11

Design of FMS: Logical architecture[1]

Alejandro Malo & Antonio Ramírez
Dpto. de Ingeniería Eléctrica - Sección de Control Automático
CINVESTAV-IPN
Av. IPN 2508, México D.F.

Abstract

The work in this article is within the framework of the design of flexible manufacturing systems. Its aim is to represent the different ways the processes and the assignment of the resources can be done; preserving the flexibility found in this kind of systems and allowing the calculus of resources. The different manufacturing sequences found in the system are represented by state machines, while the manufacturing resources are shared. All this information is used, in this report, to define a Petri net class, that in a single net permits the specification, in a single net, jobs that can be done in different ways, that is, the redundancy that exist within a manufacturing system, not found in other representations.

Keywords

FMS Design, Petri Nets

1 INTRODUCTION

The design of flexible manufacturing systems is usually decomposed in several steps, e.g.; specification of the requirements, definition of the dynamic model, organization of the information, selection of the elements of the system, plant layout and validation. In this article we are concerned with the definition of a dynamic model specified by the following data:

- description of the different jobs as operation sequences.
- desired throughput (number of jobs per unit of time).

Within the design of FMS we found different views [Kouvelis 92, ProHil 90, SilVal 89]. In this article we use the Petri Net approach because it is a formal tool that can represent the structure and dynamics of a flexible manufacturing system [Silva 85], and with the addition of time, timed Petri nets, can be used to calculate the the resources needed to fulfill a desired throughput [ProHil 90, HilPro 89, LafProXie 92].

The article is organized as follows. Section 2 briefly presents structural aspects of Petri nets, for more information an interested reader can consult [LafProXie 94, Murata 89, Silva 85, RamHo 80]). Section 2.1.1 presents the modeling of the manufacturing sequences and the machine job assigment. Section 3 presents an aproximation to the modeling of processes and control circuits when we have a redundant system. The load balancing problem and the selection the minimum number of resources that fullfil with the desired throughput is solved. The last section presents an example.

[1]This paper presents work done for the ECLA/FLEXSYS project

L. Camarinha-Matos et al. (eds.), *Balanced Automation Systems*

2 PETRI NETS

In this section the basic concepts related with Petri nets are presented.

An ordinary Petri net is a directed bipartite graph represented by the fourth-tuple $N = (P, T, Pre, Post)$ where: $P = \{p_1, p_2, ..., p_n\}$ is a set of elements called places; $T = \{t_1, t_2, ..., t_m\}$ is a set of elements called transitions; $P \cap T = \emptyset$, $P \cup T \neq \emptyset$; Pre (Post) is the pre (post) incidence function that represents all the input (output) arcs to (from) a transition, $Pre, Post : P \times T \rightarrow 1, 0$. The places are drawn as circles and the transitions as bars. The pre-post incidence functions can be represented by matrices $PRE = [a_{ij}]$ and $POST = [b_{ij}]$ respectively, where $a_{ij} = Pre(p_i, t_j)$ and $b_{ij} = Post(p_i, t_j)$.

The incidence matrix $C = [c_{ij}]$, $i = 1, .., n$ $j = 1, ..., m$, is defined as $c_{ij} = b_{ij} - a_{ij}$. All the vectors X such that $C \cdot X = 0, X \geq 0$ are called T-semiflows; all the vectors Y such that $Y^T \cdot C = 0$, $Y \geq 0$ are called P-semiflows.

2.1 Time in Petri Nets

A Timed Petri Net $TPN =< N, D >$, where $D : T \rightarrow I\!R$. $D(t_i) = d_i$ is called the delay of the transition t_i and is the time needed to accomplish the firing of the transition. We prefer to assign the time to the transitions rather than to the places [LafProXie 94], since for us transitions represent the activities of the system.

2.1.1 Some Results in Strongly Connected State Machines

A state machine is a PN or a TPN where $|^\bullet t| = |t^\bullet| = 1, \forall t \in T$. This kind of PN allows the representation of conflicts, so in order to specify the optimum cycle time π (the inverse of the throughput) we must specify the visit ratio vector v_k. The following equation computes the optimal cycle time:

$$\pi \; 1^T \cdot M_0 = 1^T \cdot Pre \cdot D \cdot v_k$$

where $D = Diag[d_i]$, and v_k is the visit ratio vector.

2.1.2 Computing the visit ratio vector

The information in this section was taken from [CamChiSil 91]. In a state machine, those transitions that have the same predecessor are in conflict since the firing of one of them disables the others. Conflicts can be solved assigning each transition a routing ratio, i.e., the ratio each one is fired. Suppose t_i has the routing ratio r_i and t_k the routing ratio r_k. then for σ a periodic sequence, and $\phi(\sigma, x)$ the number of times that the transition x appears in the sequence σ, we have

$$\lim_{length(\sigma) \rightarrow \infty} \frac{\phi(\sigma, t_i)}{\phi(\sigma, t_k)} = \frac{r_i}{r_k}$$

Petri nets that we are using are bounded and live, so σ is generated by flows that are executed several times. It means that $\phi(\sigma)$ is a vector that can be decomposed as a linear combination of T-semiflows.

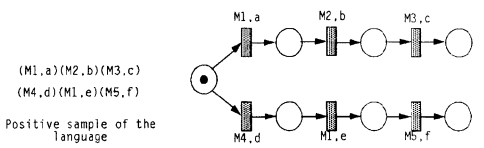

Figure 1: The job (as a positive sample and state machine).

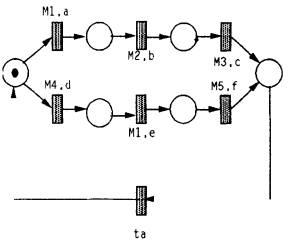

Figure 2: The job (as a strongly connected state machine).

$\phi(\sigma)$ is the Parikh vector of the σ sequence, so its elements are the elements of visit ratio vector. Then the previous equation can be rewritten as:

$$v(t_i) \cdot r_k - v(t_k) \cdot r_i = 0$$

Also, we have that the visit ratio vector is a combination of T-semiflows of the Petri net, we have

$$\begin{pmatrix} C \\ R \end{pmatrix} \cdot v = 0 \tag{1}$$

The vector is usually normalized for one transition, that is $v_k(t_j) = v_j/v_k$. After this brief introduction to Petri Nets, lets present how we use them to design FMS.

3 MODELING FLEXIBLE JOBS

Marked graphs have been used to model FMS[HilPro 89, ProHil 90, LafProXie 92], however, decisions exist in an FMS, so marked graphs are not a realistic approach. Flexible jobs have different production paths and these paths must be executed in some proportion in order to fulfill load balance constraints. We need a tool that captures this characteristic. In this paper we propose the use of a subclass of Petri nets that captures synchronization and decisions in its structure.

The job model is the result of the union of two nets: a processing net and a command net, both modeled as state machines. When we fuse both circuits, however, the resulting Petri net is no longer a state machine.

3.1 Processing Nets

A job processing net is described by a positive language sample [Angluin 87], each sample is an operation sequence (and a T-semiflow of the net), these operations are performed by machines; it is possible that one machine performs more than one operation in any sequence of the sample. A finite automata can be built from the positive language sample; this paper considers the case where the automata can be described as a state machine Petri net. In the left of figure 1 we have the operation sequence of a product represented by a positive sample. The sample says that the product can be done performing operation a on machine

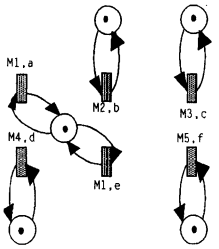

Figure 3: The command net.

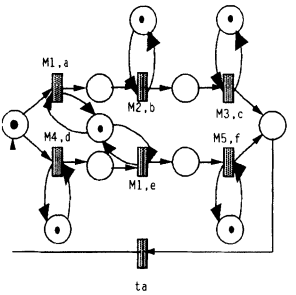

Figure 4: The global model of a job.

one, then operation b on machine 2 and finally operation c on machine 3; or by operation d on machine 4 followed by operation e on machine 1 and by operation f on machine 5. On the right side of figure 1 a Petri net model of these operation secuences is presented. The reader can observe that this Petri net model is a state machine.

From figure 1 we get, fusing all final states and adding the feedback transition t_a, figure 2 a strongly connected state machine that generates the same positive sample, so it also specifies the same job. We use the later model to solve the load balance and the throughput problems. The subclass N, of Petri nets, that models the process can be described as follows:

- $N = (P, T, I, O)$ is a strongly connected state machine
- $\exists t_a \in T \mid \|{}^\bullet t_a\| = \|t_a^\bullet\| = 1$
- $\exists p_i \in P \mid {}^\bullet p_i = \{t_a\}$
- $\exists p_f \in P \mid p_f^\bullet = \{t_a\}$
- $\forall |X_i|$ such that $C \cdot X_i = 0,\ t_a \in \|X_i\|$

Since the process net model is a strongly connected state machine, it has only one P-semiflow and the throughput and minimum initial marking can be computed in polynomial time. All P-components include p_i (initial state or raw material) and p_f (final state or product).

3.2 Command nets

The job command net is represented as a positive sample of operations. Each element of a sequence is an ordered pair (figure 3), where the first item represents the machine and the second item represents the operation. A command net will also be represented by a state machine Petri net, however, this state machine is not strongly connected (Figure 3). Since the command and processing nets perform the same activities, the set of transitions T will the same for both.

The set of places changes in the command net case. Each type of machine is represented by one place. These places are connected to transitions that represent the activities performed by the machine (represented by the place). Figure 3 shows a Petri net model of a command circuit for the positive sample (job) presented in figure 1.

3.3 Joining processing and command circuits

A global model (that includes both processing and command nets) for a job is built using both models (processing and command models). It is an easy task because both models have the same transitions. The processing circuit model is the net $N_p = (P_p, T, I_p, O_p)$ and the command circuit model is the net $N_c = (P_c, T, I_c, O_c)$, then the global model is the net $N = (P_p \cup P_c, T, I_p \cup I_c, O_p \cup O_c)$. Figure 4 presents the global model for the job presented in figure 1.

3.4 A model for all jobs

In a system we usually have several concurrent jobs. For job j_k, we have the model $N_k = (P_k, T_k, I_k, O_k)$. Each job has its own model, but they can share machines, so the places of command circuits can be shared by different jobs. Formally it means that if $N_k = (P_k, T_k, I_k, O_k)$ is a net modeling job k and $N_j = (P_j, T_j, I_j, O_j)$ is a net modeling job j, then $P_k \cap P_j \neq \emptyset$; $T_k \cap T_j = \emptyset$; $I_k \cap I_j = \emptyset$; $O_k \cap O_j = \emptyset$. Places representing machines, that are shared by different jobs, are merged into a single place, and the models can be merged into one. Figure 5 shows how a family of jobs can be modeled.

We remark that the final model is flexible. Since it allows us to model a family of jobs processed by shared machines, and, at the same time, jobs can be realized in different ways.

3.5 Computing the process net minimum initial marking

In this report we suppose the load balance problem solved, so we know, in the long term, the times each machine must be visited in relation to the remaining machines, i.e., we know the routing ratio numbers r_i, from which the visit ratio vector can be computed.

Unfortunatelly, the resulting Petri Net model of a job is not easy to analyse. However, if we suppose that the resources are infinite, then we can consider that a job model is a state machine and the result on section 2.1.1 can be used, since the resources do not introduce any constraints.

We have as input data the job specified as a State Machine; we also have the throughput and the proportion in which the different production paths must be executed.

Our problem consists in finding the minimum number of tokens in the Petri net that allows to reach the desired throughput. From [Campos 90] we know that Little's law can be expressed as:

$$Y_{job}^T \cdot M_{0_{job}} \;/\; Throughput_{job} \geq Y_{job}^T \cdot Pre \cdot Diag[d] \cdot v_a \qquad (2)$$

then we can compute the minimum initial marking, i.e., the work in process, from:

$$Y_{job}^T \cdot M_{0_{job}} = \lceil Y^T Pre \cdot Diag[d] v_a \cdot Throughput_{job} \rceil$$

3.6 Computing the minimum number of resources

In the previous section we considered the resources infinite. The previous computation gives us the work in process, for the given throughput. Now it is necessary to compute the minimum number of resources.

Unfortunately, when we introduce the resource constraints, the job model, figure 4, is no longer a state machine. However, if we only see the work of a machine, the command subnet model is strongly connected state machine, so we can use the same procedure to compute the minimum marking, this can be done considering one or several jobs.

The throughput for a machine p_c is guaranteed, without scheduling problems, if the sum of the resources needed for individual job is equal to the number of resources needed for doing simultaneously all the jobs, that is,

$$\sum_{job}\lceil Y^T(p_c)\cdot Pre\cdot Diag[d]\cdot Diag[T_{job}]\ v_a\cdot Throughput_{job}\rceil$$

$$-\lceil Y^T(p_c)\cdot Pre\cdot Diag[d]\cdot\sum_{job}v_{job}\cdot Throughput_{job}\rceil=0$$

where $v_k=\sum_{job}v_{job}$. A feasible schedule that satisfies the throughput constraints can be found if the available time is greater than the utilization time, that is, if

$$(Y^T(p_c)\cdot M_0\cdot\sum_{job}Throughput_{job})^{-1}\geq Y^T(p_c)\cdot Pre\cdot Diag[d]v_k$$

otherwise there will be. With this considerations, the initial marking can be calculated.

4 EXAMPLE

Suppose that we need to design an FMS to manufacture the products given by the following positive sample.

Job 1: $(M_1,a)(M_2,b)(M_3,c)$ and $(M_4,d)(M_1,e)(M_5,f)$

Job 2: $(M_6,g)(M_7,h)(M_8,i)$ and $(M_4,j)(M_2,k)(M_9,l)$

The throughput must be [1/4 1/2], i.e., one product of type 1 must be manufactured every 4 time units and one product of type 2 must be manufactured every 2 time units. The operation times for the first job are:

$(M_1,a)=1$; $(M_2,b)=2$; $(M_3,c)=1$; $(M_4,d)=1$; $(M_1,e)=1$; $(M_5,f)=2$

and for the second job:

$(M_6,g)=1$; $(M_7,h)=1$; $(M_8,i)=1$; $(M_4,j)=1$; $(M_2,k)=1$; $(M_9,l)=1$

In this case each job can be performed in two different ways. In both cases we will suppose that the load is equally balanced, i.e., all sequences must be performed once. The model of the first job is shown in figure 4, in this case, the visit ratio vector is given by v_{a_1} = [1 1/2 1/2 1/2 1/2 1/2 1/2] (where $v_{a_1}(1)=1$ and corresponds to t_{a_1}; $v_a(2)$ corresponds to $t_{(M_1,a)}$; $v_{a_1}(3)$ corresponds to $t_{(M_2,b)}$ and so on).

The minimum initial marking for the processing net for the first job is computed from:

$$Y^T_{job_1}\cdot M_0=\left\lceil Y^T Pre\cdot Diag[d]v_k\cdot Throughput\right\rceil$$

so the work in process needed to satisfy the throughput is $Y^T_{job_1}\cdot M_0=1$.

When we consider the command nets, the minimum number of manufacturing resources can be computed. In the case of the first job, one resource of each type is enough to fulfill throughput constraints.

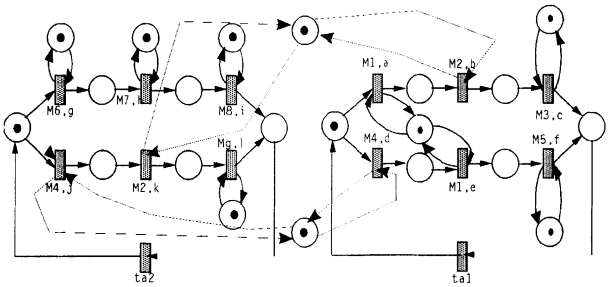

Figure 5: Model of a system performing two jobs.

The requirementes for the second A job can be calculated in a similar fashion. In this case we need two tokens to satify the throughput, that is, $Y_{job_2} \cdot M_0 = 2$. And one token for each kind of machine. So the initial marking is:

$$M_0 = [1\,0\,0\,0\,0\,0\,2\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1]^T$$

Now, both jobs are merged to get the final model. In this case, places representing same machines are merged. Figure 5 represents the final model.

When we consider both jobs we obtain a similar marking. However, due to the coupling between jobs, we must be aware that this minimum marking might not be enough to satisfy the throuhgput, and it will depend of the desired scheduling.

5 CONCLUSIONS

This paper presents a methodology to model FMS. The FMS is modeled as processes to be done and as machines assigned to these processes. Since, in many cases a product can be done in several ways, we are using state machines to represent them. We also use state machines to represent the work done by each kind of manufacturing resource. The flexibility is retained by the representation. And allow us to calculate the minimum marking of the net, that is, the resources needed to fulfil the throughput.

6 REFERENCES

[Angluin 87] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, (75):87–106, 1987.

[CamChiSil 91] J. Campos, G. Chiola, and M. Silva. Properties and performance bounds for closed free choice synchronized monoclass queueing networks. *IEEE Transactions on Automatic Control*, 36(12):1368–1382, December 1991.

[Campos 90] J. Campos. *Performance Bounds for Synchronized Queueing Networks.* PhD thesis, Universidad de Zaragoza., María de Luna 3 E-50015 Zaragoza, España., 1990.

[HilPro 89] H. Hillion and J. Proth. Performance evaluation of job-shop systems using timed event-graphs. *IEEE Transactions on Automatic Control*, 34(1):3–9, January 1989.

[Kouvelis 92] P. Kouvelis. Design and planning problems in flexible manufacturing systems: a critical review. *Intelligent Manufacturing*, (3):75–99, 1992.

[LafProXie 92] S. Laftit, J. Proth, and X. Xie. Optimization of invariant criteria for event graphs. *IEEE Transactions on Automatic Control*, 37(5):547–555, May 1992.

[LafProXie 94] S. Laftit, J.M. Proth, and X.L. Xie. Petri nets for modeling of dynamic systems - a survey. *Intelligent Manufacturing*, 30(2):175–202, 1994.

[Murata 89] T. Murata. Petri nets: properties, analysis and applications. In *Proceedings of the IEEE*, pages 541–580, April 1989.

[ProHil 90] J. Proth and H. Hillion. *Mathematical Tools in Production Management.* Plenum Press New York, 233 Spring Street, New York, NY 10013 USA, 1990.

[RamHo 80] C. Ramamoorthy and G. Ho. Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Transactions on Software Engineering*, 6(5):440–449, September 1980.

[Silva 85] M. Silva. *Las Redes de Petri: en la Automática y la Informática.* AC, Madrid, España, 1985.

[SilVal 89] M. Silva and R. Valette. Petri nets and flexible manufacturing. In Rozenberg; editor, *Advances in Petri Nets 1989*, pages 375–417, Springer-Verlag, Berlin, Germany, 1989.

7 BIOGRAPHY

Alejandro Malo Born at Guadalajara, México in 1953. An Aeronautical engineer from the Esime-IPN México, specialized at l'ENSAE, Toulouse, France; with a Master Degree from the Cinvestav-IPN, México. Currently pursues Ph.D. studies in Electrical Engineering at the Cinvestav-IPN. His current research interests are the Design of Flexible manufacturing, Dynamic Discrete Event Systems.

Antonio Ramírez was born in Monterrey, Nuevo León, México in 1964. He obtained his B.S. degree in Electrical Engineering from the Metropolitan University, México; his M.S. degree from the Cinvestav, México; and his Ph.D. from the University of Zaragoza, Spain. He is presently with the Electrical department of Cinvestav, México. His current research interest are: discrete event systems, Petri net theory, real-time control and control applications