A SIMULATION-BASED FRAMEWORK FOR INDUSTRIAL AUTOMATED WET-ETCH STATION SCHEDULING PROBLEMS IN THE SEMICONDUCTOR INDUSTRY

[a)]Adrián M. Aguirre, [a)]Vanina G. Cafaro, [a)]Carlos A. Méndez[*], [b)]Pedro M. Castro

a) INTEC (Universidad Nacional del Litoral - CONICET), Güemes 3450, 3000 Santa Fe, Argentina.
b) UMOSE, Laboratório Nacional de Energia e Geologia, 1649-038 Lisboa, Portugal

[*] cmendez@intec.unl.edu.ar

ABSTRACT

This work presents the development and application of an advanced modelling, simulation and optimization-based framework to the efficient operation of the Automated Wet-etch Station (AWS), a critical stage in Semiconductor Manufacturing Systems (SMS).

Lying on the main concepts of the process-interaction approach, principal components and tools available in the *Arena*® simulation software were used to achieve the best representation of this complex and highly-constrained manufacturing system. Furthermore, advanced *Arena* templates were utilized for modelling very specific operation features arising in the process under study.

The major aim of this work is to provide a novel computer-aided tool to systematically improve the dynamic operation of this critical manufacturing station by quickly generating efficient schedules for the shared processing and transportation devices.

Keywords: Discrete-event simulation, Semiconductor Manufacturing System (SMS), Automated Wet-Etch Station (AWS), Arena Software.

1. INTRODUCTION

Semiconductor wafer fabrication is perhaps one of the most complex manufacturing systems in the modern high-tech electronics industry. Wafer facilities typically involve many production stages with several machines, which daily perform hundreds of operations on wafer lots. Moreover, different product mixes, low volume of wafer lots and hot jobs are some of the typical issues arising in this type of system.

Wet-Etching represents an important and complex operation carried out in wafer fabrication processes. In this stage, wafer's lots are automatically transferred across a predefined sequence of chemical and water baths, where deterministic exposure times and stringent storage policies must be guaranteed. Hence, automated material-handling devices, like robots, are used as shared resources for transferring lots between consecutive baths.

An important process restriction is that each robot can only transport a single wafer lot at a time and it cannot hold a wafer lot more than the exact transfer time. Due to the lack of intermediate storage between consecutive baths, this condition can be considered as a non-intermediate storage (NIS) policy in every bath, which must be respected by robots for all transfer movements.

Another constraint adding more complexity to the system operation is that baths must process wafer lots one by one, during a predefined period of time, avoiding the overexposure in the chemical ones, which can seriously damage or contaminate the wafer lot. In spite of this, wafers can stay longer than its processing time only in water baths. So, a zero wait (ZW) and local storage (LS) policy must be strictly satisfied in every chemical and water bath, respectively.

As a direct consequence, an effective schedule of material movement devices and baths along the entire processing sequence will provide a better utilization of critical shared-resources and, at the same time, an important reduction in the total processing time.

In the last years, different methods have been developed to achieve convenient solutions to this challenging problem. Main approaches to large-sized problems lie mainly on heuristic and meta-heuristic methodologies, such as the ones presented by Geiger et al. (1997) and Bhushan and Karimi (2004). In these works, tabu search (TS) and simulated annealing (SA) procedures, together with other different algorithms, were developed to provide a quick and good-quality solution to the job sequence problem and also, a feasible activity program for the robot.

A more recent approach under the concepts of Constraint Programming (CP) was developed by Zeballos, Castro and Méndez, (2011) to handle the sequencing problem of jobs and transfers in the AWS. This method could obtain better results than the ones reported by Bhushan and Karimi (2004) for industrial problem instances in a reasonable CPU time.

To the best of our knowledge, efficient systematic solution methods need to be developed to represent and evaluate the complex dynamic behaviour of the AWS. Thus, a discrete event simulation environment becomes a very attractive tool to analyze the impact of different solution schemes in the system.

In this work, a modelling, simulation and optimization-based tool is developed to validate, test and improve the daily operation of the AWS, allowing an easy evaluation of different operative schemes and possible alternative scenarios. To do this, a discrete event simulation model was developed by using most of the tools and capabilities that are available in the Arena simulation environment. The principal aim is to provide a highly dynamic and systematic methodology to reach the best feasible schedule of limited resources by testing

different measures of effectiveness and performance rates for the system.

Thus, the paper is organized as follows: Section 2 introduces the major features of the problem addressed. Then, Section 3 describes the proposed solution method, highlighting its advantages in comparison with other existing methods and tools as well as the main objectives of this work. Later, the simulation structure is explained in detail in Section 4. A brief description concerning the simulation tool is presented. Software integration and principal interfaces between different tools are discussed. A detailed analysis regarding external and internal logic of the model and the implementation of this solution in a discrete-event simulation environment is also presented.

In Section 5, an alternative solution strategy is tested using several examples, with the main idea of validating the model and, at the same time, comparing results of different solution methods.

Finally, the solutions generated and the comparative study results are reported in Section 6. Conclusions and future work are stated at the end.

2. PROBLEM STATEMENT

The AWS scheduling problem provides a complex interplay between material-handling limitations, processing constraints and stringent mixed intermediate storage (MIS) policies (Figure 1). We can summarize major features of the system in the following way:

-Material-handling devices (robot) can only move one wafer lot at a time. No intermediate storage is allowed between successive baths. So, NIS policy is applied between consecutive baths.

-Waiting times are not allowed during the transportation of a wafer lot.

-Robots and baths are failure-free.

-Setup times are not considered for robots.

-Every bath can only process one wafer lot at a time.

-A ZW storage policy must be ensured in chemical baths whereas LS policy is allowed in water baths.

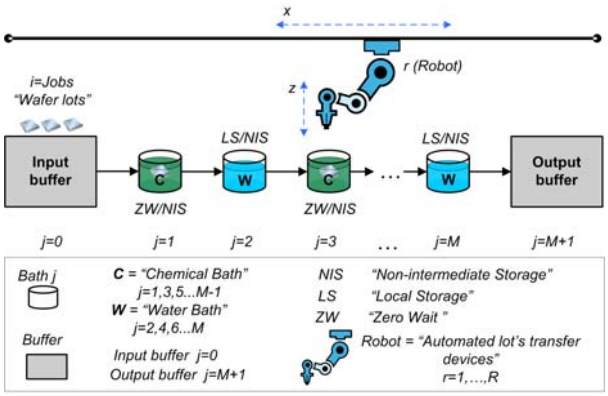


Figure 1: Automated Wet-etch Station (AWS) process scheme.

For this problem, it is assumed that each wafer lot, also called job, *i (i=1,2…N)* has to be processed in every bath *j (j=1,2…M)*, by following a predefined processing sequence. In addition, it considers that a single robot *(r=1)* is available, which has to perform all the transportation activities in the system.

Consequently, the problem to be faced corresponds to the scheduling of *N* jobs in *M* baths, in a serial multiproduct flowshop, with ZW/LS/NIS policies. The use of a single shared robot with finite load capacity for the wafer movement between consecutive baths is explicitly considered in this work.

3. PROPOSED SOLUTION METHODOLOGY

This work introduces an efficient discrete-event simulation framework, which faithfully represents the actual operation of the automated Wet-etch Station (AWS) in the wafer fabrication process.

The main advantage of this computer-aided methodology is that it permits to systematically reproduce a highly complex manufacturing process in an abstract and integrated form, visualizing the dynamic behaviour of its constitutive elements over time (Banks et al. 2004).

The proposed simulation model represents the sequence of successive chemical and water baths, considering the automated transfer of jobs.

Based on a predefined job sequence, which is provided by an optimization-based formulation, the model structure allows the evaluation of many different criteria to generate alternative efficient schedules.

The major aim here is to efficiently synchronize the use of limited processing and transportation resources. This methodology allows also evaluating and improving the operation and reliability of baths and robot schedules. What is more, simulation runs permit addressing industrial-sized problems with low computational effort.

As a result, a basic model is generated to achieve an effective solution to the whole AWS scheduling problem. It becomes also very useful for making and testing alternative decisions to enhance the current process performance.

4. THE SIMULATION-BASED FRAMEWORK

In order to formulate a computer-aided representation to the real-world Automated Wet-Etch Station (AWS) described above, it was decided to make use of the simulation, visualization and analysis tool set provided by the *Arena* discrete-event simulation environment (Law et al., 2007, Kelton et al., 2007).

The simulation model developed in *Arena Software* provides an easy way to represent the AWS by dividing the entire process in specific sub-models (Initializing, Transfer, Process and Output). For each sub-model, the detailed operative rules and strategic decisions involved are modelled using the principal blocks of *Arena Simulation Tool* and, at the same time, a set of visual monitoring objects is used to measure the

utilization performance of all baths and resources in the system.

Additionally, the model allows working with a user-friendly interface with Microsoft Excel for simultaneously reading and writing different data. In next sections, we will describe these features in detail.

4.1. Software integration

The simulator allows an easy communication with Excel spreadsheets. Thus, this tool permits reading, writing and processing important data for the simulation model. Figure 2 illustrates the data flow between Excel and Arena. Both tools support Visual Basic for Applications (VBA) that can be used to move data between them. As shown in the figure, a hybrid solution framework is proposed on these tools. The Mixed integer linear programming (MILP) model provides an initial solution that is written in Excel as input data of the Arena's model. Using that input data, Arena simulation software runs the process model to generate many important statistics that are collected by Excel files as output data. The procedure of reading and writing data is used to dynamically generate a solution schedule by updating the start and finish times of every job in each bath and, simultaneously, determine the status of each job in every stage of the system.

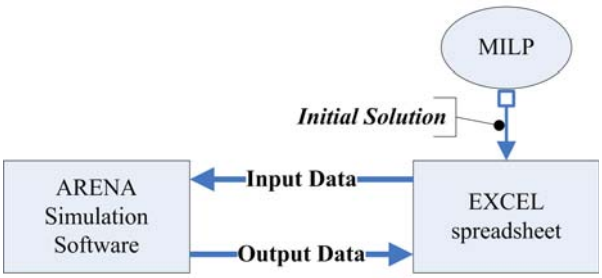


Figure 2: Information exchange between Excel – Arena – MILP Software

4.2. Proposed simulation model

As shown in Figure 3, the entire logic of the simulation model is divided into four main modules (input, transfer, process and output). The first module is the *Initializing* sub-model. The initializing process receives as input data the processing time of each job at each chemical and water bath and a job sequence provided by a MILP model, which is considered as an initial alternative solution. Then, the discrete-event simulation model generates as many entities as wafer lots are to be scheduled. Here, the logic behind the automated transfer of jobs is performed in order to generate a feasible schedule for the robot activities.

The subsequent simulation module is the *Transfer* sub-model, which defines the needed delay time to transfer a wafer lot to the next bath. This module is used to explicitly simulate the time spent to transfer the jobs between the input buffer to the first bath, between successive baths, according to the predefined sequence, and also between the last bath to the output buffer. Only after the transfer is finished, the bath from where the wafer comes is released. It should be noted that a transfer can be only executed if the robot and the destination bath are both available.

In order to simulate the process itself, one *Process* sub-model for each bath is defined. There is a different logic depending on the type of bath (chemical or water). The wafer residence times in chemical baths must be controlled strictly (as soon as chemical bath finishes, the wafer must transferred to the succeeding water bath). While holding time in water baths is allowed. Thus, for every baths, the logic performs the following tasks: (i) reports the time at which the process begins and ends; (ii) seizes the following bath after the delay time finishes; (iii) performs the transfer to the following bath, only if the robot and the destination bath are empty.

It is important to notice that the logic driving in the *Process* sub-model permits to easily identify why and when a given wafer's lot is discarded. Basically, it may occur because the robot and/or next bath are not available. This allows making a detailed analysis about the behaviour of the system, executing, if necessary, the corresponding adjustments when unexpected events occur or when different strategies are tested in the way to improve the process performance. So, *Process* sub-models permit to evaluate and also validate the feasibility of the internal logic algorithm proposed in the *Initializing Process* of the system, identifying the possible causes of infeasibility to be corrected.

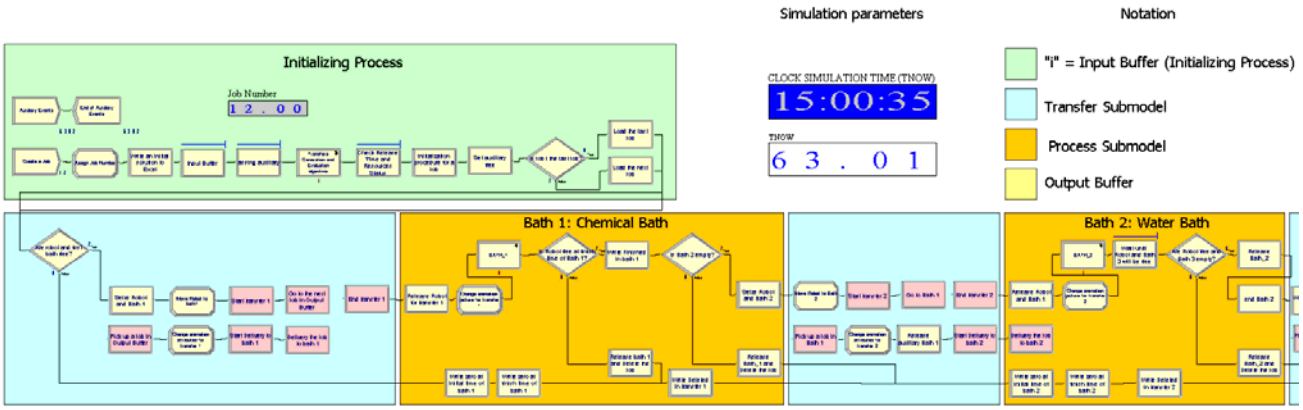


Figure 3: Partial size view of the in-progress simulation model generated in the *Arena* environment.

The last module is the *Output buffer*. The logic of this sub-model represents the final stage of each job. At this module the final processing time (*Makespan*) of each job is reported. It is the ending point for entities created at the input module. Here, the model reports if the current job has been successfully finished or has been discarded.

4.2.1. Advanced internal logic for the robot

The principal aim of modelling the internal robot logic is to explicitly represent the finite capacity of transportation resources for transferring jobs between consecutive baths. The sequence and timing of transfers will depend on the stringent storage restrictions to be satisfied in the baths (ZW / NIS / LS) as well as on the availability of a transportation resource to carry out the transfer.

Since there is only a single robot to do all the job movements, the sequence in which the transfers will be performed needs to be clearly defined. Transfers related to a particular job can never overlap because they are carried out after the corresponding processing stages finish. Consequently, no pair of transfers of the same job may be performed simultaneously.

Therefore, the sequencing problem of transfers must only be focused on the comparison of transfer activities of different jobs in order to determine a feasible robot schedule.

For that reason, a complex internal logic for the robot was embedded in the simulation model to compare and update the start $(ts_{(i,j)})$ and end times $(te_{(i,j)})$ of transfers (i,j). The aim is to define the earliest time at which each transfer can be executed. This logic permits to sequence the different transfers in a correct way, generating a feasible schedule for the robot and a near-optimal solution for the whole system, considering a predefined sequence of jobs.

By using this logic, the transfers related to a given job are sequentially inserted according to the order in which they will be processed at every different bath $(j=1,2,3,\ldots,M+1)$. Then, the transfers are compared successively with all the transfers that were previously inserted into the schedule (according to a predefined processing sequence).

The application of strict storage policies such as ZW and LS in the baths and the NIS rule in the robot significantly complicates the solution of the problem. Enforcing a ZW policy in the chemical baths j implies that the start time of the transfer to the water bath $j+1$ must strictly satisfy equation (1).

$$ts_{(i,j+1)} = ts_{(i,j)} + tp_{(i,j)} + \pi_{(j)} \qquad j = 1,3,5...M-1; \forall i = 1...N \qquad (1)$$

For that reason, the value of $ts_{(i,j)}$ allows directly determining the value of $ts_{(i,j+1)}$. Here $tp_{(i,j)}$ represents the processing time of job i in bath j while $\pi_{(j)}$ denotes the transfer time for every job from bath $j-1$ to j.

On the other hand, if the LS rule is applied to a water bath j, inequality (2) must be satisfied.

$$ts_{(i,j+1)} \geq ts_{(i,j)} + tp_{(i,j)} + \pi_{(j)} \qquad j = 2,4,6...M; \forall i = 1...N \qquad (2)$$

Let $\boldsymbol{p} = \{p_1, p_2, p_3, \ldots p_N\}$ define a permutation processing sequence N different jobs. p_w represents the *w-th* position of a job i $(i=1\ldots N)$ in the processing sequence. It means that the job processed in the *w-th* position will be always before the job processed in the $w+1$ position in the sequence $\boldsymbol{p}$.

Due to the NIS policy in the transfers and constrains on finite load capacity of the baths and the robot, the equation (3) is to be defined.

$$ts_{(w,j)} \geq ts_{(w-1,j+1)} + \pi_{(j+1)} \qquad j = 1,2,3...M+1; \forall w = 1...N \qquad (3)$$

So, any transfer of a job processed in the p_w position, at bath j, has to wait the ending of the transfer of the job located in the p_{w-1} position at the succeeding bath $j+1$ to be processed.

In the next section, we will explain the transfer comparison algorithm developed to solve the described problem. Only one robot is considered to be available for the execution of the transfers in the system.

4.2.2. Generation and evaluation algorithm for transfers

This algorithm is mainly based on the major ideas of the JAT (Job-at-a-time) algorithm, developed by Bhushan and Karimi (2004). The JAT algorithm always prioritizes the transfers related to jobs that were previously inserted in the system, following a predefined processing sequence. For transfers related to the same job, they are executed according to the fixed sequence of baths to be visited $(j=1\ldots M+1)$. So, based on processing constrains (1)-(3) and assuming that all the jobs follow the same processing stages, no job in the p_w position may leave the system before the one located in the p_{w-1} position. This means that all jobs will be processed in the different baths following the same $\boldsymbol{p}$ sequence, what is known as "flowshop permutation schedule".

Our algorithm, as the JAT algorithm, selects a job to be processed and then generates (Generation Process) and evaluates (Evaluation Process) all the transfers for this job, one at a time, before going to the next job of the sequence. The principal difference between the proposed algorithm and the JAT algorithm is the evaluation procedure used for the system transfers.

In the proposed evaluation process, every selected transfer is compared with all the transfers previously inserted into the system. Thus, a detailed schedule of the robot operations is defined.

The aim of this process is to avoid that any transfer previously inserted (w',j') can be performed (for $p_{w'} \leq p_w$

and for all *j´*) between the starting time $(ts_{(w,j)})$ and the ending time $(te_{(w,j)})$ of the inserted transfer *(w,j)*

During this iterative evaluation process the transfer times are initialized (Initialized Process), then they are compared with all the other transfers times (Comparison Process) and finally, they are updated (Updating Process). This loop is repeated successively for a given transfer, until all the comparisons, with the previously inserted transfers, do not introduce new updates at the compared transfer times. So, the comparison and updating processes end. Then, the transfer is evaluated and loaded onto the system with its respectively times $[ts_{(w,j)}, te_{(w,j)}]$, the counter number of iteration without change (*iter)* and the number of transfers loaded onto the system *(transf)* are updated and the next transfer from the list *(w,j)* with $j=j+1$ if $j < M+1$; or $w=w+1$ and $j=1$ if $w < N$, is taken for the comparison. The algorithm ends when there are no more transfers to be compared in the system $(j=M+1$ and $w=N)$.

The simplified logic proposed is summarized in Figure 4.

Next, the Generation Process is explained more in detail as well as the procedures of Initialization, Comparison and Update of the Evaluation Process, all of them generated by our algorithm.

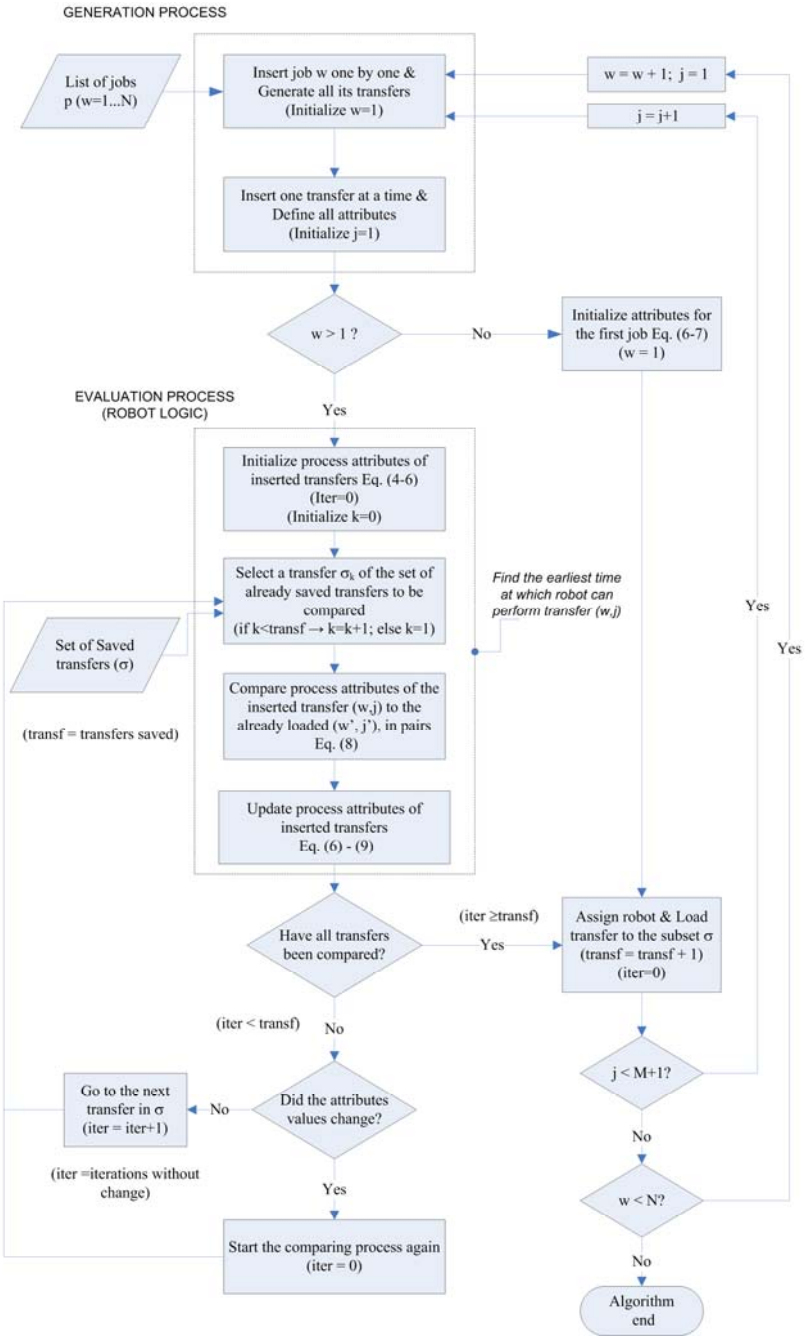


Figure 4: Pseudocode of the Generation and Evaluation Algorithm for transfers

Generation process

To apply the logic in the simulation model it was necessary to define each transfer as a particular new entity in the system, together with the entities associated to the jobs in the system. Consequently, a given job *"i"* will have associated a certain number of transfers and/or entities *(i,j)* corresponding to the quantity of baths into the system $j=1...M+1$.

Therefore, to start the processing of a given job *i,* all its *j* transfers must be pre-loaded into the system. Going back to equations (1) and (2), we can notice that the treatment of the transfers must be done in successive pairs. In order to define the start and end time of the transfer, it is necessary to correctly arrange the successive transfers in the robot, without overlap with any other transfer in the system. So, infeasible schedules are avoided. For that, it is necessary to define a set of attributes $[ts_{(i,j)},te_{(i,j)}]$ $[ts_{(i,j+1)},te_{(i,j+1)}]$, for each transfer *(i,j)* in order to define a correct sequence of transfer over time, avoiding infeasible solutions for the future transfer at the same job *i (i, j+1)*.

Evaluation process

After defining all the attributes of the inserted transfer, we proceed to determine an initial value.

Initialized Procedure: the initialization procedure consists on determining the lower value at which the transfer can be initialized, assuming that there are not limitations of resources. So, we can determine the initial value $ts_{(w,j)}$ for each transfer using the following equations (4)-(5).

For chemical baths (baths with odd number), equation (4) is applied:

$$ts_{(w,j)} = Max\begin{bmatrix} te_{(w-1,j+1)} \\ te_{(w-1,j+2)} - \pi_{(j)} - tp_{(w,j)} \\ te_{(w,j-1)} + tp_{(w,j-1)} \end{bmatrix}$$
$$j = 1,3,5...M+1; \forall w = 2...N \qquad (4)$$

While for water baths (baths with even number), equation (5) is applied:

$$ts_{(w,j)} = Max\begin{bmatrix} ts_{(w,j-1)} + \pi_{(j-1)} + tp_{(w,j-1)} \\ te_{(w-1,j+1)} \end{bmatrix}$$
$$j = 2,4,6...M; \forall w = 2...N \qquad (5)$$

There, $te_{(w,j)}$ is calculated for all baths *j* with the equation (6).

$$te_{(w,j)} = ts_{(w,j)} + \pi_{(j)}$$
$$j = 1,2,3...M+1; \forall w = 1...N \qquad (6)$$

So, for any job $w > 1$ the initial state of the attributes in the system is determined: $[ts_{(w,j)},te_{(w,j)}]$; $[ts_{(w,j+1)},te_{(w,j+1)}]$.

Instead, for $w = 1$, the initial values of the attributes are defined following equation (6) and (7).

$$ts_{(w,j)} = \sum_{j'=1}^{j'=j-1} tp_{(w,j')} + \sum_{j'=1}^{j'=j-1} \pi_{(j')}$$
$$j = 2,3...M+1; \forall w = 1 \qquad (7)$$

For the first transfer in the system *(w=1* and *j=1)*, the initial value is equal to zero $(ts_{(1,1)} = 0)$.

Comparison Procedure: Once transfers are initialized to $w=1$, they are loaded in the system by updating the subset of charged transfers σ. In σ there are all the transfers *(w,j)* that have been previously compared and assigned to the robot in a correct way. The value σ_k represents the *k-th* transfer analysed and initialized into the system according to the priorities described above.

The comparison procedure is applied to the p_w position with $w > 1$. During this iterative procedure, the inserted transfer *(w, j)* is compared in pairs with a transfer *(w´,j´)* of the subset σ, already assigned to the robot (being the $p_{w'}$ position $< p_w$, that means $w' < w$).

If analyzing the attributes *(w,j)* ($[ts_{(w,j)},te_{(w,j)}]$ and $[ts_{(w,j+1)},te_{(w,j+1)}]$) of the transfer with the ones already inserted *(w´,j´)* ($[ts_{(w',j')},te_{(w',j')}]$) there is any overlap between the values of them, then the algorithm will update them for avoiding overlaps (see Equation (8)). Otherwise, the attributes will not be updated. That means that transfer *(w,j)* does not overlap with *(w´,j´)*.

$$\begin{aligned} &If && \left(te_{(w,j)} \le ts_{(w',j')}\right) \vee \left(ts_{(w,j)} \ge te_{(w',j')}\right) \\ &Then && [ts_{(w,j)} = ts_{(w,j)}] \wedge [te_{(w,j)} = te_{(w,j)}] \\ &Else\_If && \left(ts_{(w,j)} < te_{(w',j')}\right) \wedge \left(te_{(w,j)} > ts_{(w',j')}\right) \\ &Then && [ts_{(w,j)} = te_{(w',j')}] \wedge [te_{(w,j)} = ts_{(w,j)} + \pi_{(j)}] \\ & && \forall (w,j) \ne (w',j'); \forall w = 2...N; \forall w' \le w; \forall j, j' \end{aligned} \qquad (8)$$

As can be seen, the updating process consists in delaying the start time of transfer *(w,j)* when overlapping with *(w´,j´)* are observed. Initially, it is necessary to compare the attributes $[ts_{(w,j)},te_{(w,j)}]$ vs. $[ts_{(w',j')},te_{(w',j')}]$ and then $[ts_{(w,j+1)},te_{(w,j+1)}]$ vs. $[ts_{(w',j')},te_{(w',j')}]$. Thus, we try to ensure that if $ts_{(w,j)} \ge te_{(w',j')}$, then by equation (1) and (2) $ts_{(w,j+1)} \ge te_{(w',j')}$, else if $te_{(w,j+1)} \le ts_{(w',j')}$ then $te_{(w,j)} \le ts_{(w',j')}$.

Update Procedure: This procedure is used to generate the earliest time at which the analyzed transfer *(w,j)* can be executed, in relation with the transfers previously inserted *(w´, j´)* and taking into account the resource constrains. As result, the efficient assignment and the detailed program of the robot is determined.

The procedure tries to recalculate the value of the attributes $[ts_{(w,j)},te_{(w,j)}]$ and $[ts_{(w,j+1)},te_{(w,j+1)}]$ from the *(w,j)* transfer fulfilling the equations (1) and (2). As result of the comparison process, the attributes $[ts_{(w,j)},ts_{(w,j+1)}]$ will be updated according to equation (9).

$$
\begin{aligned}
&\textit{If} && \left(ts_{(w,j)} + tp_{(w,j)} + \pi_{(j)}\right) \geq ts_{(w,j+1)} \\
&\textit{Then} && ts_{(w,j+1)} = ts_{(w,j)} + tp_{(w,j)} + \pi_{(j)} \\
&\textit{Else\_if} && \left(ts_{(w,j)} + tp_{(w,j)} + \pi_{(j)}\right) < ts_{(w,j+1)} \\
&\textit{Then} && ts_{(w,j)} = ts_{(w,j+1)} - tp_{(w,j)} - \pi_{(j)} \\
& && j = 1,3,5...M+1; \forall w = 2...N
\end{aligned} \qquad (9)
$$

For $j=2,4,6...M$ or if not met any of the conditions, only the values of $[te_{(w,j)}, te_{(w,j+1)}]$ are updated. Both of the values are recalculated according to equation (6).

It may be possible that after the end of the processes of comparison and updating, some of the analyzed transfer's attributes overlap again with the previously compared transfer or with some other in the system. If this occurs, the algorithm makes a loop in the comparison process selecting the next σ_k transfer, saved in the σ subset. It also updates the iterations counter to zero (*iter = 0*).

If there isn't any change in the attributes, the algorithm returns to the comparison process and evaluates the analyzed transfer with the next transfer in the σ sequence. Then, the iteration counter *iter* is updated to *iter + 1* (*iter = iter + 1)*.

In both cases, the comparison is made with the transfer of job σ_k, where *k=k+1* if *k < transf,* or otherwise: *k=1*; being *transf* equal to the number of elements in the σ set *(transf = card(k))*.

This iterative process is performed for all the possible comparisons. While this method may be not efficient from the procedural standpoint, since there are unnecessary and redundant comparisons, it tries to avoid the generation of unwanted or erroneous results after the updating stage.

Since to the comparison process is simple and the additional number of events does not report high updating times, we can demonstrate that our algorithm is able to deal with industrial scale problems with modest computational cost.

Finally, when the analyzed transfer *(w,j)* has been compared dynamically with all the transfers *(w´,j´)* of the σ sequence without updating attributes, that is that the algorithm iteration number *(iter)* is greater or equal than the σ set cardinality *(iter ≥ transf)*, then the last transfer is loaded into the system with its respectively times, and the number of elements of σ set are updated *(transf = transf + 1)*. The iteration counter is initialized *(iter = 0)* and the robot is assigned to the *(w,j)* transfer during the time between the interval $[ts_{(w,j)}, te_{(w,j)}]$. The next transfer will be *(j = j+1)* if *j<M+1*.

Otherwise, if *j* is the last bath of the sequence *(j=M+1)* and *w* is not the last job of the ***p*** sequence *(w < N)* then, the process continues with the next *(w = w+1)* job in the ***p*** sequence and *j=1* is established.

The algorithm ends when there are not more transfers to be evaluated. In this case, *w=N* and *j=M+1*, that means that all transfers have been loaded into the system *(transf = N*M+1)*.

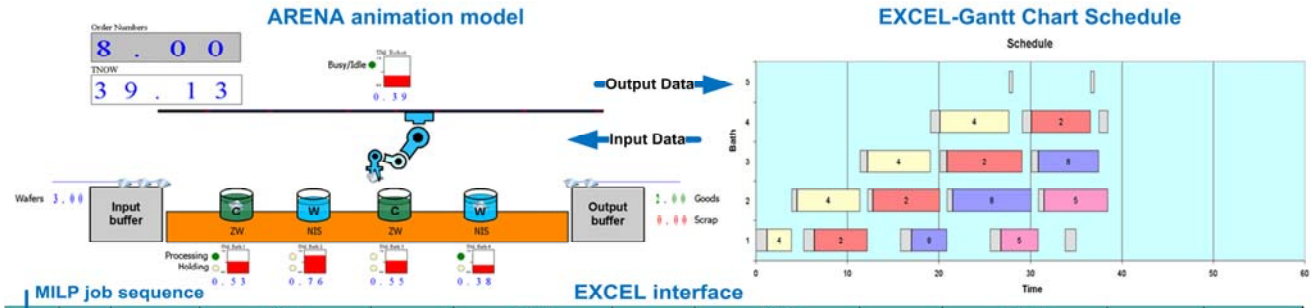
As result, our algorithm ensures that no pair of transfers inserted into the system and assigned to the robot may overlap over time. Thus, a feasible schedule for both, the process and robot, is generated.

4.3. Implementation in the simulated model

Once the timing of transfers is defined, the model is able to emulate the real system behaviour while satisfying the job processing time, the mixed intermediate storage policies and the assignment of transfers to the limited shared resource.

The simulation is run by using the model resources (baths and robot) and the waiting modules (Queues/Hold/Match). The waiting modules hold the entities until a given condition is met.

While jobs are being processed in the system, according to the predefined job sequence given by ***p***, the transfers' values are updated using specific writing and reading modules (Read/Write). Thus, a fast and simplified way of interacting with Microsoft Excel® is permitted (see Figure 5), defining dynamically the detailed schedule for the baths and robot, together with the generation of dynamic charts representing the evolution of the different works (operations and transfers) over time.



nº order	Jobs	Release Time	Bath 1 Start Time	Bath 1 Finish Time	Bath 1 Job State	Bath 2 Start Time	Bath 2 Finish Time	Bath 2 Job State	Bath 3 Start Time	Bath 3 Finish Time	Bath 3 Job State	Bath 4 Start Time	Bath 4 Finish Time	Final Job State	Final Time
1	4	0,00	1,20	3,90	finished	4,50	11,40	finished	12,20	19,10	finished	20,10	27,70	finished	28,10
2	2	5,20	6,40	12,20	finished	12,80	20,10	finished	20,90	29,10	finished	30,10	36,60	finished	37,00
3	8	15,80	17,00	20,90	finished	21,50	30,10	finished	30,90	37,50	finished	38,50			
4	5	25,60	26,80	30,90	finished	31,50	38,50	finished							
5	1	33,80	35,00												
6	7														
7	3														
8	6														

Figure 5: Dynamic Gantt Chart Schedule Generated by a User-friendly Excel Interface

As a result, the dynamic operation can be controlled and analyzed in a global perspective. Failures and/or possible improvement actions can be easily observed by analysing the graphical interface. For example, it can be easily identified how a change in the process sequence impacts over the processing time of each bath and in the availability of the shared resource.

Also, the simulated model progressively evaluates the utilization of the system resources (bath and robot), using monitors or animated screens (see Figure 6), which allow to execute a detailed control of the shared resources performance over time. Thus, it is possible to identify the critical points (resources and/or stages intensively used in the system) with the aim of evaluating alternative modifications in the process design (change the number of resources, or use parallel resources) and/or in the process operation (resources assignment and priority of processing)

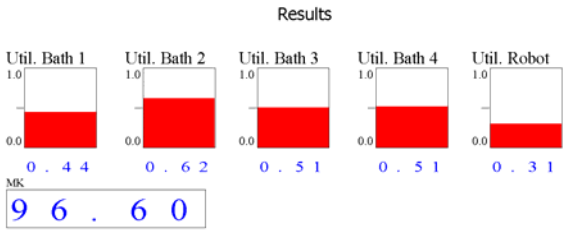


Figure 6: Monitoring the resource utilization

4.4. Animation module of the AWS station

Additionally, the model displays the dynamic behavior of the AWS station through the animation of main system components (entities, resources, performance indicators). Thus, the system operation, involving baths (chemicals and water) and robot activities can be easily evaluated (see Figure 7).

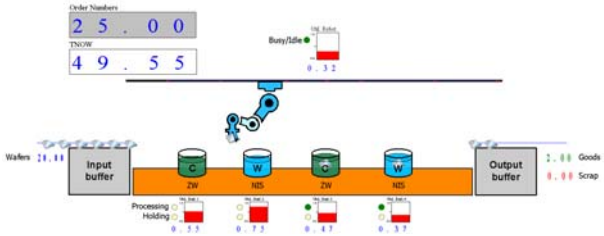


Figure 7: In-progress animation of the AWS station

4.5. Performance measures and termination criterion

The proposed algorithm looks for the best permutation sequence ***p*** of the different jobs to be scheduled. This is determined based on the timing of jobs at the consecutive stages and also by the detailed feasible schedule of the transfer robot activities.

Start and end times of activities are dynamically reported in Excel®, according to the different events that take place in the system, at each stage of the process through the simulation.

The main goal is to achieve the shortest completion time of all jobs in the system. So, the objective function can be estimated with the final time of the last transfer of the robot in the AWS station $(te_{(w,j)}$ for $w=N$ and $j=M+1)$.

For our model, the estimation of this time is determined by the *MK (Makespan)* variable. This variable analyses the simulation variable *TNOW* every time a job is finished.

TNOW is a global variable managed by the simulator that indicates the actual time at which the different events are happening throughout the simulation. In turn, the time for completing the last job in the system represents the stopping criterion of the simulation run (Termination Criterion).

Other performance measures are the utilization of baths and robot. In our particular case, they are used to compare alternative solutions in order to determine alternative policies and logic for the robot allocation.

5. ALTERNATIVE SOLUTION STRATEGIES

A natural way to get a good result of complex problems is to try to break the whole problem at different stages (Bhushan and Karimi, 2003). An iterative solution involves decomposing the whole problem into independent sub-problems, using the solution from one stage as input data for the next one, in order to obtain a global solution in a sequential manner.

In our particular case, generating a good initial ***p*** sequence for all the jobs to be processed in the system may notably reduce the complexity of sequencing robot decisions.

The use of meta-heuristics (Bhushan and Karimi, 2004) and mathematical programming models (MILP) (Bhushan and Karimi, 2003; Aguirre, Méndez and Castro, 2011; Zeballos, Castro and Méndez, 2011), are some of the existing tools used to obtain a good initial sequence ***p*** for large size AWS scheduling problems.

Here, we present an alternative solution to the robot sequencing problem, based on modern simulation techniques and tools. We also know that in these highly combinatorial problems there exist always a trade-off between computational times and optimal solutions.

For this reason, we have proposed an interesting alternative for obtaining an efficient solution. It is based on a MILP model that provides the best solution to the problem assuming unlimited robots, in order to obtain the optimal ***p*** sequence of the jobs in the system. Then, this information is taken as input data by the simulator in order to obtain a feasible and efficient solution to the whole problem, involving the sequencing robot activities. For this, we use the solution provided by a continuous-time formulation developed by Aguirre, Méndez and Castro (2011). Thus, we will initially solve different cases without considering the robot constraints, to subsequently incorporate the results of the sequence into the simulation model.

Finally, in order to validate the model developed, we compare the results with the ones obtained by a rigorous mathematical formulation (MILP), considering the same ***p*** sequence in both solutions and also with the results obtained by a full-space MILP model

considering all robots restrictions. Several examples of different sizes are efficiently solved by using this strategy. We will analyse the results obtained from the comparison of those techniques.

5.1. Cases Studies

To prove the applicability of the internal and external logic of the simulation model, different examples using the proposed method are tested. Also, the results generated are compared with optimal MILP solutions found by Aguirre, Méndez and Castro (2011) and the heuristic procedure RCURM from Bhushan and Karimi (2003), by using the previous mentioned MILP model.

The problem instances have been obtained from literature, for an specific *MxN* configuration for the first *M* baths and *N* jobs of the original problem presented by Bhushan and Karimi, 2004.

6. RESULTS AND COMPARISONS

The heuristic methodology RCURM ("Resource Constrained Unlimited Robot Mathematical Model") is based on a MILP model that can solve moderate size problems with reasonable computational effort in comparison with pure mathematical models. Two alternative models, URM ("Unlimited Robot Model") and ORM ("One Robot Model") were solved sequentially in order to obtain a solution for the whole problem.

The first one, i.e. the URM, is used to generate an optimal job sequence that ignores the robot restrictions. The URM just only takes explicitly into account the predefined transfer times, assuming that a robot will be always available to perform the transfer operations.

The ORM, in turn, considers the impact of limited transfer resources in the objective function. This proposed model also takes into consideration the sequential use of the single transfer movement device, which enforces a proper synchronization of bath schedules and robot activities.

The idea of the RCURM is to first solve the problem using the URM model, to then fix the production sequence obtained by this model and solve the detailed robot schedule through the ORM formulation. Following this idea, the simulation model will receive as input data the sequence obtained by the URM to then simulate the whole process including the robot activities. As it is shown in Table 1, for the examples validated, the Simulation Model gives the same *MK* value than the RCURM-MILP for the first three problem instances. This is a good indicator to conclude that the simulation logic may generate results that are as effective as optimal MILP solution, that can obtained with a modest computational effort.

Table 1: Model Statistics for a few *MxN* problem instances

MxN	*Statistics*	*Unlimited Robot Model (URM-MILP)*	*One Robot Model (ORM-MILP)*	*Resource Constrained Unlimited Robot Model (RCURM-MILP)*	*Arena Simulation Model using URM Sequence*
4x8	*Binary Variables*	28	588	560	-
	Makespan	95.1	95.6	95.6	95.6
	CPU Time (s)[a]	0.484	11.25	0.091	-
	Job Sequence p	4-2-8-5-1-7-3-6	4-2-5-8-1-7-3-6	4-2-8-5-1-7-3-6	
4x10	*Binary Variables*	45	945	900	-
	Makespan	115.5	115.6	116	116
	CPU Time (s)[a]	6.785	488.7	0.122	-
	Job Sequence p	9-2-5-8-10-4-1-7-3-6	9-6-5-4-10-2-8-1-7-3	9-2-5-8-10-4-1-7-3-6	
4x14	*Binary Variables*	91	1911	1820	-
	Makespan	154.7	158.8	156.2	156.2
	CPU Time (s)[a]	3600[b]	3600[b]	0.235	-
	Job Sequence p	9-12-5-8-7-11-14-10-2-4-1-13-3-6	9-2-8-12-4-14-10-11-5-1-3-7-13-6	9-12-5-8-7-11-14-10-2-4-1-13-3-6	
8x10	*Binary Variables*	45	3285	3240	-
	Makespan	149.4	154.4	156.7	166.4
	CPU Time (s)[a]	55.07	3600[b]	3.42	4.8
	Job Sequence p	6-9-2-1-3-4-7-5-10-8	4-9-3-1-2-6-7-5-10-8	6-9-2-1-3-4-7-5-10-8	
12x10	*Binary Variables*	45	7065	7020	-
	Makespan	192.2	206.3	197.2	227.8
	CPU Time (s)[a]	145.5	3600[b]	152.97	7.2
	Job Sequence p	6-8-3-2-9-5-10-7-4-1	6-1-2-10-5-3-9-7-8-4	6-8-3-2-9-5-10-7-4-1	
12x12	*Binary Variables*	66	10362	10296	-
	Makespan	210.7	NFS[c]	215.8	264.3
	CPU Time (s)[a]	3600[b]	3600[b]	2249.7	9.0
	Job Sequence p	4-8-10-3-11-2-9-5-12-1-7-6	-	4-8-10-3-11-2-9-5-12-1-7-6	
12x15	*Binary Variables*	105	16485	16380	-
	Makespan	241.9	NFS[c]	NFS[c]	334.2
	CPU Time (s)[a]	6.785	3600[b]	3600[b]	12.0
	Job Sequence p	6-8-3-11-2-13-9-5-14-10-12-1-15-4-7	-	6-8-3-11-2-13-9-5-14-10-12-1-15-4-7	
12x25	*Binary Variables*	300	47100	46800	-
	Makespan	357	NFS[c]	NFS[c]	516.8
	CPU Time (s)[a]	3600[b]	3600[b]	3600[b]	48.0
	Job Sequence p	6-16-8-11-20-4-21-18-17-19-5-10-15-22-14-2-12-3-25-13-24-9-23-7-1	-	6-16-8-11-20-4-21-18-17-19-5-10-15-22-14-2-12-3-25-13-24-9-23-7-1	

(a)MILP models were solved by using CPLEX 12 in a PC Core 2 Quad parallel processing in 4 threads. (b) Termination criterion (3600 CPU s). (c) No feasible solution found after 3600 sec.

By analyzing the model statistics, we can notice that the solutions generated by the Simulation Model, by using the URM sequence, are very close to the ones found by the ORM and RCURM models, which points out the high performance of the alternative proposed methodology for many small size cases. But, when the model size increases the solution obtained by this approach becomes poor in comparison with ones reported by RCURM and ORM models.

The most important difference between ORM/RCURM-MILP approaches and our Simulation Model lies on the computational time consumed, what is more evident in medium size and large size cases, as *4x14, 8x10, 12x10* and *12x12, 12x15, 12x25* configurations respectively.

Moreover, for many larger problems only the Simulation Model may find feasible solutions of the entire problem with very low computational cost.

In consequence, the application of the proposed solution strategy to manage the activities of the robot will compare favourably against a MILP mathematical formulation and a MILP-based decomposition method for many large-size problems in the AWS station. Also, the solution generated by this approach can be considered as an initial solution of the whole problem, which may be later enhanced by alternative meta-heuristic or optimization-based methodologies.

CONCLUSIONS AND FUTURE WORK

A novel discrete event simulation model has been developed to simultaneously address the integrated scheduling problem of manufacturing and material-handling devices in the AWS in the semiconductor industry. The proposed model can be easily used to dynamically validate, generate and improve different schedules. We have demonstrated that the proposed solution algorithm for the robot is able to generate very effective results with modest computational effort. For large-sized cases, only our simulation approach found feasible solutions to the problem in a reasonable computational time.

In addition, alternative heuristic rules can be easily embedded into the simulation framework for making convenient timing and sequencing decisions. At the same time, alternative system configurations involving several robots for wafer-handling in the AWS station can be easily considered. As a future work, a hybrid approach lying on the concepts of optimization and simulation tools will be developed in order to improve the generation of the solution for the whole scheduling problem.

ACKNOWLEDMENTS

Financial support received from Fundação para a Ciência e Tecnologia and Ministerio de Ciencia, Tecnología e Innovación Productiva, under the Scientific Bilateral Cooperation Agreement between Argentina and Portugal (2010-2011), from AECID under Grant PCI-D/030927/10, from CONICET under Grant PIP-2221 and from UNL under Grant PI-66-337 is fully appreciated.

REFERENCES

Aguirre, A. M., Méndez, C. A., Castro, P. M., 2011. *A Novel Optimization Method to Automated Wet-Etch Station Scheduling in Semiconductor Manufacturing Systems*. Comput. Chem. Eng., doi:10.1016/j.compchemeng.2011.02.14.

Banks, J., J. S. Carson, B. L., Nelson, D. M. Nicol. 2004. Discrete-Event System Simulation. 4th ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Bhushan, S., and Karimi, I. A., 2003. *An MILP approach to automated wet-etch scheduling*. Industrial and Engineering Chemistry Research, 42(7), 1391-1399.

Bhushan, S., and Karimi, I. A., 2004. *Heuristic algorithms for scheduling an automated wet-etch station.* Computers and Chemical Engineering, 28, 363-379.

Geiger, C., Kempf K. G., and Uzsoy, R., 1997. *A tabu search approach to scheduling an automated wet etch station.* Journal of Manufacturing System, 16, 102-116.

Kelton, W. D., R. P. Sadowski, D. T. Sturrock., 2007. Simulation with Arena. 4th ed. New York: McGraw-Hill Inc.

Law, A. M., 2007. Simulation Modeling and Analysis. 4th ed. New York: McGraw-Hill, Inc.

Zeballos, L. J., Castro, P. M., Méndez, C. A., 2011. *Integrated Constraint Programming Scheduling Approach for Automated Wet-Etch Stations in Semiconductor Manufacturing.* Ind. Eng. Chem., 50, 1705.

Dr. CARLOS A. MENDEZ is a Titular Professor of Industrial Engineering at Universidad Nacional del Litoral (UNL) in Argentina as well as an Adjoint Researcher of the National Scientific and Technical Research Council (CONICET) in the area of Process Systems Engineering. He has published over 100 refereed journal articles, book chapters, and conference papers. His research and teaching interests include modeling, simulation and optimization tools for production planning and scheduling, vehicle routing and logistics.

Dr. PEDRO M. CASTRO is an Assistant Researcher at Laboratório Nacional de Energia e Geologia in Portugal in the area of Process Systems Engineering. His research interests include scheduling of batch and continuous processes, mixed integer and nonlinear optimization, and process integration. He has published over 30 papers in international journals with refereeing and has been invited to give seminars in a few Universities and Research Centers worldwide.