AN EFFICIENT METHOD FOR THE REPRESENTATION AND TRANSMISSION OF MESSAGE PATTERNS

P.J. Bernhard
Department of Computer Science
Clemson University
Clemson, S.C. 29634

Abstract
In this paper we describe a formalism for the compact representation of message patterns for multistage interconnection networks. In this formalism a descriptor called an (s,d)-mask is used to represent a message pattern, or rather, a set of messages. We show that when message patterns are represented in this way a number of their properties can be determined in polynomial time. This includes determining if a message pattern creates conflicts or congestion. In addition, we show that the minimum round partitioning problem, which in general is NP-complete, can be solved in polynomial time for any message pattern which can be represented by a single (s,d)-mask. This generalizes a known result to a more general class of message patterns and a more general class of networks.

Keywords: Omega network, routing, computational complexity, SIMD, parallel processing.

1 Introduction
In [La73] Lawrie proposed the Omega network as an interconnection network for a multiprocessor system. For this network a particular message can be represented by a source-address, destination-address pair, abbreviated as an (s,d) pair, where s is the binary address of the source of the message and d is the binary address of the destination of the message. Hence, a message pattern can be represented by a set of (s,d) pairs, where each (s,d) pair corresponds to one message.

In this paper we develop a formalism, called the mask language, for the representation and transmission of message patterns on Omega networks. In this formalism a message pattern can be represented by a single descriptor called a (s,d)-mask. This representation has a number of advantages. For example, a single (s,d)-mask can represent a number of (s,d) pairs which is exponential in the size of the (s,d)-mask. Hence, it saves space and, in the context of multiprocessor communication, a single (s,d)-mask can be broadcast to all processors rather than sending the entire set of (s,d) pairs to their respective processors. In addition, we show that when a message pattern is represented by an (s,d)-mask, a number of properties of the message pattern can be determined in polynomial time simply by examining the (s,d)-mask rather than the entire corresponding message pattern. Since a message pattern can be exponentially large compared to its corresponding (s,d)-mask, this fact illustrates one of the main advantages of representing message patterns in the mask language. In addition, we show that the mask language defines a class of message patterns for which the minimum round partitioning problem can be solved in polynomial time for a general class of networks called bundled Omega networks.

2 The Omega Network
Following Lawrie [La75], an N-input N-output Omega network (also called an N x N Omega network), where N = 2^m, consists of m identical stages. Each stage consists of a perfect shuffle wire interconnection [St71] followed by N/2 switching elements. In Figure 1(a) we show an 8 x 8 Omega network, and in Figures 1(b)-1(g) we show the possible states for each of the switches. Figure 1(b) shows the "straight through" state where the input signals are sent directly to the corresponding outputs, 1(c) shows the "interchange" state where the input signals are first interchanged before being sent to the outputs and Figures 1(d)-1(g) show "incomplete" states. For example, in Figure 1(d) a signal is passed from the upper input to the upper output while nothing is on the lower input or lower output. Note how the model here differs from the one in [La75] since switches are not allowed to "broadcast" messages. In Figure 1(a) we have labeled the interconnection links for each stage, from the top down, with a log_2 N bit binary address. We have also numbered the stages and shown a path through the network from input 000 to output 011.

A particular path through the network can be represented by a source-destination pair, abbreviated as an
(s,d) pair, where the source s = s0s1... sm−1 is the binary address of the input at the first stage, the destination d= d0d1... dm−1 is the binary address of the output at the last stage and m= log2N . Careful examination of the network shows that the path code s0s1... sm−1d0d1... dm−1 completely determines a unique path through the network. Specifically, if we define an m bit window W_i as the bit pattern beginning at bit position i of the path code, we see that at stage i in the network, where 0 ≤ i ≤ m, the path which goes from s0s1... sm−1 to d0d1... dm−1 makes use of the link with address W_i= s_is_{i+1}... sm−1d0d1... dm−1 [RV86]. For example, Figure 1(a) shows a path from 000 to 011. For this path W_2=001 and at stage 2 the path makes use of the link with address 001.

The fact that a path code uniquely determines a path through the network enables communication conflicts in the network to be detected easily. Two messages that are being transmitted through the network will conflict if and only if they require use of a common link in the network. Hence, in light of the window property mentioned above, two (s,d) pairs are said to conflict if and only if there exists an i such that the two (s,d) pairs have the same bit pattern on window W_i. For example, Figure 1(a) shows the paths (0000,1000) and (1100,1001). Both (s,d) pairs have W_2=001 and at stage 2 the path makes use of the link with address 001.

This concept of an Omega network can be generalized by the addition of a new parameter b called the bundle size of the network. Specifically, we define a b(N)x b(N) Omega network, where N=2^m, to have bundle size b if each switch in the network has two bundles of inputs and two bundles of outputs, each of size b. For example, in Figure 2 we show a (3)x (3) Omega network. Each bundle in the network may carry b or fewer signals into a switch. Hence, a total of at most 2b signals may be input to a switch at any given time. Similarly, each output bundle may carry b or fewer signals out of a switch. For each input bundle, the incoming signals may be sent to the upper or lower output bundle. However, all the signals on a given input bundle don't necessarily have to go to the same output bundle. Some may go to the upper output bundle, while others may go to the lower. Similarly, two signals on different input bundles may go to the same output bundle. The only constraint is that at most b signals can use a particular output bundle at any given time. If more than b require use of the same output bundle then we say that congestion occurs. The definition of the standard Omega network is a special case of the generalized definition, where b=1. Similarly “conflict” is just a special case of “congestion”. Bundled networks have also been considered in [SH87], where a bundled network was referred to as a diluted network.

Recall that two (s,d) pairs are said to conflict at stage i in the network if and only if they have the same bit pattern on window W_i. However, when b > 1 the fact that two pairs have the same bit pattern on window W_i doesn't necessarily imply that congestion occurs. In order for congestion to occur at stage i, at least b+1 pairs must have the same bit pattern on window W_i. For example, consider the paths (0000,1000) and (1100,1001) on a (2)x(2) Omega network. These pairs have the same bit pattern 0010 on W_3, and hence, the bundle at stage 2 with address 0010 is full because the two paths are in use at the same time. If we now consider the path (0100,1010) we see that at stage 2 this also requires use of the bundle 0100. Hence, if all three paths were required to be in use at the same time, congestion would occur. An example of a bundled Omega network with b=16 is in the proposed G.E. Cross Omega machine [H86].

Finally, define a message pattern to be a set of (s,d) pairs. Each (s,d) pair in the set represents the fact that a message is to be sent from input s to output d of the network. Note that this definition imposes no restrictions on what type of message pattern the set represents. For example, many pairs may have the same source or the same destination. Furthermore, any number of conflicts may exist in the set.

3 Definition of the Mask Language

Define the mask language as follows. Symbols used in the language will include constants and literals. Constants are 0 and 1, Literals include variables "x_0", "x_1", "x_2", etc. and their complements. A mask is any sequence of symbols such as 0001, 111, z_0z_1z_2, etc. The length of a mask M is the number of symbol occurrences in the mask. Each mask has an implicit universal quantifier to the left of the mask for each variable contained within. Hence, a mask containing the variables x_0, x_1, ..., x_q−1 is said to represent the set S of 2^q addresses, each specified by one of the 2^q functions from the variables x_0, x_1, ..., x_q−1 to the set {0,1}. For example, the mask z_0z_1z_2 represents the set of addresses {0100, 0110, 1100, 1110}. Furthermore, each address in the set is said to be covered by the corresponding mask.

In the case where a mask contains no variables, such as the mask 101, then the mask represents the set which contains only itself {101}.

An (s,d)-mask consists of a left hand side and a right hand side, where each is a mask of the same length. Examples of (s,d)-masks are (000100, 0100 01) and (z_0z_1z_2z_3, z_1z_2z_3z_4). As with masks, an (s,d)-mask has an implicit universal quantifier to the left of the (s,d)-mask for each variable contained within. Hence, the (s,d)-mask is said to represent the corresponding set of (s,d) pairs. For example, the (s,d)-mask M=(z_0010, z_1z_2) represents the set S={{010, 110}, {010, 111}, {110, 010}, {110, 011}}.
4 (s,d)-Masks and Detecting Congestion

In this section, we describe a structure called a conflict-cube [RV86]. Each (s,d)-mask has a conflict-cube associated with each window of the (s,d)-mask. As we shall show, the conflict-cubes associated with a given (s,d)-mask can be used to determine a number of properties of the corresponding message pattern.

Suppose that $M$ is an (s,d)-mask and let $V$ be the set of variables which occur or whose complements occur in $M$. Furthermore, let $V_j$ be the set of variables which occur or whose complements occur in window $W_j$ of the (s,d)-mask, where $0 \leq j \leq m$. The conflict-cube $S_{M,j}$ of $M$ corresponding to window $W_j$ is the set $S_{M,j} = V \setminus V_j$. Note that this definition is a slight variation of the one given in [RV86].

Now let $S$ be the message pattern corresponding to $M$. Then the following property of $S$ holds [RV86].

**Fact 1.** Consider an Omega network with bundle size $b = 1$. Then the number of messages which conflict on a particular link at stage $j$ is given by $2^k$, where $k$ is the cardinality of the corresponding conflict-cube $S_{M,j}$. Hence, a message pattern represented by a single (s,d)-mask will contain conflicts if and only if it has a nonempty conflict-cube.

In following sections, we will show how conflict-cubes can be used to determine a number of properties of (s,d)-masks and their corresponding message patterns. Furthermore, conflict-cubes can be exploited in the solution to the minimum round partitioning problem for any message pattern which can be represented by a single (s,d)-mask.

5 Detecting Conflicts in an (s,d)-mask

The Omega network is a blocking network, and, as such, does not allow the transmission of arbitrary message patterns. Specifically, it does not allow the transmission of message patterns which give rise to communication conflicts. Hence, algorithms for detecting communication conflicts and strategies for dealing with communication conflicts have become the focus of numerous researchers. As stated in Section 1, one of the advantages of the mask language is that many properties of message patterns can be determined simply by examining (s,d)-masks rather than the entire corresponding message pattern. The following lemmas illustrate this for the detection of conflicts and congestion in a message pattern represented by one or more (s,d)-masks. It should be noted that for the lemmas and theorems in this paper we give short sketches of the proofs. We refer the interested reader to [B88] for the detailed versions.

**Lemma 2.** Let $M$ be an (s,d)-mask of length $m$. Then determining if the message pattern corresponding to $M$ contains communication conflicts can be done in $O(m)$ time.

**Proof.** (sketch) By Fact 1 in Section 4, a given (s,d)-mask will contain conflicts if and only if it has a nonempty conflict-cube. Hence, an algorithm for detecting conflicts would operate by scanning the (s,d)-mask from left to right checking for a nonempty conflict-cube. The key to the algorithm lies in the fact that each window is examined using only a constant amount of time, thus ensuring that the algorithm operates in linear time.

In [BR87] an algorithm is discussed which will determine if a given set $S$ of (s,d) pairs contains communication conflicts. The algorithm operates in time $O(m^2 \log m)$, where $m$ is the length of a corresponding mask pair, Lemma 2 illustrates one of the main advantages of using (s,d)-masks for representing message patterns. We now consider the more general case of detecting congestion in an Omega network with bundle size $b \geq 1$.

**Lemma 3.** Let $M$ be an (s,d)-mask of length $m$ and $b \geq 1$ be a bundle size. Then determining if the message pattern corresponding to $M$ is congestion-free for an Omega network with bundle size $b$ can be done in $O(m)$ time.

**Proof.** (sketch) As in the proof of Lemma 2, an algorithm for detecting congestion would scan the (s,d)-mask from left to right examining the conflict cubes at each window. However, it follows from Fact 1 in Section 4, that in order for congestion to occur on an Omega network with bundle size $b > 1$, the (s,d)-mask must contain a conflict-cube of size $k$, where $2^k > b$.

In addition to message patterns representable by a single (s,d)-mask, we consider message patterns which require more than one (s,d)-mask for their representation. Hence, it becomes important to be able to detect communication conflicts and congestion in a set of (s,d)-masks. In the following, a set of (s,d)-masks is said to be disjoint if no (s,d) pair is covered by two different (s,d)-masks in the set.

**Theorem 4.** Let $S$ be a set of disjoint (s,d)-masks where $n = |S|$ and $m$ is the length of each (s,d)-mask in $S$. Then determining if $S$ is conflict-free can be done in $O(m n^2)$ time.

**Proof.** (sketch) An algorithm for detecting if a set of (s,d)-masks contains conflicts would operate by scanning all of the (s,d)-masks, at the same time, from left to right. As it scans it would examine the set of (s,d)-masks on each window. For each window it would compare each pair of (s,d)-masks to see if they conflict on that window. This can be determined by a reduction to an instance of 2-SAT, which can be solved in $O(m)$ time [GJ79]. Since each pair of (s,d)-masks must be compared on each window, a total of $O(n^2 m)$ 2-SAT instances must be solved for each window. Hence, each window requires an $O(mn^2)$
operation. Since there are a total of \(m + 1\) windows to be examined, this gives a total running time of \(O(m^2 n^2)\).

**Theorem 5.** Let \(b \geq 1\) be a bundle size and \(S\) a set of \(n\) disjoint \((s,d)\)-masks, where \(n\) is fixed and each \((s,d)\)-mask in \(S\) is of length \(m\). Then determining if the set \(S\) is congestion-free for an Omega network with bundle size \(b\) can be done in \(O(m^2)\) time.

**Proof.** (sketch) As in Theorem 4 an algorithm for testing for congestion will check each of the \(m+1\) windows. In addition, for each window each of the \(2^n\) subsets of the \(n\) masks must be checked to see if a subset of the \((s,d)\)-masks covers a set of conflicting \((s,d)\)-pairs, at least one \((s,d)\)-pair per mask. The test of each such subset requires that an instance of 2-SATISFIABILITY be solved, which requires \(O(m)\) time. However, since the bundle size of the network may be greater than 1, an additional counting step must be performed. Since there are \(m + 1\) windows and since \(n\) is fixed the running time of the algorithm is \(O(m^2)\). It should be noted that since each window requires \(2^n\) 2-SATISFIABILITY instances to be solved the constant on the running time is exponential in \(n\).

As with Lemma 2, the above results illustrate one of the main advantages of using \((s,d)\)-masks. Specifically, the corresponding algorithms which operate on sets of \((s,d)\) pairs, instead of \((s,d)\)-masks, may require an exponential increase in time.

### 6 Minimum Round Partitioning for \((S,D)\) Masks

Suppose that a message pattern is to be transmitted on an Omega network. In addition, suppose that it has been determined that the message pattern creates congestion. One strategy for dealing with this situation is to partition the corresponding set of messages into disjoint, congestion-free subsets, called rounds, and then transmitting the set of messages by successively transmitting the messages in each round. Clearly, in order to minimize the total time for message transmission, it is important to minimize the total number of rounds.

The problem of partitioning a set of \((s,d)\) pairs into a minimum number of rounds is referred to as the minimum round partitioning problem. This problem has previously been considered by a number of authors. For example, in [A83] upper and lower bounds for the problem have been established. In [WF80] and [DF87] heuristics for the problem are given. An algorithm is given in [RV86] which will construct a "partitioning function" for a set of messages when the message pattern is represented as a "bit permute complement" permutation. And in [BR87] the computational complexity of the problem was considered. For a number of special cases the problem was shown to be solvable in polynomial time, however, it was shown in general to be NP-hard. Here, we show that the problem can be solved in polynomial time when the message pattern can be represented by a single \((s,d)\)-mask.

**Theorem 6.** Let \(S\) be a message pattern which can be represented by a single \((s,d)\)-mask. Then \(S\) can be partitioned into a minimum number of congestion-free rounds for an Omega network with bundle size \(b \geq 1\) in linear time.

**Proof.** The algorithm for performing the partitioning exploits two facts related to message patterns which can be represented by a single \((s,d)\)-mask. The first is that for any such message pattern the \((s,d)\)-mask can be computed from the set \(S\) in linear time.[B88]. And the second is that the minimum number of rounds required by the message pattern is equivalent to \(2^k\), where \(k\) is the cardinality of the largest conflict cube for the corresponding \((s,d)\)-mask. Given the message pattern as input, the algorithm will compute the \((s,d)\)-mask and then determine how many rounds are required by examining the associated conflict-cubes. Using this information it will then partition the message pattern. Each of these steps can be performed in linear time. Hence, the result follows.

### 7 Conclusion

In this paper we have described a formalism for the compact representation of message patterns. We have shown that when message patterns are represented in this formalism a number of their properties can be determined in polynomial time, simply by examining representative \((s,d)\)-masks rather than the message patterns themselves. This fact is important since a message pattern may be exponentially large compared with its corresponding \((s,d)\)-mask. In addition, we have shown that the minimum round partitioning problem, which in general is NP-complete, can be solved in polynomial time for any message pattern that is representable by a single \((s,d)\)-mask. This generalizes a known result [RV86] to a more general class of message patterns and a more general class of networks.

### References


[H86] Hardy, R., personal communication.


---

Figure 1(a): An Omega network.

Figure 1(b)-(g): Possible switch states.

Figure 2: A 3x4 by 3x4 Omega Network.