1 Lecture 1: Introduction

- Wafer fabs and the need to control them.
- Scheduling approach, queueing approach, finite horizon online formulation
- 3 step procedure
  - fluid formulation: is it a lower bound?
  - fluid solution: simplex algorithm for SCLP (demo)
  - fluid imitation: is it close to the fluid solution?
- How is this to be implemented? When should it work well?
- Video of the fab process — Silicon Run by Ruth Carranza.

1.1 Wafer Fabs

Wafer fabs are the plants in which computer microchips are built on top of pure silicon wafers. Starting from a pure silicon monocrystal round wafer, up to 300 mm in diameter and up to 0.775 mm thick, several hundred computer microchips are built on the surface of the wafer, in the most advanced, costly and sophisticated industrial process known to man.

The process is carried out in a clean room (which is achieved at a huge expense), is mostly automated, and involves hundreds of steps at dozens of machines. Furnaces at 3000\(^\circ\)C, chemicals such as pure hot Hydro-Fluoric acid, Ultraviolet (and soon to come x-ray) radiation, Ionization chambers, you name it, they've got it. Essentially it is a chemical photo-lythographic process of imprinting patterns, and implanting ions and overlaying metals according to these patterns, layer after layer, so as to build complete electronic circuits on the microchip at extreme miniaturization. The completed circuitry of a microchip is equivalent to 6,000,000 transistors (Pentium chip). We will watch it in the following ‘Silicon Run’ video of Ruth Carranza.

Wafer fabs evolved from 150–200 mm wafer size in 1991 to 300 mm in 2000. Feature size in that period stepped down from 500 nm in 1992, 350 nm in 1995, 250 nm in 1998, 180 nm in 2001, and soon to come 130 nm. Price is up from $150,000,000 (e.g Tower semiconductor ltd., Migdal Haemek, Israel, 1991), $1,500,000 (Intel Kiriyat Gat, Israel, 1998) to $3,000,000,000 or even $6,000,000,000 (IBM Fishkill plant) for the modern fabs coming on line now.

A $3,000,000,000 plant is supposed to return the investment in 3 years, which means $1,500,000,000 revenue per year — I guess this would be for the completed chips, as sold say by Intel, to the end user. The completed chips, coming from a single completed 300mm wafer with several hundred Pentium chips, will eventually retail for some $250,000 (the original silicon wafer costs a few hundred dollars only, so this is a 3 orders of magnitude leap in value).

One feature of the wafer fabs that seems fairly constant throughout the late developments seems to be a cycle time (from blank silicon wafer to completed microchips on the wafer) of 6 weeks. This gives (Little’s formula) a $180,000,000 price tag to the finished goods value of the WIP inventory.
Figure 1: Generic 130-nm Semiconductor Process Flow Showing Massively Reentrant Line

Counting the WIP in wafers, a typical new mega-fab ships 10,000 wafer per week, which is almost exactly one wafer per minute. This (by Little’s formula) will give an average WIP of 60,000 wafers (mostly they travel in lots of 24, so that is 5000 lots, one leaves about every half hour). These numbers give some justification for the use of a fluid model.

I am not quite sure that my information above in terms of $ counts and in terms of wafer counts fit with each other. In the discussion of "the business of wafer fabrication" (Chapter 15 of vanZant’s standard wafer fabrication text, [?]) wafer prices are thought to be typically $3000 and inventories are quoted as 40,000 wafers, though that sounds very tentative. I am very shaky about these numbers (which the wafer fabs of course do not reveal) and since they are crucial for application of our models and procedures, I will welcome any help/discussion from the audience.

To appreciate the full scale note that the worldwide semiconductor revenue was over $203 billion in 2000 (with 914 fabs in current operation), and is projected to be nearly $283 billion by 2004 (with an additional 38 fabs on line). Whether they admit it or not, I believe that the methods used to schedule and control these fabs currently are as-hoc, and lack a suitable theoretical basis. Our aim is to provide a theory for the control wafer fab WIP on its journey through the plant.

Figure 1 shows a generic 130-nm semiconductor process flow using copper interconnect and low-k dielectrics. Perhaps the most notable feature of this process is the highly reentrant product flows: Even an "optimized" ordering of tool types to minimize the number of product flow reentry points has the flow go back to an earlier tool type 93 times. Other features of wafer fabs that complicate analysis include large variation in processing times between different
types of equipment (for example, 1 minute in a wet bench vs. 10 hours in a furnace), large variation in processing times at different stages on the same type of equipment, setup times for switching between product types, and batching at certain types of equipment.

In what follows we will ignore setup times and batching issues, which are local, and focus on the re-entrant process as encompassing the state of the whole plant. We now outline our proposed procedure to handle it.

1.2 Problem Formulation

We use the following model to describe the wafer fab and its process:

The production process consists of steps \( k = 1, \ldots, K \). These are partitioned between machines \( i = 1, \ldots, I \): \( k \in C_i, i = \sigma(k) \). Successive wafers are indexed by \( j = 1, \ldots, \), and require processing times \( X_k(j) \). Denote by \( A(j) \geq 0 \) the arrival (release) time of wafer \( j \), and by \( D(j) \) its departure.

Plants come in 3 forms: Single product fabs e.g. intel pentium chips (Intel Kiryat Gat, Israel), Foundries which do chips for anyone, and run many differing small orders (Tower semiconductors ltd., Migdal Haemek, Israel), and fabs which are inbetween. However, even in a foundry, most wafers will follow the same process steps, even if they will require different processing times for these steps. Some plants may have as many as 5 different processes though (Harris Semiconductor, Palm Bay, FL.).

Costs are a function of the departure processes. Three examples are:

**Makespan**: Time to complete the last (of some \( N \) jobs), that is \( \max_j D_K(j) \). Because we minimize the time to produce \( N \) items we maximize the system throughput.

**Weighted flow time**: Flow time of a wafer, also known as cycle time and (though I object to that name) throughput time, is the total time spent by the wafer in the system. The cost here is the sum or the average over the wafers. By Little’s formula if is equal to charging at rate 1 for the WIP in the fab. It also is equivalent to minimizing \( \sum_j D_K(j) \). Putting weights on flowtime can be done in two ways. Either on the items, \( w_j \) for the time wafer \( j \) spends in the system, or on the buffer \( w_k \) rate of cost per time per wafer while in buffer \( k \).

**Due date adherence** Wafers have due dates, and wafer \( j \) pays \( h(D_K(j) - d(j)) \) where the penalty function can be \( h(x) = x, h(x) = x^+, h(x) = 1 \{x > 0\} \).

Due date adherence is often quoted as the most important objective of the fab. I believe that control of WIP so as to minimize cycle times is more important. This is especially so if one can achieve low cycle times together with small variability in the cycle times of the wafers, and if one can achieve a good level of overall control of the WIP so that departure times become more predictable. This can be used in setting due dates and improve due date adherence. Makespan is not a reasonable objective but studying it is quite important. We will concentrate on makespan and on weighted \( w_k \) holding costs.

We now discuss briefly two approaches to scheduling and control of manufacturing systems: Scheduling theory and multiclass queueing networks.
1.2.1 Scheduling Approach

Scheduling theory uses methods of combinatorial optimization. It is assumed that we have a finite batch of $N$ wafers (jobs) to schedule. It is also assumed that we know all the arrival (release) times and processing requirements of all the jobs. There are then a finite number of schedules among which we need to choose the best.

Use $A(j), X_k(j)$ of jobs (wafers) $j = 1, \ldots, N$ to determine the optimal schedule. A schedule consists of the assignment of start and finish times to all operations, $s_k(j), t_k(j)$. The objective is a cost function $V(D(1), \ldots, D(N))$, which is monotone increasing in the departure times. The optimal schedule chooses among the finite set of possible schedules one which minimizes the objective.

The problem of scheduling a set of $N$ jobs on a $K$ step re-entrant line, for all the above cost functions, is NP-hard (except $I = 1$, or $K = 2$ with makespan objective). This means there is no efficient algorithm to solve them. In fact small problems (e.g. $N = K = I = 10$) are very hard or impossible to solve. Our wafer fabs are impossible to solve.

However, there is a bigger issue here. Suppose we got the optimal solution. It would be a complete optimal schedule for the whole fab, giving exact start and finish time for each operation. When we start implementing it we will know what to do. But very soon the real processing will deviate from the schedule (due to errors in processing times, to machine and operator unavailabilities, etc.). The deviations will increase as the schedule progresses. Furthermore they will consist of many small deviation as well as a few very large deviations. By the time we implement it we will not be using the optimal schedule, and we have no idea how robust it is to all the deviations.

While we need a schedule for the initial time, the attempt to calculate an optimal schedule for the whole batch seems therefore to be futile. We use too much imprecise data, and we get too many details of the solution.

1.2.2 Queueing Approach

We regard the system as a Multi-Class Queueing Network (MCQN). Let $A(t), D(t)$ denote the cumulative $(0, t]$ vector arrival and departure processes, $Q(t)$ the vector queue length process (jobs which completed steps prior to $k$ but not step $k$) — we refer to these jobs also as contents of buffer $k$. The MCQN dynamics are described by:

\begin{align*}
Q(t) &= Q(0) + A(t) - D(t) \\
D_k(t) &= S_k(T_k(t)), \\
A_{k+1}(t) &= D_k(t),
\end{align*}

where $A_1$ is the input, $T_k(t)$ the time assigned by machine $\sigma(k)$ to buffer $k$, and $S_k(\cdot)$ the resulting service completions of the sequence $X_k(j)$. Here one assumes that $A_0, S_1, \ldots, S_n$ have stationary increments. One then controls the system through the choice of the time allocations $T_k(t)$. One looks for a policy, a time homogeneous choice of $T_k(t)$, which will optimize the long term average (or the steady state) system performance, e.g. by minimizing
the long term average holding costs \( \liminf \frac{1}{T} \int_0^T w'Q(t)dt \) or long term average throughput \( \liminf \frac{1}{T} D_K(t) \).

This is a very hard problem. If all processing times are exponential we can formulate it as a countable state space MDP. This cannot be solved, but if the state space is truncated it can be solved approximately, by well known algorithms. However the high dimensionality of the state space makes this still quite impractical. We shall talk a little about the diffusion approximation approach to solve it. In any case the problem is too large.

However, optimizing the steady state of the system may not be what we want to do. My perception of a fab is that it is always changing. During the 6 weeks period that the initial WIP will spend in the system, some machines may be upgraded, some of the products may be modified, some scheduled maintenance will take place, and the product mix may change. In particular, a new product will go through some initial production period before the process is mature, or just a few steps in the process may be ramping up following modifications. This means that within a reasonable horizon the process does not remain homogeneous. It certainly does not have time to reach its steady state. This makes the search for optimal steady state behavior somewhat questionable. Moreover, almost at any time you come into a plant they are coping with some crisis, and it is very important for management that the control policies of the plant will pay special attention to the present state of the system. This is again incompatible with long term optimization.

### 1.2.3 Finite Horizon On-Line Formulation

For the reasons explained above we choose to optimize the system over a finite time horizon. This means that we will want to consider the \( N_0 \) wafers which are the initial WIP (distributed over the fab) and all the wafers which come into the system over the time horizon \( T \), and to build a schedule for the period \((0, T)\). We envisage this to work for \( T \) of the order of \( N \), say by the time \( T \) we will have \( cN_0 \) departures, where \( c \) is a constant close to 1 (e.g. \( 1/2 < c < 3 \)). In this we take the scheduling rather than the queueing approach. However, we do not want to try and solve the detailed scheduling problem, instead we will ignore detailed knowledge of the processing times etc., and use an on-line policy. What we mean by on-line policy is that at any time we will use knowledge about average arrival and processing rates (could be time varying), and we will make our scheduling and control decision at time \( t \) based only on data at time \( t \).

The data at time \( t \) consists of the queue length vector \( Q(t) \), and the cumulative departures up to time \( t \), \( D(t) \). Lest we are tempted to do some sophisticated estimation and prediction, we will not even store the values of \( Q(s), D(s), s < t \) for use in our decision at time \( t \).

The resulting problem is: Find a rule to allocate machines to jobs, based at time \( t \) on the state of the system \( Q(t), D(t) \), (which is only a partial state description, so in general we do not have a Markov decision problem) over the period \((0, T)\), so as to minimize the cost function \( V(Q(\cdot), D(\cdot)) \).

This problem is at least as hard as the problem of finding optimal schedule for a finite batch with complete data using combinatorial optimization, or the MDP problem of optimizing the system steady state.

However, we can solve it approximately by using fluid approximation. We describe now
1.3 A 3 Step Procedure

1.3.1 Fluid Formulation

We approximate this problem by a fluid approximation: We replace the arrival stream \( A_1(t) \) by \( \alpha t \), the cumulative service completions \( S_k(t) \) by \( \mu_k t/m_k \). We let \( u_k(t) \geq 0 \) be a measurable function expressing the rate of flow out of buffer \( k \), so that \( \int_0^t m_k u_k(s) ds = T_k(t) \) expresses the ‘processing time’ devoted by machine \( i = \sigma(k) \) to fluid buffer \( k \).

Our fluid problem is then to choose controls \( u_k(t) \) such that

\[
q_1(t) = q_1(0) + \alpha t - \int_0^t u_1(s) ds, \\
q_k(t) = q_k(0) + \int_0^t (u_{k-1}(s) - u_k(s)) ds, \quad k = 2, \ldots, K \\
\sum_{k \in C_i} m_k u_k(t) \leq 1, \quad i = 1, \ldots, I \\
u(t), q(t) \geq 0, \quad t \in (0, T).
\]

We consider mainly two types of objectives, minimizing the makespan, time until all jobs are completed, and minimizing weighted flow time, weighted by the buffer weight:

\[
\min \int_0^\infty 1\{\sum_1^K q_k(t) \neq 0\} dt \\
\min \int_0^T \sum_1^K w_k q_k(t) dt
\]

How is the fluid related to the original problem? We can go from the original problem to the fluid via the queueing way, or via the scheduling way.

In the queueing way we take \( A(t), S(t), T(t), Q(t) \) and \( D(t) \) and rescale them on the fluid scale: \( \frac{1}{n} A(nt), \frac{1}{n} S(nt), \frac{1}{n} T(nt), \frac{1}{n} Q(nt), \frac{1}{n} D(nt) \) and we let \( n \to \infty \). We get the fluid limits \( a(t) = \alpha t, s(t) = \mu t, T(t) = \int_0^t u(s) ds, q(t) \) and \( d(t) \). All of these have initial values 0 at \( t = 0 \) except for \( Q \). However the scaling will make \( q(0) = 0 \), which may often lead to \( q(t) = 0 \), which is uninteresting. Therefore we conceptually go through a sequence of systems, having different initial buffer contents, so that \( \frac{1}{n} Q^{(n)}(0) \to q(0) \).

The scheduling way of reaching the fluid limit involves relaxation of the discrete integrality restrictions, in three stages:

**Integrality of buffers** Let \( q(0) = Q(0) \) but relax the requirement that buffer contents \( Q_k(t) \) are integer.

**Integrality of machines** Relax the requirement that a machine can only work on one job at a time, let the machine capacity be infinitely divisible, so it can be split, with the constraints: \( 0 \leq \dot{T}_k(t), \sum_{k \in C_i} \dot{T}_k(t) \leq 1 \).
**Integrality of jobs** In the discrete scheduling formulation a job moves as a whole from buffer to buffer. We now relax this requirement, so that a job in buffer \(k-1\) is squeezed piecemeal into buffer \(k\): The fraction of the job which has been processed in buffer \(k-1\) departs from \(k-1\) continuously and arrives in \(k\), and it can be processed there immediately.

Both approaches make the contents of the buffers fluid, their dynamics are now deterministic and continuous (governed by linear dynamics, with the parameters \(\alpha, \mu\)), and the controls \(T\) are now expressed as \(T_k(t) = \int_0^t v_k(s)ds\), where \(v_k(s)\) are rates of machine time allocation, giving the split of machine capacity between the buffers. In much of the literature (Harrison, Meyn, Dai, Bramson?) one works directly with \(v\). We will use \(u_k(t) = \mu_k v_k(t)\) which are the flow rates out of buffer \(k\) at time \(t\).

Since the fluid problem is obtained from the finite horizon scheduling problem by relaxing all the integral constraints, it seems that it should indeed be a relaxation. Hence, we should have that the actual optimal solution is bounded below by the optimal fluid solution:

\[
V_{\text{Fluid}} \leq V_{\text{Opt}}.
\]  

This is indeed the case for the makespan problem. However, as we relax the integrality and so relax the constraints, we also lose information: e.g. in determining the optimal \(u_k(t)\) we can no longer use the detailed processing time information \(X_k(j)\) of individual jobs — they are all mixed up in the fluid. As a result we have less control over our schedules, and the optimal fluid solution may be much inferior to the optimal scheduling solution.

However, we do not wish to compare the fluid solution with the optimal schedule. We wish to compare it with the optimal ‘on-line’ schedule, which uses only \(Q(t), D(t)\) at time \(t\). In general we do not know whether the fluid solution is a lower bound on the optimal ‘on-line’ cost. We know it is for makespan, and conjecture it is for weighted flowtime at least asymptotically as \(N_0\) becomes large. The general treatment of this is an open research question.

### 1.3.2 Fluid Solution

The above fluid problem (with weighted flowtime objective) is a special case of the following Separated Continuous Linear Program (SCLP) with linear data:

\[
\text{max } \int_0^T (T - t)c'u(t)dt \\
\text{SCLP s.t. } \int_0^t Gu(s)ds + x(t) = a + \alpha t, \\
Hu(t) = b, \\
x(t), u(t) \geq 0, \quad t \in [0, T].
\]

Here \(G\) is a fixed \(K \times J\) matrix, \(H\) a fixed \(I \times J\) matrix \((J > I)\), and the remaining problem data consists of the \(K\)-vectors \(a, \alpha, I\)-vector \(b\), and \(J\)-vector \(c\). Equation (1.3) expresses the dynamic relation between the unknown state vector \(x(t)\) and the unknown control vector \(u(t)\), with initial state \(a\) and exogenous input rate \(\alpha\). The time derivative or rate of change of the
state is $\xi(t) \overset{D}= \dot{x}(t) = \alpha - Gu(t)$. Equation (1.4) expresses resource constraints on $u(t)$ with resource limits $b$. In the objective (1.2) control values $u(t)$ at time $t$ contribute profit over the remaining time to the horizon $T - t$, at rates $c$.

Here $k, j, i$ count buffers, flows and machines, with $x_k(t)$ the fluid levels and $u_j(t)$ the flow rates. Alternatively one can use SCLP to model an economic system. In that case $k, j, i$ count assets, activities, resources respectively, and $x_k(t)$ are the asset levels, $u_j(t)$ are activity levels, and $b$ the resource limits.

Efficient solution of CLP or SCLP has been an open problem for close to 50 years. We will present here a novel simplex type algorithm to solve it, which we hope to be effective. Many research issues remain to be investigated here. To name a few: We do not have a systematic way to handle degeneracy. We wish to extend the algorithm to general linear objective $(\gamma + (T - t)c)'u(t)$, to piecewise constant $a, \gamma, \alpha, b, c, G, H$, and to non-linear objective functions.

![Figure 2: Cumulative buffer levels in the solution of a re-entrant line](image)

Perhaps the biggest advantage of the optimal fluid solution of SCLP is that it is extremely informative. Apart from the usual sensitivity analysis and shadow costs which LP solutions present, we now have also sensitivity and shadow costs of changes in the timing of events. A particularly useful tool is the graphic presentation of the fluid solution, as seen in Figure 2. Here we consider the fluid problem of a 5 machine 20 buffer re-entrant line, with given initial fluid levels, varying holding costs rates in the different buffers, and with small input rates, which is operated until the system is empty. The optimal fluid solution of this problem empties the system by time 15000, and it uses piecewise constant flow rates, which partition the total time into 35 intervals. In the plot we put the fluid levels of buffers $k = 1, \ldots, K$ one above the other as a function of time. Colors can be used to indicate the machines which
process the buffers. If we think of a point on the surface of the graph, at height \( x \) and time \( t \), as a part in production, we can see by looking at the graph vertically for fixed \( t \) and for all \( x \) what parts are in the system and in which buffers they are, and we can see by looking at the graph horizontally for fixed \( x \) the path of a part through the system.

This information gives an overall view of the projected dynamics of the state of the plant, and is obviously very useful to a plant manager. Changes in data can be used to recalculate this view, as an easy alternative to simulation. Also, detailed sensitivity analysis can be carried out for each piece of the fluid problem data.

The picture can be used to control the plant, by giving each machine foreman the flow rates indicated in the fluid solution as his production time line goals for the planning period.

Furthermore, we can zoom in on this picture and project the time line evolution of individual orders. In that case this will give us an alternative to MRP.

### 1.3.3 Fluid Imitation

Given the fluid solution, we now need to translate it into a schedule. One particularly simple procedure to do so is the following:

Consider the fluid solution, with fluid levels \( q_k(t) \) and with cumulative fluid departures \( d_k(t) \). These functions are calculated for for the period \((0, T)\) by solving the fluid problem, and are available to the scheduler from the outset, at time 0. Consider now the actual queue length and cumulative departures \( Q_k(\cdot), D_k(\cdot) \). These start at the known initial levels \( Q_k(0) = q_k(0), D_k(0) = d_k(0) = 0 \), and at time \( t \) reach the level (known at \( t \)) of \( Q_k(t), D_k(t) \). If machine \( i \) becomes available at time \( t \), assign it to work on the first job (head of the line) at buffer \( k^*_i \), where:

\[
k^*_i = \arg \max \{ d_k(t) - D_k(t) : k \in C_i, Q_k(t) > 0 \}.
\]

It is very much an open problem how efficient this procedure is, and there may well be a need for other, improved, procedures for on-line scheduling using the fluid solution. We do however already have several indications to the efficacy of this method.

Several papers have found bounds on the difference between the fluid makespan and the makespan of this procedure. Furthermore, extensive experimentation via simulation supported this claim. Also, the fluid solutions which we obtain from SCLP are fluid solutions which correspond to the fluid scaling of a stochastic multiclass queueing network with some infinite buffers. We have several indications that such stochastic systems can be run with full utilization and yet exhibit stable queues in all the buffers which are not infinite. This suggests that these fluid imitation algorithms may indeed be asymptotically optimal for a wide range of problems.

### 1.4 Implementation at the Wafer Fab

Current procedures at the wafer fab include some conceptual approaches such as JIT, conwip, etc which were developed for production lines such as car manufacturing plants, modeled by a flow shop.

Also, every plant has MRP (Materials Requirement Planning and Manufacturing Resources Planning) in place. This works back from the due date of each production order to calculate the
release times and planned work times for the parts. However, MRP is basically an accounting
device, which uses only information about each order, and which does not try to predict the
state of the plant at the future time.

One of the possible uses to our approach is to provide an improved MRP.

A major tool for running a fab is a simulation software implemented for the fab. The two
major competitors in this market are the following:

AutoSimulations, Inc. (of Bountiful, Utah) was established in 1982 by founders with experi-
ence in automated material handling engineering and software development. AutoSim-
ulations’ products include AutoMod (the dominant factory simulation software package
in the semiconductor industry) and AutoSched (a factory scheduling software package).

Systems Modeling Corporation (of Sewickley, Pennsylvania) has also been in business for more
than 18 years, and its products include Arena (the second ranked simulation software
package), OptQuest for Arena (a tool that optimizes simulation scenarios in Arena), and
Tempo (a production scheduling software package that incorporates simulation-based
scheduling).

As can be seen in these two examples, a major trend in this market is the merging of factory
simulation software and factory scheduling software.

Other companies that have sold production scheduling software packages over the past five
years include the following (along with their products):

• AT&T Istel - PROVISA
• Chesapeake Decision Sciences, Inc. - MIMI (The Manager for Interactive Modeling
  Interfaces)
• Consilium - Lot Scheduling Module (LSM) (Part of WorkStream)
• DMK Consultants, Inc. - SHIVA
• Flow Logics, Inc. - LMS
• The Goal System Group - The Goal System
• Kiran & Associates - Mixed Model Scheduler (MMS)
• Nimble Corporation - MaTISSE
• Orissa International - Resonance
• Paragon Management Systems, Inc. - Pacemaker
• Pritsker Corporation - Factor
• Promis Systems Corporation - MADEMA
• TYECIN Systems, Inc. - OnTime
• Waterloo Manufacturing Software - TACTIC The Schedulers Assistant

These simulation packages can be supplemented or even replaced for some purposes by our fluid solution.