Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

OPTIMISTIC-CONSERVATIVE SYNCHRONIZATION IN DISTRIBUTED FACTORY SIMULATION

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 complicated manufacturing systems having many tools and products, such as a semiconductor wafer fabrication line. However, conservative synchronization approaches can introduce excessive overhead in execution, and result in little parallelism, which can eliminate the speedup promised by distributed simulation. Our experiences in building a distributed simulation model for 300mm wafer fab using the High Level Architecture (HLA) shows that using model specific information in a novel adaptation of conservative synchronization can achieve very significant reduction in model execution time. This paper defines the time-chop problem for which this adaptation is effective, and formally develops our optimistic-conservative synchronization scheme.

1 INTRODUCTION

HiFiVE (High-Fidelity Virtual Environment for 300mm Wafer Fabrication), is a web-based, distributed simulator designed to: (i) promote a structured approach for fab design, analysis, and control, (ii) support fab rapid prototyping through discrete event system modeling and verification, (iii) provide a large-scale and high-speed simulation for better engineering decision support, (iv) enable plug & play controller through exploiting modularity among the model, view, and control, and (v) provide a research and development testbed (see Kim et al. 2001, McGinnis et al. 2005, Wang et al. 2005, and McGinnis et al. 2006).

Our aim has been to provide a high-fidelity modeling framework in order to simulate the fab behavior as realistically as possible. Specifically, HiFiVE allows us to model (i) equipment-level behavioral aspects such as batching, setups, rework, processing errors, preventive maintenances, and tool failures, and (ii) system-level aspects such as effects resulting from the behavior of MHS (Material Handling System) and supporting staff, and the complex resource allocation dynamics due to the routing flexibility and multiple resource sharing.

Configured as a distributed simulation, HiFiVE 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 (McGinnis et al. 2005). This paper addresses a fundamental issue we encountered in using conservative synchronization of federated simulations (Fujimoto 2000) of a specific semiconductor wafer fab model (Sematech 2005).

2 THE TIME-CHOP PROBLEM IN THE CONSERVATIVE SYNCHRONIZATION SCHEME

During the process of improving the execution time of the HiFiVE federation, we observed that the material handling federate has many events in a small logical time interval, and as a result the next event request (NER) service in HLA can only request to advance a very small amount of time. Consequently, the simulated time for the material handling federate is cut into very small pieces and little parallelization can be achieved. Because this problem is likely to occur in any high fidelity factory model incorporating a detailed model of automated material handling, we call this phenomenon the Time-Chop Problem in the Conservative Synchronization Scheme.

In order to illustrate the problem, we first introduce some terminology and notation, following Fujimoto (2000).

Definition 1 *Lookahead: If a federate at simulation time T can only schedule new events with time stamp of at least $T + L$, then L is referred to as the lookahead for the federate.*

Definition 2 *Next Event Request/Time Advance Request(NER/TAR): the Next Event Request service is used by a federate to request the next smallest time-stamped event from the RTI. The Time Advance Request service is used by a federate to request to advance to a specific future simulation time. Both services will include the future time* t*, and both imply a guarantee that no messages with time stamp less than* t *will be sent by the federate in the future unless it receives smaller time stamp messages from the RTI.*

Definition 3 *Lower Bound on Time Stamp (LBTS): LBTS, at a given point in time, is the lower bound on the time stamp of any message any federate can receive in the future in the traditional conservative simulation.*

Definition 4 *Time Advance Grant (TAG): the Time Advance Grant service issues the LBTS to federates, after sending messages to them, and guarantees that it is safe for the federate to process any message with time stamp less than the current LBTS value.*

2.1 An illustrative example of the Time-Chop Problem

We provide the following example to illustrate the Time-Chop Problem in the conventional Conservative Synchronization Scheme.

There are two federates exchanging time stamp messages. The minimum response time for federate 1 (LP1) is 8 minutes, i.e. it will send out time stamp message at least 8 minutes after it receives a message from the other federate or processes a local event in its future event list.

The minimum response time for federate 2 (LP2) is 1 minute, i.e. it will send out time stamp message at least 1 minute after it receives a message from the other federate or processes a local event in its future event list.

We further assume that LP2 is the computational bottleneck for the simulation federation as it has on average 100 local events to process between its receiving of the time stamp message from LP1 and sending out the time stamp message to LP1. LP1 does not have any local event: upon receiving the time stamp message from LP2, it will schedule the outgoing message to LP2 immediately. Therefore, suppose LP2 receives a message with time stamp t; after processing it, LP2 will schedule a local event with time stamp $t+1/100$; after processing the local event, LP2 will schedule another local event with time stamp $t+2/100$; this process is repeated on average 100 times, until LP2 reaches time $t+1$, at which point it will send out a message with time stamp $t+1+L$ to LP1.

We have the following additional assumptions:

- Initially the simulation time for both federates is at 0.
- LP1 has three pending events in its future event list with time stamps 1, 7, 10 respectively.
- LP2 has 5 pending events in its future event list, with time stamps 0, 2, 4, 6, 8.

Based on these assumptions, the simulation system as described is displayed in Figure 1, where we name the pending events for the two federates E1, E2, E3, and E4, E5, E6, E7, E8, respectively.

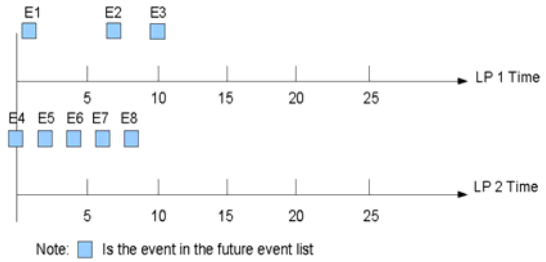

Figure 1: A Simulation System with Two Federates

We first consider the Conservative Synchronization Protocol. We set the lookahead value to 8 minutes for LP1, and 1 minute for LP2. The future event list in the simulation system is described in Figure 1.

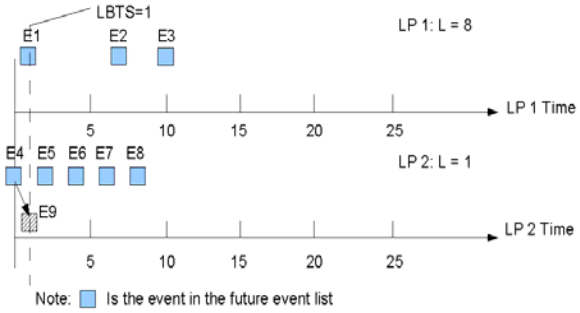

Figure 2: Simulation System with LBTS = 1

The time stamp of the next event in the future event list is 1 for LP1, and 0 for LP2. LP1 will send Next Event Request with the parameter 1+8=9, and LP2 will send Next Event Request with the parameter 0+1=1. The simulation executive will issue LBTS as 1 and send to both federates. See Figure 2. Then the event E4 can be processed. After scheduling and processing about 100 local events, it eventually schedules event E9 at time 1. This is illustrated in Figure 2.

The LBTS computation will be repeated and the new LBTS is 2, and both federates are granted time advance. Events E1 and E9 are now eligible to be processed. LP1 and LP2 send out time stamp messages E11 and E10 with time stamp 9 and 2 respectively, as described in Figure 3.

As each federate completes processing events up to the granted time advance, i.e. t=2, a new round of LBTS computation begins. LP1 will send Next Event Request with

the parameter 2+8=10, and LP2 will send Next Event Request with the parameter 2+1=3. Therefore, LBTS will be set to 3, and this process will be repeated again and again, while the increment of the LBTS is always 1.

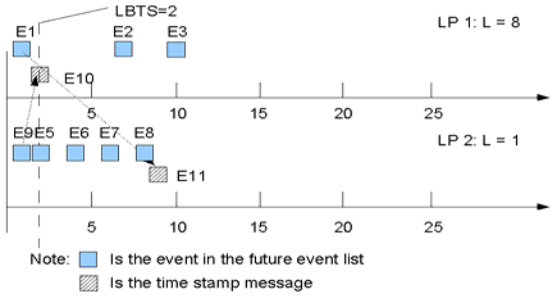

Figure 3: Simulation System with LBTS = 2

3 OPTIMISTIC-CONSERVATIVE SYNCHRONIZATION SCHEME

The Optimistic-Conservative Synchronization is an innovative synchronization first proposed in the Virtual Factory Lab at Georgia Tech to ameliorate the time chop problem for our application. It is an extension of the usual conservative synchronization that attempts to achieve some of the benefits of optimistic synchronization while eliminating the possibility of a computationally expensive rollback. In our optimistic-conservative synchronization, whenever the material handling federate finishes processing all local events at current logical time and is ready to advance to a new logical time, it calls NER(t+s), where t is the time stamp of its next local event, and s is a fixed value we choose based on an analysis of the federate's dynamic behavior. With a carefully chosen s, we can significantly improve the parallelism between federates.

The basic idea is similar to optimistic synchronization, in that an event from the material handling federate may appear at the factory federate "in its past," but because of the way we have chosen the parameter, s, we can avoid the burden of rollback. Instead, a compensation scheme is proposed, which will be used to achieve correct simulation results, without roll-back.

In order to formally define the Optimistic-Conservative Synchronization Protocol, we first introduce following set of definitions:

Definition 5 *Roll-back free compensation value: If a federate 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* $T - T' \le S$.

To illustrate the rollback free compensation value, consider the case with one automatic machine and a vehicle. Suppose the vehicle arrives and transfers a job to the machine, and a message with time stamp 10 is sent by the vehicle federate to the machine federate informing it of the drop-off. Suppose the machine federate receives the message at its local time 12. The message is received in its past. However, rollback may not be necessary if, e.g., the minimum processing time for the machine is 10 time units and its corresponding lookahead value is also 10. If so, then we can schedule the new event from the job arrival in the machine federate with time stamp of at least 20, by reducing the processing time by 12-10=2 time units, and no rollback will be 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 lookahead for the machine federate, which is not recommended because zero effective-lookahead will result. The general approach is to find the balance between the compensation value and the lookahead value.

Definition 6 *Maximum allowable advance request: If a federate is at simulation time T, and the lookhead value is L, then it can send Next Event Request/Time Advance Request with time* $T+S_{max}$ *where* S_{max} *is chosen so that event messages sent from the federate still can be compensated without rollback.*

When the federate requests to advance to $T+S_{max}$, it is possible for some message snet to other federates to be received in their past. We define S_{max} so that the late-arriving messages still can be compensated. For a federate i, let E_i be the set of federates receiving messages from LP i, then:

$$S_{\max,i} = \min_{j \in E_i}(S_j),$$

where S_j is the roll-back free compensation value for federate j.

Definition 7 *Lower Bound on Time Stamp with Compensation (LBTSwC): LBTSwC-* S_i *is the lower bound on the time stamp of any message federate i can receive in the future in the traditional conservative simulation. Different from LBTS, it is still possible to receive messages with time stamp less than LBTSwC, but the difference cannot be larger than* S_i.

The Optimistic-Conservative Synchronization Scheme can be applied when we can compensate, if we receive late messages within a specified threshold range. For a specific application, we can apply this scheme if we can find a non-zero threshold value for late messages. If the maximum possible threshold value is 0, then our approach will be identical to traditional Conservative Synchronization Scheme. The Optimistic-Conservative Synchronization Scheme does not involve a trade-off between accuracy and

performance, because it will achieve the same simulation results as would strictly conservative synchronization.

3.1 LBTS computation with compensation

With the Optimistic-Conservative Synchronization Protocol, the meaning of LBTS is changed to LBTS with compensation, or LBTSwC, which is the lower bound on the time stamp plus Roll-back free compensation value of any message a federate can receive in the future. However, it is still safe to process all events with time stamp less than LBTSwC, because of compensation.

The procedure to calculate LBTSwC in the Optimistic-Conservative Synchronization Protocol is shown in Figure 4 (assuming no transient messages).

The Federate
1. Given a LBTSwC value.
2. Process all events with time stamp less than LBTSwC.
3. Sends Next Event Request/Time Advance Request with the parameter $T+S_{max}$.
4. while (TimeAdvanceGrant(LBTSwC)) {}
5. Goto step 1.

The Simulation Executive
1.After all federates' requests are received, and suppose the received parameter value from federate i is t_i.
2. $\mathrm{LBTSwC} = \min(t_i + L_i)$,
where t_i is $T+S_{max}$ from federate i, and L_i is the lookahead value.
3. Send messages with time stamp less than LBTSwC to the respective federates.
4. Broadcast the LBTSwC through Time Advance Grant to all federates.

Figure 4: Algorithm to Calculate LBTSwC

3.2 Implementation of the Optimistic-Conservative Synchronization Protocol

The Optimistic-Conservative Synchronization Protocol can be implemented using the traditional HLA simulation executive. During the initialization step, we need additional steps as follows.

The Roll-back free compensation value S for each federate should first be defined. Based on the publish-subscription relationship between different federates, the Maximum allowable advance request value S_{max} can be calculated according to its definition. (This procedure can be computerized, but in our current implementation, we manually calculate it and define it in the simulation applications).

Then, each federate will request Next Event Request/Time Advance Request with parameter $T+S_{max}$, and the HLA simulation executive will start LBTS/LBTSwC computation using its traditional LBTS computation approach, while the LBTSwC will be calculated instead.

The last issue is to send out messages in the past between federates through the HLA simulation executive. As the HLA simulation executive does not allow any message with time stamp less than LBTS/LBTSwC to be sent, we introduce a patch in the simulation application to include two time stamps for each message: a true time stamp which is possible to happen in the past of some federates, and an HLA time stamp, which will follow the requirement of the HLA simulation executive. The HLA simulation executive will only see the HLA time stamp, while the federates will use only the true time stamp. No change to the RTI is required to compute LBTSwC, so that this synchronization approach can be easily implemented using existing HLA simulation executives.

The detailed procedure for the Optimistic-Conservative Synchronization Protocol is in Figure 5.

```
Optimistic-Conservative Synchronization
PROTOCOL
1. initialization
2. run all LPs

FEDERATE
3. while (~stop) {
4.   while (TimeAdvanceGrant(T)) {}
5.   e ← f(L)
6.   while (e.ts < T) {
7.        if (e.ts < local_clock){
8           if (local_clock – e.ts <= s){
9.            compensate with s in
              future event scheduling
10.         }
11.         else
12.         {
13.           Simulation stop
14.         }
15.       }
16.       execute (e)
17.       local_clock ←  e.ts
18.       e ← f(L)
19.  }
20.  NextEventRequest(e.ts+s')
21. }
```

Figure 5: Algorithm of Optimistic-Conservative Synchronization Protocol

4 ILLUSTRATION OF THE OPTIMISTIC-CONSERVATIVE SYNCHRONIZATION PROTOCOL

We use the example in Section 2 to illustrate that the Optimistic-Conservative Synchronization Protocol can, in fact, ameliorate the Time-Chop problem.

The minimum response time for LP1 is 8 minutes, i.e. it will send out time stamp message at least 8 minutes after it receives a message from the other federate or process a local event in this future event list. We set the lookahead value L to 4 minutes, and the Roll-back free compensation value S to 4 minutes, as $L + S \le 8$, the minimum response time.

The minimum response time for LP2 is 1 minutes, i.e. it will send out time stamp message at least 1 minutes after it receives a message from the other federate or process a local event in this future event list. We set the lookahead value L to 1 minutes, and the Roll-back free compensation value S to 0 minutes, as $L + S \le 1$, the minimum response time.

As LP1 and LP2 subscribe to each other, we can calculate their respective maximum allowable advance request values S_{max}. We have S_{max}=0 for LP1, and S_{max} =4 for LP2. Similarly LP2 is computational bottleneck for the simulation system as it has on average 100 local events to process between its receiving of the time stamp message from LP1 and sending out the time stamp message to LP1. LP1 does not have any local event: upon receiving the time stamp message from LP2, it will schedule the outgoing message to LP1 straightforwardly.

We now consider the Optimistic-Conservative Synchronization Protocol. The future event list in the simulation system is described in Figure 6: the time stamp of the three future events in LP1 are 1, 7, 10 and the time stamp of the future events in LP2 are 0, 2, 4, 6, 8 as before. In order to facilitate the following presentation, we also name the events as E1, E2, E3 and E4, E5, E6, E7, E8 respectively.

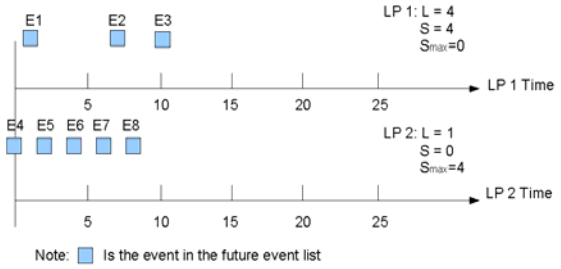

Figure 6: Future Event List using Optimistic-Conservative Synchronization Protocol

The time stamp of the next event in the future event list is 1 for LP1, and 0 for LP2. LP1 will send Next Event Request with the parameter 1+4+0=5, and LP2 will send Next Event Request with the parameter 0+1+4=5. The simulation executive will issue LBTSwC as 5 and send back to both federates. Then events E4, E5, E6 in LP2 can be processed. After scheduling and processing about 100 local events, LP2 eventually schedules events E9, E10, E11 at time 1, 3, 5 respectively. E9, E10 are also eligible to proceed, and send out messages E12, E13 to LP1 with time stamp 2 and 4 respectively. The event $E1$ can be processed by LP1, and it sends out message $E14$ with time stamp 9. These actions are shown in Figure 7.

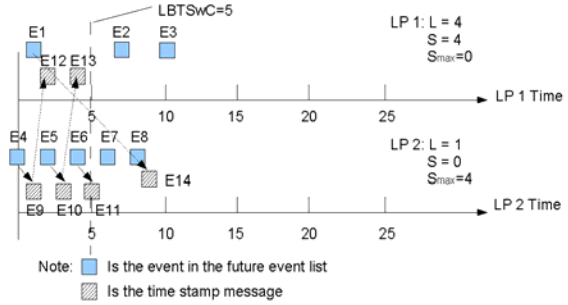

Figure 7: Simulation System with LBTSwC = 5

At this point, the time stamp of the next event in the future event list is 7 for LP1, and 5 for LP2. LP1 will send Next Event Request with the parameter 7+4+0=11, and LP2 will send Next Event Request with the parameter 5+1+4=10. The simulation executive will determine LBTSwC as 10 and send back to both federates. Then the event E12, E13, E2 in LP 1 and E11, E7, E8, E14 in LP2 can be processed as illustrated in Figure 8.

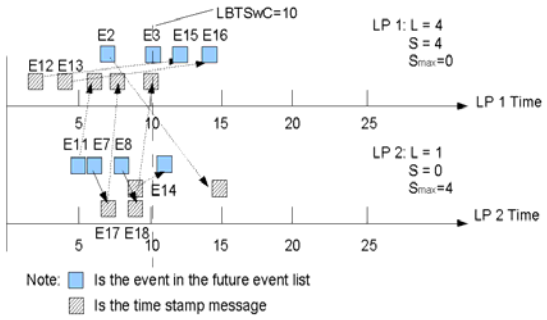

Figure 8: Simulation System with LBTSwC = 10

5 SUMMARY

We have described a common problem in high-tech factory distributed simulation, namely, the Time Chop Problem, which, when present, obviates one of the prime motivations for distributed simulation, i.e., parallelization of the computations. We have presented and illustrated a novel approach to distributed simulation synchronization which ameliorates the Time Chop Problem, which we term Optimistic-Conservative Synchronization.

Optimistic-Conservative Synchronization can be implemented with existing synchronization tools, such as HLA, although it does require some modification to the federates themselves.

For our target application, the optimistic-conservative synchronization resulted in significant improvement in performance, as illustrated in Figure 9. The speed factor is defined as log (Current Simulation Time / Cumulated CPU

Clock Time). Optimistic-Conservative synchronization achieves a dramatic improvement over conservative synchronization and Automod/ASAP using the same computational resources. One explanation for this improvement is that by allowing more events to be processed by the material handling federate in one "time slice," there are fewer switches between the federates, and thus less context switching and associated computational overhead.

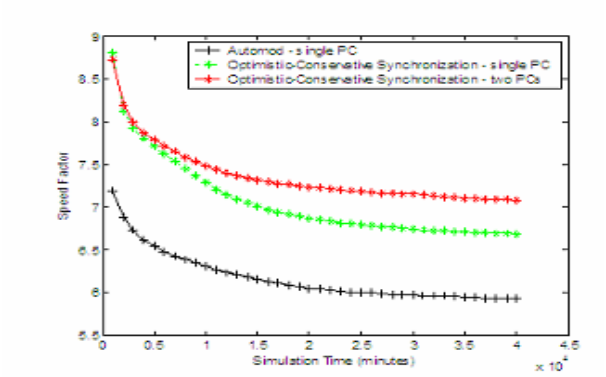

Figure 9: Conservative-Optimistic Synchronization, Conservative Synchronization, and Automod Comparison

Future research will continue to generalize the approach, and demonstrate its applicability to more than two federates.

REFERENCES

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., 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 .

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

McGinnis, L. F., K. Wang, and S. Xu. 2005. Distributed simulation modeling of large scale manufacturing systems. In *IIE 2005 Annual Conference*.

Sematech. *Fab Simulation Models*. 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. Available via <http://www.semiconductor-technology.com/projects/elpida>. [accessed April 2, 2005].

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

K. Wang, S. Xu, and L. F. McGinnis, 2005. Time Management in Distributed Factory Simulation, a Case Study Using HLA. *Proceedings of the 2005 Winter Simulation Conference*. pp. 1781-1786. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

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. His e-mail is <leon.mcginnis@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. His e-mail is <sxu@isye.gatech.edu>.