to embedded systems were built in a separately development of each part, hardware and software, and a final integration based on *ad hoc* methods. This increases mainly two financial parameters of the final product, *time-to-market* and *cost*. Using hardware/software co-design the designer exploits the flexible boundary that exist between *hardware* and *software* in a electronic system design (ESD), through a concurrent design of both parts.

Currently most behavioural system specifications are derived from the corresponding algorithmic descriptions of the system functionality, usually described in a procedural language like C or Pascal. Consequently, hardware behavioural descriptions tend to use a procedural language (e.g. VHDL [49], Verilog [59], etc.) However, when describing hardware in a procedural language, one is often faced with the difficulty of representing an essentially concurrently-executing set of operations in a linear code. Therefore, the hardware for embedded controllers is more appropriately represented by an internal design representation model, usually constructed in such a way that concurrently-executing operations are well represented.

Based on [46], Wolf divided ESD tasks into four domains [62]: *partitioning*, *allocating*, *scheduling* and *mapping*. The internal representation for each of this tasks falls into the following taxonomy [28], given by Gajski *et al*:

- **State-oriented models**. Suitable for systems where the temporal behaviour is the most important aspect of the design.
- **Activity-oriented models**. Useful when data passes through a set of transformations.
- **Structure-oriented models**. Describes a system's physical modules and interconnection between them.
- **Data-oriented models**. A model where the relations between data attributes is the main parameter, rather than its functionality.
- **Heterogeneous models**. Whenever differents views are needed (or it does not fit in all the above descriptions).

Edwards *et al* gives a good introduction to the design of embedded systems and shows that embedded systems are inherently heterogeneous [21]. Therefore, among the classification given

above, we found that the heterogeneous model is the most suitable for embedded system design representation. Most work in ESD is turning into this approach. We belive that a good ESD framework is not reached just by speeding up the algorithms involved in the process (e.g. better scheduling or partitioning). Moreover, the *embedded system* has to be naturally represented by the internal design representation (IDR). Therefore, the research programme has been focused on the creation of an efficient IDR, rather than the implementation of optimisation algorithms. The core of the IDR we have developed is a natural successor of Petri Nets, which preserves all its properties and enhance the data orientation in order to make it suitable for ESD.

A new design representation model for *Petri Net Oriented Embedded Systems Modelling* (NOEMI) is presented here. Section 1.1 gives a short review of Petri Nets. Chapter 2 puts this work in the context of related efforts in the research community. The formal definition of the NOEMI model is given in the *DATE-2001* paper. However, in Chapter 3 we present some additional examples and summarise our contributions. Also, directions for future trends are outlined.

1.1 Petri Nets

Petri Net (PN) is a graphical and mathematical modelling tool applicable to many systems. This technique has evolved from the early work of Carl Adam Petri [55] where he formulated the basis for a theory of communication among asynchronous components. In 1981, Peterson wrote the first book about Petri Nets [54]. Nowadays they are widely used by both, practitioners and theoreticians. A tutorial-review paper from Murata [48] provides a very good introduction to the field.

A Petri net is a directed, weighted, bipartite graph consisting of two kind of nodes, called *places* and *transitions*, where arcs are either from a place to a transition or from a transition to a place. In graphical representation, places are drawn as circles and transitions as bars. Arcs are labeled with their weights (positive integers), where a k-weighted arc is equivalent to have k single arcs connected in parallel [48]. Formally:

Definition 1 *The classical PN structure is a four-tuple* $N = \langle P, T, F, W \rangle$ *where:*

$P = \{p_1, p_2, \cdots, p_n\}$ *is a finite set of places,* $n \geq 0$.

$T = \{t_1, t_2, \cdots, t_m\}$ *is a finite set of transitions,* $m \geq 0$.

$F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs describing the control flow relation.*

$W : 2^F \rightarrow \mathbb{N}$ *is a weight function.*

A *marking* is an assignment of *tokens* onto the places of a PN. Tokens are represented graphically by small dots (i.e. •) placed inside the circles[1]. A marking of the PN is equivalent to a global state in the modelled system and a change in the marking corresponds to a state transition. These changes in the marking are produced during the *execution* of a PN.

As mentioned above, a Petri net executes by means of firing transitions. A transition can *fire* only if it is *enabled* - that is, if each of its input places has at least one token.

[1] which represents the places

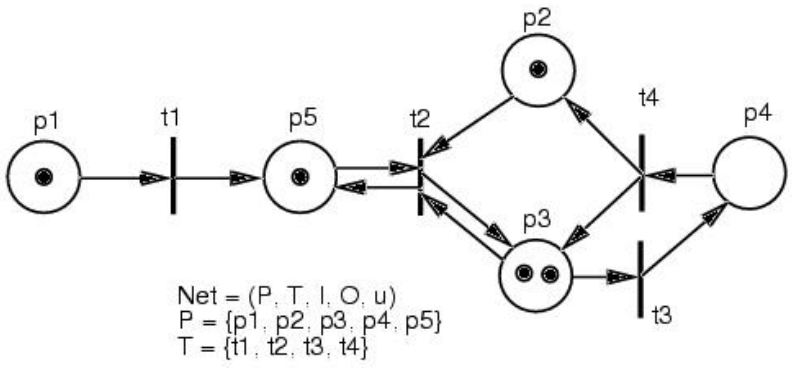


Figure 1.1: A Petri net example. [28]

In Figure 1.1 there are five places and four transitions. In this instance, the places p_2, p_3 and p_5 provides inputs to transition t_2, and p_3 and p_5 are the output places of t_2. The marking function μ assigns one token to p_1, p_2 and p_5 and two tokens to p_3. Then, we can write:

$$\mu(p_1, p_2, p_3, p_4, p_5) = (1, 1, 2, 0, 1)$$

After transition t_2 fires, the marking μ will change to:

$$\mu(p_1, p_2, p_3, p_4, p_5) = (1, 0, 2, 0, 1)$$

One can easily see that a PN is a *state-oriented model* according to Gajski's definition of *model taxonomy*, introduced in the first part of this chapter. Indeed, a FSM is a particular case of a PN, where each transition t_i has exactly one input place and one output place [48].

Chapter 2

Related Work

In this chapter, we analyse several related publications based on their main goals in developing some techniques for modelling in a embedded system design framework. Several approaches are briefly reviewed in Section 2.1 and those based on Petri Nets are detailed in Section 2.2.

2.1 Modelling of Embedded System

Most groups in the area began working at high-level synthesis (HLS) [25] and then extended their work into hardware/software co-design. For example, De Micheli's group at Stanford University developed at early '90s the Olympus Synthesis System for HLS [20] and then this became part of the *Vulcan*[1] framework. The problem has been clearly formulated [33] and a description of the target architecture addressed by this group can be found in [31], which its by itself an extension of [33]. Specifications are given in *HardwareC* [41], a HDL developed in Stanford University by late '80s, and then compiled into a sequencing graph model (SGM) [35], a graph representation based on flow graphs [19], using the program *Hercules* [42]. The system behaviour is represented by a set of acyclic sequencing graphs ($\Phi = \{G_1, G_2, \ldots, G_n\}$), timing constraints (T) and resource constraints (R). Each of this G_i is a iterative construct in a HDL. The partitioning algorithm [34] starts with a initial solution of having all operations in *hardware*. Then, based on a communication-overhead cost criterion, some operations are moved into *software*. Co-Simulation of the mixed hardware-software system is performed by *Poseidon*, a tool developed by Gupta *et al.* [32] that performs concurrent simulation of multiple functional models. The simulator accepts a gate-level description of the hardware, the assembly code of the software component[2] and a description of the interface, giving as result a cycle-by-cycle simulation.

Gajski *et al.* developed a new framework called SpecSyn [26] based on *specification refinement* [27]. The system's behaviour was captured with SpecCharts [24], a hierarchical Program State Machine (PSM) language [27]. Then, as the group evolved, they created a complete specification language, called *SpecC*[3] [29, 63, 30] which keeps the attention of leading companies in the field [14].

The *Polis* project [43, 2, 61] has focused on control-dominated applications with system architectures composed of a single processor surrounded by either custom or library hardware (e.g.,

[1] VULCAN has approximately 60,000 lines of code

[2] Given in DLX assembler

[3] SpecC's evolution can be found at http://www.ics.uci.edu/ specc/history.html

microcontroller peripherals and other IPs). Polis uses Co-design Finite State Machines (CFSM) as the internal representation for the system description. A CFSM [16] is a *extended asynchronous* FSM that operates on a set of integer variables with arithmetic, relational and logical operators. To describe a CFSM network, Chiodo *et al.* [15] uses a specific representation format called *SHIFT*, which is itself a multi-valued extension of the Berkeley Logic Interchange Format (BLIF) [10, 56]. Specifications are given in a higher level language, usually *ESTEREL* [9, 8, 7], and the designer must manually do the partitioning. However, Polis provides an integrated environment for *rapid-prototyping*, fast *co-simulation* and *formal verification*. Cosimulation is based on software timing estimates [44] and both, hardware and software, are simulated in the *Ptolemy* environment[4] [12, 13, 60]. Coverification is performed by translating CFSM into traditional FSM, such that existing FSM-based verification techniques can be applied [3]. A complete and typical design flow in Polis is shown in [44].

The kernel of *Chinook* cosynthesis system, developed at University of Washington, is a cosimulator called *Pia* [39], which has been first implemented in Ptolemy simulation domain and later turned into Java applets. Pia's main feature is its ability to represent data at different levels of abstraction[5] and to dynamically change information during run-time.

Cosyma[6] [50] uses an internal representation called Extended Syntax Graph (ESG) [6], which combines concepts of *data flow graph* and *syntax graph*. The Specifications are given in C^x [18, 38], a superset of ANSI-C and the resulting ESG is scheduled as explained in [5]. Partition algorithm starts from a initial software solution and tries to extract hardware iteratively [36]. Newer versions of COSYMA combines scheduling and partitioning [4]. The result of the partitioning process is a piece of *C* code for software and *RTL* VHDL for hardware. The first is compiled with GNU C [45] and the later synthesised with the Braunschweig Synthesis System (BSS), a HLS tool for fast coprocessor design.

More recently this group[7] is turning onto a formal IDR called *SPI* [23], which integrates various aspects of several Model of Computation (MOC) [64]. Moreover, a major enhancement of SPI called *FunState* has been presented in [58] by Thiele *et al*and they claim that various MOC can be represented as a subset of FunState. This IDR has been developed by means of functional programming and states machines.

2.2 Modelling of Embedded System with PN based techniques

Several researchers have shown that PNs are a powerful formalism to model the behaviour of parallel systems. Since embedded systems show many concurrency in their functionality, PNs has been applied to ESD in several ways.

The HW/SW representation presented by the Linköping University is based in Extended Timed Petri Nets (ETPN). They introduced this concept in their *CAMAD* high-level synthesis system [53]. *ETPN* is a unified design representation created by Peng [52] to explicity capture the intermediate

[4] interested readers searching for a deeper understanding on the way the kernel works, should refer to [11]

[5] without having to write separate specifications for each level

[6] COSYMA is a project from the T.U. Braunschweig and lies on the basis given in [22] and [37] (this one in German)

[7] together with ETH at Zurich

result of a design for making accurate decisions in the design algorithm. It consists of two separate but related parts: The *control part* and the *datapath*. The first one is captured as a Timed Petri Net (TPN)[8] and the later is represented as a directed graph where nodes are used to capture data manipulation and storage.

This intermediate representation model is part of a framework, called *PURE* [57], that has been implemented in Prolog. In this methodology, partitioning is carried out by a *inter-domain* movement of functionality based on Petri net's potential feature of concurrency representation. By making data dependencies explicit[9], communication needs can be easily identified. So far, they have exploited the fact of having a *uniform* representation for the Co-Simulation, rather than trying to jointly simulate modules from different domains.

SAVE is a joint research project of *ESDlab* at KTH and *ESLAB* at Linköping University. The framework includes a Petri net based representation, called *PRES* [17], which is mainly a extension of Petri net performed in order to make a *heterogeneous model* out of a *state-oriented model*. *PRES* is more focused on Embedded Systems than ETPN, since the later was thought to help in any CAD environment. It can be used to model a system at different levels of of abstraction using hierarchy. The model also includes an explicit notation of time, as it done by any extension of TPNs, and *tokens* are redefined in such a way they let the dynamic behaviour to deal with both, *data* and *control*. A system modelled in PRES is later formally verified using symbolic model checking to prove its correctness. This co-verification methodology is based on SMV [1], a tool developed in Carnegie Mellon University.

[8] TPNs were introduced in [47]

[9] because ETPN extends TPN by adding this information to the system representation

Chapter 3

Research Perspective

Structural definition, graphical representation and behavioural rules for NOEMI were presented in a Paper, which is included in Apendix B. Therefore, this chapter will only go further on the application examples introduced in the paper, in order to give some intuitive understanding of PN based modelling techniques.

3.1 Application Example

Apendix B shows the specification for the multiplier proposed by Cortés *et al* [17]. The algorithm takes two positives integers and produces as output the result of multiplying those numbers, by iterative sums. We build up the IDR for this specification using three different approaches: *ETPN*, *PRES* and *NOEMI*.

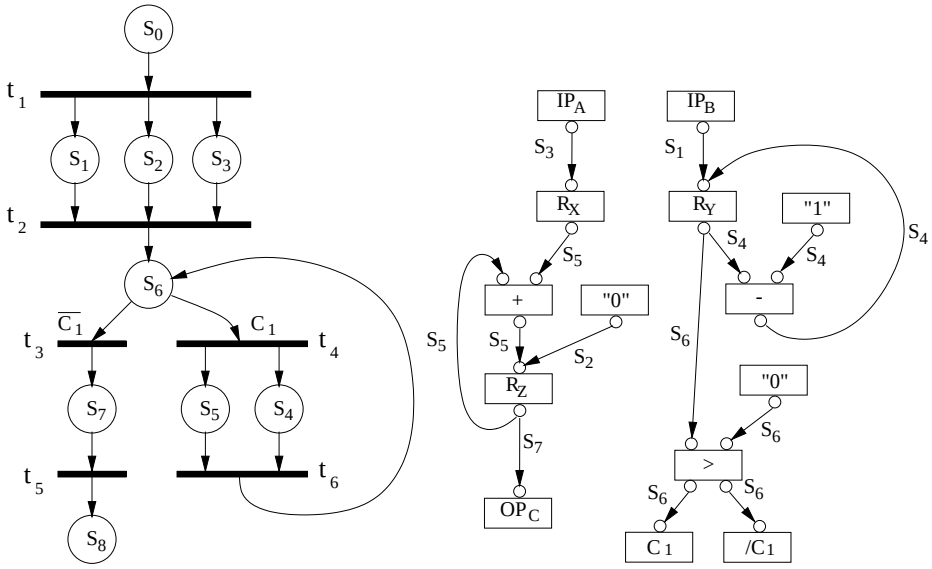
The ETPN model of this multiplier is shown in Figure 3.1. The model has two parts: *Control* (Figure 3.1(a)) and *data path* (Figure 3.1(b)). In the control part, places (S-elements) and transitions (T-elements) are graphically represented in order to follow the behaviour or the system. In the data path, rectangles represent each vertex of a graph and labels indicates its primary function. Small circles attached to a vertex represents either inputs or outputs, depending whether these are above or below the box.

To begin the multiplication, a token should be placed at S_0 assuming that IP_A is loaded with the first number and IP_B with the second number. Now, firing the first transition[1] will result in S_1, S_2 and S_3 being loaded with a token. This means that *register* R_Y will be loaded with the value from *input port* IP_B, R_Z will be loaded with the value 0 and R_X will be load with IP_A.

Any change in the marking (i.e. firing a transition) of Figure 3.1(a) will automatically cause some action in Figure 3.1(b). For example, firing T_4 will lead to the operations in feedback arcs (i.e. S_4, S_5) in Figure 3.1(b) to be carried out.

ETPN model is a good internal representation for a HLS framework, since it exploited one of the features of PN, the concurrency identification, which is very desirable for hardware synthesis. However, *data* and *control* flow are not very well linked in this model, which is certainly not a positive feature at embedded systems design. To overcome the lack of union between these two domains, NOEMI model uses only one graph.

[1] Note that only this transition is enabled.



(a) Control Part (b) Data Path

Figure 3.1: ETPN representation of the multiplier.

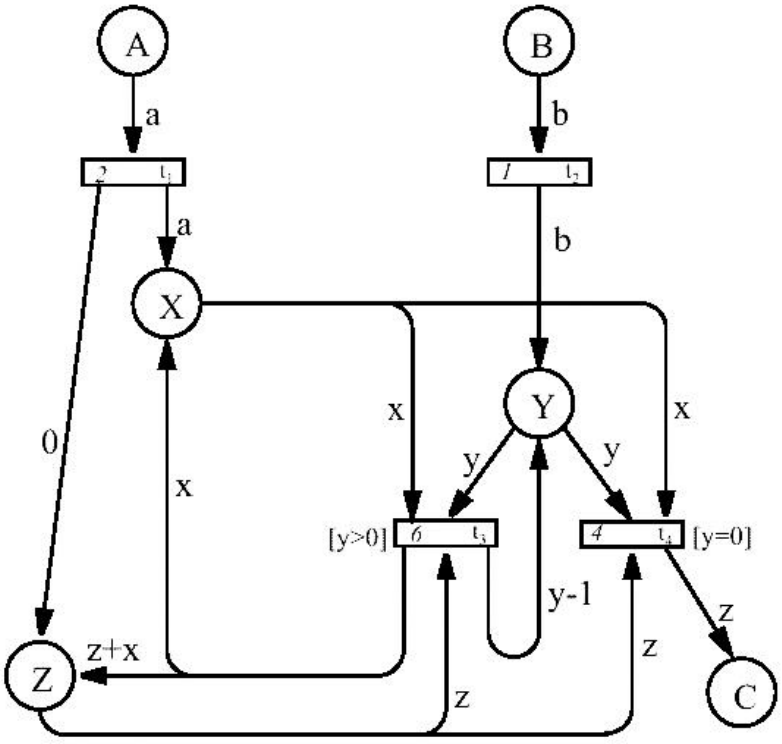


Figure 3.2: PRES representation of the multiplier.

Figure 3.2 shows the PRES model of the multiplier. Like in classical Petri nets, places are graphically represented by circles, transitions by boxes, and arcs by arrows. In this example $P = \{A, B, X, Y, Z, C\}$ and $T = \{t_1, t_2, t_3, t_4\}$. Inscriptions on the arcs are used together with transitions to define output functions (inscribed on output arcs, e.g. "$y-1$" is an output function of t_3) are captured as expressions in terms of the variables written on its input arcs (e.g. t_3 has x, y, z as input variables). Firing a transition will cause all output functions to be executed. To deal with the control flow, the concept of *guard function* is used [40]. Guards are enclosed in square brackets and are also functions of the variables on input arcs, deciding whether the transition is enabled or not.

Performing a multiplication in this model means to put two tokens[2], one in A place and other in B place, with the values we may use as inputs to the operation.

Then, either t_1 or t_2 may fire. The firing of a transition will change the marking in a similar way to PNs, with the addition that a new value has to be calculated for the token that has been moved. This is done by the use of labelled functions placed among the places.

It has to be pointed out that *operations* and *conditions* are represented in very different ways, among the three techniques used. In ETPN, a complete dataflow graph is created out of it. PRES uses labels placed in F and T to represent operations and conditions respectively. In NOEMI operations are carried out by firing *data transitions* (e.g. when firing q_4, place z acquires the value of $x + y$) and conditions are carried out by *control transitions* (e.g. transition t_1 will fire only if condition "$y > 0$" is true).

3.2 Conclusion and future work

Further investigation into NOEMI properties has to be done, in order to let it be a self-consistent theory. Currently, several embedded systems have been modelled using NOEMI representation, concluding that the technique is well suited for embedded systems design. The aim of this PhD project is to propose a hardware/software co-design framework based on the new modelling technique presented in this report, which will allow efficient co-synthesis of embedded systems. Following is a draft list of steps to perform in the ongoing research:

1. Development of State Equations (SE) for the model, based on the topology of the embedded system specification. Having SE with an algebraic structure will give the method a higher degree of formalism.

2. Reachability Analysis, which is fundamental among the study of dynamic properties of any system. After modelling embedded systems with NOEMI, we shall be able to conclude whether a specific marking is reachable or not from a initial marking.

3. Inclusion of time analysis. Guiding the research towards co-synthesis implies to have a good understanding of timing constraints.

4. Investigation of NOEMI applicability into HLS. Hardware/Software Co-Design can be interpreted as a two-dimension point of view of the same one-dimension problem formulated

[2]Should be noted that tokens are different from the definition given in Section 1.1. This new kind of tokens has two parameters, the *value* (firstly) and the *time stamp* (secondly).

in HLS. Therefore, to apply the methodology into HLS will give us better understanding of the field beforehand. This step contains:

   (a) Implementation of a *Scheduling* scheme in the framework will add valuable information to the cost and performance analysis of the system. We will explore both, Resource Constrained Scheduling (RCS) and Time Constrained Scheduling (TCS) [51].

   (b) Determination of *Allocation* and *Binding* process using NOEMI.

5. Analysis of existing and/or realisation of a new *Partitioning* algorithm to extension the framework towards co-synthesis.

Bibliography

[1] The SMV System. http://www.cs.cmu.edu/~modelcheck/smv.html. (last visited in June 2000).

[2] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, and B. Tabbara. *Hardware-Software Co-Design of Embedded Systems: The Polis Approach*. Kluwer Academic Publishers, 1997.

[3] F. Balarin, H. Hsieh, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli. Formal verification of embedded systems based on CFSM networks. In *Design Automation Conference - DAC '96*, 1996.

[4] T. Benner and R. Ernst. An approach to mixed systems Co-Synthesis. In *Proceedings of the* 5^{th} *International Workshop on Hardware/Software Codesign - CODES/CASHE '97*, pages 9–14, 1997.

[5] T. Benner, R. Ernst, and A. Österling. Scalable performance scheduling for Hardware-Software Cosynthesis. In *Proceedings of the European Conference on Design Automation*, pages 164–169, Brighton, Great Britain, Sept. 1995.

[6] T. Benner, J. Henkel, and R. Ernst. Internal representation of embedded hardware- / software- systems. In *International Workshop on Hardware/Software Codesign - CODES/CASHE '93*.

[7] G. Berry. Hardware implementation of pure Esterel. In *Proceedings of the ACM Workshop on Formal Methods in VLSI Design*, Jan. 1991.

[8] G. Berry and L. Cosserat. *The esterel synchronous programming language and its mathematical semantics*. Ecole National Superieure de Mines de Paris, 1984. language for synthesis.

[9] F. Boussinot and R. D. Simone. The ESTEREL language. *Proceedings of the IEEE*, 79(9):1293–1304, 1991.

[10] R. K. Brayton, M. Chiodo, R. Hojati, T. Kam, K. Kodandapani, R. P. Kurshan, S. Malik, A. L. Sangiovanni-Vincentelli, E. M. Sentovich, T. Shiple, and H. Y. Wang. BLIF-MV: An interchange format for design verification and synthesis. Technical report, U. C. Berkeley, 1991.

[11] J. T. Buck. The ptolemy kernel. Technical Report ERL-93-8, U. C. Berkeley, Jan. 1993.

[12] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *Int. Journal of Computer Simulation*, Jan. 1990. Special issue on simulation software development.

[13] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *Int. Journal of Computer Simulation*, 4(2):155–182, Apr. 1994.

[14] A. Cataldo. SpecC opens dialogue with SystemC camp. *EE Times*, Mar. 2000.

[15] M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli. A formal specification model for hardware/software codesign. Technical Report ERL-93-48, U. C. Berkeley, June 1993.

[16] M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli. Hardware/software codesign of embedded systems. *IEEE Micro*, 14(4):26–36, Aug. 1994.

[17] L. A. Cortés, P. Eles, and Z. Peng. A petri net based model for heterogeneous embedded systems. In *IEEE NORCHIP Conference*, pages 248–255, Oslo, Norway, Nov. 8-9 1999.

[18] J. H. D. Herrmann and R. Ernst. An Approach to the Adaptation of Estimated Cost Parameters in the COSYMA System. In *Proceedings of the* 3^{rd}*International Workshop on Hardware/Software Codesign - CODES/CASHE '94*, pages 100–107, 1994.

[19] A. L. Davis and R. M. Keller. Data Flow Program Graphs. *IEEE Computer*, 15(2), Feb. 1982.

[20] G. De Micheli, D. C. Ku, F. Mailhot, and T. Truong. The Olympus Synthesis System for Digital Design. *IEEE Design & Test of Computers*, 1990.

[21] S. Edwards, L. Lavagno, E. Lee, and A. Sangiovanni-Vincentelli. Design of embedded systems: Formal model, validation and synthesis. *Proceedings of the IEEE*, 85(3):366–390, Mar. 1997.

[22] R. Ernst and J. Henkel. Hardware-Software Co-Design of Embedded Controllers Based on Hardware-Extraction. In *International Workshop on Hardware/Software Codesign - CODES/CASHE '92*, Estes Park, Colorado, USA, 1992.

[23] R. Ernst, D. Ziegenbein, K. Richter, L. Thiele, and J. Teich. Hardware/software codesign of embedded systems - the SPI workbench. In *Proceedings of the IEEE Workshop/Conference on VLSI*, Orlando, USA, June 1999.

[24] D. Gajski, Vahid, S. Narayan, and J. Gong. *Specification and Design of Embedded Systems*. Prentice-Hall, 1994.

[25] D. Gajski, A. Wu, N. Dutt, and S.Lin. *High-Level Synthesis*. 1992.

[26] D. D. Gajski and F. Vahid. Specification and Design of Embedded Software/Hardware Systems. Technical Report CS-94-08, U. C. RiversideDepartment of Computer Science, Oct. 24 1994.

[27] D. D. Gajski, F. Vahid, and S. Narayan. A system-design methodology: Executable specification refinements. In *European Conference on Design Automation '94*.

[28] D. D. Gajski, J. Zhu, and R. Dömer. Esential issues in codesign. In J. rgen Staunstrup and W. Wolf, editors, *Hardware/Software Co-Design: Principles and Practice*, pages 1–45. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[29] D. D. Gajski, J. Zhu, and R. Domer. The SpecC+ language. Technical Report ICS-TR-97-15, U. C. Irvine, Department of Information and Computer Science, Apr. 1997.

[30] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Methodology*. Kluwer Academic Publishers, Dordrecht, Mar. 2000.

[31] R. K. Gupta, N. C. Claudionor, Jr., and G. De Micheli. Program implementation schemes for hardware-software systems. *IEEE Computer*, 27(1):48–55, Jan. 1994.

[32] R. K. Gupta, C. N. Coelho, Jr., and G. De Micheli. Synthesis and simulation of digital systems containing interacting hardware and software components. In *Proceedings of the* 29^{th} *Design Automation Conference - DAC '92*, pages 225–230, Los Alamitos, CA, USA, June 1992. IEEE Computer Society Press.

[33] R. K. Gupta and G. De Micheli. System-level synthesis using re-programmable components. In *Proceedings of the European Conference on Design Automation*, pages 2–7, Washington, Mar. 16–19 1992. IEEE Computer Society Press.

[34] R. K. Gupta and G. De Micheli. System synthesis via hardware-software co-design. Technical Report CSL-TR-92-548, Stanford University, Computer Systems Laboratory, 1992.

[35] R. K. Gupta and G. De Micheli. A Co-Synthesis aproach to embedded system design automation. *Embedded System Journal*, 1994.

[36] J. Henkel, T. Benner, and R. Ernst. Hardware generation and partitioning effects in the cosyma system. In *Proceedings of the SecondInternational Workshop on Hardware/Software Codesign - CODES/CASHE '93*, Cambridge, Massachusetts, USA, Oct. 1993.

[37] J. Henkel and R. Ernst. Hardware/Software-Co-Design für Mikrocontroller. In *Proceedings of the Workshop für Entwurfsmethodik für Integrierte Schaltungen und Systeme*, Feb. 1992. (in German).

[38] J. Henkel and R. Ernst. A Path-Based Estimation Technique for Estimating Hardware Runtime in HW/SW-Cosynthesis. In *Proceedings of the* 8^{th} *IEEE International Symposium on System Level Synthesis*, pages 116–121, Cannes, France, 1995.

[39] K. Hines and G. Borriello. Dynamic communication models in embedded system cosimulation. In *Proceedings of the* 34^{th} *Design Automation Conference - DAC '97*, June 1997.

[40] K. Jensen. *Coloured Petri Nets. Basic concepts, analysis methods and practical use.* EATCS monographs on Theoretical Computer Science. Springer-Verlag, 1992.

[41] D. Ku and G. De Micheli. HardwareC – A language for hardware design (version 2.0). Technical Report CSL-TR-90-419, Stanford University, Computer Systems Laboratory, 1990.

[42] D. Ku and G. De Micheli. High-level Synthesis and Optimization Strategies in Hercules and Hebe. In *Proceedings of the European ASIC Conference*, pages 111–120, Paris, France, May 1990.

[43] L. Lavagno, A. Sangiovanni-Vincentelli, and H. Hsieh. Embedded system co-design: Synthesis and verification. In G. DeMicheli and M. Sami, editors, *Hardware/Software Co-Design*, pages 213–242. Kluwer Academic Publishers, 1996.

[44] J. Liu, M. Lajolo, and A. Sangiovanni-Vincentelli. Software timing analysis using hw/sw cosimulation and instruction set simulator. In *International Workshop on Hardware/Software Codesign - CODES/CASHE '98*, 1998.

[45] S. Loosemore, R. L. Stallman, R. McGrath, A. Oram, and U. Drepper. *The GNU C Library Reference Manual*. Free Software Foundation, Boston, MA 02111-1307, Oct. 1996.

[46] M. C. McFarland, A. C. Parker, and R. Camposano. The high-level synthesis of digital systems. *Proceedings of the IEEE*, 78(2):301–318, Feb. 1990.

[47] P. Merlin and D. J. Faber. Recoverability of communication protocols. *IEEE Transactions on communications*, 24(9):1036–1043, 1976.

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

[49] Z. Navabi. *VHDL: analysis and modeling of digital systems*. McGraw-Hill, 2 edition, 1998.

[50] A. Österling, T. Benner, R. Ernst, D. Herrmann, T. Scholz, and W. Ye. The Cosyma System. In J. Staunstrup and W. Wolf, editors, *Hardware/Software Co-Design: Principles and Practice*, pages 263–282. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[51] P. G. Paulin and J. P. Knight. Algorithms for high-level sysnthesis. *IEEE Design & Test of Computers*, 6(6):18–31, 1989.

[52] Z. Peng. *A Formal Methodology for Auto-mated Synthesis of VLSI Systems*. PhD thesis, Linköping University, 1987.

[53] Z. Peng and K. Kuchcinski. Automated transformation of algorithms into register-transfer level implementations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(2):150–166, Feb. 1994.

[54] J. Peterson. *Petri net theory and the modeling of systems*. Prentice-Hall, Englewood Cliffs, 1981.

[55] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962. (in German).

[56] 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, U. C. Berkeley, 1992.

[57] E. Stoy. A petri net based unified representation for hardware/software co-design. Licentiate Thesis LiU-Tek-Lic 1995:21, Linköping University, S-581 83, Linköping, Sweden, 1995.

[58] L. Thiele, K. Strehl, D. Ziegenbein, R. Ernst, and J. Teich. *FunState*-An Internal Design Representation for Codesign. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design - ICCAD '99*, pages 558–565, Nov. 1999.

[59] D. E. Thomas and P. R. Moorby. *The Verilog Hardware Description Language*. Kluwer Academic Publishers, Norwell MA, 2 edition, 1994.

[60] U. C. Berkeley, California, CA 94720. *The Almagest - Ptolemy 0.7 User's Manual*. Available in: http://ptolemy.eecs.berkeley.edu/papers/almagest/.

[61] U. C. Berkeley, California, CA 94720. *POLIS - A design environment for control-dominatad embedded systems - Version 0.3* User's Manual, Aug. 1999.

[62] W. H. Wolf. Hardware-software co-design of embedded systems. *Proceedings of the IEEE*, 82(7):967–989, July 1994.

[63] J. Zhu, D. D. Gajski, and R. Doemer. Syntax and semantics of the SpecC+ language. Technical Report ICS-TR-97-16, U. C. Irvine, Department of Information and Computer Science, Apr. 1997.

[64] D. Ziegenbein, R. Ernst, K. Richter, J. Teich, and L. Thiele. Combining multiple models of computation for scheduling and allocation. In *Proceedings of the* 6^{th} *International Workshop on Hardware/Software Codesign - CODES/CASHE '98*, pages 9–13, Seattle, USA, Mar. 1998.

Appendix A

List of Symbols

$\in$ belongs to

$\Longrightarrow$ implies

$\Longleftrightarrow$ if and only if

$\forall$ for all

$\mathbb{C}$ complex domain

$\exists$ exists

$|.|$ modulus

${}^{\bullet}x$ pre-set of x in the control domain

${}^{\circ}x$ pre-set of x in the data domain

$x^{\bullet}$ post-set of x in the control domain

x° post-set of x in the data domain

$\cup$ union

$\cap$ intersection

$\emptyset$ empty set

$\wedge$ and

$\vee$ or

Appendix B

Paper: Dual Transitions Petri Net based Modelling Technique for Embedded Systems Specification

This paper, which has been submitted to Design, Automation and Test in Europe (DATE) - *2001* conference, presents the new modelling technique discussed here. The paper shows how the union between control and data flow is done (by means of NOEMI model) and briefly analyses its applicability into nested loops, which are a fundamental issue in complex embedded system design. Two examples are included to show the applicability of the technique, and one of them is further explained in Section 3.1.