IEEE Transactions On Robotics and Automation, vol. 17, no. 5, pp.562-575, 2001.
Copyright © 2001 IEEE (DOI 10.1109/70.964658).

# Timed Petri Nets in Modeling and Analysis of Cluster Tools

Wlodek M. Zuberek
medskip
Department of Computer Science, Memorial University
St.John's, NL, Canada A1B 3X5

*Abstract*— **Timed Petri nets are used as models of cluster tools representing the flow of wafers through the chambers of the tool as well as sequences of actions performed by the robotic transporter. Since the durations of all activities are also represented in the model, performance characteristics can be derived from the model for steady–state as well as transient behaviors. The performance of single–blade tools is compared with that of dual–blade tools. The effects of multiple loadlocks, redundant chambers and multiple robots are discussed and analyzed. Modeling of wafer routings with chamber revisiting and processing of wafers of multiple types is also briefly discussed.**

*Keywords*— **Cluster tools, performance analysis, steady–state behavior, transient behavior, timed Petri nets, structural analysis, place invariants, net transformations.**

## I. Introduction

A CLUSTER tool is an integrated, environmentally isolated manufacturing system consisting of process, transport, and cassette modules, mechanically linked together [3]. The factors which stimulate an increased use of clustered tools in recent years include improved yield and throughput, reduced contamination, better utilization of the floor space, and reduced human intervention [17].

Because of high throughput requirements, cluster tools perform a number of activities concurrently, for example, different wafers are processed in different chambers at the same time, and also the robotic transporter can be moving to a position required by the next step. Petri nets [15], [10] are formal models developed specifically for representation of concurrent activities and for their coordination, i.e., for ordering specific actions or for performing actions simultaneously by more than one component of a system. Petri nets are sometimes called "condition–event systems" because their two types of basic elements, called places and transitions, represent the (satisfied or unsatisfied) conditions of some events, and the events which can occur only when all conditions associated with them are satisfied. Formally, Petri nets are represented by bipartite graphs (i.e., graphs with two types of vertices, one representing places, and the other transitions), and directed arcs connecting these two types of vertices. The dynamic behavior of nets is represented by the so called tokens associated with places. The distribution of these tokens can change (as an occurrence of some events), representing the behavior of the modeled systems.

In order to analyze the performance of modeled systems, the durations of all activities must also be taken ito account. Several types of nets "with time" have been proposed by associating "time delays" with places [16], or occurrence durations with transitions [1], [13], [22] of net models. Also, the introduced temporal properties can be deterministic [13], [14], [16], [22], or can be random variables described by probability distribution functions (the negative exponential distribution being probably the most popular choice) [1], [2], [22].

Analysis of timed net models based on their behavior (represented by the set of states and transitions between states) is known as reachability analysis. For complex models, the exhaustive reachability analysis can easily become difficult because of the very large number of states (for some models the number of states increases exponentially with the size of the model, which is known as the "state explosion problem"). Several approaches can be used to deal with the excessive numbers of states. One approach reduces the number of states by using state aggregation (i.e., by combining groups of states into single 'superstates'); another uses symmetries of the state space. For some classes of net models, the performance properties can be derived from the structure of the net models; this approach is known as structural analysis. The most popular example of this approach is analysis based on place–invariants (or P–invariants) for models covered by families of simple cyclic subnets (which are implied by P–invariants).

Traditionally, performance of cluster tools was analyzed by using timing diagrams representing typical sequences of events, and deriving performance formulas from a critical path that determined the cyclic behavior of a tool [11], [12], [20]; such an approach is highly dependent on the analyzed cluster tool and its properties, and becomes quite complicated for tools which are complex. This paper presents an approach based on timed Petri nets which can be used for modeling and evaluation of a large variety of cluster tools, including single–blade and dual–blade ones, tools with multiple loadlocks, redundant chambers and multiple robots. Several such models are discussed in greater detail, and their steady–state as well as transient behaviors are analyzed. The performance of the models is obtained by structural analysis (for the steady–state), as an alternative approach to the one presented in [18], where the performance of Petri net models is obtained by reachability analysis, i.e., by the exhaustive generation and analysis of the state space that needs to be repeated for each change of any one of modeling parameters. Net models presented in this paper are composed of simple subnets implied by place invariants, and this allows to derive the performance in symbolic form, similar to the approach proposed in [13], [16]. There is, however, a significant difference between the approach proposed in [13] and the one used in this paper; [13] uses the sets of all possible circuits in nets which must be "consistent". The number of such circuits, for many nets, grows exponentially with the size of the model. The approach described in this paper is based on basic place invariants which represent only some of the circuits. Moreover, the number of place invariants can be further reduced by simple net transformations [23] which eliminate all those net elements which are insignificant for performance evaluation. In effect, the number of significant place invariants is a linear function of the model size. In addition, the approach presented in [13] is valid for basic place/transition nets only, and does not allow inhibitor arcs or arc weights, both of which are frequently used in models presented in this paper.

The approach presented in this paper is derived from earlier work on modeling and analysis of schedules for manufacturing cells [23].

Section 2 recalls basic concepts of timed Petri nets in order to avoid confusion that may arise due to a large variety of different types of Petri nets and especially timed Petri nets; in modeling with timed Petri nets, even a minor difference in the assumed behavior of net models may have a major impact on the representation of the model. Models of single–blade cluster tools are discussed and evaluated in Section 3, while Section 4 presents models of dual–blade cluster tools; it also contains a brief comparison of the performance of single–blade and dual–blade tools. Section 5 describes several extension and generalization of the models discussed in Sections 3 and 4; these extensions include cluster tools with multiple loadlocks, redundant chambers and tools with multiple robots. Performance characteristics of more general cluster tools are also included. Several concluding remarks and a number of further extensions are outlined in Section 6.

## II. Timed Petri Nets

Petri nets have been proposed as a simple and convenient formalism for modeling systems that exhibit parallel and concurrent activities [10], [15], [19]. In Petri nets, these activities are represented by the so called tokens which can move within a (static) graph–like structure of the net. More formally, a *marked* (place/transition) *inhibitor* Petri net $\mathcal{M}$ is defined as $\mathcal{M} = (\mathcal{N}, m_0)$, where the structure $\mathcal{N}$ is a bipartite directed graph, $\mathcal{N} = (P, T, A, B, w)$, with a set of places $P$, a set of transitions $T$, a set of directed arcs $A$ connecting places with transitions and transitions with places, $A \subseteq T \times P \cup P \times T$, a (possibly empty) set of inhibitor arcs which connect places with transitions, $B \subset P \times T$, arc weight function which assigns a weight (or multiplicity) to each arc of the net, $w : A \to \{1, 2, ...\}$, and an initial marking function $m_0$ which assigns nonnegative numbers of tokens to places of the net, $m_0 : P \to \{0, 1, ...\}$.

A place is *shared* if it is connected to more than one transition. A shared place $p$ is *guarded* if for each two transitions sharing it there exists another place which is connected by a directed arc to one of these sharing transitions and an inhibitor arc to the other (so only one of these two transitions can be enabled by any marking). A shared place $p$ is *free–choice* if the sets of places connected by directed arcs and inhibitor arcs for all transitions sharing $p$ are identical and the weights of the arcs are the same. An inhibitor net is free-choice if all its shared places are either guarded or free–choice. A net is (structurally or statically) conflict–free if all its shared places are guarded. A marked net is (dynamically) conflict–free if for any marking reachable from the initial marking, and for any shared place, at most one of transitions sharing this place is enabled. The models of cluster tools discussed in this paper are (statically or dynamically) conflict–free nets.

In order to study performance aspects of Petri net models, the duration of activities must also be taken into account and included into model specifications. In timed nets [22], occurrence times are associated with transitions, and transition occurrences are *real–time* events, i.e., tokens are removed from input places at the beginning of the occurrence period, and they are deposited to the output places at the end of this period (sometimes this is also called a *three–phase* firing mechanism as opposed to *one–phase* instantaneous occurrences of transitions in stochastic nets [1], [2] and time nets [6], [9]). All occurrences of enabled transitions are initiated in the same instants of time in which the transitions become enabled (although some enabled transitions cannot initiate their occurrences). If, during the occurrence period of a transition, the transition becomes enabled again, a new, independent occurrence can be initiated, which will overlap with the other occurrence(s). There is no limit on the number of simultaneous occurrences of the same transition (sometimes this is called *infinite occurrence semantics*). Similarly, if a transition is enabled "several times" (i.e., it remains enabled after initiating an occurrence), it may start several independent occurrences in the same time instant.

More formally, a *conflict–free timed* Petri net is a pair, $\mathcal{T} = (\mathcal{M}, f)$, where $\mathcal{M}$ is a marked net and $f$ is a *timing function* which assigns a (constant or randomly distributed) occurrence time to each transition of the net, $f : T \to \mathbf{R}^+$, where $\mathbf{R}^+$ is the set of nonnegative real numbers.

The occurrence times of transitions can be either deterministic or stochastic (i.e., described by some probability distribution function); in the first case, the corresponding timed nets are referred to as D–timed nets, in the second, for the (negative) exponential distribution of firing times, the nets are called M–timed nets (Markovian nets). In both cases, the concepts of state and state transitions have been formally defined and used in the derivation of different performance characteristics of the model [22]. Only D–timed Petri nets are used in this paper.

In timed nets, the occurrence times of some transitions may be equal to zero, which means that the occurrences are instantaneous; all such transitions are called *immediate* (while the others are called *timed*). Since the immediate transitions have no tangible effects on the (timed) behavior of the model, it is convenient to 'split' the set of transitions into two parts, the set of immediate and the set of timed transitions, and to first perform all occurrences of the (enabled) immediate transitions, and then (still in the same time instant), when no more immediate transitions are enabled, to start the occurrences of (enabled) timed transitions. It should be noted that such a convention effectively introduces the priority of immediate transitions over the timed ones, so the conflicts of immediate and timed transitions are not allowed in timed nets. Detailed characterization of the behavior of timed nets with immediate and timed transitions is given in [22].

Each place/transition net $\mathcal{N} = (P, T, A, B, w)$ can conveniently be represented by a *connectivity* (or *incidence*) matrix $\mathbf{C} : P \times T \rightarrow \mathbf{Z}$ ($\mathbf{Z}$ denotes the set of integer numbers) in which places correspond to rows, transitions to columns, and the entries are defined as:

$$\forall\, p \in P\, \forall\, t \in T \;:\; \mathbf{C}[p,t] = \begin{cases} -w(p,t), & \\ \qquad \text{if } (p,t) \in A \wedge (t,p) \notin A, & \\ +w(t,p), & \\ \qquad \text{if } (t,p) \in A \wedge (p,t) \notin A, & \\ w(t,p) - w(p,t), & \\ \qquad \text{if } (t,p) \in A \wedge (p,t) \in A, & \\ 0, \quad \text{otherwise.} & \end{cases}$$

Connectivity matrices disregard inhibitor arcs and ‘selfloops’, that is, pairs of arcs $(p,t)$ and $(t,p)$ with the same weights $w$. A pure net is defined as a net without selfloops [15].

A *P–invariant* (place invariant, sometimes also called S–invariant) of a net $\mathcal{N}$ is any nonnegative, nonzero integer (column) vector $I$ which is a solution of the matrix equation

$$\mathbf{C}^T \times I = 0,$$

where $\mathbf{C}^T$ denotes the transpose of matrix $\mathbf{C}$. It follows immediately from this definition that if $I_1$ and $I_2$ are P–invariants of $\mathcal{N}$, then also any linear (positive) combination of $I_1$ and $I_2$ is also a P–invariant of $\mathcal{N}$. A basic P–invariant of a net is defined as a P–invariant which does not contain simpler invariants.

Similarly, a *T–invariant* (transition invariant) of a net $\mathcal{N}$ is any nonnegative, nonzero integer (column) vector $J$ which is a solution of the matrix equation

$$\mathbf{C} \times J = 0,$$

and a basic T–invariant of a net is defined as a T–invariant which does not contain simpler invariants.

Moreover, a net $\mathcal{N}_i = (P_i, T_i, A_i, B_i, w)$ is a $P_i$-implied subnet of a net $\mathcal{N} = (P, T, A, B, w)$, $P_i \subset P$, if:

(1) $A_i \;=\; A \cap (P_i \times T \cup T \times P_i)$;
(2) $T_i \;=\; \{t \in T \mid \exists\, p \in P_i \;:\; (p,t) \in A_i \vee (t,p) \in A_i\}$;
(3) $B_i \;=\; B \cap P_i \times T_i$.

It should be observed that in a pure net $\mathcal{N}$, each P–invariant $I$ of a net $\mathcal{N}$ determines a $P_I$-implied (invariant) subnet of $\mathcal{N}$, where $P_I = \{p \in P \mid I(p) > 0\}$; $P_I$ is sometimes called the support of the invariant $I$. All nonzero elements of $I$ select rows of $\mathbf{C}$, and each selected row $i$ corresponds to a place $p_i$ with all its input and all output arcs associated with it.

Finding basic invariants is a ‘classical’ problem of linear algebra, and there are known algorithms to solve this problem efficiently [7], [8].

Net invariants can be very useful in performance evaluation of net models; if a net is covered by a family of conflict–free cyclic subnets, the cycle time of the net, $\tau_0$, is equal to the maximum cycle time of the covering subnets [13], [16]:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, ..., \tau_k)$$

where $k$ is the number of subnets covering the original net, and each $\tau_i$, $i = 1, ..., k$, is the cycle time of the subnet $i$, which is equal to the sum of occurrence times associated with the transitions, divided by the total number of tokens assigned to the subnet:

$$\tau_i \;=\; \frac{\sum_{t \in T_i} f(t)}{\sum_{p \in P_i} m(p)}.$$

In many cases, the number of basic P–invariants can be reduced by removing from the analyzed net all these elements which do not affect the performance of models [23]. Fig.1 shows one of such transformations which reduces parallel paths that have no influence on the behavior of a timed net, but which can increase the number of (basic) P–invariants.

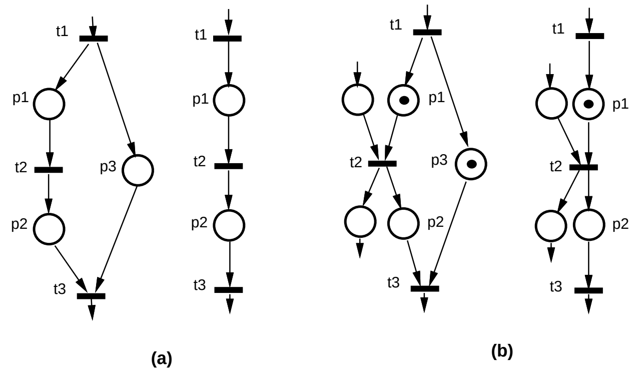

Fig. 1. Parallel path reduction.

In Fig.1, part (a) is the simple case of parallel paths, while part (b) shows a more intricate case, which still can be simplified without affecting the performance of the model (in fact, the state space in both cases is not affected by the transformation). It should be noticed that parallel paths can be either all unmarked, as in Fig.1(a), or all marked, as in Fig.1(b), but they cannot be “mixed”, i.e., one path marked and the other unmarked.

## III. Models of Single–Blade Cluster Tools

The cluster tools analyzed in this section are $m$–chamber cluster tools with one robotic transporter. Each of the chambers performs a unique process, and there is a single chamber for each process. The only explicit storage facility is the loadlock. For single–blade tools, the robotic transporter can carry only one wafer at a time. The model assumes that all wafers have the same process sequence, and that no chambers are revisited, as in [12].

A sketch of a 3–chamber cluster tool is shown in Fig.2, where LL denotes the loadlock to store cassettes of wafers; C1, C2 and C3 are process chambers which modify the properties of the wafers, and R is a robotic transporter (or simply a robot) which moves the wafers between the loadlock and the chambers as well as from one chamber to another.

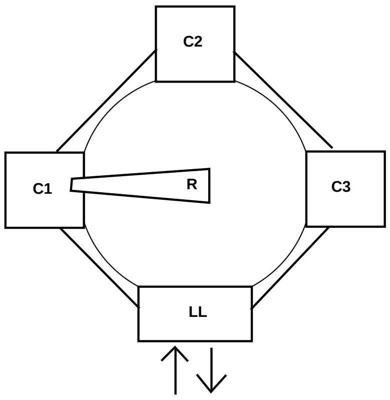

Fig. 2. An outline of a 3–chamber cluster tool.

When a batch of wafers arrives at an empty cluster tool, it is placed in the loadlock which is then typically pumped down to vacuum. All the time required to get a batch into the cluster and ready for processing is denoted as $\tau_{load}$. The robot, assumed to be idle at the loadlock, moves the first wafer to the first chamber. For simplicity, it is assumed that the chambers are numbered as they appear in the process sequence. When the process in the first chamber is finished, the wafer is moved to the second chamber, after which the second wafer can be moved into the first chamber. After a number of such wafer transports, the first wafer arrives back at the loadlock. When all wafers have been processed and returned to the loadlock, the loadlock is raised to atmospheric pressure and the batch is removed. The time interval between when the last wafer arrives at the loadlock and when the batch is removed is denoted as $\tau_{unload}$.

In general, the time to process a batch consists of the following [12]: $\tau_{load}$, the time $\tau_{init}$ to reach steady state, the time spent in steady state $\tau_{steady}$, the time $\tau_{end}$ to process final wafers, and $\tau_{unload}$.

## *A. Steady–State Behavior*

The model for the steady–state behavior is the simplest one, so it is discussed first. Several elements of the cluster tool can be ignored for steady state considerations (e.g., the loadlock). Moreover, it is assumed that all chambers are used concurrently, i.e., when the $i$-th wafer is moved to chamber 1, the $(i-1)$-th wafer is processed in chamber 2, and $(i-2)$-th wafer is processed in chamber 3. The sequence of the operations in each cycle is as follows (it is assumed that the cycle begins when a new wafer is moved to chamber 1):

- pick next wafer from loadlock, transport it to chamber 1 and load it; chamber 1 can start its process;
- move to chamber 3, unload the wafer (when ready), transport it to loadlock and drop it there;
- move to chamber 2, unload the wafer (when ready), transport it to chamber 3 and load it; chamber 3 can start its process;
- move to chamber 1, unload the wafer (when ready), transport it to chamber 2 and load it; chamber 2 can start its process;
- return to loadlock to begin another cycle.

A Petri net model of this sequence of operations is shown in Fig.3. The model contains three sections modeling the three chambers, each represented by one transition ($t_1$, $t_2$ and $t_3$, respectively). Each of these transitions has one input and one output place (representing the conditions "wafer loaded in chamber" and "wafer ready for unloading"). The remaining part of the model represents the sequence of steps corresponding to one complete cycle of the robot. This sequence begins (as indicated by the initial marking) by picking a wafer from the loadlock (transition $t_{01}$). The operations represented by transitions are shown in Tab.1.

TABLE I
Operations Represented by Transitions in Fig.3.

| *transition* | *operation* |
|---|---|
| $t_{01}$ | pick next wafer from the loadlock, move it to chamber 1 and load; |
| $t_{13}$ | move the robot to chamber 3; |
| $t_{34}$ | unload the wafer from chamber 3, move it to loadlock and drop: |
| $t_{42}$ | move the robot to chamber 2; |
| $t_{23}$ | unload the wafer from chamber 2, move it to chamber 3 and load; |
| $t_{31}$ | move the robot to chamber 1; |
| $t_{12}$ | unload the wafer from chamber 1, move it to chamber 2 and load; |
| $t_{20}$ | move the robot to loadlock. |

In order to obtain the effect of steady–state, place $p_{40}$ is used as "input" and "output" of the cluster tool. When a wafer is finished, a token is deposited in $p_{40}$, and the same token is used as the next wafer a moment later. The initial marking of $p_{40}$ is irrelevant (as long as it is nonzero), and the behavior is exactly the same if more than one token is assigned initially to $p_{40}$. Moreover, it can be observed that $p_{40}$ creates a parallel path between $t_{01}$ and $t_{34}$, so it has no effect on the performance of the model, and can be removed (with the two arcs connected to it).

All transitions are timed transitions, and the occurrence times associated with them represent the times of the cor-

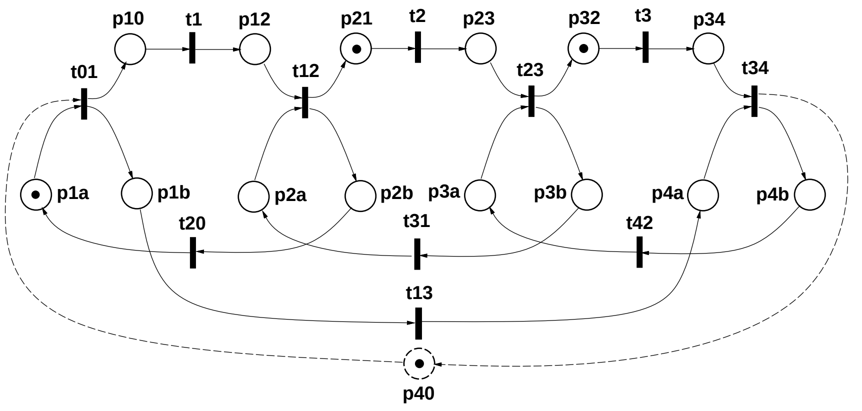

Fig. 3. Petri net model for the steady-state behavior of a single–blade 3–chamber too tool.

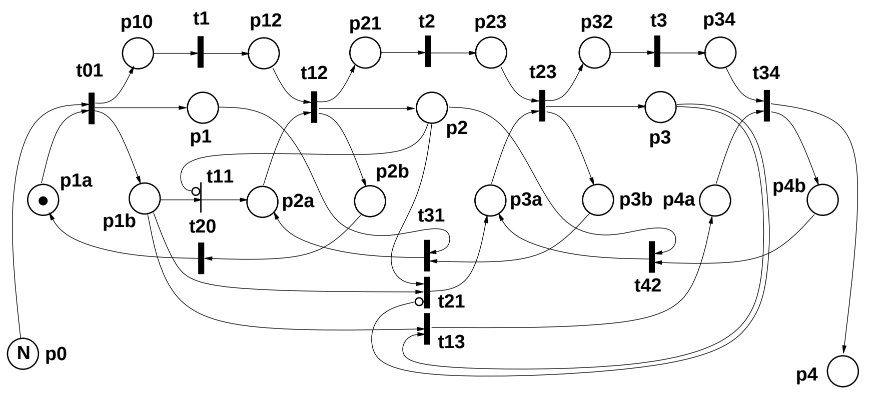

Fig. 4. Petri net model for the initial transient behavior of a single–blade 3–chamber tool.

responding operations.

The net shown in Fig.3, after removal of place $p_{40}$, has four P–invariants; the sets of transitions of subnets implied by these P–invariants are:

| *invariant* | *set of transitions* |
|---|---|
| 1 | $t_1, t_{01}, t_{12}, t_{20}$ |
| 2 | $t_2, t_{12}, t_{23}, t_{31}$ |
| 3 | $t_3, t_{23}, t_{34}, t_{42}$ |
| 4 | $t_{01}, t_{12}, t_{23}, t_{34}, t_{13}, t_{20}, t_{31}, t_{42}$ |

Because the cycle time of the model is equal to the maximum cycle time of subnets implied by P–invariants, the cycle time $\tau_0$ is:

$$\tau_0 \ = \ \max(\tau_1, \tau_2, \tau_3, \tau_4)$$

where $\tau_i$ denotes the cycle time of the subnet $i$, so, $\tau_1 = f(t_1)+f(t_{01})+f(t_{12})+f(t_{20})$, $\tau_2 = f(t_2)+f(t_{12})+f(t_{23})+f(t_{31})$, and so on.

If $\tau_0$ is equal to one (or more) of the first three terms, the model is called "process bound" because the duration of the process performed by the corresponding chamber determines the cycle time (and the throughput) of the tool; if the cycle time is equal to the last term, the model is called "transport bound" [20].

## *B. Initial Transient Behavior*

The initial transient behavior is due to the fact that the chambers, at the beginning of each batch, are empty. Consequently, the sequence of operations is slightly different at the beginning of the batch than in the steady state, and this difference must be captured by the model. Fig.4 shows the model representing the initial transient behavior (as an extension of the steady–state model).

The initial sequence of operations is described by the steps (and the corresponding transitions) shown in Tab.2.

The duration of this transient behavior can be estimated from the sequence of operations, as has been done in [12], but it can also be evaluated from a slightly modified Petri net model. It can be shown [22] that the graph of reachable states for any conflict–free bounded net can only be a straight path (if the behavior is finite) or a path with a cycle (if the behavior is infinite); the cycle in this case represents the steady state behavior of the model. So, the model shown in Fig.4 needs to be slightly modified to create its cyclic, infinite behavior representing the steady–state. This can be done by merging places $p_0$ and $p_4$. Then, however, similarly as for the net shown in Fig.3, the merged place can be deleted (together with its arcs) as it has no effect on the performance of the model. Such modified model can then be analyzed for the initial transient behavior that determines the time $\tau_{init}$.

TABLE II
Operations Represented by Transitions in Fig.4.

| *transition* | *operation* |
|---|---|
| $t_{01}$ | pick a wafer from the loadlock, move it to chamber 1 and load; |
| $t_{11}$ | since chamber 2 is empty, wait for the end of chamber 1 operation; |
| $t_{12}$ | unload the wafer from chamber 1, move it to chamber 2 and load; |
| $t_{20}$ | move the robot to loadlock; |
| $t_{01}$ | pick a wafer from the loadlock, move it to chamber 1 and load; |
| $t_{22}$ | since chamber 3 is empty, move the robot to chamber 2; |
| $t_{23}$ | unload the wafer from chamber 2, move it to chamber 3 and load; |
| $t_{21}$ | move the robot to chamber 1; |
| $t_{12}$ | unload the wafer from chamber 1, move it to chamber 2 and load; |
| $t_{20}$ | move the robot to loadlock. |

For example, assuming (just for the sake of this example) that $f(t_1) = f(t_2) = f(t_3) = 10$, and all other timed transitions have the associated time equal to 1, the state graph corresponding to the (modified) net of Fig.4 is shown in Fig.5, where the time spent in each state is given in parentheses. State 12 is the first state after the initial transient behavior, so $\tau_{init} = 26$ (the sum of times associated with states 1 to 11), while the cycle time $\tau_0 = 13$ (the sum of times associated with the cyclic states). The steady–state solution (discussed earlier), for the same timing values, results in $\tau_1 = \tau_2 = \tau_3 = 13$ and $\tau_4 = 8$, so $\tau_0 = 13$, and the model is process bound.

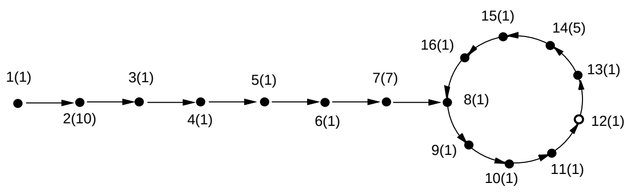

Fig. 5. State graph for the modified net of Fig.4.

## C. Final Transient Behavior

The final part of processing a batch of wafers is also different from the steady–state because there are no more wafers in the loadlock, but still the wafers remaining in the chambers must complete their operations and be transported back to the loadlock. The Petri net model for this part of processing is shown in Fig.6.

The sequence of required operations begins with the last wafer picked from the loadlock; the operations are described in Tab.3.

The duration of the final transient behavior can be estimated on the basis of required operations, or it can be obtained from analysis of the net model (Fig.6). In this case no modification is needed; the behavior of the model in Fig.6 is finite, so it directly determines the time $\tau_{end}$. For the timing information used in Section 3.3, the final transient behavior is $\tau_{end} = 34$.

TABLE III
Operations Represented by Transitions in Fig.6.

| *transition* | *operation* |
|---|---|
| $t_{01}$ | pick a wafer from the loadlock, move it to chamber 1 and load; |
| $t_{13}$ | move the robot to chamber 3; |
| $t_{34}$ | unload the wafer from chamber 3, move it to loadlock and load; |
| $t_{42}$ | move the robot to chamber 2; |
| $t_{23}$ | unload the wafer from chamber 2, move it to chamber 3 and load; |
| $t_{21}$ | move the robot to chamber 1; |
| $t_{12}$ | unload the wafer from chamber 1, move it to chamber 2 and load; |
| $t_{13}$ | move the robot to chamber 3 (loadlock is empty); |
| $t_{34}$ | unload the wafer from chamber 3, move it to loadlock and drop; |
| $t_{42}$ | move the robot to chamber 2; |
| $t_{23}$ | unload the wafer from chamber 2, move it to chamber 3 and load; |
| $t_{33}$ | since chamber 1 is empty, wait for the end of chamber 3 operation; |
| $t_{34}$ | unload the wafer from chamber 3, move it to loadlock and drop; |
| $t_{44}$ | chamber 2 is empty, end of processing. |

## D. Complete Model

The complete Petri net model of a 3–chamber cluster tool, shown in Fig.7, is obtained by merging models in Fig.4 and Fig.6 (the model shown in Fig.3 is included in both Fig.4 and Fig.6).

The total time of processing a batch of wafers in a single–blade cluster tool, $\tau_{batch}$, is thus equal to:

$$\tau_{batch} = \tau_{load} + \tau_{init} + \tau_{steady} + \tau_{end} + \tau_{unload}$$

where $\tau_{steady} = (N - m - 1) * \tau_0$, $m$ is the number of chambers, and $N$ is the number of wafers in a batch; this formula takes into account that the initial transient behavior requires $m$ wafers from the batch to initialize the $m$ chambers, and that the final transient behavior begins when the last wafer is picked from the loadlock.

## IV. Models of Dual–Blade Cluster Tools

The main difference between dual–blade cluster tools and single–blade tools is that the robotic transporter, for dual–blade tools, can carry two wafers at the same time. Consequently, the sequence of operations is different, and usually shorter, which can make a difference especially for transport bound cluster tools.

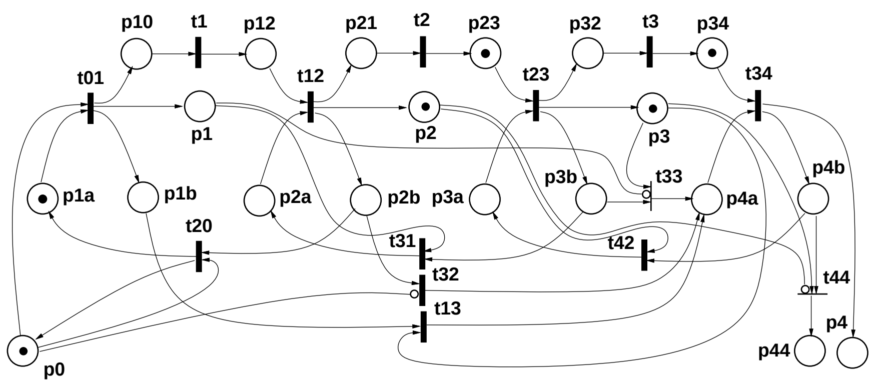

Fig. 6. Petri net model for the final transient behavior of a single–blade 3–chamber tool.

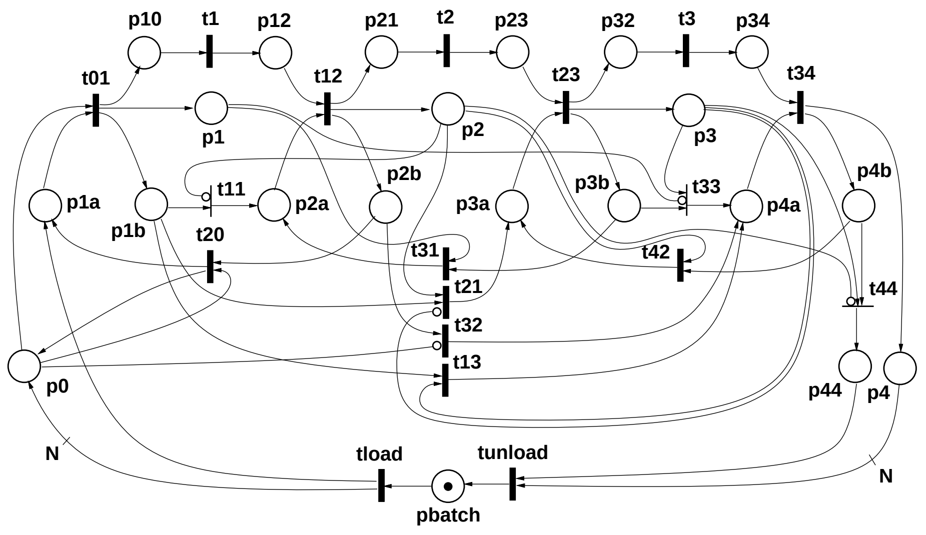

Fig. 7. Petri net model of a single–blade 3–chamber cluster tool.

## A. Steady–State Behavior

Fig.8 shows the steady-state model of a 3–chamber cluster tool with a dual–blade robot (so it corresponds to the model shown in Fig.3 for the single–blade case).

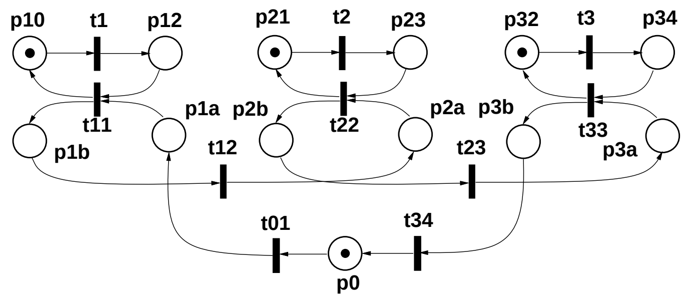

Fig. 8. Petri net model for steady–state behavior of a dual–blade 3–chamber tool.

As before, the three chambers are represented by transitions $t_1$, $t_2$ and $t_3$, each with a single input and a single output place; the remaining part of the model represents the sequence of the robot's actions, with transitions $t_{11}$, $t_{22}$ and $t_{33}$ representing sequences of "unload the wafer from a chamber, rotate, and load the wafer into a chamber", for chambers 1, 2 and 3, respectively.

The model shown in Fig.8 has four P–invariants with the following sets of transitions in invariant–implied subnets:

| *invariant* | *set of transitions* |
|---|---|
| 1 | $t_1, t_{11}$ |
| 2 | $t_2, t_{22}$ |
| 3 | $t_3, t_{33}$ |
| 4 | $t_{01}, t_{11}, t_{12}, t_{22}, t_{23}, t_{33}, t_{34}$ |

As before, the cycle time $\tau_0$ (for the steady–state behavior) is determined by the subnet with the maximum cycle time:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, \tau_3, \tau_4)$$

and the cycle times $\tau_i$ are equal to (each subnet has a single token assigned to it):

$$\begin{aligned}
\tau_1 &= f(t_1) + f(t_{11}), \\
\tau_2 &= f(t_2) + f(t_{22}), \\
\tau_3 &= f(t_3) + f(t_{33}), \\
\tau_4 &= f(t_{01}) + f(t_{11}) + f(t_{12}) + f(t_{22}) + f(t_{23}) + \\
&\qquad f(t_{33}) + f(t_{34}).
\end{aligned}$$

In order to compare this cycle time with the one for single–blade cluster tool, a more detailed analysis of the operation times is needed. The operations can be represented by collections of some elementary actions, some of which are common for single–blade and dual–blade tools (e.g., picking a wafer from a loadlock, loading a wafer into a chamber or unloading it), while some others are different (e.g., rotating the robot to get access to the other blade). Each of these actions has its execution time, and it is assumed, for simplicity, that the execution times of the same actions for different chambers are equal (it is a minor modification to make them different). The elementary actions are:

| *action* | *description* |
|---|---|
| $u$ | rotate the robot (for dual–blade case); |
| $v$ | move robot between two adjacent chambers, or between the loadlock and the first chamber, or between the last chamber and the loadlock (for simplicity all these times are assumed equal); |
| $w$ | pick a wafer from the loadlock; |
| $x$ | load a wafer into a chamber; |
| $y$ | unload a wafer from a chamber; |
| $z$ | drop a wafer in a loadlock. |

The execution time of any operation is simply equal to the sum of execution times of actions constituting the operation. For the operations represented by transitions in Fig.3 (single–blade tool) these execution times are thus as follows:

| *transition* | *execution time* |
|---|---|
| $t_{01}$ | $v+w+x$ |
| $t_{12}$ | $v+x+y$ |
| $t_{23}$ | $v+x+y$ |
| $t_{34}$ | $v+y+z$ |
| $t_{13}$ | $2v$ |
| $t_{20}$ | $2v$ |
| $t_{31}$ | $2v$ |
| $t_{42}$ | $2v$ |

The execution times of operations represented by transitions in Fig.8 (dual–blade tool) are:

| *transition* | *execution time* |
|---|---|
| $t_{01}$ | $u+v+w$ |
| $t_{11}$ | $u+x+y$ |
| $t_{12}$ | $v$ |
| $t_{22}$ | $u+x+y$ |
| $t_{23}$ | $v$ |
| $t_{33}$ | $u+x+y$ |
| $t_{34}$ | $u+v+z$ |

For transport bound single–blade and dual–blade cluster tools, the cycle times are thus:

$$\begin{aligned}\tau_4^{(1)} &= 12v+w+3x+3y+z,\\ \tau_4^{(2)} &= 5u+4v+w+3x+3y+z,\end{aligned}$$

so, the difference is $8v$ for the single–blade tool versus $5u$ for the dual–blade tool (since the other terms are equal). If the time of the operation $v$ is significantly longer than the other operations (including $u$), then the dual–blade tool offers much better performance (estimated as at least two times better than the single–blade tool in [20]). Indeed, assuming that $u = w = x = y = z = v/4$, the total times are: $\tau_4^{(1)} = 14v$ and $\tau_4^{(2)} = 7.25v$, so the cycle time of the dual–blade tool is almost one half of that of single–blade tool.

For process bound cluster tools, the differences are not so significant because the cycle times are dominated by the process times of (some) chambers, which are the same for single–blade and dual–blade tools.

## *B. Initial Transient Behavior*

The net model for the initial transient behavior is shown in Fig.9. The arc $(t_{34}, p_0)$ is an addition to convert the acyclic net into a cyclic one, so the transient behavior can be captured more easily.

For the temporal information similar to that used in Section 3.2 (i.e., for $f(t_1) = f(t_2) = f(t_3) = 10$, $f(t_{11}) = f(t_{22}) = f(t_{33}) = 2$ (in order to make the behavior comparable to that in Section 3.2), and $f(t_i) = 1$ for all other timed transitions), the transient behavior $\tau_{init}$ is equal to 44, while the cycle time $\tau_0$ is equal to 12.

## *C. Final Transient Behavior*

The net model for the final transient behavior of a dual–blade cluster tool is shown in Fig.10. As before, it is an extension of the model developed for steady–state behavior.

The time $\tau_{end}$, for the same temporal information as in Section 4.2, is equal to 53.

## V. Extensions

The basic configurations of cluster tools discussed in previous sections can easily be modified to create more complex structures. Three such extensions are presented in greater detail: cluster tools with multiple loadlocks, cluster tools with multiple identical chambers and cluster tools with multiple robots. Several generalizations of the presented models are also discussed briefly.

## *A. Multiple Loadlocks*

In the case when the $\tau_{load} + \tau_{unload}$ constitutes a significant component of the total time of processing a batch of wafers, a considerable improvement can be achieved by doubling the loadlock, as outlined in Fig.11.

The idea of using two loadlocks (instead of one) is to process the contents of one loadlock while loading and unloding the other, introducing concurrency at yet another level.

A Petri net model of a dual–loadlock cluster tool is outlined in Fig.12, where the details of the wafer processing are abstracted by a single transition $t_{sum}$ with the occurrence time equal to $\tau_{init} + \tau_{steady} + \tau_{end}$.

It can be observed in Fig.12(b) that the two loadlocks are connected by a "selection loop" (with a single token) which switches the loadlock working with the chambers, and makes the other loadlock available for loading and unloading.

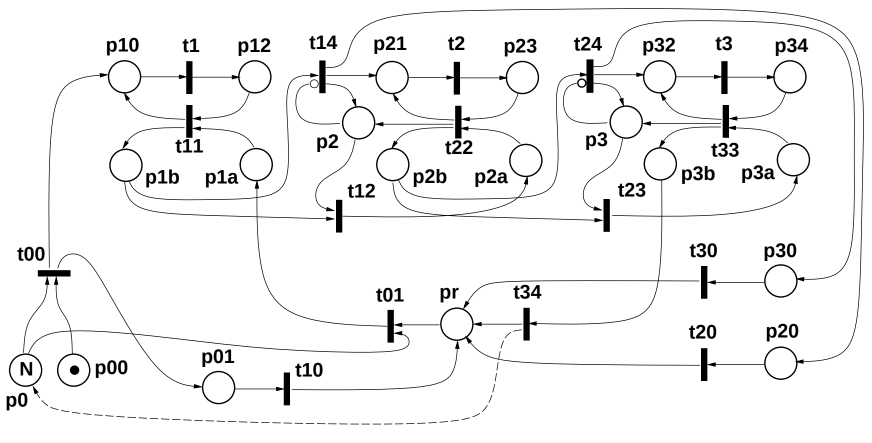

Fig. 9. Petri net model for initial transient behavior of a dual–blade tool.

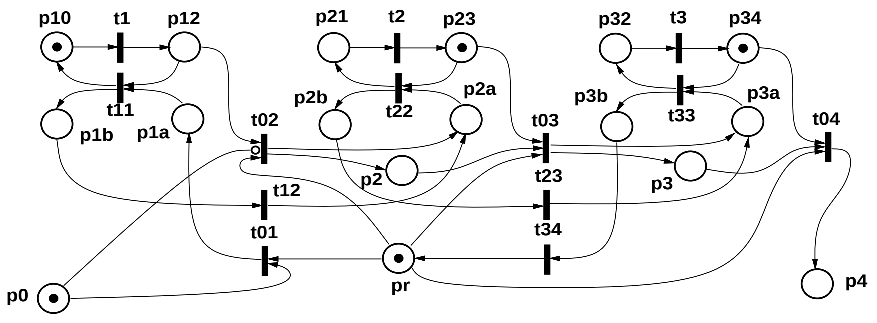

Fig. 10. Petri net model for final transient behavior of a dual–blade 3–chamber tool.

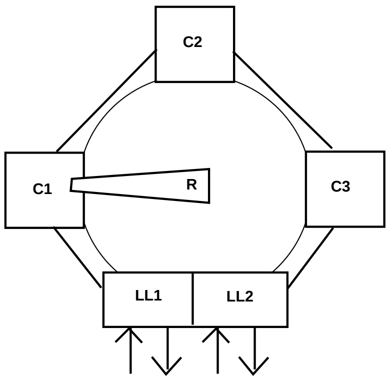

Fig. 11. An outline of a 3–chamber 2–loadlock cluster tool.

Using P–invariants and subnets implied by them, the total time for processing a batch using a single–loadlock tool is (Fig.12(a)):

$$\tau_{batch}^{(1)} \;=\; \tau_{load} + \tau_{sum} + \tau_{unload}$$

while for a dual-loadlock cluster tool (Fig.12(b)) it is:

$$\tau_{batch}^{(2)} \;=\; \max(\tau_{sum}, (\tau_{load} + \tau_{unload} + \tau_{sum})/2)$$

so, if the time $\tau_{sum}$ is comparable with $\tau_{load} + \tau_{unload}$, the dual–loadlock cluster tool can have the throughput almost twice that of a single–loadlock tool. On the other hand, if $\tau_{sum}$ is much greater than $\tau_{load} + \tau_{unload}$, the performance advantages of a dual–loadlock tool are rather insignificant (but there can be a significant difference in availability between these two types of tools, especially when the loadlock is not the most reliable component of a tool).

## B. Multiple Chambers

For process bound tools, one (or more) of the chambers is usually the bottleneck, i.e., the element which limits the performance of the entire cluster tool. It is known [4] that the bottleneck is the component with the maximum "service demand"; in the context of an $m$-chamber cluster tool, it will be the chamber with the longest processing time. An obvious approach to improve the throughput of such a system is to duplicate the critical chamber and use both chambers alternatively to increase the throughput of the tool. Fig.13 shows an outline of a 3–chamber cluster tool with replicated chamber C2.

The Petri net model of a dual–blade cluster with two identical chambers C2, for steady–state behavior, is shown in Fig.14, in which the two chambers C2 are connected by a "selection loop" $(p_2', t_{22}', p_2'', t_{22}'')$, similar to the one in Fig.12. The two copies of chamber C2 are used alternatively, processing two wafers simultaneously, and therefore increasing the throughput of the tool.

The net shown in Fig.14 has six P–invariants, four invariants corresponding to the chambers, one representing the robot's cycle of operations, and one corresponding to

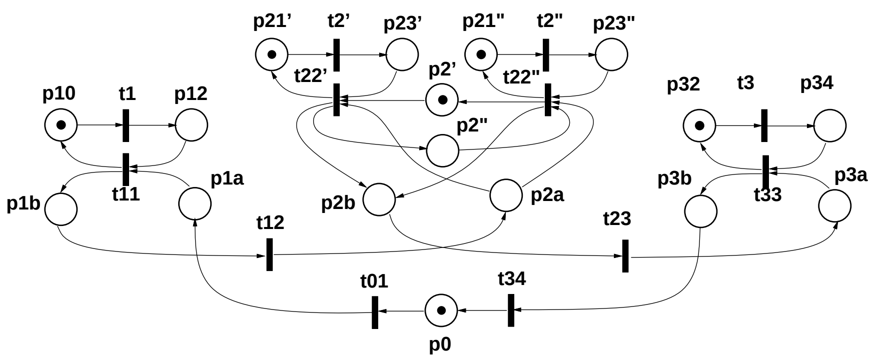

Fig. 14. Petri net model for the steady–state behavior of a dual–blade 3–chamber tool with two chambers C2.

the loop selecting the redundant chambers. The sets of transitions in the implied subnets are:

| *invariant* | *set of transitions* |
|---|---|
| 1 | $t_1, t_{11}$ |
| 2 | $t'_2, t'_{22}$ |
| 3 | $t''_2, t''_{22}$ |
| 4 | $t_3, t_{33}$ |
| 5 | $t'_{22}, t''_{22}$ |
| 6 | $t_{01}, t_{11}, t_{12}, t'_{22}, t''_{22}, t_{23}, t_{33}, t_{34}$ |

The cycle time, in this case, is determined in a slightly different way because different elements of the net are used with different frequencies within a single cycle of the net. T–invariants determine the frequencies of transition occurrences within each cycle. For the net in Fig.14, the only T–invariant is equal to 1 for $t'_2, t''_2, t'_{22}$ and $t''_{22}$, and is equal to 2 for all remaining transitions. Consequently,

$$\tau_0^{(2)} = 0.5 * \max(2\tau_1, \tau_2, \tau_3, 2\tau_4, \tau_6)$$

where $\tau_5$ is ignored because its set of transitions is a proper subset of that for invariant 6, and the initial factor 0.5 is due to the fact that 2 wafers are processed in each full cycle of this model. $\tau_6$ is a weighted sum of T–invariants and occurrence times of the corresponding transitions:

$$\tau_6 = 2f(t_{01}) + 2f(t_{11}) + 2f(t_{12}) + f(t''_{22}) + f(t'_{21}) + 2f(t_{23}) + 2f(t_{33}) + 2f(t_{34})$$

so, if the value of $\tau_0$ (Section 4.A) is determined by the value of $\tau_2$ (which is the reason of introducing multiple chamber C2), the cycle time $\tau_0^{(2)}$ is almost one half of the cycle time $\tau_0$.

It should be noted that the initial and final transient behaviors of tools with multiple chambers become more complicated, because of initial loading (and final unloading) of all wafers from the multiple chambers. The details are not presented here; they can be derived similarly as for the cases discussed earlier.

## *C. Multiple Robots*

For cluster tools with many chambers, it may be beneficial to use several robots, each of which services a group of chambers. The groups of chambers are coordinated in such a way that the outcome of one group becomes the load for the next group. Since multiple robots introduce concurrency at the transport level, for cluster tools which are transport bound, a significant reduction of the cycle time can be achieved in this way.

Fig.15 shows an outline of a cluster tool with five chambers serviced by two robots; the first robot services chambers C1 and C2 and also loads chamber C3; the second robot unloads chamber C3 (when it is ready) and moves the unloaded wafers to consecutive chambers of the tool (C4 and C5) and then back to the loadlock LL.

The Petri net model of this single–blade cluster tool, for steady–state behavior, is shown in Fig.16. The model is composed of two sections corresponding to the two robots, which are connected by the submodel representing chamber C3.

The model shown in Fig.16 has seven P–invariants, one for each chamber and one for each robot, with the following sets of transitions in invariant–implied subnets:

| *invariant* | *set of transitions* |
|---|---|
| 1 | $t_1, t_{01}, t_{12}, t_{20}$ |
| 2 | $t_2, t_{12}, t_{23}, t_{31}$ |
| 3 | $t_3, t_{23}, t_{34}$ |
| 4 | $t_4, t_{34}, t_{45}, t_{53}$ |
| 5 | $t_5, t_{45}, t_{56}, t_{64}$ |
| 6 | $t_{01}, t_{13}, t_{23}, t_{31}, t_{12}, t_{20}$ |
| 7 | $t_{34}, t_{35}, t_{56}, t_{64}, t_{45}, t_{53}$ |

If the operations required for servicing the chambers are split between the two robots, for transport bound cluster tools the reduction of the cycle time due to the second robot can be up to two times.

## *D. Generalized Models*

Since all presented models have very “modular” structure, with easily identifiable submodels corresponding to chambers, loadlocks and robots, the models can easily be generalized to cover whole families of cluster tools with similar structures.

For example, the steady–state model of a single–blade cluster tool with $k$ chambers is a straightforward extension of the model shown in Fig.3; the section representing a single chamber (and its robot's operations) is replicated $k$ times, as shown in Fig.17 for $k = 5$.

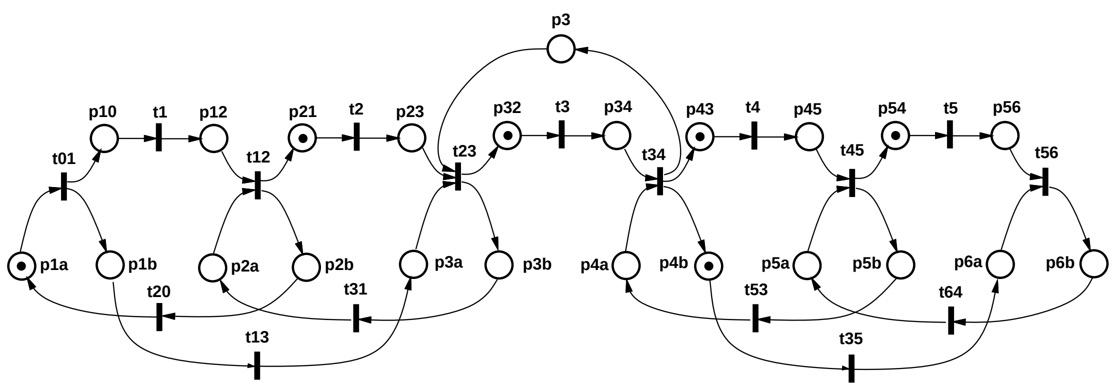

Fig. 16. Petri net model for the steady–state behavior of a single–blade 5–chamber cluster tool with two robots.

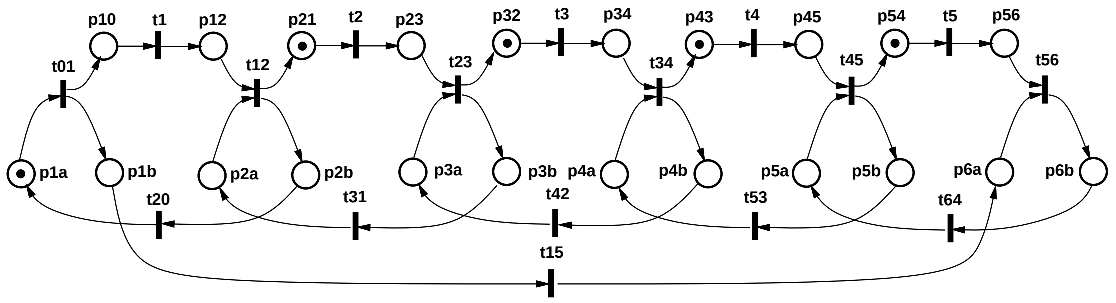

Fig. 17. Petri net model for the steady–state behavior of a single–blade 5–chamber cluster tool.

The model shown in Fig.17 has six P–invariants, five of them related to the five chambers, and the last one describing the robot's cycle of operation. In the general case, such a model for a tool with $k$ chambers will have $k+1$ P–invariants, with the cycle time defined by:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, ..., \tau_k, \tau_{k+1})$$

where $\tau_i = f(t_i) + f(t_{i-1,i}) + f(t_{i,i+1}) + f(t_{i+1,i-1})$, for $1 \le i \le k$, and $\tau_{k+1} = f(t_{01}) + f(t_{12}) + ... + f(t_{k,k+1}) + f(t_{k+1,k-1}) + f(t_{k,k-2}) + ... + f(t_{20}) + f(t_{1,k})$.

For dual–blade $k$–chamber cluster tool, the general steady–state solution is also based on $k+1$ invariant–implied subnets, and the subnet cycle times are $\tau_i = f(t_i) + f(t_{ii})$ for $1 \le i \le k$, and $\tau_{k+1} = f(t_{01}) + f(t_{11}) + f(t_{12}) + f(t_{22}) + ... + f(t_{kk}) + f(t_{k,k+1})$.

Similarly, a model of a cluster tool with $k$ loadlocks is a straightforward generalization of the model outlined in Fig.12(b), and the only difference is that the "selection loop" connects $k$ loadlocks sections instead of just two that are shown in Fig.12(b). Fig.18 shows the loadlock selection part for the case of three loadlocks; the characteristic modularity that can be seen in this model is used to derive the cycle time for a general case of a cluster tool with $k$ loadlocks (assuming that all the loadlocks are identical and that the time of switching from one loadlock to another is negligible):

$$\tau_0^{(k)} \;=\; \max(\tau_{sum}, (\tau_{load} + \tau_{unload} + \tau_{sum})/k).$$

## VI. Concluding Remarks

A systematic approach to modeling and analysis of a large variety of cluster tools has been presented. It uses timed Petri nets to represent the activities of the modeled tools, including the durations of these activities. The developed models represent steady–state and well as transient behaviors, so the analysis can cover all stages of wafer processing. On the other hand, simplified models can be derived for selected behaviors (e.g., steady–state), for which general symbolic results can be obtained by structural analysis of net models.

The proposed approach is modular in the sense, that more complicated models can be derived from simpler ones by replicating some sections of the model. This modular structure can be used to derive general performance characteristics for "standardized" tools, even without detailed analysis of their net models.

The proposed approach can also be used to model other aspects of cluster tools, which are not discussed in this paper. For example, many cluster tools use routing with revisiting some of the chambers. Although the modeling approach in such cases is slightly different, the basic ideas remain very similar. If, for example, a 3–chamber tool (as outlined in Fig.2) is used with revisiting chamber C2, the flow of wafers is:

$$\text{LL} \rightarrow \text{C1} \rightarrow \text{C2} \rightarrow \text{C3} \rightarrow \text{C2} \rightarrow \text{LL}$$

and the Petri net model can be (conceptually) derived from a 4–chamber model in which the chambers C2 and C4 are "folded" into one, by modifying the operations of the robot.

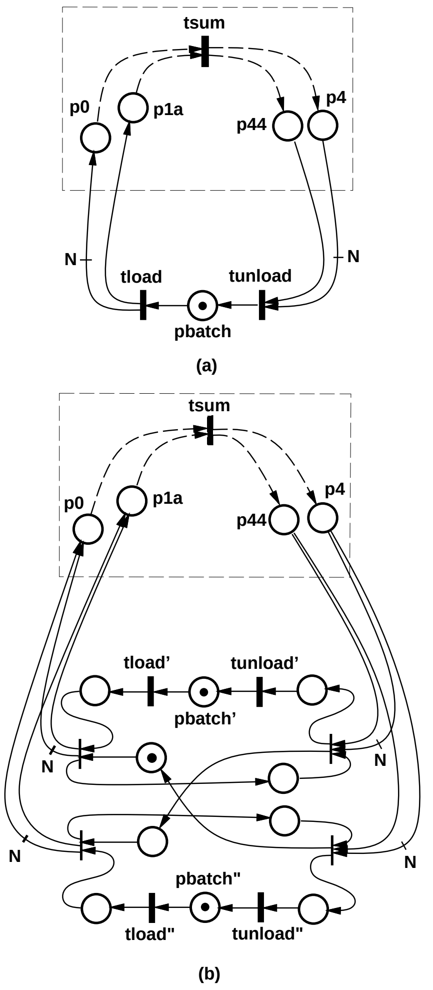

Fig. 12. Outline of a model for a cluster tool with single loadlock (a) and with two loadlocks (b).

Fig.19 shows one of possible Petri net models for such a tool with a dual–blade robot; the double use of chamber 2 in each cycle is represented by two unload/load transitions ($t_{2a}$ and $t_{2b}$) associated with chamber 2; $t_{2a}$ for the first load and second unload operations and $t_{2b}$ for the second load and first unload operations.

The model of chamber 3 is different from the others because the load and unload operations are performed separately for this chamber (the chamber is empty when the load operations is performed, so there is nothing to unload, and there is no "next" wafer to be loaded when the unload operation is performed).

The sequence of robot's operations is shown in Tab.4.

The net shown in Fig.19 has one T–invariant (with values equal to 1 for all transitions except of $t_2$, for which the value is 2, representing the two visits of each wafer), and four P–invariants, one for each chamber, and one for the robot. The cycle time can thus be evaluated in the same way as before.

The proposed approach can also be used for modeling and analysis of cluster tools which process several types of wafers within the same batch. Fig.20 shows a simple

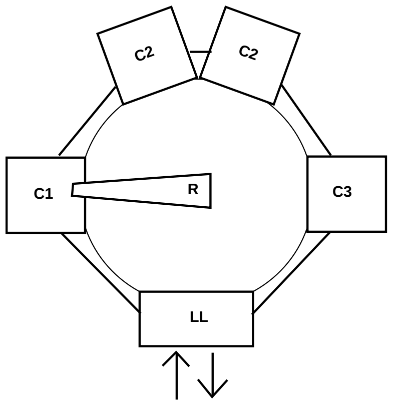

Fig. 13. An outline of a 3–chamber tool with two chambers C2.

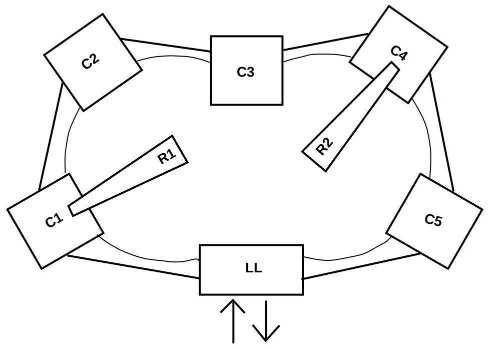

Fig. 15. An outline of a 5–chamber tool with two robots.

model for a dual–blade 3–chamber cluster tool which processes two types of wafers. The wafer type, for consecutive wafers picked from the loadlock, is generated by a simple cyclic subnet with transitions $t_{x1}$ and $t_{x2}$; by modifying this subnet, any other pattern of wafer types can be generated.

The type of a wafer is passed along the chambers using the upper (type–1) or lower (type–2) path. Each chamber is a free–choice structure (places $p_1$, $p_2$ and $p_3$) which allows two different types of processing (represented by $t'_1$, $t'_2$ and $t'_3$ for type–1 wafers, and by $t''_1$, $t''_2$ and $t''_3$ for type–2 wafers). The remaining part of the model is similar to the model shown in Fig.8 although the operations of unloading and loading the chambers are represented separately in Fig.20 because they may involve wafers of different types.

In the case when the batch is a random collection of wafer types, a different approach is needed; the cycle connecting $t_{x1}$ and $t_{x2}$ should be removed; the two transitions become free–choice, and then the selection of wafer type becomes random, described by "choice probabilities" assigned to $t_{x1}$ and $t_{x2}$.

Symbolic results derived at the end of Section 3 correspond directly to the fixed and incremental cycle time proposed in [21], where the (average) time required for processing a batch of $N$ wafers is characterized by two parameters, $\tau_{fixed}$, the 'fixed cycle time', and $\tau_0$, the incremental time per one wafer during the steady–state behavior:

$$\tau_{batch} \;=\; \tau_{fixed} + N\tau_0$$

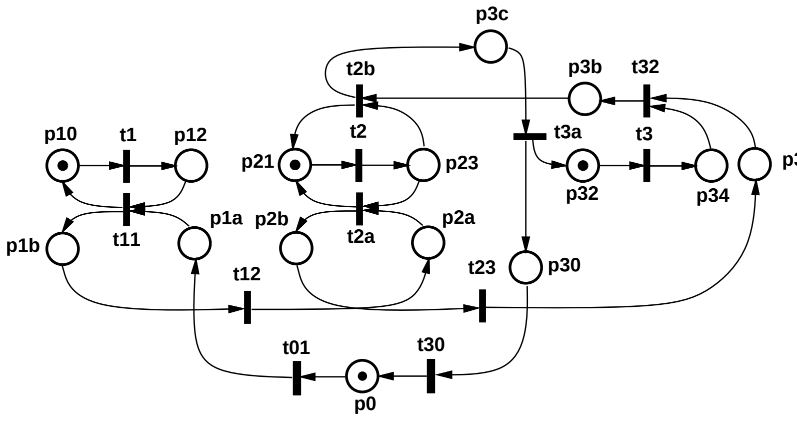

Fig. 19. Petri net model for a dual–blade 3–chamber cluster tool with revisiting C2.

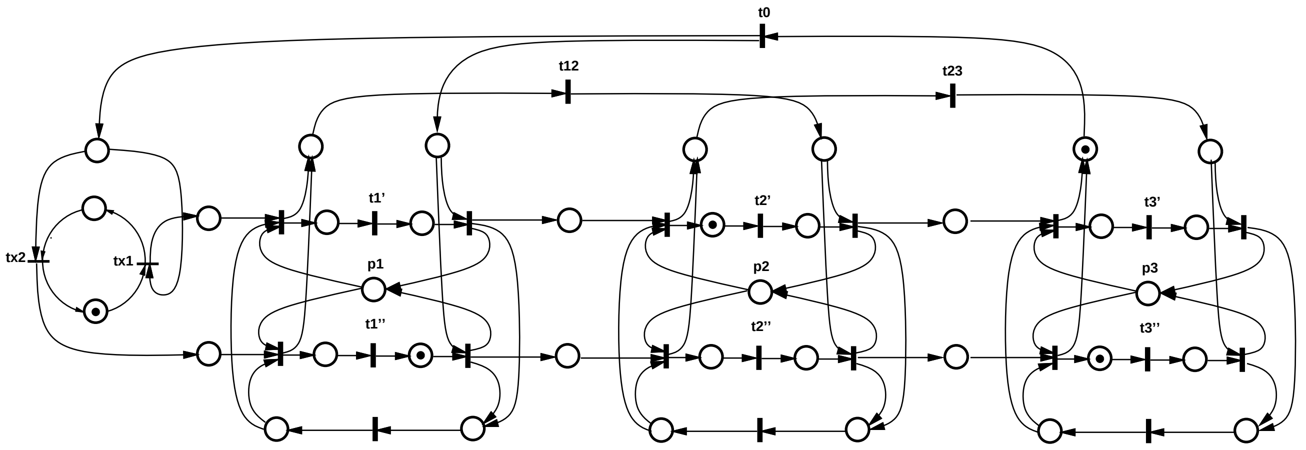

Fig. 20. Petri net model for a dual–blade 3–chamber cluster tool with two types of wafers.

The only significant difference with respect to formulas derived earlier in this paper is that the simple formula of [21] does not take the transient behaviors into account, so it underestimates the batch processing time.

Models of cluster tools contain many repetitions of the same subnets (which represent the chambers or the loadlocks). All such multiple subnets can be "folded" into one if a high–level Petri net is used. For example, in colored Petri nets [5], tokens have attributes (called colors) which may be used to distinguish the actual chamber or loadlock in such a folded structure. Colored Petri net model simplify the structure of the model, but their analysis becomes more elaborate than in the case of place/transition nets.

Finally, it should be noted that net models can easily become quite complicated, so their analysis without appropriate tools may be too difficult to be practical. Fortunately, many software tools have been developed for analysis of different classes of Petri nets [24], and there are some efforts to standardize the representation of net models which, hopefully, will further increase their popularity.

## Acknowledgements

The Natural Sciences and Engineering Research Council of Canada partially supported this research through grant RGPIN-8222.

The author is grateful for several interesting suggestions and remarks of three anonymous referees, taken into account during the revision of this paper.

## References

[1] M. Ajmone Marsan, G. Conte, G. Balbo, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems"; *ACM Trans. on Computer Systems*, vol.2, no.2, pp.93–122, 1984.

[2] F. Bause, P.S. Kritzinger, *Stochastic Petri nets – an introduction to the theory* (Academic Studies in Computer Science); Vieweg Verlag 1996.

[3] P. Burggraaf, "Coping with the high cost of wafer fabs"; *Semiconductor International*, vol.18, no.3, pp.45–50, 1995.

[4] D. Ferrari, *Computer systems performance evaluation*; Prentice–Hall 1978.

[5] K. Jensen, "Coloured Petri nets"; in *Advanced Course on Petri Nets 1986* (Lecture Notes in Computer Science 254), pp.248–299, Springer–Verlag 1987.

[6] J. Kim, A.A. Desrochers, "Modeling and analysis of semiconductor manufacturing plants using time Petri net models"; Proc. IEEE Int. Conference on Systems, Man, and Cybernetics (SMC'97), pp.3227–3232, 1997.

[7] F. Krueckeberg, M. Jaxy, "Mathematical methods for calculating invariants in Petri nets"; in *Advances in Petri Nets 1987* (Lecture Notes in Computer Science 266), pp.104–131, Springer–Verlag 1987.

[8] J. Martinez, M. Silva, "Simple and fast algorithm to obtain all invariants of a generalized Petri net"; in *Applications and*

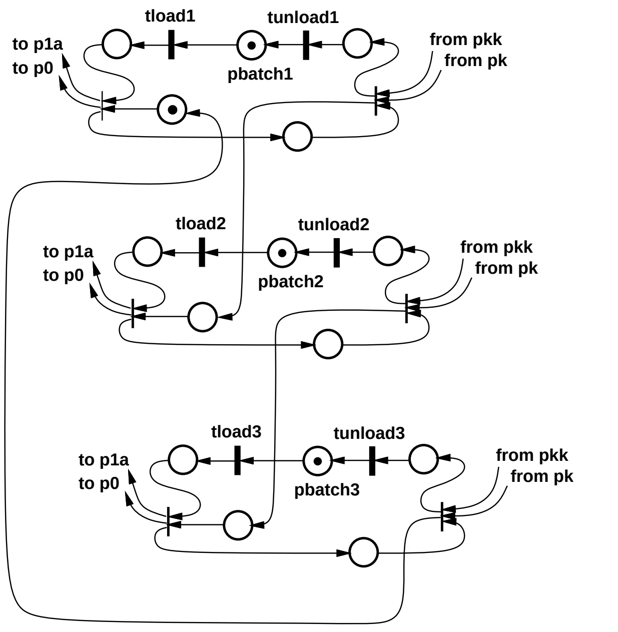

Fig. 18. Petri net model of the loadlock selection for the case of three loadlocks.

TABLE IV
Operations Represented by Transitions in Fig.19.

| *transition* | *operation* |
|---|---|
| $t_{01}$ | pick a wafer from the loadlock, rotate and move to chamber 1; |
| $t_{11}$ | unload the wafer from chamber 1, rotate and load chamber 1; |
| $t_{12}$ | move the robot to chamber 2; |
| $t_{2a}$ | unload the wafer from chamber 2, rotate and load chamber 2; |
| $t_{23}$ | move to loadlock, rotate and drop the wafer, move to chamber 3; |
| $t_{2b}$ | unload the wafer from chamber 2, rotate and load chamber 2; |
| $t_{3a}$ | move to chamber 3, rotate and load chamber 3; |
| $t_{30}$ | move the robot to loadlock. |

*Theory of Petri Nets* (Informatik Fachberichte 52); pp.301–310, Springer–Verlag 1982.

[9] P.M. Merlin, D.J. Farber, "Recoverability of communication protocols – implications of a theoretical study"; *IEEE Trans. on Communications*, vol.24, no.9, pp.1036–1049, 1976.

[10] T. Murata, "Petri nets: properties, analysis and applications"; *Proceedings of IEEE*, vol.77, no.4, pp.541–580, 1989.

[11] T.L. Perkinson, R.S. Gyurcsik, P.K. MacLarty, "Single-wafer cluster tool performance: an analysis of the effects of redundant chambers and revisitations sequences on throughput"; *IEEE Trans. on Semiconductor Manufacturing*, vol.9, no.3, pp.384-400, 1996.

[12] T.L. Perkinson, P.K. MacLarty, R.S. Gyurcsik. R.K. Cavin III, "Single–wafer cluster tool performance: an analysis of throughput"; *IEEE Trans. on Semiconductor Manufacturing*, vol.7, no.3, pp.369-373, 1994.

[13] C.V. Ramamoorthy, G.S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets"; *IEEE Trans. on Software Engineering*, vol.6, no.5, pp.440-449, 1980.

[14] R.R. Razouk, C.V. Phelphs, "Performance analysis using timed Petri nets"; in *Protocol Specification, Testing, and Verification IV* (Proc. of the IFIP WG 6.1 Fourth Int. Workshop, Skytop Lodge PA), pp.561-576, North-Holland 1985.

[15] W. Reisig, *Petri nets – an introduction*; Springer–Verlag 1985.

[16] J. Sifakis, "Use of Petri nets for performance evaluation"; in *Measuring, modeling and evaluating computer systems*, pp.75–93, North–Holland 1977.

[17] Singer, "The driving forces in cluster tool development"; *Semiconductor International*, vol.18, no.8, pp.113-118, 1995.

[18] R.S. Srinivasan, "Modeling and performance analysis of cluster tools using Petri nets"; *IEEE Trans. on Semiconductor Manufacturing*, vol.11, no.3, pp.394-403, 1998.

[19] R. Valk, "Test on zero in Petri nets"; in *Applications and Theory of Petri Nets* (Informatik-Fachberichte 52), pp.193–197, Springer Verlag 1982.

[20] S. Venkatesh, R. Davenport, P. Foxhoven, J. Nulman, "A steady–state throughput analysis of cluster tools: dual–blade versus single-blade robots"; *IEEE Trans. on Semiconductor Manufacturing*, vol.10, no.4, pp.418–423, 1997.

[21] R. Wood, "Simple performance models for integrated processing tools"; *IEEE Trans. on Semiconductor Manufacturing*, vol.9, no.3, pp.320–328, 1996.

[22] W.M. Zuberek, "Timed Petri nets – definitions, properties and applications"; *Microelectronics and Reliability* (Special Issue on Petri Nets and Related Graph Models), vol.31, no.4, pp.627–644, 1991.

[23] W.M. Zuberek, W. Kubiak, "Timed Petri nets in modeling and analysis of simple schedules for manufacturing cells"; *Journal of Computers and Mathematics with Applications*, vol.37, no.11/12, pp.191–206, 1999.

[24] DAIMI, Department of Computer Science at Aarhus University, Denmark, maintains a database of tools for analysis of Petri nets: `http://www.daimi.au.dk/PetriNets`.

**Wlodek M. Zuberek** received M.Sc. degree in Electronic Engineering and Ph.D. degree in Computer Science, both from Warsaw University of Technology. Currently he is a Professor in the Department of Computer Science of Memorial University of Newfoundland, and the Chair of the recently created Interdisciplinary Computational Science Program. His research interests include modeling and performance analysis of concurrent systems, and in particular applications of timed Petri nets, discrete–event simulation, hierarchical modeling, as well as the use of formal methods in analysis of complex concurrent systems.

Dr. Zuberek is a member of ACM, IEEE CS, and GI FG 0.0.1.