ALGORITHM FOR CONSTRUCTING A RECONFIGURABLE MANUFACTURING SYSTEM

Hsiang-Hsi Huang

Department of Industrial Management, National Pingtung University of Science and Technology
No.1, Shuefu Road, Nei-Pu Township, Pingtung 91201, TAIWAN, R.O.C.
Tel: +886-8-7703202 ext. 7665; Fax: +886-8-7740321
E-mail: hhuang@mail.npust.edu.tw

Abstract: Although severe economic impacts due to the global financial crisis in recent years, many believe that organizations and enterprises need to reconsider changes in systems and products innovation. In this paper, a system procedure for constructing a reconfigurable manufacturing system is provided. The Petri Nets theory and a system theory approach are being used to design rule-based discrete-event controllers for the sequencing of jobs in manufacturing systems. The controller is described in terms of matrix equations that are easy to implement on a personal computer. The concepts of Petri nets (PN) and the Industrial Engineering (IE) techniques are included and described into matrix forms. Within the IE techniques, the standard bill of materials (BOM) is used in the first design step to make a "task sequencing matrix." Then a resource requirement matrix is constructed to add non-shared resources and shared resources (e.g. pallets, transport robots and material handling machines.) Non-shared resources are controlled using inner decision loops. However, shared resources require outer decision loops for dispatching and routing that resolve conflicts, taking into account the specified performance measures to be optimized (e.g. percentage of idle time, throughput, etc.) The rule-based controller design algorithm is a step-by-step procedure with repeatability. The matrix formulation allows a rigorous analysis of deadlock in terms of circular wait and blocking, and the numbers of resources available.

Keywords: Petri nets, bill of materials, task sequencing, rule-based controller, manufacturing dispatching.

INTRODUCTION

Recently, severe economic impacts due to the global financial crisis, nevertheless, many believe that organizations and enterprises need to reconsider changes in systems and products innovation. To meet the needs of customers today, enterprises must find a better way of producing products. Hence, manufacturing system's behaviors obtain our focus in research. Manufacturing system is concerned as the discrete event (DE) systems which exist in high popularity today and have many vigorous research developments. Those systems include the automated manufacturing systems (AMS), highway traffic control systems, the internet, multi-platform battle groups, coordinated robotic groups, wireless communication nets, etc. Different approaches have been developed to modeling, controls design, and simulation for these discrete event systems[1], including Petri nets[4,8,10,11,12,19,21], control theoretic techniques[15,17,18], expert systems design[3,12], alphabet-based approaches[22], perturbation methods[2,7], and so on. It is unfortunately that there is no unified methodology that brings these techniques together while relating them to standard industrial engineering (IE) techniques for the design of shop-floor manufacturing dispatchers and routers. Those IE techniques include, for example, the bill of materials (BOM)[5,7], Steward's sequencing matrix[23], the resource requirement matrix[16], assembly trees[26], operation sequence[7,24], and existing dispatching algorithms[20]. A competing manufacturing system should provide the potential competition ability for business in surviving in today's world wide competitive environment. The flexibility and agility rely on the ability of the flexible and automated manufacturing systems to fast reconfigure the new products and their operation processes and real time re-program the AMS controller.

This paper presents a streamlined design algorithm for a novel matrix-based formulation for manufacturing sequencers/dispatchers for the reentrant job shop. The design procedure is repeatable, so those two sensible engineers working independently will reveal the same controller structure. Given the sequencing matrix, as used by Steward[23] and the resource requirement matrix, as used by Kusiak[16], the discrete event manufacturing controller matrices can be directly written down. The controller consists of inner loops where no shared resources are involved, and outer shared-resource loops that require some conflict-resolution decision-making. These decisions are made by outer dispatching loops and routing control loops that are based on standard IE algorithms. The resulting multi-loop rule-based controller is very convenient for computer simulation, actual implementation, and modification when performance goals are modified, resources change, or failures occur.

MATRIX FORMULATIONS IN CONTROLLER DESIGN

The following discussions are to show how to use the industrial engineering techniques to directly write down a matrix-based logical dispatching/routing controller. Demonstration of industrial engineering techniques including the bill of materials (BOM), the assembly tree/operation process tree, Steward's sequencing matrix, the route sheet, the resource requirement matrix, and dispatching rules which related to the shop-floor design are provided here. Combine these well understood techniques within the manufacturing engineering community into an overall design and analysis technique that offers rigorous algorit[illegible]nd dispatching controllers w[illegible]aranteed performance in terms of deadlock avoidance. The rigorous repeatable r[illegible]AMS design algorithm is d[illegible]ped starting with standard IE techniques, a matrix rule-based controller then can[illegible]structed that has inner-loo[illegible] non-shared resources, and outer decision-making loops for shared resources[illegible]nified controller has a mul[illegible] hierarchical structure based upon matrix formulations.

Sample

A[illegible] truction.
A standa[illegible] n shown in Figur[illegible]

Figur[illegible] (tasks

I[illegible] resource pools m[illegible] llel part paths, ir[illegible] ong one part patl[illegible] t path in the figu[illegible] We say that cert[illegible] oss parts of differ[illegible] in their processi[illegible] shared resource[illegible] re 3, is a crucial c[illegible]

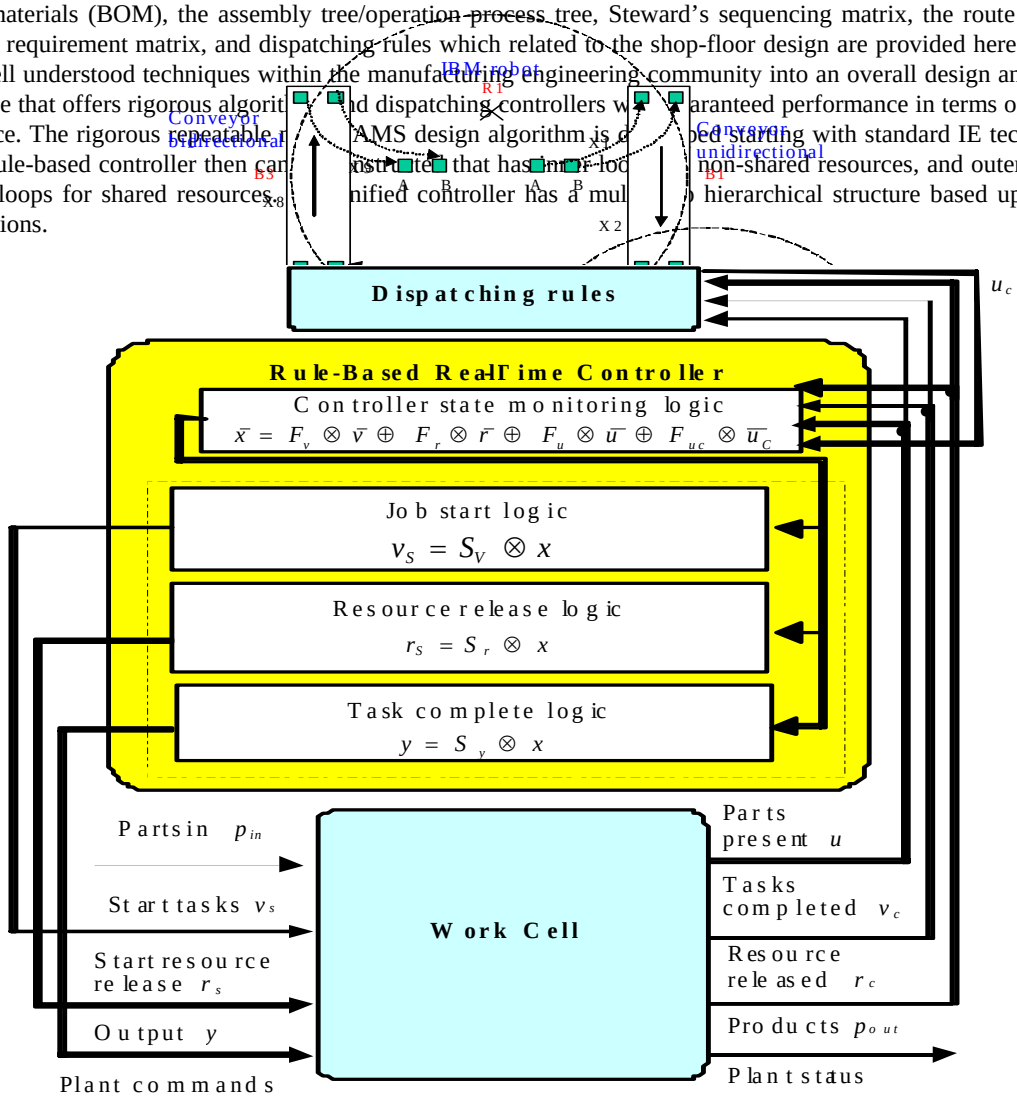


Figure 3: Matrix-based task controller with resource allocation

Review of IE Techniques

The BOM is a document containing the assembly/subassembly relationships for a specified product line [5,24]. It is usually referred to as a structural parts list and may be considered as a lookup table or a matrix *B*, whose elements *B*(*i*,*j*) are equal to 1 if subassembly *j* is required to produce subassembly *i*. This is identical to the definition of an assembly tree [26], an example of which appears later in Figure 4a. The BOM/assembly tree is generally known for a given product line. The operation process tree is an expansion of the assembly tree in that it includes all operations and inspections performed on the parts needed to be manufactured. When producing a single part without assembly, the assembly tree can be seen the same as the operation process tree.

Task Sequencing Matrix:

Steward's sequencing matrix[23] is a matrix *S* whose elements *S*(*i*, *j*) are equal to 1 if task *j* must be completed as a prerequisite for performing task *i*. This is the same as the definition of the BOM/assembly tree. In fact, in both the

assembly and operation process trees, the nodes may also be interpreted as jobs to be accomplished to produce a given product. Thus, in Figure 4a, job $R1U_1$ representing a part is transported by robot 1 the first time must be completed as precondition for performing job M1P. The sequencing matrix has been extensively analyzed by Warfield[25], who has shown that from it can be deduced the hierarchical structure of a manufacturing system. It has been used in applications by Eppinger *et. al.*[6] and others.

Route Sheet and Resource Requirement matrix:

The route sheet (or operation sheet) is also a document gathering the typical information of operations for producing a part, equipments or resources used for performing related operations, and processing time for each operation. A separate route sheet is normally required for each part that needs to be manufactured by company itself instead of purchasing from others. The route sheet provides the resource information needed for the resource requirement matrix[7]. The resource requirement matrix is a matrix R whose elements $R(i, j)$ are set equal to 1 if resource j is required for job i. The resources may include machines, robots, fixtures, tools, transport devices, and so on.

Dispatching Rules

We say a job is activated or fireable if all the preconditions for its performance are met. The problem of dispatching is that of allocating shared resources in the event that multiple jobs that require the same shared resource are activated. Dispatching can be viewed as a decision made by a resource on which jobs to accept (or parts to process). There are numerous dispatching rules [17,18,20] including first in first out (FIFO), earliest due date (EDD), select the job with the shortest operation time, last buffer first serve (LBFS), and so on. Though rigorous analyses of the performance of different dispatching rules are sparse, it is known for instance that LBFS yields bounded buffer lengths and minimizes the mean of the part delay (cycle time). EDD on the other hand minimizes the variance of the part delay. A small mean delay equates to small work-in-process (WIP), while a small variance of the delay guarantees that the system will reliably meet due dates. Given a performance measure specified by the user, one wishes to select the dispatching rule that optimizes this index. Performance measures might include machine percent idle time, part throughput time, buffer lengths, WIP, percent of jobs late, and so on.

Matrix-Based Controller Design

A rule-based matrix controller is now discussed that permit assembly/job sequencing, then addition of resources, and dispatching controller design. The matrix controller will be illustrated using a multiple-part-paths-reentrant-flow-line design example as indicated in section 2.1. Though the example is simple, the technique extends directly to more complicated systems using the notions of block matrix (e.g., subsystem) design. The AMS matrix controller is based on a matrix formulation where each matrix has a well-defined function for job sequencing, resource assignment, and resource release. In fact, the design algorithm uses task sequencing matrix (found from the BOM or assembly/operation process trees) and the manufacturing resource requirement matrix. This matrix-based logical controller has the multi-loop feedback control structure shown in Figure 3, with inner loops where there are no shared resources, and outer loops containing shared resources where dispatching and/or routing decisions are needed. It allows for shared-resource conflict resolution to provide guaranteed deadlock avoidance. Then, as is standard in industrial engineering, dispatching can be performed in accordance with prescribed performance objectives such as minimum resource idle time, maximum throughputs, task priority orderings, task due dates, minimum time of task accomplishment, and so on as prescribed by the user. In this multi-loop control structure, the inner loops can be considered as decision-free PN where no decisions are needed, and the outer decision-making loops can be considered as PN supervisors in the manner of Krogh and Holloway[14] and Ramadge and Wonham[22]. The matrix controller is described by the following equations:

Controller State Equation: $$\overline{x} = F_v \otimes \overline{v}_c \oplus F_r \otimes \overline{r}_c \oplus F_u \otimes \overline{u} \oplus F_D \otimes \overline{u}_D \quad (1)$$

Job Start Equation: $$v_s = S_v \otimes x \quad (2)$$

Resource Release Equation: $$r_s = S_r \otimes x \quad (3)$$

Product Output Equation: $$y = S_y \otimes x \quad (4)$$

These are logical equations and therefore are binary; i.e. matrix elements are either 1 or 0. The controller state equation (1) is a collection of rules so that it is formally a rule base. These equations are carried out in negative logic. The "overbar" denotes logical negation by applying the negation and de Morgan's laws. The logical negation can be defined as follows: given a natural number vector $\boldsymbol{n}$, its negation is $\overline{\boldsymbol{n}}$ such that $\overline{\boldsymbol{n}}(\boldsymbol{i}) = 0$ if $\boldsymbol{n}(\boldsymbol{i}) > 0$, and 1 otherwise. Hence, "0" entries in $\overline{v}_c$ and $\overline{r}_c$ indicate jobs complete and resources idle, respectively. And entries of "1" in v_s, r_s and y therefore indicate the jobs to be started, resources to be released and part output, respectively. A computed entry $x_i = 0$ (i.e., $\overline{x}_i = 1$) of the logic controller state vector $\overline{x}$ represents that all the conditions required to satisfy and fire the rule *i* are met. It is worth noticing that all the matrix operations here are defined in or/and algebra, so that the standard addition and multiplication are now replaced by "⊕" and "⊗" which representing the "or" and "and" operations.

It is convenient here to refer now to Figure 3 where the work-cell (also representing the PN places) is separated from the controller (PN transitions). The controller observes the status outputs of the work-cell, namely, v_C, representing "completed jobs" and r_C, representing "resources currently available or idle". The controller state equation (1), analogous to the matrix differential equation $\dot{x} = Ax + Bu$ in control system theory, checks the conditions required for performing the next jobs in the flexible manufacturing system. Based on these conditions, stored in the logical controller state vector *x*, the job start equation (2) computes which jobs are activated and may be started and the resource release equation (3) computes which resources should be released (due to completed jobs). Then, the controller sends commands to the work-cell, namely, v_S, representing jobs to be started, and r_S, representing resources to be released. Products out are given by Eq. (4). (Subscript *c* denotes 'complete' or 'current status'; *s* denotes 'start').

These equations are easy to write down, for F_V is the job-sequencing matrix and it can be determined from the BOM or assembly/flow process trees. F_R is the resource requirement matrix, which is assigned by the shop-floor engineer. Matrices S_V, S_R and S_Y are equally direct to write down (see subsequent example). Input u represents raw parts entering the cell. Input u_D is a conflict resolution input that selects which jobs to initiate when there are simultaneous requests involving shared resources; this dispatching input is selected in higher-level control loops using well-known techniques [20]. In fact, conflicts can occur in both the situations in choice tasks and shared resources. This means that the situations may occur when the job-sequencing matrix F_V and the resource requirement matrix F_R both have multiple "1"s in the same column. Sequencing matrix F_V and the resource requirement matrix F_R have long been used as heuristic design aids by industrial engineers, with some possibility for limited analysis (as described, e.g., by Warfield[25] in the case of F_V and Kusiak[16] in the case of F_R). The matrix controller elevates these design techniques to formal computation elements. The effect of the or/and matrix algebra with negative logic in Eq. (1) and positive logic in Eq. (2)–(4) is that Eq. (1) is a rule base using 'and' clauses, while Eq. (2)~(4) are rule-bases using 'or' clauses. This is exactly what is required for dealing with shared resources. It is easy to show that with the "or/and" matrix algebra, the equations just given become mathematical equations that allow formal computation of the rules in the AMS controller. This allows (1) computer simulation and (2) computer implementation of the controller on an actual workcell. The operations required in the controller equations can easily be carried out using standard real-time control software on a personal computer, including MATLAB, MATRIXx, LabVIEW and ProModel. It is noted that the coefficient matrices in Eq. (1) are sparse, so that real-time computations are very easy even for large manufacturing systems. Moreover, Eq. (1) is nothing but a rule-base, so that the rules can be fired using efficient algorithms such as the Rete algorithm.

Task Sequencing of Producing Products

An assembly tree is given In Figure 4a, which shows the required sequence of actions (jobs) to produce a product 1. This assembly tree contains information analogous to the BOM; it does not include any resource information. Raw material of part 1 is loaded to machine center 1 (M1) for milling operation (M1P) by robot R1 from warehouse 1. Then was unloaded from M1 into machine center 2 (M2) by robot R2 for grinding operation (M2P). The result is the final product 1 which is now unloaded from M2 as product output by R1 again to store in the warehouse 2 (PO_1 denotes 'product out of part type 1). It is the same structure of an assembly tree for producing the product 2.

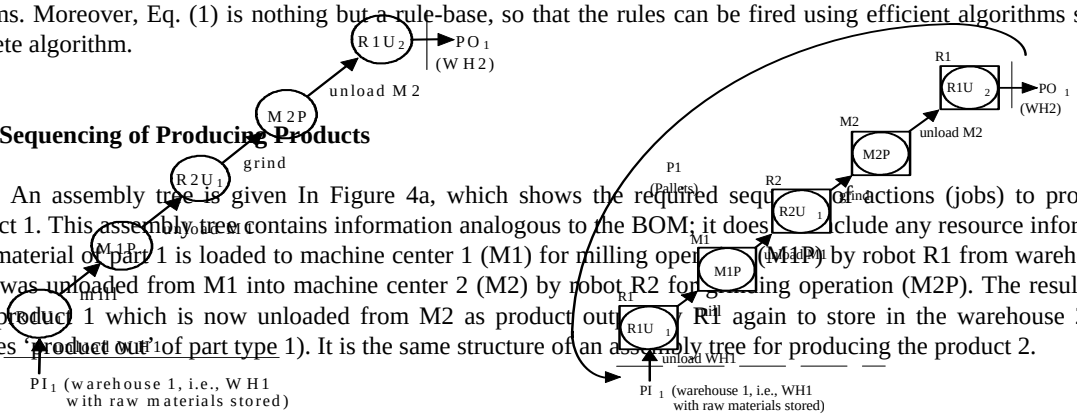
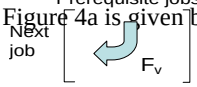


Figure 4: Product information: (a) Assembly tree, (b) Subassembly tree showing resources assigned to jobs.

Job Sequencing Matrix:

Referring to Figure 4a, define the job vector of part type 1 as $v1 = [R1U_1 \;\; M1P \;\; R2U_1 \;\; M2P \;\; R1U_2]^T$. Same for the job vector of part type 2, $v2$ is defined as $[R1U_1 \;\; M1P \;\; R2U_1 \;\; M2P \;\; R1U_2 R1U_3 \;\; M3P \;\; R1U_3]^T$. Then the job vector of the workcell is defined as $v = [v1^T \;\; v2^T]^T$. The Steward's sequencing matrix F_{v1} for the assembly tree of part 1 in Figure 4a is given by

(5)
$$F_{v1} = \begin{array}{c} \\ R1U_1 \\ M1P \\ R2U_1 \\ M2P \\ R1U_2 \\ PO \end{array} \begin{array}{c} \begin{array}{ccccc} R1U_1 & M1P & R2U_1 & M2P & R1U_2 \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}.$$

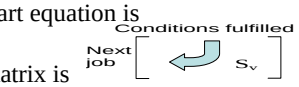
The mean of this matrix is 

In this matrix, an entry of 1 in position (i, j) indicates that job j must be completed prior to starting job i. F_{v1} is independent of available resources, and reflects the fact that the assembly tree imposes only a partial ordering on the sequence of jobs. Regardless of the resources available, F_{v1} will not change.

Controller State Vector and Job Start Equation:

The job vector v has two interpretations. As a status output of the work-cell, it denotes completed jobs; in this role it is denoted as v_c. On the other hand, as an input to the work-cell, it represents a job start command vector, denoted as v_s. To proceed, it is now necessary to define a controller state vector x, whose components are associated with the vector v = [R1U$_1$ M1P R2U$_1$ M2P R1U$_2$ PO$_1$| R2U$_2$ M3P R1U$_3$ PO$_2$]^T, that checks the conditions of the rules needed for job sequencing. Then, by applying notion of the block structure, the job start equation is

$$v_s = \left[\begin{array}{cccccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \otimes \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{array}\right] = S_v \otimes x.$$

Thus, the mean of S_V matrix is 

(6)

In the job start matrix S_v, an entry of 1 in position (i, j) indicates that job i can be started when component j of the controller state vector is active. In the reentrant flow line the matrix S_v has 1's on the diagonal. In the job shop with variable part routings it can have multiple entries in a single column, corresponding to different routing options.

The part output equation is
$$PO = \left[\begin{array}{c} PO_1 \\ \hline PO_2 \end{array}\right] \otimes x = \left[\begin{array}{cccccc|cccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \otimes x \equiv S_y \otimes x. \qquad (7)$$

Causal Ordering of Job Places:

It is important to order the job places correctly in the AMS controller formulation to obtain lower triangular matrices F_v, S_v [25], for then the sequencing of the jobs is causal. To obtain a causal ordering of the jobs, number the job places sequentially from left to right along each single part path. In the case of the multiple-part-paths reentrant flow line, numbering job places is based on the order of part types accordingly. Different part types represent different single part paths. After numbering the job places sequentially from left to right along part path 1, then continuously the numbering from left to right along part path 2, and so on (as shown in Figure 5).

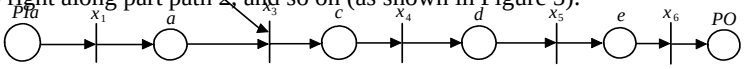


Figure 5:Job sequencing PN of assembly (no resources included).

As for the case of assembly, using Figure 5 as an example, parts *a* and *b* enter the workcell and are together assembled to produce part *c*, which is then drilled (part *d*) to result in part *e*, which is the cell output (PO denotes 'product out'). One can see from Figure 5, two part-paths merge together into one at the assembly point. Suppose part path j_1 is the complete path, with a partial part path j_2 merging into path j_1 at a transition on that path. In this situation, one may number the jobs of partial path j_2 from left to right, stopping at the last job prior to the assembly transition. Then, return to the beginning of path j_1, picking up the place ordering by numbering the job places of path j_1 from left to right. The transitions should be numbered corresponding to the job places they feed into. This procedure corresponds to numbering the jobs from bottom to top in an assembly tree. Note that each complete part path terminates in an extra transition that is required to produce the product output equations and to release the pallets, if any are used in that corresponding part path.

Adding the Resources and Conflict Resolution Input

To build a dispatching controller for shop-floor installation to perform this particular reentrant flow line task, the resources available must now be added. In Figure 4 is given a subassembly tree for product 1 of the reentrant flow line task, which includes resource requirements information. This information would in practice be obtained based on the

available manufacturing facility. The figure shows that part input PI_1 and part output PO_1 do not require resources. Pallets (P1) are needed for part 1 and its derivative operations. Robot R1 is responsible for loading work in process of part type 1 to machine center M1 for drilling, then robot R2 loads the drilled part to machine center M2 for milling, and finally unloaded by R1 again as final product 1 output to the finished product warehouse.

Note that robot R1 represents a shared resource for producing product 1, so that dispatching decision-making is needed when the two loading and unloading jobs are simultaneously requested, in order to avoid possible deadlock. There are scores of dispatching rules available and in standard use in manufacturing engineering [20] (for example, last buffer first serve, earliest due date, least slack time, clear the longest buffer, etc.). The Petri nets framework goes out of its way to obscure how these dispatching rules should be incorporated into work-cell controller design. We are showing here how the matrix controller in Figure 3 is derived in a step-by-step repeatable design algorithm in terms of standard job sequencing and resource matrices, and then dispatching rules are used in the outer loops to resolve shared-resource conflicts. This provides a repeatable design algorithm for discrete event controller design and shows how industrial engineering techniques fit into AMS controller design.

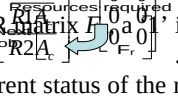
Resource Requirements (RR) Matrix:

The issue of required and available resources is now easily confronted as a separate engineering design issue from job sequence planning. As resources change or machines fail, the RR matrix is easily modified. Referring to Figure 4b and Figure 3, it is easy to define the resource vector as r = [M1A M2A M3A P1A P2A R1A R2A]^T, where 'A' denotes 'available'. By inspection, write down now the resource requirement matrix F_r and part input matrix F_u to obtain the complete controller state equation

$$\bar{x} = F_v \bar{v}_c \otimes F_r \bar{r}_c \oplus F_u u \quad (8)$$

$$= \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \bar{x}_4 \\ \bar{x}_5 \\ \bar{x}_6 \\ \bar{x}_7 \\ \bar{x}_8 \\ \bar{x}_9 \\ \bar{x}_{10} \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \otimes \begin{bmatrix} \overline{R1U_{1c}} \\ \overline{M1P_c} \\ \overline{R2U_{1c}} \\ \overline{M2P_c} \\ \overline{R1U_{2D}} \\ \overline{R2U_{2c}} \\ \overline{M3P_c} \\ \overline{R1U_{3c}} \\ \overline{R2U_{3c}} \\ \overline{R1U_{4c}} \end{bmatrix} \oplus \begin{bmatrix} 0&0&0&1&0&1&0 \\ 1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&1 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0 \end{bmatrix} \otimes \begin{bmatrix} u_{D1} \\ u_{D2} \\ u_{D3} \\ u_{D4} \\ u_{D5} \end{bmatrix} \quad (9)$$

$$\otimes \begin{bmatrix} \overline{M1A_c} \\ \overline{M2A_c} \\ \overline{M3A_c} \\ \overline{P1A_c} \\ \overline{P2A_c} \\ \overline{R1A_c} \\ \overline{R2A_c} \end{bmatrix} \oplus \begin{bmatrix} 1&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&1 \\ 0&0 \end{bmatrix} \otimes \begin{bmatrix} \overline{PI_1} \\ \overline{PI_2} \end{bmatrix} \oplus \begin{bmatrix} 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix} \otimes \begin{bmatrix} \overline{u}_{D1} \\ \overline{u}_{D2} \\ \overline{u}_{D3} \\ \overline{u}_{D4} \\ \overline{u}_{D5} \end{bmatrix}$$



F_v is the job-sequencing matrix from equation (5). In the RR matrix F_r a '1' in entry (i, j) indicates that resource j is needed to accomplish job i. Here the mean of matrix F_r is . The subscript c on the job vector denotes 'jobs complete', and on the resource vector r denotes 'current status of the resources'. Both v_c and r_c are status outputs of the work-cell. Again, Over-bars denote negation, so that, for instance the zero entries of $\bar{r}_c$ indicate which resources are idle (using negative logic). It is direct to show that using negative logic makes the conditions for the rules x_i into 'and' clauses, with zero elements of F_v, F_r becoming 'don't care' elements. For instance, the conditions for x_4 become, after some logical manipulation (deMorgan's rules): $\boldsymbol{x_4 = R2U_{1c}}$ **.AND.** $\boldsymbol{M2A_c}$, so that x_4 fires when jobs $R2U_1$ has been completed and resource M2 is available. According to equation (6), this now commands the initiation of the grinding operation that produces the final product of part type 1.

Conflict Resolution Input:

The extra input u_D is the shared-resource decision control input. It is needed since the resource requirement matrix F_r has multiple 1's in the last two columns, indicating that robots R1 and R2 are shared resources (c.f., Kusiak[16]). To resolve potential conflict and turn the controller into a 'decision-free' graph, it is therefore necessary to add the extra dispatching control input u_D, which is allowed to have only one of its entries high at any given instant for each robot. For robot R1, the high entry selects which of the three jobs, unload raw part type 1 from warehouse to M1, unload product 1 as output, or unload product 2 as output, will be performed in the event that three are requested simultaneously.

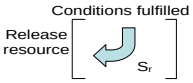
It is exactly in selecting u_D that the dispatching rules available in manufacturing texts[5,20] can be used to select which jobs to perform next in the event of simultaneous requests using the same resources. Thus, the dispatching rules belong in the outer loops of Figure 3. Our algorithm shows how to formally identify the control inputs needed by Krogh and Holloway[14] and Ramadge and Wonham[22], which are selected to avoid 'forbidden states' in PN design. Specifically, any columns in F_r containing more than a single 1 require an associated shared resource dispatching input (c.f., the use of F_r by Kusiak[16] in dispatching).

Resource Release Matrix:

The last issue to be resolved in this design is that of resource release. Thus, using manufacturing engineering experience and Figure 4b, select the resource release matrix S_r to be

$$\begin{bmatrix} M1A_s \\ M2A_s \\ M3A_s \\ P1A_s \\ P2A_s \\ R1A_s \\ R2A_s \end{bmatrix} = \begin{bmatrix} 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1 \\ 0&1&0&0&0&1&0&0&0&1 \\ 0&0&0&1&0&0&0&1&0&0 \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix}$$

Hence, the mean of the matrix S_r is



. (10)

where subscript 's' denotes a command to the workcell to start resource release. In this matrix, a '1' entry in position (*i*, *j*) indicates that resource *i* is to be released when entry *j* of *x* has become high (e.g., in this example, on completion of job *j*). It is important to note that rows containing multiple ones in S_r correspond to columns having multiple ones in F_r. For instance, the last row of S_r shows that R2A is a shared resource, since it is released after either x_4 is high or x_8 is high - that is, after either job $R2U_1$ or job $R2U_2$ is complete.

Thus the complete construction of the reconfigurable matrix framework for manufacturing and dynamic monitoring system is then developed.

CONCLUSIONS

A matrix-based formulation of AMS controllers was presented. The design procedure of the matrix controller is repeatable, so those two sensible engineers working independently will reveal the same controller structure. Complete simulation equations were provided for an AMS system. The matrices in the controller description come directly from industrial engineering tools such as the bill-of-materials, assembly tree, and resource requirement matrix. Using these matrices, algorithms will be given to compute certain critical conditions essential to the deadlock analysis later on, as well as certain critical subsystems within which work-in-process (WIP) should be limited to avoid deadlock.

Acknowledgment

This research was supported by the National Scientific Council Grants of NSC 94-2213-E-020-004- and NSC 97-2221-E-020-019-. We are grateful for the support from the NSC in TAIWAN.

REFERENCES

Cassandras (1993), C.G., *Discrete Event Systems*, Aksen Assoc. Inc., Boston.

Cao, X. and Ho, Y.C. (1990), "Models of discrete event dynamic systems," *Control Systems Magazine*, Vol. 10, No. 4, pp. 69-76.

Chen, C.L.P. and Wichman, C. (1992), "A CLIPS rule-based planning system for mechanical assembly," *Proceedings of. NSF DMS Conference*, Atlanta, pp. 837-841.

Desrochers, A.A. (1990), Modeling and Control of Automated Manufacturing Systems, *IEEE Computer Society Press*.

Elsayed, E.A. & Boucher, T.O. (1994), *Analysis and Control of Production Systems* (2nd Ed.), Prentice-Hall, Englewood Cliffs, NJ.

Eppinger, S.D., Whitney, D.E., and Smith, R.P. (1990), "Organizing the tasks in complex design projects," *Proc. ASME Int. Conf. Design Theory and Methodology*, pp. 39-46 (September).

Francis, R.L., McGinnis, L.F., and White, J.A., *Facility Layout and Location: An Analytical Approach* (2nd Ed.), Prentice Hall, Englewood Cliffs, NJ.

Gra$\breve{C}$anin, D., Srinivasan, P., and Valavanis, K. (1994), "Parametrized Petri nets: properties and applications to automated manufacturing systems," *Proc. IEEE Mediterranean Symp. New Directions in Control and Automation*, pp. 48-55 (June).

Ho, Y.C. and Cao, X. (1991), *Perturbation Analysis of Discrete Event Systems*, Kluwer, Boston.

Jeng, M.D. and DiCesare, F. (1992), "A synthesis method for Petri net modeling of automated manufacturing systems with shared resources," *Proc. IEEE Conf. Decision and Control*, pp. 1184-1189 (Dec.).

Jeng, M.D. and DiCesare, F. (1995), "Synthesis using resource control nets for modeling shared-resource systems," *IEEE Trans. Robotics and Automation*, Vol. 11, No 3, pp. 317-327 (June).

Kasturia, E., DiCesare, F. (1988), and Desrochers, A., "Real time control of multilevel manufacturing systems using colored petri nets," *Proc. IEEE Conf. Robotics Automat*., pp. 1114-1119.

Klein, M. (1991), “Supporting conflict resolution in cooperative design system,” *IEEE Trans. Sys., Man, and Cybernetics*, Vol.21, No. 6, pp. 1379-1390 (Nov./Dec.).

Krogh, B.H. and Holloway, L.E. (1991), “Synthesis of feedback control logic for discrete manufacturing systems,” *Automatica*, Vol. 27, No. 4, pp. 641-651 (July).

Kumar, P.R. and Meyn, S.P. (1993), “Stability of queueing networks and scheduling policies,” *Proc. IEEE Conf. Decision and Control*, pp. 2730-2735 (Dec.).

Kusiak, A. (1992), “Intelligent scheduling of automated machining systems,” in *Intelligent Design and Manufacturing*, editor A. Kusiak, New York: Wiley.

Lu, S.H. and Kumar, P.R. (1991), “Distributed scheduling based on due dates and buffer priorities,” *IEEE Trans. Automat. Control*, Vol. 36, No. 12, pp. 1406-1416 (Dec.).

Luh, P.B. and Hoitomt, D.J. (1993), “Scheduling of manufacturing systems using the Lagrangian relaxation technique,” *IEEE Trans. Automat. Control*, Vol. 38, No. 7 (July).

Murata, T. (1989), “Petri nets: properties, analysis and applications,” *Proc. IEEE*, Vol. 77, No. 4, pp. 541-580 (April).

Panwalker, S.S. and Iskander, W., “A survey of scheduling rules,” *Operations Research*, Vol. 26, No. 1, pp. 45-61 (1977).

Peterson, J.L. (1981), *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ.

Ramadge, P.J. and Wonham, W.M. (1989), “The control of discrete event systems,” *Proc. IEEE*, Vol. 77, pp. 81-98.

Steward, D.V. (1962), “On an approach to techniques for the analysis of the structure of large systems of equations,” *SIAM Review*, Vol. 4, No. 4, pp. 321-342 (October).

Tompkins, J.A. and White, J.A. (1984), *Facilities Planning*, John Wiley & Sons.

Warfield, J.N., “Binary matrices in system modeling,” *IEEE Trans. Systems, Man, Cybern.*, Vol. SMC-3, No. 5, pp. 441-449 (1973).

Wolter, J., Chakrabarty, S., and Tsao, J. (1992), “Methods of knowledge representation for assembly planning,” *Proc. NSF Design and Manuf. Sys. Conf.*, pp. 463-468 (Jan.).