Routing Algorithms on the Bus-Based Hypercube Network

Lee-Juan Fan, Chang-Biau Yang, and Shyue-Horng Shiau

Abstract—In this paper, we study the properties of the bus-based hypercube, denoted as $U(n,b)$, which is a kind of multiple-bus networks (MBN). $U(n,b)$ consists of 2^n processors and 2^b buses, where $0 \leq b \leq n-1$, and each processor is connected to either $\lceil \frac{b+2}{2} \rceil$ or $\lceil \frac{b+1}{2} \rceil$ buses. We show that the diameter of $U(n,b)$ is $\lceil \frac{b+1}{2} \rceil$ if $b \geq 2$. We also present an algorithm to select the best neighbor processor via which we can obtain one shortest routing path. In $U(n,b)$, we show that if there exist some faults, the fault diameter $DF(n,b,f) \leq b+1$, where f is the sum of bus faults and processor faults and $0 \leq f \leq \lceil \frac{b-3}{2} \rceil$. Furthermore, we also show that the bus-fault diameter $DB(n,b,f) \leq \lfloor \frac{b}{2} \rfloor + 3$, where $0 \leq f \leq \lceil \frac{b-1}{2} \rceil$ and f is the number of bus faults. These results improve significantly the previous result that $DB(n,b,f) \leq b+2f+1$, where f is the number of bus faults.

Index Terms—Multiple-bus network, hypercube, routing algorithm, diameter, fault tolerance.

1 Introduction

One of the most important components of a parallel processing system is the interconnection network. The method of connecting processors is also an important consideration for design and performance of the system. Various interconnection networks have been proposed and studied. Notable examples include the *crossbar switch* and *multiple bus* systems, *multistage* networks, and point-to-point (direct) connection schemes.

In this paper, we consider a kind of interconnection networks called the *multiple-bus network* (MBN) [6], [7], [8], [12], [15], [17], [19], [22], [23], [32], [33], [34]. The MBN is an extension of the single bus network. An MBN consists of a set of processors and a set of buses. Any pair of processors can communicate via the buses which they both are connected to, and only one message may be transmitted on a bus during a time step. MBNs have several advantages over point-to-point networks. Some of them are listed as follows:

- In a point-to-point network, a communication link is connected to a pair of processors. In an MBN, a cluster of processors share a single communication bus, therefore it can be used more efficiently.
- The number of buses is independent of the number of processors. Hence, the network designer can make a trade off between the performance of the architecture and the physical resource.
- MBNs allow easy broadcasts and readily extend to larger systems.

One of the disadvantages of MBNs is that it is difficult to implement buses with a large number of connections. As the number of connections to a bus increases, the system will operate slowly. Some overlapping connectivity networks were proposed to solve the problem, and bandwidth formulas for these networks were derived by using probabilistic analysis methods [33], [34]. These kind of networks allows each processor to access a group of memory modules. The simulation results of these networks show that the bandwidth of them is superior to conventional MBNs. But, additional switches are required in the overlapping connectivity network.

Vaidyanathan and Padmanabhan proposed a bus-based hypercube network, which can perform uniform [1], [26] hypercube algorithms optimally [32]. The bus-based hypercube network $U(n,b)$ consists of 2^n processors and 2^b buses, where $0 \leq b \leq n-1$. $U(n,b)$ can run any step of a uniform n-dimensional hypercube algorithm optimally in 2^{n-1-b} steps. The number of buses connected to a processor is either $\lceil \frac{b+2}{2} \rceil$ or $\lceil \frac{b+1}{2} \rceil$. In this paper, we shall concentrate our attention to the investigation of properties of $U(n,b)$.

Many other MBNs have been proposed. Ali and Vaidyanathan presented the exact lower bounds on running ASCEND/DECEND and FAN-IN algorithms on synchronous MBNs [1]. Dighe et al. proposed a class of MBNs called *bus-connected ring trees* and *bus-based trees* [7], [8]. Multiple buses have also been used in some synchronous reconfigurable systems [24], [25], [29], [31]. Some modified network topologies have been proposed to enhance communication performance [2], [9], [10], [12], [13], [30].

Ishikawa proposed a modified hypercube with multiple buses which used a bypass routing method to reduce the diameter to two [14]. Lee et al. [19] proposed the time division multiplexed hypercube (TDM-cube) and the time/wavelength division multiplexed mesh (TWDM-mesh). The TDM-cube (N) implements the binary hypercube interconnection among N nodes in $log_2 N$ time slots. The TWDM (n^2) has a small network delay along with much higher network throughput than the Bus-Mesh. Louri et al. [22], [23] proposed a hybrid network, called the spanning multichannel linked hypercube (SMLH), whose features include the possibility of a constant diameter and a constant degree network. And, the network is also scalable and fault-tolerant.

- *L.-J. Fan is with the Department of Applied Mathematics, National Sun Yat-sen University, No. 61, Wendong 5th St., Guishan Shiang, Taoyuan County, Taiwan 333. E-mail: mxlcfan@gigigaga.com.*
- *C.-B. Yang and S.-H. Shiau are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan 804. E-mail: {cbyang shiaush}@cse.nsysu.edu.tw.*

Manuscript received 20 Aug. 2003; revised 25 Mar. 2004; accepted 7 July 2004; published online 23 Feb. 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0142-0803.

Serrano and Parhami proposed an optimal architecture with separable ROW/COLUMN buses for mesh-connected parallel computers [28]. In this architecture, semigroup and prefix computations can be performed in optimal time complexity. Cheung and Lau presented an iterative walk-and-ride technique for solving routing with locality problems on meshes [5].

An efficient routing algorithm is a key to the performance of multiprocessor systems. Although there are many results on MBNs, the study of routing problems on $U(n,b)$ have not been proposed. Some good properties on $U(n,b)$ were presented by Vaidyanathan and Padmanabhan [32]. Based on the properties, in this paper, we obtain more properties on $U(n,b)$ with which messages can be transmitted efficiently. We propose an algorithm to select the best neighbor processor with which we can obtain one shortest routing path. And, we show that the diameter of $U(n,n-1)$ is $\lceil \frac{n}{2} \rceil$. The result can be extended to $U(n,b)$, where $0 \le b \le n-1$. If $b \ge 2$, its diameter is $\lceil \frac{b+1}{2} \rceil$.

Fault tolerance is also very important in multiprocessor systems. Fault diameter is one of the important measurements of fault tolerance. In this paper, we show that if there exist some faults in $U(n,b)$, the fault diameter $DF(n,b,f) \le b+1$, where f is the sum of bus faults and processor faults and $0 \le f \le \lceil \frac{b-3}{2} \rceil$. Furthermore, we also show that the bus-fault diameter $DB(n,b,f) \le \lfloor \frac{b}{2} \rfloor + 3$, where $0 \le f \le \lceil \frac{b-1}{2} \rceil$ and f is the number of bus faults. Our results on bus-fault diameter improve significantly the previous result [32] that $DB(n,b,f) \le b + 2f + 1$, where f is the number of bus faults.

The rest of this paper is organized as follows: In Section 2, we shall review the construction of the network. In Section 3, we shall give the terms and notations we shall use in this paper. In Section 4 and Section 5, the distance properties between two processors will be investigated. We show that the diameter of $U(n,b)$, $b \ge 2$, without faults is $\lceil \frac{b+1}{2} \rceil$. Accordingly, we design the shortest path routing algorithm for the network, which has never been designed before. In Section 6, we shall present some fault tolerant properties of this network. By finding disjoint paths, we obtain the diameters with some bus faults and with some bus and/or processor faults. Finally, the conclusion will be given in Section 7.

2 Construction of the Bus-Based Hypercube Network

In this section, we shall describe how to construct $U(n,b)$ [32]. $U(n,b)$ consists of 2^n processors and 2^b buses, where $0 \le b \le n-1$. The fan-out of each processor, the number of connections to buses, is either $\lceil \frac{b+2}{2} \rceil$ or $\lceil \frac{b+1}{2} \rceil$. A *high* processor has fan-out $\lceil \frac{b+2}{2} \rceil$ and a *low* processor has fan-out $\lceil \frac{b+1}{2} \rceil$. Note that one low processor must not have more connections than one high processor. The *state* of processor is either high or low.

The construction of $U(n,b)$ is similar to that of the hypercube. In $U(n,b)$, each processor has a unique identifier between 0 and 2^n-1, and each bus also has a unique identifier between 0 and 2^b-1. $U(n,b)$ can be defined by the following recursive ways:

1. The initial structures, $U(1,0)$ and $\overline{U}(1,0)$, are shown in Fig. 1. In the figure, L and H are used to denote a low processor and a high processor, respectively.

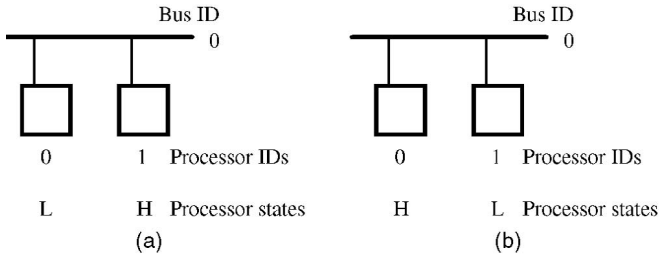


Fig. 1. Initial structures of U. (a) $U(1,0)$. (b) $\overline{U}(1.0)$.

$\overline{U}(1,0)$ is a dual of $U(1,0)$. In $U(1,0)$ or $\overline{U}(1,0)$, the only one bus is called the *host bus* of these two processors, because the processors are connected to the bus first.

2. The construction of $U(n+1,b+1)$ can be obtained by combining $U(n,b)$ and $\overline{U}(n,b)$. Assume i and j, $0 \le i,j \le 2^n-1$, are a low and a high processors in $U(n,b)$, respectively. Then, i and j are high and low in $\overline{U}(n,b)$, respectively. A connection is added from a low processor i of $U(n,b)$ to the host bus d of the high processor i of $\overline{U}(n,b)$. Here, the bus d is called a *guest bus* of i of $U(n,b)$. And, similarly, a connection is added from a low processor j of $\overline{U}(n,b)$ to the host bus c of the high processor j of $U(n,b)$, and the bus c is a guest bus of j of $\overline{U}(n,b)$. Finally, since a low processor must not have more connections than a high processor, all low processors are changed to high processors, and high processors are changed to low processors. Processor i of $U(n,b)$ is still identified as i of $U(n+1,b+1)$, and processor i of $\overline{U}(n,b)$ is identified as $i+2^n$ of $U(n+1,b+1)$. Bus k of $U(n,b)$ is still identified as k in $U(n+1,b+1)$, and bus k of $\overline{U}(n,b)$ is identified as $k+2^b$ of $U(n+1,b+1)$.
3. To obtain $U(n+1,b)$, two $U(n,b)$s are combined, but the buses are not doubled. The buses in the two $U(n,b)$s are overlapped. Since all processors are not added any extra connections, all low processors are still low processors, high processors are still high processors. Processor i of the first $U(n,b)$ is still identified as i of $U(n+1,b)$, and processor i of the second $U(n,b)$ is identified as $i+2^n$ of $U(n+1,b)$. Bus k of the first $U(n,b)$ and the second $U(n,b)$ is the same one and is identified as k of $U(n+1,b)$.

As examples, the constructions of $U(2,1)$, $U(3,1)$, and $U(3,2)$ are shown in Figs. 2, 3, and 4, respectively. In $U(3,2)$, the buses connected to processor 100 are 00 and 10. Bus 10 is the host bus of processor 100 since bus 10 is the bus connected to processor 100 first among all buses. Later, Theorem 4 will give an easy way to identify the host bus of a processor.

Constructing a $U(n,b)$ could be done in any order from two choices. The first method is to double both the number of processors and the number of buses, which is called the *fully doubling method*. The second method is doubling only the number of the processors, which is called the *partially doubling method*. Starting from $U(1,0)$, to construct $U(n,b)$, there are many permutations to apply the fully doubling method b times, and the partially doubling method $n-b-1$ times. Different orders of the two methods in construction of $U(n,b)$

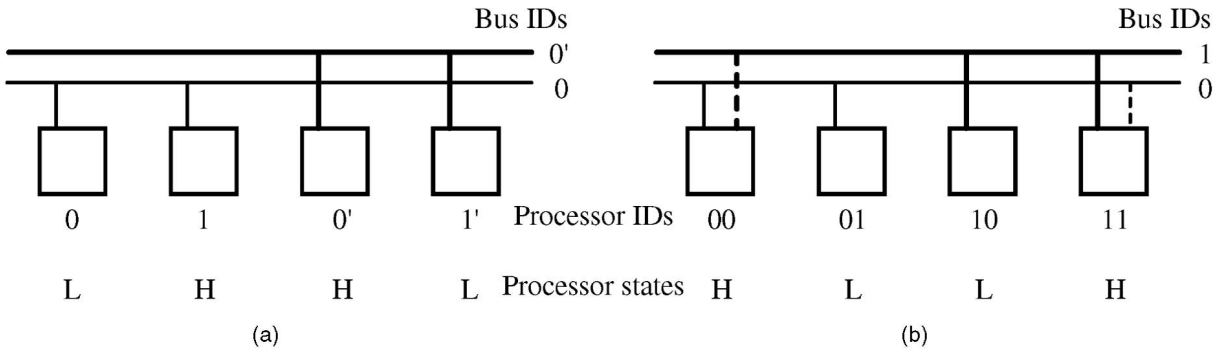


Fig. 2. Construction of $U(2,1)$. (a) Combining $U(1,0)$ and $\overline{U}(1,0)$. (b) Adding connections (dashed lines) in *low* processors in $U(1,0)$ and $\overline{U}(1,0)$. Then, change the state from low to high and the state from high to low.

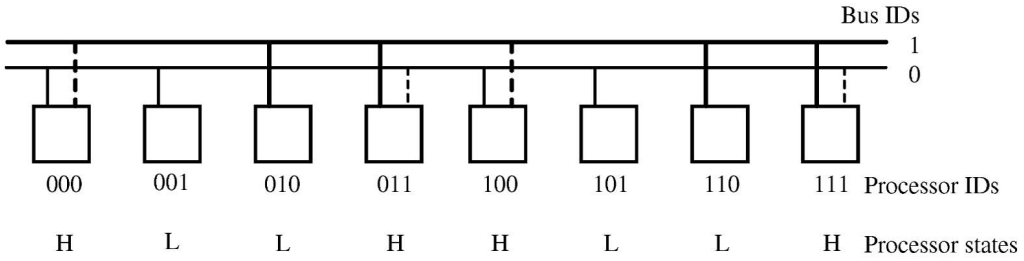


Fig. 3. The construction of $U(3,1)$. $U(3,1)$ consists of two $U(2,1)$s, and the buses are shared to each other.

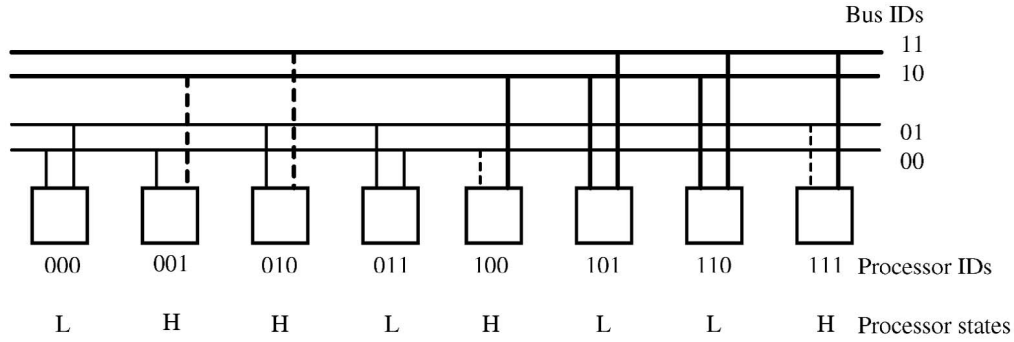


Fig. 4. The bus-based hypercube $U(3,2)$.

would produce different architectures. In this paper, we assume that $U(n,b)$ is constructed by applying fully doubling first and then partially doubling. Note that the structure $U(n,b)$ is unique if $b = n-1$, because only fully doubling is repeatedly applied.

The network $U(n,b)$ allows the network designer to choose different values of b due to the cost consideration, thus he can make a trade off between the performance of $U(n,b)$ and physical resources [32]. As we know, such a trade off is not possible for a point-to-point interconnection network. For example, an n-dimensional hypercube generally must have 2^n processors and $n2^{n-1}$ links.

The following theorem gives some properties of $U(n,b)$ and $\overline{U}(n,b)$.

Theorem 1 [32]. *For all* $0 \le b \le n-1$,

1. *$U(n,b)$ can run any step of a uniform hypercube algorithm optimally.*
2. *Each bus is connected by $(b+2)2^{n-b-1}$ processors.*
3. *Processor i, $0 \le i \le 2^n-1$, is a high (low) processor of $U(n,b)$ if and only if processor i is a low (high) processor of $\overline{U}(n,b)$.*
4. *Each bus j of $U(n,b)$ or $\overline{U}(n,b)$ is connected to at least two processors i_1 and i_2 such that $0 \le i_1 \le 2^{n-1}-1$ and $2^{n-1} \le i_2 \le 2^n-1$.*

Vaidyanathan and Padmanabhan also showed that $U(n,b)$ has a low diameter and is highly resilient to bus faults [32]. The bus-fault diameter $DB(n,b,f)$ denotes the diameter of $U(n,b)$ with any f bus faults.

Theorem 2 [32]. *For all $n > b \ge 0$, for all $0 \le f \le \lceil \frac{b-1}{2} \rceil$, $DB(n,b,f) \le b+2f+1$.*

3 Notations

We first give some notations used in this paper in the following:

- $Bin(i,n)$: n binary bits for representing i, where $0 \le i \le 2^n-1$. The rightmost and leftmost bits of $Bin(i,n)$ are counted as bits 0 and $n-1$, respectively.

- $Zero(i)$: number of bit positions (including leading zero bits) in $Bin(i,n)$ having value 0.
- $One(i)$: number of bit positions in $Bin(i,n)$ having value 1.
- $Even(i)$: number of even bit positions, *excluding bit 0*, in $Bin(i,n)$ having value 1.
- $Odd(i)$: number of odd bit positions in $Bin(i,n)$ having value 1.
- $B(i)$: the set of buses connected to processor i.
- $b_0(i)$, $\bar{b}_0(i)$, $\bar{b}_s(i)$, $\bar{b}_{s,t}(i)$: Suppose $i=(i_{n-1}, i_{n-2}, \cdots, i_1, i_0)$. Then, $b_0(i)=\bar{b}_0(i)=(i_{n-1}, i_{n-2}, \cdots, i_1)$ and $\bar{b}_s(i)=(i_{n-1}, \cdots, \overline{i_s}, \cdots, i_1)$, $1 \le s \le n-1$, where $\overline{i_s}$ denotes the inverse of i_s. $\bar{b}_{s,t}(i)=\bar{b}_s(\bar{b}_t(i))=(i_{n-1}, \cdots, \overline{i_s}, \cdots, \overline{i_t}, \cdots, i_1)$ or $(i_{n-1}, \cdots, \overline{i_t}, \cdots, \overline{i_s}, \cdots, i_1)$, $s \neq t$, and $1 \le s,t \le n-1$.
- $H(i,j)$: the distance from processor i to processor j, which is the number of hops included in the shortest path from i to j.
- R_{ij}: the bit-wise exclusive-or operation on the binary representations of i and j, i.e., $R_{ij}=Bin(i,n)\bigoplus Bin(j,n)=(r_{n-1}, r_{n-2}, \cdots, r_0)$. The operation indicates the different bit positions between i and j.
- $High(i)$: a function. $High(i)=1$ if and only if processor i is high, and $High(i)=0$ if and only if processor i is low.

$U(n,b)$ consists of 2^n processors and 2^b buses, $0 \le b \le n-1$, constructed with the method described in the previous section. Some notations for describing the properties of $U(n,b)$ are given as follows:

- $D(n,b)$: the diameter of $U(n,b)$.
- $DB(n,b,f)$: the bus-fault diameter of $U(n,b)$ with at most f bus faults and no processor faults.
- $DF(n,b,f)$: the fault diameter of $U(n,b)$, where f is the sum of bus faults and processor faults.

4 Distance Properties of $U(n,b)$ Where $b=n-1$

We shall present some distance properties of $U(n,b)$, with which we can route messages efficiently. We discuss the simple case that $b=n-1$ in this section. The properties of $U(n,b)$ when $b<n-1$ can be easily extended and will be discussed in the next section.

The proofs of Theorems 3 and 4 can be found in the work of Kamath [16] (Lemma 1 and Lemma 4) and our work [11] (Lemma 3.1 and Lemma 3.2). Therefore, the proofs are omitted here.

Theorem 3. *In $U(n,n-1)$, $Zero(i)$ is even iff $High(i)=1$, and $Zero(j)$ is odd iff $High(j)=0$.*

For example, in $U(3,2)$ (see Fig. 4), processor 100 is high and $Zero(i)=2$.

Theorem 4. *In $U(n,n-1)$, processor $i=(i_{n-1}, i_{n-2}, \cdots, i_0)$ is connected to the set $B(i)$ of buses*

1. *$b_0(i)=(i_{n-1}, i_{n-2}, \cdots, i_1)$, which is the host bus of processor i, and*
2. *$\bar{b}_s(i)=(i_{n-1}, i_{n-2}, \cdots, \overline{i_s}, \cdots, i_1)$, where $(n-s)$ is odd if $High(i)=1$, and $(n-s)$ is even if $High(i)=0$.*

In Theorem 4,

$$B(i)=\{b_0(i)\}\cup\{\bar{b}_s(i) | n-s \text{ is odd if } i \text{ is high}, \; n-s \text{ is even if } i \text{ is low}\}.$$

In the above, $b_0(i)$ is the host bus of i, and $\bar{b}_s(i)$ is one guest bus.

For example, in $U(3,2)$ (see Fig. 4), processor 100 is high, and it is connected to the host bus 10 and the guest bus 00. In $U(2,1)$ (see Fig. 2), processor 00 is high, and it is connected to the host bus 0 and the guest bus 1.

As another example, in $U(7,6)$, processor $i=(1111111)$ is high, and processor $j=(1000001)$ is low. Then,

$$\begin{aligned} B(i) &= \{111111\}\cup\{\underline{0}11111, 11\underline{0}111, 1111\underline{0}1\} \\ &= \{b_0(i)\}\cup\{\bar{b}_6(i), \bar{b}_4(i), \bar{b}_2(i)\} \text{ and} \\ B(j) &= \{100000\}\cup\{1\underline{1}0000, 100\underline{1}00, 10000\underline{1}\} \\ &= \{b_0(j)\}\cup\{\bar{b}_5(j), \bar{b}_3(j), \bar{b}_1(j)\}. \end{aligned}$$

Note that $b_0(i)=(111111)$ and $b_0(j)=(100000)$ are the host buses of processors i and j, respectively, the different bit positions of $\bar{b}_6(i)$, $\bar{b}_4(i)$, and $\bar{b}_2(i)$ to $b_0(i)$ are 2, 4, and 6, respectively. And, the different bit positions of $\bar{b}_5(j)$, $\bar{b}_3(j)$, and $\bar{b}_1(j)$ to $b_0(j)$ are 1, 3, and 5, respectively. Thus, no matter the state of the processor is high or low, the different bit positions at its guest buses jump every two bits. If two processors have the same states, the jumps will be at the same positions, otherwise the jumps will not be at the same positions.

By observation, we find some regular relations between the two bus sets of two processors. For example, in $U(7,6)$, consider two high processors $i=(1111111)$ and $j=(1101110)$. Then,

$$\begin{aligned} B(i) &= \{111111\}\cup\{\underline{0}11111, 11\underline{0}111, 1111\underline{0}1\} \\ &= \{b_0(i)\}\cup\{\bar{b}_6(i), \bar{b}_4(i), \bar{b}_2(i)\} \text{ and} \\ B(j) &= \{110111\}\cup\{\underline{0}10111, 11\underline{1}111, 1101\underline{0}1\} \\ &= \{b_0(j)\}\cup\{\bar{b}_6(j), \bar{b}_4(j), \bar{b}_2(j)\}. \end{aligned}$$

We have $B(i)\cap B(j)=\{111111, 110111\}$, $R_{ij}=(0010001)$, $One(R_{ij})=2$, and $r_4=r_0=1$. $r_4=1$ indicates that position 4 is an important position. After jumping at positions 2, 4, and 6 on the host bus of processor j, we can get the guest buses of processor j, $\{010111, 11\underline{1}111, 110101\}$. As we can see, $\bar{b}_4(j)=b_0(i)$. Similarly, $b_0(i)=(111111)$ can be jumped at position 4 and transferred to $\bar{b}_4(i)=(11\underline{0}111)=b_0(j)$. We extend the observation to Theorem 5.

Theorem 5. *In $U(n,n-1)$, let B denote the set of buses to which both processors $i=(i_{n-1}, i_{n-2}, \cdots, i_0)$ and $j=(j_{n-1}, j_{n-2}, \cdots, j_0)$ are connected, i.e., $B=B(i)\cap B(j)$. Each of the following is true:*

1. *$H(i,j)=1$ and $High(i)=High(j)=1$ iff $One(R_{ij})=2$, $r_s=r_t=1$, where $s>t$, and $(n-s)$ is odd, $(n-t)$ is odd or $t=0$, and $B=\{\bar{b}_s(i)\}\cup\{\bar{b}_t(i)\}$.*
2. *$H(i,j)=1$ and $High(i)=High(j)=0$ iff $One(R_{ij})=2$, $r_s=r_t=1$, where $s>t$, and $(n-s)$ is even, $(n-t)$ is even or $t=0$, and $B=\{\bar{b}_s(i)\}\cup\{\bar{b}_t(i)\}$.*
3. *$H(i,j)=1$ and $High(i)=1$ and $High(j)=0$ iff one of the following two subcases is true:*

TABLE 1
The Different Bits between Two Neighboring Processors

high $i=$	$(\underline{0}0\underline{0}01)$	high $i=$	$(00\underline{0}0\underline{1})$	high $u=$	$(\underline{1}1\underline{0}0\underline{1})$	high $u=$	$(\underline{1}1\underline{0}01)$
position	(43210)	position	(43210)	position	(43210)	position	(43210)
high $j=$	$(\underline{1}0\underline{1}01)$	high $k=$	$(00\underline{1}0\underline{0})$	low $v=$	$(\underline{0}0\underline{0}0\underline{0})$	low $w=$	$(1\underline{0}001)$

a. $One(R_{ij})=1$ *and* $r_s=1$, *where* $0\le s\le n-1$, *and* $B=\{\bar{b}_s(i)$ *if* $(n-s)$ *is odd, or* $b_0(i)$ *if* $(n-s)$ *is even or* $s=0\}$.
b. $One(R_{ij})=3$ *and* $r_0=r_s=r_t=1$, *where* $n-s$ *is odd and* $n-t$ *is even, and* $B=\{\bar{b}_s(i)\}$.

Proof. If $One(R_{ij})\ge 4$, by Theorem 4, i and j cannot be connected to a common bus, i.e., $H(i,j)\neq 1$. Hence, $1\le One(R_{ij})\le 3$.

We first prove the only if part of case (1). Suppose that $H(i,j)=1$ and $High(i)=High(j)=1$. By Theorem 3, if processors i and j have the same state, $One(R_{ij})$ is even. Hence, $One(R_{ij})=2$. Assume $r_s=r_t=1$, and $s>t\ge 0$. By Theorem 4, $B(i)=\{b_0(i)\}\cup\{\bar{b}_x(i))|\ n-x$ is odd$\}$. If $t=0$, then $j=(i_{n-1},i_{n-2},\cdots,\overline{i_s},\cdots,i_1,\overline{i_0})$. First, assume $n-s$ is even, then $B(j)=\{\bar{b}_s(i)\}\cup\{\bar{b}_{y,z}(i)|$ one of y and z is s, one of $n-y$ and $n-z$ is odd, and the other is even $\}$. Then, $B=B(i)\cap B(j)=\emptyset$, i.e., $H(i,j)\neq 1$. Thus, $n-s$ must be odd. Then, $B(j)=\{\bar{b}_s(i)\}\cup\{\bar{b}_{p,s}(i)|\ n-p$ and $n-s$ are both odd$\}$. Since $n-p$ and $n-s$ are both odd, we can jump position p. One of the jumps should punch at position s. $\{\bar{b}_{p,s}(i)|\ n-p$ and $n-s$ are both odd$\}$ becomes

$$\{\bar{b}_{p,s}(i)=(i_{n-1},i_{n-2},\cdots,\bar{\bar{i}}_s,\cdots,i_1)$$
$$=(i_{n-1},i_{n-2},\cdots,i_s,\cdots,i_1)=b_0(i)\}.$$

If position p does not punch at position s, it becomes $\{\bar{b}_{y,z}(i)|\ n-y$ and $n-z$ are both odd, one of y and z is $s\}$. Summarizing the previous inferences, we have

$$B(j)=\{\bar{b}_s(i)\}\cup\{b_0(i)\}\cup\{\bar{b}_{y,z}(i)|n-y \text{ and } n-z \text{ are both odd, one of } y \text{ and } z \text{ is } s\}.$$

$\{b_0(i)\}$ in $B(j)$ matches $\{b_0(i)\}$ in $B(i)$. Since $n-s$ is odd, $\{\bar{b}_s(i)\}$ in $B(j)$ should be matched to one of $\{\bar{b}_x(i)|\ n-x$ is odd $\}$ in $B(i)$. Thus, $B=B(i)\cap B(j)=\{\bar{b}_s(i),b_0(i)\}$, where $n-s$ is odd. If $t>0$, $j=(i_{n-1},\cdots,\overline{i_s},\cdots,\overline{i_t},\cdots,i_1,i_0)$. By the similar argument, we obtain that both $n-s$ and $n-t$ must be odd. We have

$$B(j)=\{\bar{b}_{s,t}(i)\}\cup\{\bar{b}_s(i)\}\cup\{\bar{b}_t(i)\}$$
$$\cup\{\bar{b}_{x,y,z}(i)|\text{ all } n-x,n-y, \text{ and } n-z \text{ are odd, and two of } x,y \text{ and } z \text{ are } s \text{ and } t\}.$$

Thus, $B=B(i)\cap B(j)=\{\bar{b}_s(i),\bar{b}_t(i)\}$, where both $n-s$ and $n-t$ are odd.

Next, we prove the if part of case (1). Suppose that $One(R_{ij})=2$, $r_s=r_t=1$, where $s>t$, and $(n-s)$ is odd, $(n-t)$ is odd or $t=0$. Since $One(R_{ij})=2$, by Theorem 3, $Zero(i)$ and $Zero(j)$ are either both even or both odd. It means that $High(i)=High(j)=1$ or $High(i)=High(j)=0$. Since i is connected to bus $\bar{b}_s(i)$, where $n-s$ is odd, by Theorem 4, i is a high processor. Thus $High(i)=High(j)=1$. The proof of case (1) of the theorem is completed.

The proofs of case (2) and case (3) are quite similar to that of case (1) and, so, are omitted. □

We use some examples to illustrate Theorem 5. In $U(5,4)$, suppose processors $i=(00001)$, $j=(10101)$, and $k=(00100)$. By Theorem 3, processors i, j, and k are all high processors. By Theorem 4, $B(i)=\{0000,1000,0010\}$, $B(j)=\{1010,0010,\ 1000\}$ and $B(k)=\{0010,1010,0000\}$. By case (1) of Theorem 5, processors j and k are neighbors of processor i, the buses $\{0010,1000\}$ are connected by both processors i and j, and the buses $\{0010,0000\}$ are connected by both processors i and k. The different bits between i and j or i and k are two and at even positions, as shown in Table 1. Note that one of the two different bits between processors i and k is at bit 0.

As an example of case (3) of Theorem 5, let $u=(11001)$, $v=(00000)$, and $w=(10001)$. Processor u is high, and processors v and w are low. $B(u)=\{$*1100*, **0100**, 1110$\}$, $B(v)=\{0000,$ **0100**$,0001\}$ and $B(w)=\{1000,$ *1100*, 1001$\}$. The bus **0100** is connected by both processors u and v. The bus *1100* is connected by both processors u and w. The different bit positions are also shown in Table 1.

Theorem 5 can be used to guide the routing from a source processor to a destination processor. If either case (1) or case (2) of Theorem 5 is applied in one routing step, we call it is a *2-bit routing step*. If case (3)a and case (3)b of Theorem 5 are applied in one routing step, we call them are *1-bit* and *3-bit routing steps*, respectively. Table 1 also explains some examples of 2-bit, 3-bit, and 1-bit routing steps. The first column at the table is an example for *even 2-bit routing step*. The changes of one *odd 2-bit routing step* are at two odd positions.

We have some observations from Theorem 5. Suppose we perform routing from processor i to processor j and $R_{ij}=Bin(i,n)\bigoplus Bin(j,n)=(r_{n-1},r_{n-2},\cdots,r_0)$. If $r_0=0$, applying only one 3-bit routing step could not make the distance shorter than changing one 2-bit if changing one 2-bit is possible. For example, in $U(5,4)$, suppose the source is the low processor $i=(00000)$ and the destination is the high processor $j=(01110)$. By Theorem 5, one of the shortest paths from i to j is $\{00000,01010,01110\}$. If we first apply a 3-bit routing step, then the path from i to j may be $\{00000,01101,01100,01110\}$, which is longer than the distance between i and j.

Therefore, to send messages from source i to destination j via one shortest path, we must select a best neighbor processor from all candidates that connect to processor i. Of course, not all candidates can be selected to route messages to the destination via one shortest path. On the other hand, there are many *permutations* and *combinations* of changing different bits, but it is important to know which is the best next intermediate processor to route messages to the destination. Before presenting how to select one best neighbor processor of the source processor, we give the distance between two processors.

Lemma 1. *In $U(n, n-1)$, the distance between processors i and j is $H(i,j) \le \lfloor \frac{One(R_{ij})}{2} \rfloor + 1$.*

Proof. By Theorem 5, from processor i to processor j, there exists one routing path P_{ij} which consists of only 2-bit and 1-bit routing steps. Thus, $H(i,j)$ is no more than the length of P_{ij}. The construction of P_{ij} can be divided into the following cases.

Case 1: $r_0 = 0$, $Even(R_{ij}) = 2x$, and $Odd(R_{ij}) = 2y$, where x and y are nonnegative integers. The routing path can be obtained by applying x even 2-bit routing steps, one 1-bit routing step (for changing states), $y-1$ odd 2-bit routing steps and one 1-bit routing step. Note that the order of even 2-bit routing steps and odd 2-bit routing steps may be exchanged. Its length is $x+1+(y-1)+1 = x+y+1 = \frac{One(R_{ij})}{2}+1 = \lfloor \frac{One(R_{ij})}{2} \rfloor + 1$.

Case 2: $r_0 = 0$, $Even(R_{ij}) = 2x$, and $Odd(R_{ij}) = 2y+1$. The routing path can be obtained by applying x even 2-bit routing steps, one 1-bit routing step and y odd 2-bit routing steps. Its length is $x+1+y = \lfloor \frac{One(R_{ij})}{2} \rfloor + 1$.

Case 3: $r_0 = 0$, $Even(R_{ij}) = 2x+1$. It can be proved similarly.

Case 4: $r_0 = 1$, $Even(R_{ij}) = 2x$. and $Odd(R_{ij}) = 2y$. The routing path consists of x even 2-bit routing steps, one 1-bit routing step (changing bit 0) and y odd 2-bit routing steps. The length is $x+y+1 = \lfloor \frac{One(R_{ij})}{2} \rfloor + 1$.

Case 5: $r_0 = 1$, at least one of $Even(R_{ij})$ and $Odd(R_{ij})$ is odd. It can be proved similarly. □

Lemma 2. *In $U(n, n-1)$, suppose i is the source processor and j is the destination processor. There exists a neighbor h of processor i, where $One(R_{hj}) \le One(R_{ij})$, such that for any neighbor k of processor i, if $One(R_{ij}) \le One(R_{kj})$, then $H(h,j) \le H(k,j)$.*

Proof. Assume we can route messages from processor i to its neighbor k such that $One(R_{kj}) \ge One(R_{ij})$. By Theorem 5, $One(R_{kj}) = One(R_{ij}) + m$, $1 \le m \le 3$. The possible minimum routing path from processor k to processor j is to apply 2-bit or 3-bit routing steps only, since one 1-bit routing step can correct only one bit. However, if we apply 3-bit routing steps in $2x$ times, where x is a positive integer, only $2x$ bits are corrected, because bit 0 is changed between 0 and 1 alternately. Thus, in the routing sequence, at most one 3-bit routing step is effective in correcting three bits. Therefore, the lower bound of the distance from processor k to processor j is either $\frac{One(R_{kj})}{2}$ if $One(R_{kj})$ is even, or $\frac{One(R_{kj})-3}{2}+1 = \frac{One(R_{kj})-1}{2}$ if otherwise. That is,

$$H(k,j) \ge \lceil \frac{One(R_{kj})-1}{2} \rceil = \lceil \frac{One(R_{ij})+m-1}{2} \rceil \ge \lceil \frac{One(R_{ij})}{2} \rceil.$$

By Lemma 1, $H(i,j) \le \lfloor \frac{One(R_{ij})}{2} \rfloor + 1$. Thus, there exists h such that

$$H(h,j) = H(i,j) - 1 \le \lfloor \frac{One(R_{ij})}{2} \rfloor \le \lceil \frac{One(R_{ij})}{2} \rceil \le H(k,j).$$

□

By Lemma 2, a routing path from processor i to processor j via a neighbor k of i, where $One(R_{ij}) \le One(R_{kj})$, may not be a shortest routing path. Thus, we can get rid of some candidates and reduce the set of candidates. The proof of Theorem 6 will conclude some rules to obtain the set of best neighbor processors and then we can construct some shortest routing paths.

Theorem 6. *In $U(n, n-1)$,*

$$H(i,j) = \begin{cases} 2 \begin{cases} \text{if } Odd(R_{ij}) = 1, Even(R_{ij}) = 0, r_0 = 1, \\ \text{and } n + High(i) = even, \text{or} \\ \text{if } Even(R_{ij}) = 1, Odd(R_{ij}) = 0, r_0 = 1 \\ \text{and } n + High(i) = odd, \text{ and} \end{cases} \\ \quad \lceil \frac{Even(R_{ij})}{2} \rceil + \lceil \frac{Odd(R_{ij})}{2} \rceil \qquad \text{if } r_0 = 0, \\ \quad \lfloor \frac{Even(R_{ij})}{2} \rfloor + \lfloor \frac{Odd(R_{ij})}{2} \rfloor + 1 \quad otherwise. \end{cases}$$

Proof. We shall prove it by induction on the value of $One(R_{ij})$. By Lemma 2, we can get one shortest routing path from processor i to processor j via a neighbor h of i, where $One(R_{hj}) < One(R_{ij})$.

We first prove the case that $r_0 = 0$ and $One(R_{ij}) \le 3$. By Theorem 5, we can find the situation that processors i and j are neighboring. And, we only consider the subcases that i and j are not neighboring. If $Even(R_{ij}) = Odd(R_{ij}) = 1$, we can apply two 1-bit routing steps and $H(i,j) = 2$. If $Even(R_{ij}) = 2$ and $Odd(R_{ij}) = 1$, or $Even(R_{ij}) = 1$ and $Odd(R_{ij}) = 2$, we can apply one 2-bit routing step and one 1-bit routing step and $H(i,j) = 2$. Similarly, $H(i,j) = 2$ if $Even(R_{ij}) = 3$ and $Odd(R_{ij}) = 0$, or $Even(R_{ij}) = 0$ and $Odd(R_{ij}) = 3$.

Next, we prove the case that $r_0 = 1$ and $One(R_{ij}) \le 3$. If $Even(R_{ij}) = 1$ and $Odd(R_{ij}) = 1$, by Theorem 5, only one 3-bit routing step is needed, and $H(i,j) = 1$. If $Odd(R_{ij}) = 2$ and $Even(R_{ij}) = 0$, or $Even(R_{ij}) = 2$ and $Odd(R_{ij}) = 0$, we can apply one 2-bit routing step and one 1-bit routing step, then $H(i,j) = 2$. For the case that $Even(R_{ij}) = 1$ and $Odd(R_{ij}) = 0$, only one 2-bit routing step is needed if $n + High(i)$ is even, and we need two 1-bit routing steps if otherwise. For the case that $Even(R_{ij}) = 0$ and $Odd(R_{ij}) = 1$, only one 2-bit routing step is needed if $n + High(i)$ is odd, and we need two 1-bit routing steps if otherwise. Hence, it is also true when $r_0 = 1$ and $One(R_{ij}) \le 3$.

For the induction hypothesis, we assume that it is true when $One(R_{ij}) \le k-1$. We want to prove that it is also true when $One(R_{ij}) = k$. By Theorem 5, we can apply one routing step to reach one neighbor h of processor i. We first prove the case that $r_0 = 0$. Let $Even(R_{ij}) = x$, $Odd(R_{ij}) = y$, and $x + y = One(R_{ij})$. We have to consider the following cases:

Case 1: 1-bit routing step. If we change an even bit, then the length of the routing path is $d_1 = 1 + H(h,j) = 1 + \lceil \frac{x-1}{2} \rceil + \lceil \frac{y}{2} \rceil$. If we change an odd bit, the length of the routing path is $d_2 = 1 + \lceil \frac{x}{2} \rceil + \lceil \frac{y-1}{2} \rceil$.

Case 2: 2-bit routing step. By Theorem 5, if we can change two even bits, then the length of the routing path is $d_3 = 1 + \lceil \frac{x-2}{2} \rceil + \lceil \frac{y}{2} \rceil = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$. If we can change two odd bits, then the length of the routing path is

$d_4 = 1 + \lceil \frac{x}{2} \rceil + \lceil \frac{y-2}{2} \rceil = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$. Note that either even 2-bit or odd 2-bit routing is valid, not both are valid.

Case 3: 3-bit routing step. The length of the routing path is $d_5 = 1 + \lfloor \frac{x-1}{2} \rfloor + \lfloor \frac{y-1}{2} \rfloor + 1 = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$.

We have that $d_1 \geq d_3$, $d_2 \geq d_4$, and $d_3 = d_4 = d_5 = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$. And, $H(i,j) = \min\{d_1, d_2, d_3, d_4, d_5\} = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$. The theorem is true when $r_0 = 0$ and $One(R_{ij}) = k$.

Next, we prove the case that $r_0 = 1$. Let $Even(R_{ij}) = x$, $Odd(R_{ij}) = y$, and $x + y + 1 = One(R_{ij})$. The following cases have to be considered.

Case a: 1-bit routing step. If we change an even bit but not bit 0, the length of the routing path is $d_1 = 1 + H(h,j) = 1 + \lfloor \frac{x-1}{2} \rfloor + \lfloor \frac{y}{2} \rfloor + 1$. If we change an odd bit, the length of the routing path is $d_2 = 1 + \lfloor \frac{x}{2} \rfloor + \lfloor \frac{y-1}{2} \rfloor + 1$. If we change bit 0, then the length of the routing path is $d_3 = 1 + \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$.

Case b: 2-bit routing step. By Theorem 5, if we can change two even bits including bit 0, the length of the routing path is $d_4 = 1 + \lceil \frac{x-1}{2} \rceil + \lceil \frac{y}{2} \rceil$. If we can change two even bits excluding bit 0, the length of the routing path is $d_5 = 1 + \lfloor \frac{x-2}{2} \rfloor + \lfloor \frac{y}{2} \rfloor + 1$. If we can change one odd bit and bit 0, the length of the routing path is $d_6 = 1 + \lceil \frac{x}{2} \rceil + \lceil \frac{y-1}{2} \rceil$. If we can change two odd bits, the length of the routing path is $d_7 = 1 + \lfloor \frac{x}{2} \rfloor + \lfloor \frac{y-2}{2} \rfloor + 1$.

Case c: 3-bit routing step. The length of the routing path is $d_8 = 1 + \lceil \frac{x-1}{2} \rceil + \lceil \frac{y-1}{2} \rceil$.We have that $d_1 \geq d_5$, $d_2 \geq d_7$, $d_3 \geq d_8$, $d_4 \geq d_8$, $d_6 \geq d_8$, and $d_5 = d_7 = d_8 = \lfloor \frac{x}{2} \rfloor + \lfloor \frac{y}{2} \rfloor + 1$. Thus, $H(i,j) = \min\{d_i | 1 \leq i \leq 8\} = \lfloor \frac{x}{2} \rfloor + \lfloor \frac{y}{2} \rfloor + 1$. Therefore, it is also true when $r_0 = 1$ and $One(R_{ij}) = k$. □

Theorem 7. *In $U(n,b)$, where $b = n-1$, the diameter is $D(n,b) = \lceil \frac{b+1}{2} \rceil$ if $b \geq 2$, and $D(n,b) = b+1$ if $0 \leq b \leq 1$.*

Proof. We can easily check the case that $n \leq 2$. For $n \geq 3$, if $(n-1) = 4m$, where m is a nonnegative integer, by Theorem 6, the maximum value of $H(i,j)$ is $\frac{n-1}{2} + 1 = \lceil \frac{n}{2} \rceil$. If $(n-1) = 4m+1$, suppose $Even(R_{ij}) = 2x$ and $Odd(R_{ij}) = 2y+1$ or $Even(R_{ij}) = 2x+1$ and $Odd(R_{ij}) = 2y$, where x, y are nonnegative integers and $2x + 2y + 1 = (n-1)$. The maximum value of $H(i,j)$ is $x + y + 1 = \frac{n}{2} = \lceil \frac{n}{2} \rceil$. The cases that $(n-1) = 4m+2$ and $(n-1) = 4m+3$ can be proved similarly. □

Suppose that the source is processor i and the destination is processor j. From the proof of Theorem 6, we obtain some rules to get the best neighbor processors of i, which is also our shortest routing algorithm. By Lemma 2, we make some restriction on the best neighbor processor h such that $One(R_{hj}) < One(R_{ij})$. By the proof of Theorem 6 for $One(R_{ij}) \leq 3$, we can obtain our shortest path routing algorithm. Thus, we present our shortest path routing algorithm for $One(R_{ij}) > 3$ in the following corollary.

Corollary 1. *In $U(n, n-1)$, suppose i is the source, j is the destination processor and $One(R_{ij}) > 3$. Any of following rules can be used to select one best neighbor of processor i if the condition is satisfied.*

1. *$r_0 = 0$: a 3-bit routing step, a 2-bit routing step, or a 1-bit routing step which changes bit x, where x is even if $Even(R_{ij})$ is odd and x is odd if $Odd(R_{ij})$ is odd.*
2. *$r_0 = 1$, $Even(R_{ij})$, and $Odd(R_{ij})$ are even: a 3-bit routing step, a 2-bit routing step, or a 1-bit routing step.*
3. *$r_0 = 1$, $Even(R_{ij})$, and $Odd(R_{ij})$ are odd: a 3-bit routing step, or a 2-bit routing step excluding changing bit 0.*
4. *$r_0 = 1$, $Even(R_{ij})$ is even, and $Odd(R_{ij})$ is odd: a 3-bit routing step, a 2-bit routing step excluding bit 0, or a 1-bit routing step which changes an even bit.*
5. *$r_0 = 1$, $Even(R_{ij})$ is odd, and $Odd(R_{ij})$ is even: a 3-bit routing step, a 2-bit routing step, or a 1-bit routing step which changes an odd bit.*

Proof. Let $Even(R_{ij}) = x$, $Odd(R_{ij}) = y$, and $x + y = One(R_{ij}) > 3$. We first prove case (1) that $r_0 = 0$. We consider the routing path using the following routing steps:

3-bit routing step or **2-bit routing step**: With the same proofs of cases 3 and 2 in Theorem 6, we can obtain the values of d_5, d_3, and d_4.

1-bit routing step: If $Even(R_{ij})$ is odd and we change an even bit, then the length of the routing path is $d_1 = 1 + \lceil \frac{x-1}{2} \rceil + \lceil \frac{y}{2} \rceil = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$. If $Odd(R_{ij})$ is odd and we change an odd bit, the length of the routing path is $d_2 = 1 + \lceil \frac{x}{2} \rceil + \lceil \frac{y-1}{2} \rceil = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$.

We have that $d_1 = d_2 = d_3 = d_4 = d_5 = \lceil \frac{x}{2} \rceil + \lceil \frac{y}{2} \rceil$. Thus, any of the above routing steps can be used to reach one best neighbor of processor i when $r_0 = 0$ and some conditions are satisfied.

The proofs of other cases can be obtained from the proof of Theorem 6 similarly. Thus, the proofs are omitted. □

For example, in $U(6,5)$, suppose the source processor $i = (000000)$ and the destination processor $j = (111110)$. Thus, we have $Even(R_{ij}) = 2$, $Odd(R_{ij}) = 3$, $r_0 = 0$, $One(R_{ij}) = 5 > 3$, and $H(i,j) = 3$. Since $r_0 = 0$, by Case 1, we can apply a 3-bit routing step, a 2-bit routing step, or a 1-bit routing step, which changes one odd bit x since $Odd(R_{ij}) = 3$ is odd.

If one 3-bit routing step is selected first, we can reach $k = (000111)$. Thus, we have $r_0 = 1$, $Even(R_{kj}) = 1$, $Odd(R_{kj}) = 2$, and $One(R_{kj}) = 4 > 3$. Since $Even(R_{kj})$ is odd, $Odd(R_{kj})$ is even and $r_0 = 1$, by Case 5, we can apply a 3-bit routing step, a 2-bit routing step, or a 1-bit routing step which changes an odd bit as follows:

- **1-bit routing step which changes an odd bit:**
  - case (a): If bit 5 is selected, we can reach $l = (100111)$ and $One(R_{lj}) = 3$. Because $One(R_{lj}) = 3 \leq 3$, we can apply one 3-bit routing step to reach (111110). Thus, $\{(000000), (000111), (100111), (111110)\}$ could be one shortest path whose length is 3.
  - case (b): If bit 3 is selected, we can obtain the shortest path $\{(000000), (000111), (001111), (111110)\}$.
- **3-bit routing step**: We can reach $l = (110110)$ and $One(R_{lj}) = 1 \leq 3$. Then, we apply a 1-bit routing step to reach (111110). Thus, $\{(000000), (000111), (110110), (111110)\}$ could be one shortest path.
- **2-bit routing step**: We can reach $l = (110111)$ and $One(R_{lj}) = 2 \leq 3$. Then, we apply a 2-bit routing step to reach (111110). Thus, $\{(000000), (000111), (110111), (111110)\}$ could be one shortest path.

If one 1-bit routing step is applied first, we can obtain some of shortest paths $\{(000000), (100000), (111000),$

(111110)} and {(000000), (100000), (100111), (111110)}. We omit the case that one 2-bit routing step is applied first.

5 Extension to an Arbitrary Value b of $U(n,b)$

In this section, we shall extend the properties of $U(n,b)$ to the case $0 \le b \le n-2$. By the constructing method of $U(n,b)$, we apply the fully doubling method in b times, and the partially doubling method in $(n-b-1)$ times. Therefore, there are $2^{n-b-1}U(b+1,b)$s in $U(n,b)$, and each $U(b+1,b)$ shares the common buses to each other. The processor $i=(i_{n-1},i_{n-2},\cdots,i_1,i_0)$ in $U(n,b)$ can be viewed as the processor $i'=(i_b,i_{b-1},\cdots,i_1,i_0)$ in $U(b+1,b)$, and the bits i_{n-1}, i_{n-2}, $\cdots$, i_{b+1} of processor i can be viewed as the binary index of the $U(b+1,b)$ which processor i belongs to.

Hence, the extension method is to reassign the processor identifier by excluding bit $(b+1)$ through bit $(n-1)$, and all properties in the previous section can also be obtained. Since the buses in $U(b+1,b)$ are the buses in $U(n,b)$, and the processors in $U(b+1,b)$ are a subset of processors in $U(n,b)$, for the routing paths on $U(b+1,b)$, we have to transform the $(b+1)$-bit identifiers of processors in $U(b+1,b)$ into n-bit identifiers of processors in $U(n,b)$. We can add arbitrary values in bit $(b+1)$ through bit $(n-1)$ in all identifiers of intermediate processors on the routing path connecting the source and the destination.

For example, in $U(7,4)$, by Theorem 3, processor $i=(0000000)$ is low, since the processor $i'=(00000)$ and $Zero(i')$ is odd. By Theorem 4, $B(i')=\{0000,0100,0001\}$. Assume processor $j=(1111111)$, then $j'=(11111)$. The distance between processor i and processor j in $U(7,4)$ is the same as that between processor i' and processor j' in $U(5,4)$. By Theorem 6, $H(i,j)=H(i',j')=3$. By our routing algorithm, we can obtain one shortest path from i to j in $U(5,4)$, which is $\{00000,01010,01011,11111\}$. It can be extended to one shortest path from i to j in $U(7,4)$, which is $\{0000000,0001010,0101011,1111111\}$, in which bit 5 through bit 6 of the intermediate processors are of arbitrary values.

By the above extension method, the diameter of $U(n,b)$ is the same as that of $U(b+1,b)$. Thus, we can extend Theorem 7 to the following theorem.

Theorem 8. *In $U(n,b)$, $0 \le b \le n-1$, the diameter is $D(n,b)=\lceil \frac{b+1}{2} \rceil$ if $b \ge 2$, and $D(n,b)=b+1$ if $0 \le b \le 1$.*

6 Fault Tolerance of the Bus-Based Hypercube

Hypercube networks have some good fault tolerance properties. Much interest has been paid on the hypercube network [3], [4], [18], [20], [21], [27]. Therefore, we are also interested in the fault tolerance properties of the bus-based hypercube network.

If a network graph with $n-1$ node failures is guaranteed to be connected, the network is said to have *node connectivity* of n. However, while the property is presented in a network, the network diameter may increase. One of the methods to judge the performance of a network with some faults is *fault diameter*. In $U(n,b)$, the fault diameter is defined as the maximum distance between two processors after some processors or buses are removed. The previous result on bus-fault diameter of $U(n,b)$ has been presented in Theorem 2 [32].

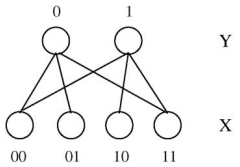


Fig. 5. A bipartite graph presentation of $U(2,1)$.

$U(n,b)$ can be represented as a bipartite graph $G=((X \cup Y),E)$, where X is the set of processors, Y is the set of buses and E represents the connections between processors and buses. For example, the bipartite graph to represent $U(2,1)$ is shown in Fig. 5. After mapping the bus-based hypercube into a bipartite graph G, we shall first present some properties of buses. And, then, we shall give the properties of node connectivity and the fault diameter in $U(n,b)$.

Lemma 3. *In $U(n,n-1)$, let $i=(i_{n-1},i_{n-2},\cdots,i_1)$ and $j=(j_{n-1},j_{n-2},\cdots,j_1)$ be two buses. Suppose $One(R_{ij})=1$, where $r_x=1$. If processor k is connected to both buses i and j, then one of the buses i and j must be the host bus of processor k, and*

$$k=\begin{cases}(i_{n-1},i_{n-2},\cdots,i_1,1)\ or\\ (j_{n-1},j_{n-2},\cdots,j_1,0) & if\ (n-x)+Zero(i)\ is\ odd,\\ (i_{n-1},i_{n-2},\cdots,i_1,0)\ or\\ (j_{n-1},j_{n-2},\cdots,j_1,1) & otherwise.\end{cases}$$

Proof. j can be rewritten as $(i_{n-1},i_{n-2},\cdots,\overline{i_x},\cdots,i_1)$, where $1 \le x \le n-1$. If none of buses i and j is the host bus of k, then i and j are guest buses of k. By Theorem 4, i and j must be different in two bits. This is contradictory to that $One(R_{ij})=1$. Hence, one of i and j must be the host bus of k.

We first assume that $n-x$ is odd. We have to consider the following cases.

Case 1: $Zero(i)$ is even. By Theorem 4, k must be a high processor. If i is the host bus of k, by Theorem 3, $k=(i_{n-1},i_{n-2},\cdots,i_1,1)$ since $Zero(i)$ is even and k is high. If j is the host bus of k, then $k=(i_{n-1},i_{n-2},\cdots,\overline{i_x},\cdots,i_1,0)=(j_{n-1},j_{n-2},\cdots,j_1,0)$ since $Zero(j)$ is odd and k is high.

Case 2: $Zero(i)$ is odd. Similarly, if i is the host bus of k, then $k=(i_{n-1},i_{n-2},\cdots,i_1,0)$. And, if j is the host bus of k, then $k=(j_{n-1},j_{n-2},\cdots,j_1,1)$.

The case that $n-x$ is even can be proved similarly.□

Lemma 4. *In $U(n,n-1)$, let $i=(i_{n-1},i_{n-2},\cdots,i_1)$ and $j=(j_{n-1},j_{n-2},\cdots,j_1)$ be two buses. Suppose $One(R_{ij})=2$ and $r_x=r_y=1$, $x>y$. Buses i and j are connected to a common processor k if and only if $n-x$ and $n-y$ are both even or both odd. Moreover, both buses i and j must be the guest buses of processor k, and*

$$k=\begin{cases}(i_{n-1},\cdots,\overline{i_x},\cdots,i_1,0)\ or\\ (i_{n-1},\cdots,\overline{i_y},\cdots,i_1,0) & if\ (n-x)+Zero(i)\ is\ odd,\\ (i_{n-1},\cdots,\overline{i_x},\cdots,i_1,1)\ or\\ (i_{n-1},\cdots,\overline{i_y},\cdots,i_1,1) & otherwise.\end{cases}$$

Proof. Rewrite j as $(i_{n-1}, i_{n-2}, \cdots, \overline{i_x}, \cdots, \overline{i_y}, i_1)$. By Theorem 4, if $n-x$ and $n-y$ are not both even or not both odd, buses i and j cannot be connected to a common processor. If one of i and j is the host bus of processor k, by Theorem 4, $One(R_{ij}) = 1$, which is contradictory to that $One(R_{ij}) = 2$. Hence, both i and j are the guest buses of processor k. Thus, by Theorem 4, the leftmost $n-1$ bits of $Bin(k, n)$ must be different from i and j exactly at one bit.

We first suppose that both $n-x$ and $n-y$ are odd. By Theorem 4, k must be a high processor. If $Zero(i)$ is even, then $Zero(j)$ is also even. By Theorem 3 and Theorem 4, $k = (i_{n-1}, i_{n-2}, \cdots, \overline{i_x}, \cdots, i_1, 0)$, or $k = (i_{n-1}, i_{n-2}, \cdots, \overline{i_y}, \cdots, i_1, 0)$. If $Zero(i)$ is odd, $Zero(j)$ is also odd, then $k = (i_{n-1}, i_{n-2}, \cdots, \overline{i_x}, \cdots, i_1, 1)$, or $k = (i_{n-1}, i_{n-2}, \cdots, \overline{i_y}, \cdots, i_1, 1)$.

The case that both $n-x$ and $n-y$ are even can be proved similarly. □

These two properties, Lemma 3 and Lemma 4, give us some ideas to find *disjoint paths* from source i to destination j. Here, the disjoint paths that we consider are both *bus* and *processor disjoint*.

A path from processor i to processor j can be represented by a sequence of buses, starting with the bus which processor i is connected to and ending with the bus which processor j is connected to. For example, in $U(5,4)$, if the source processor is 00000 and the destination processor is 11111, the path $\{P00000, b0000, P00001, b0010, P00110, b0111, P11111\}$ can be simplified as a sequence of buses $\{b0000, b0010, b0111\}$, where every element attached with P denotes a processor, and attached with b denotes a bus. Two buses are said to be *adjacent* if they are connected to a common processor. For example, $b0000$ and $b0010$ are adjacent on the above path.

By Lemma 3, processor $P00001$ is connected to buses $b000\underline{0}$ and $b001\underline{0}$, in which bus $b0000$ is the host bus of processor $P00001$. In other words, there should exist a processor that we can change $b000\underline{0}$ to $b001\underline{0}$ at bit 2 or other bit. By Lemma 4, buses $b0\underline{0}1\underline{0}$ and $b0\underline{1}1\underline{1}$ are connected to a common processor $P00110$. Both buses, $b0\underline{0}1\underline{0}$ and $b0\underline{1}1\underline{1}$, are the guest buses of processor $P00110$. It means that there should exist a processor to change $b0\underline{0}1\underline{1}$ and $b0\underline{1}1\underline{1}$ at two bits, bits 1 and 3, or other two bits that are both even or both odd.

Let $i = (i_{n-1}, i_{n-2}, \cdots, i_1, i_0)$. By Theorem 4,

$$B(i) = \{b_0(i)\} \cup \{\bar{b}_s(i) | n-s \text{ is odd if } i \text{ is high}, \; n-s \text{ is even if } i \text{ is low}\}.$$

Let $b_k(i)$ denote the kth guest bus of i, $1 \le k \le m$, where m is the fan-out of processor i. Then, $b_k(i) = \bar{b}_s(i) = (i_{n-1}, \cdots, \overline{i_s}, \cdots, i_1)$, where $k = \lceil \frac{s}{2} \rceil$, $(n-s)$ is odd if i is high, and $(n-s)$ is even if i is low.

For example, in $U(7,6)$, consider a high processor $i = (1111111)$, and a low processor $j = (1000001)$. Then,

$$\begin{aligned} B(i) &= \{111111\} \cup \{1111\underline{0}1, 11\underline{0}111, \underline{0}11111\} \\ &= \{b_0(i)\} \cup \{b_1(i), b_2(i), b_3(i)\} \text{ and} \\ B(j) &= \{100000\} \cup \{10000\underline{1}, 100\underline{1}00, 1\underline{1}0000\} \\ &= \{b_0(j)\} \cup \{b_1(j), b_2(j), b_3(j)\}. \end{aligned}$$

Note that $b_0(i) = (111111)$ and $b_0(j) = (100000)$ are the host buses for processors i and j, respectively. The guest buses of processor i are $b_1(i) = (1111\underline{0}1)$, $b_2(i) = (11\underline{0}111)$, and $b_3(i) = (\underline{0}11111)$. The guest buses of processor j are $b_1(j) = (10000\underline{1})$, $b_2(j) = (100\underline{1}00)$, and $b_3(j) = (1\underline{1}0000)$.

Lemma 5. *In $U(n, n-1)$, let i and j be two processors. If there exist k paths whose buses are all distinct and the number of different bits between every two adjacent buses on one path is one, then the k paths are processor disjoint.*

Proof. Since the number of different bit between two adjacent buses b_1 and b_2 on one path is one, by Lemma 3, we can select the intermediate processor P between these two buses such that b_1 is the host bus of P. Thus, the binary identifier of P with removing the rightmost bit is the same as that of b_1. Since the buses in the k paths are all distinct, then all intermediate processors are distinct. □

By Lemma 5, if we can find k paths from a source processor i to a destination processor j whose buses are all distinct and the difference between every two adjacent buses is one bit, then the k paths are processor disjoint between i and j. Before presenting the method to find the disjoint paths between i and j, we shall give some terms and properties of buses.

Let $(i_{n-1}, i_{n-2}, \cdots, i_1)$ be a bus in $U(n, b)$. The *even-half identifier* consists of each i_x, where x is even, and the *odd-half identifier* consists of each i_x, where x is odd. Note that the rightmost bit of a bus identifier is counted as bit 1. In $U(n, n-1)$, by Theorem 4, if $n + High(i)$ is odd (even), each bus of a processor i has a unique pattern on the odd-half (even-half) identifier. Therefore, no matter how we change the even-half identifier of each bus, all buses are still distinct. With this property, we can find the bus disjoint paths between i and j. For example, in $U(7,6)$, consider processor $i = (0000000)$. We have $B(i) = \{\underline{0}0\underline{0}0\underline{0}0, \underline{0}0\underline{0}0\underline{0}1, \underline{0}0\underline{0}1\underline{0}0, \underline{0}1\underline{0}0\underline{0}0\}$. The even-half identifiers of these buses are $\underline{000}, \underline{000}, \underline{000}, \underline{000}$, and the odd-half identifiers of these buses are $\{000, 001, 010, 100\}$, which are all distinct, since $n + High(i)$ is odd.

We use $B_k(i,j)$ to denote the bus sequence on one routing path between $b_k(i)$ and $b_k(j)$, $0 \le k \le m$, where m is the minimum of the numbers of buses which processor i and processor j are connected to, respectively. Note that $b_k(i)$ and $b_k(j)$ are also included in $B_k(i,j)$. We also assume that $b_k(i) \neq b_h(j)$ if $k \neq h$. Suppose, otherwise, we exchange the roles of $b_k(j)$ and $b_h(j)$, and put $b_k(i)$ and $b_h(j)$ into a pair, and $b_h(i)$ and $b_k(j)$ into a pair. That is, $B_k(i,j)$ is changed to denote the set of buses between $b_k(i)$ and $b_h(j)$. After this change, $B_k(i,j)$ contains only $b_k(i)$ since $b_k(i) = b_h(j)$.

For example, consider a high processor $i = (1111111)$ and a low processor $j = (0110110)$. Processors i and j have different states. Then,

$$\begin{aligned} B(i) &= \{111111, 1111\underline{0}1, 11\underline{0}111, \underline{0}11111\} \\ &= \{b_0(i), b_1(i), b_2(i), b_3(i)\} \text{ and} \\ B(j) &= \{011011, 01101\underline{0}, 011\underline{1}11, 0\underline{0}1011\} \\ &= \{b_0(j), b_1(j), b_2(j), b_3(j)\}. \end{aligned}$$

Comparing $B(i)$ with $B(j)$, we find $b_3(i) = b_2(j)$. After exchanging $b_2(j)$ and $b_3(j)$, we have

$$\begin{aligned} B(i) &= \{111111, 11110\underline{1}, 110\underline{1}11, \underline{0}11111\} \\ &= \{b_0(i), b_1(i), b_2(i), b_3(i)\}, \\ B(j) &= \{011011, 01101\underline{0}, 00\underline{1}011, 011\underline{1}11\} \\ &= \{b_0(j), b_1(j), b_2(j), b_3(j)\}, \text{ and} \\ B_k(i,j) &= \{B_0(i,j), B_1(i,j), B_2(i,j), B_3(i,j)\}. \end{aligned}$$

The paths between i and j are $B_0(i,j) = \{111111, \cdots, 011011\}$, $B_1(i,j) = \{11110\underline{1}, \cdots, 01101\underline{0}\}$, $B_2(i,j) = \{110\underline{1}11, \cdots, 00\underline{1}011\}$, and $B_3(i,j) = \{011111\}$.

By Lemmas 3 and 5, we can construct the paths without considering which processor connected between the buses on the paths. And, the buses and processors on the paths are all distinct. For example, we construct $B_1(i,j)$ by changing $b_1(i)'s$ even bits to match $b_1(j)'s$ even bits bit by bit and then matching odd bits bit by bit. Thus, we have $B_1(i,j) = \{b_1(i) = 11110\underline{1}, 11110\underline{1}1, \underline{0}1110\underline{1}1, \underline{0}1110\underline{1}0, \underline{0}1101\underline{0}0 = b_1(j)\}$. The difference between every two adjacent buses on one path is one bit. Then, all paths are processor disjoint.

Furthermore, we can change two odd bits (two even bits) each step by Lemma 4. The change is more effective than it by Lemma 3. For example, we can construct $B_2(i,j)$ by changing $b_2(i)'s$ two odd bits to match $b_2(j)'s$ two odd bits per step and then matching two even bits per step. Thus, we have $B_2(i,j) = \{b_2(i) = 110\underline{1}11, 10\underline{00}11, \underline{00}\underline{10}11 = b_2(j)\}$. The buses on the paths are all distinct but the processors may not be.

Theorem 9. *In $U(n,b)$, $b = n-1$, $b \geq 2$, there are $(m-1)$ bus disjoint and processor disjoint paths between any pair of processors where m is the minimum value of fan-outs of the two processors.*

Proof. We shall provide a systematic way to construct all bus path $B_k(i,j)$s, $1 \leq k \leq m-1$, and prove that they are disjoint. Let $i = (i_{n-1}, i_{n-2}, \cdots, i_1, i_0)$ and $j = (j_{n-1}, j_{n-2}, \cdots, j_1, j_0)$, $i \neq j$, denote the source processor and the destination processor, respectively. There are two phases to change different bits in bus sequence $B_k(i,j)$, one is the *even-phase* and the other is the *odd-phase*. In the even-phase (odd-phase), two adjacent buses in $B_k(i,j)$ are different in an even (odd) bit. And, the different bits are corrected one by one from the rightmost even (odd) bit to the leftmost even (odd) bit. We have to consider the following two cases.

Case 1: Processors i and j have different states. Without loss of generality, we assume that the buses in $B(i)$ are all distinct in their odd-half identifiers and $b_k(i)$ is unique in bit $2k-1$. Then, by Theorem 4, the buses in $B(j)$ are all distinct in their even-half identifiers, and $b_k(j)$ is unique in bit $2k$. By Lemma 3, to change $b_k(i)$ to $b_k(j)$ bit by bit, we can perform the even-phase and then the odd-phase. If two adjacent buses are the same, nothing is done. In the even-phase, bit $2k$ of $b_k(i)$ is changed to be bit $2k$ of $b_k(j)$ at the first step. (If bit $2k$ of $b_k(i)$ is equal to bit $2k$ of $b_k(j)$, nothing is done.) Then all other even-half identifiers are corrected from right to left. After the even-phase finishes, all buses are also distinct in the even-half identifiers. Then, the odd-phase begins, and we change the odd different bits from right to left, but skip over bit $2k-1$ until the last step. Bit $2k-1$ in $B_k(i,j)$ is changed in the last step if it needs to be corrected.

In the above routing method, in fact, the last bus in the even-phase is also the first bus of the odd-phase in $B_k(i,j)$. Obviously, all buses on all paths in the even-phase are distinct in their odd-half identifiers. And, all buses on all paths in the odd-phase are distinct in their even-half identifiers. We want to prove that each bus in the even-phase is also different from each bus in the odd-phase on different routing paths.

Without loss of generality, suppose buses u and v are in the even-phase of $B_t(i,j)$ and the odd-phase of $B_s(i,j)$, respectively, $t > s \geq 1$. We have to consider the following subcases:

1. $u = b_t(i)$ and $v = b_s(j)$. By the definition of $B_t(i,j)$ and $B_s(i,j)$, $u \neq v$.
2. $u = b_t(i)$ and $v \neq b_s(j)$. $b_t(i)$ and $b_s(i)$ are different at least in odd bit $2s-1$. Since in $B_s(i,j)$, bit $2s-1$ may be changed only in the last routing step of the odd phase, $b_s(i)$ and v are equal at least in bit $2s-1$. Therefore, $b_t(i)$ and v are different at least in bit $2s-1$.
3. $u \neq b_t(i)$ and $v \neq b_s(j)$. In the even-phase of $B_t(i,j)$, u and $b_t(i)$ are equal at least in odd bit $2s-1$. $b_t(i)$ and v are different at least in bit $2s-1$. Hence, u and v are different at least in bit $2s-1$.
4. $u \neq b_t(i)$ and $v = b_s(j)$. Similar to subcase 2, $b_s(j)$ and u are different at least in bit $2t$.

Hence, the longest length of $B_k(i,j)$ is b.

Case 2: i and j have the same states. And, assume that b is even. Without loss of generality, we assume $B(i)$ and $B(j)$ have unique patterns in their even-half identifiers. By Lemma 3, we can apply one routing step to change each bus in $B(i)$ to have a unique pattern in the odd-half identifier. We change $b_k(i)$ in bit $2k-1$, denoted as $b'_k(i)$. And, the bus sequence from $b'_k(i)$ to $b_k(j)$ is denoted as $B'_k(i,j)$. Now, $b'_k(i)$ has a unique pattern in the odd-half identifier. Then, we can apply the method in case 1 to construct the routing path for $B'_k(i,j)$. Thus, all buses in all $B'_k(i,j)'s$ are still distinct. We have to prove only that each $b_k(i)$ is different from each bus u in the even-phase and each bus v in the odd-phase in $B'_h(i,j)$, $h \neq k$. $b_k(i)$ and u are different at least in bit $2h-1$. If $v \neq b_h(j)$, $b_k(i)$ and v are also different at least in bit $2h-1$ since in $B'_h(i,j)$, bit $2h-1$ may be changed only in the last routing step of the odd phase. If $v = b_h(j)$, we have $b_k(i) \neq v$ since $b_k(i) \neq b_h(j)$ by the definition of $B_k(i,j)$ and $B_h(i,j)$. Therefore, the longest path length is $b+1$ since one step is added to change $b(i)$ to have the unique odd-half identifiers. The case that b is odd can be proved similarly.

Since these $m-1$ routing paths are bus disjoint and every two adjacent buses differ in only one bit, by Lemma 5, these routing paths are also processor disjoint. That is, all processors in all $B_k(i,j)$s are disjoint. □

By the proof of Theorem 9, we have the following corollary.

Corollary 2. *In $U(n,b)$, $b = n-1$, $b \geq 2$, for all $0 \leq f \leq \lceil \frac{b-3}{2} \rceil$, the fault diameter $DF(n,b,f) \leq b+1$, where f is the sum of bus faults and processor faults.*

For example, in $U(7,6)$, consider processors $i = (0000000)$, $j = (0100011)$, and $x = (1111111)$. Note that i and j have different states, and x and j are of the same state. Table 2

TABLE 2
Two Examples for Constucting Disjoint Paths

Disjoint path	$B_1(i,j)$	$B_2(i,j)$	$B_3(i,j)$	$B_1(x,j)$	$B_2(x,j)$	$B_3(x,j)$
Starting bus	$b_1(i) =$ 000001	$b_2(i) =$ 000100	$b_3(i) =$ 010000	$b_1(x) =$ 111101 $b'_1(x) =$ 111100	$b_2(x) =$ 110111 $b'_2(x) =$ 110011	$b_3(x) =$ 011111 $b'_3(x) =$ 001111
Even phase	000011 000011 000011	001100 001100 001100	110000 110000 110000	111110 110110 010110	111011 111001 011001	101111 101101 100101
Odd phase	000011 010011 010011	001101 011101 011001	110001 110001 110001	010010 010010 010011	011001 011001 011001	100101 100001 110001
Ending bus	010011 $= b_1(j)$	011001 $= b_2(j)$	110001 $= b_3(j)$	010011 $= b_1(j)$	011001 $= b_2(j)$	110001 $= b_3(j)$

shows three disjoint paths between processors i and j and also three disjoint paths between processors x and j.

By Lemma 4, in $U(n,b)$, $b = n-1$, if buses i and j are neighbors, and $One(R_{ij}) = 2$, then the two different bit positions are both even or both odd. Therefore, we can extend Theorem 9 to the bus disjoint paths in which every two adjacent buses are different in two bits if it can be done. If we do not require that the paths are both bus disjoint and processor disjoint, then we can find m bus disjoint paths in the following theorem. Note that the paths found in Theorem 9 are both bus disjoint and processor disjoint.

Theorem 10. *In $U(n,b)$, $b = n-1$, $b \geq 4$, there are m bus disjoint paths between any pair of processors where m is the minimum value of fan-outs of the two processors.*

Proof. The proof is similar to that of Theorem 9. We shall construct all bus path $B_k(i,j)$s, $0 \leq k \leq m-1$, and then prove that they are bus disjoint. Let $i = (i_{n-1}, i_{n-2}, \cdots, i_1, i_0)$ and $j = (j_{n-1}, j_{n-2}, \cdots, j_1, j_0)$, $i \neq j$, denote the source processor and the destination processor, respectively. And, let $b_1 = \lceil \frac{n-1}{2} \rceil$ and $b_2 = \lfloor \frac{n-1}{2} \rfloor$. In the even-phase (odd-phase), from one bus to the next intermediate bus, by Lemma 4, we change two even bits (two odd bits) each step if two or more different bits are left. Thus, every two adjacent buses in $B_k(i,j)$ are different in two even (odd) bits except one pair of adjacent buses are possibly not. And, the different bits are corrected from the rightmost even (odd) bit to the leftmost even (odd) bit. We have to consider the following two cases.

Case 1: Processors i and j have different states. Without loss of generality, we assume $B(i)$ has unique patterns in the odd-half identifiers. At first, we change $b_0(i)$ in two odd bits, bits 1 and 3, denoted as $b'_0(i)$. We stipulate that the unique pattern of $b'_0(i)$ is bit 1 and bit 3. We also denote the bus obtained by changing $b_0(j)$ in bits 2 and 4 as $b'_0(j)$. Note that $b'_0(j)$ will be changed to $b_0(j)$ in the last step, not in the first step. Then, we construct the m bus disjoint paths similar to the methods in Theorem 9. In the first step of the even-phase, we change bit $2k$ of $b_k(i)$ to be bit $2k$ of $b_k(j)$, $k \geq 1$, and bits 2 and 4 of $b'_0(i)$ to bits 2 and 4 of $b'_0(j)$ (a special changing method for some special subcases will be given more detailedly in subcase 4 later). Then, the even-phase begins, the even-half identifiers are corrected in two even bits per step. And, then, the odd-phase beings, we apply the same method to correct the odd-half identifiers. Bit $2k-1$ in $B_k(i,j)$, $k \geq 1$, is changed in the last step if it needs to be corrected.

Without loss of generality, suppose u and v are in the even-phase of $B_t(i,j)$ and the odd-phase of $B_s(i,j)$, respectively, $t > s \geq 0$. When $t > s > 0$, the proof is similar to that of Theorem 9. Hence, we consider only the case $s = 0$.

1. $u = b_t(i)$ and $v \neq b'_0(j)$. $b_t(i)$ and $b'_0(i)$ are different at least in bit 1 or 3. $b'_0(i)$ and v are same in bits 1 and 3. Therefore, $b_t(i)$ and v are different at least in bit 1 or 3.
2. $u \neq b_t(i)$ and $v \neq b'_0(j)$. $b_t(i)$ and v are different at least in bit 1 or 3. By the even-phase, $b_t(i)$ and u are same in odd bits. Hence, u and v are different at least in bit 1 or 3.
3. $u \neq b_t(i)$ and $v = b'_0(j)$. It can be proved similarly.
4. $u = b_t(i)$ and $v = b'_0(j)$. If $One(R_{b_0(i)b_0(j)}) \geq 4$, then obviously, $One(R_{uv}) \geq 1$.

It means that $u \neq v$. If $One(R_{b_0(i)b_0(j)}) = 1$, they are connected by a common processor directly. If $One(R_{b_0(i)b_0(j)}) = 3$ or 2, and the different bits are not all in bits 1, 2, 3, and 4, then after changing $b_0(j)$ in bits 2 and 4, $One(R_{uv}) \geq 1$. Otherwise, we can change $b_0(j)$ to $b'_0(j)$ in bits 1, 2, 3, and 4 bit by bit in some order such that $B_0(i,j)$ has no common bus with $B_k(i,j)$, $1 \leq k \leq m-1$ (we omit the detail here). In summary, $u \neq v$.

Hence, the longest length of $B_k(i,j)$ is $1 + \lceil \frac{b_1-1}{2} \rceil + \lceil \frac{b_2-1}{2} \rceil + 1 \leq \lfloor \frac{b}{2} \rfloor + 2$.

The case that u is in the odd-phase of $B_t(i,j)$ and v is in the even-phase of $B_0(i,j)$ can be proved similarly.

Case 2: i and j have the same states. And, assume that b is even. Without loss of generality, we assume $B(i)$ and $B(j)$ have unique patterns in their even-half identifiers. By Lemma 4, we change $b_0(i)$ in bits 1 and 3, denoted as $b'_0(i)$. And, by Lemma 3, we change each $b_k(i)$, $k \neq 0$, in bit $2k-1$, denoted as $b'_k(i)$. The bus sequence from $b'_k(i)$ to $b_k(j)$ is denoted as $B'_k(i,j)$. At this time, $b'_k(i)$, $0 \leq k \leq m$, has a unique pattern in the odd-half identifier. Then, we apply the method in case 1 to construct the routing path. We first

TABLE 3
An Example for Constucting Bus Disjoint Paths between Two Processors with Different States

Disjoint path	$B_0(i,j)$	$B_1(i,j)$	$B_2(i,j)$	$B_3(i,j)$	$B_4(i,j)$
Starting bus	$00000000 = b_0(i)$ $00000\underline{1}0\underline{1} = b'_0(i)$	$b_1(i) =$ 00000001	$b_2(i) =$ 00000100	$b_3(i) =$ 00010000	$b_4(i) =$ 01000000
Even phase	$0000\underline{1}11\underline{1}$ $\underline{1}0001111$	$00000\underline{1}1$ $\underline{1}0000011$	$0000\underline{1}100$ $\underline{1}0001100$	$00\underline{1}10000$ $\underline{1}0110000$	$\underline{0}1000000$
Odd phase	$1\underline{1}01\underline{1}1111$ $11011\underline{0}11$	$1\underline{1}010011$ $1101001\underline{1}$	10011101 $1\underline{1}011\underline{0}01$	$1\underline{1}110001$ $111\underline{1}0001$	$0101000\underline{1}$ $0\underline{1}010001$
Ending bus	$1101\underline{1}0\underline{1}1 = b'_0(j)$ $11010001 = b_0(j)$	11010011 $= b_1(j)$	11011001 $= b_2(j)$	11110001 $= b_3(j)$	01010001 $= b_4(j)$

perform the even-phase and then the odd-phase. Thus, all buses in all $B'_k(i,j)'s, 0 \le k \le m-1$, are still distinct. And, the proof that each $b_k(i)$ is different from each bus in $B'_h(i,j), h \ne k$, is similar to that of Case 1 in this theorem and that of Case 2 in Theorem 9.

The length of the longest paths is not greater than $\lfloor \frac{b}{2} \rfloor + 3$ since one step is added to change $B(i)$ to have unique odd-half identifier. □

For example, in $U(9,8)$, assume processors $i = (000\ 000000)$, $j = (110100011)$, and $x = (111111111)$. Note that i and j have different states, and x and j are of the same state. Table 3 shows five bus disjoint paths between processors i and j, and Table 4 shows five bus disjoint paths between processors x and j.

By the proof of Theorem 10, we also have the following corollary.

Corollary 3. *In $U(n,b)$, $b = n-1$, $b \ge 4$, for all $0 \le f \le \lceil \frac{b-1}{2} \rceil$, the fault diameter $DB(n,b,f) \le \lfloor \frac{b}{2} \rfloor + 3$, where f is the number of bus faults.*

Theorem 9 and Theorem 10 can be extended to the case $b < n-1$ easily. The $U(n,b)$, $b < n-1$, can be regarded as 2^{n-b-1} $U(b+1,b)'s$. Suppose processors $i = (i_{n-1}, \cdots, i_1, i_0)$ and $j = (j_{n-1}, \cdots, j_1, j_0)$ in $U(n,b)$. The disjoint paths between i and j can be regarded as the disjoint paths between processors $i' = (i_b, i_{b-1}, \cdots, i_1, i_0)$, and $j' = (j_b, j_{b-1}, \cdots, j_1, j_0)$. And, from bit $(b+1)$ through bit $(n-1)$ of each intermediate processor can be of arbitrary values, except the source and the destination processors are i and j, respectively. Therefore, by Theorem 9, the diameter $DF(n,b,f)$ in $U(n,b)$, $b \le n-1$, is no more than $b+1$. Similarly, Theorem 10 is also true when $b \le n-1$.

Theorem 11. *In $U(n,b)$, $2 \le b \le n-1$, for all $0 \le f \le \lceil \frac{b-3}{2} \rceil$, the fault diameter $DF(n,b,f) \le b+1$, where f is the sum of bus faults and processor faults.*

Theorem 12. *In $U(n,b)$, $4 \le b \le n-1$, for all $0 \le f \le \lceil \frac{b-1}{2} \rceil$, the fault diameter $DB(n,b,f) \le \lfloor \frac{b}{2} \rfloor + 3$, where f is the number of bus faults.*

Vaidyanathan and Padmanabhan [32] showed that in $U(n,b)$ the fault diameter $DB(n,b,f) \le b+1+2f$, where the f is the number of bus faults, and $0 \le f \le \lceil \frac{b-1}{2} \rceil$. Note that in their result, only bus faults are considered. Here, we have shown that $DF(n,b,f) \le b+1$, where $0 \le f \le \lceil \frac{b-3}{2} \rceil$ and f is the sum of bus faults and processor faults. We also have shown that $DB(n,b,f) \le \lfloor \frac{b}{2} \rfloor + 3$, where $0 \le f \le \lceil \frac{b-1}{2} \rceil$ and f is the number of bus faults. Thus, our results has significant improvements on their results.

7 Conclusion

By observing the methods of constructing the bus-based hypercube $U(n,b)$, we investigate some architecture properties of $U(n,b)$. Given a processor i in $U(n,b)$, by the binary identifier of i, the state of i, high or low, can be determined. Then, we obtain the buses which processor i is connected to.

TABLE 4
An Example for Constucting Bus Disjoint Paths between Two Processors with Same State

Disjoint path	$B_0(x,j)$	$B_1(x,j)$	$B_2(x,j)$	$B_3(x,j)$	$B_4(x,j)$
Starting bus	$11111111 = b_0(x)$ $11111\underline{0}1\underline{0} = b'_0(x)$	$b_1(x) =$ 11111101 $b'_1(x) =$ $1111110\underline{0}$	$b_2(x) =$ 11110111 $b'_2(x) =$ $1111\underline{0}011$	$b_3(x) =$ 11011111 $b'_3(x) =$ $110\underline{0}1111$	$b_4(x) =$ 01111111 $b'_4(x) =$ $0\underline{0}111111$
Even phase	$1111\underline{1}0\underline{1}0$ $11\underline{0}11010$	$111111\underline{1}0$ $11\underline{0}1\underline{0}0110$	$1111\underline{1}011$ $11\underline{0}110\underline{0}1$	$11\underline{1}01111$ $1110\underline{0}1\underline{0}1$	$\underline{0}0111111$ $0011\underline{0}1\underline{0}1$ $00\underline{0}10101$
Odd phase	11011011	$11010\underline{0}10$ $1101001\underline{1}$	$11011\underline{0}01$	$11100\underline{0}01$ $111\underline{1}0001$	$00010\underline{0}01$ $0\underline{1}010001$
Ending bus	$1101\underline{1}0\underline{1}1 = b'_0(j)$ $11010001 = b_0(j)$	11010011 $= b_1(j)$	11011001 $= b_2(j)$	11110001 $= b_3(j)$	01010001 $= b_4(j)$

TABLE 5
Some Properties of Three Networks

Network	N	Degree	Diameter	C	B
$SBH(w,D)$	w^D	$\log_w N$	$\log_w N$	w	$\frac{ND}{w}$
$SMLH(w,n)$	$w^2 2^n$	$2+\log_2 \frac{N}{w^2}$	$2+\log_2 \frac{N}{w^2}$	w	$4w2^{n-1}$
$U(n,b)$	2^n	b	$\lceil \frac{b+1}{2} \rceil$	$(b+2)2^{n-b-1}$	2^b

N: number of nodes C: number of nodes connected to each bus B: number of buses

Furthermore, the neighbor processors of i can also be obtained. In an n-dimensional hypercube, the diameter is n. Due to the multiple-access property of the media in MBN, the diameter in an MBN become smaller. From the properties of neighbor processors, we present a routing algorithm to transmit messages from the source processor to the destination processor via one shortest path, and the diameter is shown to be $\lceil \frac{b+1}{2} \rceil$ in $U(n,b)$.

In Table 5, we summarize some basic properties in the systems built from cube-like topology with some shared buses, which include the spanning multichannel linked hypercube (SMLH) [23], [22], the spanning bus hypercube (SBH) [35], and the $U(n.b)$. Parts of them were presented by Louri et al. [23], [22].

Fault tolerance is also an important property of multiprocessor systems. One of the important measures of fault tolerance in an interconnection network is the connectivity of the underlying graph. One of the methods to judge the performance of a network with some faults is fault diameter. We present a method to find $(m-1)$ bus disjoint and processor disjoint paths between the source processor and the destination processor, where m is the minimum value of fan-outs of the source processor and the destination processor. In $U(n,b)$, we show that the diameter $DF(n,b,f) \leq b+1$, where f is the sum of bus faults and processor faults and $0 \leq f \leq \lceil \frac{b-3}{2} \rceil$. We also show that $DB(n,b,f) \leq \lfloor \frac{b}{2} \rfloor + 3$, where $0 \leq f \leq \lceil \frac{b-1}{2} \rceil$ and f is the number of bus faults. Therefore, the result is superior to that was proposed by Vaidyanathan and Padmanabhan [32]. Recently, the use of MBNs has been studied extensively, we hope that there will be more and more further studies in MBNs.

ACKNOWLEDGMENTS

This research work was partially supported by the National Science Council of the Republic of China under NSC-87-2213-E-110-020.

REFERENCES

[1] A. Ali and R. Vaidyanathan, "Exact Bounds n Running ASCEND/DESCEND and FAN-IN Algorithms on Synchronous Multiple Bus Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 8, pp. 783-790, Aug. 1996.

[2] L.N. Bhuyan and D.P. Agrawal, "Generalized Hypercube and the Hyperbus Structures for a Computer Network," *IEEE Trans. Computers,* vol. 33, no. 4, pp. 323-333, Apr. 1984.

[3] J. Bruck, R. Cypher, and C. Ho, "On the Construction of Fault-Tolerant Cube-Connected Cycles Networks," *J. Parallel and Distributed Computing,* vol. 25, pp. 98-106, 1995.

[4] Y.C. Chen, "Designing Efficient Parallel Algorithms on Mesh-Connected Computers with Multiple Broadcasting," *IEEE Trans. Parallel and Distributed Systems,* vol. 1, no. 2, pp. 241-246, Apr. 1990.

[5] S. Cheung and F.C.M. Lau, "Routing with Locality on Meshes with Buses," *J. Parallel and Distributed Computing,* vol. 33, pp. 84-90, 1996.

[6] H.P. Dharmasena and R. Vaidyanathan, "Lower Bounds on the Loading of Degree-2 Multiple Bus Networks for Binary-Tree Algorithms," *Proc. 13th Int'l and 10th Symp. Parallel and Distributed Processing, IPPS/SPDP,* pp. 21-25, 1999.

[7] O.M. Dighe, R. Vaidyanathan, and S. Zheng, "Bus-Based Tree Structure for Efficeint Parallel Computation," *Proc. 1993 Int'l Conf. Parallel Processing,* pp. 158-161, 1993.

[8] O.M. Dighe, R. Vaidyanathan, and S. Zheng, "The Bus-Connected Ringed Tree: A Versatile Interconnection Network," *J. Parallel and Distributed Computing,* vol. 33, pp. 189-196, 1996.

[9] P.W. Dowd and K. Jabbour, "Spanning Multiaccess Channel Hypercube Computer Interconnection," *IEEE Trans. Computers,* vol. 37, pp. 1137-1142, 1988.

[10] A.H. Esfahanian, L.M. Ni, and B.E. Sagan, "The Twisted N-Cube with Application to Multiprocessing," *IEEE Trans. Computers,* vol. 40, no. 1, pp. 88-93, 1991.

[11] L.C. Fan, "Routing Algorithms on the Bus-Based Hypercube Network," MS thesis, Nat'l Sun Yat-sen Univ., Kaohsiung, Taiwan 804, 1997.

[12] C.M. Fiduccia, "Bused Hypercube and Other Pin-Optimal Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 3, no. 1, pp. 14-24, Jan. 1992.

[13] T. Ishikawa, "CCT-Cube: A Highly Parallel Network Featuring Short Diameter and Few Links," *Inst. of Electronics, Information and Comm. Engineers,* vol. J73-D-I, no. 6, pp. 559-602, 1990.

[14] T. Ishikawa, "Hypecube Multilprocessor with Bus Connections for Improving Communication Performance," *IEEE Trans. Computers,* vol. 44, no. 11, pp. 1338-1344, 1995.

[15] K. Jungjoon and A. El-Amawy, "Performance and Architectural Features of Segmented Multiple Bus System," *Proc. Int'l Conf. Parallel Processing,* pp. 154-160, 1999.

[16] S.T. Kamath, "Running Weak Hypercube Algorithms on Multiple Bus Networks," MS Thesis, Louisiana State Univ., Baton Rouge, LA, 1996.

[17] P. Kulasinghe and A. EI-Amawy, "Optimal Realization of Sets of Interconnection Functions on Synchronous Multiple Bus Systems," *IEEE Trans. Computers,* vol. 45, pp. 964-969, 1996.

[18] Y. Lan, "Adaptive Fault-Tolerant Multicast in Hypercube Multiprocessors," *J. Parallel and Distributed Computing,* vol. 23, pp. 80-93, 1994.

[19] K.Y. Lee, G. Liu, and H.F. Jordan, "TDM Hypercube and TWDM Mesh Optical Interconnections," *J. Parallel and Distributed Computing,* vol. 60, pp. 320-333, 2000.

[20] T.C. Lee and J.P. Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers," *IEEE Trans. Computers,* vol. 41, no. 10, pp. 1242-1256, Oct. 1992.

[21] Y. Leu and S. Kuo, "A Fault-Tolerant Tree Communication Scheme for Hypercube Systems," *IEEE Trans. Computers,* vol. 45, no. 6, pp. 641-650, June 1996.

[22] A. Louri and C. Neocleous, "A Spanning Bus Connected Hypercube: A New Scalable Optical Interconnection Network for Multiprocessors and Massively Parallel Systems," *J. Lightwave Technology,* vol. 15, no. 7, pp. 1241-1252, July 1997.

[23] A. Louri, B. Weech, and C. Neocleous, "A Spanning Multichannel Linked Hypercube: A Gradually Scalable Optical Interconnection Network for Massively Parallel Computing," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 5, pp. 497-512, May 1998.

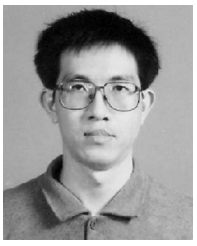
[24] H.M. Alnuwieri, M. Alimuddin, and H. Aljunaidi, "Switch Models and Reconfigurable Networks: Tutorial and Partial Survey," *Proc. First Workshop Reconfigurable Architectures,* 1994.

[25] R. Miller, V.K. Prasanna-Kumar, D. Reisis, and Q. Stout, "Parallel Computations on Reconfigurable Meshes," *IEEE Trans. Computers,* vol. 42, pp. 678-692, June 1993.

[26] F.P. Preparata and J. Vullemin, "The Cube Connected Cycles: A Versatite Network for Parallel Computation," *Comm. ACM,* vol. 24, pp. 300-309, May 1981.
[27] Y. Saad and M.H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Computers,* vol. 37, no. 7, pp. 867-872, July 1988.
[28] M.J. Serrano and B. Parhami, "Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 10, pp. 1073-1080, Oct. 1993.
[29] J.L. Trahan, R. Vaidyanathan, and R.K. Thiruchelvan, "On the Power of Segmenting and Fusing Buses," *J. Parallel and Distributed Computing,* vol. 34, pp. 82-94, 1996.
[30] N.F. Tzeng and S. Wei, "Enhance Hypercubes," *IEEE Trans. Computers,* vol. 40, pp. 284-294, 1991.
[31] R. Vaidyanathan, "Sorting on Prams with Reconfiguable Buses," *Information Processing Letters,* vol. 42, pp. 203-208, June 1992.
[32] R. Vaidyanathan and A. Padmanabhan, "Bus-Based Networks for Fan-In and Uniform Hypercube Algorithms," *Parallel Computing,* vol. 21, pp. 1807-1821, 1995.
[33] B. Wilkinson, "Multiple Bus with Overlapping Connectivity," *IEE Proc. -E,* vol. 138, no. 4, pp. 281-284, July 1991.
[34] B. Wilkinson, "On Crossbar Switch and Multiple Bus Interconnection Networks with Overlapping Connectivity," *IEEE Trans. Computers,* vol. 41, no. 6, pp. 738-746, June 1992.
[35] L.D. Wittie, "Communication Structures for Large Networks of Microcomputers," *IEEE Trans. Computers,* vol. 30, no. 4, pp. 264-273, Apr. 1981.

Lee-Juan Fan received the BS and MS degrees from the Department of Applied Mathematics at National Sun Yat-sen University, Kaohsiung, Taiwan, in 1995 and 1997, respectively. Since 1997, she has worked in the Information Technology Division of a famous semiconductor manufacturing company in Hsin-Chu, Taiwan. She dedicates herself to network security and factory automation systems.

Chang-Biau Yang received the BS degree in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982, and the MS degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1984. Then, he received the PhD degree in computer science from National Tsing Hua University in 1988. He is currently a professor in the Department of Computer Science and Engineering, National Sun Yat-sen University. His research interests include computer algorithms, interconnection networks, and bioinformatics.



Shyue-Horng Shiau received the BS degree from the Department of Engineering Science at National Cheng Kung University, Tainan, Taiwan, in 1984, and the MS degree from the Department of Applied Mathematics at National Sun Yat-sen University, Kaohsiung, Taiwan, in 1994. He is currently a PhD candidate in the Department of Computer Science and Engineering at National Sun Yat-sen University. He joined the faculty of the Department of Computer Aided Media Design, Chung Jung University, Tainan, Taiwan, as a lecturer in 1999. His research interests include algorithms and parallel processing.