Enhanced Fibonacci Cubes

HAIFENG QIAN AND JIE WU

Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL, 33431, USA

Email: jie@sunrise.cse.fau.edu

We propose the enhanced Fibonacci cube (*EFC*) structure for parallel systems. It is defined based on the sequence $F_n = 2F_{n-2} + 2F_{n-4}$. We show that the enhanced Fibonacci cube contains the Fibonacci cube (*FC*) as a subgraph and maintains virtually all the desirable properties of the Fibonacci cube. In addition, it is a Hamiltonian graph. We can embed complete binary trees into enhanced Fibonacci cubes with dilation one and with a relatively small expansion. We also propose a series of enhanced Fibonacci cubes $EFC^{(k)}$, where k is a series number. Each $EFC^{(k)}$ contains an *FC* of the same order as a subcube. Moreover, each $EFC^{(k)}$ in the series contains any other cube that precedes it as subcubes and the last one in the series is a hypercube of the corresponding order. This series of $EFC^{(k)}$s provides us with more options for selecting cubes with various sizes. Because *EFC* is a subgraph of the hypercube, it may find applications in fault-tolerant computing for degraded hypercube computer systems. As an application of *EFC*, we show that the parallel prefix sum computation can be efficiently implemented on enhanced Fibonacci cubes.

Received November 2, 1994; revised March 17, 1996

1. INTRODUCTION

Parallel processing seeks to improve the speed with which a computation can be done by breaking it into subcomputation and executing them concurrently on different processors. It is important the way these processors are connected. Among many interconnection networks, *hypercube* [1] (*HC*) is one of the widely used networks. The hypercube provides a rich interconnection structure which permits many other topologies to be efficiently emulated. Numerous research projects have been undertaken related to hypercube design aspects and hypercube applications [2]. These have resulted in several research prototypes and commercial products [3, 4, 5].

Unfortunately, the number of nodes 2^n in an n-dimensional hypercube grows rapidly as n increases. This considerably limits the choice of the number of nodes in the network. For example, when the dimension is increased by one, the total number of nodes is doubled and the total number of edges is more than doubled. Also, the larger the size of a system is, the higher the probability of some processors or links failure. The hypercube may not work properly even if only one fault occurs. Effort has been made to the study of subcubes (or subgraphs) of hypercube to circumvent the above drawbacks. These include several incomplete hypercube-like architectures [6, 7]. The *Fibonacci cube* (*FC*) [8] is a special subcube of a hypercube based on the Fibonacci number $F_n = F_{n-1} + F_{n-2}$. It has been shown that the Fibonacci cube can efficiently simulate many hypercube algorithms. The Fibonacci cube uses about one-fifth fewer links than the comparable hypercube and its size does not increase as fast as the hypercube. A Fibonacci cube can also be viewed as a hypercube with faulty nodes. With Fibonacci cubes, more choices of network size are available.

The enhanced Fibonacci cube proposed in this paper provides even more choices of network size to the family of cube-based structures. The *enhanced Fibonacci cube* (*EFC*) is defined based on the sequence $F_n = 2F_{n-2} + 2F_{n-4}$ and it maintains virtually all of the desirable properties of the Fibonacci cube. What is more, it has some desirable network topological properties such as Hamiltonian property that the Fibonacci cube does not have. We also propose a series of enhanced Fibonacci cubes $EFC^{(k)}$s, where k is a series number, each of which contains an *FC* of the same order as a subcube. An $EFC^{(k)}$ contains other *EFC*s as subcubes that precede it in the series. A hypercube of the corresponding order is the last one in the series. Briefly, we show that: (1) all algorithms of *FC*s can be run on *EFC* of the same order; (2) all *EFC*s are Hamiltonian while more than two-thirds of *FC*s are not; (3) *EFC* can embed the tree network with dilation 1, while no dilation 1 embedding of a tree into an *FC* with comparable expansion is known; and (4) hypercubes and Fibonacci cubes are special cases of enhanced Fibonacci cubes $EFC^{(k)}$.

There are two major applications of the *EFC* structure and these are also the objectives of our study:

- *EFC* can be used as a special type of incomplete hypercubes. Normally, a faulty hypercube can be reconfigured into a smaller sized system, which can be either a subcube or a subgraph (which is not a cube structure). *EFC* provides more options of incomplete hypercubes to which a faulty hypercube can be reconfigured. Therefore, *EFC* may find applications in fault-tolerant computing for degraded hypercube computer systems.
- *EFC* can be used as a new cube-based architecture. We show that the number of nodes in the series

$EFC^{(k)}$s are all different. Therefore, our study provides more options for selecting cubes with various sizes.

The rest of the paper is organized as follows. In Section 2 we give the definition of *EFC*, discuss its properties including Hamiltonian property and present results of its diameter, node degree, and node and link complexities of *EFC*. In Section 3 the embedding of trees and hypercubes into *EFC*s are studied. In Section 4 we define the enhanced Fibonacci tree, and present an algorithm for parallel prefix sum computation on *EFC*. In Section 5 we propose a series of enhanced Fibonacci cubes $EFC^{(k)}$, where k is the series number, and study the relationships among $EFC^{(k)}$, *EFC*, *FC* and *HC*. This paper is concluded in Section 6.

2. PRELIMINARIES

We first define Fibonacci cubes and enhanced Fibonacci cubes. In the following definitions, $||$ denotes the concatenation operation, e.g. $10||\{0,1\} = \{100,101\}$ and $00||\{\} = 00$. Sometimes we may leave out "$||$" for simplicity.

DEFINITION 1. [8] Let $FC_n = (V_n, E_n)$ denote the Fibonacci cube of the order n, then $V_n = 0||V_{n-1} \cup 10||V_{n-2}$, where $V_3 = \{1,0\}$ and $V_4 = \{01,00,10\}$. Two nodes in an FC_n are connected by an edge in E_n if and only if their labels differ in exactly one bit position.

DEFINITION 2. Let $EFC_n = (V_n, E_n)$ denote the enhanced Fibonacci cube of the order n, then $V_n = 00||V_{n-2} \cup 10||V_{n-2} \cup 0100||V_{n-4} \cup 0101||V_{n-4}$.[1] Two nodes in an EFC_n are connected by an edge in E_n if and only if their labels differ in exactly one bit position. As initial conditions for recursion, $V_3 = \{1,0\}$, $V_4 = \{01,00,10\}$, $V_5 = \{001,101,100,000,010\}$ and $V_6 = \{0001,0101,0100,0000,0010,1010,1000,1001\}$.

Note that in an FC_n or EFC_n, nodes are ordered in a Gray code sequence, and $n-2$ bits are used in a node address. Figure 1 shows the *EFC*s with the order of 3, 4, 5, 6, 7 and 8.

THEOREM 1. *An FC_n is a subgraph of an EFC_n.*

Proof. Because in both *EFC*s and *FC*s, nodes are connected if their labels differ in exactly one bit position, it suffices to show that $V'_n \subseteq V_n$, where V'_n and V_n are node sets of FC_n and EFC_n, respectively. Clearly $V'_n = V_n$ for $n \leq 6$. Assume $V'_n \subseteq V_n$ for all $n < k$ $(k > 6)$. When $n = k$, based on the definition of $V'_n = 0||V'_{k-1} \cup 10||V'_{k-2}$, we have,

$$\begin{aligned}
V'_k &= 0V'_{k-1} \cup 10V'_{k-2} \\
&= 0(0V'_{k-2} \cup 10V'_{k-3}) \cup 10V'_{k-2} \\
&= 00V'_{k-2} \cup 10V'_{k-2} \cup 010(0V'_{k-4} \cup 10V'_{k-5}) \\
&= 00V'_{k-2} \cup 10V'_{k-2} \cup 0100V'_{k-4} \cup 01010V'_{k-5} \\
&\subset 00V'_{k-2} \cup 10V'_{k-2} \cup 0100V'_{k-4} \cup 0101V'_{k-4} \\
&\subseteq 00V_{k-2} \cup 10V_{k-2} \cup 0100V_{k-4} \cup 0101V_{k-4} \\
&= V_k
\end{aligned}$$
□

Cong, Zheng and Sharma [9] showed that less than one-third of Fibonacci cubes are Hamiltonian. The following result shows that all the *EFC*s are Hamiltonian.

PROPERTY 2.1. *For any $n \geq 6$, an EFC_n is a Hamiltonian graph.*

Proof. We prove this theorem by induction on n. We show that, for each EFC_n, $n \geq 6$, we can construct a Hamiltonian cycle of type $\{000^*1, 010^*1, G\}$, where G is a Gray code sequence whose first and last nodes are adjacent to 010^*1 and 000^*1, respectively, and 0^* is a sequence of 0. For $n = 6$, 8, there exist Hamiltonian cycles of the specified type in EFC_6 and EFC_8.

$$\begin{aligned}
n &= 6, \{0001, 0101, 0100, 0000, 0010, 1010, 1000, 1001\} \\
n &= 8, \{000001, 010001, 010000, 010010, 010110, 010100, \\
&\quad 010101, 000101, 100101, 100001, 101001, 101000, \\
&\quad 101010, 100010, 100000, 100100, 000100, 000000, \\
&\quad 000010, 001010, 001000, 001001\}
\end{aligned}$$

Assume that this claim holds for all even EFC_{2n}s, such that $n < k$ for $k \geq 3$. Based on induction, we have a Hamiltonian cycle $\{000^*1, 010^*1, G_1\}$ for an EFC_{2k-2} and a Hamiltonian cycle $\{0^*1, G_2\}$ for an EFC_{2k-4} (note that the latter expression is a special case of the former one). Let G^{-1} denote the reversed sequence of G, we construct a Hamiltonian cycle for EFC_{2k} as follows:

$$\begin{aligned}
&\{00||000^*1, 0100||0^*1, 0100||G_2, 0101||G_2^{-1}, 0101||0^*1, \\
&\quad 00||010^*1, 00||G_1, 10||G_1^{-1}, 10||010^*1, 10||000^*1\}
\end{aligned}$$

It is easy to verify that an EFC_7 and an EFC_9 also have Hamiltonian cycles of the specified type, then the above proof also applies to odd EFC_{2n-1}s. □

The Hamiltonian property ensures that the ring network can be emulated by the enhanced Fibonacci cube of the same size efficiently.

By the recurrence $V_n = 00||V_{n-2} \cup 10||V_{n-2} \cup 0100||V_{n-4} \cup 0101||V_{n-4}$, an EFC_n can be decomposed into $00EFC_{n-2}$, $10EFC_{n-2}$, $0100EFC_{n-4}$, $0101EFC_{n-4}$ (Figure 2). $00EFC_{n-2}$ can be further divided into $0000EFC_{n-4}$, $0010EFC_{n-4}$, $000100EFC_{n-6}$, 000101 EFC_{n-6}. $10EFC_{n-2}$ has one-to-one node connections with $00EFC_{n-2}$, i.e. each node is connected by a link to its

[1] There are other coding methods which also ensure the derived Fibonacci cube to be Hamiltonian; they are: $V_n = 00||V_{n-2} \cup 01||V_{n-2} \cup 1000||V_{n-4} \cup 1001||V_{n-4}$ and $V_n = 00||V_{n-2} \cup 10||V_{n-2} \cup 0100||V_{n-4} \cup 0110||V_{n-4}$. The selection of this coding is purely for the comparison with FCs.

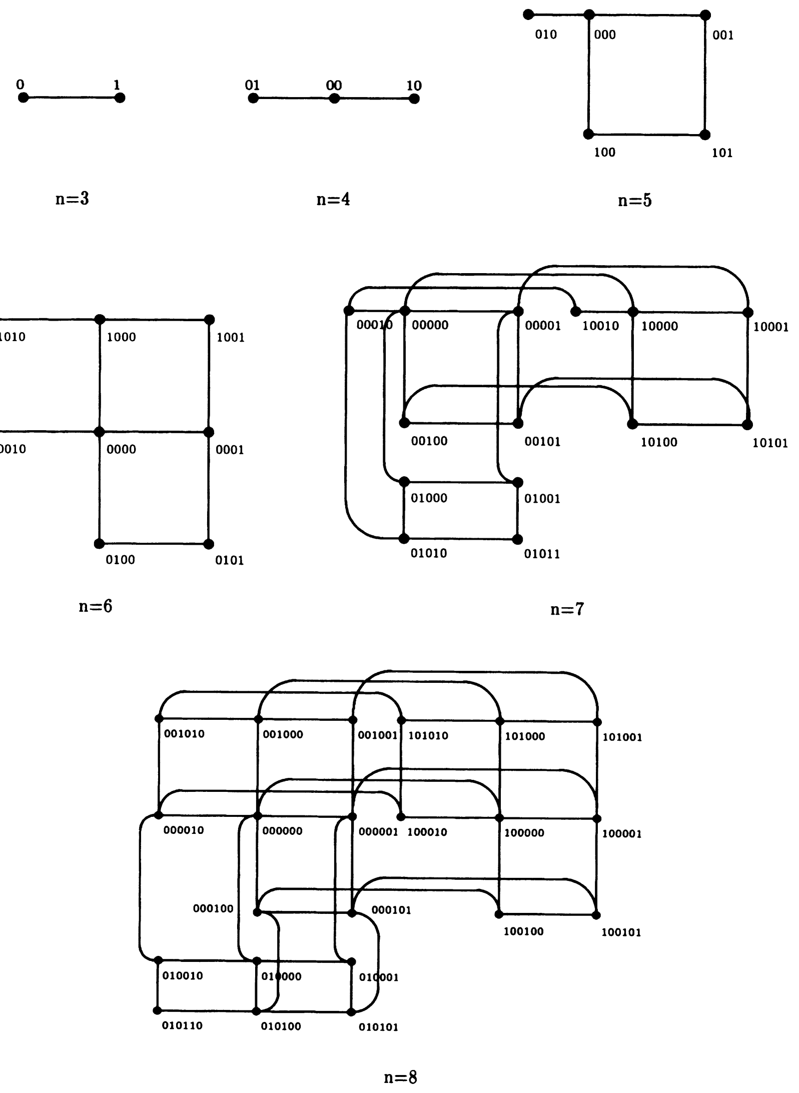

FIGURE 1. The EFC_ns for $3 \leq n \leq 8$.

correspondent node, but it has no connections with $0100EFC_{n-4}$ and $0101EFC_{n-4}$. $0100EFC_{n-4}$ has one-to-one node connections with both $0000EFC_{n-4}$ and $0101EFC_{n-4}$. The coincidence here is that the way $0000EFC_{n-4}$ is connected to $000100EFC_{n-6}$ and $000101EFC_{n-6}$ is isomorphic to the way $0101EFC_{n-4}$ is to those two EFC_{n-6}s. These observations are illustrated in Figure 2 and concluded in Lemma 1 and Property 2.2.

In Figure 2, each expression associated with a line indicates the number of links (this will be used in the

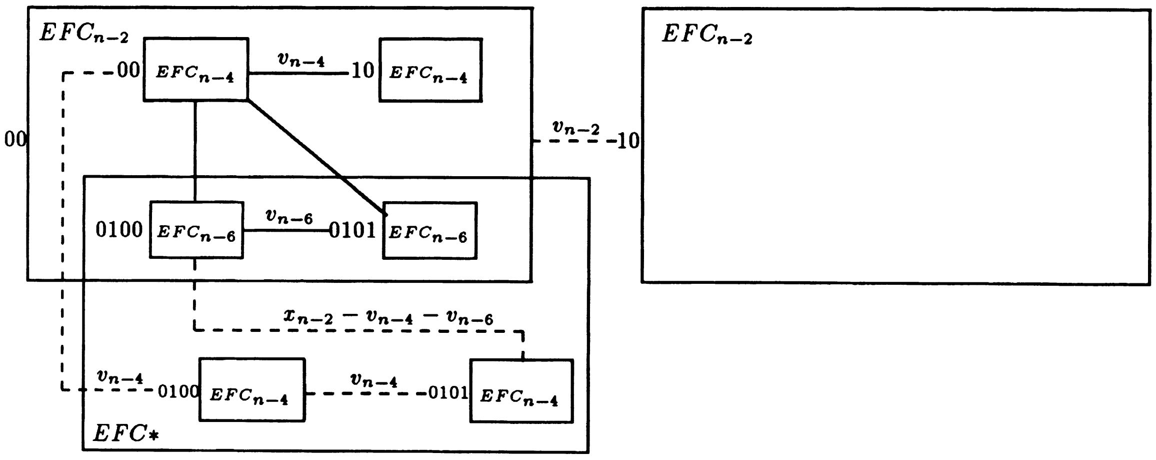

FIGURE 2. The decomposition of an EFC_n.

proof of Property 2.2). Lemma 1 is helpful to find the number of links connecting those four disjoint *EFC*s of an EFC_n, which is the key to obtain the recursive expression of the number of links in the EFC_n.

Lemma 1. *In an EFC_n (see Figure 2), the subgraph $00EFC_{n-2}$ is isomorphic to another subgraph EFC^* induced by the node set $V^* = 000100V_{n-6} \cup 000101V_{n-6} \cup 0100V_{n-4} \cup 0101V_{n-4}$.*

Proof. It suffices to show that the connections between $0000EFC_{n-4}$ and $000100EFC_{n-6}$ and $000101EFC_{n-6}$ are isomorphic to those connections between $0101EFC_{n-4}$ and $000100EFC_{n-6}$ and $000101EFC_{n-6}$. We map node $0000a$ in $0000EFC_{n-4}$ to $0101a$ in $0101EFC_{n-4}$ ($a \in V_{n-4}$), then $0000a$ is connected to node $000100b$ in $000100EFC_{n-6}$ ($b \in V_{n-6}$, $a = 00b$) if and only if $0101a$ is connected to this node $000100b$, thus the connections between $0000EFC_{n-4}$ and $000100EFC_{n-6}$ are isomorphic to those between $0101EFC_{n-4}$ and $000100EFC_{n-6}$. Similarly, it is also true for these two EFC_{n-4}s relating to $000101EFC_{n-6}$. □

Let e_n and v_n denote the number of nodes and links in an EFC_n, respectively.

Property 2.2. *An EFC_n can be decomposed into two EFC_{n-2}s and EFC_{n-4}; these four subgraphs are disjoint and are connected by $\frac{1}{3}[2v_n + v_{n-4} + \frac{1+(-1)^n}{2}]$ links.*

Proof. The first claim is obvious. Let x_n denote the number of links connecting these four *EFC*s. By Lemma 1, $x_n = v_{n-2} + v_{n-4} + v_{n-4} + (x_{n-2} - v_{n-4} - v_{n-6})$ (see Figure 2). Then $x_n = x_{n-2} + v_{n-2} + v_{n-4} - v_{n-6}$. Based on the initial conditions, we can solve for x_n:

$$x_n = \begin{cases} v_{n-4} + \sum_{k=2}^{(n-2)/2} v_{2k} + 2 & n \text{ is even} \\ v_{n-4} + \sum_{k=2}^{(n-1)/2} v_{2k-1} + 1 & n \text{ is odd} \end{cases}$$

By the recursion $v_n = 2v_{n-2} + 2v_{n-4}$ and initial conditions, we have,

$$\sum_{k=2}^{(n-2)/2} v_{2k} = \frac{1}{3}(v_n + 2v_{n-2} - 5)$$

$$\sum_{k=2}^{(n-1)/2} v_{2k-1} = \frac{1}{3}(v_n + 2v_{n-2} - 3)$$

Then, $x_n = \frac{1}{3}[2v_n + v_{n-4} + \frac{1+(-1)^n}{2}]$. □

The *Hamming distance* between two nodes is the number of bits in which the two node addresses differ. It is the lower bound of the distance between two nodes. A *Hamming distance path* is a path between two nodes with length equal to the Hamming distance of these two nodes. The following theorem shows that there exists a shortest path between any two nodes in an EFC_n.

Property 2.3. *There exists a Hamming distance path for any two nodes in an EFC_n.*

Proof. We prove this theorem by induction on n. Obviously, The theorem is true for $n = 3, 4, 5, 6$. Suppose the theorem is true for all $n < k$ ($k > 6$). By Theorem 3, an EFC_n can be decomposed into four disjoint *EFC*s (see Figure 2). There are three cases for any two nodes in the EFC_n.

1. Two nodes are within one of the four subcubes: $00EFC_{n-2}$, $10EFC_{n-2}$, $0100EFC_{n-4}$, $0101EFC_{n-4}$.
2. Two nodes are in $00EFC_{n-2}$ and $10EFC_{n-2}$ (or $0100EFC_{n-4}$ and $0101EFC_{n-4}$), respectively.
3. One node is in $00EFC_{n-2}$ or $10EFC_{n-2}$, the other is in $0100EFC_{n-4}$ or $0101EFC_{n-4}$.

Note that the second case becomes the first case after one hop. The theorem is true for the first two cases by the induction assumption. The third case can be further divided into subcases. Let $\longleftrightarrow$ denote one hop between two EFC_is which have one-to-one node connections, $\Longleftrightarrow$ a Hamming distance path, and ps (s is a node in EFC_i) a node in $pEFC_i$, where p and s represent binary strings. The Hamming distance path between two nodes in Case 3 can be described as below.

3.1 One node is in $0010EFC_{n-4}$ (or $0000EFC_{n-4}$), the other in $0101EFC_{n-4}$ (or $0100EFC_{n-4}$).
- $0010s \longleftrightarrow 0000s \longleftrightarrow 0100s \longleftrightarrow 0101s \Longleftrightarrow 0101d$.

3.2 One node is in $000100EFC_{n-6}$, the other is in $0100EFC_{n-4}$ or $0101EFC_{n-4}$.
- $000100s \longleftrightarrow 000000s \longleftrightarrow 010000s \Longleftrightarrow 0100d$.
- $000100s \longleftrightarrow 010100s \Longleftrightarrow 0101d$.

3.3 One node is in $000101EFC_{n-6}$, the other is in $0100EFC_{n-4}$ or $0101EFC_{n-4}$.
- $00010110s \longleftrightarrow 00010010s \longleftrightarrow 01010010s \Longleftrightarrow 0101d$.
- $00010100s \longleftrightarrow 01010100s \Longleftrightarrow 0101d$.
- $00010101s \longleftrightarrow 01010101s \Longleftrightarrow 0101d$.
- $000101s \Longleftrightarrow 0101d \longleftrightarrow 0100d$.

3.4 One node is in $10EFC_{n-2}$, the other in $0100EFC_{n-4}$ (or $0101EFC_{n-4}$).
- $10s \longleftrightarrow 00s \Longleftrightarrow 0100d$ (or $0101d$). □

The *distance* between two nodes is defined as the length (in terms of the number of links) of the shortest path between them. The *diameter* is the maximum distance between any two nodes in a network.

Property 2.4. *The diameter of an EFC_n is $n-2$.*

Proof. A node of an EFC_n has $n-2$ bits, by Property 2.3, the maximum distance between any two nodes is $n-2$. Because an EFC_n contains an FC_n whose diameter is $n-2$, the theorem follows. □

The *node degree* is the number of connections at a node. It reflects the number of I/O ports required per node, and hence the cost of a node.

Property 2.5. *The node degree of a node in an EFC_n is between $\lceil \frac{n}{4} \rceil$ and $n-2$.*

Proof. The maximum node degree $n-2$ is because $n-2$ bits are used in an EFC_n and EFC_n has a subgraph of FC_n which has the maximum node degree $n-2$.

Let τ_n denote the minimum node degree in an EFC_n. We show $\tau_n = \tau_{n-4} + 1$, which implies $\tau_n = \lceil \frac{n}{4} \rceil$ $(n \neq 5)$ based on initial conditions $\tau_4 = 1$, $\tau_6 = 2$, $\tau_7 = 2$, $\tau_9 = 3$. We prove $\tau_n = \tau_{n-4} + 1$ for $n = 4k$ $(k \geq 2)$, the proof for $n = 4k + 1$, $n = 4k + 2$, and $n = 4k + 3$ is similar.

We claim that $(0101)^*10$ in an EFC_{4k} is of the lowest degree. It is true for $k = 2$. Suppose the claim is true for all $k < i$ $(i \geq 2)$. As $V_{4i} = 00V_{4i-2} \cup 10V_{4i-2} \cup 0100V_{4i-4} \cup 0101V_{4i-4}$, each node in the four EFC_{4i-2}s and EFC_{4i-4}s has increased its node degree by at least one. The $(0101)^*10$ in the EFC_{4i-4} concatenated with 0101 has increased its node degree by exactly one, since it is added to only one new link connecting to its corresponding node in the EFC_{4i-4} prefixed with 0100. Then the node $(0101)(0101)^*10$ in the EFC_{4i} has the lowest degree by the induction and $\tau_{4i-4} \leq \tau_{4i-2}$, thus $\tau_{4k} = \tau_{4k-4} + 1$. □

From Property 2.2, we have the link recursion in the following theorem.

PROPERTY 2.6. *For any EFC_n with $n > 6$,*

- $v_n = 2v_{n-2} + 2v_{n-4}$, *where* $v_3 = 2$, $v_4 = 3$, $v_5 = 5$ *and* $v_6 = 8$.
- $e_n = 2e_{n-2} + 2e_{n-4} + \frac{1}{3}[2v_n + v_{n-4} + \frac{1+(-1)^n}{2}]$, *where* $e_3 = 1$, $e_4 = 2$, $e_5 = 5$ *and* $e_6 = 10$.

By generating functionology [10], we can obtain the explicit formulas for v_n and e_n. However, we verify these formulas by induction on n.

PROPERTY 2.7 *For $n \geq 2$,*

- $v_{2n} = \frac{3+\sqrt{3}}{12}(1+\sqrt{3})^n + \frac{3-\sqrt{3}}{12}(1-\sqrt{3})^n$.
- $v_{2n-1} = \frac{1}{4}(1+\sqrt{3})^n + \frac{1}{4}(1-\sqrt{3})^n$.
- $e_{2n} = -\frac{5+4\sqrt{3}}{72}(1+\sqrt{3})^n - \frac{5-4\sqrt{3}}{72}(1-\sqrt{3})^n + \frac{9+4\sqrt{3}}{72}(1+\sqrt{3})^n n + \frac{9-4\sqrt{3}}{72}(1-\sqrt{3})^n n - \frac{1}{9}$.
- $e_{2n-1} = -\frac{7\sqrt{3}}{72}(1+\sqrt{3})^n + \frac{7\sqrt{3}}{72}(1-\sqrt{3})^n + \frac{5+\sqrt{3}}{48}(1+\sqrt{3})^n n + \frac{5-\sqrt{3}}{48}(1-\sqrt{3})^n n$.

Proof. We prove the formula for v_{2n} by induction on n, the proof for other formulas is similar.

For $n = 2$, 3, $v_4 = 3$, $v_6 = 8$, the formula is correct. Assume the formula is true for $n < k$ $(k \geq 2)$, then,

$$\begin{aligned}
v_{2k} =& 2v_{2k-2} + 2v_{2k-4} \\
=& 2[\frac{3+\sqrt{3}}{12}(1+\sqrt{3})^{k-1} + \frac{3-\sqrt{3}}{12}(1-\sqrt{3})^{k-1} \\
&+ \frac{3+\sqrt{3}}{12}(1+\sqrt{3})^{k-2} + \frac{3-\sqrt{3}}{12}(1-\sqrt{3})^{k-2}] \\
=& 2\frac{3+\sqrt{3}}{12}(1+\sqrt{3})^{k-2}(2+\sqrt{3}) \\
&+ 2\frac{3-\sqrt{3}}{12}(1-\sqrt{3})^{k-2}(2-\sqrt{3}) \\
=& \frac{3+\sqrt{3}}{12}(1+\sqrt{3})^k + \frac{3-\sqrt{3}}{12}(1-\sqrt{3})^k
\end{aligned}$$
□

3. EMBEDDING OTHER NETWORKS

In this section we study embedding of rings, hypercubes and trees into enhanced Fibonacci cubes. Once these networks can be embedded into *EFC*s, they can be emulated on the enhanced Fibonacci cubes. The efficiency of the emulation depends on the quality of embedding. There are several embedding criteria that are used to evaluate an embedding.

DEFINITION 3. An *embedding* of a *guest graph G* into a *host graph H* is a mapping f of the nodes of G to the nodes of H together with a mapping P_f which assigns each edge (u, v) of G to a path between $f(u)$ and $f(v)$ in H.

The *dilation* is defined as the maximum distance between the images of the adjacent nodes in G. The *congestion* is the maximum number of edges in G mapped to those paths including a same edge in H. The *load* is the maximum number of nodes of G that are mapped to a single node of H. And the *expansion* is the ratio of the number of nodes of G to that of H.

These measures are related to communication delay, traffic density, parallelism and processor utilization, respectively. They bound the speed and efficiency with which a host network can simulate the guest network. Therefore, an embedding is good when these measures are all small. By Property 2.1 the following theorem is apparent.

THEOREM 2. *An EFC_n $(n \geq 6)$ can embed a ring network of the same size with dilation, congestion, load and expansion all one.*

Proof. This is a result from Property 2.1. The embedding can be constructed based on the proof to Property 2.1. □

Let HC_n denote an n-dimensional hypercube.

THEOREM 3. *An EFC_n is a subgraph of HC_{n-2}, and an HC_n is a subgraph of EFC_{2n+1}.*

Proof. The first statement is true because the node set of the EFC_n which contains the codes of $n - 2$ bits is a subset of that of HC_{n-2} and the link conditions are same for these two cubes. The second one follows from the fact that HC_n is a subgraph of FC_{2n+1} which is a subgraph of EFC_{2n+1}. □

From the embedding point of view, Theorem 3 can be restated as:

- EFC_n can be embedded into HC_{n-2} with dilation 1, congestion 1, load 1 and expansion $\frac{2^{n-2}}{v_n}$
- HC_n can be embedded into HC_{2n+1} with dilation 1, congestion 1, load 1 and expansion $\frac{v_{2n+1}}{2^n}$

where v_n and v_{2n+1} are the numbers of nodes of EFC_n and EFC_{2n+1}, respectively.

A conclusion from the above theorem is that the hypercube and the enhanced Fibonacci cube can emulate each other.

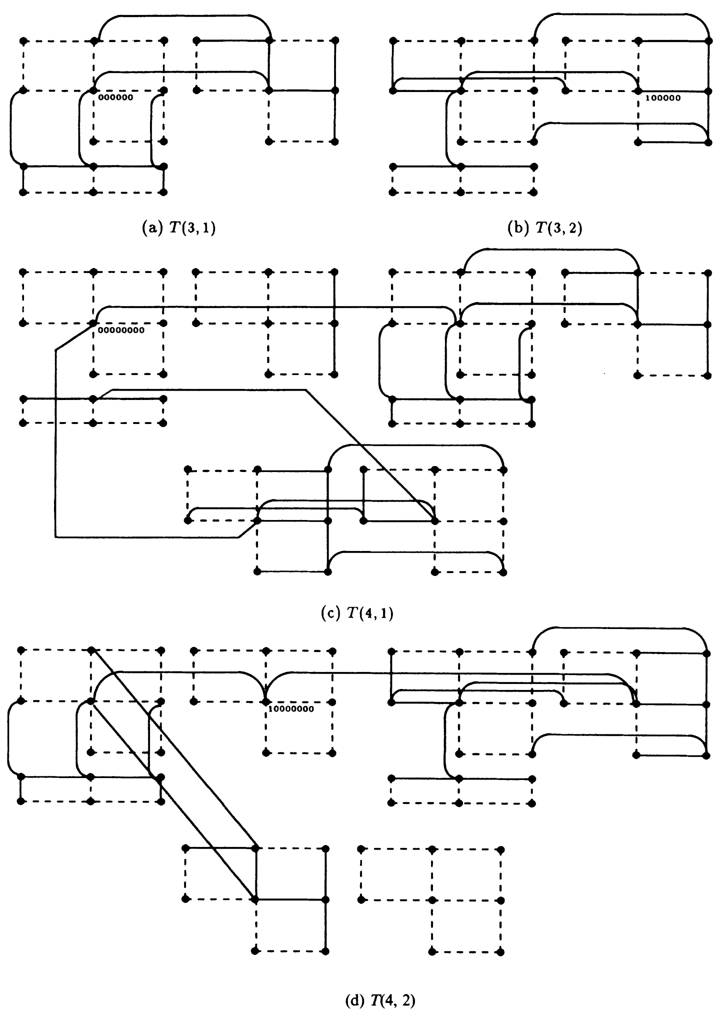

FIGURE 3. The embedding of 15- and 31-node trees in an EFC_8 and an EFC_{10} respectively.

Theorem 4. *For $n \geq 4$, the $(2^n - 1)$-node complete binary tree can be embedded into EFC_{2n} with dilation 1, congestion 1, load 1 and expansion $\frac{v_{2n}}{2^n-1}$, where v_{2n} is the number of nodes of EFC_{2n}, i.e. the $(2^n - 1)$-node tree is a subgraph of EFC_{2n}.*

Proof. We prove this theorem by induction on n. In addition we show that the $(2^n - 1)$-node tree can be embedded in EFC_{2n} in two ways, the embedded tree can be rooted at 0^{2n-2} or at $0^{2n-8}10^5$ in EFC_{2n}, where 0^k represents a sequence of 0s with length k $(k \geq 0)$.

For $n = 4, 5$, the claim is true. Let $T(h, 1)$ and $T(h, 2)$ represent the embedded trees with height h in an EFC_{2i}, where root nodes are 0^{2i-2} and $0^{2i-8}10^5$, respectively. The two-way embedding of 15-node and 31-node trees into an EFC_8 and an EFC_{10} are shown in Figure 3, where some links are omitted for simplicity.

Assume that for all $n < k$ $(k \geq 4)$ the claim is true. Then $(2^{k-1} - 1)$-node and $(2^{k-2} - 1)$-node trees can be embedded in an EFC_{2k-2} and an EFC_{2k-4}, respectively, in two ways. Figure 4 shows the embedding of a $(2^k - 1)$-node tree into an EFC_{2k} with the specified root locations. □

The embedding of trees into odd EFC_{2n-1}s can be performed in a similar way. The key is to find the basis for embedding which determines the expansion.

4. THE ENHANCED FIBONACCI TREE AND ITS APPLICATIONS

In hypercubes many applications such as broadcasting, prefix sum computing and load balancing can be solved with the aid of *binomial trees* (special spanning trees of hypercube). Fibonacci trees which are corresponding to binomial trees have been used to compute prefix sums on Fibonacci cubes [11]. Similarly, we have *enhanced Fibonacci trees* which can be used to solve some problems like broadcasting and prefix sum computing on enhanced Fibonacci cubes.

Recall that a *binomial tree of order n* (BT_n) is formed by two binomial trees BT_{n-1}s, where the root of one BT_{n-1} becomes the root of BT_n, and the root of the other BT_{n-1} becomes the child of the root of the former BT_{n-1}. A *Fibonacci tree of order n* (FT_n) is a graph that consists of an FT_{n-1} and an FT_{n-2}, where the root of FT_{n-2} becomes a child of the root of FT_{n-1}. As the initial

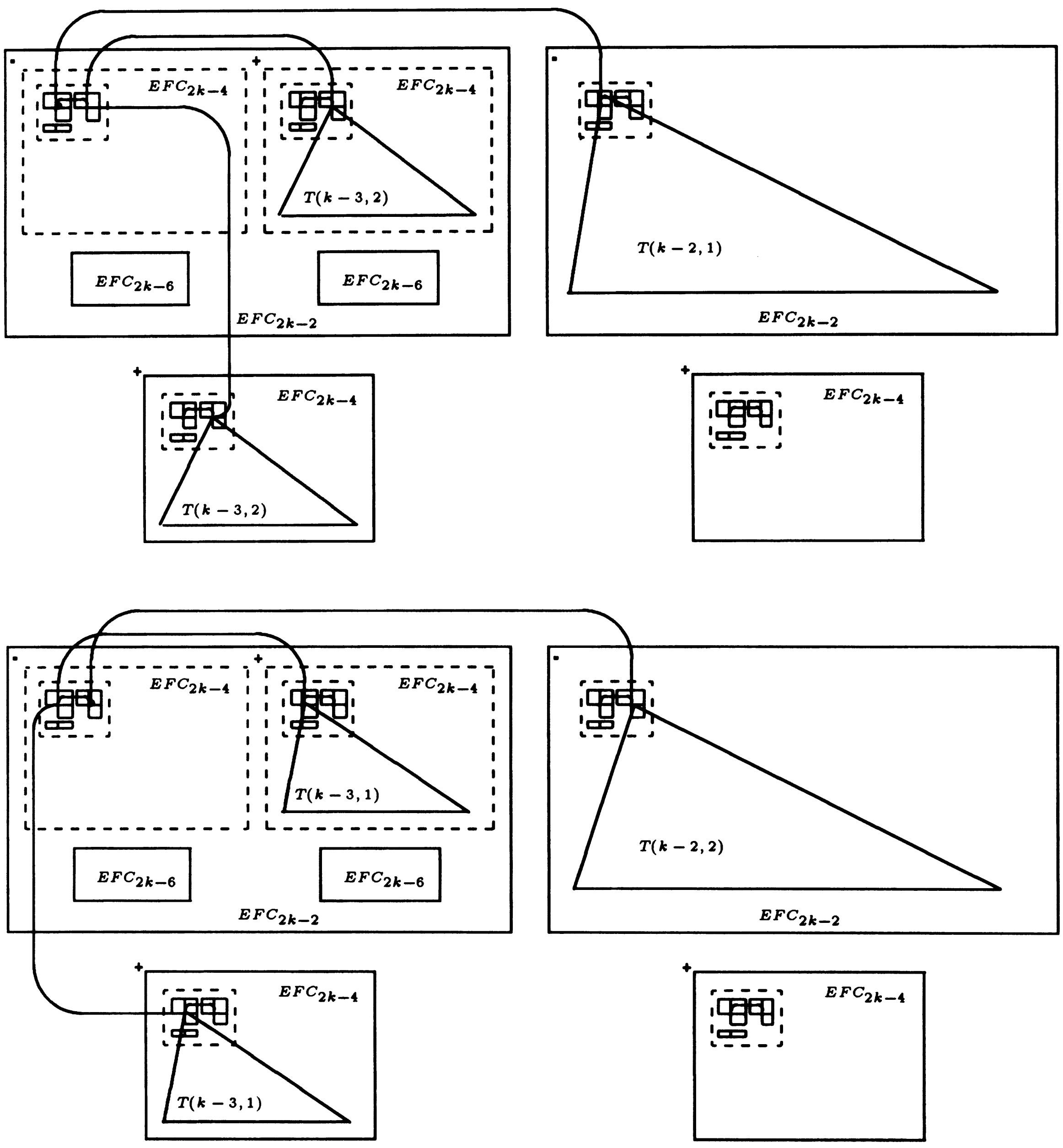

FIGURE 4. The embedding of a (2^k-1)-node tree in an EFC_{2k}.

conditions, FT_0 is an empty graph and FT_1 is a single node. Similarly, we can define an enhanced FT as below.

DEFINITION 4. *An enhanced Fibonacci tree of order n,* denoted by EFT_n, comprises two EFT_{n-2}s and two EFT_{n-4}s, where the root of $00EFT_{n-2}$ is selected as the root of EFT_n, the roots of $0100EFT_{n-4}$ and $10EFT_{n-2}$ become the children of the root of $00EFT_{n-2}$, and the root of $0101EFT_{n-4}$ becomes the child of the root of $0100EFT_{n-4}$. As the basis, EFT_3, EFT_4, EFT_5 and EFT_6 are the same as those FT of the same order, respectively.

Figure 5 shows the examples of BTs, FTs and EFTs.

As with EFCs, we first analyse topological properties of the enhanced Fibonacci tree. The proofs to some properties can be found in [12].

PROPERTY 4.1. *An EFT_n is a spanning tree of EFC_n.*

PROPERTY 4.2. *An EFT_n can be decomposed into two EFT_{n-2}s and two EFT_{n-4}s, the four subtrees are disjoint.*

The *span* of a tree is the maximum node degree of the tree. The *height* of a tree is the distance between the root and the furthermost leaf. The heights of a BT_n and an FT_n are n and $\lceil \frac{n-2}{2} \rceil$ [11] respectively.

PROPERTY 4.3. *The span of an EFT_n is $n-2$.*

PROPERTY 4.4. *The height of an EFT_n is $\lceil \frac{n-2}{2} \rceil$.*

As mentioned earlier, an EFC_n is a subgraph of a $(n-2)$-dimensional hypercube. Node $\overbrace{00\cdots0}^{n-2}$ of EFC_n has $n-2$ neighbours along $n-2$ dimensions. In the EFT

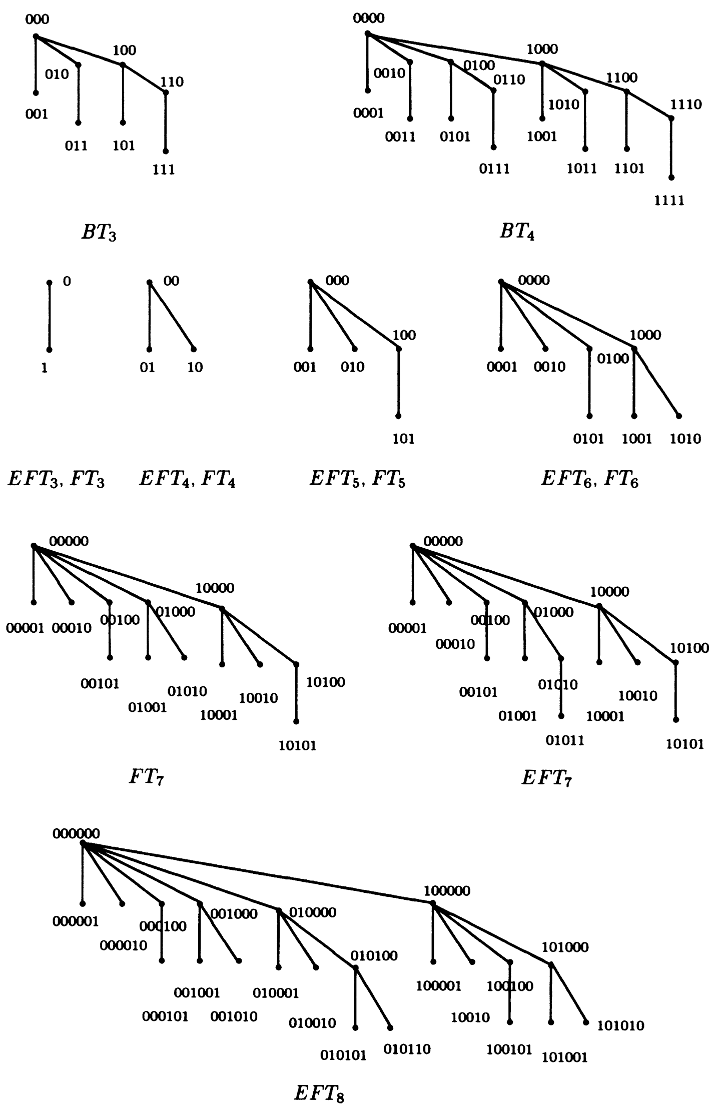

FIGURE 5. Binomial, Fibonacci and enhanced Fibonacci trees.

of order n, these $n-2$ neighbours are children of the root $\overbrace{00\cdots0}^{n-2}$ and are ordered such that the ith child (the leftmost child is the first child) is the neighbour on the ith dimension. For example, in an EFT_8 of Figure 5, the 5th child 010000 of the root is its neighbour on the 5th dimension.

PROPERTY 4.5. *In an EFT_n, the children of the root are dimension ordered, i.e. the ith child of the root is the neighbour of the root on the ith dimension.*

The *pre-order* of a tree is an order such that the root is the first, then the nodes from the first subtree of the root in pre-order, then the nodes from the second subtree in pre-order, and so on.

PROPERTY 4.6. *The pre-order of an EFT_n is the same as the order by the binary values of node addresses.*

The *EFT*s have the same relation with *BT*s and *FT*s as *EFC*s with *HC*s and *FC*s.

PROPERTY 4.7. *An EFT_n is a supergraph of an FT_n and an EFT_n is a subgraph of an BT_{n-2}.*

There is an interesting property regarding the relations between the addresses of parents and children in the EFT_n. Let a be an address of a node, $a(i)$ be the value of ith bit in a, and a^i be the address such that a and a^i differ only in bit i, i.e. a^i denotes the neighbour of node a along dimension i.

PROPERTY 4.8. *Let $a \in V(EFC_n)$, and i be the rightmost bit of a that has value 1, then:*

- *The parent of node a is a^i.*
- *The children of node a form the set Children_set, where Children_set $= \{a^j \mid a^j \in V(EFC_n)$ for $0 \leq j < i\}$*
- *$a^i(i) = 0$ and $a(i) = 1$.*

For example, in an EFT_8 (Figure 5), the rightmost bit of 010100 that has value 1 is bit 3, then the parent is 010000 and the children are 010110 and 010101. Based on this property, each node of an EFC_n can easily find its

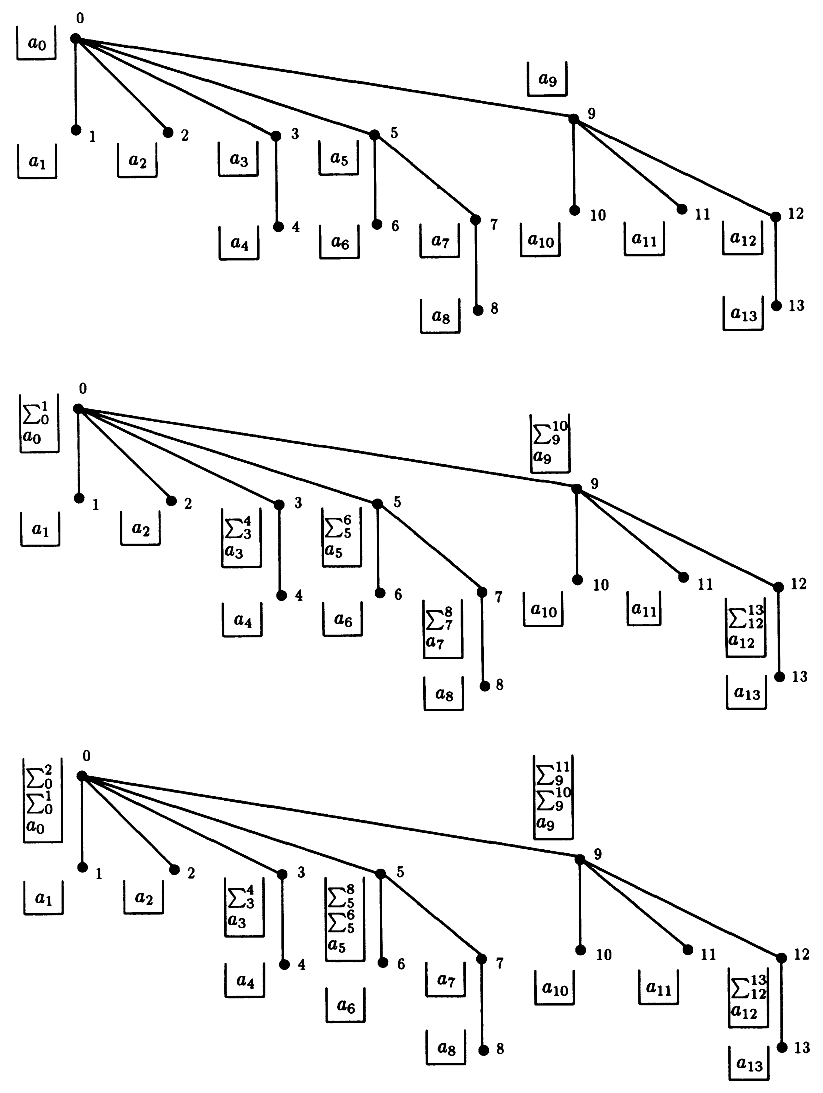

FIGURE 6. Ascending: dimensions 1, 2, 3.

parent and children in the EFT_n without information exchange with other processors.

DEFINITION 5. Let $+$ denote an associative binary operator on a semigroup $(G, +)$. Let $S : a_0, a_1, \ldots, a_{n-1}$ denote a sequence of n elements from G. The *ith prefix sum* $P(i)$ of S, where $0 \le i \le n-1$, is defined to be the sum $a_0 + a_1 + \cdots + a_{i-1}$. The *parallel prefix sum problem* is to compute all n prefix sums in parallel.

Assume that each node in an EFT_n is labelled according to the pre-order of the EFT_n, and initially node i has data element a_i, $0 \le i \le v_n - 1$, where v_n is the number of nodes of the EFC_n.

After prefix sum computation, the prefix sum $P(i) = \sum_{j=0}^{i-1} a_j$ is stored in node i.

In the algorithm *PrefixSum* (in the Appendix), a stack is used in each node i to store the local prefix sums of subtrees under the node i (including node i). Initially, the stack contains only data element a_i. There are ascending and descending phases. In the ascending phase, data are added bottom up each time along one dimension (starting from dimension 1 to dimension $n-2$). At each node i, the local prefix sums of its subtrees are pushed in its stack in the order of these subtrees (the leftmost subtree is the first), when it receives data from its children. The sum of the subtree rooted at node i is popped from its stack when it sends the sum to its parent. This sum is no longer kept in the stack since it is not needed in the descending phase. This is why there is a step to remove the top data from the stack of the root after the ascending phase ends. However, each stack always keeps a copy of the initial data element at its bottom. In the descending phase, prefix sums are transmitted down and are added to local prefix sums each time along one dimension (starting from dimension $n-2$ to dimension 1). When a node sends data to one of its children along a certain dimension, it pops the top data from its stack and sends it to this child. If the stack is empty it makes a copy of the data to keep its own prefix sum. When a node receives data from its parent, it adds the data to each element in the stack to update local prefix sums to prefix sums.

On a SIMD machine where instructions executed at different nodes are synchronized by a system clock, the

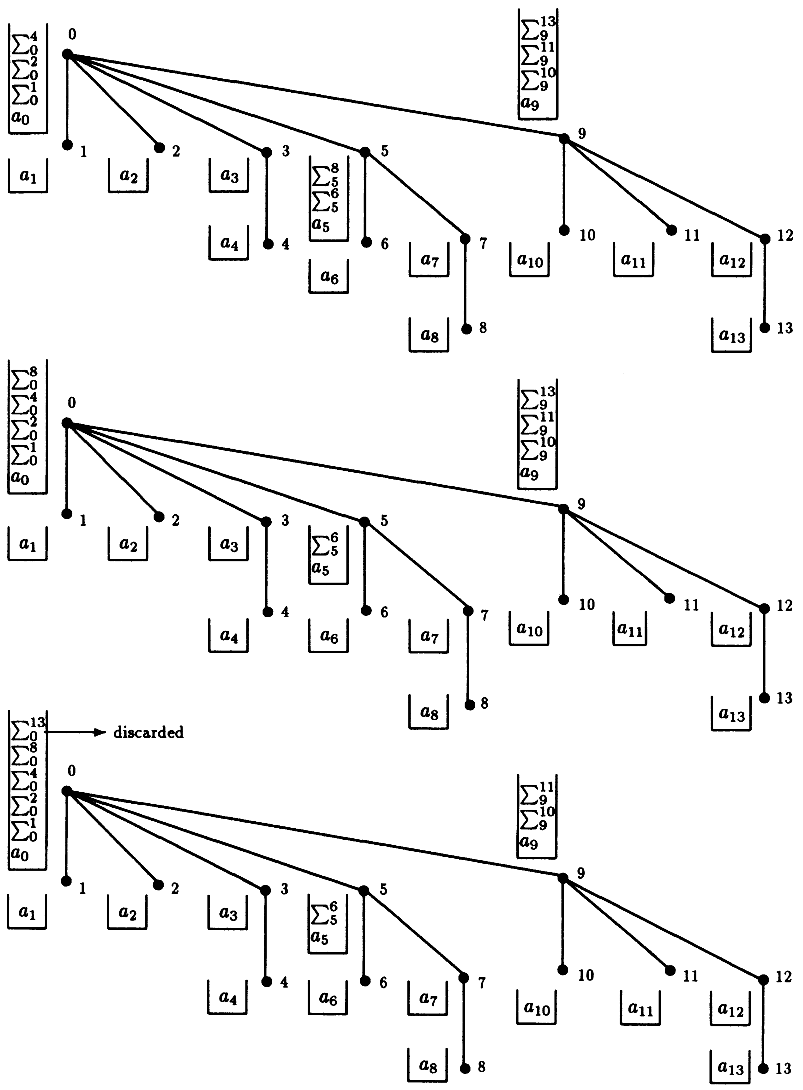

FIGURE 7. Ascending: dimensions 5, 6.

transmission of data dimension by dimension in the *PrefixSum* algorithm is automatically synchronized. Figures 6, 7, 8, 9 show an example of computing prefix sums on EFC_7.

As we know, it takes logarithmic steps to compute 2^n data item prefix sums on an n-dimensional hypercube (HC_n) [13] and $O(n)$ time to compute f_n (nth Fibonacci number) data element prefix sums on FC_n [11]. For the EFC, we have the following result.

Theorem 5. *The prefix sum of v_n data items can be computed on an EFC_n in $O(n)$ time.*

Proof. This is evident since $2 \cdot (n-2)$ steps are needed in the ascending and descending phases. □

The efficiency of the prefix sum computation on the *EFC* is between those of computations on the *HC* and *FC*. Because time complexities for both cases are $O(n)$ and the problem size varies: 2^n for HC_n, f_n for FC_n and v_n for EFC_n. When n is large enough, $f_n \rightarrow \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n$ [10] which is $O(1.62^n)$, and $v_n \rightarrow O(1.65^n)$. The problem size of EFC_n is larger than that of FC_n, but smaller than that of HC_n.

Note that studying properties and applications of individual networks is time consuming and offers little insight about the interrelations of different interconnection schemes. In the next section, we present and study a large family of enhanced *FC*s.

5. EXTENSION OF THE ENHANCED FIBONACCI CUBE

We can create a series of enhanced Fibonacci cubes $EFC_n^{(k)}$s by varying the initial values of the enhanced Fibonacci cube. $EFC_n^{(k)}$s contain EFC_n (the enhanced Fibonacci cube we have discussed so far) as a subcube and maintain all its properties. $EFC_n^{(p)}$ also contains $EFC_n^{(q)}$s that precede ($q < p$) it in the series. All $EFC_n^{(k)}$s are subcubes of hypercube HC_{n-2}. With the introduction of the series of the enhanced Fibonacci cubes, there are even more options for selecting cubes with various sizes.

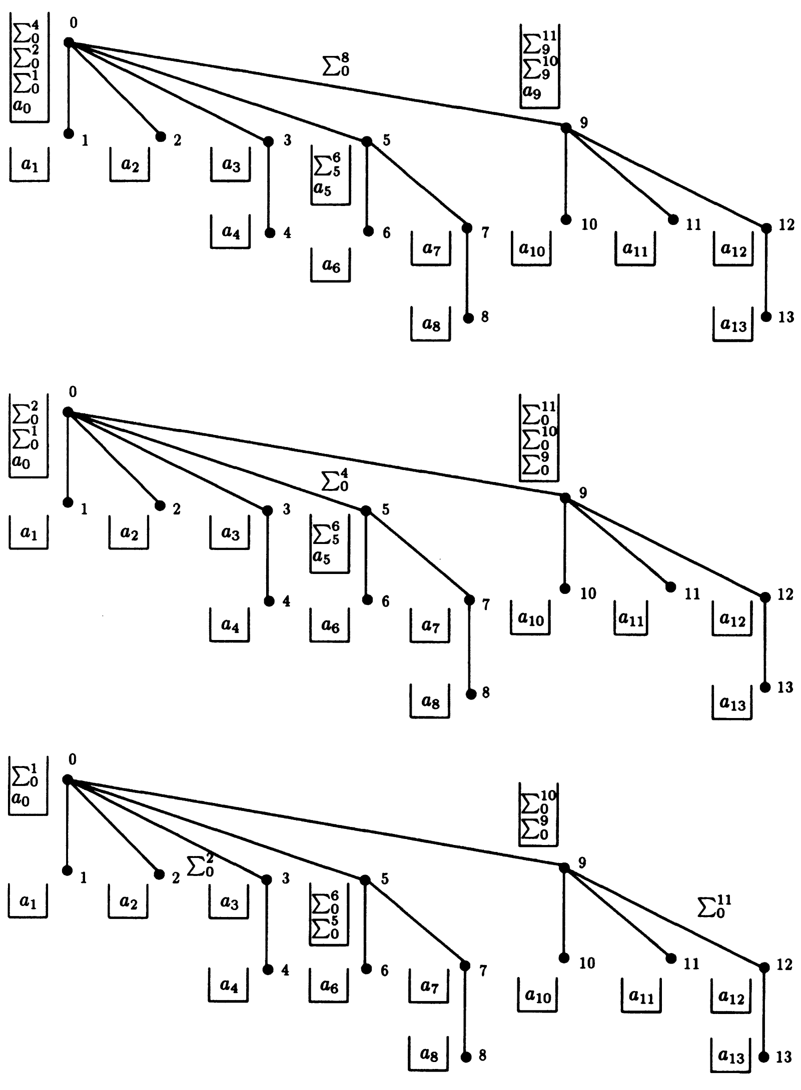
Σ_0^4 Σ_0^2 Σ_0^1 a_0
Σ_0^8
Σ_9^{11} Σ_9^{10} a_9
Σ_5^6 a_5
a_1 a_2 a_3 a_4 a_6 a_7 a_8 a_{10} a_{11} a_{12} a_{13}
0 1 2 3 4 5 6 7 8 9 10 11 12 13
Σ_0^2 Σ_0^1 a_0
Σ_0^4
Σ_0^{11} Σ_0^{10} Σ_0^9
Σ_5^6 a_5
a_1 a_2 a_3 a_4 a_6 a_7 a_8 a_{10} a_{11} a_{12} a_{13}
0 1 2 3 4 5 6 7 8 9 10 11 12 13
Σ_0^1 a_0
Σ_0^2
Σ_0^{10} Σ_0^9
Σ_0^6 Σ_0^5
Σ_0^{11}
a_1 a_2 a_3 a_4 a_6 a_7 a_8 a_{10} a_{11} a_{12} a_{13}
0 1 2 3 4 5 6 7 8 9 10 11 12 13

FIGURE 8. Descending: dimensions 5, 4, 3.

The series of extended Fibonacci cubes proposed by Wu [14] and enhanced Fibonacci cubes are defined as follows.

Definition 6. [14]: A series of extended Fibonacci cubes is defined as $\{XFC_n^{(k)} \mid k \geq 0,\ n \geq k+2\}$, where $XFC_n^{(k)} = (XV_n^{(k)},\ XE_n^{(k)})$ denotes the *kth extended Fibonacci cube of order n*, and $XV_n^{(k)} = 0\|XV_{n-1}^{(k)} \cup 10\|XV_{n-2}^{(k)}$. Two nodes in $XFC_n^{(k)}$ are connected by an edge in $XE_n^{(k)}$ if and only if their labels differ in exactly one bit position. As initial conditions for recursion, we have: $XV_{k+2}^{(k)} = \{d_{k-1}d_{k-2}\cdots d_0 \mid d_i \in \{0,1\}, 0 \leq i \leq k-1\}$ and $XV_{k+3}^{(k)} = \{d_k d_{k-1}\cdots d_0 \mid d_i \in \{0,1\}, 0 \leq i \leq k\}$.

Notice that $XFC_n^{(0)}$ is actually FC_n. Those two initial cubes of $XFC_n^{(k)}$ are two complete HCs: $XFC_{k+2}^{(k)} = HC_k$, $XFC_{k+3}^{(k)} = HC_{k+1}$.

Definition 7. A series of *EFC*s is defined as $EFC_n^{(k)} \mid k \geq 0,\ n \geq k+3$, where $EFC_n^{(k)} = (V_n^{(k)},\ E_n^{(k)})$ denotes the *kth EFC* of order *n*, and $V_n^{(k)} = 00\|V_{n-2}^{(k)} \cup 10\|V_{n-2}^{(k)} \cup 0100\|V_{n-4}^{(k)} \cup 0101\|V_{n-4}^{(k)}$. Two nodes in $EFC_n^{(k)}$ are connected by an edge in $E_n^{(k)}$ if and only if their labels differ in exactly one bit position. As initial conditions for recursion, we have: $V_{k+3}^{(k)} = XV_{k+3}^{(k)}$, $V_{k+4}^{(k)} = XV_{k+4}^{(k)}$, $V_{k+5}^{(k)} = XV_{k+5}^{(k)}$, and $V_{k+6}^{(k)} = XV_{k+6}^{(k)}$, where $XV_i^{(k)}$ $(k+3 \leq i \leq k+6)$ is the number of nodes of $XFC_i^{(k)}$.

$EFC_n^{(0)}$ is the original *EFC* we studied earlier. Table 1 shows the sizes of FC_n and $EFC_n^{(k)}$ for $0 \leq k \leq 6$ and $2 \leq n \leq 14$. The boxed numbers are the initial values for cubes. The four initial values for $EFC_n^{(k)}$ $(0 \leq k \leq 6)$ are borrowed from $XFC_n^{(k)}$.

In the notations $EFC_n^{(k)}$ and $XFC_n^{(k)}$, we call *k* a serial number and *n* an order. Now we study the properties of the series of $EFC_n^{(k)}$s. We show that the series of $EFC_n^{(k)}$s maintain all the properties of $EFC_n^{(0)}$.

First we discuss the subcube relationships between the series of the *EFC*s and the series of extended Fibonacci cubes, between two cubes in the series, and between the series of the *EFC*s cubes and *HC*s.

PROPERTY 5.1. *$XFC_n^{(k)}$ is a subcube of $EFC_n^{(k)}$ for $n \geq k+3$.*

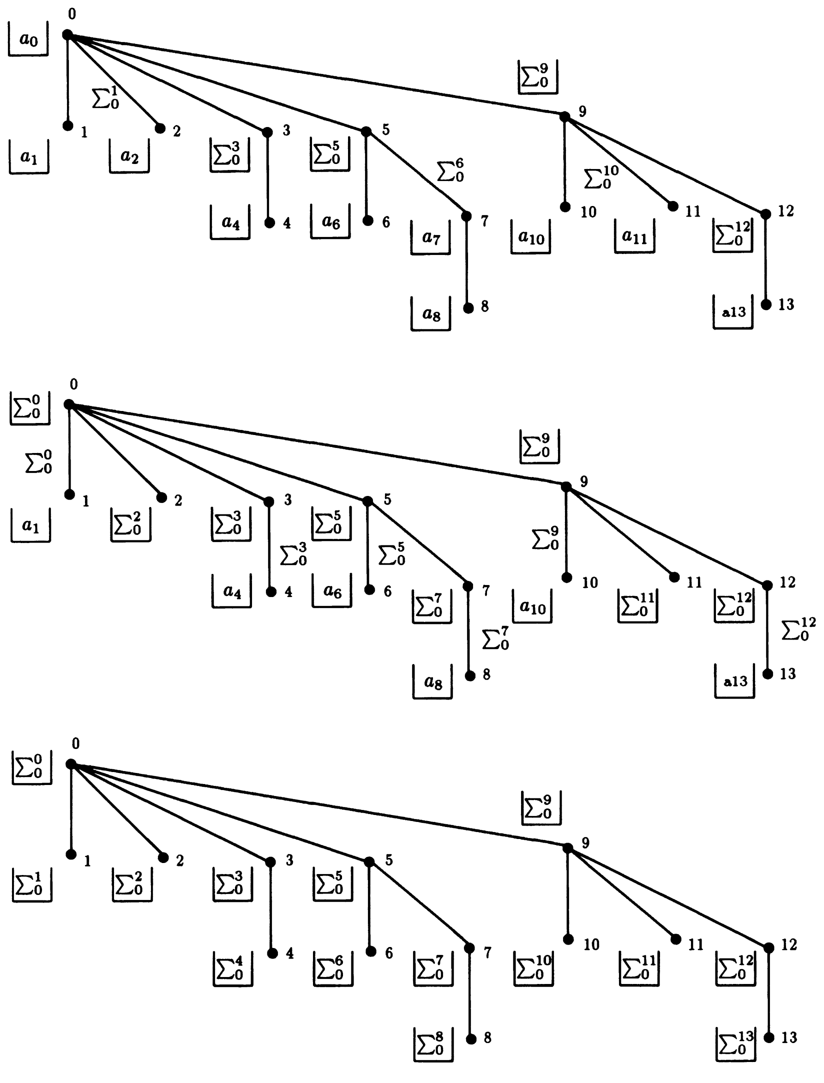

FIGURE 9. Descending: dimensions 2, 1.

PROPERTY 5.2. *$EFC_{2n}^{(k)}$ is a subgraph of $EFC_{2m}^{(k)}$ if $n \leq m$ $(2n \geq k+3)$ and $EFC_{2n-1}^{(k)}$ is a subgraph of $EFC_{2m-1}^{(k)}$ if $n \leq m$ $(2n-1 \geq k+3)$.*

PROPERTY 5.3. *For $n \geq p+3$ and $n \geq q+3$, $EFC_n^{(p)}$ is a subgraph of $EFC_n^{(q)}$ if $p \leq q$.*

The relation in Property 5.3 is shown in Figure 10.

PROPERTY 5.4. *All $EFC_n^{(k)}$s are Hamiltonian $(n \geq 6$ for $k = 0$ and $n \geq k+3$ for $k \geq 1)$.*

PROPERTY 5.5. *$EFC_n^{(k)}$ can be decomposed into two $EFC_{n-2}^{(k)}$s and two $EFC_{n-4}^{(k)}$s, and these four subgraphs are disjoint.*

The diameter of any $EFC_n^{(k)}$ keeps $n-2$, since it contains FC_n whose diameter is $n-2$ as a subcube, while it is contained in HC_{n-2}. So the diameter is a function of only n. The range of node degree of $EFC_n^{(k)}$ does depend on both n and k.

TABLE 1. Sizes of FC_ns and $EFC_n^{(k)}$s for $0 \leq k \leq 6$ and $2 \leq n \leq 14$

	2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
FC_n	1	2	3	5	8	13	21	34	55	89	144	233	377
$EFC_n^{(0)}$	–	2	3	5	8	14	22	38	60	104	164	284	448
$EFC_n^{(1)}$	–	–	4	6	10	16	28	44	76	120	208	328	568
$EFC_n^{(2)}$	–	–	–	8	12	20	32	56	88	152	240	416	656
$EFC_n^{(3)}$	–	–	–	–	16	24	40	64	112	176	304	480	832
$EFC_n^{(4)}$	–	–	–	–	–	32	48	80	128	224	352	608	960
$EFC_n^{(5)}$	–	–	–	–	–	–	64	96	160	256	448	704	1216
$EFC_n^{(6)}$	–	–	–	–	–	–	–	128	192	320	512	896	1408

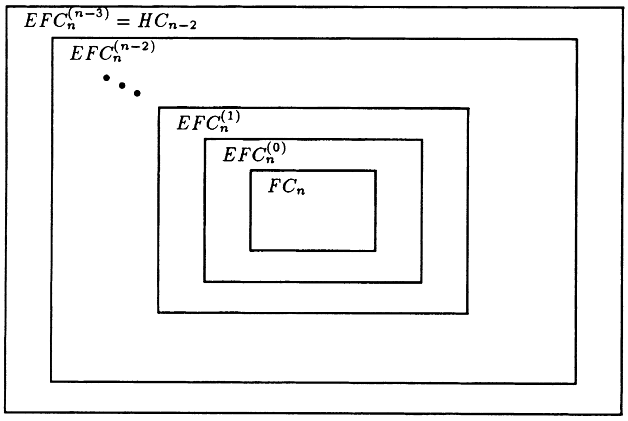

FIGURE 10. Relationship among FC_n, $EFC_n^{(k)}$s and HC_{n-2}.

PROPERTY 5.6. *The diameter of $EFC_n^{(k)}$ is $n - 2$.*

PROPERTY 5.7. *The node degree of a node in $EFC_n^{(k)}$ is between $\lceil \frac{n-k}{4} \rceil + k$ and $n - 2$.*

PROPERTY 5.8. *There exists a Hamming distance path between any two nodes in $EFC_n^{(k)}$.*

PROPERTY 5.9. *$EFC_n^{(k)}$ is a subgraph of HC_{n-2} and HC_n is a subgraph of $EFC_{2n+1}^{(k)}$.*

Similarly, since $EFC_{2n}^{(0)}$ is a subgraph of $EFC_{2n}^{(k)}$, by Theorem 4, we have the following result.

PROPERTY 5.10. *A $(2^n - 1)$-node tree is a subgraph of $EFC_{2n}^{(k)}$.*

We can define the *EFT* for $EFC_n^{(k)}$ exactly as we did for $EFC_n^{(0)}$ and show that the desirable properties of the enhanced Fibonacci tree of $EFC_n^{(0)}$ are maintained.

We have computed the sizes of some *HC*s, *EFC*s and *XFC*s through a program and converted results to curves. Figure 11 displays the plot of the size of the cubes versus order for HC_n, $EFC_n^{(k)}$ and $XFC_n^{(k)}$ ($0 \leq k \leq 5$, $30 \leq n \leq 80$). We can see that, for the same order n, HC_n has the largest size, the size of $EFC_n^{(k)}$ is next, then $XFC_n^{(k)}$. Also for $EFC_n^{(k)}$ and $XFC_n^{(k)}$, the larger k or n, the larger the size of cube. These observations are compatible with our results in Properties 5.1–5.3.

Another simulation program has been developed and run to obtain information on sizes (between 2 and 2^{30}) of FC_n, $XFC_n^{(k)}$, $EFC_n^{(k)}$ and HC_n. Results are summarized in Table 2, which shows the total number of each kind of cubes and the number of collisions with other cubes of the same or different kind, for all cubes of sizes between 2 and 2^{30}.

TABLE 2. Cube size collision between 2 and 2^{30}

	Total number between 2 and 2^{30}	*Collide with same kind*	*Collide with different kind*
FC	41	0	1
XFC	575	0	51
EFC	507	22	24
HC	30	0	30

From simulation results we have the following observations: (1) out of 507 *EFC*s with sizes between 2 and 2^{30}, only 46 of them have the same sizes as other cubes, i.e. 91% *EFC*s bring new sizes that are different from those of existing cubes; (2) in each gap $(2^i, 2^{i+1})$ $(1 \leq i < 30)$, there are less than 3 or constant 3 size collisions. In particular, in each of following three pairs, two cubes have the same size: HC_{i-3} and $XFC_i^{(i-6)}$, $EFC_i^{(i-11)}$ and $XFC_{i-1}^{(i-8)}$, $EFC_i^{(i-7)}$ and $EFC_{i+2}^{(i-12)}$. If the number of size collisions keeps constant 3 for any gap, when i becomes large, the number of collisions can be ignored compared with the gap size 2^i which is very large.

6. CONCLUSION

We have shown that the proposed *EFC* has some desirable properties. The network contains the *FC* as its subnetwork and maintains virtually all the properties of the *FC*. In addition, the *EFC* is a Hamiltonian graph. The unit dilation and low expansion embedding of rings and trees allow *EFC* to simulate ring and tree networks with high efficiency. Many *HC* algorithms can also be effectively simulated on *EFC*. We have studied the properties of *EFT*s, and have shown that they can be used to compute parallel prefix sum on the enhanced Fibonacci cube in $O(n)$ time. With the introduction of the enhanced Fibonacci cube into the cube family, there are more choices in selecting cubes with various sizes. We also have presented a series of enhanced Fibonacci cubes. This series also provides more options of network sizes and more incomplete hypercubes that faulty hypercubes can be reconfigured to. The hypercube and Fibonacci cube are special cases in the series of enhanced Fibonacci cubes, which also provides a clearer view of generalization and interrelations of a subset of cubes in the hypercube family.

Efficient routing and broadcasting algorithms are critical to the performance of the parallel and distributed systems. In a parallel scientific computing, data communication among processors is very common. It may be specified by parallel programs; or be required by load redistribution to keep each processor busy, and so on. This data communication significantly affects the efficiency of parallel programs because of communication delay. There are a few commonly used communication primitives such as unicast (one-to-one), multicast (one-to-many) and broadcast (one-to-all). Proper implementation of these basic communication operations is a key to the efficient execution of a parallel algorithm that uses them. In a separate paper [15], we have discussed the efficient implementation of these three primitives. Our result shows that unicast on EFC_n can be time and traffic optimal, where time is measured by the number of time steps used to complete a communication and traffic is measured by the number of total links visited in a communication. Broadcast on EFC_n is traffic optimal for any dimension n, and it is time optimal when n is even or one step more than the optimal case when n is odd.

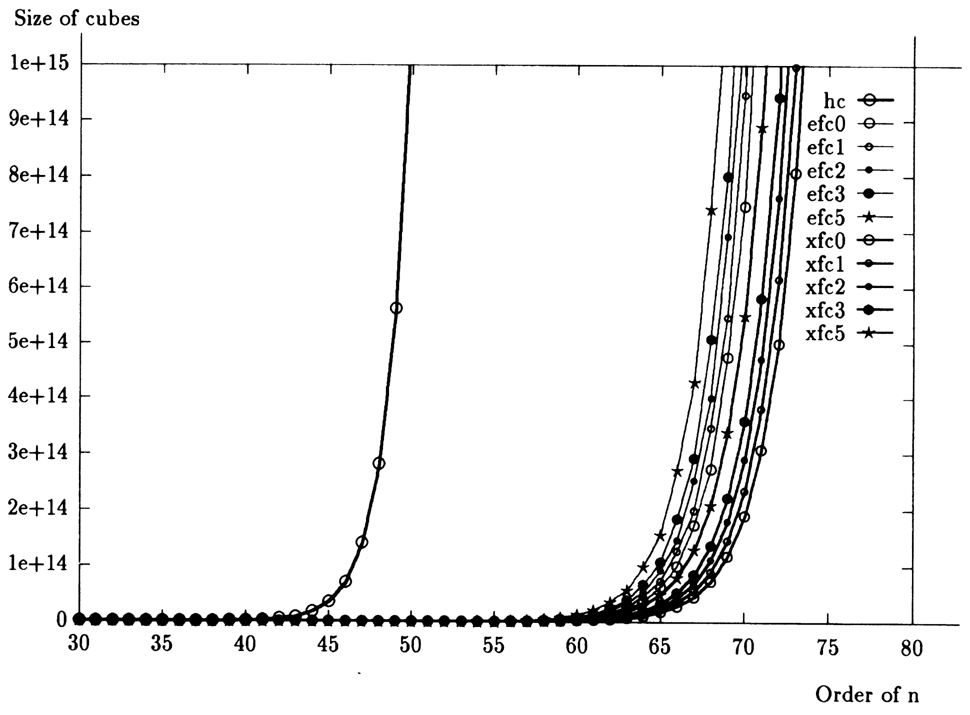

FIGURE 11. Size of cube versus order for HC_n, $EFC_n^{(k)}$ and $XFC_n^{(k)}$ ($0 \leq k \leq 5$, $30 \leq n \leq 80$).

Our future research will include the identification of more applications that are suitable to run on the enhanced Fibonacci cube and VLSI/WSI layout of the enhanced Fibonacci cube.

ACKNOWLEDGEMENTS

We thank the referees for their suggestions and corrections.

REFERENCES

[1] Saad, Y. and Schultz, M. H. (1988) Topological properties of hypercubes. *IEEE Trans. Comput.*, **37**, 867–872.

[2] Hayes, J. P. and Mudge, T. (1989) Hypercube Supercomputer. *Proc. IEEE*, **77**, 1829–1841.

[3] Athas, W. C. and Seitz, C. L. (1988) Microcomputer: Message-passing Concurrent Computers. *IEEE Comput.*, **21**, 9–25.

[4] NCUBE Company (1990) *NCUBE 6400 Processor Manual.* Foster City, CA.

[5] Durham, T. and Johnson, T. (1986) *Parallel Processing: The Challenge of New Computer Architectures.* Ovum Ltd, London, UK.

[6] Katseff, H. P. (1988) Incomplete Hypercubes. *IEEE Trans. Comput.*, **37**, 604–608.

[7] Tzeng, N. F., Chen, H. L. and Chuang, P. J. (1990) Embeddings in Incomplete Hypercubes. *Proc. of 1990 Int. Conf. on Parallel Processing*, Volume III, St Charles, IL, pp. 335–339. CRC Press, Boca Raton, FL.

[8] Hsu, W. J. (1993) Fibonacci Cubes - A New Interconnection Topology. *IEEE Trans. Parallel Distrib. Sys.*, **4**, 3–12.

[9] Cong, B., Zheng, S. Q. and Sharma, S. (1993) On Simulations of Linear Arrays, Rings and 2-d Meshes on Fibonacci Cube Networks. *Proc. of 7th Int. Parallel Processing Symp.*, Newport Beach, CA, pp. 748–751. IEEE Computer Society Press, Los Alomitos CA.

[10] Wilf, H. S. (1990) *Generating functionology.* Academic Press, Inc., San Diego, CA.

[11] Hsu, W. J. and Page, C. V. and Liu, J. S. (1992) Computing Prefixes on a Large Family of Interconnection Topologies *Proc. of 1992 Int. Conf. on Parallel Processing*, Volume III, St Charles, IL, pp. 153–159. CRC Press, Boca Raton, FL.

[12] Qian, H. and Wu, J. (1994) Prefix Computation on Enhanced Fibonacci Cubes. *Technical Report, TR-CSE-94-28*, Florida Atlantic University, FL.

[13] Johnsson, S. L. and Ho, C. T. (1989) Optimal Broadcasting and Personalized Communication in Hypercubes. *IEEE Trans. Comput.*, **38**, 1249–1268.

[14] Wu, J. (1995) Extended Fibonacci Cubes. *Proc. of IEEE Symp. Parallel Distrib. Processing*, San Antonio, TX, pp. 248–251. IEEE Computer Society Press, Los Alomitos, CA.

[15] Qian, H. and Wu, J. (1995) Unicast, Multicast and Broadcast on Enhanced Fibonacci Cubes. *Proc. of 4th Int. Conf. Comput. Commun. Networks (IC3N'95)*, pp. 20–23. Las Vegas, CA.

APPENDIX

PrefixSum algorithm

```
For each node $a$ in $EFT_n$ do:
• Boolean $seen = FALSE$
•for $i = 1$ to $n - 2$
    ◇ If not $seen$ and $a(i) = 1$ then
        – pop up the top data from its stack;
        – if the stack is empty then push back a copy of the data;
        – send the data to its parent $a^i$;
        – $seen = TRUE$;
        – $rightmost\_bit = i$.
    ◇ If node $a$ receives data from its child then
        – $sum \Leftarrow$ data on top of the stack + data received;
        – Push $sum$ in the stack.
• If $a = 00\cdots0$ then
    ◇ pop up the top data from the stack;
    ◇ $rightmost\_bit = n - 1$.
• for $i = n - 2$ to 1
    ◇ If $i < rightmost\_bit$ and $a^i \in V(EFC_n)$ then
        – pop up the top data from its stack;
        – if the stack is empty then push back a copy of the data;
        – send the data to the child $a^i$.
    ◇ If node $a$ receives data from its parent then
        – add the data to the only element in its stack.
```