Shop Floor Scheduling of Semiconductor Wafer Manufacturing

By

Mauricio Guilherme De Carvalho Resende

Engineer (Catholic University of Rio de Janeiro) 1978
M.S. (Georgia Institute of Technology) 1979

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ENGINEERING

in the

GRADUATE DIVISION

OF THE

UNIVERSITY OF CALIFORNIA, BERKELEY

Approved: Roger Glassey 13.Aug.87
Chairman Date
Robert C. Leachman 13 Aug 87
David A. Hodges 13 August 1987

**DOCTORAL DEGREE CONFERRED**
**DECEMBER 15, 1987**

# Shop Floor Scheduling of Semiconductor Wafer Manufacturing

Copyright © 1987

Mauricio Guilherme de Carvalho Resende

# Shop Floor Scheduling of Semiconductor Wafer Manufacturing

**Mauricio G.C. Resende**

Doctor of Philosophy — Industrial Engineering and Operations Research

---

C. Roger Glassey
Chairman of Committee

## Abstract

Semiconductor wafer fabrication is perhaps the most complex manufacturing process found today, with a wide range of complex issues related to production planning and scheduling. Computer-integrated manufacturing (CIM) systems make it possible, in theory, to use global factory state information for factory control decisions. The possibility of making decisions on an expanded information set raises two of the research question treated in this dissertation: How should global information be summarized and used for decisions and how much improvement can one expect as a result, compared with decisions based on local information?

This research investigates lot release control and dispatching policies for shop-level scheduling of a semiconductor fab. We introduce a closed-loop scheduling policy, called Starvation Avoidance, that makes use of global factory state information and that in simulation experiments compares favorably to traditional scheduling policies (that use local and global information) with respect to the tradeoff between

throughput and queueing time.

The virtual inventory of the bottleneck resource is defined to be the work content at the bottleneck of all jobs either at the bottleneck station or expected to arrive at the bottleneck within a given lead time. Starvation avoidance monitors the virtual inventory and controls work release to avoid idling bottleneck machines without overloading the bottleneck.

The plan for the dissertation is as follows. Chapter 1 introduces the problem. In Chapter 2 we discuss at a superficial level the processes involved in transforming a slice of raw silicon into a wafer of integrated circuits. In Chapter 3 we review the literature of job shop scheduling (in particular, job shop dispatching) and scheduling and simulation of semiconductor wafer manufacturing. In Chapter 4 we introduce a queueing model of the dynamics of a semiconductor wafer fab and present a C programming language implementation of the model. In Chapter 5 we introduce a class of new job release mechanisms called Starvation Avoidance (SA). In Chapter 6 we compare several dispatching rules and release strategies on hypothetical and real data. Concluding remarks are made in Chapter 7.

*For*

*Lucia*

## Acknowledgement

The development of this dissertation would not have been possible without the four and a half years of financial support from my Brazilian sponsoring agency, Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq. Research in Computer Integrated Manufacturing at Berkeley is supported by the Semiconductor Research Corporation, Fairchild Semiconductor Corporation, Philips Research Laboratories, and The State of California through *Project MICRO* – Microelectronics Innovation and Computer Research Opportunities. Cray Research provided financial support for computer time on the Cray X-MP/14 supercomputer at U.C. – Berkeley.

During my five years as a graduate student at Berkeley, I learned much from Professor Ilan Adler. Our research on the implementation of Karmarkar's algorithm for linear programming brought many moments of excitement and intellectual fulfillment. Professor Richard E. Barlow introduced me to network reliability and encouraged me to publish my first paper in a U.S. scientific journal. It was an honor to have been a student of Professors Adler and Barlow.

Professor David A. Hodges introduced me to the problem of scheduling semiconductor manufacturing and to computer integrated manufacturing in general. Without his drive, this research could not have taken place. Professor Robert C. Leachman gave me constant encouragement by always demonstrating interest in my research. I thank them both for having participated in my dissertation committee.

Above all, I would like to express my thanks to Professor C. Roger Glassey. As an effective advisor, he guided my way through the many phases of scientific discovery and production. It has been both a pleasure and a challenge to work under his supervision.

To my many friends and colleagues at Berkeley, I thank you all for having made these years enjoyable. I will miss you all.

Randy Hughes and Don Ruffcorn at Schlumberger Palo Alto Research – Fairchild influenced my work by helping me see some *real* scheduling problems in industry.

I thank my Mother and Father for their loving support and encouragement during my *many, many* years as a student.

Finally, I thank Lucia for being my permanent source of love and strength.

# Table of Contents

Abstract ..... 1

Dedication ..... i

Acknowledgement ..... ii

Chapter 1. Introduction ..... 1

Chapter 2. Integrated Circuit Fabrication ..... 8

2.1. Integrated Circuits ..... 9

2.1.1. Deposition and Epitaxy ..... 11

2.1.2. Oxidation ..... 12

2.1.3. Introduction of Impurities ..... 12

2.1.4. Photolithography ..... 13

2.1.5. Metalization ..... 15

2.2. Summary ..... 16

Chapter 3. Literature Review ..... 17

3.1. Problem Definition ..... 17

3.2. Single Machine Scheduling ..... 22

3.3. Multi-Machine Scheduling ..... 24

3.4. Job Shop Simulation Research ..... 28

3.5. Simple Dispatching Rules ..... 30

3.6. Heuristics ..... 34

3.7. Search-Based Dispatching Rules ..... 37

3.8. Scheduling Integrated Circuit Manufacturing ..... 39

3.9. Summary ..... 41

Chapter 4. Network Queueing Model and Simulator ..... 42

4.1. The Wafer Fabrication Model ..... 42

4.2. Simple and Weighted Priority Dispatching ..... 43

4.3.1. FabSim - A Discrete Event Fab Simulator ..... 46

4.3.2. Pattern Search ..... 54

4.3.2.1. Normalized Priority Functions ..... 58

4.4. Extensions ..... 59

4.5. Summary ..... 60

Chapter 5. Job Release by Starvation Avoidance ..... 61

5.1. One Product One Bottleneck Station Case - SA(1,1) ..... 66

5.2. The SA Dispatching Rule ..... 69

5.3. Multi-Product One Bottleneck Station Case - SA(N,1) ..... 72

5.4. One Product Two Bottleneck Stations Case - SA(1,2) ..... 74

5.5. Multi-Product Two Bottleneck Stations Case - SA(N,2) ..... 77

5.6. Summary ..... 80

Chapter 6. Simulation Experiments ..... 82

6.1. Experimental Design ..... 83

6.2. A Small Wafer Fab Queueing Network with Hub Station - Small-Fab ..... 84

6.2.1. Interpretation of Simulation Experiment - Small-Fab ..... 99

6.3. A Single Bottleneck Fab Queueing Network Dayhoff-87 ..... 106

6.3.1. Interpretation of Simulation Experiment - Dayhoff-87 ..... 112

6.4. A Bipolar Process Fab Queueing Network - Bipolar ..... 112

6.4.1. Interpretation of Simulation Experiment - Bipolar ..... 118

6.5. A Production Fab Queueing Network - SSi-Implant ..... 119

6.5.1. Interpretation of Simulation Experiment - SSi-Implant ..... 129

6.6. Another Production Fab Queueing Network - SSi-Aligner ..... 130

6.6.1. Interpretation of Simulation Experiment - SSi-Aligner ..... 135

6.7. Summary ..... 136

Chapter 7. Concluding Remarks ..... 137

References ..... 142

# Chapter 1 — Introduction

Semiconductor wafer fabrication [Old77a, Gis86a] is perhaps the most complex manufacturing process found today. This complexity is, in part, the result of constant devise 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 scheduling. The introduction of computer-integrated manufacturing (CIM) systems in the fabrication process has added a new dimension of problems.

Penfield *et al.*, in an unpublished document [Pen84a], include scheduling of lots in a wish-list for computer-aided fabrication of semiconductor wafers. They say —

> *Scheduling of Lots.* The system must provide movement recommendations for manually operated fabrication lines and it must control movements for lines with automated material handling systems. It must coordinate movements over a large, complex facility with the objective of meeting production requirements at each stage of the line. The system must have a fast-turnaround a capability, *i.e.*, it must be able to expedite high-priority lots such as prototypes without excessively retarding the production of other material. The scheduler must respond to random (*i.e.*, unplanned) and planned events that disrupt the fabrication process, including machine failures, material shortages, and operator absences. It should also schedule maintenance. Scheduling computations must be fast; the line must never be held up while movements are calculated. The scheduling algorithm should be based on historical experience on machine reliability, yield, and operation times, and it must be able to update estimates of such quantities as mean time between failures (MTBF). It should be able to provide such data to management as when a given lot will be completed. Due-date information must be available as soon as a fast-turnaround lot is loaded.

In this dissertation 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:

- How should global information be summarized and used for decisions?
- How much improvement can one expect as a result, compared with decisions based on local information?

We do not consider high-level production planning in this study. For all purposes, we assume an exogenous entity (*e.g.*, a high-level scheduling module [Lea86a]) provides the short-interval scheduling module with a product mix and lot start rates.

In this introductory Chapter we briefly outline what is to follow in the next six Chapters of this dissertation.

A modern integrated circuit fabrication facility commonly produces devices using one or more fabrication processes. Presently, the three most important processes are the Bi-Polar, NMOS (n-channel MOSFET), and CMOS (complementary MOSFET) fabrication processes. Each process has associated with it a sequence of process steps that wafers of that type, grouped in lots, must follow. Within a process, a product is defined by the set of masks used in photolithography. After fabrication products are further broken down into sub-products (bins) according to performance characteristics, *e.g.* clock speed, power dissipation, etc. The number of products a typical fab may produce can range in the hundreds. Lots are moved around in the fab, competing for scarce resources, *e.g.* unreliable equipment (processing and materials handling), labor, shelf space (buffer), and raw materials.

Market conditions generate varying demands for IC products. High-level production planning models analyze these demands and generate out schedules, product by product, that guide the shop-floor scheduler in making its decisions.

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 its contribution to low yield. Some machine failures result in process malfunctions that are not detected until 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 (Fig. 1.1). This curve, $D(t)$, describes the average delay or waiting time as a function of fab throughput, $t$. It is well known that as throughput approaches the capacity of the fab the average queueing time goes to infinity.

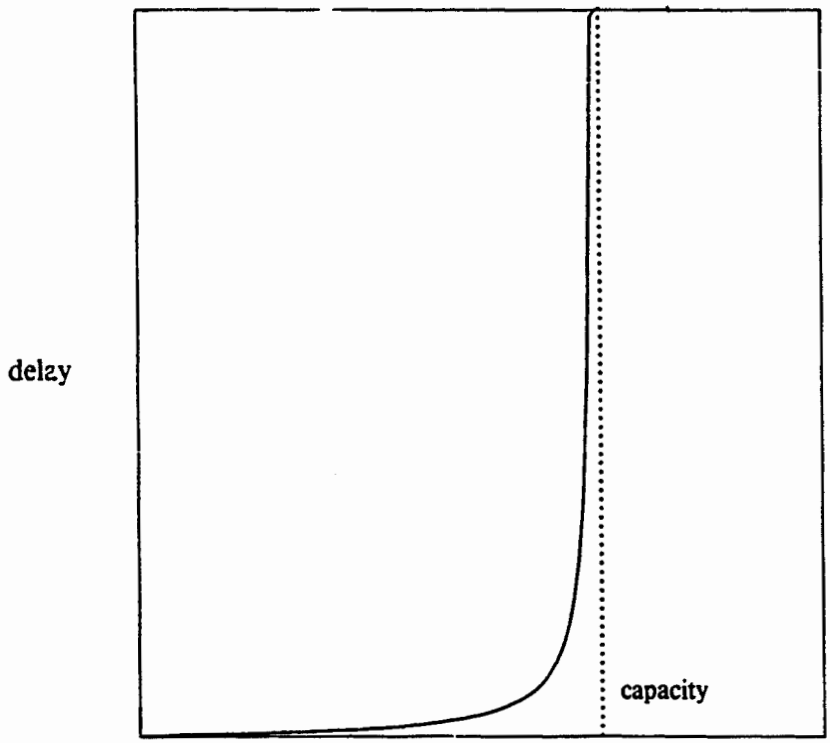


Fig. 1.1 - Mean Delay-Mean Throughput Trade-Off Curve

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. 1.2).

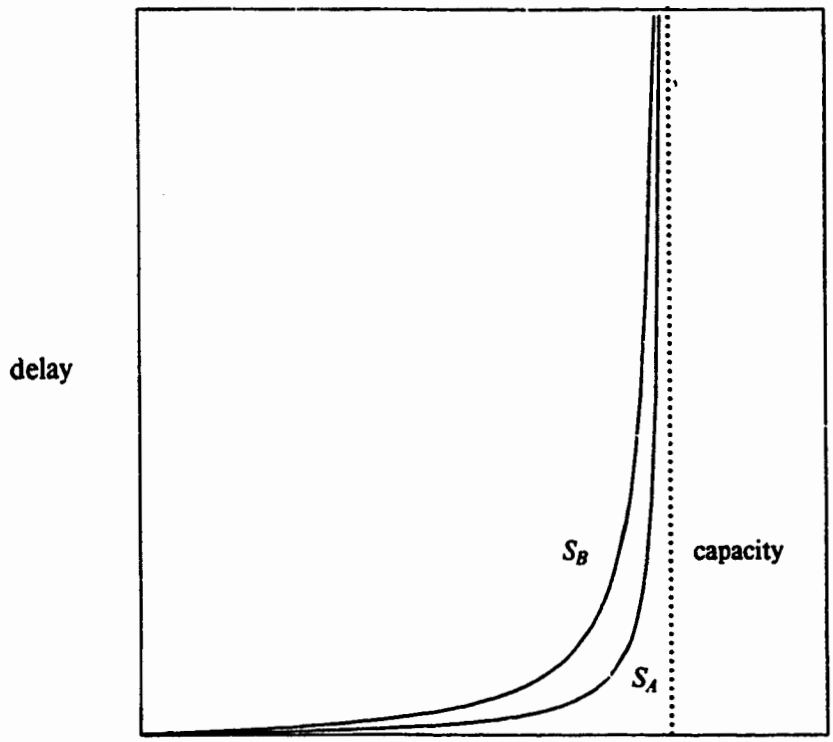


Fig. 1.2 - Scheduling Policy Comparison

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 [Lit61a], the expected inventory of work awaiting processing (WAP) equals *average delay* x *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 set ups. In this paper we are concerned only with the first tradeoff.

We limit ourselves to one particular aspect of wafer fab scheduling, *i.e.* scheduling wafers on a fab with unreliable processing equipment, having at most two bottleneck stations (such as photolithography and ion implant). There are several other scheduling problems that require further investigation, *e.g.* operator scheduling, scheduling of operations that have a long cycle time, scheduling of batch operations, and scheduling with set-up or changeover times.

The plan for the remainder of this dissertation is as follows. In Chapter 2 we discuss at a superficial level the processes involved in transforming a slice of raw silicon into a wafer of integrated circuits. We begin by presenting an overview of the overall manufacturing process (fab, sort, assembly, test) and then describe in more detail the process of fabrication. We describe deposition and epitaxy, oxidation, introduction of impurities, photolithography, and metallization.

In Chapter 3 we review the literature of job shop scheduling (in particular, job shop dispatching) and scheduling and simulation of semiconductor wafer manufacturing. The assumptions most found in the literature are listed, as are the most cited system performance measures. The notion of equivalent performance measures is reviewed. We enunciate several important theoretical results for the single-machine scheduling problem. For the multi-machine scheduling problem we review attempts at solving the problem exactly and then concentrate on heuristics for approximate solutions. In particular we review the literature of job shop dispatching heuristics and job shop simulation research. We list the relevant work related to the most studied simple dispatching rules as well as to more complex heuristics and weighted dispatching rules. Optimization techniques for weighted dispatching rules are surveyed. We conclude the Chapter reviewing the more recent literature of shop-floor level production planning in the semiconductor industry.

In Chapter 4 we introduce a queueing model of the dynamics of a semiconductor wafer fab and present a C programming language implementation of the model.

The model is a generalization of the classical job shop, allowing for unreliable parallel machines, multiple products, deterministic and stochastic (rework) job routes, deterministic or stochastic inter-station travel times, regular- and high-priority (hot) jobs, and two levels of scheduling control – job dispatching and job release. The C implementation of the model is a discrete-event simulation program called *FabSim*. Every aspect of the model is implemented in FabSim. Moreover, the program allows numerous dispatching and release control combinations, including optimal multi-rule dispatching with Hooke and Jeeves pattern search for optimization. FabSim collects statistics and displays them for analysis. Several improvements to both the model and the simulation program are suggested.

In Chapter 5 we introduce a class of new job release mechanisms called Starvation Avoidance (SA). We define the concept of *virtual inventory* in job shop scheduling and adapt ideas from inventory management to control work release into the shop. In inventory management when the inventory falls below a given safety stock, a new order is made. In our analogy, when the *virtual inventory*, *i.e.* the work content (on hand) at the bottleneck plus the work content (on order) moving towards the bottleneck (and expected to arrive within a limited time frame) falls below a given safety stock level, a new job is released into the shop. We discuss a dispatching scheme designed to aid SA achieve its objective, *i.e.* minimize idle time on critical shop equipment.

In Chapter 6 we compare several dispatching rules and release strategies on hypothetical and real data. We first compare 17 scheduling policies on a small but representative fab queueing network. Then we compare Starvation Avoidance with Uniform release on several large fab networks. FabSim is used for the simulation experiments, which are carried out on the Cray X/MP-14 supercomputer at U.C. – Berkeley.

Concluding remarks are made in Chapter 7.

# Chapter 2 — Integrated Circuit Fabrication

The worldwide integrated circuit (IC) market is expected to reach yearly sales in the order of $100 billion by 1990 [Ano81a, Ano82a, Ano82b] up from approximately $1 billion in 1970. This volume will be comparable to that of the automobile, chemical, and steel industries [Sze83a]. The foremost economic goal of the semiconductor device industry is the reduction of development and manufacturing costs per chip [Old77a]. To achieve the goal of economies of scale, circuit manufacturers must reduce device dimensions, increase the number of devices produced on a wafer, increase the number of wafers batched together, and increase the number of lots that can be processed simultaneously. In this way a large number of circuits can share the cost of labor and equipment. This, in fact, has occurred in the last 25 years, when component dimensions have shrunk exponentially while the number of components per chip has experienced exponential growth [Moo80a]. Today, chips are commercially available with minimum feature lengths in the sub-1 $\mu$m range and with over 1 million components. Consequently, an exponential decrease has occurred in the cost per bit in RAM chips [Noy81a].

Manufacturers must aim continuously to increase *yield*, *i.e.*, the proportion of good chips produced on wafer. Yields tend to increase as device size decreases [Old77a] and as processes mature, but yields of less than 30 percent for mature processes are commonly found in industry today. Equipment required for manufacturing semiconductor devices is costly. Prices in the over $1 million range are not uncommon. Efficient utilization of expensive equipment is therefore an important goal for manufacturers.

The semiconductor industry has not kept pace with other industries (such as automotive, petro-chemical, and steel) with regards to production planning and scheduling [Gar84a]. Production processes must be simplified and production logistics improved.

In this Chapter we describe the semiconductor fabrication process. We begin by presenting a brief overview of the IC fabrication process. We are far from complete in our description, limiting ourselves only to a level of detail that suffices for understanding the shop-level scheduling problems involved.

## 2.1 – Integrated Circuits

Integrated circuits are intricate three-dimensional structures with a complex internal composition built on wafers of semiconductor material. Silicon-based ICs account for 98% of all semiconductor devices sold worldwide [Sze83a]. We limit our exposition to silicon-based devices. As mentioned earlier, our discussion of IC fabrication is limited to an introductory level. For a detailed exposition of the subject, the reader is referred to Oldham [Old77a], Gise and Blanchard [Gis79a, Gis86a], and Sze [Sze83a].

The IC manufacturing process begins with the production of wafers. Raw wafers are usually produced by third parties and when purchased are ready for processing. Very pure single-crystal silicon is grown from specially pure sand (silicon dioxide). If required, the crystal is ground into a perfect cylindrical shape. Millimeter-thin slices, called raw wafers, are sawed from the crystal, polished, and shipped to IC manufacturers.

In fabrication, three dimensional structures are defined on the silicon wafer. After fabrication, wafers are probed and defective dice are marked. Wafers are then sectioned into tens to hundreds of chips. These are sorted according to performance characteristics, assembled in protective shells, further tested and sorted and finally shipped.

In this research we do not consider production problems that arise in the pre- and post-fabrication phases of circuit manufacturing, but rather limit ourselves to fabrication. See Gise and Blanchard [Gis79a, Gis86a] and Sze [Sze83a] for a discus-

sion of both wafer preparation and the post-fabrication phases of integrated circuit manufacturing. See Leachman [Lea86a] for a corporate-level production model of semiconductor manufacturing.

The IC fabrication process takes place in a clean room, where special measures are adopted to maintain the level of particle contamination low. Circuits are constructed in layers, one at a time. Each layer is defined by a pattern on a glass plate, called a photomask, whose features are transferred to the surface of the silicon wafer. This process of photo-engraving, called photolithography or simply masking, is key to fabrication. A wafer commonly undergoes over ten photolithographic steps in its fabrication process.

Between visits to the photolithography station a wafer visits stations where other physical and chemical processes are carried out. These include oxidation, diffusion, ion implantation, metallization, and etching. The fabrication process may require over 10 weeks from raw wafer to wafer probe. It can be viewed as follows.

---

```
begin process
    for k = 1 to number_layers do
        grow or deposit k-th layer
        begin photolithography
            apply photoresist
            pre-bake oven
            with k-th mask align and expose
            develop, rinse and dry
            post-bake oven
            inspect and measure
        end
        etch
        introduce impurities
        strip resist
    end
end.
```

---

In the remainder of this Chapter we describe in more detail each of the main processes encountered in semiconductor wafer fabrication.

### 2.1.1 – Deposition and Epitaxy

As each level of the wafer is defined a thin layer of material must be grown or deposited on the wafer surface to provide conducting regions within the device, protection from the environment, and electrical insulation between metals. The most common techniques for deposition are atmospheric presure chemical vapor deposition (CVD), low-presure chemical vapor deposition (LPCVD), and plasma-assisted chemical vapor deposition (PCVD).

In chemical vapor deposition, the wafer and gaseous compounds react thermally in a reaction chamber, producing a thin layer of a desired stable compound on the surface of the wafer. This is usually a batch operation with several lots processed in parallel. Reaction chambers (furnaces) vary much in type, *e.g.* horizontal, vertical, cylindrical, and gas-blanketed down-flow systems. The most commonly deposited materials include polycrystalline silicon, silicon dioxide (doped and undoped), and silicon nitride.

Epitaxial deposition is a special form of CVD, its special characteristic being that a thin *crystalline* film is deposited on the wafer surface. Several techniques have been developed for epitaxial deposition, including vapor growth, hydrogen reduction of silicon tetrachloride, and pyrolysis of silane. The deposition process begins by cleaning the wafer surface. The wafers are loaded on the epitaxial system with great care so as not to contaminate their surfaces. The epitaxial system is heated up to the proper temperature and a thin region of damaged silicon is removed from the surface of the wafer by means of an HCl etch. In the deposition step the layer is actually grown and doped. Finally the system is cooled down and the wafers are unloaded.

A more recent technology for epitaxial deposition is molecular beam epitaxy.

This process is carried out in a hard vacuum and as a result there is less diffusion of doped impurities than with furnace-based epitaxy.

### 2.1.2 – Oxidation

In the process of oxidation, a protective layer of silicon dioxide is grown on the surface of the silicon wafer. Oxidation is required throughout the entire IC fabrication process. Silicon dioxide has several uses [Sze83a]: to act as a mask against implant or diffusion of dopant into silicon; to provide surface passivation so that wafers do not degrade when exposed to air [Loz86a]; to isolate one device from another; and to serve as a component in MOS structures. The ability to grow such a layer makes silicon one of the most used semiconductor materials. Techniques for producing silicon dioxide include thermal oxidation, wet anodization, chemical vapor deposition, and plasma anodization or oxidation, the most common of which is thermal oxidation.

Thermal oxidation takes place in an oxidation furnace. It is a batch operation where several lots are processed simultaneously. Wafers are exposed to an atmosphere of $O_2$ or water vapor at temperatures in the 900–1300°C range. Oxidation is preceded by a cleaning operation designed to remove any impurities that may be present on the wafer surface. After cleaning, the wafers are dried and placed on a holder to be loaded on the furnace. The thickness of the oxide grown on the wafer surface is proportional to the furnace temperature and the length of exposure.

### 2.1.3 – Introduction of Impurities

Certain portions of the wafer require the introduction of controlled amounts of selected impurities. Two technologies are used for achieving this goal: diffusion and ion implantation. Diffusion takes place in a furnace and requires a pre-clean operation. In predeposition, the first step of the two step diffusion procedure, a carefully controlled amount of impurity is introduced into the wafer. Wafers are exposed, in batch, to a gaseous ambient of the dopant. By controlling temperature and time of

exposure the correct amount of impurities can be introduced. Another technique for predeposition makes use of source wafers, *i.e.* wafers made of the dopant material. Device wafers and source wafers are arranged next to each other and placed together in the furnace. The second step of diffusion is called drive-in. In this step no dopant is further introduced into the wafer, but rather the already introduced dopant is driven into the wafer to a desired depth. Diffusion is a batch operation.

In ion implantation, an electrical field accelerates ions of the desired dopant into the semiconductor. By scanning the wafer, a uniform predeposition can be obtained. The depth of deposition is controlled by the energy used to accelerate the particles. Ion implantation is performed one wafer at a time, with doses requiring anywhere from 5 seconds to 15 minutes of exposure. Following implantation, the wafer is often placed in a high-temperature furnace called a rapid thermal annealer to correct any damage caused by the implanter to the crystalline structure of the substrate.

### 2.1.4 – Photolithography

Photolithography, or masking, is central to semiconductor component manufacturing. It is in this step that circuit geometries are defined. Photomasks, containing the image to be transferred to the wafer are usually produced, under contract, by third parties. At the fabrication facility they are maintained in a mask library at the masking station. The mask generation process is a complex engineering task that is beyond the scope of this thesis. It is in the photolithography station that one finds the most costly processing equipment. For this reason and because process recipes require wafers to frequently return to photolithography, photomasking is commonly the bottleneck work station.

Gise and Blanchard [Gis79a] divide the photolithography process into eleven steps:

- substrate preparation

- surface preparation
- application of resist
- soft bake
- align and expose
- visual inspection - develop check
- hard bake
- etch
- strip resist
- final visual inspection

A substrate is grown on the surface of the wafer. This usually involves an oxidation step, but may also require a chemical vapor deposition (CVD), or a diffusion step.

In some instances the surface of the wafer must be cleaned prior to the application of the photoresist. Photoresist is a substance that is sensitive to ultra-violet (UV) radiation but not to the yellow light that illuminates the station. Light-hardened, or negative, resist hardens when exposed to UV radiation, while light-softened, or positive, resist softens. Resist is applied on the wafer by using a spinner. A spinner is a rapidly rotating devise on which the wafer is placed. A controlled amount of liquid photoresist is placed in the center of the spinning wafer and moves outward evenly on the wafer surface. This requires approximately 30 seconds and is done one wafer at a time.

In soft bake, or pre-bake, any extra solvent is baked out of the resist and the adherence of the resist to the wafer is increased. Soft bake is either carried out with hot air or infrared radiation. After cooling, the wafer moves to the most sensitive operation in the fabrication process – align/expose.

The photomask and wafer are brought tightly together, sometimes making use of vacuum to achieve bondness, and the mask is aligned to any previously defined

pattern on the wafer. First layer photolithography does not require alignment. After alignment has been achieved the mask and wafer are exposed to UV radiation, which reacts with the unprotected portion of the photoresist. Sze [Sze83a] estimates that in an industrial setting align/exposers can process approximately 100 wafers per hour. Aligners are characteristically unreliable and frequently go down for repair [Loz86a]. The softened part of the resist is dissolved and removed in the develop step.

The wafer is tested for proper alignment and resist quality and if approved is sent to hard bake. In hard bake more solvents are evaporated and the bond of the resist to the wafer surface is increased to withstand the subsequent etching processes. The equipment and baking times for hard bake are usually the same as for soft bake, but temperatures are higher.

The next step is etching. Two technologies exist for etching -- wet and dry etching. Wet etching takes place in a wet sink where a wafer lot is submerged in a bath of acid at a specified temperature. The acid selectively attacks portions of the wafer not protected by the hardened resist. After etching, wafers are washed and dried on a spinner dryer. In dry etching, or plasma etching, the wafer is placed in a chamber filled with a reactive gas. A magnetic field is applied to the gas, which etches away unprotected areas of the top layer. Dry etching produces sharper resolution than wet etching, where chemicals may undercut the resist.

Finally the resist is removed by either an acid bath or plasma techniques and the wafer once again is inspected. Wafers that do not pass this inspection are either scrapped for rework or are removed from the lot.

### 2.1.5 – Metallization

Metallization is one of the last steps in the fabrication process. It is in this step that the various regions of the device structure are connected to produce a circuit. Aluminum is the most used metal in the this phase, because it satisfies all of the required

constraints, *e.g.* low-resistance electrical contact to the silicon, good adherence to silicon dioxide, economically competitive, etc. The metal is usually deposited by means of vacuum deposition, one wafer at a time. The wafer is placed in a vacuum chamber which is brought down to a very low barometric presure. The aluminum is evaporated by one of several techniques (*e.g.* filament evaporation, electron-beam evaporation, sputtering) and deposits on the wafer. Heating may be required to increase adhesion of the materials.

## 2.3 – Summary

In this Chapter we discuss at a superficial level the processes involved in transforming a slice of raw silicon into a wafer of integrated circuits. We begin by presenting an overview of the entire manufacturing process (fab, sort, assembly, test) and then describe in more detail the process of fabrication. We describe deposition and epitaxy, oxidation, introduction of impurities, photolithography, and metallization.

# Chapter 3 – Literature Review

In this Chapter we review the literature of job shop dispatching and scheduling of semiconductor wafer fabrication. Several surveys of job shop dispatching have been previously published. The interested reader can refer to Sisson [Sis59a, Sis61a], Gliffer and Thompson [Gli60a], Mellor [Mel66a], Day and Hottenstein [Day70a], Panwalker and Iskander [Pan77a], and Blackstone *et al.* [Bla82a].

We first define the job shop scheduling problem and review theoretical results for the single machine scheduling problem. For all but the simplest versions, the job shop scheduling problem is NP-complete. We briefly review several attempts to obtain an exact solution and then focus on the main topic of this review – rules and heuristics for dispatching job shops. We number the usual assumptions made in the literature, survey simulation research of job shops, and then summarize results for simple and complex dispatching rules and dispatching heuristics. We review recent literature on rules that combine simple rules and use heuristic search to find a good combination. Several heuristic methods are surveyed and one in particular, pattern search, is covered in more detail. Lastly, we review the literature of scheduling integrated circuit manufacturing.

## 3.1 – Problem Definition

Lenstra and Rinnooy Kan [Len84a] define scheduling as the optimal allocation over time of scarce resources, in the form of *machines,* to activities, known as *jobs,* subject to the constraint that at any time no machine is allowed to process more than one job and no job can be processed by more than one machine. Several textbooks thoroughly cover scheduling theory. Among these, the reader is referred to Conway *et al.* [Con67a], Baker [Bak74a], French [Fre82a], and Hax and Candea [Hax84a].

Conway *et al.* [Con67a] propose notation to classify machine scheduling problems. Problems are categorized according to four parameters – $A/B/C/D$, where $A$

denotes the job arrival process, $B$ is the number of machines, $C$ denotes the flow pattern in the shop, and $D$ is the performance criterion that is to be optimized. Graham *et al.* [Gra79a] suggest a somewhat different classification scheme. Among the array of production scheduling problems discussed in the literature [Gra81a], a large portion of real-world scheduling problems fall into the category of job shop scheduling. The general job shop consists of at least two machines. A job can enter the shop at any machine and is allowed to leave the shop from any machine. Each job has associated with it a sequence of operations to be carried out in the shop. Each operation must be performed by a specific machine and has a duration that can either be deterministic or obey a probability distribution. The sequence of operations may require a job to visit a machine one or more times.

Several assumptions regarding the job shop scheduling problem are made in the literature. These assumption vary from paper to paper. The following are compiled by Gere [Ger66a].

- Machines are capable of processing only one job at a time. Scheduling more than one job at a time can be viewed as multi-processor scheduling, for which there are several surveys — Coffman [Cof76a], Gonzalez [Gon77a], Graham *et al.* [Gra79a], and Lenstra and Rinnooy Kan [Len80a].
- Preemption, or suspension of a job before completion for another job to begin, is not allowed. Preemptive scheduling is surveyed in Coffman [Cof76a], Gonzalez [Gon77a], and Graham *et al.* [Gra79a].
- Once a job enters the shop, it can only leave after completion of all its operations.
- Even though jobs may be made up of physically distinct parts, no job can be processed by more than one machine at a time.

- Set up time is included in the job processing time and hence processing times are independent of the sequence in which the operations are performed. The effect of set up time on job shop scheduling has been investigated by Baker [Bak68a] and Wilbrecht and Prescott [Wil69a].
- Transportation time between machines is negligible when compared to processing time.
- Uncapacitated queues, or in-process inventory, are allowed to build up in front of machines.
- Machines are 100% reliable. Baker and Nuttle [Bak80a] consider the case where scheduled maintenance is allowed for single machine scheduling.
- Labor is not a constraining factor. Nelson [Nel67a] studies job shops in which labor is a constraining factor.
- Due-dates are fixed for each job as it enters the shop and do not change after that.
- A penalty is usually incurred when a job leaves the shop after its due-date.
- Alternate job routes are not allowed. Morse [Mor58a] studies the case where alternate routes are allowed.

Two classes of job shop scheduling problems are discussed in the literature. In the *static deterministic* problem, $N$ jobs are given at time zero and must be feasibly scheduled to machines over time, so as to optimize some performance criterion. Most real job shops are dynamic probabilistic in nature. In the classical *dynamic probabilistic* job shop scheduling problem, $N$ jobs are given at time zero, but continue arriving throughout the planning horizon with inter-arrival times usually obeying a probability distribution. An assumption not commonly referred to in the literature is that machines may fail and require non-deterministic repair time. As with the static deterministic case, jobs must be feasibly scheduled so as to optimize a given

performance criterion. The criteria most used for measuring performance are –

- $F_{max}$ – maximum flowtime, the longest time any job spends in the shop,
- $\bar{F}$ – mean flowtime,
- $W_{max}$ – maximum waiting time, the longest time any job waits in the shop,
- $\bar{W}$ – mean waiting time,
- $C_{max}$ – maximum completion time (or makespan), the time required to complete processing on all jobs,
- $\bar{C}$ – mean completion time,
- $L_{max}$ – maximum lateness, where lateness is the amount of time by which the completion of a job exceeds its due-date,
- $\bar{L}$ – mean lateness,
- $T_{max}$ – maximum tardiness, where tardiness is the positive lateness,
- $\bar{T}$ – mean tardiness,
- $n_T$ – number of tardy jobs,
- $I_{max}$ – maximum machine idle time,
- $\bar{I}$ – mean machine idle time,
- $\bar{N}_c = (1/C_{max}) \int_0^{C_{max}} N_c(t)\, dt$ – mean number of completed jobs, where $N_c(t)$ is the number of completed jobs by time $t$.
- $\bar{N}_p = (1/C_{max}) \int_0^{C_{max}} N_p(t)\, dt$ – mean number of jobs being processed, where $N_p(t)$ is the number of jobs in-process at time $t$.
- $\bar{N}_W = (1/C_{max}) \int_0^{C_{max}} N_W(t)\, dt$ – mean number of jobs waiting for a machine

or not ready for processing, where $N_W(t)$ is the number of jobs waiting for processing or not ready for processing at time $t$.

- $\bar{N}_u = (1/C_{\max}) \int_0^{C_{\max}} N_u(t)\, dt$ — mean number of jobs still to be completed, where $N_u(t)$ is the number of jobs still to be completed, by time $t$.

We next define an important class for performance measures. Let $C_i$ be the completion time of job $i$. A measure of performance $R$ is said to be *regular* if it is non-decreasing in completion times, *i.e.* if

$$C_1 \leq C_1;\ C_2 \leq C_2;\ \cdots;\ C_n \leq C_n$$

is satisfied, then

$$R(C_1, C_2, ..., C_n) \leq R(C_1, C_2, ..., C_n).$$

Observe that $\bar{C}$, $C_{\max}$, $\bar{F}$, $F_{\max}$, $\bar{L}$, $L_{\max}$, $\bar{T}$, $T_{\max}$ and $n_T$ are all regular measures.

Two measures, $R_1$ and $R_2$, are termed *equivalent* if the following holds — a schedule $S$ is optimal with respect to $R_1$ if and only if schedule $S$ is optimal with respect to $R_2$. The following results are proved in French [Fre82a] regarding the equivalence of some of performance measures.

- $\bar{C}$, $\bar{F}$, $\bar{W}$, and $\bar{L}$ are equivalent measures of performance.
- If a schedule $S$ is optimal with respect to $L_{\max}$, then $S$ is optimal with respect to $T_{\max}$.
- $C_{\max}$, $\bar{N}_p$, and $\bar{I}$ are equivalent measures of performance.
- $\bar{N}_u$ and $\bar{C}/C_{\max}$ are equivalent measures of performance.
- $\bar{N}_W$ and $\bar{W}/C_{\max}$ are equivalent measures of performance.

- For the single machine scheduling problem $\bar{C}$, $\bar{F}$, $\bar{W}$, $\bar{L}$, $\bar{N}_n$, and $\bar{N}_W$ are equivalent measures of performance.

### 3.2 – Single Machine Scheduling

Pioneering work on single machine scheduling is that of Jackson [Jac56a] and Smith [Smi56a].

The following two theorems for single machine scheduling allow us to only consider permutation schedules when searching for an optimal single machine schedule [Con67a]. Permutation schedules are schedules that are completely determined by the job sequence.

**Theorem** – For an $n$-job single machine static scheduling problem with a regular measure of performance $B$ there exists an optimal schedule with no inserted idle time.

**Theorem** – For an $n$-job single machine scheduling problem with a regular measure of performance $B$ no improvement can be gained by allowing preemption.

Graham *et al.* [Gra79a] provide an excellent survey of results for single machine problems. We limit ourselves to four basic results. The first result is old, and is proven, among several authors, by Conway *et al.* [Con67a], Rinnooy Kan [Rin76a], and French [Fre82a].

**Theorem** – For the $n$-job single machine scheduling problem a minimum $\bar{F}$ schedule is such that

$$p_{i(1)} \leq p_{i(2)} \leq \cdots \leq p_{i(n)}$$

where $p_{i(k)}$ is the processing time of the $k$-th job processed, *i.e.* processing the shortest operation next is optimal.

The second result, which concerns scheduling with due-dates, is due to Jackson [Jac55a].

**Theorem** — For the $n$-job single machine scheduling problem a minimum $T_{\max}$ schedule is such that

$$d_{i(1)} \leq d_{i(2)} \leq \cdots \leq d_{i(n)}$$

where $d_{i(k)}$ is the due-date of the $k$-th job processed, *i.e.* processing the job with the earliest due-date next is optimal.

The third basic single-machine result reviewed here concerns preemption. Two forms of preemption have been studied in the literature. In *preempt-resume* processing is resumed at the point of preemption, whereas in *preempt-repeat* current processing is restarted from the beginning of the preempted operation.

**Theorem** — If (1) the job arrival stream is independent of the dispatching rule used, (2) processing time of a job is independent of preemption, and (3) preempt-resume is allowed, then the policy that minimizes $\bar{F}$ is one in which the job with the shortest remaining processing time is being processed. If a shorter job arrives during processing, this policy requires preemption of the current job in favor of the shorter job.

This result is proven by Schrage [Sch68a].

The final result is due to Klimov [Kli74a]. It regards scheduling jobs that require a sequence of operations on a single machine. More specifically, let us assume that jobs arrive according to a Poisson process with rate $\lambda$. Each job is processed $k$ times, with the $j$-th operation having duration that is distributed according to d.f. $\beta_j(t)$. A job waiting for its $j$-th operation is queued in buffer $j$, $j = 1, \ldots, k$, in order of arrival. There is no delay from the time a job finishes processing to the time it enters its next buffer. If preemption is not allowed, the following theorem holds.

**Theorem** — A policy that minimizes $\bar{F}$ is one that processes next the first job in the buffer with the largest index, *i.e.* shortest remaining processing time is optimal.

### 3.3 — Multi-Machine Scheduling

The general job shop problem is extremely difficult to solve optimally [Gra79a]. Say Conway *et al.* [Con67a] about this class of problems.

> The general job-shop problem is a fascinating challenge. Although it is easy to state, and to visualize what is required, it is extremely difficult to make any progress whatever toward a solution. Many proficient people have considered the problem, and all have come away essentially empty-handed. Since this frustration is not reported in the literature, the problem continues to attract investigators, who just cannot believe that a problem so simply structured can be so difficult, until they have tried it.

An O($n$log $n$) minimum makespan scheduling algorithm is proposed by Jackson [Jac56a] for the job shop with two machines and no more than two operations per job. When there are more than two machines or more than two operations per job, the problem has been shown to be NP-complete [Gar76a]. Several integer and mixed-integer programming formulations have been proposed for the static deterministic job shop scheduling problem. For a survey of these studies see Conway *et al.* [Con67a], Baker [Bak74a], and Rinnooy Kan [Rin76a]. Manne [Man60a] was among the first to propose an integer programming formulation for job shop scheduling. Greenberg [Gre68a] presents a branch and bound solution technique for Manne's formulation. Other work involving integer programming for job shop scheduling includes Wagner [Wag59a], Bowman [Bow59a], Dantzig [Dan60a] and Fisher [Fis73a, Fis73b]. Story and Wagner [Sto63a] report on computational experience with integer programming for job shop scheduling. Integer programming is itself NP-complete [Kar72a, Bor76a] and is hence computationally infeasible for problems encountered in practice. Ashour [Ash70a] investigates relaxing the integrality constraint of the integer program and solving the resulting linear program with

subsequent rounding off via a heuristic. He reports that the resulting solutions do not seem sufficiently close to the optimal to warrant the computational effort involved. Schedule construction algorithms begin with a partial schedule and at each iteration heuristically schedule another job until all jobs are scheduled. Gliffer and Thompson [Gli60a] propose a schedule construction algorithm for static deterministic job shop scheduling. Exhaustive enumeration generates all possible schedules and selects the best. This is clearly infeasible for problems of even relatively small dimension. A job shop having $n$ jobs and $m$ machines may have up to $(n!)^m$ possible schedules, some of which may be infeasible because of routing constraints. For example, a 10 job, 5 machine job shop can have $6.29 \times 10^{32}$ feasible schedules (about 100,000 times the mass of the Earth in grams). Branch and bound techniques, Ashour [Ash74a], Brooks and White [Bro65a], Balas [Bal69a], Lageweg *et al.* [Lag77a], lessen the computational burden, but only allow solving small problems. In fact, a solution to a 10-job, 10-machine static deterministic job shop problem proposed in 1963 by Muth and Thompson [Mut63a] was only proven optimal in 1986 [Len86a].

The most used approach to model dynamic probabilistic job shops is via network of queues. Early analytical studies of network of queues are due to Jackson [Jac57a, Jac63a]. His analyses, however, rely on assumptions that render them difficult to apply to real job shops. These assumptions are:

- The inter-arrival times for jobs arriving from outside the shop are exponentially distributed.
- Processing times of jobs are exponentially distributed.
- Job routes in the shop are determined by a fixed transition probability matrix.

Lemoine [Lem77a] surveys the state-of-the-art of the theory of network of queues.

The heuristic approach most widely discussed in the literature for scheduling job shops (both static and dynamic) is dispatching. A dispatching rule is used to select the next job to be processed by a machine from a group of jobs queued in front of that machine. Dispatching rules may use global shop information, but almost the totality of research has focussed on rules that involve only local information, *i.e.* information regarding the current job step and the station where the job is processed.

Panwalker and Iskander [Pan77a], and Blackstone *et al.* [Bla82a] survey dispatching rules. Panwalker and Iskander classify rules into five categories.

- Simple rules usually rely on information related to a specific job, such as due-date, processing times, size of next queue, arrival time, and number of operations. Random selection is considered to be a simple rule.
- In rules that combine simple rules, queued jobs are grouped and each group obeys a distinct dispatching rule.
- Rules that combine several rules with weighted indices are termed weighted priority rules.
- Heuristic scheduling rules take into account more complex information, such as alternate routes, machine failures, and anticipated machine loads. These rules are usually used in combination with simple rules. We later survey heuristics that are described and compared by Gere [Ger66a].
- All rules that cannot be placed in one of the first four classes are classified as *other rules.*

Others have offered different classifications. One worth mentioning is that of Conway *et al.* [Con60a]. They classify priority rules into four categories:

- lateness rules, or rules that determine priority as an increasing function of job lateness;
- arrival order rules, or rules that assign priorities according to the order in which jobs arrived at a machine;
- rules that are determined by some property of the job itself;
- the random rule, which is the rule that assigns priorities at random.

Panwalker and Iskander [Pan77a] collect over 100 dispatching rules and cross-reference them to over 30 papers. Their cross-reference shows the most discussed rules in the literature are, in decreasing order:

- shortest imminent processing time (SIPT) — Select the job with the shortest processing time at this machine.
- first in first out (FIFO) — Select the job that first arrived at the queue of this machine.
- earliest due-date (DD) — Select the job that has the earliest due-date.
- longest imminent processing time (LIPT) — Select the job with the longest processing time at this machine.
- at random (RANDOM) — Select a job at random.
- least slack time (LS) — Select the job with least difference between due-date and expected remaining processing time.
- least slack time per number of operations remaining (LS/OP) — Select the job with least ratio of slack to number of operations remaining.
- longest remaining operation (LRPT) — Select the job with the longest remaining processing time.
- shortest remaining operation (SRPT) — Select the job with the shortest remaining processing time.

- fewest operations remaining (FOPRM) – Select the job with the fewest remaining operations at the shop.
- first into shop first out (FASFO) – Select the job that has been the longest in the shop.
- most operations remaining (MOPRM) – Select the job with the most remaining operations.
- truncated shortest imminent operation (SIPT/FIFO) – For jobs in the queue more than $T$ time units select the job that first arrived in the queue. If no job is in the queue for more than $T$ time units, select the job with shortest operation at this machine.

Several of these rules as well as others will be discussed in the remainder of this paper. Most studies involve the use of simulation. We next review some aspects of simulation of job shops.

### 3.4 – Job Shop Simulation Research

Because of the stochastic nature of the dynamic job shop, most studies of job shop dispatching make use of digital simulation. Early job shop simulation research is due to Jackson [Jac57b], Jackson and Nelson [Jac57c], and Baker and Dzielinski [Bak60a]. Several reviews of job shop simulation have been published. In particular, see Conway *et al.* [Con67a], Moore and Wilson [Moo67a], Day and Hottenstein [Day70a], and Baker [Bak74a]. Day and Hottenstein [Day70a] and Gere [Ger66a] describe simulation program housekeeping in detail.

Simulation models typically have several input parameters – job arrival process (rates and probability distributions), service process (rates and probability distributions), number of machine centers, number of machines in each center, dispatching rule, and a mechanism for generating job routes. Arrivals are, in most studies, Pois-

son with arrival rates set so as to avoid overload. Reinitz [Rei63a] justifies the use of the Poisson arrival process. Kuratani and Nelson [Kur60a], however, conclude that the job inter-arrival process should have a distribution resembling both the exponential and the Erlang distributions. Processing times at each machine are also assumed to obey probability distributions. There is more variety of distributions reported in the literature for processing time than for inter-arrival times, *e.g.* exponential [Bak60a, Con62a], Erlang [Kur60a], hyper-exponential [Kur60a], and normal [Con60a].

Shop size, or number of machines in shop, range from five [Con60a] to 1000 [LeG66a]. Baker and Dzielinski [Bak60a] compare shops with nine to thirty machines and conclude that shop size is not significant. Conway and Maxwell [Con62a] and Nanot [Nan63a] agree that shop size has no significant effect on the relative performance of dispatching rules. Therefore, Blackstone *et al.* [Bla82a] conclude, a small shop can be used, with considerable computational savings. The mechanism most reported in the literature for generating job routes is the use of a probability transition matrix to determine what will be the next station to visit, upon completion of service at the current station. Conway and Maxwell [Con62a] conclude that changes in the transition probability matrix do not significantly influence the relative performance of dispatching rules. A common practice is to set bounds on the number of operations a job is allowed to perform [Con60a, Kur60a]. Morse [Mor58a] observes that when alternate routes are allowed performance improves. As with any simulation study, attention should be paid to the statistical analysis of simulation output [Law83a]. Statistical issues discussed in the literature regard warm up, data collection, and methods for output analysis. The central issue of warm up is to decide when to start data collection. Conway [Con65a] and Kuratani and Nelson [Kur60a] discuss warm up strategies. Data collection should be done in a way that all estimates are meaningful. Conway *et al.* [Con60a] suggest that a simple scheme is

to select a fixed number of consecutive jobs from the output and use only values from those jobs to compute simulation statistics. Hershauer and Ebert [Her75a] use an online method, due to Fishman [Fis71a], for computing and collecting the sample size needed to estimate the mean of a simulated process with a desired degree of accuracy. They do not, however, report any numerical results. Several statistical methods have been used to analyze simulation output — correlation [Dzi63a], Student's $t$ [Row60a], analysis of variance [Bak60a], and factorial design [Bak60a, Con62a].

### 3.5 – Simple Dispatching Rules

Of the many dispatching rules reported in the literature [Pan77a] the most studied rule is shortest imminent processing time (SIPT). SIPT assigns the highest priority to the job that has the shortest processing time for that operation. Baker and Dzielinski [Bak60a] conclude that SIPT is the best rule if the objective is to minimize mean flowtime. Conway [Con65a] compares SIPT with 31 dispatching rules and concludes that SIPT dominates those rules despite the fact that it did not achieve the best performance in any of the experiments. In another study [Con62a] SIPT was shown to be insensitive to errors in estimated processing times as well as to shop size. In yet another study [Con65b] SIPT was shown to be the least sensitive of 39 rules to the due-date selection procedure used. Elvers [Elv73a] investigates the performance of 10 dispatching rules with regards to five variations of the TWK due-date setting rule, a rule based on total work content. His conclusion is that with respect to tardiness SIPT outperforms the other 9 rules investigated.

Of the simple rules, SIPT seems to be the most promising. Says Conway about SIPT [Con65a].

> Its performance under every measure was good, it was an important factor in each of the rules that exhibit a *best* performance under some measure, and it is simpler and easier to implement than the rules that surpass it in performance. It surely should be considered the *standard* in scheduling research, against which candidate procedures must demonstrate their

virtue.

SIPT does tend, however, to cause some jobs to become extremely late. This can be somewhat overcome by applying the truncated SIPT rule, where SIPT is applied to any job having entered the queue after some time $T$, while jobs entering before $T$ have the highest priorities and are among themselves selected by the first-in-first-out (FIFO) rule. Another approach is to simply alternate between SIPT and FIFO or some other rule. Conway and Maxwell [Con62a] conclude that some form of truncated SIPT rule is the best hope for ridding SIPT of the increase in variance that often results from its use. Eilon and Cotterill [Eil68a] and Eilon *et al.* [Eil75a] report on the success of a truncated SIPT rule that they call $SI^x$. In this rule, $F_i$, a function of the time until due-date and remaining processing time, is computed for each job $i$. If any $F_i \leq 0$, the job with the smallest $F_i$ value is selected. Otherwise, SIPT is applied. Oral and Malouin [Ora73a] describe another truncated SIPT rule they name SPT-T. This rule selects the minimum between SIPT + $\gamma$ and slack time (difference between time until due-date and time of remaining operations), where $\gamma$ is a control parameter. When $\gamma$ is large the SPT-T rule becomes a pure slack rule. Yet another rule that attempts to reduce the variance in flowtime is the *COVERT* rule. This rule selects the job with the smallest ratio of delay cost to imminent processing time. In a Ph.D. thesis by Carroll [Car65a] this rule produced schedules that were superior to those generated by truncated SIPT, in the sense of less variance of flowtime.

Sen and Gupta [Sen84a] review the literature of static scheduling research involving due-dates. Their survey, however, does not cover dispatching rules. Studies involving rules that use due-date information require due-dates to be set for every job entering the shop. Conway [Con65a] classifies due-date setting rules into two categories – those that are exogenously determined and those that are set internally. In the former group he studies the constant (CON) and the random (RAN)

due-date setting procedures. In the latter group he examines total work content (TWK) and number of operations (NOP) procedures. Baker and Bertrand [Bak81a] compare three due-date setting rules: CON, TWK, and slack (SLK) and conclude that due-date setting error is minimized on average by TWK. For other studies on due-date setting rules see Baker [Bak74a], Conway [Con65b], Elvers [Elv73a], Jones [Jon73a], Reiter [Rei66a], Seidmann and Smith [Sei81a], and Baker [Bak84a]. Ragatz and Mabert [Rag84a] develop a model for due-date management.

The main characteristic of due-date based rules is that they usually produce schedules that have less number of late jobs, and flowtimes with less variance than schedules determined by processing time based rules [Con65b]. The most studied due-date based rules are —

- Earliest Due-Date (DD) — Select the job with the earliest due date.
- Least Slack Time (LS) — Select the job with the minimum difference between time remaining to due-date and remaining expected processing time.
- Least Slack per Operation — Select the job with the minimum ratio of slack to total expected remaining processing time.
- Least Job Slack Ratio — Select the jobs with the least ratio of slack to remaining time until due-date.
- Modified Minimum Slack — Select the job with the minimum difference between time remaining until due-date and expected time remaining in shop (processing and waiting).
- Least Modified Job Slack Ratio — Select the job with the least ratio of modified slack to time remaining until due-date.
- Critical Ratio — Select the job with the smallest ratio of time remaining until to due-date to lead time remaining. This rule involves estimating

queueing time remaining.

Of the rules described above, least slack per operation appears to perform best [Ger66a, Her75a]. In a study by Rochette and Sadowski [Roc76a] earliest due-date produced the least mean tardiness when the shop was 80% loaded, but at a higher 95% load SIPT proved to be superior. The critical ratio rule is used much in industry, but little has been published about its performance. For studies on the critical ratio rule, see Berger [Ber70a], Putnam *et al.* [Put71a], Wassweiler [Was72a], Berry and Rao [Ber75a], and Berry *et al.* [Ber84a].

First-in-first-out (FIFO) has been studied extensively in the literature. Panwalker and Iskander [Pan77a] refer to 16 studies on FIFO. The general consensus is that FIFO is no better, in terms of flowtime, than RANDOM, except that it produces schedules with somewhat less variance than those generated by RANDOM. Conflicting results [Con65b, Roc76a] indicate that first-into-shop-first-out (FASFO) and FIFO have no significant difference [Bla82a]. Least number of operations remaining (LNOPR) performed poorly in a study by Rochette and Sadowski [Roc76a].

The number in next queue (NINQ) selects for processing the job whose next operation is at the station with the smallest current queue length. With the intent of doing better than SIPT, Conway and Maxwell [Con62a] test NINQ using SIPT as a tie breaker. They observe that while NINQ did better than RANDOM, it failed to outperform SIPT. A variation of NINQ is the work in next queue rule (WINQ). This rule selects for processing the job that will go next to the queue with the least work content. Conway [Con65a] compares NINQ and WINQ with SIPT. Neither one of the look-ahead rules produced mean flowtimes better than those generated by SIPT.

### 3.6 – Heuristics

Gere [Ger66a] investigates how heuristics can boost the performance of simple dispatching rules. He suggests eight heuristics, of which four are describes below.

- **Alternate Operation** – If scheduling a selected job causes some other job to have a negative due-date slack, revoke this assignment and attempt to schedule the next job on the priority list. If this job does not cause any slack to become negative, schedule it. Otherwise, schedule the original job indicated by that dispatching rule.
- **Look Ahead** – If one could look ahead and anticipate every possible conflict, an optimal schedule would be possible. This would require almost exhaustive enumeration and is therefore infeasible. One can, however, examine a small portion of possible conflicts with little cost. One example of such look ahead strategy is as follows. Upon scheduling a job, attempt to identify if any late or nearly late lot is about to arrive at that station. If one is found, idle time is scheduled until this late job arrives whereupon it is scheduled.
- **Insert** – If there is a job in the queue whose processing time is less than that of the duration of idle time scheduled by the above look ahead strategy, that short job should be scheduled. The insert heuristic says that the longest of any such job should be scheduled.
- **Re-do with Adjusted Due-Dates** – This heuristic is suggested for the static scheduling problem. First devise a schedule and apply it. For all late jobs, decrease their due-dates by the amount they were late and re-do the schedule.

Gere compares eight simple rules with and without his heuristics on several job shops and concludes:

- When the performance criterion is minimization of mean tardiness, the simple priority rule selected is not as important as the selection of the set of heuristics which boost the rule.
- When a rule is mixed with one or more heuristics there makes little difference which rule was chosen. Therefore a simple rule should be used.
- Heuristics that anticipate the future state of the system, such as alternate operation and look-ahead with insert improve schedules significantly both statistically and practically.

The implication of his conclusions is that developing more complex rules is a worthwhile endeavor.

### 3.7 — Search-Based Dispatching Rules

Fischer and Thompson [Fis63a] were amongst the first to suggest a rule that would adapt itself to the shop in question. Their rule, in essence, selects at random one out of two dispatching rules every time a lot dispatch is carried out. Initially, both rules are given identical probabilities of being selected. As learning takes place the probabilities are changed. Their most important, and surprising, finding is that even when one uses a random selection of the rules, the combination almost always outperforms either rule when used individually.

Hershauer and Ebert [Her75a] generalize the Fischer and Thompson idea by introducing a priority dispatching rule defined by a linear combination of several simple rules, each of which is initially assigned a relative weight. By using simulation and a search technique values for weights are found such that the combined rule is locally optimized.

Several authors have discussed issues related to solving optimization problems involving simulation. For a survey of the field, see Smith [Smi73a] and Biles and Swain [Bil79a]. Techniques for solving such problems can be categorized into three

groups: (1) direct search techniques; (2) first-order surface response methods; and (3) second-order surface response methods. Direct search techniques are those that do not require derivatives. Most known of these methods are pattern search of Hooke and Jeeves [Hoo61a], the sequential simplex method of Spendley, Hext, and Himsworth [Spe62a], and the complex method of Box [Box65a]. First-order methods estimate the gradient experimentally and apply the method of steepest descent. To estimate the gradient about the current point, a designed experiment, such as a $2^N$ factorial design, is conducted around the point with a simulation run for each point. Then, a first-order regression model is fit on this data. In second-order response surface methods non-linear programming algorithms are applied to a surface obtained by fitting a second-order regression model to the data obtained from a second-order response surface experimental design. Biles and Swain compare algorithms of the three classes in several problems and conclude that the effectiveness of the procedures depends on the error of the estimates made with the simulation and the topology of of the search surface. First-order methods are inefficient in the presence of large errors while second-order methods fail in the presence of irregular surfaces. Direct search methods, the authors conclude, are the most promising in these situations. Of these techniques, Hooke and Jeeves pattern search [Hoo61a] has received the most attention.

Taubert [Tau68a] was the first to use the Hooke and Jeeves pattern search in the context of scheduling. He shows the effectiveness of pattern search in solving the aggregate scheduling problem by searching for a local optimum on a 20 dimension response surface.

Hershauer and Ebert [Her75a] were first to suggest the use of pattern search [Hoo61a], to optimize the weighted balance in a multi-rule dispatching scheme.

Pattern search is a direct search method used when the first derivative of the objective function is either unavailable or its computation is unreliable. This is

typically the case in the search problems that show up in optimal multi-rule dispatching. Small changes in the vector of weights w can alter the sequence in which jobs are processed and cause step changes in the observed output. This makes the search surface, $h(\mathbf{w})$, rugged or discontinuous and consequently $\nabla h(\mathbf{w})$ is undefined in many points.

The algorithm is given an initial solution $\mathbf{w}_0 \in \mathbf{R}^N$ and an initial step size $\delta_0$. At each iteration, an *exploratory search* seeks an improving neighbor, $\mathbf{n} \in \mathbf{R}^N$, of the current best w. Each evaluation of the function $h(\mathbf{w})$ is carried out by running the simulation model with weight vector w. If an improving neighbor is found, a *pattern move* is carried out. In a pattern move, an exploratory search is conducted around point $2\mathbf{n} - \mathbf{w}$. This point is a likely candidate for improvement since, at least intuitively, it follows a historic *pattern* of improvement in the objective function. If an objective value better than $h(\mathbf{n})$ is found by the pattern move, w is set to this improved solution and another pattern move is applied. Otherwise w is set to n. On the other hand, if in the exploratory search conducted around w in the main loop of the algorithm, no improving point is found, the search step size is decreased and a new iteration begins. The algorithm terminates when the step size is smaller than a given minimum step length, $\underline{\delta}$.

Harrison and O'Grady [Har85a] suggest two modifications to the algorithm. The first resets the current search step size to its initial value every time an improving solution is found. This attempts to avoid an either too coarse or too fine sampling step. The second, and perhaps most important modification, takes place after the current step size falls below the given minimum step length $\underline{\delta}$. Instead of terminating the search, as one would do in the original version, the modified algorithm is continued with the current step size being increased every time an exploratory search fails. This measure is designed to help the algorithm avoid getting caught in local minima. The search is declared over when the step size reaches a value greater

than a given maximum step length $\bar{\delta}$.

Besides the modifications described above, Harrison and O'Grady observe that there is no unified format in which individual manufacturing environments can be placed, and consequently methods must be tailored to specific cases, making them difficult to implement in practice. They propose a general representation of priority that they suggest overcomes this difficulty. They describe how this adaptive scheme can be feasibly implemented as a real-time factory control system. In numerical experimentations a weighted priority rule is set using shortest imminent processing time, earliest due-date, latest due-date, minimum slack time, random, and maximum slack time, and a pattern search attempts to optimize the sum of mean flow time, mean idle time, mean waiting time, mean lateness, mean earliness, and mean tardiness. Results on a ten-machine shop show an improvement of as much as 31 percent and on the average of 16 percent with respect to the same rules used individually. The main result is that improvement was observed in every case studied.

Bunnag and Smith [Bun85a] implement a multifactor priority rule scheme with pattern search using a generalized objective function. Their objective is to minimize the sum of the cost of tardiness, in-process inventory, and idle machine time. The priority rule used is a linear combination of shortest imminent processing time, dynamic slack time per remaining operations, time remaining per remaining unit of processing time, and work in next queue. The initial base point is selected by choosing from the extreme point set, {(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)}, the point with the best objective function value. The multifactor rule is compared to four widely used rules, random, shortest imminent processing time, slack per remaining operations, and the critical ratio rule (ratio of remining time to remaining processing time) on several test problems. Improvement of the multifactor rule with respect to these rules varied from 7.26 to 43.22%.

### 3.8 – Scheduling Integrated Circuit Manufacturing

In the last few years there have been several publications related to scheduling and performance evaluation of integrated circuit (IC) fabrication.

Lohrasbpour and Sathaye [Loh84a] describe a simulation model of IC wafer fabrication lines that estimates work-in-process (WIP), throughput time, production rate, 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, namely capacity analysis, analysis of variability in wafer input rate, the effect of express (hot) lots, WIP reduction policies, and the effect of equipment downtime. Their most important observation is that by reducing the input coefficient of variation from 0.43 to 0.08 they were able to reduce cycle times by 20%, on average. They conclude that uniform starts are highly desired.

Dayhoff and Atherton [Day84a] describe the dynamics of wafer movement and identify several problems that can be analyzed by discrete event simulation, *e.g.* fab start-up, clearing of accumulated inventory, and the analysis of inventory/cycle-time/throughput tradeoff. The authors introduce the concept of fab signatures, a trio of plots displaying the effect of lot start rate on mean inventory, mean cycle time, and departure rate. The authors illustrate the use of the model by studying the effect of a poor dispatching rule, LRPT, on an IC fab.

Leachman [Lea86a] describes a linear programming based corporate-level planning model for the semiconductor industry.

Lozinski [Loz86a] identifies technological constraints that should be satisfied by an IC wafer fab scheduler.

Wein [Wei86a] compares several dispatching rules and lot release policies on three fictitious fabs (using data gathered at a Hewlett-Packard research fab) with respect to mean cycle time. Several of the scheduling rules used by Wein are derived by approximating the fab by a sub network of queues restricted only to stations that

are heavily loaded and using a Brownian network model that approximates a multi-class queueing network model with dynamic control capability. Wein, in the same way was Lohrasbpour and Sathaye, observes that reducing the variability of the input process is important in reducing cycle time. In fact, Wein concludes, release control is more important in reducing cycle time than are dispatching rules. Wein introduces a release strategy for a single product fab, named *workload regulating input*, that releases a lot into the fab every time the remaining work content at a bottleneck station for all lots still in the fab falls below a prescribed value. This scheme outperforms uniform release in his simulations.

Chen *et al.* [Che86a] present a class of queueing network models for the analysis of wafer fabrication facilities. Using these models on several years' data of a real IC wafer fab, they are able to predict key performance measures within about 10% of the values actually observed.

Burman *et al.* [Bur86a] discuss how operations research tools can be used to analyze the behavior of IC wafer 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 [Spe86a] 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 set-up times and rework), and operating rules (lot sizing and repairman wait time).

Watts [Wat86a] discusses the integration of scheduling in the computer integrated manufacturing (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.

### 3.9 – Summary

In this Chapter we reviewed the literature of job shop scheduling (in particular, job shop dispatching) and scheduling and simulation of semiconductor wafer manufacturing.

The assumptions most found in the literature are listed, as are the most cited system performance measures. The notion of equivalent performance measures is reviewed. We enunciate several important theoretical results for the single-machine scheduling problem.

For the multi-machine scheduling problem we review attempts at solving the problem exactly and then concentrate on heuristics for approximate solutions. In particular we review the literature of job shop dispatching heuristics and job shop simulation research. We list the relevant work related to the most studied simple dispatching rules as well as more complex heuristics and weighted dispatching rules. Optimization techniques for weighted dispatching are surveyed.

We finalize by reviewing the more recent literature of shop-floor level production planning in the semiconductor industry.

# Chapter 4 — Network Queueing Model and Simulator

In this Chapter we present a queueing network model of semiconductor wafer fabrication. We formally define simple and weighted priority rule dispatching. We describe FabSim, a discrete event IC fab simulator, written in C, that allows the simulation of several release strategies and both simple and optimal multi-rule dispatching. The simulator is designed for use in research of shop-floor scheduling policies, but can also be applied to study problems such as fab layout design, capacity analysis, production forecasts, clearing of accumulated inventory, and fab start-up strategy analysis. We number several possible extensions to FabSim that, if incorporated, would improve the package's usability, efficiency, and modeling capability.

## 4.1 — The Wafer Fabrication Model

In this section we describe a wafer fab model that is implemented as a discrete event simulation engine.

Let fab $F$ be defined as a set of $M$ work stations, each having $m_i$ parallel unreliable machines, $i = 1, ..., M$. Machine $i$ of work station $j$ has time between failures and time to repair that are distributed according to probability distribution functions $G_{u+d}(i,j)$ and $G_d(i,j)$, 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 non-preemptive, *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 $p_i$, $i = 1, ..., p$ 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. The recipe defines, 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 the queue, the system must decide which lot or lots to process next when a machine becomes available. At the end of processing at a given station, the lot is placed in travel status towards its next processing station. The time it remains in travel status is a random variable specified by the recipe.

### 4.2 — Simple and Weighted Priority Dispatching

In Chapter 1 we define a shop-floor scheduling problem for semiconductor manufacturing. The approach taken in our research towards shop-floor scheduling is two-fold — lot dispatching schemes and lot release strategies. FabSim is the realization of the many strategies that we have pursued. In the dispatching arena we have investigated both simple local rules and more complex rules involving heuristic search. To understand what is available in the package we formalize, in this Section, the notion of dispatching as well as the concept of optimal multi-rule dispatching.

As defined in Chapter 3, *dispatching* is the process by which one or more queued lots are selected for processing at an idle machine. A common way to make this selection 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 $P_r$: $J_s \rightarrow \mathbf{R}$ that assigns real values to elements of $J_s$. Rule $r$ selects the lot with the smallest priority function value, *i.e.*, the lot with the highest priority. Denote $I_r(s)$ to be the index of the lot selected, *i.e.*,

$$I_r(s) = \text{argmin}\left\{ P_r(j) : j \in J_s \right\}. \tag{4.1}$$

A *normalized* priority function is a priority function $\bar{P}_r$: $J_s \rightarrow [0, 1]$ that assigns real values between 0 and 1 to elements of $J_s$.

If a batch operation is called for and more than one lot is to be selected, rule $r$ is repeatedly applied to a feasible sub-set of the remaining queued lots until the required number of lots are chosen.

Table 4.1 illustrates the above notation with some classical priority dispatching rules.

| Rule | $P_r(j)$ | Description | $I_r^s$ |
|---|---|---|---|
| FIFO | $A_s(j)$ | Arrival time of job $j$ at queue of station $s$. | Lot that has been most time in queue of station $s$. |
| RANDOM | $U(0,1)$ | Uniformly distributed number $\in$ (0,1). | Lot at queue of station $s$ selected at random. |
| SIPT | $p_s(j)$ | Processing time of job $j$ at station $s$. | Lot at queue of station $s$ with shortest processing time at that station. |
| SRPT | $r_s(j)$ | $p_s(j)$ + sum of processing times of job $j$ on all future recipe steps. | Lot at queue of station $s$ with shortest remaining processing time. |

Table 4.1 - Priority Function

As was described in Chapter 3, a priority dispatching rule can be defined to be a weighted sum of two or more rules. Let $R$ be the set of $N$ rules to be considered,

$$R = \left\{ r_1, r_2, \ldots, r_N \right\}, \tag{4.2}$$

and let $P_r$ be the priority function associated with rule $r$. Define a weighted priority rule, denoted by $\hat{r}(\mathbf{w})$, to be such that its priority function is given by

$$\hat{P}(j) = \sum_{i=1}^{N} w_i P_i(j) \tag{4.3}$$

where $\mathbf{w} \in \mathbf{R}^N$. Every input vector $\mathbf{w}$ defines a priority dispatching rule $\hat{r}(\mathbf{w})$. An obvious question to ask is — Which vector $\mathbf{w}^*$ defines the *best* dispatching rule $\hat{r}(\mathbf{w}^*)$?

To answer this question we require a model of the system under consideration. Let $S$ be a simulation model of fab $F$ with controllable input vector $\mathbf{w} \in \mathbf{R}^N$, uncontrollable input vector $\mathbf{z} \in \mathbf{R}^p$ and a measured output

$$\xi = g(\mathbf{w}, \mathbf{z}), \tag{4.4}$$

such as mean lot delay. It is important to note that $\xi$ is only an observed value and may not correspond to the true system output $y$.

By repeated measurements of the output $\xi$, an estimate $\hat{y}$ of the true system response can be computed according to

$$\hat{y} = \frac{1}{M} \sum_{i=1}^{M} \xi_i = y + \epsilon \tag{4.5}$$

where $M$ is the number of output measurements, $\xi_i$ is the $i$-th observed output, and $\epsilon$ is the estimation error, assumed to have normal distribution with mean zero and variance $\sigma^2 = \mathrm{Var}(\hat{y})$.

If we let

$$y = h(\mathbf{w}) \tag{4.6}$$

we can rewrite the estimate $\hat{y}$ as

$$\hat{y} = h(\mathbf{w}) + \epsilon \tag{4.7}$$

and therefore

$$E(\hat{y}) = y \tag{4.8}$$

and

$$Var(\hat{y}) = Var(\epsilon). \tag{4.9}$$

Our objective is to optimize the expected value of output estimate of the true system, $E(\hat{y})$. To do so, we define an optimization problem.

$$\text{opt } y = h(w) \tag{4.10}$$

$$\text{subject to: } w \in R^N \tag{4.11}$$

As discussed in Chapter 3, several techniques have been proposed for solving optimization problem (4.10-4.11). We select to implement Hooke and Jeeves pattern search due to its prior successful application in the context of job shop scheduling, *e.g.* Hershauer and Ebert [Her75a], Harrison and O'Grady [Har85a], and Bunnag and Smith [Bun85a].

### 4.3.1 — FabSim - A Discrete Event Fab Simulator

The fab model described in the previous section has been implemented as a discrete event simulator — FabSim. (See Law and Kelton [Law82a] for a discussion of computer simulation.) The code is written in C. Presently, versions of the code run on the following machines —

- VAX 11/750 and 8800 under BSD 4.3 UNIX, ULTRIX and VMS
- Micro-VAX II under ULTRIX and VMS
- IBM 3090 running VM/SP CMS (with a Waterloo C compiler)
- Cray X-MP/14 under UNICOS

In this Section data structures and logic of the code are briefly described. A list of available dispatching rules and release strategies is provided.

The program allows for two classes of dispatching schemes — with or without rule mix optimization. If optimization is required a variant of Hooke and Jeeves pattern search is performed with calls to the simulation engine. If no optimization is called for the program makes one call to the simulation engine. We begin by describing the simulator and then discuss our implementation of pattern search.

The simulation engine is designed to work as a list processor. All entities are represented as linked lists. The following lists are manipulated by the simulator.

- *product_recipe*— A linked list whose elements contain information about a single step of a product recipe. This includes step duration, step number, work station number, rework probability, yield, mean travel time to next station, and a pointer to its corresponding work station element in the list of work stations.
- *lot*— A linked list whose elements contain information about a single lot in the fab. This includes product type, lot number, hot-lot indicator, time into fab, time into present queue, time left in current equipment, priority in queue, cumulative queue time, cumulative yield, lot due date, and a pointer to its current position in the product recipe list.
- *work_station*— A linked list whose elements contain information about a single work station in the fab. Its elements contain station number, current queue size, equipment load size, cumulative queue size, pointers to the first and last elements of its queue, a pointer to the first element of the list of equipment of the work station, and a pointer to the first and last elements of the list of lots currently traveling from that station to the some other station in the fab.
- *queue*— A linked list representing a queue in a work station. Its elements contain a pointer to a lot element in the list of lots in the fab.

- *equipment*— A linked list of equipment in a given work station. Its elements contain equipment status indicator, time until equipment is to go down, time until equipment is to come back up, time until processing of current lot(s) is to terminate, cumulative time busy, cumulative time idle, cumulative time down, mean down time, mean up time, a pointer to the first element of the list of lots currently being processed by it, and a pointer to the element in the work station list that corresponds to its work station.
- *in_process*— A linked list of lots being currently processed by some equipment. Its elements contain a pointer to a lot cell in the list of lots in the fab.
- *travel*— A linked list, pointed to by a work station, of lots currently traveling from that station to some other station in the fab. Its elements contain a pointer to a lot cell in the list of lots in the fab and the lot's arrival time at the next station.

The simulation is event driven, *i.e.* time in the simulation is not incremented in units, but rather scheduled events are kept sorted in a heap data structure [Sta80a] and time is incremented to the epoch of the next scheduled event. The simulation engine is described below in pseudo-code.

---

```
procedure simulate()
begin
        set_up()
        update_times()

        while (simulation not done)
        begin
                lot_start_check()
                load_travel_check()
                load_queue_check()
                equip_status_check()
                load_equip_check()
                update_times()
        end
        output_reports()
end.
```

---

The set_up() routine sets up the program for running the simulation. In this routine the lists are initialized, problem data read, and initial simulation events are scheduled. Input includes a description of the fab and its products, and a set of simulation parameters that includes random number generator seeds, simulation horizon, release strategy, dispatching rule, residence time distribution, travel time distribution, reports to be generated, and due date setting rule. The description of the fab is given by a set of work stations, each with work station number, number of equipment in work station, equipment load size, an indication of whether the station is a hub station, and a list of equipment in each work station, with estimates of mean time to repair (MTTR) and mean time between failures (MTBF) for each piece of equipment. FabSim assumes that both MTTR and MTBF are random variables with exponential distributions. The description of the products is defined by the number of products, and for each product, the number of steps in its recipe, its wafer start rate, its profit value, the proportion of lots of that product that are hot, a due date setting parameter, and a list of steps that constitute the product's recipe, with each step having a step number, a work station where the step is to be performed, the mean step duration in minutes, the rework probability, the step yield, and the mean travel time from the current station to the next station in the recipe. Travel time can be either deterministic or random with exponential distribution.

With the present version of FabSim the following dispatching rules are available —

FIFO First-In-First-Out — Select the lot that first arrived at the queue.

SIPT Shortest Imminent Processing Time — Select the lot with the shortest processing time at the current station.

SIPT/T Truncated Shortest Imminent Processing Time — If no lot has been $T$ or more time units in the queue, select the lot with the shortest processing

time at the current station. Otherwise, select the lot that has been in the queue the longest, *i.e.* use FIFO.

LIPT Longest Imminent Processing Time — Select the lot with the longest processing time at the current station.

SRPT Shortest Remaining Processing Time — Select the lot with the shortest processing time for the rest of its operations in the fab.

LRPT Longest Remaining Processing Time — Select the lot with the longest processing time for the rest of its operations in the fab.

SIPT/V Smallest Ratio of Imminent Processing Time to Product Value — Select the lot with the smallest ratio of processing time to product value at the current station.

SRPT/V Smallest Ratio of Remaining Processing Time to Product Value — Select the lot with the smallest ratio of processing time for the rest of its operations in the fab to product value.

LNOP Least Number of Remaining Operations — Select the lot with the least number of remaining operations for the rest of its stay in the fab.

NINQ Smallest Number In Next Queue — Select the lot whose next station has the queue with the smallest number of waiting lots.

NINQ/M Smallest Ratio of Number In Next Queue to Number of Equipment — Select the lot whose next station has the queue with the smallest ratio number of waiting lots to number of equipment.

RAND Random — Select the next lot at random.

NEDNS Smallest Number of Equipment Down at Next Station — Select the lot whose next station has the smallest number of equipment down .

SCNQ Smallest Expected Queue Clear Time — Select the lot whose next station has the queue with smallest expected time to clear. This rule takes into

account the work remaining in queue and the number of down machines at the station.

SERT Shortest Expected Remaining Time in System — Select the lot with the shortest expected remaining time in the fab. Expected time remaining in system is computed by adding the remaining processing time with the expected remaining time in queue. Expected remaining time in queues is the sum of the expected time to clear queue for each queue remaining in the job's recipe. This is approximated by dividing the queue work content by the virtual number of machines at the queue's station (= number of machines x average machine availability).

DSERT Shortest Discounted Expected Remaining Time in System — Select the lot with the shortest expected discounted remaining time in the fab. This is similar to SERT with the difference that expected queueing time is discounted.

DQCT Discounted Queue Clear Time — Select the lot with the discounted expected remaining queue clear time. This is similar to DSERT except that job processing times are disregarded.

DD Earliest Due Date — Select the lot with the earliest due date. Due dates are currently set using the TWK rule, which sets the due-date of a job to $k \times p_j$ units of time after it enters the shop, where $k \geq 1$, and $p_j$ is the job's total processing time. Baker [Bak84a] describes TWK and other due-date setting rules.

MULTI Multiple rule — Given a set of simple rules and a vector of weights, MULTI combines the simple rules according to the given weights and selects the lot given the highest priority be the weighted priority function.

SA Starvation Avoidance rule — This rule combines, via the MULTI rule,

SRPT with a rule that prioritizes lots according to the ratio of duration of next bottleneck station visit to time until next bottleneck station visit. The weights are adjusted dynamically so as to give more weight to SRPT when the bottleneck is no danger of running out of work or to the ratio rule described above when there is imminent danger of starving the bottleneck. This rule is described in detail in Chapter 5.

OPTMIX Optimal multi rule — Given a set of simple rules and a vector of weights, OPTMIX combines the simple rules and uses Pattern Search to find a vector of weights that produces a good dispatching rule.

The update_times() routine finds the next event in the heap of scheduled events and updates the simulation clock and count-down times in the list structures. Count-down times include time until equipment is scheduled to go down, time until down equipment is scheduled to come up, time until a process is to finish, and time until the next lot is started.

The *while loop* is run while there are still lots in the fab that were started before the simulation horizon $H$. In the loop the program first checks, in lot_start_check(), if a new lot is to be started. This can be done in several ways. The program allows for the following release strategies —

POIS Poisson — Interarrival times for each product are distributed exponentially.

UNIF Uniform — Interarrival times for each product are fixed.

Fixed-WIP Fixed WIP — The number of lots in the fab is kept constant. When a lot of a given product departs, another of the same product is started.

WR Workload Regulation — This rule is due to Wein [Wei86a]. It releases a lot into the fab when the remaining work at the bottleneck for all lots in

the fab falls below a prescribed value.

SA Starvation Avoidance – This rule attempts to avoid starvation of any bottleneck machine. It is described in detail in Chapter 5.

If a lot is to be started, a new lot is defined, loaded into the queue of the work station of its first recipe step and (if the release policy is either POIS or UNIF) a new lot start for that product is scheduled and inserted in the event heap. In load_queue_check() equipment lists are scanned and if a lot is finished processing and no rework is required it is placed in the travel buffer of lots currently traveling out of this work station. An end travel event is scheduled and placed in the event heap. If rework is called for the lot is replaced at the head of the queue of lots waiting for processing at that station. In load_travel_check() travel buffers are scanned and if a lot is done traveling, it is removed from the buffer and is placed in the queue of the next work station in its recipe or is removed from the fab if it has just completed its last step. If the lot is removed from the fab, statistics are collected. Job statistics are collected for jobs started between times $\epsilon_1 H$ and $\epsilon_2 H$, where $0 \leq \epsilon_1 < \epsilon_2 \leq 1$ and $H$ is the simulation horizon. Station statistics are collected beginning at time $\epsilon_1 H$. The intent here is to avoid any bias (due to light traffic) before collection of lot statistics begins. In equip_status_check() all work stations are scanned for an equipment that is *idle* and scheduled to go *down* or one that is *down* and scheduled to come up, thus becoming *idle.* If found, equipment status is changed and new events scheduled and inserted in the event heap. In load_equip_check() work stations are checked for an *idle* piece of equipment and for waiting lots at the station buffer. If such a situation is identified, lots in the queue are ordered according to their priorities and one or more lots are selected for loading into the equipment. Lots in the same batch are constrained to be at the same recipe step as the first loaded lot (*i.e.* the lot with the highest priority). No more lots than the maximum load size of the equipment can be loaded at one time. The selected lots are

removed from the work station queue list and put into the equipment in_process list. An end processing event is scheduled and placed in the event heap. The iteration ends with an update_times() where the next event is retrieved from the event heap and the simulator clock and count-down times are updated.

Routine output_report() generates a report of the simulation run.

### 4.3.2 – Pattern Search

FabSim allows the choice of optimal multi-rule priority dispatching, or OPTMIX dispatching. To allow OPTMIX, FabSim makes use of Hooke and Jeeves [Hoo61a] pattern search. A detailed description of the Hooke and Jeeves pattern search algorithm implemented in FabSim is given in this Section.

As input to the algorithm one must provide –

- a simulation model $S$ of a fab $F$, with simulation horizon $H$.
- a set of dispatching rules $R$, with normalized priority functions,
- an initial weight vector $\mathbf{w}_0 \in \mathbf{R}^N$,
- an initial step size $\delta_0$,
- a minimum step size $\underline{\delta}$,
- a maximum step size $\bar{\delta}$,
- a shrinking factor $\rho$, $0 < \rho < 1$.

Pseudo code for the main procedure of the algorithm is given next. This procedure is called *patternsearch*.

---

**procedure** patternsearch(R, $w_0$, $\delta_0$, $\underline{\delta}$, $\bar{\delta}$, $\rho$)

**begin**
Set **n** = **w** = $w_0$
Set quit = FALSE
Set $\delta = \delta_0$
Compute $h_n$ = simulate(**n**)
Set $h_w = h_n$
Compute $h_n$ = explore(**n**, $h_n$)

**while** quit = FALSE **do**
**begin**
**if** $h_n < h_w$
**then** compute $h_n$ = patternmove(**w**, **n**, t, $h_w$,$h_n$,$h_t$)
**else** compute $h_n$ = homein(**w**, **n**, t, $h_w$,$h_n$,$h_t$,quit)
Set **n** = 2**n** − t
Compute $h_n$ = simulate(**n**)
Compute $h_n$ = explore(**n**, $h_n$)
**end**
**return**(**w**, $h_w$)
**end.**

---

Pattern search begins with an initial weight vector $w_0$. Two processes are available in FabSim for generating $w_0$ — the first generates *n* points at random and select the best as $w_0$ while the second selects as $w_0$ the best point out of the points $e_1, e_2, ..., e_N$, where $e_i$ has zeroes on all components except the i-th, which is unit. All point evaluations are carried out by calling procedure *simulate.* This procedure runs the simulation engine described earlier using weighted dispatching rule $\hat{r}$(w).

---

**procedure** simulate(w)

**begin**
Run fab simulation using dispatching rule $\hat{r}$(w).
Measure output $\xi$.
**return**($\xi$)
**end.**

---

At each iteration of *patternsearch*, *i.e.* when an improved point w is found, the algorithm (using procedure *explore*) generates several points in the neighborhood of w and returns the best point, n, with respect to the objective function under consideration.

---

**procedure** explore(n, $h_n$)

```
begin
    for k = 1 to N do
    begin
        Set n[k] = n[k] + δ
        Compute h_t = simulate(n)

        if h_t ≥ h_n then
        begin
            Set n[k] = n[k] - 2δ
            Compute h_t = simulate(n)

            if h_t ≥ h_n
                then set n[k] = n[k] + δ
                else set h_n = h_t
        end
        else set h_n = h_t
    end
    return(h_n)
end.
```

---

If the weight vector n is superior (in terms of the objective function) to w, a pattern move is carried out. This is implemented in procedure *patternmove*. Otherwise

a step size change action is taken in procedure *homein*.

In the pattern move, weight vector 2n - w is evaluated. If it is superior to n, then w is set to it. Else w is set to n.

---

**procedure** patternmove(w, n, $h_w$, $h_n$)

**begin**
    Set $\delta = \delta_0$
    **while** $h_n < h_w$ **do**
    **begin**
        Set t = w
        Set w = n
        Set $h_w = h_n$
        Set n = 2n − t
        Compute $h_n$ = simulate(n)
        Compute $h_n$ = explore(n,$h_n$)
    **end**
    **return**($h_n$)
**end.**

---

In the step size change procedure, *homein*, the current step size is initially reduced, and exploratory search is carried out around the current weight vector until either an improvement is found or the allowable number of candidate vectors are tested. In the former case w is set to the improving vector, the step size is reset to its initial value $\delta_0$ and control is returned to procedure *patternsearch*. In the latter case, the step size is further reduced (if possible) and another exploratory search is conducted. If no further reduction of the step size is possible, *i.e.* $\delta < \underline{\delta}$, the procedure begins its expand out phase. In this phase, the step size is increased and an exploratory search is carried out. Again, an improvement may be discovered. If this is the case w is set to this improved vector, the step size is reset to its initial value $\delta_0$ and control is returned to procedure *patternsearch*. If no improvement is found the step size is further increased, if possible, and another exploratory search is conducted. If

no further increase is possible, *i.e.* $\delta > \bar{\delta}$, the procedure returns a signal to *patternsearch* to terminate.

---

```
procedure homein(w,n, h_w, h_n, quit)

begin
        Set δ = ρδ

        if δ < δ̲ then
        begin
                Set δ = δ_0
                Compute h_n = explore(n, h_n)

                while h_n ≥ h_w do
                begin
                        Set δ = δ/ρ

                        if δ > δ̄ then
                        begin
                                Set quit = TRUE
                                break (from while loop)
                        end
                        else compute h_n = explore(n, h_n)
                end

                if quit = FALSE then
                begin
                        Set δ = δ_0
                        Set h_w = h_n
                        Set t = w
                        Set w = n
                end
        end
        return(h_n)
end.
```

---

#### 4.3.2.1. — Normalized Priority Functions

In the input list of our implementation of pattern search we include a set of dispatching rules $R$, with *normalized* priority functions. The requirement that the priority functions be normalized is practical in nature. Since the step sizes and shrinking fac-

tor are independent of the magnitude of the values of the priority functions, the algorithm frequently stalls in a non-optimal point when the dispatching rules of set $R$ have values with large disparities.

In our implementation we normalize the values by maximum possible value. For example, to compute the normalized SRPT rule we first compute for every station which process step at that given station has the maximum remaining processing time. Then, during dispatching all remaining processing time values are divided by the above maximum value. While this approach is simple, it has been able to improve the performance of pattern search substantially. This normalization, however, is rule dependent and may not be defined for some rules (*e.g.* FIFO). In our implementation we have defined normalized rules for the majority of the available dispatching rules.

### 4.4 – Extensions

Several extensions are planned for FabSim. These enhancements are designed to improve the simulator's modeling capabilities, efficiency, and usability.

To improve its modeling capability five items are of immediate importance – the modeling of setup and changeover times, the modeling of fab operators as a allocable resource, the ability to have rework loops that involve an extra step, such as layer scrape, and loops with one or *more* work stations, a greater variety of probability distributions, and the capability of restricting processing (for critical layers, for example) to a sub set of a station's machines.

As is the case with most simulators, efficiency is important. Portions of the code should be modified to enhance code efficiency. Research into parallel computer architectures for simulation should be pursued.

A user interface is needed. This interface should allow the user to easily define a simulation environment, design a simulation experiment, control the execution of

the simulation experiment, and manipulate both graphical and alpha-numeric simulation results.

### 4.5 – Summary

In this Chapter we introduced a queueing model of the dynamics of a semiconductor wafer fab and presented a C programming language implementation of the model. The model presented is a generalization of the classical job shop allowing for unreliable parallel machines, multiple products, deterministic and stochastic (rework) job routes, deterministic or stochastic inter-station travel times, regular- and high-priority (hot) jobs, and two levels of scheduling control – job dispatching and job release.

The C implementation of the model is a discrete-event simulation program called FabSim. Every aspect of the model is implemented in FabSim. Moreover, the program allows numerous dispatching and release control combinations, including optimal multi-rule dispatching with Hooke and Jeeves pattern search for optimization. Detailed statistics are collected and displayed for analysis.

Finally, several improvements to both the model and the simulation program are suggested.

# Chapter 5 — 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 re-entrant 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 irretreviable 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 Chapter 6, 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. [Bur86a] observe that there is much to be gained by reducing the variability of inter-arrival times of jobs. Wein [Wei86a] suggests 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

[Fre86a] 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 Chapter 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.

- All random variables (e.g. time between machine failures, time to repair and fraction of rework) are stationary.
- The desired output rate is constant (at least for intervals long compared to mean flow time).
- 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 that has the minimum expected idle time. For 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 increasesd by trying to keep non-bottleneck stations busy. We later allow a second station to have proportion of idle time close to that of the bottleneck station (we call it a second bottleneck).

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. 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 (see Fig. 5.1).

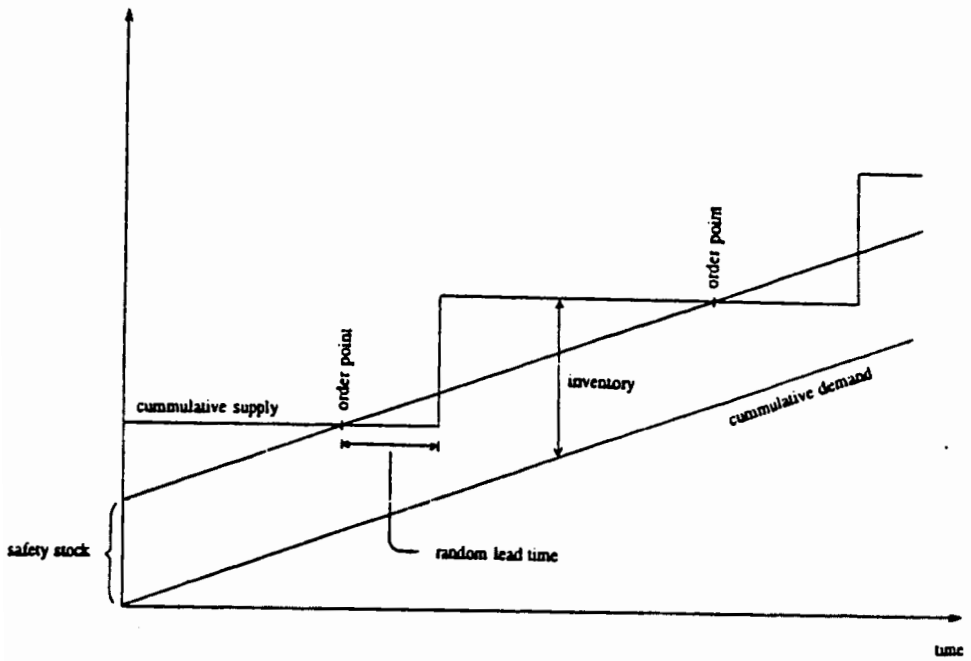


Fig. 5.1 – Safety Stock Notion in Inventory Control

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 the expected 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 (see Fig. 5.2).

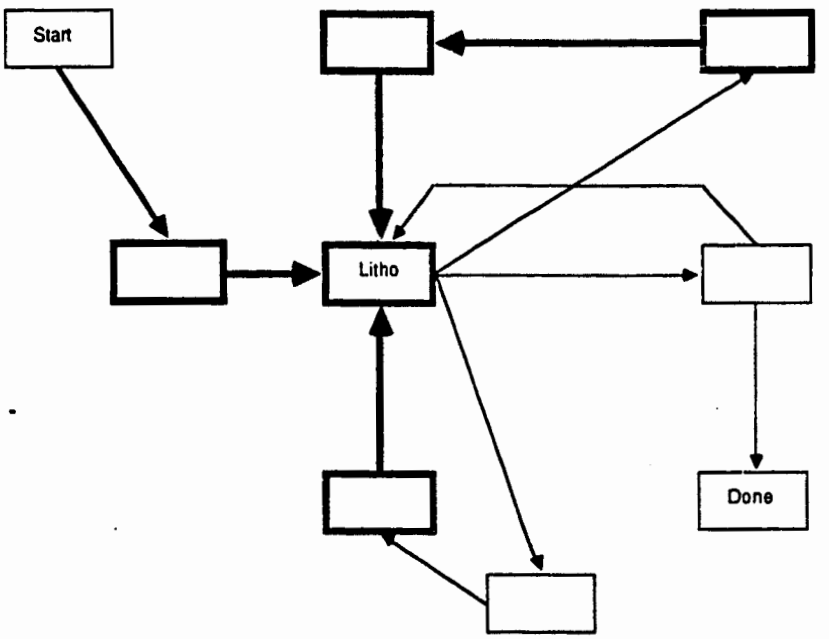


Fig. 5.2 – Virtual Inventory at Photolithography

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

We now formalize the ideas described above. We begin by describing a release strategy for the one product, one bottleneck work station case. We term this strategy SA(1,1). We describe an accompanying dispatching scheme, designed to boost the performance of SA(1,1). We then generalize this release policy to three other cases, namely: SA(N,1), the multi-product, one bottleneck work station case; SA(1,2), the one product, two bottleneck work station case; and finally SA(N,2), the multi-product, two bottleneck work station case.

### 5.1 — One Product One Bottleneck Station Case — SA(1,1)

Consider a job shop with one product and one bottleneck work station. Denote the bottleneck station by $B$. Station $B$ has $m$ unreliable machines with mean times to repair $r_1, r_2, \ldots, r_m$, respectively. Jobs are assigned a deterministic recipe of $p$ steps

$$\rho = s_1, s_2, \ldots, s_p. \tag{5.1}$$

Let $\eta_i$ be the number of jobs currently at step $i$ (either queued or in process). The $i$-th step, $s_i$, has an associated work station $w_i$ and duration $d_i$. Let $i_0$ be the step number corresponding to the first visit to station $B$,

$$i_0 = \min ( i \mid w_i = B ). \tag{5.2}$$

Let $S_B$ be the set of bottleneck work station steps,

$$S_B = (s_i \mid w_i = B ), \tag{5.3}$$

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

$$F = ( s_i \mid i = 1,\ldots,i_0-1 ). \tag{5.4}$$

This Page Left Blank

Define C to be the total processing time from start (step $s_1$) to the first visit to station $B$, excluding step $s_{i_0}$,

$$C = \sum_{i=1}^{i_0-1} d_i \tag{5.5}$$

Let $n_i$ be the step number of the next visit to station $B$ given the job is presently at step $s_i$. Furthermore, let $P$ be the set of steps such that total processing of that step plus future steps prior to the next visit to station $B$ (excluding processing at station $B$) is less than $\beta C$, where $\beta > 0$,

$$P = ( s_i \mid \sum_{j=i}^{n_i-1} d_j < \beta C ) \tag{5.6}$$

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 $D(B)$ be the index set of machines under repair at the bottleneck work station. Define the expected total repair time of equipment at the bottleneck to be

$$R = \sum_{j \in D(B)} r_j. \tag{5.7}$$

Let $W$ be the total work content at the next visit to the bottleneck station of jobs at steps in set $Q$ plus the total expected repair time of equipment down at the bottleneck, $R$,

$$W = \frac{\left\{ R + \sum_{j \in Q} \eta_j d_{n_j} \right\}}{m}. \tag{5.8}$$

$W$ can be interpreted as the expected time to starve the bottleneck. If $W$ falls below a critical value $\alpha\, C$, where $\alpha > 0$ is a safety factor, there is danger that the bottleneck will starve. Release strategy SA(1,1) triggers a job start at this time, to avoid starvation of the bottleneck.

### 5.2 – The SA Dispatching Rule

The main objective of the SA release strategy described in Section 5.1 is the minimization of idle time at bottleneck station machines. 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, $s_1, s_2, \ldots, s_{i_0-1}$, (such as the SRPT rule) will make a negative contribution to SA release when there is danger of bottleneck starvation. An ideal dispatching scheme would aid SA release in achieving its objective, by giving priority to jobs headed for the bottleneck station, when that station is in danger of starving, and giving priority to other lots when there is no imminent danger of starvation. We present such a rule in this Section.

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 shortest remaining processing time rule (SRPT) 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.

Let $p_{SRPT}$ and $\bar{p}_{SRPT}$ be the unnormalized and normalized priority functions of SRPT, respectively. Let $p_{SA}\cdot$ and $\bar{p}_{SA}\cdot$ the unnormalized and normalized priority functions of $SA^+$, respectively. Here, both normalizations are assumed to be by max-

imum value, *i.e.* the normalized priority functions are obtained by dividing the unnormalized values by the maximum absolute value the priority functions can return. Clearly the range of these functions is the real interval [0,1].

The reader can refer to Table 4.1 for a definition of the SRPT priority function. Given a lot is at its $i$-th process step, the $SA^+$ priority function is defined by the ratio of the duration of the next visit to the bottleneck to an estimate of the time until the next bottleneck visit:

$$p_{SA^+} = \frac{d_{n_i}}{\sum_{j=i}^{n_i-1} d_j} \tag{5.9}$$

If there is no future visit to the bottleneck $p_{SA^+}$ is defined to be 0. 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:

$$\bar{p}_{SA} = (1 - \gamma)\,\bar{p}_{SRPT} + \gamma\,(1 - \bar{p}_{SA^+}), \tag{5.10}$$

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

We require the dynamic weight $\gamma$ 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 in not immediate danger of starving (forcing SRPT to be used to dispatch). This can be achieved in the following way.

Define $\tau$ to be the critical work content (a constant parameter) as defined in Section 5.1,

$$\tau = \alpha C, \tag{5.11}$$

and let $W$ be the total work content at the next visit to the bottleneck station of jobs at steps in set $Q$ plus the total expected repair time of equipment in *down* state at the bottleneck, as defined in Section 5.1. 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

$$\gamma = \frac{\left\{\frac{1}{1 + e^{-\tau/W}} - \frac{1}{2}\right\}}{\frac{1}{2}} \tag{5.12}$$

Definition 5.12 produces a dynamic weight function having the desired properties, *i.e.* $\gamma \to 0$ if $W \gg \tau$ and $\gamma \to 1$ if $W \ll \tau$ (Fig. 5.3 illustrates the dynamic weight function for $\tau = 10$).

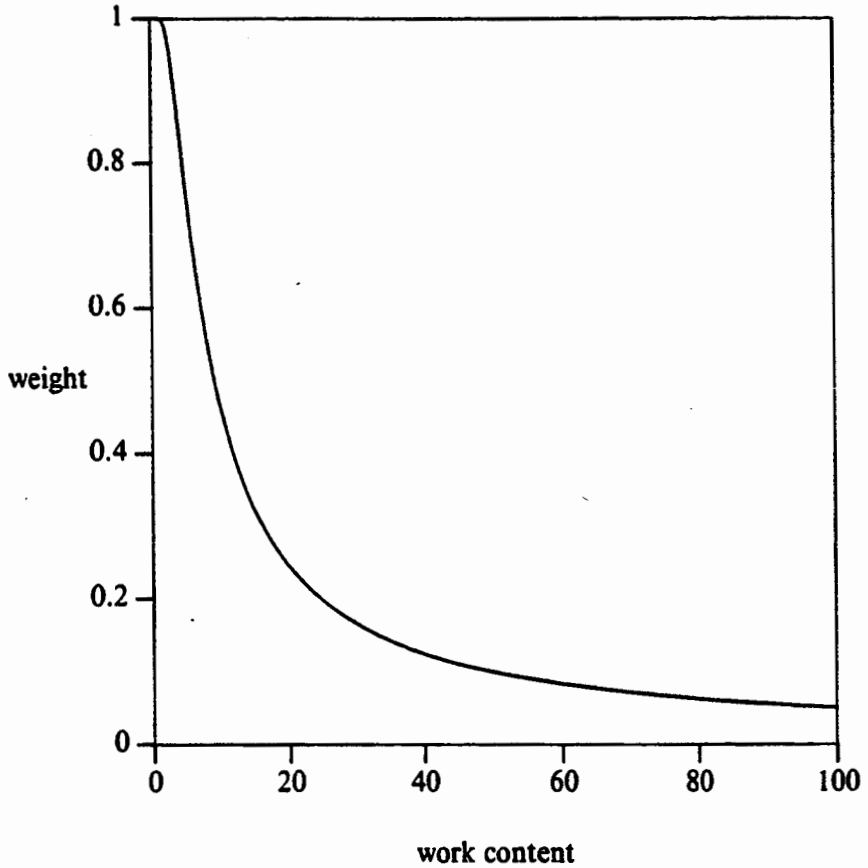


Fig. 5.3 - Dynamic Weight $\gamma$

### 5.3 – Multi Product One Bottleneck Station Case – SA(N,1)

We now consider the case where there is one bottleneck work station (denoted $B$) and $N$ products. The definitions in this Section are the N-product generalizations of the definitions of Section 5.1. Jobs of product type $j$ are assigned a deterministic recipe of $p_j$ steps

$$p_j = s_{j1}, s_{j2}, \ldots, s_{jp_j}. \tag{5.13}$$

Station $B$ has $m$ unreliable machines with mean times to repair $r_1, r_2, \ldots, r_m$, respectively. Let $\eta_{ji}$ be the number of type $j$ jobs currently at step $i$ (either queued or in

process) of process recipe $j$. The $i$-th step of a type $j$ product recipe, $s_{ji}$, has an associated work station $w_{ji}$ and duration $d_{ji}$. Let $i_{j0}$ be the step number corresponding to the first visit of a type $j$ job to station $B$,

$$i_{j0} = \min ( i \mid w_{ji} = B ), j = 1,..., N. \tag{5.14}$$

Let $S_B(j)$ be the set of bottleneck work station steps for type $j$ jobs,

$$S_B(j) = (s_{ji} \mid w_{ji} = B ), j = 1,..., N, \tag{5.15}$$

and let $F_j$ be the set of type $j$ steps prior to the first visit of a type $j$ job to station $B$,

$$F_j = ( s_{ji} \mid i = 1,...,i_{j0}-1 ), j = 1,..., N. \tag{5.16}$$

Define $C_j$ to be the total processing time from the start of a type $j$ job (step $s_{j1}$) to its first visit to station $B$, excluding step $s_{ji_{j0}}$,

$$C_j = \sum_{i=1}^{i_{j0}-1} d_{ji}, j = 1,..., N, \tag{5.17}$$

and let

$$C_{\max} = \max (C_j \mid j = 1,..., N). \tag{5.18}$$

Let $n_{ji}$ be the step number of the next visit of a type $j$ job to station $B$ given the job is presently at step $s_{ji}$ ($i$-th step of type $j$ product recipe). Furthermore, let $P_j$ be the set of type $j$ steps such that total processing of that step plus all other type $j$ steps prior to the next visit of a type $j$ job to station $B$ (excluding processing at station $B$) is less than $\beta C_j$, where $\beta > 0$,

$$P_j = ( s_{ji} \mid \sum_{k=i}^{n_j-1} d_{jk} < \beta\, C_j ),\ j = 1,\ldots, N. \tag{5.19}$$

Let

$$Q_j = F_j \cup P_j \cup S_B(j),\ j = 1,\ldots, N, \tag{5.20}$$

be the set of critical steps for type $j$ jobs, *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 $B$.

As before, let $D(B)$ be the index set of machines in *down* state at the bottleneck work station. Define the expected total repair time of equipment at the bottleneck to be given by equation 5.7.

Let $W$ be the sum over all product types $j$ of the total $j$ type work content at the next visit to the bottleneck station of type $j$ jobs at steps in set $Q_j$ plus the total expected repair time of equipment in *down* state at the bottleneck, $R$,

$$W = \frac{\left\{ R + \sum_{j=1}^{N} ( \sum_{k \in Q_j} \eta_{jk} d_{jn_k} ) \right\}}{m}. \tag{5.21}$$

If $W$ falls below a critical value $\alpha\, C_{\max}$, where $\alpha > 0$ is a safety factor, there is danger that the bottleneck will starve. Release strategy SA(N,1) triggers a job start at this time, to avoid starvation of the bottleneck. The job type with greatest negative deviation from its target mix is selected.

Dispatching policy *SA* can be used with release strategy SA(N,1).

### 5.4 – One Product Two Bottleneck Station Case – SA(1,2)

We now consider a job shop with one product but two bottleneck work stations. The notation of this Section is similar to that of Section 5.1, where lot starts were

triggered when the virtual work content at the bottleneck fell below a given value. Here, lots will be started when the virtual queue at either of the two bottleneck stations falls below given values.

Denote the bottleneck stations by $B_1$ and $B_2$. Station $B_1$ ($B_2$) has $m_1$ ($m_2$) unreliable machines with mean times to repair $r_{11}, r_{12}, \ldots, r_{1m_1}$ ($r_{21}, r_{22}, \ldots, r_{2m_2}$). Jobs are assigned a deterministic recipe of $p$ steps

$$\rho = s_1, s_2, \ldots, s_p. \tag{5.22}$$

We again assume that $\eta_i$ is the number of jobs currently at step $i$ (either queued or in process), and that the $i$-th step of the recipe is carried out at work station $w_i$ and requires $d_i$ units of processing time. Let step $i_{01}$ ($i_{02}$) be the step number corresponding to the first visit to station $B_1$ ($B_2$).

$$i_{01} = \min ( i \mid w_i = B_1 ), \tag{5.23}$$

$$i_{02} = \min ( i \mid w_i = B_2 ). \tag{5.24}$$

Let $S_{B_1}$ ($S_{B_2}$) be the set of steps carried out at station $B_1$ ($B_2$),

$$S_{B_1} = (s_i \mid w_i = B_1 ), \tag{5.25}$$

$$S_{B_2} = (s_i \mid w_i = B_2 ), \tag{5.26}$$

and let $F_1$ ($F_2$) be the set of steps prior to the first visit to station $B_1$ ($B_2$)

$$F_1 = ( s_i \mid i = 1,\ldots,i_{01}-1 ), \tag{5.27}$$

$$F_2 = ( s_i \mid i = 1,\ldots,i_{02}-1 ). \tag{5.28}$$

Define $C_1$ ($C_2$) to be the total processing time from start (step $s_1$) to the first visit to station $B_1$ ($B_2$). We exclude step $s_{i_{01}}$ ($s_{i_{02}}$) from the computation of $C_1$ ($C_2$),

$$C_1 = \sum_{i=1}^{i_{01}-1} d_i \tag{5.29}$$

$$C_2 = \sum_{i=1}^{i_{02}-1} d_i \tag{5.30}$$

Let $n_{i1}$ ($n_{i2}$) be the step number of the next visit of a job to station $B_1$ ($B_2$) given the job is presently at step $s_i$ of its process recipe. Furthermore, define $P_1$ ($P_2$) to be the set of steps such that total processing of that step and future steps prior to the next visit to station $B_1$ ($B_2$) (excluding processing at the bottleneck) is less than $\beta_1 C_1$ ($\beta_2 C_2$) where $\beta_1 > 0$ ($\beta_2 > 0$),

$$P_1 = ( s_i \mid \sum_{j=i}^{n_{i1}-1} d_j < \beta_1 C_1 ) \tag{5.31}$$

$$P_2 = ( s_i \mid \sum_{j=i}^{n_{i2}-1} d_j < \beta_2 C_2 ). \tag{5.32}$$

Let

$$Q_1 = F_1 \cup P_1 \cup S_{B_1} \tag{5.33}$$

$$Q_2 = F_2 \cup P_2 \cup S_{B_2} \tag{5.34}$$

be the sets of critical steps (for bottlenecks $B_1$ and $B_2$, respectively), *i.e.* steps that are within a critical time factor of the bottleneck stations or steps in the pipeline from start to the first visit to the bottlenecks. Let $D(B)$ be the index set of machines under repair at the bottleneck work station $B$. Define the expected total repair time

of equipment at the bottleneck $B_1$ ($B_2$) to be

$$R_1 = \sum_{j \in D(B_1)} r_{j1}, \tag{5.35}$$

$$R_2 = \sum_{j \in D(B_2)} r_{j2}. \tag{5.36}$$

Let $W_1$ ($W_2$) be the total work content at the next visit to bottleneck station $B_1$ ($B_2$) of jobs at steps in set $Q_1$ ($Q_2$) plus the total expected repair time of equipment down at the bottleneck station, $R_1$ ($R_2$),

$$W_1 = \frac{\left\{ R_1 + \sum_{j \in Q_1} \eta_j d_{n_j 1} \right\}}{m_1}, \tag{5.37}$$

$$W_2 = \frac{\left\{ R_2 + \sum_{j \in Q_2} \eta_j d_{n_j 1} \right\}}{m_2}. \tag{5.38}$$

If $W_1$ ($W_2$) falls below a critical value $\alpha_1 C_1$ ($\alpha_2 C_2$), where $\alpha_1 > 0$ ($\alpha_2 > 0$) is a safety factor, there is danger that bottleneck $B_1$ ($B_2$) will starve. Release strategy SA(1,2) triggers a job start at this time, to avoid starvation of that bottleneck.

### 5.5 – Multi Product Two Bottleneck Station Case – SA(N,2)

In this Section we consider our last generalization of SA, namely when there are two bottleneck work stations ($B_1$ and $B_2$) and $N$ products. The definitions in this Section are the N-product generalizations of the definitions of Section 5.4.

Jobs of product type $j$ are assigned a deterministic recipe of $p_j$ steps

$$\rho_j = s_{j1}, s_{j2}, \ldots, s_{jp_j}. \tag{5.39}$$

As before, station $B_1$ ($B_2$) has $m_1$ ($m_2$) unreliable machines with mean times to repair $r_{11}, r_{12}, ..., r_{1m_1}$ ($r_{21}, r_{22}, ..., r_{2m_2}$). Let $\eta_{ji}$ be the number of type $j$ jobs currently at step $i$ (either queued or in process) of process recipe $j$. The $i$-th step of a type $j$ product recipe, $s_{ji}$, is carried out at work station $w_{ji}$ and requires $d_{ji}$ units of processing time. Let $i_{j1}^0$ ($i_{j2}^0$) be the step number corresponding to the first visit of a type $j$ job to station $B_1$ ($B_2$).

$$i_{j1}^0 = \min ( i \mid w_{ji} = B_1 ), j = 1,..., N, \tag{5.40}$$

$$i_{j2}^0 = \min ( i \mid w_{ji} = B_2 ), j = 1,..., N. \tag{5.41}$$

Let $S_{B_1}(j)$ ($S_{B_2}(j)$) be the set of processing steps at bottleneck station $B_1$ ($B_2$) for type $j$ jobs,

$$S_{B_1}(j) = (s_{ji} \mid w_{ji} = B_1 ), j = 1,..., N, \tag{5.42}$$

$$S_{B_2}(j) = (s_{ji} \mid w_{ji} = B_2 ), j = 1,..., N, \tag{5.43}$$

and let $F_{j1}$ ($F_{j2}$) be the set of type $j$ steps prior to the first visit of a type $j$ job to station $B_1$ ($B_2$),

$$F_{j1} = ( s_{ji} \mid i = 1,...,i_{j1}^0 - 1 ), j = 1,..., N, \tag{5.44}$$

$$F_{j2} = ( s_{ji} \mid i = 1,...,i_{j2}^0 - 1 ), j = 1,..., N. \tag{5.45}$$

Define $C_{j1}$ ($C_{j2}$) to be the total processing time from the start of a type $j$ job (step $s_{j1}$) to its first visit to station $B_1$ ($B_2$), excluding the first visit to each bottleneck station.

$$C_{j1} = \sum_{i=1}^{i_{j1}^0 - 1} d_{ji}, \; j = 1,..., N, \tag{5.46}$$

$$C_{j2} = \sum_{i=1}^{i_{j2}^0 - 1} d_{ji}, \; j = 1,..., N, \tag{5.47}$$

and let

$$C_1^{\max} = \max (C_{j1} \mid j = 1,..., N), \tag{5.48}$$

$$C_2^{\max} = \max (C_{j2} \mid j = 1,..., N). \tag{5.49}$$

Let $n_{ji}^1$ ($n_{ji}^2$) be the step number of the next visit of a type $j$ job to station $B_1$ ($B_2$) given the job is presently at step $s_{ji}$ ($i$-th step of type $j$ product recipe). Furthermore, let $P_{j1}$ ($P_{j2}$) be the sets of type $j$ steps such that total processing of that step plus all other type $j$ steps prior to the next visit of a type $j$ job to station $B_1$ ($B_2$) (excluding processing at the corresponding bottleneck station) is less than $\beta_1 C_{j1}$ ($\beta_2 C_{j2}$), where $\beta_1 > 0$ ($\beta_2 > 0$).

$$P_{j1} = ( s_{ji} \mid \sum_{l=i}^{n_{ji}^1 - 1} d_{jl} < \beta_1 C_{j1} ), \; j = 1,..., N, \tag{5.50}$$

$$P_{j1} = ( s_{ji} \mid \sum_{l=i}^{n_{ji}^1 - 1} d_{jl} < \beta_2 C_{j1} ), \; j = 1,..., N. \tag{5.51}$$

Let

$$Q_{j1} = F_{j1} \cup P_{j1} \cup S_{B_1}(j), \; j = 1,..., N, \tag{5.52}$$

$$Q_{j2} = F_{j2} \cup P_{j2} \cup S_{B_2}(j), \; j = 1,..., N, \tag{5.53}$$

be the set of critical steps for type $j$ jobs, *i.e.* steps that are within a critical time

factor of bottleneck station $B_1$ ($B_2$) or steps in the pipeline from start to the first visit to a specific bottleneck station.

As before, let $D(B)$ be the index set of machines in *down* state at work station $B$. Define the expected total repair time of equipment at a bottleneck to be given by equation 5.7.

Let $W_1$ ($W_2$) be the sum over all product types of the total $j$ type work content at the next visit to bottleneck station $B_1$ ($B_2$) of type $j$ jobs at steps in set $Q_{j1}$ ($Q_{j2}$) plus the total expected repair time $R_1$ ($R_2$) of equipment in *down* state at station $B_1$ ($B_2$),

$$W_1 = \frac{\left\{ R_1 + \sum_{j=1}^{N} \left( \sum_{l \in Q_{j1}} \eta_{jl} d_{jn_j^l} \right) \right\}}{m_1}, \tag{5.54}$$

$$W_2 = \frac{\left\{ R_2 + \sum_{j=1}^{N} \left( \sum_{l \in Q_{j2}} \eta_{jl} d_{jn_j^l} \right) \right\}}{m_2}. \tag{5.55}$$

If $W_1$ ($W_2$) falls below a critical value $\alpha_1 C_1^{\max}$ ($\alpha_2 C_2^{\max}$), where $\alpha_1 > 0$ ($\alpha_2 > 0$) is a safety factor, there is danger that bottleneck station $B_1$ ($B_2$) will starve. Release strategy SA(N,2) triggers a job start at this time, to avoid starvation of that bottleneck. The job type with greatest negative deviation from its target mix is selected.

### 5.6 – Summary

In this Chapter we introduced a class of new job release mechanisms called Starvation Avoidance (SA). These procedures make an analogy between work content at one or more bottleneck workstation (in a job shop) and inventory (in inventory control). In inventory control, when the the difference between items (on hand and on order) and demand is less than a given value, a new item is ordered. In this analogy,

when work content (on hand) at the bottleneck and work content (on order) moving towards the bottleneck (and expected to arrive within a limited time frame) falls below a given value, a new job is released into the shop.

We present four special cases for SA: the one product one bottleneck case, SA(1,1); the multi-product one bottleneck case, SA(N,1); the one product two bottleneck case, SA(1,2); and the multi-product two bottleneck case, SA(N,2). We discuss a dispatching scheme designed to aid SA achieve its objective, *i.e.* minimize idle time on critical shop equipment.

Extensive testing was done on one case of Starvation Avoidance: SA(1,1). These results are presented in the next Chapter.

# Chapter 6 — Simulation Experiments

In this Chapter we compare several scheduling policies, *i.e.* release and dispatching schemes, on a number of wafer fab queueing networks. We limit ourselves to studying fabs with one process flow and a single bottleneck station. Mean delay/throughput tradeoff curves are generated for each policy-network pair under consideration. Simulations are carried out using FabSim on the Cray X-MP/14 supercomputer at U.C. - Berkeley.

| 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 |

Table 6.1 — Test Fab Queueing Networks

In Section 6.1 we discuss the experimental design. Table 6.1 summarizes the fab networks tested in this study. The first fab queueing network that we consider (in Section 6.2) is a small but representative hypothetical fab. This fab network consists of five work stations (one being the hub bottleneck) and a single product with a 29-step recipe. On this system we compare several dispatching and release strategies. In the remaining Sections we simulate larger queueing networks and compare Starvation Avoidance (release and dispatching) with UNIF release and either FIFO or SRPT dispatching. In Section 6.3 we adapt a fab network proposed by Dayhoff [Day87a].

In Section 6.4 we simulate a hypothetical 24 station, 49 step bipolar process fab suggested by Barry [Bar86a]. In Sections 6.5 and 6.6 we consider two 41 station, 182 step process fab networks adapted from data collected from a computer integrated manufacturing system at a Silicon Systems production fab. Both networks have one bottleneck station: the first has a bottleneck at ion implantation while the second has its bottleneck at the station of aligners.

### 6.1 – Experimental Design

To generate each average delay/throughput tradeoff curve we require the simulation of several levels of traffic intensities on the same wafer fab queueing network. One possible approach is be to run several *short* simulation experiments with different random seed sets for each configuration. We select a different approach, namely, to run a single *long* run for each traffic intensity with a single random seed set. We choose not to vary the random seed for the different traffic intensity levels as a measure of variance reduction, since this way scheduling policies are compared on the same sequence of machine up and down intervals. Within each run, statistics are collected in job batches of 50 jobs for the purpose of constructing a confidence interval. (See Law and Kelton [Law82a] for a discussion of confidence interval construction using this technique.)

For all runs, job statistics are collected for jobs that are released between simulation clock times $0.05H$ and $H$, where $H$ is the given simulation horizon. $H$ will vary according to the network being simulated to accommodate for the desired minimum sample size. Due to the large variability in uptimes and repair times introduced by the exponential distributions used in FabSim, it is necessary to collect statistics for a large number of jobs so as to generate acceptable confidence intervals. We collect statistics for at least 1000 jobs per run and in some cases for over 4000 jobs per run. Both axes of the delay/throughput tradeoff curves are scaled. The horizontal axis corresponds to the mean throughput. We scale throughput values such

that 1 corresponds to the maximum expected output of the fab. The vertical axis corresponds to mean delay. Entries there are divided by the total recipe processing time. All inter-station travel is assumed to be instantaneous.

### 6.2 – A Small Wafer Fab Queueing Network with Hub Station - Small-Fab

In this Section we compare several scheduling policies on a small hypothetical wafer fab queueing network (Small-Fab), whose topology is given in Fig. 6.2.1.

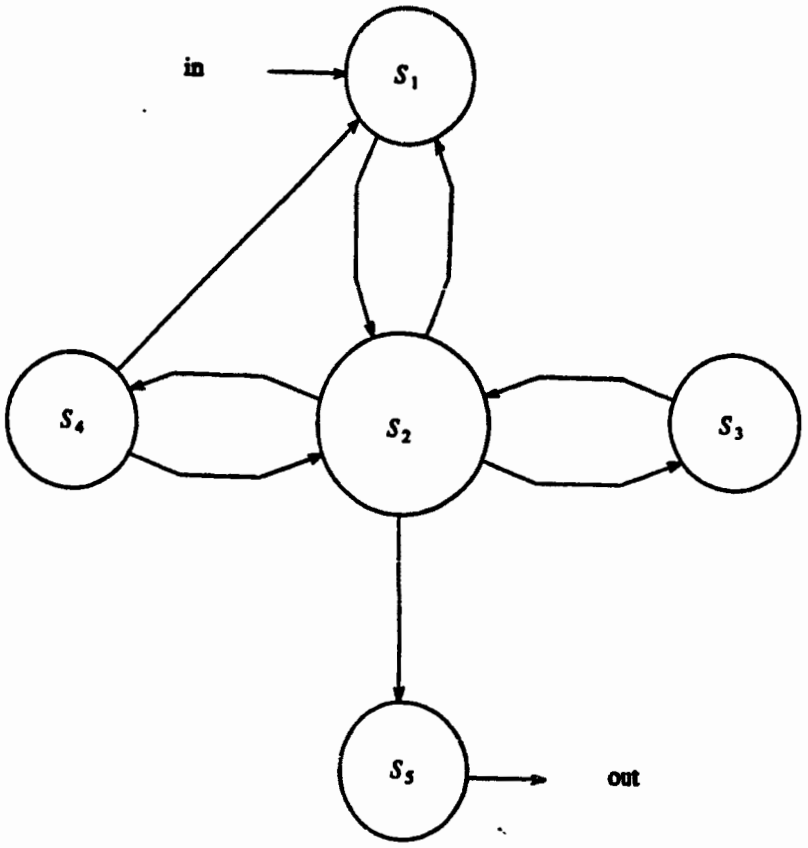


Fig. 6.2.1 – Network Topology - Small-Fab

This fab has five work stations that are summarized in Table 6.2.1. The bottleneck station is $S_2$. We assume all machines have identical values of mean time between failures (MTBF) of 12,240 minutes and mean time to repair (MTTR) of 1,440 minutes. The load size for all machines is one lot.

| work station | number machines |
|---|---|
| 1 | 5 |
| 2 | 2 |
| 3 | 6 |
| 4 | 6 |
| 5 | 10 |

Table 6.2.1 — Work Stations - Small-Fab

This fab network has one job type, whose 12 step process recipe is described in Table 6.2.2. The total duration of processing is 1,150 minutes.

| step number | work station | step duration (mins) |
|---|---|---|
| 1 | 1 | 60 |
| 2 | 2 | 120 |
| 3 | 3 | 60 |
| 4 | 2 | 90 |
| 5 | 3 | 90 |
| 6 | 2 | 120 |
| 7 | 4 | 60 |
| 8 | 2 | 60 |
| 9 | 4 | 90 |
| 10 | 1 | 60 |
| 11 | 2 | 100 |
| 12 | 5 | 240 |

Table 6.2.2 — Process Recipe - Small-Fab

No hot lots are allowed to start in this network. We also assume rework is never required. The nominal capacity of the network (*i.e.* assuming no equipment failures) is a production rate of one lot every 245 minutes (41.14 lots/week, assuming a 168-hour week). Its expected capacity (taking into account machine availability) is one lot every 277.67 minutes (36.30 lots/week).

We simulate 13 scheduling policies by making all possible combinations of release policies UNIF, SA, WR, and Fixed-WIP, with dispatching rules FIFO, SRPT, and SIPT. For SA release we also simulate the SA dispatching policy of Section 5.2. We use a simulation horizon of 500 thousand minutes (about 11 months and 13 days of a 24 hour/day, 7 days/week regime) requiring approximately 20 seconds of CPU time on the Cray X-MP/14 for each run. Tables 6.2.3 through 6.2.15 summarize the runs. The different system loads are controlled by start rates for UNIF, critical values for SA and WR, and WIP levels for Fixed-WIP.

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 400 | 223.46 | 84.30 | 383.62 | 16.00 | 1188 |
| 390 | 254.69 | 93.60 | 451.22 | 17.96 | 1218 |
| 380 | 295.43 | 125.57 | 514.35 | 20.44 | 1250 |
| 370 | 351.79 | 145.33 | 591.21 | 23.05 | 1284 |
| 360 | 448.08 | 198.22 | 763.71 | 27.82 | 1319 |
| 350 | 506.25 | 265.31 | 932.87 | 30.47 | 1357 |
| 340 | 611.66 | 331.03 | 1161.00 | 34.08 | 1397 |
| 330 | 727.80 | 429.07 | 1412.19 | 38.16 | 1440 |
| 320 | 874.22 | 479.47 | 1656.17 | 42.65 | 1484 |
| 310 | 1152.57 | 596.14 | 2000.25 | 49.56 | 1532 |
| 300 | 1527.79 | 689.06 | 2375.52 | 56.57 | 1583 |
| 290 | 2350.49 | 850.64 | 2904.76 | 66.66 | 1638 |
| 280 | 5380.26 | 1191.82 | 4115.79 | 82.01 | 1696 |

Table 6.2.3.a. — UNIF/FIFO Scheduling (part a) – Small-Fab

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 400 | 399.97 | 139.64 | 0.55 | 1.06 |
| 390 | 389.89 | 140.60 | 0.65 | 1.25 |
| 380 | 379.95 | 150.44 | 0.78 | 1.44 |
| 370 | 369.85 | 157.40 | 0.93 | 1.67 |
| 360 | 359.87 | 123.85 | 1.23 | 2.19 |
| 350 | 349.83 | 127.78 | 1.44 | 2.72 |
| 340 | 339.78 | 125.15 | 1.75 | 3.45 |
| 330 | 329.58 | 133.58 | 2.15 | 4.31 |
| 320 | 319.67 | 139.46 | 2.68 | 5.20 |
| 310 | 309.48 | 145.61 | 3.65 | 6.47 |
| 300 | 299.45 | 148.56 | 4.99 | 7.92 |
| 290 | 289.37 | 150.86 | 7.94 | 10.02 |
| 280 | 279.26 | 138.07 | 18.73 | 14.68 |

Table 6.2.3.b. — UNIF/FIFO Scheduling (part b) – Small-Fab

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 400 | 183.88 | 58.45 | 282.17 | 13.57 | 1188 |
| 390 | 209.05 | 69.13 | 331.20 | 15.25 | 1218 |
| 380 | 237.88 | 92.53 | 372.94 | 17.14 | 1250 |
| 370 | 278.13 | 100.46 | 411.67 | 19.16 | 1284 |
| 360 | 317.64 | 123.08 | 494.77 | 21.47 | 1319 |
| 350 | 366.71 | 164.56 | 588.73 | 24.10 | 1357 |
| 340 | 420.45 | 192.16 | 698.39 | 26.26 | 1397 |
| 330 | 494.27 | 254.29 | 853.60 | 29.54 | 1440 |
| 320 | 595.85 | 292.68 | 1023.89 | 33.65 | 1484 |
| 310 | 754.10 | 349.79 | 1187.33 | 39.15 | 1532 |
| 300 | 994.09 | 410.84 | 1423.35 | 45.88 | 1583 |
| 290 | 1495.27 | 495.16 | 1704.10 | 56.00 | 1638 |
| 280 | 3253.79 | 682.63 | 2356.20 | 73.40 | 1696 |
| 270 | 9867.40 | 1739.04 | 6012.92 | 89.58 | 1759 |

Table 6.2.4.a. — UNIF/SRPT Scheduling (part a) – Small-Fab

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 400 | 400.07 | 132.77 | 0.45 | 0.85 |
| 390 | 389.87 | 129.18 | 0.53 | 0.98 |
| 380 | 379.96 | 131.03 | 0.63 | 1.10 |
| 370 | 369.85 | 134.86 | 0.74 | 1.24 |
| 360 | 359.86 | 139.04 | 0.87 | 1.48 |
| 350 | 349.75 | 103.25 | 1.04 | 1.78 |
| 340 | 339.70 | 107.37 | 1.20 | 2.13 |
| 330 | 329.62 | 105.21 | 1.46 | 2.65 |
| 320 | 319.56 | 112.61 | 1.82 | 3.25 |
| 310 | 309.51 | 109.62 | 2.39 | 3.88 |
| 300 | 299.48 | 115.28 | 3.25 | 4.79 |
| 290 | 289.54 | 123.84 | 5.05 | 5.92 |
| 280 | 279.53 | 127.76 | 11.33 | 8.45 |
| 270 | 277.36 | 128.30 | 35.84 | 22.27 |

Table 6.2.4.b. — UNIF/SRPT Scheduling (part b) – Small-Fab

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 400 | 202.11 | 67.07 | 324.42 | 14.70 | 1188 |
| 390 | 229.29 | 77.80 | 378.70 | 16.47 | 1218 |
| 380 | 259.47 | 97.99 | 414.86 | 18.41 | 1250 |
| 370 | 317.19 | 127.74 | 505.22 | 21.28 | 1284 |
| 360 | 375.50 | 159.88 | 628.73 | 24.40 | 1319 |
| 350 | 415.69 | 197.96 | 706.30 | 26.47 | 1357 |
| 340 | 495.61 | 249.07 | 885.16 | 29.54 | 1397 |
| 330 | 577.95 | 317.32 | 1054.76 | 32.90 | 1440 |
| 320 | 699.12 | 362.33 | 1261.65 | 37.30 | 1484 |
| 310 | 893.53 | 435.67 | 1474.45 | 43.25 | 1532 |
| 300 | 1180.70 | 509.68 | 1762.81 | 50.17 | 1583 |
| 290 | 1782.97 | 621.72 | 2132.83 | 60.27 | 1638 |
| 280 | 4001.17 | 869.37 | 3001.76 | 77.23 | 1696 |
| 270 | 12417.08 | 2210.58 | 7643.01 | 91.54 | 1759 |

Table 6.2.5.a. — UNIF/SIPT Scheduling (part a) – Small-Fab

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 400 | 399.98 | 163.76 | 0.50 | 0.92 |
| 390 | 389.88 | 152.95 | 0.58 | 1.08 |
| 380 | 379.94 | 157.49 | 0.68 | 1.20 |
| 370 | 369.83 | 169.41 | 0.84 | 1.46 |
| 360 | 359.76 | 158.14 | 1.03 | 1.82 |
| 350 | 349.83 | 137.78 | 1.18 | 2.09 |
| 340 | 339.76 | 141.85 | 1.42 | 2.65 |
| 330 | 329.69 | 139.47 | 1.71 | 3.24 |
| 320 | 319.61 | 141.86 | 2.14 | 3.98 |
| 310 | 309.49 | 149.96 | 2.83 | 4.79 |
| 300 | 299.44 | 155.34 | 3.86 | 5.90 |
| 290 | 289.43 | 164.43 | 6.02 | 7.34 |
| 280 | 279.36 | 180.97 | 13.93 | 10.73 |
| 270 | 279.31 | 174.41 | 44.31 | 28.31 |

Table 6.2.5.b. — UNIF/SIPT Scheduling (part b) – Small-Fab

| work in process | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 4 | 191.99 | 31.06 | 229.12 | 14.23 | 1414 |
| 5 | 346.52 | 46.70 | 312.81 | 22.97 | 1588 |
| 6 | 539.76 | 48.38 | 377.01 | 31.80 | 1688 |
| 7 | 789.61 | 58.28 | 423.35 | 40.65 | 1713 |
| 8 | 1063.61 | 67.68 | 469.53 | 47.99 | 1717 |
| 10 | 1616.94 | 84.58 | 541.05 | 58.39 | 1719 |
| 14 | 2723.52 | 118.05 | 656.05 | 70.27 | 1720 |
| 16 | 3276.68 | 132.76 | 698.41 | 73.98 | 1724 |
| 26 | 6038.26 | 206.87 | 946.30 | 84.00 | 1733 |

Table 6.2.6.a. — Fixed-WIP/FIFO Scheduling (part a) – Small-Fab

| work in process | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 4 | 335.10 | 306.47 | 0.58 | 0.86 |
| 5 | 298.70 | 324.42 | 1.16 | 1.14 |
| 6 | 281.14 | 262.11 | 1.92 | 1.33 |
| 7 | 276.83 | 297.46 | 2.85 | 1.42 |
| 8 | 276.45 | 241.32 | 3.84 | 1.37 |
| 10 | 276.36 | 248.19 | 5.84 | 1.35 |
| 14 | 276.31 | 292.02 | 9.84 | 1.37 |
| 16 | 275.57 | 344.24 | 11.84 | 1.41 |
| 26 | 275.30 | 509.04 | 21.83 | 1.66 |

Table 6.2.6.b. — Fixed-WIP/FIFO Scheduling (part b) – Small-Fab

| work in process | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 4 | 169.12 | 33.48 | 233.78 | 12.62 | 1441 |
| 5 | 328.27 | 37.44 | 312.55 | 22.19 | 1602 |
| 6 | 522.78 | 49.84 | 366.40 | 31.25 | 1701 |
| 7 | 787.54 | 60.01 | 420.55 | 40.58 | 1716 |
| 8 | 1063.69 | 68.75 | 459.02 | 47.99 | 1717 |
| 12 | 2168.67 | 103.90 | 593.49 | 65.31 | 1721 |
| 16 | 3271.54 | 138.52 | 715.32 | 73.96 | 1725 |

Table 6.2.7.a. — Fixed-WIP/SRPT Scheduling (part a) – Small-Fab

| work in process | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 4 | 328.91 | 142.78 | 0.51 | 0.81 |
| 5 | 295.34 | 146.02 | 1.11 | 1.08 |
| 6 | 278.70 | 125.57 | 1.88 | 1.23 |
| 7 | 276.44 | 128.98 | 2.84 | 1.25 |
| 8 | 276.37 | 131.40 | 3.84 | 1.25 |
| 12 | 276.28 | 131.33 | 7.84 | 1.25 |
| 16 | 276.23 | 131.26 | 11.84 | 1.26 |

Table 6.2.7.b. — Fixed-WIP/SRPT Scheduling (part b) – Small-Fab

| work in process | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 4 | 186.05 | 29.06 | 233.84 | 13.90 | 1419 |
| 5 | 311.39 | 41.79 | 324.12 | 21.24 | 1624 |
| 6 | 519.03 | 50.65 | 390.38 | 31.07 | 1706 |
| 7 | 787.61 | 60.27 | 466.20 | 40.59 | 1716 |
| 8 | 1063.59 | 67.84 | 484.07 | 47.99 | 1716 |
| 10 | 1616.86 | 84.97 | 560.08 | 58.38 | 1719 |
| 16 | 3274.69 | 136.21 | 724.00 | 73.98 | 1723 |
| 20 | 3827.18 | 153.33 | 784.88 | 76.86 | 1726 |

Table 6.2.8.a. — Fixed-WIP/SIPT Scheduling (part a) – Small-Fab

| work in process | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 4 | 333.65 | 258.14 | 0.56 | 0.82 |
| 5 | 292.10 | 216.41 | 1.07 | 1.09 |
| 6 | 278.06 | 252.37 | 1.87 | 1.28 |
| 7 | 276.60 | 242.96 | 2.84 | 1.31 |
| 8 | 276.32 | 209.14 | 3.84 | 1.30 |
| 10 | 276.29 | 225.92 | 5.84 | 1.30 |
| 16 | 276.26 | 230.90 | 11.84 | 1.33 |
| 20 | 275.98 | 235.03 | 13.84 | 1.33 |

Table 6.2.8.b. — Fixed-WIP/SIPT Scheduling (part b) – Small-Fab

| critical work content | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 200 | 40.68 | 8.93 | 80.66 | 3.40 | 1056 |
| 400 | 90.32 | 19.07 | 146.50 | 7.21 | 1402 |
| 600 | 198.33 | 23.27 | 200.48 | 14.59 | 1590 |
| 800 | 335.36 | 32.63 | 239.49 | 22.47 | 1639 |
| 1000 | 519.86 | 37.36 | 297.66 | 31.09 | 1713 |
| 1400 | 944.28 | 52.52 | 369.92 | 45.04 | 1717 |
| 1800 | 1309.07 | 63.59 | 429.92 | 53.19 | 1717 |
| 2600 | 2119.87 | 88.63 | 538.29 | 64.79 | 1719 |
| 3000 | 2546.18 | 96.33 | 568.13 | 68.85 | 1720 |
| 3400 | 2898.46 | 109.62 | 616.50 | 71.56 | 1721 |

Table 6.2.9.a. — WR/FIFO Scheduling (part a) – Small-Fab

| critical work content | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 200 | 448.77 | 57.11 | 0.10 | 0.39 |
| 400 | 338.15 | 65.61 | 0.28 | 0.56 |
| 600 | 298.09 | 103.06 | 0.67 | 0.81 |
| 800 | 289.23 | 108.07 | 1.16 | 0.99 |
| 1000 | 276.82 | 183.36 | 1.88 | 1.07 |
| 1400 | 276.45 | 94.57 | 3.41 | 0.96 |
| 1800 | 276.42 | 88.02 | 4.73 | 0.97 |
| 2600 | 276.38 | 81.18 | 7.66 | 1.03 |
| 3000 | 276.37 | 116.80 | 9.19 | 1.06 |
| 3400 | 276.25 | 97.61 | 10.47 | 0.98 |

Table 6.2.9.b. — WR/FIFO Scheduling (part b) – Small-Fab

| critical work content | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 200 | 40.07 | 8.82 | 80.15 | 3.35 | 1056 |
| 400 | 87.47 | 17.76 | 135.88 | 6.97 | 1393 |
| 600 | 162.49 | 21.97 | 169.35 | 12.26 | 1496 |
| 800 | 284.23 | 23.04 | 194.62 | 19.79 | 1610 |
| 1400 | 644.05 | 31.16 | 259.44 | 35.86 | 1716 |
| 3400 | 1797.45 | 65.53 | 440.87 | 60.95 | 1720 |
| 5800 | 3127.76 | 109.79 | 612.45 | 73.09 | 1725 |

Table 6.2.10.a. — WR/SRPT Scheduling (part a) - Small-Fab

| critical work content | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 200 | 448.59 | 53.32 | 0.09 | 0.33 |
| 400 | 340.18 | 78.64 | 0.26 | 0.54 |
| 600 | 316.76 | 86.07 | 0.51 | 0.75 |
| 800 | 294.36 | 111.89 | 0.97 | 0.85 |
| 1400 | 276.32 | 131.44 | 2.33 | 0.96 |
| 3400 | 276.26 | 131.37 | 6.50 | 0.96 |
| 5800 | 276.23 | 131.26 | 11.32 | 0.96 |

Table 6.2.10.b. — WR/SRPT Scheduling (part b) - Small-Fab

| critical work content | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 200 | 40.42 | 8.73 | 80.57 | 3.38 | 1056 |
| 400 | 88.59 | 17.73 | 148.94 | 7.85 | 1389 |
| 600 | 177.68 | 24.63 | 197.45 | 13.36 | 1504 |
| 800 | 301.88 | 28.72 | 221.56 | 20.78 | 1656 |
| 1000 | 440.35 | 30.53 | 255.30 | 27.67 | 1707 |
| 1400 | 731.84 | 38.07 | 311.39 | 38.85 | 1716 |
| 2200 | 1336.06 | 54.84 | 397.19 | 53.70 | 1718 |
| 2600 | 1619.52 | 65.61 | 445.95 | 58.44 | 1719 |
| 3400 | 2181.50 | 83.67 | 520.42 | 65.45 | 1720 |
| 4200 | 2751.78 | 101.44 | 585.17 | 70.49 | 1722 |
| 5000 | 3334.95 | 118.66 | 641.44 | 74.33 | 1724 |

Table 6.2.11.a. — WR/SIPT Scheduling (part a) - Small-Fab

| critical work content | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 200 | 448.59 | 56.89 | 0.10 | 0.35 |
| 400 | 341.09 | 134.48 | 0.26 | 0.56 |
| 600 | 315.17 | 113.43 | 0.57 | 0.80 |
| 800 | 286.35 | 135.18 | 1.06 | 0.90 |
| 1000 | 277.72 | 177.70 | 1.59 | 0.95 |
| 1400 | 276.43 | 162.43 | 2.64 | 1.02 |
| 2200 | 276.40 | 141.07 | 4.83 | 0.99 |
| 2600 | 276.34 | 175.32 | 5.85 | 1.01 |
| 3400 | 276.30 | 180.97 | 7.89 | 0.97 |
| 4200 | 276.27 | 163.39 | 9.95 | 1.05 |
| 5000 | 276.13 | 170.21 | 12.06 | 1.01 |

Table 6.2.11.b. — WR/SIPT Scheduling (part b) - Small-Fab

| safety factor × $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.0 | 323.09 | 21.59 | 164.30 | 21.92 | 1628 |
| 1.3 | 459.24 | 16.48 | 189.45 | 28.48 | 1684 |
| 1.6 | 470.89 | 19.16 | 179.80 | 29.03 | 1688 |
| 1.9 | 500.15 | 23.19 | 182.04 | 30.25 | 1693 |
| 2.2 | 657.47 | 27.71 | 227.49 | 36.37 | 1709 |
| 2.8 | 682.50 | 26.34 | 229.46 | 37.20 | 1715 |
| 3.1 | 758.74 | 33.65 | 258.87 | 39.70 | 1717 |
| 3.4 | 760.74 | 32.42 | 245.24 | 39.76 | 1718 |
| 4.9 | 1048.27 | 36.11 | 285.60 | 47.66 | 1721 |
| 5.2 | 1102.40 | 43.63 | 316.35 | 48.91 | 1718 |

Table 6.2.12.a. — SA/FIFO Scheduling (part a) - Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor × $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.0 | 291.65 | 208.69 | 1.11 | 1.05 |
| 1.3 | 282.22 | 178.93 | 1.62 | 1.14 |
| 1.6 | 281.81 | 185.83 | 1.67 | 1.13 |
| 1.9 | 280.59 | 186.23 | 1.78 | 1.16 |
| 2.2 | 277.72 | 235.01 | 2.37 | 1.32 |
| 2.8 | 276.97 | 203.97 | 2.46 | 1.23 |
| 3.1 | 276.85 | 224.06 | 2.74 | 1.25 |
| 3.4 | 276.44 | 218.51 | 2.74 | 1.22 |
| 4.9 | 276.41 | 264.63 | 3.79 | 1.37 |
| 5.2 | 276.32 | 251.85 | 3.99 | 1.39 |

Table 6.2.12.b. — SA/FIFO Scheduling (part b) - Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor × $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.3 | 250.18 | 21.19 | 166.76 | 17.84 | 1640 |
| 2.2 | 480.25 | 27.25 | 238.04 | 29.41 | 1716 |
| 2.8 | 532.19 | 27.09 | 235.78 | 31.62 | 1718 |
| 3.1 | 541.72 | 27.32 | 240.66 | 31.99 | 1718 |
| 3.7 | 640.21 | 29.71 | 255.77 | 35.74 | 1719 |
| 4.3 | 741.47 | 36.52 | 282.34 | 39.18 | 1718 |

Table 6.2.13.a. – SA/SRPT Scheduling (part a) – Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor × $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.3 | 293.46 | 121.34 | 0.85 | 0.86 |
| 2.2 | 276.61 | 137.61 | 1.73 | 0.95 |
| 2.8 | 276.40 | 130.88 | 1.92 | 0.98 |
| 3.1 | 276.39 | 127.95 | 1.96 | 0.97 |
| 3.7 | 276.31 | 125.72 | 2.31 | 0.94 |
| 4.3 | 276.29 | 125.76 | 2.68 | 0.93 |

Table 6.2.13.b. – SA/SRPT Scheduling (part b) – Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor x $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.3 | 243.55 | 16.79 | 168.94 | 17.43 | 1540 |
| 1.6 | 263.66 | 17.41 | 176.05 | 18.66 | 1597 |
| 2.2 | 467.53 | 22.23 | 246.43 | 28.90 | 1702 |
| 2.8 | 488.74 | 23.68 | 248.25 | 29.81 | 1705 |
| 3.4 | 532.80 | 22.55 | 256.70 | 31.65 | 1707 |
| 3.7 | 611.44 | 24.33 | 258.09 | 34.69 | 1714 |
| 4.3 | 712.36 | 34.46 | 317.31 | 38.18 | 1718 |
| 4.6 | 718.55 | 33.53 | 315.65 | 38.42 | 1718 |
| 5.2 | 802.06 | 33.60 | 315.81 | 41.06 | 1718 |
| 5.5 | 811.09 | 32.01 | 316.72 | 41.33 | 1718 |

Table 6.2.14.a. — SA/SIPT Scheduling (part a) – Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor x $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.3 | 308.38 | 255.07 | 0.79 | 0.86 |
| 1.6 | 297.31 | 231.29 | 0.89 | 0.86 |
| 2.2 | 278.94 | 235.56 | 1.68 | 1.02 |
| 2.8 | 278.49 | 250.42 | 1.75 | 1.05 |
| 3.4 | 278.12 | 236.80 | 1.92 | 1.02 |
| 3.7 | 276.96 | 221.18 | 2.20 | 1.07 |
| 4.3 | 276.57 | 225.26 | 2.57 | 1.07 |
| 4.6 | 276.54 | 222.53 | 2.59 | 1.06 |
| 5.2 | 276.44 | 219.78 | 2.89 | 1.07 |
| 5.5 | 276.32 | 228.12 | 2.93 | 1.09 |

Table 6.2.14.b. — SA/SIPT Scheduling (part b) – Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor $\times m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.3 | 250.18 | 21.19 | 166.76 | 17.84 | 1640 |
| 2.2 | 480.25 | 27.25 | 238.04 | 29.41 | 1716 |
| 2.8 | 532.19 | 27.09 | 235.78 | 31.62 | 1718 |
| 3.7 | 700.27 | 24.34 | 260.55 | 37.89 | 1719 |
| 4.3 | 746.50 | 37.79 | 311.90 | 39.34 | 1720 |
| 5.5 | 868.86 | 33.28 | 296.87 | 43.04 | 1719 |

Table 6.2.15.a. — SA/SA Scheduling (part a) - Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor $\times m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.3 | 293.46 | 121.34 | 0.85 | 0.86 |
| 2.2 | 276.61 | 137.61 | 1.73 | 0.95 |
| 2.8 | 276.40 | 130.88 | 1.92 | 0.98 |
| 3.7 | 276.29 | 158.31 | 2.53 | 1.07 |
| 4.3 | 276.25 | 149.35 | 2.70 | 0.99 |
| 5.5 | 276.18 | 188.55 | 3.15 | 1.07 |

Table 6.2.15.b. — SA/SA Scheduling (part b) - Small-Fab

Note: $m$ = 2 is the number of machines at the bottleneck.

### 6.2.1 — Interpretation of Simulation Experiment - Small-Fab

To interpret the results of our simulation experiment on Small-Fab let us consider first each release strategy separately and observe the effect of varying the dispatching rule within each release policy. Secondly, we consider a fixed dispatching rule with different release strategies.

For UNIF release, SRPT dispatching appears to outperform both SIPT and

FIFO dispatching (Fig. 6.2.2).

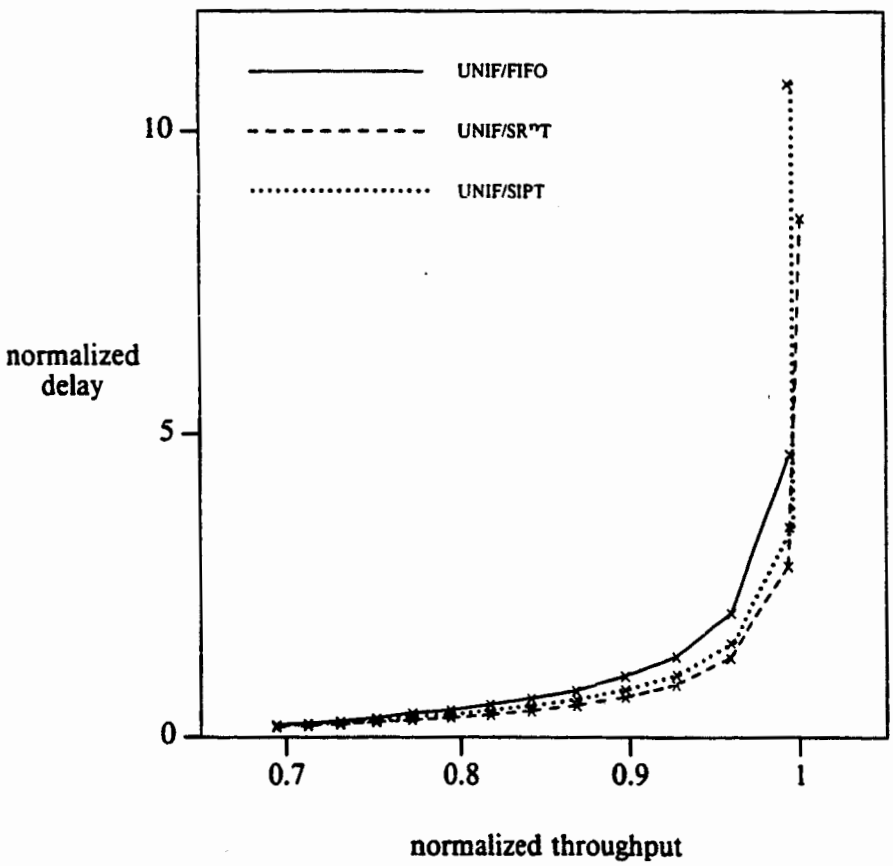


Fig. 6.2.2 — UNIF Release – Small-Fab

For all values of throughput rate SRPT produced schedules with less mean delay than both FIFO and SIPT, achieving reductions of approximately 40% and 19% (when compared to FIFO and SIPT, respectively) in mean delay at the throughput rate of about 1 lot per 280 minutes. At that throughput rate SIPT produced about 25% less delay than FIFO. With respect to the coefficient of variation of inter-departure times (CVID) SRPT and FIFO were about equal, while both produced schedules with less CVID than SIPT (see Tables 6.2.3.b, 6.2.4.b, and 6.2.5.b). CVID is a measure of the regularity of output.

Fixed-WIP release with SRPT, SIPT, and FIFO dispatching produced schedules with no noticeable mean delay differences (Fig. 6.2.3).

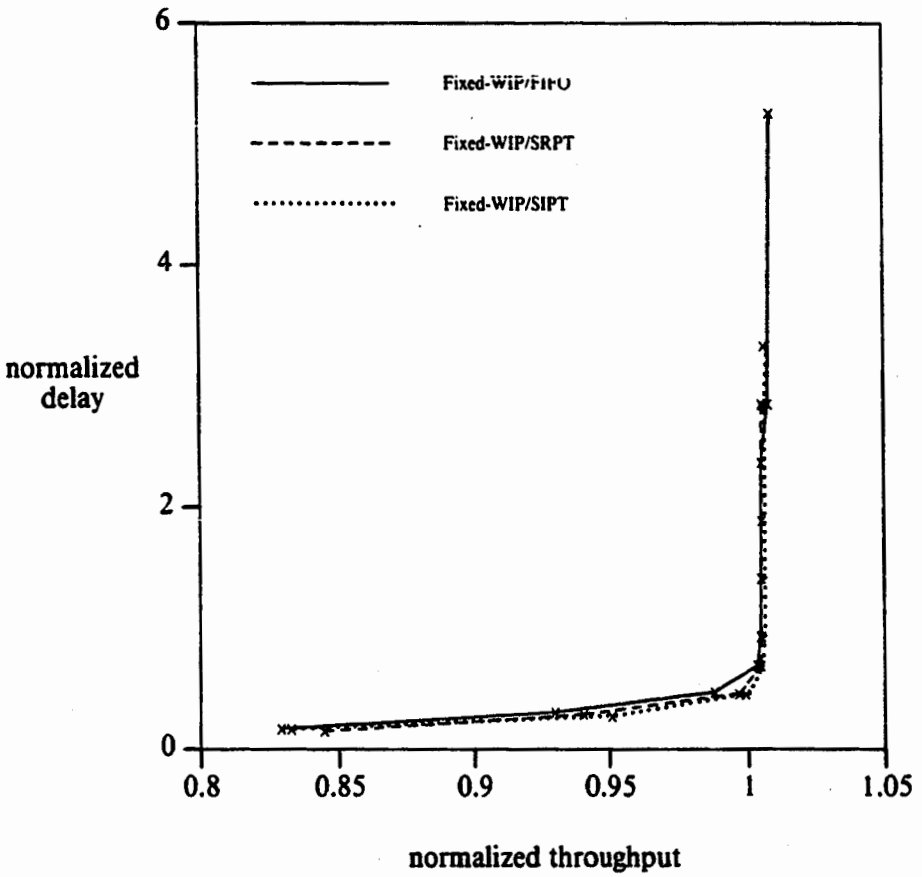


Fig. 6.2.3 — Fixed-WIP Release – Small-Fab

SRPT outperformed FIFO and SIPT dispatching in the sense that it produced schedules with less CVID (see Tables 6.2.6.b, 6.2.7.b, and 6.2.8.b).

With WR release, the relative dispatching rule performances were indifferent for the greater portion of the range of throughput rate values (Fig. 6.2.4).

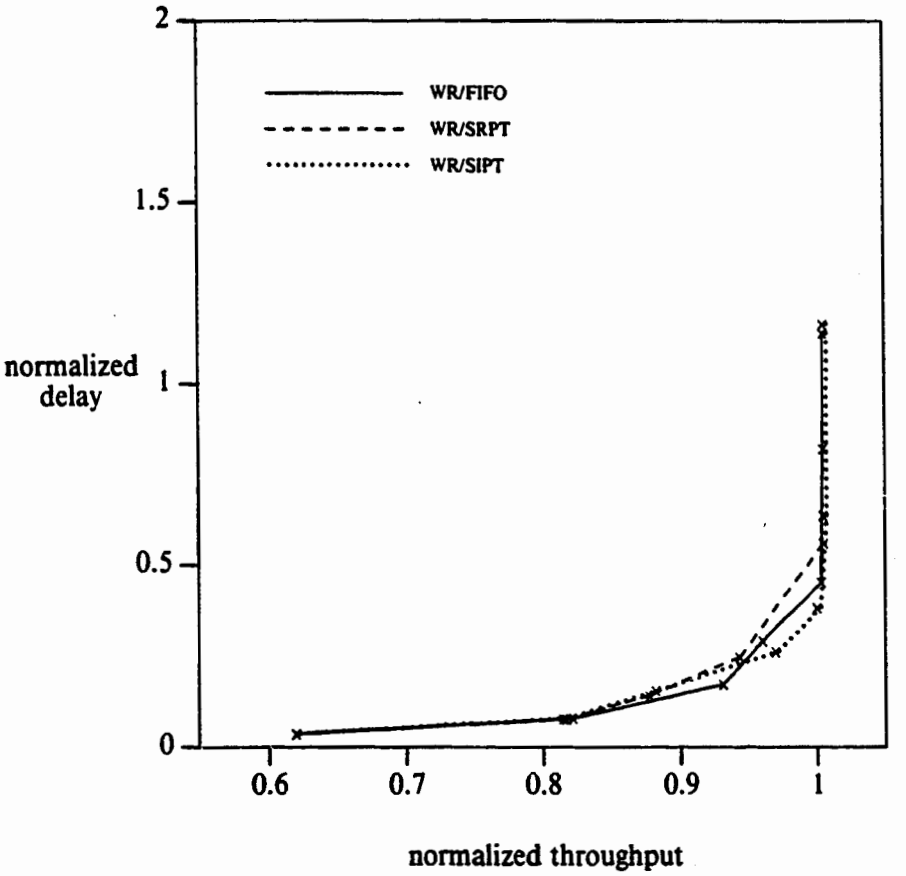


Fig. 6.2.4 — WR Release - Small-Fab

At the high end of the range, however, SRPT showed the least sensitivity to increases in throughput rate, while FIFO showed the highest, followed closely by SIPT. With respect to CVID (see Tables 6.2.9.b, 6.2.10.b, and 6.2.11.b) SIPT had the worst performance while SRPT and FIFO were better (SRPT was the most consistent in terms of inter-departure time variability).

With SA release, FIFO dispatching was clearly the worst rule of the four considered (Fig. 6.2.5).

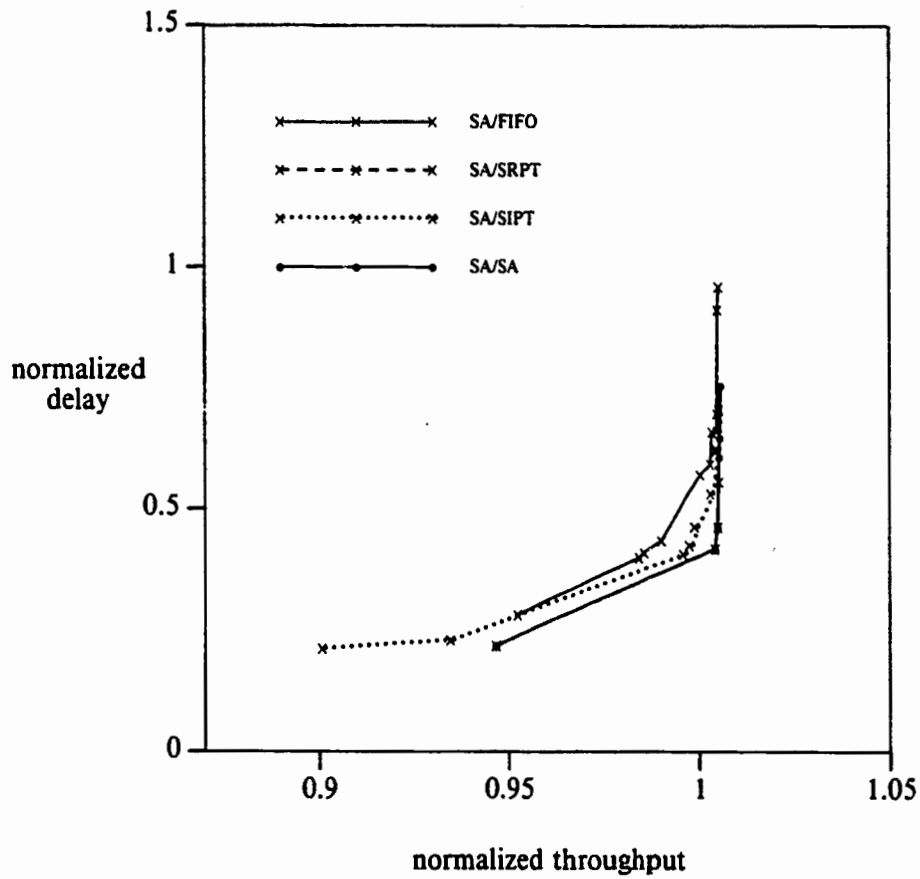


Fig. 6.2.5 — SA Release - Small-Fab

SA and SRPT dispatching were almost identical with respect to mean delay (note that their plots overlap in Fig. 6.2.5), but SRPT was slightly superior to SA with respect to CVID in the high end of the throughput range (see Tables 6.2.12.b, 6.2.13.b, 6.2.14.b, and 6.2.15.b). SIPT dispatching was better than FIFO with respect to mean delay and for a large portion of the throughput range it was superior to both SRPT and SA dispatching. For the highest throughput values, however, it could not outperform either SA or SRPT.

We now fix the dispatching strategy and compare the different release strategies. We choose for this analysis to fix the dispatching rule to SRPT, given its overall good performance. The scheduling policies we will consider are therefore - UNIF/SRPT,

Fixed-WIP/SRPT, WR/SRPT, and SA/SRPT. Within its domain of throughput values (from 95% to 100.5% of expected capacity) SA/SRPT clearly outperformed the other three scheduling policies. The improvement was most remarkable against UNIF/SRPT (Fig 6.2.6), where mean delay reductions varied from 80% to 92%.

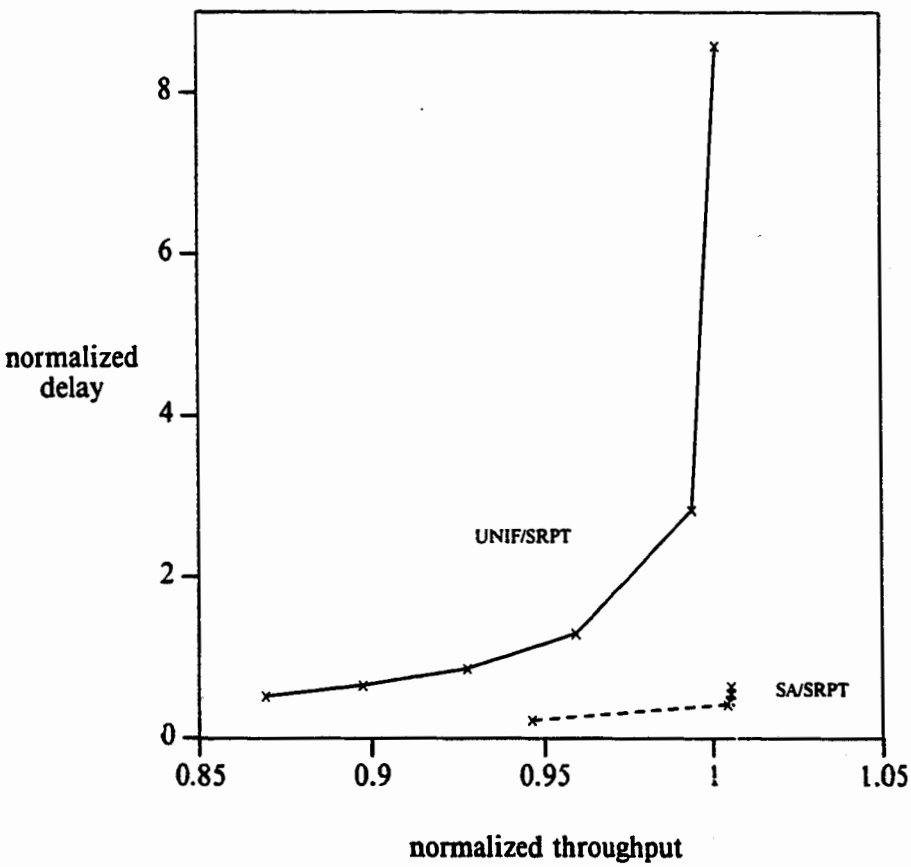


Fig. 6.2.6 – SA/SRPT vs UNIF/SRPT Release – Small-Fab

CVID values for SA/SRPT and UNIF/SRPT are similar (see Tables 6.2.4.b and 6.2.13.b). When compared to Fixed-WIP/SRPT (Fig. 6.2.7) and WR/SRPT (Fig. 6.2.8) improvements were not as marking, but SA/SRPT remained the best scheduling policy with 59% and 74% reductions in mean delay at the high end of throughput range, respectively.

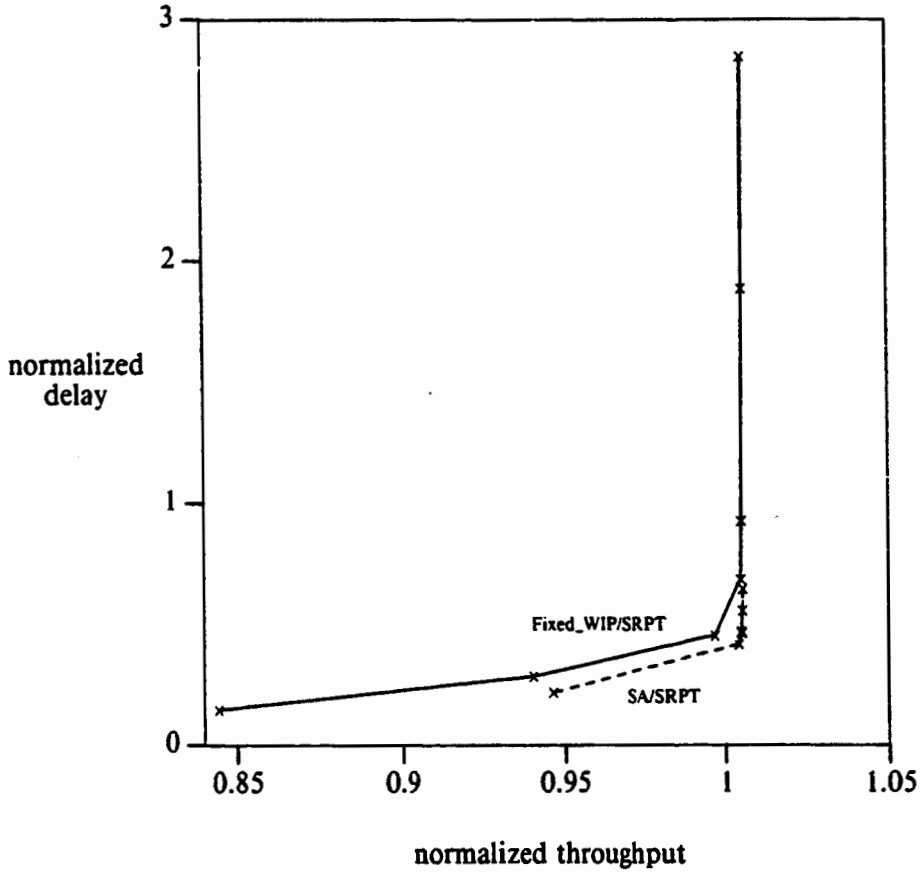


Fig. 6.2.7 — SA/SRPT vs Fixed-WIP/SRPT - Small-Fab

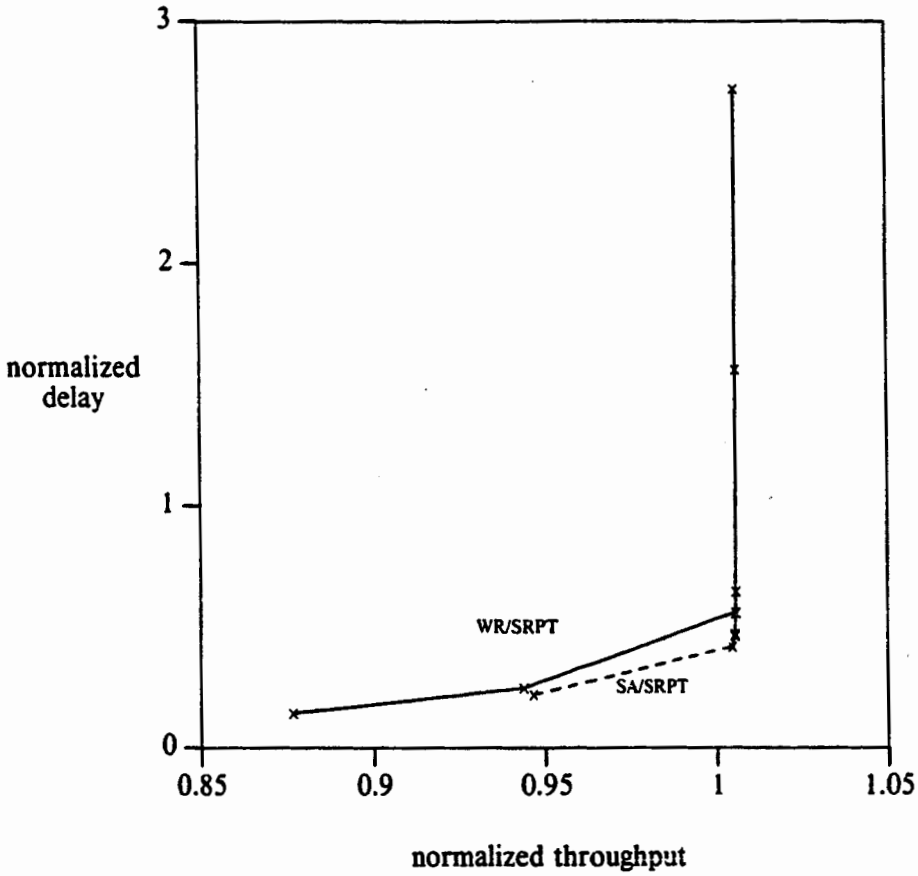


Fig. 6.2.8 — SA/SRPT vs WR/SRPT - Small-Fab

In summary, with respect to Small-Fab, varying the dispatching rule for a fixed release strategy did not show as much mean delay sensitivity as fixing the dispatching rule and varying the release strategy. SRPT was the best dispatching rule and SA the superior release strategy.

### 6.3 — A Single Bottleneck Fab Queueing Network - Dayhoff-87 [Day87a]

In this Section we compare SA scheduling with UNIF release and FIFO dispatching on a hypothetical single bottleneck wafer fab queueing network described by Dayhoff [Day87a], and whose topology is given in Fig. 6.3.1.

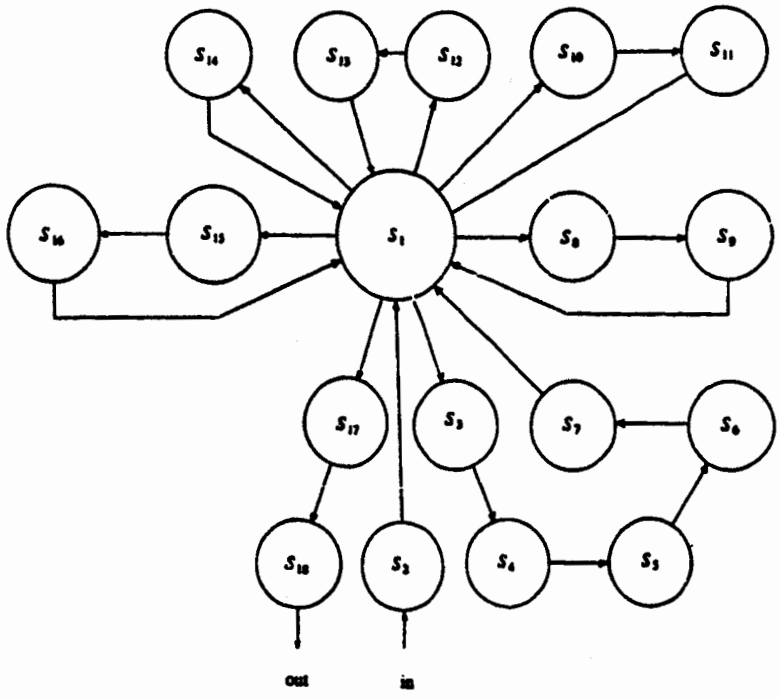


Fig. 6.3.1 — Network Topology – Dayhoff-87

This fab has 18 work stations (with one bottleneck) that are summarized in Table 6.3.1. Station $S_1$ is the bottleneck. Machine availability is not given in Dayhoff [Day87a] and is set so as to maintain station $S_1$ the bottleneck. Machine load sizes are also not provided by Dayhoff. We set load size to one for all machines.

| work station | number machines | MTTR (mins) | MTBF (mins) |
|---|---|---|---|
| 1 | 6 | 5440 | 16200 |
| 2 | 4 | 1440 | 12240 |
| 3 | 1 | 1440 | 12240 |
| 4 | 4 | 1440 | 12240 |
| 5 | 2 | 1440 | 12240 |
| 6 | 3 | 1440 | 12240 |
| 7 | 4 | 1440 | 12240 |
| 8 | 1 | 1440 | 12240 |
| 9 | 4 | 1440 | 12240 |
| 10 | 1 | 1440 | 12240 |
| 11 | 4 | 1440 | 12240 |
| 12 | 1 | 60 | 10860 |
| 13 | 4 | 1440 | 12240 |
| 14 | 1 | 40 | 10860 |
| 15 | 1 | 1440 | 12240 |
| 16 | 1 | 1440 | 12240 |
| 17 | 1 | 40 | 10840 |
| 18 | 1 | 40 | 10840 |

Table 6.3.1 — Work Stations - Dayhoff-87

Dayhoff-87 is a single product fab. The process recipe is given in Table 6.3.2. Dayhoff does not present processing times in her paper. The times given below, are again set so as to maintain $S_1$ the bottleneck. The total duration of processing is 2,600 minutes.

| step number | work station | step duration (mins) |
|---|---|---|
| 1 | 2 | 60 |
| 2 | 1 | 240 |
| 3 | 3 | 60 |
| 4 | 4 | 90 |
| 5 | 5 | 90 |
| 6 | 6 | 120 |
| 7 | 7 | 60 |
| 8 | 1 | 90 |
| 9 | 8 | 90 |
| 10 | 9 | 60 |
| 11 | 1 | 100 |
| 12 | 10 | 40 |
| 13 | 11 | 60 |
| 14 | 1 | 100 |
| 15 | 12 | 40 |
| 16 | 13 | 240 |
| 17 | 1 | 150 |
| 18 | 14 | 40 |
| 19 | 1 | 360 |
| 20 | 15 | 100 |
| 21 | 16 | 40 |
| 22 | 1 | 240 |
| 23 | 17 | 40 |
| 24 | 18 | 90 |

Table 6.3.2 — Process Recipe - Dayhoff-87

No hot lots are allowed to be started in this fab. Rework is assumed not to be required. The nominal capacity of this fab network is one lot every 213.33 minutes (47.25 lots/week). Its expected capacity is one lot every 320.79 minutes (31.42 lots/week). All simulations are run with $H$ = 1.5 million minutes (about 34 months) and each requires approximately 200 seconds of CPU time on the Cray X-MP/14. Tables 6.3.3.a and 6.3.3.b summarize runs with UNIF/FIFO scheduling, while Tables 6.3.4.a and Table 6.3.4.b show results for SA/SA scheduling.

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 500 | 1963.43 | 312.87 | 2332.84 | 43.03 | 2851 |
| 460 | 2329.18 | 365.28 | 2546.93 | 47.33 | 3097 |
| 420 | 3000.87 | 450.19 | 2981.88 | 53.84 | 3393 |
| 400 | 3732.59 | 601.24 | 3550.97 | 59.03 | 3563 |
| 390 | 4223.85 | 691.83 | 3996.93 | 61.91 | 3654 |
| 380 | 4727.87 | 781.72 | 4444.82 | 64.52 | 3750 |
| 360 | 6338.98 | 961.39 | 5478.52 | 70.94 | 3958 |
| 350 | 7550.22 | 1081.41 | 6149.94 | 74.46 | 4071 |
| 340 | 9404.08 | 1173.11 | 6680.99 | 78.47 | 4191 |
| 335 | 11077.32 | 1245.40 | 7119.31 | 81.00 | 4254 |
| 300 | 13889.32 | 1371.37 | 7823.17 | 84.25 | 4318 |
| 325 | 18328.45 | 1591.15 | 9087.56 | 87.54 | 4385 |
| 320 | 24878.26 | 2095.37 | 11997.52 | 90.54 | 4453 |

Table 6.3.3.a. — UNIF/FIFO Scheduling (part a) - Dayhoff-87

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 500 | 501.34 | 917.37 | 1.40 | 3.49 |
| 460 | 461.25 | 877.50 | 2.45 | 4.47 |
| 420 | 421.42 | 819.42 | 4.26 | 6.36 |
| 400 | 401.57 | 775.98 | 6.02 | 8.27 |
| 390 | 391.40 | 780.69 | 7.38 | 9.78 |
| 380 | 381.46 | 764.75 | 8.71 | 11.05 |
| 360 | 362.15 | 729.62 | 13.44 | 14.69 |
| 350 | 352.48 | 712.92 | 17.32 | 17.12 |
| 340 | 343.00 | 664.30 | 23.19 | 19.16 |
| 335 | 338.60 | 651.05 | 28.29 | 20.73 |
| 330 | 333.53 | 639.49 | 37.01 | 23.22 |
| 325 | 330.19 | 627.37 | 50.96 | 27.57 |
| 320 | 328.45 | 641.12 | 70.55 | 36.13 |

Table 6.3.3.b. — UNIF/FIFO Scheduling (part b) - Dayhoff-87

| safety factor x $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.0 | 4674.58 | 835.14 | 5293.20 | 64.23 | 3890 |
| 1.9 | 5373.51 | 909.33 | 5739.34 | 67.17 | 4101 |
| 2.2 | 5580.97 | 912.70 | 5898.61 | 68.31 | 4145 |
| 2.5 | 5776.96 | 965.42 | 6290.10 | 69.03 | 4164 |
| 3.1 | 5943.97 | 987.92 | 6281.35 | 69.66 | 4165 |
| 3.4 | 6031.20 | 958.11 | 6226.32 | 69.92 | 4198 |
| 4.3 | 6078.34 | 972.44 | 6308.71 | 70.07 | 4255 |
| 4.9 | 6150.24 | 995.81 | 6520.29 | 70.29 | 4263 |
| 5.5 | 6273.13 | 1015.02 | 6665.39 | 70.80 | 4279 |
| 5.8 | 6422.19 | 1019.08 | 6663.96 | 71.25 | 4289 |
| 6.1 | 6440.08 | 948.31 | 6300.48 | 71.25 | 4296 |
| 6.4 | 6703.86 | 980.86 | 6413.36 | 72.11 | 4314 |

Table 6.3.4.a. — SA/SA Scheduling (part a) – Dayhoff-87

Note: $m$ = 6 is the number of machines at the bottleneck.

| safety factor x $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.0 | 367.52 | 774.44 | 5.47 | 10.46 |
| 1.9 | 348.76 | 742.03 | 7.61 | 12.93 |
| 2.2 | 344.83 | 732.63 | 7.76 | 12.74 |
| 2.5 | 343.18 | 720.48 | 8.09 | 13.82 |
| 3.1 | 343.08 | 719.93 | 8.56 | 14.25 |
| 3.4 | 340.43 | 726.75 | 8.92 | 14.41 |
| 4.3 | 336.03 | 696.88 | 9.38 | 14.78 |
| 4.9 | 335.32 | 690.14 | 9.69 | 15.32 |
| 5.5 | 334.38 | 699.15 | 9.76 | 15.44 |
| 5.8 | 333.42 | 699.40 | 10.37 | 15.63 |
| 6.1 | 332.42 | 693.81 | 10.30 | 14.93 |
| 6.4 | 331.18 | 698.38 | 10.87 | 15.72 |

Table 6.3.4.b. — SA/SA Scheduling (part b) – Dayhoff-87

Note: $m$ = 6 is the number of machines at the bottleneck.

### 6.3.1 — Interpretation of Simulation Experiment - Dayhoff-87

SA release coupled with the SA dispatching rule outperformed UNIF/FIFO with respect to mean delay (Fig. 6.3.2) producing schedules that had from 25% to 55% less mean delay.

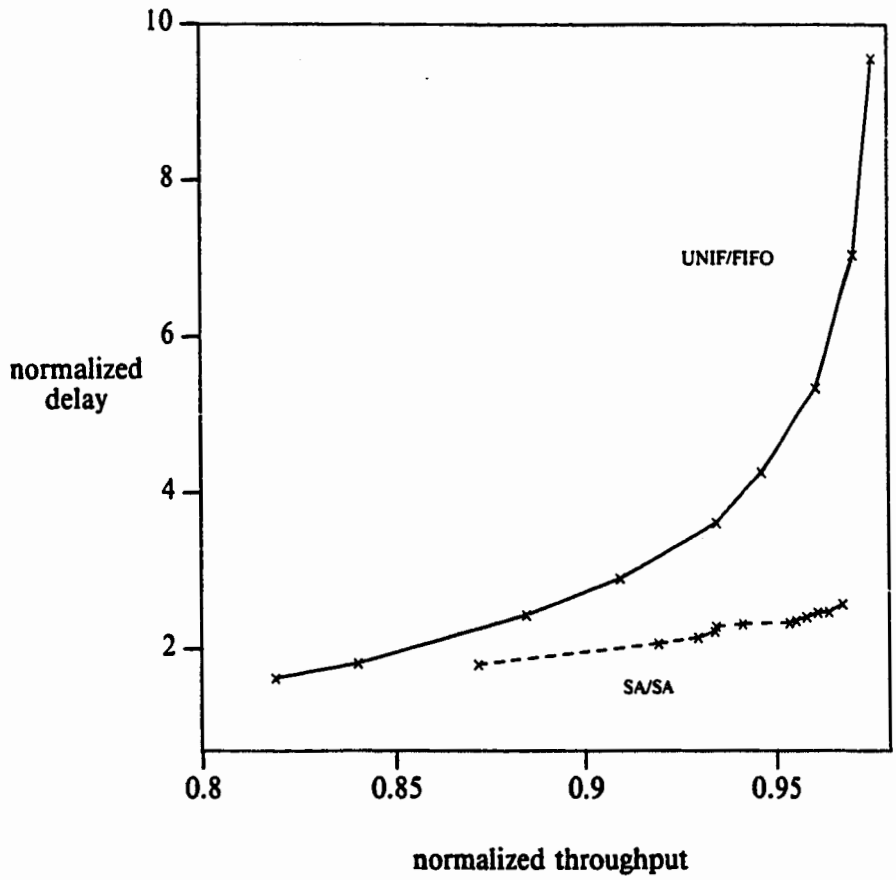


Fig. 6.3.2 — SA/SA vs UNIF/FIFO - Dayhoff-87

UNIF/FIFO did slightly better with respect to CVID (see Tables 6.3.1.b and 6.3.2.b).

### 6.4 — A Bipolar Process Fab Queueing Network - Bipolar [Bar86a]

We next compare SA scheduling with UNIF release and FIFO dispatching on a

hypothetical wafer fab queueing network suggested by Barry [Bar86a]. Its topology is given in Fig. 6.4.1. Photolithography, station $S_4$, is the bottleneck. In this example MTTR of the bottleneck machines is set to approximately 11.5 times the average step duration at the bottleneck. Bottleneck machine availability, however, is maintained relatively high, *i.e.* 82.4%.

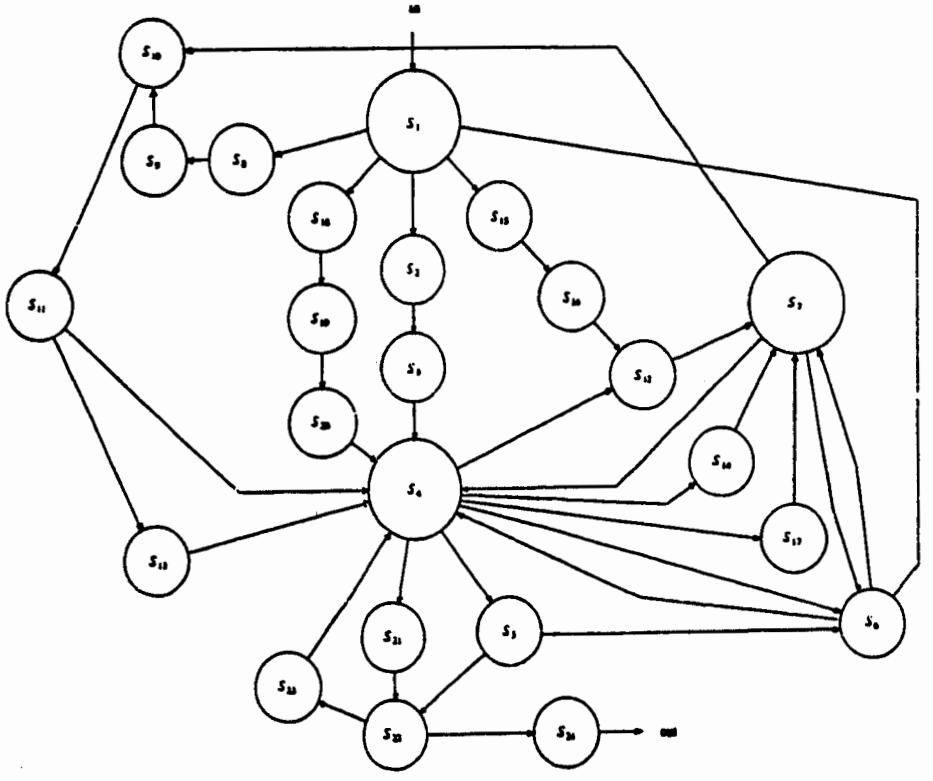

Fig. 6.4.1 — Network Topology – Bipolar

Table 6.4.1 summarizes the work stations of this fab. Load size is assumed to be a single lot for every machine.

| work station | number machines | MTTR (mins) | MTBF (mins) |
|---|---|---|---|
| 1 | 10 | 60 | 480 |
| 2 | 1 | 90 | 1200 |
| 3 | 2 | 90 | 1200 |
| 4 | 2 | 4000 | 22750 |
| 5 | 4 | 240 | 2400 |
| 6 | 8 | 360 | 1440 |
| 7 | 12 | 60 | 1440 |
| 8 | 10 | 120 | 3600 |
| 9 | 1 | 60 | 1440 |
| 10 | 10 | 90 | 1200 |
| 11 | 14 | 120 | 3600 |
| 12 | 4 | 240 | 2400 |
| 13 | 1 | 240 | 2400 |
| 14 | 4 | 240 | 2400 |
| 15 | 1 | 120 | 7200 |
| 16 | 2 | 90 | 1200 |
| 17 | 3 | 240 | 2400 |
| 18 | 1 | 120 | 7200 |
| 19 | 1 | 60 | 1440 |
| 20 | 2 | 240 | 2400 |
| 21 | 3 | 240 | 2400 |
| 22 | 4 | 60 | 1440 |
| 23 | 2 | 90 | 1200 |
| 24 | 4 | 120 | 1440 |

Table 6.4.1 — Work Stations - Bipolar

This fab processes a single product. Its total duration of processing is 8,100 minutes. The process recipe is given in Table 6.4.2.

| step number | work station | step duration (mins) | rework probability |
|---|---|---|---|
| 1 | 1 | 300 | 0.00 |
| 2 | 2 | 60 | 0.00 |
| 3 | 3 | 120 | 0.00 |
| 4 | 4 | 285 | 0.00 |
| 5 | 5 | 180 | 0.00 |
| 6 | 6 | 30 | 0.00 |
| 7 | 7 | 120 | 0.00 |
| 8 | 1 | 135 | 0.00 |
| 9 | 8 | 540 | 0.00 |
| 10 | 9 | 60 | 0.00 |
| 11 | 10 | 375 | 0.00 |
| 12 | 11 | 60 | 0.00 |
| 13 | 4 | 345 | 0.00 |
| 14 | 12 | 105 | 0.00 |
| 15 | 7 | 180 | 0.00 |
| 16 | 6 | 15 | 0.00 |
| 17 | 4 | 225 | 0.00 |
| 18 | 6 | 15 | 0.00 |
| 19 | 7 | 120 | 0.00 |
| 20 | 10 | 375 | 0.00 |
| 21 | 11 | 60 | 0.00 |
| 22 | 13 | 60 | 0.00 |
| 23 | 4 | 435 | 0.03 |
| 24 | 14 | 210 | 0.00 |
| 25 | 7 | 195 | 0.00 |
| 26 | 6 | 120 | 0.00 |
| 27 | 1 | 135 | 0.00 |
| 28 | 15 | 60 | 0.00 |
| 29 | 16 | 195 | 0.00 |
| 30 | 12 | 105 | 0.00 |
| 31 | 7 | 120 | 0.00 |
| 32 | 4 | 435 | 0.03 |
| 33 | 17 | 120 | 0.00 |
| 34 | 7 | 195 | 0.00 |
| 35 | 1 | 135 | 0.00 |
| 36 | 18 | 60 | 0.00 |
| 37 | 19 | 30 | 0.00 |
| 38 | 20 | 60 | 0.00 |
| 39 | 4 | 435 | 0.03 |
| 40 | 21 | 120 | 0.00 |

Table 6.4.2 — Process Recipe - Bipolar (continues)

| step number | work station | step duration (mins) | rework probability |
|---|---|---|---|
| 41 | 22 | 195 | 0.00 |
| 42 | 23 | 180 | 0.00 |
| 43 | 4 | 255 | 0.00 |
| 44 | 5 | 120 | 0.00 |
| 45 | 22 | 180 | 0.00 |
| 46 | 24 | 240 | 0.00 |

Table 6.4.2 — Process Recipe – Bipolar (continued)

Hot lots are not allowed to be started. The nominal capacity of the fab network is one lot every 1207.05 minutes (8.35 lots/week). Its expected capacity is one lot every 1465.10 minutes (6.88 lots/week). Simulations are run with a horizon $H$ = 4 million minutes (about 91 months), each requiring approximately one half hour of CPU time on the Cray X-MP/14.

Tables 6.4.3.a and 6.4.3.b summarize runs with UNIF/FIFO scheduling, while Tables 6.4.4.a and Table 6.4.4.b show results for SA/SA scheduling.

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 2000 | 1953.09 | 439.28 | 2599.02 | 19.45 | 1901 |
| 1940 | 2282.73 | 419.29 | 2863.91 | 21.97 | 1958 |
| 1880 | 2829.84 | 637.96 | 3490.65 | 25.89 | 2021 |
| 1820 | 3373.47 | 769.23 | 4033.26 | 29.47 | 2088 |
| 1760 | 3855.19 | 772.24 | 4578.91 | 32.27 | 2159 |
| 1700 | 4557.42 | 1050.90 | 5318.26 | 36.10 | 2235 |
| 1675 | 5003.10 | 1094.15 | 5727.34 | 38.26 | 2269 |
| 1655 | 5321.36 | 1219.39 | 6020.99 | 39.86 | 2296 |
| 1625 | 5796.61 | 1219.39 | 6519.97 | 41.88 | 2338 |
| 1595 | 6700.73 | 1443.41 | 7132.93 | 45.56 | 2382 |
| 1565 | 7563.06 | 1612.00 | 7501.18 | 48.53 | 2428 |
| 1535 | 10079.43 | 1734.84 | 8282.74 | 55.74 | 2475 |
| 1495 | 18757.37 | 2308.08 | 10481.37 | 70.33 | 2542 |

Table 6.4.3.a. — UNIF/FIFO Scheduling (part a) - Bipolar

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 2000 | 2000.50 | 890.55 | 0.93 | 1.55 |
| 1940 | 1939.98 | 876.56 | 1.13 | 1.73 |
| 1880 | 1880.34 | 972.30 | 1.44 | 2.10 |
| 1820 | 1819.65 | 1031.05 | 1.79 | 2.47 |
| 1760 | 1760.03 | 964.00 | 2.12 | 2.82 |
| 1700 | 1699.70 | 946.22 | 2.60 | 3.33 |
| 1675 | 1674.95 | 988.71 | 2.91 | 3.61 |
| 1655 | 1655.03 | 996.25 | 3.14 | 3.83 |
| 1625 | 1625.04 | 966.49 | 3.49 | 4.17 |
| 1595 | 1595.47 | 1025.47 | NA | NA |
| 1565 | 1565.48 | 986.79 | 4.73 | 4.93 |
| 1535 | 1535.18 | 998.67 | 6.44 | 5.53 |
| 1495 | 1498.75 | 982.78 | 12.31 | 7.14 |

Table 6.4.3.b. — UNIF/FIFO Scheduling (part b) - Bipolar

| safety factor × $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.0 | 4267.15 | 165.99 | 2589.64 | 34.64 | 2499 |
| 1.3 | 5138.86 | 205.73 | 2868.91 | 38.89 | 2531 |
| 1.6 | 6126.85 | 248.31 | 3187.83 | 43.21 | 2549 |
| 1.9 | 7028.25 | 247.29 | 3262.89 | 46.52 | 2554 |
| 2.2 | 8228.79 | 260.96 | 3648.70 | 50.48 | 2558 |
| 2.5 | 9068.39 | 267.93 | 3649.97 | 52.87 | 2558 |
| 2.8 | 10310.02 | 283.63 | 3924.13 | 56.05 | 2558 |
| 3.4 | 12395.65 | 356.32 | 4110.73 | 60.55 | 2556 |
| 4.0 | 14448.73 | 391.87 | 4234.76 | 64.15 | 2558 |

Table 6.4.4.a. — SA/SA Scheduling (part a) – Bipolar

Note: $m$ = 2 is the number of machines at the bottleneck.

| safety factor × $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.0 | 1520.41 | 1216.89 | 2.75 | 1.86 |
| 1.3 | 1498.84 | 1239.66 | 3.35 | 1.94 |
| 1.6 | 1488.94 | 1285.96 | 4.05 | 2.09 |
| 1.9 | 1487.69 | 1343.12 | 4.65 | 2.12 |
| 2.2 | 1486.12 | 1420.12 | 5.46 | 2.34 |
| 2.5 | 1485.41 | 1491.53 | 6.03 | 2.24 |
| 2.8 | 1485.32 | 1522.85 | 6.86 | 2.43 |
| 3.4 | 1485.13 | 1637.79 | 8.26 | 2.44 |
| 4.0 | 1483.46 | 1782.85 | 9.64 | 2.41 |

Table 6.4.4.b. — SA/SA Scheduling (part b) – Bipolar

Note: $m$ = 2 is the number of machines at the bottleneck.

### 6.4.1 — Interpretation of Simulation Experiment - Bipolar

Once again SA/SA scheduling clearly outperformes UNIF/FIFO scheduling with regards to mean delay (Fig. 6.4.2) for the throughput range going from 96.4 to 98.8% of expected capacity.

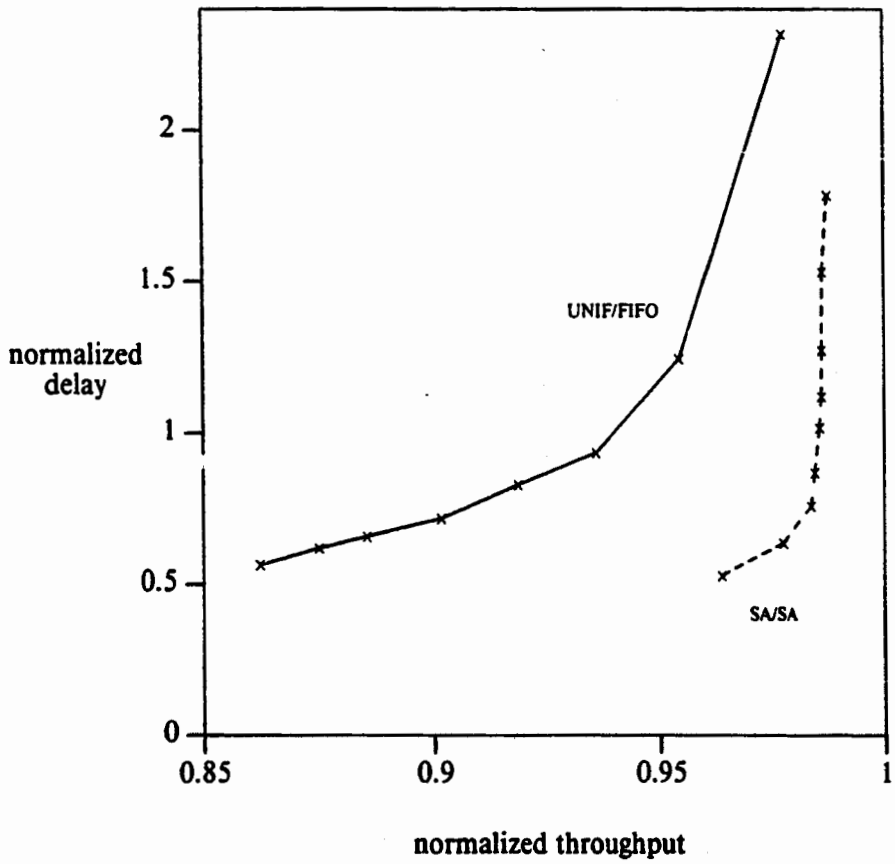


Fig. 6.4.2 — SA/SA vs UNIF/FIFO - Bipolar

In that range mean delay reductions of 50-73% were registered. UNIF/FIFO did produce schedules with less CVID (see Tables 6.4.3.b and 6.4.4.b).

### 6.5 — A Production Fab Queueing Network - SSi-Implant

We now compare SA scheduling with UNIF release and SRPT dispatching on a production wafer fab queueing network adapted from data gathered from a computer integrated manufacturing system at a Silicon Systems production fab. The topology of this network is given in Fig. 6.5.1. This fab has one bottleneck station, $S_{17}$ (ion implantation).

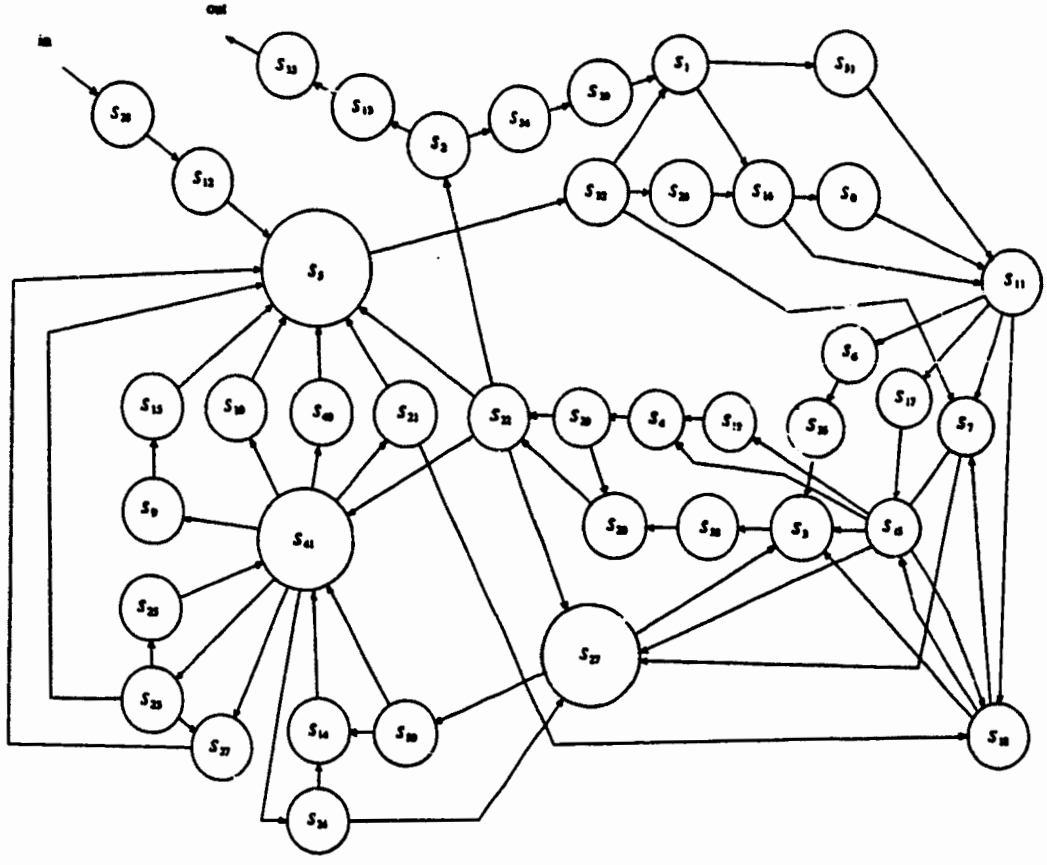

Fig. 6.5.1 — Network Topology - SSi-Implant

The network has 41 work stations and one process with a recipe length of 182 steps. Table 6.5.1 summarizes the work stations of this fab.

| work station | number machines | load size | MTTR (mins) | MTBF (mins) |
|---|---|---|---|---|
| 1 | 5 | 1 | 500.47 | 3022.27 |
| 2 | 1 | 4 | 460.50 | 4962.00 |
| 3 | 2 | 4 | 281.50 | 16637.50 |
| 4 | 2 | 4 | 281.50 | 16637.50 |
| 5 | 1 | 9 | 72.00 | 66852.00 |
| 6 | 1 | 9 | 42.00 | 111457.50 |
| 7 | 1 | 9 | 172.80 | 71592.30 |
| 8 | 1 | 1 | 15284.00 | 78228.00 |
| 9 | 2 | 4 | 850.15 | 6044.65 |
| 10 | 2 | 2 | 942.00 | 5668.50 |
| 11 | 3 | 1 | 0.00 | 113806.00 |
| 12 | 2 | 1 | 1792.36 | 32597.86 |
| 13 | 2 | 1 | 378.46 | 28305.46 |
| 14 | 1 | 4 | 1434.00 | 65833.50 |
| 15 | 1 | 4 | 913.00 | 71375.00 |
| 16 | 2 | 2 | 453.00 | 19522.50 |
| 17 | 1 | 2 | 10664.57 | 73076.57 |
| 18 | 2 | 4 | 4426.00 | 175978.00 |
| 19 | 1 | 2 | 1392.00 | 101055.00 |
| 20 | 1 | 1 | 13357.09 | 51441.09 |
| 21 | 2 | 4 | 715.38 | 4922.88 |
| 22 | 3 | 1 | 6748.50 | 114954.50 |
| 23 | 2 | 4 | 954.00 | 6427.50 |
| 24 | 2 | 4 | 1006.50 | 5320.50 |
| 25 | 1 | 2 | 372.00 | 2275.50 |
| 26 | 3 | 1 | 500.47 | 3022.27 |
| 27 | 1 | 1 | 264.75 | 1256.25 |
| 28 | 1 | 1 | 517.64 | 8661.14 |
| 29 | 1 | 2 | 23596.00 | 87832.00 |
| 30 | 4 | 1 | 824.77 | 7306.27 |
| 31 | 6 | 1 | 184.80 | 9867.30 |
| 32 | 5 | 1 | 698.40 | 8046.40 |
| 33 | 1 | 1 | 6748.50 | 114954.50 |
| 34 | 1 | 4 | 1512.00 | 5853.00 |
| 35 | 2 | 1 | 295.38 | 3193.38 |
| 36 | 1 | 1 | 1321.85 | 6502.85 |
| 37 | 2 | 4 | 1073.54 | 5078.54 |
| 38 | 2 | 4 | 114.00 | 35604.00 |
| 39 | 2 | 4 | 138.00 | 28021.50 |
| 40 | 3 | 1 | 2722.50 | 9895.50 |
| 41 | 1 | 6 | 167.54 | 1787.54 |

Table 6.5.1 — Work Stations – SSi-Implant

Machines in stations $S_{11}$ are assumed to never fail. The single process recipe of

the fab is given in Table 6.5.2. The total duration of processing for this recipe is 11,695 minutes. Rework is assumed not to be required.

| step number | work station | step duration (mins) |
|---|---|---|
| 1 | 28 | 45 |
| 2 | 12 | 20 |
| 3 | 5 | 90 |
| 4 | 32 | 90 |
| 5 | 26 | 45 |
| 6 | 16 | 50 |
| 7 | 8 | 30 |
| 8 | 11 | 50 |
| 9 | 7 | 60 |
| 10 | 35 | 50 |
| 11 | 3 | 50 |
| 12 | 38 | 35 |
| 13 | 20 | 30 |
| 14 | 22 | 15 |
| 15 | 5 | 90 |
| 16 | 32 | 90 |
| 17 | 1 | 45 |
| 18 | 16 | 35 |
| 19 | 11 | 35 |
| 20 | 7 | 60 |
| 21 | 27 | 30 |
| 22 | 3 | 50 |
| 23 | 38 | 30 |
| 24 | 22 | 15 |
| 25 | 41 | 90 |
| 26 | 9 | 600 |
| 27 | 15 | 30 |
| 28 | 5 | 90 |
| 29 | 32 | 90 |
| 30 | 1 | 45 |
| 31 | 16 | 35 |
| 32 | 11 | 30 |
| 33 | 7 | 60 |
| 34 | 27 | 30 |
| 35 | 3 | 50 |
| 36 | 38 | 30 |
| 37 | 22 | 15 |
| 38 | 27 | 30 |
| 39 | 39 | 45 |
| 40 | 41 | 90 |

Table 6.5.2 — Process Recipe – SSi-Implant (continues)

| step number | work station | step duration (mins) |
|---|---|---|
| 41 | 21 | 600 |
| 42 | 5 | 90 |
| 43 | 32 | 90 |
| 44 | 1 | 30 |
| 45 | 16 | 35 |
| 46 | 11 | 30 |
| 47 | 18 | 45 |
| 48 | 7 | 60 |
| 49 | 35 | 45 |
| 50 | 27 | 90 |
| 51 | 3 | 50 |
| 52 | 38 | 30 |
| 53 | 22 | 15 |
| 54 | 41 | 90 |
| 55 | 21 | 300 |
| 56 | 18 | 30 |
| 57 | 35 | 90 |
| 58 | 39 | 45 |
| 59 | 14 | 30 |
| 60 | 41 | 90 |
| 61 | 24 | 120 |
| 62 | 14 | 30 |
| 63 | 41 | 90 |
| 64 | 24 | 180 |
| 65 | 27 | 30 |
| 66 | 39 | 45 |
| 67 | 41 | 75 |
| 68 | 37 | 270 |
| 69 | 27 | 90 |
| 70 | 39 | 45 |
| 71 | 5 | 90 |
| 72 | 32 | 90 |
| 73 | 26 | 50 |
| 74 | 16 | 35 |
| 75 | 8 | 20 |
| 76 | 11 | 30 |
| 77 | 7 | 60 |
| 78 | 35 | 60 |
| 79 | 3 | 50 |
| 80 | 38 | 30 |

Table 6.5.2 — Process Recipe - SSi-Implant (continued)

| step number | work station | step duration (mins) |
|---|---|---|
| 81 | 20 | 30 |
| 82 | 22 | 15 |
| 83 | 5 | 90 |
| 84 | 32 | 90 |
| 85 | 1 | 30 |
| 86 | 16 | 35 |
| 87 | 11 | 30 |
| 88 | 7 | 60 |
| 89 | 27 | 30 |
| 90 | 3 | 50 |
| 91 | 38 | 30 |
| 92 | 22 | 15 |
| 93 | 39 | 45 |
| 94 | 41 | 75 |
| 95 | 23 | 300 |
| 96 | 37 | 180 |
| 97 | 5 | 90 |
| 98 | 32 | 90 |
| 99 | 7 | 60 |
| 100 | 35 | 60 |
| 101 | 18 | 60 |
| 102 | 3 | 50 |
| 103 | 38 | 30 |
| 104 | 22 | 15 |
| 105 | 41 | 90 |
| 106 | 10 | 90 |
| 107 | 5 | 90 |
| 108 | 32 | 90 |
| 109 | 1 | 30 |
| 110 | 16 | 35 |
| 111 | 11 | 30 |
| 112 | 7 | 60 |
| 113 | 35 | 60 |
| 114 | 3 | 50 |
| 115 | 38 | 30 |
| 116 | 22 | 15 |
| 117 | 5 | 90 |
| 118 | 32 | 90 |
| 119 | 1 | 30 |
| 120 | 16 | 35 |

Table 6.5.2 — Process Recipe - SSi-Implant (continued)

| step number | work station | step duration (mins) |
|---|---|---|
| 121 | 11 | 30 |
| 122 | 18 | 30 |
| 123 | 7 | 60 |
| 124 | 27 | 90 |
| 125 | 3 | 50 |
| 126 | 38 | 30 |
| 127 | 22 | 15 |
| 128 | 5 | 90 |
| 129 | 32 | 90 |
| 130 | 1 | 30 |
| 131 | 16 | 35 |
| 132 | 11 | 30 |
| 133 | 7 | 60 |
| 134 | 27 | 90 |
| 135 | 3 | 50 |
| 136 | 38 | 30 |
| 137 | 22 | 15 |
| 138 | 41 | 75 |
| 139 | 23 | 120 |
| 140 | 25 | 90 |
| 141 | 41 | 75 |
| 142 | 23 | 180 |
| 143 | 5 | 90 |
| 144 | 32 | 90 |
| 145 | 26 | 50 |
| 146 | 16 | 35 |
| 147 | 8 | 20 |
| 148 | 11 | 30 |
| 149 | 6 | 60 |
| 150 | 36 | 45 |
| 151 | 3 | 50 |
| 152 | 38 | 30 |
| 153 | 20 | 30 |
| 154 | 22 | 30 |
| 155 | 41 | 75 |
| 156 | 40 | 60 |
| 157 | 5 | 90 |
| 158 | 32 | 90 |
| 159 | 26 | 50 |
| 160 | 16 | 35 |

Table 6.5.2 — Process Recipe - SSi-Implant (continued)

| step number | work station | step duration (mins) |
|---|---|---|
| 161 | 8 | 20 |
| 162 | 11 | 30 |
| 163 | 17 | 90 |
| 164 | 35 | 15 |
| 165 | 4 | 50 |
| 166 | 29 | 30 |
| 167 | 20 | 15 |
| 168 | 22 | 30 |
| 169 | 2 | 90 |
| 170 | 34 | 120 |
| 171 | 30 | 120 |
| 172 | 1 | 30 |
| 173 | 31 | 15 |
| 174 | 11 | 15 |
| 175 | 35 | 45 |
| 176 | 19 | 30 |
| 177 | 4 | 50 |
| 178 | 29 | 30 |
| 179 | 22 | 30 |
| 180 | 2 | 90 |
| 181 | 13 | 60 |
| 182 | 33 | 15 |

Table 6.5.2 — Process Recipe – SSi-Implant (continued)

Hot lots are not allowed to be started. The nominal capacity of the fab network is one lot every 510.00 minutes (19.8 lots/week). Its expected capacity is one lot every 646.18 minutes (15.6 lots/week). Simulations are run with a horizon $H$ = 2.5 minutes (about 57 months), each requiring approximately one hour of CPU time on the Cray X-MP/14.

Tables 6.5.3.a and 6.5.3.b summarize runs with UNIF/SRPT scheduling, while Tables 6.5.4.a and Table 6.5.4.b show results for SA/SA scheduling.

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1000 | 34246.57 | 5395.83 | 25195.89 | 74.88 | 2376 |
| 950 | 36218.12 | 5379.75 | 24741.74 | 75.59 | 2500 |
| 850 | 41139.06 | 4897.90 | 24793.14 | 78.24 | 2794 |
| 750 | 55828.20 | 4622.96 | 24115.46 | 82.74 | 3167 |
| 700 | 67664.73 | 4219.06 | 23316.98 | 85.36 | 3393 |
| 675 | 77314.17 | 3847.52 | 21922.82 | 86.89 | 3518 |
| 650 | 98508.23 | 3165.40 | 18876.89 | 89.39 | 3654 |
| 630 | 126529.32 | 4685.71 | 26827.19 | 91.57 | 3770 |
| 615 | 154528.77 | 7672.96 | 41539.80 | 92.97 | 3862 |

Table 6.5.3.a. — UNIF/SRPT Scheduling (part a) - SSi-Implant

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1000 | 1028.56 | 4390.98 | 8.02 | 12.51 |
| 950 | 969.42 | 4141.39 | 9.58 | 13.66 |
| 850 | 875.16 | 3920.89 | 15.12 | 18.17 |
| 750 | 769.45 | 3755.12 | 30.95 | 26.31 |
| 700 | 718.63 | 3561.14 | 46.28 | 31.69 |
| 675 | 694.41 | 3472.06 | 60.80 | 34.27 |
| 650 | 668.75 | 3413.29 | 90.52 | 39.19 |
| 630 | 673.59 | 3633.79 | 133.21 | 49.42 |
| 615 | 659.11 | 3585.00 | 175.56 | 67.30 |

Table 6.5.3.b. — UNIF/SRPT Scheduling (part b) - SSi-Implant

| safety factor × $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.0 | 31683.88 | 3751.08 | 21410.49 | 73.20 | 2411 |
| 1.6 | 43999.04 | 4383.57 | 24274.41 | 79.17 | 2821 |
| 2.5 | 59222.63 | 4857.40 | 28137.22 | 83.72 | 3049 |
| 3.4 | 72361.52 | 5343.33 | 30392.46 | 86.17 | 3286 |
| 4.0 | 78184.96 | 5024.20 | 29717.42 | 87.09 | 3394 |
| 4.6 | 86022.43 | 4627.43 | 27423.92 | 88.06 | 3427 |
| 5.2 | 93432.15 | 5097.97 | 30741.34 | 88.95 | 3484 |
| 5.5 | 95096.49 | 5249.88 | 31499.02 | 89.07 | 3562 |

Table 6.5.4.a. — SA/SA Scheduling (part a) - SSi-Implant

Note: $m$ = 1 is the number of machines at the bottleneck.

| safety factor × $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.0 | 1013.02 | 4525.48 | 8.59 | 10.52 |
| 1.6 | 865.94 | 4357.06 | 16.33 | 16.76 |
| 2.5 | 800.81 | 4043.40 | 28.37 | 24.73 |
| 3.4 | 719.94 | 3765.06 | 42.59 | 32.22 |
| 4.0 | 696.84 | 3776.32 | 52.09 | 36.17 |
| 4.6 | 672.58 | 3829.06 | 60.39 | 39.44 |
| 5.2 | 688.53 | 3981.23 | 67.81 | 42.96 |
| 5.5 | 654.24 | 3762.82 | 69.05 | 39.51 |

Table 6.5.4.b. — SA/SA Scheduling (part b) - SSi-Implant

Note: $m$ = 1 is the number of machines at the bottleneck.

### 6.5.1 — Interpretation of Simulation Experiment - SSi-Implant

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% 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% less delay than those produced by UNIF/SRPT (Fig. 6.5.2).

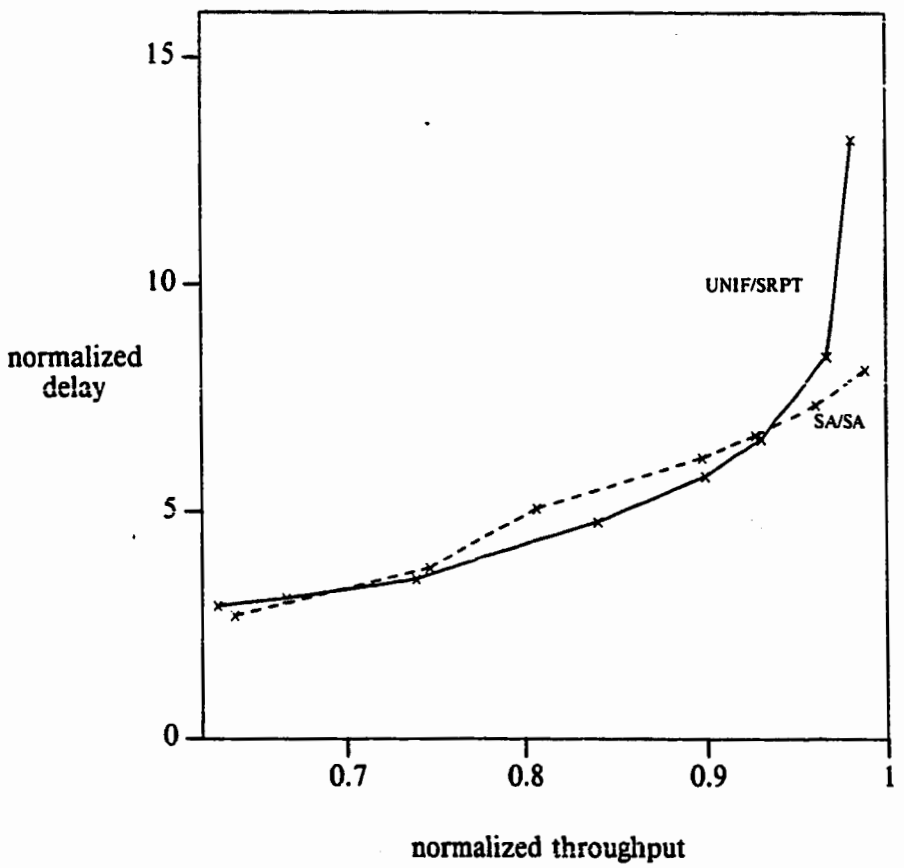


Fig. 6.5.2 – SA/SA vs UNIF/SRPT - SSi-Implant

CVID differences are negligible throughout (see Tables 6.5.3.b and 6.5.4.b).

### 6.6 – Another Production Fab Queueing Network - SSi-Aligner

In this Section we compare SA scheduling with UNIF release and SRPT dispatching on a production wafer fab queueing network generated with data obtained from the same computer integrated manufacturing system as in the last section. We modify the number of machines at some work stations so as to make the aligners, station $S_5$,

the single bottleneck station.

As stated earlier, the network has 41 work stations and one process with a recipe length of 182 steps. Table 6.6.1 summarizes the work stations of this fab.

| work station | number machines | load size | MTTR (mins) | MTBF (mins) |
|---|---|---|---|---|
| 1 | 5 | 1 | 500.47 | 3022.27 |
| 2 | 1 | 4 | 460.50 | 4962.00 |
| 3 | 2 | 4 | 281.50 | 16637.50 |
| 4 | 2 | 4 | 281.50 | 16637.50 |
| 5 | 2 | 1 | 1000.00 | 6000.00 |
| 6 | 1 | 9 | 42.00 | 111457.50 |
| 7 | 1 | 9 | 172.80 | 71592.30 |
| 8 | 5 | 1 | 15284.00 | 78228.00 |
| 9 | 2 | 4 | 850.15 | 6044.65 |
| 10 | 2 | 2 | 942.00 | 5668.50 |
| 11 | 3 | 1 | 0.00 | 113806.00 |
| 12 | 2 | 1 | 1792.36 | 32597.86 |
| 13 | 2 | 1 | 378.46 | 28305.46 |
| 14 | 1 | 4 | 1434.00 | 65833.50 |
| 15 | 1 | 4 | 913.00 | 71875.00 |
| 16 | 2 | 2 | 453.00 | 19522.50 |
| 17 | 5 | 2 | 10664.57 | 73076.57 |
| 18 | 2 | 4 | 4426.00 | 175978.00 |
| 19 | 1 | 2 | 1392.00 | 101055.00 |
| 20 | 5 | 1 | 13357.09 | 51441.09 |
| 21 | 2 | 4 | 715.38 | 4922.88 |
| 22 | 3 | 1 | 6748.50 | 114954.50 |
| 23 | 2 | 4 | 954.00 | 6427.50 |
| 24 | 2 | 4 | 1006.50 | 5320.50 |
| 25 | 1 | 2 | 372.00 | 2275.50 |
| 26 | 3 | 1 | 500.47 | 3022.27 |
| 27 | 5 | 1 | 264.75 | 1256.25 |
| 28 | 1 | 1 | 517.64 | 8661.14 |
| 29 | 1 | 2 | 23596.00 | 87832.00 |
| 30 | 4 | 1 | 824.77 | 7306.27 |
| 31 | 6 | 1 | 184.80 | 9867.30 |
| 32 | 10 | 1 | 698.40 | 8048.40 |
| 33 | 1 | 1 | 6748.50 | 114954.50 |
| 34 | 1 | 4 | 1512.00 | 5853.00 |
| 35 | 4 | 1 | 295.38 | 3193.38 |
| 36 | 1 | 1 | 1321.85 | 6502.85 |
| 37 | 2 | 4 | 1073.54 | 5078.54 |
| 38 | 2 | 4 | 114.00 | 35604.00 |
| 39 | 2 | 4 | 138.00 | 28021.50 |
| 40 | 3 | 1 | 2722.50 | 9895.50 |
| 41 | 1 | 6 | 167.54 | 1787.54 |

Table 6.6.1 — Work Stations – SSi-Aligner

Machines in stations $S_{11}$ are assumed to never fail. The single process recipe of

the fab is the same as in Table 6.5.2.

As before, hot lots are not allowed to be started. The nominal capacity of the fab network is one lot every 540.00 minutes (18.67 lots/week). Its expected capacity is one lot every 630.00 minutes (16.1 lots/week). Simulations are run with a horizon $H$ = 2 million minutes (about 46 months), each requiring approximately 50 minutes of CPU time on the Cray X-MP/14.

Tables 6.6.2.a and 6.6.2.b summarize runs with UNIF/SRPT scheduling, while Tables 6.6.3.a and Table 6.6.3.b show results for SA/SA scheduling.

| start interval | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 640 | 17204.50 | 2337.97 | 12228.57 | 59.49 | 2969 |
| 635 | 18304.03 | 2383.48 | 12503.37 | 60.99 | 2992 |
| 630 | 21171.77 | 2441.49 | 12446.01 | 64.41 | 3016 |
| 625 | 25418.44 | 2685.52 | 13721.05 | 68.53 | 3041 |
| 620 | 31464.80 | 3093.38 | 15609.13 | 72.93 | 3064 |

Table 6.6.2.a. — UNIF/SRPT Scheduling (part a) - SSi-Aligner

| start interval | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 640 | 640.87 | 2350.92 | 12.25 | 7.03 |
| 635 | 636.58 | 2359.42 | 13.86 | 7.16 |
| 630 | 634.29 | 2322.03 | 18.54 | 7.61 |
| 625 | 630.77 | 2354.51 | 25.44 | 10.26 |
| 620 | 628.94 | 2313.16 | 34.96 | 15.34 |

Table 6.6.2.b. — UNIF/SRPT Scheduling (part b) - SSi-Aligner

| safety factor × $m$ | mean delay | ± 95% delay confidence interval | std. dev. delay | % delay/ cycle time | job starts |
|---|---|---|---|---|---|
| 1.2 | 15218.28 | 2213.99 | 12134.98 | 56.43 | 2945 |
| 1.4 | 15224.22 | 2200.99 | 11994.12 | 56.54 | 2950 |
| 1.6 | 16142.28 | 2219.82 | 11903.33 | 57.90 | 2991 |
| 1.8 | 16362.69 | 2249.93 | 12021.93 | 58.20 | 2993 |

Table 6.6.3.a. — SA/SA Scheduling (part a) – SSi-Aligner

Note: $m$ = 1 is the number of machines at the bottleneck.

| safety factor × $m$ | mean departure interval | std. dev. departure interval | mean bottleneck queue | std. dev. bottleneck queue |
|---|---|---|---|---|
| 1.2 | 643.53 | 2346.77 | 8.71 | 6.96 |
| 1.4 | 643.37 | 2353.50 | 8.92 | 7.12 |
| 1.6 | 632.44 | 2316.62 | 10.21 | 7.08 |
| 1.8 | 631.70 | 2340.79 | 10.47 | 7.26 |

Table 6.6.3.b. — SA/SA Scheduling (part b) – SSi-Aligner

Note: $m$ = 1 is the number of machines at the bottleneck.

### 6.6.1 — Interpretation of Simulation Experiment - SSi-Aligner

For SSi-Aligner the range of throughput values generated went from 98-100.2% of expected capacity. In this range SA scheduling produced mean delays varying from 88% to 65% of those produced by UNIF/SRPT (Fig. 6.6.2).

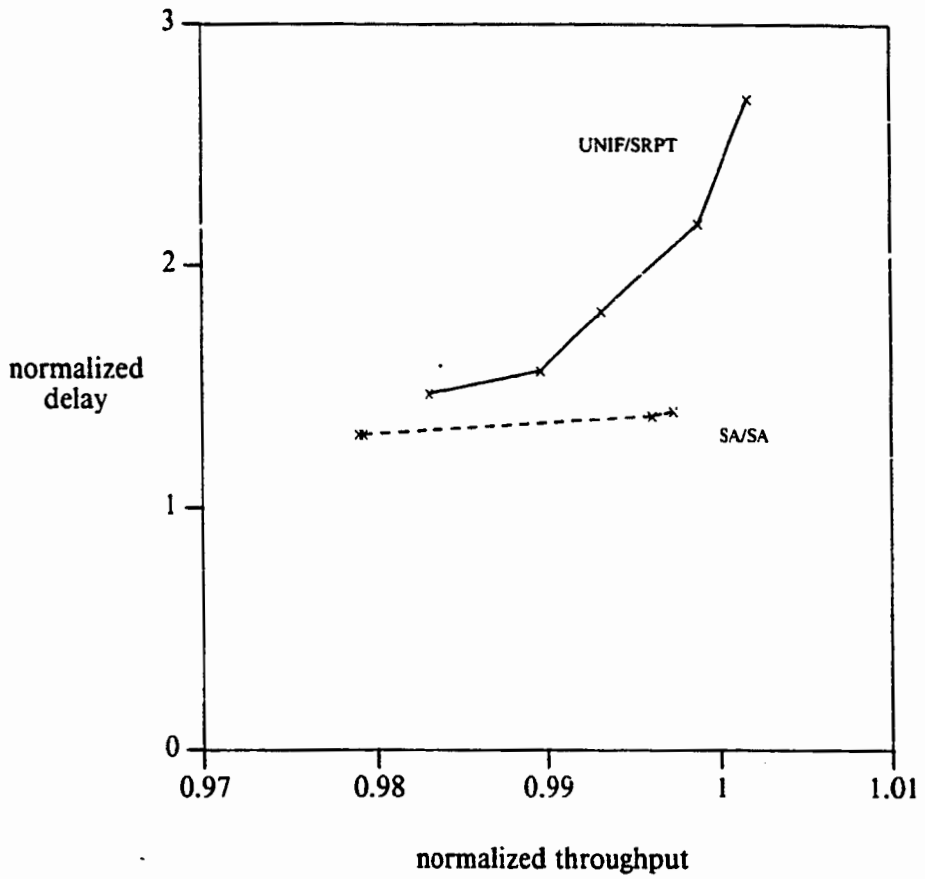


Fig. 6.6.2 — SA/SA vs UNIF/SRPT - SSi-Aligner

## 6.7 — Summary

In this Chapter we present results of testing many scheduling policies on several wafer fab queueing networks running FabSim on the Cray X-MP/14 at U.C.-Berkeley. Networks vary from small (5 stations) to large (41 stations) in size. Starvation Avoidance (SA) compares favorably against the scheduling policies tested, displaying increased improvement as the throughput approaches the network capacity.

# Chapter 7 — Concluding Remarks

In this dissertation we investigate the problem of real-time scheduling of wafer batches on unreliable machines in a semiconductor manufacturing facility (fab). We assume global factory information is readily available through a computer integrated manufacturing system. 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, the expected inventory of work awaiting 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 set ups. In this research we are concerned only with the first tradeoff.

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. 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 re-entrant 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 unrecoverable loss of final output.

One conclusion we have reached from our work and that of others [Bur86a, Wei86a] is that dispatching decisions are not as important as release decisions. Effective flow control requires a sufficiently large inventory of raw material so that new work can be introduced whenever desired. While such policies shift

inventory 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 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. In fact, wafers are usually released in lots of 25 or 50. A second release strategy is the *Fixed-WIP* rule which starts a new lot of wafers whenever a lot of wafers is completed. Wein [Wei86a] 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 one hour of work at the bottleneck whenever bottleneck machine completes an hour of work. He calls this strategy *Workload Regulating* release. Our proposed strategy is similar to Wein's but is derived from a somewhat different approach. We start with a simple idea: 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.

Starvation Avoidance was tested extensively on several networks and compared to other scheduling policies. 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 it is nega-

tively 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 re-entrant flows, Workload Regulating counts all the work remaining at the bottleneck on each lot, not just the next bottleneck operation. Let $L$ = 0.9 hour and consider two cases of an (almost) empty shop. In case 1, two jobs are in queue at the bottleneck, each with 0.5 hours 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 hours, and one hour 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.

We can make the following conclusions about SA.

- 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.

- Near capacity, SA outperformed all other policies in all test cases.
- 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 upwards (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.
- SA is easy to implement, provided a CIM system with up-to-date shop floor information is available. In practice, the throughput control parameter $\alpha$ used in SA must be set. This parameter controls the factory throughput. One way to set $\alpha$ is with simulation. This, however, requires a validated simulation model which 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 $\alpha$ one can still maintain a high throughput rate, while reducing considerably the mean delay. In practice, $\alpha$ 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.

Future research should determine if the results for the single process, single bottleneck case described in this thesis hold for the several multi-process, multi-

bottleneck cases. In this dissertation 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**

Ano81a.

Anonymous, "Ten-Year Worldwide Forecast for Electronic Equipment and Components," *Electronic Business*, Feb. 1981.

Ano82a.

Anonymous, in *Electronic Market Data Book 1982*, Electronic Industries Association, Washington, D.C., 1982.

Ano82b.

Anonymous, "World Markets Forecast for 1982," *Electronics*, vol. 55, no. 1, 1982.

Ash70a.

S. Ashour, "An Experimental Investigation and Comparative Evaluation of Flow-Shop Scheduling Techniques," *Operations Research*, vol. 18, no. 3, 1970.

Ash74a.

S. Ashour, T.E. Moore, and K.Y. Chiu, "An Implicit Enumeration Algorithm for the Nonpreemptive Shop Scheduling Problem," *AIIE Transactions*, vol. 6, 1974.

Bak60a.

C.T. Baker and B.P. Dzielinski, "Simulation of a Simplified Job Shop," *Management Science*, vol. 6, no. 3, 1960.

Bak68a.

K.R. Baker, "Priority Dispatching in the Single Channel Queue with Sequence-Dependant Setups," *Journal of Industrial Engineering*, vol. 19, no. 9, 1968.

Bak74a.

K.R. Baker, *Introdution to Sequencing and Scheduling*, John Wiley and Sons, New York, 1974.

Bak84a.

Kenneth R. Baker, "Sequencing Rules and Due-Date Assignments in a Job Shop," *Management Science*, vol. 30, no. 9, 1984.

Bak81a.

Kenneth R. Baker and J.W.M. Bertrand, "A Comparison of Due-Date Selection Rules," *AIIE Transactions*, June 1981.

Bak80a.

K.R. Baker and H.L.W. Nuttle, "Sequencing Independent Jobs Within a Single Resource," *Nav. Res. Logist. Q.*, vol. 27, 1980.

Bal69a.

Egon Balas, "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm," *Operations Research*, vol. 17, no. 6, 1969.

Bar86a.

M. Barry, Private Communication, Fairchild Semiconductor Corporation Research Center, Palo Alto, CA, 1986.

Ber70a.

R.R. Berger, "Critical Ratio Simplified," *Production and Inventory Management*, vol. 11, 1970.

Ber84a.

William L. Berry, Richard J. Penlesky, and Thomas E. Vollmann, "Critical Ratio Scheduling: Dynamic Due-Date Procedures Under Demand Uncertainty," *IEE Transactions*, March 1984.

Ber75a.

William L. Berry and Vittal Rao, "Critical Ratio Scheduling: An Experimental Analysis," *Management Science*, vol. 22, no. 2, 1975.

Bil79a.

William E. Biles and James J. Swain, "Mathematical Programming and the Optimization of Computer Simulations," in *Mathematical Programming Study 11*, North-Holland Publishing Company, 1979.

Bla82a.

John H. Blackstone Jr., Don 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, 1982.

Bor76a.

I. Borosh and L.B. Treybig, "Bounds on Positive Integral Solutions of Linear Diophantine Equations," *Proc. Amer. Math. Soc.*, vol. 55, 1976.

Bow59a.

E.H. Bowman, "The Scheduling-Sequencing Problem," *Operations Research*, vol. 7, 1959.

Box65a.

M.J. Box, "A New Method of Constrained Optimization and a Comparison with Other Methods," *The Computer Journal*, vol. 8, 1965.

Bro65a.

G.H. Brooks and C.R. White, "An Algorithm for Finding Optimal or Near-Optimal Solutions to the Production Scheduling Problem," *Journal of Industrial Engineering*, vol. 16, no. 1, 1965.

Bun85a.

Panit Bunnag and Spencer B. Smith, "A Multifactor Priority Rule for Job-Shop Scheduling Using Computer Search," *IEE Transactions*, June 1985.

Bur86a.

David Y. Burman, F. Javier Gurrola-Gal, Ardavan Nozari, Shashikant Sathaye, and Joseph P. Sitarik, "Performance Analysis Techniques for IC Manufacturing

Lines," *AT&T Technical Journal*, vol. 65, no. 4, 1986.

Car65a.

D.C. Carroll, "Heuristic Sequencing of Single and Multiple Component Jobs," Unpublished Ph.D. Dissertation, Sloan School of Management, M.I.T., Cambridge, MA, 1965.

Che86a.

Hong Chen, J. Michael Harrison, Avi Mandelbaum, Ann van Ackere, and Lawrence M. Wein, "Queueing Network Models of Semiconductor Wafer Fabrication," Technical Report, Center for Integrated Systems, Stanford University, Stanford, CA 94305, 1986.

Cof76a.

E.G. Coffman Jr., *Computer and Job-Shop Scheduling Theory*, John Wiley, New York, NY, 1976.

Con65a.

Richard W. Conway, "Priority Dispatching and Work-in-Process Inventory in a Job Shop," *Journal of Industrial Engineering*, vol. 16, no. 4, 1965.

Con65b.

Richard W. Conway, "Priority Dispatching and Job Lateness in a Job Shop," *Journal of Industrial Engineering*, vol. 16, no. 4, 1965.

Con60a.

Richard W. Conway, Bruce M. Johnson, and William L. Maxwell, "An Experimental Investigation of Priority Dispatching," *Journal of Industrial Engineering*, vol. 11, no. 3, 1960.

Con52a.

R.W. Conway and W.L. Maxwell, "Network Dispatching by the Shortest-Operation Discipline," *Operations Research*, vol. 10, 1962.

Con67a.

R.W. Conway, W.L. Maxwell, and L.W. Miller, *The Theory of Scheduling*, Addison-Wesley, 1967.

Dan60a.

G.B. Dantzig, "A Machine-Shop Scheduling Model," *Management Science*, vol. 6, 1960.

Day70a.

James E. Day and Michael P. Hottenstein, "Review of Sequencing Research," *Naval Research Logistics Quarterly*, vol. 17, 1970.

Day87a.

Judith Dayhoff, "New Techniques for Simulation Modeling and Analysis ," *CIM Review*, Winter 1987.

Day84a.

Judith E. Dayhoff and Robert W. Atherton, "Simulation of VLSI Manufacturing Areas," *VLSI Design*, Dec. 1984.

Dzi63a.

B.P. Dzielinski, C.T. Baker, and A.S. Manne, "Simulation Tests of Lot-Size Programming," in *Industrial Scheduling*, ed. by G.L. Thompson, Prentice-Hall, Englewood Cliffs, NJ, 1963.

Eil75a.

S. Eilon, I.G. Choudury, and S.S. Serghioug, "Experiments with SI Rule in Job Shop Scheduling," *Simulation*, vol. 24, 1975.

Eil68a.

S. Eilon and D.J. Cotterill, "A Modified SI Rule in Job Shop Sequencing," *International Journal of Production Research*, vol. 7, 1968.

Elv73a.

D.A. Elvers, "Job Shop Dispatching Rules Using Various Delivery Date Setting Criteria," *Production Inventory Management*, vol. 14, 1973.

Fis63a.

H. Fischer and G.L. Thompson, "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules," in *Industrial Scheduling*, ed. by G.L. Thompson, Prentice-Hall, Englewood Cliffs, N.J., 1963.

Fis73a.

M.L. Fisher, "Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part I," *Operations Research*, vol. 21, no. 5, 1973.

Fis73b.

M.L. Fisher, "Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part II," in *Symposium on the Theory of Scheduling and its Applications*, ed. by S.E. Elmaghraby, Springer-Verlag, Berlin, 1973.

Fis71a.

George S. Fishman, "Estimating Sample Size in Computing Simulation Experiments," *Management Science*, vol. 18, no. 1, 1971.

Fre86a.

Albert A. Fredericks, "Performance Analysis Modeling for Manufacturing Lines," *AT&T Technical Journal*, vol. 65, no. 4, 1986.

Fre82a.

S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, 1982.

Gar84a.

K. Gardiner, "Editorial," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, vol. CHMT-7, no. 3, 1984.

Gar76a.

M.R. Garey, D.S. Johnson, and R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," *Mathematics of Operations Research*, vol. 1, 1976.

Ger66a.

William S. Gere Jr., "Heuristics in Job Shop Scheduling," *Management Science*, vol. 13, no. 3, 1966.

Gis79a.

P. Gise and R. Blanchard, *Semiconductor and Integrated Circuit Fabrication Techniques,* Prentice-Hall (Reston), Englewood Cliffs, NJ, 1979.

Gis86a.

P. Gise and R. Blanchard, *Modern Semiconductor Fabrication Technology,* Prentice-Hall (Reston), Englewood Cliffs, NJ, 1986.

Gli60a.

B. Gliffer and G.L. Thompson, "Algorithms for Solving Production Scheduling Problems," *Operations Research*, vol. 8, no. 4, 1960.

Gon77a.

M.J. Gonzalez, "Deterministic Processor Scheduling," *Assoc. Comput. Mach. Surveys*, vol. 9, 1977.

Gra79a.

R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals of Discrete Mathematics*, vol. 5, 1979.

Gra81a.

Stephen C. Graves, "A Review of Production Scheduling," *Operations Research*, vol. 29, no. 4, 1981.

Gre68a.

Harvey Greenberg, "A Branch and Bound Solution to the General Scheduling

Problem," *Operations Research*, vol. 16, no. 2, 1968.

Har85a.

C. Harrison and P.J. O'Grady, "A General Search Sequencing Rule for Job Shop Scheduling," *International Journal of Production Research*, vol. 23, no. 5, 1985.

Hax84a.

Arnaldo C. Hax and D. Candea, *Production and Inventory Management*, Prentice-Hall, 1984.

Her75a.

James C. Hershauer and Ronald J. Ebert, "Search and Simulation Selection of a Job-Shop Sequencing Rule," *Management Science*, vol. 21, no. 7, 1975.

Hoo61a.

Robert Hooke and T.A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," *Journal of the Association of Computing Machinery*, vol. 8, no. 2, 1961.

Jac55a.

J.R. Jackson, "Scheduling a Production Line to Minimize Maximum Tardiness," Research Report 43, Management Science Research Project, University of California, Los Angeles, CA, 1955.

Jac56a.

J.R. Jackson, "An Extension of Johnson's Results on Job-Lot Scheduling," *Naval Research Logistics Quarterly*, vol. 3, no. 3, 1956.

Jac57a.

J.R. Jackson, "Networks of Waiting Lines," *Operations Research*, vol. 5, no. 4, 1957.

Jac57b.

J.R. Jackson, "Simulation Research of Job-Shop Production," *Nav. Res. Logist. Q.*, vol. 4, no. 4, 1957.

Jac63a.

J.R. Jackson, "Job Shop-Like Queueing Systems," *Management Science*, vol. 9, no. 1, 1963.

Jac57c.

J.R. Jackson and R.T. Nelson, "SWAC Computations for Some m x n Scheduling Problems," *J. Assoc. Computing Machinery*, vol. 4, no. 4, 1957.

Jon73a.

C.H. Jones, "An Economic Evaluation of Job Shop Dispatching Rules," *Management Science*, vol. 20, no. 3, 1973.

Kar72a.

R.M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*, ed. by J.W. Thatcher, Plenum Press, New York, NY, 1972.

Kli74a.

G.P. Klimov, "Time Sharing Service Systems I," *Theory of Probability and Applications*, vol. 19, 1974.

Kur60a.

Y. Kuratani and R.T. Nelson, "A Pre-Computational Report on Job-Shop Simulation Research," *J. Oper. Res. Society of Japan*, vol. 2, no. 4, 1960.

Lag77a.

B.J. Lageweg, J.K. Lenstra, and A.H.G. Rinnooy Kan, "Job-Shop Scheduling by Implicit Enumeration," *Management Science*, vol. 24, no. 4, 1977.

Law83a.

A.M. Law, "Statistical Analysis of Simulation Output Data," *Operations*

*Research*, vol. 31, no. 6, 1983.

Law82a.

Averill M. Law and W. David Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, Inc., 1982.

LeG66a.

E. LeGrande, "The Development of a Factory Simulation System Using Actual Operating Data," in *Readings in Production and Operations Management*, ed. by E.S. Buffa, John Wiley, New York, NY, 1966.

Lea86a.

R.C. Leachman, "Preliminary Design and Development of a Corporate-Level Production Planning System for the Semiconductor Industry," ORC 86-11, Operations Research Center, University of California, Berkeley, CA 94720, December 1986.

Lem77a.

A.J. Lemoine, "Networks of Queues - A Survey of Equilibrium Analysis," *Management Science*, vol. 24, no. 4, 1977.

Len86a.

Jan Karel Lenstra, "Sequencing and Scheduling: The Tools Available," ORSA/TIMS Joint National Meeting, Miami, Florida, Oct. 1986.

Len80a.

J.K. Lenstra and A.H.G. Rinnooy Kan, "An Introduction to Multiprocessor Scheduling," Preprint BW121/80, Mathematisch Centrum, Amsterdam, Holland, 1980.

Len84a.

J.K. Lenstra and A.H.G. Rinnooy Kan, "New Directions in Scheduling Theory," *Operations Research Letters*, vol. 2, no. 6, 1984.

Lit61a.

John D.C. Little, "A Proof for the Queueing Formula: $L = \lambda W$," *Operations Research*, vol. 9, no. 3, 1961.

Loh84a.

E. Lohrasbpour and S. Sathaye, "Simulation Modeling of IC Wafer Fabrication Lines," in *Technical Program Proceedings, Semicon/WEST*, 1984.

Loz86a.

Christopher Lozinski, "A Scheduling Perspective on Semiconductor Wafer Fabrication," Manuscript, Dept. Industrial Engineering and Operations Research, University of California, Berkeley, CA, 1986.

Man60a.

A.S. Manne, "On the Job-Shop Scheduling Problem," *Operations Research*, vol. 8, no. 2, 1960.

Mel66a.

P. Mellor, "A Review of Job Shop Scheduling," *O.R. Quarterly*, vol. 17, 1966.

Moo80a.

G. Moore, "VLSI, What Does the Future Hold," *Electro. Aust.*, vol. 42, 1980.

Moo67a.

J.M. Moore and R.C. Wilson, "A Review of Simulation Research in Job Shop Scheduling," *Prod. Inventory Management*, vol. 8, no. 1, 1967.

Mor58a.

P.M. Morse, *Queues, Inventory, and Maintenace*, John Wiley, New York, NY, 1958.

Mut63a.

J.F. Muth and G.L. Thompson, in *Industrial Scheduling*, ed. by G.L. Thompson, Prentice-Hall, Englewood Cliffs, NJ, 1963.

Nan63a.

Y.R. Nanot, "An Experimental Investigation and Comparitive Evaluation of Priority Disciplines in Jobshop-like Queueing Networks," Research Report No. 87, Management Science Research Project, UCLA, 1963.

Nel67a.

Rosser T. Nelson, "Labor and Machine Limited Production Systems," *Management Science*, vol. 13, no. 9, 1967.

Noy81a.

R.N. Noyce, "Microelectronics," in *The Microelectronic Revolution*, ed. by T. Forester, MIT Press, Cambridge, MA, 1981.

Old77a.

W.G. Oldham, "The Fabrication of Microelectronic Circuits," *Scientific American*, vol. 237, no. 3, 1977.

Ora73a.

M. Oral and J.L. Malouin, "Evaluation of the Shortest Processing Time Scheduling Rule with Truncation Process," *AIIE Transactions*, vol. 31, 1973.

Pan77a.

S.S. Panwalker and Wafik Iskander, "A Survey of Scheduling Rules," *Operations Research*, vol. 25, no. 1, 1977.

Pen84a.

Paul Penfield, Jr., Stanley B. Gershwin, David A. Hodges, Carlton M. Osburn, John Reynolds, John Shott, Andrew J. Steckl, and Donald E. Troxel, "Requirements for Computer-Aided Fabrication," Unpublished Document, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, January 17, 1984.

Put71a.

A.O. Putnam, R. Everdell, G.H. Dorman, R.R. Cronan, and L.H. Lindgren, "Updating Critical Ratio and Slack Time Priority Scheduling Rules," *Production and Inventory Management*, vol. 12, 1971.

Rag84a.

Gary L. Ragatz and Vincent A, Mabert, "A Framework for the Study of Due Date Management in Job Shops," *International Journal of Production Research*, vol. 22, no. 4, 1984.

Rei63a.

R. Reinitz, "On the Job Shop Scheduling Problem," in *Industrial Scheduling*, ed. by G.L. Thompson, Prentice-Hall, Englewood Cliffs, NJ, 1963.

Rei66a.

S. Reiter, "A System for Managing Job Shop Production," *J. Business*, vol. 39, no. 3, 1966.

Rin76a.

A.H.G. Rinnooy Kan, *Machine Scheduling Problems: Classification, Complexity and Computations*, Martinus Nijhoff, The Hague, Holland, 1976.

Roc76a.

R. Rochette and R.P. Sadowshi, "A Statistical Comparison of the Performance of Simple Dispatching Rules for a Particular Set of Job Shops," *Inter. J. Production Res.*, vol. 14, 1976.

Row60a.

A.J. Rowe, "Toward a Theory of Scheduling," *Journal of Industrial Engineering*, vol. 11, no. 2, 1960.

Sch68a.

L. Schrage, "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline," *Operations Research*, vol. 16, no. 3, 1968.

Sei81a.

Abraham Seidmann and Milton L. Smith, "Due Date Assignment for Production Systems," *Management Science*, vol. 27, no. 5, 1981.

Sen84a.

Tapan Sen and Sushil K. Gupta, "A State-of-the-Art Survey of Static Scheduling Research Involving Due Dates," *OMEGA*, vol. 12, no. 1, 1984.

Sis59a.

R.L. Sisson, "Methods of Sequencing in Job Shops — A Review," *Operations Research*, vol. 7, no. 10, 1959.

Sis61a.

R.L. Sisson, "Sequencing Theory," in *Progress in Operations Research, Vol. I*, ed. by R.L. Ackoff, John Wiley, New York, 1961.

Smi73a.

Dennis E. Smith, "Requirements of an Optimizer for Computer Simulations," *Naval Research Logistics Quarterly*, vol. 20, 1973.

Smi56a.

W.E. Smith, "Various Optimizers for Single-Stage Production," *Naval Res. Logist. Q.*, vol. 3, 1956.

Spe86a.

Anne M. Spence and Douglas J. Welter, Private Communication, 1986.

Spe62a.

W. Spendley, G.R. Hext, and R.F. Himsworth, "Sequential Application of Simplex Designs in Optimization and Evolutionary Operations," *Technometrics*, vol. 4, 1962.

Sta80a.

T.A. Standish, *Data Structure Techniques*, 1980.

Sto63a.

A.E. Story and H.M. Wagner, "Computational Experience with Integer Programming for Job-Shop Scheduling," in *Industrial Scheduling*, ed. by G.L. Thompson, Prentice-Hall, Englewood Cliffs, NJ, 1963.

Sze83a.

S.M. Sze, *VLSI Technology*, McGraw-Hill, 1983.

Tau68a.

William H. Taubert, "A Search Decsion Rule for the Aggregate Scheduling Problem," *Management Science*, vol. 24, no. 6, 1968.

Wag59a.

H.M. Wagner, "An Integer Programming Model for Machine Scheduling," *Nav. Res. Logis. Q.*, vol. 6, 1959.

Was72a.

W. Wassweiler, "Material Requirements Planning — The Key to Critical Ratio Effectiveness," *Production and Inventory Management*, vol. 13, 1972.

Wat86a.

Henry A. Watts, "Scheduling an Automated Manufacturing Line," Manuscript, PROMIS Systems Corporation, Santa Clara, CA, 1986.

Wei86a.

Lawrence M. Wein, Private Communication, 1986.

Wil69a.

Jon K. Wilbrecht and William B. Prescott, "The Influence of Setup Time on Job Shop Performance," *Management Science*, vol. 16, no. 4, 1969.