

A High-Fidelity, Web-Based Simulator for 300mm Wafer Fabs

Hansoo Kim, Jonghun Park, Sugje Sohn, Ying Wang, Spyros Reveliotis,
Chen Zhou, Douglas Bodner, and Leon McGinnis

School of Industrial and Systems Engineering
Georgia Institute of Technology
765 Ferst Drive, Atlanta, GA 30332-0205 USA

Abstract

Semiconductor fabs are capital intensive. The rate of capital return heavily depends on their productivity. Accordingly, simulation has been adopted in many cases as a viable design and analysis tool to achieve better productivity. However, the gap between the simplistic simulation models and real complex systems has limited the confidence to apply simulation results directly to real systems. Ideally, the simulator should support the rapid modeling and fast execution of high-fidelity fab models in order to be able to provide more realistic simulation results. In this paper, we present our current research effort to develop a high-fidelity web-based simulator for the 300mm wafer fabs. We take a model-view-control approach that allows us to develop modular and reusable fab simulation objects. The proposed approach is implemented as a distributed system that uses a message-based event synchronization mechanism to coordinate the simulation objects as well as to support distributed simulation execution. The presented simulator also provides a high quality VRML animation for visualization of real time paced simulation execution.

Keywords

High-fidelity simulation, Distributed simulation, 300mm wafer fabs, Web-based simulation, Rapid prototyping, VRML animation.

1 Introduction

Today's semiconductor industry is facing a big challenge in wafer fabrications that is characterized by a transition of wafer size from 200mm to 300mm [2]. This change not only requires that manufacturing equipment such as process tools, material transporters, and stockers be customized to the new wafer size, but also that the whole material handling be fully automated mainly due to the higher sensitivity to wafer contamination and increased wafer weights. 300mm wafer fabs have specific characteristics that differentiate them from traditional manufacturing systems [13]: The basic material-handling unit is

called FOUP (Front Opening Unified Pod), carrying 25 wafers [3]. A typical FOUP requires to visit hundreds of process tools for its successful completion, and the flow of FOUPs is re-entrant [9]. Furthermore, the existence of various engineering lots together with production lots make the material flow of wafer fabs further complicated.

In addition, wafer fab is a very large-scale manufacturing system consisting of 20 to 30 bays interconnected by an inter-bay material handling system (MHS). Each bay has tens of process tools, interconnected by intra-bay MHS [2, 10]. As a consequence, semiconductor manufacturing is very capital intensive. In fact, the cost for building a complete new wafer fab is estimated to be over \$2 billion [1], and work-in-processes (WIPs) in fabs are extremely expensive compared to those in other manufacturing lines. This implies that even small amount of improvement in productivity can result in the large amount of savings. As a result, there is a growing need for the development of a computational tool that can support the effective and efficient design and operation of 300mm fabs, in order to minimize risks associated with the transition to 300mm while maximizing the benefits from it.

This paper presents our current research effort to meet such a need. Our prototype fab simulator, to be called HiFiVE-300 (High-Fidelity Virtual Environment for 300mm Wafer Fabrication), is a web-based, distributed simulator that supports high-fidelity simulation of the complex, large-scale 300mm fabs. Unlike existing fab simulators, HiFiVE-300 aims to provide the high-fidelity modeling capability for operational aspects, including (i) equipment-level behavioral aspects such as batching, setups, reworks, processing errors, preventive maintenances, and tool failures, and (ii) system-level aspects such as effects resulting from the behavior of MHS and supporting staff, and the complex resource allocation dynamics due to the routing flexibility and multiple resource sharing. Furthermore, configured as a distributed simulator, HiFiVE-300 is able to support rapid prototyping and high-speed simulation of large-scale

high-fidelity fab models. Therefore, owing to its scalability, it is expected that HiFiVE-300 will provide more realistic simulation results within reasonable amount of time for better decision support.

This paper is organized as follows: Section 2 describes the approaches taken to systematically deal with the scalable and distributed fab simulator development problem. The detailed architecture and protocols of HiFiVE-300 are presented in Section 3. Finally, some discussions for the future extensions are given in Section 4.

2 Our Approaches

In order to systematically deal with the complexity arising from the large-scale 300mm fab configurations, we use a hierarchical decomposition approach. This decomposition is suggested naturally by the wafer fab layout itself [13]. Figure 1 shows the proposed approach in which a fab is decomposed into a set of participating units corresponding to the material flows taking place at the bay and fab-level, as well as in the MHSs inter-connecting the various units. We remark that each bay can then be further decomposed into a set of units, such as process tools and material handling devices inside the tools.

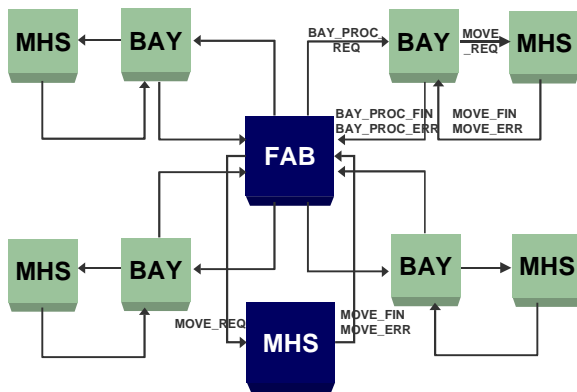


Figure 1. The hierarchical decomposition of wafer fab

Consequently, the approach transforms the complex software design problem into the ones that can be managed more easily. More specifically, in our framework, the fab simulator design problem reduces to the problems of defining (i) the behavior of constituent domains, and (ii) communication protocols which will establish the effective interaction of the various domains.

Being a software realization of a real 300mm fab, HiFiVE-300 requires a fully automatic control system responsible for coordinating its operations. To address the control problem in a domain more systematically, each domain is abstracted to a resource allocation system (RAS) [16]. RAS is an abstraction in which each domain is modeled as a discrete event dynamic system consisting of finite set of resources and concurrently executing processes competing for the resources. This formal characterization of each domain's behavior subsequently allows formal analysis and the development of control policies to establish effective and efficient fab operations. In our past work, we have established that the fab operations can be represented through a hierarchy of RAS domains. We proposed various RAS classes and corresponding control policies that can ensure logically correct and deadlock-free behavior for the domains defined in the proposed approach. For details, the reader is referred to [11, 15, 16].

Specifically, the fab domain is abstracted to an RAS in which dynamically routed FOUPs are competing for the finite bay stockers. In a similar way, FOUPs competing for process tools and auxiliary tools like reticles constitute a bay RAS. Furthermore, in an MHS domain having more than one transporter, the transportation network segments naturally correspond to the set of resources required for the transporters to finish their trips. For all those RASs defined above, it is clear that the control system should be responsible for logical issues like deadlock-free buffer allocation and collision-free transporter routing, as well as efficiency considerations like throughput maximization, WIP reduction, and transportation delay minimization.

Having established the hierarchical structure for modeling fab operations, we apply model-view-control (MVC) design paradigm to the underlying structure to further achieve modularity and reusability of the proposed approach. The MVC paradigm is a framework that divides an entire simulation system into three components, namely model, view and control [8]. In this framework, the model component is an abstraction of state and behavior of the system, while the view provides high quality run-time animation and supports user interaction handling. Finally, the control receives the state information from the model, and issues commands to satisfy some predefined qualitative / quantitative objectives. When the MVC framework is applied to the domains defined in our approach, fab facilities domain can be mapped to the model, since they contain the complete information of system states and represent the non-deterministic behaviors. All other domains, namely

the fab, bay, and MHS, are mapped to the control, as they are responsible for coordinating the operations within their domains by issuing appropriate control commands. It should be noted here that the interactions among the hierarchically decomposed controls should be provided to guarantee correct and efficient fab operations, and they are dictated by the communication protocols.

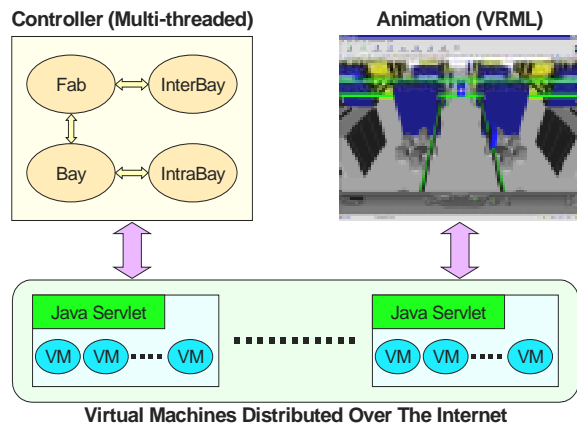


Figure 2. Model-View-Control in HiFiVE-300

Figure 2 shows the MVC separation taken in our approach. We have adopted VRML as our 3-D visualization tool. The virtual machines that correspond to the fab model are implemented as Java servlets so that they can be distributed over the Internet as necessary to achieve scalable simulation. By following the MVC approach, we are able to separate fab-dependent behavioral model from the control system and view, and thereby develop a distributed simulator that can be re-used and updated modularly.

3 HiFiVE-300 Architecture and Protocols

HiFiVE-300 is a web-based, distributed simulator that provides a structured and integrated environment for the modeling, analysis, and control of the 300mm fabs, based on the approaches outlined in the previous section. Some characteristics that differentiate HiFiVE-300 from existing semiconductor fab simulator include (i) the rapid prototyping capability through the provision of well-defined, high-fidelity fab modeling primitives, (ii) fast execution of large-scale fab simulation by exploiting scalability, and (iii) availability of plug-and-playable controller through the separation of controller from the fab model.

Specifically, HiFiVE-300 is composed of one off-line design module, called Fab Designer, that allows fab (re-)configuration, and three run-time modules, namely Virtual Machine Model, Controller, and Animation, that supports real-time, interactive simulation. In this section, we present detailed architecture and protocols as well as implementation issues for each module.

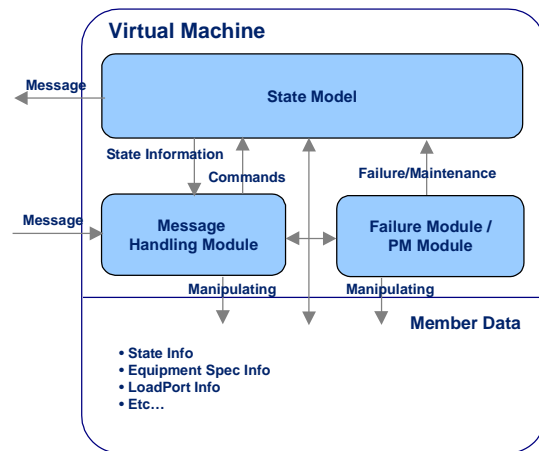


Figure 3. Internal structure of VMM

The Virtual Machine Model (VMM)

Virtual Machine (VM) is a model-based software module running on the server side, simulating the behavior of fab operator or equipment such as process tool, transporter, and stocker [7]. Each virtual machine maintains its own state model and communication channel, and it plays a role of the fab model component. In particular, since virtual machines are designed with well-defined interfaces, and implemented to run on the server side independently, they are robust and reusable with respect to the changes in behaviors and control logic. Hence, they can be modified, replaced, and/or further refined to obtain higher fidelity without affecting other modules in the system.

Figure 3 shows the internal architecture of the virtual machine. The virtual machine object consists of member data and three main threads: StateModel, MessageHandling, and Failure/Preventive Maintenance. The virtual machines interact directly with bay controller through message communication. When a message delivered to the virtual machine from the bay controller, the message is interpreted in the MessageHandling thread, and if the message is valid, it calls proper methods to change member data and/or current state in StateModel. While performing these

steps, if there are events to be reported to the controller, it broadcasts messages containing event information to the bay controller and real-time animation.

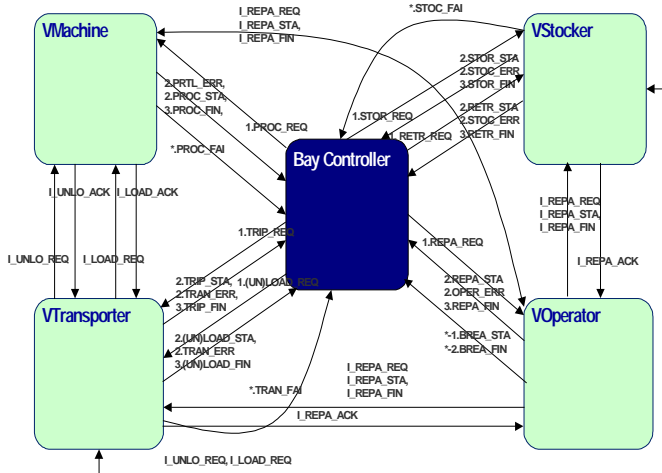


Figure 4. Message protocol between bay controller and virtual machines

The protocol governing the interaction with the bay controller is materialized by a well-defined set of messages exchanged between them. Figure 4 shows the protocol implemented for interacting between bay controller and virtual machines. As an example, we consider VStocker representing a bay stocker in a 300mm fab. Its behavioral definition in terms of a state-transition graph is depicted in Figure 5. When the bay controller sends a message (RETR_REQ) for retrieving a FOUP, if VStocker is not in a proper state (IDLE state, in this example), it sends an error message (STOC_ERR) to the bay controller. If the current state of VStocker is IDLE, then it starts simulating the retrieval operation based on its state-transition model. Before changing state to RETRIEVING, VStocker sends a message (RETR_STA) to bay controller for reporting state change, and afterwards it sends a message (RETR_FIN) again to the controller and its state transits to IDLE state, after finishing the retrieval process.

From an implementation point of view, the virtual machine model is a set of Java servlet programs running within a Java Web Server (JWS). A servlet is a small Java application that runs on the server side to extend and enhance the web server functionality. This technology provides a component-based, platform-independent method for building interactive web applications [14].

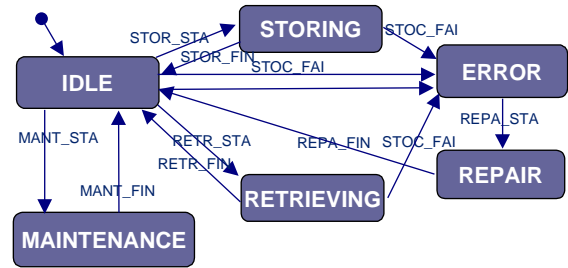


Figure 5. State transition of the VM representing a stocker

The Controller

The controllers are responsible for ensuring logically correct and high-performance operations in their respective domains. In HiFiVE-300, there are four different controller classes according to the different hierarchy in the proposed decomposition framework. At the top level, Fab controller supervises the bay-level material flow by allocating FOUPs to appropriate bay stockers, and Inter-Bay MHS controller coordinates the inter-bay transporters to achieve efficient, deadlock-free material transportations. A bay controller is the controller at the next level that interfaces with the fab controller as well as its subordinate process tools.

A bay typically consists of a number of process tools, stockers, and an MHS facilitating the intra-bay transfer of FOUPs and auxiliary tools. FOUPs are dynamically routed to the different process tools and stockers, according to their recipes, and auxiliary tools like reticles are delivered to the process tools when they are required for processing steps. At the same time, there might be more than one processing tools capable for executing a step, or a FOUP might need to be re-directed to a stocker anytime. Furthermore, additional behavioral constraints may be imposed on the bay operations, such as batch processing requirements at a furnace tool. The operational characteristics described above was formally modeled and verified using the Colored Petri Net (CPN) [6] in our past work. The proposed bay controller CPN model in [12] is capable of assuring the logically correct behaviors for the RASs characterized by routing flexibility, multiple resource acquisitions, and behavioral constraints, and it is encoded in HiFiVE-300 bay controller.

On the other hand, the performance-related control decisions, including lot induction, lot dispatch-

ing, and dynamic lot routing, can be re-configured in HiFiVE-300 controllers in terms of specifying one of available dispatching rules.

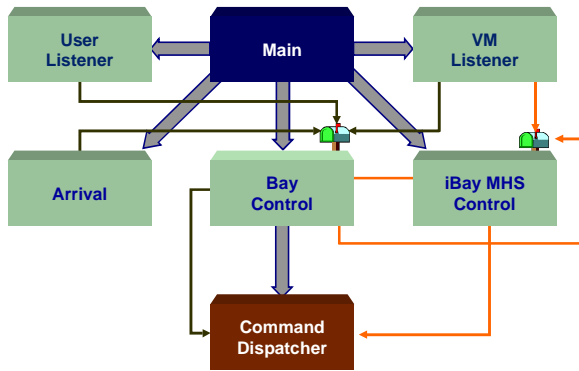


Figure 6. Internal architecture of the bay controller

The entire control system is designed as a Java applet [4], and each individual controller is implemented by use of multiple communicating threads. For instance, the internal architecture of a bay controller is shown in Figure 6 where seven concurrently running threads are defined to effectively handle multiple asynchronous events. The proposed architecture associates message queues to two important threads, bay and intra-bay MHS controls, so that they can keep track of messages generated from other threads while they are busy.

The Real-Time Animation

In simulation study, animation tools play roles not only for simple graphical visualization but also for model verification. For these purposes, HiFiVE-300 provides a 3-D graphical animation tool. This animation tool is implemented as a Java applet [4] with a communication channel connected to virtual machines. Through this channel, the animator receives messages, which are broadcasted from virtual machines, and changes the graphical presentation of system states in real time according to the contents of messages.

We have adopted VRML technology supporting 3-D features and flexible view changes that enable users to see the animation with various angles [17]. In addition, since it has been implemented as a Java applet program, users can remotely monitor the current simulation model using Internet web browsers. Because HiFiVE-300 has been implemented based on MVC paradigm, this animation tool (view component in MVC) can be replaced by new tools having better

functions, without changing any program in other components such as controller and virtual machines.

Figure 7 shows a snapshot for bay model animation running on Internet browser.

The Fab Designer

Fab Designer is an independent, web-enabled applet that provides an integrated design environment to configure a fab simulation model. A fab model is configured by specifying (i) the design parameters, including the fab, bay, and inter/intra-bay MHS layouts, (ii) the operation parameters such as lot and recipe definitions, and (iii) the control parameters represented by the selected logical and performance control policies for all control domains. Therefore, following the MVC framework proposed in Section 2, HiFiVE-300 explicitly distinguish between the model data and the applied control data.

Once the entire fab model is configured, Fab Designer can verify the input, and eventually export it to the central database so that it can be imported for simulation runs and/or further modifications.

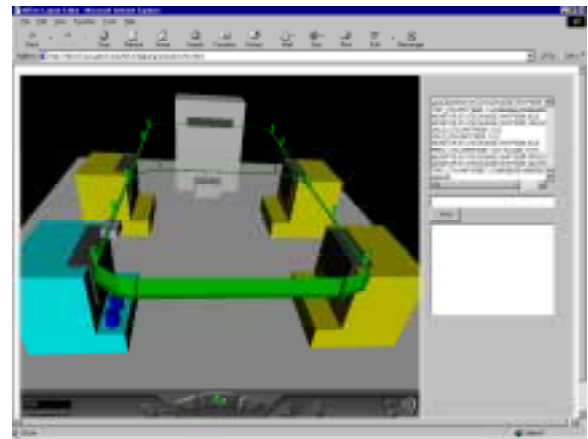


Figure 7. 3-D animation applet using VRML in HiFiVE-300

4 Discussion

The research project outlined in this paper is a major effort undertaken in the *Keck Virtual Factory Lab*, in *School of Industrial & Systems Engineering*, at *Georgia Institute of Technology*. In this project, we have studied formal methods for contemporary semiconductor manufacturing systems, and designed and implemented simulation tools for testing the effectiveness of the formal methods and supporting practi-

cal problem solving. Due to the distributed architecture based on MVC paradigm, the simulator could support the rapid modeling for large-scale fab models. In the near future, we will improve our prototype HiFiVE-300 in terms of fidelity, running speed as well as stability.

However, as the fidelity for virtual machine model increases, the phenomenon lowering system performance is inevitable within the current structure. As an alternative to solve this problem, we are considering applying PADS (Parallel and distributed simulation) [5] technology in virtual machine model. For example, if the virtual machine model will be developed as highest fidelity like real semiconductor equipment, a logical processor (LP, i.e. a simulator running in distributed environment) of each virtual machine should be distributed over the network for fast execution of high-fidelity fab models. For this case, we need more related research with industry for the methodology to distribute and design high-fidelity virtual machine.

Acknowledgments

This work is partially funded by *The W.M. Keck Foundation* and *Ford Motor Company*. We also appreciate *PRI Automation*'s help in this research.

References

1. J.Cameron, The Coming of Fab-wide Automation, *European Semiconductor*, vol.20, no.5, pp.21-22, 1998.
2. C.Cunningham, R.Wright, K. Benhayoune, E.Campbell, V.Swaminathan, and R.White, 300mm Factory Layout and Automated Material Handling System Analysis, *Proceedings of the Autosimulations Symposium*, 1999.
3. J. Ferrell and M Pratt, I300I Factory Guidelines: Version 5.0, *International SEMATECH*, 2000.
4. D. Flanagan, *Java in a Nutshell*, 2nd ed., O'Reilly, 1997.
5. R. M. Fujimoto, *Parallel and Distributed Simulation Systems*, Wiley Inter-science, 2000.
6. K. Jensen, *Coloured Petri Nets: Volume 1*, 2nd ed. Springer, NY, 1997.
7. H. Kim, H.X. Du and C. Zhou, Virtual Machines for Message Based Real-Time and Interactive Simulation, *Proceedings of the 2000 Winter Simulation Conference*, Tampa, FL, 2000.
8. G.E Krasner and S.T. Pope, A Cookbook for using model-view controller user interface paradigm in Smalltalk-80, *J. Objective-Oriented Program*, vol.1, no.3, pp. 26-49, 1988.

9. S.H. Lu and P.R. Kumar, Distributed Scheduling Based on Due Dates and Buffer Priorities, *IEEE Transactions on Automatic Control*, vol.36, no.12, 1991.
10. J. Nestel-Patt, *The Final Automation Wave*, Robotics World, Spring, 1998.
11. J.Park, and S.A.Reveliotis, Liveness-Enforcing Supervisors for Resource Allocation Systems with Re-works Forbidden States, and Uncontrollable Events, submitted to *IEEE Transactions on Robotics & Automation*, 2000.
12. J. Park, S.A. Reveliotis, D. A. Bodner, C. Zhou, and L. F. McGinnis, A Colored Petri Net-based Approach to the Design of 300mm Wafer Fab Controllers, *Proceedings of the IEEE International Conference on Robotics & Automation*, Seoul, Korea, May 2001.
13. J. Park, S.A. Reveliotis, D.A. Bodner, C. Zhou, J. Wu, and L.F. McGinnis, High-Fidelity Rapid Prototyping of 300mm Fabs through Discrete Event Systems Modeling, *Computers in Industry*, 1528, pp.1-20, 2001.
14. A. Patzer, et al., *Professional Java Server Programming: with Servlets, JavaServer Pages (JSP), XML, EnterpriseJavaBeans (EJB), JNDI, CORBA, Jini and Javaspaces*, Wrox Press, 1999.
15. S.A. Reveliotis, and P.M. Ferreira, Deadlock Avoidance Policies for Automated Manufacturing Cells, *IEEE Transaction on Robotics & Automation*, vol.12 no.6, pp845-857, 1996.
16. S.A. Reveliotis, M.A. Lawley, and P.M. Ferreira, Polynomial Complexity Deadlock Avoidance Policies for Sequential Resource Allocation Systems, *IEEE Transaction on Automatic Control*, vol.42, no.10, pp1344-1357, 1997.
17. Web 3D Consortium, Documents for the Virtual Reality Modeling Language, <http://www.vrml.org/>, 2001.