Closed-Loop Job Release Control for VLSI Circuit Manufacturing

C. ROGER GLASSEY AND MAURICIO G. C. RESENDE

Abstract—**This paper introduces a closed-loop job release mechanism for job shops where the main source of randomness is due to machine failure and repair. The release policy adapts concepts of the reorder point method of inventory control to the context of job shop scheduling. The control mechanism, called starvation avoidance, is compared empirically with other input control mechanisms on several semiconductor wafer manufacturing job shops with favorable results.**

Key Words—**Discrete event simulation, job shop scheduling, job release control, production planning, VLSI circuit manufacturing.**

I. INTRODUCTION

THE MANUFACTURING of very large scale integrated (VLSI) circuits [10], [19] is perhaps the most complex manufacturing process found today. This complexity is, in part, the result of constant device miniaturization, process intricacy, product diversity, uncertainty, and changing technologies. Aside from the obvious technical challenges that circuit designers face, semiconductor wafer fabrication offers a wide range of complex issues related to production planning and control. The introduction of computer-integrated manufacturing (CIM) systems in the fabrication process has added a new dimension of problems [21].

In this paper we are concerned with the problem of shop floor control in the first stage of VLSI manufacturing, in which many devices are simultaneously built up on a wafer of silicon. It takes place in a clean room environment known as a semiconductor fab. Partly because of contamination problems, some of these facilities are among the few existing examples of *paperless factories*, in which the status of machines and inventories is available in real-time from the factory CIM system. Hence, real-time decisions about *dispatching* (which job to do next when a machine becomes available) and *lot release* (of raw wafers into the factory) could be based on the global factory state. The possibility of making decisions based on an expanded information set raises two research questions:

1) How should global information be summarized and used for decisions?

2) How much improvement can one expect as a result, compared with decisions based on local information?

In particular, we consider in this study a special class of semiconductor production facility: High-volume fabs with output limited by machine capacity rather than market conditions or operator availability and in which the main source of randomness is due to machine failure and repair. Other scheduling issues exist in semiconductor manufacturing, but are not treated here. These include: operator scheduling, scheduling of batch operations with long cycle times, and scheduling with setup or changeover times.

There exists no consensus as to what should be the objective of a scheduling policy. One natural objective is the minimization of average cycle time, given a target average throughput rate. Cycle time is defined to be the time a job spends in the fab, i.e., processing time plus waiting time, while throughput rate is the average number of jobs that leave the fab per unit time. Minimization of cycle time, and consequently, waiting time, among many effects, decreases the time a wafer is exposed to particles in the clean room, thus increasing yield, improves the response time to changes in the demand pattern, reduces in-process inventory, and reduces the engineering response time. It also reduces many hidden costs, e.g., production control personnel costs and production losses due to hot-lot dispatching. Possibly the most significant cost of high inventory levels is their contribution to low yield. Some machine failures result in process malfunctions that are not detected until the final test, so everything that has passed the failed machine in the interim may have to be scrapped.

Scheduling policies can be compared based on the mean delay/mean throughput tradeoff curve. This curve $D(t)$ describes the average delay or waiting time as a function of the fab throughput t. It is well known that as throughput approaches the capacity of the fab the average queueing time goes to infinity.

We say that a scheduling policy S_A is superior to policy S_B for a given throughput interval T if $D_A(t) < D_B(t)$ for $t \in T$, where $D_A(t)$ and $D_B(t)$ are the tradeoff curves for policies S_A and S_B, respectively (see Fig. 1). Our objective is to find decision policies that will be efficient on the frontier of delay (or inventory level) and throughput. By Little's Law [15], the expected inventory of work await-

Manuscript received July 1, 1987; revised September 4, 1987. This work was supported in part by the Brazilian Council for the Development of Science and Technology (CNPQ), Semiconductor Research Corporation, Philips Research Laboratories, Fairchild Semiconductor Corporation, Cray Research, and the State of California through PROJECT MICRO—Microelectronics Innovation and Computer Research Opportunities.

The authors are with the Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720.

IEEE Log Number 8717623.

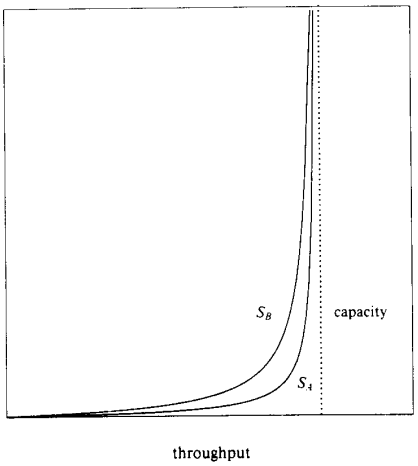


Fig. 1. Delay/throughput tradeoff curve.

ing processing (WAP) equals *average delay* × *flow rate*. Consequently, for a fixed output rate, reducing WAP reduces time spent in the shop. However, reducing WAP will tend to reduce throughput by: 1) reducing the buffering between work stations so that machine downtime at any station will have a high probability of forcing idle time elsewhere due to lack of work; and 2) reducing batch sizes so a greater fraction of machine time is spent on setups. In this paper we are concerned only with the first tradeoff.

It is well known that randomness complicates scheduling. Equipment availability, which is the main source of uncertainty considered here, can be improved by increasing equipment reliabilty and reducing the mean time to repair. This does occur in practice. However, new pieces of equipment are constantly introduced as the state of manufacturing art advances and these are often unreliable. Thus, stochastic scheduling still remains a major requirement for semiconductor fabrication.

Selection of a job from a queue at a work station is often called job shop *dispatching* and has been studied extensively in the literature. For surveys of the field we refer the reader to [26], [27], [11], [18], [5], [20], and [2]. Stochastic queueing networks have also been the subject of much study, e.g., [9], [13], [22], [25], and [31].

Production planning and control of VLSI manufacturing have only recently attracted the attention of operations researchers. Lohrasbpour and Sathaye [16] describe a simulation model of integrated-circuit (IC) fabrication used to estimate work-in-process (WIP), cycle time, production rates, and labor and equipment utilization, given a set of input parameters, e.g., product mix and wafer input rate. The authors discuss several applications of the model, including capacity analysis, analysis of variability of wafer input rate, effect of express (hot) lots, WIP reduction policies, and the effect of equipment availability. They observe that by reducing the variability of input a considerable reduction in cycle time can be measured and conclude that a uniform rate of starts (release of raw wafers into the fab at equal time intervals) is a highly good policy. Dayhoff and Atherton [7] describe the dynamics of wafer movement and identify several problems that can be analyzed by discrete-event simulation, e.g., fab startup, clearing of accumulated inventory, and the analysis of cycle time/throughput tradeoff. Resende [23] describes a public domain discrete-event simulator for semiconductor wafer manufacturing. Leachman [14] presents a linear programming based corporate level planning model for the semiconductor industry. Lozinski [17] identifies technological constraints that should be satisfied by a VLSI circuit fab scheduler. Wein [30] compares several scheduling policies on three fictitious fabs (using data gathered at a Hewlett-Packard research fab). Several of the scheduling rules used by Wein are derived by approximating the fab by a subnetwork of queues restricted only to stations that are heavily loaded and use a Brownian network model that approximates a multiclass queueing network model with dynamic control capability. He observes that job release plays a more important role in reducing cycle time than do dispatching rules. He discusses one such release mechanism that outperforms uniform starts in his simulations. Chen *et al.* [4] present a class of queueing network models for the analysis of wafer fabrication facilities. Using these models on data gathered during several years at a real VLSI circuit fab, they are able to predict key performance measures within about 10 percent of the values actually observed. Burman *et al.* [3] discuss how operations research tools can be used to analyze the behavior of VLSI circuit manufacturing lines. In particular, they present simulation, queueing, and deterministic models and suggest when it is appropriate to use each class of models. Spence and Welter [28] use simulation to study the fab cycle time/throughput tradeoff curves with respect to the sensitivity of changes in the operation of a photolithography cell. In particular, they examine the effects of adding resources (operators and equipment), process improvement (reducing setup times and rework), and operating rules (lot sizing and repairman wait time). Watts [29] discusses the integration of scheduling in the CIM environment. Composite dispatching, a dispatching scheme for load balancing, is proposed. This scheme partitions the work-in-process by current mask level. The priority of a lot is inversely proportional to the number of lots in its mask level set, i.e., the lot selected is from the mask level set with the least number of members.

In this paper we present a closed-loop job release control policy, called starvation avoidance (SA), that in simulation experiments compares favorably to traditional scheduling policies with respect to the delay/throughput tradeoff. The plan of the paper is as follows. In Section II we describe a queueing network model on which we base our analysis. An implementation of this model, with which we carry out our experiments, is briefly described. In Section III we describe SA, a control policy that attempts to maintain a *virtual* inventory of work to be done at the bottleneck station above a given safety stock level.

In Section IV we review other job release policies. In Section V we compare SA with other scheduling policies on several fab queueing networks using our simulator, FabSim, on the Cray X-MP/14 supercomputer at UC Berkeley. Conclusions and future research are outlined in Section VI.

II. Fab Queueing Network Model

We describe a queueing network model on which we carry out our analysis of the performance of scheduling policies. This model is fully implemented in the discrete event simulator FabSim [23].

A fab is the physical location where the most sensitive portion of VLSI circuit manufacturing takes place. For the purpose of our analysis let the fab be defined as a set of M work stations, each having m_i ($i = 1, \cdots, M$) identical parallel unreliable machines. Machines of work station i have time available and time to repair that are distributed according to probability distribution functions $G_u(i)$ and $G_d(i)$, respectively. Each work station has associated with it a buffer of infinite capacity, from which lots of wafers are selected for processing at that station. Machines can be in one of three states—*idle*, *down*, or *busy*. When a machine is in working condition it is either *idle* or *busy*. Machine i can process at most c_i lots simultaneously if it is in working condition and cannot process if it is *down*. Equipment failure in nonpreemptive, i.e., equipment, will complete processing of a lot before it is allowed to go down. At time $t = 0$ all machines are assumed *idle* and lot inventory in the fab is zero.

The fab produces p products. Each product p_i ($i = 1, \cdots, p$) has associated with it a *process recipe* (or simply a *recipe*). A recipe defines a sequence of work stations through which a lot must flow during its fabrication process, and for each work station, the duration of processing, inter-station travel time probability distribution, and the lot rework probability. The recipe also defines the lot start rate and the proportion of high-priority lots of that product to be started. A lot release mechanism triggers lot starts. A lot, upon arriving at a work station, is placed in a buffer of lots waiting for processing at that station. If there are one or more lots waiting in this queue, the dispatcher must decide which lot or lots to process next when a machine becomes available. At the end of processing at a station, the lot is placed in travel status toward its next processing station. The time it remains in travel status is a random variable specified by the recipe.

Shop floor scheduling control involves a set of decisions. Lot release decisions determine when a new lot of work is to be fed into the fab. Lot dispatching decisions control what lot is to be processed next when a machine becomes available. A common way to dispatch lots is with the use of a priority dispatching rule. Denote the set of $n \geq 0$ lots queued at station s by J_s. A priority dispatching rule r is defined uniquely by a priority function $P_r: J_s \rightarrow \boldsymbol{R}$ that assigns real values to elements of J_s. A *normalized* priority function is a priority function $\overline{P}_r: J_s \rightarrow [0, 1]$ that assigns real values between 0 and 1 to elements of J_s. Rule r selects the lot with the smallest priority function value, i.e., the lot with the highest priority. If a batch operation is called for and more than one lot is to be selected, rule r is repeatedly applied to a feasible subset of the remaining queued lots until the required number of lots are chosen.

FabSim[1] [23] is a C-language implementation of the model described above, using exponential distributions for equipment time to repair and time between failures. Inter-station travel time can be either zero or distributed exponentially. Exponential distributions are used with the objective of introducing randomness into the system. For this purpose, any other probability distribution could be used. FabSim allows numerous dispatching and release control combinations, including optimal multirule dispatching with pattern search optimization [12]. Detailed statistics are collected and displayed at the end of the simulation run.

III. Job Release by Starvation Avoidance

Some features of wafer fabrication are not found in most other job shops. Because each layer of the wafer requires exposure of a photoresistive material through a mask (the process is known as photolithography) each batch of wafers makes many visits to the photolithography work station before it is completed. Since this equipment is very expensive, this work station is often the bottleneck. The special feature of work flow in wafer fabrication is that it is reentrant at a bottleneck work station. A scheduling policy should focus on the queue of work at the bottleneck because it will on average be the biggest queue in the factory, and furthermore, whenever the bottleneck work station is forced to become idle due to lack of work, that lost time represents an irretrievable loss of final output. A desirable scheduling policy is one that allows high utilization rates of bottleneck machines while maintaining low levels of work-in-process. At first, these objectives may appear contradictory since increasing bottleneck utilization will cause work-in-process and mean delay also to increase. The rate of increase, however, will depend on the particular scheduling policy in use, since, as we will see in Section V, some scheduling policies are more delay-sensitive to changes in throughput rate than others. We seek a policy with less sensitivity, i.e., a policy that outperforms others in terms of the mean delay/mean throughput tradeoff.

The literature of job shop scheduling makes little, if any, reference to job release control. Most research has assumed job release to be random, commonly a Poisson process. Recently, some indication has surfaced regarding the benefits of release control in the context of job shops with unreliable machines. Burman *et al.* [3] observe that there is much to be gained by reducing the variability of inter-arrival times of jobs. Wein [30] suggests

[1]FabSim can be obtained from the Earthquake Engineering Research Center, 404A Davis Hall, University of California, Berkeley, California 94720, at a nominal charge.

that job release control plays a much more significant role than dispatching in terms of reducing mean job delay. In a paper reviewing performance analysis modeling for manufacturing systems, Fredericks [8] suggests that controlling the number of lots in the pipeline from start to the bottleneck resource can lead to a reduction in work-in-process when compared to the case where no control takes place. Fredericks, however, gives no details of how this could be accomplished.

In this section we present a class of job release control mechanisms that we call starvation avoidance (SA), designed to maximize utilization of critical bottleneck equipment, while at the same time controlling the growth of work-in-process.

We make the following assumptions about the production system.

1) All random variables (e.g., time between machine failures, time to repair, and fraction of rework) are stationary.

2) The desired output rate is constant (at least for intervals long compared to mean flow time).

3) A single work station is the unique bottleneck for the shop and machine time is the limiting resource. In other words, there is a single work station, called the *bottleneck*, that has the minimum expected idle time. For any work station j, the proportion of idle time is given by

$$I_j = 1 - F \times \frac{W_j}{N_j A_j} \tag{1}$$

where F is the average flow rate of new work into the shop (wafers per hour), W_j is the work load (machine hours) at station j per wafer, N_j is the number of machines at station j, and

$$A_j = \frac{(MTBF_j - MTTR_j)}{MTBF_j} \tag{2}$$

is average machine availability, where $MTTR_j$ and $MTBF_j$ are mean time to repair and mean time between failures of machine j. We argue that this assumption is not restrictive because an exactly balanced shop is very difficult to achieve in practice. Machines can only be purchased in integer quantities from among a very limited set of production rates. Average machine reliability changes slowly over time as do workloads, so even if the shop were designed to be exactly balanced, it would not remain balanced. The concept of a bottleneck work station is the key to the analysis. Idle time at the bottleneck represents a permanent loss of factory output, and output cannot be increased by trying to keep nonbottleneck stations busy.

4) A single product is manufactured. This assumption is merely to simplify the notation and is otherwise irrelevant.

The central idea behind SA is simple: To reduce inventory, do not start new work. The difficulty with that idea is that eventually the bottleneck work station starves and no finished work leaves. Hence, we are lead to the *starvation avoidance* rule: Start new work just in time to avoid idling the bottleneck work station due to lack of work. An analogy can be made between work content of the queue at the bottleneck station (in the context of job shop scheduling) and inventory (in the context of inventory control). In inventory control the main objective is to optimize the tradeoff between inventory holding costs and the costs of stock outs. If demand and delivery lead time (the time lag between order and delivery) are deterministic this task is trivial. However, in practice both demand and supply (delivery lead time) are random. Protection from the uncertainty comes from the concept of safety stock. In a reorder point inventory control system (see Fig. 2), whenever on hand inventory plus the quantity on order but not yet delivered falls below the safety stock level, an order is placed to bring the total up to the safety stock level. Provided the safety stock level is big enough, the next order will arrive in time to avoid running out of stock, with sufficiently high probability.

Let T_E and T_R be the times to exhaust and replenish stock, respectively. The expected time to exhaust stock $E(T_E)$ can be calculated by dividing the physical inventory I by the demand rate $\bar{d}$

$$E(T_E) = I/\bar{d}. \tag{3}$$

If $T_E < T_R$, a stock out occurs. In an attempt to avoid stock outs a reorder is made whenever

$$I < \bar{d} \times T_R + ss \tag{4}$$

i.e., the inventory falls below the total demand during the time to replenish plus an amount of safety stock to safeguard against uncertainty. This occurs when

$$E(T_E) < \alpha \times T_R \tag{5}$$

where $\alpha > 1$. This way a reorder policy can be based on times to exhaust and replenish inventory.

In a job shop, the variability of physical inventory at the bottleneck results from the interaction of random demand (induced by the failure and repair at the bottleneck work station), the uncertain lead time required for raw wafers newly injected into the system to reach the bottleneck, and the arrival of work already in process from other work stations (a phenomenon not present in standard inventory models). We define the concept of *virtual inventory* to be the work content (measured in work hours at the bottleneck) of all jobs either at the bottleneck work station or expected to arrive at the bottleneck within a given lead time. We take as lead time for replenishment an estimate of the time required for a job to arrive at the bottleneck for the first time, once it has entered the shop. We account for the current repair state of machines at the bottleneck work station by including an estimate of the time to repair any failed machines as part of the virtual inventory. As in inventory control, whenever the virtual inventory falls below a given (safety stock) level, a new job is released.

Effective flow control requires a sufficiently large inventory of raw material so that new work can be introduced whenever desired. While such policies shift inven-

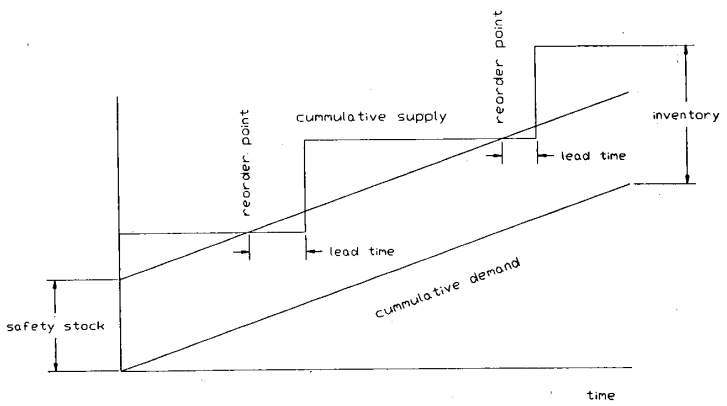


Fig. 2. Reorder point inventory control.

tory from WAP to raw material, it is much less expensive to hold inventory in that form. The raw wafers can be made into any product, so the risk of obsolescence due to demand shifts or engineering changes is lower. They are not as subject to contamination and yield loss.

The objective is to make the new job arrive at the bottleneck *just in time* to avoid equipment starvation there. The safety stock level is the control parameter for the system. Increasing this number will increase average inventory levels but will reduce the probability of starving the bottleneck due to lack of work and hence increase average output of finished wafers. By varying the safety stock level, one can generate a tradeoff curve $D(t)$ of delay versus throughput.

Let us now formalize the notion presented above for the case in which the job shop has a single bottleneck resource and a single product. Several other cases are described in [24]. Denote the bottleneck station by B. Station B has m unreliable machines with mean times to repair $MTTR_B$.

Let K_i be the number of jobs currently at step i (either queued or in process). The i^{th} step has an associated work station w_i and processing duration d_i. Let i_0 be the step number corresponding to the first visit to station B

$$i_0 = \min\,(i \,|\, w_i = B). \tag{6}$$

Let S_B be the set of bottleneck work station steps

$$S_B = (i \,|\, w_i = B) \tag{7}$$

and let F be the set of steps prior to the first visit to station B

$$F = (1, \cdots, i_0 - 1). \tag{8}$$

Define L to be the total processing time from start (step 1) to the first visit to station B, excluding step i_0

$$L = \sum_{i=1}^{i_0-1} d_i. \tag{9}$$

Let n_i be the step number of the next visit to station B given that the job is presently at step i. Furthermore, let P be the set of steps such that total processing duration of that step plus future steps prior to the next visit to station B is less than L

$$P = \left(i \,\middle|\, \sum_{j=i}^{n_i-1} d_j < L\right). \tag{10}$$

Let $Q = F \cup P \cup S_B$ be the set of critical steps, i.e., steps that are within a critical time factor of the bottleneck station or steps in the pipeline from start to the first visit to B. Let $N(B)$ be the number of machines under repair (down) at the bottleneck work station. We estimate the total repair time of equipment at the bottleneck to be

$$R = MTTR_B \times N(B). \tag{11}$$

If repair times are exponential, this is the conditional mean total repair time of all down machines. Let the *virtual inventory* W be the total work content at the next visit to the bottleneck station of jobs at steps in set Q plus the estimate of total repair time of equipment down at the bottleneck R scaled by the number of machines at the bottleneck station m

$$W = \left(R + \sum_{i \in Q} K_i d_{n_i}\right) / m. \tag{12}$$

If W falls below a critical value αL, where $\alpha > 0$ is a safety factor, there is danger that the bottleneck will starve, so SA release triggers a job start. The main objective of the SA release strategy is the attainment of an efficient tradeoff frontier of idle time at bottleneck station machines and inventory level. A particular operating point on this tradeoff curve results from the choice of α.

In the description of the release policy we make no mention of dispatching policies for lots at individual stations. It is likely that local dispatching may affect SA. For example, a dispatching policy that makes an intentional effort to delay processing of jobs in the initial pipeline steps, $1, 2, \cdots, i_0 - 1$, such as the shortest remaining processing time (SRPT) rule, will make a negative contribution to SA release when there is danger of bottleneck starvation. An ideal dispatching scheme would aid SA re-

lease in achieving its objective, by giving priority to jobs headed for the bottleneck station when that station is in danger of starving, and by giving priority to other lots when there is no imminent danger of starvation. We present such a rule next.

The main characteristic of this SA-booster dispatching rule is its dynamic behavior. The rule combines two simple rules by means of weights and dynamically changes the weights according to the state of the system. If the bottleneck is in no danger of starvation the dispatching rule gives more weight to the SRPT rule, while if there is imminent danger, more weight is given to a rule (SA^+) that gives high priority to lots that are headed for the bottleneck station.

The SRPT priority rule dispatches the job with the shortest total processing still to be done in the shop. For a job in step i of its process recipe, let its unnormalized SRPT priority function be

$$p_{SRPT} = \sum_{j=i}^{T} d_j \tag{13}$$

where T is the number of steps in the recipe, and let $\bar{p}_{SRPT}$ be the corresponding normalized priority function. Let p_{SA^+} and $\bar{p}_{SA^+}$ be the unnormalized and normalized priority functions of SA^+, respectively. Here, both normalizations are assumed to be by maximum value, i.e., the normalized priority functions are obtained by dividing the unnormalized values by the maximum absolute value the priority functions can return.

Given a lot is at its i^{th} process step, the SA^+ priority function is defined by the ratio of the following estimates:

$$p_{SA^+} = \frac{\text{work content of next bottleneck station visit}}{\text{estimate of time until next bottleneck station visit}} = \frac{d_{n_i}}{\sum_{j=i}^{n_i-1} d_j}. \tag{14}$$

Notice that in (14) we assume travel time to be negligible and that all stations leading toward the bottleneck have sufficient capacity. A factor could be included to account for travel time. Stations leading toward the bottleneck on average have sufficient capacity for they are nonbottleneck, but instantaneously failed machines may exist, leading to uncertainty in the lead time.

Since we require priority functions to be decreasing, we use the complement of $\bar{p}_{SA^+}$, i.e., $1 - \bar{p}_{SA^+}$, as our normalized priority function. This function ($\overline{SA}^+$) assigns high priority to jobs that are close to the bottleneck station and/or that contribute a large amount of work content to the station.

The composite SA dispatching rule is a weighted mix of SRPT and $\overline{SA}^+$. Its normalized priority function is given by

$$\bar{p}_{SA} = (1 - \gamma)\bar{p}_{SRPT} + \gamma(1 - \bar{p}_{SA^+}) \tag{15}$$

where $0 \leq \gamma \leq 1$ is the dynamic weight.

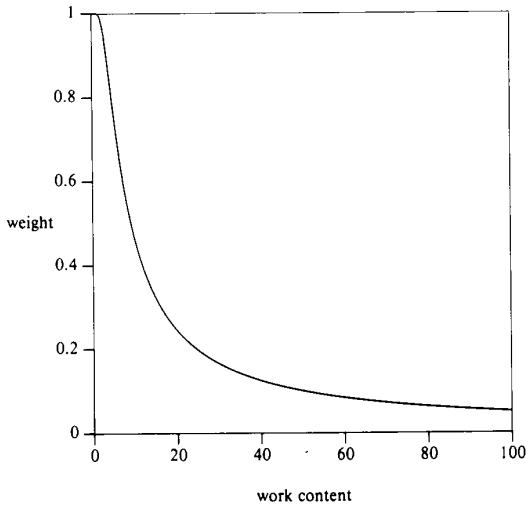


Fig. 3. Dynamic weight γ with $\tau = 10$.

We require the dynamic weight γ to be close to 1 when there is imminent danger of bottleneck starvation (so that $\overline{SA}^+$ will dominate dispatching) and to approach 0 when the bottleneck is not in immediate danger of starving (forcing SRPT to be used to dispatch). This can be achieved in the following way.

Let τ be the critical work content discussed earlier (a constant parameter)

$$\tau = \alpha L \tag{16}$$

and let W be the virtual inventory as defined in (12). By assuming that an infinite supply of new jobs is available for release we can guarantee that the SA release strategy will never allow W to become null.

The dynamic convex weight is defined for all values of $W > 0$, as

$$\gamma = \frac{\{1/1 + e^{-\tau/W} - \frac{1}{2}\}}{\frac{1}{2}}. \tag{17}$$

This definition produces a dynamic weight function having the desired properties, i.e., $\gamma \to 0$ if $W >> \tau$ and $\gamma \to 1$ if $W << \tau$ (Fig. 3 illustrates the dynamic weight function for $\tau = 10$).

Observe that p_{SA^+} may not be defined for all lots. In particular, if the lot is at a bottleneck step or at a step at the end of the process, for which there is no future visit to the bottleneck, the priority function is undefined. For these two exceptions lots are displaced according to SRPT.

IV. Other Release Control Rules

The most common release control used in VLSI fabrication is the open loop strategy of *uniform starts*, i.e., release new work into the shop at a constant rate equal to the desired output rate, and independent of current inventory levels or machine status. A second release strategy is the *Fixed-WIP* rule, which starts a new lot of wafers

whenever a lot of wafers is completed. Wein [30] has proposed a variation of the Fixed-WIP strategy in which inventory is measured, not by counting wafers, but by summing the remaining work to be performed at the bottleneck work station. This strategy releases wafers containing the equivalent of 1 h of work at the bottleneck whenever a bottleneck machine completes 1 h of work.

V. Simulation Results

In this section we compare SA with several other scheduling policies on a number of wafer fab queueing networks.

A scheduling policy is defined by a release control and dispatching policy pair (e.g., UNIF/SRPT is a scheduling policy that uses uniform (UNIF) release control and the SRPT dispatching rule). For each comparison we generate delay/throughput tradeoff curves. Simulations are carried out using FabSim on the Cray X-MP/14 super-computer at UC Berkeley. All queueing networks process a single product and have a single bottleneck resource.

To generate each tradeoff curve requires the simulation of several levels of traffic intensities on each network/policy pair. One approach is to run several *short* simulation experiments with different sets of random number generator seeds. We select a different approach, namely to run a *long* run for each configuration (network/policy). Within each run, statistics are collected in job batches of 50 for the purpose of constructing a confidence interval. For all runs statistics are collected for jobs that are released between simulation clock times $0.05H$ and H, where H, the horizon, is set to accommodate for the desired sample size. In all our runs we collect statistics for at least 1000 jobs and in some cases for over 4000 jobs. Both axes of the tradeoff curve are normalized. Entries in the delay axis are divided by the total recipe processing time. Entries in the throughput axis are normalized so that 1 represents the maximum expected output

$$\frac{N(B) \times A_B}{\sum_{i \in S_B} d_i} \tag{18}$$

of the network. Because of sampling variability it is possible to observe a simulated throughput exceeding this limit.

Table I summarizes the fab networks tested in this study. All networks have the main characteristic of VLSI manufacturing, i.e., work returns periodically to a *hub* station (e.g., photolithography). Small-Fab is a small but representative network with work at hub station S_2 in 5 of the 12 recipe steps. Dayhoff-87 is a hypothetical network described by Dayhoff [6] with work periodically returning to the bottleneck resource (station S_1) in 7 of the 24 steps. Bipolar was suggested to us by Barry [1] of Fairchild Semiconductor Corporation. It is a hypothetical bipolar process fab with the bottleneck resource at photolithography. Of the 46 steps in the process recipe, 7 occur at the bottleneck. Fab networks SSi-implant and SSi-aligner are adapted from data gathered at Silicon Systems. SSi-implant has its bottleneck resource at ion implantation S_{27}, while SSi-aligner has its bottleneck at the aligners S_5. Fabs Small-Fab, Dayhoff-87, Bipolar, and SSi-aligner are similar in that the first visit to the bottleneck resource occurs early in the process recipe. For fab SSi-implant, however, the first visit to the bottleneck station for this fab is at step 21 of the process recipe. A detailed description of the fabs and process recipes is given in Resende [24].

TABLE I
Test Fab Queueing Network

Fab Network Name	Number of Stations	Bottleneck Station	Number of Recipe Steps
Small-Fab	5	S_2	12
Dayhoff-87	18	S_1	24
Bipolar	24	S_4	46
SSi-implant	41	S_{27}	182
SSi-aligner	41	S_5	182

The dispatching rule used in all simulation runs of Small-Fab was SRPT. Four release strategies were compared, namely uniform starts (UNIF), work load regulating (WR), Fixed-WIP, and SA. UNIF releases jobs into the network at fixed intervals. Throughput is controlled by varying the interval duration. WR [30] monitors the sum of the remaining processing time at the bottleneck resource for all jobs in the network. When this sum falls below a critical value a new job is released into the system. Throughput can be controlled by changing the critical value. In Fixed-WIP the number of jobs in the system is maintained constant. A new job is released whenever a finished job leaves the network. Throughput, in this strategy, is controlled by adjusting the WIP level. In SA a new job is released whenever the virtual inventory falls below a critical safety stock level. Throughput is controlled by beginning with $\alpha = 1/m$ (where m is the number of machines in the bottleneck station) and gradually adjusting α until the bottleneck idle time falls to 0 percent.

Within its range of throughput values (94–100 percent of expected capacity) SA release outperformed the other three scheduling policies with respect to the delay/throughput tradeoff. The improvement was most remarkable when compared to UNIF (Fig. 4), where SA mean delay varied from 25 percent to as low as 5 percent of the corresponding UNIF delay. Negligible differences were registered in the coefficient of variation (CV) of inter-departure times. CV of inter-departure times is a measure of the regularity of output.

When compared to Fixed-WIP (Fig. 5) and WR (Fig. 6) improvements were not as marked as when compared to UNIF. Still, near capacity SA maintained its overall good performance with mean delays of 83 and 50 percent of those produced by WR and Fixed-WIP, respectively.

UNIF release with first-in-first-out (FIFO) dispatching was compared to SA scheduling (SA release and SA dispatching) on Dayhoff-87 (Fig. 7). SA scheduling produced schedules having mean delays up to three times shorter than those produced by UNIF/FIFO in the

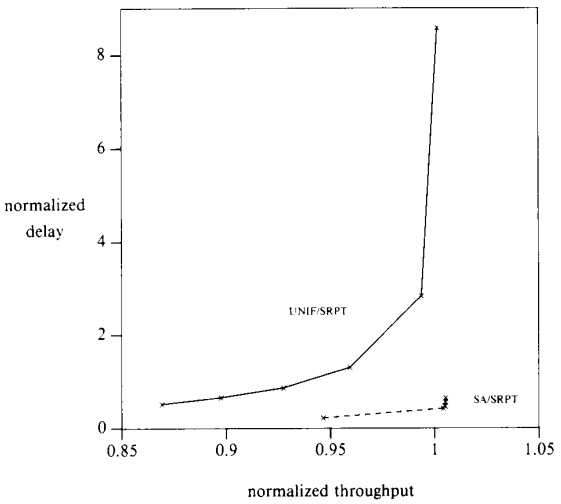


Fig. 4. SA/SRPT and UNIF/SRPT on Small-Fab.

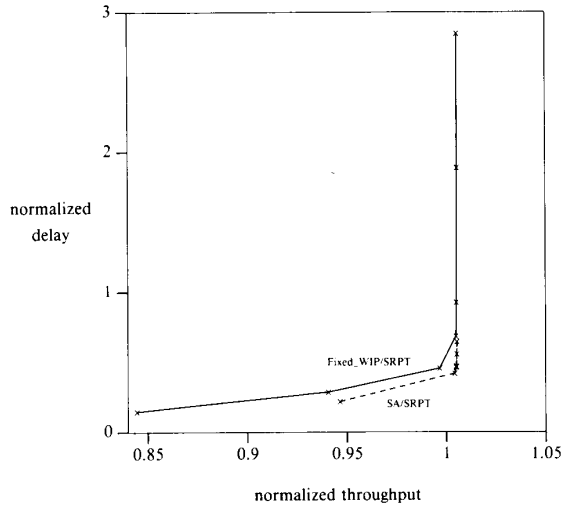


Fig. 5. SA/SRPT and Fixed-WIP/SRPT on Small-Fab.

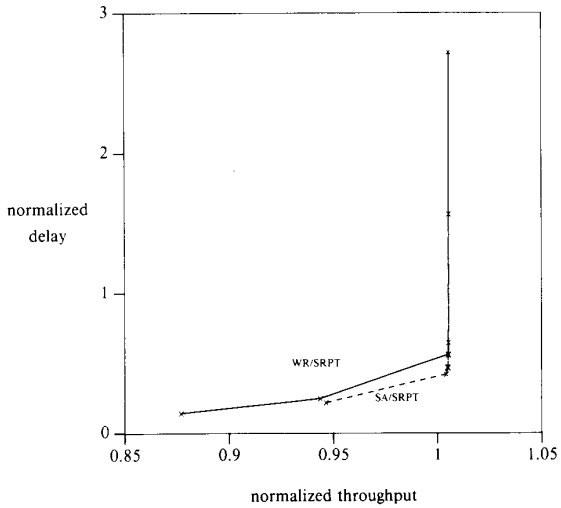


Fig. 6. SA/SRPT and WR/SRPT on Small-Fab.

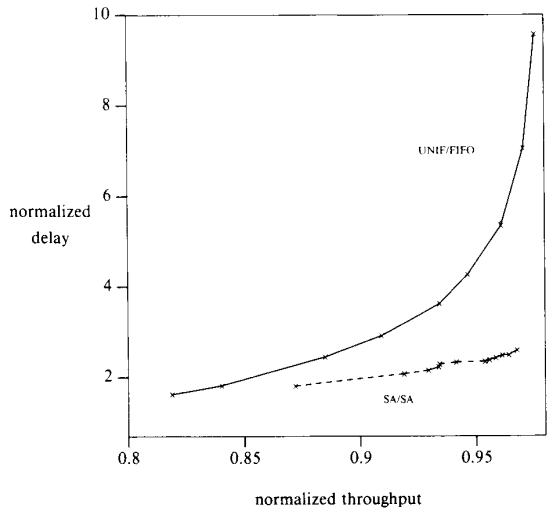


Fig. 7. SA/SA and UNIF/FIFO on Dayhoff-87.

throughput range of SA scheduling (87–97 percent of expected capacity). UNIF/FIFO, however, produced inter-departure times having about 5–8 percent less CV than those of SA scheduling.

SA scheduling was compared to UNIF/FIFO on the Bipolar fab of Barry (Fig. 8). Once again, SA outperformed UNIF/FIFO in its range of throughput values (96–99 percent of expected capacity) with reductions in mean delay varying from 50–73 percent. SA scheduling, however, produced inter-departure time CV up to 30 percent greater than those of UNIF/FIFO.

SA scheduling was compared to UNIF/SRPT on both SSi-implant (Fig. 9) and SSi-aligner (Fig. 10).

In SSi-implant, where, unlike the other examples, the first visit to the bottleneck only occurs at step number 21, SA scheduling only shows a clear improvement over UNIF/SRPT for throughput values greater than 94 percent of expected capacity. Because of sampling variability there were no statistically significant differences between the two policies below that throughput rate. In the high end of this range (i.e., close to capacity) SA is able to produce schedules with approximately 40 percent less delay than those produced by UNIF/SRPT. Inter-departure time CV differences are negligible throughout.

For SSi-aligner the range of throughput values generated went from 98–100.2 percent of expected capacity. In this range SA scheduling produced mean delays varying from 88 to 65 percent of those produced by UNIF/SRPT (Fig. 10).

Wein [30] observes that release control has substantially more impact on cycle time than does dispatching. Our experiments confirm his observations. Figs. 11–13 compare SA/SA scheduling with SA/SRPT, shortest imminent processing time (SA/SIPT), and SA/FIFO, respectively. By fixing the release policy to SA and varying the dispatching rule used we do not register the disparities observed when different release strategies are compared.

In Fig. 11 the tradeoff curves are almost identical, with

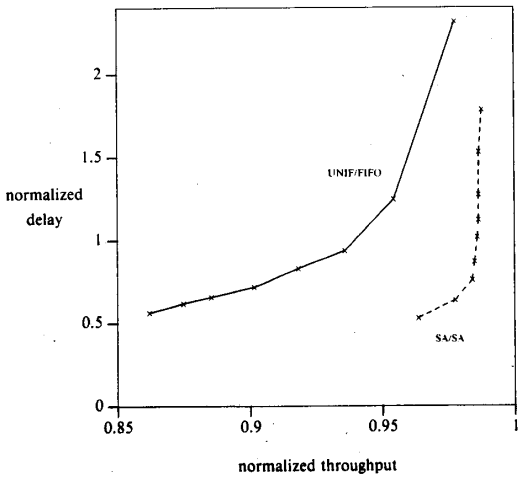


Fig. 8. SA/SA and UNIF/FIFO on Bipolar.

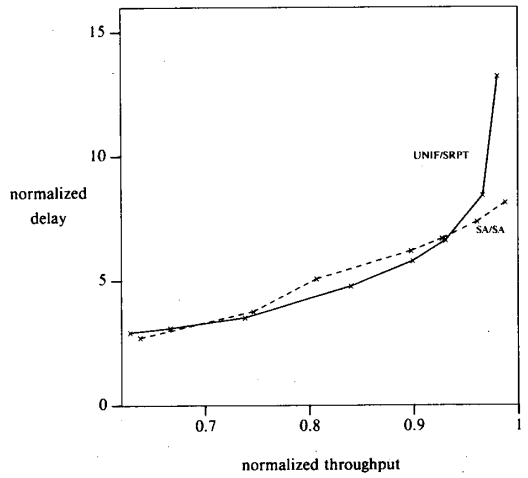


Fig. 9. SA/SA and UNIF/SRPT on SSi-implant.

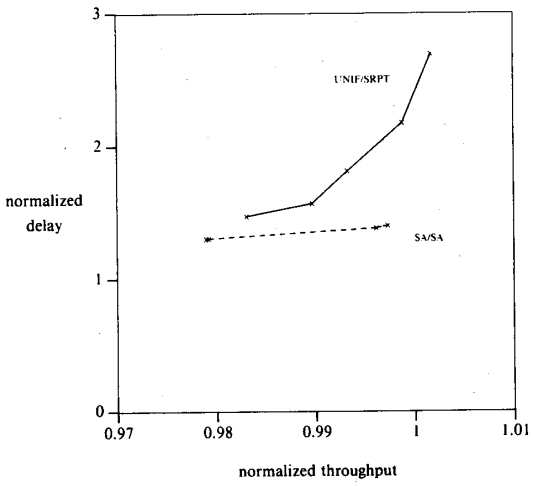


Fig. 10. SA/SA and UNIF/SRPT on SSi-aligner.

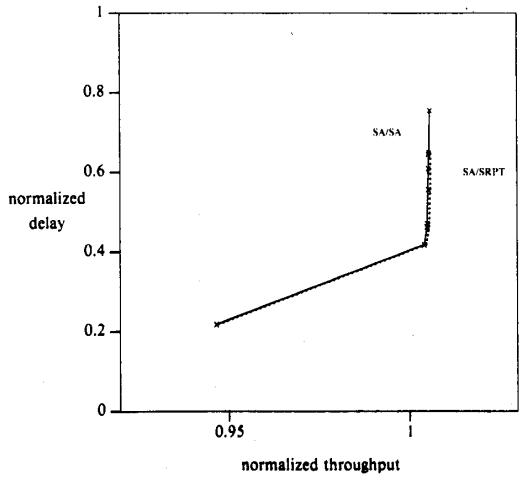


Fig. 11. SA/SA and SA/SRPT on Small-Fab.

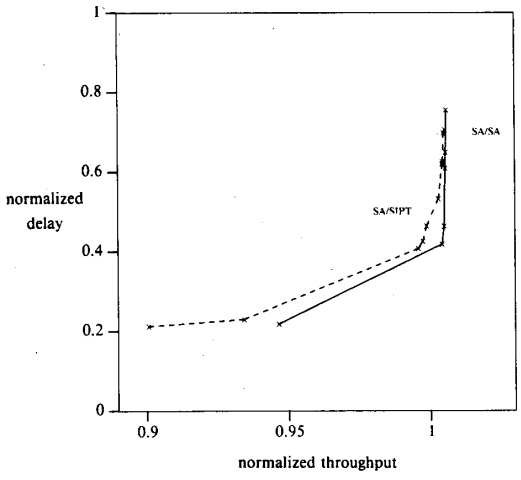


Fig. 12. SA/SA and SA/SIPT on Small-Fab.

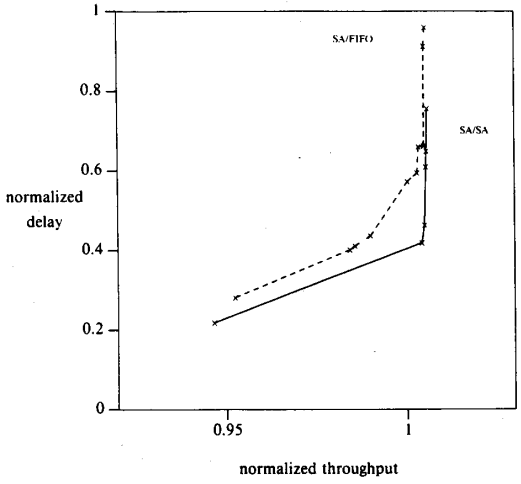


Fig. 13. SA/SA and SA/FIFO on Small-Fab.

SA/SA producing schedules having a slightly higher throughput near capacity. CV of inter-departure times were slightly higher for SA dispatching.

In Fig. 12 SA/SA scheduling is somewhat superior to SA/SIPT. Inter-departure time CVs for SA/SA are approximately 60 percent of those of SA/SIPT.

SA/SA was, again, slightly superior to SA/FIFO (Fig. 13). The CV of inter-departure times were approximately 60 percent of those of SA/FIFO.

With respect to delay/throughput tradeoff SA/SA and SA/SRPT were the best policies, followed closely by SA/SIPT and SA/FIFO. With regards to inter-departure time CV SA/SRPT was the best, followed by SA/SA. SA/FIFO and SA/SIPT were the worst with similar CV values.

VI. Concluding Remarks

In this paper we discuss how the philosophy of inventory management can be adapted to job shop scheduling with unreliable machines. We devise a scheduling policy, called Starvation Avoidance (SA), that monitors the virtual inventory of the bottleneck resource and releases work into the shop whenever this inventory falls below a safety stock level.

The following ranking of release strategies in order of increasing effectiveness: Uniform, Fixed-WIP, Workload Regulating, and Starvation Avoidance, seems to hold for all the experiments we performed. We would expect this ranking to hold for a very large class of shops for the following reasons: Any reasonable closed loop control should be better than open loop control, so the Uniform rule is the worst. All the closed loop rules adjust the arrival rate to the shop so that it is negatively correlated with the queue length at the bottleneck, (thus showing that increasing the variability of the arrival process to a queueing network does not necessarily increase delays). All the closed loop rules are equivalent if the bottleneck is the last operation before completed work leaves the shop (and each lot visits there only once). Otherwise, a simple thought experiment suggests why Fixed-WIP is the worst of the closed loop controls. Imagine a breakdown of the last work station. Then no work leaves, so under Fixed-WIP no new work starts, and, if the breakdown lasts long enough, the bottleneck starves. But both SA and Workload Regulating ignore all lots that have passed the bottleneck for the last time and continue to feed work into the bottleneck. Similarly, if the bottleneck station breaks down, Fixed-WIP will continue to pile up the inventory of new work in front of the bottleneck (until everything after the bottleneck has left the shop), but both SA and Workload Regulating will stop releases. In straight line flow shops, SA and Workload Regulating are equivalent. In reentrant flows, Workload Regulating counts all the work remaining at the bottleneck on each lot, not just the next bottleneck operation. Let $L = 0.9$ h and consider two cases of an (almost) empty shop. In case 1, two jobs are in queue at the bottleneck, each with 0.5 h of work at the next bottleneck operation. SA would not start a new lot (assuming $\alpha = 1$). Now suppose only one job is in queue with two bottleneck operations to be performed, each taking 0.5 h, and 1 h of processing on some other machine between them. Workload Regulating would not distinguish between these cases, but SA would start a new lot in the second case.

The policy was extensively tested on several fab networks. We can make the following conclusions about SA.

1) SA is an effective scheduling policy in that it produces near-capacity throughput while maintaining the average job delay considerably lower than when traditional job release policies are used.

2) Near capacity, SA outperformed all other policies in all test cases.

3) As is the case with inventory control, SA is sensitive to the randomness of the lead time. If the lead time from new wafer starts to the bottleneck work station increases in length or variability, not only will the tradeoff curve shift upward (longer delays for any output rate) but the relative reduction in delay achieved by SA compared with UNIF will be not as dramatic. This is clearly seen in comparing the two versions of the Silicon Systems model. This is an example of a common phenomenon in control systems: Introducing time lags or noise in the feedback loop of a stochastic system will degrade performance. Randomness of lead time can be reduced by having the first bottleneck visit occur early in the process recipe.

4) SA is easy to implement, provided a CIM system with up-to-date shop floor information is available. In practice, the throughput control parameter α used in SA must be set. This parameter controls the factory throughput. One way to set α is with simulation. This, however, requires a validated simulation model that may not be available. Another approach is as follows: Run the fab for a period of time releasing work with the UNIF control policy and measure the virtual inventory W. By using $\alpha = W/L$ as the control parameter setting (where L is the lead time for replenishment) then with high probability the throughput will be larger than the average throughput measured under the UNIF release policy. Since near capacity the slope of the delay/throughput curve for SA is steep, by reducing α one can still maintain a high throughput rate, while reducing considerably the mean delay. In practice, α can be decreased slowly, with careful monitoring of the corresponding delay and throughput rates, until a desired point on the delay/throughput curve is reached.

5) For simplicity, we assume in this paper that the fab has a unique process flow. This assumption can be relaxed easily. Let L_i be the expected time required for a lot of type i ($i = 1, \cdots, p$) to reach the bottleneck for the first time. Let the lead time for replenishment be

$$L = \max (L_i \mid i = 1, \cdots, p). \tag{19}$$

Furthermore, define a start slack for product i to be $\delta_i = L - L_i$. Whenever the virtual inventory falls below the safety stock level, select the product (k) that is furthest behind its cumulative starts schedule for release. The cumulative starts schedule is derived from the outs schedule

by shifting back by the lead time and scaling by an appropriate value to account for yield fallout. Release the lot after δ_k time units, but include it in the virtual inventory count immediately.

Future research should determine if the results for the single-process single bottleneck case described in this paper hold for the several multiprocess multibottleneck cases. In this paper we describe only one interpretation of the SA principle. With other definitions of virtual inventory (e.g., making use of improved lead time and equipment downtime estimates) other interpretations of SA can be investigated.

REFERENCES

[1] M. Barry, private communication, Fairchild Semiconductor Corporation Research Center, Palo Alto, CA, 1986.

[2] J. H. Blackstone Jr., D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *International Journal of Production Research*, vol. 20, no. 1, pp. 27-45, 1982.

[3] D. Y. Burman, F. Javier Gurrola-Gal, A. Nozari, S. Sathaye, and J. P. Sitarik, "Performance analysis techniques for IC manufacturing lines," *AT&T Technical Journal*, vol. 65, no. 4, 1986.

[4] H. Chen, J. M. Harrison, A. Mandelbaum, A. van Ackere, and L. M. Wein, "Queueing network models of semiconductor wafer fabrication," Center for Integrated Systems, Stanford Univ., Stanford, CA, Tech. Rep., 1986.

[5] J. E. Day and M. P. Hottenstein, "Review of sequencing research," *Naval Res. Logist. Quart.*, vol. 17, no. 1, pp. 11-39, 1970.

[6] J. Dayhoff, "New techniques for simulation modeling and analysis," *CIM Review*, Winter 1987.

[7] J. E. Dayhoff and R. W. Atherton, "Simulation of VLSI manufacturing areas," *VLSI Design*, Dec. 1984.

[8] A. A. Fredericks, "Performance analysis modeling for manufacturing lines," *AT&T Tech. J.*, vol. 65, no. 4, 1986.

[9] E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems*. New York: Academic, 1980.

[10] P. Gise and R. Blanchard, *Modern Semiconductor Fabrication Technology*. Englewood Cliffs, NJ: Prentice-Hall, 1986, pp. 487-503.

[11] B. Gliffer and G. L. Thompson, "Algorithms for solving production scheduling problems," *Oper. Res.*, vol. 8, no. 4, 1960.

[12] R. Hooke and T. A. Jeeves, "Direct search solution of numerical and statistical problems," *Journal of the Association of Computing Machinery*, vol. 8, no. 2, pp. 212-229, 1961.

[13] F. P. Kelly, *Reversibility and Stochastic Networks*. New York: Wiley, 1979.

[14] R. C. Leachman, "Preliminary design and development of a corporate-level production planning system for the semiconductor industry," ORC 86-11, Operations Research Center, Univ. California, Berkeley, CA, ORC 86-11, Dec. 1986.

[15] J. D. C. Little, "A proof for the queueing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, no. 3, pp. 383-387, 1961.

[16] E. Lohrasbpour and S. Sathaye, "Simulation modeling of IC wafer fabrication lines," in *Tech. Program Proc., Semicon/WEST*, 1984.

[17] C. Lozinski, "A scheduling perspective on semiconductor wafer fabrication," manuscript, Dept. Industrial Engineering and Operations Research, Univ. California, Berkeley, CA, 1986.

[18] P. Mellor, "A review of job shop scheduling," *O.R. Quarterly*, vol. 17, no. 2, pp. 161-171, 1966.

[19] W. G. Oldham, "The fabrication of microelectronic circuits," *Sci. Amer.*, vol. 237, no. 3, 1977.

[20] S. S. Panwalker and Wafik Iskander, "A survey of scheduling rules," *Oper. Res.*, vol. 25, no. 1, pp. 45-61, 1977.

[21] P. Penfield, Jr., S. B. Gershwin, D. A. Hodges, C. M. Osburn, J. Reynolds, J. Shott, A. J. Steckl, and D. E. Troxel, "Requirements for computer-aided fabrication," Dept. Elect. Eng. Computer Science, Massachusetts Institute of Technology, Cambridge, MA, unpublished document, Jan. 17, 1984.

[22] K. G. Ramakrishnan and D. Mitra, "An overview of PANACEA, a software package for analyzing markovian queueing networks," *Bell Syst. Tech. J.*, vol. 61, no. 10, pp. 2849-2872, Dec. 1982.

[23] M. G. C. Resende, "Computer simulation of semiconductor wafer fabrication," Operations Research Center, Univ. California, Berkeley, CA revised as Rep. ORC 86-14, 1985.

[24] ——, "Shop floor scheduling of semiconductor wafer manufacturing," Ph.D. dissertation, Dept. Industrial Engineering Operations Research, Univ. California, Berkeley, CA, 1987.

[25] C. H. Sauer and K. M. Chandy, *Computer Systems Performance Modeling*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[26] R. L. Sisson, "Methods of sequencing in job shops—A review," *Oper. Res.*, vol. 7, no. 10, pp. 10-29, 1959.

[27] ——, "Sequencing theory," in *Progress in Operations Research*, vol. I, R. L. Ackoff, Ed. New York: Wiley, 1961.

[28] A. M. Spence and D. J. Welter, private communication, 1986.

[29] H. A. Watts, "Scheduling an automated manufacturing line," PROMIS Systems Corp., Santa Clara, CA, manuscript, 1986.

[30] L. M. Wein, private communication, 1986.

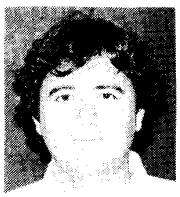
[31] W. Whitt, "The queueing network analyzer," *Bell Syst. Tech. J.*, vol. 62, no. 9, pp. 2779-2815, Nov. 1983.

*

C. Roger Glassey received the B.S. degree in mechanical engineering from Cornell University, Ithaca, NY, 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, Rochester, NY, in 1961, and the Ph.D. degree in operations research at Cornell in 1965.

He served for three years as an officer in the U.S. Navy, then worked for the Eastman Kodak Co., Rochester, first as an Industrial Engineer, then in the corporate operations research group. He has been a member of the faculty of Industrial Engineering and Operations Research at the University of California, Berkeley, since 1965, and served as chairman of the department from 1980 to 1986. His primary research interests are production planning and scheduling and mathematical optimization, and another of his interests is microcomputers.

*



Mauricio G. C. Resende was born in Maceió, Brazil on July 27, 1955. He received the Electrical Engineer degree from the Pontificial Catholic University of Rio de Janeiro in 1978 and the Master of Science degree in operations research from the Georgia Institute of Technology, Atlanta, in 1979. He received the Ph.D. degree in operations research from the University of California, Berkeley, in 1987.

He was an Engineer at the operations research group of Furnas-Centráis Elétricas S.A., Rio de Janeiro, from 1979 to 1982. During the last two years of his doctorate studies, he was a consultant for Schlumberger Palo Alto Research—Fairchild. Currently, he is a full-time consultant for AT&T Advanced Decision Support Systems and AT&T Bell Laboratories. His research interests include mathematical programming, combinatorial optimization, production planning and control, and information systems.