Norges Teknisk-Naturvitenskapelige Universitet

TPK4186 - Advanced Tools for Performance Engineering
Spring 2023

Assignment 3: Wafer Production Line

Prepared by Antoine Rauzy

Contents

1 Introduction

1.1 Presentation of the Problem

Wafers In electronics, a wafer (also called a slice or substrate) is a thin slice of semiconductor, such as a crystalline silicon (c-Si), used for the fabrication of integrated circuits and, in photovoltaics, to manufacture solar cells, see Figure 1. The wafer serves as the substrate for microelectronic devices built in and upon the wafer. It undergoes many microfabrication processes, such as doping, ion implantation, etching, thin-film deposition of various materials, and photolithographic patterning. Finally, the individual microcircuits are separated by wafer dicing and packaged as an integrated circuit (source Wikipedia).

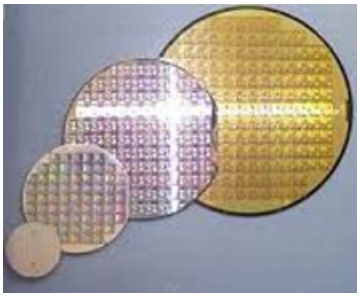

Figure 1: Wafers

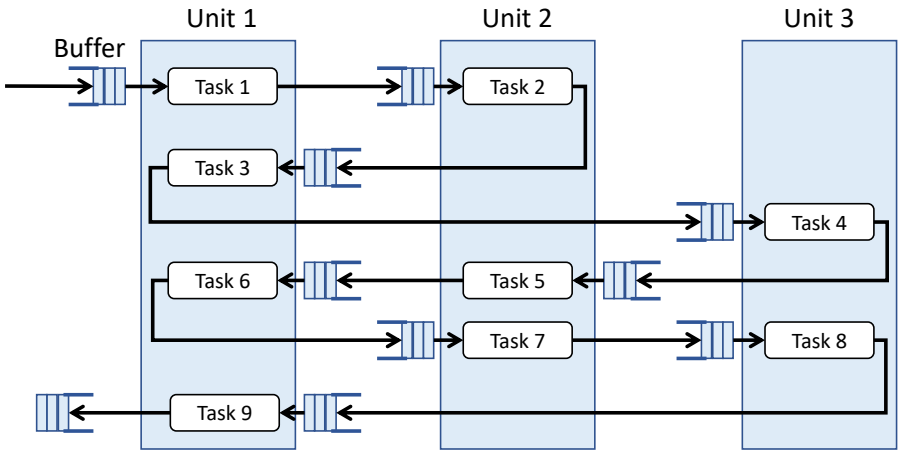

Figure 2: Production line

Table 1: Processing time per wafer for each task

Task	1	2	3	4	5	6	7	8	9
Processing time	0.5	3.5	1.2	3	0.8	0.5	1	1.9	0.3

Wafers can be of different sizes ranging from 25.4mm to 300mm for a thickness of about 0.7mm. The trend is to use wafers as large as possible in order to be able to print out as many microprocessors as possible and to limit the lost on the edge of the wafer.

Production Lines Wafers are produced in white rooms, with a highly purified air, as any impurity on a wafer lead to discard it. The production consists of up to 20 successive highly complex tasks, performed by specialized units. The production process is extremely expensive, hence the need to optimize it as much as possible.

Figure 2 shows a simplified wafer production line.

In consists of three units performing nine tasks. Tasks must be performed in the order indicated on the figure, i.e. from Task 1 to Task 9. A task consists in doing some treatment on of a batch of wafers. Batches are thus sent in the input buffer and then processed by the units.

The constraints are the following.

- A unit can perform only one task at a time.
- Batch sizes can vary from 20 to 50 wafers.
- It takes one minute to load a batch of wafers from an input buffer into a machine, no matter how many wafers this batch contains. It takes also one minute to unload the batch from the machine and to store them in the next buffer, again no matter how many wafers this batch contains.
- Batches must be unloaded from units as soon as they are processed.
- The processing time of a batch of wafers depends on the task and the number of wafers the batch contains. Table 1 gives the processing time per wafer for each task.
- Buffers cannot contain more than 120 wafers, except for the very last buffer, which has an unlimited capacity.

The objective of the assignment is to design a simulator for the production line and to use

the simulator to optimize the production. Namely, you should try to produce 1000 wafers in the shortest time possible.

1.2 Requirements

The objective of this assignment is to show your Pythonic skills.

Here follows a number of requirements.

1. You must provide your program together with a small document explaining how it is organized, what it is doing (which functionalities are implemented) and reporting experiments you have performed with it.
2. The assignment can be made either individually or in group (maximum 3 persons, preferably 2).
3. Assuming you are Jack Sparrow, all the files you deliver must be included in a zip archive named:

   `TPK4186 - 2023 - Assignment 3 - Jack Sparrow.zip`

   The deliverable of the assignment is this zip archive. Please recall your names in addition to the group name both in the header of the program and in the accompanying document.
4. The quality of a program is judged along three criteria: its completeness, its correctness and its maintainability:
   - A program is complete if it provides all functionalities demanded by the client. Some functionalities are however more important than other. You must first concentrate on the main functionalities, then develop the "nice-to-have" ones.
   - A program is correct if it is bug free. To ensure that your program is correct, you must test it extensively. Design tests before writing the first line of code. There is no such a thing than a program or a functionality that works "most of the time". Either it works, or not. If you are not able to make a functionality work, do not deliver it.
   - A program is maintainable if it is well presented, if the identifiers are significant, and so on. But before all, a program is maintainable if it well organized and as modular as possible. Separate the concerns.

2 Tasks

For this assignment you are asked to design several data structures and to implement quite a few functions. Each of these functions must be thoroughly tested.

The assignment consists in the following tasks.

2.1 Production Line

The first step consists in designing (and testing) data structures to implement the production line.

Task 1. Design a data structures to encode and to manage batches, buffers, tasks, units and finally the production line as a whole.

You need for instance to implement methods:

– To assign input and output buffers to each task.

– To assign tasks to machines.

– To load and unload buffers and units with batches.

– To calculate the time taken to process a batch for a given task.

– ...

Note that each unit must implement a method that selects the next batch to treat in one of its input buffers, when the machine is ready to perform an new task. This heuristic plays a central role in the optimization process. As a start, you can just consider the tasks in the order they have been assigned to the machine.

Task 2. Design a printer that makes it possible to observe the production process at work. Use this printer to test the management functions you designed in Task 1..

2.2 Simulator

The simulation process involves three types of actions:

– Load a batch in the input buffer of the line.

– Take a batch in the input buffer of a task, load it into the corresponding unit and start the treatment of the batch.

– Complete the treatment on a batch, unload the unit, and store the batch into the output buffer of the task.

Note that to start a task on a unit, the unit must be in standby and there must be a batch waiting to be processed in the input buffer of the task.

Similarly, to complete a task on batch, the output buffer of this task must contain enough room to store the batch. As it is not possible to keep a processed batch into a unit, this means that before starting a task, one must be sure that the output buffer of this task has enough room to store the batch.

Each of the above actions start at a certain time and is completed after a certain time. The times to load and unload batches from units and to process batches have been described in the introduction.

To implement a simulator of the production line, we need to design a scheduler that stores actions in increasing order of their completion dates. The simulator executes then a loop consisting, until there is no more actions to perform, in the following steps:

– Remove the action from the scheduler.

– Perform the changes in the state of the production line.

– Schedule the actions that become possible in the new state.

To simulate the production of 1000 wafers, we must thus:

– Split these 1000 wafers into batches.

– Load these batches in the input buffer of the production line at predetermined times.

– Run the process described above until all the batches are in the output buffers of the production line (and thus no more action is possible).

Task 3. Design data structures for actions, for the scheduler and for the simulator. Implement the simulation loop.

Task 4. Test your simulator first with one batch, then with few batches, and eventually with enough batches to produce 1000 wafers. Trace these simulations by printing out their actions into text files.

2.3 Optimization

Now comes the real stuff: the optimization of the production process using your simulator. Recall that the objective is to produce 1000 wafers as fast as possible. The parameters you can tune to do so are the following.

- The way the 1000 wafers are grouped into batches.
- The times at which the batches are loaded into the input buffer of the production line.
- The heuristic used by each unit to select the next batch to process.

The problem is indeed that there is a gigantic number of possible setting of these parameters. You cannot test them all. You can however try to perform a few tests so to better understand the problems at stake.

The overall strategy consists in fixing all parameters but one, and studying *mutatis mutandis* what is the best value for the selected parameter. It is important, in the three following tasks, to keep track of all experiments you perform. Your report should contain tables, plots or any other means you think convenient to show the influence of the parameters on the production time.

We shall start with the second point.

Task 5. Design a function to optimize the times at which the batches are loaded into the input buffer of the production line. Start by determining a worst case scenario where one waits that the previous batch is produced to start producing the next one. Then, reduce the interval between two loading.

We can now look at the heuristic used by each unit to select the next batch to process. We can assume that the units consider the input buffers of their tasks in a fixed order. For instance, unit 1 can consider tasks in the order (1, 3, 6, 9), or in the order (1, 3, 9, 6), or in the order (1, 6, 3, 9) and so on.

Task 6. Suggest (and implement) an ordering heuristic. Perform an experimental study to justify your choice.

Finally, we can look at the first parameter, namely the way the 1000 wafers are grouped into batches.

Task 7. Design a function to optimize experimentally the size(s) of the batches.