Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

TIME MANAGEMENT IN DISTRIBUTED FACTORY SIMULATION, A CASE STUDY USING HLA

Ke Wang
Sheng Xu
Leon F. McGinnis

School of Industrial & Systems Engineering
765 Ferst Drive, N.W.
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

ABSTRACT

Distributed simulation is attractive for modeling large scale manufacturing systems. However, synchronization requirements present distinct challenges in designing and implementing a suitable modeling structure. Our experience in building a distributed simulation model for 300mm wafer fab using the High Level Architecture (HLA) shows that time management should be considered at both the design and the implementation stages. This paper proposes an optimistic-conservative synchronization scheme by using model specific information at the implementation stage. Computational results are then presented to show that properly implemented distributed factory simulation can achieve significant reduction in model execution time.

1 INTRODUCTION

A semiconductor wafer fab is a very complicated system with many tools and products. Today, the investment for a single 300mm wafer fab can exceed $4B (Semiconductor-technology.com). Due to the high investment and complicated operations, simulation has been adopted in many cases as a viable design and analysis tool to achieve better productivity (Fowler et al. 1998).

Ideally, simulation should support rapid modeling and fast execution of high-fidelity fab models in order to be able to provide realistic simulation results with acceptable computational costs and lead times (Kim et al. 2001). Therefore, distributed simulation, because of its convenience in developing modules independently and increased parallelism on the computational side, looks very attractive. However, synchronization requirements present distinct challenges in designing and implementing a suitable modeling structure. Our experience in building a high fidelity virtual environment for 300mm wafer fab (HiFiVE) provides a good case study.

The objective of this paper is to discuss the time management issues that should be considered in the modeling and implementation stages of a distributed factory simulation. The remainder of this paper is organized as follows. Section 2 introduces High Level Architecture (HLA) and its time management services. Section 3 describes the 300mm wafer fab we model. Section 4 presents the model structure. Section 5 describes a new optimistic-conservative synchronization scheme we implemented in HiFiVE. Section 6 presents some computational results. Section 7 concludes with thoughts on further research.

2 HLA AND ITS TIME MANAGEMENT SERVICES

The high level architecture (HLA) is a reusable software architecture for the development and execution of distributed simulation applications. It defines an elementary simulation unit as a federate, and a set of federates working together as a federation .

HLA is an architecture, not software (Turrell). As an implementation of the HLA specification, a runtime infrastructure (RTI) software is required to support operations of a federation execution. The RTI software provides a set of services used by federates to coordinate their operations and data exchange during a runtime execution, and time management services are one important category of these services. A brief tutorial of HLA time management and synchronization is available in McGinnis (2004).

Time management is concerned with the mechanisms for controlling the advancement of time during the execution of a federation. For the purpose of this paper, we only focus on time stamp order (TSO) service and next event request (NER) service. Detailed discussions on other HLA time management services are available in Fujimoto and Weatherly (1996).

2.1 TSO Service and Lookahead

In operation, federates interact through messages. A message from one federate to another may represent an event originating with the first federate but affecting the second. Thus, when the second federate receives the message, it can "experience" the corresponding event by placing it in its local event queue. Synchronization requires that the second federate's local time should not be greater than the time stamp of the arriving message, otherwise the second federate experiences an event "now" that occurred in its "past" (McGinnis et al. 2005). HLA TSO service is designed to facilitate the message interactions between federates. It ensures that messages will be delivered to federates in time stamp order, and no TSO message contains a time stamp less than the federate's current time will be delivered to it.

The TSO service requires specification of lookahead, where lookahead is defined as the minimum distance into the future that a TSO event will be scheduled (Fujimoto 2000). Lookahead is a guarantee from the federate that it will not generate any messages with a time stamp smaller than its current time plus the value of lookahead. Thus lookahead can be used to define a lower bound on the time stamp (LBTS) of messages produced by a federate later during the execution. By computing a minimum of this lower bound across all federates that can send messages to a given federate, call it federate S, one can derive a lower bound on the time stamp of messages that will be delivered to S in the future. This lower bound is important because it means S can process messages with time stamp smaller than this lower bound without fear of later receiving a smaller time stamped message (Fujimoto 1997).

The lookahead of a federate is bounded by the federate's quickest response time to TSO messages it subscribes. If, for example, the federate should respond to messages from other federates instantly, it can only guarantee a zero lookahead. Lookahead value has great implication for simulation run time. If lookahead value is large, federates potentially can achieve a high degree of parallelism in processing events. However, if lookahead value is small, there will be a high level of synchronization overhead and little parallelism.

2.2 NER Service

Next event request is a service for advancing logical time of federates. When used in an event driven federate, NER parameter t usually indicates the time stamp of the next local event within the federate. By invoking NER with parameter t, NER(t), the federate is making a conditional guarantee that if it does not receive any additional TSO messages in the future with time stamp less than t, the federate will not later generate any TSO messages with time stamp less than t plus the federate's lookahead (Fujimoto and Weatherly 1996). A time advance grant will then be issued to indicate the completion of the NER(t). If no present or future TSO message has a time stamp less or equal to t, the RTI will grant the federate to advance to logical time t; Otherwise, the next smallest TSO message destined for this federate will be delivered, and the logical time of the federate will be granted to advance to the time stamp of that TSO message.

3 HIFIVE SYSTEM DESCRIPTION

The high fidelity virtual environment for 300mm wafer fab (HiFiVE) is our attempt to create a "virtual wafer fab", a facility in which we can test new ideas for fab layout, new methods for real-time control, new approaches to lot release and scheduling, and investigate newly emerging problems such as reticle management The fab we currently model and test has the same spine structure as the Sematech base-line model (Sematech), with 310 tools distributed in 24 bays. The fab layout is illustrated in Figure 1.

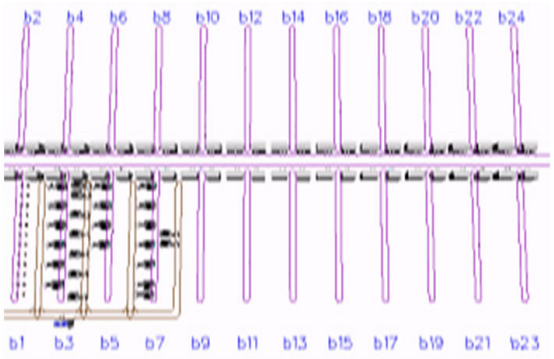

Figure 1: Fab Layout

The fab processes 14 types of products, each with 316 process steps. It uses overhead hoist transporters (OHTs), and has 58 OHTs for transportation between bays, and one to three OHTs for transportation within a bay, depending on the number of tools in the bay.

Considering the complicated operations of the fab, HiFiVE is designed for distributed computing, using HLA. To avoid the significant workload of state saving and rollbacks, we use conservative synchronization supported by HLA. In more detail, we model our system as several independent simulation units, or federates. The federates interact with each other by publishing and subscribing TSO messages. Each federate advances its logical time by invoking NER service.

4 MODEL STRUCTURE

Our experience in HiFiVE shows that different model structures can differ greatly in synchronization conven-

ience, and therefore have significantly different synchronization overhead and model execution time. We tested two model structures in HiFiVE. Testing result shows that design structure that considers the tradeoff between module independency and synchronization convenience can achieve much better model execution speed (McGinnis et al. 2005). The optimistic-conservative synchronization we implemented in HiFiVE is therefore based on a design structure that gives enough considerations to synchronization convenience.

4.1 Model Structure

Figure 2 shows the model structure we use. In this design, we divide the fab into two major federates, the factory federate and the material handling federate. The factory federate simulates the physical behavior of the tools, including the breakdowns, load/unload activities at the tools, and processing lots at tools. The material handling federate contains three relatively independent sub-units, fab control unit, bay control unit and transporter unit. These sub-units work together and are responsible for all issues related to the control and physical delivery of materials inside the fab, including the lot release control, lot dispatch control, OHT dispatch control, OHT movement control, and even load/unload activities at the stockers.

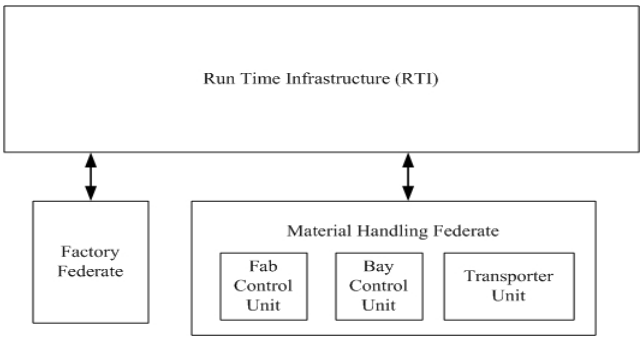

Figure 2: HiFiVE Design Structure

4.2 Model Lookahead

In this design, since the factory federate is mainly in charge of the behaviors at the tools, and the smallest processing time of a lot on the tools is more than 8.1 minutes, the lookahead value of the factory federate can be as large as 8.1 minutes. For the material handling federate, the lookahead value depends on how fast the fab can send the material to its destination, which in most cases is the summation of transportation time and load/unload time at the stockers. Detailed calculations show that lookahead value of the material handling federate can be up to 52 seconds in our case (Wang 2004).

5 OPTIMISTIC-CONSERVATIVE SYNCHRONIZATION

5.1 Objective and Basic Idea

We notice that the material handling federate in our design has many events in a small logical time interval. Therefore, when using the regular NER(t) service, since t is the time stamp of next event in the local event queue, the federate only requests to advance a very small amount of time, which chops the simulation time into very small pieces, and limits the parallelization of the system.

The objective of the optimistic-conservative synchronization scheme we propose here is to achieve better parallelism than the regular conservative synchronization without introducing too much additional workload for status saving and roll back. Specific information about the system we are modeling is used to implement the optimistic-conservation synchronization scheme.

The basic idea of the optimistic-conservative synchronization includes two parts. We have a federate F that must process many local events to advance to a time larger than the time stamp of the next "external" event, or TSO message. We would like to make it so F can process all or most of these local events with a single time advance request. To accomplish this, we artificially inflate the time in the NER request from F. Conceptually, this allows federate F to process all events up to the time advance grant. However, in reality, F now may generate a TSO that arrives in the "past" of some other federate. Thus, a "compensation" is required in the receiving federate of TSO messages from F.

5.2 Time Advancing Request Scheme

In our optimistic-conservative synchronization, whenever the material handling federate, call it F_m, finishes processing all local events at current logical time T and is ready to advance to a new logical time, it calls NER(t+s), where t is the time stamp of next local event of F_m , and s is a fixed value we choose based on an analysis of the federate's dynamic behavior, described further below.

The RTI, upon receiving NER(t+s), will check if there are any TSO messages subscribed by F_m with time stamp less than or equal to t+s. If there are such TSO messages, the smallest next TSO message will be delivered, and F_m will be granted to advance to time T', where T' is the time stamp of the delivered TSO message, and $T \le T' \le t+s$. Otherwise, no TSO message will be delivered, and F_m will be granted a time advance to t+s.

Suppose RTI now grants F_m to advance to a new logical time T', it is then prohibited that the federate sends out any TSO messages with time stamp less than T'+L, where L is the lookahead value of F_m. However, the smallest possible time stamp of TSO messages F_m might want to send

out can be as small as t+L. To address this problem, we add one more field in the messages sent out by F_m. If the message should have a time stamp, call it ts, and ts ≤T'+L, we set the time stamp of the message to be T'+L, and include ts as a field of the TSO message. Since the smallest possible "correct" time stamp is t+L, and T'≤ t+s , we introduce at most s "error" in the time stamp by using this new time advancing request scheme.

5.3 Compensation Scheme

A TSO message sent out by the material handling federate F_m might not have the "correct" time stamp, but it contains a field that allows us to retrieve the "correct" time stamp value ts. In this part, we will address the compensation problem in the factory federate, call it F_f.

Suppose F_f is now granted to advance to time T', and it receives a TSO message from F_m with time stamp T'. As a response, F_f will schedule a future TSO event at time T'+p representing the finish of the process, where p is the processing time.

In the optimistic-conservative synchronization, however, the time stamp T' of the received TSO message can be larger than its "correct" value ts. It is trivial to see that if we want the factory federate to send out TSO messages with "correct" time stamp, we should schedule a future event at time ts+p, instead of T'+p as usual.

The problem then is: can we always successfully schedule a future event at time ts+p? Suppose the lookahead value of factory federate is L_f ; any scheduled future TSO event should have a time stamp of at least T'+ L_f . Otherwise, the factory federate will have problem in sending out the TSO message. Therefore, to ensure that we can successfully schedule the future TSO event "correctly" , we require T'+ L_f ≤ ts+p, that is: T'-ts+ L_f ≤ p.

According to the time advancing request scheme we described before, the difference between T' and ts can be at most s. Therefore, if we choose s and L_f such that : s+ L_f ≤ Min{p}, we can always successfully compensate the time stamp "error" in the factory federate.

Applying this rule in our case study, we have Min{p} = 8.1 minutes, we therefore choose s = 4 minutes, and L_f = 4.1 minutes.

5.4 Conservative Synchronization vs. Optimistic-Conservative Synchronization

Figure 3 is an illustration of how the traditional conservative synchronization and our optimistic-conservative synchronization works differently.

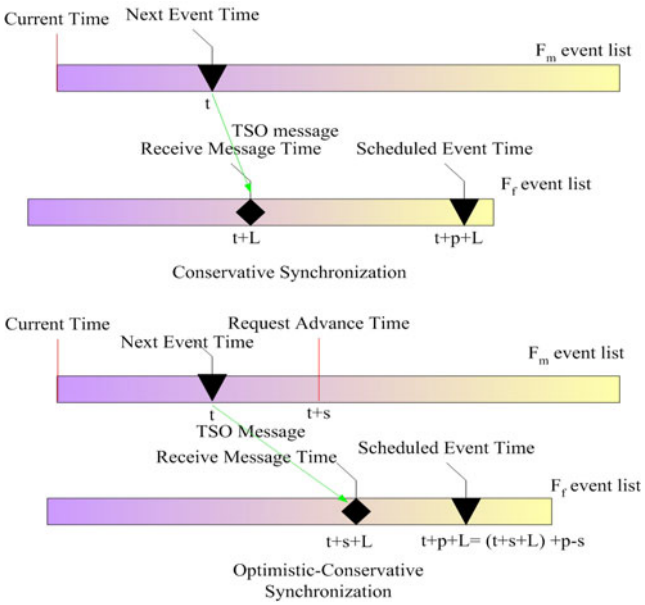

Figure 3: Conservative Synchronization vs. Optimistic-Conservative Synchronization

As shown in Figure 3, the optimistic-conservative synchronization does not change the scheduled TSO event time in federate F_f , but potentially allows federate F_m to process multiple events with a single time advance request. Therefore, the optimistic-conservative synchronization reduces the overall synchronization overhead of the system and can potentially improve the parallel level of the system when simulation is distributed on multiple machines.

6 COMPUTATIONAL RESULTS

We implemented both the conservative synchronization and the optimistic-conservative synchronization on the design structure described in section 4, and tested the model speed. During these tests, we record the current CPU clock time when the simulation time goes forward for 1000 minutes. A speed factor is used to represent how fast the simulation can run, and is defined as following:

Speed Factor = log (Current Simulation Time / Cumulated CPU Clock Time)

Figure 4 shows the result of single machine test.

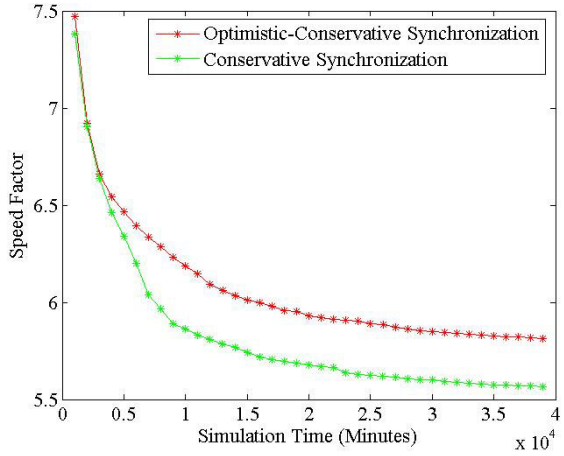

Figure 4: Single Machine Computational Result

The single machine test shows that optimistic-conservative synchronization can reduce the model execution time by 30% in our case.

Furthermore, significant benefit in parallel computation is also observed for the optimistic-conservative synchronization. The test result shows that the optimistic-conservative synchronization model, when distributed on two PCs, can run faster than Automod ASAP model. Figure 5 shows the computational test result.

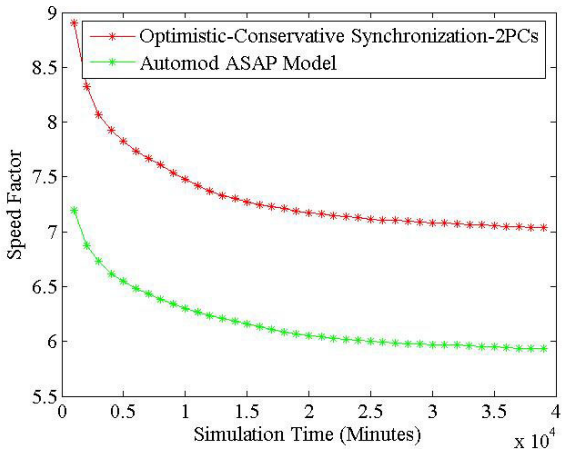

Figure 5: Distributed Test Result

7 CONCLUSION

This paper presents our experience in time management issues when modeling and implementing a distributed factory simulation. It shows that both the independency of the modules and the convenience of time management must be considered at the system design stage. Using specific information about the system in time management of a distributed factory simulation model can sometimes improve the model execution time greatly. Further work involves following:

- Implement and test bidirectional optimistic-conservative synchronization, where factory federate also request to advance to time larger than its next local event.
- Analyze how to choose the optimal s and lookahead values for each federate.

REFERENCES

Fowler, J. W., M. C. Fu, L. W . Schruben, S. Brown, F. Chance, S. Cunningham, C. Hilton, M. Janakiram, R. Stafford, and J. Hutchby. 1998. Operational modeling and simulation in semiconductor manufacturing. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.E Watson, J.S. Carson and M.S. Manivannan, 1035-1040. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Fujimoto, R. M. and R. M. Weatherly. 1996. Time management in the DoD high level architecture. In *Proceedings of the IEEE 10th Workshop on Parallel and Distributed Simulation*. 60-67.

Fujimoto, R. M. 1997. Zero lookahead and repeatability in the high level architecture. In Spring Simulation Interoperability Workshop [online]. Available via <http://www.coc.gatech.edu/computing/pads/PAPERS/Zero_Lookahead_and_Repeatability.pdf> [accessed April 8, 2005].

Fujimoto, R. M. 2000. *Parallel and distributed simulation systems*. New York: John Wiley & Sons.

Kim, H., J. Park, S. Sohn, Y. Wang, S. Reveliotis, C. Zhou, D. A. Bodner, and L. F. McGinnis. 2001. A high-fidelity, web-based simulator for 300mm wafer fabs. In *2001 IEEE International Conference on Systems, Man, and Cybernetics*. 1288-1293 .

McGinnis, L. F., K. Wang, and S. Xu. 2005. Distributed simulation modeling of large scale manufacturing systems. In *Proceedings of the 2005 IIE Annual Conference*. Atlanta, U.S.A., May14-18, 2005.

McGinnis, L. F. 2004. Distributing a large-scale, complex fab simulation using HLA and java: issues and lessons. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters, 807-811. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Sematech. *Fab Simulation Models*. [online]. Available via <http://www.ismi.sematech.org/modeling/simulation/index.htm> [accessed April 8, 2005]

Semiconductor-technology.com. Elpida Memory 300mm Wafer Fab, Hiroshima, Japan [online]. Available via

<http://www.semiconductor-technology.com/projects/elpida> [accessed April 2, 2005].

Turrell, C. High level architecture [online]. Available via<http://www.sisostds.org/webletter/siso/iss_18/art_149.htm> [accessed April 3, 2005].

Wang, K. 2004. Response time of HiFiVE federates. HiFiVE Internal Report August 2004, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA

AUTHOR BIOGRAPHIES

LEON F. MCGINNIS is Eugene C. Gwaltney Professor of Manufacturing Systems at Georgia Tech., where he also serves as Associate Director of the Manufacturing Research Center and founding Director of the Keck Virtual Factory Lab. His research focuses on the application of operations research and computer science to solve decision problems arising in the design and operation of industrial logistics systems. He can be reached via email at <leon.mcginnis@isye.gatech.edu>

KE WANG earned her BS in Electronic Engineering at ShangHai JiaoTong University, China, and Ph.D. in Management Engineering at ZheJiang University, China. She is currently a Ph.D. student at Georgia Institute of Technology. She is also working as a graduate research assistant at Georgia Institute of Technology, and her current research focuses on modeling and simulation of large scale manufacturing systems. She can be reached via email at <kwang@isye.gatech.edu>.

SHENG XU earned his M.ENG. in Mechanical Engineering and Automation at Zhejiang University, China, M.ENG. in High Performance Computing at National University of Singapore (the Singapore-MIT Alliance Program). He worked as an Associate Research Fellow/Research Engineer in Singapore Institute of Manufacturing Technology. He also worked as a software engineer, senior program designer, systems analyst etc. for many years in China. He is currently a Ph.D. student at the Georgia Institute of Technology, and his current research focuses on modeling and simulation of manufacturing and logistics systems. . He can be reached via email at <sxu@isye.gatech.edu>.