A HARDWARE IMPLEMENTATION OF SAFE PETRI NET MODELS

Viorica SUDACEVSCHI[1], Victor ABABII[2], Valentin NEGURA[3]
Technical University of Moldova
Str. Stefan cel Mare, 168, Chisinau
Republic of Moldova
[1)] svm@mail.utm.md, [2)] avv@mail.utm.md, [3)] vnegura_md@yahoo.fr

Abstract. *In this work is proposed a hardware implementation method of Safe Petri Net model (SaPN). The processing structure presents an interaction between homogeny processing elements functional defined. On dependence of interconnections of elements, there can be implemented any complexity of Safe Petri Net model. The processing elements represent a flexible architecture, which permit to adjust them to Safe Petri Net model. Using this architecture and interactions of elements we obtained a reduced time of processing to checking Safe Petri Net to reachability, viability and other behavior properties.*
Keywords: *Petri Net models Hardware implementation, Safe Petri Net, FPGA, AHDL, parallel data processing.*

Introduction

The Safe Petri Net (SaPN) is often used for modeling parallel data processing system. As the computer systems become more and more complex a special attention is granted to the elaboration of parallel systems with a non-determinist behavior, where the data processing components operate independently and interact with each other only in certain time moments. In multi-computer systems, distributed memory systems, communication networks performance must be guaranteed to a very high degree of certainty. In practically all cases a comprehensive test program cannot offer this degree of certainty. Design for these systems requires software-modeling tools that are capable of verifying temporal specifications as well as functional specifications [2].

There are many computational models that can be used as the basis for construction of a software model for complex target systems. One of the main requirements is that concurrency and synchronization must also be supported. A modeling paradigm that supports all these requirements is Petri Net model [4].

Safe Petri net model definition

Petri Nets are defined as mathematical models that determine an efficient theoretical support for description of parallel processes behavior of discrete systems with asynchronous interactions [1, 3]. Petri Net describes in a compact form the internal structure of the system and relations between its elements, the modifications of the system states in a dynamic mode, parallel processes that occur in such systems, local interactions between processes and their non-determinist behavior. Petri Nets are able to model such spread situations as concurrency, cooperation, synchronization, unsafe states, deadlocks that is very important for a design process. The abstraction level of Petri Net model is very high and corresponds to the description of system interactions in terms of two fundamental notions: events- transitions and conditions- places.

A SaPN is a five-tuple $SaPN = \{P, N, IN, OUT, M_0\}$ where:

$P = \{p_1, p_2, ..., p_n\}$ is a set of places;

$T = \{t_1, t_2, ..., t_m\}$ is a set of transitions;

$IN : (P \times T) \to N$ is an input functions that defines directed arc from places to transitions;

$OUT : (P \times T) \to N$ is an output functions that defines directed arc from transitions to places;

$M_0 = (\mu_0(p_1), \mu_0(p_2), .., \mu_0(p_n))$ is initial marking of places μ;

and $P \cup T \neq \mathrm{O}$, $P \cap T = \mathrm{O}$, N is a set of non-negative integers, and $n = |P|$, $m = |T|$ [1, 10].

A marking $M_k = (\mu_k(p_1), \mu_k(p_2), .., \mu_k(p_n))$ can be interpreted as an integer vector which includes per place $\mu(p_i), \forall i = \overline{1,n}$ one element

which correspond to the number of tokens on place $p_i, \forall i = \overline{1,n}$, for SaPN $\mu_k(p_i) \in \{0,1\}$. The marking describes the state of the adequate dynamic system, and dynamic changes are modeled as tokens movement from one place to another.

Transition $t \in T$ is enabled in marking M_k, if $\forall p \in P : \mu_k(p) = (p,t)$. Any transition enabled in M_k can fire, changing the marking of any $p \in P$ to marking M_{k+1}, according to conditions

$M_{k+1}(p) = M_k(p) - (p,t) + (t,p)$,

k - it is a step of data processing.

The advantages of Petri Nets implementation in FPGA architectures

The major drawback of Petri Nets is the larger state space that can be generated even by rather simple models. That's why the software realization or hardware realization on sequential processor requires radical model simplification or extremely long run times.

The advent of Field-Programmable Gate Array circuits allows using them as hardware solutions for simulation of Petri Net models with a large number of states. FPGA circuits contain thousands of gates equivalents and provide enough logic to implement several small processors on a single chip. There are many available tools for their programming as high level hardware description languages (e.g. VHDL, AHDL, Verilog, ABEL) or traditional schematics. These languages also can be used to check the model before it is loaded on to the hardware. The possibility of a run-time reconfiguration allows the use of adoptive algorithms that can reduce the time that is necessary for Petri Net simulation.

In the literature several studies of FPGA implementation of Petri Net have been described. In [6] reachability analyze of Petri Net by using an FPGA based accelerator is proposed. An FPGA implementation to execute the Petri Net transition firing algorithm is described in [5].

The structure of system for modeling of SaPN

The structure of system (figure 1) consists of: PC – personal computer architecture; ISA Interface – ISA standard interface; HI SaPN models - hardware implementation of Safe Petri Net models in FPGA architecture; JTAG Interface – JTAG standard interface for FPGA circuit's configuration.

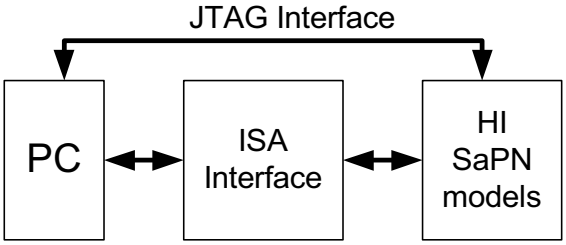

Figure 1. **The structure of system.**

The functional algorithm of system

The functional algorithm of system is presented in figure 2, where:

START – beginning of algorithm;

Design SaPN Models – design and edit the SaPN model;

SaPN Models to Object Models Conversion – conversion SaPN model from graphical model to object oriented code structure;

OM to AHDL Codes Conversion - conversion SaPN object oriented structure to AHDL code;

ERROR – check errors;

Compilation AHDL Codes – compile the AHDL code to obtain the code of FPGA architecture;

Load Config File in FPGA Architecture – load the configuration code of FPGA architecture;

Simulation Processes;

Load Simulation Results – load the simulations results in RAM PC;

View Simulation Results – view simulations results on the screen;

STOP – the end of algorithm.

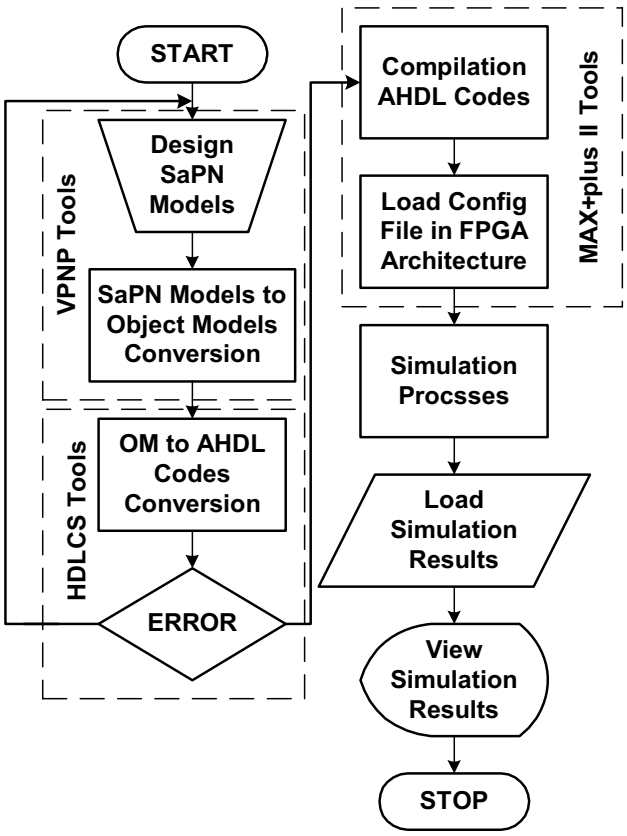

Figure 2. The Algorithm of Functioning of the System.

Interface architecture and hardware structure

Interface architecture and hardware structure of SaPN is presented in figure 3. This architecture consists of two main blocks: **Interface** – computer interface built on the basis of CPLD (MAX3000A) and **HI SaPN Model** – the hardware implementation of SaPN model built on the basis of FPGA (FLEX10K), technology SRAM FPGA FLEX10K which permits to configure the computational structure on-line to configuration of HI SaPN models.

Existence of integrated memory in FPGA FLEX10K architecture permits to storage data without to consume the main logic. The FPGA configuration is done writing the configure file via JTAG port [7].

Interface block consist of:

DB (Data Bus) – bidirectional data bus;

I/O Control (Input/Output Control) – the input/output control block;

DC (Decoder) – Address bus decoder.

HI SaPN model consists of:

T_j – set of transitions T;

P_i – set of place P;

RAM – Memory where is stored the reachability graph;

B SYN – Synchronization' block of data processing operation;

RAM SYN – Memory where is storage the synchronization signals of data processing operation.

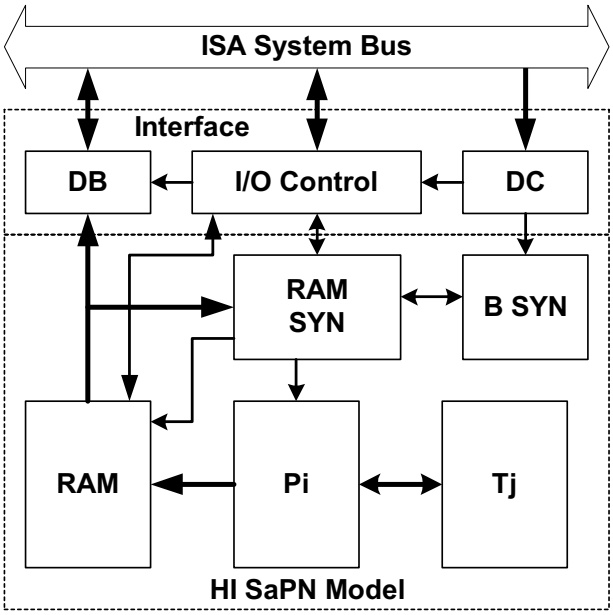

Figure 3. Interface and Hardware Safe Petri Net Architecture.

The functional principles of system consist of following steps:

1. ***Configure the FPGA architecture.*** Via JTAG interface is written the configure file which determine the SaPN model structure.

2. ***Load the RAM SYN.*** The computer writes via interface consecutively the content of synchronization file of data processing operation;

3. ***Launch the simulation process.*** Launching the simulation process consist in generating the output operation on address reserved for system;

4. ***The simulation process.*** **B SYN** block permit to select the synchronizations signals from **RAM SYN** which run in parallel the data processing in this places. At each synchronization step the SaPN state is written in **RAM**. During is completed the **RAM**, is

generated the DRQ signal, which launches the DMA that read the data from **RAM** to PC memory.

The AHDL codes for Place *P* and Transition *T*

The structures of Place P. The structure of place ***P*** depends on ***P*** object model formed at the conversion operation of graphic SaPN model to Object SaPN model. Object Model of SaPN place ***P*** has the following format:

```
<object class="TclsNodeDiscrete"
name="p1" left="265" top="105"
width="30" height="30" group=""
name="p1" capacity="1" size="osNormal"
markers="1">
</object>
```

Where ***capacity*** determines the maximum number of markers in place ***P***, ***markers*** determine the number of markers in initial state [8, 9].
The AHDL code of place ***P*** for SaPN has the following format:

```
SUBDESIGN Safe_Place
(       Dec_0, Dec_1, Dec_2, Dec_3, Inc_0, Inc_1,
Inc_2, Inc_3, SYN, nCLR, nPR    : input;
        Mout : output;
)
VARIABLE      FF        : SRFF;
                        Dec : NODE;
                        Inc : NODE;
BEGIN
        Dec = (Dec_0 # Dec_1 # Dec_2 # Dec_3);
        FF.R = Dec;
        Inc = (Inc_0 # Inc_1 # Inc_2 # Inc_3);
        FF.S = Inc;
        FF.clk = SYN & (Dec $ Inc);
        FF.clrn = nCLR;
        FF.prn = nPR;
        Mout = FF.q;
END;
```

The AHDL code of place ***P*** description:
Dec - – decrement of place ***P***;
Inc – increment of place ***P***;
SYN – synchronization of increment and decrement operation.
nCLR – reset place ***P***;
nPR – set place ***P***;
Mout – determine the number of markers in place ***P***.
Suggested AHDL code satisfies the following conditions

$$\mu_{k+1}(p_i)=\begin{cases}1\ IF\ (I\ \&\ \overline{D}),\\ \mu_k(p_i)\ IF\ (\overline{I}\ \&\ \overline{D}),\\ \mu_k(p_i)\ IF\ (I\ \&\ D),\\ 0\ IF\ (\mu_k(p_i)=1\ \&\ (\overline{I}\ \&\ D)).\end{cases}.$$

$$\forall i=\overline{1,n}$$

The structure of Transition T. The structure of transition ***T*** depends of ***T*** object model formed at the converting operation from graphical SaPN model to object model.
The object model of SaPN transition ***T*** has the following format:

```
<object class="TclsTransInstant"
name="t1" left="315" top="77.5"
width="30" height="5" group=""
size="osNormal" angle="0">
  <priority formula="RefV"
Value_Reference="1" Value_Actual="1"
Time_Value_Reference="1"
Time_Value_Actual="1" />
  <speed formula="RefV"
Value_Reference="100"
Value_Actual="100"
Time_Value_Reference="0.01"
Time_Value_Actual="0.01" />
</object>
```

The AHDL model of transition ***T*** for SaPN has the following format:

```
SUBDESIGN transition4
( Min_0, Min_1, Min_2, Min_3 : input;
        Dec_Inc : output;
)
BEGIN
        Dec_Inc = (Min_0 & Min_1 & Min_2 & Min_3);
END;
```

The AHDL code of transition ***T*** description:
Min – transition ***T*** inputs - the signals which are applied from the outputs of respective place.
Dec_Inc – the decrement of ascendant place and increment of incident place.

The connection between elements transition T and place P is done according the incidence matrix that is generated by object ***Link L***. Object Model of SaPN link ***L*** has the following format:

```
<object class="TclsLinkNormal"
name="L1" left="200" top="139.5"
width="0" height="36.5" group=""
name="L1" src="p1" dst="t1">
  <weight formula="RefV"
Value_Reference="1" Value_Actual="1" />
</object>
```

where scr=***"p1"*** points to the source name ***(***place ***p1***), and ***dst="t1"*** points to the destination name (transition ***t1***) of the link with ***name="L1"***.

Example of Hardware SaPN models implementation

There are proposed for examination a simple SaPN model for functional checking of elaborated system (Figure 4.). The graphical SaPN model which includes 5 places $P = \{p_1, p_2, p_3, p_4, p_5\}$ and 4 transitions $T = \{t_1, t_2, t_3, t_4\}$. The hardware (FPGA-based) implementation of SaPN model is presented in figure 5.

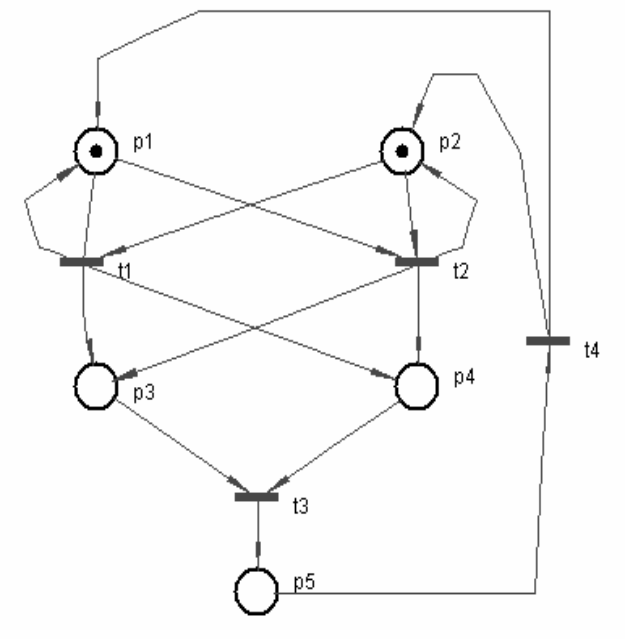

Figure 4. The SaPN Model.

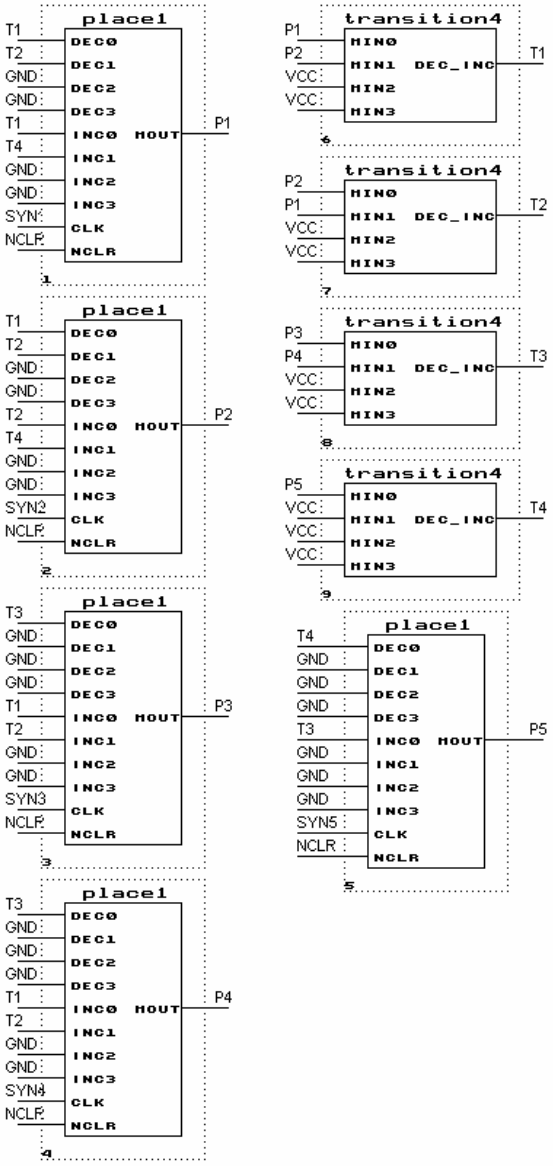

Figure 5. The HI SaPN model.

Synchronization in the simulation process

The data processing in places ***P*** is done in parallel in order to increase the calculation speed. This is possible when the synchronization signals SYN are generated to enable the operation of **BSYN** block and **RAM SYN** memory in parallel.

The synchronization algorithm consists of:

Consecutively analysis of each transition;

Process the data in those places where transition has links.

The ***RAM SYN*** memory is organized according to Table 1. It consists of m blocks, where m is

the number of transitions in Petri net model. In each block the order of transition firing is changed to achieve all possible states of the model. In rows the transition links to each position are shown.

Table 1. RAM SYN memory

	P1	P2	P3	P4	P5
Set M0					
T1	1	1	1	1	0
T2	1	1	1	1	0
T3	0	0	1	1	1
T4	1	1	0	0	1
Set M0					
T2	1	1	1	1	0
T3	0	0	1	1	1
T4	1	1	0	0	1
T1	1	1	1	1	0
…					
Set M0					
T4	1	1	0	0	1
T1	1	1	1	1	0
T2	1	1	1	1	0
T3	0	0	1	1	1

Conclusion

The system modeling is one of the most important domains for developing nowadays research. The system which was elaborated and examined in this work is a particularly case from this domain and permits to reduce the modeling time on the basis of SaPN model. The major problem of hardware (FPGA-based) implementation SaPN systems are the correctness of results obtained in modeling. It was obtained adequately results referred to reachability graph using the parallel-consecutively synchronization method of data processing in Place ***P***.

The structure of functional elements place ***P*** and transition ***T*** was elaborated to contain a lower number of logical gates from FPGA FLEX10K architecture. Examining some structure of ***P*** and ***T*** elements, was selected the structure which was presented in this work, with capacity consuming – 4 logical cells for place ***P*** and 1 logical cell for transition ***T***. Was taken in count and RAM memory in FPGA FLEX10K architecture which is used for storage the generated state from SaPN.

Directly connections of interface to system bus and using DMA communications, permit to reduce the time of data input in PC.

A very important field of research and development is implementation of different SaPN models for examination of systems and real process. For example: technological processes and automatic systems.

References

[1]. Peterson, J. (1984) *Petri Net theory and modeling of systems*. New York.

[2]. Agrawal D. et al. (1989) *Evaluation the performance of multicomputer configurations* // IEEE Compute., may 1989.

[3]. G. Ciardo and K. S. Trivedi. (1991) *A decomposition approach for Stochastic Petri Net models.* In Proc. 4th Intern. Workshop on Petri Nets and Performance Models, Melbourne, Australia, December 1991.

[4]. Garlic A., Gutuleac E., Enicov I. (1994) *A Software Tool for Distributed System Evaluation* // Proceedings of the Symposium on Electronics and Telecommunications, vol. 3, Romania, Timisoara, Sept., 1994.

[5]. G. A. Bundell. (1997) *An FPGA implementation of the Petri Net firing algorithm.* In Proc. 4th Australasian Conf. on Parallel and Real-Time Systems, pp. 434-445.

[6]. John Morris et al. (2000) *A Re-configurable Processor for Petri Net Simulation.* Proceedings of the 33rd Hawaii International Conference on System Sciences.

[7]. http://www.altera.com/.

[8]. E. Gutuleac, C. Bosneaga, A. Reilean. (2002) *VPNP – Software Tool for Modeling and Performance Evaluation Using Generalized Stochastic Petri Nets.* 6th International Conference on DAS – 2002, Suceava, Romania, May 23-25.

[9] Wikarski, D. (1996) *Petri Net Tools: a Comparative Study*, ISST-Bericht 39/96 of the Fraunhofer-Gesellschaft e.g., Berlin.

[10]. T. Murata (1989) *Petri Nets: properties, analysis and applications* Proceedings of IEEE, vol. 77, pp. 541-580, Apr. 1989.