J. King Saud Univ., Vol. 13, *Comp. & Info. Sci*., pp. 19-47 (A.H. 1421/2001)

CN-Nets for Modeling and Analyzing Neural Networks

Samir M. Koriem
Department of Systems and Computer Engineering,
Faculty of Engineering, Al-Azhar University, Nasr City, Cairo, Egypt

(Received 24 February 1998; accepted for publication 9 February 2000)

Abstract. The concept of colored timed neural Petri nets (CTNPN or Shortly CN-net) which are isomorphic to neural architectures is proposed. The CN-net technique incorporates the basic features of the neural net and the modeling capabilities of both colored and timed Petri nets. The essential principles involved in the construction of the CN-net are discussed in detail. The computation power of the CN-net model is demonstrated through the "timed reachability graph" (TRG) that is developed from this model. The CN-net is designed to study the structure properties of the artificial neural networks (ANNs) while the TRG is used for verifying the dynamic behavior of these networks. Furthermore, the CN-net offers simple and readable model representation making it easy to design fitting VLSI circuits for complex ANNs. Practical examples are given illustrating the way in which the CN-net as a novel modeling technique can be employed to simulate the dynamic behavior and parallel activities of the ANNs.

Keywords: Modeling; Neural networks; Specification; Colored and timed petri nets; Verification.

1. Introduction

The current interest in the development of artificial neural networks (ANNs) is largely due to their brain-like organizational structure and learning ability. Although still in an evolutionary stage, these networks have been used in a wide range of real-world applications such as pattern classification, function approximation, automatic control and optimization [1-3]. These networks have multiple layers of neurons that process information (a neuron integrates the incoming weighted inputs by addition, then fires if the result exceeds a threshold) in parallel, and act asynchronously in real-time through feedforward and feedback interconnections.

In order to give an accurate description of the neural network, a model has to be formulated which provides the structure and behavior of the neuron prior to VLSI hardware realization. A strong candidate for modeling ANNs are Petri nets [4]

due to its graphical representation for describing and studying information processing systems that are characterized as being parallel. Also the behavior of the modeled system can be analyzed through the movement of tokens which are used in these nets to simulate the dynamic and parallel activities within this system. As a mathematical tool, it is possible to set state equations governing the behavior of the modeled system.

Many conventional ANN paradigms employ the McCullock-Pitts [1] model of the biological neuron to build various feedforward and feedback architectures. Recently, there has been some attempt to develop Petri net models for the biological neuron. These models have directed towards increasing the understanding of the human brain behavior [5], representing simple neural networks [6 - 8] and implementing digital logic circuits for neural networks [9]. Unfortunately, these models are not able to provide a complete picture for the structural and behavioral properties of the ANNs and do not propose adequately methodology for studying and analyzing the specification and verification aspects of the ANNs. Furthermore, we need a methodology able to accurately represent many of the capabilities of the ANNs deemed necessary to transform the design of the desired ANN to fit VLSI circuit. For these reasons, in this paper, we propose a "colored timed neural Petri net" (CTNPN or shortly CN-net) as an effective modeling framework for representing the complexity characteristics of the ANNs. The CN-net technique combines the basic aspects of the neural architecture [10] with the modeling capabilities of both colored [11 - 14] and timed [15 - 18] Petri nets.

The rest of this paper is organized as follows. The development of the CN-net model is discussed with details in Section 2. The flexibility of the CN-net model and its analysis methodology (including dynamic analysis and performance analysis) are demonstrated through a simple practical example (*XOR neural network*) in Section 3. In Section 4, we explain how the CN-net model can be used as a powerful analysis tool for studying the *feedforward neural network*. This type of networks has been widely used in the literature as a practical neural network for illustrating the dynamics of ANNs [3]. Section 5 concludes this paper.

2. Basic Features of the CN-Nets

In the ANNs [3], the *i*th neuron (NE_i) consists of a processing element (PE_i) with synaptic input connections (i.e. communication paths), a threshold logic unit (TLU_i), and a single output (RES_i) as shown in Fig. 1. In this figure, x_j (j = 1, 2, ..., n) represents the *input data* that is proceeded to the NE_i and w_j (j = 1, 2, ..., n) represents the *weighted data* that is associated with the input data x_j. The weighted data w_j may be positive or negative value according to the excitation or the inhibition operation that should be performed on the NE_i . The PE_i performs the scalar product of an n-vector

input data $X = x_1, x_2, ..., x_n$ with an n-vector weighted data $W = w_1, w_2, ..., w_n$. The results are then passed through the TLU_i to perform a thresholding of value θ_i. The input-output relationship of the NE_i can then be expressed as follows: $RES_i = F(W^T X - \theta_i) = F(\sum_i - \theta_i)$, where T denotes the transpose, and $\sum_i = W^T X$.

In order to model the dynamic behavior of each neuron in the ANNs, we propose a *colored timed neural Petri net* (CN-net) as a novel modeling technique. In this section, we begin with the mathematical description of the CN-net. Then, we explain in detail the firing rules of our proposed technique.

2.1 Mathematical structure

$$\text{CN-net} = (PN, M, \beta, G, H, \alpha, \tau, OP, COM, SH)$$

In the following, we discuss the formal definitions of the CN-net parameters. We have developed these parameters based on the modeling capabilities of colored PN [11, 12, 14], timed PN [15 - 17] and neural nets [2, 10].

Definition 1. An *ordinary Petri net* (PN) without any specific initial marking is a 3-tuple $PN = (P, T, A)$ where

$P = \{p_1, p_2, ..., p_n\}$ is a finite set of places;
$T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions;
$A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
$P \cup T \neq \varnothing$ and $P \cap T = \varnothing$.

Definition 2. Let $\beta = \{c \mid c \text{ is a color type}\}$ be a *finite set of colors*. The set β is divided into two sets of colors $\beta(p)$ and $\beta(t)$ such that $\beta = \beta(p) \cap \beta(t)$. The set $\beta(p)$ describes the colors associated with each place $p \in P$ (token colors) of the underlying net. The set $\beta(t)$ describes the colors associated with each transition $t \in T$ (occurrence colors) of the same net.

- The set $\beta(p_i) \in \beta$ is used to attach to each place $p \in P$ a set of possible token colors and their priorities.

$$\beta(p_i) = \{<cp_{i1}, pr_{i1}>, <cp_{i2}, pr_{i2}>, ..., <cp_{iui}, pr_{iui}>\};$$
$$u_i = |\beta(p_i)|, \quad i = 1, 2, ..., n$$

  where u_i represents the number of colored tokens in the place $p_i \in P$. The notation $<cp_{iui}, pr_{iui}>$ denotes that the colored token $cp_{iui} \in \beta(p_i)$ of the priority $pr_{iui} \in \beta(p_i)$ is able to enter (or leave) the place $p_i \in P$.

- The set $\beta(t_j) \in \beta$ is used to attach to each transition $t_j \in T$ a set of possible occurrence colors and their priorities.

$$\beta(t_j) = \{< ct_{j1},\ pr_{j1} >,\ < ct_{j2},\ pr_{j2} >,\ \ldots ,\ < ct_{jvj},\ pr_{jvj} >\};$$
$$v_j = |\beta(t_j)|,\quad j = 1, 2, \ldots, m$$

  where v_j represents the number of colors associated with a transition $t_j \in T$. The notation $<ct_{jvj}, pr_{jvj}>$ describes the occurrence color $ct_{jvj} \in \beta(t_j)$ of priority $pr_{jvj} \in \beta(t_j)$ at the transition $t_j \in T$. It is interesting to note that the priority associated with each color is used to organize the sequence of events that can occur at the transition $t \in T$.

To clarify the relation between the color and its priority, in our CN-net models, the colors red, green, blue, black, yellow, and white are denoted by the letters c_6, c_5, c_4, c_3, c_2 and c_1, respectively. These colors have the priorities (in descending order) 1, 2, 3, 4, 5 and 6, respectively. For example, the green color c_5 and its priority 2 can be illustrated by the notation $<c_5, 2>$.

Definition 3. Let G(a): $A(\beta) \rightarrow N^+$ be a *colored multiplicity function* which describes the type and number of colors associated with each arc $a \in A$, where N^+ being the set of integers. G(a) can be defined on the set of arcs A as follows:

- $G_I(p, t)$: $\beta(p) \times \beta(t) \rightarrow N^+$; $\quad G_I(p, t) \in G(a) \quad \forall\ a \in A$
  The *input multiplicity function* $G_I(p, t)$ is used to label the arc from $p \in P$ to $t \in T$.

- $G_O(t, p)$: $\beta(t) \times \beta(p) \rightarrow N^+$; $\quad G_O(t, p) \in G(a) \quad \forall\ a \in A$
  The *output multiplicity function* $G_O(t, p)$ is used to label the arc from $t \in T$ to $p \in P$.

In general, the function G(a) is capable of deciding the colored tokens that should be removed from the input place $p_{in} \in P$ of the transition t_j and the colored tokens that should be deposited in the output place $p_{out} \in P$ of the transition t_j when it fires.

- When the multiple arcs have different types of colors, the G(a) function takes the following formulas:
  $G_{DI}(p, t) = G_D(<cp_1>, <cp_2>, \ldots, <cp_n>)$; a color $cp_n \in \beta(p)$
  $G_{DO}(t, p) = G_D(<ct_1>, <ct_2>, \ldots, <ct_m>)$; a color $ct_m \in \beta(t)$
  (e.g. $G_D(c_1, c_3, c_5)$)

- When the multiple arcs have *n* similar types of colors, the G(a) function takes the following formulas:
  $G_{SI}(p, t) = G_S(n <cp>)$; a color $<cp> \in \beta(p)$

G_{SO} (t, p) = G_S(n <ct>); a color <ct> ∈ β(t)
(e.g. G_S(3 c_2))

- The following G(a) formulas permit the use of one arc from p ∈ P to t ∈ T or from t ∈ T to p ∈ P with a specific color:
  G_{1I} (p, t) = G_1(<cp>); a color <cp> ∈ β(p)
  G_{1O} (t, p) = G_1(<ct>); a color <ct> ∈ β(t)
  (e.g. G_1(c_6))

- The following G(a) formulas allow the single arc to carry any type of color v ∈ β:
  G_{vI} (p, t) = G_v(v); a color v ∈ β(p)
  G_{vO} (t, p) = G_v(v); a color v ∈ β(t)
  (e.g. G_v(v); where v any type of color)

Definition 4. A *marking* of the CN-net model is a function M defined on P such that for p ∈ P, M(p): β(p)→ N^+, where N^+ being the set of integers. M(p) can also be represented by an (n×1) vector $[M(p_1), M(p_2), ..., M(p_n)]^T$. The marking $M(p_i)$ of a place p_i ∈ P is generally represented by formal sum of colors, i.e., $M(p_i) = \Sigma^{ui}_{k=1}\ n_{ik}\ <cp_{ik}, pr_{ik}>$, where n_{ik} is the number of tokens of color $<cp_{ik}, pr_{ik}>$ in a place p_i ∈ P. In other words, the marking $M(p_i)$ gives the number of tokens of each color in the place p_i ∈ P. For example, $M(p_1)$ = (< c_6, 1>, < c_3, 4>) denotes the place p_1 contains red and black colored tokens.

Definition 5. Let A^* = { a^* ∈ A^* | a^* is an inhibitor arc} be a set of inhibitor arcs such that A^* ⊆ A. Let P_h = { p_h | (p_h, t) ∈ A^* } be a set of the inhibitor places, where A^* ⊆ (P_h × T), p_h ∈ P_h, and P_h ⊆ P. Let H(p_h <c ∈ β>, t): A^* → β(p_h) × β(t) be an *inhibitor function*. It describes each inhibitor arc H(p_h <specific color type>, t) in the set A^*. When the inhibitor arc H(p_h <c ∈ β>, t) is labeled with a specific colored type c ∈ β and there is a token with the same colored type marking the corresponding inhibiting input place p_h ∈ P_h, then the transition t ∈ T does not fire. It is interesting to note that there is no movement of tokens along the inhibitor arc H(p_h <c ∈ β>, t) when the transition t ∈ T fires.

Definition 6. Let α(p): p→ <ℑ, β(p)> be a *description function* which describes the colored tokens β(p) ∈ β that enter (or leave) the place p ∈ P. In the CN-net model, the tokens are defined as tuples of attributes and colors separated by commas and enclosed in angular brackets. The values of these attributes are modified by transitions, enabling them to carry data. The attribute ℑ is used to describe the important processes of the modeled neuron as follows:

- α(p): p→ <x_i, β(p)>

The attribute token x_i of the color <cp, pr> $\in \beta(p)$ is used to carry the input data x_i to the *j*th neuron (*j* = 1, 2, ..., n) over the communication path *i* (see Fig. 1).

- $\alpha(p)$: $p \rightarrow <w_i, \beta(p)>$
  The attribute token w_i of the color <cp, pr> $\in \beta(p)$ is used to carry the weighted data w_i to the *j*th neuron (*j* = 1, 2, ..., n) over the communication path *i*. The weighted data w_i is related to the input data x_i. Both the tokens x_i and w_i have the same color.
- $\alpha(p)$: $p \rightarrow <\theta_j, \beta(p)>$
  The attribute token θ_j of the color <cp, pr> $\in \beta(p)$ is used to carry the threshold value θ_j of the *j*th neuron.
- $\alpha(p)$: $p \rightarrow <RES_j, \beta(p)>$
  The attribute token RES_j of the color <cp, pr> $\in \beta(p)$ is used to carry the final output result of the *j*th neuron.
- $\alpha(p)$: $p \rightarrow <DATA_{ij}, \beta(p)>$
  The attribute token $DATA_{ij}$ of the color <cp, pr> $\in \beta(p)$ is used to carry information about the input data x_k and its corresponding weighted data w_k that are required to pass to a neuron *j* from a neuron *i*. Based on this information, a neuron *i* organizes its communication behavior with a neuron *j*.
- $\alpha(p)$: $p \rightarrow <s_j, \beta(p)>$
  The attribute token s_j of the color <cp, pr> $\in \beta(p)$ represents the status of the *j*th neuron. If the place $p_i \in P$ contains a token, then a neuron *j* is busy and cannot receive data from its neighboring neurons.

Definition 7. Let τ: $T \times \beta(t) \rightarrow R^+$ be a *firing time function*. It assigns the time of firing $\tau(t)$ to each occurrence color at a transition $t \in T$ in the net, where R^+ denotes the set of non-negative deterministic numbers.

Definition 8. Let r(t, $\beta(t)$) be a *remaining firing time function*. It assigns the remaining time of firing r(t) to each independent firing (if any) of each occurrence color $\beta(t) \in \beta$ at a transition $t \in T$.

Definition 9. Let COM(t): $T \rightarrow <COM_{ij}, \tau, \beta(t)>$ be a *communication function*. It defines the necessary parameters for firing the "communication transition" $T_{com} \in T$ when the color <ct, pr> $\in \beta(t)$ occurs. The transitions t_{send}, $t_{end\text{-}rec} \in T_{com}$ are used to model the communication behavior (sending or receiving data, respectively) between a neuron *i* and a neuron *j*. Let τ be the communication time required for transmitting (or receiving) data from a neuron *i* to a neuron *j*.

Definition 10. Let $OP_{\times}(t)$: $T \rightarrow <(w_i \times x_i), \tau, \beta(t)>$ be a *computation function*. It defines the necessary parameters for firing the "multiplication operation transition" $T_{mult} \in T$ due to the occurrence color <cp, pr> $\in \beta(t)$. The $T_{mult} \in T$ models the neuron when

executing the multiplication operation ($w_i \times x_i$). Let τ be the processing time of this multiplication operation.

Definition 11. Let $OP_+(t)$: $T \rightarrow <\Sigma_j, \tau, \beta(t)>$ be a *computation function*. It defines the necessary parameters for firing the "addition operation transition" $T_{add} \in T$ due to the occurrence color $<cp, pr> \in \beta(t)$. The $T_{add} \in T$ models a neuron *j* when executing its addition operation $\Sigma_j = w_1x_1 + w_2x_2 + ...+ w_nx_n$. Let τ be the processing time of this addition operation. Note that when the summation operation incorporates data with different types of colors, the computation result Σ_j takes the color of the highest priority.

Definition 12. Let $SH(t)$: $T \rightarrow <(\Sigma_j \otimes \theta_j), \tau, \beta(t)>$ be a *computation function*. It defines the necessary parameters for firing the "threshold transition" $T_{shold} \in T$ when the color $<ct, pr> \in \beta(t)$ occurs. The $T_{shold} \in T$ models a neuron *j* when executing its comparison operation ($\otimes$) between the values of θ_j and Σ_j. Let τ be the processing time of this comparison operation. The output result of this comparison operation $<RES_j, \beta(p)>$ is calculated as follows:

- If $\Sigma_j < \theta_j$, then the output place $\alpha(p_i)$ of the $T_{shold} \in T$ will contain the token <zero, $\beta(p)$>.
- If $\Sigma_j > \theta_j$, then the output place $\alpha(p_i)$ of the $T_{shold} \in T$ will contain the token <one, $\beta(p)$>.

2.2 Firing rules

The modeling power of CN-net lies in the color-priority attributes associated with each place $\beta(p) \in \beta$ and transition $\beta(t) \in \beta$ in the net. In the CN-net, there exists a functional dependency between the color-priority of the enabled transitions and the color-priority of the tokens marking the input places of these transitions. Following is an explanation of how the CN-net uses this functional dependency to organize its transition firing rules.

Consider a colored token $<cp_{ix}, pr_{ix}> \in \beta(p_i)$ is marking the input place $p_i \in P$ of the transition $t_j \in T$. A transition t_j can be enabled with respect to a color $<ct_{jx}, pr_{jx}> \in \beta(t_j)$. A place $p_k \in P$ is the output place of a transition t_j. The inhibitor place $p_h \in P_h$ is connected to the transition t_j. The firing of a transition t_j is carried out through the following two steps.

Step 1. A transition $t_j \in T$ is enabled with respect to a color $<ct_{jx}, pr_{jx}> \in \beta(t_j)$ in a marking M iff

- $M(p_i <cp_{ix}, pr_{ix}>) \geq G_{11}(p_i <cp_{ix}, pr_{ix}>, t_j <ct_{jx}, pr_{jx}>)$; $\forall p_i \in P$, and

- $H(p_h, t_j) = H(p_h <\varnothing>,\ t_j <ct_{jx}, pr_{jx}>)$; $p_h \in P_h$ ($<\varnothing>$: empty)

The notation $p_h <\varnothing>$ denotes that there is no token in the inhibitor place $p_h \in P_h$. The notation $t_j <ct_{jx}, pr_{jx}>$ denotes that the transition t_j is enabled with respect to a color $<ct_{jx}, pr_{jx}>$.

It is interesting to note that if $H(p_h, t_j) = H(p_h <cp_{hx}, pr_{hx}>,\ t_j <ct_{jx}, pr_{jx}>)$, then t_j is disabled. The notation $p_h <cp_{hx}, pr_{hx}>$ denotes that the inhibitor place $p_h \in P_h$ contains the colored token $<cp_{hx}, pr_{hx}>$ which is the same as the color associated with t_j and the colored token marked the place $p_i \in P$.

Step 2. When a transition $t_j \in T$ (enabled in a marking M) fires with respect to a color $<ct_{jx}, pr_{jx}> \in \beta(t_j)$, a new marking $M^{\blacklozenge}$ is reached according to the following:

$$M^{\blacklozenge}(p_k <cp_{ky}, pr_{ky}>) = M(p_i <cp_{ix}, pr_{ix}>) + G_{1I}(p_i <cp_{ix}, pr_{ix}>, t_j <ct_{jx}, pr_{jx}>) - G_{1O}(t_j <ct_{jx}, pr_{jx}>, p_k <cp_{ky}, pr_{ky}>);$$
$$\forall p_i \in P,\ <cp_{ky}, pr_{ky}> \in \beta(p_k),\ <cp_{ix}, pr_{ix}> \in \beta(p_i),\ <ct_{jx}, pr_{jx}> \in \beta(t_j)$$

The first step of the firing rule of t_j is needed to be generalized to incorporate the following effects: (i) increasing the number of tokens in the place p_i; (ii) inscriptions on the arcs; and (iii) the priority level assigned to each colored token. For this purpose, we illustrate the following cases for the marking M of the first step of t_j firing rule.

Case A. Consider the place $p_i \in P$ contains two colored tokens with *different priorities*: $<cp_{ix}, pr_{ix}>,\ <cp_{iy}, pr_{iy}> \in \beta(p_i)$, and $pr_{ix} > pr_{iy}$. It is interesting to note that when the number of tokens in the place $p_i \in P$ is more than one, a transition $t_j \in T$ can be enabled iff the colors of these tokens are members of the color set associated with this transition.

A-1. To proceed these tokens one by one from a place $p_i \in P$, the CN-net uses the arc function $G_{VI}(p_i, t_j) = G_{VI}(v)$. In this case, all the tokens in the place p_i (simultaneously) enable the transition t_j and the one of the highest priority color fires this transition.

- $M(p_i <cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>) \geq G_{VI}(p_i <v>,\ t_j (<cp_{ix}, pr_{ix}>,\ <cp_{iy}, pr_{iy}>))$; $(<cp_{ix}, pr_{ix}>,\ <cp_{iy}, pr_{iy}> \in v)$ and
- $H(p_h, t_j) = H(p_h <\varnothing>,\ t_j (<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>))$; $p_h \in P_h$

  If $H(p_h, t_j) = H(p_h <cp_{ix}, pr_{ix}>,\ t_j (<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>))$,
  then t_j is only disabled for the color $<cp_{ix}, pr_{ix}>$.

Once t_j fires, the highest priority colored token $<cp_{ix}, pr_{ix}>$ is removed from a place p_i to a transition t_j through the arc function $G_{VI}(v)$. In this case, the transition t_j

uses the remaining time function r to keep track of its firing sequences for the other colored tokens.

A-2. If the arc (p_i, t_j) is labeled with the function $G_{DI}(p_i, t_j)$, then, once the number of colored tokens in a place $p_i \in P$ satisfy this function, the transition t_j fires.

- $M(p_i <cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>) = G_{DI}(p_i (<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>), t_j (<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>))$; and
- $H(p_h, t_j) = H(p_h <\varnothing>, t_j (<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>))$; $p_h \in P_h$
  If $H(p_h, t_j) = H(p_h <cp_{ix}, pr_{ix}>, t_j (<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>))$,
  then t_j is disabled.

Once a transition t_j fires, the appropriate colored tokens specified by the function $G_{DI}(<cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>)$ are removed from a place $p_i \in P$.

Case B. Consider a place $p_i \in P$ contains two colored tokens with the *same priorities*: $<cp_{ix}, pr_{ix}>, <cp_{ix}, pr_{ix}> \in \beta(p_i)$.

B-1. To proceed these tokens one by one from a place $p_i \in P$, the CN-net uses the arc function $G_{1I}(p_i, t_j)$. In this case, all the tokens in the place $p_i \in P$ (simultaneously) enable the transition t_j and any of them fires the transition t_j.

- $M(p_i <cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>) \geq G_{1I}(p_i (<cp_{ix}, pr_{ix}>, t_j <cp_{ix}, pr_{ix}>)$; and
- $H(p_h, t_j) = H(p_h <\varnothing>, t_j (<cp_{ix}, pr_{ix}>)$; $p_h \in P_h$
  If $H(p_h, t_j) = H(p_h <cp_{ix}, pr_{ix}>, t_j (<cp_{ix}, pr_{ix}>)$, then t_j cannot fire.

Once a transition t_j fires, the appropriate token $<cp_{ix}, pr_{ix}>$ is removed from the place $p_i \in P$ to the transition t_j through the arc function $G_{1I}(<cp_{ix}>)$. To keep track of the firing sequence of the other colored tokens, the transition t_j uses the remaining time function r.

B-2. To proceed all the similar tokens in the place $p_i \in P$ to the transition t_j, the CN-net inscripts the arc (p_i, t_j) with the function $G_{SI}(p_i, t_j)$.

- $M(p_i <cp_{ix}, pr_{ix}>, <cp_{iy}, pr_{iy}>) = G_{SI}(p_i (2 <cp_{ix}, pr_{ix}>, t_j <cp_{ix}, pr_{ix}>)$; and
- $H(p_h, t_j) = H(p_h <\varnothing>, t_j (<cp_{ix}, pr_{ix}>)$; $p_h \in P_h$
  If $H(p_h, t_j) = H(p_h <cp_{ix}, pr_{ix}>, t_j (<cp_{ix}, pr_{ix}>))$, then t_j cannot fire.

For the purpose of modeling the neural networks, two types of transitions will be defined in our CN-net: *operation transitions* and *communication transitions*. The former ones model the computation behavior of each neuron ($OP_{\times}(t)$, $OP_{+}(t)$, $SH(t)$; $t \in T$). The later ones model the communication behavior between a neuron i and a neuron j

(COM_{ij}). The firing time of the operation transition or the communication transition is a "deterministic time" τ (i.e. the tokens are removed from the input place at the beginning of firing period, and they are deposited to the output places at the end of this period). In some cases, we need to model activities with no time duration (i.e. τ is equal to zero). We use this concept to model the logical behavior among the neurons. When τ takes deterministic value, the transition is drawn as a thick bar. When τ takes zero value, the transition is drawn as a thin bar. A transition with no time duration can fire as soon as it is enabled and cannot remain enabled for any duration of time. Thus, if the marking M comprises both types of transitions, only the transitions with no duration times can fire.

3. Analysis of the CN-Nets

In this section, we first explain how our proposed CN-net technique can be used for modeling and analyzing the structure and behavior properties of the neural networks. Finally, the performance analysis of the CN-net model is discussed.

3.1 Dynamic behavior

For better understanding the dynamics of the CN-net model, we start our explanation by modeling the functioning of the basic elements of the single neuron shown in Fig.1. The CN-net model for a single neuron is depicted in Fig. 2. The meanings associated with the places and transitions of this model are summarized in Table 1. To build a realistic model, we have assumed that the modeled neuron *i* (NE_i) communicates with its neighboring neurons NE_1, NE_2, and NE_3. The NE_i is within the layer-2 and the other neurons are in the layer-1 as shown in Fig. 2. To allow the NE_i to excite or inhibit its neighboring neurons, we use the inhibitor arc $H(\theta_i, T_{end\text{-}reci})$. When there is a token in the place θ_i, the inhibitor arc prevents the NE_i from receiving data from the other neurons until this neuron completes its existing calculation. When the place θ_i is empty, the NE_i performs its communication behavior with the neighboring neurons.

Now, we need a methodology for verifying whether the developed CN-net model is accurately representing both structure and dynamic behavior of the modeled neural network. For this purpose, we should develop the "timed reachability graph" TRG for the desired CN-net neural model.

Definition 13. The "timed reachability graph" TRG for the CN-net model is the set of all states S of the CN-net model which can be reached from the initial state $S_1 \in S$ by firing a finite number of transitions.

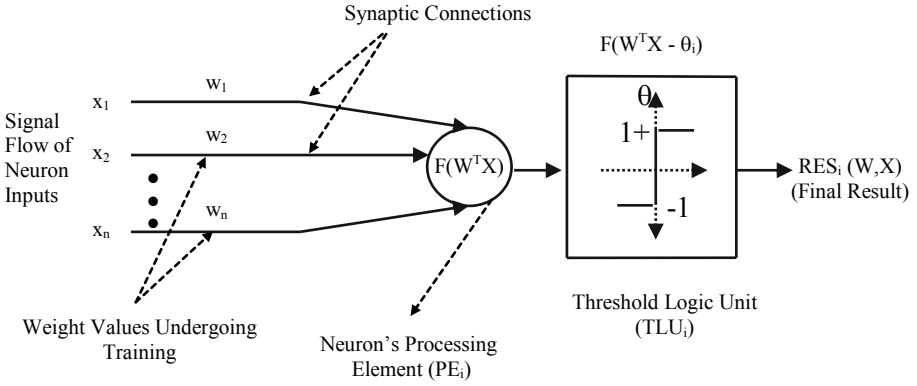

Fig. 1. A single artificial neuron *i* (NE_i).

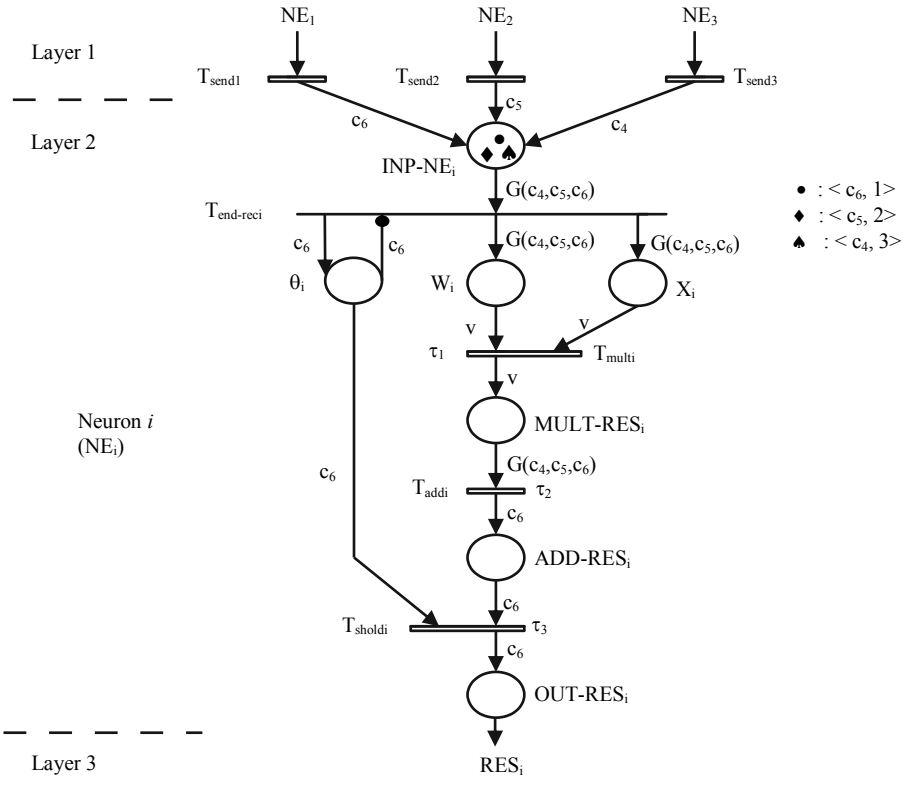

Fig. 2. A CN-Net single-neuron model.

Table 1. Annotation of the places and transitions of the CN-Net single-neuron model shown in Fig. 2

T_{sendj}	= t_j (j=1,2,3): A NE_j in the layer-1 transmits its output data ($DATA_{ji}$) to a NE_i in the layer-2.
INP-NE_i	= p_1: A NE_i receives data from its neighboring neurons: NE_1, NE_2, and NE_3.
$T_{end\text{-}reci}$	= t_4: A NE_i has completely received its required data from the neighboring neurons.
X_i	= p_2: A NE_i recognizes its *input data vector* X =x_1, x_2, x_3 from the data coming from the neighboring neurons.
W_i	= p_3: A NE_i recognizes its *weighted data vector* W = w_1, w_2, w_3 from the data coming from the neighboring neurons, where w_j is the corresponding weight to the *j*th input process x_j.
θ_i	= p_4: A threshold value of the NE_i.
T_{multi}	= t_5: A NE_i executing its multiplication operation x_j w_j.
MULT-RES_i	= p_5: The results of the multiplication operations.
T_{addi}	= t_6 : A NE_i executing its addition operation: $x_1 w_1 + x_2 w_2 + x_3 w_3 = \sum_i$.
ADD-RES_i	= p_6: The result of the addition operation.
T_{sholdi}	= t_7 : A NE_i executing its comparison operation between $\sum_i$ and θ_i.
OUT-RES_i	= p_7: An output result of the NE_i.

Definition 14. A state $S_i \in TRG(S)$ is defined by three descriptive attributes MRK_i, SET_i and INH_i. Based on the function M_i, the attribute MRK_i illustrates the distribution of tokens in the various places of the current state S_i. Based on the function H_i, the attribute INH_i illustrates the distribution of tokens in the inhibitor places and the inscriptions on the inhibitor arcs that are shown in the current state S_i. Attribute SET_i indicates the status of the enabled transitions in the current state S_i. This attribute has three parameters T_{REM}, T_{NEW} and T_{FIR}. The parameter T_{REM} indicates the transitions that have remaining firing times. The parameter T_{NEW} indicates the new enabled transitions in the current state S_i. The parameter T_{FIR} tests the minimum time that is associated with all the enabled transitions shown in both T_{REM} and T_{NEW} to select the actual firing transition(s) in the current state S_i.

In the initial state of the CN-net neural model shown in Fig. 2, a place INP-NE_i contains three colored tokens: <$DATA_{1i}$, (c_6,1)>, <$DATA_{2i}$, (c_5,2)>, and <$DATA_{3i}$, (c_4,3)>. These tokens represent the different data arriving to the NE_i from its neighboring neurons NE_1, NE_2, and NE_3. Given an initial state to our developed model, the TRG is obtained by firing (executing) consequent enabling transitions according to the proposed execution rules of the CN-net, as shown in Fig. 3. Execution is continued until there is no more enabling transitions. The TRG shown in Fig. 3 is formally described as follows. To simplify our explanation of this TRG, we give other names for the transitions and places of the model of Fig. 2 as shown in Table 1.

S_1 : $MRK_1 : \alpha(p_1) = <DATA_{1i}, (c_6,1)>, <DATA_{2i}, (c_5,2)>, <DATA_{3i}, (c_4,3)>$
$SET_1 : T_{FIR} : t_4 = <(c_6,1), (c_5,2), (c_4,3)>$

$S_2 : MRK_2 : \alpha(p_2) = <x_1, (c_6,1)>, <x_2, (c_5,2)>, <x_3, (c_4,3)>,$
$\alpha(p_3) = <w_1, (c_6,1)>, <w_2, (c_5,2)>, <w_3, (c_4,3)>, \ \alpha(p_4) = <\theta_i, (c_6,1)>$
$SET_2 : T_{FIR} : t_5 = <(w_i \times x_i), \tau_1, ((c_6,1), (c_5,2), (c_4,3))>$
$INH_2 : H(p_4 <c_6,1>, t_4)$

$S_3 : MRK_3 : \alpha(p_2) = <x_2, (c_5,2)>, <x_3, (c_4,3)>, \quad \alpha(p_4) = <\theta_i, (c_6,1)>$
$\alpha(p_3) = <w_2, (c_5,2)>, <w_3, (c_4,3)>, \quad \alpha(p_5) = <w_1 \ x_1, (c_6,1)>$
$SET_3 : T_{FIR} : t_5 = <(w_i \times x_i), \tau_1, ((c_6,1), (c_5,2), (c_4,3))>$
$INH_3 : H(p_4 <c_6,1>, t_4)$

$S_4 : MRK_4 : \alpha(p_2) = <x_3, (c_4,3)>, \quad \alpha(p_3) = <w_3, (c_4,3)>, \quad \alpha(p_4) = <\theta_i, (c_6,1)>$
$\alpha(p_5) = <w_1 \ x_1, (c_6,1)>, <w_2 \ x_2, (c_5,2)>$
$SET_4 : T_{FIR} : t_5 = <(w_i \times x_i), \tau_1, ((c_6,1), (c_5,2), (c_4,3))>$
$INH_4 : H(p_4 <c_6,1>, t_4)$

$S_5 : MRK_5 : \alpha(p_4) = <\theta_i, (c_6,1)>,$
$\alpha(p_5) = <w_1 \ x_1, (c_6,1)>, <w_2 \ x_2, (c_5,2)>, <w_3 \ x_3, (c_4,3)>$
$SET_5 : T_{FIR} : t_6 = < \Sigma_i, \tau_2, ((c_6,1), (c_5,2), (c_4,3))>$
$INH_5 : H(p_4 <c_6,1>, t_4)$

$S_6 : MRK_6 : \alpha(p_4) = <\theta_i, (c_6,1)>, \quad \alpha(p_6) = <\Sigma_i, (c_6,1)>$
$SET_6 : T_{FIR} : t_7 = <(\Sigma_i \otimes \theta_i), \tau_3, (c_6,1)>$
$INH_6 : H(p_4 <c_6,1>, t_4)$

$S_7 : MRK_7 : \alpha(p_7) = <RES_i, (c_6,1)>$

From the TRG shown in Fig. 3 and its formal description illustrated above, various properties such as liveness, deadlock-free operations, execution transition scenarios, time delays of transitions, and the movement of data at the various elements of each neuron in the network can be studied. These properties help us to verify the correctness of the dynamics of our CN-net model shown in Fig. 2. Furthermore, the formaldescription of the TRG affords an easy and simple methodology for understanding (also verifying) the *learning algorithm* that can be applied on the CN-net neural model. The CN-net modeling technique is suitable for the supervised learning algorithm. It can be extended to unsupervised learning or reinforcement learning algorithm [2] Based on this evaluation, we will use the CN-net single-neuron model shown in Fig. 2 as a basic module for constructing the whole model of the required neural network, as we will explain in Section 4.

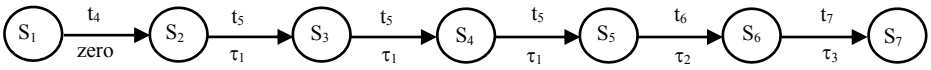

Fig. 3. The TRG of the CN-Net single-neuron model shown in Fig. 2.

3.2 Performance

The rapidly changing technology of very large integrated circuits (VLSI) has provided us with a means in which it is now possible to fabricate tens of millions of transistors interconnected on a single silicon wafer. Such capability has opened new directions for the implementation of neural networks into silicon structures. The complexity of such computational silicon structure derives from the multiple ways in which a large collection of its components is made to interact with each other.

With reference to a simple VLSI neural circuit one may wish to evaluate the maximum amount of time which must elapse from a given *input data pattern* application to the successive one, knowing the propagation delay of each element in this circuit. From more complex VLSI implementation of neural networks, with many cascaded stages for creating a multilayer neural network, this task is complicated by the possibility to vary the *input data pattern* X (or the *weighted data pattern* W), while the previous pattern still propagates in the network.

From the continuing VLSI revolution, we found that a vital need to develop specification and verification technique such as the CN-nets to ameliorate the difficulties associated with validating VLSI neural circuit designs. For this purpose, in the previous section, we have studied through the TRG how our proposed CN-net technique is able to provide a formal specification concept for a neural network. The TRG permits the automatic translation of behavioral specification neural model into a state transition graph made up of a set of states, a set of actions (firing the transitions), and a succession relation associating states to actions. Furthermore, from the TRG, we are able to check the validity of the modeled neural network. In this section, we complete the framework of CN-net methodology by evaluating the performance of a VLSI neural network.

Given time delays of elementary VLSI circuit of each neuron structure in the ANNs, the CN-net model of the desired neural network can be executed to obtain a TRG whose arcs are labeled with firing transitions numbers and their time delays. As an example, see Fig. 3. For evaluating the performance of VLSI circuit that implements the desired neural network, we calculate the *Maximum Processing Rate* (MPR) for the CN-net model of this network. To calculate this performance measure, we apply the results

obtained by Ramchandani [4] to the TRG of the CN-net model as follows. After developing the TRG, we search all the paths of the TRG (from the initial state to the final state) to find the path with the maximum time delays. Then, the MPR can be obtained from the TRG by simply adding up the firing delays of the transitions over this path. Thus, the MPR can be defined as the determination of the maximum speed at which signals can flow in the various paths assuming that the modeled circuit is operating correctly.

Let $\delta_k = t_1\ t_2\ ...\ t_i\ ...\ t_f$ be a firing sequence of transitions of a path *k* in a TRG, then the total delay time τ_k of δ_k is equal to $\tau_1 + \tau_2 + ... + \tau_i + ... + \tau_f$, where τ_i is a given delay time of a transition t_i in a CN-net model. Based on this concept, we can formally describe the MPR as follows:

$$MPR = \max\{\tau_k : k=1, 2, ..., q\}$$

where q is the number of signal paths in the TRG and τ_k is the sum of the firing delays of the transitions in the path k.

Example: Consider the TRG shown in Fig. 4 is developed from a CN-net neural model. Firing transition sequences or execution scenarios of successful execution are given as: (a) $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_8$; and (b) $t_1 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7 \rightarrow t_8$. The execution time of the first scenario $\tau_a = \tau_1 + \tau_2 + \tau_3 + \tau_4 + \tau_8 = 2 + \text{zero} + 3 + 1 + 1 = 7$ time units. The execution time of the second scenario $\tau_b = \tau_1 + \tau_5 + \tau_6 + \tau_7 + \tau_8 = 2 + \text{zero} + 3 + 3 + 1 = 9$ time units. By enumerating all paths in the net, the MPR is equal to 9 time units.

4. Application of a CN-Net to Feedforward Neural Networks

In this section, we explain how the CN-net model can be used as a powerful analysis tool for studying the various *feedforward neural network* configurations. This type of networks has been widely used in the literature as a practical neural network for illustrating the dynamics of ANNs [3].

4.1 XOR Feedforward neural network

Feedforward networks constitute an important variety of neural networks [2, 3]. For such networks, it is possible to index the neurons in such a way that the output of a neuron *j* is connected to an input of a neuron *i* when $j < i$. The final output of this network depends only on synaptic weights and the pattern presented at the input of the neural network because there is no feedback. This type of network can be connected in cascade to create a multilayer network. Thus, we can call such network a *multilayer feedforward neural network*. The most famous practical example of such network is the

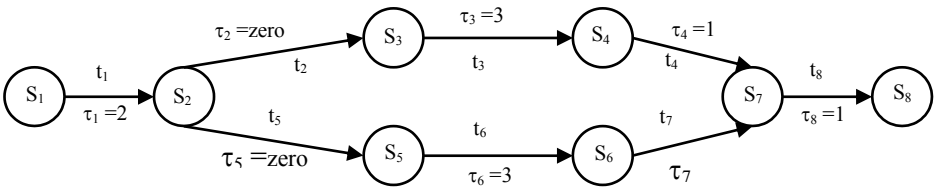

Fig. 4. An example of TRG.

exclusive or (XOR) network [2]. This network responds {1} if it receives {0, 1} or {1,0} and responds {0} otherwise. Fig. 5 shows a network capable of this pattern.

As shown in Fig. 5, the input layer contains NE_1 and NE_2, the hidden layer contains NE_3 and NE_4, and the output layer contains NE_5. In Fig. 5, NE_1 and NE_2 transmit, in parallel fashion, the input data x_1 (along with the weighted data w_1) and the input data x_2 (along with the weighted data w_4) to NE_3 and NE_4, respectively. Then, NE_1 and NE_2 transmit, in parallel fashion, the input data x_1 (along with the weighted data w_3) and the input data x_2 (along with the weighted data w_2) to NE_4 and NE_3, respectively. In other words, NE_1 and NE_2 are only used for transmitting the required data to the network (e.g. each one has a threshold of value zero). Both NE_3 and NE_4 represent the hidden neurons. The NE_5 delivers an output of this network.

The structure and behavior of the XOR neural network shown in Fig. 5 can be described using the CN-net model shown in Fig. 6. This model is also used to represent the communication and computation behavior of each neuron in the XOR network.

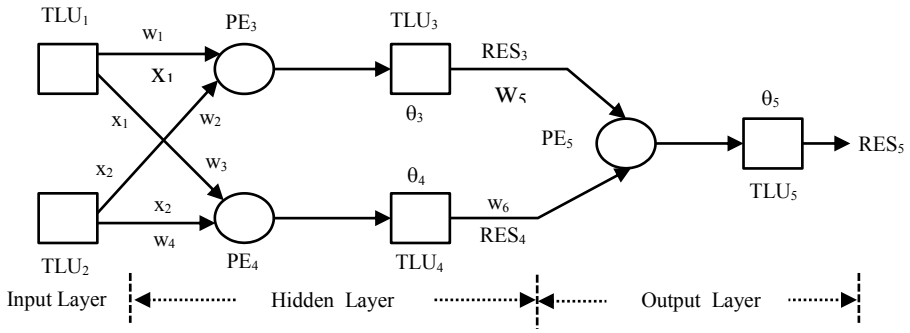

Fig. 5. A XOR feedforward neural network.

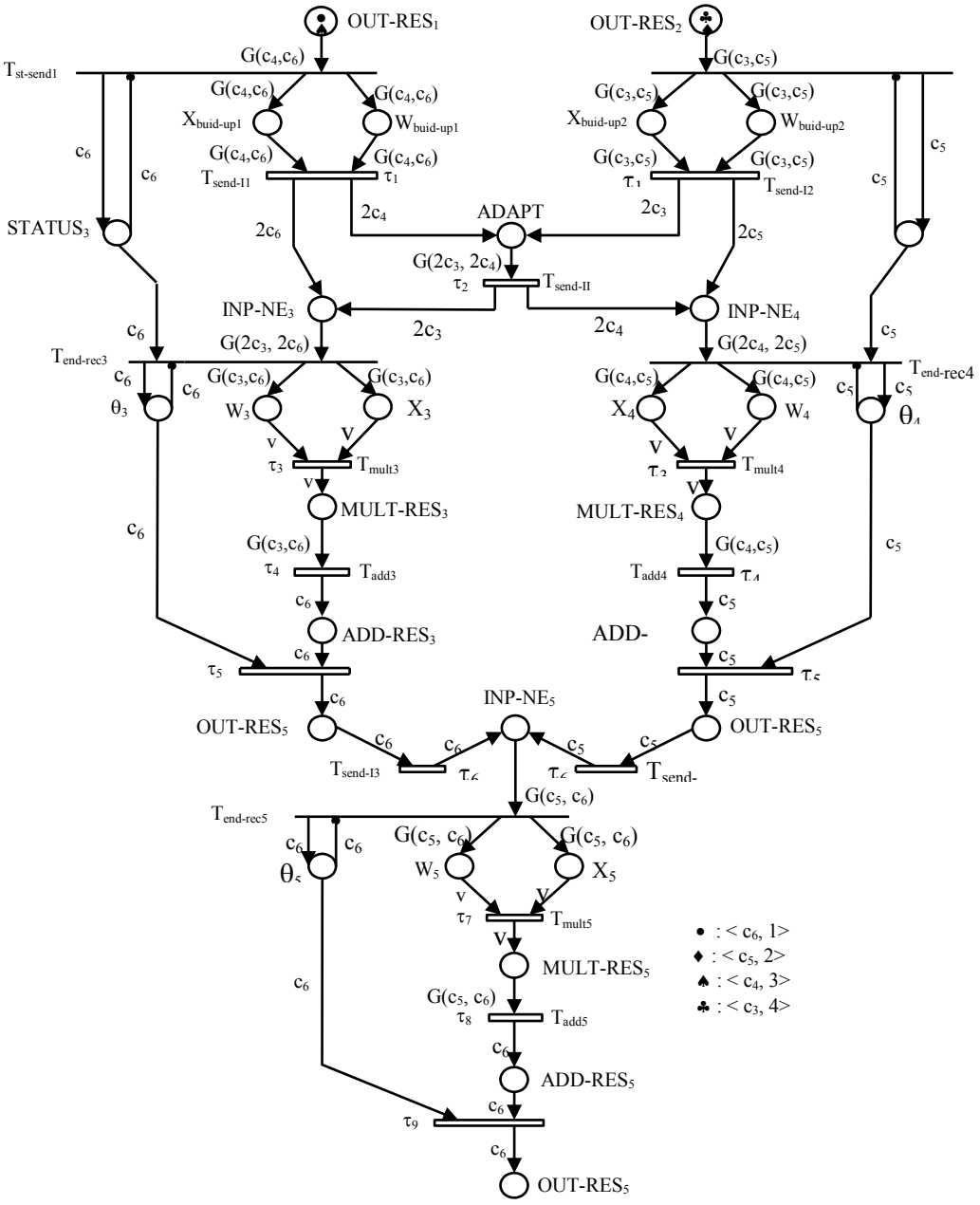

Fig. 6. A CN-Net model for the feedforward neural network.

A description of the CN-net model of Fig. 6 is illustrated in Table 2. In the initial state of this model, a place OUT-RES_1 contains the colored tokens <$DATA_{13}$, (c_6,1)>, <$DATA_{14}$, (c_4,3)> and a place OUT-RES_2 contains the colored tokens <$DATA_{23}$, (c_5,2)>, <$DATA_{24}$, (c_3,4)>. Starting from this initial state, we can generate the TRG of the CN-net model of XOR network as shown in Fig.7. The formal description of this TRG is presented in the Appendix.

Table 2. Annotation of the places and transitions of the XOR CN-Net model shown in Fig. 6

$OUT\text{-}RES_i$	(p_1, p_2) : A NE_i (i = 1, 2) contains its output results. The tokens shown in this place represent the output results obtained from a NE_i.
$T_{st\text{-}send\ i}$	(t_1, t_2): A NE_i (i = 1, 2) starts to send its output data to NE_3 and NE_4.
$X_{build\text{-}up1}$	p_3: A NE_1 allocates the *data pattern* x_1, x_1 that should be send to NE_3 and NE_4, respectively.
$X_{build\text{-}up2}$	p_5: A NE_2 allocates the data pattern x_2, x_2 that should be send to NE_3 and NE_4, respectively.
$W_{build\text{-}up1}$	p_4: A NE_1 allocates the *weight pattern* w_1, w_3 that should be send to NE_3 and NE_4, respectively. The weights w_1, and w_3 are related to the data x_1 and x_1, respectively.
$W_{build\text{-}up2}$	p_6 : A NE_2 allocates the weight pattern w_2, w_4 that should be send to NE_3 and NE_4, respectively. The weights w_2 and w_4 are related to the data x_2 and x_2, respectively.
$T_{send\text{-}I\ i}$	(t_3, t_4): Neurons i and i+1 (i = 1) transmit, in parallel fashion, the data (x_1, w_1) and (x_2, w_4) to NE_4 and NE_3, respectively.
ADAPT	p_9 : Neurons i and i+1 (i = 1) are ready to transmit the data (x_1, w_3) and (x_2, w_2) to NE_3, and NE_4, respectively.
$T_{send\text{-}II}$	t_5 : Neurons i and i+1 (i = 1) transmit, in parallel fashion, the data (x_1, w_3) and (x_2, w_2) to NE_3, and NE_4, respectively.
$STATUS_i$	(p_7, p_8): A NE_i (i = 3, 4) is busy, when the place $STATUS_i$ contains a token. This means that a NE_i is not ready to accommodate another connection with its neighboring neurons.
$T_{send\text{-}Ii}$	t_{14}, t_{15}: A NE_i (i = 3, 4) transmits its output data to the NE_5.
$INP\text{-}NE_i$	(p_{10}, p_{11}, p_{24}): A NE_i (i = 3, 4, 5) receives data from its neighboring neurons.
$T_{end\text{-}rec\ i}$	(t_6, t_7, t_{16}): A NE_i (i = 3, 4, 5) has completely received its required data from the neighboring neurons.
X_i	(p_{12}, p_{15}, p_{25}): A NE_i (i = 3, 4, 5) recognizes its *input data vector* X from the data coming from the neighboring neurons.
W_i	(p_{13}, p_{16}, p_{26}): A NE_i (i = 3, 4, 5) recognizes its *weighted data vector* W from the data coming from the neighboring neurons.
θ_i	(p_{14}, p_{17}, p_{27}): A threshold value of the NE_i (i = 3, 4, 5).
$T_{mult\ i}$	(t_8, t_9, t_{17}): A NE_i (i = 3, 4, 5) executing its multiplication operation.
$MUL\text{-}RES_i$	(p_{18}, p_{19}, p_{28}): The results of the multiplication operations of the NE_i (i = 3, 4, 5).
$T_{add\ i}$	(t_{10}, t_{11}, t_{18}): A NE_i (i = 3, 4, 5) executing its addition operation.
$ADD\text{-}RES_i$	(p_{20}, p_{21}, p_{29}): The result of the addition operation of the NE_i (i = 3, 4, 5).
$T_{shold\ i}$	(t_{12}, t_{13}, t_{19}): A NE_i (i = 3, 4, 5)) executing its comparison operation.
$OUT\text{-}RES_i$	(p_{22}, p_{23}, p_{30}): The output result of the NE_i (i = 3, 4, 5).

As we have illustrated in the Appendix, the mathematical analysis of TRG of Fig.7 provides us with the complete information regarding the movement of tokens, the existence of different types of conditions at different levels and the firing sequences of transitions. This information help us in studying the dynamics of the modeled networks. The term dynamics applied to nets describes how the net functions over time. Firing sequences or execution scenarios of transitions of Fig. 7 are given below:

- $t_1 \rightarrow t_2 \rightarrow t_3,t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7 \rightarrow t_8,t_9 \rightarrow t_8,t_9 \rightarrow t_{10},t_{11} \rightarrow t_{12},t_{13} \rightarrow t_{14},t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{17} \rightarrow t_{18} \rightarrow t_{19}$.
- $t_2 \rightarrow t_1 \rightarrow t_3,t_4 \rightarrow t_5 \rightarrow t_7 \rightarrow t_6 \rightarrow t_8,t_9 \rightarrow t_8,t_9 \rightarrow t_{10},t_{11} \rightarrow t_{12},t_{13} \rightarrow t_{14},t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{17} \rightarrow t_{18} \rightarrow t_{19}$.

Since the transitions t_1, t_2, t_6, t_7 and t_{16} represent the immediate transitions in both the above paths, the MPRs of these paths are equal. Then, the MPR of the CN-net XOR model is equal to $\tau_1 + \tau_2 + \tau_3 + \tau_4 + \tau_5 + \tau_6 + \tau_7 + \tau_8 + \tau_9$ time units.

From the above analysis of the CN-net XOR model shown in Fig. 6 and its TRG shown in Fig. 7, we observe the following interesting points.

- The model provides us with clear and understandable graphical representation for the modeled neural network. This representation has a potential for being useful in the design of fitting VLSI circuit for the desired network.

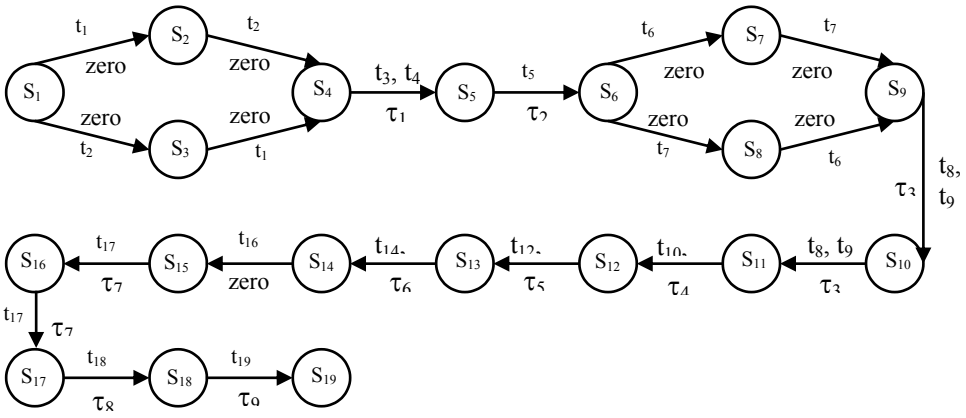

Fig. 7. The TRG of the CN-Net model of Fig. 6.

- There is a direct correlation between elements of the model and circuit realizations: places, tokens, and transitions could represent wires, signals, and actions, respectively.

- The model does not serve only as a description scheme but it is accompanied by a mathematical methodology which allows the dynamic analysis of the desired neural network. Furthermore, the formal description of the TRG illustrates the various steps of the *learning algorithm* that can be applied to the CN-net neural model. However, the CN-net model can be used as a *learning model*. This learning model will help the user to apply different types of learning algorithms [3] to the modeled neural network and study them in an easy way.

4.2 Multilayer neural network

The motivation for the introduction of our proposed CN-net modeling technique is not only for facilitating the modeling of neural networks in an elegant way but also for performing a *compact representation* for a complex neural network. In order to understand how the CN-net can be useful for providing this compact representation, we use the multilayer neural network shown in Fig. 8. This figure illustrates the structure of a six-layer feedforward neural network with eleven neurons. The circles and arcs of the network represent the computing neuron elements and their communication paths, respectively. As shown in Fig. 8, the input layer contains NE_1 and NE_2. The hidden layers compress layers 1, 2, 3, and 4. The output layer contains NE_{11}. In this network, we consider NE_1 and NE_2 are only used for transmitting the required data to the network (e.g. each one has a threshold of value zero). In practice, the neurons in one layer generate and transmit outputs to the following layer in accordance with the weighted inputs from the previous layer and the thershould values.

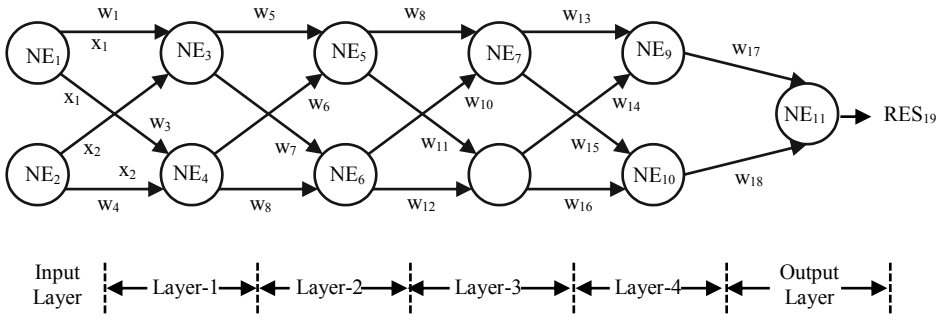

Fig. 8. Multilayer feedforward neural network.

From the above description of the network of Fig. 8, we observe that the dynamic behavior of the neurons of each layer is similar except the output layer. Since the tokens of the CN-net framework are designed to carry the input data, the weighted data, the threshold value, or the output result of each neuron, we can easily use this CN-net modeling flexibility to model the dynamic behavior between the neurons of two similar layers and then repeat this model according to the number of layers that have similar behaviors.

In order to facilitate the modeling of the network of Fig. 8, we consider the CN-net model of Fig. 6 as consisting of two submodels as shown in Fig. 9: (i) a "submodel A" which is started from the places OUT-RES_1, and OUT-RES_2 to the transitions T_{shold3} and T_{shold4}; and (ii) a "submodel B" which is started from the place INP-NEU_5 to the place OUT-RES_5. According to the models shown in Fig. 6 and 9, a "submodel A" is used to describe the computation and communication behavior between the neurons of the input layer and those of the hidden layer. Also, a "submodel B" is used to describe the behavior between the neurons of the hidden layer and the neuron of the output layer.

For the purpose of modeling the network of Fig. 8, a "submodel A" can be used to model the dynamic behavior between any two similar layers (they have the same number of neurons) such as input layer and layer-1, layer-1 and layer-2, layer-2 and layer-3, as well as layer-3 and layer-4. Thus, the number of these dynamics is four. Since the neurons of these layers perform the same behavior, the dynamics among these layers are also samiliars. To model these dynamics, we can easily repeat the computation and communication behavior of a "submodel A" according to the number of dynamics among the desired layers. To model the mechanism of this repetition, we have used the places $COUNT_I$, $COUNT_{II}$, $TEST_I$, and $TEST_{II}$ as well as the transitions T_{countI}, $T_{countII}$, T_{testI}, and T_{testII} shown in Fig.9. Finally, a "submodel B" can be used to describe the dynamic behavior between the neurons of the layer-4 and the neuron of the output layer.

A place $COUNT_I$ ($COUNT_{II}$) is used to count the number of dynamics among the layers that should be repeated. As shown in Fig. 9, there are three colored tokens in each of the places $COUNT_I$ and $COUNT_{II}$. These colored tokens represent the number of dynamics among the layers: layer-1 and layer-2, layer-2 and layer-3, as well as layer-3 and layer-4. In the initial state of the model of Fig. 9, a place OUT-RES_1 contains the colored tokens $<DATA_{13}, (c_6, 1)>$, $<DATA_{14}, (c_4, 3)>$ and a place OUT-RES_2 contains the colored tokens $<DATA_{23}, (c_5, 2)>$, $<DATA_{24}, (c_3, 4)>$. By running these tokens in the "submodel A", we exhibit the computation and communication behavior between the input layer and layer-1. Once the results of this behavior reach the places $TEST_I$ and $TEST_{II}$, the transitions T_{countI} and $T_{countII}$ start to exhibit the dynamics among the layers 2, 3, and 4.

To organize our modeling, a place $TEST_I$ ($TEST_{II}$) is used to collect the results obtained from a neuron *i* (neuron *j*) in the layer k. These results are used as input data to the neurons of the layer k+1. Firing a transition T_{countI} ($T_{countII}$) denotes that the model is still performing the dynamics among the layers 2, 3, and 4. The firing of a transition T_{testI} (T_{testII}) indicates that the behaviors of these layers are exhibited. When the places OUT-RES_3, and OUT-RES_4 of Fig. 9 receive tokens from the transitions T_{testI} and T_{testII} respectively, the final results of the layer-4 of Fig. 8 are obtained. These results proceed to the place INP-NE_5 of the "submodel B" through the transitions $T_{send-I3}$ and $T_{send-I4}$, asshown in Fig. 9. Then, the "submodel B" is used to exhibit the dynamic behavior between the neuron of layer 4 and a neuron 11 of the output layer. When the place OUT-RES_5 receives a token, the final result of the network of Fig. 8 is obtained.

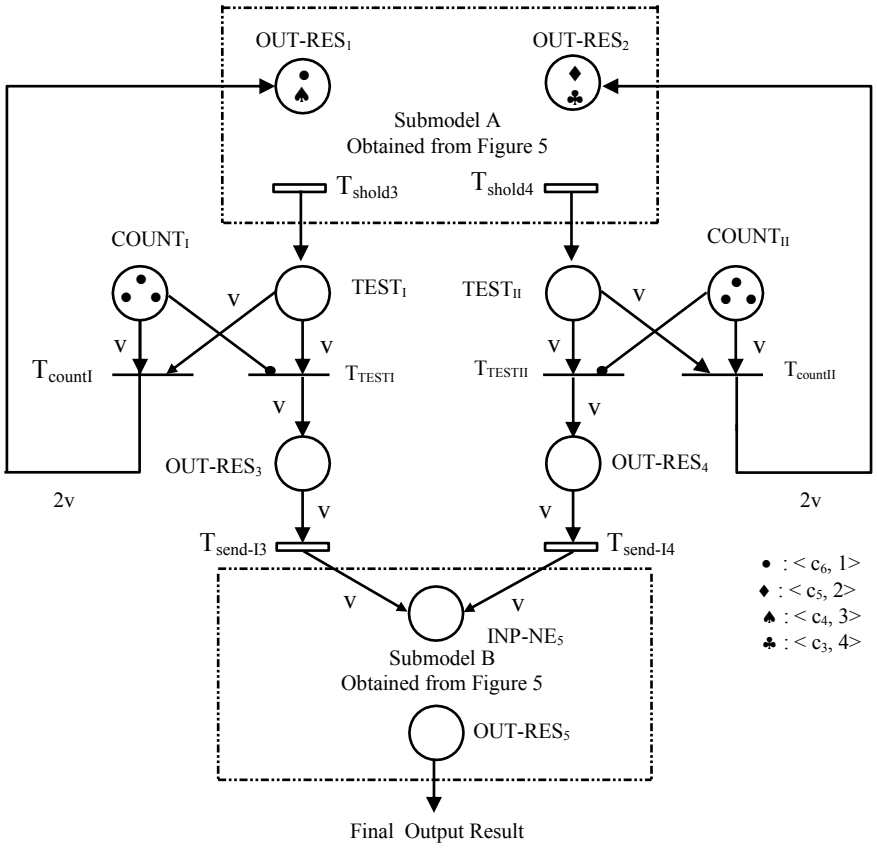

Fig. 9. A CN-Net model for the multilayer feedforward neural network of figures 8.

According to the above explanation, we can easily use our modular approach (e.g. submodel A and submodel B) to exhibit the behavior of n-layers feedforward neural networks such as that of Fig. 8. Based on this modular approach, the performance measure MPR of the network of Fig. 8 can be calculated as follows. From the TRG of Fig. 7, we calculate the MPR for the submodel A. Thus, the MPR of a submodel A is equal to $\tau_1 + \tau_2 + \tau_3 + \tau_3 + \tau_4 + \tau_5$ time units. Subsequently, the MPR of the layers 1, 2, 3 and 4 of Fig. 8 is equal to $4(\tau_1 + \tau_2 + \tau_3 + \tau_3 + \tau_4 + \tau_5)$ time units. From the TRG of Fig. 3, we calculate the MPR for the submodel B. Thus, the MPR of submodel B is equal to $\tau_1 + \tau_1 + \tau_2 + \tau_3$ time units. Finally, the MPR of a six-layer feedforward neural network of Fig. 8 is calculated as follows: $4 (\tau_1 + \tau_2 + \tau_3 + \tau_3 + \tau_4 + \tau_5) + (\tau_1 + \tau_1 + \tau_2 + \tau_3)$ time units.

One of the main problems in neural networks is to obtain a *topological structure* and a *learning rule* which guarantee the stability of the network [3]. It is clear from the network modeled above that the CN-net is capable of providing a mechanism for studying the different *topological structures* of the desired neural network (e.g. see Figures 6 and 9) as well as performing the various *learning algorithms* that are required for such network (e.g. see Fig. 7). Furthermore, we observe that the computations in a neural network are basically matrix products. Matrix products are also the only operations inside the network which absolutely require data communications between the processing elements of the neurons. Thus, if we implement the neural network onto VLSI chip, we can easily parallelize these matrix products. This parallelization achieves high speed computations and few data communications. Also, this implementation approach makes the matrix product algorithms run by each processing element identical and simple as well as makes the communication between processing elements very regular. Another implementation approach can be found in [3]. As shown in Figures 6 and 9, the CN-nets can model the neural network hierarchically; at various levels of abstraction and detail. This methodology enables us to design VLSI circuits for the complicated neural network in an elegant way.

To understand the mechanism with which the design of CN-net neural model can be transformed to VLSI circuit, we illustrate the isomorphisms between a neuron architecture and a CN-net concept.

- A place $p \in P$ represents the output through one arc. The place / soma is the "storage" or waiting element of the model.
- The arc between the place $p \in P$ and the transition $t \in T$ represents the axon and it is working with the multiplicity colored function G(a).
- The output arcs from the typical CN-net transition are corresponding to the axon terminals in the neuron.
- The input arcs to the place $p \in P$ represent the axon terminals from other neurons.

- The colored tokens in the CN-net represent the various input/output values that effect on the behavior of the neuron. The colored tokens carry the input data to the neuron, the corresponding weighted data, the threshold value of the neuron, or the output result of the neuron.
- The transitions in the CN-net represent the various computation (e.g. addition, multiplication and comparison) and communication (e.g. transmitting and receiving data between the active neuron and its neighboring neurons) events that occur in the neuron.
- The predefined delay times for the transitions of different colors represent the different capabilities of the various neuron cells.

5. Conclusion

We have developed a CN-net as a novel modeling technique for studying and analyzing the structure properties and dynamic behaviors of the ANNs. A generic CN-net model for a single artificial neuron is developed and analyzed. This generic model has the capability to accurately describe the characteristics of any neuron in the ANNs. To provide a practical insight into the application of CN-net technique to the ANNs, we have used this generic module for developing CN-net models for various feedforward neural network configurations. This study shows that the CN-net can be easily used as a modular approach for modeling the dynamics of n-layers feedforward neural networks.

We have explained how the formal description of the TRG of the desired CN-net model can be easily used for studying (i) the various steps of the learning algorithm that can be applied to this model; and (ii) the movement of data at the various elements of each neuron in the network. We have also explained how the CN-net model is very useful in the design of VLSI circuits for ANNs.

References

[1] Habib, M. K. and Akel, H. "A Digital Neuron Type Processor and Its VLSI Design." *IEEE Transactions on Circuits and Systems*, 36, No. 5 (May 1989) , 739-746.

[2] Kim, S.T., Suwunboriruksa, K., Herath, S. Jayasumana, A. and Herath, J. "Algorithmic Transformations for Neural Computing and Performance of Supervised Learning on a Data Flow Machine." *IEEE Transactions on Software Engineering*, 18, No. 7 (July 1992), 613-623.

[3] Petrowski, A., Dreyfus, G. and Girault, C. "Performance Analysis of a Pipelined Backpropagation Parallel Algorithm." *IEEE Transactions on Neural Networks*, 4, No. 6 (Nov. 1993), 970-981.

[4] Freedman, P. "Time, Petri Nets, and Robotics." *IEEE Transactions on Robotics and Automation*, 7, No. 4 (August 1991), 417-433.

[5] Zargham, M. R. and Jyman, M. "Neural Petri Nets." In: *Proceedings of the International Workshop Timed Petri Nets,* Torino, Italy (1985), 72-77.

[6] Ahson, S.I. "Petri Net Models of Fuzzy Neural Networks." *IEEE Transactions on Systems, Man, and Cybernetics*, 25, No. 6 (June 1995), 926-932.

[7] Chamas, N., Anneberg, L. and Yaprak, E. "Timed Neural Petri Nets." In: *Proceedings of the 36th Midwest Symposium on Circuits and Systems*, Detroit, MI, USA (16-18 August 1993) , 926-929.
[8] Hadjinicolaou, M. G., Abdelrazik, M. B. and Musgrave, G. "Structured Analysis for Neural Networks Using Petri Nets." In: *Proceedings of the 33rd. Midwest Symposium on Circuits and Systems*, Calgary, Alta., Canada (Aug. 1990) , 770-773.
[9] Habib, M. K. and Newcomb, R. W. "Neuron Type Processor Modeling Using a Timed Petri Net." *IEEE Transactions on Neural Networks*, 1, No. 4 (Dec. 1990), 282-289.
[10] Smith, L. S. "A Framework for Neural Net Specification." *IEEE Transactions on Software Engineering*, 18, No. 7 (July 1992) ,601-612.
[11] Gaeta, R. "Efficient Discrete-Event Simulation of Colored Petri Nets." *IEEE Transactions on Software Engineering*, 22, No. 9 (Sept. 1996).
[12] Lakos, C. "On the Abstaction of Colored Petri Net." In: *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, Toulouse, France (June 23-27, 1997).
[13] Shapiro, R. M. "Validation of a VLSI Chip Using Hierarchical Colored Petri Nets." *Journal of Microelectronics and Reliability*, Special Issue on Petri Nets, Pergamon Press (1991), 667-687.
[14] Stiege, G. "Equivalencies of Colored Petri Nets." *Petri Net Newsletters*, No. 52 (April 1997), 33-58.
[15] Aura, T. and Lilius, J. "Time Process for Time Petri Nets." In*: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, Toulouse, France (June 23-27), 1997.
[16] Nicol, D. M. and Mao, W. "Automated Parallelization of Timed Petri Net Simulations." *Journal of Parallel and Distributed Computing*, 29, No. 1 (August 1995), 60-74.
[17] Tanabe, M. "Timed Petri Nets and Temporal Linear Logic." In: *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, Toulouse, France (June 23-27, 1997).
[18] Zuberek, M. "Throughput Analysis in Timed Petri Nets." In: *Proceedings of the 35th Midwest Symposium on Circuits and Systems*, Washington DC, USA (August 9-12, 1992), 1576-1580.

Appendix

In the following, we formally describe the dynamic behavior of our CN-net XOR model of Fig. 6 through its developed TRG shown in Fig. 7. To simplify our explanation of this TRG, we give other names for the transitions and places of the model of Fig. 6 as shown in Table 2.

$S_1 : MRK_1 : \alpha(p_1) = <DATA_{13}, (c_6,1)>, <DATA_{14}, (c_4,3)>,$
$\alpha(p_2) = <DATA_{23}, (c_5,2)>, <DATA_{24}, (c_3,4)>$
$SET_1 : T_{NEW} : t_1 = <(c_6,1), (c_4,3)>, \quad t_2 = < (c_5,2), (c_3,4)>$
$T_{FIR} : t_1$ can fire, t_2 can fire

$S_2 : MRK_2 : \alpha(p_2) = <DATA_{23}, (c_5,2)>, <DATA_{24}, (c_3,4)>,$
$\alpha(p_3) = <x_1, (c_6,1)>, <x_1, (c_4,3)>,$
$\alpha(p_4) = <w_1, (c_6,1)>, <w_3,(c_4,3)>, \quad \alpha(p_7)= <s_1, (c_6,1)>$
$SET_2 : T_{FIR} : t_2 = < (c_5,2), (c_3, 4)>$
$INH_2 : H(p_7 <c_6,1>, t_1)$

$S_3 : MRK_3 : \alpha(p_1) = <DATA_{13}, (c_6,1)>, <DATA_{14}, (c_4,3)>,$
$\alpha(p_5) = <x_2, (c_5,2)>, <x_2, (c_3,4)>,$
$\alpha(p_6) = <w_4, (c_5,2)>, <w_2,(c_3,4)>, \quad \alpha(p_8) = <s_2, (c_5,2)>$
$SET_3 : T_{FIR} : t_1 = < (c_6,1), (c_4,3)>$
$INH_3 : H(p_8 <c_5, 2>, t_2)$

$S_4 : MRK_4 : \alpha(p_3) = <x_1, (c_6,1)>, <x_1, (c_4,3)>, \ \alpha(p_4) = <w_1, (c_6,1)>, <w_3,(c_4,3)>,$
$\alpha(p_7) = <s_1, (c_6,1)>, \quad \alpha(p_5) = <x_2, (c_5,2)>, <x_2, (c_3,4)>,$
$\alpha(p_6) = <w_4, (c_5,2)>, <w_2,(c_3,4)>, \quad \alpha(p_8) = <s_2, (c_5,2)>$
$SET_4 : T_{FIR} : t_3 = <COM_{13}, \tau_1, ((c_6,1), (c_4,3))>,$
$t_4 = <COM_{24}, \tau_1, ((c_5,2), (c_3,4))>$
$INH_4 : H(p_7 <c_6,1>, t_1), \ H(p_8 <c_5,2>, t_2)$

$S_5 : MRK_5 : \alpha(p_{10}) = <x_1, (c_6,1)>, <w_1, (c_6,1)>,$
$\alpha(p_{11}) = <x_2, (c_5,2)>, <w_4, (c_5,2)>,$
$\alpha(p_9) = <x_1, (c_4,3)>, <w_3, (c_4,3)>, <x_2, (c_3,4)>, <w_2, (c_3,4)>,$
$\alpha(p_7) = <s_1, (c_6,1)>, \quad \alpha(p_8)= <s_2, (c_5,2)>$
$SET_5 : T_{FIR} : t_5 = <(COM_{14}, COM_{23}), \tau_2, ((c_4,3), (c_3,4))>$
$INH_5 : H(p_7 <c_6,1>, t_1), \ H(p_8 <c_5,2>, t_2)$

$S_6 : MRK_6 : \alpha(p_{10}) = <x_1, (c_6,1)>, <w_1, (c_6,1)>, <x_2, (c_3,4)>, <w_2, (c_3,4)>,$
$\alpha(p_{11}) = <x_2, (c_5,2)>, <w_4, (c_5,2)>, <x_1, (c_4,3)>, <w_3, (c_4,3)>,$

$\alpha(p_7) = \langle s_1, (c_6,1)\rangle$, $\alpha(p_8) = \langle s_2, (c_5,2)\rangle$

SET_6 : T_{NEW} : $t_6 = \langle (c_6,1), (c_3,4)\rangle$ $t_7 = \langle (c_4,3), (c_5,2))\rangle$

T_{FIR} : t_6 can fire, t_7 can fire

INH_6 : $H(p_7 \langle c_6,1\rangle, t_1)$, $H(p_8 \langle c_5,2\rangle, t_2)$

S_7 : MRK_7 : $\alpha(p_{11}) = \langle x_2, (c_5,2)\rangle, \langle w_4, (c_5,2)\rangle, \langle x_1, (c_4,3)\rangle, \langle w_3, (c_4,3)\rangle$,

$\alpha(p_{12}) = \langle x_1, (c_6,1)\rangle, \langle x_2, (c_3,4)\rangle$,

$\alpha(p_{13}) = \langle w_1, (c_6,1)\rangle, \langle w_2, (c_3,4)\rangle$, $\alpha(p_{14}) = \langle \theta_3, (c_6,1)\rangle$

SET_7 : T_{FIR} : $t_7 = \langle (c_4,3), (c_5,2)\rangle$

INH_7 : $H(p_{14} \langle c_6,1\rangle, t_6)$

S_8 : MRK_8 : $\alpha(p_{10}) = \langle x_1, (c_6,1)\rangle, \langle w_1, (c_6,1)\rangle, \langle x_2, (c_3,4)\rangle, \langle w_2, (c_3,4)\rangle$,

$\alpha(p_{15}) = \langle x_2, (c_5,2)\rangle, \langle x_1, (c_4,3)\rangle$,

$\alpha(p_{16}) = \langle w_4, (c_5,2)\rangle, \langle w_3, (c_4,3)\rangle$, $\alpha(p_{17}) = \langle \theta_4, (c_5,2)\rangle$

SET_8 : T_{FIR} : $t_6 = \langle (c_6,1), (c_3,4)\rangle$

INH_8 : $H(p_{17} \langle c_5,2\rangle, t_7)$

S_9 : MRK_9 : $\alpha(p_{12}) = \langle x_1, (c_6,1)\rangle, \langle x_2, (c_3,4)\rangle$,

$\alpha(p_{13}) = \langle w_1, (c_6,1)\rangle, \langle w_2, (c_3,4)\rangle$, $\alpha(p_{14}) = \langle \theta_3, (c_6,1)\rangle$,

$\alpha(p_{15}) = \langle x_2, (c_5,2)\rangle, \langle x_1, (c_4,3)\rangle$,

$\alpha(p_{16}) = \langle w_4, (c_5,2)\rangle, \langle w_3, (c_4,3)\rangle$, $\alpha(p_{17}) = \langle \theta_4, (c_5,2)\rangle$

SET_9 : T_{FIR} : $t_8 = \langle (x_1 \times w_1), \tau_3, ((c_6,1), (c_3,4))\rangle$,

$t_9 = \langle (x_2 \times w_4), \tau_3, ((c_5,2), (c_4,3))\rangle$

INH_9 : $H(p_{14} \langle c_6,1\rangle, t_6)$, $H(p_{17} \langle c_5,2\rangle, t_7)$

S_{10} : MRK_{10} : $\alpha(p_{12}) = \langle x_2, (c_3,4)\rangle$, $\alpha(p_{13}) = \langle w_2, (c_3,4)\rangle$,

$\alpha(p_{14}) = \langle \theta_3, (c_6,1)\rangle$, $\alpha(p_{18}) = \langle x_1\, w_1, (c_6,1)\rangle$,

$\alpha(p_{15}) = \langle x_1, (c_4,3)\rangle$, $\alpha(p_{16}) = \langle w_3, (c_4,3)\rangle$,

$\alpha(p_{17}) = \langle \theta_4, (c_5,2)\rangle$, $\alpha(p_{19}) = x_2\, w_4, (c_5,2)\rangle$

SET_{10} : T_{FIR} : $t_8 = \langle (x_2 \times w_2), \tau_3, ((c_6,1), (c_3,4))\rangle$,

$t_9 = \langle (x_1 \times w_3), \tau_3, ((c_5,2), (c_4,3))\rangle$

INH_{10} : $H(p_{14} \langle c_6,1\rangle, t_6)$, $H(p_{17} \langle c_5,2\rangle, t_7)$

S_{11} : MRK_{11} : $\alpha(p_{18}) = \langle x_1\, w_1, (c_6,1)\rangle, \langle x_2\, w_2, (c_3,4)\rangle$,

$\alpha(p_{19}) = \langle x_2\, w_4, (c_5,2)\rangle, \langle (x_1\, w_3), (c_4,3)\rangle$

$\alpha(p_{14}) = \langle \theta_3, (c_6,1)\rangle$, $\alpha(p_{17}) = \langle \theta_4, (c_5,2)\rangle$

SET_{11} : T_{FIR} : $t_{10} = \langle (x_1\ w_1 + x_2\ w_2), \tau_4, ((c_6,1), (c_3,4))\rangle$,

$t_{11} = \langle (x_1\ w_3 + x_2\ w_4), \tau_4, ((c_5,2), (c_4,3))\rangle$

INH_{11} : $H(p_{14} \langle c_6,1\rangle, t_6)$, $H(p_{17} \langle c_5,2\rangle, t_7)$

S_{12} : MRK_{12} : $\alpha(p_{20}) = \langle \Sigma_3, (c_6,1)\rangle$, $\alpha(p_{21}) = \langle \Sigma_4, (c_5,2)\rangle$,

$\alpha(p_{14}) = <\theta_3, (c_6,1)>$, $\alpha(p_{17}) = <\theta_4, (c_5,2)>$
$SET_{12} : T_{FIR} : t_{12} = <(\Sigma_3 \otimes \theta_3), \tau_5, (c_6,1)>$, $t_{13} = <(\Sigma_4 \otimes \theta_4), \tau_5, (c_5,2)>$,
INH_{12} : $H(p_{14} <c_6,1>, t_6)$, $H(p_{17} <c_5,2>, t_7)$

$S_{13} : MRK_{13} : \alpha(p_{22}) = <DATA_{35}, (c_6,1)>$, $\alpha(p_{23}) = <DATA_{45}, (c_5,2)>$
$SET_{13} : T_{FIR} : t_{14} = <COM_{35}, \tau_6, (c_6,1) >$, $t_{15} = <COM_{45}, \tau_6, (c_5,2)>$
$S_{14} : MRK_{14} : \alpha(p_{24}) = <DATA_{35}, (c_6,1)>, <DATA_{45}, (c_5,2)>$,
$SET_{14} : T_{FIR} : t_{16} = <(c_6,1), (c_5,2)>$

$S_{15} : MRK_{15} : \alpha(p_{25}) = <RES_3, (c_6,1)>, <RES_4, (c_5,2)>$,
$\alpha(p_{26}) = <w_5, (c_6,1)>, <w_6, (c_5,2)>$
$\alpha(p_{27}) = <\theta_5, (c_6,1)>$
$SET_{15} : T_{FIR} : t_{17} = <(RES_3 \times w_5), \tau_7, ((c_6,1), (c_5,2))>$
INH_{15} : $H(p_{27} <c_6,1>, t_{16})$

$S_{16} : MRK_{16} : \alpha(p_{25}) = <RES_4, (c_5,2)>$, $\alpha(p_{26}) = <w_6, (c_5,2)>$,
$\alpha(p_{28}) = <(RES_3\ w_5), (c_6,1)>$
$\alpha(p_{27}) = <\theta_5, (c_6,1)>$
$SET_{16} : T_{FIR} : t_{17} = <(RES_4 \times w_6), \tau_7, ((c_6,1), (c_5,2))>$
INH_{16} : $H(p_{27} <c_6,1>, t_{16})$

$S_{17} : MRK_{17} : \alpha(p_{28}) = <(RES_3\ w_5), (c_6,1)>, <(RES_4\ w_6), (c_5,2)>$
$\alpha(p_{27}) = <\theta_5, (c_6,1)>$
$SET_{17} : T_{FIR} : t_{18} = <((RES_3\ w_5) + (RES_4\ w_6)), \tau_8, ((c_6,1), (c_5,2))>$
INH_{17} : $H(p_{27} <c_6,1>, t_{16})$

$S_{18} : MRK_{18} : \alpha(p_{29}) = <\Sigma_5, (c_6,1)>$, $\alpha(p_{27}) = <\theta_5, (c_6,1)>$
$SET_{18} : T_{FIR} : t_{19} = <(\Sigma_5 \otimes \theta_5), \tau_9, (c_6,1)>$
INH_{18} : $H(p_{27} <c_6,1>, t_{16})$

$S_{19} : MRK_{19} : \alpha(p_{30}) = <RES_5, (c_6,1)>$

شبكات "سي إن" لتمثيل وتحليل الشبكات العصبية

سمير إم. كريم
قسم نظم وهندسة الحاسب، كلية الهندسة،
جامعة الأزهر، القاهرة، مصر
(قدّم للنشر في ١٩٩٨/٢/٢٤م؛ وقبل للنشر في ٢٠٠٠/٢/٩م)

ملخص البحث. يقدم هذا البحث مفهوم شبكات "بتري العصبية الزمنية الملونة" (اختصارا يطلق عليها شبكات "سي إن") التي تشبه في أدائها الهياكل العصبية. والتكنيك الخاص لشبكات "سي إن" يتضمن المعالم المميزة للشبكات العصبية، بالإضافة إلى إمكانية التمثيل التي تتمتع بها كلا من شبكات بتري الزمنية والملونة. وهذا البحث يقدم شرحا وافيا للأساسيات الضرورية لبناء شبكات "سي إن". ولقد تم إظهار المقدرة الحسابية لنموذج شبكات "سي إن" من خلال "الشكل الزمني الموصل" المستنتج من هذا النموذج. وصممت شبكات "سي إن" بغرض دراسة الخواص البنائية للشبكات العصبية الصناعية بينما استخدم "الشكل الزمني الموصل" للتحقق من صحة السلوك الديناميكي لهذه الشبكات. بالإضافة إلى ذلك فإن شبكات "سي إن" تقدم تمثيلا سهلا ومقروءا للنموذج المطلوب، مما يسهل تصميم الدوائر المتكاملة كثيفة العدد المستخدمة في الشبكات العصبية المعقدة. وقد تم تقديم أمثلة عملية لتوضيح طريقة توظيف شبكات "سي إن" كتقنية جديدة لها القدرة على محاكاة السلوك الديناميكي والنشاطات المتوازية للشبكات العصبية الصناعية.