VI. Summary

m/u-degradable agreement protocol that achieves Lamport's Byzantine agreement [6] up to m faults, and a degraded form of agreement with more than m but at most u faults is proposed. Up to m faults, all the fault-free nodes agree on an identical value. For more than m faults (up to u faults), the degraded form of agreement allows the fault-free nodes to agree on at most two different values, one of which is necessarily the default value; the other value is the sender's value if the sender is fault-free. It is shown that $2m + u + 1$ nodes are necessary and sufficient to achieve m/u-degradable agreement. An m/u-degradable agreement algorithm is presented for more than $2m + u$ nodes. Also, it can be shown that network connectivity of $m + u + 1$ is necessary and sufficient to perform m/u-degradable agreement.

Further research is required to explore applications of degradable agreement. This paper also formulates the problem of degradable clock synchronization which is a subject of further research. Degradable agreement approach has been extended to the hybrid fault model [10]. This recent work shows that degradable agreement can effectively trade reliability with safety.

VII. Appendix

Algorithm BYZ (0,0) below achieves $0/u$-degradable agreement, provided $N \geq u + 1$.

1) The sender sends its value to all the $(N - 1)$ receivers.
2) Each receiver broadcasts the value it received from the sender to $(N - 2)$ other receivers. As there are $(N - 1)$ receivers, each receiver now has $(N - 1)$ values.
3) Each receiver uses VOTE $(N - 1, N - 1)$ of these $(N - 1)$ values.

References

[1] D. Dolev, "The Byzantine generals strike again," *J. Algorithm*, pp. 14–30, 1982.
[2] D. Dolev, J. Y. Halpern, and H. R. Strong, "On the possibility and impossibility of achieving clock synchronization," *J. Comput. and Syst. Sci.*, vol. 32, pp. 230–250, 1986.
[3] R. E. Harper, J. H. Lala, and J. J. Deyst, "Fault tolerant parallel processor architecture overview," in *Dig. of Papers: The 18th Int. Symp. Fault-Tolerant Computat.* 1988, pp. 252–257.
[4] C. M. Krishna and I. S. Bhandari, "On graceful degradation of phase locked clocks," in *IEEE Real-Time Syst. Symp.*, 1988, pp. 202–211.
[5] L. Lamport, "The weak Byzantine generals problem," *J. ACM*, vol. 30, pp. 668–676, July 1983.
[6] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Prog. Lang. Syst.*, vol. 4, pp. 382–401, July 1982.
[7] J. F. Paris, "Voting with witnesses: A consistency scheme for replicated files," in *Int. Conf. Distrib. Computing Syst.* 1986, pp. 606–612.
[8] T. B. Smith, III *et al.*, *The Fault-Tolerant Multiprocessor Computer*. Park Ridge, NJ: Noyes Publications.
[9] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *J. ACM*, pp. 626–645, July 1987.
[10] N. H. Vaidya, "Degradable agreement with hybrid faults (an algorithm and reliability-safety analysis)," Tech. Rep. 93-037, Comput. Sci. Dep., Texas A&M Univ., College Station, TX, College Station, TX, Aug. 1993.
[11] N. H. Vaidya and D. K. Pradhan, "Degradable agreement in the presence of Byzantine faults," in *Int. Conf. Distrib. Computing Syst.* May 1993, pp. 237–244.
[12] ——, "Degradable Byzantine agreement," Tech. Rep. 93-015, Comput. Sci. Dep., Texas A&M Univ., Mar. 1993. (supercedes report #92-020 by N. H. Vaidya.).
[13] N. H. Vaidya, "Degradable agreement in the presence of Byzantine faults," Tech. Rep. 92–020, Comput. Sci. Dep., Texas A&M Univ., 1992.

Wildcard Dimensions, Coding Theory and Fault-Tolerant Meshes and Hypercubes

Jehoshua Bruck, Robert Cypher, and Ching-Tien Ho

***Abstract*—Hypercubes, meshes and tori are well known interconnection networks for parallel computers. The sets of edges in those graphs can be partitioned to dimensions. It is well known that the hypercube can be extended by adding a *wildcard* dimension resulting in a *folded* hypercube that has better fault-tolerant and communication capabilities. First we prove that the folded hypercube is optimal in the sense that only a single wildcard dimension can be added to the hypercube. We then investigate the idea of adding wildcard dimensions to d-dimensional meshes and tori. Using techniques from error correcting codes we construct d-dimensional meshes and tori with wildcard dimensions. Finally, we show how these constructions can be used to tolerate edge and node faults in mesh and torus networks.**

I. Introduction

The mesh and the torus are two of the most important networks for parallel computers. A great deal of research has focused on the mesh and torus networks and several parallel computers have been built with 2- or 3-dimensional mesh or torus topologies. Examples include the MPP (of Goodyear Aerospace), the MP-1 (sold by MASPAR), Victor (of IBM), Paragon (of Intel), T3D (of Cray), and Parsytec. One of the most important issues in the design of a system which contains many components is the system's performance in the presence of faults. Hence, it is of major practical importance to develop efficient techniques (in terms of the cost of the redundancy) to handle faults in mesh and torus architectures.

Our approach is based on a graph model. In this model the architecture is viewed as a graph, where the nodes represent the processors and the edges represent communication links between the nodes. A target graph with N nodes is first selected. Next, the required amount of fault-tolerance, k, is determined. Then a fault-tolerant graph with N nodes is defined with the property that given any set of k or fewer faulty edges, the remaining graph (after removal of the faulty edges) is guaranteed to contain the target graph as a subgraph. Note that this approach guarantees that any algorithm designed for the target graph will run *with no slowdown* in the presence of k or fewer edge faults in the fault-tolerant graph, regardless of their distribution. Minimizing the cost in this model amounts to constructing a fault-tolerant graph with minimum degree.

While our focus is on tolerating edge faults, it is natural to also consider what happens when there are node faults [6]. Because our fault-tolerant constructions do not have redundant nodes, it is clear that the target graph will not be contained in the healthy portion of the fault-tolerant graph when it contains node faults. However, we will show that our constructions also have optimal connectivity properties, given their degree. More specifically, we will show that

Manuscript received May 10, 1993.

J. Bruck is with the California Institute of Technlogy, Mail Code 116–81, Pasadena, CA 91125 USA; E-mail: bruck@systems.caltech.edu.

R. Cypher is with the Computer Science Department, The Johns Hopkins University, Baltimore, MD 21228 USA; E-mail: cypher@cs.jhu.edu.

C.-T. Ho are with the IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120 USA; E-mail: ho@almaden.ibm.com.

IEEE Log Number 9404681.

if the number of node faults is less than the degree of the fault-tolerant graph, the healthy portion of the fault-tolerant graph remains connected.

The key to our fault-tolerant mesh constructions is a technique for adding redundant edges using what we call *wildcard dimensions*. While it was known that it is possible to add a single wildcard dimension to a hypercube [1], it was not known if more wildcard dimensions can be added to hypercubes, nor was it known if wildcard dimensions can be added to mesh and torus networks. In this brief contribution, we present the following results: 1) a proof that it is *not* possible to add more than a single wildcard dimension to a hypercube, 2) a technique, based on error-correcting codes, for adding wildcard dimensions to mesh and torus networks, and 3) a proof that meshes and tori with wildcard dimensions can tolerate edge faults with no slowdown in performance and that they also have optimal connectivity properties.

The brief contribution is organized as follows. In Section II, we define the concept of a wildcard dimension and prove that only a single wildcard dimension can be added to the hypercube. In Section III, we show how tori networks can be represented using a new algebraic framework. We use this new framework to characterize tori with wildcard dimensions. In Section IV, we use techniques from error correcting codes to construct tori networks with wildcard dimensions. In Section V, we show how these constructions can be used to tolerate edge faults in mesh and tori networks. Finally, in Section VI, we prove that our constructions have optimal connectivity properties.

II. Wildcard Dimensions in Hypercubes

Let Q_d denote the d-dimensional hypercube. It consists of $N = 2^d$ nodes, where node $0 \leq i \leq N-1$ is represented by the d bit binary representation of i. Two nodes, say $X = (x_{d-1}x_{d-2}\cdots x_0)$ and $Y = (y_{d-1}y_{d-2}\cdots y_0)$, are connected by an edge iff there is a single j for which $x_j \neq y_j$, and this edge is called a dimension-j edge. Hence, the set of edges in the hypercube can be partitioned to d dimensions.

In [11] it was suggested to add another set of edges to the hypercube, called the wildcard dimension, resulting in a *folded* hypercube network. (A related idea of adding an extra dimension was presented in the definition of the extra-stage cube network [1].) The folded hypercube topology was also defined independently in [8], but with a different name (the bisectional interconnection network) and with a different addressing scheme for the nodes. A formal definition of the folded hypercube network is given next.

Definition 1: A d-dimensional *folded* hypercube, denoted by F_d, is a d-dimensional hypercube to which extra links are added connecting every pair of nodes that are bit-wise complements of each other.

A wildcard edge in F_d is an edge that connects node $X = (x_{d-1}x_{d-2}\cdots x_0)$ and node $\bar{X} = (\bar{x}_{d-1}\bar{x}_{d-2}\cdots\bar{x}_0)$. For example, in F_4, the neighbors of node (0000) are $(0001), (0010), (0100)$, (1000) and (1111).

The structure of Q_d and F_d can be described using a more general framework, see also [10]. The idea is to assume that every node in a graph with 2^d nodes is represented by a unique string of d bits. The edges are specified by a set of *offsets*, denoted by S, of binary vectors of length d. Any two nodes, say X and Y, are connected by an edge iff there exists a vector $V \in S$ such that $X + V = Y$ where addition is vector addition performed over $GF(2)$. We use the notation $\{0,\cdots,n-1\}^d$ to denote the set $\{V \mid V = (v_1,\cdots,v_d) \text{ where } v_i \in \{0,\cdots,n-1\}\}$. In general we have the following.

Definition 2: Let n and d be positive integers. Let $S \subseteq \{0,\cdots,n-1\}^d$. The graph $G(n,d,S)$ is a graph with n^d nodes and edges specified by the set of vectors in S. Namely, two nodes, say X and Y, are connected by an edge iff there exists a vector $V \in S$ such that $X + V = Y$ or $Y + V = X$ where addition is performed modulo n.

Definition 3: A dimension in the graph $G(n,d,S)$ is the set of edges that corresponds to a vector in the set S. Namely, an edge (with endpoints X and Y) is in the dimension that corresponds to a vector $V \in S$ if either $X + V = Y$ or $Y + V = X$ where addition is performed modulo n.

Now we define the concept of a wildcard dimension.

Definition 4: Let n and d be positive integers. Let $S_1 \subseteq \{0,\cdots,n-1\}^d$, $S_2 \subseteq \{0,\cdots,n-1\}^d$ and $\hat{S} = S_1 \cup S_2$. The vectors in the set S_2 correspond to wildcard dimensions if for every set $S_3 \subseteq \hat{S}$ with $|S_1|$ vectors the graph $G(n,d,S_3)$ is isomorphic to $G(n,d,S_1)$.

We note here that in most of our results we assume that n is a prime. In this case, addition and multiplication over $GF(n)$ are defined as addition and multiplication modulo n. Some of our results are correct also for n which is a power of a prime. In this case, addition and multiplication over $GF(n)$ can be defined modulo a preselected irreducible polynomial of degree n.

Using this framework it is clear that $Q_d = G(2,d,S_1)$ where, $S_1 = \{000\cdots001, 000\cdots010, \cdots, 100\cdots000\}$, namely, the set of d vectors with Hamming weight one. (The Hamming weight of a vector is the number of nonzero entries in a vector.) Also, the folded hypercube $F_d = G(2,d,S_2)$ where $S_2 = S_1 \cup \{1\cdots11\}$, namely, the set S_1 augmented by the all-1 vector.

It was proven in [10] that Q_d and F_d are related in a rather interesting way. Given an arbitrary set A, $|A| = d$, where $A \subset S_2$, the graph $G(2,d,A)$ is isomorphic to Q_d. This property was used to obtain more efficient communication algorithms for F_d [9] as well as fault-tolerant constructions for cube-connected-cycles networks [4].

A natural question is whether it is possible to extend the hypercube by more than a single wildcard dimension. Namely, is it possible to find a set of offsets S, $|S| > d+1$, such that for an arbitrary $A \subset S$, with $|A| = d$, $G(2,d,A)$ is isomorphic to Q_d? In the following theorem we prove that the folded hypercube F_d is optimal in the sense that the hypercube can be extended only by a single wildcard dimension.

Theorem 1: Let S be a set of vectors of length d such that for every $A \subseteq S$, where $|A| = d$, the graph $G(2,d,A)$ is isomorphic to Q_d. Then $|S| \leq d+1$.

Proof: Assume, without loss of generality, that there exists a set $S = \{S_0, S_1, \cdots, S_{d+1}\}$ with the desired property, with $|S| = d+2$, and prove that it leads to a contradiction.

For every set A such that $G(2,d,A)$ is isomorphic to Q_d the set A has rank d. This follows from the fact that Q_d is a connected graph, hence, A spans the whole space $\{0,1\}^d$. Namely, any set of d vectors in S also has rank d. Hence, any two vectors in S are distinct and any vector in S can be specified as a linear combination of any other d vectors in S.

In particular, S_d and S_{d+1} are distinct and both can be specified as linear combinations of the vectors $\{S_0, S_1, \cdots, S_{d-1}\}$. Now, since they are distinct, one of the linear combinations consists of at most $d-1$ vectors. Assume, without loss of generality, that $S_d = \sum_{i=0}^{d-2} S_i$. Hence, the set $\{S_0, S_1, \cdots, S_{d-2}, S_d\}$ has rank smaller than d which is a contradiction. □

In the foregoing theorem we have proven that for hypercubes one can add at most one wildcard dimension. Surprisingly, we will show that when n is a prime and $n \geq d$, it is possible to add $n+1-d$ wildcard dimensions to a d-dimensional torus of the form $n \times n \times \cdots \times n$. This result, which is based on ideas from error-correcting codes, is developed in the remaining sections.

III. Wildcard Dimensions in Tori

In this section, we investigate the issue of wildcard dimensions in d-dimensional torus networks. In particular, we will characterize the sets of offsets that correspond to tori networks with wildcard dimensions. We denote a d-dimensional mesh and a d-dimensional torus of the form $n \times n \times \cdots \times n$ by M_n^d and T_n^d, respectively. First, we will define these graphs using the framework described in Definition 2.

Definition 5: S_d is defined to be the set of d vectors of length d which have Hamming weight 1. The torus graph T_n^d is defined to be isomorphic to $G(n, d, S_d)$. The mesh graph M_n^d is defined to be a torus without wraparound edges. More formally, each node X in M_n^d is connected to all nodes of the form $X \pm V$, where $V \in S_d$, (in this case, the addition is *not* performed modulo n) provided $(X \pm V) \in \{0, 1, \cdots, n-1\}^d$.

The set S_d in the foregoing definition is one possible choice of a set of offsets that represents a torus. The question is what other choices of S will result in a graph isomorphic to a torus. In the following theorem, we will give a necessary and sufficient condition on the set S such that $G(n, d, S)$ is isomorphic to the torus T_n^d. Through the rest of this section, we will assume that the parameter n is prime.

Theorem 2: Let d be a positive integer, let n be a prime number and let $S \subseteq \{0, 1, \cdots, n-1\}^d$. The graph $G(n, d, S)$ is isomorphic to the torus T_n^d if and only if S is a set of d linearly independent vectors over $GF(n)$.

Proof: Assume that $G(n, d, S)$ is isomorphic to T_n^d. We first prove that the set S spans the space $\{0, 1, \cdots, n-1\}^d$, i.e., it contains d linearly independent vectors. This follows from the fact that the graph T_n^d is connected, namely, there is a path from the all-0 node to every other arbitrary node. This path corresponds to a linear combination of vectors in S. Hence, the vectors in S span $\{0, 1, \cdots, n-1\}^d$. Now since the degree of T_n^d is $2d$ the set S must be of size d for $n > 2$. This completes the proof of the necessary part.

Now we assume that S is a set of d linearly independent vectors over $GF(n)$ and prove that $G(n, d, S)$ is isomorphic to T_n^d. We prove it by showing that there is a 1–1 mapping between $G(n, d, S)$ and T_n^d defined by a linear transformation, using S, that will be denoted by ϕ. Let $S = \{\underline{s}_0, \underline{s}_1, \cdots, \underline{s}_{d-1}\}$. Since S spans the whole space, every vector $\underline{v} \in \{0, 1, \cdots, n-1\}^d$ can be uniquely written as $\underline{v} = \phi(b_{d-1}, b_{d-2}, \cdots, b_0) \stackrel{\text{def}}{=} b_{d-1}\underline{s}_{d-1} + b_{d-2}\underline{s}_{d-2} + \cdots + b_0\underline{s}_0$, where computations are performed over $GF(n)$ and for all $0 \le i \le d-1$, $b_i \in \{0, 1, \cdots, n-1\}$. The definition of the 1–1 mapping between nodes of $G(n, d, S)$ and nodes of T_n^d depends on the above linear transformation ϕ. In particular, a node $\underline{b}$ in T_n^d corresponds to node $\phi(\underline{b})$ in $G(n, d, S)$. We now have to prove that this 1–1 mapping between the nodes results in a 1–1 mapping between the edges. Let $(\underline{a}, \underline{b})$ be an edge in T_n^d. There is a vector $\underline{c} \in S_d$ ($\underline{c}$ has Hamming weight 1) such that either $\underline{a} + \underline{c} = \underline{b}$ or $\underline{b} + \underline{c} = \underline{a}$. Also notice that $\phi(\underline{c}) \in S$. Hence, $(\phi(\underline{a}), \phi(\underline{b}))$ is an edge in $G(n, d, S)$. Note that the mapping of the edges is also a 1–1 mapping. □

Example: Let $S = \{(1, 2), (1, 4)\}$ then $G(5, 2, S)$ shown in Fig. 1(a) is isomorphic to a 5×5 torus, because $(1, 2)$ and $(1, 4)$ are linearly independent over $GF(5)$. Fig. 1(b) shows the new labeling as a torus.

It is clear that we can use Theorem 2 to add wildcard dimensions to tori. In particular, all that is required is a set of offsets S, where $|S| \ge d$, such that every d vectors in S are linearly independent. Then, applying Theorem 2, it follows that the graph that consists of d arbitrary dimensions in $G(n, d, S)$ is isomorphic to a torus. Thus, the key is having a construction defined as follows.

Defenition 6: Let n be a prime number and d a positive integer. We define $V(n, d)$ to be a set of vectors of length d over the finite field $GF(n)$, that has the property that any d vectors in the set are linearly independent over $GF(n)$. For convenience, we will represent $V(n, d)$ in a matrix form with the columns of the matrix being the vectors in $V(n, d)$.

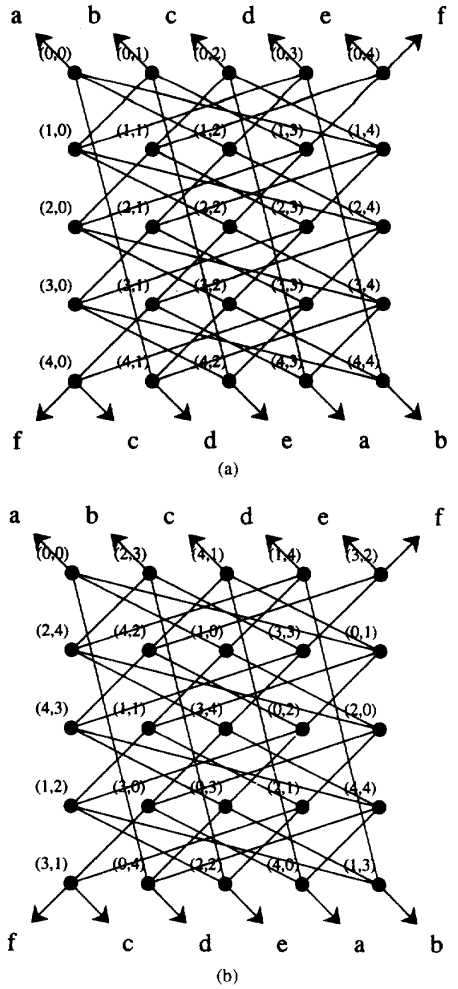

Fig. 1. (a) An example of $G(5, 2, \{(1, 2), (1, 4)\})$, on the top. (b) A labeling of the graph as a 5×5 torus, on the bottom.

For example,

$$V(5,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 0 \\ 1 & 4 & 4 & 1 & 0 & 1 \end{pmatrix}.$$

The graph $G(5, 3, V(5, 3))$ is a three-dimensional torus with 3 wildcard dimensions. In the next section we will present a general method for constructing $V(n, d)$, which is based on coding theory.

IV. Linearly Independent Vectors Over Finite Fields

Our method for constructing fault-tolerant tori is based on the existence of a set of vectors of length d over the finite field $GF(n)$, that has the property that any d vectors in the set are linearly independent over $GF(n)$. We denote this set by $V(n, d)$. In particular, constructing a fault-tolerant d-dimensional torus is based

upon the construction of $V(n,d)$. In this section we will present a construction of $V(n,d)$ of $n+1$ vectors of length d over $GF(n)$, for any n which is prime and for any $d \leq n$. Our construction follows from known constructions in error-correcting block codes. For more details on this subject refer to [12].

The main issue in the theory of error correcting codes is to construct a large set of vectors (code) with the property that the Hamming distance between any two vectors in the code is larger than a predefined parameter d' (the minimum distance). Linear block codes are codes that have the property that the set of codewords forms a *vector space* over $GF(n)$. These codes can be described by a matrix known as the *parity check* matrix. In particular, a code of length n' (the length of vector in the code) and dimension k' (consists of $2^{k'}$ codewords) can be described by an $(n'-k') \times n'$ parity check matrix H. A vector V of length n' is in the code if and only if $V \cdot H^T = \underline{0}$, where $\underline{0}$ denotes an all-0 vector of length $(n'-k')$. Let H be a parity check matrix of a linear code over $GF(n)$ of length n' dimension k' and minimum distance d'. The following facts are known in coding theory.

Fact 1) The Singleton bound: $n'-k'+1 \geq d'$. Codes for which $n'-k'+1 = d'$ are called MDS (Maximum Distance Separable) codes.

Fact 2) Any $d'-1$ columns in H are linearly independent over $GF(n)$.

The following theorem follows from the two previous facts.

Theorem 3: The columns of the parity check matrix of an MDS code over $GF(n)$ form a $V(n,d)$ set with $d = n'-k'$ and $n = n'-1$.

(Note that we let $d = d'-1$, $n = n'-1$ and $n'-k' = d$ in relating coding theory notation to the matrix $V(n,d)$.) MDS codes that are based on codes known as extended Reed-Solomon codes are known for all $GF(n)$ and $n' = n+1$. This construction exists for arbitrary fields (even in the case where n is a power of a prime). Here we present one way to construct a parity check matrix of the form $d \times (n+1)$ for a Reed-Solomon code by extending the *Vandermonde matrix*.

Construction 1:

$$V(n,d) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & 2 & 3 & \cdots & (n-1) & 0 & 0 \\ 1^2 & 2^2 & 3^2 & \cdots & (n-1)^2 & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 1^{d-2} & 2^{d-2} & 3^{d-2} & \cdots & (n-1)^{d-2} & 0 & 0 \\ 1^{d-1} & 2^{d-1} & 3^{d-1} & \cdots & (n-1)^{d-1} & 0 & 1 \end{pmatrix}.$$

Since $V(n,d)$ as defined above is a parity check matrix of an MDS code we get the following.

Corollary 1: Let n be a positive prime and let d be an integer with $d \geq 2$. Any d column vectors in $V(n,d)$ of Construction 1, are linearly independent over $GF(n)$.

$V(n,d)$ in Construction 1 has $n+1$ vectors. We note that finding the longest MDS code for a given n is a well known open problem and a well known conjecture is that $n+3$ is an upper bound.

As an example for $d = 2$ and an arbitrary prime number n,

$$V(n,2) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & 2 & 3 & \cdots & (n-1) & 0 & 1 \end{pmatrix}.$$

It can be easily verified that any two column vectors in $V(n,2)$ are linearly independent over $GF(n)$. This set can be used to obtain a construction of an $n \times n$ 2-dimensional torus with $n-1$ wildcard dimensions.

As another example consider $n = 7$ and $d = 3$, we get

$$V(7,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 & 0 & 0 \\ 1 & 4 & 2 & 2 & 4 & 1 & 0 & 1 \end{pmatrix}.$$

Any 3 vectors in $V(7,3)$ are linearly independent over $GF(7)$. $V(7,3)$ can be used to obtain a construction of a $7 \times 7 \times 7$ 3-dimensional torus with 5 wildcard dimensions.

We conclude this section with the following corollary.

Corollary 2: Given the set of columns $V(n,d)$ in Construction 1, the graph $G(n,d,V(n,d))$ is a d-dimensional torus of form $n \times n \times \cdots \times n$ with $n+1-d$ wildcard dimensions.

V. Edge Fault-Tolerant Meshes and Tori

In this section, we will give constructions of edge-fault-tolerant d-dimensional meshes and tori. The constructions are based on $V(n,d)$ that was described in Construction 1. We first define more precisely the notion of edge fault tolerance.

Definition 7: Let k be a nonnegative integer and let $G = (V,E)$ be a graph. We say that the graph $G' = (V,E')$ is (k,G)-edge-tolerant if the subgraph of G' induced by removing any k edges from G' contains G as a subgraph.

Note that both G and G' have the same node set. Next, we present a simple technical lemma.

Lemma 1: Let d and s be positive integers with $d < s$. For any assignment of $2s-2d+1$ balls to s buckets, there exist at least d buckets, each of which contains at most one ball.

Proof: Assume, for the sake of contradiction, that there are only $d-1$ buckets with at most 1 ball in each. Then there are $s-d+1$ buckets that have at least two balls. Namely, there are at least $2s-2d+2$ balls, which contradicts the number of balls given. □

Note that $2s-2d+1$ is the maximum number of balls one can distribute over s buckets to guarantee that there exist at least d buckets each with at most one ball in it. We use this lemma to utilize the wildcard dimensions as follows,

Theorem 4: Let d be an integer where $d \geq 2$, let n be a prime number where $n+1 \geq d$, and let $S \subseteq V(n,d)$ where $d \leq |S| \leq n+1$. Let $|S| = s$. Then the graph $G(n,d,S)$ is $(s-d, T_n^d)$-edge-tolerant and $(2s-2d+1, M_n^d)$-edge-tolerant.

Proof: By Corollary 2 there are $s-d$ wildcard dimensions, which implies that there are s (wildcard and standard) dimensions. Thus, if there are at most $s-d$ edge faults, at least d dimensions contain no faulty edges. These d nonfaulty dimensions form the torus T_n^d.

For the case where the target graph is M_n^d, consider the distribution of $2s-2d+1$ edge faults over the s dimensions. According to Lemma 1, there must exist d dimensions which contain at most one faulty edge each. The graph defined by these d dimensions is a torus T_n^d which contains at most 1 faulty edge in each dimension. The healthy edges in this graph contain the mesh M_n^d, because it is possible to rotate each dimension, independently, so as to place the single faulty edge in that dimension (if it exists) in the position of a wraparound edge in the torus, which is not required in the mesh. The proof for that is by induction on the number of dimensions and rotation is performed on a single dimension at a time. □

Note that the fault-tolerant graph $G(n,d,S)$ has degree $2s$, where $d \leq s \leq n+1$, and can tolerant $2s-2d+1$ edge faults such that the remaining graph still contains M_n^d as a subgraph.

VI. Connectivity Properties

Throughout this section, let $V(n,d)$ be the $d \times (n+1)$ matrix defined in Construction 1. In this section, we will show that the

connectivity of the graph $G(n, d, S)$ is the same as its degree, where $S \subset V(n, d)$ and $d \leq |S| \leq n$.

A. Preliminaries

The vertex set of a graph W is denoted by $V(W)$. A graph W is *vertex-transitive* if for any two vertices u and v of W, there is an automorphism σ of W satisfying $\sigma(u) = v$. The connectivity of a graph W, denoted $conn(W)$, is the smallest number j such that there exists some set of j vertices whose deletion results either in a disconnected subgraph or a single vertex. A graph W is *regular* if all vertices have the same degree. The degree of a regular graph W, denoted $deg(W)$, is the degree of any of its vertices. A graph W is *optimally connected* if it is regular and $deg(W) = conn(W)$.

A *cutset* in a graph W is a set of vertices whose deletion leaves a disconnected graph. Let $W \backslash C$ denote the subgraph remaining after the vertices of a cutset C have been deleted. Now let C be a minimum cutset of W, that is, $|C| = conn(W)$. The connected components of $W \backslash C$ are called *parts* of W. All the parts of W of minimum cardinality are called *atoms*. (In other words, for each minimum cutset of W, the parts are listed and the lists are then combined. The *atoms* are all of the parts of the same minimum cardinality occurring in the combined list.) Let $a(W)$ denote the cardinality of an atom of a vertex-transitive graph W. Two useful lemmas of [13] (See also [2]) are now stated.

Lemma 2 [13]: Let W be a connected and vertex-transitive graph with $conn(W) < deg(W)$. Then $|V(W)| = ca(W)$ for some integer $c \geq 2$.

Lemma 3 [13]: Let W be a connected and vertex-transitive graph with $conn(W) < deg(W)$. Then $conn(W) = ca(W)$ for some integer $c \geq 2$.

B. The Connectivity of Tori with Wildcard Dimensions

We begin with a general theorem about the connectivity of vertex-transitive graphs.

Theorem 5: Let G be a connected and vertex-transitive graph of N nodes, where $N = \prod_{i=1}^{k} p_i^{r_i}$, $p_1 < p_2 < \cdots < p_k$, p_i's are prime, r_i's are positive integers, and $deg(G) \leq 2p_1$. Then G is optimally connected.

Proof: Assume G is not optimally connected, i.e., $conn(G) < deg(G) \leq 2p_1$. From Lemma 3, $conn(G) \geq 2a(G)$. Thus, $a(G) < p_1$, which implies $a(G) = 1$ because p_1 is the smallest prime that divides N and $a(G)$ divides N (by Lemma 2). But, $a(G) = 1$ implies that G is optimally connected, which is a contradiction. □

Corollary 3: Let n be prime, let d be any positive integer with $d \geq 2$, and let $S \subset V(n, d)$ with $d \leq |S| \leq n$. Then the graph $G(n, d, S)$ is optimally connected.

Proof: Clearly, $G(n, d, S)$ is connected (as $|S| \geq d$) and vertex-transitive and $deg(G(n, d, S)) \leq 2n$. Applying Theorem 5 completes the proof. □

Corollary 4: Let G be a graph which is connected, vertex-transitive and has N vertices where N is prime. Then G is optimally connected.

Note that the parameter n in Corollary 3 is a prime number. This restriction can be removed while still guaranteeing that the graph $G(n, d, S)$ is optimally connected. We now present this general theorem. The proof is omitted here and can be found in [5].

Theorem 6: Let d be any integer where $d \geq 2$, let n be an integer where $n \geq 3$, and let $S \subset V(n, d)$ where $d \leq |S| \leq n$. Then the graph $G(n, d, S)$ is optimally connected.

We now give a few remarks. Note the restriction that $|S| \leq n$ in the previous theorem. This is because if $|S| = n + 1$, i.e., $S = V(n, d)$, the graph $G(n, d, S)$ may not be optimally connected. For instance, the graph $G(n, 2, V(n, d))$ where $n \geq 4$, has degree $2n + 2$ and connectivity $2n$. The latter can be shown by removing any two nonadjacent rows (of $2n$ nodes totally) to disconnect the graph, where rows are identified by the first element of the vectors in $V(n, d)$.

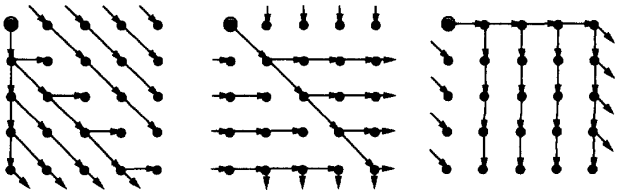

Fig. 2 Three edge-disjoint spanning trees on a 5×5 torus with a wildcard dimension.

Although we only give an existence proof for Theorem 6, we actually have a construction of $2s$ node-disjoint paths between any two vertices in $G(n, 2, S)$, where S is the first s vectors of $V(n, d)$ and $2 \leq s \leq n$. The basic idea is to define s parallel paths from the s "forward" neighbors of the source node to the s "backward" neighbors of the destination node, such that no two paths touch the same column. Then, another set of s parallel paths from s "backward" neighbors of the source node to the s "forward" neighbors of the destination node can be similarly defined. The construction has separate cases depending on whether $s \leq \lceil n/2 \rceil$ and depending on the difference between the column indices of the source and destination nodes.

We also have a construction of $2s$ (the maximum possible) edge-disjoint spanning trees in $G(n, 2, S)$ for any $S \subseteq V(n, d)$ where n is prime. Note that there are known constructions of four edge-disjoint spanning trees on an $n \times n$ torus [3], [7]. Since any two vectors in S form the whole edge set of an $n \times n$ torus, one can construct four edge-disjoint spanning trees using only two vectors. When S contains an even number of vectors, four such edge-disjoint spanning trees are constructed for each pair of vectors. When S contains an odd number of vectors, in addition to creating four trees for each pair of vectors, six trees are constructed for a set of three vectors. Without loss of generality, we can assume the three vectors are $(0, 1)$, $(1, 0)$ and $(1, 1)$ since n is prime. Fig. 2 gives an example of three edge-disjoint spanning trees on a 5×5 torus with a wildcard dimension using only "forward" edges. The other three edge-disjoint spanning trees can be similarly derived using only "backward" edges.

REFERENCES

[1] G. B. Adams and H. J. Siegel, " The extra stage cube: A fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.*, vol. 31, no. 5, pp. 443–454, May 1982.

[2] B. Alspach, "Cayley graphs with optimal fault tolerance," *IEEE Trans. Comput.*, vol. 41, no. 10, pp. 1337–1339, Oct. 1992.

[3] R. S. Bajwa and S. R. Seidel, " Communication algorithms for tori and grids," Tech. Rep. TR 89-04, Dep. of Comput. Sci., Michigan Technological Univ., Nov. 1989.

[4] J. Bruck, R. Cypher, and C.-T. Ho, "On the construction of fault-tolerant cube connected cycles networks," in *Proc. Int. Conf. Parallel Processing*, vol. 1, pp. 692–693, Penn State Univ., 1991.

[5] ——, "Wildcard dimensions, coding theory and fault-tolerant meshes and hypercubes," in *Proc. 23rd Int. Symp. Fault-Tolerant Computing*, June 1993, pp. 260–267.

[6] ——, "Fault-tolerant meshes and hypercubes with minimal numbers of spares," *IEEE Trans. Comput.*, vol. 42, no. 9, pp. 1089–1104, Sept. 1993.

[7] P. Fraigniaud and E. Lazard, "Methods and problems of communication in usual networks," Tech. Rep. 91-33, IMAG, Ecole Normale Supérieure de Lyon, France, Oct. 1991.

[8] A. Ghafoor, T. R. Bashkow, and I. Ghafoor, "Bisectional fault-tolerant communication architecture for supercomputer systems," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1425–1446, Oct. 1989.

[9] C.-T. Ho, " Full bandwidth communications on folded hypercubes," in *Proc. Int. Conf. Parallel Processing*, vol. I, Penn State, 1989, pp. 276–280.
[10] ——, "An observation on the bisectional interconnection networks," *IEEE Trans. Comput.*, vol. 41, no. 7, pp. 873–877, July 1992.
[11] S. Latifi and A. El-Amawy, "on folded hypercubes," in *Proc. Int. Conf. Parallel Processing*, vol. I, Penn State, 1989, pp. 180–187.
[12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1977.
[13] M. E. Watkins, "Connectivity of transitive graphs," *J. Combinat. Theory*, vol. 8, 23–29, 1970.

Implementation of Four Common Functions on an LNS Co-Processor

Debasish Das, Krishnendu Mukhopadhyaya, and Bhabani P. Sinha

***Abstract*— We propose a scheme for evaluating four commonly used functions namely, 1) inverse trigonometric functions, 2) trigonometric functions, 3) the exponential function, and 4) the logarithmic function with the help of a logarithmic number system (LNS) processor. A novel idea of *series folding* has been introduced for computing the above functions, expressed in the form of infinite series. We also show that with a suitable choice of the radix for the LNS we can evaluate exponential and logarithmic functions without using any extra hardware.**

***Index Terms*— Logarithmic number system, series folding, inverse trigonometric functions, arctangent function, trigonometric functions, exponential function, logarithmic function.**

I. Introduction

Many real time applications in the areas of signal processing, process control, etc., require very fast evaluation of a large number of mathematical functions. Contemporary Arithmetic Logic Units (ALU) of a general purpose computer may often find it difficult to meet this requirement of massive real-time computations. One of the main reasons for not achieving such high rate of arithmetic computations is the relative inefficiency associated with floating-point operations.

To overcome this difficulty, the idea of the Logarithmic Number System (LNS) for the representation and manipulation of numbers was proposed [1]. LNS offers several advantages compared to the floating-point representation. A fast arithmetic unit was developed [2] for obtaining very high computational data rates. Such processors are very efficient in performing operations like multiplication, division, squaring and square-rooting, but slow for operations like addition and subtraction. Addition and subtraction operations in LNS need to perform a table look-up and this table is to be stored as part of the LNS processor. A technique for reducing the size of the look-up tables has been proposed in [3].

One of the problems associated with the design of the LNS processor was the conversion of a number from the binary floating-point system to LNS and vice-versa. In the design given in [2], the authors proposed the use of the look-up tables to perform these operations. However, methods for generating the logarithm of a number using PLAs have evolved, as given in [4] and have solved this problem too. To reduce the number of conversions from floating-point to LNS, it is reasonable to think about an LNS co-processor which can perform all the arithmetic operations performed by a commercially available numeric co-processor and with the same precision. The LNS co-processor as designed in [2] was of very limited application. The processor could perform only six basic arithmetic operations 1) addition, 2) subtraction, 3) multiplication, 4) division, 5) squaring, and 6) square-rooting. It is difficult for a co-processor of such limited capability to satisfy the demanding needs of computing other mathematical functions involved in many real-time applications.

In this brief contribution, we propose the idea of implementing four other general purpose functions in such a co-processor, by using suitable algorithms and a little amount of extra hardware. The four functions that we have chosen for implementation are very fundamental and frequently needed. They are 1) inverse trigonometric functions, 2) trigonometric functions, 3) the exponential function, and 4) the logarithmic function. Each of these functions is first expressed as a power series and then evaluated by the LNS co-processor. In evaluating such infinite power series of a variable x, the primary problem is that the number of terms which are to be evaluated and then summed up, depends on the precision and the value of the argument x. If x is a small fraction close to zero, the number of terms to be summed up will be small; otherwise it will be large. The proposed technique is centered around developing an algorithm based on an idea of *series folding*, such that by suitably transforming the argument, we can compute the value of the function, by evaluating a small number of terms for a given precision. Since the number of multiplications/divisions to be performed in a power series evaluation soon overrides the number of additions/subtractions, and also because the multiplication/division dominated computations can be executed at a faster rate on an LNS co-processor, the proposed technique helps to compute these functions very quickly with a reasonable accuracy.

In the final part of our work, we will show that a suitable choice of the radix r of the LNS, may eliminate the requirement of any extra computation other than table look-up for the exponential and logarithmic functions, again without affecting the precision.

II. Logarithmic Number System and Associated Arithmetic

In LNS, a number x is represented in signed magnitude form, i.e., as a pair (S, e), where $x = (-1)^S(r)^e$, S being the sign bit (which is either 0 or 1 according to the sign of x) and e being the signed exponent of the radix r. The exponent e is expressed in fixed point binary mode with say, I bits for the integer part and F bits for the fractional part and one bit for the sign of the exponent, i.e., with a total of $(I + F + 1)$ bits. If the radix is considered to be 2, then the smallest number that can be represented using the scheme is 2^{-N}, where $N = (2^I - 1) + (1 - 2^{-F}) = (2^I - 2^{-F})$. The ratio between two consecutive numbers is equal to $r^{2^{-F}}$, and the corresponding precision ϵ is roughly $(\ln r)2^{-F}$. Typically, if $I = 5$, $F = 26$, and $r = 2$, we can have a precision of 26 bits in radix 2. However, for the purpose of comparison with the precision of floating-point representation, ϵ will be assumed as $2^{-23}(\approx 10^{-7})$.

Arithmetic operations involving manipulation of the exponent part only can very easily be performed using such a representation. Assume that two numbers A and B are represented in the LNS format as the tuples $(S(A), e(A))$ and $(S(B), e(B))$ respectively where $A = (-1)^{S(A)}r^{e(A)}$ and $B = (-1)^{S(B)}r^{e(B)}$. Now, if $C = A * B$, and C is represented in the LNS form as the pair $(S(C), e(C))$, then $e(C) = e(A) + e(B)$ and $S(C) = S(A) \oplus S(B)$. For $C = \frac{A}{B}$,

Manuscript received June 22, 1993; revised November 1, 1993.
D. Das and B. P. Sinha are with the Electronics Unit, Indian Statistical Institute, Calcutta 700 035, India.
K. Mukhopadhyaya is with the Jadavpur University, Calcutta 700 032, India.
IEEE Log Number 9404682.