SYSTEMS OF COMMUNICATING MACHINES WITHOUT DEADLOCKS

Mohamed G. Gouda

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

TR-199 April 1982

ABSTRACT

Consider a system of two or more finite state machines which communicate exclusively by exchanging messages. We discuss two sufficient conditions to ensure that such a system is deadlock-free. The first condition ensures that at any deadlock state of the system, there exist exactly two machines which are deadlocked with one another. The second condition ensures that no two machines in the system can become deadlocked with one another. A number of VLSI arrays satisfy these two conditions; therefore, their freedom of communication deadlocks can be established based on our result.

KEYWORDS: Communicating finite state machines, communication deadlock, verification, VLSI array.

I. INTRODUCTION

Systems of processes which communicate exclusively by exchanging messages should be free of communication deadlocks. This has motivated a number of methodologies [4,6] to prove that such systems are deadlock-free. One methodology which has been widely used in the verification of communication protocols [1,3,5,10] can be summarized as follows. First, abstract each process in the considered system by suppressing its internal data structures and internal operations and by replacing each of its data-dependent decisions by a non-deterministic (i.e., arbitrary) decision. The abstract process is called a communicating finite state machine; and it contains only sending and receiving operations. Second, prove that the resulting system of communicating finite state machines is deadlock-free; this is sufficient to establish that the original system is deadlock-free. (The converse is not necessarily true.)

In this paper, we focus on the second step of the above methodology; in particular, we discuss a technique to prove that a system of communicating finite state machines is deadlock-free. The technique is based on two sufficient conditions which if satisfied by any system, then the system is deadlock-free.

The paper is organized as follows. Systems of communicating finite state machines are defined in Section II. The first and second sufficient conditions are discussed in Sections III and IV respectively. In Section V, we state the result that these two conditions are sufficient to establish freedom of deadlocks for any system of communicating machines. In Section VI, two examples of a vector multiplier and a rebound sorter are shown to be deadlock-free based on our technique. Concluding remarks are in Section VII.

II. SYSTEMS OF COMMUNICATING MACHINES

A system is a set of two or more communicating finite state machines (to be defined later). Let S be a system of r ($r \geq 2$) communicating finite state machines $M_1,\ldots,M_r$. We reserve the two variables i and j to be used as indices for the machines in S; thus, $1 \leq i, j \leq r$.

A communicating finite state machine M_i in S is a directed labelled graph with two types of nodes namely, sending and receiving nodes. A sending node to machine M_j ($j \neq i$) has one or more output edges; each of which is labelled s_j. A receiving node from machine M_j ($i \neq j$) has one or more output edges; each of which is labelled r_j. One of the nodes in M_i is identified as its initial node; and each node in M_i is reachable by a directed path from its initial node.

A state t of S is an r x r matrix. An entry t_{ij} in the ith row and jth column of t is defined as follows. If i = j then t_{ij} denotes a node in machine M_i else t_{ij} is a nonnegative integer.

The initial state t^0 of S is a state whose entries are defined as follows. If i = j then $t_{ij}{}^0$ denotes the initial node in machine M_i else $t_{ij}{}^0 = 0$.

Let t and t′ be two states of S. t′ is said to follow t by M_i sending to M_j ($i \neq j$) iff the following three conditions are satisfied:

1. Every entry in t' equals its corresponding entry in t except possibly for t_{ii}' and t_{ji}'.
2. t_{ii} is a sending node to M_j; and there is a directed edge from t_{ii} to t_{ii}'.
3. $t_{ji}' = t_{ji} + 1$.

Let t and t' be two states of S. t' is said to *follow* t *by* M_i *receiving from* M_j $(i \neq j)$ iff the following three conditions are satisfied:

1. Every entry in t' equals its corresponding entry in t except possibly for t_{ii}' and t_{ij}'.
2. t_{ii} is a receiving node from M_j; and there is a directed edge from t_{ii} to t_{ii}'.
3. $t_{ij} > 0$ and $t_{ij}' = t_{ij} - 1$.

Let t and t' be two states of S. t' is said to *follow* t if there exists M_i and M_j $(i \neq j)$ in S such that t' follows t either by M_i sending to M_j or by M_i receiving from M_j.

Let t and t' be two states of S, t' is said to be *reachable from* t if either $t' = t$ or there are states $t^1,\ldots,t^n$ such that $t = t^1$, $t' = t^n$, and t^{k+1} follows t^k for $1 \leq k \leq n - 1$.

A state t of S is said to be *reachable* if it is reachable from the initial state of S.

k distinct machines $M_{i_1},\ldots,M_{i_k}$ $(k \leq 2)$ in S are said to be *deadlocked at* state t of S iff the following $k + 1$ conditions are satisfied:

1. $t_{i_1 i_1}$ is a receiving node from M_{i_2}.

2. $t_{i_2 i_2}$ is a receiving node from M_{i_3}.

   ...

k. $t_{i_k i_k}$ is a receiving node from M_{i_1}.

k+1. $t_{i_1 i_2} = t_{i_2 i_3} = \ldots = t_{i_k i_1} = 0$.

A state t of S is said to be a deadlocked state iff there are k ($k \geq 2$) distinct machines in S which are deadlocked at t.

System S is deadlock-free if no deadlock state of S is reachable.

In the next section. we discuss a sufficient condition to ensure that t is a reachable deadlock state of S iff there are two distinct machines in S which are deadlocked at t.

III. BINARY SYSTEMS

The system graph G for a system S is a directed graph which satisfies the following two conditions:

1. For each machine M_i in S, there is a distinct node n_i in G.
2. There is a directed edge from n_i to n_j in G iff M_i has a sending node to M_j.

A system S is called binary iff any directed cycle with distinct nodes in its system graph has exactly two nodes.

Many VLSI arrays can be represented as binary systems; Figure 1 shows the system graphs for four such arrays. For the reader's convenience, we have labelled the directed edges in these system graphs to show the information being exchanged between different machines; for example, in Figure 1a, machine X_1 sends x_1 values to machine P_1 and receives acknowledgement "ack" messages from it. Two of those arrays, namely the vector multiplier (Figure 1a) and the rebound sorter (Figure 1d) are discussed in more detail in Section VI.

Theorem 1: Let S be a binary system. For any deadlock state t of S, there are two distinct machines which are deadlocked at t.

Proof: (by contradiction) Let t be a deadlock state of S; and assume that there are k ($k > 2$) distinct machines $M_{i_1},\ldots,M_{i_k}$ which are deadlocked at t. Therefore, the following k conditions are satisfied.

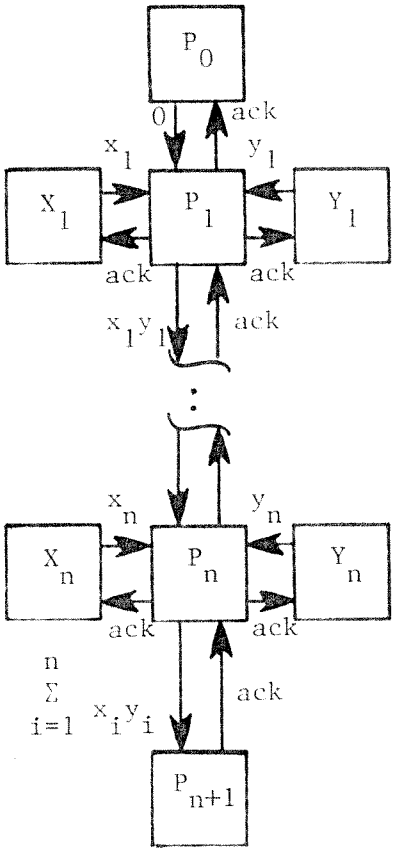
1. $t_{i_1 i_1}$ is a receiving node from M_{i_2}.
2. $t_{i_2 i_2}$ is a receiving node from M_{i_3}.

   ...

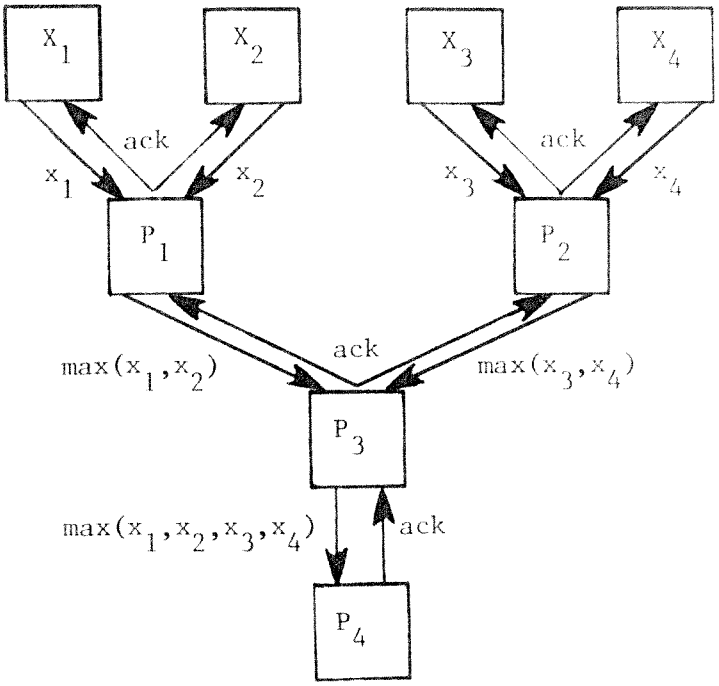
k. $t_{i_k i_k}$ is a receiving node from M_{i_1}.

From 1 to k, there is a directed cycle, in the system graph of S, which consists of the distinct nodes $n_{i_1},n_{i_2},\ldots,n_{i_k}$, where $k > 2$. This contradicts that S is binary. []

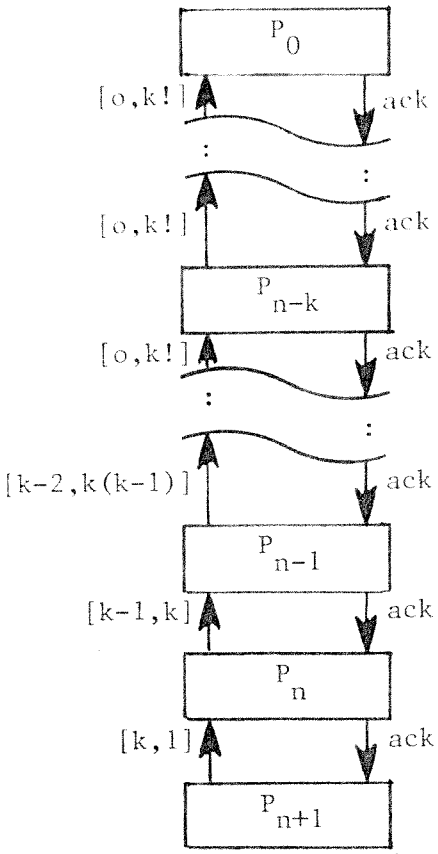
From Theorem 1 if no two machines in a binary system S can become deadlocked with one another, then S is deadlocked-free. In the next section, we discuss a sufficient condition to ensure that no two machines in a system S can become deadlocked with one another.



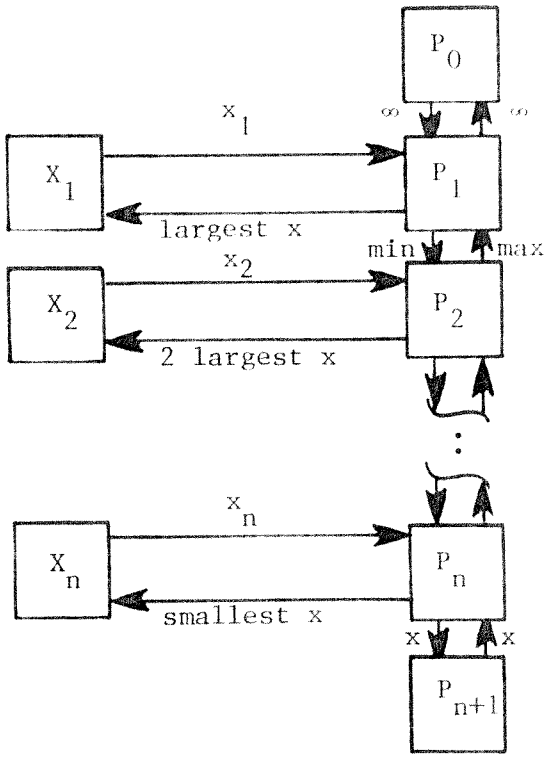
(a) Vector Multiplier.



(b) Comparator.



(c) Factorial Pipeline.



(d) Rebound Sorter.

Figure 1. System graph examples for binary VLSI arrays.

IV. MACHINE PROJECTIONS

A node in machine M_i is called an M_j-node iff it is either a sending node to M_j or a receiving node from M_j.

A machine M_i is said to be well-formed with respect to another machine M_j iff the following two conditions are satisfied:

1. There exists exactly one M_j-node n in M_i such that if a directed path from the initial node of M_i reaches an M_j-node then it must reach n before reaching any other M_j-node. Node n is called the initial M_j-node in M_i.
2. There exists a directed path from any M_j-node to some M_j-node in M_i.

Let M_i be a well-formed machine with respect to another machine M_j. A projection P_{ij} of M_i onto M_j is a communicating finite state machine constructed from M_i by the following two steps.

1. For any M_j-node in M_i
   do add a corresponding node to P_{ij}. The node in P_{ij} which corresponds to the initial M_j-node in M_i is called the initial node of P_{ij}.
2. For any directed path, without internal M_j-nodes, from an M_j-node m_1 to an M_j-node m_2 in M_i
   do add a directed edge to P_{ij} from node n_1 which corresponds to m_1 to node n_2 which corresponds to m_2 provided there is no edge from n_1 to n_2 already in P_{ij}. The added edge is labelled s_{ji} (or r_{ji}) if m_1 is a sending (or receiving respectively) node.

Let M_i and M_j ($i < j$) be two well-formed machines with respect to one another; and let P_{ij} and P_{ji} be the projections of M_i onto M_j and of M_j onto M_i respectively. The set $\{P_{ij}, P_{ji}\}$ is a system; its state s is a 2 x 2 matrix whose entries are defined as follows: s_{11} is a P_{ji}-node in P_{ij}, s_{22} is a P_{ij}-node in P_{ji}, and s_{12} and s_{21} are nonnegative integers. The above definitions of initial state, follows, reachable and deadlock states are all still valid in this case.

The concept of "machine projection" is related to the concept of "protocol projection" discussed in [2,8]. However, the development and application of this concept in this context is unique to our work.

Theorem 2: Let M_i and M_j ($i < j$) be two machines (in a system S) which are well-formed with respect to one another; and let P_{ij} and P_{ji} be the projections of M_i onto M_j and of M_j onto M_i respectively.

If S can reach a state t where:

1. t_{ii} is an M_j-node in M_i, and
2. t_{jj} is an M_i-node in M_j

then the system $\{P_{ij},P_{ji}\}$ can reach a state s where:

3. s_{11} is the P_{ji}-node (in P_{ij}) which corresponds to t_{ii},
4. s_{22} is the P_{ij}-node (in P_{ji}) which corresponds to t_{jj}, and
5. $s_{12} = t_{ij}$ and $s_{21} = t_{ji}$.

Proof: Assume that system S can reach a state t which satisfies conditions 1 and 2. Therefore, there are states $t^0,t^1,t^2,\ldots,t^n$ where t^0 is the initial state of S, $t^n = t$, and t^{k+1} follows t^k for $k = 0,\ldots,n - 1$. The distinct nodes in $t_{ii}^0,t_{ii}^1,\ldots,t_{ii}^n$ form a

directed path, in M_i, which starts from the initial node and ends at the M_j-node t_{ii}^n in M_i. This path should include the initial M_j-node in M_i. Let the M_j-nodes in this path be $t_{ii}^a, t_{ii}^b, \ldots, t_{ii}^c, t_{ii}^n$ where $0 \leq a < b \ldots < c < n$ and t_{ii}^1 is the initial M_j-node in M_i. From the definition of P_{ij}, P_{ij} must contain a directed path whose nodes are $u_a, u_b, \ldots, u_c, u_n$ where u_a is the initial P_{ji}-node (in P_{ij}) which corresponds to t_{ii}^a, u_b is the P_{ji}-node (in P_{ij}) which corresponds to t_{ii}^b, and so on. Also, u_a is the initial P_{ji}-node in P_{ij}. Similarly, the distinct nodes in $t_{jj}^x, t_{jj}^y, \ldots, t_{jj}^z, t_{jj}^n$ where $0 \leq x < y \ldots < z < n$ and t_{jj}^x is the initial M_i-node in M_j. P_{ji} must contain a directed path whose nodes are $v_x, v_y, \ldots, v_z, v_n$ where v_x is the initial P_{ij}-node (in P_{ji}) which corresponds to t_{jj}^x, v_y is the P_{ij}-node (in P_{ji}) which corresponds to $t_{jj}^y, \ldots$ and so on.

The two integers t_{ij}^n and t_{ji}^n can be computed as follows:

t_{ij}^n = The number of sending nodes in $(t_{jj}^n, t_{jj}^y, \ldots, t_{jj}^z)$

- The number of receiving nodes in $(t_{ii}^a, t_{ii}^b, \ldots, t_{ii}^c)$

= The number of sending nodes in $(v_x, v_y, \ldots, v_z)$

- The number of receiving nodes in $(u_a, u_b, \ldots, u_c)$,

and

t_{ji}^n = The number of sending nodes in $(u_a, u_b, \ldots, u_c)$

- The number of receiving nodes in $(v_x, v_y, \ldots, v_z)$

Therefore, if system S can reach state t^n, then system $\{P_{ij}, P_{ji}\}$ can reach a state s where, $s_{11} = u_n$, $s_{22} = v_n$, $s_{12} = t_{ij}^n$, and $s_{21} = t_{ji}^n$. []

Corollary 1: If system $\{P_{ij},P_{ji}\}$ is deadlock-free, then system S can never reach a state at which M_i and M_j are deadlocked.

Proof: (by contradiction) Assume that the system $\{P_{ij},P_{ji}\}$ is deadlock-free, and that system S can reach a state t at which M_i and M_j are deadlocked. State t must satisfy the following three conditions:

1. t_{ii} is a receiving node from M_j in M_i.
2. t_{jj} is a receiving node from M_i in M_j.
3. $t_{ij} = t_{ji} = 0$.

From Theorem 2, system $\{P_{ij},P_{ji}\}$ can reach a state s which satisfies the following three conditions:

1. s_{11} is a receiving node from P_{ji} in P_{ij}.
2. s_{22} is a receiving node from P_{ij} in P_{ji}.
3. $s_{12} = s_{21} = 0$.

But state s is a deadlock state for $\{P_{ij},P_{ji}\}$. Contradiction. []

V. SUFFICIENT CONDITIONS FOR FREEDOM OF DEADLOCKS

The next theorem follows immediately from Theorem 1 and Corollary 1; it states sufficient conditions for freedom of deadlocks for a system of communicating machines.

Theorem 3: Let S be a system of r communicating finite state machines $M_1,\ldots,M_r$.

If

1. S is binary, and

2. for any i and j ($1 \leq i$, $j \leq r$, and $i \neq j$): M_i and M_j are well-formed with respect to M_j and M_i respectively, and the system $\{P_{ij}, P_{ji}\}$ is deadlock-free.

then S is deadlock-free. []

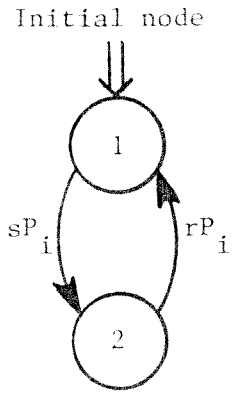
It is straightforward to verify that a system S is binary by examining its system graph G. It is also straightforward to verify that each machine M_i in S is well-formed with respect to each other machine in S; and to construct its projection P_{ij} onto each other machine M_j in S. A recent polynomial algorithm [8] can be used to verify that each system $\{P_{ij}, P_{ji}\}$ is deadlock-free; its time is $O(p_{ij}^3\ p_{ji}^3)$ and its space is $O(p_{ij}\ p_{ji})$ where p_{ij} and p_{ji} are the numbers of nodes in P_{ij} and P_{ji} respectively.

In the next section, we illustrate by two examples how to use Theorem 3 to verify that a VLSI array is free of deadlocks.

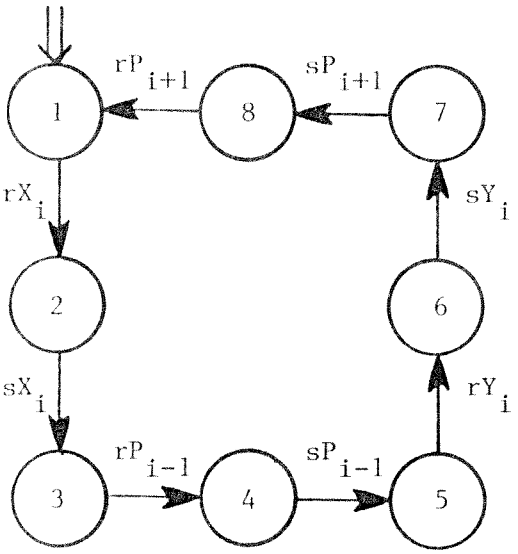
VI. EXAMPLES

A. A Vector Multiplier:

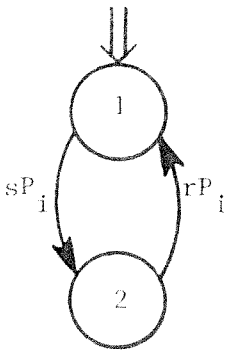
Consider the vector multiplier whose system graph is in Figure 1a. It consists of n X_i machines, n Y_i machines, and n+2 P_i machines. The machine X_i (i=1,...,n) is shown in Figure 2a. At each cycle, X_i sends an x_i value to P_i, then waits to receive an acknowledgement "ack" from it. Notice that an output of a sending node to P_i is labelled sP_i; and an output of a receiving node from P_i is labelled rP_i. The machine Y_i in Figure 2c is identical to the machine X_i (i=1,...,n) except for one "implicit" difference, namely X_i sends x_i values to P_i while Y_i sends y_i values to P_i.



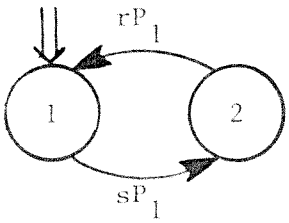
(a) Machine X_i ($i=1,\ldots,n$)



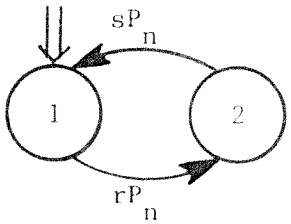
(b) Machine P_i ($i=1,\ldots,n$)



(c) Machine Y_i ($i=1,\ldots,n$)



(d) Machine P_0



(e) Machine P_{n+1}

Figure 2. Communicating machines in the vector multiplier whose system graph is in Figure 1a.

The machine P_i (i=1,...,n) is shown in Figure 2b. At each cycle, P_i receives an x_i value from X_i, and sends an ack to it. Then, P_i receives a partial sum $s_i = \sum_{j=1}^{i-1} x_j * y_j$ from P_{i-1}, and sends an ack to it. Then, P_i receives a y_i value from Y_i and sends an ack to it. Then, P_i sends a partial sum $s_{i+1} = s_i + x_i * y_i$ to P_{i+1} and waits to receive an ack from it. The two machines P_0 and P_{n+1} shown in Figures 2d and 2e respectively are special cases of P_i in Figure 2b.

To prove that this system is deadlock-free, it is sufficient by Theorem 1 to prove the following three assertions:

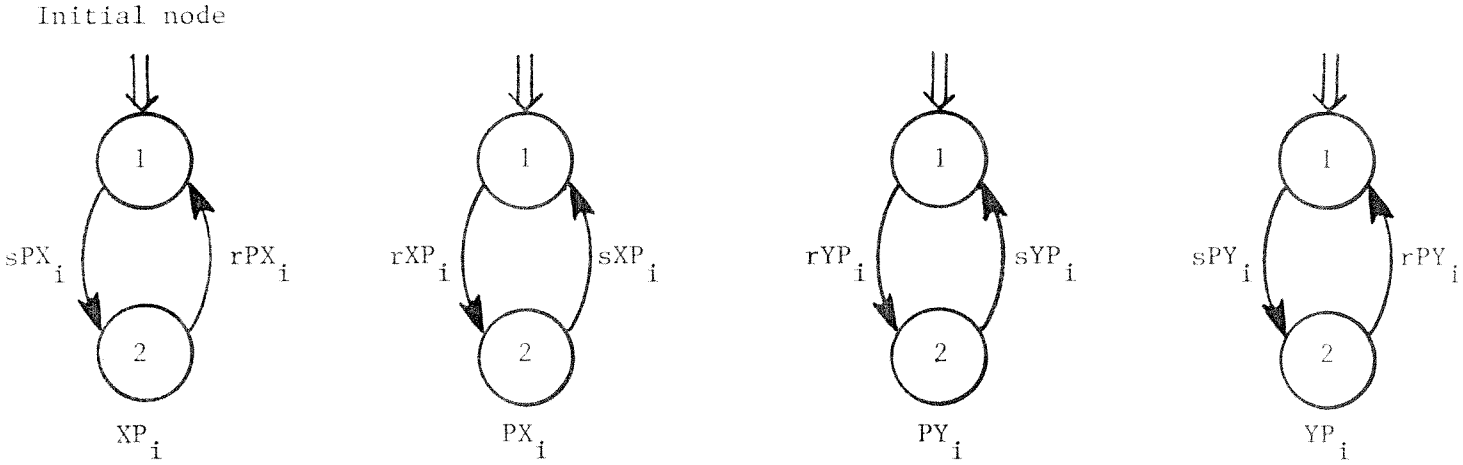
(i) For i=1,...,n, the two machines X_i and P_i can never become deadlocked with one another.

(ii) For i=1,...,n, the two machines Y_i and P_i can never become deadlocked with one another.

(iii) For i=1,...,n+1, the two machines P_{i-1} and P_i can never become deadlocked with one another.

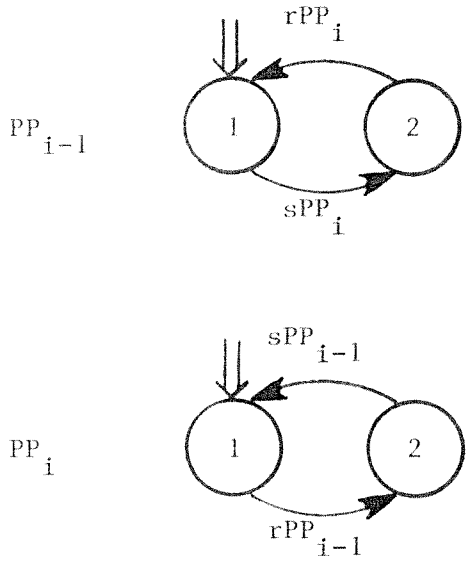
By Theorem 2, assertion (i) can be proved by showing that for i=1,...,n, the system $\{XP_i, PX_i\}$ is deadlock-free, where XP_i is the projection of X_i onto P_i and PX_i is the projection of P_i onto X_i. Figure 3a shows the two machines XP_i and PX_i, it is straightforward to show that they are deadlock-free [8].

Similarly, assertions (ii) and (iii) can be proved by showing that the two systems $\{YP_i, PY_i\}$ and $\{PP_i, PP_i\}$ in Figures 3b and 3c respectively are deadlock-free. This completes the proof that the vector multiplier in Figure 2 is deadlock-free. []



(a) The system $\{XP_i, PX_i\}$ $(i=1,\ldots,n)$

(b) The system $\{PY_i, YP_i\}$ $(i=1,\ldots,n)$



(c) The system $\{PP_{i-1}, PP_i\}$ $(i=1,\ldots,n+1)$

Figure 3. Three projection systems for the vector multiplier in Figures 1a and 2.

B. A Rebound Sorter:

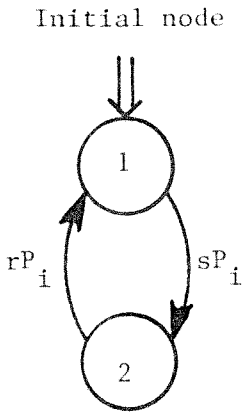
Consider the rebound sorter whose system graph is in Figure 1d. It consists of n X_i machines, and n+2 P_i machines. The machine X_i (i=1,...,n) is shown in Figure 4a. At each cycle, X_i sends an x_i value to P_i, then waits to receive the ith largest value in the set of values $\{x_1,x_2,...,x_n\}$.

The machine P_i (i=1,...,n) is shown in Figure 4b. The two outputs of node 5 constitute a nondeterministic decision which is an abstraction of a data-dependent decision as discussed in Section I. Machines P_0 and P_{n+1} in Figures 5c and 5d are special cases of P_i in Figure 4b.

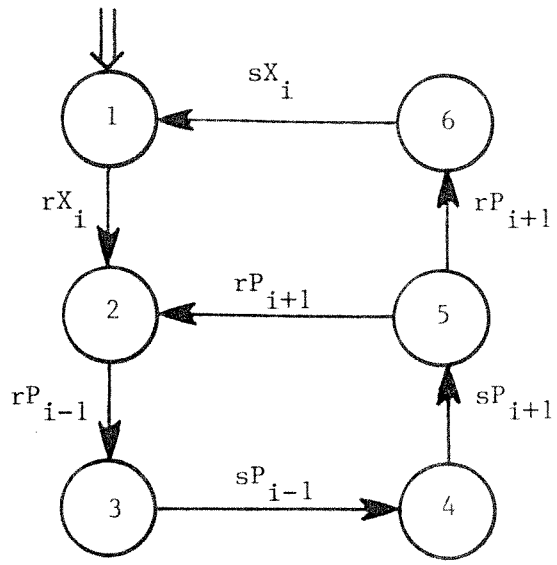
Notice that the projections XP_i and PX_i (i=1,...,n), of X_i onto P_i and of P_i onto X_i respectively, for the rebound sorter are identical to those in Figure 3a for the vector multiplier. Similarly, the projections PP_{i-1} and PP_i (i=1,...,n+1), of P_{i-1} onto P_i and of P_i onto P_{i-1} respectively, for the rebound sorter are identical to those in Figure 3c for the vector multiplier. Therefore, following a similar argument as in the previous example, it can be shown that the rebound sorter is free of communication deadlocks. []

VII. CONCLUDING REMARKS

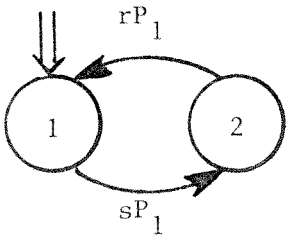
We have presented two sufficient conditions which can be used in conjunction with the algorithm in [8] to prove that a system of communicating finite state machines is deadlock-free. This technique has been applied successfully to a number of systems, including the VLSI arrays whose system graphs are in Figure 1.



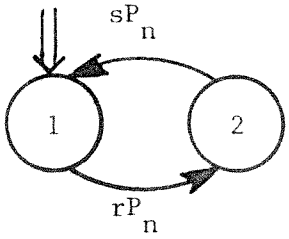
(a) Machine X_i
($i=1,\ldots,n$)



(b) Machine P_i
($i=1,\ldots,n$)



(c) Machine P_0



(d) Machine P_{n+1}

Figure 4. Communicating machines in the rebound sorter whose system graph is in Figure 1d.

The technique is attractive for VLSI arrays in particular since the resulting proofs are independent of the array sizes. For example, the vector multiplier of size "n" (Figure 2) is shown to be deadlock-free by showing that three projection systems of fixed sizes (Figure 3) are deadlock-free. Also, the rebound sorter of size "n" (Figure 4) is shown to be deadlock-free by showing that two projection systems of fixed sizes (Figures 3a and 3b) are deadlock-free.

Another attractive feature of this technique is that different systems can have the same projection systems and so can be proved deadlock-free by the same proof. For example, the projection systems for the rebound sorter (Figure 4) are part of the projection systems (Figure 3) of the vector multiplier (Figure 2). Therefore, proving that the vector mulltiplier is deadlock-free implies that the rebound sorter is deadlock-free.

The technique discussed in this paper is not applicable to important classes of VLSI arrays, e.g., two-dimensional rectangular and hexagonal arrays [7], since these arrays do not correspond to binary systems. Other techniques to deal with such classes are still needed.

REFERENCES

[1] G. V. Bochmann, "Finite state description of communication protocols," Computer Networks, Vol. 2, 1978, pp. 361-372.

[2] G. V. Bochmann, and P. Merlin, "On the construction of communication protocols," Proc. 5th ICCC, Oct. 1981.

[3] D. Brand, and P. Zafiropulo, "On communicating finite state machines," IBM Research Report, RZ1053, (#37725) Jan. 81.

[4] K. M. Chandy, and J. Misra, "Deadlock absence proofs for networks of communicating processes," Information Processing Letters, Vol. 9, No. 4, Nov. 1979, pp. 185-189.

[5] M. G. Gouda and Y.-T. Yu, "Designing deadlock-free and bounded communication protocols," Tech. Report 179, Dept. of Comp. Sciences, Univ. of Texas at Austin, June 1981.

[6] J. R. Jump, and P. S. Thiagarajan, "On the interconnection of asynchronous control structures," JACM, Vol. 22, No. 4, Oct. 1975, pp. 596-612.

[7] C. Mead, and L. Conway, Introduction to VLSI systems, Addison-Wesley Pub. Co., Inc., 1980.

[8] S. S. Lam, and A. U. Shankar, "Protocol projections: a method for analyzing communication protocols," Proc. National Telecomm. Conf., 1981.

[9] Y.-T. Yu, and M. G. Gouda, "Deadlock detection for a class of communicating finite state machines," Tech. Report 193, Dept. of Comp. Sciences, Univ. of Texas at Austin, Feb. 1982.

[10] P. Zafiropulo, et al, "Towards analyzing and synthesizing protocols," IEEE Trans. on Comm., Vol. COM-28, No. 4, April 1980, pp. 651-661.