

## RESOURCE-DRIVEN AND JOB-DRIVEN SIMULATIONS

Theresa M. Roeder\*, Seth A. Fischbein†, Mani Janakiram†, and Lee W. Schruben\*

\*Department of Industrial Engineering and Operations Research  
University of California, Berkeley  
Berkeley, CA 94720  
e-mail: [roeder@ieor.berkeley.edu](mailto:roeder@ieor.berkeley.edu)

†Intel Corporation  
5000 W. Chandler Blvd.  
Chandler, AZ 85248

**KEYWORDS**  
Simulation, Wafer Fab

Resource-driven, Job-driven,

orders of magnitude faster than the job-driven simulation model currently in use.

### ABSTRACT

The typical approach taken to simulating semiconductor wafer fabs is a time- and memory-intensive process that tracks individual wafers as they pass through the fab. This modeling paradigm is known as a job-driven simulation. In this work, we describe research done at UC Berkeley in cooperation with Intel, the Semiconductor Research Consortium, and SEMATECH to develop a resource-driven simulation to modeling fabs. Developing the new model provided interesting insights into modeling assumptions typically made. It also showed that some aspects of the fab can be modeled more accurately in the resource-driven model, while others are more accurately modeled using a job-driven approach. However, the resource-driven model runs considerably faster, making more extensive experimentation possible.

As anticipated, there are tradeoffs between the two approaches and using them together is more effective than relying on either one alone. However, a resource-driven simulation can easily be enriched to include all the job details of a job-driven simulation whenever and wherever such information is desired, but with a corresponding loss in execution speed. It is not necessary to have two *different* simulations to use both approaches.

Using a purely resource-driven approach, it is difficult to simulate job-dispatching rules that select jobs for processing based on the characteristics of individual jobs. At the same time, dispatching rules that use product information are easy to model. It also appears difficult to model dedication constraints accurately. Dedication constraints refer to the restriction that re-entrant jobs must use *exactly* the same resource, not just the same *type* of resource on successive processing passes – common for some critical lithography operations.

### INTRODUCTION

This paper compares two methodologies for simulating the operations of a semiconductor wafer fabrication facility (fab). The conventional approach, referred to as a job-driven simulation in the literature, models the flow of individual wafers or lots through the system. In this type of simulation, system resources are modeled as passive and are “seized” by jobs when they are needed and available. A different viewpoint, which we will call a resource-driven approach, focuses on modeling resource cycles. Instead of records of each job in the factory, only integer counts of the numbers of each type of job at each processing step and of available system resources are maintained (Schruben 2000; Hyden et al. 2001). Here, jobs are passive and are actively processed by resources. This perspective better represents the actual relationship between resources and jobs in many systems and helps avoid some common modeling errors such as queued resource failures or artificial resource deadlock (Venkatesh et al. 1998). Furthermore, our resource-driven implementation runs

In the following section, we describe the two contrasting implementations of the wafer fab simulation. We will focus on two important aspects of the system that are modeled in very different ways in job-driven and resource-driven simulations. We conclude with a discussion of the study’s results.

### SYSTEM DESCRIPTION

To contrast the job-driven and resource-driven methods, we developed a resource-driven simulation of a job-driven model of an actual wafer fab. That is, we simulated a simulation using a different modeling technique and software. This allowed us to do a much more detailed validation of the resource-driven simulation than would otherwise be possible. In the fab being modeled, more than five different part types are processed using hundreds of tools of over 80 different types. Individual wafers are grouped together in lots of size 25, and are processed as a lot. Jobs/lots visit the

same tool repeatedly, usually requiring different processing times at different stages. Depending on the tool, lot, and processing stage, different setups may be required for the tools. Each job has a basic route, but certain steps (such as metrology steps) on each route are skipped with a given frequency. For example, only every 5<sup>th</sup> lot to reach a metrology step will be processed at this step. All other lots jump immediately to their next step. The basic route has over 500 steps.

In the job-driven simulation model, tools are grouped into functionally identical tool types called “families”. Each tool family has a different number of tools. Tools in a tool family are functionally identical; any tool in the family can process any lot, and each lot is processed by exactly one tool. Each lot’s route specifies the sequence of “tool families” it must visit in order to be processed. The software that was used for developing the job-driven simulation, ASAP, refers to tools as “stations” and tool groups as “station families”. Lots are also referred to generically as “parts”.

At a high level there are three fundamental classes of tools: serial-processing tools, batch tools, and stepper tools. Serial-processing tools are not subject to strict rules on which lots can be processed next; they select the next lot waiting in queue. Batch tools, as their name suggests, process lots in batches whose size depend on the tool family. The next batch is selected based on the first job type where there are enough lots of this type waiting in queue to form a batch. Lots at the same processing step can be batched together. For example, job type 1 at step 6 can be batched with job type 2 at step 6, but not with job type 1 at step 10. Stepper tools are the tools used for photolithography, and follow a complex set of guidelines governing which and how many lots should be processed next. They may also be subject to dedication constraints.

Most tools have associated load and unload times for processing lots. Stepper tools also have setup times. These setup times are different depending on the tool family, and on current and future setups.

Tools are subject to failures and preventive maintenance events (PMs), and periodic test and rework wafers visit the tools. PMs must be performed at specified intervals, whether or not the tool is experiencing processing difficulties. Failures occur at random times and can result in extended downtimes for a tool. Each tool family has a set of PMs and failures associated with it. They can be scheduled by calendar (e.g., a tool must receive PM *i* every 2-3 days) or by lot (e.g., a tool must receive PM *j* after processing *n* lots). Each tool in a family has its own record of PMs and failures, and follows a different PM schedule. The PM and failure distributions for each tool in the family are the same and all PMs and failures are modeled as occurring independently of one another. Due to a limitation in the job-driven modeling approach, it is possible for PMs and failures to queue a tool – PM *k* may be scheduled at a time the tool is still receiving PM *i* or

is being repaired from failure *j*. This occurs because, in the job-driven model, tools are taken off-line by dummy “jobs” that represent failures or PMs. This modeling problem can be easily avoided in resource-driven simulations.

In this comparison, operators were not modeled (more will be said about this later). Additionally, generic tools such as reticles and masks were not included to simplify the modeling.

## COMPARISON

The job-driven simulation was built in AutoSched AP (ASAP), a popular commercial simulation package used almost exclusively for wafer fab simulation. Data and options are specified in spreadsheets and processed by the software. Some customized functionality and reports have been added to the simulation software.

The resource-driven simulation was implemented using SIGMA to build the underlying simulation engine, and to automatically generate C source code. Additional functionality to mimic the behavior of the job-driven simulation was added in C. The spreadsheet input data for the ASAP model were imported into a MS Access database, which, along with MS Excel, was used to convert the data into the plain text input files used by the resource-driven simulation.

## Differences in Modeling Approaches

There are major differences in how job-driven and resource-driven simulations model certain aspects of the system. This paper will focus on two of these differences: queueing disciplines and PMs/failures. Both illustrate advantages and disadvantages of the two simulation paradigms.

The vast majority of tools use critical ratio ranking to order jobs in queue. This discipline compares a job’s actual time in the system to the amount of time it should have been there. The further a job is past the average time it takes to get to this stage, the closer it moves to the front of the queue. When the tool has finished processing its current job, it begins processing the next job in queue. This ranking is used to allow the artificial jobs that represent PMs or failures to be placed at the front of the job queue. (They are assigned extremely negative critical ratios to bump them to the front of the queue.)

In order to implement critical ratio ranking, jobs need to know how much time they have spent in the system. This information is readily available in the job-driven simulation, which tracks each job individually. In the resource-driven simulation, however, we do not immediately have access to this information, and would therefore have to expand our state space to model critical ratio ranking.

In addition, we are only maintaining integer counts of each job type in queue – which alone does not give us enough information to determine which job (type) is “next” in queue. If, for example, there are 3 job 1 lots and 4 job 2 lots in queue, we do not know if job 1 or job 2 is at the front of the line. Again, we could expand our state space to gather this information if we chose. For this project, however, we were interested in comparing the results from a “pure” resource-driven simulation to those of the job-driven simulation to see how much could be modeled using a resource-driven simulation, and how accurate the results would be. As a consequence, we did not expand our state space. Since we cannot, in this case, implement the queueing disciplines used in the job-driven simulation, we (arbitrarily) decided to process the job with the largest number waiting in queue next.

Another major difference in the two approaches can be seen in how they model PMs/failures. In the job-driven approach, there is no differentiation between preventive maintenances (PMs) and failures. If a PM is scheduled to occur and the tool is processing a lot, the PM will be performed after the current job has finished processing. Unfortunately, due to the inherent limitation of the job-driven approach, this is, unrealistically, the same way failures are modeled. PMs and failures are represented as extremely high-priority jobs that must be processed. Consequently, the memory footprint of the simulation increases as the number of PMs and failures increase. Failures are modeled as PMs because they are difficult to model correctly. Internal studies showed that there is little difference in system performance if failures are modeled as such or approximated by PMs.

In resource-driven simulations, it is quite simple to treat PMs and failures properly. For example, failures will generally occur while a lot is being processed, and will not wait until it has finished doing so. The lot either is discarded; or rejoins the queue to finish its current service, or to restart processing. Any of these possibilities can be modeled easily. For example, if the job rejoins the queue, we can simply maintain a separate integer count of jobs that have begun, but not finished, processing. In this fab model, jobs complete processing once the failure has been repaired.

Neither PMs nor failures are modeled using actual events. Instead, each tool (in each tool family) has an array containing the times each of its PMs and failures are scheduled to occur next. Every time a job begins processing, we pass through the array to see whether a PM (or failure) occurred between the time at which the tool last finished processing and the current time; and whether there are PMs/failures scheduled to take place during the current lot’s processing time. In the former case, we add the PM or repair times to the tool’s cumulative “time spent in PM” or “time spent in repair” counter. In the latter case, the `FINISH PROCESSING` event is scheduled after the additional PM or repair time. This does not imply that the tool has spent the entire time processing the lot. Rather, the PM or repair time is

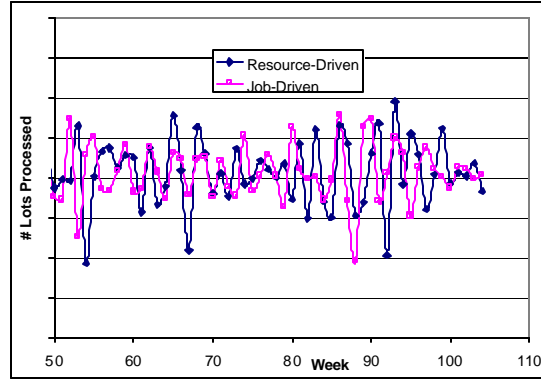


Figure 1: Throughput for one of the tool families

added to the “PM” or “repair” counters mentioned above. The lot’s processing time is unaffected by this. We should note here that PMs and failures are effectively modeled the same way in our model as in the job-driven model. However, this is only because the decision was made to resume processing a job once the repair was completed. Had the job’s processing time changed due to the failure, or had the job simply been discarded, there would have been differences. These differences would have been very simple to model in the resource-driven environment, however they would be difficult to model with a job-driven simulation.

### Experimental Results

Both models have advantages and disadvantages that will be discussed in this and the next sections. The main advantage of the resource-driven model is that it runs an order of magnitude faster than the job-driven one. The following are execution speeds for 2 years of simulated data on a Dell Dimension Pentium 4 1.7GHz desktop with 512MB of RAM. (The processing times include reading data into the simulation and generating output reports.)

	JOB-DRIVEN	RESOURCE-DRIVEN
With PMs/Failures	≈ 3 hours	< 10 minutes (≈ 352 seconds)
Without PMs/Failures	≈ 1.5 hours	< 10 minutes (≈ 300 seconds)

There are two interesting differences in speed to note. The first is between job-driven and resource-driven. The speedup is dramatic – the resource-driven model runs an order of magnitude faster, both with and without PMs/failures. This can be attributed mainly to the large memory footprint for the job-driven approach, as there are many wafers traversing the system. Since the resource-driven model is merely keeping track of the *number* of wafers in the system, it is not affected by the large scale of the simulation. (The speed comparisons cannot entirely be attributed to the difference in modeling approach since ASAP probably has a greater

execution overhead than the SIGMA-generated C executable.)

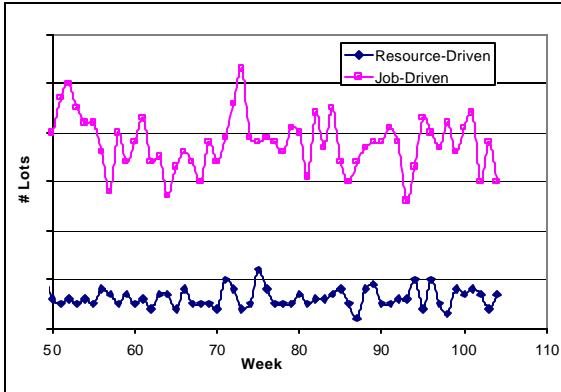


Figure 2: Weekly ending WIP for a stepper tool with dedication

The second difference in speed is for the job-driven model, between the simulation with PMs/failures and the one without. The execution time roughly doubles when PMs/failures are included in the model. This demonstrates quite readily the impact of being forced to model PMs/failures as high-priority jobs: System congestion increases. For example, the largest number of PMs/failures associated with a tool family is close to 100. That means that, for this tool family, there are nearly an additional  $100 \times (\text{number of tools})$  jobs that must be processed repeatedly! Though 100 is an upper bound on the maximum number of PMs/failures associated with a family, it illustrates nicely how treating PMs/failures as jobs can increase congestion. The system requires additional memory for the new “jobs” that have been added, but there is also a lower-level system impact – the memory for the new jobs must be allocated and deallocated. This is typically an expensive operation. In contrast, there is very little impact on the resource-driven model. The difference in speed there is because no PM-related function calls are required when the model does not include PMs/failures.

As discussed already, this dramatic increase in execution speed comes at a loss of ability to model certain aspects of the real system (e.g., queueing disciplines), and to obtain all output statistics that would be available otherwise. On the other hand, features such as PMs/failures or travel times can be modeled in a straight-forward, realistic manner with the resource-driven approach.

In general, the output from the resource-driven simulation is quite comparable to that of the job-driven simulation. In addition to a visual inspection, paired- $t$  tests were performed to see whether there were statistically significant differences between the two. The weekly statistics were grouped into batches of 4 weeks each, and the first 10 batches of data were discarded. This was done to approximate Normality for the paired- $t$  test and to remove initialization bias from the study. The tests showed that there was no statistically significant

difference in the vast majority of the cases. We have been able to account for all but two of the discrepancies.

Figure 1 shows the throughput for one of the tool families. (All figures show the output after discarding the data for the warm-up periods.) It is not possible to differentiate between the outputs of the two models. These results are typical for almost all tool families. One interesting artifact we uncovered was that the resource-driven simulation was not modeling tool dedication for the stepper tools. Figure 2 shows the weekly ending WIP (work-in-progress) profiles for a stepper tool when the job-driven simulation uses dedication. WIP for the resource-driven model is significantly lower than that for the job-driven simulation. This result is not surprising, as tools are more likely to go idle with jobs waiting in queue when the tools are subject to dedication: Tool  $i$  may not be able to process waiting jobs since the jobs were previously serviced by tool  $j$ . Figure 3 shows the WIP profiles when the tool is not subject to dedication.

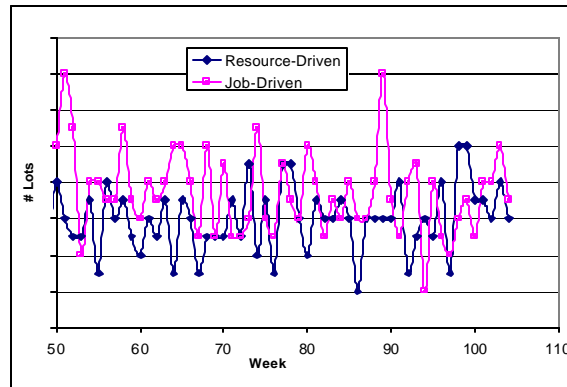


Figure 3: Weekly ending WIP for the stepper tool in Figure 2, without dedication

Both the average throughput and average WIP levels for the resource-driven simulation are accurate for all but a handful of tool families. It is interesting to note, however, that the variances of the weekly ending WIP values appear to be larger for the resource-driven simulation than they are for the job-driven simulation. While we pass the paired- $t$  test for equality of averages, we are likely to fail  $\chi^2$  tests for equality of variances. (Note that we expect the averages to be equal since the average queue size is independent on queueing

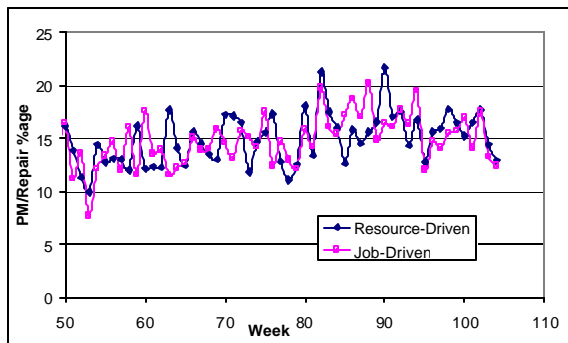


Figure 4: Percentage of time a tool family spends in PM/repair

discipline. The variance of the queue size is *not* independent of queuing discipline.)

Despite the very different approaches taken in modeling PMs/failures, there are no differences in the amounts of time tools spend in maintenance or being repaired. Figure 4 shows the percentage of time tools for a specific tool family spent in PM or repair every week. The two curves are indistinguishable.

## DISCUSSION

In the discussion of the differences in the modeling approaches, we alluded to the advantages and disadvantages of job-driven and resource-driven simulations. In this section, we elaborate further on the differences.

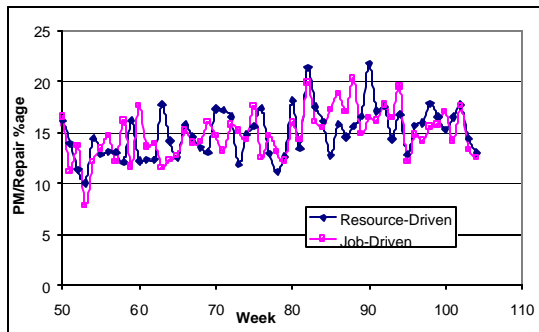


Figure 4: Percentage of time a tool family spends in PM/repair

Clearly, queueing disciplines that require global job information are more easily modeled in the job-driven simulation. The resource-driven simulations must be enriched to capture the delay-based information for ranking jobs in the queue, or to capture the ordering of jobs in the queue. Similarly, resource-driven simulations would have to be augmented to model dedication constraints, as it is necessary to differentiate between jobs in order to know which tool serviced every job on previous visits to the tool family.

Another advantage of job-driven simulations is that they can readily return any output desired – job waiting time distributions, cycle times, etc. Resource-driven simulations can approximate averages, e.g. the average waiting time can be obtained using Little’s Law – the average queue size is a known quantity, and the effective arrival rates to the tools can be estimated. For more detailed information, however, the resource-driven model would have to be enriched.

Event Graphs can be used to model both pure resource-driven and job-driven models – and all “shadings” between. Using them, the pure resource-driven simulation can easily be modified to include information contained in the job-based simulation. This will slow execution speed as the memory footprint of the simulation increases with the amount of information included in the model. However, the “augmented”

resource-driven simulation *can* capture the system aspects the “pure” resource-driven model is unable to, though the job-driven simulation is. In addition, some very promising approximation techniques for solving these problems are currently under investigation.

On the other hand, (even) pure resource-driven simulations are able to more accurately model preventive maintenances and failures. Job-driven simulations are forced to simulate PMs and failures as extremely high-priority “dummy” jobs that “seize” the tool. Resource-driven simulations are not constrained in this way, and can model them accurately. They are able to model time to failure rather than time between failures, and so are not subject to the problem of queued PMs/failures. In addition, it is not difficult to represent failures as failures, which allows greater flexibility in modeling what happens to parts when the tool fails – they can be discarded, set aside to resume, etc. It is possible to model this in the job-driven approach, but is likely to involve ingenuity in “tricking” the software to perform the desired behavior.

The main advantage of resource-driven simulations is their speed. In this study, we achieved an order of magnitude speedup over the job-driven model. While the execution time roughly doubles when the job-driven simulation includes PMs/failures, the additional time required for the resource-driven simulation is minimal. This is because the resource-driven model does not create artificial congestion in the system in order to model PMs/failures.

When modeling the full fab, including operators and generic tools, the job-driven simulation triples its execution time to roughly 9 hours. We believe that there would not be such a dramatic increase in time for the resource-driven simulation. The main difference in the simulation would be additional integer counts of the other available resources. Even if this causes the execution speed to increase tenfold (which we consider extremely unlikely), the simulation would complete in less than 2 hours.

## CONCLUSIONS

We have shown that resource-driven simulations can have much faster execution speed; however, important aspects of the fab such as delay-based queueing disciplines and tool dedication may require enriching them to include some of the information in a job-driven simulation. In addition, output statistics such as waiting and cycle time distributions also may require expanding our state space considerably (or again, simply including job information in local resource-driven job queues). We believe that resource-driven simulation can and should be a useful tool for the simulation practitioner. For example, in the time it takes to run a single job-driven run, we can run a full 2-level/4-factor experiment with the resource-driven simulation. Since the output from the resource-driven simulation is comparable to

that of the job-driven simulation in most cases, the resource-driven model could be used to find an initial set of parameters for more detailed system optimization. A job-driven simulation (or better yet, the enriched resource-driven simulation) can then be used to obtain detailed output statistics on the reduced set of configurations to make the final decision.

Lee Schruben is a professor of Industrial Engineering and Operations Research at the University of California, Berkeley, and a former member of many honor societies.

## ACKNOWLEDGEMENTS

The authors are grateful to the Intel Corporation, SRC, and SEMATECH for funding this research. In addition, we would like to thank Paul Hyden and Hyun-Soo Ahn for useful discussions on resource-driven simulations and fab simulations. We appreciate Ed Yellig's support throughout this project.

## REFERENCES

AutoSimulations, Inc. 1999. AutoSched AP User's Guide v 6.2. Bountiful, Utah. (April).

Hyden, P.; L. Schruben; and T. Roeder. 2001. "Resource Graphs for Modeling Large-Scale, Highly Congested Systems." Proceedings of the 2001 Winter Simulation Conference: 523~529.

Schruben, D. and L. Schruben. 2000. Graphical Simulation Modeling Using SIGMA. Custom Simulations.

Schruben, L. 2000. "Mathematical Programming Models of Discrete Event System Dynamics." Proceedings of the 2000 Winter Simulation Conference.

Venkatesh, S.; J. Smith; B. Deuermeyer; and G. Curry. "Deadlock Detection and Resolution for Discrete Event Simulation: Multiple-Unit Seizes." IIE Transactions, vol.30 no.3 (March): 202-216.

## BIOGRAPHIES

Theresa Roeder is a Ph.D. candidate at the University of California, Berkeley. Prior to coming to Berkeley, she earned B.S. and M.S. degrees in Management Science from Case Western Reserve University.

Seth A. Fischbein is a Manufacturing Modeling Engineer at Intel Corp. in Chandler, AZ. Seth has a B.S. degree in Operations Research and Industrial Engineering from Cornell University.

Mani Janakiram is a Systems Engineering Program Manager at Intel Corp. in Chandler, AZ. Mani has a Ph.D. in Industrial Engineering from Arizona State University.