7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

A HARDWARE IMPLEMENTATION OF PETRI NETS MODELS

Viorica SUDACEVSCHI, Ludmila GUTULEAC, Victor ABABII
Technical University of Moldova
Str. Stefan cel Mare, 168, Chisinau
Republic of Moldova
ababii@mail.utm.md, svm@mail.utm.md

Abstract. *This paper presents the structure of a reconfigurable FPGA system for Petri Net simulation. The system is described as the interactions between processed elements that are functionally defined. According to the interconnection mode of the processed elements Petri Net models with any complexity can be implemented. The processed elements have a flexible architecture that allows their adaptation to the simulation of Petri Net model. The processed elements architecture and their interconnection mode gives the possibility to reduce the execution time that is necessary for reachability analysis of Petri Net.*
Keywords: *Petri Net simulation, Petri Net Hardware implementation.*

Introduction

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 multicomputer 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].

Petri Net Model

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 Petri-net is a 5-tuple $N= \{P, T, Pre, Post, M_0\}$, where:

$P= \{P_1, P_2, ..., P_n\}$ is a finite and non-empty set of places;
$T= \{T_1, T_2, ..., T_m\}$ is a finite and non-empty set of transitions satisfying $P \cup T \neq \varnothing$ and $P \cap T = \varnothing$;
Pre: $P \times T \rightarrow N^+$ - is a function of forward incidence;
Post: $T \times P \rightarrow N^+$ - is a function of backward incidence:
M_0: $P \rightarrow N^+$ is the initial marking.

A marking M can be interpreted as an integer vector which includes per place p one element which correspond to the number of tokens on place p. The marking describes the state of the adequate dynamic system, and dynamic changes are modeled as tokens movement from one place to another.

The sets of input and output places for a transition are noticed as ${}^{*}t$ and t^{*}. The sets of input and output transitions for a place are noticed as ${}^{*}p$ and p^{*}.

Transition $t \in T$ is enabled in marking M, if $\forall p \in P : M(p) \geq (p,t)$. Any transition enabled in M_i can fire, changing the marking of any $p \in P$ to marking $M_{i+1}(p)=M(p)-(p,t)+(t,p)$.

A marking M is reachable from the initial marking M_0 if there exists a sequence of firings that transform M_0 to M. A set of all reachable markings in the net forms a reachability set (RS). If it is accompanied by the transition relation then a reachability graph (RG) is formed.

A Petri-net is said to be k-bounded if for all reachable markings, the number of tokens in any place is less than or equal to k. A 1-bounded net is said to be safe.

The advantages of FPGA implementation

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, Verilog, ABEL, PALASM, Max+Plus) 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].

Structure of the system

The general structure of the system (Figure1) contains: PC – personal computer; I – interface; HPN - hardware implementation of Petri Net.

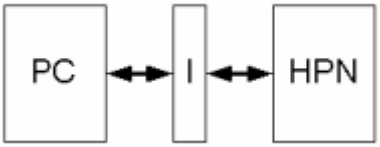

Figure 1. The general structure of the system

Hardware Petri Net Architecture

HPN (Figure 2) consists from: PCC – communication channel with PC; SYN – synchronization block for transition firing control; RAM – memory for markings and places storage; P_i – place simulation block; T_i – transition simulation block; C – communication lines.

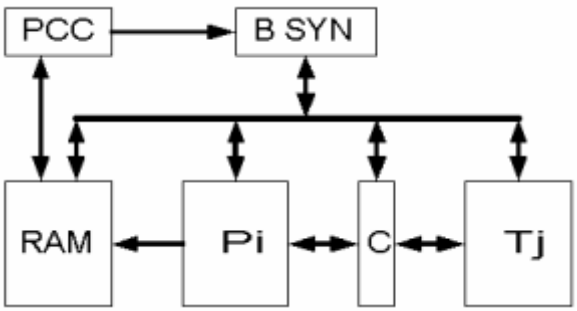

Figure 2. Hardware Petri Net Architecture

P_i (Figure 3) has the following inputs and outputs:

SET, RESET – set and reset inputs for place p_i; *W(t,p)* – the set of input arc weights for place p_i to increase the number of tokens; *W(p,t)* – the set of output arc weights from place p_i to decrease the number of tokens; $M(p_i)$ – the number of markings in place p_i; *F(Pi)* – the set of firing conditions, where:
For direct arcs,

$$F^d(Pi)=\{f_i\},\ f_i=\begin{cases}0, M(p_i)<W(p,t)\\ 1, M(p_i)\ge W(p,t)\end{cases}.$$

For inhibited arcs,

$$F^i(Pi)=\{f_i\},\ f_i=\begin{cases}1, M(p_i)<W(p,t)\\ 0, M(p_i)\ge W(p,t)\end{cases}.$$

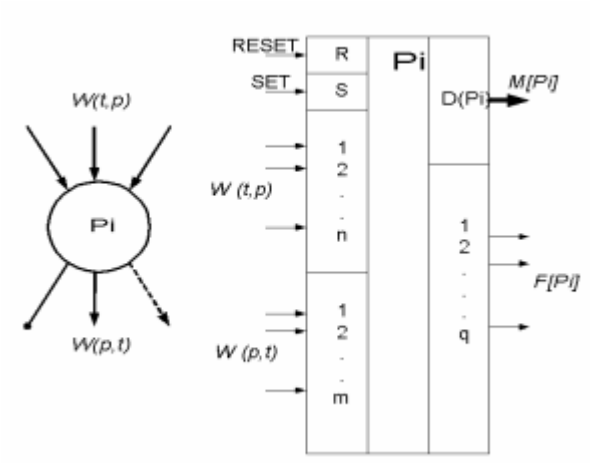

Figure 3. The Petri Net place module

For test arcs,

$$F^t(Pi)=\{f_i\},\ f_i=\begin{cases}0, M(p_i)<W(p,t)\\ 1, M(p_i)\ge W(p,t)\end{cases}.$$

$$F(Pi)=F^d(Pi)\wedge F^i(Pi)\wedge F^t(Pi).$$

If $F^i(Pi)$ or $F^t(Pi)$ are not defined, then $F^i(Pi)=1$ and $F^t(Pi)=1$.
The RESET signal zeroes the p_i state. The SET signal loads in the state register the initial number of tokens. When *W(t,p)* is enabled the register value is increased by the number that correspond to the arc weight. When signal *W(p,t)* is enabled the register value is decreased by the number that correspond to the arc weight. Output $M(p_i)$ determines the number of tokens in the place p_i. Outputs $F(P_i)$ represent the result of the register state verification. If the number of tokens is larger then 0 then $M(P_i)=1$.
The algorithm for Pi performance:

```
BEGIN
  Procedure RESET;
  Procedure SET;
  Procedure INCREMENT_Pi
      BEGIN
      FOR I=1 TO N DO
      IF  W_i(t,p)=1  THEN INCREMENT(Pi);
                      ELSE CONTINUE;
      END;

 Procedure DECREMENT_Pi
      BEGIN
      FOR I=1 TO M DO
      IF  W_i(p,t)=1  THEN DECREMENT(Pi);
                      ELSE CONTINUE;
      END;

  Procedure CONDITION_Pi
      BEGIN
      FOR I=1 TO Q DO
      IF  M(Pi)≥W(p,t)  THEN  f_i=1;
                              ELSE  f_i=0;
      END;
END.
```

The transition simulation block is represented in Figure 4, where: SYN - synchronization signal that allows the transition firing; *F(Pi)* - a set of firing conditions. If al these inputs are equal to 1 the transition is enabled. *W(t,p)* and *W(p,t)* were described previously.

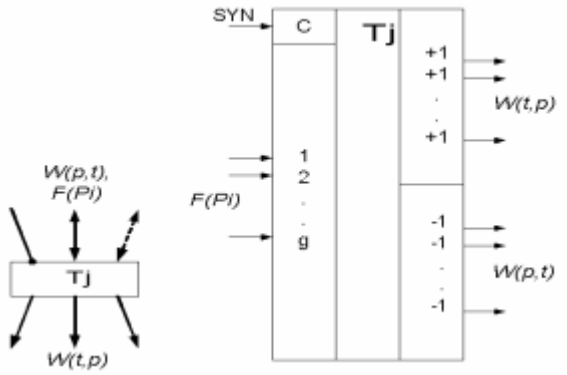

Figure 4. The Petri Net transition module

The RAM memory is presented in Figure 5, where WR RAM – enables the writing operation execution to the memory; $M(p_1)$ – $M(p_k)$ – state

inputs for current place; CT/RG ADR – the counter/register where the memory address is formed; PC Control – the unit for memory control by PC; Sel RAM – signal for memory selection for operation execution; CLR – zeroes the CT/RG ADR; Read RAM – enables the data reading operation and the increment of the CT/RG ADR; Data OUT – output data for PC.
The algorithm for Tj performance:

```
BEGIN
IF (F_i(Pi) = 1, i = 1,G) & (SYN = 1)   THEN
                    BEGIN
                    W(t, p) = 1 ,
W(p,t) = 1 );
                    END;
                    ELSE  CONTINUE;
END.
```

At CT/RG ADR inputs the state signals for each place are applied which generate the address of the memory cell for writing data operation. At each application of the writing signal WR RAM in the memory cell the transition code which contribute to the place state forming is memorized.

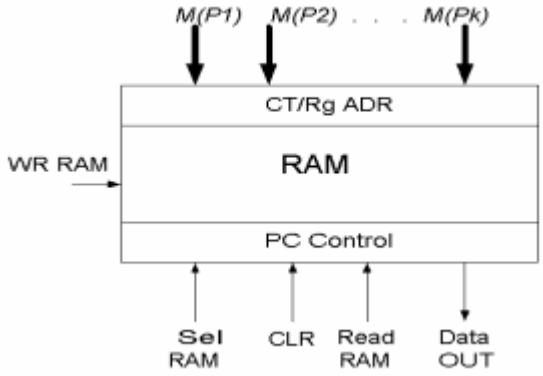

Figure 5. The RAM memory organization

PC Control port forms the signal Sel RAM which enables the memory operation. At each Read RAM signal the Data OUT outputs contain the transition code that is processed by PC. At the same time the CT/RG ADR is increment for next memory cell selection.

The algorithm for RAM performance:

```
BEGIN
     Procedure DATA_WRITE
      BEGIN
      WHILE WR_RAM=1 THEN
CT/Rg_ADR[M(pi)]=Tj;
      END;
     Procedure DATA_READ
      BEGIN
      (Sel_RAM = 1, CLR=1);
      ROR I=1 TO M(Pi) DO
      (Sel_RAM=1, Read_RAM=1, OUT
DATA);
      END;
END.
```

The synchronization block is presented in Figure 6, where:

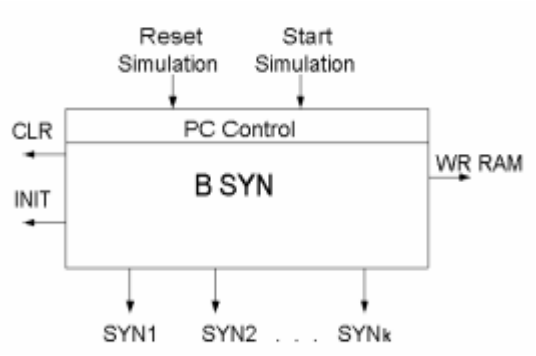

Figure 6. The synchronization block

CLR – the signal that zeroes the places P_i, i=1,2,..,N. INIT – initialization of the position's states; $SYN_1,\ldots,SYN_k$ – synchronization signals for transition firing; WR RAM - signal for writing operation in RAM memory; PC Control – the port for system control by PC; Reset Simulation – system initialization for the next simulation model; Start Simulation – the beginning of the simulation.

The algorithm for B SYN performance:

```
BEGIN
   Procedure RESET_SIMULATION;
   Procedure START_SIMULATION;
   Procedure SIMULATION


BEGIN
      CLR=1;
      INIT=1;
      FOR I=1 TO K DO
      BEGIN
      INIT=1;
      FOR J=1 TO K DO
      BEGIN
      OUT (SYN1, SYN2, …, SYNk);
WR_RAM=1;
      END;
```

```
        END;
        END;
END.
```

The table of synchronization signals creation is presented (Table 1).

Table 1. Table of synchronization signals

I,J	SYN_1	SYN_2	SYN_3		SYN_{k-2}	SYN_{k-1}	SYN_k
1,1	1	0	0		0	0	0
1,2	0	1	0		0	0	0
…	..	..	..		..	..	..
1,k	0	0	0		0	0	1
2,1	0	1	0		0	0	0
2,2	0	0	1		0	0	0
…	..	..	..		..	..	..
2,k	1	0	0		0	0	0
k,1	0	0	0		0	0	1
k,2	1	0	0		0	0	0
…	..	..	..		..	..	..
k,k	0	0	0		0	1	0

References

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

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

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

[4] Gavriliuc A., Gutuleac E., Enicov I. *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. *An FPGA implementation of the Petri Net firing algorithm.* In Proc. 4th Australasian Conf. on Parallel and Real-Time Systems, pp. 434-445, 1997.

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