Cluster Tools with Chamber Revisiting – Modeling and Analysis Using Timed Petri Nets

W.M. Zuberek

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

Abstract— **Timed Petri nets are formal models of discrete concurrent systems. Since the durations of all activities are included in the model descriptions, many performance characteristics can be derived from such models. In the case of cluster tools, net models represent the flow of wafers through the chambers of the tool as well as consecutive actions performed by the robotic transporter. Steady–state performance of cluster tools with chamber revisiting is investigated in this paper. A systematic development of detailed tool schedules, based on a general behavioral description of the tool, is proposed and is used to derive the corresponding Petri net models. Symbolic performance characteristics of the modeled tools are obtained by using place invariants, without exhaustive reachability analysis. Simple examples presented in the paper can be easily extended in many ways.**

Keywords— **Cluster tools, chamber revisiting, steady–state behavior, timed Petri nets, net invariants, performance analysis.**

I. INTRODUCTION

A CLUSTER tool is an integrated manufacturing system consisting of process, transport, and cassette modules, mechanically linked together [4]. The use of cluster tools has been stimulated in recent years by a number of factors which include higher yield [28], shorter cycle time (or higher throughput) [23], [28], [38], tighter process control [23] [38], better utilization of the cleanroom floor space [4], [38], reduced human intervention [11], [38], reduced working capital tied up in work–in–progress [25], and lower capital costs [11], [38]. The advantages of cluster tools are closely related to the trend of moving from processing batches of wafers to single–wafer processing – as the wafer size increases, maintaining the required process uniformity for the whole batch of wafers becomes increasingly difficult [11]. Basic concepts of cluster tools and their advantages over conventional fabs are discussed in [4], [11], [10], [23], [28], [38].

Performance comparisons of cluster–based fabs with conventional ones usually concentrate on two aspects, the throughputs and the costs. Results of such comparisons, obtained by computer simulation of fabrication processes [43], [44], indicate that cluster–based fabs can operate at considerably reduced throughput times for a relatively small increase in production cost per wafer. However, these results are sensitive to configurations and scheduling of cluster tools, so further research in these areas is needed. Moreover, simulation does nor reveal the relationships between tool characteristics and tool performance; simulation predicts *how* a tool performs, but does not explain *why* the tool performs in this way; analytical models are needed to provide more insight into such relationships.

Simple, intuitive models of the cycle time, throughput, and wafer cost of integrated single–wafer tools have been proposed in [41]. The models use two measurable parameters that aggregate tool operations: the *incremental cycle time*, which is the average increase in cycle time resulting from a lot size increment of one wafer, and the *fixed cycle time*, which is the portion of the cycle time that is independent of lot size. Analytical models of the incremental and fixed cycle times are used in [41] to study the effects of tool configuration on its performance.

Simple models of manufacturing processes are also used in [42] to analyze the effects of integrated single–wafer processing on fab cost and cycle time. Simulation results suggest that integrated single–wafer processing can reduce the cycle time of conventional fabrication by about 50 percent without having a significant effect of wafer production cost. However, tool integration and single–wafer processing must be used together to achieve these performance improvements.

Optimization of the throughput in cluster tools with different designs of the robotic transporter is discussed in [12]. Using simulation methods, the paper compares the throughputs of tools with single–blade (or single–pan) and two types of dual–blade robots, with same–side and opposite–side blades. The analysis of the throughput also takes into account the process times of modules; significant improvements of the throughput can be obtained by splitting the operation of a single module (which is a bottleneck) into two consecutive operations (the additional module performs a "preheat" operation in this particular case).

Throughputs and cycle times of systems of cluster tools with the same sequences of process steps and the same total numbers of modules, but in which the modules are grouped in different ways, are studied in [20]. The two extreme module configurations are called *serial* and *parallel* configurations. In serial configurations, each wafer visits each module in a cluster, but different process steps are performed concurrently on different wafers (in different modules). In parallel configurations, each cluster is composed of identical modules performing identical operations (on different wafers), so each wafer visits one of the modules in each cluster. Hybrid configurations are different combinations of parallel and serial configurations, e.g., each wafer is processed in several modules of several cluster tools. From reliability point of view, parallel configurations are better [21] (in a serial configuration, a failure of any one of the tool's modules makes the tool unavailable for further processing, while in a parallel configuration, a tool with a faulty module can continue its operation, although with reduced throughput); serial configurations, however, provide better throughputs, especially when the transport of wafers between the cluster tools is taken into account, and equipment downtimes are relatively infrequent. [21] concludes that the fabs will gradually migrate from parallel to serial configurations as cluster tools become more reliable and the cycle time becomes more important.

Because the manufacture of semiconductor devices is a highly complex and expensive process, it is often important to predict the operating characteristics of the fabrication

facility even before it is constructed. An open queueing network model for rapid performance analysis of semiconductor fabs is described in [7]. While the use of queueing models for performance evaluation of manufacturing systems is not new, the approach presented in [7] differs from others in the detailed models in which different tool groups are represented and in the way in which the effects of wafer rework and scrap are characterized. The solution follows the decomposition–based approximation approach, so each of dozens of nodes in the network is analyzed separately, with a set of renewal input processes capturing the interdependence among the nodes. Although [7] reports good conformance of obtained estimates with the results of simulation studies, the development of the queueing network model relied heavily on an existing simulation model developed earlier by IBM, and extensively validated and tested against actual IBM semiconductor lines. [7] concludes that a combination of queueing network models and simulation techniques is a more promising approach to performance prediction of semiconductor fabs. A general overview of application of queueing network models to modeling and performance prediction of wafer fabs, with some directions for future research, is given in [19].

More detailed representation of the behavior of cluster tools and analysis of their performances can follow the critical path approach, or an activity–oriented approach. The critical path approach is based on detailed timing diagrams, similar to Gantt charts, which represent one typical sequence of events in a cluster tool, and which derives the performance formulas from a critical path in the analyzed sequence. [29], [30] and [40] are good examples of this approach. The derived results are symbolic, so they capture the influence of different parameters on the performance of the tool, but the formulas are valid for only one, analyzed sequence of events, and the feasible values of parameters are restricted to this analyzed sequence of events. Any change of the sequence of events (which may be caused by a change of some parameters) requires a new analysis of the tool's behavior.

Activity–oriented approach considers cluster tools as (discrete) concurrent systems, in which several activities can be performed simultaneously, for example, different wafers can be processed by different modules at the same time, and also the robotic transporter can be moving to a position required by a future step [26]. Activity–based approach does not focus on any specific sequence of events; it takes into account the whole spectrum of possible behaviors of the analyzed system, so it is more general than the critical–path approach. Petri nets are formal models developed specifically for representation and analysis of concurrent activities [27], [34]. Since the use of Petri nets in modeling and analysis of manufacturing systems has become a "standard" approach [8], [9], [13], [31], [46], using Petri nets for analysis of cluster tools seems to be a natural consequence of their acceptance in other areas of manufacturing, robotics and systems automation. A broad survey of issues related to modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems using Petri nets is provided in [47].

Original Petri net models (sometimes also called condition–event systems) represent events and the causal relationships among them. In order to analyze the performance of such models, the durations of all activities must be included in the model description. Several types of nets "with time" have been proposed by associating "time delays" with places [36], or occurrence durations with transitions [2], [32], [48] of net models. Also, the introduced temporal properties can be deterministic [32], [33], [36], [48], or

can be random variables described by probability distribution functions (the negative exponential distribution being the most popular choice) [2], [3], [48]. For analysis of temporal constraints imposed on some events (e.g., a message must be received within x time units from the moment of sending), time intervals can be associated with places or transitions of net models [1], [24].

Two basic approaches to analysis of "timed" net models are known as *reachability analysis* and *structural analysis*. Reachability analysis is based on the detailed behavior of models, represented by the set of states and transitions between the states. For complex models, the exhaustive reachability analysis can easily become difficult because of a very large number of states (the so called "state explosion problem"). For some classes of net models, performance properties can be derived from the structure of the net models; this approach is called 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 the P–invariants).

One of first in–depth applications of timed Petri nets to analysis of cluster tools is [39]. Since the approach proposed there is based on the exhaustive analysis of the state space (i.e., reachability analysis), its applicability is restricted to models with rather small state spaces. Structural analysis of timed net models is used in [49], which is a comprehensive study of modeling and performance evaluation of a variety of cluster tools that includes single–blade and dual–blade tools, tools with redundant chambers (or modules), multiple loadlocks, multiple robots, and so on. A different temporal model, adopted from [5] and [16], is used in [17]. Intervals of time are used in this model to specify the lower and the upper bounds on the execution time of each operation. The upper bounds are used as time constraints, which limit the waiting times of wafers in (dual–arm or dual–blade) cluster tools. Schedulability (i.e., the existence of a schedule which satisfies all time constraints) for such models is determined by using linear programming to find a solution describing the behavior of derived net models. [17] also discusses a modification of the conventional swap operation of the dual–arm robot in which a wafer, that must wait for the availability of another module but cannot remain in the current module because the prolonged exposure to heat and/or chemical agents can create quality problems, is delayed on the robot's arm rather than in a module. A different approach to dealing with waiting time constraints of wafers (or residency constraints) is proposed in [35] where first an unconstrained schedule is considered, and if this schedule violates any residency constraints, some heuristic methods are used to modify the schedule (by increasing its cycle time).

Other applications of Petri nets to modeling and analysis of semiconductor manufacturing systems include [6], [14] and [15]. [6] uses a combination of timed colored Petri nets and genetic algorithms to optimize the performance of wafer fabrication systems. A timed colored net models all possible behaviors of the analyzed manufacturing system, and the genetic algorithm explores this space of behaviors with the objective of finding an optimal or near–optimal schedule for the manufacturing system. A special class of Petri nets, called RCN* merged nets, is used in [14] to represent fabrication system's degraded behavior, such as reworks, failures and maintenance. Structural methods, based on net siphons, are used to show the liveness (i.e., the absence of deadlocks and livelocks) and reversibility of the net models. [15] uses a subclass of generalized stochastic Petri nets (GSPNs) with priorities, called Markovian

timed Petri nets, to model semiconductor manufacturing systems with process priorities, routing priorities, resource re-entrance, and nonpreemptive operations. Lower and upper bounds on the performance of the analyzed system are obtained by using linear programming techniques.

Modern semiconductor devices are composed of many layers of different materials created in consecutive processing steps. Steps which involve similar technological processes can be performed in the same chambers of a cluster tool; the chambers are then "revisited" by wafers. Chamber revisiting can reduce potential inefficiencies of cluster tools but it complicates the detailed scheduling of operations within the tool. Because of complexity of modern cluster tools, a general approach is needed for systematic derivation of such detailed schedules. This paper proposes such a general approach.

The presented approach is an extension of previous work on modeling and analysis of cluster tools without chamber revisiting [49]. The paper shows that chamber revisiting can be systematically described by a set of discrete configurations which can also be used to derive the net model of the tool. This new approach can be applied to a large variety of tools, including single–blade and dual–blade ones, tools with multiple loadlocks, redundant chambers and multiple robots. Chamber revisiting, however, requires a slightly different models of chambers (to capture the revisiting process properly). Therefore, the paper first discusses how net models of cluster tools without chamber revisiting can be derived from their general discrete descriptions, and then extends this systematic approach to tools with chamber revisiting. Although the presented models of cluster tools are different than in [49], their performance analysis is also based on place invariants and place–invariant–implied subnets.

Section 2 recalls basic concepts of timed Petri nets; this material is available elsewhere, but is included in order to avoid misinterpretations due to subtle differences among many existing variants of Petri nets and temporal properties associated with them. Section 3 introduces simple models of steady–state behavior of single–blade cluster tools without chamber revisiting, and Section 4 presents performance analysis based on place invariants, as in [49]. Section 5 proposes a formal description of cluster tools with chamber revisiting. Systematic derivation of net models from the schedules is described in Section 6, which also illustrates the analysis of the derived model. Several concluding remarks are given in Section 7.

II. TIMED PETRI NETS

Petri nets are known as a simple and convenient formalism for modeling systems that exhibit parallel and concurrent activities [27], [34]. 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 structure $\mathcal{N}$ is a bipartite directed graph, $\mathcal{N} = (P, T, A)$, 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, $A \subseteq T \times P \cup P \times T$, 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 to all transitions sharing p are identical. A net is free-choice if all its shared places are free–choice. A 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 and 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 [48], 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 [2], [3] and time nets [24]). 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 an (average) 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 [48]. Only D–timed Petri nets are used in this paper.

Each place/transition net $\mathcal{N} = (P, T, A)$ can be conveniently represented by a connectivity (or incidence) matrix $\mathbf{C} : P \times T \to \{-1, 0, 1\}$ in which places correspond to rows, transitions to columns, and the entries are defined as:

$$\forall\, p \in P \,\forall\, t \in T \,:\, \mathbf{C}[p, t] = \left\{ \begin{array}{l} -1, \\ \quad \text{if } (p, t) \in A \wedge (t, p) \notin A, \\ +1, \\ \quad \text{if } (t, p) \in A \wedge (p, t) \notin A, \\ 0, \ \text{otherwise.} \end{array} \right.$$

Connectivity matrices disregard *selfloops*, that is, pairs of arcs (p, t) and (t, p). A pure net is defined as a net without selfloops [34].

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

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

A subnet $\mathcal{N}_I = (P_I, T_I, A_I)$ implied by a P–invariant I is obtained from a net $\mathcal{N} = (P, T, A)$ by selecting all those places, for which the corresponding elements of I are nonzero, $P_I = \{p \in P \mid I(p) > 0\}$ (P_I is sometimes called the support of the invariant I), and taking all arcs incident with these places in net $\mathcal{N}$ (part (1) of the definition), and then including all transitions which are associated with the included arcs (part (2) of the definition). It should be observed that all arcs incident with places in P_I belong to A_I but some arcs incident with transitions in T_I are usually left out.

There are efficient algorithms for finding all basic invariants of a net [18], [22].

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 (i.e., if each place and each transition of a net belongs to one of subnets), the cycle time of the net, τ_0, is equal to the maximum cycle time of the covering subnets [32], [36]:

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

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, 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 [49].

III. Simple Cluster Tools

The cluster tools discussed 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 [29], [49].

A sketch of a 4–chamber cluster tool (used as a running example) is shown in Fig.1, 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
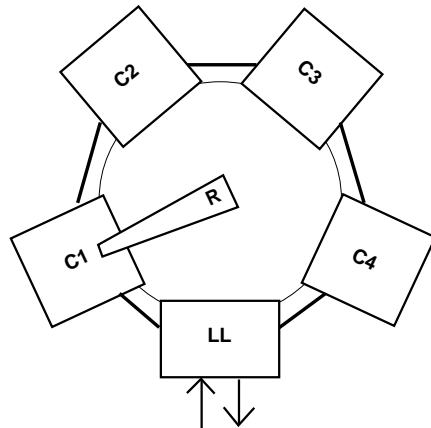

Fig. 1. An outline of a 4–chamber cluster tool.

the wafers between the loadlock and the chambers as well as from one chamber to another.

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 from the tool. 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 [29]: 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 supply 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 the maximally concurrent use of the chambers is shown in Tab.1.

Each change of configurations corresponds to a wafer moving from one chamber to another, from the loadlock

TABLE I

SEQUENCE OF CONFIGURATIONS FOR THE MAXIMALLY CONCURRENT
USE OF A 4–CHAMBER TOOL.

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.

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 $(k_1, ..., k_{i-1}, 1, 0, ..., k_m)$ allows moving a wafer from chamber C_i to C_{i+1}, so it always derives the configuration $(k_1, ..., k_{i-1}, 0, 1, ...k_m)$, $i = 1, ..., m - 1$.

• A configuration $(k_1, k_2, ..., 1)$ allows moving a wafer from the last chamber to the loadlock, so it always derives the configuration $(k_1, k_2, ..., 0)$.

• It is assumed that each cycle begins by moving a (new) wafer from the loadlock to chamber C1, so the first step always changes a configuration $(0, k_2, ..., k_m)$ 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. For the initial configuration $(0,1,1,0)$, 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 II

ALTERNATIVE SEQUENCES OF CONFIGURATIONS FOR A 4–CHAMBER
TOOL.

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 C4 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 C4 can be moved to the loadlock (step 5b);
5a : (0,1,1,1)	the wafer from C4 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.

The sequences of configurations for the initial configura-

tions $(0,1,0,1)$ and $(0,0,1,1)$ are derived in a very similar way; these sequences share some configurations with the sequences shown in Tab.2.

IV. MODELS OF SIMPLE CLUSTER TOOLS

The description of a cluster tool introduced in the previous section can easily be converted into a timed Petri net model of this tool. In this model, each chamber C_i is represented by a simple subnet shown in Fig.2. 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 represents 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}).
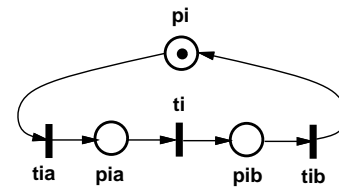

Fig. 2. Petri net model of a chamber.

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 steps of a single–blade robot (starting with moving the next wafer from loadlock LL to chamber C1):

LL $\Rightarrow$ C1 $\rightarrow$ C4 $\Rightarrow$ LL $\rightarrow$ C3 $\Rightarrow$ C4 $\rightarrow$
$\qquad\qquad$ C2 $\Rightarrow$ C3 $\rightarrow$ C1 $\Rightarrow$ C2 $\rightarrow$ 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 models of chambers and the robotic transporter can be combined into a complete model of the tool shown in Fig.3. The four chambers are represented (in the upper part of Fig.3) 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 indicates that the robot begins its (cyclic) sequence of operations by picking a wafer from the loadlock and moving to C1 (transition t_{01}), then the wafer is loaded 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; the behavior of the model is exactly the same if more than one token is assigned initially to p_0. Moreover, it can be observed that
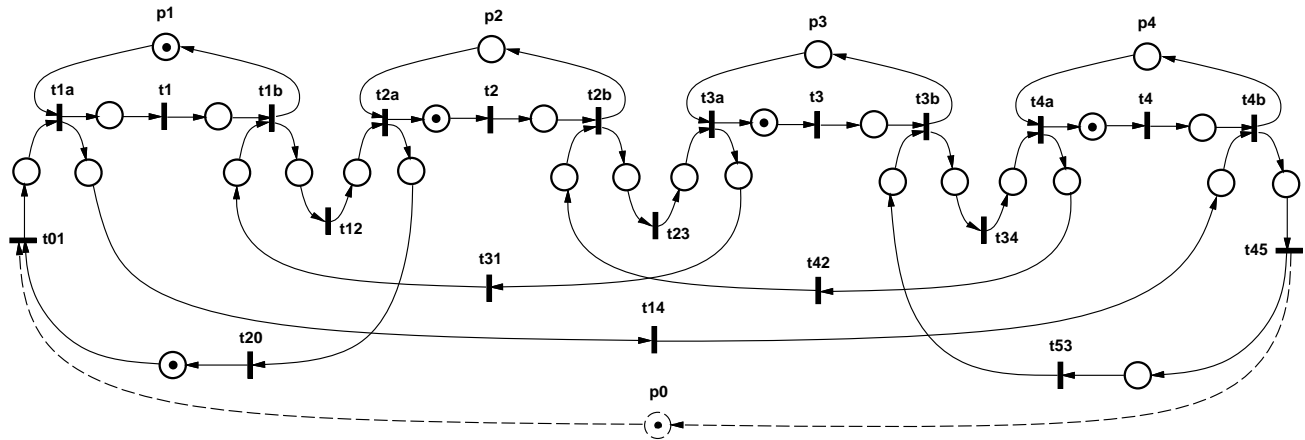
338

Fig. 3. Petri net model of a 4–chamber tool with a single–blade robot.

TABLE III

Operations represented by transitions in Fig.3.

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_{23}	move from C2 to C3;
t_{31}	move from C3 to C1;
t_{34}	move from C3 to C4;
t_{42}	move from C4 to C2;
t_{45}	move to LL and drop the wafer;
t_{53}	move from LL to C3.

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) [49]. 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 [37] such places are called "implicit places").

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

The net shown in Fig.3 has five basic P–invariants (after removal of places p_0, p_1, p_2, p_3 and p_4); the sets of transitions of subnets implied by these P–invariants are shown in Tab.4.

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:

TABLE IV

Sets of transitions of P–invariant–implied subnets in Fig.3.

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	$t_{01}, t_{12}, t_{23}, t_{34}, t_{45}, t_{14}, t_{20}, t_{31}, t_{42}, t_{53},$ $t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{4b}$

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

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, τ_5, the model is called "transport bound" [40].

The temporal characteristics associated with the transitions of the model can be determined by representing each step as a sum 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 and their execution times are shown in Tab.5.

The execution time of any operation is assumed to be the sum of execution times of actions constituting the operation. For the operations represented by transitions in Fig.3, these execution times are shown in Tab.6.

The cycle times of the subnets are obtained by adding the execution times (Tab.6) corresponding to transitions of the subnets:

TABLE V

ELEMENTARY ACTIONS AND THEIR EXECUTION TIMES.

exec time	action
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).

TABLE VI

EXECUTION TIMES ASSOCIATED WITH TRANSITIONS IN FIG.3

transition	exec time	transition	exec time
t_{01}	$v + z$	t_{12}	z
t_{1a}	x	t_{14}	$2z$
t_{1b}	y	t_{20}	$2z$
t_{2a}	x	t_{23}	z
t_{2b}	y	t_{31}	$2z$
t_{3a}	x	t_{34}	z
t_{3b}	y	t_{42}	$2z$
t_{4a}	x	t_{45}	$w + z$
t_{4b}	y	t_{53}	$2z$

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

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:

$$LL \Rightarrow C1 \to C3 \Rightarrow C4 \to C2 \Rightarrow C3 \to$$
$$C1 \Rightarrow C2 \to C4 \Rightarrow LL.$$

The net model derived from this sequence of operations is shown in Fig.4.

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:

The formulas describing the cycle times of this model can be derived similarly as for the model shown in Fig.3.

V. 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.

In steady–state, the cyclic behavior of a cluster tool can be described by a sequence of tool configurations where each configuration characterizes the distributions of wafers in the chambers of the tool. If chambers are not revisited

TABLE VII

SETS OF TRANSITIONS OF P–INVARIANT–IMPLIED SUBNETS IN FIG.4.

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	$t_{01}, t_{12}, t_{23}, t_{34}, t_{45}, t_{13}, t_{24}, t_{31}, t_{42},$ $t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{3a}, t_{3b}, t_{4a}, t_{4b}$

(Section 3), each configuration is a vector of m variables, with variable i describing the "status" (empty or not) of chamber C_i. For chamber revisiting, an extended description is needed, 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 revisits C2 and finally revisits C3, the configurations are described by 6 variables, but some variables are "coupled" as they refer to the same chamber. Each change to any one of such variables implies a change of all other variables which are coupled with it. 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 chambers C2 and C3, respectively. If any one of the coupled variable becomes non-zero, the corresponding chamber becomes unavailable, so all other variables coupled with the changing variable become marked by "x". 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 tool is "deadlocked").

The possible changes of extended configurations are described by the following rules:

• A configuration $(k_1, ..., k_{i-1}, 1, 0, ..., k_m)$ always derives 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.

• A configuration $(k_1, ..., k_{m-1}, 1)$ always derives configuration $(k_1, ..., k_{m-1}, 0)$; all variables coupled with variable m become 0. This change of configurations corresponds to unloading the wafer (after the last operation) and returning it to the loadlock.

• It is assumed that each cycle begins with loading new wafer into the first chamber; the starting configuration is thus $(0, k_2, ..., k_m)$, and this configuration always derives $(1, k_2, ..., k_m)$; all variables coupled with the first variable become marked by "x".

For the 4–chamber tool with process 1–2–3–4–2–3, the maximally concurrent sequence of configurations is shown in Tab.8.

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 cannot be (further) changed, 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 con-
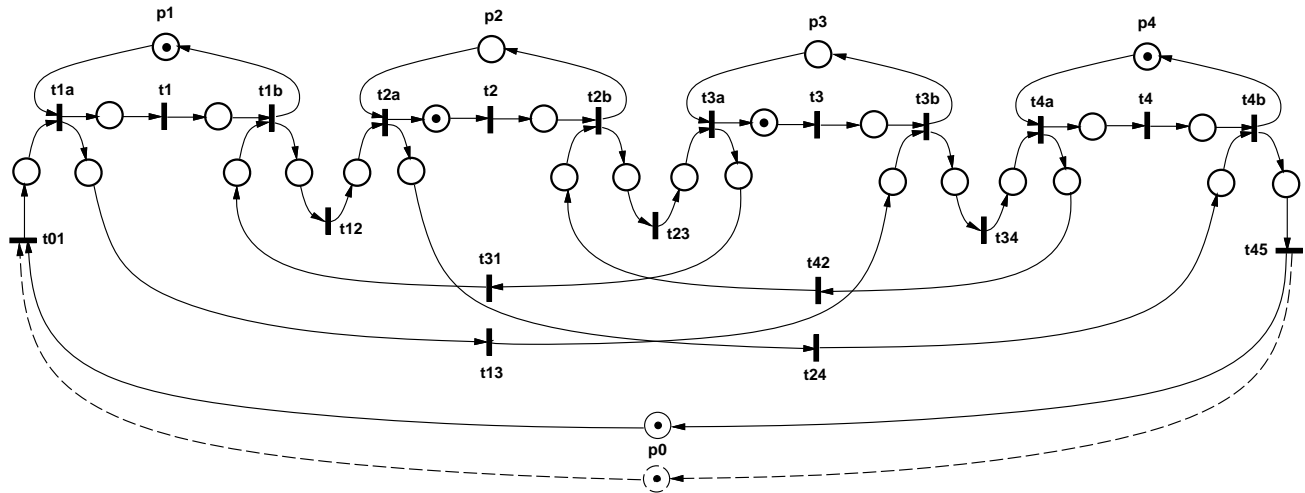
Fig. 4. Alternative Petri net model of a 4–chamber tool.

TABLE VIII

SEQUENCE OF CONFIGURATIONS FOR PROCESS 1-2-3-4-2-3.

configuration	next operation
(0,1,x,1,x,1)	pick new wafer and load into C1
(1,1,x,1,x,1)	unload C3 and return wafer to LL
(1,1,0,1,x,0)	unload C2, move and load into C3
(1,0,1,1,0,x)	unload C4, move and load into C2
(1,x,1,0,1,x)	unload C3, move and load into C4
(1,x,0,1,1,0)	unload C2, move and load into C3
(1,0,x,1,0,1)	unload C1, move and load into C2
(0,1,x,1,x,1)	the initial configuration.

TABLE X

ALTERNATIVE SEQUENCE OF CONFIGURATIONS FOR PROCESS 1-2-3-4-2-3.

configuration	next operation
(0,0,1,1,0,x)	pick new wafer and load into C1
(1,0,1,1,0,x)	unload C4, move and load into C2
(1,x,1,0,1,x)	unload C3, move and load into C4
(1,x,0,1,1,0)	unload C2, move and load into C3
(1,0,x,1,0,1)	unload C1, move and load into C2
(0,1,x,1,x,1)	unload C3 and return the wafer to LL
(0,0,1,1,0,x)	the initial configuration.

figuration (0,0,1,1,0,x) can lead to a deadlock, as shown in Tab.9.

derived configurations, as shown in Tab.11.

TABLE IX

DEADLOCKED SEQUENCE OF CONFIGURATIONS FOR PROCESS 1-2-3-4-2-3.

configuration	next operation
(0,0,1,1,0,x)	pick new wafer and load into C1
(1,0,1,1,0,x)	unload C1, move and load into C2
(0,1,1,1,x,x)	pick new wafer and load into C1
(1,1,1,1,x,x)	deadlock.

TABLE XI

ALTERNATIVE SEQUENCE OF CONFIGURATIONS FOR PROCESS 1-2-3-4-2-3.

configuration	next operation
(0,x,x,1,1,1)	pick new wafer and load into C1
(1,x,x,1,1,1)	unload C3 and return wafer to LL
(1,x,0,1,1,0)	unload C2, move and load into C3
(1,0,x,1,0,1)	unload C1, move and load into C2
(0,1,x,1,x,1)	the initial configuration of Tab.8.

If, however, in configuration (1,0,1,1,0,x), a wafer is moved from chamber C4 to C2 (for the second visit), the sequence can be continued, as shown in Tab.10.

All sequences of operations leading to deadlocks are easily identified at the level of changes of configurations and are eliminated from further considerations. Consequently, only deadlock–free sequences are analyzed with respect to their performance and the "best" sequence, i.e., the sequence with the maximum throughput, is chosen.

Some configurations are acyclic, i.e., they cannot be derived from itself. Obviously, such acyclic configurations cannot be used as starting configuration for the description of steady–state behavior. For example, the initial configuration (0,x,x,1,1,1) cannot be repeated in the sequence of

It should be noted that each cyclic sequence of configurations implies to a deterministic schedule of the robotic transporter. This schedule is obtained by "implementing" the consecutive changes of configurations and moving the robot from one chamber to another, as required by (consecutive) changes of configurations.

VI. NET MODELS AND THEIR ANALYSIS

As in Section 4, the general Petri net model of a cluster tool is composed of models of all chambers and the model of robot's schedule. Each chamber which is not revisited, is represented by a subnet shown in Fig.2. Models of revisited chambers are slightly more complex because they must provide different temporal characterizations for each visit.

The model is in the form of a free–choice structure with the number of choices representing the number of visits of the same wafer to this particular chamber (this number can be different for each chamber). Fig.5 shows a model of a chamber C_i for two visits; for each additional visit there is another cycle on place p_i.
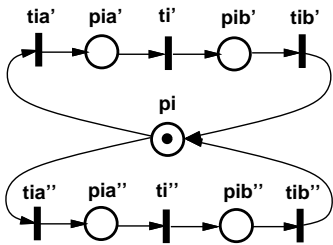

Fig. 5. Petri net model of a chamber with two visits.

In Fig.5, t'_{ia} and t''_{ia} represent chamber loading for the first and second visits, respectively, t'_i and t''_i represent chamber operations for the first and second visits, respectively, and t'_{ib} and t''_{ib} – chamber unloading for the first and second visits, respectively.

The model of the sequence of robot operations is derived from the sequence of configuration changes. For the sequence shown in Tab.8, the robot follows the cycle:

$$LL \Rightarrow C1 \rightarrow C3 \Rightarrow LL \rightarrow C2 \Rightarrow C3 \rightarrow C4 \Rightarrow C2 \rightarrow$$
$$C3 \Rightarrow C4 \rightarrow C2 \Rightarrow C3 \rightarrow C1 \Rightarrow C2 \rightarrow LL.$$

The complete model is shown in Fig.6. The 4 chambers are represented by (cyclic) subnets associated with places p_1, p_2, p_3 and p_4. The subnets for C2 and C3 are free–choice structures (as in Fig.5) with the upper branches representing the first visits and the lower branches representing the second visits of wafers. The initial marking indicates that, when a new wafer is picked from the loadlock to be loaded into C1 (say wafer k), C2 is visited for the first time (by the previous wafer, $k-1$), and C3 is visited for the second time (by wafer $k-3$); C4 is also loaded (with wafer $k-2$). This is the initial configuration $(0,1,x,1,x,1)$ from the previous section (Tab.8).

The "flow" of consecutive wafers, denoted "a", "b", "c", "d" and "e", through the chambers of the cluster tool can be represented by the following table in which the visits to the same chambers are indicated by subscripts:

C1	C2	C3	C4
–	c_1	a_2	b
d	c_1	a_2	b
d	c_1	–	b
d	–	c_1	b
d	b_2	c_1	–
d	b_2	–	c
d	–	b_2	c
–	d_1	b_2	c
e	d_1	b_2	c
...	...	...	...

The same flow of wafers is outlined in Fig.7 in a form of Gantt chart (with quite arbitrary durations of operations).

The subnet representing the robot seems to be convoluted but it is rather straightforward to see its correspondence to the sequence of operations given above; t_0 models picking a (new) wafer from the loadlock and carrying it to C1; t_{1a} represents loading the wafer into C1, after which
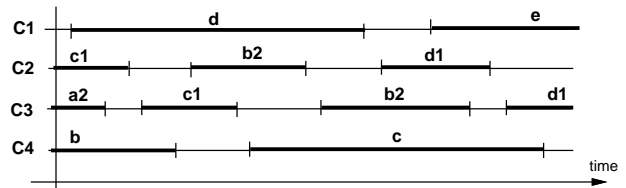

Fig. 7. A sketch of chamber occupancy times.

the robot moves to C3 (transition t_{13}) to unload the wafer (transition t_{3d}) and carry it to the loadlock, drop it there and move to C2 (all represented by transition t_{32}), then unload C2 (transition t_{2b}), and so on.

The operations represented by transitions in Fig.6, and their execution times (composed of a few elementary operations, as in Section 4), are shown in Tab.12 (for chambers with no revisiting, the chamber operation times are denoted by o_i where i is the chamber number; for chambers with revisiting they are denoted by o_{ij}, where i is the chamber number and j is the visit number).

In Fig.6, places p_1 and p_4 can be removed without any effect on the performance of the model. After removal of these two places, the net has 14 basic place invariants. Subnets implied by these invariants have the sets of transitions shown in Tab.13.

The cycle time is thus

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

where the cycle times of the implied subnets are obtained by adding the execution times associated with the transitions and dividing this sum by the total count of tokens in the subnet (if it is greater than one):

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

The cycle time τ_1 corresponds to the robot's submodel, so if τ_0 is equal to τ_1, the model is "transport bound" and a different schedule should be considered to reduce the robot operations, otherwise the model is "process bound" and one of the chambers limits the performance of the tool.

VII. Concluding Remarks

Realistic cluster tools are much more complicated than the one presented in this paper. Modern semiconductor devices are composed of many layers of different materials, with complex technological processes creating these layers in consecutive processing steps. Consequently, there are tens of processing steps, and the scheduling problems for such tools are correspondingly complex. The approach
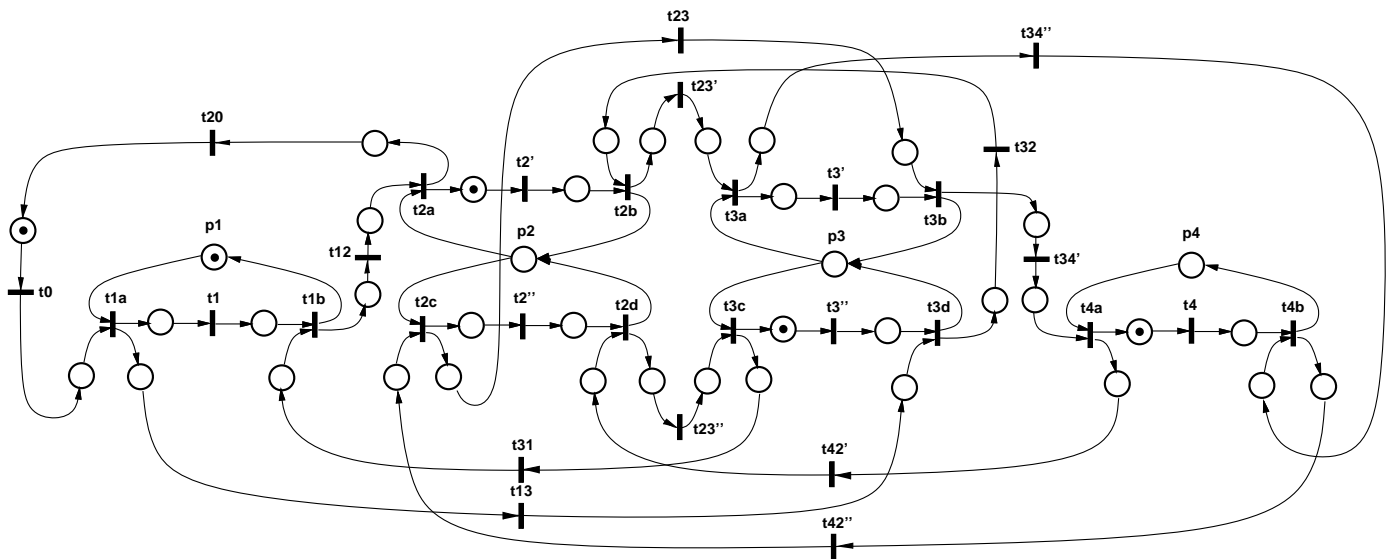
Fig. 6. Petri net model of a single–blade 4–chamber cluster tool for process 1-2-3-4-2-3.

described in this paper can be used for such complex cluster tools as it can easily be automated. The general rules describing the possible changes of configurations can be implemented as computer programs, deriving deadlock–free sequences of configurations and their Petri net models. Model simplifications, based on elimination of simple parallel paths (or implicit places) are also rather straightforward to implement. Finding basic place invariants is a standard feature supported by many tools for analysis of Petri nets [50]. Consequently, all these elements can be integrated into a single tool for modeling and performance analysis of a large class of cluster tools.

The solution discussed in this paper is derived with the assumption that maximum concurrency of chamber operations is required. The obtained results are relevant to the "process bound" case in which the operation times of the chambers are comparable (for chambers which are revisited, the total time for all visits is used), as outlined in Fig.7. If this is not the case, the most heavily used chambers could be duplicated to improve the performance of the whole tool. Chamber duplication can easily be taken into account in Petri net models [49].

The performance characteristics for steady–state behavior are derived in symbolic form, which provides a very efficient analysis of specific schedules, described by sets of numerical parameters. The steady–state model can be used for the estimation of the initial and final transient behaviors with only minor changes [49].

Only single–blade robots were discussed in this paper. For dual–blade robots, a slightly different approach is needed because the transportation of wafers from one chamber to another is done in a different way (the robot swaps the carried wafer with the wafer in a chamber). A net model of a 4–chamber cluster tool with a dual–blade robot is shown in Fig.8. After removing places p_1, p_2, p_3 and p_4, this net model has 5 place invariants, one for each chamber, and one for the robot. It should be observed that the times of operations associated with transitions in Fig.8 are different than for tools with a single–blade robot (a more detailed comparison of single–blade and dual–blade tools is given in [49]).

Chamber revisiting discussed is Sections 5 and 6 for cluster tools with single–blade robots applies as well to tools with dual–blade robots. The only difference is in the mod-

eling of the robot which, in dual–blade tools, has a slightly different schedule (as shown in Fig.8).

Only static scheduling of robot operations has been considered in this paper. Dynamic scheduling, i.e., scheduling during the operation of a cluster tool [45], can also use a high–level characterization of the tool's behavior in the form of tool configurations. For example, the dynamically scheduled cluster tool can maintain the current tool configuration and use a preselected "best" scheduling choices stored as a finite–state automaton (or just its transition function) within the tool. Deadlock avoidance, one of important aspects of dynamic scheduling, would be easily provided in such a case.

Acknowledgement

References

[1] W.M.P. van der Aalst, "Interval timed colored Petri nets and their analysis"; in *Application and Theory of Petri Nets 1993* (LNCS 691), pp.453-472, 1993.

[2] M. Ajmone Marsan, G. Conte, and 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.

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

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

[5] S. Calvez, P. Aygalinc, and W. Khansa, "P-time Petri nets for manufacturing systems with staying time constraints"; Proc. IFAC Conf. on Control of Industrial Systems, pp.1487-1492, 1997.

[6] J-H. Chen, L-C. Fu, M-H.Lin and A-C. Huang, "Petri net and GA-based approach to modeling, scheduling, and performance evaluation for wafer fabrication"; *IEEE Trans. on Robotics and Automation*, vol.17, no.5, pp.619-636, 2001.

[7] D.P. Connors, G.E. Feigin, and D.D. Yao, "A quequeing network model for semiconductor manufacturing"; *IEEE Trans. on Semiconductor Manufacturing*, vol.9, no.3, pp.412-427, 1996.

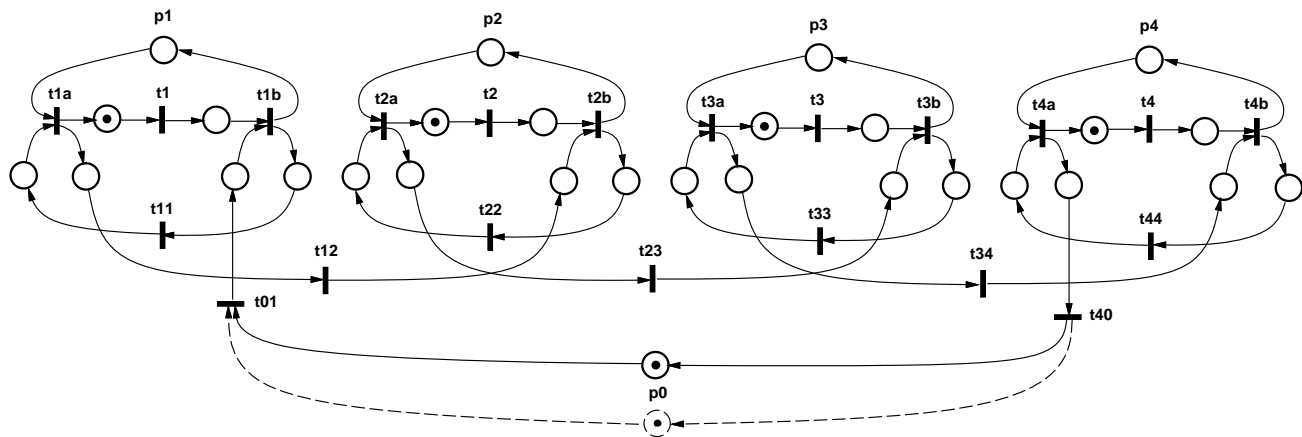[8] A.A. Desrochers and R.Y. Al-Jaar, *Applications of Petri Nets in Manufacturing Systems*; IEEE Press 1995.

Fig. 8. Petri net model of a 4-chamber cluster tool with a dual–blade robot.

[9] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat, *Practice of Petri Nets in Manufacturing*; Chapman and Hall 1993.

[10] B. Hansen, "Benefits of cluster tool architecture for implementation of evolutionary equipment improvements and applications"; in *SPIE Process Module Metrology, Control, and Clustering*, vol.1594, pp.83-91, 1991.

[11] J.R. Hauser and S.A. Rizvi, "Cluster tool technology"; in *SPIE Process Module Metrology, Control, and Clustering*, vol.1594, pp.45-54, 1991.

[12] R.A. Hendrickson, "Optimizing cluster tool throughput"; *Solid State Technology*, vol.40, no.7, pp.217-222, 1997.

[13] M.D. Jeng and F. DiCesare, "A review of synthesis techniques for Petri nets i with applications to automated manufacturing systems"; *IEEE Trans. on Systems, Man, and Cybernetics*, vol.23, no.1, pp.301-312, 1993.

[14] M-D. Jeng and X.Xie, "Modeling and analysis of semiconductor manufacturing systems with degraded behavior using Petri nets and siphons"; *IEEE Trans. on Robotics and Automation*, vol.17, no.5, pp.576-588, 2001.

[15] M-D. Jeng, X. Xie, and Y. Hung, "Markovian timed nets for performance analysis of semiconductor manufacturing systems"; *IEEE Trans. on Systems, Man, and Cybernetics, pt.B: Cybernetics*, vol.30, no.5, pp.757-771, 2000.

[16] W. Khansa, J-P. Denat, and S.D. Collart, "P-time Petri nets for manufacturing systems"; Proc. IEEE Int. Workshop on Discrete Event Systems, pp.94-102, 1996.

[17] J-H. Kim, T-E. Lee, H-Y. Lee, and D-B. Park, "Scheduling analysis of time-constrained dual–armed cluster tools"; *IEEE Trans. on Semiconductor Manufacturing,* vol.16, no.3, pp.521-534, 2003.

[18] F. Krueckeberg and 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.

[19] S. Kumar and P.R. Kumar, "Queueing network models in the design and analysis of semiconductor wafer fabs"; *IEEE Trans. on Robotics and Automation*, vol.17, no.5, pp.548-561, 2001.

[20] M.J. Lopez and S.C. Wood, "Systems of multiple cluster tools: configuration, and performance under perfect reliability"; *IEEE Trans. on Semiconductor Manufacturing,* vol.11, no.3, pp.465-474, 1998.

[21] M.J. Lopez and S.C. Wood, "Systems of multiple cluster tools: configuration, reliability, and performance"; *IEEE Trans. on Semiconductor Manufacturing*, vol.16, no.2, pp.170-178, 2003.

[22] J. Martinez and 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.

[23] T.K. McNab, "Cluster tools, pt.1: emerging processes"; *Semiconductor International*, vol.13, no.9, pp.58-63, 1990.

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

[25] D.J. Miller, "Simulation of a semiconductor manufacturing line";

[26] M.W. Moslehi, R.A. Chapman, M. Wong, A. Parandi, H.N. Najm, J. Kuhne, R.L. Yeakley, and C.J. Davis, "Single–wafer integrated semiconductor device processing"; *IEEE Trans. on Electronic Devices*, vol.39, no.1, pp.4-32, 1992.

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

[28] B. Newboe, "Cluster tools: a process solution"; *Semiconductor International*, vol.12, no.8, pp.82-88, 1990.

[29] T.L. Perkinson, P.K. MacLarty, R.S. Gyurcsik, and 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.

[30] T.L. Perkinson, R.S. Gyurcsik, and 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.

[31] J.M. Proth and X. Xie, *Petri Nets*; Wiley & Sons 1996.

[32] C.V. Ramamoorthy and 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.

[33] R.R. Razouk and 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.

[34] W. Reisig, *Petri nets – an introduction* (EATCS Monographs on Theoretical Computer Science 4); Springer-Verlag 1985.

[35] S. Rostami, B. Hamidzadeh, and D. Camporese, "An optimal periodic scheduler for dual–arm robots in cluster tools with residency constraints"; *IEEE Trans. on Robotics and Automation*, vol.17, no.5, pp.609-628, 2001.

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

[37] M. Silva, E. Teruel, and 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.

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

[39] 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.

[40] S. Venkatesh, R. Davenport, P. Foxhoven, and 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.

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

[42] S.C. Wood, "Cost and cycle time performance of fabs based on integrated single–wafer processing"; *IEEE Trans. on Semiconductor Manufacturing*, vol.10, no.1, pp.98-111, 1997.

TABLE XII

Execution times associated with transitions in Fig.6.

trans.	operations	exec time
t_0	pick a wafer and move 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	$w + 4z$
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 C2	$2z$

TABLE XIII

Sets of transitions of P–invariant–implied subnets in Fig.6.

invariant	transitions
1	$t_0, t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{3d},$ $t_{4a}, t_{4b}, t_{12}, t_{13}, t_{20}, t_{23}, t_{23}', t_{23}'', t_{31}, t_{32}, t_{34}',$ $t_{23}'', t_{42}', t_{42}'';$
2	$t_3'', t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{3d}, t_{4a}, t_{4b}, t_{23},$ $t_{23}', t_{23}'', t_{32}, t_{34}', t_{34}'', t_{42}', t_{42}'';$
3	$t_2', t_{1b}, t_{2a}, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{4a}, t_{4b},$ $t_{12}, t_{23}, t_{23}', t_{23}'', t_{31}, t_{34}', t_{34}'', t_{42}', t_{42}'';$
4	$t_0, t_2'', t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3c}, t_{3d},$ $t_{4b}, t_{12}, t_{13}, t_{20}, t_{23}', t_{23}'', t_{31}, t_{32}, t_{34}'', t_{42}'';$
5	$t_2'', t_3'', t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3c}, t_{3d}, t_{4b}, t_{23}', t_{23}'',$ $t_{32}, t_{34}'', t_{42}'';$
6	$t_2', t_2'', t_{1b}, t_{2a}, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3c}, t_{4b}, t_{12},$ $t_{23}', t_{23}'', t_{31}, t_{34}'', t_{42}'';$
7	$t_0, t_2'', t_3', t_4, t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3b},$ $t_{3c}, t_{3d}, t_{4a}, t_{4b}, t_{12}, t_{13}, t_{20}, t_{23}', t_{23}'', t_{31},$ $t_{32}, t_{34}', t_{42}'';$
8	$t_2'', t_3', t_3'', t_4, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{3d}, t_{4a},$ $t_{4b}, t_{23}', t_{23}'', t_{32}, t_{34}', t_{42}'';$
9	$t_2', t_2'', t_3', t_4, t_{1b}, t_{2a}, t_{2b}, t_{2c}, t_{2d}, t_{3a}, t_{3b}, t_{3c},$ $t_{4a}, t_{4b}, t_{12}, t_{23}', t_{23}'', t_{31}, t_{34}', t_{42}'';$
10	$t_4, t_{2c}, t_{3b}, t_{4a}, t_{4b}, t_{23}, t_{34}', t_{42}'';$
11	$t_0, t_3', t_{1a}, t_{1b}, t_{2a}, t_{2b}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{3d},$ $t_{4a}, t_{12}, t_{13}, t_{20}, t_{23}', t_{23}'', t_{31}, t_{32}, t_{34}', t_{42}';$
12	$t_3', t_3'', t_{2b}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{3d}, t_{4a}, t_{23}', t_{23}'',$ $t_{32}, t_{34}', t_{42}';$
13	$t_2', t_3', t_{1b}, t_{2a}, t_{2b}, t_{2d}, t_{3a}, t_{3b}, t_{3c}, t_{4a}, t_{12},$ $t_{23}', t_{23}'', t_{31}, t_{34}', t_{42}';$
14	$t_0, t_1, t_{1a}, t_{1b}, t_{2a}, t_{12}, t_{20}.$

Wlodek M. Zuberek received M.Sc. degree in Electronic Engineering, and Ph.D. and D.Sc. degrees in Computer Science, all from Warsaw University of Technology. Currently he is a Professor in the Department of Computer Science of Memorial University in St.John's, Canada. His research interests include modeling and performance analysis of concurrent systems, and in particular applications of timed Petri nets, discrete–event simulation, and 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.

[43] S.C. Wood and K.C. Saraswat, "The economic impact of single wafer multiprocessors"; in *SPIE Rapid Thermal and Related Processing Technique*, vol.1393, p.36, 1990.

[44] S.C. Wood and K.C. Saraswat, "Modeling the performance of cluster–based fabs"; *Proc. 1991 IEEE/SEMI Int. Semiconductor Manufacturing Science Symp.*, pp.8-14, 1991.

[45] N. Wu and M-C. Zhou, "Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems"; *IEEE Trans. on Robotics and Automation*, vol.17, no.5, pp.658-669, 2001.

[46] M-C. Zhou, *Petri Nets in Flexible and Agile Automation*; Kluwer Academic Publishers 1995.

[47] M-C. Zhou and M.D. Jeng, "Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: a Petri net approach"; *IEEE Trans. on Semiconductor Manufacturing*, vol.11, no.3, pp.333-357, 1998.

[48] 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.

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

[50] A database of tools for analysis of Petri net models is maintained at DAIMI, Department of Computer Science, Aarhus University, Denmark: `http://www.daimi.au.dk/PetriNets`.