Petri-Nets for Robot Lattices*

Z.-N. Cai*, A. Farnham*, A. Z Ghalwash*,
P. Gómez^, V. Rodellar^ and R.Newcomb*^

*Microsystems Laboratory
Electrical Engineering Department
University of Maryland
College Park, Maryland 20742
Phone: (301) 454-6869

^Grupo de Sistemas (PARCOR)
Facultad de Informática
Universidad Politécnica de Madrid
Ctra. de Valencia, Km. 7,00
28031 Madrid, Spain

Abstract:

An augmented lattice is introduced for the analysis of production lines of robots with the analysis set up using binary Petri-nets.

It is known that the lattice production line configuration of robots is about the most flexible for industrial manufacturing. This lattice configuration is slightly extended to further increase the flexibility. This augmented lattice is investigated using a single-place, multiple transition model for the robot information flow and the connection decision making. Using this model and binary Petri-net theory, the equations describing the system are obtained for a cascade of n augmented lattices. From the equations, a discussion is given on prevention of net deadlock, system tolerance, and production efficiency.

I. Introduction

Among the robot production lines of most interest in today's high technology world, the lattice lines, indicated in Fig. 1, show high reliability [1]; and, hence, they are of prime interest for analysis. Among the available tools for analysis of such structures, Petri-net theory appears the most promising. This is because it has been generated for concurrency [2], and production lines are highly concurrent. Our modelling of the Petri-net, via the binary Petri-net, offers a simple relationship among the input, marking, and firing functions. For real time systems, the binary Petri-net is very programmable.

In Fig. 1 the nodes represent robots and the branches represent communication paths between the robots. In this figure we have augmented the standard lattice by the vertical paths so that each robot can communicate to all of its adjacent neighbors. In Fig. 1 we will consider the a-axis as the main line, with the b-line as a backup one. The item being assembled or repaired is worked on in turn, from left to right, on the a-axis manufacturing line. If a robot on the a-line fails, an adjacent backup robot on the b-axis will take over the tasks of the failed one. This function is under the control of a central CPU controller. To simplify Fig. 1, the controller signals are left out, but its bidirectional bus is shown in Fig. 2, as discussed below. Figure 2 is divided into two figures, 2a) and 2b), that make up a vertical pair of robots in the manufacturing line. The connections shown in Fig. 1 are bidirectional signal-flow lines and allow for information to be routed around any failed robot. When this happens, an external decision, via the controller, will be made that replaces the failed robot with one of the three backup line robots connected to it. This quality allows the system to run continuously, avoiding deadlock. However, in the interest of cost effectiveness, one would not like to have the b-line inactive when not being used for backup. Consequently, in the presence of the b-axis line's other uses, deadlock could occur. This deadlock can be determined using computer-aided analysis and Petri-net models of production lines.

System problems such as deadlock can be handled by data base theory. When the central CPU reads the system's states, all this information is placed into a data base. This allows us to prevent deadlock by using software to monitor the data base. When there are conflicting actions in the lattice, the central CPU will be able to recognize whether the actions are logical contradictions or deadlock. However, binary Petri-net theory guarantees no logic conflict during system execution. We can therefore free the CPU to deal only with the deadlock problem. We can use the analogy that the central CPU controlled robot lattice is like a distributed operating system structure. Therefore, when the central CPU recognizes a deadlock condition, distributed operating system's algorithms will be used to prevent or to solve the deadlock problem. [4], [5].

Supported in part by the US-Spain Joint Committee for Scientific and Technological Cooperation under Grant No. CCB-84-002-002.

In section II, we introduce the Petri-net model, Fig. 2, to be used for a robot in the line. In section III, the describing equations for a lattice line are set up within the framework of binary Petri-net theory [3]. Then in section IV, the full robot lattice is examined.

II. Robot Petri-Net Model

Figure 2 gives the Petri-net model we will use to represent any robot in the production line. Figure 2 shows only the connections for the transitions leaving the robot; and, for space reasons, no connections are shown for the transitions entering the robot. In Fig.s 2a) and 2b), each robot place has five inputs coming from the five adjacent robots; a token in the robot place represents that the robot has finished its job and can send its part to any one of the five other robots to which it is connected. To which robot it actually sends is under the control of the central CPU; hence, the central CPU controls the firing of the output transitions as shown by input places connected to the transitions. To keep the resolution of the CPU consistent with the resolution of our Petri-net model of the robot lattice, we would model the CPU as a binary Petri-net. However, in this paper, we assume this model for the CPU and then we develop the robot lattice.

In this representation, it is understood that the central CPU enables only one of the five output transitions of any given robot at any given time. This enabling is accomplished by placing a token in only one of the places connecting to these transitions. That is, there is 1 safety. However, it naturally might enable one transition of every robot on the line at a given instant. For example, this would be necessary if the robots were working on parts that were sitting on a conveyer belt. It should be noted that in this model we are assuming that the central CPU signals are external inputs to the Petri-net. Such an assumption also means that we do not need to incorporate the communication path for interrupt signals and status signals going from the robot to the central CPU, though it is implied there are such signals in order for the central CPU to make the necessary decisions.

III. Review of Binary Petri-nets

The token that is transferred from place to place in a Petri-net represents the system implementation. That is, the system state changes when an event occurs, or when the conditions in the system are varied. Transitions and places may have many inputs and outputs. At firing time λ, a token is moved through a fired transition. It moves from a place connected to the transition to a place on which the transition is incident. Simultaneously, the system marking changes to that of the next discrete time, $\lambda+1$.

If p is the number of places, the p-vector containing the number of markings (tokens) at time λ is taken to be $M(\lambda)$. The vector containing the firing of transitions is the t-vector $F(\lambda)$. An entry of 1 denotes a firing at time λ, while a 0 represents no firing at that time. The vector containing the external inputs, all of which are assumed to go to places, is the p-vector $I(\lambda)$. All these vectors contain nonnegative integers, which are 1 and 0 according to the fact that we limit ourselves to binary Petri-nets (since a part or signal is either present or absent).

Following the established binary Petri-net arithmetic, we are now ready to introduce some needed operations and notation [3]. The incidence matrix is $C=C^{+}-C^{-}$. The entry $C_{i,j}^{+}$ of C^{+} is 1 or 0, depending on whether transition i is incident upon place j. The entry $C_{i,j}^{-}$ of C^{-} is 1 or 0, depending on whether place j is incident upon transition i. The dot equality "$\doteq$" denotes that the operations on the right side of an equation be carried out using normal integer arithmetic with, in the final results, positive numbers on the right side being replaced by a 1, and every other number replaced by a 0. The symbol $\square$ is defined for a row p-vector "a" and a column p-vector "b", both of which have 0 and 1 entries according to the following equality.

$$a\square b = \begin{cases} 1 & \text{if at least one pair } a_i b_i \text{ has } a_i=1,\ b_i=1, \text{ and no pair has } a_i=1,\ b_i=0 \text{ for } i=1,2,\ldots,p \\ 0 & \text{otherwise} \end{cases}$$

In order to take the input into account, we define the input independent intermediate variable $M_o(\lambda+1)\doteq M(\lambda)+C^{T}F(\lambda+1)$, where superscript T denotes matrix transposition. To find the firing vector, we note that the ith transition fires when all the places incident to the transition are filled with tokens. The firing of the ith transition is denoted by the ith entry of the firing vector being set to unity, $f_i=1$, while no firing is denoted by $f_i=0$. As above, $F(\lambda)$ can be projected onto the output Y via an output matrix D. The resulting equations of binary Petri-net theory are then:

$$M(\lambda)\doteq M_o(\lambda)+I(\lambda) \qquad (1a)$$
$$F(\lambda+1)=C^{-}\square M(\lambda) \qquad (1b)$$
$$M_o(\lambda+1)\doteq M(\lambda)+C^{T}F(\lambda+1) \qquad (1c)$$
$$Y(\lambda)=DF(\lambda) \qquad (1d)$$

The equations (1a) and (1d) represent the system's current states, including current system input and output. Equations (1b) and (1c) describe how a system, without input, is marked for the next discrete time $\lambda+1$. These two equations also describe system preparation for firing at time $\lambda+1$. According to (1), the system is

time invariant because the incidence matrix and output projection matrix are constant.

The firing function and the system marking function are related to the external input function $I(\lambda)$. Because $I(\lambda)$ is determined by the central CPU, a relationship exists between the system's state consistency and system implementation. That is, $I(\lambda)$ can adjust system states by $M_0(\lambda)=M(\lambda)+I(\lambda)$; and $I(\lambda)$ can, as a consequence, affect the system implementation by $F(\lambda+1)=C^{-}\square M(\lambda)$. This defines when the system should execute the next task, while allowing the entire system consistency to be maintained. The system described by (1) is also determined by current marking and input.

IV. The Lattice

a. Characteristics of our lattice

A lattice of robots having four major characteristics can be represented by using binary Petri-net theory. The first characteristic is that there is consistency throughout the entire lattice. No robot repeats the task already completed by another robot; and for the item being worked on, no task is left incomplete. This is accomplished by allowing only one central CPU to direct all firing functions. Also, further consistency is maintained by setting the communication protocols in the central CPU and each of the related local computers and robot control computers.

The last two characteristics define the conditions in a place. A place is in working condition if an item is present in that place. This lattice condition, or state, is sent to the system central CPU. Each place K_n of Fig. 2 is further replaced by two places K_n' and K_n'' in Fig. 3. This model of Fig.s 2 and 3 allows us to assign states to a place: States being the data transmitted from the robot to the central CPU. A place in working condition can be in one of two states: The busy state or the task completed state. In the busy state, a token is in the K_n' place of Fig. 3. In the task completed state, the transition t_n has been fired and the token is transferred to K_n'' of Fig. 3. The place K_n is ready to accept next token when in the task completed state. If necessary, the central CPU input could also be inserted in Fig. 3 to allow it to repeat, or backup, the transmission of the token. The communication protocols of the robot computer control system should be designed to make this action stable and reliable.

When information is to be transferred to all the transitions connected to a place, these transitions are said to be ready to be fired, or enabled. The firing actually takes place when the enable firing signals are transferred from the central CPU. When these transitions accept firing signals from both the places and the central CPU, the event occurs. When the event occurs, the token is actually transferred from one place to another. We say that transitions are controlled by both inside and outside environments. We say inside environments because transitions relate conditions to places; and we say outside environments because the central CPU relates the system's states and a topview of the entire lattice. These inside and outside environments are conditions themselves. When all environments satisfy the token transfer conditions, the firing function fully enables the transitions.

Put another way, the central CPU corrects the nondeterministic nature of an otherwise randomly operating robot line; and binary Petri-net theory does the same to the Petri-net representation by putting the CPU signals into the input vector. The central computer makes a firing decision based on the system's current states and any external inputs. By centrally controlling all firing decisions, the system consistency is maintained and information flow deadlock is prevented.

b. Constructing the lattice

A single robot connected at the kth position in the a-path of the lattice in Fig. 1 is represented by the upper middle portion of the lattice shown in Fig. 2a). Again, the a-axis represents the main line, while the b-axis represents the backup line. We assume that there are n robots in the lattice. To distinguish between robots, they each have their own unique index K, where $1\leq K\leq n$. To simplify Fig.s 2a) and 2b), we show only the out-transitions for the Kth robot vertical pair. That is, we show only the connections for the transitions leaving the robot, and no connections for the transitions entering the robot. K_a and K_b each represent a robot in the vertical pair shown in Fig.s 2a) and 2b), respectively. $K_{a,i}$ and $K_{b,i}$, where $1\leq i\leq 5$, represent the information transferred from the central CPU to the transitions on the main line and backup line, respectively. The transitions for the a-axis are numbered counter clockwise, while the transitions for the b-axis are numbered clockwise. $t_{Ka,i}$ and $t_{Kb,i}$, with $1\leq i\leq 5$, represent the transitions on the main line and the backup line, respectively.

In Fig. 2b), the b-axis connections are shown from the transitions to their destination places. The transition is controlled by both the central CPU and the place at which the transition originates. When both the central CPU and the original place enable the transition, the transition fires to its destination place.

c. The matrices describing the lattice

The robots K_a and K_b can be represented as a block within the incidence matrix C. The robots K_a and K_b are described by the matrix, C_K, given in Fig. 4a), $C_K=C_K^{+}-C_K^{-}$. A similar matrix exists

for each vertical pair of robots in the lattice, except the first and last pair of robots. In Fig. 4a), the first twelve columns and last twelve columns are a result from the K-1 and the K+1 robots, respectively. With this in mind, we can see that the first robot of the lattice, K=1, does not have a K-1 robot. Similarly, the last robot of the lattice, K=n, does not have a K+1 robot. As a result of this physical condition, the matrix for the first vertical pair of robots is similar to that in Fig. 4a), but with a few important changes. First, we must zero fill the first twelve columns of the matrix in Fig. 4a). Similarly, we zero fill the last twelve columns of the matrix in Fig. 4a) for the last vertical pair of robots. In addition to the subtraction of columns, we must zero fill the $K_{a(b),1\&2}$ and the $K_{a(b),4\&5}$ columns for the first and last robot, respectively. For C_1, we also zero fill the $t_{Ka,1\&2}$ and $t_{Kb,1\&2}$ elements of the K_a and K_b columns, respectively. Similarly, for the C_n matrix of the last robot, we zero fill the $t_{Ka,4\&5}$ and $t_{Kb,4\&5}$ elements of the K_a and K_b columns, respectively. This is done because when using the first and last robots as the extremes of the lattice, no transitions exist on the outside of the lattice. Excluding the matrices for the first and last vertical pair of robots, the matrices for every other vertical pair of robots are identical to the one in Fig. 4a). Together, these blocks make up the incidence matrix C. The blocks of Fig. 4a) hold the positions in C as shown in Fig. 4b).

To form the entire C matrix, we start with C_1, the first submatrix, and place it in the uppermost left hand corner of the C matrix that we are constructing. From the uppermost left hand position, we shift to the right twelve positions and shift down ten positions. From here, we place a duplicate of the C_K matrix pictured in Fig. 4a). From the uppermost lefthand corner of this C_K matrix, we shift down ten and to the right twelve positions. From this new starting point, we position the matrix of Fig. 4a) again. We continue (using the shift down ten, shift to the right twelve procedure) this positioning of matrices until we encounter the last robot. When we add the matrix C_n for the last robot, we do not include the last twelve columns of the matrix in Fig. 4a); also, we zero fill the forementioned columns, because no K=n+1 robot exists for the last robot. Figure 3b) shows the C matrix as it should appear after the positioning of all the C_K's, where $1 \le K \le n$ and n is the number of robots. These blocks of Fig. 4a) are slightly off the diagonal of C, as indicated by the forementioned shifting procedure; zeroes fill all the positions not held by the C_K's. To recover the C^+ and C^- matrices from C, we simply extract the nonnegative terms or nonpositive terms, respectively.

We form the marking function matrix M_o in much the same way as the incidence matrix C, but we use a vector block to describe the Kth robot. The vector for the Kth robot is given by (here ^{T} denotes transpose)

$$M_{oK}(\lambda)^T = [1,0,0,0,0,0,1,0,0,0,0,0,1,$$
$$0,0,0,0,0,1,0,0,0,0,0,$$
$$1,0,0,0,0,0,1,0,0,0,0,0]$$

In this vector, the numbering of the rows is identical to the numbering of the columns for the matrix C.

Again, the vectors describing the first and last robots differ from this vector by the first twelve and last twelve entries, respectively, the entries of which are zero filled. We also must remember that for the first and last robot, no transitions exist towards the outside of the lattice; consequently, we must zero fill the $K_{a(b),1\&2}$ and the $K_{a(b),4\&5}$ entries for the first and last robot, respectively. To form the entire marking matrix M_o, we start at the top, and duplicate the above vector, zero filling the first twelve and the forementioned entries that do not exist for this vector. Then we add blocks to the column vector by going from K to K+1 via a shift down of twelve positions. We continue this process, as before, until we reach the last robot. At this point, we eliminate the final twelve entries of the above specified vector, while zero filling the missing transitions; we use this as the vector for the last robot. To determine the input vector I, we repeat this process, giving us for the kth block

$$I_K(\lambda)^T = [0,0,0,0,0,0,0,0,0,0,0,0,0,I_{Ka,1},$$
$$I_{Ka,2}, I_{Ka,3}, I_{Ka,4}, I_{Ka,5}, 0,$$
$$I_{Kb,1}, I_{Kb,2}, I_{Kb,3}, I_{Kb,4}, I_{Kb,5},$$
$$0,0,0,0,0,0,0,0,0,0,0,0]$$

V. Discussion

We have shown how a lattice of robots can be represented by a binary Petri-net. In the augmented lattice treated here, each robot is connected to its five nearest neighbors; this connection gives considerable flexibility for use in the backup of failed robots. The resulting Petri-net then has each robot place connected to five other robot places through controlling transitions. In turn, the firing of each of the transitions is controlled by a central CPU input; this necessitates another place for each transition in order that the describing equations, (1) above, will hold. As far as the robot line itself is concerned, these CPU signals are external controlling signals; and such an interpretation fits nicely into the binary Petri-net framework to allow us to use the equations (1). And because we have a nice interconnection pattern of robots, the incidence matrix has attractive block patterns with individual

blocks being identical, except at the ends, as illustrated in Fig. 4.

Besides the connections of a robot to its five nearest neighbors, there is also connection to the CPU. We have shown the input controlling Petri-net connections from the CPU to the places, but we have left off the connections of the robot outputs to the CPU; binary Petri-net description of the full set of places and transitions is the next step in continuing this study.

References

[1]. H. Alayan, C. A. Niznik, and R. W. Newcomb, "Reliability of Basic Robot Automated Manufacturing Networks," Proceedings of IEEE Southeastcon'84, Louisville, April 1984, pp. 291-294.

[2]. R. Johnsonbaugh and T. Murata, "Petri Nets and Marked Graphs - Mathematical Models of Concurrent Computation," American Mathematical Monthly, Vol. 89, No. 8, October 1982, pp. 552-566.

[3]. H. Alayan and R. W. Newcomb, "Binary Petri-Net Relationships," manuscript prepared for publication.

[4]. J. Gray, Distributed Operating System, IBM Research Laboratory, Springer-Verlag, N.Y., 1977

[5]. S. Ceri and G. Pelagatti, Distributed Databases - Principles and Systems, McGraw Hill Computer Science Series, N.Y., 1984

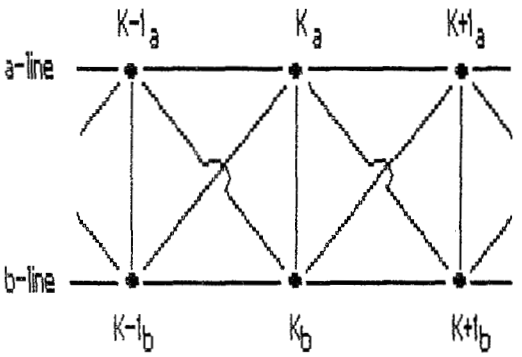

Figure 1. Lattice Robot Line
(a = primary, b = secondary/backup)

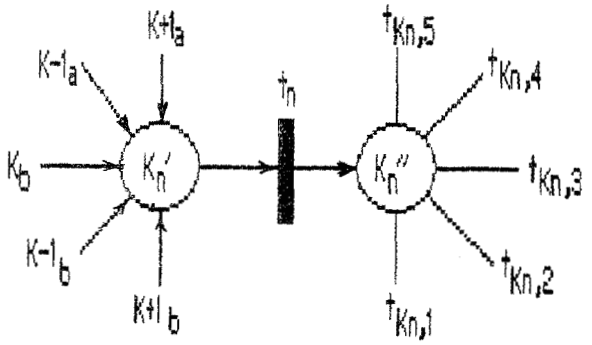

Figure 3. The Single Place Model of K_n

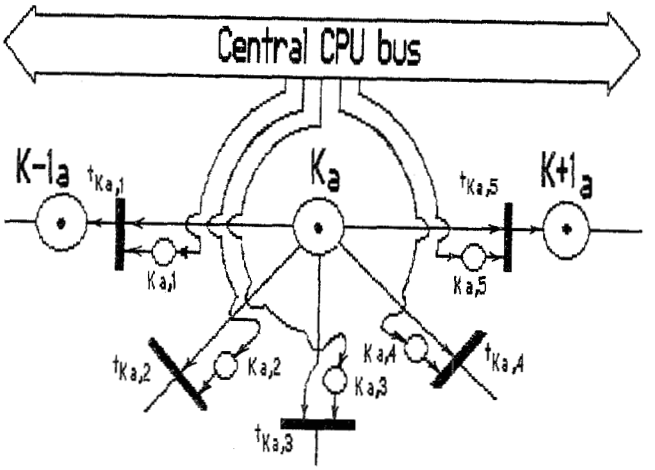

Figure 2a) Places and Out-transitions for the Kth Robot Pair of the Main Line

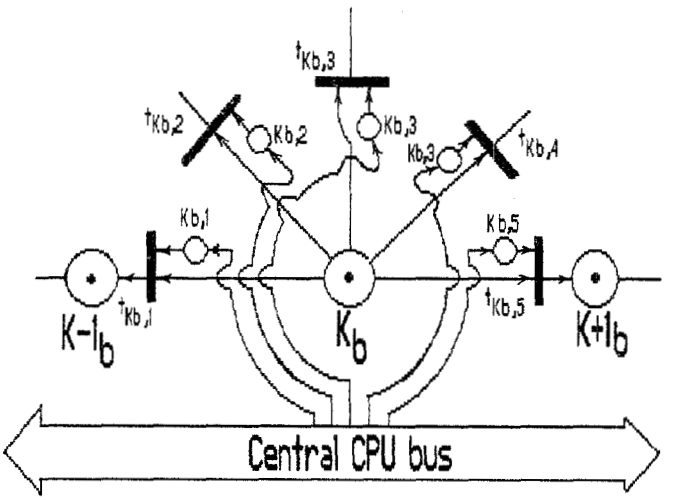

Figure 2b) Places and Out-transitions for the Kth Robot Pair of the backup line

Figure 2. Vertical Pair of Robots

	$K\text{-}1_a$	$K\text{-}1_b$	K_a	$K_{a,1}$	$K_{a,2}$	$K_{a,3}$	$K_{a,4}$	$K_{a,5}$	K_b	$K_{b,1}$	$K_{b,2}$	$K_{b,3}$	$K_{b,4}$	$K_{b,5}$	$K\text{+}1_a$	$K\text{+}1_b$
$t_{Ka,1}$	1	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0
$t_{Ka,2}$	0	1	-1	0	-1	0	0	0	0	0	0	0	0	0	0	0
$t_{Ka,3}$	0	0	-1	0	0	-1	0	0	1	0	0	0	0	0	0	0
$t_{Ka,4}$	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	1
$t_{Ka,5}$	0	0	-1	0	0	0	0	-1	0	0	0	0	0	0	1	0
$t_{Kb,1}$	0	1	0	0	0	0	0	0	-1	-1	0	0	0	0	0	0
$t_{Kb,2}$	1	0	0	0	0	0	0	0	-1	0	-1	0	0	0	0	0
$t_{Kb,3}$	0	0	1	0	0	0	0	0	-1	0	0	-1	0	0	0	0
$t_{Kb,4}$	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0	1	0
$t_{Kb,5}$	0	0	0	0	0	0	0	0	-1	0	0	0	0	-1	0	1

Figure 4a)

(The large 0s represent 5 columns of zeroes)

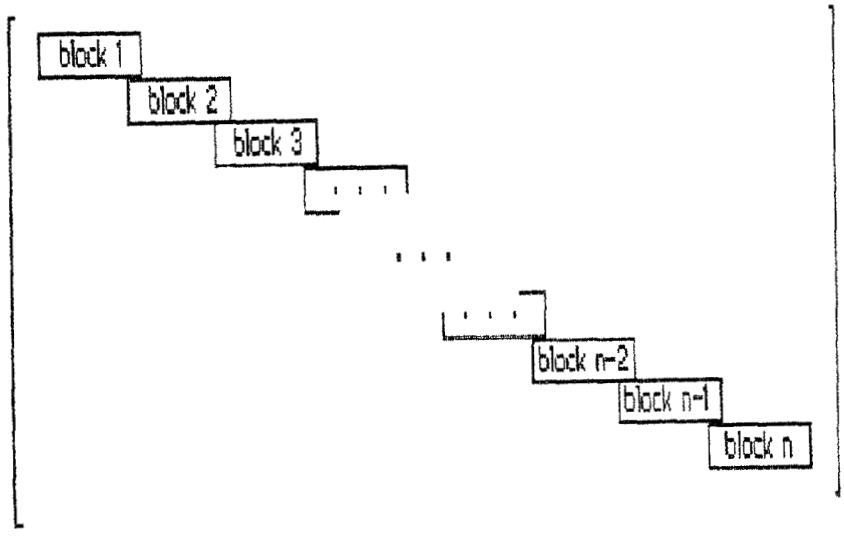

Figure 4b) Block configuration of C matrix

Figure 4. The Incidence Matrix