[16] E. Zussman and M. C. Zhou, "Design and implementation of an adaptive process planner for disassembly processes," *IEEE Trans. Robot. Automat.*, vol. 16, no. 2, pp. 171–179, Apr. 2000.

Modeling, Scheduling, and Performance Evaluation for Wafer Fabrication: A Queueing Colored Petri-Net and GA-Based Approach

Tsung-Che Chiang, An-Chih Huang, and Li-Chen Fu

***Abstract*—In this paper, we propose a modeling tool named Queueing Colored Petri nets (QCPN) for performance evaluation and scheduling for wafer fabrication. The main idea of this tool is to combine colored timed Petri nets with the queueing systems, and it aims to make simulation over the model more efficient. Due to the wide acceptance of priority rules in the wafer manufacturing industry, we also proposed a mechanism to realize priority rules in the QCPN models. Since it is known that no single rule can dominate in any circumstance, we proposed a genetic algorithm (GA) to search for the optimal combination of a number of priority rules based on the status and performance measures of the fab. Our approach can be considered as taking the advantage of the lot execution sequence generated by priority rules to guide the search. This approach can reduce the solution space and help us find the good solution more quickly. In addition, the QCPN-based GA scheduler can greatly reduce the computation time so that this GA scheduler can meet the need for a rapidly changing environment.**

***Note to Practitioners*—Performance evaluation and scheduling are two functions required by fab managers and engineers. This paper proposed a tool which consists of a simulator and a scheduler. By connecting to the Manufacturing Execution System (MES) and providing the scheduling rules, we can see how the fab runs virtually with the simulator. General information such as throughput and average cycle time and specific information like lot activity history can be obtained. This can be used for decision making, delivery prediction, bottleneck seeking, and testing of newly developed heurisitcs. The implementation cost is only on data communication between the MES and the simulator and the incorporation of rule modules. The scheduler, which takes the simulator as the performance evaluation module, can generate the suitable scheduling rule based on the current fab status, preference of performance criteria, and rule candidates. There is almost no extra cost after the simulator is connected to the MES. The scheduler can be easily made faster by common parallelization techniques.**

***Index Terms*—Genetic algorithms (GAs), Petri nets, queueing systems, semiconductor manufacturing scheduling.**

I. Introduction

In recent years, the semiconductor manufacturing industry has grown rapidly and it has become one of the most competitive fields. Due to equipment uncertainty, product diversity, process intricacy, and ever-improving technologies, the semiconductor manufacturing is perhaps the most complicated manufacturing process today. In such a competitive market, wafer fab managers must try hard to reduce the cycle time, improve quality, and to lower production cost. In addition, they have to respond quickly to orders and to make the orders meet due dates. If they fail to do so, they will lose customers' goodwill as well as a long-term sales opportunity.

This paper addresses the problem of performance evaluation and scheduling in the wafer manufacturing industry. If we want to meet the requirements of customers, we have to precisely estimate the cycle time of all orders as well as other performance measures, such as work-in-process (WIP) and machine utilization first. Based on these measures, the manager or fab engineers can control the operations and the manufacturing times easily, and customers can know the status of their goods. Thus, performance evaluation is an important and challenging task.

Petri net has played an important role in the modeling field for a long time. It has been developed into a powerful tool for modeling discrete event systems, particularly for manufacturing systems. Many extensions of Petri nets were introduced. For example, colored Petri nets [2]–[4] and timed Petri nets were widely used in manufacturing systems. In [2] and [4], the CTPN was used to model the furnace and the whole wafer fabrication manufacturing system. Zhou *et al.* [5] reviewed applications of PNs in semiconductor manufacturing automation. Jeng *et al.* [6] reported a project of applying Petri net methodologies to detailed modeling, qualitative analysis, and performance evaluation of the etching area in an integrated-circuit (IC) wafer fabrication system. Due to the strong expressiveness, the CTPN is taken as the basis for modeling the complex semiconductor manufacturing systems in our approach.

Successful examples of combination of Petri nets and queueing systems can be found in the literature [7]–[11]. In [7], Balbo *et al.* constructed a model for concurrent execution and synchronization of tasks. The part of blocking and forking was modeled by Petri nets and the part of the server subsystem was simplified by solving its corresponding queueing network model to get a flow-equivalent server. The focus of this work and some other similar works is on the superiority of modeling ability of the combination of Petri nets and queueing systems over the Markov chain approach. Bause [8] proposed a formalism which combines Petri nets and queueing networks. He emphasized the ease of describing scheduling strategies (e.g., first in first out (FIFO) and process sharing) with queueing networks. Becker *et al.* [9] also proposed a modeling tool, Petri Nets including queueing networks (PNiQ), which generalized the idea in [7] and gave a formal definition. This tool was then extended in [10] and an application was shown in [11]. In these works, the idea of combination of Petri nets and queueing systems was demonstrated only with small systems except in [11]. However, even in [11], there was not a complete and clear model for describing a complex manufacturing system with a rule-based flow controller. Moreover, there was no mention of applications like scheduling. These two points will be addressed in this paper.

There were many papers on planning and scheduling in semiconductor fabrication. Recent papers by Johri [12], Duenyas *et al.* [13], and Uzsoy *et al.* [14], [15] highlighted the difficulties in planning and scheduling of wafer fabrication facilities. A practical solution to scheduling such a complex system is by using priority rules [16], [17], [25]–[30] because of satisfactory performance, small computation requirement, ease of implementation, and flexibility to incorporate domain knowledge and experience. To enhance the performance of priority rules, in this paper, we propose a genetic-algorithm (GA)-based framework to set a suitable combination of priority rules for different types of products and equipment so as to explore the features of rules according to performance criteria, equipment characteristics, and fab

Manuscript received March 27, 2005; revised May 30, 2005 and July 21, 2005. This work was supported by the National Science Council of Taiwan, R.O.C., under Research Grant NSC 94-2752-E-002-007-PAE. This paper was recommended for publication by Associate Editor M. Zhou and Editor N. Viswanadham upon evaluation of the reviewers' comments.

The authors are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: lichen@ntu.edu.tw; r88008@csie.ntu.edu.tw; r88054@csie.ntu.edu.tw).

Digital Object Identifier 10.1109/TASE.2005.862198

status. In this framework, the proposed QCPN model will serve as a platform upon which efficient performance evaluation of chromosomes in GAs can be achieved. The GA is used since its performance was proven by many applications on production scheduling [20]–[24].

The organization of this paper is described as follows. Section II is about the proposed QCPN. It shows how we introduce the queueing theory into the original CTPN and why the integration improves efficiency. The model of equipment with QCPN is provided in Section III. In Section IV, we describe how we realize priority rules in the QCPN model for simulation and performance evaluation. Section V gives the details of the proposed GA for searching a suitable combination of priority rules. In Section VI, we demonstrate the application of using the proposed mechanism and analyze the performance. Finally, conclusions are given in Section VII.

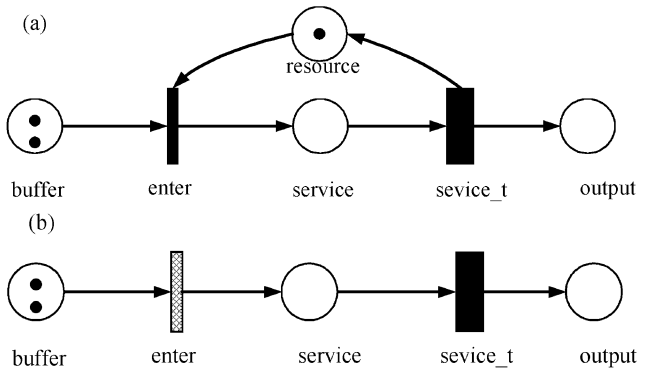

Fig. 1. (a) Original resource contention. (b) QCPN resource contention.

II. Queueing Colored PNs

In our method of combining PNs and queueing systems, we introduced a special kind of transition named the queueing transition into the CTPN to represent the role of the queueing system. The modeling power of CTPN is exploited to elegantly model processing capability and routing while the queueing system is used to simplify the model and accelerate the execution. For the definitions of CTPN, readers are directed to our previous work [26].

A. Queueing Transition

Queueing transitions are used to model the waiting/response time of parts in a manufacturing system. An enabled queueing transition represents that some parts are in the service queue, and the firing represents that the part leaves. Each queueing transition is embedded with a priority queueing model, and the firing time can be calculated by the model. A token that enables the queueing transition will be assigned a queueing time. After the time delay, this transition can be fired.

In Fig. 1(a), a common CTPN model for resource contention is given. The token in the place resource is initially marked to represent that the machine is idle at the beginning. When there are tokens in the place buffer, one will fire the immediate transition enter and mark the place service. The transition service_t is a timed transition. After the delay time elapses, the machine releases the part, and the place resource is marked again. Another token waiting in place buffer can fire through the transition enter.

With the proposed QCPN, this resource contention subset is modeled in Fig. 1(b). In this figure, the place resource in Fig. 1(a) is removed and is replaced by a queueing transition enter. The time attribute of the queueing transition represents the waiting time of a token in the place buffer. As mentioned above, this time delay is calculated by the embedded priority queueing model of the queueing transition.

The order in which tokens in the place buffer fire the transition enter in Fig. 1(a) actually represents the processing sequence of parts on the machine. In the industry, this sequence is typically determined by applying priority rules. It means that when the transition enter is enabled, a priority value should be calculated for each token in buffer. It takes computation time proportional to the number of tokens. The priority values are calculated by rules based on the status of queued lots, and the status changes with time. Thus, these priority values are dynamic in time, and the calculation should be done each time the transition enter is enabled. Suppose there are n parts in buffer and no new lot comes, the time required for solving resource contention for these n lots is $n \cdot t + (n-1) \cdot t + \cdots + 2 \cdot t = \theta(n^2) \cdot t$ where t is the time used to calculate the priority value for a token.

However, the processing sequence in the model in Fig. 1(b) is determined by the time delays of firings of the queueing transition enter. The token with higher priority will have a shorter time delay and, thus, will pass through the resource earlier than those with lower priority and longer time delay. In other words, the condition that a lower-priority token is blocked from processing by the occupation of resource by a higher-priority token is realized by the queueing time delay. The queueing time delay of a token represents the sum of processing times of tokens that block its processing.

For example, a token i enters the place buffer and there is one token j already in buffer. The resource is busy. Assume token i has a lower priority value than token j according to the adopted priority rule. With the model in Fig. 1(a), token j is the next one to fire the transition enter after the resource is released. The token i must wait until the resource is released again when token j fires the transition service_t. This is how the processing sequence of lots is realized in the model in Fig. 1(a). However, in the model in Fig. 1(b), the token i will be given a time delay to fire the transition enter when it enters the place buffer. This time delay is an estimate of the sum of the time required for the resource to finish the current in-process lot and the processing time of token j. In this way, the processing sequence in which j is earlier than i is realized by firing the transition enter with the estimated time delay.

By using the embedded priority queueing model, the time delays can be determined at the time when the token arrives at the place buffer, and will not change thereafter. Accordingly, given n parts in buffer, the time required for solving resource contention is only $\theta(n) \cdot t$ where t is the time used to calculate the time delay for a token. This reduction of time complexity from $\theta(n^2)$ to $\theta(n)$ is the main reason why simulation over the QCPN model is much more efficient than that over the CTPN model.

B. Firing Time of Queueing Transitions

As mentioned earlier, the firing time of the queueing transition represents the time taken by a part to wait for starting its processing on a machine. To realize the priority rules, we classify parts into several classes based on their priority values, and then adopt the G/G/1/priority model to calculate the firing times of queueing transitions. The mean waiting time of parts in class p (class 1 has the lowest priority, and there are a total of P classes) can be approximated by incorporating an adjustment factor j into the M/G/1/priority model [1]

$$W_p = \frac{\sum_{i=1}^{P} \frac{\lambda_i \overline{x_i^2} j_i}{2}}{(1-\sigma_p)(1-\sigma_{p+1})} \tag{1}$$

where $j_i \equiv (c_{a,i}^2 + c_{s,i}^2)/(c_{s,i}^2 + 1)$, $c_{a,i}^2$ and $c_{s,i}^2$ are the squares of variation coefficients of the interarrival and processing times of the parts in class i and $\sigma_p = \sum_{i=p}^{P} \rho_i$, $\rho_i = \lambda_i \overline{x_i}$. The parameters $c_{a,i}^2$ and $c_{s,i}^2$ can be obtained from the historical data of the wafer fab.

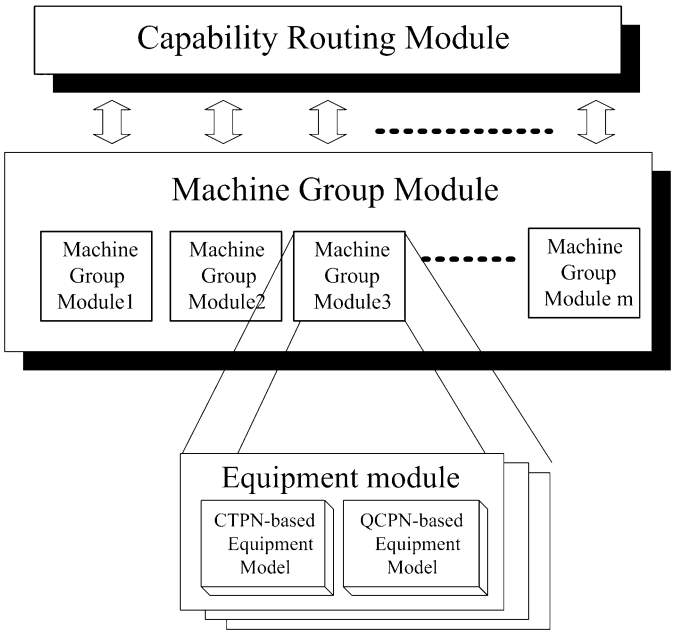

Fig. 2. Conceptual architecture of the wafer fab model.

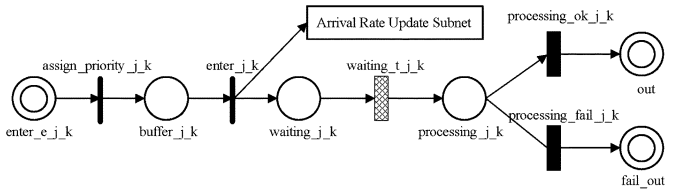

Fig. 3. QCPN-based equipment module.

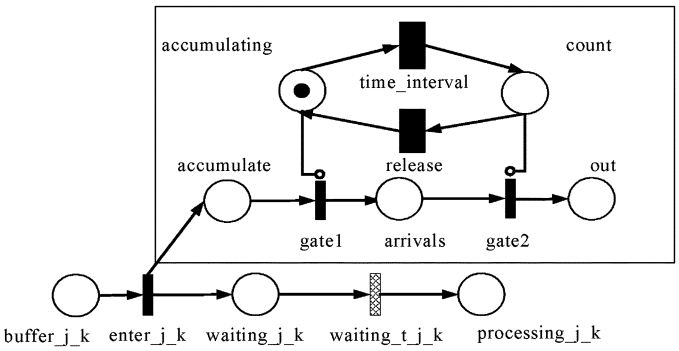

Fig. 4. Arrival rate update subnet.

III. Wafer Fab Model

A. Wafer Fab Model Based on QCPN

A wafer fab can be considered as a job shop containing a large number of machines. The number of machines differs widely and depends on the size of the fab, and normally ranges from 50 to 500. Wafers are grouped in lots, which travel together in a standard container and are destined to become the same final product. Each lot has an associated process flow that consists of precisely specified operations to be executed in a prescribed sequence. Each process step denotes the processing capability that the operation requires. We use the proposed QCPN to model the whole wafer manufacturing system. The wafer processing model developed here is a general model, and will not be restricted to only some specific processes. When the information of equipment and route is given, the wafer processing model is automatically generated by the model generator. Different process flows of different products can be performed based on this model. The model includes the capability routing module, machine group module and equipment module. Fig. 2 illustrates the conceptual architecture of the proposed model, which shows the relationship among all of these modules. With the limitation of space, only the equipment module is given. Details of the complete model can be found in [27].

B. Equipment Module

The equipment module constructed by QCPN is illustrated in Fig. 3. After the transition enter_j_k is fired, two tokens are generated. One will be sent to the place waiting_j_k for normal operations, and the other will be sent to the arrival rate update subnet, which will be described later. After processing, the lot may enter the communication place[1] out or fail_out and continue the next processing step.

The machine failure and periodical maintenance are not modeled explicitly here. Such occasional events are modeled as dummy lots that directly arrive at the highest priority queue [1] in the priority queueing model [e.g., The arrival rate of the event of machine failure is the reciprocal of mean time between failure (MTBF), and the service time is mean time to repair (MTTR)].

The notation used in Fig. 3 is described below.

[1]Communication places are places common in more than one module.

Assign_Priority_j_k: A token going through this transition will change its corresponding color of priority value according to the dispatching rule adopted. The output function of this transition varies with time since the calculated priority value can dynamically change with the online situation.

Buffer_j_k: A token in this place represents that the lot is currently in the buffer of the kth machine and is ready to be processed.

Waiting_j_k: A token in this place represents a lot is waiting in the queue of the machine.

Waiting_t_j_k: The transition *waiting_t_j_k* is a queueing transition, and takes some time to fire. The time is calculated using the priority queueing model of the queueing transition, and it represents the waiting time (in queue) of the lots.

Processing_j_k: A token in *processing_j_k* represents that the corresponding lot is in process on the machine k in the jth machine group. The token can enable the transitions *process_ok_j_k* and *process_fail_j_k*, and one of them is fired according to the rework probability set in this place. The rework probability is greater than zero only if the machine contains an inspection device.

Process_ok_j_k: The time delay (processing time) of this transition depends on the current route and step of the lot (token). The firing of *process_ok_j_k* represents that the lot has finished the current operation successfully and will be sent back to the capability routing module through the communication place *out*.

Process_fail_j_k: The time delay (processing time) is determined in the same way as *process_ok_j_k* does. When a token is passed to the place *fail_out*, one token will be generated to indicate that the lot must be reworked from the specific operation step.

For batch processing machines, we adopt the full-batch control policy and use the same model as what we proposed in the previous work [26].

C. Arrival Rate Update Subnet

Since the calculation of waiting time of lots heavily depends on their arrival rates, we provide a mechanism to periodically update the arrival rates of lots to each machine in order to obtain accurate results. We use an additional subnet to achieve our goal, and it is illustrated in Fig. 4. In the equipment module, a token firing the transition enter_j_k will produce a token entering the arrival rate update subnet. In this subnet, it accumulates the tokens (lots) arrived in a predefined time interval.

After the number of tokens is obtained, the arrival rate can be calculated by

$$\lambda(\text{arrival rate of an interval}) = \frac{\text{The number of token arrivals}}{\text{Time interval}}.$$

The detailed description of this subnet is given as follows.

Accumulate: The token arrived will be temporarily put into this place.

Accumulating: The place accumulating is initially marked with one token, and a token in this place represents that the tokens are being accumulated. After the token here is removed, tokens in accumulate are sent to the place arrivals.

Time_Interval: It is a timed transition. The time associated with this transition is a user-defined value. Firing this transition means that it is time to update the rate.

Count: A token in this place represents that the tokens are being counted. Because the token in the place count blocks the transition gate2, the tokens from accumulate will not be sent to the place out directly. At this time, we can count the tokens in arrivals, which represent the lots arrived during this time interval.

Release: It is a timed transition. The delay time associated with this transition is very small. Firing this transition will indirectly move all of the tokens in arrivals to the place *out.* A token will enter accumulating and start the next accumulation.

Note that the selection of the value of the update time interval must be done with care. If the chosen value is too small, the calculated waiting time may be infinite due to the fact that the machine utilization in this time interval is larger than one. On the other hand, if the selected time interval is too long, the precision of the prediction result will be affected. The appropriate time interval can be obtained by some trials.

IV. Priority Class Assigning Method for General Priority Rules

In semiconductor manufacturing systems, the fab managers often adopt priority rules to control the manufacturing processes. If there is more than one lot waiting in the queue when the machine is available, the priority rule calculates a priority value for each lot, and the lot with the highest (or the lowest) value is processed next.

The queueing systems often assume that service requests are served according to some simple rules such as FIFO. However, the priority rules commonly used in the fab such as earliest due date (EDD) and critical ratio (CR) are much more complicated and the exact solutions of the queueing systems based on these rules have not been proposed yet. Therefore, we propose a method to realize the priority rules in the queueing system by adopting the multiclass priority queueing model and developing a procedure to assign lots to different priority classes.

The simplest way to accomplish this assignment is as follows. First, we sort the lots in the fab according to its priority calculated by the dispatching rule. Then, we divide them into P priority classes. The first (highest priority queue) class contains the first $1/P$ portion of lots with the highest priority values; the second class contains the next $1/P$ portion of the lots and so on. With the assigned priority class of each lot, we can use the priority queueing model to calculate the waiting time and model the processing sequences determined by the corresponding rule. Thereafter, we can use the QCPN-based simulator to evaluate the performance of the selected dispatching rule. Fig. 5 illustrates the idea of the priority class assigning method.

Since the priority value of lots may vary from time to time, this method must be executed periodically. That means we have to sort and assign the lots to different classes periodically and, thus, adding computation time to the simulation over the QCPN model. A more efficient way is to keep only the highest and lowest value of lot priorities. The complexity of doing this is only $\mathrm{O}(N)$, where N is the number of lots. It is faster than the original way with time complexity $O(N \cdot \log N)$ from sorting. After the highest and lowest priority values of lots in the fab are found, we can assign the lot to different priority classes according to the following equation:

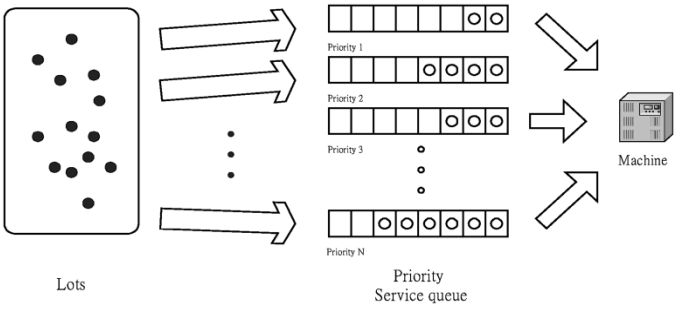

Fig. 5. Priority class assigning method.

$$C = \left\lceil \frac{p - V_{\min}}{V_{\max} - V_{\min}} \times P \right\rceil \ (V_{\max} \geq p \geq V_{\min})$$
$$C = P(p > V_{\max})$$
$$C = 1(p < V_{\min})$$

where P is the number of priority classes, $V_{\max}$ and $V_{\min}$ are the highest and lowest priority values, p is the priority value of the target lot, and C is the priority class to which this lot is assigned. Note that the values of $V_{\max}$ and $V_{\min}$ need to be updated periodically.

In this section, we introduced how we realize the priority rules in the QCPN model so that performance evaluation can be achieved by simulation over the model. With the ability to do performance evaluation, the QCPN-based simulator also serves as the evaluation procedure in the GA optimization process described in the next section.

V. Wafer Fab Scheduling

In a typical semiconductor wafer fab, there are many different products competing for limited resources. Each product flow can include hundreds of processing steps. Because of the long sequences of operations required for the wafers, most semiconductor manufacturing companies suffer from a high WIP level and long cycle times. Hence, it is important to develop a production scheduling system that can help to minimize cycle time and WIP as well as to meet the due-date constraints.

For simplicity, the present work assumes that the stocker capacity between machines is infinite, and the material handling system is not taken into account. However, such assumption does not invalidate the overall idea behind the present work. Instead, it allows us to concentrate on the ingenious integration of a rather concise QCPN and GA for scheduling.

A. Overview of the GA

The GA is a class of algorithms that provides a computation model for the biological evolutionary process following the principles of natural selection and survival of the fittest. By relating the artificial evolutionary process and the solution-finding process, GA is taken as a search algorithm or an optimization approach for decades. In the literature, GA was shown to be highly effective in searching large and complex response surfaces even in the presence of difficulties such as high dimensionality, multimodality, discontinuity, and noise. It is a multipoint, random, guided, and parallel-in-nature algorithm, and these properties contribute to the GAs' recombinative power, ability to avoid

local optima, unsupervised self-learning ability, and ease of parallelization. With its general framework, GAs have been used pervasively for solving many different kinds of scheduling problems, and its good performance was well studied.

To develop a GA, there are four major components:

- chromosome representation of the solutions to be found, which is an encoding mechanism;
- decoding mechanism to interpret chromosomes back to their corresponding solutions;
- evaluation function that rates the chromosomes/solutions in terms of their fitness;
- genetic operators that alter the composition of solutions during a reproduction process.

The details of these components are given in the following subsections.

B. Mixed Rules

The main purpose of the GA here is to find the appropriate lot release rule (for the fab), machine selecting rules (for each product), and dispatching rules (for each machine group). To obtain more effective dispatching rules, we try to mix multiple rules by a linear weighted combination.

Before combining different rules, a normalization procedure should be done since different rules may have different ranges of values. For example, the priority values by the shortest remaining processing time (SRPT) rule may range from zero to the expected cycle time of the lot; the priority values by the EDD rule may range from the value of current time to some time far later. If we transform these values into the same range, the combination will be more reasonable.

Since the priority values of some rules change with time (e.g., CR rule), we must periodically track the range of the priority values, and maintain the highest and lowest priority values to do the transformation. Note that in some rules, the smaller priority value represents the higher priority, so we need to make the value negative. After the highest and lowest values are calculated, the equation for transformation is

$$C = \frac{p - V_{\min}}{V_{\max} - V_{\min}}$$

where $V_{\max}(V_{\min})$ is the largest (smallest) priority value, C is the transformed value, and p is the target priority (the value to be transformed). Then, the priority values of all rules can be transformed into values between 0 and 1.

C. Chromosome Representation

Our approach uses a priority rule-based representation of chromosomes in the GA. It is a kind of indirect representation approach. This approach brings us a number of advantages, such as the simplicity of the chromosome structure, simple GA operators, shorter computation time, etc.

There are three types of genes in our chromosome g_l, g_m and g_d, which denotes lot release policy, machine selecting rule, and dispatching rule, respectively. The contents of g_l and g_m are the indicators for which rules are applied while $g_d = (a, b, c, d, e)$ is a five-tuple where a, b, c, d, and e represent the weights of SRPT, EDD, CR, least slack (LS), and Penalty rules in the linear combination. The weights range from zero to three.

Three lot release policies, including uniform (UNIF), constant WIP (CONWIP), and workload regulation (WR) rules [16], are used as candidates. As for machine selecting rules, there are three machine selecting rules including the lowest utilization (UTIL), shortest queue length (SQL), and shortest expected processing time (SEPT). The definitions of these rules can be found in [27], and other rules can be easily added because of the flexibility of GA.

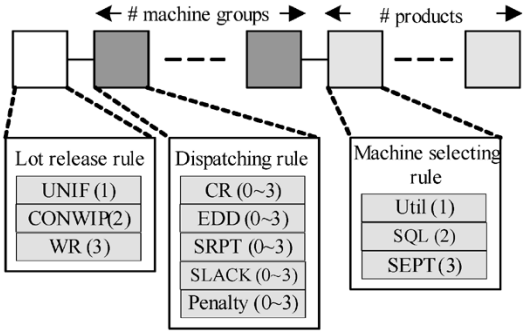

Fig. 6. Chromosome representation.

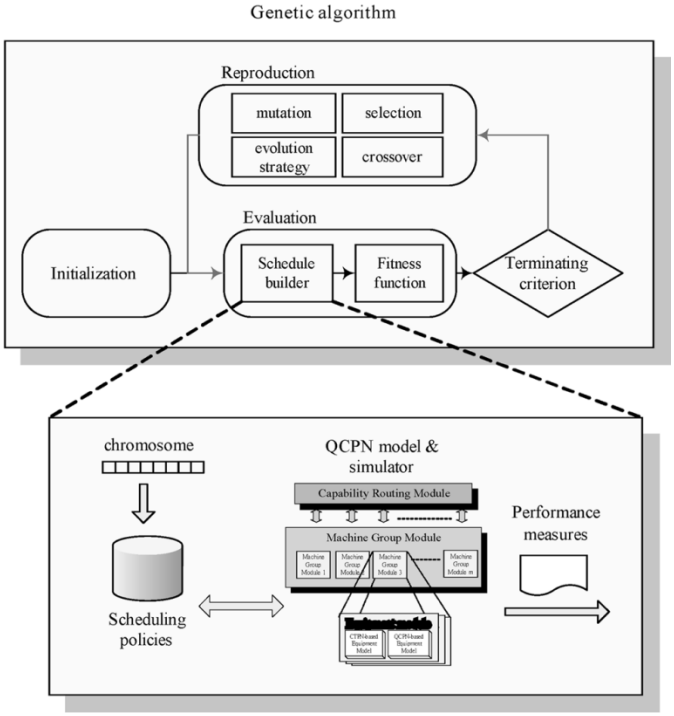

Fig. 7. Architecture of the scheduler.

The length of a chromosome is fixed and the structure of the chromosome is illustrated in Fig. 6. The first gene is the g_l type, followed by M g_d-type genes and, finally, P g_m-type genes where M is the number of machine groups and P is the number of products.

D. Schedule Builder

Based on the QCPN model, the simulation of the system can be addressed by the change of marking in the net. All possible kinds of behaviors of the system can be completely tracked by the reachability graph of the net. In other words, we can track the status of lots and machines from the QCPN model while the schedule is performed.

A schedule builder is dedicated to transforming a chromosome to a feasible schedule so that we can evaluate it. A chromosome consists of many genes, each of which indicates a (mixed) rule which is directly related to the execution of the QCPN model. Once a conflict representing a decision of sequencing or machine selecting occurs in the model, the corresponding (mixed) rule is invoked to resolve it. Thus, given a QCPN model and a chromosome, the schedule builder can generate a feasible schedule guided by rules recorded in the chromosome. Then, the performance measures like mean cycle time can be obtained from the generated schedule, and a fitness function will be used to evaluate this chromosome based on those measures. The architecture of the scheduler is shown in Fig. 7.

E. Fitness Function

In common situations, we have multiple objective measures. One popular method to combine them is the weighted-sum approach

$$f(c) = w_1 \cdot f_1(c) + w_2 \cdot f_2(c) + \cdots + w_n \cdot f_n(c)$$

where c is a chromosome (i.e., solution), $f(c)$ is the fitness function, $f_i(c)$ is the ith objective function, w_i is a constant weight for $f_i(c)$, and n is the number of the objective functions.

In this paper, we consider three criteria including mean cycle time, throughput rates, and average penalties resulting from tardy jobs. The associated weights are open to users. The reason for choosing these three criteria is that they are the factors that fabs concern the most according to previous experience. Theoretically, there is no limit to the number of criteria when forming the fitness function. Three criteria are sufficient to demonstrate the hereby proposed idea.

F. Reproduction

Classic operators including roulette wheel selection, one-point crossover, and one-gene-substitution mutation operators are used. Let N be the population size and r be the crossover rate; the fittest N chromosomes among the N parents and $N \cdot r$ offspring will survive to the next generation.

G. Cutoff Method by Machine Utilization

In the proposed method, each machine group can adopt a mixed dispatching rule. However, it is reasonable that the selected dispatching rule of the bottleneck machine is more important than that of other machines. Since the bottleneck machine is often determined by machine utilization, we use the utilization of each machine group to determine its importance. Another reason for using machine utilization as an index is that lower utilization represents a longer idle time. For example, a machine group with a utilization of 50% represents that the queue of this machine group is empty in half of the total system time. Thus, we do not need to select among lots but just wait for the next lot's arrival.

The cutoff method simply means to "cutoff" the machine groups with lower utilization in the chromosome. These machine groups will use the same dispatching rule that is also obtained from the GA scheduler. The chromosome representation is illustrated in Fig. 8. Only the machine groups with high utilization are separately listed in the chromosome. Reducing the length of the chromosome by cutting off light-loaded machines is expected to effectively reduce the search space of the GA with little sacrifice of performance.

Selecting the appropriate cutoff level requires some experimental procedures. With a higher cutoff level, the search space is smaller and, hence, the convergence speed is fast, but may be accompanied with the cost of losing some good solutions. The testing of different cutoff levels is conducted in the experiments.

VI. Experiments and Results

The model used in our experiments is built based on data from the semiconductor manufacturing companies located in Hsin Chu Science-Based Industrial Park. In this model, there are totally 477 machines in 43 machine groups. The number of machines in each machine group may range from 3 to 45. There are two production lines in this model. The first production line is a re-entrant line comprising 416 operations, and the second production line comprises 322 operations.

In order to demonstrate the rework feature in production flows, we randomly select some steps in these flows and set the rework probability. We also create some rework flows from some segments in these production flows. After the lot finishes these steps, the lot may be sent to the rework flow with the predefined rework probability. After a lot finishes its process on the rework flow, it can rejoin the normal flow and continue its production. Besides, we assume the re-entrance of operations on the same machine may have different processing times since they have different recipes to be processed.

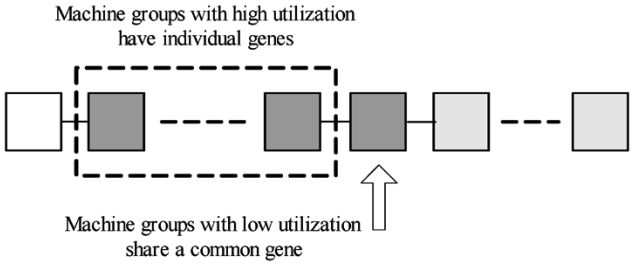

Fig. 8. Chromosome representation after cutting off.

In this model, we also assume that some machine groups contain batch-processing machines. We select the machine groups with long processing times (e.g., $>$ 3 h) since the operations on batching machines, such as oxidation, usually take several hours. The parameters, such as MTBF, MTTR, and scheduled maintenance are also included in this model. Scheduled maintenance means that the machine will be stopped to be maintained after a fixed number of operations.

A. Implementation

For the purpose of performance evaluation and testing the proposed method, we developed a tool named NTU Sim. This tool consists of three parts. The first two parts are two simulators based on CTPN and QCPN, respectively. Both simulators provide functions, such as performance evaluation and decision support. In the experiments, the CTPN-based simulator is taken to verify the accuracy and to show the efficiency of the QCPN-based simulator. The difference between these two simulators is that the CTPN-based simulator uses the equipment module proposed in our previous work [26], in which resource contention is modeled in the way as shown in Fig. 1(a) in Section II.

The third part of this tool is the GA scheduler. Before running the scheduler, we have to fill in the parameters, such as the generation size, population size, and the crossover rate, etc. Then, the importance (weight) of each performance criterion also needs to be defined. After all of these are done, we can use the scheduler to find out appropriate scheduling policies and, of course, the corresponding high-quality schedules.

B. Experiment Results

In this subsection, we make some experiments to evaluate the performance of our simulator and scheduler. In these experiments, the fab is not empty at the beginning. It means that there are lots already being processed in the fab. We generate the information of these lots by taking a snapshot of the simulation model in the steady-state. The period of evaluation is 3000 h (about 4–5 months). The parameters of three release policies UNIF, CONWIP, and WR are set as 7.5 h, 150 lots, and 10 h, respectively. The values are set so as to keep the utilization rate of the bottleneck machine around 90%. Our experiments are performed on the mixed production lines, and the proportion of two types of lots is 1:1. Finally, all of the experiments were done on the PC with a Pentium II 450-MHz central processing unit (CPU), 128-MB random-access memory (RAM), and the Windows NT system.

1) Accuracy of the QCPN-Based Simulator: In this experiment, we arbitrarily chose three orders and compared their delivery dates obtained from the QCPN-based simulator with the results from the CTPN-based one. In order to verify the accuracy of the simulator on different

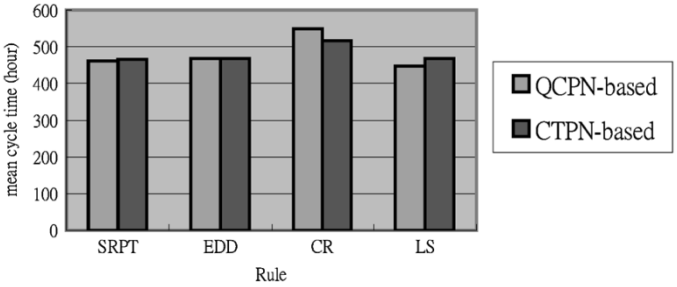

Fig. 9. Mean cycle time by two simulators.

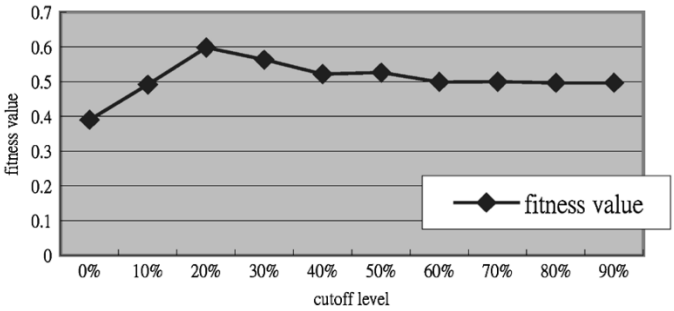

Fig. 10. Mean best fitness values on different cutoff levels.

TABLE I
COMPARISON OF QCPN-GA AND OTHER HEURISTICS

	SRPT	EDD	CR	LS	FCFS	GA
Mean CT	482	500	595	507	565.5	483
Thr. Rate	147	146.5	143.8	146.9	142.3	148.5
Penalty	0.98	0.41	1.9	0.16	1.74	0.15

TABLE II
RUNNING TIME OF CTPN-BASED AND QCPN-BASED GA

	CTPN based GA (min)	QCPN based GA (min)
Mean	20.75	2.98

dispatch rules, four rules SRPT, EDD, CR, and LS are chosen for comparison. The accuracy of prediction ranges from 90% to 98%, which is quite satisfactory.

In Fig. 9, we show the mean cycle time obtained by two simulators. The result represents that the performance measures based on the QCPN model are very close to those based on the CTPN model. The comparisons of other performance criteria show the same result. In other words, the process flows in the fab are correctly modeled by the proposed QCPN model.

2) Performance of the GA Scheduler: In this subsection, we first compare the performance measures of our scheduler with other priority-based heuristic rules. The interval of order release is 7.5 h. The due date of the orders is the order release time plus 640 h, and the penalty is randomly generated from 0 to 10. The experimental results are given in Table I. The GA parameters we use to search for a better chromosome are listed as follows:

population size 10;
generation size 10;
crossover rate 0.7;
mutation probability 0.1;
weight of cycle time 3;
weight of throughput rate 1;
weight of penalty 3;
cutoff utilization level 50%.

Then, we compared the computation times of the GA running over the CTPN-based and the QCPN-based simulators. We have run each combination ten times. The results are listed in Table II.

On the other hand, in order to verify the effect of the cutoff method, we did several experiments on different cutoff levels. The experiment was repeated for 100 times on each cutoff level, and we calculated the mean best fitness of these 100 trials. Fig. 10 shows the results.

From Table I, we showed that the proposed GA scheduler outperforms the benchmark priority rules in all criteria. Note that here we assume the variation of performance of a chromosome due to the stochastic nature (such as machine breakdown) inside the wafer fab itself does not overwhelm the difference of performance among different chromosomes. Otherwise, some other mechanism will be required to precisely obtain the objective measures for each chromosome under the large variation. In Table II, the scheduler, which uses the QCPN-based simulator, shows great efficiency in comparing it to which one uses the CTPN-based one. Since the evaluation stage, which includes simulation (Fig. 7), is the most time-consuming part in the GA and the QCPN-based simulator runs faster than the CTPN-based one (Section II-A), the QCPN-GA approach can provide solutions within much shorter time. Fig. 10 shows that although the cutoff method can raise the performance, we must take care in choosing the cutoff level. In the beginning, performance is gradually improved by cutting off more and more genes. It implies that disregarding the machine groups with lower utilization is helpful to reduce the size of solution space without deterioration to the solution quality. However, cutting off too many genes (here, machine groups with utilization greater than 20%) could eliminate good solutions from the solution space so that the performance degrades.

In conclusion, the proposed QCPN-based simulator is efficient and accurate, and the GA scheduler, which uses the proposed simulator as the evaluation function, is the potential to set priority rules quickly, automatically, and intelligently so that schedules with satisfactory performance measures can be generated.

VII. CONCLUSION

Performance evaluation and scheduling are two important functions in operational control of the manufacturing systems. In complex manufacturing systems, such as wafer fab, simulation and rule-based control policy are the major means to achieve the above two functions. In this paper, the QCPN is proposed to construct models of wafer fab and a mechanism is proposed to realize any priority rules in the QCPN model. The simulation time is much shorter by using the proposed model to replace the CTPN model. This speedup causes only small loss of precision. In other words, the proposed solution not only has the modeling power of PN but also provides much quicker responses so that engineers can complete their missions more efficiently.

Due to various conditions in the fab, engineers usually develop more than one rule to direct process flows. The selection of a suitable rule specific to the current condition is a challenging and burdensome task. Here, we propose a GA-based scheduler, which is able to select the appropriate rule or even to combine the existing ones to a better one based on the users' focused performance measures. The engineers only need to define their preferences on different measures (if more than one), and the rule is generated intelligently and automatically. In the experiments, the proposed scheduler can generate better solutions than any single rule. Moreover, by using our fast performance evaluation tool, the scheduler takes only several minutes. This is quite satisfactory to the users, and the time can be shortened easily since parallelization of the GA is effortless.

REFERENCES

[1] D. P. Connors, G. E. Feigin, and D. D. Yao, "A queueing network model for semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 9, no. 3, pp. 412–427, Aug. 1996.

[2] M. H. Lin and L. C. Fu, "Modeling, analysis, simulation, and control of semiconductor manufacturing systems: A generalized stochastic colored-timed petri-net approach," in *Proc. IEEE Int. Conf. Systems Man, and Cybernetics*, vol. 3, Tokyo, Japan, Oct. 1999, pp. 769–774.

[3] J. Ezpeleta and J. M. Colom, "Automatic synthesis of colored petri nets for the control of FMS," *IEEE Trans. Robot. Automat.*, vol. 13, no. 3, pp. 327–336, Jun. 1997.

[4] S. Y. Lin and H. P. Huang, "Modeling and emulation of a furnace in IC fab based on colored-timed Petri net," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 410–420, Aug. 1998.

[5] M. C. Zhou and M. D. Jeng, "Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A Petri net approach," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 333–357, Aug. 1998.

[6] M. D. Jeng, X. Xie, and S. W. Chou, "Modeling, qualitative analysis, and performance evaluation of the etching area in an IC wafer fabrication system using Petri nets," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 358–373, Aug. 1998.

[7] G. Balbo, S. C. Bruell, and S. Ghanta, "Combining queueing networks and generalized stochastic Petri nets for the solution of complex models of system behavior," *IEEE Trans. Comput.*, vol. 37, no. 10, pp. 1251–1268, Oct. 1988.

[8] F. Bause, "Queueing Petri nets: A formalism for the combined qualitative and quantitative analysis of systems," in *Proc. 5th Int. Workshop Petri Nets and Performance Models*, Oct. 1993, pp. 14–23.

[9] M. Becker and H. Szczerbicka, "PNiQ: Integration of queueing networks in generalized stochastic Petri nets," in *Proc. Inst. Elect. Eng., Softw.*, vol. 146, 1999, pp. 27–32.

[10] ——, "Integration of multi-class queueing networks in generalized stochastic Petri nets," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 2, Tucson, AZ, Oct. 2001, pp. 1137–1142.

[11] M. Becker, "Modeling and simulation of a complete semiconductor manufacturing facility using Petri nets," in *Proc. IEEE Conf. Emerging Technologies and Factory Automation*, vol. 2, Sep. 2003, pp. 153–156.

[12] P. K. Johri, "Practical issues in scheduling and dispatching in semiconductor wafer fabrication," *J. Manuf. Syst.*, vol. 12, no. 6, pp. 474–485, Dec. 1993.

[13] L. Duenyas, J. W. Fowler, and L. W. Schruben, "Planning and scheduling in Japanese semiconductor manufacturing," *J. Manuf. Syst.*, vol. 13, no. 5, pp. 323–332, Oct. 1994.

[14] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry part I: System characteristics, performance evaluation and production planning," *IIE Trans.*, vol. 24, no. 4, pp. 47–60, Sep. 1992.

[15] ——, "A review of production planning and scheduling models in the semiconductor industry part I: Shop-floor control," *IIE Trans.*, vol. 26, no. 5, pp. 44–55, Sep. 1994.

[16] L. M. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Trans. Semicond. Manuf.*, vol. 1, no. 3, pp. 115–130, Aug. 1988.

[17] C. R. Glassey and M. G. C. Resende, "Closed-loop job release control for VLSI circuit manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 1, no. 1, pp. 36–46, Feb. 1988.

[18] K. R. Baker, "Sequencing rules and due-date assignments in job shop.," *Manage. Sci.*, vol. 30, no. 9, pp. 1093–1104, Sep. 1984.

[19] K. R. Baker and J. W. M. Bertrand, "A dynamic priority rule for scheduling against due-dates," *J. Oper. Manage.*, vol. 3, no. 1, pp. 37–42, Nov. 1982.

[20] T. Murata, H. Ishibuchi, and H. Tanaka, "Multi-objective genetic algorithm and its applications to flowshop scheduling," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 957–968, Sep. 1996.

[21] R. S. Lee and M. J. Shaw, "A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 27, no. 1, pp. 36–54, Feb. 1997.

[22] R. Sikora, "A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 969–981, Sep. 1996.

[23] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithm, part I: Representation," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 983–997, Sep. 1996.

[24] C. Y. Lee, S. Piramuthu, and Y. K. Tsai, "Job Shop Scheduling with a genetic algorithm and machine learning," *Int. J. Prod. Res.*, vol. 35, no. 4, pp. 1171–1191, Apr. 1997.

[25] S. C. H. Lu, D. Ramaswamy, and P. R. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Trans. Semicond. Manuf.*, vol. 7, no. 3, pp. 374–388, Aug. 1994.

[26] J. H. Chen, L. C. Fu, M. H. Lin, and A. C. Huang, "Petri-net and GA-based approach to modeling, scheduling, and performance evaluation for wafer fabrication," *IEEE Trans. Robot. Automat.*, vol. 17, no. 5, pp. 619–636, Oct. 2001.

[27] A. C. Huang, "Queueing colored Petri-net and GA based approach to modeling, prediction, and scheduling for wafer fabrication," M.S. thesis, National Taiwan Univ., Taipei, Taiwan, 2002. R.O.C..

[28] Y. D. Kim, J. G. Kim, B. Choi, and H. U. Kim, "Production scheduling in a semiconductor wafer fabrication facility producing multiple product types with distinct due dates," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 589–598, Oct. 2001.

[29] C. C. Chern and Y. L. Liu, "Family-based scheduling rules of a sequence-dependent wafer fabrication system," *IEEE Trans. Semicond. Manuf.*, vol. 16, pp. 15–25, Feb. 2003.

[30] B. W. Hsieh, C. H. Chen, and S. C. Chang, "Scheduling semiconductor wafer fabrication by using ordinal optimization-based simulation," *IEEE Trans. Robot. Automat.*, vol. 17, no. 5, pp. 599–608, Oct. 2001.