行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※

※ 半導體代工廠分解式佇列線模型之發展 ※

※ **Development of Hybrid Decomposed Queueing Network** ※

※ **Model for an IC Foundry** ※

※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：□X 個別型計畫　　□整合型計畫

計畫編號：NSC 89－2212－E－002－091－

執行期間：89 年　8 月 01 日至 90 年 07 月 31 日

計畫主持人：黃漢邦

共同主持人：

研究人員：游志源

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

X□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：國立台灣大學機械工程學系

中　華　民　國　90 年　08 月　01 日

行政院國家科學委員會專題研究計畫年度報告

半導體代工廠分解式佇列線模型之發展

(Development of Hybrid Decomposed Queueing Network Model for an IC Foundry)

計畫編號：NSC89-2212-E-002-091
執行期限：89 年 8 月 1 日至 90 年 7 月 31 日

主持人：黃漢邦　　研究人員：游志源
執行機構：國立台灣大學機械工程學系
Email: hanpang@ccms.ntu.edu.tw

中文摘要

本計畫主要目的在發展一以分解式佇列線模型 (Decomposed Queueing Model) 爲基礎的二層式 (Two-Tier) 製程時間預測系統。此系統有兩個主要模組，分別爲系統分析模組 (System Analyzer Module) 及性能指數預測模組 (Performance Measures Predictor Module)。系統分析模組主要功能爲：採用經驗機率分佈分析法 (Empirical Distribution Analysis)，針對廠內各類機台群分析晶圓到達模式 (Arrival Pattern) 的分佈及該機台群加工時間分佈(Service Pattern)，並分析各機台的整體設備效率(OEE)。性能指數預測模組則根據分析模組分析結果，選用適當的佇列線模型針對各個機台群計算其性能指數，如：製程時間(Cycle Time)、佇列線長度(Queueing Length)、產出量(Move)等，其中製程時間的估測至爲重要。本計畫採用分解式佇列線模型，即每個機台群建模成一個佇列線模型，模型間各自獨立。採用此種方法使得模型不受實際半導體廠機台、機台群、產品種類繁多等特性之限制，且可順利引進優先模型 (Priority Queue)。在系統實作方面，採用二層式模組化架構，其優點即在於它的易維護性及易擴充性。發展出之系統可以稱爲 "QFAB (Queueing FAB)"。

本計畫對於機台群之間嚴重耦合的情況將提供一可行之解決方案，由代工廠實際數據比較得知，分解式佇列線模型預測產品製程時間可達一定程度的準確度，此外，分析模組的分析結果更可提供有用的機台資訊。

關鍵詞：分解式佇列線模型、IC 代工廠、生產週期預測、整體機台效率指標。

Abstract

In this project, we will develop a hybrid decomposed queueing network model based on actual operating data from one particular foundry fab for rapid analysis of several performance measures. System analyzer module provides the analysis of arrival pattern and service pattern for each tool group and overall equipment effectiveness (OEE) analysis for each tool. System predictor module provides the forecast of important performance measures, such as products cycle time, lots remaining cycle time, tool utilization, queueing length, tool moves, stage moves. In the hybrid decomposed queueing network model, every tool group is modeled as a single queue. The empirical data of lot arrivals and services are utilized as the input of the queues. Any two queues are

assumed to be independent of each other. By utilizing this decomposition concept in the hybrid decomposed queueing network model, there is no limitation on the number of tool groups as well as product families and hence priority queue model can be utilized. Besides, a modularized two-tier system, QFAB, will be designed to implement the hybrid decomposed queueing network model proposed in this project.

A systematic analysis of arrival pattern and service pattern for tool groups is proposed. An approach to compute the effective tool number for tool groups will be also addressed. Based on the analysis, the supervisors can gain more insights and choose the proper queueing models.

Keywords: Hybrid Qeueuing Model, IC Foundry, Cycle Time Estimation, Overall Equipment Efficiency

計畫緣由及目的

In recent years, semiconductor -manufacturing industry grows at astonishing speed. According to the survey by Dataquest [1] in 1995, the total production value of global semiconductor industry will continue to grow rapidly with 13.8% compound annual growth rate. It predicts that the total production value of global semiconductor industry will be up to 110 billion U.S. dollars before year 2000. In Asia, it will keep growing at even higher growth rate, 36.0% and will achieve 11.8 billion dollars in year 2000. In the upcoming years, we can see that the strong demand for IC products will continually force the industry to develop with more vigor.

The semiconductor foundry industry in Taiwan is known for the high manufacturing and service quality. Nowadays, Taiwan has been one of the important semiconductor manufacturing country. After the foundation of UMC, as the first manufacturer of integrated circuits in Taiwan in 1980, there are about 20 semiconductor foundry fabs founded, such as TSMC, WINBOND, MOSEL-VITELIC, MACRONIX, TI-ACER, etc. Because of the contribution of these companies, the economy of Taiwan has achieved substantial advancement.

One of the most important distinctions between pure foundry fabs and R&D fabs is that the former has to meet the requirements of customers. Accordingly, how to precisely handle the manufacturing time (or cycle time, flow time) of all products (or orders) as well as other performance measures, such as average work in process at tool groups, throughput, and tool group utilization, becomes an important and challenging task for managers and engineers. The importance is described as follows:

- Once the fab engineers can precisely control the manufacturing times and fab operation conditions, such as fab capacity, yield rates, the sales can decide which orders to be taken and assign proper due-dates.
- Once the orders (or products) are released to the fab, the engineers have to confirm if all the orders can be completed on time. It should be noticed that on-time delivery performance is one of the most important indices for an IC foundry.
- Cycle times of products are highly correlated with other performance measures, such as work in process,

throughput rate, yield rate, etc. These performance measures not only affect the level of service, but also determine the quality, cost and profits.

In order to accomplish the above task, a detailed hybrid decomposed queueing network model for semiconductor foundries is proposed. Though the use of queueing models for performance evaluation of semiconductor manufacturing systems is not new, our models differs from others in the broader sense. The model can be utilized in more complex foundry fabs and can be extended to more general queueing models, including G/G/c priority queues [4].

The procedure of orders receiving can be found that due-date of products is assigned after repeated negotiation among sales, customers and production department. As to the due date assignment issues, there are two kinds of due dates: the external due date (EDD) and the internal due date (IDD) [3]. Since the determination of EDD, assigned by the sales department, usually ignores the shop floor conditions, EDD tends to have higher prediction error. The main goal of this project focuses on models to predict performance, in particular the cycle time, as well as to gain insight into the factors influencing performance of the shop floor. Cycle time is the time elapsed from a wafer starting being processed in a fab to finishing entire operation sequence [5]. After that, the wafer is ready for final probe, dicing, and packaging. Once the cycle time can be accurately forecast, the due date of orders can be properly assigned. Since the cycle time prediction is based on our queueing model considering the shop floor conditions, the procedure belongs to IDD.

A tool-model [6] based queueing model is constructed for a rather complex semiconductor foundry. Comparing to other methods proposed by many other researchers that are applied or verified in a highly simplified fab or computer simulation, our model aims at a "real" foundry fab. The forecast results are compared with actual observed fab data.

In the project, we do not attempt to propose a new exact solution or approximation for specific queueing models, such as the solution of M/G/c or G/G/c queues. Instead, we intend to develop a procedure of performance estimation with available queueing models in the literature. Only the process and inspection tools are considered. Other types of machines, such as material handling system and storage system are not modeled here.

The model developed here is not concerned with describing individual processes, and the physical and chemical principles that determine how and why a process operates. The information needed is the nature of the disturbances that influence time and quality, and particularly its frequency, duration, and pattern of occurrence.

Scheduling problems, such as lots release policy and lots dispatching policy, are not considered in this project. Since we adopt queueing theory as the underlying approach, only the steady state (or mean) values of the performance measures can be obtained.

研究方法

The development procedures can be summarized as follow. First, the hybrid decomposed queueing model concept is

proposed. Second, a classification of typical tool types in a fab is made for the model. Then, the relevant queueing models are derived. At last, the hybrid decomposed queueing network model procedure is addressed.

1. Hybrid Decomposed Queueing Network Model and Model Assumptions

A semiconductor foundry in abstraction is a set $\Gamma=\{1,...,G\}$ of distinct tool groups among which lots can be moved from one tool group to another. These lots are mainly controlled by a central transportation center between areas and by operators between tool groups. Lots belonging to a set $\Phi=\{1,...,F\}$ of different product families are released into the foundry. Each lot has a prescribed sequence of tool groups or tools it will visit before completion.

Typically, a semiconductor foundry can be viewed as a multiple-class open queueing network. Each tool group is modeled as a node in the network. Unlike the traditional open queueing network models in the literature, the determination of traffic equations is bypassed, and the empirical data of lot arrivals is used as the input for each node. The reasons to do so are: first, traditional queueing network analysis is often accompanied with the calculation of traffic equations or a normalizing constant. As the number of tool groups or product families increases, this approach becomes infeasible. Second, in the literature, the queue discipline of all queueing network models is "first come and first serve". Unfortunately, almost all foundry fabs apply "priority policy" as a dispatching rule. In this model, the nodes are treated as stochastically independent and are approximated using a GI/G/c queue having a renewal arrival process independent of service times that are independently and identically distributed within a general distribution. Specifically speaking, the models for each of the tool groups found in a fab are constructed individually. Apparently, this approach is indeed an approximation. The independence can be regarded as a generalization of the product-form solution that is valid for Markovian networks. Fig. 1 shows the concept of the hybrid decomposed queueing model.

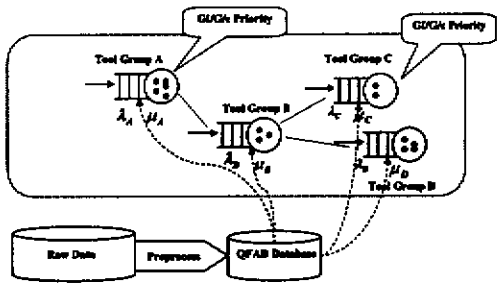

Fig. 1 Illustration of hybrid decomposed queueing network model

2. Classification of Tool Types

The tool classification criterion is according to the operation characteristics. Once the mean and variance of processing time are derived, combined with the first two moments of the interarrival times, we can determine the specific performance measures for each tool group using the queueing formulas.

According to the number of wafers being process simultaneously, the tools can be roughly categorized into single-run and batch-run tools. In our classification, single-run tools consist of single-wafer, conveyor, inspection, and multi-chamber tools. Batch-run tools consist of normal batch-run and multi-stage tools.

All of the operation information or probability distribution described in the following section can be obtained from empirical data in the foundry.

3. Derivation of Queueing Models

The typical queueing formulas utilized here involve calculating the mean queueing delay (mean waiting time) or mean queue size in terms of the first two moments (mean and variance) of interarrival times and process times. Once one particular performance measure is obtained, the others can be calculated using the Little's formula. In our model, non-available events and normal product priority were taken into consideration. Here we assumed that non-available events are the highest priority "customer" and there are k priority classes for normal products in the foundry.

Before the discussion of the queueing model, we describe each type of non-available event that can affect the tool utilization. Then, the subsequent subsections will discuss two categories of queueing models: Single-run and batch-run tools.

3.1 Non-Available Events

In the foundry, a tool can remain in several possible status. The status can be roughly classified into two types: available status and non-available status. When a tool is in one of the available status, it is or has the potential to process lots. On the other hand, when a tool is in one of the non-available status, such as PM, breakdown and etc., it cannot proceed any normal operation.

In our model, the non-available events are modeled as non-preemptive priority lots that arrive to each tool group according to renewal processes with known distributions.

3.2 Single-Run Tools

Single-run tools are the majority in the fabs. A great portion of tools falls into this type. For single-run tools, we derive two different queueing models: M/G/c/Priority and G/G/c/Priority. It should be noted that all of the models used are priority queues in order to capture the traits of operations in a real foundry.

M/G/c/Priority queues

In many real job shop systems it has been observed that the Poisson process is an adequate representation of the arrival process. Exponential distributions may not be good representations of the processing times. Therefore, the M/G/c model becomes valuable in modeling the shop floor applications.

In this case, we still assume that both lots and non-available events arrive following Poisson processes but the assumption that the processing time of all lots is exponentially distributed is removed.

There is no exact explicit solution for the M/G/c/Priority model when c is greater than unity. Consequently, the mean waiting time at tool group g, for Poisson arrivals, is

$$W_{q,k}^{(i)} = \frac{\rho_g^{\sqrt{2(c_g+1)}}}{c_g^2} \cdot \frac{\sum_{k=1}^{K} \lambda_{g,k} E[S_{g,k}^2] + \sum_{m \in \Omega_i} \lambda_{g,m} E[S_{g,m}^2]}{2(1-\sigma_{g,i-1})(1-\sigma_{g,i})} \quad (1)$$

where ρ_g is the utilization of tool group g, including non-available events. Note that Eq. (1) is exact for the case c=1.

Other performance measures can be obtained by Little's formulas.

GI/G/c/Priority queues

Queueing theory was studied thoroughly throughout the 1950s, but many problems still remains unsolved, especially the related research about GI/G/c queues. There exist several approximations for the mean queueing delay (or waiting time) of GI/G/c queues, but up to now there is no specific formula suitable for all kinds of circumstances and there is no specific approximation absolutely more

accurate than the others.

For the GI/G/c/Priority queueing system, the mean waiting time is approximated by incorporated an adjustment factor ϕ into Eq. (1). The mean waiting time at tool group g is given by

$$W_{q,g}^{(k)} = \frac{\rho_g^{\sqrt{2(c_g+1)}}}{c_g^2} \cdot \frac{\sum_{m=1}^{k} \lambda_{g,m} E[S_{g,m}^2]\phi_{g,m} + \sum_{m\in\Omega_2} \lambda_{g,m} E[S_{g,m}^2]\phi_{g,m}}{2(1-\sigma_{g,k-1})(1-\sigma_{g,k})} \quad (2)$$

where $\phi_{g,k} = (c_{a,g,k}^2 + c_{s,g,k}^2)/(c_{s,g,k}^2 + 1)$, the quantities $c_{a,g,k}^2$ and $c_{s,g,k}^2$ represent the squares of the variation coefficients (SVC) of the interarrivals and processing times of the lots with priority k at tool group g respectively. Note that $\phi_{g,\omega} = (c_{a,g,\omega}^2 + c_{s,g,\omega}^2)/(c_{s,g,\omega}^2 + 1)$. The quantities $c_{a,g,\omega}^2$ and $c_{s,g,\omega}^2$ represent the squares of the variation coefficients (SVC) of the MTBF and MTTR of non-available events at tool group g respectively.

Other performance measures can be obtained using the Little's formula.

3.3 Batch-Run Tools

Batch-run tools process in batch. This type of tool group can be modeled as a bulk service queueing system. Here, we will discuss two kinds of bulk service queues: M/M[K]/c/FCFS and G/G[K]/c/FCFS. The derivation for the mean waiting time of a lot at tool group g is then determined using

$$\begin{aligned}
(P_{0,0})^{-1} &= \frac{(c_g r_g)^{c_g}}{c_g!}\left(1-\frac{1}{V}\right)^{-1} + \sum_{i=0}^{c_g-1} \frac{r_g^i}{i!} \\
P_{m,0} &= P_{0,0}\frac{r_g^m}{m!}, \quad m = 1,\ldots,c_g-1 \\
P_{c_g,n} &= P_{0,0}\frac{r_g^{c_g}}{c_g!}\left(\frac{1}{V}\right)^n, \quad n = 0,1,\ldots
\end{aligned} \quad (3)$$

where cg is the tool number of tool group g, $r_g = \lambda_g/\mu_g$; $g \in \Gamma$, Γ is the set of tool groups, and V is the single real root, lying in the interval $(1, c_g K_g / r_g)$, of the following equation

$$f(V) = \frac{r_g}{c_g} V^{(K_g+1)} - \left(1+\frac{r_g}{c_g}\right) V^{K_g} + 1 = 0$$

where Kg is the maximum batch size. The Secant root-finding method can solve the above equation.

The mean waiting time is then determined using

$$W_{q,M/M^{K_g}/c_g/FCFS} = \frac{P_{c_g,0} V}{\lambda_g (V-1)^2} \quad (4)$$

G/GK/cg/FCFS queueing systems mimic the well-known approximation for the GI/G/c queues as

$$W_{q,G/G^{K_g}/c_g/FCFS} \approx \frac{P_{c_g,0} V}{\lambda_g (V-1)^2} \cdot \frac{(c_{a,g}^2 + c_{s,g}^2)}{2} \quad (5)$$

where $c_{a,g}^2$ and $c_{s,g}^2$ are the squares of the variation coefficients of tool group g; $g \in \Gamma$. V is the same as the one in Eq. (5).

In this project, we use FCFS queueing models rather than priority models for batch-run tools. The reason is twofold: batch-run tool processes and batch operations are time-consuming. Hence, the operator is apt to make the batch size per operation as large as possible. Consequently, the priority of lots does not have much influence. The queueing model G/GK/cg/Priority is too complicated to provide any explicit form solution. In order to take into consideration the non-available events of batch-run tool groups, $r_g = \lambda_g/\mu_g$ in Eq. (4) should be adjusted as $r_g = \lambda_g \cdot E[\tilde{S}_g]$, where $\tilde{S}_g$ is the adjusted processing time of tool group g, which incorporates the duration of

non-available events. Adjusted processing time $\tilde{S}_g$ is defined by

$$\tilde{S}_g = S_g + X_g \tag{6}$$

where Sg is the processing time of normal lots and X_g is defined as

$$X_g = \begin{cases} S_{g,\omega}, & \text{w.p.} \quad (\lambda_{g,\omega}/\lambda_g), \ \omega \in \Omega_g \\ 0, & \text{w.p.} \quad \left(1 - \sum_{\omega \in \Omega_g} \lambda_{g,\omega}/\lambda_g\right) \end{cases} \tag{7}$$

The first two moments of $\tilde{S}_g$ are then determined using

$$E[\tilde{S}_g] = E[S_g] + E[X_g]$$

$$E[\tilde{S}_g^2] = E[S_g^2] + E[X_g^2] + 2E[S_g]E[X_g]$$

The SVC for the adjusted process time, $c_{\tilde{s},g}^2$, used to replace $c_{s,g}^2$ in Eq. (5), is

$$c_{\tilde{s},g}^2 = \frac{E[\tilde{S}_g^2] - E^2[\tilde{S}_g]}{E^2[\tilde{S}_g]} \tag{8}$$

4. Modeling Procedure

Queueing Model for Each Tool Group

The distributions of interarrival times and processing times for each tool group are determined. As a result, we can chose the most appropriate queueing model prepared in Section 3 for each tool group.

However, there is still one problem that remains unsolved. As mentioned earlier, the overlapping phenomenon between tool groups is heavy. Therefore, the nominal tool number of a tool group cannot be used as the number of servers in service facility while utilizing the queueing formulas.

Here, we propose an approach to tackle this problem by making a modification on the number of tools for each tool group. The modified number of tools for tool group g is called the effective tool number and is denoted by c_g^* in contrast to the nominal tool number, $\bar{c}_g$. Assume that the tool group operation is stationary during a short time span.

Let random variable Wq,g denote the waiting time of a lot at tool group g, $f_g(\lambda, c_a^2, \mu, c_s^2, c)$ is the function of mean waiting time for the tool group g, and n is the sample size. Denote the tool set vector as $\mathbf{c} = (c_1, \ldots, c_G)$. The optimal tool set vector, or effective tool number vector, $\mathbf{c}^*$ is then determined using

$$\min_{\mathbf{c}} \left(\sum_{g \in \Gamma} \left| f_g(\lambda, c_a^2, \mu, c_s^2, c) - \Sigma_{i=0}^{n} w_{q,g,i}/n \right| \right) \tag{9}$$

given $f_g(\cdot)$, $\lambda_g, c_{a,g}^2, \mu_g, c_{s,g}^2$ for $g \in \Gamma$ and n observed waiting times Wq,I for each tool group.

Note that Eq. (9) turns out to be a first-order linear equation with one set of independent variables, c, after substituting the given values. Hence, the existence and uniqueness of the solution of c is guaranteed. The solution of Eq. (9) is the optimal value c* and it serves as the number of tools for each tool group in our queueing models.

Once the effective tool number for each tool group is obtained, the performance measures for each tool group as well as the performance measure for the entire system can be calculated.

Mean Cycle Time of Products

From our previous analysis, the mean waiting time and mean processing time are calculated for each tool group. This information, together with the routing flow

information for each product family, allows us to compute the estimates for the average cycle time for each product family.

Suppose that we wish to calculate the average cycle time for product family $f \in \Phi$. Let $N^f = \{1,2,...,N\}$ denote the set of nominal operations for product family f. Then the cycle time of product family f with priority i, CTf,(i), is approximately determined using

$$CT^{f,(i)} \approx \sum_{n \in N^f} \left(W^{(i)}_{q,g(n)} + S_{g(n),r(n)} \right) \quad (10)$$

where the mean waiting time and mean processing time are the same as before.

結論與成果

Fig. 2 shows the actual and predicted cycle time of the products. Note that the average cycle time of PROD_A is computed as the average of the cycle time of LOT_1, LOT_2, LOT_3, LOT_4, and LOT_5 and so forth. The forecasting time instant is on Nov. 18, 1998.

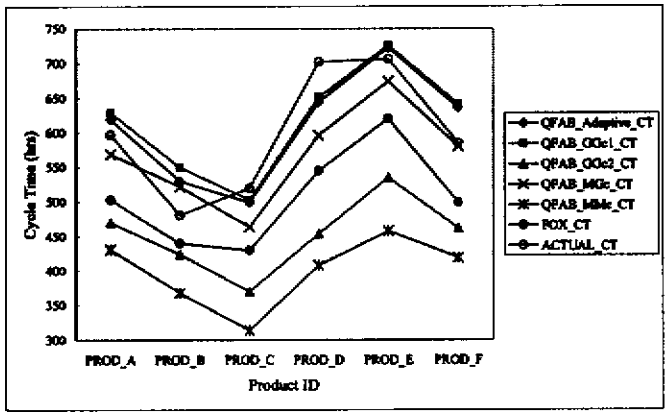

Fig. 2 Actual and predicted cycle time of six different products.

The meanings of the curves in Fig. 2 are

- ACTUAL_CT: Actual cycle time of product
- QFAB_Adaptive_CT: Predicted cycle time of products by QFAB_Adaptive model
- QFAB_GGc1_CT: Predicted cycle time of products by QFAB_GGc model 1
- QFAB_GGc2_CT: Predicted cycle time of products by QFAB_GGc model 2
- QFAB_MGc_CT: Predicted cycle time of products by QFAB_MGc model
- QFAB_MMc_CT: Predicted cycle time of products by QFAB_MMc model
- FOX_CT: Predicted cycle time of products by FOX model

Fig. 3 shows the comparison error of several models, including QFAB and FOX. The cycle time forecast error of the product family $f \in \Phi = \{1,...,F\}$ is computed by

$$\text{Cycle Time Error}^f = \frac{\text{Estimated Cycle Time}^f - \text{Actual Cycle Time}^f}{\text{Actual Cycle Time}^f} \times 100\% \quad (11)$$

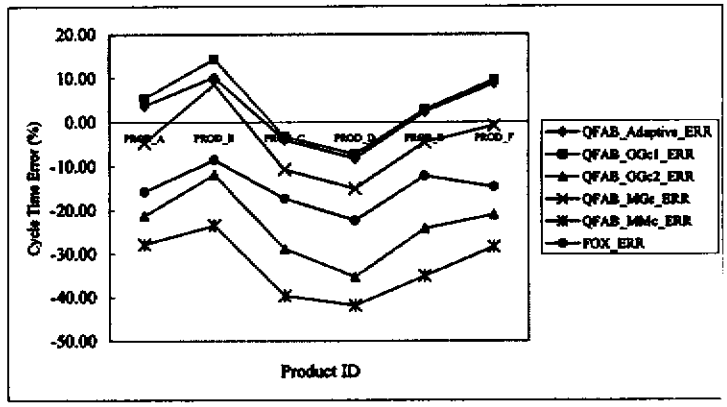

Fig. 3 Comparison of forecast errors of several models

In Fig. 2 and Fig. 3, six types of models are used. They are QFAB_Adaptive, QFAB_GGc model 1, QFAB_GGc model 2, QFAB_MGc, QFAB_MMc, and FOX. Different from other approaches, QFAB_Adaptive model constructs individual sub-model for each tool group, based on the analyzed information, such as arrival pattern and service pattern. For example, if tool group 1 has Poisson arrival and no evidence indicates that it has exponential process time distribution, it will be modeled as M/G/c priority queue. If both the distribution

of arrival and service pattern can not be proven as exponential ones, the tool group will be modeled as a G/G/c priority queue. However, all the tool groups are modeled as G/G/c model 1, G/G/c model 2, M/G/c, and M/M/c priority queues in QFAB_G/G/c model 1, QFAB_G/G/c model 2, QFAB_M/G/c, and QFAB_M/M/c, respectively. The cycle time estimation of lots for some specific product is conducted at release time of the lots, and forecast error is computed by Eq. (11) after the lots complete all the operation steps. From Fig. 3 and our analysis, a number of important points are observed as follows.

- The accuracy of QFAB_Adaptive model is better than others for most product families.
- The error of FOX is approximately 18%. The accuracy is better than QFAB_GGc model 2 and QFAB_MMc, but is worse than the other two models.
- The cycle time estimation of QFAB_Adaptive model is between the one of QFAB_GGc model 1 and QFAB_MGc. In fact, the results of QFAB_GGc model 1, and QFAB_MGc are the upper bound and lower bound of QFAB_Adaptive, respectively.
- Due to high SVC of interarrival times of some tool groups, the cycle time estimation of QFAB_GGc modle 1 is usually greater than the one of QFAB_MGc model. Remember that the former is equal to the latter multiplied by a factor.
- It is not surprised that QFAB_MMc model has the lowest cycle time estimation.
- The cycle time estimation of QFAB_GGc model 2 is underestimated.
- Compared with the aggregated cycle time, the forecast error of waiting time for each tool group is rather large. One of the possible reasons is that discrepancies in waiting time estimation tend to cancel each other out when the aggregated cycle time is calculated.

While we cannot report any explicit comparison of cycle times for proprietary reasons, it is observed that the forecast error of QFAB_Adaptive model fall within 15% for a majority portion of products. Note that the values computed in Fig. 2 are average or expected ones for *all* priority classes of lots. Specific analysis of SH lots will be given in the subsequent paragraph.

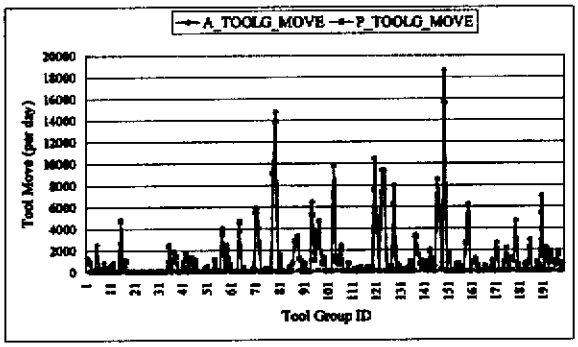

Fig. 4 Tool group moves comparison between actual data (A_TOOLG_MOVE) and predicted data (P_TOOLG_MOVE) for 1998-11-21. The prediction is conducted on 1998-11-20. Overall prediction error is 6.2%.

Tool Group Move

Fig. 4 shows the actual and predicted moves for all tool groups on Nov. 21, 1998. The forecast result is quite satisfactory. We define the forecast error of tool moves and stage moves as

$$\text{Forecasted Error (\%)} = \frac{\text{Forecasted Total Moves} - \text{Actual Total Moves}}{\text{Actual Total Moves}} \times 100\% \quad (12)$$

The forecast error of total moves is about -6.2%. The result is shown in Fig. 5 The forecast error of total stage moves is -3.6%.

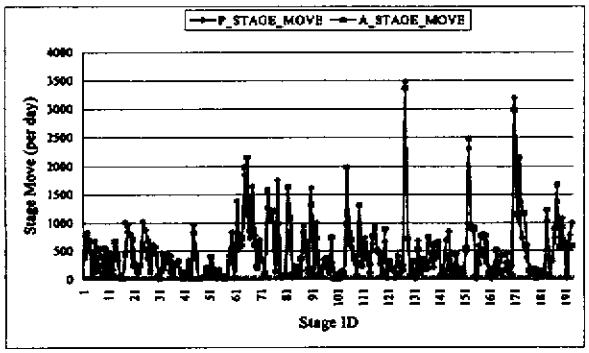

Fig. 5.18 Stage moves comparison between actual data (A_STAGE_MOVE) and predicted data (P_STAGE_MOVE) for 1998-11-21. The prediction is conducted on 1998-11-20. Overall prediction error is 3.6%.

Tool Group Utilization

Due to the phenomenon of tool groups overlapping, the actual utilization toward a specific tool group is not easy to identify. For example, if Tool Group A contains three distinct tools, Tool A, Tool B, and Tool C, with actual utilization ρ A, ρ B, ρ C, respectively. The utilization of Tool Group A cannot be estimated as the average of ρ A, ρ B, and ρ C, because some portion of ρ A, ρ B, and ρ C may belong to other tool groups.

However, the utilization of tool groups can be estimated by QFAB The illustration of forecast result is shown in Fig.6.

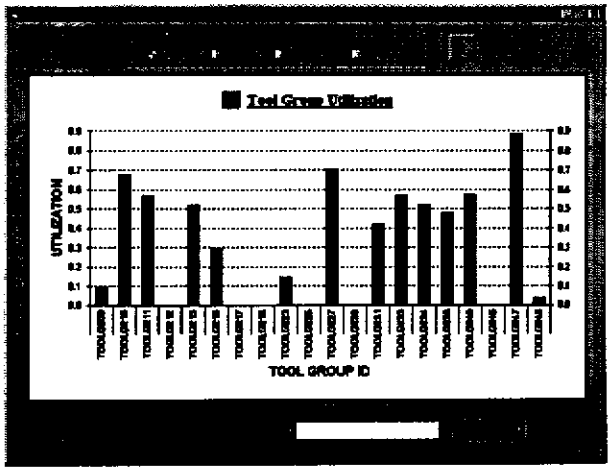

Fig. 6 Illustration of the estimation of tool group utilization

參考文獻

[1] J. Banks, J. S. Carson II, and B. L. Nelson, *Discrete-Event System Simulation*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] C.M. Cheng, "The Design of Due-Date Assignment Model for a Wafer Fabrication Factory," Master Thesis, Institute of Industrial Engineering, National Chiao Tung University, 1996.

[3] T. C. E. Cheng, and M. C. Gupta, "Survey of Scheduling Research Involving Due Date Determination Decisions," European Journal of Operational Research, vol. 38, no. 2, pp. 156-166, 1989.

[4] L. Sattler, "Using Queueing Curve Approximations in a Fab to Determine Productivity Improvements," *Proceedings of 1996 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pp. 140-145, 1996.

[5] W. Whitt, "The Queueing Network Analyzer," *Bell System Technical Journal*, vol. 62, no. 9, pp. 2779-2815, 1983.

[6]C.Y. Yu, S.W. Chen, H.P. Huang, "Estimation of Waiting Time for Products in an IC Fab," *Joint Conference of the 5th Intl. Conf. on Automation Technology and 1998 Intl. Conf. Of Production Research* Taiwan, 1998.

行政院國科會補助參加國際會議報告

2001 IEEE Robotics & Automation

一. 參加會議經過

二. 與會心得

三. 參觀活動

四. 建議

五. 攜回資料

參加 2001 年 IEEE 國際機器人暨自動化會議報告

黃漢邦 教授　台灣大學機械系

一. 參加會議經過

本次會議在韓國漢城舉行，會議期間從五月二十一日至五月二十六日。我於五月二十日從台北搭機前往漢城。從機場到 COEX 飯店途中，頗驚訝漢城的改變。寬廣的街道、高聳的大樓、起伏的地形，彰顯韓國的企圖心。街道上夾雜著小販，依稀又是台灣街道的景象。

本次大會在 COEX 飯店舉行，該飯店位於市中心，與世貿展示中心相鄰，自成一個集旅館、飲食和展覽的綜合體。因飯店地處鬧區，入夜後仍見遊客熙來攘往，在街上行走還算安全。大會在五月二十二日晚上舉行歡迎酒會，並舉行討論會，直到晚上九點半結束。在五月二十三日傍晚，大會特別安排漢江的船上之旅。從下午五點半在碼頭搭上船，沿著漢江航行，一邊吃晚餐，一邊觀賞漢城夜景，直到晚上十點才結束這浪漫之旅。

五月二十四日晚上，大會舉辦今年最佳論文及晉升 Fellow 頒獎儀式。今年，我與研究生發表的論文，順利入圍大會最佳五篇自動化論文，但最後未能獲得最佳論文獎，僅獲頒入圍獎狀。當天晚上亦為大會晚宴及專題演講，在十點半左右結束。在晚宴結束後，來自台灣、大陸、新加坡、香港及美國的華人一群浩浩蕩蕩前往中國餐廳聚餐和討論。五月二十五日是大會論文發表最後一天，同時舉辦餞別晚會，隔天仍有討論會。我於二十六日從漢城搭飛機返回台北。

二. 與會心得

此次會議共有 1150 篇投搞，僅接受 678 篇，接受率約 58%，全部文章分 113 個場次發表，同一時段則有十三個場次同時進行。本人在會中一共發表二篇文章，分別是"The Modeling and Control of the Cluster Tool in Semiconductor Fabrication," 和 "Priority-Based Tool Capacity Allocation in the Foundry Fab." 其中第二篇入圍最佳論文獎。

本次會議安排五個 video sessions，由 video proceeding 的作者各自解說五分鐘，現場反應熱烈。此外，大會安排了二個座談會和三次大會特別演講。二次座談會的主題分別

是 "Manufacturing Automation in the Era of Information Technology,"及"Advances in Service Robotics for Caregiving and Medical Assistance."大家的重點集中在未來的方向和研發計劃. 三次大會特別演講的主題分別是 "Medical Robotics and Computer-Integrated Surgery: coupling Information to Action in 21'st Century Medicine," "New Trends in Machine Tool Building, Fast, and Reliable," 及 "Nano Robotic World—From Micro to Nano."

從今年的論文發表及大會所安排的特別演講與座談會來看，有下面幾個特點：

1. 全方位的發展機器人 - 從工業、農業、營建和辦公室的應用，到娛樂、家庭、個人、軍隊、太空、醫療和海底的應用。這方面以日、美、德三國為發展重心。

2. 人性納入自動化技術 - 純技術無法根本解決問題，而需考慮以人為主體的製造自動化技術。這方面以日本和美國為代表。

3. 人形化機器人的發展 – 從研究走向娛樂及軍事運用。從 1999 年起也另外有一個探討人形化機器人的國際會議。

4. Nano Technology --- 由 MEMS 走向 Nano。 但如何組裝成一有用的裝置，為目前另一發展重點。

5. Medical Robot 的發展 – 將 Robot 及自動化技術應用到 Health care，Surgery，Rehabilitation 為一發展重點，人工義肢即為一例。而遠端醫療及遠距開刀亦是一重點。

6. 機器人及自動化技術的倫理觀 – 由於機器人及自動化系統愈來愈聰明，甚至具有某種程度的思考能力，因而它們與人的互動關係愈顯微妙。

三. 參觀活動

在開會期間，大會安排參觀 Samsung Electronics，現代造船及漢城大學，因與我論文發表時間衝突，故未前往。

四. 建議

本會議是機器人及自動化領域最重要也是最具水準的會議之一，國科會每年應儘量補助所有論文發表者與會。

五. 攜回資料

大會論文集的光碟片一張及大會論文摘要一本。

Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea • May 21-26, 2001

Priority-Based Tool Capacity Allocation in the Foundry Fab

Chih-Yuan Yu*, Han-Pang Huang**
Robotics Laboratory, Department of Mechanical Engineering
National Taiwan University, Taipei, 10660, Taiwan.
TEL/FAX: (886) 2-23633875, E-mail: hphuang@w3.me.ntu.edu.tw
**Professor and correspondence addressee *Graduate student

Abstract

A virtual unit, tool group, is used to represent the group of tools in semiconductor foundry. Hence, a tool may belong to several tool groups because it may be handle several kinds of recipes. The coupling relation between the tools and the tool groups is a complex many-to-many relation. This paper aims to decide the tool-dispatching rule in tool capacity allocation. The term, "priority-based", is two-fold. One is to prioritize the lots in the fab, the other is to prioritize the tools in the correlated tool groups. The priority-based algorithm is used to estimate the tool utilization, bottleneck, tool-sharing in the tool groups, throughput of the tool groups and WIP (Work In Process) in a specific day. The proposed algorithm is applied to a real foundry fab and the results show that maximum fuzzy candidative ratio is better than other rules. Besides, a demonstration shows that the supervisor can find the bottleneck tools by the proposed algorithm and release the bottleneck by adding the right tools.

Key words: tool capacity allocation, neural network, cycle time estimation, priority

1. Introduction

The main purpose of this paper is focused to the optimization of daily tool capacity allocation. Once the tool capacity is available, the supervisors are eager to find the best configuration of on-time delivery and tool capacity allocations. The question arises from the coupling effects between tools and tool groups. Tools, which can handle the same recipe, will backup each other. The term, "tool group", is a group of backup tools. Usually, a tool group has more than one tool. Besides, a tool may process more than one recipe. It may belong to more than one tool group. Therefore, the coupling relation between the tools and the tool groups is a many-to-many relation. For convenient management, the route of a lot is not listed tool by tool. Instead, it is listed tool-group by tool-group. Any available tool in the tool group can serve the processing lots in this tool group. As a result, the supervisor wants to know the division of tool capacities for its tool groups. The tool capacity allocation is to decide the capacities of each tool for its tool group. In Fig. 1, the tools, "CETM03", "CMTM01", and "CETM02" are all enlisted in two tool groups, respectively. Then, the tool capacity allocation should optimize the tool utilization and on-time delivery of lots constrained with given tool capacities.

Wu *et al.* [9] proposed the Target Setting System (TSS) to calculate the daily target of each stage, a set of sequential operations. Chang *et al.* [1] proposed a formulation for optimizing long-term production and machine allocation for semiconductor manufacturing facilities. This paper shows a new approach to allocate the short-term capacities of tools for tool groups.

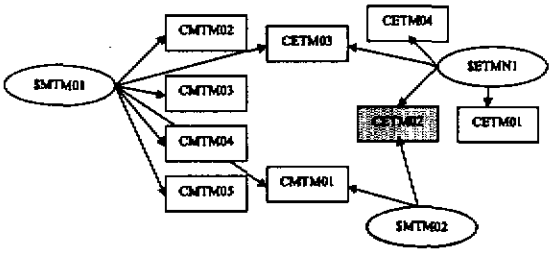

Fig. 1 The coupling effect between tools and tool groups

2. System Overview

2.1 On-time delivery performance in long-term planning

Assigning the release policy, allocating tools' capacities, and deciding a best lot-dispatching rule are the main decisions in the foundry fab.The order release plan depends on the due date of the lots and capacities of the fab and correlates with the WIP control level. A good order release plan can help to achieve the goals. However, the production control department gives the release policy. It is the first assumption in this paper. The dispatching rule adjusts the processing priorities for all lots in order to shorten the mean and variance of the cycle time and then fulfill the due-date [7]. In order to achieve on-time delivery, the least slack policy is usually used as a lot-dispatching policy [8] in the semiconductor foundry. Hence, the least slack policy is used as the lot-dispatching rule in this paper. This is the second assumption in the paper.

Suppose that the lot i arrives to the fab with a due date denoted as • *(i)*. The estimated remaining cycle time for the lot i obtained by the estimator is denoted as ERCT(i), i.e., the slack s(i) of the lot i can be defined as

$$s(i) = \bullet\ (i) - t - ERCT(i) \qquad (1)$$

The quantity• *(i) - t - ERCT(i)* denotes the relative urgency of the lot i, therefore, the slack value, *s(i)*, can tell how much time a lot has been ahead or behind its committed due date. If an idle tool has to decide which lot it should serve next, *s(i)* is reasonable to select the lot with the smallest *s(i)*; i.e., the most urgent lot. As the degree of urgency is determined, on-time delivery performance can be improved by adjusting lots' priorities based on least slack policy.

The cycle time estimation [2, 3, 4, 5,] is the great topic in semiconductor industry. Yu *et al.* [11] shows three different algorithms to predict the product cycle time and lot remaining cycle time. The neural network approach shows good forecasting ability and short computation time.

Besides, it also provides the estimated cycle time of lots for least slack policy. After comparison, the on-line learning system [10] will be used to forecast the operation time and the estimated cycle time for lot in this paper.

2.2 Tool capacity allocation in short-term planning

The term "priority-based" is two-fold. One is to prioritize the lots in the fab by applying lot-dispatching rule, and the other is to prioritize the tools in the correlated tool groups by applying tool-dispatching rule.

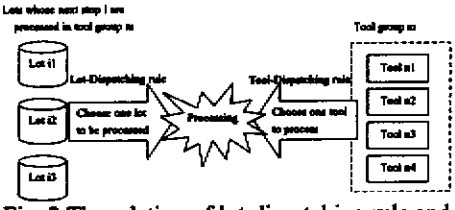

Fig. 2 The relation of lot-dispatching rule and tool-dispatching rule

In the first phase, each lot in the fab is assigned a priority class to identify its emergency and importance. The priority assignment of lots is based on least slack policy as mentioned in section 2.1. A lot with higher priority value is always selected first for processing. The priority classes used in this paper include Super Hot lot (SH), Hot lot (H), Rush lot (R), Normal lot (N), and Slow lot (S).

Given tools capacities, PM schedule, and release schedule, the capacity consumption of tools is assigned according to lots' priority classes. SH lots will be allocated first, then H lot and so on. The time horizon for the tool capacity allocation is one day. The estimated operation steps of a lot are based on the on-line learning system and will be further described in section 5.2. The first block in Fig. 3 shows the required time slot that the lot K completes operation 31. The time includes the waiting time and the processing time. The total flow in Fig. 3 denotes that if the tools of the fab are not finite capacity, then the lot can go through from operation 31 to 42 in the same day. Unfortunately, the capacity is finite.

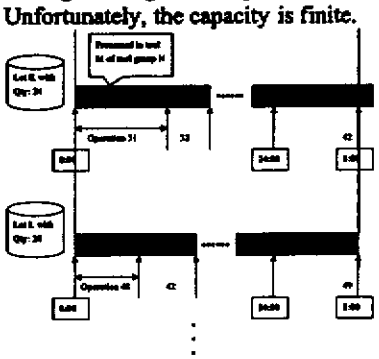

Fig. 3 The estimated steps of lot K, L in a specified day

However, the lot with the highest priority class can consume the capacities first. If the estimated throughput of the tool-group N in that day is 300 wafers, 24 (the number of wafers of lot K) is reserved for the lot K. If there is no more capacity in a tool group, the lot will be blocked from that tool group As a result, the lower the priority class the lot has, the higher opportunity it is blocked due to the priority-based tool capacity allocation.

Which available tool in the tool group will be chosen to process the next lot is the second phase of prioritization. The tools in a specified tool group are assigned a priority value to indicate the service sequence of the tools. That is the tool-dispatching rule will be discussed in section 4. After reserving the tool capacity for lots, the blocked lots flag the bottleneck tools and tool groups. The shop floor manager can use this useful information to adjust the PM schedule or add tools to solve the bottleneck. After rearranging the tool capacity configuration, the manager should re-run the tool capacity allocation again until a balance among tool capacity, tool utilization and on-time delivery is achieved.

The symbols used are described below.

L(i)	the lot i, $L(i) \in L$, L is the set of all lots.
NL(i)	the quantities of the lot i.
s(i)	the slack value of the lot i.
W(i)	the waiting time of the lot i.
P(i)	the processing time of the lot i.
CT(i)	the cycle time of the lot i.
ERCT(i)	the estimated remaining cycle time of the lot i.
S(i)	the step that the lot i will go through in one day.
T(n)	the tool n, $T(n) \in T$, T is the set of all tools.
TG(m)	the tool group m, $TG(m) \in TG$, TG is the set of all tool groups.
NT(n)	the number of tool groups that the tool n is enlisted.
NTG(m)	the number of tools in the tool group m.
TGT(n)	the set of the tool groups that tool n is enlisted.
TTG(m)	the set of the tools in tool group m.
CT(n)(t)	the capacities of the tool n at day t.
$e_{T(n)}(t)$	the utilization of the tool n at day t.
$UC_{T(n)}(t)$	the upper bound of the capacities of the tool n at day t.
$EC_{T(n)}(t)$	the estimated capacities of the tool n at day t.
$RC_{T(n)}$	the remaining capacities of the tool n.
$N_B(m)$	the number of blocked lots in the tool group m.
BR(n)	the bottleneck responsibility of the tool n.
$Move_{T(n)}$	the moves of the tool n.
$Move_{TG(m)}$	the moves of the tool group m.

3. Tool-Dispatching Rules

The coupling effect arises from the many-to-many relation between tools and tool groups. Due to the coupling effect, it is difficult to allocate the capacity of a tool to its tool groups. When a lot will consume the capacity of a tool group, the system should decide which tool in the tool group can serve the lot, i.e., the tool-dispatching rule in the tool groups. There are six types of tool-dispatching rules considered in this paper.

3.1 Maximum Remaining Capacity (MRC)

MRC means the tool with the maximum remaining capacity is chosen to serve the lot in the tool group m and can be expressed by

$$T^* = \arg\max_{T} \{RC_{T(n)}\} \quad \forall T(n) \in T_{TG(m)}, \quad \forall RC_{T(n)} > 0 \quad (2)$$

3.2 Maximum Remaining Capacity Ratio (MRCR)

Remaining Capacity Ratio of tool n at time t ($RCR_{T(n)}(t)$) is defined as $RC_{T(n)}(t)$/ $EC_{T(n)}(t)$ and $RC_{T(n)}(t)$ ranges from 0 to 1. MRCR can be expressed by

$$T^* = \arg\max_T \{RCR_{T(n)}\} \quad \forall T(n) \in T_{TG(m)}, \ \forall RC_{T(n)} > 0 \quad (3)$$

3.3 Maximum Dedicative Ratio (MDR)

Dedicative Ratio of tool n at time t ($DR_{T(n)}$) is defined as $1/N_{T(n)}$. The less number of the tool groups a tool is enlisted (larger $DR_{T(n)}$), the more capacity the tool can contribute to each tool group. MDR can be expressed by

$$T^* = \arg\max_T \{DR_{T(n)}\} \quad \forall T(n) \in T_{TG(m)}, \ \forall RC_{T(n)} > 0 \quad (4)$$

3.4 Any Available Capacity (AAC)

AAC can be expressed by

$$T^* = Random(T(n)) \quad \forall T(n) \in T_{TG(m)}, \ \forall RC_{T(n)} > 0 \quad (5)$$

3.5 Sequence Available Capacity (SAC)

SAC can be expressed by

$$T^* = Seqence(T(n)) \quad \forall T(n) \in T_{TG(m)}, \ \forall RC_{T(n)} > 0 \quad (6)$$

3.6 Maximum Fuzzy Candidative Ratio (MFCR)

In the first three approaches, MRC, MRCR, MDR are all the attributes to choose the candidate tool. As a result, the combined approach is usually used to include all these effects. Then the following relation can be found,

$$T^* = \arg\max_T \{\alpha \cdot RC_{T(n)} + \beta \cdot RCR_{T(n)} + \gamma \cdot DR_{T(n)}\} \quad (7)$$

$$\forall T(n) \in T_{TG(m)}, \ \forall RC_{T(n)} > 0, \ \alpha + \beta + \gamma = 1$$

The settings of • ;" ;" • are usually subjective or by trial and error. However, fuzzy inference is suitable for consider multiple attributes with reasoning.

In the fuzzy system, there are two inputs and one output. Inputs are only chosen as $RC_{T(n)}$ and $DR_{T(n)}$ because $RC_{T(n)}$ and $RCR_{T(n)}$ are similar. The output is the fuzzy candidative ratio (FCR). MFCR means the tool with the maximum fuzzy candidative ratio is chosen to serve the lot in the tool group m and can be expressed by

$$T^* = \arg\max_T \{FCR_{T(n)}\} \quad \forall T(n) \in T_{TG(m)}, \ \forall RC_{T(n)} > 0 \quad (8)$$

The fuzzy inference system contains the following properties.

- Linguistic variables {remaining capacity (RC), dedicative ratio (DR), candidative ratio (CR)}.
- U(RC) = [0, •], U(DR) = [0,1], U(CR) = [0,1]
- Membership functions of RC, DR, CR are shown in Fig. 4, 5, 6, respectively
- Fuzzy rules are shown in Table. 1.
- The Mamdani fuzzy inference is adopted and uses product and max for T-norm and T-connorm operators, respectively, and Centriod of area (COA) is used as the defuzzification strategy.

The detailed descriptions of all procedures are described in next section.

Table 1 Fuzzy Rules

Dedicative Ratio	Remaining capacity			
	Too Small	Small	Large	Too Large
Too Low	Worst	Worse	Worse	Good
Low	Worst	Worse	Good	Good
Medium	Worse	Bad	Good	Better
High	Bad	Better	Better	Best

4. Priority-Based Tool Capacity Allocation

This paper discusses tool capacity allocation in short-term planning, while the on-time delivery in long-term planning follows the description given in section 2.1. This paper makes the following assumptions. (1) Release schedule and PM schedule are inputs of the system. (2) The lot-dispatching rule is least slack policy. (3) Step is adopted as the basic unit for describing the process flow. (4) All lots are assigned a priority class according to the least slack policy. (5) The proportion of five priority classes is fixed.

4.1 Procedures

Priority-based tool capacity allocation is a decision support system for the shop floor management. It runs daily in the beginning of a day in order to forecast the bottleneck of tool groups and tools in advance. The procedures of priority-based tool capacity allocation can be listed as follows.

(1) Keep running on-line learning system, (2) Obtain the PM (Preventive Maintenance) and release schedules, (3) Obtain the WIP information, (4) Adjust the priorities of lots, (5) Obtain the capacities of the tools, (6) Allocate the capacities of the tools, (7) Find the bottleneck tools and tool groups, (8) Take action to solve the bottleneck tools and tool groups and repeat procedures (1) to (8).

4.2 Running on-line learning system

Readers can reference [10] if interested in the on-line learning system.

4.2.1. Support for the tool capacity allocation

When the server program of on-line learning system keeps running, the information of the number of steps that the lot will go through in a day can be obtained by

$$S(i) = \left\{ \arg\min_s \left\{ \sum_{i=k+1}^{s} (W_i + P_i) > 1 \ day \right\} \right\} - k \quad (9)$$

where k is the current step, and s is the step that the lot i will arrive one day later.

4.2.2. Support for least slack policy

The ERCT of the lot i can be obtained by

$$ERCT(i) = \sum_{i=m+1}^{N} (W_i + P_i) \quad (10)$$

where m is the current step and N is the total steps of the lot i.

4.3 Obtaining the PM (Preventive Maintenance) and the release schedules

The PM and release schedules in this paper are obtained from the automation department.

4.4 Obtaining the WIP information

In this paper, the WIP information is just the snapshot of the lots in the fab. Therefore, the WIP information is retrieved from the shop floor. It provides the information of lots' current location, current step, quantities, and priority class, etc.

4.5 Adjusting the priorities of lots

The least slack policy is used to prioritize all lots daily in the fab before executing the tool capacity allocation procedure. The proportion of each priority class

is fixed as 5% lots for SH, 15% for H, 30% for R, 45% for N, and 5% for S. The determination of the proportion is based on the research in [2] and the engineers' experiences from the real fab. The estimated remaining cycle time and the slack value of wach lot can be calculated from Eq. (10) and Eq. (1), respectively. If the slack s(i) > 0, it means that the lot commits its due date on time; otherwise, the lot will be delayed. The smaller the slack value of the lot has, the higher the priority class of the lot will be re-assigned.

The steps to re-assign the priority class of each lot are listed below.

Step 1: Calculate the slack values of all lots in the fab,

Step 2: Highlight some important lots by the managers,

Step 3: Sort the lots in terms of the larger slack value and importance,

Step 4: Reassign the priority class according to the proportion, (SH, H, R, N, S) = (5%, 15%, 30%, 45%, 5%).

4.6 Obtaining the capacities of the tools

The capacity of a tool is number of wafers that the tool can process in a day regardless the recipes. In general, the capacity of a tool depends on the PM schedule of the tool, the frequency of the setup change, recipes and the idle time of the tool. It is difficult to estimate the actual capacity of the tool due to the unpredicted interrupt and the idle time of the tool. The method for estimating the capacities of the tools in this paper is the dynamic moving average of the past seven days. Let $AC_{T(n)}(t)$ be the actual capacity of the tool n at day t, and $e_{T(n)}(t)$ be the utilization of the tool n at day t. The upper bound of the capacity of the tool n at day t, $UC_{T(n)}(t)$, can be obtained by

$$UC_{T(n)}(t) = \frac{AC_{T(n)}(t)}{e_{T(n)}(t)}, \qquad \forall T(n) \in T \tag{11}$$

then the estimated capacity, $EC_{T(n)}(t)$, can be obtained by

$$EC_{T(n)}(t) = \frac{\sum_{i=1}^{7} UC_{T(n)}(t-i)}{7} \cdot \Omega_{T(n)}, \qquad \forall T(n) \in T \tag{12}$$

$$\Omega_{T(n)} = \frac{24hours - PM\ \ time\ \ period}{24hours}$$

where $\Omega_{T(n)}$ is the portion of available time for processing.

4.7 Allocating the capacities of the tools

The main purpose of this section is to decide the amount of the tool capacities for each tool group. For example, the upper bound capacity of the tool n is 300 wafers and the tool n is enlisted in two tool groups. After running the capacity allocation module, the system will reserve 100-wafer processing capacity for the tool group 1, 150-wafer capacity for the tool group 2, and 50-wafer capacity is not used.

The capacity allocation algorithm is given below.

Step 1: Load all necessary data

The WIP information, the release schedule, the capacities of the tools, the mapping table of the tools and the tool groups are loaded into the system.

Step 2: Sort the lots by their priority class

Apply the method mentioned in section 4.5.

For i = 1 to L (L: the number of all lots in the fab plus new released lots)

Step 3: Calculate the steps that lot i will go through in the day

Applying eq (9).

For k = 1 to *S(i)*

Step 4: Find the next tool group (NTG)

Looking up the route of lot i. Let NTG = TG(m).

Step 5: Find the candidate tool in the tool group

Applying one tool-dispatching rule to find the candidate tool.

If the remaining capacity of T*, RC_{T^*} > the number of the lot i, $N_L(i)$, then reserve the capacity for the lot i and $RC_{T^*} = RC_{T^*} - N_L(i)$

else the lot is blocked in this tool group due to lack of the capacity, Exit the following steps of allocating

End *for-loop*

End *for-loop*

End of the procedure

The results of this procedure are the WIP information for one day later, reserved capacity of the tool for each tool group, the bottleneck tools, and the bottleneck tool groups.

4.8 Finding the bottleneck tools and tool groups

The bottleneck tools and bottleneck tool groups can be easily found from the capacity allocation algorithm. Let $N_B(m)$ = the number of blocked lots in the tool group m. It tells that $N_B(m)$ lots are blocked and can not continue their routes due to the lack of capacity of the tool group m. If $N_B(m)$ is too large, this tool group is called a bottleneck tool group.

A bottleneck tool group results in the blockage of all the tools enlisted in this tool group. The bottleneck responsibility (BR) of the tool n is defined by

$$BR(n) = \sum_{k \in TG}^{N_{T(n)}} \frac{N_B(k)}{N_{TG(k)}}, \quad for\ TG\ enlisting\ T(n) \tag{13}$$

where $N_B(k)$ is the number of blocked lots in the tool group k; $N_{T(n)}$ is the number of tool groups that the tool n is enlisted; $N_{TG(k)}$ is the number of tools that the tool group k enlists; $N_B(k)/N_{TG(k)}$ means the bottleneck responsibility of the tool m in the tool group k. Similarly, if $BR(n)$ is too large, the tool n is called the bottleneck tool.

4.9 Action and re-allocation

The shop floor manager should take action to solve the bottleneck issue based on the information provided above. The action depends on the extra available resources, the experience of the managers, the PM schedule of the tools and the on-time delivery performance. Although it is not easy to make decision, the manager can adjust the capacities of the tools with the aid of the above results. Which tool should be added or which tool's PM schedule can be changed in order to solve the bottleneck can rely upon the list of the bottleneck tools and tool groups.

After adjusting the capacities and the usage of the tools, re-run the procedures (5) to (7) in 4.1 until the balance between the tool utilization and bottleneck is achieved

5. Results

The data used in this paper is retrieved from a real fab. The on-line learning system uses the on-line data from the real fab in real time. The time horizon of the data set is from 1998/11/1 to 1998/12/30 and the testing was performed on 1998/12/31. The system is executed at 8:00 AM. After running the system, the followings can be obtained for the manager.

5.1 WIP information

The total WIP distribution of all tool groups is shown in Fig. 7. The estimated total WIP for one day later is 20313 wafers.

5.2 Tool group moves

If the tool group move of the tool group m is $Move_{TG}(m)$, it means that there are $Move_{TG}(m)$ wafers flowed through the tool group m, i.e. the throughput of the tool group. The tool group move is defined as when a lot completes an operation, the lot contributes moves to this tool group. The moves that added to this tool group are equal to the number of the wafer of the lot. For example, if a lot with 24 wafers has been processed by the tool group m, then 24 tool group moves are added to the tool group m. Thus, we have

$$Move_{TG}(m) = \sum_{i \in \tilde{L}(m)}^{N} N_L(i) \tag{14}$$

where N is the number of tools in $\tilde{L}(m)$; $N_L(i)$ is the quantity of $L(i)$; $\tilde{L}(m)$ is the set of lots which are processed in the $TG(m)$; L is the set of all lots. Note that $\tilde{L}(m)$ is the subset of L. The estimated total tool group moves in "1998-12-31" are 54597 wafers as shown in Fig. 8. The move of the tool group 33 is as high as 2850 wafer moves.

5.3 Tool moves

Similar to the tool group moves, the estimated tool moves in the tool n, $Move_T(n)$, is the number of wafers that are processed in the tool n. We have

$$Move_T(n) = \sum_{i \in \tilde{L}(n)}^{N} N_L(i) \tag{15}$$

where $\tilde{L}(n)$ is the set of lots which are processed in $T(n)$. Fig. 9 shows not only the allocated capacity but also the utilization of the tools. From Eq. (11), the utilization of the tool m is defined as

$$e_{T(n)}(t) = \frac{AC_{T(n)}(t)}{UC_{T(n)}(t)} \tag{16}$$

where $AC_{T(n)}(t)$ is the allocated capacity of the tool n at day t.

5.4 Bottleneck analysis

Only the tool groups, which block the lots, are listed in Fig. 10. It can be found that the tool groups 3 and 7 are critical to the system. Over 10 lots are blocked in these two tool groups and they are called the bottleneck tool groups. In order to resolve the bottleneck tool groups 3 and 7, three and two new tools, for example, are added to the tool groups 3 and 7, respectively. The result shows that the bottleneck tool group 3 is no more the bottleneck tool group, while the number of the blocked lots of the tool group 7 is reduced from 15 lots to 6. Note that it is only a reference for the manager. The bottleneck responsibility of tools is shown in Fig. 11. The manager can identify the high-utilization tools and decide whether the PM schedule of the tool can be adjusted or not.

5.5 Tool-dispatching rule analysis

The six tool-dispatching rules described in section 4 are compared. Number of blocked lots, number of blocking tool groups, moves and WIP are the performance index to evaluate the rules. Fig. 12 shows the blocked lots after one day later. Fig. 13 shows the number of tool groups that block the lots. Fig. 14 shows the total moves in 1998/12/31. Fig. 15 shows the WIP one day later. In the above figures, it is obvious that MFCR is better than others, especially in "Blocked Lots" and "Moves". As a result, MFCR is used as the tool-dispatching rule.

6. Conclusion

This paper proposes a new approach to allocate the capacities of the tools for their tool groups. Besides, the maximum fuzzy candidative ratio is proved better than other tool-dispatching rules. Besides, the priority-based algorithm is used to estimate the tool utilization, bottleneck, tool sharing in the tool groups, throughput of the tool groups and WIP in a specific day.

With the support of the on-line learning system, which provides the cycle time information of lots, the computation time is small for simulating one-day tool capacity allocation. In addition, the accurate estimation of the remaining lot cycle time assures the correctness of the tool capacity allocation procedure. Based on the approach proposed in this paper, the shop floor manager can solve the bottleneck tools and reschedule the PM schedule of the tools.

Acknowledgement

This work is partially supported by National Science Council, Taiwan, R.O.C. under grant number NSC 89-2218-E-002-004-, and Taiwan Semiconductor Manufacturing Co. Ltd.

References

[1] S.C. Chang, L.H. Lee, L.S. Pang, W.Y. Chen, Y.C. Weng, "Iterative Capacity Allocation and Product Flow Estimation for Scheduling Semiconductor Fabrication," *1995 IEEE/CPMT Int'l Electronics Manufacturing Technology Symposium*, pp.508-512, 1995.

[2] C. C. Cheng, "A Study of Lot Priority Setting for Wafer Fabrication," *Master Thesis*, Institute of Industrial Engineering, Chung Yuan Christian University, 1998.

[3] S. H. Chung, H. W. Huang, "The Design of Constraint-Oriented Target Planning System for Wafer Fabrication Factories," *Joint Conference of the 5th International Conference on Automation Technology*, Taiwan, 1998.

[4] B. Ehteshami, R.G. Petrakian, P.M. Shabe, "Trade-off in Cycle Time Management: Hot Lots," *IEEE Transaction on Semiconductor Manufacturing*, vol.5, no.2, pp.101-106, 1992.

[5] J.Y. Juang, H.P. Huang, "Queueing Network Analysis for an IC Foundry," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation San Francisco, CA,*

pp.3389-3394, April, 2000.
[6] C.H. Lu, D. Ramaswamy, P.R. Kumar, "Efficient Scheduling Policies to Reduce Mean and Variance of Cycle-Time in Semiconductor Manufacturing Plants," *IEEE Transaction on Semiconductor Manufacturing*, vol.7, no.3, pp.374-388, 1994.
[7] M.A. Quaddus, "A Generalized Model of Optimal Due-Date Assignment by Linear Programming," *Journal of Operation Research Society*, vol.38, no.4, pp.353-359, 1987.
[8] K. S. Tsakalis, "Hierarchical Modeling and Control for Re-entrant Semiconductor Fabrication Lines: A Mini-Fab Benchmark," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pp.508-513, 1997.
[9] G.L. Wu, K. Wei, C.Y. Tsai, S.C. Chang, N.J. Wang, R.L. Tsai, H.P. Liu, "TSS: a daily production target setting system for fabs," *Semiconductor Manufacturing Technology Workshop*,p p.86-98, 1998.
[10] C.Y. Yu and H. P. Huang, "Fab Model Based on Distributed Neural Network," *National Conference on Automation Technology*, ChiaYi, pp.271-277, 1999.
[11] C.Y. Yu, H.P. Huang, J.Y. Juang, C.F Yeh, "Cycle Time Estimation for Highly Product-Mixed Semiconductor Foundry," *The First Cross Strait Workshop on manufacturing*, Taipei, 2000.

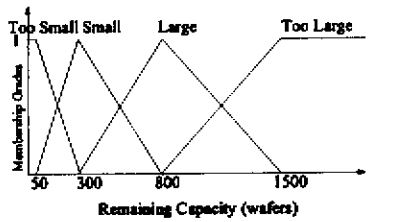

Fig.4 MF of Input 1 (remaining capacity)

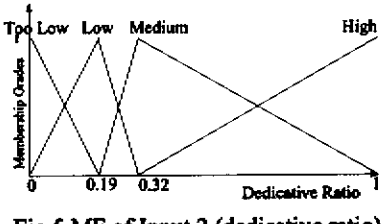

Fig.5 MF of Input 2 (dedicative ratio)

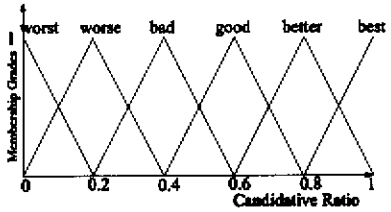

Fig.6 MF of Output (candidative ratio)

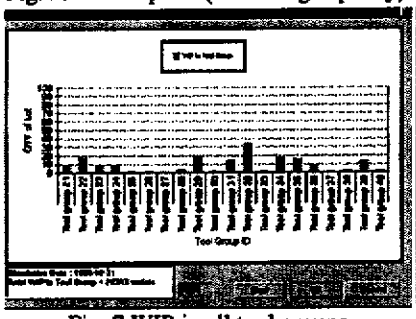
Fig. 7 WIP in all tool groups

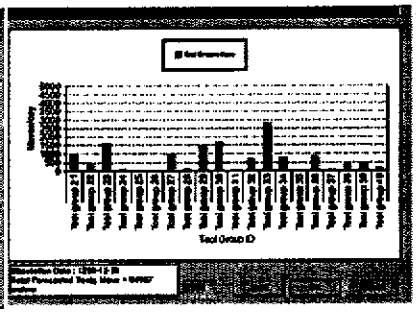
Fig. 8 Estimated tool group moves

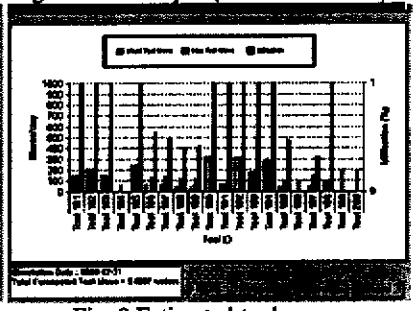
Fig. 9 Estimated tool moves

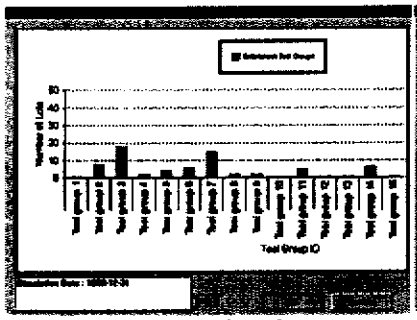
Fig. 10 Bottleneck tool groups

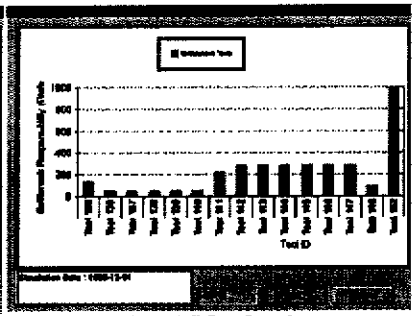
Fig. 11 BR of tools

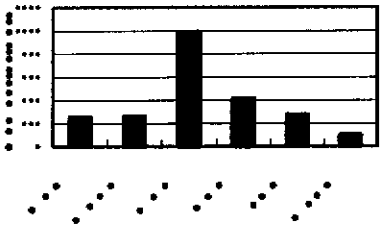
Fig. 12 Blocked Lots analysis

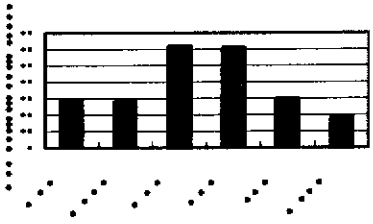
Fig. 13 Blocking Tool Groups analysis

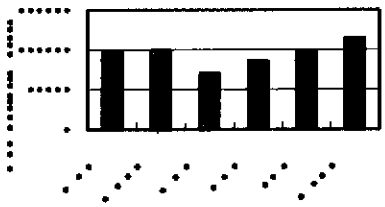
Fig. 14 Moves analysis

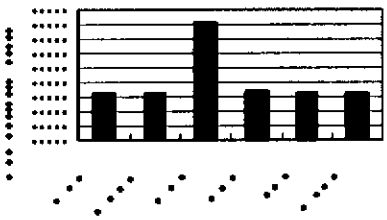
Fig. 15 Blocked Lots analysis

行政院國科會補助參加國際會議報告

2001 Intl. Symposium on Robotics

一. 參加會議經過

二. 與會心得

三. 參觀活動

四. 建議

五. 攜回資料

參加 2001 年國際機器人會議報告 (ISR 2001)

黃漢邦 教授　台灣大學機械系

一. 參加會議經過

本次會議在韓國漢城舉行，會議期間從四月十九日至四月二十一日。我於四月十八日從台北搭機前往漢城。從機場到飯店途中，頗驚訝漢城的改變。寬廣的街道、高聳的大樓、起伏的地形，彰顯韓國的企圖心。而街道上夾雜著小販，依稀又是台灣街道的景象。

本次大會在 Renaissance 飯店舉行，該飯店位於市中心，鄰近世貿展示中心。因飯店地處鬧區，入夜後仍見遊客熙來攘往，在街上行走還算安全。大會在四月十九日舉行開幕，韓國總理並親臨至辭。當天晚上舉行歡迎酒會，並舉行討論會，直到晚上九點半結束。四月二十日晚上為大會晚宴及專題演講，在十點半左右結束。四月二十一日是大會論文發表最後一天，隔天仍有討論會。我於二十二日從漢城搭飛機返回台北。

二. 與會心得

此次會議共有 440 篇論文發表，全部文章分 63 個場次發表，同一時段則有六個場次同時進行。本人在會中一共發表三篇文章，分別是"Adaptive Hybrid Impedance Force Control of Robotic Deburring Processes," "Remote Control of an Intelligent Wheelchair for Hospital Automation,"和 "Fuzzy CMAC Control for Soccer Robots."

本次會議安排三次大會特別演講，演講的主題分別是 "The Honda Humanoid Robot," "Paradigm Shift in Robot Based Automation Through Industrial IT," 及 "Embedded Robotics and MEMS."

從今年的論文發表及大會所安排的特別演講來看，有下面幾個特點：

1. 全方位的發展機器人 - 從工業、農業、營建和辦公室的應用，到娛樂、家庭、個人、軍隊、太空、醫療和海底的應用。這方面以日、美、德三國為發展重心。

2. 人形化機器人的發展 – 從研究走向娛樂及軍事運用。從 1999 年起也另外有一個探討人形化機器人的國際會議。

3. MEMS Technology ---以 micro sensor 及 micro actuator 為發展主軸，但如何組裝成一有用的裝置，為目前另一發展重點。另一方向是奈米技術及有機裝置的應用。

4. Soccer Robot 的發展 – 將 Robot 應用於娛樂。

三. 參觀活動

在開會期間，大會安排參觀 Samsung Electronics 及現代造船。我選擇前往參觀 Samsung Electronics。

四. 建議

本會議是機器人領域相當重要的會議之一，主要以應用為主，國科會每年應儘量補助所有論文發表者與會。

五. 攜回資料

大會論文集的光碟片一張及大會論文摘要一本。

Proceedings of the 32nd ISR(International Symposium on Robotics), 19-21 April 2001

Fuzzy CMAC Control for Soccer Robots

Han-Pang Huang* and Chao-Chiun Liang**

Robotics Laboratory, Department of Mechanical Engineering
National Taiwan University, Taipei, Taiwan 10660
Email: hanpang@ccms.ntu.edu.tw
TEL/FAX: (+886-2) 2363-3875
* Professor and correspondence addressee **Graduate student

Abstract

Based on the structure of Cerebellar Model Articulation Controllers (CMAC), this paper proposes a self-organizing fuzzy CMAC, the Model-based Fuzzy CMAC (MFCMAC), for the motion control of soccer robots. The objective is to control the soccer robots to track the planned trajectories in the unknown and dynamic environment of a robot soccer game. MFCMAC uses fuzzification at its input to increase its input resolution and effectively compress the input data. In addition, it performs an on-line model-identification, then uses the result to calculate the required modification for the output of MFCMAC, and adjusts its weights to obtain the desired solution. MFCMAC was applied to the control of a mobile robot to track the planned trajectories. In several experiments of tracking various trajectories, MFCMAC is significantly superior to the traditional PID controller.

Keywords: CMAC, Fuzzy CMAC, Fuzzy control, Learning, Soccer robots

Prooceding s d fne o

Albus proposed Cerebellar Model Articulation Controller (CMAC) to mimic the structure and functions of the human cerebellum [1]. CMAC possesses the following advantages over other neural network models: fast recall process, quick learning law, and rapid convergence speed. Based on the structure of CMAC, this paper proposes a self-organizing fuzzy CMAC, the Model-based Fuzzy CMAC (MFCMAC), for the motion control of soccer robots. A soccer robot system, which requires precise and fast controller response, is a good test bed for real time control. The objective is to control the soccer robots to track the planned trajectories in the unknown and dynamic environment of a robot soccer game. The characteristics of fast learning and convergence make CMAC a real-time controller. However, CMAC suffers from limitation in its input resolution. For practical applications, CMAC uses some method to reduce the required memory space. This causes some troubles in the learning process of CMAC, and hence reduces its performance. The proposed MFCMAC uses fuzzification at its input to increase its input resolution and effectively compress the input data. This can naturally reduce the required memory space and the troubles in the learning process. The learning law of traditional CMAC is supervised. This is not suitable for the soccer robot control and other applications where the desired output of CMAC is unknown. Therefore, it is necessary to extend the learning law of CMAC to unsupervised learning. To fulfill this requirement, the proposed MFCMAC performs an on-line model-identification, then uses the result to calculate the required modification for the output of MFCMAC, and adjusts its weights to obtain the desired solution. MFCMAC will be applied to the control of a mobile robot to track the planned trajectories. The simulations and experiments will be conducted to justify the performance of MFCMAC. It will be shown that MFCMAC is significantly superior to the traditional PID controller.

tro oh32ief e I dh2S onRϕ a 1 (o

The learning law of CMAC is presented as follows. Assume that the output of CMAC is $f(x)$ when the input is x. For each training pair (x', y'), the amount the output of the CMAC should be modified at each training-instant is:

$$\delta = \beta (y' - f(x')) \qquad (1)$$

where β is the learning rate and $0 \le \beta \le 1$. According to Eq.

(1), the weights for output i should be modified by the following equation:

$$\omega_{ij}(k+1) = \omega_{ij}(k) + \delta_i / c, \qquad j = 1, 2, \ldots, c \qquad (2)$$

where ω_{ij} is the j^{th} activated weight for the i^{th} output, δ_i is the i^{th} component of δ and a scalar representing the amount that the i^{th} output should be modified, and k is a positive integer representing the k^{th} training cycle. The learning law of CMAC can be further understood from Fig.1

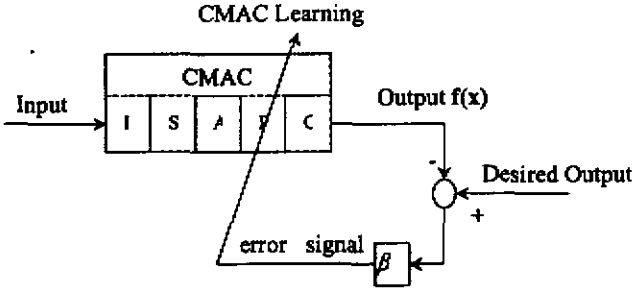

Fig.1 The block diagram of the CMAC learning

Considering Eqns. (1) ~ (2), it is evident that the learning law of CMAC are simple arithmetic operations and can be implemented in real-time. As to the convergence property of CMAC, Wong proved that CMAC learning always converges [2]. This fact can be stated by the following theorem:

Theorem 1 [2]: Given a set of training samples composed of input-output pairs from $R^n \rightarrow R^m$ CMAC always learns the training set with arbitrary accuracy if the input space is discretized such that no two training input samples excite the same set of association cells.(4.3)

yroomsppud(a 1 (o

Structure of Fuzzy CMAC

The structure of fuzzy CMAC is shown in Fig.2. The example of fuzzy CMAC in Fig.2 has 2 inputs, 1 outputs, and 3 active elements. From Fig.2, it is apparent that a fuzzy CMAC also contains five layers. The functions and operations of the five layers are described in the subsequent section. They are different from those in CMAC.

There are also four mappings in a fuzzy CMAC. They are different from those in CMAC. The four mappings are formulated by the following four equations:

$$\Theta_f: I_f \rightarrow S_f \qquad (3)$$

$$\Phi_f: S_f \rightarrow A_f \qquad (4)$$

$$\Gamma_f: A_f \rightarrow P_f \qquad (5)$$

$$\Omega_f: P_f \rightarrow O_f \qquad (6)$$

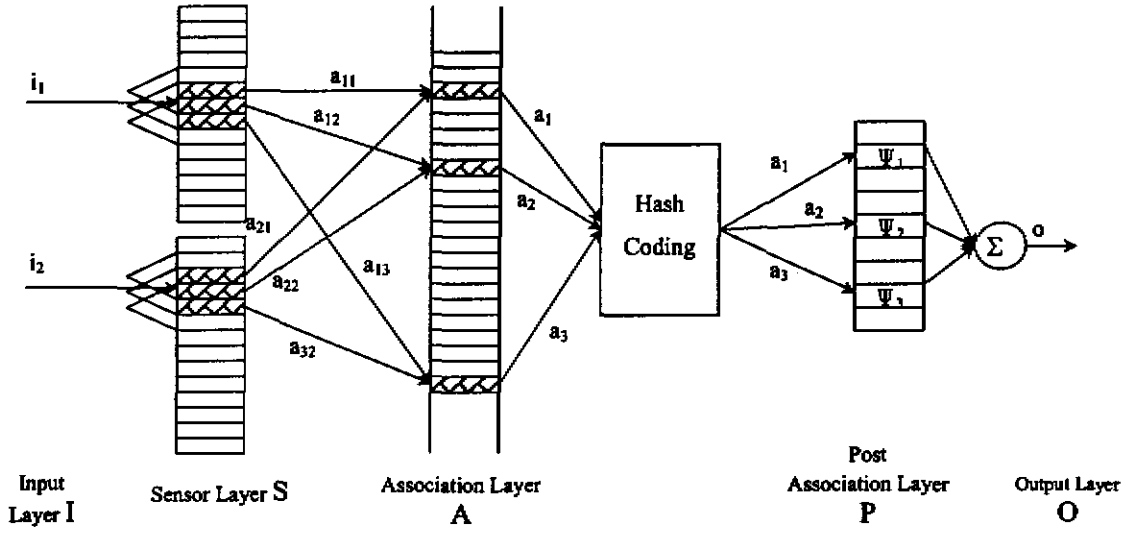

Fig.2 The structure of fuzzy CMAC with 2 inputs, 1 outputs, and 3 active elements

where I_f, S_f, A_f, P_f and O_f represent the input layer, sensor layer, association layer, post association layer, and output layer, respectively. Unlike the sensor layer S in a CMAC, the sensor layer S_f in a fuzzy CMAC has a number of fuzzy sets arranged on each input dimension so that every input will activate c fuzzy sets.

Recall Process of Fuzzy CMAC

The recall process of the proposed fuzzy CMAC consists of four mappings. Using the fuzzy CMAC in Fig. 2 as an example, the four mappings are introduced as follows.

(1) $I_f \rightarrow S_f$: This is the mapping from the input layer to the sensor layer. Assume the input $\mathbf{x}$ is an n-dimension vector: $\mathbf{x} = (d_1, d_2, \ldots, d_n)$, there are n sensors in the sensor layer to sense n inputs. The mapping performs the following two steps:

Step 1: For each d_i, i=1, 2, ..., n, activate c fuzzy sets F_{ij}, j=1, 2, ..., c. Due to the arrangement of fuzzy sets in Fig.2, at most c fuzzy sets are activated. Each activated fuzzy set F_{ij} is represented by a memory element in the sensor layer and the membership grade a_{ij} to which the input belongs.

Step 2: Find the corresponding membership grades a_{ij}, j=1, 2, ..., c.

(2) $S_f \rightarrow A_f$: This is the mapping from the sensor layer to the association layer. By concatenating the corresponding activated elements of the sensors in the input layer, c elements in the association layer are activated [2]. The firing strength a_i, i=1, 2, ..., c, of the i^{th} activated memory element is calculated by the following min-operator:

$$a_i = \min(a_{i1}, a_{i2}, \ldots, a_{in}) \tag{7}$$

(3) $A_f \rightarrow P_f$: This is the mapping from the association layer to the post association layer. The proposed fuzzy CMAC uses the same hash-coding technique to reduce the required memory elements. The c elements activated in the association layer activate c elements for each output in the post association layer. The contents of the post association layer are fuzzy sets. For the i^{th} output, c fuzzy sets Ψ_{ij}, j=1, 2, ..., c, in the post association layer will be activated. The firing strength of Ψ_{ij} is a_j j =1, 2, ..., c.

(4) $P_f \rightarrow O_f$: This is the mapping from the post association layer to the output layer. If the fuzzy sets in the post association layer are all fuzzy singleton [3] and the values with full membership are represented by $|\Psi_{ij}|$, the value o_i of the i^{th} output can be given by the following equation:

$$o_i = \sum_{j=1}^{c} a_j |\Psi_{ij}| \tag{8}$$

Otherwise, the value o_i of the i^{th} output should be calculated by centroid method of defuzzification given by the following equation:

$$o_i = \sum_{j=1}^{c} A_{ij}(a_j) |\Psi_{ij}| \tag{9}$$

where $A(a_j)$ is the area of the fuzzy set Ψ_{ij} cut by the membership grade a_j, as shown in Fig.3.

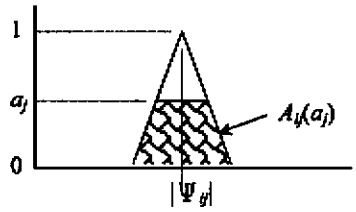

Fig.3 The area of a fuzzy set cut by the firing strength a_j

Learning Law of Fuzzy CMAC

The learning law of the proposed fuzzy CMAC is presented as follows. Assume that the output of CMAC is $\mathbf{f}(\mathbf{x})$ when the input is $\mathbf{x}$. For each training pair $(\mathbf{x}', y')$, the amount of the output of the CMAC is modified at each training-instant is:

$$\delta_f = \beta (y' - f(x')) \qquad (10)$$

where β is the learning rate and $0 \le \beta \le 1$. According to Eq. (10), the fuzzy sets Ψ_j, j=1, 2, ..., c, for output i is modified by the following equation:

$$|\Psi_j(k+1)| = |\Psi_j(k)| + \frac{a_j \delta_{fi}}{\sum_{j=1}^{c} a_j}, \qquad j = 1, 2, \ldots, c \qquad (12)$$

where δ_{fi} is the i^{th} component of δ_f and a scalar representing the amount the i^{th} output should be modified; k is a positive integer representing the k^{th} training cycle.

Advantages of Fuzzy CMAC

From the description of the mapping Θ_f: $I_f \rightarrow S_f$, it is apparent that fuzzy CMAC does not need quantization operation. This feature not only increases the input resolution of the fuzzy CMAC but also reduces the computation time of one recall cycle by eliminating the computation load for quantization operation. A traditional CAMC increases its resolution by increasing the number of memory elements. This approach not only increases the computation load for recall process but also raises the difficulty of learning convergence. A fuzzy CMAC, however, increases its resolution due to its nature. This approach is more reasonable than traditional CMACs and does not require extra memory elements. As a fuzzy CMAC increases its resolution naturally, it needs less memory elements for keeping the performance than the tradition CMAC. Therefore, a fuzzy CMAC can save storage memory, speed up the recall process, and reduce the collision problem.

Hash-coding is not the only way to reduce the number of the required memory elements. The clustering technique can be applied to the input space to reduce the number of fuzzy sets required to cover the input data. For an n inputs case, if the number of fuzzy sets for each input dimension can be reduced by a ratio of 2, the number of the required memory elements is reduced by a ratio of 2^n. If this reduction-ratio is not enough, two or higher dimensional clustering can be applied and the reduction-ratio will be even more large. This fact is summarized in the following theorem.

Theorem 2: The reduction-ratio for the number of the required memory elements increases exponentially with the increase of the number of the cluster dimension.

broo(n),2 ifelo ms ppuo(a 1 (o dho 2o1 39RAil2 e fpfeI o ms ppuo(n e din993io

It is interesting to compare the proposed fuzzy CMAC with the Linguistically Approached Self-Organizing fuzzy logic Controller (LASOC) proposed in [4, 5].

Using an input vector as an address, a fuzzy CMAC produces an output vector in response to the input vector in a look-up table fashion. Similarly, LASOC has a rule table and uses an input vector to find the activated rules from the rule table to compute an output vector. Therefore, fuzzy CMAC and LASOC use similar ways to find an output vector from an input vector. Furthermore, the fuzzy sets in the post association layer of a fuzzy CMAC correspond to the fuzzy sets in the rule table of its corresponding LASOC. As to the fuzzificaiton and defuzzification of a fuzzy CMAC, they are the same as those of its corresponding LASOC.

There are several points where a fuzzy CMAC is different from its corresponding LASOC. First, the learning law of the proposed fuzzy CMAC is supervised learning, whereas the corresponding LASOC is unsupervised learning. Second, the structure of the proposed fuzzy CMAC is suitable for MIMO applications, while the corresponding LASOC is not. Third, the structure of the proposed fuzzy CMAC is appropriate for parallel implementation, while the corresponding LASOC is not. Finally, the proposed fuzzy CMAC has the collision problem, while the LASOC does not have.

Based on the different points discussed above, it is straightforward to think that fuzzy CMAC and LASOC can mutually reinforce. The structure of fuzzy CMAC can be used to implement LASOC in MIMO applications and/or parallel processing. On the other hand, fuzzy CMAC can use the learning law of LASOC to carry out the unsupervised learning. In the subsequent section, this paper applies the learning law of LASOC to implement model-based unsupervised learning.

0roo l 39R-Ail2 e fpfe lomspp uc(a 1 (Ho

The recall process of the proposed Model-based fuzzy CMAC (MFCMAC) is the same as that described in Fig.2

The unsupervised learning law of MFCMAC is

presented as follows. The block diagram of the learning law of MFCMAC is shown in Fig.4. In unsupervised learning, only the input to MFCMAC and the desired output of the plant are known, whereas the desired output of MFCMAC is not known. Therefore, the error signal e is formed by the desired output vector y and the plant output vector y′:

$$e = y - y' \tag{12}$$

The error signal vector e is sent to the performance evaluator in Fig.4 to evaluate the current performance of the plant. The performance evaluator computes a performance index vector Δo from e by the following equation:

$$\Delta o = P(e) = \gamma[(1-\zeta)e + \zeta T\, \Delta e] \tag{13}$$

where ζ is a weighting factor to represent the relative importance between e and Δe, γ is the learning rate, and T is the sampling period. The Model Identifier in Fig.4 receives the input and output data of the plant and on-line identifies the model matrix **M** of the plant. Through the identified model matrix **M**, the performance index vector Δo is then transformed into the amount Δu the output of MFCMAC should be modified. This transformation is carried out by the following equation:

$$\Delta \mathbf{u} = \mathbf{M}^{-1}\Delta o \tag{14}$$

Having Δu, the fuzzy sets in the post association layer of MFCMAC can then be modified by the following equation:

$$|\Psi_{ij}(k+1)| = |\Psi_{ij}(k)| + \frac{a_j \Delta u_i}{\sum_{j=1}^{c} a_j}, \qquad j = 1, 2, \ldots, c \tag{15}$$

where Ψ_{ij} and a_j have the same meanings as those defined previously, and Δu_i is a scalar and the i^{th} component of Δu.

*roo l f)s %2dfneHo2egoCL,3 i f) 3edHo

The simulation is based on the soccer robot for FIRA MIROSOT games. Each team has three member robots and each robot is a bi-wheeled mobile robot conformed to cubic 7.5 cm volume limit. These robots are controlled by visual servoing, which feedbacks the robots' positions and orientation, and the computer calculates the desired velocity of the wheels and transmits the commands to the robots via radio. Maneuverability is essential for this application, thus, fast and precise control of the robot is the goal of applying this control scheme to the soccer robot motion control.

The simulation programs were written in Visual C++ and implemented on a Pentium III 450 PC. For displaying the 3-D characteristics in the simulations, OpenGL is adopted as the API for graphics hardware.

Tracking

Tracking the planned curves is the fundamental skill of mobile robots. Based on this fundamental skill, most advanced skills can easily be implemented. This paper controls the angular velocities of the two wheels of a mobile robot to make it to track the planned curves. Assume that the relative distance and angle of the current desired position to the mobile robot are $Dist_{dr}$ and Ang_{dr}, the difference and summation of Ang_{dr} are $DiffAng_{dr}$ and $SumAng_{dr}$ respectively, and the angular velocities of the right and left wheels are $AngVel_r$ and $AngVel_l$ respectively. The PID control law can then be implemented by the following equations:

$$AngVel_r = K_e \times Dist_{dr} - K_p \times Ang_{dr} - K_d \times DiffAng_{dr} - K_i \times SumAng_{dr}$$

$$AngVel_l = K_e \times Dist_{dr} + K_p \times Ang_{dr} + K_d \times DiffAng_{dr} + K_i \times SumAng_{dr} \tag{4.16}$$

where k_e, k_p, k_d, and k_i are the constant gains to emphasize the importance of their corresponding terms. These gains k_e, k_p, k_d, and k_i are set as 3, 0.2, 0.15, and 0.1. The PID control law is applied to control a soccer robot to track a sine curve. The root-mean-squared error in position caused by the PID controller is 0.077129 m. This error is mainly due to the time delay in tracking the sine curve. The proposed MFCMAC is then combined with the PID controller to control the robot to track the same curve. After 5 runs of training, the result is shown in Fig. 5

The root-mean-squared error in position caused by the proposed controller is 0.030665 m. The error due to the time delay is reduced significantly by MFCMAC. This result demonstrates the fast learning and convergence capacities of MFCMAC.

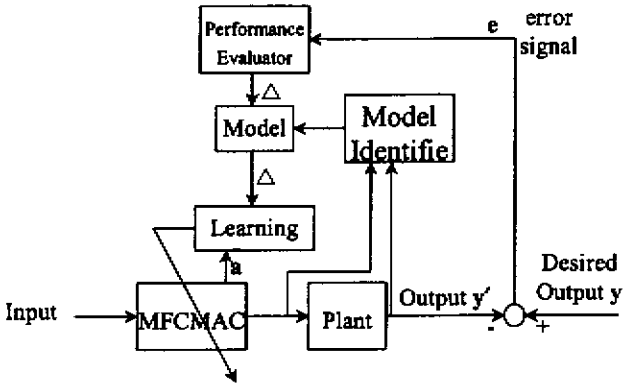

Fig.4 The learning law of MFCMAC

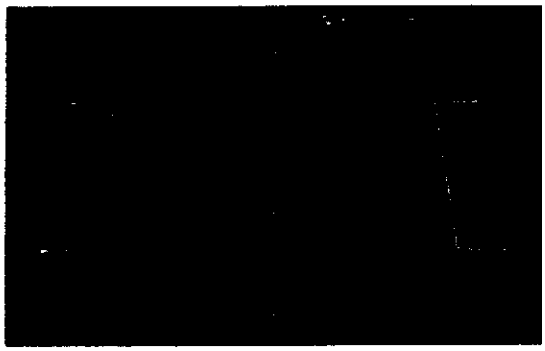

Fig. 5 Combine MFCMAC and the PID control law

Drox(ne 9 s Hfi eo

A self-organizing fuzzy CMAC, the Model-based Fuzzy CMAC (MFCMAC), is proposed in this paper for the motion control of soccer robots. Comparing to traditional CMAC, MFCAMC increases the resolution of the input space, reduces the required memory space and collision problem, and extend the learning law to unsupervised learning. The simulation and experiment results reveal that MFCMAC is significantly superior to the traditional PID controller.

M3Bi3 e 3 Ho

[1] J. S. Albus," A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," Trans. ASME, J. Dynamic Syst. Meas. Contr., Vol. 97, pp. 220-227, 1975.

[2] Y.-F. Wong and A. Sideris, "Learning Convergence in the Cerebellar Model Articulation Controller," IEEE Trans. on Neural Network, Vol. 3, No. 1, pp. 115-121, January 1992.

[3] D. Duboid and H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press Inc., 1980.

[4] C. C. Liang," Self-Organizing Fuzzy Force Control of Manipulators," Master Thesis, Department of Mechanical Engineering, National Taiwan University, 1993.

[5] C. J. Yang, "A Real-Time Linguistically Approached Self-Organizing Fuzzy Logic Controller - LASOC," Master Thesis, Department of Mechanical Engi-neering, National Taiwan University, 1992.

[6] H. P. Huang and C. C. Liang, "Robotic Deburring by Using Learning Fuzzy Controllers," The Third International Conference on Automation Technology, Vol. 5, pp.19-25, 1994.

[7] M. Hormel, "A Self-Organizing Associative Memory System for Control Ap-plications," Advance in Neural Information Processing Systems 2, D. S. Touretzky Ed., Los Altos, CA: Morgan Kaufmann, pp. 332-339, 1990.

[8] M. J. Carter, F. J. Rudolph, and A. J. Nucci, "Operational Fault Tolerance of CMAC Networks," Advance in Neural Information Processing Systems 2, D. S. Touretzky Ed., Los Altos, CA: Morgan Kaufmann, pp. 340-347, 1990.