A STUDY OF DEVS-BASED PROCESS SCHEDULING ON MULTI-USER SEMICONDUCTOR TEST EQUIPMENT

Soonchul Lim[(a),(b)], Youngsin Han[(a)], Chilgee Lee[(a)]

[(a)] College of Information and Communication Engineering, Sungkyunkwan University
[(b)] Development and Evaluation Group, Memory Business, Samsung Electronics

[(a)]scandth1@naver.com, yshan@skku.edu, cslee@skku.edu

ABSTRACT

Single processor semiconductor test equipment inevitably experiences idle time between tests. This idle time is increased when multiple operators use the same equipment. An increase in idle time is considered a loss factor and produces low equipment efficiency; therefore, it is important to decrease it. In this paper, we offer two methods to effectively decrease idle time in multi-user test equipment. The methods proposed here were developed using an atomic model and were coupled with discrete event system specification methodology. Features of our model include idle time that can be decreased more than the typical sequence and equipment status that can be monitored beforehand without adding extra time.

Keywords: equipment efficiency, idle time, process scheduling, DEVS

1. INTRODUCTION

The market for semiconductor memory devices is rapidly changing. The demand for desktop PC memory is stable or slightly decreasing, but the demand for memory for mobile devices is rapidly increasing. These changes require an increased investment and reduce the life of certain equipment. The effective use of limited resources is essential in the semiconductor industry, which already requires heavy investment.

Equipment efficiency can be expressed numerically and most manufacturing processes apply various methods to increase equipment efficiency. Semiconductor test equipment is used to screen devices and evaluate their characteristics. Test equipment efficiency needs to be improved by optimizing test items, decreasing test times, and reducing unnecessary idle time. Overall equipment efficiency (OEE) is one of the methods used to illustrate how effectively equipment and resources are utilized. The overall performance of a piece of equipment or a factory is always governed by the cumulative impact of three OEE factors: availability, performance rate, and quality rate (A.J. De, Ron, 2006). The OEE is defined as:

$$\%OEE = (\%Availability) \times (\%Performance) \times (\%Quality) \quad (1)$$

Our proposal relates to the availability of the OEE factors.

2. BACKGROUND AND METHODOLOGY

2.1. Background

The test equipment process sequence applied by multiple users is expressed as shown in Figure 1. User *A*, occupying the test processor, needs idle time (*I1*) to set up the test environment. After the test program is executed, idle time (*I2*) is generated from the end of the program, or by user interruption, before starting the next program. After user *A* completes the procedure on the test processor, idle time (*I3*) is generated until user *B* occupies the test processor. The process sequence hereafter is identical to the previously described sequence of user *A*. Thus, idle time (*Itp*) between users occupying the test processor and idle time (*Is*) between the test programs, are generated. The total idle time is defined by the formula below:

$$I_{total} = \sum I_{tp} + \sum I_s \quad (2)$$

The total idle time is one of the loss items in an availability factor; the higher the total idle time, the lower the system's efficiency. Process scheduling is a method used to decrease the waiting time and a common representation of process scheduling is a queuing system (Silberschatz and Galvin, 1994; Hopp, 2008). We propose the queuing system using discrete event system specification (DEVS) methodology to minimize the total idle time on multi-user equipment.

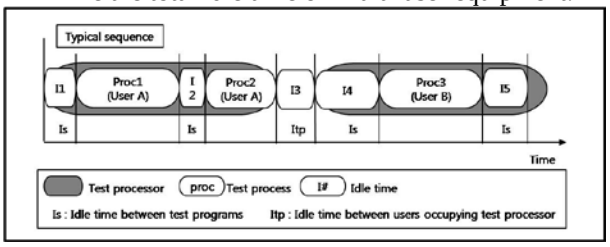

Figure 1: Typical sequence for multi-user equipment

2.2. Methodology

The DEVS is a modeling methodology used in a system that operates discrete events in linear time (Hill, 1996). This methodology is based on a set theory to extract the structure and behaviors of the model. It is easily expressed in hierarchical and modular systems. To

provide these features, a DEVS has an atomic model and a coupled model. An atomic model (*M*) is organized to express the dynamic characteristic, as follows:

$$M = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

The coupled model (*CM*) expresses the interaction between the components of the system and the hierarchical structure of the system, as follows:

$$CM = < X, Y, M, EIC, EOC, IC, SELECT >$$

Detailed descriptions and modeling methods of DEVS can be found in Zeigler, Praehofer and Kim (2000), Kim (2007), and Han and Song (2012).

3. SYSTEM MODELING

We propose a queuing system that consists of the InQueue model, the processor model, and the InQueuehandler model using the DEVS to produce a system with minimal idle time.

3.1. InQueue model

A state diagram of the InQueue model is shown in Figure 2. There are three input messages and two output messages. The initial state of the InQueue model is the WaitForPgm state with a time of infinity. The InQueue model accumulates the program in a queue if receiving a PgmRaw message. The model produces PgmRaw and Qval messages if receiving ReqPgmRaw and ReqQval messages, respectively.

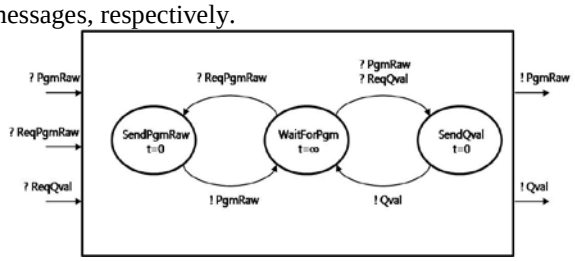

Figure 2: State diagram for the InQueue model

3.2. Processor model

The processor model's behavior is depicted in Figure 3. The processor model has two states, with an input message and an output message. The initial state of the processor model is the WaitForPgm with infinity time. If receiving a PgmRaw message, the state of the model is changed to BUSY which has a random *ta*. After executing the PgmRaw, the model generates a TestDat message and returns to its initial state.

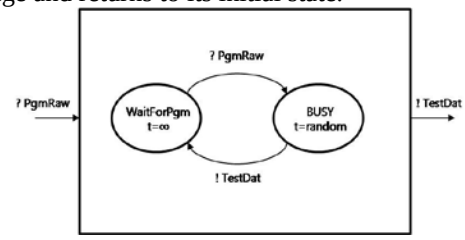

Figure 3: State diagram for the processor model

3.3. InQueuehandler model

A state diagram for the InQueuehandler model is provided in Figure 4. In the figure, there are three input messages and four output messages. The model has a WaitForQval state with infinity time at the initial state. If Qval is not 0, the model moves to the CheckTP state. If the test processor state is free, the model goes to the ReqForPgmRaw state. If it is not free, the model goes to the ReqForPreemptiveTP state and waits until the specified time. A ReqForPgmRaw message is produced and the state of the model is changed to WaitForPgmRaw. When receiving a GetPgmRaw message, the model goes to the SendPgmRaw state and immediately produces a SendPgmRaw message. Subsequently, the model goes to WaitForProcessDone with infinity time. After receiving the GetDoneSig message, the state of the model is changed to SendDoneSig state and the model generates a SendPgmRaw message. The CheckInQ state produces a Qval message and the model returns to the WaitForQval state.

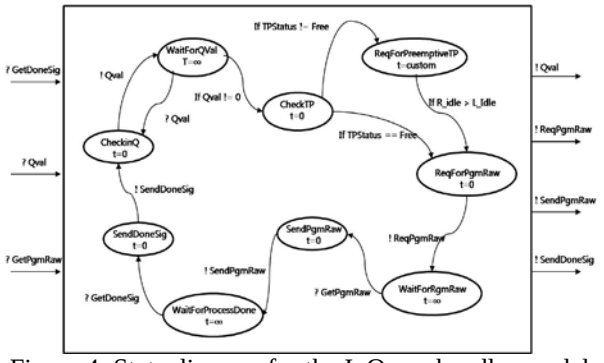

Figure 4: State diagram for the InQueuehandler model

3.4. Coupled model

Figure 5 represents the overall coupled model of the queuing system to apply idle time to the effective process. This coupled model consists of three atomic models and combines the InQueue, processor, and InQueuehandler models. In the InQueuehandler, the WaitForPgmRaw state changes to the SendPgmRaw state when receiving PgmRaw from the inGetPgmRaw port. PgmRaw is transferred from the InQueuehandler output port to the processor model and the Qval is changed. The inputted PgmRaw is executed by the defined environment process and the model produces TestDat messages for the user that initially provided the PgmRaw to the InQueue model.

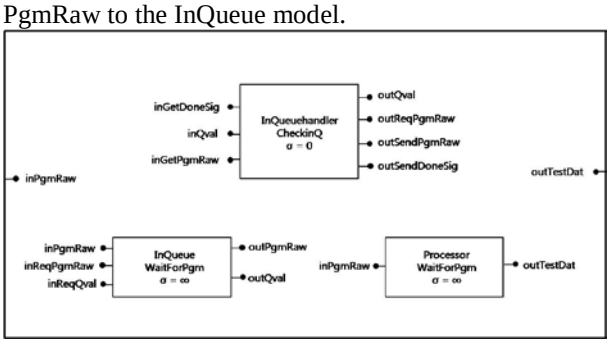

Figure 5: Overall view of the coupled model

4. RESULTS

4.1. Queuing system using the DEVS

Our queuing system using the DEVS is shown in Figure 6. This method can be executed without the loss of time. The upper sequence in Figure 6 is a typical user sequence (*A*, *B*, *C*); it is necessary for each user to occupy the test processor. In Figure 1, as described above, idle time *(I1 ~ I4*) occurs due to latency times (*Itp* and *Is*). The queuing system using the DEVS is executed with non-preemptive processes, as in the lower sequence in Figure 6. This method has features for minimizing *Itp* and *Is*; however, it has an assumption that requires the same hardware infrastructure for proc1, proc2, and proc3. In the experimental results, as shown in Figure 7, this method works more efficiently for multi-user equipment than the previous process, but the data for the single-user equipment does not seem to be effective.

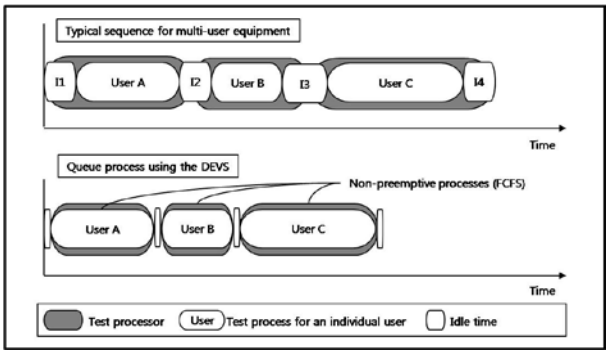

Figure 6: Queue process using the DEVS

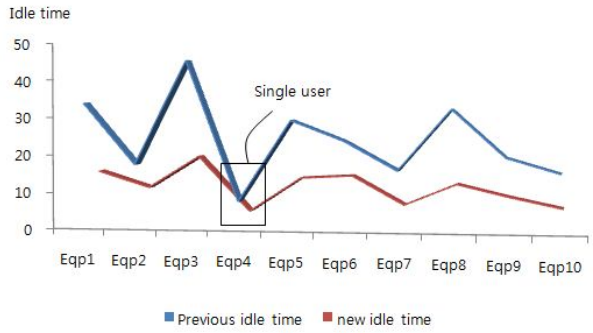

Figure 7: Comparison of the idle time

4.2. Self-diagnostic process using the DEVS

The upper image in Figure 8 shows a typical sequence. Idle time occurs between the test programs. The length of idle time is different according to the type of work the user is performing. The lower image in Figure 8 shows our method with a self-diagnostic process, applying a preemptive process to the user's idle time. If the idle time (*I3*) exceeds the limited time defined by the system user, the user's test processor is terminated and changed to the system user. Subsequently, a self-diagnostic process (SDP) is executed by the queuing system. For effective execution, the running time of the SDP should be considered, depending on the type of work. Once the SDP ends, the test processor is free for the next user's process.

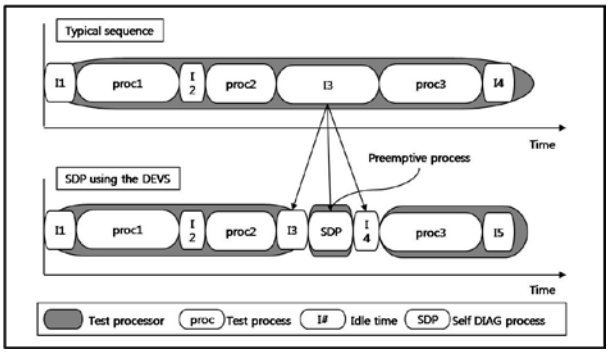

Figure 8: Self-diagnostic process using the DEVS

5. CONCLUSION

Using limited resources and minimizing loss of time are important factors in test equipment efficiency. In this paper, we proposed two methods to effectively utilize idle time during test processes. One of the proposals is a queuing system with non-preemptive scheduling to minimize idle time on multi-user equipment. The other is a self-diagnostic process with preemptive scheduling, applying the user's idle time. As shown in the results, idle time using the DEVS-based queuing system is effective for multi-user equipment but its effectiveness in single-user systems is negligible. We also confirmed that self-diagnostic processes with the DEVS, applying the user's idle time, is available for such systems.

REFERENCES

A.J.De, Ron, 2006. OEE and Equipment effectiveness: an evaluation. *International journal of production Research*, Vol 44, No. 23, 4987-5003.

Avi Silberschatz, Peter Galvin, 1994. Operaing system (8th Edition), *Addison-Wesley*.

Wallace J. Hopp, 2008. Single server queuing models. *International Series in Operations Research & Management Scienceh*, Vol 115, pp 51-79.

David R. Hill, 1996. Object-Oriented Analysis and Simulation. *Addison-Wesley*

Bernard P. Zeigler, Herbert Praehofer and Tag Gon Kim, 2000. Theory of Modeling and Simulation (2nd Edition), *Academic Press*.

Tag Gon Kim, 2007. EE612 Lecture Note, EECS, KAIST, http://smslab.kaist.ac.kr/Course/EE612/.

Tag Gon Kim, 2007 DEVSim++ User's Manual, http://smslab.kaist.ac.kr

Youngsin Han, Hae Sang Song, 2011. Co-modeling Methodology for Semiconductor Manufacturing and Automobile. *Communications in Computer and Information Science*, Vol 341, pp 7-14