TCN: Scalable Hierarchical Hypercubes

Trong-Yen Lee
Department of Electronic Engineering
National Taipei University of Technology, Taipei, Taiwan, R.O.C.
tylee@ntut.edu.tw

Pao-Ann Hsiung
Department of Computer Science and Information Engineering
National Chung Cheng University, Chia-yi, Taiwan, R.O.C.
hpa@computer.org

Sao-Jie Chen
Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan R.O.C.
csj@cc.ee.ntu.edu.tw

Abstract

Hierarchical hypercubes, such as Extended Hypercube (EH), Hyperweave (HW), and Extended Hypercube with Cross Connections (EHC), have been proposed to overcome the scalability limitation of conventional hypercubes through the use of fixed dimension hypercubes of processing elements (PEs) as basic modules interconnected by network controllers (NC) which are themselves interconnected into hypercubes. The scalability of all these three hierarchical hypercube networks is still limited, because the average network communication load in each NC increases as the number of interconnected PEs become very large. In this work, a generalization scheme is proposed for improving network scalability, namely Transformer Cube Network (TCN). For illustration purpose, generalized TCN is presented only for EH, though the same scheme can be applied to HW and EHC as well. Several characteristics of TCN, such as topological properties, message routing complexity, fault tolerance, and scalability are analyzed. We present a communication algorithm for one-to-one message passing in a fault-free case. Further, the application of TCN to a class of divide-and-conquer problems is shown to have a time complexity of $O(\log_2 N)$, where N is the total number of PEs.

1. Introduction

In recent years, the progress in VLSI technology has made feasible the fabrication of massively parallel computers. Factors that affect the performance of a parallel system include the topology of interconnection among processors, its scalability, and its fitting to the algorithms to be executed on the system. The hypercube topology proposed by Squire and Palais in 1963 [1] has been proved to be a very powerful topology, in which many other topologies, such as rings, trees, and meshes can be embedded. Other topologies revolving around the basic hypercube have also been proposed [2]-[4]; they are hypercubes with a slight generalization or modification, each attempting to improve some of the hypercube properties. Of course, these improvements have to be achieved at additional costs.

However, when hypercubes are used in large systems, there exists some practical limitations. For example, since the node degree reflects the number of I/O ports required per node, the fact that the hypercube node degree grows with its dimension size is difficult for VLSI implementation. This is the most serious drawback of hypercube and it is considered the main limiting factor for the use of hypercube in large systems [5]. In summary, the node degree should be kept as a constant, as small as possible, because a small constant node-degree is very much desirable to achieve modularity in building basic modules for scalable systems [6]. Therefore, several new multiprocessor architectures have been proposed in the literature [7]-[17] to increase computing speed and to complement the advances in VLSI design. Such related networks will be discussed in Section 2.

Among the class of interconnection networks proposed for overcoming the hypercube limitation of increasing node degree, hierarchical hypercubes including *Extended Hypercube* (EH), *Hyperweave* (HW), and *Extended Hypercube with Cross Connections* (EHC) are the most promising ones because of their high scalabilities and fault-tolerance capabilities. Though it appears that these networks have high scalabilities, yet there is an inherent structural limitation that would inhibit total scalability when the number of processors becomes very large. The main observation is that the *network communication load* increases with the total number of PEs. This is partly due to the fact that the number of PEs increases at a much faster rate than the number of NCs in a system. Increasing network communication load implies an increased message traffic at each NC, causing a slow-down in message routing. This degrading factor will ultimately be a bottleneck in the overall network throughput. At this stage, the network will no longer be scalable.

To achieve higher scalability, we propose a generalization scheme for hierarchical hypercubes. The concept is relatively simple. Instead of letting each NC be connected to a fixed number 2^r of PEs, where r is the dimension of a basic hypercube module, each NC is now allowed to connect to 2^i PEs, where i is an integer from 1 to r. In fact, we obtain a class of interconnection networks, called *Transformer Cube Network* (TCN). As the total number of PEs increases in a massively parallel computer system, we can increase the number of NCs to decrease the number of PEs connected to each NC. In such a way, the network communication load is decreased and thus greater scalability can be achieved.

Besides greater scalability, TCN has other advantages also. When an EH is generalized using our TCN scheme, a greater bisection-width, thus a greater fault-tolerance capability, is achieved through the increased number of links in a TCN. The NC degree is also reduced because we have more NCs now in a TCN.

This paper is organized as follows. Hypercube related networks are briefly described in Section 2. The method of constructing TCNs will be discussed in Section 3. Some important topological properties of the proposed TCN are explored in Section 4. Message routing algorithm for fault-free-NC networks is proposed in Section 5. In Section 6, we show how TCN can solve of a class of divide-and-conquer problems as in a hypercube. Section 7 concludes the paper with some additional remarks.

2. Previous Work

Hwang and Ghosh proposed a class of modular networks called *Hypernets* [9], which integrate the positive features of a hypercube and a tree-based topologies, and maintain a constant node degree when the network size increases. Thus, hypernets are more efficient than hypercubes for very large system sizes. Recently, Malluhi and Bayoumi proposed a *Hierarchical HyperCube* (HHC) [8] that combines hypercube and *Cube-Connected Cycles* (CCC) [5] in a hierarchical form. The attraction of this interconnection network emerges from the fact that it still retains a good performance when it is implemented with thousands of processors. The *Hierarchical Cubic Network* (HCN), presented by Ghose and Desai in [14], is a cube-based static interconnection network which provides the same performance as a comparable hypercube, but with only half as many links per node. In recent years, a new class of *Hierarchical Hypercubes with Network Controllers* (HHNC) has been proposed [11]-[13]. This class of networks uses r-dimensional hypercubes as its basic modules and hierarchically builds up another level of hypercube with these basic modules. The nodes at the lowest level, level 0, are processing elements (PEs) and the nodes at level 1 to level l are network controllers (NCs). These networks have the advantage of constant degree of nodes, low diameter, and scalable hypercube-based basic modules. This class of networks includes *Extended Hypercube* (EH) [11], *Hyperweave* (HW) [12], and *Extended Hypercube with Cross Connections* (EHC) [13] interconnection networks.

The *Extended Hypercube* (EH) [11], proposed by Kumar and Patnaik, is recursively built up with basic modules which are interconnected as r-dimensional hypercubes. The 2^r PEs within each building block of basic modules are all connected to an NC. However, EH has a poor fault-tolerance for communication from one level to another, because all the nodes of each r-cube rely on only one NC for their interaction with NCs at higher levels. Thus, HW and EHC networks focus on improving the fault-tolerance capability of EH. The *Hyperweave* (HW), being an improvement over the EH topology, has a better node fault-tolerance, because it has a greater number of disjoint paths between any two nodes, and a larger bisection width than EH. Later, a combination of the EH and HW networks, the *Extended Hypercube with Cross Connections* (EHC) [13], was proposed to overcome the poor fault-tolerance characteristics of the EH architectures. In these networks, an NC node always connects to a fixed number of 2^r PEs, where r is the dimension of a basic hypercube module.

In this paper, we propose a new generalized interconnection scheme called the *Transformer Cube Network* (TCN), which can be applied to all the above three hierarchical hypercubes. For a clearer illustration, we will describe the application of TCN to EH in details. Briefly, in a TCN, PEs are always interconnected into basic modules. Each module TCN(r, i, 1) has an r-cube of processing elements (PEs) and 2^{r-i} network controllers (NCs) as shown in Figure 1. The PEs in a basic module are responsible for computation and the NCs for interblock communications, such that if any one of the NCs is faulty, the r-cube processing elements can still communicate through the other 2^{r-i}-1 NCs. When the system is scaled up, the messages can be distributed through the 2^{r-i} NCs connected to other basic modules. Just like an EH, when a TCN is scaled up, its structure grows hierarchically with the number of basic modules, but the degree of PE nodes in each basic module is kept constant; this constant node-degree is necessary for scalable systems. Different

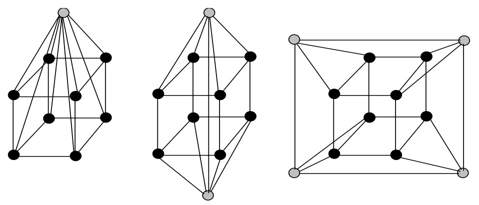

Figure 1 Interconnections of TCN(3, *i*, 1) where ○ represent *NC* nodes and ● represent *PE* nodes.

from an EH, TCN has an average message traffic density and NC communication load lower than its counterpart EH structure.

3. Transformer Cube Network

The proposed cube-based network, called *Transformer Cube Network* (TCN), is a hierarchical hypercube network that uses hypercubes as its basic modules. We use TCN(r, i, l) to denote a TCN network where r is the dimension of a basic module, i is the dimension of a sub-cube of PE or NC connected to a single NC, and l is the level of hierarchy.

A formal method for constructing TCN is as follows. A basic single-level TCN is denoted as TCN(r, i, 1) which has an r-dimensional hypercube of PE nodes at level 0 and an (r–i)-dimensional hypercube of NC nodes at level 1. An NC node at level 1 connects to 2^i PE nodes at level 0. For example, a TCN(3, 2, 2) is shown in Figure 2. In general, a TCN(r, i, l) has (l+1) levels, such that at level k, there are $2^{i(l-k-1)}$ r-dimensional hypercubes, where $0 \le k \le l-1$ and at level l there is an (r–i)-dimension hypercube. For an r-dimensional hypercube at level k, 2^i nodes are connected to a node at level (k+1).

A TCN(r, i, l) can be modeled by an undirected graph $G(V, E)$, where the set of vertices V represents $P \cup Q$, P is the set of PEs, and Q is the set of NCs. The nodes are labeled as follows:

$P = \{X_l X_{l-1} \ldots X_j \ldots X_0 \mid 0 \le X_l < 2^{r-i}, 0 \le X_j < 2^i, 0 \le j \le l-1\}$

$Q = \{X_l X_{l-1} \ldots X_j \mid 0 \le X_l < 2^{r-i}, 0 \le X_j < 2^i, 1 \le j \le l-1\}$.

Alternatively, the binary form for labeling a node is denoted as follows:

Binary($X_l X_{l-1} \ldots X_j \ldots X_0$) = bits$_{r-i}(X_l)$ bits$_i(X_{l-1})$ ... bits$_i(X_0)$,

where bits$_{r-i}(X_l)$ is the binary representation X_l in r–i bits and bits$_i(X_j)$ is the binary representation of X_j in i bits, $0 \le j \le l-1$. Processing elements (PEs) and network controllers (NCs) of the network plus the set of edges (links) E, respectively serve as the computation and communication links of a network. The set of vertices V includes $2^{i\cdot(l-1)}$ r-cubes of processing elements at level 0 and $2^{r-i}(2^{i\cdot l}-1)/(2^i-1)$ network controllers at l levels (from level 1 to level l).

The set of edges E is the union of two sets $E_1 \cup E_2$. E_1 represents the links between PEs in each r-cube at level 0 or the links between NCs in each r-cube at level 1 to level l, these links are called *hypercube links*. E_2 represents the links connecting an NC node at a higher level to a set of NC nodes at the next lower level or to PE nodes at level 0, these links are called *tree links*. Let $E(X, X')$ represent a link between nodes X and X'. The union of sets E_1 and E_2 is defined as follows:

$E_1 = \{(X, X') \mid X$ and X' are neighboring nodes in the same hypercube$\}$

$E_2 = \{(X, X') \mid X = X_l X_{l-1} \ldots X_j \in Q, X' = X_l X_{l-1} \ldots X_j X_{j-1} \in P$ or Q, where $1 \le j \le l\}$

As an illustration, Figure 2 shows a TCN(3, 2, 2), which has four basic 3-cube modules of PE nodes at level 0, interconnected by 8 NC nodes at level 1 and 2 NC nodes at level 2.

4. Topological Properties of Transformer Cube Network

In this section, some important properties of TCN are analyzed. Using the basic properties, we will derive the average network communication load for each NC. We will then deduce the fact that the NC load in a TCN(r, i, l) becomes lower than that in an EH(r, l) as i is decreased, $i < r$.

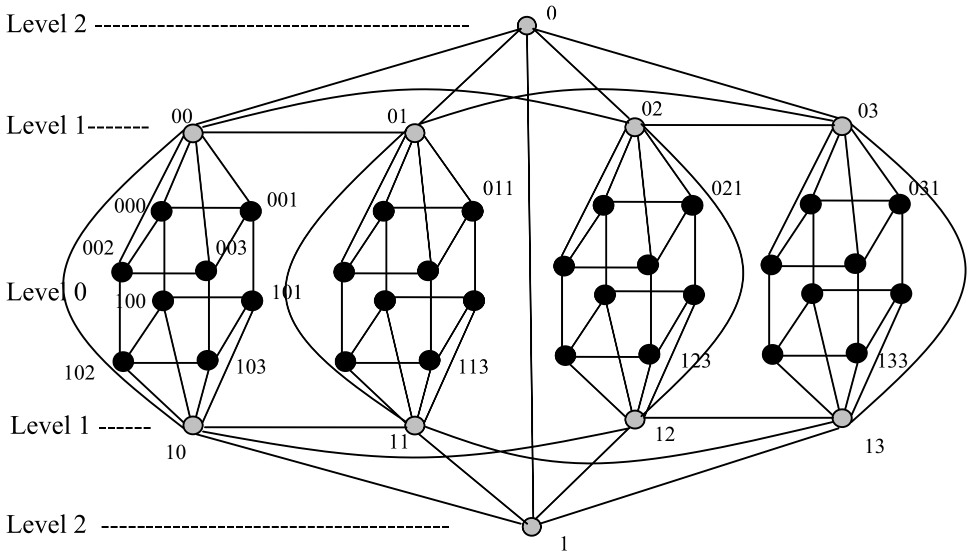

Figure 2 Interconnections in a TCN(3, 2, 2)

4.1. Network Diameter

The *diameter* of a network is defined as the maximum distance between any pair of processors. The *distance* between a pair of processors is the smallest number of links that have to be traversed from one processor to the other [18]. The diameter of a TCN(r, i, l) is given as:

$$D = i + 2l, \text{ where } l \geq 1. \tag{1}$$

4.2. Network Nodes and Link Complexity

A basic module TCN(r, i, 1) consists of 2^r PE nodes and 2^{r-i} NC nodes. If the parameters r and i of a basic module are decided, then the number of PE and NC nodes in a TCN(r, i, l) will grow with l only, a general expression is given as:

$$\text{Number of PE nodes} = N_{\text{PE}}(r,i,l) = 2^{r+i\cdot(l-1)}, \quad \text{where } l \geq 1,\ 1 \leq i \leq r. \tag{2}$$

$$\text{Number of NC nodes} = N_{\text{NC}}(r,i,l) = 2^{r-i} \cdot \frac{2^{l\cdot i}-1}{2^i-1}, \quad \text{where } l \geq 1,\ 1 \leq i \leq r. \tag{3}$$

The degree of an interconnection is defined as the maximum node-degree [18]. The PE and NC node-degrees in a TCN are respectively given as follows:

$$DG_{\text{PE}}(r,i,l) = r + 1, \tag{4}$$

$$DG_{\text{NC}}(r,i,l) = 2^i + r + 1, \tag{5}$$

where $l \geq 1$.

In a TCN network, since the degrees of PE and NC nodes are different, the average node degree, denoted as ADG [11] in Equation (6), will be used instead:

$$ADG = (N_{\text{PE}} \times DG_{\text{PE}} + N_{\text{NC}} \times DG_{\text{NC}})/(N_{\text{PE}} + N_{\text{NC}}). \tag{6}$$

The total number of links $L(r, i, l)$ in a TCN(r, i, l) is given by

$$L(r,i,l) = r \cdot 2^{r-1} \cdot 2^{i\cdot(l-1)} + (r-i)\cdot 2^{r-i-1} + 2^r \cdot \frac{2^{i\cdot l}-1}{2^i-1} + r \cdot 2^{r-1} \cdot \frac{2^{i\cdot(l-1)}-1}{2^i-1}, \text{ where } l \geq 1 \text{ and } 1 \leq i \leq r. \tag{7}$$

A multiprocessor system with a large diameter network has a very low message passing bandwidth. Further, a network with a high node degree is very expensive to build. A multiprocessor system is often evaluated by the system cost factor which is given by the product of diameter and network node-degree. Hence, the system cost factor of TCN, denoted as CF, is given by:

$$CF = D \times ADG \tag{8}$$

4.3. Average Distance and Message Traffic Density

One important measure of merit of a multiprocessor system is the average internode distance that messages must travel in the interconnection network. It is advantageous to make this parameter as low as possible, because it will not only reduce message traveling time, but also minimize message density in the link.

The average distance, denoted by P, is calculated as the sum of the distance of a particular node to nodes within the same basic module and to nodes in the other basic modules divided by N-1, as shown below:

$$P = \begin{cases} \left(\sum_{d=1}^{r} \binom{r}{d} \cdot d\right)/(N-1), & \text{for TCN}(r,i,1); \\ \left(\sum_{d=1}^{r} \binom{r}{d} \cdot d + \sum_{j=2}^{l} \sum_{d=2j-1}^{r+2(j-1)} q \cdot 2^{i\cdot(j-1)} \cdot d\right)/(N-1), & \\ & \text{for TCN}(r,i,l), l \geq 2. \end{cases} \tag{9}$$

The *average message traffic density*, denoted by ρ, in a TCN network is defined as:

$$\rho = P \cdot N_{\text{PE}} / L \tag{10}$$

where P is given in Equation (9), N_{PE} in Equation (2), and L in Equation (7).

4.4. Connectivity Analysis

The node (link) connectivity of a network is the minimum number of nodes (links) whose removal results in a disconnected network [19]. If the node (link) connectivity is k, then the network can tolerate the failure of at most (k – 1) nodes (links), while maintaining its connectivity. In a TCN, it is easy to observe that the PE node (or link) connectivity is r+1 and the NC node (or link) connectivity is ($2^i + r + 1$). As for the basic module connectivity (B_C), if the 2^{r-i} NCs in any basic module are faulty, then this basic module is disconnected from the other basic modules. Therefore, the basic module node connectivity is 2^{r-i}.

4.5. Network Communication Load

The average network communication load is the total message traffic at each NC node. This can be obtained by the product of the average message traffic density and the NC degree.

$$NC\_load = \rho \times D_{NC} = P \times N_{\text{PE}} \times D_{NC} / L \tag{11}$$

As we can see from Equation (11), *NC_load* is directly proportionate to the total number of PE nodes. Hence, on scaling up the system, a large N_{PE} would result in a high communication load.

4.6. Typical Values of TCN and Comparision with Other Networks

Some typical important properties of TCN are shown in Table 1. Comparisions with other neworks are shown in Table 2.

Table 1 Summary of Topology Properties for a TCN(*r*, *i*, *l*)

r	i	L	N_{PE}	N_{NC}	D_{NC}	L	P	ρ	*NC_load*	D	B_c
2	1	1	4	2	5	9	1.33	0.59	2.05	3	2
		2	8	6	5	25	2.57	0.82	4.10	5	2
		3	16	14	5	57	4.13	1.16	5.80	7	2
		4	32	30	5	121	5.87	1.55	7.75	9	2
		5	64	62	5	249	7.71	1.98	9.90	11	2
	2	1	4	1	7	8	1.33	0.67	4.69	4	1
		2	16	5	7	40	2.93	1.17	8.19	6	1
		3	64	21	7	168	4.76	1.81	12.67	8	1
		4	256	85	7	680	6.70	2.52	17.64	10	1
		5	1024	341	7	2728	8.68	3.26	22.82	12	1
3	1	1	8	4	6	24	1.71	0.57	3.42	4	4
		2	16	12	6	64	2.93	0.32	1.92	6	4
		3	32	28	6	144	4.52	1.00	6.00	8	4
		4	64	60	6	304	6.29	1.32	7.92	10	4
		5	128	124	6	624	8.16	1.67	10.02	12	4
	2	1	8	2	8	21	1.71	0.65	5.20	4	2
		2	32	10	8	101	3.35	1.06	8.48	6	2
		3	128	42	8	421	5.23	1.59	12.72	8	2
		4	512	170	8	1701	7.18	2.16	17.28	10	2
		5	2048	682	8	6821	9.17	2.75	22.00	12	2
	3	1	8	1	12	20	1.71	0.68	8.16	5	1
		2	64	9	12	180	3.47	1.23	14.76	7	1
		3	512	73	12	1460	5.44	1.91	22.92	9	1
		4	4096	585	12	11700	7.43	2.60	31.20	11	1
		5	32768	4681	12	93620	9.43	3.30	39.60	13	1

Table 2 Comparisions of TCN(*r*, *i*, *l*) with Other Networks

	TCN(r, i, l)	EH(r, l)	EHC(r, l)	HW(r, l)
Basic Module Connectivity	2^{r-i}	1	r+1	r
NC_load	Low	High	Middle	High

5. Message Routing in TCN

In this section, we will generalize fault-free message routing algorithm for TCN. Routing is defined as the passing of messages from a source node through a network path to a destination node.

Suppose that all nodes are available, the procedure for routing a message M from a source PE node S with label S_l S_{l-1} ... S_0 to a destination PE node D with label D_l D_{l-1} ... D_0 in a TCN(r, i, l) involves the following two cases.

Case 1: Source and destination PE nodes are located within the *same* basic module.

Case 2: The source node and the destination nodes are located in two *different* basic modules.

The routing algorithm is given in Algorithm 1. As an illustration, let us consider the topology of a TCN(3, 2, 2) in Figure 2. Assume that the source node has label “000” and the destination node has label “011”. The source node and destination node are located in difference basic modules. The value of $(r - i)$ is 1; therefore, the value of j is $(l-1)$, *i.e.* 2. So, the message is routed from the source PE node “000” to the NC node “00” through tree links. Using a hypercube self-routing algorithm, the message is routed from NC nodes “00” to “01”. Finally, the message is routed from the NC node “01” to to the destination node “011” through tree links.

Algorithm 1 Fault-Free Message Routing on a TCN(*r*, *i*, *l*)

```
procedure fault-free-node-to-node-routing ($S$, $D$, $M$)
{Route message $M$ from source $S$, labeled by $S_l\ S_{l-1}$ ... $S_0$, to
  destination $D$, labeled by $D_l\ D_{l-1}$ ... $D_0$ in TCN($r$, $i$, $l$)}
begin
  if $A = B$ then {Source and destination are the same node}
    Terminate;
  end;{if}
  if $S_l\ S_{l-1}\ S_{l-2}$ ... $S_1 = D_l\ D_{l-1}\ D_{l-2}$ ... $D_1$ && $(r - i) < 2$ then
    {Case 1: $S$ and $D$ are located within the same basic module}
    route message $M$ from $S$ to $D$ using a hypercube self-routing
    algorithm;
  else begin{Case 2: $S$ and $D$ are located in different basic
    modules}
    if $S_{l-1}\ S_{l-2}$ ... $S_{r-i} = D_{l-1}\ D_{l-2}$ ... $D_{r-i}$ && $(r - i) \geq 2$ then
      for $k = 0$ to $(r - i - 2)$ do
          if bit$_k(S_l) \neq$ bit$_k(D_l)$; break;{exit if $S$ and $D$ are in
          different basic modules}
          end;{if}
      end;{for}
      route message $M$ from $S$ to $D$ using a hypercube
      self-routing algorithm
    else begin
      if $(r - i) \geq 2$ then
        for $k = 0$ to $(r - i - 2)$ do
          if bit$_k(S_l) \neq$ bit$_k(D_l)$ then $j = l - 1$; break; end;{if}
        end;{for}
      end;{if}
      if $(r - i) < 2$ || $j \neq l - 1$ then
        for $= (l - 1)$ down to $(r - i - 2)$ do
          if $S_k \neq D_k$ then $j = k$; break; end;{if}
        end;{for}
      end;{if}
      shift message $M$ from NC node $S_l\ S_{l-1}$ ... $S_0$ to NC node
      $S_l\ S_{l-1}$ ... $S_j$ through tree links
      route message $M$ from NC node $S_l\ S_{l-1}$ ... $S_j$ to NC node
      $D_l\ D_{l-1}\ D_{l-2}$ ... $D_j$ using a hypercube self-routing
      algorithm
      shift message $M$ from NC node $D_l\ D_{l-1}\ D_{l-2}$ ... $D_j$ to PE
      node $D_l\ D_{l-1}\ D_{l-2}$ ... $D_1D_0$
    end;{begin}
    end;{if}
  end;{begin}
  end;{if}
end;{begin}
```

6. Embedding TCN for the Class of Divide-and-Conquer Algorithms

In this section, we demonstrate how the TCN topology can be employed to solve the divide-and-conquer (D&Q) class of problems. The three steps involved in a divide-and-conquer paradigm were described in [8] including divide, conquer, and combine.

Assume that the input U consists of $n = 2^k$ values U(0), U(1), ..., U(n-1). Each basic operation OP(U[m], U[m+2^m]) modifies the two data items presented in storage locations U[m] and U[m+2^m]; the computation performed affects only the contents of locations U[m] and U[m+2^m]. The class of algorithms described above is similar to the ascent class [2]. In the following, we explain how TCN can be used to execute the parallel_D&Q algorithm.

Suppose we have 32 data to store in the PEs of a TCN(3, 2, 2) and each PE holds one data. For example, input data U(0), U(1), ..., U(7) are stored in 000, 001, ..., 103 PE nodes of the left-most basic module, and others data are stored similarly in PE nodes of other basic modules. Applying the D&Q algorithm to the TCN(r, i, l), for $0 \leq j < r$, each operation OP(U[m], U[$m+2^m$]) is separated by a single link of the r-dimensional hypercube, whereas for $r \leq j < q$, each operation OP(U[m], U[$m+2^m$]) is separated by external links (tree links) and links passing through NC nodes. The implementation of the D&Q on a TCN(r, i, l) is shown in Algorithm 2. The D&Q algorithm can be implemented in $O(\log_2 N)$ steps.

Algorithm 2 Divide-and-Conquer Algorithm in a TCN(r, i, l)

```
procedure TCN_D&Q(U)
  for $j$ = 0 to $q$-1 do {where $q = r + i(l-1)$; $r$, $i$ and $l$ represent the
                          dimension of a basic module, the dimension
                          of PEs or NCs connected by a NC, and the
                          network level, respectively.}
    for each $m$| $0 \leq m < N$ do
      cobegin
        if bit$_j$ ($m$) = 0 then
          OP(U[$m$], U[$m+2^m$]);
      coend
end {end of TCN_D&Q}
```

7. Conclusion

We have presented a scheme called the *Transformer Cube Network* (TCN) to generalize hierarchical hypercube networks, such as *Extended Hypercube* (EH), *Hyperweave* (HW), and *Extended Hypercube with Cross Connections* (EHC). Just like its predecessors, TCN can be used to interconnect massively parallel systems but owns a greater scalability, because the number of processing elements connected to a single network controller in TCN is flexible.

Other attractive properties of TCN are its higher bisection width and its greater fault-tolerance capability. We have shown that the cost factor of a TCN, given by (degree of node × diameter), is less than that of a hypercube. We have also presented one-to-one message routing algorithm dealing with fault-free-NC. Lastly, the paper described how TCN can be efficiently used to execute a large class of algorithms, the divide-and-conquer (D&Q) class. These versatile properties of TCN make it an attractive candidate for practical multiprocessor and distributed computing systems.

References

[1] S. Sqire and S. M. Palais, "Programming and design consideration for a highly parallel computer," in *Proc. AFIP Spring Joint Comp. Conf.*, vol. 23, pp. 395-400, 1963.

[2] E1-Amawy and S. Latifi, "Properties and performance of folded hypercubes," *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 31-42, January 1991.

[3] H. P. Katseff, "Incomplete hypercubes," *IEEE Trans. on Computers*, vol. C-37, no. 5, pp. 604-608, April 1988.

[4] L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. on Computers*, vol. C-33, no. 4, pp. 323-333, April 1984.

[5] F. P. Preparata and J. E. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, no. 5, pp. 300-309, May 1981.

[6] K. Hwang, *Advanced Computer Architecture*: *Parallelism, Scalability, Programmability*, McGraw-Hill Inc. 1993.

[7] E. Ganesan and D. K. Pradhan, "The hyper-deBruijn networks: Scalable versatile architecture," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 9, pp. 962-978, September 1993.

[8] Q. M. Malluhi and M. A. Bayoumi, "The hierarchical hypercube: A new interconnection topology for massively parallel systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 17-30, January 1994.

[9] K. Hwang and J. Ghosh, "Hypernet: A communication-efficient architecture for constructing massively parallel computers," *IEEE Trans. on Computers*, vol. C-36, no. 12, pp. 1450-1466, December 1987.

[10] S. Lakshmivarahan and S. K. Dhall, "A new hierarchy of hypercube interconnection schemes for parallel computers," *Journal of Supercomputing,* vol. 2, pp. 81-108, 1988.

[11] J. M. Kumar and L. M. Patnaik, "Extended hypercube: A hierarchical interconnection network of hypercubes," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 1, pp. 45-57, January 1992.

[12] G. Rarnanathan, M. Clement and P. Crandall, "Hyperweave: A fault-tolerant expandable interconnection network," in *IEEE Symposium on Parallel and Distributed Processing*, Dallas, USA, pp. 479-482, Dec. 1992.

[13] J. M. Kumar, "Fault-tolerant hierarchical network of hypercubes," in *Proc. of the First International Workshop on Parallel Processing*. Bangalore, India, pp. 29-34, Dec. 1994.

[14] K. Ghose and K. R. Desai, "Hierarchical cubic networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 6, no. 4, pp. 427-435, April 1995.

[15] S. P. Dandamudi and D. L. Eager, "Hierarchical interconnection networks for multicomputer systems," *IEEE Trans. on Computers*, vol. 39, no. 6, pp. 786-797, June 1990.

[16] S. P. Dandamudi and D. L. Eager, "On hypercube-based hierarchical interconnection network design," *Journal of Parallel and Distributed Computing*, vol.12, pp. 283-289, 1991.

[17] M. A. Aboelaze, "MLH: A hierarchical hypercube network," Networks, vol. 28, pp. 157-165, 1996.

[18] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, SanMateo, Calif., 1992.

[19] D. Z. Du and F. K. Hwang, "Generalized deBruijn digraphs," *Networks*, vol. 18, no. 1, pp. 27-38, 1988.