A Program for Simulation of Semiconductor Wafer Fabrication

Mauricio G.C. Resende

Operations Research Center
University of California
Berkeley, CA 94720

December 1985

Abstract— A program for discrete event simulation of a semiconductor wafer fab is presented. The program is designed to serve as a tool in the investigation of efficient fab shop floor scheduling disciplines, but can also be used to analyze issues such as fab layout design, capacity analysis, production forecasts, and fab start-up strategies. The semiconductor fab scheduling problem is presented and the job shop model used to represent the problem described. The C implementation of the model is discussed. Input and output of the model are described by presenting several test simulations. Future extensions to the model and research directions are proposed.

Key Words— Production Planning, Scheduling, Semiconductor Fabrication, Simulation.

Introduction

This paper describes a C implementation of a simulation model of a semiconductor wafer fab. The program is designed for use in research of fab shop scheduling schemes, but can also be applied to problems such as fab layout design, capacity analysis, production forecasts, and fab start-up strategy analysis. Other authors have developed simulation models for semiconductor fab analysis [Da84, Lo84], but those models are not intended for studying scheduling dispatching schemes and lack some important capabilities.

The semiconductor fab is first described and the fab scheduling problem defined. The simulation model is then presented and its implementation in C described. Several simulation runs are executed on the code. Extensions to the model and code are proposed.

The Semiconductor Fab

The manufacturing of integrated circuits is perhaps one of the most complex manufacturing processes in existence today. This complexity is the result of process intricacy, product diversity, random elements, and evolving technologies that contribute to varying products and product recipes. We next briefly describe the dynamics of a semiconductor fab and state the fab scheduling problem.

For an introductory discussion of integrated circuit manufacturing see [O177]. Semiconductor devices are three-dimensional structures etched on silicon wafers by chemical and physical processes. The manufacturing process consists of four phases: fabrication, sort, assembly, and test. In fabrication, semiconductor devices are constructed on silicon wafers. During sort, individual finished circuit chips are sorted according to quality. After defective chips are discarded, the remaining circuits are packaged in protective plastic shells during the assembly phase. Finally, in the test phase, circuits are tested, further classified, and shipped. In this paper we consider only the fabrication phase of manufacturing.

show

Could show a picture of each phase & Equipment box

The fabrication process begins with lots of polished, millimeter-thin silicon wafers. Fabrication is carried out in a *clean room environment* since even the smallest dust particle can ruin the minute circuits found on a chip. Wafers follow ~~a~~ prescribed ~~precise~~ sequence of process steps, which transform a blank wafer into a finished batch of semiconductor devices. Process sequences are cyclic, with each cycle beginning with the wafer visiting a masking station, where a layer of the three-dimensional structure is defined by a photolithographic process. In each cycle the wafer will visit other, perhaps different, stations where processes such as chemical or physical etching of the structure takes place or where the silicon is doped with charged impurities. Devices may have more than ten masking layers, with

Get cycle picture

what [illegible] [illegible]

each cycle corresponding to a single layer of the device. Often wafers will require over ten weeks in the fab to complete the hundreds of steps in its process sequence.

The semiconductor fab can be viewed as a general job shop with a hub station. A *general job shop* is a production shop consisting of at least two stations. A *station* is a set of identical machines with a single, perhaps empty, queue or buffer of lots waiting for an available machine at the station. A job can enter the shop at any station and is allowed to leave from any station. Jobs have a sequence of operations to be performed on them by a subset of shop stations. This sequence is termed a *recipe*. The recipe may require a job to return to a station more than once. *Cyclic recipes* are recipes that require jobs to return to a fixed station, called the *hub station*, periodically. Exhibit 1 illustrates a cyclic recipe, *(in, A, B, C, A, E, F, G, H, I, J, K, A, out)*.

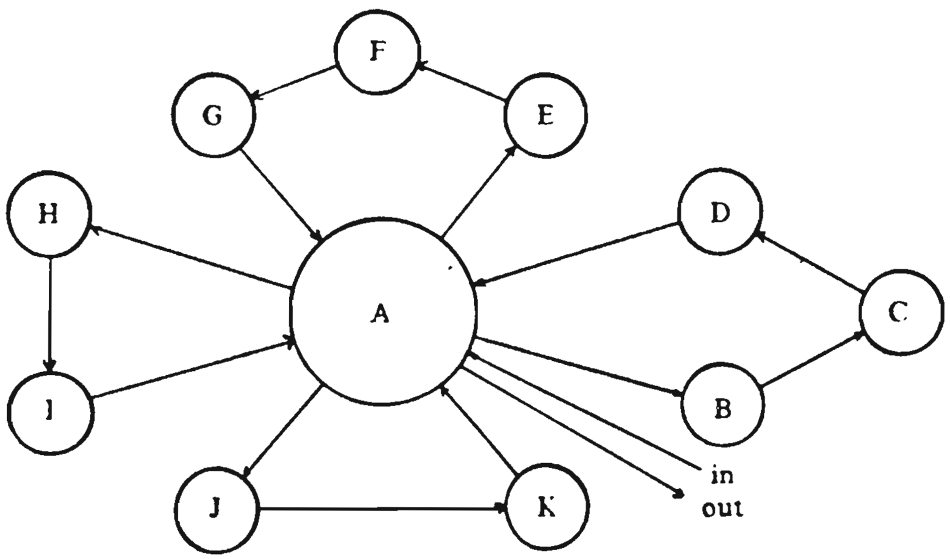

Exhibit 1 - Cyclic Recipe

The general job shop has several stochastic elements. Machines fail at random, requiring a random amount of time to be repaired. A job may require rework at a given machine. Random setup times are sometimes required at a station for a given sequence of jobs going through the station. Defects are introduced at random on the devices causing random yields.

In industry lots are started at a fixed rate for each product. When a machine becomes idle a lot must be selected for processing from the set of lots currently queued at the station queue of that machine. The selection scheme most used is one where the lot with the highest priority is selected for processing. Priorities are given to lots by a dispatching priority rule. Examples of such rules are: FIFO - select the lot that first arrived at the queue; SIPT - select the lot with the shortest processing time in that station; SRPT - select the lot with the shortest remaining processing time in the shop; NINQ - select the lot whose next station in its recipe has the smallest number of lots queued; and RANDOM - select the next job for processing at random from the queued jobs. Panwalker and Iskander [Pa77] classify over one hundred dispatching rules. The objective of the fab scheduler is to determine when lots should begin processing in the fab and which lot should be dispatched to a machine that has just become available, such that some measure of performance is optimized. Performance measures include minimization of expected flow time (cycle time), minimization of work-in-progress, minimization of maximum flowtime, and minimization of idle time for a bottleneck machine.

The general job shop scheduling problem is too complex for exact solution even when there are no random elements present. Attempts to solve the problem exactly have not succeeded, e.g. [Ma60, Gi60, Ba69, Gr68, La77]. In fact, a 10-job, 10-machine job shop problem proposed in 1963 has not yet been solved to optimality [Le84]. It has been shown [Le77] that for the case with two stations and recipes of three steps or more and the case of two step recipes on three or more stations the job shop scheduling problem is NP-complete [Ga79], even for the case where randomness is ruled out. Consequently, there is little hope that an efficient algorithm for job shop scheduling will ever be found. To solve real-world problems, heuristics that obtain good but not necessarily optimal solutions must be used. Several alternative approaches have been proposed in the literature. These include experimentation with real job shops [El63], simulation [Da70, Da84, Ge66, Lo84, Sc84, Ta80, Co85], perturbation analysis [Ho84, Su84], artificial intelligence [Bu80], and perti nets [No80, Du84, Fa85].

Get Grossman definition of simulation

Since it is not desirable to test possibly poor scheduling algorithms on a real fab, and certain situations, such as multiple machine shut-downs, for which one wants to experiment with are not always available, a tool for simulating the dynamics of a fab is required. A discrete event simulation model is such a tool.

The Model

In this section we discuss the model used to describe the fab in the discrete event simulation program.

The fab is modeled as a network of queues, where each network node corresponds to a fab workstation. A *workstation* is a group of identical pieces of processing equipment. Equipment are not reliable and fail periodically. Both failure and repair times of equipment are modeled as exponentially distributed random variables. Equipment have three possible states: *BUSY* (up and processing), *IDLE* (up and not processing), and *DOWN*.

— Maintenance time? very big. Known time

A fab produces, in general, more than one product. For each product, a process recipe, a lot start rate, and a hot-lot ratio are specified. A *recipe* is a sequence of workstation visits, where the following parameters are fixed for each visit: processing time, rework probability, and yield. The *lot start rate* for a product is defined to be the number of lots of that product that are started per unit of time. A *hot-lot* is a lot that has priority over all other lots in a queue. The *hot-lot ratio* is the number of high priority lots of a given product started over the total number of lots of that product started.

- Sometimes more than one process [illegible]

Lots and batches

Devices are produced on wafers and are processed in lots, usually of 25 to 50 wafers each. For modeling purposes we aggregate production into lots, and thus do not consider wafers as an entity. A lot will follow its product recipe throughout the fab.

? Need to discuss this as an option or something

Upon entering the fab a lot is placed in the queue in front of the first workstation on its recipe. When a processing equipment becomes *IDLE*, two cases can occur. If the queue of lots in front of its workstation is not empty a lot from that queue is selected, by a dispatching rule, for processing. If there is one or more hot-lots in the queue a hot-lot will be chosen. The selected lot is taken off the queue and placed on the equipment for processing, which then goes into the *BUSY* state. If the queue is empty, the equipment remains in the *idle* state until it either goes *DOWN*, or is given a lot for processing, becoming *BUSY*. Equipment in the *DOWN* state cannot process lots. We assume equipment can only go to the *DOWN* from the *IDLE* state. Transitions from *BUSY* to *DOWN* are not allowed. This assumption is also made in [Da84]. Upon termination of processing on a piece of equipment three cases may occur. First, the lot may require rework. In this case the lot is replaced on the equipment and processing restarts. In the second case, the lot is successfully processed and requires further processing in the fab. The lot is placed in the queue of the next workstation on its recipe. Finally, when the lot has no more steps on its recipe for processing, it is removed from the fab.

[illegible] ?

A C Programming Language Implementation

The fab model described in the section is implemented as a discrete event simulator [La82]. The code is written in C and is currently running experimentally on a VAX 11/750 minicomputer under BSD 4.2 Unix, and on an IBM 3081 mainframe running VM/SP CMS. The code is written in a way to facilitate porting to an IBM Personal Computer. In this section the data structures and logic of the code are briefly described.

The program is designed to work as a list processor. All entities in the program are represented as linked lists. Below is a brief description of the lists found in the program.

> *product_recipe:* A linked list whose elements contain information about a single step of a product recipe. This includes step duration, step number, workstation number, rework probability, yield, and a pointer to its corresponding worksta-

tion element in the list of workstations.

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, and a pointer to its current position in the product recipe list.

work_station: A linked list whose elements contain information about a single workstation 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, and a pointer to the first element of the list of equipment of the workstation.

queue: A linked list representing a queue in a workstation. 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 workstation. 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 downtime, mean uptime, a pointer to the first element of the list of lots currently being processed by it, and a pointer to the element in the workstation list that corresponds to its workstation.

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.

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 [St80] and time is incremented to the epoch of the next scheduled event. The main program in C is presented in exhibit II.

```
main()
{
        set_up();
        update_times();

        while (time <= horizon)
        {
                lot_start_check();
                load_queue_check();
                equip_status_check();
                load_equip_check();
                update_times();
        }
        output_report();
}
```

Exhibit II - Main Program

The set_up() routine sets up the program for running the simulation. In it the lists are initialized, the problem data is input, and the initial simulation events are scheduled. Input includes descriptions of the fab ~~and~~ plus the products, and a set of simulation parameters. The description of the fab is given by a set of workstations, each with workstation number, number of equipment in workstation, and equipment load size; and a list of equipment in each workstation, with mean downtime and uptime estimates for each equipment. 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, the proportion of lots of that product that are hot, 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, a step duration in minutes, the rework probability, and the step yield. The set of simulation parameters include seeds for the random number generator, the simulation horizon, and an indication of the dispatching rule to be used. At present the following rules are available: first-in-first-out, shortest imminent operation time, shortest remaining operation time, and

Long sentences
(May be a
list
o
o
o
o
o

shortest queue size of next step. Finally, variable *time* is set to next event time.

The update_times() routine finds the next event in the heap of scheduled events and updates *time* 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.

? Flowchart,

The *while loop* is run while *time* is less than or equal to the simulation *horizon*. In the loop the program first checks, in lot_start_check(), if a new lot is ready to be started. If so, a new lot is defined, loaded into the queue of the workstation of its first recipe step, and a new lot start for that product is scheduled. In load_queue_check() the equipment lists are scanned and if a lot is finished processing, it is placed in the queue of the next workstation 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. In equip_status_check() all workstations 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. In load_equip_check() all workstations are checked for an *IDLE* equipment and for a number of waiting lots equal to the load size of the equipment in the workstation. If such a situation is identified, lots in the queue are prioritized and one or more lots are selected for loading into the equipment. These lots are removed from the workstation queue list and put into the equipment in_process list. An *end processing* event is scheduled and placed in the heap. The iteration ends with an update_times() where the next event is retrieved from the event heap and *time* and count-down times updated.

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

Example Simulations

In this section we run several simulations on a hypothetical fab.

The fab has the hub workstation characteristic. There are 14 workstations. Each workstation has one or more identical pieces of processing equipment that we consider to have the same mean up and down times (the model allows them to be different). We also assume that equipment load size is a single lot of wafers (the model allows the load size to be of any size). Exhibit III gives a description of the fab.

Workstation	Equipment Copies	Mean Uptime (mins.)	Mean Downtime (mins.)
1	6	20000	1800
2	2	10000	1000
3	2	10000	1000
4	2	10000	1000
5	2	10000	1000
6	1	50000	200
7	1	50000	200
8	1	50000	200
9	1	50000	200
10	1	50000	200
11	1	50000	200
12	1	50000	200
13	1	50000	200
14	1	50000	200

Exhibit III - Fab Description

The fab is assumed to process one product (the model allows several) with a 27 step recipe. The total processing time for a lot is 2200 minutes. Two sets of simulation runs are made. In the first set of runs the wafer start rate is set at one lot every 300 minutes and set the hot-lot ratio to 0.1. All rework probabilities are set to 0.01 and yield is 0.99 for every step. In the second set of runs, the fab is loaded more heavily by increasing the start rate to one lot every 230 minutes. Exhibit IV describes the process recipe used in the simulations. Notice that workstation 1 is the hub station of the fab.

Step Number	Workstation	Duration (mins.)
1	8	60
2	4	30
3	9	70
4	4	40
5	1	275
6	2	50
7	10	70
8	2	30
9	11	60
10	2	30
11	3	40
12	1	185
13	3	50
14	12	80
15	13	60
16	3	30
17	14	90
18	3	50
19	1	180
20	5	40
21	6	60
22	5	25
23	1	300
24	4	20
25	1	180
26	5	35
27	7	60

Exhibit IV - Product Recipe

The simulations were run on the IBM 3081 at Berkeley. The code is compiled on the Waterloo C compiler using the object code optimization option. Simulations are run with a horizon of 120,000 minutes (approximately 12 seven-day weeks with 24-hour days) with some 400 lot starts per run in the first set of runs and 520 in the second. Over 10,000 events occur in each simulation. A total of 16 sets of four runs are made, ten with the 1/300 start rate and six with the 1/230 rate. Each set has a different random number generator seed set. In each set a run is made for each dispatching rule presently available on the system: first-in-first-out (FIFO), shortest imminent processing time (SIPT), shortest remaining processing time (SRPT) and shortest queue at next step (NINQ). The mean CPU time for a run was 7.15 secs. Mean cycle time is measured. Exhibits Va and Vb summarizes the simulation runs.

Mean Cycle Time (mins.)				
Set Run	FIFO	SIPT	SRPT	NINQ
1	2721.05	2573.32	3469.93	2548.61
2	3462.93	4001.55	3566.76	3366.96
3	2917.47	3526.37	3010.48	2871.03
4	3305.86	4057.36	3995.62	4231.36
5	2619.72	2586.17	2586.72	2621.54
6	2802.16	2692.58	2484.36	3424.23
7	2986.17	3609.87	3039.45	2397.04
8	2834.31	2945.79	3247.47	2675.61
9	4446.57	2572.58	2920.71	3648.28
10	2917.47	3526.37	3010.48	2871.03

Exhibit Va - Example Simulation Runs (1/300 Start Rate)

Mean Cycle Time (mins.)				
Set Run	FIFO	SIPT	SRPT	NINQ
1	5485.09	5531.88	4735.08	3276.47
2	3826.74	4517.70	4364.94	3993.38
3	9268.26	9897.44	8393.65	12102.80
4	6046.31	5256.61	3588.10	3865.71
5	3394.40	4278.79	3631.75	3170.83
6	5961.68	3819.50	4486.09	4166.47

Exhibit Vb - Example Simulation Runs (1/230 Start Rate)

The means of the mean cycle times for the first set of runs were 3089.54, 3124.37, 3081.76, and 3089.02 for FIFO, SIPT, SRPT, and NINQ, respectively. For the second set of runs the means of the mean cycle time were respectively 5663.75, 5550.32, 4866.60, and 5095.94, for FIFO, SIPT, SRPT, and NINQ. Besides computing the above means, we make no further attempt to analyze the results of the simulations but rather intend these examples to serve as an illustration of the use of the code.

— Give some graphs of results

Future Directions

The objective of the code described in this paper is to aid us in the investigation of effective fab shop floor dispatching rules. With this in mind the next phase in its development can be divided into two main efforts. The first is the validation of the model. We must determine if the model well describes the behavior of a semiconductor fab. If not, extensions must be developed to make it a good approximation of a fab. This may require the introduction of set-up times, fab personnel, factors inter-equipment transfer mechanisms, statistical distribution of yield, etc. The second effort is the inclusion of more dispatching rules in the code. This should also include more complex rules, such as those that are combinations of simple rules [Bu85], heuristics similar to those in [Ge66], and rules that differentiate between stations being dispatched.

good vs bad operators

More data can still be captured from the simulation. Statistical requirements should be specified so that the desired data set can be collected. Finally, a concise output report must be designed.

Acknowledgement

This research was partially funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, Brazil, Project MICRO (Microelectronics Innovations and Computer Research Opportunities) with the University of California, the Semiconductor Research Corporation, Philips Research Laboratory, and Fairchild Semiconductor Camera & Instrument Corporation Research Center. The author greatly appreciates the support and guidance given by Professors C.R. Glassey and D.A. Hodges at UC-Berkeley.

References

[Ba69] Balas. E.. "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm". Operations Research. vol. 17. no. 6. 1969

[Bu80] Bullers. W.I.. Nof. S.Y.. Whinston. A.B.. "Artificial Intelligence in Manufacturing Planning and Control". AIIE Transactions. vol. 12. no. 4. 1980

[Bu85] Bunnag. P.. Smith. S.B.. "A Multifactor Priority Rule for Jobshop Scheduling Using Computer Search". IEE Transactions. vol. 17. no. 2. 1985

[Co85] Coll. A.. Brennan. L.. Browne. J.. "Digital Simulation of Production Systems". in *Modelling Production Management Systems*. P. Falster. R.B. Mazumder. eds.. Elsevier Science Publications B.V.. 1985

[Da70] Day. J.E.. Hottenstein. M.P.. "Review of Sequencing Research". Naval Research Logistics Quarterly. vol. 17. 1970

[Da84] Dayhoff. J.E.. Atherton. R.W.. "Simulation of VLSI Manufacturing Areas". VLSI Design. Dec. 1984

[Du84] Dubois. D.. Stecke. K.E.. "Using Petri Nets to Represent Production Processes". Proceedings First ORSA/TIMS Conference on FMS. 1984

[El73] Elvers. D.A.. "Job Shop Dispatching Rules Using Various Delivery Date Setting Criteria". Production Inventory Management. vol. 14. 1973

[Fa85] Favrel. J.. Lee. K.H.. "Modelling. Analyzing. Scheduling. and Control of Flexible Manufacturing Systems by Petri Nets". in *Modelling Production Management Systems*. P. Falster. R.B. Mazumder. eds.. Elsevier Science Publications

B.V., 1985

[Ga79] Garey, M.R., Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979

[Ge66] Gere, W.S., "Heuristics in Job Shop Scheduling", Management Science, vol. 13, no. 3, 1966

[Gi60] Giffler, B., Thompson, G.L., "Algorithms for Solving Production Scheduling Problems", Operations Research, vol. 8, no. 4, 1960

[Ho84] Ho, Y.C., "Perturbation Analysis of Discrete Manufacturing Systems: An Overview", Proceedings First ORSA/TIMS Conference on FMS, 1984

[La77] Lageweg, B.J., Lenstra, J.K., Rinnooy Kan, A.H.G., "Job-Shop Scheduling by Implicit Enumeration", Management Science, vol. 24, no. 4, 1977

[Le77] Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P., "Complexity of Machine Scheduling Problems", Annals of Discrete Mathematics, vol. 1, 1977

[Le84] Lenstra, J.K., Rinnooy Kan, A.H.G., "New Directions in Scheduling Theory", Operations Research Letters, vol. 2, no. 6, 1984

[Lo84] Lohrashbpour, E., Sathaye, S., "Simulation Modeling of IC Wafer Fabrication Lines", Technical Program Proceedings, Semicon/West, San Mateo, CA, May 1984

[Ma60] Manne, A.S., "On the Job-Shop Scheduling Problem", Operations Research, vol. 8, no. 2, 1960

[No80] Nof, S.Y., Whinston, A.B., Bullers, W.I., "Control and Decision Support in

Automatic Manufacturing Systems", AIIE Transactions, vol. 12, no. 2, 1980

[O177] Oldham, W.G., "The Fabrication of Microelectronic Circuits", Scientific American, vol. 237, no. 3, Sept. 1977

[Pa77] Panwalker, S.S., Iskander, W., "A Survey of Dispatching Rules", Operations Research, vol. 25, no. 1, 1977

[Sc84] Schriber, T., Talbot, F.B., "The Use of GPSS/H in Modeling a Typical FMS", Proceedings First ORSA/TIMS Conference on FMS, 1984

[St80] Standish, T.A., *Data Structure Techniques*, Addison-Wesley, 1980, pp. 91-92

[Su84] Suri, R., Dille, J.W., "On-Line Optimization of FMS Using Perturbation Analysis", Proceedings First ORSA/TIMS Conference on FMS, 1984