Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

A DETAILED MODEL FOR A HIGH-MIX LOW-VOLUME ASIC FAB

Mike Gissrau

X-FAB Dresden GmbH & Co. KG
Grenzstrasse 28
D-01109 Dresden, Germany

Oliver Rose

Institute of Applied Computer Science
Dresden University of Technology
D-01187 Dresden, Germany

ABSTRACT

Looking for new improvement options such as new dispatching rules of an existing semiconductor fabrication facility, a detailed model is indispensable to check the data quality as well as detecting main influences of the facility and finally testing the new optimization approaches. In this paper, we describe the whole modeling process starting from the data acquisition to the verification and validation of the resulting model. In this study, the modeling tool AnyLogic 6 is used. The evaluation reveals the importance of a reliable factory database. In addition, we show first ideas about automated model generation. An other important problem is the validation of the model against real factory performance indicators.

1 INTRODUCTION

In a typical ASIC (application-specific integrated circuit) semiconductor facility there are hundreds of different products or product variants processed on dozens of different equipments. In addition, every product has its own manufacturing process consisting of hundreds of different steps. Characteristics like sequence-depended setup times, equipment failures, and preventive maintenance activities as well as product-specific re-entrant flows cause considerable variability in factory performance measures like cycle times and inventory levels. Often such factories have high product mix and low product volume characteristics which will increase the complexity of the factory operations even more. Good scheduling and dispatching strategies for the lots in such a factory environment are hard to develop

To improve the main factory performance key indicators and to accelerate the product flow, a variety of production control techniques are applied (Fowler and Robinson 1995, Wein 1988). Starting with simple dispatching approaches like FIFO, SPT, and ODD (Rose 2001, Rose 2003, Rose 2002) there is a wide range of different approaches for optimizing one or more factory performance measures like cycle time and work in process (WIP).

The main problem of finding an appropriate operational control solution is hard to solve. Because of complexity and variety of the production processes in an ASIC facility, mathematical methods (e.g. by queuing theory as described in (Allen 1990)) for the evaluation of different dispatching and scheduling approaches are still not capable to represent the huge variety of impacts on the factory operations.

A different approach to analyze the whole factory performance is the application of simulation. Different simulation techniques are available like discrete event simulation or agent-based simulation approaches and are used in a lot of application areas (Law and Kelton 2000). Even in the case of simulation developing a fab model and finding the correct parameters for a real factory is a difficult task.

In this paper, we describe the development of a simulation model for a real ASIC factory by means of the simulation tool AnyLogic 6. It is based on the object-oriented programming language Java and offers a high flexibility to model the specific details of this particular factory.

2 MODEL CREATION

In this section, we describe the model creation process starting from the modeling environment and closing with the model itself.

2.1 The Modeling Environment

In the beginning, the selection of the right modeling environment is indispensable. On the market we find a large variety of discrete-event simulation modeling tools. In Figure 1 we depict a short overview of a general modeling tool classification.

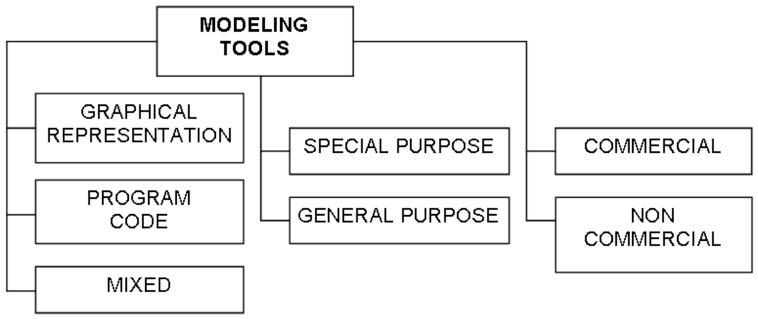


Figure 1: Modeling tool overview.

In general, the tools can be divided into general purpose and special purpose modeling tools. Special purpose tools are available for semiconductor manufacturing for example Factory Explorer®from WWK. Special purpose tools are suitable for problems in their particular domain with low to medium modeling effort, but adaptation to special conditions may be difficult. General purpose tools can be adapted to any domain. Often such tools use interfaces to well-known programming languages (e.g. Java). These tools often provide both a graphical user interface for model creation and a programming interface. Besides this, frameworks are usually available which offer the functionality for model creation and reporting without graphical support.

Creating a detailed model of a factory requires a modeling tool which can be customized to different problem scenarios of a real factory. In addition, it should provide access to the whole model creation and runtime environment to facilitate the introduction of a particular model behavior like complex dispatching regimes. In our case, this is the most important aspect because the company intends to develop and test a new Manufacturing Execution System (MES) on the basis of this simulation model.

Because of these requirements, we chose AnyLogic 6, a Java based modeling environment which can be customized on programming level. Besides that a fast model creation is possible by means of user-defined modeling objects built from objects of the tool's own libraries.

2.2 Main Simulation Objects

Looking at a real ASIC factory an enormous number of influences and entities can be found. The first task of the modeler is to select the main entities in the facility and their interactions. In Figure 2 we give an overview of important entities in a semiconductor facility and the interactions between them. Developing model components for these entities is the first step for modeling a complex system. For our figures we use a simplified UML notation (Stoerle 2007).

The entities shown in Figure 2 allow us to divide the complex system of a semiconductor facility into smaller objects for better understanding:

- **Product and Process:** Each product lot follows a particular process routing in the facility until it is finished. Each process element is described as a step. A process can have several hundred steps.

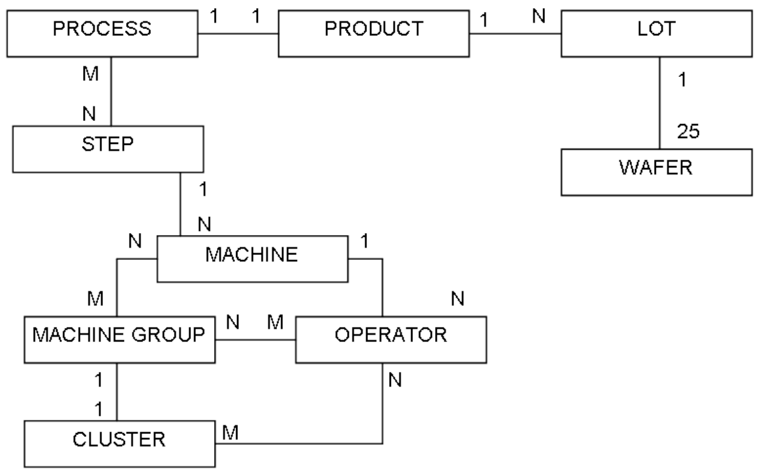


Figure 2: Model entities and their relations.

- **Step:** A step is an unique description of one part of the process flow. Each step requires one or several machines. The lot can be processed by the type of operator which is required for this step. In addition, each step has a mean processing time and a mean operator time.
- **Lot and Wafer:** Each lot has at maximum 25 Wafers. Every Lot corresponds to one product (or product type).
- **Machine:** A machine is a unique processing station which is able to process one or more lots at a time (batching or non-batching). Each machine can have different setup states corresponding to the different lots being processed on it. An example is the different dopant usage on implanter equipment. Each machine has different states like “down”, “maintenance”, “idle”, and “working”. The “down” state is caused by an unplanned event where the machine is not able to process lots, e.g. a part of the machine has a unexpected failure. Maintenance activities are usually scheduled by a planning tool to maintain the machine and prevent unexpected machine downs (Rose 2004).
- **Machine Group and Cluster:** Every machine can be assigned to a machine group but not all machines of a group may be eligible for a particular step. Note that machines from different groups may be eligible for a step. Each machine group can be assigned to a logical cluster, e.g. metalization or lithography.
- **Operator:** The most difficult modeling part of a non-fully automated factory is modeling the operator behavior. Each operator is usually assigned to one cluster and there to a certain group of machines. In addition, non-standard behavior like switching between clusters and machine groups is also possible.

2.3 Data Acquisition

The model data acquisition is another important part of the model creation process. The data sources for the model can be divided in two main categories:

- **Informal Data:** Informal data is often unpublished, i.e. only operators or process engineers can give details about system properties like dispatching practices. This data has to be added manually to the model. This data often has a considerable vagueness.
- **Formal Data:** Formal data is generated by the wide range of data collection and storage systems available in a semiconductor facility. Often a large part of data needed for a model can be automatically collected from the factory data base, e.g. from the MES. For automated model data creation, a data test software system must be implemented to ensure the validity and consistency of the data.

For data acquisition, we built a model data generator to simplify the model data collection process. It is not possible to deal with such large amount of data from the factory database manually. In Figure 3 there is an overview of the data acquisition process.

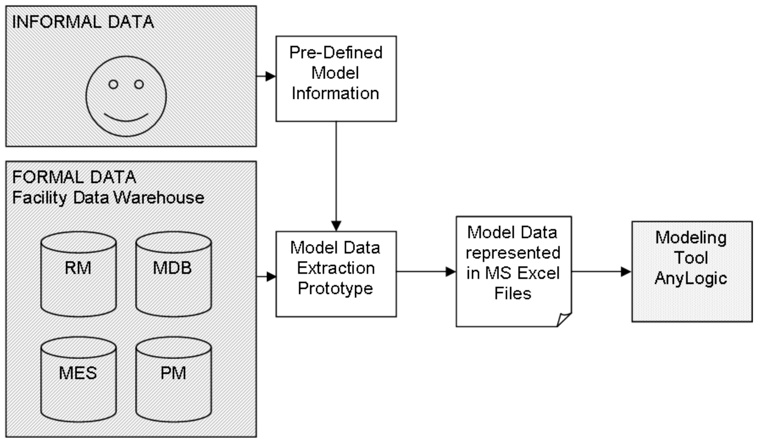
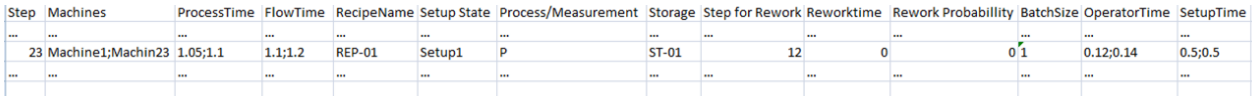

INFORMAL DATA
Pre-Defined Model Information
FORMAL DATA Facility Data Warehouse
RM
MDB
MES
PM
Model Data Extraction Prototype
Model Data represented in MS Excel Files
Modeling Tool AnyLogic

Figure 3: Model data acquisition process.

As mentioned above, there are two main data sources: the facility data warehouse and the so-called insider knowledge from, for instance, process engineers or operators. In the considered facility the data warehouse consists of the Manufacturing Execution System (MES), the Recipe Management (RM), the Preventive Maintenance management (PM), and further data saved in a Mass DataBase (MDB). These data sources were used by the model data acquisition prototype to create model data represented as MS Excel spreadsheets.

The generation process creates three excel files corresponding to the required entities. First, the processes for all product lots are created. The file contains all steps needed for the products as well as the corresponding machines, process times and setup states. The data originates mainly from the RM and from discussions with process engineers. A second MS Excel file is generated to represent the machine data, including maintenance data and unexpected down data to facilitate the generation of down time distributions for each machine. Finally, a file containing the lot start plan is created. This data is mainly extracted from the MES. Figure 4 shows a snippet from one of the created Excel file.



Step	Machines	ProcessTime	FlowTime	RecipeName	Setup State	Process/Measurement	Storage	Step for Rework	Reworktime	Rework Probabillity	BatchSize	OperatorTime	SetupTime
...	...	...	...	...	...	...	...	...	...	...	...	...	...
23	Machine1;Machin23	1.05;1.1	1.1;1.2	REP-01	Setup1	P	ST-01	12	0	0	1	0.12;0.14	0.5;0.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 4: Example row in machine Excel file.

The data quality of the data warehouse is still an issue. As a consequence the extraction prototype has several functions for testing the data, in particular with respect to logical problems. For example, crashes of some machines do not always lead to valid downtime signals. Therefore some downtime periods seem to last much longer than they actually are. It is a big challenge to find out such data inconsistencies in the factory data base but it is also a very important to run the model with valid data proven by an expert user.

Besides the off-line collection of model data there is also the possibility to generate data for a single simulation run directly from the factory data base at runtime. For larger simulation studies, this approach is not viable because the data amount is relatively high and thus each simulation run will need a large time amount for the generation of the data compared to using the MS Excel files in a fast and reliable way for different simulation scenarios.

2.4 Factory Model Creation

In the following, we present the simulation model generation environment. Because AnyLogic is a general purpose modeling tool, a lot of extensions have to be included in the model to match the needs of a high-fidelity factory simulation. In the next sections, the most important modeling components are explained.

2.4.1 The Equipment Representation

The most important and complex part of the model is the representation of processing equipment. In this model component, different characteristics like setups, lot processing, batching, etc. have to be included. In AnyLogic, these components consist of graphical descriptions and program code. An example is shown in Figure 5.

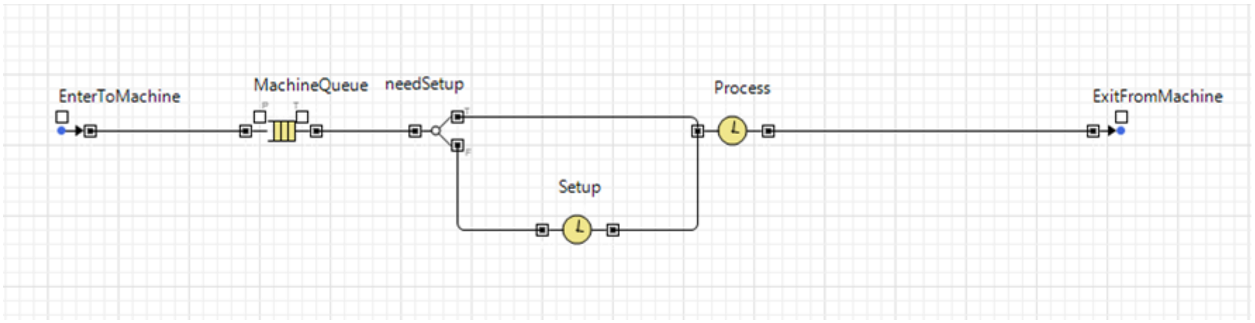


Figure 5: Example for a simple machine model.

In this example, a simple machine behavior is presented. The model contains predefined Java classes provided via the AnyLogic library, extended by user source code in additional Java classes plus existing library classes. The library objects used in this example are a queue object, an output selector object and two delay objects.

Each lot (the class lot is a user defined class) is moved to the machine defined by the current step and then it is stored in the queue of this machine. With FIFO dispatching (the behavior can be changed in the code) the first lot of the queue is released if the process element and the setup element is empty. If a setup element is defined by the step, the current setup state will be compared with the setup state required by the processing step of the lot. If there is a mismatch, the setup branch will be executed, otherwise the lot will be sent directly to processing. After processing, a user defined selector delivers the lot to one of the downstream machines. For our fab model creation this simple model is extended with a lot of additional characteristics like operator interaction, batching rules and rework.

Furthermore the failure and maintenance behavior has to be included. This is realized by a state model (Figure 6).

The states are triggered by different events like down events and maintenance events as well as process-dependent events like setup and processing or waiting on operators. Down events are generated in two ways: planned maintenance downs after fixed time intervals (currently determined manually because of insufficient data quality) as well as failures according to random processes (based on histograms for the actual time off-line (TO) and the actual time between off-line (TBO) of this machine). An example for the TO case is shown in Figure 7 .

In the case of our ASIC factory this approach works better than assuming independent exponentially distributed periods of downs which are often used to model time off-line and time between failures.

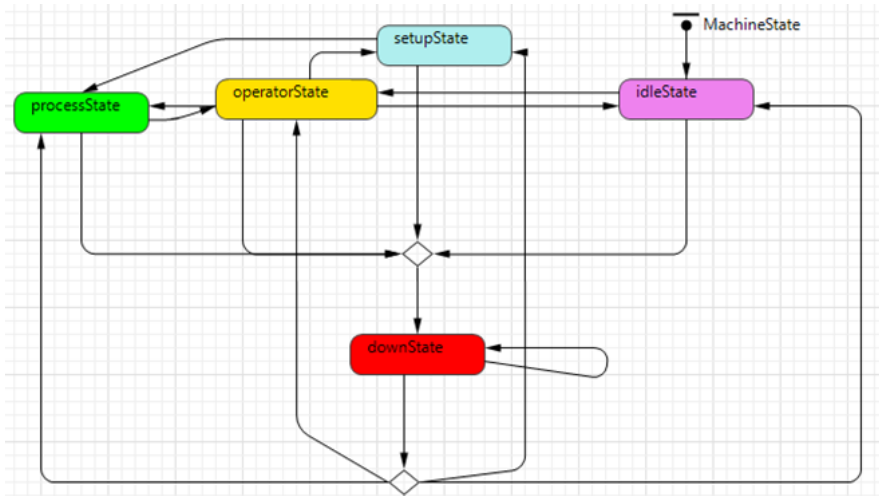


Figure 6: State model of the machine class.

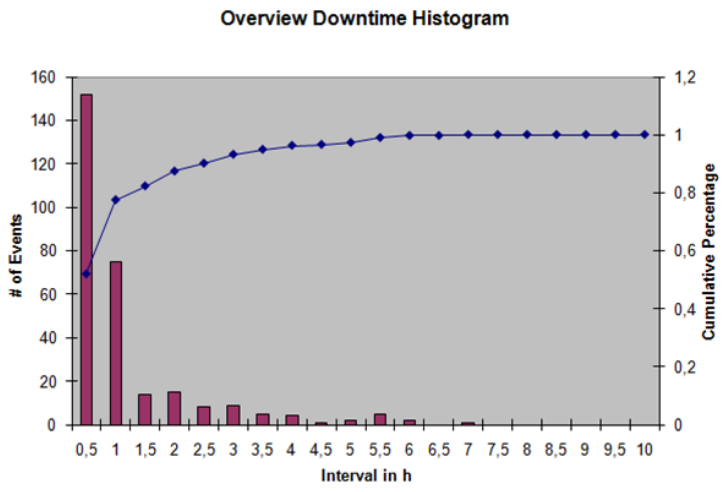


Figure 7: Down time histogram of a machine for a given interval.

2.4.2 The Operator Representation

For modeling the operator behavior a simple approach is used because of the relatively sparse information about operators in the database. Each operator is defined as a resource needed by a machine for processing. Without the operator the processing cannot be started and the lot will have to wait. Each machine corresponds to a certain operator group. In future versions of the fab model the training state and skill level of each operator will be considered to avoid model inaccuracies. In some cases an operator is only allowed to work at certain machines or machine groups after training sessions at these particular machines.

Each operator has a certain break time behavior in their working shift. The breaks do not happen in a random manner but rather in relatively fixed scheme of time phases (Figure 8).

The scheme is repeated for every twelve hour work shift and shows the number of operators of a certain group. In this case the operators pause for an hour twice a shift.

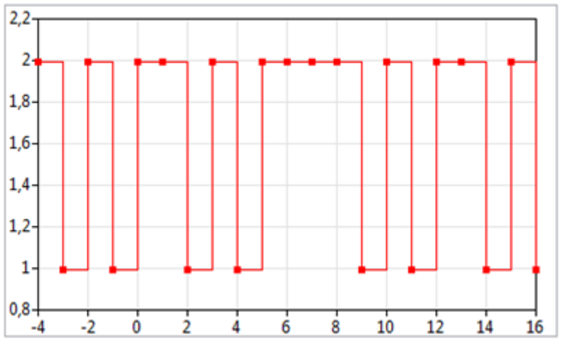


Figure 8: Example for break behavior of an operator group.

2.4.3 The Factory Representation

Several other entities are also integrated into the model. The resulting class diagram corresponds to the entity diagram shown in Figure 7. The whole model contains several hundred machines and about thirty operator groups. Every machine is placed manually to a scaled layout to facilitate the representation the routes of each lot and to visualize the model behavior. Figure 9 shows a part of the generated layout (machines - light blue, storage places - orange).

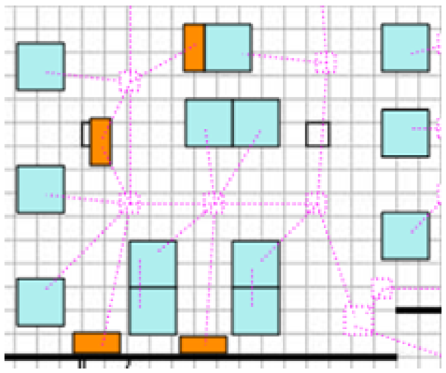


Figure 9: Part of the layout of the facility.

As mentioned in the equipment section, two important characteristics are integrated:

- **Rework:** A certain rework probability is assumed for each step. Rework is often caused after measurement steps. After a certain rework time the lot can be re-inserted to the normal process at a certain step number.
- **Lot Stop (hold lots):** In our fab, a lot is stopped relatively often for additional measurement steps to verify certain characteristics. This is modeled assuming exponentially distributed lot stop times and times between stops (similar to random machine failures).
- **Transport of lots:** The transport of lots in the model is also be done by the operators. When an operator is going to process a certain equipment, the corresponding lots will be transported to the equipment from the storage. Between the clusters not each lot will be transported individually. There are transport vehicles which must be filled (equal to batching) with a certain number of lots. Then the transport will happen. Also if a pre-defined time is expired, the transport starts even the batch size is not reached.

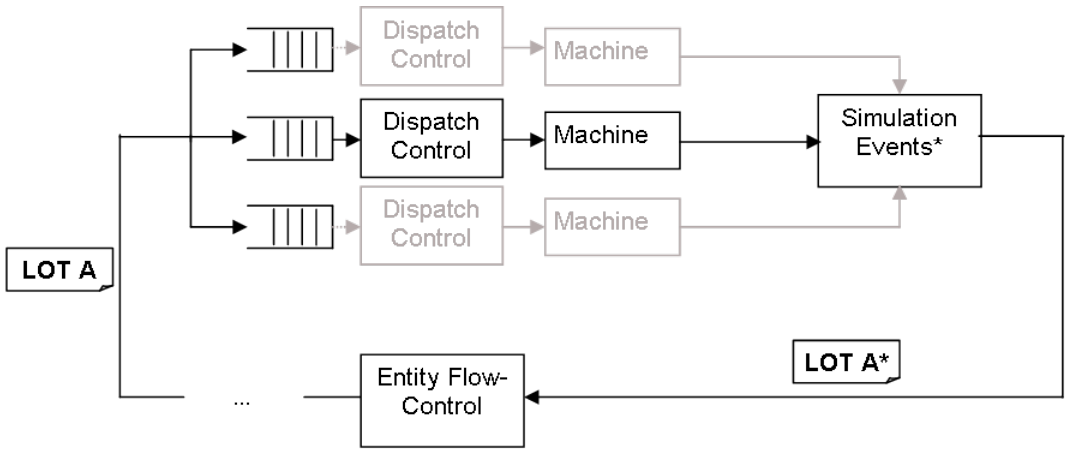


Figure 10: Flow of the entity lot through the model.

With the components mentioned above it is possible to model the general production process of a lot shown in Figure 10. If a lot is created by the EntityFlowController due to a lot generation event, it is processed in the factory model according to its process routing. At each step the entity is duplicated and assigned to each queue of each machine which is able to process the lot at the current step. The implemented dispatch control system which is responsible for choosing the best lot or batch according to the dispatching discipline, setup state or batch regime selects the next set of lots to be processed when the machine is ready (e.g. machine Y). The lot, e.g. LOT A, is removed from the other queues if a machine starts to process the current set of lots including entity LOT A. After processing a step, several potential events are evaluated like lot stops and rework demands. Then the entity (LOT A*) is moved back to the EntityFlowController which prepares the next actions and moves the lot to the next step or finishes the lot processing. Then a number of performance indicators are computed (cf. Section 2.5).

2.5 Result Processing

For the description of the output parameters P_i the following notation is used:

- RPT: raw process time
- PT: process time
- FF_F: flow factor of factory
- FF_p: flow factor of product p
- FF_m: flow factor of machine m
- CT_F: cycle time of factory
- CT_p: cycle time of product p
- $CT_{F_{ML}}$: cycle time per mask of factory
- $CT_{p_{ML}}$: cycle time per mask of product p
- WIP_F: number of all lots in factory (WIP)
- WIP_p: number of all lots of product p in factory (WIP)
- D_F: delay of lots in factory
- TW_m: waiting time of machine m
- QL_m: queue length of machine m

Each output parameter P_i represents a collection of statistical data including:

- $P_{i_{max}}$ - maximum of P_i
- $P_{i_{min}}$ - minimum of P_i

- $P_{i_{mean}}$ - mean of P_i
- $P_{i_{deviation}}$ - deviation of P_i

The output parameters P_i are generated for each model run and saved to several MS Excel files for reporting:

- **Factory Level:**
  - $P_{FF} = \frac{RPT}{PT}$
  - P_{CT_F}
  - $P_{CT_{ML_F}} = \frac{CT_F}{ML}$ with ML - mask layer
  - P_{WIP_F}
  - P_{D_F}
  - $P_{D_F}(weekly) = \{\frac{\sum P_{D_{F_i}}}{P_{D_F}} : w_u \leq P_{D_{F_i}} \leq w_o\}$ with w_u lower weekly border, w_o upper weekly border
- **Product Level:** the same parameters P_i measured on factory level corresponding to product level
- **Machine Level:**
  - Percentage PC_j of idle, process, down and setup
  - QL_m
  - FF_m
  - TW_m

The implementation of further output parameters is straightforward. The parameters shown here are the most important factory performance indicators and they are currently used for model verification and validation.

3 MODEL VALIDATION

Model validation is not an easy job in our case. All model parameters P_i have to match the real factory's parameters Q_i with some fault tolerance. First of all the definition of an acceptable fault tolerance is very important. This tolerance depends both on the modeling goal and on the data quality for estimating the model parameters.

Often the factory MES provides some reporting functionality which could be used to test the model output. Before this can be done, however, the calculation of the performance indicators has to be the same for the model and for the real fab. In some cases it might happen that performance indicators are defined differently.

In our case, we assume that all the parameters mentioned in Section 2.5 have to be validated. For the first versions of the model, verification and validation have to be performed manually. Each parameter P_i has to satisfy the validation function:

$$VAL(Q_i,P_i) = \begin{cases} true, & \text{if } Q_i - T_i \leq P_i \leq Q_i + T_i \text{ with } T_i \text{ tolerance of parameter i} \\ false, & \text{otherwise} \end{cases}$$

The validation function is implemented in the resulting MS Excel files and checks whether each parameter P_i is within the tolerance interval T_i of Q_i.

In Figure 11 we present an example for testing a parameter P_i for each available product against its tolerance level Q_{i_o} and Q_{i_u}. The violation of the tolerance is marked with a red circle. In our case we take mean, standard deviation, minimum and maximum values into account for validation.

4 CONCLUSION

In this paper, we consider the model creation process of an real ASIC factory from data acquisition to model verification and validation. We show how we developed the main components of the model. These

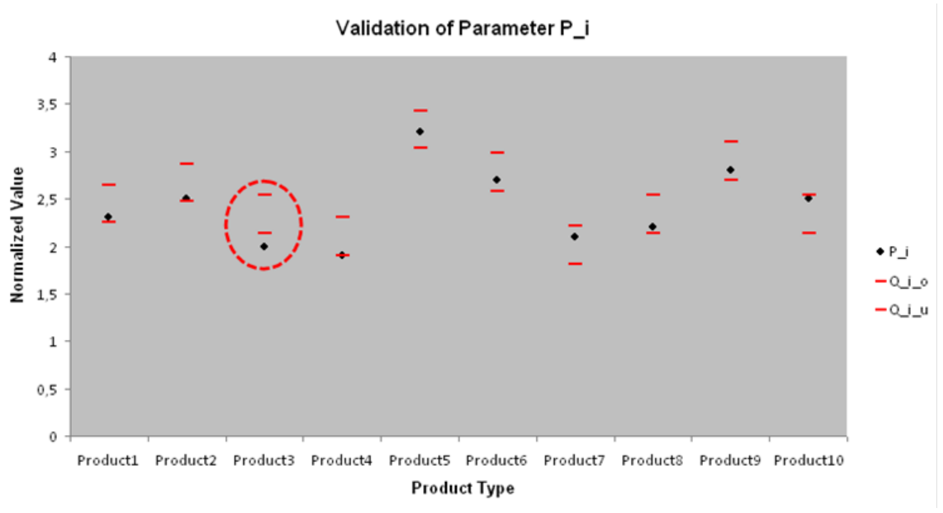


Figure 11: Verification example of one model parameter.

building blocks of a factory model can be described both as classes in an object-oriented programming language or in a modeling tool. In our case, we use a combination of library components of the general purpose modeling tool AnyLogic 6 and additional Java classes. After validation, we obtain a first version model which is appropriate to test several new approaches for improving line performance and stability.

The model is the cornerstone for further research activities like integrating line balancing techniques and testing new dispatching and scheduling approaches. After finding a appropriate dispatching and scheduling approach with the usage of the model, it will be implemented in the MES for daily usage. May also a model dependent dispatching and scheduling strategy will be taken into account.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Matthias Haenisch and Michael Zadlo at X-Fab Dresden for their valuable and fruitful discussions and ideas.

REFERENCES

Allen, A. O. 1990. *Probability, statistics, and queueing theory : with computer science applications*. 2nd ed. Acad. Press.

Fowler, J., and J. Robinson. 1995. “Measurement and improvement of manufacturing capacities (mimac): Final report”. Technical Report 95062861A-TR, SEMATECH, Austin.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling & Analysis*. 3rd ed. New York: McGraw-Hill, Inc.

Rose, O. 2001, December. “The Shortest Processing Time First Dispatch Rule and some Variants in Semiconductor Manufacturing”. In *Proceedings of the 2001 Winter Simulation Conference*, edited by B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 1220–1224. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rose, O. 2002, December. “Some Issues of the Critical Ratio Dispatch Rule in Semiconductor Manufacturing”. In *Proceedings of the 2002 Winter Simulation Conference*, edited by E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, 1401–1405. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rose, O. 2003, December. “Accelerating Products under Due Date Oriented Dispatching Rules in Semiconductor Manufacturing”. In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick,

P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1346 – 1350. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rose, O. 2004, December. “Modeling Tool Failures in Semiconductor Fab Simulation”. In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1910 – 1914. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Stoerle, H. 2007. *UML 2 für Studenten (UML 2 for Students)*. Pearson.

Wein, L. M. 1988. “Scheduling semiconductor wafer fabrication”. In *IEEE Transactions on Semiconductor Manufacturing*, 115 – 130.

AUTHOR BIOGRAPHIES

OLIVER ROSE holds the Chair for Modeling and Simulation at the Institute of Applied Computer Science of the Dresden University of Technology, Germany. He received an M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI, and General Chair of WSC 2012. His web address is www.simulation-dresden.com.

MIKE GISSRAU is a PhD student at Dresden University of Technology. He is member of the X-FAB Dresden Semiconductor Facility. He received his M.S. degree in Computational Engineering from Dresden University of Technology. His research interests include different dispatching concepts and their realization in factory environment of complex production facilities. His e-mail address is Mike.Gissrau@xfab.com.