# High-Fidelity Rapid Prototyping of 300mm Fabs through Discrete Event System Modeling*

Jonghun Park, Spyros A. Reveliotis†, Douglas A. Bodner,
Chen Zhou, Jennfong Wu, and Leon F. McGinnis
School of Industrial & Systems Engineering
Georgia Institute of Technology

**Abstract**

Traditionally, the semiconductor manufacturing industry has been driven by continuous technological advancement of the underlying production processes. Yet, as the industry matures, mere technology development is no longer sufficient. The effective deployment and exploitation of the system production capacity and operational capability become critical for competitive success. Hence, currently, there is an increasing interest towards the development of a control paradigm / framework that will allow the effective and efficient deployment and operation of contemporary fabs, including the upcoming 300mm fab. The research program presented in this paper seeks to define a detailed modeling and control framework for the real-time 300mm fab operations, by exploiting and integrating emerging results in Discrete Event Systems theory. The proposed approach is demonstrated through a small-scale example, modeling the operation of a 300mm fab bay. Furthermore, in addition to the development of the formal specification, the presented program will also implement the proposed fab modeling and control framework in a Web-based simulation platform, that will function as a fab (re-)configuration and control synthesis tool for the fab control engineer, and as an educational tool for manufacturing system modeling and control.

**Keywords: High-fidelity modeling, 300mm Fab, Flexible Automation, Colored Petri Nets, Web-based Simulation, Rapid Prototyping**

*A preliminary version of this paper appeared in [28].

†Corresponding author: phone #: (404) 894-6608, Fax #: (404) 894-2301, e-mail: *spyros@isye.gatech.edu*, mail address: 765 Ferst Dr., Atlanta, GA 30332

# 1 Introduction

Modern high-tech industry is characterized by a number of trends that affect the way in which manufacturers produce finished products. In today's environment, manufacturers achieve competitive advantage by offering a variety of product types, a high level of product quality, and short order lead times (i.e., the time from which an order is placed until the product is received). In addition, there is a constant pressure for innovative product design and customized products, which results in compressed product life cycles and the need for frequent facility reconfiguration to support the next generation of product. These trends contrast significantly with the traditional paradigm of mass production, which places its emphasis on efficient production of high volumes of a standardized and relatively stable product type.

The semiconductor industry is a prime example of such trends. Manufacturers produce a variety of semiconductor-based products or integrated circuits (e.g., memory chips or processors). At the same time, product quality is a critical consideration. Semiconductor material is sensitive to the slightest contamination; hence, manufacturers employ extensive automation to ensure contamination-free, or clean room, environments. To meet customer lead time demands, manufacturers focus on reducing cycle time - i.e., the time that it takes to produce a finished product once the raw material has been released for production - and also, cycle time variability. From a planning perspective, reduced customer lead times and increased system responsiveness require the continuous re-alignment of the planned production activity to externally imposed demand. Finally, semiconductor manufacturers are faced with the constant challenge of having to adapt and produce new types of integrated circuits, given the rapid pace of technological advancement in product design [26].

In the past, the semiconductor manufacturing industry has tried to address these market challenges primarily through *technological innovation*. For example, most major improvements in system performance have occurred from: (i) the stabilization of the production processes, which increases production yields, (ii) the development of integrated, highly controlled production environments [2], such as cluster tools [38], that reduce the time wasted in unproductive wafer transfers, and the risks arising from their exposure to environmental contaminants, and (iii) the continuous reduction in feature size, which increases the number of integrated circuits produced per wafer, and eventually, the system throughput. At the same time, there has been considerable effort to promote the *hardware-based integration* of the shop-floor activity. More specifically, the industry has actively pursued: (i) the deployment of powerful manufacturing execution systems (MES), that enable the real-time monitoring of the fab operations, and (ii) the development and adoption of standard operational models and protocols, like SECS / GEM and ARAMS, that facilitate the easy and seamless interface between supervisory controllers and tools [36, 13].

However, currently the industry is lacking a control paradigm that will take advantage of the aforementioned real-time shop-floor monitoring and command capability, and will provide an integrated framework for the efficient and holistic management of the flow of product, material, information, people, and other resources. Most prior attempts to address the real-time production control in the fab operational context have focused exclusively on the *"scheduling"* problem, i.e., the development of part-flow control policies, that when imposed on the system operation, lead to high production throughput through effective utilization of the system bottleneck resources. Typically, these policies are structured as a set of (i) *lot release (wafer-start)* and (ii) *lot dispatching* rules, heuristically developed for controlling the wafer flow among the different bays or bay processes [20, 23]. The efficiency of these rules is established / claimed on the basis of (i) accumulated past experience, (ii) simulation experiments in the particular fab context [31, 40], and, in certain cases, (iii) their proved optimality in the context of some simplifying analytical characterizations of the fab operations [19]. It is also noticed that some more involved scheduling schemes, attempting a more global view of the system operations, and a more systematic performance optimization, have been proposed in the literature [9, 37], but their effectiveness and implementability, especially in an extensively automated context like that contemplated for the 300mm fab, remains to be seen.

Indeed, in order to successfully deploy large-scale flexible automation, especially through the advent of 300mm wafer manufacturing technologies, new methods are needed for systematic characterization, analysis and control of the underlying system behavior, from the perspective of logical correctness and consistency [1]. The resulting problems are collectively characterized as the *logical / structural control* problem of the fab operations, and they can be perceived as complementary to the more traditional performance-related fab scheduling problem [7, 6]. Moreover, to the extent that structural control guarantees the correctness of the system behavior, it defines its admissible behavioral space, and therefore, any performance optimizing policy should be aligned with the *structural control policy (SCP)* requirements. Yet, in the past, logical / behavioral problems arising in the real-time control of the fab environments have been largely ignored during the system design and development, and they have been resolved mainly through external human intervention. More recently, there has been some effort to address these problems during the controller design phase, but the currently implemented policies seem to be completely customized to the particular fab operation, developed in an ad-hoc fashion, and validated only through custom-supported, simulated experimentation [31, 40]. Hence, there is no rigorous verification of the correctness of the developed control logic, and more importantly, the resulting controller cannot be easily reconfigured under dynamically changing circumstances. The bottom line is a user-unfriendly controller, characterized by high developmental costs and operational inflexibility.

Motivated from the above remarks, the work described in this paper seeks to develop a high-fidelity, rapid prototyping framework, appropriate for the systematic modeling, analysis and control of the real-time 300mm fab operations. Specifically, our work employs formal models - *Colored Petri Nets (CPN)* [16] - from the burgeoning field of *Discrete Event Systems (DES)* theory [8], in a hierarchical architecture, that allows the detailed modeling of the fab activity at the process tool, bay, and fab level. The developed models allow the representation of process tool-level behavioral aspects like batching, setups, reworks, processing errors, preventive maintenances, and machine failures.[1] Attention is also given to the high fidelity modeling of previously ignored operational aspects, including (i) the internal behavior of the inter and intra-bay material handling systems (MHS) (e.g. transporter failure or conflict-free transporter routing), (ii) effects resulting from the behavior of the supporting staff (e.g. maintenance personnel's scheduled break or random unavailability), and (iii) complex resource allocation dynamics due to the routing flexibility and auxiliary resource sharing. Kempf's mini-fab specification [17] was initially used for defining the detailed model specifications, and for assessing the representational adequacy of the proposed framework.

The formal characterization of the uncontrolled fab behavior subsequently lends itself to: (i) formal analysis through the DES theory, and (ii) the development of control policies that are able to establish the logical consistency / correctness and efficiency of the fab operations. Specifically, this part of the work leverages the aforementioned fab models in order to develop a control architecture for the real-time fab operations, addressing logical concerns - like deadlock-free buffer space allocation and/or collision-free vehicle traveling in the fab MHS's - as well as efficiency considerations - like production throughput maximization and cycle-time reduction. To this end, we employ recently emerged results from the areas of *structural control of resource allocation systems (RAS)* [34, 29], and *scheduling theory* [39, 20]. Our ultimate objective is the development of a *Web-based computational platform* that will facilitate the effective and efficient deployment of 300mm fab operations and their underlying control logic, through the integration of structured fab modeling, control synthesis, validation of the controlled system behavior, and evaluation of its performance, by means of available analytical techniques and simulation experiments. A pictorial representation of the undertaken research program and of the way in which it seeks to integrate and promote existing analytical results, is depicted in Figure 1.

The rest of the paper is structured as follows: Section 2 delineates the proposed 300mm fab modeling framework, discussing the underlying modeling principles, the employed modeling abstractions / entities, and their presumed relationships. It also maps the proposed fab behav-

[1] As it is expected that loading and unloading operations are fully automated in 300mm fab, the loading / unloading operator is not considered in the model presented in this paper. However, the model presented herein can be easily extended to account for this additional system element, by representing it as another system resource shared among the system processes.

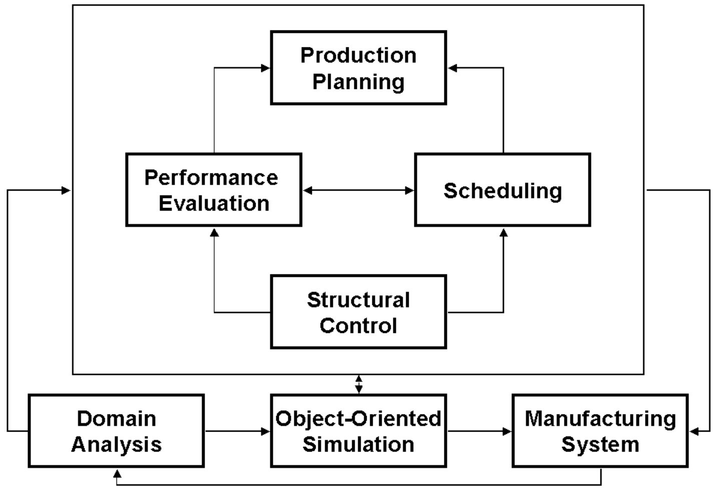


Figure 1: The structure of the undertaken research program

ioral models to a hierarchical control framework, consisting of a series of control sub-problems that correspond naturally to the basic fab modeling entities. Subsequently, it indicates how the controller synthesis for the identified control (sub-)problems can be supported through existing literature results, and how these partial solutions can be integrated to a seamless control framework for the entire fab operation. Section 3 concretizes the discussion of Section 2, by developing a CPN-based model for the bay-level operations currently contemplated for the emerging 300mm fab. Section 4 discusses the software architecture of the proposed Web-based simulator, and its intended functionality. Finally, Section 5 concludes the paper by discussing the current progress of our efforts, and the intended future work.

## 2 The Proposed 300mm Fab Modeling Framework and the Underlying Control Problem

The purpose of our work is to support *easy development* and *reconfiguration* of real-time controllers for the upcoming flexibly automated 300mm fab, and *assessment* of the controlled system *robustness* and *performance*, by focusing on the *logistics* of material flow and process deployment. Some specific principles that have driven our work are as follows: (i) the developed modeling methodology should require the smallest possible number of modeling primitives (*Occam's razor*); (ii) the developed modeling methodology should permit a high degree of model *configurability* and *customization*, in order to accommodate a wide variety of applications; (iii) the resulting models should map naturally onto real 300mm fab environments, i.e., *the abstraction gap between reality and the model should be as small as possible.*

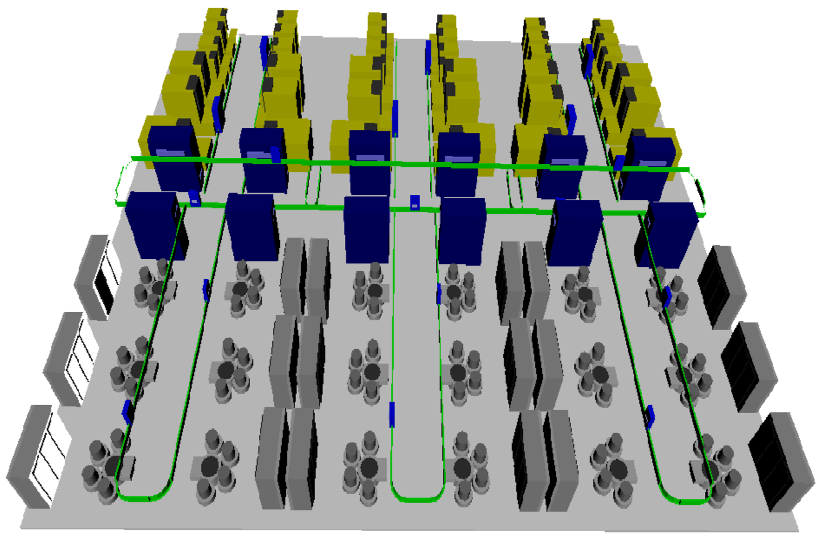

Figure 2: The typical layout for the emerging 300mm semiconductor fab

Manufacturing systems are often viewed as *hierarchical* organizations. Although this is not the only way to view such systems, currently it is the most conventional, and it applies naturally to the currently adopted fab layouts. As a result, it constitutes the basis of many existing control implementations, and it is the view adopted in this work. As a case in point, consider the fab layout illustrated in Figure 2. This layout is a typical layout contemplated for the emerging 300mm fab [11], and it consists of a number of *bays*, stemming from a *central aisle*, which gives it the characterization of a *spine* configuration. The central aisle contains the *interbay MHS*, typically a *monorail* or *overhead transport* system, transferring FOUPs (*Front Opening Unified Pod*) of wafers among the fab bays. The bays themselves are interfaced to the interbay MHS through the bay *stocker*, i.e., an automated storage / retrieval system, which accommodates wafer FOUPs waiting for processing in the bay area, or FOUPs that have completed processing in the current bay, and request transferring to some other bay. Within the bay, process tools are also organized around another automated material handling system, typically an *Automated Guided Vehicle (AGV), Rail Guided Vehicle (RGV)* or *Overhead Hoist Transport (OHT)* system, facilitating *intrabay* wafer transfers. Depending on the specific layout adopted, process tools in a certain bay may be aggregated on the basis of a common functionality (*farm* layout), or the support of a certain processing sequence (*modified hybrid* layout). Quite often, a third layout, known as *hybrid* layout, is obtained from the farm layout, by adding into each bay area the metrology tools required for evaluating the outcome of the bay processes.

For the modeling purposes pursued in this work, the fab layout illustrated in Figure 2 is viewed as having the hierarchical structure displayed in Figure 3. Specifically, our modeling methodology represents each element in the hierarchy as a *"control domain"*, with a particular structure. Furthermore, adopting the modeling perspectives of the prior OOSIM effort

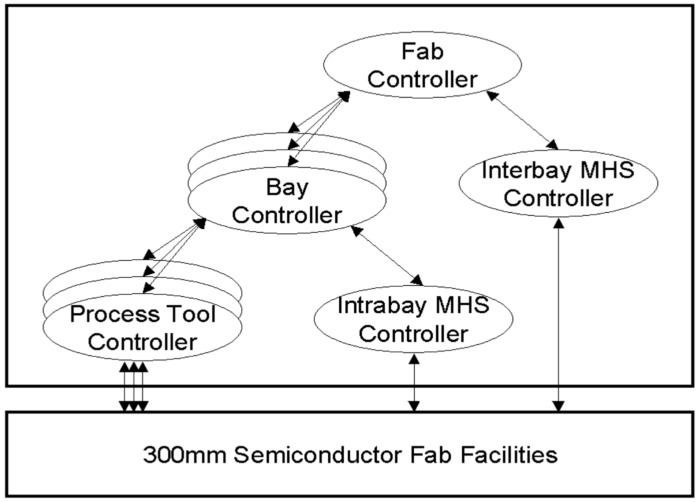


Figure 3: The proposed hierarchical modeling and control architecture

towards modeling discrete manufacturing systems [25, 5], the modeling entities characterizing the behavior of a certain control domain can be discriminated on the basis of (i) *processing* vs. *logistics*, and (ii) *plant* vs. *control*.

The fundamental abstractions supporting the *"plant"* view of the fab operations are material and location, defined as follows: (i) *Material* is a discrete job entity that is transported, stored and processed in the considered controller domains. A material unit has a unique *identity*, as well as handling and processing requirements, which constitute the *recipe* associated with it. Depending on the specific fab operational context, a material unit can correspond to a single wafer, or a FOUP collectively processed through the system as a lot. (ii) *Location* is a specific identifiable physical space where material may reside while it is being processed, transported or stored. Each location has a *capacity* (in terms of wafers or wafer FOUPs), and a *capability* (i.e., a set of functions that it can perform, each of which may have associated with it some processing time requirements). Each location is also characterized by a *state*, determined by: (a) its current content (e.g., a lot with a specific ID), and (b) its status (e.g., idle, busy, failed, etc.). Furthermore, each location has a *"behavior"* that can be characterized analytically. Hence, a process tool can be abstracted to a location that transforms wafers (material) from one state (prior to an operation) to another state (after an operation). An inspection station transforms material logically, by adding information about the condition of the material. Storage systems consist of locations that transform material in time. An overhead hoist or an AGV is a location that may contain a number of units of material, and transforms them in terms of both spatial and temporal state.

In our approach to fab modeling, the *control system* is represented by a set of entities that are distinct from their associated plant entities. These entities respond to changes in the state of material and locations of their associated plants, by initiating certain behaviors at the plant locations. The control system, then, is naturally described in terms of collections of locations that are managed by a specific controller. We will refer to such collections as *cells*. In general, a cell will contain two types of locations, those that transform the physical or information properties of the material, and those that transport or store the material. This distinction is captured in the *processing vs. logistics* view, in which the cell consists of two types of subsystems: (i) The cell *production sub-system*, i.e., the set of locations performing physical or logical transformations of material. These locations have an associated *operation-location map*, which specifies the set of production locations that can perform a particular operation, or step in a process plan / recipe. (ii) The cell *material handling system (MHS)*, i.e., the set of locations associated with storage of material and movement operations that transfer material between production locations, storage locations, and cell input/output locations. A *domain map* specifies the set of possible routes associated with each material handling device or location.

In the context of the fab operations illustrated in Figure 2, and depending on the level of abstraction in the fab modeling hierarchy, a cell can model the operation of a process or cluster tool, a bay area, or the entire fab. Its physical domain is defined by the stockers interfacing the cell production units to the inter-cell MHS, and the stocker owned by the cell itself. The cell controller coordinates the wafer flow within the cell production subsystem, i.e., the *induction* of new jobs into the cell, and their *dispatching* to the required cell production units. On the other hand, the internal operation of each of the cell production units and its MHS is controlled by dedicated production unit[2] and MHS controllers, respectively. These controllers are in subordinate relationship to the parent cell controller, in the sense that (i) their processed tasks are defined by *commands* generated by the parent cell controller, and (ii) they provide *feedback* information to it regarding the processing status of their task assignments. In turn, these lower level controllers supervise the operation of the hardware devices and logical system components in their domain, by exchanging a series of communication signals with the controller in these units. Hence, the proposed control scheme is characterized by a regularity that eventually reduces the control synthesis problem to the specification of: (i) the *generic internal control logic* for (a) the *equipment (process tool)-level controller*, that constitutes the leaf entities of the proposed command hierarchy, (b) the *intra-cell MHS controller*, and (c) the *cell-level controller*, as well as (ii) the *communication messages* exchanged between the cell controller, and its subordinate production unit and MHS controllers.

The development of a generic control logic for the three controller classes – i.e., the process-tool, MHS and cell-level controllers – can be further systematized by noticing that the control

[2]Depending on the complexity of the material flow taking place in these production units, the corresponding controllers can be basic workstation / process-tool or cell-level supervisors.

problem underlying each class can be abstracted to the problem of supporting the concurrent execution of a sequence of externally specified jobs, through logically correct and efficient allocation of the limited resources of the corresponding controller domain. Hence, for the purposes of the further analysis and controller design, it is pertinent to abstract each of the identified controller domains to a *resource allocation system (RAS)* [34, 35, 29]. This abstraction facilitates the application in the underlying controller design of more general methodological approaches proposed for DES analysis and control [8]. Specifically, according to the RAS theory developed in [34, 29], a RAS model is further represented as a *controlled DES plant* model, such as a *finite state automaton* [18] or a *Petri net* [12]. In this DES plant model, the RAS *state* is essentially the composite of the states of the locations contained in the RAS domain, while RAS *events* correspond to job initiations, terminations, and advancements between their processing stages, commanded by the domain controller to its subordinate controllers, which execute and confirm the corresponding task(s). From a physical standpoint, the material transfer among the different control sub-domains of a given cell controller- i.e., the units defining the cell production sub-system and the MHS - is facilitated through a set of locations modeling the buffering capacity of the access points of these units. Since these locations define the interface points between the considered control domain and its subordinate control domains, they are characterized as *shared locations*. Then, from an information-flow / logical perspective, the problem of establishing effective communication / coordination among the different system controllers can be posed as the requirement for *effective management of the shared locations interfacing the controller domains*. At the same time, it should be realized that for the DES plant modeling the behavior of the considered controller domain, the communication exchanges among the domain RAS controller and its subordinate controllers define the set of events that can be perceived by the RAS controller, and therefore, they determine its perception capabilities regarding its underlying operational domain, its discriminatory power, and eventually, the set of its potential behaviors. As previously noticed, the specification of (i) the messages to be exchanged between the various system controllers, and (ii) a supporting communication protocol that will ensure logically correct and robust behavior for the entire flexibly automated fab shop-floor, are among the main requirements of the presented research program.

The DES representation of the operation of the controller domains defined in Figure 3 is also instrumental for structuring the underlying decision making / control process. Figure 4 illustrates the proposed *generic control scheme*, in which the controller decision making process is triggered by the occurrence of *communication events* that either (i) *inform* the controller about changes taking place in its environment, or (ii) *command* actions imposed by the unit controller on its domain. All these events are reflected as changes in the perceived RAS state. Furthermore, the system controller reacts to a certain event by: (i) *evaluating the set of admissible actions among the set of feasible actions* associated with the perceived RAS state resulting

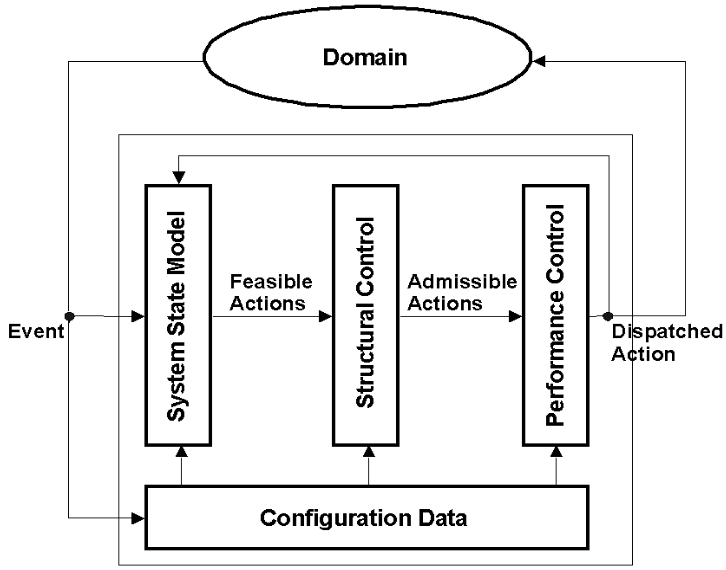


Figure 4: A generic, event-driven RAS control scheme [29]

from the communicated event, and (ii) *selecting a specific admissible action for immediate execution* by the control domain. This selection is reflected in the data structure representing the system state, and the entire cycle repeats itself.

The first part of the generic RAS control logic indicated in Figure 4 – i.e., determining the admissibility of any feasible control action in a given RAS state – is characterized as the (applied) *structural control policy* (SCP). Currently, the most conspicuous problem addressed by structural control in contemporary flexibly automated production environments is the resolution of the potential system *deadlocks* arising from the competition of a number of concurrently executing jobs for the finite set of the system resources. Computationally efficient policies that have been rigorously shown to establish deadlock-free system operation under a series of RAS behavioral contexts, can be found in [34, 35, 21, 22, 32, 29]. More recently, the work presented in [30] has extended and integrated these results so that they apply in a distributed real-time control architecture. Specifically, [30] uses the formalism of *communicating colored Petri nets (CPN)* in order to characterize the underlying domain / RAS behaviors, and the imposed structural control logic. With this preliminary result, in this paper, we extend the basic RAS models defined in [30], in order to accommodate additional operational features pertaining to the 300mm fab operations, including (i) the coupling of the local process dynamics due to *auxiliary resource (reticle) sharing* (in addition to the finite process buffering capacity), (ii) *batch*-based processing at certain process tools, (iii) *sequence dependent setups*, (iv) *processing errors* and *rework* requirements, (v) *preventive maintenance*, and (vi) process tool *failures*.

The second part of the generic RAS control logic –i.e., the selection of a specific action among the set of admissible actions to be commanded for immediate execution – constitutes the performance-oriented scheduling policy implemented by the controller. Most dispatching / scheduling policies existing in the literature (e.g., the *dispatching rules* of [23], the *distributed scheduling policies* of [19], and the more computationally involved *tracking policies* of [37]) can be easily incorporated in the context of the control scheme depicted in Figure 4. However, some recently obtained results [33] indicate that the effectiveness and efficiency of the aforementioned scheduling policies in the contemporary flexibly automated fab is an open research issue. We believe that the representation of the fab operations developed in this paper, and the associated control architecture, provide the appropriate context for addressing the underlying performance evaluation and control problem. Yet, the resulting control problem is essentially equivalent to the notorious *"job-shop scheduling"* problem [24], reformulated in the operational context of the proposed framework, and therefore, its consideration is deferred to future work. In the meantime, the high-fidelity Web-based simulator of the real-time 300mm fab operations, outlined in Section 4, should function as a compromising solution towards the systematic fab performance evaluation, and the synthesis of efficient, heuristic-based fab scheduling policies.

# 3 DES-based Modeling of a 300mm Bay Using Colored Petri Nets

In this section, we demonstrate the efficacy of the proposed modeling framework through the presentation of a high-fidelity CPN model for a small-scale 300mm bay. According to the hierarchical controller decomposition framework defined in Section 2, the CPN model is presented in terms of *process tool, intrabay MHS*, and *bay* CPNs. For each constituent CPN, first we provide a generic CPN structure which accommodates all possible behavioral characteristics of the 300mm fab considered in this paper, and subsequently show how it is tailored to the considered example, by defining appropriate colors. More specifically, each of the following three subsections (i) presents the structure of the corresponding generic CPN, (ii) provides a concrete CPN instantiation for the example, and (iii) discusses how the detailed operational characteristics considered in the example are accommodated in the CPN model. The subsequent discussion presumes that the reader is familiar with the CPN modeling framework. A brief introduction to CPN-based modeling is provided in the Appendix.

## 3.1 Description of the Example 300mm Bay

An example 300mm bay is depicted in Figure 5. It consists of two process tools, $p_1$ and $p_2$, and two bay stockers, $s_1$ and $s_2$, all inter-connected by an intrabay single-vehicle AGV system, configured as a loop layout. Process tool $p_1$ ($p_2$, respectively) has two (one, respectively)

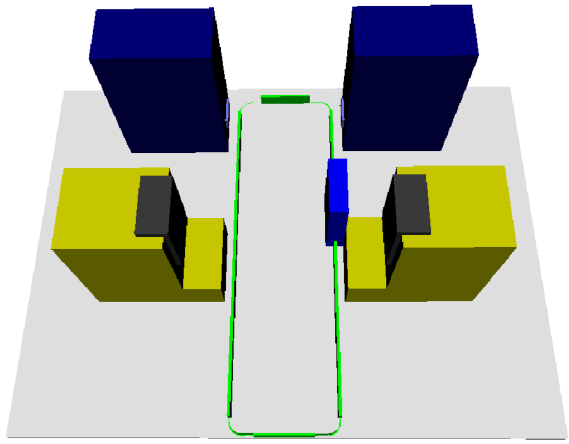

Figure 5: An example 300mm bay

loadport(s) where a FOUP can be loaded / unloaded by the AGV. Bay stocker $s_1$ is for storage and retrieval of wafer FOUPs, whereas stocker $s_2$ supports the storage of the auxiliary tool(s).[3] Each stocker also has a loading / unloading port dedicated to its interfacing with the bay AGV system. The AGV transfers, both, FOUPs and auxiliary tools to the bay process-tools, but it can carry only one FOUP or auxiliary tool at a time.

We consider two lot types, $w_1$ and $w_2$, each of which requires two serial stages of processing, denoted as $w_{11} \rightarrow w_{12}$ and $w_{21} \rightarrow w_{22}$, respectively. Lots waiting for the initiation of their processing are said to be in the initial *idle* stage, which is respectively denoted by $w_{10}$ and $w_{20}$, for each of the two job types. Stages $w_{11}$ and $w_{22}$ require process tool $p_1$ for their successful execution, while stages $w_{12}$ and $w_{21}$ require process tool $p_2$. Furthermore, stages $w_{11}$, $w_{21}$, and $w_{22}$ also require an auxiliary tool, $t_1$, in addition to the corresponding process tool. In order to accommodate prevailing fab practices, we allow that stocker $s_1$ may be selected as a next possible destination after stages $w_{11}$ or $w_{21}$; the resulting stages are denoted as $w_{1s}$ and $w_{2s}$, respectively.

Processing may also result in error, requiring a rework. For each processing stage, $w_{ij}$, $i, j \in 1, 2$, the corresponding error stage is represented as $w_{ije}$. For the sake of presentational brevity, we assume that the rework is performed at the same process tool where the error has occurred. Furthermore, process tool $p_1$ is a batch processing unit, requiring one instance from two different lot types, $w_1$ and $w_2$, to be present at the loadports, in order to start processing. On the other hand, process tool $p_2$ experiences sequence-dependent set-up times. Finally, the process tools and the AGV are prone to failure, and preventive maintenance is performed on the process tools according to a pre-determined schedule. However, maintenance can only be

[3]For instance, it is customary that, in lithography bays, expensive reticles are shared by a number of process tools.

| Data type name | Interpretation | Elements |
|---|---|---|
| LN | Lot no. | Natural numbers |
| LS | Lot stage | $w_{ij}, w_{is}$, $i = 1, 2$, and $j = 0, 1, 2$<br>$w_{ije}$, $i = 1, 2$, and $j = 1, 2$ |
| B | Port | $b_1, b_2, b_3, b_4, b_5$ |
| C | Logical resource | $c_1, c_2$ |
| M | Mode | $m_s, m_e, m_f$ |
| P | Process tool availability | $p_1, p_2$ |
| $E_F$ | Failure / repair event | $e_f, e_r$ |
| $E_M$ | Maintenance event | $e_{ms}, e_{mf}$ |
| $E_S$ | Stocker event | $e_{rr}, e_{rc}, e_{sr}, e_{sc}$ |
| $E_T$ | Intrabay transport system event | $e_{tr}, e_{tc}$ |
| $E_A$ | Auxiliary tool system event | $e_{ar}, e_{ac}$ |
| $E_V$ | Vehicle event | $e_{is}, e_{if}, e_{os}, e_{of}, e_{ls}, e_{lf}, e_{us}, e_{uf}$ |

Table 1: The basic data types employed and their interpretation

started when a process tool is idle.

The data types providing the basis for the definitions of the CPN models developed in this example, are listed in Table 1. We notice that elements $b_1$ and $b_2$, appearing in Table 1, correspond to the ports of process tool $p_1$. Element $b_3$ corresponds to the port of process tool $p_2$, while elements $b_4$ and $b_5$ respectively denote the ports of stockers $s_1$ and $s_2$. The detailed semantics of the remaining elements of Table 1 will be defined in the discussion of the following subsections.

### 3.2 Process Tool CPN

The *process tool* CPN, depicted in Figure 6, represents the behavior of a single equipment unit in the 300mm fab. Specifically, this net models the various functional / behavioral aspects that can characterize the operation of a fab process tool, including batching, sequence dependent setups, tool failure, processing error / rework, and preventive maintenance. Furthermore, the presented net models also the communication of the corresponding equipment controller with its supervisory cell (bay) controller, through the mechanism of *communication places*. These places hold tokens that correspond to *messages* exchanged between the process tool controller and its supervisory bay controller(c.f., Appendix), and they are represented as *double circles* in Figure 6.

The arrival of a new lot at the process tool, requiring the execution of some processing stage, is represented by the presence of a token at place $LWP$. If the process tool is currently idle (represented by the token availability in $PTA$) and the batching condition is satisfied (represented by the color set of transition $BPS$), processing can be initiated, and the tokens

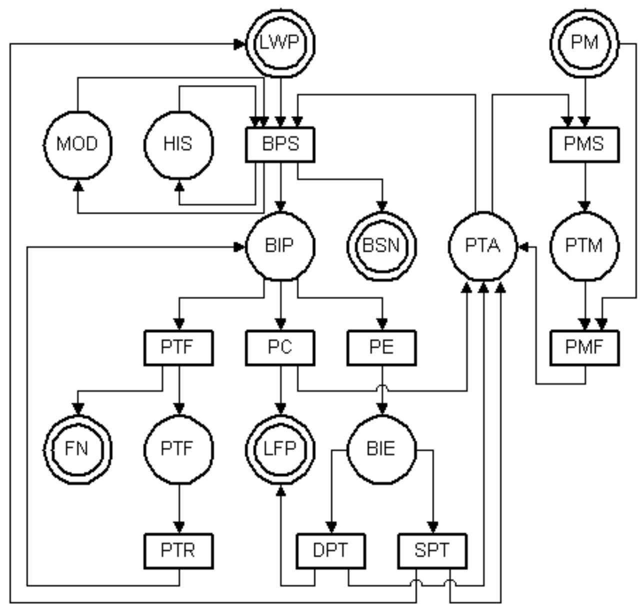


Figure 6: The process tool CPN

corresponding to a single batch are transfered from place $LWP$ to place $BIP$. Furthermore, the token in place $HIS$ provides information on the type of the last loaded batch, which is used in computing the required setup time. Finally, in the proposed process tool CPN, processing of any single batch may result in: (i) normal completion ($PC$), (ii) processing error ($PE$), or (iii) tool failure ($PTF$). This nondeterminism is simulated by the token at place $MOD$, the color of which is set to one of the three values, $m_s$ (success), $m_e$ (error), or $m_f$ (failure), and it is updated according to some probability distribution whenever a batch processing is initiated.

A token of color $m_f$ in place $MOD$ indicates that the next batch to be processed will be handled through the failure mode. In this case, transition $PTF$ fires after a time delay, generated according to a "failure time" distribution, and this firing results in a state with the batch-modeling token at place $PTF$. Furthermore, the failure event is reported to the supervising bay controller through communication place $FN$. Subsequently, transition $PTR$ fires after some simulated "tool repair" time, and the tool resumes its current processing. On the other hand, if the token in place $MOD$ indicates success mode (i.e., $m_s$), processing finishes successfully after a simulated processing time, at which point, transition $PC$ fires, placing a token in communication place $LFP$. This notifies the supervising bay controller that a batch having finished processing, is available at the tool loadport. Finally, when the token in place $MOD$ is of color $m_e$, transition $PE$ fires after a simulated time delay determined according to processing error characteristics of the process tool, and the token in place $BIP$ is directed to place $BIE$, where the process tool necessary for a rework is decided: if the recipe requires the same process tool for the rework, transition $SPT$ fires; otherwise, transition $DPT$ fires.

Place $PM$ holds tokens related to preventive maintenance events. A token of color $e_{ms}$ at place $PM$ indicates a preventive maintenance "start" event. This token enables the firing of transition $PMS$, upon which, a token is transferred from place $PTA$ to place $PTM$, as soon as the process tool becomes idle.[4] This blocks the initiation of any further new processing, until a token of color $e_{mf}$, representing completion of the ongoing preventive maintenance, appears in communication place $PM$. Then, transition $PMF$ fires, returning the "tool availability" token to place PTA, and the tool resumes its normal operation. Finally, we notice that in the proposed tool modeling framework, the required performance control logic is comprises: (i) lot induction logic, that decides whether or not to start a new batch, when the tool batching constraint is satisfied, and (ii) logic selecting the next batch to be processed, when more than one admissible batches are waiting. All these rules regulate the firing of transition $BPS$.

Having defined the generic CPN structure modeling the process tool behavior, we next present the concrete CPN specification for the two process tools considered in the example bay of Section 3.1. These CPN's are obtained by assigning semantics and colors to the generic CPN of Figure 6, in a way that it expresses the operational assumptions presented in Section 3.1. The complete specification of the place and transition colors for the two process tool CPN's are provided in Table 2. Hence, for instance, in Table 2, types $\delta_1$ and $\delta_2$ together specify the set of all admissible batches for process tool $p_1$; that is, transition $BPS$ in the CPN modeling process tool $p_1$, will not be enabled, unless there is a batch-modeling token of color $\delta_1$ or $\delta_2$ at communication place $LWP$ (modeling the loadport content of tool $p_1$). Furthermore, notice that color $(LN, b_1, w_{11e}, LN, b_2, w_{21}, p_1, m_s)$ is a valid color of the $BPS$ instantiation for process tool $p_1$, since it indicates that two lots of stages $w_{11e}$ and $w_{21}$ are waiting for processing at the tool loadports $b_1$ and $b_2$. In contrast, color $(LN, b_1, w_{2s}, LN, b_2, w_{2s}, p_1, m_s)$ is not in the color set of the $BPS$ instantiation for process tool $p_1$, since stages $w_{2s}$ and $w_{2s}$ cannot form an admissible batch for processing tool $p_1$ (c.f., Section 3.1).

The sequence dependent setup times associated with process tool $p_2$, can be computed based on the colors associated with place $BIP$, as follows: (i) Colors $\gamma_1 \star \{p_2\} \star \{w_{12}\} \star M$ and $\gamma_2 \star \{p_2\} \star \{w_{21}\} \star M$ imply that the corresponding batch does not require additional setup, since there is no change in processing stages; (ii) color $\gamma_1 \star \{p_2\} \star \{w_{21}\} \star M$ implies a non-zero setup time, due to the change of lot stage from $w_{12}$ to $w_{21}$; (iii) color $\gamma_2 \star \{p_2\} \star \{w_{12}\} \star M$ implies a non-zero setup time, due to the change of lot stage from $w_{21}$ to $w_{12}$. Finally, we remark that, when transition $PE$ fires, indicating a batch processing error, all lot stages, $w_{ij}$, $i = 1, 2$, and $j =, 1, 2$, are updated to $w_{ije}$, $i = 1, 2$, and $j =, 1, 2$, by the output arc of $PE$. Subsequently, based on the assumptions of Section 3.1, in this particular example, transition $SPT$ fires, and the tokens modeling the processed batches, are transfered back to place $LWP$, indicating rework requirements.

[4] In the proposed process tool CPN, priority is given to transition $PMS$ over $BPS$, when they are in conflict.

| Name | Interpretation | Color ($p_1$) | Color ($p_2$) |
|---|---|---|---|
| LWP | Lot waiting processing | $LN \star \{b_1, b_2\} \star$ $\{w_{10}, w_{21}, w_{2s}, w_{11e}, w_{22e}\}$ | $LN \star \{b_3\} \star$ $\{w_{11}, w_{20}, w_{1s}, w_{12e}, w_{21e}\}$ |
| HIS | History | $\emptyset$ | $\{w_{12}, w_{21}\}$ |
| MOD | Mode | $M$ | $M$ |
| BIP | Batch in processing | ${\beta_1}^a \star \{p_1\} \star M$ $\cup {\beta_2}^c \star \{p_1\} \star M$ | ${\gamma_1}^b \star \{p_2\} \star \{w_{12}, w_{21}\} \star M$ $\cup {\gamma_2}^d \star \{p_2\} \star \{w_{12}, w_{21}\} \star M$ |
| FN | Failure notification | $\{p_1\}$ | $\{p_2\}$ |
| PTF | Process tool failed | $\beta_1 \star \{p_1\} \star \{m_f\}$ $\cup \beta_2 \star \{p_1\} \star \{m_f\}$ | $\gamma_1 \star \{p_2\} \star \{m_f\}$ $\cup \gamma_2 \star \{p_2\} \star \{m_f\}$ |
| LFP | Lot finished processing | $LN \star \{b_1, b_2\} \star \{w_{11}, w_{22}\}$ | $LN \star \{b_3\} \star \{w_{12}, w_{21}\}$ |
| BSN | Batch start notification | $\beta_1 \cup \beta_2$ | $\gamma_1 \cup \gamma_2$ |
| BIE | Batch in error | ${\beta_3}^e \star \{p_1\} \cup {\beta_4}^f \star \{p_1\}$ | ${\gamma_3}^g \star \{p_2\} \cup {\gamma_4}^h \star \{p_2\}$ |
| PTA | Process tool availability | $\{p_1\}$ | $\{p_2\}$ |
| PM | Preventive maintenance | $E_M$ | $E_M$ |
| PTM | Process tool in maintenance | $\{p_1\}$ | $\{p_2\}$ |
| BPS | Batch processing start | ${\delta_1}^i \star \{p_1\} \star M$ $\cup {\delta_2}^j \star \{p_1\} \star M$ | $\gamma_1 \star \{p_2\} \star \{w_{12}, w_{21}\} \star M$ $\cup \gamma_2 \star \{p_2\} \star \{w_{12}, w_{21}\} \star M$ |
| PTF | Process tool failure | $\beta_1 \star \{p_1\} \star \{m_f\}$ $\cup \beta_2 \star \{p_1\} \star \{m_f\}$ | $\gamma_1 \star \{p_2\} \star \{m_f\}$ $\cup \gamma_2 \star \{p_2\} \star \{m_f\}$ |
| PTR | Process tool repair | $\beta_1 \star \{p_1\}$ $\cup \beta_2 \star \{p_1\}$ | $\gamma_1 \star \{p_2\}$ $\cup \gamma_2 \star \{p_2\}$ |
| PC | Processing completion | $\beta_1 \star \{p_1\} \star \{m_s\}$ $\cup \beta_2 \star \{p_1\} \star \{m_s\}$ | $\gamma_1 \star \{p_2\} \star \{m_s\}$ $\cup \gamma_2 \star \{p_2\} \star \{m_s\}$ |
| PE | Processing error | $\beta_1 \star \{p_1\} \star \{m_e\}$ $\cup \beta_2 \star \{p_1\} \star \{m_e\}$ | $\gamma_1 \star \{p_2\} \star \{m_e\}$ $\cup \gamma_2 \star \{p_2\} \star \{m_e\}$ |
| DPT | Different process tool | $\emptyset$ | $\emptyset$ |
| SPT | Same process tool | $\beta_3 \star \{p_1\}$ $\cup \beta_4 \star \{p_1\}$ | $\gamma_3 \star \{p_2\}$ $\cup \gamma_4 \star \{p_2\}$ |
| PMS | Preventive maintenance start | $\{e_{ms}\} \star \{p_1\}$ | $\{e_{ms}\} \star \{p_2\}$ |
| PMF | Preventive maintenance finish | $\{e_{mf}\} \star \{p_1\}$ | $\{e_{mf}\} \star \{p_2\}$ |

[a] $\beta_1 = LN \star \{b_1\} \star \{w_{11}\} \star LN \star \{b_2\} \star \{w_{22}\}$
[b] $\gamma_1 = LN \star \{b_3\} \star \{w_{12}\}$
[c] $\beta_2 = LN \star \{b_1\} \star \{w_{22}\} \star LN \star \{b_2\} \star \{w_{11}\}$
[d] $\gamma_2 = LN \star \{b_3\} \star \{w_{21}\}$
[e] $\beta_3 = LN \star \{b_1\} \star \{w_{11e}\} \star LN \star \{b_2\} \star \{w_{22e}\}$
[f] $\beta_4 = LN \star \{b_1\} \star \{w_{22e}\} \star LN \star \{b_2\} \star \{w_{11e}\}$
[g] $\gamma_3 = LN \star \{b_3\} \star \{w_{12e}\}$
[h] $\gamma_4 = LN \star \{b_3\} \star \{w_{21e}\}$
[i] $\delta_1 = LN \star \{b_1\} \star \{w_{10}, w_{11e}\} \star LN \star \{b_2\} \star \{w_{21}, w_{2s}, w_{22e}\}$
[j] $\delta_2 = LN \star \{b_1\} \star \{w_{21}, w_{2s}, w_{22e}\} \star LN \star \{b_2\} \star \{w_{10}, w_{11e}\}$

Table 2: Definition of place and transition colors for the process tool CPN

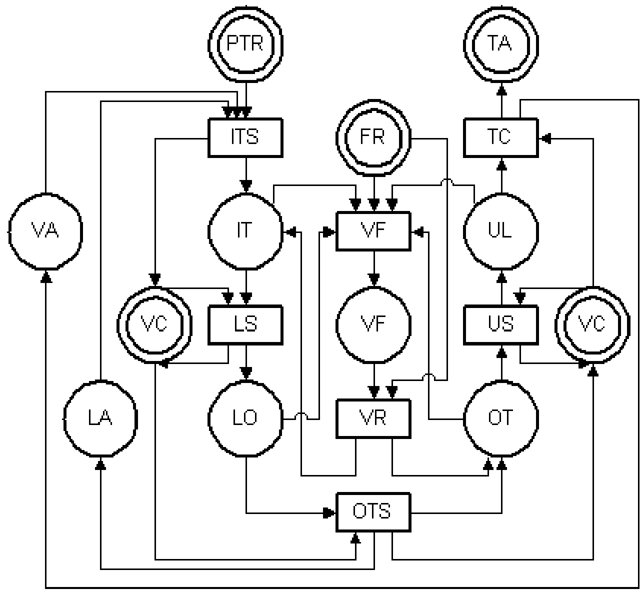


Figure 7: Intrabay MHS CPN

### 3.3 Intrabay MHS CPN

The *intrabay* MHS facilitates wafer lot and auxiliary tool transfers among the process tools and bay stockers. Hence, the shared locations interfacing the MHS to the bay controller are the ports, $b_i$, $i = 1, \ldots, 5$, of the bay process tools and stockers.[5] Furthermore, in the proposed approach, the control of the material flow among these shared locations is the responsibility of the supervising bay controller; for instance, it is the responsibility of the bay controller to ensure that the material transport operation is deadlock-free with respect to the allocation of the port buffering capacity. On the other hand, the MHS controller is responsible for ensuring conflict-free and efficient vehicle routing. In this subsection, we present a generic CPN for a *failure-prone, single-transporter* intrabay MHS, which is appropriate for modeling the behavior of various modes of transportation, including OHT, AGV, and RGV. The proposed model is depicted in Figure 7. Similar to the CPN of Figure 6, in Figure 7, double-circled places denote communication places. Furthermore, we notice that the two places labeled by $VC$ are essentially denoting the same place, which is replicated for presentational clarity.

The MHS controller receives messages from the bay controller indicating lot (or auxiliary tool, respectively) transfer requests, which are represented by an appropriately colored token in communication place $PTR$. In case that the vehicle is idle, the request is handled by executing the following series of operations: (i) First, the vehicle is commanded to make an *inbound trip* to the source port, where the lot (or the auxiliary tool, respectively) is currently residing, by

[5]We notice that in the presented CPN model, the auxiliary tool does not occupy the loadports of process tools while in use. Instead, the auxiliary tool is assumed to be delivered to the tool mounting device directly.

firing transition $ITS$ and putting a token of color $e_{is}$ at place $VC$; (ii) when the vehicle finishes its inbound travel (represented by a token of color $e_{if}$ at place $VC$), it is commanded to *load* the lot (or the auxiliary tool, respectively) to be transfered, by firing transition $LS$ and putting a token of color $e_{ls}$ at place $VC$; (iii) when the vehicle finishes loading (represented by a token of color $e_{lf}$ at place $VC$), it is commanded to make an *outbound trip* to the destination port, by firing transition $OTS$ and putting a token of color $e_{os}$ at place $VC$; (iv) finally, when the vehicle finishes its outbound travel (represented by a token of color $e_{of}$ at place $VC$), it is commanded to *unload* the transfered lot (or the auxiliary tool, respectively), by firing transition $US$ and putting a token of color $e_{us}$ at place $VC$.

After the successful completion of all the above four operations, the intrabay MHS controller places a token at place $TA$, that notifies the bay supervising controller about the transfer completion, and releases the vehicle. However, in case that a token of color $e_f$ is deposited from an external event generator to place $FR$, the vehicle is enters a "failure" mode, and no further material handling operation is possible, until a token of color $e_r$, representing a repair event, is available at place $FR$. Finally, whenever the vehicle successfully completes a transfer request, the MHS controller selects from place $PTR$ a pending transfer request, for which the destination port is empty (if there are any). Otherwise, the vehicle becomes idle. The associated selection rule is known as *vehicle initiated dispatching* [14], and it should be embedded in the specification of transition $ITS$.

The complete specification of the place and transition colors for intrabay MHS CPN of the considered example, is provided in Table 3. Notice that place $PTR$ has two kinds of color sets associated with it: Color set $LN \star B \star B \star e_{tr}$ indicates a lot transfer request, whereas color set $LN \star \{t_1\} \star Q \star Q \star e_{tr}$ indicates an auxiliary tool transfer request. Furthermore, the constituent data types for defining the place colors are decreasing along with the completion of the four sub-operations characterized above. For instance, after the vehicle starts its outbound travel (through the firing of transition $OTS$), two component data types, $B$ and $Q$, are removed from the trip-modeling token, indicating that the information regarding the source location is not needed any more.

Finally, we remark that, in the general case, the complexity of the MHS control logic depends on the number of transporters in the system, as well as the topology of the interconnecting guidepath network. A generic CPN specification model of the MHS controller dealing with more general, zone controlled AGV systems, is provided in [30].

## 3.4 Bay CPN

A bay consists of a number of process tools, stockers, and a MHS facilitating the intrabay transfer of lots and auxiliary tools. Lots are routed to the different process tools and stockers

| Name | Interpretation | Color |
|---|---|---|
| PTR | Pending transfer requests | $LN \star B \star B \star e_{tr} \cup LN \star \{t_1\} \star Q^a \star Q \star e_{tr}$ |
| VA | Vehicle availability | $\{v_1\}$ |
| LA | Location availability | $B \cup Q$ |
| IT | Inbound travel | $LN \star B \star B \star \{v_1\} \cup LN \star \{t_1\} \star Q \star Q \star \{v_1\}$ |
| LO | Loading | $LN \star B \star B \star \{v_1\} \cup LN \star \{t_1\} \star Q \star Q \star \{v_1\}$ |
| OT | Outbound travel | $LN \star B \star \{v_1\} \cup LN \star \{t_1\} \star Q \star \{v_1\}$ |
| UL | Unloading | $LN \star B \star \{v_1\} \cup LN \star \{t_1\} \star Q \star \{v_1\}$ |
| TA | Transfer acknowledgment | $LN \star B \cup LN \star \{t_1\} \star Q$ |
| VC | Vehicle communication | $E_V$ |
| FR | Failure / repair | $E_F$ |
| VF | Vehicle failed | $LN \star B \star B \star \{v_1\} \cup LN \star \{t_1\} \star Q \star Q \star \{v_1\}$ $\cup\ LN \star B \star \{v_1\} \cup LN \star \{t_1\} \star Q \star \{v_1\}$ |
| ITS | Inbound travel start | $LN \star B \star B \star v_1 \star e_{is} \cup LN \star \{t_1\} \star Q \star Q \star v_1 \star e_{is}$ |
| LS | Loading start | $LN \star B \star B \star v_1 \star e_{if} \star e_{ls} \cup LN \star \{t_1\} \star Q \star Q \star v_1 \star e_{if} \star e_{ls}$ |
| OTS | Outbound travel start | $LN \star B \star B \star v_1 \star e_{lf} \star e_{os} \cup LN \star \{t_1\} \star Q \star Q \star v_1 \star e_{if} \star e_{os}$ |
| US | Unloading start | $LN \star B \star v_1 \star e_{of} \star e_{us} \cup LN \star \{t_1\} \star Q \star v_1 \star e_{of} \star e_{us}$ |
| TC | Transfer completion | $LN \star B \star v_1 \star e_{uf} \cup LN \star \{t_1\} \star Q \star v_1 \star e_{uf}$ |
| VF | Vehicle failure | $LN \star B \star B \star \{v_1\} \star \{e_f\} \cup LN \star \{t_1\} \star Q \star Q \star \{v_1\} \star \{e_f\}$ $\cup\ LN \star B \star \{v_1\} \star \{e_f\} \cup LN \star \{t_1\} \star Q \star \{v_1\} \star \{e_f\}$ |
| VR | Vehicle repair | $LN \star B \star B \star \{v_1\} \star \{e_r\} \cup LN \star \{t_1\} \star Q \star Q \star \{v_1\} \star \{e_r\}$ $\cup\ LN \star B \star \{v_1\} \star \{e_r\} \cup LN \star \{t_1\} \star Q \star \{v_1\} \star \{e_r\}$ |

[a] $Q$ represents the set of locations where the auxiliary tool can be placed. It follows that $b_5 \in Q$.

Table 3: Definition of place and transition colors for the intrabay MHS CPN

according to their pre-defined recipes, and auxiliary tools are delivered to the process tools when they are required for the execution of the various processing steps. Hence, in a 300mm bay, a single step may require multiple resources from different resource types (e.g. acquisition of a tool loadport and reticle for a lithography operation). At the same time, there might exist more than one process tools capable for executing a step, or a lot might need to be dynamically re-directed to a stocker. Furthermore, due to technological and/or economical considerations, additional constraints may be imposed on the bay processes, such as the requirement for batch-based processing at a bay furnace.

The bay operational characteristics outlined above can be formally modeled by the *bay* CPN depicted in Figure 8. This CPN allows the modeling of all the aforestated operational features of 300mm bay, more formally characterized as *routing flexibility, multiple resource acquisitions*, and *logical/behavioral constraints*. In the CPN of Figure 8, place $RES$ maintains two types of resources: *physical* and *logical*. While physical resources represent the availability of (buffering capacity at the) bay production equipment and/or auxiliary resources, logical resources model additional behavioral constraints. The role of logical resources will be demonstrated later in

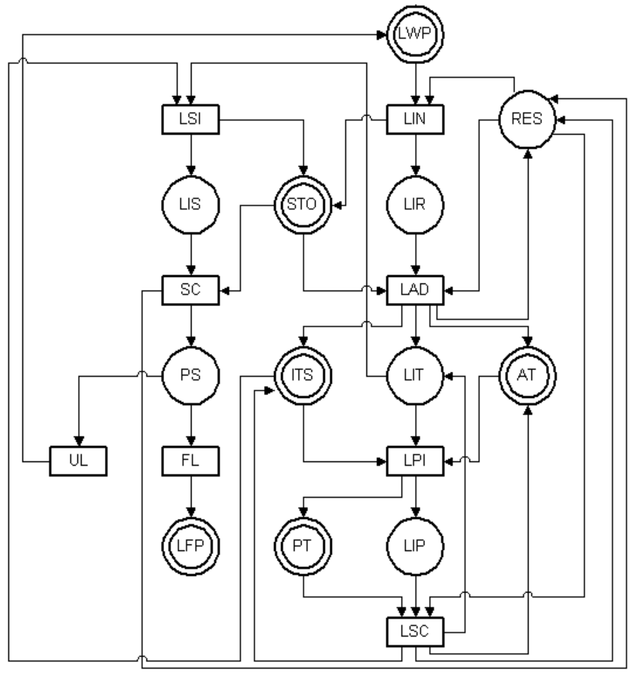


Figure 8: Bay CPN

this section, by means of the considered example.

The arrival of a lot at the bay stocker is indicated by the presence of a token in place $LWP$. New lot arrivals, waiting to the stocker, are induced into the bay through the firing of transition $LIN$, and they are advanced to the bay process tools according to their predefined recipe, through the firing of transition $LAD$, that regulates the allocation of the bay resources. When transition $LAD$ fires, a token representing a lot transfer request, is placed on communication place $ITS$, and, if an auxiliary tool delivery is also necessary, another token is placed on place $AT$. Actual lot processing at a certain tool is authorized only after the corresponding lot (and the potentially required auxiliary tool) transfer are acknowledged from the intrabay MHS controller, by placing an appropriately colored token at communication place $PT$. Hence, place $PT$ in Figure 8 is the counterpart of place $LWP$ in Figure 6, in the underlying communication channel from the bay to the process tool controller.

Similarly, a stage completion event is indicated by a token in communication place $PT$, coming from the underlying process tool controller. The availability of this token enables the lot advancement to the next required process tool, or its re-direction to a stocker for temporary storage. The corresponding decision rule is incorporated into the firing logic of transition $LSC$, and the necessary transfer requests are made accordingly: If a lot is re-directed to a stocker, transition $LSI$ fires, consuming the token in place $LIT$ that requests storage to the stocker. After the storage operation is finished (firing $SC$), in case that the lot is re-routed to a stocker before it finishes all the steps in the bay, the firing of transition $UL$ re-introduces the lot-modeling token to place $LWP$, signaling its waiting for induction into the bay, or, in the special case that the lot has completed its processing in the considered bay, the corresponding token is deposited to communication place $LFS$ (through the firing of transition $FL$), which signals the completion event to the higher-level (fab) supervising controller.

Controlling the material flow in a bay running according to the above operational scheme, gives rise to both logical and performance decisions. In particular, the firing of transitions $LAD$ and $LAC$, dealing with the bay resource allocation, must be regulated by a *structural control policy (SCP)*, in order to ensure the logically correct and deadlock-free behavior. In fact, the resource allocation dynamics defined for the bay model in this paper, can be effectively modeled as a C/D-RAS (*Conjunctive / Disjunctive Resource Allocation System*), and it is possible to apply the structural control theory developed in [29]. On the other hand, the proposed bay CPN gives rise to three performance-related decision problems, that can be identified as follows: (i) *lot induction*, (ii) *lot dispatching*, and (iii) *dynamic lot routing*.[6] To the best of our knowledge, currently, there are no rigorous computationally efficient solutions to the underlying mathematical formulations of the aforementioned scheduling problem, when all the operational details addressed in this paper are taken into consideration. Therefore, under the existing circumstances, the most viable solution for the bay performance control logic seems to be the employment of some of the existing *dispatching rules* (e.g. [3]). The specific set of rules to be employed will be context/configuration specific, and it should be identified through (simulation-based) experimentation.

The specification of the place and transition colors for the bay CPN modeling the wafer lot flow in the considered example, is provided in Table 4. Notice that the multiple resource requirements for stages $w_{11}$, $w_{21}$, $w_{2s}$, and $w_{22}$, are inscribed by the color definitions associated with transitions $LAD$ and $LSC$. For instance, transition color $LN \star \{w_{10}\} \star \{b_4\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \{e_{rc}\} \star \epsilon_1$ of $LAD$ represents that a new lot currently residing at $b_4$ (represented by a token at place $LIR$), requires loadport $b_1$ or $b_2$, and auxiliary tool $t_1$ for executing stage $w_{11}$. Furthermore, since the color of the token returned to place $PT$ – by a process tool CPN upon a processing completion – indicates the next requested stage (e.g. from $w_{10}$ to $w_{11}$, or from $w_{2s}$ to $w_{22}$), it follows that all the possible lot routings in the considered bay, can be characterized by the following five colors associated with transition $LSC$: (i) $LN \star \{w_{11}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \{b_3\} \star \{e_{tr}\}$ ($w_{11}$ proceeds to $w_{12}$), (ii) $LN \star \{w_{11}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \{w_{1s}\} \star \{b_4\} \star \{e_{tr}\}$ ($w_{11}$ is re-directed to the stocker), (iii) $LN \star \{w_{21}\} \star \{b_3\} \star \{t_1\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_2\} \star \epsilon_1$ ($w_{21}$ proceeds to $w_{22}$), (iv) $LN \star \{w_{21}\} \star \{b_3\} \star \{t_1\} \star \{w_{2s}\} \star \{b_4\} \star \{e_{tr}\}$ ($w_{21}$ is re-directed to the stocker), and finally (v) $LN \star \{w_{12}, w_{22}\} \star \{b_3\} \star \{b_4\} \star \{e_{tr}\}$ ($w_{12}$ and $w_{22}$ are directed to the

[6] This last issue includes the selection of a process tool, for stages presenting routing flexibility, and the potential dynamic re-routing of a lot to a stocker.

| Name | Interpretation | Color |
|---|---|---|
| RES | Resource | $B \cup \{t_1\} \cup C$ |
| LWP | Lot waiting processing | $LN \star \{w_{10}, w_{1s}, w_{20}, w_{2s}\}$ |
| LIR | Lot in retrieval | $LN \star \{w_{10}, w_{1s}, w_{20}, w_{2s}\} \star \{b_4\}$ |
| LIT | Lot in transfer | $LN \star \{w_{10}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\}$<br>$\cup LN \star \{w_{21}, w_{2s}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_2\}$<br>$\cup LN \star \{w_{20}\} \star \{b_3\} \star \{t_1\}$<br>$\cup LN \star \{w_{11}, w_{1s}\} \star \{b_3\}$<br>$\cup LN \star \{w_{1s}\} \star \{b_4\}$<br>$\cup LN \star \{w_{2s}\} \star \{b_4\}$ |
| LIP | Lot in processing | $LN \star \{w_{10}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\}$<br>$\cup LN \star \{w_{21}, w_{2s}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_2\}$<br>$\cup LN \star \{w_{20}\} \star \{b_3\} \star \{t_1\}$<br>$\cup LN \star \{w_{11}, w_{1s}\} \star \{b_3\}$ |
| AT | Auxiliary tool | $LN \star \{t_1\} \star P \star E_A$ |
| STO | Stocker | $LN \star E_S$ |
| ITS | Intrabay transport system | $LN \star B \star B \star \{e_{tr}\} \cup LN \star B \star \{e_{tc}\}$ |
| PT | Process tool | $LN \star \{w_{10}, w_{21}, w_{2s}\} \star \{b_1, b_2\}$<br>$LN \star \{w_{20}, w_{11}, w_{1s}\} \star \{b_3\}$ |
| LIS | Lot in storage | $LN \star \{w_{1s}, w_{12}, w_{2s}, w_{22}\} \star \{b_4\}$ |
| PS | Processing status | $LN \star \{w_{1s}, w_{12}, w_{2s}, w_{22}\}$ |
| LFP | Lot finished processing | $LN \star \{w_{12}, w_{22}\}$ |
| LIN | Lot induction | $LN \star \{w_{10}, w_{1s}, w_{20}, w_{2s}\} \star \{b_4\} \star \{e_{rr}\}$ |
| LAD | Lot advancement | $LN \star \{w_{10}\} \star \{b_4\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \{e_{rc}\} \star \epsilon_1$ [a]<br>$\cup LN \star \{w_{2s}\} \star \{b_4\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_2\} \star \{e_{rc}\} \star \epsilon_1$<br>$\cup LN \star \{w_{20}\} \star \{b_4\} \star \{b_3\} \star \{t_1\} \star \{e_{rc}\} \star \epsilon_1$<br>$\cup LN \star \{w_{1s}\} \star \{b_4\} \star \{b_3\} \star \{e_{rc}\} \star \{e_{tr}\}$ |
| LPI | Lot processing initiation | $LN \star \{w_{10}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \epsilon_2$ [b]<br>$\cup LN \star \{w_{21}, w_{2s}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_2\} \star \epsilon_2$<br>$\cup LN \star \{w_{20}\} \star \{b_3\} \star \{t_1\} \star \epsilon_2$<br>$\cup LN \star \{w_{11}, w_{1s}\} \star \{b_3\} \star \{e_{tc}\}$ |
| LSC | Lot stage completion | $LN \star \{w_{11}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \{b_3\} \star \{e_{tr}\}$<br>$\cup LN \star \{w_{11}\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_1\} \star \{w_{1s}\} \star \{b_4\} \star \{e_{tr}\}$<br>$\cup LN \star \{w_{21}\} \star \{b_3\} \star \{t_1\} \star \{b_1, b_2\} \star \{t_1\} \star \{c_2\} \star \epsilon_1$<br>$\cup LN \star \{w_{21}\} \star \{b_3\} \star \{t_1\} \star \{w_{2s}\} \star \{b_4\} \star \{e_{tr}\}$<br>$\cup LN \star \{w_{12}, w_{22}\} \star \{b_3\} \star \{b_4\} \star \{e_{tr}\}$ |
| LSI | Lot storage initiation | $LN \star \{w_{1s}, w_{12}, w_{2s}, w_{22}\} \star \{b_4\} \star \{e_{tc}\} \star \{e_{sr}\}$ |
| SC | Storage completion | $LN \star \{w_{1s}, w_{12}, w_{2s}, w_{22}\} \star \{b_4\} \star \{e_{sc}\}$ |
| UL | Unfinished lot | $LN \star \{w_{1s}, w_{2s}\}$ |
| FL | Finished lot | $LN \star \{w_{12}, w_{22}\}$ |

[a] $\epsilon_1 = \{e_{tr}\} \star \{e_{ar}\}$

[b] $\epsilon_2 = \{e_{tc}\} \star \{e_{ac}\}$

Table 4: Definition of place and transition colors for the bay CPN

stocker, since all the steps in the bay have been completed).

The batching requirement for process tool $p_1$ is enforced by two logical resource tokens, $c_1$, and $c_2$, introduced to impose this behavioral restriction. Specifically, the key mechanism for enforcing the batching condition in the considered example, is based on the observation that this condition can be satisfied if and only if, among the lots waiting for processing at the loadports of process tool $p_1$, the number of lots requesting the execution of stage $w_{11}$ are strictly less than 2, and the number of lots requesting the execution of stage $w_{22}$ is also strictly less than 2. This last constraint is imposed on the CPN behavior, by (i) introducing to place $RES$ two new tokens, with respective colors $c_1$ and $c_2$, and (ii) requiring that a lot in stage $w_{10}$, needs resource $c_1$, as well as the corresponding physical resources, for advancing to stage $w_{11}$, and similarly, a lot finishing stages $w_{21}$ or $w_{2s}$, needs resource $c_2$, as well as the corresponding physical resources, for advancing to stage $w_{22}$. These additional logical resource allocation requirements are captured through the color definitions associated with transitions $LAD$ and $LSC$.

Finally, we remark that an additional CPN component, modeling and controlling the auxiliary tool flow, is required to complete the whole specification of the example 300mm bay. The role of this additional CPN component is to keep track of the current location of the auxiliary tool, so that the appropriate tool transfer requests are made to the intrabay MHS CPN. Its formal specification can be obtained through a straightforward modification of the lot tracking mechanism in the presented bay CPN model, by replacing the lot data types with the auxiliary tool data type, and it is, therefore, omitted for the sake of brevity.

# 4 A Web-Based Simulation Platform for the Proposed 300mm Fab Modeling and Control Framework

The application of discrete event simulation for (i) the *empirical validation of the correctness* of synthesized real-time control policies, and (ii) the *evaluation of the resulting system performance*, is already an established practice in the semiconductor manufacturing industry. Currently, however, the development of the fab operational model and the synthesis of the control policies is more of an art rather than a structured engineering process, and the deployment and support of the simulation capability almost exclusively requires the employment of externally hired simulation experts. Furthermore, given the capabilities of the existing commercial simulation platforms, in many cases, the underlying fab control logic is confounded with the representation of the physical and logistical operations, and therefore, it is only partially or implicitly stated in the resulting software model. As a result, the development of high-fidelity simulation models for fab operations is expensive, time consuming and quite complicated. Moreover, the end product is rather esoteric, and therefore, not easily manageable by the shop-floor

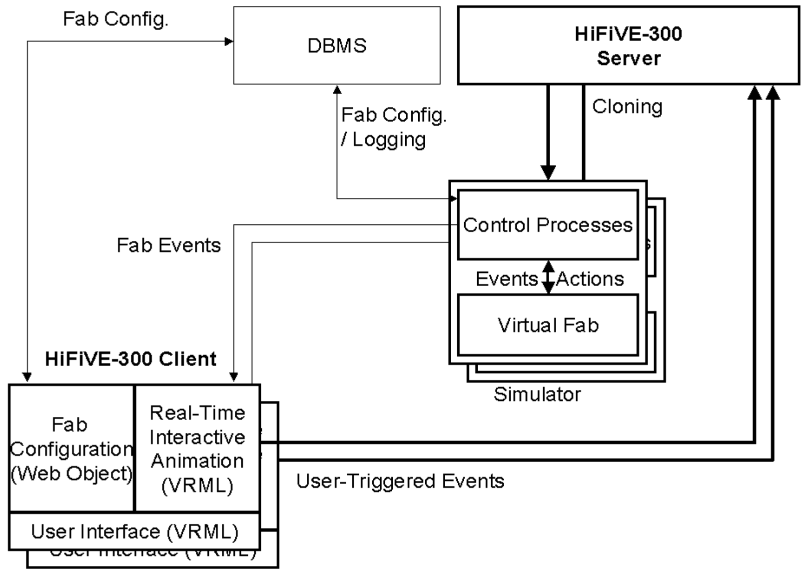


Figure 9: The HiFiVE-300 architecture

control engineers [26].

Motivated by the above remarks, our proposition is the development of a *Web-based simulator* that will provide a *structured* and *integrated* environment for the *modeling, analysis and control of the 300mm fab operations*, based on the fab modeling, behavioral analysis, and control synthesis framework presented in Section 2. More specifically, the simulator user will interact with the software through a *client web-based application*, and she will be able to *configure* the simulated fab model in terms of (i) a number of *operational parameters / features*, including the fab, bay and the inter/intra-bay MHS layouts, and (ii) the applied *structural* and *performance control policies*. Hence, following the modeling principles outlined in Section 2, the proposed simulation environment will explicitly distinguish between the plant behavioral model, and the applied control policies. Furthermore, following the generic scheme of Figure 4, the control logic itself will be differentiated into *logical / structural* and *performance*-related policies, with the former addressing *admissibility decisions* relating to the establishment of deadlock-free material flow, conflict-free vehicle routing, consistent tool / reticle management and acceptable batching patterns, and the latter addressing *scheduling decisions*, like job induction and dispatching to the different system processing units, efficient vehicle allocation to job transfer requests and routing, and efficient tool / reticle allocation to the requesting processes. Finally, the simulator will provide for the specification of *maintenance policies*, and the *handling of exceptional situations*, like equipment failures and the accommodation of expedient jobs.

Once the user has configured the fab model and the underlying operational logic, the data will be stored in a *database system*, remotely located on the simulation *server*. Upon the user

/ client's command, the server will initiate the simulation of the real-time fab operations, while the simulation *output events* will be communicated to the client through CORBA *middleware* [27], and it will drive a *client-based* VRML *animation* (cf. http://www.vrml.org). The simulation itself will be *distributed*, consisting of a number of communicating concurrent processes, corresponding to the fab operational units / control domains and their associated controllers. In this way, it would be conceptually possible to (eventually) substitute plant modeling units with their corresponding actual processes, converting the simulator to a *real-time controller*. Under more conventional usage, the user will be able to obtain a number of *performance indices*, including flow time, resource utilization and yields. Furthermore, the user will have the capability to interact with the simulation in real-time, by triggering a number of events like machine failures / repairs, and the arrival of expedient jobs, and thus, she will be able to test the controlled system behavior under a number of *dynamically generated scenaria*. A schematic representation of the proposed software architecture - to be called the *High Fidelity Virtual Environment for 300mm Fab (HiFiVE-300)* architecture - is given in Figure 9. The formal and structured modeling approach underlying the software development, combined with the application of prevailing *object-oriented* technologies [15] in the software implementation, will also facilitate the *maintenance of the proposed simulation platform* in terms of modification and inclusion of new basic component behaviors and control policies.

# 5 Discussion

The research program outlined in this paper is a major effort undertaken by the *Virtual Factory Lab (VFL)*, in the *School of Industrial & System Engineering*, at *Georgia Institute of Technology*. Currently we are in the stage of detailing and validating the formal models - Colored Petri Nets [16] - representing the 300mm fab operation at the process tool, bay and fab level, in the proposed hierarchical framework. Kempf's mini-fab specification [17] was our initial reference point for the real-time fab operation. However, part of our current research effort seeks to validate the initial modeling assumptions through industry interaction and collaboration (e.g.,[1]). Subsequent steps will involve the *software implementation* of the parameterized models, their *integration* in the architecture of Figure 9, and the development of the *user interfaces* that will facilitate the effective and efficient utilization of the developed simulation capability by the fab control engineers and the interested educational audiences.

# A Appendix: Colored Petri Net-based Modeling

In this appendix, we present fundamental concepts behind the CPN modeling framework that facilitate the interpretation of the CPN-based 300mm bay model of Section 3. For a more extensive discussion on the CPN modeling framework, its applicability, and the available analysis techniques, the reader is referred to [12, 4, 16].

According to [4], a *Colored Petri net* is formally defined by a 6-tuple $CPN = (P, T, C, I^-, I^+, M_0)$, where:

1. $P$ is a finite and non-empty set of *places*,

2. $T$ is a finite and non-empty set of *transitions*,

3. $P \cap T = \emptyset$,

4. $C$ is a *color* function defined from $P \cup T$ into finite and non-empty sets,

5. $I^-$ and $I^+$ are the *backward* and *forward incidence* functions defined on $P \times T$ such that $I^-(p,t), I^+(p,t) \in C(t) \rightarrow C(p)_{MS}$, $\forall (p,t) \in P \times T$,

6. $M_0$ is a function defined on $P$, describing the net *initial marking*, such that $M_0(p) \in C(p)_{MS}$, $\forall p \in P$.

Furthermore, in the above definition, $C(p)_{MS}$ denotes the *set of all multisets* defined over the set $C(p)$, the color set associated with place $p$. The concept of a *multiset* $m$ over a non-empty set $S$ is itself formally defined as a function $m \in [S \rightarrow Z^+]$, where $Z^+$ represents the set of natural numbers. Usually, $m$ is represented by the formal sum $\sum_{s \in S} m(s)`s$.

The concepts of *places, transitions* and *arcs* in the above definition have the same significance, and they are represented by the same semantics, as in the ordinary Petri net framework. However, tokens occupying any CPN place, $p$, possess additional attributes, which are formally expressed by the *color set*, $C(p)$, associated with place $p$. Formally, a color set is a *type set*, and associates with the tokens of place $p$ a type, in the same way that types are associated to variables in any structured programming language. In particular, types can be defined over any set of primitives, and, similar to the programming language case, composite types can be defined from simpler types; of particular interest to this work is the *product* operation, $S_1 \star S_2$, over two (or more) type sets $S_1$ and $S_2$, which essentially defines a *record* type, with two *fields* of type $S_1$ and $S_2$.

In CPN, the logic behind the *firing* of a net transition $t \in T$ can be formally stated as follows: A transition $t \in T$ is *enabled* in a marking $M$ with respect to a color $c' \in C(t)$, denoted by $M[(t,c')\rangle$, iff $M(p)(c) \geq I^-(p,t)(c')(c), \forall p \in P, c \in C(p)$. An enabled transition $t \in T$ may *fire* in a marking $M$ with respect to a color $c' \in C(t)$ yielding a new marking $M'$, denoted by $M[(t,c')\rangle M'$ with $M'(p)(c) = M(p)(c) + I^+(p,t)(c')(c) - I^-(p,t)(c')(c), \forall p \in P, c \in C(p)$. A sequence of transition firings such that $M_1[(t_1,c_1)\rangle M_2[(t_2,c_2)\rangle M_3 \ldots M_{n-1}[(t_{n-1},c_{n-1})\rangle M_n$ is characterized as a *finite occurrence sequence* between markings $M_1$ and $M_n$, and it is more concisely denoted by $M_1[(t_1,c_1)(t_2,c_2)\ldots(t_{n-1},c_{n-1})\rangle M_n$. Furthermore, we say that marking

$M'$ is *reachable* from marking $M$ iff there exists a finite occurrence sequence leading from $M$ to $M'$.

Finally, communication between the various system controllers is modeled in our CPN representation through the employment of *communication places* in the proposed CPN structures (similar to [10]). Specifically, communication places appear in pairs of *sender* and *receiver*, each contained in a separate net instantiation, and they model the operation of a *unidirectional communication channel* among the corresponding controllers. Hence, every time that a token is placed in the sender communication place, it is automatically consumed by being transferred to the corresponding receiver place. The color content of the token should be interpreted as the message content / parameters passed from the sender to the receiver controller.

**ACKNOWLEDGMENT** The authors would like to acknowledge the support of (i) W. M. Keck Foundation and (ii) NSF - Grant no. DUE-9950301, for the presented research effort.

## References

[1] In *Minutes of 1st Meeting of the PRI-A Colloquium on Future Semiconductor Manufacturing*. PRI-A, 2000.

[2] M. E. Bader, R. P. Hall, and G. Strasser. Integrated processing equipment. *Solid State Technology*, 33(5):149–154, 1990.

[3] A. D. Baker. A survey of factory control algorithms that can be implemented in a multi-agent heterarchy : Dispatching, scheduling, and pull. *Journal of Manufacturing Systems*, 17(4):297–320, 1998.

[4] F. Bause and P.S. Kritzinger. *Stochastic Petri Nets: An Introduction to the Theory*. Vieweg, 1996.

[5] D. A. Bodner. *Real-Time Control Approaches to Deadlock Management in Automated Manufacturing Systems*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1996.

[6] D. A. Bodner, J. Park, S. Reveliotis, and L. F. McGinnis. Integration of structural and performance-oriented control in flexibly automated manufacturing. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 1999.

[7] D. A. Bodner and S. A. Reveliotis. Flexible semiconductor manufacturing. In J. G. Web-ster, editor, *Encyclopedia of Electrical and Electronics Engineering*, volume 7, pages 596–607. John Wiley & Sons, 1999.

[8] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[9] D. Connors, G. Feigin, and D. Yao. Scheduling semiconductor lines using a fluid network model. *IEEE Transactions on Robotics & Automation*, 10:88–98, 1994.

[10] R. Cossins and P. Ferreira. Celeritas: A coloured petri net approach to simulation and control of flexible manufacturing systems. *International Journal of Production Research*, 30:1925–1956, 1992.

[11] C. Cunningham, R. Wright, K. Benhayoune, E. Campbell, V. Swaminathan, and R. White. 300mm factory layout and automated material handling system analysis. In *Proceedings of the Autosimulations Symposium.* Autosimulations, 1999.

[12] R. David and H. Alla. *Petri Nets and Grafcet.* Prentice Hall, 1992.

[13] H. Du and C. Zhou. Message-oriented decomposition for supervisory control in manufacturing system. *Robotics & Computer Integrated Manufacturing*, 15(6):441–452, 1999.

[14] T. Ganesharajah, N. G. Hall, and C. Sriskandarajah. Design and operational issues in agv-served manufacturing systems. *Annals of Operations Research*, 76:109–154, 1998.

[15] I. Jacobson. *Object-Oriented Software Engineering.* Addison-Wesley, 1992.

[16] K. Jensen. *Coloured Petri Nets*, volume 1. Springer-Verlag, NY, 2nd edition, 1997.

[17] K. Kempf. Intel five-machine six-step mini-fab description. Technical Report http://www.eas.asu.edu/ aar/research/intel/papers/fabspec.html, Intel, 1994.

[18] D. C. Kozen. *Automata and Computability.* Springer Verlag, 1997.

[19] P. R. Kumar. Scheduling manufacturing systems of re-entrant lines. In D. D. Yao, editor, *Stochastic Modeling and Analysis of Manufacturing Systems*, pages 325–360. Springer-Verlag, 1994.

[20] P. R. Kumar. Scheduling semiconductor manufacturing plants. *IEEE Control Systems Magazine*, 14(6):33–40, 1994.

[21] M. Lawley, S. Reveliotis, and P. Ferreira. The application and evaluation of banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 10(1):73–100, 1998.

[22] M. Lawley, S. Reveliotis, and P. Ferreira. A correct and scalable deadlock avoidance policy for flexible manufacturing systems. *IEEE Transactions on Robotics & Automation*, 14(5):796–809, 1998.

[23] T. McCutchen and C-Y. Lee. An analysis of dispatching rules in a semiconductor wafer fabrication environment. *Journal of Electronics Manufacturing*, 5:165–174, 1995.

[24] T. E. Morton and D. W. Pentico. *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*. Wiley, 1993.

[25] S. Narayanan, D. A. Bodner, U. Sreekanth, T. Govindaraj, L. F. McGinnis, and C. M. Mitchell. Modeling control decisions in manufacturing systems simulation using objects. In *Proceedings of the 1994 IEEE International Conference on Systems, Man, & Cybernetics*, volume 2, pages 1392–1397. IEEE, 1994.

[26] NEMI. National electronics manufacturing technology roadmaps. Technical report, National Electronics Manufacturing Initiative (NEMI), Inc., 1998.

[27] R. Orfali and D. Harkey. *Client / Server Programming with Java and CORBA*. John Wiley & Sons, 2nd edition, 1998.

[28] J. Park, S. Reveliotis, D. Bodner, C. Zhou, J-F. Wu, and L. F. McGinnis. High-fidelity rapid prototyping of the real-time fab operations through discrete event system modeling. In *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing*, pages 163–171, 2000.

[29] J. Park and S. A. Reveliotis. Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings. Technical Report (submitted to IEEE Transactions on Automatic Control), School of Industrial & Systems Engineering, Georgia Institute of Technology, 2000.

[30] J. Park, S. A. Reveliotis, D. A. Bodner, and L. F. McGinnis. A distributed, event-driven control architecture for flexibly automated manufacturing systems. Technical Report (submitted to International Journal of Computer Integrated Manufacturing), School of Industrial & Systems Engineering, Georgia Institute of Technology, 1999.

[31] N. Patel. Using simulation to evaluate complex control logic for lot delivery. In *Proceedings of Autosimulations Symposium*. Autosimulations, 1999.

[32] S. A. Reveliotis. Conflict resolution in agv systems. *IIE Transactions*, 32(7):647–659, 2000.

[33] S. A. Reveliotis. The destabilizing effect of blocking due to finite buffering capacity in multi-class queueing networks. *IEEE Transactions on Automatic Control*, 45(3):585–588, 2000.

[34] S. A. Reveliotis and P. M. Ferreira. Deadlock avoidance policies for automated manufacturing cells. *IEEE Transactions on Robotics & Automation*, 12(6):845–857, 1996.

[35] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira. Polynomial complexity deadlock avoidance policies for sequential resource allocation systems. *IEEE Transactions on Automatic Control*, 42(10):1344–1357, 1997.

[36] SEMI. Generic model for communications and control of semi equipment (gem). Technical Report SEMI E30-95, Semiconductor Equipment and Materials International (SEMI), 1999.

[37] A. Sharifnia. Stability and performance of a simple distributed tracking policy for production control of manufacturing systems. *IEEE Transactions on Automatic Control*, 40:1109–1113, 1995.

[38] P. Singer. The driving forces in cluster tool development. *Semiconductor International*, 18(1):113–118, 1995.

[39] R. Uzsoy, C-Y. Lee, and L. A. Martin-Vega. A review of production planning and scheduling models in the semiconductor industry part ii : Shop-floor control. *IIE Transactions*, 26:44–55, 1994.

[40] J. Zoghby. Improving planning decisions through autosched ap. In *Proceedings of Autosimulations Symposium*. Autosimulations, 1999.