한국시뮬레이션학회 논문지
Vol. 19, No. 3, pp. 45-54 (2010. 9)

시뮬레이션 기반 적응형 실시간 작업 제어 프레임워크를 적용한 웨이퍼 제조 공정 DEVS 기반 모델링 시뮬레이션

송해상[1†] · 이재영[2] · 김탁곤[3]

DEVS-based Modeling Simulation for Semiconductor Manufacturing Using an Simulation-based Adaptive Real-time Job Control Framework

Hae Sang Song · Jae Young Lee · Tag Gon Kim

ABSTRACT

The inherent complexity of semiconductor fabrication processes makes it hard to solve well-known job scheduling problems in analytical ways, which leads us to rely practically on discrete event modeling simulations to learn the effects of changing the system's parameters. Meanwhile, unpredictable disturbances such as machine failures and maintenance diminish the productivity of semiconductor manufacturing processes with fixed scheduling policies; thus, it is necessary to adapt job scheduling policy in a timely manner in reaction to critical environmental changes (disturbances) in order for the fabrication process to perform optimally. This paper proposes an adaptive job control framework for a wafer fabrication process in a control system theoretical approach and implements it based on a DEVS modeling simulation environment. The proposed framework has the advantages in view of the whole systems understanding and flexibility of applying new rules compared to most ad-hoc software approaches in this field. Furthermore, it is flexible enough to incorporate new job scheduling rules into the existing rule set. Experimental results show that this control framework with adaptive rescheduling outperforms fixed job scheduling algorithms.

Key words : DEVS modeling simulation, Real-time Job Scheduling, Semiconductor Fabrication Process

요 약

반도체 제조공정에 내재된 복잡성은 작업일정(job scheduling) 문제를 해석적 방법으로는 풀기 어렵기 때문에 보통 시스템 파라미터의 변화에 대한 효과를 이산사건 모델링 시뮬레이션에 의존하여 왔다. 한편 장비 고장 등 예측 불가능한 사건들은 고정된 작업일정 기법을 사용할 경우 전체 공정의 효율을 악화시킨다. 따라서 이러한 불확실성에 대해 최적의 성능을 내기 위해서는 작업일정을 실시간으로 대처 변경하는 것이 필요하다. 본 논문은 반도체 제조 공정에 대해 시스템 제어관점의 접근방법을 적용하여 이 문제에 적응형 실시간 작업제어 틀을 제안하고, DEVS 모델링 시뮬레이션 환경을 기반으로 제안된 틀을 설계 구현하였다. 제안된 방법은 기존의 임기응변적인 소프트웨어적인 방법에 비추어볼 때 전체 시스템을 이해하기 쉬우면서도 또한 추가되는 작업제어 규칙도 쉽게 추가 적용할 수 있는 유연성을 장점으로 가지고 있다. 여러 가지 실험결과 제안된 적응형 실시간 작업제어 프레임워크는 고정 작업규칙 방법에 비해 훨씬 나은 결과를 보여주어 그 효용성을 입증하였다.

주요어 : DEVS 모델링 시뮬레이션, 실시간 작업 스케쥴링, 반도체 제조 공정 제어

2010년 6월 7일 접수, 2010년 9월 7일 채택
[1)] 서원대학교 컴퓨터공학과
[2)] (주)하이닉스반도체
[3)] 한국과학기술원 전기및전자공학과
주 저 자 : 송해상
교신저자 : 송해상
E-mail; hssong@seowon.ac.kr

1. INTRODUCTION

The job scheduling problem of semiconductor fabrication processes has attracted great attention in recent decades, since the impact is huge with little increase in cost to an expensive facility. Due to the inherent high complexity of the process, most of the literature has exploited discrete event modeling simulation techniques. Fowler[6] lists the following complex characteristics of semiconductor manufacturing:

· reentrance flows due to expensive limited machinery,
· large number of products with mixed product types,
· unexpected internal and external disturbances, such as machine breakdown,
· mixed processing type of batch or a single lot job,
· sequence dependent setup times,
· a group of same type of machinery deployed,
· diverse process flows even on a single product.

Even in manufacturing a single type of product, there are many factors, such as dispatching rules, limited machinery, process time, input rate, product mix, and disturbances, which make the factory perform very differently. Thus it seems that operational planning and scheduling can create great opportunities to reduce cost by increasing performance in throughput, WIP (wafers in process) and tardiness. Due to the complexity of semiconductor manufacturing, discrete event modeling simulation has attracted more attention for real-time scheduling of semiconductor fabrication systems, as it can show the effect of system parameter changes on performance variation[2].

Simulation is used not only at the design stage of the facility, but also during running. It plays an essential part in determining the capacity of machinery, line layout, and dispatching policy by predicting the performance of the design alternatives. It is, however, still needed even after facility construction for two reasons: unexpected disturbances such as machine breakdown can distort the expected performance and necessitate prompt rescheduling, and new product orders introduced into a running facility would disorder the schedule. Thus it seems that the simulator needs to keep current facility status up-to-date all the time; therefore, if any serious factor change in the factory is detected, the simulator should be used for real-time rescheduling to maintain set production objectives, as proposed in the next section.

Most of the literature in this field is based on operational research[13] or software engineering approaches [9,10]. However, the job scheduling scheme proposed in the paper is based on system control theory, which separates the controller from the system to be controlled, called the plant; the approach has an explicit interface between components to be modular.

In this paper we propose and also implement a system control theoretic framework for the job scheduling problem of complex semiconductor fabrication, DEVS-based discrete event system modeling simulation. This work makes it clear that any job scheduling problem is merely abstracted to a controller design problem, where the plant is the fabrication facility and the controller is an entity which controls the plant to meet given control objectives from the user cost function, such as WIP, throughput, tardiness, and orders met within due date. The simulation experiment shows that this framework has sufficient potentiality and flexibility to be applicable in the real world. The contribution of this paper is not to compare the performance of specific scheduling algorithms, but to propose a general adaptive control theoretical framework for simulation-based real-time job scheduling and show the feasibility of realizing the framework in a DEVS M&S environment.

This paper is organized as follows. The next section contains a review of related literature and our proposed control framework for job scheduling. The SEMATECH data set we use to model semiconductor fabrication processes and dispatching rules we use for experiments are also mentioned in the section. The next section briefly describes the design and implementation of the framework. Section 4 presents experiment results and remarks. In the last section, we summarize and conclude our work.

2. PROPOSED FRAMEWORK

2.1. Related Work

Most simulation-based real-time job scheduling uses simulation for exploring alternatives when decisions need to be made. There are two types of real-time scheduling approaches in terms of decision point: continuous and exception-based. Continuous usually monitors the real system and makes use of simulation for continuous decision making. Krishnamurthi[3] proposes an online simulation framework as a general decision support system; the framework simulator always monitors the status of the real system to reflect that information in the simulator itself. Smith[4] uses simulation as a continuous decision maker for analysis and a generator of tasks to be done in the near future. Meanwhile, exception-based scheduling makes decisions only when exceptional events such as machine breakdown or repair occur. Manivannan[5] proposes such a mechanism which modifies the temporal events calendar when real system events occur. Kim[8,9] uses the notion of a soft exception; furthermore, he proposes a simulation-based real-time scheduling mechanism which changes the current dispatching rule to the best one obtained by decision making using simulation when a soft exception occurs. Whenever the controller detects an abnormal condition, it commands the simulator to evaluate alternative dispatching rules in light of the current system state. The major issues in this approach are the length of look-ahead window, how to initialize simulator state in accordance to the real system, and how fast the simulation runs for real-time decisions. Although the longer the look-ahead window is, the better the decisions are made. It requires more computing power and thus gives longer response time, which is not good for real-time decisions. Although all these systems yield fairly good suggestions, their frameworks are all based on a software engineer's approach, not on a control system theoretical view which could be applied to general systems. To the authors' knowledge, little work has been done similar to our approach.

2.2. Proposed Adaptive Job Control Framework

This subsection describes the proposed simulation-based control framework in detail as shown in Figure 1. The framework consists of three entities: controller model *Mc,* virtual plant model *Mv,* and evaluator model *Me* as a decision support expert. Instead of a real plant, we use virtual plant simulator *Mv* in two reasons; first, the real plant might not have been built yet; second even if it has been built already, a virtual plant model is required to rebuild the evaluation model as quickly as possible for fast evaluation. However we consider that the virtual plant behaves in the same way as the real one, assuming that the status of the real plant is directly reflected in the virtual plant through fab control software.

The controller takes control objectives from the user as input and output *Yv* of virtual plant simulator *Mv.* Observing the output of the virtual plant *Yv* and comparing it with control objective *Pt,* the controller determines a decision point (or time) when *Yv* does not match *Pt.* This is mostly owing to disturbances. Then it commands the virtual plant to verbatim-copy the state of the plant *Mv* at that time of decision point, *Sv'* this is possible because the virtual plant model itself is a software implementation. Then the controller spawns a new simulation-based evaluation model *Mv'* which is exactly the same model with *Mv* but constructed from the current state *Sv'* captured at the time of decision point. With evaluation condition *Re* of dispatching rules and evaluation period, the evaluation model runs a simulation to produce performance output *Ye,* which is feed-backed to the controller. Multiple evaluation runs could be done in parallel with different evaluation conditions to speed up the total evaluation time using parallel or multi-threaded computation. Summing up the results, the controller chooses the best-fit dispatching rule along with the control objectives and applies it to

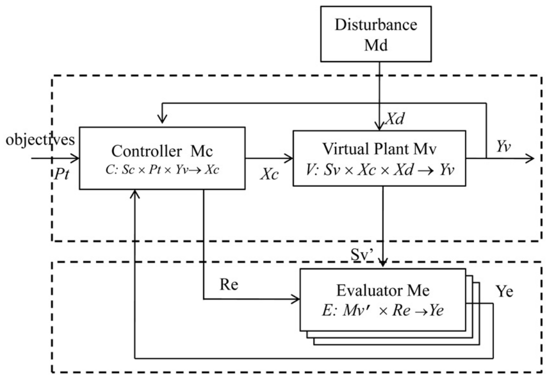

Fig. 1. Simulation-based Adaptive Job Control Framework

Table 1. System variables

Item	Variable Name	Description
Pt	Control objectives	Cost function such as throughput, tardiness, WIP
Xc	Control Variables	Dispatching rules, input ratio
Yv	Output Variables	Throughput, WIP, tardiness
Sv	State Variables	Queue length, processing time
Xd	Disturbances	Machine breakdown/up
Re	Conditions	States, controls, time to evaluate
Ye	Evaluation results	Outputs from evaluation

the plants by issuing command *Xc*. Disturbance *Xd* affects the plant performance, yielding change of performance *Yv*. Table 1 summarizes the system variables in Figure 1.

In view of implementation, to make this framework possible, the modeling simulation environment should meet the following requirements:

- capture the state of the virtual plant model
- transfer the captured state to evaluator
- fast construction of the evaluation model from the state of the virtual plant model
- fast evaluation speed
- real-time and logical time simulation support

2.3. DEVS M&S Environment

To satisfy the above requirements, we choose the DEVS environment as the best one to represent and implement the proposed control system framework, since DEVS formalism has a sound semantics for representing modular and hierarchical system structure and behavior[11]. The DEVS formalism is composed of two types of model: coupled models represent the structure of a system, and an atomic model models the behavior of a basic component which cannot be decomposed further. It also supports the notion of separation of the model from the simulation engine, called an abstract simulator. Moreover there are versions of proven simulation engines for DEVS models, one of which is DEVSim++[12]. It supports real-time and/or logical time DEVS simulation and has convenient auxiliary tools such as simulation controls, logging, networking, external messaging interface, flattening, multi-threading, and so on. Recently, concepts of co-modeling architecture have been introduced to many practical simulation developments, such as war-games, which separate discrete event models with corresponding software objects that can help simulation experts and software experts work together[14]. This paper won't repeat DEVS formalism, modeling notations, and other detailed descriptions due to lack of space. Note, however, that we utilize all the benefits of the DEVS/DEVSim++ environment mentioned above.

2.4. Data Set for Virtual Plant Modeling

For scalable and standardized wafer fabrication process (fab for short) modeling simulation we utilize SEMATECH data set no. 5 provided by Fowler[6] as de-facto benchmark data, which has the following characteristics:

- Each lot has 25~50 wafers
- There are 21 different product types with different input rates.
- Each workstation has more than one machine.
- Total number of machines in the line is 175.
- Each product type has 121~266 processing steps.
- Each machine has a queue for waiting lots.
- The line has 85 workstations, 20 of which are lot processing type and the remaining 65 are batch type.
- Lot arrival rate varies according to its type.

A lot of the other systems use toy mini-fab models in the data set, whereas we use a more complicated data set for practical use.

2.5. Dispatching Rules for Scheduling

Generally job scheduling in semiconductor fabrication lines is conducted via changing dispatching rules[7] at each station - priority rules to select lots waiting in a queue to be processed by the workstation. Each lot (a group of wafers) has a different processing route according to product type, processing time up to now, processing time left hereafter, processing time in the current step, and due date for packaging. A workstation can make a decision as to which lot in the queue should be processed first based on this information.

Table 2. Dispatching Rule Set for Experiments

No.	Rule	Description
1	EFIT	Earliest Fab In Time (earliest start time)
2	SOPT	Shortest Processing Time in this step
3	SJPT	Shortest Job Processing Time in total
4	SRW	Shortest Remaining Work (time left)
5	SRPDJT	Smallest Ratio of Processing Time Divided by Total Processing Time
6	LRPDRW	Largest Ratio of Processing Time Divided by Remaining Work
7	SPMJT	Smallest Processing Time Multiplied by Job Processing Time
8	SPMRW	Smallest Processing Time Multiplied by Remaining Work
9	SLACK	Smallest Margin for Due Date (time left up to due date minus processing time left)
10	LSPRO	Least Slack per Remaining Operation (SLACK divided by number of steps left)
11	LSPRW	Least Slack per Remaining Work (SLACK divided by processing time left)
12	EDD	Earliest Due Date

From among a variety of combinations, we selected 12 dispatching rules for our experiments as shown in the following table, though more are possible.

3. DESIGN AND IMPLEMENTATION

3.1. Controller Model

Figure 2 depicts the controller. Instead of mathematical representations of models, we use a DEVS diagram similar to a state diagram to help readers understand the models. A circle represents a state, and a solid arrow from one circle to another other depicts an external transition, while a dotted arrow represents an internal transition.

The atomic controller model has three ports *Pt, Ye,* and *Yv*, and two output ports *Xc* and *Yc*. Each port carries structured messages to contain sufficient information. As shown in Figure 2, at the initial state of the controller, 'monitor and control,' it waits for any input event. When it receives a periodic performance message

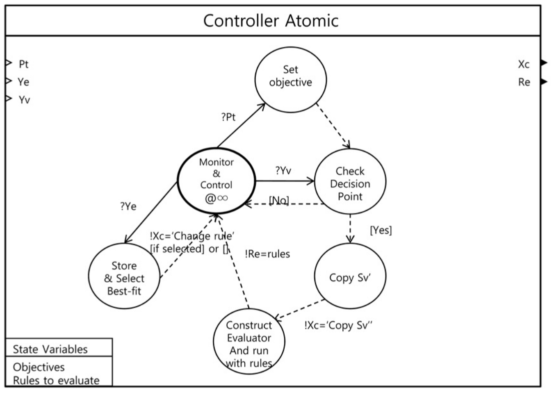

Fig. 2. Atomic controller model

Yv from the virtual plant, it checks whether it is time for decision making. The threshold conditions for decision making are either that the performance *Yv* is significantly lower than the objective performance, or there is significant change in the system state, such as machine breakdown or repair. This information is assumed to be observable through structured messages from *Yv*. If decision making is required, it sends a 'copy state' command message through output port *Xc* and sends a rule message through port *Re* which is connected to the evaluator model, or it returns to the 'monitor and control' state and waits indefinitely. If the controller receives a new objective message through *Pt*, which could be a new production order or a new performance goal, then it updates the state variable 'Objectives' for decision making.

3.2. Simulation-based Evaluator

On the other hand, when a decision point is made by the controller, the evaluator constructs a fab simulator *Mv'* with a snapshot of current virtual fab system status *Sv'*. This means the simulation environment needs to have the capability to reconstruct a simulator from a given system state. Then the evaluator iterates each simulation for a given time window for each rule in the rule set. Figure 3 shows the overall evaluation and simulation control process in detail, as follows.

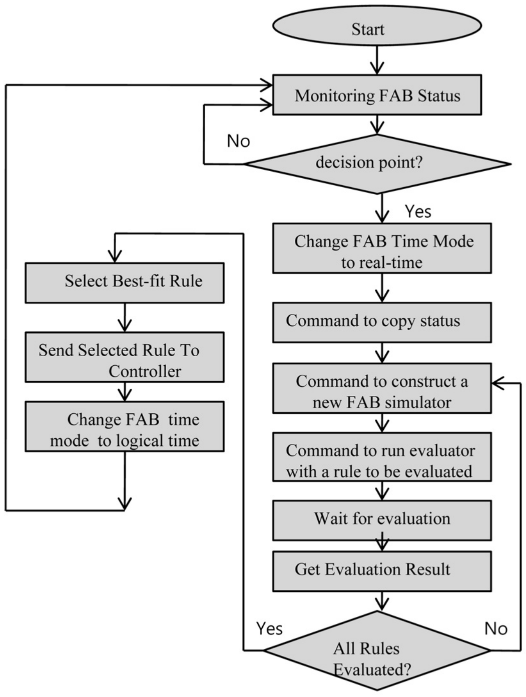
Start
Monitoring FAB Status
No
decision point?
Yes
Change FAB Time Mode to real-time
Command to copy status
Command to construct a new FAB simulator
Command to run evaluator with a rule to be evaluated
Wait for evaluation
Get Evaluation Result
All Rules Evaluated?
Yes
No
Select Best-fit Rule
Send Selected Rule To Controller
Change FAB time mode to logical time

Fig. 3. Flow chart of Evaluation Process

Initially it monitors the status of the virtual plant *Mv* until it detects an abnormal condition. On an abnormal condition, it commands the virtual fab simulator to copy the current system status, which is transferred to the evaluator. For each rule in the rule set, it initiates a new simulation with the chosen rule in a given time window. Repeating simulations with each rule in the rule set, the evaluator chooses a best-fit result with respect to the control objective, such as maximum throughput. Then it sends notification of the best-fit rule to the controller, which in turn commands the virtual plant to change the dispatching rule. Taking into account the real circumstance, we note that simulation mode is real-time when evaluation is in process; otherwise virtual time mode is used to speed up the whole experiments. To reduce total evaluation time, it is possible that the evaluator can launch multiple simulation threads simultaneously in different computers. As new rule policies can be added to the rule set, the evaluation

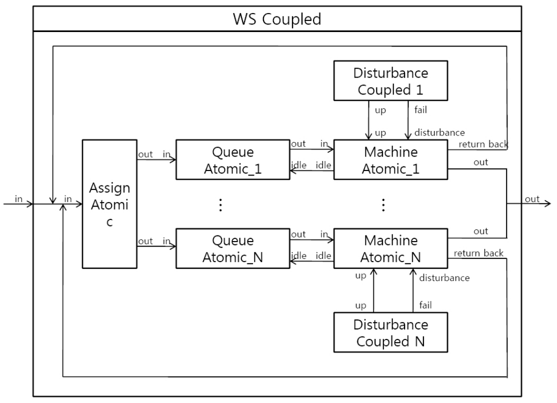

Fig. 4. Coupled Workstation Model

time could take longer. Note that the developer can determine which dispatching rules to include in the rule set from the results so as to expand the rule set.

3.3. Virtual Plant Model Construction

In compliance with SEMATECH data set specification[6], we create a virtual fab model in DEVS general enough to construct a virtual fab simulator from any SEMATECH data set. In this paper, we construct a virtual fab simulator from data set 5 automatically at run-time.

Figure 4 depicts a basic workstation coupled model, which is the basic element for constructing the whole fab simulator. The coupled workstation model in the Figure consists of 4 types of different sub-models: Queue, Machine, Disturbance and Assign. In the queuing theoretic view, it is a single-queue single-server model with assignment. A single lot-message not only has a reference to its processing information according to the product type to be made, but contains statistical data such as waiting time, served time, current processing step, and so on. The model 'Disturbance' simulates unpredictable failure and repair of a machine. Considering the behavior of these models, if a lot-message arrives at port 'in', model 'Assign Atomic' distributes it to one of the queues, each of which is attached to a model 'Machine'.

We assume that a lot is assigned to the shortest queue. A lot in a queue should wait until the machine attached to the queue is idle. Then the queue dispatches a lot based on its dispatching rule, and the lot is sent to the machine. Note that each workstation model could have a variable number of queue-machine pairs whose number comes from SEMATECH data set specifications.

The machine model in Figure 5 follows typical normal processing behavior, i.e., Idle > Load > Process > Unload > Idle. An external disturbance forces the machine to stop normal processing, so that some lot in work is returned for rework or another lot is sent to output port 'out' with a failure indication. We assume that group setup time and lot transfer time in the specification are contained in 'Load' and 'Unload' time for simplification. Detailed parameters of this model are obtained from SEMATECH data set 5.

Figure 6 depicts the whole virtual plant model specified in Figure 1, which consists of an experimental

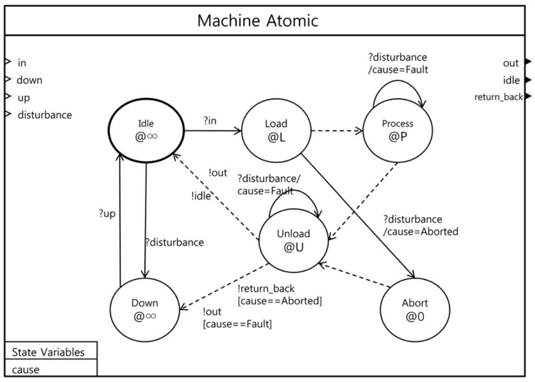

Fig. 5. Atomic Machine Model

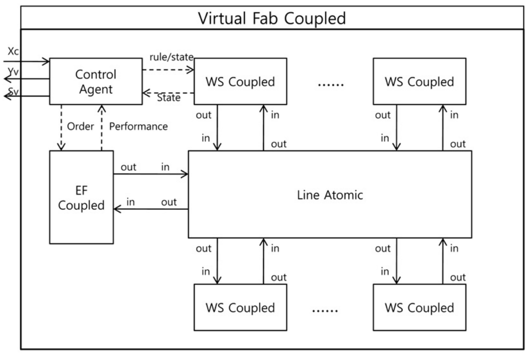

Fig. 6. Coupled Virtual Fab Model

frame which generates initial lots and accepts produced lots from which we make statistics, a line model which represents routes from workstation to workstation, and a number of workstation models which contain queues and machines as in Figure 4.

Finally, a control agent is responsible for interfacing between the controller and the virtual plant model. There are hot message lines (couplings) between the control agent and every other model for processing commands.

4. EXPERIMENTAL RESULTS

4.1. Experiment I: Throughput

The experiments are intended to demonstrate that the proposed real-time control framework works well, as well as that the adaptive rescheduling scheme of the controller outperforms any other with a fixed dispatching rule. The performance index includes average waiting time, average cycle time, and throughput per hour. We conduct experiments on a desktop computer, and the implemented simulation software for the framework is written in C++ with simulation engine DEVSim++ 3.0[12].

Experiment I compares the performances among schemes with fixed dispatching rules for a period of 6 months of logical processing time, and a simulation-based rescheduling scheme assuming that unpredicted disturbances occur. Lot input rate is constant as specified in the specification. Disturbances are injected in exponential distribution, with mean time between failures and mean time to repair as specified. The objective (cost function) of the control is to maximize throughput with the given set of dispatching rules.

Figure 7 shows the simulation results of Experiment I over six months. The first rule 'Controller' means the adaptive rescheduling scheme proposed in the paper and the others are schemes with fixed rules. The left Y axis is time in hours, and the other axis is the number of lots.

The results show that the proposed adaptive control scheme (rule Controller) outperforms any other: the cycle time is 36% faster, the wait time is 26% less, and throughput is greater by more than 90%, on average.

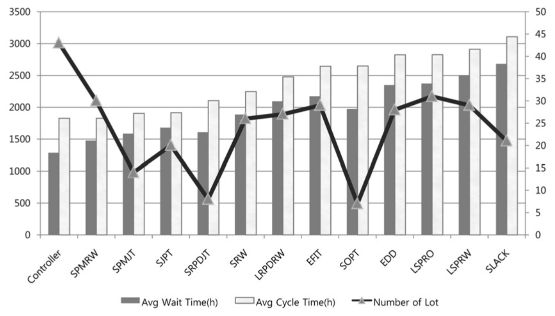

Fig. 7. Performances of Experiment I

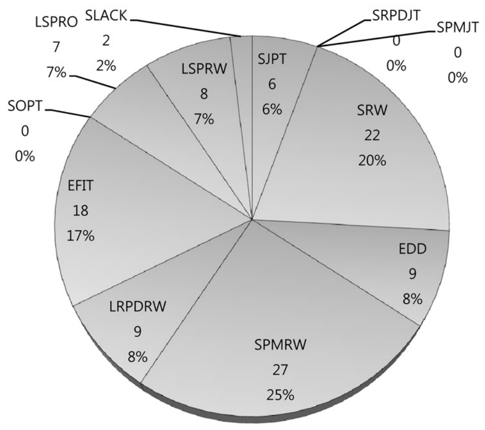

Fig. 8. Distribution of Rule Selection by the Controller

Also note that SPMRW, SRW, and EFIT in Table 2 are dominant rules selected very frequently by the controller as Figure 8 Distribution of Rule Selection by the Controller shows. The SPMRW rule chooses a lot with smallest processing time multiplied by remaining work, as shown in Table 3. Figure 9 shows the throughput change over time obtained from the experiment I. After one month, a transient state, the throughput shows above 0.008; thus, if the control objective is to keep the desired throughput above this value, the controller can satisfy the control objective by the adaptive controller scheme.

4.2. Experiment II: adaptive control with dominant rules

In this experiment, we use only three dominant rules obtained from Experiment I, SPMRW, SRW and EFIT,

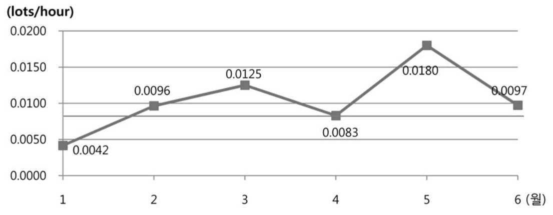

Fig. 9. Throughput Control Over Time

Table 3. Performance of Modified Algorithm

Controller Rule	Wait Time (hours)	Cycle Time (hours)	Number of Lots	Ratio
12-rule	1286.01	1826.66	43	100%
3-rule	1553.19	2089.45	41	95%
3-rule random	1738.56	2208.02	29	65%

for the evaluation rule set to see if the controller produces the similar results, hopefully with little loss of performance.

The second row in Table 3 shows that controlling with the dominant 3 rules gives us similar performance with 5% degradation compared to the 12-rule control in Experiment I, though the computing power required for the 3-rule controller is about 1/4 of the 12-rule controller, since evaluation is carried out with only 3 rules. The third row is the result from random rule selection by the 3-rule controller without evaluation, which shows the worst result; again, it proves the effectiveness of the proposed adaptive control scheme.

5. CONCLUSION

This paper proposed a real-time adaptive control framework for job scheduling in semiconductor fabrication using a system theoretical approach. Through design and implementation of the framework based on a DEVS modeling simulation environment, we show that the proposed framework is quite effective. Experimental results show that the simulation-based adaptive evaluation scheme produces better performance than one with any fixed dispatching rule. Since the approach is explicit and the cost function of control objectives and evaluation rules can vary, this framework can be applied to real factories in flexible and explicit ways.

REFERENCES

1. J. Hunter, et al., "Understanding a semiconductor process using a full-scale model," *IEEE Transactions on Semiconductor Manufacturing,* vol. 15, no. 2, pp. 285-289, 2002.
2. C.M. Harmonosky, "Simulation-based real-time scheduling: Review of recent developments," *In Proceedings of the 27th Winter Simulation Conference, Arlington, Virginia, USA,* pp. 220-225, 1995.
3. Krishnamurthi and M.S. Vasudevan, "Domain-based online simulation for real-time decision support," *In Proceedings of the 25th Winter Simulation Conference, LA, CA, USA,* pp. 1304-1312, 1993.
4. J.S. Smith, et al., "Discrete event simulation for shop floor control," *In Proceedings of the 26th Winter Simulation Conference, Orlando, FL, USA,* pp. 962-969, 1994.
5. S. Manivannan and J. Banks, "Real-time control of manufacturing cell using knowledge-based simulation," *In Proceedings of the 23th Winter Simulation Conference, Phoenix, Arizona, USA,* pp. 251-260, 1991.
6. J. Fowler, "Modeling and Analysis for Semiconductor Manufacturing Laboratory: SEMATECH Dataset," Arizona State Univ., http:// www.eas.asu.edu/~masmlab/, 2006.
7. C. Rajendran and O. Holthaus, "A comparative study of dispatching rules in dynamic flowshops and jobshops," *European Journal of Operational Research*, vol. 116, pp. 156-170, 1999.
8. M.H. Kim and Y.D. Kim, "Simulation- based real-time scheduling mechanism in flexible manufacturing system," *J. Manufacturing Systems,* vol. 13, pp. 85-93, 1994.
9. K.D. Kim, et al., "Simplification methods for accelerating simulation-based real-time scheduling in a semiconductor wafer fabrication facility," *IEEE Transactions on Semiconductor Manufacturing,* vol. 16, no. 2, pp. 290-298, 2003.
10. L. Monch, "Simulation-based benchmarking of production control schemes for complex manufacturing systems," *Control Engineering Practice,* vol. 15, pp. 1381-1393, 2007.
11. Zeigler, et al., *Theory of Modeling and Simulation*. 2nd ed., Academic Press, 2000.
12. T.G. Kim, *DEVSim++ User's Manual 3.0,* KAIST, http://smslab.kaist.ac.kr/DES/, 2008.
13. J. Kim, et al., "Dynamic release control policy for the semiconductor wafer fabrication lines," *J. Operational Research Society,* vol. 47, no. 12, pp. 1516-1525, 1996.
14. J. Kim and T.G. and Kim, "Parametric Behavior Modeling Framework for War Game Models Development Using OO Co-Modeling Methodology," *In 2006 Spring Simulation Multi-conference, Huntsville, USA,* pp. 69-75, 2006.

송 해 상 (hssong@seowon.ac.kr)

1991　한국과학기술원 전기및전자공학과 석사
2000　한국과학기술원 전자전산학과 공학박사
1999～2000　고등기술연구원 연구원
2002～현재　서원대학교 컴퓨터공학과 부교수

관심분야 : 이산사건시스템 M&S, 임베디드 시스템, 시스템공학.

이 재 영 (jaeyoung3.lee@hynix.com)

2005　금오공과대학교 전자공학부 학사
2009　KAIST 전기및전자공학과 석사
2009～현재　하이닉스반도체 연구소 설계 2팀

관심분야 : 반도체 공정, DRAM 설계, Low Power DRAM.

김 탁 곤 (tkim@ee.kaist.ac.kr)

1988　Univ. of Arizona, 전기및컴퓨터공학과 박사
1980～1983　부경대학교 통신공학과 전임강사
1987～1989　(미)아리조나 환경연구소 연구엔지니어
1989～1991　Univ. of Kansas 전기및컴퓨터공학과 조교수
1991～현재　KAIST 전기및전자공학과 교수

학술활동 : 한국시뮬레이션학회 회장 역임, 국제 시뮬레이션학회(SCS) 논문지(*Simulation*) Editor-In-Chief 역임, SCS Fellow, M&S 기술사(미국), Who's Who in the World (Marguis 16th Ed., 1999) 등재.
자문활동 : 국방부/합참 자문위원 역임, KIDA Fellow 역임.
현재 - 연합사, 해군전발단, ADD, 국방기술품질원 자문위원.
관심분야 : 모델링/시뮬레이션 이론, 방법론 및 환경개발, 시뮬레이터 연동.