Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

SIMULATION-BASED FRAMEWORK TO AUTOMATED WET-ETCH STATION SCHEDULING PROBLEMS IN THE SEMICONDUCTOR INDUSTRY

Adrián M. Aguirre
Vanina G. Cafaro
Carlos A. Méndez

INTEC (UNL-CONICET)
Güemes 3450
Santa Fe, 3000, Argentina.

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). Principal components, templates and tools available in the *Arena*® simulation software are used to achieve the best representation of this complex and highly-constrained manufacturing system. 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. This model presents a flexible structure that can be easily adapted to emulate random scenarios with uncertain processing and transfer times. A user-friendly interface for dealing with real-world applications in industry is also introduced.

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 (Geiger et al. 1997).

An important process restriction is that each robot can only transport a single wafer lot at a time without holding the wafer lot more than the exact transfer time. Since there is no intermediate storage between consecutive baths, this condition can be considered as a non-intermediate storage (NIS) policy in every bath, which must be followed 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 addition, wafer 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 (Bhushan and Karimi 2003).

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. Contributions lying on classical optimization methods, such as mixed integer linear programming (MILP) procedures (Bhushan and Karimi 2003; Aguirre and

Méndez 2010), are traditionally useful to solve moderate size problems, due to the excessive CPU time required.

To the best of our knowledge, efficient systematic solution methods still need to be developed to represent and evaluate the complex dynamic behaviour of the AWS for any system size. 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 with randomness in the processing and transfer times of the system. 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.

2 PROBLEM STATEMENT

The AWS scheduling problem provides a complex interplay between material-handling limitations, processing constraints and stringent mixed intermediate storage (MIS) policies. 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, following a non-intermediate storage (NIS) policy between successive 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 policy must be ensured in chemical baths whereas LS policy is allowed in water baths.

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 limited number of robots $(r=1,...,M+1)$ is available, which have to perform all the transportation activities in the system (see Figure 1).

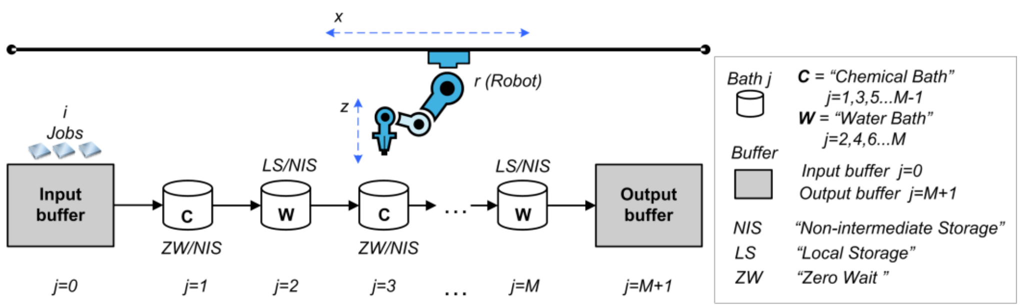

Figure 1: Automated Wet-etch Station (AWS) process scheme

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 $(r=1)$ with finite load capacity for the wafer movement between consecutives 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 solution structure is capable of effectively managing limited or unlimited capacity of material-handling devices. Also, it permits to accomplish a systematic analysis to study how the system will perform in its "as-is" configuration and under a numerous of potential alternatives, when the random behaviour is applied to processing and transfer activities.

This methodology allows also evaluating and improving the operation and reliability of baths and robot schedules, in addition permits evaluating 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 of random system scenarios.

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 2007; Kelton et al. 2007; Seppanen et al. 2005).

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 the 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.

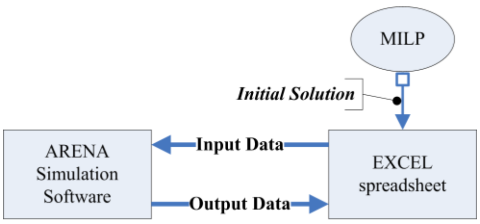

Figure 2: Information exchange between *Excel – Arena – MILP Software*

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 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. Otherwise, the piece must be discarded or held, depending on the type of bath (chemical or water).

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 while in water baths waiting times of wafers are allowed. For these baths, the logic takes into account the following tasks: (i) reports the time at which the process begins and ends; (ii) seizes the corresponding bath after the delay time finishes; (iii) performs the transfer to the following bath, only if the robot and the destination bath are available, otherwise the piece must be discarded or held. Notice that when chemical bath finishes, the wafer must be immediately removed and transferred to the succeeding water bath while in water baths the wafer can be held until this condition occurs.

It is important to note that the logic driven the chemical and water process permits to easily identify why/when a given wafer is discarded. Basically, it may occur because the robot and/or next bath are not available. This allows making a detailed analysis about the system behaviour, executing, if it is necessary, the corresponding adjustments.

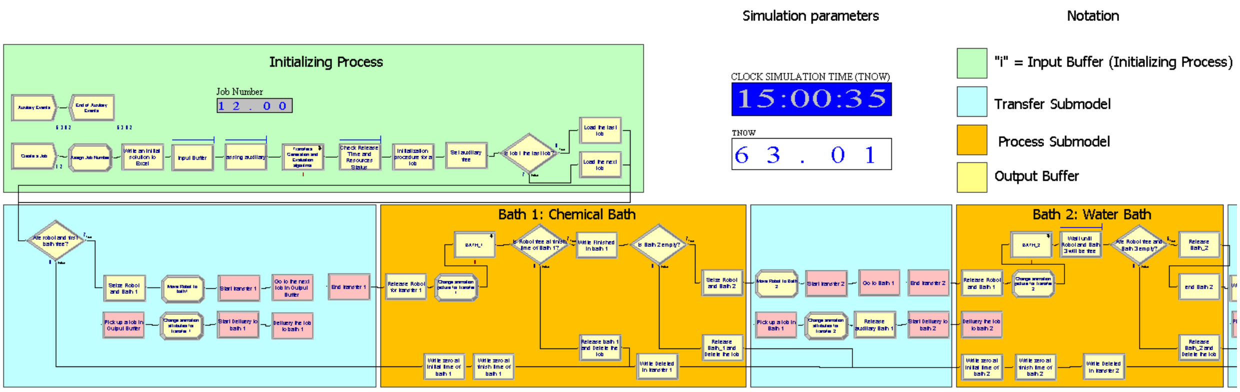

Figure 3: Partial size view of the in-progress simulation model generated in the *Arena* environment

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. Consequently, 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.

Due to there is only a single robot to do all the job movements, the sequence in which the transfers will be performed needs to be explicitly 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 times $ts_{(i,j)}$ 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,…,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 $ts_{(i,j+1)}$ of the transfer to the water bath j+1 must strictly satisfy equation (1).

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

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.

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

Let $\boldsymbol{p} = \{p_1, p_2, p_3, ..., p_N\}$ define a permutation processing sequence N different jobs. p_w represents the *w-th* position of a job i (i=1, 2, ..., 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.

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

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 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 are inserted before 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, 2, ..., *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 $p_{w'} \leq p_w$ & 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 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 conclude. 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 & 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 *& w=N)*.

The simplified logic proposed is summarized in Figure 4.

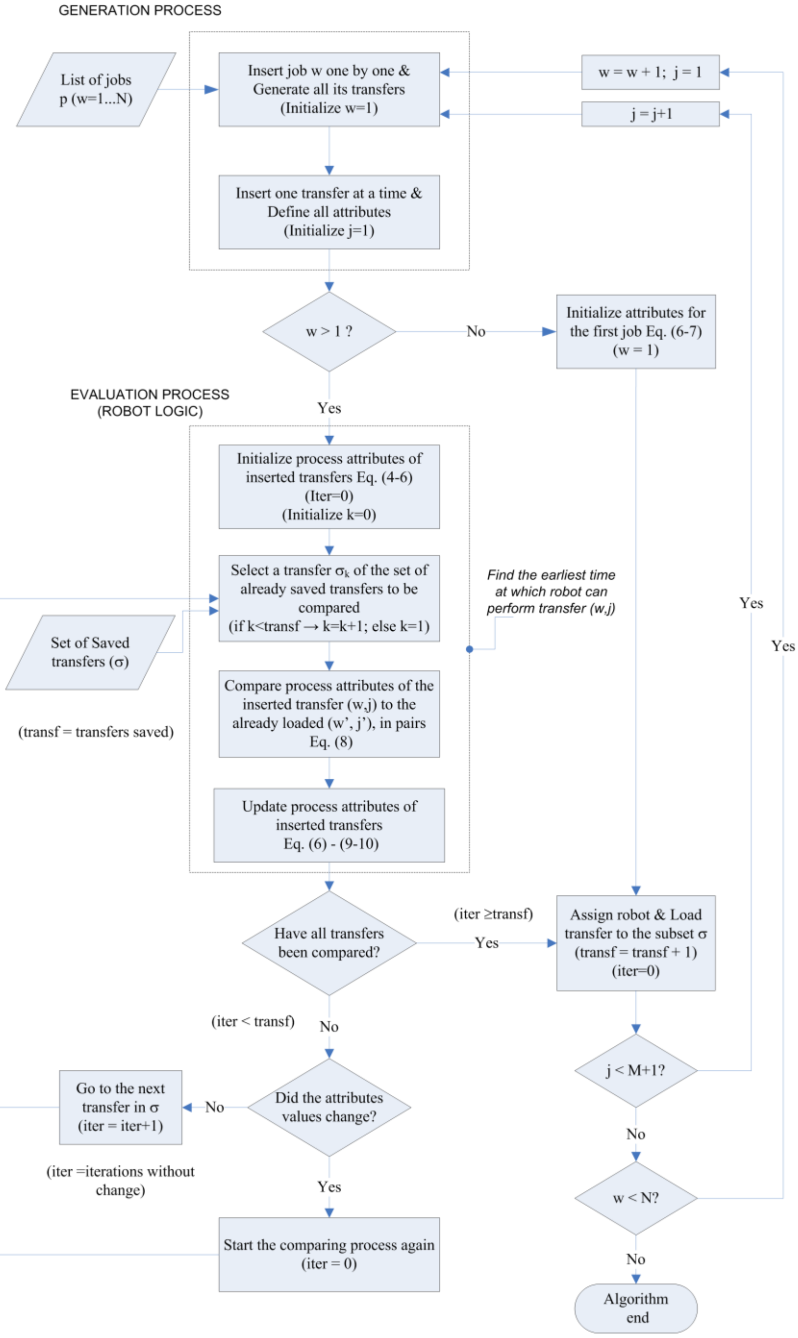

Figure 4: Pseudocode of the Generation and Evaluation Algorithm for transfers

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, 2, ..., 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 reason, 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: Initial values for attributes of the inserted transfers are determined.

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:

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

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

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

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

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

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 by equations (6-7).

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

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′)*.

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

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 $ts_{(w,j)} \geq te_{(w´,j´)}$, then by equation (1) and (2) $ts_{(w,j+1)} \geq te_{(w´,j´)}$, else if $te_{(w,j+1)} \leq ts_{(w´,j´)}$ then $te_{(w,j)} \leq ts_{(w´,j´)}$.

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 try 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).

$$\begin{array}{lll} \textit{If} & \left(ts_{(w,j)} + tp_{(w,j)} + \pi_{(j)}\right) \geq ts_{(w,j+1)} & \\ \textit{Then} & ts_{(w,j+1)} = ts_{(w,j)} + tp_{(w,j)} + \pi_{(j)} & j = 1,3,5...M+1; \forall w = 2...N \quad (9) \\ \textit{Else\_if} & \left(ts_{(w,j)} + tp_{(w,j)} + \pi_{(j)}\right) \leq ts_{(w,j+1)} & \\ \textit{Then} & ts_{(w,j)} = ts_{(w,j+1)} - tp_{(w,j)} - \pi_{(j)} & j = 1,3,5...M+1; \forall w = 2...N \quad (10) \end{array}$$

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 comparison and updating processes, some of the analyzed transfer 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 in zero (*iter* = 0).

If there isn't any change in the attributes, the algorithm returns to the comparison process and evaluate 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. Beside 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.

Due 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 no more transfers to be evaluated. In this case, w=N & j=M+1, it 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 Table 1), defining 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 (see Figure 5 and 6).

Table 1: Schedule Generated by a User-friendly Excel Interface.

			Bath 1			Bath 2		
nº order	Jobs	Release Time	Start Time	Finish Time	Job State	Start Time	Finish Time	Job State
1	4	0,00	1,20	3,90	finished	4,50	11,40	finished
2	2	5,20	6,40	12,20	finished	12,80	20,10	finished
3	8	15,80	17,00	20,90	finished	21,50	30,10	finished
4	5	25,60	26,80	30,90	finished	31,50	38,50	finished
5	1	33,80	35,00	39,30	finished	39,90	51,30	finished
6	7	40,40	41,60	52,10	finished	52,70	66,60	finished
7	3	55,6-56,9	56,8-58,1	67,4-68,7	finished	68,0-69,3	75,7-76,2	finished
8	6	72,10	73,30	77,00	finished	77,60	84,50	finished

As a result, the dynamic operation can be controlled and analyzed in a global perspective. Fails and/or possible improvement actions can be easily observed by analyzing 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 & robot), using monitors or animated screens (see Figure 5), 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 5: 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 6).

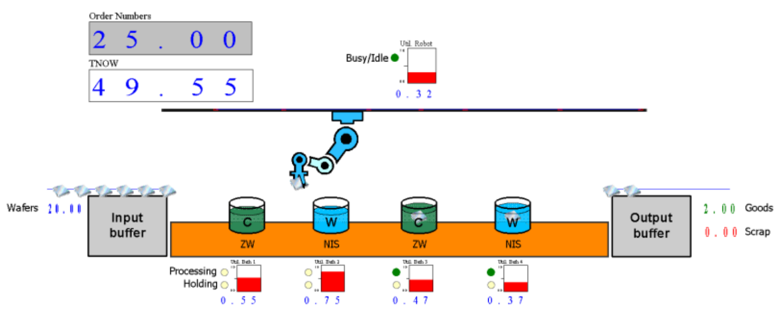

Figure 6: In-progress animation of the AWS station

4.5 Performance measures and termination criterion

The proposed model 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 ($te_{(w,j)}$ for $w=N$ and $j=M+1$).

For our model, the estimation of this time is determined by the *MK (Makespan)* variable. This variable analyzes 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).

Others performance measures are the utilization of baths and robot. These measures will be used to compare alternative solutions in order to determine the best policies for the robot allocation.

5 ALTERNATIVE SOLUTION STRATEGIES

A natural way to get a good result of complex problem 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 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 the system may notably reduce the complexity of sequencing robot decisions.

The use of heuristics, meta-heuristics (Bhushan and Karimi 2004) and mathematical programming models (MILP) (Bhushan and Karimi 2003; Aguirre and Méndez 2010), are some of the existing tools used to obtain a good initial sequence ***p***.

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 and Méndez (2010). Initially, we solve different cases without considering the robot constraints, assuming that unlimited robots are available, to subsequently incorporate the results of the job sequence ***p*** into the simulation model in order to obtain the robot schedule. Several examples of moderate size are efficiently solved by using this strategy.

The simulated results compare favourably with the ones obtained by the robust MILP considering all robot constraints. Finally, in order to validate the model developed, we compare the results with the ones obtained by a rigorous mathematical formulation (MILP), by considering the same ***p*** sequence in both solutions. We will analyze the results generated by comparing the performance of both techniques.

5.1 Application example

In order to validate the applicability of the internal and external logic of the simulation model, different problem instances are tested by using the proposed method. Also, the results generated are compared with optimal MILP solutions found by Aguirre and Méndez (2010).

These instances have been obtained from literature (Bhushan and Karimi 2004), for a specific *MxN* configuration of baths (M=4) and jobs (N=10,14) in the system and these results are reported in the next section. It is worth to remark that the information obtained for all tested instances are deterministic in order to compare both methodologies in a rigorous way. Anyway, our Simulation Model allows the possibility to work with probability distributions for processing and transfer times without any change in the logical structure.

As an example we consider a small-size problem configuration introduced by Bhushan and Karimi, (2003). It comprises *M*=4 baths and *N*=8 jobs in the system, wherein all the jobs have to be processed in every bath and the transfers should be done in one robot without overlaps. The schedule found by using the MILP model and the proposed Simulation Model with a predefined sequence ***p*** is showed in Figure 7. The detailed job schedule of both solution methods is presented in Table 2.

This example shows the dynamic operation of the AWS station and allows demonstrating the effective behaviour of the simulation model in comparison with the MILP solution.

Table 2: Timetable of a *4x8* problem by using either a simulation tool or a MILP model

			Bath 1			Bath 2			Bath 3			Bath 4			
nº order	Jobs	Release Time	Start Time	Finish Time	Job State	Start Time	Finish Time	Job State	Start Time	Finish Time	Job State	Start Time	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	44,90	finished	45,30
4	5	25,60	26,80	30,90	finished	31,50	38,50	finished	39,30	50,30	finished	51,30	58,10	finished	58,50
5	1	33,80	35,00	39,30	finished	39,90	51,30	finished	52,10	63,40	finished	64,40	70,70	finished	71,10
6	7	40,40	41,60	52,10	finished	52,70	66,60	finished	67,40	71,10	finished	72,10	78,70	finished	79,10
7	3	55,6-56,9	56,8-58,1	67,4-68,7	finished	68,0-69,3	75,7-76,2	finished	76,5-77,0	79,1-79,6	finished	80,1-80,6	86,5-87,0	finished	86,9-87,4
8	6	72,10	73,30	77,00	finished	77,60	84,50	finished	85,30	87,80	finished	88,80	95,20	finished	95,60

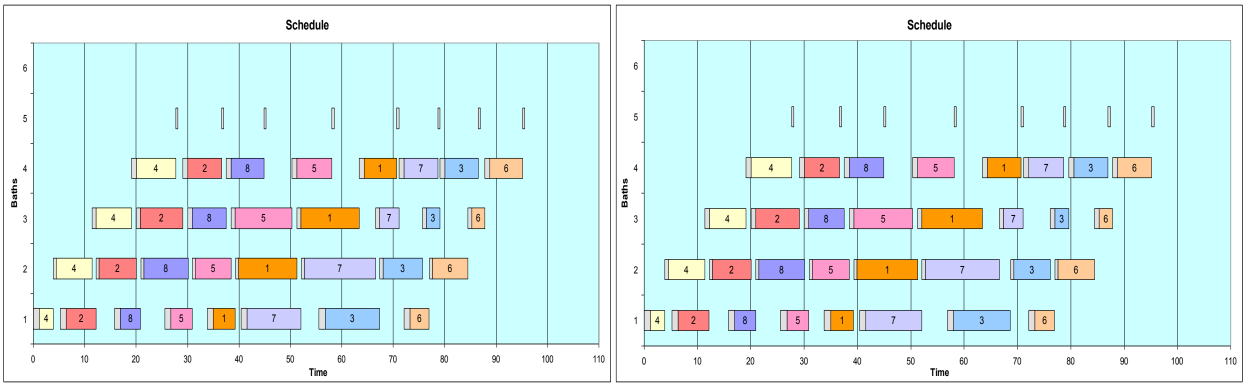

Figure 7: Gantt Chart schedule for *4x8* problem by using the MILP model and our simulation model.

6 RESULTS AND COMPARISONS

The results reported for the problem instances obtained from Bhushan and Karimi (2003, 2004) by using the simulation model, are compared to those obtained by the *Resource Constrained Unlimited Robot Mathematical Model* (RCURM-MILP).

The heuristic methodology RCURM 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, 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-MILP to then simulate the whole process including the robot activities.

As shown in Table 3, the Simulation Model finds the same *MK* value than the RCURM-MILP for the examples validated. This is a good indicator to conclude that the simulation logic may generate results that are as effective as optimal MILP solution, which can be obtained with a modest computational effort.

By analyzing the model statistics, we can notice that the solutions generated by the Simulation Model using the URM sequence, are very close to the ones found by the ORM-MILP, which points out the high performance of the alternative proposed methodology. The most important difference between ORM-MILP and the Simulation Model lies on the computational time consumed, what is more evident in medium size cases, as *4x10* and *4x14* configurations. Moreover, for larger problems, the RCURM-MILP and the simulation model may find better *MK* solutions than the ORM-MILP, with a comparable computational cost and alternative job sequences, i.e.: for *4x14* example, the solution reported by ORM-MILP *(158.8 units)* is higher than the *156.2 units* provided by the sequential use of the RCURM-MILP and the simulation model developed.

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 moderate-size problems in the AWS station.

Table 3: Model Statistics for *NxM* examples obtained from Bhushan and Karimi (2003,2004)

MxN	*Statistics*	*ORM-MILP*	*URM-MILP*	*RCURM-MILP*	*Arena Simulation Model*
4x8	*Binary Variables*	588	28	560	-
	Makespan	95.6	95.1	95.6	95.6
	CPU Time (s)[a]	11.25	0.484	0.091[c]	0.10[c]
	Job Sequence p	4-2-5-8-1-7-3-6		4-2-8-5-1-7-3-6	
4x10	*Binary Variables*	945	45	900	-
	Makespan	115.6	115.5	116	116
	CPU Time (s)[a]	488.7	6.785	0.122[c]	0.20[c]
	Job Sequence p	9-6-5-4-10-2-8-1-7-3		9-2-5-8-10-4-1-7-3-6	
4x14	*Binary Variables*	1911	91	1820	-
	Makespan	158.8	154.7	156.2	156.2
	CPU Time (s)[a]	3600[b]	3600[b]	0.235[c]	0.50[c]
	Sequence p	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	

(a) MILP models were solved by using GAMS/CPLEX 12 while Arena 12.0 was used for Simulation Models. All results reported were run in a PC Core 2 Quad parallel processing in 4 threads. (b) Termination criterion (3600 CPU sec). (c) Runtime without considering URM model.

7 CONCLUSIONS

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 moderate-sized cases, results were usually even better than those reported by rigorous optimization methodologies.

ACKNOWLEDGEMENTS

Financial support received from AECID under Grant PCI-D/030927/10, from CONICET under Grant PIP-2221, from ANPCyT under Grant PICT-2010-1073 and from UNL under Grant PI-66-337 is fully appreciated.

REFERENCES

Aguirre, A.M., and C. Méndez. 2010. "A Novel Optimization Method to Automated Wet-Etch Station Scheduling in Semiconductor Manufacturing Systems." *Computer-aided Chem. Eng.*, 28, 883-888.

Banks, J., J. Carson, B. Nelson and D. Nicol. 2004. *Discrete event system simulation,* 4th ed. Prentice-Hall.

Bhushan, S., and I. Karimi. 2003. "An MILP approach to automated wet-etch scheduling." *Ind. and Eng. Chem. Res.*, 42(7), 1391-1399.

Bhushan, S., and I. Karimi. 2004. "Heuristic algorithms for scheduling an automated wet-etch station." *Comp. & Chem. Eng.*, 28, 363-379.

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

Kelton, W., R. Sadowski and D. Sturrock. 2007. *Simulation with Arena*. 4th Edition, McGraw-Hill.

Law, A., 2007. *Simulation Modeling and Analysis.* 4th Edition, McGraw-Hill.

Seppanen, S., S. Kumar and C. Chandra. 2005. *Process Analysis and Improvement.* McGraw-Hill/Irwin.

AUTHOR BIOGRAPHIES

ADRIÁN M. AGUIRRE is a PhD student of the National Scientific and Technical Research Council (CONICET) in Argentina. His works include integrated optimization & simulation tools for the production planning & scheduling of real industrial problems. His email is <aaguirre@intec.unl.edu.ar>.

VANINA G. CAFARO is a PhD student of the National Agency for Promotion of Science and Technology of Argentina. Her research interest includes production planning and scheduling of logistics operations, optimization and discrete-event simulation. Her email address is <vcafaro@fiqus.unl.edu.ar>.

CARLOS A. MÉNDEZ is a Titular Professor of Industrial Engineering at Universidad Nacional del Litoral (UNL) in Argentina and Adjoint Researcher of CONICET in the area of Process Systems Engineering. His research and teaching interests include modeling, simulation and optimization tools for production planning & scheduling, vehicle routing & logistics. His email is <cmendez@intec.unl.edu.ar>.