Acta Informatica
DOI 10.1007/s00236-008-0084-1

ORIGINAL ARTICLE

Exploiting colored Petri nets to decide on permutation admissibility

Rza Bashirov · Fabrice Kordon · Hüseyin Lort

Received: 29 June 2007 / Revised: 2 October 2008

Abstract In this work, we propose an innovative approach to investigate the admissibility of permutations to multistage interconnection networks—a challenging problem of switching theory. The proposed approach is centered upon modeling of multistage interconnection networks with colored Petri nets and use of Petri net analysis tools such as the unfolding technique and the invariants method. To assess the feasibility of the proposed approach we demonstrate that the complete unfoldings obtained in this work are polynomial in the problem size and employ an acyclic structure. The approach takes advantage of easy to use, yet extremely efficient, software tools.

1 Introduction

The performance of multiprocessor systems today is limited by their communication or interconnection, not by their logic or memory. Designing fast interconnection networks therefore becomes a critical issue to exploit the performance of multiprocessors. Data routing between the local memories of processing elements in multiprocessor systems can be represented as a variety of regular and irregular permutations of the network's inputs into its outputs. Efficient data routing in multiprocessors essentially depends on whether permutations demanded by a particular application are admissible to the network topology.

This work has been partially supported by the TRNC Ministry of Education and Culture under project MEKB-08-09.

R. Bashirov (✉) · H. Lort
Department of Applied Mathematics and Computer Science,
Eastern Mediterranean University, via Mersin 10, North Cyprus, Turkey
e-mail: rza.bashirov@emu.edu.tr

H. Lort
e-mail: huseyin.lort@emu.edu.tr

F. Kordon
Université P. & M. Curie, LIP6/MoVe, 4 place Jussieu, 75252 Paris Cedex 05, France
e-mail: fabrice.kordon@lip6.fr

Published online: 28 November 2008

We say an $N \times N$ multistage interconnection network (MIN) generates a permutation $\pi : N \to N$ if there exists a setting of the switches in the MIN such that the input i is connected to the output $\pi(i)$ for $0 \leqslant i < N$. Determining whether a given permutation can be generated by a MIN in a single pass is referred to as the *permutation admissibility problem*. Deciding on permutation admissibility in an arbitrary MIN is a well-known difficult problem. For instance, it has been reported [20,21] that the permutation admissibility problem for some MINs is equivalent to 2^k-colorability in graphs, which is a NP-complete problem.

In this paper, we present a colored Petri net (CP-net) model of a MIN and show how the permutation admissibility problem can be analyzed in terms of CP-nets. In this model, a permutation is represented as a place marking. A permutation is admissible to the MIN if the marking induced by this permutation is reachable from the initial marking in associated CP-net. This constitutes the key idea behind of this work: *by adopting CP-nets for MIN modeling we reduce the permutation admissibility problem to the reachability in CP-nets.*

To verify the reachability in a CP-net we first unfold the CP-net into equivalent place/transition net (P/T-net) and then implement the invariants method to related P/T-net. The proposed approach takes advantage of powerful existing software tools, particularly, CPN-AMI [14], a Petri net CASE (Computer Aided Software Engineering) environment for the verification of concurrent systems, which is used to unfold CP-nets into equivalent optimized complete unfoldings. The approach is easy to use and can be efficiently applied to MINs made of crossbar switches.

The paper is organized as follows. Section 2 reviews key work, to date, in this field. Section 3 gives background on frequently used permutations and interconnection networks, provides basics about CP-nets, and succinctly overviews the optimized unfolding technique and the invariants method. The main results are presented in Sects. 4, 5, and 6. Section 4 describes the CP-net model adopted in this paper. Section 5 presents the feasibility analysis for the unfolding technique and the invariants method regarding the Petri nets obtained in this work. Section 6 presents and analyzes the experimental results. Finally, Sect. 7 contains conclusions and observations about future work.

2 Related work

Much research work has been done on permutation admissibility of MINs. The research in this area is mainly focused on two aspects of permutation admissibility: determining the type of permutations that are admissible to the given MIN and finding out a MIN topology that would best match the given class of permutations.

In the first direction, the researchers have extensively studied the admissibility of permutations to $N \times N$ full-access and unique-path MINs made of 2×2 switches. There are only $N^{N/2}$ out of $N!$ possible permutations that are admissible to any $N \times N$ full-access and unique-path MIN. Determining the permutations that are indeed admissible to this type MINs is an important problem in optimally supporting application needs.

A necessary condition that a permutation must satisfy for being admissible to an m stage shuffle-exchange network (SEN) for $\log_2 N < m \leqslant 2\log_2 N - 1$, was formulated in [5]. In [6] this result was used to determine the minimum number of stages that is necessary for permutations to be admissible to an optical MIN. An $O(Nn)$ algorithm that determines whether a permutation is admissible to $N \times N$ multistage cube-type networks (MCTNs), that are topologically equivalent to many full-access and unique-path MINs [24], was introduced in [20]. In [21] this result was extended to k-extra stage MCTN with $k = 1$. In the same paper it was shown that a permutation is admissible to a k-extra stage MCTN if and only

if the conflict graph is 2^k-colorable. NP-completeness of 2^k-coloring problem in graphs, for $k > 1$, does not permit the development of general methods to analyze the permutation admissibility with polynomial dependence on the number of extra stages. Although there exist efficient algorithms for checking the permutation admissibility for $k = 0$ and $k = 1$, no such algorithm is known for $k > 1$ [6]. In other works [4,22,23] permutation admissibility of binary hypercube, R-path omega and generalized SENs were investigated both at theoretical and algorithmic levels.

On a different tangent, many researchers have studied the admissibility of frequently used regular permutations which belong to BP (bit permute), BPC (bit permute complement), LIN (linear combination), LC (linear combination complement) and other permutation classes [6,9,10,16,19–24]. A comprehensive survey of permutations frequently used in parallel processing is given in [8].

The approach proposed in this paper differs from existing ones in three aspects. Firstly, aforesaid works are mainly restricted to regular permutation classes such as BP, BPC, etc. There are $n!$ bit permute permutations, and we can generate 2^n bit permute complement permutations from each bit permute permutation, meaning that there are $n!2^n$ distinct bit permute complement permutations [6,8,23]. It is known that there are $2^{n(n+1)/2} \prod_{i=1}^{n} (2^i - 1)$ permutations in LC class [18]. However, there still exist important permutations of more peculiar structure that belong to neither of these permutation classes. Some examples of regular permutations that do not belong to BPC are [8]:

* *cyclic shift of amplitude k* permutation defined as $\pi(x) = (jx + k) \mod N$ where $1 \leqslant k \leqslant N$ and j is odd number;
* *cyclic shift within segments* permutation defined as $\pi(x) = \delta_2(x+k) \mod 2^{n-j}$ where δ_2 is the decimal number equivalent to the j most significant bits in the binary representation of x, and $1 \leqslant k \leqslant N$;
* *unscrambling j-ordered vectors* permutation defined as $\pi(x) = (jx) \mod 2^k + (x_n, x_{n-1}, \ldots, x_{k+1})2^k$ with $1 \leqslant k \leqslant N$ where j is odd numbers.

This paper explores the admissibility of an arbitrary permutation.

Secondly, there are a variety of MIN structures of practical interest that are not topologically equivalent to MCTN. These MINs are not covered by traditional algorithms. Our approach is applicable to any MIN made of crossbar switches though we use 8×8 and 16×16 SENs as examples to illustrate the proposed technique.

Finally, traditional algorithms are mostly based on linear algebraic and graph theoretic methods such as the balanced matrices characterization or the window method. In this paper we exploit CP-nets to decide on permutation admissibility.

3 Preliminaries

3.1 Permutations and interconnection networks

Let $x, y \in X$ where $X = \{0, 1, \ldots, N-1\}$, $N = 2^n$, $x = x_n x_{n-1} \cdots x_1$ and $y = y_n y_{n-1} \cdots y_1$ in binary with $y_i, x_i \in \{0, 1\}$ for $1 \leqslant i \leqslant n$. A permutation π is a bijection $\pi : X \to X$. Although expressed in a different form, the following definitions are equivalent to the ones given in many papers.

The *bit complement permutation* is obtained using only operations defined as $C(k) : x_n \cdots x_k \cdots x_1 \to x_n \cdots \overline{x}_k \cdots x_1$. The class of all bit complement permutations is denoted by BC. The *bit complement permutation* involves only operations defined as $P(r, t)$:

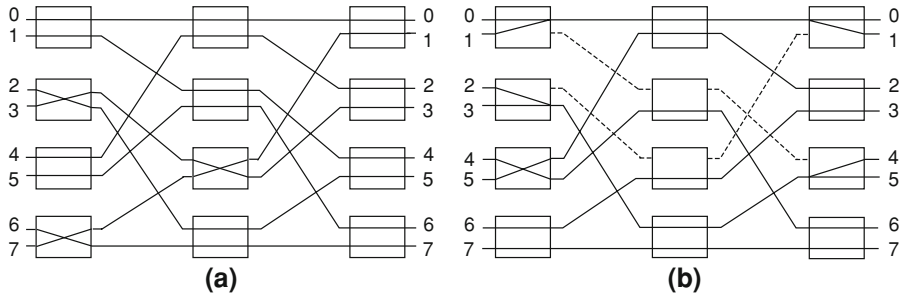

Fig. 1 An illustration of permutation admissibility: **a** 04532671 is admissible; **b** 01456271 is not admissible to 8 × 8 3-stage SEN

$x_n \cdots x_r \cdots x_t \cdots x_1 \rightarrow x_n \cdots x_t \cdots x_r \cdots x_1$. The class of all bit permute permutations is denoted by BP. The *bit permute complement permutation* is a composition of operations of the type $PC(r,t) : x_n \cdots x_r \cdots x_t \cdots x_1 \rightarrow x_n \cdots x_t \cdots \overline{x}_r \cdots x_1$. The class of all bit permute complement permutations is denoted by BPC.

A permutation is a *linear permutation*, if there exists a nonsingular (or invertible) binary matrix $[\lambda_{ij}]$ such that $y_i = \sum_{j=1}^{n} \lambda_{ij} x_j$, $1 \leqslant i \leqslant n$ where the addition and multiplication operations are of those modulo 2 arithmetic. The class of all linear permutations is denoted by LIN. The permutation is a *linear complement permutation* if and only if $y_i = \sum_{j=1}^{n} \lambda_{ij} x_j \oplus 1$ for $1 \leqslant i \leqslant n$, where $\oplus$ stands for modulo 2 bitwise addition operation. The class of all linear complement permutations is denoted by LC. Finally, the *perfect shuffle* σ is a permutation $\sigma : X \rightarrow X$ such that $\sigma(x) = x_{n-1}x_{n-2} \cdots x_1 x_n$, that is, $\sigma(x)$ is a left circular shift of the bits in the binary representation of x.

Each (2×2)-switch is a two-input/two-output device that can be set to either *through* state or *cross* state. A MIN is typically organized in stages of 2×2 switches and described by the set of interconnection patterns employed between neighboring stages. MCTNs represent a minimal cost network structure since they provide exactly one path between any input and output pair. n stages of 2×2 switches with perfect shuffle pattern between neighboring stages is an example of MCTN (see Fig. 1). A MIN is said to be *rearrangeable*, if it is capable of performing all the permutations of MIN's inputs into its outputs.

3.2 Colored Petri nets

P/T-nets are mathematical and graphical modeling tool that are evolved from finite state automata to describe and analyze the problems arising in scientific, engineering and industrial domains. CP-nets are a class of flexible specification languages generalized from P/T-nets. CP-nets are suitable for modeling and analysis of concurrent systems with dynamically changing structured objects.

Below we recall basic concepts about CP-nets and refer readers to [11,12] for detailed information on CP-net formalism.

3.2.1 Basic definitions

A P/T-net is a 5-tuple, PN $= (\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{W}, \mathcal{M}_0)$ where $\mathcal{P}$ is the finite set of *places*, $\mathcal{T}$ is the finite set of *transitions*, such that $\mathcal{P} \cap \mathcal{T} = \emptyset$ and $\mathcal{P} \cup \mathcal{T} \neq \emptyset$, $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is

the finite set of *arcs*, $\mathcal{W} : \mathcal{A} \to \mathbb{N}$ is the *weight* function, and $\mathcal{M}_0 : \mathcal{P} \to \mathbb{N}$ is the *initial marking*.

A CP-net is a 9-tuple [11] CPN $= (\Sigma, \mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{C}, \mathcal{G}, \mathcal{E}, \mathcal{I})$, where Σ is the finite set of non-empty *color sets*, $\mathcal{P}$ is the finite set of *places*, $\mathcal{T}$ is the finite set of *transitions*, $\mathcal{A}$ is the finite set of *arcs*, $\mathcal{N} : \mathcal{A} \to \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$ is the *node function*, $\mathcal{C} : \mathcal{P} \to \Sigma$ is the *color function*, $\mathcal{G}$ is the *guard function*, $\mathcal{E}$ is the *arc expression function* and $\mathcal{I}$ is the *initialization function*.

A *binding* b of a transition $t \in \mathcal{T}$ replaces each variable occurring in the arc expressions on the surrounding arcs of t with an element of an appropriate color set. The set of bindings for t is denoted as $\mathcal{B}(t)$. A *token element* is a pair (p, c) where $p \in \mathcal{P}$ and $c \in \mathcal{C}(p)$. A *marking* and a step Y are non-empty and finite multi-sets over the set of binding elements. A step Y is *enabled* in a marking M, i.e., ready to occur, if $\forall p \in \mathcal{P} : \sum_{(t,b)\in Y} \mathcal{E}(p, t) < b > \leq M(p)$. An enabled step Y may *occur* (or *fire*), changing the present marking M_1 into another marking M_2.

A marking M_2 is *directly reachable* from M_1, denoted by $M_1[Y \succ M_2$, if there exists a step Y changing M_1 into M_2. A *finite occurrence sequence* is a sequence of markings and steps $M_1[Y_1 \succ M_2[Y_2 \succ M_3 \ldots M_n[Y_n \succ M_{n+1}$ for $n \in \mathbb{N}$. Markings M_1 and M_{n+1} are called respectively *start* (or *initial*) and *end* marking, while the value n is the *length* of the occurrence sequence. A marking M_{n+1} is *reachable* from M_1, if there exists a finite occurrence sequence changing marking M_1 into marking M_{n+1}.

A finite occurrence sequence with matching start and end markings forms a *directed circuit*. A CP-net contains a directed circuit if and only if $\exists M^{'} \in \mathbb{M}$ such that $M^{'} \in [M^{'} \succ$. A CP-net having no directed circuits is called an *acyclic* CP-net.[1]

3.2.2 Unfolding technique

Unfolding technique is a method used to transform a high-level Petri net into P/T-net preserving the main properties of the original net. Both our understanding of the unfolding technique and its practical use in this work are influenced by optimized unfolding of CP-nets, proposed in [15]. To increase the readability of the paper below we succinctly describe main phases of the optimized unfolding technique and refer readers to [15] for detailed explanation of these phases.

1. Unfold each colored place $p \in \mathcal{P}$ into a set of places $p^{'} \in \mathcal{P}^{'}$, one for each color of tokens $c \in \mathcal{C}(p)$.
2. Unfold each colored transition $t \in \mathcal{T}$ into a set of transitions $t^{'} \in \mathcal{T}^{'}$, one for each binding $b \in \mathcal{B}(t)$.
3. Represent an unfolded net in a compact symbolic form by utilizing Data Decision Diagrams.
4. Optimize an unfolded net by removing all unnecessary components such as 0-bounded places, transitions with false-valued guards, etc.

Below we illustrate construction of optimized complete unfolding introducing a simple example of (2×2)-switch. A CP-net shown in Fig. 2a is a model of (2×2)-switch, which is detailed in Sect. 4. Corresponding complete unfolding is represented in Fig. 2b. In this figure, switch_1_1 and switch_0_0 are unnecessary components since these transitions are disabled in the initial marking and will stay so permanently. By removing switch_1_1, switch_0_0 together with the arcs surrounding these components we obtain optimized complete unfolding shown Fig. 2c.

[1] Acyclicity in P/T-nets can be defined similarly to that in CP-nets.

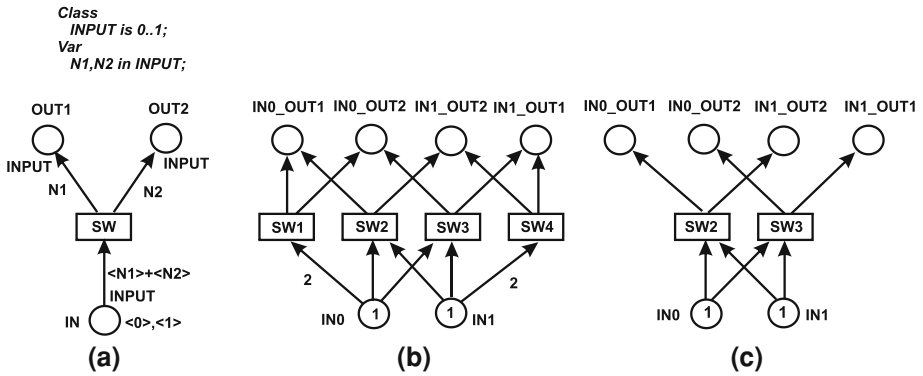

Fig. 2 An illustration of the optimized unfolding technique: **a** CP-net model of (2×2)-switch; **b** related complete unfolding; **c** unfolded net after optimization

3.2.3 Invariants method

For a P/T-net with n transitions and m places, the incidence matrix $A = [a_{ij}]$ is an $n \times m$ matrix with integer entries defined as $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$ where $a_{ij}^{+} = w(i, j)$ is the weight of the arc from transition i to its output place j and $a_{ij}^{-} = w(j, i)$ is the weight of the arc to transition i from its input place j. Each invariant in a P/T-net can be expressed as the system of linear algebraic equations called state equation

$$A^T \cdot x = M_d - M_0 \tag{1}$$

where x is an $n \times 1$ column vector of nonnegative integers and is called the *firing count vector*, M_0 is the initial marking, and M_d is the destination marking. The ith entry of x denotes the number of times that transition i must fire to transform M_0 to M_d.

It has been shown [17] that the existence of a nonnegative integer solution x satisfying the state equation (1) is a necessary but, in general, not sufficient condition for M_d to be reachable from M_0. For acyclic P/T-nets the above condition is also sufficient.

As an immediate consequence of the above theorem, we can now use the state equation (1) to verify the reachability in acyclic P/T-nets. Given acyclic P/T-net and two markings M_0 and M_d, we only need to write related state equation and find a nonnegative integer solution x of the state equation. If such x exists then M_d is reachable from M_0. Otherwise, it is not.

4 Modeling with CP-nets

Below we formally define CP-net model of a MIN and detail its components, which will be instrumental in successive sections.

Definition 1 [2] A CP-net of a (2×2)-switch is a 9-tuple $\text{CPN-SW}_{(2\times2)} = (\Sigma, \mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{C}, \mathcal{G}, \mathcal{E}, \mathcal{I})$, such that

- $\Sigma = \{INPUT\}$, *and* INPUT *is an integer color set with 0 and 1*;
- $\mathcal{P} = \{IN, OUT1, OUT2\}$;
- $\mathcal{T} = \{SW\}$;
- $\mathcal{A} = \{INtoSW, SWtoOUT1, SWtoOUT2\}$;

- $\mathcal{N}(a) = (SOURCE, DESTINATION)$, *if* a *is in the form* $SOURCEtoDESTINATION$;
- $\mathcal{C}(p) = INPUT \ \ \forall p \in \mathcal{P}$;
- $\mathcal{G}(t) = \text{true} \ \ \forall t \in \mathcal{T}$;
- $\mathcal{E}(a) = \begin{cases} 1'N1+1'N2, & \text{if } a = INtoSW \\ N1, & \text{if } a = SWtoOUT1 \\ N2, & \text{if } a = SWtoOUT2; \end{cases}$
- $\mathcal{I}(p) = \begin{cases} 1'0+1'1, & \text{if } p = IN \\ \text{empty}, & \text{otherwise.} \end{cases}$

The corresponding CP-net diagram is shown in Fig. 2a. In this figure, place IN represents the inputs of the (2×2)-switch. Place OUT1 (OUT2) corresponds to switch output 0 (1). The two steps $Y_1 = (\text{SW}, < \text{N1} = 0, \text{N2} = 1 >)$ and $Y_2 = (\text{SW}, < \text{N1} = 1, \text{N2} = 0 >)$ are enabled in M_0. An occurrence of Y_1 changes M_0 into $M_1 = 1'(\text{OUT1}, 0) + 1'(\text{OUT2}, 1)$, indicating that the (2×2)-switch is set through. Similarly, an occurrence of Y_2 changes M_0 into $M_2 = 1'(\text{OUT1}, 1) + 1'(\text{OUT2}, 0)$, which sets the (2×2)-switch cross. Then the CP-net becomes dead since both M_1 and M_2 are dead markings.

The CP-net model of (2×2)-switch fully describes the switch functionality. The through (cross) state of the (2×2)-switch can be obtained by the occurrence of Y_1 (Y_2).

Definition 2 A CP-net of $2^n \times 2^n$ m-stage interconnection network is given by a 9-tuple $\text{CPN-MIN}_{2^n,\,m} = (\Sigma, \mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{C}, \mathcal{G}, \mathcal{E}, \mathcal{I})$, such that

- $\Sigma = \{INPUT\}$, *where* $INPUT = \{0, \ldots, 2^n - 1\}$,
- $\mathcal{P} = \mathcal{P}_{IN} \cup \mathcal{P}_{AUX} \cup \mathcal{P}_{OUT} \cup \mathcal{P}_{CDN}$, *where*
  $\mathcal{P}_{IN} = \{INi\}, i = 1, \ldots, 2^{n-1}$, *is the set of* input *places,*
  $\mathcal{P}_{AUX} = \{AUXi\}, i = 1, \ldots, (m-1)2^{n-1}$, *is the set of* auxiliary *places,*
  $\mathcal{P}_{OUT} = \{OUTi\}, i = 1, \ldots, 2^n$, *is the set of* output *places,*
  $\mathcal{P}_{CDN} = \{Ci\}, i = 1 \ldots, m2^{n-1}$, *is the set of* condition *places;*
  $\mathcal{P}_{IN} \cap \mathcal{P}_{AUX} \cap \mathcal{P}_{OUT} \cap \mathcal{P}_{CDN} = \emptyset$,
- $\mathcal{T} = \{SWi\}, i = 1, \ldots, m2^{n-1}$, *is the set of transitions,*
- $\mathcal{A} = \mathcal{A}_{ST} \cup \mathcal{A}_{PTN} \cup \mathcal{A}_{CDN} \cup \mathcal{A}_{OUT}$, *where*
  $\mathcal{A}_{ST} \subseteq (\mathcal{T} \times \mathcal{P}_{AUX}) \cup (\mathcal{P}_{AUX} \times \mathcal{T})$ *is the set of* straight *arcs,*
  $\mathcal{A}_{PTN} \subseteq (\mathcal{P}_{IN} \times \mathcal{T}) \cup (\mathcal{P}_{AUX} \times \mathcal{T})$ *is the set of* pattern *arcs,*
  $\mathcal{A}_{CDN} \subseteq (\mathcal{P}_{CDN} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P}_{CDN})$ *is the set of* condition *arcs,*
  $\mathcal{A}_{OUT} \subseteq (\mathcal{T} \times \mathcal{P}_{OUT})$ *is the set of* output *arcs*;
- $\mathcal{N}(a) = (SOURCE, DESTINATION)$, *if* a *is in the form* $SOURCE\text{to}DESTINATION$;
- $\mathcal{C}(p) = INPUT, \forall p \in \mathcal{P}_{IN} \cup \mathcal{P}_{AUX} \cup \mathcal{P}_{OUT}$;
- $\mathcal{G}(t) = \text{true} \ \ \forall t \in \mathcal{T}$;
- $\mathcal{E}(a) = (Var(a))_{MS}\ s.\ t.\ Type(Var(a)) = INPUT, \forall p \in \mathcal{P}_{IN} \cup \mathcal{P}_{AUX} \cup \mathcal{P}_{OUT}$;
- $\mathcal{I}(p) = \begin{cases} 1'(2i-2) + 1'(2i-1), & \text{if } p = INi \text{ for } i = 1, \ldots, 2^{n-1} \\ 1, & \text{if } p = Ci \\ \text{empty}, & \text{otherwise.} \end{cases}$

$\text{CPN-MIN}_{2^n,\,m}$ is composed of transitions representing 2×2 switches, and places (excluding condition places) representing inputs and outputs of individual switches. Condition places together with condition arcs are used to impose occurrence of the steps according to predefined order given by

$$M_0[Y_1 \succ M_1[Y_2 \succ M_2 \cdots M_{m2^{n-1}-1}[Y_{m2^{n-1}} \succ M_{m2^{n-1}} \tag{2}$$

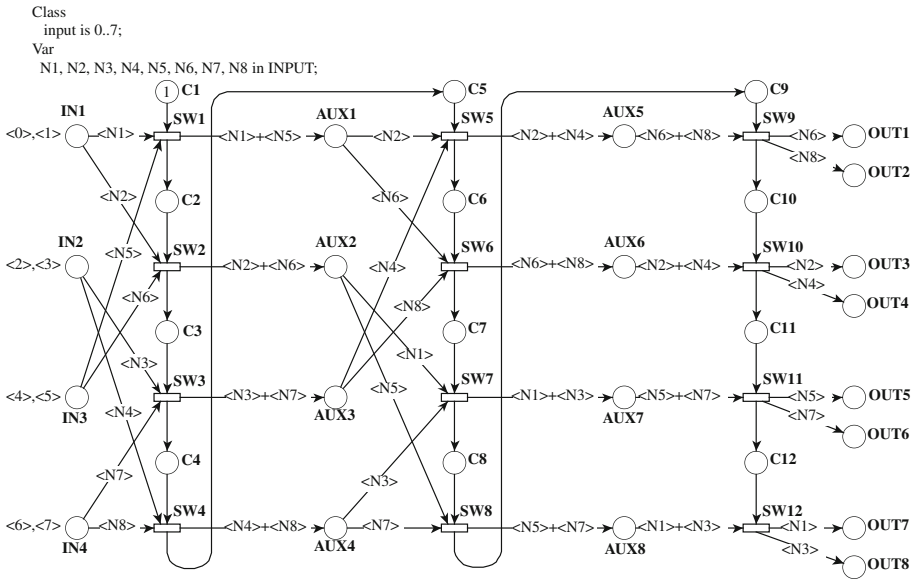

Fig. 3 A CP-net of 8 × 8 3-stage SENs

where $Y_i = (\mathrm{SW}i, b)$ and $b \in \mathcal{B}(\mathrm{SW}i)$ for $1 \leqslant i < m2^{n-1}$. Every transition appears exactly once in this occurrence sequence. Pattern arcs correspond to interstage channels linking neighboring stages in a MIN. Output arcs are used to display the final arrangement of tokens in output places. We control the flow of tokens in CPN-MIN$_{2^n,\,m}$ by evaluating arc expressions belonging to related straight arcs.

As an example consider a CP-net diagram (Fig. 3) of 8 × 8 3-stage SEN (Fig. 1). In order to ease the explanation, the net components are grouped in columns (Fig. 3) similar to the way the switches are arranged in stages (Fig. 1). Thus, whole net is represented as a cascade of columns alternating in type of the components being either place or transition. Transitions occur column-wise from the leftmost to the rightmost and in columns from the topmost to the bottommost. It can be easily seen that no token can visit a place more than once, and thus occurrence-depth of every transition is 1. Direction of the arcs indicates the flow of tokens through the net.

5 Characterization of complete unfoldings

The size of a complete unfolding, which we measure in terms of number of places and number of transitions, is directly proportional to storage capacity demanded to store the complete unfolding. It is quite often the case that large complete unfolding causes memory overflow and becomes a bottleneck of an application. In general, the unfolding techniques yield P/T-nets that are larger than the original CP-nets. It is quite usual that an unfolded P/T-net exponential in the size of the original CP-net is built even for a relatively simple problem [13]. The following proposition shows that for the CPN-MIN$_{2^n,\,m}$ the algorithm described in Sect. 3.2.2 builds complete unfoldings that are polynomial in the MIN's size N, and consequently more sharper upper bounds can be obtained for optimized complete unfoldings.

Let $|T_{CP}|$, $|P_{CP}|$, $|T_{Unf}|$ and $|P_{Unf}|$ be the number of the transitions and places in CPN-MIN$_{2^n,\,k}$ and corresponding unfolded net, respectively.

Proposition 1 *The following holds*:

$$|T_{CP}| = \frac{Nk}{2}; \quad |T_{Unf}| = \frac{N^3k}{2}; \quad |P_{CP}| = N(k+1) \quad \textit{and} \quad |P_{Unf}| = \frac{N^2(k+2)}{2} + \frac{Nk}{2}.$$

Proof In fact, we obtain an exact bound for $|T_{CP}|$ as a consequence of the following observations. It turns out that CPN-MIN$_{2^n,\,k}$ is composed of $\frac{Nk}{2}$ net patterns with each pattern representing CPN-SW$_{(2\times2)}$, similar to the way $2^n \times 2^n$ k-stage MIN is made of $\frac{Nk}{2}$ 2×2 switches. Since CPN-SW$_{(2\times2)}$ contains a single transition, we conclude that $|T_{CP}| = \frac{Nk}{2}$.

We can easily estimate $|T_{Unf}|$ by taking into account that the unfolding algorithm creates one transition for each binding and that each binding can be regarded as a 2-color sample $(c_1, c_2) \in \text{INPUT} \times \text{INPUT}$ where INPUT $\in \Sigma$ and $|\text{INPUT}| = N$. If repetition is allowed and order is important then there exist N^2 ways to compose 2-color samples from a color set with N colors, meaning that $|T_{Unf}| = N^2|T_{CP}| = \frac{N^3k}{2}$.

It is worth to mention that $\mathcal{P} = \mathcal{P}_{IN} \cup \mathcal{P}_{AUX} \cup \mathcal{P}_{OUT} \cup \mathcal{P}_{CDN}$ due to Definition 2, where $\mathcal{P}_{IN}$, $\mathcal{P}_{AUX}$, $\mathcal{P}_{OUT}$ and $\mathcal{P}_{CDN}$ are pairwise disjoint subsets of $\mathcal{P}$. Then $|\mathcal{P}| = |\mathcal{P}_{IN}| + |\mathcal{P}_{AUX}| + |\mathcal{P}_{OUT}| + |\mathcal{P}_{CDN}| = 2^{n-1} + 2^{n-1}(k-1) + 2^n + 2^{n-1}k = 2^n(k+1) = N(k+1)$. Thus, $|P_{CP}| = N(k+1)$.

Algorithm described in Sect. 3.2.2 unfolds each $p \in \mathcal{P}_{IN} \cup \mathcal{P}_{AUX} \cup \mathcal{P}_{OUT}$ into $|\text{INPUT}| = N$ ordinary places. According to Definition 2, on the other hand, there are $\frac{N(k+1)}{2}$ places of color type INPUT, meaning that P_{Unf} contains $\frac{N^2(k+1)}{2}$ places unfolded from places in $\mathcal{P}_{IN} \cup \mathcal{P}_{AUX} \cup \mathcal{P}_{OUT}$. In addition, P_{CP} has $\frac{Nk}{2}$ non-colored places. From which we conclude that $|P_{Unf}| = \frac{N^2(k+2)}{2} + \frac{Nk}{2}$. □

Proposition 1 gives a feel about the complexity of complete unfoldings created by algorithm discussed in Sect. 3.2.2 and allows us to measure the compactness of resulting P/T-nets with respect to the original CP-nets. The important conclusion is that the complete unfoldings obtained in this paper are polynomial in the size of original CP-nets (measured in terms of $\frac{|T_{Unf}|}{|T_{CP}|}$ and $\frac{|P_{Unf}|}{|P_{CP}|}$ ratios) and also demonstrate polynomial dependence on the MIN's size N.

Next propositions capture the intuition behind of acyclicity of P/T-nets unfolded from CPN-MIN$_{2^n,\,m}$.

Proposition 2 CPN-MIN$_{2^n,\,m}$ *is an acyclic net.*

Proof The proof is by contradiction. Suppose that CPN-MIN$_{2^n,\,m}$ has a directed circuit. Then there exists a token that visits a place more than once and there exists a transition that occurs more than once in (2). This contradicts the fact that no token can visit a place more than once in (2) and that the occurrence-depth of every transition is 1. □

Proposition 3 *Complete unfolding of* CPN-MIN$_{2^n,\,m}$ *is an acyclic net.*

Proof The proof is straightforward from acyclicity of CPN-MIN$_{2^n,\,m}$ and construction of unfolding discussed in Sect. 3.2.2. □

Proposition 4 *Optimized complete unfolding of* CPN-MIN$_{2^n,\,m}$ *is an acyclic net.*

Proof The proof is straightforward from the fact that removing components from acyclic net results in another net that is also acyclic. □

With this background, we are now allowed to use state equation, that has been discussed in Sect. 3.2.3, to verify the reachability property in optimized complete unfoldings constructed from CPN-MIN$_{2^n,\,m}$.

6 Experimental results

To ascertain the effectiveness of our approach we have conducted numerous computer experiments on PC/Linux/Windows XP platforms with 3 GHz of CPU frequency and 3GB RAM. The optimized complete unfoldings in our experiments were constructed with the GreatSPN tool [3], which is integrated into the CPN-AMI software package [14], and reachability analysis of optimized complete unfoldings by means of state equation (1) was performed with high-level language and interactive environment Matlab. The unfolding results were verified with the HELENA software tool [7].

We have tested the applicability of the proposed approach on two series of instances: 8×8 and 16×16 SENs. It has been shown [1] that $2 \log_2 N - 1$ stages of 2×2 switches with perfect shuffle pattern between neighboring stages is rearrangeable. This particularly means that 8×8 5-stage and 16×16 7-stage SENs are capable of generating any input/output permutation. So, we have checked the admissibility of permutations to 8×8 1 through 4-stage and 16×16 1 through 6-stage SENs.

6.1 Construction of optimized complete unfoldings

Table 1 shows the results of our experiments for construction of optimized complete unfoldings. In this table, $|P_{CP}|$, $|T_{CP}|$, $|P_{Unf}|$, $|T_{Unf}|$, $|P_{Opt}|$ and $|T_{Opt}|$ respectively indicate the number of places and transitions in related CP-nets, complete unfoldings and optimized complete unfoldings. We use $\frac{|P_{Opt}|}{|P_{CP}|}$ and $\frac{|T_{Opt}|}{|T_{CP}|}$ to measure the compactness of the optimized complete unfoldings relative to the original CP-nets. Time needed to create unfolded P/T-nets is expressed in seconds.

It must be noticed that our simulation results for T_{CP}, P_{CP}, T_{Unf} and P_{Unf} were consistent with the analytical ones claimed in Proposition 1.

The comparison of experimental results within the series shows that number of places and number of transitions in complete unfoldings increase linearly with increase of the number of stages. More precisely, for 8×8 k-stage SEN ($1 \leqslant k \leqslant 4$) the associated P/T-net has $100 + 36 \cdot (k - 1)$ places and $256 \cdot k$ transitions. We have made similar observations for complete unfoldings of 16×16 k-stage SENs, which contain $392 + 136 \cdot (k - 1)$ places and $2048 \cdot k$ transitions ($1 \leqslant k \leqslant 6$).

Table 1 Experimental results for optimized complete unfoldings

MIN	CP-net		Unfolded net			Optimized net		Rates	
	$\|P_{CP}\|$	$\|T_{CP}\|$	$\|P_{Unf}\|$	$\|T_{Unf}\|$	Time	$\|P_{Opt}\|$	$\|T_{Opt}\|$	$\frac{\|P_{Opt}\|}{\|P_{CP}\|}$	$\frac{\|T_{Opt}\|}{\|T_{CP}\|}$
8×8 1-Stage SEN	16	4	100	256	1.6	28	8	1.75	2.00
8×8 2-Stage SEN	24	8	136	512	9	64	80	2.67	10.00
8×8 3-Stage SEN	32	12	172	768	25	132	304	4.13	25.30
8×8 4-Stage SEN	40	16	208	1024	55	136	592	3.40	37.00
16×16 1-Stage SEN	32	8	392	2048	17	56	16	1.75	2.00
16×16 2-Stage SEN	48	16	528	4096	126	128	160	2.67	10.00
16×16 3-Stage SEN	64	24	664	6144	110	264	608	4.13	25.30
16×16 4-Stage SEN	80	32	800	8192	363	528	1794	6.60	56.06
16×16 5-Stage SEN	96	40	936	10240	1190	536	2840	5.58	71.00
16×16 6-Stage SEN	112	48	1072	12288	1792	544	3916	4.87	81.58

Table 2 Permutations tested for admissibility

8×8 permutations	16×16 permutations
$\pi_1 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7 \\ 3\,0\,5\,6\,2\,1\,4\,7 \end{pmatrix}$	$\pi_5 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \\ 0\,2\,6\,4\,9\,11\,15\,12\,1\,3\,5\,7\,10\,8\,13\,14 \end{pmatrix}$
$\pi_2 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7 \\ 0\,1\,3\,2\,5\,4\,6\,7 \end{pmatrix}$	$\pi_6 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \\ 14\,3\,9\,10\,11\,15\,1\,2\,13\,0\,7\,8\,5\,6\,12\,4 \end{pmatrix}$
$\pi_3 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7 \\ 7\,0\,3\,5\,4\,2\,1\,6 \end{pmatrix}$	$\pi_7 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \\ 0\,2\,4\,6\,8\,10\,12\,14\,1\,3\,5\,7\,9\,11\,13\,15 \end{pmatrix}$
$\pi_4 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7 \\ 0\,1\,6\,3\,7\,5\,2\,4 \end{pmatrix}$	$\pi_8 = \begin{pmatrix} 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \\ 0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15 \end{pmatrix}$

Increase of the MIN size, however, has lead to more rapid growth of number of components in related P/T-nets, e.g., for 16×16 k-stage SEN ($1 \leqslant k \leqslant 4$) corresponding P/T-net contains exactly 8 times more transitions and 3.84 to 3.92 times more places than that of 8×8 k-stage SEN does. But this has not affected the unfolding performance much, in the sense that we did not experience any illegal termination of unfolding procedure due to unacceptable performance. In fact, the number of components in constructed complete unfoldings by orders of magnitude much more smaller than the upper bound for the number of nodes that implemented software tools can handle, e.g., according to system specifications GreatSPN tool is capable of creating an unfolding with 10^{25} nodes.

The optimization was an essential part of our experiments as we wanted to build the state equations on more compact nets. As we expected, the optimized complete unfoldings were inherently more compact than corresponding complete unfoldings due to the elimination of unnecessary components such as dead transitions, 0-bounded places, etc. The figures in the last two columns of Table 1 illustrate the rates of growth of optimized complete unfoldings with respect to original CP-nets. For the number of places the rate of growth was in the order of N being less than N for all instances, while this ratio for number of transitions was bounded by N^2.

Finally we turn to run times, as seen in column 6 of Table 1. For the most time-consuming instance of 16×16 6-stage SEN it took less than 30 min to create the components of associated optimized complete unfolding.

6.2 Verification of reachability by means of invariants

The data transfers between the GreatSPN tool and the Matlab were implemented using interface program designed by means of C program code. For each instance in Table 1, the GreatSPN tool returned an output file containing related optimized complete unfolding stored in rather scrambled manner. The interface program extracted the entries of corresponding incidence matrix and initial marking from an output file and stored in separate text files. Finally, the destination markings were created in accordance with the randomly chosen permutations (see Table 2), that we tested for admissibility to SENs. For each instance and each permutation, the interface program returned three text files; one for the incidence matrix, one for the initial marking, and one for the destination marking.

We used the Gaussian elimination method, which is integrated into the Matlab standard functions library, to solve the system of linear algebraic equations. It should be noticed we did not attempt a sparse or matrix-free implementation since the time taken by the Matlab

Table 3 Experimental results on permutation admissibility

Instances	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8
8×8 1-stage SEN	×	✓	×	×				
8×8 2-stage SEN	✓	×	×	×				
8×8 3-stage SEN	×	×	✓	×				
8×8 4-stage SEN	×	×	×	✓				
16×16 1-stage SEN					×	×	×	✓
16×16 2-stage SEN					✓	×	✓	×
16×16 3-stage SEN					×	×	×	×
16×16 4-stage SEN					×	×	×	×
16×16 5-stage SEN					×	×	×	✓
16×16 6-stage SEN					✓	✓	✓	×

solver was negligible small, e.g., for the most time-consuming case of 16×16 6-stage SEN it took a little over 2 s to complete the task.

The results of computer experiments for admissibility of permutations are summarized in Table 3. In this table, a check mark indicates that the permutation is admissible to the MIN. We verified the consistency of these results with known analytical methods as far as this was possible. For instance, due to unique-path structure of 8×8 1- through 3-stage and 16×16 1- through 4-stage SENs, we were capable to use the balanced matrices characterization to check the admissibility of aforementioned permutations. For all such cases the results of computer experiments were confirmed by the analytical results.

7 Conclusions and further work

This paper exploits the relationship between interconnection networks and Petri nets to the benefit of both fields. We are not aware of any work that explores the potential of modern Petri net analysis methods for investigation of interconnection networks, in the depth given here. To the best of authors' knowledge the previous work [2], where we explored the permutation capability characterization of MIN's through state space analysis of CP-nets, was the first attempt to adopt CP-nets for analysis of interconnection networks. The present work assesses the applicability of the unfolding technique and the invariants method to interconnection networks. The results of computer experiments demonstrate the feasibility of our approach.

The main outcomes of this work are summarized below:

- It is easy to design a CP-net model of interconnection network.
- Complete unfoldings obtained before and after the optimization step are polynomial in the MIN's size, which allows us to avoid a memory overflow.
- Optimized complete unfoldings obtained in this work employ acyclic structure, which enables us to implement the invariants method.
- The solution of permutation admissibility problem in terms of CP-nets can be handled with existing powerful software tools.

We believe that a more extensive study of the relationship between interconnection networks and Petri net analysis methods can be fruitful for investigation of fault-tolerance and reliability characterizations of interconnection networks.

References

1. Bashirov, R.: Rearrangeability of $2\log -1$ stage networks employing an uniform connection pattern. Calcolo **38**, 85–95 (2000)
2. Bashirov, R., Crespi, V.: Analyzing the permutation capability of multistage interconnection networks with colored Petri nets. Inform. Sci. **176**, 3143–3165 (2006)
3. Chiola, G., Franceschinis, G., Gaeta, R., Ribaudo, M.: GreatSPN 1.7—graphical editor and analyzer for timed and stochastic Petri nets. Perform. Eval. **24**, 47–68 (1995)
4. Das, R.K., Das, N.: GSE—a generalized full-access multistage interconnection network with minimum cost. In: Proceedings of HiPC'96, Trivandrum, India, IEEE Computer Society, pp. 176–181 (1996)
5. Das, N., Bezrukov, S.L.: Permutation admissibility in shuffle-exchange networks with arbitrary number of stages. In: Proceedings of HiPC'98, Madras, India, IEEE Computer Society, pp. 270–276 (1998)
6. Das, N., Bhattacharya, B.B., Bezrukov, S., Menon, R., Sarkar, A.: Permutation routing in optical MINs with minimum number of stages. J. Syst. Archit. **48**, 311–323 (2003)
7. Evangelista, S.: High level Petri nets analysis with Helena. In: Proceedings of ATPN'05. Miami, FL, LNCS 3536, pp. 455–464 (2005)
8. Grammatikakis, M.D., Hsu, D.F., Sibeyn, J.F.: Packet routing in fixed-connection networks: a survey. J. Paral. Distrib. Comput. **54**, 77–132 (1998)
9. Hsia, C.J.A., Chen, C.Y.R.: Permutation capability of multistage interconnection networks. In: Proceedings of ICPP'90, Urbana-Champaign, IL, pp. 1338–1346 (1990)
10. Hu, Q., Shen, X., Liang, W.: Optimally routing LC permutations on k-extra-stage cube-type networks. IEEE Trans. Comput. **45**, 97–103 (1996)
11. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, vol. 1. Springer, Berlin (1997)
12. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, vol. 2. Springer, Berlin (1997)
13. Khomenko, V., Kondratyev, A., Koutny, A., Vogler, W.: Merged processes: a new condenced representation of Petri net behaviour. Acta Inf. **43**, 307–330 (2006)
14. Hamez, A., Hillah, L., Kordon, F., Linard, A., Paviot-Adet, E., Renault, X., Thierry-Mieg, Y.: New features in CPN-AMI 3: focusing on the analysis of complex distributed systems. In: Proceeding of ACSD'06, June, Turku, Finland, IEEE Computer Society, pp. 273–275 (2006)
15. Kordon, F., Linard, A., Paviot-Adet, E.: Optimized colored nets unfolding. In: Proceedings FORTE'06, LNCS 4229, pp. 339–355 (2006)
16. Lenfant, J.: Parallel permutations of data: a Benes network control algorithm for frequently used permutations. IEEE Trans. Comput. **27**, 637–647 (1978)
17. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**, 541–580 (1989)
18. Raghavendra, C.S., Boppana, R.V.: On self-routing in Beneš and shuffle-exchange networks. IEEE Trans. Comput. **40**, 1057–1064 (1991)
19. Sahni, S.: Matrix multiplication and data routing using a partitioned optical passive stars network. IEEE Trans. Paral. Distrib. Comput. **11**, 720–728 (2000)
20. Shen, X., Xu, M., Wang, X.: An optimal algorithm for permutation admissibility to multistage interconnection networks. IEEE Trans. Comput. **44**, 604–608 (1995)
21. Shen, X.: An optimal $O(N \log N)$ algorithm for permutation admissibility to extra-stage cube-type networks. IEEE Trans. Comput. **44**, 1144–1149 (1995)
22. Shen, X., Yang, F., Pan, Y.: Equivalent permutation capabilities between time-division optical omega networks and non-optical extra-stage omega networks. IEEE Trans. Network. **9**, 518–524 (2001)
23. Veselovsky, G., Batovski, D.A.: A study of the permutation capability of a binary hypercube under deterministic dimension-order routing. In: Proceedings of PDPTA'03, Las Vegas, NV, SCREA Press, pp. 173–177 (2003)
24. Wu, C., Feng, T.: On a class of multistage interconnection networks. IEEE Trans. Comput. **29**, 694–702 (1980)