Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

A MODULAR STRUCTURE FOR A HIGHLY DETAILED MODEL OF SEMICONDUCTOR MANUFACTURING

Sarah Jean Hood
IBM Corporation
T.J. Watson Research Center
Yorktown Heights, NY 10598

Amy E. B. Amamoto
Antonie T. Vandenberge
Systems Modeling Corporation
Sewickley, PA 15143

ABSTRACT

A model of semiconductor manufacturing lines, using the discrete event simulation language, SIMAN, is described. A detailed, flexible, and generic model of this large, complex system was constructed by using a highly modular structure. A user interacts with the model through an interface which requires no programming and no recompiling even if major changes to the process sequence, number of resources (tools and operators), queue control, or number of job types are made.

1. INTRODUCTION

A discrete event simulation model has been developed for simulating large, complex, semiconductor manufacturing lines. The goal in designing the model was to represent the manufacturing lines in considerable detail, yet provide a flexible interface for structuring different configurations. Thus, adding a new tool group, process step, or product type requires no programming, but only data entry through the user interface. All anticipated policies concerning job release into the line, job selection, and rework were coded and the user simply sets a switch to select among them. In the event that the user wishes to add a policy, the modular code ensures a direct interface with the existing 10,000 lines of code. The reasons for modeling in such detail are given in Section 3.

2. THE MANUFACTURING ENVIRONMENT

The semiconductor manufacturing environment is a complex and dynamic one. After the transistors have been built, the process of joining them into circuits, called the personalization process, involves hundreds of process steps at many tens of tool groups. The flow is highly reentrant; a job may revisit one tool group ten times or more as it advances through its process plan. Figure 1 shows a typical process plan. A job starts the personalization process at Level A Apply Photoresist. When it exits Level A Clean it enters Level B Quartz Deposit and so on until it exits the personalization process at Level N Via Etch. In addition to the highly reentrant flow, a line may handle multiple products. The competitive market drives the frequent introduction of new products with smaller, tighter tolerances; thus, tools may be operated at the limit of their specifications and therefore, may introduce significant variability.

The cycle time, the time elapsed from when a job enters the line to when it exits the line, is on the order of months. One of the objectives in managing the line is to reduce cycle time. Reducing the cycle time results in accelerated learning due to more cycles being completed in the same amount of time, quicker response to changing demands, and shorter exposure time to contaminating particles.

3. REASONS FOR A DETAILED MODEL

In the past, analytic models such as linear programming and mean value analysis, simulations with many simplifying assumptions, and manufacturing experience have been exploited to their limits to reduce cycle time. An alternative which could provide thorough (detailed) insight was needed. The long cycle times and complex dynamics made the alternative of evaluating new scheduling or flow control policies by experimenting on the line itself difficult (not to mention using resources that could be going to revenue generating production). Thus, a detailed model was needed to serve as a substitute for experimenting on the line. Details such as rework, test wafers, and operators were represented explicitly in order to reliably evaluate the impact of detailed scheduling and flow control policies proposed to reduce cycle time.

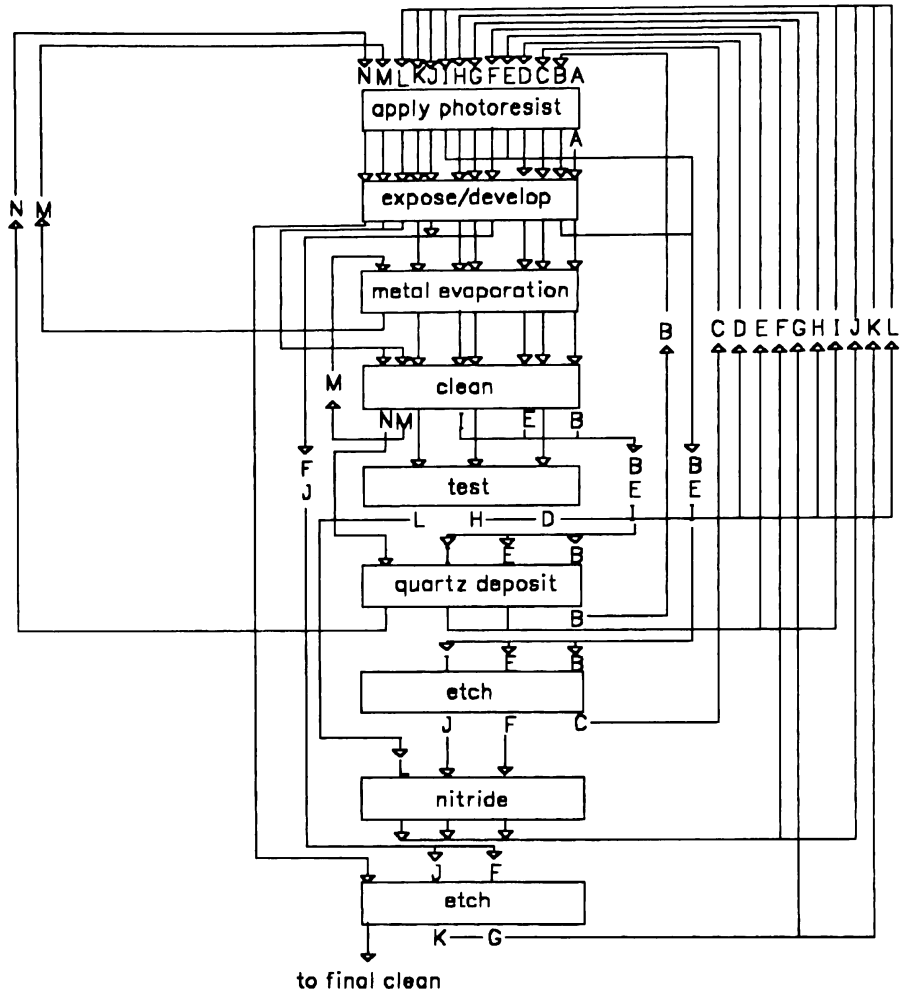

Figure 1. Job Flow in a Semiconductor Manufacturing Line

4. A DETAILED, FLEXIBLE, GENERIC MODEL

In designing the model to accommodate this level of detail, the two seemingly conflicting requirements of flexible and generic also had to be incorporated. A flexible model ensures that changes in the details can be easily integrated into the model and a generic model allows for straightforward application to other semiconductor manufacturing lines.

The discrete event simulation language SIMAN (Pegden, 1987) was used, but the requirements for a detailed, flexible, and generic model led to a model that is structured differently than the conventional model and experiment frames. A customized configuration file serves as the user interface to the model. It contains all the input data describing the line and the information necessary to conduct simulation runs. The model frame uses submodels called station macros to create modular units representing the tool groups and tools. These macros call events to execute FORTRAN subroutines which make determinations such as how to assign operators, which job to process next, and how to route a job needing rework. These FORTRAN subroutines depict detailed aspects of the system and are easily manipulated by turning options on or off in the configuration file. The model also uses FORTRAN user functions which replace the need for many entity attributes, resulting in a flexible structure with reduced memory requirements.

4.1 Configuration File: The User Interface

The standard approach for a SIMAN model requires that the structure and relationships of the system's components be described in the model frame and that the specific variable values be entered in the experiment frame. Changing either of these files requires recompilation and relinking before executing the updated model. However, a different approach was needed to implement the semiconductor manufacturing model. An interface was designed to provide a simpler means of entering the voluminous data and to avoid the recompilation step.

The interface is an ASCII file referred to as the configuration file. The model and experiment frames were built to accommodate an extendible model with many manufacturing options. In the configuration file, the user customizes this generic simulation model by providing information on tool groups, operator pools, control policies, schedules, and other parameters. At the beginning of the simulation run, the configuration file is read in, and the appropriate variables are set. Since this file contains all the system specific information, recompilation and relinking is unnecessary. The user simply modifies the file to represent the changed system (or new system) and executes the simulation model.

Specifically, the configuration file contains sections which represent the following: input control parameters, the input schedule, shift and simulation run (time) parameters, expediting policies and parameters, rework policies and parameters, job selection policies and parameters, product types, standard and rework sequences (routing), tool group parameters (capacity changes, failures, setups, operator ratio), and operator pool parameters (capacity changes, staffing levels).

The configuration file is processed using unformatted read statements (list-directed input). This provides latitude in laying out the configuration file and in entering the data. However, it also mandates considerable error checking to prevent the user from inadvertently running the model with incorrect parameter settings. Several issues which aided in the implementation of the configuration file include the following: an organization of the file which is easily understood by the user (related sections are near each other and infrequently used sections are grouped at the end of the file), complete, concise directions in the file for entering the data with examples provided in some sections, and error checking during the processing of the configuration file. The placement of check points helps the user identify the section in the configuration file in which the error occurs; errors checked for include too little data, too much data, mismatched data types, and an incomplete data file.

4.2 SIMAN Model

The configuration file is integrated with the model in the Initialize event (see Figure 2 for the model layout). The model is customized based on the information entered in the configuration file. The modular, generic model design ensured an easy approach for controlling the numerous options in the configuration file and a simple method for expanding on those options.

The SIMAN station macro provides such a modular design. A station macro is similar to a subroutine in that it is a generic set of logical statements which act on the parameters passed into it. Each macro consists of a range of individual stations which represent a range of related resources such as tool groups or tools within a tool group. In the model, two macros represent the flow of jobs through the manufacturing process. One station macro contains the generic code which applies to all tool groups. The other station macro comprises the generic code for all individual tools within a tool group. Each station in the macros embodies each tool group or tool. Thus, jobs flow from tool group to tool group by looping through the two macros. As the job reenters either macro with its current tool group or tool number, the same model code is traversed with the options pertinent to that particular tool group or tool.

Station macros were also used for three tool-based external events: preventive maintenance, major setups, and failures. Again, in each macro, the same model code applies to all tools and only the values of the variables change to specify the particular tool and its maintenance, setup, and failure parameters. With tools numbering in the thousands, this design greatly reduces the overhead of these events by only activating the control entities for the duration of the events. The external events are independent occurrences which affect the system. Jobs do not flow through these parts of the model; instead, these events affect the system's performance by changing resource availability or job attributes. These events were implemented as separate sections of model code with their own control entities. The time between execution of these events and their durations are specified in the configuration file and may be random. In addition to the three tool-based external events which use macros, there are four external events which did not require the macro structure: initialization of variables, assignment of a roving operator group called the flying squad, expediting jobs, and preloading the queues at the beginning of a simulation run.

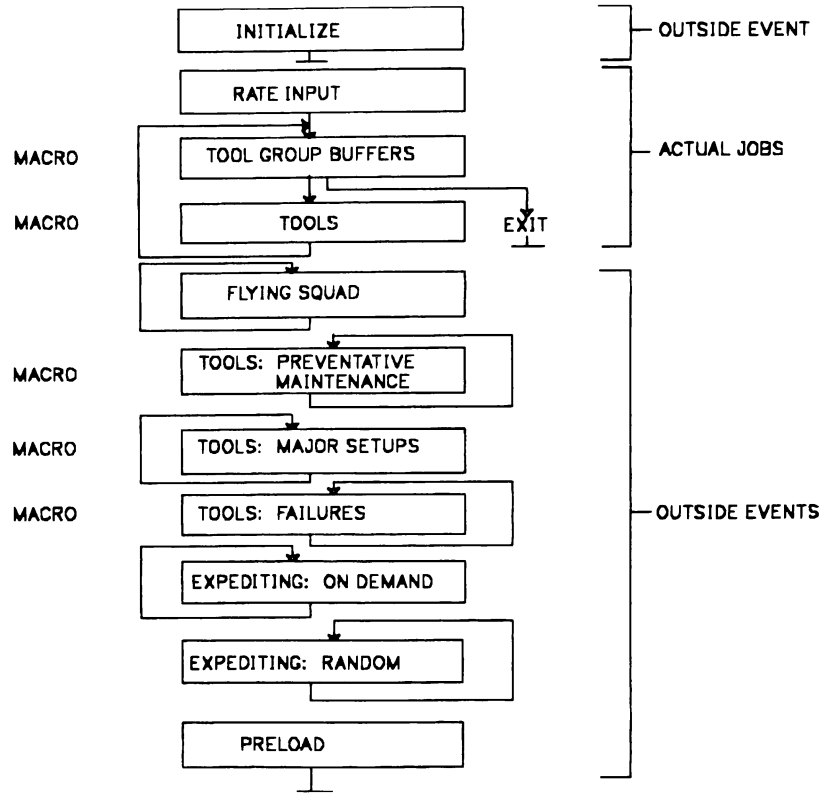

Figure 2. Model Structure

Future enhancements are simplified by the use of the station macros. For instance, each tool (station) in the tool macro may refer to another macro in which several stations relate to the one tool. In this way, separate stations in the new macro may represent multiple processing steps at one tool. The station macro design also allows for the direct integration of material handling schemes between stations (tool group to tool group, or tool to tool group).

4.3 FORTRAN Code

The simulation model interfaces with the FORTRAN code by issuing event calls which indicate the subroutine to execute. An event call may be initiated by a job flowing through a tool group or tool, or by a control entity in an external event loop. Similarly, FORTRAN functions are also called from the model by explicitly referring to the function in the simulation model statements. The FORTRAN code depicts detailed aspects of the input schedule, shift (time) events, operator assignments, resource (tool and operator) capacity changes, expediting policies, rework policies, and job selection rules.

The independent external event loops deal with either input controlled or shift controlled events. The input controlled events are split into either rate or job list. The rate input provides a job creation macro with one control entity for each job type and uses a random number generator to pick job attributes from distributions. Alternatively, the job list provides input from a sequential list of jobs with all attributes explicitly provided. The job list has a daily release schedule pointing into the sequential list of jobs and a cycle schedule pointing into the daily release schedule. A single control entity creates the next X number of jobs from the job list at a time specified in the daily release schedule. The control entity is then placed onto the event calendar until the next specified release time in the daily release schedule. See Job List Input in Figure 3.

The shift controlled events are all called either at the start of every break period, shift, day, or month (no weekly events have so far been implemented). See Shift Logic in Figure 3. The operator pool capacities can be varied throughout the shift to reflect break periods. At the start of each shift, individual operators are assigned to individual tools. This is accomplished by using indices in the resource names and in the queues of the station macros. These indices are examined by the job

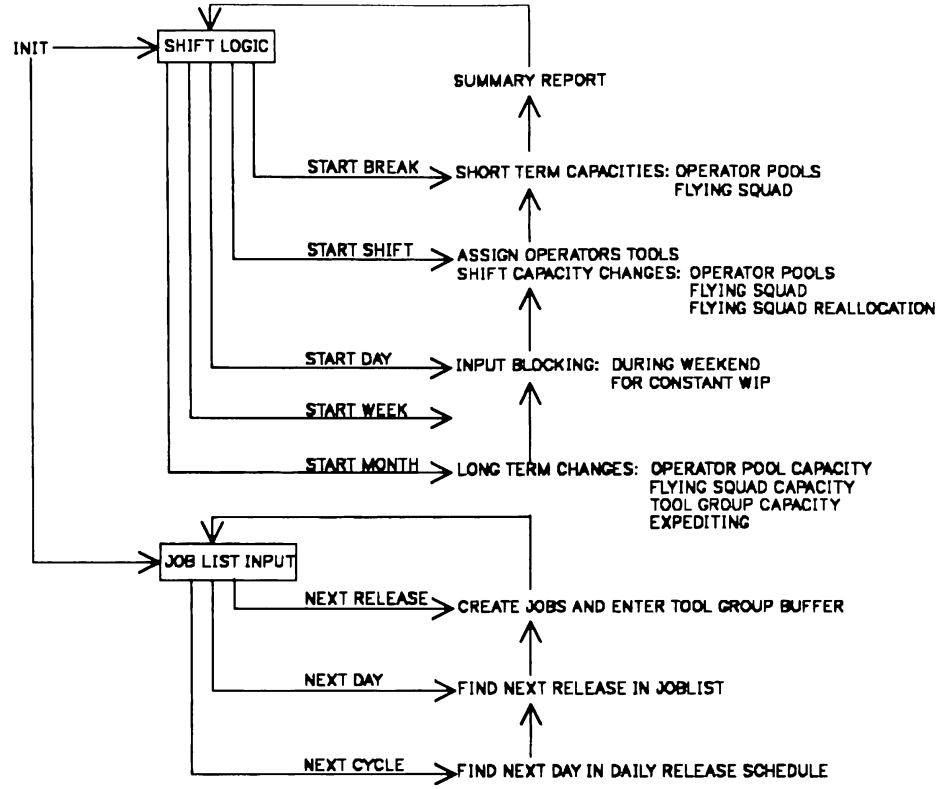

Figure 3. FORTRAN Structure

selection subroutine to ensure that the correct operator from the correct pool serves the corresponding tool. Operators may be assigned to more than one tool, and may perform up to three different tasks at a single tool: load, monitor, and unload. By assigning individual operators to specific tools, the simulation model controls the fact that operators may monitor up to N tools (tool group attribute) but perform load and unload on only one tool at a time. A secondary operator pool called the flying squad is also reassigned at the start of every shift to tool groups (attribute) to help out the operator pools. Long term capacity changes to tool groups, operator pools, and the flying squad are applied on a monthly basis.

The code for expediting jobs describes three policies for altering a job's priority. The policies are referred to as random expediting, end of month expediting, and expediting based on demand. Random expediting reassigns a job's priority at random intervals. The job's priority may be increased or decreased, and the affected job may reside at any stage in its sequence. End of month expediting represents the push to ship product at the end of the accounting month. A few days before the end of the month, those jobs which can be completed by the end of the month are assigned priorities above all other jobs. Expediting based on demand is an event which occurs at regular intervals. This event considers all jobs and compares their current progress with their due dates. Based on these comparisons, the jobs' priorities are either increased or decreased. Each of these three policies may be turned on or off, independently of each other, through the configuration file.

The rework code represents various possible policies on how to split bad wafers out of a job and how to recombine reworked wafers with original good wafers.

The job selection code embodies the various rules which determine which job to process next. Job selection is performed when a tool and its operator are available. The waiting jobs are considered for processing in the following manner: jobs with the highest priority ever possible are selected first, split jobs (the first part of the job has already been processed and the remainder is waiting due to the tool's batch size) are selected next, and then if no job has been found, the job is selected based either on highest priority remaining in the queue or to minimize setup. The user chooses the rule through the configuration file. The selected job may be immediately rejected if its downstream tool group's queue is full or if it cannot be

completely processed before the end of the current shift. Once a job has been found, it is then checked for how it should be processed. It may be a job which requires a test wafer (sending part of the job ahead first, and then sending the remainder through), it may require batching with other jobs, or it may require splitting due to its job size relative to the batching capability of the tool.

The FORTRAN code is split into approximately sixty subroutines, continuing support of the modular theme. For instance, all code relating to job expediting is contained in three subroutines, thus limiting the amount of code one has to deal with when making modifications.

5. MANAGEMENT ISSUES

There are several management issues which were also vital to the successful completion of a project this large and complex. They include development of the functional specification document, management of data, frequent meetings between the manufacturing members and the design members of the modeling team, and preparation of the user's guide.

5.1 Functional Specification

Of primary importance was the development of a functional specification. This document established a common understanding, of the model's objectives, among the team members and provided an outline for planning the project's milestones. The functional specification contained an explicit description of the data required, the output to be provided, and the details to be modeled. The initial objectives were revised based on their importance within the limitation of resources. After the first draft was written, the team members revised, circulated, and rewrote the document. Once this cycle was completed, there was a mutual, comprehensive understanding of what the model would and would not do.

The starting date of any milestones should be based on the completion of the functional specification rather than a specific date. Estimating the development time of the functional specification itself may be the most difficult milestone to ascertain.

5.2 Data Management

Due to the copious amount of data required by this model, the considerations for obtaining and managing the data was important. Design issues were approached with the data requirements in mind. Wherever possible, the input data format for the model mirrored that which already existed for other uses. The output could also have been overwhelming in size. The pertinent performance measures and their exact meanings were defined as part of the functional specification. Defining the model output at the beginning of the project enabled the team members to know exactly what information the model would provide and how to interpret that information.

5.3 Frequent Meetings

Throughout the development of the functional specification and into the design and implementation of the model, the participation of a team was crucial. The primary importance of these meetings was the transfer of expertise between team members. At first, the manufacturing members conveyed their knowledge of the semiconductor process to the design members. Then during implementation, the design members explained their implementation to the manufacturing team members. The team functioned best when all members were able to participate fully and when an agenda of clearly defined problems was constructed. Meetings of two days seemed to be sufficient; maintaining the intensity beyond two days would have been difficult. Those two days of problem solving usually provided an implementation window of three to four weeks, at which time the same process was repeated.

5.4 User's Guide

The user's guide contains sections on how to set up a configuration file, how to run the model, and how to interpret the output. Two other sections address topics concerned with the maintenance of the model structure and the FORTRAN structure. Additionally, it contains an updated version of the functional specification listing the final assumptions and details modeled.

The user's guide is arranged to serve two levels of users' needs. Most users will be concerned only with running the model by using the configuration file as the interface to the model. They will not have to do any programming or compiling. A few users will be involved with the maintenance and enhancement of the

model and will have to deal with the SIMAN and FORTRAN code directly.

6. CONCLUSION

This project demonstrates that simulation models can be developed to represent large, complex systems with great detail and flexibility while maintaining their generic nature. The techniques used to design and implement the model were presented; they were the Configuration File (the user interface), the SIMAN Model, the FORTRAN Code, and the Management Issues.

The goals of the simulation model design were attained. Considerable detail from the manufacturing lines was incorporated in the model; the user has a simple, flexible interface for choosing these detailed options and the generic structure allows for the direct application of this model to other semiconductor manufacturing systems and configurations.

ACKNOWLEDGEMENTS

The authors extend their appreciation and thanks to their fellow team members: Paul Feeney (IBM East Fishkill, NY), Bob Leavitt (IBM T. J. Watson Research Center, Yorktown Heights, NY), and Dave Robideau (IBM Burlington, VT). The authors would also like to thank Manu Patel (Manager, Industrial Engineering, Fishkill) for his support of the project, Randy Sadowski for his help with the project in all its phases, and Barbara Werner and David Amamoto for proofreading this manuscript.

REFERENCES

Pegden, C. Dennis (1987). *Introduction to SIMAN*. Systems Modeling Corporation, 504 Beaver Street, Sewickley, PA 15143.

AUTHORS' BIOGRAPHIES

SARAH JEAN HOOD received her Ph.D. in Mechanical Engineering from the University of California, Davis. She joined the Manufacturing Research department at the IBM Thomas J. Watson Research Center and worked in the continuous system domain on an automated method of modeling electromechanical systems for the purpose of design evaluation. She is now working in the discrete system domain exploring various methods and tools including Petri Nets and discrete event simulation languages for modeling semiconductor manufacturing. She is a member of the Society for Computer Simulation and the Institute of Electrical and Electronics Engineers.

Sarah Jean Hood 2-114
IBM Corporation
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Hts., NY 10598

AMY E. B. AMAMOTO is currently an Engineer performing manufacturing systems analysis for the Simulation Services division of Systems Modeling Corporation. She received her Bachelors Degree in Industrial Engineering from the University of Pittsburgh. She has developed SIMAN and SimKit simulation models used in the evaluation of various electronics and transformer components manufacturing systems. She is a member of the Institute of Electrical and Electronics Engineers and the Institute of Industrial Engineers.

Amy E. B. Amamoto
Systems Modeling Corporation
504 Beaver Street
Sewickley, PA 15143

ANTONIE T. VANDENBERGE is a Senior Project Engineer at Systems Modeling Corporation. He received his Bachelors and Masters Degrees in Industrial Engineering from Purdue University. He joined Systems Modeling Corporation in July 1986, where his responsibilities include performing manufacturing systems analysis and training functions for the Simulation Services division, and product research and development for the Software Development division. His areas of interest include using shop floor information systems to aid in the simulation and scheduling of manufacturing and material handling systems. He is a member of the Institute of Industrial Engineers and the Society of Manufacturing Engineers CASA.

Antonie T. Vandenberge
Systems Modeling Corporation
504 Beaver Street
Sewickley, PA 15143