Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

DISCRETE-EVENT SIMULATION USING SYSTEMC: INTERACTIVE SEMICONDUCTOR FACTORY MODELING WITH FABSIM

Holger Vogt

Fraunhofer IMS
Finkenstr. 61
47057 Duisburg, GERMANY

ABSTRACT

Semiconductor fabrication factories are large enterprises with many toolsets, each having multiple production machines. The process flow is highly reentrant, therefore modeling is best done by discrete-event simulation. To describe such a fab, the author has developed a new discrete event simulator called FabSim. It is written in C++. As the simulation engine it uses SystemC, a C++ class library originally developed for modeling "Systems on a Chip". The factory with its machines and lots traveling and in process is mapped onto SystemC like a hardware description during RTL (register transfer) modeling. The resulting simulator is compact, fast and efficient. In a special configuration as a MS Windows dynamic link library, the simulator is fully interactive. At any time you may define a stop in the simulation flow, retrieve the state of the whole system, change parameters, add lots, or even enter a new state and continue with the simulation.

1 INTRODUCTION

Semiconductor fabrication factories today represent one of the most complex industrial processes. Investments into a modern fab are far beyond the 1 billion US$ mark. Therefore during the planning phase and during fab operation simulation of the whole factory is an absolute requirement. During planning the optimum toolset combination for a given throughput and the intended process flows has to be determined. During manufacturing the factories continuously search for the optimized machine usage, minimum cost, and due date delivery.

The simulation task is very complex. Multiple toolsets with several machines each serve a large variety of process flows with more than 400 steps each. The process flow is highly reentrant, that is toolsets a visited more than once during processing. For example all. wafers will enter the lithography area more than 20 times.

Analytical simulation is thus limited to special cases with high abstraction. Discrete-event simulation (Fishman 2001) is the method of choice to describe the whole factory.

Today fab engineers very often use standard simulator packages with dedicated settings, see Hunter et al. (2002) or Shikalgar, Fronckowiak, and MacNair (2002).

To take advantage of the flexibility and high speed of a universal computer language like C++ and to relieve the fab engineer from becoming absorbed by simulator and model setup problems, we have developed a compact fab simulation tool named FabSim. It is based on the simulation engine SystemC, which is illustrated in Section 2. The simulator structure is described in Section 3. Section 4 shows the capability of FabSim as a truly interactive simulator. Simulation examples are presented in Section 5.

2 SYSTEMC

SystemC stems from numerous efforts by members from the Open SystemC Initiative <www.systemc.org>. It is a new system-level specification and design language which contributes to functional modeling and hardware description (Grötker 2002). Thus it adds to and enhances the high definition languages like VHDL and Verilog which are widely used for integrated circuit and system on chip design.

In the course of evaluating this language in the IC arena the idea came up to apply SystemC to the totally different field of discrete-event simulation. SystemC allows to model pin-accurate, cycle-accurate and functionally accurate hardware. If factory "hardware" could be mapped onto SystemC entities, it could be modeled with high efficiency.

The SystemC simulation engine is a C++ class library, available by free download at <www.systemc.org> for several operating systems. It contains an event-driven simulation kernel, classes of modules with process member functions, as well as classes of channels (ports and signals) for interconnection of modules. Events may be defined as changes of signals at the input ports of a module. An event driven by a continuously running clock emulates a syn-

chronous system. The process inside a module is the basic unit of functionality. Processes are organized as coroutines and provide the mechanism for concurrent simulation. The simulation model is set up during compilation of the code which describes the system. At runtime, first of all (multiple) objects from modules and interconnecting channels are instantiated. The instantiation may occur dynamically, driven by external data entered into the simulator. After that the simulation proceeds for a given amount of time.

3 SIMULATOR DETAILS

To create the simulator, we need a suitable factory model and map it onto the SystemC entities described above.

The semiconductor factory comprises of toolsets, each with one or several identical machines. Lots are processed inside the machines and move from toolset to toolset according to the process flow chart. Toolsets are objects instantiated from a SystemC module. This module contains a process which is an universally applicable machine model. The number of toolsets is determined by the factory layout and instantiated automatically at runtime. As many processes are started per toolset object as machines are needed. The number of toolsets is currently limited (tested) to 128. The toolset module is responsible for organizing the buffer in front of the toolset, for handling lot priority, organize batch processing. Two priority levels are available above standard (no priority, prio 0). Prio 2 will always put the lot in front of the buffer. Hot lot prio 1 will preemptively reserve machines. As soon as the hot lot arrives at the next toolset, it will be processed immediately. Lots may be grouped into batches and processed together (e.g. in diffusion furnaces). If required, distinct toolsets may initiate reworks and send back lots to temporarily follow a rework flow chart.

Each process modeling a machine is a finite state machine with states like IDLE, DOWN LOAD, PROC, UNLOAD and others. The machine model contains features like multiple down times (scheduled or statistically distributed), setup procedures (especially for implanters and litho clusters), loading and unloading, and up to 9 recipes with different lot processing times. The maximum number of machines per toolset is in principal limited only by the memory made available to FabSim, but currently set to 50.

Signals between the modules define the lot movement paths. Whereas digital hardware models require simple boolean signals, we also may send and retrieve complex C++ types. Here we use a lot carrying type, derived from a generic lot class.

After start FabSim reads in toolset and machine data and sets its toolsets and interconnection objects as shown in Figure 1. Lots flow back and forth between the toolsets and a scheduler. The scheduler gets the information about flow charts and directs lots to the appropriate toolsets. Lots ready or scrapped are also handled here.

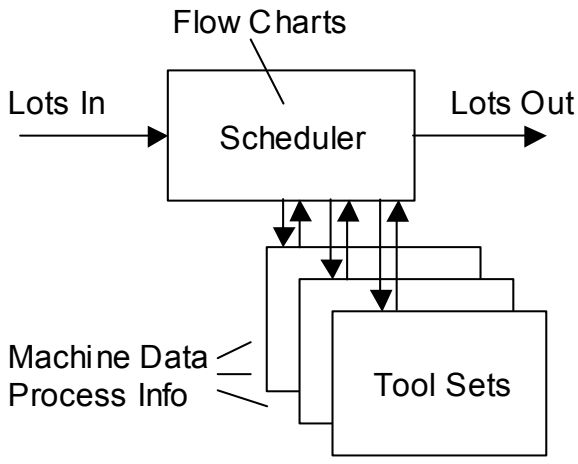

Figure 1: Basic Structure of FabSim

Lot movement is modeled by taking either a fixed transport time (determined by the sending toolset) or by reading in a from-to matrix at initialization, where a file contains individual transport times.

The user will provide all input data as ASCII files, as described in more detail in chapter 4. Then he starts the simulator with command line parameters to set various options. FabSim reads the input data, instantiates all objects and thus sets up the fab model (in a fraction of a second). During the simulation phase all processes are interrogated synchronously. In case of an event (lot movement) all necessary calculations are done and all the states are updated.

Computation time depends on the time step (clock cycle) chosen, the number of toolsets and machines and some options. A typical fab with tools set for 500 wafer starts per day, will require 135 seconds of CPU time on a 1.4GHz LINUX PC for a simulation period 139 days at a time resolution of 1 minute.

FabSim outputs comprise of several ASCII files containing lots processed or scrapped, machine usage. Optionally the simulator creates additional log files on buffer occupancy, machine idle or down time, and a large step by step log. EXCEL based tools serve to arrange and display the output data graphically.

4 SETTING UP THE FACTORY MODEL

To generate the factory model, the user defines several ASCII data files as an input to FabSim. Three files are required to provide the toolset definition, a from-to matrix, and the lots to be started. Additional files contain the process flows as run sheets, one for each process. Furthermore rework run sheets may be dedicated to any step in a process flow as separate files.

Currently the user has to rely on the extensive machine model presented by FabSim, there is no user interface to add extra code.

The factory is composed of an ensemble of toolsets. Each toolset is a row in the toolset definition file. The user has to provide data like: number of machines in a toolset, minimum and maximum batch size, delays for loading and unloading, mtbf, mttr, scheduled downtimes, type of setup and setup times, lot size dependence of process step duration, up to nine process step times.

The from-to matrix contains transport times between any couple of toolsets, overriding a toolset specific time.

The sequence of lots to be started is set in the third file, one line per lot. The data include start time, product, lot size, process, priority. Loops may be set for simplifying the definition of a regular sequence of lot starts. In the interactive mode described in chapter 5 this file is not used, lots may be started by the supervisor program.

Run sheet files contain one row per process step, setting the step number, the toolset, the recipe and a rework probability of this step.

An additional input file will be required by a coming FabSim version to describe operator availability, including operator count, experience and shift data.

5 INTERACTIVE SIMULATION

The simulator presented so far is a single executable file. You may start it, provided all input files are available, from the command line or out of a batch file. It will simulate for the prescribed duration and then generate all output files.

In a completely different configuration we have compiled FabSim into a dynamic link library (currently only available for MS Windows). A supervisor program now may control FabSim.dll and allow fully interactive simulation. The basic simulator structure still resembles Fig. 1. The dll however exports several function which serve to control nearly every aspect of the simulation. The general structure is shown in Fig. 2.

In a batch mode FabSim.dll behaves like its command line counterpart. In interactive mode of FabSim we firstly initialize FabSim by loading the dll and reading all factory input data. The factory status saved during a previous simulation may now serve as a starting point.

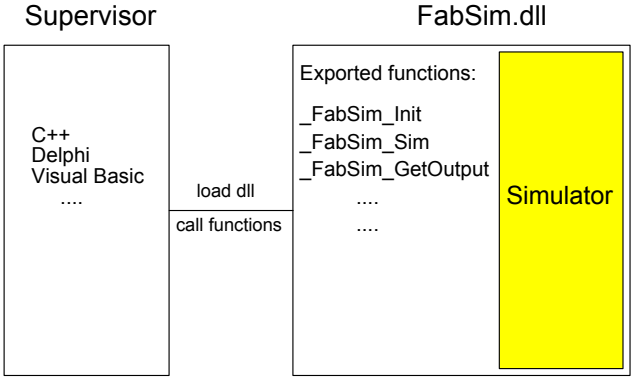

Figure 2: Structure of FabSim Interactive

A lot may now be entered. The supervisor then starts FabSim.dll for at least one clock cycle. After this simulation period has passed, FabSim stops and waits for input. All data are stored and are accessible by several of the exported functions. You may search for lots, save the complete lot or fab status, enter a new lot, move a lot out of the process flow and onto a shelf, retrieve another lot from the shelf for further processing, set machines down for a given period, change lot priority, ask for toolset and machine utilization. You may even toggle from the push mode (each lot leaving a toolset will find its way into the buffer and then into an idle machine of the following toolset) into a mode where the supervisor actively has to move each lot from the toolset buffer into a machine currently available. Thus the supervisor has full control over the simulation procedure. During periods where FabSim.dll is allowed to run uninterrupted, it will retain the same high simulation speed as the command line version.

We have developed a sample supervisor program which offers a graphical user interface and provides access to the functions controlling FabSim.dll.

Several applications of an interactive simulation come into mind. Loaded with the actual fab status, it may predict fab behavior for a given period (e.g. the next few days). If the time resolution of one minute is not sufficient, all time based inputs simply may be related to another base unit (e.g. one tenth of a minute, or seconds). Of course simulation time for a given real time will increase. Simulation for one week with resolution of one second will take 80 seconds under MS Windows.

Individual lots may be pushed to predict urgent product delivery.

Fab throughput may be controlled. A simple control mechanism by keeping WIP constant is provided as standard, other control mechanisms based on buffer occupancy or bottleneck usage may offer in depth factory control.

6 SIMULATION EXAMPLES

Some simulation results of FabSim Interactive are shown in this section.

Fig. 3 depicts a simulation of a small pilot line with 42 toolsets. Two simulation runs are plotted with lot cycle time versus lot start time. Each lot is represented by a data point. The first run simulates the cycle time with 100% standard lots started at regular intervals. In the second run 5% “super hot” lots and 95% standard lots are started.

The hot lots cycle through the fab with optimum speed, however the remaining standard lots are slowed down by 10% compared to the cycle time only standard lots.

The graphs showing the standard lots are relatively “noisy”. The pilot line has only one, sometimes two machines per toolset. Thus machine downtime immediately leads to a kink in the cycle time.

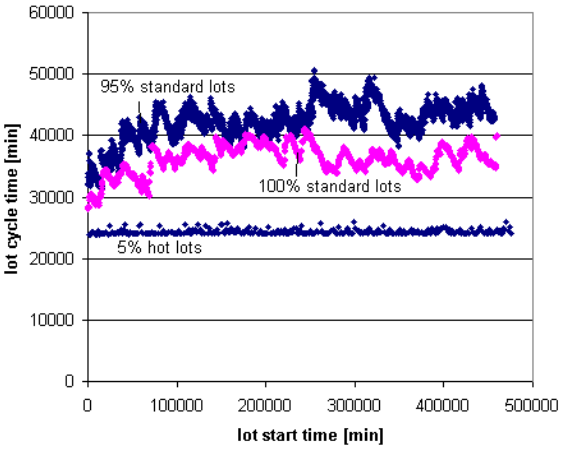

Figure 3: Influence of Hot Lots on Lot Cycle Time

In a factory with 500 wafer starts per day the lot priority of lot 6000 (processed in a quarter micron process) has been increased at 445.000 minutes. The plot in Fig. 4 shows the lot's progress through the fab. After changing the priority from standard to super hot, the slope increases considerably.

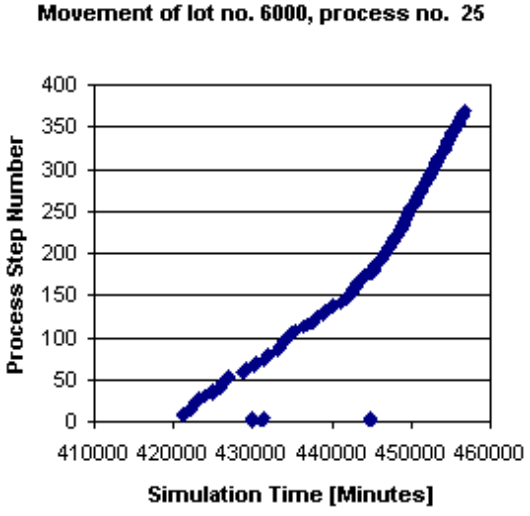

Figure 4: Lot Movement Versus Time. Priority Set at 445.000 Minutes

Three small interruptions can be seen where the lot suffers a rework.

All data are excerpted from the log file which stores every entry of a lot into a machine. Thus you may analyze data for each lot after the simulation.

A dramatic influence of downtime is plotted in Fig. 5. Three out of five I-line steppers are down for 100.000 minutes. A drastic increase of lots in the buffer in front of the litho toolset is accompanied by a strong increase in cycle time. Even another 100000 minutes after the steppers are online again, and the buffer usage has returned to normal (Fig. 6), the fab has not yet recovered.

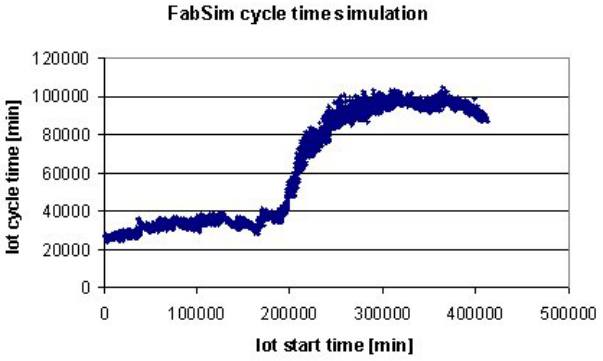

Figure 5: 3 out of 5 Steppers Down at 200.000 for 100.000 Minutes

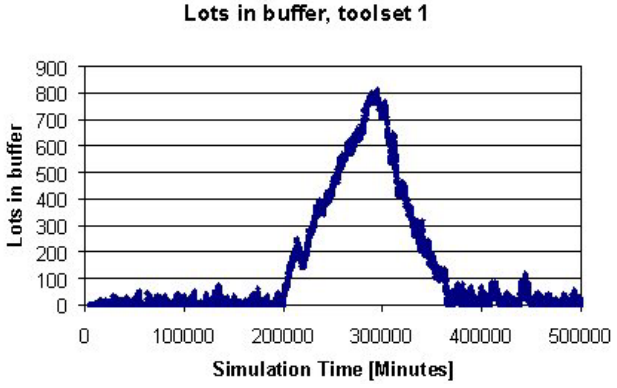

Figure 6: Lots in Buffer During Downtime of Steppers

7 SUMMARY AND OUTLOOK

We have presented a compact simulator including a full-scale model of a semiconductor factory. It is ready for use, if factory and process flow data are provided. Simulation is fast and efficient because of its SystemC engine and C++ code. Output data in ASCII format, together with simple visualization tools offer quick access to relevant simulation results.

Still some enhancements have to be added. In small to medium fabs operator availability is not always guaranteed. Its influence therefore should be modeled.

A suitable interface to factory data will easy simulation setup even more. Experience will tell if a universal interface may be generated.

FabSim.dll may be loaded in several instances to model factories of a complete IC manufacturing chain (from wafer over chip to back end). Some research into a suitable supervisor is due.

REFERENCES

Fishman, G. S. 2001. *Discrete-event simulation.* New York: Springer.

Grötker, T. et al. 2002. *System design with SystemC.* Boston: Kluwer Academic Publishers.

Hunter, J. et al. 2002. Understanding a semiconductor process using a full-scale model. IEEE Transactions on Semiconductor Manufacturing, vol. 15, no. 2, May 2002: 285-289.

Shikalgar, S. T., D. Frockowiak, and E. A. MacNair. 2002. 300mm wafer fabrication line simulation model. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan et. al. 1365-1368. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHY

HOLGER VOGT has a diploma degree in electrical engineering. His dissertation dealt with CMOS on buried nitride, a new SOI technology. Since 1985 he is with Fraunhofer Institute of Microelectronic Circuits and Systems, Duisburg, Germany. He has managed the CMOS pilot line at IMS. Currently he is responsible for R&D on innovative processes and devices, including smart sensors and power devices. He heads the program to set up 0.25 µm technology on 200 mm wafers at IMS. Since 1997 he is also professor at the EE&CS department of University Duisburg-Essen, Germany, teaching semiconductor technology and packaging.

The FabSim web site is <http://www.fabsim.com>. <holger.vogt@ims.fraunhofer.de> is the authors email address.