Hierarchical Model of a Systolic Array for Solving Differential Equations Implemented as an Upgraded Petri Net

PERICA STRBAC
College of Computer Science
Megatrend University Belgrade
Bulevar umetnosti 29
SERBIA
strbac@beotel.rs

MILAN TUBA
Faculty of Mathematics
University of Belgrade
Studentski trg 16
SERBIA
tubamilan@ptt.rs

DANA SIMIAN
Department of Computer Science
Lucian Blaga University of Sibiu
5-7 dr. I. Ratiu str.
ROMANIA
d_simian@yahoo.com

Abstract: - The objective of this paper is to develop a hierarchical model of a systolic array which solves differential equations by using Upgraded Petri Net (UPN). The Upgraded Petri Net presents a formal modeling tool, based on mathematical apparatus, used for simulation and analysis of processes, particularly at the register transfer level. Original software for modeling and simulations of Upgraded Petri Nets, PeM (Petri Net Manager), is developed and used for all models described in this paper. This software supports: UPN formal theory, graphical modeling, simulation and analysis of an UPN model. This paper includes a preface on the UPN theory, the UPN models formulation, the UPN models of a processing element of a systolic array which solves differential equation, their simulation and analysis. The first hierarchical level refers to the high level systolic array modeling. This model shows processing elements that calculate or pass their input to the output. Thus, the next processing element uses the results from the previous one. The second level model is more detailed and refers to a single processing element. This model is capable to generate solution of a differential equation through UPN execution based on Taylor's method. This execution is based on parralel firing of a group of transitions. The results are stored into a *.mem file inside the PeM. The suitability of UPN for modelling of the systems at different levels of abstraction is examined and established.

Key-Words: - Upgraded Petri Net, Systolic Architecture, Modeling

1 Introduction

Upgraded Petri nets are a formal mathematical apparatus which enables modeling, simulation and process analysis [1]. They enable interactive monitoring of process operations and its gradual improvement from the initial phase, all the way to the final version.

The hierarchical structure of an UPN gives wide possibilities for abstraction. This feature of the UPN provides the model implementation consisting at the same time of elaborate pieces essential for the analysis at a certain level, and also of some general pieces whose details are irrelevant for the analysis at the given level of abstraction [2], [3].

This paper presents the usage of UPN for modeling, simulation and analysis of a processing element of a systolic array for numerical solving of differential equations [4], [5], [6]. The UPN model concerns the visualization of data flow and functional solving of a given differential equation. The model implements Taylor's method for numerical solving of differential equations. For given input parameters the model generates solutions of a differential equation in a *.mem file whose content represents the analyzed system's memory state (image).

2 UPN Formal Theory

Upgraded Petri nets formal theory is based on functions [1]. Upgraded Petri net is a 9-tuple:

$$\mathbf{C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)}$$

where:

$\mathbf{P=\{p_1,p_2,p_3,...,p_n\}, n>0}$
- a finite nonempty set of places $\mathbf{p_i}$

$\mathbf{T=\{t_1,t_2,t_3,...,t_m\}, m>0}$
- a finite nonempty set of transitions $\mathbf{t_j}$

$\mathbf{F:T\times P\rightarrow N_0}$ - Input Function;

$\mathbf{B:T\times P\rightarrow N_0}$ - Output Function;

$\mathbf{\mu:P\rightarrow N_0}$ - Marking Function:

$\mathbf{\theta:T\times\Delta\rightarrow\lambda}$ - Timing Function;

$\mathbf{TF:T\rightarrow A}$ - Transition Function;

$\mathbf{TFL:T\rightarrow N_0}$ - Transition Firing Level;

$\mathbf{PAF: P\rightarrow(x,y)}$ - Place Attributes Function;

The input function assigns a non-negative number to an ordered pair (t_i,p_j). The assigned non-negative number defines how many times the place p_i is input, as compared to the transition t_i. N_0 represents the set of non-negative numbers. The set of places which are input, as compared to the transition t_j, is presented as follows ${}^*t_j=\{ p_i \in P, F(t_j, p_i)>0\}$. For the presentation of the place $p_i \in {}^*t_j$ which have the standard input compared to the t_j, the sign $t_j {}^*t_j^S$ will be used, and sign $F^S(t_j, p_i)$ will be used for the such input function. For the places $p_i \in {}^*t_j$ with inhibitor input in relation to the t_j transition, the sign ${}^*t_j^I$ will be used and sign $F^I(t_j, p_i)$ for the input function.

The output function assigns a non-negative number to the ordered pair (t_i,p_j). The assigned number shows how many times the place p_i is input in relation to the t_i transition. The set of places which are input in relation to the t_j transition is presented as follows $t_j^*=\{ p_i \in P, F(t_j, p_i)>0\}$.

The marking function assigns a non-negative number to the p_i place. The marking function can be defined as *n*-dimension vector (marking): $\mu=(\mu_1,\mu_2,...,\mu_n)$, where $n=|P|$. Instead of sign μ_i the sign $\mu_{(pi)}$ can be used.

The timing function assigns the probability λ_j to the ordered pair (t_i, Δ^q_{tj}). The t_i transition can be fired at the interval Δ^q_{tj} with probability λ_j. The sign Δ^q_{tj} is the **q**-th interval of the t_j transition.

The transition function gives an operation $\alpha_j \in A$ to the t_j transition. Sign **A** is the set of operations which can be assigned to the transition.

The firing level function of the transition assigns a non-negative number to the transition t_j. If the number not equals zero, it shows the number of $p_i \in {}^*t_j$ places takes part in the transition firing, and if the number equals zero, then all the places $p_i \in {}^*t_j$ affect the t_j transition firing.

Place attributes function assigns an ordered pair **(x,y)** to the place p_i. The **x** component is a real number called **x attribute**, and **y** is a non-negative number called **y attribute** (i.e. $x \in R$, $y \in N_0$). Over the x attribute belonging to the $p_i \in {}^*t_j$ places, the α_j operation assigned to the t_j transition executes where the order of operands in the operation α_j is defined by the y attributes which belong to the p_i place in accordance to TFL function take part in the transition firing.

2.1. An Operation Assigned to a Transition

Function **TF** assigns to a transition t_j one operation. This operation can be: arithmetical, operation, logical operation or file operation [1]. Inside the suite PeM a file which is target of file operation function has an **.mem* extension. This **.mem* file is a text file and it is used for simulation of computer system memory. One line inside the **.mem* file refers to context of one memory location of computer system that we are modeling. The first column of the line, up to an empty char represents address of a memory location, while the second column represents a value of the memory location.

An arithmetical operation $\alpha_j \in A$, which is assigned to the transition $t_j \in T$, uses attributes x which belong to the places $p_i \in {}^*t_j$ as operands of that operation. A result of an arithmetical operation $\alpha_j \in A$ will be placed into the attributes x which belong to the places $p_i \in t_j^*$. The order of an operand (i.e. order of attributes x which belong to the places $p_i \in {}^*t_j$) in an arithmetical operation $\alpha_j \in A$ is defined by attributes y which belong to the places $p_i \in {}^*t_j$.

A logical operation $\alpha_j \in A$ which is assigned to the transition $t_j \in T$, uses attributes x which belong to the places $p_i \in {}^*t_j$ as operands of that operation. If a result of the logical operation $\alpha_j \in A$ is logical false (FALSE) the transition $t_j \in T$ is disabled and will stay in that state until the result of this logical operation $\alpha_j \in A$ becomes logical (TRUE). The order of an operand (i.e. order of attributes x which belong to the places $p_i \in {}^*t_j$) in a logical operation $\alpha_j \in A$ is defined by attributes y which belong to the places $p_i \in {}^*t_j$.

A file operation $\alpha_j \in A$ which is assigned to the transition $t_j \in T$ performs over the context of a file which extension is equal to **.mem*. This **.mem* file is a text file and it is used for simulation of computer system memory. A File Operation $\alpha_j \in A$ addresses context of a **.mem* by using attribute x which belongs to the places $p_i \in {}^*t_j$. A result of this operation $\alpha_j \in A$ changes the value of attributes x which belong to the places $p_i \in t_j^*$. The result also can change attributes y which belong to the places $p_i \in t_j^*$, or can change context of addressed line into the **.mem* file.

2.2. UPN Graph

Upgraded Petri Net is represented via formal mathematical apparatus or graphically. Upgraded Petri Net is represented by bipartite multigraph [1].
A graph of Upgraded Petri Net is 2-touple:

$$G=(V,A)$$

where:

$$V=\{v_1, v_2, ..., v_s\}$$

represents set of nodes which consists of the two disjoint subsets (set of places and set of transitions):

$$V=P\cup T,\ P\cap T=\varnothing$$

$$A=\{a_1, a_2, ..., a_r\}$$

represent set of arcs where:

$$\text{if} \quad \forall\, a_i \in A,\ a_i=(v_j,v_k)$$

$$\text{then} \quad (v_j\in P \wedge v_k\in T) \vee (v_j\in T \wedge v_k\in P)$$

Graphical representation of an Upgraded Petri Net includes: places, transitions, marking, input function and output function.

Fig. 1 shows an example of graphical representation of elements of Upgraded Petri Net. A place is represented by circle. A transition is represented by short bold or tiny line. A marking $\mu_{(pi)}$ is represented by dots inside the place p_i. The number of dots in the place p_i is equal to $\mu_{(pi)}$. The term marker is used as a synonym for dot.

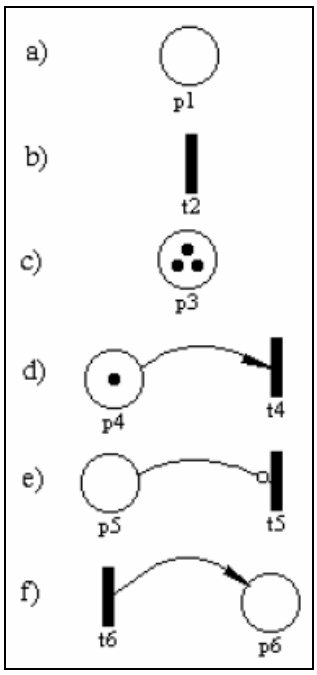

Fig. 1. A graphical representation of the elements of UPN: a) place b) transition c) marking d) input function with standard arc e) input function with inhibitor arc f) output function

Input and output functions are represented by directed arcs. There are two type of arcs which are used in UPN-graph:

- an input arc (which is directed from a place to a transition)
- an output arc (which is directed from a transition to a place).

There are two type of input arcs which are used in UPN-graph:

- standard arc (which is ended by arrow)
- inhibit arc (which is ended by small circle).

If an input function $F(t_j,p_i)>0$ then there is one or more arcs between the place p_i and the transition t_j. If an output function $B(t_j,p_i)>0$ then there is one or more arcs between the transition t_j and the place p_i. If the value of an input or an output function is more than 1 then the value of multiplicity of arc is shown over the arc (input arc or output arc).

Fig. 2. shows a model of a simple adder. The places are labeled as: *p-1*, *p-2* and *p-3,* respectively. A transition is labeled as *t-1*. An input function is between the place *p-1* and the transition *t-1*, as is between the place *p-2* and the transition *t-1*. An output function is between the transition *t-1* and the place *p-3*.

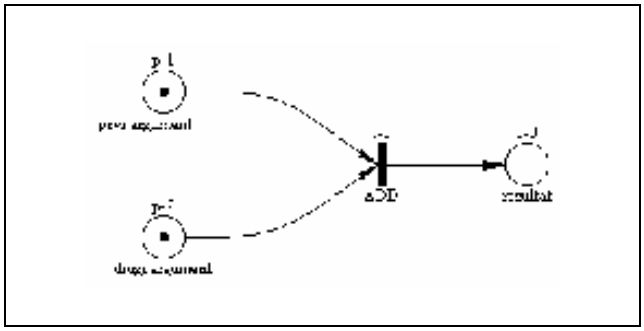

Fig. 2. UPN-graph, model of simple adder

In graphical representation of an Upgraded Petri Net timing function, transition function, transition firing level function and place attributes function can be shown by using labels. Inside the PeM software these functions are accessible via popup menu, and they are not shown on the graph.

2.3. UPN Execution

An Upgraded Petri Net execution represents change of system state from the current state to the next state. This migration from one state to the other one is triggered by firing of the transitions. By UPN execution:

- marking vector can change
- contents of *.mem file can change
- attributes which belong to the places $p_i\in t_j^*$ of fired transition t_j can change

A transition $t_j\in T$ can be enabled (flammable) in Upgraded Petri Net [1]:

$$C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)$$

if the next 3 conditions are true:

1. condition, (1)

if timing function $\theta(t_j, \Delta^q_{tj}) > 0$ over an interval Δ^q_{tj} where transition t_j can become flammable

2. condition, (2)

if $TFL(t_j)>0$,
then,
$(\#p_i \in {}^*t_j)+(\#p_k \in {}^*t_j)= TFL(t_j)$

or,

if $TFL(t_j)=0$,
then,
$(\#p_i \in {}^*t_j) + (\#p_k \in {}^*t_j) = |{}^*t_j|$

where:

$\#p_i$ represents number of places $p_i \in {}^*t_j^S$ which satisfied: $\mu_{(pi)} \geq F^S(t_j, p_i))$

$\#p_k$ represents number of places $p_i \in {}^*t_j^I$ which satisfied: $\mu_{(pk)}=0$

3. condition, (3)

if a logical operation $\alpha_j \in A$ assigned to the transition t_j
then,
the result of the logical operation α_j must be equal to true (TRUE).

A marking vector μ will be changed to new marking vector μ' by firing of transitions t_j where:

$$\mu'_{(pi)} = \mu_{(pi)} - F(t_j, p_i) + B(t_j,p_i) \quad (4)$$

for places $p_i \in {}^*t_j^S$

and

$$\mu'_{(pk)} = \mu_{(pk)} + B(t_j,p_i) \quad (5)$$

for places $p_k \in {}^*t_j^I$

By firing of the transition t_j an arithmetic operation is executed or a file operation is executed with respect to the operation that is assigned to t_j by function $TF(t_j)$.

A logical operation which is assigned to the transition t_j by function $TF(t_j)$ will be executed if conditions (1) and (2) related to t_j are equal to true.

2.4. Conflict Solving in UPN

A conflict in an Upgraded Petri Net influences UPN-execution. A conflict in UPN occurs if the following inequality is satisfied:

$$0 < \mu_{(pi)} < \sum_{j=1}^{|T|} F(t_j, p_i) \quad (6)$$

for places $p_i \in {}^*t_j^S$.

If a conflict occurred then marking vector μ has such value that all transitions which have satisfied conditions (1), (2), (3) cannot be fired concurrently, or if there is one transition from the set of flammable transitions which prevents at least one transition from the set of flammable transitions to fire.

After the firing of transitions which satisfy (1), (2) and (3) a change from marking μ^k into the new marking μ^{k+1} is determined by next state function (NSF) ψ . This function is defined as:

$$\psi(\mu^k, T^k) = \mu^{k+1} \quad (7)$$

where

- T^k, represents a set of all transitions where conditions **(1)**, **(2)** and **(3)** are satisfied for marking μ^k
- μ^k, represents marking before firing of the transitions which belong to set T^k;
- μ^{k+1}, represents marking after firing of the transitions which belong to set T^k;

A set of maximal sets of transitions where these transitions can be fired concurrently for marking μ^k refers to as T^k_{max} :

$$T^k_{max} = \{T^X_1, T^X_2, T^X_3, \ldots, T^x_q\} \quad (8)$$

The conflicts in Upgraded Petri Net can be solved as follows:

- find maximal sets of transitions where these transitions can be fired concurrently;
- choose one of the maximal sets of transitions where these transitions can be fired concurrently;
- probability of choosing for any maximal set have the same value.

An example of a conflict in UPN is shown on Fig. 3. In this example maximal sets of transition where these transition can be fired concurrently are: **{t1,t3}** and **{t2}**.

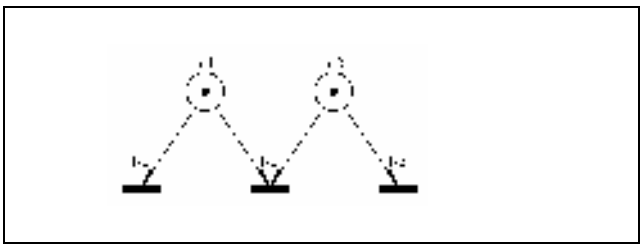

Fig. 3. An example of a conflict in UPN

In this example number of maximal sets of transitions where these transitions can be fired concurrently are two. Now, probability of choosing any maximal set is equal to 0.5.

2.5. UPN Reachability Tree

An UPN reachability tree graphically represents all possible marking vectors which can occurs during an UPN execution for given initial marking. Reachability tree shows all states which model can reach from the initial state.

Nodes of reachability tree represent marking vectors. A root of the reachability tree represents marking vector μ^0 which refers to initial marking. Initial marking μ^0 represents initial state of the modeled system.

Reachability tree includes three type of nodes:

- inner node, graphically representation is □
- dead node, graphically representation is ⊠
- double node, graphically representation is ◎

During the generating of reachability tree boundary nodes can occur. A boundary node is μ^0 and any node μ^X which is processing. A processing of a node μ^X determines does the node is inner node or dead node or double node.

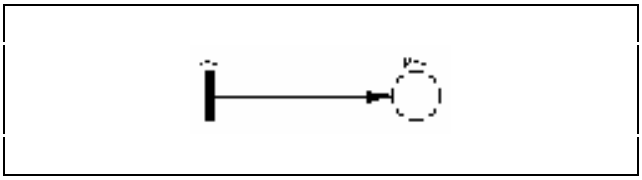

Fig. 4. UPN model which generates infinite number of markers

A term **ω** refers to an infinite number of markers. Fig. 4. shows an example of UPN model which generates an infinite number of markers. Initial vector marking is $\mu^0=(0)$, so by every firing of transition *t-1* value of marking vector increases. This firing makes the next sequence of marking::

$$\mu^1=(1),\ \mu^2=(2),\ldots,\ \mu^\infty=(\infty)$$

Number of firing of the transitions t_j can be adjusted via conditions **(1)** and **(3)** so model can generate wanted number of markers.

An UPN reachability tree is generated by the next algorithm [1]:

1. let μ^X is a boundary node (node which is in processing);
2. **if** there is node μ^Y in reachability tree and that node is not boundary and $\mu^X=\mu^Y$, **then** μ^X is a double node;
3. **if** there is not the transition t_j which satisfied conditions **(1)**, **(2)**, **(3)** for marking μ^X, **then** μ^X is a dead node;
4. for every transition t_j which satisfied conditions **(1)**, **(2)**, **(3)** for marking μ^X, new boundary nodes will be created while μ^X becomes an inner node;
5. let μ^Z be a new boundary node which is generated in step 4. For every position ***i*** of the node μ^Z check:
   a) if $\mu^X_i=\omega$ then $\mu^Z_i=\omega$,
   b) if there is node μ^C on the path from root node of the reachability tree to the node μ^X where:
      $\mu^C < \psi(\mu^X, t_j)$ **i** $\mu^C_i < \psi(\mu^X, t_j)_i$
      then
      $\mu^Z_i=\omega$
   c) if not satisfied a) or b)
      then $\mu^Z_i < \psi(\mu^X, t_j)_i$
6. if there is a node in reachability tree which is not processed go to the step 1.

2.6. UPN Flammability Tree

UPN execution refers to a concurrent firing of the flammable transitions. UPN execution generates a UPN flammability tree. A UPN flammability tree is such tree where a node of the tree is a set of transitions which are fired at the same time. If there is a same node as the current node in the flammability tree then generating of the flammability tree will be stopped.

There are four types of nodes in a flammability tree: root node, double node, dead node, and inner node (as in a UPN reachability tree).

3 An UPN Model Formulation

A systolic array [7], [8], [9], [10] belongs to MISD (Multiple Instruction Stream Single Data Stream) architecture in the sense of subdivision based on the instruction flow and data flow [11]. UPN model of a systolic array for numerical solving of a differential equation is given in Fig. 5. The shown UPN model has the properties of systolic architecture: synchronism, regularity, time locality and pipelining [12].

Places *p-1, p-2* and *p-3* model initial conditions for solving a differential equation for x_0, y_0 and h respectively.

Assume that $\{x_n\}$ is a set of points defined as

$$x_n = x_0 + nh \text{ , where } h>0.$$

If we use Taylor series as a representation of a solution of differential equation

$$y'(x) = f(x, y(x)), \quad y(x_0) = y_0$$

as an infinite sum of terms calculated from the values of its derivates at a single point then we have:

$$y(x_{n+1}) = y(x_n + h) = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \cdots \\ \ldots + \frac{h^p}{p!}y^{(p)}(x_n) + \frac{y^{(p+1)}(\xi)}{(p+1)!}h^{p+1} \quad (9)$$

and if we use denotation:

$$y'(x) = f(x,y) = f^{(0)}(x,y)$$

formula (9) takes the form:

$$y''(x) = \frac{\partial f}{\partial x}(x,y) + \frac{\partial f}{\partial y}(x,y)y'(x) =$$

$$= \frac{\partial f}{\partial x}(x,y) + \frac{\partial f}{\partial y}(x,y)f(x,y) = f^{(1)}(x,y)$$

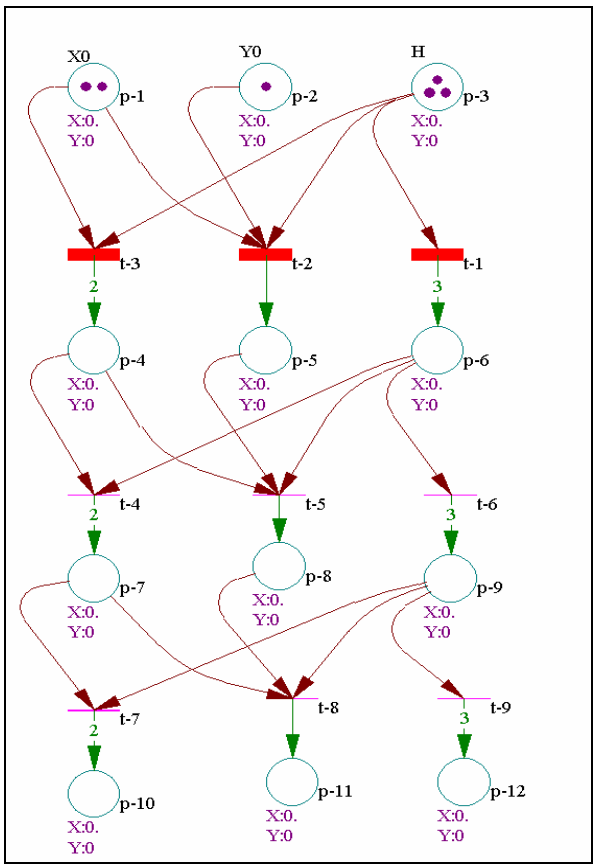

Fig. 5. UPN model of a systolic field for numerical solving of differential equations

In general it is:

$$y^{(k+1)}(x) = f^{(k)}(x,y) =$$

$$= \frac{\partial f^{(k-1)}}{\partial x}(x,y) + \frac{\partial f^{(k-1)}}{\partial y}(x,y)f(x,y), \quad k = 0,1,2,...$$

Let $y(x_{n+1})$ denotes value of solution y at point x_{n+1} and let y_{n+1} denotes approximation of that value given by using Taylor series around point x_n. Then we have:

$$y_{n+1} = y_n + hf^{(0)}(x_n,y_n) + \frac{h^2}{2!}f^{(1)}(x_n,y_n) + \cdots + \frac{h^p}{p!}f^{(p)}(x_n,y_n)$$

Further, we use denotation:

$$T_p(x,y) = f(x,y) + \frac{h}{2!}f^{(1)}(x,y) + \cdots + \frac{h^{p-1}}{p!}f^{(p-1)}(x,y)$$

Now we have:

$$y_{n+1} = y_n + hT_p(x_n,y_n)$$

By using the Taylor's method where $p = 2$, the solution of a differential equation takes the form:

$$y_i = y_{i-1} + h \cdot f(x_{i-1},y_{i-1}) + \frac{h^2}{2}f'(x_{i-1},y_{i-1}) \qquad (10)$$

where:

y_i → solution of a differential equation for $x_i = x_0 + i\,h$

x_{i-1} → values $x = x_0 + (i-1)\,h$

y_{i-1} → solution of a differential equation for $x_{i-1} = x_0 + i\,h$

h → step for x in every iteration

i → iteration number

$f(x_{i-1},y_{i-1})$ → derivation y by x at point $A(x_{i-1}, y_{i-1})$

$f'(x_{i-1},y_{i-1})$ → first derivation of f by x at point $A(x_{i-1}, y_{i-1})$

The initial marking of the model implies that the initial conditions are given, the value of x_o, value of the function $y_0(x_0)$ and step h (these values are uppercase in the Fig. 5).

Transition *t-3* models calculating the next value of variable x. Calculating the y function using Equation (9) is modeled with transition *t-2*. Passing of parameter h through to the next processing element of the systolic array is modeled with transition *t-1*.

Transitions *t-4* and *t-7* are equivalent to transition *t-3*. Also, transitions *t-5* and *t-8* are equivalent to transition *t-2*, while transitions *t-6* and *t-9* are equivalent to transition *t-1*. These equivalent transition pairs are in relation to the second and the third systolic array processing elements.

With the execution of the UPN model shown in Fig. 5. a parallel firing of group of transitions*{t-3, t-2, t-1}, {t-4, t-5,t-6}, {t-7, t-8, t-9}* occurs. This models the execution of operations in every processing element of the systolic array.

The result of the operation in a given processing element of systolic array depends on the result in the previous processing element of the systolic array. This models a recursive Taylor's function from Equation (10). Reachability tree represents a set of all possible states that could occur while the UPN model is executing with the given initial marking.

By executing the UPN model shown in Fig. 5 a dead node is reached. This node represents a state when there are no flammable transitions.
In this state places *p-2, p-5, p-8* and *p-11 m*odel solutions $y(x_0)$, $y(x_0+h)$ and $y(x_0+2h)$, $y(x_0+3h)$ respectively.

3 UPN Model of a Processing Element

Fig. 6. shows a part of UPN model of systolic array processing element which sets up initial value to y_0, calculates a part of Equation (9) i.e.

$$y_i = y_{i-1} + h \cdot f(x_{i-1}, y_{i-1}) + \frac{h^2}{2} f'(x_{i-1}, y_{i-1})$$

then saves result into the modeled memory and passes y result to the next processing element. Place *p-1* models initial value of y_0, i.e. input value of function y which has been calculated in the previous step. Transition *t-2* models ADD operation in Equation (10).

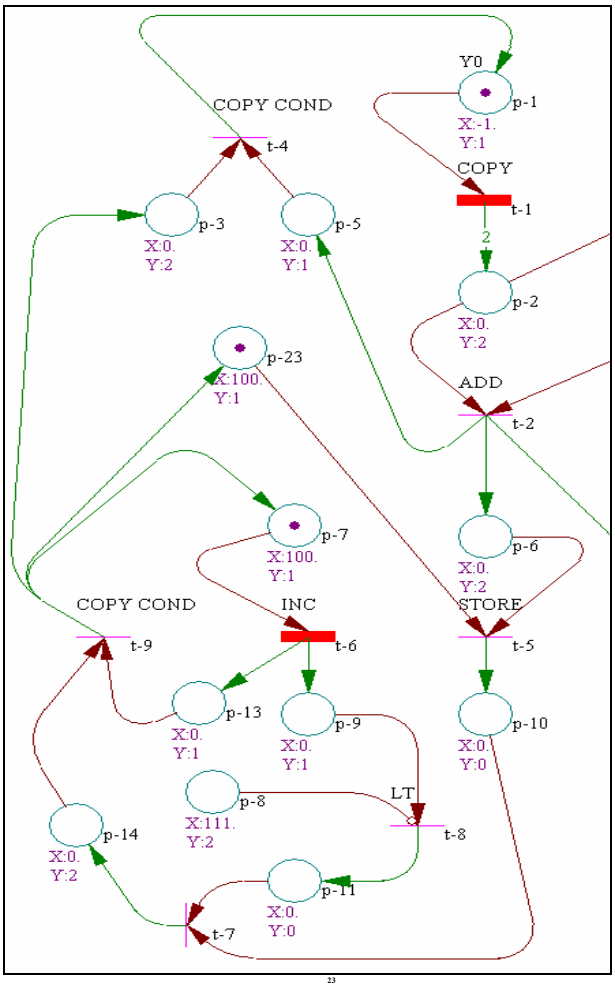

Fig. 6. A-part of the UPN model of a processing element of a systolic array

Transition *t-5* models saving value y into the modeled memory. This transaction attaches TF - transition function STORE. Value of attribut x of place *p-23* models start memory location. The results are saved into the file *.mem whose image refers to a pair memory location – value. Transition *t-6* increments address to saving result.

Attribut x of place *p-8* models value of upper border x neded for calculating value of function y. Transition *t-8* models a testing if the model reached end of given x value interval. This transition attaches TF equals to LT (less than) function. Input places to this function are place *p-8* and place *p-9*. Attribut x value of the place is equal to value of memory address where result will be saved. Synchronization of result saving, address incrementing and address testing, where the result is saved, is realized by place *t-7*. If conditions for further iteration are true then by firing of place *t-7* place *t-9* becomes flammable. After its firing new value of address will be saved as new value of y, and place *t-4* becomes flammable.

By firing of place *t-4* attribut x of place *p-1* gets value equal to the value of function y. This value is the next input into the next processing element of systolic array.

Fig. 7. shows a part of UPN model of a processing element of systolic array which refers to modeling of function *f(x,y)*, calculating the value in the Equation (10), and place firing sequence synchronizing. Transition *t-11* models function *f(x,y)* by using appropriate transition function. Input places for this transition are *p-18* and *p-17*.

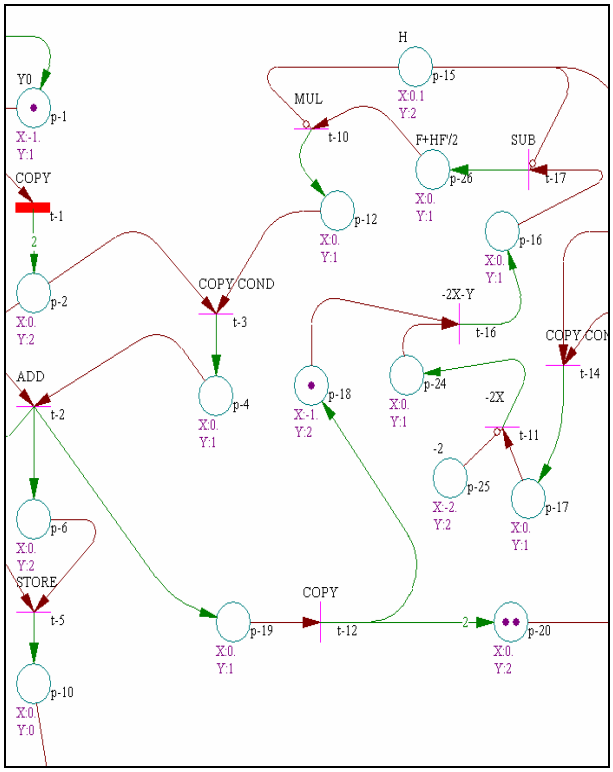

Fig. 7. B-part of the UPN model of an processing element of a systolic array

Attribut x of place *p-18* models y_{i-1} value, while attribut x of place *p-17* models x_{i-1} value. Initial marking of place *p-18* is equal to 1. This marking means that the UPN model will wait for x value to be calculated for given step of computing result of differential equation. By firing of transition *t-11*,

transition *t-16* becomes flammable. Firing sequence of transitions *t-16, t-17* prepares a value equal to $f(x_{i-1},y_{i-1})+h\cdot(f'(x_{i-1},y_{i-1}))/2$. This value is one input to transition *t-10*.

Transition *t-10* models calculating product $h\cdot f(x_{i-1},y_{i-1})+h^2\cdot(f'(x_{i-1},y_{i-1}))/2$. Attribut *x* of place *p-15* models value of step *h*. This value is a constant so we are using inhibit input arcs from place *p-15* to relevant transitions (*t-10* and *t-13*). By firing of transition *t-10* transition *t-3* remains flammable. If marking of place *p-2* equals to 2, firing sequence of transitions *{t-3},{t-2}* is initiated.

Now, *x* attribut of place *p-6* and *x* attribut of place *p-19* have y_i value. By firing transition *t-5* this value will be passed to the part of model shown in Fig. 6 earlier described. By firing of transition *t-12* value y_i will be copied into the *x* attribut of place *p-18* so result will be passed to the next processing element of systolic array.

Fig. 8. shows a part of UPN model of a processing element of a systolic array which refers to synchronization of calculating function $f(x_{i-1},y_{i-1})$ and calculating value x_{i-1}.

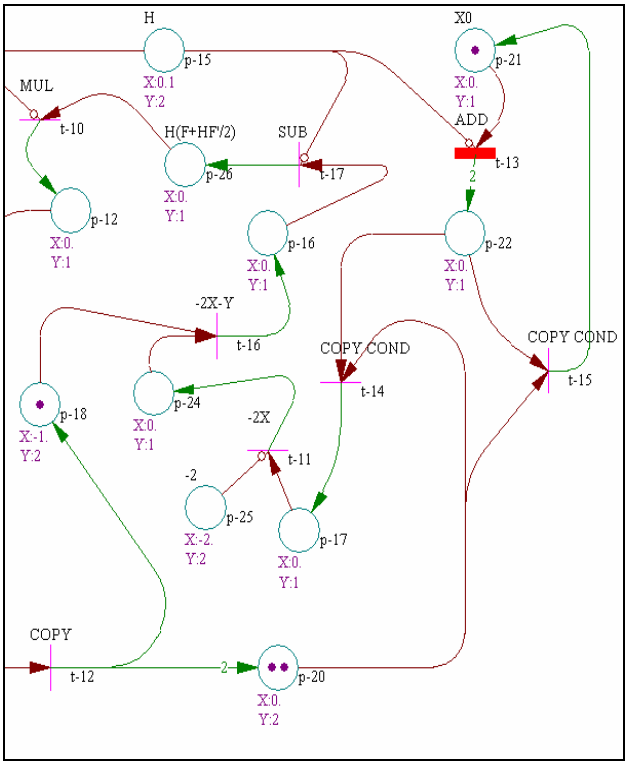

Fig. 8. C-part of the UPN model of an processing element of a systolic array

Initial marking is shown in Fig. 8. which refers to shown part (C-part) of the model. Attribut *x* of place *p-15* models value of step *h* in Equation (10). It corresponds to the constant *h* so this place is connected to transition *t-13* by using inhibit input arc. Place *p-21* has marking equals to 1, so it models existence of start condition x_0. This x_0 value is equal to the value of *x* atrribut of place *p-21*. Start condition y_0 is modeles by place *p-18*. Place *p-20* has marking equal to 2, so it synchronizes passing of *x* value for calculating Equation (10).

By executing of UPN model shown in Fig. 8. place *t-13* fires. Now, marking of the place *p-22* equals 2 because of double output arc. Attribut *x* of place *p-22* has value of summ x_0+h. At this moment, group of transitions *{t-14, t-15}* is flammable.

By firing of the group transitions value x_0+h will be copied into the *x* attribute of place *p-17* and *x* attribute of place *p-21* and marking of place *p-20* will equal to 0. Now, marking of place *p-17* equals to 1, transition *t-11* is flammable, so, all data for calculating function *f(x,y)* are ready.

Transition *t-13* remains flammable and new *x* value will be calculated. This value equals to $(x_0+h)+h$ now, but further passing of the value is disabled while marking of place *p-20* is different from 2.

Marking of place *p-20* becomes 2 after firing of transition *t-12*. This firing models end of one cycle of generating value *y* i.e. models end of one step of result generating.

4 UPN Model Analysis

By executing the UPN model for given initial marking shown on Fig.s 2, 3 and 4 the next sequence of transitions firing will happen :

{t1, t6, t13} → {t8, t14, t15} →
{t11, t13} → {t16} → {t17} →
{t10} → {t3}→ {t2} →
{t5, t12} → {t7, t14, t15} →
{t9, t11, t13} → {t16} →
{t17} → {t4, t6, t16} →
{t1, t8,t17} → {t10} →
{t3} → {t2} → {t5, t12}...

The cycle

{t3} → {t2} →
{t5, t12} → ... → {t3} →
{t2} → {t5, t12} ...

repeats until the end of given *x* value interval is reached. This end *x* value is modeled by attribut *x* of place *p-8*. When the end is reached, model reached dead node by firing of set of parallel flammable transitions *{t5, t12}*. This situation means the end of calculating results of differential equation for given interval and initial conditions.

UPN model shown in this paper generates solution of differential equation:

$$\frac{dy(x)}{dx} = -2x - y(x) \qquad (11)$$

for given initial conditions

$$x_0 = 0$$
$$y_0 = y(x_0) = -1 \qquad (12)$$

in the interval [0,1] by using step h=0.1.

Table 1. shows content of the file *.mem. The first column represents address of memory location where a result is saved, the second column represents value of the result and the third one represents x value.

Address	y(x)	x
100	-1.000000	0.0
101	-0.915000	0.1
102	-0.857075	0.2
103	-0.823653	0.3
104	-0.812406	0.4
105	-0.821227	0.5
106	-0.848211	0.6
107	-0.891631	0.7
108	-0.949926	0.8
109	-1.021683	0.9
110	-1.105623	1.0

Table 1. Contents of file * .mem

Exact solution for differential equation (11) with initial conditions (12) is:

$$y(x) = -3e^{-x} - 2x + 2 \qquad (13)$$

Table 2. compares exact solution with results generated by UPN model execution.

Exact solution	Solution given by UPN model	Value x
-1.000000	-1.000000	0.0
-0.914512	-0.915000	0.1
-0.856192	-0.857075	0.2
-0.822455	-0.823653	0.3
-0.810960	-0.812406	0.4
-0.819592	-0.821227	0.5
-0.846435	-0.848211	0.6
-0.889756	-0.891631	0.7
-0.947987	-0.949926	0.8
-1.019709	-1.021683	0.9
-1.103638	-1.105623	1.0

Table 2. Comparative results, exact solution and solution generated by UPN model

The results given by UPN model are as expected because we are using Taylor's method which has good accuracy.

Fig. 9. shows graphical insight which refers to relation between exact results and results generated by the UPN model for given data (11) and (12).

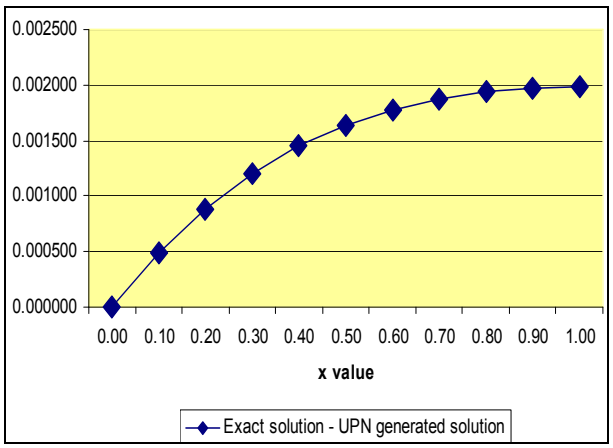

Fig. 9. Graphical representation of difference between exact solutions and solutions generated by UPN model

5 Conclusion

The aim of this paper was to establish suitability of Upgraded Petri Nets for hierarchical modeling of systolic array for solving differential equations. Two models at different levels are presented. The first model refers to a general representation of systolic architecture which is able to solve differential equations by using Taylor's method. This method is chosen as an example for implementation. The second UPN model refers to a single processing element of a systolic array. This UPN model can generate results for given differential equation and given initial conditions. During execution of given UPN model of a processing element of systolic array the model passes through many states. These states model properties of systolic architecture: synchronism, regularity, time locality and pipelining.

Correctness of given UPN model was checked by execution of the model and the results generated dynamically during concurrent firing of set of transitions. At the end of execution all results are saved into the *.mem file.

In the iterative cycle *modeling - simulation – analysis – improving model* a model can be reached which represents system that could be physically implemented. UPN facilitated a complete system formation and specification. The hierarchical structure of UPN allows great possibilities of abstraction. This feature of UPN facilitates construction of the model that consists at the same time of elaborate pieces essential for the analysis at a certain level, and also of some general pieces

whose details are irrelevant for the analysis at the given level of abstraction. Original software for modeling and simulations of Upgraded Petri Nets, PeM (Petri Net Manager), is developed and used for all models described in this paper.

References:

[1] Štrbac, P. S., Đurašinović, M., "A Modification of the DSP TMS320C30 Organization Using One Original Petri Nets Based Simulation Analysis", *3rd Balkan Conference on Operational Research*, pp. 1157-1170, 16-19 October 1995.

[2] J. L. Peterson, *"Petri Net Theory and Modeling of Systems",* Englewood Cliffs, , Prentice Hall, New York 1981.

[3] E. Antonidakis, "Conferencing Protocols and Petri Net Analysis", pp.3112-3119, *WSEAS Transactions on Computers*, Issue 1, Volume 4, January 2005

[4] G. F. Simmons, J. S. Robertson, *"Differential Equations with Applications and Historical Notes"*, 2nd Ed., McGraw-Hill, Inc., New York, 1991.

[5] Khalil Shihab, *"Simulating ATM Switches Using Petri Nets"*, pp.1495-1502, *WSEAS Transactions on Computers*, Issue 11, Volume 4, November 2005

[6] Leopoulos Vrassidas, "Integrated modeling & simulation of customer order process, production order process and discrete production system, for production planning evaluation using Petri Nets", pp.237-241, *WSEAS Transactions on Computers*, Issue 1, Volume 2, January 2003

[7] Chiou-Yng Lee, "K-bit-Parallel Systolic Multiplier and Squarer over GF", pp.1-8, *WSEAS Transactions on Computers*, Issue 1, Volume 4, January 2005

[8] Dragan M. Randjelovic, "Objective functions for systolic arrays", pp.1015-1021, *WSEAS Transactions on Computers* , Issue 4, Volume 2, October 2003

[9] Dragan M. Randjelovic, "Objective Functions for Flowing Systolic Arrays", pp.973-979, *WSEAS Transactions on Computers*, Issue 1, Volume 4, January 2005

[10] Jan Glasa, "On bit-level systolic arrays for digitized contour derivatives estimation (invited paper)", pp 1114-1120, *WSEAS Transactions on Computers*, Issue 4, Volume 2, October 2003

[11] M. J. Flynn, "Some Computer Organizations and Their Effectiveness" , *IEEE Trans. Comput*., Vol. C-21, pp. 948, 1972.

[12] Glenford Majers, *"Advances in computer architecture"*, John Wiley, New York, 1978.