8-th Int. Conf. on Emerging Technologies and Factory Automation, Juan–les–Pins, France, 15-18 Oct. 2001, pp.105-112.
Copyright © 2001 IEEE (DOI 10.1109/ETFA.2001.997676).

Petri Net Modeling and Performance Analysis of Cluster Tools with Chamber Revisiting

W.M. Zuberek

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

Abstract. **Timed Petri nets are convenient models of cluster tools as they represent the flow of wafers through the chambers of the tool as well as consecutive actions performed by the robotic transporter. Since the durations of all activities are also represented in such model, performance characteristics can be derived for steady–state as well as for transient behaviors. Steady–state performance of tools with chamber revisiting is investigated in this paper. A general description of cluster tools is proposed for systematic derivation of schedules, and a Petri net model is automatically derived from this description. The performance of the modeled system is derived by using place invariants, without exhaustive reachability analysis.**

I. INTRODUCTION

A cluster tool is an integrated manufacturing system consisting of process, transport, and cassette modules, mechanically linked together [3]. The factors which stimulate an increased use of clustered tools in recent years include improved yield and throughput, reduced contamination, better utilization of the floor space, and reduced human intervention [12].

Because of high throughput requirements, cluster tools perform a number of activities concurrently, for example, different wafers are processed in different chambers at the same time, and also the robotic transporter can be moving to a position required by the next step. Petri nets [9, 4] are formal models developed specifically for representation of concurrent activities and for their coordination, i.e., for ordering specific actions or for performing actions simultaneously by more than one component of a system.

In order to analyze the performance of modeled systems, the durations of all activities must also be taken ito account. Several types of nets "with time" have been proposed by associating "time delays" with places [10], or occurrence durations with transitions [1, 7, 16] of net models. Also, the introduced temporal properties can be deterministic [7, 8, 10, 16], or can be random variables described by probability distribution functions (the negative exponential distribution being probably the most popular choice) [1, 2, 16].

Analysis of timed net models based on their behavior (represented by the set of states and transitions between states) is known as reachability analysis. For complex models, the exhaustive reachability analysis can easily become difficult because of a very large number of states (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; state reduction methods eliminate all these states which are inessential for performance properties of the model. For some classes of net models, 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 (implied by the P–invariants).

Traditionally, performance of cluster tools was analyzed by using timing diagrams representing typical sequences of events, and deriving performance formulas from a critical path that determined the cyclic behavior of a tool [6, 5, 14]; such an approach is highly dependent on the analyzed cluster tool and its properties, and becomes quite complicated for tools which are complex. This paper presents an approach based on timed Petri nets which can be used to 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. Cluster tools with chamber revisiting are discussed in greater detail; the paper presents a systematic derivation of Petri net models for such tools and then performance analysis based on place invariants.

The approach presented in this paper is derived from earlier work on modeling and analysis of schedules for manufacturing cells [18].

Section 2 introduces simple models of steady–state behavior of single–blade cluster tools without chamber revisiting. Section 3 proposes a state descriptions of cluster tools which takes chamber revisiting into account and which is the basis for systematic derivation of possible schedules for cluster tools. Derivation of Petri net models from the schedules is presented in Section 4, and

several concluding remarks are given in Section 5.

II. SIMPLE CLUSTER TOOLS

The tools analyzed in this paper are m–chamber cluster tools with a single–blade 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 [5].

A sketch of a 4–chamber cluster tool 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 the wafers between the loadlock and the chambers as well as from one chamber to another.

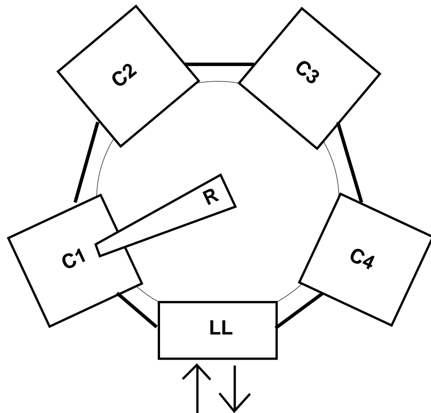

Fig.1. An outline of a 4–chamber cluster tool.

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 [5]: 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}, as sketched in Fig.2.

The initial transient period τ_{init} is due to the fact that the tool is empty at the beginning of each batch, and the final transient period τ_{end} processes the final wafers until the tool becomes empty. The steady part of the batch processing typically includes $n - m + 1$ identical cycles, where n is the number of wafers in the batch, and m – the number of chambers.

It is assumed that all chambers are used concurrently, i.e., when the i-th wafer is moved to chamber 1, the $(i-1)$-th wafer is processed in chamber 2, and $(i-2)$-th wafer is processed in chamber 3, and so on. The sequence of the operations in each cycle of the steady–state behavior is as follows (it is assumed that the cycle begins when a new wafer is moved to chamber 1, which must be empty at this stage):

- pick next wafer from loadlock, transport it to chamber 1 and load it into the chamber; chamber 1 can start its process;
- move to chamber 4, unload the wafer (when ready), transport it to loadlock and drop it there;
- move to chamber 3, unload the wafer (when ready), transport it to chamber 4 and load it into the chamber; chamber 4 can start its process;
- move to chamber 2, unload the wafer (when ready), transport it to chamber 3 and load it into the chamber; chamber 3 can start its process;
- move to chamber 1, unload the wafer (when ready), transport it to chamber 2 and load it into the chamber; chamber 2 can start its process;
- return to loadlock to begin another cycle.

A Petri net modeling this sequence of operations is shown in Fig.3.

The model shown in Fig.3 contains four sections modeling the four chambers, each represented by one transition (t_1, t_2, t_3 and t_4, respectively). Each of these transitions has one input and one output place to model the conditions "chamber is loaded" (so its operation can begin), and "chamber operation is completed" (so the wafer can be unloaded). The remaining part of Fig.3 represents the sequence of steps of one complete cycle of the robot. This sequence begins (as indicated by the initial marking) by picking a wafer from the loadlock (transition t_{01}). The initial marking of places p_{21}, p_{34} and p_{43} indicates that the chambers (except of chamber 1), in the moment of picking a new wafer from the loadlock, are loaded with (previous) wafers.

The operations represented by transitions are as follows:

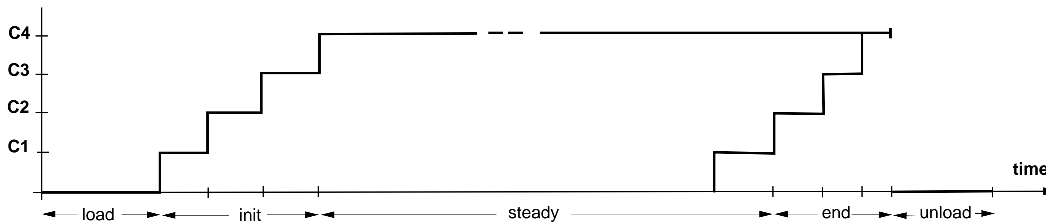

Fig.2. A sketch of batch processing.

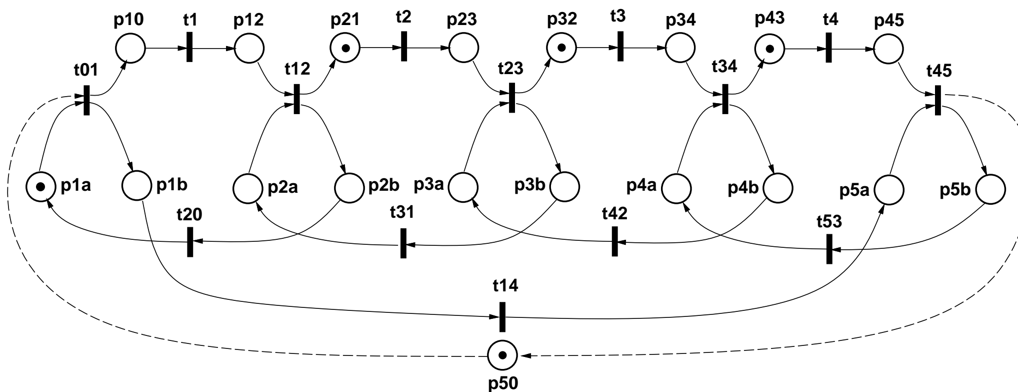

Fig.3. Petri net model for the steady-state behavior.

transition	*operations*
t_{01}	pick next wafer from the loadlock, move it to chamber 1 and load;
t_{14}	move the robot to chamber 4;
t_{45}	unload the wafer from chamber 4, move it to loadlock and drop;
t_{53}	move the robot to chamber 3;
t_{34}	unload the wafer from chamber 3, move it to chamber 4 and load;
t_{42}	move the robot to chamber 2;
t_{23}	unload the wafer from chamber 2, move it to chamber 3 and load;
t_{31}	move the robot to chamber 1;
t_{12}	unload the wafer from chamber 1, move it to chamber 2 and load;
t_{20}	move the robot to loadlock.

In order to obtain the effect of steady–state, place p_{50} is used as "input" and "output" of the cluster tool. When a wafer is finished, a token is deposited in p_{50}, and the same token is used as the next wafer a moment later. The initial marking of p_{50} 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_{50}. Moreover, it can be shown that this place has no effect on the performance of the model; if it is removed (with the two arcs shown in broken lines), the state graph of the simplified net is isomorphic to the state graph of the original neti shown in Fig.3.

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

The net shown in Fig.3, after removal of place p_{50}, has five basic P–invariants; the sets of transitions of subnets implied by these P–invariants are:

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

It can be observed that the first four P–invariants (and their implied subnets) correspond to simple cycles of consecutive machines and their load and unload operations. The last invariant represents the cyclic operations of the robot. Consequently, for net models of the type presented in Fig.3, the number of basic P–invariants increases linearly with the number of chambers, and the P–invariant–implied subnets can easily be predicted directly from the net model.

It is known [10] that if a net is covered by a family of conflict–free cyclic subnets, the cycle time of the net is equal to the maximum cycle time of the covering subnets:

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

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:

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

Since the P–invariant implied subnets cover the model shown in Fig.3, its 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 (all invariant–implied subnets contain exactly one token):

$$\begin{aligned}
&\tau_1 = f(t_1) + f(t_{01}) + f(t_{12}) + f(t_{20}),\\
&\tau_2 = f(t_2) + f(t_{12}) + f(t_{23}) + f(t_{31}),\\
&\tau_3 = f(t_3) + f(t_{23}) + f(t_{34}) + f(t_{42}),\\
&\tau_4 = f(t_4) + f(t_{34}) + f(t_{45}) + f(t_{53}),\\
&\tau_5 = f(t_{01}) + f(t_{14}) + f(t_{45}) + f(t_{53}) + f(t_{34}) +\\
&\qquad f(t_{42}) + f(t_{23}) + f(t_{31}) + f(t_{12}) + f(t_{20}).
\end{aligned}$$

If τ_0 is equal to any one of the first four terms, the model is called "process bound" because the duration of the process performed by the corresponding chamber determines the cycle time (and the throughput) of the tool; if the cycle time is equal to the last term, the model is called "transport bound" [14]. The performance of process bound tools can be improved by replicating the critical chambers, while for transport bound tools the performance can be improved by using multiple robots, each serving a subset of chambers.

III. TOOLS WITH 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.

The steady–state, cyclic behavior of cluster tool can be described by a sequence of tool configurations that characterize the distributions of wafers in the chambers of the tool. For a sequence of operation used in the previous section, such a description uses a vector of four variables corresponding to the four processing steps performed by the four chambers of the tool, each variable indicating whether the chamber is empty (value "0") or busy (value "1"). The initial configuration shown in Fig.3 is thus represented by vector [0,1,1,1], and loading of the next wafer into C1 changes this configuration to [1,1,1,1]. The only possible operation in this new configuration is to unload C4 (when its operation is finished), which changes the configuration to [1,1,1,0], so the next operation is to move the wafer from C3 to C4 (when C3 has finished its operation), and this creates configuration [1,1,0,1]. The remaining operations of the cycle create configurations [1,0,1,1] and [0,1,1,1], which completes the cycle of configurations.

For processing with chamber revisiting, such tool description needs to be extended to include the revisiting of chambers. 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 variables are "coupled" because they refer to the same 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 variable becomes non-zero, all remaining coupled variables become marked by "x" to indicate that the corresponding chamber is busy. So, for an implementation of the process 1–2–3–4–2–3, 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 new wafer into the first chamber; the starting configuration is thus $[0, k_2, ..., k_m]$, and this configuration always derives the configuration $[1, k_2, ..., k_m]$; 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 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.

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 into C1
[1,0,1,1,0,x]	unload C1, move and load into 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 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 previous initial configuration.

IV. PETRI NET MODELS

The general Petri net model is composed of models of all chambers and the model of robot. Each chamber is represented, similarly as in Fig.3, by a (timed) transition with one input and one output place. For revisited chambers, chamber model is slightly more complex because it should allow different temporal characterizations for each visit. Therefore it is in the form of a free–choice structure with the number of choices representing the number of visits of the same wafer to the particular chamber (this number can be different for each chamber).

The model of the sequence of robot operations is derived from the sequence of configuration changes. For the example presented in the previous section, the robot follows the cycle ($\Rightarrow$ indicates that the robot carries a wafer, and $\rightarrow$ that it moves empty):

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

The complete model is shown in Fig.4. The 4 chamber models are represented by subnets associated with places p_1, p_2, p_3 and p_4. The subnets for C2 and C3 are free–choice structures with the upper parts representing the first visits and the lower parts representing the second visits of the wafers. Places p_1 and p_4 could be removed (together with incident arcs) as they do not contribute to the performance characteristics of the models; 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 above.

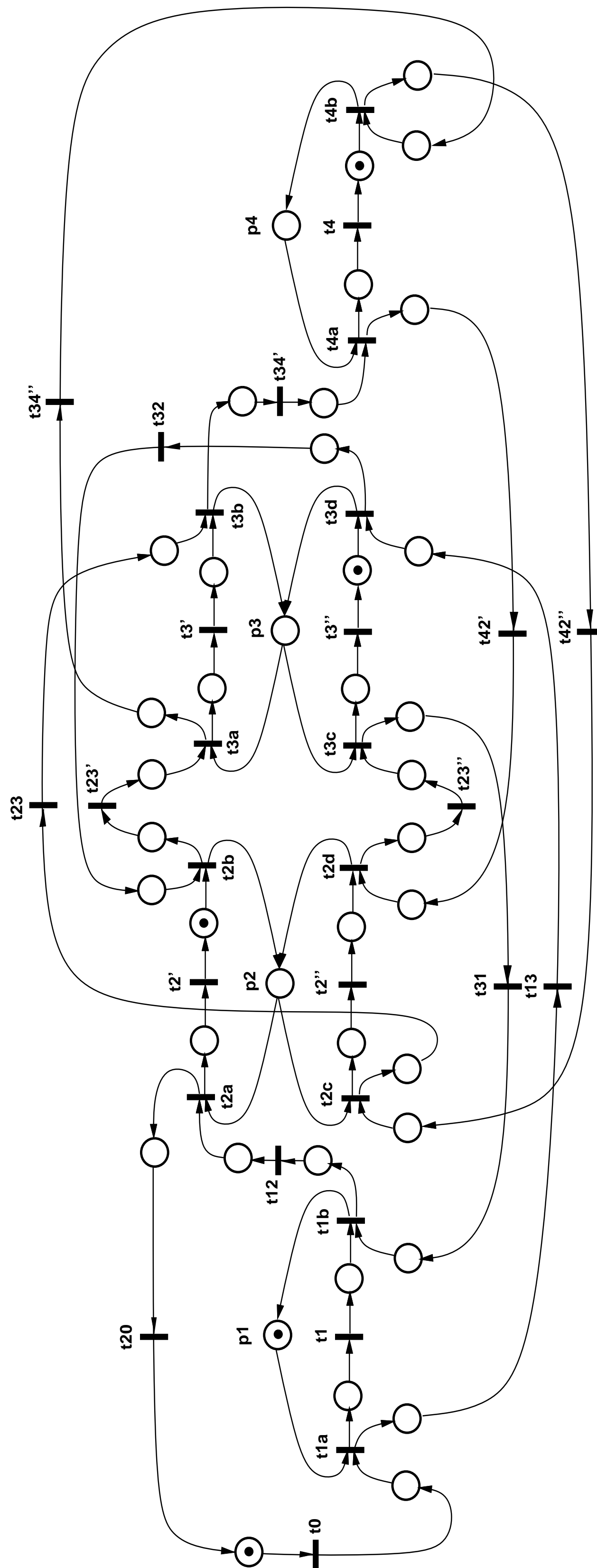

Fig.4. Petri net model.

In order to evaluate the cycle time of the model shown

in Fig.4, the durations of all operations must be associated with the transitions of the model. This duration times can be composed of just a few elementary operations, such as 'load a chamber', 'unload a chamber', 'move to the next chamber', and so on. In order to make the presentation as simple as possible, it is assumed that the loading and unloading times for all chambers (and for all visits to the same chambers) are equal, and that the time required for the move from one chamber to another is the same for all pairs of adjacent chambers, as well as for the move from the loadlock to the first chamber, and from the last chamber to the loadlock. Also, all other moves (between non–adjacent chambers) are simply composed of several elementary moves. Any changes of these assumptions can easily be taken into account by changing the occurrence times assigned to appropriate transitions.

There are five elementary operations:

v – time needed to pick a new wafer from the loadlock,
w – time needed to deposit a wafer in the loadlock,
x – time needed to load a wafer into a chamber,
y – time needed to unload a wafer from a chamber,
z – time needed to move between adjacent chambers.

The operations represented by transitions in Fig.4, and their execution times, are as follows (all chamber operation times are denoted by o_i where i is the chamber number, and by o_{ij} for revisited chambers, where j is the visit number):

trans.	*operations*	*exec time*
t_0	pick a new 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	$2z+w+2z$
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	$2z$

Places p_1 and p_4 in Fig.4 can be removed without any effect on the performance of the model (in [11] such places are called "implicit places"). After removal of these two places, the net shown in Fig.4 has 14 basic place invariants; subnets implied by these invariants have the following sets of transitions:

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	$t_0, t_1, t_{1a}, t_{1b}, t_{2a}, t_{12}, t_{20}$.

The cycle time is thus

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

and 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):

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

The cycle time τ_1 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.

V. 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 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 shown in Fig.5.

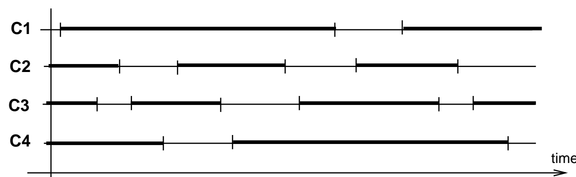

Fig.5. A sketch of chamber occupancy times.

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.

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 [17].

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). Consequently, the description of configurations and their changes must be different than the one presented in this paper.

Symbolic results derived from analysis of net models correspond directly to the fixed and incremental cycle time proposed in [15], where the (average) time required for processing a batch of n wafers is characterized by two parameters, τ_{fixed}, the 'fixed cycle time', and τ_0, the incremental time per one wafer during the steady–state behavior:

$$\tau_{batch} = \tau_{fixed} + n\tau_0$$

It should be noticed that the simple formula of [15] does not take the transient behaviors into account, so it underestimates the batch processing time.

VI. ACKNOWLEDGEMENT

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

VII. REFERENCES

[1] Ajmone Marsan, M., Conte, G., Balbo, G., "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] Bause, F., Kritzinger, P.S., "Stochastic Petri nets – an introduction to the theory" (Academic Studies in Computer Science); Vieweg Publ. 1996.

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

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

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

[6] Perkinson, T.L., Gyurcsik. R.S., MacLarty, P.K., "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.

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

[8] Razouk, R.R., Phelphs, C.V., "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.

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

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

[11] Silva, M., Teruel, E., Colom, J.M., "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.

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

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

[14] Venkatesh, S., Davenport, R., Foxhoven, P., Nulman, J., "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.

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

[16] Zuberek, W.M., "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.

[17] Zuberek, W.M., "Timed Petri net models of cluster tools"; Proc. IEEE Int. Symp. on Systems, Man, and Cybernetics (SMC'2000), vol.4, pp.3021-3026, 2000.

[18] Zuberek, W.M., Kubiak, W., "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.