Petri Net-Based Emulation for a Highly Concurrent Pick-and-Place Machine

ANNA SCIOMACHEN, STEVE GROTZINGER, AND FRANCESCO ARCHETTI

Abstract—The emulation of pick-and-place machines, in order to compute their throughput rate for different products, is a key ingredient for capacity planning and scheduling of high-volume assembly lines where these machines are used.

Manuscript received January 28, 1988; revised February 23, 1989.
A. Sciomachen and F. Archetti are with the Department of Information Sciences, University of Milan, Milan, Italy.
S. Grotzinger is with Manufacturing Research, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.
IEEE Log Number 8930423.

In this communication it is shown how an emulator can be built in the framework of Petri nets and how it can be instantiated for a particular setup and sequence of placements.

Three advantages are argued for the proposed emulator. First, the emulation code can be generated automatically. Second, the model graphically represents the concurrency and synchronization aspects of the machine's operations. Third, it allows a formal representation of the machine's operations.

I. INTRODUCTION
The activity reported in this communication is part of a research project aimed at establishing a framework in which to develop modeling and simulation tools for an important class of CNC machines, namely, pick-and-place machines, whose use is becoming increasingly important in high-volume/low-cost production of printed circuit boards.

Typically, these machines have a very high throughput potential or burst rate resulting from a highly concurrent design. This very feature makes them vulnerable to severe performance degradation when the sequence of placements and the arrangement of the components on the delivery carrier are not interrelated so as to exploit the concurrency of the machine.

In order to plan the capacity of the assembly line and to schedule the production, an accurate estimation of the machine cycle time for a given product is required before the start of production.

Unfortunately, creating a setup and sequencing for a new board requires extensive and time-consuming machine reprogramming and components loading. The elimination of this trial-and-error method on the machine is the main reason behind the need to develop emulators for these machines. DPERM, an emulator of a specific machine of this class, has been written in PL/I and extensively tested and validated for different setups and pick-and-place sequences [2].

This communication, a different approach, first suggested in [4], based on Petri net models and the use of a software tool called MEGAS [10] for their analysis and simulation, is shown to be a general framework for the development, validation, and numerical utilization of an emulator for this machine.

Petri nets that represent asynchronous concurrent systems [9], have been increasingly used recently to model manufacturing systems [1], [3], [6] and their control [7], [11]–[13].

One advantage of Petri net based models over other design languages is that they can be analyzed for structural properties of the system-like boundedness and liveness, i.e., absence of deadlocks.

Petri net-based models are executable. This feature is particularly important for the problem considered in this communication because it allows the emulation to be generated automatically, using a package such as MEGAS, from the Petri net and the production data streams without the need of writing more software for the specific emulation instance.

The challenge in emulation by Petri nets for this machine is to model, in a dynamic environment, the concurrency, synchronization, and control of the machine's operations. One wants to keep the model complete yet easy to understand and modify, executable and yet with a good run-time performance.

In Section II, the structure of the machine is briefly outlined stressing the role of concurrency and the situations which may adversely affect the throughput rate. In Section III, after a brief reminder of the basic definitions and properties of Petri nets, the model of the machine is presented. To the authors' knowledge, this model is considerably more complex than many presented in the literature. This section is written not only to describe a specific model but also to outline a general methodology for building Petri nets models. In Section IV, it is explained how the Petri net model can be
instantiated for a particular setup and sequence of placements while Section V is devoted to the numerical results. These show agreement with the results obtained using DPERT for the particular products examined. Work is now in progress to extend the Petri net model to represent other aspects of the control of the machine.

It is important to note that another approach to the representation and analysis of discrete event systems has been recently proposed in [5]. Using an unusual algebra called "Mimax Algebra" the authors in [5] establish a state-space representation of discrete event systems which allows to calculate efficiently the steady-state condition of the system. One limitation of this approach is the requirement for the system to be decision-free. This is not the case of the machine analyzed in this communication as can be clearly seen looking at the net which displays a number of conflicts.

II. DESIGN AND CHARACTERIZATION OF THE PICK-AND-PLACE MACHINE

Among the CNC pick-and-place machines, one machine, whose structure is shown in Fig. 1, has been considered. This machine has a fixed-length arm with pick-and-place heads at its ends. The arm moves between two fixed positions in the Y direction and the heads are independently capable of making a pickup and a placement, rotating and tweezing.

The workboard moves independently along the X and Y directions so that its next specified place position is in a fixed position under the head that will perform the placement.

There are two component delivery carriers which can move independently of each other in the X direction to their pick position. Each carrier has a tool magazine holding pickup and placement nozzles and slots holding component feeders.

To get an idea of the operations of the machine, a pickup and a placement being done concurrently are first considered. The following concurrent operations are performed: the respective head is being moved to the pickup position while the carrier aligns the feeder with the required component at the proper pickup position. This head may now make the pickup. Concurrently with the above set of operations, the alternate head is moving to the place position while rotating to the placement orientation, then standard or alternate tweezing must be completed. Also concurrently the board is aligning the proper place position. This head may now make the reverse vacuum placement.

Once both pick and place have been completed the arm moves the heads to the other fixed position. This set of operations of the machine will be referred to as module C or the standard scenario.

Two other operating modes of the machine are a placement and no pick (module E) and a pick and no placement (module D). These modules are typically performed at the end and the beginning of the sequence of operations.

More complex is the case in which a tool change is required. In this case, referred to as module A, a nozzle change is required accompanied by a placement being done concurrently by the alternate head. The placement requires the set of concurrent operations already described in module C. Concurrent with this set of operations, the carrier of interest is moving to align the tool magazine at the pick location. Once both those sets of operations are completed the nozzle change is initiated.

Only when the nozzle change is completed do the carriers begin their movements to align the component feeder stations with their respective fixed pick positions for the next pick and only then does the board begin to move to align the next place position. The head can now make the vacuum pickup.

Module B is the instance where a nozzle change is required while the alternate head is idle.

The production capability of the machine is represented by a minimum cycle time or correspondingly the "burst rate." If we denote by a circle (square) the time when a pick (place) is performed the behavior of the machine, with respect to the minimum cycle time, is represented by Fig. 2.

In this module all operations are working concurrently, the system is in balance, and no nozzle change is required. The throughput rate in the production environment is substantially lower than the burst rate. It is determined by the fact that the next head movement cannot begin until all processes requiring the current head position have been completed. Concurrent processes are: head movement, board movement, carriers movements, rotation for pickup, and rotation for placement.

Operations which are done in a serial manner include: indexing, standard or alternate tweezing, z-action, and nozzle change. Degradation of the performance can happen even in module C due to delays from board alignment, carrier alignment, or alternate tweezing. Other degradations of the throughput rate are typically due to nozzle changes and an imbalance in the number of components obtained from each carrier.

III. THE PETRI NET MODEL OF THE MACHINE

The basic movement of the four main subsystems of the machine, namely: arm, board, left and right carriers, are, respectively, represented by the timed transitions AM, BM, LCM, and RCM. To each of these transitions two places, input and output, are associated indicating the state of the system components whose activity is
represented by the transition. For instance, the start of the arm movement is denoted by a token in AS; when this movement is complete a token in AR indicates that the arm is ready for the next firing. Analogously, it is not possible to perform a pickup or a placement until the corresponding carrier and the board, respectively, are ready.

It is important to remark that these basic movements are concurrent. This fact is modeled in the Petri net by allowing the corresponding transitions to be enabled independently of each other in the same marking. Each transition has the potential to be the next firing transition.

The places marked by L and R denote whether the arm is currently on the left or right side of the machine. For instance, a token in L permits the activity, determined by the appropriate module, to begin execution on the left side of the machine. The operations performed on the left side are the same as those performed on the right side of the machine. For this reason the net contains two symmetric parts which are executed in turn. For the sake of simplicity, we shall only analyze activities in the left sector.

Initially, the focus is on the timed transitions ZPKL, ZPLL, TWL, and NCL which model the operations to be performed when the arm is aligned, as they are specified by the module being executed. NCL (Nozzle Change Left) represents the tool change, which will be required after the firing of either A or B, i.e., when a tool change has been actually requested by the selection of either module A or B.

TWL, standard or alternate tweezing, and ZPLL, z-action for placement, are serially required for the placement. These transitions will be required whenever a placement is required, as in the modules A, C, and E. The transition ZPKL represents the z-action for the pickup in modules C and D.

The selection of the module to be executed is performed in the net by solving the conflict arising from transitions A, B, C, D, E all being enabled after the arm movement, i.e., after the firing of AM, when both AR and L contain tokens. The conflict is solved by giving priority to one of the enabled transitions according to the production data. In the next section, it will be explained in detail how this priority is assigned.

The immediate transitions in the net model the synchronization aspects of the machine. That is, a transition which requires the completion of several activities before it is enabled, thereby initiating a subsequent activity. For instance, in the module A, to enable the transition NCL, i.e., before the execution of a tool change, a placement is to be completed, i.e., the transition ZPLL must have fired. Moreover, for the tool change to start, the left carrier must be also aligned, i.e., the transition LCM must have fired. It can be noted that the firing of transition ZPLL requires in turn the completion of tweezing, i.e., the firing of transition TWL, and the board alignment (a token in place BR).

The Petri net model, as it will be illustrated in the section devoted to the computational results, allows one to compute delays due to carrier or board movement. Inhibitor arcs are used to model some aspects of the control of the machine. The reader is referred also to [1] where inhibitor arcs are used to represent the control logic of an AGV system. A transition linked by an inhibitor arc to one of its input places cannot fire as long as there is a token in that place. For a timed transition, if the inhibitor arc is activated after the transition has been enabled, the time elapsed between enabling and inhibiting is, in this case, accounted for by updating the time required for this transition once it becomes enabled again. In the net in Fig. 3, the tool change cannot start as long as the carriers and the board are moving to align themselves. They must be frozen for the tool change to take place. An inhibitor arc connects the place TL and transitions BM and RCM to effect the freezing.

IV. DATA COLLECTION AND MODEL VALIDATION

In the previous section, we addressed control logic. In this section, it will be explained how to translate the behavior of a machine by the execution of the net.

Before the net can be executed, an initial marking $M^0$ has to be provided. This initial marking depends on the set of operations included in the initial load sequence. This typically includes the carrier and board move times to the first location. So, at the beginning, the simulation clock is set equal to the load time and 5 tokens are in the net, in the places AR, BR, L, and the places CS for both the right and left sectors. The net is now in its initial marking and the simulations can start with the first module to be executed.

As it has been already mentioned in the previous section, this marking results in a conflict among transitions A, B, C, D, and E. In order to solve this conflict, a priority is given to each of these transitions in such a way that only one transition at a time, representing the selection of the module to be executed next, can fire in this particular marking. The priority of firing assigned to each of the conflicting transitions is represented by a Boolean value $A_i$, $i = 1, \ldots, 5$, such that $A_{i+13} A_i = 1$. This means that only the transition having priority 1 can fire. The resolution of this conflict, and of the same conflict arising when the arm is aligned right, requires that a data stream of the respective priority values be provided for each transition involved.

During the simulation of the net, a value is assigned to the transitions A, B, C, D, and E, as soon as they are enabled, according to the next entry in the corresponding data stream. The data streams are implemented by dynamic lists in which a pointer returns the next value. The feature of automatically handling data streams associated with the transitions of a Petri net is allowed by the software tool MEGAS [10].

MEGAS handles the data-dependent time transitions of the net which model operations (such as BM, LCM, and RCM) in this same manner. For example, the time associated with transition BM depends on the previous place location on the workboard relative to the new place location. For this reason, a data stream containing all the board move time required is the values in the data stream are generated by a board move time function

$$b = \max \left\{ \frac{|X_p - X_c|}{s}, \frac{|Y_p - Y_c|}{s} \right\}$$

where $X_p$ and $X_c$ are the X positions of place location related, respectively, to the component previously placed and the component to be placed, $Y_p$ and $Y_c$ are the Y positions of place location related, respectively, to the component previously placed and the component to be placed, and $s$ is the average board speed in millimeters per second. The times associated with transitions LCM and RCM depend on the feeder station from which the previous component was acquired relative to the feeder station from which the current component is to be acquired. Two data streams containing all the left and right carriers move times, respectively, are required. The values in these data streams are generated by a carrier move time function

$$c = \mu + \frac{d}{v}$$

where $d$ is the distance traveled by the carrier in millimeters, $v$ is the average carrier speed in millimeters per second, and $\mu$ is a "settle" time which is zero when $d = 0$ or when the movement is to a tool magazine.

The last data stream is related to the component centering activity or tweezing and is associated with transitions TWL and TWR. The tweezing time is either 0.26 s for standard and 0.51 s when alternate tweezing is required. All the other timed transitions in Fig. 3 have a
fixed time. Once enabled, they can fire after the associated fixed time has been accounted for. This is the time required to complete the corresponding activity, e.g., z-actions (0.34 s) and arm movements (0.265 s).

As mentioned in the Introduction, Petri net models can be used, in the context of model validation, also to analyze the system for some logical properties. Of particular importance are boundedness, i.e., there is an upper bound on the number of tokens in each place, and liveness, i.e., the system is deadlock-free.

The first property is related to the analysis of the place invariants, i.e., positive integer solutions of the linear homogeneous system $C^t X = 0$ where $C$ is the incidence matrix $C = [c_{ij}], i = 1, \cdots, n, j = 1, \cdots, m$, and $c_{ij} = -1$ if $p_i$ is input to $t_j$, $c_{ij} = 0$ if no arc links $p_i$ to $t_j$, $c_{ij} = 1$ if $p_i$ is output to $t_j$. The net is bounded if there exists one place invariant strictly positive.

For the net of Fig. 3, one such place invariant is the vector $(1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 2, 3, 1, 2, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$, and therefore the net is bounded.

In general, a nonzero component of a place invariant corresponds to the output place of the transition which synchronizes the start of a new activity upon the completion of a number of activities given by the value of the component. As far as liveness is concerned, this
TABLE I
FIRING SEQUENCE (STANDARD SCENARIO)

<table>
<thead>
<tr>
<th>Simulated Time</th>
<th>Transition</th>
<th>Delta</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>74.523</td>
<td>AM</td>
<td>0.265</td>
<td>0.265</td>
</tr>
<tr>
<td>74.669</td>
<td>BM</td>
<td>0.146</td>
<td>0.411</td>
</tr>
<tr>
<td>74.783</td>
<td>TWL</td>
<td>0.114</td>
<td>0.260</td>
</tr>
<tr>
<td>74.802</td>
<td>RCM*</td>
<td>0.019</td>
<td>1.39</td>
</tr>
<tr>
<td>74.863</td>
<td>ZPKL</td>
<td>0.061</td>
<td>0.340</td>
</tr>
<tr>
<td>75.123</td>
<td>ZPLL</td>
<td>0.260</td>
<td>0.340</td>
</tr>
</tbody>
</table>

* At the start of the cycle considered in the table, RCM has been already enabled for 0.865 s.

TABLE II
FIRING SEQUENCE (BOARD DELAY)

<table>
<thead>
<tr>
<th>Simulated Time</th>
<th>Transition</th>
<th>Delta</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>105.201</td>
<td>AM</td>
<td>0.265</td>
<td>0.265</td>
</tr>
<tr>
<td>105.461</td>
<td>TWL</td>
<td>0.260</td>
<td>0.260</td>
</tr>
<tr>
<td>105.541</td>
<td>ZPKL</td>
<td>0.080</td>
<td>0.340</td>
</tr>
<tr>
<td>105.624</td>
<td>RCM*</td>
<td>0.083</td>
<td>1.533</td>
</tr>
<tr>
<td>105.639</td>
<td>BM</td>
<td>0.015</td>
<td>0.703</td>
</tr>
<tr>
<td>105.979</td>
<td>ZPPLL</td>
<td>0.340</td>
<td>0.340</td>
</tr>
</tbody>
</table>

* At the start of the cycle considered in the table, RCM has been already enabled for 0.865 s.

TABLE III
FIRING SEQUENCE (NOZZLE CHANGE)

<table>
<thead>
<tr>
<th>Simulated Time</th>
<th>Transition</th>
<th>Delta</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>257.174</td>
<td>AM</td>
<td>0.265</td>
<td>0.265</td>
</tr>
<tr>
<td>257.203</td>
<td>BM</td>
<td>0.029</td>
<td>0.294</td>
</tr>
<tr>
<td>257.309</td>
<td>RCM*</td>
<td>0.106</td>
<td>0.971</td>
</tr>
<tr>
<td>257.434</td>
<td>TWL</td>
<td>0.125</td>
<td>0.260</td>
</tr>
<tr>
<td>257.774</td>
<td>ZPPLL</td>
<td>0.340</td>
<td>0.340</td>
</tr>
<tr>
<td>260.574</td>
<td>NCL</td>
<td>2.800</td>
<td>2.800</td>
</tr>
<tr>
<td>260.853</td>
<td>BM</td>
<td>0.279</td>
<td>0.279</td>
</tr>
<tr>
<td>262.555</td>
<td>LCM</td>
<td>1.702</td>
<td>1.981</td>
</tr>
<tr>
<td>262.895</td>
<td>ZPPLL</td>
<td>0.340</td>
<td>0.340</td>
</tr>
</tbody>
</table>

* At the start of the cycle considered in the table, RCM has been already enabled for 0.865 s.

analysis can be performed using transition invariants, i.e., integer solutions of the equation CX = 0 or exploring the graph whose nodes are the markings of the net (reachability graph). Absence of deadlocks is equivalent to the fact that from each node there is a path to any other node. This can be checked in O(s²) by the Warshall algorithm, where s is the number of nodes. This technique is not always feasible due to the large number of markings. In the case of the net of Fig. 3 the number of states is 312 and the exploration was clearly feasible. The Petri net is deadlock-free.

It is important to remark that liveness also implies reinitiatability of the system.

V. NUMERICAL RESULTS

The Petri net based emulator of the machine has been tested with two different products, referred to as P1 and P2. P1 and P2 require 132 and 284 placements, respectively. They require different component types and thus the component delivery carrier setup changes as does the sequence of required placements. In particular, the placement sequence of P1 has two imbalances.

Once all the data required for timing the transitions according to the machine operations related to P1 and P2 were specified via the data streams, MEGAS was used to execute the net. The completion time for P1 is 135.545 s and for P2 283.195 s. If there is no delay due to board movement, carrier movement, and alternate tweezing, each cycle of module C would take 0.865 s given by the arm movement, z-action, and standard tweezing. During the execution of the net it is possible to compare the emulation time of the machine after the completion of each cycle, i.e., when a token is in place AS. The comparison between the numerical results obtained by DPRT and MEGAS showed agreement. Moreover, the run time of these two emulators is of the same order of magnitude, a few seconds for P1 and P2.

The Petri net based emulator also permits one to identify the blocking processes or transitions. That is, the operations which cause potentially avoidable delays. Therefore, one can interactively analyze the sequence of firings for the product under consideration.

As an illustration, consider the firing sequences in the following tables. These sequences are obtained via the procedure SIMUL explained in [4]. Table I displays the firing sequence between completion of placement 70 at the simulated time 74.258 s and completion of placement 71 according to module C in the case of no delays (standard scenario). The value "Delta" denotes the time elapsed between the firing of the transition in the same row and the firing of the transition in the previous row. Time is the duration of the activity associated with that transition.

The cycle time, as remarked above, is 0.865 = 75.123 - 74.258, given by the sum of the times of the three operations AM, TWL, and ZPPLL done serially. Other operations are performed concurrently with each of the above operations. For instance, from the table, one can understand that BM is concurrent with AM, which is completed before BM fires. Indeed only 0.146 s, after the arm move is completed, must elapse for the completion of the board movement which requires 0.411 s. TWL is concurrent with BM which is completed before TWL fires. Indeed adding the corresponding "Delta" values of 0.146 and 0.114 one gets the time required for the transition TWL to fire. Analogously, one can analyze ZPPLL, enabled by the firing of TWL and BM, and its concurrent transitions.

The delays occurring in processing P1 and P2 are caused by carrier movements required to align the tool magazine for a nozzle change and also by board movements and alternate tweezing. A board move delay occurs when the time required to align the board from the current position to the next one happens to be larger than the time required to complete the other concurrent operations. For example, Table II displays the firing sequence between completion of placement 101 at the simulated time 104.936 s and completion of placement 102 for P1 again according to module C, but now a board delay occurs. In this case, the cycle time is 1.043 = 105.979 - 104.936, given by the sum of the times of the three operations AM, TWL, and ZPPLL done serially plus an addition time (0.178 s) caused by the board move time. In fact, transition ZPPLL, concurrent with transition ZPPLL in the standard scenario, has to wait for the board to be aligned before firing. This wait time is given by the sum of the "Delta" values of the three transitions ZPPLL, RCM, and BM. Note that the "Delta" value of transition ZPPLL is equal to its time. This means that no other previously completed transitions were concurrent and that ZPPLL was enabled after the completion of all the other concurrent operations performed in this cycle.

As far as P2 is concerned, the analysis performed by MEGAS shows that the delays are not due to board movements but rather to the carrier movements, particularly those between the tool magazine and the feeder station. As an example, consider Table III. It displays the firing sequence between completion of placement 264 at the
simulated time 256.909 s and completion of placement 265 for P2 in which a nozzle change is required (module A) and a carrier delay from the tool magazine to the feeder station occurs. It is important to remark that the time required to complete a cycle in the case of module A is 4.005 s, given by the sum of the time of the transitions AM, TWW, ZPIL, NCL, and ZPKL done serially, plus an additional time T due to the carrier movement from the tool magazine to the feeder station. In fact, the carrier is released only when the nozzle change operation is completed and transition ZPKL has to wait for the carrier to be aligned before firing.

From Table III one can see that the time in order to complete this particular placement is 5.986 = 262.895 − 256.909 and T is 1.981 s. It has to be noted that the left carrier is already aligned to the tool magazine at the beginning of this cycle. One can see from the table that the only concurrent operations, in case of nozzle change, are the board movement with the arm movement and subsequently with the standard tweezing. All the other operations are performed serially: indeed, their “Delta” values are the same as their times.

VI. CONCLUSIONS

In the authors’ opinion the problem dealt with in this communication is a realistic application of Petri nets to a problem of industrial relevance. The following points have been also shown to hold:

1) The features of concurrency and synchronization which are crucial to the actual performance of the machine can be naturally represented in the model and in the stepwise analysis of the simulation.

2) No specific emulation program has to be written. All the input required is contained in the net.

3) Structural properties, besides the usual performance indices, can be derived.

REFERENCES


