Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

A NEW METHOD TO DETERMINE THE TOOL COUNT OF A SEMICONDUCTOR FACTORY USING FABSIM

Holger Vogt

Fraunhofer IMS
Finkenstr. 61
47057 Duisburg, GERMANY

ABSTRACT

Tool count optimization is mandatory for an efficiently organized semiconductor factory. This paper describes an efficient heuristic to determine the tool count using the compact fab simulator FabSim Interactive. A combination of the Simulated Annealing algorithm and the knowledge of toolset usage, which is gained by repeated simulation of the factory, results in a fast approach. There are no restrictions concerning multiple products and processes during optimization. A simple cost model (revenue per wafer out minus tool depreciation) yields the objective function to be maximized, tool count values per toolset are the decision variables, and a lot start sequence determines the fab throughput required. Depending on the factory size, optimization results may be available within a few hours of simulation time on a standard PC.

1 INTRODUCTION

Today's semiconductor factories are large enterprises dedicated to a single product family or allowing multiple process flows. Wafer starts of one thousand per day are not uncommon. There are also smaller size factories which often allow a broad process mix at lower volume. In any case a huge investment into production tools is required, being larger than one billion US dollars in some cases.

Thus we need to model the complete factory and simulate it's operation. If then the fab is model available, it will be very attractive to apply it to optimize the size of the factory. Given a large process mix, the task seems to be tedious. Numerical optimization of a factory, especially it's tool count, has not been reported so far.

2 FABSIM

FabSim Interactive (Vogt 2003) is a compact fab simulator contained in a single C++ Windows dynamic link library. The dll is controlled by a supervisor program written in Borland Delphi. The simulator structure is depicted in Figure 1. The first function called by the supervisor is *Init*, which reads toolset data from a file, generates toolset objects from a toolset class (one per toolset), tool (machine) objects from a machine class and sets all start values. Function *MachineCount* is used to change tool count values (as a result of the optimization algorithm), function *Sim* starts the simulation run, after simulation function *GetOutWafCount* returns simulation data to the supervisor. During the simulation FabSim.dll continuously sends data to the supervisor via the SendMessage procedure. Currently 24 functions are available for interactive simulation control.

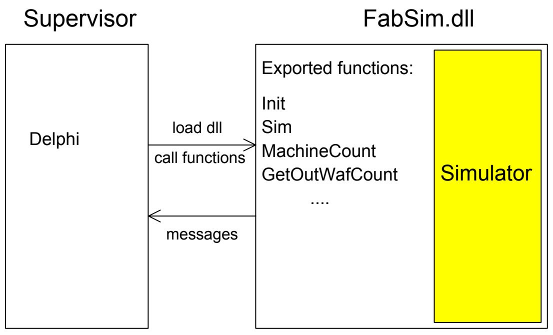

Figure 1: Structure of FabSim Interactive

FabSim, in contrast to most of the general purpose simulation programs, applies "fixed-increment time advance" as a discrete-event simulation model (Law and Kelton 2000). Simulation proceeds in user definable fixed time steps, e.g. 1 minute. During each time step all state changes are recalculated, triggered by an event happening. Events may be loading or unloading wafer lots from a tool, start of a new lot or release of a lot from the fab. Machines change state, e.g. decrement the time counter if a process runs or go from IDLE to PROCEES, from PROCESS to DOWN etc. If no state change is required, the respective function call returns immediately. Wafer lots are exchanged between the machines as objects derived from a lot class. A scheduler distributes all lots to the toolsets as prescribed by the process flow chart.

Borrowed from SystemC (Grötker et al. 2002) is the way to exchange data between all objects. Only when all calculations for the current time step inside all objects are finished, quasi in an infinitesimal short time period, immediately before the next one minute time step commences, all data are exchanged. Thus all calculations are always based on the data available at the beginning of the time step. This method relives the simulator from deciding which event to process first. The approach resembles an electronic system which runs synchronized by a steady clock.

The setup of the fab is entered via a machine tool list which contains process step and machine data of up to 128 toolsets. Processes are described by flow charts which apply the process steps offered by available machines. Alternative toolsets and processing time constraints as well as send ahead wafers are defined. Simulation is controlled by a lot list, which defines lots with start time, process name, priority and other data. As output FabSim offers several tables including lots processed, machine usage, and buffer occupancy.

The fab model includes various advanced features and options: batch operation, three priority levels (including hot lot capability with machine reservation), fixed or throughput dependent or statistically distributed downtime, fixed or lot size dependent process time, transport time (fixed per toolset or loaded from a from-to matrix), WIP controlled or time based lot starts, unlimited number of different flow charts, dispatching rules like operation due date, critical ratio or shortest processing time first. The optional machine setup includes setup avoidance policies. Lot or wafer yield may be taken into account. A factory cost estimation scheme is included.

3 SIMULATED ANNEALING

The semiconductor fab model cannot be evaluated exactly. Firstly the re-entrant process flow does not allow an analytical solution. Secondly the fab model contains "noise". Machine downtime, transport time, operator availability and even process time induce stochastic variations. Therefore solutions to the tool count optimization problem have to be estimated by simulation.

We try to solve this discrete stochastic optimization problem with an algorithm derived from simulated annealing (Andradóttir 1998). The general optimization goal is to find the global maximum of the objective function $f(\theta_n)$, where θ_n is the vector of the decision variables before the n^{th} iteration. The basic idea behind simulated annealing is to allow down-climbing moves so that the algorithm can escape from local solutions. A new vector θ'_n will be generated as a neighbor to θ_n. If θ_n. is a vector of integers, this may happen by randomly adding -1, or 1 to each element (Alrefaei and Andradóttir 1999). If now $f(\theta'_n)$ is a new candidate solution [θ'_n has been be generated as a neighbor to θ_n], and $f(\theta'_n) > f(\theta_n)$, that is $f(\theta'_n)$ is a better alternative, then $\theta_{n+1} = \theta'_n$. If however $f(\theta'_n) < f(\theta_n)$, the algorithm will stay at the better alternative θ_n with a probability $1 - \exp[(f(\theta'_n) - f(\theta_n))/T_n]$, where $T_n > 0$ and will move downhill to the worse alternative θ'_n with the remaining probability. T_n may either slowly decrease according to a "cooling schedule" [$T_n \geq C/\log(n + 1)$, $C =$ const.] or stay constant [$T_n = T > 0$ for all n].

4 OPTIMIZATION SETUP IN FABSIM

Whereas simulated annealing alone is capable to find an estimated global optimum, it will not be useful in the current context of tool count estimation in a semiconductor factory. The main reason is the relatively long computing time required to generate a single data point $f(\theta_n)$. Even if FabSim may evaluate a factory cycle of one year within a few minutes, the total amount of data points needed in a pure stochastic search is prohibitive. Each toolset contributes one decision variable (number of tools), and there may be more than 50 toolsets. Even if we set upper and lower bounds to each variable or fix some tool count numbers, the search space is still large.

The solution to this challenge is to add further information available from the factory simulation results to make the decision if to move on with the candidate solution.

The objective function $f(\theta_n)$ in the current setup is kept very simple: The fab profit has to be maximized. It is the revenue of all good wafers out of the fab minus the depreciation cost of all toolsets for the given simulation period.

Among other output information FabSim delivers a list of toolset and machine utilization data. An excerpt of this list is shown in Table 1. The columns "per machine usage" list the utilization of each machine in a toolset in decreasing order from left to right.

Table 1: Toolset Utilization Output

Buffer and machine usage						
toolset number	mean wait	mean occu-pancy	average usage[%]	per machine usage[%]		
0	8	1.8	72	79	77	73 ...
1	9	2.1	72	83	81	77 ...
2	6	1.3	76	90	88	87 ...
3	173	6.3	53	67	61	49 ...
4	243	5.0	50	56	43	
5	201	5.0	43	56	45	27
...						
12	1982	6.8	69	77	69	62
...						

These data are used to aid the optimization. Starting with a given amount of machines in each toolset a first simulation run is done, calculating the factory output for a preset time period and evaluating the toolset utilization. Utilization is defined as the percentage of wafer processing time (excluding setup and idle) versus the total time. Next the objective function $f(\theta_I)$ is evaluated. The algorithm then checks the machine utilization by scanning through all toolsets. If a toolset is found with heavy underutilized machines (by looking at the rightmost machine in columns “per machine usage”), the least one with e.g. under 25% utilization is removed (if there are still more than 2 machines in the toolset, a machine with less than 40% utilization will be taken out). For example in Table 1, toolset number 5, the machine with 27% utilization would be removed. If a toolset with heavily used machines is found, e.g. the utilization lies above 65%, a new machine will be added. The utilization percentage values 25%, 40%, 65% are derived empirically (see below).

$f(\theta_1)$ now becomes the current solution for SA. The second simulation run with the refined toolset count commences. As it’s result the candidate solution $f(\theta'_1)$ is determined. If the function value is larger than the current $f(\theta_1)$, the new decision variable vector is set according to $\theta_2 = \theta'_1$. If the new objective function value is less than $f(\theta_1)$, the simulated annealing algorithm decides if θ_2 remains θ_1 or becomes θ'_1.

The procedure of reevaluating the toolset count, running the factory simulation and performing a SA decision is repeated for a predetermined number of iterations. After each cycle the toolset count is stored along with the objective function value and some other factory data.

During initialization FabSim reads in data from a file setting upper and lower limits to the decision variables, a start vector and the investment cost per machine. Further inputs are a list of all toolsets with proper toolset description, the process flow charts and a lot start sequence list. All toolsets are run under the dispatching rule ODD (operation due date) (Rose 2003), which sorts lots in each toolset buffer according to their local tardiness. ODD typically offers the smoothest factory operation.

The optimization routine is coded into the supervisor program, using Delphi’s Pascal statements. The simulator FabSim.dll itself remains untouched. Objective function evaluation and vector generation is very fast, it’s runtime is negligible compared to the simulation run.

5 SIMULATION RESULTS

As a first example we have optimized the tool count of a fab with 360 wafer starts per day using two different process flows (CMOS 1.2 and 0.6 µm). Figure 2 compares the results of a pure simulated annealing approach and the new algorithm based on tool utilization. Each iteration (input is a vector with the number of machines per toolset) results from a fab simulation of 500,000 minutes (approximately 1 year), where data are recorded staring after 200,000 minutes. The simulated annealing curve barely shows any progress. The number of 60 iterations is much too small against the vector of 41 decision variables (toolsets) using SA. The quick convergence of the new utilization algorithm however is obvious. After approximately 25 iterations the value of the objective function saturates.

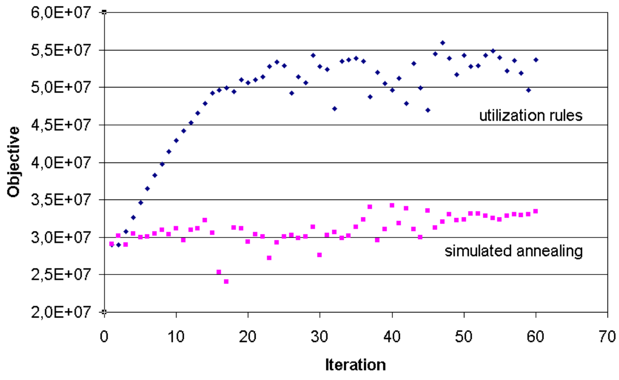

Figure 2: Tool Count Optimization: Convergence of New Utilization Rules Compared to Simulated Annealing

The utilization percentage for adding or removing tools has been evaluated using sample simulations. There is a correspondence to the maximum utilization for a given toolset. Fortunately the spectrum of percentage values yielding good convergence is relatively broad. Figure 3 gives a comparison of several percentage triples. std denotes 25%, 40%, 65%. 20 stands for 20, 40, 65%, 30 for: 30, 40, 65%, 35 for: 35, 40, 65, 70 for: 25, 40, 70%, 75 for: 25, 40, 75% and 80 for: 25, 40, 80%. A fab with 0.8 µm automotive CMOS with 10,000 wafer starts per month is chosen. Simulated time is 200,000 minutes, the final 100,000 minutes are used for data evaluation. The mean simulation time is 90 seconds per data point.

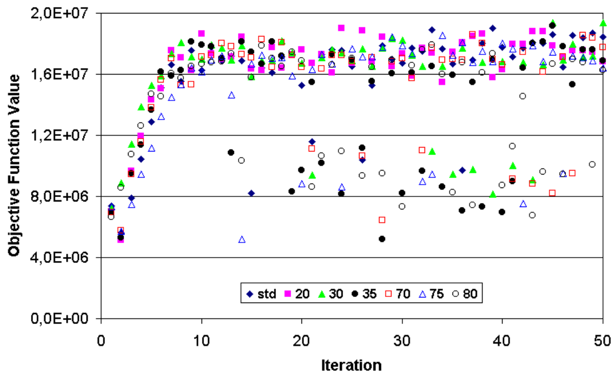

Figure 3: Evaluation of Percentage Triples

All percentage selections lead to convergence after more than 10 iterations. There is however a certain noise in the data for certain selections during further iteration steps. 20% or 25% removal threshold and 65% adding threshold yield the least spread in objective values after convergence is achieved. Other toolset combinations (other factories) behave similarly. More research however is necessary to establish a thorough understanding of the correlations between toolset mix, maximum possible utilization and optimum percentage for toolset count evaluation.

Figure 4 devises further improvement of the optimization algorithm. Limiting the Wip and it’s standard deviation improves factory performance. Data may be selected according to the additional rules as plotted in Figure 4. Manually selected points (with EXCEL AutoFilter) with *Wip* < 400, $StdDev_{Wip}$ < 10 are marked with a black arrow. It would also be simple to add the rules to the optimization algorithm itself as an additional condition to determine θ_{n+1}.

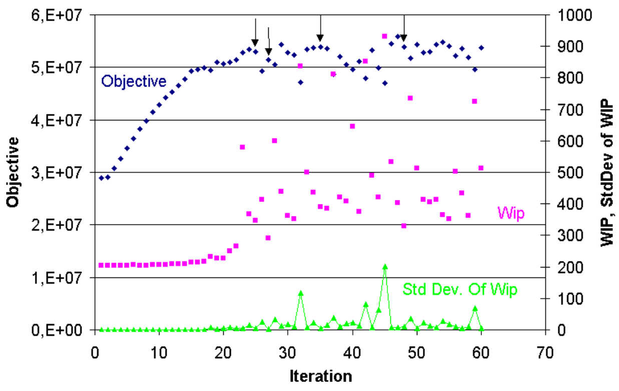

Figure 4: Objective Function, Wip and it's Standard Deviation with Optimum Points Selected Manually

Another example factory toolset optimization is plotted in Figure 5. This factory runs a 0.8 µm CMOS process with 720 wafer starts per day. Simulated and data recording times are the same as in the first example Plotted against the number of iterations is the resulting objective function. In addition you see in parallel the work in progress (Wip). Within a very short period (less than 40 iterations) the objective function converges. The 40 iterations require a CPU time of less than 2 hours.

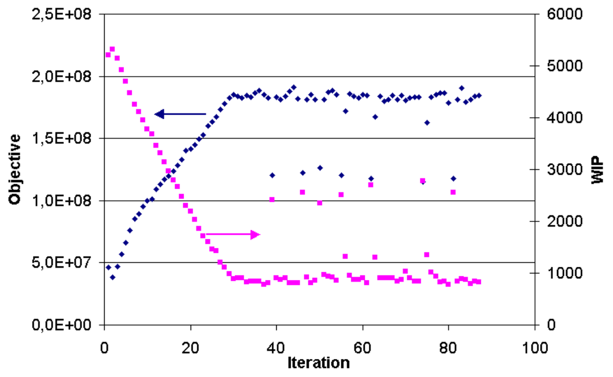

Figure 5: Tool Count Optimization, 0.8 µm CMOS, 720 Wafer Starts per Day

A few excursions (data points with low "Objective") are obvious. These points correspond to high Wip, and also to high standard deviation of Wip (not shown). They are an indication that the optimization moves along the border of stable fab operation. High Wip and standard deviation denote a fab with Wip increasing during the simulation run. The factory will not release all wafers which have been started, wafers accumulate in toolset buffers.

The tool count evolution during the iterations is shown in Figure 6. It is smooth, as imposed by the neighbor selection rule. Starting figures are 1, 10 or 20 machines. The uppermost line with nearly 50 tools needed indicated an improperly configured oxide deposition tool, which was replaced by an optimized equipment better suited for intermetal dielectric deposition in a multilevel metal environment. All data are stored in an output text file for further analysis.

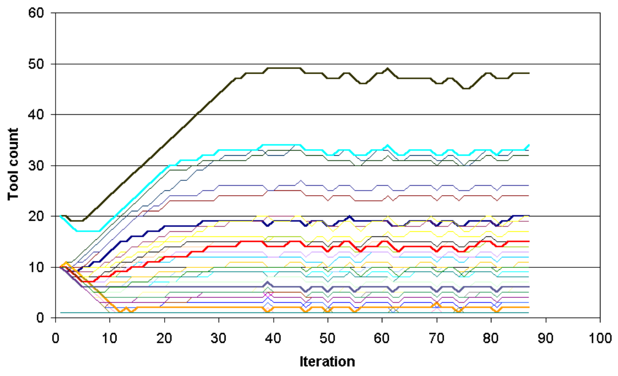

Figure 6: Evolution of Tool Count versus Iteration for all 45 Toolsets

6 CONCLUSION

A compact and simple schedule for optimizing the machine quantity in a semiconductor factory using FabSim Interactive has been presented. Extending a simulated annealing algorithm with toolset and machine utilization data serves to obtain quick convergence. Optimization rules are easily modified to improve the simulation optimization.

REFERENCES

Alrefaei, M. H., and S. Andradóttir. 1999. A Simulated Annealing Algorithm with Constant Temperature for Discrete Stochastic Optimization. *Management Science* 45 (5): 748–764.

Andradóttir, S. 1998. Simulation Optimization. In *Handbook of simulation*, ed. J. Banks, 326-328. New York: Wiley.

Grötker, T., S. Liao, G. Martin, and S. Swan. 2002. *System design with SystemC.* Boston, MA: Kluwer Academic Publishers.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and Analysis*. Boston, MA: McGraw-Hill. 93-94.

Rose, O. 2003. Accelerating Products under Due-Date Oriented Dispatching Rules in Semiconductor Manufacturing. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1346–1350. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via `<http://www.`

informs-cs.org/wsc03papers/168.pdf> [accessed February 2, 2004].

Vogt, H. 2003. Discrete-Event Simulation using SystemC: Interactive Semiconductor Factory Modeling with FabSim. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1383–1387. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via <http://www.informs-cs.org/wsc03papers/174.pdf> [accessed February 2, 2004].

AUTHOR BIOGRAPHY

HOLGER VOGT has a diploma degree in electrical engineering from University of Dortmund, Germany. 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 and 0.18 µm technologies on 200 mm wafers at IMS. Since 1997 his is also professor at the EE&CS department of University Duisburg-Essen, Germany, teaching semiconductor technology and packaging. He is a member of IEEE, ECS, and VDE. The FabSim web site is <http://www.fabsim.com>. <holger.vogt@ims.fraunhofer.de> is the author's email address.