14 Congresso Brasileiro de Automatica, Natal Brazil, 2-5 September 2002
Brazilian Petri Net Workshop, 4 September 2002

Systematic Construction and Performance Analysis of Cluster Tools Using Timed Petri Net Models

W.M. Zuberek

Department of Computer Science
Memorial University
St.John's, Canada A1B 3X5
e-mail: wlodek@cs.mun.ca

Abstract. A cluster tool is an integrated, environmentally isolated manufacturing system consisting of process, transport, and cassette modules, mechanically linked together, that is used in manufacturing of semiconductor chips. Because of high throughput requirements, cluster tools perform a number of activities concurrently. Petri nets are formal models developed specifically for representation of concurrent activities and for their coordination. In timed nets, the durations of modeled activities are represented by occurrence times associated with transitions, and this allows to study the performance characteristics of the modeled systems.

Since cluster tools can be quite complex, a systematic approach to generating net models is proposed. Net models derived in such a way have modular structure, which is used to determine model's steady–state performance on the basis of net invariants, without the exhaustive reachability analysis. Performance characteristics are obtained in symbolic form, in terms of modeling parameters, so different variants of cluster tools can be evaluated and compared very efficiently, without repetitive model analyses.

1. Introduction

A cluster tool is an integrated, environmentally isolated manufacturing system consisting of process, transport, and cassette modules, mechanically linked together [3], [11], [21]. 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 [18]. 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. 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, are known as Petri nets [15], [10]. 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 events, and the (occurrences of) events; events 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 (sometimes called the causality relation). The dynamic behavior of nets is represented using the so called tokens assigned to places. The occurrences of events (represented by transitions) change the distribution of tokens, which is used to represent the behavior of the modeled system.

In order to analyze the performance of modeled systems, the durations of all activities must also be taken into account. Several types of nets "with time" have been proposed by associating "time delays" with places [16], or occurrence durations with transitions [1], [13], [23] of net models. Also, such temporal properties can be deterministic [13], [14], [16], [23], or can be random variables described by probability distribution functions (the negative exponential distribution being probably the most popular choice) [1], [2], [23].

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 quite difficult because of a 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, the performance of cluster tools was analyzed by using timing diagrams representing typical sequences of events, with performance formulas derived from the critical path determining the cyclic behavior of a tool [11], [12], [21]; 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 proposes a systematic approach to deriving timed Petri nets models of cluster tools. The approach is based on formal description of tool configurations and changes of configurations corresponding to transfers of wafers between the chambers and loadlocks of the tool. The approach 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. The performance of the derived models is obtained by structural analysis (for the steady–state), as an alternative approach to the one presented in [19], where the performance of Petri net models is obtained 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 generated by the proposed approach 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 proposed 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 [26] 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 to use net extensions (e.g., multiple arcs) which are useful in modeling of cluster tools [25].

The approach presented in this paper is similar to an approach developed earlier for modeling and analysis of schedules for manufacturing cells [26].

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 using 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. Section 3 introduces single–blade cluster tools and their description, and Section 4 presents net models of cluster tools derived from the descriptions discussed in Section 3. Dual–blade cluster tools are introduced in Section 5. Section 6 discusses some generalization of the model presented in Section 4, and Section 7 describes an extension of the approach presented in Section 3, that is needed to model and evaluate cluster tools with chamber revisiting. Several concluding remarks are given in Section 8.

2. 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], [20]. 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) Petri net $\mathcal{M}$ is defined as $\mathcal{M} = (\mathcal{N}, m_0)$, where the net structure $\mathcal{N}$ is a bipartite directed multigraph, $\mathcal{N} = (P, T, A, w)$, 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, $A \subseteq T \times P \cup P \times T$, 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 *free–choice* if the sets of places connected by directed arcs for all transitions sharing p are identical and the weights of the arcs are the same. A net is free-choice if all its shared places are free–choice. A marked net is (structurally or statically) *conflict–free* if it does not contain shared places. 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 [23], 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 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 [23]. 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 [23].

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

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

Connectivity matrices disregard '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 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 $\mathcal{N}_i = (P_i, T_i, A_i, w)$ is a P_i-implied subnet of a net $\mathcal{N} = (P, T, A, w)$, $P_i \subset P$, if the following holds:

(1) $A_i = A \cap (P_i \times T \cup T \times P_i)$, and

(2) $T_i = \{t \in T \mid \exists\, p \in P_i : (p,t) \in A_i \vee (t,p) \in A_i\}$.

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 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, τ_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 τ_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 [26]. Fig.1 shows one of such transformations; it reduces a parallel path which has no influence on the behavior of a timed net, but which can increase the number of inessential (basic) P–invariants.

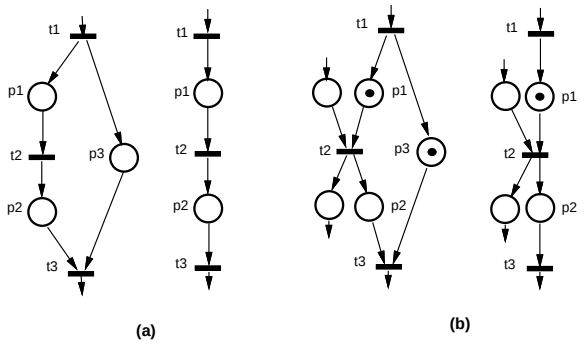

Fig.1. Parallel path reduction.

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 noted that the reduction of parallel paths can be performed only if either all paths are unmarked, as in Fig.1(a), or all are marked, as in Fig.1(b); the paths to be reduced cannot be "mixed", i.e., one path marked and the other unmarked.

3. 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 4–chamber cluster tool (used as a running example) is shown in Fig.2, where LL denotes the loadlock to store cassettes of wafers; C1, C2, C3 and C4 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.

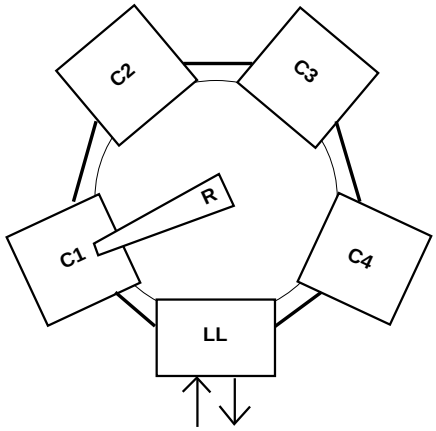

Fig.2. An outline of a 4–chamber cluster tool.

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 τ_{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 τ_{unload}.

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

Since most of the batch processing time is spent in the steady–state, the analysis of steady–state processing is usually the most interesting one. The initial and final transient behaviors can be approximated reasonably well by the cycle time of the steady–state behavior.

The behavior of a cluster tool, with a single–blade robot, can be represented as a sequence of "configurations", where each configuration corresponds to a distribution of wafers among the chambers of the tool (when the robot does not carry a wafer); more specifically, for an m–chamber tool, each configuration is described by an m–tuple of chamber descriptions (it should be noted that loadlocks are excluded from these descriptions in order to capture the cyclic behavior of the steady–state; from the steady–state point of view, the loadlocks provide an infinite suply of wafers for processing):

$$(k_1, k_2, ..., k_m)$$

where each chamber description k_i is "1" if the chamber C_i is loaded with a wafer in this configuration, and otherwise is "0". For example, the sequence of configurations for a 4–chamber tool with all chambers used concurrently is shown in Tab.1.

Each change of configurations corresponds to a wafer moving from one chamber to another, from the loadlock to the first chamber, or from the last chamber back to the loadlock; it is assumed that each cycle uniformly begins by moving a (new) wafer from the loadlock to the first chamber (so, in the first configuration, $k_1 = 0$).

The changes of configurations correspond to the following general rules:

- a configuration $(k_1, ..., k_i, k_{i+1}, ..., k_m)$ derives a configuration $(k_1, ..., k_i - 1, k_{i+1} + 1, ...k_m)$ if and only if the value of k_i is "1" and the value of k_{i+1} is "0", $i = 1, ..., m - 1$;
- a configuration $(k_1, k_2, ..., 1)$ always derives a configuration $(k_1, k_2, ..., 0)$ (this change corresponds to moving a wafer from the last chamber Cm to loadlock),
- it is assumed that each cycle begins by moving a (new) wafer from the loadlock to chamber C1, so the first derivation is always from $(0, k_2, ...k_m)$ to $(1, k_2, ..., k_m)$.

It can be easily verified that for the case of maximally concurrent use of chambers, there is only one sequence of operations, as shown in Tab.1. However, if the concurrency is reduced, and only two chambers are performing their operations when the next wafer is loaded into C1, there are several possible sequences of operations, as shown in Tab.2. These sequences are 1–2–3–4a–5a–1, 1–2–3–4a–5b–1, 1–2–3–4b–5b–1. In general case, one of these sequences will provide a better throughput than the others.

Table 1: Sequence of configurations for a 4–chamber tool with the maximally concurrent use of chambers.

configuration	*next operation*
(0,1,1,1)	next wafer is moved from LL to C1;
(1,1,1,1)	the wafer from C4 is moved to LL;
(1,1,1,0)	the wafer from C3 is moved to C4;
(1,1,0,1)	the wafer from C2 is moved to C3;
(1,0,1,1)	the wafer from C1 is moved to C2;
(0,1,1,1)	this is the initial configuration.

4. Models of Cluster Tools

The description of a cluster tool discussed in the previous section can easily be converted into a timed Petri net model of this tool. In this model, each chamber is represented by a simple subnet shown in Fig.3. Place p_i is marked if the chamber is empty. Transition t_{ia} represents the operation of loading the chamber, and place p_{ia} – the condition "wafer is loaded into chamber", so the chamber operation can begin; transition t_i represent the operation performed by the chamber with the occurrence time equal to the duration of this operation. Place p_{ib} represents the condition "chamber operation is completed", so the unloading can be performed (transition t_{ib}).

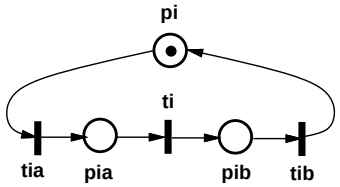

Fig.3. Petri net model of a chamber.

The (cyclic) sequence of operations performed by the robotic transporter is derived from the sequence of configurations of the cluster tool. For example, the sequence of configurations corresponding to the maximally concurrent operation of a 4–chamber cluster tool implies the following sequence of robot steps (starting with moving the next wafer from loadlock LL to chamber C1):

Table 2: Alternative sequences of configurations for a 4–chamber tool.

step	*configuration*	*next operation*
1	(0,1,1,0)	next wafer is moved from LL to C1;
2	(1,1,1,0)	the wafer from C3 is moved to C4;
3	(1,1,0,1)	the wafer from C2 can be moved to C3 (step 4a), or the wafer from C3 can be moved to the loadlock (step 4b);
4a	(1,0,1,1)	the wafer from C1 can be moved to C2 (step 5a), or the wafer from C3 can be moved to the loadlock (step 5b);
5a	(0,1,1,1)	the wafer from C3 is moved to LL;
1	(0,1,1,0)	the initial configuration;
4b	(1,1,0,0)	the wafer from C2 is moved to C3;
5b	(1,0,1,0)	the wafer from C1 is moved to C2;
1	(0,1,1,0)	the initial configuration.

$$\text{LL} \Rightarrow \text{C1} \rightarrow \text{C4} \Rightarrow \text{LL} \rightarrow \text{C3} \Rightarrow \\ \text{C4} \rightarrow \text{C2} \Rightarrow \text{C3} \rightarrow \text{C1} \Rightarrow \text{C2} \rightarrow \text{LL}$$

where X $\Rightarrow$ Y represents a move of the robot carrying a wafer from X to Y, and X $\rightarrow$ Y, a move without carrying a wafer. The model of this sequence is a simple cyclic net composed of transitions representing the steps and the intermediate places.

The model of chambers and the robotic transporter can be combined into a complete model of the tool shown in Fig.4.

The four chambers are represented (in the upper part of Fig.4) by subnets with transitions t_1, t_2, t_3 and t_4; the initial markings of chambers C2, C3 and C4 correspond to the maximum concurrency assumption – when a new wafer is picked from the loadlock, all chambers except C1 are loaded and perform their operations. The operations represented by the remaining transitions are described in Tab.3.

The initial marking (place p_{01}) indicates that the first robot's operation is to pick a wafer from the loadlock and move to C1 (transition t_{01}), then load the wafer in C1 (transition t_{1a}), and so on.

In order to obtain the effect of steady–state behavior, the loadlock is assumed to have an infinite capacity and is represented by place p_0 which is used as "input" and "output" of the cluster tool. When processing a wafer is finished, a token is deposited in p_0, and the same token is used as the next wafer a moment later. The initial marking of p_0 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_0. Moreover, it can be observed that p_0 creates a parallel path between t_{01} and t_{45}, so it has no effect on the performance of the model, and can be removed (with the two arcs connected to it). Similarly, places p_1, p_2, p_3 and p_4 can also be removed (with their incident arcs) without any effect on the performance of the model as they all create parallel paths (in [17] such places are called "implicit places").

Table 3: Operations represented by transitions in Fig.4.

transition	*operation*
t_{01}	pick next wafer from LL and move to C1;
t_{1a}	load the wafer into C1;
t_{1b}	unload C1;
t_{2a}	load the wafer into C2;
t_{2b}	unload C2;
t_{3a}	load the wafer into C3;
t_{3b}	unload C3;
t_{4a}	load the wafer into C4;
t_{4b}	unload C4;
t_{12}	move from C1 to C2;
t_{14}	move from C1 to C4;
t_{20}	move from C2 to LL;
t_{31}	move from C3 to C1;
t_{42}	move from C4 to C2;
t_{45}	move to LL and drop the wafer;
t_{53}	move from LL to C3.

All transitions are timed transitions, and the occurrence times associated with them represent the times of the corresponding operations.

The net shown in Fig.4 (after removal of places p_0, p_1, p_2, p_3 and p_4) has five basic P–invariants; the sets of transitions of subnets implied by these P–invariants are:

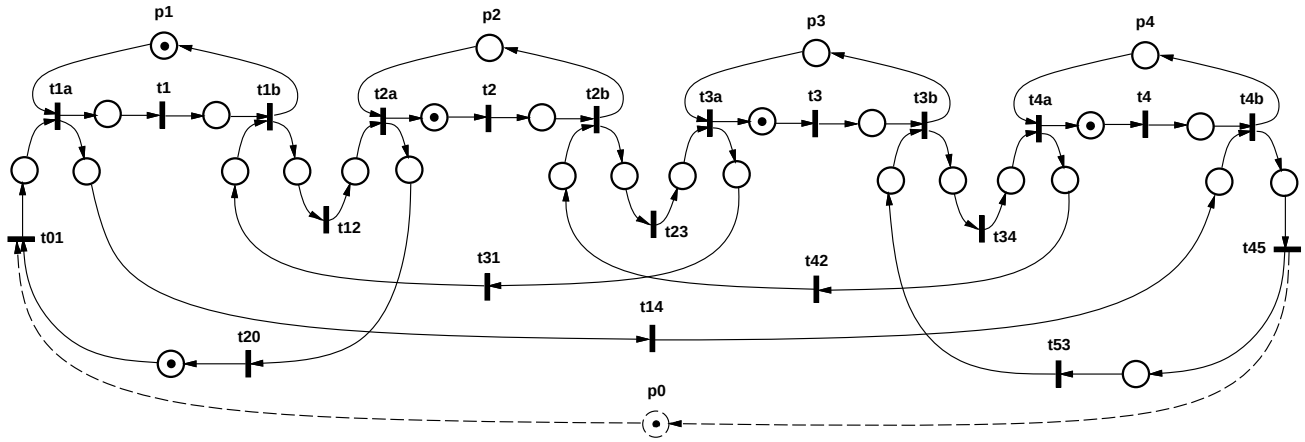

Fig.4. Petri net model of a 4–chamber tool.

invariant	*set of transitions*
1	$t_1, t_{1a}, t_{1b}, t_{2a}, t_{01}, t_{12}, t_{20}$
2	$t_2, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{12}, t_{23}, t_{31}$
3	$t_3, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{23}, t_{34}, t_{42}$
4	$t_4, t_{3b}, t_{4a}, t_{4b}, t_{34}, t_{45}, t_{53}$
5	$t_{01}, t_{12}, t_{23}, t_{34}, t_{45}, t_{14}, t_{20}, t_{31}, t_{42}, t_{53}$, $t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{4b}$

Because the cycle time of the model is equal to the maximum cycle time of subnets implied by P–invariants, the cycle time τ_0 is:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5)$$

where τ_i denotes the cycle time of the subnet i, so, $\tau_1 = f(t_1)+f(t_{1a})+f(t_{1b})+f(t_{2a})+f(t_{01})+f(t_{12})+f(t_{20})$, $\tau_2 = f(t_2)+f(t_{1b})+f(t_{2a})+f(t_{2b})+f(t_{3a})+f(t_{12})+f(t_{23})+f(t_{31})$, and so on (each P–invariant–implied subnet contains exactly one token).

If τ_0 is equal to one (or more) of the first four terms, the model is called "process bound" because the duration of the process performed by one of the chambers 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" [21].

The temporal characteristics associated with the transitions of the model can be determined by representing each step as a collection of some elementary actions such as picking a wafer from a loadlock, loading a wafer into a chamber or unloading it. 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*
v	pick a wafer from the loadlock;
x	load a wafer into a chamber;
y	unload a wafer from a chamber;
w	drop a wafer in the loadlock;
z	move the 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).

The execution time of any operation is simply assumed to be equal to the sum of execution times of actions constituting the operation. For the operations represented by transitions in Fig.4, these execution times are thus as follows:

transition	*execution time*
t_{01}	$v+z$
t_{1a}	x
t_{1b}	y
t_{2a}	x
t_{2b}	y
t_{3a}	x
t_{3b}	y
t_{4a}	x
t_{4b}	y
t_{12}	z
t_{14}	$2z$
t_{20}	$2z$
t_{31}	$2z$
t_{42}	$2z$
t_{45}	$w+z$
t_{53}	$2z$

and then the cycle times of the subnets are:

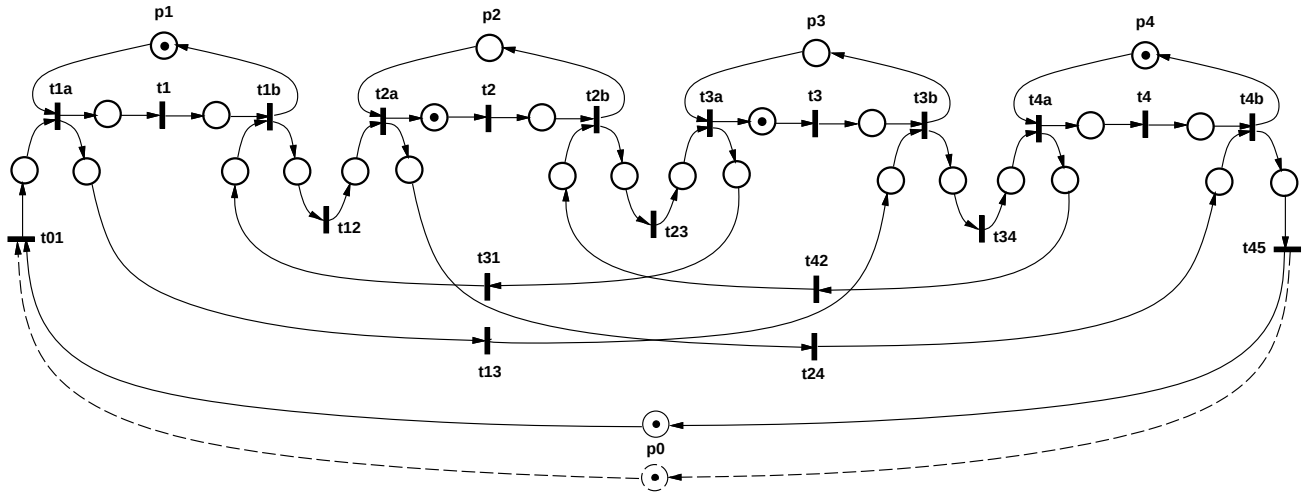

Fig.5. Alternative Petri net model of a 4–chamber tool.

$$\tau_1 = o_1 + v + 2x + y + 4z,$$
$$\tau_2 = o_2 + 2x + 2y + 4z,$$
$$\tau_3 = o_3 + 2x + 2y + 4z,$$
$$\tau_4 = o_4 + w + x + 2y + 4z,$$
$$\tau_5 = v + w + 4x + 4y + 15z,$$

where o_i denotes the duration of the operation performed by chamber C_i (or the occurrence time associated with transition t_i).

Similarly, the sequence of configurations 1–2–3–4a–5a–1 (Tab.2) corresponds to the following sequence of robot's moves:

$$\text{LL} \Rightarrow \text{C1} \rightarrow \text{C3} \Rightarrow \text{C4} \rightarrow \text{C2} \Rightarrow \text{C3} \rightarrow \text{C1} \Rightarrow \text{C2} \rightarrow \text{C4} \Rightarrow \text{LL}.$$

The net model derived from this sequence of operations is shown in Fig.5.

After removing p_0 with the two incident arcs, and also p_1, p_2, p_3 and p_4 with their arcs (as they all create parallel paths), the net has 6 basic P–invariants, and the sets of transitions implied by these invariants are:

invariant	*set of transitions*
1	$t_1, t_2, t_3, t_4, t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{4b}, t_{01}, t_{12}, t_{23}, t_{34}, t_{45}$
2	$t_1, t_{1a}, t_{1b}, t_{2a}, t_{4b}, t_{01}, t_{12}, t_{24}, t_{45}$
3	$t_2, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{12}, t_{23}, t_{31}$
4	$t_3, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{23}, t_{34}, t_{42}$
5	$t_4, t_{1a}, t_{3b}, t_{4a}, t_{4b}, t_{01}, t_{13}, t_{34}, t_{45}$
6	$t_{01}, t_{12}, t_{23}, t_{34}, t_{45}, t_{13}, t_{24}, t_{31}, t_{42}, t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{4b}$

The formulas describing the cycle times of this model can be derived similarly as for the model shown in Fig.4.

5. Dual–blade Tools

A dual–blade robot can hold two wafers at the same time, which makes the operations of moving wafers from one chamber to another significantly simpler than in the case of single–blade robots. Assuming that the robot carries a single wafer from one chamber to another (i.e., the second blade is empty), the typical steps, repeated for each chamber, are:

– unload the chamber,
– rotate the robot,
– load the wafer,
– move to the next chamber,

and then the (cyclic) sequence of robot moves, for the case of maximally concurrent use of chambers, is simply:

$$\text{LL} \Rightarrow \text{C1} \Rightarrow \text{C2} \Rightarrow \text{C3} \Rightarrow \text{C4} \Rightarrow \text{LL}.$$

The complete net model is shown in Fig.6, in which the initial marking indicates that the next wafer is picked (t_{01}) when all chambers are loaded with wafers.

The net shown in Fig.6 (after removing places p_0, p_1, p_2, p_3 and p_4 with their incident arcs, since they all create parallel paths) has 5 basic invariants, which imply subnets with the following sets of transitions:

invariant	*set of transitions*
1	$t_1, t_{1a}, t_{1b}, t_{11}$
2	$t_2, t_{2a}, t_{2b}, t_{22}$
3	$t_3, t_{3a}, t_{3b}, t_{33}$
4	$t_4, t_{4a}, t_{4b}, t_{44}$
5	$t_{01}, t_{12}, t_{23}, t_{34}, t_{40}, t_{11}, t_{22}, t_{33}, t_{44}, t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{4b}$

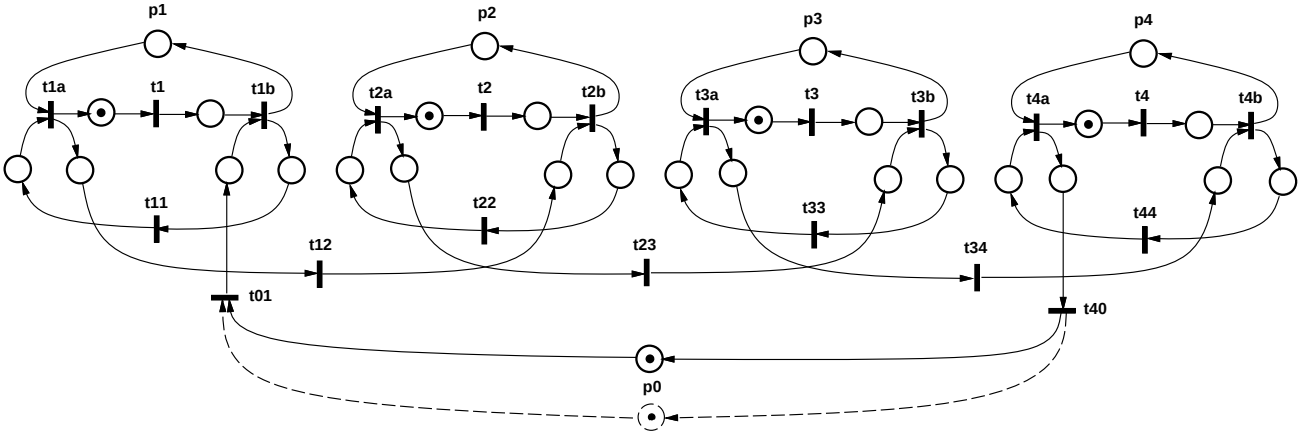

Fig.6. Petri net model of a 4–chamber tool with a dual–blade robot.

The following occurrence times are assigned to transitions in Fig.6 (u denotes the time required to rotate the robot; all other elementary actions are as before):

transition	*execution time*
t_{01}	$u+v+z$
t_{1a}	x
t_{11}	u
t_{1b}	y
t_{2a}	x
t_{22}	u
t_{2b}	y
t_{3a}	x
t_{33}	u
t_{3b}	y
t_{4a}	x
t_{44}	u
t_{4b}	y
t_{12}	z
t_{23}	z
t_{34}	z
t_{40}	$u+w+z$

The cycle time is equal to $\tau_0 = \max(\tau_1, ..., \tau_5)$, where:

$$\begin{aligned}
\tau_1 &= o_1 + u + x + y,\\
\tau_2 &= o_2 + u + x + y,\\
\tau_3 &= o_3 + u + x + y,\\
\tau_4 &= o_4 + u + x + y,\\
\tau_5 &= 6u + v + w + 4x + 4y + 5z,
\end{aligned}$$

where o_i, as before, denotes the duration of the operation performed by chamber C_i (or the occurrence time associated with transition t_i).

For the 'transport bound' case (i.e., when $\tau_0 = \tau_5$), a chamber tool with a dual–blade robot offers better performance than a tool with a single–blade robot if $6u < 10z$. Many similar conclusions can easily be derived by comparing the symbolic formulas describing the cycle times of the models.

6. Model Extensions

The steady–state performance of the (single–blade as well as dual–blade) model is limited by the capacity of a cassette of wafers; when processing of all wafers is completed, the cassette is replaced by another one, and the processing continues. Since, during reloading of the loadlock, all chambers remain idle, the performance of a cluster tool can be improved by using multiple loadlocks, as shown in Fig.7; when one of the loadlocks is being reloaded, processing of wafers can continue from the other loadlock.

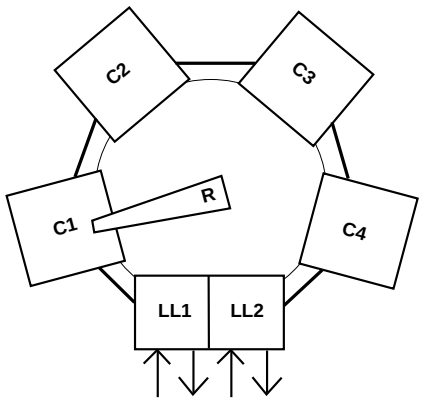

Fig.7. An outline of a 2–loadlock cluster tool.

Petri net model of a cluster tool with two loadlocks is outlined in Fig.8, in which all the chambers are replaced by a single transition t_{sum} with the occurrence

time equal to the time required for processing the entire batch of wafers.

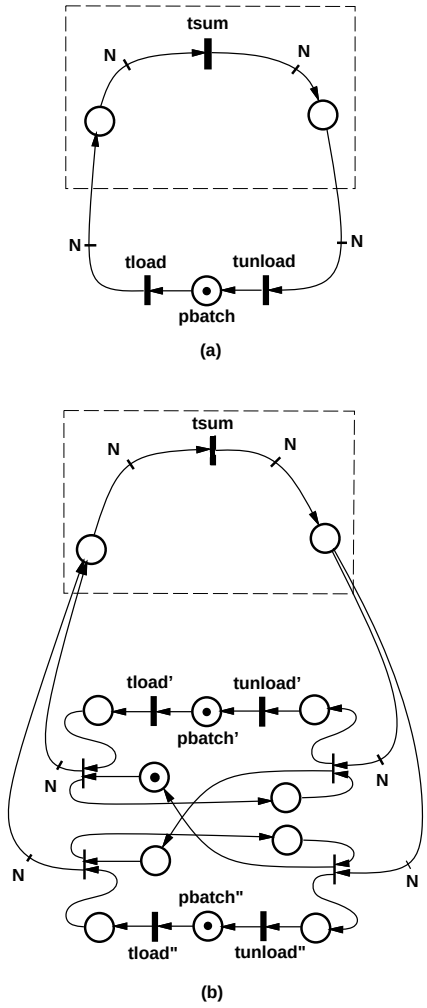

Fig.8. Outline of a model of a cluster tool with a single loadlock (a) and with two loadlocks (b).

It can be observed in Fig.8(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. Multiple arcs are used in Fig.8 to represent a batch of N wafers arriving to a loadlock as a single cassette, and processed in the time interval associated with t_{sum} as its occurrence time.

Using P–invariants and subnets implied by them, the total time for processing a batch using a single–loadlock tool (Fig.8(a)) is:

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

For a dual-loadlock cluster tool (Fig.8(b)), assuming the the loadlock loading and unloading times are the same for both loadlocks, the batch processing time is:

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

so, if the time τ_{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 τ_{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).

If the processing time of one of the chambers is significantly longer than processing times of the other chambers, increased performance can be obtained by using multiple chambers performing the same operation. An outline of a 3–chamber tool with dual chamber C2 is shown in Fig.9.

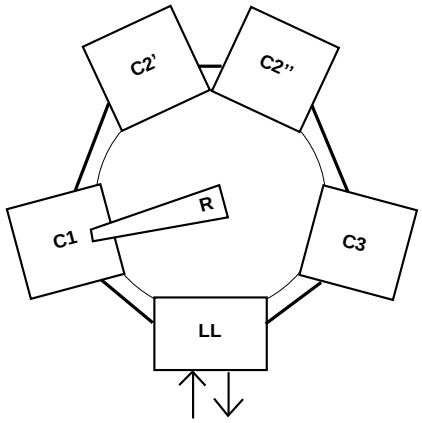

Fig.9. An outline of a tool with two chambers C2.

The Petri net model of a dual–blade cluster with two identical chambers C2, for steady–state behavior, is shown in Fig.10, in which the two chambers C2 are connected by a "selection loop" $(p'_{22}, t'_{2b}, p_{22}{''}, t_{2b}{''})$, similar to the one in Fig.8. 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.10 (after removing places p_1, p'_2, p''_2 and p_3) has six P–invariants, four invariants corresponding to the chambers, one representing the robot's cycle of operations, and one corresponding to the loop selecting the replicated chambers. The sets of transitions in the implied subnets are:

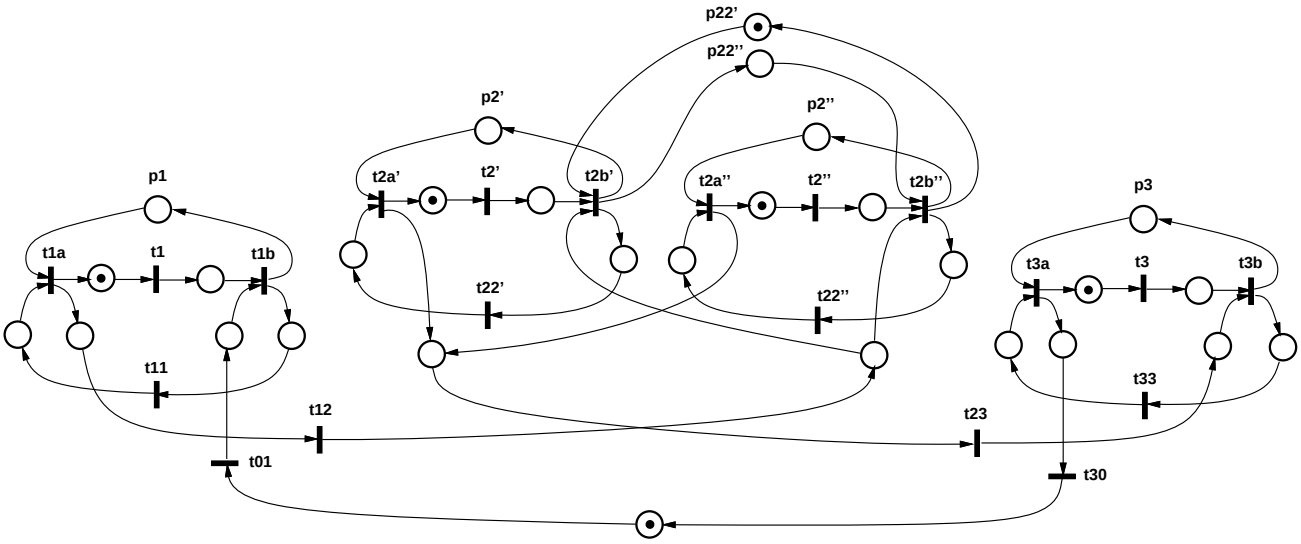

Fig.10. Petri net model for the steady–state behavior of a dual–blade 3–chamber tool with two chambers C2.

invariant	*set of transitions*
1	$t_1, t_{1a}, t_{1b}, t_{11}$
2	$t'_2, t'_{2a}, t'_{2b}, t'_{22}$
3	$t''_2, t''_{2a}, t''_{2b}, t''_{22}$
4	$t_3, t_{3a}, t_{3b}, t_{33}$
5	t'_{2b}, t''_{2b}
6	$t_{01}, t_{1a}, t_{11}, t_{1b}, t_{12}, t'_{2a}, t'_{22}, t'_{2a}$, $t''_{2a}, t''_{22}, t''_{2b}, t_{23}, t_{3a}, t_{33}, t_{3b}, t_{30}$

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. The frequencies of transition occurrences within each cycle are determined by T–invariants. The net shown in Fig.10 has only one T–invariant (i.e., the net is conflict–free), and the elements of this invariant are equal to 1 for t'_2, t''_2, t'_{2a}, t''_{2a}, t'_{22}, t''_{22}, t'_{2b} and t''_{2b}, and are equal to 2 for all remaining transitions. Furthermore, each cycle of the net corresponds to processing 2 wafers. Consequently,

$$\tau_0 \;=\; 0.5 * \max(\tau_1, \tau_2, \tau_3, \tau_4, \tau_6)$$

where τ_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 cycle of this model. Each of the cycle times τ_i is, in this case, a weighted sum of T–invariant components and occurrence times of the corresponding transitions:

$$\tau_1 \;=\; 2f(t_1) + 2f(t_{1a}) + 2f(t_{11}) + 2f(t_{1b}),$$

...

$$\begin{aligned}\tau_6 \;=\; & 2f(t_{01}) + 2f(t_{1a}) + 2f(t_{11}) + 2f(t_{1b}) + 2f(t_{12}) + \\ & f(t'_{2a}) + f(t'_{22}) + f(t'_{2b}) + f(t''_{2a}) + f(t''_{22}) + f(t''_{2b}) + \\ & 2f(t_{23}) + 2f(t_{3a}) + 2f(t_{33}) + 2f(t_{3b}) + 2f(t_{30})\end{aligned}$$

so, if the value of τ_2 is significantly greater than τ_1 and τ_4, and the tool is process bound (these are the reasons of introducing dual chamber C2), then the cycle time τ_0 can be as small as one half of the cycle time τ_2 (or τ_3), providing a very convincing justification for using two chambers C2.

7. Chamber Revisiting

In cluster tools with chamber revisiting, wafers pass through some chambers more than once. Coordinating the flow of wafers is more complicated in this case than for processing without chamber revisiting.

Similarly as before, the steady–state, cyclic behavior of a cluster tool can be described by a sequence of tool configurations that characterize the distributions of wafers in the chambers of the tool. However, in order to take into account chamber revisiting, the description needs to be extended. The extended configuration is a vector with components corresponding to all steps of the processing cycle, including the revisiting of (some) chambers. For example, if the sequence of processing steps is 1–2–3–4–2–3, which means that each wafer first visits C1, then C2, then C3 and C4, then revisit C2 and finally C3, the configurations are described by 6 variables, but some of these variables are "coupled" because they refer to the same physical chamber; for the sequence 1–2–3–4–2–3, variables 2 and 5 as well as 3 and 6 are coupled because they correspond to the first and second visits to C2 and C3, respectively. If any one of the coupled variables becomes non-zero, all remaining coupled variables become marked by "x" to indicate that the corresponding

chamber is not available. So, for an implementation of the process 1–2–3–4–2–3 with maximum concurrency, an initial configuration (i.e., a configuration just before loading a new wafer into the first chamber) can be (0,1,x,1,x,1) or (0,x,x,1,1,1); (0,1,1,1,x,x) is yet another initial configuration but it is of little interest because, after loading chamber C1, no further continuation is possible.

The possible changes of configurations can be described by the following rules.

- A configuration $(k_1, ..., k_{i-1}, 1, 0, ..., k_m)$ derives a configuration $(k_1, ..., k_{i-1}, 0, 1, ..., k_m)$; all variables coupled with variable $i+1$ become marked by x, and all variables coupled with variable i become 0.
- For the steady–state consideration, the cycle is assumed to begin with loading a new wafer into the first chamber; the starting configuration is thus $(0, k_2, ..., k_m)$, and this configuration always derives the configuration $(1, k_2, ..., k_m)$, moreover, all variables coupled with the first variable become marked by x.
- A configuration $(k_1, ..., k_{m-1}, 1)$ always derives configuration $(k_1, ..., k_{m-1}, 0)$; this change of configurations corresponds to unloading the wafer (after the last operation) and returning it to the loadlock.

For the 4–chamber tool and for the processing sequence 1–2–3–4–2–3, the sequence of configurations can be as follows:

configuration	*next operation*
(0,1,x,1,x,1)	pick new wafer and load it into C1,
(1,1,x,1,x,1)	the wafer from C3 is moved to LL,
(1,1,0,1,x,0)	the wafer from C2 is moved to C3,
(1,0,1,1,0,x)	the wafer from C4 is moved to C2,
(1,x,1,0,1,x)	the wafer from C3 is moved to C4,
(1,x,0,1,1,0)	the wafer from C2 is moves to C3,
(1,0,x,1,0,1)	the wafer from C1 is moved to C2,
(0,1,x,1,x,1)	the initial configuration.

For some configurations there may be more than one possible next operation, which leads to several different schedules with possibly different performances. It is also possible that a configuration has no possible operation, which indicates that the corresponding initial configuration leads to a deadlock. For example, for the previously discussed processing sequence 1–2–3–4–2–3, the initial configuration (0,0,1,1,0,x) leads to a deadlock:

configuration	*next operation*
(0,0,1,1,0,x)	pick new wafer and load it into C1,
(1,0,1,1,0,x)	the wafer from C1 is moved to C2,
(0,1,1,1,x,x)	deadlock.

Sequences of operations leading to a deadlock can easily be identified at the level of changes of configurations. Consequently, the deadlocks can be eliminated at a very early design stages.

The initial configuration (0,x,x,1,1,1) is acyclic, i.e., it is never repeated in the sequence of configurations which can be derived from it:

configuration	*next operation*
(0,x,x,1,1,1)	pick new wafer and load it into C1,
(1,x,x,1,1,1)	the wafer from C3 is returned to LL,
(1,x,0,1,1,0)	the wafer from C2 is moves to C3,
(1,0,x,1,0,1)	the wafer from C1 is moves to C2,
(0,1,x,1,x,1)	the previous initial configuration.

The model of the sequence of robot operations is derived from the sequence of configuration changes. For the processing sequence 1–2–3–4–2–3, the robot follows the cycle:

$$\text{LL} \Rightarrow \text{C1} \rightarrow \text{C3} \Rightarrow \text{LL} \rightarrow \text{C2} \Rightarrow \text{C3} \rightarrow \text{C4} \Rightarrow \text{C2} \rightarrow \text{C3} \Rightarrow \text{C4} \rightarrow \text{C2} \Rightarrow \text{C3} \rightarrow \text{C1} \Rightarrow \text{C2} \rightarrow \text{LL}.$$

For the case of chamber revisiting, the model of each (revisited) chamber becomes a free–choice structure. Such a structure is shown in Fig.11 for the case of two visits; similar paths represent the two visits (so the duration of each visit can be modeled independently).

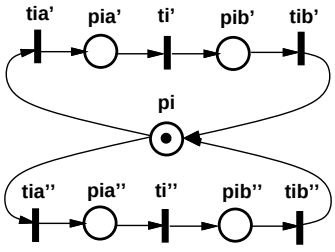

Fig.11. Petri net model of a chamber with two visits.

The complete model is shown in Fig.12. The 4 chamber models are represented by subnets associated with places p_1, p_2, p_3 and p_4. Places p_1 and p_4 can be removed (together with their incident arcs) as they do not contribute to the performance characteristics

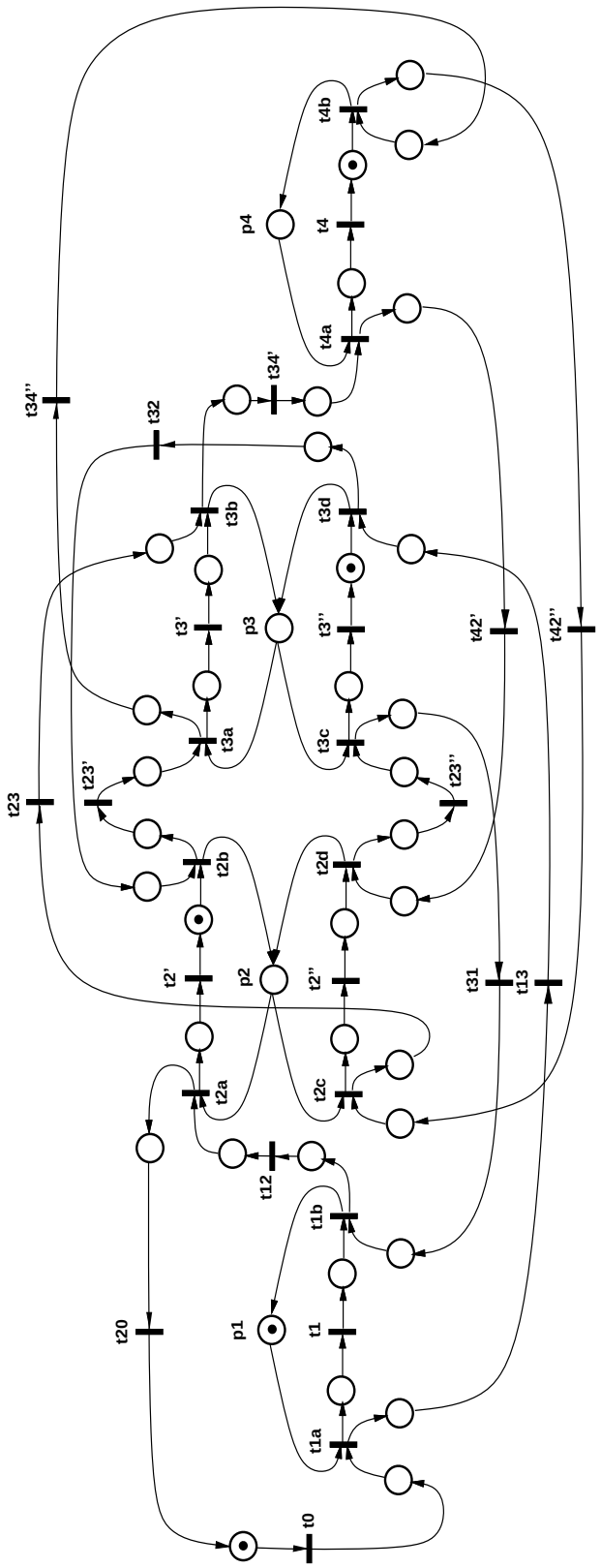

Fig.12. Petri net model of a 4–chamber tool with revisiting chambers C2 and C3.

of the model; they are preserved exclusively for the consistency of the representation.

The subnet representing the robot seems to be convoluted but it is quite straightforward to see its correspondence to the sequence of operations given

It should be observed that although the models of (revisited) chambers are free–choice subnets, the complete model is (dynamically) conflict–free; there is only one sequence of transition occurrences which represents the cyclic behavior of the model.

The operations represented by transitions in Fig.12 are shown in Tab.4 (using the same elementary operations as before).

Table 4: Operations represented by transitions in Fig.12.

trans.	*operations*	*exec time*
t_0	pick a new wafer and	
	move it to C1	$v+z$
t_1	perform C1 operation	o_1
t'_2	perform first C2 operation	o_{21}
t''_2	perform second C2 operation	o_{22}
t'_3	perform first C3 operation	o_{31}
t''_3	perform second C3 operation	o_{32}
t_4	perform C4 operation	o_4
t_{1a}	load C1	x
t_{1b}	unload C1	y
t_{2a}	load C2 (first visit)	x
t_{2b}	unload C2 (first visit)	y
t_{2c}	load C2 (second visit)	x
t_{2d}	unload C2 (second visit)	y
t_{3a}	load C3 (first visit)	x
t_{3b}	unload C3 (first visit)	y
t_{3c}	load C3 (second visit)	x
t_{3d}	unload C3 (second visit)	y
t_{4a}	load C4	x
t_{4b}	unload C4	y
t_{12}	move from C1 to C2	z
t_{13}	move from C1 to C3	$2z$
t_{20}	move from C2 to LL	$2z$
t_{23}	move from C2 to C3	z
t'_{23}	move from C2 to C3	z
t''_{23}	move from C2 to C3	z
t_{31}	move from C3 to C1	$2z$
t_{32}	move to LL, drop	
	the wafer, move to C2	$2z+w+2z$
t'_{34}	move from C3 to C4	z
t''_{34}	move from C3 to C4	z
t'_{42}	move from C4 to C2	$2z$
t''_{42}	move from C4 to C3	$2z$

After removal of places p_1 and p_3, the net shown in

Fig.12 has 14 basic place invariants, so the cycle time is equal to:

$$\tau_0 \;=\; \max(\tau_1, \tau_2, ..., \tau_{14})$$

If the chamber operations are denoted by o_i where i is the chamber number, and by o_{ij} for revisited chambers, where j is the visit number, the cycle times of the subnets implied by P–invariants are (some subnets contain more than one token, e.g., subnet 7, 8 or 9):

$$\begin{aligned}
\tau_1 &= v + w + 6x + 6y + 21z;\\
\tau_2 &= o_{32} + w + 4x + 5y + 13z;\\
\tau_3 &= o_{21} + 5x + 5y + 12z;\\
\tau_4 &= o_{22} + v + w + 5x + 5y + 17z;\\
\tau_5 &= o_{22} + o_{32} + w + 3x + 4y + 9z;\\
\tau_6 &= o_{21} + o_{22} + 4x + 4y + 8z;\\
\tau_7 &= (o_{22} + o_{31} + o_4 + v + 6x + 6y + 14z)/2;\\
\tau_8 &= (o_{22} + o_{31} + o_{32} + o_4 + w + 4x + 4y + 9z)/2;\\
\tau_9 &= (o_{21} + o_{22} + o_{31} + o_4 + 5x + 5y + 8z)/2;\\
\tau_{10} &= o_4 + 2x + 2y + 4z;\\
\tau_{11} &= o_{31} + v + w + 5x + 5y + 16z;\\
\tau_{12} &= o_{31} + o_{32} + 3x + 4y + 6z;\\
\tau_{13} &= o_{21} + o_{31} + 4x + 4y + 8z;\\
\tau_{14} &= o_1 + v + 2x + y + 4z.
\end{aligned}$$

The cycle time τ_1 (the only invariant with no chamber operations) corresponds to the robot's submodel, so if τ_0 is equal to τ_1, the model is "transport bound" and a different schedule might be considered to reduce the robot operations, otherwise the model is "process bound", which means that one of the chambers limits the performance of the tool.

8. Concluding Remarks

A systematic approach to modeling and analysis of a large variety of cluster tools has been described. 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 behavior, but transient behavior can also be analyzed using slightly modified models [25].

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; for example, a model of a 5–chamber tool will be similar to that shown in Fig.4, but will contain one more "chamber section" and a corresponding extension of the sequence of moves performed by the robot. This modular structure can be used to derive general performance characteristics for "standardized" tools, even without detailed analysis of their net models.

The performance of the derived models is obtained in symbolic form, in terms of modeling parameters (i.e., the occurrence times associated with transitions). Evaluations of different models with the same "structure" can be done very efficiently as they only require reevaluation of the derived symbolic formulas for different sets of modeling parameters (i.e., the occurrence times associated with transitions of the model).

The derived results also indicate whether the analyzed tool is process bound or transport bound. For process bound tools, the performance can be increased by introducing another level of currency through replication of the 'critical' chambers (i.e., the chambers which determine the performance of the entire tool). For transport bound tools, the performance can be increased by introducing additional concurrency in the form of multiple robots, each of which serves a subset of chambers, as discussed in [24].

Many simplifications were assumed in previous section in order to make the presentation as simple as possible. Many of these simplifications can easily be removed; for example, there is no need to assume that all the moves between the chambers require the same amount of time or that the load and unload operations for all chambers are the same, and so on.

Acknowledgements

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

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 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] M. Silva, E. Teruel, J.M. Colom, "Linear algebraic and linear programming techniques for the analysis of place/transition net systems"; in *Lectures on Petri Nets I: Basic Models* (Lecture Notes in Computer Science 1491), pp.309–373, Springer-Verlag 1998.

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

[19] 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.

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

[21] 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.

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

[23] 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.

[24] W.M. Zuberek, "Timed Petri net models of multi-robot cluster tools"; Proc. IEEE Conf. on Systems, Man, and Cybernetics (SMC'01), Tucson, AZ, pp.2729-2734, 2001.

[25] W.M. Zuberek, "Timed Petri nets in modeling and analysis of cluster tools"; *IEEE Trans. on Robotics and Automation*, vol.17, no.5, pp.562–575, 2001.

[26] 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.

[27] 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`.