Embedding trees in recursive circulants

Hyeong-Seok Lim\textsuperscript{a,b}, Jung-IIeum Park\textsuperscript{b,1}, Kyung-Yong Chwa\textsuperscript{b,*}

\textsuperscript{a} Department of Computer Science, Chonnam National University, Yongbong-dong 300, Buk-gu, Kwangju 500-757, South Korea
\textsuperscript{b} Department of Computer Science, KAIST, Kusong-dong 373-1, Yusong-gu, Taejon 305-701, South Korea

Received 2 April 1993; revised 10 April 1995

Abstract

We present the results concerning the embedding of trees into recursive circulants. Recursive circulant $G(N,d)$ is a circulant graph with $N$ vertices and jumps of powers of $d$. We present dilation 1 embeddings of Fibonacci trees and full quaternary trees in $G(2^m,2)$, and full binary trees and binomial trees in $G(2^m,4)$.

1. Introduction

Recent advances in integrated circuit technology make it possible to construct very large interconnection networks. Together with these advances, many interconnection network topologies have been proposed and investigated in the literature. Interconnection networks are often modeled as graphs. Hypercubes are one of the most popular interconnection networks being used. A hypercube $Q_m$ of dimension $m$ is a graph with $2^m$ vertices labeled $\{0,1,\ldots,2^m-1\}$; two vertices are joined by an edge if and only if their binary representations differ in exactly one bit position.

One important consideration for a network topology is whether there exists good mappings from various kinds of trees to the topology. This is due to the fact that trees play an important role as data structures and as the computational graphs underlying divide-and-conquer algorithms. The mapping of a data structure or an interconnection structure into another has been studied as graph embedding. In particular, there have been many papers on embedding trees in hypercubes [2,9,12,13]. An embedding of a (guest) graph $G$ into a (host) graph $H$ is a one-to-one mapping $\phi$ of vertices of $G$ into vertices of $H$, combined with an assignment of each edge $e = (v,w)$ of $G$ to a path between $\phi(v)$ and $\phi(w)$ in $H$. One of the most important measures of the quality...
of an embedding $\phi$ is dilation. The dilation of an edge $e$ in $G$ under the embedding $\phi$ is the length of the path in $H$ to which $e$ is assigned, and the dilation of $\phi$ is the maximum dilation over all edges in $G$.

In this paper we consider the problem of mapping trees into the interconnection networks called recursive circulants. Recursive circulants are proposed in [10]. Recursive circulant $G(N,d)$, $d \geq 2$, has $N$ vertices labeled by integers between 0 and $N - 1$. Two vertices $v, w$ are joined by an edge if and only if there exists $i, 0 \leq i \leq \lfloor \log_d N \rfloor - 1$, such that $v + d^i \equiv w \pmod{N}$. $G(N,d)$ is a circulant graph with $N$ vertices and jumps of all powers of $d$ less than $N$. Examples of recursive circulants are given in Fig. 1.

Recursive circulants are a class of circulant graphs. Circulant graphs, also known as star-polygons [3], are a generalization of the graphs constructed by Harary [5] for achieving maximum connectivity. Every circulant graph is a vertex transitive graph and a Cayley graph. Also, circulant graphs are a class of generalized codal ring graphs [1], which is an important topology in interconnection networks.

Recursive circulants $G(2^m,2)$ and $G(2^m,4)$ are the major concern in this paper. $G(2^m,2)$, also known as a barrel shifter or PM21 [7], is a regular graph with degree $2m - 1$. Both $G(2^m,4)$ and $Q_m$ are subgraphs of $G(2^m,2)$. $G(2^m,4)$ has the connectivity $m$ and the diameter $\lceil (3m - 1)/4 \rceil$; which is less than $m$, the diameter of $Q_m$. Network metrics, such as connectivity, diameter, average distance, and visit ratio, for recursive circulants are analyzed in [10].

As an approach to embedding problems in recursive circulants, we introduce a graph labeling problem called $d$-edge labeling. A labeling on a graph $G(V,E)$ is a one-to-one mapping of the vertices $V$ into distinct integers $\{1, 2, \ldots, |V|\}$. Each edge of the graph has an edge label induced by the labeling. The edge label of an edge is the absolute difference between the labels of end-vertices of the edge. A $d$-edge labeling on a graph $G$ is defined to be a labeling such that the set of edge labels is a subset of $\{d^0, d^1, d^2, \ldots\}$ for some integer $d$, that is, each edge label should be a power of $d$. Clearly, if a graph $G$ has a $d$-edge labeling then $G$ is a subgraph of $G(N,d)$ with $N \geq |V(G)|$. Here we are interested in the case $d = 2$ or 4. We show that every Fibonacci tree and full quaternary tree has a 2-edge labeling and that every full binary
tree and binomial tree has a 4-edge labeling. The labeling results on the trees directly imply their dilation 1 embeddings into $G(2^n, 2)$ or $G(2^n, 4)$.

In Section 2, we give 2-edge labeling schemes on Fibonacci trees and full quaternary trees, and 4-edge labeling schemes on full binary trees and binomial trees. A summary of this paper and further remarks are given in Section 3.

2. Labeling schemes on trees

Terms and notation not defined here follow those used in [6]. Here we give 2-edge labeling schemes on Fibonacci trees and full quaternary trees, and 4-edge labeling schemes on full binary trees and binomial trees. For each type of tree, we first give an inductive labeling scheme, and then give the proof of the scheme.

2.1. 2-edge labeling on Fibonacci trees

Fibonacci trees are defined by the following.
(1) An empty tree and a tree consisting of a single vertex are Fibonacci trees and denoted by $FT_{-1}$ and $FT_0$, respectively.
(2) Fibonacci tree, $FT_k$ for $k \geq 1$, is a binary tree such that its left and right subtrees are $FT_{k-1}$ and $FT_{k-2}$, respectively.

$FT_k$ has the fewest possible vertices among all possible height balanced binary trees of height $k$ [8]. There is a search method involving only addition and subtraction based on the Fibonacci trees [8]. Examples of 2-edge labelings on some Fibonacci trees are shown in Fig. 2.

The following properties are known for the Fibonacci tree $FT_k$, or can be easily proved by induction on $k$.
(1) The number of vertices in $FT_k$ is $F_{k+3} - 1$, where $F_i$ is the $i$th Fibonacci number defined by the recurrence relation, $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.
(2) The number of leaves in $FT_k$ is $F_{k+1}$.

Fig. 2. 2-edge labelings on $FT_1$, $FT_2$, and $FT_3$. 
(3) Suppose we remove the leaves of FT\(k\) for \(k \geq 2\), then there remains FT\(_{k-1}\). And removing the leaves in FT\(_{k-1}\), there remains FT\(_{k-2}\). Let \(L_1(L_2)\) be the set of vertices in FT\(_k\) which become leaves of FT\(_{k-1}(FT_{k-2})\) by the removal. The leaves in FT\(_k\) are adjacent to the vertices in \(L_1 \cup L_2\) in a one-to-one fashion.

Our labeling scheme on FT\(_k\) satisfies the following properties.

(1) The label of the root is 1.

(2) The labels on the leaves of FT\(_k\) are consecutive integers, \(F_{k+2}, F_{k+2}+1, \ldots, F_{k+3} - 1\).

(3) The labeling of the tree obtained by removing leaves of FT\(_k\) is the 2-edge labeling of FT\(_{k-1}\).

In our labeling scheme, 2-edge labeling on FT\(_{k+1}\) is obtained by joining vertices to the 2-edge labeled FT\(_k\) and assigning consecutive integers to the vertices so that each edge label of the attached edge is a power of 2. The 2-edge labelings in Fig. 2 is obtained in this way. The following lemma deals with the general case of the assignment of integers.

**Lemma 1.** For each integer from 1 to \(n\), there is a matching of distinct integers from \(n + 1\) to \(2n\) such that the difference of each matched pair is a power of 2.

**Proof.** We prove the lemma by induction on \(n\). For \(n = 1\), the lemma holds by matching 2 to 1. Assume that for \(m\) less than \(n\), the lemma holds. Let \(2^{i-1} < n \leq 2^i\).

First we match \(2n\) to \(2n - 2^i\). Such a \(p = 2n - 2^i\) always exists between 1 and \(n\), because \(2^i < 2n \leq 2^{i+1}\). Note that \(p\) is the integer of which the difference from \(2n\) is the largest power of 2 in the range. Next, we match \(2n - 1, 2n - 2, \ldots, 2^i + 1\) to \(2n - 2^i - 1, 2n - 2^i - 2, \ldots, 1\), respectively. The remaining matching is between integers \(2n - 2^i + 1, 2n - 2^i + 2, \ldots, n\) and \(n + 1, n + 2, \ldots, 2^i\). By the induction hypothesis, there is a matching \(M\) between 1, 2, \ldots, \(2^i - n\) and \(2^i - n + 1, 2^i - n + 2, \ldots, 2^{i+1} - 2n\). By adding \(2n - 2^i\) to each integer in \(M\), we can obtain the remaining matching. Thus, the lemma holds. \(\square\)

Given the 2-edge labeled FT\(_k\) satisfying the previous labeling properties, the labeling scheme on FT\(_{k+1}\) is as follows:

(1) The labeling of the tree except for the leaves of FT\(_{k+1}\) is the same as the labeling of FT\(_k\).

(2) The set of labels which will be given to the leaves is \(\{F_{k+3}, F_{k+3} + 1, \ldots, F_{k+4} - 1\}\). The labels are considered in decreasing order and assigned to the unlabeled leaf \(v\) such that the edge label of the edge incident to \(v\) becomes the largest power of 2.

The correctness of the labeling scheme is shown in the following.

**Theorem 1.** Every Fibonacci tree has a 2-edge labeling.

**Proof.** The proof will be given by induction on \(k\), the height of a Fibonacci tree. For \(k \leq 3\), the theorem holds from the labelings given in Fig. 2. Also, the labelings satisfy
the previous labeling properties. Assume that for \( m \) less than \( k + 1 \), there exists a 2-edge labeling on \( FT_m \) satisfying the labeling properties. In the labeling scheme on \( FT_{k+1} \), the set of labels of the vertices adjacent to the leaves is \( \{F_{k+1}, F_{k+1} + 1, \ldots, F_{k+2} - 1\} \cup \{F_{k+2}, F_{k+2} + 1, \ldots, F_{k+3} - 1\} \). The labels in the set are consecutive integers, and the number of labels is \( F_{k+2} \). The labels given to the leaves are consecutive \( F_{k+2} \) integers from \( F_{k+3} \) to \( F_{k+4} - 1 \). The labeling to the leaves satisfying the constraint can be obtained by the method described in Lemma 1. It is evident that the labeling satisfies the previous properties. Thus, the theorem holds. □

**Corollary 1.** \( FT_k \) with vertices \( 2^m \) or less can be embedded in \( G(2^m, 2) \) with dilation 1.

### 2.2. 2-edge labeling on full quaternary trees

We denote by \( QT_k \) the full quaternary tree with height \( k \). Examples of 2-edge labelings on \( QT_1 \) and \( QT_2 \) are shown in Fig. 3.

\( QT_k \) has \( (4^k + 1)/3 \) vertices, so vertex labels \( 1, 2, \ldots, (4^k + 1)/3 \) are used. Our labeling scheme on \( QT_k \) satisfies the following properties:

1. The label of the root is \( (2 \cdot 4^k + 1)/3 \), that is, the middle number.
2. Let the four subtrees of the root be \( T_1, T_2, T_3, \) and \( T_4 \). Then the labels on \( T_1 \) and \( T_2 \) are \( \{1, 2, \ldots, (2 \cdot 4^k + 1)/3 - 1\} \) and the labels on \( T_3 \) and \( T_4 \) are \( \{(2 \cdot 4^k + 1)/3 - 1, \ldots, (4^{k+1} - 1)/3\} \).
3. Labelings of \( T_3 \) and \( T_4 \) are symmetric with those of \( T_2 \) and \( T_1 \), respectively. That is, the sum of the label of a vertex in \( T_1(T_2) \) and the label of the corresponding vertex in \( T_4(T_3) \) is \( (4^{k+1} + 2)/3 \), which is the double of the label of the root.

Thus, we only show the labeling scheme on the root and \( T_1 \) and \( T_2 \). We use two types of labeling patterns in a recursive manner. For convenience, we linearly order the vertices in consecutive positions instead of labeling them consecutively, and the position of the vertices are numbered from left to right starting from 1. Two types of linear ordering are defined by the following:

![Fig. 3. 2-edge labelings on QT1 and QT2.](image-url)
Definition 1. \( A(k) \)-type is a linear ordering of vertices of two \( QT_{k-1} \) such that the position numbers of \( u \) and \( v \) differ by a power of 2 for each edge \((u,v)\) and the positions of two roots of \( QT_{k-1} \)'s are \((2 \cdot 4^{k-1} + 1)/3\) and \((2 \cdot 4^k - 2)/3\). \( B(k) \)-type is defined in a similar way to \( A(k) \)-type but the root positions are different. The position numbers of the two roots are \((4^k - 1)/3\) and \((4^k - 1)/3 + 1\).

\( A(1) \)-type and \( B(1) \)-type are the same, and they are orderings of two trivial vertices. In Fig. 4, \( A(2) \)-type and \( B(2) \)-type are shown.

Once we have an \( A(k) \)-type, a 2-edge labeling on \( QT_k \) which satisfies the properties (1)–(3) can be obtained. By placing the root at the right of an \( A(k) \)-type and joining by edges between the root and the two roots of \( QT_{k-1} \)'s, we obtain a 2-edge labeling on the root and two subtrees of \( QT_k \). Note that the labels of two added edges are 1 and \( 2 \cdot 4^{k-1} \). The labeling of the remaining two subtrees can be obtained using the property (3).

Next, we construct \( A(k+1) \)-type and \( B(k+1) \)-type using \( A(k) \)-type, \( A(k) \)-type, and \( B(k) \)-type, where \( A(k) \)-type is the reverse ordering of \( A(k) \)-type. In Fig. 5, the construction scheme is specified. \( A(k+1) \)-type is constructed with two \( A(k) \)-types, one \( A(k) \)-type, one \( B(k) \)-type, and two additional vertices. Each rounded box represents the linear ordering of the type designated below. Small circles in it represent the two roots in each type. Two additional vertices, \( x \) and \( y \), become the two roots of \( QT_{k} \)'s, and they are joined by edges with roots in the types designated in Fig. 5. Now, the four types and the two vertices constitute a new linear ordering, and the position numbers of vertices are updated by numbering from left to right starting from 1.

\( B(k+1) \)-type is constructed with one \( A(k) \)-type, one \( A(k) \)-type, two \( B(k) \)-types, and two additional vertices \( u, v \), as in Fig. 5(h). The construction is similar to that of \( A(k+1) \)-type, thus the description of it is omitted.

The correctness of the construction is shown in the following theorem.

Theorem 2. Every full quaternary tree has a 2-edge labeling.
Proof. For the validity of the above construction with respect to Definition 1, it is sufficient to check the edge labels of added edges and the positions of two roots. Let the labels of four edges incident to a vertex \( x \) in Fig. 5 be \( l_1(x) \), \( l_2(x) \), \( l_3(x) \), and \( l_4(x) \). They can be easily calculated and summarized in Table 1. The number of vertices in \( A(k) \)-type and \( B(k) \)-type is \( 2(4^k - 1)/3 \), so the position numbers of \( u \), \( v \), \( x \), \( y \), and \( z \) in Fig. 5 are \( (2 \cdot 4^k + 1)/3 \), \( (2 \cdot 4^k - 2)/3 \), \( (4^k - 1)/3 \), and \( (4^{k+1} - 1)/3 + 1 \), respectively. From the above facts, the validity of the construction scheme is shown. Recursively applying the construction scheme, \( A \)-type and \( B \)-type of higher order can be obtained, and a 2-edge labeling on \( QT_k \) can be obtained from \( A(k) \)-type. Thus, the theorem holds.

### Table 1

<table>
<thead>
<tr>
<th>Edge labels of added edges in the construction scheme</th>
<th>( l_1() )</th>
<th>( l_2() )</th>
<th>( l_3() )</th>
<th>( l_4() )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>1</td>
<td>2 \cdot 4^k - 1</td>
<td>2</td>
<td>( 4^k )</td>
</tr>
<tr>
<td>( y )</td>
<td>1</td>
<td>2 \cdot 4^k - 1</td>
<td>4</td>
<td>2 \cdot 4^k</td>
</tr>
<tr>
<td>( u )</td>
<td>1</td>
<td>2 \cdot 4^k - 1</td>
<td>( 4^k )</td>
<td>4</td>
</tr>
<tr>
<td>( v )</td>
<td>1</td>
<td>2 \cdot 4^k - 1</td>
<td>4</td>
<td>( 4^k )</td>
</tr>
</tbody>
</table>

Corollary 2. \( QT_k \) with vertices \( 2^m \) or less can be embedded in \( G(2^n, 2) \) with dilation 1.
2.3. 4-edge labeling on full binary trees

A 2-edge labeling of full binary trees can be found in [2], where the labeling is given by assigning successive integers according to the inorder traversal sequence of full binary trees. Using preorder traversal and postorder traversal, we can easily obtain 2-edge labelings on full binary trees.

Let $BT_k$ be the full binary tree with height $k$. Examples of 4-edge labelings on some $BT_k$'s are shown in Fig. 6.

Our labeling scheme for $BT_k$ will be presented in a similar way to the labeling scheme on $QT_k$. First, the three basic types used in the labeling scheme are defined, and we construct the types of higher order using these three types. $BT_k$ has $2^{k+1} - 1$ vertices, so labels $1, 2, \ldots, 2^{k+1} - 1$ are used. Our labeling scheme on $BT_k$ satisfies the following properties:

(1) The label of the root is $2^k$, that is, the middle number.
Let $T_r$ and $T_l$ be the left subtree and the right subtree of the root, respectively. The labeling on $T_r$ is symmetric with the labeling on $T_l$, that is, the sum of the label of a vertex in $T_r$ and the label of the corresponding vertex in $T_l$ is $2^{k-1}$.

(3) In the case of an odd $k$, the labeling on $T_r$ is the same as the labeling on $BT_{k-1}$.

(4) In the case of an even $k$, the labels on $T_r$ are $\{1, 2, \ldots, 2^{k-1} - 1\} \cup \{2^k + 1, \ldots, 3 \cdot 2^{k-1}\}$.

We specify the labeling scheme on $BT_k$ only in the case of an even $k$. The labeling on $BT_k$ for odd $k$ can be obtained from the labeling on $BT_{k-1}$ which satisfies the above properties, and will be discussed later. We further restrict our attention to the labeling on the right subtree of $BT_k$. As described in property (4), the labels in the right subtree are not consecutive integers. They consist of two sets of consecutive integers. For convenience, the labeling on $T_r$ of $BT_k$ is represented by a pair of separated linear orderings of vertices. They are depicted in the form of Fig. 7. The $V_L$ and $V_R$ are sets of vertices on the left side and right side of Fig. 7, respectively. The difference of position numbers between a vertex in $V_L$ and the horizontally corresponding vertex in $V_R$ is $2^k$.

Now, we define three types used in the labeling scheme.

**Definition 2.** $X(k)$-type is a pair of linear orderings of vertices in $BT_{k-1}$ such that $|V_L| = |V_R| = 2^{k-1} + 1$ and the difference between position numbers of $u$ and $v$ is a power of 4 for each edge $(u, v)$. In this type, vertices in positions 0, $2^{k-1}$, and $2^k$ are not used and the position number of the root is $5 \cdot 2^{k-2}$, that is, the middle of $V_R$.

Examples of $X(k)$-types are shown in Fig. 8. Double circles are the roots in their types.

Once we have an $X(k)$-type, 4-edge labeling of $BT_k$ can be obtained. Labeling the root $2^k$ and joining the root and the vertex in position $5 \cdot 2^{k-2}$ by an edge, we obtain a labeling of the root and $T_r$ of $BT_k$. The labeling for $T_r$ is obtained from the procedure described in property (2). For an odd $k$, the label of the root $r$ of $BT_k$ is $2^k$. Vertex $r$ is joined with the root of 4-edge labeled $BT_{k-1}$ which becomes $T_r$. Also, the labeling on $T_r$ is obtained from the procedure described in property (2). The 4-edge labelings in Fig. 6 are obtained in this way.

```
<table>
<thead>
<tr>
<th>position</th>
<th>$V_L$</th>
<th>$V_R$</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$\bigcirc$</td>
<td>$\bigcirc$</td>
<td>$2^k$</td>
</tr>
<tr>
<td>$1$</td>
<td>$\bigcirc$</td>
<td>$\bigcirc$</td>
<td>$2^k + 1$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$2^{k-1}$</td>
<td>$\bigcirc$</td>
<td>$\bigcirc$</td>
<td>$3 \cdot 2^{k-1}$</td>
</tr>
</tbody>
</table>
```

Fig. 7. Pair of linear orderings of vertices.
For the construction of $X(k)$-type of higher order, the following two types are defined.

**Definition 3.** $Y(k)$-type and $Z(k)$-type are the same as the $X(k)$-type, except for the unused positions. The unused positions in $Y(k)$-type are $0$, $2^{k-2}$, and $2^{k-1}$, and the unused positions in $Z(k)$-type are $0$, $2^{k-1} - 1$, and $2^{k-1}$.

Examples of $Y(k)$-type and $Z(k)$-type are shown in Fig. 9.

$X(k + 2)$-type is constructed by combining $X(k)$-type, $Y(k)$-type, $Z(k)$-type, and $Y'(k)$-type, which is the modification of $Y(k)$-type. We describe the construction scheme by the pictorial representation in Fig. 10.
In Fig. 10, dotted boxes, circles and double circles represent types, unused vertices and roots in the types, respectively. The four types are ordered as in the figure, and the vertex contained in the intersection of two types is an unused vertex in one or both of the two types. Whether they are used in one type or the other type will be explained after the following description of $Y'(k)$-type.

$Y'(k)$-type is similar to $Y(k)$-type but is different in one edge. In $Y'(k)$-type the vertex in position $3 \cdot 2^{k-1} - 1$ is joined by an edge to an unused vertex in position $2^k - 1 - 1$ of $Z(k)$-type other than the vertex in position $3 \cdot 2^{k-1}$. In Fig. 10, the vertex $v$, depicted by the square in $Y'(k)$-type, is joined by an edge to $z$ which is the unused vertex in $Z(k)$-type. Vertex $w$ becomes an unused vertex in $Y'(k)$-type and used in $Y(k)$-type in the below. Except for $u'$, each vertex in the intersection of two types is used in the type containing it in $V_R$.

The $BT_{k+2}$ in $X(k + 2)$-type is constructed by using the four types of $BT_k$ in the following way.
(1) The vertex $r$, which is the unused vertex both in $Y(k)$-type and $Y'(k)$-type, becomes the root of $B_T^{k+2}$. The vertex $r$ is joined to $x$, which is the unused vertex in the middle of $V_L$ of $Y'(k)$-type, and joined to $y$, which is the unused vertex in the middle of $V_L$ of $Y(k)$-type.

(2) The vertex $x$ is joined to the root of $Y'(k)$-type and to the root of $Z(k)$-type. The vertex $y$ is joined to the root of $X(k)$-type and to the root of $Y(k)$-type.

Now the position numbers of vertices are updated from the uppermost vertex in the left part starting from 0 and from the uppermost vertex in the right part starting from $2^{k+2}$, respectively.

Constructions of $Y(k + 2)$-type and $Z(k + 2)$-type are depicted in Figs. 11 and 12. The bar above the type name indicates the ordering is reversed, that is, upside down. Their descriptions are similar to that of $X(k + 2)$-type, and thus omitted.

The correctness of the construction is shown in the following.
**Theorem 3.** Every full binary tree has a 4-edge labeling.

**Proof.** For the validity of the construction of $X(k+2)$-type with respect to Definition 2, it is sufficient to check the edge labels of added edges, the position of new root $r$, and the positions of unused vertices. Unused vertices in $X(k+2)$-type are the uppermost and the lowermost vertices in the left part and the uppermost vertex in the right part in the figure, which is valid under the definition. Let $l(e_i)$ be the edge labels of the edges $e_i$, $1 \leq i \leq 7$, in Fig. 10. From the size of each type and from the positions of the unused vertices or root in each type, they can be easily calculated and summarized in Table 2. The edge labels of the added edges in $Y(k+2)$-type and $Z(k+2)$-type are similar to those of $X(k+2)$-type. Recursively applying the construction scheme, $X$-type, $Y$-type, and $Z$-type of higher order can be obtained. A 4-edge labeling on $BT_k$ for an even $k$ can be obtained from $X(k)$-type. Also, a 4-edge labeling on $BT_k$
for an odd \( k \) can be obtained from the 4-edge labeling of \( BT_{k-1} \). Thus, the theorem holds. \( \square \)

| Table 2 |
|-----------------|-----------------|
| **Edge labels of added edges in the construction scheme** | **\( l() \)** |
| \( e_1, e_2 \) | \( 2^{k-2} \) |
| \( e_3, e_4, e_7 \) | \( 2^k \) |
| \( e_5, e_6 \) | \( 2^k + 2 \) |

**Corollary 3.** \( BT_k \) with vertices \( 2^m \) or less can be embedded in \( G(2^m, 4) \) with dilation 1.

### 2.4. 4-edge labeling on binomial trees

Binomial trees play an important role in broadcasting messages in parallel networks and implementing mergeable priority queues [4]. Binomial trees are defined by the following.

1. A tree consisting of a single vertex is a binomial tree denoted by \( B_0 \).
2. Suppose that \( T_r \) and \( T_l \) are disjoint \( B_{k-1} \) for \( k \geq 1 \). The tree obtained by adding an edge to make the root of \( T_r \) become the leftmost child of the root of \( T_l \) is the binomial tree \( B_k \).

It follows from the definition that \( B_k \) has \( 2^k \) vertices. Some examples of binomial trees and the 4-edge labeling on them are shown in Fig. 13.

In the labeling scheme on \( B_k \), we have two cases depending on whether \( k \) is even or odd. The labeling scheme on \( B_{k+1} \) is as follows.

1. In the case of an even \( k \), the labeling on \( T_r \) of \( B_{k+1} \) is the same as the labeling on \( B_k \) and the label of a vertex in \( T_r \) of \( B_{k+1} \) is obtained by adding \( 2^k \) to the label of the corresponding vertex in \( B_k \).
2. In the case of an odd \( k \), first, the labeling on \( B_k \) is obtained by (1). Let the labeled tree be \( T' \). The label of the vertex in \( T_r \) of \( B_{k+1} \) is obtained by adding \( 2^k \) to the label of the corresponding vertex in \( T' \). The label of each vertex in \( T_r \) of \( T' \) is interchanged with the label of the corresponding vertex in \( T_r \) of \( T' \). The resulting labeled tree \( T' \) constitutes \( T_r \) of \( B_{k+1} \).

**Theorem 4.** Every binomial tree has a 4-edge labeling.

**Proof.** To prove the validity of the above labeling scheme, only the edge label of the edge joining \( T_r \) and \( T_l \) need to be shown. In the case of an even \( k \), we add \( 2^k \) to the
labels of the vertices in $T_r$, thus the edge label is $2^k$ for some even $k$. Let $l_k$ be the label of the root of $B_k$. In the case of an odd $k$, $l_{k+1} = l_k + 2^{k-1}$ and the label of the root of $T_r$ is $l_k + 2^k$. The edge label of the edge joining $T_r$ and $T_i$ is $2^{k-1}$ for some odd $k$. Thus, the theorem holds. □

**Corollary 4.** $B_k$ with vertices $2^m$ or less can be embedded in $G(2^m, 4)$ with dilation 1.

3. Concluding remarks

In this paper we have investigated the embedding trees in recursive circulants. We introduced the $d$-edge labeling problem to attack the embedding problems and presented 2- and 4-edge labeling schemes on some trees. In the labeling schemes on $QT_k$ and $BT_k$, we used some labeling patterns in a recursive way. Other labelings can be obtained by modifying the inner constructions of the types with the positions of the roots in the types unchanged. The labeling schemes given in this paper can be employed to design labeling algorithms which run in linear time to the number of vertices.
If $\pi_1$ and $\pi_2$ are $d$-edge labelings on $G_1$ and $G_2$, respectively, and $|V_1|$ or $|V_2|$ are some powers of $d$, then $d$-edge labeling of graph product $G_1 \times G_2$ can be easily obtained. Suppose that $|V_2|$ is a power of $d$, then the labeling scheme that assigns the label $(\pi_1(u) - 1)|V_2| + \pi_2(v)$ to the vertex $uv$, $u \in V_1, v \in V_2$, in $G_1 \times G_2$ achieves $d$-edge labeling on $G_1 \times G_2$.

2-edge labeling problems on other trees and graphs, such as full ternary trees, arbitrary binary trees, and meshes remain open. Some examples of the labelings on them are given in Fig. 14.

There have been many papers dealing with the embedding of the arbitrary binary trees into $Q_m$ [2, 12, 13]. Dilation 2 embedding of arbitrary binary tree with $2^m$ vertices or less into $Q_m$ is one of the long-standing open problems. Recently, embeddings among hypercubes and recursive circulants are investigated in [11]. One of the main results is that $G(2^m, 2)$ can be embedded into $Q_m$ with dilation 2. The 2-edge labeling problem of arbitrary binary trees is a more restricted problem than dilation 2 embedding of it into $Q_m$ or into $G(2^m, 4)$. But, 2-edge labeling can be a simple and good approach to attacking the embedding problems.
References