EMBEDDING PYRAMIDS IN ARRAY PROCESSORS WITH PIPELINED BUSSES

Zicheng Guo and Rami G. Melhem

Departments of Electrical Engineering and Computer Science
The University of Pittsburgh
Pittsburgh, PA 15260

Abstract

The unidirectional propagation of optical signals in waveguides allows for the construction of pipelined optical busses on which many processors may write their messages simultaneously. In [5], a multiprocessor system architecture has been proposed based on a two dimensional array of processors connected by horizontal and vertical pipelined busses, and efficient interprocessor communications have been presented for it. In this paper we present an efficient embedding of pyramids onto this architecture. The embedding has the property that neighboring nodes in the pyramid are mapped to the same bus, thus allowing any two neighbors in the embedded pyramid to communicate with each other using a single bus cycle.

1. Introduction

Mesh computers are among the most promising parallel architectures in image processing and computer vision [10, 12], and have been extensively studied. Meshes, however, are inefficient in global interprocessor communications because of their large communication diameter. For this reason, approaches have been considered to augment the communication capabilities of meshes with busses [1, 14, 15]. Although this approach decreases the communication diameter, it does not substantially increase the communication bandwidth because of the *exclusive access* property of electronic busses. Specifically, if n processors are connected to a bus, only one processor is allowed to write a message on the bus at any given time. This disadvantage may be overcome if optical busses are used instead of electronic busses. In this case, all n processors can write their respective messages on the bus *simultaneously*, and all the messages will then travel on the bus in a pipelined fashion [3, 9]. This is possible because optical signals propagate unidirectionally in waveguides.

In order to describe the operation of pipelined optical busses, we consider Fig 1.1(a) which shows a linear array of n processors connected by an optical bus (waveguide). Each processor is coupled to the bus with two couplers [8], one for injecting (writing) signals, and the other for receiving (reading) signals. Assume that a message on an optical bus consists of a sequence of optical pulses, each having a width (or duration) w in seconds. The existence of an optical pulse of length w represents a binary bit 1, and the absence of such a pulse represents a 0. For analytical convenience, we let D_o be the optical distance between each pair of adjacent nodes and τ be the time taken for an optical signal to traverse the optical distance D_o. As in the case of

CH2920-7/90/0000/0665$01.00 © 1990 IEEE

electronic busses, each node j communicates with any other node i by sending a message to i through the common bus. However, because optical signals propagate in one direction, a node j in the system of Fig 1.1(a) may send signals to another node i only if $i > j$. To transfer a message from a node j to node i, $i>j$, the sending node j writes its message on the bus. After a time $(i-j)\tau$, the message will arrive at the receiving node i, which then reads the message from the bus.

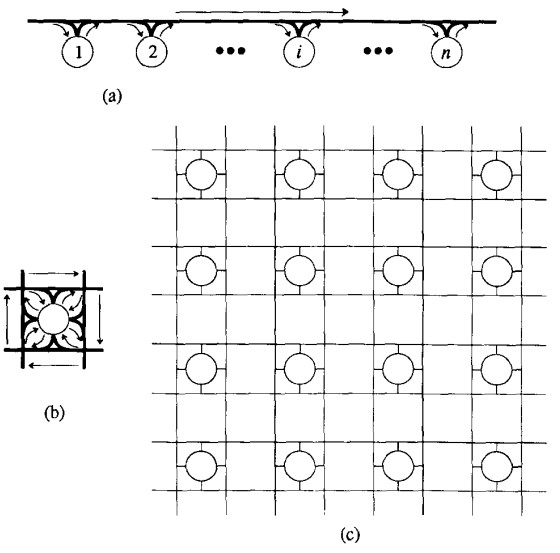

Fig 1.1. Array processors with pipelined busses (APPB). (a) A linear array connected with a single optical bus. (b) A node coupled to four busses (c) A schematic drawing of 2-D APPB with each node coupled to four busses as shown in (b).

Unlike the case of an electronic bus, where writing access to the bus is *exclusive*, all the processors may send their messages on the bus *simultaneously* if every node writes its message at the same instant and the length of each message is smaller than the optical distance between any two consecutive processors. Let b be the maximum number of bits in each message, w, as before, be the width (duration) of the pulse used to represent one bit, and c_g be the speed of light in the waveguide. Then we may ensure that messages sent by distinct processors do not collide with one another if the following collision-free condition is satisfied

$$D_o > bwc_g .$$

Here by colliding we mean two opitcal signals injected on the bus by any two distinct nodes arrive at some point on the bus simultaneously. With this collision-free condition satisfied, every node can, in parallel, send a message to some other node, and the messages will all travel from

left to right in a pipelined fashion, thus the term *pipelined bus*. Note that the requirement for synchronized message generation is restrictive but it can be met in several ways. An optically distributed clock can be broadcast without skew to each node [4] or electro-optical switches [13] can be used in place of sources to "switch in" pulses generated from a single source.

In cases where the communication pattern is known, a *wait* register in each processor may be programmed such that it indicates the number of messages that a processor has to skip before reading the message destined to it. Alternatively, the *wait* register may indicate the time, relative to the beginning of a bus cycle, at which the processor should read its message.

To show how message pipelining works, let us look at a simple example where each processor wants to send a message to the k^{th} processor (if it exists) to its right. If all processors start injecting their messages at the beginning of a bus cycle, then all the messages will travel on the bus in a pipelined fashion without collision. The control function $wait(i)$ is defined at each processor i such that processor i is to receive a message from processor $i-k$. Then $wait(i) = (i - (i - k))\tau = k\tau$. If τ is considered as a unit time, we can simply write $wait(i) = k$. That is, each processor i must read its message from the bus after k time units from the beginning of the bus cycle. In this way a simple permutation, $perm(j) = j + k$, has been realized in a single bus cycle. Note that the bus shown in Fig 1.1(a) supports only message transfer from left to right, and that a second bus should be added to support message transfer from right to left, that is, from a processor j to a processor i, $i < j$. In general, by using two optical busses, one for message transfer in each direction, arbitrary permutations may be realized in one bus cycle, using only $O(n)$ hardware [9].

A two-dimensional array with horizontal and vertical optical busses, called *Array Processors* with *Pipelined Busses* (APPB), has been proposed in [5]. In this architecture each node is connected to four busses, as shown in Fig 1.1(b), and the entire array is schematically drawn as in Fig 1.1(c). Many efficient interprocessor communication schemes have been suggested for the APPB, and it has been shown that the communication bandwidth of the APPB is substantially higher than conventional parallel architectures based on nearest neighbor and exclusive access bus interconnections [5, 6]. Thus it is of substantial interest to consider how we may take advantage of the APPB architecture to accomplish more parallel processing tasks in an efficient way. In particular we present in this paper an efficient embedding of pyramids, which, like meshes, are another very promising architecture for image processing and computer vision [2, 16]. Our embedding of pyramids in the APPB has the property that all the neighboring nodes in a pyramid are mapped to the same bus in the APPB, thus allowing any two neighbors in the embedded pyramid to communicate with each other using a single bus cycle. With such an embedding, all algorithms designed for pyramids can be efficiently executed on the APPB.

The dilation cost is an important measure used to evaluate the quality of embeddings of any source graph $S = \{V_s, E_s\}$ with a set of nodes V_s and a set of edges E_s into a target mesh with nearest neighbor connections. Specifically, the dilation of an edge $(u, v) \in E_s$, which is mapped to a path Q in the target mesh, is $|Q| - 1$, where $|Q|$ is the number of processors on Q. This measure, however, is of little value when S is mapped to an APPB because the bus connections in the APPB will allow processors u and v to communicate in one bus cycle if Q is either a horizontal path or a vertical path, and in two bus cycles, otherwise. Hence, a good embedding of S into the target APPB should map every two neighbors in S onto the same row or the same column in the APPB. A mapping which satisfies this condition will be said to satisfy the *alignment condition*. As will be seen, the pyramid embedding presented in Section 3 will satisfy the alignment condition. We start, however, by introducing a new coding scheme, called the reflection code, for meshes, which will be used to define our pyramid embedding.

2. The Reflection Code

In this section we define a coding scheme, called the reflection code, for meshes. This code will be used to define our embedding of pyramids in APPB since each level of a pyramid is a mesh. The coding scheme is a two step scheme, which uses two transforms, $G(\,)$ and $R(\,)$. The first transform, $G(\,)$, results in a Gray code [7], to which the second transform, $R(\,)$, is then applied to obtain the reflection code. $G(\,)$ is defined as follows. Let s be an integer with binary representation $s_{k-1} \cdots s_0$. Then

$$G(s_{k-1}s_{k-2} \cdots s_1s_0) = t_{k-1}t_{k-2} \cdots t_1t_0 \tag{1}$$

where $t_i = s_{i+1} \oplus s_i$ for $i = 0, \cdots, k-2$, $t_{k-1} = s_{k-1}$, and $\oplus$ is the logical Exclusive-OR operator.

Now consider a mesh of size $2^k \times 2^k$. The row/column position of each node in this mesh is given by (x, y), where $0 \le x, y < 2^k$. If $x_{k-1} \cdots x_0$ and $y_{k-1} \cdots y_0$ are the binary representations of x and y, respectively, then the row major index for (x, y) is $z = xy = x_{k-1} \cdots x_0y_{k-1} \cdots y_0$ (see Fig 2.1(a)). The Gray code for (x, y) is given by (e, f) where

$$(e, f) = (G(x), G(y)). \tag{2}$$

The Gray code index for (x, y) is then defined as $g = ef = e_{k-1} \cdots e_0f_{k-1} \cdots f_0$, that is, the Gray code of x concatenated with the Gray code of y (see Fig 2.1(b)).

0 (0,0)	1 (0,1)	2 (0,2)	3 (0,3)
4 (1,0)	5 (1,1)	6 (1,2)	7 (1,3)
8 (2,0)	9 (2,1)	10 (2,2)	11 (2,3)
12 (3,0)	13 (3,1)	14 (3,2)	15 (3,3)

(a)

0 (0,0)	1 (0,1)	3 (0,3)	2 (0,2)
4 (1,0)	5 (1,1)	7 (1,3)	6 (1,2)
12 (3,0)	13 (3,1)	15 (3,3)	14 (3,2)
8 (2,0)	9 (2,1)	11 (2,3)	10 (2,2)

(b)

0 (0,0)	1 (0,1)	5 (1,1)	4 (1,0)
2 (0,2)	3 (0,3)	7 (1,3)	6 (1,2)
10 (2,2)	11 (2,3)	15 (3,3)	14 (3,2)
8 (2,0)	9 (2,1)	13 (3,1)	12 (3,0)

(c)

Fig 2.1. Indexing schems: (a) Row major index z and row/column position (x, y). (b) Gray code index g and Gray code (e, f). (c) The reflection code index r and the reflection code (p, q).

If k is even, then the reflection code (p, q) can be defined by a transform $R(\,)$ applied to the Gray code (e, f) as follows:

$$(p, q) = R[(e, f)] = (e_{k-1}f_{k-1}e_{k-3}f_{k-3} \cdots e_1f_1, e_{k-2}f_{k-2}e_{k-4}f_{k-4} \cdots e_0f_0) \tag{3}$$

It is easy to show that transform $R(\,)$ belongs to the class of BPC permutations [11]. Although this definition is given for k being even, the reflection code for a mesh of size $2^j \times 2^j$, where j is odd, can also be defined. In this case the binary representations of e and f are both augmented with a 0 at the highest bit. That is, in definition (3) we assume $k = j + 1$, and $e_{k-1} = 0$ and $f_{k-1} = 0$. We may then augment the original $2^j \times 2^j$ mesh to a size $2^{j+1} \times 2^{j+1}$. After applying the transform $R(\,)$, the reflection code for our original $2^j \times 2^j$ mesh is obtained in the first quadrant of the augmented mesh.

Thus given a node (x, y) in a mesh of size $2^k \times 2^k$, its reflection code is obtained by

$$(p, q) = R[(G(x), G(y))] \tag{4}$$

The reflection code index of (x, y) is then given by $r = pq$ (see Fig 2.1(c)).

If k is even, it is possible to define the reflection code in a modular, recursive way. That is, instead of encoding each node, the recursive approach encodes each block of nodes of successively larger sizes in a 4×4 mesh of these blocks. Specifically, a mesh of size $2^{2i+2} \times 2^{2i+2}$ is considered as a 4×4 mesh of blocks each of size $2^{2i} \times 2^{2i}$. If $(e_{2i+1} \cdots e_0, f_{2i+1} \cdots f_0)$ is the gray code of a node in the mesh, then $(e_{2i+1}e_{2i}, f_{2i+1}f_{2i})$ is considered to be the gray code of the $2^{2i} \times 2^{2i}$ block that contains the node, and $(e_{2i-1} \cdots e_0, f_{2i-1} \cdots f_0)$ is considered to be the gray code of the node within the block. The reflection code of each $2^{2i} \times 2^{2i}$ block in the 4×4 mesh of such blocks is determined by a block transform defined as follows:

$$R(e_{2i+1}e_{2i}, f_{2i+1}f_{2i}) = (e_{2i+1}f_{2i+1}, e_{2i}f_{2i}).$$

Now, to find the reflection code of $(e_{2k+1} \cdots e_0, f_{2k+1} \cdots f_0)$, we start with $i = 0$, that is, each block containing a single node. We then apply the block transform to successively larger i until the desired mesh size is reached. This definition reveals the recursive patterns of the reflection code, as shown in Fig 2.2.

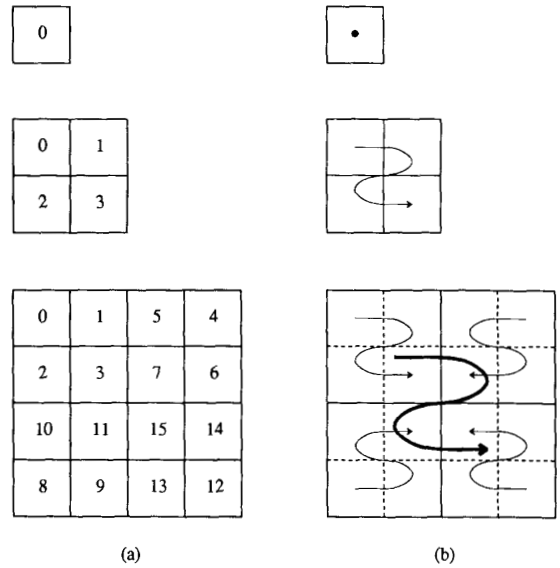

Fig 2.2. Recursive patterns of the reflection code index.
(a) Numerical representation. (b) Graphical representation.

The reflection code index gets its name from the fact that it can be obtained through successive column and row reflections as seen from the reflexive relation among the four shorter arrowed curves in Fig 2.2(b). This code has the following properties:

1) *Squaring effect*: Nodes of indices r such that $0 \le r < 2^{4i}$ are arranged in a $2^{2i} \times 2^{2i}$ square area at the upper-left corner of the mesh.
2) *Adjacency*: The reflection codes for two adjacent nodes (x, y) and $(x, y+1)$ or $(x+1, y)$ are at Hamming distance 1. This will become clear from a Lemma in the next section.

It is interesting to compare the reflection code index with two well known indexing schemes: the Gray code index and the shuffled row major index [17]. Like the reflection code index, the Gray code index also possesses the adjacency property, but it does not have the squaring effect. On the contrary, the shuffled row major index also has the squaring effect, it, however, does not possess the adjacency property. Thus neither the Gray code index nor the shuffled row major index has both squaring effect and adjacency properties, which are both crucial to our pyramid embedding presented in the next section.

3. Embedding Pyramids in Array Processors with Pipelined Busses

As mentioned at the end of Section 1, in an embedding of a source graph into the APPB if every pair of neighboring nodes in the source are mapped to the same bus in the APPB, then the embedding is said to satisfy the alignment condition. In this section we first define specific alignment conditions for pyramid embeddings, and then present our pyramid embedding and show that it satisfies the alignment conditions.

3.1. Alignment Conditions for Pyramid Embeddings

Consider a pyramid of L levels with the apex at level 0. Level l, $0 \le l \le L-1$, of the pyramid can be viewed as a mesh of size $2^l \times 2^l$. To each node at level l we assign a label (x, y, l), where (x, y), $0 \le x, y < 2^l$, is the row/column position of a node at level l. As a result, the four children of a node (x, y, l), $0 \le l \le L-2$, have labels $(2x, 2y, l+1)$, $(2x+1, 2y, l+1)$, $(2x, 2y+1, l+1)$ and $(2x+1, 2y+1, l+1)$, respectively. Equivalently, if the binary representations of x and y are $x_{l-1} \cdots x_0$ and $y_{l-1} \cdots y_0$, respectively, then the children of (x, y, l) are $(x_{l-1} \cdots x_0 X_0, y_{l-1} \cdots y_0 Y_0, l+1) = (xX_0, yY_0, l+1)$, where X_0 and Y_0 are either 0 or 1. For convenience we may also refer to node (x, y, l) by (e, f, l) or (p, q, l), where (e, f) and (p, q) are the Gray code and the reflection code of (x, y), respectively. In the case where the level number l in the label of a node is not of interest, we may omit l and simply write (x, y), (e, f) or (p, q).

Given the above labeling scheme, a mapping of the pyramid to the APPB satisfies the alignment condition if the following are satisfied:

C1) All the neighboring nodes at the same level, i.e., (x, y, l) and $(x, y+1, l)$ or (x, y, l) and $(x+1, y, l)$, are mapped to either the same row or the same column in the APPB.

C2) Each node (x, y, l) and its four children $(xX_0, yY_0, l+1)$, $X_0, Y_0 = 0, 1$, are mapped to the same row or column in the APPB. This condition will be satisfied if the following are met

- C2.1) Nodes $(xX_0, yY_0, l+1)$, $X_0, Y_0 = 0, 1$, are mapped to the same row or column.
- C2.2) If $(xX_0, yY_0, l+1)$ are mapped to row i of the APPB, then (x, y, l) is also mapped to row i of the APPB. Similarly, if $(xX_0, yY_0, l+1)$ are mapped to column j of the APPB, then (x, y, l) is also mapped to column j.

In terms of the above conditions, we will present an embedding which maps the nodes of each level, l, of the pyramid such that C1) and C2.1) are satisfied, and maps the nodes of two consecutive levels, l and $l+1$, such that C2.2) is satisfied.

3.2. An Embedding Satisfying the Alignment Conditions

The embedding of an L–level pyramid onto an APPB (the target APPB) is obtained by mapping each level l, $0 \le l < L$, (source meshes) of the pyramid into a square or rectangular area (a target mesh) in the target APPB. Specifically if l is even, then the target mesh is of size $2^l \times 2^l$ (of the same size as the source mesh); while if l is odd, then the target mesh is of size $2^{l+1} \times 2^{l-1}$ (see Figure 3.1(a)). The nodes in each level are mapped to a target mesh, and the L resulting target meshes are then properly positioned to form the target APPB. For odd L, the resulting APPB is a square array, while for even L the resulting APPB is a rectangular array. In the remainder of this section, we will simplify our formulas by assuming that L is odd. The formulas for the case of even L are slightly different and may be obtained in a similar manner. Specifically, for odd L, a node (x, y, l) of the pyramid is mapped to node

$$P(x, y, l) = \begin{cases} (p + \alpha_l, q + \beta_l), & l = even \\ (q + \alpha_l, p + \beta_l), & l = odd \end{cases} \tag{5}$$

in the target APPB, where $(p, q) = R(G(x), G(y))$ is the reflection code of (x, y), as defined in Equation (4), and

$$\beta_l = \begin{cases} 0, & l = L-1 \\ \beta_{l+1} + 2^{l+1}, & l = odd \\ \beta_{l+1}, & l = even \end{cases}$$

$$\alpha_l = \begin{cases} 0, & l = L-1 \\ \alpha_{l+1}, & l = odd \\ \alpha_{l+1} + 2^{l+2}, & l = even \end{cases}$$

That is, node (x, y, l) in the pyramid is mapped to a node at row/column position $(p + \alpha_l, q + \beta_l)$ in the APPB. In other words, a node (x, y) on level l of the pyramid is mapped to position (p, q) in a $2^l \times 2^l$ mesh if l is even, or a $2^{l+1} \times 2^{l-1}$ mesh if l is odd. The origin of this mesh is, then, shifted to position (α_l, β_l) of the target APPB. The recursive formulas for β_l and α_l have the solution

$$\beta_{L-1} = \alpha_{L-1} = \alpha_{L-2} = 0,$$

$$\beta_l = \beta_{l-1} = \alpha_{l-1} = \alpha_{l-2} = \sum_{j=1}^{(L-l)/2} 2^{L-2j+1}, \qquad l = L-2, L-4, \cdots, 1$$

An example of embedding P is shown in Fig 3.2.

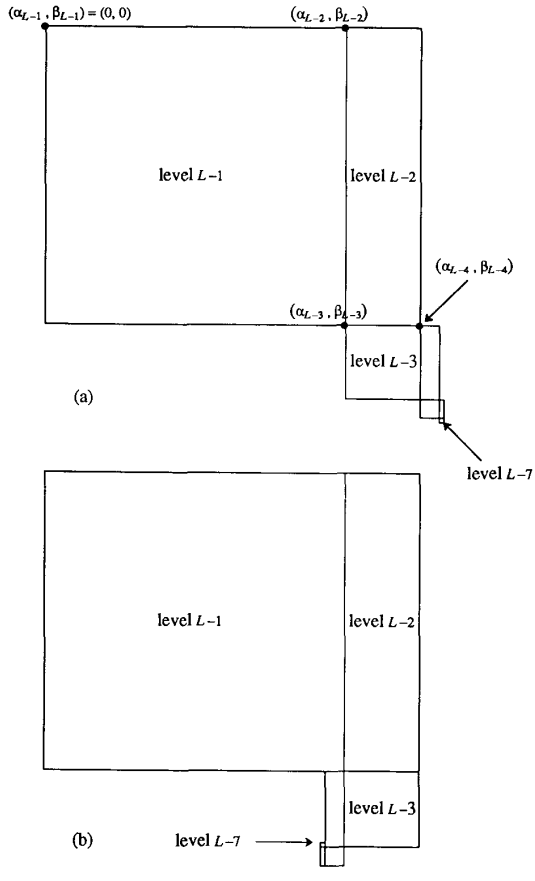

Fig 3.1. Embeddings of an L-level pyramid, L odd, obtained by mapping each level of the pyramid into a square or rectangular area in APPB.

In order to prove that the embedding P in Equation (5) satisfies the alignment conditions, we start by proving the following well known result [7].

Lemma: $G(s)$ and $G(s+1)$, where $G(\,)$ is as defined in (1), differ at exactly one bit in their binary representations, i.e., the Gray codes of s and $s+1$ are at Hamming distance 1.

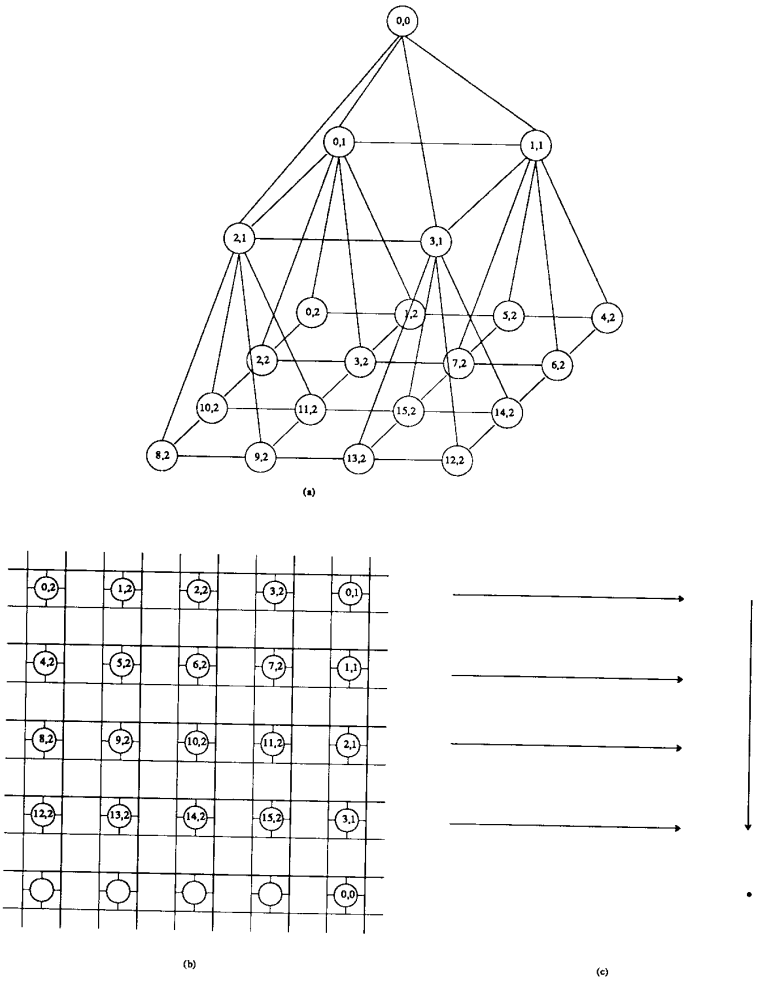

Fig 3.2. An embedding of pyramids. (a) A pyramid with the two numbers in each node indicating its reflection code index r and level number l, respectively. (b) Numerical presentation of the embedding. (c) Graphical presentation (compare with Fig 2.2(b)).

Proof: Let b^k be a string of b's of length k for some integer $k \geq 0$, and be an empty string if $k = 0$. Any integer s can be put in the form $s = ds_{i+1}01^i$, where $d = s_{k-1}s_{k-2} \cdots s_{i+2}$, $s_i = 0$, and $s_{i-1}s_{i-2} \cdots s_0 = 11 \cdots 1$. Then $s + 1 = ds_{i+1}10^i$. We have $G(s) = hc_i 10^{i-1}$, where $h = G(ds_{i+1})$ and $c_i = s_{i+1} \oplus 0$, and $G(s+1) = hC_i 10^{i-1}$, where $C_i = s_{i+1} \oplus 1$. Clearly $C_i \neq c_i$, and thus $G(s)$ and $G(s+1)$ differ in exactly one bit. This completes the proof.

This Lemma tells us that, in a mesh, the Gray codes of two neighboring nodes, say (x, y) and $(x, y+1)$ or (x, y) and $(x+1, y)$, are at Hamming distance 1. Since the reflection codes for (x, y) and $(x, y+1)$ or $(x+1, y)$ are obtained by permuting the binary bits of their Gray codes, respectively, the reflection codes for any two neighboring nodes are also at Hamming distance 1. This is the adjacency property mentioned in the previous section. The Lemma simplifies the proof of the following Proposition.

Proposition: The embedding P in Equation (5) satisfies alignment conditions C1) and C2).

Proof: Consider a node (x, y, l) in the pyramid. Our proof will be given for l being even and the case for l being odd follows similarly. Let $P(x, y, l) = (p + \alpha_l, q + \beta_l)$, where $(p, q) = R[(e, f)]$ and $(e, f) = (G(x), G(y))$, and let $P(x, y+1, l) = (\bar{p} + \alpha_l, \bar{q} + \beta_l)$ where $(\bar{p}, \bar{q}) = R[(e, \bar{f})]$ and $(e, \bar{f}) = (G(x), G(y+1))$. Then from Equation (3) we have

$$(p, q) = (e_{l-1}f_{l-1}e_{l-3}f_{l-3} \cdots e_1 f_1, e_{l-2}f_{l-2}e_{l-4}f_{l-4} \cdots e_0 f_0).$$

Now from the Lemma, f and $\bar{f}$ differ in exactly one bit, say bit j. In other words, if $f = f_{l-1} \cdots f_j \cdots f_0$, then $\bar{f} = f_{l-1} \cdots f'_j \cdots f_0$, where f'_j is the complement of f_j. If j is even, then $p = \bar{p}$ since they are independent of the even bits of f and $\bar{f}$, respectively. Thus $(x, y+1, l)$ and (x, y, l) are mapped to the same row. Similarly if j is odd, then $(x, y+1, l)$ and (x, y, l) are mapped to the same column. A similar argument may be used to prove that (x, y, l) and $(x+1, y, l)$ are mapped to either the same row or column, thus proving that P satisfies C1).

To show C2.1) is also satisfied, note that if (e, f, l) is the parent of $(E, F, l+1)$, then

$$(E, F) = (E_l \cdots E_1E_0, F_l \cdots F_1F_0)$$
$$= (e_{l-1} \cdots e_0E_0, f_{l-1} \cdots f_0F_0) = (eE_0, fF_0)$$

Then C2.1) in fact requires that the four nodes $(eE_0, fF_0, l+1)$ be mapped to the same row or column. Given that the Gray codes for these four nodes differ only in one or both of the bits E_0 and F_0 and that $l + 1$ is odd, we conclude that the four nodes are mapped to the same column.

Now we prove C2.2). According to Equation (5) the parent is mapped to $(p_p + \alpha_l, q_p + \beta_l)$, and the children are mapped to $(q_c + \alpha_{l+1}, p_c + \beta_{l+1})$. While from Equation (3) we have

$$(p_p, q_p) = (e_{l-1}f_{l-1}e_{l-3}f_{l-3} \cdots e_1 f_1, e_{l-2}f_{l-2}e_{l-4}f_{l-4} \cdots e_0 f_0)$$

and

$$(p_c, q_c) = (E_{l-1}F_{l-1}E_{l-3}F_{l-3} \cdots E_1F_1, E_lF_lE_{l-2}F_{l-2} \cdots E_0F_0)$$
$$= (e_{l-2}f_{l-2}e_{l-4}f_{l-4} \cdots e_0f_0, E_lF_lE_{l-2}F_{l-2} \cdots E_0F_0)$$

Thus $q_p = p_c$. Since for even l, $\beta_l = \beta_{l+1}$, we have $q_p + \beta_l = p_c + \beta_{l+1}$. That is, the parent

(e, f, l) and its four children $(eE_0, fF_0, l+1)$ are mapped to the same column. Thus C2.2) is satisfied, completing the proof.

3.3. Analysis of the Expansion Cost

Given a source graph $S = \{V_s, E_s\}$ with a set of nodes V_s and a set of edges E_s, the efficiency of an embedding of S into a two-dimensional target array with V_t processors is measured by the expansion cost defined as $C_e = \frac{|V_t|}{|V_s|}$.

Assume that the embedding P of an L–level pyramid (L odd) occupies H rows and W columns in the target APPB. It may be easily shown that

$$W = H = \sum_{i=0}^{(L-1)/2} 2^{2i} = \frac{2^{L+1} - 1}{3}$$

Therefore, the size of the target APPB, in number of nodes, is

$$|V_t| = W \times H = \frac{4^{L+1} - 2^{L+2} + 1}{9}$$

Since the number of nodes in the pyramid is $|V_s| = (4^L - 1)/3$, the expansion cost is

$$C_e = \frac{|V_t|}{|V_s|} = \frac{4(4^L - 2^L + 1/4)}{3(4^L - 1)}$$

which is always less than 1.33. This expansion cost can be improved by flipping over levels L–4 through 0 as shown in Fig 3.1(b). For this embedding we have

$$W = 2^{L-1} + 2^{L-3}$$

and

$$H = 2^{L-1} + 2^{L-3} + 2^{L-5}$$

Thus the area taken by the embedding is

$$|V_t| = \frac{105 \times 4^L}{256}$$

from which it is straight forward to show that the expansion cost is always less than 1.23.

4. Summary

The concept of *pipelined busses* for parallel architectures diverges from the conventional *exclusive access busses*, and offers both possibilities and challenges for significantly improving the efficiency of interprocessor communications in parallel computers. We have presented an efficient embedding of pyramids in array processors with pipelined busses. The embedding has the property that all the neighboring nodes in the pyramid are mapped to the same bus. Thus any two neighbors in the embedded pyramid can communicate with each other using a single bus cycle.

Acknowledgement

The authors would like to thank Professor Richard W. Hall for carefully reading this manuscript and for providing valuable feedback. This work was, in part, supported under Air Force grant AFOSR-88-198, and under NSF grant MIP-8901053.

References

1. S.H. Bokhari, "Finding Maximum on an Array Processor with a Global Bus," *IEEE Trans Comput*, vol. C-32, no. 2, pp. 133-139, 1984.
2. V. Cantoni and S. Levialdi, *Pyramidal Systems for Computer Vision, NATO ASI Series F: Computer and Systems Sciences*, 25, Springer-Verlag, New York, 1986.
3. D.M. Chiarulli, S.P. Levitan, and R.G. Melhem, "Optical Bus Control for Distributed Multiprocessors," *Journal of Parallel and Distributed Computing*, to appear.
4. B.D. Clymer and J.W. Goodman, "Optical Clock Distribution to Silicon Chips," *SPIE Proceedings*, vol. 625, pp. 134-138, 1986.
5. Z. Guo, R.G. Melhem, R.W. Hall, D.M. Chiarulli, and S.P. Levitan, "Array Processors with Pipelined Optical Busses ," *3rd Symp on Frontiers of Massively Parallel Computation*, College Park, MD, 1990, to appear.
6. Z. Guo, "Array Processors with Pipelined Busses and Their Implication in Optically and Electronically Interconnected Multiprocessor Architectures," , Ph.D. Thesis, Dept of Electrical Engineering, University of Pittsburgh, in preparation.
7. M. Karnaugh, "A Map Method for Synthesis of Combinational Logic Circuits," *Trans AIEE, Comm and Electronics*, vol. 72, no. 1, pp. 593-599, Nov 1953.
8. B.S. Kawasaki, K.O. Hill, and R.G. Lamont, "Biconical-Taper Single-Mode Fiber Coupler," *Optics Letters*, vol. 6, no. 7, pp. 327-328, July 1981.
9. R.G. Melhem, D.M. Chiarulli, and S.P. Levitan, "Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems," *The Computer Journal*, vol. 32, no. 4, pp. 362-369, 1989.
10. R. Miller and Q.F. Stout, "Mesh Computer Algorithms for Computational Geometry," *IEEE Trans Comput*, vol. C-38, no. 3, pp. 321-340, 1989.
11. D. Nassimi and S. Sahni, "An Optimal Routing Algorithm for Mesh-Connected Parallel Computers," *Journal ACM*, vol. 27, no. 1, pp. 6-29, January 1980.
12. D. Nassimi and S. Sahni, "Finding Connected Components and Connected Ones on a Mesh-Connected Parallel Computer," *SIAM J. Computing*, vol. 9, pp. 744-457, 1980.
13. A. Neyer, "Electro-Optic X-Switch Using Single-Mode Ti:$LiNbO_3$ Channel Waveguides," *Electronics Letters*, vol. 19, no. 14, pp. 553-554, July 1983.
14. V.K. Prasanna-Kumar and D. Reisis, "Image Computations on Meshes with Multiple Broadcast," *IEEE Trans PAMI*, vol. PAMI-11, no. 11, pp. 1194-1202, 1989.
15. Q.F. Stout, "Mesh Connected Computers with Broadcasting," *IEEE Trans Comput*, vol. C-32, pp. 826-630, 1983.
16. S.L. Tanimoto, "A pyramidal Approach to Parallel Processing," *10th International Symp Comput Arch*, Stockholm, 1983.
17. C.D. Thompson and H.T. Kung, "Sorting on a Mesh-Connected Parallel Computer," *Commun ACM*, vol. 20, no. 4, pp. 263-271, Apr 1977.