Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

HIERARCHICAL DISTRIBUTED SIMULATION FOR 300MM WAFER FAB

Sheng Xu
Leon F. McGinnis

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

ABSTRACT

Distributed simulation promises benefits in large-scale simulations, such as in high fidelity simulation of 300mm wafer fabs, although these benefits have been hard to achieve in practice. This paper examines the fundamentals of distributed simulation, and proposes a hierarchical approach to distributed wafer fab simulation, which has the potential to achieve significant reduction in model execution time.

1 INTRODUCTION

Simulation is widely used in analyzing manufacturing systems to provide a comprehensive view of the system's overall performance. High fidelity simulation incorporates representation of material handling, as well as production processes and shop floor control and is especially important for technically complex and highly capitalized manufacturing, such as 300 mm wafer fabs. High fidelity simulation applied to 300mm wafer fabs can support increasing the system throughput and resource utilization, and reducing work-in-process and cycle time.

Prohibitively large run times prevent simulation models from achieving their full potential (Schruben and Roeder 2003). To overcome this difficulty, a number of approaches have been suggested, and one of them is to use parallel and distributed computing technology. One obvious advantage of this is the ability to run more complex simulation models faster.

The High-Fidelity Virtual Environment for 300mm Wafer Fabrication (HiFiVE) project addresses distributed simulation to support rapid prototyping and high-speed analysis of large-scale high-fidelity fab models. The goal of the research has been to achieve scalability, and more reliable simulation results within reasonable computation time for better decision support.

2 DISTRIBUTED SIMULATION FOR 300MM WAFER FAB

There are a number of approaches to synchronizing federates in a distributed simulation (see, e.g., Fujimoto, 2000). Because different components in manufacturing systems interact intensively, we choose to avoid approaches that involve rollback. In this section, we will examine the fundamentals of the conservative synchronization approach.

The basic structure of a distributed simulation is a set of "logical processes" (LPs) or federates, which may be executed independently, e.g., on different processes. Because the LPs interact, perhaps by exchanging messages, some mechanism is required to synchronize their execution, so that a given LP never receives messages representing an interaction that should have occurred prior to its current "local time."

2.1 Synchronous Execution

Conservative distributed simulation is based on synchronous execution: during each synchronous phase, the simulation executive will publish a Lower Bound on Timestamp (LBTS) to all LPs, guaranteeing that no event with time stamp smaller than LBTS will be delivered to them during the message exchange stage at the end of this phase. Therefore, it is safe for each LP to process any message or event with time stamp less than LBTS.

Figure 1 provides an example of the conservative distributed simulation. At the beginning of each phase, the simulation executive publishes the LBTS, and each LP finishes all messages and events with time stamp less than LBTS, sends out all messages to other processors generated during the current phase, and sends out the request to the simulation executive to advance to next phase by sending out the smallest timestamp in its own pending event list (PEL).

The simulation executive waits until all LPs signal that they have finished their computation tasks in the current phase. With the smallest timestamp from all LPs, and their lookahead values, the simulation executive can calculate the next LBTS. Then the simulation executive can broadcast LBTS after making sure that all messages sent in the previous phase has arrived at their destination LP.

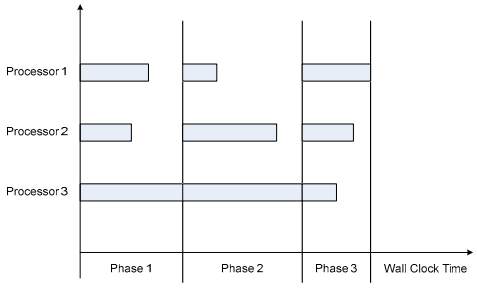

Figure 1: Phases in Conservative Distributed Simulation

We can make the following observation on the conservative synchronization:

- The simulation computation proceeds in phases.
- At the beginning of each phase, all LPs are the same in simulation time, and wall clock. They also have the valid states and the same LBTS, which specified the logical time on which they will be blocked from execution.
- Each LP may take different computation time to finish its computation tasks in each phase, and LPs which finish earlier have to wait for other LPs, i.e. their computing power will be wasted. From the illustration in Figure 1, LP1 and LP2 have to wait for LP3 to finish its computation tasks before all of them can move to next phase.
- During each phase, the LP with the longest computation time will dominate the execution time for each phase. Therefore, it is desirable to allocate computation tasks evenly across all LPs, so that the time spent in waiting for other LPs can be minimized.
- The length of the simulation time for each phase is determined by the LBTS calculated by the simulation executive. The length of the simulation time across different phases can be different because the difference in the timestamp of the earliest event in the PEL in each LP, and the lookahead values.
- Phases with smaller length of the simulation time lead to more synchronization overhead thus longer total execution time.
- Smaller phases can lead to more total time wasted in waiting for other LPs to finish their computation tasks. Figure 2 has the same computer time for each LP as in Figure 1, its larger phase length can void some wasted waiting time, and result in shorter overall execution time.

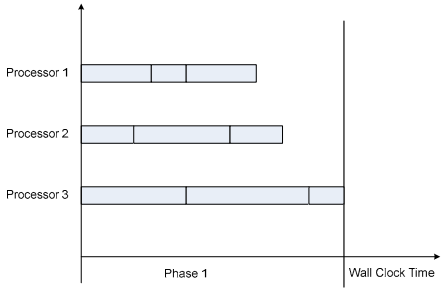

Figure 2: Conservative Distributed Simulation with Fewer Phases

2.2 The Lookahead Value

The lookahead value is critical for larger synchronization phase length in simulation time. It is well known in the distributed simulation community that large lookahead value is crucial for improving the performance of conservative distributed simulation. Therefore, the simulation program must be written to maximize its lookahead value (Fujimoto 2000).

Definition 1 *Lookahead: if a logical process at simulation T can only schedule new events with timestamp of at least T+L, then L is the lookahead for the logical process.* (Fujimoto 2000)

Lookahead value can be derived as follows (Fujimoto 2000):

- Limitations concerning how quickly physical processes can interact with each other.
- Physical limitations concerning how quickly one LP can react to a new event.
- Tolerance to temporal inaccuracies.
- Non-preemptive behavior.
- Pre-computing simulation activities.

However, small lookahead values in some of the LPs can limit the simulation performance even if other LPs have large lookaheads. This is due to the way LBTS is calculated.

Let the set of LPs in a distributed simulation federation be F. For any LP $i \in F$, let T_i be the current simulation time in LP i, t_i be the time stamp of the smallest time stamp event in LP i's pending event list, and L_i be the lookahead value for LP i. Then the LBTS is calculated by:

$$LBTS = \min_{i \in F}(t_i + L_i)$$

Even if t_i are comparable in different LPs, the LPs with smallest lookahead values will determine the LBTS values, thus the length of synchronous phases in simulation time.

In conservative synchronization approaches, larger lookahead values in some LPs may not be able to boost the simulation performance. Therefore, the optimistic-conservative synchronization approach is proposed to exploit the larger lookahead values in some LPs.

2.3 The Request Ahead Value

In conservative synchronization approaches, we impose the local causality constraint on each LP to ensure the correctness of the simulation results:

Definition 2 *Local Causality Constraint: A discrete-event simulation obeys the local causality constraints if and only if each LP processes events in nondecreasing time stamp order.*(Fujimoto 2000)

The local causality constraint is a sufficient condition to ensure the correctness of simulation results, but it is not a necessary condition. We can find cases where a local causality constraint is violated but the system simulation results are still correct. For example, we have two events to be processed in the same LP: event 1 is to start processing on a machine in bay 1, and event 2 is to start processing on another machine in the same bay. The difference in timestamp for the two events is 1 second. That is $t_1 = t_2 - 1$. Suppose the processing time for both events are in minutes, we can process either of the two events without compromising the correctness of the simulation results.

Based on the above observations, we proposed the optimistic-conservative synchronization approach (Xu and McGinnis 2006).

Definition 3 *The Request Ahead Value: if a logical process at simulation T with lookahead value L can only schedule new events with timestamp of at least T+L+R, then R is the Request Ahead Value for the logical process..*

Definition 4 *Roll-back free compensation value: If a logical process at simulation time T received a message with time stamp T' where $T' < T$, then S is referred to as the rollback free compensation value if no rollback is necessary when $T - T' \le S$.*(Xu and McGinnis 2006)

To show an example of the roll-back free compensation value, we consider the case with one automatic machine and a vehicle. Suppose the vehicle arrives at the machine, and drops a job on the machine, a message with time stamp 10 is sent by the vehicle federate to the machine federate informing it of the drop-off. Suppose further that the machine federate receives the message at its local time 12, i.e., the message is received in its past, which might be dealt with by a rollback. However, rollback may be avoided, e.g., if the minimum processing time for the machine is 10 time units. Then we can schedule the new event in the machine federate with time stamp of at least 20, and no rollback is necessary. The rollback free compensation value can be any value between 0 and 10. For the two boundary values, 0 means no compensation at all, which reduces to the normal conservative synchronization protocol, and 10 will use up all the room for the lookahead value, which is not recommended because zero-lookahead will result. The general approach is to find the balance between the compensation value and the lookahead value. (Xu and McGinnis 2006)

To calculate the request ahead value for LP j, we first define the set of LPs in a distributed simulation federation receiving timestamp messages from LP j as F_j. For any LP $i \in F_j$, let S_i be the rollback free compensation value in LP i. Then the Request Ahead Value for LP j is calculated by:

$$R_j = \min_{i \in F_j}(S_i)$$

Similar to the lookahead value, the rollback free compensation value S is also derived from:

- Limitations concerning how quickly physical processes can interact with each other.
- Physical limitations concerning how quickly one LP can react to a new event.
- Tolerance to temporal inaccuracies.
- Non-preemptive behavior.
- Pre-computing simulation activities.

Clearly, the rollback free compensation value S will affect the selection of the lookahead value. If we define a larger compensation value to an LP, its lookahead value may decrease accordingly.

Therefore, we can exploit the difference in lookahead values across different LPs by imposing compensation values to some of LPs with large lookaheads, which will reduce their lookahead value. On the other hand, we can gain the request ahead value in the LPs with small lookahead values so that the sum of the lookahead and request ahead values across LPs will be more balanced.

One example in exploiting the compensation value is given in (Xu and McGinnis 2006): we have two LPs, a factory federate with lookahead value of 8 minutes, and a material handling federate with a lookahead value of 1 minute. The LBTS is calculated by:

$$LBTS = \min(t_1 + 8, t_2 + 1)$$

The smaller lookahead value in the material handling federate always dominates the LBTS calculation, and results in small LBTS values.

We add 4 minutes of compensation value in the factory federate, which resulted in 4 minutes Request Ahead Value in the material handling federate. At the same time, the lookahead value in the factory federate is reduced to 4 minutes.

Table 1: Comparison of lookahead *L*, compensation value *S* and request ahead value *R*

Conservative Synchronization		**Optimistic-Conservative Synchronization**	
Factory Federate	Material Handling Federate	Factory Federate	Material Handling Federate
L=8	L=1	L=4	L=1
S=0	S=0	S=4	S=0
R=0	R=0	R=0	R=4
L+R = 8	L+R=1	L+R =4	L+R =5

Then the LBTS is calculated as:

$$LBTS = \min(t_1 + 4 + 0, t_2 + 1 + 4)$$

Therefore, we can exploit the difference in lookahead values across different LPs by imposing compensation values to some of LPs, and gaining request ahead values in some other LPs. As a result, larger LBTS values can be expected.

2.4 Minimum FEL Time

In the previous two sections, we studied the lookahead and request ahead values and their relationship to the LBTS calculation. We assume that the time stamp of the smallest timestamp events in the FEL in different LPs are similar.

However, the message density in different LPs can be quite different. Suppose two LPs have significantly different minimum FEL time, i.e.

$$t_1 >> t_2 \text{ and } t_1 - t_2 >> L + R .$$

Then the minimum FEL time will affect LBTS calculation significantly.

As the minimum FEL time is always positive, it may result in larger LBTS value, which can improve the simulation performance.

Based on these observation, we may include LPs with no FEL, and small lookahead value in the distributed simulation federation, i.e. standalone controllers.

Modeling controllers as federates is often difficult in conservative distributed simulation, because of their very small response time, i.e. a fraction of a second. As we discussed in the previous sections, an LP with small lookahead value may dominate LBTS calculation, and result in poor performance in distributed simulation.

However, if we design the controller federate by processing messages based on received order, and other incoming messages are processed in the same phase in synchronous execution, then the minimum FEL time will always be ∞ as there is no event in the FEL. Thus, the controller federate will not affect LBTS calculation even if it has very small lookahead value. For example:

$$LBTS = \min(\infty + L_1, t_2 + L_2)$$

where LP 1 is a controller federate with small lookahead L_1, and LP 2 is a federate with normal lookhead $L_2 >> L_1$.

Although LPs without FEL can have smaller lookahead values, they must send out messages at the same phase as they receive the incoming message triggering the outgoing messages. This requirement does not align with the synchronous execution procedure, and extra manipulation is required. It can be applied in conservative distributed simulation when stand alone controller cannot be avoided.

In distributed simulation for 300mm wafer fab, we avoided using the stand alone controller by combining them with other components in the manufacturing system.

2.5 Maximum Workload

Large LBTS values are important to reduce the synchronization overhead and to improve parallelism in execution. However, the computational burden across different LP in each synchronous phase has significant impact on simulation performance.

According to Amdahl's law in parallel computation, the speedup is limited by the percentage of the sequential part of the program, which must be computed by one processor, with other processors waiting for its completion.

$$speedup \le \frac{1}{f + (1-f)p} = \frac{p}{1 + f(p-1)}$$

Where f is percentage of the sequential part of the program, and p is the number of processors. (Parhami 2002)

When the sequential part is small, the speedup is significant, but when the percentage of the sequential part increases, the speedup becomes less significant. If the majority of the program is sequential, the benefits from parallel computation will be minimized. Amdahl's law is illustrated in Figure 3.

If $f = 0.1$, the speedup will be bounded by a factor of 10 no matter how many processors are used, and even if no synchronization penalty is occurred.

$$speedup_{p\to\infty} \le \lim_{p\to\infty} \frac{p}{1 + 0.1(p-1)} = 10$$

If $f = 0.9$, the speedup will be bounded by a factor of $\frac{10}{9}$ no matter how many processors are used, and even if no synchronization penalty is occurred.

$$speedup_{p\to\infty} \le \lim_{p\to\infty} \frac{p}{1 + 0.9(p-1)} = \frac{10}{9}$$

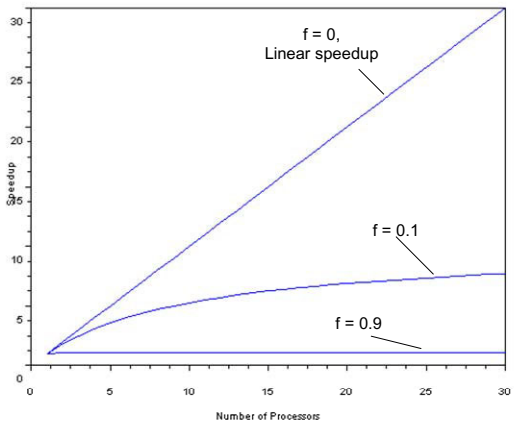

Figure 3: Amdahl's law

Therefore, we should evenly distribute the computation tasks to different LPs so that no single LP has large percent of the total computation tasks, if large synchronization overhead can be avoided.

3 HIERARCHICAL DISTRIBUTED SIMULATION

Various federation design alternatives have been tested in HiFiVE (High-Fidelity Virtual Environment for 300mm Wafer Fabrication) (McGinnis et al. 2005, Wang et al. 2005, McGinnis et al. 2006, Xu and McGinnis 2006). We found that the Factory/AMHS approach with optimistic-conservative synchronization approach delivers best computational performance by reducing the synchronization overhead.

However, the Factory/AMHS approach did not balance the computation tasks across the LPs in the federation; based on the discussion in the previous section, there may be potential to further improve the computaional performance if a more balanced federation can be developed.

In this section, we will first introduce the Factory/AMHS approach, and analyze its performance and limitations. Then an innovative new design with hierarchical distributed simulation is proposed and its correctness is analyzed.

3.1 The Factory/AMHS Approach (McGinnis et al. 2005)

The Factory/AMHS Approach has three federates: the Factory Federate, the AMHS Federate and the Fab Control Federate. The Factory Federate models all process tools, and the AMHS Federate models stockers, Interbay/Intrabay transporters, and the AMHS controller, which manage the flow of the lots in the Fab. The Fab Control Federate models the lot release operations in the fab, simulating the job arrivals to the system.

The reason for such a configuration is mainly from the lessons learned from large computational burden resulting from the previous distributed simulation design, which has a root in the previous non-distributed HiFiVE version. We analyze this design based on the principles we derived in Section 2:

- Lookahead: The Factory/AMHS approach has relatively large lookahead values in both the Factory federate and the AMHS federate, as we incorporated the controllers within them. The Fab Control federate has large lookahead as we precompute the lot release time.
- Compensation and Request Ahead Values: we implemented the compensation value in the Factory Federate, and the Request Ahead value in the AMHS federate.
- Time Complexity: same as the sequential simulation, but we eliminated the global event list, which may reduce the time complexity.
- Message Complexity: for each process step in the wafer fab, we need one message from the factory federate and one message from the AMHS federate. These are the essential messages for distributed simulation, and significantly reduced from our previous distributed simulation design.

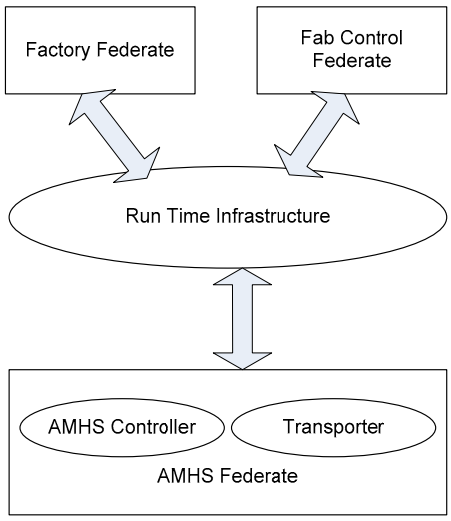

Figure 4. The Factory/AMHS Design.

Figure 4 illustrates the Factory/AMHS federation, with three federates, where the AMHS federate consists of the transportation systems and the AMHS controller.

3.2 The Hierarchical Distributed Simulation Model for 300mm Wafer Fab

Between two consecutive process steps in the factory federate, there are a set of intrabay and/or interbay transportation activities. For each process step in the factory, we on-

ly need to process two events, the start processing event and end processing event, while in the AMHS federate, up to hundreds of events are possible to move the lot in the transportation system from the machine to the next machine, possibly moving the lot between stockers or storing it temporarily if the machine for next process step is not available.

Within each LP, all the associated events are in the FEL. As the number of events in the AMHS federate is one or two orders of magnitude larger than the number of events in the factory federate, the computational time for each computer processor to finish their respective computation tasks are quite different, which makes the AMHS federate the bottleneck of the simulation execution.

In order to further improve the distributed simulation performance, we propose the hierarchical distributed simulation in Figure 5:

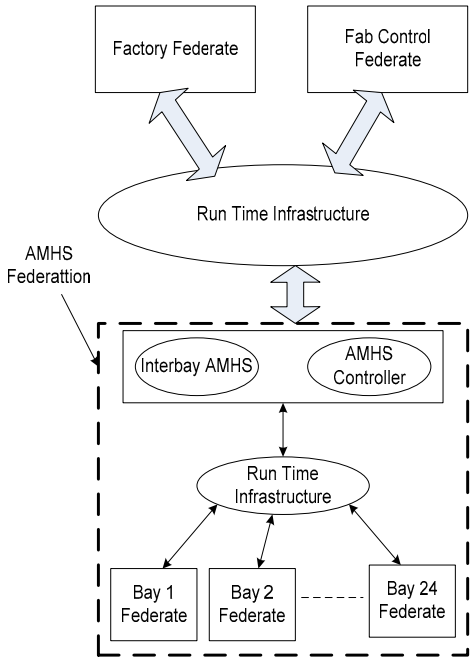

Figure 5. The Hierarchical Distributed Simulation Design.

The hierarchical distributed simulation design replace the AMHS federate in the AMHS federate in the Factory/AMHS design with a child federation, i.e. the AMHS federation, which distributes the AMHS subsystem in the 300mm Wafer Fab.

It has two levels of distributed simulation, a parent federation, and a child federation. The parent federation consists of three federates: factory, fab control, and AMHS. The AMHS subsystem is itself distributed across 25 federates: 24 bay federates, and an interbay federate (include interbay AMHS and the AMHS controller). We combine the interbay AMHS and the AMHS controller to avoid small lookahead federate in the child federation. The details are illustrated in Figure 6.

Conservative Synchronization PROTOCOL for Hierarchical Distributed Simulation
1. initialization
2. run all LPs

FEDERATE
3. while (~stop) {
4. while (TimeAdvanceGrant(T)) {}
5. e ← f(L)
6. while (e.ts < T) {
7. execute (e)
8. local_clock ← e.ts
9. e ← f(L)
10. }
11. NextEventRequest(e.ts)
12.}

CHILD FEDERATION
13. while (~stop) {
14. while (TimeAdvanceGrant(T)) {}
15. T' ← min(T, LBTS)
16. while (T' < T) {
17. broadcast T' to child federates
18. receive NextEventRequest(t) from child federates
19. LBTS calculation
20. T' ← min(T, LBTS)
21. }
22. NextEventRequest (LBTS)
23.}

CHILD FEDERATE
24. while (~stop) {
25. while (TimeAdvanceGrant(T)) {}
26. e ← f(L)
27. while (e.ts < T) {
28. execute (e)
29. local_clock ← e.ts
30. e ← f(L)
31. }
32. NextEventRequest(e.ts)
33.}

Figure 6: Algorithm of Conservative Synchronization Protocol for Hierarchical Distributed Simulation

The federates in Figure 6 proceed as normal federates in distributed simulation, the only difference between a federate in the parent federation and a federate in the child federation is that the child federate talks only to the Run Time Infrastructure (RTI) in the child federation, while the parent federate talks only to the RTI in the parent federation.

However, the LBTS calculation in the child federation has to take into consideration the LBTS value from the parent federation.

Suppose $LBTS_{parent}$ is the LBTS granted from the parent RTI, $LBTS_{child}$ is the LBTS for the child federation, F_{child} is the set of LPs in the child federation, t_i is the minimum FEL timestamp in federate $i \in F_{child}$, and L_i is the lookahead value in federate $i \in F_{child}$. We can calculate LBTS for the child federation by:

$$LBTS_{child} = \min(LBTS_{parent}, \min_{i \in F_{child}} (t_i + L_i))$$

To avoid violating the local causality constraint, we should ensure that $LBTS_{child} \le LBTS_{parent}$. We can continue to run the child federation as long as

$$LBTS_{parent} \ge \min_{i \in F_{child}} (t_i + L_i)$$

However, if $LBTS_{parent} < \min_{i \in F_{child}} (t_i + L_i)$, which means the child federation wants to proceed beyond the parent LBTS limit, it must call NextEventRequest with the parameter $\min_{i \in F_{child}} (t_i + L_i)$ just as a normal federate in the parent federation. After that, the child federation will wait for the parent federation to broadcast next $LBTS_{parent}$.

3.3 The Optimistic-Conservative Synchronization for the Hierarchical Distributed Simulation

In the previous section, we assume that the conservative synchronization protocol is used in the hierarchical distributed simulation design. However, the optimistic-conservative synchronization protocol also can be used:

- The optimistic-conservative protocol is applied in the parent federation.
- The child federation still uses the conservative synchronization protocol. This is due to the factor that the lookahead values in the child federation are all equal, and the optimistic-conservative protocol is not applicable for the child federation.
- The compensation value is applied to the factory federate in the parent federation, and the request ahead value is applied to the AMHS federation, which is a federation in the parent federation.

4 SUMMARY

In this paper, we introduced the hierarchical distributed simulation approach for the 300mm wafer fab. It is the extension of the Factory/AMHS distributed simulation approach applied successfully in HiFiVE. While the hierarchical distributed simulation is promising in achieving significantly speedup via massive distributed simulation of the 300mm wafer fab, extensive computational experiments (in progress at the time of this writing) are required to test actual speedup resulting from this approach.

REFERENCES

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*, John Wiley & Sons.

McGinnis, L. F., S. Xu and K. Wang. 2005. Evaluate Simulation Design Alternatives for Large Scale Manufacturing Systems. In Proceeding of the IEEE International Symposium on Semiconductor Manufacturing. Page(s): 344 – 347.

McGinnis, L. F. K. Wang, and S. Xu, 2006. Distributed Simulation of Manufacturing Systems, In Proceedings of the 16th Flexible Automation and Intelligent Manufacturing Conference. 26-28 June, 2006 in the University of Limerick, Ireland.

Parhami, B. 2002. Introduction to Parallel Processing Algorithms and Architectures, Kluwer Academic Publishers.

Wang K., S. Xu, L. F. McGinnis. 2005. Time management in distributed factory simulation, a case study using HLA. In Proceeding of Proceedings of the Winter Simulation Conference. Page(s): 1781-1786.

Xu, S., L. F., McGinnis. 2006. Optimistic-conservative synchronization in distributed factory simulation. In Proceeding of Proceedings of the 37th conference on Winter simulation Conference. Page(s): 1069-1074.

AUTHOR BIOGRAPHIES

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 an OR analyst at Sabre Airline Solutions, 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>.

LEON MCGINNIS is Gwaltney Professor of Manufacturing Systems at Georgia Tech, where he also serves as Director of the Product and Systems Lifecycle Management Center, Associate Director of the Manufacturing Research Center, and Director of the Keck Virtual Factory Lab. His research is focused on the representation of complex industrial systems, such as warehouses and factories, to enable analytic and simulation modeling to support performance assessment, behavioral prediction, and system design. His email address is <leon.mcginnis@gatech.edu>.