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

THE USE OF BOTTLENECK STARVATION AVOIDANCE WITH QUEUE PREDICTIONS IN SHOP FLOOR CONTROL

C. Roger Glassey
Raja G. Petrakian
Department of Industrial Engineering and Operations Research
University of California
Berkeley, CA 94720, U.S.A.

ABSTRACT

The accurate estimation of lead times and the use of factory-wide information can improve the performance of dynamic shop floor control. This paper presents a dispatching policy that is based on the concept of bottleneck starvation avoidance and relies on frequently updated queue predictions for all workstations. The queue predictions are used to dynamically estimate the lead times required for lots to reach workstations on their routes, particularly the bottleneck workstation. Object-oriented simulation experiments were run for several wafer fab configurations with results showing a consistantly good behavior of the computationally intensive control mechanism presented here.

1. INTRODUCTION

Production control in wafer fab, the clean room where integrated circuits are first fabricated, presents a scheduling challenge that has yet to be tackled successfully. One of the complicating factors is that most wafers must flow many times through the same workstation which happens often to be the bottleneck. Thus many products, at different stages of their manufacturing cycle, are competing for capacity on the same workstation. In addition, the semi-conductor industry is a late comer in the world of computer-integrated manufacturing (CIM), and the huge amount of data produced by real-time tracking systems has largely gone unused.

Actually, the increasing availability of large amounts of data in CIM environments creates many new possibilities and needs in the area of wafer fab production control (Hughes and Shott 1986). Usage of this data will determine the extent of improvements that are made in controlling production. The challenge is to transform this type of data into meaningful information for job shop scheduling. Traditionally, scheduling has been based on information that is local both spatially and temporally. For example, when making dispatching decisions, ie. selecting a job from a queue at a workstation, the present statuses of the workstation and of the queued work orders are often the only types of dynamic information that are taken into consideration. Adding spatial and time dimensions to the information provided by a manufacturing control system may result in improved decisions. The extra computing times and computing power required to generate a much richer pool of information has often stood in the way of such developments. Also, programming the code that will perform data analysis is a complex task that requires both programming skills and understanding of the subtleties of the manufacturing environment. Testing for possible flaws in both the code and the logic behind it, is also very difficult because most factory simulation packages provide very little flexibility.

Among the many objectives one could consider for production control, minimizing cycle time is of a great importance in wafer fabrication. Shorter cycle times have the effect of improving the reaction time to demand fluctuations and to yield crashes, decreasing work-in-process, and also reducing the time during which wafers are exposed to impurities and thereby increasing yields. A large fraction of the cycle time for a lot (more than 70% in general) is the time spent waiting for production, particularly at the bottleneck workstation. Notice that by Little's law (Stidham 1974 and Wolff 1989), minimizing cycle time is in fact equivalent to minimizing total inventory on the shop floor when the troughput rate is held constant. This justifies the usage in this work of the average total inventory on the floor as a measure of performance.

In this paper, a scheduling approach that aims at minimizing the queue size at the bottleneck workstation and which relies on the usage of predictions and floor-wide information is presented. Only the details and results of research done on dispatching are described, work on lot release is ongoing.

2. DESCRIPTION OF THE FAB MODEL

The model described next is used to test, via simulation experiments, all of the scheduling policies.

A fab is defined to be a set of K workstations. Each workstation is a group of m_k (k = 1, 2, ..., K) parallel identical and unreliable machines. The transportation time for a lot from a workstation k to another workstation k' is considered to be deterministic with value $T_{k,k'}$. A machine can be either idle, busy, or down. It can be used to proces lots only when it is not down. Machines fail in a nonpreemptive manner, thus a busy machine is not allowed to fail until it completes the current operation. The time to repair and the up time between failures, for a machine at workstation k, have exponential probability distributions with means $MTTR_k$ and $MTTF_k$ respectively. In front of every workstation a queue of infinite capacity is placed, into which lots are put when they arrive at the station, and from which they are taken to be processed in an idle mahine of the workstation.

Each product i (i = 1, 2, ..., I) is defined by its route, which is a list of operations. Notice that there is a one-to-one correspondance between products and routes. Next, every operation h (h = 1, 2, ..., H) is to be performed on a workstation k(h) with processing time p_h. Step s of product i's route (labelled (i,s) with s = 0, 1, ..., S_i), corresponds to some operation h. It is assumed that products are started at constant intervals determined by r_i, the release rate for product i.

A dispatching decision has to be made anytime a workstation has at least one idle machine and one or more lots are in the queue. For any lot j, a dynamically updated priority index r_j ($r_j \geq 0$, for all j) is assigned. The lot with the highest priority index in the queue will be selected for processing at the idle machine.

3. LITERATURE REVIEW

Job shop scheduling has been the object of a large number of studies (Blackstone, Phillips, and Hogg 1982), but research on production control for wafer fabrication is still a new field with not many practitioners. Resende (1987) presents an extensive review of publications related to scheduling of integrated circuits manufacturing systems. Some of the work that has been conducted since then in this area is discussed next.

In the context of implementing a just-in-time (JIT) approach in a fab owned by Harris Semiconductor Corp., Martin-Vega et al. (1989) show that changes in lot size and fab layout, as well as extra operator flexibility led to important reductions in cycle times and inventory levels. Ehteshami and Rohani (1989) discuss the extent to which current experience in assembly line automation and production control can be applied to wafer fabrication. They also argue that manufacturing management approaches used presently in fabs are too static for a CIM environment. An ambitious program, the Logistics Management System (LMS), has been developed and implemented at one of IBM's fabs (Sullivan and Fordyce 1989); it is a real-time tracking system coupled with a knowledge-based expert system used to support short-term manufacturing decisions. It balances between conflicting goals such as improving machine utilization, delivering products on time, reducing work-in-process (WIP), and insuring that hot lots are given higher priority. This system is supposed to maintain estimates of upper and lower bounds on the queuing times for the remaining steps of every job as well as estimates of the availability of bottleneck workstations. The authors did not explain how the estimates are calculated or how they are used for scheduling.

Lou (1989) presents a flow rate control rule derived by solving simple stochastic optimal control problems. In this approach a workstation is loaded only when the inventory behind it and the downstream surplus are below the inventory hedging point and the surplus hedging point respectively. These threshold points, which are prespecified using some algorithms that were not discussed, are crucial. They determine how capacity, for these workstations which are visited more than once by a product, will be divided among the visits. The simulations that Lou has performed show the robustness of the flow rate control rule. Wein (1988) has conducted a large number of simulations to compare the performance of a variety of scheduling policies on fab performance. He introduces the workload regulating input control rule; it allows new lots to start in the fab

only if the total remaining workload from WIP, for any heavily utilized workstation, falls below a prespecified target level. Wein also presents some new dispatching rules derived from a Brownian network model that ignores all workstations that are not heavily utilized. These dispatching rules did not perform very well because the excluded workstations formed such a large fraction of the fab and became temporary bottlenecks so frequently that ignoring them had a negative effect of the overall factory performance. An important observation made by Wein is that the choice of the dispatching rule has less effect on reducing cycle time than that of the input control.

A similar conclusion is reached by Glassey and Resende (1988) who introduce both a new dispatching rule and a continuous-review lot release policy based on the concept of bottleneck starvation avoidance. The dispatching control rule is a weighted combination of Shortest Remaining Processing Time (SRPT) and a decision rule that favors jobs closer to the bottleneck and/or bringing more work there; the weights are dynamically changed as functions of the bottleneck utilization status. The objective of the release rule is to start lots in the fab whenever the work, expected to arrive at the bottleneck within a lead time equal to the time it takes a released lot to reach the bottleneck for the first time, falls below a prespecified safety level. This workload target level is set to be equal to the total capacity of the bottleneck over the lead time plus a safety margin. Notice that the lead time calculations don't include either transport times or queue times thus leading to possible uncertainties in the estimates. The bottleneck starvation avoidance release rule comparerd favorably with many scheduling rules including Wein's workload regulating input policy. A graphic tool that supports this release policy has been developed and tested in an existing fab (Lozinski and Glassey 1988).

Leachman, Solorzano, and Glassey (1988) show that methods based on comparing, for the bottleneck, a total load to a total capacity over some time horizon might result in underestimating the queue size at the bottleneck at the end of the lead time. They propose to maintain information about WIP arrivals at all workstations and use it to calculate the sizes of all queues at discrete time periods. Lot releases are assumed to be possible only at the end of a review interval since, according to the authors, in most real-world factories release decisions are made on a periodic review basis, such as once a shift or once a day. They consider that release of an order is desirable only when the projected queue of every workstation to be visited by the lot is below a prespecified safety level, in the predicted time period of arrival at the workstation, or in any following time period less than a review interval later. This approach uses average flow times derived from historical data to estimate the WIP arrivals at the workstations and updates all information once every review period.

These two previous approaches have in common that lead times estimates are very imprecise because either waiting times are just ignored or are calculated using historical averages. Since waiting times, and thus lead times, vary widely from lot to lot and from one production step to another, lead time estimates that do not attempt to capture these variations fail to predict actual lead times with much accuracy. As a result, predictions of WIP arrivals, and consequently estimates of queue sizes, become in turn highly imprecise.

4. BOTTLENECK QUEUE PREDICTION

While a periodic review approach for lot release is practical given today's business constraints, it is not a superior one. It fails to give manufacturing management the ability to respond quickly to changes on the factory floor. Before CIM systems were available, information describing the state of the factory could only be gathered infrequently and imprecisely. At the time, management had to wait for the marketing group to state its requests and for feedback from the factory floor to materialize in the form of status reports before it could decide on the course of action to take. In a CIM environment, real-time feedback is provided and can be used both to start and to dispatch lots in an event-based manner. An event is defined to be the occurence of a change in the state of the factory. Given the highly volatile nature of wafer fabrication, reaction time to changes in the fab is very important, and postponing decisions ultimately negates performance. On the other hand, assigning a person to perform scheduling decisions on a full-time basis is often a costly and infeasible approach. A solution would be to allow a scheduling module, integrated with a CIM system, to either release and dispatch lots in an event-activated mode, or to assist decision-makers by providing them with useful and frequently updated information. With the increasing computing power available today, a scheduling module has the ability to retrieve data and transform it into relevant information quickly; it can also base scheduling recommendations and decisions on a much larger number of parameters. The aim of this work is to show how the use of

frequently updated predictions and global information can improve on scheduling decisions.

4.1. Estimating Queue Sizes and Lead Times

In the approach presented here, estimates of the queue sizes at all the workstations on the floor are maintained and updated at the occurence of most events. These estimates are then used to calculate, for every lot, the arrival times to all the workstations remaining to be visited. Prediction of the time a lot will spend queued at a workstation is computed on the basis of the estimate of the queue size as well as on the expected workstation's status at the estimated lot's arrival time.

Four major events trigger updates of estimates: a release of a new lot into the fab, the start of an operation on a lot, the arrival of a lot to a workstation, and the completion of all operations on a lot. The sequence for updating information is dependent on the type of event that occurs, but nonetheless a common pattern exists. For a given lot, the waiting times and the arrival times at all remaining steps are first recalculated using the maintained queue size estimates. Next, all workstations which are still to be visited by the lot are sent information regarding changes in the arrival times. These workstations update information on WIP arrival and recalculate estimates of the queue size for the affected time periods.

The amount of extra computation per event is very large and the implementation of these calculations requires that approximations be made whenever possible (Leachman, Solorzano, and Glassey 1988). It is easier to maintain a higher degree of precision in estimating a lot's lead times and arrival times because thay can be naturally expressed in continuous terms. On the other hand, projections of workload arrivals and queue sizes over time at a workstation are more difficult unless they are approximated by the use of discrete time periods. Notice that increasing the size of the time periods decreases the amount of calculations but it also causes estimates to be less accurate. Thus the length of the time period should be chosen with special regard for the trade-off between added precision and increased calculations. In all the simulation experiments in this work, the size of the time periods was set to be equal to 20 time units. Another parameter that can be used to alter that trade-off is the time horizon over which predictions are made. To calculate queue sizes, the following approach is used: given that the workload is the sum of new work and remaining work, the queue size at the end of a time period is set equal to the difference between the workload and the capacity available during that time period unless this difference is less than zero in which case it is set to zero. Queues and arrivals at a workstation are measured in machine-hours. A workstation's capacity is calculated differently depending on the time period. It is expected to remain unchanged from its current level until some time period beyond which it can be approximated by an estimate of the workstation's average capacity. Calculating the time a lot will spend in queue at a workstation during a future visit is crucial for estimating the lot's lead times. To simplify the amount of required calculations, it is assumed that a lot will be be processed only when all of the lots expected to arrive at the workstation earlier or during the same time period, have already been taken from the queue. Again the rate at which lots are expected to be processed varies from one period to another. To illustrate this dicussion, the algorithm used to update predictions in the event of a lot start is presented in Appendix A.

4.2. The Use of Predictions for Dispatching

Maintaining and updating frequently a large pool of information is justified only if this information can be actually used to generate a superior schedule for operations in the fab. The Bottleneck Queue Prediction (BQP) approach is a policy for real-time dispatching that makes use of queue size projections and lead time estimates. Since the bottleneck is the workstation whose queue affects the most waiting time performances, the immediate objective of this policy is to minimize the size of the queue in front of the bottleneck. Consequently, lots' priority indices are calculated using an approach that tries to achieve this goal. In order to make dispatching decisions using the most recent information, the priority index r_j of each lot j in the queue is always recalculated. It is estimated using

$$r_j = w_j \lambda_j$$

where λ_j is a multiplier that reflects the expected congestion at the bottleneck when the lot arrives there next and where w_j is a weight given by management to lot j at the time it was released. In this work, all lots are weighted equally with a value of 100,000. Calculating λ_j for a particular lot j requires first that the time of its next visit to the bottleneck be estimated. Then, the queue size that lot j is expected to find at its next arrival to the bottleneck is projected. All of these calculations are done using an approach similar to the

one presented in Appendix A. Next, the bottleneck queue size estimate, noted Q, is used in the following manner to calculate the multiplier λ_j:

$$\lambda_j = \exp (- \alpha Q / I)$$

where α is a prespecified control parameter and I is the target work inventory for the bottleneck. Notice that λ_j decreases as a function of the queue size at the bottleneck: it is equal to one when the queue is empty, it takes the value exp ($-\alpha$) when Q is equal to the target inventory, and it reaches zero when Q grows to infinity. If lot j has no remaining visits to the bottleneck then λ_j is set equal to exp ($-\alpha$). After many experimentations, it was determined that BQP gives the best results when α is set to one. Consequently, this value of α is used for all the simulations conducted in this work.

5. SIMULATION RESULTS

To compare BQP with other dispatching policies, a large number of simulations were run using two different fab models. The two fabs correspond to two different configurations of the mathematical model described in section 2. The simulations and their results are described next.

The Berkeley Library of Objects for Control and Simulation (BLOCS), provided the development environment in which this work was conducted. Glassey and Adiga (1989) describe BLOCS and show how the use of object-oriented programming allows a flexible representation of manufacturing systems as well as an easy construction of simulations with special decision rules. All the dispatching rules, except BQP, were tested using standard objects from BLOCS. Since BQP is a new scheduling policy, two special software objects were required. They were easily coded using BLOCS' object-oriented implementation language, Objective-C, a product of Stepstone Corporation. All simulations were run on a Microvax II workstation at U.C. Berkeley.

The release strategy in the simulations is to start every product at constant intervals. BQP is compared to four dispatching rules: Shortest Remaining Processing Time (SRPT), Shortest Imminent Processing Time (SIPT or sometimes SPT), First In First Out (FIFO), and Longest Delay per Unit of Processing Time (LDUPT).

Simulations runs are all started with an empty fab where machines are up and idle. To allow the construction of confidence intervals (Law and Kelton 1982), statistics are collected in job batches of 20, of which the first two are ignored. Statistics are gathered for at least 26000 jobs in the first fab configuration FabGla, and 1200 jobs for the other configuration Fab1.

Tables 1 and 2 contain the details of FabGla. In short, the first configuration corresponds to a simple hypothetical fab formed of four workstations that are used to fabricate two products with distinct routes. Each route cycles three times through workstation 1 which is also the bottleneck. A start ratio of N/M is defined to mean that for every N starts of product 1 there are M starts of product 2. Three sets of simulations with three different product mixes are conducted; the start ratios are 1/2, 5/8, and 2/5 respectively. The simulation results using the five dispatching rules described earlier are shown in Figures 1, 2, and 3, for start ratios 1/2, 2/5, and 5/8 respectively. Every data point in the figures corresponds to a simulation: it represents the sum of queue sizes cumulated over all workstations and graphed against the bottleneck utilization.

From Figure 1, it can be seen that, for a starts ratio of 1/2, BQP outperforms all other dispatching rules over the 90%-100% range of bottleneck utilization, with FIFO a close second. At 99.7% utilization the average total inventory for BQP is 7.0% lower than that of FIFO and 14.2% lower than that of SRPT. The differences in the coefficients of variation (CV) are negligible.

Table 1: Description of Equipment in the FabGla Model

Workstation Number	Number of Machines	MTTF	MTTR
1	3	900	100
2	2	700	100
3	2	1500	100
4	2	1350	150

Table 2: Description of the Products' Routes for FabGla			
Route of Product 1		Route of Product 2	
Workstation Number	Processing Time	Workstation Number	Processing Time
2	36	3	31
1	20	1	25
4	120	2	29
1	25	1	20
2	29	3	34
1	25	1	25

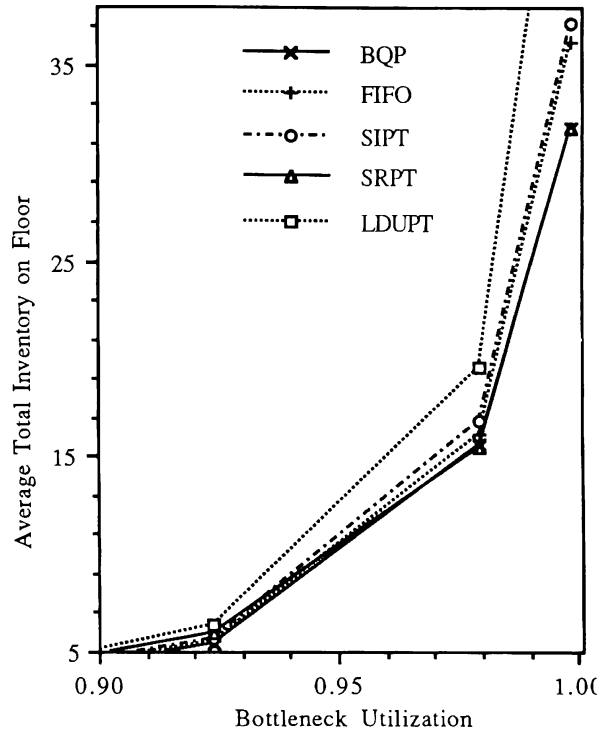

Figure 2: Results for FabGla with Starts Ratio 2/5

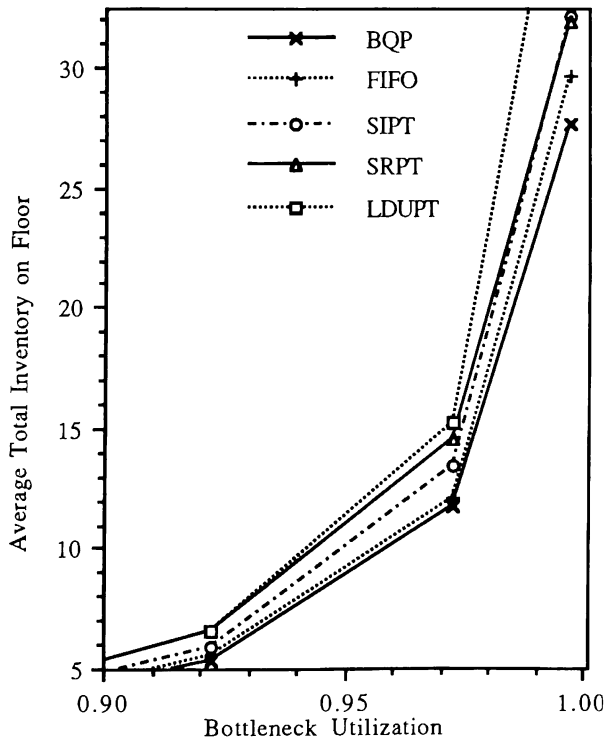

Figure 1: Results for FabGla with Starts Ratio 1/2

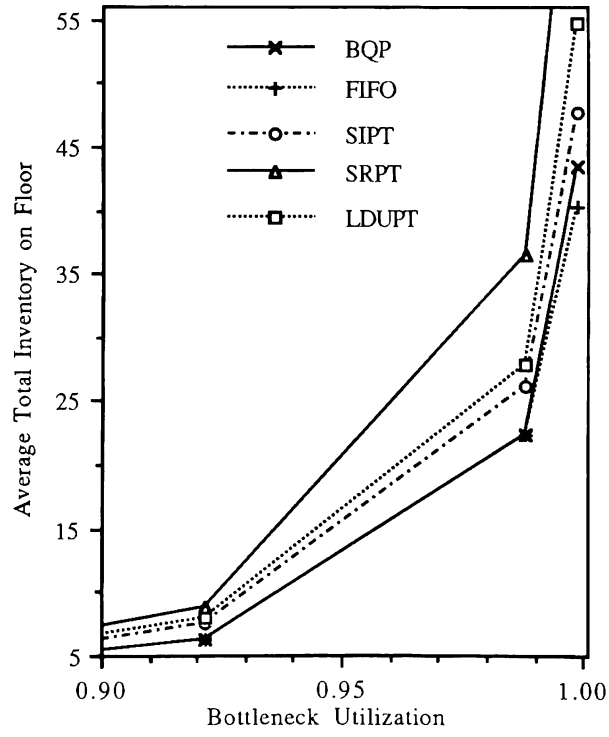

Figure 3: Results for FabGla with Starts Ratio 5/8

For a starts ratio of 2/5, Figure 2 shows that at 99.9% utilization, the total average inventory for BQP is the same as SRPT and 13.2% lower than that of FIFO. From simulations ran outside the range of the graph it can be observed that BQP outperforms all other dispatching policies for bottleneck utilizations lower than 92%. SRPT gave good results under high utilzation because it tends to favor product 2 which has higher starts here.

When the starts ratio is 5/8 thus becoming less favorable for product 2, Figure 3 shows that SRPT is a disaster. BQP outperforms all other dispatching rules throughout most of the bottleneck utlization range. At 99.9% utlization though, the average total inventory for BQP is 7.7% higher than FIFO's. Again the differences in the CVs are small.

To illustrate the effect that dispatching rules have on the product cycle times, the average waiting times of product 1 and 2, as well as the overall average waiting time, are shown in Table 3 for a starts ratio of 1/2 and a utilization of 99.72%. The results included are for BQP, FIFO, and SRPT. It can be observed that BQP gives better results than FIFO for both products. On the other hand, SRPT does very well for product 2, but its overall performance is inferior to that of the two other dispatching policies.

Table 3: Effect of Dispatching Rule on the Products Queuing Times in FabGla with Bottleneck Loaded at 99.72% and Starts Ratio Equal to 1/2.

	BQP	FIFO	SRPT
Average Waiting Time for Product 1	772.9	816.0	2120.8
Average Waiting Time for Product 2	694.1	750.3	184.3
Average Waiting Time Overall	720.4	772.2	829.8

Table 4 shows the dispatching rules' impact on the queue sizes at all the workstations when the starts ratio is 1/2. It also provides information regarding workstation utilizations. By establishing a negative correlation at the bottleneck between lots' arrivals and queue size, BQP succeeds in decreasing the bottleneck's average queue size. On the other hand, BQP tends to increase the non-bottleneck workstation queue sizes because it adds randomness to the lots' arrival process at these workstations. On the average, BQP seems to minimize the overall average queue size, and thus it minimizes the cycle time.

Table 4: Effect of Dispatching Rule on the Workstations' Average Queue Sizes in FabGla with Starts Ratio of 1/2. Queue Sizes are Expressed in Terms of Lots.

Workstation Number	Utilization in %	BQP	FIFO	SIPT
1	96.73	6.941	7.89	7.766
2	87.70	2.979	2.576	2.327
3	86.51	1.242	1.087	1.355
4	83.13	.6249	.5232	1.940
Queue Sizes Cumulated		11.79	12.08	13.39

Product mixes have often been chosen in such a way that workstations 2, 3, and 4, are utilized at levels comparable to the bottleneck's. For many simulations, such as when the starts ratio is 2/5 and the bottleneck utilization rate is 97.94%, or when the starts ratio is 5/8 and the bottleneck utilzation is 84.26%, some workstation utilzations are within 1% of the bottleneck's. Thus the results which have been obtained here are actually valid for fabs with more than one bottleneck.

The second fab configuration used for simulations,

Fab1, is also a hypothetical fab with ten workstations. Only one product is fabricated using a nineteen steps route that requires a lot to visit seven times the bottleneck before its completion. For every dispatching policy, eleven simulations were run, each using a different bottleneck utilization rate. The results are displayed in Figure 4 for simulations in which the utilization rate is 75% or higher. At 99.86% utilization, the average total inventory for SRPT is 19.7% lower than that of BQP, but for a lower utilization such as 89.96% it is lower by only 3.7%. Overall, BQP ranks second for Fab1 with the three remaining rules far behind.

Notice that SRPT which does so well for the one-product Fab1 has inferior results for the two-product FabGla. Also, FIFO which is often a close second in FabGla is a distant third in Fab1.

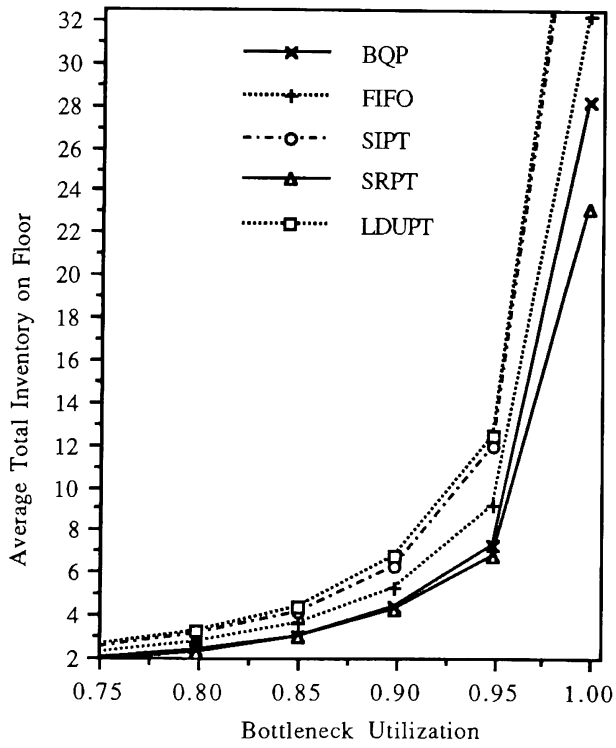

Figure 4: Results for Fab1

6. CONCLUSION

This paper has presented the Bottleneck Queue Prediction (BQP) dispatching policy which is based on the concept of bottleneck starvation avoidance and relies on frequently updated queue size projections and lead time estimates. This approach assumes the availability in the wafer fab of a CIM system that provides data describing the factory floor. The transformation of large amounts of data into meaningful and useful information is also discussed. The need for predictions and global information is particularly emphasized. In addition, the advantage of make decisions in real-time as opposed to using periodic based methods is discussed.

The BQP policy aims at minimizing the average queue size in front of the bottleneck; it gives higher priority to lots that are expected to encounter a smaller queue at their next visit there. The net effect of BQP is that it tends to spread evenly over time the arrival process at the bottleneck. To calculate lots arrival times at the workstations heavy usage is made of the maintained and dynamically updated queue size projections. BQP is indeed a computationally intensive method that has to be implemented carefully so that it makes effective usage of computer time and memory. Running a simulation of FabGla, where over 33,000 jobs are completed, takes approximately two hours in real-time when BQP is used as opposed to half an hour for standard rules.

When compared with other dispatching rules BQP has shown a consitantly good behaviour. In some cases it is top rated, in others it is a close second. But more importantly, BQP has never performed poorly thus indicating that it is indeed a very robust policy. The same can not be said about any other dispatching rule tested here: the ones which perform well for some fab configurations are disastrous for others. BQP's main advantage is that it is based on reliable factory-wide information and future predictions. Since all simulations involved fab configurations where machines are highly unreliable, the results obtained here are valid in a stochastic environment. BQP circumvents randomness by updating all estimates at the advent of changes. For example, the effect of a machine failure is to reduce the workstation's capacity and possibly cause an increase in the queue size estimates for a number of time periods. This in turn leads to longer waiting time and lead time estimates for lots that are expected to arrive at the workstation during the affected time periods.

BQP represents an attempt to develop a dispatching policy that taps on the large amount of information provided by CIM systems in manufacturing. Although the simulation experiments testing BQP have given consistantly favorable results, this approach still needs to be refined to reach its full potential. As dicussed earlier, experiments have shown that lot release policies

have more effect on fab performance than dispatching. Thus, the application of the concepts presented here to lot release should yield more dramatic improvements.

ACKNOWLEDGEMENTS

This research was supported in part by the Semiconductor Research Corporation, IBM Corporation, NCR Microelectronics, Texas Instruments, and the State of California MICRO program.

APPENDIX A: EXAMPLE OF AN ALGORITHM

When a lot is started, lead time estimates for the new lot are constructed and projections of future workstation queue sizes are updated. The algorithm used to perform these calculations is presented next. Some notations and definitions are introduced first. Time is always measured with t = 0 as the current time. For certain calculations, time is divided into periods with size Δt units. Period N is defined to be the time horizon beyond which no predictions will be made, N_k is the last time period for which projections have been made for workstation k ($N_k < N$ for all k), and $N_{k,safe}$ is the time period until which it can be safely assumed that capacity stays unchanged for workstation k. Let F_{js} be the estimate of the time it would take lot j to reach, given its present status, the workstation where step s is to be performed. Now, let S_j be the last step number in lot j's recipe, k(j,s) the workstation where step s for lot j is to be done, and w_{js} the expected waiting time for lot j in front of workstation k(j,s). Next, $x_k[n]$ and $q_k[n]$ are defined to be respectively the projected workload from WIP arriving at workstation k during period n and the projected queue of work at workstation k at the end of period n (let $q_k[-1] = 0$). Let I_k be the target inventory of work for workstation k. All work is expressed in terms of machine-hours. Next, define $m_{k,now}$ to be the number of machines available at workstation k currently. Then the variable $m_k[n]$, the estimate of the number of machines available at workstation k during period n, is defined in the following way:

$$m_k[n] = \begin{cases} m_{k,now}, & \text{if } n <= N_{k,safe} \\ m_k \dfrac{MTTF_k}{(MTTF_k + MTTR_k)}, & \text{otherwise} \end{cases}$$

The following is used to estimate the flow times:

step 0: let s = 0
let $F_{js} = 0$

step 1: if $s \geq S_j$ go to step 4

step 2: let n be an integer s.t. $n\,\Delta t \leq F_{js} < (n+1)\,\Delta t$
let k = k(j,s)

$$w_{js} = \begin{cases} I_k, & \text{if } n > N_k \\ \dfrac{(x_k[n] + q_k[n-1])}{m_k[n]}, & \text{if } mk[n] > 0 \text{ and } n <= N_k \\ x_k[n] + q_k[n-1] + \Delta t\, N_{k,safe}, & \text{otherwise} \end{cases}$$

step 3: let s = s + 1
let k' = k(j,s)
let $F_{js} = F_{j,s-1} + w_{j,s-1} + p_{j,s-1} + T_{k,k'}$
go to step 1

Next, queue size predictions are updated:

step 4: let s = 0

step 5: let n be an integer s.t. $n\,\Delta t \leq F_{js} < (n+1)\,\Delta t$
let k = k(j,s)

step 6: if $n \geq N$ go to step 10

step 7: if $n > N_k$ set $N_k = n$ and expand q_k and x_k
let $x_k[n] = x_k[n] + p_{js}$

step 8: let $q_k[n] = \max\{0, x_k[n] + q_k[n-1] - \Delta t\, m_k[n]\}$
let n = n + 1

step 9: if $q_k[n-1] > 0$ and $n \leq N_k$ go to step 8

step 10: if $s < S_j$ then set s = s + 1 and go to step 5

REFERENCES

Blackstone Jr., J. H., Phillips, D. T., and Hogg, G. L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research* **20**, 27-45.

Ehteshami, B. and Rohani, D. (1989). Wafer fabrication automation. Private communication.

Glassey, C. R. and Adiga, S. (1989). Conceptual design of a software object library for simulation of semiconductor manufacturing systems. *Journal of Object-Oriented Programming*, September/October.

Glassey, C. R. and Resende, M. G. C. (1988). Closed-loop job release control for VLSI circuit manufacturing. *IEEE Transactions on Semiconductor Manufacturing* **1**, 36-46.

Hughes, R. A. and Shott, J. D. (1986). The future of automation for high-volume wafer fabrication and ASIC manufacturing. In *Proceedings of the IEEE* **74**, 1775-1793.

Law, A. M. and Kelton, W. D. (1982). *Simulation Modeling and Analysis*. McGraw-Hill, New York.

Leachman, R. C., Solorzano, M., and Glassey, C. R. (1988). A queue management policy for the release of factory work orders. Research Report 88-19, Engineering Systems Research Center, University of California at Berkeley.

Lou, S. X. C. (1989). Wafer fabrication scheduling. Private communication.

Lozinski, C. and Glassey, C. R. (1988). Bottleneck starvation avoidance indicators for shop floor control. *IEEE Transactions on Semiconductor Manufacturing* **1**, 147-153.

Martin-Vega, L. A., Pippin, M., Gerdon, E., and Burcham, R. (1989). Applying just-in-time in a wafer fab: a case study. *IEEE Transactions on Semiconductor Manufacturing* **2**, 16-22.

Resende, M. G. C. (1987). Shop floor scheduling of semiconductor wafer manufacturing. Research Report 87-1, Engineering Systems Research Center, University of California at Berkeley.

Stidham, S., Jr. (1974). A last word on $L = \lambda W$. *Operations Research* **22**, 417-421.

Sullivan, G. and Fordyce, K. (1989). Logistics Management System (LMS): implementing the technology of logistics with an advanced decision support system. Unpublished paper.

Wein, L. M. (1988). Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing* **1**, 115-130.

Wolff, R. W. (1989). *Stochastic Modeling and the Theory of Queues*. Prentice Hall, New Jersey.

AUTHORS' BIOGRAPHIES

C. ROGER GLASSEY is a professor and former chairman of the Industrial Engineering and Operations Research department at the University of California at Berkeley. He received the B.S. degree in mechanical engineering from Cornell University in 1957, and completed a graduate program in industrial administration at the University of Manchester, England, as a Fulbright scholar. He received the M.S. degree in applied mathematics from the University of Rochester in 1961, and the Ph.D. degree in operations research at Cornell in 1965. He worked for the Eastman Kodak Company, first as an industrial engineer, then in the corporate operations research group. His primary research interests are production planning and scheduling and mathematical optimization.

Professor C. Roger Glassey
Industrial Engineering and Operations Research Dept.
4135 Etcheverry Hall
University of California
Berkeley, CA 94720, U.S.A.
(415) 642-4997

RAJA GEORGES PETRAKIAN is a Ph.D. student in the Industrial Engineering and Operations Research department at the University of California at Berkeley. He received the B.S. and M.S. degrees in systems engineering from Case Western Reserve University in 1985 and 1986 respectively. He was an intern at Intel Corporation during the summer of 1988 working as an information system analyst in the central planning group. He was awarded an IBM Manufacturing Research Graduate Fellowship for the 1989-1990 academic year. His current research interests include production control, queuing theory, and expert systems.

Raja G. Petrakian
Industrial Engineering and Operations Research Dept.
4135 Etcheverry Hall
University of California
Berkeley, CA 94720, U.S.A.
(415) 642-8255